

An Introduction to R

Scott A. Handley, PhD
April 25, 2013

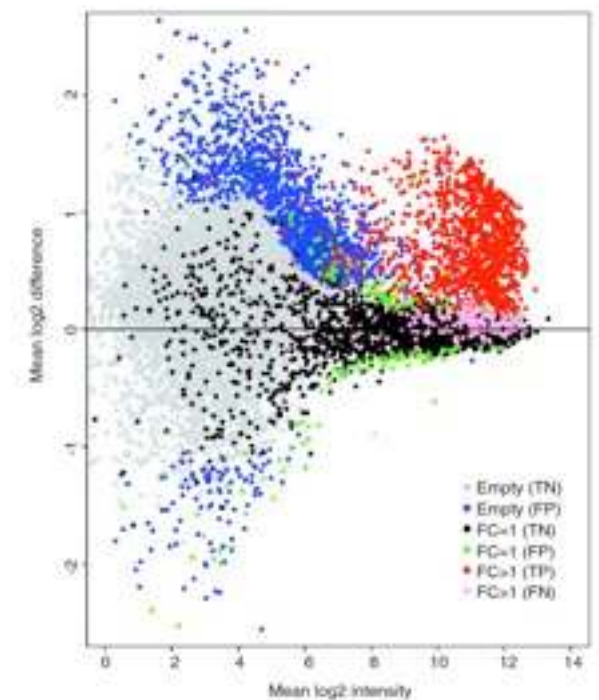
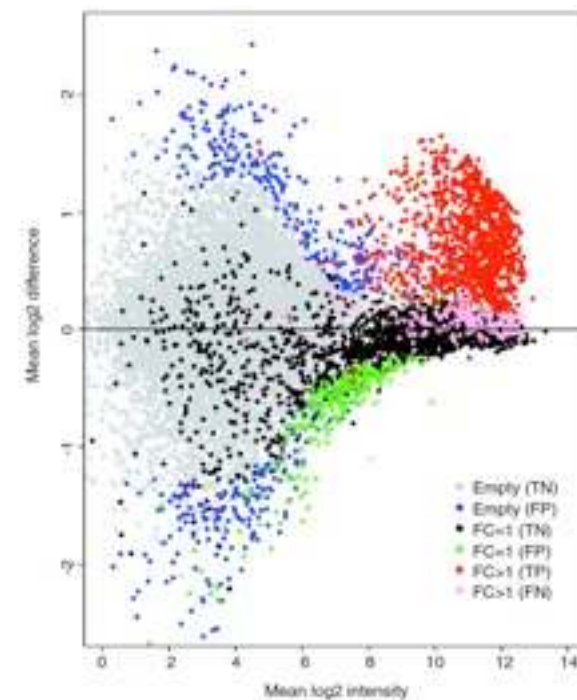
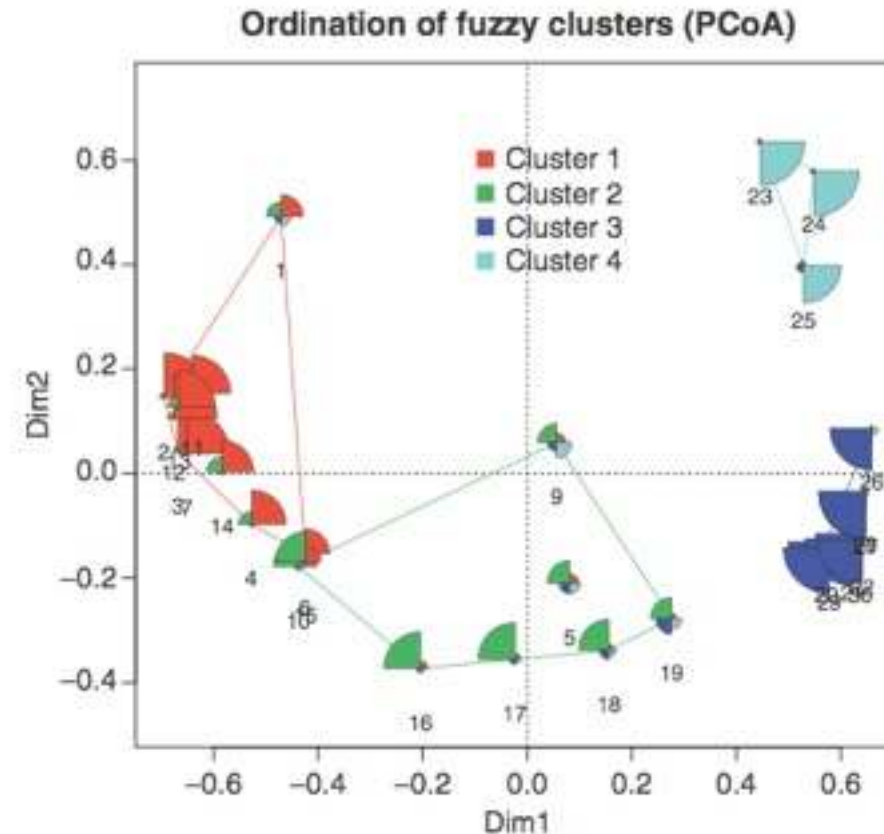


Goals

- Mechanical, not theoretical
- Upon leaving, you should be able to:
 - Get data into and out of R
 - Install specialized packages
 - Complete basic summary statistics on a set of data
 - Graph data

What is R?

A free software environment for statistical computing and graphics



Why is R useful?

- Data management and manipulation
- Well established system of packages and documentation
- Support for rich statistical simulation and modeling
- Cutting-edge graphical data visualization
- Development
 - High-level interpreted language to prototype new computational methods
 - Active development and dedicated community
- 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
- Take an on-line course (<https://www.coursera.org/course/compdata>)

Why is R Useful for Biologists

- Bioconductor (<http://bioconductor.org>)
 - 671 packages (4/25/2013):
 - Variant detection: coding changes, PolyPhen database
 - Annotation: pathway analysis, access GO, KEGG, NCBI and many others
 - High-throughput assays: flow cytometry, mass spec
 - Transcription factor binding detection
- Ecology (see: <http://cran.r-project.org/web/views/Environmetrics.html>)
 - Ordination (exploratory data analysis, not hypothesis testing. Ex. PCA, PCoA)
 - 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

Obtaining R

- Ubuntu

```
sudo apt-get update
```

```
sudo apt-get install r-base r-base dev
```

... or use the Ubuntu package manager

- Install directly from CRAN (The Comprehensive R Archive Network)

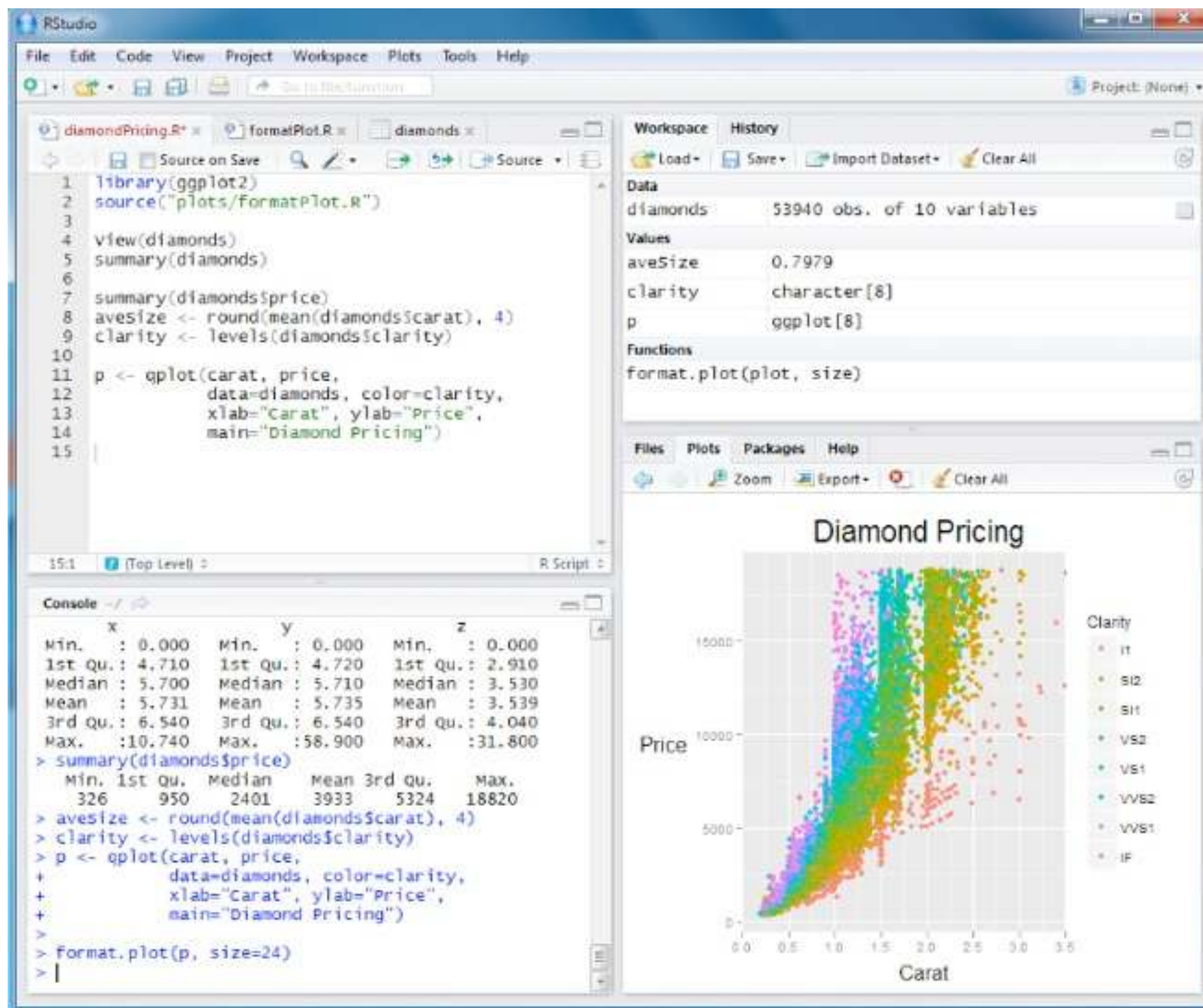
- <http://cran.r-project.org/bin/macosx/>



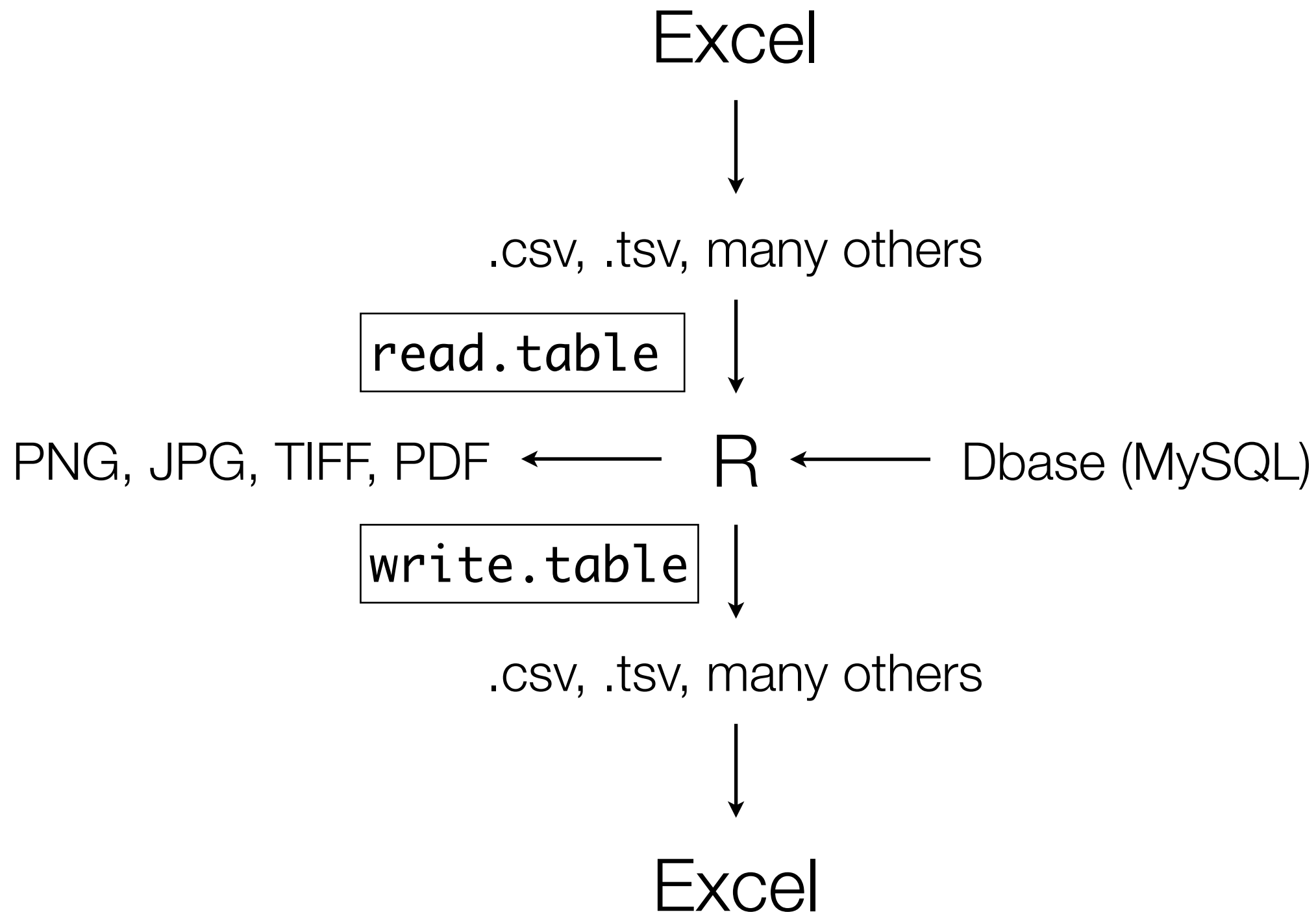
Running R

- Install a 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
 - Useful for scripting

R Studio



Getting data into and out of R (one example)



Advanced Interactive Visualization Outputs

Interactive Scatterplot Matrix

http://healthviz.appspot.com/display/hs_43002

Interactive HeatMap

http://healthviz.appspot.com/display/hs_64003

Not R, but another great example

http://www.brown.edu/Faculty/Dunn_Lab/

Getting Started

- 1) Open RStudio
- 2) Get familiar with the interface
- 3) Set a working directory (setwd)

Data Conventions

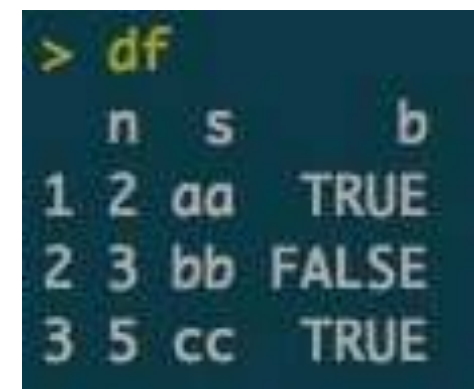
no_whitespace_
cAsE_sENsItive*

* You can have whitespace separating commands, just not within commands or filenames. Just don't do it. If something isn't working it is due to whitespace or the wrong case 75% of the time

Creating Data in R

Create a simple data frame

```
n = c(2,3,5)
s = c("aa", "bb", "cc")
b = c(TRUE, FALSE, TRUE)
df = data.frame(n,s,b)
df
```



```
> df
  n  s    b
1 2 aa TRUE
2 3 bb FALSE
3 5 cc  TRUE
```

Note: c stands for *combine*

Calling Positions Within a Data Frame

Convention

`data.frame[row, column]`

`df[1,1]`

`df[1,2]`

`df[1:3,1]`

`df[1,1:2]`

`df.2 <- df[-1,-2]`

```
> df
  n  s    b
1 2 aa  TRUE
2 3 bb FALSE
3 5 cc  TRUE
```

Data in R

	taxon_1	taxon_2	taxon_3	taxon_4	taxon_5
control_1	10	43	143	506	6004
control_2	8	53	198	786	5647
exper_1	120	67	212	103	5674
exper_2	8	89	98	978	3012
exper_3	38	12	101	1005	4056

`data.frame[1:2,3:5]`

r comes with many example data frames (mtcars)

Core R functionality

Calculator

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

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

Set Variables / Vectors

```
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 51 103

`sum(n)` [1] 175

`mean(n)` [1] 35

`var(n)` [1] 1796.5

`min(n)` [1] 3

`max(n)` [1] 103

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

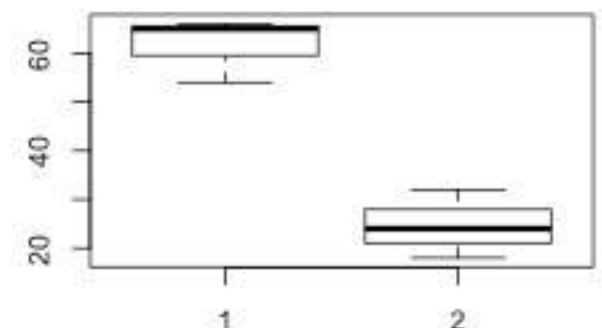
```
data.frame$proteobacteria
```

```
[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])
```



Functions Applied to Data Frames

`tapply()`: applies a function over a ranged array (data table)

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

Basic syntax:

- A **numeric** distribution
- A **factor** dividing the numeric distribution into groups
- A **function** (mean, var, sd, sum ...)

tapply()

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

What is the mean number of clostridia in Males? Females?

- numeric distribution
- factor dividing numeric distribution
- function

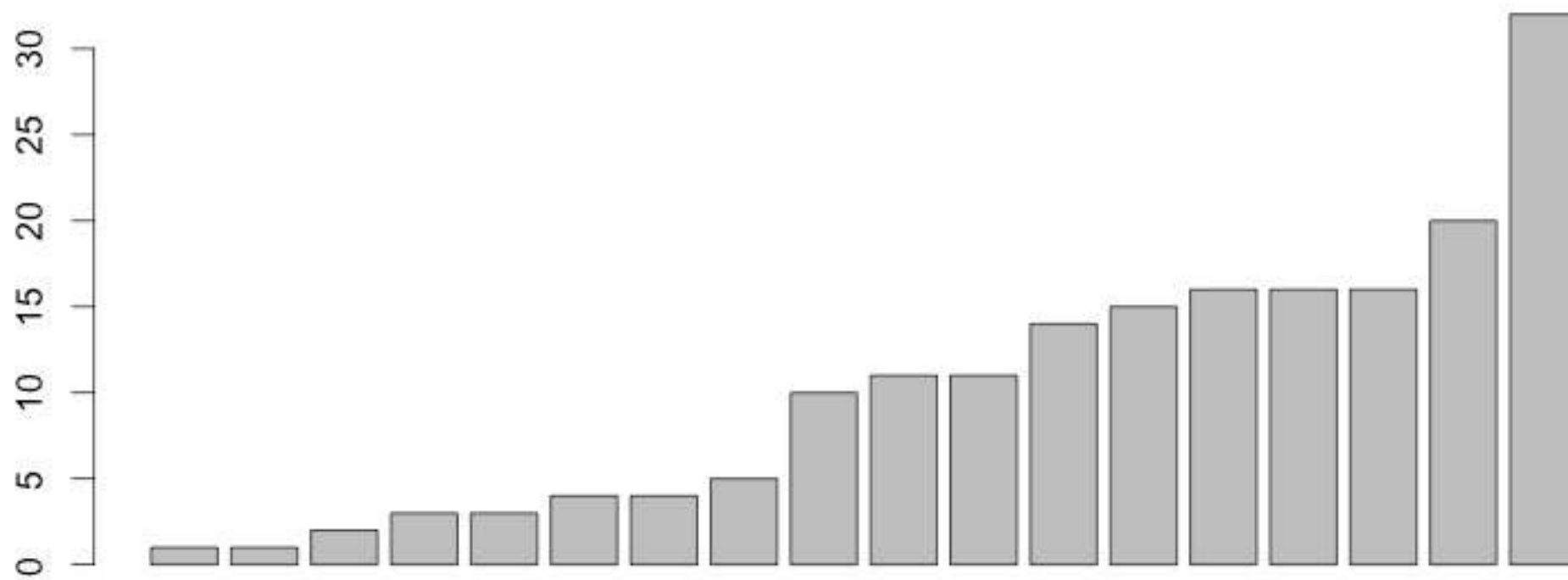
```
tapply(bac$clostridia, bac$sex, mean)
```

```
      F      M  
29.00 19.25
```

Basic Visualization I

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

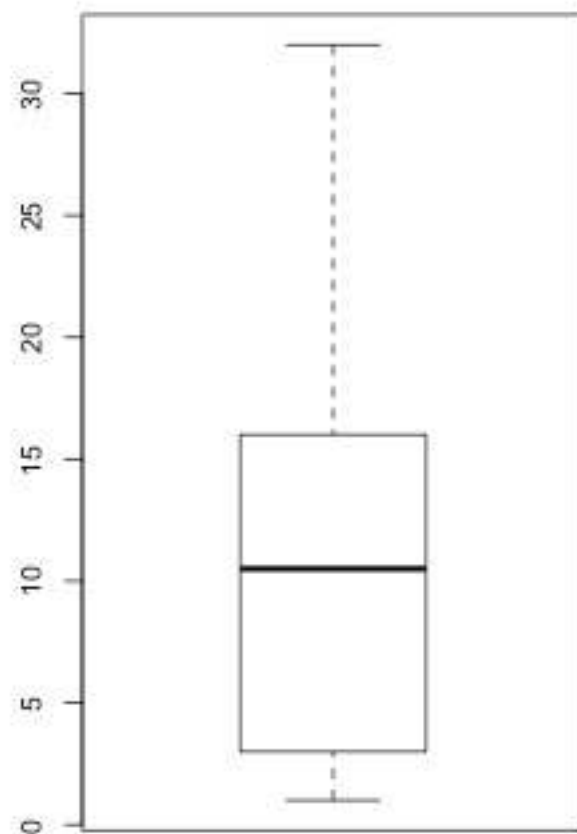
`barplot(y)`



Basic Visualization II

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

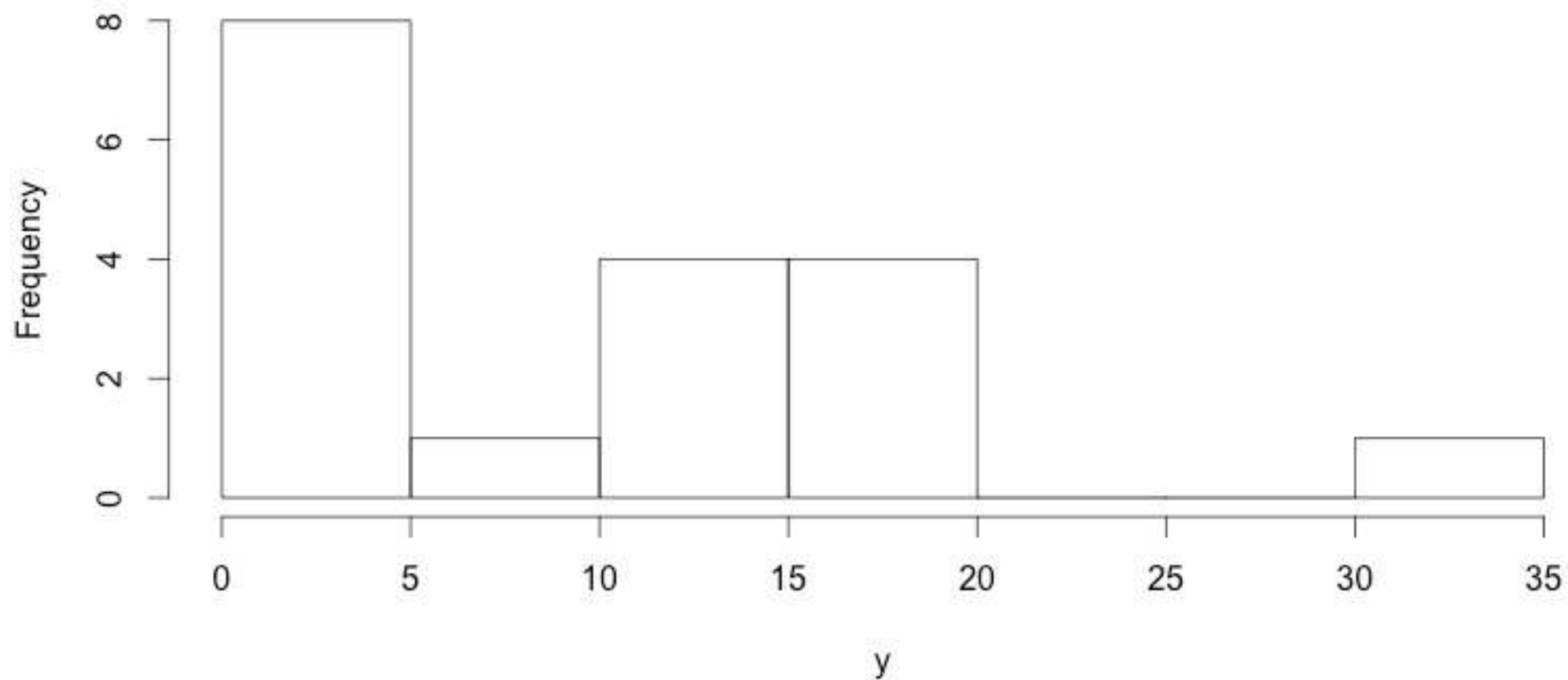
```
boxplot(y)
```



Basic Visualization III

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

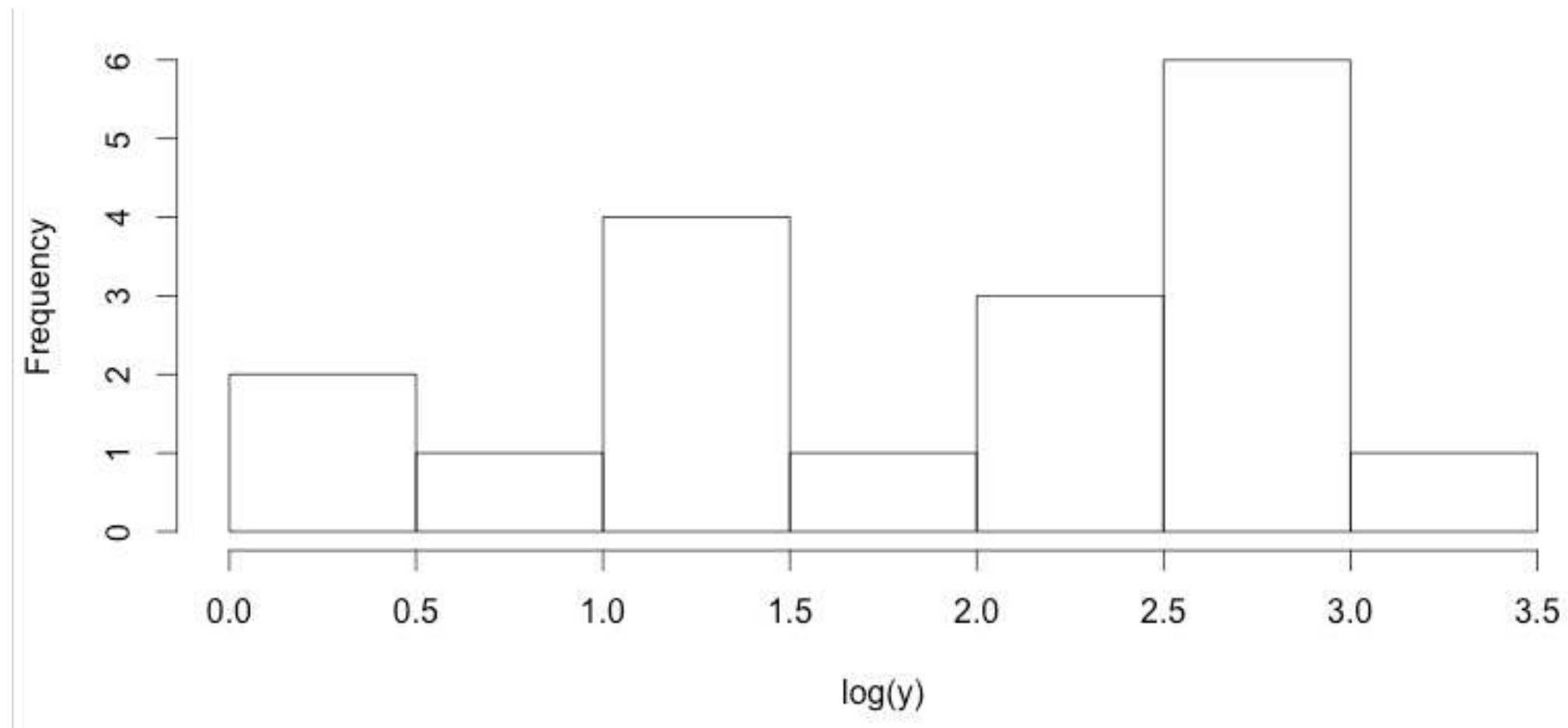
`hist(y)`



Basic Visualization III.i

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

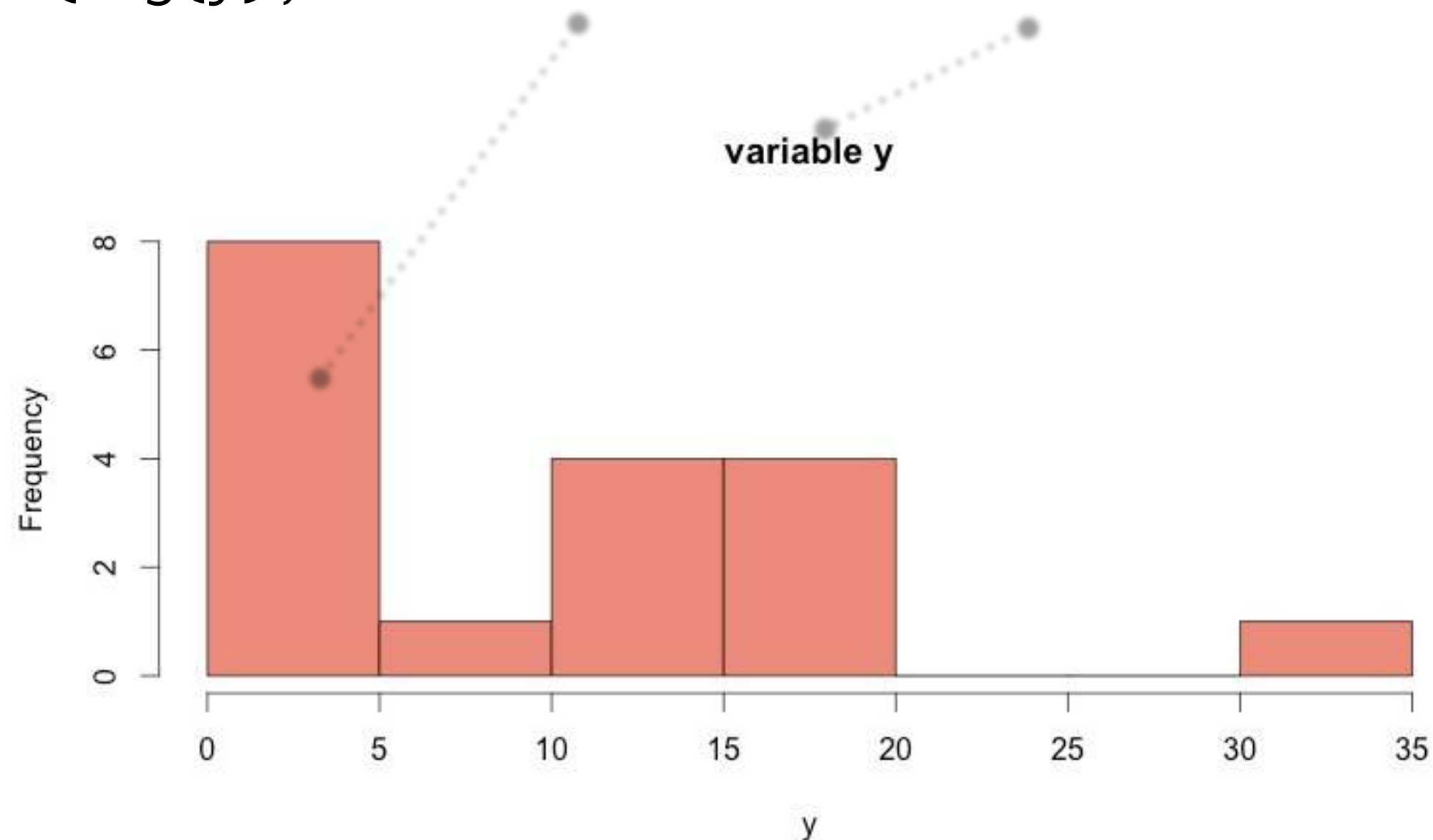
`hist(log(y))`



Basic Visualization III.ii

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

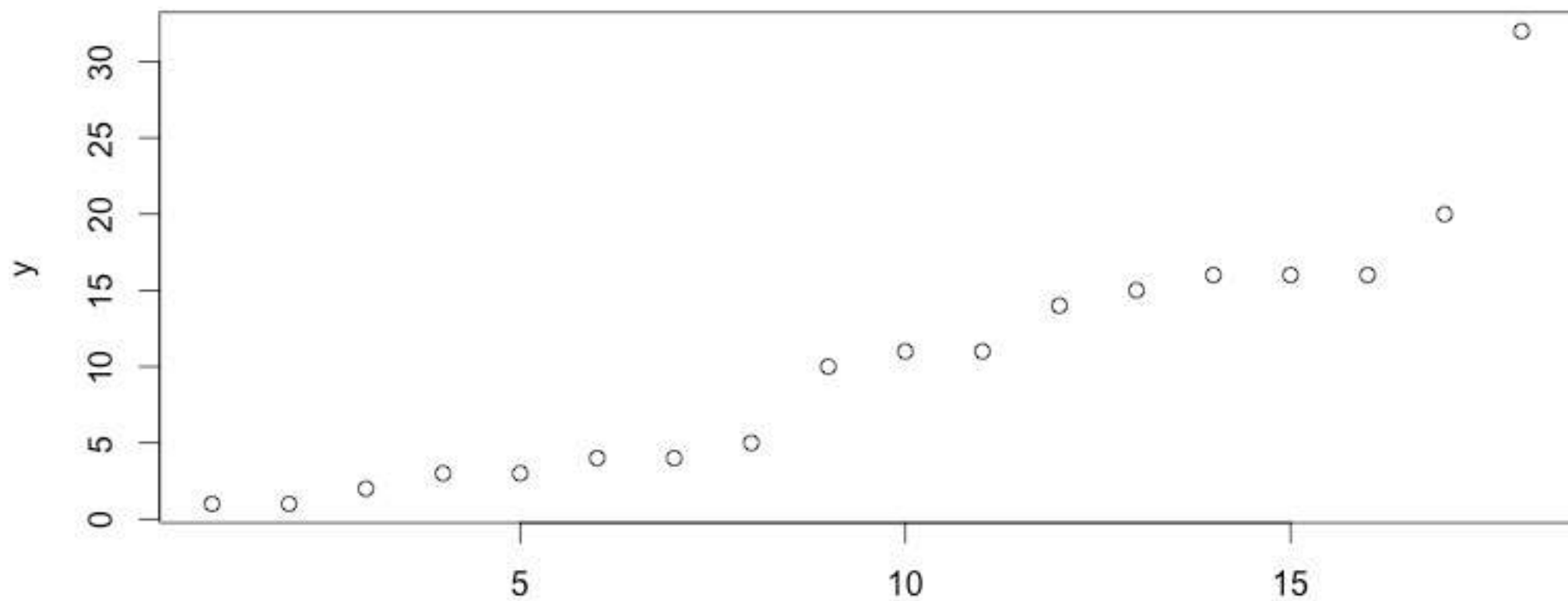
```
hist(log(y),
```



Basic Visualization IV

$y=(1,1,2,3,3,4,4,5,10,11,11,14,15,16,16,16,20,32)$

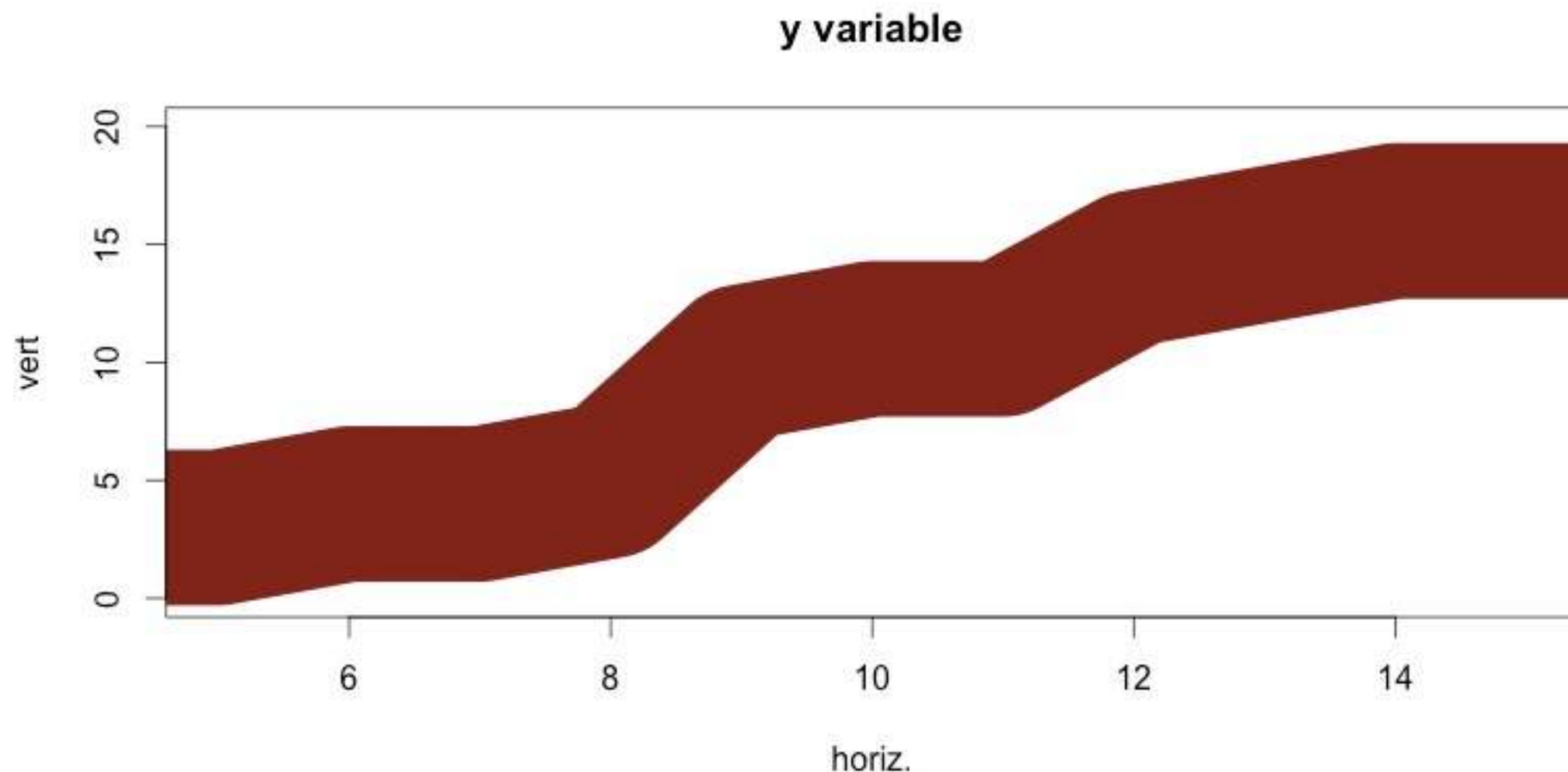
`plot(y)`



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, including [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.

Factors & Quick Summary Statistics

```
fac=as.factor(c("saint", "louis", "beer", "herpes",  
"beer", "herpes"))
```

```
levels(fac)
```

```
[1] "beer" "herpes" "louis" "saint"
```

```
summary(fac)
```

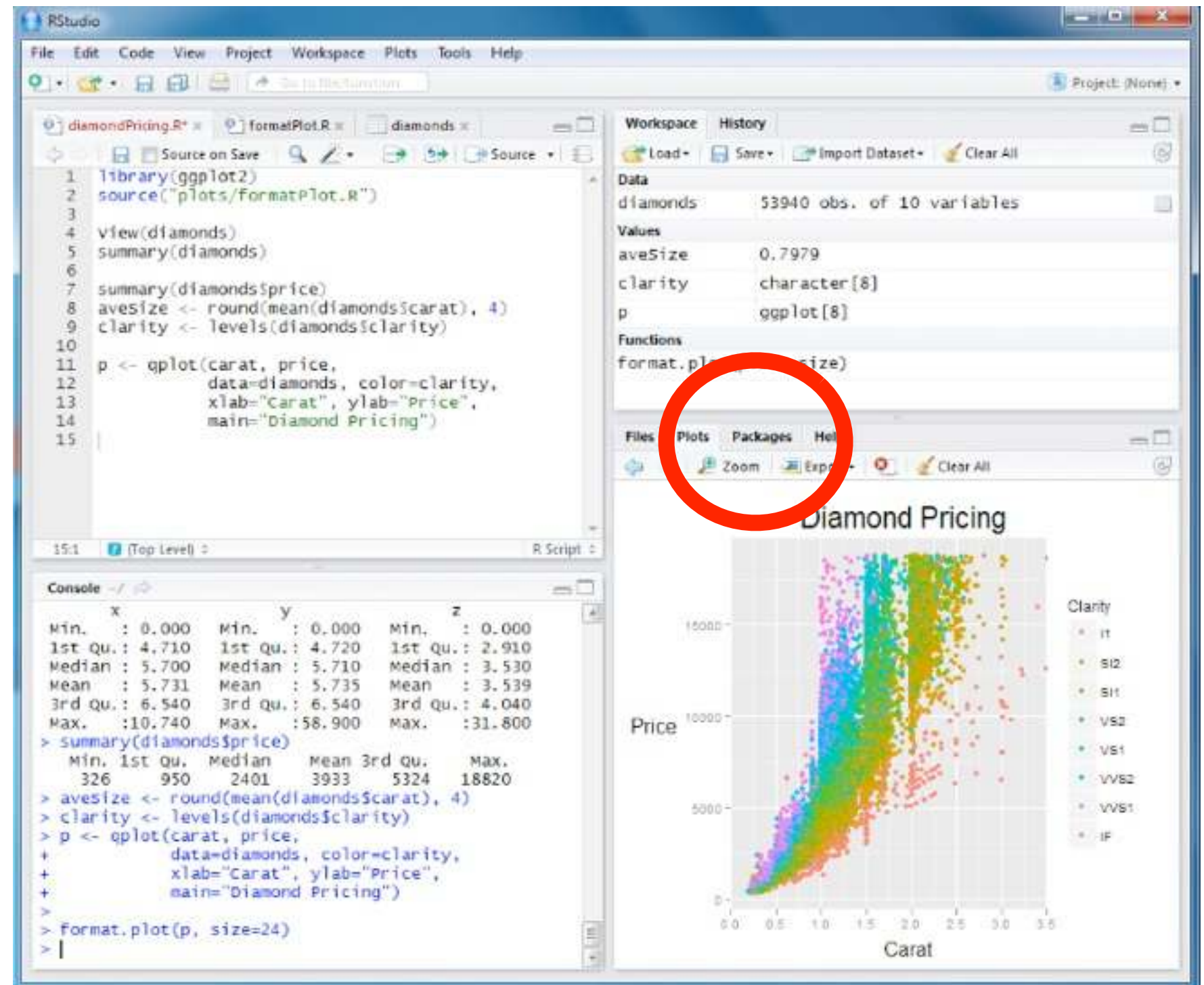
```
> summary(fac)  
beer herpes louis saint  
      2      2      1      1
```

```
summary(y)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.00	3.25	10.50	10.22	15.75	32.00

Other Miscellaneous but Essential R Mechanics

R Mechanics - Installing Packages



`install.packages("vegan")`

Getting Help in R

?write.table

Description

`write.table` prints its required argument `x` (after converting it to a data frame if it is not one nor a matrix) to a file or [connection](#).

Usage

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")
```

```
write.csv(...)
write.csv2(...)
```

Arguments

<code>x</code>	the object to be written, preferably a matrix or data frame. If not, it is attempted to coerce <code>x</code> to a data frame.
<code>file</code>	either a character string naming a file or a connection open for writing. "" indicates output to the console.
<code>append</code>	logical. Only relevant if <code>file</code> is a character string. If <code>TRUE</code> , the output is appended to the file. If <code>FALSE</code> , any existing file of the name is destroyed.

R Studio Overview

- <http://evomics.org/learning/programming/introduction-to-r/>