Chapter 4 - Test Pathway Significance

Gabriel Odom

2018-04-18

Table of Contents

[Overview 1](#_Toc511835362)

[Pathway Testing Setup 3](#_Toc511835363)

[Pathway Significance Testing Overview 3](#_Toc511835364)

[Clean Up the Pathway List 3](#_Toc511835365)

[Extract Pathway PCs 4](#_Toc511835366)

[Test Pathway Association 4](#_Toc511835367)

[Adjust the Pathway -Values for FDR 4](#_Toc511835368)

[Output a Sorted Data Frame / Tibble 4](#_Toc511835369)

[AES-PCA 5](#_Toc511835370)

[Method Details 5](#_Toc511835371)

[AES-PCA Examples 6](#_Toc511835372)

[Supervised PCA 8](#_Toc511835373)

[Method Details 8](#_Toc511835374)

[Supervised PCA Examples 9](#_Toc511835375)

[Analyze the Results 11](#_Toc511835376)

[Tidy the Pathway Results 11](#_Toc511835377)

[Plot Significant Survival Pathways 12](#_Toc511835378)

[Rank Significant Genes 14](#_Toc511835379)

# Overview

This vignette is the fourth chapter in the “Pathway Significance Testing with pathwayPCA” workflow. This vignette builds on the material covered in the [“Import and Tidy Data”](https://gabrielodom.github.io/pathwayPCA/articles/C2-Importing_Data.html) and [“Creating -Omics Data Objects”](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html) vignettes. This guide will outline the major steps needed analyze Omics\*-class objects with pathway-level adaptive, elastic-net, sparse PCA (AES-PCA) or Supervised PCA. We will consider three examples: survival, regression, and binary responses. The main goal of pathway significance testing is to discover potential relationships between pathways (bundles of genes) and a response. This package (in its current form) should not be used for prediction, or regressing on multiple variables, but should simply be used to screen or identify potential pathways for futher inspection.

### Outline

Before we move on, we will outline our steps. After reading this vignette, you should

1. Understand the basics of the AES-PCA pathway-significance testing approach.
2. Understand the basics of the Supervised PCA pathway-significance testing approach.
3. Be able to apply AES-PCA or Supervised PCA to analyze survival (OmicsSurv), regression (OmicsReg), or classification (OmicsCateg) data containers.

### Load Packages

Before we begin, make sure you have the latest version of the package. In order to install a package from GitHub, you will need the devtools:: package (<https://github.com/r-lib/devtools>) and either [Rtools](https://cran.r-project.org/bin/windows/Rtools/) (for Windows) or [Xcode](https://developer.apple.com/xcode/) (for Mac). Then you can install the development version of the [pathwayPCA package](https://github.com/gabrielodom/pathwayPCA) from [GitHub](https://github.com/):

devtools::install\_github("gabrielodom/pathwayPCA")

library(pathwayPCA)  
library(parallel)

### Load Omics\* Data

Because you have already read through the [Import and Tidy Data](https://gabrielodom.github.io/pathwayPCA/articles/Importing_Data.html) and [Creating -Omics Data Objects](https://gabrielodom.github.io/pathwayPCA/articles/Create_Omics_Objects.html) vignettes, we will pick up with the colon\_OmicsSurv object we created in the last vignette. For our pathway analysis to be meaningful, we need gene expression data (from a microarray or something similar), corresponding subject information (such as weight, type of cancer, or survival time and censoring indicator), and a gene set list. The colon\_OmicsSurv data container we constructed in [Chapter 3](https://gabrielodom.github.io/pathwayPCA/articles/Create_Omics_Objects.html) has all of this.

colon\_OmicsSurv  
#> Formal class 'OmicsSurv' [package "pathwayPCA"] with 4 slots  
#> ..@ eventTime : num [1:250] 64.9 59.8 62.4 54.5 46.3 ...  
#> ..@ eventObserved: logi [1:250] FALSE FALSE FALSE FALSE TRUE FALSE ...  
#> ..@ assayData\_df :Classes 'tbl\_df', 'tbl' and 'data.frame': 250 obs. of 656 variables:  
#> ..@ pathwaySet :List of 3  
#> .. ..- attr(\*, "class")= chr [1:2] "pathwaySet" "list"

# Pathway Testing Setup

In this section, we will describe the workflow of the Supervised PCA (superPCA\_pVals) and AES-PCA (AESPCA\_pVals) pathway significance-testing methods. **The implementation of Supervised PCA in this package does not currently support analysis of responses with missingness.** If you plan to test yourpathways using the SUpervised PCA method, please remove observations with missing entries beforeanalysis. Unlike the current implementation of Supervised PCA, our current implementation of AES-PCA can handle some missingness in the response.

Also, when we compare computing times, we use a Dell Precision Tower 5810 with 64-bit Windows 7 Enterprise OS. This machine has 64 GB of RAM and an Intel Xeon E5-2640 v4 2.40 GHz processor with 20 threads. We use two threads for parallel computing. Please adjust your expectations of computing time accordingly.

## Pathway Significance Testing Overview

Now that we have our data stored in an Omics\*-class object, we can test the significance of each pathway with AES- or Supervised PCA. These functions both

1. “Trim” the pathway gene set list.
2. Extract the first principal components (PCs) from each expressed pathway in the MS design matrix.
3. Test their association with the response matrix or vector.
4. Adjust the pathway -values for False Discovery Rate (FDR).
5. Return a sorted data frame of the adjusted -values for each pathway.

The major differences between the two methods involve the execution of (1) and (2), which we will describe in their methods sections.

## Clean Up the Pathway List

The analysis functions within the pathwayPCA package subset the feature data frame by the genes in each pathway. Therefore, if we have genes in the pathways that aren’t recorded in the data frame, then we will necessarily create missing (NA) predictors. To circumvent this issue, we check if each gene in each pathway is recorded in the data frame, and remove the genes for which we have not recorded expression levels. However, if we remove genes from the pathways which do not have recorded levels in the predictor data frame, we could theoretically remove all the genes from a given pathway. Thus, we also check to make sure that each pathway still has some minimum number of genes present (defaulting to three or more) after we have removed the genes without corresponding expression levels.

The expressedOmes() function performs these two actions simultaneously. This function is called and executed automatically within the setup step. This function takes in a valid Omics\*-class object, removes the unexpressed genes from each pathway, subsequently trims the pathways that have fewer than the minimum number of genes allowed, and returns an object of the same Omics\* class as the input. If there are any pathways that are removed due to this execution, the pathways list within the pathwaySet object within each Omics\* object will have a character vector of the pathways removed stored as the "missingPaths" attribute. Access this attribute with the attr() function.

## Extract Pathway PCs

The details of this step will depend on the method, but the overall idea remains the same. First, select the columns of the assay data frame that correspond to each of the clean pathways. Then, use the chosen PCA method to extract the first PCs from those pathways.

## Test Pathway Association

The details of this step will also depend on the method. Now that we have a list of PCs representing the data from each pathway, we apply simple models to test if the PCs are significantly related to the output. For survival output, we use Cox Proportional Hazards (Cox PH) regression. For categorical output, because we only support binary responses in this version, we use logistic regression to test for a relationship between pathway PCs and the response. For continuous output, we use a simple multiple regression model. The AES- and Supervised PCA methods differ on *how* to calculate -values from these models, but the end result of this step is a -value for each of the trimmed pathways.

## Adjust the Pathway -Values for FDR

At this step, we have a vector of -values corresponding to the list of trimmed pathways. We know that repeated comparisons inflate the Type-I error rate, so we adjust these -values. We use the FDR adjustments executed in the mt.rawp2adjp function from the Bioconductor package [multtest](https://www.bioconductor.org/packages/3.7/bioc/manuals/multtest/man/multtest.pdf). We copied the code from this function into our package, with some slight modifications in implementation. However, we do not depend on this package directly. We acknowledge their work in this area and express our gratitude. Common adjustment methods to control the family-wise error rate or FDR are the Bonferroni, Sidak, Holm, or Benjamini and Hochberg techniques.

## Output a Sorted Data Frame / Tibble

The end result of either PCA variant is a data frame. The data frame has the following columns:

* pathways: The names of the pathways in the Omics\* object (given in object@@pathwaySet$pathways).
* setsize: The number of genes in each of the original pathways (given in the object@pathwaySet$setsize object).
* terms: The pathway description, as given in the object@pathwaySet$TERMS object.
* rawp: The unadjusted -values of each pathway.
* ...: Additional columns for each requested FDR adjustment.

The data frame will have its rows sorted in increasing order by the adjusted -value corresponding to the first adjustment method requested. Ties are broken by the raw -values. Additionally, if you use the tidyverse package suite (and have these packages loaded), then the output will be a tibble object, rather than a data frame object. This object class comes with enhanced printing methods and some other benefits.

# AES-PCA

Now that we have described the overview of the pathway analysis methods, we can discuss and give examples in more detail.

## Method Details

### AES-PCA Method Sources

Adaptive, elastic-net, sparse PCA is a combination of the [Adaptive Elastic-Net](https://doi.org/10.1214/08-AOS625) of Zou and Zhang (2009) and [Sparse PCA](https://doi.org/10.1198/106186006X113430) of Zou et al. (2006). This method was applied to pathways association testing by [Chen (2011)](https://doi.org/10.2202/1544-6115.1697). Accoding to Chen (2011), the “AES-PCA method removes noisy expression signals and also account[s] for correlation structure between the genes. It is computationally efficient and the estimation of the PCs does not depend on clinical outcomes.” This package uses a legacy version of the [LARS algorithm](https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf) of Efron et al. (2003) to calculate the PCs.

### Calculate Pathway-Specific Model -Values

For the AES-PCA method, pathway -values are calculated with a permutation test. Therefore, when testing the relationship between the response and the PCs extracted by AES-PCA, the accuracy of the permuted -values will depend on how many permutations you call for. The default number in the AESPCA\_pVals() function is 1000. Be warned, however, that this may be too few permutations to create accurate seperation in pathway significance. You could increase the permutations to a larger value, should your computing resources allow for that. For even moderately-sized data sets (~2000 features) and 1000 pathways, this could take half an hour or more. About 20-25% of the computing costs will be extracting the AES-PCs from each pathway, and the remaining 75-80% of the cost will be the permutation test (for 1000 reps).

### AES-PCA Pros and Cons

Pros:

* The AES-PCA method can handle some missingness in the response.
* The -values are calculated non-parametrically.

Cons:

* The AES-PCA algorithm requires optimization over two tuning parameters and can therefore be considerably slower than the singular value decomposition or eigendecomposition.
* The -values calculated may be too discrete for fewer than 10,000 permutations, which can affect the behavior of the adjustment procedures.

## AES-PCA Examples

Now that we have discussed both the overview of the two methods at large, and some of the details specific to AES-PCA, we can run some examples. Included in this package, we have a small tidy assay and corresponding gene subset list. This data set has 656 gene expression measurements on 250 colon cancer patients. Survival responses pertaining to these patients are also included. Further, we also have a small list of 15 pathways which correspond to our example colon cancer assay.

### Survival Response

We will use two of our available cores with the parallel computing approach. We will adjust the -values with the Hochberg (1988) and Sidak Step-Down FDR-adjustment procedures. For the tiny assay with 15 associated pathways, this calculation is completed in 28 seconds. If we increase the number of permutations from 1000 to 10,000, this calculation takes 222 seconds ( longer). Even though we increased the permutations tenfold, the function completed execution less than 10 times longer (as we mentioned above, roughly a quarter of the computing time is extracting the PCs from each pathway).

surv\_aes\_pVals\_df <- AESPCA\_pVals(object = colon\_OmicsSurv,  
 numReps = 1000,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("Hoch", "SidakSD"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Part 1: Calculate Pathway AES-PCs  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway PCs in Parallel  
#> DONE  
#>   
#> Part 2: Calculate Permuted Pathway p-Values  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway p-Values in Parallel  
#> DONE  
#>   
#> Part 3: Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

### Regression Response

We can also make a mock regression data set by treating the event time as the necessary continuous response. For this example, we will adjust the -values with the Holm (1979) and Benjamini and Hochberg (1995) FDR-adjustment procedures. This calculation took 17 seconds. For 10,000 permutations, this calculation takes 102 seconds ( longer).

reg\_aes\_pVals\_df <- AESPCA\_pVals(object = colon\_OmicsReg,  
 numReps = 1000,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("Holm", "BH"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Part 1: Calculate Pathway AES-PCs  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway PCs in Parallel  
#> DONE  
#>   
#> Part 2: Calculate Permuted Pathway p-Values  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway p-Values in Parallel  
#> DONE  
#>   
#> Part 3: Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

### Binary Classification Response

Finally, we can simulate a mock classification data set by treating the event indicator as the necessary binary response. For this example, we will adjust the -values with the Sidak Single-Step and Benjamini and Yekutieli (2001) FDR-adjustment procedures. This calculation took 30 seconds. For 10,000 permutations, this calculation takes 226 seconds ( longer).

categ\_aes\_pVals\_df <- AESPCA\_pVals(object = colon\_OmicsCateg,  
 numReps = 1000,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("SidakSS", "BY"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Part 1: Calculate Pathway AES-PCs  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway PCs in Parallel  
#> DONE  
#>   
#> Part 2: Calculate Permuted Pathway p-Values  
#> Initializing Cluster  
#> DONE  
#> Extracting Pathway p-Values in Parallel  
#> DONE  
#>   
#> Part 3: Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

# Supervised PCA

We now discuss and give examples of the Supervised PCA method.

## Method Details

### Supervised PCA Method Sources

While PCA is a commonly-applied *unsupervised* learning technique (i.e., response information is not necessary), one limitation of this method is that ignoring response information may yield to a first PC completely unrelated to outcome. In an effort to bolster this weakness, [Bair et al. (2006)](https://doi.org/10.1198/016214505000000628) employed response information to rank predictors by the strength of their association. Then, they used PCA to extract PCs from feature design matrix subsets constructed from the predictors most strongly associated with the response. [Chen et al. (2008)](https://doi.org/10.1093/bioinformatics/btn458) extend this technique to subsets of biological features within pre-defined biological pathways; they independently applied the Supervised PCA routine to each pathway in a pathway set. [Chen et al. (2010)](https://doi.org/10.1002/gepi.20532) built on this work, testing if pathways were significantly associated with a given biological or clinical response.

### Calculate Pathway-Specific Model -Values

As thoroughly discussed in Chen et al. (2008), the model fit and regression coefficient test statistics no longer come from their expected distributions. Necessarily, this is due to Supervised PCA’s strength in finding features already associated with outcome. Therefore, for the Supervised PCA method, pathway -values are calculated from a mixture of extreme value distributions. We use numerical optimization routines to calculate the maximum likelihood estimates of the mean, precision, and mixing proportion components of a mixture of two Gumbel extreme value distributions (for minima and maxima of a random normal sample). The -values from the pathways after permuting the response is used to estimate this null distribution, so the results may be less accurate for a very small set of pathways.

### Supervised-PCA Pros and Cons

Pros:

* The Supervised PCs are extracted without numerical optimization, so calculating the PCs for each pathway is considerably faster than calculating AES-PCs.
* The -values are calculated parametrically, so calculating the -values is considerably faster than AES-PCA.

Cons:

* In rare cases, numerical routines used to find the maximum likelihood estimates for the mixture distribution needed to calculate the -values in Supervised PCA can fail to converge.
* The Supervised PCA method cannot have missing values in the response.

## Supervised PCA Examples

### Survival Response

We will use two of our available cores with the parallel computing approach. We will adjust the -values with the Hochberg (1988) and Sidak Step-Down FDR-adjustment procedures. For the tiny assay with 15 associated pathways, this calculation is completed in 6 seconds. If we compare this to AES-PCA at 1000 permutations, Supervised PCA is faster; for 10,000 permutations, it’s faster.

surv\_spr\_pVals\_df <- superPCA\_pVals(object = colon\_OmicsSurv,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("Hoch", "SidakSD"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Initializing Cluster  
#> DONE  
#> Calculating Pathway Test Statistics in Parallel  
#> DONE  
#> Calculating Pathway Critical Values in Parallel  
#> DONE  
#> Calculating Pathway p-Values  
#> Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

### Regression Response

We can also make a mock regression data set by treating the event time as the necessary continuous response. For this example, we will adjust the -values with the Holm (1979) and Benjamini and Hochberg (1995) FDR-adjustment procedures. This calculation took 5 seconds. If we compare this to AES-PCA at 1000 permutations, Supervised PCA is faster; for 10,000 permutations, it’s faster.

reg\_spr\_pVals\_df <- superPCA\_pVals(object = colon\_OmicsReg,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("Holm", "BH"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Initializing Cluster  
#> DONE  
#> Calculating Pathway Test Statistics in Parallel  
#> DONE  
#> Calculating Pathway Critical Values in Parallel  
#> DONE  
#> Calculating Pathway p-Values  
#> Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

### Binary Classification Response

Finally, we can simulate a mock classification data set by treating the event indicator as the necessary binary response. For this example, we will adjust the -values with the Sidak Single-Step and Benjamini and Yekutieli (2001) FDR-adjustment procedures. This calculation took 8 seconds. If we compare this to AES-PCA at 1000 permutations, Supervised PCA is faster; for 10,000 permutations, it’s faster.

categ\_spr\_pVals\_df <- superPCA\_pVals(object = colon\_OmicsCateg,  
 parallel = TRUE,  
 numCores = 2,  
 adjustpValues = TRUE,  
 adjustment = c("SidakSS", "BY"))  
#> Of the 676 unique genes in the input pathway set, 9.0% were not expressed in  
#> the input data and were therefore removed.  
#> After trimming unexpressed genes from the 15 supplied pathways, we removed 0  
#> pathway(s) because they contained 3 or fewer genes.  
#> Of the 656 measured genes in the input data frame, 93.8% were included in at  
#> least one pathway after trimming.  
#> Initializing Cluster  
#> DONE  
#> Calculating Pathway Test Statistics in Parallel  
#> DONE  
#> Calculating Pathway Critical Values in Parallel  
#> DONE  
#> Calculating Pathway p-Values  
#> Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

# Analyze the Results

## Tidy the Pathway Results

Now that we have the pathway-specific -values, we can inspect the top pathways by significance. We will tidy the data frame returned by the superPCA\_pVals() and AESPCA\_pVals() functions, then plot the most significant -values using the ggplot2:: package. For graphics, we will need two new packages:

library(tidyverse)  
#> -- Attaching packages ------------------------------------------------------------------------------------------------------------------- tidyverse 1.2.1 --  
#> v ggplot2 2.2.1 v purrr 0.2.4  
#> v tibble 1.4.2 v dplyr 0.7.4  
#> v tidyr 0.8.0 v stringr 1.3.0  
#> v readr 1.1.1 v forcats 0.3.0  
#> -- Conflicts ---------------------------------------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
#> x dplyr::filter() masks stats::filter()  
#> x dplyr::lag() masks stats::lag()  
library(reshape2)  
#>   
#> Attaching package: 'reshape2'  
#> The following object is masked from 'package:tidyr':  
#>   
#> smiths

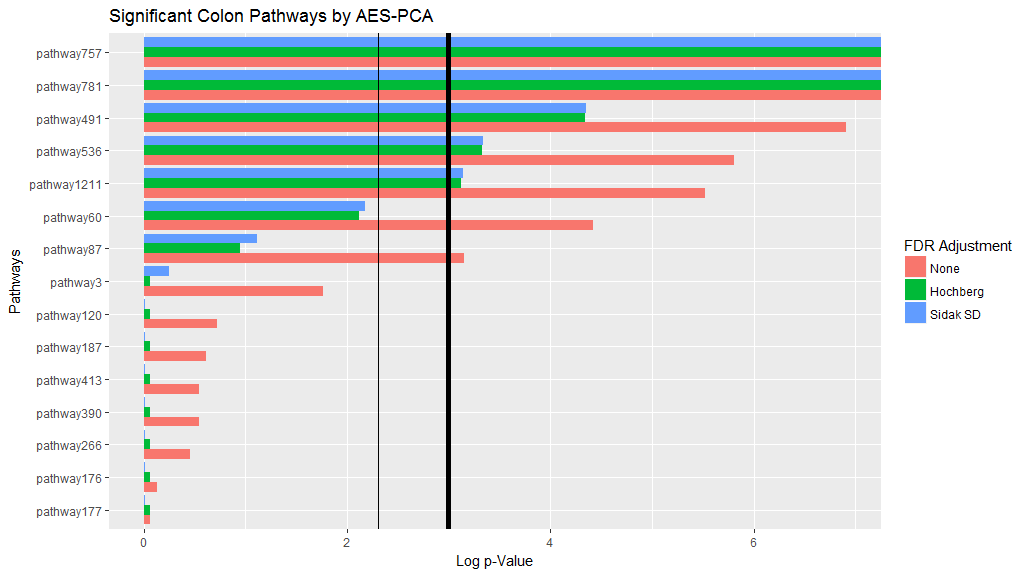
For the two survival data pathway -value data frames, we will transform the data for better graphics.

surv\_aes\_melt\_df <- surv\_aes\_pVals\_df %>%  
 select(-terms, - setsize, -trim\_size) %>%  
 melt(id.vars = "pathways") %>%  
 mutate(score = -log(value)) %>%  
 mutate(pathways = factor(pathways,  
 levels = rev(unique(pathways)),  
 ordered = TRUE))  
  
surv\_spr\_melt\_df <- surv\_spr\_pVals\_df %>%  
 select(-terms, - setsize, -trim\_size) %>%  
 melt(id.vars = "pathways") %>%  
 mutate(score = -log(value)) %>%  
 mutate(pathways = factor(pathways,  
 levels = rev(unique(pathways)),  
 ordered = TRUE))

## Plot Significant Survival Pathways

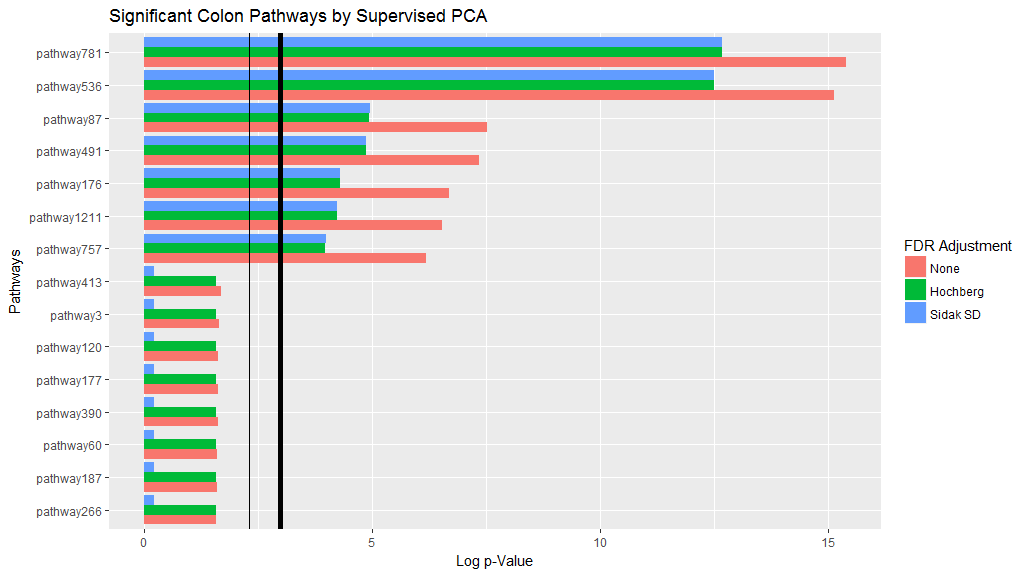
Now we will plot the pathway -values for the most significant pathways as a horizontal bar chart. The thick vertical black line is at , and the thin vertical black line is at . This figure shows that a few of the simulated pathways are significant at the for either the Hochberg or Sidak FDR approaches.

ggplot(surv\_aes\_melt\_df) +  
 aes(x = pathways, y = score, fill = variable) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 scale\_fill\_discrete(name = "FDR Adjustment",  
 breaks = c("rawp", "Hochberg", "SidakSD"),  
 labels = c("None", "Hochberg", "Sidak SD")) +  
 ggtitle("Significant Colon Pathways by AES-PCA") +  
 xlab("Pathways") +  
 ylab("Log p-Value") +  
 geom\_hline(yintercept = -log(0.05), size = 2) +  
 geom\_hline(yintercept = -log(0.1)) +  
 coord\_flip()



For Supervised PCA, we see similar results to AES-PCA, but nearly half of the simulated pathways are significant. (Recall that the example data was curated so that seven of the 15 pathways would be significant.) Further note that the order of the significance may not necessarily be preserved.

ggplot(surv\_spr\_melt\_df) +  
 aes(x = pathways, y = score, fill = variable) +  
 geom\_bar(stat = "identity", position = "dodge") +  
 scale\_fill\_discrete(name = "FDR Adjustment",  
 breaks = c("rawp", "Hochberg", "SidakSD"),  
 labels = c("None", "Hochberg", "Sidak SD")) +  
 ggtitle("Significant Colon Pathways by Supervised PCA") +  
 xlab("Pathways") +  
 ylab("Log p-Value") +  
 geom\_hline(yintercept = -log(0.05), size = 2) +  
 geom\_hline(yintercept = -log(0.1)) +  
 coord\_flip()



## Rank Significant Genes

Given that we have so far only considered collections of genes, rather than the genes themselves, we can inspect which genes show up the most often in the top-ranked pathways. Our method is to build a matrix of pathways and their genes: each column is a pathway, and each row is a gene. The entry of this matrix is a 1 if gene is contained in pathway (after trimming to the genes measured in the supplied assay). Now consider the adjusted -values of pathway , , where are the FDR adjustment methods; we define the pathway score for this pathway as

We then mulitply each of the columns in the matrix by its respective score.

Now consider the rows of this matrix. Each row is a gene, and the entry in the row is equal to , if the gene is included in pathway , and 0 otherwise. These genes can be compared by the raw row sum of scores, or by the average of the non-zero scores. The topGenes() function returns the top 5% of both. As we can see, most of the genes included in the summedRank vector are the in the list of “usual suspects” for colon cancer.

topGenes(object = colon\_OmicsSurv, pVals\_df = surv\_aes\_pVals\_df, percentile = 0.02)  
#> $summedRank  
#> PIK3CA PIK3R1 PIK3CB PIK3R2 PIK3R4 PIK3C3 MAPK1 MAP2K1   
#> 26.43745 26.43745 22.10158 22.10158 21.03848 21.03848 17.51650 17.48348   
#> NRAS GRB2 MAPK3 HRAS KRAS   
#> 17.48348 17.48348 17.48348 17.48348 17.48348   
#>   
#> $averagedRank  
#> COL4A3BP LPCAT3 CEPT1 CDS1 PEMT CDIPT AGPAT1 AGPAT2   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> SPTLC1 NEU3 CHAT PLA2G16 CHKA CHKB MGLL PLD4   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> DEGS2 ACER1 SLC44A3 MBOAT2 NEU4 SGPP2 CSNK1G2 PIK3R6   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> GPAT2 PPM1L ARSK MBOAT1 PHOSPHO1 SGMS2 PIKFYVE PLD6   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> CERS3 SLC44A5 ARSG SACM1L GPD1L LPIN1 PIP5K1C SLC44A1   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> PITPNB KDSR LCLAT1 CERS6 LPCAT4 PLA2G4F GALC SUMF2   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> SGMS1 PLA2G2D INPP5J GLA GM2A GPD1 PLA2G4D SUMF1   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> CERS2 HADHA HADHB HEXA HEXB PLA2G2E ENPP7 ARSI   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> INPPL1 ARF1 ARF3 ARSA ARSB STS ARSE ARSF   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> ASAH1 ACHE MTM1 NEU1 NEU2 OCRL OSBP PLA2G3   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> PNPLA8 PCYT1A INPP5K PIK3C2A PIK3C2B PIK3C2G PI4KA PI4KB   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> PIP4K2A PLA2G1B PLA2G2A PLA2G4A PLA2G5 CRLS1 CTSA LPCAT2   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> ETNK2 AGPAT5 ACER3 PI4K2A ETNK1 SMPD3 VAC14 PRKD1   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> GPCPD1 PSAP INPP5E SPHK2 AGPAT3 AGPAT4 CHPT1 PNPLA2   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> SLC44A2 PTEN GPAM GBA2 GBA3 PCYT2 MTMR14 PLA2G2F   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> CERK LPIN3 TAZ UGCG MBOAT7 CERS4 ARSJ PLBD1   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> LPCAT1 PNPLA3 SLC44A4 PTDSS2 SGPP1 PLA2G12A PIP5K1B PIP4K2B   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> PLA2G6 GNPAT DEGS1 PLA2G4C CDS2 MTMR1 INPP4B SYNJ2   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> SPHK1 SGPL1 MTMR3 MTMR2 CERS5 MTMR6 MTMR7 MTMR4   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> VAPB VAPA PCYT1B PGS1 GAL3ST1 SPTLC2 LPIN2 PTDSS1   
#> 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972 8.953972   
#> FIG4 LPGAT1   
#> 8.953972 8.953972

Currently, this is the last vignette in the workflow. We are exploring options to connect this package to other pathway-testing software. We are further considering other summary functions and/or graphics to help display the output of our \*\_pVals() functions. If you have any questions, complaints, or suggestions, please [email us](mailto:gabriel.odom@med.miami.edu) and give us some feedback. Also you can visit our [development page](https://github.com/gabrielodom/pathwayPCA).