Chapter 1 - Quickstart Guide

Gabriel Odom

2018-04-27

Table of Contents

[1. Overview 1](#_Toc512613227)

[2. Import Data 2](#_Toc512613228)

[2.1 Import .gmt Files 2](#_Toc512613229)

[2.2 Import and Tidy Assay Data 3](#_Toc512613230)

[2.3 Import Phenotype Info 4](#_Toc512613231)

[2.4 Match the Phenotype and Assay Data 5](#_Toc512613232)

[3. Create an Omics Data Object 6](#_Toc512613233)

[3.1 Create an Object 6](#_Toc512613234)

[3.2 Inspect the Object 6](#_Toc512613235)

[3.3 Detailed Object Views 7](#_Toc512613236)

[4. Test Pathways for Significance 8](#_Toc512613237)

[4.1 AES-PCA 8](#_Toc512613238)

[4.2 Supervised PCA 9](#_Toc512613239)

[5. Inspect Results 10](#_Toc512613240)

[5.1 Analysis Output Table 10](#_Toc512613241)

[5.2 Graph of Top Pathways 10](#_Toc512613242)

[6. Review 12](#_Toc512613243)

# 1. 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 Objects ([*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. 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")

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 (it comes with R automatically). 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).

install.packages("tidyverse")

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)

# 2. 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.

## 2.1 Import .gmt Files

The .gmt format is a commonly used file format for storing [pathway set information](https://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats#GMT:_Gene_Matrix_Transposed_file_format_.28.2A.gmt.29). Gene sets in the Molecular Signatures Database (MSigDB) can be downloaded from the MSigDB Collections page. Use the read\_gmt function to import a .gmt file into R. All .gmt files have a “description” field, but it can be empty. We use description = FALSE argument to skip importing that field.

gmt\_path <- system.file("extdata", "c2.cp.v6.0.symbols.gmt",  
 package = "pathwayPCA", mustWork = TRUE)  
gene\_set\_ls <- read\_gmt(gmt\_path, description = FALSE)  
gene\_set\_ls

#> Object with Class(es) 'pathwaySet', 'list' [package 'pathwayPCA'] with 2 elements:   
#> $ pathways:List of 1329  
#> $ TERMS : chr [1:1329] "KEGG\_GLYCOLYSIS\_GLUCONEOGENESIS" ...

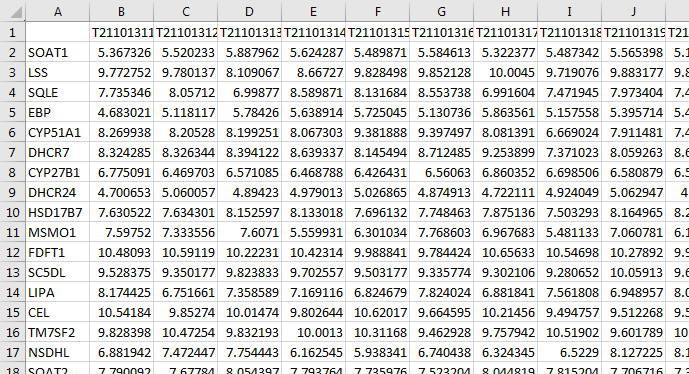
The imported .gmt file is stored as a pathwaySet list object. This list contains:

* a list of the gene names contained in each pathway set (pathways),
* the names of the pathways (TERMS), and
* (*OPTIONAL*) the hyperlink to the pathway description card on the GSEA website (description) if you specify description = TRUE.

Specifically, the description field contains any notes or details on the pathway that is stored in the second entry of each row of the .gmt file. This field is often left empty.

## 2.2 Import and Tidy Assay Data

We assume that the assay data (e.g. 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. The values in each row are the -Omic expression measurements for that gene. Use the read\_csv function from the readr package to import .csv files into R as [tibble](https://cran.r-project.org/web/packages/tibble/vignettes/tibble.html) (table *and* data frame) objects. The read\_csv function prints messages to the screen informing you of the assumptions it makes when importing your data. Specifically, this message tells us that all the imported data is numeric (col\_double()) except for the gene name column (X1 = col\_character()).

assay\_path <- system.file("extdata", "ex\_assay\_subset.csv",  
 package = "pathwayPCA", mustWork = TRUE)  
assay\_df <- read\_csv(assay\_path)

#> 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.

Incidentally, we consider gene names to adhere to the following conditions:

* gene names must start with an English letter (a-z or A-Z), and
* gene names can only contain English letters, Arabic numerals (0-9), and possibly a dash (-).

Furthermore, if your data has samples in the columns and -omic feature measurements in the rows (like the data set above), you’ll need to “tidy” the imported assay with the transpose\_assay function. Now see that the rows from our Excel file are the columns in the tidy data before performing data analysis.

#> # 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>

## 2.3 Import Phenotype Info

Use the read\_csv function to import the phenotype data. Once again, the read\_csv function displays a message informing us of the types of data in each column. This data is subject survival information which contains the subject ID (Sample), survival time after disease onset in months (eventTime), and an indicator if the subject died or was lost to follow up (eventObserved).

pInfo\_path <- system.file("extdata", "ex\_pInfo\_subset.csv",  
 package = "pathwayPCA", mustWork = TRUE)  
pInfo\_df <- read\_csv(pInfo\_path)  
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

## 2.4 Match the Phenotype and Assay Data

Now that you have the measurement data in tidy form (assayT\_df) and the response data (pInfo\_df), you can use the inner\_join function from the dplyr package to match the assay measurements to phenotype 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>

# 3. Create an Omics Data Object

This section is a quick overview of the material covered in the [Creating Data Objects](https://gabrielodom.github.io/pathwayPCA/articles/C3-Create_Omics_Objects.html) vignette.

## 3.1 Create an Object

Using the data you just imported, create a data object specific to survival, regression, or categorical responses. There are three classes of Omics objects to match the three types of response, but only one function to learn: the create\_Omics function. Each class contains assay information (in tidy format) and pathwaySet (in list format), but the classes differ in the type of response information they can hold. The classes, and their responses, are

1. OmicsSurv—a data object 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 object for continuous responses (usually a vector of linear regression responses).
3. OmicsCateg—a data object for categorical responses, the dependent variable of a generalized linear model. Currently, we only support binary classification (through logistic regression).

We will create a survival Omics 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.

## 3.2 Inspect the Object

After you create an Omics object, print the object to the screen to view 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"

## 3.3 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 object you created accurately reflects the data you intend to analyze.

### 3.3.1 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>, ...

### 3.3.2 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 ...

### 3.3.3 View the Event Time

We can use the vector subsetting mechanic in R ([]) to view only the first ten event times.

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

### 3.3.4 View the Event Indicator

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

# 4. 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 as follows. Both the AESPCA\_pVals and superPCA\_pVals functions take in an Omics object as the value to the object argument. 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).

## 4.1 AES-PCA

Perform AES-PCA pathway significance testing on the Omics 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 = "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 Computing Cluster  
#> DONE  
#> Extracting Pathway PCs in Parallel  
#> DONE  
#>   
#> Part 2: Calculate Permuted Pathway p-Values  
#> Initializing Computing Cluster  
#> DONE  
#> Extracting Pathway p-Values in Parallel  
#> DONE  
#>   
#> Part 3: Adjusting p-Values and Sorting Pathway p-Value Data Frame  
#> DONE

## 4.2 Supervised PCA

Perform Supervised PCA pathway significance testing on the Omics 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 = "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 Computing 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

# 5. 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.

## 5.1 Analysis Output Table

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 6  
#> pathways setsize trim\_size terms rawp BH  
#> \* <chr> <int> <int> <chr> <dbl> <dbl>  
#> 1 pathway536 46 44 PID\_TNF\_PATHWAY 2.42e-4 0.00112  
#> 2 pathway781 198 180 REACTOME\_PHOSPHOLIPID\_ME~ 2.53e-4 0.00112  
#> 3 pathway87 87 86 KEGG\_ERBB\_SIGNALING\_PATH~ 2.59e-4 0.00112  
#> 4 pathway491 40 40 PID\_EPHB\_FWD\_PATHWAY 2.99e-4 0.00112  
#> 5 pathway390 29 29 BIOCARTA\_TNFR1\_PATHWAY 4.41e-4 0.00132  
#> 6 pathway176 54 54 KEGG\_NON\_SMALL\_CELL\_LUNG~ 6.58e-4 0.00162  
#> 7 pathway1211 108 104 REACTOME\_SIGNALING\_BY\_IN~ 8.31e-4 0.00162  
#> 8 pathway413 23 23 ST\_GA12\_PATHWAY 8.62e-4 0.00162  
#> 9 pathway757 87 83 REACTOME\_INSULIN\_RECEPTO~ 1.26e-3 0.00210  
#> 10 pathway177 30 26 KEGG\_ASTHMA 6.24e-3 0.00936  
#> 11 pathway3 27 26 KEGG\_PENTOSE\_PHOSPHATE\_P~ 7.77e-3 0.0106   
#> 12 pathway187 16 16 BIOCARTA\_RELA\_PATHWAY 1.05e-2 0.0131   
#> 13 pathway60 64 45 KEGG\_RETINOL\_METABOLISM 2.05e-2 0.0236   
#> 14 pathway266 11 11 BIOCARTA\_SET\_PATHWAY 5.17e-2 0.0554   
#> 15 pathway120 89 73 KEGG\_ANTIGEN\_PROCESSING\_~ 1.76e-1 0.176

## 5.2 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.

### 5.2.1 Tidy Up the Data

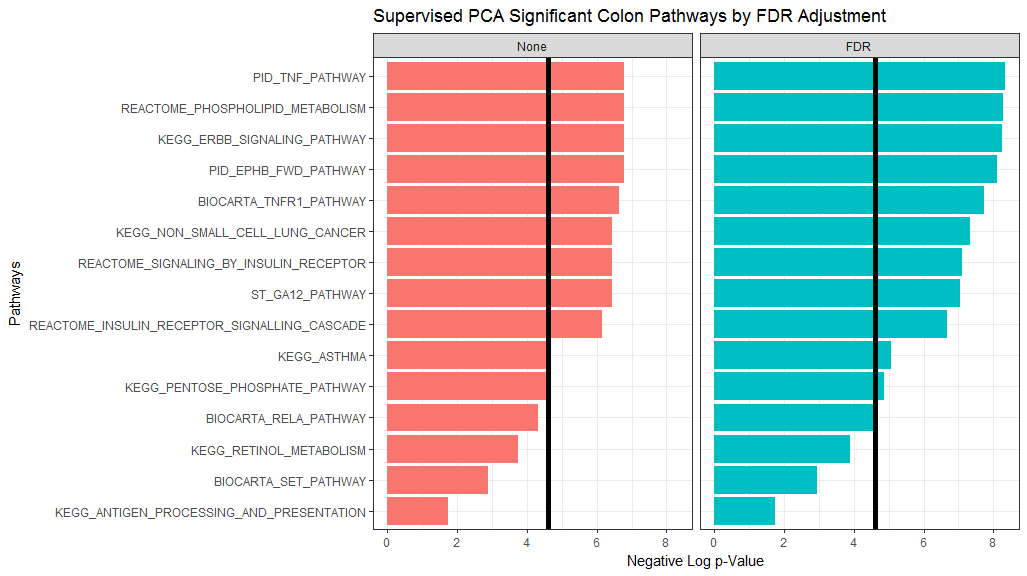
In order to take advantage of the publication-quality graphics created with the ggplot2 package, we first need to tidy the data frames returned by the AESPCA\_pVals and superPCA\_pVals functions. This code takes in the -values data frame from the Supervised PCA method and modifies it to be compatible with ggplot.

surv\_spr\_gather\_df <- surv\_spr\_pVals\_df %>%  
 select(-pathways, - setsize, -trim\_size) %>%  
 gather(variable, value, -terms) %>%  
 mutate(score = -log(value)) %>%  
 mutate(variable = factor(variable, labels = c("None", "FDR")))  
  
surv\_spr\_gather\_df  
#> # A tibble: 30 x 4  
#> terms variable value score  
#> <chr> <fct> <dbl> <dbl>  
#> 1 PID\_TNF\_PATHWAY FDR 0.000242 8.33  
#> 2 REACTOME\_PHOSPHOLIPID\_METABOLISM FDR 0.000253 8.28  
#> 3 KEGG\_ERBB\_SIGNALING\_PATHWAY FDR 0.000259 8.26  
#> 4 PID\_EPHB\_FWD\_PATHWAY FDR 0.000299 8.12  
#> 5 BIOCARTA\_TNFR1\_PATHWAY FDR 0.000441 7.73  
#> 6 KEGG\_NON\_SMALL\_CELL\_LUNG\_CANCER FDR 0.000658 7.33  
#> 7 REACTOME\_SIGNALING\_BY\_INSULIN\_RECEPTOR FDR 0.000831 7.09  
#> 8 ST\_GA12\_PATHWAY FDR 0.000862 7.06  
#> 9 REACTOME\_INSULIN\_RECEPTOR\_SIGNALLING\_CASCADE FDR 0.00126 6.68  
#> 10 KEGG\_ASTHMA FDR 0.00624 5.08  
#> # ... with 20 more rows

### 5.2.2 Graph Pathway Ranks

Now that our output is tidy, we can make a bar chart of the pathway significance. First, we consider the raw -value

ggplot(surv\_spr\_gather\_df) +  
 theme\_bw() +  
 aes(x = reorder(terms, score), y = score) +  
 geom\_bar(stat = "identity", position = "dodge", aes(fill = variable)) +  
 scale\_fill\_discrete(guide = FALSE) +  
 ggtitle("Supervised PCA Significant Colon Pathways by FDR Adjustment") +  
 xlab("Pathways") +  
 ylab("Negative Log p-Value") +  
 geom\_hline(yintercept = -log(0.01), size = 2) +  
 coord\_flip() +   
 facet\_grid(. ~ variable)



# 6. 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)