An Introduction to R

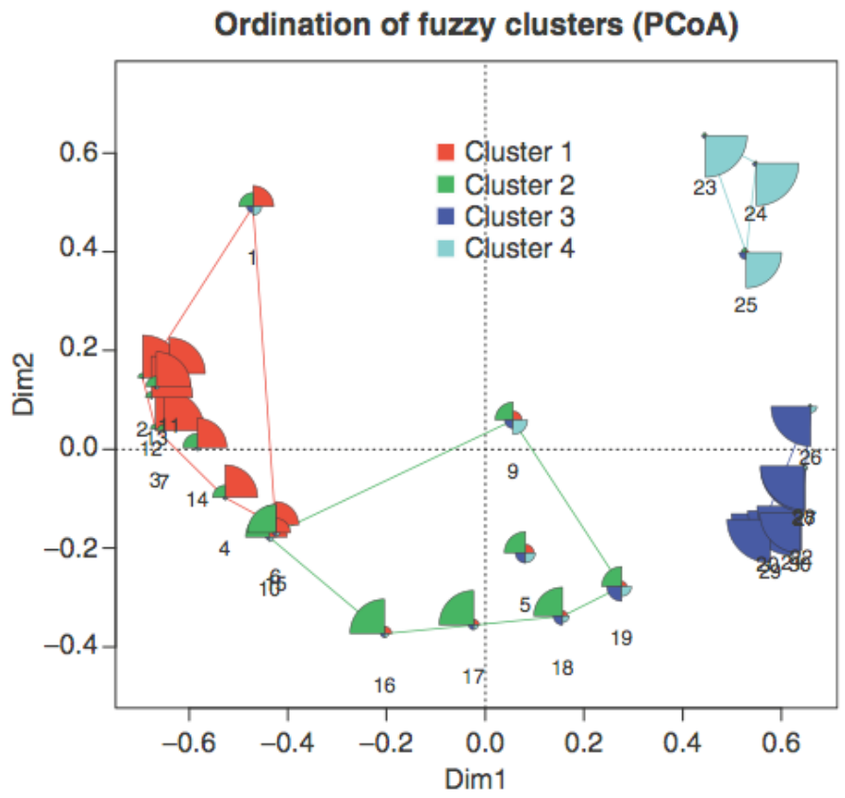
by Elin Videvall and Lokesh Mano

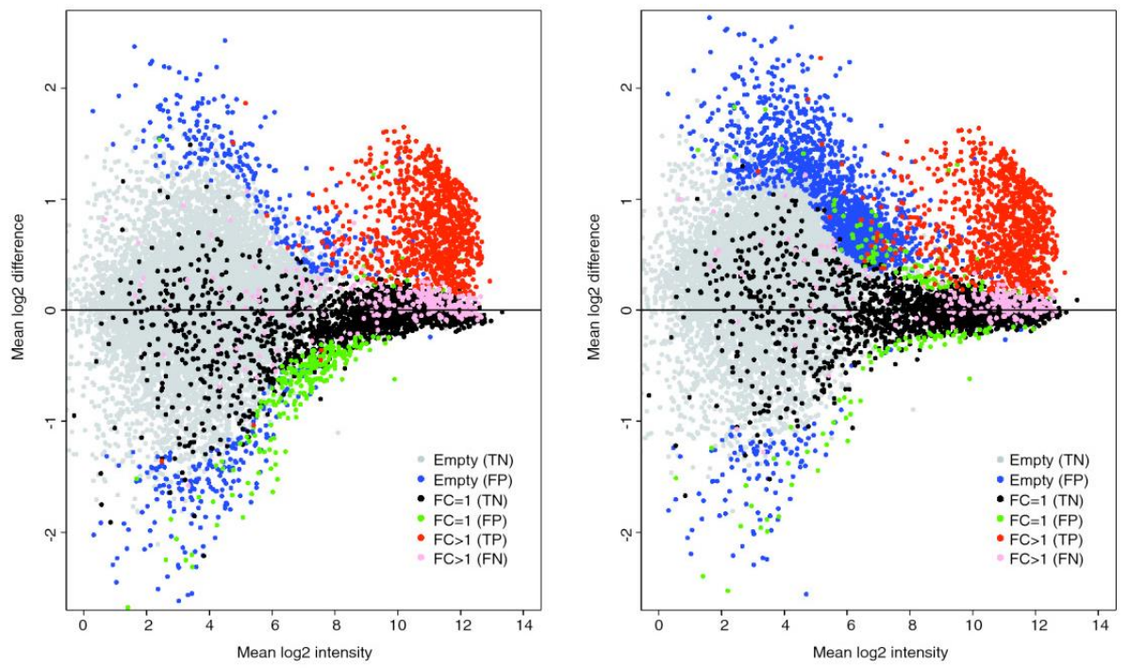
13 January, 2016

Slides courtesy of Dr Scott Handley

What is R?

A free software environment for statistical computing and graphics





Why is R so useful (and popular)?

- Data management and manipulation
- Well established system of packages and documentation
- Support for rich statistical simulation and modeling
- High-level interpreted language to prototype new computational methods
- Active development and dedicated community
- Cutting-edge graphical data visualization
- Free!

Where to learn more about R

- The R Project Homepage: http://www.r-project.org
- Quick R Homepage: http://www.statmethods.net
- Bioconductor: http://www.bioconductor.org
- An Introduction to R (long!): http://cran.r-project.org/doc/manuals/R-intro.html
- Google - there are tons of tutorials, guides, demos, packages and more

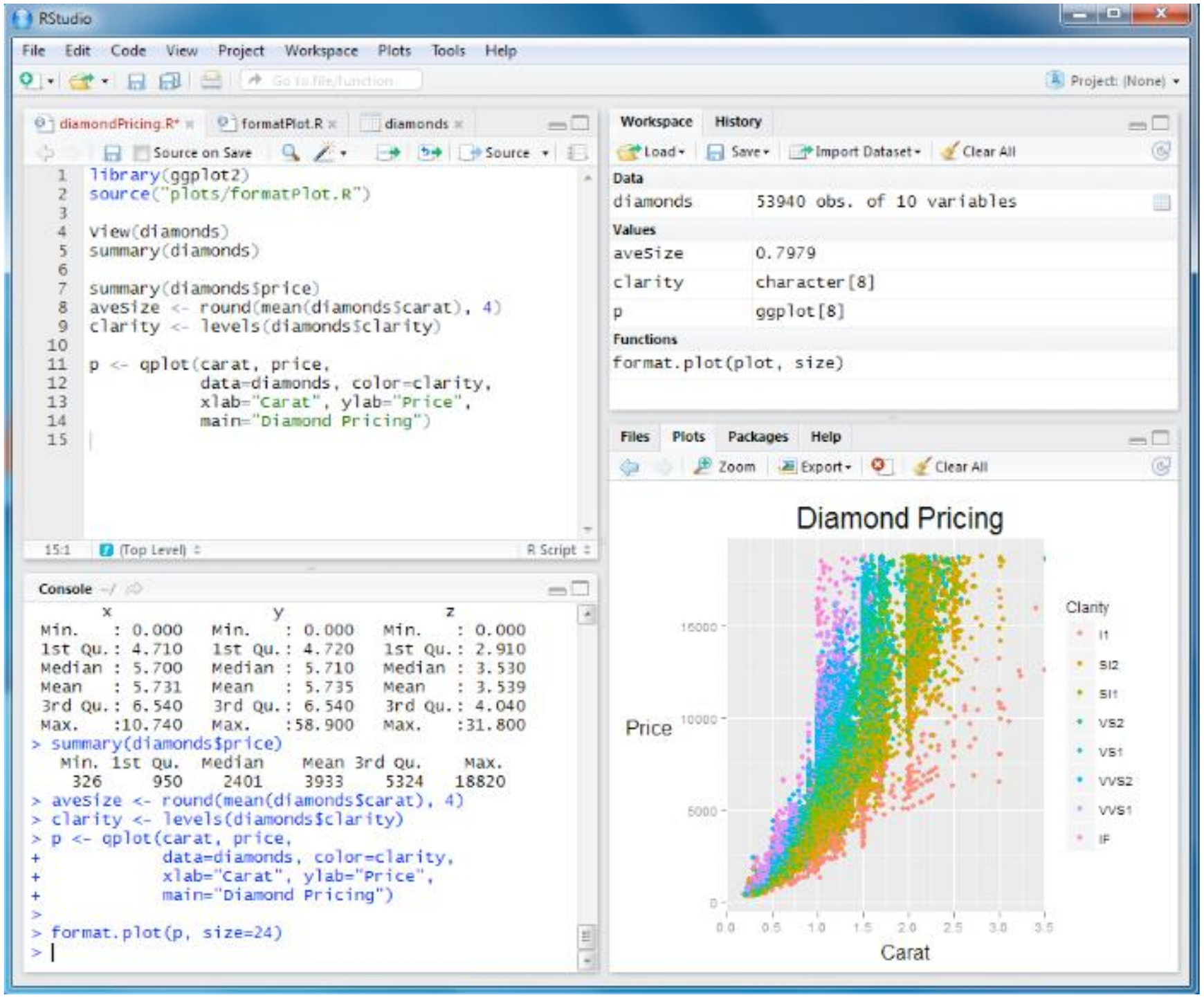
R for Biologists

- Bioconductor (http://bioconductor.org)
  - >1,100 packages:
    - Variant detection: coding changes, PolyPhen database
    - Annotation: pathway analysis, BioMart, GO, KEGG, NCBI and many others
    - High-throughput assays: flow cytometry, mass spec
    - Transcription factor binding detection, differential gene expression analysis
- Ecology (see: http://cran.r-project.org/web/views/Environmetrics.html)
    - Ordination
    - Cluster Analysis
    - Ecological Theory
    - Population Dynamics
    - Spatial Data Analysis
- Phylogenetics and Evolution (see: http://cran.r-project.org/web/views/Phylogenetics.html)
  - Ancestral State Reconstruction
  - Phylogenetic Inference
  - Trait Evolution

Running R

- Install an R Integrated Development Environment (IDE)
  - RStudio: http://www.rstudio.com
  - R Commander: http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/
  - Both projects can make working with R much easier, particularly for a new R user
    - Both IDE's run on Windows, Mac or Linux OS
- Or from the command line, type R

R Studio


RStudio
File Edit Code View Project Workspace Plots Tools Help
Project: (None)
diamondPricing.R* formatPlot.R diamonds
Source on Save
Source
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12 data=diamonds, color=clarity,
13 xlab="Carat", ylab="Price",
14 main="Diamond Pricing")
15
15:1 (Top Level) R Script
Workspace History
Load Save Import Dataset Clear All
Data
diamonds 53940 obs. of 10 variables
Values
aveSize 0.7979
clarity character[8]
p ggplot[8]
Functions
format.plot(plot, size)
Files Plots Packages Help
Zoom Export Clear All
Diamond Pricing
Price
Carat
Clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Console ~/
x y z
Min. : 0.000 Min. : 0.000 Min. : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean : 5.731 Mean : 5.735 Mean : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max. :10.740 Max. :58.900 Max. :31.800
> summary(diamonds$price)
Min. 1st Qu. Median Mean 3rd Qu. Max.
326 950 2401 3933 5324 18820
> avesize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+ data=diamonds, color=clarity,
+ xlab="Carat", ylab="Price",
+ main="Diamond Pricing")
>
> format.plot(p, size=24)
>

Basic R functionality

Calculator

- +, -, /, *, ^, log(), exp(), sqrt(), abs(), cos(), sin(), tan(), ...

```
(4+5^2)/3.14
[1] 9.235669
```

Set Variables / Vectors

<- or =

```
y <- 13.4
y
[1] 13.4
```

```
y <- c(1,2,3,4,5)
y
[1] 1 2 3 4 5
```

Sequences

```
y <- rep(2,10)
[1]  2 2 2 2 2 2 2 2 2 2
```

```
y <- 2:8
[1]  2 3 4 5 6 7 8
```

Statistics

```
t.test(7:34, 5:29)
```

```
t = 1.6348, df = 50.999, p-value = 0.1082
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.797982  7.797982
sample estimates:
mean of x mean of y
     20.5      17.0
```

Manipulation I

```
n <- c(3, 7, 12, 50, 103)
```

`n[4]`	`[1] 50`
`n[-2]`	`[1] 3 12 50 103`
`n[1:3]`	`[1] 3 7 12`
`n[c(1,3,5)]`	`[1] 3 12 103`
`n[n < 50]`	`[1] 3 7 12`
`n[n > 8 & n != 50]`	`[1] 12 103`

Manipulation II

```
n <- c(3, 7, 12, 50, 103)
```

```
n+1
```

```
[1] 4 8 13 51 104
```

```
sum(n)
```

```
[1] 175
```

```
mean(n)
```

```
[1] 35
```

```
var(n)
```

```
[1] 1796.5
```

```
min(n)
```

```
[1] 3
```

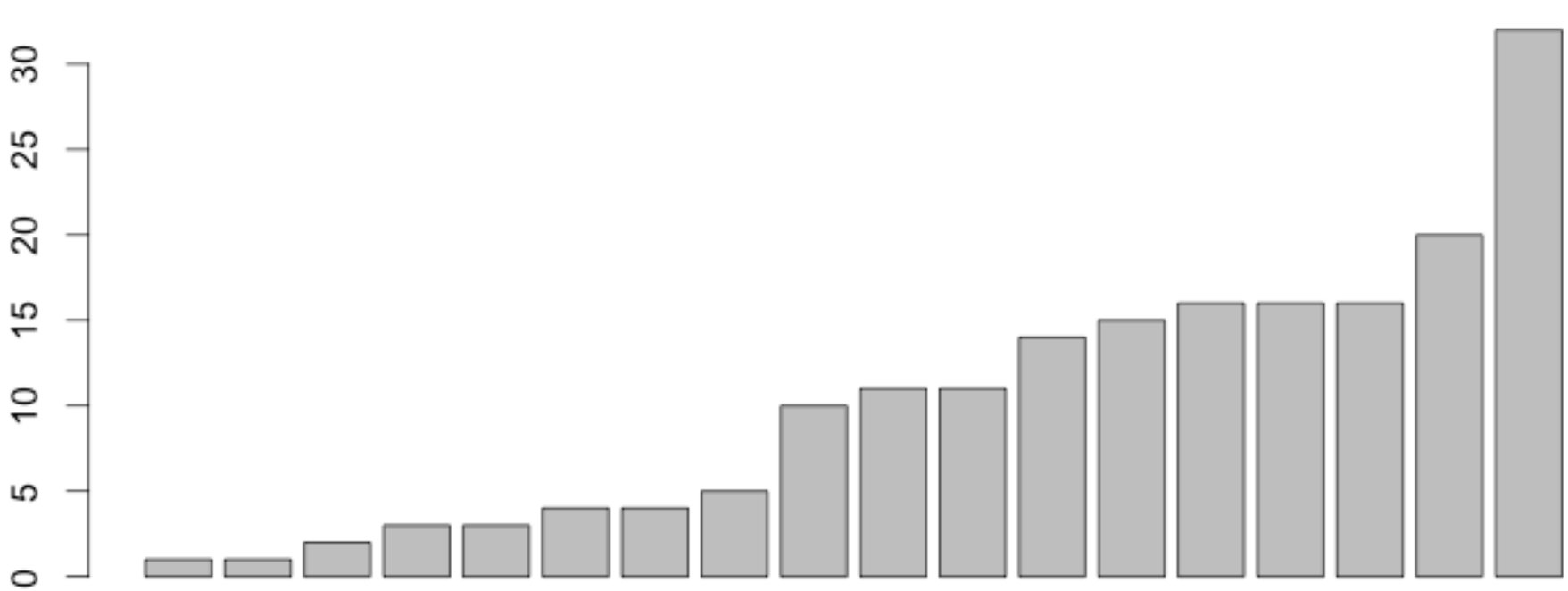
```
max(n)
```

```
[1] 103
```

Basic Visualization I

```
y <- c(1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

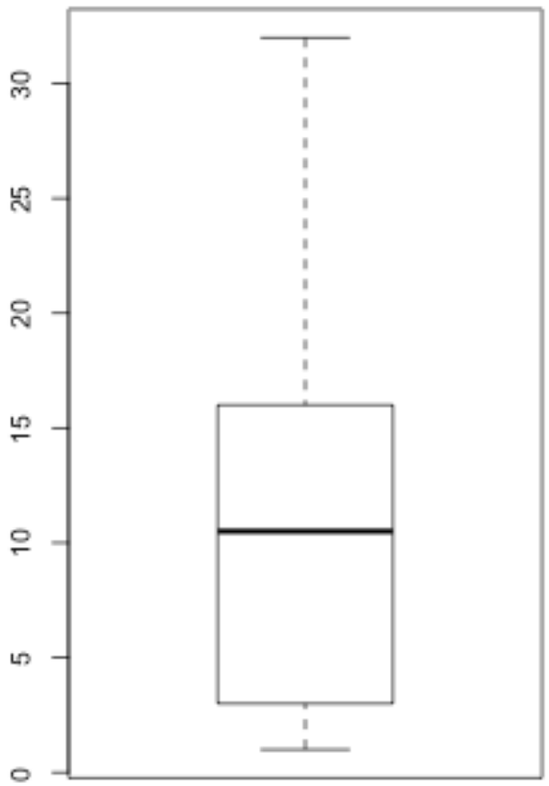
```
barplot(y)
```



Basic Visualization II

```
y <- c(1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

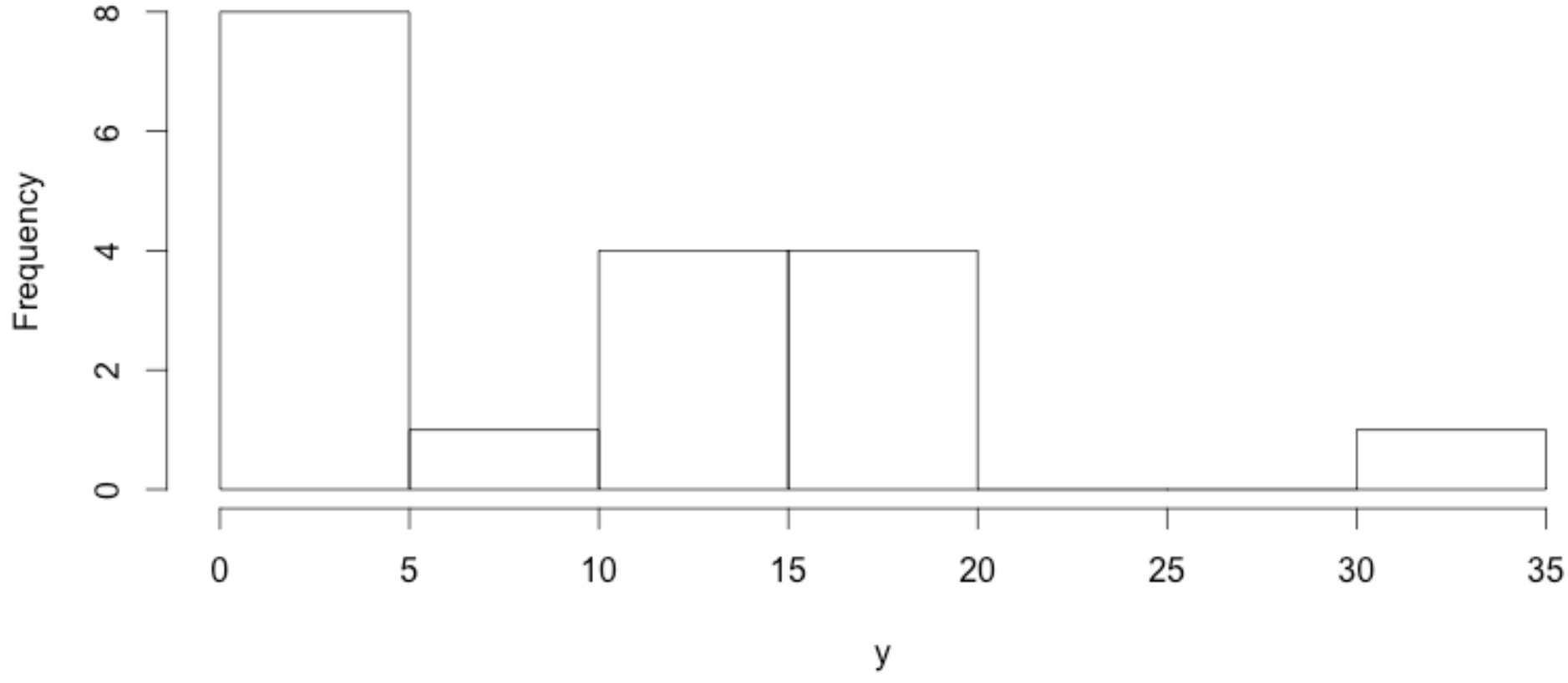
```
boxplot(y)
```



Basic Visualization III

```
y <- c(1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

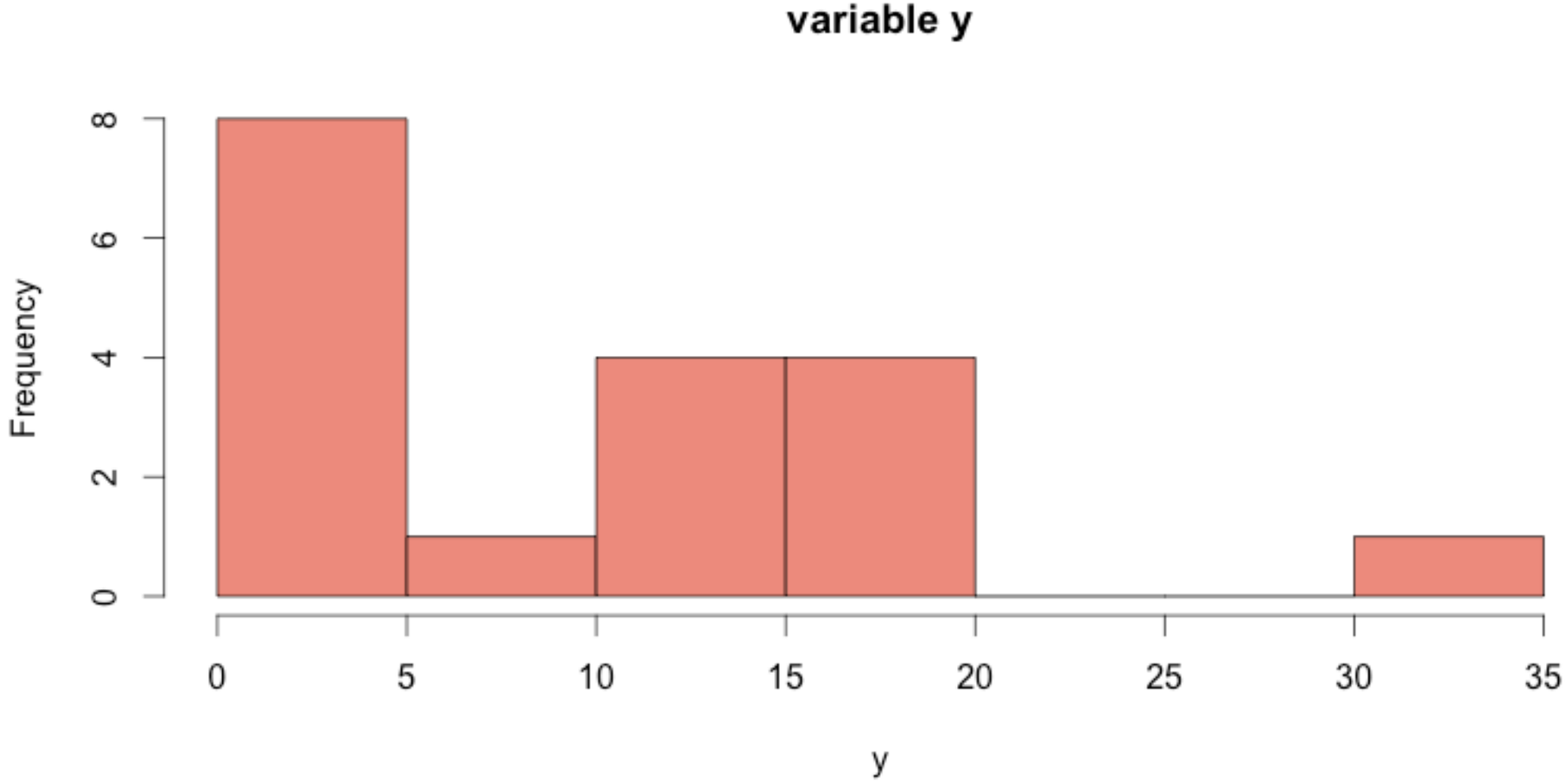
```
hist(y)
```



Basic Visualization III.ii

```
y <- c(1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

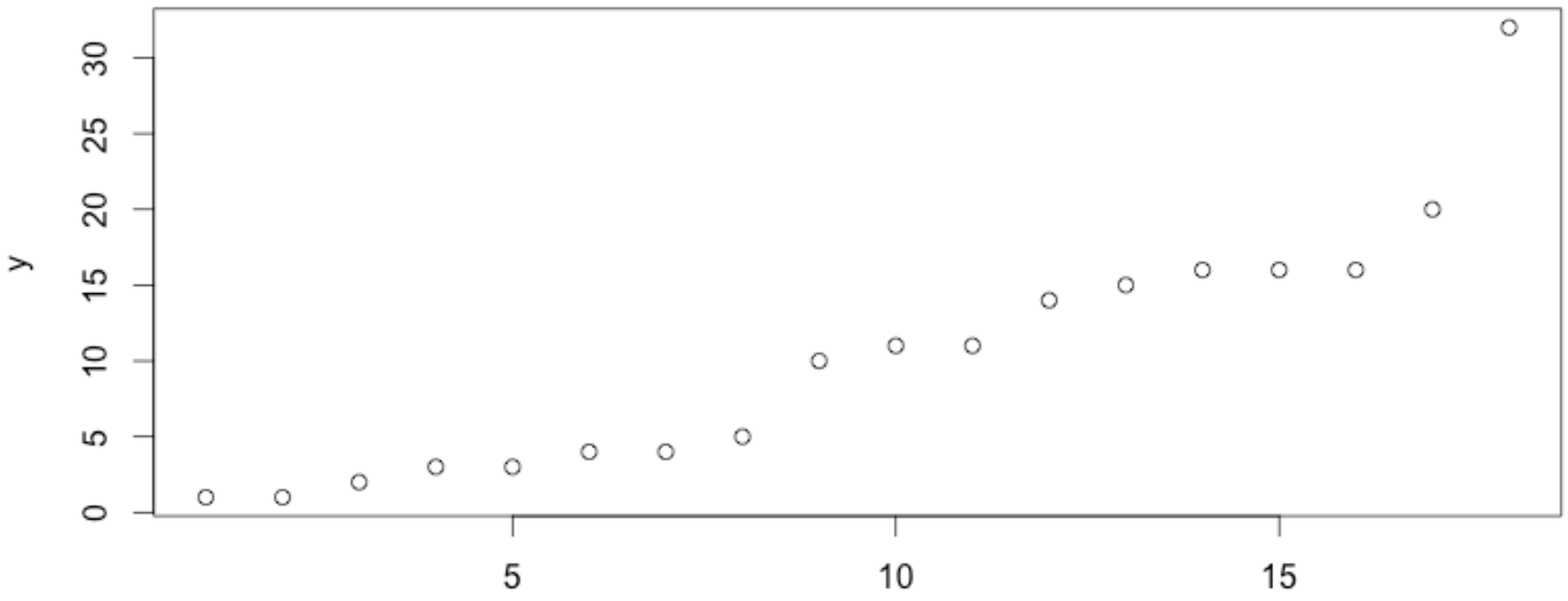
```
hist(log(y), col="salmon", main="variable y")
```



Basic Visualization IV

```
y <- (1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

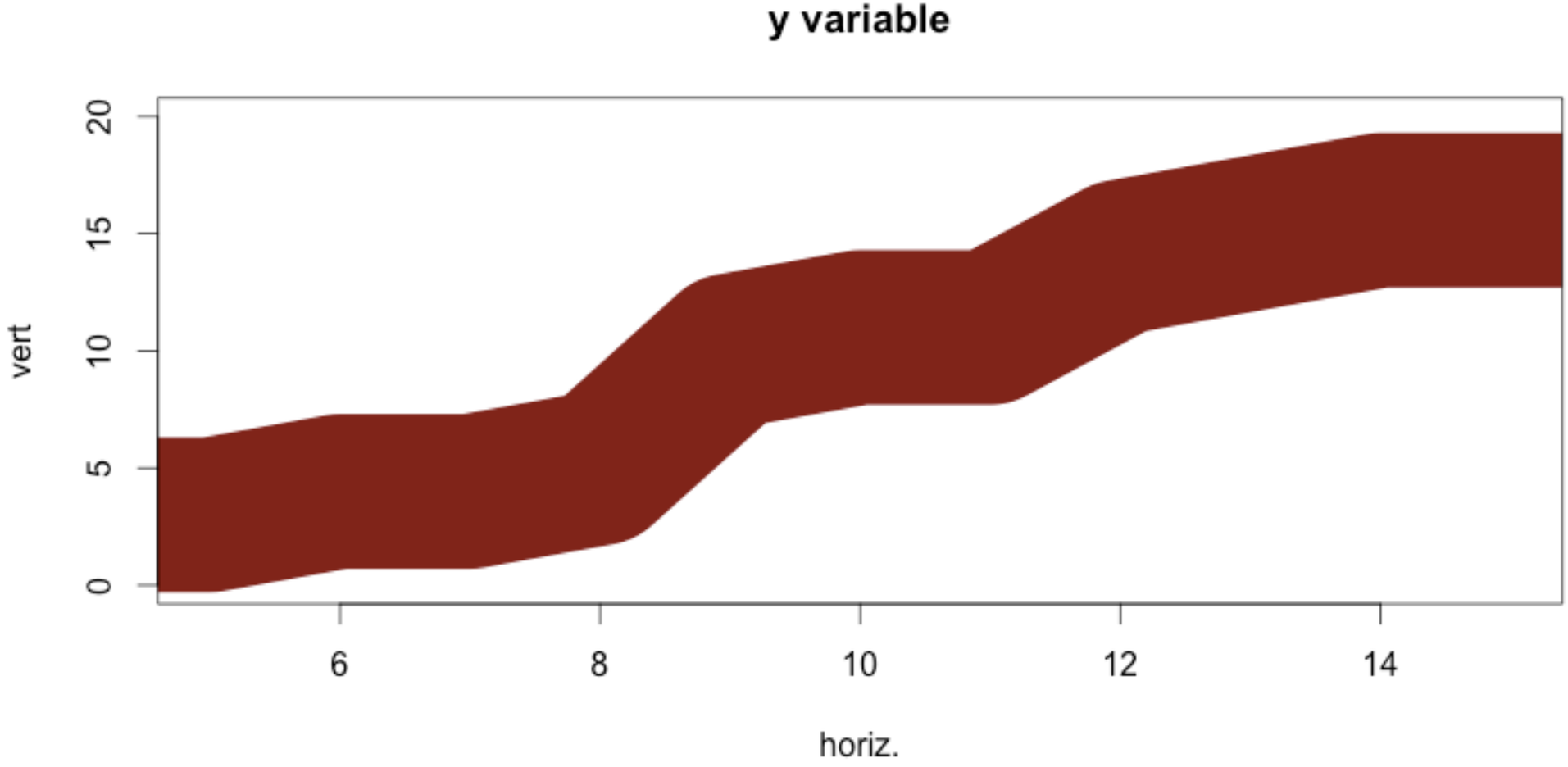
```
plot(y)
```



Basic Visualization IV.ii

```
y <- (1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)
```

```
plot(y, type="l", col="dark red", lwd=100, main="y variable", ylim=c(0,20),
xlim=c(5,15), ylab="vert", xlab="horiz.")
```



Help

- Do you need to remember all of the variables?
- ? is your friend
- ?plot

plot {graphics}

Generic X-Y Plotting

Description

Generic function for plotting of R objects. For more details about the graphical parameter arguments, see par.

For simple scatter plots, plot.default will be used. However, there are plot methods for many R objects, includi
functions, data.frames, density objects, etc. Use methods(plot) and the documentation for these.

Usage

```
plot(x, y, ...)
```

R Documentation

`type`

what type of plot should be drawn. Possible types are

- "p" for **p**oints,
- "l" for **l**ines,
- "b" for **b**oth,
- "c" for the lines part alone of "b",
- "o" for both '**o**verplotted',
- "h" for '**h**istogram' like (or 'high-density') vertical lines,
- "s" for stair **s**teps,
- "S" for other **s**teps, see 'Details' below,
- "n" for no plotting.

Data Frames

- A data.frame is essentially a table

columns can be mixes types

numeric, text strings

	clostridia	proteobacteria	bacteroides
01_healthy	22	54	245
02_healthy	26	65	265
03_healthy	34	66	262
01_sick	32	32	116
02_sick	12	24	101
03_sick	9	18	87

data.frame[-1,-2]

	clostridia	bacteroides
02_healthy	26	265
03_healthy	34	262
01_sick	32	116
02_sick	12	101
03_sick	9	87

Data Frame Manipulations

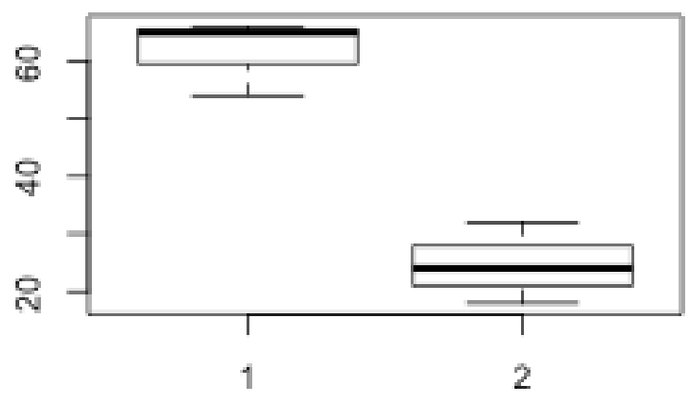
	clostridia	proteobacteria	bacteroides
01_healthy	22	54	245
02_healthy	26	65	265
03_healthy	34	66	262
01_sick	32	32	116
02_sick	12	24	101
03_sick	9	18	87

data.frame$proteobacteria [1] 54 65 66 32 24 18

t.test(data.frame$proteobacteria[1:3], data.frame$proteobacteria[4:6])

p-value = 0.002725

boxplot(data.frame$proteobacteria[1:3],
data.frame$proteobacteria[4:6])



Basic R Mechanics

R Mechanics - Installing & Loading Packages

Installing regular R packages:
install.packages(“vegan”)

Bioconductor packages:
source(“http://bioconductor.org/biocLite.R”)
biocLite(“DESeq2”)

Loading packages in R:
library(“vegan”)
library(“DESeq2”)

Getting Data Into and Out of R

Step 1: **Se**t **W**orking **D**irectory

setwd(~/scott/data/R)

Step 2: Read in some data

bacteria <- read.table(“bacterial_table.txt”)

Step 3: Work with data

bacteria_2 <- bacteria[-2]

Step 4: Write data

write.table(bacteria_2, file=”updated_bacteria”)

Basic R functionality

Don’t forget to add comments to your R code!
Use #

```
x <- c(1,2,3,4,5,6)   # Create ordered collection (vector)
y <- x^2              # Square the elements of x
mean(y)               # Calculate average (arithmetic mean) of y
[1] 15.16667
```

Basic R functionality

TAB completion
Up-arrow for last command

In RStudio:
Ctrl + Enter to run current line or selection
(**Cmd + Enter** in Mac OS)

Exercises

1. Introduction to R
2. Introduction to ggplot