UTSouthwestern Medical Center
Lyda Hill Department of Bioinformatics

BioHPC

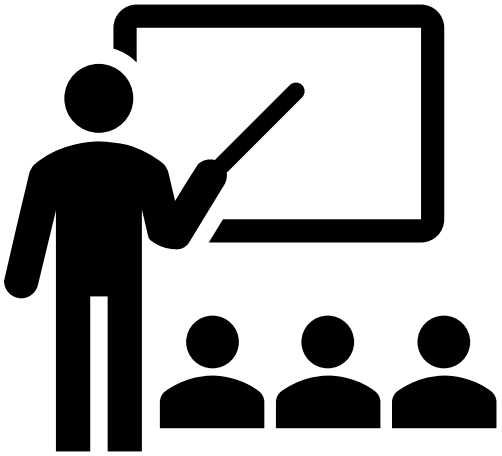
Data handling with R

The meeting will begin soon.
Please mute your microphone.

[email] biohpc-help@utsouthwestern.edu
[register] portal.biohpc.swmed.edu/accounts/register
[portal] portal.biohpc.swmed.edu

Welcome! A Quick Note Before Getting Started…

- Most future BioHPC training sessions will be hybrid!
- Choose to join us online, or **In-Person**
- **Classroom Location: G9.102**
- Users are encouraged to attend in-person.



Outline

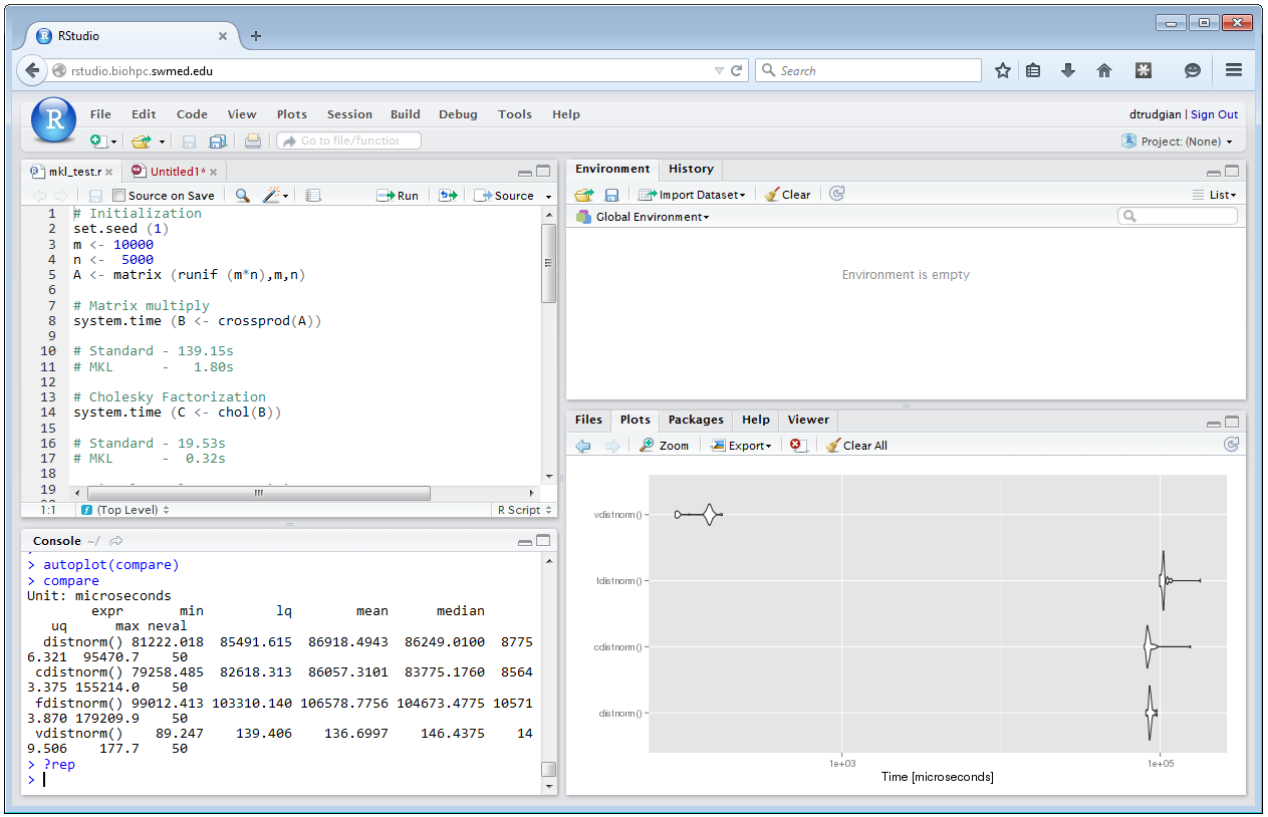
- Background
  - Advantages of R
  - R and Rstudio on BioHPC
  - R markdown
  - Variable types and data structure in R
- Why we need data preprocessing?
- Tasks in data preprocessing
  - Data cleaning, data integration, data transformation, data reduction, and data discretization
  - R code examples

Advantages of R

- The dominant statistics environment in academia
- Large number of packages to do a lot of different analysis
- Excellent packages in Bioinformatics
  - Bioconductor: open-source project, provide tools for the analysis and comprehension of high-throughput genomic data
  - Bioconductor: has different way to install packages
- (Relatively) easy to accomplish complex statistic work

RStudio – An IDE for R, on the web

Rstudio Server, http://rstudio.biohpc.swmed.edu



BioHPC optimized R, access to cluster storage, persistent sessions, R 3.3.2

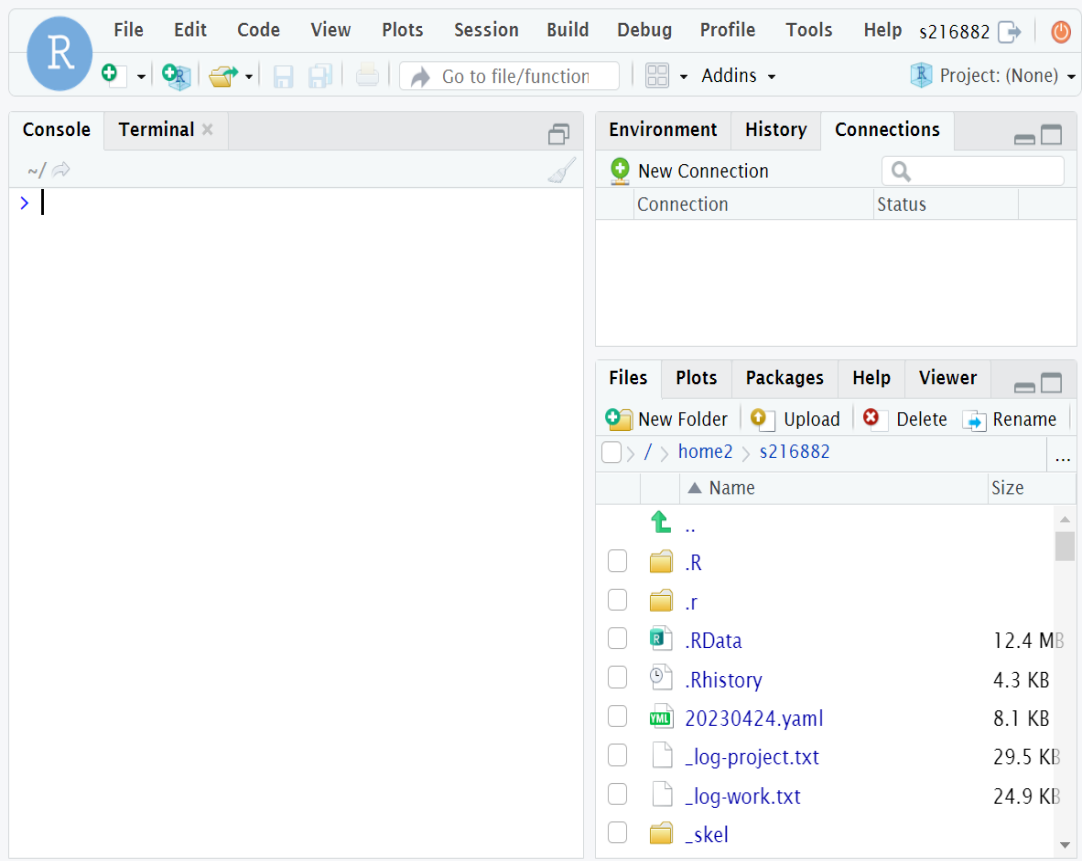
Rstudio on demand

Standard 20 hr limit

Whole node to yourself

You can choose R/3.3.2, 3.4.1, 3.5.1 with Seurat, R/3.6.1, R/4.0.2, R/4.1.1

We are working on R/4.2



Portal.biohpc.swmed.edu --> BioHPC Ondemand --> Ondemand Rstudio

https://portal.biohpc.swmed.edu/intranet/terminal/ondemand_rstudio/

When to use Rstudio server or Rstudio OnDemand

- Any small, short-running data analysis tasks
- Development work with small datasets
- Creating R Markdown documents

Rstudio Server
rstudio.biohpc.swmed.edu

Large datasets, parallel code	RStudio OnDemand
Long running jobs	Terminal R on the cluster…

Using R on the cluster / clients

```
[s216882@Nucleus006 ~]$ module av R/

------------------------------------ /cm/shared/modulefiles ---
R/2.15.3-intel          R/3.3.2-gccmkl_20181025 R/3.6.1-img
R/3.0.2                 R/3.4.1-gccmkl          R/4.0.2-gccmkl
R/3.1.0                 R/3.4.1-gccmkl_20181025 R/4.1.1-gccmkl
R/3.1.0-intel           R/3.5.1-gccmkl          R/4.1.1-img
R/3.2.1-intel           R/3.5.1-gccmkl_20181025 R/4.2.2-img
R/3.3.2-gccmkl          R/3.6.1-gccmkl
```

Default is R/3.3.2-gccmkl – module load <package_name>

Rstudio On Demand support R/3.3.2, 3.4.1, 3.5.1 with Seurat, 3.6.1, 4.0.2, R/4.1.1
All gccmkl

Use 'R' for command line R, or run scripts with 'Rscript'/submit your script

$ Rscript –vanilla myjob.R

Installing Packages and troubleshooting

We have a set of common packages pre-installed in the R module

You can install your own into your home directory "check **.libPaths()** in R"

```
install.packages("microbenchmark")
```

```
personal directory: ~/R/x86_64-pc-linux-gnu-library/<R ver>
```

Some packages need additional libraries, won't compile successfully.

- Ask us to install them for you (biohpc-help@utsouthwestern.edu)

Personal R library – troubleshooting package installation

- Terminal R and Rstudio server, Rstudio OnDemand shared personal libraries
- For example, if you has error in Rstudio OnDemand 4.0.2, please try corresponding terminal R/4.0.2-gccmkl to install the package into your personal library
- Sys.setenv(http_proxy= 'http://proxy.swmed.edu:3128')
  Sys.setenv(https_proxy= 'http://proxy.swmed.edu:3128')

Bioconductor

A comprehensive set of Bioinformatics related packages for R

Bioconductor - BiocViews

www.bioconductor.org/packages/release/BiocViews.html#__Software

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

Search:

Home | Install | Help | Developers | About

Home » BiocViews

All Packages

Bioconductor version 3.1 (Release)

Autocomplete biocViews search:

- Software (1024)
  - AssayDomain (345)
  - BiologicalQuestion (313)
  - Infrastructure (211)
  - ResearchField (225)
  - StatisticalMethod (293)
  - Technology (645)
  - WorkflowStep (525)
- AnnotationData (883)
- ExperimentData (241)

Packages found under Software:

Show All entries Search table:

Package	Maintainer	Title
a4	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Umbrella Package
a4Base	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Base Package
a4Classif	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Classification Package
a4Core	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Core Package
a4Preproc	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Preprocessing Package
a4Reporting	Tobias Verbeke, Willem Ligtenberg	Automated Affymetrix Array Analysis Reporting Package
ABarray	Yongming Andrew Sun	Microarray QA and statistical data analysis for Applied Biosystems Genome Survey Microrarray (AB1700) gene expression data.
ABSSeq	Wentao Yang	ABSSeq: a new RNA-Seq analysis method based on absolute expression differences and generalized Poisson model
aCGH	Peter Dimitrov	Classes and functions for Array Comparative Genomic Hybridization data.

Software *and* datasets

Bioconductor

Base packages installed, plus some commonly used extras

Install additional packages to home directory:

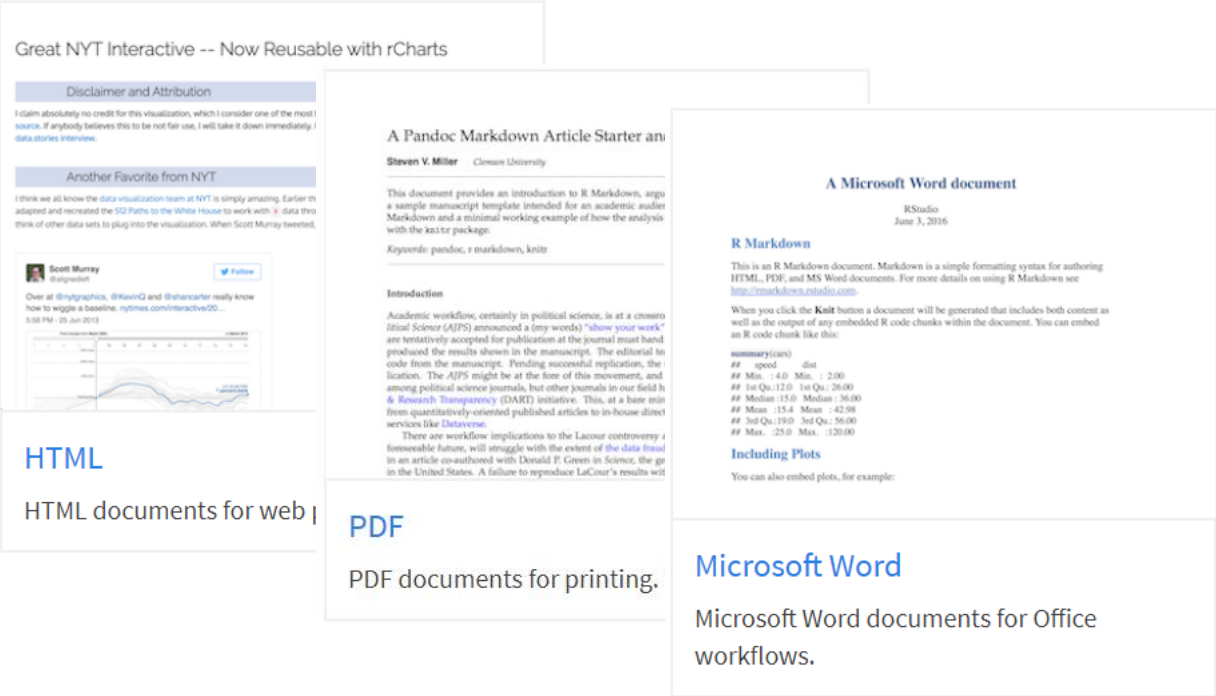
```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("GenomeInfoDb") #specific package
```

Ask biohpc-help@utsouthwestern.edu for packages that fail to compile

Rmarkdown / Knitr

Write R code inside markdown documents

Create attractive HTML, PDF, Word files that include both code and output



https://rmarkdown.rstudio.com/lesson-1.html

Variables in R

- character: "treatment", "123", 'A', "A"
- numeric: 23.44, 120, NaN, Inf
- integer: 4L, 1123L
- logical: TRUE, FALSE, NA
- factor: factor("Hello"), factor(8)
  - categorical variables

```
> class("hello")
[1] "character"

> class(3.844)
[1] "numeric"

> class(77L)
[1] "integer"

> class(factor("yes"))
[1] "factor"

> class(TRUE)
[1] "logical"
```

AA, “Data handling with R”, 2022

Operators

- Arithmetic Operators
  - +,-,*,/,^,%%
- Relational Operators
  - >,<,==,!=
- Logical Operators
  - &,|,!
- Assignment Operators
  - <- or = or ->
- Miscellaneous Operators
  - :, %in%

R data structures summary

	Homogeneous	Heterogeneous
1d	Atomic vector	List
2d	Matrix	Data Frame Tibble
nd	Array	

R data structures

- **Vectors**

```
> a <- c(1,2,5.3,6,-2,4) # numeric vector
> a
> b <- c("one","two","three") # character vector
> b
> c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE) #logical vector
> (c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)) #logical vector
```

- **Matrices** (All columns in a matrix must have the same mode(numeric, character, etc.) and the same length)

```
> y <- matrix(1:20, nrow=5, ncol=4) # generates 5 x 4 numeric matrix

> cells <- c(1,26,24,68)
> rnames <- c("R1", "R2")
> cnames <- c("C1", "C2")
> mymatrix <- matrix(cells, nrow=2, ncol=2, byrow=TRUE,
  dimnames=list(rnames, cnames))
```

R data structure 2

- **Arrays** are similar to matrices but can have more than two dimensions

```
> a <- array(c("green","yellow"),dim = c(3,3,2))
```

- **Data Frames** are more general than a matrix, in that different columns can have different modes (numeric, character, factor, etc.)

Are the most commonly used data structure in R

```
> d <- c(1,2,3,4)
> e <- c("red", "white", "red", NA)
> f <- c(TRUE,TRUE,TRUE,FALSE)
> mydata <- data.frame(d,e,f)
> mydata
> names(mydata) <- c("ID","Color","Passed") # variable names
```

Import data

R can read data from files

- Very important concept: **Working Directory** (this is where R will read data from by default)

```
> getwd()              # get current working directory

> setwd("<new path>")  # set working directory
```

Note that the forward slash should be used as the path separator even on Windows platform `> setwd("C:/MyDoc")`

File import - csv

CSV File

- Each cell inside is separated by a special character, which usually is a comma, although other characters can be used as well. The first row of the data file should contain the column names instead of the actual data.

```
> mydata = read.csv("mydata.csv")  # read csv file
```

```
Col1,Col2,Col3
100,a1,b1
200,a2,b2
300,a3,b3
```

- more import functions - http://www.r-tutor.com/r-introduction/data-frame/data-import

File import – csv example

The behavior of the different import functions varies slightly.

```
> data<-
read.csv("household_power_consumption.txt",
sep=";", header = FALSE, stringsAsFactors=FALSE,
na.strings = "?", skip=66637 , nrows=2880)
```

```
> colnames(data) <-
names(read.csv("household_power_consumption.txt"
, sep=";", nrows=1))
```

#set the column names

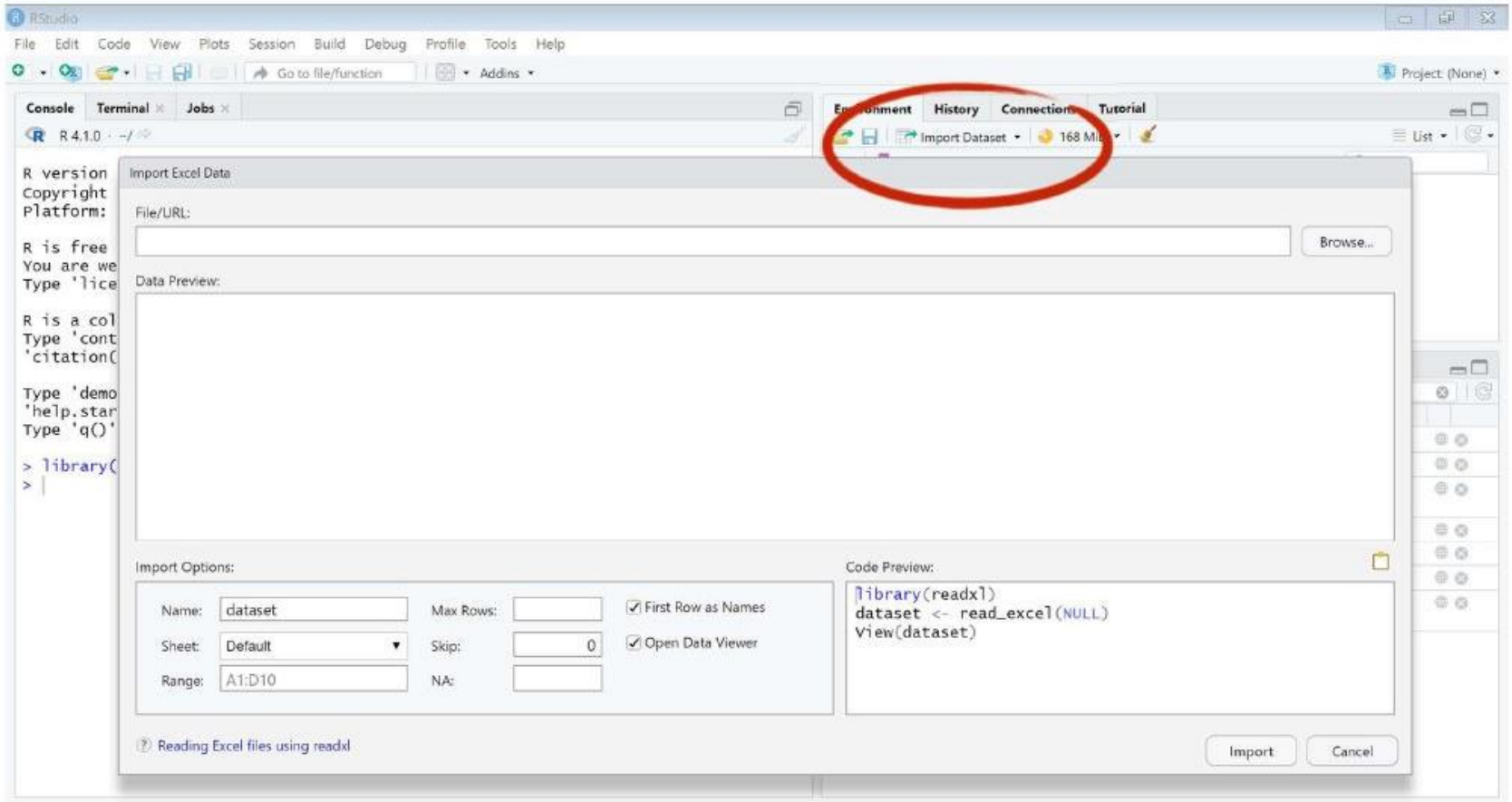
File import – excel file

- Quite frequently, the sample data is in Excel format, and needs to be imported into R prior to use. For this, we can use the functions from the *readxl* package. It reads from an Excel spreadsheet and returns a data frame.

```
> library(readxl)  # load readxl package

> mydata <- read_xls("mydata.xls")  # read from first sheet

> mydata <- read_excel("mydata.xlsx")
```

- Recommendation when issues occur: Store Excel file as tab separated file and use RStudio "Import" function.

Using Rstudio to import



Working with data – helpful commands

Get to know your data ...

```
> ?mtcars                # General info about data set

> head(mtcars)           # First couple of lines

> str(mtcars)            # Shows that the data is a data frame: A rectangular structure
                         # Each column has same type, but different
                         # columns may have different types

> names(mtcars)          # List the column names

> summary(mtcars)        # summary statistics
```

Dealing with missing values

- Counting missing values

```
> x <- c(1, 2, NA, 4)

> sum(is.na(x)) # sums up the missing values
in a column

> 1
```

- Which one is NA?

```
> which(is.na(x))

> 3
```

Dealing with missing values

- Excluding Missing Values from Analyses is often necessary since the default is to propagate missing values. Many functions have *na.rm* argument to remove them

```
> x <- c(1,2,NA,3)
> mean(x) # returns NA
> mean(x, na.rm=TRUE) # returns 2
```

- The function *complete.cases()* returns a logical vector indicating which cases are complete.

```
# list rows of data that have missing values
> mydata[!complete.cases(mydata),]
```

- The function *na.omit()* returns the object with listwise deletion of missing values.

```
# create new dataset without missing data
> newdata <- na.omit(mydata)
```

Data export

- As for import there are endless export options
- Check the arguments in the documentation for special cases

```
> write.table(mydata, “c:/Users/[username]/mydata.txt”, sep=“\t")

> write.csv(mydata, file = “mydata.csv", row.names = FALSE, quote
= FALSE)

> library(xlsx)

> write.xlsx(mydata, "c:/Users/[username]/mydata.xlsx")

> write.xlsx(x = mydata, file = “testexcelfile.xlsx", sheetName =
"TestSheet", row.names = FALSE)
```

Why we need data preprocessing?

Real world data is generally:

- Incomplete
  - Certain attributes or values or both are missing, or only aggregate data is available
- Noisy
  - Data contains errors or outliers
- Inconsistent
  - Data contains differences in codes or names

- Ref to https://towardsdatascience.com/data-preprocessing-e2b0bed4c7fb

Tasks in data preprocessing

- Data cleaning
  - Involves filling of missing values
  - Smoothing or removing noisy data and outliers along with resolving inconsistencies
- Data integration
  - Involves integrating data from multiple sources such as databases and files, etc
  - The data obtained can be structured, unstructured or semi-structured in format
- Data transformation
  - Involves normalization and aggregation of data according to the needs of the data set
- Data reduction
  - The number of records or the number of attributes or dimensions can be reduced
  - reduced data should produce the same results as original data
- Data discretization
  - as a part of data reduction
  - The numerical attributes are replaced with nominal ones

Data cleaning: Imputation

- Task: filling of missing values
- Remove variables having missing values

```
> na.omit(df)  ## method-1 for removing rows with some NAs
> df[complete.cases(df), ]  ## method-2
```

- Using most frequent or zero values

```
> df[is.na(df)] <- 0
```

- Using mean/median values

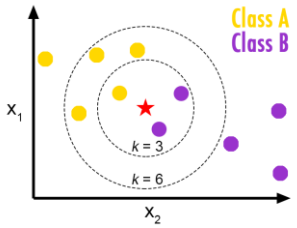
```
> mean(df[,1], na.rm=TRUE)
```

	col1	col2	col3	col4	col5
0	2	5.0	3.0	6	NaN
1	9	NaN	9.0	0	7.0
2	19	17.0	NaN	9	NaN

mean() →

	col1	col2	col3	col4	col5
0	2.0	5.0	3.0	6.0	7.0
1	9.0	11.0	9.0	0.0	7.0
2	19.0	17.0	6.0	9.0	7.0

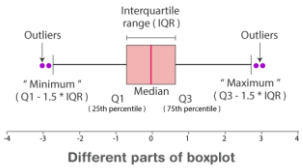
- Using k-Nearest Neighbors (k-NN)
  - Using k-nearest neighbor averaging, impute.knn()

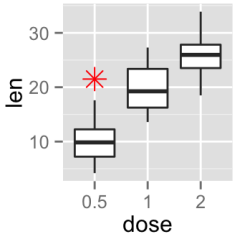


Jeon Lee, “machine learning using R”, 2022

Data cleaning: Outlier Removal

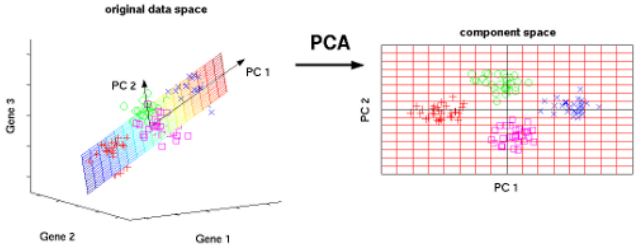
- Outlier detection
  - Using percentile info, boxplot

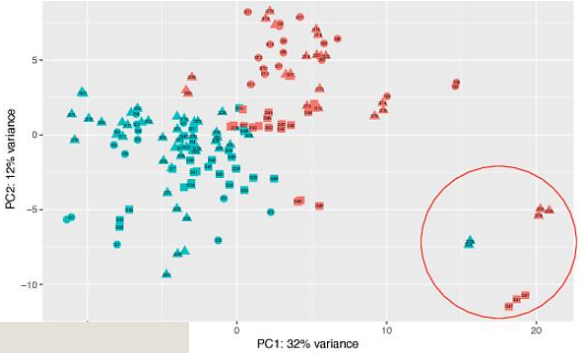





```
> ggplot(ToothGrowth, aes(x=dose, y=len)) +
  geom_boxplot(outlier.colour="red", outlier.shape=8, outlier.size=4)
  ## example for 'ToothGrowth' data
```

  - Using sample distance, PCA plot



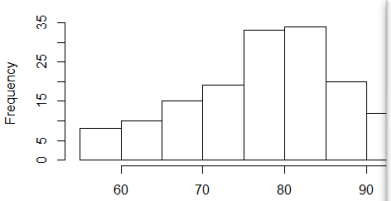


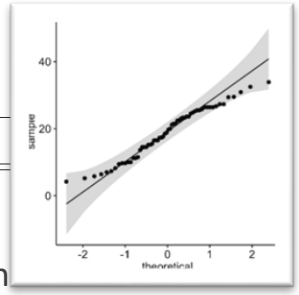

```
> pca_res <- prcomp(df, scale. = TRUE)
> autoplot(pca_res)  ## 'ggfortify' package
```

Data transformation: Normalization (1)

- Check each variable's distribution

```
> hist(values, breaks=10, xlim=c(50, 100))  ## using a histogram
> ggqqplot(values)  ## using a Q-Q plot, ggpubr package
```

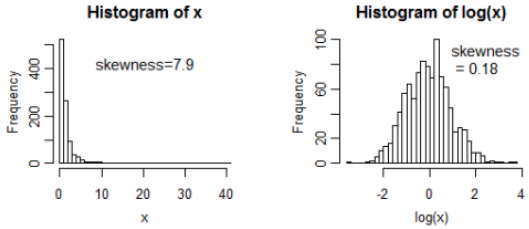




- Test normality
  - *P*-value < 0.05 implying that the distribution is significantly different from normal distribution

```
> shapiro.test(values)
```

- Transformation sometimes required
  - Log-transform, log(x): RNA-seq, MS metabolomics, etc.
  - Square root transform, sqrt(x)
  - Reciprocal Transformation, 1/x



Data transformation: Normalization (2)

- Z-transformation required
  - To make variables comparable
  - values converted into z-scores: $z_i = \frac{x_i - \bar{x}}{s}$

```
> scale(x, center=TRUE, scale=TRUE)  ## z-transformation
```

- Variance Stabilizing Transformation (VST) normalization