Deconvolution Method and Test Cases

Jason A. Hackney

February 13, 2017

1 Introduction

Given a gene expression dataset derived from mixed samples (*e.g.*, whole blood or tissue biopsies), we frequently find that the major source of variability in the dataset is related to changes in cellularity. These may or may not be correlated with the phenotype of interest for the dataset. In either case, we'd like to be able to quantify these changes in cellularity, and incorporate them into our analyses. In the case of uncorrelated changes in cellularity, we'd like to take advantage of the additional information of the cellularity to improve our ability to detect differentially expressed genes. In the case of correlated changes in cellularity, we'd like to be able to estimate gene expression variability that is associated with cellularity. In many cases, we're particularly interested in changes in gene expression within a cell type of interest, as a result of some disease process. Our main use case here is to identify changes in immune cell gene expression in disease tissues.

There are several ways to go about doing deconvolution of gene expression data. In essence, deconvolution is an attempt to identify underlying variables that are present in the data set of interest. In many cases, these underlying variables will not be directly correlated with our condition of interest, and are merely increasing the noise in the data set. In these cases, identifying these underlying variables will allow us to better identify genes that are related to the condition of interest. In other cases, we will find variables that are somewhat correlated with, but not responsible for the condition of interest. A good example of this is the presence of infiltrating immune cells in a solid tumor. In this case, the infiltrating immune cells are different between normal tissue and tumor, but are unrelated to the phenotype of interest. Again, directly modeling the infiltrating immune cells can help to identify differential gene expression between tumor and normal, censoring the genes that are differential due to differing quantities of immune cells.

One way this type of analysis has been done is using surrogate variable analysis [1], [2]. Surrogate variable analysis is an attempt to identify unmodeled variables in a linear model context. There are several methods that have been proposed for identifying how many surrogate variables and estimating the variables themselves. Two R packages, `sva` and `isva`, provide implementations for surrogate variable analysis.

We propose an alternate method that makes use of prior biological knowledge in identifying and constructing our surrogate variables, called eigengenes. In this case, our prior knowledge is in the form of tissue- and cell type-specific gene sets. These sets were identified as being specifically expressed in a given tissue compared to other normal tissues in the body. Altogether there are 23 gene sets for normal tissues. We also have 7 gene sets that are specific to immune cell function. These are taken from the IRIS project [3], or manually curated based on literature, and represent genes specifically expressed in several compartments of the immune system.

2 Methodology

2.1 Input expression data

The method employed by GSDecon makes use of log_2-transformed data. In the case of microarray data from Affymetrix, the rma method, and its variants produce data appropriate for use with the decon method. Data from Agilent two color arrays should be log_2 ratios between the two channels. For RNA-seq data, raw count data can be provided as a number of formats: `DGEList`, `CountDataSeq` or `DESeqDataSet`. These data will be preprocessed using the voom algorithm to generate log_2-scale data appropriate for use with the downstream method.

2.2 Identifying informative gene sets

Since most transcriptomic measurements only give relative expression values to a total RNA pool, we are actually only interested in the expression from a given cell type if it varies across a set of samples. If a cell type is present in differing amounts across a sample set, then its marker genes should vary across the sample set with a high degree of intercorrelation. For each cell type, i, we start with a set of marker genes m_i, that are relatively specific to that cell type. We can determine how much support we have for a cell type varying across a set of samples by calculating the first eigenvalue, λ_{m_i}, of the expression of our marker genes, and compare this value to a null distribution of eigenvalues.

In the case of random normally distributed data, we could calculate a p-value for finding an eigenvalue of at least λ_{m_i}, as is done in independent surrogate variable analysis [2]. However, we have found that expression data has quite a bit more correlation structure than a set of random normal variables. So we instead calculate an empirical p-value by randomly selecting a set of the same number of genes, q, from the total set of measured genes (G, $q \in G$), and calculating the first eigenvalue, λ_q, of that set of genes. Our p-value is the number of times a randomly selected gene set has a higher eigenvalue, λ_q, than the first eigenvalue of our marker gene matrix (λ_{m_i}, above).

2.3 Summarizing gene sets to get relative proportionality

In the previous section, we argued that since the expression of marker genes, m_i, is dependent on the fraction of cell type i within the mixture, we could use the relative expression of our marker genes as a surrogate for relative proportionality. However, not all marker genes perform equivalently across datasets and across technology platforms. To account for this technical variability, we need a robust estimate of the mean expression of our marker genes. We start with a matrix of gene log_2-transformed expression values M, where each row is a gene, and columns, p, represent samples. We first standardize each gene, g, to have mean 0 and standard deviation 1, retaining the offset (c_g), and the scaling factor (s_g). The resultant matrix, X_{m_i}, is the standardized expression values of marker genes m_i in our dataset. We then take the singular value decomposition:

$$X_{m_i} = U\Sigma V^T$$

Where columns of U and V^T are the left and right singular vectors, respectively, and Σ is a diagonal matrix that has the singular values. We then derive a new matrix of singular values, Σ', that retains the first singular value, but which has all other singular values set to 0. We calculate a set of normalized values, X'_{m_i}, for the gene set:

$$X'_{m_i} = U\Sigma' V^T$$

The rows, x_g, of X_{m_i} are rescaled using the previously saved gene specific offsets and scaling values:

$$y_g = s_g x_g + c_g$$

This gives a normalized expression matrix, Y_{m_i}, that contains values that are on the same log_2 scale as the original data. We calculate an eigengene, e_{m_i}, by taking the column-wise mean of the normalized values. For each sample, s, and gene g in a gene set m_i of size j:

$$e_{sm_i} = \sum_{g=1}^{j} \frac{Y'_{sg}}{j}$$

Each eigengene is an estimate of the relative proportion of a particular cell or tissue type in the mixture.

3 Test case: mixed rat cRNA

Two data sets that are publicly available and have sufficient information that let us determine whether our analyses have been successful. One comes from a paper describing the CS-SAM algorithm for determining differential gene expression in the background of known cellular proportions [4]. The other dataset has data from mixtures of known proportions of cell lines, described along with a deconvolution algorithm based on least squares fitting [5].

We can gauge the performance of our different deconvolution schemes using Spearman's correlation coefficient between the values estimated by the deconvolution algorithm and the known values from our test data sets.

3.1 Loading the tissue specific gene sets

First, we load the gene set collection for our tissue specific gene sets. This is done using the `DeconGeneSetCollection` method.

```
> deconGSC <- DeconGeneSetCollection()
```

This generates a `GeneSetCollection` instance that has each of our gene sets, using Entrez Gene numbers as the accession numbers. We will use the `mapIdentifiers` method to remap these gene sets to the different identifiers of interest.

3.2 Estimating proportions of mixed rat tissue

We'll start with a data set from Shen-Orr, *et al.*, where known quantities of RNA from rat brain, liver and lung were mixed and hybridized to microarrays. First we'll retrieve the `ExpressionSet` for the rat experiment from the GEO database, and filter the dataset using the `featureFilter` function from the `genefilter` package. This reduces our dataset to one probe per gene.

```
> library(GEOquery)
> library(genefilter)
> GSE19830 <- getGEO("GSE19830")[[1]]
> annotation(GSE19830) <- "rat2302"
> GSE19830 <- featureFilter(GSE19830)
> tissueFractions <- read.csv(system.file("extdata", "tissueFractions.csv",
+     package = "GSDecon"), row.names = 1)
> GSE19830 <- GSE19830[, rownames(tissueFractions)]
>
```

Since the IRIS gene sets were derived from human tissue, we'll have to map our human Entrez Genes to rat genes. We'll do this using the orthology maps provided by the EnsEMBL biomart. This mapping uses EnsEMBL genes to identify orthologs, so we'll first have to map from human Entrez genes to human EnsEMBL genes, and then to rat EnsEMBL genes, next to rat Entrez Genes, and finally to probes on the microarray platform used in this experiment. Since our analysis uses a dataset from brain, liver and lung, we'll restrict our `DeconGeneSetCollection` to sets derived from these tissues.

```
> library(org.Hs.eg.db)
> library(biomaRt) # For mapping human to rat
> library(hgu133plus2.db)
> library(org.Rn.eg.db)
> library(rat2302.db)
> decon.ensg.gsc <- mapIdentifiers(deconGSC, ENSEMBLIdentifier(),
+     org.Hs.egENSEMBL)
> decon.ensg.gsc <- decon.ensg.gsc[names(decon.ensg.gsc) %in%
+         c('Brain', 'Liver', 'Lung')]
> all.ensgs <- unique(unlist(geneIds(decon.ensg.gsc)))
> mart <- useMart("ENSEMBL_MART_ENSEMBL")
> mart <- useDataset("hsapiens_gene_ensembl", mart=mart)
> ratOrthologs <- getBM(mart = mart,
+     attributes=c("ensembl_gene_id", "rnorvegicus_homolog_ensembl_gene"),
+     filters = "ensembl_gene_id", values = all.ensgs)
> ratOrthologs <- ratOrthologs[ratOrthologs[,2] != "",]
> ratOrthologList <- split(ratOrthologs[,2], ratOrthologs[,1])
> decon.rnoensg.gsc <- mapIdentifiers(decon.ensg.gsc, ENSEMBLIdentifier(),
+     as.environment(ratOrthologList))
> decon.rnoeg.gsc <- mapIdentifiers(decon.rnoensg.gsc, EntrezIdentifier(),
+     org.Rn.egENSEMBL2EG)
> decon.rno.gsc <- mapIdentifiers(decon.rnoeg.gsc, AnnotationIdentifier(),
+     revmap(rat2302ENTREZID))
>
```

Now that we have a GeneSetCollection for our gene expression platform, we can test out our deconvolution approach. For this analysis, we need the expression matrix, a linear model, and a `GeneSetCollection` to derive our eigengenes. The linear model in this case is a null model that we are using as our baseline information about the expression in the dataset. In many cases, the moel of interest is an intercept-only model, as we are using in this case.

```
> GSE19830_model <- model.matrix(~1, GSE19830)
> GSE19830_decon <- decon(GSE19830, GSE19830_model,
+     decon.rno.gsc, pvalueCutoff = 0.01, seed = 358353, nPerm = 249)
> GSE19830_eigengenes <- eigengenes(GSE19830_decon)
> GSE19830_decon
```

```
DeconResults
  pvalueCutoff: 0.01
  significant components: Brain, Liver, Lung (3 total)
```

The return value from `decon` is an object of type `DeconResults` that contains the surrogate variables for the different tissues, and the empirical p-values for those gene sets. In this fairly contrived example, all three gene sets have low p-values, indicating that they might provide more information about the expression data than our null model.

```
> pvalues(GSE19830_decon)
```

```
Brain Liver  Lung
0.008 0.004 0.004
```

	Brain	Liver	Lung
Brain	0.984	-0.509	-0.410
Liver	-0.477	0.983	-0.237
Lung	-0.535	-0.229	0.990

Table 1: Eigengene Correlation Coefficient

We can then check if the eigengenes determined by decon correlate with the known proportion of tissue RNA that went into the experiment. The results are presented in Table 1.

Visualizing the surrogate variables compared to the known values for the different tissues shows a strong correlation between the estimated fraction of each cell type and the logarithm of the actual abundance (Figure 1).

4 Deconvolution of cell line mixing experiment

A second contrived data set we can work with was published along with an alternative deconvolution algorithm [5]. This data is from an experiment in which four cell lines from hematopoietic cancers were mixed in known proportions. The cell lines represent three major cell types: B cells (Raji and IM-9 cells), T cells (Jurkat), and monocytes (THP-1). The analysis of this data set is similar to the previous one. We start by loading the `ExpressionSet` and making our factors of interest. The data from GEO in this case is normalized using the MAS5 algorithm, so we'll have to log-transform the data before running the `decon` algorithm.

```
> GSE11058 <- getGEO("GSE11058")[[1]]
> exprs(GSE11058) <- log2(exprs(GSE11058))
> annotation(GSE11058) <- "hgu133plus2"
> GSE11058 <- featureFilter(GSE11058)
> cellFractions <- read.csv(system.file("extdata", "cellFractions.csv",
+     package = "GSDecon"), row.names = 1, check.names = FALSE)
> GSE11058 <- GSE11058[, rownames(cellFractions)]
```

We start as by remapping the IRIS `GeneSetCollection` to the identifiers of choice; in this case the platform is the Affymetrix HG-U133plus2 microarray.

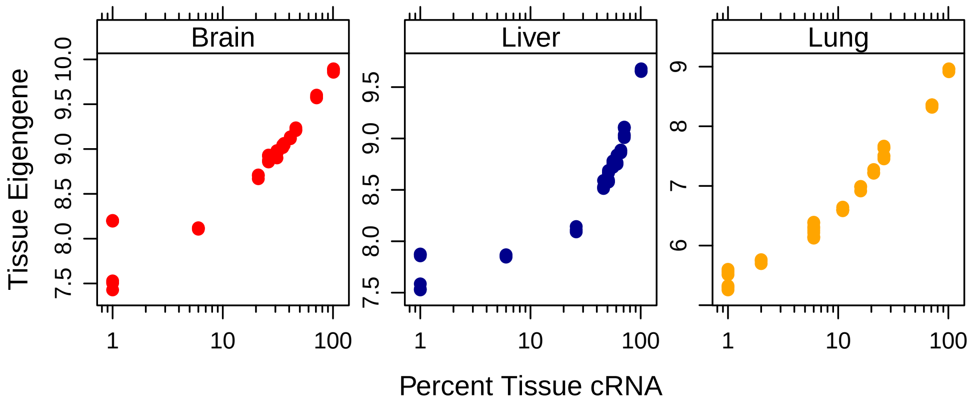

Figure 1: Known v. estimated quantities of tissue RNA.

```
> decon.blood.gsc <- deconGSC[grep('IRIS', description(deconGSC))]
> decon.blood.u133.gsc <- mapIdentifiers(decon.blood.gsc, AnnotationIdentifier(),
+     revmap(hgu133plus2ENTREZID))
```

In the deconvolution analysis, we'll again use an intercept-only linear model, since we have no predictors of interest.

```
> GSE11058_model <- model.matrix(~1, GSE11058)
> GSE11058_decon <- decon(GSE11058, GSE11058_model, decon.blood.u133.gsc,
+     pvalueCutoff=0.05, seed = 358353)
> GSE11058_eigengenes <- eigengenes(GSE11058_decon)
> pvalues(GSE11058_decon)
```

```
       B Monocyte        T 
   0.004    0.016    0.008 
```

	Jurkat	IM-9	Raji	THP-1	B cell
B	-0.471	0.318	0.730	-0.385	0.931
Monocyte	-0.226	0.081	0.000	0.888	-0.343
T	0.926	0.057	-0.218	-0.086	-0.411

Table 2: Eigengene Correlation Coefficient

From this we can see that gene sets for B cells, T cells, and monocytes all have low p-values. Checking the correlation between our eigengenes and the proportion of cells in the experiment using Spearman's rank correlation (Table 2), we find that our eigengenes are again performing as expected, with THP-1 showing a high correlation with the monocyte eigengene and Jurkat cells having a high correlation with the T cell eigengene. The combined Raji and IM-9 mix is also highly correlated with the B cell eigengene, but less so. Visualization of the eigengenes (Figure 2) might help us understand why the reason for this.

We again see a log-linear relationship between the actual cell line proportion and the eigengenes. Additionally, something is going wrong with estimation of the B cell content. This is apparent in the two groups of samples with high numbers of B cells, but with differing estimation from the B cell eigengene. This is likely due to differences in the Raji and IM-9 cells that make one more like the idealized version of a B cell from the IRIS gene set. This can be seen directly if we plot the individual cell types against the B cell eigengene (Figure 3).

References

[1] J. T. Leek, W. E. Johnson, H. S. Parker, A. E. Jaffe, and J. D. Storey. The sva package for removing batch effects and other unwanted variation in high-throughput experiments. *Bioinformatics*, 28(6):882–883, mar 2012.

[2] A. E. Teschendorff, J. Zhuang, and M. Widschwendter. Independent surrogate variable analysis to deconvolve confounding factors in large-scale microarray profiling studies. *Bioinformatics*, 27(11):1496–1505, jun 2011.

[3] A R Abbas, D Baldwin, Y Ma, W Ouyang, A Gurney, F Martin, S Fong, M van Lookeren Campagne, P Godowski, P M Williams, A C Chan, and H F Clark. Immune response in silico (IRIS): immune-specific genes identified from a compendium of microarray expression data. *Genes Immun*, 6(4):319–331, 2005.

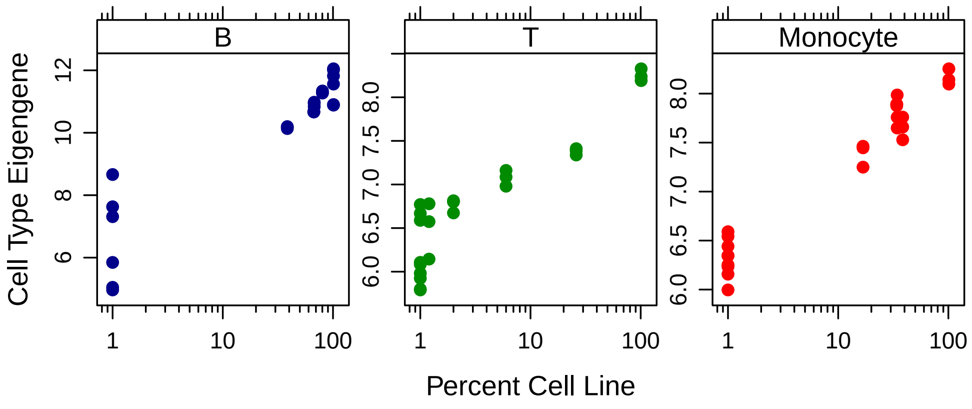

Figure 2: Known v. estimated quantities of tissue RNA

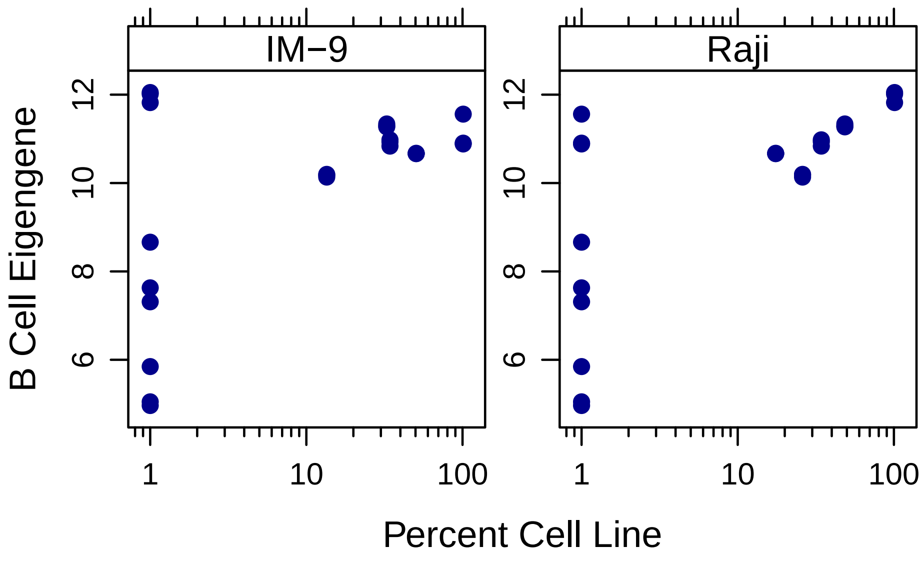

Figure 3: B Cell Eigengene by Cell Line

[4] S S Shen-Orr, R Tibshirani, P Khatri, D L Bodian, F Staedtler, N M Perry, T Hastie, M M Sarwal, M M Davis, and A J Butte. Cell type-specific gene expression differences in complex tissues. *Nat Methods*, 7(4):287–289, 2010.

[5] A R Abbas, K Wolslegel, D Seshasayee, Z Modrusan, and H F Clark. Deconvolution of blood microarray data identifies cellular activation patterns in systemic lupus erythematosus. *PLoS One*, 4(7):e6098, 2009.