Chapter 3 - Creating Data Objects

Gabriel Odom

2018-04-27

Table of Contents

[1. Overview 1](#_Toc512613816)

[1.1 Outline 1](#_Toc512613817)

[1.2 Import Data 2](#_Toc512613818)

[2. Omics-Class Objects Defined 4](#_Toc512613819)

[2.1 Class Overview 4](#_Toc512613820)

[2.2 Review of Data Types in R 4](#_Toc512613821)

[3. Create New Omics Objects 5](#_Toc512613822)

[3.1 Overview of Subtypes 5](#_Toc512613823)

[3.2 Create a Survival Omics Data Object 6](#_Toc512613824)

[3.3 Regression and Classification Omics Data Objects 7](#_Toc512613825)

[4. Inspecting and Editing Omics-Class Objects 7](#_Toc512613826)

[4.1 Example “Get” Function 7](#_Toc512613827)

[4.2 Example “Set” Function 8](#_Toc512613828)

[4.3 Table of Accessors 9](#_Toc512613829)

[4.4 Inspect the Updated pathwaySet List 10](#_Toc512613830)

[5. Review 10](#_Toc512613831)

# 1. Overview

This vignette is the third chapter in the “Pathway Significance Testing with pathwayPCA” workflow, providing a detailed perspective to the [Creating Data Objects](https://gabrielodom.github.io/pathwayPCA/articles/C1-Quickstart_Guide.html#create-an-omics-data-object) section of the Quickstart Guide. This vignette builds on the material covered in the [“Import and Tidy Data”](https://gabrielodom.github.io/pathwayPCA/articles/C2-Importing_Data.html) vignette. This guide will outline the major steps needed to create a data object for analysis with the pathwayPCA package. These objects are called Omics-class objects.

## 1.1 Outline

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

1. Describe the components of the Omics object class.
2. Create a few Omics objects.
3. Inspect and edit individual elements contained in these objects.

First, load the pathwayPCA package and the [tidyverse package suite](https://www.tidyverse.org/).

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(pathwayPCA)

## 1.2 Import Data

Because this is the third chapter in the workflow, we assume that

1. Your assay is “tidy”.
2. Your gene pathway set / GMT information is stored in a pathwaySet object.
3. Your response and assay data have already been ID-matched.

If you are unsure about any of the three points above (or you don’t know what these mean), please review the [Import and Tidy Data](https://gabrielodom.github.io/pathwayPCA/articles/Importing_Data.html) vignette first. It isn’t very long, but it will help you set up your data in the right way.

For the purpose of example, we will load some “toy” data: a combined assay / phenotype data frame and a pathwaySet list which already fits the three criteria above. This tidy data set has 656 gene expression measurements (columns) on 250 colon cancer patients (rows). Notice that the assay and survival response information have already been merged, so we have two additional columns (for Overall Survival Time and its corresponding death indicator).

data("colonSurv\_df")  
colonSurv\_df  
#> # A tibble: 250 x 658  
#> OS\_time OS\_event JUN SOS2 PAK3 RAF1 PRKCB BTC SHC1 PRKCA ELK1  
#> <dbl> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
#> 1 64.9 0 9.29 5.48 8.21 8.03 5.49 6.65 8.26 8.94 7.38  
#> 2 59.8 0 9.13 6.35 8.33 7.94 6.26 7.02 8.39 9.61 7.53  
#> 3 62.4 0 9.37 5.67 7.82 7.74 6.05 7.52 8.69 8.40 7.25  
#> 4 54.5 0 10.6 4.94 8.79 7.64 5.37 6.87 7.81 9.80 7.79  
#> 5 46.3 1 8.70 5.60 8.75 8.05 6.07 6.49 8.45 8.21 7.60  
#> 6 55.9 0 9.78 5.36 7.56 8.07 5.90 6.39 8.87 8.22 7.35  
#> 7 58.0 0 9.22 5.05 8.20 7.80 5.55 6.86 8.28 8.97 7.43  
#> 8 54.0 0 10.3 5.33 7.82 7.89 6.27 6.25 8.66 9.71 7.38  
#> 9 0.427 1 10.8 5.07 7.63 7.69 5.48 7.57 8.36 9.69 6.66  
#> 10 41.4 0 9.52 5.50 7.48 7.53 5.71 7.33 8.54 8.14 6.88  
#> # ... with 240 more rows, and 647 more variables: NRG1 <dbl>, PAK2 <dbl>,  
#> # MTOR <dbl>, 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>, ...

We also have a small list of 15 pathways which correspond to our example colon cancer assay.

data("colon\_pathwaySet")  
colon\_pathwaySet  
#> Object with Class(es) 'pathwaySet', 'list' [package 'pathwayPCA'] with 2 elements:   
#> $ pathways:List of 15  
#> $ TERMS : chr [1:15] "KEGG\_PENTOSE\_PHOSPHATE\_PATHWAY" ...  
str(colon\_pathwaySet$pathways, list.len = 10)  
#> List of 15  
#> $ pathway3 : chr [1:27] "RPE" "RPIA" "PGM2" "PGLS" ...  
#> $ pathway60 : chr [1:64] "RPE65" "CYP3A5" "UGT2B28" "CYP4A11" ...  
#> $ pathway87 : chr [1:87] "JUN" "SOS2" "PAK3" "RAF1" ...  
#> $ pathway120 : chr [1:89] "HLA-DOA" "HLA-DOB" "KLRC3" "KLRD1" ...  
#> $ pathway176 : chr [1:54] "CASP9" "SOS2" "E2F1" "PRKCB" ...  
#> $ pathway177 : chr [1:30] "HLA-DRB4" "HLA-DRB5" "HLA-DOA" "HLA-DOB" ...  
#> $ pathway187 : chr [1:16] "IKBKG" "CHUK" "EP300" "RELA" ...  
#> $ pathway266 : chr [1:11] "PRF1" "DFFA" "DFFB" "HMGB2" ...  
#> $ pathway390 : chr [1:29] "JUN" "BAG4" "CASP8" "MAPK8" ...  
#> $ pathway413 : chr [1:23] "PLD1" "RAF1" "EPHB2" "VAV1" ...  
#> [list output truncated]

The pathway set list and tidy assay (with matched phenotype information) are all the information we need to create an Omics-class data object.

# 2. Omics-Class Objects Defined

Now that we have our data loaded, we can create an analysis object for the pathwayPCA package.

## 2.1 Class Overview

In this package, all primary input data will be in an Omics data object. There are three classes of Omics\* objects, but one function (create\_Omics) creates all of them. Each class contains a tidy assay and pathwaySet gene set list. 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 linear regression response).
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).
4. OmicsPathway—a data object with no response. This is the “parent” class for the other three Omics classes.

## 2.2 Review of Data Types in R

Take a quick look back at the structure of our colonSurv\_df object. We have an expression design matrix with the first two columns as subject response information. Look at the types of the columns of this data frame (notice the <dbl> <int> tags directly under the column names). These tags tell us that the columns contain “double / numeric” (dbl) and “integer” (int) information. The other tags we could potentially see here are <chr> (character), <lgl> (logical), or <fct> (factor). These tags are important because they identify which “class” of data is in each column. Here are some examples of how to change data between types:

head(colonSurv\_df$OS\_event)  
#> [1] 0 0 0 0 1 0  
head(as.character(colonSurv\_df$OS\_event))  
#> [1] "0" "0" "0" "0" "1" "0"  
head(as.logical(colonSurv\_df$OS\_event))  
#> [1] FALSE FALSE FALSE FALSE TRUE FALSE  
head(as.factor(colonSurv\_df$OS\_event))  
#> [1] 0 0 0 0 1 0  
#> Levels: 0 1

The create\_Omics function puts the response information into specific classes:

* Survival data is stored with a pair of numeric (time) and logical (death indicator) vectors.
* Regression data is stored in a numeric or integer vector.
* Binary classification data is stored in a factor vector.

These restrictions are on purpose: the internal data creation functions in the pathwayPCA package have very specific requirements about the types of data they take as inputs.

# 3. Create New Omics Objects

## 3.1 Overview of Subtypes

All new Omics objects are created with the create\_Omics function. You should use this function to create Omics-class objects for survival, regression, or categorical responses. This create\_Omics function *internally* calls on a specific creation function for each response type:

* Survival response: the create\_OmicsSurv() function creates an Omics object with class OmicsSurv. This object will contain:
  + eventTime: a numeric vector of event times.
  + eventObserved: a logical vector of death (or other event) indicators. This format precludes the option of recurrent-event survival analysis.
  + assayData\_df: a tidy data.frame or tibble of prediction information. Rows are observations or subjects; the columns are -Omics measures (e.g. transcriptome). The column names *must* match a subset of the genes provided in the gene set list (in the pathwaySet object).
  + pathwaySet: a list of pathway information, as returned by the read\_gmt function (see [the import vignette](https://gabrielodom.github.io/pathwayPCA/articles/Importing_Data.html#the-read_gmt-function) for more details). The names of the genes in these pathways *must* match a subset of the genes recorded in the prediction data frame (in the assayData\_df object).
* Regression response: the create\_OmicsReg() function creates an Omics object with class OmicsReg. This object will contain:
  + response: a numeric vector of the response.
  + assayData\_df: a tidy data.frame or tibble of prediction information, as described above.
  + pathwaySet: a list of pathway information, as described above.
* Binary Classification response: the create\_OmicsCateg() function creates an Omics object with class OmicsCateg. In future versions, this function will be able to take in -ary responses and ordered categorical responses, but we only support binary responses for now. This object will contain:
  + response: a factor vector of the response.
  + assayData\_df: a tidy data.frame or tibble of prediction information, as described above.
  + pathwaySet: a list of pathway information, as described above.

In order to create example Omics-class objects, we will consider the overall patient survival time (and corresponding censoring indicator) as our survival response, the event time as our regression response, and event indicator as our binary classification response.

## 3.2 Create a Survival Omics Data Object

Now we are prepared to create our first survival Omics object for later analysis with either AES-PCA or Supervised PCA. Recall that colonSurv\_df has the survival time in the first column, the event indicator in the second column, and the assay expression data in the subsequent columns. Therefore, the four arguments to the create\_Omics function will be:

* assayData\_df: this will be only the *expression columns* of the colonSurv\_df data frame (i.e. all but the first two columns). In R, we can remove the first two columns of the colonSurv\_df data frame by negative subsetting: colonSurv\_df[, -(1:2)].
* pathwaySet\_ls: this will be the colon\_pathwaySet list object. Recall that you can import a .gmt file into a pathwaySet object via the read\_gmt function, or create a pathwaySet list object by hand with the create\_pathwaySet function.
* response: this will be the first two columns of the colonSurv\_df data frame. The survival time stored in the OS\_time column and the event indicator stored in the OS\_event column.
* respType: this will be the word "survival" or an abbreviation of it.

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.

In order to view a summary of the contents of the colon\_OmicsSurv object, you need simply to print it to the R console.

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 Regression and Classification Omics Data Objects

We create regression- and categorical-type Omics data objects identically to survival-type Omics objects. We will use the survival time as our toy regression response and the death indicator as the toy classification response.

colon\_OmicsReg <- create\_Omics(assayData\_df = colonSurv\_df[, -(1:2)],  
 pathwaySet\_ls = colon\_pathwaySet,  
 response = colonSurv\_df$OS\_time,  
 respType = "reg")  
#> Creating object of class OmicsReg.  
colon\_OmicsReg  
#> Formal class 'OmicsReg' [package "pathwayPCA"] with 3 slots  
#> ..@ response : num [1:250] 64.9 59.8 62.4 54.5 46.3 ...  
#> ..@ 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"

colon\_OmicsCateg <- create\_Omics(assayData\_df = colonSurv\_df[, -(1:2)],  
 pathwaySet\_ls = colon\_pathwaySet,  
 response = colonSurv\_df$OS\_event,  
 respType = "categ")  
#> Creating object of class OmicsCateg.  
colon\_OmicsCateg  
#> Formal class 'OmicsCateg' [package "pathwayPCA"] with 3 slots  
#> ..@ response : Factor w/ 2 levels "0","1": 1 1 1 1 2 1 1 1 2 1 ...  
#> ..@ 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"

# 4. Inspecting and Editing Omics-Class Objects

In order to access or edit a specific component of an Omics object, we need to use specific *accessor* functions. These functions are named with the component they access.

## 4.1 Example “Get” Function

The get\* functions access the part of the data object you specify. You can save these objects to their own variables, or simply print them to the screen for inspection. Here we print the assay data frame contained in the colon\_OmicsSurv object to the screen:

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>, ...

This function is rather simple: it shows us what object is stored in the assayData\_df slot of the colon\_OmicsSurv data object. As we should expect, we see all the columns of the colonSurv\_df data frame except for the first two (the survival time and event indicator).

## 4.2 Example “Set” Function

If we needed to edit the assay data frame in the colon\_OmicsSurv object, we can use the “replacement” syntax of the getAssay function. These are the “set” functions, and they use the getSLOT(object) <- value syntax. For example, if we wanted to remove all of the genes except for the first ten from the assay data, we can replace this assay data with a subset of the the original colonSurv\_df data frame. The SLOT shorthand name is Assay, and the replacement value is the first ten gene expression columns (in columns 3 through 12) of the colonSurv\_df data frame: colonSurv\_df[, (3:12)].

getAssay(colon\_OmicsSurv) <- colonSurv\_df[, (3:12)]

Now, when we inspect the colon\_OmicsSurv data object, we see only ten variables measured in the assayData\_df slot, instead of our original 656.

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 10 variables:  
#> ..@ pathwaySet :List of 3  
#> .. ..- attr(\*, "class")= chr [1:2] "pathwaySet" "list"

Before we move on, we should resest the data in the assayData\_df slot to the full data by

getAssay(colon\_OmicsSurv) <- colonSurv\_df[, -(1:2)]

## 4.3 Table of Accessors

Here is a table listing each of the “get” and “set” methods for the Omics class, and which sub-classes they can access or modify.

|  |  |  |
| --- | --- | --- |
| Command | Omics Sub-class | Function |
| getAssay(object) | All | Extract the assayData\_df data frame stored in object. |
| getAssay(object) <- value | All | Set assayData\_df stored in object to value. |
| getPathwaySet(object) | All | Extract the pathwaySet list stored in object. |
| getPathwaySet(object) <- value | All | Set pathwaySet stored in object to value. |
| getEventTime(object) | Surv | Extract the eventTime\_num vector stored in object. |
| getEventTime(object) <- value | Surv | Set eventTime\_num stored in object to value. |
| getEvent(object) | Surv | Extract the eventObserved\_lgl vector stored in object. |
| getEvent(object) <- value | Surv | Set eventObserved\_lgl stored in object to value. |
| getResponse(object) | Reg or Categ | Extract the response vector stored in object. |
| getResponse(object) <- value | Reg or Categ | Set response stored in object to value. |

The response vector accessed or edited with the getResponse method depends on if the object supplied is a “regression” Omics-class object or a “categorical” one. For regression Omics objects, getResponse(object) and getResponse(object) <- value get and set, respectively, the response\_num slot. However, for categorical Omics objects, getResponse(object) and getResponse(object) <- value get and set, respectively, the response\_fact slot. This is because regression objects contain numeric response vectors while categorical objects contain factor response vectors.

## 4.4 Inspect the Updated pathwaySet List

As we mentioned in the [Importing with the read\_gmt Function](https://gabrielodom.github.io/pathwayPCA/articles/Importing_Data.html#importing-with-the-read_gmt-function) subsection of the previous vignette, the pathwaySet object will be modified upon Omics-object creation. Before, this list only had two elements, pathways and TERMS (we skipped importing the “description” field). Now, it has a third element: setsize—the number of genes contained in each pathway.

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 ...

# 5. Review

We now summarize our steps so far. We have

1. Defined the Omics class and three sub-classes: survival, regression, and categorical (and the “parent” class).
2. Created an Omics object for the three sub-classes.
3. Inspected and edited individual elements contained in these objects.

Now we are prepared to analyze our created data objects with either AES-PCA or Supervised PCA. Please read the [Test Pathway Significance](https://gabrielodom.github.io/pathwayPCA/articles/C4-Methods_Walkthrough.html) vignette next.