Intro to R and Bioconductor

HMS Research Computing

William G. Rodriguez <William_Rodriguez@hms.harvard.edu>

Please fill out the survey

- Accessible through the Harvard Training Portal
- https://trainingportal.harvard.edu/
- Click on “Me” then “Intro to O2”
- Scroll to “Evaluations” and click on the survey
- We appreciate any feedback or comments!

Course Objectives

- Learn to run Rstudio on O2
- Gain familiarity with R and Bioconductor
- Learn to import and export data
- Class Exercise

Notation

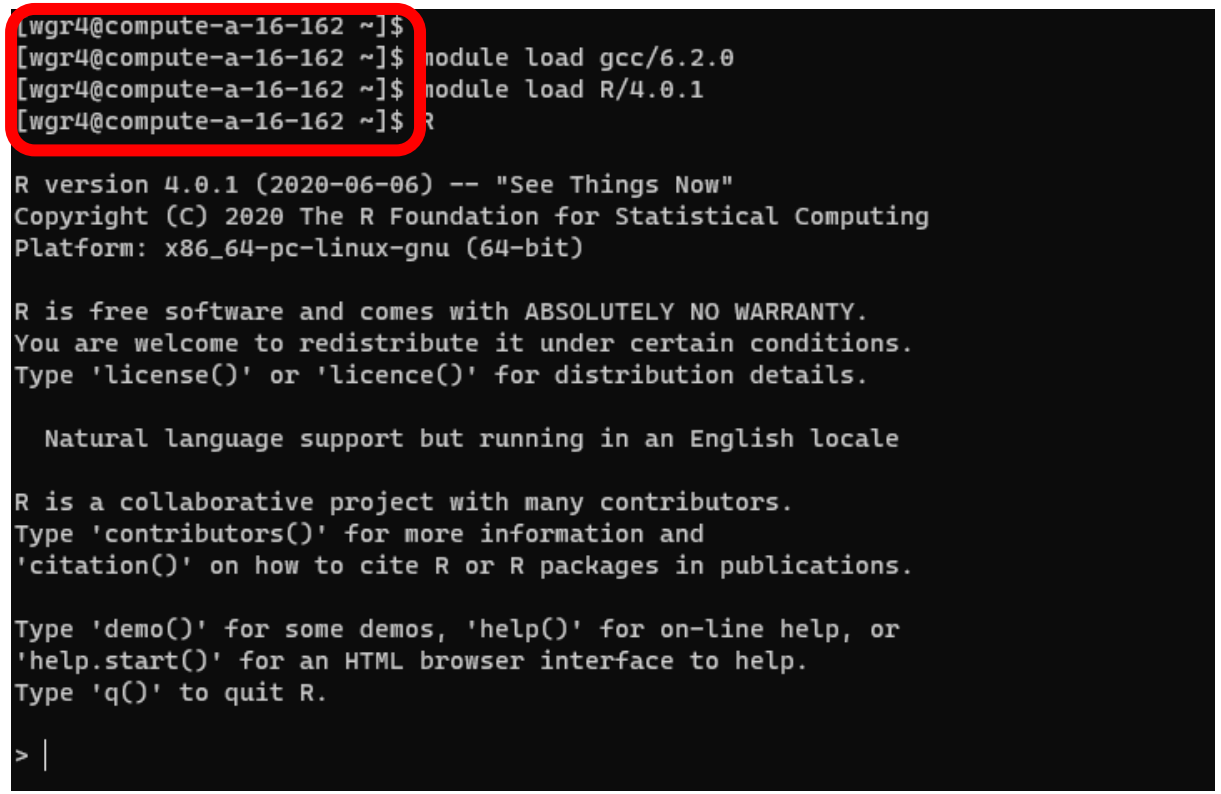
Blue content: try it out!

Notation

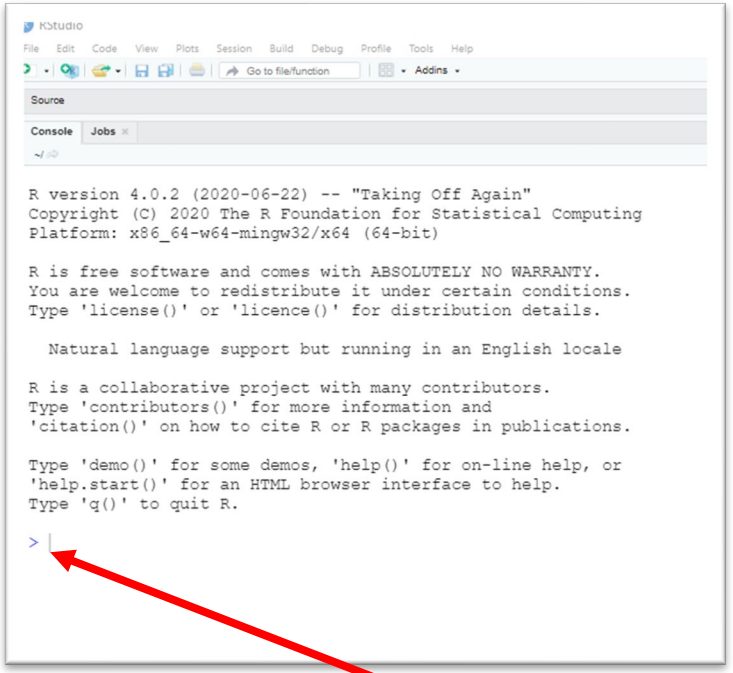
Bash

(line ends with “$”)

Terminal



RStudio



Terminal

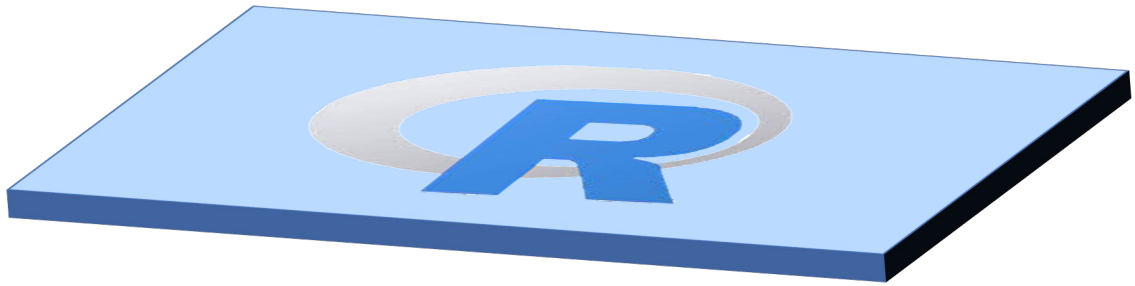
```
[wgr4@compute-a-16-162 ~]$
[wgr4@compute-a-16-162 ~]$ module load gcc/6.2.0
[wgr4@compute-a-16-162 ~]$ module load R/4.0.1
[wgr4@compute-a-16-162 ~]$ R

R version 4.0.1 (2020-06-06) -- "See Things Now"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

R

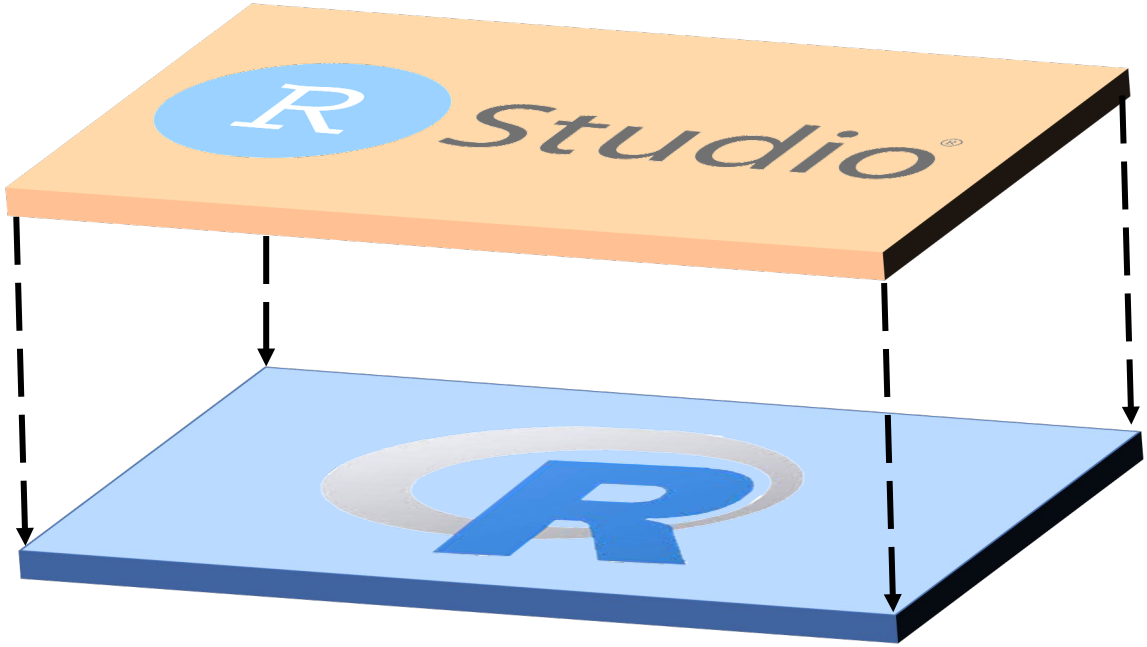
(line starts with “>”)

R on O2

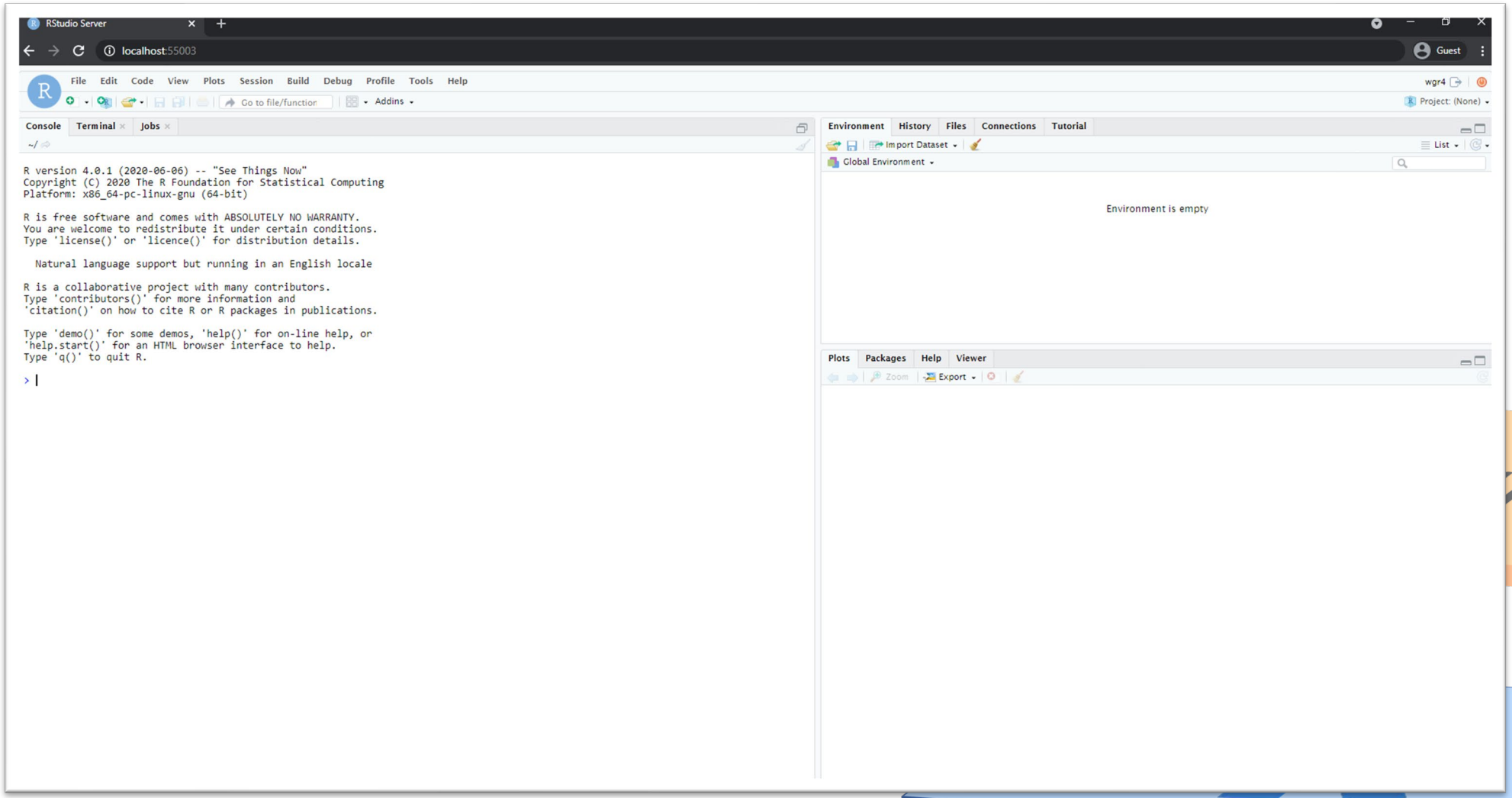
```
R version 4.0.1 (2020-06-06) -- "See Things Now"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```



RStudio on O2

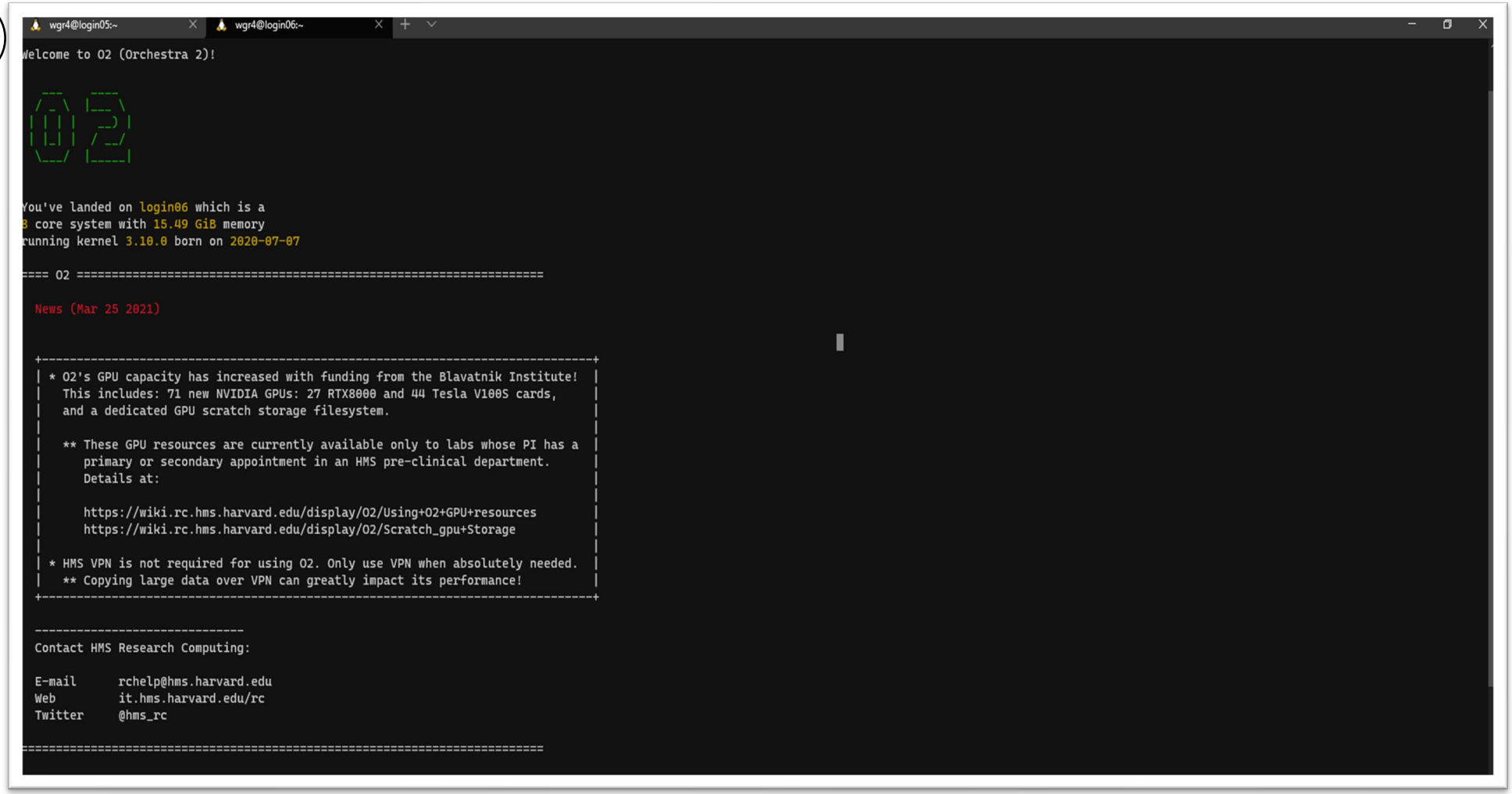


RStudio on O2

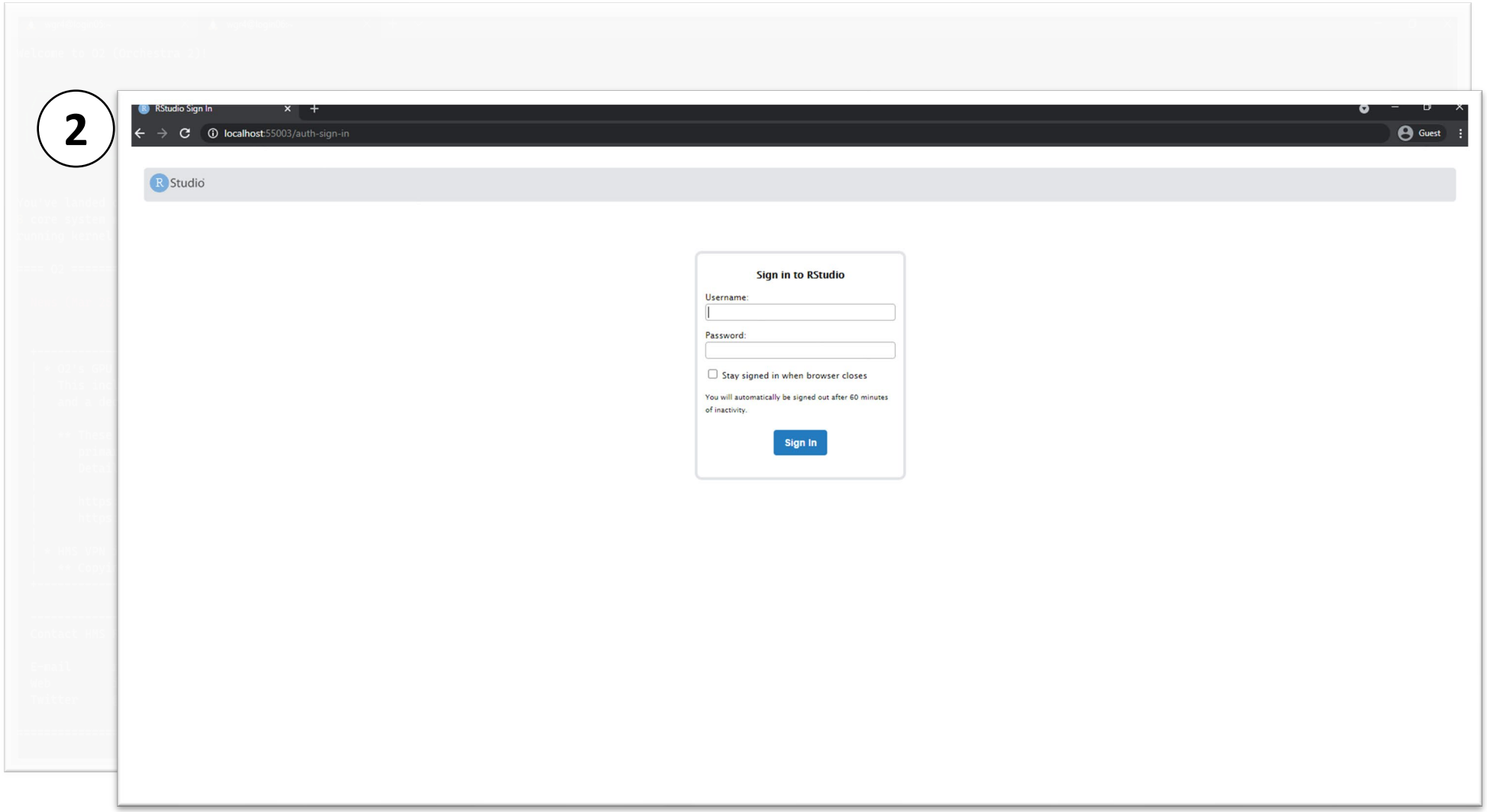


Launch RStudio on O2

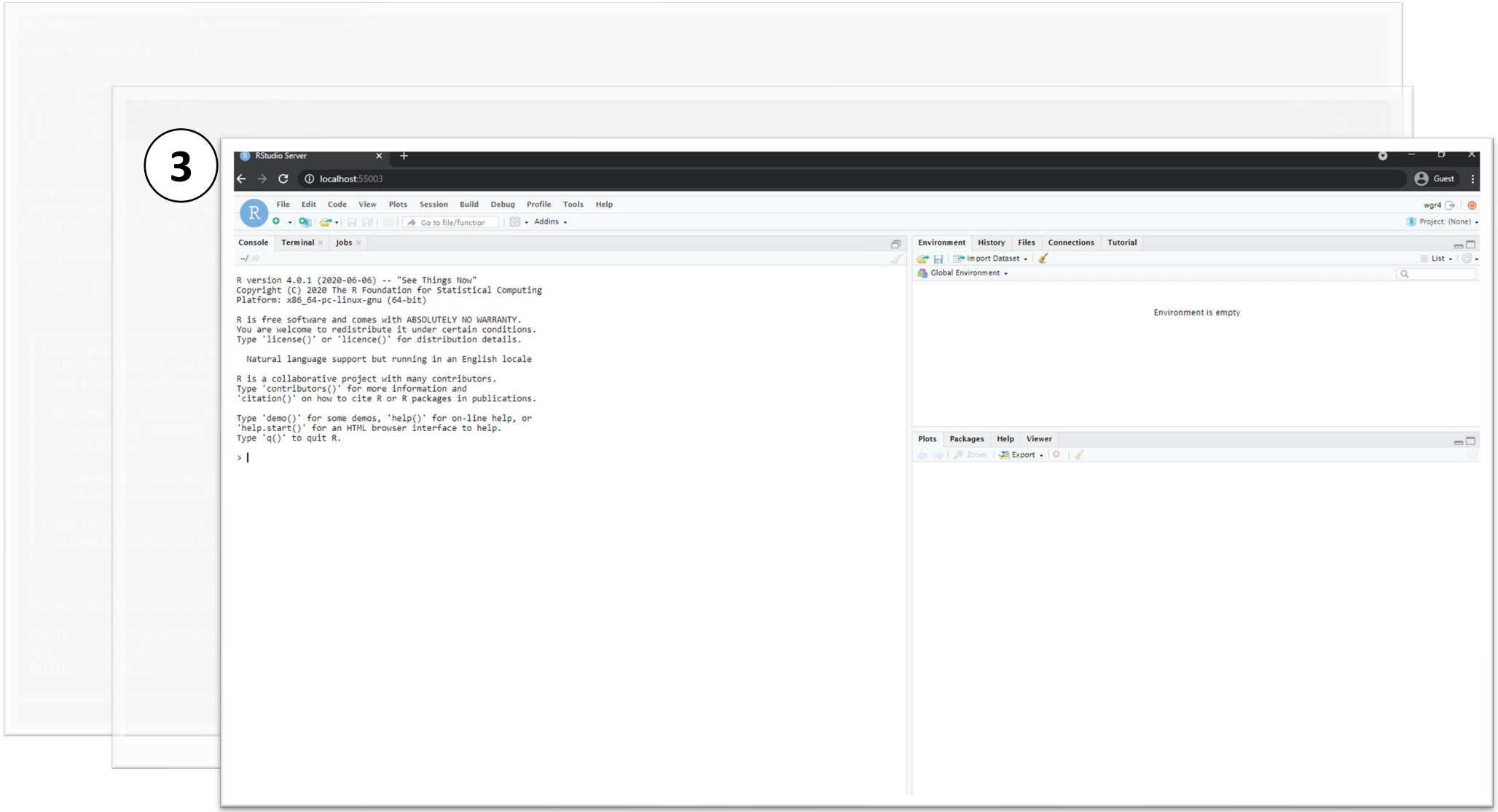
1



Launch RStudio on O2



Launch RStudio on O2



R on O2

- Available versions

  $ module spider R

- Load R module

  $ module load gcc/6.2.0 R/version

- How to unload an R module

  $ module unload R/version

- Important: start R from an interactive session (not login!)

  $ R

Managing your R packages on O2

- An R Personal Library is required on O2
- You must create an R Personal Library per version.

Managing your R packages on O2

- An R Personal Library is required on O2
- You must create an R Personal Library per version.
- Let's set up an R Personal Library in 2 simple steps!

1) Create an R Personal Library directory

$ mkdir ~/R-4.0.1

2) Create an .Renviron file

$ echo 'R_LIBS_USER="~/R-4.0.1”’> $HOME/.Renviron'

RStudio on O2

1. Connect to O2

   me@my_computer:~$ **ssh -Y -L PORT:127.0.0.1:PORT ecommons@o2.hms.harvard.edu**

2. Load Modules

   ecommons@login01:~$ **module load rstudio_launcher/1.0**

   ecommons@login01:~$ **module load gcc/6.2.0**

   ecommons@login01:~$ **module load R/4.0.1**

3. Launch RStudio

   ecommons@login01:~$ **srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 –tunnel PORT:PORT RStudio_launcher.sh PORT**

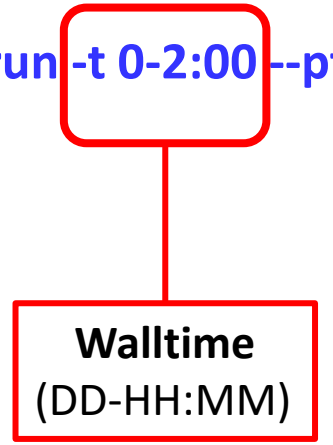
Launch RStudio command on O2

```
$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT
```

SLURM command to obtain a job allocation

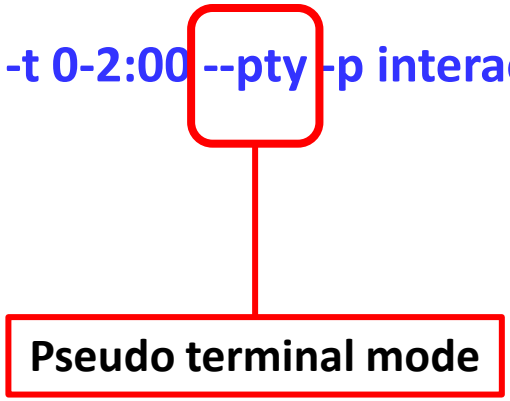
Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



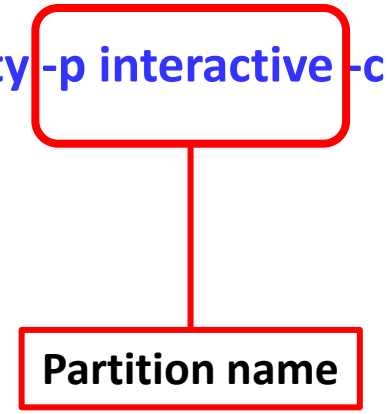
Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



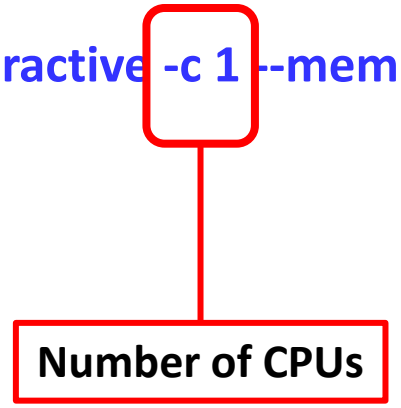
Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



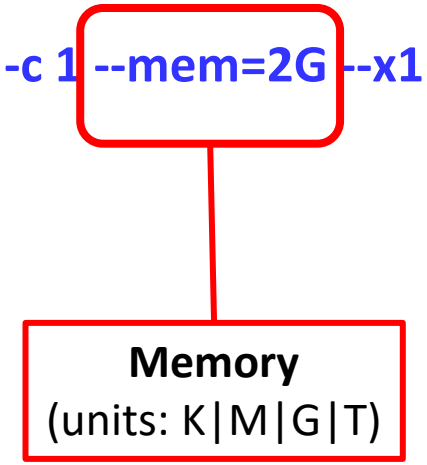
Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



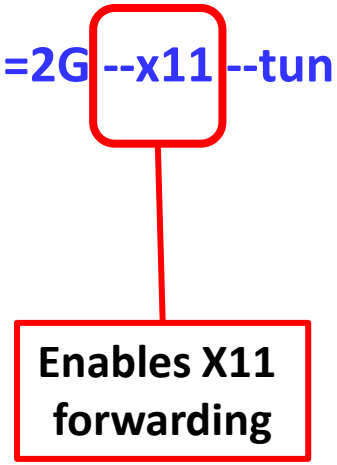
Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT



Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT

Required to execute RStudio launcher

Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT

Launch RStudio command on O2

$ srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 --tunnel PORT:PORT RStudio_launcher.sh PORT

RStudio on O2

1. Connect to O2

   me@my_computer:~$ **ssh -Y -L PORT:127.0.0.1:PORT ecommons@o2.hms.harvard.edu**

2. Load Modules

   ecommons@login01:~$ **module load rstudio_launcher/1.0**

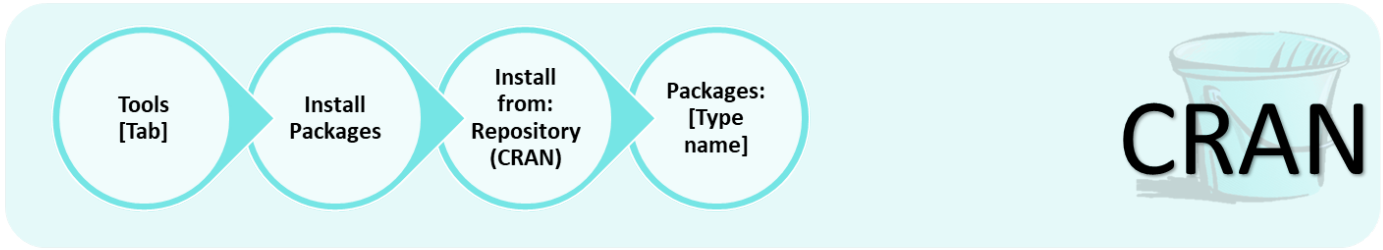
   ecommons@login01:~$ **module load gcc/6.2.0**

   ecommons@login01:~$ **module load R/4.0.1**

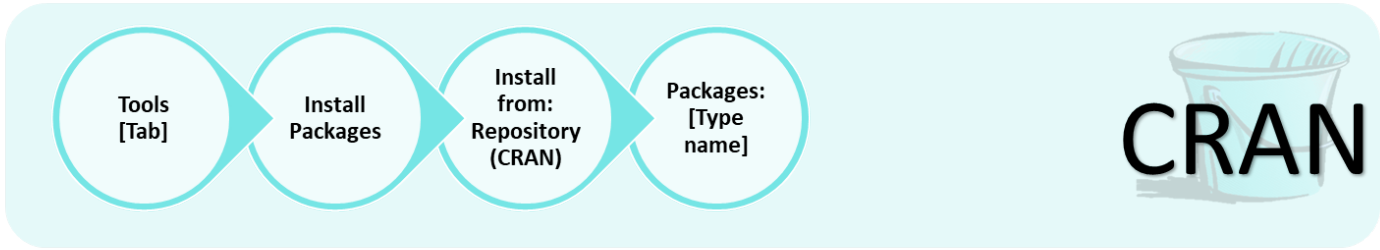
3. Launch RStudio

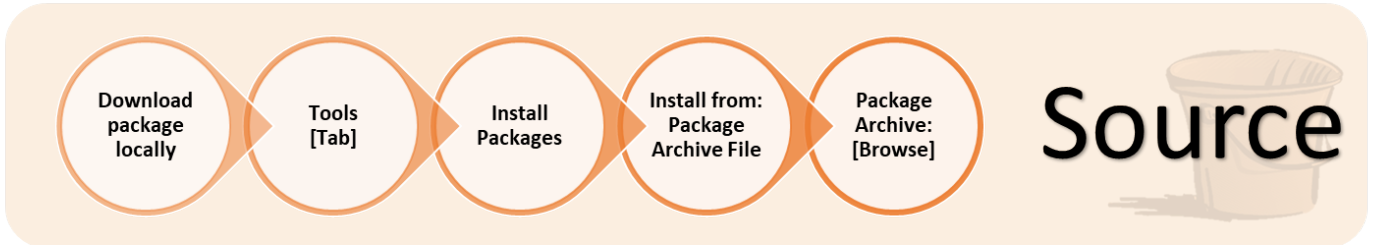
   ecommons@login01:~$ **srun -t 0-2:00 --pty -p interactive -c 1 --mem=2G --x11 –tunnel PORT:PORT RStudio_launcher.sh PORT**

Installing R packages - four general “bins”

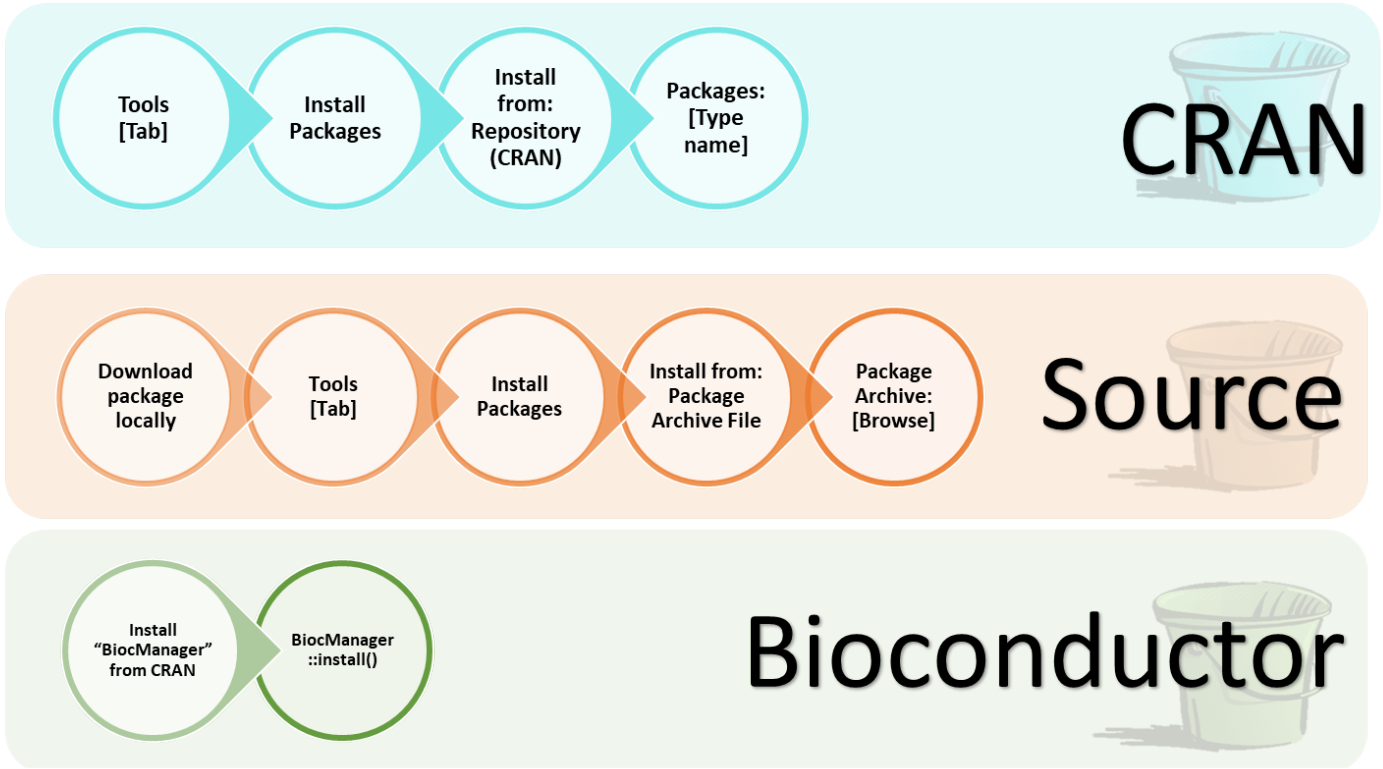


Installing R packages - four general “bins”

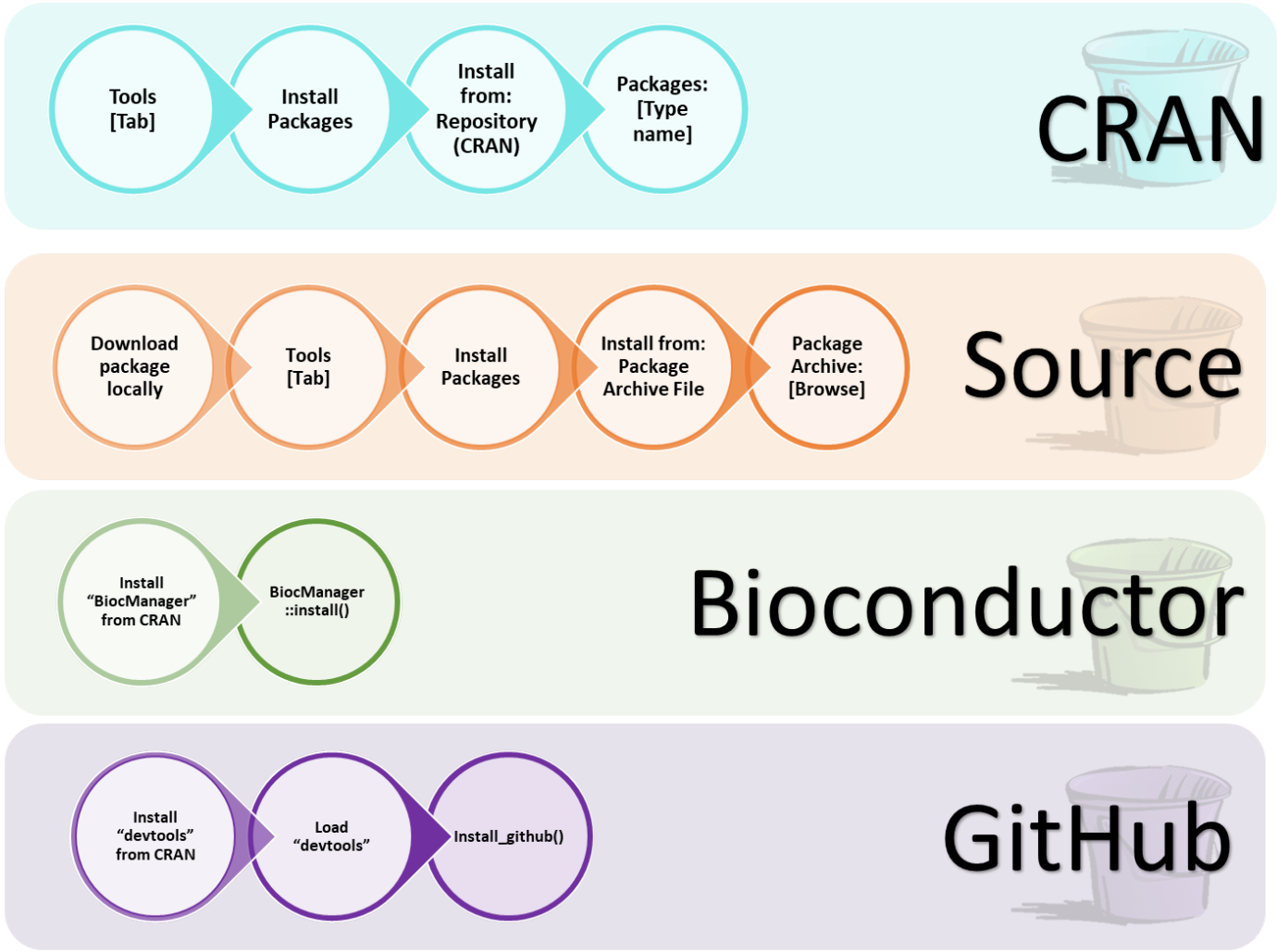




Installing R packages - four general “bins”



Installing R packages - four general “bins”



Exercise: Install and load the edgeR package from Bioconductor

- Install package from Bioconductor

  > BiocManager::install("edgeR")

- Load package

  > library("edgeR")

R documentation

- General R help on a function

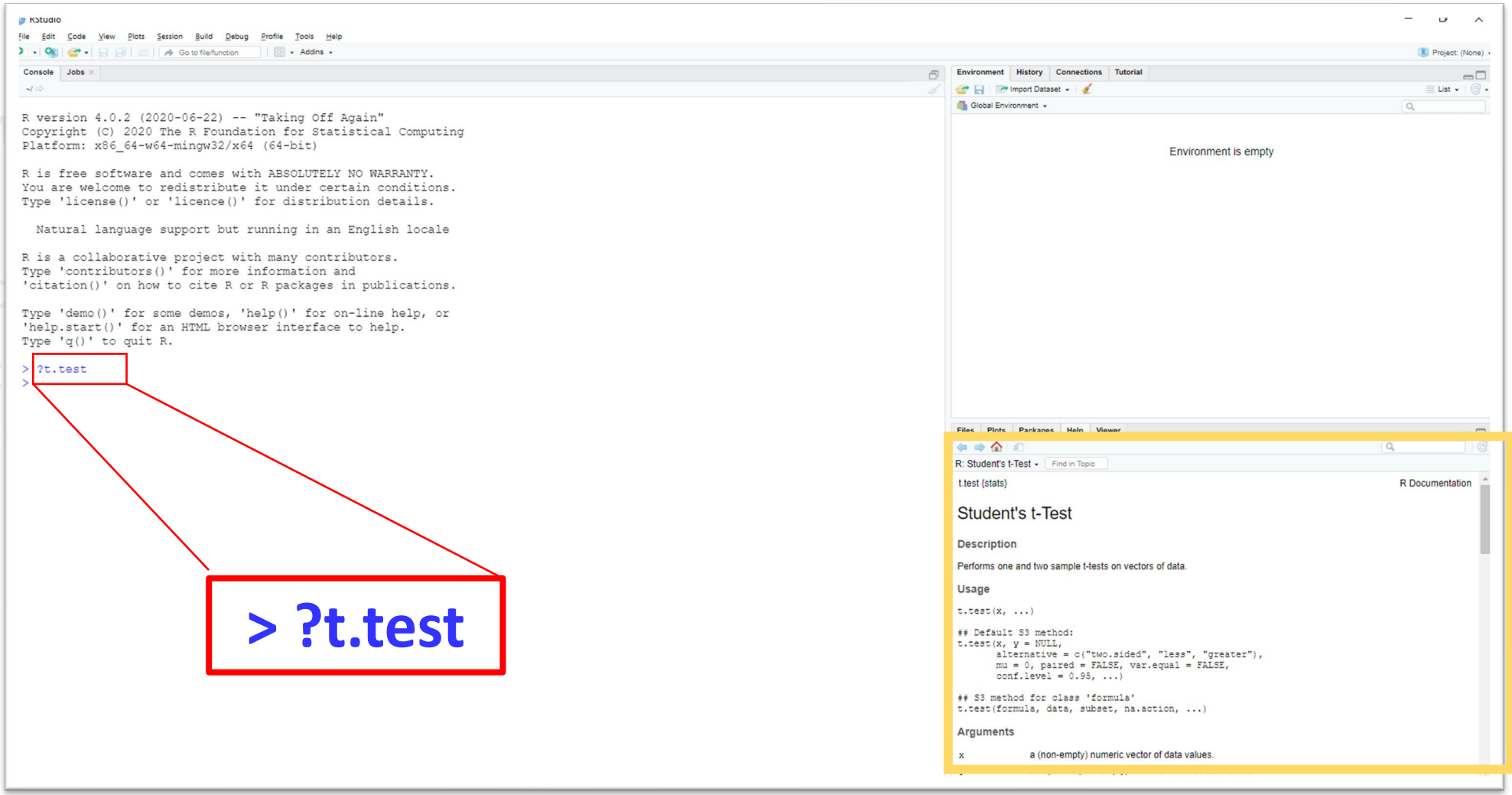
  ?name_of_function

- For example:

  > ?t.test

R documentation

- General R
  ?nam
- For examp
  > ?t.t



Setting your R “working directory”

- Returns filepath to current **w**orking **d**irectory:

  > getwd()

- Setting your working directory

  > setwd(“a/file/path/somewhere”)

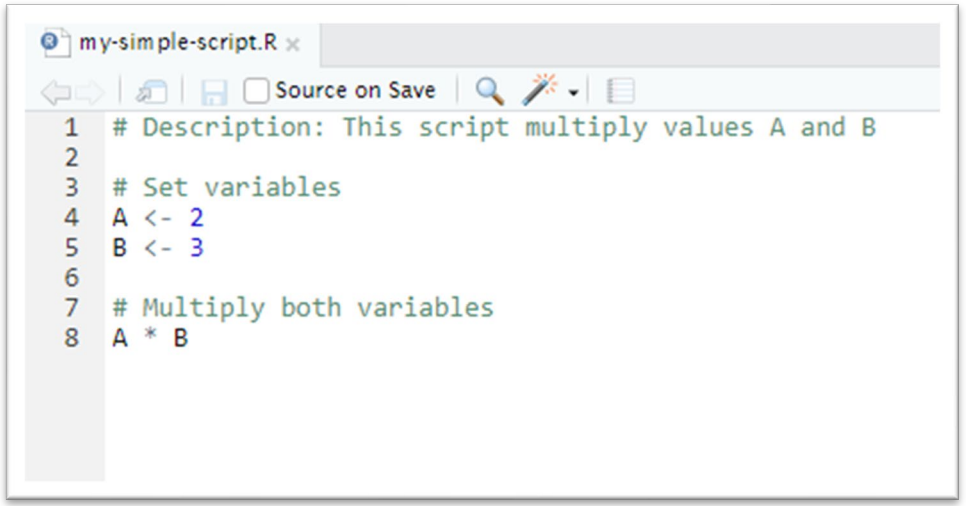
  -OR-

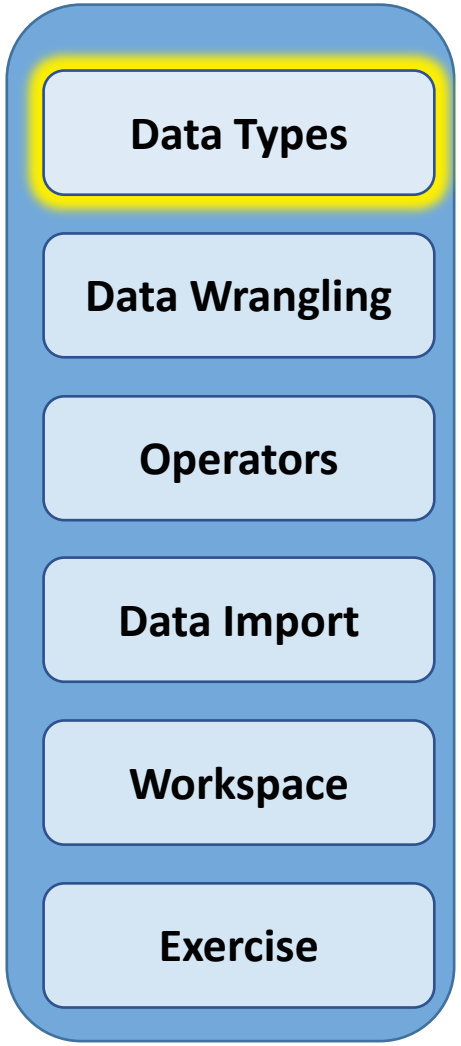
  ctrl + shift + H

Coding in R

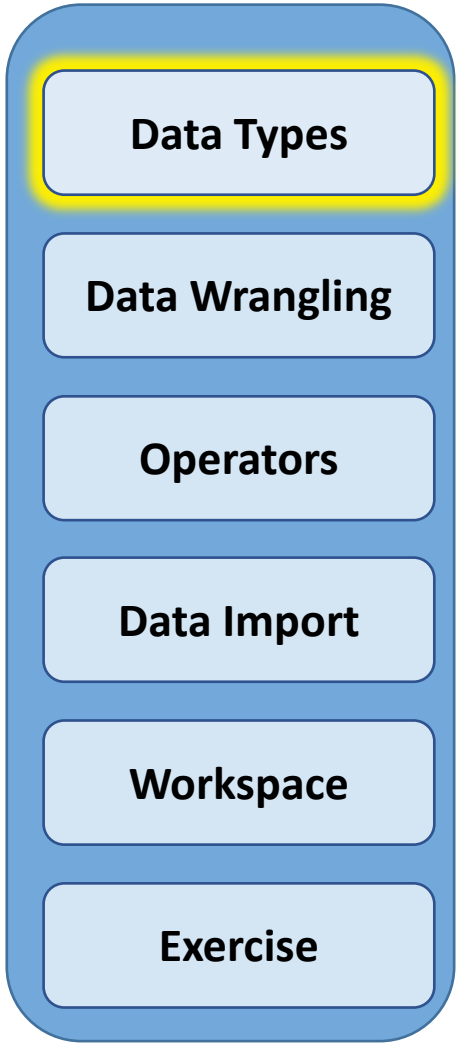
- Create an R script is easy!
- Start a line with “#” to create a comment
- Comments are important
- Assign variables with a “<-”
- Example:

> **myX <- 5**




Data Types
Data Wrangling
Operators
Data Import
Workspace
Exercise

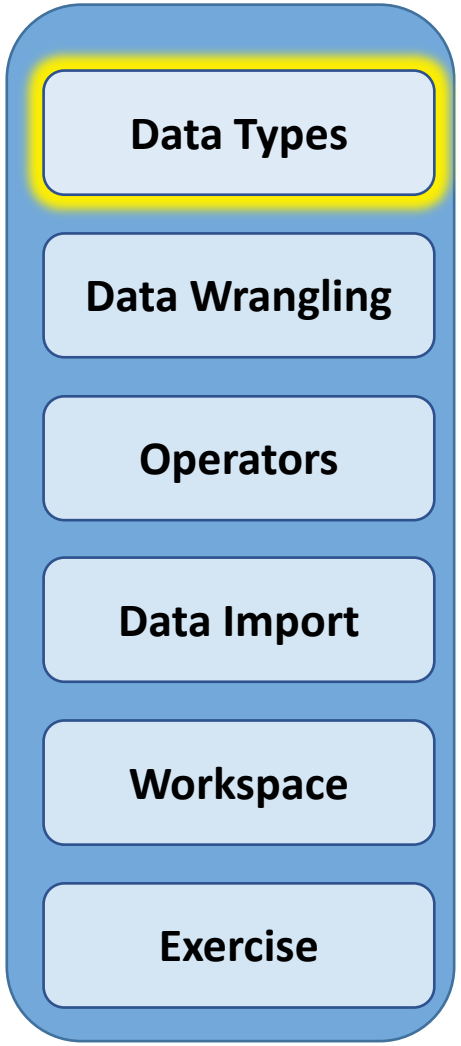
Vectors



- Basic way to store data
- “c” is used to create a vector
- Vectors types: numeric, character, & logical
- Example:

```
> myvector <- c(3,5,7)
```

Lists

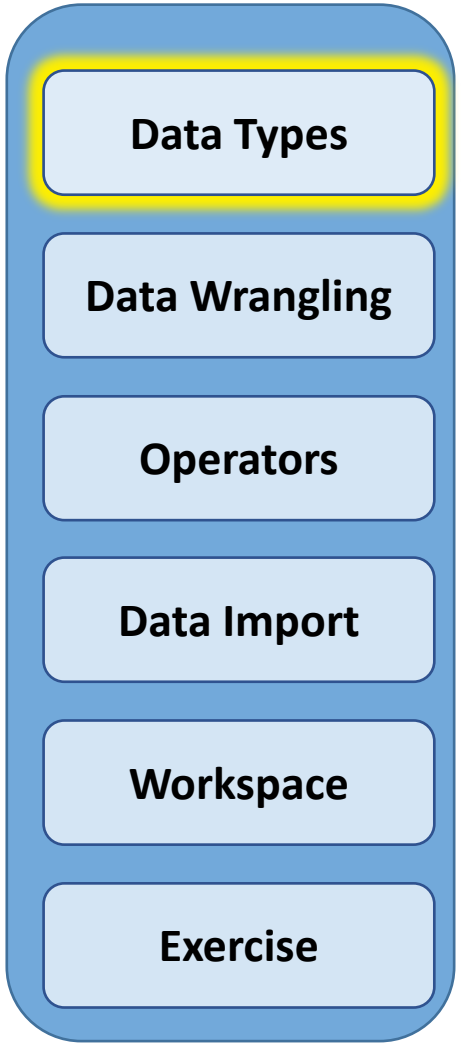


- Similar to vectors, but with mixed data types

- Example:

  > myvector <- c(3,“Tp53”,7)

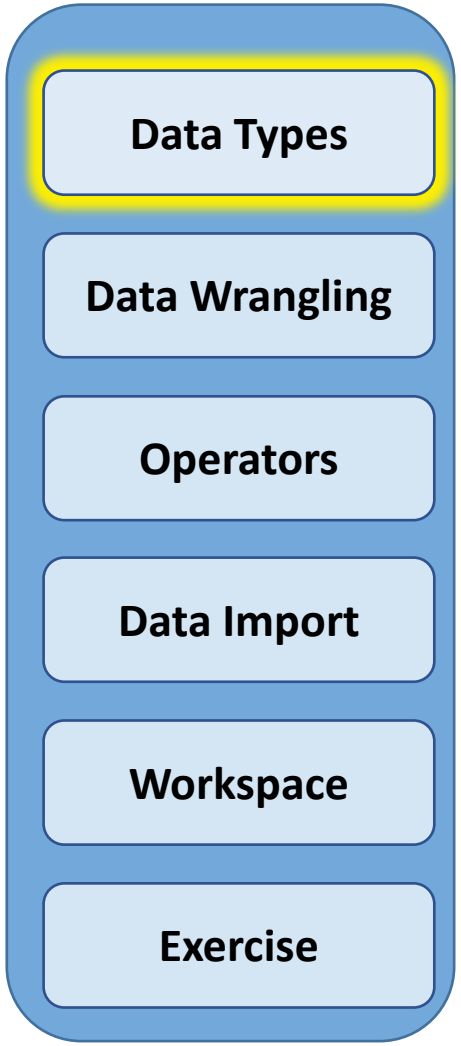
Factors



- Makes a vector nominal
- Mostly use in statistical modeling
- Numeric and character vectors can be made into factors
- Example:

```
> gender <- c("male", "male", "female", "female", "female")
> gender <- factor(gender)
```

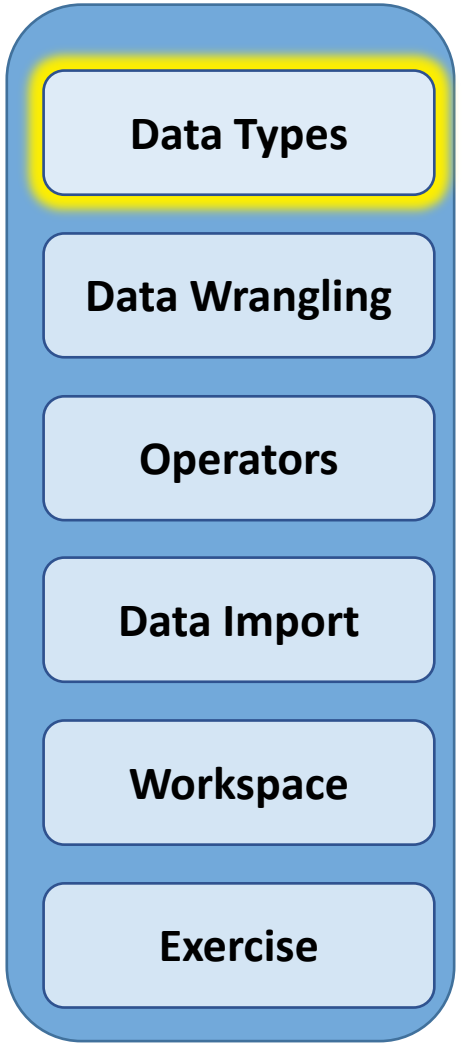
Matrices



- Data must be all the same type (i.e., numeric, character, or logical)
- Columns must have the same length
- Example:

```
> mymatrix <- matrix(c(1:6), nrow=3, ncol=2)
```

Data frame



- Subset of matrices allowing mixed types

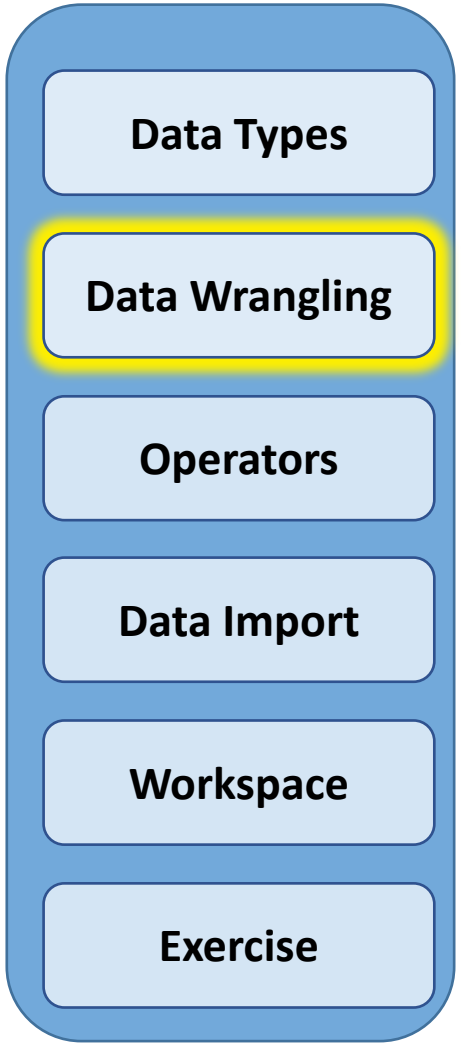
  > mydataframe <- as.data.frame(mymatrix)

- Columns can be named

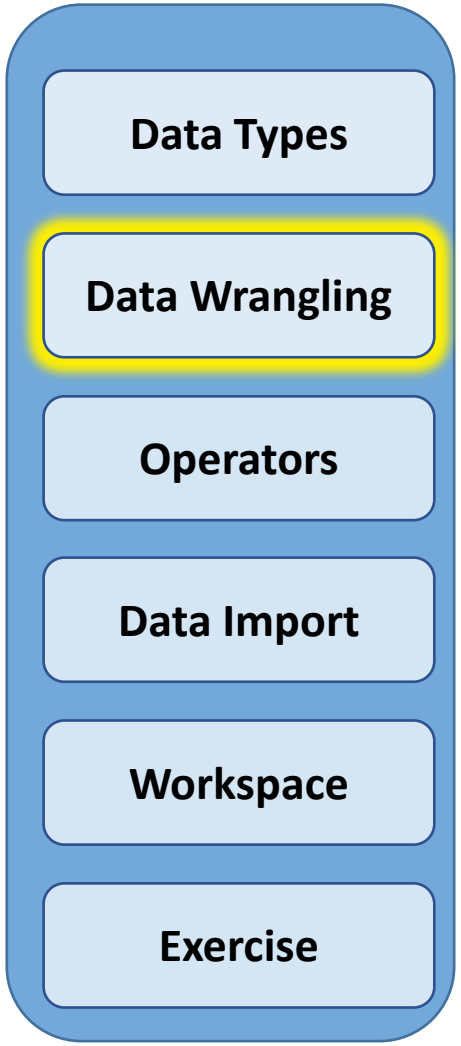
  > names(mydataframe) <- c("col1", "col2")

- Rows as well

  > row.names(mydataframe) <- mydataframe[,1]


Data Types
Data Wrangling
Operators
Data Import
Workspace
Exercise

Indexing

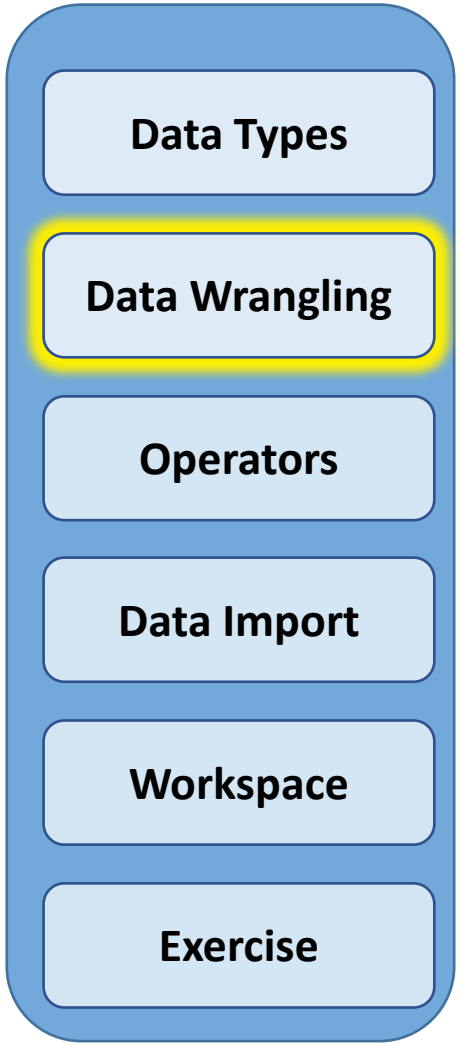


- Accessing elements

- Example:

```
> mymatrix <- matrix(c(1:6), nrow=3, ncol=2)
> mymatrix[1,2] #returns element in row 1 and column 2
> mymatrix[1,] #return all elements in row 1
> mymatrix[,1] #return all elements in column 1
```

Joining rows or columns



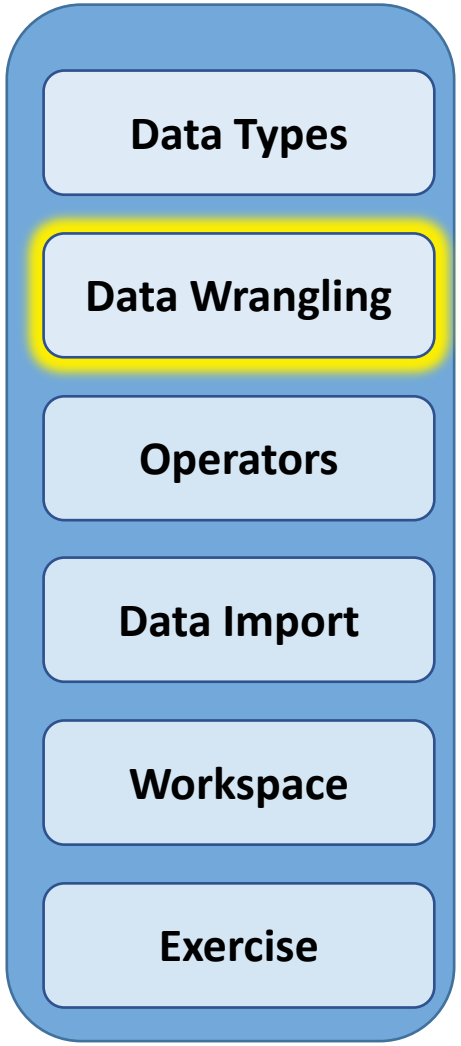
- **“rbind”** to add **r**ow(s) to a pre-existing data frame or matrix

  > mymatrix <- rbind(mymatrix, newrow)

- “**cbind”** to add column(s) to a pre-existing data frame or matrix

  > mymatrix <- cbind(mymatrix, newcol)

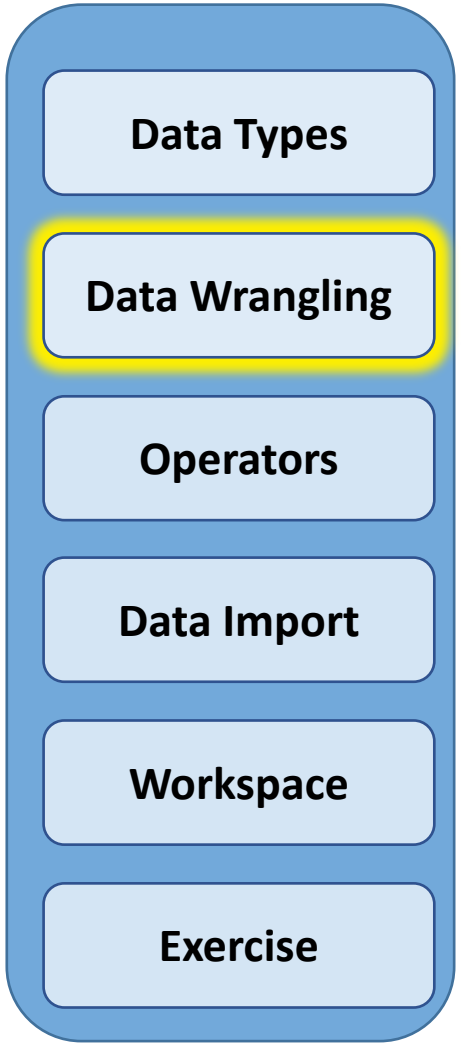
Missing values



- **NA**: Not Available
- **NaN**: Not a Number
- Example:

```
> is.na(x) #logical test for NA or NaN
> is.nan(x) #logical test for only NaN
> x[!is.na(x)] #subsets and excludes NAs
```

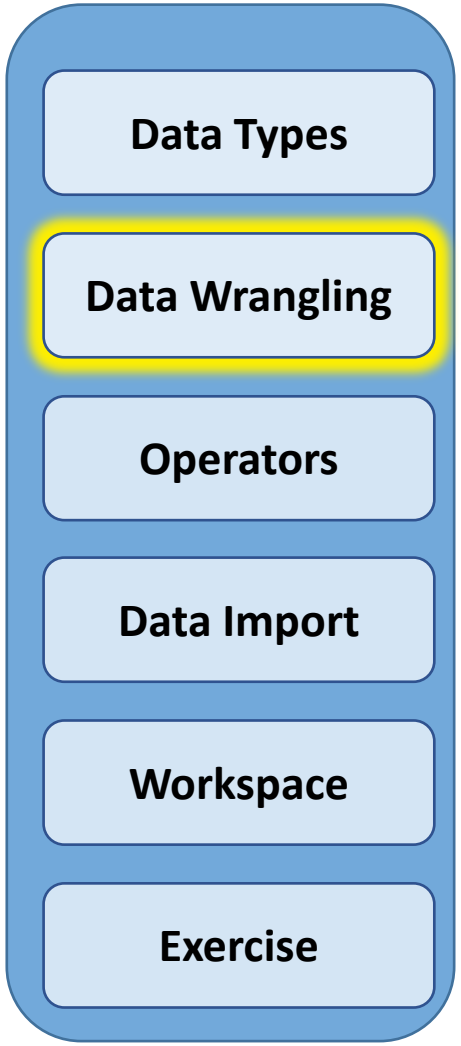
Change data type



- Functions start with “as.” followed by the type
- Example:

```
> myvector <- c(3,5,7)
> myvector <- as.character(myvector)
```

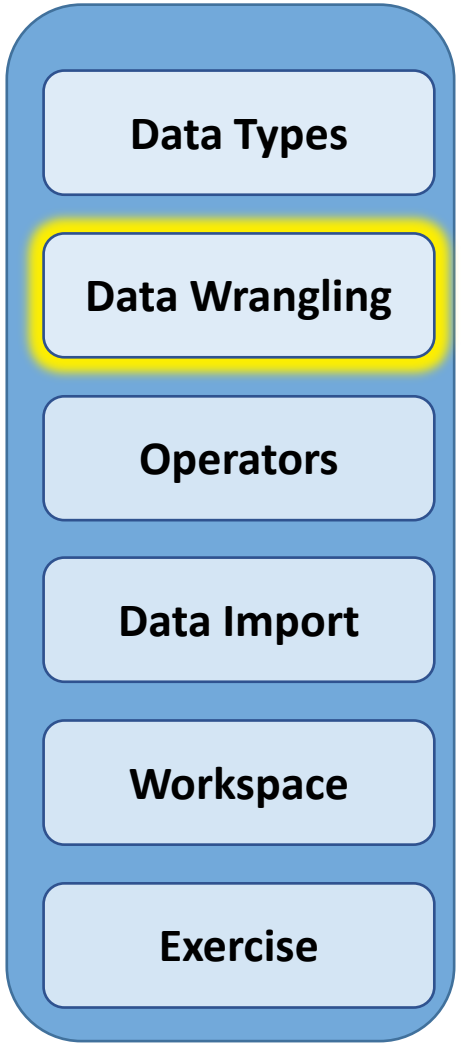
Apply function



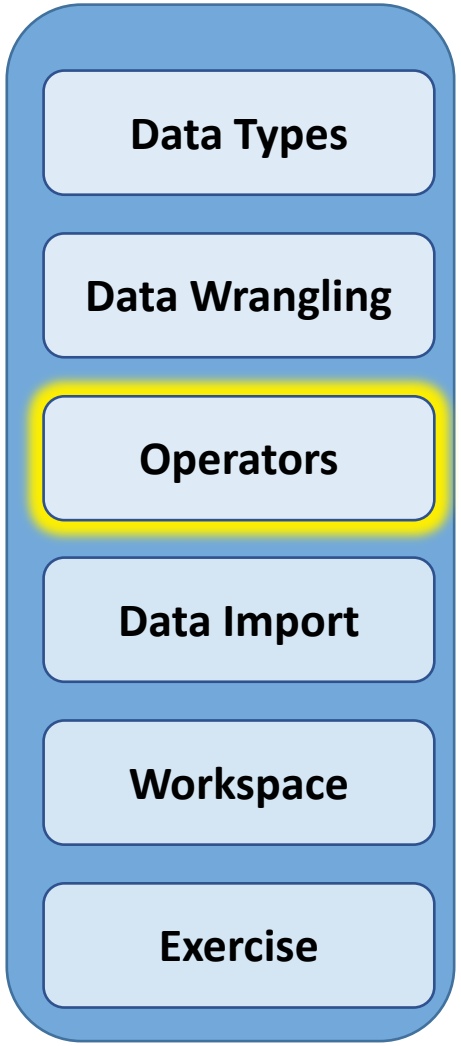
- *Returns a vector, array, or list of values obtained by applying a function to margins of an array or matrix*
- Format: **apply (to_what, *how*, *function*)**
- Where ***how*** accepts a "1" to apply the ***function*** over rows or "2" to apply over columns
- For example:

```
> apply(mymatrix, 1, sum) #row sums
> apply(mymatrix, 2, sum) #column sums
```

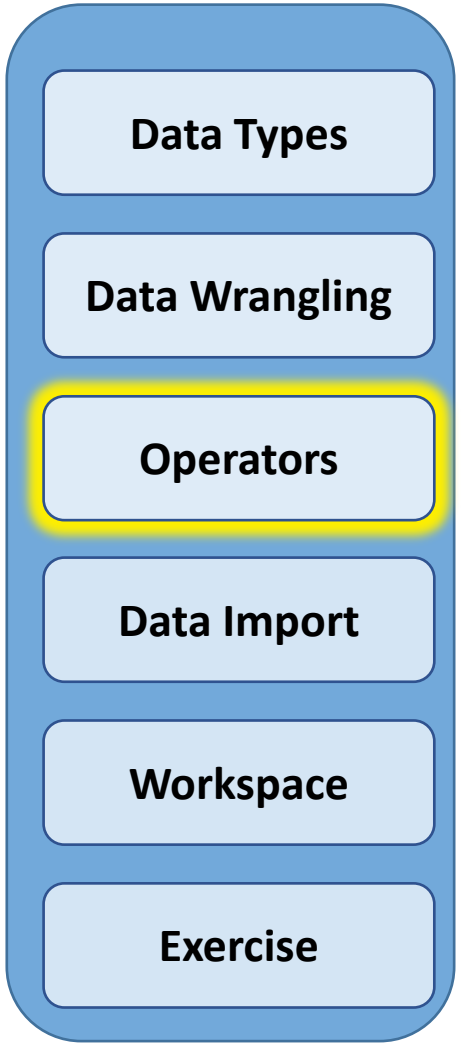
Useful functions



> **class(object)** #gives object class
> **mode(object)** #gives object type
> **length(vector)** #gives length
> **head(object)** #gives first 6 rows
> **tail(object)** #gives last 6 rows
> **summary()** #quick statistics
> **nrow(object)** #gives number of rows
> **ncol(object)** #gives number of columns
> **str(object)** #gives object structure
> **dim(object)** #gives matrix/df dimensions


Data Types
Data Wrangling
Operators
Data Import
Workspace
Exercise

Arithmetic

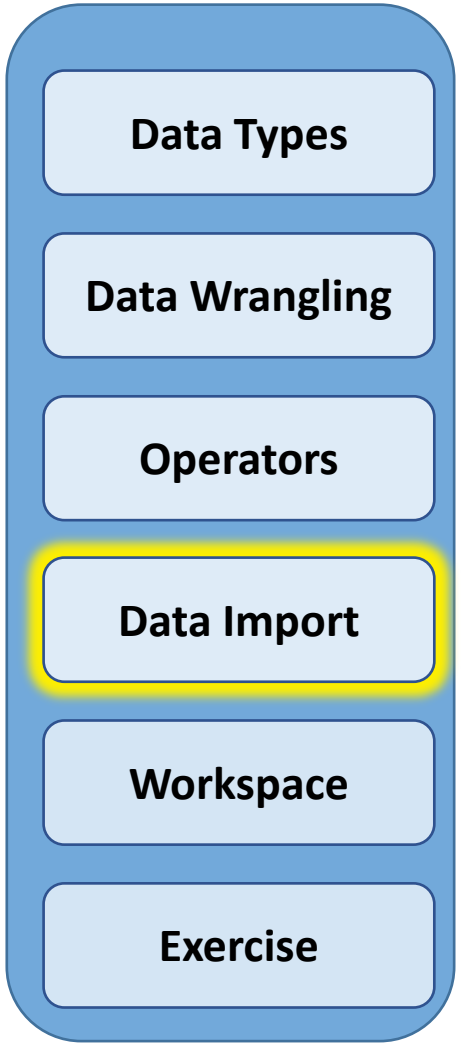


Operator	Description
+	Addition
-	Substraction
*	Multiplication
/	Division
^	Exponent
%%	Modulo (remainder)

Logical

- Data Types
- Data Wrangling
- Operators
- Data Import
- Workspace
- Exercise

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to
|	OR
&	AND


Data Types
Data Wrangling
Operators
Data Import
Workspace
Exercise

Text file

- Data Types
- Data Wrangling
- Operators
- **Data Import**
- Workspace
- Exercise

Format: **mydata <- read.table(file=" filename.csv", header=TRUE, sep=",")**

- "sep=" field separator character
- "header=" logical value to specify whether the file contain column names
- "row.names=" a vector of row names (must be unique identifiers!)

MS Excel

- R package is required (e.g., “xlsx”)

  > install.packages(“xlsx”)

  > library (“xlsx”)

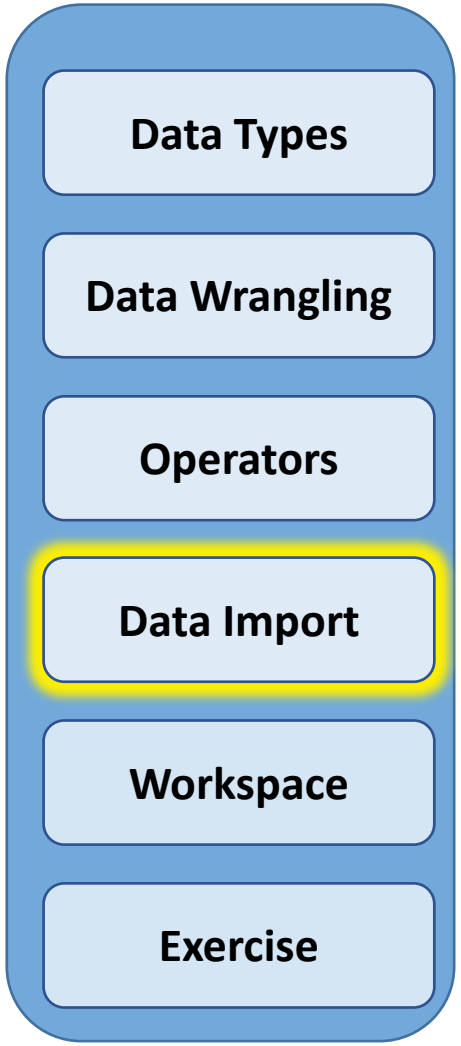
- Read in the first worksheet from the workbook myexcel.xlsx

  > **mydata <- read.xlsx(“FileName.xlsx”, sheetIndex=1)**

- Read in the worksheet named mysheet

  > **mydata <- read.xlsx(“FileName.xlsx", sheetName = "mysheet")**

Export



Format: **write.table(x="ObjectName", file="FileName.txt", sep="\t")**

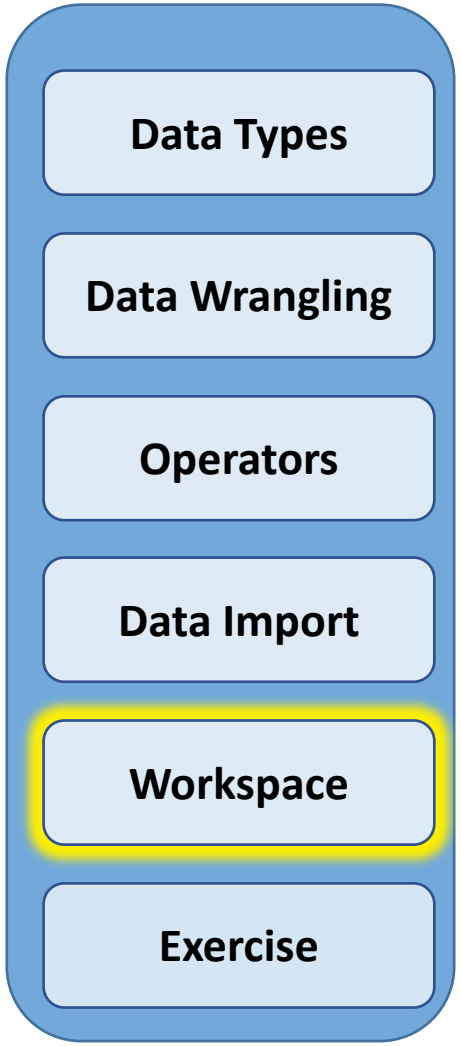
- Optional arguments:

  row.names=FALSE #turn off row names

  col.names=FALSE #turn off column names

  quote=FALSE #turn off character string quoting

Saving and Loading your workspace



- Save and pick up where you leave off – saves variables

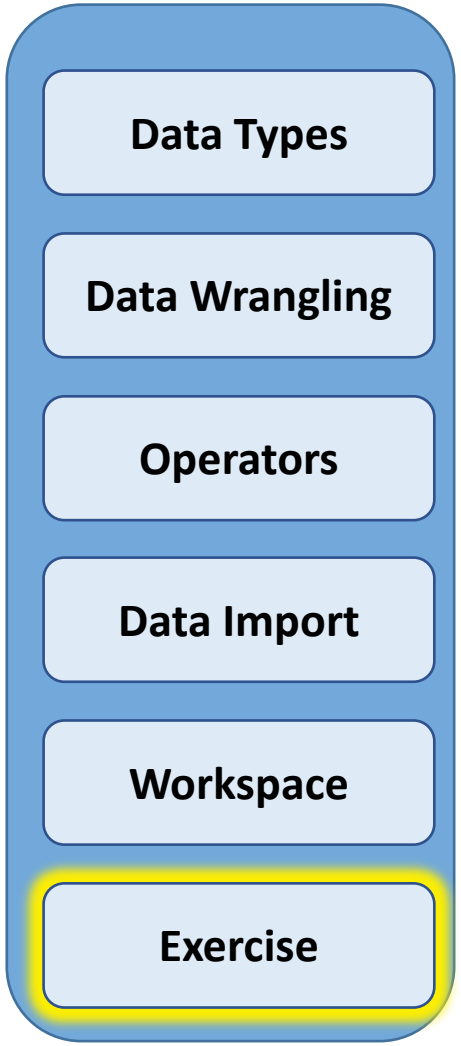
  Format: **save.image(file="FileName.RData")**

  -OR-

  Format: **save(object list, file="FileName.RData")**

- Load workspace

  Format: **load(file="FileName.RData")**


Data Types
Data Wrangling
Operators
Data Import
Workspace
Exercise

Import Data

- Data Types
- Data Wrangling
- Operators
- Data Import
- Workspace
- Exercise

- Import Rcoursetestdata1.csv using a comma separator, header set to true, and row names to first column

```
> mydf <- read.table("Rcoursetestdata1.csv", header=TRUE,  row.names=1, sep=",")
> head(mydf)
```

```
> head(mydf)
                TNBC1 TNBC2  TNBC3 Normal1 Normal2 Normal3
ENSG00000008988 15258 15077 144720   12095   43544   46883
ENSG00000009307 14660 20767   8678   13774   23030   18917
ENSG00000019582 50866 55775  15089    6696   13754   86319
ENSG00000026025 21174 47966  26682    6068   21126   12728
ENSG00000034510 25645 31574  56403   29590   25216   37199
ENSG00000044574 23910 27200  13757   13364   10852   12378
```

Basic Statistics

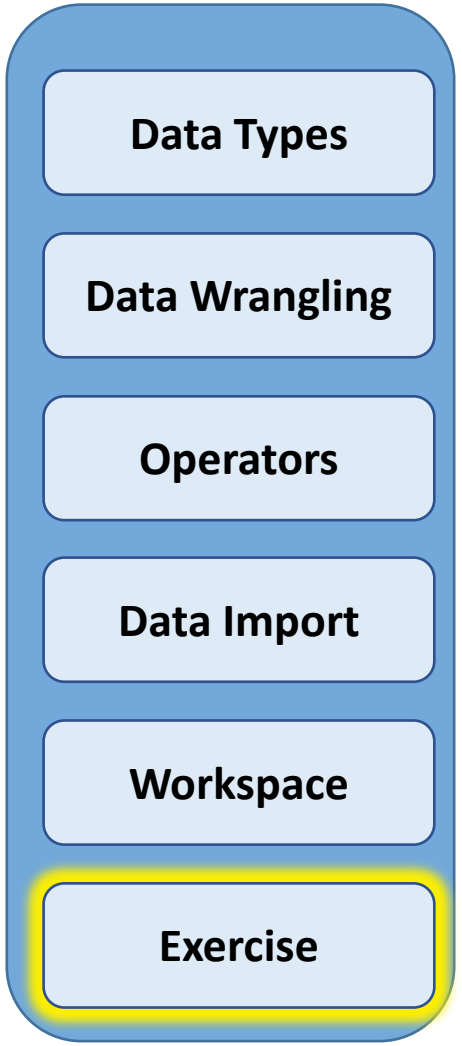
- Data Types
- Data Wrangling
- Operators
- Data Import
- Workspace
- Exercise

- Return basic statistics

  > summary(mydf)

```
> summary(mydf)
     TNBC1             TNBC2             TNBC3             Normal1
 Min.   :     0   Min.   :    65   Min.   :    31   Min.   :   22
 1st Qu.:  7888   1st Qu.:  9538   1st Qu.:  9324   1st Qu.: 5074
 Median : 13034   Median : 16568   Median : 19108   Median :10869
 Mean   : 18596   Mean   : 26036   Mean   : 25646   Mean   :14746
 3rd Qu.: 23850   3rd Qu.: 28194   3rd Qu.: 30389   3rd Qu.:18866
 Max.   :103007   Max.   :351603   Max.   :272582   Max.   :89837
    Normal2           Normal3
 Min.   :   208   Min.   :    15
 1st Qu.:  7124   1st Qu.:  8944
 Median : 14005   Median : 17710
 Mean   : 19425   Mean   : 25481
 3rd Qu.: 21576   3rd Qu.: 32191
 Max.   :212582   Max.   :244692
```

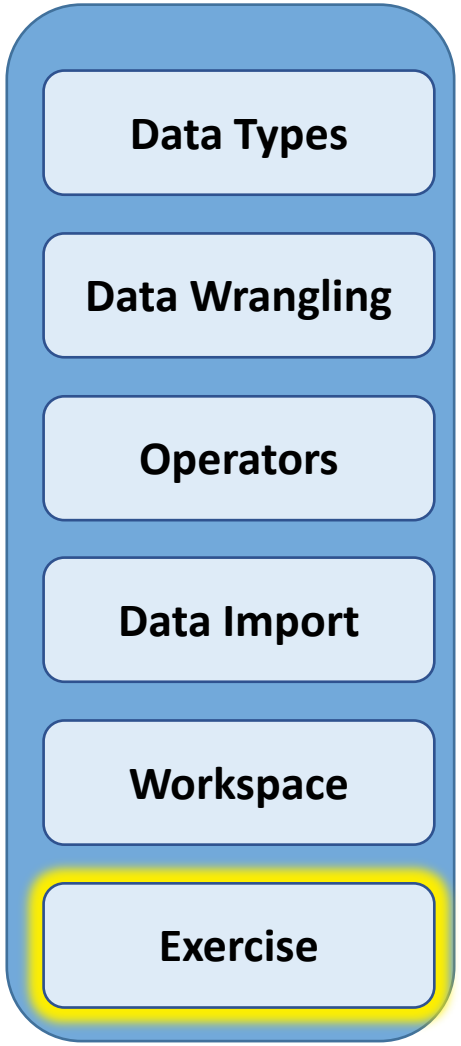
Transposing Data



- Need your data to read the other way? Turn it into a matrix, and transpose!
- For example:

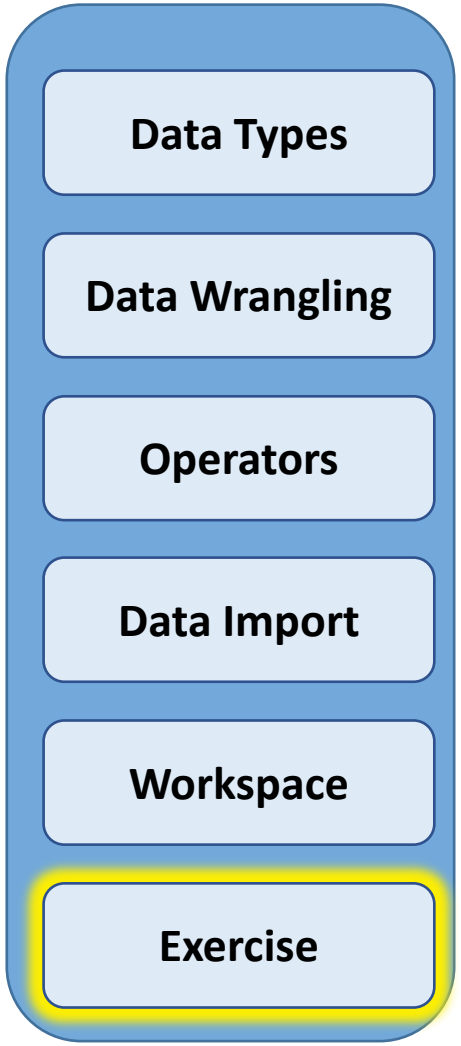
```
> mymatrix <- as.matrix(mydf)
> myTmatrix<- t(mymatrix) #t = transpose
> myTdf <- as.data.frame(myTmatrix) #as data frame again
```

Plotting with ggplot2



- To explore later
- Three general components
  - Data set
  - Coordinate system
  - Geoms

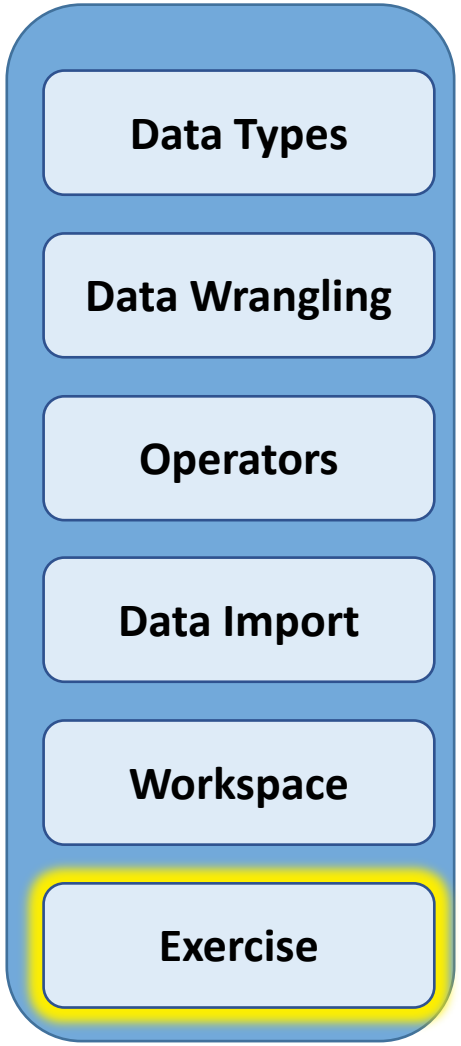
Plotting with ggplot2



- To explore later
- Three general components
  - Data set
  - Coordinate system
  - Geoms

```
> library ("ggplot2")

> data("midwest", package = "ggplot2")

> gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  labs(y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")
> gg
```

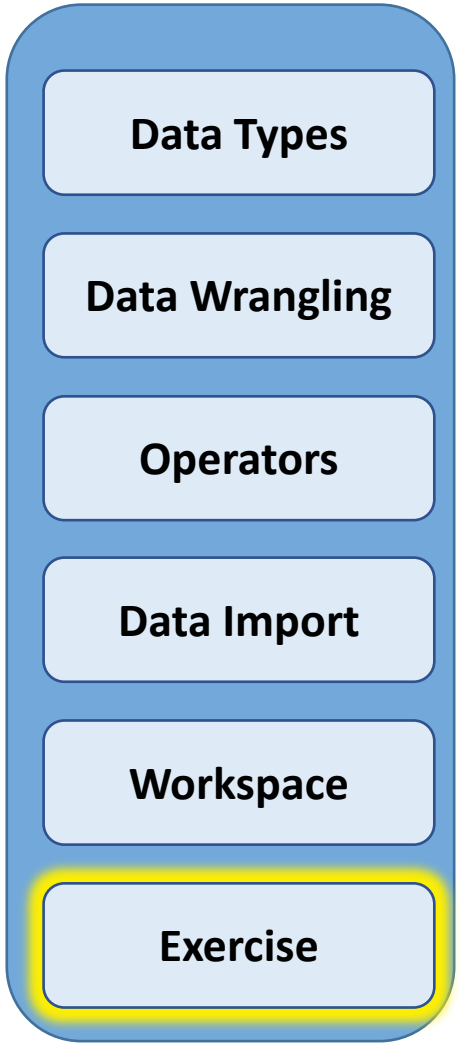
Plotting with ggplot2



- To explore later
- Three general components
  - Data set
  - Coordinate system
  - Geoms

```
> library ("ggplot2")

> data("midwest", package = "ggplot2")

> gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  labs(y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")
> gg
```

Plotting with ggplot2



- To explore later
- Three general components
  - Data set
  - Coordinate system
  - Geoms

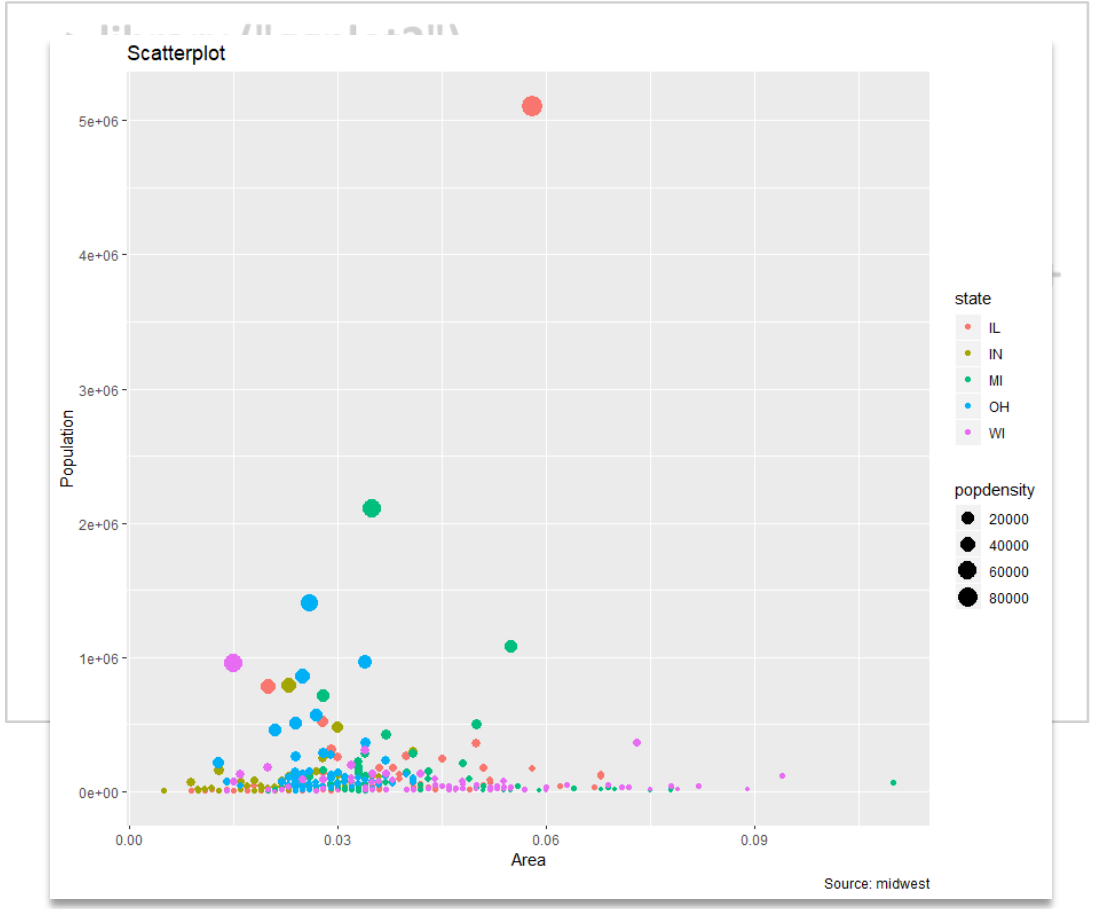
```
> library ("ggplot2")

> data("midwest", package = "ggplot2")

> gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  labs(y="Population",
       x="Area",
       title="Scatterplot",
       caption = "Source: midwest")
> gg
```

Plotting with ggplot2

- Data Types
- Data Wrangling
- Operators
- Data Import
- Workspace
- Exercise

- To explore later
- Three general components
  - Data set
  - Coordinate system
  - Geoms
- For more info:

  Chan Bioinformatics Core – [GitHub training class]



Contact

Email: rchelp@hms.harvard.edu

Website: https://it.hms.harvard.edu/our-services/research-computing

Wiki: https://wiki.rc.hms.harvard.edu/display/O2

Phone: 617-432-2000 (HMS IT Service Desk, 8a-5p)

Twitter: @hms_rc

Location: Gordon Hall 500, 5th Floor, 25 Shattuck Street

- https://rc.hms.harvard.edu/office-hours/ for Zoom web conferencing during remote work

Office hours: Wednesdays 1-3p for pressing needs, but appointments encouraged.