Introduction to Bioconductor

Data Wrangling in R

```r
#AnVIL::install("SummarizedExperiment")
#AnVIL::install("biomaRt")
```

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

https://bioconductor.org/packages/release/bioc/html/VariantAnnotation.html https://bioconductor.org/packages/release/bioc/html/Rsamtools.html

https://bioconductor.org/packages/release/bioc/vignettes/Rsamtools/inst/doc/Rsamtools-Overview.pdf

Getting started

```r
# Note that this is not evaluated here, so you will have to
install.packages("BiocManager")
# Install all core packages and update all installed packag
BiocManager::install()
```

Getting started

You can also install specific packages

```
# Note that this is not evaluated here, so you will have to
BiocManager::install(c("GEOquery", "limma", "biomaRt", "Su
```

Bioconductor Workflows

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

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 pa
```

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

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

Getting data from GEO

We can get the phenotypic data using the `pData()` function from `Biobase`

```
tibble(Biobase::pData(geo_data))
```

```
## # A tibble: 11 x 44
##    title    geo_a~1 status submi~2 last_~3 type  chann~4
##    <chr>    <chr>   <chr>  <chr>   <chr>   <chr> <chr>
##  1 OCC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  2 OCC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  3 OCC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  4 OCC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  5 PFC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  6 PFC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  7 PFC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  8 PFC dif~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
##  9 NSC-1 [~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
## 10 NSC-2 [~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
## 11 NSC-3 [~ GSM440~ Publi~ Mar 10~ Jul 02~ SRA   1
```

Getting data from GEO

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

Wh

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

```
##      GSM4405470 GSM4405471 GSM4405472 GSM4405473 GSM4405
##      GSM4405476 GSM4405477 GSM4405478 GSM4405479 GSM4405
```

```
Biobase::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. We can check it out using the `GEOquery` package. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE146760

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

```
##
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
##
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
##
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
##
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
## /Users/carriewright/Documents/GitHub/Teaching/Data-Wrang
##
```

Getting data from GEO

OK! so now we can start working with our data... first, we want to make sure these two files have all the same corresponding sample IDs. We want the `pheno$Prefix` column to be the same as the colnames of our count data. This is going to take some wrangling!

```
colnames(counts) = str_remove(string = colnames(counts), pa
identical(colnames(counts), pheno$Prefix)
```

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

OK could be a bit more clear

Now that we know they are identical, let's replace the column names of counts with the Status column values of pheno.

```
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

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

biomaRt

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

Guide:https://www.bioconductor.org/packages/devel/bioc/vignettes/biomaRt/inst/doc/accessing_ensembl.html

```
library(biomaRt)

if(interactive()){
listEnsembl()
}
#datasets <- listDatasets(ensembl)
#head(datasets)
#searchAttributes(mart = ensembl, pattern = "hgnc")
```

biomaRt

Guide:https://www.bioconductor.org/packages/devel/bioc/vignettes/biomaRt/inst/doc/accessing_ensembl.html

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

Biomart

```
head(geneMap)
```

```
##   ensembl_gene_id chromosome_name start_position end_pos
## 1 ENSG00000210049              MT            577
## 2 ENSG00000211459              MT            648
## 3 ENSG00000210077              MT           1602
## 4 ENSG00000210082              MT           1671
## 5 ENSG00000209082              MT           3230
## 6 ENSG00000198888              MT           3307
##   external_gene_name
## 1              MT-TF
## 2            MT-RNR1
## 3              MT-TV
## 4            MT-RNR2
## 5             MT-TL1
## 6             MT-ND1
```

Great! now we have info about the different ensemble genes!

Genomic Ranges

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

```
geneMap <-geneMap %>% mutate(chromosome_name = paste0("chr'
geneMap <-geneMap %>% mutate(strand = case_when(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 68324 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      +
```

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/

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