Probabilistic Machine Learning with Omics Data and Biological Prior Knowledge

by

David Merrell

A dissertation submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2023

Date of final oral examination: 06/30/2023

The dissertation is approved by the following members of the Final Oral Committee:
Anthony Gitter, Associate Professor, Biostatistics and Medical Informatics
Michael Newton, Professor, Statistics
Xiaojin Jerry Zhu, Professor, Computer Sciences
Yixuan Sharon Li, Assistant Professor, Computer Sciences

© Copyright by David Merrell 2023
All Rights Reserved

To Ma and Pa.

ACKNOWLEDGMENTS

A long list of people helped me finish this dissertation.

My advisor, Tony Gitter, accepted me into his group as I started my fourth year of graduate school. He took a big chance on me. I am extremely grateful for that and have striven to make him proud of his decision.

My family has been a reliable source of support and encouragement. We are scattered across the country, but our group chat and Zoom calls keep us close.

I write these acknowledgements just after finishing a delightful dinner with my girlfriend. She has been an anchor of sanity as I approach the end of my PhD. Danielle: thank you, from the bottom of my heart, for being so kind and generous through this time.

Friends and labmates occasionally pulled me away from my computer/books/whiteboard to have some fun. These included Mark Mansi, Sam Gelman, Ben Kaufman, Akhil Guliani, and Danny McNeela among others. The bike rides, SACM events, rooftop barbeques, and evenings at the Terrace lifted my spirits.

Am I allowed to acknowledge a city? Madison has been an ideal setting for my graduate studies. I grew a lot during my years here. I will always regard Madison as home.

I can trace the credit for this dissertation much farther back in time than grad school, though. Specific people opened doors of opportunity for me when I was young—educators and other mentors. It may surprise some of those people to find their names here, but I want to mention them anyway.

Debbie Luoma taught me to read, write, and sail by ash breeze. I've been sailing that way ever since. If my writing has any quality, I have Mrs. Luoma to thank.

Russell Lee and Marlene Dwyer awakened me to the challenge and beauty of physics and mathematics. I would not have found my way to science, mathematics or computing without their thoughtful teaching and encouragement.

I worked with Jeff Christian, Lynn Wood, Allen Robinson, Chris Mouton, and Jia Xu during internships and other employment prior to graduate school. Each of these people gave me the freedom to tackle challenging, meaningful projects in a self-directed fashion. Those experiences molded my professional interests and served as crucial stepping stones in my career.

Finally, I'll emphasize the credit due to my parents. My dad has been a constant source of wisdom and advice. Beyond merely keeping me alive and raising me out of childhood, my mom educated me and cultivated my interest in math, science and engineering. I can't summarize her contributions here. That would take another book.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Modern biological research requires sophisticated analysis of *omics data*. For the purposes of this dissertation, "omics data" includes many commonly-used modalities collected in biological experiments. These include genomic, transcriptomic, proteomic, epigenomic, and other kinds of data. Omics datasets can be high-dimensional and complex. For instance, RNA-seq datasets with tens of thousands of dimensions are common. Datasets may be multimodal, with measurements collected by distinct assay technologies. Furthermore, samples are not usually independent and identically distributed (*i.i.d.*). Nuisance factors or experimental conditions may cause distributional differences between groups of samples. Samples may come from a time series, or different points in space. Missing values are common, but not necessarily random. These issues can lead to incorrect conclusions if they aren't modeled correctly.

We do not analyze biological data in a vacuum, though. Biologists have spent decades accumulating insights about biological systems. In principle, this prior knowledge has the potential to strengthen data analyses by (*i*) *biasing* inferences toward more probable solutions and (*ii*) making the solutions more interpretable. *Biological pathways* are a particularly rich form of prior knowledge. Pathways encode well-studied molecular processes that govern cells. Public databases like Reactome and KEGG curate these pathways in forms that are computationally accessible—formal ontologies, networks, or gene sets. However, biological prior knowledge also poses challenges. It may be too *incomplete* or *context-specific* to be useful in analyzing new data.

Probabilistic models provide a natural framework for extracting insights from data *and* prior knowledge. Data can be modeled as *observed variables* and prior knowledge can be encoded in a *prior distribution*. We can then estimate quantities of interest via *posterior inference*.

A central question motivating this dissertation is *what value, if any, can biological prior knowledge provide in omics data analysis?* To that end, this dissertation presents two probabilistic models that combine omics data and pathway prior knowledge. Chapter 2 presents a method to infer the *structure* of a signaling pathway from time series phosphoproteomic data and prior knowledge about signaling pathways. Chapter 3 proposes a matrix factorization model for multiomic data. The method uses biological prior knowledge to generate an *interpretation* of its outputs.

These models also entail the design of inference procedures that are principled and computationally efficient. The performance of each model is evaluated on simulated and real datasets. Their code is distributed on GitHub, and care is taken to make the analyses reproducible via workflow managers.

Biological prior knowledge is found to be a mixed blessing. In particular: while pathways are a useful tool for biologists to organize their knowledge, their utility as a Bayesian prior is found to be questionable.

1 INTRODUCTION

1.1 Motivation

Modern biological research involves (*i*) collecting large amounts of experimental data and (*ii*) extracting useful insights from it via computational tools. There are many *kinds* of data a biologist might collect, depending on the phenomenon they're investigating. These include genomic, transcriptomic, proteomic, and epigenomic data, among others. Biologists refer to these kinds of data generically as "omics data."

Extracting insight from omics datasets can be statistically challenging for several reasons. Omics data is typically high-dimensional. For instance, RNA-seq datasets with tens of thousands of features are commonplace. Moreover, the number of samples is often much smaller than the number of features. Omics data can also be multimodal—i.e., features may represent different *views* of a biological system, measured by totally different technologies.

Additional challenges come from the fact that samples in biological data are not usually *i.i.d.*. Some distributional differences arise from *technical effects* during data collection. For example, data collected on different machines, at different times, or in different laboratories, will generally contain systematic differences. These effects may account for a large fraction of the variation in a dataset. Other distributional differences have a biological source. Samples may belong to different *conditions*: treatment vs. control, healthy vs. diseased, etc. They may represent points in a time series, or measurements at different spatial locations in a tissue sample. Careful modeling must discern between (*i*) nuisance effects and (*ii*) scientifically interesting variation.

Omics data can be challenging to model. However, omics data is not analyzed in a vacuum. Biologists have already spent decades accumulating knowledge about biological systems. In principle, this prior knowledge has the potential to strengthen data analyses by (*i*) *biasing* inferences toward probable solutions and (*ii*) *interpreting* the outputs in biologically familiar terms. Biological prior knowledge poses its own challenges, though. It may be too *incomplete* or *context-specific* to be useful for the task at hand.

Ideally, a bioinformatician would have computational tools that make the most of available data and prior knowledge. This dissertation documents my work building and evaluating tools of that kind.

1.2 Scope

The work in this dissertation stems from a simple idea: that *probabilistic models* are a natural way to extract signal from complicated datasets and prior knowledge. In particular, this dissertation describes probabilistic models for two different settings involving *omics* data and *biological* prior knowledge.

- Chapter 2 presents a model that infers the structure of a signaling pathway from time series phosphoproteomic data and a prior network.

- Chapter 3 demonstrates a matrix factorization model for multiomic data. The model uses curated gene sets to interpret the outputs in a biologically meaningful way.

I find that probabilistic models are a powerful tool for addressing the challenges of omics datasets. However, I also arrive at some nuanced conclusions about the value of biological prior knowledge.

Abstractly, both models in this dissertation use the typical Bayesian formulation to combine data and priors in a posterior distribution:

$$\mathsf{P}(\mathsf{Z} \mid \mathsf{X}, \mathsf{B}) \;\propto\; \mathsf{P}(\mathsf{X} \mid \mathsf{Z}) \cdot \mathsf{P}(\mathsf{Z} \mid \mathsf{B})$$

where Z is an unknown quantity of interest, X is the data, and B is the biological prior knowledge. However, they apply this abstraction in very different fashions, highlighting the flexibility and creativity afforded by a Bayesian perspective.

A model is only the beginning of a solution, though—inference procedures present the bulk of the creative work in probabilistic modeling. Both projects in this dissertation required the design of bespoke inference techniques. For example, the model in Chapter 2 uses a specially tailored Markov Chain Monte Carlo algorithm (MCMC) (Gilks, 2005) to sample from the posterior distribution much more efficiently than off-the-shelf solutions. Meanwhile, the model in Chapter 3 employs a combination of (*i*) Expectation-Maximization (Dempster et al., 1977) and (*ii*) gradient-based optimization to obtain a point estimate maximizing the posterior density. In both cases I highlight helpful computational frameworks that aid algorithm development.

The models in this dissertation are useful for some purposes, but not others. They are best understood as tools for *exploratory data analysis*. In the parlance of machine learning, they are *unsupervised learners*. In a scientific setting this class of tool is valuable for summarizing and interpreting data, detecting patterns, and generating hypotheses. For example, principal component analysis (PCA) (Hotelling, 1933) and hierarchical clustering (Murtagh and Contreras, 2012) are standard tools in a bioinformatician's kit. I imagine the models in this dissertation playing a similar role to those classic techniques.

It's also worth specifying some topics *not* in the scope of this dissertation. Parts of my PhD research will not be discussed—I exclude them because they have no relation to omics data or biological priors. My work in the design of adaptive clinical trials (Merrell et al., 2023) falls into this category. This dissertation also elides my earlier research at the intersection of probabilistic inference and formal logic (Merrell et al., 2017).

This dissertation will *not* focus on hypothesis testing. The models are not constructed to *test* hypotheses. So they do not come equipped with ways to e.g., compute p-values for model outputs. A p-value is only as good as its null hypothesis. These models may be applied in a variety of contexts, without an obvious choice of null hypothesis. This is by intention—it is not necessarily a shortcoming. Scientists need tools to help them interpret complicated datasets and *form* hypotheses. Then, given a hypothesis, biostatisticians are free to apply context-appropriate statistical tests as they see fit.

Neural networks do not show up anywhere in this dissertation. There are practical and principled reasons for this. Interpretability is important in scientific settings. There exist many techniques to interpret neural networks and their inferences, but it can be difficult to choose one suitable for a given model. For example, a recent review by Chen et al. (2023) describes 24 distinct techniques based on Shapley values alone, and identifies important differences between them. In contrast, the methods in this dissertation are interpretable *by construction*, and in terms that a biologist would find familiar.

Issues related to data provide another reason to avoid neural networks. Not all laboratories produce the quantity of data necessary to fit neural networks. Furthermore, the data produced in different experiments contains *technical variation*—i.e., non-biological distributional differences. This creates a challenge for combining datasets or sharing pretrained models between experiments.

Computational expense is yet another consideration. Not all labs have the computational resources to fit competitive neural networks on their data. In contrast, the methods in this dissertation are comparatively lightweight. They can be run in minutes or hours on a consumer-grade workstation. This is an important niche to fill—productive biologists rely on such methods to interpret their data and generate hypotheses. I suspect deep learning is a valuable tool for analyzing omics data in the largest, best-resourced laboratories. But the need for flexible, inexpensive techniques on modest datasets will not disappear soon.

1.3 Biological background

This is a Computer Sciences dissertation, but it relies on a fair amount of biological background. A brief primer on the relevant biology seems appropriate, in order to make this self-contained and accessible to a multidisciplinary audience. Biologists may take issue with the coverage of some topics, but this section targets a broader "data scientist" audience.

Omics data

Anyone who begins working with biological data quickly comes into contact with the "-ome/-omic" suffix (Baker, 2013). Bioinformaticians use this jargon to denote a collection of items that form a whole. For instance, the set of all genes in a cell forms its *genome*, and the set of all proteins forms its *proteome*. Since biology is complicated, a large number of measurements are necessary to accurately capture the state of a biological system. Measuring *all* items in an "-ome" has appeal, since it offers the most complete view of a system.

Many -omes have been widely studied. One useful way to map them out and understand their relationships is through the *central dogma of molecular biology.* Every cell contains genes encoded in DNA. The central dogma says that genes are *transcribed* into mRNA, which is then *translated* into protein. In other words, there is a flow of information from DNA to mRNA, and from mRNA to protein. The protein ultimately determines the behavior and structure of the cell. See Figure 1.1 for illustration. We call the set of all genes the *genome*; the set of all mRNA the *transcriptome*; and the set of all proteins the *proteome*.

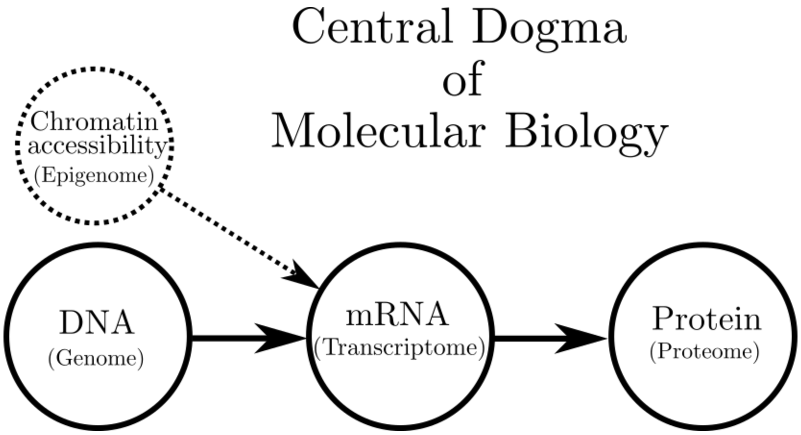


Figure 1.1: The central dogma. DNA is *transcribed* into mRNA, which is then *translated* into protein. Epigenetics affects the *accessibility* of DNA, exerting downstream effects on mRNA and protein. The epigenome is not usually included in diagrams of the central dogma, but I include it with dashed lines since epigenetic data appears in this dissertation.

An important related phenomenon is *epigenetics*, which involves certain chemical modifications to the DNA. These modifications do *not* change the genetic code itself; however, in some cases they make the DNA inaccessible for transcription (and, in other cases, make it more accessible). Hence, epigenetics has a downstream effect on mRNA (the transcriptome) and protein (the proteome). The set of all epigenetic modifications is sometimes referred to as the epigenome. Epigenetics is not usually included in the central dogma, but it bears mentioning here since epigenomic data will play a role in Chapter 3.

Several kinds of omics data appear in this dissertation—genomic, transcriptomic, proteomic, and epigenomic. I provide brief descriptions of them for the benefit of non-biologist readers. Perhaps the most important takeaway is that omics datasets come from complicated laboratory procedures. There are many points where inconsistencies may be introduced between datasets. Data scientists can easily draw incorrect conclusions from omics data if they don't understand these issues at a basic level.

Genomic data. Genomic data captures a biological system's genetic code, i.e., its DNA. In its most complete form, genomic data comprises the entire *sequence* of nucleotide bases ("characters" *A, T, G*, and *C*) for the system in question. For example, the human genome is a sequence of 3.2 billion bases.

However, for many analyses the full genomic sequence is unnecessary; a higher-level summary of the genome suffices. For instance, one may be interested in a genome's *differences* with respect to some reference genome. Genomic differences can be summarized in terms of *insertions, deletions*, and *substitutions* of nucleotides with respect to the reference genome. Chapter 3 employs *somatic mutation* data, which represents genomic differences between a cancer patient's *tumor* and *healthy* tissue. That somatic mutation data comes from The Cancer Genome Atlas; see their documentation for details about their data collection and processing (Documentation, 2022b).

Copy number alteration (CNA) gives an even coarser summary of genomic state. The human genome contains roughly 20,000 genes—sequences known to encode proteins or have some other function. A human genome normally possesses two copies of each gene, inherited from the mother and father. However, errors in DNA replication and repair can cause entire sections of the genome to be deleted or copied within a cell. As a result, the cell will no longer have *two* copies of the affected genes; it may have greater or fewer. Copy number alteration data summarizes these changes in a biological sample. The analyses in Chapter 3 use CNA data from a patient's tumor, relative to healthy tissue. See The Cancer Genome Atlas documentation for details about their CNA data preparation (Documentation, 2022a).

Other kinds of genomic data exist and are widely used, but will not appear in this dissertation. Single Nucleotide Polymorphism (SNP) arrays identify the nucleotides at predefined locations in the genome (LaFramboise, 2009) that are known to vary between individuals. SNP arrays are widely used in Genome-Wide Association Studies (GWAS) and consumer genetic tests such as 23andMe (23andMe, 2013). Whole-genome sequencing is less common, though its cost continues to decrease.

Transcriptomic data. Cells *transcribe* DNA into messenger RNA (mRNA). mRNA plays a key role in cellular processes, especially the creation of protein. The *transcriptome* refers to the collection of all RNA transcripts in a biological sample. Hence, the transcriptome provides a highly informative view of the *state* of a biological sample. Transcriptomic data aims to summarize the transcriptome in a useful, quantitative way.

Gene expression refers to the amount of mRNA transcribed from a given gene. We say a gene is *highly expressed* if its transcripts are highly abundant. Increased expression suggests that a gene is actively playing a role in the biological system's molecular processes. RNA-sequencing (RNA-seq) is a widely used transcriptomic assay that quantifies the gene expression in a biological sample. RNA-seq produces a vector of numeric values for each biological sample, having dimension equal to the number of genes. In a correctly-prepared RNA-seq dataset, higher numeric values indicate higher gene expression.

Importantly, each biological sample typically contains many cells. A small tissue or blood sample, for instance, may contain thousands of cells. Hence, the gene expression vector for a sample—measured by RNA-seq—represents an *aggregate* across all the cells in that sample. For that reason, bioinformaticians frequently refer to this kind of data as *bulk* RNA-seq. Conesa et al. provide a detailed survey of best practices for collecting and processing RNA-seq data (Conesa et al., 2016). Chapter 3 in this dissertation uses bulk RNA-seq data from The Cancer Genome Atlas; See TCGA's documentation for more details about their assay protocols (Documentation, 2022d).

Transcriptomic assays are an area of rapid technological progress. A full survey of recent developments is not appropriate for this dissertation. However, I briefly describe two of them in order to provide a sense of the emerging opportunities in this area, especially for data scientists.

Single-Cell RNA-seq (scRNA-seq) has gained widespread adoption in very recent years. Given a biological sample, scRNA-seq measures the gene expression *for each cell in the sample*, revealing its cell-level heterogeneity. For that reason, scRNA-seq can be much more informative than bulk RNA-seq. However, scRNA-seq also poses new challenges for data analysis. Samples can exhibit greater noise and more missing values than bulk RNA-seq. Luecken and Theis provide a helpful overview of the opportunities and challenges posed by scRNA-seq (Luecken and Theis, 2019).

Spatial transcriptomics improves on RNA-seq by introducing *spatial context* to transcriptomic measurements. Real biological systems—e.g., tissues—occupy three-dimensional space, and their spatial organization carries important biological information. Spatial transcriptomics aims to measure gene expression as a function of *location* in a biological sample. As a result, machine learning on spatial transcriptomics entails methods similar to those in computer vision. A review by Zhuang describes some of these developments, with particular emphasis on the Multiplexed Error-Robust Fluorescence in situ Hybridization (MERFISH) assay (Zhuang, 2021; Chen et al., 2015).

Despite these exciting developments in transcriptomics, only bulk RNA-seq makes an appearance in this dissertation. The model described in Chapter 3 could be used with scRNA-seq data, but this document does not explore that possibility.

Proteomic data. The *proteome* refers to the set of all proteins in a biological sample. Proteins serve as the building material for all living things. Proteins also carry out the complex molecular processes that govern cells. Hence, proteomic data can provide rich insights about the structure and function of cells.

Many assays exist for collecting proteomic data. A full survey is far beyond the scope of this dissertation. However, Reverse Phase Protein Array (RPPA) data is worth discussing since it appears in both Chapters 2 and 3. I will also describe mass spectrometry, since it has overtaken RPPA to become the dominant proteomics assay technology—I would be remiss not to mention it.

RPPA is a widely-used assay with many applications in proteomics. RPPA exposes a biological sample to specially-designed *antibodies* that (*i*) bind to specific proteins in the sample and (*ii*) emit light of a particular frequency via *fluorescence*. The assay apparatus quantifies the abundance of a protein via imaging, by measuring the intensity of fluorescence for the corresponding antibody. A recent paper by Coarfa et al. provides a helpful description of RPPA best practices in a cancer research setting (Coarfa et al., 2021).

RPPA has some notable limitations. While the human body contains roughly 20,000 unique proteins, the typical RPPA assay can only quantify tens or hundreds of proteins. Correctly prepared RPPA data is normalized in a relative fashion, such that comparisons are valid *within* an experiment. However, comparisons are not generally valid *between* experiments. For the work in Chapter 3 of this dissertation, I considered modeling RPPA data from The Cancer Genome Atlas (Documentation, 2023). However, concerns about the relative quantity and quality (Upadhya and Ryan, 2023) of TCGA's RPPA data led us to ignore it in favor of more informative omics modalities.

Beyond quantifying the *abundance* of a protein, RPPA can also measure the *proportion* of a protein that has been *modified* in some fashion. An important class of protein modification is *phosphorylation*, in which a phosphoryl molecular group binds to a protein. Phosphorylation plays a vital role in cellular function. It controls much of the communication—i.e., *signaling*—between cells and within cells. Phosphorylation in many cases "switches on" signaling proteins (e.g., kinases) that would otherwise be inactive. *Phosphoproteomics* aims to shed light on cell signaling by quantifying the phosphorylation of proteins in a biological sample. RPPA can measure phosphoproteomics by employing antibodies that specifically bind to phosphorylated proteins (Coarfa et al., 2021). The resulting data allows comparison of protein phosphorylation levels between samples or biological conditions. With adequate statistical modeling, this can yield insight about signaling processes in the samples.

In this dissertation, Chapter 2 describes a probabilistic model that infers the structure of a signaling network from phosphoproteomic time series data. The model is evaluated using RPPA phosphoproteomic data from a DREAM challenge (Hill et al., 2016). The challenge website provides other details about their RPPA experimental protocols (info@sagebase.org, 2013).

It's also worth mentioning mass spectrometry ("mass spec"), the dominant proteomics assay technology (Aebersold and Mann, 2003). Broadly speaking, a mass spectrometer is an instrument that identifies molecules by (*i*) converting them to an ionized gas, (*ii*) emitting them at high velocity through an electric (or magnetic) field, and (*iii*) observing their *mass-to-charge ratio*. This can be applied to the proteins in a biological sample. Proteins are broken down into smaller pieces called *peptides*, and then passed through a mass spectrometer. The peptides' mass-to-charge ratios provide a remarkably specific signature, allowing software to identify and quantify proteins in the sample. Mass spec is a flexible and highly sensitive technology, allowing it to be used in a variety of proteomic applications. For instance, it can detect phosphorylation or other minute changes to proteins (Grimsrud et al., 2010). It can also be used to identify protein-protein interactions via cross-linking (O'Reilly and Rappsilber, 2018).

Epigenomic data. Abstractly, DNA is a sequence of nucleotide "characters" drawn from $\{\mathrm{A}, \mathrm{T}, \mathrm{G}, \mathrm{C}\}$. However, in reality it exists as *chromatin* molecules in three-dimensional space, and possesses a complex, coiled structure. This structure implies that a given gene may or may not be *accessible* for transcription, depending on the physical context. In particular, a gene is only accessible if its portion of the chromatin *uncoils*, exposing the gene for transcription.

Epigenetics encompasses mechanisms that govern the *accessibility* of DNA. Epigenomic data quantifies the epigenetics of a biological system, and can be a useful complement to other omics. Many epigenomic assays exist, but I restrict this discussion to *methylation bisulfite sequencing* data, since it appears in Chapter 3. Methylation is an epigenetic mechanism wherein a methyl group binds to the chromatin, almost always on a C (cytosine). In many cases methylation acts on the chromatin in a way that renders the the DNA *inaccessible* (Moore et al., 2013). Bisulfite sequencing exposes the DNA to sodium bisulfite. This causes the *unmethylated* Cs in the DNA to convert to a different molecule (uracil); but has *no* effect on the methylated Cs. After many additional steps (elided for brevity), software deduces the locations of methylated Cs by comparing the sequence of the DNA exposed to sodium bisulfite against the sequence of the original DNA (Rauluseviciute et al., 2019).

At this point the methylation can be quantified in various ways. For instance, the data used in Chapter 3 consist of numeric values for each *gene*, indicating the extent to which that gene (and regions related to it) was methylated in a given sample. That particular data comes from The Cancer Genome Atlas; see their documentation (Documentation, 2022c) and the review by Dedeurwaerder et al. (2014) for additional details.

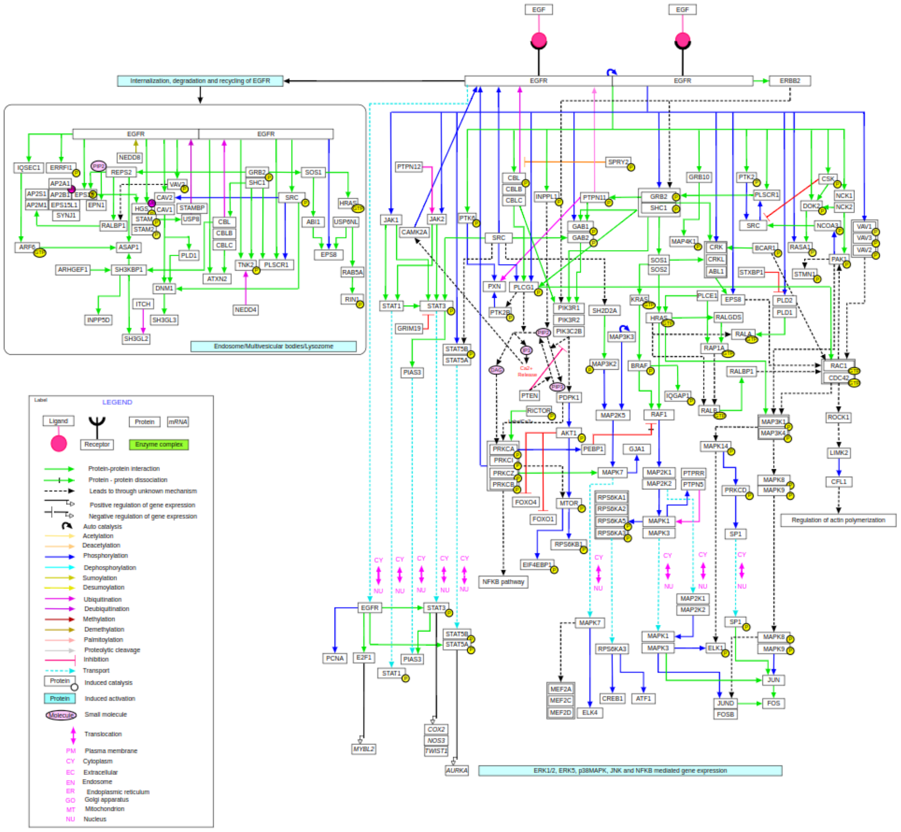


Figure 1.2: Example of a curated biological pathway from the WikiPathways database. This depicts the EGF/EGFR signaling pathway, known to be disregulated in many cancers. Nodes represent proteins (boxes) and small molecules (circles). Edges represent causal dependencies via phosphorylation (blue) or protein-protein interactions (green). Credit Pandey et al. (2023).

Biological pathways

For the purposes of this document, a *biological pathway* is a molecular process that performs some recognizable function in a cell. Biologists have mapped out thousands of pathways that carry out diverse roles, such as (*i*) regulating gene expression; (*ii*) processing energy, nutrients, and waste (i.e., *metabolism*); and (*iii*) communication (*signaling*). Biologists typically depict a pathway as a directed graph, where the vertices are proteins (or other entities), and edges represent dependencies between them. See figure 1.2 for illustration.

Biologists have accumulated increasingly detailed knowledge of pathways through decades of careful experimentation. This knowledge is curated in several publicly available pathway databases, such as Reactome (Gillespie et al., 2022), MSigDB (Liberzon et al., 2015) NCIPID (Schaefer et al., 2009), Wikipathways (Pico et al., 2008), and KEGG (Kanehisa and Goto, 2000). Clearly, pathway databases are a rich form of prior knowledge. They encode hard-won scientific insights gathered by armies of scientists. It seems wasteful to ignore this information when analyzing biological data.

In fact, numerous techniques *do* exist for analyzing biological data in a pathway-informed fashion. Chapters 2 and 3 list many of them. These techniques vary (*i*) in their modeling assumptions and (*ii*) in the way they *represent* pathways. Some representations are quite complicated: hypergraphs (Ritz et al., 2014), process algebras (Regev et al., 2000), or systems of differential equations (Smolen et al., 2000). Others are much simpler: undirected graphs, or even simply *gene sets*. An old (but thorough) review by de Jong (2002) describes many of the computational and mathematical representations used to describe pathways and incorporate them into data analysis.

Despite their detail and breadth of coverage, curated pathways do have important limitations that should be kept in mind—especially when using pathways to analyze data. Given the complicated nature of biological systems, one can be confident that curated pathways are incomplete. Even worse, it's well-understood that pathway databases don't capture the full set of interactions known to biologists (Hanspers et al., 2020). One can also wager that the molecular processes governing cells are highly interconnected and lack well-defined boundaries, implying that the notion of a distinct pathway is not well-defined, strictly speaking.

It is also well-understood that the structure of a pathway can vary between contexts—e.g., between diseased and healthy cells (Hill et al., 2017). This implies that in some situations, curated pathways may match the true molecular processes quite poorly. These issues suggest that simpler pathway representations could be more practical for data analysis, since they would be less sensitive to context dependence. I pursue these questions to a limited degree in Chapters 3 and 4.

1.4 Unique challenges for machine learning on omics data

Machine learning on omics data involves some unique challenges that may be unfamiliar to practitioners from other areas, such as text and images. I sketch out some of them here, to benefit a broader machine learning audience.

Section 1.1 already lists several difficulties posed by the data. There are additional data-related issues, though. Omics data may be high-dimensional, but there aren't necessarily many samples. In fact, the typical dataset has far more features than samples. This results from the relatively high *cost* of omics data—additional samples cost material, time, and labor. Data may also be constrained by the availability of human or animal subjects. Contrast this with the abundance of text and image data available on the internet.

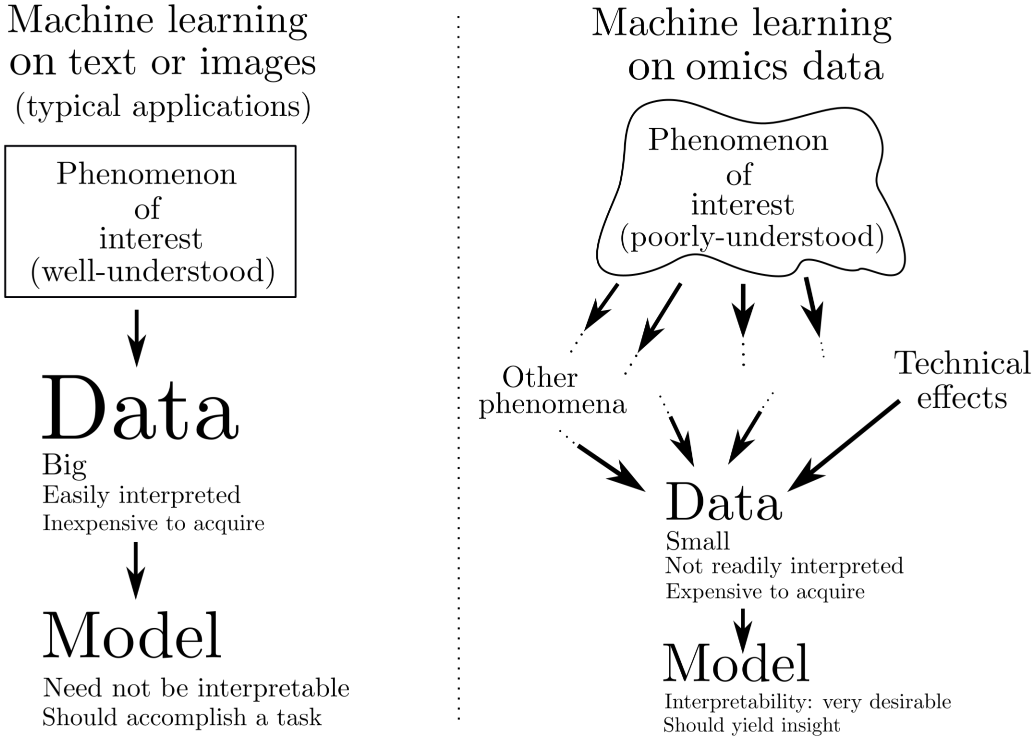


Figure 1.3: A cartoon of some challenges posed by machine learning on omics data. The underlying biological phenomena are only partially understood. Data are less abundant and harder for a human to interpret. The data are less directly connected to the phenomenon of interest, and are confounded by other phenomena. Models are usually designed to yield insight about the phenomenon of interest, rather than merely accomplishing some task (e.g., prediction).

The typical omics dataset is highly context-specific. Biologists usually collect data to shed light on a specific scientific question. This informs their choice of (*i*) omics assay, (*ii*) species and model system (i.e., cells in a dish; mice; humans), and (*iii*) experimental interventions on the samples. This targeted approach to data collection helps the biologist answer their immediate question, but reduces the dataset's utility for other contexts. Furthermore, omics datasets are not annotated in a consistent way that could, hypothetically, allow a model to fully account for experimental context.

Omics data can hardly be interpreted by the human eye—one of the motivations for this dissertation. In contrast, text and images have an inherent interpretability that aids model development. For example, a human can easily look at an image and recognize that it's a cat in grayscale. On the other hand, most bioinformaticians can't look at a sample of RNA-seq data and discern whether it comes from healthy or diseased tissue.

Other difficulties are related to model evaluation. Many scientifically interesting problems cannot be framed as supervised learning tasks. Open-ended biological research does not always involve predicting well-understood labels from omics data. Instead, tasks are frequently *unsupervised*.

At the same time, biologists want models to reveal something *true* about the biology. For example, causal relationships or reliable biomarkers. Of course, this is in a scientific setting where the ground truth is *unknown*. Hence, validating a model requires costly, error-prone experiments. For these reasons, the feedback loop between modeling and experimentation can be very slow in biological applications.

Some of these evaluation difficulties show up in Chapter 2. I use experimental results from a DREAM challenge (Hill et al., 2016) to evaluate a model, but find that the validation is too imprecise to reliably discern between competing models.

Evaluation issues also appear in Chapter 3. The model in that chapter produces a biological interpretation for its outputs. However, it's difficult to judge the accuracy of that interpretation in the absence of experimental validation.

Structure of this dissertation

The background and motivation for this dissertation have been firmly established. The remainder of this document is organized as follows:

- Chapter 2 details Sparse Signaling Pathway Sampling, a probabilistic model that infers the structure of a signaling pathway from phosphoproteomics time series data.

- Chapter 3 describes PathMatFac, a matrix factorization model for multiomic datasets.

- Lastly, Chapter 4 reflects on these projects and offers some parting observations about this field of work.

2 INFERRING THE STRUCTURE OF A SIGNALING PATHWAY FROM PHOSPHOPROTEOMIC TIME SERIES

2.1 Introduction

Signaling pathways enable cells to process information rapidly in response to external environmental changes or intracellular cues. One of the core signaling mechanisms is protein phosphorylation. Kinases add phosphate groups to substrate proteins and phosphatases remove them. These changes in phosphorylation state can act as switches, controlling proteins' activity and function. A protein's phosphorylation status affects its localization, stability, and interaction partners (Newman et al., 2014). Ultimately, phosphorylation changes regulate important biological processes such as transcription and cell growth, death, and differentiation (Hunter, 2009; Kholodenko et al., 2010).

Pathway databases characterize the signaling relationships among groups of proteins but are not tailored to individual biological contexts. Even for well-studied pathways such as epidermal growth factor receptor-mediated signaling, the proteins significantly phosphorylated during a biological response can differ greatly from those in the curated pathway (Köksal et al., 2018). The discrepancy can be due to context-specific signaling (Hill et al., 2017), cell type-specific protein abundances, or signaling rewiring in disease (Pawson and Warner, 2007). Therefore, there is a need to learn context-specific signaling pathway representations from observed phosphorylation changes. In the clinical setting, patient-specific signaling pathway representations may eventually be able to guide therapeutic decisions (Drake et al., 2016; Halasz et al., 2016; Eduati et al., 2020).

Diverse classes of techniques have been developed to model and infer signaling pathways (Kholodenko et al., 2012). They take approaches including Granger causality (Shojaie and Michailidis, 2010; Carlin et al., 2017), information theory (Cheong et al., 2011; Krishnaswamy et al., 2014), logic models (Eker et al., 2002; Guziolowski et al., 2013; Gjerga et al., 2020), differential equations (Schoeberl et al., 2002; Molinelli et al., 2013; Henriques et al., 2017), non-parametric statistical tests (Zhang and Song, 2013), and probabilistic graphical models (Sachs et al., 2005) among others. Some signaling pathway reconstruction algorithms take advantage of perturbations such as receptor stimulation or kinase inhibition. Although perturbing individual pathway members can causally link them to downstream phosphorylation changes, characterizing a complex pathway can require a large number of perturbation experiments. Inferring pathway structure from temporal phosphorylation data presents an attractive alternative. A single time series phosphorylation dataset can reveal important dynamics without perturbing individual pathway members. For instance, a kinase cannot phosphorylate substrates before it is activated.

An alternative approach to pathway reconstruction selects a context-specific subnetwork from a general background network. These algorithms can use phosphorylation data to assign scores to protein nodes in a protein-protein interaction network. They then select edges that connect the high-scoring nodes, generating a subnetwork that may explain how the induced phosphorylation changes arise from the source of stimulation. Extensions accommodate temporal scores on the nodes (Patil et al., 2013; Budak et al., 2015; Köksal et al., 2018; Norman and Cicek, 2019).

Our present work builds on past techniques that formulate signaling pathway inference as a Dynamic Bayesian Network (DBN) structure estimation problem. This family of techniques relies on two core ideas: (*i*) we can use a DBN to model phosphorylation time series data; and (*ii*) the DBN's structure translates directly to a directed graph representing the signaling pathway. Rather than identifying a single DBN that best fits the data, these techniques take a Bayesian approach—they yield a *posterior distribution* over possible DBN structures. Some techniques use Markov

Chain Monte Carlo (MCMC) to sample from the posterior (Werhli and Husmeier, 2007; Gregorczyk, 2010). Others use exact, enumerative inference to compute posterior probabilities (Hill et al., 2012; Oates et al., 2014; Spencer et al., 2015).

We present a new Bayesian DBN-based technique, Sparse Signaling Pathway Sampling (SSPS). It improves on past MCMC methods by using a novel proposal distribution specially tailored for the large, sparse graphs prevalent in biological applications. Furthermore, SSPS makes weaker modeling assumptions than other DBN approaches. As a result, SSPS scales to larger problem sizes and yields superior predictions in comparison to other DBN techniques.

We implement SSPS using the `Gen` probabilistic programming language (PPL). Probabilistic programming is a powerful methodology for building statistical models. It enables the programmer to build models in a legible, modular, reusable fashion. This flexibility was important for prototyping and developing the current form of SSPS and readily supports future improvements or extensions.

2.2 Materials and methods

Model formulation

SSPS makes specific modeling assumptions. We start with the DBN model of Hill et al. (2012), relax some assumptions, and modify it in other ways to be better-suited for MCMC inference.

Preliminary definitions. We first define some notation for clarity's sake. Let G denote a *directed graph* with vertices V and edges $E(G)$. Graph G will represent a signaling pathway, with vertices V corresponding to proteins and edges $E(G)$ indicating their influence relationships. We use $\text{pa}_G(i)$ to denote the *parents* of vertex i in G.

Let X denote our time series data, consisting of $|V|$ variables measured at T timepoints. X is a $T \times |V|$ matrix where the jth column corresponds to the jth variable and the jth graph vertex. As a convenient shorthand, let X_+ denote the *latest* $T-1$ timepoints in X, and let X_- denote the *earliest* $T-1$ timepoints in X. Lastly, define $B_j \equiv X_{-,\mathrm{pa}_G(j)}$. In other words, B_j contains the values of variable j's parents at the $T-1$ earliest timepoints. In general, B_j may also include columns of nonlinear interactions between the parents. We will only include linear terms, unless stated otherwise.

Model derivation. In our setting, we aim to infer G from X. In particular, Bayesian approaches seek a *posterior distribution* $P(G|X)$ over possible graphs. From Bayes's rule we know $P(G|X) \propto P(X|G) \cdot P(G)$. That is, a Bayesian model is fully specified by its choice of *prior distribution* $P(G)$ and *likelihood function* $P(X|G)$.

We derive our model from the one used by Hill et al. (2012). They choose a prior distribution of the form

$$P(G \mid G', \lambda) \propto \exp\left(-\lambda |E(G) \setminus E(G')|\right) \qquad (2.1)$$

parameterized by a *reference graph* G' and *inverse temperature* λ. This prior gives uniform probability to all subgraphs of G' and "penalizes" edges not contained in $E(G')$. λ controls the "importance" given to the reference graph.

Hill et al. choose a Gaussian DBN for their likelihood function. Intuitively, they assume linear relationships between variables and their parents:

$$X_{+,j} \sim \mathcal{N}(B_j \beta_j, \sigma_j^2) \qquad \forall j \in \{1 \ldots |V|\}.$$

A suitable prior over the regression coefficients β_j and noise parameters σ_j^2 (Figure 2.1) allows us to integrate them out, yielding this *marginal likelihood* *function*:

$$P(X|G) \propto \prod_{j=1}^{|V|} T^{-\frac{|\mathrm{pa}_G(j)|}{2}} \left(X_{+,j}^{\top} X_{+,j} - \frac{T-1}{T} X_{+,j}^{\top} (B_j \hat{\beta}_{ols}) \right)^{-\frac{T-1}{2}} \tag{2.2}$$

where $\hat{\beta}_{ols} = (B_j^{\top} B_j)^{-1} B_j^{\top} X_{+,j}$ is the ordinary least squares estimate of β_j. For notational simplicity, Equation 2.2 assumes we have a single time course of length T. In general, there may be multiple time course replicates with differing lengths. The marginal likelihood generalizes to that case in a straightforward way.

In SSPS we use the same marginal likelihood function (Equation 2.2), but a different prior distribution $P(G)$. We obtain our prior distribution by decomposing Equation 2.1 into a product of independent Bernoulli trials over graph edges. This decomposition in turn allows us to make some useful generalizations. Define *edge existence variables* $z_{ij} \equiv \mathbb{1}[(i,j) \in E(G)]$. Let Z be the $|V| \times |V|$ matrix of all z_{ij}. Then we can rewrite Equation 2.1 as follows:

$$P(G|G',\lambda) \equiv P(Z|G',\lambda) \propto \prod_{(i,j) \notin E(G')} e^{-z_{ij}\lambda}$$

$$= \prod_{(i,j) \in E(G')} \left(\frac{1}{2}\right)^{z_{ij}} \left(\frac{1}{2}\right)^{1-z_{ij}} \prod_{(i,j) \notin E(G')} \left(\frac{e^{-\lambda}}{1+e^{-\lambda}}\right)^{z_{ij}} \left(\frac{1}{1+e^{-\lambda}}\right)^{1-z_{ij}}$$

where the last line is a true equality—it gives a normalized probability measure. We see that the original prior is simply a product of Bernoulli variables parameterized by a shared inverse temperature, λ. See Appendix 2.4 for a more detailed derivation.

Rewriting the prior in this form opens the door to generalizations. First, we address a shortcoming in the way reference graph G' expresses prior knowledge. The original prior assigns equal probability to every edge of G'. However, in practice we may have differing levels of prior confidence in the edges. We address this by allowing a real-valued prior confidence

c_{ij} for each edge:

$$P(Z|C,\lambda) = \prod_{(i,j)} \left(\frac{e^{-\lambda}}{e^{-c_{ij}\lambda} + e^{-\lambda}} \right)^{z_{ij}} \left(\frac{e^{-c_{ij}\lambda}}{e^{-c_{ij}\lambda} + e^{-\lambda}} \right)^{1-z_{ij}} \tag{2.3}$$

where C is the matrix of all prior confidences c_{ij}, replacing G'. Notice that if every $c_{ij} \in \{0,1\}$, then Equation 2.3 is equivalent to the original prior. In effect, Equation 2.3 *interpolates* the original prior, permitting a continuum of confidences on the interval $[0,1]$.

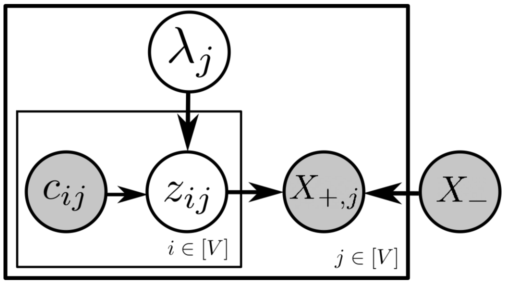
We make one additional change to the prior by replacing the shared λ inverse temperature variable with a collection of variables, $\Lambda = \{\lambda_j \mid j = 1,\ldots,|V|\}$, one for each vertex of the graph. Recall that the original λ variable determined the importance of the reference graph. In the new formulation, each λ_j controls the importance of the prior knowledge for vertex j and its parents:

$$P(Z|C,\Lambda) = \prod_{(i,j)} \left(\frac{e^{-\lambda_j}}{e^{-c_{ij}\lambda_j} + e^{-\lambda_j}} \right)^{z_{ij}} \left(\frac{e^{-c_{ij}\lambda_j}}{e^{-c_{ij}\lambda_j} + e^{-\lambda_j}} \right)^{1-z_{ij}} \tag{2.4}$$

We introduced Λ primarily to help MCMC converge more efficiently. Experiments with the shared λ revealed a multimodal posterior that tended to trap λ in high or low values. The introduction of vertex-specific λ_j variables yielded faster convergence with weaker modeling assumptions—an improvement in both respects.

We implicitly relax the model assumptions further via our inference procedure. For sake of tractability, the original exact method of Hill et al. (2012) imposes a hard constraint on the in-degree of each vertex. In contrast, we use a MCMC inference strategy with no in-degree constraints.

In summary, our model departs from that of Hill et al. (2012) in three important respects. It permits real-valued prior confidences C, introduces vertex-specific inverse temperature variables Λ, and places no constraints



$$\begin{aligned}
\lambda_j &\sim \text{Uniform}(\lambda_{\min}, \lambda_{\max}) && \forall j \in \{1 \ldots |V|\} \\
z_{ij} \mid c_{ij}, \lambda_j &\sim \text{Bernoulli}\left(\frac{e^{-\lambda_j}}{e^{-c_{ij}\lambda_j} + e^{-\lambda_j}}\right) && \forall i, j \in \{1 \ldots |V|\} \\
\sigma_j^2 &\propto \frac{1}{\sigma_j^2} && \forall j \in \{1 \ldots |V|\} \\
\beta_j \mid \sigma_j^2 &\sim \mathcal{N}\left(0, T\sigma_j^2(B_j^\top B_j)^{-1}\right) && \forall j \in \{1 \ldots |V|\} \\
X_{+,j} \mid B_j, \beta_j, \sigma_j^2 &\sim \mathcal{N}\left(B_j\beta_j, \sigma_j^2 I\right) && \forall j \in \{1 \ldots |V|\}
\end{aligned}$$

Figure 2.1: The generative model for SSPS. (top) Plate notation. DBN parameters β_j and σ_j^2 have been marginalized out. (bottom) Full probabilistic specification. We usually set $\lambda_{\min} \simeq 3$ and $\lambda_{\max}$=15. If $\lambda_{\min}$>0 is too small, Markov chains will occasionally be initialized with very large numbers of edges, causing computational issues. The method is insensitive to $\lambda_{\max}$ as long as it's sufficiently large. Notice the improper prior $1/\sigma_j^2$. In this specification B_j denotes $X_{-,\text{pa}_Z(j)}$; that is, the parents of vertex j depend on edge existence variables Z.

on vertices' in-degrees. See the full model in Figure 2.1 and Appendix 2.4 for additional details.

Inference procedure

Our method uses MCMC to infer posterior edge existence probabilities. As described in Section 2.2, our model contains two classes of unobserved random variables: (*i*) the edge existence variables Z and (*ii*) the inverse temperature variables Λ. For each step of MCMC, we loop through these variables and update them in a Metropolis-Hastings fashion.

Main loop. At a high level, our MCMC procedure consists of a loop over the graph vertices, V. For each vertex j, we update its inverse temperature variable λ_j and then update its *parent set* $\text{pa}_G(j)$. All of these updates are Metropolis-Hastings steps; the proposal distributions are described below. Each completion of this loop yields one iteration of the Markov chain.

Proposal distributions. For the inverse temperature variables we use a symmetric Gaussian proposal: $\lambda_j' \sim \mathcal{N}(\lambda_j, \xi^2)$. In practice the method is insensitive to ξ; we typically set ξ=3.

The parent set proposal distribution is more complicated. There are two principles at work when we design a graph proposal distribution: (*i*) the proposal should efficiently traverse the space of directed graphs, and (*ii*) it should favor graphs with higher posterior probability. The most widely used graph proposal distribution selects a *neighboring* graph uniformly from the set of possible "add-edge," "remove-edge," and "reverse-edge" actions (Werhli and Husmeier, 2007; Gregorczyk, 2010). We'll refer to this traditional proposal distribution as the *uniform graph proposal*. In our setting, we expect sparse graphs to be much more probable than dense ones—notice how the marginal likelihood function (Equation 2.2) strongly penalizes $|\text{pa}_G(j)|$. However, the uniform graph proposal exhibits a preference toward *dense graphs*. It proposes "add-edge" actions too often. This motivates us to design a new proposal distribution tailored for sparse graphs—one that operates on our sparse *parent set* graph representation.

For a given graph vertex $j \in V$, the parent set proposal distribution updates $\text{pa}_G(j)$ by choosing from the following actions:

- `add-parent`. Select one of vertex j's non-parents uniformly at random, and add it to $\text{pa}_G(j)$.

- `remove-parent`. Select one of vertex j's parents uniformly at random, and remove it from $\text{pa}_G(j)$.

- `swap-parent`. A simultaneous application of `add-parent` and `remove-parent`. Perhaps surprisingly, this action is not made redundant by the other two. It plays an important role by yielding updates that maintain the size of the parent set. Because the marginal likelihood (Equation 2.2) changes steeply with $|\text{pa}_G(j)|$, Metropolis-Hastings acceptance probabilities will be higher for actions that keep $|\text{pa}_G(j)|$ constant.

These three actions are sufficient to explore the space of directed graphs, but we need another mechanism to bias the exploration toward *sparse* graphs. We introduce this preference via the *probability* assigned to each action. Intuitively, we craft the action probabilities so that when $|\text{pa}_G(j)|$ is too small, `add-parent` moves are most probable. When $|\text{pa}_G(j)|$ is too big, `remove-parent` moves are most probable. When $|\text{pa}_G(j)|$ is about right, all moves are equally probable.

We formulate the action probabilities for vertex j as follows. As a shorthand, let $s_j = |\text{pa}_G(j)|$ and define the *reference size* $\hat{s}_j = \sum_{i=1}^{|V|} c_{ij}$. That is, $\hat{s}_j$ uses the prior edge confidences C to estimate an appropriate reference size for the parent set. Then, the action probabilities are

$$\begin{aligned}
p(\texttt{add-parent}|s_j, \hat{s}_j) &\propto 1 - \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)} \\
p(\texttt{remove-parent}|s_j, \hat{s}_j) &\propto \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)} \\
p(\texttt{swap-parent}|s_j, \hat{s}_j) &\propto 2\left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)} \cdot \left(1 - \left(\frac{s_j}{|V|}\right)^{\gamma(\hat{s}_j)}\right)
\end{aligned}$$

where $\gamma(\hat{s}_j) = 1/\log_2(|V|/\hat{s}_j)$. We use these functional forms only because they have certain useful properties: (*i*) when $s_j{=}0$, the probability of `add-parent` is 1; (*ii*) when $s_j{=}|V|$, the probability of `remove-parent` is 1; and (*iii*) when $s_j{=}\hat{s}_j$, all actions have equal probability (Figure 2.2). Be-

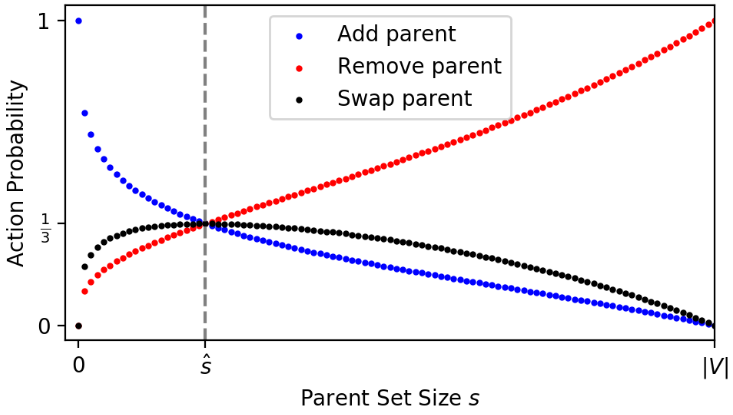


Figure 2.2: Action probabilities as a function of parent set size. The reference size $\hat{s}$ is determined from prior knowledge. It approximates the size of a "typical" parent set. When $s<\hat{s}$, `add-parent` is most probable; when $s>\hat{s}$, `remove-parent` is most probable; and when $s=\hat{s}$, all actions have equal probability.

yond that, these probabilities have no particular justification. We provide additional information about the parent set proposal in Appendix 2.4.

Recall that Metropolis-Hastings requires us to compute the reverse transition probability for any proposal we make. This could pose a challenge given our relatively complicated parent set proposal distribution. However, `Gen` provides a helpful interface for computing reverse probabilities. The user can provide an *involution* function that returns the reverse of a given action. `Gen` then manages the reverse probabilities without further intervention. This makes it relatively easy to implement Metropolis-Hastings updates with unusual proposal distributions.

Termination, convergence, and inference. We follow the basic MCMC protocols described by Gelman et al. (2014). This entails running multiple (i.e., 4) Markov chains and discarding the first half of each chain as burnin. In all of our analyses, we terminate each Markov chain when it either (*i*) reaches a length of 100,000 iterations or (*ii*) the execution time exceeds 12 hours. These termination conditions are arbitrary but emulate a real-world setting where it may be acceptable to let the method run overnight.

Upon termination, we assess convergence with two diagnostics: Potential Scale Reduction Factor (PSRF) and effective number of samples (N_{eff}). PSRF identifies cases where the Markov chains fail to mix or achieve stationarity. N_{eff} provides a sense of "sample size" for our inferred quantities. It adjusts the number of MCMC samples by accounting for autocorrelation in each chain. For our purposes, we say a quantity has *failed to converge* if its PSRF $\geqslant 1.01$ or $\mathsf{N}_{\text{eff}} < 10$. Note that satisfying these criteria does not guarantee convergence. However, failure to satisfy them is a reliable flag for non-convergence.

Assuming a quantity hasn't failed to converge, we estimate it by simply taking its sample mean from all samples remaining after burnin. In our setting we are primarily interested in *edge existence* probabilities; i.e., we compute the fraction of samples containing each edge.

Probabilistic programming implementation

We implemented SSPS using the `Gen` PPL. We briefly describe the probabilistic programming methodology and its advantages in our setting.

Probabilistic programming. Probabilistic programming is a methodology for building statistical models. It's based on the idea that statistical models are *generative processes*—sequences of operations on random variables. In probabilistic programming, we express the generative process as a program written in a PPL. This program is then compiled to produce a log-probability function, which can be used in inference tasks. Probabilistic programming systems typically provide a set of generic inference methods for performing those tasks—e.g., MCMC or Variational Bayes.

Compare this with a more traditional approach, where the user must (*i*) derive and implement the log-probability function and (*ii*) implement an inference method that operates on the log-probability function. This process of manual derivation and implementation is error-prone and requires a high degree of expertise from the user. In contrast, probabilistic programming only requires the user to express their model in a PPL. The probabilistic programming system manages other details.

Probabilistic programming also tends to promote good software engineering principles such as abstraction, modularity, and legibility. Most PPLs organize code into functions, which can be reused by multiple statistical models.

Probabilistic programming languages. Several PPLs have emerged in recent years. Examples include `Stan` (Carpenter et al., 2017), `Edward2` (Dillon et al., 2017), `Pyro` (Bingham et al., 2018), `PyMC3` (Salvatier et al., 2016), and `Gen` (Cusumano-Towner et al., 2019). PPLs differ in how they balance *expressive power* and *ease of use*. For example, `Stan` makes it easy to build hierarchical statistical models with continuous variables but caters poorly to other model classes. At the other extreme, `Gen` can readily express a large class of models—discrete and continuous variables with complex relationships—but requires the user to design a customized inference procedure.

Implementation in `Gen`. We use the `Gen` PPL precisely for its expressive power and customizable inference. While implementing SSPS, the customizability of `Gen` allowed us to begin with simple prototypes and then make successive improvements. For example, our model initially used a dense *adjacency matrix* representation for G, but subsequent optimizations led us to use a sparse *parent set* representation instead. Similarly, our MCMC method started with a naïve "add or remove edge" proposal distribution; we arrived at our sparse proposal distribution (Section 2.2)

PPL	Host language	Primary model class	Primary inference method
`Stan`	custom language	hierarchical, cont's vars	Black-box HMC
`Edward2`	Python/ Tensor-Flow	"deep", cont's vars	Black-box variational
`PyMC3`	Python/ Theano	"deep", cont's vars	Black-box HMC
`Pyro`	Python/ PyTorch	"deep", cont's vars	Black-box variational
`Gen`	Julia	discrete and cont's vars; highly flexible	Customizable MCMC

Table 2.1: A coarse comparison of noteworthy PPLs. `Gen` provides expressiveness but requires the user to implement an inference program for their model. Cont's vars: continuous variables; HMC: Hamiltonian Monte Carlo.

after multiple refinements. Other PPLs do not allow this level of control (Table 2.1).

Simulation study evaluation

We use a simulation study to answer important questions about SSPS: How does its computational expense grow with problem size? Is it able to correctly identify true edges? What is its sensitivity to errors in the prior knowledge? Simulations allow us to answer these questions in a controlled setting where we have access to ground truth.

Data simulation process. We generate each simulated dataset as follows:

1. Sample a random adjacency matrix $A \in \{0,1\}^{|V|\times|V|}$, where each entry is the outcome of a Bernoulli(p) trial. A specifies the *structure* of a DBN. We choose p=5/|V| so that each vertex has an average of 5

Parameter	Meaning	Values
$\|V\|$	Number of variables	40, 100, 200
T	Time course length	8
M	Number of time courses	4
r	Fraction of original edges removed	0.1, 0.5, 0.75, 1.0
a	Fraction of spurious edges added	0.1, 0.5, 0.75, 1.0

Table 2.2: These parameters define the grid of simulated datasets in our simulation study. There are 3×4×4=48 distinct grid points. For each one, we generate K=5 replicates for a total of 240 simulated datasets. The graph corruption parameters, r and a, range from very little error (0.1) to total corruption (1.0).

parents. This approximates the sparsity we might see in signaling pathways. We denote the size of the original edge set as $|E_0|$.

2. Let the weights β for this DBN be drawn from a normal distribution $\mathcal{N}(0, 1/\sqrt{|V|})$. We noticed empirically that the $1/\sqrt{|V|}$ scale prevented the simulated time series from diverging to infinity.

3. Use the DBN defined by A, β to simulate M time courses of length T. We imitate the real datasets in Section 2.2 by generating M=4 time courses, each of length T=8.

4. Corrupt the adjacency matrix A in two steps: (*i*) remove $r \cdot |E_0|$ of the edges from A; (*ii*) add $a \cdot |E_0|$ spurious edges to the adjacency matrix. This corrupted graph simulates the *imperfect prior knowledge* encountered in reality. The parameters r and a control the "false negatives" and "false positives" in the prior knowledge, respectively.

We use a range of values for parameters $|V|, r$, and a, yielding a grid of simulations summarized in Table 2.2. See Appendix 2.4 and Figure 2.6 for additional details.

Performance metrics. We are primarily interested in SSPS's ability to correctly recover the structure of the underlying signaling pathway. The simulation study allows us to measure this in a setting where we have access to ground truth. We treat this as a probabilistic binary classification task, where the method assigns an *existence confidence* to each possible edge. We measure classification performance using area under the precision-recall curve (AUCPR). We use *average precision* to estimate AUCPR, as opposed to the trapezoidal rule (which tends to be overly-optimistic, see Davis and Goadrich (2006); Flach and Kull (2015)).

Our decision to use AUCPR is motivated by the sparseness of the graphs. For sparse graphs the number of edges grows linearly with $|V|$ while the number of possible edges grows quadratically. Hence, as $|V|$ grows, the proportion of positive instances decreases and the classification task increasingly becomes a "needle-in-haystack" scenario.

Performance measurements on simulated data come with many caveats. It's most instructive to think of simulated performance as a sanity check. Since our data simulator closely follows our modeling assumptions, poor performance would suggest serious shortcomings in our method.

HPN-DREAM network inference challenge evaluation

We measure SSPS's performance on experimental data by following the evaluation outlined by the HPN-DREAM Breast Cancer Network Inference Challenge (Hill et al., 2016). Signaling pathways differ across contexts—e.g., cell type and environmental conditions. The challenge is to infer these context-specific signaling pathways from time course data.

Dataset. The HPN-DREAM challenge provides phosphorylation time course data from 32 biological contexts. These contexts arise from exposing 4 cell lines (BT20, BT549, MCF7, UACC812) to 8 stimuli. For each context, there are approximately $M=4$ time courses, each about $T=7$ time points in length. Cell lines have differing numbers of phosphosite measurements (i.e., differing $|V|$), ranging from 39 (MCF7) to 46 (BT20).

Prior knowledge. Participants in the original challenge were free to extract prior knowledge from any existing data sources. As part of their analysis, the challenge organizers combined participants' prior graphs into a set of edge probabilities. These *aggregate priors* summarize the participants' collective knowledge. They were not available to participants in the original challenge, but we use them in our analyses of HPN-DREAM data. We provide them to each of the baseline methods (see Section 2.2), so the resulting performance comparisons are fair. We do not compare any of our scores to those listed by Hill et al. (2016) in the original challenge results.

Performance metrics. The HPN-DREAM challenge aims to score methods by their ability to capture causal relationships between pairs of variables. It estimates this by comparing predicted *descendant sets* against experimentally generated descendant sets. More specifically, the challenge organizers exposed cells to AZD8055, an mTOR inhibitor, and observed the effects on other phosphosites. From this they determined a set of phosphosites *downstream* of mTOR in the signaling pathway. These include direct substrates of the mTOR kinase as well as indirect targets.

Comparing predicted descendants of mTOR against experimentally generated descendants of mTOR gives us a notion of *false positives* and *false negatives*. As we vary a threshold on edge probabilities, the predicted mTOR descendants change, which allows us to make a receiver operating characteristic (ROC) curve. We calculate the resulting area under the ROC curve (AUCROC) with the trapezoidal rule to score methods' performance on the HPN-DREAM challenge. Hill et al. (2016) provide more details for this descendant set AUCROC scoring metric. AUCROC is sensible for this setting since each descendant set contains a large fraction of the variables. Sparsity is not an issue.

Because SSPS is stochastic we run it $K=5$ times per context, yielding 5 AUCROC scores per context. Meanwhile the baseline methods are all deterministic, requiring only one execution per context. We use a simple terminology to compare SSPS's scores against those of other methods. In a given context, we say SSPS *dominates* another method if its *minimum* score exceeds that of the other method. Conversely, we say the other method dominates SSPS if its score exceeds SSPS's *maximum* score. This *dominance* comparison has flaws—e.g., its results depend on the sample size K. However, it errs on the side of strictness and suffices as an aid for summarizing the HPN-DREAM evaluation results.

Baseline pathway inference algorithms

Our evaluations compare SSPS against a diverse set of baseline methods.

Exact DBN (Hill et al., 2012). This method was an early inspiration for SSPS and is most similar to SSPS. However, the exact DBN method encounters unique practical issues when we run it on real or simulated data. The method's computational expense increases rapidly with problem size $|V|$ and becomes intractable unless the "max-indegree" parameter is set to a small value. For example, we found that the method used more than 32GB of RAM on problems of size $|V|=100$, unless max-indegree was set $\leqslant 3$. Furthermore, the exact DBN method only admits prior knowledge in the form of Boolean *reference edges*, rather than continuous-valued edge confidences. We overcame this by using a threshold to map edge confidences to 1 or 0. We chose a threshold of 0.25 for the HPN-DREAM challenge evaluation because it yielded a reasonable number of prior edges. We ran Hill et al.'s implementation using MATLAB 2018a.

`FunChisq` (Zhang and Song, 2013). This method is based on the notion that two variables X, Y have a causal relationship if there exists a *functional dependence* $Y{=}f(X)$ between them. It detects these dependencies using a chi-square test against the "no functional dependence" null hypothesis. `FunChisq` was a strong competitor in the HPN-DREAM challenge, despite the fact that it uses no prior knowledge. In order to use `FunChisq`, one must first discretize their time course data. We followed Zhang and Song's recommendation to use 1D k-means clustering for discretization. Detailed instructions are given in the HPN-DREAM challenge supplementary materials (Hill et al., 2016). We used the `FunChisq` (v2.4.9.1) and `Ckmeans.1d.dp` (v4.3.0) R packages.

LASSO. We included a variant of LASSO regression as a simple baseline. It incorporates prior knowledge into the typical primal formulation:

$$\hat{\beta_j} = \text{argmin}_{\beta} \left\{ \|X_{+,j} - B_j \beta\|_2^2 \; + \; \alpha \sum_{i=1}^{V} e^{-c_{ij}} |\beta_i| \right\}$$

where c_{ij} is the prior confidence (either Boolean or real-valued) for edge (i, j). That is, the method uses *penalty factors* $e^{-c_{ij}}$ to discourage edges with low prior confidence. The method selects LASSO parameters, α, using the Bayesian Information Criterion described by Zou et al. (2007). We use `GLMNet` (Friedman et al., 2010) via the `GLMNet.jl` Julia wrapper (v0.4.2).

Prior knowledge baseline. Our most straightforward baseline simply reports the prior edge probabilities, performing no inference at all. Ideally, a Bayesian method should do no worse than the prior—new time course data should only *improve* our knowledge of the true graph. In reality, this improvement is subject to caveats about data quality and model fit.

$\lvert V\rvert$	N/cpu-hr	N_{eff}/cpu-hr	MB per chain
40	70000	400	500
100	9000	140	1200
200	3000	60	1000

Table 2.3: Computational expense of SSPS as a function of problem size $\lvert V\rvert$. N is the number of iterations completed by a Markov chain. N_{eff} accounts for burnin and autocorrelation in the Markov chains, giving a more accurate sense of the method's progress. The last column gives the approximate memory footprint of each chain. The non-monotonic memory usage is an artifact of the chain termination conditions ($N>100{,}000$ or time >12 hours).

SSPS software availability

We provide the SSPS code, distributed under a MIT license, via GitHub (https://github.com/gitter-lab/ssps) and archive it on Zenodo (https://doi.org/10.5281/zenodo.3939287). It includes a Snakemake workflow (Koster and Rahmann, 2012) for our full evaluation pipeline, enabling the reader to reproduce our results. The code used in this manuscript corresponds to SSPS v0.1.1.

2.3 Results

We describe evaluation results from the simulation study and HPN-DREAM network inference challenge. SSPS competes well against the baselines, with superior scalability to other DBN-based approaches.

Simulation study results

We compare our method to the baselines listed in Section 2.2. We focus especially on the exact DBN method of Hill et al. (2012), as SSPS shares many modeling assumptions with it.

|V|	max indeg	"linear"	"full"
40	4	66s	210s
	5	770s	3900s
	6	6700s	**TIMEOUT**
	7	**OOM**	**OOM**
100	3	250s	520s
	4	**OOM**	**OOM**
200	2	53s	140s
	3	**OOM**	**OOM**

Table 2.4: Computational expense of the exact DBN method of Hill et al. (2012) measured in CPU-seconds, as a function of problem size |V| and various parameter settings. The method imposes an in-degree constraint on each vertex, shown in the "max indeg" column. The columns "linear" and "full" correspond to different *regression modes*, i.e., which interaction terms are included in the DBN's conditional probability distributions. "**OOM**" (Out Of Memory) indicates that the method exceeded a 32GB memory limit. "**TIMEOUT**" indicates that the method failed to complete within 12 hours.

Computational expense. Because SSPS uses MCMC, the user may allow it to run for an arbitrary amount of time. With this in mind, we summarize SSPS's timing with two numbers: (*i*) N/cpu-hr, the number of MCMC samples per CPU-hour; and (*ii*) N_{eff}/cpu-hr, the *effective* number of samples per CPU-hour. We also measure the memory footprint per Markov chain, subject to our termination conditions. We measured these numbers for each simulation in our grid (see Table 2.2).

Table 2.3 reports average values of N/cpu-hr, N_{eff}/cpu-hr, and memory footprint for each problem size. As we expect, N/cpu-hr and N_{eff}/cpu-hr both decrease approximately with the inverse of |V|. In contrast, the non-monotonic memory usage requires more explanation. It results from two causes: (*i*) our termination condition and (*ii*) the sparse data structures we use to store samples. On small problems (|V|=40), the Markov chain terminates at a length of 100,000—well within the 12-hour limit. On larger problems (|V|=100 or 200) the Markov chain terminates at the 12-hour timeout. This accounts for the 500MB gap between small and large problems. The *decrease* in memory usage between |V|=100 and 200 results from our sparse representations for samples. Roughly speaking, the sparse format only stores *changes* in the variables. So the memory consumption of a Markov chain depends not only on |V|, but also on the *acceptance rate* of the Metropolis-Hastings proposals. The acceptance rate is smaller for |V|=200, yielding a net decrease in memory usage.

Recall that SSPS differs from more traditional MCMC approaches by nature of its parent set proposal distribution, which is specially designed for sparse graphs (see Section 2.2). When we modify SSPS to instead use a naïve uniform graph proposal, we see a striking difference in sampling efficiency. The uniform graph proposal distribution attains N_{eff}/cpu-hr of 100, 10, and 0.2 for |V|=40, 100, and 200, respectively—drastically smaller than those listed in Table 2.3 for the parent set proposal. It's possible that the traditional proposal could achieve higher N_{eff}/cpu-hr by simply running faster. However, the more important consideration is how N_{eff}/cpu-hr changes with |V|. Our parent set proposal distribution's N_{eff}/cpu-hr decays approximately like $O(1/|V|)$. This is better than what we might expect from a simple analysis (Appendix 2.4). Meanwhile, the traditional proposal distribution's N_{eff}/cpu-hr decays faster than $O(1/|V|^4)$. This gap between $O(1/|V|)$ and $O(1/|V|^4)$ sampling efficiencies makes an enormous difference on large problems.

Table 2.4 summarizes the computational expense of the exact DBN method (Hill et al., 2012). The method quickly becomes impractical as the problem size grows, unless we enforce increasingly strict in-degree restrictions. In particular, the exact DBN method's memory cost grows exponentially with its "max in-degree" parameter. The growth becomes increasingly sharp with problem size. When |V|=200, increasing the maximum in-degree from 2 to 3 makes the difference between terminating in

<1 minute and exceeding 32GB of memory. Such low bounds on in-degree are unrealistic, and will likely result in poor inference quality. In comparison, SSPS makes no constraints on in-degree, and its memory usage scales well with problem size.

The other baseline methods—`FunChisq` and LASSO—are much less computationally expensive. Both finish in seconds and require less than 100MB of memory for each simulated task. This highlights the computationally intense nature of Bayesian approaches. Not every scenario calls for Bayesian inference. However, Bayesian inference is valuable in scientific settings where we're concerned with uncertainty quantification.

Predictive performance. The simulation study provides a setting where we have access to "ground truth"—the true simulated graph. We use AUCPR to score each method's ability to recover the true graph's edges.

Figure 2.3 shows the AUCPR scores for our grid of simulations. Each heat map shows AUCPR as a function of graph corruption parameters, $\mathtt{r}$ and $\mathtt{a}$. The heat maps are arranged by method and problem size $|V|$. Each AUCPR value is an average over 5 replicates. SSPS maintains fairly consistent performance across problem sizes. In contrast, the other methods' scores decrease with problem size. For the exact DBN method, this is partially due to the small in-degree constraints imposed on the large problems. It is forced to trade model accuracy for tractability.

Figure 2.4 reveals further insights into these results. It plots *differential* performance with respect to the prior knowledge, in a layout analogous to Figure 2.3. Specifically, it plots the t-statistic of each method's AUCPR, paired with the prior baseline's AUCPR. Whenever the prior graph has some informative edges, SSPS outperforms the prior. On the other hand, SSPS's performance deteriorates whenever the prior contains *no* true edges (i.e., $\mathtt{r}=1$). Under those circumstances `FunChisq` may be a better choice. Since it doesn't rely on prior knowledge at all, it outperforms the other methods when the prior is totally corrupted. However, we expect that in

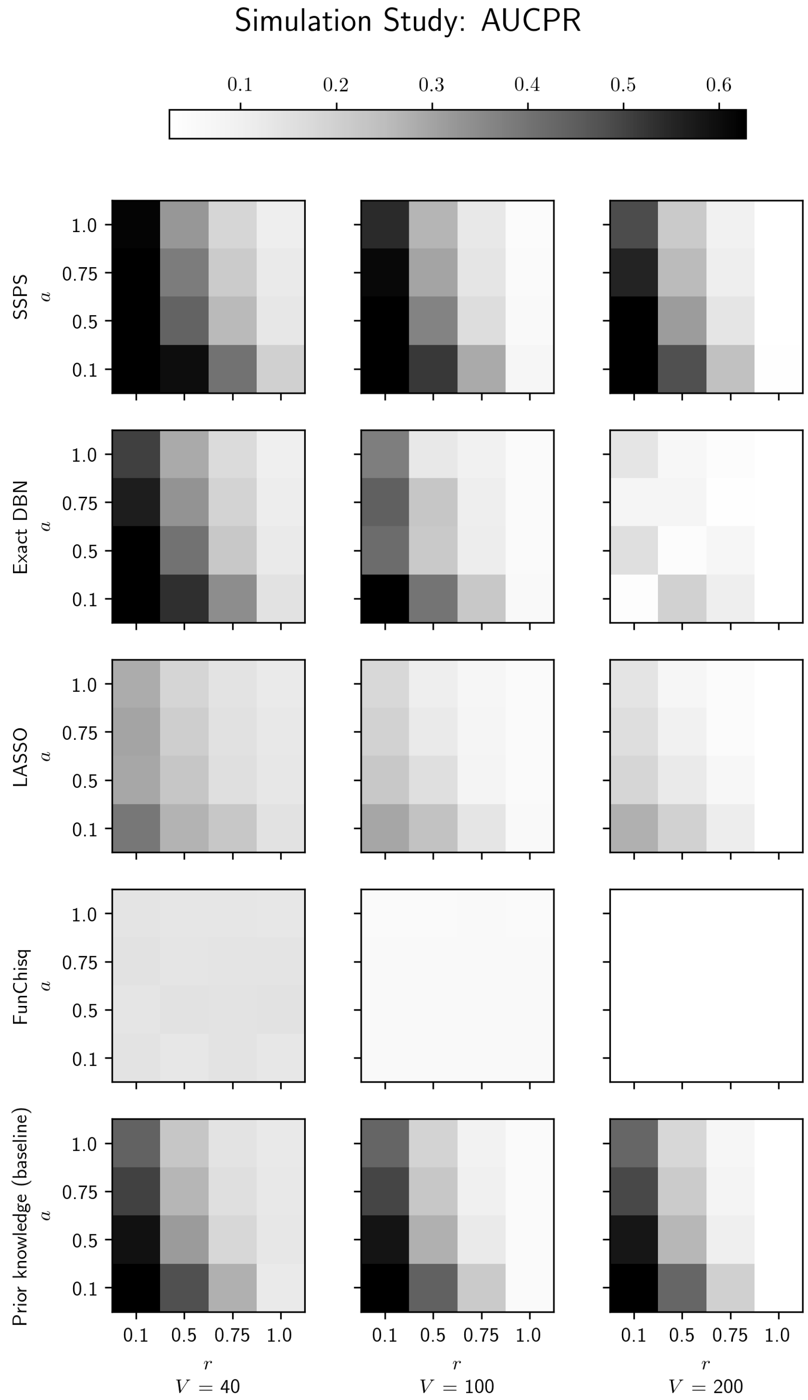


Figure 2.3: Heatmap of AUCPR values from the simulation study. Both DBN-based techniques (SSPS and the exact method) score well on this, since the data is generated by a DBN. On large problems the exact DBN method needs strict in-degree constraints, leading to poor prediction quality. LASSO and `FunChisq` both perform relatively weakly. See Figure 2.7 for representative ROC and PR curves.

most realistic settings there exists partially-accurate prior knowledge, in which case we expect FunChisq to perform worse than SSPS.

These results confirm SSPS's ability to identify the true network, given partially-accurate prior knowledge and time series data consistent with the modeling assumptions. SSPS is fairly robust with respect to the prior's quality and has consistent performance across different problem sizes.

HPN-DREAM challenge results

We evaluated SSPS on experimental data from the HPN-DREAM challenge. The challenge includes time series phosphorylation data from 32 biological contexts: 8 stimuli applied to 4 breast cancer cell lines. Methods are scored on their ability to correctly identify the experimentally derived descendants of mTOR. Figure 2.5 shows bar charts comparing the methods' AUCROC scores in each context. Appendix 2.4 provides additional details.

SSPS performs satisfactorily on this task overall. Employing terminology from Section 2.2, SSPS dominates the exact DBN method in 18 of the 32 contexts, whereas the exact DBN method dominates SSPS in only 9 contexts. Meanwhile, SSPS dominates FunChisq in 11 contexts and is dominated by FunChisq in 15. This is not surprising because FunChisq was among the top competitors in the original challenge. LASSO, on the other hand, performs poorly. SSPS dominates LASSO in 18 contexts and is dominated in only 6.

More puzzling is the strong performance of the prior knowledge baseline. SSPS dominates the aggregate prior in only 9 contexts and is dominated in 21. This is not isolated to our method. FunChisq outperforms and is outperformed by the prior knowledge in 11 and 21 contexts, respectively. The aggregate prior's strong performance is consistent with the results from the original HPN-DREAM challenge; this prior outperformed all individual challenge submissions (Hill et al., 2016). Even though the aggregate prior gives identical predictions for each context and totally ig-

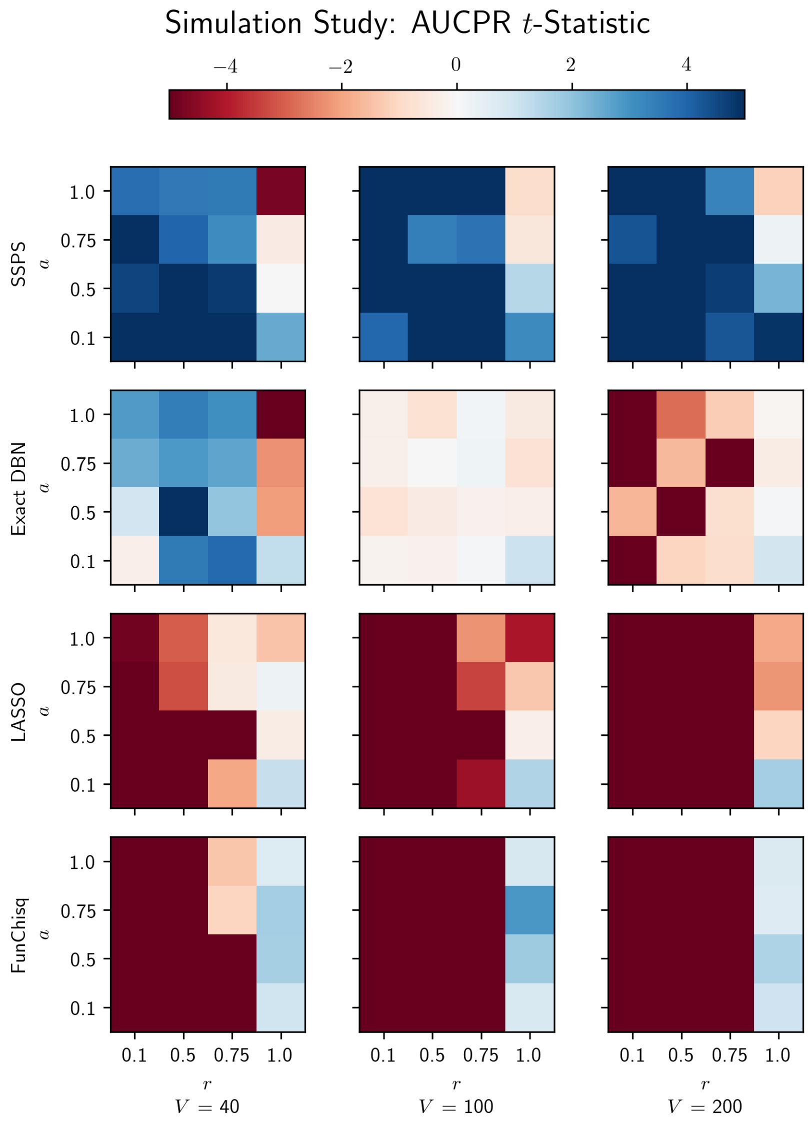


Figure 2.4: Heatmap of differential performance against the prior knowledge, measured by AUCPR paired t-statistics. SSPS consistently outperforms the prior knowledge across problem sizes and shows robustness to errors in the prior knowledge.

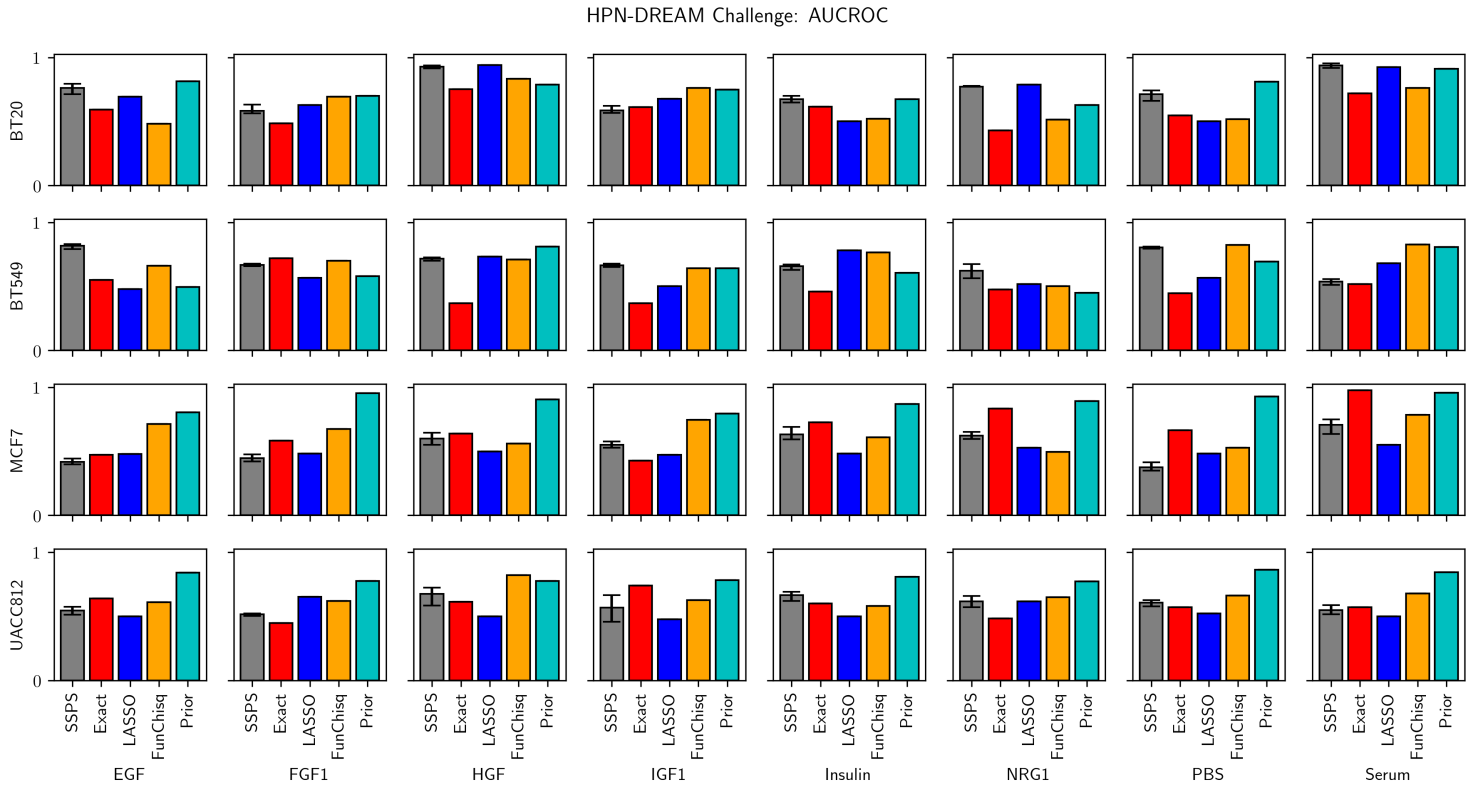


Figure 2.5: Methods' performances across contexts in the HPN-DREAM Challenge. MCMC is stochastic, so we run SSPS 5 times; the error bars show the range of AUCROC scores. The other methods are all deterministic and require no error bars. See Figure 2.8 for example predicted networks, Figure 2.9 for AUCPR scores, and Figure 2.10 for representative ROC and PR curves.

nores the time course data, it still attains better performance than the other methods. This suggests either (i) the data is relatively uninformative or (ii) the evaluation metric based on mTOR's descendants isn't sufficiently precise to measure context-specific performance. We suspect the latter, because `FunChisq` uses no prior knowledge but was the top performer in the HPN-DREAM challenge's *in silico* tasks. An evaluation based on one node's descendants is not as discriminative as an evaluation of the directed edges. Many different directed graphs can have equivalent or similar mTOR descendants. However, it is experimentally impractical to generate the context-specific gold standard networks that would be required for a more precise edge-based evaluation.

2.4 Discussion

We presented SSPS, a signaling pathway reconstruction technique based on DBN structure estimation. It uses MCMC to estimate the posterior probabilities of directed edges, employing a parent set proposal distribution specially designed for sparse graphs. SSPS is a Bayesian approach. It takes advantage of prior knowledge with edge-specific confidence scores and can provide uncertainty estimates on the predicted pathway relationships, which are valuable for prioritizing experimental validation.

SSPS scales to large problems more efficiently than past DBN-based techniques. On simulated data, SSPS yields superior edge predictions with robustness to flaws in the prior knowledge. Our HPN-DREAM evaluation shows SSPS performs comparably to established techniques on a community standard task. It is difficult to make stronger statements in the HPN-DREAM setting because the prior knowledge baseline performs so well and we can only evaluate the predicted mTOR descendants, not the entire pathway. However, SSPS's scalability among Bayesian methods, strong results in the simulation, and competitive performance in the HPN-

DREAM challenge make it an attractive option for further investigation of real phosphorylation datasets.

There are several potential limitations of SSPS relative to alternative pathway signaling models. Prior knowledge is not available in some organisms or biological conditions, reducing one advantage of our Bayesian approach. Although SSPS is more scalable than related DBN techniques, it would struggle to scale to proteome-wide phosphoproteomic data measuring thousands of phosphosites. For large datasets, we recommend running SSPS on a pruned version that includes only the highest intensity or most variable phosphosites. SSPS, like most DBN techniques, models only observed variables. It will erroneously exclude important pathway members, such as scaffold proteins, that are not phosphorylated. Latent variable models or background network-based algorithms are better suited for including unphosphorylated proteins in the pathway. Background network methods can also impose global constraints on the predicted pathway structure, such as controlling the number of connected components or proteins' reachability from relevant receptors (Köksal et al., 2018).

There are many possible ways to improve SSPS. For example, it could be extended to jointly model related pathways in a hierarchical fashion, similar to Oates et al. (2014) and Hill et al. (2017). Alternatively, SSPS could be modified to accommodate causal assumptions via Pearl's intervention operators; see the model of Spencer et al. (2015) for a relevant example. Combining temporal and interventional data (Cardner et al., 2019) is another rich area for future work. On the algorithmic side, we could improve our MCMC procedure by adaptively tuning the parameters of its proposal distributions, as described by Gelman et al. (2014). Because SSPS is a probabilistic program, it is naturally extensible.

Appendix

Model formulation details

We provide additional information about our graph prior and marginal likelihood function. We also describe some implications of SSPS's model assumptions.

Derivation of graph prior (Equation 2.4). We step through a more detailed derivation of SSPS's new graph prior. We begin with the original graph prior (Equation 2.1) and rewrite it in terms of the edge existence variables Z:

$$\begin{aligned}
P(G|G',\lambda) &\propto \exp\left(-\lambda|E(G) \setminus E(G')|\right) \\
&= \exp\left(-\lambda \sum_{(i,j)\notin E(G')} z_{ij}\right) \qquad (2.5) \\
&= \prod_{(i,j)\notin E(G')} e^{-\lambda z_{ij}} \\
&= \prod_{(i,j)\notin E(G')} \left(e^{-\lambda}\right)^{z_{ij}} \\
&\propto \left(\frac{1}{1+e^{-\lambda}}\right)^{V^2-|E(G')|} \cdot \prod_{(i,j)\notin E(G')} \left(e^{-\lambda}\right)^{z_{ij}} \\
&= \prod_{(i,j)\notin E(G')} \left(\frac{1}{1+e^{-\lambda}}\right) \left(e^{-\lambda}\right)^{z_{ij}} \\
&= \prod_{(i,j)\notin E(G')} \left(\frac{1}{1+e^{-\lambda}}\right)^{1-z_{ij}} \left(\frac{e^{-\lambda}}{1+e^{-\lambda}}\right)^{z_{ij}} \qquad (2.6)
\end{aligned}$$

Equation 2.6 shows the original prior is in fact a product of independent Bernoulli variables—the edge existence variables z_{ij}. Equation 2.6 explicitly assigns probability to the edges *not* contained in $E(G')$. However, it also implicitly assigns uniform probability to every edge *contained* in $E(G')$. We deduce that they are Bernoulli(0.5) variables, allowing us to write the prior $P(Z \mid G', \lambda)$ in the following form:

$$\prod_{(i,j) \in E(G')} \left(\frac{1}{2}\right)^{z_{ij}} \left(\frac{1}{2}\right)^{1-z_{ij}} \prod_{(i,j) \notin E(G')} \left(\frac{e^{-\lambda}}{1+e^{-\lambda}}\right)^{z_{ij}} \left(\frac{1}{1+e^{-\lambda}}\right)^{1-z_{ij}} \tag{2.7}$$

just as shown in Section 2.2.

Now we modify the prior to use continuous-valued edge confidences c_{ij} instead of Boolean reference edges $E(G')$. Intuitively, we want to restate Equation 2.7 as a single product over all Z variables, rather than two separate products. Our goal is to find a function $q(c_{ij})$ such that

$$P(Z \mid C, \lambda) = \prod_{(i,j)} q(c_{ij})^{z_{ij}} (1 - q(c_{ij}))^{1-z_{ij}}.$$

However, in order to remain consistent with the original prior $q(c_{ij})$ ought to be monotone-increasing and satisfy these criteria:

$$q(0) = e^{-\lambda}/(1 + e^{-\lambda}) \qquad \text{and} \qquad q(1) = 1/2.$$

It turns out that choosing

$$q(c_{ij}) = \frac{e^{-\lambda}}{e^{-c_{ij}\lambda} + e^{-\lambda}}$$

satisfies these requirements. This brings us to Equation 2.3 of Section 2.2.

From there, it is straightforward to replace the single shared λ variable with a set of vertex-specific Λ variables and arrive at Equation 2.4.

Marginal likelihood function details. Equation 2.2 is obtained by (*i*) using a Gaussian DBN as the likelihood function for G, (*ii*) assuming certain prior distributions for the DBN parameters, and (*iii*) integrating the DBN parameters out. Specifically, let β_j and σ_j^2 $\forall j \in \{1 \ldots |V|\}$ be the DBN's weight and noise parameters, respectively. We assume an improper prior $\sigma_j^2 \propto 1/\sigma_j^2$ for the noise and a Gaussian prior for the weights:

$$\beta_j|\sigma_j^2 \sim \mathcal{N}\left(0, T\sigma_j^2(B_j^\top B_j)^{-1}\right).$$

In other words, SSPS uses an improper joint prior $P(\beta_j, \sigma_j^2) = P(\beta_j|\sigma_j^2)P(\sigma_j^2)$ with $P(\sigma_j^2) \propto 1/\sigma_j^2$. This choice allows β_j and σ_j^2 to be marginalized, yielding Equation 2.2.

The power $-|\mathrm{pa}_G(j)|/2$ in Equation 2.2 is correct when the DBN only uses linear terms. Recall that B_j may in general contain columns of nonlinear interactions between parent variables. When that is true, the quantity $|\mathrm{pa}_G(j)|$ should be replaced by the *width of* B_j. We elide this detail in Section 2.2 for brevity. Our implementation uses the correct exponent.

Our implementation of the marginal likelihood function employs least recently used caching to reduce redundant computation. Code profiling shows that this yields a substantial improvement to efficiency. For additional in-depth discussion of Equation 2.2, we recommend the supplementary materials of Hill et al. (2012).

Additional insights about SSPS's model assumptions. SSPS's model has interesting properties that could lead to method improvements. For example, when we replace the shared λ variable with vertex-specific Λ variables, the model effectively becomes a set of $|V|$ independent models. The plate notation in Figure 2.1 makes this clear; X_- is the only shared variable, and it's fully observed. This has algorithmic implications. For example, future versions of SSPS could parallelize inference at the vertex level, allocating more resources to the parent sets that converge slowly.

In the course of deriving Equation 2.6, we showed that our prior is a log-linear model over edge features. Equation 2.5 shows this most clearly.

Future versions of SSPS could use the expressiveness of log-linear densities over higher-order graph features to capture richer forms of prior knowledge.

Parent set proposal details

A key component of SSPS is its novel *parent set proposal distribution*. We motivate its design and discuss its computational complexity in greater detail.

Parent sets instead of edges. The marginal likelihood (Equation 2.2) is a function of the graph G. However, it depends on G only via its *parent sets*, which are encoded in the matrices B_j. Accordingly, SSPS represents G by storing a list of parents for each vertex.

It makes sense to use a proposal distribution that operates directly on SSPS's internal parent set representation. This motivates our choice of the `add-parent`, `remove-parent`, and `swap-parent` proposals listed in Section 2.2. There is a natural correspondence between (*i*) likelihood function, (*ii*) data structure, and (*iii*) proposal distribution.

Sampling efficiency. We provide some intuition for the parent set proposal's superior sampling efficiency. Let z_{ij} be a particular edge existence variable. The estimate for z_{ij} converges quickly if MCMC updates z_{ij} frequently. Hence, as a proxy for sampling efficiency, consider the number of times z_{ij} gets updated per unit time. We decompose this quantity into three factors:

$$\frac{z_{ij} \text{ updates}}{\text{unit time}} = \epsilon \cdot \tau \cdot \alpha$$

where

$$\epsilon = \frac{\text{graph proposals}}{\text{unit time}} \qquad \tau = \frac{z_{ij} \text{ proposals}}{\text{graph proposal}}$$

$$\alpha = z_{ij} \text{ acceptance probability}$$

In other words, ϵ is the time efficiency of the proposal distribution. The factor τ is the probability that a given proposal *touches* z_{ij}. Lastly, α is the proposal's Metropolis-Hastings acceptance probability.

For a given proposal distribution, we're interested in how these factors depend on $|V|$. For simplicity of analysis, assume the Markov chain is in a typical state where the graph is sparse: $|E(G)| = O(|V|)$.

For the parent set proposal, execution time has no dependence on $|V|$ and hence $\epsilon = O(1)$. Recall that the parent set proposal resides in an outer loop, which iterates through all $|V|$ vertices. It follows that for any particular proposal there is a $1/|V|$ chance that it acts on vertex j. After choosing vertex j, there is on average a $O(1/|V|)$ chance that the proposal affects z_{ij}. This follows from the sparsity of the graph: vertex i is typically a non-parent of j and the probability of choosing it via an `add-parent` or `swap-parent` action is $O(1/|V|)$. Hence, the parent set proposal has a probability τ=$O(1/|V|^2)$ of choosing z_{ij}. Lastly, the acceptance probability α has no dependence on $|V|$ and therefore $\alpha = O(1)$. The product of these factors gives an overall sampling efficiency of $O(1/|V|^2)$ for the parent set proposal.

For the uniform graph proposal, ϵ's complexity depends on the particular implementation. For sake of generosity we assume an efficient implementation with $\epsilon = O(1)$. The proposal chooses uniformly from $O(|V|^2)$ actions: add-, remove-, or reverse-edge. The probability of choosing one that affects z_{ij} is $\tau = O(1/|V|^2)$. Recall that the marginal likelihood decreases steeply with parent set size. It follows that add-edge actions will typically have low acceptance probability. Since the graph is sparse, add-edge actions are overwhelmingly probable; the probability of *not* landing on one is $O(1/|V|^2)$. If we assume the acceptance probability is high for remove-edge and reverse-edge actions, (i.e., they are accepted whenever they're proposed), then this suggests $\alpha = O(1/|V|^2)$, averaged over many proposals. The product of these factors suggests a sampling efficiency that decays like $O(1/|V|^4)$.

This gap between $O(1/|V|^2)$ and $O(1/|V|^4)$ sampling efficiencies explains most of the difference that we saw in Section 2.3. A more detailed analysis may reveal why the parent set proposal attains sampling efficiencies closer to $O(1/|V|)$ in practice.

Simulation study details

We give additional details about the simulation study's methodology and results.

Simulation process. The simulation process described in Section 2.2 differs from SSPS's modeling assumptions in several ways. Recall that the simulator constructs a DBN to generate time series data. This simulated DBN employs nonlinear interaction terms. The simulator assumes that the data at each timepoint is a *cubic* function of the data at the previous timestep. In contrast, all of our analyses ran SSPS with an assumption of *linear* dependencies. In other words, the data contained complexities that SSPS was unable to capture. SSPS's performance in the simulation study suggests that it has some robustness to modeling assumption mismatches.

We provide an illustration of the simulation process in Figure 2.6. It is interesting to notice that the simulated networks do not resemble directed acyclic graphs (DAGs) in any way. They do not have any sense of directionality. Contrast this with the biological graphs shown in Figure 2.8. Strictly speaking these are not DAGs, but they do have an overall direction. Some vertices are source-like, and others are sink-like. Future simulations and models could be more biologically realistic if they incorporated this kind of structure.

Simulation study results. Figure 2.7 gives some representative ROC and PR curves from the simulation study. On problem sizes up to $|V| = 100$, SSPS and the exact DBN method yield similar curves in both ROC and

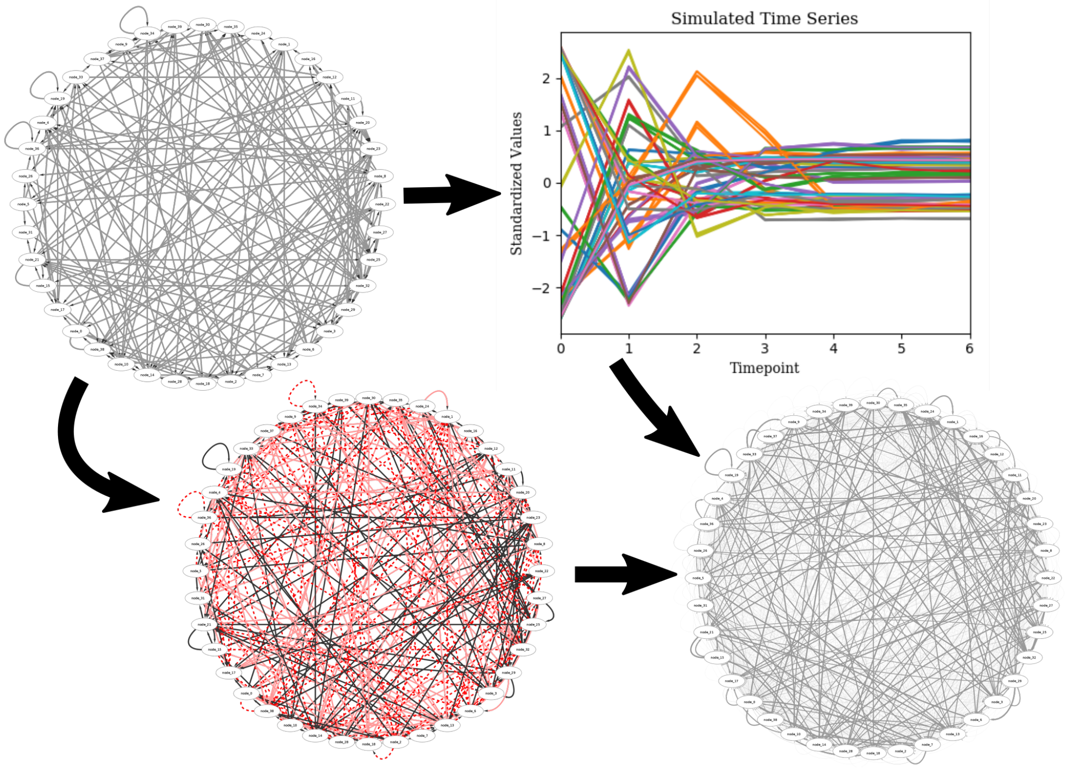


Figure 2.6: A schematic of the simulation study. We randomly generate a true network (upper left) and use it to simulate a time series dataset (upper right). We corrupt the true network by adding and removing edges (lower left); solid red edges have been added, dashed red edges have been removed, and black edges are original. This corrupted network serves as partially inaccurate prior knowledge for the inference techniques. Each technique produces a predicted network (lower right) by assigning a score to each possible edge. The predicted network is evaluated with respect to the true network.

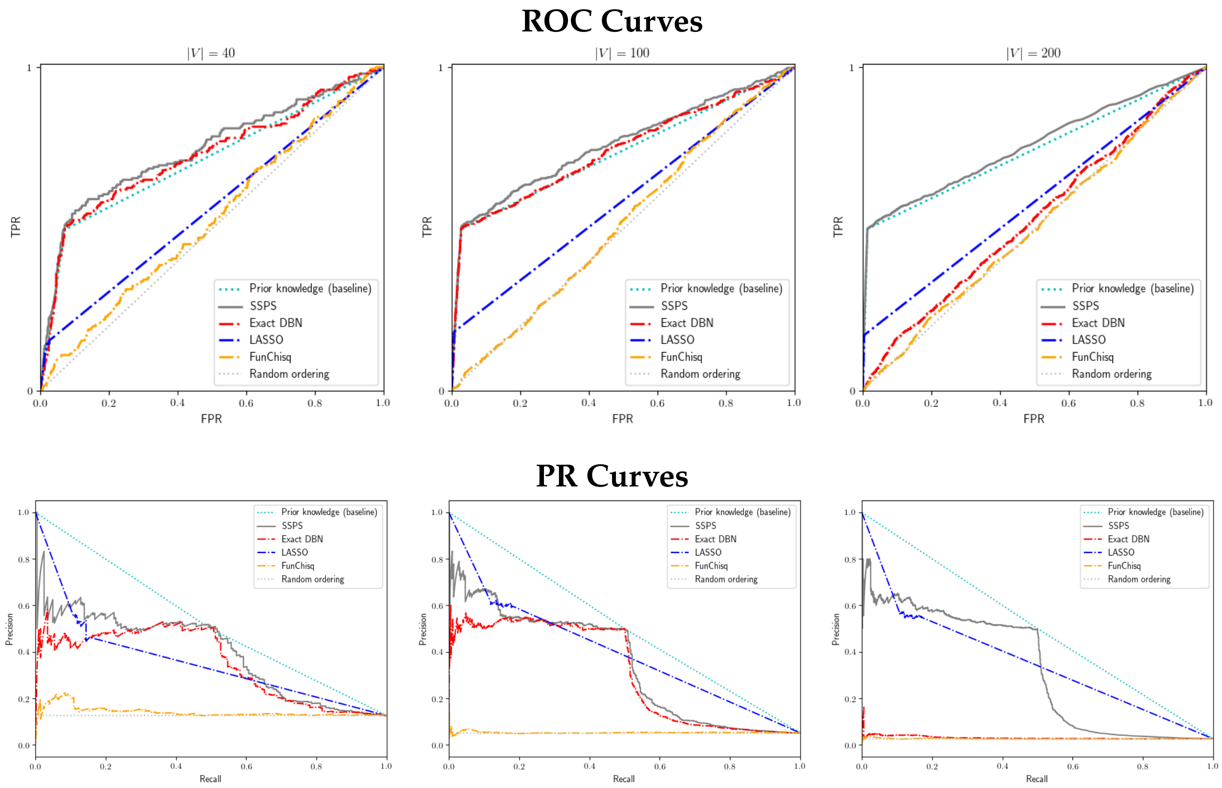


Figure 2.7: Representative ROC curves (top) and PR curves (bottom) from the simulation study. We show curves for three different simulations: $|V| = 40, 100$, and 200 (left, middle, right respectively). Each of these simulations used corruption parameters $r = a = 0.5$.

PR space—though SSPS's curves clearly dominate. On larger problems the exact DBN method's performance quickly deteriorates. Computational tractability requires the exact method to impose highly restrictive in-degree constraints. These observations are consistent with the heatmaps of Figures 2.3 and 2.4 in Section 2.3.

HPN-DREAM challenge details

We provide additional details for the methodology and results of the HPN-DREAM challenge evaluation.

Data preprocessing. The HPN-DREAM challenge data needed to be preprocessed before it could be used by the inference methods. The choices we made during preprocessing most likely affected the inference results.

Many of the time series contain duplicate measurements. We managed this by simply averaging the duplicates. We log-transformed the time series since they were strictly positive and some methods (SSPS and exact DBN) assume normality. This probably made little difference for `FunChisq`, which discretizes the data as part of its own preprocessing.

Predicted networks. Figure 2.8 visualizes networks from two biological contexts in the HPN-DREAM challenge evaluation. This gives a sense of how the different inference methods' predictions differ from each other. All of the predicted networks are fairly different, though the SSPS and exact DBN predictions are more similar to each other than they are to `FunChisq`. `FunChisq` predicts more self-edges than the other methods.

In the BT549 cell line, the experimentally detected mTOR descendants include receptor proteins that would traditionally be considered upstream of mTOR in the pathway. The experimental results are reasonable due to the influence of feedback loops in signaling pathways. However, the number and positioning of the mTOR descendants highlights the differences between the coarse HPN-DREAM challenge evaluation, which is based

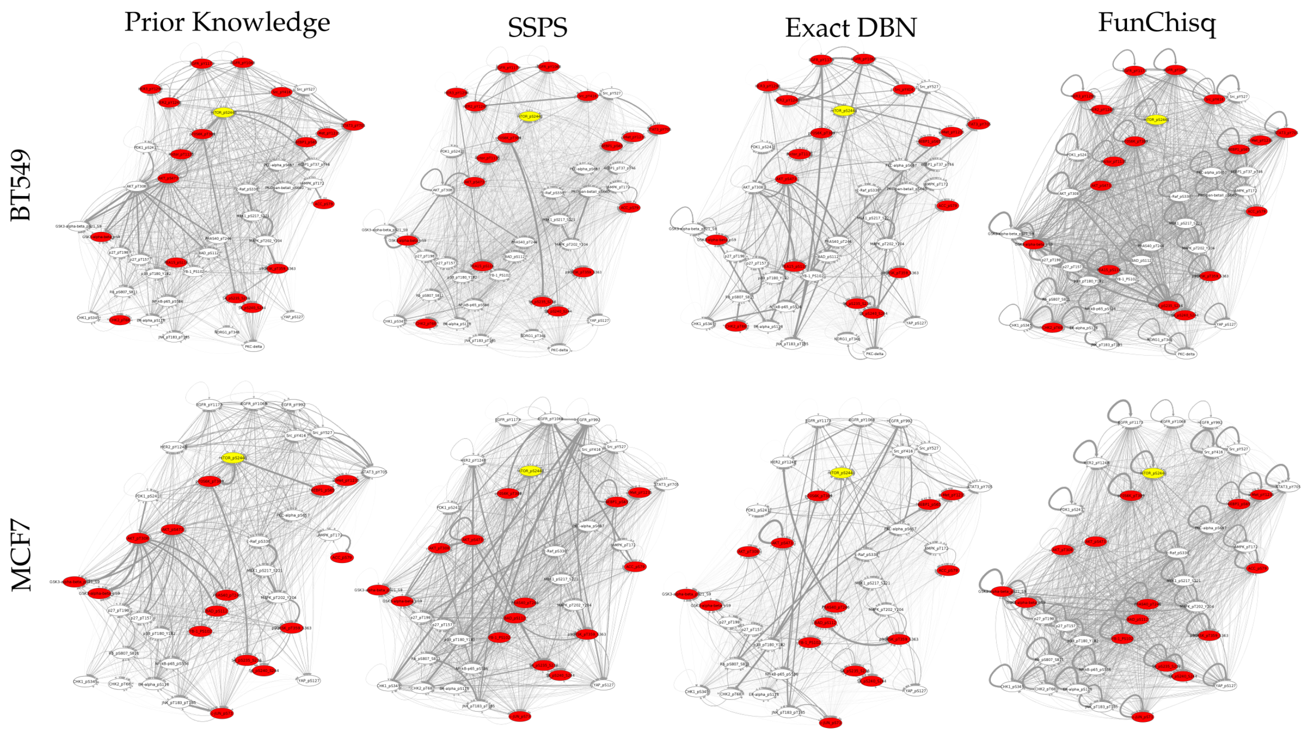


Figure 2.8: Prior and predicted pathways from the HPN-DREAM challenge. We show pathways from two contexts: cell lines BT549 (top row) and MCF7 (bottom row). The stimulus is EGF for both contexts. SSPS attained the best AUCROC of all methods in the (BT549, EGF) context and the worst in the (MCF7, EGF) context. The yellow node is mTOR; red nodes are the experimentally generated ("ground truth") descendants of mTOR.

on reachability in a directed graph, and the more precise evaluation in our simulation study, where we have the edges in the ground truth network.

HPN-DREAM AUCPR. For completeness, we complement the AUCROC results of Section 2.3 with the corresponding AUCPR results. Figure 2.9 shows AUCPR in bar charts, with an identical layout to Figure 2.5.

AUCPR leads us to similar conclusions as those from AUCROC. SSPS dominates the exact DBN method in 19 contexts and is dominated in 10. Both SSPS and `FunChisq` dominate each other in 14 contexts. However, SSPS dominates the prior knowledge in only 9 contexts, and is dominated in 21. As before, we conclude that SSPS attains similar performance to established methods on this task.

ROC and PR curves. Figure 2.10 shows ROC and PR curves from our HPN-DREAM evaluation. We focus on two representative contexts: cell lines BT549 and MCF7, with EGF as the stimulus.

The bar charts in Figure 2.9 tell us that SSPS was the top performer in the (BT549, EGF) context. The ROC and PR curves are consistent with this. SSPS dominates the other methods in ROC and PR space. In contrast, SSPS was the worst performer in the (MCF7, EGF) context. The curves show SSPS performing worse than random.

The LASSO ROC and PR curves are interesting. Its ROC curves show nearly random performance. Its PR curves are straight lines. Manually inspecting its predictions yields an explanation: (*i*) LASSO gives nonzero probability to a very small number of edges; (*ii*) that small set of edges results in a very small descendant set for mTOR; (*iii*) that small descendant set is incorrect.

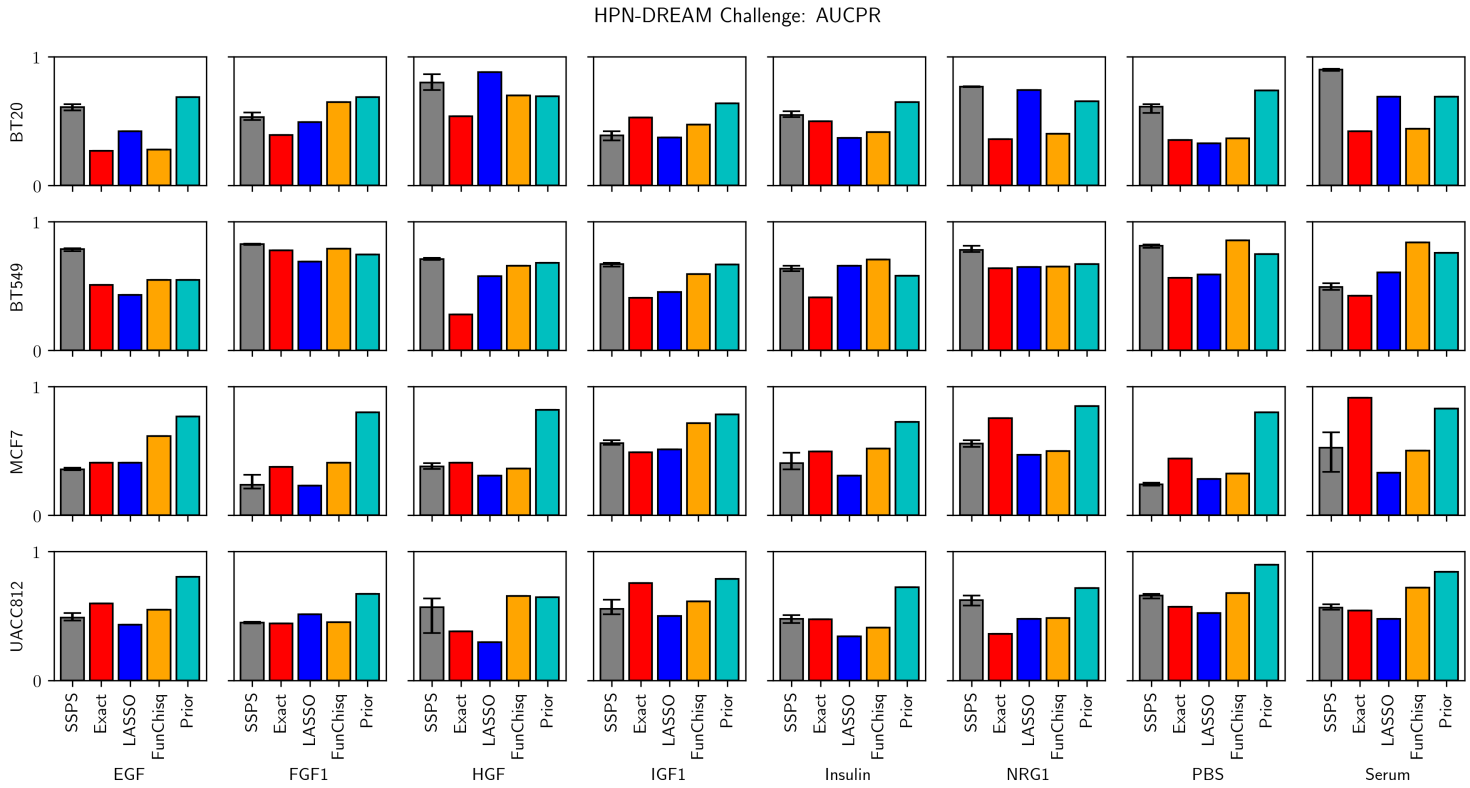


Figure 2.9: A bar chart similar to Figure 2.5 except that it shows AUCPR rather than AUCROC. See Figure 2.5 for details about the layout.

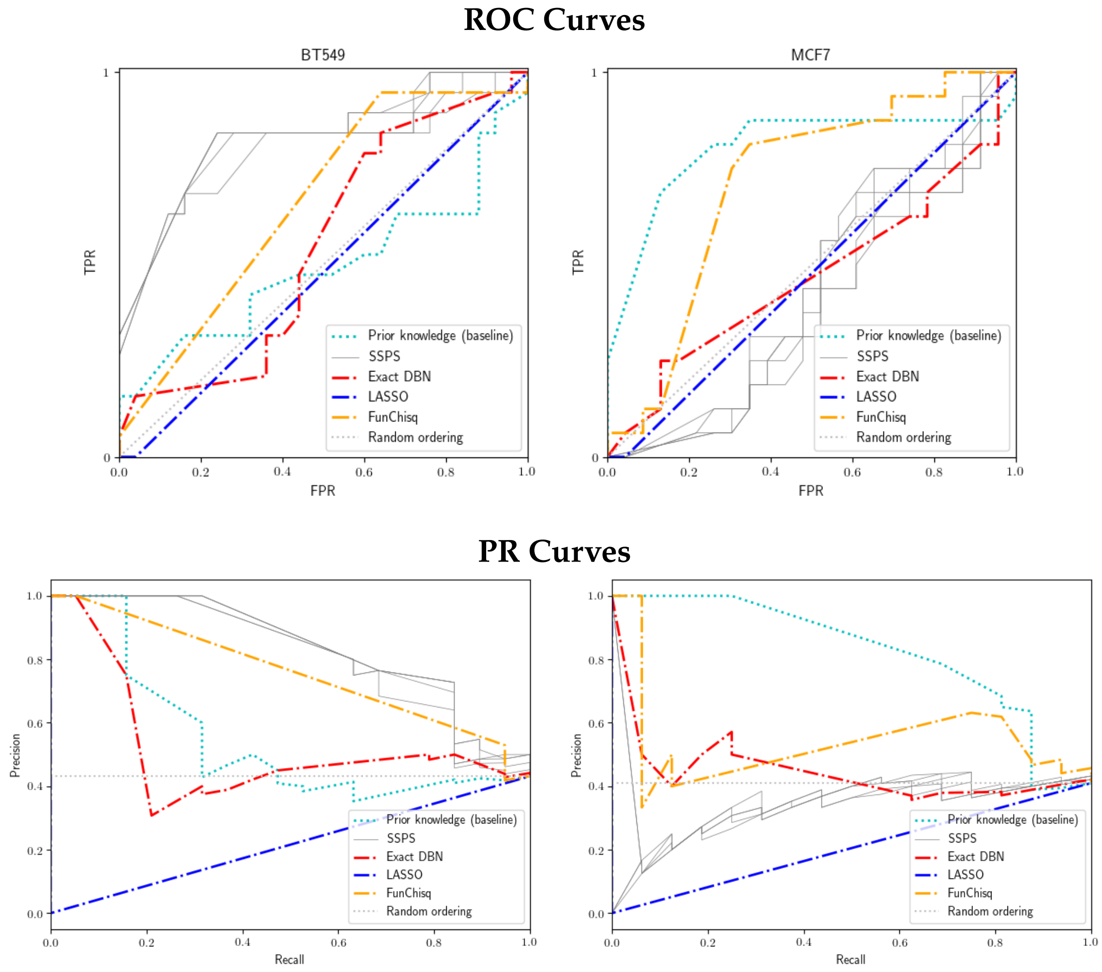


Figure 2.10: ROC curves (top) and PR curves (bottom) from the HPN-DREAM challenge. We show results for two contexts: cell line BT549 (left) and MCF7 (right). The stimulus is EGF for both contexts. Since SSPS is stochastic, we show all 5 of its curves in each plot. The other methods are all deterministic, and therefore only have one curve in each plot.

3 VIEWING MULTIOMIC DATA THROUGH THE LENS OF MATRIX FACTORIZATION AND GENE SETS

This chapter describes PathMatFac, a matrix factorization model for multiomic data. PathMatFac is equipped with a mechanism that *interprets* the factorization in terms of curated gene sets. Evaluations on simulated and real datasets demonstrate that PathMatFac succeeds at (i) obtaining a unified, informative factorization of multiomic data and (ii) summarizing the factors in ways a biologist may find helpful.

3.1 Introduction

Motivation

Biological systems are complicated. The state of a cell, tissue sample, or organism can hardly be captured by any single measurement. Biologists gain insight into these systems by collecting high-dimensional data from them. It's not unusual for an omics datasets to contain thousands of features.

This situation leads to a perennial challenge in bioinformatics: *how do we summarize high-dimensional data in a way that yields biological insight?* An enormous set of methods exists to solve this problem, for diverse contexts and kinds of data. One way to understand these tools is to map them onto a continuum between two extremes: (i) *data-centric* methods, which aim only to model the data; and (ii) *biology-centric* methods, which summarize the data in terms of existing biological knowledge. The data-centric techniques include many standard tools of data science. For example, principal components analysis (PCA) and hierarchical clustering appear frequently in bioinformatics.

In contrast, biology-centric methods use extant biological knowledge as a fixed *vocabulary* to describe new data. The description is generally lossy, since modeling the data is not a priority. Gene Set Enrichment Analysis (GSEA) is a classic example of a biology-centric technique (Subramanian et al., 2005). GSEA uses prior knowledge in the form of *gene sets* to identify differences between populations of samples. Each gene set is curated such that it contains genes related to a particular phenomenon or process of interest—for example, a biological pathway. Other enrichment-based techniques like Gene Ontology (GO) analysis (Young et al., 2010) and Gene Set Variation Analysis (GSVA) (Hänzelmann et al., 2013) also fall into this category.

Both classes of method have their uses. Data-centric techniques compress and summarize the data with relatively little bias. This makes them well-suited to explore *new* phenomena that don't fit cleanly into existing knowledge. On the other hand, biology-centric methods produce summaries that are highly interpretable, especially to subject matter experts (the biologists). A skilled biologist can readily inspect the output of GSEA, for instance, and begin to reason and form hypotheses about the underlying biology.

In addition to its dimensionality, the *heterogeneity* of omics data poses other challenges. The *samples* of an omics dataset may exhibit heterogeneity in several ways. Datasets may contain samples collected from different biological conditions. For example, they may come from different cell types, tissues, or organisms. Samples may possess different disease status, or be exposed to different drugs. Depending on the scientific context, these biological differences may be (*i*) the phenomenon of interest, or (*ii*) a nuisance to be modeled away.

Samples may also differ in non-biological ways. The experimental procedures for collecting omics data are complicated and provide many opportunities for technical artifacts and inconsistencies to arise between samples. For instance, omics data is usually collected in *batches*. Batches of data measured by different laboratories—or machines within a laboratory—are not typically *i.i.d.*. It's common for batch effects to account for a large fraction of the variation in an omics dataset. Naïvely ignoring these effects will lead to incorrect analytical conclusions.

The *features* of an omics dataset may also be heterogeneous. For instance, *multiomic* datasets contain features from multiple omics modalities. Multiomic data can be advantageous, since different omics provide complementary views of the underlying biology. However, they also pose a challenge since different modalities possess distinct distributional properties. Modeling them jointly requires judgment and creativity.

Furthermore, biological datasets often contain missing values. Sometimes the missingness is a function of the biology, and therefore contains signal. In other cases it results from randomness, or practical limitations of the assay technology. Models need to appropriately capture these kinds of missingness.

Figure 3.1 shows a concrete example of the high-dimensional, heterogeneous omics data we're discussing. This dataset comes from The Cancer Genome Atlas (TCGA), a series of studies that collected multiomic data from thousands of tumor samples representing many kinds of cancer (The Cancer Genome Atlas Network et al., 2013). Most samples in the TCGA dataset have tens of thousands of features belonging to several omics modalities: RNA-seq, CNA, somatic mutations, and methylation (among others). TCGA was a large, collaborative effort from several laboratories over the span of years. As a result, the samples were collected and processed at different times and locations, creating the possibility for nuisance technical variation. Multiomic datasets like TCGA present difficulties for modeling. However, they may reveal rich and valuable biological insights, and are therefore worth the additional modeling effort.

In this chapter we present PathMatFac, a new tool for extracting insights from multiomic datasets. PathMatFac is a matrix factorization model that has the strengths of both data-centric *and* biology-centric methods. It achieves this by:

1. obtaining an informative factorization of the multiomic data; and
2. *interpreting* the linear factors in terms of curated gene sets.

Related works

There are a vast assortment of methods that employ some combination of (*i*) multiomic data, (*ii*) biological prior knowledge; or (*iii*) matrix factorization. We mention some of them in order to provide context for the original contributions in this chapter.

Closely related works. We highlight several works as *direct* inspirations for PathMatFac. PARADIGM is a tool that uses biological pathways to construct a factor graph model for multiomic data (Vaske et al., 2010b). It uses the factor graph to produce smoothed estimates of "activity levels" for pathway entities (e.g., genes, and proteins). The activity levels can then be used for downstream analyses. PARADIGM was successful and influential in its time. The Cancer Genome Atlas thoroughly incorporated it into their analysis workflows (The Cancer Genome Atlas Network, 2012a,b,c, 2014; Hoadley et al., 2018). However, PARADIGM has notable shortcomings. It is designed specifically for RNA-seq and CNA data, and does not readily incorporate other kinds of omics. PARADIGM contains some dubious modeling choices; omics data must be discretized, and biological pathways are translated into factor graphs in a very literal and mechanistic fashion that assumes full knowledge of the molecular processes. Furthermore, the software is difficult to acquire and use. An executable binary is available online, but the source code must be requested by email (Stuart, 2023). The need for newer, better tools combining multiomic data and biological pathways was an initial impetus for PathMatFac.

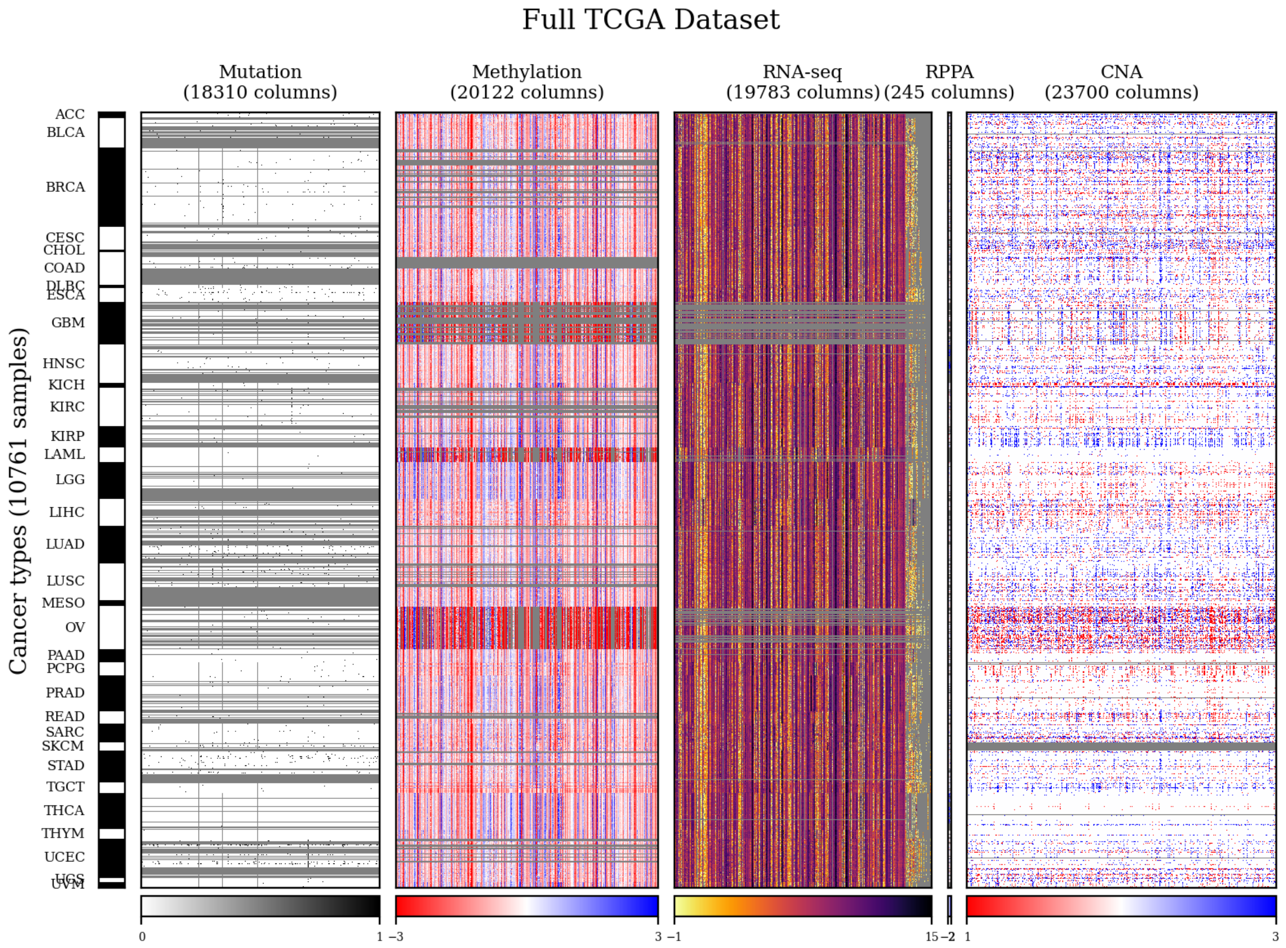


Figure 3.1: A visualization of the TCGA dataset used in this work. Each row is a sample; each column is an omic feature. Rows are grouped by cancer type, indicated by the black-and-white bar on the left. Columns are grouped by omics type (i.e., modality). Gray entries indicate missing data. There are important distributional differences between the modalities. Somatic mutation data are binarized; methylation and RNA-seq data are assumed Gaussian; and CNA is ordinalized with three levels. Notice how little RPPA data exists in comparison to the other modalities. We visualize RPPA here, but exclude it from our analyses since it has (*i*) few features, (*ii*) many missing entries, and (*iii*) potential quality issues (Upadhya and Ryan, 2023).

PLIER is a matrix factorization model for omics data (Mao et al., 2019). It uses curated gene sets to (*i*) inform and (*ii*) interpret its linear factors. This represents a compromise between data-centric and biology-centric approaches. PLIER fits factors to the data, but seeks to explain those factors in terms of gene sets. PLIER is only intended to model a single omics dataset—typically RNA-seq. PathMatFac can be construed as a multiomic improvement to PLIER, though its probabilistic assumptions differ from PLIER in other important ways.

MOFA+ is a Bayesian matrix factorization model targeted at multiomic datasets (Argelaguet et al., 2020). It uses a combination of priors to induce sparsity on its factorization. MOFA+ employs variational Bayes inference to fit a mean field approximation of its posterior distribution. The sparsity of MOFA+'s factors aids interpretation, but does not explicitly link them to biological prior knowledge (e.g., gene sets or pathways). PathMatFac is similar to MOFA+, in that it is also a multiomic matrix factorization. However, PathMatFac's probabilistic assumptions are somewhat different, and include dependencies between the factors and user-provided gene sets. PathMatFac also relies on a simpler MAP estimation procedure that can easily accommodate future model improvements.

PathMatFac took significant inspiration from the work of Udell (2015) in Generalized Low Rank Models (GLRMs). GLRMs provide a helpful framework for thinking about regularized matrix factorizations with heterogeneous features. PathMatFac is a GLRM equipped with some additional machinery for interpreting its linear factors.

ComBat is a widely used tool for modeling and removing batch effects from omics data (Johnson et al., 2007). It originally targeted assay technologies like DNA microarrays, which have since become outdated. However, ComBat continues to prove useful, even for modern single cell RNA-seq datasets (Luecken and Theis, 2019). PathMatFac borrows probabilistic ideas from ComBat to account for batch effects.

Method attribute	PathMatFac	MOFA+	PLIER	PARADIGM
Provides biological interpretations	✓	✗	✓	✗
Accommodates multiomic data	✓	✓	✗	✓
GPU-accelerated	✓	✓	✗	✗
Models batch effects	✓	✗	✗	✗

Table 3.1: Comparison of PathMatFac against closely related techniques. PathMatFac occupies a unique niche and combines many of the strengths of existing techniques.

Table 3.1 compares PathMatFac's capabilities against the most closely-related methods for embedding omics data. PathMatFac shares many attributes with these techniques, though it combines them in a unique way. Its modeling assumptions and inference procedure also differ meaningfully from those of existing methods.

Other related works. For a richer context, we mention some other methods that utilize (*i*) multiomic data or (*ii*) biological prior knowledge to accomplish related tasks. We focus especially on tools for multiomic data, since the field is developing at a rapid pace. Methods exist for a variety of tasks on multiomic data: embedding, clustering, supervised prediction, imputing missing values, and hypothesis testing. The literature often lumps these tasks together under the umbrella term "multiomic integration." We avoid using that term, since it has become clichéd and uninformative.

Techniques for *embedding* multiomic data have proliferated in very recent years. Neural networks, and especially Variational Autoencoders (VAEs), have gained currency. The Multimodal VAE (MVAE) network architecture (Wu and Goodman, 2018) is a key influence for many of them. MVAE uses a *product of experts* design to combine representations from different modalities. Cobolt (Gong et al., 2021) adapts MVAE to the multiomics domain and employs likelihoods tailored to specific omics modalities. MultiVI (Ashuach et al., 2021) and Multigrate (Lotfollahi et al.,

2022) improve on Cobolt by (*i*) enabling imputation of missing modalities and (*ii*) accounting for sources of technical variation, such as batch effect. TotalVI (Gayoso et al., 2021) has similar goals but is tailored specifically for CITE-seq data, a joint transcriptomic and proteomic assay technology (Stoeckius et al., 2017). moETM (Zhou et al., 2023) uses fully-connected encoders and a *linear* decoder, in order to promote interpretability. pmVAE (Gut et al., 2021) uses biological pathways to inform the *architecture* of a VAE and make its inferences more interpretable. Finally, SMILE (Xu et al., 2022) uses a self-supervised neural network with noise-contrastive loss (as opposed to a VAE) to embed multiomic data.

Linear embedding techniques continue to be relevant in the multiomic setting. They possess desirable properties such as (*i*) interpretability, (*ii*) suitability for smaller datasets and (*iii*) modest computational expense. MOFA+ (Argelaguet et al., 2020), described above, is inspired by Group Factor Analysis (Klami et al., 2015) and Bayesian CCA (Klami et al., 2013). Canonical Correlation Analysis (CCA) is itself a classic statistical technique for multimodal data; CCA finds directions of greatest correlation between modalities (HOTELLING, 1936). See the review by Hardoon et al. (2004) for additional details. Integrative nonnegative matrix factorization (iNMF) (Yang and Michailidis, 2016) is a multimodal extension of nonnegative matrix factorization.

We also mention one *biology-centric* embedding technique for omics data. GSVA (Hänzelmann et al., 2013) scores gene set enrichment for individual samples, revealing variation between them. GSVA is not usually described as an *embedding*, though it amounts to a transformation of the data. GSVA is intended for a single omics modality, typically RNA-seq. We employ GSVA as a baseline in Section 3.3.

Clustering is another common task for multiomic data. Multiomic clustering usually consists of (*i*) an embedding procedure coupled with (*ii*) a standard clustering procedure. For example, the iCluster (Shen et al., 2009;

Mo et al., 2013, 2018) and moCluster (Meng et al., 2016, 2019) techniques use matrix factorization to embed the multiomic data, followed by k-means clustering. LIGER (Welch et al., 2019) employs iNMF to embed multiomic data, together with a neighbor-graph clustering procedure. See the review by Chauvel et al. (2020) for other examples. Meanwhile, Similarity Network Fusion (SNF) takes a distinct approach to multiomic clustering (Wang et al., 2014). SNF uses the multiomic data to produce a similarity matrix for pairs of samples, and then performs spectral clustering on that matrix.

Other "multiomic integration" techniques consist of statistical tests applied to multimodal data from *populations* of samples. In the terminology of this chapter, almost all of these techniques fall into the *biology-centric* category. Most of them are multiomic extensions of classic enrichment-based comparisons, like GSEA (Subramanian et al., 2005). The comprehensive review by Maghsoudi et al. (2022) provides many examples of these. A related class of techniques uses multiomic data to identify significant *perturbations* in a given biological network. For example, HotNet2 (Leiserson et al., 2015) and Conflux (Mezlini and Goldenberg, 2017) use one or more genomics modalities to identify perturbed subnetworks in a curated Protein-Protein Interaction (PPI) network.

Finally, many techniques exist for supervised learning on multiomic data. A recent review by Wysocka et al. (2023) thoroughly explores the neural networks proposed for that task, with an emphasis on models that incorporate biological prior knowledge in some fashion. The review also offers a perspective on the relationships between (*i*) interpretability, (*ii*) biological prior knowledge, and (*iii*) network architectures.

New contributions in this chapter

We present PathMatFac, a new matrix factorization technique for multiomic data. The model flexibly accommodates realistic omics datasets containing (*i*) multiple data modalities with distinct distributional properties; (*ii*) batch effects; (*iii*) multiple biological conditions; and (*iv*) missing values.

PathMatFac uses Automatic Relevance Determination (ARD) to regularize its factors. The model's probabilistic assumptions *connect* the ARD regularizer to user-provided gene sets, enabling the model to *interpret* the factors in terms of those gene sets. We refer to this new probabilistic mechanism as Feature Set ARD (FSARD).

We evaluate PathMatFac on simulated and real datasets. Simulations confirm that the model reliably recovers its true parameters. Tests with real data from The Cancer Genome Atlas (TCGA) demonstrate that PathMatFac yields an informative embedding of multiomic data, competitive against baseline matrix factorizations. They also show PathMatFac's ability to summarize factors in terms of curated gene sets, whenever that is possible.

We distribute PathMatFac as a Julia package on GitHub:

https://github.com/dpmerrell/PathwayMultiomics.jl

3.2 Proposed method

Probabilistic model

This section lays out PathMatFac's probabilistic assumptions and model parameters. An explanation of its inference procedure comes later.

We designed PathMatFac to model a particular class of dataset. Some notation will clarify this discussion. Let D be an $\mathrm{M} \times \mathrm{N}$ array of numeric data. Each row $\mathrm{i} \in [\mathrm{M}]$ represents a sample; each column $\mathrm{j} \in [\mathrm{N}]$ represents a feature. For a concrete example see Figure 3.1, which visualizes multiomic data from The Cancer Genome Atlas (TCGA). That dataset contains $\mathrm{M}{=}10,761$ samples belonging to 33 different cancer types. Multiple kinds

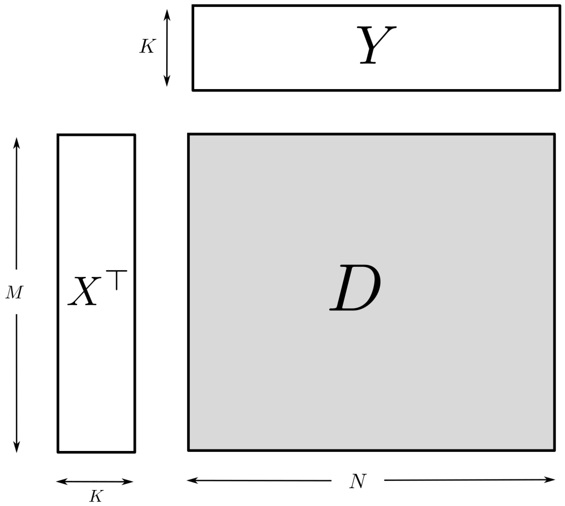


Figure 3.2: Illustration of a basic matrix factorization model. D is the dataset, an $\mathsf{M}\times\mathsf{N}$ array where rows are samples and columns are features. The model assumes $\mathsf{D} \sim \mathsf{X}^\top\mathsf{Y}$. Rows of Y are *linear factors*; X contains a K-dimensional *embedding* of the data. PathMatFac adds several parameters to this model to accommodate multiomic datasets.

of omics data were collected for each sample, totalling approximately $\mathsf{N}\simeq 80,000$ features.

PathMatFac models the dataset D via matrix factorization. That is, it assumes D is generated by multiplying two much smaller matrices $\mathsf{X}\in\mathbb{R}^{\mathsf{K}\times\mathsf{M}}$ and $\mathsf{Y}\in\mathbb{R}^{\mathsf{K}\times\mathsf{N}}$:

$$\mathsf{D} = \mathsf{X}^\top\mathsf{Y} + \epsilon,$$

where ϵ is *i.i.d.* Gaussian noise (in the case of Gaussian data; analogous noise models apply for non-Gaussian data).

It can be instructive to write this dependence in some alternative—

though equivalent—forms. First, in terms of the *rows* of X and Y:

$$D = \sum_{k=1}^{K} \underline{x}_k \underline{y}_k^\top \; + \; \epsilon,$$

where $\underline{x}_k$ is the k^{th} row vector of X.

Or, if we focus on a single entry $d_{i,j}$ of the dataset, the dependence can be written in terms of *columns* of X and Y:

$$d_{i,j} \sim x_i^\top y_j \; + \; \epsilon_{i,j} \qquad \forall i, j$$

where x_i is the i^{th} column vector of X. See Figure 3.2 for an illustration of the matrix factorization setup.

This formulation is analogous to PCA. In our case, the row vectors of Y play a similar role to principal components. That is, the rows of Y ought to span a subspace of greatest variation in the dataset, defining a K-dimensional linear subspace of $\mathbb{R}^N$. We refer to the rows of Y as *linear factors* or simply *factors*. Then, as in PCA, the matrix X contains a K-dimensional embedding of the dataset on that subspace—the projection of each sample onto the factors.

The analogy to PCA is exact for simple Gaussian datasets that have been standardized to zero-mean and unit variance. However, in our case the data have a much more complicated structure, calling for a somewhat more complicated probabilistic model. PathMatFac introduces additional model parameters to accommodate non-Gaussian, multiomic datasets. Other parameters "model away" nuisance batch effects common in omics data. PathMatFac employs specially chosen regularizers that encourage the linear factors to be sparse and statistically robust. Lastly, the model connects the linear factors to prior knowledge in a way that aids biological interpretation. These design choices are all grounded in principled, probabilistic considerations.

Accommodating heterogeneous features. PathMatFac aims to model a multiomic dataset. By "multiomic," we mean a multimodal dataset where features may represent measurements from different omics assays. This requires modifications to the matrix factorization described above.

Some more notation will aid this discussion. We assume the N features of data are partitioned into a set of *views* $V = \{v_1, v_2, \ldots\}$. In other words, each feature j belongs to exactly one $v \in V$. In practice these views represent different modalities of omics data. For example, there could be a view of RNA-seq features, a view of methylation features, and so on. We use $v(j)$ to denote the view containing feature j. We occasionally use v as a subscript to indicate view-specific quantities. For example, $N_{v_1} = |v_1|$ is the number of columns belonging to view v_1. We use the terms *view*, *modality* and *omics assay* interchangeably in this chapter.

In general the views of data may have distinct distributional properties. Data from some omics assays may be treated as Gaussian, while others may be, e.g., Bernoulli- or Poisson-distributed. PathMatFac accommodates this heterogeneity by allowing column-specific distributional assumptions:

$$d_{i,j} \sim P_j \left(d \mid x_i^\top y_j \right).$$

That is, the model allows columns of data to have different putative distributions. At the time of this writing PathMatFac admits Gaussian, Bernoulli, ordinal, and Poisson features.

PathMatFac maps these distributional assumptions to appropriate column-specific *losses* and *link functions*:

$$d_{i,j} \sim P_j \left(d \mid x_i^\top y_j \right) \quad \rightarrow \quad \mathcal{L}_j \left(d_{i,j} \, , \, x_i^\top y_j \right)$$

where $\mathcal{L}_j$ represents the composition of an *inverse* link function and loss function. For instance, putatively Gaussian features call for a quadratic loss and an identity link function. Other distributions may admit different

Assay	Distribution	Loss	Link
RNA-seq (log-transformed)	Gaussian	quadratic	identity
Methylation (logit-transformed)	Gaussian	quadratic	identity
Somatic mutation (binarized)	Bernoulli	squared hinge	NA
CNA (ternarized)	ordinal	ordinal squared hinge	NA

Table 3.2: The omics modalities used in this chapter, with their assumed distributions, loss functions, and link functions.

combinations of loss and link functions. For example, Bernoulli features may use cross-entropy loss with a probit or logistic link function; or a totally different loss, like squared hinge loss (without a link function). The *link function* terminology originates from Generalized Linear Models; we recommend the review by Neuhaus and McCulloch (2011) for additional background. All analyses in this chapter use the distributions, losses, and link functions given in Table 3.2.

The matrix factorization model's loss (i.e., negative log-likelihood) is simply the sum of these losses across the dataset:

$$\mathcal{L}\left(D, X^\top Y\right) \stackrel{\text{def}}{=} \sum_{i,j} \mathcal{L}_j\left(d_{i,j}\ ,\ x_i^\top y_j\right),$$

though this will soon be augmented with terms for (*i*) additional parameters and (*ii*) regularizer losses on X and Y.

Notice from Table 3.2 that we use squared hinge loss for Bernoulli features. Earlier versions of the model used cross-entropy loss with a logistic link. However, we saw empirically that this produced poor fits on multiomic datasets. Gaussian features tended to overpower the Bernoulli features, yielding factors that spanned the Gaussian features while ignoring the Bernoulli ones. We hypothesized that the bounded gradient of logistic cross-entropy caused this imbalance. In contrast, *probit* cross-entropy has an increasing gradient. Squared hinge loss also possesses that property, but is much simpler and less computationally expensive than probit cross-entropy. See Figure 3.3 for an illustration of these losses. We find that squared hinge loss yields improved balance between Bernoulli and Gaussian features.

To the extent that squared hinge loss approximates probit cross-entropy, there exists a *probabilistic* rationale for choosing squared hinge loss. Probit regression assumes the Bernoulli outcome is weighted by the probability of a latent Gaussian variable *exceeding a threshold*. Meanwhile, binarized omics data (e.g., our somatic mutation data) often arises from *thresholding* some numeric score. This suggests probit cross entropy (and hence squared hinge loss) could be more consistent with the data's generative process than logistic cross entropy.

Straightforward analogies hold for losses on ordinal variables; i.e., for (*i*) ordinal logistic loss, (*ii*) ordinal probit loss, and (*iii*) ordinal squared hinge loss. We noticed similar issues with ordinal logistic loss and obtained better fits with ordinal squared hinge loss.

Columns of data will generally have different means and variances. This poses another form of heterogeneity the model must accommodate. In the case of Gaussian data we would simply standardize the columns to have zero mean and unit variance. However, in our setting the dataset may contain Gaussian, Bernoulli, and ordinal features (among others). Standardization would be inappropriate for many of these.

A more appropriate strategy is to introduce column-specific *shift* and *scale* parameters, μ and σ, to the model:

$$d_{i,j} \sim P_j \left(d \mid x_i^\top y_j \cdot \sigma_j + \mu_j \right),$$

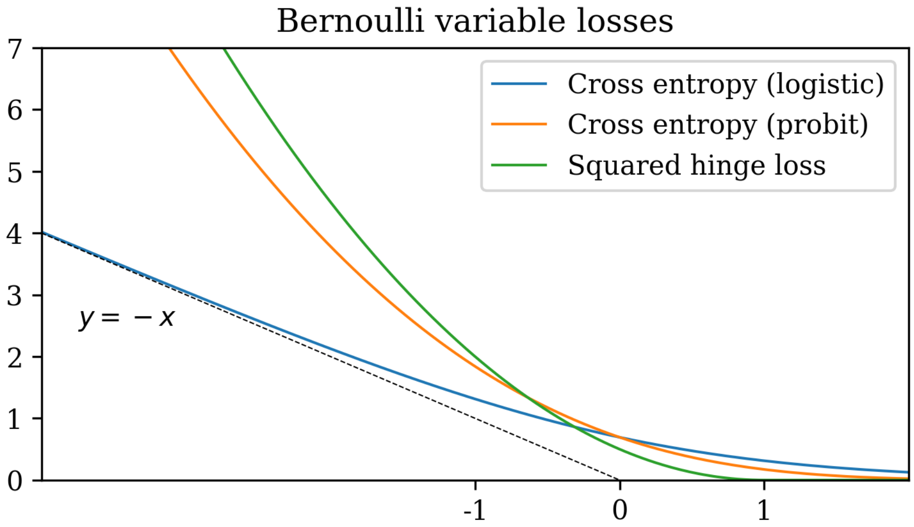


Figure 3.3: A plot of different loss functions for Bernoulli data (when d=1). Cross-entropy with a logistic link has a linear asymptote, resulting in weak gradients. In contrast, a probit link produces gradients that increase in magnitude as the model becomes less correct. Squared hinge loss shares this property with probit/cross-entropy, but is much simpler and entails less computational expense.

with an analogous modification to the model's loss:

$$\sum_{i,j} \mathcal{L}_j \left(d_{i,j} \, , \, x_i^\top y_j \cdot \sigma_j + \mu_j \right) .$$

On Gaussian data, μ and σ take the place of means and standard deviations for standardization, respectively. More generally, the column shifts μ serve as *intercept terms* or *biases* that account for systematic differences between columns. The column scales σ account for differences in dynamic range between columns.

Degeneracy exists between μ and σ for some kinds of data. For example, Bernoulli data with a small mean can be modeled equally well by (*i*) small, negative μ with small σ or (*ii*) large, negative μ with a commensurate increase in σ. In these cases—i.e., Bernoulli and ordinal data—we set σ=1 and rely on μ to capture the systematic differences between columns.

Another measure that helps PathMatFac accommodate heterogeneous features is *column loss reweighting*. This consists of multiplying each column's loss function $\mathcal{L}_j$ by a weight w_j in a way that makes columns exert similar "influence" on the factorization. We credit the Generalized Low Rank Model code of Udell (2015) for inspiring this technique.

Column loss reweighting works as follows. Assume the column shifts and scales, μ and σ, have already been assigned. One way to quantify column j's influence on the factorization is with the *norm of the gradient* of $\mathcal{L}_j$ with respect to X. Define $l_j(X)$, the loss from column j as a function of X:

$$l_j(X) \overset{\text{def}}{=} \sum_i \mathcal{L}_j(x_i^\top y_j \cdot \sigma_j + \mu_j).$$

The gradient of l_j is

$$\nabla_X l_j = \sigma_j \cdot y_j \left(\vec{\mathcal{L}}_j' \right)^\top$$

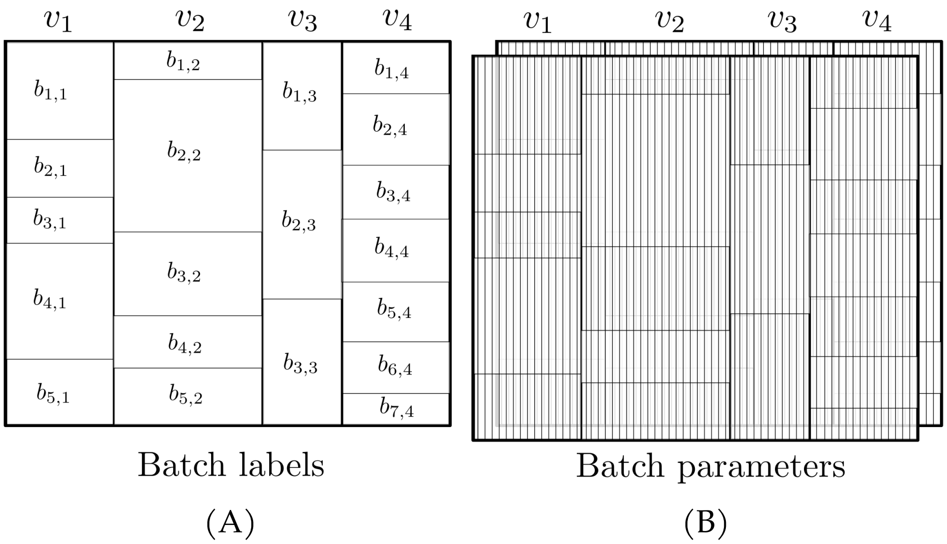
where $\vec{\mathcal{L}}_j'$ is the vector of partial derivatives $\left[l_j'(x_1), \cdots, l_j'(x_M) \right]^\top$. We can compute the norm of the gradient:

$$\|\nabla_X l_j\|_F = \sigma_j \cdot \|y_j\| \cdot \left\| \vec{\mathcal{L}}_j' \right\|$$

(with equality because it's an outer product).

Hence, we can ensure each column of data exerts a gradient on X of similar size by multiplying the loss of each column by the following weight:

$$w_j = \frac{1}{\frac{1}{\sqrt{M}} \cdot \sigma_j \cdot \left\| \vec{\mathcal{L}}_j' \right\|}$$



Batch labels

(A)

Batch parameters

(B)

Figure 3.4: Visualization of (A) batch labels in a multiomic dataset and (B) the batch parameters constructed by PathMatFac for that dataset. In reality the situation may be slightly more complicated: the batches of rows need not be contiguous. Two sets of batch parameters are constructed: *batch shifts* and *batch scales*. PathMatFac only constructs batch parameters for views of Gaussian data.

Notice that this weight ignores the contribution from $\|y_j\|$, since columns of Y are initialized with similar norm and their fitted norms are unknown. We also introduce a factor of $\sqrt{M}$ to remove w_j's dependence on problem size.

Accounting for batch effects. Up to this point we have described ways that PathMatFac accounts for heterogeneity in the *features* of a multiomic dataset. We now consider sources of heterogeneity between *samples* of data, and introduce model parameters to account for them. When biologists produce an omics dataset, this typically involves processing batches of samples. Batches processed in different laboratories, or on different machines, or even at different times, are subject to subtle differences in protocol or environmental conditions. This manifests itself in the dataset as distributional differences between batches. Nuisance variation of this kind, termed "batch effect," can obscure the data's biological signal if it isn't properly modeled away.

PathMatFac models two simple kinds of batch effect: (*i*) batch shifts and (*ii*) batch scales. These represent systematic differences in mean and variance between batches. PathMatFac currently only models batch effect for Gaussian features, though that may change in the future. We define some notation for batches and batch effect parameters. Consider a particular *view* of features, v. The rows of data in v are partitioned by a set of batch labels $B_v = \{b_{1,v}, b_{2,v}, \ldots\}$. That is, each sample i belongs to exactly one $b \in B_v$. We sometimes use $b_v(i)$ to indicate the batch containing row i in view v; or $b(i)$, if the view is clear from context.

Batches of samples will generally differ between views. Visually, the situation resembles a "patchwork quilt" of batch labels covering the dataset. See Figure 3.4(A) for illustration. PathMatFac relies on these batch labels to construct a set of batch parameters, θ and δ, modeling *batch shift* and *batch scale* respectively. For each view v and for each batch $b \in B_v$, we introduce column-specific parameters $\theta_{b,j}$ and $\sigma_{b,j}$. In other words, the model defines a vector of column shifts and a vector of column scales for each batch in the dataset. Figure 3.4(B) visualizes these parameters and shows their relationship to the batch labels.

PathMatFac incorporates batch parameters into the generative model as follows:

$$d_{i,j} \sim P_j \left(d \mid x_i^\top y_j \cdot \sigma_j \cdot \delta_{b,j} + \mu_j + \theta_{b,j} \right).$$

θ and δ can be regarded as *adjustments* to the column shifts (μ) and scales (σ), respectively. Figure 3.5 shows an updated schematic of PathMatFac's model parameters that includes θ and δ.

PathMatFac makes generative assumptions for θ and δ similar to those of ComBat (Johnson et al., 2007), a widely used tool for modeling away batch effect:

$$\begin{aligned} \delta^2_{b,j} &\sim \text{InvGamma}\left(\hat{\alpha}_b, \hat{\beta}_b\right) \\ \theta_{b,j} &\sim \mathcal{N}\left(m_b,\ s^2_b\right) \\ d_{i,j} &\sim \mathcal{N}\left(\mu_j + \theta_{b,j} + C_i w_j,\ \sigma^2_j \cdot \delta^2_{b,j}\right) \end{aligned} \tag{3.1}$$

where C is a matrix of *biological conditions* and w_j is a vector of regression coefficients against those conditions for column j, fit in a least-squares fashion. The idea is to disentangle batch effects from biological effects in a simple way, ensuring θ and δ capture mostly non-biological variation.

The modeling assumptions given in Equations 3.1 have the effect of *sharing information* between columns in a batch. Parameters within a batch tend to take similar values, greatly reducing the variance of the estimates. PathMatFac uses an Expectation-Maximization (EM) algorithm to fit θ and δ, similar to the procedure described in ComBat. This is described later in detail.

Regularizing the factors. PathMatFac employs Automatic Relevance Determination (ARD) to regularize the factors (Y), making them more interpretable and statistically robust. ARD is a Bayesian modeling concept that tends to impose sparsity on model parameters (Neal, 2012). The idea is to assume a Gaussian prior for the parameter:

$$y \sim \mathcal{N}(0, \xi^2).$$

However, unlike ridge regression, ARD further assumes the variance ξ^2 is itself an unknown, random variable.

ARD typically parameterizes the Gaussian with a *precision*, $\tau = \frac{1}{\xi^2}$ and gives it a Gamma hyperprior:

$$\tau \sim \text{Gamma}(\alpha, \beta).$$

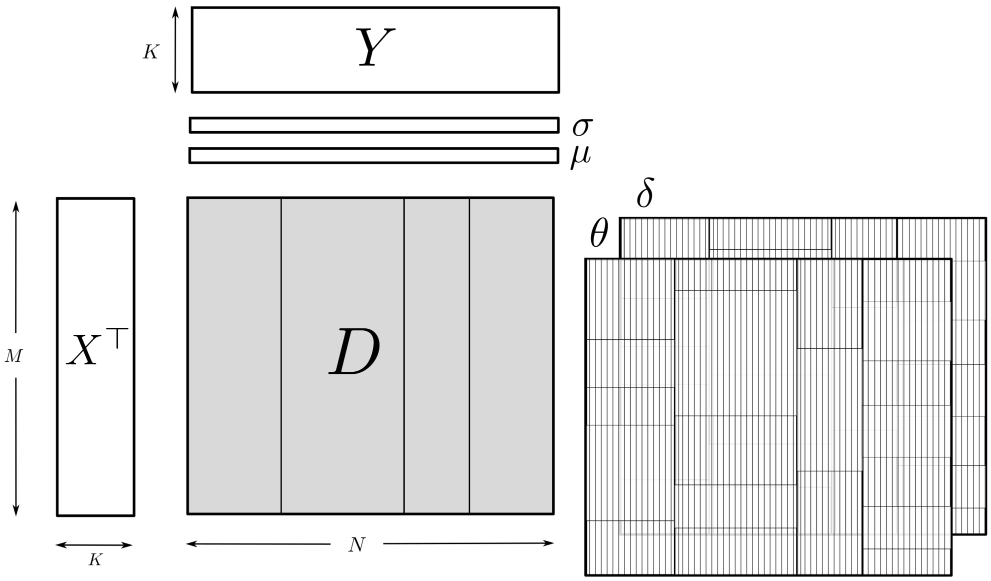


Figure 3.5: Diagram of the PathMatFac model, updated to include column and batch effect parameters. Parameters μ and σ are column shifts and scales. Parameters θ and δ account for batch shifts and scales.

Inference in such a model has the effect of driving τ to large values whenever y has little predictive power. This in turn drives y to exceedingly small values. As a result, only a sparse set of "relevant" model parameters take magnitudes much larger than zero.

There are different ways to incorporate ARD into a model. For example, MOFA+ employs a *structural* ARD, where τ variables are shared by entire vectors of model parameters (Argelaguet et al., 2020). Other models give each individual parameter its own independent τ. PathMatFac takes this latter approach to regularize the matrix Y. That is, each entry $y_{k,j}$ has the following generative assumptions:

$$\begin{aligned} \tau_{k,j} &\sim \text{Gamma}(\alpha, \beta) \\ y_{k,j} &\sim \mathcal{N}(0, 1/\tau_{k,j}) \end{aligned} \tag{3.2}$$

It can be challenging to fit models with ARD priors. This should be kept in mind when introducing ARD to a model. One may use the typical approximate Bayesian inference procedures—e.g., sampling or variational Bayes. For instance, MOFA+ uses variational Bayes with a mean-field approximation of the posterior (Argelaguet et al., 2018). *Evidence maximization* is another common approach. Evidence maximization first computes point estimates for each τ. Then, holding the estimated τs fixed, other model parameters are fit to the data (MacKay, 1999).

A less common approach for ARD models is straightforward maximum a posteriori (MAP) estimation. This involves (*i*) marginalizing out the τ variables and then (*ii*) optimizing the model parameters on the resulting posterior density. In other words, MAP reduces the ARD prior to a simple regularizer on the model parameters. There are principled arguments against MAP estimation for ARD models: the posterior is generally nonconvex and the maximum may not represent the bulk of the posterior probability mass (MacKay, 1999; Tipping, 2001). Despite these arguments, we experimented with MAP inference for PathMatFac and found that it reliably yields good fits on simulated and real datasets. We speculate that matrix factorization has unusual properties that make ARD and MAP inference a benign choice. Matrix factorization is already highly nonconvex; it may be that marginalized ARD regularization introduces other nonconvexities that are fairly harmless.

Since we employ MAP inference, we can marginalize out the latent τ variables. The probabilistic assumptions in Equation 3.2 allow us to integrate τ in closed form:

$$\begin{aligned} P(y_{k,j} \mid \alpha, \beta) &= \int P(y_{k,j} \mid \tau) \cdot P(\tau \mid \alpha, \beta) d\tau \\ &\propto \frac{\beta^{\alpha}}{\left(\beta + \frac{y_{k,j}^2}{2}\right)^{\alpha + \frac{1}{2}}}. \end{aligned} \tag{3.3}$$

This density is in fact a *scaled Student's* t*-distribution* for $y_{k,j}$. Under these assumptions, $y_{k,j}$ has finite variance *only* if $\alpha > 1$

$$\mathrm{Var}[y_{k,j}] = \frac{\beta}{\alpha - 1}. \tag{3.4}$$

It follows that choosing $\alpha{>}1$ is consistent with factor entries $y_{k,j}$ possessing finite variance. This is an appropriate choice for PathMatFac, since we have no reason to expect the entries of Y possess undefined or infinite variance. Furthermore, choosing $\alpha{=}1{+}\epsilon$ and $\beta{=}\epsilon$ for some small ϵ implies $\mathrm{Var}[y_{k,j}]{=}1$ and retains the sparsity-inducing properties of ARD. Contrast this with the typical practice in ARD, which constructs an *uninformative* prior on τ by setting $\alpha{=}\beta{=}\epsilon$ to a small number. Unless otherwise specified, our marginalized ARD regularization uses $\alpha = 1.001$ and $\beta = 0.001$.

In practice, MAP inference uses the log-density to regularize $y_{k,j}$:

$$\log P(y \mid \alpha, \beta) = -\left(\alpha{+}\frac{1}{2}\right) \cdot \log\left(\beta + \frac{y^2}{2}\right) \; + \; \text{const.} \tag{3.5}$$

Figure 3.6 contrasts this marginalized ARD regularizer with more traditional ones.

Finally, it bears mentioning that PathMatFac imposes L_2 regularization on X, consistent with a Gaussian prior:

$$X \sim \mathcal{N}(0, I). \tag{3.6}$$

At various points in the fitting procedure, this assumption is also enforced by *whitening* the embedding, such that each row of X has variance 1.

Interpreting factors with FSARD. We propose a probabilistic framework for interpreting the linear factors, called Feature Set Automatic Relevance Determination (FSARD). The goal of FSARD is to interpret the linear factors in terms of curated *gene sets* provided by the user. Gene

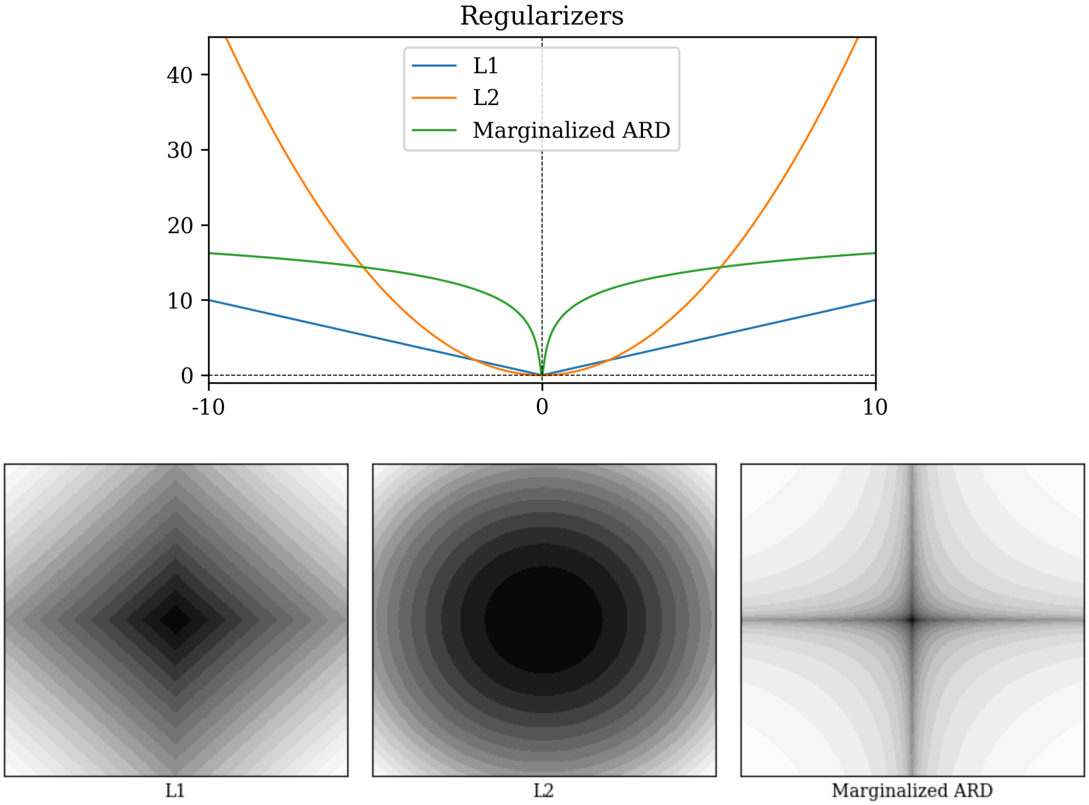


Figure 3.6: Plots of different regularizer losses. After marginalizing τ, ARD may be regarded as a regularizer on the model parameters. As a regularizer, marginalized ARD induces sparsity but is not convex.

sets are a commonly-used form of biological prior knowledge. A gene set is usually constructed to contain genes associated with a specific biological process or phenomenon. Collections of them can be found in public databases. A famous example is the Molecular Signatures Database (MSigDB) (Liberzon et al., 2015), which contains thousands of gene sets belonging to distinct categories. In this chapter we will use gene set collections from MSigDB based on *cancer hallmarks*, *oncogenic markers*, and *genomic locations*—and possibly others. In the ideal case, FSARD will iden-

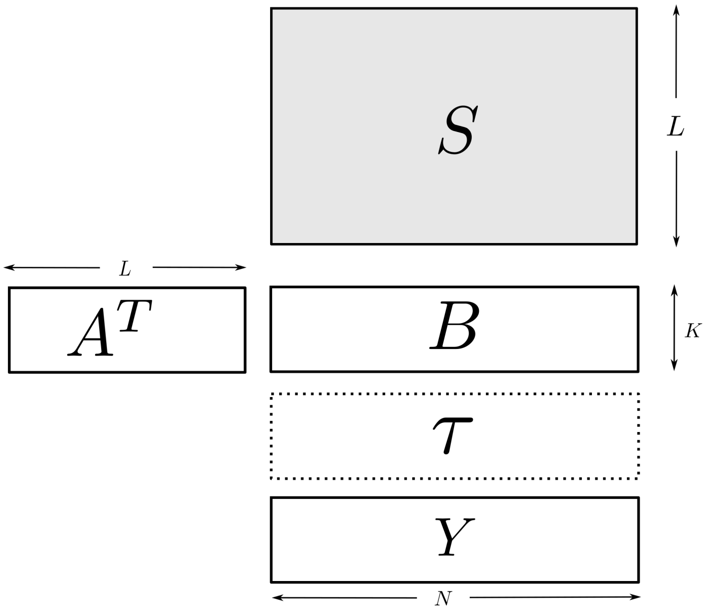


Figure 3.7: Diagram of Feature Set Automatic Relevance Determination (FSARD). See Equations 3.7 for the full specification. S is a sparse matrix that encodes gene sets provided by the user; and A is a nonnegative matrix of inferred *assignments* from gene sets to factors. Matrix B is generated from $A^\top S$. Each entry $\beta_{k,j}$ in B serves as the β parameter for an ARD prior on $y_{k,j}$. The goal is to "explain" the entries of Y by estimating A. The matrix τ is marginalized away, so we give it a dashed outline.

tify a small number of these gene sets that cover the nonzero entries of each linear factor.

First we frame the situation mathematically. Suppose the user provides a collection of L curated gene sets. We encode the gene sets in a sparse binary matrix $S \in \mathbb{B}^{L \times N}$, where $s_{l,j} = 1$ whenever feature j of the data involves gene set l. Suppose each factor $\overline{y}_k$ is sparse. In the ideal case, we would identify a small number of gene sets that, in combination, match the nonzero entries of $\overline{y}_k$. We frame this mathematically by introducing a nonnegative matrix of *assignments* $A \in \mathbb{R}_{\geqslant 0}^{L \times K}$, where $a_{l,k} > 0$ whenever gene set l matches factor k. When this is the case, we say gene set l is *assigned* to factor k.

FSARD aims to populate A with assignments in a sparse fashion. The challenge is to do this in a manner consistent with the probabilistic assumptions of ARD. Recall that ARD is parameterized by α and β, and that the variance of $y_{k,j}$ is $\beta/(\alpha - 1)$ (as in Equation 3.4). Hence, large magnitudes for $y_{k,j}$ are consistent with large β.

With these probabilistic considerations in mind, we modify ARD in the following ways:

- Allow each entry $y_{k,j}$ to have its own $\beta_{k,j}$ variable, rather than having all of them share a single β hyperparameter. In other words, there will be a matrix $B \in \mathbb{R}^{K \times N}$, having the same shape as Y.

- Treat each $\beta_{k,j}$ as a random variable that depends on matrices A and S. More precisely, FSARD assumes B depends on $A^\top S$.

Making it fully explicit, FSARD employs the following generative assumptions:

$$\begin{aligned} a_{l,k} &\sim \text{Exponential}(\lambda_k) & \forall k \in [K], l \in [L] \\ \beta_{k,j} &= (\alpha{-}1)\cdot(\gamma + a_k^\top s_j) & \forall k \in [K], j \in [N] \\ \tau_{k,j} &\sim \text{Gamma}(\alpha,\ \beta_{k,j}) & \\ y_{k,j} &\sim \mathcal{N}(0, 1/\tau_{k,j}). & \end{aligned} \tag{3.7}$$

Figure 3.7 depicts the new model parameters involved in FSARD, and their relationships to Y.

There is much to unpack here. Notice that

- The last two lines of Equations 3.7 are essentially the same generative assumptions as in ARD.

- $\beta_{k,j}$ is larger whenever A assigns gene sets containing feature j to factor k. This translates to weaker regularization on $y_{k,j}$.
- $\beta_{k,j}$ depends on a hyperparameter, γ. The meaning of γ becomes clear when $a_k^\top s_j = 0$. In that case, $\beta_{k,j}{=}(\alpha{-}1){\cdot}\gamma$, and the variance for $y_{k,j}$ implied by the generative process is

  $$\frac{\beta_{k,j}}{\alpha{-}1} = \gamma.$$

  In other words, γ represents the implicit variance for $y_{k,j}$ when no assignments are made. Small γ encourages FSARD to assign *more* gene sets to factors.
- FSARD assumes the entries of A are exponentially distributed. During inference, this has the effect of L_1-regularizing the entries of A. The exponential distribution has mean λ_k, a hyperparameter. We describe a technique for setting λ_k later on.

Throughout this description of FSARD, we have explained it as though there were a single assignment matrix A and gene set matrix S. In reality PathMatFac employs *view-specific* FSARD matrices: $A_{v_1}, A_{v_2}, \ldots$ and $S_{v_1}, S_{v_2}, \ldots$. This gives the model flexibility to use view-specific gene sets that are appropriate for the omics assay. For example, our analyses use gene sets based on *genome location* to interpret factors for copy number alteration features; and *cancer hallmark* gene sets to interpret factors for RNA-seq features. Figure 3.8 depicts PathMatFac's model parameters in their full complexity, including FSARD.

Inference procedure

PathMatFac has many parameters. We fit them to a multiomic dataset in multiple stages. The following coarse-grained steps summarize the fitting procedure:

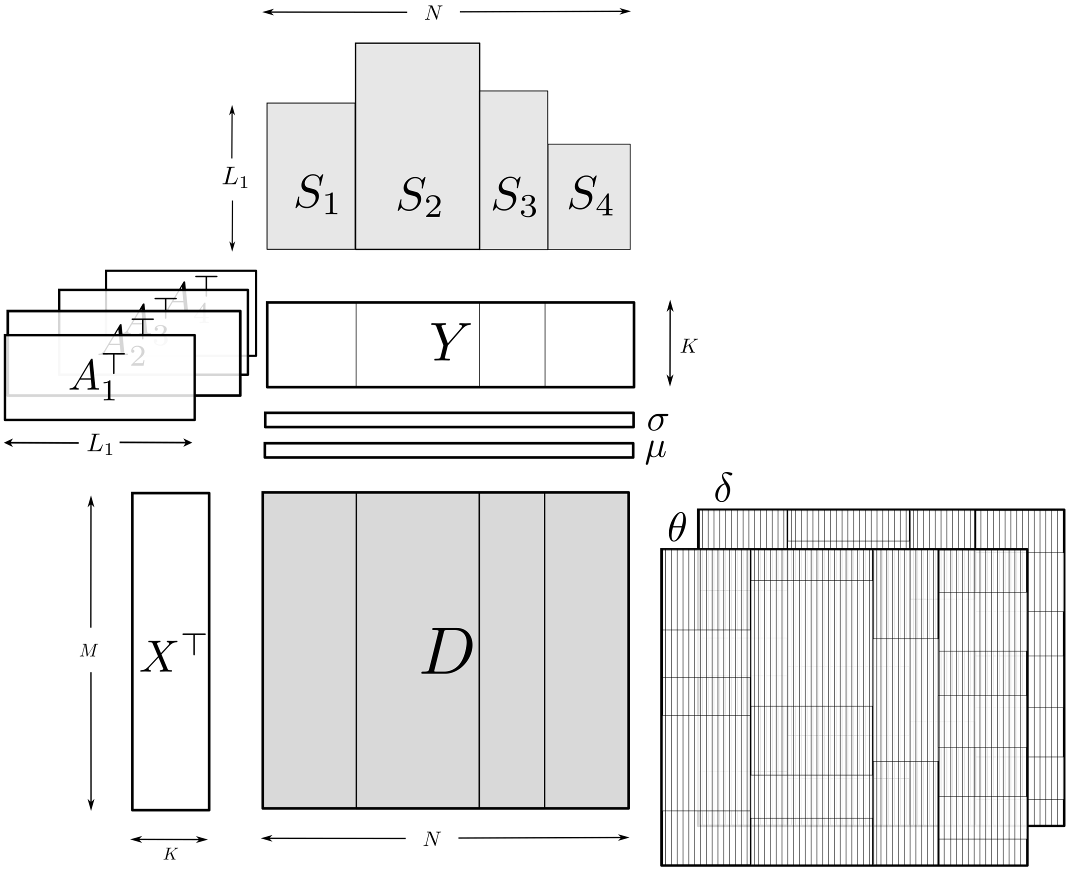


Figure 3.8: Full diagram of the PathMatFac data and model parameters. For Feature Set ARD (FSARD), each view v may have its own gene sets S_v and assignments A_v.

1. Fit the column shift and scale parameters, μ and σ.
2. If the user provides batch labels, then estimate the batch parameters θ and δ.
3. Fit the matrices X and Y via AdaGrad, under a simple quadratic regularization.
4. Adjust the matrices X and Y, with ARD regularization on Y.
5. Interpret the linear factors, Y, by fitting the FSARD assignment matrices $A_1, \dots, A_{|V|}$.

These steps amount to (one loop of) block-coordinate ascent on the model's posterior density. In the future it may be interesting to consider multiple loops; this could allow information-sharing between the linear factors and batch parameters, for instance. However, we do not explore that in this chapter. The remainder of this section describes the fitting steps in more detail.

Fitting column parameters. The goal is for each μ_j to be an *intercept term* encoding a "central" value for column j. We initialize each μ_j with the sample mean of column j. However, the sample mean isn't quite the correct value for all data types (e.g., ordinal data); so we refine the estimate via AdaGrad. In practice this converges in very few steps.

We set the column scales σ as described in Section 3.2. That is, we use the sample standard deviation for Gaussian features, and a fixed value of 1 for others. The fitting procedure will hold μ and σ fixed during the subsequent steps.

Fitting batch parameters. We devise an Expectation-Maximization (EM) procedure for θ and δ, holding μ and σ fixed. It employs the probabilistic assumptions listed in Equations 3.1. For each batch of Gaussian data b, the procedure initializes θ_b and δ_b^2 at their *least-squares* estimates, $\tilde{\theta}_b$ and

$\tilde{\delta^2}_b$. Then the procedure applies the following updates to m_b, s_b^2, $\hat{\beta}_b$, $\hat{\alpha}_b$, θ_b, δ_b^2 until convergence:

$$\begin{aligned}
m_b &\leftarrow E_j\langle\theta_{b,j}\rangle \\
s_b^2 &\leftarrow \mathrm{Var}_j\langle\theta_{b,j}\rangle \\
\\
\hat{\alpha}_b &\leftarrow 2 + \frac{E_j\langle\delta_{b,j}^2\rangle^2}{\mathrm{Var}_j\langle\delta_{b,j}^2\rangle} \\
\hat{\beta}_b &\leftarrow (\hat{\alpha}_b - 1)\cdot E_j\langle\delta_{b,j}^2\rangle \\
\\
\theta_{b,j} &\leftarrow \frac{|b|\cdot s_b^2\cdot\tilde{\theta}_{b,j} + \sigma_j^2\cdot\delta_{b,j}^2\cdot m_b}{|b|\cdot s_b^2 + \sigma_j^2\cdot\delta_{b,j}^2} \\
\\
\delta_{b,j}^2 &\leftarrow \frac{\beta_b + \frac{1}{2}\sum_{i\in b}\frac{(d_{i,j}-\mu_j-\theta_{b,j}-C_i w_j)^2}{\sigma_j^2}}{\alpha_b + \frac{1}{2}|b| - 1},
\end{aligned}$$

where $E_j\langle\cdots\rangle$ and $\mathrm{Var}_j\langle\cdots\rangle$ represent sample means and variances over index j, respectively.

Note that the updates for m_b, s_b^2, $\hat{\alpha}_b$, $\hat{\beta}_b$ are Method of Moments (MoM) estimates, so this isn't strictly an EM procedure. Meanwhile, θ_b and δ_b^2 are updated to their posterior means. Experiments with simulated data confirm that this yields better estimates than naïve least squares estimates in many cases; see Section 3.3 for details.

Fitting the matrices X and Y: stage 1. We fit X and Y in two stages. The first stage imposes a simple L_2 regularizer on the factors and fits them via AdaGrad.

The weight of the L_2 regularizer is set in a simple, empirical fashion based on the variance in the dataset. Suppose $d_{i,j}$ is a Gaussian entry in the dataset. By assumption,

$$d_{i,j} = x_i^\top y_j\cdot\sigma_j + \mu_j + \epsilon_{i,j}$$

$$\Rightarrow d_{i,j} - \epsilon_{i,j} = x_i^\top y_j \cdot \sigma_j + \mu_j,$$

where we have ignored batch effects. It follows that

$$\mathrm{Var}[d_{i,j} - \epsilon_{i,j}] = \sigma_j^2 \cdot \sum_k \mathrm{Var}[x_{k,i}] \cdot \mathrm{Var}[y_{k,j}],$$

where σ and μ are regarded as constants. Assume $\mathrm{Var}[x_{k,i}]=1$ (as in Equation 3.6) and that $\mathrm{Var}[y_{k,j}]=\xi^2$, $\forall k, j$. Then we have

$$\mathrm{Var}[d_{i,j}] + \mathrm{Var}[\epsilon_{i,j}] = \sigma_j^2 \cdot K \cdot \xi^2$$

$$\Rightarrow \xi^2 = \frac{\mathrm{Var}[d_{i,j}] + \mathrm{Var}[\epsilon_{i,j}]}{\sigma_j^2 \cdot K},$$

which implies a regularizer weight of $1/\xi^2 = \sigma_j^2 \cdot K / \mathrm{Var}_i \langle d_{i,j} \rangle$ for y_j. We derived this weight under Gaussian assumptions, but empirically we observe it also works well for Bernoulli and ordinal data. That is, it prevents overfitting during this stage of training.

We fit X and Y via AdaGrad with a simple adaptive rule for the learning rate, η. The optimizer begins with $\eta=1$. Then, if the loss *increases* for some number of steps, we update $\eta \leftarrow \eta/2$. This proceeds until we either reach a maximum number of steps or we meet a tolerance threshold for the loss decrease. The rule aims to (*i*) quickly find a basin in the loss and (*ii*) "slow down" to the extent necessary to approach a minimum. AdaGrad provides an additional measure of adaptivity to poorly-conditioned minima. Empirically we observe that the rule robustly minimizes loss across a variety of datasets and configurations for PathMatFac.

Matrix factorization is famously nonconvex; under L_2 regularization it has infinitely many solutions that are equally valid. We take some measures to address non-uniqueness at this point in the fitting process. The L_2 regularization at this stage ensures uniqueness of the *norms* of X and Y. Furthermore, we apply *whitening* to the factorization, which forces each row of X to have unit variance.

However, at this point we still only have uniqueness *up to an orthogonal transformation.* This poses an opportunity: can we pick an *advantageous* transformation from the infinite possibilities?

There are several commonly-used rules for applying orthogonal transformations in linear factor analysis. These include the varimax and quartimax transformations, among others (Abdi, 2003). We choose a transformation consistent with principal component analysis, defined as follows. Let $Y = U\Sigma V^T$ be the singular value decomposition (SVD) of Y. Then we transform X and Y by U:

$$X \leftarrow XU \qquad\qquad Y \leftarrow U^{\top}Y$$

For standardized Gaussian data, this is equivalent to using the (truncated) SVD of the dataset. More generally, this endows the factors with properties similar to PCA: it greedily aligns the factors with the directions of greatest variance in $\mathbb{R}^N$. This is desirable—we would rather have a small number of informative factors than a long tail of weakly informative ones. This is especially true since ARD regularization will *prune away* uninformative factors later on.

Fitting the matrices X and Y: stage 2. The next stage re-fits X and Y, this time with marginalized ARD regularization on Y and L_2 regularization on X. This amounts to MAP estimation of the factors, given an ARD prior on Y and a $\mathcal{N}(0, I)$ prior on X. The non-convex ARD regularizer has a steep gradient at small values and a gentle gradient at large ones. This tends to drive small values to zero, leaving a sparse set of large values. The adaptive AdaGrad rule described above is particularly useful for minimizing this loss, since the marginalized ARD produces large gradients. It employs similar termination conditions as in the previous stage. In practice, the optimizer can stop quite early and still yield a good factorization; see

Section 3.3 for a demonstration.

Interpreting the factors via FSARD. At this point we have fit the linear factors Y. Now our task is to *assign* gene sets to the factors, yielding biological interpretations for them. In other words, we need to fit the matrix of assignments A.

For clarity of exposition we will describe the procedure as though there were a single A matrix and S matrix. Recall, however, that FSARD employs *view-specific* S and A matrices. Accordingly, the following algorithm is applied to each *view* of Y with its corresponding S_v and A_v matrices.

We fit A in a MAP fashion, holding Y fixed. The probabilistic assumptions in Equations 3.7 yield the following loss function (i.e., negative log-density) for A:

$$\begin{aligned}\mathcal{L}_{\text{FSARD}}(A) = \sum_{k,j} \Big[& \alpha \log(\beta \cdot (\gamma + a_k^\top s_j)) \qquad (3.8)\\ & -(\alpha + \frac{1}{2}) \log(\beta \cdot (\gamma + a_k^\top s_j) + \frac{y_{k,j}^2}{2}) \Big] \\ & + \sum_k \frac{1}{\lambda_k} \cdot \|a_k\|_1 \qquad (3.9)\\ \text{s.t.} \qquad & A \geqslant 0.\end{aligned}$$

The first summation (3.8) results from the density of the marginalized ARD prior (i.e., scaled Student's t-distribution). Each term encourages $a_k^\top s_j$ to take larger values as $y_{k,j}^2$ increases, attaining a minimum loss when

$$a_k^\top s_j = \max\left(0, \frac{\alpha}{\beta} \cdot y_{k,j}^2 - \gamma\right).$$

As a consequence, this loss pushes $a_k^\top s_j$ toward zero whenever

$$y_{k,j}^2 < \gamma \cdot \frac{\beta}{\alpha}.$$

The second summation (3.9) and nonnegativity constraint come from the exponential prior for A. This encourages sparsity in A and enforces its intended semantics: large $a_{l,k}$ indicates high confidence that gene set l matches factor k.

Notice that each column a_k (and hence each row of Y) has its own regularizer weight, $1/\lambda_k$. We use the variance of the entries of Y to set each λ_k in a principled and inexpensive fashion.

$$\begin{aligned} Var[y_{k,j} \mid a_k, s_j] &= \frac{\beta_{k,j}}{\alpha_0 - 1} \\ &= \gamma + a_k^\top s_j. \end{aligned}$$

By the law of total variance we have

$$\begin{aligned} Var[y_{k,j}] &= \mathrm{E}\left[\mathrm{Var}[y_{k,j}|a_k, s_j]\right] + \mathrm{Var}\left[\mathrm{E}[y_{k,j}|a_k, s_j]\right] \\ &= \mathrm{E}\left[\gamma + a_k^\top s_j\right] \ + \ 0 \\ &= \gamma + L{\cdot}\mathrm{E}[a_{l,k}]{\cdot}\mathrm{E}[s_{l,j}] \\ &= \gamma + L{\cdot}\lambda_k{\cdot}\mathrm{E}[s_j]. \end{aligned}$$

From these observations we derive the following rule for setting λ_k:

$$\lambda_k \leftarrow \max\left(\epsilon,\ \frac{\mathrm{Var}_{k,j}\langle y_{k,j}\rangle - \gamma}{L{\cdot}\mathrm{E}_{l,j}\langle s_{l,j}\rangle}\right) \tag{3.10}$$

for some small $\epsilon > 0$. This rule has some intuitive properties. The L_1 regularizer weight $1/\lambda_k$ increases as the sample variance of Y decreases. It also increases with the number of gene sets (L) and the *density* of the gene set matrix, S.

At this point we have the loss function (Equation 3.8) and regularizer weights $1/\lambda_k \forall k$. We minimize this constrained, L_1-regularized loss via an Iterative Shrinkage and Thresholding Algorithm (ISTA) (Beck and Teboulle, 2009). ISTA is a simple and effective procedure for minimizing L_1-regularized losses. It wraps a gradient-based optimizer with an additional rule that (*i*) adjusts parameters toward zero, consistent with the regularization and then (*ii*) sets them to zero whenever they would otherwise cross the origin.

We choose ISTA over alternatives like coordinate descent for its simplicity, especially for GPU acceleration. We modify ISTA to (*i*) wrap AdaGrad rather than gradient descent and (*ii*) project the update into the feasible region (i.e., nonnegative values). It uses similar termination conditions as before. In practice ISTA yields useful parameter estimates in a very short time—the time expense is dwarfed by that of fitting X and Y.

PathMatFac's fitting procedure terminates once A has converged. On the other hand it is possible to continue alternating between fitting A and Y, in the fashion of Expectation-Maximization or block-coordinate ascent. This allows the gene sets to *inform* the linear factors, rather than merely *interpret* them. However, we see in practice that this severely distorts the factors and results in a poor fit to data. We recommend using gene sets only to *interpret* Y, rather than to bias the inference.

Notes about software implementation

PathMatFac is written in the Julia programming language and distributed as a package on GitHub:

https://github.com/dpmerrell/PathwayMultiomics.jl

It makes heavy use of Julia's automatic differentiation packages, namely ChainRules.jl, Zygote.jl, and Flux.jl (White et al., 2023; Innes, 2018; Innes et al., 2018). PathMatFac also enjoys GPU acceleration thanks to the CUDA.jl package (Besard et al., 2018).

Despite the merits of probabilistic programming described in Chapter 2, we chose not to use it for PathMatFac since (*i*) prominent probabilistic programming languages had limited support for automatic differentiation and (*ii*) we wanted the freedom to consider model choices not necessarily grounded in probability—e.g., squared hinge loss. We considered several model concepts prior to the one described in this chapter; it wasn't clear whether probabilistic programming would be the right tool for this job.

We built some additional software as a byproduct, in the course of implementing PathMatFac. MatFac.jl is a general-purpose, GPU-accelerated matrix factorization package written in Julia:

https://github.com/dpmerrell/MatFac.jl

To a large extent, PathMatFac is simply a wrapper around MatFac.jl.

We also built an automated Snakemake workflow to download, preprocess, and tabulate omics data from TCGA; we distribute the workflow on GitHub:

https://github.com/dpmerrell/tcga-pipeline

and provide the processed data on Zenodo (Merrell, 2022).

3.3 Evaluation

We test PathMatFac's capabilities in a suite of experiments on (*i*) simulated and (*ii*) real data. Simulations show that PathMatFac reliably estimates its parameters, and that FSARD assigns gene sets to linear factors with high precision. Runs on real multiomic data from TCGA show that PathMatFac and FSARD yield informative factors with biologically plausible interpretations.

Simulations

Some properties of PathMatFac are best demonstrated on simulated data, where we have access to ground truth. In each of the following experiments, we generate data consistent with the modeling assumptions outlined in Section 3.2. That is, we (*i*) generate *true* parameters, X_{True}, Y_{True}, μ_{True}, σ_{True}, θ_{True}, and δ_{True}; (*ii*) use them to simulate a dataset resembling real multiomic data, and (*iii*) fit PathMatFac to the simulated data. Then we compare its estimated parameters X_{Fitted}, Y_{Fitted}, μ_{Fitted}, σ_{Fitted}, θ_{Fitted}, and δ_{Fitted} against the true parameters, and score their agreement.

Measuring performance. Measuring the performance of matrix factorization on simulated data requires a careful choice of scores. We take a moment to define the scores used in our simulation study. At a minimum, PathMatFac should correctly fit the factors X and Y. However, the non-uniqueness of matrix factorization creates a challenge for scoring fitted factors—naïve approaches may unfairly penalize a valid factorization. What really matters is that the fitted factors span the same space as the true factors and assign magnitudes (i.e., singular values) to them in a consistent way.

We define a new score to capture this notion of correctness, called *span similarity*:

$$\text{spansim}(Y_{\text{True}}, Y_{\text{Fitted}}) = \frac{\|Y_{\text{True}} Y_{\text{Fitted}}^\top\|_F^2}{\sum_k \sigma_{\text{True},k}^2 \sigma_{\text{Fitted},k}^2} \tag{3.11}$$

where $\|\cdot\|_F$ is the Frobenius norm and $\sigma_{\text{True},k}$ is the kth singular value of Y_{True} (with the analogous definition for $\sigma_{\text{Fitted},k}$). In cases where Y_{True} and Y_{Fitted} have different numbers of rows, the denominator sums over the top K_{min} of them—whichever number is fewer.

Span-similarity measures the agreement between the rowspaces of Y_{True} and Y_{Fitted}. Its rationale becomes more apparent when we substitute the SVDs of both Y_{True} and Y_{Fitted} into the numerator:

$$\begin{aligned}
\|Y_{\text{True}} Y_{\text{Fitted}}^\top\|_F^2 &= \|U_T \Sigma_T V_T^\top V_F \Sigma_F U_F^\top\|_F^2 \\
&= \|\Sigma_T V_T^\top V_F \Sigma_F\|_F^2 \\
&= \sum_{i,j} \sigma_{\text{True},i}^2 \cdot \sigma_{\text{Fitted},j}^2 \cdot (\nu_{T,i}^\top \nu_{F,i})^2.
\end{aligned}$$

I.e., the numerator is maximized when Y_{Fitted} and Y_{True} have the same row space and assign magnitudes to their basis vectors in a consistent way. The denominator represents the maximal value of the numerator, attained when the row spaces agree exactly. So span similarity takes values between 0 (the row spaces are fully orthogonal) and 1 (the row spaces fully agree). Span-similarity can be regarded as a generalization of (squared) cosine-similarity. Whenever we compute span similarity we also compute a simple empirical p-value for it, in order to check whether the score is better than random. We sample 10,000 times from a simple null distribution by randomly permuting each row of Y_{Fitted}. We observe that in almost all cases, *none* of the null samples attain a higher span similarity than Y_{Fitted}, implying $p < 10^{-4}$. We will report whenever this is not true.

Span-similarity is also an appropriate score for the embedding matrix, X. We want to score agreement between the true embedding X_{True} and fitted embedding X_{Fitted}, in a way that ignores orthogonal transformations. This is exactly the rationale for span-similarity, so we use it to score X in our simulations.

Furthermore, we need to measure PathMatFac's ability to correctly interpret factors via gene sets. The data simulation process assigns a small number of gene sets to each linear factor, informing the factors' nonzero entries. Ideally, PathMatFac and FSARD would recover these gene sets with high fidelity. We adopt the following strategy to score this:

1. Compute the squared cosine similarity between every pair $(\underline{y}_{\text{True},k}, \underline{y}_{\text{Fitted},k})$; that is, every row of Y_{True} with every row of Y_{Fitted}.

2. Solve the *linear assignment* from rows of Y_{True} to rows of Y_{Fitted}, maximizing the sum of squared cosine similarities (computed in step 1). This naïvely matches *fitted* linear factors to *true* linear factors. When there are different numbers of true and fitted factors, the linear assignment will leave "left-over" factors which are ignored in the subsequent steps.

3. The match between true and fitted factors implies a match between the columns of true and fitted FSARD matrices, A_{True} and A_{Fitted}. Given that matching, we compute the AUCPR (area under the precision-recall curve—or, more accurately, the *average precision*) for each pair of columns $a_{\text{True},k}$ and $a_{\text{Fitted},k}$. This scores FSARD's ability to correctly recover A_{True}.

We take care to ensure this strategy does not artificially inflate scores. The criterion used to *match* rows of Y_{True} and Y_{Fitted} (i.e., cosine similarity) differs from the score used to measure agreement between A_{True} and A_{Fitted} (AUCPR). Furthermore, recall that each *view* of data has its own A_{True} and A_{Fitted} matrices. We use the same linear assignment to match A_{True} and A_{Fitted} for *every* view of data. Then we report the AUCPR, averaged across views.

The non-uniqueness of matrix factorization and the non-convexity of ARD suggest that recovering the assignments, A_{True}, could be a futile inference task. Perhaps surprisingly, the simulation results will demonstrate otherwise.

Other model parameters can be evaluated in simpler ways. We focus particularly on the batch effect parameters, since their inference procedure is relatively complicated. Plots of θ and δ allow us to judge whether the EM procedure fits them correctly. We also score the fit of θ via its coefficient of determination R^2 with the true (simulated) values. On the other hand, column shifts and scales (μ and σ) are estimated in straightforward ways and tend to fit well in practice.

Sensitivity to method hyperparameters. PathMatFac has free hyperparameters that need to be specified at training time. The most notable hyperparameters are K, the number of latent factors; and the number of training iterations. Simulations provide some insight into PathMatFac's sensitivity to these hyperparameters.

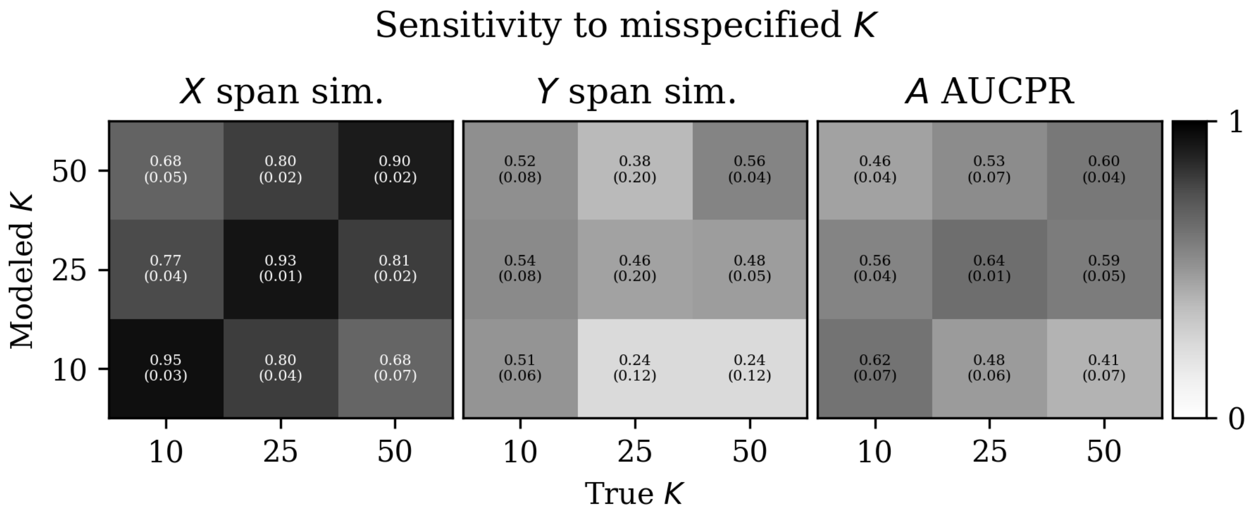


Figure 3.9: Sensitivity of PathMatFac parameter recovery to misspecified K, on simulated data. Each grid shows how a score varies with *true* K and *modeled* K, indicated by shade. "Span sim.": span similarity. Annotations report the mean value from 5 simulations, with the standard deviation in parentheses. The fidelity of fitted parameters noticeably degrades when K is misspecified.

Simulations demonstrate that PathMatFac has some sensitivity to the choice of K. Our first set of simulations varies (*i*) the *true* (simulated) value of K and (*ii*) the *modeled* value of K; fits the models, and then scores the resulting parameters. Other attributes of the problem are held fixed: the problems have size M=1,463 and N=4,108 with roughly equal numbers of features emulating somatic mutation, methylation, RNA-seq, and CNA modalities. The data simulator operates by mimicking a given *real* dataset—hence the M and N that are not clean, round numbers. We use this particular problem configuration because it resembles a *real* subset of the TCGA data that will be employed later in the evaluations.

Figure 3.9 shows the results. Whenever the modeled K does not match the true K, we see some degradation in the faithfulness of fitted parameters.

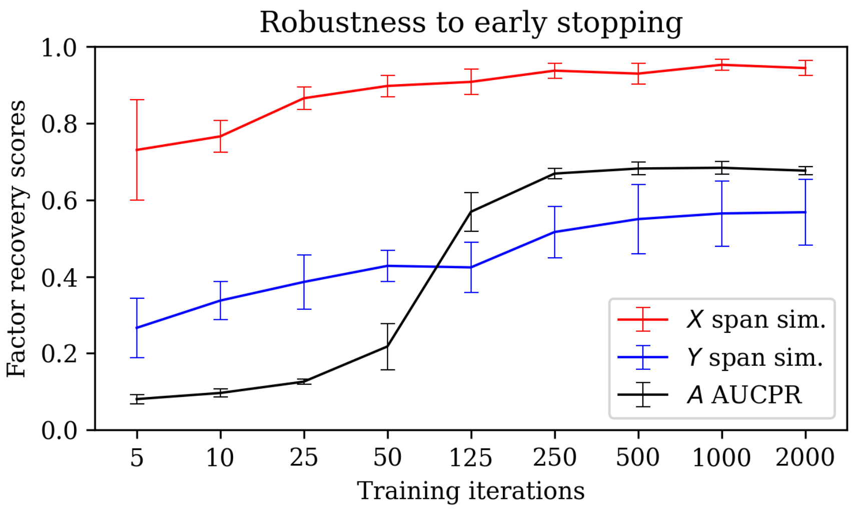


Figure 3.10: Robustness of PathMatFac's parameter recovery to early stopping. Each line reports the mean from 5 simulations; error bars show the standard deviation. Notice the nonlinear scale of the horizontal axis.

For Y, this is especially true when the modeled K is *smaller* than the true K. The degradation is roughly symmetric for X and A; that is, the degradation is roughly equal when K is too large or too small.

Despite this degradation, the scores are convincingly better than random in all cases. We see $p{<}10^{-4}$ for the X and Y span similarities in each of these simulations. For A, the AUCPR of a trivial predictor (i.e., the baserate) is 0.028 for these problems, far smaller than any scores attained by PathMatFac.

We conclude that, while the choice of K clearly matters, the output of PathMatFac may be *useful* even with incorrect K. Furthermore, the span similarities for Y suggest that it may be prudent to err on the side of *too many* factors in an initial run of PathMatFac.

On the other hand, simulations also show that PathMatFac's parameter recovery is remarkably robust to the number of training iterations. We run a suite of simulations that vary the number of training iterations, holding

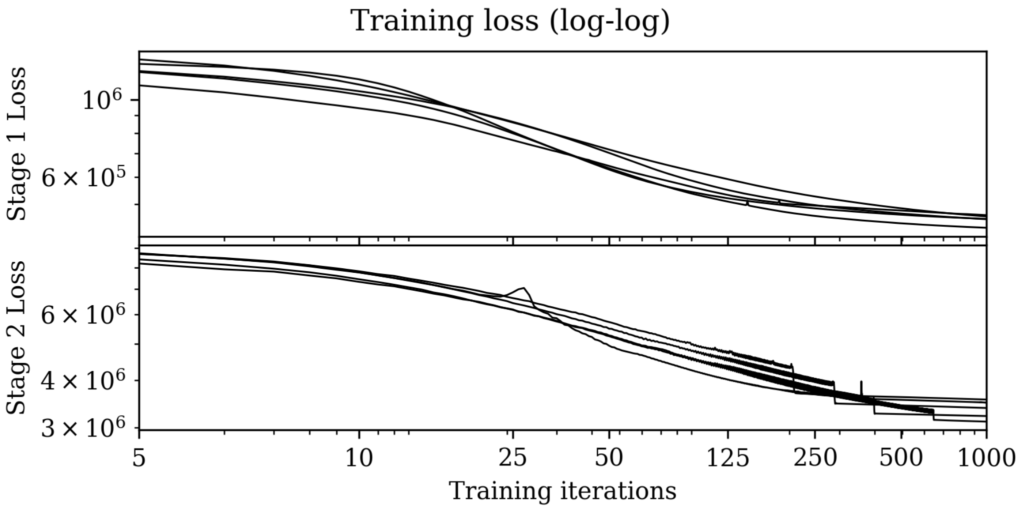


Figure 3.11: Plots of PathMatFac's training losses from five simulations. Recall that X and Y are fit in two stages: a first stage with L_2 regularization and another with ARD regularization. The top and bottom plots show losses from those stages, respectively. The step-like decreases during stage 2 result from the adaptive learning rate described in Section 3.2. Losses during stage 2 are larger since the ARD regularizer loss takes higher numeric values.

other problem attributes fixed; with M=1,463 and N=4,108 as before, and K=25 (both true and modeled).

Figure 3.10 shows the results for these runs. Parameter recovery scores do increase with the number of training iterations, but tend to plateau quickly. The scores degrade strongly only with an unrealistically small number of training iterations. We say "unrealistic" because, on these problems, the loss is still rapidly decreasing at 250 iterations. It would have been unreasonable to expect a good fit any earlier than that. Figure 3.11 demonstrates this with plots of training loss from five simulations.

This experiment provides some useful takeaways. PathMatFac does not need to meet a tolerance criterion (i.e., absolute or relative decrease in loss) to achieve a good fit. Instead, the number of iterations can be chosen to satisfy some other consideration, e.g., a *time budget*, without fear of a terrible fit. We employ a default of 1,000 training iterations for the experiments in this chapter, unless specified otherwise. We recommend increasing the number of iterations for larger datasets.

At this point it's worth observing that the span similarities for X are consistently higher than those for Y. We hypothesize that this is caused by Y's ARD regularization, in at least two ways. First, since Y is regularized to be sparse, its rows will be less capable of spanning arbitrary subspaces of $\mathbb{R}^N$. Second, it's also possible that the *nonconvexity* of ARD tends to trap Y at local optima with lower span similarities. We also observe that the *variances* for Y's span similarities are higher than those of X. This, too, could be explained by the nonconvexity of ARD; different local optima could yield distinct span similarities, resulting in greater variance. However, any disadvantage posed by nonconvexity is sufficiently innocuous that X is recovered with high fidelity—PathMatFac *embeds* the data correctly.

Robustness to missing values. PathMatFac is highly robust to missing values in the data. Figure 3.12 shows results from simulations with varying fractions of missing values. The data simulator imposes missing values in each view by selecting entire *batches* of samples and setting them to NaN. This resembles patterns of missing values in real data more faithfully than, for example, selecting individual entries uniformly at random. It also creates a more challenging inference task for PathMatFac, since it causes entire modalities of data to go missing for each sample.

The recovery of X, Y, and A shows very little degradation between (*i*) scenarios with no missing values and (*ii*) scenarios with half of the entries missing. The simulations are limited to fractions up to 0.5 only for technical reasons related to the data simulator. Namely, the current scheme for imposing missing values tends to produce entire *rows* of missing values when the fraction exceeds 0.5. A more sophisticated scheme would allow us to explore higher fractions.

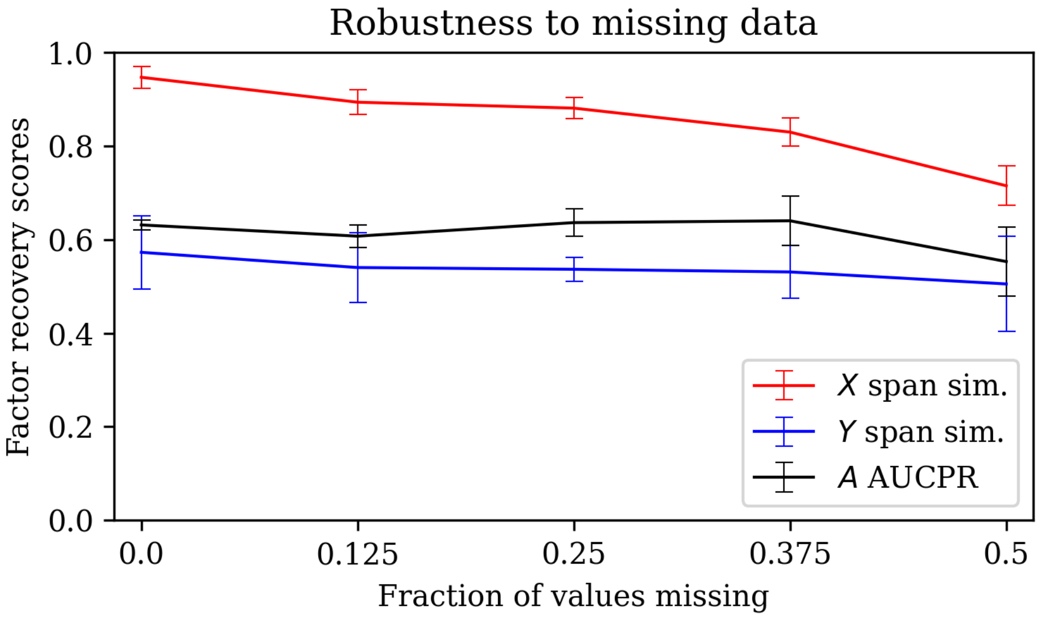


Figure 3.12: PathMatFac's parameter recovery scores on simulated data, as the fraction of missing values in the dataset increases.

Dependence on problem size. We can also use simulations to explore how PathMatFac's performance varies with (*i*) the number of samples and (*ii*) the number of features. Figure 3.13 shows results from a suite of simulated problems of different sizes. In these simulations, we hold K=25 fixed, both true and modeled. Each of the scores degrade noticeably when M becomes very small. Recovery of X improves as N increases. Recovery of A appears to improve as both M and N increase. Beyond that, patterns become less clear. It seems plausible that the recovery scores for X and Y tend to *plateau* when M and N become sufficiently large. With obvious caveats about the validity of simulations, it seems adviseable to use PathMatFac on datasets with M in the low hundreds. For smaller problems, PathMatFac should be run with smaller K and its output should be considered less reliable.

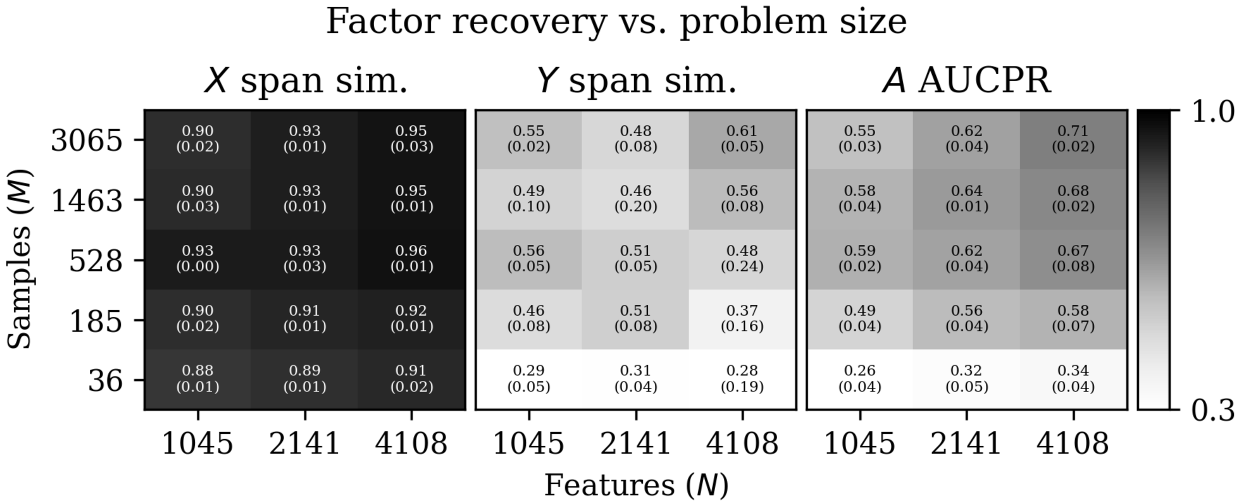


Figure 3.13: Sensitivity of PathMatFac to problem size. Each grid shows a parameter recovery score as in 3.9. However, in this case the horizontal axis scans across numbers of features and the vertical axis scans across numbers of samples.

	N=2, 060	N=4, 124
M=1, 463	750s 3.8GB	1,623s 3.3GB
M=528	291s 2.34GB	534s 2.7GB

Table 3.3: Computational expense of PathMatFac for varying problem sizes. We provide total execution time and peak memory usage. In each test, PathMatFac is configured with K=25 and 1,000 training iterations. Each test used a single Intel i7 3.00 GHZ CPU.

Computational expense. PathMatFac entails moderate computational expense when run on CPU. Table 3.3 shows total training time and peak memory usage for several problem sizes. PathMatFac uses K=25 factors and 1,000 training iterations in each of these tests. The tests use a single CPU with no multithreading. A factorization problem with ~4,000 multiomic features and ~1,400 samples requires ~27 minutes of execution time. Holding the number of training iterations fixed, time expense scales with the size of the problem $M \cdot N$ to a reasonable approximation. Peak memory usage is more puzzling. The top row of the table shows that it is not always monotonic with problem size. This likely results from internal details of Julia's memory management.

This computational expense is low enough to be practical. However, it's possible that additional code optimizations and multithreading would yield improved time and memory costs. Code profiling shows that batch parameters and ordinal loss functions create much of the time expense.

Batch effect estimation. Simulations demonstrate that the EM procedure outlined in Section 3.2 is quite effective at estimating shift (θ) and scale (δ) batch effects consistent with the model. Figure 3.14 shows results from a representative simulation, configured with M and N as above. Batch effects were only introduced and modeled for (simulated) methylation and RNA-seq features. Each scatter plot compares *true* parameter values (horizontal axis) against *estimated* parameter values (vertical axis).

Estimates for θ tend to lie near the diagonal, exhibiting a strong correlation with the true values. Estimates for δ tend to be *ordered* correctly, though they exhibit a noticeable bias toward 1. The EM procedure visibly squeezes estimates *within* a batch toward each other, producing the horizontal banding patterns. This is especially clear for the δ estimates. Exceptions exist, though. In Figure 3.14, the EM procedure fails to reduce the variance of one batch in the RNA-seq data (shaded pink).

We quantify PathMatFac's performance at estimating θ via the coeffi-

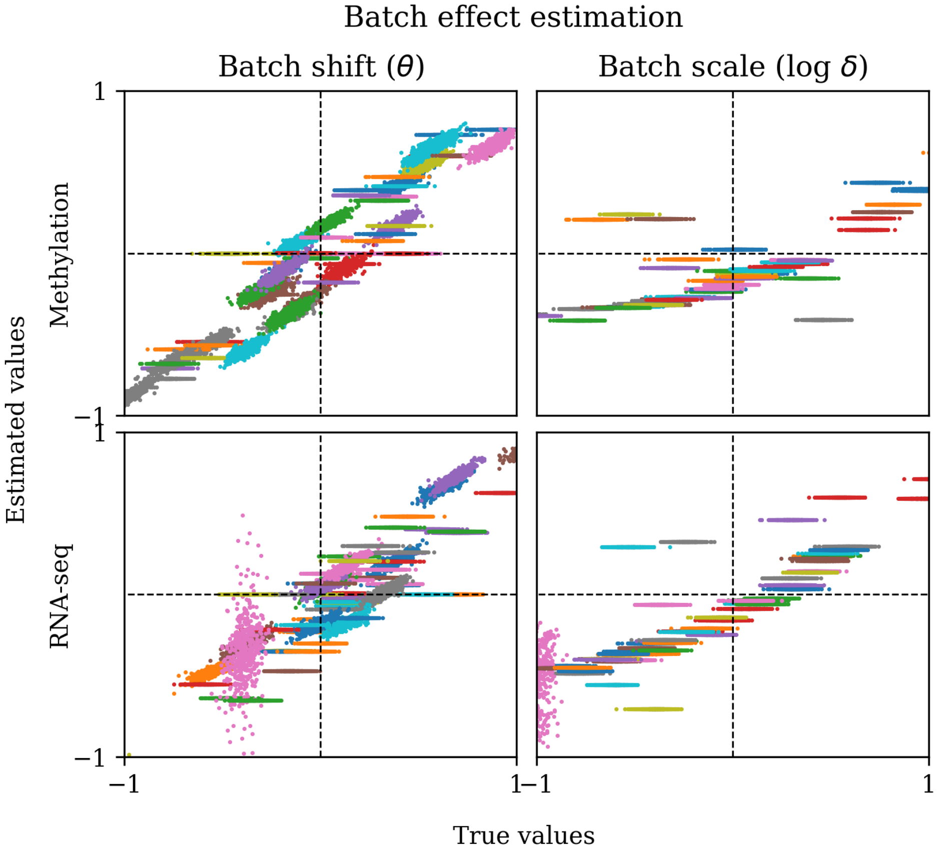


Figure 3.14: Comparison of true and estimated batch parameters from a representative simulation. Left column shows estimates for θ; right column shows the logarithm of the estimates for δ. Colors indicate batch membership (with some colors repeated due to a limited color palette). Estimates would ideally lay on the diagonal.

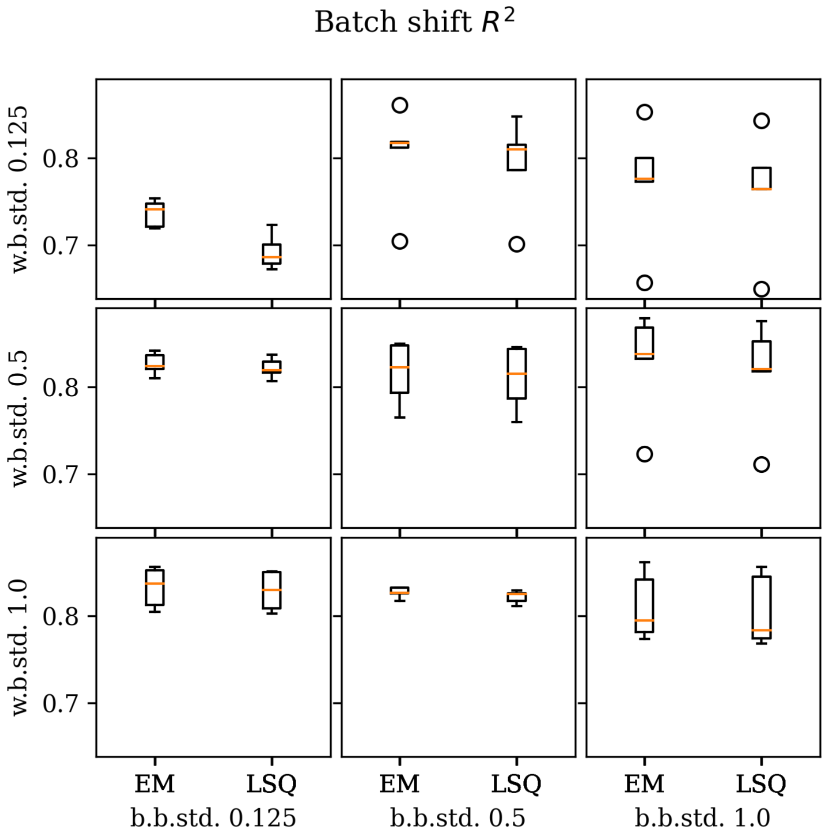


Figure 3.15: Comparison of batch shift (θ) estimation performance on simulated data, between the expectation-maximization (EM) procedure and simple least-squares estimates (LSQ). Performance is measured by coefficient of determination R^2 between the estimated (θ_{Fitted}) and simulated (θ_{True}) values. Simulations differ by the amount of *within-batch* standard deviation and *between-batch* standard deviation used to generate θ_{True}, corresponding to the rows and columns in this figure. Each boxplot summarizes five simulations. "b.b.std": between-batch standard deviation; "w.b.std": within-batch standard deviation.

cient of determination (R^2) between simulated and fitted values. Figure 3.15 shows the results from a grid of simulations. The simulations differ in the amount of *within-batch* standard deviation and *between-batch* standard deviation used when sampling θ_{True}. We use simple least-squares estimates as a baseline of comparison. The EM procedure yields a modest though reliable improvement over least-squares in these simulations. The improvement is most noticeable when the within-batch and between-batch standard deviations are both small.

Quantifying embedding quality via prediction tasks

Simulations serve as a helpful sanity check. However, the more interesting question is whether PathMatFac does anything useful with *real* datasets.

PathMatFac aims to embed multiomic data in an informative fashion. We want to measure PathMatFac's performance as an embedding technique on real data; however, directly quantifying the performance of a dimension reduction can be challenging. Instead, we score PathMatFac by applying it to supervised prediction tasks. The idea is to use PathMatFac as a dimension reduction technique in conjunction with a strong supervised learner (e.g., a random forest), to predict labels from the multiomic features. If the labels are grounded in biology, then a better dimension reduction (i.e., embedding of the data) should yield better predictive performance. This is a somewhat indirect evaluation, but it allows quantitative comparisons between PathMatFac and other omics embedding techniques.

We emphasize that the goal of this evaluation is *not* to find the best supervised method for multiomic datasets. Many supervised models already exist for multiomic data, and PathMatFac does not aim to compete with them. Instead, the goal is to evaluate the *quality of embeddings*. An informative embedding can be used for a variety purposes—clustering, statistical comparisons, or other analyses—in addition to predictions.

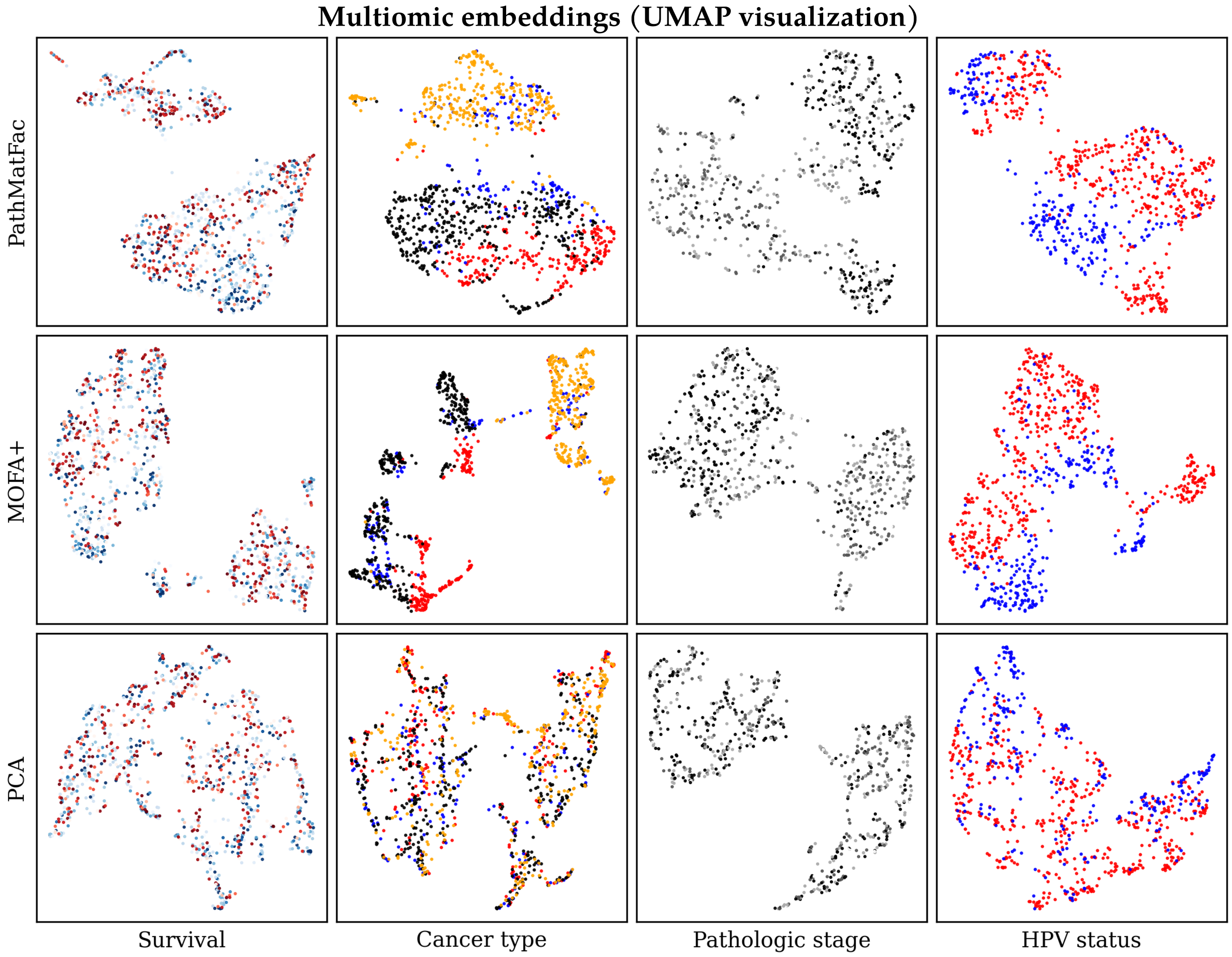


Figure 3.16: Comparison of multiomic embeddings produced by different techniques on the {HNSC, CESC, ESCA, STAD} subset of TCGA. Only PathMatFac, MOFA+, and the PCA baseline can be readily applied to multiomic data. The embeddings are 25-dimensional; this figure uses UMAP to visualize them in 2D. Each column colors the scatterplot with labels from a different prediction task. The key takeaway: in some cases the embeddings clearly separate the labels, but not in others.

Prediction tasks. The prediction tasks use a *subset* of the full TCGA dataset shown in Figure 3.1—specifically, tumors belonging to head and neck (HNSC), cervical (CESC), esophageal (ESCA), and stomach (STAD) cancers, for a total of M=1,463 samples. We are restricted to the {HNSC, CESC, ESCA, STAD} subset because those samples possess prediction labels of interest. Samples from TCGA are labeled with a variety of clinical indications, but many of the samples are missing labels. We identified four clinical labels that seemed plausibly correlated with omics measurements from tumor samples: cancer type, survival, pathologic stage, and HPV infection status. In contrast, labels like age or sex seemed less credible or interesting as prediction targets. Our evaluations use the {HNSC, CESC, ESCA, STAD} subset *precisely* because those samples possess all four of these labels.

The experiments include scenarios with differing sets of *features*. These comprise (*i*) a full multiomic scenario that uses somatic mutation, methylation, RNA-seq, and CNA features; (*ii*) a partial multiomic scenario that keeps RNA-seq and CNA features; and (*iii*) a scenario that uses only RNA-seq features. Only a small fraction of the available features are used in each scenario. Features are selected by (*i*) discarding columns without a sufficient number of observations; (*ii*) computing the variances of the remaining columns; and (*iii*) for each omic assay, selecting the top-p% highest-variance features. The value of p varies between scenarios, such that the total number of features remains approximately the same: 5% for the full multiomic scenario; 10% for the RNA-seq/CNA scenario; and 20% for the RNA-seq scenario, for a total of $\simeq$ 4,000 features in each scenario.

Different labels call for different scores. We treat HPV status prediction as a binary classification task, and score it with AUCROC. Cancer type prediction is multi-class classification with a reasonably balanced class distribution, so classification accuracy is an informative metric. Pathologic stage is treated as a regression task; stages (*i*) through (*iv*) are encoded as integers, and performance is measured by mean squared error (MSE). Survival prediction uses *concordance* as a performance metric. Concordance is a *ranking* metric closely related to AUCROC, capable of handling censored observations in survival data (Terry M. Therneau and Patricia M. Grambsch, 2000). As in AUCROC, trivial predictors have concordance 0.5 and perfect predictors attain concordance 1.0.

All prediction tasks use five-fold cross-validation to estimate predictive performance. For each prediction task we prepare five splits of the dataset. Splits are *stratified* by the label, and *grouped* by batches in the RNA-seq data.

All tasks use random forests with 500 trees for prediction. Classification and regression tasks use SCIKIT-LEARN's random forest implementations (Pedregosa et al., 2011). Survival tasks rely on the R package RANDOMFORESTSRC (Ishwaran and Kogalur, 2007).

Baselines of comparison. Many techniques already exist to summarize or reduce the dimension of omics data. We compare PATHMATFAC's dimension reduction performance against several baseline methods: PCA, MOFA+, PLIER, PARADIGM, and GSVA. We also compare against the original *raw* omics features, filtered by variance but with no other dimension reduction.

Principal components analysis (*PCA*) (Hotelling, 1933). PCA is widely used for summarizing and visualizing omics data. For supervised tasks on omics data, PCA is known to be a strong baseline dimension reduction despite its simplicity (Makrodimitris et al., 2023). We apply PCA to multiomic data by (*i*) computing principal components separately for each modality and (*ii*) concatenating the resulting view-specific embeddings. We let each modality contribute an equal number of dimensions to the combined embedding. This is not generally optimal, but it is a simple rule that allows straightforward comparisons to other techniques. That is, it lets us compare a K-dimensional PCA embedding of multiomic data against a K-dimensional PathMatFac embedding of multiomic data.

Multi-Omic Factor Analysis 2 (*MOFA+*) (Argelaguet et al., 2020). MOFA+ is the latest in a family of Bayesian matrix factorization methods intended for multiomic data. MOFA+ uses variational inference and a combination of priors to infer *sparse* linear factors for a multiomic dataset. This sparsity helps make the factors interpetable—each factor has a small number of nonzero entries which can be visually inspected. It's important to mention that the MOFA+ software does *not* include a method to embed (i.e., transform) new samples of data. For the purposes of our evaluation, we use a naïve least-squares procedure to project test-set samples on to MOFA+'s fitted factors. Anything more sophisticated would be beyond the scope of this project. However, it's worth noting MOFA+'s performance in this evaluation is, to some degree, hampered by its lack of a transform function. Its scores do not necessarily capture the quality of its embeddings.

Pathway Level Information ExtractoR (*PLIER*) (Mao et al., 2019). PLIER is a matrix factorization technique intended for Gaussian datasets (especially log-transformed RNA-seq). It uses curated gene sets to inform *and* interpret the factorization. Specifically, PLIER (*i*) assumes the factors are themselves generated from linear combinations of gene sets; and (*ii*) alternates between fitting the factors and updating the gene set combinations until convergence. PLIER is a primary inspiration for PathMatFac. PathMatFac can be regarded as a multiomic improvement of PLIER—though their probabilistic assumptions do differ in meaningful ways. We only compare against PLIER on RNA-seq prediction tasks, since PLIER is not suited for multiomic datasets.

PARADIGM (Vaske et al., 2010a). PARADIGM uses biological pathways to construct a factor graph model for multiomic data. Then, for each sample, it uses this factor graph to produce smoothed estimates of "activity levels" for pathway entities (e.g., genes, and proteins). The activity levels can then be used for downstream analyses. TCGA made extensive use of

PARADIGM for its analyses of multiomic data. The PARADIGM software is somewhat difficult to acquire and use; instead, we use the precomputed outputs provided by Hoadley et al. (2018). Their outputs were computed from RNA-seq and CNA data. Accordingly, our experiments only compare against PARADIGM on the partial multiomic RNA-seq/CNA tasks.

Gene Set Variation Analysis (*GSVA*) (Hänzelmann et al., 2013). GSVA is a gene set enrichment technique similar to GSEA. However, it differs from GSEA in that it produces enrichment scores for individual samples in a dataset, rather than making population-level comparisons. In the parlance of Section 3.1, GSVA represents the *biology-centric* methods. Our experiments only compare against GSVA on the single-omic RNA-seq tasks, since it's not intended for multiomic datasets. GSVA employs *cancer hallmark* and *oncogenic* gene sets curated by MSigDB in this evaluation.

Recall that we aim to measure PathMatFac's ability to produce an informative embedding of multiomic data. In order to create fair comparisons between techniques, in every task we configure each technique to yield an embedding of equal dimension K=25. This is consistent with recommended settings for PLIER and MOFA+ (Mao et al., 2019; Argelaguet et al., 2020). PARADIGM and GSVA are not designed to produce embeddings of a given dimension, so we use them in conjunction with PCA to attain the desired K=25 embedding dimension.

Results from multiomic prediction tasks. Prediction tasks on multiomic data confirm that PathMatFac produces informative embeddings. Figure 3.16 visualizes examples of embeddings produced by PathMatFac, MOFA+, and PCA on multiomic data in these tasks. The K=25-dimensional embeddings are visualized via UMAP. Colors indicate labels for the prediction tasks. The most important takeaway from Figure 3.16 is that, in some cases, the embeddings visibly capture variation related to the prediction labels. For example, PathMatFac and MOFA+ both produce embeddings that contain information about cancer type and HPV infec-

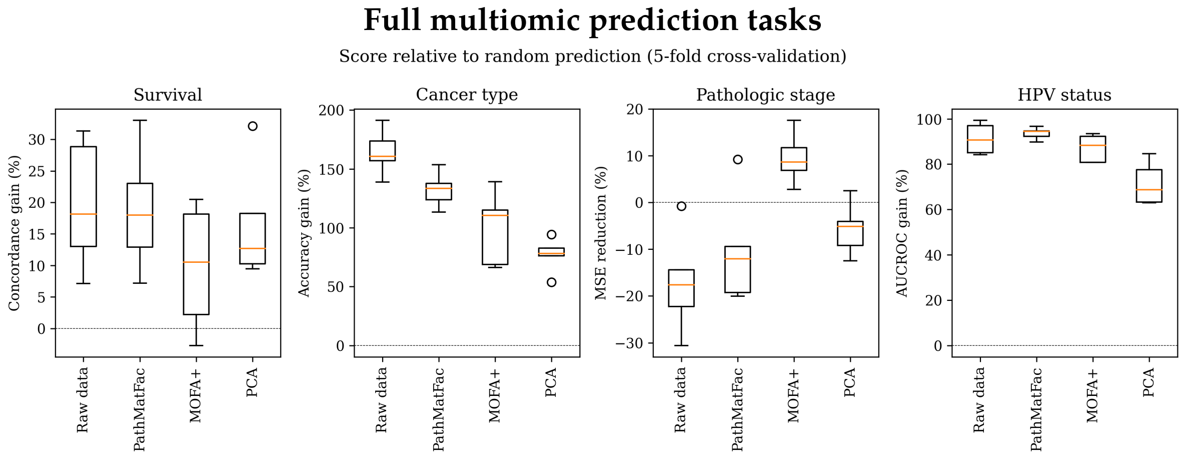


Figure 3.17: Predictive performance for different multiomic embedding techniques, relative to a trivial predictor. Higher is better *except* in the case of pathologic stage, which compares mean-squared error (MSE). Prediction on the raw data outperforms prediction on dimension-reduced data on most tasks, suggesting that each of these embeddings is lossy.

tion status, visible to the eye. Other cases are less obvious, like survival. MOFA+ appears to separate pathologic stages in this figure but performs poorly in the evaluation; this could be a result of its ad-hoc method for transforming test samples.

Figure 3.17 shows prediction performance in the full multiomic scenario for PathMatFac, MOFA+, and PCA for all four prediction labels. It also shows prediction performance using the raw, untransformed multiomic data. Predictions on the raw data tend to score better than predictions on the embedded data, suggesting that each of these dimension reductions is lossy. A possible exception is HPV infection status, where PathMatFac attains slightly higher average AUCROC with much lower variance.

PathMatFac tends to yield better average scores than the other dimension reduction techniques, though the magnitude of improvement differs between tasks. For instance, PathMatFac attains similar scores to the other methods on survival prediction, but clearly dominates them on cancer type prediction. We also see that the prediction tasks vary in difficulty. None of the dimension reductions yield strong predictions for survival or pathologic stage, but all of them predict cancer type and HPV status much better than random. We mention in passing that the ability to predict cancer type and HPV status from multiomic data is *not* clinically useful, since reliable (and inexpensive) diagnostics already exist for those indications. These tasks only serve to create quantitative comparisons of embedding quality.

An informative embedding suggests that PathMatFac's linear factors capture predictive signals in the multiomic data. We can safely conclude that PathMatFac succeeds at factorizing multiomic data in an informative way.

PathMatFac's performance as a multiomic dimension reduction, relative to other matrix factorizations, suggests that its factors are more predictive than theirs. Of course, this is subject to caveats about MOFA+ lacking a transform method. In any case, we can confidently claim that PathMatFac competes strongly among multiomic matrix factorizations.

Results from (RNA-seq, CNA) prediction tasks. Figure 3.18 shows results from an analogous set of prediction tasks, this time using only RNA-seq and CNA omics modalities. This allows us to make comparisons against PARADIGM, a classic technique that only accommodates those modalities. We see that dimension-reduction via PARADIGM competes very strongly in all of these tasks, outperforming the other dimension reductions (though still lagging predictions on raw data).

This comparison against PARADIGM may not be completely fair. Recall that we employ a variance-filtering feature selection step prior to dimension reduction. In contrast, the archived PARADIGM outputs are generated from *all* available RNA-seq and CNA features. Hence, the PARADIGM-based embedding is informed by more features than the other embeddings.

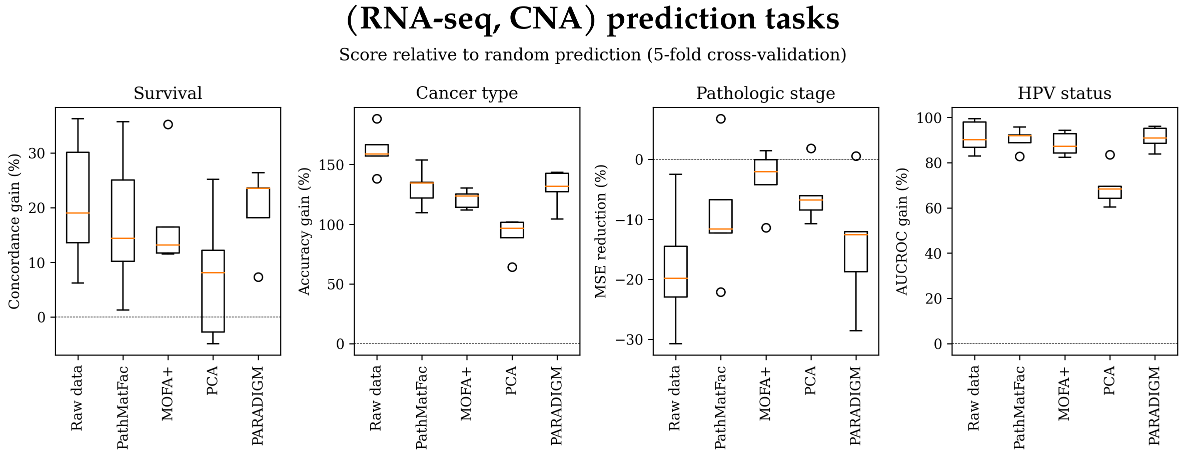


Figure 3.18: Predictive performance for different embedding techniques on RNA-seq and CNA multiomic features, analogous to figure 3.17. PARADIGM is a competitive baseline, though it still lags prediction on the raw data in most tasks.

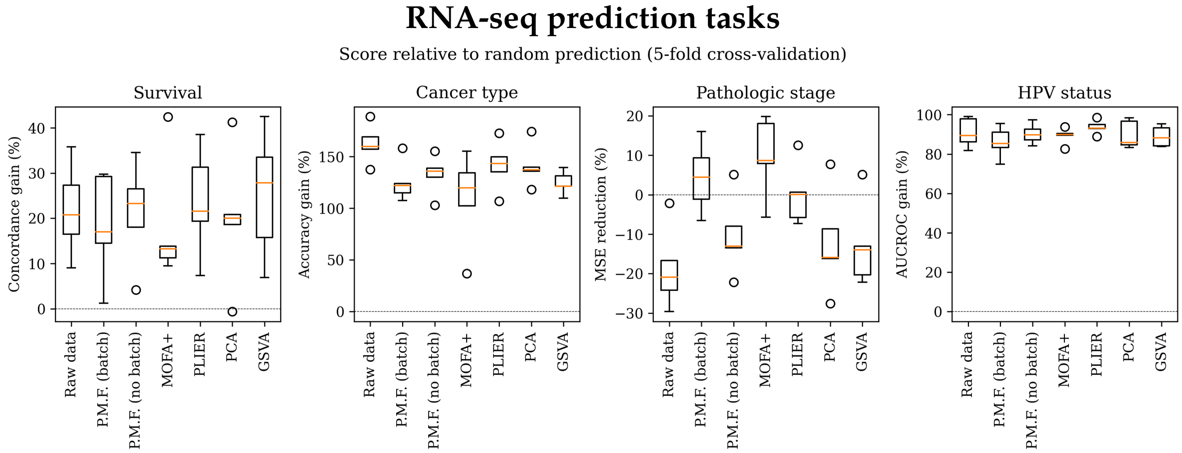


Figure 3.19: Comparison of predictive performance for RNA-seq embedding techniques. "P.M.F. (batch)": PathMatFac, with batch effect modeling; "P.M.F. (no batch)": PathMatFac, *without* batch effect modeling. Batch effects contain (non-biological) predictive signal, so PathMatFac's performance lags other techniques' when it models them away. However, the performance gap decreases when PathMatFac does *not* account for batch effects.

Results from RNA-seq prediction tasks. Prediction tasks on RNA-seq data tell a more subtle story. Batch effects are an especially important phenomenon in RNA-seq data. In the TCGA dataset, batches depend on cancer type, which correlates with the prediction labels. It follows that batch effects contain artificial predictive signal in this task; when PathMatFac models them away, then it loses predictive power. Figure 3.19 compares prediction scores between PathMatFac and baseline techniques, similar to Figure 3.17. PathMatFac's predictive performance noticeably lags other techniques when it models away batch effects. However, its performance is on par with the others' when, like the rest, it exploits the predictive signal in batch effect.

Comparisons between the multiomic and RNA-seq scenarios (i.e., Figures 3.17 and 3.19) are also instructive. The total number of omic features is approximately the same for the multiomic scenario and the RNA-seq scenario. However, PathMatFac (with batch effect modeling) attains better predictive performance in the multiomic scenario—this is especially apparent for cancer type and pathologic stage prediction. This suggests that modalities other than RNA-seq contain important predictive signal for those labels, and that PathMatFac can exploit that signal better than other techniques.

GSVA is worth highlighting since it's the only *biology-centric* technique in these experiments. Recall that this evaluation uses GSVA in conjunction with PCA to (*i*) compute gene set enrichments and (*ii*) reduce dimension to K=25. GSVA attains scores roughly on par with PCA—with the exception of survival prediction, where it attains a higher average concordance. This suggests GSVA's gene set enrichments probably do not expose *additional* predictive signal beyond that available in the RNA-seq data.

PathMatFac is an effective dimension reduction for RNA-seq data. However, it doesn't necessarily outperform existing RNA-seq factorization techniques. PathMatFac *does* stand out when applied to multiomic data, which was its intended purpose.

Exploring multiomic TCGA data with PathMatFac

We illustrate PathMatFac's utility for exploring multiomic datasets by applying it to data from TCGA and examining its outputs. PathMatFac identifies linear factors that explain variance in (*i*) individual modalities and (*ii*) combinations of modalities. Its FSARD mechanism then summarizes the linear factors in terms of curated gene sets, to the extent possible.

Exploring the {HNSC,ESCA,CESC,STAD} subset. We first inspect PathMatFac's outputs for the same multiomic dataset used in the supervised tasks. The supervised tasks show that PathMatFac learns *informative* representations of the data. However, we have yet to see if the model parameters offer any human-interpretable insights.

A full discussion of the linear factors' biological plausibility is out of scope for this chapter—biologists are better-equipped to assess them. Instead, we highlight some of the factors' interesting properties and hypothesize about their biological basis.

Figure 3.20 shows the matrix Y, fitted on the {HNSC, CESC, ESCA, STAD} TCGA samples. PathMatFac employs the same model configuration as before, with K=25 and 1,000 training iterations. The factors clearly possess structure and exhibit sparsity. Many factors explain variance in a single omics modality, while others span dimensions from multiple modalities. For example, factor 1 has notable components belonging to mutation, methylation, and RNA-seq modalities. The scree plot in figure 3.21 visualizes this another way, by showing the squared norms of linear factors and decomposing them into modality-specific contributions.

Some linear factors are mostly devoted to variation in CNA features. A close examination of factors 2, 5, 7, and so on, reveals that their nonzero entries *partition* the CNA features. Since copy number alteration tends

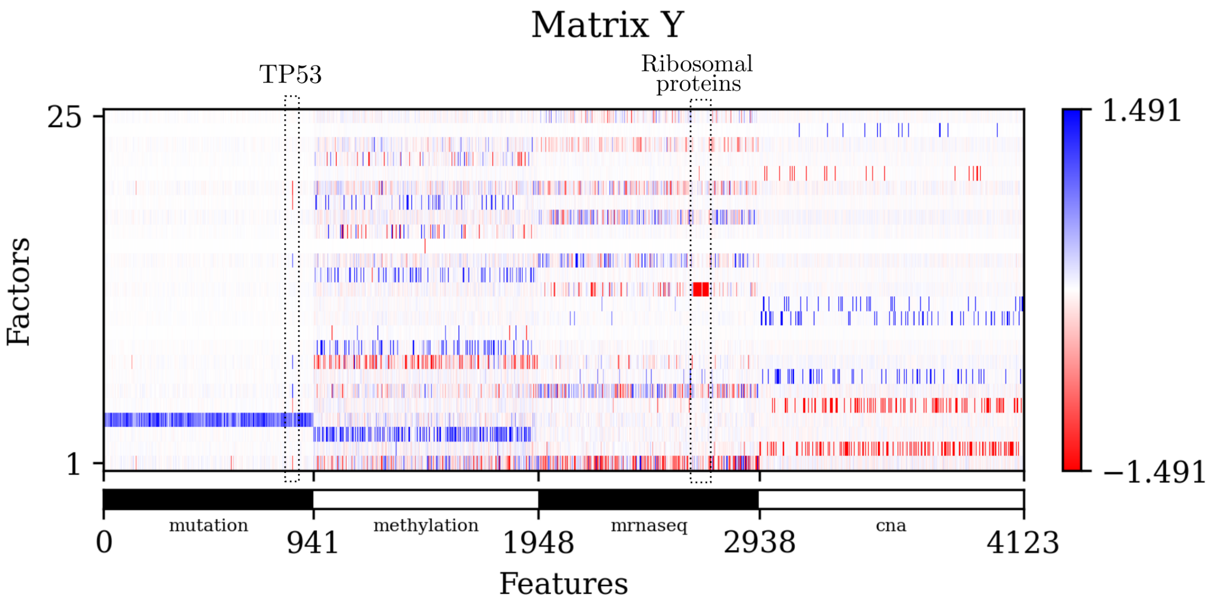


Figure 3.20: Heatmap of the matrix Y, estimated from the {HNSC, CESC, ESCA, STAD} TCGA subset. Recall that each row is a linear factor. Rows are ordered from largest norm (at the bottom) to smallest norm (at the top). Bear in mind that in matrix factorization, the signs of factors do not matter.

to simultaneously affect contiguous portions of the genome, we might expect CNA to be highly correlated between genes neighboring each other on chromosomes. Later, we will see this confirmed by FSARD's factor interpretations.

Figure 3.20 also highlights other details. TP53 is perhaps the most well-known *tumor suppressor* gene, with mutations occurring in many cancers (Olivier et al., 2010). Several of PathMatFac's linear factors (e.g., factors 1, 5, 6, and 8) possess a TP53 component and show joint variation with other omics modalities.

Gene expression (i.e., RNA-seq features) for ribosomal proteins exhibits some of the greatest variation in this dataset. However, factor 13

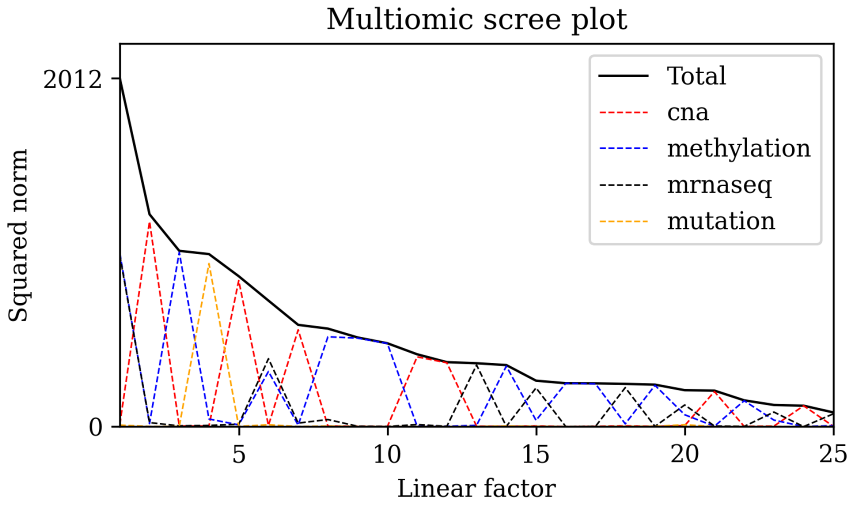


Figure 3.21: Scree plot for the same factors shown in Figure 3.20. The plot shows contributions from different data modalities. Most factors explain variance in a single modality, though there are notable exceptions.

captures nearly all of that variation and shows that the ribosomal protein gene expression varies together. Since a complete ribosome is composed of these proteins, we would expect their abundances to be strongly correlated. Gene expression variation for these genes may be of biological interest, or it may be regarded as a nuisance to ignore. In either case, the phenomenon is almost completely captured in a single factor.

Figure 3.22 visualizes six of the linear factors in greater detail and highlights other interesting properties. Factor 4 is unique in that it focuses almost entirely on somatic mutation features. Its mutation components all have the same sign, suggesting that it encodes widespread *mutational burden* in the biological samples. Factor 4's largest "spike" is on TTN, the largest known protein-coding gene in the human genome (Chauveau et al., 2014). This would be consistent with a "mutational burden" interpretation—we expect longer genes to be mutated more often, by random chance. Figure 3.23 shows X, the embedding matrix. Inspecting its

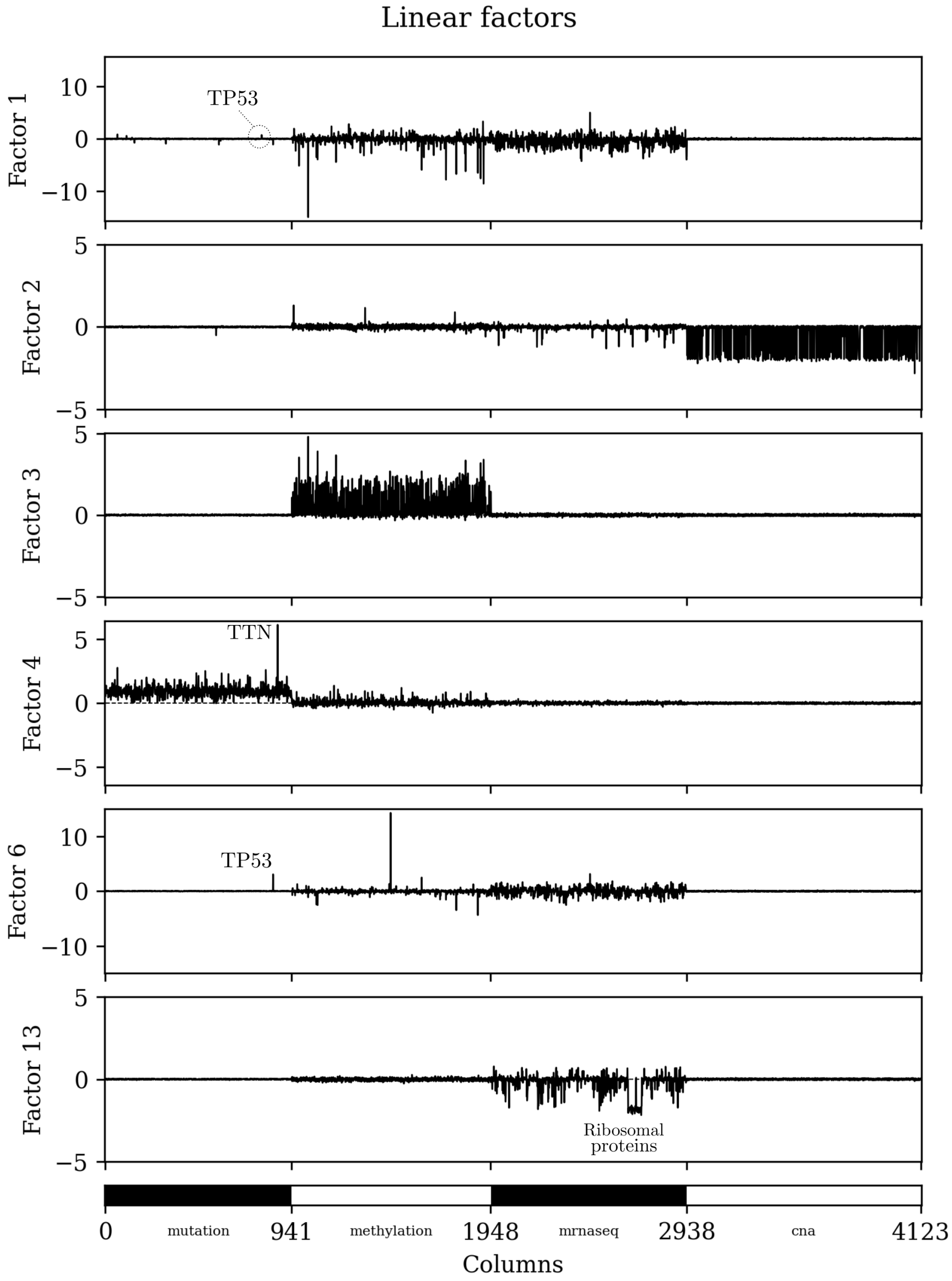


Figure 3.22: Line plots of some linear factors from the TCGA subset.

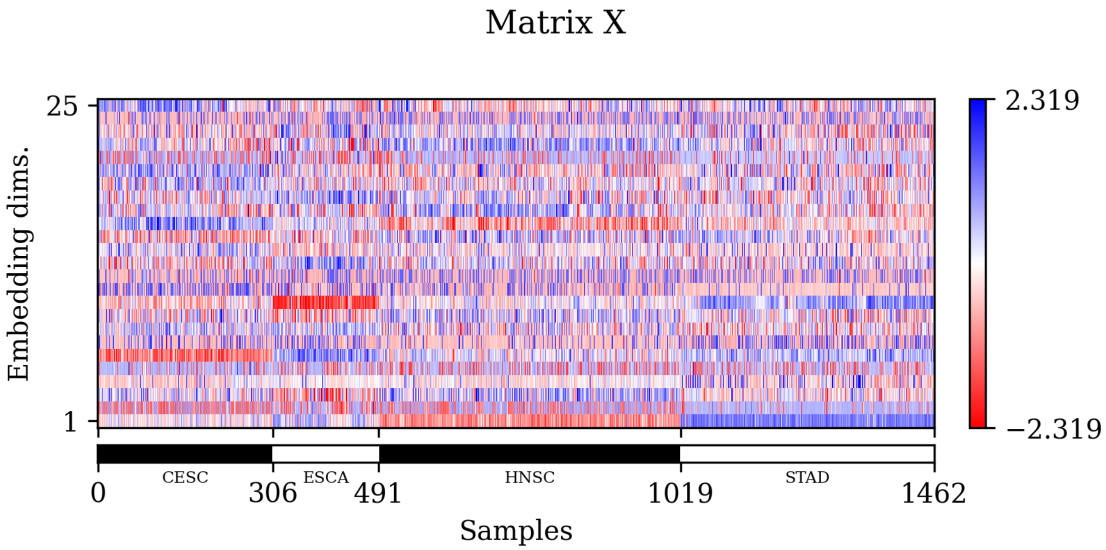


Figure 3.23: Heatmap of the embedding, X, estimated from the TCGA subset.

4th row reveals that most samples have very little contribution from factor 4, with notable exceptions in the STAD samples. This is consistent with other observations that stomach adenocarcinomas exhibit relatively high mutation frequencies in the TCGA dataset (Lawrence et al., 2013).

Figure 3.24 shows the fitted column shift and scale parameters, μ and σ. For the most part, these do not have interesting biological interpretations. However, column shifts for Bernoulli features do reveal the columns with higher rates of positive outcomes. I.e., peaks in μ for somatic mutation data show the genes that are most frequently mutated in the dataset. In Figure 3.24, these are TP53 and TTN.

Figure 3.25 shows the estimated batch shift and scale parameters. Recall that batch parameters are only estimated for Gaussian features, in this case the methylation and RNA-seq features. The batch shift parameters (θ) tend to take values grouped near zero, though in some batches of RNA-seq they vary much more. Most of the batch scale parameters (δ) are grouped

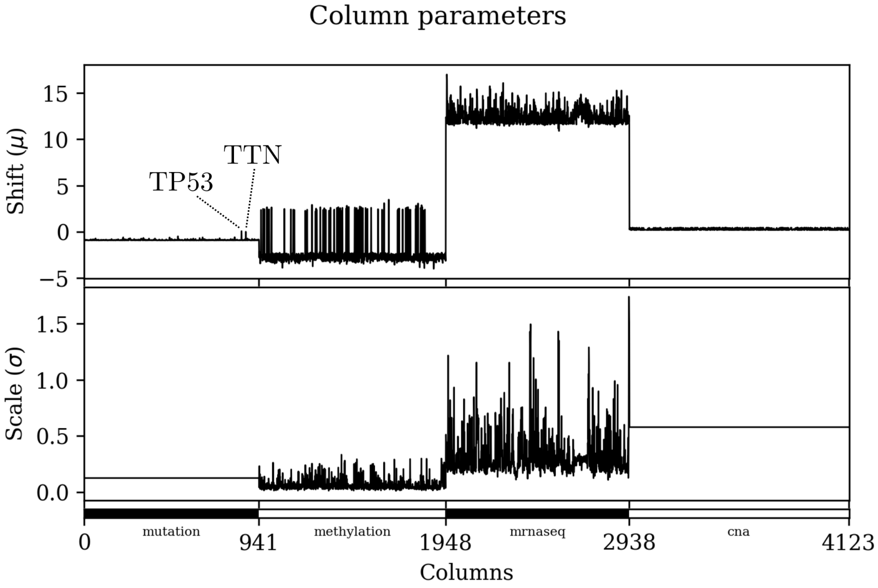


Figure 3.24: Visualization of column parameters μ and σ, estimated from the TCGA subset.

near one. However, in some batches of methylation data the estimated δs take much larger values. Likewise, in a small number of RNA-seq batches the estimated δs are quite distant from one. We also see the δ estimates group together much more tightly within batches for the RNA-seq data, than for the methylation data.

Finally, we inspect FSARD's interpretations of the linear factors in Table 3.4. The PathMatFac software outputs a table of interpretations; each row of the table indicates the assignment of a gene set to a factor. Table 3.4 shows an abbreviated summary of that output, for this problem instance.

The quality of FSARD's interpretations depends in part on the *choice* of gene sets. It's instructive to look at the first two factors from this example. For factor 1, FSARD assigns (*i*) 19 gene sets to the methylation view and (*ii*) 125 gene sets to the RNA-seq view. The weights of these assignments (shown in the right-most column) decrease in a gradual fashion, without

Factor	View	Gene set	Coeff.
1	Methylation	STK33_DN	0.67
		STK33_NOMO_DN	0.59
		JNK_DN.V1_UP	0.47
		PRC2_EED_DN.V1_DN	0.45
		HALLMARK_PI3K_AKT_MTOR_SIGNALING	0.43
		(14 more rows)	
	RNA-seq	HALLMARK_MYC_TARGETS_V1	1.83
		HALLMARK_MITOTIC_SPINDLE	1.77
		HALLMARK_APICAL_JUNCTION	1.68
		HALLMARK_MTORC1_SIGNALING	1.57
		CAMP_UP.V1_DN	1.51
		HALLMARK_P53_PATHWAY	1.46
		(119 more rows)	
2	CNA	chr3q26	2.61
		chr3q25	2.57
		chr3q27	2.55
		chr3q29	2.53
		chr3q23	2.27
		chr3q24	2.08
		chr3q28	2.02
3	Methylation	HALLMARK_E2F_TARGETS	1.43
		ERBB2_UP.V1_DN	1.40
		HALLMARK_UNFOLDED_PROTEIN_RESPONSE	1.26
		HALLMARK_GLYCOLYSIS	1.19
		RB_P107_DN.V1_DN	1.14
		(34 more rows)	
⋮	⋮	⋮	⋮
8	RNA-seq	HALLMARK_INTERFERON_GAMMA_RESPONSE	1.97
		HALLMARK_INTERFERON_ALPHA_RESPONSE	1.71
		PKCA_DN.V1_UP	0.77
		PTEN_DN.V1_DN	0.44
		BCAT.100_UP.V1_UP	0.11
		(2 more rows)	
⋮	⋮	⋮	⋮
25	RNA-seq	GCNP_SHH_UP_EARLY.V1_UP	0.96
		HALLMARK_DNA_REPAIR	0.86
		HALLMARK_PANCREAS_BETA_CELLS	0.60
		HALLMARK_MYC_TARGETS_V1	0.58
		NRL_DN.V1_DN	0.49
		(5 more rows)	

Table 3.4: Factor interpretations by FSARD, for the TCGA subset. Each row represents a nonzero entry in an FSARD assignment matrix, A. Rows are sorted by factor, view, and regression coefficient (i.e., the entry in A). Factor 1 captures variance in the methylation and RNA-seq features, but the curated gene sets summarize it poorly. In contrast, Factor 2 captures variance in the CNA features and is very neatly summarized by locational gene sets. Coefficients may be compared *within* a view, but not *between* views or factors. For reference, FSARD used a collection of 239 gene sets to summarize RNA-seq and methylation views in these results.

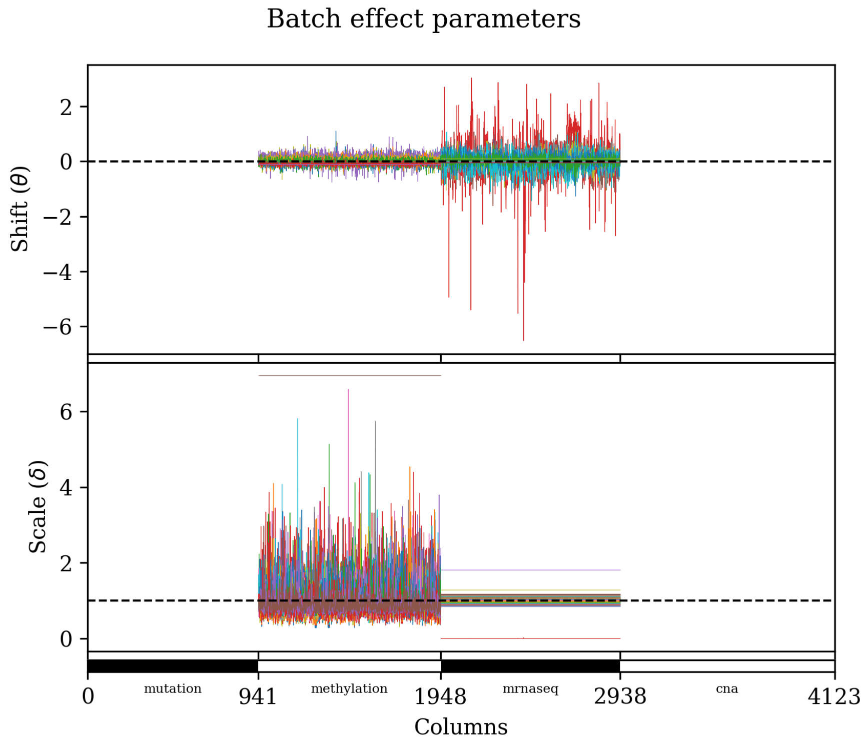


Figure 3.25: Visualization of batch parameters θ, δ, estimated from the TCGA subset. Color indicates batch (sometimes colors are reused due to the limited color palette).

a clear threshold. This suggests that the gene sets provided to FSARD were unable to *explain* factor 1 in a parsimonious fashion. Of course, it's also possible that factor 1 is too dense for *any* small number of sparse gene sets to summarize it.

In contrast, FSARD summarizes factor 2 with a total of 7 gene sets, all with similar weight. The fact that factor 2 can be neatly summarized with a small number of gene sets suggests that the gene sets were well-chosen. Furthermore, the assigned gene sets have a biologically plausible relationship to each other; namely, they represent copy number alterations of a neighborhood on chromosome 3 of the human genome. For these reasons, we put greater confidence on factor 2's interpretation than on that of factor 1.

The full table of FSARD interpretations for this dataset contains additional lessons. The case of factor 2 is not unique; in fact, *every* factor explaining variation in the CNA data is summarized parsimoniously by genomic-location gene sets. In contrast, most factors devoted to RNA-seq or methylation features are poorly captured by the provided gene sets. There are exceptions, though. For example, the RNA-seq components of factor 8 are well-summarized by two *cancer hallmark* gene sets from MSigDB:
HALLMARK_INTERFERON_ALPHA_RESPONSE and
HALLMARK_INTERFERON_GAMMA_RESPONSE.

Future versions of FSARD could use statistical hypothesis testing to filter out insignificant gene set assignments. This would be especially valuable when FSARD fails to produce a parsimonious interpretation. Simple permutation tests could accomplish this, though they would entail additional computational expense. The improved utility of FSARD's interpretations would likely be worth the added computational cost, though.

Exploring a large pan-cancer TCGA dataset. We now emphasize PathMatFac's ability to model *large* datasets by applying it to a much more comprehensive body of TCGA data. The following analyses employ a pan-cancer dataset of 9,369 samples belonging to 31 different cancer types. Each sample is represented by 16,459 features belonging to four distinct omic assays (somatic mutation, methylation, RNA-seq, and CNA). Contrast this against the {HNSC, CESC, ESCA, STAD} subset, which includes only 1,462 samples and 4,123 features. As before, missing values appear frequently.

This pan-cancer dataset contains the majority of tumor samples from TCGA, though we do exclude the glioblastoma multiforme (GBM) and ovarian carcinoma (OV) cohorts. At first we tried to include GBM and OV

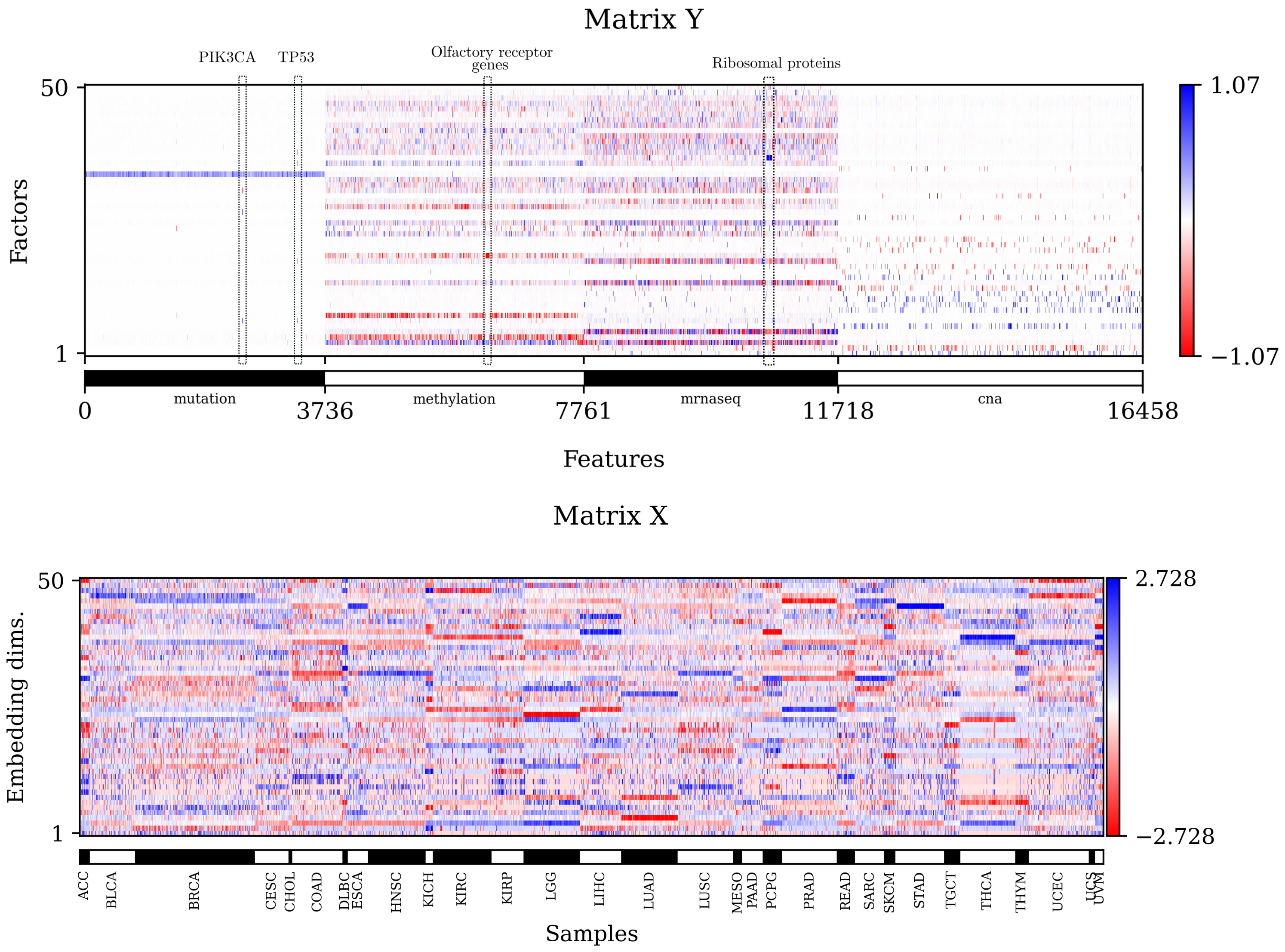


Figure 3.26: Linear factors (Y) and embedding (X), fit on the pan-cancer TCGA dataset. PathMatFac used K=50 factors in this run. As before, factors are ordered from largest norm (at the bottom of each heatmap) to smallest norm (at the top).

in the factorization, but their variance tended to dominate that of the other samples. Visual inspection of the data (as in Figure 3.1) reveals that the GBM and OV samples have many missing values in their RNA-seq features; and their methylation features differ from those of other subpopulations in noticeable, systematic ways. Given these issues around data quality, we choose to hold out the GBM and OV samples from this factorization.

We fit PathMatFac on this pan-cancer dataset for 3,000 training iterations, using K=50 linear factors. This takes approximately 5.5 hours on a consumer-grade NVIDIA RTX 2080 Ti GPU, possessing 11GB of memory. This modest computational expense is well within reach for many bioinformaticians.

As before, we configure FSARD to use *genomic-location* gene sets for CNA features. However, in this run we provide *gene ontology: biological process* gene sets to explain the other features; these gene sets also come from MSigDB (Liberzon et al., 2015). We also set a higher value of γ=2 for FSARD's hyperparameter. This penalizes spurious gene set assignments more strongly, ensuring that FSARD assigns a smaller number of better-fitting gene sets to the linear factors.

Figure 3.26 shows the results from running PathMatFac on the pan-cancer TCGA dataset. We focus especially on matrix Y, the linear factors. These bear some similarities to the factors fitted on the {HNSC, CESC, ESCA, STAD} subset. However, the inclusion of more features and a larger, more diverse population of samples does yield patterns not seen previously.

As before, the factors exhibit great sparsity. Most factors explain variation in one or two data modalities, with components near zero elsewhere. The factors are especially sparse for the somatic mutation features. As before, TP53 mutations stand out prominently—several factors possess nonzero components for them. However, the pan-cancer results also grant similar status to PIK3CA mutations. That is, a handful of factors have large nonzero components for PIK3CA, capturing joint variation between PIK3CA mutations and other features in the data. Similar to before, one linear factor (in this case factor 33) seems to encode widespread tumor mutational burden in the samples.

Recall that in the results for {HNSC, CESC, ESCA, STAD} we saw practically all of the gene expression for the *ribosomal protein* gene family captured by a single factor. The results on the pan-cancer dataset recapitulate this pattern. Furthermore, they identify a similar pattern in *methylation* for another large gene family—the *olfactory receptor* genes. A single factor, the 19th, captures nearly all of the variance in methylation for this family of genes.

PathMatFac's FSARD mechanism yields interesting interpretations for many of these linear factors. Table 3.5 shows some of them. As we saw previously, factors devoted to CNA features are concisely explained by locations in the human genome. For example, factor 1 explains copy number alterations in a neighborhood on chromosome 8.

Other factors are not easily summarized by gene sets. Factor 3 explains variation in RNA-seq features, and FSARD assigns many diverse gene sets to it. It's tempting to see a shared a theme related to cell cycle or metabolism in these gene sets; however, this should be regarded with some skepticism.

FSARD's interpretation of factor 19 jives well with our visual observations of the matrix Y. Most of the gene sets assigned to factor 19 relate to chemical sensing or detection, consistent with the olfactory receptor gene family.

FSARD yields biologically plausible interpretations for other linear factors, as well. Factor 25 may be associated with respiration or metabolism. Factor 29 appears to have a theme related to collagen and blood vessel formation (a *wound healing* gene set is also assigned further down the list). Factor 33 seems related to antigen presentation. These interpretations are highly suggestive, and could be helpful for a biologist exploring their data.

This pan-cancer setting has allowed us to demonstrate PathMatFac's ability to extract insights from large datasets. However, it's worth noting some biological limitations of this pan-cancer analysis. As a matrix factorization model, PathMatFac fundamentally aims to *explain variation* between samples in a dataset. The samples in this dataset belong to diverse kinds of cancer, stemming from diverse tissues. As a result, much of the variation captured by PathMatFac will simply be differences between tissues, rather than "interesting" variation that sheds light on cancer types or their treatment. In practice, one should thoughtfully curate their samples in a way that ensures PathMatFac will illuminate meaningful variation. For example, the dataset could include samples from closely related cancers; or tumor-normal pairs collected from the same patients.

3.4 Discussion

We have presented PathMatFac, a matrix factorization model for multiomic data. PathMatFac (*i*) effectively reduces the dimension of a multiomic dataset, and (*ii*) uses a new technique, FSARD, to summarize the linear factors in terms of curated gene sets.

PathMatFac entails moderate computational expense, and is capable of learning from small or large datasets. The method flexibly accommodates realistic data containing batch effects and missing values. Tests on simulated data show that PathMatFac reliably recovers its true parameters. When we apply PathMatFac as a dimension reduction in supervised learning tasks, we find that it outperforms other linear embeddings, such as MOFA+ or PCA. When we use PathMatFac to explore a real dataset from TCGA, we observe biologically plausible patterns in its linear factors, by visual inspection. We also find that FSARD succeeds at interpreting *some* linear factors via curated gene sets, though many factors are *not* easily

Factor	View	Gene set	Coeff.
1	CNA	chr8q21	2.28
		chr8q22	2.27
		chr8q12	1.95
		chr8q13	1.91
		chr8q11	1.69
		(3 more rows)	
⋮	⋮	⋮	⋮
3	RNA-seq	POSITIVE_REGULATION_OF_MACROMOLECULE...	0.87
		CELL_CYCLE	0.71
		POSITIVE_REGULATION_OF_RNA_METABOLIC_PROCESS	0.67
		REGULATION_OF_CELL_POPULATION_PROLIFERATION	0.58
		CYTOSKELETON_ORGANIZATION	0.56
		(36 more rows)	
⋮	⋮	⋮	⋮
19	Methylation	GOBP_SENSORY_PERCEPTION_OF_SMELL	1.71
		DETECTION_OF_CHEMICAL_STIMULUS	1.64
		SENSORY_PERCEPTION_OF_CHEMICAL_STIMULUS	1.64
		DETECTION_OF_STIMULUS_INVOLVED_IN_SENSORY_PERCEPTION	1.58
		DETECTION_OF_STIMULUS	1.56
		(6 more rows)	
⋮	⋮	⋮	⋮
25	RNA-seq	CELLULAR_RESPIRATION	0.64
		ELECTRON_TRANSPORT_CHAIN	0.50
		AEROBIC_RESPIRATION	0.49
		RESPIRATORY_ELECTRON_TRANSPORT_CHAIN	0.37
		ENERGY_DERIVATION_BY_OXIDATION_OF_ORGANIC_COMPOUNDS	0.31
		(2 more rows)	
⋮	⋮	⋮	⋮
29	RNA-seq	EXTERNAL_ENCAPSULATING_STRUCTURE_ORGANIZATION	1.46
		COLLAGEN_FIBRIL_ORGANIZATION	1.10
		BLOOD_VESSEL_MORPHOGENESIS	0.77
		CELL_SUBSTRATE_ADHESION	0.46
		VASCULATURE_DEVELOPMENT	0.45
		(16 more rows)	
⋮	⋮	⋮	⋮
33	RNA-seq	PRESENTATION_OF_PEPTIDE_ANTIGEN	0.74
		...PRESENTATION_OF_ENDOGENOUS_ANTIGEN	0.62
		...PRESENTATION_OF_EXOGENOUS_PEPTIDE_ANTIGEN	0.54
		...PRESENTATION_OF_ENDOGENOUS_PEPTIDE_ANTIGEN	0.51
		...PRESENTATION_OF_EXOGENOUS_ANTIGEN	0.35
		(8 more rows)	
⋮	⋮	⋮	⋮

Table 3.5: PathMatFac's FSARD interpretation for selected factors, fitted on the pan-cancer TCGA dataset. This table is structured analogously to Table 3.4.

captured by gene sets.

PathMatFac has important limitations worth keeping in mind. As a matrix factorization, PathMatFac imposes greater inductive bias than neural networks. Hence, it will not generally capture as much of the data's variation as techniques based on deep learning, e.g., variational autoencoders. Unlike other techniques that estimate a posterior distribution for their parameters (e.g., MOFA+ or VAEs), PathMatFac does not quantify uncertainty in its estimates. As mentioned previously, the quality of FSARD's factor interpretations depends strongly on the collection of gene sets provided to it.

Practical recommendations for using PathMatFac. Using PathMatFac requires the user to make a few choices. We present some simple guidance for properly configuring the method. Selecting an appropriate K is made slightly easier by the fact that PathMatFac's ARD regularizer tends to prune away inconsequential factors. Hence, we recommend erring on the side of *too many* factors. Then, if there are many factors with small norm, the user may (*i*) ignore them or (*ii*) rerun PathMatFac with a reduced number of factors. When we apply PathMatFac to real data, we often begin with K=50.

The simulation study in Section 3.3 suggests that choosing an appropriate value for K can help improve the embedding. In principle one could use cross-validation or a permutation test to select the number of factors. Of course, that incurs the computational expense of fitting the model many times. We do not explore that methodology in this chapter.

As shown in the simulations, PathMatFac is quite robust to the number of training iterations. For modest datasets with M<5,000 and $N < 10,000$, we find that one thousand training iterations is a good rule of thumb. More iterations may be necessary for larger datasets. PathMatFac records the loss during training; the user may inspect this and judge whether the loss decrease per iteration has become sufficiently small.

The choice of curated gene sets requires more judgment. MSigDB offers a broad selection of gene sets (Liberzon et al., 2015). Its *gene ontology* collection may be a sensible default choice for generic bioinformatics applications, since it covers many genes. However, specific contexts and omics assays may call for different ones. For example, MSigDB's *locational* gene sets appear to be an excellent choice for CNA data. The evaluations in this chapter focus on a cancer application, so collections of *cancer hallmark* and *oncogenic* gene sets seemed like potentially useful choices. However, they did not always produce clear, unambiguous factor summaries for RNA-seq, methylation, or somatic mutation data. Other online sources of gene sets exist and may be worth exploring, such as Pathway Commons (Cerami et al., 2011).

Potential applications. We imagine PathMatFac could be useful for other applications not explored in this chapter. Single-cell multiomic assay technologies are gaining adoption (Lee et al., 2020; Flynn et al., 2023). These technologies simultaneously collect multiple omics measurements from individual cells in a biological sample. While none of our evaluations apply PathMatFac to single-cell data, this could be easily accomplished by giving PathMatFac appropriate loss functions. For example, loss derived from a negative-binomial model can yield a better fit on scRNA-seq data than simpler ones. Extending PathMatFac in this way would require minimal effort.

Many of the VAE models mentioned in Section 3.1 are designed to *impute* missing modalities of data. This is motivated by the fact that multiomic datasets are more costly to collect than single-omics datasets (Argelaguet et al., 2021). The goal is to get the benefits of multiomic data at much lower cost, by (*i*) collecting multiple omics for a *small* number of samples; (*ii*) using inexpensive single-omics assays to collect *many more* samples; and then (*iii*) impute missing modalities for the majority of samples, which can be used for downstream analyses. PathMatFac could conceivably be applied to this task. However, PathMatFac's primary goal is to model the variation in a dataset, and explain it in terms that are familiar to biologists. Imputation was not a priority in this chapter, but we may explore it in the future.

PathMatFac is motivated by biological applications, but could easily be used in other settings. It can be regarded as a flexible and powerful matrix factorization software package, especially for multimodal data. However, we don't imagine many application domains having the equivalent of curated gene sets, so FSARD in particular may be less useful outside of biological applications.

Method improvements. PathMatFac, and especially its FSARD mechanism, could be improved in several ways. We saw in Section 3.3 that FSARD tends to assign many gene sets to a factor whenever the factor is poorly captured by them. In that case we have a problem of high *false positives*. The false positives could be addressed by introducing an hypothesis testing framework that computes p-values for gene set assignments, with a multiple-testing correction. For instance, FSARD's assignments could be recomputed many times with the entries of Y permuted within each factor and view. The fraction of permutations yielding that assignment would amount to a p-value; and a Benjamini-Hochberg correction within each factor and view (Benjamini and Hochberg, 1995) would account for the fact that we are performing many such tests. Although this approach would come with increased computational cost, it could be worthwhile in some scenarios.

Perhaps the best way to improve FSARD is by choosing better gene sets. Clearly, the selection of gene sets should depend on the biological setting as well as the omics assay. However, other criteria could inform the curation of better collections of gene sets. For example, we ought to prefer collections of gene sets with higher *coverage* of the genome. We may also prefer collections that are *irredundant*, or that have redundancy in a structured fashion that prevents ambiguity in the assignments. A more careful mathematical formulation of these criteria could inform the construction of better collections (for the purposes of FSARD).

Other potential improvements relate to PathMatFac's fundamental probabilistic assumptions and inference procedure. Despite the successes shown in this chapter, it is possible that the nonconvexity of PathMatFac's ARD regularizer hinders its performance. On one hand, this could be addressed by improving the inference procedure. For example, an algorithm based on evidence maximization (MacKay, 1999), or variational Bayes as in MOFA+ (Argelaguet et al., 2020), could sidestep this issue.

Other strategies to address the nonconvexity of PathMatFac's loss would keep the MAP inference, but choose different probabilistic assumptions than ARD. For example, a Laplace distribution prior for Y would be an obvious substitution for ARD. FSARD would be reformulated accordingly, as a hyperprior for that Laplace distribution.

4 DISCUSSION AND CLOSING REMARKS

> *When one tugs at a single thing in nature, he finds it is attached to the rest of the world.*
>
> — John Muir

This chapter makes some final comments about the research described in the previous chapters—some reflections and perspective gained from time and hindsight.

It also leaves remarks that are broader in scope. I have included some observations about (*i*) the value of biological prior knowledge in omics data analysis, (*ii*) research practices for modeling omics data, and (*iii*) the near-term prospects for machine learning on omics data.

4.1 Parting thoughts on SSPS, PathMatFac, and biological prior knowledge

SSPS. Over time I have grown increasingly skeptical toward the modeling assumptions of SSPS. Real biological signaling operates at a variety of time scales (seconds, minutes, hours) but SSPS assumes all interactions occur at a similar timescale. SSPS makes a "closed-world" assumption: that the behaviors of observed pathway entities are fully explained by the other *observed* entities. These issues seem fundamental to the class of techniques based on Dynamic Bayesian Network structure-learning, e.g., those of Spencer et al. (2015), Hill et al. (2012), and Werhli and Husmeier (2007). The class of techniques could be modified to address those issues, but only with substantial conceptual changes. For instance, continuous-time models based on differential equations or Gaussian processes could address the concerns about time scales. The closed-world assumption is less tractable, but could be addressed in part by, e.g., modeling *changes* in the time dynamics In its current state, SSPS could be a valid model for identifying DBN structures in certain restricted settings. However, it's not immediately clear what those settings would be.

Despite this, I do think SSPS has *some* merit. It represents a legitimate improvement over *past* techniques based on DBN structure-learning. It also contains worthwhile innovations in MCMC sampling for sparse networks (in the context of DBN structure-learning). The asymmetric *parent set* proposal distribution clearly biases the Markov chain in a way that yields better sampling efficiency.

PathMatFac. A brief story of PathMatFac's development may help the reader understand the perspective I've reached. The version of PathMatFac described in Chapter 3 is the latest of several attempts at modeling multiomic data and biological pathways. Earlier versions made much stronger probabilistic assumptions about pathways and their relationship to omics data. Ultimately, I found that these versions of PathMatFac fit the data poorly. The current version improves on them greatly.

From the beginning, PathMatFac aimed to (*i*) explain the variance in samples of multiomic data and (*ii*) describe this variance in terms of biological pathways. It always took an approach based on matrix factorization. However, several important aspects of the model changed over time in response to empirical evaluations.

Perhaps the most important modeling assumption to change was the assumed relationship between linear factors and biological pathways. At first, PathMatFac assumed a one-to-one correspondence between factors and pathways. In other words, the matrix factorization would include exactly one linear factor for each biological pathway considered in the analysis. This amounts to an assumption that (*i*) the included pathways are complete; and (*ii*) each pathway makes a single additive contribution to each sample in the dataset. A practical implication of this was that the matrix factorization would necessarily include a large number of factors in order to cover a reasonable set of pathways.

Another key modeling assumption to change was the *regularization* applied to linear factors. The "one-to-one" assumption mentioned above implied that each linear factor *represented* a pathway. Early versions of PathMatFac attempted this through a *graph regularization* scheme. The idea was to translate each pathway into an undirected graph, and then use the graph's Laplacian matrix to define a quadratic regularizer. Probabilistically, this amounts to a multivariate normal prior on the linear factor (where the graph Laplacian serves as a precision matrix). From a biological perspective, this modeling assumption asserts that multiomic features near each other in the pathway will tend to share correlations.

I conceived of these modeling ideas by thinking abstractly about pathways, without relying on observation or data. I reasoned that pathways represent dependencies between molecules. Surely these dependencies imply the existence of correlations between the multiomic features?

Unfortunately, these ideas crashed into reality as soon as I tested the model on simulated and real data. Simulations showed that graph regularization was ineffective at biasing the factors as intended—even on synthetic data, consistent with the modeling assumptions. And the problems became worse as the number of factors (i.e., pathways) increased from 10s to 100s. Fits on real data also failed to produce factors corresponding to pathways in any reasonable way. Over time, this empirical feedback led to several important changes to the model. In particular, I abandoned strong assumptions about pathways and their relationships to multiomic data.

The current form of PathMatFac is much more grounded in empiricism than earlier versions. It imposes no mechanistic assumptions about the underlying biology. PathMatFac's FSARD hyperprior *does* posit a probabilistic dependence between the linear factors and the biological prior knowledge (i.e., gene sets). However, under recommended usage, the gene sets do not *inform* the matrix factorization—they are only used to *interpret* it.

The value of PathMatFac can best be understood in relation to PLIER (Mao et al., 2019). The methods have similar goals: (*i*) factorize an omics dataset and (*ii*) summarize the factorization in terms of curated gene sets. However, PathMatFac has a distinct advantage over PLIER by accommodating multiomic data; in fact, its ability to model *non-Gaussian* data at all gives PathMatFac an edge over PLIER.

PathMatFac is one more tool for embedding multiomic datasets, in an already-crowded field of options. VAEs have gained dominance in this area, in a very short time (Gong et al., 2021; Ashuach et al., 2021; Gut et al., 2021; Gayoso et al., 2021; Lotfollahi et al., 2022; Zhou et al., 2023). Matrix factorizations still have a role to play, although MOFA+ fills that role fairly well (Argelaguet et al., 2020). In this crowded ecosystem, PathMatFac occupies a small—though unique—niche, precisely because of its emphasis on *interpretability*. None of the other listed techniques are designed to yield gene set interpretations. PathMatFac could secure its place in this niche with continued development, including the improvements mentioned in Section 3.4.

The value of biological prior knowledge. My work in probabilistic modeling for omics data has brought me to a nuanced position regarding the value of biological prior knowledge, and especially pathways. I entered this research area with naïve assumptions:

1. The thousands of curated biological pathways capture *most* of the molecular processes governing cells—or, at least, enough to be useful.

2. The edges in curated pathways *approximately* reflect the pairwise interactions in real cell processes.

3. Omics data relate to pathways in a fairly straightforward way. Pathway edges ought to manifest themselves as probabilistic dependencies in the data.

However, I now take the concept of a "biological pathway" much less seriously.

It seems instructive to make an analogy between (*i*) biological pathways for modeling omics data; and (*ii*) formal grammars for natural language processing (NLP). Formal grammars help linguists organize their knowledge about languages. However, NLP based on formal grammar never performed well. NLP only began to see success when datasets became larger and techniques relied on statistical patterns rather than grammars (Manning and Schutze, 1999). Similarly, pathways help biologists organize their knowledge of biological systems; but there's no guarantee they will help in modeling omics data.

Pathways and gene sets seem most useful as tools for *interpreting* data. They can serve as statistically useful collections. For example, comparisons between gene sets can be more statistically robust than comparisons between individual genes. Classic, enrichment-based techniques like GSEA (Subramanian et al., 2005) and GO analysis (Young et al., 2010) are based on this principle. I am more skeptical of pathways and gene sets as Bayesian priors to *inform* modeling of omics data.

A wiser person may have reached similar conclusions without years of misguided effort.

4.2 Research practices and methodology

Doubts about black-box inference

In their text *Bayesian Data Analysis*, Gelman et al. (2014) lay out an idealized process for probabilistic modeling:

1. Define the *full probability model*. In other words, implement a joint distribution for all observed and latent variables.

2. Condition on observed data. That is, hold the observed variables fixed and estimate quantities of interest from the posterior distribution by some inference procedure.

3. Evaluate the model's fit and inferences. Do the estimated quantities make sense? Are they sensitive to the modeling assumptions?

4. Repeat steps 1-3 until the results are satisfactory.

This is a sound plan for modeling, but it's worth bearing in mind that *computational tools* play a crucial role in each step.

Gelman et al. (2014) recommend the Stan probabilistic programming language to carry out steps 1 and 2 (Carpenter et al., 2017). Stan enables the user to easily define a probabilistic model and make posterior inferences. Stan's abilities to *automate* inference can be especially valuable, since the design and implementation of Bayesian inference procedures can be quite challenging. This design philosophy, which hides the details of inference from the user, has been called *black-box inference* (Ranganath et al., 2014; Salimans and Knowles, 2014).

While black-box inference can be helpful, it places important limitations on (*i*) model design and (*ii*) the efficiency of inference. Stan in particular makes inferences via Hamiltonian Monte Carlo (HMC) sampling (Betancourt, 2018), or in some cases variational Bayes (Kucukelbir et al., 2015). These are sophisticated techniques, but they have limitations. As a small example, both techniques only operate on continuous variables—the user is responsible for marginalizing any discrete variables, which isn't always practical and seems contrary to the "black box" value proposition. Furthermore, they are unable to benefit from domain-specific knowledge that could improve inference efficiency. For example, we saw in Chapter

2 that an asymmetric proposal distribution informed by knowledge of the posterior greatly improved sampling efficiency for MCMC. Black-box procedures cannot in general exploit knowledge in that way. I see similar issues in other probabilistic programming environments based on black-box inference, such as Pyro (Bingham et al., 2018) or, historically, Edward (Dillon et al., 2017).

These issues lead me to think that the aspirations of black-box inference are mistaken. A better design would provide the user with a set of reasonable default inference algorithms, and provide ways to modify or adapt those algorithms to the problem at hand. This is the design philosophy behind Gen, the probabilistic programming language used in Chapter 2 (Cusumano-Towner et al., 2019). Gen offers *building blocks* for inference, and enables the user to assemble them into an *inference program* appropriate for the task at hand. This provides much of the convenience of black-box inference, but gives the user greater agency to design models and inference algorithms.

Reproducibility in computational research

Like any science, computational research should be reproducible. Ideally, all research software and data would be publicly available, and published results would be reproducible without undue effort. Distributing source code via GitHub (or some other Git service) is a minimal standard for software availability. Distributing the software as a package—e.g., an R, Python, or Julia package—is even better. Data (and code) can be archived at Zenodo and given a DOI, which provides some guarantee for its long-term availability.

Beyond *availability*, software should be *usable* without undue effort. Distributing a package can also help in this respect. Publishing a container (e.g., Docker or Singularity) can be an excellent solution when the software has complicated dependencies.

Computational biologists sometimes set up *web servers* to demonstrate their software tools, as a "reproducibility" measure. However, web servers are *at best* a partial solution. They can only be used in a manual ad-hoc manner, and can't be incorporated into a laboratory's analysis pipelines. It's been documented that the availability and maintenance is quite poor for most of these research-related web servers (Kern et al., 2020).

Finally, the published results of analyses should be reproducible without unreasonable effort. Scripts and Jupyter notebooks are better than nothing. However, *workflow managers* like Snakemake, NextFlow, and AirFlow have many strengths over scripts or notebooks. If a researcher defines their analyses as a workflow in one of these systems, and distributes the workflow, then this allows others to run the identical set of analyses. Workflow managers confer other benefits, though. They allow the analyses to be run with arbitrary amounts of parallelism. If a workflow gets interrupted for some reason, then the workflow manager can resume where it stopped, without rerunning the entire set of analyses. Some workflow managers, like Snakemake, can also manage software environments via containers. Beyond reproducibility, workflow managers can be extremely useful for computational work in general. A well-constructed workflow can automate otherwise-tedious analysis steps. Both projects in this dissertation made extensive use of Snakemake to manage their analyses.

An opinionated review of programming languages for model development

I avoid being ideological when I choose software tools—in every case I try to use the right tool for the job. However, my PhD research has made me an enthusiastic proponent of the Julia programming language. People frequently ask me about this after perusing my work. A brief discussion about programming language choice seems appropriate in this chapter.

A useful programming language for model implementation should possess a rich collection of relevant libraries or packages. R has the most complete ecosystem of bioinformatics and biostatistics packages. The Bioconductor package repository provides convenient access to the most widely used models and analyses. However, my research does not focus on *performing* analyses. It focuses on *developing* new methods. This entails a different set of requirements. I need packages for automatic differentiation; GPU programming; probabilistic programming; and numerical linear algebra. Both Julia and Python have a strong advantage over R in this regard.

A language for model development should be designed according to sound software engineering principles. Historically, R was developed by statisticians to *perform analyses*. In contrast, languages like Python and Julia were developed by computer scientists to *build tools*. These origins manifest themselves in language design, in ways that are apparent to software engineers but less apparent to others (Smith and Ushey, 2023; Hacker News, 2023). Without a doubt, R is an excellent language for performing analyses. However, I'm in the business of building tools and Julia and Python are much better suited for that.

A language for model development should produce fast, performant code. Python and R are both interpreted, though many of their packages call efficient compiled code. In contrast, Julia is *dynamically compiled*. This means it incurs an up-front compilation cost at run-time, but generates efficient compiled code. Since I'm building tools with a typical runtime of minutes or hours, the gains of dynamic compilation generally outweigh the losses.

A language for model development should allow me to write *free* software—free as in freedom (libre) *and* as in beer (gratis). This is a reasonable requirement for code in a collaborative, scientific setting. However, it immediately excludes MATLAB or Mathematica, which require a paid license. In contrast, Python, Julia, and R have fairly permissive licenses; in those languages, intellectual property rarely poses practical issues in the development of research software.

I admit that *novelty* also factored into my choice. It seemed worthwhile to experiment with Julia and see how it compared to other options. I was not disappointed!

4.3 Prospects for machine learning on omics data

Machine learning on omics data has historically been *data-constrained*. The *quantity* of data has been limited, since (*i*) assay technologies are relatively expensive, and (*ii*) large sample sizes can be hard to acquire, especially if they come from humans or live animals. Data *quality* has also been an issue. As documented previously in this dissertation, omics data are almost never *i.i.d.*—for biological and technical reasons. In principle a machine learning model could account for this heterogeneity between datasets, similar to how we modeled away batch effects in Chapter 3. However the omics datasets are rarely annotated with sufficient metadata to model these systematic differences. These issues can make machine learning on omics data more difficult than in other domains, e.g., text or images.

We can expect the quantity and quality of omics data to increase over time, though. Investment in biomedical technologies will most likely continue to increase, for demographic and economic reasons. Populations across the developed world are aging. Accordingly, age-related disease—cancer, neurodegenerative disease, heart disease, etc.—will increasingly burden healthcare systems. Governments and corporations are strongly incentivized to pursue technological solutions to these problems.

Machine learning has already demonstrated value in the domains of text and image processing. Investors are eager to see it make an impact in biotech, and understand that this will require large amounts of high-quality omics data. For instance, venture capitalists traditionally focused on software have been turning their sights toward biotech and healthcare. Blog posts from Andreesen Horowitz describe a strategic rationale for this (Conde and Rughani, 2023; Wolf and Pande, 2023). Money speaks louder, though: Andreesen Horowitz and Sequoia both have large portfolios devoted to AI in biotech (cipherbio, 2023; Andreesen Horowitz, 2023)

Accordingly, a large number of data-centric biotech and drug discovery startups have begun to emerge. Notable examples include Recursion Pharmaceuticals (Recursion Pharmaceuticals, 2023a) and Insitro (Insitro, 2022), which both emphasize a "lab in the loop" business model. This amounts to active learning: an automated laboratory generates data in a consistent and highly controlled fashion; machine learning models learn from the data and inform the next round of experiments. The public datasets released by Recursion Pharmaceuticals provide a sense of the scale of data collection and machine learning taking place there. Recursion's public RxRx dataset is approximately 83TB, and comprises less than 1% of their full dataset (Recursion Pharmaceuticals, 2023b).

This investment is also yielding assay technologies that are (*i*) more informative and (*ii*) less expensive. Chapter 1 already mentioned some of these. Another area worth paying attention to is microscopy. Developments in microscopy will make computer vision techniques increasingly relevant in the life sciences. For example, quantitative phase microscopy (Park et al., 2018) can capture the shape and organization of cells in greater detail and with less distortion than older techniques based on staining. Daphne Köller spoke specifically about this during an ICML 2022 workshop talk (Köller, 2022).

All of this is to say, machine learning on omics data has a bright future. There are many impactful and intellectually stimulating problems to solve.

It's difficult to imagine a field of work with greater potential human and economic value.

The work in this dissertation was based on a simple idea: that probabilistic models could provide a natural way to learn from (*i*) omics data and (*ii*) biological prior knowledge. The lessons I took away were more nuanced. Biological prior knowledge may have a place in the bright future described above—pathways and gene sets are a useful *vocabulary* for biologists when they interpret data and inferences. However, I don't expect pathways and gene sets will have success as prior knowledge in a Bayesian sense.

DISCARD THIS PAGE

COLOPHON

This document was typeset in LaTeX using Will Benton's template:

https://github.com/willb/wi-thesis-template.git

The default font is Gyre Pagella.

This document was written 100% free of LLM labor.

REFERENCES

23andMe. 2013. Our Service: Genotyping Technology - 23andMe. https://web.archive.org/web/20131202222247/https://www.23andme.com/more/genotyping/.

Abdi, Herve. 2003. Factor Rotations in Factor Analyses. https://personal.utdallas.edu/~herve/Abdi-rotations-pretty.pdf.

Aebersold, Ruedi, and Matthias Mann. 2003. Mass spectrometry-based proteomics. *Nature* 422(6928):198–207. Number: 6928 Publisher: Nature Publishing Group.

Andreesen Horowitz. 2023. a16z Bio + Health Tech Investments | Andreessen Horowitz. https://web.archive.org/web/20230526053229/https://a16z.com/bio/.

Argelaguet, Ricard, Damien Arnol, Danila Bredikhin, Yonatan Deloro, Britta Velten, John C. Marioni, and Oliver Stegle. 2020. MOFA+: a statistical framework for comprehensive integration of multi-modal single-cell data. *Genome Biology* 21(1):111.

Argelaguet, Ricard, Anna S. E. Cuomo, Oliver Stegle, and John C. Marioni. 2021. Computational principles and challenges in single-cell data integration. *Nature Biotechnology* 39(10):1202–1215. Number: 10 Publisher: Nature Publishing Group.

Argelaguet, Ricard, Britta Velten, Damien Arnol, Sascha Dietrich, Thorsten Zenz, John C Marioni, Florian Buettner, Wolfgang Huber, and Oliver Stegle. 2018. Multi-Omics Factor Analysis—a framework for unsupervised integration of multi-omics data sets. *Molecular Systems Biology* 14(6):e8124. Publisher: John Wiley & Sons, Ltd.

Ashuach, Tal, Mariano I. Gabitto, Michael I. Jordan, and Nir Yosef. 2021. MultiVI: deep generative model for the integration of multi-modal data. preprint, Bioinformatics.

Baker, Monya. 2013. Big biology: The 'omes puzzle. *Nature* 494(7438): 416–419. Number: 7438 Publisher: Nature Publishing Group.

Beck, Amir, and Marc Teboulle. 2009. A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Benjamini, Yoav, and Yosef Hochberg. 1995. Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society: Series B (Methodological)* 57(1):289–300. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1995.tb02031.x.

Besard, Tim, Christophe Foket, and Bjorn De Sutter. 2018. Effective extensible programming: Unleashing Julia on GPUs. *IEEE Transactions on Parallel and Distributed Systems*. `1712.03112`.

Betancourt, Michael. 2018. A Conceptual Introduction to Hamiltonian Monte Carlo. ArXiv:1701.02434 [stat].

Bingham, Eli, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. 2018. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*.

Budak, Gungor, Oyku Eren Ozsoy, Yesim Aydin Son, Tolga Can, and Nurcan Tuncbag. 2015. Reconstruction of the temporal signaling network in Salmonella-infected human cells. *Frontiers in Microbiology* 6:730.

Cardner, Mathias, Nathalie Meyer-Schaller, Gerhard Christofori, and Niko Beerenwinkel. 2019. Inferring signalling dynamics by integrating interventional with observational data. *Bioinformatics* 35(14):i577–i585.

Carlin, Daniel E., Evan O. Paull, Kiley Graim, Christopher K. Wong, Adrian Bivol, Peter Ryabinin, Kyle Ellrott, Artem Sokolov, and Joshua M. Stuart. 2017. Prophetic granger causality to infer gene regulatory networks. *PLOS ONE* 12(12):e0170340.

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. Stan: A probabilistic programming language. *Journal of Statistical Software* 76(1).

Cerami, Ethan G., Benjamin E. Gross, Emek Demir, Igor Rodchenkov, Özgün Babur, Nadia Anwar, Nikolaus Schultz, Gary D. Bader, and Chris Sander. 2011. Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Research* 39(Database issue):D685–D690.

Chauveau, Claire, John Rowell, and Ana Ferreiro. 2014. A Rising Titan: TTN Review and Mutation Update. *Human Mutation* 35(9):1046–1059. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/humu.22611.

Chauvel, Cécile, Alexei Novoloaca, Pierre Veyre, Frédéric Reynier, and Jérémie Becker. 2020. Evaluation of integrative clustering methods for the analysis of multi-omics data. *Briefings in Bioinformatics* 21(2):541–552.

Chen, Hugh, Ian C. Covert, Scott M. Lundberg, and Su-In Lee. 2023. Algorithms to estimate Shapley value feature attributions. *Nature Machine Intelligence* 1–12. Publisher: Nature Publishing Group.

Chen, Kok Hao, Alistair N. Boettiger, Jeffrey R. Moffitt, Siyuan Wang, and Xiaowei Zhuang. 2015. Spatially resolved, highly multiplexed RNA profiling in single cells. *Science (New York, N.Y.)* 348(6233):aaa6090.

Cheong, Raymond, Alex Rhee, Chiaochun Joanne Wang, Ilya Nemenman, and Andre Levchenko. 2011. Information Transduction Capacity of Noisy Biochemical Signaling Networks. *Science* 334(6054):354–358.

cipherbio. 2023. Sequoia Capital Investment Information | CipherBio. https://www.cipherbio.com/data-viz/investor/Sequoia%2BCapital.

Coarfa, Cristian, Sandra L. Grimm, Kimal Rajapakshe, Dimuthu Perera, Hsin-Yi Lu, Xuan Wang, Kurt R. Christensen, Qianxing Mo, Dean P. Edwards, and Shixia Huang. 2021. Reverse-Phase Protein Array: Technology, Application, Data Processing, and Integration. *Journal of Biomolecular Techniques : JBT* 32(1):15–29.

Conde, Jorge, and Jay Rughani. 2023. Doing More with Moore: Biotech's Tech Moment | Andreessen Horowitz. https://web.archive.org/web/20230515020608/https://a16z.com/2023/02/14/doing-more-with-moore/.

Conesa, Ana, Pedro Madrigal, Sonia Tarazona, David Gomez-Cabrero, Alejandra Cervera, Andrew McPherson, Michał Wojciech Szcześniak, Daniel J. Gaffney, Laura L. Elo, Xuegong Zhang, and Ali Mortazavi. 2016. A survey of best practices for RNA-seq data analysis. *Genome Biology* 17(1):13.

Cusumano-Towner, Marco F., Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. 2019. Gen: A general-purpose probabilistic programming system with programmable inference. In *Proceedings of the 40th acm sigplan conference on programming language design and implementation*. PLDI 2019, New York, NY, USA: ACM.

Davis, Jesse, and Mark Goadrich. 2006. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd international conference on machine learning*.

Dedeurwaerder, Sarah, Matthieu Defrance, Martin Bizet, Emilie Calonne, Gianluca Bontempi, and François Fuks. 2014. A comprehensive overview of Infinium HumanMethylation450 data processing. *Briefings in Bioinformatics* 15(6):929–941.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)* 39(1):1–38. Publisher: [Royal Statistical Society, Wiley].

Dillon, Joshua V., Ian Langmore, Dustin Tran, Eugene Brevdo, Srinivas Vasudevan, Dave Moore, Brian Patton, Alex Alemi, Matt Hoffman, and Rif A. Saurous. 2017. Tensorflow distributions. *arXiv* arXiv:1711.10604.

Documentation, NCI GDC. 2022a. Bioinformatics Pipeline: Copy Number Variation Analysis - GDC Docs. `https://web.archive.org/web/20221202025900/https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/CNV_Pipeline/`.

———. 2022b. Bioinformatics Pipeline: DNA-Seq Analysis - GDC Docs. `https://web.archive.org/web/20221203145132/https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/DNA_Seq_Variant_Calling_Pipeline/`.

———. 2022c. Bioinformatics Pipeline: Methylation Analysis Pipeline - GDC Docs. `https://web.archive.org/web/20220924191217/https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Methylation_Pipeline/`.

———. 2022d. Bioinformatics Pipeline: mRNA Analysis - GDC Docs. `https://web.archive.org/web/20221116232737/https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/Expression_mRNA_Pipeline/`.

———. 2023. Bioinformatics Pipeline: Protein Expression - GDC Docs. `https://web.archive.org/web/20230227225743/https://docs.gdc.cancer.gov/Data/Bioinformatics_Pipelines/RPPA_intro/`.

Drake, Justin M., Evan O. Paull, Nicholas A. Graham, John K. Lee, Bryan A. Smith, Bjoern Titz, Tanya Stoyanova, Claire M. Faltermeier, Vladislav Uzunangelov, Daniel E. Carlin, Daniel Teo Fleming, Christopher K. Wong, Yulia Newton, Sud Sudha, Ajay A. Vashisht, Jiaoti Huang, James A. Wohlschlegel, Thomas G. Graeber, Owen N. Witte, and Joshua M. Stuart. 2016. Phosphoproteome Integration Reveals Patient-Specific Networks in Prostate Cancer. *Cell* 166(4):1041–1054.

Eduati, Federica, Patricia Jaaks, Jessica Wappler, Thorsten Cramer, Christoph A Merten, Mathew J Garnett, and Julio Saez-Rodriguez. 2020. Patient-specific logic models of signaling pathways from screenings on cancer biopsies to prioritize personalized combination therapies. *Molecular Systems Biology* 16(2):e8664.

Eker, Steven, Merrill Knapp, Keith Laderoute, Patrick Lincoln, Jose Meseguer, and Kemal Sonmez. 2002. Pathway logic: symbolic analysis of biological signaling. *Pacific Symposium on Biocomputing* 400–412.

Flach, Peter A., and Meelis Kull. 2015. Precision-recall-gain curves: PR analysis done right. In *Advances in neural information processing systems*.

Flynn, Emily, Ana Almonte-Loya, and Gabriela K. Fragiadakis. 2023. Single-Cell Multiomics. *Annual Review of Biomedical Data Science* 6(1):null. _eprint: https://doi.org/10.1146/annurev-biodatasci-020422-050645.

Friedman, Jerome, Trevor Hastie, and Rob Tibshirani. 2010. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software* 33.

Gayoso, Adam, Zoë Steier, Romain Lopez, Jeffrey Regier, Kristopher L. Nazor, Aaron Streets, and Nir Yosef. 2021. Joint probabilistic modeling of single-cell multi-omic data with totalVI. *Nature Methods* 18(3):272–282. Number: 3 Publisher: Nature Publishing Group.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2014. *Bayesian data analysis*. 3rd ed. CRC Press.

Gilks, W. R. 2005. Markov Chain Monte Carlo. In *Encyclopedia of Biostatistics*. John Wiley & Sons, Ltd. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/0470011815.b2a14021.

Gillespie, Marc, Bijay Jassal, Ralf Stephan, Marija Milacic, Karen Rothfels, Andrea Senff-Ribeiro, Johannes Griss, Cristoffer Sevilla, Lisa Matthews, Chuqiao Gong, Chuan Deng, Thawfeek Varusai, Eliot Ragueneau, Yusra Haider, Bruce May, Veronica Shamovsky, Joel Weiser, Timothy Brunson, Nasim Sanati, Liam Beckman, Xiang Shao, Antonio Fabregat, Konstantinos Sidiropoulos, Julieth Murillo, Guilherme Viteri, Justin Cook, Solomon Shorser, Gary Bader, Emek Demir, Chris Sander, Robin Haw, Guanming Wu, Lincoln Stein, Henning Hermjakob, and Peter D'Eustachio. 2022. The reactome pathway knowledgebase 2022. *Nucleic Acids Research* 50(D1): D687–D692.

Gjerga, Enio, Panuwat Trairatphisan, Attila Gabor, Hermann Koch, Celine Chevalier, Francesco Ceccarelli, Aurelien Dugourd, Alexander Mitsos, and Julio Saez-Rodriguez. 2020. Converting networks to predictive logic models from perturbation signalling data with CellNOpt. *bioRxiv* 2020.03.04.976852.

Gong, Boying, Yun Zhou, and Elizabeth Purdom. 2021. Cobolt: integrative analysis of multimodal single-cell sequencing data. *Genome Biology* 22(1):351.

Gregorczyk, Marco. 2010. An introduction to Gaussian Bayesian networks. In *Systems biology in drug discovery and development: Methods and protocols*, ed. Qing Yan, chap. 6, 121–147. Springer.

Grimsrud, Paul A., Danielle L. Swaney, Craig D. Wenger, Nicole A. Beauchene, and Joshua J. Coon. 2010. Phosphoproteomics for the Masses. *ACS Chemical Biology* 5(1):105–119. Publisher: American Chemical Society.

Gut, Gilles, Stefan G. Stark, Gunnar Rätsch, and Natalie R. Davidson. 2021. pmVAE: Learning Interpretable Single-Cell Representations with Pathway Modules. *bioRxiv* 2021.01.28.428664. Publisher: Cold Spring Harbor Laboratory Section: New Results.

Guziolowski, Carito, Santiago Videla, Federica Eduati, Sven Thiele, Thomas Cokelaer, Anne Siegel, and Julio Saez-Rodriguez. 2013. Exhaustively characterizing feasible logic models of a signaling network using Answer Set Programming. *Bioinformatics* 29(18):2320–2326.

Hacker News. 2023. A newcomer's (angry) guide to R | Hacker News. `https://web.archive.org/web/20230515051910/https://news.ycombinator.com/item?id=17305878`.

Halasz, Melinda, Boris N. Kholodenko, Walter Kolch, and Tapesh Santra. 2016. Integrating network reconstruction with mechanistic modeling to predict cancer therapies. *Science Signaling* 9(455):ra114.

Hanspers, Kristina, Anders Riutta, Martina Summer-Kutmon, and Alexander R. Pico. 2020. Pathway information extracted from 25 years of pathway figures. *Genome Biology* 21(1):273.

Hardoon, David R., Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation* 16(12):2639–2664. Conference Name: Neural Computation.

Henriques, David, Alejandro F. Villaverde, Miguel Rocha, Julio Saez-Rodriguez, and Julio R. Banga. 2017. Data-driven reverse engineering of signaling pathways using ensembles of dynamic models. *PLOS Computational Biology* 13(2):e1005379.

Hill, Steven M., Laura M. Heiser, Thomas Cokelaer, Michael Unger, Nicole K. Nesser, Daniel E. Carlin, Yang Zhang, Artem Sokolov, Even O. Paull, Chris K. Wong, Kiley Graim, Adrian Bivol, Haizhou Wang, Fan Zhu, Bahman Afsari, Ludmila V. Danilova, Alexander V. Favorov, Wai Shing Lee, Dane Taylor, Chenyue W. Hu, Byron L. Long, David P. Noren, Alexander J. Bisberg, HPN-DREAM Consortium, Joe W. Gray, Michael Kellen, Thea Norman, Stephen Friend, Amina A. Qutub, Elana J. Fertig, Yuanfang Guan, Mingzhou Song, Joshua M. Stuart, Paul T. Spellman, Heinz Koeppl, Gustavo Stovolitzky, Julio Saez-Rodriguez, and Sach Mukherjee. 2016. Inferring causal molecular networks: empirical assessment through a community-based effort. *Nature Methods*.

Hill, Steven M., Yiling Lu, Jennifer Molina, Laura M. Heiser, Paul T. Spellman, Terence P. Speed, Joe W. Gray, Gordon B. Mills, and Sach Mukherjee. 2012. Bayesian inference of signaling network topology in a cancer cell line. *Bioinformatics* 28:2804–2810.

Hill, Steven M., Nicole K. Nesser, Katie Johnson-Camacho, Mara Jeffress, Aimee Johnson, Chris Boniface, Simon E.F. Spencer, Yiling Lu, Laura M. Heiser, Yancey Lawrence, Nupur T. Pande, James E. Korkola, Joe W. Gray, Gordon B. Mills, Sach Mukherjee, and Paul T. Spellman. 2017. Context specificity in causal signaling networks revealed by phosphoprotein profiling. *Cell Systems* 73–83.

Hoadley, Katherine A., Christina Yau, Toshinori Hinoue, Denise M. Wolf, Alexander J. Lazar, Esther Drill, Ronglai Shen, Alison M. Taylor, Andrew D. Cherniack, Vésteinn Thorsson, Rehan Akbani, Reanne Bowlby, Christopher K. Wong, Maciej Wiznerowicz, Francisco Sanchez-Vega, A. Gordon Robertson, Barbara G. Schneider, Michael S. Lawrence, Houtan Noushmehr, Tathiane M. Malta, The Cancer Genome Atlas Network, Joshua M. Stuart, Christopher C. Benz, and Peter W. Laird. 2018. Cell-of-Origin Patterns Dominate the Molecular Classification of 10,000 Tumors from 33 Types of Cancer. *Cell* 173(2):291–304.e6. Publisher: Elsevier.

Hotelling, H. 1933. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology* 24:417–441. Place: US Publisher: Warwick & York.

HOTELLING, HAROLD. 1936. RELATIONS BETWEEN TWO SETS OF VARIATES*. *Biometrika* 28(3-4):321–377.

Hunter, Tony. 2009. Tyrosine phosphorylation: thirty years and counting. *Current Opinion in Cell Biology* 21(2):140–146.

Hänzelmann, Sonja, Robert Castelo, and Justin Guinney. 2013. GSVA: gene set variation analysis for microarray and RNA-Seq data. *BMC Bioinformatics* 14(1):7.

info@sagebase.org, Sage Bionetworks. 2013. Hpn-dream breast cancer network inference challenge; additional data details. `https://www.synapse.org/#!Wiki:syn1720047/ENTITY/56210`.

Innes, Michael. 2018. Don't unroll adjoint: Differentiating ssa-form programs. *CoRR* abs/1810.07951. `1810.07951`.

Innes, Michael, Elliot Saba, Keno Fischer, Dhairya Gandhi, Marco Concetto Rudilosso, Neethu Mariya Joy, Tejan Karmali, Avik Pal, and Viral

Shah. 2018. Fashionable modelling with flux. *CoRR* abs/1811.01457. 1811.01457.

Insitro. 2022. Insitro - Approach. https://web.archive.org/web/20220923225015/https://www.insitro.com/approach.

Ishwaran, H., and U.B. Kogalur. 2007. Random survival forests for r. *R News* 7(2):25–31.

Johnson, W. Evan, Cheng Li, and Ariel Rabinovic. 2007. Adjusting batch effects in microarray expression data using empirical Bayes methods. *Biostatistics* 8(1):118–127.

de Jong, Hidde. 2002. Modeling and Simulation of Genetic Regulatory Systems: A Literature Review. *Journal of Computational Biology* 9(1): 67–103. Publisher: Mary Ann Liebert, Inc., publishers.

Kanehisa, Minoru, and Susumu Goto. 2000. KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research* 28(1):27–30.

Kern, Fabian, Tobias Fehlmann, and Andreas Keller. 2020. On the lifetime of bioinformatics web services. *Nucleic Acids Research* 48(22):12523–12533.

Kholodenko, Boris, Michael B. Yaffe, and Walter Kolch. 2012. Computational Approaches for Analyzing Information Flow in Biological Networks. *Science Signaling* 5(220):re1.

Kholodenko, Boris N., John F. Hancock, and Walter Kolch. 2010. Signalling ballet in space and time. *Nature Reviews Molecular Cell Biology* 11(6):414–426.

Klami, Arto, Seppo Virtanen, and Samuel Kaski. 2013. Bayesian Canonical Correlation Analysis. *Journal of Machine Learning Research* 14(30):965–1003.

Klami, Arto, Seppo Virtanen, Eemeli Leppäaho, and Samuel Kaski. 2015. Group Factor Analysis. *IEEE transactions on neural networks and learning systems* 26(9):2136–2147.

Köksal, Ali Sinan, Kirsten Beck, Dylan R. Cronin, Aaron McKenna, Nathan D. Camp, Saurabh Srivastava, Matthew E. MacGilvray, Rastislav Bodik, Alejandro Wolf-Yadlin, Ernest Fraenkel, Jasmin Fisher, and Anthony Gitter. 2018. Synthesizing signaling pathways from temporal phosphoproteomic data. *Cell Reports* 24:3607–3618.

Köller, Daphne. 2022. ICML 2022 workshop: AI for Science. `https://web.archive.org/web/20230607051326/https://icml.cc/Conferences/2022/Schedule?showEvent=13450`.

Koster, J., and S. Rahmann. 2012. Snakemake–a scalable bioinformatics workflow engine. *Bioinformatics* 28(19):2520–2522.

Krishnaswamy, Smita, Matthew H. Spitzer, Michael Mingueneau, Sean C. Bendall, Oren Litvin, Erica Stone, Dana Pe'er, and Garry P. Nolan. 2014. Conditional density-based analysis of T cell signaling in single-cell data. *Science* 346(6213):1250689.

Kucukelbir, Alp, Rajesh Ranganath, Andrew Gelman, and David M. Blei. 2015. Automatic Variational Inference in Stan. ArXiv:1506.03431 [stat].

LaFramboise, Thomas. 2009. Single nucleotide polymorphism arrays: a decade of biological, computational and technological advances. *Nucleic Acids Research* 37(13):4181–4193.

Lawrence, Michael S., Petar Stojanov, Paz Polak, Gregory V. Kryukov, Kristian Cibulskis, Andrey Sivachenko, Scott L. Carter, Chip Stewart, Craig H. Mermel, Steven A. Roberts, Adam Kiezun, Peter S. Hammerman, Aaron McKenna, Yotam Drier, Lihua Zou, Alex H. Ramos, Trevor J. Pugh, Nicolas Stransky, Elena Helman, Jaegil Kim, Carrie Sougnez, Lauren

Ambrogio, Elizabeth Nickerson, Erica Shefler, Maria L. Cortés, Daniel Auclair, Gordon Saksena, Douglas Voet, Michael Noble, Daniel DiCara, Pei Lin, Lee Lichtenstein, David I. Heiman, Timothy Fennell, Marcin Imielinski, Bryan Hernandez, Eran Hodis, Sylvan Baca, Austin M. Dulak, Jens Lohr, Dan-Avi Landau, Catherine J. Wu, Jorge Melendez-Zajgla, Alfredo Hidalgo-Miranda, Amnon Koren, Steven A. McCarroll, Jaume Mora, Ryan S. Lee, Brian Crompton, Robert Onofrio, Melissa Parkin, Wendy Winckler, Kristin Ardlie, Stacey B. Gabriel, Charles W. M. Roberts, Jaclyn A. Biegel, Kimberly Stegmaier, Adam J. Bass, Levi A. Garraway, Matthew Meyerson, Todd R. Golub, Dmitry A. Gordenin, Shamil Sunyaev, Eric S. Lander, and Gad Getz. 2013. Mutational heterogeneity in cancer and the search for new cancer-associated genes. *Nature* 499(7457):214–218. Number: 7457 Publisher: Nature Publishing Group.

Lee, Jeongwoo, Do Young Hyeon, and Daehee Hwang. 2020. Single-cell multiomics: technologies and data analysis methods. *Experimental & Molecular Medicine* 52(9):1428–1442. Number: 9 Publisher: Nature Publishing Group.

Leiserson, Mark D. M., Fabio Vandin, Hsin-Ta Wu, Jason R. Dobson, Jonathan V. Eldridge, Jacob L. Thomas, Alexandra Papoutsaki, Younhun Kim, Beifang Niu, Michael McLellan, Michael S. Lawrence, Abel Gonzalez-Perez, David Tamborero, Yuwei Cheng, Gregory A. Ryslik, Nuria Lopez-Bigas, Gad Getz, Li Ding, and Benjamin J. Raphael. 2015. Pan-cancer network analysis identifies combinations of rare somatic mutations across pathways and protein complexes. *Nature Genetics* 47(2):106–114. Number: 2 Publisher: Nature Publishing Group.

Liberzon, Arthur, Chet Birger, Helga Thorvaldsdóttir, Mahmoud Ghandi, Jill P. Mesirov, and Pablo Tamayo. 2015. The Molecular Signatures Database (MSigDB) hallmark gene set collection. *Cell systems* 1(6):417–425.

Lotfollahi, Mohammad, Anastasia Litinetskaya, and Fabian J. Theis. 2022. Multigrate: single-cell multi-omic data integration. Pages: 2022.03.16.484643 Section: New Results.

Luecken, Malte D, and Fabian J Theis. 2019. Current best practices in single-cell RNA-seq analysis: a tutorial. *Molecular Systems Biology* 15(6): e8746. Publisher: John Wiley & Sons, Ltd.

MacKay, David J. C. 1999. Comparison of Approximate Methods for Handling Hyperparameters. *Neural Computation* 11(5):1035–1068. Conference Name: Neural Computation.

Maghsoudi, Zeynab, Ha Nguyen, Alireza Tavakkoli, and Tin Nguyen. 2022. A comprehensive survey of the approaches for pathway analysis using multi-omics data integration. *Briefings in Bioinformatics* 23(6): bbac435.

Makrodimitris, Stavros, Bram Pronk, Tamim Abdelaal, and Marcel Reinders. 2023. An in-depth comparison of linear and non-linear joint embedding methods for bulk and single-cell multi-omics. Pages: 2023.04.10.535672 Section: New Results.

Manning, Christopher, and Hinrich Schutze. 1999. *Foundations of Statistical Natural Language Processing*. MIT Press. Google-Books-ID: 3qnuDwAAQBAJ.

Mao, Weiguang, Elena Zaslavsky, Boris M. Hartmann, Stuart C. Sealfon, and Maria Chikina. 2019. Pathway-level information extractor (PLIER) for gene expression data. *Nature Methods* 16(7):607–610.

Meng, Chen, Azfar Basunia, Bjoern Peters, Amin Moghaddas Gholami, Bernhard Kuster, and Aedín C. Culhane. 2019. MOGSA: Integrative Single Sample Gene-set Analysis of Multiple Omics Data *. *Molecular & Cellular Proteomics* 18(8):S153–S168. Publisher: Elsevier.

Meng, Chen, Dominic Helm, Martin Frejno, and Bernhard Kuster. 2016. moCluster: Identifying Joint Patterns Across Multiple Omics Data Sets. *Journal of Proteome Research* 15(3):755–765. Publisher: American Chemical Society.

Merrell, David. 2022. Tcga hdf file pipeline. `https://doi.org/10.5281/zenodo.6977490`.

Merrell, David, Aws Albarghouthi, and Loris D'Antoni. 2017. Weighted Model Integration with Orthogonal Transformations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 4610–4616.

Merrell, David, Thevaa Chandereng, and Yeonhee Park. 2023. A Markov decision process for response-adaptive randomization in clinical trials. *Computational Statistics & Data Analysis* 178:107599.

Mezlini, Aziz M., and Anna Goldenberg. 2017. Incorporating networks in a probabilistic graphical model to find drivers for complex human diseases. *PLOS Computational Biology* 13(10):e1005580.

Mo, Qianxing, Ronglai Shen, Cui Guo, Marina Vannucci, Keith S Chan, and Susan G Hilsenbeck. 2018. A fully Bayesian latent variable model for integrative clustering analysis of multi-type omics data. *Biostatistics* 19(1):71–86.

Mo, Qianxing, Sijian Wang, Venkatraman E. Seshan, Adam B. Olshen, Nikolaus Schultz, Chris Sander, R. Scott Powers, Marc Ladanyi, and Ronglai Shen. 2013. Pattern discovery and cancer gene identification in integrated cancer genomic data. *Proceedings of the National Academy of Sciences* 110(11):4245–4250. Publisher: Proceedings of the National Academy of Sciences.

Molinelli, Evan J., Anil Korkut, Weiqing Wang, Martin L. Miller, Nicholas P. Gauthier, Xiaohong Jing, Poorvi Kaushik, Qin He, Gordon Mills, David B. Solit, Christine A. Pratilas, Martin Weigt, Alfredo Braunstein, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. 2013. Perturbation biology: inferring signaling networks in cellular systems. *PLOS Computational Biology* 9.

Moore, Lisa D., Thuc Le, and Guoping Fan. 2013. DNA Methylation and Its Basic Function. *Neuropsychopharmacology* 38(1):23–38. Number: 1 Publisher: Nature Publishing Group.

Murtagh, Fionn, and Pedro Contreras. 2012. Algorithms for hierarchical clustering: an overview. *WIREs Data Mining and Knowledge Discovery* 2(1):86–97. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.53.

Neal, Radford M. 2012. *Bayesian Learning for Neural Networks*. Springer Science & Business Media. Google-Books-ID: LHHrBwAAQBAJ.

Neuhaus, John, and Charles McCulloch. 2011. Generalized linear models. *WIREs Computational Statistics* 3(5):407–413. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/wics.175.

Newman, Robert H., Jin Zhang, and Heng Zhu. 2014. Toward a systems-level view of dynamic phosphorylation networks. *Frontiers in Genetics* 5: 263.

Norman, Utku, and A. Ercument Cicek. 2019. ST-Steiner: a spatio-temporal gene discovery algorithm. *Bioinformatics* 35(18):3433–3440.

Oates, Chris J., Jim Korkola, Joe W. Gray, and Sach Mukherjee. 2014. Joint estimation of multiple related biological networks. *The Annals of Applied Statistics* 8:1892–1919.

Olivier, Magali, Monica Hollstein, and Pierre Hainaut. 2010. TP53 Mutations in Human Cancers: Origins, Consequences, and Clinical Use. *Cold Spring Harbor Perspectives in Biology* 2(1):a001008.

O'Reilly, Francis J., and Juri Rappsilber. 2018. Cross-linking mass spectrometry: methods and applications in structural, molecular and systems biology. *Nature Structural & Molecular Biology* 25(11):1000–1008. Number: 11 Publisher: Nature Publishing Group.

Pandey, Akhilesh, WikiPathways Maintenance Bot, Nathan Salomonis, Michiel Adriaens, Alex Pico, Kristina Hanspers, NetPath, Christine Chichester, Zahra Roudbari, Lauren J. Dupuis, Egon Willighagen, Eric Weitz, and Finterly Hu. 2023. EGF/EGFR signaling pathway. `https://www.wikipathways.org/instance/WP437`.

Park, YongKeun, Christian Depeursinge, and Gabriel Popescu. 2018. Quantitative phase imaging in biomedicine. *Nature Photonics* 12(10): 578–589. Number: 10 Publisher: Nature Publishing Group.

Patil, Ashwini, Yutaro Kumagai, Kuo-ching Liang, Yutaka Suzuki, and Kenta Nakai. 2013. Linking Transcriptional Changes over Time in Stimulated Dendritic Cells to Identify Gene Networks Activated during the Innate Immune Response. *PLOS Computational Biology* 9(11):e1003323.

Pawson, T., and N. Warner. 2007. Oncogenic re-wiring of cellular signaling pathways. *Oncogene* 26(9):1268–1275.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Pico, Alexander R., Thomas Kelder, Martijn P. van Iersel, Kristina Hanspers, Bruce R. Conklin, and Chris Evelo. 2008. WikiPathways: Pathway Editing for the People. *PLOS Biology* 6(7):e184. Publisher: Public Library of Science.

Ranganath, Rajesh, Sean Gerrish, and David Blei. 2014. Black Box Variational Inference. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics*, 814–822. PMLR. ISSN: 1938-7228.

Rauluseviciute, Ieva, Finn Drabløs, and Morten Beck Rye. 2019. DNA methylation data by sequencing: experimental approaches and recommendations for tools and pipelines for data analysis. *Clinical Epigenetics* 11(1):193.

Recursion Pharmaceuticals. 2023a. Approach. `https://web.archive.org/web/20230514150945/https://www.recursion.com/approach`.

———. 2023b. RxRx. `https://web.archive.org/web/20230514073900/https://www.rxrx.ai/`.

Regev, Aviv, William Silverman, and Ehud Shapiro. 2000. Representation and simulation of biochemical processes using the π-calculus process algebra. In *Biocomputing 2001*, 459–470. WORLD SCIENTIFIC.

Ritz, Anna, Allison N. Tegge, Hyunju Kim, Christopher L. Poirel, and T. M. Murali. 2014. Signaling hypergraphs. *Trends in Biotechnology* 32(7):356–362.

Sachs, Karen, Omar Perez, Dana Pe'er, Douglas A. Lauffenburger, and Garry P. Nolan. 2005. Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. *Science* 308(5721):523–529.

Salimans, Tim, and David A. Knowles. 2014. On Using Control Variates with Stochastic Approximation for Variational Bayes and its Connection to Stochastic Linear Regression. ArXiv:1401.1022 [stat].

Salvatier, John, Thomas V. Wiecki, and Christopher Fonnesbeck. 2016. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science* 2:e55.

Schaefer, Carl F., Kira Anthony, Shiva Krupa, Jeffrey Buchoff, Matthew Day, Timo Hannay, and Kenneth H. Buetow. 2009. PID: the Pathway Interaction Database. *Nucleic Acids Research* 37(Database issue):D674–D679.

Schoeberl, Birgit, Claudia Eichler-Jonsson, Ernst Dieter Gilles, and Gertraud Müller. 2002. Computational modeling of the dynamics of the MAP kinase cascade activated by surface and internalized EGF receptors. *Nature Biotechnology* 20(4):370–375.

Shen, Ronglai, Adam B. Olshen, and Marc Ladanyi. 2009. Integrative clustering of multiple genomic data types using a joint latent variable model with application to breast and lung cancer subtype analysis. *Bioinformatics* 25(22):2906–2912.

Shojaie, Ali, and George Michailidis. 2010. Discovering graphical Granger causality using the truncating lasso penalty. *Bioinformatics* 26(18): i517–i523. `https://academic.oup.com/bioinformatics/article-pdf/26/18/i517/536841/btq377.pdf`.

Smith, Tim, and Kevin Ushey. 2023. aRrgh: a newcomer's (angry) guide to R. `https://web.archive.org/web/20230515051445/http://arrgh.tim-smith.us/`.

Smolen, Paul, Douglas A. Baxter, and John H. Byrne. 2000. Modeling transcriptional control in gene networks—methods, recent results, and future directions. *Bulletin of Mathematical Biology* 62(2):247–292.

Spencer, Simon E.F., Steven M. Hill, and Sach Mukherjee. 2015. Inferring network structure from interventional time-course experiments. *The Annals of Applied Statistics* 9:507–524.

Stoeckius, Marlon, Christoph Hafemeister, William Stephenson, Brian Houck-Loomis, Pratip K. Chattopadhyay, Harold Swerdlow, Rahul Satija, and Peter Smibert. 2017. Simultaneous epitope and transcriptome measurement in single cells. *Nature Methods* 14(9):865–868. Number: 9 Publisher: Nature Publishing Group.

Stuart, Joshua. 2023. PARADIGM | Systems Biology Group. `https://web.archive.org/web/20230607223156/https://sysbiowiki.soe.ucsc.edu/paradigm`.

Subramanian, A., P. Tamayo, V. K. Mootha, S. Mukherjee, B. L. Ebert, M. A. Gillette, A. Paulovich, S. L. Pomeroy, T. R. Golub, E. S. Lander, and J. P. Mesirov. 2005. Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* 102(43):15545–15550.

Terry M. Therneau, and Patricia M. Grambsch. 2000. *Modeling survival data: Extending the Cox model*. New York: Springer.

The Cancer Genome Atlas Network. 2012a. Comprehensive genomic characterization of squamous cell lung cancers. *Nature* 489(7417):519–525.

———. 2012b. Comprehensive molecular characterization of human colon and rectal cancer. *Nature* 487(7407):330–337.

———. 2012c. Comprehensive molecular portraits of human breast tumours. *Nature* 490(7418):61–70.

———. 2014. Comprehensive molecular characterization of urothelial bladder carcinoma. *Nature* 507(7492):315–322.

The Cancer Genome Atlas Network, John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, and Joshua M Stuart. 2013. The Cancer Genome Atlas Pan-Cancer analysis project. *Nature Genetics* 45(10):1113–1120.

Tipping, Michael E. 2001. Sparse Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research* 1(Jun):211–244.

Udell, Madeleine. 2015. Generalized Low Rank Models. Ph.D. thesis, Stanford University.

Upadhya, Swathi Ramachandra, and Colm J. Ryan. 2023. Antibody reliability influences observed mRNA–protein correlations in tumour samples. *Life Science Alliance* 6(8). Publisher: Life Science Alliance Section: Research Articles.

Vaske, Charles J., Stephen C. Benz, J. Zachary Sanborn, Dent Earl, Szeto Christopher, Jingchun Zhu, David Haussler, and Joshua M. Stuart. 2010a. Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using paradigm. *Bioinformatics* 26:i237–i245.

Vaske, Charles J., Stephen C. Benz, J. Zachary Sanborn, Dent Earl, Christopher Szeto, Jingchun Zhu, David Haussler, and Joshua M. Stuart. 2010b. Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using PARADIGM. *Bioinformatics* 26(12):i237–i245.

Wang, Bo, Aziz M. Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains, and Anna Goldenberg. 2014. Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* 11(3):333–337. Number: 3 Publisher: Nature Publishing Group.

Welch, Joshua D., Velina Kozareva, Ashley Ferreira, Charles Vanderburg, Carly Martin, and Evan Z. Macosko. 2019. Single-Cell Multi-omic Integration Compares and Contrasts Features of Brain Cell Identity. *Cell* 177(7):1873–1887.e17. Publisher: Elsevier.

Werhli, Adriano V., and Dirk Husmeier. 2007. Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology* 6.

White, Frames, Michael Abbott, Miha Zgubic, Jarrett Revels, Seth Axen, Alex Arslan, Simeon Schaub, Nick Robinson, Yingbo Ma, Gaurav Dhingra, Will Tebbutt, David Widmann, Niklas Heim, Niklas Schmitz, Christopher Rackauckas, Carlo Lucibello, Rainer Heintzmann, frankschae, Andreas Noack, Keno Fischer, Alex Robson, cossio, Jerry Ling, mattBrzezinski, Rory Finnegan, Andrei Zhabinski, Daniel Wennberg, Mathieu Besançon, and Pietro Vertechi. 2023. Juliadiff/chainrules.jl: v1.49.0. https://doi.org/10.5281/zenodo.7870094.

Wolf, Daisy, and Vijay Pande. 2023. Hey Tech, It's Time To Build. In Healthcare. | Andreessen Horowitz. https://web.archive.org/web/20230515020712/https://a16z.com/2023/04/20/hey-tech-its-time-to-build-in-healthcare/.

Wu, Mike, and Noah Goodman. 2018. Multimodal Generative Models for Scalable Weakly-Supervised Learning. In *Advances in Neural Information Processing Systems*, vol. 31. Curran Associates, Inc.

Wysocka, Magdalena, Oskar Wysocki, Marie Zufferey, Dónal Landers, and André Freitas. 2023. A systematic review of biologically-informed deep learning models for cancer: fundamental trends for encoding and interpreting oncology data. ArXiv:2207.00812 [cs, q-bio].

Xu, Yang, Priyojit Das, and Rachel Patton McCord. 2022. SMILE: mutual information learning for integration of single-cell omics data. *Bioinformatics* 38(2):476–486.

Yang, Zi, and George Michailidis. 2016. A non-negative matrix factorization method for detecting modules in heterogeneous omics multi-modal data. *Bioinformatics* 32(1):1–8.

Young, Matthew D., Matthew J. Wakefield, Gordon K. Smyth, and Alicia Oshlack. 2010. Gene ontology analysis for RNA-seq: accounting for selection bias. *Genome Biology* 11(2):R14.

Zhang, Yang, and Mingzhou Song. 2013. Deciphering interactions in causal networks without parametric assumptions. *arXiv* arXiv:1311.2707. `1311.2707`.

Zhou, Manqi, Hao Zhang, Zilong Bai, Dylan Mann-Krzisnik, Fei Wang, and Yue Li. 2023. Single-cell multi-omic topic embedding reveals cell-type-specific and COVID-19 severity-related immune signatures. Pages: 2023.01.31.526312 Section: New Results.

Zhuang, Xiaowei. 2021. Spatially resolved single-cell genomics and transcriptomics by imaging. *Nature methods* 18(1):18–22.

Zou, Hui, Trevor Hastie, and Robert Tibshirani. 2007. On the "degrees of freedom" of the lasso. *The Annals of Statistics* 35:2173–2192.