Tradict enables accurate prediction of eukaryotic transcriptional states from 100 marker genes

Supplemental Information

Surojit Biswas, Konstantin Kerner, Paulo José Pereira Lima Teixeira, Jeffery L. Dangl, Vladimir Jojic, Philip A. Wigge

Supplemental Analysis 1 - Our training transcriptomes are reflective of biology and are of high technical quality

We manually annotated metadata for 1,626 (62.6%, *A. thaliana*) and 6,682 (32.1%, *M. musculus*) of the training transcriptomes for both organisms, and found that the major drivers of variation were tissue and developmental stage (Figure 1a-b, main text). The first three principal components of our training collection explained a substantial proportion of expression variation for each organism (43.1% *A. thaliana*, 39.3% *M. musculus*). For A. thaliana PC1 was primarily aligned with the physical axis of the plant, with above ground, photosynthetic tissues having lower PC1 scores and below ground, root tissues having higher PC1 scores. Interestingly, samples found intermediate to the major below- and above-ground tissue clusters consisted of seedlings grown in constant darkness or mutant seedlings (e.g. det1, pif, phy) compromised for photomorphogenesis. Thus, PC1 can also be considered to align with light perception and signaling. By contrast, PC2 represented a developmental axis, with more embryonic tissues (seeds, endosperms) having lower PC2 scores, and more developed tissues having higher PC2 scores (Figure 1a, main text).

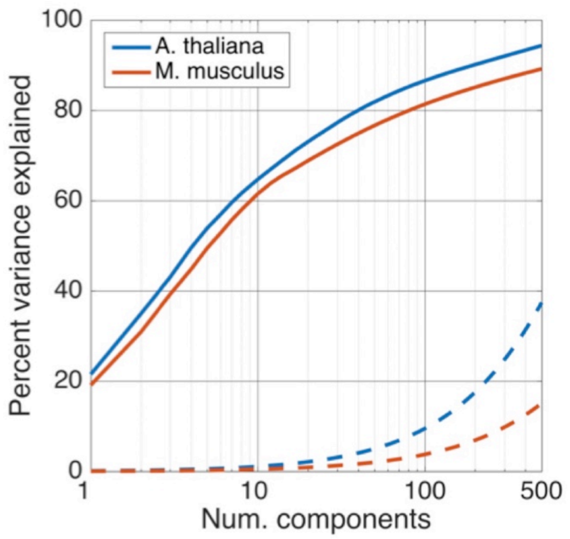

Figure S1. The eukaryotic transcriptome is compressible. The transcriptome is of low dimensionality, with 100 principal components able to explain 80% or more of expression variation. Dotted lines illustrate cumulative expression variation explained on a null model realization, where each gene's expression vector was permuted to break correlative ties to other genes.

For *M. musculus,* PC1 described a hematopoetic-nervous system axis. Cardiovascular, digestive, respiratory, urinary and connective tissues were found intermediate along this axis, and with the exception of liver tissue, were not differentiable along the first three PCs. Interestingly, as observed for *A. thaliana*, PC2 represented a developmental axis, with general "stemness" decreasing with increasing PC2 score. Consistent with this trend, nervous tissue from embryos and postnatal mice had consistently lower PC2 scores than mature nervous tissue. We did not find any significant correlation between *Xist* expression and any of the top twenty PCs, suggesting that sex was not a major driver of global gene expression relative to tissue and developmental context. This is consistent with findings reported in Crowley *et al.* (2015)[1].

To understand the compressibility of our training transcriptome collection beyond the first three PCs, we examined the percent of expression variation explained by subsequent components. Strikingly, we found the first 100 principal components were sufficient to explain 86.6% and 81.4% of expression variation in the observed transcriptomes for *A. thaliana* and *M. musculus*, respectively. By contrast, the first 100 principal components of a null model realization, in which the expression vectors for each gene were independently permuted, could only explain 5-10% of expression variation for both organisms (Figure S1). Given the phylogenetic distance spanned by *A. thaliana* and *M. musculus*, this transcriptomic compressibility is likely a shared property of all eukaryotes.

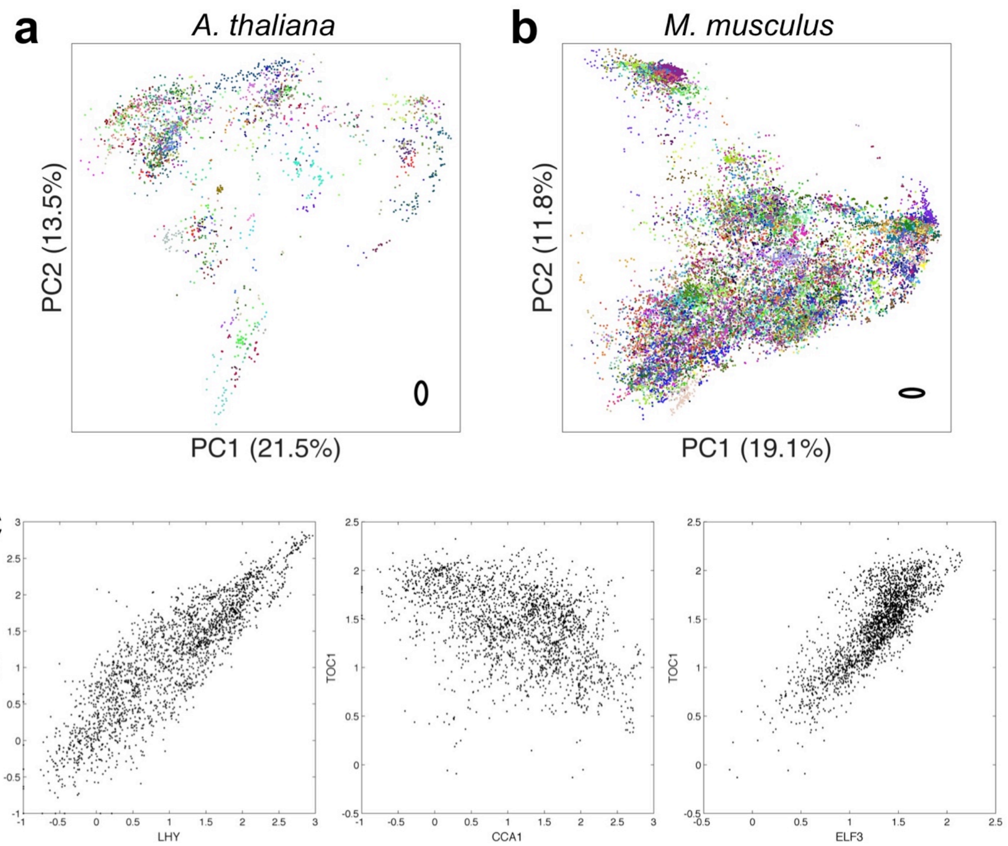

Figure S2. Our training collection is of high technical quality. Two dimensional principal components analysis for a) *A. thaliana* and b) *M. musculus*, where each sample is colored by the submission it belongs to. Note that while multiple submissions may have similar colors, each expression cluster contains many submissions. Bold, black ovals in the bottom left of each plot illustrate two standard deviation covariances for the median variance submission. c) Expression of late and early elements of the *A. thaliana* circadian clock matches expectations. Scatter plots of *LHY, CCA1,* and *ELF3* expression across all observed transcriptomes. *LHY* and CCA1 expression is activated by TOC1. CCA1 and LHY protein inhibits *TOC1* and *ELF3* transcription.

To further assess the quality and representativeness of our training collection, we examined the distribution of SRA submissions across the expression space, compared inter-submission variability within and between tissues, inspected expression correlations among genes with well established regulatory relationships, and assessed the evolution of the expression space across time. Technical variation due to differences in laboratory procedures across labs is difficult assess since this requires two different labs to perform the same, equivalently aimed experiment. Nevertheless, for both organisms, each tissue or development specific cluster was supported by multiple submissions, and importantly, inter-submission variability within a tissue or developmental context was significantly smaller than inter-tissue/developmental stage variability (p-value = 1.23e-16, F-test; Figures S2a-b). We also compared the expression of *ELF3*, *LHY*, and *TOC1* -- early and late elements of the *A. thaliana* circadian clock -- and found strong correlation in their expression with a direction and magnitude that fit established expectations (Figure S2c)[2].

We next performed a temporal rarefaction analysis. We compared (measured by Pearson correlation) how past distributions of samples along each of the first 100 principal components compared to their present distribution. Figures S3a-b illustrate that the expression space stabilized 2-3 years ago, and that new transcriptome samples that are added to the SRA tend to fall within already established clusters. We further note that the amount of usable transcriptomic data deposited on the SRA, and hence the representativeness of our sample, is increasing exponentially (Figure S4).

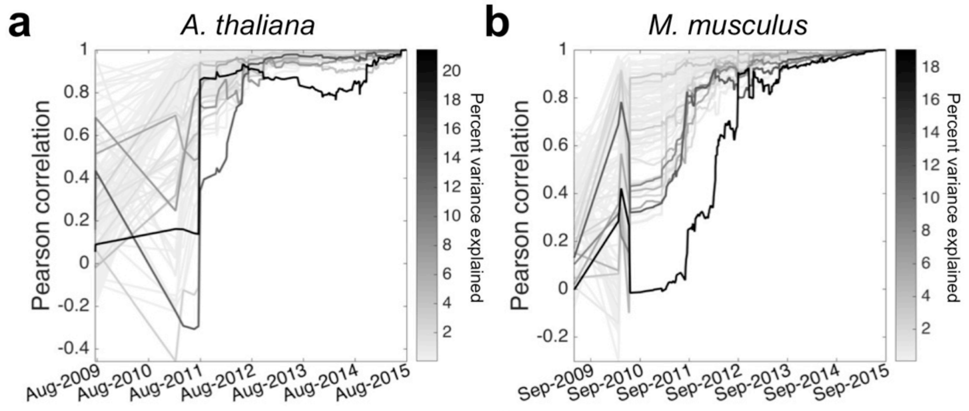

Figure S3. The expression space has stabilized. For each of the first 100 principal components (PCs), depicted is the Pearson correlation between how samples were distributed along the PC at a select point in the past and how they are distributed currently. Each line, representing a PC, is shaded by the percent variance explained by that PC. a) *A. thaliana*. b) *M. musculus*.

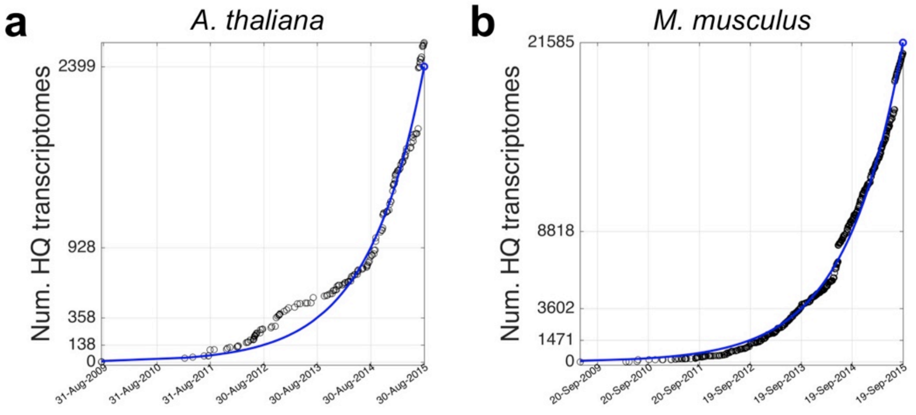

Figure S4. The number of high quality transcriptomes deposited in the SRA is growing exponentially. SRA growth for a) *A. thaliana*, and b) *M. musculus*.

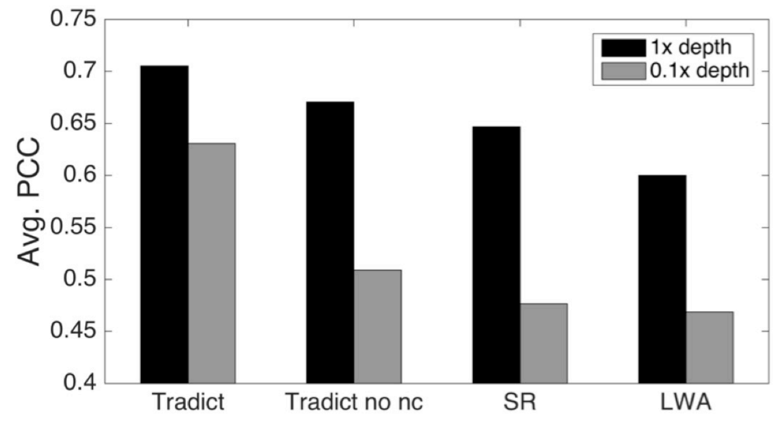

Figure S5. Tradict outperforms leading methods and is robust to noise. Tradict was trained on the first (historically speaking) 90% of SRA submissions and then tasked with predicting the remaining 10% of "test-set" submissions. a) Average intra-submission Pearson correlation coefficients between predicted and actual expression of genes (left) and transcriptional programs (tr. programs; right) in the test-set as a function of the number of markers used in the model. b) Intra-submission prediction accuracy of gene expression on the same test-set processed normally or rarefied to 0.1x depth. 'Tradict no nc' uses the same algorithm as Tradict, however, a diagonal covariance is used over markers, instead of a full one.

Supplemental Analysis 2 - Tradict outperforms leading approaches and is robust to noise from low sequencing depth and/or corrupted marker measurements

Baseline descriptions: As baselines for Tradict, we considered three alternative approaches. The first two, locally weighted averaging (LWA) and structured regression (SR) are the two best performing methods used in Donner *et al.* (2012)[3]. LWA, a non-parametric and non-linear approach, formulates predictions as weighted averages of the entire training set, where weights are determined by the distance between a query set of marker expressions and the expression of those markers in a training transcriptome. The exact weighting function is given by a Gaussian kernel, whose bandwidth we learn through cross-validation. This method is conceptually similar to nearest-neighbor based imputation methods in that predictions of gene expression come in the form of weighted averages of neighbor transcriptomes. In Donner *et al.* (2012), LWA performed superiorly to a simple nearest neighbor approach. In contrast, SR selects markers and predicts expression using regularized regression and the $L_{0,\infty}$ objective. The appropriate level of regularization is again learned through cross-validation. Given these methods were built for use on microarray data and hence their dependence on normality, we applied them to a log-transformed version of our training collection (log[TPM + 0.1]).

In the third baseline (Tradict Shallow-Seq), we employ Tradict as usual; however, we restrict Tradict's selected markers to be the 100 most abundant genes in the transcriptome. This provides a control for Tradict's marker selection algorithm, and simulates a situation that would be typical of shallow sequencing, where only the most abundant genes are used to make conclusions about the rest of the transcriptome.

Figure 3e in the main text illustrates a performance comparison between Tradict and these three methods.

Robustness to noise: We noticed that though Tradict iteratively selects markers to maximize explanatory power, these markers are not orthogonal. Consequently, during inference of the marker latent abundances, on which all expression predictions are based, the internal covariance among the markers will be used during estimation. In increasing data (larger sequencing depth, higher *a priori* abundance) the latent abundance inference will place less emphasis on this internal covariance; however, in situations of measurement inadequcy or error, the internal covariance will help to learn the correct latent abundances, which in turn, should stabilize predictions in noisy situations. To test this hypothesis, we considered a version of Tradict, 'Tradict no nc' (noise correction), in which only the diagonal of the internal marker covariance was used, effectively decoupling marker abundances in Tradict's underlying model. We re-evaluated intra-submission prediction accuracy for all of the methods, excluding Tradict Shallow-Seq, on the same training and test set above using 100 markers. However this time, in order to simulate situations of high measurement error, we rarefied samples in the test set to 0.1x depth and evaluated each method's predicted (depth-normalized) expression accuracy; the original 1x depth values formed the basis of comparison. The 10th, 25th, 50th, 75th, and 90th percentiles of read depths in the 0.1x scenario were 0.65, 1.1, 2.1, 3.1, and 4.4 million reads, respectively -- all below the recommended depths for *A. thaliana*. 30-40% of the markers had zero abundance in nearly half of the samples. Figure 3b illustrates that though all methods perform worse at 0.1x depth, Tradict is least affected. Importantly, we notice that Tradict no nc's performance is substantially reduced at lower depth, confirming our hypothesis that the internal marker covariance provides a valuable source of noise correction.

Supplemental Analysis 3 - Tradict's limitations as revealed by error, power, and program annotation robustness analyses

I. Error analysis - We first performed an error analysis in order to better understand the factors that contribute toward incorrect predictions. As done previously in Supplemental Analysis 2, we partitioned our transcriptome collection for *A. thaliana* into a training set and test set by submission and historical date. Like before, in order to mimic Tradict's use in practice as closely as possible, the training set contained the first 90% of submissions (208 submissions comprised of 2,389 samples) deposited on the SRA, and the test set contained the remaining 10% (17 submissions comprised of 208 samples). We trained Tradict on the training set, and subsequently predicted program and gene expression in the test set using only the expression values of the selected markers as input. We evaluated test-set intra-submission performance using PCC and the normalized unexplained variance that Tradict's prediction could not account for. Mathematically, the normalized unexplained variance metric is the ratio of the residual variance divided by the total variance of the target:

$$\frac{Var(true\_expression - predicted\_expression)}{Var(true\_expression)}.$$

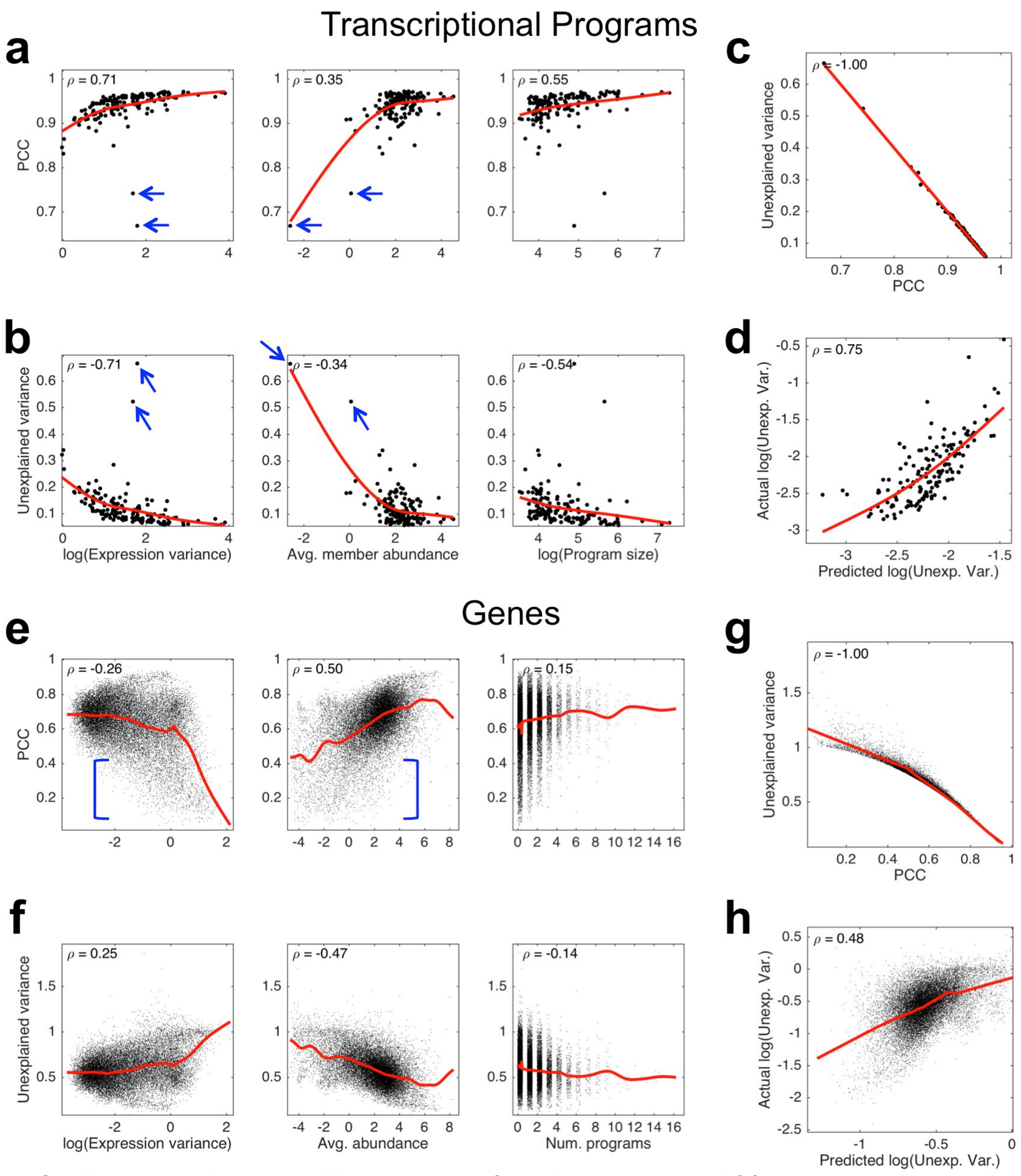

Figure S6. Error analysis reveals likely sources of prediction error. a) PCC between predicted and actual expression of transcriptional programs versus the logarithm of program expression variation (left), average abundance of genes within the program (middle), and the logarithm of the number of genes contained within the program. b) Same as (a) but with the proportion of unexplained variance as the measure of predictive performance instead of PCC. c) Relationship between PCC and unexplained variance. d) Actual log(unexplained variance) vs. predicted log(unexplained variance) based on a linear model that uses log(expression variation), average member abundance, and log(program size) as predictors of error. e-h) Same as (a-d) but for genes instead of programs. Here 'avg. abundance' denotes the average abundance of the gene, and 'num. programs' denote the number of programs the gene participates in. Spearman correlation coefficient (ρ) is noted in each plot. Red lines illustrate a cubic spline interpolation.

The above expression is equivalent to one minus the coefficient of determination between the prediction and the target. For each program, we then correlated these measures of performance to the magnitude of training-set expression variation, average training-set abundance of constituent genes, and the number of genes contained within the program. Similarly, for each gene, we correlated the above measures of performance to the magnitude of training-set expression variation, average training-set abundance, and the number of programs in which the gene participates.

Figure S6a-b illustrate that the expression variance of the program correlates positively with better prediction performance. This makes intuitive sense, as it should be easier to understand marker-program covariance relationships and predict expression for those programs that vary more. We note, however, two outlier programs that have reasonably high expression variance, but low prediction accuracy (blue arrows, Fig. S6a-b). These programs are composed of lowly expressed genes (Fig S6a-b, middle), suggesting that the mean expression level of genes contained within a program also positively correlate with Tradict's ability to predict that program's expression. Finally, we note that the more genes contained within the program, the easier it is to accurately predict (Fig S6a-b, right).

We built a linear model to model prediction accuracy -- as measured by log(unexplained variance) -- of a program as a function of its log(expression variance), average member abundance (as log-latent abundances), and log(program size). This model could predict log(unexplained variance) with a Spearman correlation coefficient of 0.75, suggesting that the three studied variables account for most of Tradict's errors (Fig. S6d). We note that our performance measures -- unexplained variance and PCC -- are nearly perfectly correlated in rank (Fig. S6c), and thus the above results also apply for the PCC performance criterion.

We performed a similar characterization for gene expression prediction. Unexpectedly, we found that better performance negatively correlated with increasing training-set expression variance, but only weakly so (Fig. S6e-f, left, ρ ~ 0.25). Further examination of poorly predicted, high variance genes revealed that these genes were largely lowly expressed (Fig. S6e-f, middle, blue brackets). Generally, measurements of lowly expressed genes tend to be contaminated with technical noise, making marker-gene covariance relationships difficult to estimate. Additionally, many of these genes generally have zero expression except for in a small subset of rarely sampled tissues (e.g. flower and bud, as opposed to leaf). This logistic-like distribution contributes strongly to training-set variance, but may make it difficult for Tradict, a linear method in the log-latent space, to train and predict accurately. We did not notice a strong correlation between prediction performance and the number of programs the gene participates in (Fig S6e-f, right).

This latter result is not unexpected. Though it is conceptually nice to think of Tradict making gene expression predictions by conditioning on program expression predictions, statistically these predictions are decoupled (see "Tradict - mathematical details" at the end of this document). Thus, there is no direct, statistical reason or methodological artifact as to why gene expression prediction accuracy should co-vary with the number of programs the gene is contained within. This result is important as it suggests that Tradict's gene expression predictions are robust to the choice of transcriptional program annotation used.

As was done for programs, we attempted to account for the log(unexplained variance) of Tradict's gene expression predictions using a linear model with the following predictors: log(expression variance), mean (log-latent) abundance, and the number of programs the gene participates in. We could not achieve the same explanatory power for genes as we did for programs, but we could still predict prediction error with a Spearman correlation of 0.48. Like before, we note a near perfect (up to 2-decimal precision) rank-correlation between our performance criterion, PCC and unexplained variance (Fig S6g).

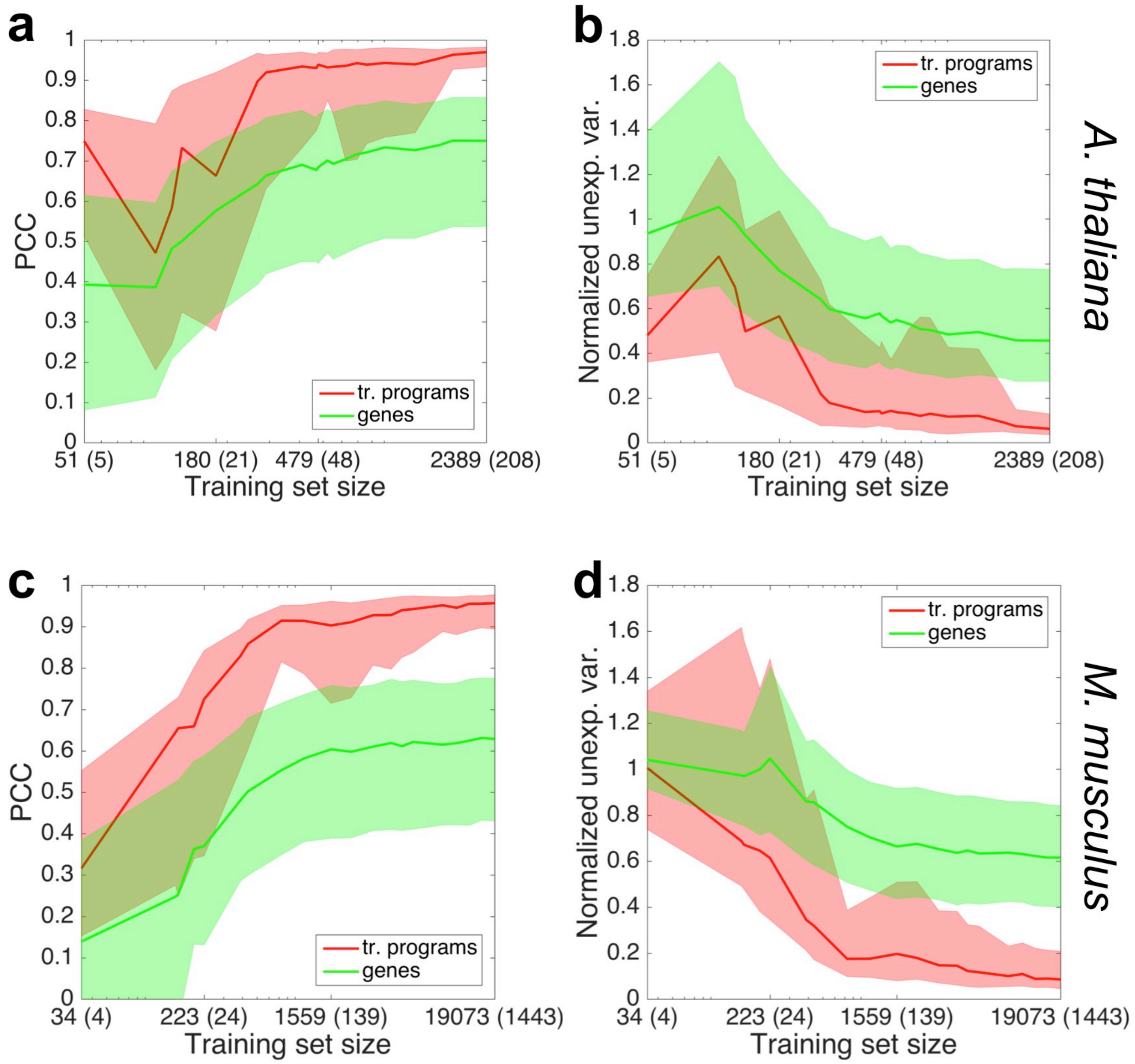

Figure S7. Power analysis reveals Tradict needs approximately 1000 samples to make accurate predictions. Test-set prediction accuracies in the form of a) PCC or b) normalized unexplained variance as a function of the size of the *A. thaliana* training set. X-axis tick labels are in the form of "Y (Z)" where Y denotes the number of samples in the training set and Z denotes the number of unique submissions to which these training set samples belong. The solid line depicts the median program (red) or gene (green) and the shaded error bands denote the 20th and 80th percentile program or gene. c-d) same as (a) and (b) but for *M. musculus*. Plots in (a) and (c) are plotted on a base 10 logarithmic scale.

II. Power Analysis - We next performed a power analysis in which we examined the number of samples required for Tradict to achieve its best prediction accuracy. As done previously, we partitioned our transcriptome collection for both *A. thaliana* and *M. musculus* into a training set and test set by submission and historical date. The training set contained the first 90% of submissions (208 submissions comprised of 2,389 samples for *A. thaliana*, and 1,443 submissions comprised of 19,703 samples for *M. musculus*) deposited on the SRA, and the test set contained the remaining 10% (17 submissions comprised of 208 samples for *A. thaliana*, and 159 submissions comprised of 1,774 samples for *M. musculus*).

We then trained Tradict using different sized subsets of the training set and evaluated its predictive performance on the test set using the PCC and normalized unexplained variance criteria. The different sized subsets were chosen sequentially such that each subsequent subset included the submissions in the previous subset as well as more recent submissions (by date) to the SRA. Consequently, this analysis aims to mimic reality in that it shows how Tradict's prospective test-set performance increases as more samples are submitted to the SRA.

Figure S7 shows that for both performance criterion and for both organisms, predictive performance begins to saturate for nearly all programs and genes after 750-1,000 samples are included in the training set. We note that not just any collection of 1,000 samples will do. These samples must be sufficiently varied in context in order for Tradict to perform adequate training over the variety possible transcriptomic states. By the same token, the first 1,000 samples to the SRA were likely not chosen to maximize exploration of the transcriptome. Thus, it may be possible to generate training sets that maximize Tradict's performance with much fewer than 1,000 samples. However, this latter hypothesis requires further investigation.

The requirement for 1,000 samples is already met for many commonly studied model organisms. Below are listed several eukaryotic model organisms and the number of publicly available samples that are available for it on the SRA (current as of September 23, 2016).

6.9K	*A. thaliana*
110.6K	*M. musculus*
8.6K	*D. melanogaster*
5.7K	*S. cerevisiae*
72.1K	*H. sapiens* (public)
2.7K	*C. elegans*
18.1K	*D. Rerio*

Supplemental Table S5. Number of SRA RNA-Seq records for several major model organisms. Reproduced from Supplemental Table Excel document.

Investigators working with any of these model organisms should have enough samples (even after quality filtering) to reliably use Tradict. Importantly, they may add their own samples to the publicly available collection to make Tradict's predictions more accurate for their contexts of interest.

III. Program annotation robustness analysis - In order to examine the impact of how the gene assignments used to define transcriptional programs affect Tradict's performance we performed a program annotation robustness analysis. We first partitioned our transcriptome collection for both *A. thaliana* and *M. musculus* into a training set and test set by submission and historical date as done in the previous section. For each transcriptional program we then exchanged 0%, 1%, 2%, 5%, 10% 20%, 50%, 80%, or 100% of the genes annotated to be in the program for another equivalent number of genes from the transcriptome that were not in the program. This gene exchange mimics corruption in the annotation. For each of these adjusted annotations, we examined Tradict's test-set prediction performance in the form of PCC and normalized unexplained variance.

Figure S8a-b illustrates how the PCC and normalized unexplained variance performance metrics behave as a function of the percentage of genes exchanged from each program in the *A. thaliana* test-set. Both performance criteria for program expression prediction show near equivalent performance for up to a 20% mis-annotation rate, which in practice is a comfortable cushion, especially for well controlled annotations, such as GO and KEGG. After a 20% mis-annotation rate, the prediction accuracy for many (20-50%) programs begins to sharply deteriorate.

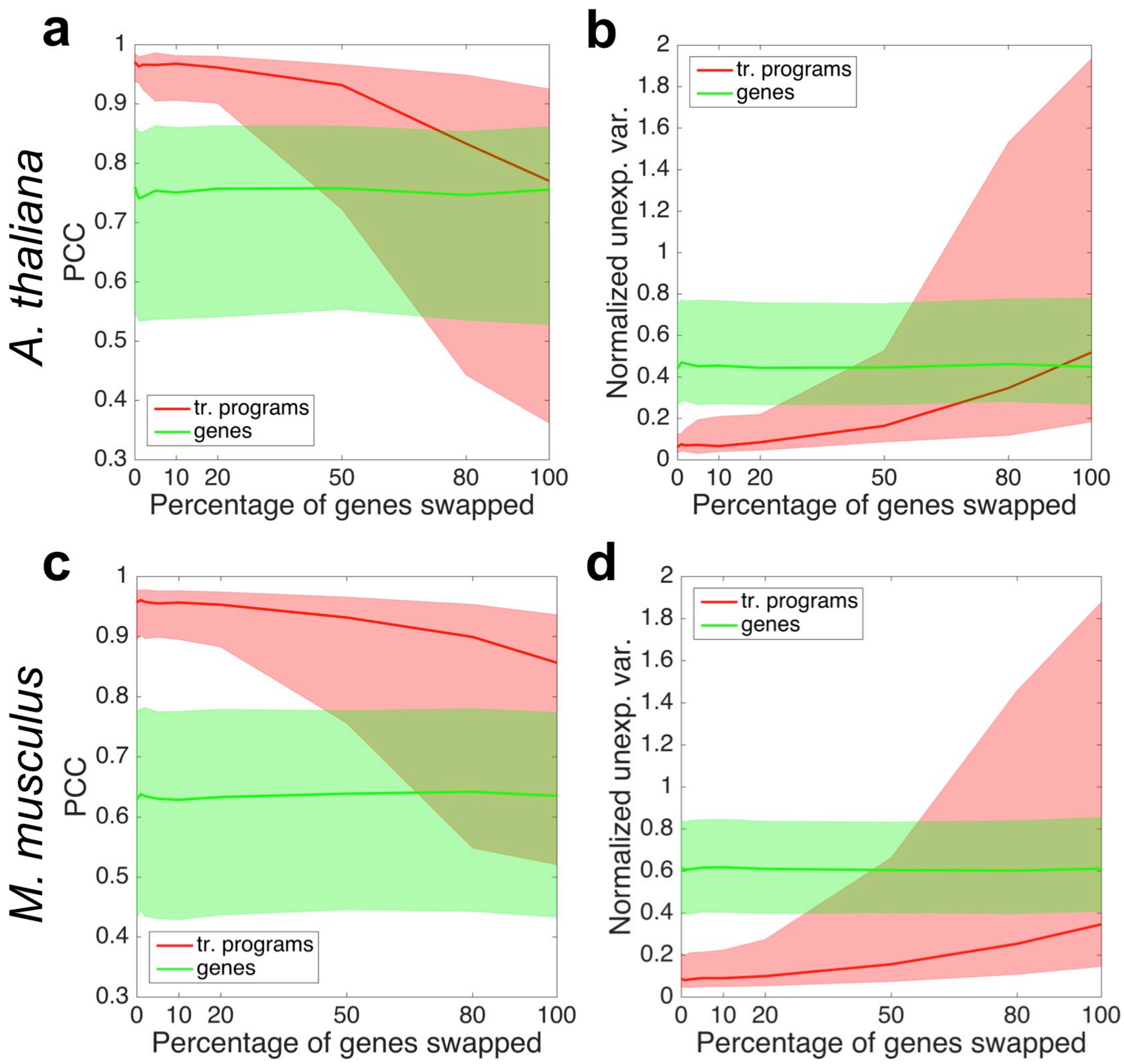

Figure S8. Tradict is robust with respect to the annotations used to define transcriptional programs. Test-set prediction accuracies in the form of a) PCC or b) normalized unexplained variance as a function of the percentage of genes randomly exchanged for each *A. thaliana* transcriptional program. The solid line depicts the median program (red) or gene (green) and the shaded error bands denote the 20th and 80th percentile program or gene. c-d) same as (a) and (b) but for *M. musculus*.

Interestingly, we note that even when 100% of genes in each program are exchanged for random ones during training, prediction PCC is high for many (>50%) of programs. To investigate this further, we examined the types of programs that maintain predictability versus those that lose it. Supplemental Table 6 shows that the programs that maintain high prediction accuracy are heavily enriched for global, transcriptionally far-reaching, "housekeeping" processes, and include processes related to growth, development, and metabolism. By contrast, the programs that are most sensitive to mis-annotation are those generally related to biotic and abiotic stress response regulons (e.g. response to light, and immune response).

We note that test-set gene expression prediction performance is invariant with respect to the level of program mis-annoation. This is expected because, as described in the "Error Analysis" section, Tradict's gene expression predictions are statistically decoupled from program expression prediction.

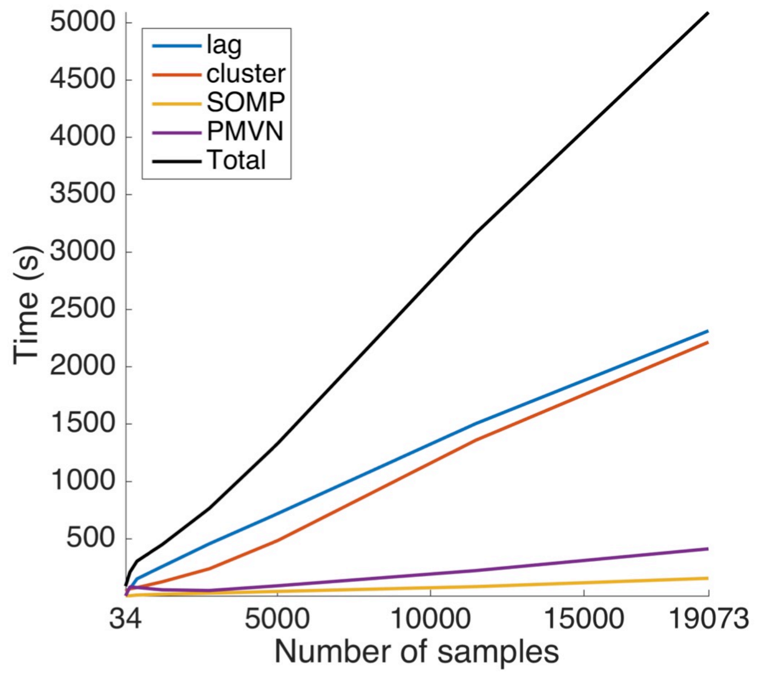

Figure S9. Timing analysis. Training time vs. training set size in terms of number of samples. Black line denotes the total training time and colored lines depict training times for each component of training. 'lag' (blue) and 'cluster' (orange) are the times needed to compute the lag transformation of the training set and to define and cluster the transcriptional programs, respectively. 'SOMP' (yellow) denotes the time required to perform the Simultaneous Orthogonal Matching Pursuit decomposition of the transcriptional programs, and 'PMVN' (purple) denotes the time required to learn the parameters of the Continuous-Poisson Multivariate Normal hierarchical model.

Supplemental Analysis 4 - Timing and memory requirements

We performed a training time analysis on the *M. musculus* transcriptome collection. Specifically, we recorded the time required to train Tradict as a function of the size of the training set in terms of the number of samples. Figure S9 illustrates these results, and shows that training time was approximately linear in the size of the input (0.25 seconds/sample). The largest bottlenecks during training come from lag-transforming the training-set and defining (computing the first principal component) and clustering the transcriptional programs for subsequent decomposition with Simultaneous Orthogonal Matching Pursuit. The range of training sample sizes explored here should be applicable for most contexts as the number publicly available samples for other model organisms (Supplemental Analysis 3.II) tend to be less than the number available for *M. musculus.* Additionally, the linear increase in time requirements suggests the method will scale well to larger datasets, with timing requirements in the hours range.

We also timed Tradict's prediction times. We found that prediction times were linear in the number of samples and that generating a prediction for each sample required 3.1 seconds. The limiting factor here comes from MCMC sampling of the conditional posterior distributions of each gene and program. We have also developed a subroutine that allows users to just obtain maximum *a posteriori* estimates of gene and program expression. This prediction task is considerably faster, only requiring 0.02 seconds per sample.

Tradict's peak memory usage during training scaled linearly with training input size. At the largest training set size examined (19,703 samples), peak memory consumption was 25.3 GB. Loading the training set expression matrix alone (values stored as double precision floats) consumed 5.2 GB of memory. Regressing peak memory consumption onto training-set size we found the equation MEMORY (GB) = 0.0011*NUM_SAMPLES + 5.2 described memory usage well.

All computations were performed using one core of a Lenovo P700 ThinkStation with two Intel Xeon E5-2620 v3 processors and 32 GB of DDR4 ECC RDIMM RAM.

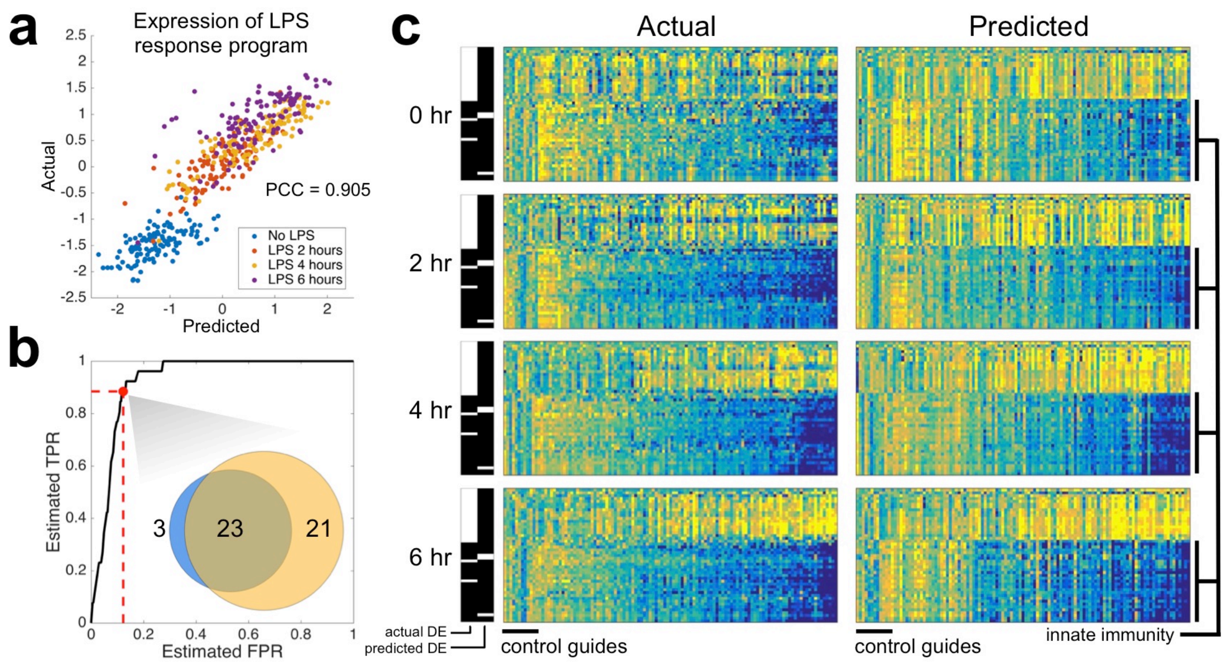

Figure S10. Tradict accurately predicts temporal transcriptional responses to lipopolysaccharide treatment in a dendritic cell line CRISPR library. a) Actual vs. predicted z-score standardized expression of the "response to lipopolysachharide" transcriptional program. Samples are colored by time point. b) Receiver operator characteristic (ROC) curve illustrating Tradict's accuracy for identifying differentially expressed (DE) transcriptional programs. Here the "truth set" was considered to be all DE programs with FDR < 0.01 based on actually measured expression values. The marked point along the ROC curve and the inset venn diagram depict the concordance between the predicted and actual set of DE transcriptional programs when an FDR threshold of 0.01 for predicted DE programs was also used. c) Predicted vs actual heatmaps of DE transcriptional programs (rows) across time for different CRISPR lines (columns). Here, DE programs included those found either in actuality or by prediction and are accordingly marked by the black and white indicator bars on the left of each sub-block. Columns of these heat maps represent different profiled lines. The first 12 correspond to negative control guides, whereas the remaining columns correspond to positive regulators of Tnf expression. The expression of programs in each sub-block is z-score normalized to their expression in the negative control guide lines. The bottom 26 programs are all of those directly related to innate immunity among the 368 programs we've defined for *M. musculus*. All heatmaps are clustered in the same order across time, genotype, and between predicted and actual.

Supplemental Analysis 5 - Tradict accurately predicts temporal dynamics of innate immune signaling in CRISPRed in primary immune cells

To further dissect Tradict's capabilities, we examined a *M. musculus* dataset from Parnas *et al.* (2015) in which one of the first CRISPR screens was performed on primary immune cells to look for regulators of tumor necrosis factor (Tnf) expression[5]. They found many positive regulators of Tnf expression and created clonal bone-marrow derived dendritic cell (BDMC) lines where each positive regulator was disrupted using CRISPR. They used shallow RNA-sequencing (2.75 +/- 1.2 million reads) to profile the transcriptomes of these lines for 6 hours after lipopolysaccharide (LPS) treatment.

We asked whether Tradict's predictions could quantitatively recapitulate actuality, despite the challengingly noisy marker measurements due to the low sequencing depth. To be specific, approximately 30% of the markers had zero measured expression in greater than 40%

of samples. After performing the batch correction described in Parnas *et al.* (2015), we examined the expression of the “response to lipopolysaccharide” transcriptional program. Figure S6a illustrates that despite the limitation on marker measurement accuracy, Tradict predicts response to LPS with a PCC accuracy of 0.905. Differential transcriptional program expression analysis revealed that DE programs based on Tradict’s predictions were highly concordant with those based on actual measurements (Figure S6b). Strikingly, programs found DE based on Tradict predictions included 92% of those directly related to innate immune signaling in mice.

We next examined the quantitative quality of Tradict predictions by observing how the DE programs found by either analysis of actual measurements or predictions behave across time. Figure S6c illustrates that despite the high marker measurement error, Tradict’s predictions are quantitatively concordant with actuality. As expected most lines of CRISPRed positive regulators demonstrate loss of innate immune signaling.

Materials and Methods

Data acquisition and transcript quantification

Data acquisition and transcript quantification were managed using a custom script, `srafish.pl`. The `srafish.pl` algorithm and its dependencies are described below. Complete instructions for installing (including all dependencies) and using `srafish.pl` are available on our GitHub page:
https://github.com/surgebiswas/transcriptome_compression/tree/master/data_download.

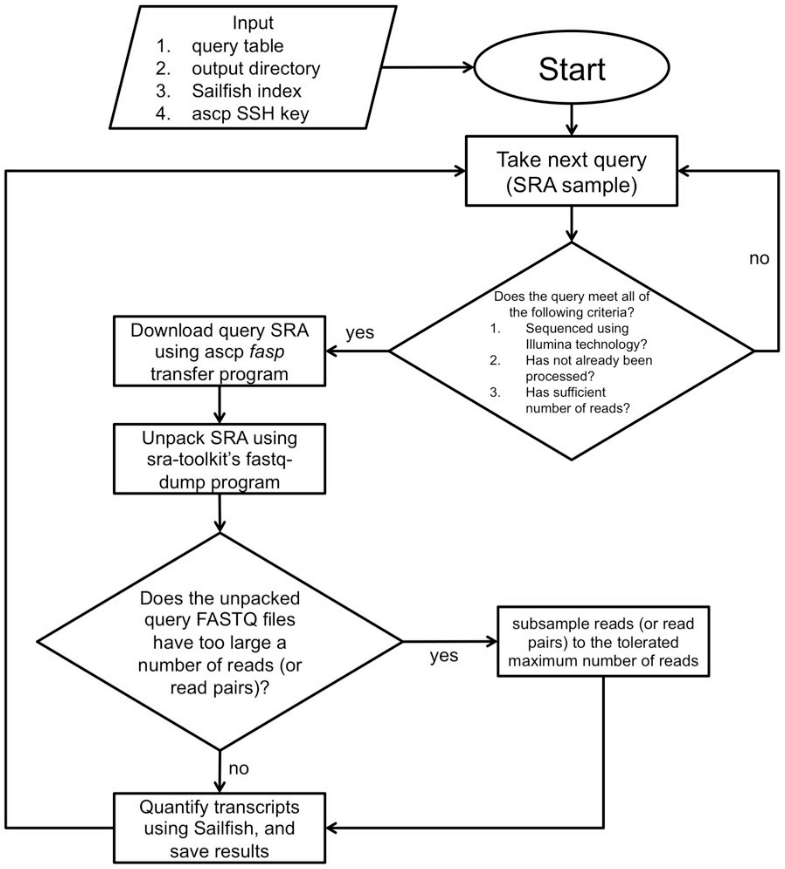
Input
1. query table
2. output directory
3. Sailfish index
4. ascp SSH key
Start
Take next query (SRA sample)
Does the query meet all of the following criteria?
1. Sequenced using Illumina technology?
2. Has not already been processed?
3. Has sufficient number of reads?
no
yes
Download query SRA using ascp *fasp* transfer program
Unpack SRA using sra-toolkit's fastq-dump program
Does the unpacked query FASTQ files have too large a number of reads (or read pairs)?
yes
subsample reads (or read pairs) to the tolerated maximum number of reads
no
Quantify transcripts using Sailfish, and save results

Figure SM1. Algorithmic workflow of data acquisition and quantification as implemented by `srafish.pl`.

Figure SM1 illustrates the workflow of `srafish.pl`. Briefly, after checking it meets certain quality requirements, `srafish.pl` uses the ascp *fasp* transfer program to download the raw sequence read archive (.sra file) for an SRA RNA-Seq sample. Transfers made using `ascp` are substantially faster than traditional FTP. The .sra file is then unpacked to FASTQ format using the `fastq-dump` program provided with the SRA Toolkit (NCBI)[11]. The raw FASTQ read data is then passed to Sailfish[12], which uses a fast alignment-free algorithm to quantify transcript abundances. To preserve memory, files with more than 40 million reads for *A. thaliana* and 70 million reads for *M. musculus* are downsampled prior to running Sailfish. Samples with fewer than 4 million reads are not downloaded at all. This workflow is then iterated for each SRA RNA-Seq sample available for the organism of interest.

The main inputs into `srafish.pl` are a query table, output directory, Sailfish index, and ascp SSH key, which comes with each download of the aspera ascp client. `srafish.pl` depends on Perl (v5.8.9 for Linux x86-64), the aspera ascp client (v3.5.4 for Linux x86-64), SRA Toolkit (v2.5.0 for CentOS Linux x86-64), and Sailfish (v0.6.3 for Linux x86-64).

Query table construction
For each organism, using the following (Unix) commands, we first prepared a "query table" that contained all SRA sample ID's as well as various metadata required for the download:

```
qt_name=<query_table_file_name>
sra_url=http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?save=efetch&db=sra&rettype=ru
ninfo&term=
organism=<organism_name>
wget -O $qt_name '$url($organism[Organism]) AND "strategy rna seq"[Properties]'
```

Where fields in between <> indicate input arguments. As an example,

```
qt_name=Athaliana_query_table.csv
sra_url=http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?save=efetch&db=sra&rettype=ru
ninfo&term=
organism="Arabidopsis thaliana"
wget -O $qt_name '$url($organism[Organism]) AND "strategy rna seq"[Properties]'
```

Reference transcriptomes and index construction
Sailfish requires a reference transcriptome -- a FASTA file of cDNA sequences -- from which it builds an index it can query during transcript quantification. For the *A. thaliana* transcriptome reference we used cDNA sequences of all isoforms from the TAIR10 reference. For the *M. musculus* transcriptome reference we used all protein-coding and long non-coding RNA transcript sequences from the Gencode vM5 reference.

Sailfish indices were created using the following command:

```
sailfish index -t <ref_transcriptome.fasta> -k 20 -p 6 -o .
```

Here, `<ref_transcriptome.fasta>` refers to the reference transcriptome FASTA file. Copies of the reference transcriptome FASTA files used in this study are available upon request.

Quality and expression filtering
Upon completion of download and transcript quantification of all samples, we assembled an n-samples x p-isoforms matrix of transcripts per million (TPM) values as calculated by Sailfish. We then proceeded to quality and expression filter the data as follows:

1. We first removed samples with a read depth and mapping rate below 4 million reads and 0.75 respectively for *A. thaliana* and 4 million reads and 0.70 for *M. musculus* (Figure SM2a-b). We used a slightly lower mapping rate threshold for *M. musculus* because the average mapping rate for *M. musculus* was lower than that of *A. thaliana*. We reasoned this was due to the fact that the Gencode vM5 reference is likely less complete than the TAIR10 reference for *M. musculus.* Though these read count thresholds may be considered slightly lower than what is ideal for both organisms, raising them much higher removed a large number of samples from analysis. Importantly, low read count samples should only add to the noise in the dataset, and so

the performance results presented in the main text are, if anything, artificially lower than they should be.

2. Subsequently, we collapsed the isoform expression table into a gene expression table by setting a gene's expression to be the sum of the expression values of all isoforms of that gene.
3. We next removed all non-protein coding transcripts except for long non-coding RNAs, and removed samples with large amounts (>30%) of non-protein coding contamination (e.g. rRNA).
4. The dataset was then expression filtered by only keeping genes with expression greater than 1 TPM in at least 5% of all samples. The latter requirement ensures that outlier or extreme expression in just a few samples is not enough to keep the gene for analysis.
5. We then removed samples with an abnormally large number of genes with expression values of zero. To do this we calculated the mean and standard deviation of the number of genes with zero expression across all samples. Samples with the number of zero expressed genes greater than the mean plus two times the standard deviation were removed.
6. Finally, we removed outlier samples by first examining the proportion of zeros contained in each sample and by computing the pairwise Pearson correlation coefficient between the gene expression profiles of all samples. To improve heteroskedasticity, raw TPM values for each gene were converted to a log-scale ($\log_{10}$[TPM + 0.1]) prior to calculating correlations. For *A. thaliana,* the majority of samples had an average correlation with other samples of greater than 0.45 and fewer than 20% percent zero values. Samples with lower correlation or a greater percentage of zeros were removed (Figure SM2c). By similar arguments, samples with less average correlation than 0.55 with other samples and greater than 30% zeros were removed for *M. musculus* (Figure SM2d). Manual inspection of ~100 of these samples revealed they were highly enriched for non-polyA selected samples and samples made from low-input RNA (e.g. single-cells).

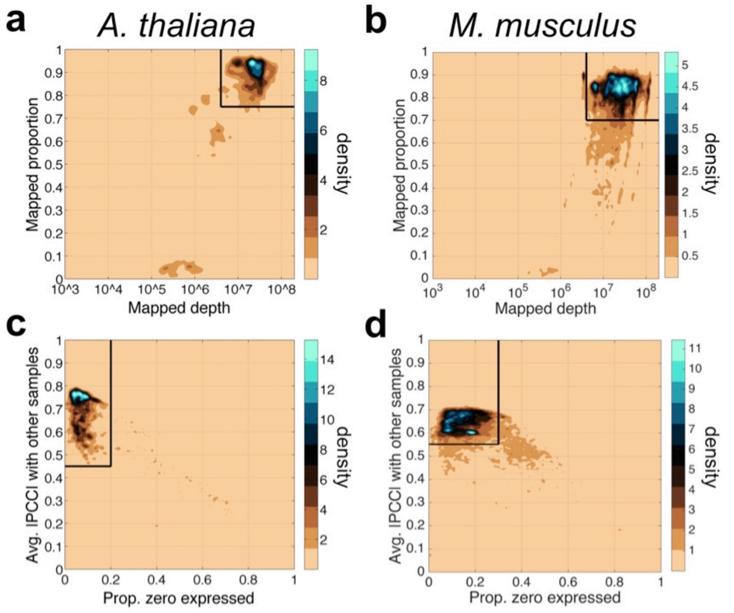

Figure SM2. Quality filtering thresholds for mapping depth and proportion (a,b), and for average correlation to other samples and proportion of zeros (c,d).

Metadata annotation

RNA-Seq samples are submitted to the SRA with non-standardized metadata annotations. For example, for some samples tissue and developmental stage are clearly noted as separate fields, whereas in others such information can only be found the associated paper's abstract or sometimes only in its main text. In order to ensure the maximum accuracy when performing metadata annotations, we annotated samples manually until the structure of the gene expression space represented by the first three principal components was clear. Annotation was accomplished by first finding those few submissions with samples in multiple clusters. These submissions revealed that the likely separating variables of interest were issue and developmental context. For each major cluster in the PCA (determined visually) we then annotated samples by size of their submission until the tissue or developmental context of that cluster became qualitatively clear.

Tradict algorithm

Tradict's usage can be broken down into two parts: 1) Training, and 2) Prediction. Training is the process of learning, from training data, the marker panel and its predictive relationship to the expression of transcriptional programs and to the remaining genes in the transcriptome. In essence, during training we begin with full transcriptome data and collapse its information into a subset of marker genes. Prediction is the reverse process of predicting the expression of transcriptional programs and non-marker genes from the expression measurements of just the selected markers.

Our training algorithm can be broken down into several steps: 1) Computing the latent logarithm of the training transcriptome collection, 2) defining transcriptional programs, 3) marker selection via Simultaneous Orthogonal Matching Pursuit, 4) building a predictive Multivariate Normal Continuous-Poisson hierarchical model.

1) **Computing the latent logarithm of the training transcriptome collection -** Expression values in our training dataset are stored as transcripts per million (TPM), which are non-negative, variably scaled, and strongly heteroscedastic, similar to read counts. For subsequent steps in our algorithm and analysis it will be important transform this data to improve its scaling and heteroscedasticity.

   Often, one log transforms such data. However, to avoid undefined values where the data are zeros, one also adds a pseudocount (e.g. 1). This pseudocount considers neither the gene's *a priori* abundance nor the confidence with which the measurement was made, making this practice convenient but statistically unfounded. In previous work, we introduced the latent logarithm, or "lag"[13]. lag assumes that each observed expression value is actually a noisy realization of an unmeasured *latent abundance*. By taking the logarithm of this latent abundance, which considers both sampling depth and the gene's *a priori* abundance, lag provides a more nuanced and statistically principled alternative to the conventional "log(x + pseudocount)". In increasing data, lag quickly converges to log, but in the absence of it, lag relies on both sampling depth and the gene's *a priori* abundance to make a non-zero estimate of the gene's latent abundance.

   With these intuitions in mind, we apply the lag transformation to our entire training dataset. The lag-transformed expression matrix demonstrated a Pearson correlation of 0.98 to the log(TPM + 0.1) transformed expression matrix for both *A. thaliana* and *M. musculus*, but again, especially for samples with 0 expression, lag is able to make better estimates of their true abundance in the log-domain.

   Availibility: https://github.com/surgebiswas/latent_log.git

2) **Defining transcriptional programs -** We define a transcriptional program to be the first principal component of the z-score standardized lag expression of the set of genes involved in a certain response or pathway[14,15]. This virtual program marker maximally captures (in one dimension) the information contained in the transcriptional program. We considered three criteria for defining a globally comprehensive, but interpretable list of transcriptional programs for *A. thaliana* and *M. musculus*:

    a) In order to capture as much information about the transcriptome as possible, we wanted to maximize the number of genes covered by the transcriptional programs.
    b) In order to improve interpretability, we wanted to minimize the total number of transcriptional programs.
    c) The number of genes in a transcriptional program should not be too large or too small -- genes in a transcriptional program should be in the same pathway.

    Rather than defining these transcriptional programs *de novo*, we took a knowledge-based approach and defined them using Gene Ontology (GO). We also tried using KEGG pathways, but found these were less complete and nuanced than GO annotations. Gene Ontology is made of three sub-ontologies or aspects: Molecular Function, Biological Process, and Cellular Component. Each of these ontologies contains terms that are arranged as a directed acyclic graph with the above three terms as roots. Terms higher in the graph are less specific than those near the leaves[16,17]. Thus, with respect to the three criteria above, we wanted to find GO terms with low-to-moderate height in the graph such that they were neither too specific nor too general. Given we were interested in monitoring the status of different processes in the organism, we focused on the Biological Process ontology.

    We downloaded gene association files for *A. thaliana* and *M. musculus* from the Gene Ontology Consortium (http://geneontology.org/page/download-annotations). We then examined for each of several minimum and maximum GO term sizes (defined by the number of genes annotated with that GO term) the number of GO terms that fit this size criterion and the number of genes covered by these GO terms.

    Supplemental Tables 1 and 2 contain the results of this analysis for *A. thaliana* and *M. musculus*, respectively. *A. thaliana* has 3333 GO annotations for 27671 genes. We noticed that when the minimum GO term size was as small as it could be (1) and we moved from a maximum GO term size of 5000 to 10000, we jumped from covering 18432 genes (67% of the transcriptome) to covering the full transcriptome (black bolded two rows of Supplemental Table 1). This is due to the addition of one GO term, which was the most general, "Biological Process," term. Thus, we concluded that 33% of the genes in the transcriptome have only "Biological Process" as a GO annotation, and therefore that we do not need to capture these genes in our GO term derived gene sets. Though these genes are not informatively annotated, we Tradict still model their expression all the same. We hereafter refer to the set of genes annotated with more than just the "Biological Process" term as *informatively annotated*.

    We reasoned that a minimum GO term size of 50 and a maximum size of 2000, best met our aforementioned criteria for defining globally representative GO term derived gene sets. These size thresholds defined 150 GO terms, which in total covered 15124 genes (82.1% of the informatively annotated, and 54.7% of the full transcriptome). These 150 GO-term derived, globally comprehensive transcriptional programs covered the major pathways related to growth, development, and response to the environment.

    We performed a similar GO term size analysis for *M. musculus*. *M. musculus* has 10990 GO annotations for 23566 genes. Of these genes, 6832 (29.0%) had only the "Biological

Process" term annotation and were considered not informatively annotated. As we did for *A. thaliana*, we selected a GO term size minimum of 50 and a maximum size of 2000. These size thresholds defined 368 GO terms, which in total covered 14873 genes (88.9% of the informatively annotated, 63% of the full transcriptome). As we found for *A. thaliana,* these 368 GO-term derived, globally comprehensive transcriptional programs covered the major pathways related to growth, development, and response to the environment.

Supplemental Tables 3 and 4 contain the lists of the globally comprehensive transcriptional programs as defined by the criteria above. For each of these programs, we then computed its first principal component over all constituent genes.

3) **Marker selection via Simultaneous Orthogonal Matching Pursuit -** After defining transcriptional programs we are left with a #-training-samples x #-transcriptional-programs table of expression values. We decompose this matrix using an adapted version of the Simultaneous Orthogonal Matching Pursuit, using the #-training-samples x #-transcriptional-programs table as a dictionary[18,19]. Because transcriptional programs are often correlated with other programs, we first cluster them using consensus clustering[20,21], which produces a robust and stable clustering by taking the consensus of many clusterings performed by a *base* clustering algorithm. 100 independent iterations of K-means are used as the base-clusterings, and the number of clusters is determined using the Davies-Bouldin criterion[22]. The decomposition is greedy, in which during each iteration, the algorithm first finds the transcriptional program cluster with the largest unexplained variance. It then finds the gene contained within this cluster of transcriptional programs with the maximum average absolute correlation to the expression of all transcriptional programs. This gene is then added to an "active set," onto which the transcriptional program expression matrix is orthogonally projected. This fit is subtracted to produce a residual, on which the above steps are repeated until a predefined number of genes have been added to the active set or the residual variance of the transcriptional program expression matrix falls below some predefined threshold.

4) **Building a predictive Multivariate Normal Continuous-Poisson hierarchical model**

Here we describe conceptually how we fit a predictive model that allows us to predict gene and transcriptional program expression from expression measurements of our selected markers. Readers interested in the full mathematical details of the Multivariate Normal Continuous-Poisson hierarchical model are referred to the attached "Tradict - mathematical details" document.

The Multivariate Normal Continuous Poisson distribution offers us a way of modeling statistically coupled count based or, more generally, non-negative random variables, such as the TPM or count-based expression values of genes[23–27]. Here it is assumed the TPM expression of each gene in a given sample is a noisy, Continuous-Poisson realization of some unmeasured latent abundance, the logarithm of which comes from Multivariate-Normal distribution over the log-latent abundances of all genes in the transcriptome.

Given the marginalization properties of the multivariate normal distribution, we are only interested in learning relationships between the selected markers and non-marker genes. For the purposes of prediction, we need to estimate 1) the mean vector and 2) covariance matrix over the log-latent TPMs of the markers, 3) the mean vector of the log-latent TPMs of the non-markers, and 4) cross-covariance matrix between the log-latent TPMs of markers and non-markers.

Note that before we can estimate these parameters, we must learn the log-latent TPMs of all genes. To do this we first lag-transform the entire training dataset. We then learn the marker log-latent TPMs, and their associated mean vector and covariance matrix using an iterative conditional modes algorithm. Specifically, we initialize our estimate of the marker log-latent TPMs to be the lag-transformed expression values, which by virtue of the lag's probabilistic assumptions are also derived from a Normal Continuous-Poisson hierarchical model. We then iterate 1) estimation of the mean vector and the covariance matrix given the current estimate of log-latent TPMs, and 2) maximum *a posteriori* estimation of log-latent TPMs given the estimated mean vector, covariance matrix, and the measured TPM values of the selected markers. A small regularization is added during estimation of the covariance matrix in order to ensure stability and to avoid infinite-data-likelihood singularities that arise from singular covariance matrices. This is most often happens when a gene's TPM abundance is mostly zero (i.e. there is little data for the gene), giving the multivariate normal layer an opportunity to tightly couple this gene's latent abundance to that of another gene, thereby producing a nearly singular covariance matrix.

Learning the mean vector of the non-marker genes and the marker x non-marker cross-covariance matrix is considerably easier. For the mean vector, we simply take the sample mean of the lag-transformed TPM values. For the cross-covariance matrix we compute sample cross-covariance between the learned log-latent marker TPMs and the log-latent non-marker TPMs obtained from the lag transformation. We find hat these simple sample estimates are highly stable given that our training collection includes thousands to tens of thousands of transcriptomes.

Using similar ideas, we can also encode the expression of the transcriptional programs. Recall that a principal component output by PCA is a linear combination of input features. Thus by central limit theorem, the expression of these transcriptional programs should behave like normal random variables. Indeed, after regressing out the first 3 principal components computed on the entire training samples x genes expression matrix from the expression values of the transcriptional programs (in order to remove the large effects of tissue and developmental stage), 85-90% of the transcriptional programs had expression that was consistent with a normal distribution (average p-value = 0.43, Pearson's chi-squared test). Consequently, as was done for non-marker genes and as will be needed for decoding, we compute the mean vector of the transcriptional programs and the markers x transcriptional programs cross covariance matrix. These are given by the standard sample mean of the training transcriptional program expression values and sample cross-covariance between the learned log-latent TPMs of the markers and the transcriptional program expression values.

To perform prediction, we must translate newly obtained TPM measurements of our marker genes into expression predictions for transcriptional programs and the remaining non-marker genes. More specifically, we'd like to formulate these predictions in the form of conditional posterior distributions, which simultaneously provide an estimate of expression magnitude and our confidence in that estimate. To do this, we first sample the latent abundances of our markers from their posterior distribution using the measured TPMs, and the 1 x markers mean vector and markers x markers covariance matrix previously learned from the training data. This is done using Metropolis-Hastings Markov Chain Monte Carlo sampling (see "Tradict - mathematical details" attached to this document for greater details on tuning the proposal distribution, sample thinning, sampling depth, and burn-in lengths). Using these sampled latent abundances and the previously estimated mean vectors and cross-covariance matrices, we then can use standard Gaussian conditioning to sample the log-latent expression of the transcriptional programs and the remaining genes in the transcriptome from their conditional distribution. These samples, in aggregate, are samples from the conditional posterior distribution of each gene and program and can be used to approximate properties of this distribution (e.g. posterior means, and/or credible intervals).

References

1. Crowley, J. J. *et al.* Analyses of allele-specific gene expression in highly divergent mouse crosses identifies pervasive allelic imbalance. *Nat. Genet.* **47,** (2015).
2. Greenham, K. & McClung, C. R. Integrating circadian dynamics with physiological processes in plants. *Nat Rev Genet* **16,** 598–610 (2015).
3. Donner, Y., Feng, T., Benoist, C. & Koller, D. Imputing gene expression from selectively reduced probe sets. *Nat. Methods* **9,** (2012).
4. Gelman, A. *et al. Bayesian Data Analysis*. (Chapman & Hall, 2013).
5. Parnas, O., Jovanovic, M., Eisenhaure, M. & Zhang, F. A Genome-wide CRISPR Screen in Primary Immune Cells to Dissect Regulatory Networks. *Cell* **162,** 1–12 (2015).
6. New England BioLabs Inc. SplintR Ligase. at <https://www.neb.com/products/m0375-splintr-ligase>
7. Lohman, G. J. S., Zhang, Y., Zhelkovsky, A. M., Cantor, E. J. & Jr, T. C. E. Efficient DNA ligation in DNA – RNA hybrid helices by Chlorella virus DNA ligase. *Nucleic Acids Res.* 1–14 (2013). doi:10.1093/nar/gkt1032
8. Rohland, N. & Reich, D. Cost-effective, high-throughput DNA sequencing libraries for multiplexed target capture. *Genome Res.* **22,** 939–946 (2012).
9. Yang, L. *et al.* The Pseudomonas syringae type III effector HopBB1 fine tunes pathogen virulence by gluing together host transcriptional regulators for degradation. *Submitted* (2016).
10. Lundberg, D. S., Yourstone, S., Mieczkowski, P., Jones, C. D. & Dangl, J. L. Practical innovations for high-throughput amplicon sequencing. *Nat. Methods* **10,** 999–1002 (2013).
11. Leinonen, R., Sugawara, H. & Shumway, M. The Sequence Read Archive. **39,** 2010–2012 (2011).
12. Patro, R., Mount, S. M. & Kingsford, C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.* **32,** 462–4 (2014).
13. Biswas, S. The latent logarithm. *arXiv* 1–11 (2016).
14. Ma, S. & Kosorok, M. R. Identification of differential gene pathways with principal component analysis. *Bioinformatics* **25,** 882–889 (2009).
15. Fan, J. *et al.* Characterizing transcriptional heterogeneity through pathway and gene set overdispersion analysis. *Nat. Methods* **13,** 241–244 (2016).
16. Ashburner, M. *et al.* Gene Ontology: tool for the unification of biology. *Nat Genet* **25,** 25–29 (2000).
17. The Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic Acids Res.* **43,** D1049–D1056 (2015).
18. Tropp, J. a & Gilbert, A. C. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. *IEEE Trans. Inf. Theory* **53,** 4655–4666 (2007).
19. Tropp, J. a., Gilbert, A. C. & Strauss, M. J. Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing* **86,** 572–588 (2006).
20. Monti, S., Tamayo, P., Mesirov, J. & Golub, T. Consensus Clustering : A Resampling-Based Method for Class Discovery and Visualization of Gene Expression Microarray Data. *Mach. Learn.* **52,** 91–118 (2003).
21. Yu, Z., Wong, H.-S. & Wang, H. Graph-based consensus clustering for class discovery from gene expression data. *Bioinforma.* **23 ,** 2888–2896 (2007).
22. Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **2,** 224–227 (1979).
23. Aitchison, J. & Shen, S. M. Logistic-Normal Distributions: Some Properties and Uses. *Biometrika* **67,** 261 (1980).

24. Aitchison, J. & Ho, C. H. The multivariate Poisson-log normal distribution. *Biometrika* **76,** 643–653 (1989).
25. Biswas, S., Mcdonald, M., Lundberg, D. S., Dangl, J. L. & Jojic, V. Learning Microbial Interaction Networks from Metagenomic Count Data. in *Res. Comput. Mol. Biol.* **1,** 32–43 (2015).
26. Ho, C. H. & Kong, H. The multivariate Poisson-log normal distribution. **2,** (1989).
27. Madsen, L. & Dalthorp, D. Simulating correlated count data. *Environ. Ecol. Stat.* **14,** 129–148 (2007).

Tradict - mathematical details

Surojit Biswas, Konstantin Kerner, Paulo José Pereira Lima Texeira, Jeffery L. Dangl, Vladimir Jojic, Philip A. Wigge

Contents

This document describes the full mathematical details for the concepts presented in the "Tradict algorithm" section, "Building a predictive Multivariate Normal Continuous-Poisson hierarchical model" subsection of the Materials and Methods in the Supplemental Information. Specifically, we present exactly how Tradict uses a selected set of markers to 1) complete training, and 2) to perform prediction.

1 Preliminaries

For a matrix A, $A_{:i}$ and $A_{i:}$ index the i^{th} column and row, respectively. For a set of indices, q, we use $-q$ to refer to all indices not specified by q.

2 Model

Tradict uses a Continuous-Poisson Multivariate Normal (CP-MVN) hierarchical model to model the expression of transcriptional programs and all genes in the transcriptome. Multivariate Normal hierarchies have been explored in the past as a means of modeling correlation structure among count based random variables [1, 2, 3, 4]. However, given we will be working with abundances as transcripts per million (TPM), which are non-negative (can equal zero) and fractional, we relax the integral assumption of the Poisson so it is continuous on $[0, \infty)$. Specifically, we define the continuous relaxation of the Poisson distribution (hereafter, Continuous-Poisson) to have the following density function:

$$f(x|\lambda) = C_\lambda \frac{e^{-\lambda}\lambda^x}{\Gamma(x+1)}$$

where C_λ is a normalization constant. The mean of this distribution is given by λ, just as the Poisson.

We begin by building a predictive model of gene expression, and thereafter discuss a predictive model for the expression of transcriptional programs. Let z_j denote the log-*latent abundance* of gene j, such that $\exp(z_j)$ is the *latent abundance* of that gene (in TPM) whose measured abundance is given by t_j. Let $T_j = t_j o$ be the measured total number of transcripts of gene j. Here o is the sequencing depth in millions of reads of the sample under consideration. We assume then,

$$\begin{aligned} z &\sim \mathcal{N}(\mu, \Sigma) \\ T_j &\sim \text{Continuous-Poisson}(\exp(z_j)o) \end{aligned}$$

where μ and Σ are of dimension $1 \times$ #-genes and #-genes $\times$ #-genes, respectively. In effect, we are assuming that the measured number of transcripts for gene j is a noisy realization of a latent abundance $\exp(z_j)$ times the sequencing depth, o. The dependencies between log-latent abundances (the z_j's) are then encoded by the covariance matrix of the Multivariate Normal layer of the model.

Note that we could model the TPM measurements directly in the second layer by assuming $t_j \sim$ Continuous-Poisson$(\exp(z_j))$; however, this formulation does not consider sequencing depth, which can be a valuable source of information when inferring latent abundances for rare/poorly sampled genes [5].

During prediction, we are interested in building a predictive model between markers and all genes in the transcriptome. Therefore, we need to consider a conditional model of the transcriptome given the log-latent abundances of the markers. Let m be the set of indices for the given panel of selected markers, which are the subset of genes Tradict selects as representative of the transcriptome. To perform prediction we therefore need $p(z_{-m}|z_m)$. We have,

$$\begin{aligned} z_m &\sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)}) \\ z_{-m}|z_m &\sim \mathcal{N}(\mu_{z_{-m}|z_m}, \Sigma_{z_{-m}|z_m}) \\ T_j &\sim \text{Continuous-Poisson}(\exp(z_j)o) \end{aligned}$$

Here, $\mu^{(m)}$ and $\Sigma^{(m)}$ refer to mean vector and covariance matrix of z_m. Given these, the conditional mean of the log-latent abundances for all non-marker genes can be obtained through Gaussian conditioning. Specifically, for two normally distributed row-vector variables a and b the conditional mean of b given a is given by $\mu_{b|a} = \mu_b + (a - \mu_a)\Sigma_a^{-1}\sigma_{ab}$ and $\Sigma_{b|a} = \Sigma_b - \sigma_{ab}^T\Sigma_a^{-1}\sigma_{ab}$, where σ_{ab} is the cross-covariance between a and b, and Σ_a and Σ_b are the covariance matrices of a and b, respectively.

Given the expression of a transcriptional program is a linear combination of the latent abundances of its constituent genes, they will be normally distributed given 1) Central Limit Theorem, and 2) the latent abundances themselves are normally distributed (convolutions of normals are normals). Let s be the expression of all transcriptional programs. We posit the following model,

$$\begin{aligned} z_m &\sim \mathcal{N}\left(\mu^{(m)}, \Sigma^{(m)}\right) \\ s|z_m &\sim \mathcal{N}(\mu_{s|z_m}, \Sigma_{s|z_m}) \end{aligned}$$

To use these models for prediction, we must learn their parameters from training data. This would complete the process of training described in the Supplemental Information. Specifically, we need to learn $\mu^{(m)}$, $\Sigma^{(m)}$, μ_s, $\mu_{z_{-m}}$, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$.

3 Training

As described in the Supplemental Information, given an estimate of z_m, $\hat{z}_m$, inference of μ_s, $\mu_{z_{-m}}$, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$ is straightforward. In lag transforming the entire training TPM expression matrix, $t \in \mathbb{R}^{\text{samples}\times\text{genes}}$, we have an estimate of z, $\hat{z} = \text{lag}(t)$ [5]. Thus, an estimate of $\mu_{z_{-m}}$ is given by the usual column-wise sample mean of $\hat{z}_{-m}$.

Let $\Lambda \in \mathbb{R}^{\text{genes}\times\text{transcriptional programs}}$ be a matrix of principal component 1 coefficients over genes for each transcriptional program. Note, that $\Lambda_{ij} = 0$ if gene i is not in transcriptional program j. An estimate of s is given by $\hat{s} = \hat{z}\Lambda$, and so an estimate for μ_s, $\hat{\mu}_s$, is given by the usual column-wise mean of $\hat{s}$.

Given $\hat{z}_m$ the cross-covariances, $\sigma_{z_m,s}$ and $\sigma_{z_m,z_{-m}}$, are given by the usual sample cross-covariance between $\hat{z}_m$ and $\hat{s}$ and between $\hat{z}_m$ and $\hat{z}_{-m}$, respectively.

Now, though we could use the lag-transformed values of t_m as our estimate for z_m, we have an opportunity to improve this estimate by virtue of having to estimate $\mu^{(m)}$ and $\Sigma^{(m)}$. More specifically, given z_m, estimates of $\mu^{(m)}$ and $\Sigma^{(m)}$ are given by – up to some regularization – the usual sample mean and covariance of z_m. Furthermore, given $\mu^{(m)}$ and $\Sigma^{(m)}$, we can update our estimate of z_m to the maximum of its posterior distribution. This suggests an alternating iterative procedure in which we iterate 1) estimation of $\mu^{(m)}$ and $\Sigma^{(m)}$, and 2) maximum *a posteriori* inference of z_m until convergence of their joint likelihood. It is the $\hat{z}_m$ that we obtain from this procedure that we use in the cross-covariance calculations above. The following section details this procedure.

3.1 Inference of z_m given $\mu^{(m)}$ and $\Sigma^{(m)}$

Suppose Tradict has estimates of $\mu^{(m)}$ and $\Sigma^{(m)}$ given by $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$, and let $T_m = t_m(o \times \mathbf{1}_{1\times\text{markers}})$ be a matrix of the total measured number of transcripts for each marker. Here $o \in \mathbb{R}^{\text{samples}\times 1}$ is a vector of sample sequencing depths in millions of reads. Given these, we would like to calculate the maximum *a posteriori* (MAP) estimate of $\hat{z}_m = \text{argmax}_{z_m}\, p(z_m|o, T_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})$.

The posterior distribution over z_m is given by

$$\begin{aligned}
p(z_m|o, T_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) &= \frac{p(T_m|o, z_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_m|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})}{\int_k p(T_m|o, k, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(k|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})\mathrm{d}k} \\
&\propto \prod_{i=1}^{n} p(T_{im}|o, z_{im}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_{im}|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \prod_{i=1}^{n} \left[\prod_{j=1}^{|m|} C_{[\exp(z_{ij})o_i]}[\exp(z_{ij})o_i]^{T_{ij}} e^{-[\exp(z_{ij})o_i]} / \Gamma(T_{ij}+1) \right] \\
&\times \frac{1}{\sqrt{2\pi|\hat{\Sigma}^{(m)}|}^{|m|}} \exp\left(-\frac{1}{2}(z_{i:} - \hat{\mu}^{(m)}) \text{inv}\left(\hat{\Sigma}^{(m)}\right)(z_{i:} - \hat{\mu}^{(m)})^T \right)
\end{aligned}$$

where for notational clarity we have used $\text{inv}(\cdot)$ to represent matrix inverse.

Given z is a matrix parameter, this may be difficult to solve directly. However, note that given z_{ij}, T_{ij} is conditionally independent of $T_{i,-j}$. Additionally, given $z_{i,-j}$, z_{ij} is normally distributed with mean and covariance

$$\begin{aligned}
a_{ij} &= \mu_j^{(m)} + \left(z_{i,-j} - \mu_{-j}^{(m)}\right) \text{inv}\left(\Sigma_{-j,-j}^{(m)}\right) \Sigma_{-j,j}^{(m)} \\
\sigma_{m(j)} &= \Sigma_{j,j}^{(m)} - \Sigma_{j,-j}^{(m)} \text{inv}\left(\Sigma_{-j,-j}^{(m)}\right) \Sigma_{-j,j}^{(m)}
\end{aligned}$$

respectively. Taken together, this suggests an iterative conditional modes algorithm [6] in which we maximize the posterior one column of z at a time, while conditioning on all others.

Let $\hat{z}_m$ denote our current estimate of z_m. Let $m(j)$ denote the index of the j^{th} marker and let $m(-j)$

denote the indices of all markers but the j^{th} one. The above sub-objective is given by,

$$\begin{aligned}
\hat{z}_{im(j)} &= \underset{z_{im(j)}|z_{im(-j)}}{\operatorname{argmax}} \log p(z_{im(j)}|T_{im(j)}, o_i, \hat{z}_{im(-j)}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{im(j)}|z_{im(-j)}}{\operatorname{argmax}} \log p(T_{im(j)}|z_{im(j)}, o_i, \hat{z}_{im(-j)}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) p(z_{im(j)}|\hat{z}_{im(-j)}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{im(j)}|z_{im(-j)}}{\operatorname{argmax}} \log p(T_{im(j)}|z_{im(j)}, o_i) p(z_{im(j)}|\hat{z}_{im(-j)}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\
&= \underset{z_{im(j)}|z_{im(-j)}}{\operatorname{argmax}} \log \left[[\exp(z_{im(j)}) o_i]^{T_{im(j)}} e^{-[\exp(z_{im(j)}) o_i]} \exp\left(-\frac{1}{2\sigma_{m(j)}} (z_{im(j)} - a_{im(j)})^2 \right) \right] \\
&= \underset{z_{im(j)}|z_{im(-j)}}{\operatorname{argmax}} T_{im(j)} \exp(z_{im(j)}) o_i - \exp(z_{im(j)}) o_i - \frac{1}{2\sigma_{m(j)}} (z_{im(j)} - a_{im(j)})^2
\end{aligned}$$

Differentiating we get,

$$\begin{aligned}
&\frac{\partial}{\partial z_{im(j)}} T_{im(j)} z_{im(j)} o_i - \exp(z_{im(j)}) o_i - \frac{1}{2\sigma_{m(j)}} (z_{im(j)} - a_{im(j)})^2 \\
&\qquad = T_{im(j)} o_i - \exp(z_{im(j)}) o_i - \frac{1}{\sigma_{m(j)}} (z_{im(j)} - a_{im(j)})
\end{aligned}$$

Because $z_{im(j)}$ appears as a linear and exponential term, we cannot solve this gradient analytically. We therefore utilize Newton-Raphson optimization. For this we also require the Hessian, which is given by,

$$\begin{aligned}
&\frac{\partial}{\partial z_{im(j)}} T_{im(j)} o_i - \exp(z_{im(j)}) o_i - \frac{1}{\sigma_{m(j)}} (z_{im(j)} - a_{im(j)}) \\
&\qquad = -\exp(z_{im(j)}) o_i - \frac{1}{\sigma_{m(j)}} < 0
\end{aligned}$$

Notice the Hessian is always negative-definite, which implies each update has a single, unique optimum.

In practice, the Newton-Raphson updates can be performed in vectorized fashion iteratively for each column of z. We generally find that this optimization takes 5-15 iterations (full passes over all columns of z) and less than a minute to converge. We refer to the program that performs these calculations as $\hat{z}_m = \texttt{MAP\_z}\left(t, o, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$.

3.2 Complete inference of $\mu^{(m)}$, $\Sigma^{(m)}$, and z_m

For complete inference we use the following iterative conditional modes algorithm [6]:

- Initialize $T_m = t_m(o \times \mathbf{1}_{1\times\text{markers}})$, $\hat{z}_m = \text{lag}(t_m)$.
- Until convergence of $\log p(T_m|o, \hat{z}_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) + \log p(\hat{z}_m|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})$, iterate:
  - Update $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$:

$$\begin{aligned}
\hat{\mu}^{(m)} &= \frac{1}{\#\text{samples}} \sum_i \hat{z}_{im} \\
\hat{\Sigma}^{(m)} &= \frac{1}{\#\text{samples} - 1} \sum_i (\hat{z}_{im} - \hat{\mu}^{(m)})^T (\hat{z}_{im} - \hat{\mu}^{(m)}) + \lambda \text{diag}\left[\text{cov}\left(\hat{z}_m^{(\text{init})}\right)\right]
\end{aligned}$$

  - Update $\hat{z}_m = \texttt{MAP\_z}\left(t, o, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$.

Here $\mathrm{diag}(x)$ of the square matrix x returns an equivalently sized matrix with only the diagonal of x preserved and 0's for the off-diagonal terms. $\mathrm{cov}(\cdot)$ denotes the usual sample covariance matrix.

Note that in this algorithm we have added a regularization to the estimate of the covariance matrix. This is done in order to ensure stability and to avoid infinite-data-likelihood singularities that arise from singular covariance matrices. This is most often happens when a gene's TPM abundance is mostly zero (i.e. there is little data for the gene), giving the multivariate normal layer an opportunity to increase the data likelihood (via the determinant of the covariance matrix) by tightly coupling this gene's latent abundance to that of another gene, thereby producing a singularity. This regularization is probabilistically equivalent to adding an Inverse-Wishart prior over $\Sigma^{(m)}$. The parameter λ controls the strength of the regularization. In practice, we find $\lambda = 0.1$ leads to good predictive performance, stable (non-singular) covariance matrices, and reasonably quick convergence.

4 Prediction

During prediction we are given new measured TPM measurements for our markers, $t^*_m \in \mathbb{R}^{\text{query samples} \times |m|}$, and we must make predictions about the expression of all transcriptional programs and the remaining non-marker genes. We have two options available to us: 1) Calculate a point (MAP) estimate or 2) calculate the complete posterior distribution over each non-marker gene and transcriptional program in a fully Bayesian manner. The former option is faster, but the second gives more information on the uncertainty of the prediction. We therefore implement both options in Tradict and detail their derivation below. Note that knowing the entire posterior distribution allows one to derive whatever estimator they would like, and so option 2, informationally speaking, supersets option 1.

4.1 MAP estimation of gene and program abundances

We first need an estimate of the log-latent abundances $\hat{z}^*_m$ associated with t^*_m. Given the estimates $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$ obtained from the training data, we obtain these estimates as

$$\hat{z}^*_m = \texttt{MAP\_z}\left(t^*_m, \mathbf{1}_{\text{query samples}\times 1}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}\right)$$

Given the inferred marker latent abundances, we let our estimates of s^* and t^*_m be the maximizers of their probability distribution. In other words, $\hat{s}^* = \mathrm{argmax}_{s^*}\, p(s^*|\hat{z}^*_m)$ and $\hat{t}^*_m = \mathrm{argmax}_{t^*_m}\, p(t^*_m|\hat{z}^*_m)$.

Our estimate for the expression of all transcriptional programs is given by

$$\underset{s^*}{\mathrm{argmax}}\, p(s^*|\hat{z}^*_m) = \mathbb{E}[s^*|\hat{z}^*_m] = \mu_{s^*|\hat{z}^*_m} = \hat{\mu}_s + \left(\hat{z}^*_m - \hat{\mu}^{(m)}\right) \mathrm{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,s}.$$

Here, $\hat{\mu}_s$ and $\hat{\sigma}_{z_m,s}$ represent estimates of the unconditional mean of s and the cross-covariance matrix between z_m and s previously learned during training.

Similarly, for the entire transcriptome we have,

$$\hat{t}^*_{ij} = \underset{t}{\mathrm{argmax}}\, p(t|\hat{z}^*_{im}) = \exp\left(\mu_{z_{ij}|\hat{z}^*_{im}}\right).$$

where,

$$\mu_{z_{ij}|\hat{z}^*_{im}} = \hat{\mu}_j + \left(\hat{z}^*_{im} - \hat{\mu}^{(m)}\right) \mathrm{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_j}$$

We could also use the expected value of t as our estimate.

$$\begin{aligned}
\mathbb{E}[t^*_{ij}|\hat{z}^*_{im}] &= \int_{-\infty}^{\infty} \mathbb{E}[t^*_{ij}|z^*_{ij}]p(z_{ij}|\hat{z}^*_{im})\mathrm{d}z_{ij} \\
&= \int_{-\infty}^{\infty} \exp(z_{ij})\mathcal{N}(z_{ij}|\mu_{z_{ij}|\hat{z}^*_{im}}, \Sigma_{z_{ij}|\hat{z}^*_{im}})\mathrm{d}z_{ij} \\
&= \mathbb{E}_{\mathcal{N}}[\exp(z_{ij})|\hat{z}^*_{im}]
\end{aligned}$$

The Moment Generating Function of a Normal random variable X with mean μ and variance σ^2 is given by $M(t) = \mathbb{E}[\exp(tX)] = \exp(\mu t + \sigma^2 t^2/2)$. Therefore we have,

$$\mathbb{E}[t^*_{ij}|\hat{z}^*_{im}] = \mathbb{E}_{\mathcal{N}}[\exp(z_{ij})|\hat{z}^*_{im}] = M(1) = \exp\left(\mu_{z_{ij}|\hat{z}^*_{im}} + \frac{1}{2}\Sigma_{z_{ij}|\hat{z}^*_{im}}\right)$$

where,

$$\begin{aligned}\mu_{z_{ij}|\hat{z}^*_{im}} &= \hat{\mu}_j + \left(\hat{z}^*_{im} - \hat{\mu}^{(m)}\right) \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_j} \\ \Sigma_{z_{ij}|\hat{z}^*_{im}} &= \hat{\sigma}_{jj} - \hat{\sigma}^T_{z_m,z_j} \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_j}\end{aligned}$$

Here, $\hat{\mu}_j$ and $\hat{\sigma}_{z_m,z_j}$ represent estimates of the unconditional mean of z_j and the cross-covariance matrix between z_m and z_j. These were learned from the training data during encoding.

Though this predictor is unbiased, it does not produce a good prediction for most samples. This is due to the right-skew of the Poisson, which drags its mean away from the most likely values.

4.2 Posterior density estimation of gene and program abundances

The above predictions represent point estimates. Ideally, we would like to know the uncertainty around these estimates. Given measurements of the representative markers, we can estimate the posterior distribution of expression values for transcriptional programs and the non-markers, and therein calculate any point estimates and/or measures of uncertainty. Recall that for transcriptional programs:

$$\begin{aligned} z_m &\sim \mathcal{N}\left(\mu^{(m)}, \Sigma^{(m)}\right) \\ s|z_m &\sim \mathcal{N}(\mu_{s|z_m}, \Sigma_{s|z_m}) \end{aligned}$$

And similarly for genes (among which the marker genes are included) we have:

$$\begin{aligned} z_m &\sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)}) \\ z_{-m}|z_m &\sim \mathcal{N}(\mu_{z_{-m}|z_m}, \Sigma_{z_{-m}|z_m}) \\ T_j &\sim \text{Continuous-Poisson}(\exp(z_j)o) \end{aligned}$$

Given z_m, the distribution of expression values are simple normal distributions with analytically available means and covariances. However, because z_m is unknown, we must factor into our estimate its distribution, which is both a function of observed data (t_m, o) and prior information (in the form of $\hat{\mu}^{(m)}$ and $\hat{\Sigma}^{(m)}$). Our strategy to estimate the posterior density of programs and non-markers will therefore be to sample from the posterior of z_m, and then given these draws, sample from the conditional Normal distribution of each program and non-marker gene.

4.2.1 Sampling z_m via MCMC

To sample z_m we use Metropolis-Hastings Markov Chain Monte Carlo (MCMC) sampling [7], using the following posterior density function:

$$\begin{aligned} p(z_m|o, T_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) &= \frac{p(T_m|o, z_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_m|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})}{\int_k p(T_m|o, k, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(k|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})\mathrm{d}k} \\ &\propto \prod_{i=1}^{n} p(T_{im}|o, z_{im}, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})p(z_{im}|\hat{\mu}^{(m)}, \hat{\Sigma}^{(m)}) \\ &= \prod_{i=1}^{n}\left[\prod_{j=1}^{|m|} C_{[\exp(z_{ij})o_i]}[\exp(z_{ij})o_i]^{T_{ij}} e^{-[\exp(z_{ij})o_i]}/\Gamma(T_{ij}+1)\right] \end{aligned}$$

Note that we do not require the marginal distribution for Metropolis-Hastings sampling.

As our proposal distribution we use:

$$z_m^{(i+1)} = \mathcal{N}\left(z_m^{(i)}, \gamma \mathbb{I}_{|m|\times|m|}\right).$$

Here $z^{(i)}$ is the i^{th} draw from the sampler, and γ represents the width (variance) of the proposal distribution. To choose this width, we examine, for a schedule of proposal widths (50 logarithmically spaced widths between $10^{3.5}$ and 10^{-1}), which width gives an acceptance rate closest to 0.234 – the ideal rate for a high dimensional parameter [7]. Using this width, we sample 20,100 times from the sampler. We burn-in the first 100 samples and keep every 100^{th} sample thereafter (to offset the effects of the chain's auto-correlation) as our draws from the distribution. Note that we initialize the chain at the MAP estimate of z_m. This ensures the chain is stationary from the beginning.

4.2.2 Sampling program and gene abundances

Given our $M = 200$ draws, $\left[z_m^{(i)}\right]_{i=1}^{M}$, we can sample from the conditional distribution of each program and gene.

Our $i^{(th)}$ draw from the posterior distribution over all programs is obtained from sampling the following Multivariate-Normal,

$$s^{(i)}|z_m^{(i)} \sim \mathcal{N}\left(\mu_{s|z_m^{(i)}}, \Sigma_{s|z_m^{(i)}}\right)$$

where

$$\begin{aligned}\mu_{s|z_m^{(i)}} &= \hat{\mu}_s + \left(z_m^{(i)} - \hat{\mu}^{(m)}\right) \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,s} \\ \Sigma_{s|z_m^{(i)}} &= \hat{\Sigma}_s - \hat{\sigma}_{z_m,s}^T \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,s}\end{aligned}$$

Similarly, our $i^{(th)}$ draw from the posterior distribution over all genes *could be* obtained from sampling the following Multivariate-Normal,

$$z_{-m}^{(i)}|z_m^{(i)} \sim \mathcal{N}\left(\mu_{z_{-m}|z_m^{(i)}}, \Sigma_{z_{-m}|z_m^{(i)}}\right)$$

where

$$\begin{aligned}\mu_{z_{-m}|z_m^{(i)}} &= \hat{\mu}_{z_{-m}} + \left(z_m^{(i)} - \hat{\mu}^{(m)}\right) \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_{-m}} \\ \Sigma_{z_{-m}|z_m^{(i)}} &= \hat{\Sigma}_{z_{-m}} - \hat{\sigma}_{z_m,z_{-m}}^T \operatorname{inv}\left(\hat{\Sigma}^{(m)}\right) \hat{\sigma}_{z_m,z_{-m}}\end{aligned}$$

However, given the size of $\Sigma_{z_{-m}|z_m^{(i)}}$ (approximately 21000×21000), this is not easily doable. Recall, though, that one of our basic assumptions is that the conditional mean abundance of all genes given the abundance of our markers has the covariance structure of all genes sufficiently built in. Thus, we assume

$$\mathcal{N}\left(\mu_{z_{-m}|z_m^{(i)}}, \Sigma_{z_{-m}|z_m^{(i)}}\right) \approx \mathcal{N}\left(\mu_{z_{-m}|z_m^{(i)}}, \operatorname{diag}\left(\Sigma_{z_{-m}|z_m^{(i)}}\right)\right)$$

Here $\operatorname{diag}(\cdot)$ replaces all off-diagonal entries with zeros. Consequently, we only need to compute the diagonal entries of the conditional covariance matrix. Futhermore, given the conditional mean of each gene, we can sample it's abundance in parallel and independently of all others.

From the M samples we have from the conditional posterior distribution of each program and gene, we can estimate properties of the posterior distribution. As point estimates for expression we can use the posterior mean or mode. As confidence estimates for expression we can build credible intervals.

5 References

[1] C H Ho and Hong Kong. The multivariate Poisson-log normal distribution. 2, 1989.

[2] L. Madsen and D. Dalthorp. Simulating correlated count data. *Environmental and Ecological Statistics*, 14(2):129–148, March 2007.

[3] Surojit Biswas, Meredith Mcdonald, Derek S Lundberg, Jeffery L Dangl, and Vladimir Jojic. Learning Microbial Interaction Networks from Metagenomic Count Data. In *Research in Computational Molecular Biology*, volume 1, pages 32–43, 2015.

[4] Hao Wu, Xinwei Deng, and Naren Ramakrishnan. Sparse Estimation of Multivariate Poisson Log-Normal Models from Count Data. *arXiv*, 2016.

[5] Surojit Biswas. The latent logarithm. *arXiv*, pages 1–11, 2016.

[6] Julian Besag. On the Statistical Analysis of Dirty Pictures. *Journal of the Royal Statistical Society*, 48(3):259–302, 1986.

[7] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman & Hall, 3rd edition, 2013.