Omics Data management

Some useful tools and strategies

Ashfaq Ali

NBIS, SciLife lab, Lund University, Sweden

2019/08/14 (updated: 2021-09-01)

Why have a data management strategy?

It leads us to follow basic scietific pronciples

-- *Repoducibility*

-- *Avoid redunduancy*

-- *Productivity*

-- *Collaboration*

-- *Creativity*

Outline: Data Management Lecture

--**Background, Basics and resources**

- Componenets of data for reporducible research
- Basic data types in R (Vectors, matrices, data frames and Lists). Tutorial
- RMarkdown, Rdata, RDS objects to store and access data on disk

-- **Bioconductor packages and objects useful for omics data analyses**

- S3 and S4 class objects and methods
- IRAnges, GRanges, GenomicsRanges, BSGenomes, , BSGenome, BioMArt
- ExpressionSet, SummarisedExperiments, MultiAssayExperiments

-- **Data structures and manipulation in Python (Rui Benfeitas)**

Complementary tutorial is available with some practical examples and R codes.

Here we donot cover pre-processing of data from different platforms and assume that we will work with the processed data.

Componenets of data for reporduciple research

- **FAIR data guidelins** Findability, Accessibility, Interoperability, and Reusability FAIR
- **Raw Data** Raw data represent data from the facility where the data was obtaned from. Raw reads, spectra, images, counts etc. This depends on at what stage you boarded the ship.
- **Tiday Data** Organized and it's shareable and ready to be used.
  More on it later.
- **Code Book** This would represent things like a) What 0,1 codes represent b) Measurement scales (cm, ng/ml etc).
  R has an excellent way of storing this information in the form of factors (tutorial)
- **Scripts, Codes, Recipie, Github** An easy way to keep your analyses 'recipe' is by documentin the whole analyses or set of analyses in a script.
  A text/lig file for steps in ***GUI's*** (Cytoscape, MZmine etc). NBIS Course
- **Sensitive data and GDPR compliance** Be aware of the rules about person ***senstive data***.
  Read more about at these pages NBIS Website, GDPR, Data Protection.

Basic data types in R

- **Vectors**: Represent a set of single type of data such as string, Numerics an dintegers
    - *Fctors* are special vectors in R that allow dummy coding of elements. *Very helpful but source of confuion at times.*
- **Matrices** hold numerical data in two dimensions.
- **Arrays** Multodimensional
- **Data Frame** can hold any kind of data.
- **Lists** are very useful data structure for Bioindormatics Hold any type of data with any dimention.
- **S3 and S4 system** are very central to R programming: allowing to create objects to represent any data types, access and manupulate data using ***accessor*** functions. More later

A well known example linear model

```
mat<- matrix(rnorm(100), 25,4)
dat<- as.data.frame(cbind(rowSums( c(0.01, 0.02, 0.03, 0.04)* mat),mat))
colnames(dat) <- c("y","x1","x2","x3","x4")
fit<- lm(data = as.data.frame(dat), formula = y~x1+x2+x3+x4)
print(class(fit) ); print(typeof(fit))
```

```
## [1] "lm"
```

```
## [1] "list"
```

All functions applicable to lists are useable but it is reccomened to use class specific functions. Try `methods(summary)` on your computer. or try `methods(class="GRanges")`

```
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4, data = as.data.frame(dat))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.029910 -0.012573  0.001741  0.013091  0.026515
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.003764   0.004508  -0.835    0.414
## x1           0.030171   0.005305   5.687 1.45e-05 ***
## x2           0.027333   0.004397   6.216 4.53e-06 ***
## x3           0.022455   0.004512   4.977 7.24e-05 ***
```

Bioconductor packages and objects useful for omics data analyses

IRanges class to represent data that occurs in a sequence.

An IRanges **constructor** can be used to create and object of ***IRanges class***

```
ir <- IRanges(start=c(8, 12, 17, 21), width=c(4, 5, 4, 10),
              names = c("Morning", "Afternoon", "Evening", "Night"))
print(ir[1:2,])
```

```
## IRanges object with 2 ranges and 0 metadata columns:
##                 start       end     width
##             <integer> <integer> <integer>
##     Morning         8        11         4
##   Afternoon        12        16         5
```

One can apply class specific methods to objects.

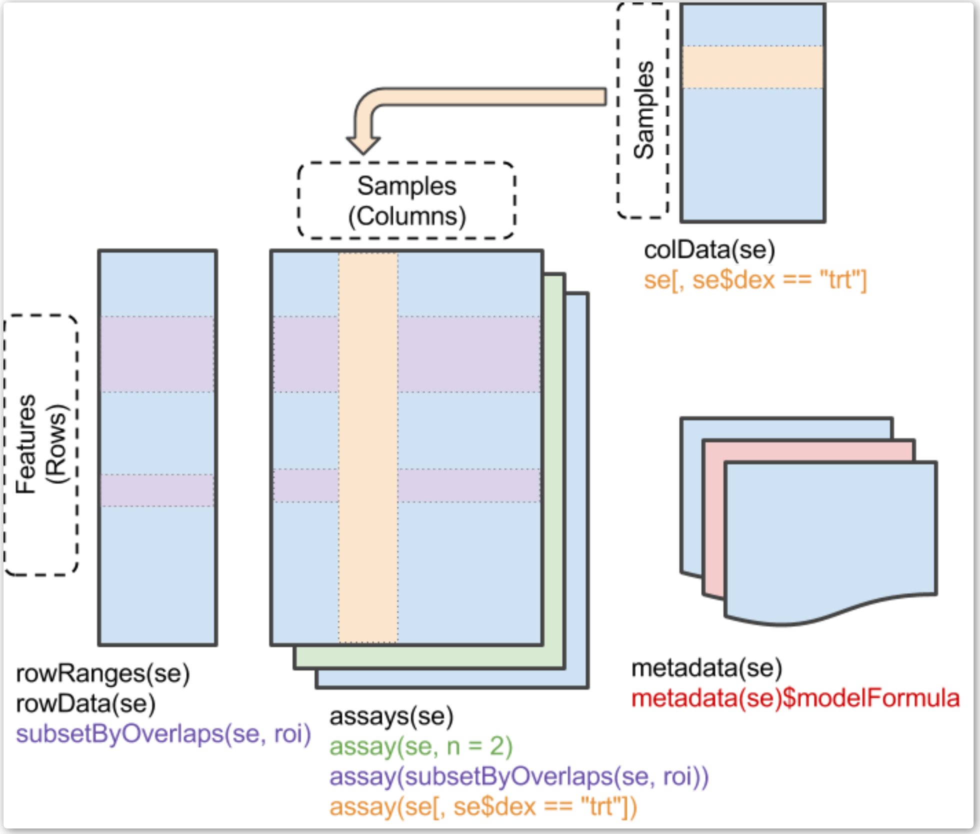
GenomicsRAnges

Extention of `IRanges`

```
library(GenomicRanges)
gr <- GRanges( seqnames = Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
    ranges = IRanges(101:110, end = 111:120, names = head(letters, 10)),
    strand = Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)),
    score = 1:10,
    GC = seq(1, 0, length=10))
gr [1:4,]
```

```
## GRanges object with 4 ranges and 2 metadata columns:
##     seqnames    ranges strand |     score        GC
##        <Rle> <IRanges>  <Rle> | <integer> <numeric>
##   a     chr1   101-111      - |         1  1.000000
##   b     chr2   102-112      + |         2  0.888889
##   c     chr2   103-113      + |         3  0.777778
##   d     chr2   104-114      * |         4  0.666667
##   -------
##   seqinfo: 3 sequences from an unspecified genome; no seqlengths
```

SummarizedExperiment class in R



A visual overiview of a SummarizedExperiment. Notice the similarity to and *ExpressionSet* class. The *ExpressionSet* Class is the source of inspiration form Bioconductor *SummarizedExperimet* and *MultiAssayExperiment* owing to it's success.

Demonstrate with real data

Annotations from BioMart for Feature data

- *txdb from biomart*
- *Gene annotations from biomart*

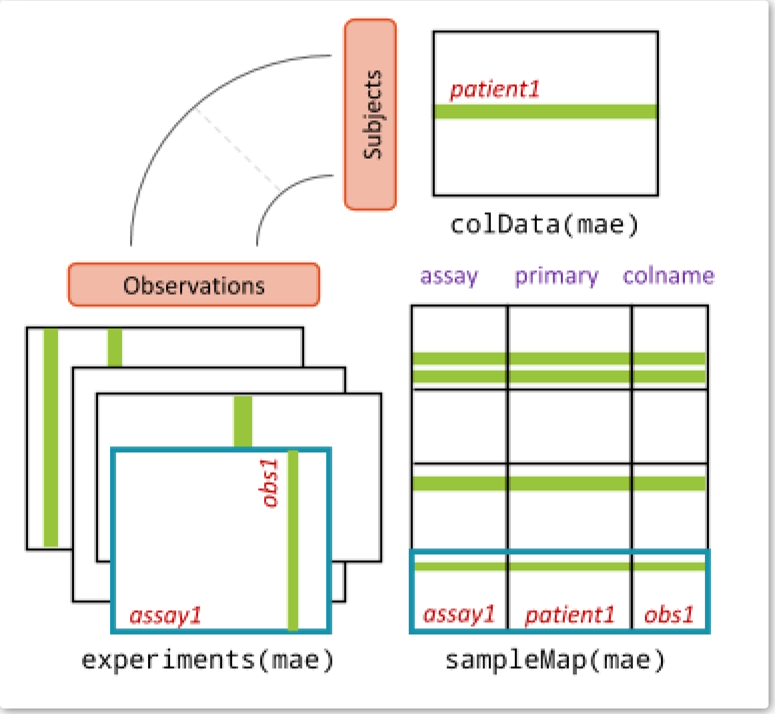
Summarized Experiment

- *RowData from txdb object such as genomic coordinates and gene/transcript annotations*
- *PhenoData such as patient IDs and characteristics*
- *Assays such as counts, intensities, normalized counts*
- *MetaData such as nature of experiment, how wa it performed*

MultiAssayExperiment

- *Create differnet experiments*

Multi Assay Experiment



ExperimentList. colData": “**primary**”, *sampleMap*, *metaData*

```
mes<- MultiAssayExperiment::MultiAssayExperiment()
slotNames(mes)
```

```
## [1] "ExperimentList" "colData"        "sampleMap"      "drops"
## [5] "metadata"
```

Create MultiAssayExperiment

- *Create differnet experiments*
- *ExperimentList* The datasets contained in elements of the ExperimentList can have:
    - column names (required)
    - row names (optional)
- *colData"*: **“primary”**
    - metadata that describes the ‘biological unit’ which can refer to specimens, experimental subjects, patients, etc.
    - colData dataset must contain patient identifiers.

Multi Assay Experiment (Cont)

- *sampleMap*
  - The sampleMap is a DataFrame that relates the "primary" data (colData) to the experimental
  - assays:assay provides the names of the different experiments / assays performed. These are user-defined, with the only requirement that the names of the ExperimentList, where the experimental assays are stored, must be contained in this column.
  - primary provides the "primary" sample names. All values in this column must also be present in the rownames of - colData(MultiAssayExperiment).
  - colname provides the sample names used by experimental datasets, which in practice are often different than the primary sample names. For each assay, all column names must be found in this column.
- *metadata*
  - Metadata can be added at different levels of the MultiAssayExperiment. Can be of ANY class, for storing study-wide metadata, such as citation information.

Save and loading the data (demonstration)

RMarkdown, Rdata, RDS objects to store and access data on disk

```
fil <- tempfile("women", fileext = ".rds")
#save a single object to file
#saveRDS(women, fil)
## restore it under a different name
#women2 <- readRDS(fil)
#identical(women, women2)
## or examine the object via a connection, which will be opened as needed.
#con <- gzfile(fil)
#readRDS(con)
#close(con)
```

Thank You