

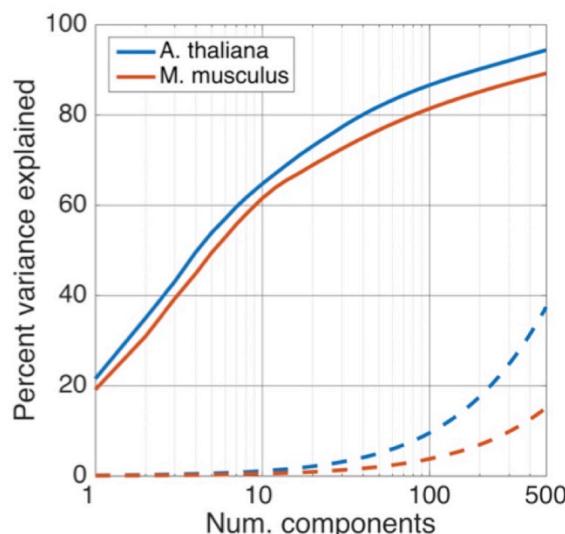
1      Tradict enables accurate prediction of eukaryotic transcriptional states  
2      from 100 marker genes

3                          Supplemental Information

4  
5      Surojit Biswas, Konstantin Kerner, Paulo José Pereira Lima Teixeira, Jeffery L. Dangl, Vladimir  
6      Jovic, Philip A. Wigge

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

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

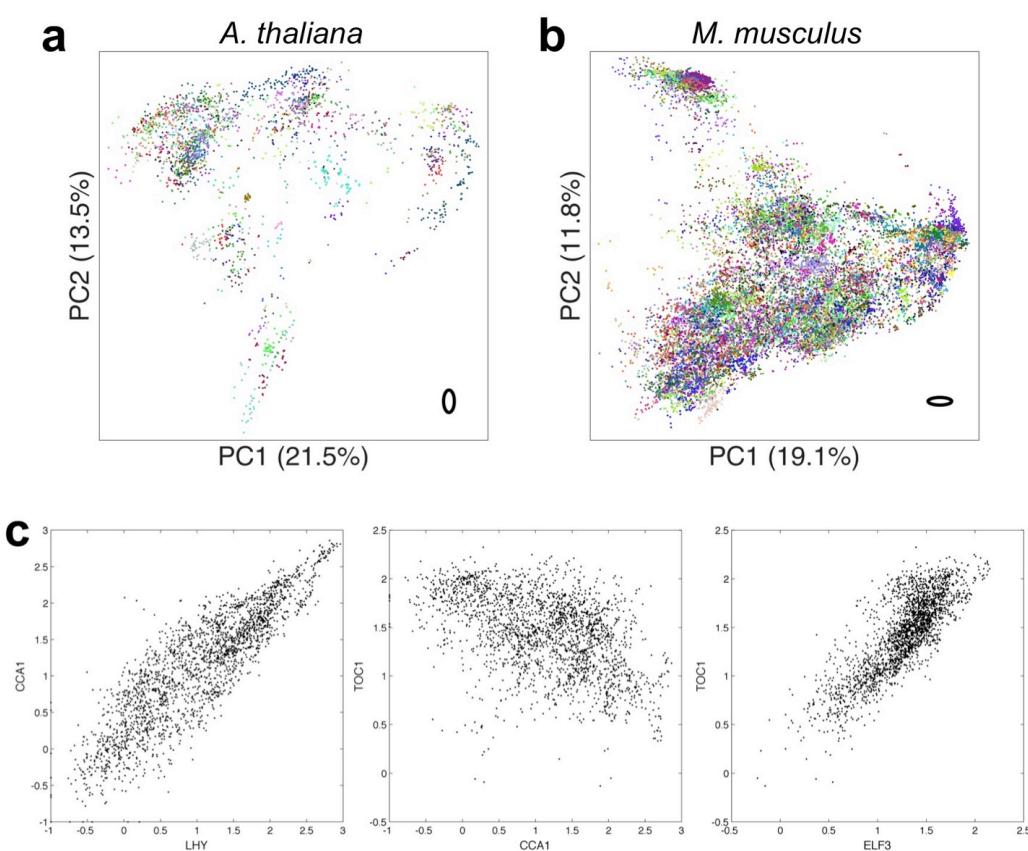


23  
24      **Figure S1. The eukaryotic transcriptome is compressible.** The transcriptome is of low dimensionality, with 100  
25     principal components able to explain 80% or more of expression variation. Dotted lines illustrate cumulative  
26     expression variation explained on a null model realization, where each gene's expression vector was permuted to  
27     break correlative ties to other genes.

28  
29      For *M. musculus*, PC1 described a hematopoietic-nervous system axis. Cardiovascular,  
30     digestive, respiratory, urinary and connective tissues were found intermediate along this axis,  
31     and with the exception of liver tissue, were not differentiable along the first three PCs.  
32     Interestingly, as observed for *A. thaliana*, PC2 represented a developmental axis, with general  
33     "stemness" decreasing with increasing PC2 score. Consistent with this trend, nervous tissue

34 from embryos and postnatal mice had consistently lower PC2 scores than mature nervous  
 35 tissue. We did not find any significant correlation between *Xist* expression and any of the top  
 36 twenty PCs, suggesting that sex was not a major driver of global gene expression relative to  
 37 tissue and developmental context. This is consistent with findings reported in Crowley *et al.*  
 38 (2015)<sup>1</sup>.

39 To understand the compressibility of our training transcriptome collection beyond the first  
 40 three PCs, we examined the percent of expression variation explained by subsequent  
 41 components. Strikingly, we found the first 100 principal components were sufficient to explain  
 42 86.6% and 81.4% of expression variation in the observed transcriptomes for *A. thaliana* and *M.*  
 43 *musculus*, respectively. By contrast, the first 100 principal components of a null model  
 44 realization, in which the expression vectors for each gene were independently permuted, could  
 45 only explain 5-10% of expression variation for both organisms (Figure S1). Given the  
 46 phylogenetic distance spanned by *A. thaliana* and *M. musculus*, this transcriptomic  
 47 compressibility is likely a shared property of all eukaryotes.  
 48



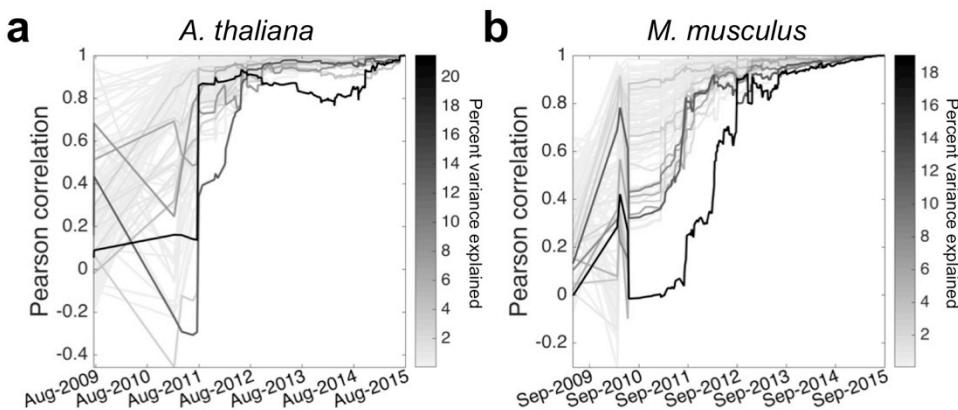
49  
 50 **Figure S2. Our training collection is of high technical quality.** Two dimensional principal components analysis for  
 51 a) *A. thaliana* and b) *M. musculus*, where each sample is colored by the submission it belongs to. Note that while  
 52 multiple submissions may have similar colors, each expression cluster contains many submissions. Bold, black ovals  
 53 in the bottom left of each plot illustrate two standard deviation covariances for the median variance submission.  
 54 c) Expression of late and early elements of the *A. thaliana* circadian clock matches expectations. Scatter plots of *LHY*,  
 55 *CCA1*, and *ELF3* expression across all observed transcriptomes. *LHY* and *CCA1* expression is activated by *TOC1*.  
 56 *CCA1* and *LHY* protein inhibits *TOC1* and *ELF3* transcription.  
 57

58 To further assess the quality and representativeness of our training collection, we  
 59 examined the distribution of SRA submissions across the expression space, compared inter-  
 60 submission variability within and between tissues, inspected expression correlations among

61 genes with well established regulatory relationships, and assessed the evolution of the  
 62 expression space across time. Technical variation due to differences in laboratory procedures  
 63 across labs is difficult assess since this requires two different labs to perform the same,  
 64 equivalently aimed experiment. Nevertheless, for both organisms, each tissue or development  
 65 specific cluster was supported by multiple submissions, and importantly, inter-submission  
 66 variability within a tissue or developmental context was significantly smaller than inter-  
 67 tissue/developmental stage variability ( $p$ -value = 1.23e-16, F-test; Figures S2a-b). We also  
 68 compared the expression of *ELF3*, *LHY*, and *TOC1* -- early and late elements of the *A. thaliana*  
 69 circadian clock -- and found strong correlation in their expression with a direction and magnitude  
 70 that fit established expectations (Figure S2c)<sup>2</sup>.

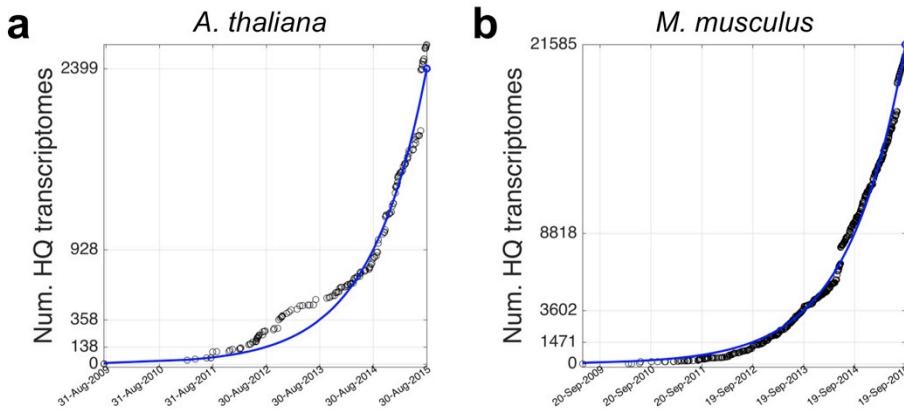
71 We next performed a temporal rarefaction analysis. We compared (measured by  
 72 Pearson correlation) how past distributions of samples along each of the first 100 principal  
 73 components compared to their present distribution. Figures S3a-b illustrate that the expression  
 74 space stabilized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

78



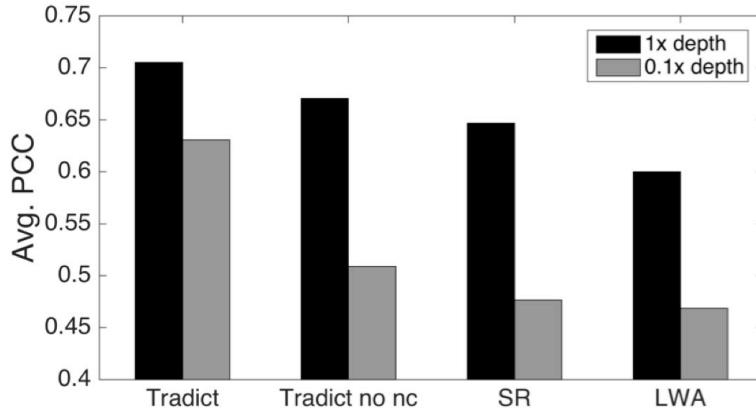
79  
 80  
 81  
 82  
 83

**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*.



84  
 85  
 86  
 87

**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*.



88  
89  
90  
91  
92  
93  
94  
95  
96

**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)<sup>3</sup>. 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 inadequacy or error, the internal covariance will help to learn the correct latent abundances, which in turn, should

128 stabilize predictions in noisy situations. To test this hypothesis, we considered a version of  
129 Tradict, ‘Tradict no nc’ (noise correction), in which only the diagonal of the internal marker  
130 covariance was used, effectively decoupling marker abundances in Tradict’s underlying model.  
131 We re-evaluated intra-submission prediction accuracy for all of the methods, excluding Tradict  
132 Shallow-Seq, on the same training and test set above using 100 markers. However this time, in  
133 order to simulate situations of high measurement error, we rarefied samples in the test set to  
134 0.1x depth and evaluated each method’s predicted (depth-normalized) expression accuracy; the  
135 original 1x depth values formed the basis of comparison. The 10<sup>th</sup>, 25<sup>th</sup>, 50<sup>th</sup>, 75<sup>th</sup>, and 90<sup>th</sup>  
136 percentiles of read depths in the 0.1x scenario were 0.65, 1.1, 2.1, 3.1, and 4.4 million reads,  
137 respectively -- all below the recommended depths for *A. thaliana*. 30-40% of the markers had  
138 zero abundance in nearly half of the samples. Figure 3b illustrates that though all methods  
139 perform worse at 0.1x depth, Tradict is least affected. Importantly, we notice that Tradict no nc’s  
140 performance is substantially reduced at lower depth, confirming our hypothesis that the internal  
141 marker covariance provides a valuable source of noise correction.

142

#### 143 **Supplemental Analysis 3 - Tradict’s limitations as revealed by error, power, and program 144 annotation robustness analyses**

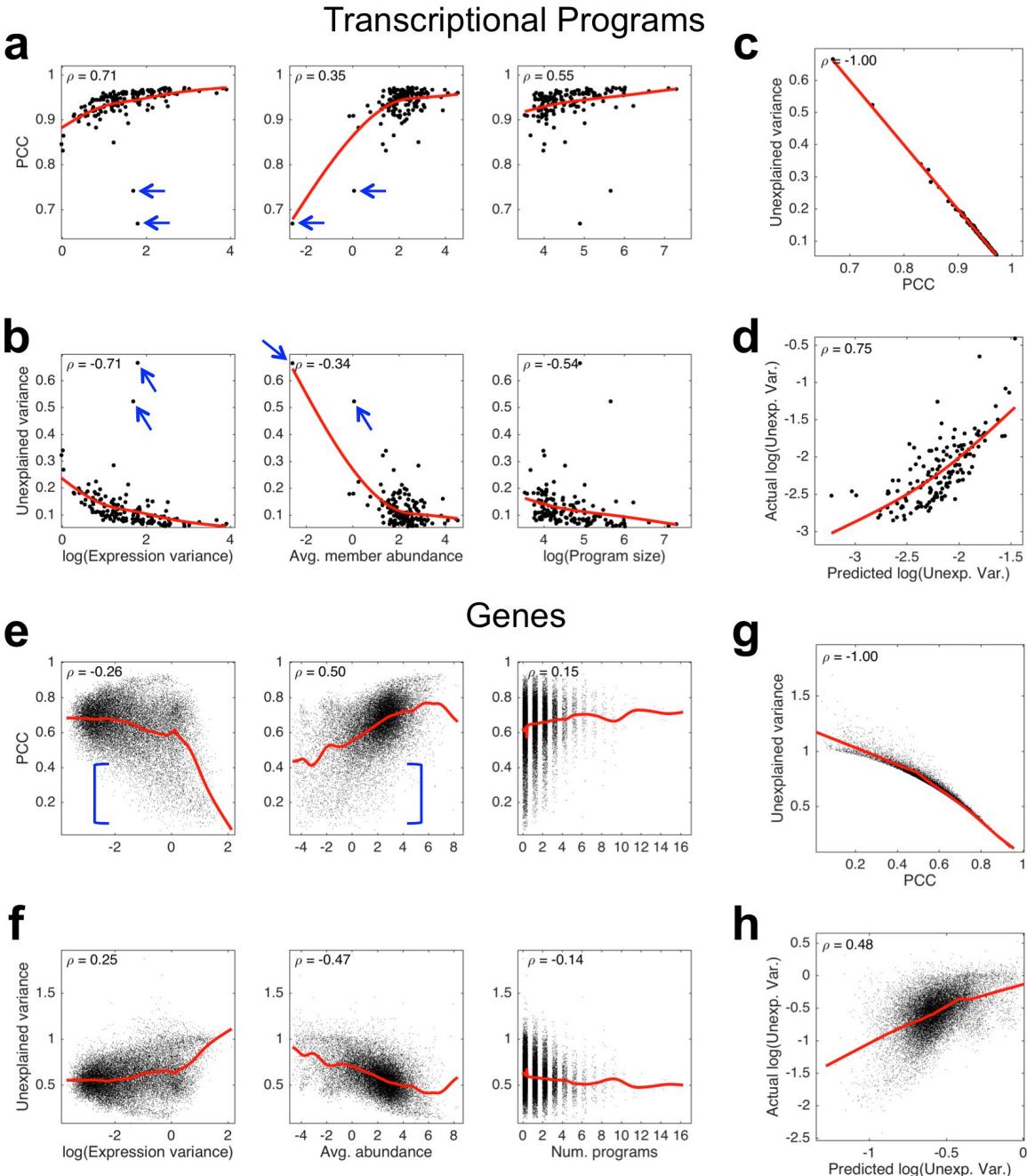
145

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

158

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

159  
160



161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171

**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 ( $\rho$ ) is noted in each plot. Red lines illustrate a cubic spline interpolation.

172 The above expression is equivalent to one minus the coefficient of determination between the  
173 prediction and the target. For each program, we then correlated these measures of performance

174 to the magnitude of training-set expression variation, average training-set abundance of  
175 constituent genes, and the number of genes contained within the program. Similarly, for each  
176 gene, we correlated the above measures of performance to the magnitude of training-set  
177 expression variation, average training-set abundance, and the number of programs in which the  
178 gene participates.

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

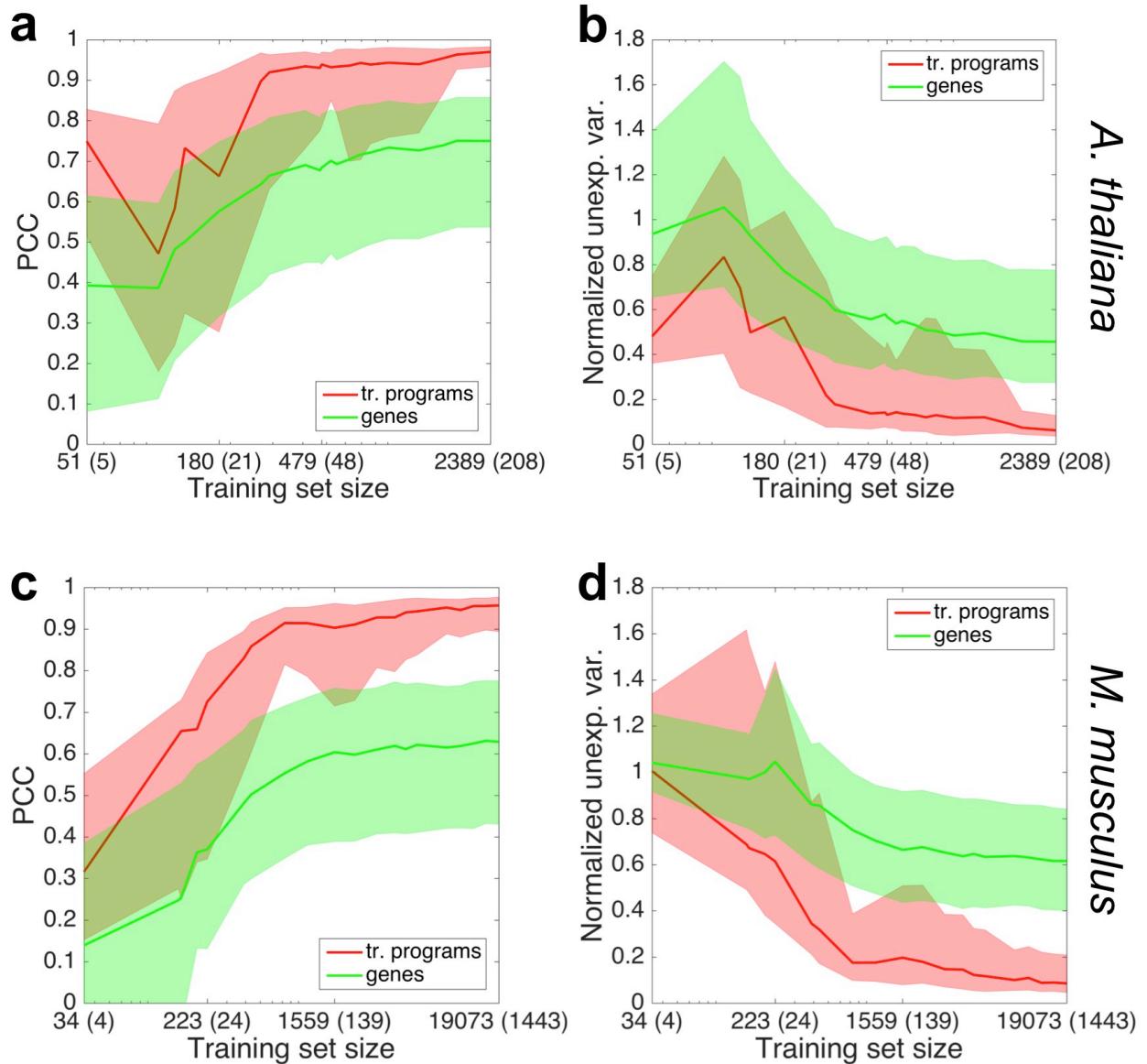
188 We built a linear model to model prediction accuracy -- as measured by log(unexplained  
189 variance) -- of a program as a function of its log(expression variance), average member  
190 abundance (as log-latent abundances), and log(program size). This model could predict  
191 log(unexplained variance)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

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

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

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

221  
222



223

224 **Figure S7. Power analysis reveals Tradict needs approximately 1000 samples to make accurate predictions.**  
225 Test-set prediction accuracies in the form of a) PCC or b) normalized unexplained variance as a function of the size  
226 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  
227 the training set and Z denotes the number of unique submissions to which these training set samples belong. The  
228 solid line depicts the median program (red) or gene (green) and the shaded error bands denote the 20<sup>th</sup> and 80<sup>th</sup>  
229 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  
230 10 logarithmic scale.

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

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

245 Figure S7 shows that for both performance criterion and for both organisms, predictive  
246 performance begins to saturate for nearly all programs and genes after 750-1,000 samples are  
247 included in the training set. We note that not just any collection of 1,000 samples will do. These  
248 samples must be sufficiently varied in context in order for Tradict to perform adequate training  
249 over the variety possible transcriptomic states. By the same token, the first 1,000 samples to the  
250 SRA were likely not chosen to maximize exploration of the transcriptome. Thus, it may be  
251 possible to generate training sets that maximize Tradict's performance with much fewer than  
252 1,000 samples. However, this latter hypothesis requires further investigation.

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

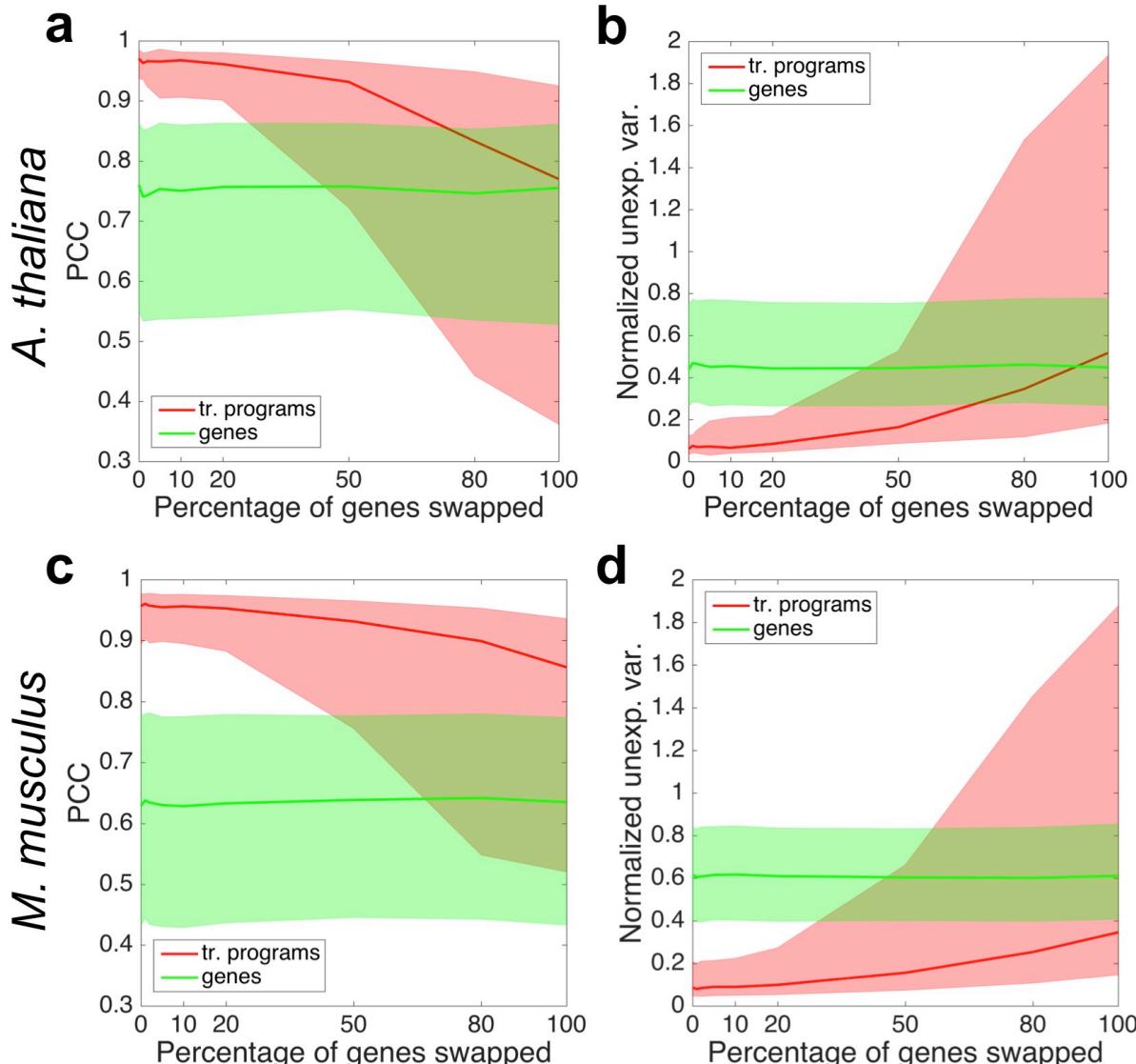
257	6.9K	<i>A. thaliana</i>
258	110.6K	<i>M. musculus</i>
259	8.6K	<i>D. melanogaster</i>
260	5.7K	<i>S. cerevisiae</i>
261	72.1K	<i>H. sapiens</i> (public)
262	2.7K	<i>C. elegans</i>
263	18.1K	<i>D. Rerio</i>

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

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

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

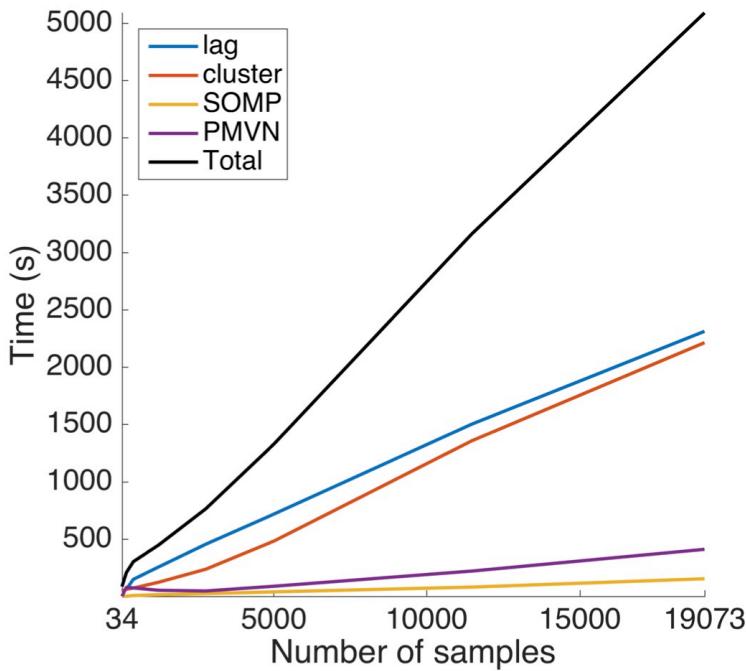
282 Figure S8a-b illustrates how the PCC and normalized unexplained variance performance  
283 metrics behave as a function of the percentage of genes exchanged from each program in the  
284 *A. thaliana* test-set. Both performance criteria for program expression prediction show near  
285 equivalent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



289  
290 **Figure S8. Tradict is robust with respect to the annotations used to define transcriptional programs.** Test-set  
291 prediction accuracies in the form of a) PCC or b) normalized unexplained variance as a function of the percentage  
292 of genes randomly exchanged for each *A. thaliana* transcriptional program. The solid line depicts the median program  
293 (red) or gene (green) and the shaded error bands denote the 20<sup>th</sup> and 80<sup>th</sup> percentile program or gene. c-d) same as  
294 (a) and (b) but for *M. musculus*.

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

303 We note that test-set gene expression prediction performance is invariant with respect to  
304 the level of program mis-annotation. This is expected because, as described in the “Error  
305 Analysis” section, Tradict’s gene expression predictions are statistically decoupled from  
306 program expression prediction.



307  
308  
309  
310  
311  
312  
313  
314

**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.

315

#### Supplemental Analysis 4 - Timing and memory requirements

316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327

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.

328  
329  
330  
331  
332  
333

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.

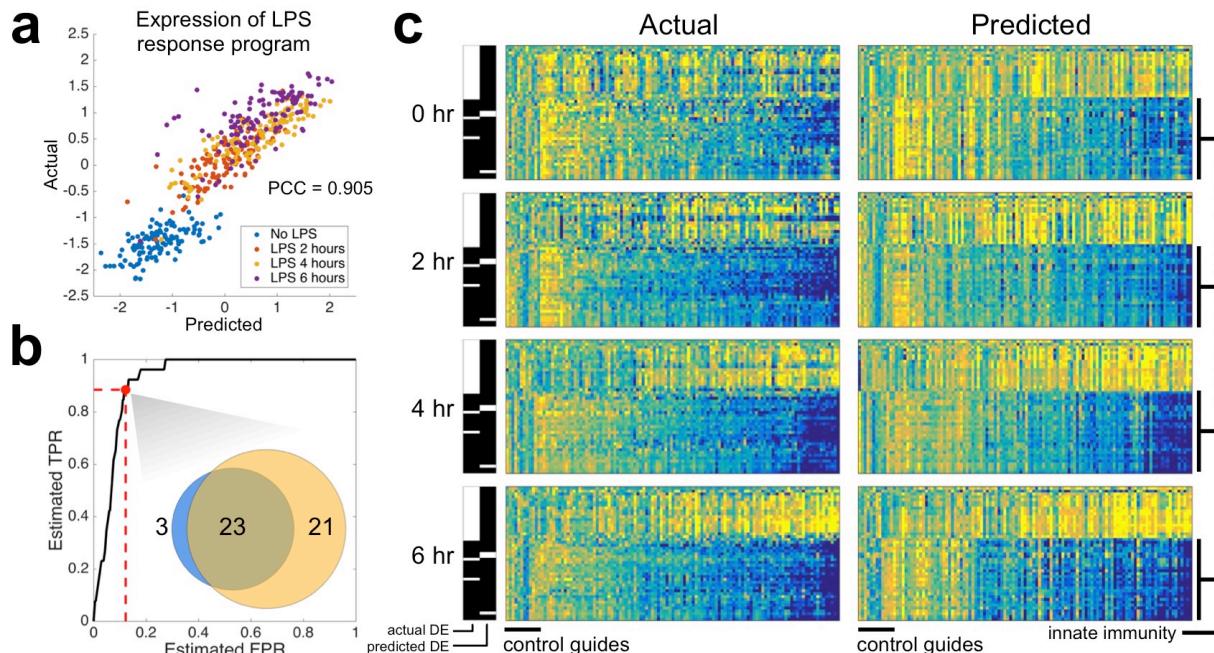
334  
335  
336  
337

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

338 found the equation MEMORY (GB) = 0.0011\*NUM\_SAMPLES + 5.2 described memory usage  
339 well.

340 All computations were performed using one core of a Lenovo P700 ThinkStation with  
341 two Intel Xeon E5-2620 v3 processors and 32 GB of DDR4 ECC RDIMM RAM.  
342

343



344  
345 **Figure S10. Tradict accurately predicts temporal transcriptional responses to lipopolysaccharide treatment in**  
346 **a dendritic cell line CRISPR library.** a) Actual vs. predicted z-score standardized expression of the “response to

347 lipopolysaccharide” transcriptional program. Samples are colored by time point. b) Receiver operator characteristic

348 (ROC) curve illustrating Tradict’s accuracy for identifying differentially expressed (DE) transcriptional programs. Here

349 the “truth set” was considered to be all DE programs with FDR < 0.01 based on actually measured expression values.

350 The marked point along the ROC curve and the inset venn diagram depict the concordance between the predicted

351 and actual set of DE transcriptional programs when an FDR threshold of 0.01 for predicted DE programs was also

352 used. c) Predicted vs actual heatmaps of DE transcriptional programs (rows) across time for different CRISPR lines

353 (columns). Here, DE programs included those found either in actuality or by prediction and are accordingly marked by

354 the black and white indicator bars on the left of each sub-block. Columns of these heat maps represent different

355 profiled lines. The first 12 correspond to negative control guides, whereas the remaining columns correspond to

356 positive regulators of Tnf expression. The expression of programs in each sub-block is z-score normalized to their

357 expression in the negative control guide lines. The bottom 26 programs are all of those directly related to innate

358 immunity among the 368 programs we’ve defined for *M. musculus*. All heatmaps are clustered in the same order

359 across time, genotype, and between predicted and actual.

360

### 361 **Supplemental Analysis 5 - Tradict accurately predicts temporal dynamics of innate** 362 **immune signaling in CRISPRed in primary immune cells**

363 To further dissect Tradict’s capabilities, we examined a *M. musculus* dataset from  
364 Parnas *et al.* (2015) in which one of the first CRISPR screens was performed on primary  
365 immune cells to look for regulators of tumor necrosis factor (Tnf) expression<sup>5</sup>. They found many  
366 positive regulators of Tnf expression and created clonal bone-marrow derived dendritic cell  
367 (BDMC) lines where each positive regulator was disrupted using CRISPR. They used shallow  
368 RNA-sequencing (2.75 +/- 1.2 million reads) to profile the transcriptomes of these lines for 6  
369 hours after lipopolysaccharide (LPS) treatment.  
370

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

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

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

386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421

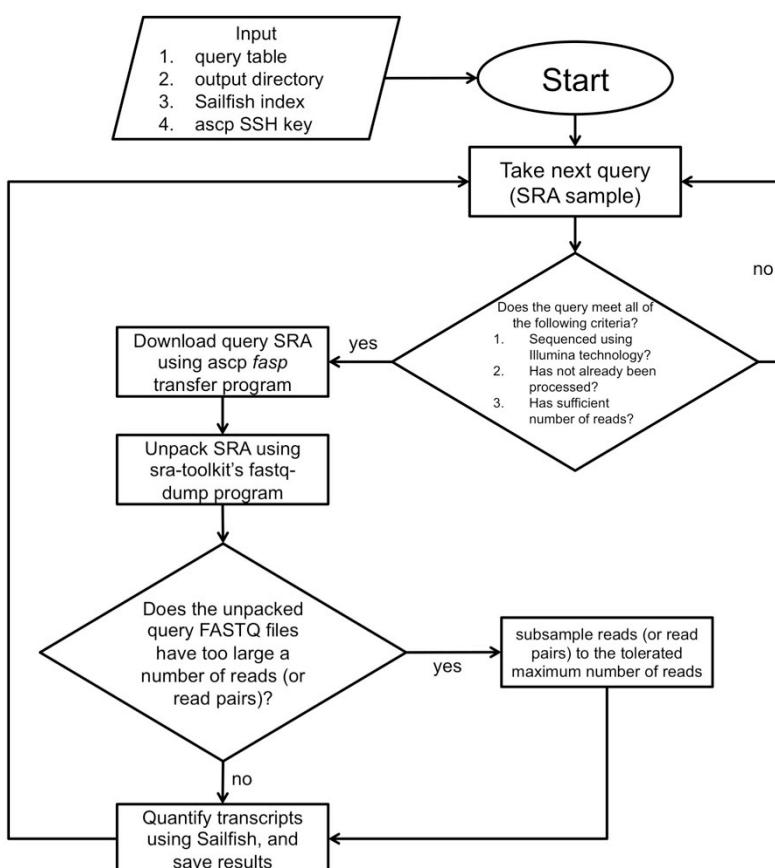
422 **Materials and Methods**

423

424 **Data acquisition and transcript quantification**

425 Data acquisition and transcript quantification were managed using a custom script,  
 426 `srafish.pl`. The `srafish.pl` algorithm and its dependencies are described below.  
 427 Complete instructions for installing (including all dependencies) and using `srafish.pl` are  
 428 available on our GitHub page:

429 [https://github.com/surgebiswas/transcriptome\\_compression/tree/master/data\\_download](https://github.com/surgebiswas/transcriptome_compression/tree/master/data_download).



430

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

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

442       The main inputs into `srafish.pl` are a query table, output directory, Sailfish index, and  
443       ascp SSH key, which comes with each download of the aspera ascp client. `srafish.pl`  
444       depends on Perl (v5.8.9 for Linux x86-64), the aspera ascp client (v3.5.4 for Linux x86-64), SRA  
445       Toolkit (v2.5.0 for CentOS Linux x86-64), and Sailfish (v0.6.3 for Linux x86-64).  
446

#### 447       **Query table construction**

448       For each organism, using the following (Unix) commands, we first prepared a “query table” that  
449       contained all SRA sample ID's as well as various metadata required for the download:  
450

```
451 qt_name=<query_table_file_name>
452 sra_url=http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?save=efetch&db=sra&rettype=ru
453 ninfo&term=
454 organism=<organism_name>
455 wget -O $qt_name '$url($organism[Organism]) AND "strategy rna seq"[Properties]'
456
```

457       Where fields in between <> indicate input arguments. As an example,  
458

```
459 qt_name=Athaliana_query_table.csv
460 sra_url=http://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?save=efetch&db=sra&rettype=ru
461 ninfo&term=
462 organism="Arabidopsis thaliana"
463 wget -O $qt_name '$url($organism[Organism]) AND "strategy rna seq"[Properties]'
464
465
```

#### 466       **Reference transcriptomes and index construction**

467       Sailfish requires a reference transcriptome -- a FASTA file of cDNA sequences -- from  
468       which it builds an index it can query during transcript quantification. For the *A. thaliana*  
469       transcriptome reference we used cDNA sequences of all isoforms from the TAIR10 reference.  
470       For the *M. musculus* transcriptome reference we used all protein-coding and long non-coding  
471       RNA transcript sequences from the Gencode vM5 reference.  
472

473       Sailfish indices were created using the following command:  
474

```
475 sailfish index -t <ref_transcriptome.fasta> -k 20 -p 6 -o .  
476
```

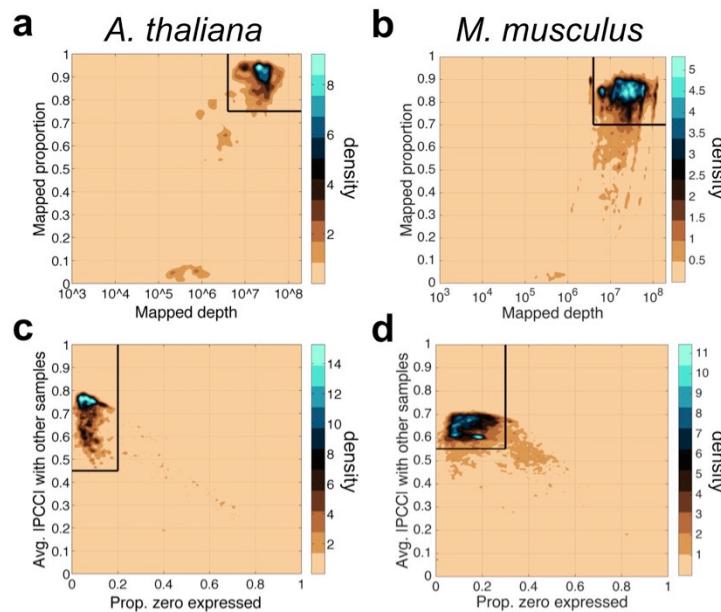
477       Here, <ref\_transcriptome.fasta> refers to the reference transcriptome FASTA file.  
478       Copies of the reference transcriptome FASTA files used in this study are available upon  
479       request.  
480

#### 481       **Quality and expression filtering**

482       Upon completion of download and transcript quantification of all samples, we assembled  
483       an n-samples x p-isoforms matrix of transcripts per million (TPM) values as calculated by  
484       Sailfish. We then proceeded to quality and expression filter the data as follows:  
485

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

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



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

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

539  
540 **Tradict algorithm**  
541 Tradict's usage can be broken down into two parts: 1) Training, and 2) Prediction. Training is  
542 the process of learning, from training data, the marker panel and its predictive relationship to the  
543 expression of transcriptional programs and to the remaining genes in the transcriptome. In  
544 essence, during training we begin with full transcriptome data and collapse its information into a  
545 subset of marker genes. Prediction is the reverse process of predicting the expression of  
546 transcriptional programs and non-marker genes from the expression measurements of just the  
547 selected markers.

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

552  
553 **1) Computing the latent logarithm of the training transcriptome collection** - Expression  
554 values in our training dataset are stored as transcripts per million (TPM), which are non-  
555 negative, variably scaled, and strongly heteroscedastic, similar to read counts. For  
556 subsequent steps in our algorithm and analysis it will be important transform this data to  
557 improve its scaling and heteroscedasticity.

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

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

574 Availability: [https://github.com/surgebiswas/latent\\_log.git](https://github.com/surgebiswas/latent_log.git)

575

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

- 582
- 583   a) In order to capture as much information about the transcriptome as possible, we wanted  
584   to maximize the number of genes covered by the transcriptional programs.
  - 585   b) In order to improve interpretability, we wanted to minimize the total number of  
586   transcriptional programs.
  - 587   c) The number of genes in a transcriptional program should not be too large or too small --  
588   genes in a transcriptional program should be in the same pathway.

589

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

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

606   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  
607   that when the minimum GO term size was as small as it could be (1) and we moved from a  
608   maximum GO term size of 5000 to 10000, we jumped from covering 18432 genes (67% of  
609   the transcriptome) to covering the full transcriptome (black bolded two rows of Supplemental  
610   Table 1). This is due to the addition of one GO term, which was the most general, "Biological  
611   Process," term. Thus, we concluded that 33% of the genes in the transcriptome have only  
612   "Biological Process" as a GO annotation, and therefore that we do not need to capture these  
613   genes in our GO term derived gene sets. Though these genes are not informatively  
614   annotated, we Tradict still model their expression all the same. We hereafter refer to the set  
615   of genes annotated with more than just the "Biological Process" term as *informatively  
616   annotated*.

617

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

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

626 Process" term annotation and were considered not informatively annotated. As we did for *A.*  
627 *thaliana*, we selected a GO term size minimum of 50 and a maximum size of 2000. These  
628 size thresholds defined 368 GO terms, which in total covered 14873 genes (88.9% of the  
629 informatively annotated, 63% of the full transcriptome). As we found for *A. thaliana*, these  
630 368 GO-term derived, globally comprehensive transcriptional programs covered the major  
631 pathways related to growth, development, and response to the environment.

632 Supplemental Tables 3 and 4 contain the lists of the globally comprehensive  
633 transcriptional programs as defined by the criteria above. For each of these programs, we  
634 then computed its first principal component over all constituent genes.  
635

- 636 **3) Marker selection via Simultaneous Orthogonal Matching Pursuit** - After defining  
637 transcriptional programs we are left with a #-training-samples x #-transcriptional-programs  
638 table of expression values. We decompose this matrix using an adapted version of the  
639 Simultaneous Orthogonal Matching Pursuit, using the #-training-samples x #-transcriptional-  
640 programs table as a dictionary<sup>18,19</sup>. Because transcriptional programs are often correlated  
641 with other programs, we first cluster them using consensus clustering<sup>20,21</sup>, which produces a  
642 robust and stable clustering by taking the consensus of many clusterings performed by a  
643 base clustering algorithm. 100 independent iterations of K-means are used as the base-  
644 clusterings, and the number of clusters is determined using the Davies-Bouldin criterion<sup>22</sup>.  
645 The decomposition is greedy, in which during each iteration, the algorithm first finds the  
646 transcriptional program cluster with the largest unexplained variance. It then finds the gene  
647 contained within this cluster of transcriptional programs with the maximum average absolute  
648 correlation to the expression of all transcriptional programs. This gene is then added to an  
649 "active set," onto which the transcriptional program expression matrix is orthogonally  
650 projected. This fit is subtracted to produce a residual, on which the above steps are  
651 repeated until a predefined number of genes have been added to the active set or the  
652 residual variance of the transcriptional program expression matrix falls below some  
653 predefined threshold.  
654

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

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

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

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

673 Note that before we can estimate these parameters, we must learn the log-latent TPMs  
674 of all genes. To do this we first lag-transform the entire training dataset. We then learn the  
675 marker log-latent TPMs, and their associated mean vector and covariance matrix using an  
676 iterative conditional modes algorithm. Specifically, we initialize our estimate of the marker

677 log-latent TPMs to be the lag-transformed expression values, which by virtue of the lag's  
678 probabilistic assumptions are also derived from a Normal Continuous-Poisson hierarchical  
679 model. We then iterate 1) estimation of the mean vector and the covariance matrix given the  
680 current estimate of log-latent TPMs, and 2) maximum *a posteriori* estimation of log-latent  
681 TPMs given the estimated mean vector, covariance matrix, and the measured TPM values  
682 of the selected markers. A small regularization is added during estimation of the covariance  
683 matrix in order to ensure stability and to avoid infinite-data-likelihood singularities that arise  
684 from singular covariance matrices. This is most often happens when a gene's TPM  
685 abundance is mostly zero (i.e. there is little data for the gene), giving the multivariate normal  
686 layer an opportunity to tightly couple this gene's latent abundance to that of another gene,  
687 thereby producing a nearly singular covariance matrix.

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

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

709  
710 To perform prediction, we must translate newly obtained TPM measurements of our marker  
711 genes into expression predictions for transcriptional programs and the remaining non-marker  
712 genes. More specifically, we'd like to formulate these predictions in the form of conditional  
713 posterior distributions, which simultaneously provide an estimate of expression magnitude and  
714 our confidence in that estimate. To do this, we first sample the latent abundances of our  
715 markers from their posterior distribution using the measured TPMs, and the 1 x markers mean  
716 vector and markers x markers covariance matrix previously learned from the training data. This  
717 is done using Metropolis-Hastings Markov Chain Monte Carlo sampling (see "Tradict -  
718 mathematical details" attached to this document for greater details on tuning the proposal  
719 distribution, sample thinning, sampling depth, and burn-in lengths). Using these sampled latent  
720 abundances and the previously estimated mean vectors and cross-covariance matrices, we  
721 then can use standard Gaussian conditioning to sample the log-latent expression of the  
722 transcriptional programs and the remaining genes in the transcriptome from their conditional  
723 distribution. These samples, in aggregate, are samples from the conditional posterior  
724 distribution of each gene and program and can be used to approximate properties of this  
725 distribution (e.g. posterior means, and/or credible intervals).

726

727

728 **References**

- 729 1. Crowley, J. J. *et al.* Analyses of allele-specific gene expression in highly divergent mouse  
730 crosses identifies pervasive allelic imbalance. *Nat. Genet.* **47**, (2015).
- 731 2. Greenham, K. & McClung, C. R. Integrating circadian dynamics with physiological  
732 processes in plants. *Nat Rev Genet* **16**, 598–610 (2015).
- 733 3. Donner, Y., Feng, T., Benoist, C. & Koller, D. Imputing gene expression from selectively  
734 reduced probe sets. *Nat. Methods* **9**, (2012).
- 735 4. Gelman, A. *et al.* *Bayesian Data Analysis*. (Chapman & Hall, 2013).
- 736 5. Parnas, O., Jovanovic, M., Eisenhaure, M. & Zhang, F. A Genome-wide CRISPR Screen  
737 in Primary Immune Cells to Dissect Regulatory Networks. *Cell* **162**, 1–12 (2015).
- 738 6. New England BioLabs Inc. SplintR Ligase. at <<https://www.neb.com/products/m0375-splintr-ligase>>
- 739 7. Lohman, G. J. S., Zhang, Y., Zhelkovsky, A. M., Cantor, E. J. & Jr, T. C. E. Efficient DNA  
740 ligation in DNA – RNA hybrid helices by Chlorella virus DNA ligase. *Nucleic Acids Res.*  
741 1–14 (2013). doi:10.1093/nar/gkt1032
- 742 8. Rohland, N. & Reich, D. Cost-effective, high-throughput DNA sequencing libraries for  
743 multiplexed target capture. *Genome Res.* **22**, 939–946 (2012).
- 744 9. Yang, L. *et al.* The *Pseudomonas syringae* type III effector HopBB1 fine tunes pathogen  
745 virulence by gluing together host transcriptional regulators for degradation. *Submitted*  
746 (2016).
- 747 10. Lundberg, D. S., Yourstone, S., Mieczkowski, P., Jones, C. D. & Dangl, J. L. Practical  
748 innovations for high-throughput amplicon sequencing. *Nat. Methods* **10**, 999–1002  
749 (2013).
- 750 11. Leinonen, R., Sugawara, H. & Shumway, M. The Sequence Read Archive. *39*, 2010–  
751 2012 (2011).
- 752 12. Patro, R., Mount, S. M. & Kingsford, C. Sailfish enables alignment-free isoform  
753 quantification from RNA-seq reads using lightweight algorithms. *Nat. Biotechnol.* **32**, 462–  
754 4 (2014).
- 755 13. Biswas, S. The latent logarithm. *arXiv* 1–11 (2016).
- 756 14. Ma, S. & Kosorok, M. R. Identification of differential gene pathways with principal  
757 component analysis. *Bioinformatics* **25**, 882–889 (2009).
- 758 15. Fan, J. *et al.* Characterizing transcriptional heterogeneity through pathway and gene set  
759 overdispersion analysis. *Nat. Methods* **13**, 241–244 (2016).
- 760 16. Ashburner, M. *et al.* Gene Ontology: tool for the unification of biology. *Nat Genet* **25**, 25–  
761 29 (2000).
- 762 17. The Gene Ontology Consortium. Gene Ontology Consortium: going forward. *Nucleic  
763 Acids Res.* **43**, D1049–D1056 (2015).
- 764 18. Tropp, J. a & Gilbert, A. C. Signal Recovery From Random Measurements Via  
765 Orthogonal Matching Pursuit. *IEEE Trans. Inf. Theory* **53**, 4655–4666 (2007).
- 766 19. Tropp, J. a., Gilbert, A. C. & Strauss, M. J. Algorithms for simultaneous sparse  
767 approximation. Part I: Greedy pursuit. *Signal Processing* **86**, 572–588 (2006).
- 768 20. Monti, S., Tamayo, P., Mesirov, J. & Golub, T. Consensus Clustering : A Resampling-  
769 Based Method for Class Discovery and Visualization of Gene Expression Microarray  
770 Data. *Mach. Learn.* **52**, 91–118 (2003).
- 771 21. Yu, Z., Wong, H.-S. & Wang, H. Graph-based consensus clustering for class discovery  
772 from gene expression data. *Bioinforma.* **23**, 2888–2896 (2007).
- 773 22. Davies, D. L. & Bouldin, D. W. A Cluster Separation Measure. *IEEE Trans. Pattern Anal.  
774 Mach. Intell.* **2**, 224–227 (1979).
- 775 23. Aitchison, J. & Shen, S. M. Logistic-Normal Distributions: Some Properties and Uses.  
776 *Biometrika* **67**, 261 (1980).

- 778 24. Aitchison, J. & Ho, C. H. The multivariate Poisson-log normal distribution. *Biometrika* **76**,  
779 643–653 (1989).
- 780 25. Biswas, S., Mcdonald, M., Lundberg, D. S., Dangl, J. L. & Jovic, V. Learning Microbial  
781 Interaction Networks from Metagenomic Count Data. in *Res. Comput. Mol. Biol.* **1**, 32–43  
782 (2015).
- 783 26. Ho, C. H. & Kong, H. The multivariate Poisson-log normal distribution. **2**, (1989).
- 784 27. Madsen, L. & Dalthorp, D. Simulating correlated count data. *Environ. Ecol. Stat.* **14**, 129–  
785 148 (2007).
- 786

## Tradict - mathematical details

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

### 3 Contents

<b>4</b>	<b>1</b>	<b>Preliminaries</b>	<b>1</b>
<b>5</b>	<b>2</b>	<b>Model</b>	<b>1</b>
<b>6</b>	<b>3</b>	<b>Training</b>	<b>2</b>
7	3.1	Inference of $z_m$ given $\mu^{(m)}$ and $\Sigma^{(m)}$	3
8	3.2	Complete inference of $\mu^{(m)}$ , $\Sigma^{(m)}$ , and $z_m$	4
<b>9</b>	<b>4</b>	<b>Prediction</b>	<b>5</b>
10	4.1	MAP estimation of gene and program abundances	5
11	4.2	Posterior density estimation of gene and program abundances	6
12	4.2.1	Sampling $z_m$ via MCMC	6
13	4.2.2	Sampling program and gene abundances	7
<b>14</b>	<b>5</b>	<b>References</b>	<b>8</b>

15 This document describes the full mathematical details for the concepts presented in the “Tradict algo-  
16 rithm” section, “Building a predictive Multivariate Normal Continuous-Poisson hierarchical model” subsec-  
17 tion of the Materials and Methods in the Supplemental Information. Specifically, we present exactly how  
18 Tradict uses a selected set of markers to 1) complete training, and 2) to perform prediction.

19 1 Preliminaries

20 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  
 21 refer to all indices not specified by  $q$ .

22 2 Model

23 Tradict uses a Continuous-Poisson Multivariate Normal (CP-MVN) hierarchical model to model the expression  
 24 of transcriptional programs and all genes in the transcriptome. Multivariate Normal hierarchies have  
 25 been explored in the past as a means of modeling correlation structure among count based random variables  
 26 [1, 2, 3, 4]. However, given we will be working with abundances as transcripts per million (TPM), which  
 27 are non-negative (can equal zero) and fractional, we relax the integral assumption of the Poisson so it is  
 28 continuous on  $[0, \infty)$ . Specifically, we define the continuous relaxation of the Poisson distribution (hereafter,  
 29 Continuous-Poisson) to have the following density function:

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

30 where  $C_\lambda$  is a normalization constant. The mean of this distribution is given by  $\lambda$ , just as the Poisson.  
 31 We begin by building a predictive model of gene expression, and thereafter discuss a predictive model  
 32 for the expression of transcriptional programs. Let  $z_j$  denote the log-latent abundance of gene  $j$ , such that  
 33  $\exp(z_j)$  is the latent abundance of that gene (in TPM) whose measured abundance is given by  $t_j$ . Let  
 34  $T_j = t_j o$  be the measured total number of transcripts of gene  $j$ . Here  $o$  is the sequencing depth in millions  
 35 of reads of the sample under consideration. We assume then,

$$z \sim \mathcal{N}(\mu, \Sigma)$$

$$T_j \sim \text{Continuous-Poisson}(\exp(z_j)o)$$

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

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

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

$$z_m \sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)})$$

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

$$T_j \sim \text{Continuous-Poisson}(\exp(z_j)o)$$

48 Here,  $\mu^{(m)}$  and  $\Sigma^{(m)}$  refer to mean vector and covariance matrix of  $z_m$ . Given these, the conditional  
 49 mean of the log-latent abundances for all non-marker genes can be obtained through Gaussian conditioning.  
 50 Specifically, for two normally distributed row-vector variables  $a$  and  $b$  the conditional mean of  $b$  given  $a$  is  
 51 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  $\sigma_{ab}$  is the cross-covariance between  
 52  $a$  and  $b$ , and  $\Sigma_a$  and  $\Sigma_b$  are the covariance matrices of  $a$  and  $b$ , respectively.

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

$$z_m \sim \mathcal{N}(\mu^{(m)}, \Sigma^{(m)})$$

$$s|z_m \sim \mathcal{N}(\mu_s|z_m, \Sigma_{s|z_m})$$

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

## 60 3 Training

61 As described in the Supplemental Information, given an estimate of  $z_m$ ,  $\hat{z}_m$ , inference of  $\mu_s$ ,  $\mu_{z_{-m}}$ ,  $\sigma_{z_m,s}$  and  
 62  $\sigma_{z_m,z_{-m}}$  is straightforward. In lag transforming the entire training TPM expression matrix,  $t \in \mathbb{R}^{\text{samples} \times \text{genes}}$ ,  
 63 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  
 64 mean of  $\hat{z}_{-m}$ .

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

68 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  
 69  $\hat{z}_m$  and  $\hat{s}$  and between  $\hat{z}_m$  and  $\hat{z}_{-m}$ , respectively.

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

### 78 3.1 Inference of $z_m$ given $\mu^{(m)}$ and $\Sigma^{(m)}$

79 Suppose Tradict has estimates of  $\mu^{(m)}$  and  $\Sigma^{(m)}$  given by  $\hat{\mu}^{(m)}$  and  $\hat{\Sigma}^{(m)}$ , and let  $T_m = t_m(o \times \mathbf{1}_{1 \times \text{markers}})$   
 80 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  
 81 of sample sequencing depths in millions of reads. Given these, we would like to calculate the maximum *a*  
 82 *posteriori* (MAP) estimate of  $\hat{z}_m = \operatorname{argmax}_{z_m} p(z_m|o, T_m, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})$ .

83 The posterior distribution over  $z_m$  is given by

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

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

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

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

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

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

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

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

92 Differentiating we get,

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

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

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

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

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

## 100 3.2 Complete inference of $\mu^{(m)}$ , $\Sigma^{(m)}$ , and $z_m$

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

- 102 • Initialize  $T_m = t_m(o \times \mathbf{1}_{1 \times \text{markers}})$ ,  $\hat{z}_m = \text{lag}(t_m)$ .
- 103 • 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:
- 104 – Update  $\hat{\mu}^{(m)}$  and  $\hat{\Sigma}^{(m)}$ :

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

- 105 – Update  $\hat{z}_m = \text{MAP\_z}(t, o, \hat{\mu}^{(m)}, \hat{\Sigma}^{(m)})$ .

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

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

## 117 4 Prediction

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

### 126 4.1 MAP estimation of gene and program abundances

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

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

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

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

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

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

134 Similarly, for the entire transcriptome we have,

$$\hat{t}_{ij}^* = \text{argmax}_t p(t|\hat{z}_{im}^*) = \exp(\mu_{z_{ij}|\hat{z}_{im}^*}).$$

135 where,

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

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

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

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

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

139 where,

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

140 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  
 141 between  $z_m$  and  $z_j$ . These were learned from the training data during encoding.

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

## 144 4.2 Posterior density estimation of gene and program abundances

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

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

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

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

150 Given  $z_m$ , the distribution of expression values are simple normal distributions with analytically available  
 151 means and covariances. However, because  $z_m$  is unknown, we must factor into our estimate its distribution,  
 152 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  
 153 strategy to estimate the posterior density of programs and non-markers will therefore be to sample from  
 154 the posterior of  $z_m$ , and then given these draws, sample from the conditional Normal distribution of each  
 155 program and non-marker gene.

### 156 4.2.1 Sampling $z_m$ via MCMC

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

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

159 Note that we do not require the marginal distribution for Metropolis-Hastings sampling.  
 160 As our proposal distribution we use:

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

161 Here  $z^{(i)}$  is the  $i^{th}$  draw from the sampler, and  $\gamma$  represents the width (variance) of the proposal distribution.  
 162 To choose this width, we examine, for a schedule of proposal widths (50 logarithmically spaced widths between  
 163  $10^{3.5}$  and  $10^{-1}$ ), which width gives an acceptance rate closest to 0.234 – the ideal rate for a high dimensional  
 164 parameter [7]. Using this width, we sample 20,100 times from the sampler. We burn-in the first 100 samples  
 165 and keep every 100<sup>th</sup> sample thereafter (to offset the effects of the chain's auto-correlation) as our draws  
 166 from the distribution. Note that we initialize the chain at the MAP estimate of  $z_m$ . This ensures the chain  
 167 is stationary from the beginning.

#### 168 4.2.2 Sampling program and gene abundances

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

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

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

173 where

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

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

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

176 where

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

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

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

180 Here  $\text{diag}(\cdot)$  replaces all off-diagonal entries with zeros. Consequently, we only need to compute the diagonal  
 181 entries of the conditional covariance matrix. Furthermore, given the conditional mean of each gene, we can  
 182 sample its abundance in parallel and independently of all others.

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

## 186 5 References

- 187 [1] C H Ho and Hong Kong. The multivariate Poisson-log normal distribution. 2, 1989.
- 188 [2] L. Madsen and D. Dalthorp. Simulating correlated count data. *Environmental and Ecological Statistics*,  
189 14(2):129–148, March 2007.
- 190 [3] Surojit Biswas, Meredith McDonald, Derek S Lundberg, Jeffery L Dangl, and Vladimir Jovic. Learning  
191 Microbial Interaction Networks from Metagenomic Count Data. In *Research in Computational Molecular  
192 Biology*, volume 1, pages 32–43, 2015.
- 193 [4] Hao Wu, Xinwei Deng, and Naren Ramakrishnan. Sparse Estimation of Multivariate Poisson Log-Normal  
194 Models from Count Data. *arXiv*, 2016.
- 195 [5] Surojit Biswas. The latent logarithm. *arXiv*, pages 1–11, 2016.
- 196 [6] Julian Besag. On the Statistical Analysis of Dirty Pictures. *Journal of the Royal Statistical Society*,  
197 48(3):259–302, 1986.
- 198 [7] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin.  
199 *Bayesian Data Analysis*. Chapman & Hall, 3rd edition, 2013.