Introduction to Bioconductor

R for genomics data science

Kevin Rue-Albrecht
University of Oxford (for the African Institute of Biomedical Science and Technology)
2022-11-16 (updated: 2022-11-04)

Lesson goals and objectives

Learning goals

- Describe the Bioconductor project beyond software packages.
- Identify best practices to get help from packages developers and peers.
- Identify classes and methods re-used across Bioconductor packages.

Learning objectives

- Bookmark online websites where help can be found.
- Install core Bioconductor packages and workflow-specific packages.
- Create Bioconductor objects and access their contents.

Prerequisites

- A computer with Microsoft Windows.
- A working installation of R.
- A working installation of RStudio Desktop.

Lessons

- Introduction to base R.
- Object Oriented Programming (OOP).
- Introduction to *ggplot2*.

Set up

- Launch RStudio Desktop on your Windows computer.
- Make a copy of the template notebook for this lesson in your `git` repository.
- Make a symbolic link to your copy of the notebook in the RStudio project for this week.
- Open the notebook and follow along, editing and running the code as needed.

Bioconductor resources and help

Resource	Location
Main website	https://www.bioconductor.org/
Support site	https://support.bioconductor.org/
Courses & Conferences materials	https://www.bioconductor.org/help/course-materials/
YouTube videos	https://www.youtube.com/user/bioconductor
Books	http://www.bioconductor.org/books/release/
Slack workspace	https://bioc-community.herokuapp.com/

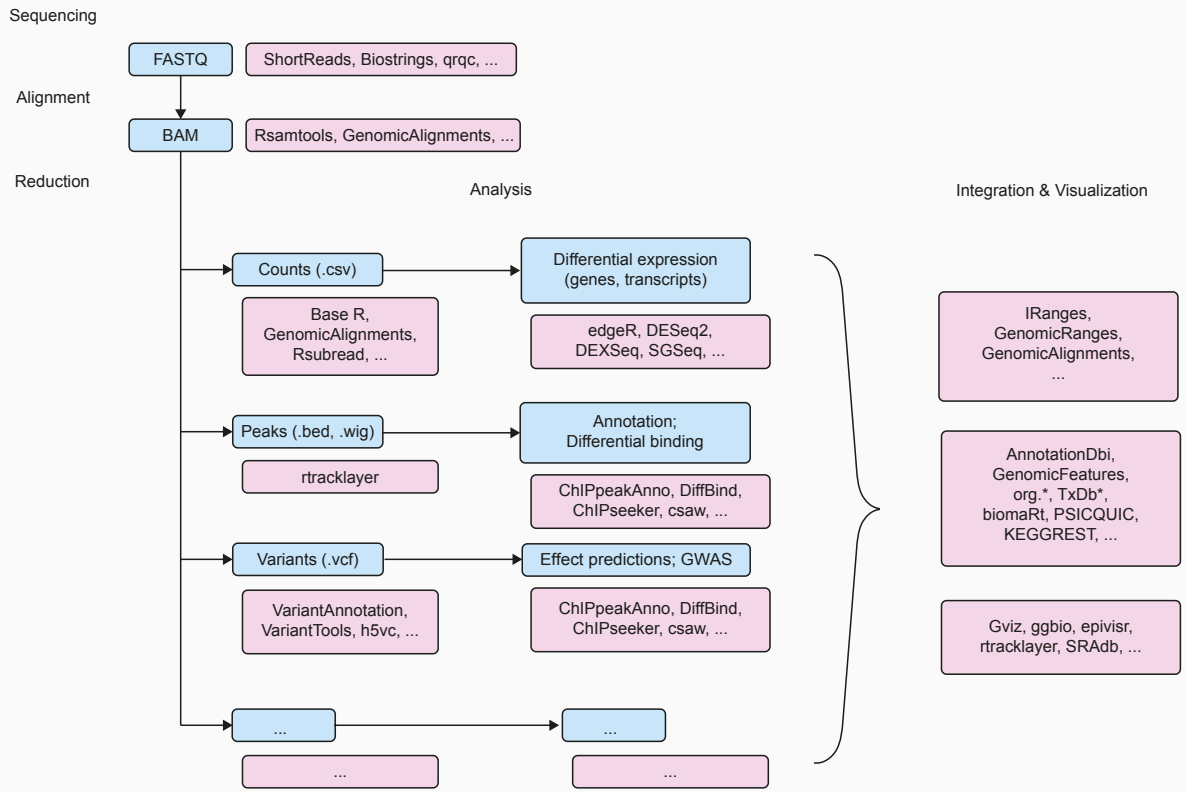
... and more! Can you think of others?

Project objectives

Analysis and comprehension of high-throughput genomic data

- Statistical analysis: large data, technological artifacts, designed experiments; rigorous, robust.
- Comprehension: biological context, visualization, reproducibility.
- High-throughput.
  - Sequencing: RNA-seq, ChIP-seq, variants, copy number, ...
  - Microarrays: gene expression, SNP, ...
  - Flow cytometry, proteomics, images, ...

Bioconductor packages in a workflow



Adapted from: Introduction to R / Bioconductor (2019) by Martin Morgan

Bioconductor release 3.14 (October 27, 2021)

- Contents
  - 2083 software packages
  - 408 experiment data packages
  - 904 annotation packages
  - 29 workflows
- Updates
  - 89 new software packages
  - 13 new data experiment packages
  - 10 new annotation packages
  - 1 new workflow
  - no new books
- Compatibility
  - Bioconductor 3.14 is compatible with R 4.1.1
  - Supported on Linux, 32- and 64-bit Windows, and Intel 64-bit macOS 10.13 (High Sierra) or higher

Source: https://bioconductor.org/news/bioc_3_14_release/

Finding packages and information

- Discover and navigate packages via biocViews.
  - Filter: Software, Annotation, Experiment, Workflow, ...
- Learn more on each package landing page
  - Title, authors, maintainer, short description, citation, installation instructions, ..., download statistics.
- All user-visible functions have help pages, most with runnable examples.
  - Learn by doing on a minimal example; positive control; sanity check.
- Vignettes
  - Narrative documents illustrating how to use the package, integrating code in a workflow, e.g. `browseVignettes("Biobase")`.
- Release cycle
  - Every six months; development takes place on a separate branch of the repository.

Exercise

Exploring software packages

- Visit the listing of packages on the Bioconductor biocViews web page.
- Use the `Autocomplete biocViews search` box in the upper left to identify packages that have been tagged for RNA sequencing analysis.
- Explore other analysis like ChIP-seq, epigenetics, variant annotation, proteomics, single-cell genomics, etc.
- Explore the graph of software packages by expanding and contracting individual terms.
- In the RNA-seq category, find out which of `DESeq2` and `edgeR` is more popular, and go to their landing page.
- Briefly explore the vignette and reference manual links.

When would you consult the vignette? When would the reference manual be helpful?

Exercise

Exploring annotation packages

- Visit the listing of packages on the Bioconductor biocViews web page.
- Click on the `AnnotationData` top-level term.
- Search, using the box on the right-hand side, for annotation packages that start with the following letters to get a sense of the packages and organisms available.
  - `org.` : symbol mapping
  - `TxDb.` and `EnsDb.` : gene models
  - `BSgenome.` : reference genomes

Exploring workflow packages

Workflow packages are meant to provide a comprehensive introduction to workflows that require several different packages. These can be quite extensive documents, providing a very rich source of information.

- Briefly explore the 'Simple Single Cell' workflow (or other workflow relevant to your domain of interest) to get a sense of the material the workflow covers.

The Bioconductor philosophy

- Common data structures
  - e.g., `DNAStringSet`, `GRanges`, `GAlignments`, `SummarizedExperiment`, `TxDb`
  - High standards of software engineering; Core team.
  - Reduce redundancy; Promotes interoperability.
  - ..., but community adoption can take time.
- Release cycle every 6 months (April, October)
  - Packages on the *release* branch are stable for at least 6 months (except for exceptional bug fixes). Active development takes place on the *devel* branch.
  - ..., but new features can take up to 6 months to be released (*devel* $\rightarrow$ *release X.Y*).
- New packages are thoroughly reviewed by members of the core team and volunteers
  - Packages are submitted as issues on Bioconductor/Contributions
  - Review and fixes can take time, but accepted packages are generally high quality!

Bioconductor uses the S4 class system

- Best practices: classes and methods implemented in a dedicated package.
- Formal definition: parent class (optional), new slot names and types.
- Internal validity checking method.
- Methods and generics implemented outside the class definition.
    - "What can I do with this object?"

```
setClass("Person",
  representation(
    name = "character",
    age = "numeric")
  )
```

Source: http://adv-r.had.co.nz/S4.html

```
kevin <- Person(name = "Kevin", age = 21)
kevin
```

```
## An object of class "Person"
## Slot "name":
## [1] "Kevin"
##
## Slot "age":
## [1] 21
```

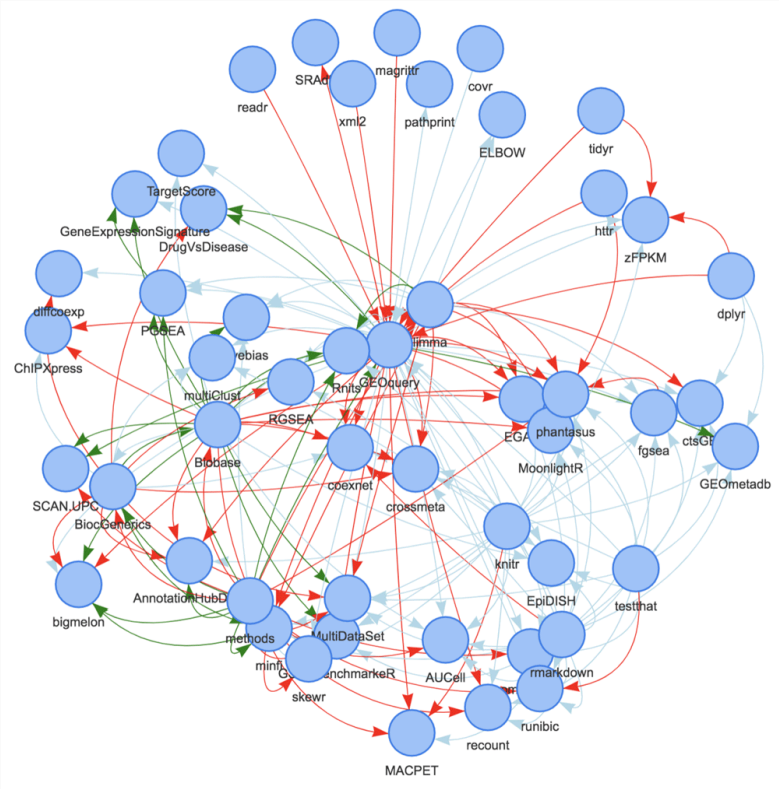
```
kevin@age # don't do that
```

```
## [1] 21
```

```
get_age(kevin) # do that (if possible)
```

```
## [1] 21
```

Bioconductor packages depend on each other



Data visualization of dependencies between all packages within one degree of the GEOquery package.

Links are colored based on type (Suggests [light blue], Depends [green], and Imports [red]) and arrows point to the "dependent" package.

BiocPkgTools (Su, Carey, Shepherd, Ritchie, Morgan, and Davis, 2019)

BiocManager

Prior to the course, we have provided you with an R script that installed all the packages required for the course. As a result, you do not need install any package during this course.

Briefly, what the R script does essentially comes down to:

```
install.packages("BiocManager")
BiocManager::install(c("Biostrings", "S4Vectors"))
```

The package *BiocManager* is the only official Bioconductor package on CRAN. It provides functions to install, update, and more generally manage your library of Bioconductor (and CRAN) packages.

The function `BiocManager::install()` can install packages from Bioconductor, CRAN, and even GitHub. Here, we demonstrate it with a vector of two package names. However, it can be used to install a single package name as a character value.

- Run the function `BiocManager::valid()`. What does it do?

Updating Bioconductor packages

```
BiocManager::valid()
```

```
Bioconductor version '3.13'

  * 6 packages out-of-date
  * 0 packages too new

create a valid installation with

  BiocManager::install(c(
    "e1071", "htmlwidgets", "httpuv", "knitr", "parallelly", "RcppAnnoy"
  ), update = TRUE, ask = FALSE)

more details: BiocManager::valid()$too_new, BiocManager::valid()$out_of_date

Warning message:
6 packages out-of-date; 0 packages too new
```

Exercise

Use Bioconductor packages and help pages

- Load the *Biostrings* package.
- The package provides the object `DNA_ALPHABET`. Print the object in the console and use the help page to explain the contents of the object.
- The package provides the function `IlluminaQuality()`. Run some code from the "Examples" section of the help page, and describe what the function does.
- Access the package vignette(s). How many vignette does the package include?
- Import sequences in the file `TruSeq3-PE-2.fa`. What is the class of the object?

BSgenome packages

Bioconductor provide genome sequences for a range of model organisms and their incremental versions over time.

- Software (2041)
- AnnotationData (977)
  - ChipManufacturer (396)
  - ChipName (196)
  - CustomArray (2)
  - CustomDBSchema (8)
  - FunctionalAnnotation (36)
  - Organism (639)
  - PackageType (686)
    - AnnotationHub (17)
    - BSgenome (103)
    - cdf (118)
    - ChipDb (175)
    - db0 (19)
    - EuPathDB (1)
    - FRMA (11)
    - MeSHDb (74)

- The *BSgenome* package provides core functionality.
- Other package names start with `"BSgenome."`
  - e.g. `BSgenome.Hsapiens.UCSC.hg19`

Exercise

BSgenome packages

- Load the package *BSgenome*.
- Use the function `BSgenome::available.genomes()`. What does it do?
- Load the package `BSgenome.Hsapiens.UCSC.hg38.masked`.
- Assign the genome object provided in the package to a new object called `genome`.

An E-box (enhancer box) is a DNA response element found in some eukaryotes that acts as a protein-binding site and has been found to regulate gene expression in neurons, muscles, and other tissues.

The E-box motif is `"CANNTG"`.

- Extract the sequence of chromosome Y from the `genome` object above.
- Find out the appropriate method in the *Biostrings* and count the number of E-box motifs present on chromosome Y.

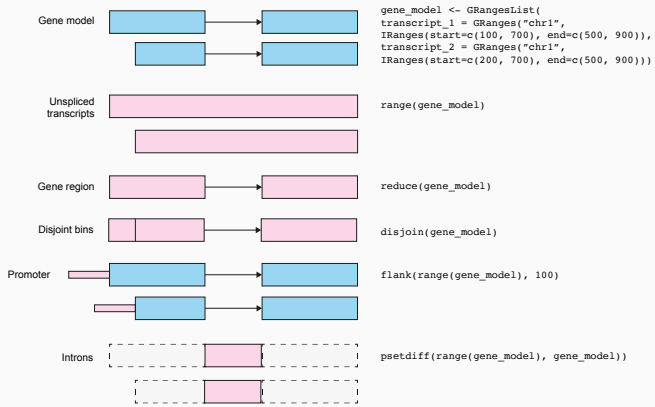
Genomic ranges

Coordinates on a genomic scale.

Information

- Chromosome, contig
- Start
- End
- Strand (optional)
- ... and other metadata

Ranges and operations



Creating genomic ranges

```r
library(GenomicRanges)
demo_granges <- GRanges(
  seqnames = c("chr1", "chr2"),
  ranges = IRanges(
    start = c(10, 20),
    end = c(25, 35)),
  metadata1 = c("control", "target"),
  metadata2 = c(1, 2))
demo_granges
```

```
## GRanges object with 2 ranges and 2 metadata columns:
##       seqnames    ranges strand |   metadata1 metadata2
##          <Rle> <IRanges>  <Rle> | <character> <numeric>
##   [1]     chr1     10-25      * |     control         1
##   [2]     chr2     20-35      * |      target         2
##   -------
##   seqinfo: 2 sequences from an unspecified genome; no seqlengths
```

What genomics file format does this remind you of?

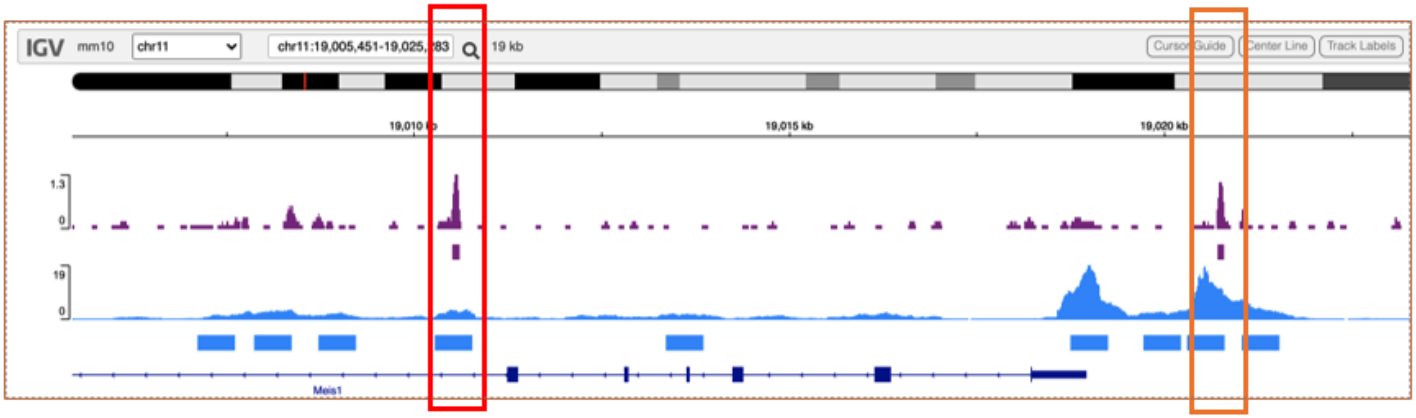
BED file format

```
chr7  127471196  127472363  Pos1  0  +  127471196  127472363  255,0,0
chr7  127472363  127473530  Pos2  0  +  127472363  127473530  255,0,0
chr7  127473530  127474697  Pos3  0  +  127473530  127474697  255,0,0
chr7  127474697  127475864  Pos4  0  +  127474697  127475864  255,0,0
chr7  127475864  127477031  Neg1  0  -  127475864  127477031  0,0,255
chr7  127477031  127478198  Neg2  0  -  127477031  127478198  0,0,255
chr7  127478198  127479365  Neg3  0  -  127478198  127479365  0,0,255
chr7  127479365  127480532  Pos5  0  +  127479365  127480532  255,0,0
chr7  127480532  127481699  Neg4  0  -  127480532  127481699  0,0,255
```

- **chrom** - name of the chromosome or scaffold.
- **chromStart** - Start position (first base is 0).
- **chromEnd** - End position.
- **name** - Label to be displayed for the feature.
- **score** - A score between 0 and 1000.
- **strand** - defined as + (forward) or - (reverse).
- **thickStart** - start coordinate to draw a solid rectangle.
- **thickEnd** - end coordinate to draw a solid rectangle.
- **itemRgb** - an RGB colour value (e.g. 0,0,255).
- **blockCount** - the number of exons within the feature.
- **blockSizes** - the size of these sub-elements.
- **blockStarts** - the start coordinate of each sub-element

ensembl.org

Genomic ranges in action



ATAC-Seq bedfile (file A)

chr11	19007120	19007620
chr11	19007874	19008374
chr11	19008731	19009231
chr11	19010281	19010781
chr11	19013360	19013860
chr11	19018750	19019250
chr11	19019722	19020222
chr11	19020303	19020803
chr11	19021033	19021533
chr11	19026581	19027081
chr11	19027171	19027671
chr11	19091279	19091779
chr11	19107100	19107600
chr11	19111900	19112400
chr11	19120454	19120954

Atoh1-Chipseq bedfile (file B)

chr11	19010514	19010612
chr11	19020709	19020796
chr11	19204142	19204308
chr11	19204579	19204671
chr11	19423843	19423946
chr11	19521861	19521953
chr11	19560612	19560813
chr11	19601766	19601836
chr11	19769818	19769953
chr11	19816862	19816932

ATAC Sites that overlap with Atoh1 Sites

chr11	19010281	19010781
chr11	19020303	19020803

Find motifs etc ...

Credits: Charlotte George

Accessing the contents of GRanges objects

The functions `seqnames()`, `start()`, and `end()` access the components of the same name.

```
seqnames(demo_granges)
```

```
## factor-Rle of length 2 with 2 runs
##   Lengths:    1    1
##   Values : chr1 chr2
## Levels(2): chr1 chr2
```

```
as.character(seqnames(demo_granges))
```

```
## [1] "chr1" "chr2"
```

```
start(demo_granges)
```

```
## [1] 10 20
```

```
end(demo_granges)
```

```
## [1] 25 35
```

Chromosome names are conceptually similar to factors; a `GRanges` object often contains many ranges on each chromosome. The function `seqlevels()` returns the list of unique chromosome names in the object (including levels that may not be present in a subset).

```
seqlevels(demo_granges)
```

```
## [1] "chr1" "chr2"
```

Metadata columns

`GRanges` is one of many classes that derive from the Bioconductor virtual class `Vector`.

- Virtual classes cannot be instantiated into objects.
- They are used to define slots and methods that be will passed down the inheritance path to all classes that derive from this virtual class.

The virtual class `Vector` defines two slots:

- `metadata`, to store metadata related to the entire object.
- `elementMetadata`, to store metadata for each element in the vector; accessed using `mcols()`.

```
mcols(demo_granges)
```

```
## DataFrame with 2 rows and 2 columns
##     metadata1 metadata2
##   <character> <numeric>
## 1     control         1
## 2      target         2
```

```
demo_granges$metadata1
```

```
## [1] "control" "target"
```

```
metadata(demo_granges)
```

```
## list()
```

Bioconductor has its own DataFrame class

A Bioconductor `DataFrame` is constructed like a base R `data.frame`.

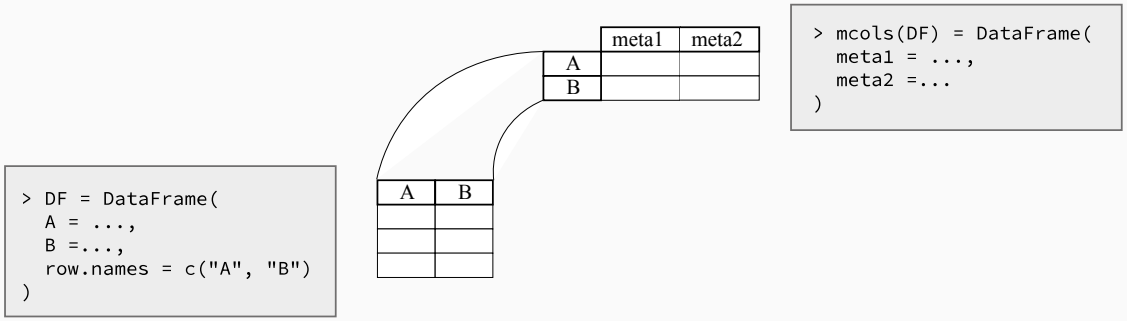
```
library(S4Vectors)
demo_DataFrame <- DataFrame(
    column1 = c("A", "B"),
    column2 = c(1, 2)
)
demo_DataFrame
```

```
## DataFrame with 2 rows and 2 columns
##       column1   column2
##   <character> <numeric>
## 1           A         1
## 2           B         2
```

However, a `DataFrame` can have additional metadata columns.

```
mcols(demo_DataFrame) <- DataFrame(
  column_type = c("character", "numeric")
)
mcols(demo_DataFrame)
```

```
## DataFrame with 2 rows and 1 column
##         column_type
##         <character>
## column1   character
## column2     numeric
```



rtracklayer

- Software (2041)
  - AssayDomain (819)
  - BiologicalQuestion (866)
  - Infrastructure (480)
  - ResearchField (953)
  - StatisticalMethod (762)
  - Technology (1301)
  - WorkflowStep (1121)
    - Alignment (82)
    - Annotation (127)
    - BatchEffect (51)
    - ExperimentalDesign (27)
    - GenomeBrowsers (2)
    - MultipleComparison (168)
    - Normalization (138)
    - Pathways (189)
    - Preprocessing (237)

- The package *rtracklayer* was developed as an extensible framework for interacting with multiple genome browsers
  - e.g., UCSC
- To that end, it also provides functionality for manipulating annotation tracks in various formats (currently and 2bit built-in)
  - e.g., GFF, BED, bedGraph, BigWig

Exercise

rtracklayer

- Load the *rtracklayer* package.
- Import the file `Homo_sapiens.GRCh38.104.gtf.gz` using the function `rtracklayer::import()`. Assign value to an object called `gtf_data`.
- What are the metadata columns available in the object?
- Use the function `subset()` to filter the annotations for the gene `ACTB`.
- How many transcripts do the annotations describe for that gene?
- Subset the annotations to exons, and use the `split()` function to separate exons from different transcripts. What is the class of the object that is returned?

Annotations - An example

New | Count | Results | URL | XML | Perl | Help

Dataset
Human genes (GRCh38.p10)

Filters
HGNC symbol(s) [e.g. A1BG]: [ID-list specified]

Attributes
Gene stable ID
Chromosome/scaffold name
Gene start (bp)
Gene end (bp)
Gene name
HGNC symbol

Dataset
[None Selected]

Export all results to File TSV Unique results only Go

Email notification to

View 20 rows as HTML Unique results only

Gene stable ID	Chromosome/scaffold name	Gene start (bp)	Gene end (bp)	Gene name	HGNC symbol
ENSG00000141736	17	39687914	39730426	ERBB2	ERBB2
ENSG00000141738	17	39737927	39747291	GRB7	GRB7
ENSG00000161405	17	39757715	39864188	IKZF3	IKZF3
ENSG00000141741	17	39728496	39730787	MIEN1	MIEN1
ENSG00000265178	17	39726495	39726561	MIR4728	MIR4728
ENSG00000171532	17	39603536	39609777	NEUROD2	NEUROD2
ENSG00000161395	17	39671122	39696797	PGAP3	PGAP3
ENSG00000141744	17	39667981	39670475	PNMT	PNMT
ENSG00000131771	17	39626740	39636626	PPP1R1B	PPP1R1B
ENSG00000131748	17	39637065	39663484	STARD3	STARD3
ENSG00000173991	17	39664187	39666555	TCAP	TCAP

Datasets -> Filters (filtering and inputs) -> Attributes (desired output) -> Results

BioMart tutorial | YouTube | YouKu

Annotation packages and biomaRt

Packages dedicated to query annotations exist in the `Software` and `Annotation` categories of biocViews.

- The *biomaRt* package provides an interface to the Ensembl BioMart data repository.
- A collection of annotation packages provide most of that information for several model organisms.

Bioconductor version 3.13 (Release)

Autocomplete biocViews search:

Apis_mellifera (4)
Arabidopsis_thaliana (13)
Asparagus_officinalis (1)
Bacillus_subtilis (2)
Bos_taurus (15)
Caenorhabditis_elegans (12)
Callithrix_jacchus (1)
Canis_familiaris (13)
Chlamydomonas_reinhardtii (1)
Cicer_arietinum (1)
Danio_rerio (15)
Drosophila_melanogaster (16)
Drosophila_virilis (1)
Escherichia_coli (12)
Gallus_gallus (14)
Gasterosteus_aculeatus (2)
Homo_sapiens (242)

Packages found under Homo_sapiens:

Rank based on number of downloads: lower numbers are more frequently downloaded.

Show All entries Search table:

Package	Maintainer	Title
org.Hs.eg.db	Bioconductor Package Maintainer	Genome wide annotation for Human
TxDb.Hsapiens.UCSC.hg19.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
BSgenome.Hsapiens.UCSC.hg19	Bioconductor Package Maintainer	Full genome sequences for Homo sapiens (UCSC version hg19, based on GRCh37.p13)
BSgenome.Hsapiens.UCSC.hg38	Bioconductor Package Maintainer	Full genome sequences for Homo sapiens (UCSC version hg38, based on GRCh38.p12)
TxDb.Hsapiens.UCSC.hg38.knownGene	Bioconductor Package Maintainer	Annotation package for TxDb object(s)
hgu133plus2.db	Bioconductor Package Maintainer	Affymetrix Affymetrix HG-U133_Plus_2 Array annotation data (chip hgu133plus2)

Annotation packages

Each annotation package contains a database of information.

The `OrgDb` family of packages provide annotations that link several types of identifiers for genes, proteins, and pathways (e.g. KEGG, Gene Ontology).

- Load the package *org.Hs.eg.db*.

Each `OrgDb` package contains an object of the same name as the package.

- What is the class of the object?
- Use the `columns()` method to discover which sorts of annotations can be extracted from the object.
- Use the `keytypes()` method to discover which columns can be used to query information.
- Use the `keys()` method to get the first six gene symbols in the database. Store as `human_symbols`.
- Use the `AnnotationDbi::select()` method to fetch the Entrez and Ensembl identifiers for those gene symbols.
  - What happens if you do not specify `AnnotationDbi`?
- Use the `mapIds()` method to get the Ensembl identifier for those gene symbols.

Exercise

biomaRt

- Load the package *biomaRt*.
- Use the `listMarts()` function to list the BioMart databases to which the *biomaRt* package can connect.
- Use the `listEnsemblArchives()` function to lists the available archived versions of Ensembl.
- Use the `useMart()` function to create an object named `mart` using the `"ENSEMBL_MART_ENSEMBL"` BioMart database and set the host option to `"https://may2021.archive.ensembl.org"`. Why would one do that in practice?
- Use the `listDatasets()` function to list the datasets available in the selected BioMart database. What is the identifier of the database that contains information for *Homo sapiens*?
- Use again the `useMart()` function to replace the `mart` object by a new one that points to information for *Homo sapiens*.
- Use the `listAttributes()` function to list the fields of information available in the dataset.
- Use the `getBM()` function to fetch the chromosome, start and end positions, and strand for the gene symbols that you stored as `human_symbols`.

EnsDb packages

The `EnsDb` family of packages provide annotations that encapsulate individual versions of the Ensembl annotations in Bioconductor packages.

- Once the package is installed, annotations are stored locally, without the need for internet.
- The series of packages seems to have ended at Ensembl version 86 (!)
- Bioconductor annotation packages share the `BiMap` class, meaning that functions like `columns()`, `keytypes()`, and `mapIds()` work in the same way for the different types of annotation packages.

```
library(EnsDb.Hsapiens.v86)
class(EnsDb.Hsapiens.v86)
```

```
## [1] "EnsDb"
## attr(,"package")
## [1] "ensembldb"
```

```
columns(EnsDb.Hsapiens.v86)
```

```
##  [1] "ENTREZID"            "EXONID"              "EXONIDX"             "EXONSEQEND"          "EXONSEQSTART"
##  [6] "GENEBIOTYPE"         "GENEID"              "GENENAME"            "GENESEQEND"          "GENESEQSTART"
## [11] "INTERPROACCESSION"   "ISCIRCULAR"          "PROTDOMEND"          "PROTDOMSTART"        "PROTEINDOMAINID"
## [16] "PROTEINDOMAINSOURCE" "PROTEINID"           "PROTEINSEQUENCE"     "SEQCOORDSYSTEM"      "SEQLENGTH"
## [21] "SEQNAME"             "SEQSTRAND"           "SYMBOL"              "TXBIOTYPE"           "TXCDSSEQEND"
## [26] "TXCDSSEQSTART"       "TXID"                "TXNAME"              "TXSEQEND"            "TXSEQSTART"
## [31] "UNIPROTDB"           "UNIPROTID"           "UNIPROTMAPPINGTYPE"
```

TxDb packages

The `TxDb` family of packages provide annotations that encapsulate individual versions of the annotation databases generated from UCSC in Bioconductor packages.

```
library(TxDb.Hsapiens.UCSC.hg38.knownGene)
class(TxDb.Hsapiens.UCSC.hg38.knownGene)
```

```
## [1] "TxDb"
## attr(,"package")
## [1] "GenomicFeatures"
```

```
columns(TxDb.Hsapiens.UCSC.hg38.knownGene)
```

```
##  [1] "CDSCHROM"   "CDSEND"     "CDSID"      "CDSNAME"    "CDSPHASE"   "CDSSTART"   "CDSSTRAND"  "EXONCHROM"  "EXONEND"
## [10] "EXONID"     "EXONNAME"   "EXONRANK"   "EXONSTART"  "EXONSTRAND" "GENEID"     "TXCHROM"    "TXEND"      "TXID"
## [19] "TXNAME"     "TXSTART"    "TXSTRAND"   "TXTYPE"
```

```
keytypes(TxDb.Hsapiens.UCSC.hg38.knownGene)
```

```
## [1] "CDSID"    "CDSNAME"  "EXONID"   "EXONNAME" "GENEID"   "TXID"     "TXNAME"
```

```
keys(TxDb.Hsapiens.UCSC.hg38.knownGene, "GENEID") %>% head()
```

```
## [1] "1"         "10"        "100"       "1000"      "10000"     "100008586"
```

Annotations - Summary

biomaRt

Interface to the Ensembl BiomaRt database.

OrgDb

Family of packages that provide mapping between various types of gene identifiers and pathway information.

EnsDb

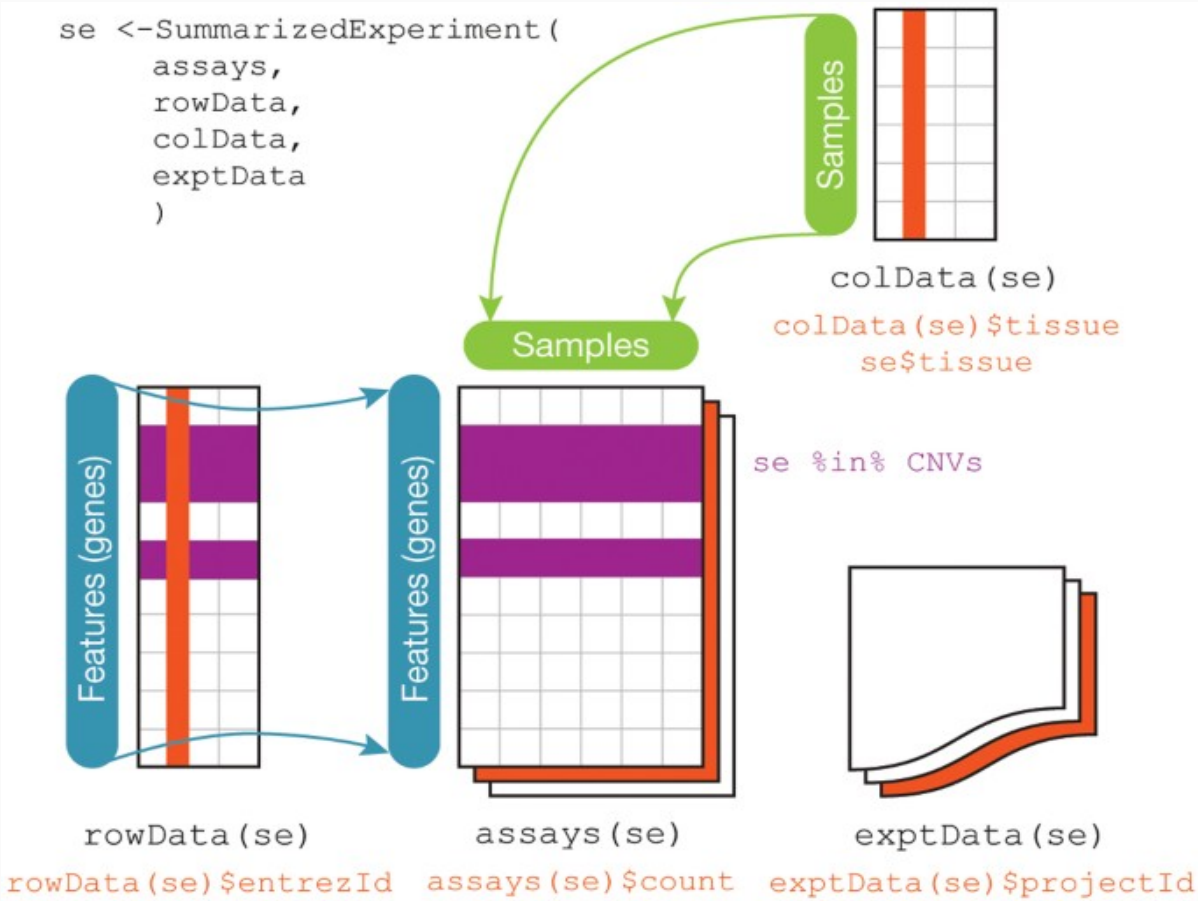
Family of packages that each provide one release of gene annotations from Ensembl.

TxDb

Family of packages that each provide one release of transcript annotations from UCSC.

Do you know any other?

The Bioconductor SummarizedExperiment



Source: https://www.nature.com/articles/nmeth.3252 (Figure 2)

Exercise

Creating a SummarizedExperiment object

```r
library(SummarizedExperiment)
demo_count_matrix <- matrix(
  data = c(1, 2, 3, 4, 5, 6, 7, 8), nrow = 4, ncol = 2,
  dimnames = list(
    paste0("gene", 1:4),
    paste0("sample", 1:2)
  ))
demo_rowdata <- DataFrame(
  symbol = head(letters, 4),
  length = c(1234, 5678)
  )
demo_coldata <- DataFrame(
  condition = c("control", "treated"),
  age = c(18, 20)
  )
demo_se <- SummarizedExperiment(
  assays = list(
    counts = demo_count_matrix),
  rowData = demo_rowdata,
  colData = demo_coldata)
```

What does the code above do?

```r
demo_se
```

```
## class: SummarizedExperiment
## dim: 4 2
## metadata(0):
## assays(1): counts
## rownames(4): gene1 gene2 gene3 gene4
## rowData names(2): symbol length
## colnames(2): sample1 sample2
## colData names(2): condition age
```

Use the methods `assays()`, `assayNames()` `assay()`, `colData()`, `rowData()`, `mcols()` and `metadata()` on the object `demo_se`.

What do those functions do?

Exercise

Assembling a SummarizedExperiment object

- Import the matrix in the file `counts.csv`. Call it `assay_counts`.
- Import the data frames in the files `sample_metadata.csv` and `gene_metadata.csv`. Call them `sample_metadata` and `gene_metadata`.
- Use the function `SummarizedExperiment()` to combine those three objects into a single `SummarizedExperiment` object. Call it `demo_se`.
    - Assign the matrix to an assay named `counts`.

Inheritance and coercion

The `as()` function can be used to convert an object between classes that are related through inheritance. This is called *coercing* the object to another class.

For instance:

```
demo_dataframe <- data.frame(A = c(1, 2), B = c(3, 4))
demo_dataframe
```

```
##   A B
## 1 1 3
## 2 2 4
```

```
demo_DataFrame <- as(demo_dataframe, "DataFrame")
demo_DataFrame
```

```
## DataFrame with 2 rows and 2 columns
##           A         B
##   <numeric> <numeric>
## 1         1         3
## 2         2         4
```

- If the new class is simpler, some information may be lost (e.g., coercing to a parent class).
- If the new class is more complex, new components may be filled with automatically computed or default values. (e.g., coercing to child class).

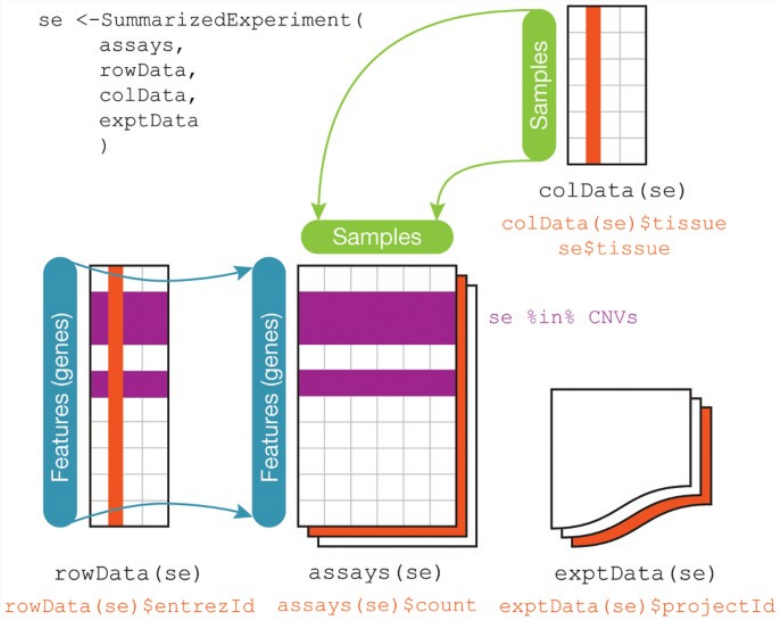
Exercise

SummarizedExperiment extensions

- Load the package *DESeq2*.
- Convert your earlier `demo_se` to a `DESeqDataSet` object, using the `as()` function. Call the object `demo_deseq`.
- What are the slots of information present in the `DESeqDataSet` object? Use the function `slotNames()`.
- Which slots are new compared to the `SummarizedExperiment` object?
- Which components of the `DESeqDataSet` object can you access using accessor functions (e.g., `assays()`) ? Do all of those accessors work on the original `demo_se` object?

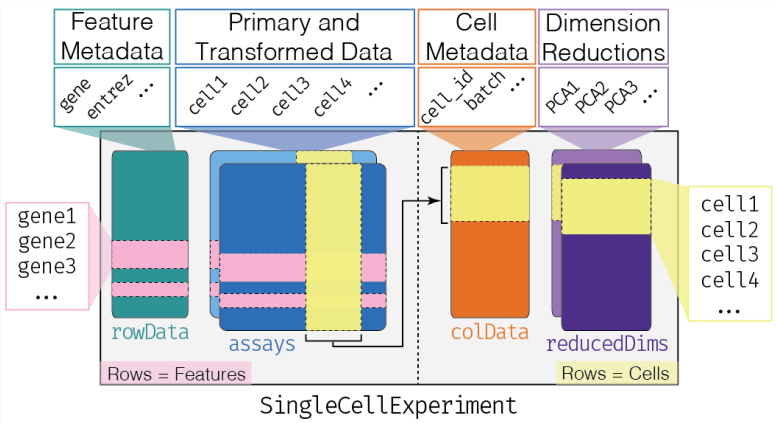
SingleCellExperiment extends SummarizedExperiment

SummarizedExperiment.



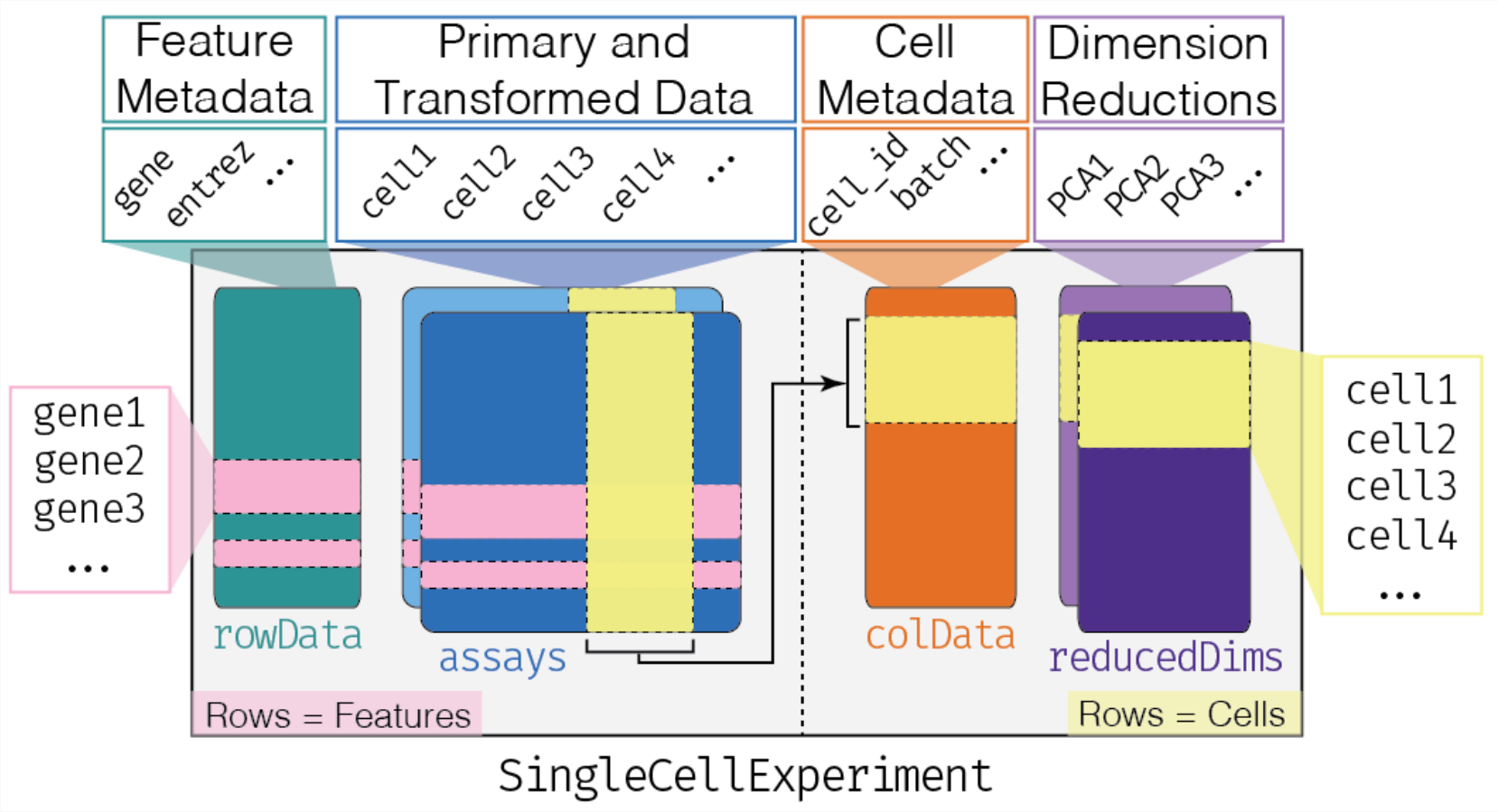
Source: https://www.nature.com/articles/nmeth.3252 (Figure 2)

SingleCellExperiment



Source: https://osca.bioconductor.org/data-infrastructure.html

SingleCellExperiment



Source: https://osca.bioconductor.org/data-infrastructure.html

Exercise

SingleCellExperiment

The class name `SingleCellExperiment` is only a name. While developed with single-cell genomics in mind, it is not limited to that use. Remember that classes are simply defined by their slots and methods. If the slots and methods are suitable to store and process another type of data, then go for it!

- Load the package *SingleCellExperiment*.
- Convert your earlier `demo_se` to a `SingleCellExperiment` object, using the `as()` function. Call the object `demo_sce`.
- Display the object. What additional information can the `SingleCellExperiment` class store, relative to the `SummarizedExperiment` class?
- Use the methods `reducedDimNames()`, `reducedDims()`, and `reducedDim()` on the object `demo_sce`. What do they do?
- **Bonus point:** Use the method `reducedDim()` to store a new dimensionality reduction matrix called `"PCA"` in the object. Display and inspect the updated object.

The ExperimentHub package and resources

The *ExperimentHub* package provides a portal to a large collection of preprocessed datasets available directly as Bioconductor objects.

The package can be used to query, search, and filter the list of available datasets, before downloading and importing those of interest into your R session.

```r
library(ExperimentHub)
ehub <- ExperimentHub()
query(ehub, c("SingleCellExperiment"))
```

```
## ExperimentHub with 69 records
## # snapshotDate(): 2022-04-26
## # $dataprovider: Robinson group (UZH), 10x Genomics, Zheng et al (2017), SRA...
## # $species: Homo sapiens, Mus musculus
## # $rdataclass: SingleCellExperiment
## # additional mcols(): taxonomyid, genome, description,
## #   coordinate_1_based, maintainer, rdatadateadded, preparerclass, tags,
## #   rdatapath, sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["EH1433"]]'
##
##            title
##   EH1433 | GEO accession data GSE71585 as a SingleCellExperiment
##   EH1500 | sce_full_Koh
##   EH1501 | sce_filteredExpr10_Koh
##   EH1502 | sce_filteredHVG10_Koh
##   EH1503 | sce_filteredM3Drop10_Koh
##   ...      ...
##   EH6748 | hipsc
##   EH6749 | cellmix
##   EH6750 | pbmc
##   EH6751 | hca_bm
```

```r
eh1433 <- ehub[["EH1433"]]
eh1433
```

```
## class: SingleCellExperiment
## dim: 24150 1809
## metadata(0):
## assays(3): counts tpm rpkm
## rownames(24150): 0610005C13Rik 0610007C21Rik ... ERCC-00171 tdTomato
## rowData names(0):
## colnames(1809): Calb2_tdTpositive_cell_1 Calb2_tdTpositive_cell_2 ...
##   Rbp4_CTX_250ng_2 Trib2_CTX_250ng_1
## colData names(12): mouse_line cre_driver_1 ... secondary_type
##   aibs_vignette_id
## reducedDimNames(0):
## mainExpName: NULL
## altExpNames(0):
```

Description of a dataset in the ExperimentHub

A single pair of square brackets `[]` shows information about a given identifier, while a double pair `[[]]` downloads the dataset and imports the object.

```
ehub["EH1433"]
```

```
## ExperimentHub with 1 record
## # snapshotDate(): 2022-04-26
## # names(): EH1433
## # package(): allenpvc
## # $dataprovider: GEO
## # $species: Mus musculus
## # $rdataclass: SingleCellExperiment
## # $rdatadateadded: 2018-05-02
## # $title: GEO accession data GSE71585 as a SingleCellExperiment
## # $description: Celular taxonomy of the primary visual cortex in adult mice ...
## # $taxonomyid: 10090
## # $genome: mm10
## # $sourcetype: CSV
## # $sourceurl: https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE71585
## # $sourcesize: NA
## # $tags: c("ExperimentData", "ExpressionData", "SequencingData",
## #   "RNASeqData")
## # retrieve record with 'object[["EH1433"]]'
```

Exercise

Interactive visualisation using iSEE

is a package that leverages the reliable stability of the `SummarizedExperiment` data structure - and its extensions - to produce an interactive graphical user interface (GUI) for visualisation of the object contents.

Load the package *iSEE*.

Use the `iSEE()` function on the `SingleCellExperiment` object `eh1433`.

Subset the object `eh1433` to remove cells where the sum of counts is `NA`.

```
eh1433 <- eh1433[, !is.na(colSums(assay(eh1433)))]
```

Use the packages *scuttle* and *scater* to compute and add a PCA result to the object `eh1433`

```
library(scuttle)
library(scater)
eh1433 <- logNormCounts(eh1433)
eh1433 <- runPCA(eh1433)
```

Launch the `iSEE` GUI again. What changed?

More popular Bioconductor packages

Gviz — Genome browser visualisations.

affy — Affymetrix microarrays.

ChIPseeker — Annotation and visualisation.

bsseq — DNA methylation.

Spectra — Mass spectrometry.

VariantAnnotation — DNA variation.

SpatialExperiment — Spatial transcriptomics.

tt — Tidy transcriptomics.

Final advice

Choosing a package

With so much choice, it can be difficult to decide which package to try first.

- Go by popularity (e.g., biocViews, download stats, word of mouth)
  - More users $\rightarrow$ more chance to find help, online and around you.
- Read and manually run code in the package vignette(s).
  - A good package that makes sense to you is better than an expert package that feels obscure.

Getting help

- https://support.bioconductor.org/
  - Public and searchable: asked once, answered once (avoid emails!).
  - MRE: Minimal reproducible example (e.g., use demo object in help page).

Further reading

- CSAMA 2019 programme
- Bioconductor course materials
- Introduction to R / Bioconductor (2019), by Martin Morgan
- Introduction to R / Bioconductor (2016), by Martin Morgan

BiocManager repositories

Bioconductor packages can be installed and managed using base R function such as `install.packages()` with a minimum additional setup.

- Run `options("repos")`. What do you see?

The function `options()` can be used to get or set global options of the current R session.

- Run `BiocManager::repositories()`.

The function reports the URLs from which to install Bioconductor and CRAN packages.

- Set the `repos` option to the value of `BiocManager::repositories()`.

```
options(repos = BiocManager::repositories())
```

- Run `options("repos")` again. What do you see now? What does it mean?

Import SummarizedExperiment objects using tximeta

- Load the packages *tximeta* and *tximportData*.

The help page of the `tximeta()` function refers to an example output of the Salmon program that is installed on your computer at `system.file("extdata/salmon_dm/SRR1197474/quant.sf", package="tximportData")`.

- Navigate to that location in the file explorer of your respective operating system, and examine its contents, in a text editor or your terminal application.
- Run the rest of the example code, and discuss the use of a linked transcriptome with `tximeta` (set `write=TRUE` instead of using the example code as-is, call the file `tximeta.json`, and read the help page of `makeLinkedTxome()`).
- What other types of quantification programs does `tximeta` support? Where do you find that information?

References

Su, S., V. Carey, et al. (2019). "BiocPkgTools: Toolkit for mining the Bioconductor package ecosystem [version 1; peer review: 2 approved, 1 approved with reservations] ". In: *F1000Research* 8.752. DOI: 10.12688/f1000research.19410.1.