Introduction to Bioconductor

Data Wrangling in R

The Bioconductor project

- Bioconductor is an open source, open development software project to provide tools for the analysis and comprehension of high-throughput genomic data. It is based primarily on the R programming language.
- Most Bioconductor components are distributed as R packages. The functional scope of Bioconductor packages includes the analysis of microarray, sequencing, flow sorting, genotype/SNP, and other data.

Project Goals

The broad goals of the Bioconductor project are:

- To provide widespread access to a broad range of powerful statistical and graphical methods for the analysis of genomic data.
- To facilitate the inclusion of biological metadata in the analysis of genomic data, e.g. literature data from PubMed, annotation data from Entrez genes.
- To provide a common software platform that enables the rapid development and deployment of extensible, scalable, and interoperable software.
- To further scientific understanding by producing high-quality documentation and reproducible research.
- To train researchers on computational and statistical methods for the analysis of genomic data.

Quick overview of the website

- biocViews
- Support site
- Teaching material
- Installation

Getting started

```
# Note that this is not evaluated here, so you will have to do it before using this knitr doc
install.packages("BiocManager")
# Install all core packages and update all installed packages
BiocManager::install()
```

Getting started

You can also install specific packages

```
# Note that this is not evaluated here, so you will have to do it before using this knitr doc
BiocManager::install(c("GEOquery", "limma", "biomaRt", "SummarizedExperiment"))
```

Bioconductor Workflows

https://bioconductor.org/packages/release/workflows/vignettes/sequencing/inst/doc/s

The Gene Expression Omnibus (GEO)

The Gene Expression Omnibus is an international public repository that archives and freely distributes microarray, next-generation sequencing, and other forms of high-throughput functional genomics data submitted by the research community.

The three main goals of GEO are to:

- Provide a robust, versatile database in which to efficiently store high-throughput functional genomic data
- Offer simple submission procedures and formats that support complete and well-annotated data deposits from the research community
- Provide user-friendly mechanisms that allow users to query, locate, review and download studies and gene expression profiles of interest

Getting data from GEO

For individual studies/datasets, the easiest way to find publicly-available data is the GEO accession number found at the end of publications.

Getting data from GEO

The `GEOquery` package can access GEO directly.

https://www.bioconductor.org/packages/release/bioc/html/GEOquery.html

```
library(GEOquery)
```

```
## Setting options('download.file.method.GEOquery'='auto')
```

```
## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
# https://pubmed.ncbi.nlm.nih.gov/32619517/
geo_data = getGEO("GSE146760")[[1]] # find accession in paper
```

```
## Found 1 file(s)
```

```
## GSE146760_series_matrix.txt.gz
```

```
## Parsed with column specification:
## cols(
##   ID_REF = col_character(),
##   GSM4405470 = col_character(),
```

Getting data from GEO

```
tibble(pData(geo_data))
```

```
## # A tibble: 11 x 44
##    title geo_accession status submission_date last_update_date type
##    <chr> <chr>         <chr>  <chr>           <chr>            <chr>
##  1 OCC … GSM4405470    Publi… Mar 10 2020     Jul 02 2020      SRA
##  2 OCC … GSM4405471    Publi… Mar 10 2020     Jul 02 2020      SRA
##  3 OCC … GSM4405472    Publi… Mar 10 2020     Jul 02 2020      SRA
##  4 OCC … GSM4405473    Publi… Mar 10 2020     Jul 02 2020      SRA
##  5 PFC … GSM4405474    Publi… Mar 10 2020     Jul 02 2020      SRA
##  6 PFC … GSM4405475    Publi… Mar 10 2020     Jul 02 2020      SRA
##  7 PFC … GSM4405476    Publi… Mar 10 2020     Jul 02 2020      SRA
##  8 PFC … GSM4405477    Publi… Mar 10 2020     Jul 02 2020      SRA
##  9 NSC-… GSM4405478    Publi… Mar 10 2020     Jul 02 2020      SRA
## 10 NSC-… GSM4405479    Publi… Mar 10 2020     Jul 02 2020      SRA
## 11 NSC-… GSM4405480    Publi… Mar 10 2020     Jul 02 2020      SRA
## # … with 38 more variables: channel_count <chr>, source_name_ch1 <chr>,
## #   organism_ch1 <chr>, characteristics_ch1 <chr>, characteristics_ch1.1 <chr>,
## #   growth_protocol_ch1 <chr>, molecule_ch1 <chr>, extract_protocol_ch1 <chr>,
## #   extract_protocol_ch1.1 <chr>, taxid_ch1 <chr>, description <chr>,
## #   description.1 <chr>, data_processing <chr>, data_processing.1 <chr>,
## #   data_processing.2 <chr>, data_processing.3 <chr>, platform_id <chr>,
## #   contact_name <chr>, contact_department <chr>, contact_institute <chr>,
## #   contact_address <chr>, contact_city <chr>, contact_state <chr>,
```

Getting data from GEO

Actual gene expression data, ie RNA-seq read counts, is less commonly stored in GEO.

```
exprs(geo_data) # gene expression
```

```
##      GSM4405470 GSM4405471 GSM4405472 GSM4405473 GSM4405474 GSM4405475
##      GSM4405476 GSM4405477 GSM4405478 GSM4405479 GSM4405480
```

```
fData(geo_data) # gene/feature/row annotation
```

```
## data frame with 0 columns and 0 rows
```

Getting data from GEO

Sometimes the gene expression matrices are stored as supplementary data.
https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146760

```
getGEOSuppFiles("GSE146760")
```

```
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASa
## /Users/johnmuschelli/Dropbox/Teaching/SISBID_Module1/lecture_notes/GSE146760/GSE146760_RNASe
##
```

Getting data from GEO

```
colnames(counts) = sapply(str_split(colnames(counts), "Aligned"), "[[", 1)
identical(colnames(counts), pheno$Prefix)
```

```
## [1] TRUE
```

```
rownames(pheno) = pheno$Status
colnames(counts) = pheno$Status
```

Getting data from GEO

SummarizedExperiment objects are probably the standard data structure for gene expression data.

https://bioconductor.org/packages/release/bioc/html/SummarizedExperiment.html

```
rse = SummarizedExperiment(assays = list(counts = counts),
                           colData = DataFrame(pheno))
```

Getting data from GEO

We can also add gene annotation information with the `biomaRt` package

```
library(biomaRt)
ensembl <- useEnsembl(biomart = "genes", dataset = "hsapiens_gene_ensembl")
geneMap = getBM(attributes = c("ensembl_gene_id",
                "chromosome_name","start_position",
          "end_position", "strand", "external_gene_name"),
            values=rownames(counts), mart=ensembl)
```

Genomic Ranges

Convert the data frame to a G[enomic]Ranges object:

```
geneMap$chromosome_name = paste0("chr", geneMap$chromosome_name)
geneMap$strand = ifelse(geneMap$strand == 1, "+", "-")
geneMap_gr = makeGRangesFromDataFrame(geneMap,
             seqnames.field = "chromosome_name",
             start.field = "start_position",
             end.field = "end_position")
names(geneMap_gr) = geneMap$ensembl_gene_id
geneMap_gr
```

```
## GRanges object with 67149 ranges and 0 metadata columns:
##                                seqnames              ranges strand
##                                   <Rle>           <IRanges>  <Rle>
##   ENSG00000210049                  chrMT             577-647      +
##   ENSG00000211459                  chrMT             648-1601     +
##   ENSG00000210077                  chrMT            1602-1670     +
##   ENSG00000210082                  chrMT            1671-3229     +
##   ENSG00000209082                  chrMT            3230-3304     +
##               ...                    ...                 ...    ...
##   ENSG00000285065  chrCHR_HSCHR11_2_CTG8   90223153-90226538      +
##   ENSG00000284997  chrCHR_HSCHR11_2_CTG8   90313371-90314983      +
##   ENSG00000284805 chrCHR_HSCHR3_9_CTG2_1 128148917-128149019      -
##   ENSG00000284869 chrCHR_HSCHR3_9_CTG2_1 128160388-128415576      +
```

Genomic Ranges

```
identical(rownames(counts), names(geneMap_gr))
```

```
## [1] FALSE
```

```
table(rownames(counts) %in% names(geneMap_gr))
```

```
##
## FALSE  TRUE
##   830 57221
```

```
mm = match(rownames(counts), names(geneMap_gr))
geneMap_gr = geneMap_gr[mm[!is.na(mm)]]
counts = counts[!is.na(mm),]
```

Summarized Experiments

```
rse = SummarizedExperiment(assays = list(counts = counts),
                           colData = DataFrame(pheno),
                           rowRanges = geneMap_gr)
rse
```

```
## class: RangedSummarizedExperiment
## dim: 57221 11
## metadata(0):
## assays(1): counts
## rownames(57221): ENSG00000000003 ENSG00000000005 ... ENSG00000283698
##   ENSG00000283699
## rowData names(0):
## colnames(11): Neuron01 Neuron02 ... NSC03 NSC04
## colData names(5): Status Replicate Prefix Code Context_Reps
```

Getting data from the Sequence Read Archive (SRA)

GEO originated for microarray data, which has largely become replaced by data produced using next-generation sequencing technologies. Depositing raw sequencing reads into the Sequence Read Archive (SRA) is often a condition of publication in many journals.

https://trace.ncbi.nlm.nih.gov/Traces/sra/?study=SRP044749

Raw data is annoying to process into gene counts

So we created the `recount` project https://jhubiostatistics.shinyapps.io/recount/

```
source("scale_counts.R") # or install recount package
load(file.path('SRP044749', 'rse_gene.Rdata'))
rse_gene = scale_counts(rse_gene)
rse_gene
```

```
## class: RangedSummarizedExperiment
## dim: 58037 6
## metadata(0):
## assays(1): counts
## rownames(58037): ENSG00000000003.14 ENSG00000000005.5 ...
##   ENSG00000283698.1 ENSG00000283699.1
## rowData names(3): gene_id bp_length symbol
## colnames(6): SRR1523347 SRR1523349 ... SRR1523354 SRR1523355
## colData names(21): project sample ... title characteristics
```