BioConductor Overviewr

2016-09-28

Contents

Software developed by the BioConductor project http://www.bioconductor.org is provided in the form of R packages. For each package, a vignette illustrating its usage is provided. There are three main package types, software, annotation data, and experimental data (see http://www.bioconductor.org/packages/release/BiocViews.html).

Annotation data are packages that can be used to map mappings from probe identifiers used by the manufacturer to gene-related information, such as Entrez Gene ID, chromosome on which the gene is located, genomic coordinates of the gene, gene symbol, etc.

Installing Bioconductor

Running the `biocLite.R` script will install a subset of the most frequently used Bioconductor packages. From the R prompt,

```
source("http://www.bioconductor.org/biocLite.R")
biocLite()
```

To install additional Bioconductor packages, use `biocLite("package_name")`. Instead of sourcing `biocLite.R` all the time, install `BiocInstaller` package, load it in your `.Rprofile` file using `library(BiocInstaller)`, and have `biocLite()` always available.

Bioconductor basics

Once the base Bioconductor packages have been installed, you can access the vignettes for a specific package as follows:

```
library("Biobase")
openVignette()
```

```
Please select a vignette:

1: Biobase - An introduction to Biobase and ExpressionSets
```

```
2: Biobase - esApply Introduction
3: Biobase - Notes for eSet developers
```

Press "1" to read the first one - it is the foundation of genomics data formats used in R. Or, press "0" to quit.

ExressionSet

Recall that objects in R can be either a `vector`, `factor`, `matrix`, `array`, `data.frame`, `list`, or `ts`. The Biobase package of the Bioconductor project is fundamental, and established new objects that can be used to store gene expression data. An `ExpressionSet` is an object that is a wrapper for the following associated with a microarray study:

- `assayData` - Consists of expression data from a microarray experiment (the expression part hints at the methods used to access it, as we will see below);
- `phenoData` - 'meta-data' describing samples in the experiment;
- `featureData` - annotations and meta-data about the features on the chip or technology used for the experiment;
- `protocolData` - information related to the protocol used for processing each sample (and usually extracted from manufacturer files); and
- `experimentData` - a flexible structure to describe the experiment.

Let's look at one ExpressionSet object:

```
`?`(ExpressionSet)
data("sample.ExpressionSet")
sample.ExpressionSet
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 500 features, 26 samples
  element names: exprs, se.exprs
protocolData: none
phenoData
  sampleNames: A B ... Z (26 total)
  varLabels: sex type score
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: hgu95av2
```

For adventurous, let's peek under the hood to see the slots of the ExpressionSet object. Access them as `sample.ExpressionSet@experimentData`

assayData (gene expression)

First, the most important part of the high-throughput genomic experiment is the matrix of expression values. The underlying structure of an expression matrix in Bioconductor is that the probes (i.e., genes) are in rows while the samples are in columns. Let's read in an example expression matrix and then store it as an ExpressionSet. Once created, `exprs` is the extractor function that is used to access the expression values.

Let's read in a gene expression matrix.

```
expression <- read.csv("data/genedata.csv")

dim(expression)
```

```
[1] 1505   36
```

```
class(expression)
```

```
[1] "data.frame"
```

```
names(expression)[1:4]
```

```
[1] "D.345.Cirrhosis" "D.334.Cirrhosis" "D.520.Cirrhosis" "D.451.Cirrhosis"
```

```
head(expression)[1:3]
```

```
               D.345.Cirrhosis D.334.Cirrhosis D.520.Cirrhosis
AATK_E63_R_01        0.88449182      0.92280276      0.88430897
AATK_P519_R_01       0.75047686      0.73598163      0.71758445
AATK_P709_R_01       0.85082316      0.89804059      0.84933914
ABCA1_E120_R_01      0.95319422      0.91306182      0.94553088
ABCA1_P45_F_01       0.04818880      0.03573306      0.07866697
ABCB4_E429_F_01      0.03291049      0.03847083      0.03623496
```

```
rownames(expression)[1:4]
```

```
[1] "AATK_E63_R_01"   "AATK_P519_R_01"  "AATK_P709_R_01"  "ABCA1_E120_R_01"
```

Having just expression values, we can construct minimal expression set.

```
minimalSet <- ExpressionSet(assayData = as.matrix(expression))
minimalSet
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1505 features, 36 samples
  element names: exprs
protocolData: none
phenoData: none
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
exprs(minimalSet)[1:5, 1:2]
```

```
               D.345.Cirrhosis D.334.Cirrhosis
AATK_E63_R_01         0.8844918      0.92280276
AATK_P519_R_01        0.7504769      0.73598163
AATK_P709_R_01        0.8508232      0.89804059
ABCA1_E120_R_01       0.9531942      0.91306182
ABCA1_P45_F_01        0.0481888      0.03573306
```

```
featureNames(minimalSet)[1:4]
```

```
[1] "AATK_E63_R_01"   "AATK_P519_R_01"  "AATK_P709_R_01"  "ABCA1_E120_R_01"
```

phenoData (sample annotations)

Phenotypic data provides information about the samples, such as normal/abnormal, age, gender, etc. The phenotypic data is represented such that samples appear in rows while the variables appear in columns. Notice that when including phenotypic data in an ExpressionSet, the `row.names` in the `phenoData` must match the sample names in the expression matrix.

```
characteristics <- read.csv("data/phenodata.csv", row.names = 1)
summary(characteristics)
```

```
    Gender           Diagnosis
 Length:36          Length:36
 Class :character   Class :character
 Mode  :character   Mode  :character
```

```
all.equal(rownames(characteristics), names(expression))
```

```
[1] TRUE
```

You will get a warning if there is a mismatch. Before including the `phenoData` into the `ExpressionSet`, we may add some documentation describing information about each covariate (what does the variable name represent, what units the covariates are measure in, etc). This is done by creating a metadata table.

```
metadata <- data.frame(labelDescription = c("Patient gender (Male or Female)",
    "Tissue type (cirrhotic or cirrhotic without HCC)"), row.names = c("Gender",
    "Diagnosis"))
metadata
```

```
                                       labelDescription
Gender                     Patient gender (Male or Female)
Diagnosis Tissue type (cirrhotic or cirrhotic without HCC)
```

```
phenoChar <- new("AnnotatedDataFrame", data = characteristics, varMetadata = metadata)
phenoChar
```

```
An object of class 'AnnotatedDataFrame'
  rowNames: D.345.Cirrhosis D.334.Cirrhosis ...
    D.132.Cirrhosis.non.HCC (36 total)
  varLabels: Gender Diagnosis
  varMetadata: labelDescription
```

```
pData(phenoChar)[1:5, ]
```

```
               Gender Diagnosis
D.345.Cirrhosis   Male Cirrhosis
D.334.Cirrhosis   Male Cirrhosis
D.520.Cirrhosis Female Cirrhosis
D.451.Cirrhosis   Male Cirrhosis
D.473.Cirrhosis   Male Cirrhosis
```

```
pData(phenoChar)$Gender[1:5]
```

```
[1] "Male"   "Male"   "Female" "Male"   "Male"
```

Once a phenoData set is created, it can be accessed using the pData accessor function. Adding phenoData to samples from your ExpressionSet but ensure the phenotypic characteristics stored with it are properly aligned.

```
anotherSet <- ExpressionSet(assayData = as.matrix(expression), phenoData = phenoChar)
anotherSet
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1505 features, 36 samples
  element names: exprs
protocolData: none
phenoData
```

```
  sampleNames: D.345.Cirrhosis D.334.Cirrhosis ...
    D.132.Cirrhosis.non.HCC (36 total)
  varLabels: Gender Diagnosis
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

```
males <- anotherSet[, pData(anotherSet)$Gender == "Male"]
pData(males)$Gender
```

```
 [1] "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male"
[11] "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male"
[21] "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male" "Male"
```

The following code shows what happens when the phenotypic and expres- sion data do not include matching sample names (output suppressed).

```
phony.pheno <- characteristics
rownames(phony.pheno)[1] <- "wrong.sample.name"
phenoPhony <- new("AnnotatedDataFrame", data = phony.pheno, varMetadata = metadata)
phony.pheno[1:3, ]
pData(phenoPhony)[1:3, ]
errorSet <- ExpressionSet(assayData = as.matrix(expression), phenoData = phenoPhony)
```

Annotation (featureData, annotation)

After an analysis, one is usually left with cryptic manufacturer labels of the probes that were significant in your data analysis. To provide meaning to these probes, annotations represent meta data about the probes. The annotation package provides some basic tools for annotation packages.

```
library(annotate)
library("GGHumanMethCancerPanelv1.db")
withannoSet <- ExpressionSet(assayData = as.matrix(expression), phenoData = phenoChar,
    annotation = "GGHumanMethCancerPanelv1.db")
withannoSet
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1505 features, 36 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: D.345.Cirrhosis D.334.Cirrhosis ...
    D.132.Cirrhosis.non.HCC (36 total)
  varLabels: Gender Diagnosis
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation: GGHumanMethCancerPanelv1.db
```

```
featureNames(withannoSet) <- gsub("_01", "", featureNames(withannoSet))
symbol <- getSYMBOL(featureNames(withannoSet), annotation(withannoSet))
entrez <- getEG(featureNames(withannoSet), annotation(withannoSet))
entrez[1:10]
```

```
  AATK_E63_R  AATK_P519_R  AATK_P709_R ABCA1_E120_R  ABCA1_P45_F
```

```
          NA           NA           NA         "19"         "19"
ABCB4_E429_F  ABCB4_P51_F ABCB4_P892_F  ABCC2_E16_R  ABCC2_P88_F
      "5244"       "5244"       "5244"       "1244"       "1244"
```

```
CpG <- mget(featureNames(withannoSet), env = GGHumanMethCancerPanelv1ISCPGISLAND)
CpG[1:5]
```

```
$AATK_E63_R
[1] 0

$AATK_P519_R
[1] 1

$AATK_P709_R
[1] 1

$ABCA1_E120_R
[1] 1

$ABCA1_P45_F
[1] 1
```

experimentData

Data about the experiment can be stored in the `experimentData` slot.

```
experimentData <- new("MIAME", name = "The Author", lab = "Biostat lab", contact = "theauthor@vcu.edu",
    title = "Liver tissue study of cirrhosis vs non-HCC cirrhosis", abstract = "Compare values between
    url = "www.vcu.edu", pubMedIds = "PMC124", other = list(notes = "Further information"))
experimentData
```

```
Experiment data
  Experimenter name: The Author
  Laboratory: Biostat lab
  Contact information: theauthor@vcu.edu
  Title: Liver tissue study of cirrhosis vs non-HCC cirrhosis
  URL: www.vcu.edu
  PMIDs: PMC124

  Abstract: A 7 word abstract is available. Use 'abstract' method.
  notes:
   notes:
      Further information
```

```
abstract(experimentData)
```

```
[1] "Compare values between two liver tissue type"
```

```
notes(experimentData)
```

```
$notes
[1] "Further information"
```

Putting it all together

```
withexpSet <- ExpressionSet(assayData = as.matrix(expression), phenoData = phenoChar,
    annotation = "GGHumanMethCancerPanelv1.db", experimentData = experimentData)
withexpSet
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 1505 features, 36 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: D.345.Cirrhosis D.334.Cirrhosis ...
    D.132.Cirrhosis.non.HCC (36 total)
  varLabels: Gender Diagnosis
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: PMC124
Annotation: GGHumanMethCancerPanelv1.db
```

```
experimentData(withexpSet)
```

```
Experiment data
  Experimenter name: The Author
  Laboratory: Biostat lab
  Contact information: theauthor@vcu.edu
  Title: Liver tissue study of cirrhosis vs non-HCC cirrhosis
  URL: www.vcu.edu
  PMIDs: PMC124

  Abstract: A 7 word abstract is available. Use 'abstract' method.
  notes:
   notes:
      Further information
```

```
abstract(experimentData(withexpSet))
```

```
[1] "Compare values between two liver tissue type"
```

SummarizedExperiment

The next generation of an object that can hold annotated 'omics' data is `SummarizedExperiment`. It is not limited to genes, but instead holds information about genomic regions of interest.

```
library(SummarizedExperiment)
`?`(SummarizedExperiment)
```

We'll look at an example of the `SummarizedExperiment` object in the `parathyroidSE` SummarizedExperiment library. The loaded data is a `SummarizedExperiment`, which summarizes counts of RNA sequencing reads in genes for an experiment on human cell culture. The `SummarizedExperiment` object has 63,000 rows, which are genes, and 27 columns, which are samples, and the matrix, in this case, is called counts. And we have the row names, which are ensemble genes, and metadata about the row data, and metadata about the column data.

```
### SummarizedExperiment
library(parathyroidSE)
# RNA sequencing reads
data(parathyroidGenesSE)
se <- parathyroidGenesSE
se
```

```
class: RangedSummarizedExperiment
dim: 63193 27
metadata(1): MIAME
assays(1): counts
rownames(63193): ENSG00000000003 ENSG00000000005 ... LRG_98 LRG_99
rowData names(0):
colnames: NULL
colData names(8): run experiment ... study sample
```

`assay` function can be used get access to the counts of RNA sequencing reads. `colData` function , the column data, is equivalent to the `pData` on the `ExpressionSet`. Each row in this data frame corresponds to a column in the `SummarizedExperiment`. We can see that there are indeed 27 rows here, which give information about the columns. Each sample in this case is treated with two treatments or control and we can see the number of replicates for each, using the as.numeric function again.

```
# Dimension of the SummarizedExperiment
dim(se)
```

```
[1] 63193    27
```

```
# Get access to the counts of RNA sequencing reads, using assay function.
assay(se)[1:3, 1:3]
```

```
                [,1] [,2] [,3]
ENSG00000000003  792 1064  444
ENSG00000000005    4    1    2
ENSG00000000419  294  282  164
```

```
# Dimensions of this assay is a matrix, which has the same dimensions as the
# SummarizedExperiment.
dim(assay(se))
```

```
[1] 63193    27
```

```
# Get information about samples
colData(se)[1:3, 1:6]
```

```
DataFrame with 3 rows and 6 columns
          run experiment  patient treatment     time submission
  <character>   <factor> <factor>  <factor> <factor>   <factor>
1   SRR479052  SRX140503        1   Control      24h  SRA051611
2   SRR479053  SRX140504        1   Control      48h  SRA051611
3   SRR479054  SRX140505        1       DPN      24h  SRA051611
```

```
# dimension of column data
dim(colData(se))
```

```
[1] 27  8
```

```
# characteristics of the samples
names(colData(se))
```

```
[1] "run"        "experiment" "patient"    "treatment"  "time"
```

```
[6] "submission" "study"      "sample"
```

```
# Get access to treatment column of sample characteristics
colData(se)$treatment
```

```
 [1] Control Control DPN     DPN     OHT     OHT     Control Control
 [9] DPN     DPN     DPN     OHT     OHT     OHT     Control Control
[17] DPN     DPN     OHT     OHT     Control DPN     DPN     DPN
[25] OHT     OHT     OHT
Levels: Control DPN OHT
```

See https://bioconductor.org/packages/devel/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html for the full description.

Diagnostics

```
diagnostics <- devtools::session_info()
platform <- data.frame(diagnostics$platform %>% unlist, stringsAsFactors = FALSE)
colnames(platform) <- c("description")
pander(platform)
```

	description
version	R version 3.3.1 (2016-06-21)
system	x86_64, darwin15.5.0
ui	unknown
language	(EN)
collate	en_US.UTF-8
tz	America/New_York
date	2016-09-28

```
packages <- as.data.frame(diagnostics$packages)
pander(packages[packages$`*` == "*", ])
```

	package	*	version	date	source
1	annotate	*	1.50.0	2016-07-31	Bioconductor
2	AnnotationDbi	*	1.34.4	2016-07-31	Bioconductor
5	Biobase	*	2.32.0	2016-08-11	Bioconductor
6	BiocGenerics	*	0.18.0	2016-07-31	Bioconductor
10	dplyr	*	0.5.0	2016-06-24	CRAN (R 3.3.1)
13	GenomeInfoDb	*	1.8.7	2016-09-05	Bioconductor
14	GenomicRanges	*	1.24.3	2016-09-12	Bioconductor
15	GGHumanMethCancerPanelv1.db	*	1.4.1	2016-08-19	Bioconductor
17	IRanges	*	2.6.1	2016-07-31	Bioconductor
18	knitr	*	1.14	2016-08-13	CRAN (R 3.3.1)
21	org.Hs.eg.db	*	3.3.0	2016-07-31	Bioconductor
22	pander	*	0.6.0	2015-11-23	CRAN (R 3.3.1)
23	parathyroidSE	*	1.10.0	2016-09-25	Bioconductor
28	S4Vectors	*	0.10.3	2016-08-19	Bioconductor
31	SummarizedExperiment	*	1.2.3	2016-07-31	Bioconductor
34	XML	*	3.98-1.4	2016-03-01	CRAN (R 3.3.1)