Chapter 1 - Quickstart Guide

Gabriel Odom

2018-04-19

Table of Contents

[Overview 1](#_Toc511925741)

[Import Data 2](#_Toc511925742)

[Import .gmt Files 2](#_Toc511925743)

[Import and Tidy Assay Data 3](#_Toc511925744)

[Import Patient Info 4](#_Toc511925745)

[Match the Patient and Assay Data 5](#_Toc511925746)

[Create an Omics Data Object 5](#_Toc511925747)

[Create an Object 5](#_Toc511925748)

[Inspect the Object 6](#_Toc511925749)

[Detailed Object Views 6](#_Toc511925750)

[Test Pathways for Significance 8](#_Toc511925751)

[AES-PCA 8](#_Toc511925752)

[Supervised PCA 9](#_Toc511925753)

[Inspect Results 9](#_Toc511925754)

[Graph of Top Pathways 10](#_Toc511925755)

[Extract Genes from the Top Pathways 11](#_Toc511925756)

[Review 12](#_Toc511925757)

# Overview

This guide will serve as a brief introduction to pathway significance testing with the pathwayPCA package. We will discuss four steps. For detailed discussion of these four steps, see the appropriate vignettes. This vignette is the introduction and overview chapter in the “Pathway Significance Testing with pathwayPCA” workflow.

1. Download Packages
2. Import Data ([*vignette*](https://gabrielodom.github.io/pathwayPCA/articles/C2-Importing_Data.html))
3. Create Omics Data Object ([*vignette*](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html))
4. Test Pathway Significance ([*vignette*](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html))
5. Inspect Results

Before we get started, you need the pathwayPCA package to run your analysis. Because we are currently in the development phase for Bioconductor, you can install this package from GitHub. Also, if you want your analysis to be performed with parallel computing, you will need a package to help you. We recommend the parallel package. We also recommend the tidyverse package to help you run some of the examples in these vignettes (while the tidyverse package suite is required for many of the examples in the vignettes, it is not required for any of the functions in this package).

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

library(pathwayPCA)  
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(parallel)

# Import Data

This section is a quick overview of the material covered in the [Import and Tidy Data](https://gabrielodom.github.io/pathwayPCA/articles/C2-Importing_Data.html) vignette. We will cover three data import steps.

## Import .gmt Files

GMT files are one form of gene set file officially recognized by the Gene Set Enrichment Analysis committee of the Broad Institute. These Molecular Signatures Database (MSigDB) GMT files can be downloaded from the [MSigDB Collections](http://software.broadinstitute.org/gsea/msigdb/collections.jsp) page. Use the read\_gmt function to import a .gmt file into R.

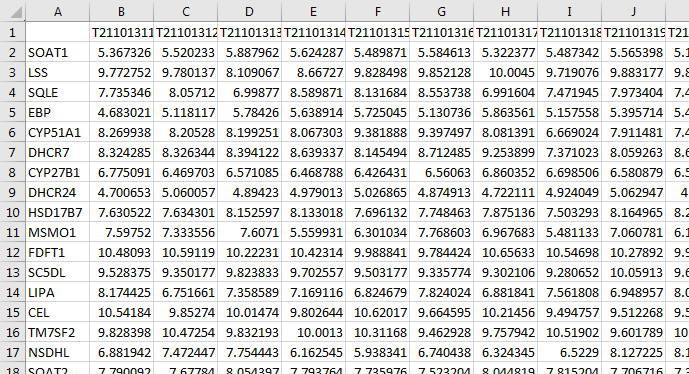
gene\_set\_ls <- read\_gmt("path\_to\_your\_directory/your\_file\_name.gmt")  
gene\_set\_ls

#> Object with Class(es) 'pathwaySet', 'list' [package 'pathwayPCA'] with 3 elements:   
#> $ pathways :List of 1329  
#> $ TERMS : chr [1:1329] "KEGG\_GLYCOLYSIS\_GLUCONEOGENESIS" ...  
#> $ GSEA\_link: chr [1:1329] "http://www.broadinstitute.org/gsea/msigdb/cards/KEGG\_GLYCOLYSIS\_GLUCONEOGENESIS" ...

The imported .gmt file is stored as a pathwaySet list object. This list contains the names of the pathways (TERMS), the hyperlink to the pathway description card on the GSEA website (GSEA\_link), and a list of all the pathway sets (pathways).

## Import and Tidy Assay Data

We assume that the assay data (be it genomic, proteomic, metabolomic, lipidomic, or transcriptomic data) is either in an Excel file or flat text file. For example, your data may look like this:



In this data set, the columns are individual samples, patients, tumors, cell lines, etc. The rows are the -Omic expression measures. Use the read\_csv function from the readr package to import .csv files and the like. If your data has samples in the columns and -omic feature measurements in the rows, you’ll need to “tidy” the imported assay with the transpose\_assay function.

assay\_df <- read\_csv("path\_to\_your\_directory/your\_assay.csv")  
assayT\_df <- transpose\_assay(assay\_df)  
assayT\_df

#> Warning: Missing column names filled in: 'X1' [1]  
#> Parsed with column specification:  
#> cols(  
#> .default = col\_double(),  
#> X1 = col\_character()  
#> )  
#> See spec(...) for full column specifications.  
#> # A tibble: 36 x 18  
#> Sample SOAT1 LSS SQLE EBP CYP51A1 DHCR7 CYP27B1 DHCR24 HSD17B7  
#> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 T21101311 5.37 9.77 7.74 4.68 8.27 8.32 6.78 4.70 7.63  
#> 2 T21101312 5.52 9.78 8.06 5.12 8.21 8.33 6.47 5.06 7.63  
#> 3 T21101313 5.89 8.11 7.00 5.78 8.20 8.39 6.57 4.89 8.15  
#> 4 T21101314 5.62 8.67 8.59 5.64 8.07 8.64 6.47 4.98 8.13  
#> 5 T21101315 5.49 9.83 8.13 5.73 9.38 8.15 6.43 5.03 7.70  
#> 6 T21101316 5.58 9.85 8.55 5.13 9.40 8.71 6.56 4.87 7.75  
#> 7 T21101317 5.32 10.0 6.99 5.86 8.08 9.25 6.86 4.72 7.88  
#> 8 T21101318 5.49 9.72 7.47 5.16 6.67 7.37 6.70 4.92 7.50  
#> 9 T21101319 5.57 9.88 7.97 5.40 7.91 8.06 6.58 5.06 8.16  
#> 10 T21101320 5.16 9.87 7.42 5.50 7.43 8.68 6.55 4.85 8.20  
#> # ... with 26 more rows, and 8 more variables: MSMO1 <dbl>, FDFT1 <dbl>,  
#> # SC5DL <dbl>, LIPA <dbl>, CEL <dbl>, TM7SF2 <dbl>, NSDHL <dbl>,  
#> # SOAT2 <dbl>

## Import Patient Info

Use the read\_csv function to import the patient data.

pInfo\_df <- read\_csv("path\_to\_your\_directory/your\_subject\_info.csv")  
pInfo\_df

#> Parsed with column specification:  
#> cols(  
#> Sample = col\_character(),  
#> eventTime = col\_double(),  
#> eventObserved = col\_logical()  
#> )  
#> # A tibble: 36 x 3  
#> Sample eventTime eventObserved  
#> <chr> <dbl> <lgl>   
#> 1 T21101311 14.2 TRUE   
#> 2 T21101312 1.00 TRUE   
#> 3 T21101313 6.75 FALSE   
#> 4 T21101314 8.50 TRUE   
#> 5 T21101315 7.25 FALSE   
#> 6 T21101316 5.00 TRUE   
#> 7 T21101317 20.0 TRUE   
#> 8 T21101318 13.2 FALSE   
#> 9 T21101319 7.75 FALSE   
#> 10 T21101320 9.00 FALSE   
#> # ... with 26 more rows

## Match the Patient and Assay Data

Now that you have the measurement data in tidy form (assayT\_df) and the patient response data (pInfo\_df), you can use the inner\_join function from the dplyr package to match the assay measurements to patient information by subject identifier.

(joinedExperiment\_df <- inner\_join(pInfo\_df, assayT\_df, by = "Sample"))  
#> # A tibble: 36 x 20  
#> Sample eventTime eventObserved SOAT1 LSS SQLE EBP CYP51A1 DHCR7  
#> <chr> <dbl> <lgl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 T21101311 14.2 TRUE 5.37 9.77 7.74 4.68 8.27 8.32  
#> 2 T21101312 1.00 TRUE 5.52 9.78 8.06 5.12 8.21 8.33  
#> 3 T21101313 6.75 FALSE 5.89 8.11 7.00 5.78 8.20 8.39  
#> 4 T21101314 8.50 TRUE 5.62 8.67 8.59 5.64 8.07 8.64  
#> 5 T21101315 7.25 FALSE 5.49 9.83 8.13 5.73 9.38 8.15  
#> 6 T21101316 5.00 TRUE 5.58 9.85 8.55 5.13 9.40 8.71  
#> 7 T21101317 20.0 TRUE 5.32 10.0 6.99 5.86 8.08 9.25  
#> 8 T21101318 13.2 FALSE 5.49 9.72 7.47 5.16 6.67 7.37  
#> 9 T21101319 7.75 FALSE 5.57 9.88 7.97 5.40 7.91 8.06  
#> 10 T21101320 9.00 FALSE 5.16 9.87 7.42 5.50 7.43 8.68  
#> # ... with 26 more rows, and 11 more variables: CYP27B1 <dbl>,  
#> # DHCR24 <dbl>, HSD17B7 <dbl>, MSMO1 <dbl>, FDFT1 <dbl>, SC5DL <dbl>,  
#> # LIPA <dbl>, CEL <dbl>, TM7SF2 <dbl>, NSDHL <dbl>, SOAT2 <dbl>

# Create an Omics Data Object

This section is a quick overview of the material covered in the [Create an Omics\* Data Container](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html) vignette.

## Create an Object

Using the data you just imported, create a data container specific to survival, regression, or categorical responses. There are three classes of Omics\* objects to match the three types of response. Each class contains a tidy assay and pathwaySet gene set list, but the classes differ in the type of response information they can hold. The classes, and their responses, are

1. OmicsSurv—a data container for survival information, which includes event time (the time of last follow-up with a subject) and event indicator (did the subject die, or was the observation right-censored?).
2. OmicsReg—a data container for continuous responses (usually a linear regression response).
3. OmicsCateg—a data container for categorical responses, the dependent variable of a generalized linear model. Currently, we only support binary classification (through logistic regression).

We will create an OmicsSurv object to hold our assay, pathway set list, and survival response.

colon\_OmicsSurv <- create\_Omics(assayData\_df = colonSurv\_df[, -(1:2)],  
 pathwaySet\_ls = colon\_pathwaySet,  
 response = colonSurv\_df[, 1:2],  
 respType = "surv")  
#> Creating object of class OmicsSurv.

## Inspect the Object

After you create an Omics\*-class object, print the object to the screen to see a summary of the data contained therein.

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"

## Detailed Object Views

Because the printing procedure for Omics\*-objects is to show a summary of the contents, you need to use the get\*() functions to view the individual components of the colon\_OmicsSurv object we just created. Overall, you can use accessor functions to extract, edit, or replace data contained in the object. The accessor functions are listed in more detail in the [Table of Accessors](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html#table-of-accessors) subsection of Chapter 3. Use these functions to confirm that the data container you created accurately reflects the data you intend to analyze.

### View the Assay

getAssay(colon\_OmicsSurv)  
#> # A tibble: 250 x 656  
#> JUN SOS2 PAK3 RAF1 PRKCB BTC SHC1 PRKCA ELK1 NRG1 PAK2 MTOR  
#> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 9.29 5.48 8.21 8.03 5.49 6.65 8.26 8.94 7.38 7.50 7.32 6.96  
#> 2 9.13 6.35 8.33 7.94 6.26 7.02 8.39 9.61 7.53 7.68 6.80 6.96  
#> 3 9.37 5.67 7.82 7.74 6.05 7.52 8.69 8.40 7.25 7.33 7.48 7.15  
#> 4 10.6 4.94 8.79 7.64 5.37 6.87 7.81 9.80 7.79 8.38 6.16 6.48  
#> 5 8.70 5.60 8.75 8.05 6.07 6.49 8.45 8.21 7.60 6.65 7.04 6.66  
#> 6 9.78 5.36 7.56 8.07 5.90 6.39 8.87 8.22 7.35 7.83 7.39 6.90  
#> 7 9.22 5.05 8.20 7.80 5.55 6.86 8.28 8.97 7.43 7.20 7.04 6.96  
#> 8 10.3 5.33 7.82 7.89 6.27 6.25 8.66 9.71 7.38 7.09 7.22 7.11  
#> 9 10.8 5.07 7.63 7.69 5.48 7.57 8.36 9.69 6.66 7.22 6.99 6.89  
#> 10 9.52 5.50 7.48 7.53 5.71 7.33 8.54 8.14 6.88 7.31 7.01 6.82  
#> # ... with 240 more rows, and 644 more variables: PAK4 <dbl>,  
#> # MAP2K4 <dbl>, EIF4EBP1 <dbl>, BAD <dbl>, PRKCG <dbl>, NRG3 <dbl>,  
#> # MAPK9 <dbl>, ERBB4 <dbl>, MAPK10 <dbl>, PTK2 <dbl>, ERBB2 <dbl>,  
#> # ERBB3 <dbl>, MAP2K2 <dbl>, TGFA <dbl>, BRAF <dbl>, MAP2K1 <dbl>,  
#> # MAP2K7 <dbl>, ABL1 <dbl>, NRG2 <dbl>, AKT1 <dbl>, ABL2 <dbl>,  
#> # AKT2 <dbl>, SHC4 <dbl>, RPS6KB1 <dbl>, RPS6KB2 <dbl>, AKT3 <dbl>,  
#> # NRAS <dbl>, GRB2 <dbl>, AREG <dbl>, STAT5B <dbl>, MAPK3 <dbl>,  
#> # STAT5A <dbl>, PAK6 <dbl>, SOS1 <dbl>, MYC <dbl>, MAPK1 <dbl>,  
#> # NCK1 <dbl>, PIK3R5 <dbl>, NRG4 <dbl>, HRAS <dbl>, MAPK8 <dbl>,  
#> # EGFR <dbl>, GSK3B <dbl>, CBLB <dbl>, KRAS <dbl>, CBL <dbl>,  
#> # SHC3 <dbl>, CDKN1B <dbl>, CDKN1A <dbl>, EGF <dbl>, EREG <dbl>,  
#> # ARAF <dbl>, NCK2 <dbl>, SRC <dbl>, PIK3R3 <dbl>, CAMK2A <dbl>,  
#> # CAMK2B <dbl>, CAMK2D <dbl>, CAMK2G <dbl>, PAK1 <dbl>, CBLC <dbl>,  
#> # CRK <dbl>, PIK3CA <dbl>, PIK3CB <dbl>, CRKL <dbl>, PIK3CD <dbl>,  
#> # GAB1 <dbl>, PLCG1 <dbl>, PLCG2 <dbl>, SHC2 <dbl>, HBEGF <dbl>,  
#> # PIK3CG <dbl>, PIK3R1 <dbl>, PIK3R2 <dbl>, EPHB2 <dbl>, EPHB4 <dbl>,  
#> # EFNA5 <dbl>, PXN <dbl>, CDC42 <dbl>, EFNB3 <dbl>, RRAS <dbl>,  
#> # GRB7 <dbl>, SYNJ1 <dbl>, EPHB3 <dbl>, EFNB1 <dbl>, DNM1 <dbl>,  
#> # MAP4K4 <dbl>, GRIA1 <dbl>, EPHB1 <dbl>, ROCK1 <dbl>, ITSN1 <dbl>,  
#> # RAP1A <dbl>, RAC1 <dbl>, RAP1B <dbl>, EFNB2 <dbl>, WASL <dbl>,  
#> # TF <dbl>, KALRN <dbl>, RASA1 <dbl>, CASP9 <dbl>, ...

### View the pathwaySet List

getPathwaySet(colon\_OmicsSurv)  
#> Object with Class(es) 'pathwaySet', 'list' [package 'pathwayPCA'] with 3 elements:   
#> $ pathways:List of 15  
#> $ TERMS : Named chr [1:15] "KEGG\_PENTOSE\_PHOSPHATE\_PATHWAY" ...  
#> $ setsize : Named int [1:15] 27 64 ...

### View the Event Time

head(getEventTime(colon\_OmicsSurv), 10)  
#> [1] 64.8657534 59.7698630 62.4000000 54.5095890 46.2904110 55.8575343  
#> [7] 57.9616438 54.0493151 0.4273973 41.4246575

### View the Event Indicator

head(getEvent(colon\_OmicsSurv), 10)  
#> [1] FALSE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE

# Test Pathways for Significance

After you have confirmed that the create\_Omics function created the Omics\* object you wanted, you can analyze the object with adaptive, elastic-net, sparse (AES) PCA or supervised PCA. This section is a quick overview of the material covered in the “AES-PCA” and “Supervised PCA” sections of the [Test Pathway Significance](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html) vignette. For details of these methods functions, please see their respective sections in Chapter 4.

The function arguments are these. AES-PCA uses permutation-based -values, so the numReps argument controls how many permutations to take. The numPCs argument specifies how many principal components will be extracted from each pathway. The parallel and numCores arguments are used to control if and how the functions make use of parallel computing. Finally, the adjustment argument allows you to specify a family-wise error rate (FWER) or false discovery rate (FDR) adjustment for the pathway -values. These options are documented in the adjustRaw\_pVals function (see the [help documentation](https://gabrielodom.github.io/pathwayPCA/reference/adjustRaw_pVals.html) for details).

## AES-PCA

Perform AES-PCA pathway significance testing on the object with the AESPCA\_pVals function. For more details on this function, see the [AES-PCA](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html#aes-pca) section of Chapter 4.

surv\_aes\_pVals\_df <- AESPCA\_pVals(object = colon\_OmicsSurv,  
 numReps = 1000,  
 numPCs = 2,  
 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

## Supervised PCA

Perform Supervised PCA pathway significance testing on the object with the superPCA\_pVals function. For more details on this function, see the [Supervised PCA](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html#supervised-pca) section of Chapter 4.

surv\_spr\_pVals\_df <- superPCA\_pVals(object = colon\_OmicsSurv,  
 numPCs = 2,  
 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

# Inspect Results

This section is a quick overview of the material covered in the “Analyze the Results” section of the [Test Pathway Significance](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html) vignette. For a quick and easy view of the pathway significance testing results, we can simply print the output data frame. If you are not using the tidyverse package suite, your results will print differently (use the head function to print the top pathways instead).

surv\_spr\_pVals\_df  
#> # A tibble: 15 x 7  
#> pathways setsize trim\_size terms rawp Hochberg SidakSD  
#> \* <chr> <int> <int> <chr> <dbl> <dbl> <dbl>  
#> 1 pathway87 87 86 KEGG\_ERBB\_SIGNA~ 4.89e-5 0.000733 7.33e-4  
#> 2 pathway491 40 40 PID\_EPHB\_FWD\_PA~ 5.59e-5 0.000783 7.83e-4  
#> 3 pathway176 54 54 KEGG\_NON\_SMALL\_~ 1.48e-4 0.00192 1.92e-3  
#> 4 pathway1211 108 104 REACTOME\_SIGNAL~ 2.02e-4 0.00242 2.42e-3  
#> 5 pathway757 87 83 REACTOME\_INSULI~ 3.30e-4 0.00363 3.62e-3  
#> 6 pathway781 198 180 REACTOME\_PHOSPH~ 1.63e-3 0.0151 1.62e-2  
#> 7 pathway536 46 44 PID\_TNF\_PATHWAY 1.67e-3 0.0151 1.62e-2  
#> 8 pathway177 30 26 KEGG\_ASTHMA 2.17e-3 0.0169 1.73e-2  
#> 9 pathway390 29 29 BIOCARTA\_TNFR1\_~ 2.74e-3 0.0169 1.90e-2  
#> 10 pathway3 27 26 KEGG\_PENTOSE\_PH~ 2.82e-3 0.0169 1.90e-2  
#> 11 pathway413 23 23 ST\_GA12\_PATHWAY 4.68e-3 0.0234 2.32e-2  
#> 12 pathway60 64 45 KEGG\_RETINOL\_ME~ 9.51e-3 0.0380 3.75e-2  
#> 13 pathway187 16 16 BIOCARTA\_RELA\_P~ 3.37e-2 0.101 9.78e-2  
#> 14 pathway266 11 11 BIOCARTA\_SET\_PA~ 1.19e-1 0.238 2.24e-1  
#> 15 pathway120 89 73 KEGG\_ANTIGEN\_PR~ 3.30e-1 0.330 3.30e-1

## Graph of Top Pathways

To visualize the significance of the pathways based on the selected FDR method, we can use the ggplot2 package to create summary graphics of the analysis results.

### Tidy Up the Data

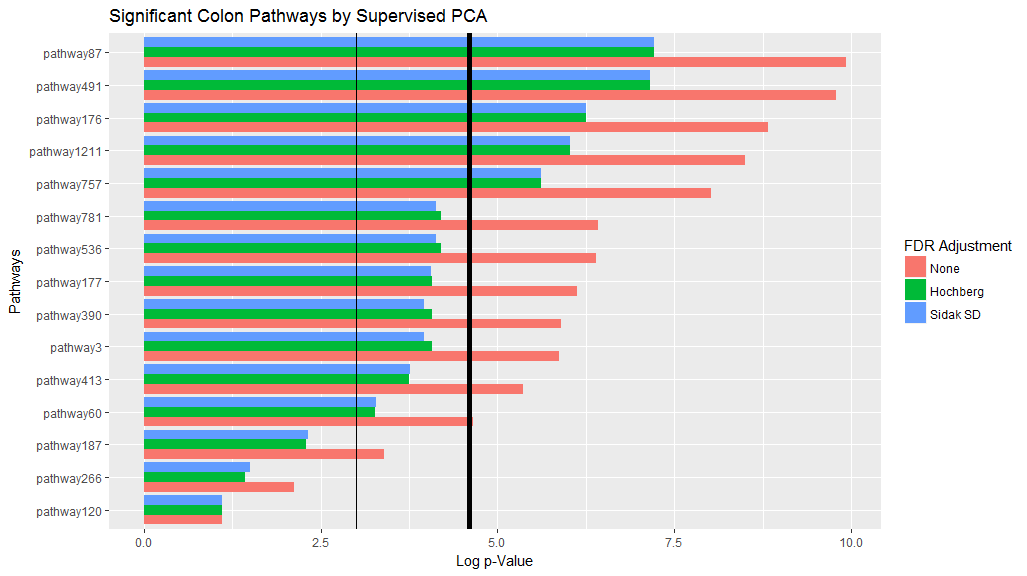
In order to take advantage of the publication-quality graphics created with the ggplot2 package, we first need to tidy the data frames returned by the AESPCA\_pVals and superPCA\_pVals functions.

library(reshape2)  
#>   
#> Attaching package: 'reshape2'  
#> The following object is masked from 'package:tidyr':  
#>   
#> smiths  
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))

### Graph Pathway Ranks

Now that our output is tidy, we can make a bar chart of the pathway significance.

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(0.01), size = 2) +  
 geom\_hline(yintercept = -log(0.05)) +  
 coord\_flip()



## Extract Genes from the Top Pathways

Use the topGenes function to “score” the genes contained in the top significant pathways. 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.

topGenes(object = colon\_OmicsSurv, pVals\_df = surv\_spr\_pVals\_df)  
#> $summedRank  
#> PIK3CA PIK3R1 MAPK1 MAP2K1 NRAS GRB2 MAPK3 HRAS   
#> 36.43021 36.43021 36.03006 32.27087 32.27087 32.27087 32.27087 32.27087   
#> KRAS   
#> 32.27087   
#>   
#> $averagedRank  
#> PAK3 BTC ELK1 NRG1 PAK4 NRG3 MAPK9 ERBB4   
#> 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563   
#> MAPK10 ERBB3 ABL1 NRG2 ABL2 SHC4 RPS6KB2 AREG   
#> 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563   
#> STAT5B STAT5A PAK6 MYC NRG4 GSK3B CBLB CBL   
#> 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563   
#> CDKN1B CDKN1A EREG NCK2 CAMK2A CAMK2B CAMK2D CAMK2G   
#> 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563 7.218563   
#> CBLC CRKL HBEGF   
#> 7.218563 7.218563 7.218563

# Review

Now that you have an idea of how to use this package, please see each of our vignettes for detailed and thorough commentary and guiding information on each of the three topics discussed herein. The vignettes are:

* [Chapter 2: Import Data](https://gabrielodom.github.io/pathwayPCA/articles/C2-Importing_Data.html)
* [Chapter 3: Create Omics Data Objects](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html)
* [Chapter 4: Test Pathway Significance](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html)