Fast inference in flexible high dimensional models of gene co-expression and genotype-gene expression interaction

Dissertation
for the award of the degree
"Doctor of Philosophy"
Division of Mathematics and Natural Sciences
of the Georg-August-Universität Göttingen

within the doctoral program Genome Science
of the Georg-August University School of Science (GAUSS)

submitted by

Étienne Morice

from Massy, France

Göttingen 2024

Thesis Advisory Committee:

Dr. Johannes Söding, Research Group Quantitative and Computational Biology,
Max Planck Institute for Multidisciplinary Sciences

Prof. Dr. Heike Bickeböller, Department of Genetic Epidemiology,
University Medical Center Göttingen

Prof. Dr. Thomas Kneib, Chair of Statistics,
Georg-August-Universität Göttingen

Members of the Examination Board:

Referee: Dr. Johannes Söding,
Research Group Quantitative and Computational Biology,
Max Planck Institute for Multidisciplinary Sciences

2nd Referee: Prof. Dr. Heike Bickeböller, Department of Genetic Epidemiology,
University Medical Center Göttingen

Further members of the Examination Board:

Prof. Dr. Thomas Kneib, Chair of Statistics,
Georg-August-Universität Göttingen

Jun.-Prof. Dr. Anne-Christin Hauschild, Department of Medical Informatics,
University Medical Center Göttingen

Prof. Dr. Michael Altenbuchinger, Department of Medical Bioinformatics,
University Medical Center Göttingen

Dr. Juliane Liepe, Research Group Quantitative and Systems Biology,
Max Planck Institute for Multidisciplinary Sciences

Date of oral examination: November 28, 2024

Abstract

As more high-throughput sequencing data is produced from ever-growing human cohorts, understanding human biology becomes a more and more quantitative and statistical task. Starting notably with the data made available in 2020 by the Genotype-Gene Expression project, we undertake a systematic exploration of models to describe gene co-expression patterns, with a new emphasis on probabilistic modeling and predictive out-of-sample performance. We especially generalize some Bayesian linear models and the associated optimization methods to achieve scalability on thousands of genes and flexibility in the model by careful linear algebra manipulations. We then build on these foundation models and demonstrate their downstream usefulness by tackling their integration into state of the art expression quantitative trait loci discovery pipelines. In the process, we re-analyze and improve the control and multiple testing procedures that this objective requires. We eventually demonstrate how better co-expression modeling translates to increased genotype association discovery power, and overall aim to set an example of deep probabilistic integration of omics data in modern bioinformatics pipelines.

Contents

Key:

- X† Division presenting background material with little to no original contribution but whose understanding is a prerequisite to exposing our original work.
- X# Division likewise exposing background material but presented or derived in a novel way. For details see page 167.
- X** Division presenting, to our knowledge, entirely original material.

Introduction

The last decades have seen the development of high-throughput measurement methods in biology. We now have the ability to obtain, at a reasonable cost, the complete genome sequence, along with the quantification of the abundance of transcripts, proteins, metabolites in many biological samples. All fields of science have historically evolved along with better measurement techniques; but often this has been in the sense of higher precision or resolution of the instruments: microscopes that can see smaller, telescopes that can see farther. Modern biology has the somewhat more exceptional characteristic that we haven't been merely measuring the same thing than before with increased precision; we're also and mostly measuring a huge amount of different variables — effectively, the relative concentrations of a myriad of distinct chemical species, be they nucleic, amino acids or small ligands — at the same time.

Often, the approximate completion of the sequencing of the first human genome is cited as a turning point of genetics at the turn of the century [Lan+01]. However, the human genome project was more of an unavoidable intermediary, long after researchers had started to measure variation in genomes and long before they would be able to measure genomic and transcriptomic variation at the scale necessary for significant scientific discoveries. And above all, the human genome project was statistically just one sample, with not much to feed the quantitative appetite of statical geneticists. But in parallel of the first sequencing race, an other genetic revolution was slightly more quietly unfolding. If we need a starting point of modern statistical genetics, it could rather be in the early 90s when high density synthesis of peptides on glass slides was developed [Fod+91], which would lead to the first commercial genotyping microarray in 1994. Around the same date appeared the first complementary DNA microarray for quantitative gene expression measurement [Sch+95]. Coincidentally, this turns out to be not long before the author of this thesis was born; so the scene for the present work was set in more ways than one in the final years of the 20th century. At the time, the first glass slides had been printed to allow simultaneous measurement of 45 Arabidopsis transcripts, a modest beginning in the ever increasing multiplexing in transcriptomics. Perhaps nothing tells better the enthusiasm and hope that the new technology spurred, and the care that is needed in interpreting the output of such technology, than the tragic history of the microarray co-inventor, Mark Schena. Following the success of his work, he went on to create his own company to market and produce microarrays, until about one year ago, when he was sentenced to eight years in prison when his promotion of microarray-based allergy and COVID-19 tests that did not actually work as advertised was judged a fraud [AP23]. But if the most extraordinary promises of simultaneous genetic measurements failed to materialize, the revolution did go on. While the robotic microarray construction was limited in the density of measurement units that could be achieved, the optical synthesis control methods developed for genotyping arrays proved their versatility in expanding to gene expression quantification. Above all, they also proved their scalability to an ever increasing number of markers, ensuring along the way the success of the Affymetrix company which was to sat at the center of the microarray scene and in the method sections of virtually all breakthroughs of the time. The company's arrays, for instance, were used in early human gene expression studies of the Framingham heart study, then limited to a few dozens individuals but already including 18,000 genes [Joe+12]. Technology kept evolving. RNA-Seq has displaced microarrays as the dominant way of quantifying gene expression at the genomic scale [Nag+08]. With every base pair being potentially quantified individually, only the aggregation strategy now defines how many variables we effectively measure. The data that we will focus on in this thesis would expand to the measurement of around 50,000 genes, that is as many genes as found in the human genome under a very broad definition of "gene".

In parallel, geneticists have been interested in measuring the difference of genomic sequence between individuals, here again a large number of variables, binary this time, representing the presence or absence of a certain DNA sequence or genetic marker. Historically, maps of such markers could be obtained through polymerase chain reaction (PCR) amplification of targeted genome sequences, typically simple sequence repeats whose expansion would yield products of noticeably different molecular weight after amplification, requiring a separate PCR and gel electrophoresis for each primer to be tested and each individual. Due to the workload implied, this was typically limited to animals, where the genotyping work could be leveraged by typing once a given animal line, and then re-using the breed and its known genotype in numerous experiments [TNP94]. Such lines of mice and rats enabled quantitative trait-loci (QTL) analyses by measuring phenotypes in the genotyped lines [Pra+95]. Soon the need for a mechanistic explanation of the effect of identified QTLs led researchers to turn their attention to gene expression as phenotype, with dot blot RNA assays used as a quantitative measurement of gene expression [Dum+00]. The need for individual measurements of markers and RNAs limited the scope of the studies, which could have a few hundreds genetic markers, in one or two dozen animal lines, with a handful of genes quantified for expression. The tremendous amount of work required, especially for genotyping, precluded systematic study of even small human samples. Microarrays here again were a game changer. A few years after the microarray technology emerged, researchers started to exploit the potential to also measure genetic variation [Win+98]. Before long, this led to gene expression eQTL studies in e.g. yeast with several thousands genes and markers measured simultaneously with a few microarrays [Bre+02]. Similar studies on humans soon followed, with for instance around 8500 genes measured in around 100 individuals, which constitute the prototype of the modern human eQTL studies that we will focus on [Mor+04]. Given the lack of need for accurate quantification of the underlying molecules when genotyping — we care for the presence or absence of a nucleic acid sequence, not for how much of it is present in the sample — and the size of the sequence of interest, that is the whole genome, microarrays stayed the dominant technology for genotyping longer than for gene expression quantification. The Framingham heart study, for instance, used genotype arrays to measure around 100,000 markers, in one or two thousands individuals [Her+06][Cup+07]. The UK Biobank, notably, genotyped its 500,000 participants with large microarrays targeting around 800,000 markers [Byc+18]. Over time, the decreasing cost of whole genome sequencing (WGS) and the appeal of the more complete information it could provide would lead to direct sequencing of an individual's DNA to also become the preferred way of genotyping. The UK Biobank, still, progressively released whole genome sequencing data based on Illumina technology from subsets of its participants [Hal+22], with the effort to sequence the complete cohort just recently announced to be finished [Li+23].

For the purpose of this thesis, the project that interests us and sits at the convergence of these developments is the Genotype-Tissue Expression (GTEx) project [Lon+13]. After a pilot phase to establish feasibility, the design targeted the enrollment of 900 postmortem donors, from whom tissue samples were to be collected and used for genotyping and gene expression quantification. Remarkably, the original plans illustrate well the technology shifts of the era. Gene expression was already meant to be measured with RNA-Seq, with microarrays also used to provide comparison between technologies. Meanwhile, genotyping was initially performed with bead-based microarrays, but with the understanding that the individuals were to eventually be fully sequenced. Over the decade that followed, enrollment, collection, sequencing were steadily carried out, resulting in a sequence of data releases: a pilot data release in 2015 [GTE+15], a "v6" partial release in 2017 that already established most of the analysis pipeline [Agu+17], and a "v8" version which corresponds more or less to the completion of the initial plan [GTE20]. This last release from 2020, right before the start of this thesis, is where most of the data analysed herein is sourced from. The GTEx project, however, carried on, with more data modalities released in later versions, such as single-nucleus RNA-Seq in the "v9" [Era+22] — but in a very limited number of samples which makes this more recent data less relevant for our purposes. Most recently, the project also evolved with the extension to sister projects such has the "developmental" GTEx targeting younger individuals. The RNA-Seq data from the eighth release, meanwhile, remains the most comprehensive gene-expression resource at our disposal. A notable comparable resource comes from the RNA-Seq data from around 900 individuals from the Depression Genes and Networks (DGN) cohort [Bat+14], contemporary with the GTEx project and similar in cohort size, though limited to blood. Larger eQTL analyses have been performed, but are typically the result of the integration of several smaller cohorts, as was notably done in the eQTLGen consortium which totaled more than 30,000 individuals across many cohorts and mixing RNA-Seq and microarray data, including notably the aforementioned Framingham, GTEx and DGN cohorts [Võs+21].

Although no proteomics data has been analyzed in this work, direct quantification of protein abundances rather than RNA has also further enriched the field of quantitative gene expression analysis over time. Traditionally, mass-spectrometry based methods have been used, e.g. early on in yeast to measure more than 500 protein abundances [Fos+07]. More recently, methods based on proximity immunoassays, using two complementary antibodies binding to the same protein of interest to produce a DNA fragment that can then be detected, have gained much traction. Originally used for the detection of protein complexes [Söd+06], the methods have been extended and marketed as a technique for single protein quantification with high potential for multiplexing [Wei+10][Dar+11]. The approach has seduced a number of biobank investigators, leading to a natural extension of eQTL methods to protein QTL (pQTL) studies. An example on the UK Biobank used the proximity extension assay to multiplex the measurement of close to 3,000 protein abundances in plasma from around 50,000 individuals [Sun+23]. These numbers illustrate reduced multiplexing ability compared to RNA-Seq measurements, since protein-specific antibodies are required for each target, but also the remarkable scalability potential. We expect that most of the statistical methods and finding reported in this work for transcriptomics have the potential to transfer readily to proteomics as data sources become available, but we have not investigated this claim.

With these new developments in how we measure the biology of humans, methods for data analysis should evolve too. The ability to obtain so many variables at once gives the opportunity to change how we analyze any subset of these variables. In a study of a single phenotype, we may have known nothing else of each individual that what has been explicitly measured, such as sex and age: a handful at most of covariates. With microarrays and RNA-Seq, we have dozens of thousands of new measurements that paint a extremely rich molecular picture of each individual in a cohort. Likewise, we never measure a genotype alone: arrays and WGS also give us millions of markers from which an extremely detailed ancestry and resolution for each locus can be obtained. In short, we now have *context* for every association that we may be interested in.

We focus a lot on eQTL studies, that is the study of associations between genotype and quantitative gene expression in this work. This is less because we see eQTL studies as an ultimate goal, and more because they are one of the immediate target for new methods that aim to exploit fully the richness of the measurements that we have. The naive way of measuring a genotype-gene expression association would be to discard all of the other information, and reduce our data to a two-variable series of one marker and one gene. This, as the eQTL community knows well, would be as inefficient as it is misleading. The observed association would be biased by population structure, meaning changes in expression that vary with ancestry but are not causally associated with any particular locus. At the same time, the expression would be affected by innumerable unmeasured confounders that vary from individual to individual: cell type composition varying between biological samples, time between death and measurement varying from one sample to the next, nutrition, medication, intoxicant consumption, cause of death, phases of biological cycles, immune activity. . . The detailed genotype picture gives us a way to disentangle the first effects; the numerous gene expression measurements are a path to tackle the others. All the effects that we've cited are typically unmeasured, but have one thing in common: they are expected to broadly affect many genetic markers. The breadth of the data we can access today is a chance to describe better than ever the complex background of genetic variation and association on top of which every single precise association occurs. Without models that capture this background, little can actually be extracted of any observational study. But when it comes to humans, observational studies of genetics is all that we will ever get when it comes to complete individuals and tissues in natural environments. Capturing that background of unspecific associations will therefore not cease to be relevant in the foreseeable future.

The application of the models we develop, though we demonstrate it on eQTLs, may ultimately rather be in even larger pipelines, including even more data. The next step is to also combine phenotypes from genome-wide association studies. This may be done through variant interpretation methods [Ger+21], this may be done through predictive gene expression models in transcriptome-wide analysis [Gam+15][Gus+16]. In all cases, the first bricks of modern pipelines towards more integrated methods to exploit ever-growing and heterogeneous data sets need a starting point, and we propose to begin here with gene expression and the first step of integration with genotypes.

Notations

We use different case and fonts to distinguish scalars, vectors, and matrices; as well as random from deterministic variables, when possible.

- Deterministic scalars use lowercase italics: x, σ
- Random scalars use uppercase italics for Latin names: X, σ
- Deterministic vectors use lowercase bold, roman for Latin names: $\mathbf{x}, \boldsymbol{\sigma}$
- Random vectors use lowercase italics: $\boldsymbol{x}, \boldsymbol{\sigma}$
- Deterministic matrices use uppercase bold, roman for Latin names: $\mathbf{X}, \boldsymbol{\Sigma}$
- Random matrices use uppercase bold italics: $\boldsymbol{X}, \boldsymbol{\Sigma}$

The field of real numbers is called $\mathbb{R}$. We write the space of real matrices with n rows and p columns as $\mathbb{R}^{n \times p}$. In contrast, np-dimensional vectors are elements of $\mathbb{R}^{np}$ though of course the two are trivially isomorphic.

The transpose of matrix $\mathbf{A}$ is written $\mathbf{A}'$. Vectors are identified to matrices with one column for the purpose of matrix multiplication and transpose. Therefore the inner product of vectors $\mathbf{a} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$ is $\mathbf{a}'\mathbf{b} \in \mathbb{R}$, where $\mathbf{a}'$ is seen as a matrix with one row or row vector, and their outer product is $\mathbf{a}\mathbf{b}' \in \mathbb{R}^{n \times n}$.

When indexing a vector or matrix, we use the case and font corresponding to the result of the operation. Therefore, the elements of the vector $\mathbf{x} \in \mathbb{R}^n$ are $x_1, \dots, x_n$. Similarly, the transposed rows of the matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ are the vectors $\mathbf{x}_1, \dots, \mathbf{x}_n$. The matrix $\mathbf{X}$ can therefore be written:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1' \\ \vdots \\ \mathbf{x}_n' \end{bmatrix} \tag{1}$$

We may also write its columns as the vectors $\mathbf{x}^1, \dots, \mathbf{x}^p$ to differentiate them from rows, as an exponent applied to a vector can't be mistaken for actual exponentiation. The individual coefficients of $\mathbf{X}$ are usually written x_{ij} for the row i and column j.

If $\mathbf{A} \in \mathbb{R}^{n \times n}$, we write $\mathrm{Tr}[\mathbf{A}] = \sum_{i=1}^{n} a_{ii}$ its trace and $|\mathbf{A}|$ its determinant. The identity matrix of $\mathbb{R}^{n \times n}$ is $\mathbf{I_n}$. Given scalars $x_1, \ldots, x_n$, the diagonal matrix $\mathbf{D}$ such that $d_{ii} = x_i$ is $\mathbf{D} = \mathbf{Diag}(x_1, \ldots, x_n)$. The rank of matrix $\mathbf{A} \in \mathbb{R}^{n \times p}$, meaning the smallest integer k such that there exist $\mathbf{B} \in \mathbb{R}^{n \times k}, \mathbf{C} \in \mathbb{R}^{k \times p}, \mathbf{A} = \mathbf{BC}$ is written $k = \mathrm{rk}(\mathbf{A})$. The squared Frobenius norm of a matrix is written:

$$\|\mathbf{X}\|^2 = \mathrm{Tr}[\mathbf{XX}'] = \sum_{i,j} x_{ij}^2 \tag{2}$$

this coincides with the euclidean norm when applied to a vector, $\|\mathbf{x}\|^2 = \sum x_i^2$.

For a symmetric matrix $\mathbf{S}$, we may write $\mathbf{S} \geqslant 0$ to mean that $\mathbf{S}$ is positive semi-definite and $\mathbf{S} > 0$ to mean positive definite. If not stated, any of these notations also implies that $\mathbf{S}$ is symmetric in the first place. For two symmetric or implied symmetric matrices we may use the partial ordering relationships $\mathbf{A} \geqslant \mathbf{B}$ and $\mathbf{A} > \mathbf{B}$ to mean that their difference $\mathbf{A} - \mathbf{B}$ is positive semi-definite or definite.

For any matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ (including a non-square one), we write $\mathbf{A}^+$ its Moore-Penrose inverse or pseudoinverse, which is the matrix obtained by inverting all the non-zero singular values of $\mathbf{A}$. The notion is further discussed in Chapter 16. Therefore, $\mathbf{AA}^+$ is the orthogonal projection on $\mathrm{im}(\mathbf{A})$, the vector sup-space of $\mathbb{R}^n$ spawned by the columns of $\mathbf{A}$. Because this operator will appear extensively in this work, we denote by $\mathbf{A}^\perp = \mathbf{I_n} - \mathbf{AA}^+$ the complementary orthogonal projection on the orthogonal complement $\mathrm{im}(\mathbf{A})^\perp$.

We consider a random variable X on a measurable set A to be exactly the same as a normalized positive measure on A; that is, a function that associate to each measurable subset $B \subset A$ a number $X(B) \geqslant 0$ and that obeys the usual axioms of probability measures.

Integrals of functions with respect to measures, that is, expected values, can be written by integrating against $\mathrm{d}X$. This means that for any f such that such expectation exists, we can define the expectation operator as:

$$\mathbb{E}\big[f(X)\big] = \int_A f \,\mathrm{d}X = \int_{x \in A} f(x) \,\mathrm{d}X(x) \tag{3}$$

If X has a density, we usually write it p_X, and we therefore have $\mathrm{d}X = p_X(x)\,\mathrm{d}x$. In context where the variable of which the density is considered is clear enough, we may drop the subscript and write simply $p(x)$.

When X is discrete or semi-discrete, note that the measure theoretic definition above allow us to still write expectations as integrals despite the absence of density. For instance if X is a discrete variable on $\mathbb{N}$, we therefore have:

$$\mathbb{E}\big[f(X)\big] = \int_{k \in \mathbb{N}} f(k) \,\mathrm{d}X(k) = \sum_{k \in \mathbb{N}} f(k) P(X = k) \tag{4}$$

and if X is a point distribution at $a \in \mathbb{R}$,

$$\mathbb{E}\big[f(X)\big] = \int_{x\in\mathbb{R}} f(x)\,\mathrm{d}X(x) = f(a) \tag{5}$$

We may do simple calculus directly with measures using differential form notations, for instance we may write $\mathrm{d}x^2 = 2\,|x|\,\mathrm{d}x$. In these cases it must be remembered that we only work with positive measures, so the absolute values of the associated differential forms. We can therefore write the implicit wedge product $\mathrm{d}x\,\mathrm{d}x = 0$, but on the other hand note that $\mathrm{d}x\,\mathrm{d}y = \mathrm{d}y\,\mathrm{d}x$ and not $-\,\mathrm{d}y\,\mathrm{d}x$ as with differential forms.

When doing differentiation, we may use a distinct use of the symbol d: given a function f on a linear space (or a differential variety) A, we may write its differential at point x as $\mathrm{d}f(x,\ \mathrm{d}x)$ where $\mathrm{d}x$ is a tangent vector of A at x (informally, a small increment to x). As the amount of differential calculus done here is limited, whether we deal with measures, or with differentials and tangents is always clear in context and specified if need be.

The covariance of two random vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ is the matrix:

$$\mathbb{C}\mathrm{ov}\big[\boldsymbol{x},\boldsymbol{y}\big] = \mathbb{E}\big[\boldsymbol{x}\boldsymbol{y}'\big] - \mathbb{E}[\boldsymbol{x}]\,\mathbb{E}\big[\boldsymbol{y}\big]' \tag{6}$$

and the covariance matrix of a vector is $\mathbb{C}\mathrm{ov}[\boldsymbol{x}] = \mathbb{C}\mathrm{ov}[\boldsymbol{x},\boldsymbol{x}]$. When X is a random scalar we may write its scalar variance as $\mathbb{V}[X] = \mathbb{C}\mathrm{ov}[X, X]$ instead.

Part I

Gene expression imputation

1 Introduction to gene expression imputation

1.1 Concept and objectives

In the following part, which constitutes the largest portion of this thesis work, we will focus in depth on statistical methods to predict the expression of a gene, in a given individual, having observed other gene expression values for this individual and the same gene in other individuals. We do not consider, at this point, any non-expression predictors such as genotype, such associations will be discussed later on. At first glance, this may seem a quite odd objective. If we can measure gene expression, we can surely do so in all genes of interest, and it is unlikely that missing values in gene expression constitute much of a problem in everyday bioinformatics.

Imputation, however, is the only task that can give us unambiguous metrics to objectively compare methods. The better a model describes and captures patterns of gene co-expression, the better it will be able to predict held-out values of gene expression that it has not seen during training. Compared to the more involved pipelines that we will describe later on, prediction errors are fast to compute, easy to interpret, and hard to mis-implement in ways that would lead to unduly rank bad models too high. Here, we need not concern ourselves yet with questions of data leakage and over-correction, of multiple testing and subtly wrong covariate handling; there is only the cold and simple truth of blind prediction performance.

Of course, the whole exercise still relies on a crucial assumption: we are here making the bet that whichever model is best able to capture patterns of co-expression and predict held-out values will in turn be the most useful starting point for other applications, such as confounder correction in eQTL calling. With this in mind, we pay mostly attention to models that have either history or potential of being used to pre-process or correct gene expression data as a step of larger pipeline, but we strive to measure and understand more systematically their predictive performance.

1.2 Dataset

As we need to evaluate all our models on an equal footing, we need to define some training and test data that will be used for all the evaluations of the following chapters. Our dataset will be the RNA-Seq expression collected by the GTEx project, specifically the `v8` release from 2017-2020 [GTE20], which is at this date publicly available at `https://gtexportal.org/home/downloads/adult-gtex/bulk_tissue_expression`. The GTEx project collected data from many tissues — up to 54 at the finest classification level. However, at this point our concern is to train a large number of different models and model variations to compare them, so repeating the exploration on several tissues is very computationally expensive. Furthermore, by leaving tissues aside during development, we save data that will not have been used at all during model building and selection for further unbiased replication of the result, whose value we fully appreciate. Therefore, this part will deliberately make use of only one tissue; we selected "Whole Blood" as it is the sample type for which the largest sample size is available. This is important for two reasons; first, this allows us to test scalability. If a method is fast enough for this dataset, it will be only faster on any other. Second, this maximizes the power to detect co-expression patterns that a model can exploit — it constitutes an "easier" dataset to predict — as well as the power to detect differences in prediction quality in the evaluation phase. Hence, it maximizes our chance to observe meaningful differences between the models we evaluate. We randomly split the data 5-fold, and then use 80% of the samples as a training set (604 participants) and 20% for evaluation (151 participants, total 755).

In terms of pre-processing, inside each sample genes are normalized by library size to count per million (CPM), then expression is quantile normalized inside each gene across samples. Genes are filtered to remove lowly expressed transcripts, keeping only genes with at least 6 reads in at least 20% of the participants. This results in 21375 "expressed" genes being retained for analysis. We then train models to perform what we will call "leave-one-gene-out", or LOGO, prediction: each expressed gene is considered in turn as being a dependent variable to predict, with all the other genes being predictors. We thus build 21375 models, which can be used to predict one by one the expression of each gene for the test individuals. Once compared to the measured values, this gives us a prediction performance metric for each gene and each method, which we need to further aggregate to obtain interpretable model comparisons.

2 Singular value decomposition (SVD)

2.1 Common use of SVD for gene expression analysis

Initially, singular value decomposition (SVD) is a linear algebra technique that is typically defined independent of any probabilistic interpretation.

Given a rectangular matrix $\mathbf{Y}$ of shape (n, p), there exist matrices $\mathbf{U}, \mathbf{D}, \mathbf{V}'$, such that:

- $\mathbf{Y} = \mathbf{UDV}'$
- $\mathbf{UU}' = \mathbf{U}'\mathbf{U} = \mathbf{I}_n$
- $\mathbf{VV}' = \mathbf{V}'\mathbf{V} = \mathbf{I}_p$
- $\mathbf{D}$ is diagonal with decreasing diagonal coefficients $d_{11} \geqslant d_{22} \geqslant \cdots \geqslant 0$

This form is often used to decompose a matrix of observations into leading and trailing components; in the context of gene expression analysis the leading component are often interpreted as confounders and factors to be removed from downstream analyses.

The jump to the probabilistic realm starts with the Eckart-Young-Mirsky [EY36] theorem. Consider the problem of finding the matrix $\mathbf{Z} \in \mathbb{R}^{n \times p}$ minimizing:

$$\|\mathbf{Z} - \mathbf{Y}\|^2 \tag{2.1}$$

under the constraint:

$$\mathrm{rk}(\mathbf{Z}) \leqslant k \tag{2.2}$$

Then this problem is solved by the leading components of $\mathbf{Y}$, namely the matrix:

$$\mathbf{L}^{(k)} = \mathbf{UD}^{(k)}\mathbf{V}' \tag{2.3}$$

where $\mathbf{D}^{(k)}$ is diagonal with the same k first diagonal coefficients as $\mathbf{D}$, $d_{11}, \ldots, d_{kk}$, and 0 for the rest of the $d_{jj}^{(k)} = 0, j > k$. Equivalently, this problem is often formulated as the search for two matrices $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times p}$ minimizing $\|\mathbf{AB} - \mathbf{Y}\|^2$, in which case the task is often called a matrix factorization problem and one of $\mathbf{A}$ or $\mathbf{B}$ is typically interpreted as the (estimated latent) *factors* of the observed data.

By itself, this result is easily spun into a first probabilistic version. If we consider a random model where:

$$\boldsymbol{Y} = \mathbf{M} + \boldsymbol{H} \tag{2.4}$$

and $\boldsymbol{H}$ is random matrix whose coefficients are $n \times p$ independent identically distributed normal variables ($\boldsymbol{H}_{ij} \sim \mathcal{N}(0, \sigma^2)$ for some $\sigma > 0$), then given an observation $\boldsymbol{Y} = \mathbf{Y}$, the Eckart-Young-Mirsky theorem tells us that the estimator $\widehat{\mathbf{M}} = \mathbf{L}^{(k)}(\mathbf{Y})$ maximizes the likelihood for $\mathbf{M}$ under the regularizing constraint $\mathrm{rk}(\mathbf{M}) \leqslant k$ (the unconstrained estimate being trivially $\mathbf{Y}$).

This first probabilistic interpretation, however, is not very useful for evaluating performance as a predictive method: here, an observation, a single sample, is a whole rectangular matrix $\mathbf{Y}$, and we would need a whole other $n \times p$ matrix of new measurements of the same values to define an out-of-sample metric. Since our parameter already has a fixed $n \times p$ shape, there is no clear way to "add new rows" or columns of previously unobserved data about which we could predict something. For this reason, in common use, instead of blind predictions, the estimated $\mathbf{L}^{(k)}$ is often directly used as an *in-sample* estimate of the expected expression values due to co-expression signal.

2.2 Extending SVD into a predictive framework

Instead, we gave ourselves the task of finding an interpretation and blind prediction setting in which we can truly evaluate how much signal is being captured with each retained singular value.

What is then more useful is to observe that the singular value decomposition of $\mathbf{Y}$ comes with two additional ones and the associated interpretations. With the notations above, $\mathbf{U}\mathbf{D}^{(k)}\mathbf{D}^{(k)}\mathbf{U}'$ also gives us the leading components of $\mathbf{Y}\mathbf{Y}'$. Consequently, we are inspired to look at a probabilistic model and estimation procedure in which we focus on the covariance matrix of $\mathbf{Y}$, now seen not as a single matrix variate but a collections of independent identically distributed or columns. Lets us say that we see $\mathbf{Y}$ as p realizations $(\mathbf{y}^1, \dots, \mathbf{y}^p)$ of a random vector on $\mathbb{R}^n$. If this vector is drawn from a multivariate random normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$, the log likelihood of $\boldsymbol{\Sigma}$ is:

$$\mathcal{L}(\boldsymbol{\Sigma}) = -\frac{p}{2} \ln |\boldsymbol{\Sigma}| - \frac{1}{2} \mathrm{Tr}(\boldsymbol{\Sigma}^{-1}\mathbf{Y}\mathbf{Y}') + \text{const} \tag{2.5}$$

It makes no sense to impose a rank constraint on $\boldsymbol{\Sigma}$ directly as it must stay regular; $\boldsymbol{\Sigma}$ here also includes the "noise" component of the model. Rather, what is attractive is to parametrize $\boldsymbol{\Sigma}$ as a low rank matrix plus a scalar multiple of the identity matrix that represents the white noise component. We therefore assume a parameter $\sigma > 0$ and a symmetric positive semi-definite matrix $\mathbf{Q}$ such that

$$\boldsymbol{\Sigma} = \mathbf{Q} + \sigma^2 \mathbf{I_n} \tag{2.6}$$

$$\mathrm{rk}(\mathbf{Q}) \leqslant k \tag{2.7}$$

Following then in the footsteps of Eckart and Young's proof, we then observe that we can parametrize

$$\mathbf{\Sigma} = \sigma^2 \mathbf{I_n} + \mathbf{WFW'} = \mathbf{W}(\mathbf{F} + \sigma^2 \mathbf{I_n})\mathbf{W'} \tag{2.8}$$

where $\mathbf{F}$ is diagonal with at most the first k diagonal coefficients being non-zero, and $\mathbf{W'W} = \mathbf{WW'} = \mathbf{I_n}$. The tangents $\mathrm{d}\mathbf{W}$ to orthonormal matrices in $\mathbf{W}$ are the matrices that satisfy $\mathrm{d}\mathbf{W'W} + \mathbf{W'}\mathrm{d}\mathbf{W} = 0$, so exactly the matrices of the form $\mathrm{d}\mathbf{W} = \mathbf{W}\mathrm{d}\mathbf{S}$ with $\mathrm{d}\mathbf{S}$ a skew-symmetric matrix. At the minimum of $\mathcal{L}$, we must therefore have for any skew-symmetric $\mathrm{d}\mathbf{S}$:

$$\mathrm{d}\mathcal{L}(\mathrm{d}\mathbf{S}) = -\mathrm{Tr}(\mathrm{d}\mathbf{S}(\mathbf{F} + \sigma^2 \mathbf{I_n})^{-1}\mathbf{W'YY'W}) = 0 \tag{2.9}$$

which exactly mean that $(\mathbf{F} + \sigma^2 \mathbf{I_n})^{-1}\mathbf{W'YY'W}$ must be symmetric. Without entering into the detailed characterization of suitable $\mathbf{W}$ matrices, it suffices for our purposes to observe that $\mathbf{W} = \mathbf{U}$ is a suitable choice as the latter expression then simplifies to $(\mathbf{F} + \sigma^2 \mathbf{I_n})^{-1}\mathbf{D}^2$ which is diagonal. We can therefore look for $\mathbf{\Sigma}$ under the form:

$$\mathbf{\Sigma} = \mathbf{U}(\mathbf{F} + \sigma^2 \mathbf{I_n})\mathbf{U'} \tag{2.10}$$

for which the likelihood reduces to:

$$\mathcal{L}(\mathbf{F}, \sigma) = -\frac{p}{2}\sum_{i=1}^{n} \ln(f_{ii} + \sigma^2) - \frac{1}{2}\sum_{i=1}^{n} \frac{d_{ii}^2}{f_{ii} + \sigma^2} \tag{2.11}$$

For each possibly non-zero f_{ii}, $i \in I$, $|I| = k$, we maximize this likelihood by setting it such that $f_{ii} + \sigma_i^2 = d_{ii}^2/p$, leaving σ^2 to maximize the remaining terms which gives the solution:

$$\sigma^2 = \frac{1}{p}\frac{1}{n-k}\sum_{i \notin I} d_{ii}^2 \tag{2.12}$$

It can then be verified, conforming to intuition, that the optimal choice of set I corresponds to the k values of i with largest d_{ii}. We do not detail the proof, but in brief, consider ad absurdum that one d_{ii}, $i \in I$ has smaller value than a d_{jj}, $j \notin I$, and observe how the optimal likelihood would change by swapping the assignments to I and how can it can only increase. Note that this utilizes the assumption $f_{ii} \geqslant 0$.

Therefore, the optimal covariance estimator $\hat{\mathbf{\Sigma}}$ under the regularization constraints $\mathbf{\Sigma} = \mathbf{Q} + \sigma^2 \mathbf{I_n}$, $\mathrm{rk}(\mathbf{Q}) \leqslant k$ is obtained by taking the singular value decomposition of $1/p\mathbf{Y'Y}$, keeping the k leading principal values and setting all the remaining ones to their average. From this, it is simply a matter of calculating its inverse $\hat{\mathbf{\Lambda}} = \hat{\mathbf{\Sigma}}^{-1}$, which is trivial given that we know by construction the diagonalization of $\hat{\mathbf{\Sigma}}$, and we can apply (17.6) to predict the conditional distribution of a coordinate of a new observation $\mathbf{y}$ given all of the other coordinates.

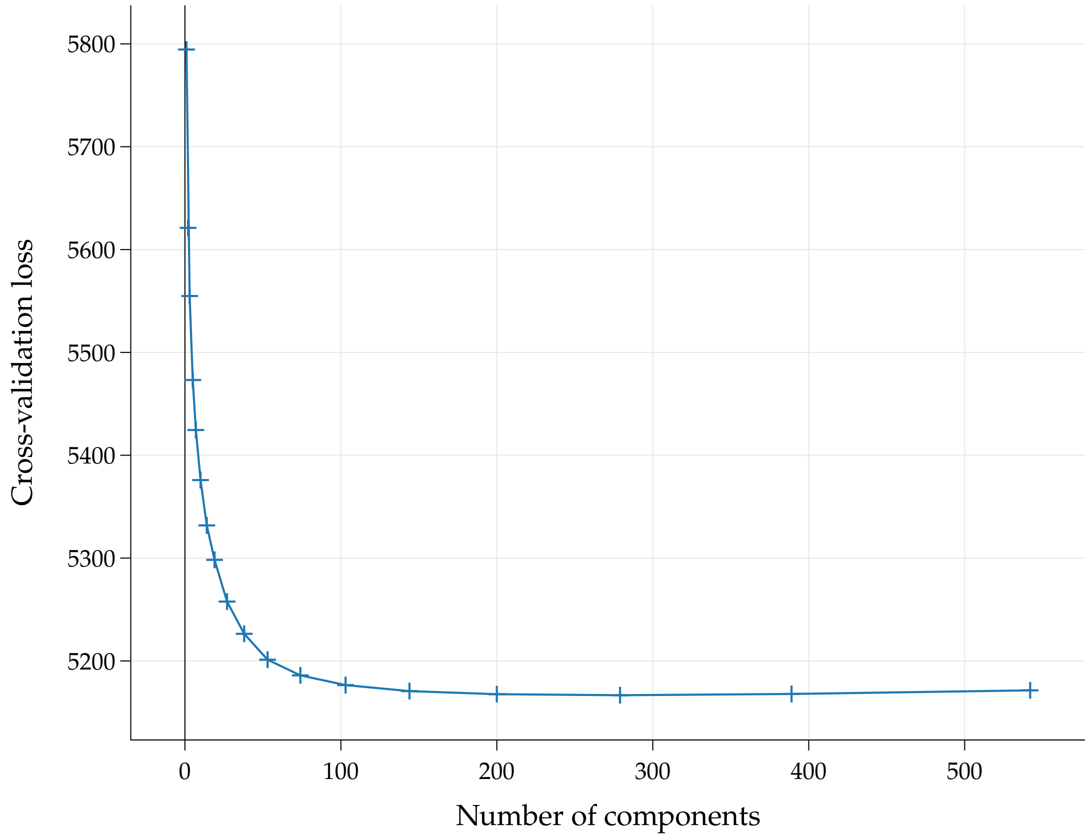

Figure 2.1: Cross-validated errors on training set as a function of the number of SVD component retained.

2.3 Commented results on real gene expression

Figure 2.1 shows how the ten-fold cross-validated predictive performance changes as a function of k, the rank of $\mathbf{Q}$ above or number of free eigenvalues of $\mathbf{\Sigma}$, when applied to our GTEx sample dataset. The performance is measured as the geometric average of the sum-of-squares errors over all genes, when comparing the blind LOGO predictions described above to the observed values. The number of components extends all the way to the maximum possible, around 540, corresponding to the number of samples and therefore the maximum number of non-zero singular values of the expression matrix $\mathbf{Y}$.

While the minimum can be located at the point right under $k = 300$, as visible on the close-up figure 2.2, it is striking that after $k = 150$ the curve is essentially flat. There is barely any noticeable "overfit" of the model when no rank constraint is applied. The reason for this is that the *concentration* of the problem, that is, the ratio of dimensions of the random vector of interest (around 20,000) to the number of samples (around 540) is very far from 1. In such a situation, despite the fact that we're estimating

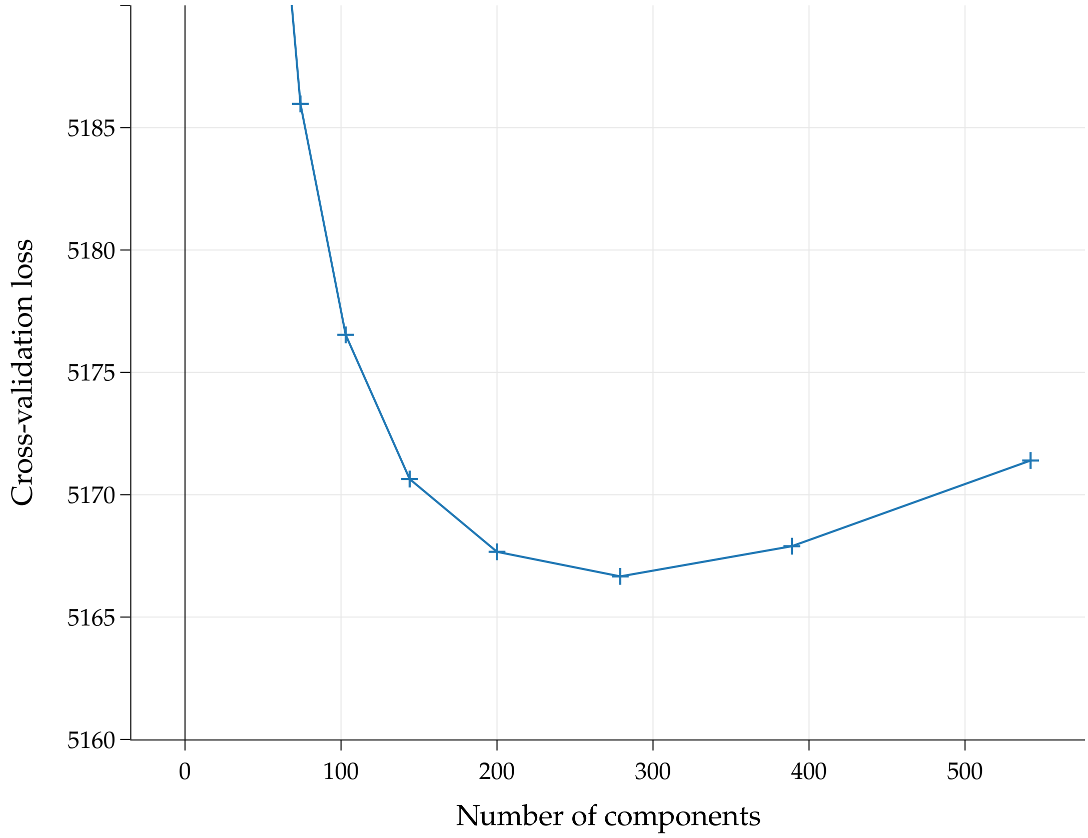

Figure 2.2: Close-up of the lower part of figure 2.1.

a huge covariance matrix from few samples, the part of the spectrum of Σ that we recover is actually pretty stable and accurate. The true covariance matrix has 20,000 eigenvalues, but we only observe the first few hundreds, so even the last of these is only moderately distorted; and the regularization introduced by the rank constraint has a very limited positive impact. Said yet in an other way, since the estimated precision matrix is completely dominated by the 10,460 components that have an observed variance of exactly 0, we do not make much of a change by adding one or two hundreds components to the lower end of the spectrum.

Conversely, it is not uncommon to see applications where only a few dozen components are used. For this case the results suggests that this may be too little and that some co-expression signal would be lost. The reason why such restricted numbers are found is because the pipelines in which they are used typically do not have a blind setting, in which the factors would be estimated on different samples from the ones that are to be predicted. In that case of course catastrophic leakage and overfit occur as the number of components retained grows.

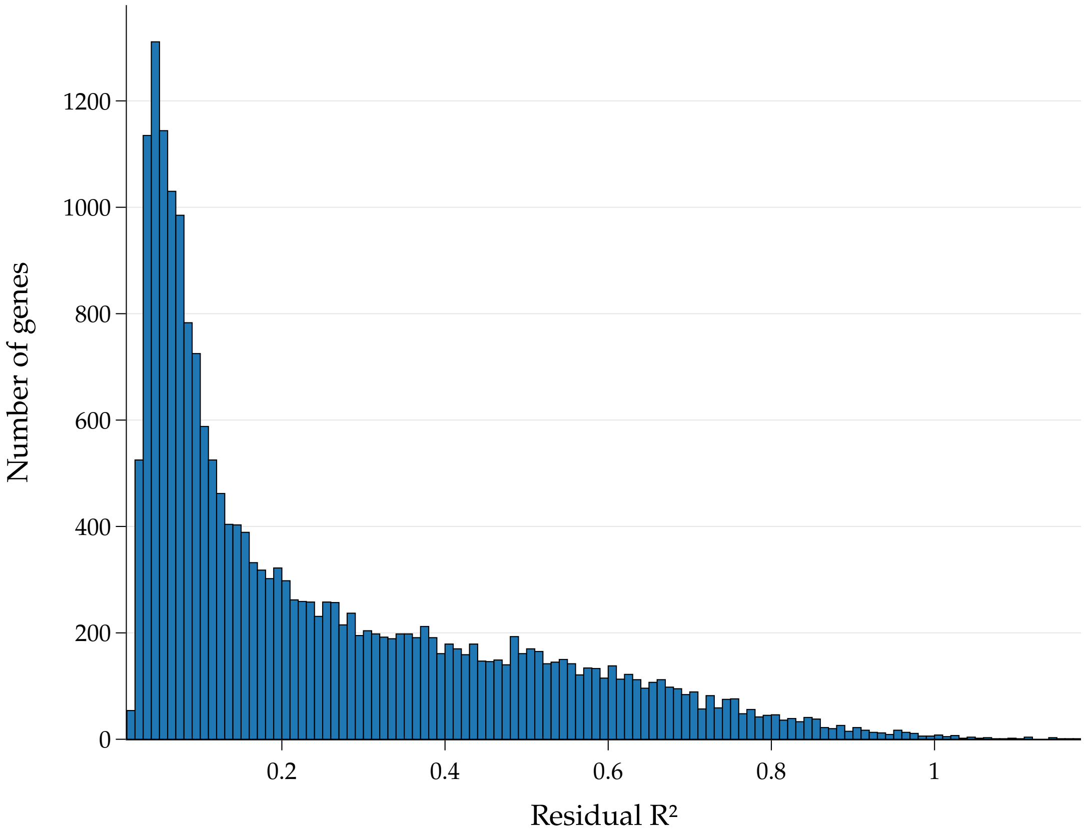

Figure 2.3: Gene predictions errors (LOGO) for SVD method.

3 The PEER method

3.1 Description

The PEER software (for Probabilistic Estimation of Expression Residuals) offers a method for dimensionality reduction and residualization often described as a Bayesian version of singular value decomposition [Ste+10]. It is obtained by taking a probabilistic formulation of singular value decomposition, complete with priors and hyperpriors. The resulting hierarchical model cannot be integrated out analytically, so textbook mean-field variational inference is applied, resulting in a coordinate-ascent evidence lower bound (ELBO) maximization algorithm that results in probabilistic estimates of the parameters.

As for the singular value decomposition or classical matrix factorization described in Chapter 2, the method is described as a model for generating a single matrix which is considered a unique observation; and as such also hasn't been evaluated in a blind context. However, as its formulation is already a factored generative model, the extension is here rather straightforward. We start by reproducing here the necessary part of the model building, stripped of some detail and discussion of known factors integration.

In the PEER model, we also try to approximate the $n \times p$ (samples $\times$ genes) expression matrix $\boldsymbol{Y}$ by a rank-deficient factorization $\boldsymbol{ZT}$. Each sample, a row of the expression matrix $\boldsymbol{Y}$, is described as generated as a combination of latent hidden factors $\boldsymbol{T}$ with latent *activations* $\boldsymbol{Z}$ plus some Gaussian noise. To give an intuitive interpretation, a dataset where the proportion of a particular cell type varied from sample to sample, causing a co-expression pattern to appear, could be generated by such a model by having a row $\boldsymbol{t}_l \in \mathbb{R}^p$, with each entry corresponding to a gene, with higher values for genes that are expressed in the cell type, then the corresponding column $\boldsymbol{z}^l \in \mathbb{R}^n$ would have for each sample i a higher value Z_{li} if that sample has a higher proportion of the cell type. In practice there is however no guarantee that the learned factors would have any straightforward biological interpretation.

Probabilistically, each sample is then modeled as:

$$\boldsymbol{y}_i' = \boldsymbol{z}_i'\boldsymbol{T} + \boldsymbol{\varepsilon}_i \tag{3.1}$$

with each coordinate of the noise being independent but this time with its own precision

$\tau_j > 0$:

$$\boldsymbol{\varepsilon}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Diag}(\tau_1, \ldots, \tau_p)^{-1}) \tag{3.2}$$

while the factors $\boldsymbol{T} \in \mathbb{R}^{k \times p}$ are shared across all samples, and the loadings $\boldsymbol{z}_i$ differ between each sample and are independently drawn from a standard normal prior:

$$\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{I_k}) \tag{3.3}$$

Note that the latent factors $\boldsymbol{T}$ and the precisions $\boldsymbol{\tau}$ themselves have priors and hyperparameters which are non-trivial and important for the method but that we do not detail here as they do not impact evaluation. Instead, it is enough to say that based on these priors and variational inference principles, PEER maintain variational distributions $\tilde{\boldsymbol{T}}$, $\tilde{\boldsymbol{\tau}}$, $\tilde{\boldsymbol{Z}}$, and that where point estimates are needed we can access the variational posterior means of these quantities as our estimators of the model variables as:

$$\hat{\mathbf{T}} = \mathbb{E}\big[\tilde{\boldsymbol{T}}\big] \tag{3.4}$$

$$\hat{\boldsymbol{\tau}} = \mathbb{E}[\tilde{\boldsymbol{\tau}}] \tag{3.5}$$

$$\hat{\mathbf{Z}} = \mathbb{E}\big[\tilde{\boldsymbol{Z}}\big] \tag{3.6}$$

3.2 Extensions to blind prediction

Given the hierarchical model above, we can deduce a generative model for a new sample (a row $i \notin \{1, \ldots, n\}$) while leaving the parameters $\hat{\mathbf{T}}$ and $\hat{\boldsymbol{\tau}}$ to their estimated value on the training samples. In that sense, note that we're doing a plug-in or collapsing of the posterior for $\boldsymbol{T}$ and $\boldsymbol{\tau}$ that PEER learns and we use only the variational posterior mean. Then, given these point estimates,

$$\boldsymbol{z}_i'\hat{\mathbf{T}} \sim \mathcal{N}\big(\mathbf{0}, \hat{\mathbf{T}}'\hat{\mathbf{T}}\big) \tag{3.7}$$

and

$$\boldsymbol{y}_i \sim \mathcal{N}\big(\mathbf{0}, \hat{\mathbf{T}}'\hat{\mathbf{T}} + \mathbf{Diag}(\hat{\tau}_1, \ldots, \hat{\tau}_p)^{-1}\big) \tag{3.8}$$

This approach, it must be noted, discards the estimation of the activations $\boldsymbol{Z}$ that PEER performs. If the generative model of PEER is a good fit, then the estimated $\boldsymbol{z}_i, i \in \{1, \ldots, n\}$ should look like standard Gaussian samples and contain no relevant information. On the other hand, the mean-field variational algorithm used for fitting has the capacity to learn some activations that do not fit the prior that well. In that case, the predictive model above may suffer from being quite different from what PEER actually saw during training. To account for that, a modification to the prediction extension is to consider that the new activations for $i > n$ are actually from a different multivariate normal:

$$\boldsymbol{z}_i \sim \mathcal{N}(0, \hat{\mathbf{\Sigma}}_{\boldsymbol{Z}}) \tag{3.9}$$

where we estimate:

$$\hat{\mathbf{\Sigma}}_Z = \frac{1}{n}\hat{\mathbf{Z}}'\hat{\mathbf{Z}} \tag{3.10}$$

We call this alternative the “covariance re-estimation” strategy in what follows. Once again, we actually expect $\hat{\mathbf{\Sigma}}_Z$ to ideally be very close to $\mathbf{I_k}$, but this gives a bit more flexibility to the model at the expense of the risk of overfitting $\hat{\mathbf{\Sigma}}_Z$ to the training sample and therefore hindering generalization.

Since we extract only the approximate posterior means of the parameters and use them as point estimates, we also fear that the point $\hat{\boldsymbol{\tau}}$ may not be accurate for our use case. Therefore, instead of using the variational posterior means as-is, we may opt for replacing them with their maximum likelihood estimates, by extracting the residuals:

$$\hat{\mathbf{R}} = \mathbf{Y} - \hat{\mathbf{Z}}\hat{\mathbf{T}} \tag{3.11}$$

and then setting $\hat{\boldsymbol{\tau}}$ to the inverse mean square of each column of residuals. This we denote as the “precisions re-estimation” strategy going forward.

Therefore, by choosing to use any combination of approaches on these two questions, we obtain four different strategies to try out.

3.3 Comparison of predictive strategies

Figure 3.1 show the impact of the extension choices discussed in Section 3.2. The overall conclusions is that the impact of these strategies is extremely limited and inconsistent. The covariance re-estimation strategy has the least noticeable impact, which tends to confirm that the estimated PEER activations behave very much like white noise and fit well their standard normal prior. The impact of re-estimating the precisions is more subtle; it results overall in smaller estimated gene residual variance, which translates essentially to less shrinkage or regularization of the predictions for the same number of factors. This is beneficial for low-error genes, which pull the mean down, but tend to work less well for typical or high-error genes.

3.4 Impact of the number of factors

Figure 3.2 shows how the cross-validated performance on the training data varies with more PEER factors. Note that we are limited to 100 components by run time constraints. Fitting a model with default parameters, which implies up to 1000 iterations of the mean-field updates, can take up to 10 hours on our test hardware for the models with 100 components (smaller models are much faster to fit). With this limitation, we can only observe that we are not able to reach a potential point where too many factors would start to penalize the models, which we would look for at around 300 components based on the equivalent results for singular value decompositions from Chapter 2.

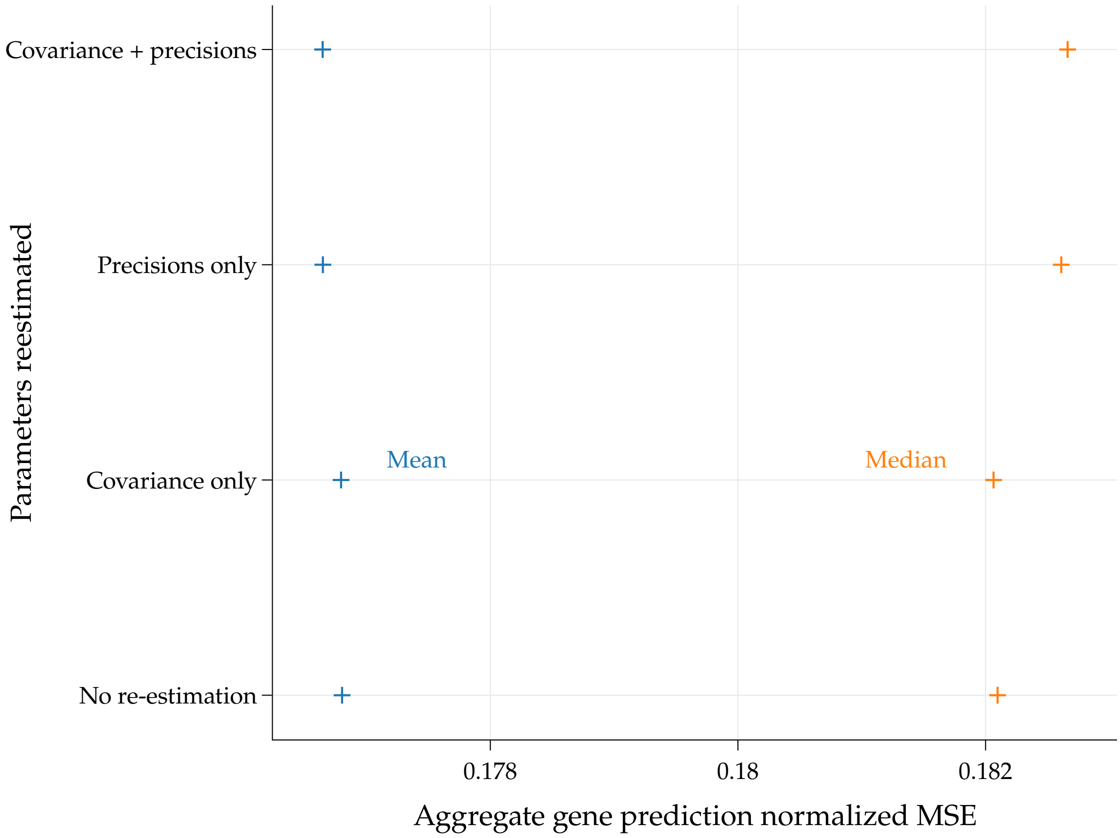

Figure 3.1: Effect on predictive performance of the extension strategies. This is evaluated on GTEx blood expression, with 60 PEER components (arbitrary choice). Mean here is taken to be the geometric mean, or mean in log-space.

3.5 Integration of PEER in bioinformatic pipelines

We've seen how many potential options offer themselves when it comes to turning the statistical modeling and fitting algorithm of PEER into usable outputs. Crucially, prediction is not the only application in which this issue comes up, there are also several ways the PEER model and algorithm can be employed to control for learned covariates. Anticipating slightly on Part II, we want to enumerate how similar challenges appear.

- The original PEER publication aimed at fitting jointly the learned factors and some genetic effects, and demonstrated the usefulness of the method. However, the initial implementation had important constraints, such as fitting on all genetic markers jointly but only allowing one to be active. It therefore offered no way to efficiently share calculations for the estimation of many associations,and to our knowledge this original design has not really been re-used as originally designed.

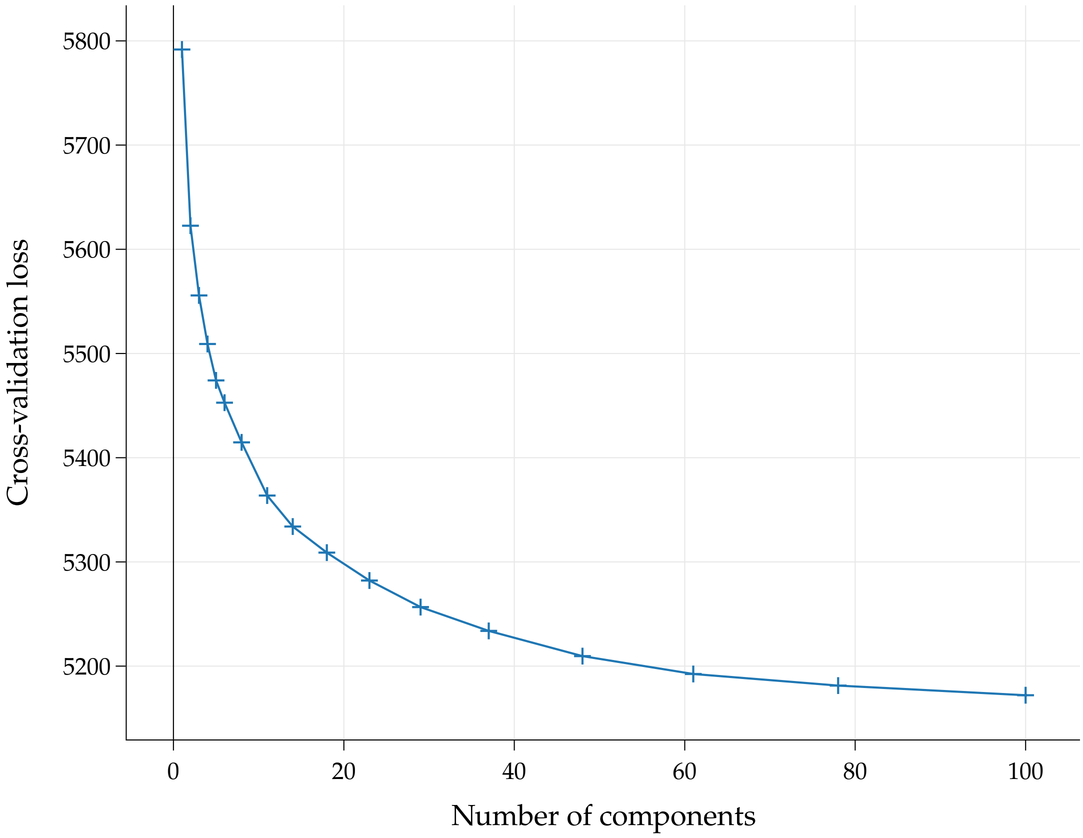

Figure 3.2: Effect on predictive performance of the number of PEER factors.

- Alternatively, the algorithm yields an estimation of the residuals of the models, that can be interpreted as "corrected" expression values and used in downstream steps. This is the use of PEER that has probably been the most common, though we detail in Section 10.1 why this alone cannot be considered a satisfying integration strategy.
- Finally, the learned factors of the decomposition can be interpreted as "inferred factors" and the corresponding activations; and such factors can be added to the set of control covariates used in the rest of the analysis, which is how we've seen the model employed most frequently in eQTL studies. This, however, poses a problem of leakage as factors learned from the data are treated as covariates completely unrelated to it, as we will see.

4 Bayesian linear models and the MacKay optimization method

4.1 Efficient optimization in Bayesian linear models

An important strategy to build linear models with large numbers of predictors is to shrink effect sizes, which can be achieved in a Bayesian framework by using an informative prior on said effect sizes. Equivalently, models of this type are known in the frequentist literature as random effects models (or mixed effects models when fixed effects are also included).

We will denote as Y_i the target variable we want to model for a sample $i \in \{1, \dots, N\}$. In our applications, this would be the expression value for a given gene. For this target, we assume a large number of predictors $x_{i,j}$ with $j \in \{1, \dots, P\}$ with P being large, and notably possibly larger than N. As we will see in Section 4.4, it will eventually make sense to use as predictors other genes, though doing so with any success will require complicating the method that we limit ourselves to describe in this introductory section. The effect of each of these predictors will be given by a random weight or effect size W_j.

Adding the model parameters σ_y and σ_w controlling respectively the amount of noise in the target and the size of the random effects, such a model can be written hierarchically as:

$$\forall i, 1 \leqslant i \leqslant N, \quad Y_i \sim \mathcal{N}\left(\sum_{j=1}^{P} W_j x_{ij}, \sigma_y^2\right) \tag{4.1}$$

$$\forall j, 1 \leqslant j \leqslant P, \quad W_j \sim \mathcal{N}(0, \sigma_w^2) \tag{4.2}$$

Or more compactly with multivariate notations:

$$\boldsymbol{y} \sim \mathcal{N}\left(\mathbf{X}\boldsymbol{w}, \sigma_y^2 \mathbf{I_N}\right) \tag{4.3}$$

$$\boldsymbol{w} \sim \mathcal{N}\left(0, \sigma_w^2 \mathbf{I_P}\right) \tag{4.4}$$

It can be seen readily that:

$$\mathbf{X}\boldsymbol{w} \sim \mathcal{N}\left(0, \sigma_w^2 \mathbf{X}\mathbf{X}'\right) \tag{4.5}$$

hence:

$$\boldsymbol{y} \sim \mathcal{N}\left(0, \sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right) \tag{4.6}$$

Thus the log-likelihood, (or *marginal* log-likelihood) of that model can be written:

$$\mathcal{L}(\sigma_y^2, \sigma_w^2) = -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right| - \frac{1}{2}\mathbf{y}'\left(\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1}\mathbf{y} \tag{4.7}$$

This likelihood can naturally be optimized by black-box gradient-based optimizers. A crucial part of getting models of this family to scale up, however, is to derive an optimization update rule based on a more global approximation of the objective function, which ensures very fast convergence to a neighbourhood of the optimum even from bad initialization points.

In short, black-box optimization methods work by constructing an at best quadratic approximation of the objective function in the neighbourhood of the current trial point. Such an approximation only becomes accurate if we are indeed already close to the optimum. For quick global convergence, we want an approximation of the objective function which, even when based on a point far from the optimum, gives a good idea of where the global optimum is.

We learned of a solution for this problem from two papers of David J.C. MacKay from 1992 [Mac92b][Mac92a], who in turn cites Stephen F. Gull [Gul89] as his inspiration for the construction. The formulation in these papers is more generic that what will follow, and the authors develop it quite concisely in between other topics; therefore we deem suitable to offer in the next sections our own exposition of the underlying principles in the more specific context of Bayesian linear models. This also greatly simplifies the explanation of the extension that we build in section 4.4 and which constitutes our main contribution to this theory and its applications.

The determinant $\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|$ has a simple approximate behavior as a function of σ_w^2: for small σ_w^2, it is constant since $\sigma_y^2 \mathbf{I_N}$ dominates; for large σ_w^2, it is a power function $\propto (\sigma_w^2)^R$ with R the rank of $\mathbf{X}\mathbf{X}'$. This motivates approximating it by "some" power function, with the exponent interpolating between 0 and R, and representing the *effective model size*: when shrinkage is weak this is the number of linearly independent covariates, and as shrinkage increases (σ_w^2 becomes small) it decreases to make a "smaller model" down to 0. Practically, we want to linearize $\ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|$ as a function of $\ln \sigma_w^2$. Given a trial point s_w^2, we therefore approximate as constant:

$$\left.\frac{\partial \ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|}{\partial \ln \sigma_w^2}\right)_{\sigma_w^2} \approx \left.\frac{\partial \ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|}{\partial \ln \sigma_w^2}\right)_{s_w^2} \tag{4.8}$$

whence:

$$\left.\frac{\partial \ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|}{\partial \sigma_w^2}\right)_{\sigma_w^2} \approx \frac{s_w^2}{\sigma_w^2} \left.\frac{\partial \ln\left|\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right|}{\partial \sigma_w^2}\right)_{s_w^2} \tag{4.9}$$

$$\approx \frac{s_w^2}{\sigma_w^2} \mathrm{Tr}\left(\left[s_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right]^{-1} \mathbf{X}\mathbf{X}'\right) \tag{4.10}$$

$$\approx \frac{1}{\sigma_w^2}\left(N - \mathrm{Tr}\left(\left[s_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right]^{-1} \sigma_y^2\right)\right) \tag{4.11}$$

$$\approx \frac{1}{\sigma_w^2}\left(N - \mathrm{Tr}\left[s_w^2 \sigma_y^{-2} \mathbf{X}\mathbf{X}' + \mathbf{I_N}\right]^{-1}\right) \tag{4.12}$$

The quantity $\gamma := N - \mathrm{Tr}\left[s_w^2 \sigma_y^{-2} \mathbf{X}\mathbf{X}' + \mathbf{I_N}\right]^{-1}$ is the aforementioned exponent that represents the model size, it can be seen how it effectively counts the number of significant eigenvalues of $\mathbf{X}\mathbf{X}'$ at the current noise level.

For the other σ_w^2-dependent term in the likelihood, the important global feature we want to capture is how it converges to the unpenalized residuals (which may be 0 if $\mathbf{X}\mathbf{X}'$ has full rank, i.e., if we have enough covariates to fully explain $\boldsymbol{Y}$ without noise) with asymptotic behavior $\propto \sigma_w^{-2}$. For this term, we therefore approximate as constant:

$$\left.\frac{\partial \mathbf{y}'\left(\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{y}}{\partial \sigma_w^{-2}}\right)_{\sigma_w^2} \approx \left.\frac{\partial \mathbf{y}'\left(\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{y}}{\partial \sigma_w^{-2}}\right)_{s_w^2} \tag{4.13}$$

whence:

$$\left.\frac{\partial \mathbf{y}'\left(\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{y}}{\partial \sigma_w^2}\right)_{\sigma_w^2} \tag{4.14}$$

$$\approx \frac{s_w^4}{\sigma_w^4} \left.\frac{\partial \mathbf{y}'\left(\sigma_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{y}}{\partial \sigma_w^2}\right)_{s_w^2} \tag{4.15}$$

$$\approx -\frac{s_w^4}{\sigma_w^4} \mathbf{y}'\left(s_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{X}\mathbf{X}'\left(s_w^2 \mathbf{X}\mathbf{X}' + \sigma_y^2 \mathbf{I_N}\right)^{-1} \mathbf{y} \tag{4.16}$$

$$\approx -\frac{1}{\sigma_w^4}\left\|\mathbf{X}'\left(\mathbf{X}\mathbf{X}' + \sigma_y^2 s_w^{-2} \mathbf{I_N}\right)^{-1} \mathbf{y}\right\|^2 \tag{4.17}$$

Putting these approximations together and solving for $\frac{\partial L}{\partial \sigma_w^2} = 0$ gives the guess for the optimum of the likelihood:

$$\sigma_w^2 = \frac{\left\|\mathbf{X}'\left(\mathbf{XX}' + \sigma_y^2 s_w^{-2}\mathbf{I_N}\right)^{-1}\mathbf{y}\right\|^2}{N - \mathrm{Tr}\left[s_w^2\sigma_y^{-2}\mathbf{XX}' + \mathbf{I_N}\right]^{-1}} \tag{4.18}$$

The reasoning is identical for σ_y, with the roles of the parameters σ_y and σ_s swapped as well as the covariance components $\mathbf{XX}'$ and $\mathbf{I_N}$. Given a trial point s_y^2, this gives the guess for the optimum as:

$$\sigma_y^2 = \frac{\left\|\left(\sigma_w^2 s_y^{-2}\mathbf{XX}' + \mathbf{I_N}\right)^{-1}\mathbf{y}\right\|^2}{\mathrm{Tr}\left[s_w^2\sigma_y^{-2}\mathbf{XX}' + \mathbf{I_N}\right]^{-1}} \tag{4.19}$$

These formula have a further interpretation. First, readers used to Bayesian or random effect linear estimation will have recognized in (4.18) the posterior mean for the latent effect sizes:

$$\hat{\mathbf{w}} := \mathbb{E}[w|Y] = (\sigma_y^{-2}\sigma_w^2\mathbf{I_P} + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{4.20}$$

so that the expression for σ_w^2 reads:

$$\sigma_w^2 = \frac{\|\hat{\mathbf{w}}\|^2}{\gamma} \tag{4.21}$$

In other words, the estimate for the variance of the weights is given by the total sum of squares of the posterior mean weights, divided by the *effective number of weights*.

Then, consider the push-through identity, for any pair of matrices $\mathbf{U}, \mathbf{V}$:

$$\mathbf{U} + \mathbf{UVU} = \mathbf{U}(\mathbf{I} + \mathbf{VU}) = (\mathbf{I} + \mathbf{UV})\mathbf{U} \tag{4.22}$$

so by multiplying on each side, given invertibility:

$$(\mathbf{I} + \mathbf{UV})^{-1}\mathbf{U} = \mathbf{U}(\mathbf{I} + \mathbf{VU})^{-1} \tag{4.23}$$

This can be used to decompose $\mathbf{y}$ as:

$$\begin{aligned}
\mathbf{y} &= (\mathbf{I_N} + \sigma_w^2\sigma_y^{-2}\mathbf{XX}')^{-1}(\mathbf{I_N} + \sigma_w^2\sigma_y^{-2}\mathbf{XX}')\mathbf{y} && (4.24)\\
&= \mathbf{X}(\mathbf{I_P} + \sigma_w^2\sigma_y^{-2}\mathbf{X}'\mathbf{X})^{-1}\sigma_w^2\sigma_y^{-2}\mathbf{X}'\mathbf{y} + (\mathbf{I_N} + \sigma_w^2\sigma_y^{-2}\mathbf{XX}')^{-1}\mathbf{y} && (4.25)\\
&= \mathbf{X}(\sigma_y^{-2}\sigma_w^2\mathbf{I_P} + \mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} + (\mathbf{I_N} + \sigma_w^2\sigma_y^{-2}\mathbf{XX}')^{-1}\mathbf{y} && (4.26)\\
&= \mathbf{X}\hat{w} + (\mathbf{I_N} + \sigma_w^2\sigma_y^{-2}\mathbf{XX}')^{-1}\mathbf{y} && (4.27)
\end{aligned}$$

Therefore, $\hat{\mathbf{r}} := (\mathbf{I_N} + \sigma_w^2 \sigma_y^{-2} \mathbf{X}\mathbf{X}')^{-1} \mathbf{y} = \mathbf{y} - \mathbf{X}\hat{w}$ correspond to the posterior mean residuals of the models, so (4.19) can be rewritten:

$$\sigma_y^2 = \frac{\|\hat{\mathbf{r}}\|^2}{N - \gamma} \tag{4.28}$$

meaning, the posterior mean residual sum of squares divided by the effective residual degrees of freedom.

4.2 Generalization to an arbitrary number of covariance components

More generally, given K positive semi-definite matrices $\mathbf{C}_1, \ldots, \mathbf{C}_K$ of which at least one is positive definite, we can consider a random vector $\boldsymbol{y}$ distributed as a function of parameters $\sigma_1^2, \ldots, \sigma_K^2 > 0$ as:

$$\boldsymbol{y} \sim \mathcal{N}\left(0, \sum_{i=1}^{K} \sigma_i^2 \mathbf{C}_i\right) \tag{4.29}$$

with log-likelihood:

$$\mathcal{L}(\boldsymbol{\sigma}) = -\frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln|\boldsymbol{\Sigma}(\boldsymbol{\sigma})| - \frac{1}{2} \mathbf{y}' \boldsymbol{\Sigma}(\boldsymbol{\sigma})^{-1} \mathbf{y} \tag{4.30}$$

where

$$\boldsymbol{\Sigma}(\boldsymbol{\sigma}) := \sum_{i=1}^{K} \sigma_i^2 \mathbf{C}_i \tag{4.31}$$

As before, to find good global update rules, we want to approximate $|\boldsymbol{\Sigma}(\boldsymbol{\sigma})|$ by some function $\propto \prod_{i=1}^{K} (\sigma_i^2)^{\gamma_i}$ which has the same gradient at a given point $\boldsymbol{\sigma} = \mathbf{s}$. We therefore approximate:

$$\frac{\partial \ln|\boldsymbol{\Sigma}(\boldsymbol{\sigma})|}{\partial \sigma_i^2} \approx \frac{s_i^2}{\sigma_i^2} \operatorname{Tr}\left[\boldsymbol{\Sigma}(\mathbf{s})^{-1} \frac{\partial \boldsymbol{\Sigma}(\boldsymbol{\sigma})}{\partial \sigma_i^2}\right] \tag{4.32}$$

$$\approx \frac{\gamma_i(\mathbf{s})}{\sigma_i^2} \tag{4.33}$$

with

$$\gamma_i(\mathbf{s}) := \operatorname{Tr}\left[\boldsymbol{\Sigma}(\mathbf{s})^{-1} s_i^2 \mathbf{C}_i\right] \tag{4.34}$$

the effective number of parameters for group i, or the *effective rank* of $\mathbf{C}_i$.

Then we approximate the term $\mathbf{y}'\boldsymbol{\Sigma}(\boldsymbol{\sigma})^{-1}\mathbf{y}$ by some function $\sum_{i=1}^{k} m_i \sigma_i^{-2}$ + const with a matching gradient at $\mathbf{s}$:

$$\frac{\partial \mathbf{y}'\boldsymbol{\Sigma}(\boldsymbol{\sigma})^{-1}\mathbf{y}}{\partial \sigma_i^2} \approx -\frac{s_i^4}{\sigma_i^4}\mathbf{y}'\boldsymbol{\Sigma}(\mathbf{s})^{-1}\mathbf{C}_i\boldsymbol{\Sigma}(\mathbf{s})^{-1}\mathbf{y} \tag{4.35}$$

$$\approx -\frac{m_i(\mathbf{s})}{\sigma_i^4} \tag{4.36}$$

where:

$$m_i(\mathbf{s}) := s_i^4 \mathbf{y}'\boldsymbol{\Sigma}(\mathbf{s})^{-1}\mathbf{C}_i\boldsymbol{\Sigma}(\mathbf{s})^{-1}\mathbf{y} \tag{4.37}$$

Given these auxiliary parameters, the optimum of the surrogate objective function is given by:

$$\sigma_i^2 = \frac{m_i(\mathbf{s})}{\gamma_i(\mathbf{s})} \tag{4.38}$$

4.3 Automatic relevance determination (ARD)

Automatic relevance determination (ARD) is an example of the general model above that is often optimized with MacKay updates. It may actually be the most popular use of the method, and has also been developed under the name of "sparse Bayesian learning" [Tip01]. In that case, the model has one variance parameter per covariate and is given by:

$$\boldsymbol{y} \sim \mathcal{N}\left(\sum_{j=1}^{P} W_j \mathbf{x}_j, \sigma_y^2 \mathbf{I_N}\right) \tag{4.39}$$

$$W_j \sim \mathcal{N}(0, \sigma_j^2) \tag{4.40}$$

which matches the description of (4.29) with $P+1$ components, the last being $\mathbf{C}_{P+1} = \mathbf{I_N}$ and all the others being the rank-one contributions of each variable to the covariance structure $\mathbf{C}_j = \mathbf{x}_j\mathbf{x}_j'$.

As such, ARD is an extreme case of the model above, since all components have rank one. In practice, introducing that many variance components has rather niche usefulness, because it defeats the point of random effect models: pooling information and reducing model complexity by introducing variance parameters that average over several features whose weights are hard to estimate in the first place. From the original paper, it can be seen that this is mostly useful when the number of parameters is low compared to the sample size, so that the number of parameters is not the main concern, and the true model has some sparsity, that is, there is a large dispersion in the scales of the true effect sizes. This is far from the case we are facing with observational gene expression, and we therefore focus our attention on building meaningful groups of covariates and variance components instead.

4.4 The clustered MacKay model (CMK) and algorithm

The idea of the model that we propose is to apply the concept of Bayesian linear models and the MacKay optimization method summarized in sections 4.1 and following with a careful parametrization and parameter-sharing strategy. It applies on top of a pre-processing step, in which the genes are first clustered through the k-means algorithm, based on the euclidean distance between the gene expression vectors. For this reason we call the resulting model the *clustered MacKay* model, or CMK for short. Through this pre-processing we obtain groups of genes, typically, one would cluster 20 000 genes into 100 groups, thus forming groups with on average 200 genes each.

Formally, we assume that each gene $g \in \{1, \ldots, P\}$ is assigned to a group $G(g) \in \{1, \ldots, K\}$.

Given these groups, we can build a random effects linear regression model for each gene, conditionally on all other genes (therefore, CMK belongs to the high-dimensional regression framework in the taxonomy developed before). The essential idea is that we want to learn variance hyperparameters that will strongly regularize most of the effect sizes of this very large model in which each gene has tens of thousand of predictors, balancing the need to have enough groups of jointly regularized predictors to keep a very flexible problem while not falling in the extreme of one hyperparameter per predictor as with ARD — which we empirically found to overfit quite badly in this setting. Therefore, we elect to keep one hyperparameter for each group of target genes and for each group of predictors, reasoning that two correlated genes are likely to be similarly useful to include or exclude in the model together, and that conversely two correlated target genes will likely be well predicted by including or excluding a similar set of other genes as their predictors. Since all of our gene expression values are quantile normalized to begin with, we also don't have to worry about the scale of the effect sizes, and can directly use the same exact effect size prior variance for different predictors, thinking of it as a continuous dimensionless variable controlling inclusion/exclusion of predictors in the model.

Such a model can be formally written, for a target gene g, as:

$$\mathbf{Y}_g \sim \mathcal{N}(\boldsymbol{\mu}_g, \sigma_g^2 \mathbf{I_N}) \tag{4.41}$$

$$\boldsymbol{\mu}_g := \sum_{h \neq g} w_{h,g} \mathbf{y}_h \tag{4.42}$$

$$w_{h,g} \sim \mathcal{N}(0, \sigma_g^2 v_{G(h)G(g)}) \tag{4.43}$$

Therefore, the conditional effect sizes $w_{h,g}$ can be integrated out, and the resulting model is parametrized by the gene residual variances σ_g^2 and the effect size variance hyperparameters $v_{G(h)G(g)}$. This way of formulating the model, while far from being the only or even best alternative, keeps it in the family of models defined in sections 4.1 and following, meaning that we can employ the same basic ideas to derive efficient update strategies for the σ_g^2 and v_{kt} parameters.

As the notation implies, we further *couple* the regression models for all target genes by assuming that the variances v only depend on the *groups* of genes considered. Formally, the priors on all effect sizes w are tied by the relationship:

$$\forall g, g', h, h', \quad G(g) = G(g') \wedge G(h) = G(h') \implies \sigma_g^{-2}\mathbb{V}\left[w_{h,g}\right] = \sigma_{g'}^{-2}\mathbb{V}\left[w_{h',g'}\right] \tag{4.44}$$

Crucially, this coupling is taken to mean that all the parameters for a target group T are optimized together with the aggregated objective function:

$$L_T\left((\sigma_g)_{G(g)=T}, (v_{k,T})_{1\leqslant k\leqslant K}\right) = \sum_{G(g)=T} \ln p\left(\mathbf{y}_g|(\mathbf{y}_h)_{h\neq g}, \sigma_g, (v_{k,T})_{1\leqslant k\leqslant K}\right) \tag{4.45}$$

All the essence and the difficulty of the CMK algorithm consists in computing L_T and its gradients or updates in reasonable time by exploiting the structure of this compound loss function. Indeed, the naive complexity of computing the likelihood of models of the type of (4.29) is $O(N^3)$ (due to the cost of computing the determinant and inverse of the N-by-N covariance matrix), so independent computation of the P such likelihoods that occur in the $(L_t)_{1\leqslant t\leqslant K}$ would have the unacceptable complexity $O(PN^3)$.

Instead, we will establish here that all the necessary computations can be performed in $O(PN^2 + N^3)$ complexity.

The log-likelihood for one gene g in a target group $G(g) = t$ can be written:

$$L_g(\sigma_g, \mathbf{v}_{.,t}) = -\frac{N}{2}\ln(2\pi) - \frac{N}{2}\ln\sigma_g^2 - \frac{1}{2}\ln|\mathbf{\Sigma}_g(\mathbf{v}_t)| - \frac{1}{2}\sigma_g^{-2}\mathbf{y}_g'\mathbf{\Sigma}_g(\mathbf{v}_t)^{-1}\mathbf{y}_g \tag{4.46}$$

where

$$\mathbf{\Sigma}_g(\mathbf{v}_t) := \mathbf{I_N} + \sum_{k=1}^{K} v_{k,t} \sum_{\substack{G(h)=k\\ h\neq g}} \mathbf{y}_h\mathbf{y}_h' \tag{4.47}$$

We shorten the Gram matrices that will occur often in the following derivations as:

$$\mathbf{C}_k := \sum_{G(h)=k} \mathbf{y}_h\mathbf{y}_h' \tag{4.48}$$

for the Gram matrix of group k, and

$$\mathbf{C}_k^{\neg g} := \sum_{\substack{G(h)=k\\ h\neq g}} \mathbf{y}_h\mathbf{y}_h' \tag{4.49}$$

for the same with $\mathbf{y}_g$ removed. Note that $\mathbf{C}_k^{\neg g}$ is just trivially $\mathbf{C}_k$ for $k \neq G(g)$. With these, we have:

$$\mathbf{\Sigma}_g(\mathbf{v}_t) = \mathbf{I_N} + \sum_{k=1}^{K} v_{k,t}\mathbf{C}_k^{\neg g} \tag{4.50}$$

$$= \mathbf{I_N} + v_{t,t}\mathbf{C}_t^{\neg g} + \sum_{k\neq t} v_{k,t}\mathbf{C}_k \tag{4.51}$$

To be able to compute the likelihood, we need the two quantities $\ln|\mathbf{\Sigma}_g|$ and $\mathbf{y}_g'\mathbf{\Sigma}_g^{-1}\mathbf{y}_g$. For optimization purposes, we need the gradients of these two functions with respect to $\mathbf{v}_t$ as well, or directly estimates of the zeros of said gradients. In all cases, as we have seen from (4.10) or (4.33), we will more or less directly need the effective numbers of parameters:

$$\gamma_{kt}^{(g)} := \frac{\partial \ln|\mathbf{\Sigma}_g(\mathbf{v}_t)|}{\partial \ln v_{kt}} \tag{4.52}$$

$$= v_{kt}\frac{\partial \ln|\mathbf{\Sigma}_g(\mathbf{v}_t)|}{\partial v_{kt}} \tag{4.53}$$

$$= v_{kt}\operatorname{Tr}\left[\mathbf{\Sigma}_g^{-1}\mathbf{C}_k^{\neg g}\right] \tag{4.54}$$

Likewise, as we saw in (4.16) and (4.36), we want to compute the effective model size:

$$m_{kt}^{(g)} := \frac{\partial \mathbf{y}_g'\mathbf{\Sigma}_g^{-1}\mathbf{y}_g}{\partial v_{kt}^{-1}} \tag{4.55}$$

$$= -v_{kt}^2\frac{\partial \mathbf{y}_g'\mathbf{\Sigma}_g^{-1}\mathbf{y}_g}{\partial v_{kt}} \tag{4.56}$$

$$= v_{kt}^2\mathbf{y}_g'\mathbf{\Sigma}_g^{-1}\mathbf{C}_k^{\neg g}\mathbf{\Sigma}_g^{-1}\mathbf{y}_h \tag{4.57}$$

The gain in complexity that we target comes from the fact that eventually the matrices $\mathbf{\Sigma}_g$ for all the g of group t are closely related, as they all differ from the matrix:

$$\mathbf{\Sigma}_{(t)} := \mathbf{I_N} + \sum_{k=1}^{K} v_{kt}\mathbf{C}_k \tag{4.58}$$

by a rank-one update, namely,

$$\mathbf{\Sigma}_g = \mathbf{\Sigma}_{(t)} - v_{tt}\mathbf{y}_g\mathbf{y}_g' \tag{4.59}$$

Note that it is out of question to merely approximate $\mathbf{\Sigma}_g$ with $\mathbf{\Sigma}_{(t)}$, as this would correspond, essentially, to models that predict $\mathbf{y}_g$ out of covariates that include $\mathbf{y}_g$ itself, and would result in a strongly biased likelihood that would overestimate v_{tt} at the expense of the other groups. However, all the computations that we use can exploit this relatedness to erase one degree of algorithmic complexity. To do so, we first need to introduce a few well-known central matrix identities.

Consider the block matrix:

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} \tag{4.60}$$

It is possible to eliminate the **U** block from the right, with block-row operations:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{U} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{0} & \mathbf{A} - \mathbf{U}\mathbf{V}' \end{bmatrix} \tag{4.61}$$

or, if **A** is invertible, from the other side:

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}^{-1}\mathbf{U} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U} & \mathbf{V}' \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{4.62}$$

As we only multiplied with triangular matrices with identity diagonals, these four expressions have the same determinant, thus:

$$|\mathbf{A} - \mathbf{U}\mathbf{V}'| = |\mathbf{A}| \cdot |\mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U}| \tag{4.63}$$

This can be applied to (4.59), giving:

$$|\mathbf{\Sigma}_g| = |\mathbf{\Sigma}_{(t)} - v_{tt}\mathbf{y}_g\mathbf{y}_g'| \tag{4.64}$$

$$= |\mathbf{\Sigma}_{(t)}| \left(1 - v_{tt}\mathbf{y}_g'\mathbf{\Sigma}_{(t)}^{-1}\mathbf{y}_g\right) \tag{4.65}$$

Here, the calculation of the determinant $|\mathbf{\Sigma}_{(t)}|$ has $O(N^3)$ complexity for each group t, and so has the inversion of the same. Given this, however, $\mathbf{y}_g'\mathbf{\Sigma}_{(t)}^{-1}\mathbf{y}_g$ can be computed in $O(N^2)$ for each g, giving a total complexity of $O(PN^2)$ to derive all the $|\mathbf{\Sigma}_g|$ from the $|\mathbf{\Sigma}_{(t)}|$.

Similar strategies can be applied for the other quantities of interest. Starting again from (4.61), we can eliminate the V' block too:

$$\begin{bmatrix} \mathbf{I} & -\mathbf{V}'(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{U} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} - \mathbf{U}\mathbf{V}' \end{bmatrix} \tag{4.66}$$

$$\begin{bmatrix} \mathbf{I} + \mathbf{V}'(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1}\mathbf{U} & -\mathbf{V}'(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1} \\ -\mathbf{U} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} - \mathbf{U}\mathbf{V}' \end{bmatrix} \tag{4.67}$$

Then, with a last block-diagonal multiplication:

$$\begin{bmatrix} \mathbf{I} + \mathbf{V}'(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1}\mathbf{U} & -\mathbf{V}'(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1} \\ -(\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1}\mathbf{U} & (\mathbf{A} - \mathbf{U}\mathbf{V}')^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.68}$$

Meanwhile, starting this time from (4.62) and continuing the elimination:

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{A}^{-1}\mathbf{U} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -(\mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}' \\ \mathbf{0} & \mathbf{I} \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{4.69}$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -(\mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}' \\ -\mathbf{A}^{-1}\mathbf{U} & \mathbf{I} + \mathbf{A}^{-1}\mathbf{U}(\mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}' \end{bmatrix} = \begin{bmatrix} \mathbf{I} - \mathbf{V}'\mathbf{A}^{-1}\mathbf{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{A} \end{bmatrix} \tag{4.70}$$

$$\begin{bmatrix} \mathbf{I} & \mathbf{V}' \\ \mathbf{U} & \mathbf{A} \end{bmatrix} \begin{bmatrix} (\mathbf{I}-\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1} & -(\mathbf{I}-\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}'\mathbf{A}^{-1} \\ -\mathbf{A}^{-1}\mathbf{U}(\mathbf{I}-\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1} & \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{U}(\mathbf{I}-\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}'\mathbf{A}^{-1} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{4.71}$$

We thus obtain two expressions for the inverse of our starting matrix. The identification of the blocks of these two expressions give us the so called Woodbury identity, when looking at the bottom-right block:

$$(\mathbf{A}-\mathbf{U}\mathbf{V}')^{-1} = \mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{U}(\mathbf{I}-\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}'\mathbf{A}^{-1} \tag{4.72}$$

In other parts of this work we'll more often use the variant of this equation with the sign swapped, namely:

$$(\mathbf{A}+\mathbf{U}\mathbf{V}')^{-1} = \mathbf{A}^{-1}-\mathbf{A}^{-1}\mathbf{U}(\mathbf{I}+\mathbf{V}'\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{V}'\mathbf{A}^{-1} \tag{4.73}$$

This new identity can also be applied to (4.59), giving:

$$\boldsymbol{\Sigma}_g^{-1} = (\boldsymbol{\Sigma}_{(t)} - v_{tt}\mathbf{y}_g\mathbf{y}_g')^{-1} \tag{4.74}$$

$$= \boldsymbol{\Sigma}_{(t)}^{-1} + \frac{v_{tt}}{1-v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1} \tag{4.75}$$

Since, given the inverse $\boldsymbol{\Sigma}_{(t)}^{-1}$, this can be computed in $O(N^2)$ per gene, this expression hold the key to computing all the derived quantities of $\boldsymbol{\Sigma}_g^{-1}$ that we need at a manageable computational cost. We will however avoid to explicitly compute and materialize $\boldsymbol{\Sigma}_g^{-1}$ which would still have a non-negligible $O(PN^2)$ memory complexity, and instead focus on deriving more specific formulas for the derived quantities we seek.

Namely, to evaluate the likelihood, we need $\mathbf{y}_g'\Sigma_g^{-1}\mathbf{y}_g$. Using (4.75), we now obtain:

$$\mathbf{y}_g'\boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g = \mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g + \frac{v_{tt}}{1-v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}(\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g)^2 \tag{4.76}$$

$$= \mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g\left(1+\frac{v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1-v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}\right) \tag{4.77}$$

$$= \mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g \cdot \frac{1}{1-v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} \tag{4.78}$$

Then, we turn our attention to the $\gamma_{kt}^{(g)}$ defined in (4.54). We can once again find an update formula by injecting (4.75):

$$\gamma_{kt}^{(g)} = v_{kt}\mathrm{Tr}\left[\boldsymbol{\Sigma}_g^{-1}\mathbf{C}_k^{\neg g}\right] \tag{4.79}$$

$$= v_{kt}\mathrm{Tr}\left[\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k^{\neg g}\right] + \frac{v_{kt}v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k^{\neg g}\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1-v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} \tag{4.80}$$

If $k \neq t$ (i.e., $k \neq G(g)$), $\mathbf{C}_k^{\neg g} = \mathbf{C}_k$ and this concludes the calculation:

$$\gamma_{kt}^{(g)} = v_{kt} \mathrm{Tr}\left[\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\right] + \frac{v_{kt}v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1 - v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} \tag{4.81}$$

Note that all terms here can be computed in $O(N^2)$ for a given g once $\boldsymbol{\Sigma}_{(t)}^{-1}$ has been pre-computed or at least pre-factored.

When $k = t$, however $\mathbf{C}_t^{\neg g}$ truly depends on g. While the time complexity of truly computing each $\mathbf{C}_t^{\neg g}$ and the $\gamma_{tt}^{(g)}$ from it is still acceptable, we once again would prefer not to do, mostly for memory footprint reasons. Therefore, we also want to expand (4.80) using:

$$\mathbf{C}_t^{\neg g} = \mathbf{C}_t - \mathbf{y}_g\mathbf{y}_g' \tag{4.82}$$

Which yields:

$$\begin{aligned}
\gamma_{tt}^{(g)} &= v_{tt} \mathrm{Tr}\left[\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\right] + \frac{v_{tt}^2\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1 - v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} \\
&\quad - v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g - \frac{v_{tt}^2\left(\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g\right)^2}{1 - v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} && (4.83) \\
&= v_{tt} \mathrm{Tr}\left[\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\right] + \frac{v_{tt}^2\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{C}_k\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1 - v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} \\
&\quad - \frac{v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g}{1 - v_{tt}\mathbf{y}_g\boldsymbol{\Sigma}_{(t)}^{-1}\mathbf{y}_g} && (4.84)
\end{aligned}$$

Therefore, in the case $k = t$, we get a formula similar to the $k \neq t$ case from (4.81) but with an additional corrective term to add. Note that this additional term comes with effectively no cost since it's a simple scalar that we already compute for other purposes.

Finally, we need the model sizes defined in (4.57). Since we have two occurrences of $\boldsymbol{\Sigma}_g^{-1}$ in this formula, we need to expand it using (4.75) twice, yielding a few more update terms:

$$
\begin{aligned}
m_{kt}^{(g)} &= v_{kt}^2 \mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_h && (4.85)\\
&= v_{kt}^2 \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_h \\
&\quad + 2 v_{kt}^2 \frac{v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g}{1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g} \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g \\
&\quad + v_{kt}^2 \left(\frac{v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g}{1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g} \right)^2 \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g && (4.86)\\
&= v_{kt}^2 \left(1 + \frac{v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g}{1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g} \right)^2 \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g && (4.87)\\
&= \frac{v_{kt}^2}{\left(1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g\right)^2} \cdot \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k^{\neg g} \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g && (4.88)
\end{aligned}
$$

Here again, for $k \neq t$, this completes the calculation:

$$
m_{kt}^{(g)} = \frac{v_{kt}^2}{\left(1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g\right)^2} \cdot \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g \tag{4.89}
$$

While for $k = t$, we still need to expand $\mathbf{C}_t^{\neg g}$:

$$
\begin{aligned}
m_{tt}^{(g)} &= \frac{v_{tt}^2}{\left(1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g\right)^2} \cdot \mathbf{y}_g' \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{C}_k \boldsymbol{\Sigma}_{(t)}^{-1} \mathbf{y}_g \\
&\quad - \left(\frac{v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g}{1 - v_{tt} \mathbf{y}_g' \boldsymbol{\Sigma}_{(g)}^{-1} \mathbf{y}_g} \right)^2 && (4.90)
\end{aligned}
$$

which ones again gives us the same formula than for $k \neq t$, with a corrective term that happens to be the square of the corrective term we already encountered for $\gamma_{tt}^{(g)}$.

We now have everything needed to compute the gradient of the log-likelihood for gene g that was defined in (4.46):

$$
\frac{\partial L_g(\sigma_g, \mathbf{v}_{.,t})}{\partial v_{kt}} = -\frac{1}{2} \frac{\gamma_{kt}^{(g)}}{v_{kt}} + \frac{1}{2} \sigma_g^{-2} \frac{m_{kt}^{(g)}}{v_{kt}^2} \tag{4.91}
$$

By summing over all the genes in group t, we can obtain the gradient of our weight-sharing objective function:

$$
\frac{\partial L_{(t)}\left((\sigma_g)_{G(g)=t}, \mathbf{v}_{.,t}\right)}{\partial v_{kt}} = -\frac{1}{2} \frac{1}{v_{kt}} \sum_{G(g)=t} \gamma_{kt}^{(g)} + \frac{1}{2} \frac{1}{v_{kt}^2} \sum_{G(g)=t} \sigma_g^{-2} m_{kt}^{(g)} \tag{4.92}
$$

And, if we globally approximate the gradient by holding γ and $\mathbf{m}$ constant, we can solve for the approximate optimum from the current trial point:

$$\hat{v}_{kt} = \frac{\sum_{G(g)=t} \sigma_g^{-2} m_{kt}^{(g)}}{\sum_{G(g)=t} \gamma_{kt}^{(g)}} \tag{4.93}$$

That is, the ratio of the "total effective model size" and the "total effective number of parameters" across all the models for the individual genes g in group t.

For an implementation that tries to maximize the use of automatic differentiation to keep expression simple, it is possible to re-express these updates as a function of gradients of some terms of the objective function as:

$$\hat{v}_{kt} = \frac{\sum_{G(g)=t} \sigma_g^{-2} \left(-v_{kt}^2 \frac{\partial \mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_g}{\partial v_{kt}} \right)}{\sum_{G(g)=t} \left(v_{kt} \frac{\partial \ln |\boldsymbol{\Sigma}_g(\mathbf{v}_t)|}{\partial v_{kt}} \right)} \tag{4.94}$$

$$= v_{kt} \frac{-\frac{\partial}{\partial v_{kt}} \sum_{G(g)=t} \sigma_g^{-2} \mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_g}{\frac{\partial}{\partial v_{kt}} \sum_{G(g)=t} \ln |\boldsymbol{\Sigma}_g(\mathbf{v}_t)|} \tag{4.95}$$

This provides an other practical computation avenue: compute, separately, the "total misfits" across all genes and the "total log determinants", with their gradients through automatic differentiation, then divide said gradients and use the ratio as a multiplicative update. Tricks to compute efficiently the leave-one-gene out quantities are still necessary for the forward pass implementation, but the additional complexity of the explicit formulas for $\boldsymbol{m}$ and γ can be avoided.

Meanwhile, we also have to set the individual gene noise levels σ_g. Starting from the same likelihood (4.46), we have:

$$\frac{\partial L_{(t)}\left((\sigma_g)_{G(g)=t}, \mathbf{v}_{\cdot,t} \right)}{\partial \sigma_g^2} = -\frac{N}{2} \sigma_g^{-2} + \frac{1}{2} \sigma_g^{-4} \mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.96}$$

And we can set:

$$\sigma_g^2 = \frac{\mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_g}{N} \tag{4.97}$$

Where the term $\mathbf{y}_g' \boldsymbol{\Sigma}_g^{-1} \mathbf{y}_g$ is non-trivial to compute, but we've already seen how to obtain it with efficient update formulas in (4.78).

Note that equivalently, we could have obtained this last update by also seeing σ_g^2 as the weight of a variance component that happens to be the whole $\boldsymbol{\Sigma}_g$, which would have the "model size"

$$m_g = \frac{\partial \mathbf{y}_g'(\sigma_g^2 \mathbf{\Sigma})^{-1} \mathbf{y}_g}{\sigma_g^{-2}} \tag{4.98}$$

$$= \sigma_g^4 \sigma_g^{-4} \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} \mathbf{\Sigma}_g \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.99}$$

$$= \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} \mathbf{\Sigma}_g^{-1} \mathbf{y}_g + \sum_k \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} v_{kt} \mathbf{C}_k^{\neg g} \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.100}$$

$$= \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} \mathbf{\Sigma}_g^{-1} \mathbf{y}_g + \sum_k \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} v_{kt} \mathbf{C}_k^{\neg g} \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.101}$$

$$= \left\| \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \right\|^2 + \sum_k \frac{1}{v_{kt}} m_{kt}^{(g)} \tag{4.102}$$

While not easier for computation, this expression allows one to see how the numerator in (4.97) decomposes into the contributions of the various groups $m_{kt}^{(g)}$, plus the contribution of the "variance component" $\mathbf{I_N}$, i.e. the noise, with $\left\| \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \right\|^2$ being the residual sum of squares of the model.

This last observation alone motivates finding an update formula to compute:

$$\left\| \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \right\|^2 = \mathbf{y}_g' \mathbf{\Sigma}_g^{-1} \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.103}$$

$$= \mathbf{y}_g' \mathbf{\Sigma}_{(t)}^{-1} \mathbf{\Sigma}_{(t)}^{-1} \mathbf{y}_g \cdot \left(1 + \frac{v_{tt} \mathbf{y}_g' \mathbf{\Sigma}_{(t)}^{-1} \mathbf{y}_g}{1 - v_{kt} \mathbf{y}_g' \mathbf{\Sigma}_{(t)}^{-1} \mathbf{y}_g} \right)^2 \tag{4.104}$$

$$= \left\| \mathbf{\Sigma}_{(t)}^{-1} \mathbf{y}_g \right\|^2 \cdot \left(\frac{1}{1 - v_{kt} \mathbf{y}_g' \mathbf{\Sigma}_{(t)}^{-1} \mathbf{y}_g} \right)^2 \tag{4.105}$$

4.5 Making predictions on new samples with CMK

For select tasks, including evaluating the predictive performance of CMK, we want to predict the unobserved expression value of a gene in a blind setting — that is, with the predicting method never using the known value. Since the model contain gene-specific pre-processing and parameters (genes have to be clustered, and each gene has its own σ_g^2 noise level), it is highly inconvenient to make predictions for a new gene. Therefore, we focus instead on making predictions of already seen genes, but for a yet unseen individual or sample. Since CMK only models the distribution of genes one at a time, the only scenario that can be tested is the case of predicting the value of gene expression for an individual when the expression values of all other genes are known, which can be dubbed "leave-one-gene-out" scenario.

To this end, given a new individual $n + 1$ and known gene g, in group $t = G(g)$ and knowing the expression $z_h, h \neq g$ of their other genes, we need the distribution of its expression Z_g. To obtain it, we start from the assumption that, along the observed expression $\boldsymbol{y}_g$ of the n previous individuals, the expressions for the target gene g and the other genes $h \neq g$ form vectors:

$$\tilde{\boldsymbol{y}}_g = \begin{bmatrix} \boldsymbol{y}_g \\ Z_g \end{bmatrix} \tag{4.106}$$

$$\forall h \neq g, \quad \tilde{\mathbf{y}}_h = \begin{bmatrix} \mathbf{y}_h \\ z_h \end{bmatrix} \tag{4.107}$$

where $\tilde{\boldsymbol{y}}_g$ has multivariate normal distribution:

$$\tilde{\boldsymbol{y}}_g = \mathcal{N}(\mathbf{0}, \sigma_g^{-2}\tilde{\boldsymbol{\Sigma}}_g^{-1}) \tag{4.108}$$

where:

$$\tilde{\boldsymbol{\Sigma}}_g = \mathbf{I_{n+1}} + \sum_k v_{kt} \sum_{\substack{G(h)=k \\ h\neq g}} \tilde{\mathbf{y}}_h \tilde{\mathbf{y}}_h' \tag{4.109}$$

From there, the distribution of Z_g given $\boldsymbol{y}_g$ can be derived as the conditional distribution of a coordinate of the joint multivariate normal vector $\tilde{\boldsymbol{y}}_g$. This is easily obtained given $\tilde{\boldsymbol{\Sigma}}_g^{-1}$, which in turn can be expressed from the block definition:

$$\tilde{\boldsymbol{\Sigma}}_g = \begin{bmatrix} \boldsymbol{\Sigma}_g & \sum_{h\neq g} v_{kt}\mathbf{y}_h z_h \\ \sum_{h\neq g} v_{kt}\mathbf{y}_h' z_h & 1 + \sum_{h\neq g} v_{kt} z_h^2 \end{bmatrix} \tag{4.110}$$

From this expression of the covariance, it can be obtained that the conditional mean of Z_g is:

$$\mathbb{E}\Big[Z_g | \boldsymbol{y}_g = \mathbf{y}_g\Big] = \left(\sum_{h\neq g} v_{kt}\mathbf{y}_h z_h\right)' \boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g \tag{4.111}$$

Naturally, we do not want to compute the $O(P)$ sum in the first factor for each of the P genes as this would have total complexity $O(P^2)$, so when we already know all of the true $(z_h)_{1\leqslant h \leqslant P}$ and want to perform leave-one-out prediction, we rewrite the conditional mean as:

$$\mathbb{E}\Big[Z_g | \boldsymbol{y}_g = \mathbf{y}_g\Big] = \left(\sum_{h=1}^{P} v_{kt}\mathbf{y}_h z_h - v_{tt}\mathbf{z}_g z_g\right)' \boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g \tag{4.112}$$

$$= \left(\sum_{h=1}^{P} v_{kt}\mathbf{y}_h z_h\right)' \boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g - z_g v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g \tag{4.113}$$

While this expression apparently depends on the true value z_g that we are predicting, the derivation precisely means that the dependencies on z_g cancel out. This formula can therefore be used to efficiently compute benchmark predictions for all the Z_g in turn.

If we want to perform such leave-one-gene-out prediction for several new samples and predict for each gene g a vector z_g, and knowing the other genes $\mathbf{z}_h$, the expression above is easily vectorized as:

$$\mathbb{E}\left[z_g | y_g = \mathbf{y}_g\right] = \left(\sum_{\substack{h=1 \\ k=G(h)}}^{P} v_{kt} \mathbf{z}_h \mathbf{y}'_h \right) \mathbf{\Sigma}_g^{-1} \mathbf{y}_g - \mathbf{z}_g v_{tt} \mathbf{y}'_g \mathbf{\Sigma}_g^{-1} \mathbf{y}_g \tag{4.114}$$

4.6 Parametrization of per-gene noise in CMK

Based on the introduction in 4, one may have expected our per-gene models to be more simply defined by:

$$y_g \sim \mathcal{N}(0, \tilde{\mathbf{\Sigma}}_g) \tag{4.115}$$

with

$$\tilde{\mathbf{\Sigma}}_g := \sum_k v_{kt} \mathbf{C}_k^{\neg g} + \sigma_g^2 \mathbf{I_N} \tag{4.116}$$

instead of the covariance introduced in 4.4:

$$\mathbf{\Sigma}_g = \sigma_g^2 \left(\sum_k v_{kt} \mathbf{C}_k^{\neg g} + \mathbf{I_N} \right) \tag{4.117}$$

This choice defines how we expect the distribution of the genes in the same target group to be related. With $\tilde{\mathbf{\Sigma}}_g$, all the covariances differ by adding a variable amount of spherical noise. With $\mathbf{\Sigma}_g$, the covariances are all scale versions of the same identical shape. There is actually no subtle modeling choice in this parametrization, the decision comes down to the fact that $\tilde{\mathbf{\Sigma}}_g$ is simply not a computationally practical option. The transformation:

$$\mathbf{\Sigma} \mapsto \sigma_g^2 \mathbf{\Sigma} \tag{4.118}$$

is trivial from the point of view of computing derived quantities:

$$|\sigma_g^2 \mathbf{\Sigma}| = \sigma_g^{2N} \cdot |\mathbf{\Sigma}| \tag{4.119}$$

$$(\sigma_g^2 \mathbf{\Sigma})^{-1} = \sigma_g^{-2} \cdot \mathbf{\Sigma}^{-1} \tag{4.120}$$

In contrast, it is much harder to derive $|\mathbf{\Sigma} + \sigma_g^2 \mathbf{I_N}|$ or $(\mathbf{\Sigma} + \sigma_g^2 \mathbf{I_N})^{-1}$. One possibility is to diagonalize $\mathbf{\Sigma}$ first:

$$\mathbf{\Sigma} = \mathbf{P}.\mathbf{Diag}(d_1, \ldots, d_N).\mathbf{P}' \tag{4.121}$$

In which case we can update e.g. the determinant as:

$$|\mathbf{\Sigma} + \sigma_g^2 \mathbf{I_N}| = \prod_i (d_i + \sigma_g^2) \tag{4.122}$$

and the inverse as:

$$(\mathbf{\Sigma} + \sigma_g^2 \mathbf{I_N})^{-1} = \mathbf{P}.\mathbf{Diag}\left(\frac{1}{d_1 + \sigma_g^2}, \ldots, \frac{1}{d_N + \sigma_g^2}\right).\mathbf{P}' \tag{4.123}$$

The complication of such an approach is that the transformation used to include σ_g^2 must be chained with the rank-one update of adding or removing a gene from the definition of $\mathbf{\Sigma}$. What we truly need is to compute e.g.

$$|\mathbf{\Sigma} - v_{tt}\mathbf{y}_g\mathbf{y}_g' + \sigma_g^2 \mathbf{I_N}| \tag{4.124}$$

as a function of some pre-computation on $\mathbf{\Sigma}$. However, there is to our knowledge no efficient way to update the diagonalization of $\mathbf{\Sigma}$ following a rank-one update to efficiently obtain the eigenvalues of $\mathbf{\Sigma} - v_{kt}\mathbf{y}_g\mathbf{y}_g'$.

It may be possible to compute the CMK objective efficiently by therefore doing the transformation in the reverse order: first, diagonalize a source matrix $\mathbf{\Sigma}$. Then, derive the diagonalization of all the variants with a variable $\sigma_g^2 \mathbf{I_N}$ term by updating the diagonal. Finally, apply the rank-one updates formulas to *these* variants to obtain the quantities of interest. While, at first glance, the computational complexity of such an approach seems to still work out to be $O(PN^2)$, the additional implementation challenge justified experimenting with the much simpler option from (4.117) first. Additionally, even at the same computational complexity, diagonalizing $\mathbf{\Sigma}$ as required with (4.116) is certainly more expensive in absolute computational cost that the simpler Cholesky decomposition that can be used with (4.117) (both determinant and inverse can be efficiently found from a Cholesky decomposition, but the latter cannot be updated under addition of a scalar multiple of the identity matrix as a diagonalization can).

4.7 Use of automatic differentiation for CMK

In Section 4.4, we developed at length how to compute what are effectively the gradients of our compound objective function (4.45). In the age of generalized automatic differentiation in machine learning, and with the availability of frameworks such as Torch, TensorFlow and JAX that we do use to implement CMK, this may look like an oddly outdated concern.

First, we want to highlight that nothing prevents the use of automatic differentiation to fit the CMK model. Even if one wishes to use the MacKay updates instead of a black-box gradient descent optimization, we've taken care of highlighting how the "effective number of parameters" and "effective model sizes" that we compute can be defined as gradients through (4.54) and (4.57).

The key to the efficiency of the algorithm is to make use of sharing of computations between the terms of (4.45). This can be done already when implementing the likelihood in complexity $O(PN^2)$ through the update formulas (4.65) and (4.78) only. Given such an implementation, a backward differentiation algorithm should generally not introduce any computation in the graph that would raise the algorithmic complexity, and the involved calculation of the $\gamma_{kt}^{(g)}$ and $m_{kt}^{(g)}$ can be spared. We would absolutely recommend such an approach for prototyping and experimenting with CMK, as we did ourselves at an early stage.

If a production implementation is needed however, the derivations of Section 4.4 mainly aimed to explicit the differentiation and rank-one updates in order to identify the intermediate calculated quantities that can be simplified and found re-occurring across all these formulas, such as the ubiquitous "rank-one update factor" $\left(1 - v_{tt}\mathbf{y}_g'\boldsymbol{\Sigma}_g^{-1}\mathbf{y}_g\right)^{-1}$. In our experience, the implementation that can be done by manually identifying simplification and re-occurring terms in equations leads to a more optimized computation graph than backward differentiation, which applies chain rules automatically but cannot "simplify" the resulting equations. The final result usually beat automatic differentiation in terms of speed and, more crucially, memory footprint. Therefore, while it should never be the first step of developing such models, we find that careful re-implementation of some differentiation steps is profitable once a project has reached sufficient stability to consider such optimizations.

4.8 Number of CMK groups

The CMK models depends on a choice of a number of groups, in which genes are initially set by k-means clustering. Cross-validation is a logical approach to choosing that number. Since the model has relatively few parameters — the regression weights between all pairs of genes are integrated out — it is rather hard to make the model over-fit with the hyperparameters that are left to optimize. If we think that each group has K parameters and PN/K observations, we expect to really experience problems when K^2 becomes comparable to PN. With our typical cases of $P \approx 20000$, $N \approx 1000$, we would therefore need to split the data in several thousands groups, each having only a few genes, to reach that point. In practice, we find that we are rather limited by how large a model we manage to fit at all, in terms of numerical and computational limits. Conversely, when genes are put into different groups, the model is still free to set the corresponding prior weight as large as it wants, and especially equal to the "self" prior weight given to genes of the same group as the target. Concretely, this means that the model always has the choice to disregard the distinction between two groups and "merge" them by setting all the prior weights to the same values. We therefore never lose any flexibility or information by making more groups.

Figure 4.1 shows the result of a cross-validation procedure. The performance drastically degrades with more than 100 groups; as we just discussed, this is not due to

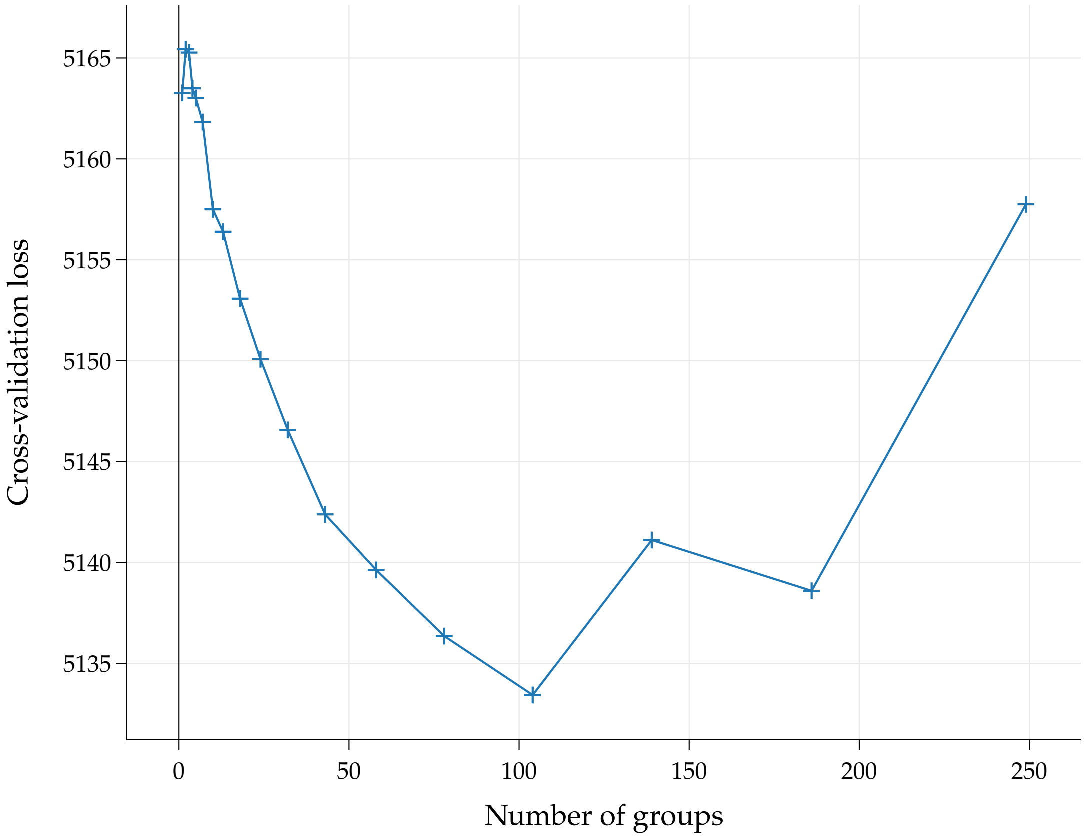

Figure 4.1: Predictive performance of CMK by number of groups.

over-fit, but rather is due to the fact that fitting larger models can sometimes fail. In that precise case, post-mortem analysis showed that with more groups, the estimation of the number of effective parameters for some cells can be numerically unstable, leading the algorithm to abort before full convergence. While this data is not shown here, occasional experiments with more groups up to 1000 have typically not shown much improvement over 100 groups when they do happen to converge.

5 Regularized joint linear models

While CMK was built as a collection of conditional models assembled in a heuristic way, it would be formally more satisfying to consider models in which we first try to build a joint model of expression of all genes, and then derive from it a set of conditional models. In that way, the parameters of the conditional have a formal reason to be coupled, as they are really a unique set of shared parameters in the joint. In such a situation, the CMK loss would have a formal justification as the pseudo-likelihood of the model.

We are still bound by computational constraints; we can only consider models for which simultaneously computing all conditional distributions is practical.

5.1 Joint multivariate normal

The first model of this family that we'll consider is the case where all the gene expressions $Y_1, \dots, Y_P$ of an individual jointly follow a Gaussian distribution:

$$\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{5.1}$$

where $\boldsymbol{\Sigma}$ represents the gene co-expression covariance matrix and is shared across all individuals or samples. For the rest of these derivations, it will be advantageous to instead parametrize this model with its precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$. As a function of all samples, the log-likelihood of $\boldsymbol{\Lambda}$ is therefore:

$$L(\boldsymbol{\Lambda}) = \sum_{i=1}^{N} \left(\frac{1}{2} \ln |\boldsymbol{\Lambda}| - \frac{1}{2} \mathbf{y}_i' \boldsymbol{\Lambda} \mathbf{y}_i \right) \tag{5.2}$$

or more compactly:

$$L(\boldsymbol{\Lambda}) = \frac{N}{2} \ln |\boldsymbol{\Lambda}| - \frac{1}{2} \mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\boldsymbol{\Lambda}) \tag{5.3}$$

In a hypothetical regular case, where we would have more samples $N > P$ than genes, the likelihood admits a well-defined maximum at:

$$\hat{\boldsymbol{\Lambda}} = \left(\frac{1}{N} \mathbf{Y}'\mathbf{Y} \right)^{-1} \tag{5.4}$$

Immediately, we run into the problem that with $N < P$ samples, such a model is ill-defined: by shrinking the variance $\mathbf{\Sigma}$ among any direction orthogonal to all observed expression vectors, $|\mathbf{\Sigma}|$ goes to zero while none of the $\mathbf{y}'\mathbf{\Sigma}^{-1}\mathbf{y}$ changes and the likelihood tends towards $+\infty$. Alternatively, the problem can be illustrated by seeing that the empirical gene covariance matrix $\mathbf{Y}'\mathbf{Y}$ becomes rank-deficient and does not have an inverse anymore. This last observation is the basis for generating practical methods: we are looking for ways to define "some kind of inverse" for $\mathbf{Y}'\mathbf{Y}$ that will allow us to recover the ability to make predictions.

The general way to achieve this is to perturb our model to make it well-defined again, and then look at how the solution behaves when we take this perturbation to zero. This gives a "solution" in the form of a limit of well-behaved perturbed solutions. What's a perturbation here? We can define roughly three overlapping types:

1. Free-form perturbation. Just take anything that looks sensible and can be computed efficiently and try it out.
2. Constraints. Limit your parameters to exclude the problematic parts of their domains, and see what happens when you progressively relax them.
3. Probabilistic perturbation. Add information to your model, typically under the form of priors.

Often, a perturbation does not belong cleanly to one of these categories, but can instead be justified or interpreted with any of these understandings.

Here, we'll start with a constraint-motivated approach: since our likelihood diverges to $+\infty$ as the precision matrix $\mathbf{\Lambda}$ grows in some direction, we can examine which is the best possible $\mathbf{\Lambda}$ of any given "size". For this, we need a function that quantifies how "big" $\mathbf{\Lambda}$ is. Since $\mathbf{\Lambda}$ is a symmetric positive definite matrix, there's no need to bound any entry by below, and all off-diagonal entries are bound by the diagonal ($\left|\mathbf{\Lambda}_{ij}\right| \leqslant \sqrt{\mathbf{\Lambda}_{ii}\mathbf{\Lambda}_{jj}}$), so we can look for any $c > 0$ for solutions satisfying the *total precision* constraint:

$$\mathrm{Tr}(\mathbf{\Lambda}) \leqslant c \tag{5.5}$$

This leads to a constrained optimization problem whose Lagrangian can be written as a function of a scalar Lagrange multiplier $\beta \geqslant 0$:

$$\mathcal{L}(\mathbf{\Lambda}, \beta) = \frac{N}{2} \ln |\mathbf{\Lambda}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\mathbf{\Lambda}) - \frac{\beta}{2}(\mathrm{Tr}(\mathbf{\Lambda}) - c) \tag{5.6}$$

(where we use $\frac{1}{2}(\mathrm{Tr}(\mathbf{\Lambda}) - c) \leqslant 0$ as our constraint) which has gradient:

$$\nabla_{\mathbf{\Lambda}}\mathcal{L} = \frac{N}{2}\mathbf{\Lambda}^{-1} - \frac{1}{2}(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P}) \tag{5.7}$$

leading to the solution:

$$\hat{\mathbf{\Lambda}}_{\beta} = \left[\frac{1}{N}(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})\right]^{-1} \tag{5.8}$$

As c goes to $+\infty$ and β to 0, $\hat{\mathbf{\Lambda}}_\beta$ naturally diverges, but we can take a closer look at how it diverges by inspecting:

$$\beta\hat{\mathbf{\Lambda}}_\beta = \beta\left[\frac{1}{N}(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})\right]^{-1} \tag{5.9}$$

$$= \left[\frac{1}{N}(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})\right]^{-1}\left[\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P} - \mathbf{Y}'\mathbf{Y}\right] \tag{5.10}$$

$$= N\left[\mathbf{I_P} - \underbrace{(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})^{-1}\mathbf{Y}'}_{\to\mathbf{Y}^+}\mathbf{Y}\right] \tag{5.11}$$

according to (16.19):

$$\xrightarrow[\beta\to 0]{} N\,(\mathbf{I_P} - \mathbf{Y}^+\mathbf{Y}) \tag{5.12}$$

To make predictions using the limit case, the multivariate normal kernel can then be generalized to a singular precision matrix $\mathbf{\Lambda}$ to replace the regular inverse covariance $\mathbf{\Sigma}^{-1}$. The kernel thus defined is therefore not normalizable anymore, but as long as the conditionals are well defined it still yields a usable leave-one-gene-out prediction method. Discarding the diverging scale, we can use for our limiting plug-in distribution:

$$\hat{\mathbf{\Lambda}} \propto \mathbf{I_P} - \mathbf{Y}^+\mathbf{Y} \tag{5.13}$$

Nicely, if we store such a matrix as the three components $\mathbf{I_P}, \mathbf{Y}^+, \mathbf{Y}$, this can be efficiently precomputed in $O(PN^2)$ time complexity and then stored in $O(PN)$ memory.

Now, given such a $\hat{\mathbf{\Lambda}}$, comes the question of efficient conditional derivation. Given a new vector $\mathbf{z}$, we known from (17.6) that the prediction for the coordinate j, understood as the conditional mean given all of the other coordinates, is given by:

$$\hat{z}_j = -\frac{1}{\hat{\mathbf{\Lambda}}_{jj}}\sum_{k\neq j}\hat{\mathbf{\Lambda}}_{kj}z_k \tag{5.14}$$

Crucially, because the regression weights $-\hat{\mathbf{\Lambda}}_{kj}/\hat{\mathbf{\Lambda}}_{jj}$ are ratios of entries in $\hat{\mathbf{\Lambda}}$, they are independent of the scale of the matrix $\hat{\mathbf{\Lambda}}$ and can therefore be consistently interpreted as point predictions despite the solution of the maximum likelihood problem being ill-defined. In other words, while the maximum likelihood estimate of the parameters does not exists as it lies outside the likelihood domain, the predictions made by using this estimator as a plug-in do converge. On the other hand, the predictive precision diverge towards $+\infty$, so the predictive distribution around $\hat{z}_j$ converges to a point distribution which makes no probabilistic sense: the model is catastrophically overconfident, and cannot give a measure of the accuracy of its own predictions.

To practically compute that point estimate, doing P sums naively with $O(P)$ terms in each would have prohibitive time complexity $O(P^2)$. Instead, if we want to efficiently estimate all the leave-one-gene-out predictions, we can re-express this as:

$$\hat{z}_j = z_j - \frac{1}{\hat{\mathbf{\Lambda}}_{jj}} \sum_k \hat{\mathbf{\Lambda}}_{kj} z_k \tag{5.15}$$

which does not actually depends on the true value z_j; and in vector form:

$$\hat{\mathbf{z}} = \mathbf{z} - \mathbf{Diag}(\hat{\mathbf{\Lambda}})^{-1} \hat{\mathbf{\Lambda}} \mathbf{z} \tag{5.16}$$

Note that $\mathbf{Diag}(\hat{\mathbf{\Lambda}})$ can be efficiently computed in $O(PN)$ given $\mathbf{Y}^+$, and $\hat{\mathbf{\Lambda}}\mathbf{z}$ in $O(PN^2)$ as:

$$\hat{\mathbf{\Lambda}}\mathbf{z} = \mathbf{z} - \mathbf{Y}^+ (\mathbf{Y}\mathbf{z}) \tag{5.17}$$

5.2 Uniform regularization

While section 5.1 shows that making predictions from an incompletely defined linear model is possible, for most purposes it is superior to include at least some basic regularization. While this regularization can be probabilistically motivated, in theory, any modification of the maximum likelihood estimate of the regular case can be used as a candidate method.

This can be achieved by adding a prior on the covariance and using as our covariance estimate the posterior mode instead of the maximum likelihood, or adding a constraint on the precision matrix. As we've seen, a natural candidate is then:

$$\hat{\mathbf{\Lambda}} = N(\mathbf{Y}'\mathbf{Y} + \beta\, \mathbf{I_P})^{-1} \tag{5.18}$$

for some value of $\beta > 0$. This estimator can be obtained, equivalently, as the maximum likelihood estimate of the precision matrix of a multivariate normal under a constrain on $\mathrm{Tr}[\mathbf{\Lambda}]$, or as maximum a posteriori estimate under a matrix-valued exponential prior distribution:

$$p(\mathbf{\Lambda}) \propto \exp\Big(- \frac{\beta}{2} \mathrm{Tr}(\mathbf{\Lambda}) \Big) \tag{5.19}$$

This leaves two difficulties in transforming this estimator into a practical prediction method.

First, to be able to make predictions with this precision matrix, we need efficient ways of computing $\mathbf{Diag}(\hat{\mathbf{\Lambda}})$ and $\hat{\mathbf{\Lambda}}\mathbf{z}$ for some vector $\mathbf{z} \in \mathbb{R}^P$. Once again we wish to avoid a naive $O(P^3)$ computation of the inverse, using an other time the Woodbury matrix identity from (4.72):

$$\hat{\mathbf{\Lambda}} = \frac{N}{\beta} \mathbf{I_P} - \frac{N}{\beta} \mathbf{Y}'(\beta\, \mathbf{I_N} + \mathbf{Y}\mathbf{Y}')^{-1} \mathbf{Y} \tag{5.20}$$

Here, the N by N matrix to invert can be factored in $O(N^3)$, then the product $(\beta\,\mathbf{I_N} + \mathbf{YY}')^{-1}\mathbf{Y}$ can be done in $O(PN^2)$, from which we can see that both $\mathbf{Diag}(\hat{\mathbf{\Lambda}})$ and $\hat{\mathbf{\Lambda}}\mathbf{z}$, computed for instance as:

$$\hat{\mathbf{\Lambda}}\mathbf{z} = \frac{N}{\beta}\mathbf{z} - \frac{N}{\beta}\mathbf{Y}'\left[(\beta\mathbf{I_N} + \mathbf{YY}')^{-1}\mathbf{Yz}\right] \tag{5.21}$$

can be obtained in $O(PN^2)$.

Then, given the ability of making such predictions for a value of β, we still need to choose such a value. A probabilistic approach could be to interpret the influence of β as a prior on $\mathbf{\Lambda}$, and optimize the marginal likelihood:

$$p(\mathbf{Y}|\beta) = \int p(\mathbf{Y}|\mathbf{\Lambda})p(\mathbf{\Lambda}|\beta)\,\mathrm{d}\mathbf{\Lambda} \tag{5.22}$$

Some analysts could be troubled here by the technical difficulty of integrating out analytically the s.p.d. matrix variable $\mathbf{\Lambda}$, but this can actually be done efficiently. See (17.6) for a derivation. However, even if the marginal likelihood can be computed, we've found empirically that we run into a case of marginal-conditional trade-off discussed more generally in (6.1) that makes the maximum likelihood β unsatisfying. Instead, we prefer to optimize β directly on some conditional measure of fitness. That is, separating $\mathbf{Y}$ into two sets of samples $\mathbf{Y}_1$ and $\mathbf{Y}_2$, we can choose β maximizing $p(\mathbf{Y}_1|\mathbf{Y}_2, \beta)$. Naturally this scheme lends itself to quite a few variations. We can use for some $K > 0$ a K-fold validation scheme, dividing $\mathbf{Y}$ into K blocks $\mathbf{Y}_1, \ldots, \mathbf{Y}_K$ then optimizing:

$$\prod_{k=1}^{K} p(\mathbf{Y}_k|(\mathbf{Y}_l)_{l\neq k}, \beta) \tag{5.23}$$

And in the limit one can use leave-one-out cross validation on the samples. For $i \in \{1, \ldots, N\}$ let $\mathbf{y}_i$ be the rows of $\mathbf{Y}$, and $\mathbf{Y}^{\neg i}$ the matrix $\mathbf{Y}$ with row i removed, we can then optimize:

$$\prod_{i=1}^{N} p(\mathbf{y}_i|\mathbf{Y}^{\neg i}, \beta) \tag{5.24}$$

In general, a low-fold cross-validation scheme has the advantage to be trivial to implement in a model-agnostic way, making it very easy to stack it on top of a prototype model implementation, but at the cost of typically K time the computational cost of a prediction for K-fold. Leave-one-out cross-validation is prohibitively slow without per-model optimization, and therefore require an extra effort; but once implemented typically takes closer to twice the load of a single prediction, making it particularly attractive when available.

Next we need to choose how we want to handle the elimination of $\mathbf{\Lambda}$ in each factor of the form $p(\mathbf{Y}_1|\mathbf{Y}_2, \beta)$. The two choices we've explored are:

1. Interpret β as a Lagrange multiplier expressing a constraint or a parameter of prior from which we optimize out $\mathbf{\Lambda}$ by maximum a posteriori estimation and plug-in prediction.

2. Interpret β as parametrizing a full-blown prior and analytically integrating out $\mathbf{\Lambda}$.

In the first case, we'll do a plug-in with the expression from (5.18), while in the second case the formulas would come from the properties of the matrix-t distribution explored in Section 19. Finally, there is an approximation to choose for the prediction:

1. we can use the full probabilistic objective $p(\mathbf{Y}_1|\mathbf{Y}_2, \beta)$, or

2. we can extract only the predictive mean $\mathbb{E}\left[\mathbf{Y}_1|\mathbf{Y}_2, \beta\right]$ and evaluate its predictive quality with some loss e.g. $\|\mathbf{Y}_1 - \mathbb{E}[\mathbf{Y}_1|\mathbf{Y}_2]\|^2$.

At first glance one could wonder why anyone would want to use the second, "less probabilistic" objective. The motivation lies in yet an other form of trade-off, which we will call the *bias-calibration trade-off*[1]. We can illustrate this on an example. Let's assume that we have some scalar observation X, which the best possible, ideal predictive method (the conditional of the "true" generative model) would be able to describe as coming from a normal distribution $\mathcal{N}(\mu, \sigma^2)$, and say that we have a predictive model that outputs a distribution $\mathcal{N}(\hat{\mu}, \hat{\sigma}^2)$. Then the expected log-likelihood of this predictive model can be decomposed as:

$$\begin{aligned}
\mathbb{E}\left[\ln \mathcal{N}(X|\hat{\mu}, \hat{\sigma})\right] &= -\frac{1}{2}\ln(2\pi) - \ln\hat{\sigma} - \frac{1}{2\hat{\sigma}^2}\mathbb{E}\left[\left\|X - \hat{\mu}\right\|^2\right] && (5.25)\\
&= \underbrace{-\frac{1}{2}\ln(2\pi)}_{\text{constant}} - \underbrace{\frac{1}{2\hat{\sigma}^2}(\mu - \hat{\mu})^2}_{\text{bias}} - \underbrace{\frac{1}{2}\left(\ln\hat{\sigma}^2 + \frac{\sigma^2}{\hat{\sigma}^2}\right)}_{\text{calibration loss}} && (5.26)
\end{aligned}$$

As can be seen, the log-likelihood measures both bias (a term that reaches its optimum of 0 if and only if $\hat{\mu} = \mu$) and calibration (a term that reaches its optimum of 0 if and only if $\hat{\sigma}^2 = \sigma^2$). Therefore a model that had a perfect $\hat{\mu}$ but a bad $\hat{\sigma}$ would end up with a bad likelihood. Concretely, this means a model that perfectly predict on average where the observation should be, but is either under or over-confident about its own performance in doing so. Do we want to always reject such a model? Certainly not! Such a model would give optimal point predictions, which is all that we ask for in many applications. In a case like above when we optimize a hyper-parameter β, the value that minimize the bias and the one minimizing the likelihood have no reason to be identical, meaning that optimizing the likelihood actually introduces extra bias in the predictive

[1]Not to be confused with the more common bias-variance trade-off, which typically concerns itself with how stable the point predictions are under perturbation of the training, re-sampling for non-deterministic predictions or generalization and does not address calibration at all.

mean in exchange for obtaining a correctly calibrated model. Whether this makes sense or not to do is application-specific.

Notably, consider again the precision estimator from (5.13) and the predictions we get from it: since it is only defined up to a scale, it carries no information about uncertainty and is "hopelessly miscalibrated". But it does give perfectly valid point predictions in many cases. If we tried to judge the usefulness of it from its predictive likelihood, which would simply be 0 here (the prediction is a point distribution with variance 0), we would think that the model is the worst possible and we would not at all get a good sense of how valuable the prediction method is.

Finally, it's worth observing that there can be compensations between the approximations that we're making. It is well known that using a single point estimate of a latent variable (as we propose to sometimes do here for $\mathbf{\Lambda}$) discards uncertainty, and, everything left equal, leads to overconfident predictions as opposed to integrating out over the full posterior of the latent variable. However, if the second stage of out model optimize hyperparameters against an objective that measures calibration, the hyperparameter choice can compensate for this known bias. Concretely, if using point estimate of latent parameters, we will obtain optimal parameters that create more regularization and will compensate the point-estimate overconfidence bias by shrinking estimates more.

If we seriously want to train properly calibrated models, however, and escape the trade-off between bias and calibration, it is important to have enough degrees of freedom in the hyper-parameters (or parameters of the prior) to adjust both the shrinkage towards a spherical distribution and the overall scaling. Therefore, if we want to optimize the actual likelihood and get calibration, rather than the estimator in (5.18) one will use the two-parameters variant:

$$\hat{\mathbf{\Lambda}} = \alpha\left(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P}\right)^{-1} \tag{5.27}$$

We can get a probabilistic interpretation or derivation of this more flexible estimator if we think of it as the mode of

$$\frac{N}{2}\ln|\mathbf{\Lambda}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\mathbf{\Lambda}) + \frac{\nu}{2}\ln|\mathbf{\Lambda}| - \frac{\beta}{2}\mathrm{Tr}(\mathbf{\Lambda}) \tag{5.28}$$

where $\alpha = (N+\nu)^{-1}$ which is our likelihood with a Wishart prior:

$$p(\mathbf{\Lambda}) \propto |\mathbf{\Lambda}|^{\frac{\nu}{2}}\exp(-\frac{\beta}{2}\mathrm{Tr}(\mathbf{\Lambda})) \tag{5.29}$$

In some cases, this additional parameter α can be optimized out analytically. If we perform a plug-in prediction on unobserved data $\mathbf{Y}_1$ with a posterior mode of the form $\hat{\mathbf{\Lambda}} := \alpha\mathbf{L}(\beta)$, the conditional likelihood becomes:

$$p(\mathbf{Y}_1|\mathbf{\Lambda}) = \frac{N}{2}\ln|\alpha\mathbf{L}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\alpha\mathbf{L}) \tag{5.30}$$

which can be seen to have an optimum in α at:

$$\alpha^* = NP\,\mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\mathbf{L})^{-1} \tag{5.31}$$

and re-injecting gives the likelihood:

$$p(\mathbf{Y}_1|\mathbf{L}) = \frac{NP}{2}\ln(NP) - \frac{NP}{2}\ln\mathrm{Tr}(\mathbf{Y}_1'\mathbf{Y}_1\mathbf{L}) + \frac{N}{2}\ln|\mathbf{L}| - \frac{NP}{2} \tag{5.32}$$

$$= -\frac{NP}{2}\ln\mathrm{Tr}(\mathbf{Y}_1'\mathbf{Y}_1\mathbf{L}) + \frac{N}{2}\ln|\mathbf{L}| + \mathrm{const} \tag{5.33}$$

Note how, compared to the original multivariate normal expression, we now have an extra ln around the trace. This different yet related to the multivariate-t distribution that could be obtained by integrating out a scale parameter, as seen in Section 17.6 and especially (17.156). The result is an expression that can be used as a valid objective function to measure the fit of an *unscaled* precision matrix **L**. Note how the likelihood is unchanged when **L** is multiplied by a scalar as the perturbations to the two terms compensate.

5.3 Heterogeneous regularization

Arguably one of the most important point of this work is the exposition of how vital per-gene regularization is when dealing with gene expression data. In the previous Section 5.2, we considered shrinking the precision matrix, and therefore the regression weights, through linear interpolation, *uniformly* through the whole matrix. However, the density of information in the empirical covariance $\mathbf{Y}'\mathbf{Y}$ and the various generalized inverses thereof actually varies drastically: simply put, some genes are noisier than others, and consequently the weaker correlations between noisier genes are less well estimated than those between “high noise” genes. We therefore want to shrink harder correlations between weaker genes. This principle is effectively already employed by the CMK model of Section 4.4, but in a somewhat obscure way: clustering genes based on Euclidian distance in the gene expression space has the effect of creating clusters of somewhat homogeneous signal strength. By then introducing hyper-variance parameters per group, we gave CMK the ability to effectively tune out these clusters of noisier genes and therefore the pollution they introduce in the model.

We can, as this section aims to demonstrate, reproduce the same effect in less elaborate models that are more reusable as building blocks of statistical procedures. We will start by thinking of estimators that achieve this goal, and work our way backwards to probabilistic constructions from which they can be derived.

As a starting point, the uniform regularization above revolved around the modification of the empirical covariance matrix represented by:

$$\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P} \tag{5.34}$$

for some $\beta > 0$.

But, surely, not all the areas of the matrix $\mathbf{Y'Y}$ are to be subjected to the same shrinkage. Where the genes are more reliably measured, the correlations may be well estimated already without any need to shrink them towards 0. Therefore, a simple modification that would yield a more flexible regularization would be a regularized covariance estimator of the form:

$$\mathbf{Y'Y} + \mathbf{D} \tag{5.35}$$

where $\mathbf{D} = \mathbf{Diag}(d_1, \ldots, d_P)$ is now a whole diagonal matrix of P per-gene shrinkage parameters. The simplest corresponding precision matrix estimator is then:

$$\hat{\mathbf{\Lambda}} = (\mathbf{Y'Y} + \mathbf{D})^{-1} \tag{5.36}$$

which can be used with prediction formulas such as (5.14).

There are various ways to interpret this expression in a probabilistic way. If we think of each d_j as a Lagrange multiplier as we did for β, we can see that $\hat{\mathbf{\Lambda}}$ is also the optimum of an optimization problem, but instead of a unique constraint:

$$\mathrm{Tr}\mathbf{\Lambda} \leqslant c \tag{5.37}$$

it is instead the bound maximum of the log-likelihood:

$$\mathcal{L}(\mathbf{\Lambda}) = \frac{N}{2}\ln|\mathbf{\Lambda}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y'Y\Lambda}) \tag{5.38}$$

under P constraints of the form:

$$\Lambda_{jj} \leqslant c_j \tag{5.39}$$

That is, we now have independent hyperparameters that can force the precision of any single gene to be lower. But since Λ describe a full joint distribution, forcing any gene to have a lower precision also effectively removes it from the model, by decreasing all the partial correlations and therefore regression weights involving this gene.

Practical computation will once again require an expression for $\hat{\mathbf{\Lambda}}$ after Woodbury (4.73) application, this time:

$$\hat{\mathbf{\Lambda}} = \mathbf{D}^{-1} - \mathbf{D}^{-1}\mathbf{Y'}(\mathbf{I_n} + \mathbf{YD}^{-1}\mathbf{Y'})^{-1}\mathbf{YD}^{-1} \tag{5.40}$$

This algebraic manipulation makes an interesting term appear, namely the N by N matrix $\mathbf{YD}^{-1}\mathbf{Y'}$. If $\mathbf{YY'}$ can be interpreted, up to scaling, as an estimator of the covariance of the expression of the N samples, then $\mathbf{YD}^{-1}\mathbf{Y'}$ appears as a weighted covariance estimator that gives more weights to genes with a larger d_j^{-1} i.e. smaller Lagrange multiplier i.e. looser precision constraint. Thus the $\mathbf{D}$ hyperparameters can also simply be seen as providing weights for the different genes to use when predicting any of the others; as d_j goes to $+\infty$ the expression of gene j gets asymptotically ignored completely by the model.

Naturally the interpretation with constraints can be replaced by an other based on priors, as $\hat{\mathbf{\Lambda}}$ can also be seen as the mode of the log-posterior:

$$\frac{N}{2}\ln|\mathbf{\Lambda}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y}'\mathbf{Y}\mathbf{\Lambda}) - \frac{1}{2}\mathrm{Tr}(\mathbf{D}\mathbf{\Lambda}) \tag{5.41}$$

where the prior on $\mathbf{\Lambda}$ is now formed by a product of P independent exponential priors $\propto \exp(-\frac{1}{2}d_j\Lambda_{jj})$ on each of the Λ_{jj}, of which the d_j are the rates, with a larger d_j thus corresponding to a more strongly zero-peaked prior that forces Λ_{jj} towards 0. Note that due to the positive definiteness constraint on $\mathbf{\Lambda}$ this also shrinks the off-diagonal coefficients: the log-determinant $\ln|\mathbf{D}|$ explodes towards $-\infty$ as the diagonal elements are shrunk towards 0 if the off-diagonal coefficients don't move.

To complete the method, we've seen that when calibration matter, estimators of the form $(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})^{-1}$ can exhibit an unwanted bias-calibration trade-off that lead us to introduce a more generic $\alpha(\mathbf{Y}'\mathbf{Y} + \beta\,\mathbf{I_P})^{-1}$. Naturally, we'd expect something similar to happen with our gene-wise shrinkage estimator $(\mathbf{Y}'\mathbf{Y} + \mathbf{D})^{-1}$. If we simply try to add a scalar scale α and use $\hat{\mathbf{\Lambda}} = \alpha(\mathbf{Y}'\mathbf{Y} + \mathbf{D})^{-1}$, we may run into the problem that since we're using different shrinkage strength for different regions of the matrix, we probably need different scales too. Therefore we suggest to rescale the precision estimator by introducing a whole diagonal matrix $\mathbf{S}$, and using this time:

$$\hat{\mathbf{\Lambda}} = \mathbf{S}(\mathbf{Y}'\mathbf{Y} + \mathbf{D})^{-1}\mathbf{S} \tag{5.42}$$

as our well-scaled estimator, where each gene now has two hyperparameters d_j and s_j.

We could wish to reiterate the process that lead us to derive the optimal scaling in (5.31) and optimize out $\mathbf{S}$ analytically. Unfortunately, the likelihood of a partition of the sample $\mathbf{Y}_1$ given a precision of the form $\hat{\mathbf{\Lambda}} = \mathbf{SLS}$ is a somewhat complicated function of $\mathbf{S}$:

$$\ln|\mathbf{S}| - \frac{1}{2}\mathrm{Tr}(\mathbf{Y}_1'\mathbf{Y}_1\mathbf{SLS}) \tag{5.43}$$

and attempts to find a zero of its gradient with respect to $\mathbf{S}$ reduces to finding the roots of a mostly arbitrary multivariate quadratic polynomial, which cannot be done analytically to the best of my knowledge. What can be expressed is the optimal s_j as a function of all the others $s_k, k \neq j$, but iteratively optimizing the coordinates one by one when we have thousands of them is not empirically very efficient. Therefore, for that gene-wised scaled estimator, the main reasonable approach is to optimize jointly $\mathbf{S}$ and $\mathbf{D}$ analytically; which gives a somewhat more complicated optimization task than the uniform case.

It is informative to compare the parametrization that we obtained here to what CMK offers. We end up with two parameters per gene: a "shrinkage" parameters that controls the weight of each gene when aggregating the weighted equivalent of the $\mathbf{Y}'\mathbf{Y}$ matrix, plus a "scale" parameter that allows to still adjust the marginal variance of each gene. CMK also has these two concepts, since it also features a weighted covariance matrix aggregation in (4.47) where the variances parameters v_{kl} play the role of our d_j^{-1}, but also requires on top some σ_g parameters for each gene g that are the analogues of our s_j. In short the models differ in that:

1. they are derived from different statistical frameworks; CMK being a collection of individual linear models while here we aim at expressing first a joint probability;
2. CMK constrains the weights to be equal inside each cluster;
3. CMK stack several such models since we have different sets of weights on all genes, one set for each cluster.

Otherwise, the way we fit the model parameters is different but conceptually very close. In CMK, we manipulate matrices that represent the joint distribution of the N samples, construct them from the data of all genes but one and evaluate the fit on the left-out gene. This is the exact orthogonal of the method that we suggest here, namely to interpret the parameters as determining the joint expression of the P genes and fitting them through cross-validation on the samples. Both approaches can be at least intuitively unified by imagining that we're really building a generative model of the total data matrix, as developed in Section 6, and then thinking of the two losses as two different kinds of partial pseudo-likelihoods with different choices of masking patterns.

6 Conclusion and a few reflections on model design

The exploration of the models presented up to now have revealed the difficulty of combining mathematical elegance in statistical modeling and performance. The best-performing method at our disposal, as Chapter 7 will explain more in detail, happens to be CMK, the clustered MacKay method. Yet this model has a few inelegant characteristics. It is a patchwork of several models, without generative modeling of how a gene ends up in one class or an other. It has a glaring redundancy in its asymmetry: the prior weight for the effects of gene of class k when predicting genes of class l can obviously not be independent from the prior when predicting class k from class l, yet we treat them as completely unrelated parameters. We optimize out most of the hyperparameters, without offering fully Bayesian inference on them. And yet, all of our attempts at "fixing" these traits of the model, most of which are not included in this work, have invariably led to poorer predictive performance.

In the end, commitment to building the best possible probabilistic predictions methods for a given data source proves to require balancing two complementary approaches to data science.

- On the one hand, the number of "ad-hoc" methods, understood as computational procedures that can give a plausible answer to a prediction task with some sort of rational motivation, is too large to be explored and rigorously compared on empirical grounds. Therefore, we regard as necessary to adopt a "probabilistic" design method: start with assuming as simple as possible a generative probabilistic model for the data at hand — that is, a joint distribution of all available data with no free variables or parameters, and as few knobs and hyperparameters to tune as possible; then formally *derive* predictions formulas. Approximations can be employed as needed as part of these derivations to obtain expressions that can be translated to computer programs with reasonable computational requirements. If the predictions turn out to be systematically flawed in a way we can understand, always prefer to address the problem by modifying the initial generative model, then iterate the derivation-approximation-implementation-evaluation cycle. This methodological choice has the goal of reducing as much as possible the number of "arbitrary" design decisions, to keep the complexity of the method as low as possible and, above all, reduce the number of method variants to try and compare to something manageable. If we just apply a large number of seemingly random transformations to our data depending on dozens of fudge factors, exploring all the alternatives simply becomes practically implausible.

- On the other hand, we recognize that sticking to methods that can mathematically be derived from a satisfying generative model in such a way as to obtain something that can be practically implemented *restricts our design space*. We admit that the best performing procedure, and the most useful to the end users, is all but guaranteed to contain some tweaks and idiosyncrasies that will have no good-faith theoretical explanation, and for which we won't be able to offer better a justification than "we tried to modify the procedure in this way and it just works better in our benchmarks though we don't quite understand why". We opt for a pragmatic approach that accepts and allows this fact of bioinformatic method development. However, this does not cancel out our first point about the value of theory. Instead, we try to keep the design under control to avoid building a statistical Rube-Goldberg machine, by always trying to fix the theory first before introducing ad-hoc corrections on top, and always trying to come up with explanations of why we *need* to depart from a simple theory-driven design by explaining what is it exactly we can't achieve within it; and in general reducing to a minimum the number of such alterations.

Following this, we can refine our vision of a radically theoretical design process in biostatistical method development. In this vision, the totality of the data, both known and unknown, is packed into a single random variable X, of which we only consider a single observation. The entirety of the modeling resides in the choice of a probability measure $p(\mathbf{X})\,\mathrm{d}\mathbf{X}$. Predictions tasks are specified by a pair of functions π_1 and π_2 (thus named because they are typically projections), meaning that we want to know the distribution of $\pi_2(X)$ knowing $\pi_1(X)$. Having chosen p already, obtaining it is a purely mathematical derivation of usable expressions of $p(\pi_2(\mathbf{X})|\pi_1(\mathbf{X}))\,\mathrm{d}\pi_2(\mathbf{X})$.

Under this formalism, we need not talk of samples, variables, training and testing data; nor of a training step or model fitting. Indeed, all of these concepts emerge from additional assumptions and approximations; purely mathematical design is free from any restriction to a particular framework.

Regarding the distinction of samples and variables, these are merely arbitrarily names for some dimensions of X. For modeling, we can start by considering the data as an arbitrarily 2-d tensor or matrix. Here, we could believe that there is a fundamental difference between these two axes; after all, we usually start the modeling by assuming, e.g., i.i.d. samples from a multivariate distribution. As we'll see, however, this is already an arbitrary modeling choice, corresponding to a particular mathematical form of p. Indeed, if we have gene expression for different genes and different individuals, gathered into a matrix X with N rows, one by individual, and P columns, one by gene, we could model it:

- by considering each row as being a new sample of a P-dimensional distribution representing the joint expression of all genes, and express p as:

$$p(\mathbf{X}) = \int \prod_{i=1}^{N} p(\mathbf{x}_i|\theta).p(\theta)\,\mathrm{d}\theta \tag{6.1}$$

- by considering each column as being a new sample a N-dimensional distribution representing the joint expression of a gene for all individuals:

$$p(\mathbf{X}) = \int \prod_{j=1}^{P} p(\mathbf{x}^j|\theta).p(\theta)\,\mathrm{d}\theta \tag{6.2}$$

Simple models may actually admit a representation of both forms. More complex ones may fit in neither.

Another distinction often made is between training and testing data. At first glance, it could look like this corresponds to our $\pi_1(\boldsymbol{X})$ and $\pi_2(\boldsymbol{X})$. However this would be incorrect because formally, $\pi_2(X)$ contains only the unobserved data, whereas what is typically called *testing data* also actually contains *observed* values, namely the testing covariates, predictors or features, which formally are inputs of the procedure and known parts of $\boldsymbol{X}$ just as the training data, and therefore belong in $\pi_2(\boldsymbol{X})$: testing predictors are parts of the data that the prediction method has knowledge of when making guesses for the testing target variables. Actually, the distinction between training data and testing predictors is less a formal and more a practical one: many methods involve a lengthy calculation on some part of the observed data, the "training", that can be done before integrating the rest of the observed data (the testing features) in a second step that is usually faster.

Finally, the fitting of a model is also a concept that is not part of the overall predictive framework, and instead only a detail of how a given prediction method work. Formally, there's only a single step, and that's approximately computing a conditional. Now, if the chosen generative model p can be written such that two halves of the data are independent given some latent variable, it can be natural to first compute a compact representation of the posterior of the latent variable given one half and then integrate the conditional of the other half over it. For instance, the combination maximum likelihood estimation followed by plug-in estimation can be seen under this light. Optimizing the likelihood on one part of the data can be understood as computing a point estimate approximation to the posterior of the latent variable; and "plugging in" this estimate can be interpreted as simply doing an integration over the resulting point distribution; as such the whole procedure can be understood as being an approximate conditional distribution evaluation. But in general, a prediction method can have any form of such a step (like obtaining posterior samples or fitting a parametric variational approximation) or none at all (if all latent variables can be integrated out, the conditional calculation may be expressed directly in terms of the observed data without any step of "optimizing", "fitting" or "sampling").

6.1 Pseudo-likelihoods and bags of conditionals

Despite the simplicity of deriving every method from a generative model of the whole data, we need to punctually and carefully depart from it for a variety of practical reasons. One that we've mentioned already, for instance, is how we may want to approximate the integrals over latent variable as high-dimensional integration is not generally efficiently possible. An other, deeper reason is that even for a single dataset, it may not be possible to find a single, all-purpose generative models for all our needs. Especially, we may encounter trade-off in designing our models.

- Marginal versus conditional trade-off: one model may be a better description of the whole distribution of the data, i.e. a better generative model, yet have conditional distributions that fit the data less well.
- Conditional versus conditional: one model may describe better a conditional of the data while an other model may describe better an other one.

In theory, these trade-offs should not exist, and should be taken as a hint that we can improve our overall model to better capture all aspects of the data. But in practice, computational requirements mean that we can not afford infinitely flexible models, and among the models that we can use in reality, trade-offs will appear. A useful mental model is to imagine that:

1. There exist an ideal generative model which describes the data perfectly, under all angles.
2. This ideal model is invariably too complicated for calculations to be practical.
3. Therefore, all of our work relies on models that are pragmatic simplifications of the ideal model.
4. Because this simplification only approximates the ideal model, there exist several approximations that are each good at recovering a particular aspect of the real ideal model.

In other words, "some models are somewhere useful" [Gelman], and if we want to get it right everywhere we will most likely have to accept to rely on multiple models, instead of losing ourselves in the search of grand unified theories.

For tasks such as eQTL discovery, we genuinely care about the ability of our models to describe leave-one-gene-out conditionals. Therefore, it make sense to eliminate some latent variables, not by setting them close to their posterior mode, that is optimizing the joint likelihood of the data; but some other functions such as a pseudo-likelihood of the model. If we accept that a model is a simplification of an ideal model meant to capture correctly its conditional distributions, then the "best" value of a latent variable may not be anymore the one that should, in a Bayesian interpretation, be closest to the "true value": if we don't have the best model to work with, a latent variable may not be a genuine parameters that we want to "find" or "estimate"; it's a knob that we want to tune to guest the best approximation to the conditional. For this reason, we may decide to set parameters to their maximum pseudo likelihood estimate rather that the joint likelihood estimate *even in cases where we can efficiently compute the latter*.

Furthermore, if we aim to predict variables from one another — what the conditionals actually capture — it is natural to use the conditionals derived from a joint model, but there is actually no obstacle to use conditionals derived from several, distinct joint distributions; with possibly different set of parameters optimized with different, partial, pseudo-likelihoods. By the same logic, the best set of conditional distributions should in theory be derived from the unique true joint, but in reality we can only approximate the conditionals of the ideal model, and the best approximation of the conditionals may be better described as being conditionals of distinct joint models, with the parameters capturing necessary trade-offs to approximate a particular conditional. Informally, we call such models "bags of conditionals"; a set of functions from which all predictions can be made, and something akin to a pseudo-likelihood can be defined, but which is not actually a consistent set of conditionals from the same unified joint model.

6.2 Single-point, a.k.a. plug-in approximations to marginals versus conditionals

Still on the pragmatic aspects, it also turns out that evaluating model conditionals, and therefore pseudo-likelihoods, is as a rule also much simpler than evaluating marginals. We can illustrate this phenomenon by comparing the ease with which the two can be simplified with single-point integral approximations.

The gist of such an approximation consists in replacing a distribution by a point distribution located at one of its mode (hopefully the only mode). That is, given a density p and a function f, where θ^* is the mode of p, we may approximate $p(\theta)\,\mathrm{d}\theta$ by a Dirac delta distribution $\delta_{\theta^*}(\theta)\,\mathrm{d}\theta$, resulting in:

$$\int p(\theta)f(\theta)\,\mathrm{d}\theta \approx \int \delta_{\theta^*}(\theta)f(\theta)\,\mathrm{d}\theta \approx f(\theta^*) \tag{6.3}$$

This approximation of course only get as good as p is concentrated around its mode in a region small enough compared to the variation of f to locally approximate f as constant; and it requires p to be normalized.

If we now consider the distributions of two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ that are conditional independent given a third, latent variable $\boldsymbol{\theta}$ that we want to integrate out, the joint distribution can be expressed as:

$$p(\mathbf{x}, \mathbf{y}) = \int p(\mathbf{x}|\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta} \tag{6.4}$$

The only well-normalized factor we can apply the single-point approximation to here is p, so with $\boldsymbol{\theta}^*$ the mode of p, one could in theory think to approximate:

$$p(\mathbf{x}, \mathbf{y}) \approx p(\mathbf{x}|\boldsymbol{\theta}^*)p(\mathbf{y}|\boldsymbol{\theta}^*) \tag{6.5}$$

The conditional distribution of $\boldsymbol{x}$ given $\boldsymbol{y}$, in turn, can be expressed as:

$$p(\mathbf{x}|\mathbf{y}) = \frac{\int p(\mathbf{x}|\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}}{\iint p(\tilde{\mathbf{x}}|\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}\,\mathrm{d}\tilde{\mathbf{x}}} \tag{6.6}$$

$$= \frac{\int p(\mathbf{x}|\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}}{\int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}} \tag{6.7}$$

This conditional is the ratio of two integrals, which seems no simpler. One could even imagine applying the prior single point approximation to each integral. But in that case, the obvious and correct answer is to instead consider:

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}}{\int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}} \tag{6.8}$$

as our normalized distribution, and then use the mode $\hat{\boldsymbol{\theta}}(\mathbf{y})$ of $p(\boldsymbol{\theta}|\mathbf{y})$ to approximate:

$$p(\mathbf{x}|\mathbf{y}) \approx p(\mathbf{x}|\hat{\boldsymbol{\theta}}(\mathbf{y})) \tag{6.9}$$

In real life, the quality of these two approximations is drastically different, to the point were the first is nonsense while the second is so extremely useful as to be completely routine. This can be understood in terms of how Dirac-like our latent distribution is. For the marginal case, the prior is typically a very wide distribution with all the integral mass being quite far away from the prior mode. For the conditional, as we condition on more data, the posterior becomes narrower and narrower, and is asymptotically very well approximated by a Dirac as we become more conditional. This constitutes an other element of the marginal-conditional trade-off:

- Marginal distributions $\int p(\mathbf{x}|\boldsymbol{\theta})p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}$, $\int p(\mathbf{x}|\boldsymbol{\theta})p(\theta)\,\mathrm{d}\boldsymbol{\theta}$, $\int p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{\theta}$ are hard to approximate by using a modal point; but
- Conditionals, i.e. ratios of such marginal integrals, are much easier to compute that the dividend or divisor involved separately.

Conceptually, it's like if the difficulty of computing the broad marginals "canceled out" when dividing two of them.

7 Results summary

Figure 7.1 shows the predictive performance of all the models discussed in the chapters above. The performance is evaluated on around 20,000 genes, each providing a data point for the methods. There is a huge spread in how predictable genes in the dataset are, and changing prediction methods cannot be expected to overcome that: noisy, low-signal genes stay noisy and hard to predict no matter the method used. Therefore, we chose to report quantiles of the prediction error over all genes: the median error, which can be interpreted as prediction performance on a typical gene; the first quartile for high-signal, easy to predict genes, and the third quartile for low-signals, hard to predict genes.

More precisely, the shorthand from worse (bottom) to best (top) performing models are as follow.

`peer/60r, peer/60rp, peer/60p, peer/60` These four models correspond to the variants of predictive PEER from Chapter 3. The suffix "r" indicates use of the re-estimation of per-gene precision described in this chapter, while the "p" suffix denotes use of the original covariance matrix as described in that chapter. The number 60 was chosen to match what has been selected in published eQTL pipelines to maximize eQTL discovery [Agu+17]. These models therefore provide a good representation of the predictive power of commonly used control methods. The difference between variants can be seen to be negligible compared to between-models variation.

`cv/peer` This is the first PEER variant, but where the number of factors has been selected by cross-validation. The final number of factors is closer to 100, but do not represent a local minimum as computational constraints prevented us to try larger numbers of factors.

`igmm/2` A Gaussian mixture model with just two components, introduced to test whether splitting genes in two learned clusters would show any improvement for one of the two. As can be seen, this provides no improvement over a linear baseline.

`cv/svd, cv/svd2` These two models are technically different implementations of the same theory developed in Chapter 2, and correspond to singular value decomposition based prediction with the number of factors optimized by cross-validation.

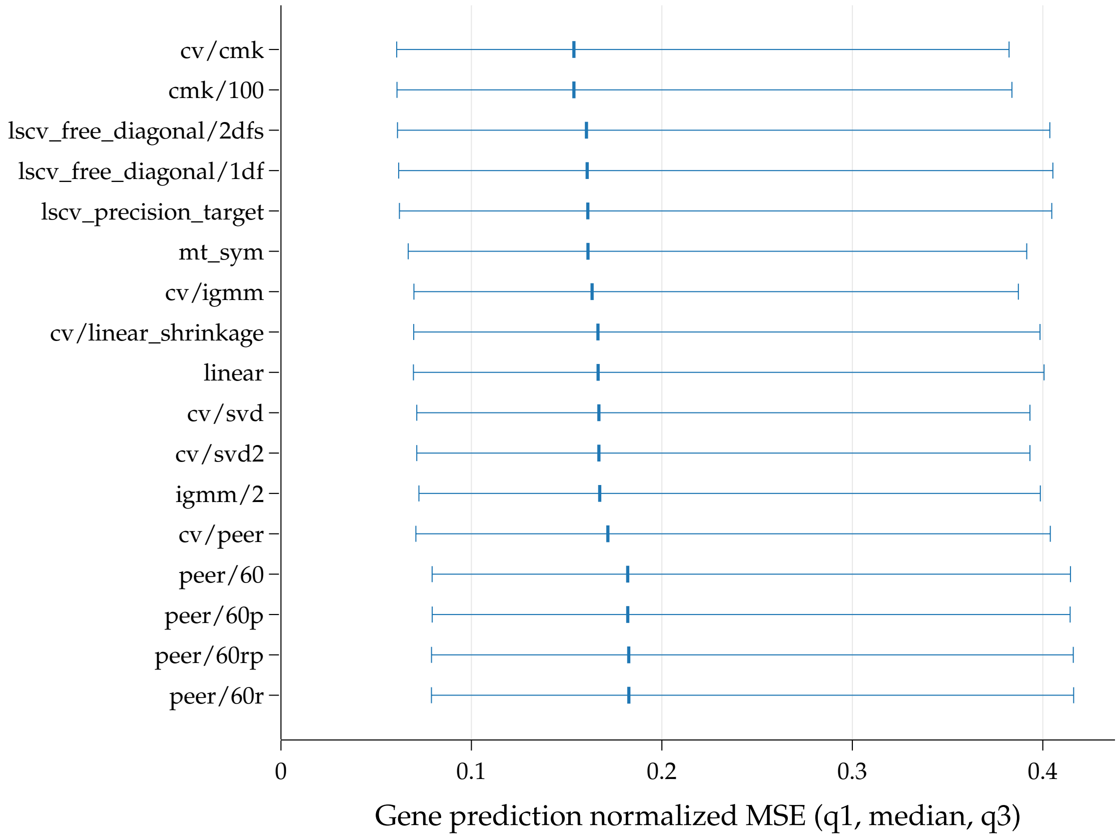

Figure 7.1: Performance of each prediction method across all genes. See main text for description of each method.

In contrast to PEER, there are no difficulties in fitting larger models, and the cross-validation can be described as successful. The performance closely match, on average, the linear baseline. A slight improvement can be seen on low-signal genes (third quantile).

`linear` This is our baseline linear prediction, with no regularization.

`cv/linear_shrinkage` Described in Chapter 5, this corresponds to a linear model modified to include homogeneous L2 regularization, meaning the same strength or regularization jointly when predicting all genes. The regularization strength is optimized by cross-validation.

`cv/igmm` This is also a Gaussian mixture, but with the number of classes optimized by cross-validation. The procedure ended up selecting around 20 groups, but is plagued with difficult fitting of models which does not exclude that larger models could theoretically perform better if reliable convergence was achieved.

`mt_sym` The shorthand stands for a symmetric matrix-t model. The principle is the same as described in Chapter 5 and conceptually very close to the `cv/linear_shrinkage` case, meaning that we look at a linear model with a uniform regularization, essentially a probabilistic multivariate Ridge regression. However, in this case, the high-dimensional gene expression precision matrix is integrated out using the theory from Chapter 19. Given how close the concepts are, the superior performance of the fully integrated variant is absolutely remarkable, making it by far the best method for its number of parameters (namely, two: the prior degrees of freedom of its Wishart prior and scale of the prior identity precision matrix).

`lscv_precision_target, lscv_free_diagonal/1df, .../2dfs` These are three variants of the heterogeneous joint regularization strategies described in Chapter 5. The `precision_target` variant corresponds to shrinkage strengths not optimized out individually, but set to be jointly proportional to the residual variance of a baseline model, meaning that higher-signal noises are shrunk less. The `1df` variant corresponds to each of the 20,000 shrinkage parameters optimized freely together, the `2dfs` variant has an additional trainable re-scaling parameter for each gene.

`cmk/100, cv/cmk` The cmk model from chapter 4, with 100 groups and with the number of groups optimized by cross-validation, which ended up being technically limited to around a 100 groups too.

The PEER method, introduced in Chapter 3, with a number of factors set to 60 or optimized by cross-validation but still limited to 100 by computational limits, performs worse than a linear baseline. This points to it being over-regularized, with the too limiting number of factors used ending up discarding important signal. In light of this, it may surprising that small number of factors are used in eQTL studies — with amount of eQTLs found decreasing on larger models. We attribute the difference to the fact that common use of PEER or SVD is often non-blind: factors trained on some genes and samples are then re-used as controls for these very same genes and samples. Naturally, this leaks information about the target gene inside the learned controls. By contrast, the benchmarks presented here are rigorously blinded, and over-fitting attributable to leakage cannot happen, which likely explain why we do not observe decreased performance past 50 or 60 factors.

Along the same idea, regularized methods where the shrinkage strength can be correctly optimized typically select very low regularization strength, at which point they are essentially identical to the linear baseline. Thus singular value based-methods from Chapter 2 do reach the baseline, as the lower computational cost makes it practical to use a large number of factors at which point the model is essentially identical to linear. The same phenomenon happens with homogeneous L2 shrinkage from Chapter 5. In the case of the SVD models, the additional regularization, if it does not help on the median gene, does seem to improve prediction for lower-signal genes.

Models with adaptive per-gene shrinkage are the first group to quite clearly break the ceiling, and the additional flexibility offered by CMK as described in Chapter 4 brings a clear amount of increased performance over it. The performance improvement is visible across the whole distribution of errors, showing that these models effectively break the regularization trade-off that made some models perform better only on low or high signal genes. In the next part, we will therefore naturally focus on how to integrate CMK into larger pipelines for eQTL calling, and explore whether we can take advantage of the large improvement over the low-factors PEER or SVD models that have been typically used in the field.

Part II

eQTL pipelines

8 Introduction

While Part I delved into how to best capture co-expression signal, we have yet to demonstrate how this can translate to meaningful applied gains. We therefore turn our attention back to integrating the results of this modeling effort into pipelines for expressive quantitative trait loci (eQTL) discovery.

In cohorts such as GTEx where quantitative expression data has been collected along with genotype information, it becomes possible to look for correlations between the presence or absence of a genetic marker in individuals, and increased or decreased expression of a gene. Most commonly these markers will be single-nucleotide polymorphisms (SNPs). As they are discovered de novo by whole genome sequencing of the participants, they tend to be mostly in intergenic or regulatory regions, and to be rare: most SNPs found will be seen in only few individuals, and consequently have very low sample minor allele frequency (MAF).

We focused only on cis-eQTLs, and therefore on markers found close to the transcription start site (TSS) of a gene. For these, we expect the typical mechanism for a mutation to affect gene expression to be the modification of the affinity of some DNA-binding protein involved more or less directly in the transcription process. Conveniently, this allow us to decide than any shared pattern of variation across several genes simultaneously can be ruled to not be in general attributable to a cis-eQTL, though it may result from a cis-eQTL affecting a gene which in turns regulate other genes. Our starting point is the remarkable eQTL pipeline used by the version 8 release of the genotype-tissue expression project (GTEx) in 2020 [GTE20]. However, most of the technical details of the pipeline implementation are to be found in the publication of the previous GTEx release[Agu+17]. We applied the same filtering used in these pipelines, considering markers within one megabase of each TSS, and only markers with MAF > 0.01. Further filters from that work are applied to select only expressed genes in whole blood, similar to what was done in Part I. In spite of this filtering, there are still thousands of markers to test for statistical association with expression for each gene.

The reason why correlation of expression between genes is so crucial while we wish to find associations between expression of individual genes and genotype comes from the necessary observational nature of the study. Far from a control experiment in which nothing else would change from one sample to an other that the genetic marker under test, a myriad of other factors will also randomly vary. For instance, from one sample to the next, the material collected will consist of a variable proportion of human cell types. Consequently, the expression of all the genes preferentially expressed in one cell type will appear to rise together from one individual to an other. The same kind of effect will be produced by many other measured or unmeasured factors: the time between sample collection and sequencing — though this is more relevant for tissues other than whole blood, but also many lifestyle factors (how did an individual lived? Were they smoking? What was their diet? Their exposition to pollutants?) and circumstances around collection (how recent was their last meal? Where they currently sick and fighting a flu infection?). These factors, taken together, explain how the expression of any gene can be so well predicted, in general, from other genes: we are implicitly exploiting all the correlations created at the cohort level by these hidden factors, and this is how we routinely obtain 90% prediction accuracy.

When comes the time of actually testing for genotype associations, all these other sources of variation, which represent one order of magnitude more variation, typically, than the genetic effects we're looking for, become a hindrance. If the cohort was so large or so carefully selected that we could assume that for each marker, unmeasured factors affecting expression were balanced between carriers and non-carriers of the markers, this could be ignored. But as these factors are so numerous, unknown, and sample size so limited by the difficulty of collecting gene expression information at scale, we expect on the contrary that for each marker, the samples collected from individuals with the alternative allele will, at random, have systematic differences in all these factors compared to the rest of the cohort with the reference allele; and even more so for rare variants where only a dozen individuals would be found carrying them. Thus, information coming from other genes is absolutely central to maintaining good discovery power, as it has been found already in the GTEx publications which opted for the solution of including a carefully chosen number of PEER factors inferred from the expression matrix as controls during the genotype association calling.

Testing out new transcriptomic processing methods involved three stages. First, we replicated the findings of the original release [GTE20; Agu+17] as closely as we could. Then, we tackled a few pipeline modifications that proved necessary to accommodate more flexible methods of pre-processing than the original approach, owing especially to the specificity of the classical multiple testing correction approach that was used. Finally, we could combine the more modular pipeline thus obtained with the clustered MacKay approach of Chapter 4 to build careful and meaningful comparisons of different co-expression modeling approaches with regard to the statistical power for eQTL discovery.

9 Fundamentals of QTL calling

Given a genotyped variant measured in a population and a quantitative variable of interest, measuring the association strength between the two is a rather straightforward task.

Let $x_i, i \in \{1, \ldots, N\}$ be a measured variant. Often, the variant is coded by its *dosage*, meaning that x_i has support $\{0, 1, 2\}$. In this coding, 0 denotes that the individual i is homozygous with the reference allele at this position, 1 that they are heterozygous with one allele being the reference and one allele being the alternative one, and 2 codes for a homozygous individual with twice the alternate allele.

The target variable is represented by a random variable Y_i, typically real-valued. Then, the association strength can be measured from a linear model:

$$Y_i = w x_i + \varepsilon_i \tag{9.1}$$

$$\varepsilon_i \sim \mathcal{N}(1, \sigma^2) \tag{9.2}$$

Where the entries ε_i are independent; equivalently,

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \sigma^2 I_N) \tag{9.3}$$

Needless to say, computing and obtaining a p-value for the null hypothesis $w = 0$ is a solved problem. However, complications will arise in our case when we start considering the problems of including covariates, pre-processing data, and multiple testing. For this reason, we will take the time to discuss the construction here, mainly to make obvious what assumptions are required, and provide a basis for further discussion about how to rescue violations of these assumptions.

The log-likelihood of such a model is:

$$L(w, \sigma^2) = \sum_{i=1}^{N} \left(-\frac{1}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} (Y_i - w x_i)^2 \right) \tag{9.4}$$

The maximum likelihood estimator of the parameters can then be deduced to be:

$$\hat{w} = \frac{\sum_{i=1}^{N} Y_i x_i}{\sum_{i=1}^{N} x_i^2} \tag{9.5}$$

$$\hat{\sigma^2} = \frac{1}{N} \sum_{i=1}^{N} (Y_i - \hat{w} x_i)^2 \tag{9.6}$$

Which we will shorten to the corresponding vector notations:

$$\hat{w} = \frac{\mathbf{y}'\mathbf{x}}{\|\mathbf{x}\|^2} \tag{9.7}$$

$$\hat{\sigma}^2 = \|\mathbf{y} - \hat{w}\mathbf{x}\|^2 \tag{9.8}$$

To then "call" a locus as being a QTL, the standard procedure is to perform classical hypothesis testing. Under the null hypothesis $w = 0$, we simply have $\boldsymbol{y} = \boldsymbol{\varepsilon}$. In that case, the linear regression decomposes the random vector $\boldsymbol{\varepsilon}$ into two deterministic orthogonal projections of itself:

$$\boldsymbol{y} = \boldsymbol{\varepsilon} = \hat{w}\mathbf{x} + \boldsymbol{r} \tag{9.9}$$

where

$$\hat{w} = \frac{\mathbf{x}'}{\|\mathbf{x}\|^2}\boldsymbol{\varepsilon} \tag{9.10}$$

and $\boldsymbol{r}$ are the residuals:

$$\boldsymbol{r} = \boldsymbol{y} - \hat{w}\mathbf{x} = \boldsymbol{\varepsilon} - \mathbf{x}\frac{\mathbf{x}'}{\|\mathbf{x}\|^2}\boldsymbol{\varepsilon} = \left(\mathbf{I_N} - \frac{\mathbf{x}\mathbf{x}'}{\|\mathbf{x}\|^2}\right)\boldsymbol{\varepsilon} \tag{9.11}$$

Since $\hat{w}$ and $\boldsymbol{r}$ are therefore two projections of $\boldsymbol{\varepsilon}$ orthogonal to each other, we have $\mathbb{C}\mathrm{ov}[\boldsymbol{r}, \hat{w}] = \mathbf{0}$. Crucially, because $\boldsymbol{\varepsilon}$ is a spherical multivariate Gaussian, these two components are Gaussian and decorrelated, and therefore independent, which is a property we will need later.

The estimator $\hat{w}$, as a linear projection of $\boldsymbol{\varepsilon}$, is therefore normally distributed with mean 0 and variance:

$$\mathbb{V}[\hat{w}] = \mathbb{V}\left[\frac{\mathbf{x}'}{\|\mathbf{x}\|^2}\boldsymbol{\varepsilon}\right] \tag{9.12}$$

$$= \frac{\mathbf{x}'}{||\mathbf{x}||^2}\mathbb{V}[\boldsymbol{\varepsilon}]\frac{\mathbf{x}}{||\mathbf{x}||^2} \tag{9.13}$$

$$= \frac{\mathbf{x}'}{||\mathbf{x}||^2}\sigma^2\mathbf{I_N}\frac{\mathbf{x}}{||\mathbf{x}||^2} \tag{9.14}$$

$$= \frac{\sigma^2}{||\mathbf{x}||^2} \tag{9.15}$$

The main obstacle we encounter in trying to obtain a useful test statistic from $\hat{w}$ is that since σ^2 is unknown, we cannot compute any quantile of the estimator's distribution.

Meanwhile, the residuals $\mathbf{r} = \boldsymbol{\varepsilon} - \hat{w}\mathbf{x}$ are normally distributed with mean 0 and the (rank-deficient) covariance matrix:

$$\mathbb{V}[\mathbf{r}] = \sigma^2 \left(\mathbf{I_N} - \frac{\mathbf{x}\mathbf{x}'}{\|\mathbf{x}\|^2} \right) \tag{9.16}$$

Let $\mathbf{v}_1 = \mathbf{x}\mathbf{x}'/\|\mathbf{x}\|^2$, and $\mathbf{v}_2, \dots, \mathbf{v}_N$ chosen so that $\mathbf{v}_.$ is an orthonormal basis of $\mathbb{R}^N$, and let $\mathbf{V}$ be the matrix whose columns are the $\mathbf{v}_j$; we have $\mathbf{V}'\mathbf{V} = \mathbf{V}\mathbf{V}' = \mathbf{I_N}$.

Since $\mathbf{V}$ is an orthonormal matrix, it preserves norms. Especially,

$$\|\mathbf{r}\|^2 = \mathbf{r}'\mathbf{r} = \mathbf{r}'\mathbf{V}\mathbf{V}'\mathbf{r} = \|\mathbf{V}'\mathbf{r}\|^2 \tag{9.17}$$

Therefore, $\|\mathbf{r}\|^2$ has the same distribution as $\|\mathbf{u}\|^2$ where $\mathbf{u} = \mathbf{V}\mathbf{r}$ is a multivariate normal of mean 0 and covariance matrix:

$$\mathbb{V}[\mathbf{u}] = \mathbf{V}'\mathbb{V}[\mathbf{r}]\,\mathbf{V} \tag{9.18}$$

$$= \sigma^2 \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \tag{9.19}$$

So $\mathbf{u}$ has independent normal components, with $u_1 = 0$ and $u_i \sim \mathcal{N}(0, \sigma^2)$ for $i > 1$. Therefore we obtain the distributions:

$$\frac{\|\mathbf{r}\|^2}{\sigma^2} \sim \chi^2(N-1) \tag{9.20}$$

$$\sim \mathrm{Ga}\left(\frac{N-1}{2}, \frac{1}{2} \right) \tag{9.21}$$

$$\frac{\|\mathbf{r}\|^2}{(N-1)\sigma^2} \sim \mathrm{Ga}\left(\frac{N-1}{2}, \frac{N-1}{2} \right) \tag{9.22}$$

Thus $\|\mathbf{r}\|^2$ provides us with the independent estimate of σ^2 that we need to normalize $\hat{w}$ and turn it into a testable statistic; given the distributions of $\hat{w}$ and $\|\mathbf{r}\|^2$ derived above and the independence between these two random variables,

$$T := \sqrt{N-1}\frac{\|\mathbf{x}\|}{\|\mathbf{r}\|}\hat{w} \tag{9.23}$$

$$= \left(\frac{\|\mathbf{x}\|\hat{w}}{\sigma} \right) \left(\frac{\|\mathbf{r}\|^2}{(N-1)\sigma^2} \right)^{-\frac{1}{2}} \tag{9.24}$$

$$\sim t(N-1) \tag{9.25}$$

according to (17.42).

Since the T thus constructed can be computed from known quantities (i.e., it is a statistic), but has a distribution which is independent of any unknown quantity (i.e., it is a pivotal quantity), the quantiles of T can be computed and used to measure if T is implausibly large under the null hypothesis. Especially, given an other independent $U \sim t(N-1)$, we can compute a p-value $p = P(|U| \geqslant |T|)$ using (17.74) that will be uniformly distributed under the null.

As can be seen, the assumption of $\boldsymbol{\varepsilon}$ being a spherical multivariate normal vector is crucial to this result. We relied on this assumption:

- In (9.9). If the input data violates this assumption, the residuals and the estimate of the effect size are in general either correlated, or uncorrelated but otherwise dependent.
- To derive the distribution of $\hat{w}$.
- To derive the distribution of $||\mathbf{r}||^2$.

Without this assumption, we face two different kinds of problems:

- If T is still an ancillary statistic, we may not know what distribution it has and how to compute its quantiles. This can however be dealt with by some empirical calibration scheme.
- In a more problematic way, T may not be a pivotal quantity anymore. In that case, even empirical calibration of T is generally impossible, since the null distribution of T changes as a function of some unobserved parameters. Much of the construction above has only one goal: removing the influence of the nuisance parameter σ^2. If the noise distribution is mischaracterized, some nuisance effects can still be present.

As an example of how less careful construction of a test statistic can pose challenges, under the null, we could have the intuition to estimate σ^2 by directly averaging over $\boldsymbol{y} = \boldsymbol{\varepsilon}$. We could take as our test statistic the apparently simpler expression:

$$\sqrt{N}\frac{||\mathbf{x}||}{||\boldsymbol{y}||}\hat{w} \tag{9.26}$$

Intuitively, this would also be a construction where the scale parameter σ cancels out, and could thus be a candidate for an ancillary statistic. However, in that case it is not clear what the distribution of this statistic is; if we fail to derive it, we could have to resort to simulating null draws to build empirical quantiles. One complication would be that, as shown in the standard case, the statistic's distribution may have a non-trivial dependency on N (even if the scaling is done correctly with respect to N, the tails of the distribution are polynomial with a tail index depending on N); which could require a different series of simulations for every different N, and so forth for every additional known parameter that may affect the ancillary distribution.

10 Inclusion of covariates and associated pitfalls

A particular reason for dwelling on this test statistic construction is to understand what happens when we *pre-process* the data fed to a QTL caller. As this is the delicate part of improving the calling procedure with better gene expression models, we need to careful consider how to perform the integration. To this end, Sections 10.1 and 10.2 will detail how covariates integration has traditionally be done in the field, based especially on our knowledge of existing callers [Sha12; Ong+16], and suggest a "plain T-statistic" variant that is easier to extend. Then, in sections 10.3 and 10.4, we propose corrections and extensions suitable to integrate more elaborate covariate control strategies.

10.1 Classical estimation with linear covariates

We now consider, that besides the possible predictor variant $\mathbf{x}$, whose effect on y we are interested in estimating, we have knowledge of an other covariate vector $\mathbf{c}$ that can also act as a predictor for y, be it an other variant or a variable of any other kind. In general, we would always observe some empirical correlation $|\mathbf{c}'\mathbf{x}| > 0$. Including $\mathbf{c}$ can influence our estimate in two conceptually different ways.

- If we have reason to think that $\mathbf{c}$ and $\mathbf{x}$ are correlated in the population we are sampling from (for instance, $\mathbf{c}$ is a population structure covariate, such as a binary variable coding for membership to an ancestry group or a genetic principal component, while $\mathbf{x}$ is a variant dosage variable), then we assume that the value of $\mathbf{c}'\mathbf{x}$ is meaningful. Then $\mathbf{c}$ is a confounder, and including $\mathbf{c}$ can radically change our estimation of effect size.

- If we think that $\mathbf{c}$ and $\mathbf{x}$ should not have any meaningful correlation (for instance, $\mathbf{c}$ is the age or reported sex of the individual and $\mathbf{x}$ is the dosage of an autosomal variant), we're assuming that $\mathbf{c}'\mathbf{x}$ has no interpretable meaning and is just an effect of sampling noise. Including $\mathbf{c}$ still matters because it explains away some of the variation in the target variable, which in turns make our estimation more reliable. If we do not include such a covariate, we indirectly rely on its effect averaging out to 0, which may be true in expectation but constitutes a source of variance in our estimate.

Let us have a look at how the estimate of $\hat{w}$ and the testing procedure is affected when including covariates. We directly add $K \in \mathbb{N}$ covariates $\mathbf{c}_1, \ldots, \mathbf{c}_K$ and call $\mathbf{C}$ the matrix of shape (N, K) made of these K columns, and the corresponding vector of linear weights $\mathbf{w}_c$

The log-likelihood of the model becomes:

$$L(w, \mathbf{w}_c, \sigma^2) = -\frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \|\boldsymbol{Y} - \mathbf{x}w - \mathbf{C}\mathbf{w_c}\|^2 \tag{10.1}$$

While we could simply formulate the linear model with a multivariate predictor matrix $\mathbf{X}$, this approach makes more obvious what the "effect" of $\mathbf{C}$ on the estimation is compared to the univariate case. With this goal in mind, we first isolate the partial maximum likelihood estimator of $\mathbf{w}_c$. Note that from the point of view of finding $\hat{\mathbf{w}}_c$, this is an usual linear regression, but with $\boldsymbol{y} - w\mathbf{x}$, depending on a yet unknown w, as the target, and the maximum likelihood estimator is:

$$\hat{\mathbf{w}}_c = \mathbf{C}^{+}(\boldsymbol{y} - \mathbf{x}w) \tag{10.2}$$

By re-injecting this solution into the log-likelihood, we can obtain the modified log-likelihood for w:

$$\begin{aligned} L(w, \sigma^2) &= -\frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \left\| \boldsymbol{y} - \mathbf{x}w - \mathbf{C}\mathbf{C}^{+}(\boldsymbol{y} - \mathbf{x}w) \right\|^2 && (10.3) \\ &= -\frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \left\| (\mathbf{I_N} - \mathbf{C}\mathbf{C}^{+})(\boldsymbol{y} - \mathbf{x}w) \right\|^2 && (10.4) \\ &= -\frac{N}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \left\| \mathbf{C}^{\perp}\boldsymbol{y} - w\mathbf{C}^{\perp}\mathbf{x} \right\|^2 && (10.5) \end{aligned}$$

where we have previously defined $\mathbf{C}^{\perp} = \mathbf{I_N} - \mathbf{C}\mathbf{C}^{+}$.

As can be read from this expression, the regression problem with plug-in of the maximum likelihood estimator of $\mathbf{w}_c$ now looks again like our original univariate regression in (9.4), but, crucially, *both* the target $\boldsymbol{Y}$ and the predictor $\mathbf{x}$ need to be replaced by the target residuals $\mathbf{C}^{\perp}\boldsymbol{Y}$ and the predictor residuals $\mathbf{C}^{\perp}\mathbf{x}$. This is a classical result that is also known in econometrics as the Frisch–Waugh–Lovell theorem [FW33][Lov63].

Alternatively, this can be seen to coincide with the log-likelihood of a model:

$$\boldsymbol{y} = w\mathbf{x} + \boldsymbol{\varepsilon} \tag{10.6}$$

where $\boldsymbol{\varepsilon}$ now follows a multivariate normal distribution with mean 0 and a non-spherical precision matrix $\sigma^{-2}\mathbf{C}^{\perp}$ instead of $\sigma^{-2}\mathbf{I_N}$. Note that this matrix is generally singular with precisions of 0 along some axes, meaning $\boldsymbol{\varepsilon}$ is improper with infinite variance along some directions. This effectively makes the model "blind" to what happens in the subspace spanned by $\mathbf{c}_1, \ldots, \mathbf{c}_K$.

Given this, the maximum likelihood estimator for w is:

$$\hat{W} = \frac{\boldsymbol{y}'\mathbf{C}^{\perp}\mathbf{x}}{\mathbf{x}'\mathbf{C}^{\perp}\mathbf{x}} \tag{10.7}$$

If $\mathbf{x}'\mathbf{C}^{\perp}\mathbf{x} = 0$, this is undefined, as in that case the predictor $\mathbf{x}$ lies completely within the space linearly generated by the covariates $\mathbf{c}_1, \ldots, \mathbf{c}_K$, and it becomes impossible to distinguish the effect of the predictor of interest from the effect of the covariates.

This allows us to see under which circumstances a program performing a linear regression will obtain a sensible estimate of the effect size w if inputs that have been pre-processed to remove the influence of $\mathbf{C}$ are given to it.

- If both $\boldsymbol{y}$ and $\mathbf{x}$ are replaced by $\mathbf{C}^{\perp}\boldsymbol{Y}$ and $\mathbf{C}^{\perp}\mathbf{x}$, the estimate is the same as if all variables were jointly regressed.
- If only $\boldsymbol{y}$ is replaced by $\mathbf{C}^{\perp}\boldsymbol{y}$ but $\mathbf{x}$ is left unchanged, the estimate will be subtly wrong, since it will be calculated as:

  $$\hat{W} = \frac{\boldsymbol{y}'\mathbf{C}^{\perp}\mathbf{x}}{\mathbf{x}'\mathbf{x}} \tag{10.8}$$

  As can be seen, the numerator is a correct measurement of how $\boldsymbol{y}$ and $\mathbf{x}$ align, accounting for $\mathbf{C}$, but the denominator is incorrect. Specifically, since

  $$\mathbf{x}'\mathbf{C}^{\perp}\mathbf{x} \leqslant \mathbf{x}'\mathbf{x} \tag{10.9}$$

  the effect of $\mathbf{x}$ will tend to be *systematically underestimated* when "correcting" only $\boldsymbol{y}$.
- Finally, and perhaps surprisingly, if only $\mathbf{x}$ is replaced by its regressed version $\mathbf{C}^{\perp}\mathbf{x}$ and $\boldsymbol{y}$ is left unchanged, the estimated effect size turns out to be actually correct.

This, however, only pertains to the point estimate $\hat{W}$. If we also want a sense of how significant that effect is, we also need to understand how manipulating inputs affects the estimation of the standard error of $\hat{W}$ and the computation of test statistics.

Under the null hypothesis $w = 0$, and $\boldsymbol{y} = \mathbf{C}\mathbf{w}_c + \boldsymbol{\varepsilon}$. While $\mathbf{w}_c$ is unknown, we only care about the projection through $\mathbf{C}^{\perp}$, and $\mathbf{C}^{\perp}\boldsymbol{y} = \mathbf{C}^{\perp}\boldsymbol{\varepsilon}$. Then, $\hat{W}$ is again normally distributed with mean 0 and variance:

$$\mathbb{V}\left[\hat{W}\right] = \frac{\sigma^2}{\mathbf{x}'\mathbf{C}^{\perp}\mathbf{x}} \tag{10.10}$$

Note how the denominator is smaller than $\|\mathbf{x}\|^2$ and therefore the variance higher than in the univariate case for the same value of σ^2. Since the variance is minimal when

$\mathbf{x}$ is completely orthogonal to the covariates $\mathbf{c}_1, \ldots, \mathbf{c}_K$, and explodes when $\mathbf{x}$ becomes linearly dependent on them, this can be understood as the effect of the "confusion" about whether the effect must be attributed to the covariates or the predictor of interest. Essentially, this shows how it is very hard to achieve confidence that a predictor has an effect on the response when said predictor is strongly confounded. On the other hand, if some variance in the target is really explained by the covariates, a real multivariate model will typically have a smaller value of the numerator σ^2. In general, adding a covariate to a model can thus result in both higher or lower variance in the estimate, depending on whether the covariate acts as a confounder that must be accounted for, or an additional source of information on the target that helps reduce the target noise.

Here again, it can be seen that the expression coincides with what would be obtained by feeding $\mathbf{C}^{\perp}\mathbf{x}$ as a modified covariate to a univariate linear regression, but this only conditional on the true value of σ^2. It is therefore crucial to understand how the estimation of σ^2, in turn, is affected.

The residuals can be expressed as:

$$\begin{aligned} \boldsymbol{r} &= (\mathbf{C}^{\perp}\mathbf{x})^{\perp}\mathbf{C}^{\perp}\boldsymbol{y} && (10.11) \\ &= (\mathbf{C}^{\perp}\mathbf{x})^{\perp}\mathbf{C}^{\perp}\boldsymbol{\varepsilon} && (10.12) \end{aligned}$$

that is, as the succession of two linear projections orthogonally to the covariates, then to the modified predictor, applied to $\boldsymbol{y}$ or $\boldsymbol{\varepsilon}$. Assuming that $\mathbf{x}$ and all the K covariates are linearly independent (if not, one can remove covariates until this holds for some smaller value K), it can be seen by the argument used in (9.19) that

$$\frac{\|\boldsymbol{r}\|^2}{\sigma^2} \sim \chi^2(N - K - 1) \tag{10.13}$$

Critically, this means that

$$\mathbb{E}\left[\|\boldsymbol{r}\|^2\right] = (N - K - 1)\sigma^2 \tag{10.14}$$

Therefore, to get an unbiased estimator of σ^2, $\|\boldsymbol{r}\|^2$ must be divided not by $N - 1$, but by the smaller $N - K - 1$, resulting in a higher estimate that the one obtained if K is forgotten. This finally gives an understanding of the more fundamental and hard to alleviate problem that occurs when pre-processing data to be used in a linear regression.

If covariates are regressed out of the target variable and predictors before these are used as inputs to a linear regression method, and the method does not offer a way to take this information into account, the linear regression will underestimate the amount of noise in the model, leading to misleading small effect size error estimates and p-values.

Essentially, the method is now "tricked" into believing that the squared residuals constitute $N - 1$ independent estimates of σ^2 to average over, or it is made unaware of the optimism bias created by using a maximum likelihood estimate of the residuals with respect to the covariates.

10.2 Empirical estimation of degrees of freedom

As established above, correct analysis of data that has been pre-processed requires knowing how many degrees of freedom have been removed from it, that is, how many linearly independent covariates have been regressed out from it. This can pose two problems depending on the pipeline:

1. If linear regression was indeed performed as a first step, building a statistical pipeline requires tools that correctly report this information, correctly require and use this information, and users who are aware of the need to propagate this information.

2. If some pre-processing was done that is not a linear regression, it is likely that the data will still behave in a somewhat similar fashion. Pre-processing will introduce some invariants, making the data somewhat redundant, and trying to apply an uncorrected association test to such data will underestimate noise as a consequence.

A possible solution to this category of problems can be obtained if we have the ability to simulate data under the null hypothesis. Fundamentally, the idea is that if we can simulate null data, and then apply the same pre-processing steps to it, we can directly estimate some unknown information about the null distribution of test statistics from it.

As we saw, with K covariates, the T-statistic for linear association has the distribution:

$$\sqrt{N-K-1}\frac{\|\mathbf{x}\|^2}{\|\boldsymbol{r}\|^2}\hat{W} \sim t(N-K-1) \tag{10.15}$$

We face the first technical problem that we may not be able to even compute that statistic if K is unknown. This however is easily worked around by using the scaled or plain T-statistic instead:

$$\bar{T} := \frac{\|\mathbf{x}\|^2}{\|\boldsymbol{r}\|^2}\hat{W} \sim \mathfrak{T}(N-K-1) \tag{10.16}$$

Indeed, the $\sqrt{N-K-1}$ factor is purely a convention about how we define the t-statistic and Student's t distribution that is driven by our preference to work with asymptotically standard distributions.

With this, given simulations under the null, we can compute null samples $\bar{T}_1, \ldots, \bar{T}_S$, and estimate the number of covariates K, now treated as an unknown parameter of our model, by using e.g. a maximum likelihood estimate.

If the pre-processing was really a linear regression, this sounds like a lot of effort when we could just directly go find what was the correct K that was used. The interest of the procedure, therefore, appears when we want to rescue the calculation of a sensible test statistic in cases where the pre-processing was a more custom operation. In that case, the idea is that the estimate of K will tells us that the data “looks like” it was obtained by regressing K covariates out of it, judging by the tail index of the association statistics generated under the null; and that this gives us a reasonable way to adjust our calculations to compensate for the effect of pre-processing. In other words, the estimate of K obtained when applying such a procedure to non-linearly corrected data can be interpreted as the *effective number of covariates* of the correction method.

10.3 Null simulation and inter-chromosomal controls

In Section 10.2, we've seen that we can recover meaningful association scores given association data from “null” scenarios. For this, we need to repeat our testing procedure on data that we now have no real associations. The existing approach from widely used software, meaning MATRIX eQTL [Sha12] and FASTQTL [Ong+16], relies on permutation controls. In that case, to simulate the wanted null data for a given gene, each individual gets assigned a randomly chosen expression value taken from all participants for that same gene, without replacement, thus creating expression values that have no association with the genotype data anymore but have the same empirical distribution over all participants.

Crucially, shuffling the expression values after covariates have been regressed out would yield an invalid procedure. Indeed, we've seen in Section 10.1 that correct estimation of the association strength requires linearly correcting *both* the genotype and expression together. If the expression is shuffled again after this step, it will acquire a random correlation with the covariates attributable to the permutation noise. The only exception is with a “permutation-invariant” covariate, meaning a constant one: data that was centered before will stay centered after permutation, so there is e.g. no need to re-fit the model intercept in theory. Similarly, quantile-normalization is preserved by shuffling. However all of the non-trivial covariates (e.g. sex, which is not permutation invariant unless permutation was done within each group) would require re-fitting.

For this reason, the normal procedure shuffles the raw expression values (meaning, after quantile normalization) first thing, and then re-applies the rest of the association testing procedure, starting with regressing out the covariates from the shuffled values.

Unfortunately, this approach is incompatible with any “local” covariate correction approach, in which the choice of covariates (or any model parameters in general) depends on the expression value. For instance, imagine a simple local correction approach consisting of:

- selecting the gene in the genome, which has the strongest association with the target gene (e.g. for a hemoglobin gene, this would very likely select an other hemoglobin gene, which would indeed likely be an excellent choice of control),

- then regress out of the target gene the part of gene expression variance explained (meaning, predictable from) by this selected gene.

Then, if we shuffle the target gene expression values, we obtain a gene expression value that is not at all correlated with the selected gene for the original target. For a linear method, this would not matter: only the number of linearly independent covariates matters, each costing one residual degree of freedom. However this does not hold anymore for any method that does any sort of adaptive shrinkage, in which covariates can be essentially turned on or off depending on the correlation with the target (this actually corresponds also to what we called effective degree of freedoms in our introduction to the MacKay method in (4.33)). As a result, if the correction procedure has a variable number of degrees of freedom depending on that association, they would be badly underestimated by this procedure. We could think of re-executing the selection procedure too, but unfortunately, shuffling will generate a gene that has no real link with the original expression and will therefore badly correlate with *all* genes.

Essentially, we are blocked by the fact that for the null simulation procedure to be valid with local correction methods, we need an expression vector that "looks like" a real gene, but shuffling does not produce one.

Our proposed solution to this problem is simple: we propose for the null simulation part of the process to simply take the real expression, *without shuffling*, of genes situated on an other chromosome than the gene of interest. By definition, such a gene may have no association to the genotype being tested (which consists of variants in the vicinity of the transcription start site of the target gene) that can validly be interpreted as a cis-eQTL effect. Figure 12.2 in Chapter 12 shows how the two approaches to null simulation compare, after including steps for multiple testing adjustments which are the topic of Chapter 11.

10.4 Extending covariate control for CMK-like models

Ultimately, our objective is to perform association testing that achieves better power by benefiting from better modeling of what expression data should look like beside the genetic effect. In Section 10.1, we've seen than in the linear case, and therefore likely in general, it is not enough to only take the residuals of the expression after a model has been fitted to it. The associations with the original genotype would in that case be biased. More specifically, the genotype may have some accidental variation that correlates with what has been removed from the expression, that will now make it look like less correlated with the expression where this variation has been forcefully eliminated.

We must therefore find out for other models how we can build an equivalent to the step of "also regressing out covariates from genotype" that needs to be done in the linear case. Fortunately for us, this works out quite well for random effects, now turned into mixed effects, Gaussian-linear models, and this happens to be a family of models large enough to encompass some of the interesting models of Part I. An extended discussion of strategies to assess significance in linear mixed models is out of the scope of this work, and starting points to this topic are widely available [Luk17]. For our purposes, what is particularly crucial is that we do not exactly fit mixed-effects models; we optimize ahead of time some prior parameters for the regression of expression on the values from other genes, and then extend it without fine-tuning the learned hyperparameters again. Combined with the need for very fast evaluation given the number of associations, this guides and already constrains our design to the following.

Let us assume that we have a random vector $\boldsymbol{y}$ on $\mathbb{R}^n$ (n observations of a variable in several samples) and that we have previously established it to be well modeled as coming from a multivariate normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$. In Chapter 4, we've seen that this naturally occurs when assuming that $\boldsymbol{y}$ is a linear combination of predictors plus a Gaussian noise, when the weights are themselves drawn from various independent normal priors. To add a fixed effect w of a predictor $\mathbf{x}$, we can modify this model as:

$$\boldsymbol{y} = w\mathbf{x} + \alpha\boldsymbol{\varepsilon} \tag{10.17}$$

$$\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}) \tag{10.18}$$

where α is as additional parameter to represent the relative weight of the residuals, as we would expect it to shrink compared to the same model without fixed effects. Of course, we would ideally fit jointly the parameters of $\boldsymbol{\Sigma}$ and w, thus avoiding that scaling parameters altogether; but this is simply not an option for efficient large-scale testing of a dozen thousands genetic variants per expression phenotype.

The log-likelihood of the model on an observation $\mathbf{y}$ can then be written:

$$\mathcal{L}(w, \alpha) = -\frac{n}{2}\ln\alpha - \frac{1}{2\alpha}(\mathbf{y} - w\mathbf{x})'\boldsymbol{\Sigma}^{-1}(\mathbf{y} - w\mathbf{x}) + \text{const} \tag{10.19}$$

We would like to obtain an expression akin to (10.5) from the fixed effect of covariates case. This is achieved by considering any factorization of $\boldsymbol{\Sigma}$ that gives a square-transpose root of it, for instance a Cholesky:

$$\boldsymbol{\Sigma} = \mathbf{U}'\mathbf{U} \tag{10.20}$$

where $\mathbf{U}$ is upper triangular, which in turns gives the upper-lower factorization:

$$\boldsymbol{\Sigma}^{-1} = \mathbf{U}^{-1}\mathbf{U}^{-1\prime} \tag{10.21}$$

where the inverse is obtained efficiently by triangular solving.

With this, we can rewrite the likelihood:

$$\begin{aligned}
\mathcal{L}(w, \alpha) &= -\frac{n}{2}\ln\alpha - \frac{1}{2\alpha}(\mathbf{y} - w\mathbf{x})'\mathbf{U}^{-1}\mathbf{U}^{-1\prime}(\mathbf{y} - w\mathbf{x}) + \text{const} &(10.22)\\
&= -\frac{n}{2}\ln\alpha - \frac{1}{2\alpha}\left(\mathbf{U}^{-1\prime}\mathbf{y} - w\mathbf{U}^{-1\prime}\mathbf{x}\right)'\left(\mathbf{U}^{-1\prime}\mathbf{y} - w\mathbf{U}^{-1\prime}\mathbf{x}\right) + \text{const} &(10.23)\\
&= -\frac{n}{2}\ln\alpha - \frac{1}{2\alpha}\left\|\mathbf{U}^{-1\prime}\mathbf{y} - w\mathbf{U}^{-1\prime}\mathbf{x}\right\|^2 + \text{const} &(10.24)
\end{aligned}$$

and we have obtained our analog of (10.5) where the inverse square root of $\mathbf{\Sigma}$, $\mathbf{U}^{-1\prime}$, plays the role of the orthogonal projection to the covariates $\mathbf{C}^{\perp}$. Since $\mathbf{C}^{\perp}$ is an orthogonal projection, and therefore both an idempotent map and symmetric, $\mathbf{C}^{\perp} = \mathbf{C}^{\perp}\mathbf{C}^{\perp\prime}$, so on top of identifying $\mathbf{C}^{\perp}$ with $\mathbf{U}^{-1}$, we can just as well say that its analogue is $\mathbf{U}^{-1}\mathbf{U}^{-1\prime} = \mathbf{\Sigma}^{-1}$. Therefore, we can interpret the precision matrix $\mathbf{\Lambda} := \mathbf{\Sigma}^{-1}$ as a "soft", less discretized, alternative to $\mathbf{C}^{\perp}$. In directions where the random effect predictors with large prior weights have a lot of variance, $\mathbf{\Sigma}$ is large, and $\mathbf{\Lambda}$ closer to zero; so the effect of applying $\mathbf{U}^{-1\prime}$ can be understood, instead of a zeroing out of any variation in the space spanned by the covariates as does $\mathbf{C}^{\perp}$, rather as a shrinking in all directions according to their importance in the fitted random effect model.

This observation suggests a simple association testing procedure: apply the $\mathbf{U}^{-1\prime}$ transform to both the expression and the covariates, and then proceed with the normal procedure described in Chapters 9 and present. Our estimator of the slope, for instance, will be:

$$w^* = \frac{\left(\mathbf{U}^{-1\prime}\mathbf{y}\right)'\left(\mathbf{U}^{-1\prime}\mathbf{x}\right)}{\left(\mathbf{U}^{-1\prime}\mathbf{x}\right)'\left(\mathbf{U}^{-1\prime}\mathbf{x}\right)} = \frac{\mathbf{y}'\mathbf{\Lambda}\,\mathbf{x}}{\mathbf{x}'\mathbf{\Lambda}\,\mathbf{x}} \tag{10.25}$$

While we may form our test statistic as before, we of course may not conclude that it will follow a t-distribution with a known number of degrees of freedom. But as we've seen in Section 10.2, provided that we are content with the assumption than the plain T-statistic that we may calculate will still behave like *some* kind of plain T-statistic for an unknown number of degrees of freedom ν, we can infer a value for ν by maximum likelihood estimation on simulated negative (null) data. This is especially reasonable in light of the interpretation that the random effect controls are little more than a "smooth" version of fixed effect controls, and that we expect them to interpolate between harder forms of control that fully regress out some dimensions of the input. Essentially, while adding a fixed-effect covariate removes exactly one degree of freedom, adding a random-effect covariate should remove from 0 to 1 degrees of freedom as the prior on the effect goes from very narrow (thus forcing the latent effect to 0, same as excluding the covariate from the model) to very wide (thus not constraining the effect size at all and producing the same effect as a fixed-effect model).

11 Multiple testing

By far one of the most challenging aspect in interpreting the results of linear association tests in the context of QTL calling is adjusting for the massive number of variables tested. When considering a window of ±1 Mb around a site in the human genome, and including variants with a minor allele frequency of more than 0.01, we routinely obtain around $P = 10\,000$ variables to test. Briefly, in the case of frequentist hypothesis testing, if we perform 10 000 tests for variants that do not have any effect, and if these tests are independent, then we we would expect around 100 of these tests to be significant at the 0.01 level. Therefore, it would be completely normal to observe a large number of failing tests for a gene that has actually no eQTL at all. If we want to answer a question such as "does this gene has an eQTL?", we therefore need to be more stringent.

Now, if we were really doing 10 000 independent tests, there are well-known strategies to analyze a set of p-values and draw conclusions from them, such as the Bonferroni or the Benjamini-Hochberg strategies. Our problem is that we are very much *not* doing many independent tests when testing for many variants to be eQTLs for a given gene.

Let us look at the effect size estimates first. If we are testing two variants $\mathbf{x}_1$ and $\mathbf{x}_2$, we are computing maximum likelihood estimates based on the same response $\boldsymbol{y}$ as:

$$\hat{w}_1 = \frac{\boldsymbol{y}'\mathbf{x}_1}{\|\mathbf{x}_1\|^2} \tag{11.1}$$

$$\hat{w}_2 = \frac{\boldsymbol{y}'\mathbf{x}_2}{\|\mathbf{x}_2\|^2} \tag{11.2}$$

Under the null where $\boldsymbol{y} = \boldsymbol{\varepsilon}$ is a spherical multivariate normal, $\hat{w}_1$ and $\hat{w}_2$ are independent, if and only if, $\mathbf{x}_1'\mathbf{x}_2 = 0$.

We immediately see from this two reasons why "10 000 independent tests" is not even remotely a reasonable way to think about our QTL estimates.

1. Since we typically have $N \leqslant 1000$ samples, it is mathematically not even possible for 10 000 covariates $\mathbf{x}_1, \ldots, \mathbf{x}_P$ to exist and be each orthogonal to each other. Far from that, assuming that the covariates are normalized and that, at most, $\mathbf{x}_1$ to $\mathbf{x}_N$ are linearly independent, then there exist a linear combination such that

$\mathbf{x}_{N+1} = \sum_{j=1}^{N} \lambda_j \mathbf{x}_j$, and then:

$$\hat{w}_{N+1} = \sum_{j=1}^{N} \lambda_j \hat{w}_j \tag{11.3}$$

Meaning that past N covariates, each new estimate of effect size is completely predictable from the previous ones in a deterministic way.

2. Even with fewer covariates and larger sample size, genetic variants exhibit linkage disequilibrium. Since variants in proximity of each other on the same chromosome tend to be inherited together by offspring with only occasional recombination between them, $\mathbf{x}_i'\mathbf{x}_j$ tend to be far from zero even in a large population, creating correlations between the $\hat{w}_i$ that are not artifacts of limited sample size.

What can we say of the lowest p-value then? Essentially, since some of our P tests are "redundant" because they test closely related hypotheses, we expect that seen as a, whole, by e.g. looking at the minimum p-value obtained, these p-values will behave like a smaller set of independent null p-values. This approach, that we will summarize here, was pioneered in FastQTL [Ong+16]. The existing implementation, however, has some subtle aspects that we analyze in the dedicated Section 11.3, while our proposed implementation of the idea, which is quite different, is exposed in Section 11.1.

Here again, empirical estimation of null properties comes handy. Let U and V be two independent random variables distributed as $\mathrm{Beta}(1, \beta)$ and $\mathrm{Beta}(1, \tilde{\beta})$, respectively, and let $M = \min(U, V)$. Then:

$$\begin{aligned}
\mathrm{d}M(m) &= \int_{\min(u,v)=m} \mathrm{d}U(u)\,\mathrm{d}V(v) &(11.4)\\
&= \int_{u=m, v\geqslant u} \mathrm{d}U(u)\,\mathrm{d}V(v) + \int_{v=m,u>V} \mathrm{d}U(u)\,\mathrm{d}V(v) &(11.5)\\
&= \int_{v\geqslant m} \mathrm{d}V(v)\,\mathrm{d}U(m) + \int_{u>m} \mathrm{d}U(u)\,\mathrm{d}V(m) &(11.6)\\
&= \int_{v\geqslant m} \mathrm{d}V(v)\,\mathrm{d}U(m) + \int_{u>m} \mathrm{d}U(u)\,\mathrm{d}V(m) &(11.7)\\
&\propto \int_{v\geqslant m} (1-v)^{\tilde{\beta}-1}\,\mathrm{d}v(1-m)^{\beta-1}\,\mathrm{d}m + \int_{v>m} (1-u)^{\beta-1}\,\mathrm{d}u(1-m)^{\tilde{\beta}-1}\,\mathrm{d}m &(11.8)\\
&\propto -\left[\frac{(1-v)^{\tilde{\beta}}}{\tilde{\beta}}\right]_{v=m}^{1} (1-m)^{\beta-1}\,\mathrm{d}m - \left[\frac{(1-u)^{\beta}}{\beta}\right]_{u=m}^{1} (1-m)^{\tilde{\beta}-1}\,\mathrm{d}m &(11.9)\\
&\propto \frac{(1-m)^{\tilde{\beta}}}{\tilde{\beta}}(1-m)^{\beta-1}\,\mathrm{d}m + \frac{(1-m)^{\beta}}{\beta}(1-m)^{\tilde{\beta}-1}\,\mathrm{d}m &(11.10)\\
&\propto (1-m)^{\beta+\tilde{\beta}-1}\,\mathrm{d}m &(11.11)
\end{aligned}$$

So

$$M \sim \text{Beta}(1, \beta + \tilde{\beta}) \tag{11.12}$$

Since a uniform is just a Beta$(1, 1)$, the minimum of P independent uniforms, such as P independent null p-values, is therefore a Beta$(1, P)$.

If we can simulate several sets of related null p-values, take the minimum of each set, and fit a beta distribution to it, we can therefore hope to obtain an *effective number of independent tests* that can be used to interpret p-values in this context.

Of particular importance is the combination of the empirical estimation of the effective number of independent tests with the empirical estimation of the parameters of each effective test described in Section 10.2.

We could perform these two estimation procedures independently, one after the other, and start by simulating null test statistics for individual variants and estimating the effective number of covariates, then compute p-values, now correctly approximately uniformly distributed under the null, with this estimated parameter, and finally take the best out of all these many times and fit the effective number of tests in a second step. However, this would miss an opportunity to do a joint estimation that would provide a better fit to the final best test statistics.

To illustrate it with values typical of what we've encountered, it could be that, when simulating individual null test statistics, and fitting our plain-t test statistic distribution, we've found that the null distribution is well modeled by a plain-t with an estimated effective number of covariates $\hat{K} = 70$ in the notation of Section 10.2. We now take 10,000 such tests, and take the best statistic out of all of these. Since our tests are not independent, we know that the resulting statistic will not follow the same distribution as the best of 10,000 independent plain-t, and we therefore repeat the process, generate enough samples, and look to fit a distribution to these, which we postulate to behave like the best of unspecified number of virtual, nonexistent plain-t statistics that we expect is much lower 10,000. The subtlety is that we need not, and should, not, restrict ourselves to only modeling as the best of an unknown number of imagined plain-t statistics with $\hat{K} = 70$ effective covariates, because the "independent effective statistics" that we imagine are different from the real statistics we're choosing our best value from. Rather, we can fit anew a distribution that models our values as the best out of an unknown number of plain-t statistics with an unknown number of effective covariates, which we can estimate jointly with the effective number of tests, and when doing so we will generally find the best fit to be quite different from $\hat{K} = 70$. In practice, we find for example that the best statistic could be best described, for instance, as the best out of 500 imaginary independent statistics, each distributed according to a plain-t distribution with $\hat{K} = 100$ rather than 70 effective covariates.

The result of this is that we effectively fit a two-parameters distribution family to a set of independently simulated null statistics. In that framework, the parameter of the original tests, that is the number of effective degrees of freedom, or effective covariates changes interpretation. Not only the second layer of the estimation can model the distribution as the best of several tests, it can also modify what is the distribution of each test statistic. In this way, the parameters of the base test can be adjusted, typically in the conservative direction, to provide an even better fit than if the base parameters were known and fixed and only the effective number of repetitions learned.

Intuitively, in the context of genotypes, we can imagine that taking the best of several associations from neighboring, strongly associated genetic markers has the effect of decreasing residual degrees of freedom as we're effectively doing feature selection of the best covariate based on data. The number of effective tests can then be understood to conceptually represent a number of effective independent loci, with each locus providing a feature selection model that has an inflated number of degrees of freedom removed. This is not to say that the genotype can exactly be modeled this way, but rather gives an intuition of why it is interesting and logical to allow both parameters to vary during the selection process, and why the final estimation of the degrees of freedom does not depend solely on the covariates used inside each individual association test.

11.1 Direct inference of null effective parameters and effective number of tests

If one has generated samples several test of null statistics, but the parameters of the null distribution are not known yet, it is not possible to compute p-values, and therefore not possible to apply the section above to find the effective number of tests.

One solution to that is to first generate sets of statistics, fit the null distribution, and then recompute all of the p-values; but this requires storing and scanning all the statistics again.

The alternative is to first select the best statistic, without knowing its distribution, from each set. In that case, we are left with samples $X_1, \ldots, X_N$, each of which is distributed as (say) the max of an unknown number K of random variables with density $f(x|\theta)$ for a known f but unknown θ. From these values, we're interested in fitting both K and θ jointly.

Fortunately, this is rather straightforward. Let

$$F(x|\theta) = \int_{-\infty}^{x} f(t|\theta)\,\mathrm{d}t \tag{11.13}$$

be the cumulative distribution function associated with f. The cumulative distribution function of the max of K random variables $Y_1, \ldots, Y_K$ with cumulative distribution function F is:

$$\begin{aligned} P(\max(\boldsymbol{y}) \leqslant y|\theta) &= \prod_{i=1}^{K} P(Y_i \leqslant y|\theta) && (11.14)\\ &= F(y|\theta)^K && (11.15) \end{aligned}$$

Therefore, the distribution of $\max(\boldsymbol{y})$ is:

$$\begin{aligned} \mathrm{d}P(\max(\boldsymbol{y}) \leqslant y|\theta) &= \mathrm{d}F(y|\theta)^K && (11.16)\\ &= KF(y|\theta)^{K-1}\,\mathrm{d}F(y|\theta) && (11.17)\\ &= KF(y|\theta)^{K-1} f(y|\theta)\,\mathrm{d}y && (11.18)\end{aligned}$$

The log-likelihood for N i.i.d. samples with such a distribution is therefore:

$$L(\mathbf{x}|K,\theta) = N\ln K + (K-1)\sum_{i=0}^{N} \ln F(x_i|\theta) + \sum_{i=0}^{N} \ln f(x_i|\theta) \tag{11.19}$$

which has as derivative:

$$\frac{\partial L(\mathbf{x}|K,\theta)}{\partial K} = \frac{N}{K} + \sum_{i=0}^{N} \ln F(x_i|\theta) \tag{11.20}$$

from which we derive the partial maximum likelihood estimate

$$\hat{K}(\theta) = \left(\frac{1}{N}\sum_{i-0}^{N} (-\ln F(x_i|\theta))\right)^{-1} \tag{11.21}$$

This estimate can be re-injected into (11.19), yielding the partially optimized likelihood:

$$\begin{aligned} L(\mathbf{x}|\theta) &= N\ln \hat{K}(\theta) - N - \sum_{i=0}^{N} \ln F(x_i|\theta) + \sum_{i=0}^{N} \ln f(x_i|\theta) && (11.22)\\ &= N(\ln N - 1) - \ln \sum_{i=0}^{N} (-\ln F(x_i|\theta)) - \sum_{i=0}^{N} \ln F(x_i|\theta) + \sum_{i=0}^{N} \ln f(x_i|\theta) && (11.23)\end{aligned}$$

11.2 Parametric solutions for null value distributions

In the above, we've emphasized fitting a parametric distribution to simulated null statistics. An alternative, which has been historically used for QTL calling, is to directly infer empirical null values distributions from simulations.

Let us imagine that we observed a test statistic value x. If we can simulate test statistics $X_1, \ldots, X_S$ from the null, we can compute an empirical p-value estimate as:

$$\hat{P}(X > x) = \frac{1}{S}|\{i, X_i > x\}| \tag{11.24}$$

While this method is theoretically completely sound, and has the advantage of being non-parametric and therefore not needing any assumption of the form of the null distribution of X, it poses a practical problem: the ability to compute it depends on the possibility of actually generating samples $X_i > x$. If x was actually generated under an alternative hypotheses, it may lie far in the tails of the null distribution, which means we would observe simulated null samples very rarely. This would give us the qualitative information that x is unlikely under the null, but almost no quantitative information about how unlikely it is.

For instance, imagine that the true p-value we aim to estimate is $p := P(X > x) = 0.001$, which is pretty realistic if x was actually not from the null. Then each $\mathbb{1}_{X_i > x}$ is an independent Ber(p), and:

$$|\{i, X_i > x\}| \sim \mathrm{B}(p, S) \tag{11.25}$$

has mean Sp and variance $Sp(1-p)$, therefore $\hat{P}(X > x)$ has mean p and variance $\frac{p(1-p)}{S}$, and its coefficient of variation is $\frac{\sqrt{1-p}}{\sqrt{pS}}$. To attain a coefficient of variation c, we therefore need:

$$c = \frac{\sqrt{1-p}}{\sqrt{pS}} \iff \sqrt{S} = \frac{\sqrt{1-p}}{c\sqrt{p}} \tag{11.26}$$

$$\iff S = \frac{1-p}{pc^2} \tag{11.27}$$

Therefore, estimating a true $p = 0.001$ with a coefficient of variation of $c = 0.1$ (that is, a mere 10% precision) requires of the order of $S = 100\,000$ samples. When obtaining a null sample requires extensive calculations to fit a model, this quickly becomes impractical. The fundamental problem is the factor $\frac{1}{p}$ in the formula for S above: to estimate the probability of a very rare event under the null, one first needs to obtain samples of comparable rarity, which is expensive. By contrast, parametric estimation, when reasonably employed, offer the promise of accurately estimating the probability of very rare events based on parameters estimated on a moderately small number of not-so-rare events.

11.3 Investigation of the FastQTL strategy

The widely used FastQTL program is a QTL caller (that is, a program design to take genotypes and variables of interest as inputs, and output measures of association strengths between genetic variants and target variables) that introduced to QTL calling the parametric estimation of effective degrees of freedom and effective number of tests [Ong+16]. However, the actual estimation process is quite different from the one we propose above, and, to the best of our knowledge, undocumented, including in the FastQTL publication and manual. We therefore propose to explain our understanding of said procedure here, reconstructed from analysis of the recovered source code and our re-implementation efforts.

To explain the working of it, it is necessary to first highlight a few properties of the beta distribution. If $X \sim \mathrm{Beta}(\alpha, \beta), \alpha > 0, \beta > 0,$

$$\begin{aligned}
\mathbb{E}[X] &= \frac{1}{\mathrm{B}(\alpha,\beta)} \int_0^1 u.u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u && (11.28)\\
&= \frac{\mathrm{B}(\alpha+1,\beta)}{\mathrm{B}(\alpha,\beta)} && (11.29)\\
&= \frac{\alpha}{\alpha+\beta} \qquad \text{according to (17.29)} && (11.30)
\end{aligned}$$

and

$$\begin{aligned}
\mathbb{E}\left[X^2\right] &= \frac{1}{\mathrm{B}(\alpha,\beta)} \int_0^1 u^2.u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u && (11.31)\\
&= \frac{\mathrm{B}(\alpha+2,\beta)}{\mathrm{B}(\alpha,\beta)} && (11.32)\\
&= \frac{\alpha+1}{\alpha+\beta+1}\frac{\mathrm{B}(\alpha+1,\beta)}{\mathrm{B}(\alpha,\beta)} && (11.33)\\
&= \frac{\alpha+1}{\alpha+\beta+1}\frac{\alpha}{\alpha+\beta} && (11.34)
\end{aligned}$$

by repeated application of (17.29), therefore

$$\begin{aligned}
\mathbb{V}[X] &= \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2 && (11.35)\\
&= \frac{\alpha+1}{\alpha+\beta+1}\frac{\alpha}{\alpha+\beta} - \left(\frac{\alpha}{\alpha+\beta}\right)^2 && (11.36)\\
&= \left(\frac{\alpha+1}{\alpha+\beta+1} - \frac{\alpha}{\alpha+\beta}\right)\frac{\alpha}{\alpha+\beta} && (11.37)\\
&= \left(\frac{\alpha(\alpha+\beta)+(\alpha+\beta)-\alpha(\alpha+\beta)-\alpha}{(\alpha+\beta+1)(\alpha+\beta)}\right)\frac{\alpha}{\alpha+\beta} && (11.38)\\
&= \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} && (11.39)\\
&= \frac{\mathbb{E}[X]\,(1-\mathbb{E}[X])}{\alpha+\beta+1} && (11.40)
\end{aligned}$$

From these expressions one can derive the successive equations:

$$\begin{aligned}
\alpha+\beta &= \frac{\mathbb{E}[X]\,(1-\mathbb{E}[X])}{\mathbb{V}[X]} - 1 && (11.41)\\
\alpha &= \mathbb{E}[X]\left(\frac{\mathbb{E}[X]\,(1-\mathbb{E}[X])}{\mathbb{V}[X]} - 1\right) && (11.42)\\
\beta &= (1-\mathbb{E}[X])\left(\frac{\mathbb{E}[X]\,(1-\mathbb{E}[X])}{\mathbb{V}[X]} - 1\right) && (11.43)
\end{aligned}$$

which gives the moment-matching estimators of the beta distribution.

The reasoning behind the FastQTL procedure is that if the parameters θ of the null distribution of a test statistics X have been correctly identified, then $S(X|\theta)$ should be uniformly distributed, and then $S(M|\theta)$ where M is the max of some number K of independent such statistics should follow a Beta$(1, K)$ distribution. If θ is far from its real value, then such a derivation does not work, and $S(M|\theta)$ would have some other distribution. Therefore, some criterion that measures how similar to some Beta$(1, K)$ its distribution is could be used to find θ, and one possible criterion is to fit a more general Beta(α, K) and see how far from 1 the estimated α is. When used with the moment matching estimator of α above, this gives the following loss function for θ:

$$\ell(\theta) := \left| m \left(\frac{m(1-m)}{v} - 1 \right) - 1 \right| \tag{11.44}$$

$$m := \frac{1}{N} \sum_{i=1}^{N} S(M_i|\theta) \tag{11.45}$$

$$v := \frac{1}{N-1} \sum_{i=1}^{N} (S(M_i|\theta) - m)^2 \tag{11.46}$$

This loss function is used by FastQTL to choose a reasonable effective number of degrees of freedom ν. We however put forward no reason to prefer it to the more direct derivation of the log-likelihood of M that we instead performed.

12 Compared results of eQTL pipelines

To draw conclusions on the relative performance of gene expression control methods, we need a metric that accurately translates to discovery power. Inspecting how many associations we find at a given p-value threshold would be one possibility, but even the multiple testing correction strategies listed above make this a bit difficult to interpret. While we can build null models for where we expect the best p-value to sit, this tells us little of the subsequent ones: what about the second best p-value and so on? We could count how many markers have an association p-value better than the expected best, but this is doubly unsatisfying.

- Since we expect subsequent p-values to be larger, this would be a very conservative way of counting associations.
- Conversely, due to linkage disequilibrium being widespread for variants in close proximity to one another, when one marker has a significant association, we expect that many neighboring markers will also have an inflated association strength. Therefore, when the best p-value makes the cut, it could well happen that a dozen or so other markers strongly correlated with the first would also be called as eQTLs, despite all of them truly representing one underlying causal association.

Instead, it is more stable to focus only on the top association: given that we have roughly 20,000 genes to look at, this still constitutes more than enough data to draw comparison between pipelines, even if in this way we're only looking at an extremely small fraction of the caller output. This lone p-value can be adjusted as explained with the methods described above, producing only one call per expressed gene; and telling whether this gene has or not at least one detectable eQTL at a given threshold. Counting the number of such genes, termed "eGenes" in the parlance of previous GTEx analysts, is in fact exactly what was used historically to select control models, when this choice meant selecting a number of PEER factors [Agu+17].

Figure 12.1 shows the results of our reproduction efforts from the reference GTEx publication. We compared the list of genes with significant eQTL publicly available with what we could obtain by executing eQTL calling ourselves from the raw data files of the project and the available code. The agreement between the two is high enough for the figure to not actually show that small differences occurred, of less than 10 genes for

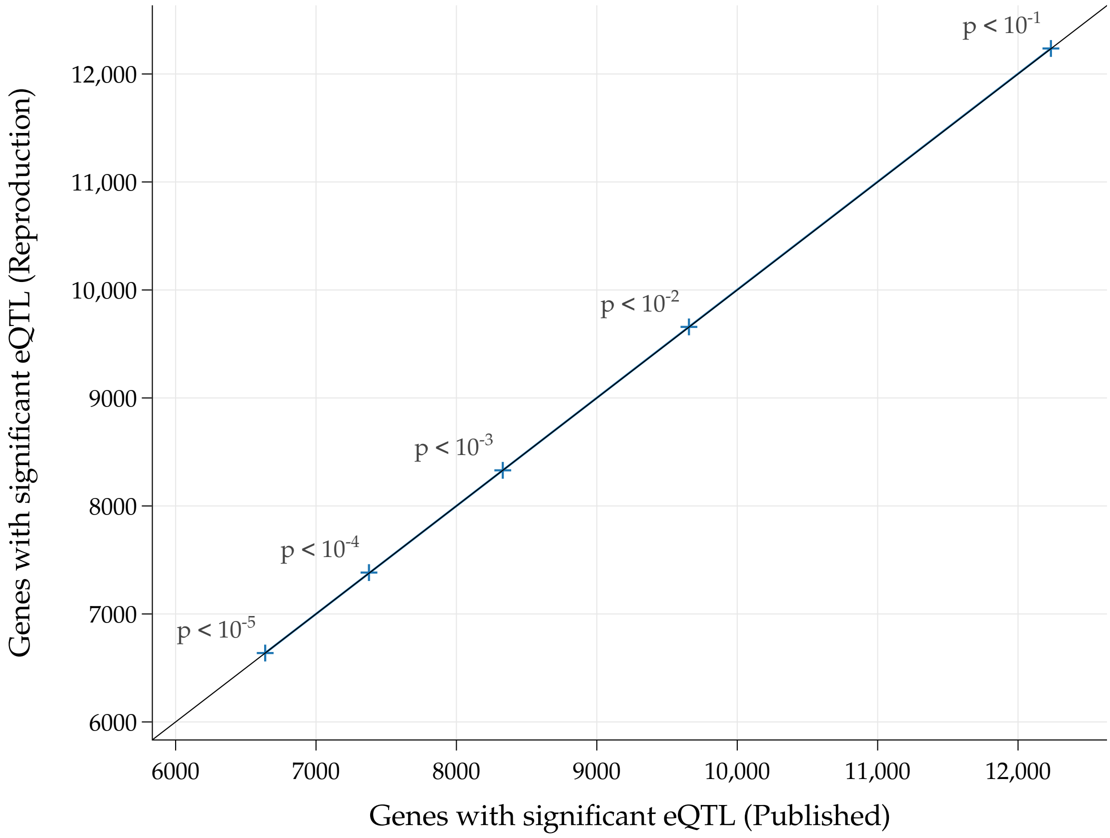

Figure 12.1: Number of genes for which the p-value of the most significant association, adjusted for multiple comparison, is inferior to a given threshold.

each data point shown. Overall, we consider this a replication success, enabled by the care with which the original code was released under the form of complete Workflow Description Language (WDL) pipelines and with suitable Docker images. This enables us to obtain a high confidence, in later comparisons, that we are effectively comparing new methods to a correct implementation of the state of the art.

Figure 12.2 shows how the number of genes with called eQTLs is affected by using expression from genes from other chromosomes instead of using permuted labels, as detailed in Section 10.3. Inter-chromosome controls can be seen to be more conservative that permutation controls. Since this analysis was performed without changing anything else to the way the control of genes was performed compared to original publications, this need to raise our attention and calls for further analysis. Our understanding is that the problem, here again, most likely comes from the fact that the traditional use of principal components or here PEER factors in eQTL calling is "leaky": the factors used for correction are not independent from the expression being corrected, but on the contrary are learned from it. This should produce an inflation in the number of

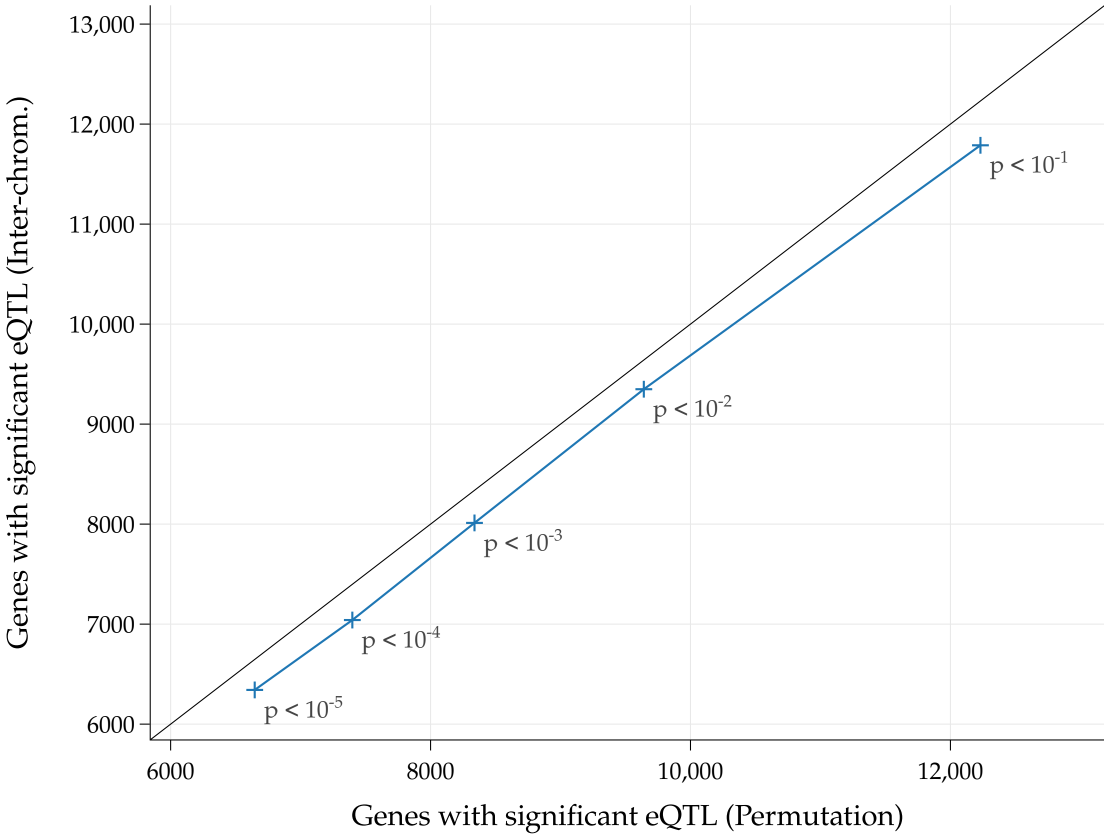

Figure 12.2: Number of genes for which the p-value of the most significant association is inferior to a given threshold, by adjustment strategy.

degrees of freedom being removed from the model. To explain why, consider an extreme case where a target variable was regressed by using itself as a control: we would observe the "predictor" explain 100% of the variance with an effect size of 1, and the residuals would be exact zeros, making the residual sum of squares a chi-squared with 0 degrees of freedom. In the more general and realistic case where some information was leaked into the covariates, we would expect this to cause some variable amount of loss of residual degrees of freedom. If the same phenomenon arises when simulating null p-values, this would be non problematic at least on average, as the null simulations would then produce p-values correctly inflated to compare to. However, this breaks down in the case of null genes simulated by permutation: if the PEER factors or principal components are not re-fitted with the permuted values, which is of course not practical at all, then no leakage occurs during the null simulation step, and the null statistics are generated with a systematically increased number of residual degrees of freedom compared to the target gene. As a result, the overall procedure would be slightly biased and anti-conservative, which would explain the result observed here.

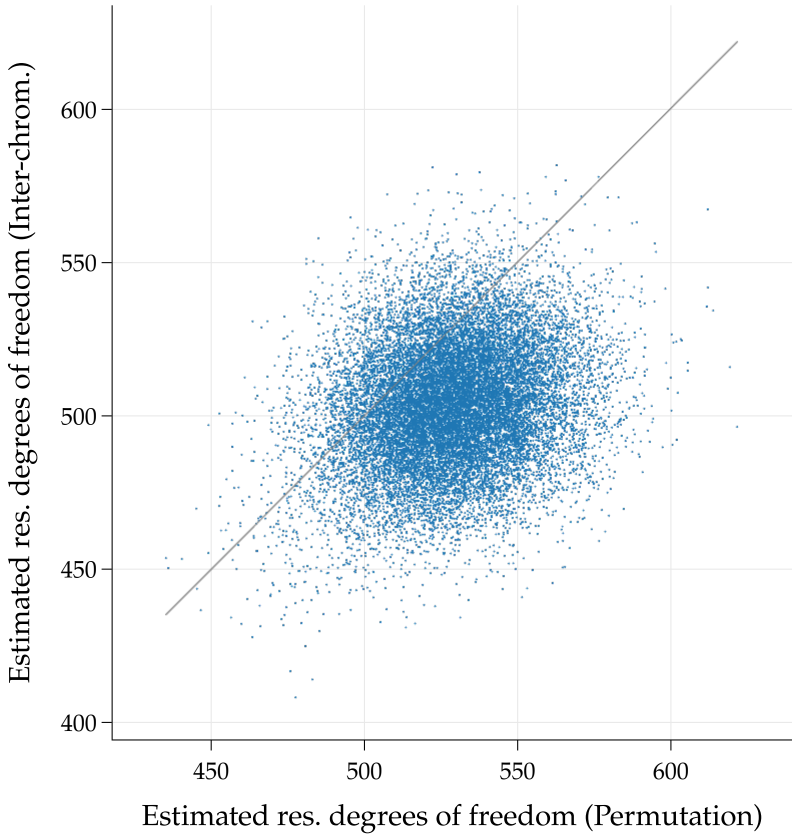

Figure 12.3: Estimated number of residuals degrees of freedom during null simulation for each individual gene, for the two proposed null simulation procedures.

This is in line with what a closer inspection of the estimated number of residual degrees of freedom during the null simulation phase shows, as depicted in Figure 12.3. A number of things stand out. First, there is significant variation from one gene to an other, but completely uncorrelated between the two methods. This is expected if that estimation noise is attributed to the randomness in generating null genes, which can only be done completely independently between the two methods. However, it suggests that we would benefit form increasing the number of null genes samples; this was set to a thousand to enable reasonable run times but can be realistically be set an order of magnitude higher for production use. Second, the number of degrees of freedom is overall rather low: we have 670 samples for this experiments (this is a bit less less than in Part I, as some samples may not have quality genotypes available), which means that we lost around 170 degrees of freedom. However, we cannot compare that directly to the number of covariates (namely 68: 60 PEER factors, 5 genotype principal components for population structure correction, sex, and two variables related to measurement methods if amplification (PCR) was used and with which of two sequencing platforms). Indeed,

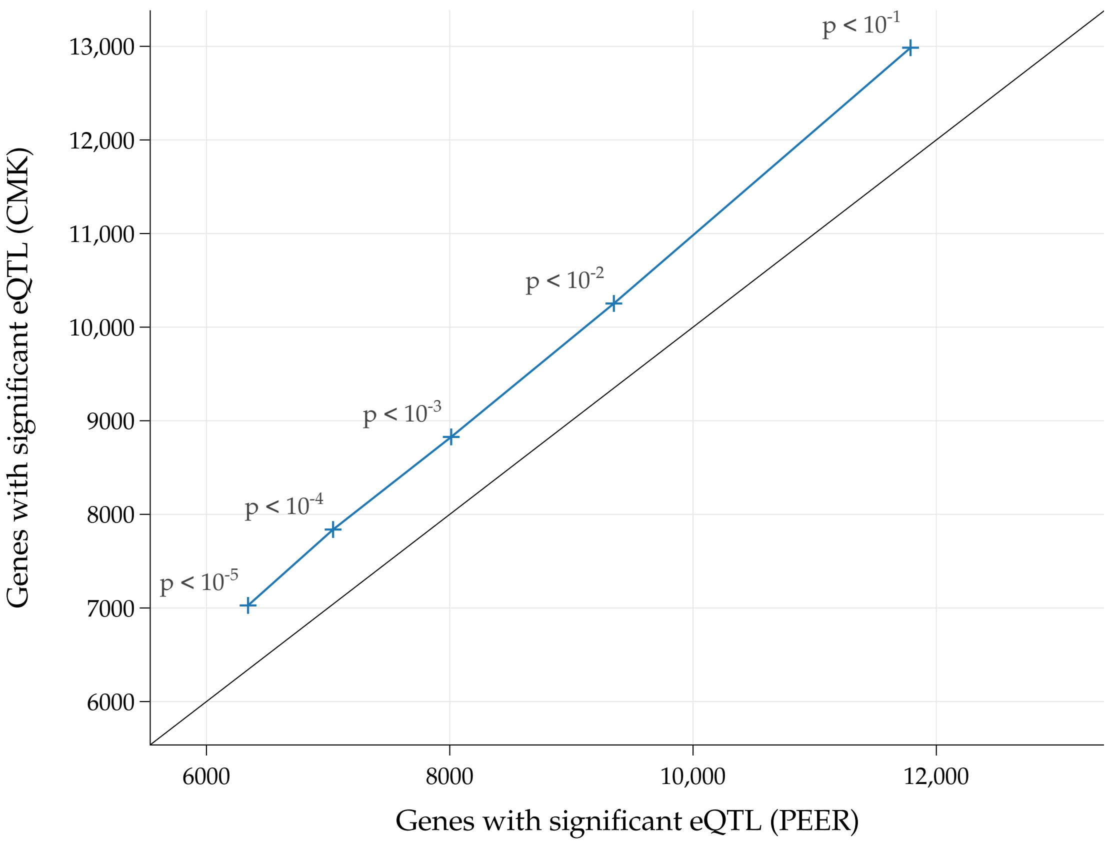

Figure 12.4: Number of genes for which the p-value of the most significant association, adjusted for multiple comparison, is inferior to a given threshold, by co-expression control method.

that number is estimated on the best p-value, and therefore also includes the correction for the bias coming from multiple testing: if we select only the best association out of several, the fit will appear to be better and less noisy than with single independent associations, as discussed in Chapter 11. Finally and most interestingly, the number of degrees estimated with inter-chromosomal sampling instead of permutation is around 25 degrees lower for the first (around 529) than the original (around 504), which is what suggests a noticeable bias coming from using permutations instead of real genes. In a complex pipeline and correction strategy such as this one, we cannot exclude that other factors that we have failed to consider would contribute to the difference; but at any rate this presents a serious argument in favor of the traditional permutation strategy being anti-conservative and makes us favor the inter-chromosomal strategy for the rest of our analyses.

Finally, figure 12.4 shows, in turn, the number of genes called when our clustered MacKay prediction method is swapped in in place of PEER in the re-engineered pipeline.

This is obtained by using the transformation procedure described in Section 10.4 along with the inter-chromosomal simulation strategy just discussed. Consistent increases in the number of genes called, while maintaining the same false discovery control procedures, can be observed, as was desired. This tends to validate our initial assumption that a more powerful and localized modeling of the gene co-expression signal noticeably improves the power in eQTL discovery. However, we believe that the most interesting consequences of this result will in the end not be most meaningful in eQTL calling, which was mostly and despite the complexity of the pipeline only the simplest of the applications that we could target in the scope of this thesis.

Rather, the same principles should also apply to more ambitious modeling of genotypes and expression data, such as transcriptome-wide association studies (TWAS), in which the increased power in describing genetic effects should also translate to an improved estimation of genetically-determined gene expression. Such pipelines routinely also used PEER factors, and often a restricted number thereof, to disentangle genetic from non-genetic effects and are a logical target for similar or larger gains [Gam+15]. Briefly, the first step of these methods is to build a multi-loci model to predict gene expression from genotype, for a given gene. This predictive model is then applied to large biobanks containing genotypes and phenotypes of clinical interest but no expression data, producing a predicted genetically determined expression value for the gene in each participant. Then a classical association test can be executed between this predicted expression and the phenotype of interest. If additional predictors are present in the original expression cohort used to train the method, just as the power to detect eQTLs is lowered as we discussed in Section 10.1 when no effort is made to control for them, then likewise the ability to accurately predict how the expression value should change from one participant to the next in the cohort of the second step decreases. Since the use that we have made of the CMK model here shows better eQTL discovery power, extending it from a one-locus association test method to a multi-loci prediction method while keeping the idea of locally correcting for hidden covariates captured by the gene co-expression pattern with the same model is expected to also lead to better predictive performance. This in turn is expected to give more useful predicted expression values and increase the power for discovering associations between imputed expression and clinical phenotype in the second step of TWAS.

But above all, we believe that this example demonstrates the value of a theoretical shift in bioinformatic analysis of modern quantitative biological data, away from scripted assembly of incompatible tools that introduce bias and discard information from one pipeline step to the next, and towards a more programmatic and integrated approach to data analyses that strives to maintain scalability while modeling jointly as much available information as feasible. The same philosophical principles call for a more detailed integration, for instance, of genotype models beyond the list of markers and genotype principal components used here, for a joint analysis of multiple tissues, and integration of external expression data; which by the same logic should all contribute to clearer and more powerful models.

Part III

Foundational Software

The models and pipeline presented in Parts I and II are implemented and released as their own python packages. However, there can be no good bioinformatics done without high standards of software development, and this implies an effort to identify the redundant difficulties of bioinformatic work, develop solutions to these, and package them as re-usable software components. Therefore, we present here a few packages and libraries developed as part of this thesis, on which we relied heavily for the applied work described in the previous parts. The interest of these modules is that none of their functionality is tied to the tasks on which we focused, namely gene expression imputation and eQTL calling. Instead, we expect that their functionality is completely general and will be of use to the author, their co-workers and others for the years to come.

All the code developed during this work will be released online under free and open-source licenses along with the related scientific publications. Until then, early access to any part of the software can be obtained upon request to the author or directly on the author's GitHub page where a selection of packages has already been released.

13 A virtual linear algebra layer

13.1 Problem description

The Python ecosystem that we favor to implement our models and pipelines shines by the somewhat confusing number of options it offers when it comes to libraries to manipulate vectors, matrices and higher order tensors.

- NumPy is the de facto standard for scientific calculation [Har+20]. It offers efficient containers, but with limited functionality when it comes to linear algebra (there is e.g. no Cholesky decomposition) and with no support for automatic differentiation.
- SciPy does not offer its own data types and containers, but contains essential linear algebra routines and universal functions (functions that can be applied to arbitrary tensors with typically good performance) that operate on NumPy arrays, and therefore supplements NumPy [Vir+20].
- TensorFlow is one of the important provider of tensor containers that come with automatic differentiation by building a graph of operation when functions of the package are applied to its data types. It is backed by Google [Aba+16].
- PyTorch is in a way the concurrent of TensorFlow backed by Facebook [Ans+24]. It has essentially equivalent functionality, and is often seen as more beginner-friendly.
- JAX is an other Google-supported framework, that make use of the same back-end as TensorFlow, namely the Accelerated Linear Algebra (XLA) compiler [FJL18]. It features a radically different user interface, however, oriented around the programmatic manipulation and transformation of functions through intermediate representations (Jaxprs). For the more conventional part of the framework, it closely mirrors the NumPy API — in fact, as much as its internal design allows.

All of these libraries offer distinct and incompatible interfaces to perform semantically the same operations. Some of them have an obvious enough mapping in the Python object model that they are implemented consistently. For instance, `A + B` performs an element-wise addition of the tensors `A` and `B` no matter whether these are

NumPy arrays, TensorFlow tensors, Torch tensors, JAX arrays, and produces a result of the same type than the argument. But for anything less trivial, the interfaces are different: taking the element-wise exponential, for instance, will be done with `np.exp(A)`, `tf.exp(A)`, `A.exp()`, `jnp.exp(A)` depending on the type of the tensor `A`. Trying to mix operations from different libraries gives inconsistent results. It may preserve the type or the original tensor, convert it silently to a data type of the source library, or outright fail. It may even have framework-specific meaning, as is the case for JAX which encourage mixing of NumPy and JAX operations to control which operation should take place when tracing/compiling or executing a function. This limits code re-use and encourages developer of computational methods to lock themselves in by committing to the use of a particular framework, requiring different independent implementations of the same algorithms merely to obtain working code with a different library.

Overall, writing code that is completely independent of the framework is not possible, as each may have specific operations that may not be one-to-one translatable to operations in others. For instance, JAX offers lot of control on automatic vectorization, functional differentiation (that is, automatic differentiation implemented as a function which takes as an argument an other function, and returns e.g. a generated pullback operator), and compilation that may simply not be offered by an other back-end. For a given application, the high level code may need to be specific to a framework. Conversely, scientific software should be organised around building blocks of particular methods, which will often be functions that merely do common arithmetic and linear algebra operation on tensors without relying on any library-specific advanced features, implement common probabilistic tools (e.g, the log-density of a multivariate normal at a given point). There is then no reason why the block could not be written in a framework-agnostic form, and then freely composed inside a specific framework to create high-level applications.

A notable effort in this direction is the Array API standardization effort started mainly by the NumPy project [Con23]. It proposes, for instance, for compatible frameworks to implement a `.__array_namespace__()` method through which framework-specific implementations of a standard set of operation may be accessed by application code without knowledge of the specific framework in use. Unfortunately, as can be gathered from browsing the standard, it is very much in its infancy, restricted to a particularly small set of operations and not widely implemented. For our needs, we therefore developed a software solution that essentially anticipates this functionality with no cooperation from the included frameworks required.

13.2 Solution

Our concept builds on exiting efforts of the NumPy project to make the library extensible to new container (tensor) types. On top of the common python operators such as `add`, the NumPy project already offers many *protocols* which describe how custom classes may decide what should happen when NumPy operations are performed on them [Hoy+18; HS18; Ent19]. This already makes it possible to implement the wanted feature — a way to write code that will work with several linear algebra and automatic differentiation frameworks. The idea is to wrap the containers provided by the different framework, each having its own interface, into proxy objects that implement various NumPy protocols and dispatch the supported operation to the framework specific variants. The basic skeleton may start with code similar to the following snippet:

```
import numpy as np

class MyTensorProxy:
    def __init__(self, backend_tensor):
        self.backend_tensor = backend_tensor
    def __array_function__(self, func, _, args, kwargs):
        if func is np.linalg.inv:
            # Custom code to perform a matrix inversion on
            # self.backend_tensor, returning a new TensorProxy
            return ...
```

One can then wrap an arbitrary back-end tensor in the proxy object:

```
proxy = MyTensorProxy(tensor)
```

and then perform operations on the proxy as if it were a NumPy array:

```
...
inverse = np.linalg.inv(proxy)
...
```

which will perform the intended operation on the underlying `tensor` in cases where `np.linalg.inv(tensor)` would not work or not have the intended behavior — say, convert the tensor to a NumPy array, which would break automatic differentiation.

To further simplify the use of the technique, the wrapping and unwrapping of the back-end tensors into proxy objects can be done inside a function wrapper, which in turn can be applied using Python's decorator syntax. Under its simplest form, our library can therefore be used, in the case of JAX that is the main one that we use, by merely adding a `@jaxify` decorator to a function:

```
@jaxify
def my_loss_function(x, y, z):
        # Inside the function decorated with jaxify, you can use
        # the supported subset of numpy functions without regard
        # to whether the tensor are NumPy arrays. In other
        # words, you can implement calculations and re-use code
        # as if you were always using NumPy.
        r = np.tensordot(x, y) + np.linalg.slogdet(z)[1]
```

```
        ...

def outer(...):
        ...
        X = jnp.array(...)
        ...
        # Calling code only sees my_loss_function as a function
        # that accepts and return jax types. jaxify takes care
        # of wrapping the JAX arrays into proxies that
        # understand NumPy operations, and unwrapping the
        # result.
        R = jax.vmap(my_loss_function)(X, Y, Z)
        ...
```

An other common and convenient interface to this functionality that we use is with our JAX helpers for SciPy optimizers. Using the `jaxify` transform above, we can have optimization routines that accepts loss functions written in standard simple NumPy syntax, and automatically transform them into JAX functions. They can then be traced, differentiated, and compiled using JAX, before being used with differentiation-aware optimizers (gradient descent and refinements). From the point of view of the user, the optimizer simply looks like it's working out of the box with NumPy, with the differentiation and compilation coming for free without the user even needing to be aware that JAX is being used.

14 A library for linear algebra automation

We've seen in Section 4.4 how central some linear algebra identities can be in efficient model fitting and evaluation, especially the matrix determinant lemma from (4.63) that we used to derive (4.65) and the Woodbury identity from (4.72) that was applied to obtain (4.75). Most of the difficulty of implementing an optimized model like CMK lies in correctly and repetitively applying such linear algebra tricks. This raises the question, from a computational application perspective, of how much of it can be automated. Since much of the differentiation work needed to fit models can be relatively efficiently automated away by automatic differentiation frameworks, it is tempting to consider whether we can achieve something akin to "automatic low rank updates" programmatically to make the implementation of models like CMK drastically easier — and, by consequence, iterate much faster in writing and testing such algorithms.

Example of similar solutions can already be seen in scientific computing libraries. Handling sparse matrices, for instance, is an example of a related problem. The `sparse` module of SciPy exploits polymorphism to offer classes that behave much like NumPy arrays, but whose implementation has reduced complexity in storage and time for truly sparse arrays. A related feature is the way `scipy.optimize` handles Hessian approximations for approximate second order methods. L-BFGS, for instance, can be implemented with the `LbfgsInvHessProduct` class, which represents an inverse Hessian approximation, but only computes it as a dense matrix on demand, which is normally never required by the algorithm, as long as the possibility of calculating a matrix-vector product exists. Finally, automatic differentiation itself effectively manipulates large linear maps, as the Jacobians of multivariate functions; but such Jacobians are never explicitly calculated and are instead represented by the composition of several efficient push-forward or pull-back operators.

In such a context, one can write objects that would encapsulate a low-rank linear map and provide custom, efficient implementations of select operations. Imagine a matrix $\mathbf{A}$ defined by

$$\mathbf{A} = \mathbf{U}\mathbf{V} \tag{14.1}$$

where the inner matrix product has a low dimension. For a given vector $\mathbf{x}$, naively computing

$$\mathbf{A} = \mathbf{U}\mathbf{V} \tag{14.2}$$

$$\mathbf{y} = \mathbf{A}\mathbf{x} \tag{14.3}$$

would not exploit the low-rank structure of the problem when translated into regular, say, NumPy operations:

```
A = U @ V
# Other operations...
y = A @ x
```

By contrast, one can create an object that would remember the structure of A and customize its behavior:

```
class LowRankProduct:
        def __init__(self, U, V):
                self.U = U
                self.V = V
        def __matmul__(self, x):
                return self.U @ (self.V @ x)
```

that can be used as:

```
A = LowRankProduct(U, V)
# Other operations...
y = A @ x
```

The resulting code retains much of the readability of the naive implementation, with the underlying storage and operation being efficient.

For cases resembling CMK, of course, this isn't the feature we need, but something along the same idea can be used. What we really want is a way to represent the tensor or order three:

$$\mathbf{\Sigma}_g = \mathbf{\Sigma} - v_{tt}\mathbf{y}_g\mathbf{y}_g' \tag{14.4}$$

In such a way that computing determinant and inverses exploit the rank-one update structure. Generally, we propose to store tensors of rank three of the form:

$$t_{ijk} = b_{ij} + \sum_{l,m} u_{ilk}c_{lmk}v_{mjk} \tag{14.5}$$

Here, t_{ijk} can be thought of as a stack of matrices indexed by k. Then b_{ij} is the base matrix that is shared across the whole stack, while the second term is the update; typically c_{lmk} is a very small weight matrix for each k, typically 1×1 for rank-one update scenarios. u_{ilk} and v_{mjk} are then normally tall and wide rectangular matrices for each of the few (l, m) combinations. In software, the tensor t is then represented by having the four components b, u, c, v of the tensor stored individually, with none of the tensor contractions or addition being actually computed.

Now, let us consider the inverse of $t_{..k}$ for each k, that we'll write t^{-1} though the notation is otherwise ambiguous. Applying the Woodbury (4.72) identity for each k yields:

$$t^{-1}_{jik} = b^{-1}_{ji} - \sum_{l,m} \left(\sum_i b^{-1}_{ji} u_{ilk} \right) \left(c^{-1}_{mlk} + \sum_{i,j} v_{mjk} b^{-1}_{ji} u_{ilk} \right)^{-1} \left(\sum_j v_{mjk} b^{-1}_{ji} \right) \tag{14.6}$$

The crucial observation is that this batch-inverse tensor had the same form as the original one: a batched matrix b^{-1}_{ji} shared for all k, plus a low-rank update centered around a small batched matrix. This means that the type that we use to store such a tensor is software can also be used to store its inverse, and we can therefore easily implement an efficient inversion on this representation.

From this point, the same design principles than in Chapter 13 can be employed, as we've hinted at before with our `LowRankProduct` example: just as we could write proxy tensors that interpret NumPy operations and translate them to back-end dependent operations, we can write specialization for matrices and higher order tensors such as this batched low-rank updates and implement efficient inversion, determinants, etc. in terms of the component tensors. From the point of view of the programmer, these can be used by simply applying the normal NumPy endpoints (`np.linalg.inv`, `np.linalg.slogdet`) to instances of these custom classes. Most interestingly, the technique presented here and in Chapter 13 combine: we can implement low rank update formulas in terms of component tensor operations without knowing what kind of tensors they exactly are, and rely on our transforms to make these optimized implementation automatically work with JAX and therefore differentiate and compile them. In fact, this exact technique was used quite extensively to implement or prototype statistical methods of Part I.

15 A workflow manager tightly integrated with Python

Software designed to help bioinformaticians to write pipelines have become a central tool in the landscape of modern computational biology. In the following, we briefly detail what we define as a "workflow manager", the current leading software falling under that term, and the motivation and process of developing a new tool offering unique features as part of this thesis.

A bioinformatics analysis will usually consists of running a large number of programs, or tools, on a set of files; each tool producing as outputs new files that can become in turn the inputs of an other tool in a subsequent (or "downstream") analysis step. A step, in this context, can be something as simple as downloading a reference genome file with `wget` or decompressing a dataset with `gunzip`, or as complicated as fitting a Bayesian model with `stan` or executing a whole other workflow as a sub-pipeline.

However, a long analysis pipeline will normally not be written in one go. Instead, it is the result of an iterative process in which the analyst experiments with adding one step to the pipeline, changing the tool and option used and solving problems before moving on to the next. The time spent on adding one step can be days of development, and as much of waiting for the results.

The time scale of workflow writing and execution creates a conflict between two constraints.

- For the analysis to be reproducible and easy to re-run with a different data source or different parameters, it is vital that a single entry point somewhere exists, that allows to run the whole pipeline from scratch with few commands, ideally only one.
- On the other hand, since the time needed for a full run of the pipeline may be prohibitively high, it is equally vital that the pipeline may also be run step-wise, or *incrementally*, typically re-running only the last added step currently under development while keeping the other files unchanged.

Computer scientists at large have long encountered the problem in the specific case of compilation: source files need to be turned into object files that need to be linked together to form a library or executable, and we want to be able to add or modify a source file without recompiling everything. This lead to programs such as Make and its modern iterations such as GNU Make. Bioinformaticians noted the similarity with their problems and came up with largely Make-inspired software more specialized for bioinformatics, such as Snakemake [KR12] and Nextflow [Di +17] to only cite the most popular tools.

While these workflow manager are invaluable, for the data scientist or the bioinformatician who spends more time writing and fitting statistical models than running aligners, they suffer from a rather coarse *granularity*: a step in the pipeline must be its own executable, loading its data from disk given paths passed by the workflow manager and writing all of its results back to output paths before finishing and advancing to the next step. This makes pipeline composed of shorter, simpler data manipulation operations more tedious in two ways.

1. Since each step takes paths as inputs and outputs, the code that define each step tends to rapidly grow redundant and include lot of lines purely dedicated to input/output. Before each step, we must have code to reload all of the relevant data from CSV, HDF5, XML files etc.

2. Especially when using workflow managers on high-performance computing resources and workload managers such as SLURM or SGE, each step is submitted as its own cluster job. This results in a typically very high overhead of seconds to hours per job, and makes it unscalable to write pipelines that consist of huge amounts of short independent steps. Therefore, the pipeline writer needs to take care of manually merging enough computation in a single pipeline step, instead of keeping steps as the smallest logical units of the computation.

After some time of experimenting with pushing the limits of Snakemake and similar by abusing dynamic rules and input functions, we eventually settle on our own workflow manager that explicitly targets a finer granularity. While it stays strongly inspired by its predecessors, its defining characteristics are as follows.

- Each pipeline step is a single function, not an independent executable from start to finish.

- When data has to be written to disk or serialized for transfer between compute nodes, this by defaults happens automatically by exploiting the rich and efficient serialization routines of the Python ecosystem. The pipeline writer need not care about the files in which the data passed from one step to an other is stored.

- The whole pipeline can be written and executed by using a normal dedicated Python library. Contrary to Make, Snakemake, Nextflow and consorts, there is no need of a domain specific language, custom file types to describe the pipeline or anything of the sort. Vanilla Python programming is enough.

- While pipeline steps that have no data dependency on each other can be executed in parallel in different processes, the Python interpreter and processes can be recycled from one pipeline step to the other. This means that two sequential steps may end up being executed back to back inside the same python interpreter, avoiding the overhead of starting a new process, let alone a whole new SLURM job. Thanks to that, our lightweight workflow manager can achieve overhead of the order of 10ms per pipeline step. This significantly reduce the need to think about strategically segmenting a pipeline into steps that are large enough, and avoids sacrificing any parallelism as a result.

In many of the aspects above, our solution is strongly influenced by Dask, a noted Python library for distributed computing sometime described as a more pythonic Spark. But while not reaching the performance of Dask, we add persistent progress tracking between program re-execution, which is the core of workflow management systems, considered as incremental distributed script runners. In a way, we therefore sit in a niche at the intersection of the features of Nextflow or Snakemake and Dask.

Part IV

Statistical and Algebraic Identities

16 A variational view of the Moore-Penrose pseudoinverse

16.1 The pseudoinverse as limit solution of approximate inverse matrix problems

Just as solving linear systems of equations frequently connect to fitting models, models that admit no unique best fit naturally feature non-invertible linear maps, and the Moore-Penrose inverse, that we'll simply call "the pseudoinverse" in the following often appear in our equations. Traditionally, the pseudoinverse is defined algebraically as the only matrix satisfying a set of algebraic conditions; or constructively from a singular value decomposition by inverting all of the nonzero singular values. In our view, these definitions help little to understand why and how such a concept appear so often in our statistical procedures. We therefore propose here to focus on *variational* introductions to pseudoinverse constructions.

Let $n, p \in \mathbb{N}_*$ and $\mathbf{A} \in \mathbb{R}^{n \times p}$ some rectangular matrix. If $n = p$ and $\mathbf{A}$ is invertible, there exist a unique matrix $\mathbf{X}$ such that $\mathbf{XA} = \mathbf{I_p}$ and $\mathbf{AX} = \mathbf{I_n}$. But in general, there may exist no such matrix, and we may face linear problems that admit no solution. Still, in that case, we may be interested in defining something as close as possible to an inverse, a "best effort" linear map that tries to reconstruct the inputs of $\mathbf{A}$. To this effect, we can consider "how close" $\mathbf{AX}$ and $\mathbf{XA}$ get to the identity matrices of their respective dimensions. For instance, we may consider the matrices that minimize the discrepancies:

$$\|\mathbf{XA} - \mathbf{I_P}\|^2 = \operatorname{Tr}\left[(\mathbf{XA} - \mathbf{I_P})'(\mathbf{XA} - \mathbf{I_P})\right] \tag{16.1}$$

$$\|\mathbf{AX} - \mathbf{I_N}\|^2 = \operatorname{Tr}\left[(\mathbf{AX} - \mathbf{I_N})'(\mathbf{AX} - \mathbf{I_N})\right] \tag{16.2}$$

Even given the existence of minimal solutions, which as we will see is not problematic, we'll face the problem that there may be several minimal solutions. We therefore need an additional way to disambiguate them, which can be done by imposing further constraints on the norm $\|\mathbf{X}\|^2$. For some $c > 0$, we can then consider problems such as:

$$\text{minimize } \left\|\mathbf{XA} - \mathbf{I_p}\right\|^2 \tag{16.3}$$
$$\text{subject to } \|\mathbf{X}\|^2 \leqslant c \tag{16.4}$$

Solutions where the constraint is active will be found from the Lagrangian:

$$\mathcal{L}(\mathbf{X}, \beta) = \left\|\mathbf{XA} - \mathbf{I_p}\right\|^2 + \beta(\|\mathbf{X}\|^2 - c) \tag{16.5}$$

which has gradient:

$$\nabla_{\mathbf{X}}\mathcal{L} = (\mathbf{XA} - \mathbf{I_p})\mathbf{A}' + \beta\,\mathbf{X} \tag{16.6}$$
$$= \mathbf{X}(\mathbf{AA}' + \beta\,\mathbf{I_n}) - \mathbf{A}' \tag{16.7}$$

and therefore, since $\mathbf{AA}' + \beta\,\mathbf{I_n}$ is always invertible for $\beta > 0$, such bound solutions have the form:

$$\mathbf{X}_\beta = \mathbf{A}'(\mathbf{AA}' + \beta\mathbf{I_n})^{-1} \tag{16.8}$$

For the alternative problem:

$$\text{minimize } \|\mathbf{AX} - \mathbf{I_n}\|^2 \tag{16.9}$$
$$\text{subject to } \|\mathbf{X}\|^2 \leqslant c \tag{16.10}$$

a similar derivation lead to the gradient:

$$\nabla_{\mathbf{X}}\mathcal{L} = \mathbf{A}'(\mathbf{AX} - \mathbf{I_n}) + \beta\,\mathbf{X} \tag{16.11}$$
$$= (\mathbf{A}'\mathbf{A} + \beta\,\mathbf{I_p})\mathbf{X} - \mathbf{A}' \tag{16.12}$$

and solutions:

$$\mathbf{X}_\beta = (\mathbf{A}'\mathbf{A} + \beta\mathbf{I_p})^{-1}\mathbf{A}' \tag{16.13}$$

However, by the push-through identity that we already encountered in (4.23):

$$\mathbf{A}'\mathbf{AA}' + \beta\,\mathbf{A}' = (\mathbf{A}'\mathbf{A} + \beta\,\mathbf{I_p})\mathbf{A}' = \mathbf{A}'(\mathbf{AA}' + \beta\,\mathbf{I_n}) \tag{16.14}$$

so

$$\mathbf{A}'(\mathbf{AA}' + \beta\mathbf{I_n})^{-1} = (\mathbf{A}'\mathbf{A} + \beta\mathbf{I_p})^{-1}\mathbf{A}' \tag{16.15}$$

So for every $c > 0$ where the constraint is active, there is a unique matrix $\mathbf{X}_\beta$ that minimizes both $\left\|\mathbf{XA} - \mathbf{I_p}\right\|^2$ and $\|\mathbf{AX} - \mathbf{I_n}\|^2$, i.e a unique "best almost inverse of a given size" to $\mathbf{A}$.

Of special interest is therefore the matrix obtained when the constraint is barely binding and $\beta \to 0$. In that case, we reach the unconstrained minimum of $\left\|\mathbf{XA} - \mathbf{I_p}\right\|^2$

and $\|\mathbf{AX} - \mathbf{I_n}\|^2$ but consider only the unique minimal solution with also smallest $\|\mathbf{X}\|^2$. This turns out to define the Moore-Penrose inverse, which can be therefore obtained equivalently as:

$$\mathbf{A}^+ = \underset{\mathbf{X}\in M}{\operatorname{argmin}}\|\mathbf{X}\|^2, \quad M = \underset{\mathbf{Y}\in\mathbb{R}^{p\times n}}{\operatorname{Argmin}}\left\|\mathbf{YA} - \mathbf{I_p}\right\|^2 \tag{16.16}$$

$$= \underset{\mathbf{X}\in M}{\operatorname{argmin}}\|\mathbf{X}\|^2, \quad M = \underset{\mathbf{Y}\in\mathbb{R}^{p\times n}}{\operatorname{Argmin}}\|\mathbf{AY} - \mathbf{I_n}\|^2 \tag{16.17}$$

$$= \lim_{\beta\to 0} \mathbf{A}'(\mathbf{AA}' + \beta\mathbf{I_n})^{-1} \tag{16.18}$$

$$= \lim_{\beta\to 0} (\mathbf{A}'\mathbf{A} + \beta\mathbf{I_p})^{-1}\mathbf{A}' \tag{16.19}$$

Thus the pseudoinverse $\mathbf{A}^+$ is the "smallest best possible inverse" of $\mathbf{A}$. Under this light, it is unsurprising to find it involved in statistical modeling, where we often look for the "most parsimonious best fit" of a model that has neither an exact fit not a unique best fit.

Furthermore, this exposition highlights the relationship between $\mathbf{A}^+$ and all of the $(\mathbf{A}'\mathbf{A} + \beta\mathbf{I_p})^{-1}\mathbf{A}'$ for $\beta > 0$, allowing us to understand the latter as smooth alternatives to the pseudoinverse with β quantifying the trade-off between obtaining more exact inverses of $\mathbf{A}$ and bounding the explosion of the solution when $\mathbf{A}$ becomes close to singular. This naturally connects this family of matrices to "regularized" estimation problems.

16.2 The pseudoinverse as solution map to optimal vector problems

While we presented the pseudoinverse and its smooth approximations as solutions of a matrix problem, note that they also offer solutions to related vector problems.

For a matrix $\mathbf{A}$ and vector $\mathbf{y}$, $\mathbf{Ax} = \mathbf{y}$ could have several or no solutions in $\mathbf{x}$, so we may want to solve the relaxed problem:

$$\text{minimize } \|\mathbf{Ax} - \mathbf{y}\|^2 \tag{16.20}$$

$$\text{subject to } \|\mathbf{x}\|^2 \leqslant c \tag{16.21}$$

with Lagrangian ($\beta \geqslant 0$):

$$\mathcal{L}(\mathbf{x}, \beta) = \|\mathbf{Ax} - \mathbf{y}\|^2 + \beta(\|\mathbf{x}\|^2 - c) \tag{16.22}$$

and gradient:

$$\nabla_{\mathbf{x}}\mathcal{L} \propto \mathbf{A}'(\mathbf{Ax} - \mathbf{y}) + \beta\mathbf{x} \tag{16.23}$$

thus solution:

$$\mathbf{x} = (\mathbf{A}'\mathbf{A} + \beta\,\mathbf{I})^{-1}\mathbf{A}'\mathbf{y} \tag{16.24}$$

Thus $(\mathbf{A}'\mathbf{A} + \beta\,\mathbf{I})^{-1}\mathbf{A}'$, and its limit $\mathbf{A}^{+}$, that we have just derived as the "best inverse of $\mathbf{A}$ under a certain matrix norm", can also be understood as the family of linear maps that gives the individual "best solutions under a certain vector norm" to linear problems defined by $\mathbf{A}$. Fundamentally, this works because (16.23) is still linear in $(\mathbf{x}, \mathbf{y})$, so it is still possible to find solutions to (16.21) by decomposing $\mathbf{y}$ on some basis and then combining the solutions for the basis vectors.

16.3 Connection to linear regression

Given a matrix of covariates $\mathbf{X}$ and target observations $\mathbf{y}$, linear regression looks for a weight vector $\mathbf{w}$ such that $\mathbf{X}\mathbf{w}$ best fits $\mathbf{y}$. In common cases there is no exact fit and this means looking for the unique minimum of $\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$, given i.i.d. Gaussian errors. But while $\mathbf{w} = \mathbf{X}^{+}\mathbf{y}$ then turns out to be the solution, the derivations below explain clearly why $\mathbf{X}^{+}\mathbf{y}$ still makes sense as best solution of the so-called *indefinite least square* problem where there is an infinity of optimal solutions: it corresponds to the smallest (in the sense of $\|\mathbf{w}\|^2$), and therefore most likely under a spherical Gaussian effect size prior, solution. Indefinite least square especially appears when:

- there are less samples than covariates and therefore effects of individual covariates cannot be identified (and there are typically an infinity of *exact* fits), or
- some covariates are linearly dependent (in which case we may have inexact fits, but still an infinity of best ones).

Furthermore, this suggests that $(\mathbf{X}'\mathbf{X} + \beta\,\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}$ for some $\beta > 0$ should almost always be considered as an alternative solution that sacrifices some model fitness to guarantee that effect sizes do not diverge when the problem is "almost" indefinite, that is when we have only a few more samples that covariates or when covariates are almost linearly dependent.

17 Fundamental kernels, distributions, functions and change of variables

In the following, we list the definitions and, with selected proofs, the essential properties of the distributions that arise in our work. Rather than grouping the result by distribution in tables as one can find in encyclopedic references for easy lookup, we prefer to list them here in the order in which these properties can be deduced from one to another.

17.1 The multivariate normal distribution

While the multivariate Gaussian needs not be introduced, some properties of its conditionals are noteworthy. Let $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ on $\mathbb{R}^n$ for some symmetric positive definite matrix $\boldsymbol{\Sigma}$. The covariance matrix $\boldsymbol{\Sigma}$ has an inverse, the precision matrix $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$. Then, the vector:

$$\begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{\Lambda x} \end{bmatrix} \tag{17.1}$$

is multivariate normal, has mean $\mathbf{0}$ and covariance matrix:

$$\begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{I_n} \\ \mathbf{I_n} & \boldsymbol{\Lambda} \end{bmatrix} \tag{17.2}$$

This means that for each $i \in \{1, \dots, n\}$, the quantities:

$$\sum_{j=1}^{n} \lambda_{ij} X_j, \quad \text{and therefore,} \quad X_i + \frac{1}{\lambda_{ii}} \sum_{j \neq i} \lambda_{ij} X_j \tag{17.3}$$

have mean 0 and are independent (by Gaussian decorrelation) from each of the $X_j, j \neq i$. We can therefore decompose:

$$X_i = -\frac{1}{\lambda_{ii}} \sum_{j \neq i} \lambda_{ij} X_j + \varepsilon \tag{17.4}$$

where

$$\varepsilon = X_i + \frac{1}{\lambda_{ii}} \sum_{j \neq i} \lambda_{ij} X_j \tag{17.5}$$

is a random normal variable with mean 0, variance $1/\lambda_{ii}$ and is independent from $X_j, j \neq i$, which means that we have the conditional "leave-one-out" distribution:

$$X_i | (X_j)_{j \neq i} \sim \mathcal{N}\left(-\frac{1}{\lambda_{ii}} \sum_{j \neq i} \lambda_{ij} X_j, \frac{1}{\lambda_{ii}}\right) \tag{17.6}$$

For a more elaborate proof of the same result with extensions to other conditionals and distributions, see Chapter 18.

17.2 The Gamma kernel, function and distribution

Since the Gaussian kernel depends only on the square of its argument, it can be reparametrized with $y = x^2$ as:

$$\mathrm{e}^{-\frac{\lambda}{2} x^2} \, \mathrm{d}x = \frac{1}{2} y^{\frac{1}{2}} \mathrm{e}^{-\frac{\lambda}{2} y} \, \mathrm{d}y \tag{17.7}$$

revealing a new kernel on $\mathbb{R}^+$ closely associated with Gaussian distributions.

The gamma kernel is defined for any $\alpha > 0$ on $\mathbb{R}^+_*$ as:

$$x \longmapsto x^{\alpha - 1} \mathrm{e}^{-x} \tag{17.8}$$

The gamma function is then defined on $\mathbb{R}^*_+$ by its integral for $x > 0$, as a function of α:

$$\forall \alpha > 0, \quad \Gamma(\alpha) = \int_0^{+\infty} x^{\alpha - 1} \mathrm{e}^{-x} \, \mathrm{d}x \tag{17.9}$$

Integration by parts readily yields:

$$\begin{aligned}
\Gamma(\alpha + 1) &= \int_0^{+\infty} x^{\alpha} \mathrm{e}^{-x} \, \mathrm{d}x & (17.10)\\
&= -[x^{\alpha} \mathrm{e}^{-x}]_0^{+\infty} + \int_0^{+\infty} \alpha x^{\alpha - 1} \mathrm{e}^{-x} \, \mathrm{d}x & (17.11)\\
&= \alpha \Gamma(\alpha) & (17.12)
\end{aligned}$$

Note that $\Gamma(1) = 1$ by direct antidifferentiation.

By definition of the gamma function, for $\alpha > 0$,

$$\frac{1}{\Gamma(\alpha)} x^{\alpha - 1} \mathrm{e}^{-x} \mathbb{1}_{x > 0} \, \mathrm{d}x \tag{17.13}$$

is a well-normalized continuous distribution. It is usually extended with a rate parameter $\beta > 0$ through the substitution $x \leftrightarrow \beta x$, giving the two-parameter gamma distribution, that we'll denote $\mathrm{Ga}(\alpha, \beta)$:

$$\frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} \exp(-\beta x) \mathbb{1}_{x>0} \, \mathrm{d}x \tag{17.14}$$

Based on the derivation in the introduction, if

$$X \sim \mathcal{N}(0, \lambda^{-1}) \tag{17.15}$$

then

$$X^2 \sim \mathrm{Ga}\left(\frac{1}{2}, \frac{\lambda}{2}\right) \tag{17.16}$$

17.3 Sum and normalization of two random variables, beta kernel, function and distribution

We consider the measure on $\mathbb{R}^{+2}$ $\mathrm{d}x\,\mathrm{d}y$ where x and y are the linear projection on the first and second coordinate. Any point of $\mathbb{R}^{+2}$ can be instead identified based on the two functions:

$$\mathbb{R}^+, \mathbb{R}^+ \longrightarrow \mathbb{R}^+, [0, 1] \tag{17.17}$$

$$x, y \mapsto s(x, y) = x + y, u(x, y) = \frac{x}{x+y} \tag{17.18}$$

Any measure expressed as a function of $\mathrm{d}x\,\mathrm{d}y$ can instead be re-expressed as a function of $\mathrm{d}u\,\mathrm{d}s$:

$$f(x, y)\,\mathrm{d}x\,\mathrm{d}y = f(x, s-x)\,\mathrm{d}x\,\mathrm{d}s \tag{17.19}$$

$$= f(us, [1-u]s)s\,\mathrm{d}u\,\mathrm{d}s \tag{17.20}$$

This change of variable links the gamma and beta kernels, since the measure defined by two independent gamma variables with shapes α and β and common rate 1 is proportional to:

$$x^{\alpha-1}\mathrm{e}^{-x}y^{\beta-1}\mathrm{e}^{-y}\,\mathrm{d}x\,\mathrm{d}y = (us)^{\alpha-1}\mathrm{e}^{-us}([1-u]s)^{\beta-1}\mathrm{e}^{-[1-u]s}s\,\mathrm{d}u\,\mathrm{d}s \tag{17.21}$$

$$= u^{\alpha-1}(1-u)^{\beta-1}s^{\alpha+\beta-1}\mathrm{e}^{-s}\,\mathrm{d}u\,\mathrm{d}s \tag{17.22}$$

Which can be recognized to be the product of a gamma kernel with an other kernel, namely the beta kernel defined for $\alpha > 0, \beta > 0$ by:

$$u \mapsto u^{\alpha-1}(1-u)^{\beta-1} \tag{17.23}$$

The integral of this kernel over $[0,1]$ defines the beta function:

$$\forall \alpha > 0, \beta > 0, \quad \mathrm{B}(\alpha, \beta) = \int_0^1 u^{\alpha-1}(1-u)^{\beta-1}\,\mathrm{d}u \tag{17.24}$$

and dividing the beta kernel with the beta function naturally gives rise to the density function of the beta distribution $\mathrm{Beta}(\alpha, \beta)$ on $[0,1]$:

$$\frac{1}{\mathrm{B}(\alpha,\beta)} u^{\alpha-1}(1-u)^{\beta-1}\mathbb{1}_{0<u<1}\,\mathrm{d}u \tag{17.25}$$

Therefore, in common probabilistic language, when X and Y are two independent gamma-distributed random variables with shapes α, β and rate 1, then $S = X + Y$ and $U = \frac{X}{X+Y}$ are two independent random variables following a $\mathrm{Beta}(\alpha, \beta)$ and a $\mathrm{Ga}(\alpha+\beta, 1)$ distribution respectively.

By integrating these kernels over all of $\mathbb{R}^{+2}$, we also obtain the identity between their normalization constants:

$$\Gamma(\alpha)\Gamma(\beta) = \mathrm{B}(\alpha, \beta)\Gamma(\alpha+\beta) \tag{17.26}$$

which gives the useful expression for beta:

$$\mathrm{B}(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{17.27}$$

Therefore, the functional equation (17.12) of the gamma function translates to:

$$\begin{aligned} \mathrm{B}(\alpha+1, \beta) &= \frac{\Gamma(\alpha+1)\Gamma(\beta)}{\Gamma(\alpha+\beta)} && (17.28) \\ &= \frac{\alpha}{\alpha+\beta}\mathrm{B}(\alpha, \beta) && (17.29) \end{aligned}$$

Since we know how to derive gamma distributions by squaring Gaussian random variables, the properties above apply to squares of Gaussians. Let $X_1, \ldots, X_n$ be n independent random variable of distribution $\mathcal{N}(0, \lambda^{-1})$. We already knew that their squares are independent $X_1^2, \ldots, X_n^2 \sim \mathrm{Ga}\left(\frac{1}{2}, \frac{\lambda}{2}\right)$. But further, for any $1 \leqslant k \leqslant n$, $\sum_{j=1}^k X_j^2$ and $\sum_{j=k+1}^n X_j^2$ are independent with distributions $\mathrm{Ga}\left(\frac{k}{2}, \frac{\lambda}{2}\right)$ and $\mathrm{Ga}\left(\frac{n-k}{2}, \frac{\lambda}{2}\right)$, and therefore:

$$\frac{\sum_{j=1}^k X_j^2}{\sum_{j=1}^n X_j^2} \sim \mathrm{Beta}\left(\frac{k}{n}, \frac{n-k}{n}\right) \tag{17.30}$$

17.4 Student's t distribution and the normal distribution

Student's t distribution frequently arises as an over-dispersed normal distribution obtained through compounding, and has been repetitively been discovered and re-discovered by statisticians [Stu08]. Here, we will briefly derive a select few properties that we need, and introduce some custom parametrization that we rely on in some places. Namely, if

$$X|\lambda \sim \mathcal{N}(0, \lambda^{-1}) \tag{17.31}$$

$$\lambda \sim \mathrm{Ga}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.32}$$

or, equivalently, if

$$X = \frac{U}{\sqrt{G}} \tag{17.33}$$

$$U \sim \mathcal{N}(0, 1) \tag{17.34}$$

$$G \sim \mathrm{Ga}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.35}$$

then X has the marginal density:

$$p_X(x) \propto \int_{\lambda>0} \lambda^{\frac{1}{2}} \mathrm{e}^{-\frac{1}{2}\lambda x^2} \lambda^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\lambda} \,\mathrm{d}\lambda \tag{17.36}$$

$$\propto \left(\frac{1}{2} + \frac{1}{2}x^2\right)^{-\frac{\nu+1}{2}} \tag{17.37}$$

$$\propto \left(1 + x^2\right)^{-\frac{\nu+1}{2}} \tag{17.38}$$

Based on this, one could define a plain t-distribution for $\nu > 0$ by:

$$\mathfrak{T}(x|\nu) \,\mathrm{d}x = \frac{1}{\tau(\nu)} (1 + x^2)^{-\frac{\nu+1}{2}} \,\mathrm{d}x \tag{17.39}$$

where

$$\tau(\nu) := \int_{-\infty}^{+\infty} \left(1 + x^2\right)^{-\frac{\nu+1}{2}} \,\mathrm{d}x \tag{17.40}$$

However, the most common parametrization is the rescaled ($x \leftrightarrow \frac{x}{\sqrt{\nu}}$) Student's t distribution:

$$t(x|\nu) \,\mathrm{d}x = \frac{1}{\sqrt{\nu}\tau(\nu)} \left(1 + \frac{1}{\nu}x^2\right)^{-\frac{\nu+1}{2}} \,\mathrm{d}x \tag{17.41}$$

The latter is obtained if we define a scaled $Y = \sqrt{\nu}X$ variable:

$$Y = \frac{\sqrt{\nu}X}{\sqrt{G}} = \frac{U}{\sqrt{\frac{1}{\nu}G}} \sim t(\nu) \tag{17.42}$$

where $\frac{1}{\nu}G \sim \mathrm{Ga}\left(\frac{\nu}{2}, \frac{\nu}{2}\right)$ is a scaled gamma distribution that has expectation 1.

Notably, if $X_0, \ldots, X_n$ are independent standard normal variables,

$$\frac{X_0}{\sqrt{\frac{1}{n}\sum_{i=1}^{n} X_i^2}} \sim t(n) \tag{17.43}$$

Many properties of the t-distribution are not easily deduced from this construction. Instead, it helps to draw a link with the beta distribution which was also obtained through a ratio. Especially,

$$X^2 = \frac{U^2}{G} \tag{17.44}$$

is the ratio of independent $\mathrm{Ga}\left(\frac{1}{2}, \frac{1}{2}\right)$ and $\mathrm{Ga}\left(\frac{\nu}{2}, \frac{1}{2}\right)$ random variables, which makes this t^2 distribution very close to the construction of beta. All that's missing is the transformation:

$$\frac{1}{1+X^2} = \frac{1}{1+\frac{U^2}{G}} \tag{17.45}$$

$$= \frac{G}{G+U^2} \tag{17.46}$$

This explains why the beta and plain t-kernel are easily linked by the change of variable $u(x) = \frac{1}{1+x^2}$:

$$\left(1+x^2\right)^{-\frac{\nu+1}{2}} \mathrm{d}x = u^{\frac{\nu+1}{2}} \mathrm{d}\sqrt{\frac{1-u}{u}} \tag{17.47}$$

$$= \frac{1}{2} u^{\frac{\nu+1}{2}} u^{-\frac{3}{2}} (1-u)^{-\frac{1}{2}} \mathrm{d}u \tag{17.48}$$

$$= \frac{1}{2} u^{\frac{\nu}{2}-1} (1-u)^{\frac{1}{2}-1} \mathrm{d}u \tag{17.49}$$

from which we can see again that

$$\frac{1}{1+X^2} \sim \mathrm{Beta}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.50}$$

and by integrating the kernel for $x = 0$ to $+\infty$, u likewise,

$$\tau(\nu) = \mathrm{B}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.51}$$

The functional equation of the gamma and beta functions can be this time translated to:

$$\tau(\nu+2) = \mathrm{B}\left(\frac{\nu}{2}+1, \frac{1}{2}\right) \tag{17.52}$$

$$= \frac{\frac{\nu}{2}}{\frac{\nu+1}{2}} \mathrm{B}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.53}$$

$$= \frac{\nu}{\nu+1}\tau(\nu) \tag{17.54}$$

The identities above motivates a closer look at a specific value of the beta function, namely $\mathrm{B}\left(\frac{1}{2}, \frac{1}{2}\right)$, which can be solved with the change of variable $u = \cos^2\theta$:

$$\mathrm{B}\left(\frac{1}{2}, \frac{1}{2}\right) = \int_0^1 \frac{\mathrm{d}u}{\sqrt{u(1-u)}} \tag{17.55}$$

$$= \int_0^{\frac{\pi}{2}} \frac{2\cos\theta\sin\theta\,\mathrm{d}\theta}{\sqrt{\cos^2\theta\sin^2\theta}} \tag{17.56}$$

$$= \pi \tag{17.57}$$

This gives the related identities:

$$\mathrm{B}\left(\frac{1}{2}, \frac{1}{2}\right) = \pi \tag{17.58}$$

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\Gamma\left(\frac{1}{2}\right)\Gamma\left(\frac{1}{2}\right)} \tag{17.59}$$

$$= \sqrt{\mathrm{B}\left(\frac{1}{2}, \frac{1}{2}\right)\Gamma(1)} \tag{17.60}$$

$$= \sqrt{\pi} \tag{17.61}$$

$$\tau(\nu) = \mathrm{B}\left(\frac{\nu}{2}, \frac{1}{2}\right) \tag{17.62}$$

$$= \sqrt{\pi}\frac{\Gamma\left(\frac{\nu}{2}\right)}{\Gamma\left(\frac{\nu+1}{2}\right)} \tag{17.63}$$

The variance of a plain-t distribution would exist for $\nu > 2$ ($\nu \leqslant 2$ means the tails are too heavy for the variance to be finite), and could be computed as:

$$\frac{1}{\tau(\nu)}\int x^2(1+x^2)^{-\frac{\nu+1}{2}}\,\mathrm{d}x \tag{17.64}$$
$$= \frac{1}{\tau(\nu)}\int (1+x^2-1)(1+x^2)^{-\frac{\nu+1}{2}}\,\mathrm{d}x \tag{17.65}$$
$$= \frac{1}{\tau(\nu)}\left(\int (1+x^2)^{-\frac{\nu-2+1}{2}}\,\mathrm{d}x - \int (1+x^2)^{-\frac{\nu+1}{2}}\right)\mathrm{d}x \tag{17.66}$$
$$= \frac{\tau(\nu-2)}{\tau(\nu)} - 1 \tag{17.67}$$

using (17.54):

$$= \frac{\nu-1}{\nu-2} - 1 \tag{17.68}$$
$$= \frac{1}{\nu-2} \tag{17.69}$$

This gives the expression of Student's t with the expression of τ as a function of Γ fully expanded:

$$t(x|\nu)\,\mathrm{d}x = \frac{\Gamma\left(\frac{\nu+1}{2}\right)}{\sqrt{\pi\nu}\,\Gamma\left(\frac{\nu}{2}\right)}\left(1+\frac{1}{\nu}x^2\right)^{-\frac{\nu+1}{2}}\mathrm{d}x \tag{17.70}$$

which has, accordingly, variance $\frac{\nu}{\nu-2}$ when $\nu > 2$.

The link between the t and beta distributions is also crucial for convenient calculation of quantiles. If X follows a plain-t distribution, for $x \in \mathbb{R}$,

$$|X| \geqslant |x| \iff \frac{1}{1+X^2} \leqslant \frac{1}{1+x^2} \tag{17.71}$$

which allow us to express:

$$P(|X| \geqslant |x|) = \frac{\mathrm{B}\left(\frac{1}{1+x^2}; \frac{\nu}{2}, \frac{1}{2}\right)}{\mathrm{B}\left(\frac{\nu}{2}, \frac{1}{2}\right)} \tag{17.72}$$

where the right-hand term features the regularized incomplete beta function.

For Y following a Student's t, we merely need to get the extra scaling right:

$$|Y| \geqslant |y| \iff \frac{|Y|}{\sqrt{\nu}} \geqslant \frac{|y|}{\sqrt{\nu}} \tag{17.73}$$
$$P(|Y| \geqslant |y|) = \frac{\mathrm{B}\left(\frac{\nu}{\nu+y^2}; \frac{\nu}{2}, \frac{1}{2}\right)}{\mathrm{B}\left(\frac{\nu}{2}, \frac{1}{2}\right)} \tag{17.74}$$

17.5 The multivariate Gamma function and the Wishart distribution

Extensions of gamma distributions to matrix variables, which are particularly important for Bayesian handling of covariance and precision matrices, have been described by various authors [Wis28; And03]. We however offer our own construction of the distribution here, with a somewhat more compact presentation and proofs that we have found expressed in terms of individual coefficients, while we favor the use of matrix variates in integration, differentiation and characteristic functions as much as possible. The extension to improper distributions that we propose at the end is, to our knowledge, original.

We consider, for $\mathbf{\Lambda}$ a symmetric positive definite (s.p.d.) matrix of dimension (n, n), kernels of the form

$$|\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{\Lambda})) \tag{17.75}$$

we want to find out for which value of α this kernel stays integrable over SPD matrices, and what is the value of the corresponding integral.

This requires first to clarify which base measure is used to define it. There are indeed two natural choices of measure on the space of symmetric matrices S_n. One consists in identifying directly S_n to $\mathbb{R}^{\frac{1}{2}n(n+1)}$ through the coordinate system $\mathbf{e}_i\mathbf{e_i}', 1 \leqslant i \leqslant n, \mathbf{e}_i\mathbf{e}_j' + \mathbf{e}_j\mathbf{e}_i', 1 \leqslant i < j \leqslant n$, in which case $\mathbf{e}_i\mathbf{e}_j' + \mathbf{e}_j\mathbf{e}_i'$ is considered to be of unit length. The second option consists in first identifying the matrix space to $\mathbb{R}^{n^2}$, then taking the geometry induced on the subspace S_n. In that case, $\mathbf{e}_i\mathbf{e}_j' + \mathbf{e}_j\mathbf{e}_i'$ is of norm $\sqrt{2}$. Here, we'll use the first option:

$$\mathrm{d}\mathbf{\Lambda} = \prod_{j \leqslant i} \mathrm{d}\Lambda_{ij} \tag{17.76}$$

Note that the scalar product on symmetric matrices naturally associated with this measure is then

$$\mathbf{A}, \mathbf{B} \mapsto \sum_{j \leqslant i} A_{ij} B_{ij} \tag{17.77}$$

and *not* the equally common

$$\mathbf{A}, \mathbf{B} \mapsto \mathrm{Tr}(\mathbf{A'B}) = \sum_{i,j} A_{ij} B_{ij} \tag{17.78}$$

This being defined, we need the change of variable $\mathbf{\Lambda} = \mathbf{LL}'$ where $\mathbf{L}$ spans T_n^-, the space of lower triangular matrices of shape (n, n). We have the differential

$$\mathrm{d}\mathbf{\Lambda} = \mathbf{L}\,\mathrm{d}\mathbf{L}' + \mathrm{d}\mathbf{L}\mathbf{L}' \tag{17.79}$$

To get the corresponding determinant, we can evaluate it on an orthonormal basis of T_n^-. Given the orthonormal basis $\mathbf{e}_i, i \in \{1, \dots, n\}$ of $\mathbb{R}^n$, $\mathbf{e}_i\mathbf{e}_j', 0 \leqslant j \leqslant i \leqslant n$ forms a orthonormal basis of T_n^- (for the usual scalar product $\mathbf{L}, \mathbf{M} \mapsto \mathrm{Tr}(\mathbf{L}'\mathbf{M})$, this time the two candidate scalar products above coincide).

We get for $j \leqslant i$:

$$\mathbf{L}(\mathbf{e}_i\mathbf{e}_j')' + (\mathbf{e}_i\mathbf{e}_j')\mathbf{L}' = \mathbf{L}\mathbf{e}_j\mathbf{e}_i' + \mathbf{e}_i(\mathbf{L}\mathbf{e}_j)' \tag{17.80}$$

$$= \sum_{j \leqslant k} L_{kj} \left(\mathbf{e}_k\mathbf{e}_i' + \mathbf{e}_i\mathbf{e}_{k'}\right) \tag{17.81}$$

We observe that the matrix of the differential in the given bases is thus block diagonal (with one block for each value of i) with each block being triangular. For each block indexed by i, the diagonal coefficients are L_{jj} for $j < i$, and $2L_{ii}$ for $j = i$. We therefore get the wanted determinant and change of variable as:

$$\mathrm{d}\mathbf{\Lambda} = \prod_i 2 \prod_{j \leqslant i} L_{jj}\, \mathrm{d}\mathbf{L} \tag{17.82}$$

$$= \prod_i 2 \prod_{j \leqslant i} L_{jj}\, \mathrm{d}\mathbf{L} \tag{17.83}$$

$$= 2^n \prod_j \prod_{i \geqslant j} L_{jj}\, \mathrm{d}\mathbf{L} \tag{17.84}$$

$$= 2^n \prod_i L_{ii}^{n-i+1}\, \mathrm{d}L \tag{17.85}$$

We can then compute

$$\int_{\mathbf{\Lambda}\,\mathrm{spd}} |\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{\Lambda}))\, \mathrm{d}\mathbf{\Lambda} \tag{17.86}$$

$$= \int_{\mathbf{L} \in T_n^-} |\mathbf{L}|^{2(\alpha-1)} \exp(-\mathrm{Tr}(\mathbf{L}L'))2^n \prod_i L_{ii}^{n-i+1}\, \mathrm{d}\mathbf{L} \tag{17.87}$$

$$= \left(\prod_{j<i} \int_{-\infty}^{+\infty} \exp(-L_{ij}^2)\, \mathrm{d}L_{ij}\right)\left(\prod_i 2 \int_0^{+\infty} L_{ii}^{2(\alpha-1)+n-i+1} \exp(-L_{ii}^2)\, \mathrm{d}L_{ii}\right) \tag{17.88}$$

$$= \left(\int_0^{+\infty} t^{-\frac{1}{2}} \exp(-t)\, \mathrm{d}t\right)^{\frac{n(n-1)}{2}} \left(\prod_i \int_0^{+\infty} t^{\alpha-1+\frac{n-i+1}{2}-\frac{1}{2}} \exp(-t)\, \mathrm{d}t\right) \tag{17.89}$$

$$= \pi^{\frac{n(n-1)}{4}} \prod_{i=1}^{n} \Gamma\left(\alpha + \frac{n-i}{2}\right) \tag{17.90}$$

$$= \pi^{\frac{n(n-1)}{4}} \prod_{i=0}^{n-1} \Gamma\left(\alpha + \frac{i}{2}\right) \tag{17.91}$$

which stays finite if and only if $\alpha > 0$.

This results suggests the definition of a "natural", "ascending" generalization of the gamma function:

$$\Gamma_{\overline{n}}(\alpha) = \int |\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}\mathbf{\Lambda})\, \mathrm{d}\mathbf{\Lambda} = \pi^{\frac{n(n-1)}{4}} \prod_{i=0}^{n-1} \Gamma\left(\alpha + \frac{i}{2}\right) \tag{17.92}$$

defined for $\alpha > 0$, but in practice one often encounters the alternative "descending" parametrization:

$$\Gamma_{\underline{n}}(\alpha) = \int |\mathbf{\Lambda}|^{\alpha-\frac{n+1}{2}} \exp(-\mathrm{Tr}\mathbf{\Lambda})\, \mathrm{d}\mathbf{\Lambda} = \Gamma_{\overline{n}}\left(\alpha - \frac{n-1}{2}\right) \tag{17.93}$$

$$= \pi^{\frac{n(n-1)}{4}} \prod_{i=0}^{n-1} \Gamma\left(\alpha - \frac{i}{2}\right) \tag{17.94}$$

which is therefore defined for $\alpha > \frac{n-1}{2}$.

Conjecture: The surface (total induced measure) of the unitary group $U(n)$ is

$$\frac{2^n \pi^{\frac{n^2}{2}}}{\Gamma_{\overline{n}}\left(\frac{1}{2}\right)} \tag{17.95}$$

Given a SPD matrix $\mathbf{G} = \mathbf{M}\mathbf{M}'$, $\mathbf{M} \in T_n^-$, we can also rescale $\mathbf{L}$ and $\mathbf{\Lambda}$ as

$$\mathrm{d}\mathbf{M}\mathbf{L} = \prod_{i=1}^{n} M_{ii}^{i}\, \mathrm{d}\mathbf{L} \tag{17.96}$$

which yields the conjugation

$$\mathrm{d}\mathbf{M}\mathbf{\Lambda}\mathbf{M}' = 2^n \prod_i (\mathbf{M}\mathbf{L})_{ii}^{n-i+1}\, \mathrm{d}\mathbf{M}\mathbf{L} \tag{17.97}$$

$$= 2^n \prod_i M_{ii}^{n-i+1} L_{ii}^{n-i+1}\, \mathrm{d}\mathbf{M}\mathbf{L} \tag{17.98}$$

$$= 2^n \prod_i M_{ii}^{n-i+1} L_{ii}^{n-i+1} \prod_i M_{ii}^{i}\, \mathrm{d}\mathbf{L} \tag{17.99}$$

$$= \prod_i M_{ii}^{n+1}\, \mathrm{d}\mathbf{\Lambda} \tag{17.100}$$

$$= |\mathbf{M}|^{n+1}\, \mathrm{d}\mathbf{\Lambda} \tag{17.101}$$

This in turns gives the more general gamma-like integral:

$$\int |\mathbf{G}\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{G}\mathbf{\Lambda}))\, \mathrm{d}\mathbf{\Lambda} = |\mathbf{G}|^{-\frac{n+1}{2}} \int |\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{\Lambda}))\, \mathrm{d}\mathbf{\Lambda} \tag{17.102}$$

which we can also write

$$\int |\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{G}\mathbf{\Lambda}))\,\mathrm{d}\mathbf{\Lambda} = |\mathbf{G}|^{-\frac{2\alpha+n-1}{2}} \int |\mathbf{\Lambda}|^{\alpha-1} \exp(-\mathrm{Tr}(\mathbf{\Lambda}))\,\mathrm{d}\mathbf{\Lambda} \tag{17.103}$$

$$= |\mathbf{G}|^{-\frac{2\alpha+n-1}{2}} \pi^{\frac{n(n-1)}{4}} \prod_{i=0}^{n-1} \Gamma\left(\alpha + \frac{i}{2}\right) \tag{17.104}$$

To make this into a distribution, we can add two small convenience reparametrizations: scale the kernel by a factor two so that the $L_{ij}, j < i$ have a standard Gaussian distribution, and set $\alpha = \frac{\nu}{2}$ where ν is can be interpreted as a number of degrees of freedom in some contexts.

If $L_{ij}, j \leqslant i$ are independent random variables such that:

$$L_{ij} \sim \mathcal{N}(0,1), \qquad 1 \leqslant j < i \leqslant n \tag{17.105}$$

$$L_{ii}^2 \sim \chi^2(\nu + n - i), \qquad 1 \leqslant i \leqslant n \tag{17.106}$$

then $\mathbf{\Lambda} = \mathbf{L}\mathbf{L}'$, with the measure $\mathrm{d}\mathbf{\Lambda}$ defined above, has the density

$$\frac{|\mathbf{\Lambda}|^{\frac{\nu}{2}-1} \exp\left(-\frac{1}{2}\mathrm{Tr}\mathbf{\Lambda}\right)}{2^{\frac{n(\nu+n-1)}{2}} \Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} = \frac{|\mathbf{\Lambda}|^{\frac{\nu}{2}-1} \exp\left(-\frac{1}{2}\mathrm{Tr}\mathbf{\Lambda}\right)}{\pi^{\frac{n(n-1)}{4}} 2^{\frac{n(\nu+n-1)}{2}} \prod_{i=0}^{n-1} \Gamma\left(\frac{\nu+i}{2}\right)} \tag{17.107}$$

which is also known has a Wishart with $\nu + n - 1 > n - 1$ degrees of freedom.

Its mean can easily be shown to be

$$\mathbb{E}\left[\Lambda_{ij}\right] = \sum_{k \leqslant \min i,j} \mathbb{E}\left[L_{ik} L_{jk}\right] \tag{17.108}$$

$$= \delta_{ij} \sum_{k \leqslant i} \mathbb{E}\left[L_{ik}^2\right] \tag{17.109}$$

$$= \delta_{ij} \left(\sum_{k<i} 1 + \mathbb{E}\left[L_{ii}^2\right] \right) \tag{17.110}$$

$$= \delta_{ij}\left[(i-1) + (\nu + n - i)\right] \tag{17.111}$$

$$= (\nu + n - 1)\delta_{ij} \tag{17.112}$$

i.e.

$$\mathbb{E}[\mathbf{\Lambda}] = (\nu + n - 1) I_n \tag{17.113}$$

To add a scaling, given once again $\mathbf{G} = \mathbf{M}'\mathbf{M}$ s.p.d., $\mathbf{M}$ lower triangular, $\mathbf{G}^{-1} =$

$\mathbf{M}^{-1}\mathbf{M}^{-1\prime}$ and $\mathbf{\Lambda}$ as previously, we can define $\tilde{\mathbf{\Lambda}} = \mathbf{M}^{-1}\mathbf{\Lambda}\mathbf{M}^{-1\prime}$. Then

$$\begin{aligned}
p_{\mathbf{\Lambda}}(\mathbf{\Lambda})\,\mathrm{d}\mathbf{\Lambda} &= p_{\mathbf{\Lambda}}(\mathbf{\Lambda})|\mathbf{M}|^{n+1}\,\mathrm{d}\mathbf{M}^{-1}\mathbf{\Lambda}\mathbf{M}^{-1\prime} && (17.114)\\
&= p_{\mathbf{\Lambda}}(\mathbf{M}\tilde{\mathbf{\Lambda}}\mathbf{M}')|\mathbf{G}|^{\frac{n+1}{2}}\,\mathrm{d}P && (17.115)\\
&= \frac{1}{2^{\frac{n(\nu+n-1)}{2}}\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)}|\mathbf{M}\tilde{\mathbf{\Lambda}}\mathbf{M}'|^{\frac{\nu}{2}-1}\exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{M}\tilde{\mathbf{\Lambda}}\mathbf{M}')\right)|G|^{\frac{n+1}{2}}\,\mathrm{d}\tilde{\mathbf{\Lambda}} && (17.116)\\
&= \frac{|\frac{1}{2}\mathbf{G}|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)}|\tilde{\mathbf{\Lambda}}|^{\frac{\nu}{2}-1}\exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\mathbf{\Lambda}})\right)\,\mathrm{d}\tilde{\mathbf{\Lambda}} && (17.117)
\end{aligned}$$

and $\tilde{\mathbf{\Lambda}}$ with this density has mean

$$\begin{aligned}
\mathbb{E}\left[\tilde{\mathbf{\Lambda}}\right] &= \mathbb{E}\left[\mathbf{M}^{-1}\mathbf{\Lambda}\mathbf{M}^{-1\prime}\right] && (17.118)\\
&= \mathbf{M}^{-1}\mathbb{E}[\mathbf{\Lambda}]\,\mathbf{M}^{-1\prime} && (17.119)\\
&= \mathbf{M}^{-1}(\nu+n-1)I_n\mathbf{M}^{-1\prime} && (17.120)\\
&= (\nu+n-1)\mathbf{G}^{-1} && (17.121)
\end{aligned}$$

Mean and characteristic function. Note that the mean can also be recovered from the characteristic function of $\tilde{\mathbf{\Lambda}}$, namely

$$\begin{aligned}
\phi_{\tilde{\mathbf{\Lambda}}}(\mathbf{K}) &= \mathbb{E}\left[\mathrm{e}^{i\mathrm{Tr}(\mathbf{K}'\tilde{\mathbf{\Lambda}})}\right] && (17.122)\\
&= \frac{\int|\tilde{\mathbf{\Lambda}}|^{\frac{\nu}{2}-1}\exp\left(-\frac{1}{2}\mathrm{Tr}\left[(\mathbf{G}-2i\mathbf{K})'\tilde{\mathbf{\Lambda}}\right]\right)\,\mathrm{d}\tilde{\mathbf{\Lambda}}}{\int|\tilde{\mathbf{\Lambda}}|^{\frac{\nu}{2}-1}\exp\left(-\frac{1}{2}\mathrm{Tr}\left[\mathbf{G}'\tilde{\mathbf{\Lambda}}\right]\right)\,\mathrm{d}\tilde{\mathbf{\Lambda}}} && (17.123)\\
&= \left(\frac{|\mathbf{G}|}{|\mathbf{G}-2i\mathbf{K}|}\right)^{\frac{\nu+n-1}{2}} && (17.124)\\
&= |I_n - 2i\mathbf{K}\mathbf{G}^{-1}|^{-\frac{\nu+n-1}{2}} && (17.125)
\end{aligned}$$

from which one can recover

$$\begin{aligned}
\mathbb{E}\left[\tilde{\mathbf{\Lambda}}\right] &= -i\nabla\phi_{\tilde{\mathbf{\Lambda}}}(\mathbf{K})\Big|_{\mathbf{K}=0} && (17.126)\\
&= -i\times\left(-\frac{\nu+n-1}{2}\right)\times 1\times I_n^{-1}\left(-2i\mathbf{G}^{-1}\right) && (17.127)\\
&= (\nu+n-1)\mathbf{G}^{-1} && (17.128)
\end{aligned}$$

Inverse mean. The expectation $\mathbb{E}\left[\tilde{\boldsymbol{\Lambda}}^{-1}\right]$ can be obtained by integration by part over each coordinate. Indeed, we have the gradients:

$$\nabla_{\tilde{\boldsymbol{\Lambda}}} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\boldsymbol{\Lambda}})} = -\frac{1}{2}\mathbf{G}\,\mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\boldsymbol{\Lambda}})} \tag{17.129}$$

$$\nabla_{\tilde{\boldsymbol{\Lambda}}} |\tilde{\boldsymbol{\Lambda}}|^{\frac{\nu}{2}-1} = \left(\frac{\nu}{2}-1\right)\tilde{\boldsymbol{\Lambda}}^{-1}\,|\tilde{\boldsymbol{\Lambda}}|^{\frac{\nu}{2}-1} \tag{17.130}$$

therefore if $\nu > 2$, since the integrands vanish at all the boundaries,

$$\int \tilde{\boldsymbol{\Lambda}}^{-1}\,|\tilde{\boldsymbol{\Lambda}}|^{\frac{\nu}{2}-1}\mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\boldsymbol{\Lambda}})}\,\mathrm{d}\boldsymbol{\Lambda} \tag{17.131}$$

$$= -\left(\frac{\nu}{2}-1\right)^{-1}\int |\tilde{\boldsymbol{\Lambda}}|^{\frac{\nu}{2}-1}\left(-\frac{1}{2}\mathbf{G}\right)\mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\boldsymbol{\Lambda}})}\,\mathrm{d}\boldsymbol{\Lambda} \tag{17.132}$$

$$= \frac{1}{\nu-2}\mathbf{G}\int |\tilde{\boldsymbol{\Lambda}}|^{\frac{\nu}{2}-1}\mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\tilde{\boldsymbol{\Lambda}})}\,\mathrm{d}\boldsymbol{\Lambda} \tag{17.133}$$

from which we deduce

$$\mathbb{E}\left[\tilde{\boldsymbol{\Lambda}}^{-1}\right] = \frac{1}{\nu-2}\,\mathbf{G} \tag{17.134}$$

17.6 Wishart-Normal mixture

If $\boldsymbol{\Lambda}$ has the density

$$p_{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda}) = \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)}|\boldsymbol{\Lambda}|^{\frac{\nu}{2}-1}\exp\left(-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\boldsymbol{\Lambda})\right) \tag{17.135}$$

and $x \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}^{-1})$, then x has the density

$$p_x(\mathbf{x}) = \int \frac{|\mathbf{\Lambda}|^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}} \mathrm{e}^{-\frac{1}{2}\mathbf{x}'\mathbf{\Lambda}\mathbf{x}} \cdot \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} |\mathbf{\Lambda}|^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(G\mathbf{\Lambda})} \,\mathrm{d}\mathbf{\Lambda} \tag{17.136}$$

$$= \frac{1}{(2\pi)^{\frac{n}{2}}} \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \int |\mathbf{\Lambda}|^{\frac{\nu+1}{2}-1} \exp\left(-\frac{1}{2}\mathrm{Tr}((\mathbf{G}+\mathbf{x}\mathbf{x}')\mathbf{\Lambda})\right) \mathrm{d}\mathbf{\Lambda} \tag{17.137}$$

$$= \frac{\Gamma_{\overline{n}}\left(\frac{\nu+1}{2}\right)}{(2\pi)^{\frac{n}{2}}\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\left|\frac{1}{2}(\mathbf{G}+\mathbf{x}\mathbf{x}')\right|^{\frac{\nu+n}{2}}} \tag{17.138}$$

$$= \frac{\Gamma_{\overline{n}}\left(\frac{\nu+1}{2}\right)}{\pi^{\frac{n}{2}}\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \frac{|\mathbf{G}|^{\frac{\nu+n-1}{2}}}{|\mathbf{G}+\mathbf{x}\mathbf{x}'|^{\frac{\nu+n}{2}}} \tag{17.139}$$

$$= \frac{1}{\pi^{\frac{n}{2}}} \frac{\prod_{i=0}^{n-1}\Gamma\left(\frac{\nu+i+1}{2}\right)}{\prod_{i=0}^{n-1}\Gamma\left(\frac{\nu+i}{2}\right)} \frac{|\mathbf{G}|^{\frac{\nu+n-1}{2}}}{|\mathbf{G}+\mathbf{x}\mathbf{x}'|^{\frac{\nu+n}{2}}} \tag{17.140}$$

$$= \frac{1}{\pi^{\frac{n}{2}}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \frac{|\mathbf{G}|^{\frac{\nu+n-1}{2}}}{|\mathbf{G}+\mathbf{x}\mathbf{x}'|^{\frac{\nu+n}{2}}} \tag{17.141}$$

$$= \frac{|\mathbf{G}|^{-\frac{1}{2}}}{\pi^{\frac{n}{2}}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left|I_n + \mathbf{x}\mathbf{x}'\mathbf{G}^{-1}\right|^{-\frac{\nu+n}{2}} \tag{17.142}$$

$$= \frac{|\mathbf{G}|^{-\frac{1}{2}}}{\pi^{\frac{n}{2}}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(1+\mathbf{x}'\mathbf{G}^{-1}\mathbf{x}\right)^{-\frac{\nu+n}{2}} \tag{17.143}$$

which can be recognized to be a multivariate t distribution with ν degrees of freedom and scale matrix $\frac{1}{\nu}\mathbf{G}$.

Interestingly, if $\lambda \sim \mathrm{Ga}\left(\frac{\nu}{2}, \frac{1}{2}\right)$, $\mathbf{\Lambda} = \lambda\mathbf{G}^{-1}$ and $x \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}^{-1})$, x also has the marginal density

$$p_x(\mathbf{x}) = \int \frac{|\mathbf{\Lambda}|^{\frac{1}{2}}}{(2\pi)^{\frac{n}{2}}} \mathrm{e}^{-\frac{1}{2}\lambda\mathbf{x}'\mathbf{G}^{-1}\mathbf{x}} \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)} \lambda^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\lambda} \,\mathrm{d}\lambda \tag{17.144}$$

$$= \frac{|\mathbf{G}|^{-\frac{1}{2}}}{2^{\frac{n+\nu}{2}}\pi^{\frac{n}{2}}\Gamma\left(\frac{\nu}{2}\right)} \int \lambda^{\frac{\nu+n}{2}-1} \exp\left(-\frac{1}{2}(1+\mathbf{x}'\mathbf{G}^{-1}\mathbf{x})\lambda\right) \mathrm{d}\lambda \tag{17.145}$$

$$= \frac{|\mathbf{G}|^{-\frac{1}{2}}}{2^{\frac{n+\nu}{2}}\pi^{\frac{n}{2}}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(\frac{1}{2}(1+\mathbf{x}'\mathbf{G}^{-1}\mathbf{x})\right)^{-\frac{\nu+n}{2}} \tag{17.146}$$

$$= \frac{|\mathbf{G}|^{-\frac{1}{2}}}{\pi^{\frac{n}{2}}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(1+\mathbf{x}'\mathbf{G}^{-1}\mathbf{x}\right)^{-\frac{\nu+n}{2}} \tag{17.147}$$

Intuitively, this can be understood as taking a single normal sample only revealing one rank of the latent precision matrix, therefore the additional information contained in a Wishart sample is simply lost in that case.

The difference between the two constructions becomes clearer when the latent $\mathbf{\Lambda}$ is shared between several normal samples. If $\boldsymbol{X}$ is a random matrix whose columns $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$ are independent random variables with distribution $\mathcal{N}(\mathbf{0}, \mathbf{\Lambda}^{-1})$, then if $\mathbf{\Lambda}$ is Wishart-distributed as above, $\boldsymbol{X}$ has density

$$p_X(\mathbf{X}) = \int_{\mathbf{\Lambda}>0} \frac{|\mathbf{\Lambda}|^{\frac{p}{2}}}{(2\pi)^{\frac{np}{2}}} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{X}'\mathbf{\Lambda}\mathbf{X})} \cdot \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} |\mathbf{\Lambda}|^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\mathbf{\Lambda})}\, \mathrm{d}\mathbf{\Lambda} \tag{17.148}$$

$$= \frac{1}{(2\pi)^{\frac{np}{2}}} \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \int |\mathbf{\Lambda}|^{\frac{\nu+p}{2}-1} \exp\left(-\frac{1}{2}\mathrm{Tr}((\mathbf{G}+\mathbf{X}\mathbf{X}')\mathbf{\Lambda})\right) \mathrm{d}\mathbf{\Lambda} \tag{17.149}$$

$$= \frac{\Gamma_{\overline{n}}\left(\frac{\nu+p}{2}\right)}{(2\pi)^{\frac{np}{2}}\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \frac{\left|\frac{1}{2}\mathbf{G}\right|^{\frac{\nu+n-1}{2}}}{\left|\frac{1}{2}(\mathbf{G}+\mathbf{X}\mathbf{X}')\right|^{\frac{(\nu+p)+(n-1)}{2}}} \tag{17.150}$$

$$= \frac{\Gamma_{\overline{n}}\left(\frac{\nu+p}{2}\right)}{\pi^{\frac{np}{2}}\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \frac{|\mathbf{G}|^{\frac{\nu+n-1}{2}}}{|\mathbf{G}+\mathbf{X}\mathbf{X}'|^{\frac{\nu+p+n-1}{2}}} \tag{17.151}$$

$$= \frac{|\mathbf{G}|^{-\frac{p}{2}}}{\pi^{\frac{np}{2}}} \frac{\Gamma_{\overline{n}}\left(\frac{\nu+p}{2}\right)}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} \left|I_n + \mathbf{G}^{-1}\mathbf{X}\mathbf{X}'\right|^{-\frac{\nu+p+n-1}{2}} \tag{17.152}$$

which corresponds to matrix t distribution, a further generalization of the multivariate t.

In contrast, if $\mathbf{\Lambda} = \lambda\mathbf{G}^{-1}, \lambda \sim \chi^2(\nu)$ again, $\boldsymbol{X}$ has density

$$p_X(\mathbf{X}) = \int_{\lambda>0} \frac{|\mathbf{\Lambda}|^{\frac{p}{2}}}{(2\pi)^{\frac{np}{2}}} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}(\mathbf{X}'\mathbf{\Lambda}\mathbf{X})} \cdot \frac{1}{2^{\frac{\nu}{2}}\Gamma\left(\frac{\nu}{2}\right)} \lambda^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\lambda}\, \mathrm{d}\lambda \tag{17.153}$$

$$= \frac{|\mathbf{G}|^{-\frac{p}{2}}}{2^{\frac{np+\nu}{2}}\pi^{\frac{np}{2}}\Gamma\left(\frac{\nu}{2}\right)} \int \lambda^{\frac{\nu+np}{2}-1} \exp\left(-\frac{1}{2}(1+\mathrm{Tr}(\mathbf{X}'\mathbf{G}^{-1}\mathbf{X}))\lambda\right) \mathrm{d}\lambda \tag{17.154}$$

$$= \frac{|\mathbf{G}|^{-\frac{p}{2}}}{2^{\frac{np+\nu}{2}}\pi^{\frac{np}{2}}} \frac{\Gamma\left(\frac{\nu+np}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(\frac{1}{2}(1+\mathrm{Tr}(\mathbf{X}'\mathbf{G}^{-1}\mathbf{X}))\right)^{-\frac{\nu+np}{2}} \tag{17.155}$$

$$= \frac{|\mathbf{G}|^{-\frac{p}{2}}}{\pi^{\frac{np}{2}}} \frac{\Gamma\left(\frac{\nu+np}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)} \left(1+\mathrm{Tr}(\mathbf{X}'\mathbf{G}^{-1}\mathbf{X})\right)^{-\frac{\nu+np}{2}} \tag{17.156}$$

which is this time still a multivariate t distribution, when $\boldsymbol{X}$ is seen as a np-dimensional vector, and with a np-dimensional, block-diagonal scale matrix, made up of p identical blocks $\frac{1}{\nu}\mathbf{G}$. Actually, this result can be derived without the calculation above, and instead directly from the vector case, by considering $\boldsymbol{X}|\lambda$ as a np-dimensional normally distributed vector in the first place.

17.7 Improper Wishart-Normal mixture

When incomplete information on a random matrix $\mathbf{\Lambda}$ is available, the partial knowledge available can be expressed by the Wishart-type kernel

$$k_{\mathbf{\Lambda}}(\mathbf{\Lambda}) = |\mathbf{\Lambda}|^{\frac{\nu}{2}-1} e^{-\frac{1}{2}\mathrm{Tr}(\mathbf{G}\mathbf{\Lambda})} \tag{17.157}$$

where $\mathbf{G}$ is a singular PSD matrix.

Due to the singularity, this kernel is not normalizable, and the subsequent mixture with a normal distribution is undefined. However, it is possible to give sense to such a mixture by treating the kernel above as the limit of a Wishart with shape matrix $\mathbf{G} + \varepsilon I_n$ when $\varepsilon \to 0$. That is, intuitively, the missing eigenvalues in $\mathbf{G}$ are assumed to be "something small, but equal across all unobserved eigenvalues"

In that case, a renormalization exercise gives that X inherits an improper distribution

$$k_X(\mathbf{x}) = \left(\mathbf{x}'\mathbf{G}^{\perp}\mathbf{x}\right)^{-\frac{\nu+n}{2}} \tag{17.158}$$

where $\mathbf{G}^{\perp} = I_n - \mathbf{G}\mathbf{G}^{+}$ is the orthogonal projection on $\mathrm{Ker}\,\mathbf{G}$.

Note that this kernel is improper in two different ways:

- Along the singular directions of $\mathbf{G}^{\perp}$, k is flat.
- Along the regular directions of $\mathbf{G}^{\perp}$, k has a polynomial tail that is integrable, but an unintegrable singularity at 0.

Thus this kernel is flat in some directions and sharply peaked at 0 in others, defining a subspace $\mathbb{R}^n$ and a relative geometry in this subspace.

Matrix variate case If XX' is regular (which is almost everywhere the case where X has shape (n, p) with $n < p$), the matrix-t distribution defined by the Wishart-Normal mixture can still be derived without renormalization when an improper Wishart is used as the mixing distribution, with resulting kernel:

$$k(\mathbf{X}) = |\mathbf{G} + \mathbf{X}\mathbf{X}'|^{-\frac{\nu+n+p-1}{2}} \tag{17.159}$$

which is defined almost everywhere even with a singular shape matrix $\mathbf{G}$. Since the normalization constant is known as a function of $|\mathbf{G}|$ in the regular case, it can be seen that the matrix-t is however improper in the singular case.

As a special case, the improper matrix-t stemming from taking the least informative prior thinkable would be obtained for $\mathbf{G} = 0$ and 0 priors degrees of freedom, and would be simply:

$$k(\mathbf{X}) = |\mathbf{X}\mathbf{X}'|^{-\frac{n+p-1}{2}} \tag{17.160}$$

18 Factorisation of PSD matrices and decorrelation

Let

$$\mathbf{G} = \begin{bmatrix} \mathbf{G}_{11} & \mathbf{G}_{12} \\ \mathbf{G}_{12}' & \mathbf{G}_{22} \end{bmatrix} \tag{18.1}$$

be a symmetric positive semi-definite matrix (with blocks of size n_1, n_2). The PSD property can be understood to imply that $\mathbf{G}_{12}$ is "bounded" by $\mathbf{G}_{11}$ and $\mathbf{G}_{22}$. Indeed, for any $\mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}_{n_2}$:

$$|\mathbf{u}'\mathbf{G}_{12}\mathbf{v}| \leqslant \frac{1}{2}(\mathbf{u}'\mathbf{G}_{11}\mathbf{u} + \mathbf{v}'\mathbf{G}_{22}\mathbf{v}) \tag{18.2}$$

For instance, $\mathbf{G}_{11} = 0, \mathbf{G}_{22} = 0$ would imply $\mathbf{G}_{12} = 0$ too. Among other, PSD implies that:

$$\mathbf{G}_{22}\mathbf{G}_{22}^{+}\mathbf{G}_{12}' = \mathbf{G}_{12}' \tag{18.3}$$

In other words, the linear system $\mathbf{G}_{22}\mathbf{X} = \mathbf{G}_{12}'$ always has a solution in $\mathbf{X}$, even when $\mathbf{G}_{22}$ is singular.

As a consequence, it always possible to factor such a PSD matrix as:

$$\mathbf{G} = \begin{bmatrix} I_{n_1} & \mathbf{G}_{12}\mathbf{G}_{22}^{+} \\ 0 & I_{n_2} \end{bmatrix} \begin{bmatrix} \mathbf{G}_{11} - \mathbf{G}_{12}\mathbf{G}_{22}^{+}\mathbf{G}_{12}' & 0 \\ 0 & \mathbf{G}_{22} \end{bmatrix} \begin{bmatrix} I_{n_1} & 0 \\ \mathbf{G}_{22}^{+}\mathbf{G}_{12}' & I_{n_2} \end{bmatrix} \tag{18.4}$$

This can be applied to covariance matrices, if x_1, x_2 are random vector with covariance

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_{11} & \mathbf{\Sigma}_{12} \\ \mathbf{\Sigma}_{12}' & \mathbf{\Sigma}_{22} \end{bmatrix}, \tag{18.5}$$

then

$$\begin{bmatrix} x_1 - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}x_2 \\ x_2 \end{bmatrix} = \begin{bmatrix} I_{n_1} & -\mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+} \\ 0 & I_{n_2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{18.6}$$

has covariance

$$\begin{bmatrix} \mathbf{\Sigma}_{11} - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}\mathbf{\Sigma}_{12}' & 0 \\ 0 & \mathbf{\Sigma}_{22} \end{bmatrix} \tag{18.7}$$

which implies that x_1 can be decomposed as

$$x_1 = \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}x_2 + (x_1 - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}x_2) \tag{18.8}$$

where $\mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}x_2$ is a linear transformation of x_2 and the residual $x_1 - \mathbf{\Sigma}_{12}\mathbf{\Sigma}_{22}^{+}x_2$ is *decorrelated* (but in general not independent) from x_2.

Note that this requires neither any condition on the distribution of x, nor that $\mathbf{\Sigma}$ be invertible.

18.1 Application to conditional distributions

Let $\mathbf{H}$ be a block PSD matrix as above with block sizes n_1 and n_2. By using the factorization introduced above, we have, for $\mathbf{x}' = (\mathbf{x}_1'\mathbf{x}_2')$ with $x_1 \in \mathbb{R}^{n_1}$ and $x_2 \in \mathbb{R}^{n_2}$,

$$\mathbf{x}'\mathbf{H}\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}' \begin{bmatrix} I_{n_1} & \mathbf{H}_{12}\mathbf{H}_{22}^{+} \\ 0 & I_{n_2} \end{bmatrix} \begin{bmatrix} \mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}' & 0 \\ 0 & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} I_{n_1} & 0 \\ \mathbf{H}_{22}^{+}\mathbf{H}_{12}' & I_{n_2} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \tag{18.9}$$

$$= \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 + \mathbf{H}_{22}^{+}\mathbf{H}_{12}'\mathbf{x}_1 \end{bmatrix}' \begin{bmatrix} \mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}' & 0 \\ 0 & \mathbf{H}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 + \mathbf{H}_{22}^{+}\mathbf{H}_{12}'\mathbf{x}_1 \end{bmatrix} \tag{18.10}$$

$$= \mathbf{x}_1'(\mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}')\mathbf{x}_1 + (\mathbf{x}_2 - (-\mathbf{H}_{22}^{+}\mathbf{H}_{12}')\mathbf{x}_1)'\mathbf{H}_{22}(\mathbf{x}_2 - (-\mathbf{H}_{22}^{+}\mathbf{H}_{12}')\mathbf{x}_1) \tag{18.11}$$

$$= \mathbf{x}_1'\mathbf{M}_{11}\mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1)'\mathbf{H}_{22}(\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1) \tag{18.12}$$

where we abbreviate

$$\mathbf{M}_{11} = \mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}' \tag{18.13}$$

$$\mathbf{W}_{21} = -\mathbf{H}_{22}^{+}\mathbf{H}_{12}'\,. \tag{18.14}$$

Note that if $\mathbf{H}$ is invertible with $\mathbf{H} = \mathbf{G}^{-1}$, the matrix in the first term is simply $\mathbf{M}_{11} = \mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}' = \mathbf{G}_{11}^{-1}$.

Such quadratic forms appear frequently in multivariate distributions, with $\mathbf{H}$ usually being some sort of proto-precision matrix (i.e. the precision matrix of the distribution up to some factor), $-\mathbf{H}_{22}^{+}\mathbf{H}_{12}'$ being a weight matrix giving the conditional mean of $\mathbf{x}_2$ given $\mathbf{x}_1$, $\mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}'$ being the marginal proto-precision of $\mathbf{x}_1$ and $\mathbf{H}_{22}$ the residual proto-precision of $\mathbf{x}_2$ given $\mathbf{x}_1$.

When $\mathbf{H}$ is singular, the distribution involving the form is usually improper, but note that that $\mathbf{H}_{22}$ or $\mathbf{H}_{11} - \mathbf{H}_{12}\mathbf{H}_{22}^{+}\mathbf{H}_{12}'$ may be invertible; which allows an improper distribution to yield a correctly normalized one when "marginalized" or "conditioned" over. For an intuition, consider a rank one matrix $\mathbf{H} = \mathbf{h}\mathbf{h}'$; it is highly singular while all the diagonal coefficients h_i^2 are typically positive. More generally, a rank k matrix will typically have regular submatrices of side length up to k, so $\mathbf{H}$ having rank k typically means that conditionals of no more than k unobserved dimensions are proper.

As an important example, consider $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ having the improper t-like distribution

$$k_X(\mathbf{x}) = (\mathbf{x}'\mathbf{H}\mathbf{x})^{-\frac{\nu+n}{2}} \tag{18.15}$$

that we encountered before.

With the factorization of $\mathbf{H}$ above, k can in turn be factored as

$$k_{\mathbf{X}}(\mathbf{x}) = \left[\mathbf{x}_1'\mathbf{M}_{11}\mathbf{x}_1 + (\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1)'\mathbf{H}_{22}(\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1)\right]^{-\frac{\nu+n}{2}} \tag{18.16}$$

$$= \left[\mathbf{x}_1'\mathbf{M}_{11}\mathbf{x}_1\right]^{-\frac{\nu+n}{2}} \left[1 + (\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1)' \frac{\mathbf{H}_{22}}{\mathbf{x}_1'\mathbf{M}_{11}\mathbf{x}_1} (\mathbf{x}_2 - \mathbf{W}_{21}\mathbf{x}_1)\right]^{-\frac{\nu+n}{2}} \tag{18.17}$$

Therefore, provided that

$$\mathbf{H}_{22} > 0 \tag{18.18}$$

$$x_1'\mathbf{M}_{11}x_1 > 0\,, \tag{18.19}$$

k can be normalized over $\mathbf{x}_2$ and $x_2|x_1$ has the *proper* t-distribution

$$p_{x_2|x_1}(\mathbf{x}_2|x_1) = \left|\frac{\mathbf{H}_{22}}{x_1'\mathbf{M}_{11}x_1}\right|^{\frac{1}{2}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\pi^{\frac{n_2}{2}}\Gamma\left(\frac{\nu+n_1}{2}\right)} \times \left[1 + (\mathbf{x}_2 - \mathbf{W}_{21}x_1)' \frac{\mathbf{H}_{22}}{x_1'\mathbf{M}_{11}x_1} (\mathbf{x}_2 - \mathbf{W}_{21}x_1)\right]^{-\frac{(\nu+n_1)+n_2}{2}} \tag{18.20}$$

with moments:

$$\mathbb{E}[x_2|x_1] = \mathbf{W}_{21}x_1 = -\mathbf{H}_{22}^{-1}\mathbf{H}_{12}'x_1 \tag{18.21}$$

$$\mathbb{C}\mathrm{ov}[x_2|x_1] = \frac{x_1'\mathbf{M}_{11}x_1}{\nu + n_1 - 2}\mathbf{H}_{22}^{-1} \tag{18.22}$$

Note that both the mean and variance in that case are unchanged under multiplication of $\mathbf{H}$ by a positive scalar, consistent with the idea that $\mathbf{H}$ only defines a subspace but does not encode absolute variance information.

As stated before, the condition $\mathbf{H}_{22} > 0$ is typically a matter of rank of $\mathbf{H}$ and dimension of x_2; it is often satisfied when x_2 is low-dimensional and $\mathbf{H}$ high rank but eventually fails as the rank of $\mathbf{H}$ falls below the dimension of x_2. Intuitively, conditioning on one more dimension makes up from one rank deficiency of $\mathbf{H}$.

Finally, notice a classical paradox here: in the case where $\mathbf{H} = \mathbf{G}^{\perp}$ that we encountered above, below n, as one gets more data, the rank of $\mathbf{G}$ grows, the rank of $\mathbf{H}$ gets *lower* and the distribution is *less* well behaved with more conditionals being improper, until $\mathbf{G}$ gets full rank where everything becomes proper again. This behavior is well known to most if the additional ranks of $\mathbf{G}$ are thought of as additional features in a linear model: as the number of features approaches the number of samples, the unregularized estimators become badly behaved.

The same factorization also gives the conditional distribution for a regular multivariate t-distribution: if x has density

$$p_x(\mathbf{x}) = |\mathbf{H}|^{\frac{1}{2}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\pi^{\frac{n}{2}}\Gamma\left(\frac{\nu}{2}\right)}(1+\mathbf{x}'\mathbf{H}\mathbf{x})^{-\frac{\nu+n}{2}}, \tag{18.23}$$

the same factorization of $\mathbf{H}$ gives the conditional distribution of $x_2|x_1$ as

$$\begin{aligned} p_{x_2|x_1}(\mathbf{x}_2|x_1) = &\left|\frac{\mathbf{H}_{22}}{1+x_1'\mathbf{M}_{11}x_1}\right|^{\frac{1}{2}} \frac{\Gamma\left(\frac{\nu+n}{2}\right)}{\pi^{\frac{n_2}{2}}\Gamma\left(\frac{\nu+n_1}{2}\right)} \\ &\left[1+(\mathbf{x}_2-\mathbf{W}_{21}x_1)'\frac{\mathbf{H}_{22}}{1+x_1'\mathbf{M}_{11}x_1}(\mathbf{x}_2-\mathbf{W}_{21}x_1)\right]^{-\frac{(\nu+n_1)+n_2}{2}} \end{aligned} \tag{18.24}$$

with moments:

$$\mathbb{E}[x_2|x_1] = \mathbf{W}_{21}x_1 = -\mathbf{H}_{22}^{-1}\mathbf{H}_{12}'x_1 \tag{18.25}$$

$$\mathbb{C}\mathrm{ov}[x_2|x_1] = \frac{1+x_1'\mathbf{M}_{11}x_1}{\nu+n_1-2}\mathbf{H}_{22}^{-1} \tag{18.26}$$

which is naturally similar to the improper case above, except that the condition $x_1'\mathbf{M}_{11}x_1 > 0$ is no longer necessary for the conditional to be proper and that $\mathbf{H}$ this time does encode scale information.

Finally, in the Gaussian case, with $\Lambda = \nu\mathbf{H}$ the precision matrix of x, we have simply that x having the density

$$p_x(\mathbf{x}) \propto \exp\left(-\frac{\nu}{2}\mathbf{x}'\mathbf{H}\mathbf{x}\right) \tag{18.27}$$

$$\propto \exp\left(-\frac{\nu}{2}\mathbf{x}_1'\mathbf{M}_{11}\mathbf{x}_1\right)\exp\left(-\frac{\nu}{2}(\mathbf{x}_2-\mathbf{W}_{21}\mathbf{x}_1)'\mathbf{H}_{22}(\mathbf{x}_2-\mathbf{W}_{21}\mathbf{x}_1)\right) \tag{18.28}$$

entails that $x_2|x_1$ is Gaussian with moments:

$$\mathbb{E}[x_2|x_1] = \mathbf{W}_{21}x_1 = -\mathbf{H}_{22}^{-1}\mathbf{H}_{12}'x_1 \tag{18.29}$$

$$\mathbb{C}\mathrm{ov}[x_2|x_1] = \frac{1}{\nu}\mathbf{H}_{22}^{-1} \tag{18.30}$$

Once again, the conditional mean is unchanged, but here the conditional covariance is independent from x_1: in a normal distribution, the covariance structure is assumed to be already known with infinite accuracy and there is no latent scale to infer from

x_1. This constitutes in a way the polar opposite of the improper t case: all the scale information is already in $\mathbf{H}$ and none comes from x_1.

The results above, all consequences of same factorization of $\mathbf{H}$, can be summarized in the table:

	Improper t-like	Multivariate-t	Gaussian
Joint kernel	$(\mathbf{x}'\mathbf{H}\mathbf{x})^{-\frac{\nu+n}{2}}$	$(1+\mathbf{x}'\mathbf{H}\mathbf{x})^{-\frac{\nu+n}{2}}$	$\exp\left(-\frac{\nu}{2}\mathbf{x}'\mathbf{H}\mathbf{x}\right)$
$\mathbb{E}[x_2\|x_1]$		always $-\mathbf{H}_{22}^{-1}\mathbf{H}_{12}'x_1$	
$\mathbb{C}\mathrm{ov}[x_2\|x_1]$	$\frac{x_1'\mathbf{M}_{11}x_1}{\nu+n_1-2}\mathbf{H}_{22}^{-1}$	$\frac{1+x_1'\mathbf{M}_{11}x_1}{\nu+n_1-2}\mathbf{H}_{22}^{-1}$	$\frac{1}{\nu}\mathbf{H}_{22}^{-1}$

18.2 Factorization of renormalized inverse squares

Commonly, the formulas above will be used with $\mathbf{H}$ obtained by inverting an empirical covariance matrix $\propto \mathbf{A}'\mathbf{A}$ of some observed matrix $\mathbf{A}$. If $\mathbf{A}'\mathbf{A}$ has full rank, this will typically be done by direct numerical calculation, but if $\mathbf{A}$ is a wide matrix and $\mathbf{A}'\mathbf{A}$ is singular, we will need to use the formula above with the renormalized

$$\mathbf{H} = (\mathbf{A}'\mathbf{A})^{\perp} \tag{18.31}$$
$$= I_p - (\mathbf{A}'\mathbf{A})^{+}(\mathbf{A}'\mathbf{A}) \tag{18.32}$$
$$= I_p - \mathbf{A}^{+}\mathbf{A} \tag{18.33}$$
$$= I_p - \mathbf{A}'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A} \tag{18.34}$$

(with the last definition valid if $\mathbf{A}\mathbf{A}'$ has full rank, as is typically the case for us with a wide $\mathbf{A}$)

Now, if $\mathbf{A}$ is made of two blocks:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \tag{18.35}$$

with p_1 and p_2 columns respectively, then $\mathbf{H}$ in turn would have the block structure

$$\mathbf{H} = \begin{bmatrix} I_{p_1} - \mathbf{A}_1'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1 & \mathbf{A}_1'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_2 \\ \mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1 & I_{p_2} - \mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_2 \end{bmatrix} \tag{18.36}$$

The inverses of the diagonal blocks have the expression

$$\mathbf{H}_{11}^{-1} = I_{p_1} + \mathbf{A}_1'(\mathbf{A}\mathbf{A}' - \mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_1 \tag{18.37}$$
$$= I_{p_1} + \mathbf{A}_1'(\mathbf{A}_2\mathbf{A}_2')^{-1}\mathbf{A}_1 \tag{18.38}$$
$$\tag{18.39}$$

and symmetrically

$$\mathbf{H}_{22}^{-1} = I_{p_2} + \mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_2 \tag{18.40}$$
$$\tag{18.41}$$

provided that $\mathbf{A}_2\mathbf{A}_2'$ or $\mathbf{A}_1\mathbf{A}_1'$ respectively are regular.

The weight matrix defined above as $\mathbf{W}_{21} = -\mathbf{H}_{22}^{-1}\mathbf{H}_{12}'$ then takes the simple expression

$$
\begin{aligned}
\mathbf{W}_{21} &= -\mathbf{H}_{22}^{-1}\mathbf{H}_{12}' && (18.42)\\
&= -\left[I_{p_2} + \mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_2\right]\left[\mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1\right] && (18.43)\\
&= -\left[\mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1 + \mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_2\mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1\right] && (18.44)\\
&= -\left[\mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1 + \mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}(\mathbf{A}\mathbf{A}' - \mathbf{A}_1\mathbf{A}_1')(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1\right] && (18.45)\\
&= -\left[\mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1 + \mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_1 - \mathbf{A}_2'(\mathbf{A}\mathbf{A}')^{-1}\mathbf{A}_1\right] && (18.46)\\
&= -\mathbf{A}_2'(\mathbf{A}_1\mathbf{A}_1')^{-1}\mathbf{A}_1 && (18.47)\\
&= -\mathbf{A}_2'\mathbf{A}_1^{+\prime} && (18.48)
\end{aligned}
$$

19 The Matrix-t distribution

The matrix t-distribution is a natural extension of t-distribution to matrix variates, as its name suggests. It has been described before by a few authors [GN99], though some the results that we present here especially for the improper case leave-one-out conditionals by integrating low-rank updates, and necessary for our work, have not been encountered by us in the existing literature. Here again, we briefly derive ourselves the properties that we rely on with compact proofs where we can formulate them.

19.1 The standard matrix t distribution

As part of the Wishart-Normal mixture derivation, we found that the density on $\mathbb{R}^{n\times p}$:

$$p_X(\mathbf{X}) = \pi^{-\frac{np}{2}} \frac{\Gamma_{\overline{n}}\left(\frac{\nu+p}{2}\right)}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} |I_n + \mathbf{X}\mathbf{X}'|^{-\frac{\nu+p+n-1}{2}} \tag{19.1}$$

is the correctly normalized density of $\boldsymbol{X}$ when $\boldsymbol{X}$ is a random matrix whose columns $\boldsymbol{x}^1, \ldots, \boldsymbol{x}^p$ are independent random variables with distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$, and $\boldsymbol{\Lambda}$ in turn is Wishart-distributed with density

$$p_{\boldsymbol{\Lambda}}(\boldsymbol{\Lambda}) = \frac{2^{-\frac{n(\nu+n-1)}{2}}}{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)} |\boldsymbol{\Lambda}|^{\frac{\nu}{2}-1} \mathrm{e}^{-\frac{1}{2}\mathrm{Tr}\boldsymbol{\Lambda}} \tag{19.2}$$

We define:

$$\begin{aligned} \tau_{\nu,n,p} &= \int |I_n + \mathbf{X}\mathbf{X}'|^{-\frac{\nu+p+n-1}{2}} \, \mathrm{d}\mathbf{X} && (19.3) \\ &= \pi^{\frac{np}{2}} \frac{\Gamma_{\overline{n}}\left(\frac{\nu}{2}\right)}{\Gamma_{\overline{n}}\left(\frac{\nu+p}{2}\right)} && (19.4) \end{aligned}$$

and write more concisely:

$$p(\mathbf{X}) = \tau_{\nu,n,p}^{-1} \, |I_n + \mathbf{X}\mathbf{X}'|^{-\frac{\nu+p+n-1}{2}} \tag{19.5}$$

Mean and variance. By symmetry, the mean of such a distribution can only be 0.

We also know that the column $\boldsymbol{x}^j$ has conditional covariance:

$$\mathbb{C}\mathrm{ov}\left[\boldsymbol{x}^j|\boldsymbol{\Lambda}\right] = \boldsymbol{\Lambda}^{-1} \tag{19.6}$$

Therefore, by total expectation and using (17.134), for $\nu > 2$,

$$\mathbb{C}\mathrm{ov}\left[\boldsymbol{x}^j\right] = \frac{1}{\nu - 2} I_n \tag{19.7}$$

Note that by duality of the construction of the matrix-t, the rows $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n$ of $\boldsymbol{X}$ can also be seen as independent Gaussian variables given a p-dimensional precision matrix following a plain Wishart distribution with ν degrees of freedom, therefore we also have

$$\mathbb{C}\mathrm{ov}[\boldsymbol{x}_i] = \frac{1}{\nu - 2} I_p \tag{19.8}$$

More generally, one can write

$$\mathbb{C}\mathrm{ov}[\boldsymbol{X}] = \frac{1}{\nu - 2} I_n \otimes I_p \tag{19.9}$$

by which we mean

$$\mathbb{C}\mathrm{ov}\left[X_{ij}, X_{kl}\right] = \begin{cases} \frac{1}{\nu-2} & \text{if } i = j \text{ and } k = l \\ 0 & \text{otherwise} \end{cases} \tag{19.10}$$

Notice how this generalizes the one-dimensional case (17.69).

19.2 Translated and scaled matrix-t

Let $\mathbf{U} \in GL_n(\mathbb{R})$, $\mathbf{V} \in GL_p(\mathbb{R})$ and $\mathbf{M} \in \mathbb{R}^{n \times p}$. If $\mathbf{Z}$ follows the plain matrix-t distribution above, we can define

$$\boldsymbol{X} = \mathbf{U}\mathbf{Z}\mathbf{V}' + \mathbf{M} \tag{19.11}$$

we therefore have

$$\boldsymbol{Z} = \mathbf{U}^{-1}(\boldsymbol{X} - \mathbf{M})\mathbf{V}^{-1\prime} \tag{19.12}$$

and the change of variable

$$\mathrm{d}\mathbf{Z} = |\mathbf{U}|^{-p}|\mathbf{V}|^{-n}\,\mathrm{d}\mathbf{X} \tag{19.13}$$

From this we get the density of X as:

$$p_X(\mathbf{X}) = \frac{|\mathbf{U}|^{-p}|\mathbf{V}|^{-n}}{\tau_{\nu,n,p}} \begin{vmatrix} I_n & \mathbf{U}^{-1}(\mathbf{X}-\mathbf{M})\mathbf{V}^{-1\prime} \\ -\mathbf{V}^{-1}(\mathbf{X}-\mathbf{M})'\mathbf{U}^{-1\prime} & I_p \end{vmatrix}^{-\frac{\nu+p+n-1}{2}} \tag{19.14}$$

$$= \frac{|\mathbf{U}|^{-p}|\mathbf{V}|^{-n}}{\tau_{\nu,n,p}} \left[\begin{vmatrix} \mathbf{U}^{-1} & \\ & \mathbf{V}^{-1} \end{vmatrix}^2 \begin{vmatrix} \mathbf{U}\mathbf{U}' & \mathbf{X}-\mathbf{M} \\ -(\mathbf{X}-\mathbf{M})' & \mathbf{V}\mathbf{V}' \end{vmatrix} \right]^{-\frac{\nu+p+n-1}{2}} \tag{19.15}$$

$$= \frac{|\mathbf{U}|^{\nu+n-1}|\mathbf{V}|^{\nu+p-1}}{\tau_{\nu,n,p}} \begin{vmatrix} \mathbf{U}\mathbf{U}' & \mathbf{X}-\mathbf{M} \\ -(\mathbf{X}-\mathbf{M})' & \mathbf{V}\mathbf{V}' \end{vmatrix}^{-\frac{\nu+p+n-1}{2}} \tag{19.16}$$

or, if we set

$$\mathbf{B} = \mathbf{U}\mathbf{U}' \tag{19.17}$$

$$\mathbf{T} = \mathbf{V}\mathbf{V}', \tag{19.18}$$

$$p_X(\mathbf{X}) = \frac{|\mathbf{B}|^{\frac{\nu+n-1}{2}}|\mathbf{T}|^{\frac{\nu+p-1}{2}}}{\tau_{\nu,n,p}} \begin{vmatrix} \mathbf{B} & \mathbf{X}-\mathbf{M} \\ -(\mathbf{X}-\mathbf{M})' & \mathbf{T} \end{vmatrix}^{-\frac{\nu+p+n-1}{2}} \tag{19.19}$$

From the definition (19.11) and (19.9), we naturally have

$$\mathbb{E}[X] = \mathbf{M} \tag{19.20}$$

$$\mathbb{C}\mathrm{ov}[X] = \frac{1}{\nu-2}\mathbf{B}\otimes\mathbf{T} \tag{19.21}$$

where the tensor notation once again stands for

$$\mathbb{C}\mathrm{ov}\left[X_{ij}, X_{kl}\right] = \frac{1}{\nu-2} B_{ik}T_{jl} \tag{19.22}$$

19.3 Conditionals

Now, we consider the case where $\mathbf{X}$ is divided into four blocks of variables as:

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 & \mathbf{X}_2 \\ \mathbf{X}_3 & \mathbf{X}_4 \end{pmatrix} \tag{19.23}$$

with n_1, n_2 rows and p_1, p_2 columns per block. We will establish conditional distributions for increasingly general forms of the matrix-t. The general formulas being rather convoluted, having dedicated sections and proofs for the simple cases first is convenient.

For the standard matrix-t, the generating determinant can be factored as

$$\begin{aligned}
&\begin{vmatrix} I_{n_1} & & \mathbf{X}_1 & \mathbf{X}_2 \\ & I_{n_2} & \mathbf{X}_3 & \mathbf{X}_4 \\ -\mathbf{X}_1' & -\mathbf{X}_3' & I_{p_1} & \\ -\mathbf{X}_2' & -\mathbf{X}_4' & & I_{p_2} \end{vmatrix} \\
&\quad = \begin{vmatrix} I_{n_2} & \mathbf{X}_3 \\ -\mathbf{X}_3' & I_{p_1} \end{vmatrix} \times \\
&\qquad \begin{vmatrix} I_{n_1} + \mathbf{X}_1(I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_1' & \mathbf{X}_2 - \mathbf{X}_1(I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{X}_4 \\ -(\mathbf{X}_2 - \ldots)' & I_{p_2} + \mathbf{X}_4'(I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4 \end{vmatrix}
\end{aligned} \tag{19.24}$$

by noting that the central block matrix has inverse

$$\begin{bmatrix} I_{n_2} & \mathbf{X}_3 \\ -\mathbf{X}_3' & I_{p_1} \end{bmatrix}^{-1} = \begin{bmatrix} (I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1} & -(I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_3 \\ \mathbf{X}_3'(I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1} & (I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1} \end{bmatrix} \tag{19.25}$$

$$= \begin{bmatrix} (I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1} & -\mathbf{X}_3(I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1} \\ (I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_3' & (I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1} \end{bmatrix} \tag{19.26}$$

So $\mathbf{X}_2$ is again matrix-t distributed with $\nu + n_2 + p_1$ degrees of freedom, mean

$$\mathbb{E}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] = \mathbf{X}_1(I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{X}_4 \tag{19.27}$$

$$= \mathbf{X}_1\mathbf{X}_3'(I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4 \tag{19.28}$$

and covariance

$$\begin{aligned}
&\mathbb{C}\mathrm{ov}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] \\
&\quad = \frac{(I_{n_1} + \mathbf{X}_1(I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_1') \otimes (I_{p_2} + \mathbf{X}_4'(I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4)}{\nu + n_2 + p_1 - 2}
\end{aligned} \tag{19.29}$$

Note that if either $\mathbf{X}_3'\mathbf{X}_3 \gg I_{p_1}$ or $\mathbf{X}_3\mathbf{X}_3' \gg I_{n_2}$ (i.e, $\mathbf{X}_3$ is either a very tall or very wide rectangular matrix, but typically *not* when it is a large squarish matrix), this mean approaches the classical unregularized formula:

$$\mathbb{E}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] \to \mathbf{X}_1\mathbf{X}_3^{+}\mathbf{X}_4 \tag{19.30}$$

Such a distribution makes sense only for a dimensionless variable; more realistically one would introduce at least a scale $\sigma > 0$ such that $\frac{1}{\sigma}\mathbf{X}$ is standard-MT distributed, in which case by substituting above one would get:

$$\mathbb{E}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] = \mathbf{X}_1(\sigma^2 I_{p_1} + \mathbf{X}_3'\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{X}_4 \tag{19.31}$$

$$= \mathbf{X}_1\mathbf{X}_3'(\sigma^2 I_{n_2} + \mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4 \tag{19.32}$$

More generally, with matrices $\mathbf{B}_{12}, \mathbf{B}_{34}, \mathbf{T}_{13}, \mathbf{T}_{24}$, if $\mathbf{X}$ has density

$$p_{\mathbf{X}}(\mathbf{X}) \propto \begin{vmatrix} \mathbf{B}_{12} & & \mathbf{X}_1 & \mathbf{X}_2 \\ & \mathbf{B}_{34} & \mathbf{X}_3 & \mathbf{X}_4 \\ -\mathbf{X}_1' & -\mathbf{X}_3' & \mathbf{T}_{13} & \\ -\mathbf{X}_2' & -\mathbf{X}_4' & & \mathbf{T}_{24} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.33}$$

the pivot matrix can be inverted as

$$\begin{aligned} &\begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} (\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{12}^{-1}\mathbf{X}_3')^{-1} & -(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_3\mathbf{T}_{13}^{-1} \\ \mathbf{T}_{13}^{-1}\mathbf{X}_3'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{12}^{-1}\mathbf{X}_3')^{-1} & (\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1} \end{bmatrix} && (19.34) \\ &= \begin{bmatrix} (\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{12}^{-1}\mathbf{X}_3')^{-1} & -\mathbf{B}_{34}^{-1}\mathbf{X}_3(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1} \\ (\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{B}_{34}^{-1} & (\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1} \end{bmatrix} && (19.35) \end{aligned}$$

Then, the joint density can be factored similarly, yielding:

$$p(\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4) \propto$$

$$\begin{vmatrix} \mathbf{B}_{12} + \mathbf{X}_1(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_1' & \mathbf{X}_2 - \mathbf{X}_1(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_4 \\ -(\mathbf{X}_2 - \ldots)' & \mathbf{T}_{24} + \mathbf{X}_4'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_4 \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.36}$$

(Alternatively, this could be obtained by applying linear transformations to each block).

From this density, we identify the moments:

$$\begin{aligned} \mathbb{E}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] &= \mathbf{X}_1(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_4 && (19.37) \\ &= \mathbf{X}_1\mathbf{T}_{13}^{-1}\mathbf{X}_3'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_4 && (19.38) \end{aligned}$$

and

$$\begin{aligned} &\mathbb{C}\mathrm{ov}[\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4] = \frac{1}{\nu + n_2 + p_1 - 2} \times && (19.39) \\ &(\mathbf{B}_{12} + \mathbf{X}_1(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_1') \otimes (\mathbf{T}_{24} + \mathbf{X}_4'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_4) \end{aligned}$$

Note that the two factors in the tensor product can be expressed as functions of the same matrix inverse through the Woodbury identity, for instance

$$(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1} = \mathbf{T}_{13}^{-1} - \mathbf{T}_{13}^{-1}\mathbf{X}_3'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_3\mathbf{T}_{13}^{-1} \tag{19.40}$$

so

$$\begin{aligned}&\mathbf{B}_{12} + \mathbf{X}_1(\mathbf{T}_{13} + \mathbf{X}_3'\mathbf{B}_{34}^{-1}\mathbf{X}_3)^{-1}\mathbf{X}_1' = \\ &\quad \mathbf{B}_{12} + \mathbf{X}_1\mathbf{T}_{13}^{-1}\mathbf{X}_1' - \mathbf{X}_1\mathbf{T}_{13}^{-1}\mathbf{X}_3'(\mathbf{B}_{34} + \mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_3')^{-1}\mathbf{X}_3\mathbf{T}_{13}^{-1}\mathbf{X}_1' \end{aligned} \tag{19.41}$$

Finally, we give the expressions for the full matrix-t. If $\mathbf{X}$ has density

$$p_{\mathbf{X}}(\mathbf{X}) \propto \begin{vmatrix} \mathbf{B} & \mathbf{X} \\ -\mathbf{X}' & \mathbf{T} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.42}$$

$$\propto \begin{vmatrix} \mathbf{B}_{12} & \mathbf{B}_c & \mathbf{X}_1 & \mathbf{X}_2 \\ \mathbf{B}_c & \mathbf{B}_{34} & \mathbf{X}_3 & \mathbf{X}_4 \\ -\mathbf{X}_1' & -\mathbf{X}_3' & \mathbf{T}_{13} & \mathbf{T}_c \\ -\mathbf{X}_2' & -\mathbf{X}_4' & \mathbf{T}_c & \mathbf{T}_{24} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.43}$$

then, using the same pivot inverse, we obtain that the $\mathbf{X}_2$ has a conditional density of the form:

$$p(\mathbf{X}_2|\mathbf{X}_1,\mathbf{X}_3,\mathbf{X}_4) \propto \begin{vmatrix} \tilde{\mathbf{B}} & \mathbf{X}_2 - \tilde{\mathbf{M}} \\ -(\mathbf{X}_2 - \tilde{\mathbf{M}})' & \tilde{\mathbf{T}} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.44}$$

with:

$$\tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{B}_c & \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}_4 \\ \mathbf{T}_c \end{bmatrix} \tag{19.45}$$

$$\tilde{\mathbf{B}} = \mathbf{B}_{12} - \begin{bmatrix} \mathbf{B}_c & \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{B}_c \\ -\mathbf{X}_1' \end{bmatrix} \tag{19.46}$$

$$\tilde{\mathbf{T}} = \mathbf{T}_{24} - \begin{bmatrix} -\mathbf{X}_4' & \mathbf{T}_c \end{bmatrix} \begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}_4 \\ \mathbf{T}_c \end{bmatrix} \tag{19.47}$$

Since we know how to normalize this matrix-t distribution, we can establish the full factorization including normalization constants as

$$p_{\mathbf{X}}(\mathbf{X}) = \frac{|\mathbf{B}|^{\frac{\nu+n-1}{2}}|\mathbf{T}|^{\frac{\nu+p-1}{2}}}{\tau_{\nu,n,p}} \begin{vmatrix} \mathbf{B} & \mathbf{X} \\ -\mathbf{X}' & \mathbf{T} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.48}$$

$$\begin{aligned} &= \frac{|\mathbf{B}|^{\frac{\nu+n-1}{2}}|\mathbf{T}|^{\frac{\nu+p-1}{2}}}{\tau_{\nu,n,p}} \left[\begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} \begin{vmatrix} \tilde{\mathbf{B}} & \mathbf{X}_2 - \tilde{\mathbf{M}} \\ -(\mathbf{X}_2 - \tilde{\mathbf{M}})' & \tilde{\mathbf{T}} \end{vmatrix} \right]^{-\frac{\nu+n+p-1}{2}} \\ &= \frac{\tau_{(\nu+n_2+p_1),n_1,p_2}}{\tau_{\nu,n,p}} \cdot \frac{|\mathbf{B}|^{\frac{\nu+n-1}{2}}}{|\tilde{\mathbf{B}}|^{\frac{\nu+n+p_1-1}{2}}} \cdot \frac{|\mathbf{T}|^{\frac{\nu+p-1}{2}}}{|\tilde{\mathbf{T}}|^{\frac{\nu+n_2+p-1}{2}}} \cdot \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \\ &\times \frac{|\tilde{\mathbf{B}}|^{\frac{\nu+n+p_1-1}{2}}|\tilde{\mathbf{T}}|^{\frac{\nu+n_2+p-1}{2}}}{\tau_{(\nu+n_2+p_1),n_1,p_2}} \begin{vmatrix} \tilde{\mathbf{B}} & \mathbf{X}_2 - \tilde{\mathbf{M}} \\ -(\mathbf{X}_2 - \tilde{\mathbf{M}})' & \tilde{\mathbf{T}} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \end{aligned} \tag{19.49}$$

From this, we obtain the distribution of the marginal distribution of $(\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4)$ with $\mathbf{X}_2$ integrated out:

$$p(\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4) = \frac{\tau_{(\nu+n_2+p_1),n_1,p_2}}{\tau_{\nu,n,p}} \cdot \frac{|\mathbf{B}|^{\frac{\nu+n-1}{2}}}{|\tilde{\mathbf{B}}|^{\frac{\nu+n+p_1-1}{2}}} \cdot \frac{|\mathbf{T}|^{\frac{\nu+p-1}{2}}}{|\tilde{\mathbf{T}}|^{\frac{\nu+n_2+p-1}{2}}} \cdot \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.50}$$

Since $\tilde{\mathbf{B}}$ depends on $\mathbf{X}_1$ and $\mathbf{X}_3$ but not $\mathbf{X}_4$, while $\tilde{\mathbf{T}}$ depends on $\mathbf{X}_3$ and $\mathbf{X}_4$ but not $\mathbf{X}_1$, this marginal contains no factor involving both $\mathbf{X}_1$ and $\mathbf{X}_4$. Therefore, $\mathbf{X}_1$ and $\mathbf{X}_4$ are independent given $\mathbf{X}_3$ (when $\mathbf{X}_2$ is marginalized out) and the marginal above can more simply be written:

$$\begin{aligned} p(\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4) &= p(\mathbf{X}_1|\mathbf{X}_3)p(\mathbf{X}_4|\mathbf{X}_3)p(\mathbf{X}_3) & (19.51)\\ &= \frac{p(\mathbf{X}_1, \mathbf{X}_3)p(\mathbf{X}_4, \mathbf{X}_3)}{p(\mathbf{X}_3)} & (19.52) \end{aligned}$$

The normalization constant of the conditional distribution of $\mathbf{X}_1$ involves the determinants of $\tilde{\mathbf{B}}$ and $\tilde{\mathbf{T}}$. Owing to the particular form of these matrices, these determinants also appear in the factorizations of the sub-matrices:

$$\begin{aligned} \begin{vmatrix} \mathbf{B}_{12} & \mathbf{B}_c & \mathbf{X}_1 \\ \mathbf{B}_c & \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_1' & -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} &= \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} \left| \mathbf{B}_{12} - \begin{bmatrix} \mathbf{B}_c & \mathbf{X}_1 \end{bmatrix} \begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{B}_c \\ -\mathbf{X}_1' \end{bmatrix} \right| \\ &= \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} |\tilde{\mathbf{B}}| & (19.53) \\ \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 & \mathbf{X}_4 \\ -\mathbf{X}_3' & \mathbf{T}_{13} & \mathbf{T}_c \\ -\mathbf{X}_4' & \mathbf{T}_c & \mathbf{T}_{24} \end{vmatrix} &= \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} \left| \mathbf{T}_{24} - \begin{bmatrix} -\mathbf{X}_4' & \mathbf{T}_c \end{bmatrix} \begin{bmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{X}_4 \\ \mathbf{T}_c \end{bmatrix} \right| \\ &= \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} |\tilde{\mathbf{T}}| & (19.54) \end{aligned}$$

In summary, the three determinants needed to express $p(\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4)$ can be obtained as:

$$\begin{vmatrix} \tilde{\mathbf{B}} & \mathbf{X}_2 - \tilde{\mathbf{M}} \\ -(\mathbf{X}_2 - \tilde{\mathbf{M}})' & \tilde{\mathbf{T}} \end{vmatrix} = \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B} & \mathbf{X} \\ -\mathbf{X}' & \mathbf{T} \end{vmatrix} \tag{19.55}$$

$$\left|\tilde{\mathbf{T}}\right| = \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 & \mathbf{X}_4 \\ -\mathbf{X}_3' & \mathbf{T}_{13} & \mathbf{T}_c \\ -\mathbf{X}_4' & \mathbf{T}_c & \mathbf{T}_{24} \end{vmatrix} \tag{19.56}$$

$$\left|\tilde{\mathbf{B}}\right| = \begin{vmatrix} \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B}_{12} & \mathbf{B}_c & \mathbf{X}_1 \\ \mathbf{B}_c & \mathbf{B}_{34} & \mathbf{X}_3 \\ -\mathbf{X}_1' & -\mathbf{X}_3' & \mathbf{T}_{13} \end{vmatrix} \tag{19.57}$$

19.4 Conditionals of the improper standard matrix-t

We already stated that for $n < p$, the matrix-t kernel can still be defined, but improper, for singular B. In this case, a factorization similar to the one above may still hold and the resulting conditional kernel may even become proper. Such an example is the case $B = 0, T = I_p, \nu = 0$.

Then if $\mathbf{X}_3\mathbf{X}_3'$ is invertible, the central block matrix is too, with inverse:

$$\begin{bmatrix} I_{p_1} & -\mathbf{X}_3' \\ \mathbf{X}_3 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} I_{p_1} - \mathbf{X}_3'(\mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_3 & \mathbf{X}_3'(\mathbf{X}_3\mathbf{X}_3')^{-1} \\ -(\mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_3 & (\mathbf{X}_3\mathbf{X}_3')^{-1} \end{bmatrix} \tag{19.58}$$

$$= \begin{bmatrix} I_{p_1} - \mathbf{X}_3^+\mathbf{X}_3 & \mathbf{X}_3^+ \\ -\mathbf{X}_3^{+\prime} & (\mathbf{X}_3\mathbf{X}_3')^{-1} \end{bmatrix} \tag{19.59}$$

$$= \begin{bmatrix} (\mathbf{X}_3'\mathbf{X}_3)^{\perp} & \mathbf{X}_3^+ \\ -\mathbf{X}_3^{+\prime} & (\mathbf{X}_3\mathbf{X}_3')^{-1} \end{bmatrix} \tag{19.60}$$

and the matrix-t kernel factorizes as

$$k(\mathbf{X}_2|\mathbf{X}_1, \mathbf{X}_3, \mathbf{X}_4) \propto \begin{vmatrix} \mathbf{X}_1(\mathbf{X}_3'\mathbf{X}_3)^{\perp}\mathbf{X}_1' & \mathbf{X}_2 - \mathbf{X}_1\mathbf{X}_3^+\mathbf{X}_4 \\ -(\mathbf{X}_2 - \ldots)' & I_{p_2} + \mathbf{X}_4'(\mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4 \end{vmatrix}^{-\frac{n+p-1}{2}} \tag{19.61}$$

which is proper when $\mathbf{X}_1(\mathbf{X}_3'\mathbf{X}_3)^{\perp}\mathbf{X}_1'$ and $I_{p_2} + \mathbf{X}_4'(\mathbf{X}_3\mathbf{X}_3')^{-1}\mathbf{X}_4$ are regular, and improper but still well-defined if at least $\mathbf{X}_1(\mathbf{X}_3'\mathbf{X}_3)^{\perp}\mathbf{X}_1'$ (assuming $n_1 < p_2$) is.

19.5 Leave-one-out conditionals

We now consider a matrix variable $\boldsymbol{X}$ with values in $\mathbb{R}^{n,p}$ split into $2p$ blocks as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^1 & \cdots & \boldsymbol{x}_1^p \\ \boldsymbol{x}_2^1 & \cdots & \boldsymbol{x}_2^p \end{bmatrix} \tag{19.62}$$

where each $\boldsymbol{x}_i^j \in \mathbb{R}^{n_i}$ is a column vector with n_i rows. We abbreviate

$$\boldsymbol{X}_i^{-j} = \boldsymbol{x}_i^1, \ldots, \boldsymbol{x}_i^{j-1}, \boldsymbol{x}_i^{j+1}, \ldots, \boldsymbol{x}_i^p \tag{19.63}$$

We assume $\boldsymbol{X}$ follows either an improper or proper matrix-t:

$$k(\mathbf{X}) = \begin{vmatrix} \mathbf{B} & \mathbf{X} \\ -\mathbf{X}' & I_p \end{vmatrix} \tag{19.64}$$

with $\mathbf{B}$ possibly singular ($\mathbf{B} = I_n$ gives the standard proper matrix-t, $\mathbf{B} = \mathbf{0}$ gives the standard improper matrix-t). We split accordingly:

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{B}_c \\ \mathbf{B}_c & \mathbf{B}_2 \end{bmatrix} \tag{19.65}$$

We are interested in the conditionals $x_1^j|X_1^{-j}, X_2$. From (19.44) and (19.55), we know them to be:

$$p(\mathbf{x}_1^j|\mathbf{X}_1^{-j},\mathbf{X}_2) \propto \begin{vmatrix} \tilde{\mathbf{B}}_j & \mathbf{x}_1^j - \tilde{\mathbf{m}}_j \\ -(\mathbf{x}_1^j - \tilde{\mathbf{m}}_j)' & \tilde{t}_j \end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.66}$$

with:

$$\begin{vmatrix} \tilde{\mathbf{B}}_j & \mathbf{x}_1^j - \tilde{\mathbf{m}}_j \\ -(\mathbf{x}_1^j - \tilde{\mathbf{m}}_j)' & \tilde{t}_j \end{vmatrix} = \begin{vmatrix} \mathbf{B}_2 & \mathbf{X}_2^{-j} \\ -\mathbf{X}_2^{-j\prime} & I_{p-1} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B} & \mathbf{X} \\ -\mathbf{X}' & I_p \end{vmatrix} \tag{19.67}$$

$$= \frac{|\mathbf{B}+\mathbf{X}\mathbf{X}'|}{|\mathbf{B}_2 + \mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|} \tag{19.68}$$

$$|\tilde{t}_j| = \begin{vmatrix} \mathbf{B}_2 & \mathbf{X}_2^{-j} \\ -\mathbf{X}_2^{-j\prime} & I_{p-1} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B}_2 & \mathbf{X}_2 \\ -\mathbf{X}_2' & I_p \end{vmatrix} \tag{19.69}$$

$$= \frac{|\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'|}{|\mathbf{B}_2 + \mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|} \tag{19.70}$$

$$|\tilde{\mathbf{B}}_j| = \begin{vmatrix} \mathbf{B}_2 & \mathbf{X}_2^{-j} \\ -\mathbf{X}_2^{-j\prime} & I_{p-1} \end{vmatrix}^{-1} \begin{vmatrix} \mathbf{B} & \mathbf{X}^{-j} \\ -\mathbf{X}^{-j\prime} & I_{p-1} \end{vmatrix} \tag{19.71}$$

$$= \frac{|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}|}{|\mathbf{B}_2 + \mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|} \tag{19.72}$$

Therefore, we want to calculate determinants of the form

$$\begin{vmatrix} \mathbf{B} & \mathbf{X}^{-j} \\ -\mathbf{X}^{-j\prime} & I_{p-1} \end{vmatrix} = |\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}| \tag{19.73}$$

All these determinants are related through the rank-one update formula:

$$|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}| = |\mathbf{B}+\mathbf{X}\mathbf{X}' - \mathbf{x}^j\mathbf{x}^{j\prime}| \tag{19.74}$$

$$= |\mathbf{B}+\mathbf{X}\mathbf{X}'| \cdot (1 - \mathbf{x}^{j\prime}\,[\mathbf{B}+\mathbf{X}\mathbf{X}']^{-1}\,\mathbf{x}^j) \tag{19.75}$$

For instance, by applying the update with $\mathbf{B}_2$ and $\mathbf{X}_2$, $\tilde{t}_j = |\tilde{t}_j|$ can be computed as:

$$\tilde{t}_j = \left(1 - \mathbf{x}_2^{j\prime}\left[\mathbf{B}_2 + \mathbf{X}_2\mathbf{X}_2'\right]^{-1}\mathbf{x}_2^j\right)^{-1} \tag{19.76}$$

Contrast this expression with the one obtained by direct application of (19.47):

$$\tilde{t}_j = 1 + \mathbf{x}_2^{j\prime}\left[\mathbf{B}_2 + \mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}\right]^{-1}\mathbf{x}_2^j \tag{19.77}$$

Given these relationships, we can express the conditional distribution of x_1^j as:

$$p(\mathbf{x}_1^j|\mathbf{X}_1^{-j},\mathbf{X}_2) = \frac{|\tilde{\mathbf{B}}_j|^{\frac{\nu+n+p-2}{2}}|\tilde{t}_j|^{\frac{\nu+n_2+p-1}{2}}}{\tau_{(\nu+n_2+p-1),n_1,1}}\begin{vmatrix}\tilde{\mathbf{B}}_j & \mathbf{x}_1^j-\tilde{\mathbf{m}}_j\\ -(\mathbf{x}_1^j-\tilde{\mathbf{m}}_j)' & \tilde{t}_j\end{vmatrix}^{-\frac{\nu+n+p-1}{2}} \tag{19.78}$$

$$= \frac{|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}|^{\frac{\nu+n+p-2}{2}}|\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2|^{\frac{\nu+n_2+p-1}{2}}}{\tau_{\dots}\cdot|\mathbf{B}_2+\mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|^{\frac{\nu+n_2+p-2}{2}}}|\mathbf{B}+\mathbf{X}\mathbf{X}'|^{-\frac{\nu+n+p-1}{2}} \tag{19.79}$$

$$= \tau^{-1}\cdot\frac{|\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'|^{\frac{\nu+n_2+p-1}{2}}}{|\mathbf{B}_2+\mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|^{\frac{\nu+n_2+p-2}{2}}}\cdot\frac{|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}|^{\frac{\nu+n+p-2}{2}}}{|\mathbf{B}+\mathbf{X}\mathbf{X}'|^{\frac{\nu+n+p-1}{2}}} \tag{19.80}$$

$$= \tau^{-1}\cdot\frac{|\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'|^{\frac{1}{2}}\cdot(1-\mathbf{x}^{j\prime}\,[\mathbf{B}+\mathbf{X}\mathbf{X}']^{-1}\,\mathbf{x}^j)^{\frac{\nu+n+p-2}{2}}}{(1-\mathbf{x}_2^{j\prime}\left[\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'\right]^{-1}\mathbf{x}_2^j)^{\frac{\nu+n_2+p-2}{2}}\cdot|\mathbf{B}+\mathbf{X}\mathbf{X}'|^{\frac{1}{2}}} \tag{19.81}$$

$$= \tau^{-1}\cdot\frac{|\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'|^{\frac{1}{2}}}{|\mathbf{B}+\mathbf{X}\mathbf{X}'|^{\frac{1}{2}}}\cdot\frac{(1-\mathbf{x}^{j\prime}\,[\mathbf{B}+\mathbf{X}\mathbf{X}']^{-1}\,\mathbf{x}^j)^{\frac{\nu+n+p-2}{2}}}{(1-\mathbf{x}_2^{j\prime}\left[\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'\right]^{-1}\mathbf{x}_2^j)^{\frac{\nu+n_2+p-2}{2}}} \tag{19.82}$$

As a special case, for $n_2 = 0$, we obtain a practical expression for the conditional distribution of a column x^j of X:

$$p(\mathbf{x}^j|\mathbf{X}^{-j}) = \frac{(1-\mathbf{x}^{j\prime}\,[\mathbf{B}+\mathbf{X}\mathbf{X}']^{-1}\,\mathbf{x}^j)^{\frac{\nu+n+p-2}{2}}}{\tau\cdot|\mathbf{B}+\mathbf{X}\mathbf{X}'|^{\frac{1}{2}}} \tag{19.83}$$

Contrast this expression with the one obtained without the rank-one update manipulations:

$$p(\mathbf{x}^j|\mathbf{X}^{-j}) = \frac{(1+\mathbf{x}^{j\prime}\left[\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}\right]^{-1}\mathbf{x}^j)^{-\frac{\nu+n+p-1}{2}}}{\tau\cdot|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}|^{\frac{1}{2}}} \tag{19.84}$$

In further sections we will establish radial distributions, the eventual application to the present LOO case is that we know the distribution:

$$\mathbf{x}^{j\prime}\,[\mathbf{B}+\mathbf{X}\mathbf{X}']^{-1}\,\mathbf{x}^j \mid \mathbf{X}^{-j} \sim \text{Beta}\left(\frac{n}{2},\frac{\nu+p-1}{2}\right) \tag{19.85}$$

More generally, for $n_2 \geqslant 0$, we know that

$$z_j = \tilde{\mathbf{B}}_j^{\frac{1}{2}}(x_1^j-\tilde{\mathbf{m}}_j)\tilde{t}_j^{\frac{1}{2}} \tag{19.86}$$

follows a standard multivariate t, given $\mathbf{X}_1^{-j}$ and $\mathbf{X}_2$. Therefore, we only need to know how to compute

$$\left(1+\mathbf{z}_j'\mathbf{z}_j\right)^{-1} = \left(1+\tilde{t}_j^{-1}(\mathbf{x}^{j\prime}-\tilde{\mathbf{m}}_j)\tilde{\mathbf{B}}_j^{-1}(\mathbf{x}^j-\tilde{\mathbf{m}}_j)\right)^{-1} \tag{19.87}$$

$$= \frac{\left|\tilde{\mathbf{B}}_j\right|\left|\tilde{t}_j\right|}{\begin{vmatrix}\tilde{\mathbf{B}}_j & \mathbf{x}_1^j-\tilde{\mathbf{m}}_j \\ -(\mathbf{x}_1^j-\tilde{\mathbf{m}}_j)' & \tilde{t}_j\end{vmatrix}} \tag{19.88}$$

$$= \frac{|\mathbf{B}+\mathbf{X}^{-j}\mathbf{X}^{-j\prime}||\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'|}{|\mathbf{B}+\mathbf{X}\mathbf{X}'||\mathbf{B}_2+\mathbf{X}_2^{-j}\mathbf{X}_2^{-j\prime}|} \tag{19.89}$$

$$= \frac{1-\mathbf{x}^{j\prime}\left[\mathbf{B}+\mathbf{X}\mathbf{X}'\right]^{-1}\mathbf{x}^j}{1-\mathbf{x}_2^{j\prime}\left[\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'\right]^{-1}\mathbf{x}_2^j} \tag{19.90}$$

Therefore,

$$\frac{1-\mathbf{x}^{j\prime}\left[\mathbf{B}+\mathbf{X}\mathbf{X}'\right]^{-1}\mathbf{x}^j}{1-\mathbf{x}_2^{j\prime}\left[\mathbf{B}_2+\mathbf{X}_2\mathbf{X}_2'\right]^{-1}\mathbf{x}_2^j} \mid \mathbf{X}_1^{-j},\mathbf{X}_2 \sim \mathrm{Beta}\left(\frac{\nu+n_2+p-1}{2},\frac{n_1}{2}\right) \tag{19.91}$$

19.6 Radial distribution for the multivariate-t

In the case $n = 1$, that is, when X is simply a (row) vector, the matrix-t simplifies to a multivariate-t distribution with density:

$$p_X(\mathbf{x}) = \pi^{-\frac{p}{2}}\frac{\Gamma\left(\frac{\nu+p}{2}\right)}{\Gamma\left(\frac{\nu}{2}\right)}\left(1+\|\mathbf{x}\|^2\right)^{-\frac{\nu+p}{2}} \tag{19.92}$$

This distribution is an example of a spherical distribution (the level sets of the density are spheres centered around 0; the distribution is left unchanged by orthonormal changes of coordinates). In that case, we can derive the marginal distribution of $r^2 = \|X\|^2$ through the polar change of variable $\mathbf{x} = r\mathbf{u}$, $r > 0$, $\mathbf{u} \in S^{p-1}$:

$$\left(1+\|\mathbf{x}\|^2\right)^{-\frac{\nu+p}{2}}\mathrm{d}\mathbf{x} = \left(1+r^2\right)^{-\frac{\nu+p}{2}}r^{p-1}\,\mathrm{d}r\,\mathrm{d}\mathbf{u} \tag{19.93}$$

which can be seen as a transformation of a beta distribution through the change of variable $v = (1+r^2)^{-1}$:

$$\left(1+r^2\right)^{-\frac{\nu+p}{2}}r^{p-1}\,\mathrm{d}r\,\mathrm{d}\mathbf{u} = \frac{1}{2}\left(1+r^2\right)^{-\frac{\nu+p}{2}}r^{p-2}\,\mathrm{d}r^2\,\mathrm{d}\mathbf{u} \tag{19.94}$$

$$= \frac{1}{2}v^{\frac{\nu+p}{2}}\left(\frac{1-v}{v}\right)^{\frac{p}{2}-1}v^{-2}\,\mathrm{d}v\,\mathrm{d}\mathbf{u} \tag{19.95}$$

$$= \frac{1}{2}v^{\frac{\nu}{2}-1}(1-v)^{\frac{p}{2}-1}\,\mathrm{d}v\,\mathrm{d}\mathbf{u} \tag{19.96}$$

In other words,

$$\frac{1}{1+\|X\|^2} \sim \text{Beta}\left(\frac{\nu}{2}, \frac{p}{2}\right) \tag{19.97}$$

This allows us to define radial credible regions for X, namely, given a vector $\mathbf{x}$, the probability of observing a standard-t distributed random vector X which lies further away from 0 — the radial survival function — is

$$P(\|X\|^2 \geqslant \|\mathbf{x}\|^2) = \frac{1}{\mathrm{B}\left(\frac{\nu}{2}, \frac{p}{2}\right)} \int_0^{\frac{1}{1+\|\mathbf{x}\|^2}} v^{\frac{\nu}{2}-1}(1-v)^{\frac{p}{2}-1}\,\mathrm{d}v \tag{19.98}$$

$$= I_{\frac{1}{1+\|\mathbf{x}\|^2}}\left(\frac{\nu}{2}, \frac{p}{2}\right) \tag{19.99}$$

where I is the regularized incomplete beta function.

19.7 Matrix-radial distribution for the matrix-t

It is also possible to perform a polar change of variable with matrix variables. If $X = RU$, with R, and consequently R^2, symmetric positive definite matrices and U a $n \times p$ semi-orthogonal matrix ($UU' = I_n$), we have

$$\mathrm{d}X = 2^{-n}|R|^{p-n-1}\,\mathrm{d}R^2\,\mathrm{d}U \tag{19.100}$$

(where $\mathrm{d}R^2$ is this time the measure induced by the embedding in $\mathbb{R}^{n,n}$).

Furthermore, R^2 can reparametrized by $V = (I_n + R^2)^{-1}$ s.p.d. such that $I_n - V$ is s.p.d too (which could be written $0 \leqslant V \leqslant I_n$) through the change:

$$\mathrm{d}R^2 = |V|^{-(n+1)}\,\mathrm{d}V \tag{19.101}$$

so the matrix-t distribution can be rewritten

$$|I_n + XX'|^{-\frac{\nu+p+n-1}{2}}\,\mathrm{d}X = 2^{-n}\left|I_n + R^2\right|^{-\frac{\nu+p+n-1}{2}}|R|^{p-n-1}\,\mathrm{d}R^2\,\mathrm{d}U \tag{19.102}$$

$$= 2^{-n}|V|^{\frac{\nu+p+n-1}{2}}|V^{-1}(I_n - V)|^{\frac{p-n-1}{2}}|V|^{-(n+1)}\,\mathrm{d}V\,\mathrm{d}U$$

$$= 2^{-n}|V|^{\frac{\nu}{2}-1}|I_n - V|^{\frac{p-n-1}{2}}\,\mathrm{d}V\,\mathrm{d}U \tag{19.103}$$

which generalize the polar forms of the multivariate-t from (19.94) and (19.96) to $n \geqslant 1$. However, since R^2 or V are still matrices here, distributed as matrix generalizations of the F and Beta distributions, respectively, this does not provide a way to compute a cumulative distribution for X.

20 Variational extensions of matrix-t distributions

The matrix-t distribution can be understood as modeling rows or columns of data drawn from a shared but unknown multivariate normal distribution. This is a strict assumption that do not allow any outliers: if a column, say, were to lie far outside the distribution defined by the rest of the columns, it would exert a strong pull on the posterior distribution of the latent shared covariance matrix.

We consider vectors $X^1, \dots, X^p$ with values in $\mathbb{R}^n$, each independently following a centered multivariate normal distribution $\mathcal{N}(0, [\lambda_j \mathbf{\Lambda}]^{-1})$ with precision matrix $\lambda_j \mathbf{\Lambda}$.

We then have the conditional density:

$$p(X|\boldsymbol{\lambda}, \mathbf{\Lambda}) = \prod_{j=1}^{p} p(\mathbf{x}^j|\lambda_j, \mathbf{\Lambda}) \tag{20.1}$$

$$= \prod_{j=1}^{p} (2\pi)^{-\frac{n}{2}} \lambda_j^{\frac{n}{2}} |\mathbf{\Lambda}|^{\frac{1}{2}} \mathrm{e}^{-\frac{1}{2}\lambda_j \mathbf{x}^{j\prime} \mathbf{\Lambda} \mathbf{x}^j} \tag{20.2}$$

Under a mean-field approximation, with independent priors on the λ_j, and independent auxiliary variables $\tilde{\lambda}_j$, we have the usual lower bound:

$$p(X|\mathbf{\Lambda}) = \mathbb{E}_{\mathbf{\Lambda}}\left[\prod_{j=1}^{p} p(\mathbf{x}^j|\lambda_j, \mathbf{\Lambda})\right] \tag{20.3}$$

$$= \prod_{j=1}^{p} \mathbb{E}_{\mathbf{\Lambda}}\left[p(\mathbf{x}^j|\lambda_j, \mathbf{\Lambda})\right] \tag{20.4}$$

$$\geqslant \prod_{j=1}^{p} \exp\left(\mathbb{E}_{\mathbf{\Lambda}}\left[\ln p(\mathbf{x}^j|\tilde{\lambda}_j, \mathbf{\Lambda})\right]\right) \mathrm{e}^{-\mathbb{KL}\left[\tilde{\lambda}_j \| \lambda_j\right]} \tag{20.5}$$

$$\geqslant \prod_{j=1}^{p} p(\mathbf{x}^j|\mathbb{E}\tilde{\lambda}_j, \mathbf{\Lambda}) \, \mathrm{e}^{-\frac{n}{2}(\ln \mathbb{E}\tilde{\lambda}_j - \mathbb{E}\ln\tilde{\lambda}_j)} \mathrm{e}^{-\mathbb{KL}\left[\tilde{\lambda}_j \| \lambda_j\right]} \tag{20.6}$$

$$\geqslant \prod_{j=1}^{p} p(\mathbf{x}^j|\mathbb{E}\tilde{\lambda}_j, \mathbf{\Lambda}) \, \mathrm{e}^{-\frac{n}{2}\mathbb{D}\tilde{\lambda}_j - \mathbb{KL}\left[\tilde{\lambda}_j \| \lambda_j\right]} \tag{20.7}$$

Where we define the logarithmic dispersion operator:

$$\mathbb{D}[Z] = \ln \mathbb{E}[Z] - \mathbb{E}[\ln Z] \geqslant 0 \tag{20.8}$$

which is non-negative by Jensen's inequality and reaches 0 when Z is constant (hence $\mathbb{D}$ can be seen as a measure of dispersion; at the first order, $\mathbb{D}[Z] \approx \frac{1}{2}\mathbb{V}[Z]/\mathbb{E}[Z]^2$).

Therefore, the marginal likelihood integrated over the λ_j can be approximated by the conditional likelihood at an average $\mathbb{E}\tilde{\lambda}_j$ point, with an extra penalty term that regularizes this average point.

The exact expression of the penalty term can be worked for simple priors. For instance, if we choose for $p(\lambda)$ a Gamma distribution $\mathrm{Ga}(\alpha, \beta)$, we have:

$$\begin{aligned}
\mathbb{E}\left[\ln p_{\lambda_j}(\tilde{\lambda}_j)\right] &= \ln p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) + (\alpha - 1)(\mathbb{E}\ln\tilde{\lambda}_j - \ln \mathbb{E}\tilde{\lambda}_j) && (20.9)\\
&= \ln p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) - (\alpha - 1)\mathbb{D}\tilde{\lambda}_j && (20.10)
\end{aligned}$$

Therefore the evidence lower bound is of the form

$$F(\mathbb{E}[\tilde{\lambda}_j]) + \frac{n}{2}\mathbb{E}[\ln\tilde{\lambda}_j] + (\alpha - 1)\mathbb{E}[\ln\tilde{\lambda}_j] + \mathbb{H}[\tilde{\lambda}_j] \tag{20.11}$$

for some function F.

The optimum of such a functional of the distribution of $\tilde{\lambda}_j$ can then be found by functional analysis to be given by:

$$p(\tilde{\lambda}_j) \propto \tilde{\lambda}_j^{\frac{n}{2}+\alpha-1} \mathrm{e}^{F'(\mathbb{E}\tilde{\lambda}_j)\tilde{\lambda}_j} \tag{20.12}$$

In other word, the optimal $\tilde{\lambda}_j$ is also Gamma-distributed as $\mathrm{Ga}(\alpha_j, \beta_j)$ with α_j known to be $\alpha_j = \alpha + \frac{n}{2}$ and we can express $\beta_j = \frac{\alpha_j}{\mathbb{E}\tilde{\lambda}_j}$ as a function of a single still unknown variational statistic $\mathbb{E}\tilde{\lambda}_j$.

Then, as a function of this unique variational parameter, the penalty can be re-expressed using:

$$\mathbb{E}\Big[\ln p_{\tilde{\lambda}_j}(\tilde{\lambda}_j)\Big] = \ln p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j) - (\alpha_j - 1)\mathbb{D}\tilde{\lambda}_j \tag{20.13}$$

and the corresponding divergence expression:

$$\begin{aligned}
\mathbb{KL}\left[\tilde{\lambda}_j \| \lambda_j\right] &= \mathbb{E}\Big[\ln p_{\tilde{\lambda}_j}(\tilde{\lambda}_j)\Big] - \mathbb{E}\left[\ln p_{\lambda_j}(\tilde{\lambda}_j)\right] && (20.14)\\
&= \ln p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j) - \ln p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) + (\alpha - \alpha_j)\mathbb{D}\tilde{\lambda}_j && (20.15)
\end{aligned}$$

to obtain the simplified expression:

$$\begin{aligned}
&\frac{n}{2}\mathbb{D}\tilde{\lambda}_j + \mathbb{KL}\left[\tilde{\lambda}_j \| \lambda_j\right] && (20.16)\\
&= \frac{n}{2}\mathbb{D}\tilde{\lambda}_j + \ln p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j) - \ln p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) + (\alpha - \alpha_j)\mathbb{D}\tilde{\lambda}_j && (20.17)\\
&= \ln p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j) - \ln p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) && (20.18)
\end{aligned}$$

The ELBO in the gamma prior case is then explicitly given by

$$p(X|\mathbf{\Lambda}) \geqslant \prod_{j=1}^{p} p(\mathbf{x}^j|\mathbb{E}\tilde{\lambda}_j, \mathbf{\Lambda}) \frac{p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j)}{p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j)} \tag{20.19}$$

where

$$p_{\lambda_j}(\mathbb{E}\tilde{\lambda}_j) = \mathrm{Ga}(\mathbb{E}\tilde{\lambda}_j|\alpha, \beta) \tag{20.20}$$

$$p_{\tilde{\lambda}_j}(\mathbb{E}\tilde{\lambda}_j) = \mathrm{Ga}\left(\mathbb{E}\tilde{\lambda}_j \middle| \alpha + \frac{n}{2}, \frac{\alpha + \frac{n}{2}}{\mathbb{E}\tilde{\lambda}_j}\right) \tag{20.21}$$

The hyperparameter β is easy to optimize out. The derivative of the log lower bound in (20.19) is:

$$\sum_{j=1}^{p} \left(\frac{\alpha}{\beta} - \mathbb{E}\tilde{\lambda}_j\right) \tag{20.22}$$

from which we can set

$$\beta = \alpha\left(\frac{1}{p}\sum_{j=1}^{p} \mathbb{E}\tilde{\lambda}_j\right)^{-1} \tag{20.23}$$

Code availability

Here, we recapitulate how the relevant code pertaining to the work of this thesis is organized and where it may be found.

- The repository at `https://github.com/emorice/gex-impute` contains the pipeline for benchmarking gene expression models, but not the models themselves, with which the results of Part I and all the figures therein were generated.

- The repository at `https://github.com/emorice/gemz` contains the statistical part, including all the models discussed in Part I and used by the pipeline above. It notably contains our SVD-derived probabilistic predictors from Chapter 2, the glue code to re-use the PEER software from Chapter 3, and above all the version of CMK from Chapter 4.

- The repository at `https://github.com/emorice/peer` hosts our re-packaging of the original PEER software, with no meaningful modification to the original code: the Python bindings and build system alone were updated by us for compatibility with modern Python pipelines.

- The repository at `https://github.com/emorice/gtex-qtl` contains the pipeline for eQTL calling on GTEx data, which generated the results of Part II, and therefore all QTL callers, our re-implementation of FastQTL and our integration of CMK included.

- The repository at `https://github.com/emorice/galp` hosts the workflow manager briefly described in Chapter 15 that powers both pipelines.

- The repository at `https://github.com/emorice/gemz-galp` is for integration of the pipeline manager and statistical models.

Each of these is an installable Python package, with a standardized `pyproject.toml` that specifies among other things whichever of the other packages it depends on. Licensing information is to be found along each package, typically the MIT free software license is offered.

Contributions

In some of the sections of this work, we have discussed material from the existing scientific literature, but to be able to build upon such material for our own implementations, evaluations and extensions, we found necessary to re-derive, comment, interpret or extend what we learned from existing authors. To help distinguish the novel elements of our work, we detail here for each section of this kind how our discussion of the matters differs from our source material.

Section	Contribution
3.5	While PEER has been widely used, we haven't found in papers citing PEER any note of how the use of the software differs from the original intended use described in the original publication.
4.1	This section mostly treats of the work of MacKay [Mac92b; Mac92a] but we focus on a particular case and explain the method in more details as we found the original material somewhat too brief to give a good intuition of the method.
4.2	This is the second part of our explanation of MacKay's ideas, here again developed in greater depth.
5.1	While this part fundamentally describes nothing else than indeterminate least squares (ILS), we framed the idea in a more probabilistic and multivariate angle that the material we have encountered, with an emphasis on the leave-one-gene-out case which is our independent derivation.
5.2	Here again, the concept is fundamentally the same as Tikhonov regularization and Ridge regression, but with a more probabilistic framing and more details on how to generalize this to a multivariate method and the leave-one-gene-out scenario.
10.2	This describes the approach pioneered by FastQTL, but we make the subtle change to our "plain" t-statistic that allow us to develop more direct estimation procedures than the original software which was forced to choose a trial value of degrees of freedom to proceed.
11.2	This exposes additional examples to argue in favor of the fundamental idea of parametric null distribution, still from FastQTL.

Section	Contribution
11.3	The details of the estimation procedure of FastQTL were not documented in the original publication and the logic of it had to be reconstructed by us from the source code.
16.1 ff.	The Moore-Penrose inverse is a classic tool that we used extensively in this work, but the link between it and the limit of L2 regularization procedures only became apparent to us at a later stage without having seen it discussed in existing material.
17.4	Though extremely classical we introduce a slightly different scaling with important consequences for our testing procedures.
17.5, 17.6	Our main source is Anderson, but we re-derived all results using more abstract and concise mathematics.

Bibliography

[Aba+16] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*. Mar. 16, 2016. DOI: 10.48550/arXiv.1603.04467. arXiv: 1603.04467[cs]. URL: http://arxiv.org/abs/1603.04467 (visited on 09/23/2024) (cit. on p. 107).

[Agu+17] François Aguet et al. "Genetic effects on gene expression across human tissues". In: *Nature* 550.7675 (Oct. 12, 2017), pp. 204–213. ISSN: 0028-0836. DOI: 10.1038/nature24277. URL: http://www.nature.com/articles/nature24277 (cit. on pp. 3, 65, 71, 72, 95).

[And03] T. W. Anderson. *An Introduction to Multivariate Statistical Analysis*. Google-Books-ID: A5QlrgEACAAJ. Wiley, July 25, 2003. 752 pp. ISBN: 978-0-471-36091-9 (cit. on p. 133).

[Ans+24] Jason Ansel et al. "PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation". In: *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. Vol. 2. ASPLOS '24. New York, NY, USA: Association for Computing Machinery, Apr. 27, 2024, pp. 929–947. ISBN: 9798400703850. DOI: 10.1145/3620665.3640366. URL: https://dl.acm.org/doi/10.1145/3620665.3640366 (visited on 09/23/2024) (cit. on p. 107).

[AP23] AP. *California tech CEO convicted in COVID-19 and allergy test fraud case sentenced to 8 years in prison*. AP News. Section: Business. Oct. 18, 2023. URL: https://apnews.com/article/mark-schena-arrayit-medical-fraud-sentencing-ad5e46e4805c9617c5ba57bffeb17670 (visited on 09/17/2024) (cit. on p. 2).

[Bat+14] Alexis Battle et al. "Characterizing the genetic basis of transcriptome diversity through RNA-sequencing of 922 individuals". In: *Genome Research* 24.1 (Jan. 1, 2014). Company: Cold Spring Harbor Laboratory Press Distributor: Cold Spring Harbor Laboratory Press Institution: Cold Spring Harbor Laboratory Press Label: Cold Spring Harbor Laboratory Press Publisher: Cold Spring Harbor Lab, pp. 14–24. ISSN: 1088-9051, 1549-5469. DOI: 10.1101/gr.

155192.113. URL: https://genome.cshlp.org/content/24/1/14 (visited on 09/20/2024) (cit. on p. 3).

[Bre+02] Rachel B. Brem et al. “Genetic Dissection of Transcriptional Regulation in Budding Yeast”. In: *Science* 296.5568 (Apr. 26, 2002). Publisher: American Association for the Advancement of Science, pp. 752–755. DOI: 10.1126/science.1069516. URL: https://www.science.org/doi/full/10.1126/science.1069516 (visited on 09/17/2024) (cit. on p. 2).

[Byc+18] Clare Bycroft et al. “The UK Biobank resource with deep phenotyping and genomic data”. In: *Nature* 562.7726 (2018). ISSN: 14764687. DOI: 10.1038/s41586-018-0579-z (cit. on p. 3).

[Con23] Consortium for Python Data API Standards. *Python array API standard — Python array API standard 2023.12 documentation*. Dec. 2023. URL: https://data-apis.org/array-api/latest/index.html (visited on 09/23/2024) (cit. on p. 108).

[Cup+07] L. Adrienne Cupples et al. “The Framingham Heart Study 100K SNP genome-wide association study resource: overview of 17 phenotype working group reports”. In: *BMC Medical Genetics* 8.1 (Sept. 19, 2007), S1. ISSN: 1471-2350. DOI: 10.1186/1471-2350-8-S1-S1. URL: https://doi.org/10.1186/1471-2350-8-S1-S1 (visited on 09/18/2024) (cit. on p. 3).

[Dar+11] Spyros Darmanis et al. “ProteinSeq: High-Performance Proteomic Analyses by Proximity Ligation and Next Generation Sequencing”. In: *PLOS ONE* 6.9 (Sept. 29, 2011). Publisher: Public Library of Science, e25583. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0025583. URL: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0025583 (visited on 09/20/2024) (cit. on p. 4).

[Di +17] Paolo Di Tommaso et al. “Nextflow enables reproducible computational workflows”. In: *Nature Biotechnology* 35.4 (Apr. 2017). Publisher: Nature Publishing Group, pp. 316–319. ISSN: 1546-1696. DOI: 10.1038/nbt.3820. URL: https://www.nature.com/articles/nbt.3820 (visited on 09/23/2024) (cit. on p. 116).

[Dum+00] Pierre Dumas et al. “Mapping of quantitative trait loci (QTL) of differential stress gene expression in rat recombinant inbred strains”. In: *Journal of Hypertension* 18.5 (2000), p. 545. ISSN: 0263-6352. URL: https://journals.lww.com/jhypertension/abstract/2000/18050/mapping_of_quantitative_trait_loci__qtl__of.6.aspx (visited on 09/17/2024) (cit. on p. 2).

[Ent19] Peter Andreas Entschev. *NEP 35 — Array creation dispatching with __array_function__ — NumPy Enhancement Proposals*. Oct. 15, 2019. URL: https://numpy.org/neps/nep-0035-array-creation-dispatch-with-array-function.html (visited on 09/23/2024) (cit. on p. 109).

[Era+22] Gökcen Eraslan et al. "Single-nucleus cross-tissue molecular reference maps toward understanding disease gene function". In: *Science* 376.6594 (May 13, 2022). Publisher: American Association for the Advancement of Science, eabl4290. DOI: 10.1126/science.abl4290. URL: https://www.science.org/doi/10.1126/science.abl4290 (visited on 09/20/2024) (cit. on p. 3).

[EY36] Carl Eckart and Gale Young. "The approximation of one matrix by another of lower rank". In: *Psychometrika* 1.3 (Sept. 1, 1936), pp. 211–218. ISSN: 1860-0980. DOI: 10.1007/BF02288367. URL: https://doi.org/10.1007/BF02288367 (visited on 09/05/2024) (cit. on p. 15).

[FJL18] Roy Frostig, Matthew James Johnson, and Chris Leary. "Compiling machine learning programs via high-level tracing". In: *Systems for Machine Learning* 4.9 (2018). Publisher: SysML (cit. on p. 107).

[Fod+91] Stephen P. A. Fodor et al. "Light-Directed, Spatially Addressable Parallel Chemical Synthesis". In: *Science* 251.4995 (Feb. 15, 1991). Publisher: American Association for the Advancement of Science, pp. 767–773. DOI: 10.1126/science.1990438. URL: https://www.science.org/doi/abs/10.1126/science.1990438 (visited on 09/18/2024) (cit. on p. 1).

[Fos+07] Eric J. Foss et al. "Genetic basis of proteome variation in yeast". In: *Nature Genetics* 39.11 (Nov. 2007). Publisher: Nature Publishing Group, pp. 1369–1375. ISSN: 1546-1718. DOI: 10.1038/ng.2007.22. URL: https://www.nature.com/articles/ng.2007.22 (visited on 09/20/2024) (cit. on p. 4).

[FW33] Ragnar Frisch and Frederick V. Waugh. "Partial Time Regressions as Compared with Individual Trends". In: *Econometrica* 1.4 (1933). Publisher: [Wiley, Econometric Society], pp. 387–401. ISSN: 0012-9682. DOI: 10.2307/1907330. URL: https://www.jstor.org/stable/1907330 (visited on 09/13/2024) (cit. on p. 78).

[Gam+15] Eric R. Gamazon et al. "A gene-based association method for mapping traits using reference transcriptome data". In: *Nature Genetics* 47.9 (2015). ISSN: 15461718. DOI: 10.1038/ng.3367 (cit. on pp. 5, 100).

[Ger+21] Zachary F Gerring et al. "E-MAGMA: an eQTL-informed method to identify risk genes using genome-wide association study summary statistics". In: *Bioinformatics* 37.16 (2021). ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btab115 (cit. on p. 5).

[GN99] A. K. Gupta and D. K. Nagar. *Matrix Variate Distributions*. New York: Chapman and Hall/CRC, Oct. 22, 1999. 384 pp. ISBN: 978-0-203-74928-9. DOI: 10.1201/9780203749289 (cit. on p. 149).

[GTE+15] The GTEx Consortium et al. "The Genotype-Tissue Expression (GTEx) pilot analysis: Multitissue gene regulation in humans". In: *Science* 348.6235 (May 8, 2015). Publisher: American Association for the Advancement of Science, pp. 648–660. DOI: 10.1126/science.1262110. URL: https://www.science.org/doi/abs/10.1126/science.1262110 (visited on 11/10/2022) (cit. on p. 3).

[GTE20] The GTEx Consortium. "The GTEx Consortium atlas of genetic regulatory effects across human tissues". In: *Science* 369.6509 (Sept. 11, 2020). Publisher: American Association for the Advancement of Science, pp. 1318–1330. DOI: 10.1126/science.aaz1776. URL: https://www.science.org/doi/10.1126/science.aaz1776 (visited on 09/14/2022) (cit. on pp. 3, 14, 71, 72).

[Gul89] Stephen F. Gull. "Developments in Maximum Entropy Data Analysis". In: *Maximum Entropy and Bayesian Methods*. Dordrecht: Springer Netherlands, 1989, pp. 53–71. DOI: 10.1007/978-94-015-7860-8_4. URL: http://link.springer.com/10.1007/978-94-015-7860-8_4 (cit. on p. 28).

[Gus+16] Alexander Gusev et al. "Integrative approaches for large-scale transcriptome-wide association studies". In: *Nature Genetics* 48.3 (2016). ISSN: 15461718. DOI: 10.1038/ng.3506. URL: https://doi.org/10.1038/ng.3506 (visited on 10/27/2020) (cit. on p. 5).

[Hal+22] Bjarni V. Halldorsson et al. "The sequences of 150,119 genomes in the UK Biobank". In: *Nature* 607.7920 (July 2022). Publisher: Nature Publishing Group, pp. 732–740. ISSN: 1476-4687. DOI: 10.1038/s41586-022-04965-x. URL: https://www.nature.com/articles/s41586-022-04965-x (visited on 09/18/2024) (cit. on p. 3).

[Har+20] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020). Publisher: Nature Publishing Group, pp. 357–362. ISSN: 1476-4687. DOI: 10.1038/s41586-020-2649-2. URL: https://www.nature.com/articles/s41586-020-2649-2 (visited on 09/23/2024) (cit. on p. 107).

[Her+06] Alan Herbert et al. "A Common Genetic Variant Is Associated with Adult and Childhood Obesity". In: *Science* 312.5771 (Apr. 14, 2006). Publisher: American Association for the Advancement of Science, pp. 279–283. DOI: 10.1126/science.1124779. URL: https://www.science.org/doi/full/10.1126/science.1124779 (visited on 09/18/2024) (cit. on p. 3).

[Hoy+18] Stephan Hoyer et al. *NEP 18 — A dispatch mechanism for NumPy's high level array functions — NumPy Enhancement Proposals*. May 29, 2018. URL: https://numpy.org/neps/nep-0018-array-function-protocol.html (visited on 09/23/2024) (cit. on p. 109).

[HS18] Stephan Hoyer and Nathaniel J. Smith. *NEP 22 — Duck typing for NumPy arrays – high level overview — NumPy Enhancement Proposals*. Mar. 22, 2018. URL: https://numpy.org/neps/nep-0022-ndarray-duck-typing-overview.html (visited on 09/23/2024) (cit. on p. 109).

[Joe+12] Roby Joehanes et al. "Gene expression analysis of whole blood, peripheral blood mononuclear cells, and lymphoblastoid cell lines from the Framingham Heart Study". In: *Physiological Genomics* 44.1 (Jan. 2012). Publisher: American Physiological Society, pp. 59–75. ISSN: 1094-8341. DOI: 10.1152/physiolgenomics.00130.2011. URL: https://journals.physiology.org/doi/full/10.1152/physiolgenomics.00130.2011 (visited on 09/18/2024) (cit. on p. 2).

[KR12] Johannes Köster and Sven Rahmann. "Snakemake—a scalable bioinformatics workflow engine". In: *Bioinformatics* 28.19 (Oct. 1, 2012), pp. 2520–2522. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts480. URL: https://doi.org/10.1093/bioinformatics/bts480 (visited on 09/23/2024) (cit. on p. 116).

[Lan+01] Eric S. Lander et al. "Initial sequencing and analysis of the human genome". In: *Nature* 409.6822 (Feb. 2001). Publisher: Nature Publishing Group, pp. 860–921. ISSN: 1476-4687. DOI: 10.1038/35057062. URL: https://www.nature.com/articles/35057062 (visited on 09/17/2024) (cit. on p. 1).

[Li+23] Shuwei Li et al. *Whole-genome sequencing of half-a-million UK Biobank participants*. Pages: 2023.12.06.23299426. Dec. 8, 2023. DOI: 10.1101/2023.12.06.23299426. URL: https://www.medrxiv.org/content/10.1101/2023.12.06.23299426v1 (visited on 09/18/2024) (cit. on p. 3).

[Lon+13] John Lonsdale et al. "The Genotype-Tissue Expression (GTEx) project". In: *Nature Genetics* (2013). ISSN: 10614036. DOI: 10.1038/ng.2653 (cit. on p. 3).

[Lov63] Michael C. Lovell. "Seasonal Adjustment of Economic Time Series and Multiple Regression Analysis". In: *Journal of the American Statistical Association* 58.304 (Dec. 1, 1963), pp. 993–1010. ISSN: 0162-1459. DOI: 10.1080/01621459.1963.10480682. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10480682 (visited on 09/13/2024) (cit. on p. 78).

[Luk17] Steven G. Luke. "Evaluating significance in linear mixed-effects models in R". In: *Behavior Research Methods* 49.4 (Aug. 1, 2017), pp. 1494–1502. ISSN: 1554-3528. DOI: 10.3758/s13428-016-0809-y. URL: https://doi.org/10.3758/s13428-016-0809-y (visited on 09/28/2024) (cit. on p. 84).

[Mac92a] David J. C. MacKay. "A Practical Bayesian Framework for Backpropagation Networks". In: *Neural Computation* 4.3 (May 1992), pp. 448–472. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.3.448. URL: https://direct.mit.edu/neco/article/4/3/448-472/5654 (cit. on pp. 28, 167).

[Mac92b] David J. C. MacKay. "Bayesian Interpolation". In: *Neural Computation* 4.3 (May 1992), pp. 415–447. ISSN: 0899-7667. DOI: 10.1162/neco.1992.4.3.415. URL: https://direct.mit.edu/neco/article/4/3/415-447/5639 (cit. on pp. 28, 167).

[Mor+04] Michael Morley et al. "Genetic analysis of genome-wide variation in human gene expression". In: *Nature* 430.7001 (Aug. 2004). Publisher: Nature Publishing Group, pp. 743–747. ISSN: 1476-4687. DOI: 10.1038/nature02797. URL: https://www.nature.com/articles/nature02797 (visited on 09/17/2024) (cit. on p. 3).

[Nag+08] Ugrappa Nagalakshmi et al. "The Transcriptional Landscape of the Yeast Genome Defined by RNA Sequencing". In: *Science* 320.5881 (June 6, 2008). Publisher: American Association for the Advancement of Science, pp. 1344–1349. DOI: 10.1126/science.1158441. URL: https://www.science.org/doi/abs/10.1126/science.1158441 (visited on 09/17/2024) (cit. on p. 2).

[Ong+16] Halit Ongen et al. "Fast and efficient QTL mapper for thousands of molecular phenotypes". In: *Bioinformatics* 32.10 (May 15, 2016), pp. 1479–1485. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btv722. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/btv722 (cit. on pp. 77, 82, 88, 92).

[Pra+95] M. Pravenec et al. "Mapping of quantitative trait loci for blood pressure and cardiac mass in the rat by genome scanning of recombinant inbred strains". In: *The Journal of Clinical Investigation* 96.4 (Oct. 1995), pp. 1973–1978. ISSN: 0021-9738. DOI: 10.1172/JCI118244 (cit. on p. 2).

[Sch+95] Mark Schena et al. "Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray". In: *Science* 270.5235 (Oct. 20, 1995). Publisher: American Association for the Advancement of Science, pp. 467–470. DOI: 10.1126/science.270.5235.467. URL: https://www.science.org/doi/abs/10.1126/science.270.5235.467 (visited on 09/17/2024) (cit. on p. 1).

[Sha12] A. A. Shabalin. "Matrix eQTL: ultra fast eQTL analysis via large matrix operations". In: *Bioinformatics* 28.10 (May 15, 2012), pp. 1353–1358. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts163. URL: https://academic.oup.com/bioinformatics/article-lookup/doi/10.1093/bioinformatics/bts163 (cit. on pp. 77, 82).

[Söd+06] Ola Söderberg et al. "Direct observation of individual endogenous protein complexes in situ by proximity ligation". In: *Nature Methods* 3.12 (Dec. 2006). Publisher: Nature Publishing Group, pp. 995–1000. ISSN: 1548-7105. DOI: 10.1038/nmeth947. URL: https://www.nature.com/articles/nmeth947 (visited on 09/20/2024) (cit. on p. 4).

[Ste+10] Oliver Stegle et al. "A Bayesian Framework to Account for Complex Non-Genetic Factors in Gene Expression Levels Greatly Increases Power in eQTL Studies". In: *PLoS Computational Biology* 6.5 (May 6, 2010). Ed. by Aviv Regev, e1000770. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1000770. URL: https://dx.plos.org/10.1371/journal.pcbi.1000770 (cit. on p. 21).

[Stu08] Student. "The Probable Error of a Mean". In: *Biometrika* 6.1 (1908). Publisher: [Oxford University Press, Biometrika Trust], pp. 1–25. ISSN: 0006-3444. DOI: 10.2307/2331554. URL: https://www.jstor.org/stable/2331554 (visited on 09/29/2024) (cit. on p. 129).

[Sun+23] Benjamin B. Sun et al. "Plasma proteomic associations with genetics and health in the UK Biobank". In: *Nature* 622.7982 (Oct. 2023). Publisher: Nature Publishing Group, pp. 329–338. ISSN: 1476-4687. DOI: 10.1038/s41586-023-06592-6. URL: https://www.nature.com/articles/s41586-023-06592-6 (visited on 09/20/2024) (cit. on p. 4).

[Tip01] Michael E. Tipping. "Sparse Bayesian Learning and the Relevance Vector Machine". In: *Journal of Machine Learning Research* 1 (Jun 2001), pp. 211–244. ISSN: ISSN 1533-7928. URL: https://jmlr.csail.mit.edu/papers/v1/tipping01a.html (visited on 09/13/2024) (cit. on p. 32).

[TNP94] Benjamin A. Taylor, Ann Navin, and Sandra J. Phillips. "PCR-Amplification of Simple Sequence Repeat Variants from Pooled DNA Samples for Rapidly Mapping New Mutations of the Mouse". In: *Genomics* 21.3 (June 1, 1994), pp. 626–632. ISSN: 0888-7543. DOI: 10.1006/geno.1994.1323. URL: https://www.sciencedirect.com/science/article/pii/S0888754384713231 (visited on 09/17/2024) (cit. on p. 2).

[Vir+20] Pauli Virtanen et al. "SciPy 1.0: fundamental algorithms for scientific computing in Python". In: *Nature Methods* 17.3 (Mar. 2020). Publisher: Nature Publishing Group, pp. 261–272. ISSN: 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: https://www.nature.com/articles/s41592-019-0686-2 (visited on 09/23/2024) (cit. on p. 107).

[Võs+21] Urmo Võsa et al. "Large-scale cis- and trans-eQTL analyses identify thousands of genetic loci and polygenic scores that regulate blood gene expression". In: *Nature Genetics* 53.9 (Sept. 2021). Publisher: Nature Publishing Group, pp. 1300–1310. ISSN: 1546-1718. DOI: 10.1038/s41588-021-00913-z. URL: https://www.nature.com/articles/s41588-021-00913-z (visited on 09/20/2024) (cit. on p. 4).

[Wei+10] Irene Weibrecht et al. "Proximity ligation assays: a recent addition to the proteomics toolbox". In: *Expert Review of Proteomics* 7.3 (June 1, 2010). Publisher: Taylor & Francis _eprint: https://doi.org/10.1586/epr.10.10, pp. 401–

409. ISSN: 1478-9450. DOI: 10.1586/epr.10.10. URL: https://doi.org/10.1586/epr.10.10 (visited on 09/20/2024) (cit. on p. 4).

[Win+98] Elizabeth A. Winzeler et al. “Direct Allelic Variation Scanning of the Yeast Genome”. In: *Science* 281.5380 (Aug. 21, 1998). Publisher: American Association for the Advancement of Science, pp. 1194–1197. DOI: 10.1126/science.281.5380.1194. URL: https://www.science.org/doi/full/10.1126/science.281.5380.1194 (visited on 09/18/2024) (cit. on p. 2).

[Wis28] John Wishart. “The Generalised Product Moment Distribution in Samples from a Normal Multivariate Population”. In: *Biometrika* 20A.1 (Dec. 1, 1928), pp. 32–52. ISSN: 0006-3444. DOI: 10.1093/biomet/20A.1-2.32. URL: https://doi.org/10.1093/biomet/20A.1-2.32 (visited on 09/29/2024) (cit. on p. 133).