Lightweight transcriptomics

Supplemental Information

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

**Supplemental Note 1 - Characterization of**

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).

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 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; SI Figure 1a). 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 (SI Figure 1b).

We next performed a temporal rarefaction analysis. Specifically, we compared (measured by Pearson correlation) how past distributions of samples along each of the first 100 principal components compared to their present distribution. Figure 1d-e 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 (SI Figure).

**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>.

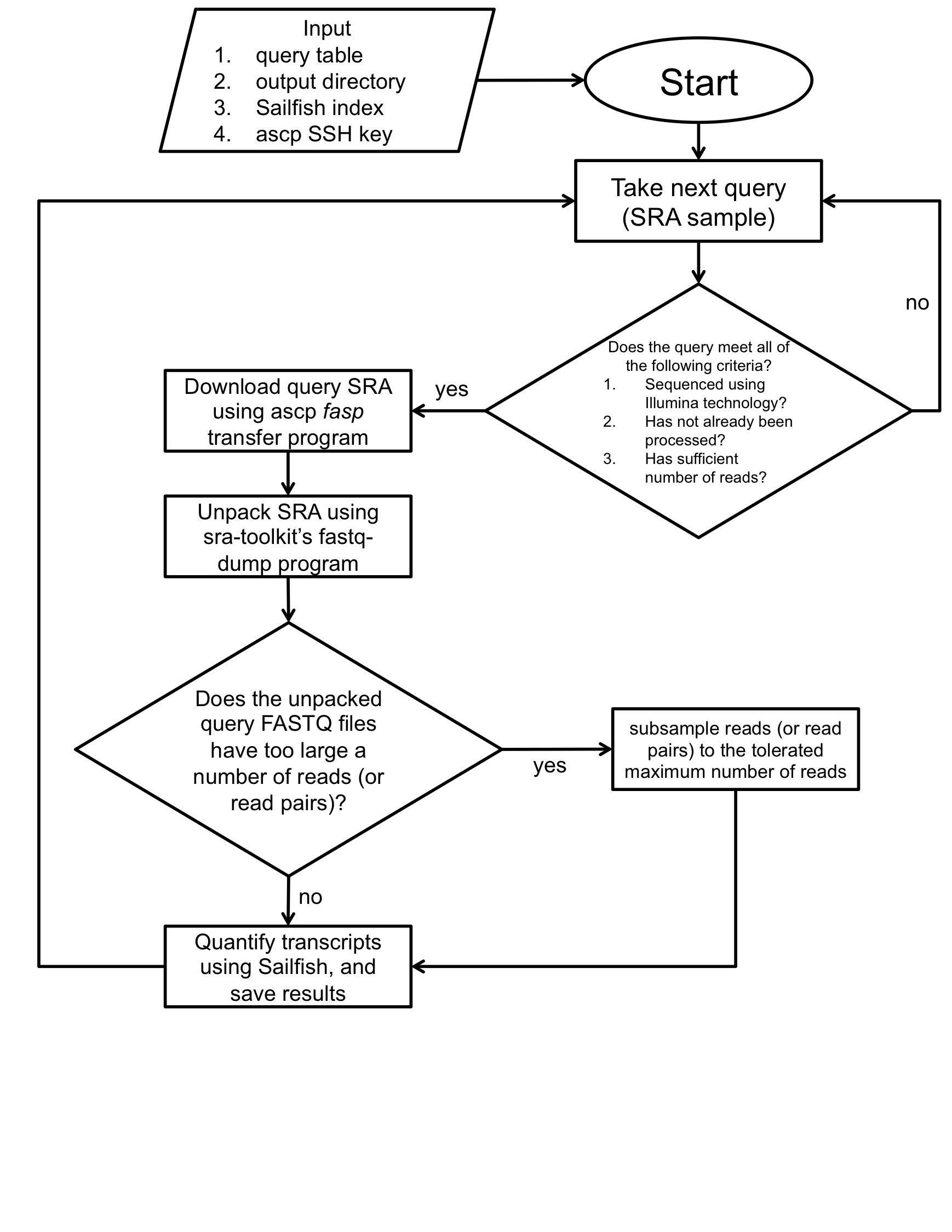
M

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). The raw FASTQ read data is then passed to Sailfish2, 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=runinfo&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=runinfo&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 a 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 (log10[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 SM2b). 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 SM2c). 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).

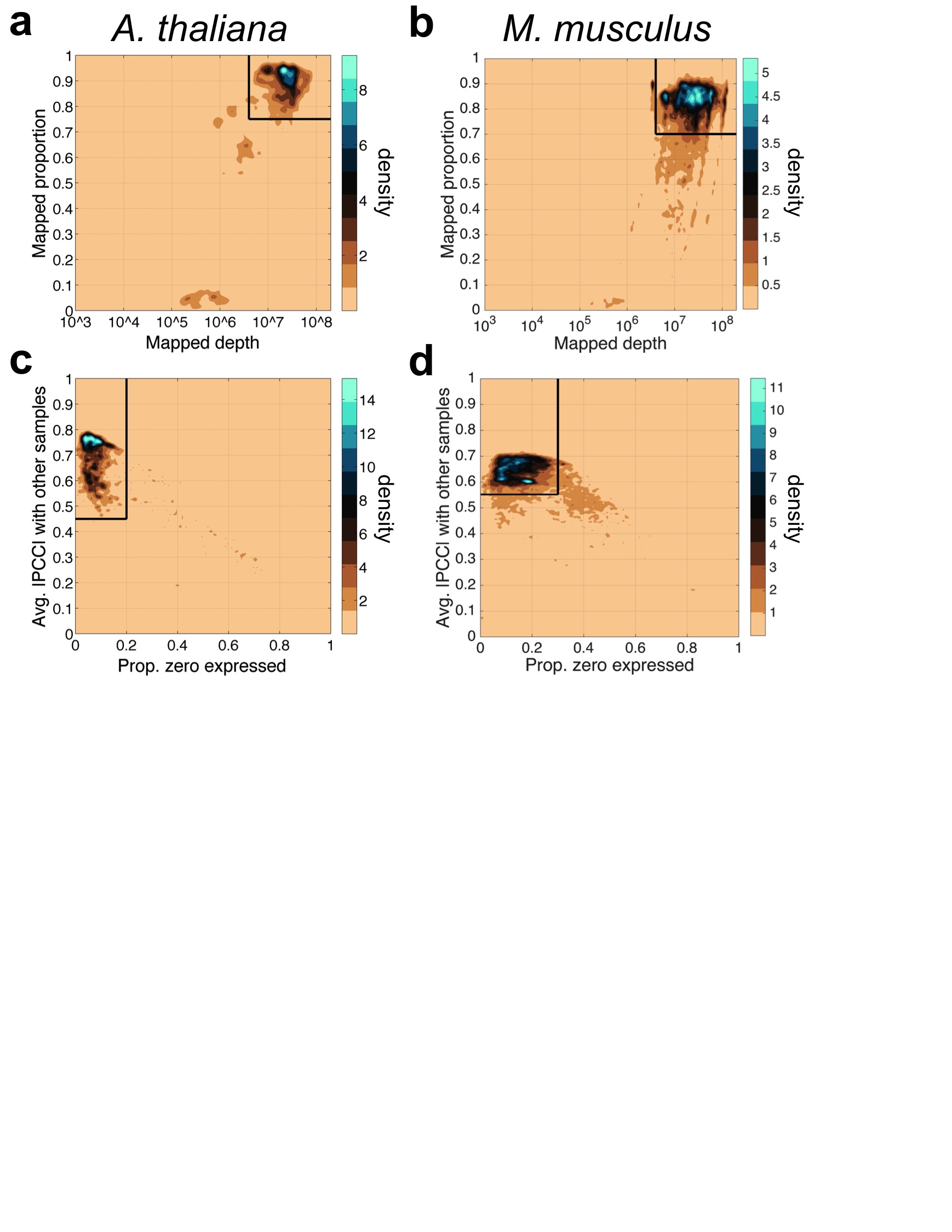


Figure SM2. Quality filtering thresholds for mapping depth and proportion (a,b), and for average correlation to other samples and proportion of zeros (c,d).

**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.

**Temporal rarefaction analysis**

[pending]

**Tradict algorithm**

Tradict’s usage can be broken down into two parts: 1) Encoding, and 2) Decoding. Encoding 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 encoding we begin with full transcriptome data and collapse its information into a subset of marker genes. Decoding 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 encoding 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”3. 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, 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

1. **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 [REF]. This virtual “eigengene” [REF] 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*:
   1. 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.
   2. In order to improve interpretability, we wanted to minimize the total number of transcriptional programs.
   3. 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 [REF]. 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. 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.

1. **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 [describe uses in signal processing?], using the #-training-samples x #-transcriptional-programs table as a dictionary4,5. Because transcriptional programs are often correlated with other programs, we first cluster them using consensus clustering [REF], 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 [REF]. The decomposition is greedy, in which for each iteration the algorithm first find 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.
2. **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. Readersinterested 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 [REF, Aitchison, Biswas, and others]. Here it is assumed the TPM expression of each gene in a given sample is a noisy, 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 decoding, 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 that 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 decoding, we must translate newly obtained TPM measurements of our marker genes into expression predictions for transcriptional programs and the remaining non-marker genes. To do this, we first learn the latent abundances of our markers 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 maximum *a posteriori* estimation. Using these latent abundances and the previously estimated mean vectors and cross-covariance matrices, we can use standard Gaussian conditioning to estimate the log-latent expression of the transcriptional programs and the remaining genes in the transcriptome.

**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. 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).

3. Biswas, S. The latent logarithm. *arXiv* 1–11 (2016).

4. Tropp, J. a & Gilbert, A. C. Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit. **53,** 4655–4666 (2007).

5. Tropp, J. a., Gilbert, A. C. & Strauss, M. J. Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. *Signal Processing* **86,** 572–588 (2006).