

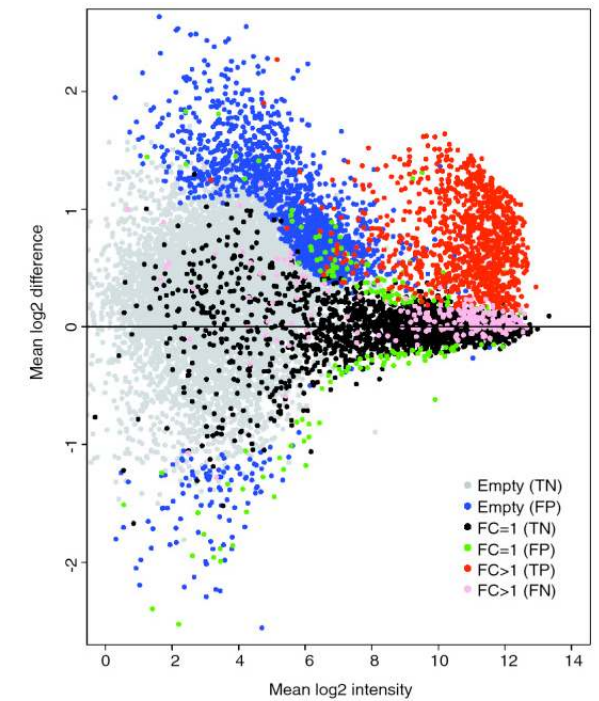
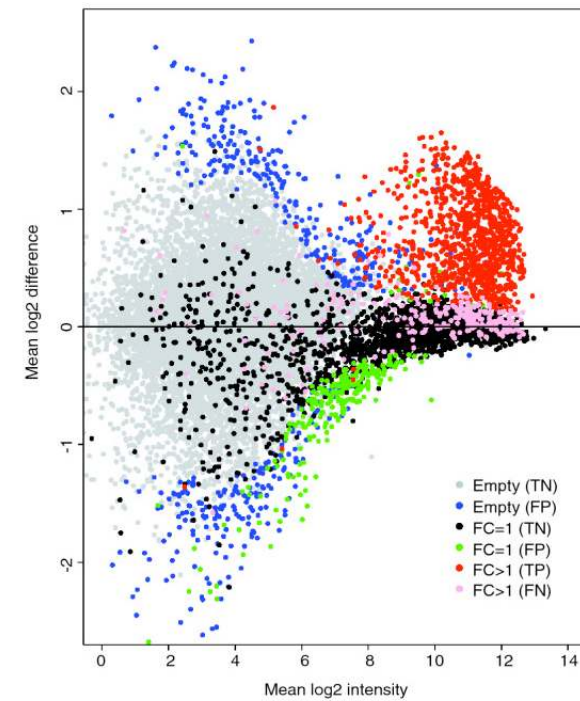
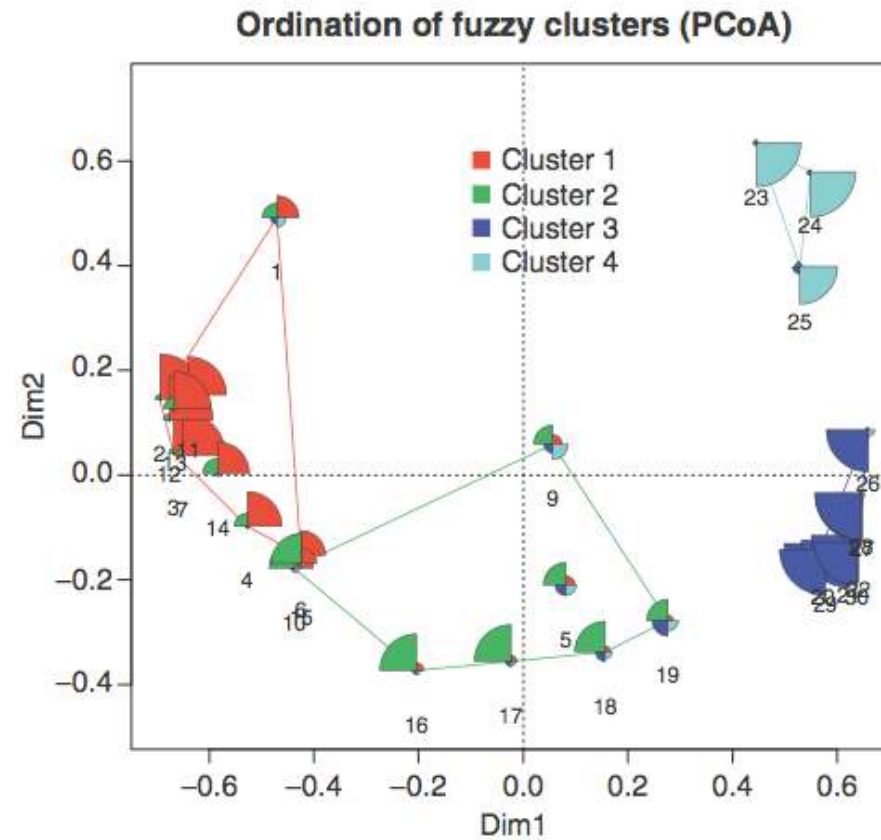
An Introduction to R

Scott A. Handley, PhD
Washington University St Louis



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
- 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>)
 - 1560 packages (07-10-2018):
 - 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
 - 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

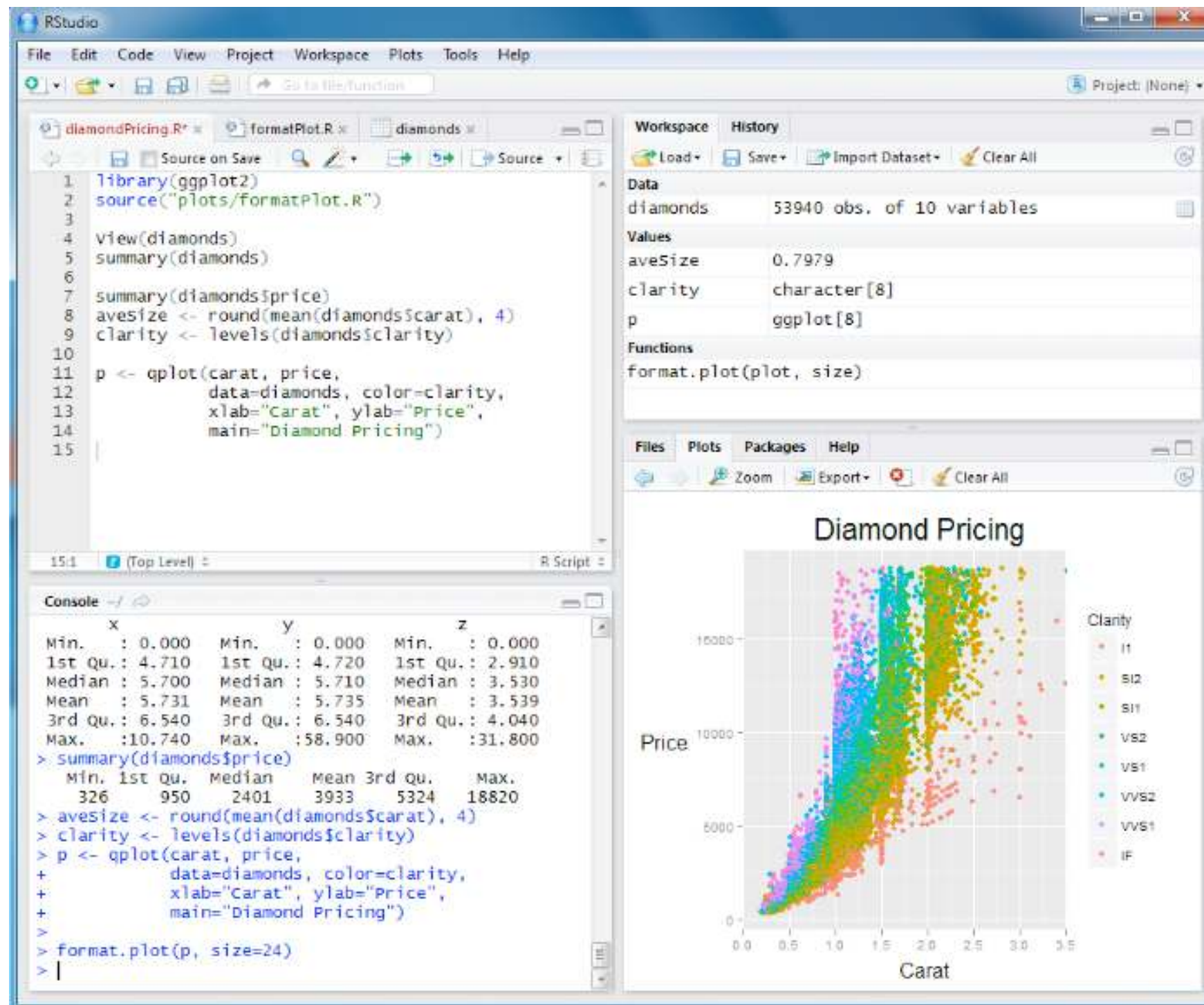
- Windows, Mac or Linux OS: <https://www.r-project.org>



Running R

- Install a R Integrated Development Environment (IDE)
 - RStudio: <http://www.rstudio.com>
 - Makes working with R much easier, particularly for a new R user
 - Run on Windows, Mac or Linux OS
- Or from the command line, type R

R Studio



Basic 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 51 104
```

```
sum(n)
```

```
[1] 175
```

```
mean(n)
```

```
[1] 35
```

```
var(n)
```

```
[1] 1796.5
```

```
min(n)
```

```
[1] 3
```

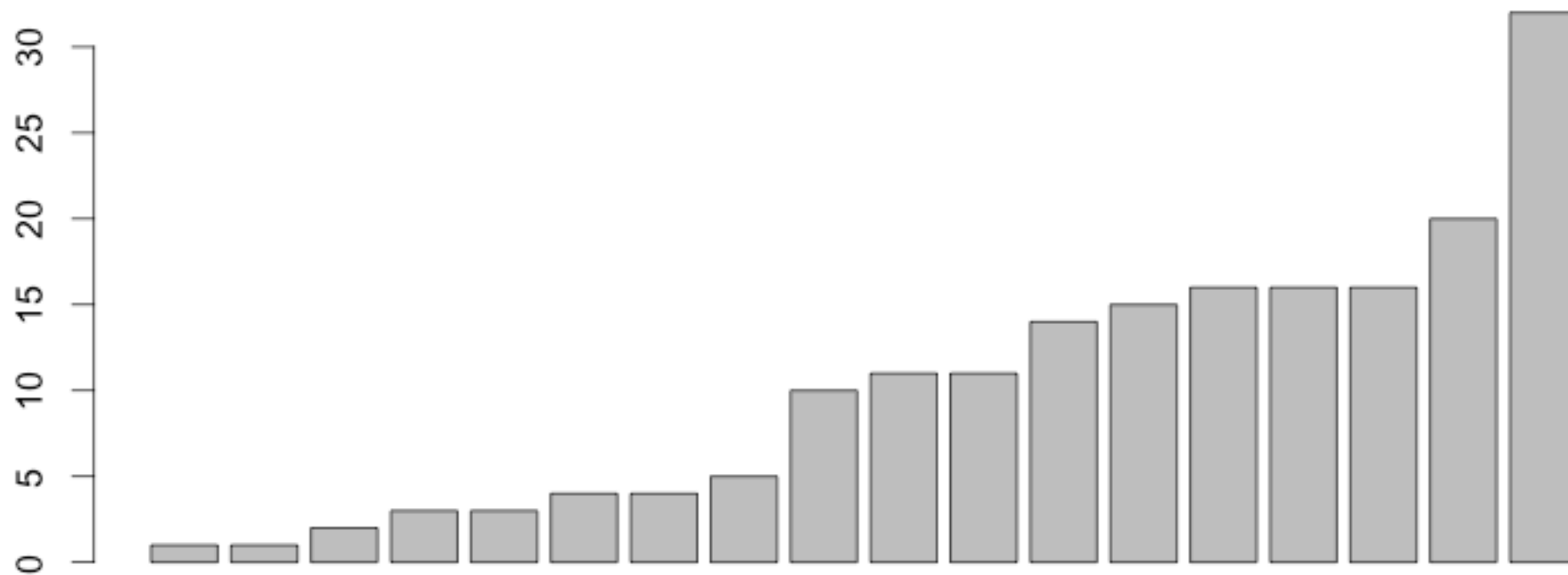
```
max(n)
```

```
[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)`

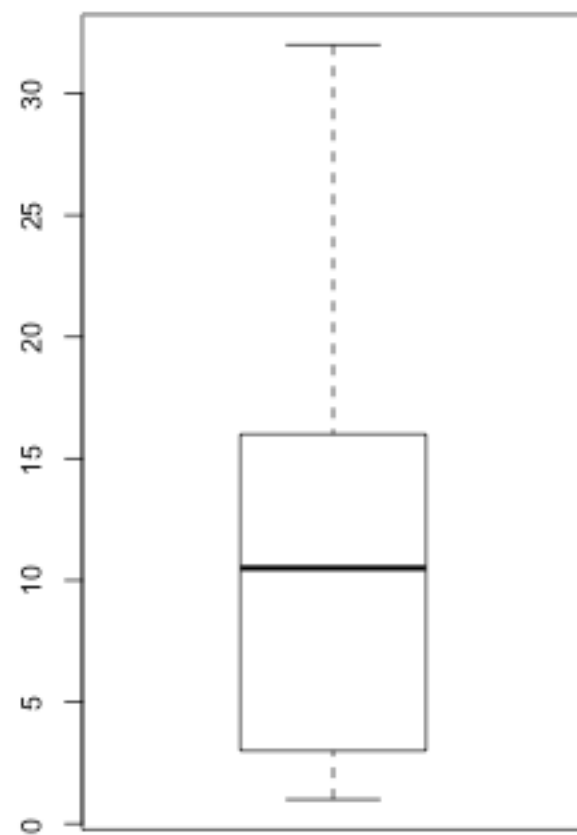
`barplot(y)`



Basic Visualization II

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

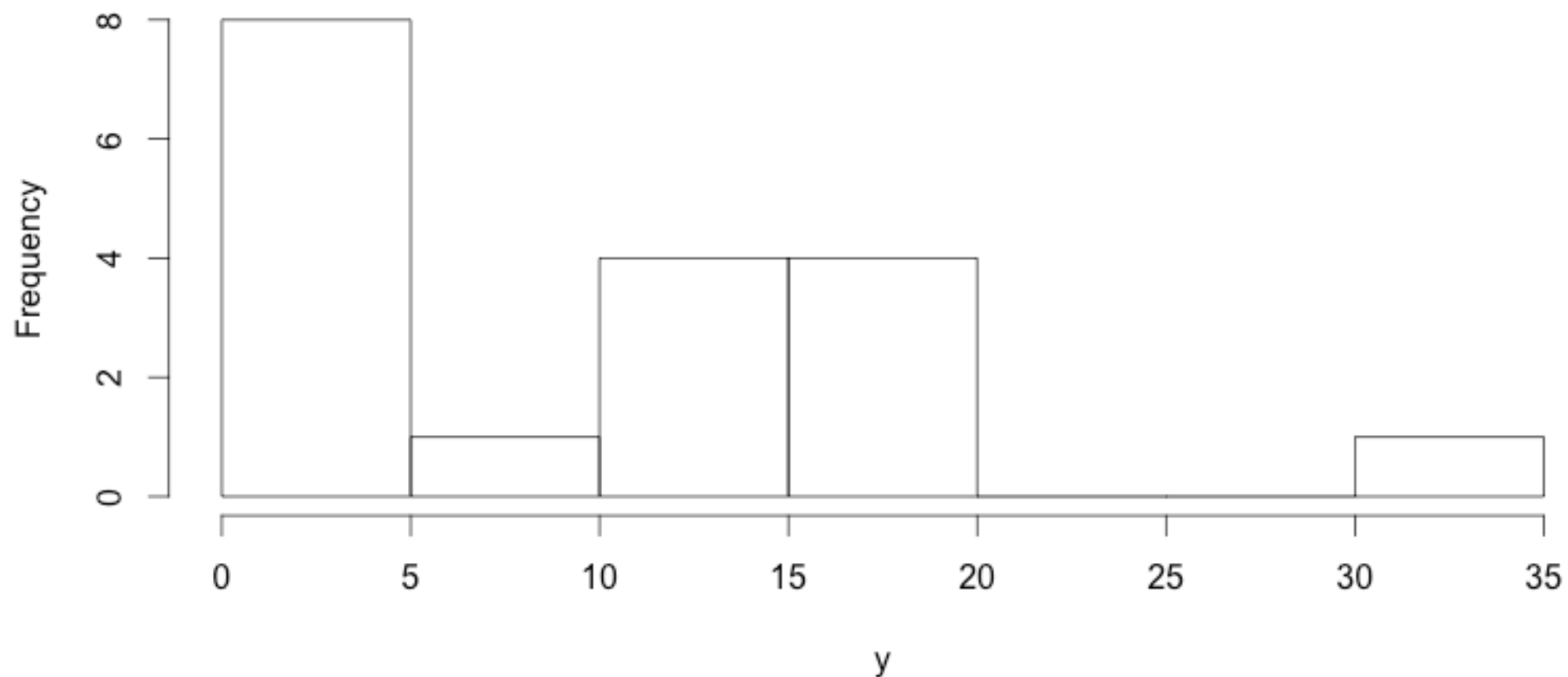
```
boxplot(y)
```



Basic Visualization III

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

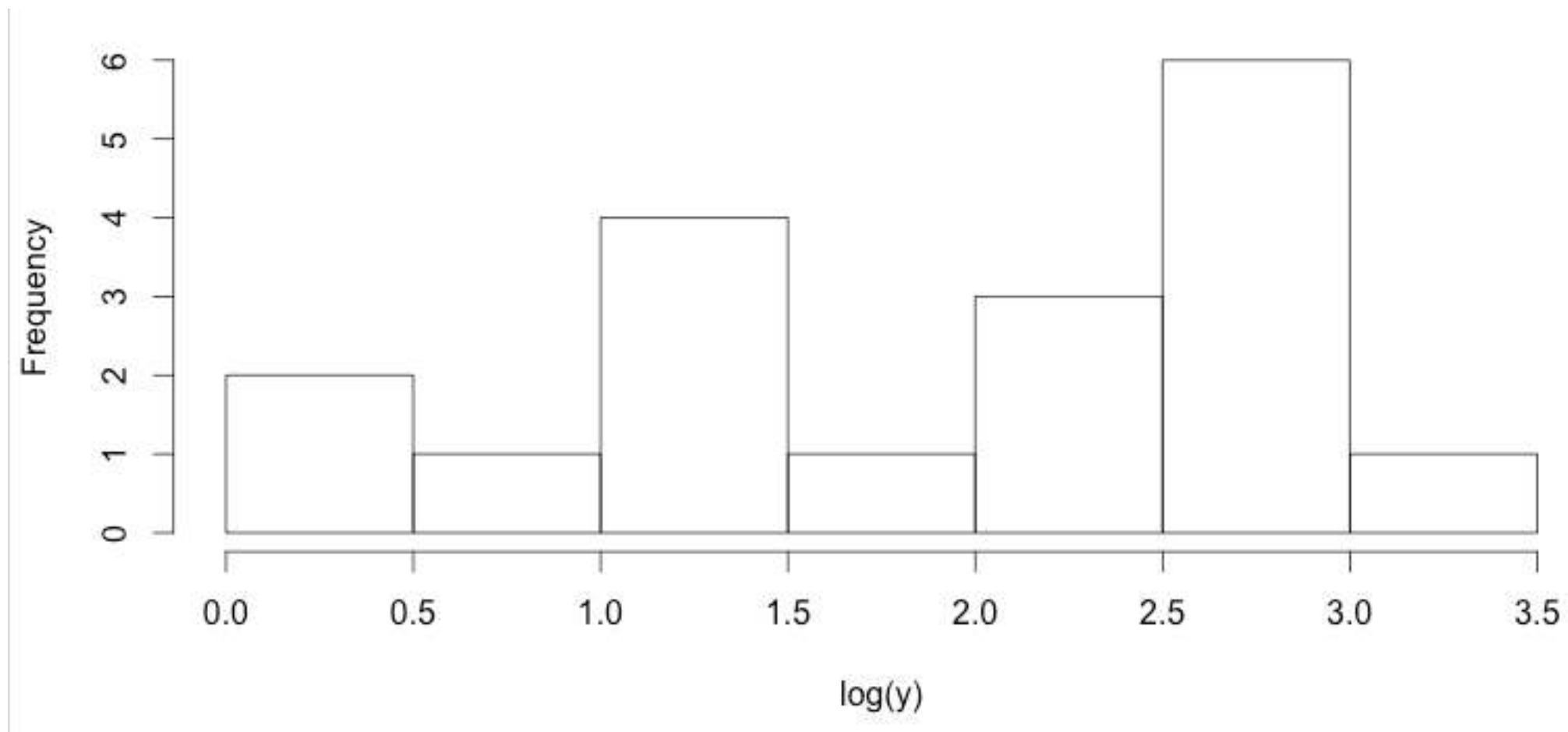
`hist(y)`



Basic Visualization III.i

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

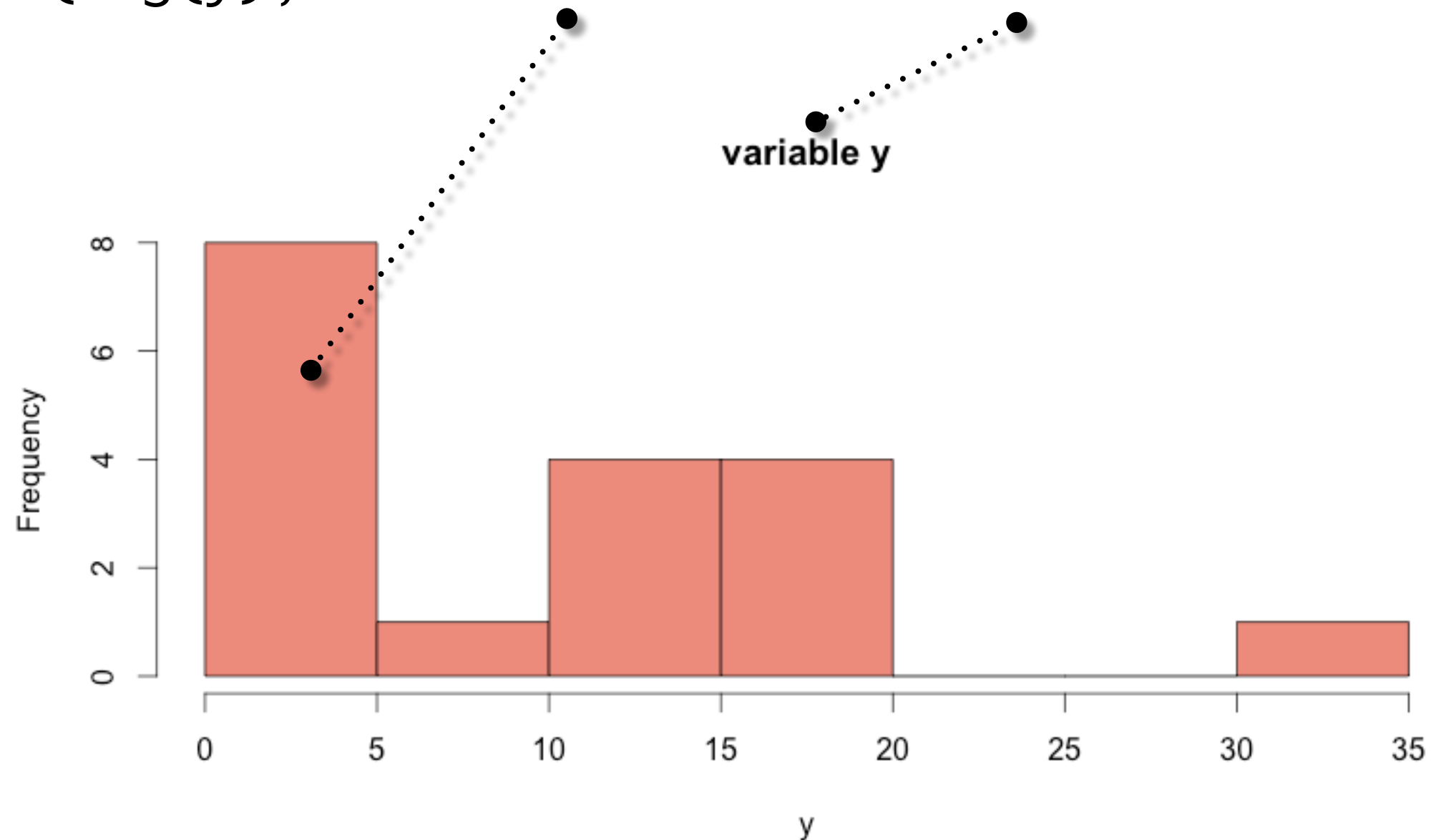
`hist(log(y))`



Basic Visualization III.ii

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

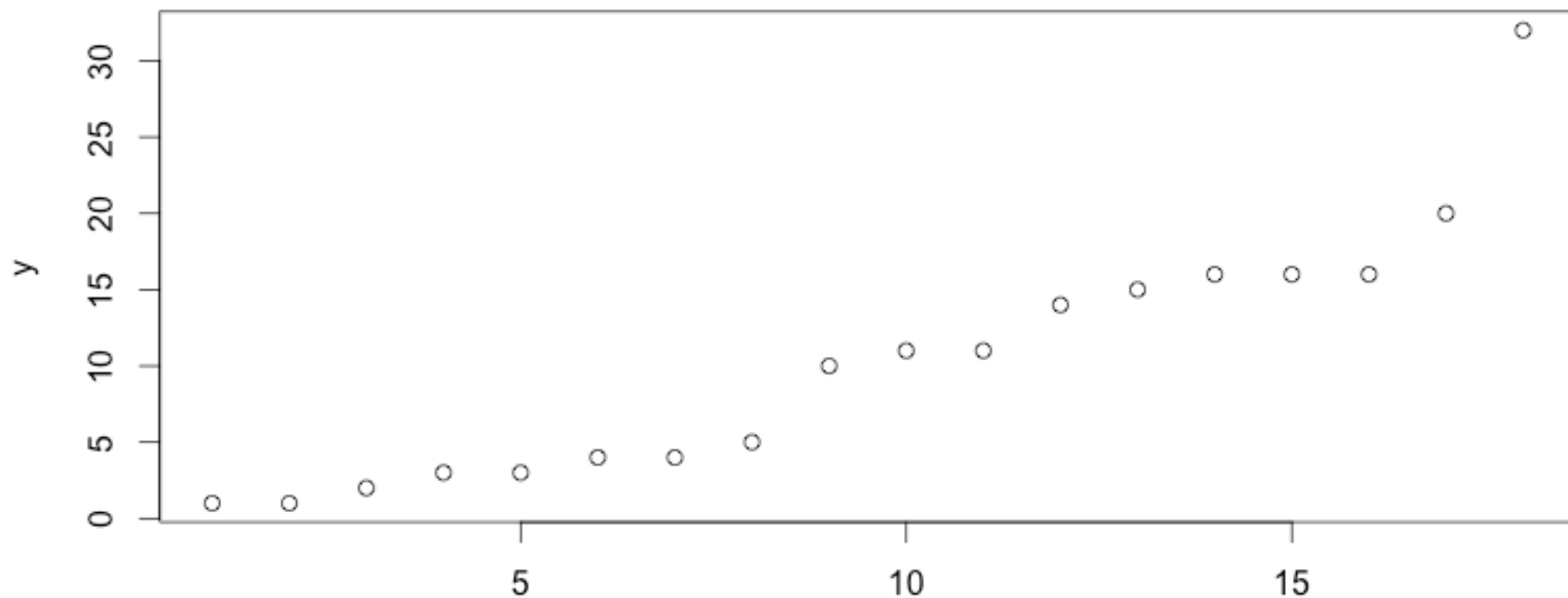
`hist(log(y),`



Basic Visualization IV

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

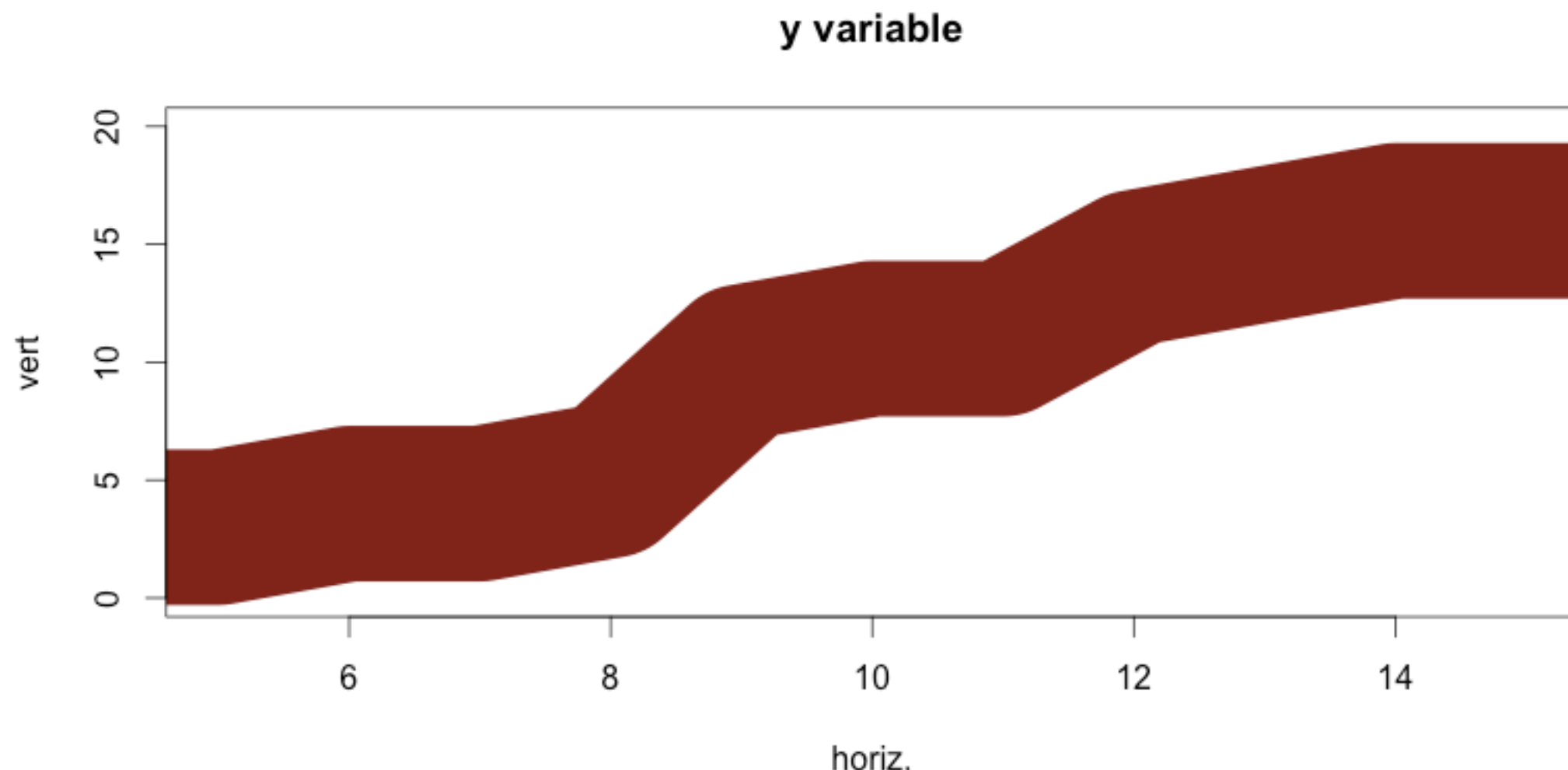
`plot(y)`



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.

Data Frames

- A data.frame is essentially a table
- columns can be mixed types
 - numeric, text strings
- rows must be same type

	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]`

row

column

	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

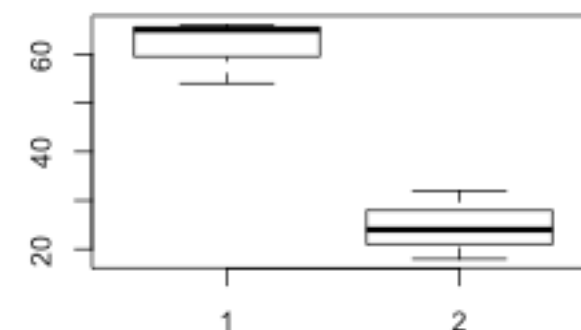
```
data.frame$proteobacteria
```

```
[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])
```



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.

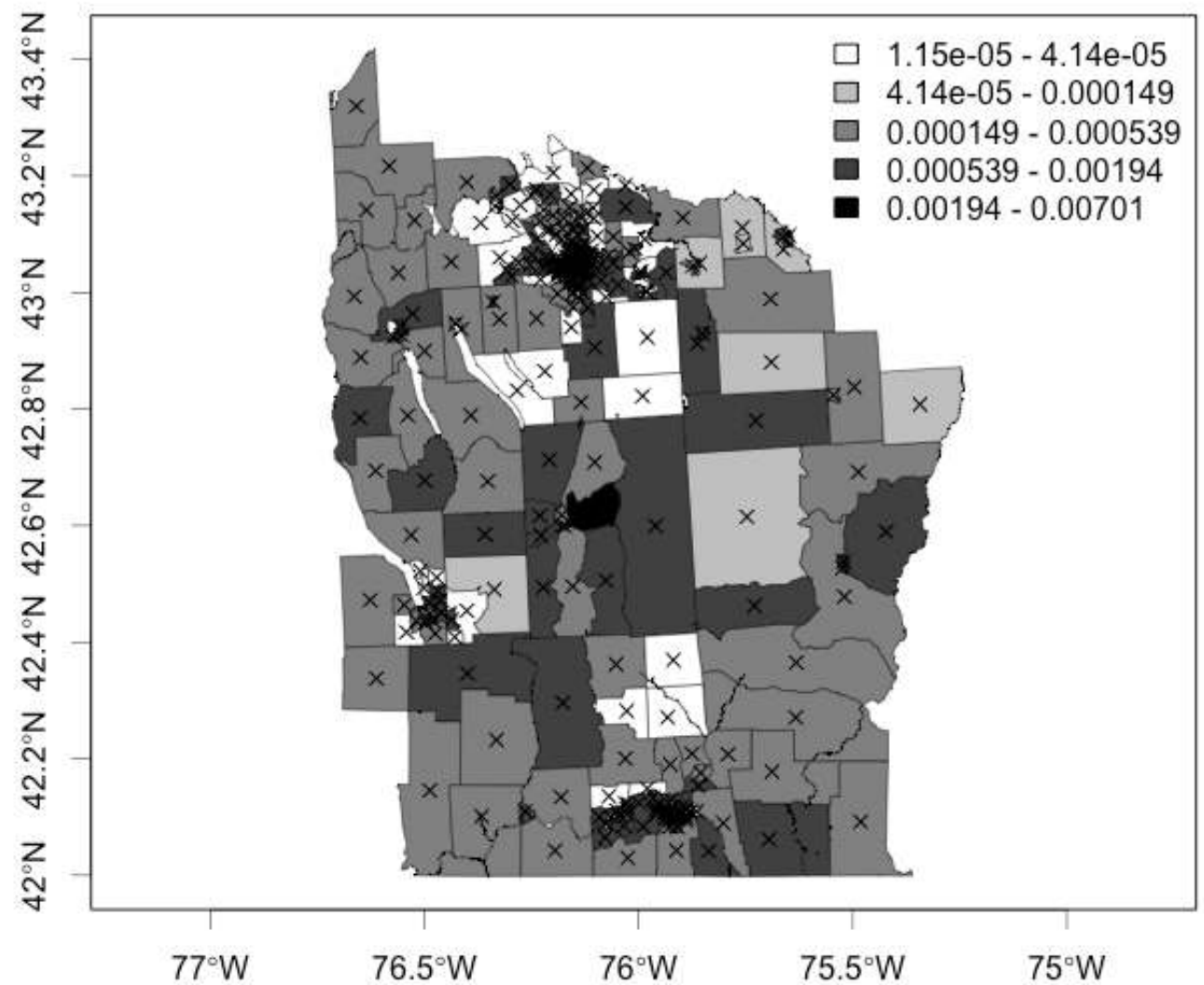
Advanced and Other Applications in R

Spatial Epidemiology and Maps

```
library(SpatialEpi)

data(NYleukemia)
sp.obj <- NYleukemia$spatial.polygon
centroids <- latlong2grid(NYleukemia$geo[, 2:3])
population <- NYleukemia$data$population
cases <- NYleukemia$data$cases

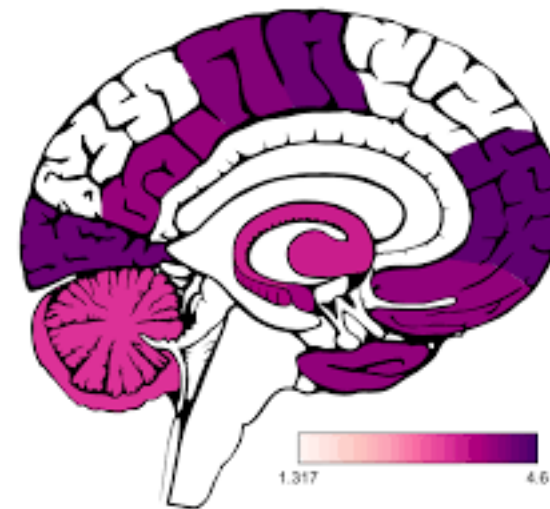
plotmap(cases/population, sp.obj, log=TRUE, nclr=5)
points(grid2latlong(centroids), pch=4)
```



Anatomical Mapping

CerebroViz

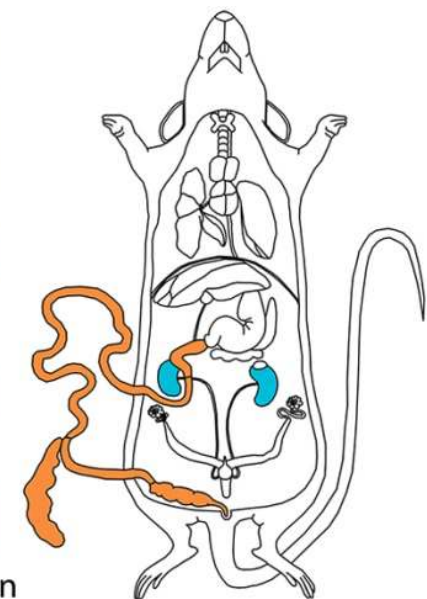
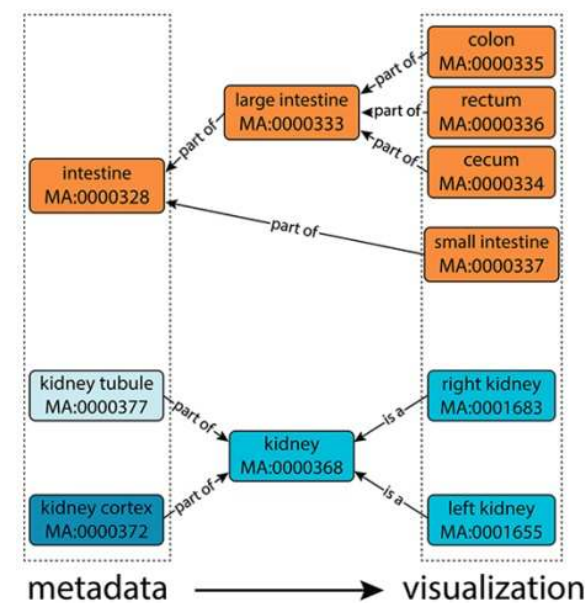
<https://github.com/ethanbahl/cerebroViz>



```
library("cerebroViz")  
data(cerebroEx)  
head(cerebroEx)[, c(1:7)]
```

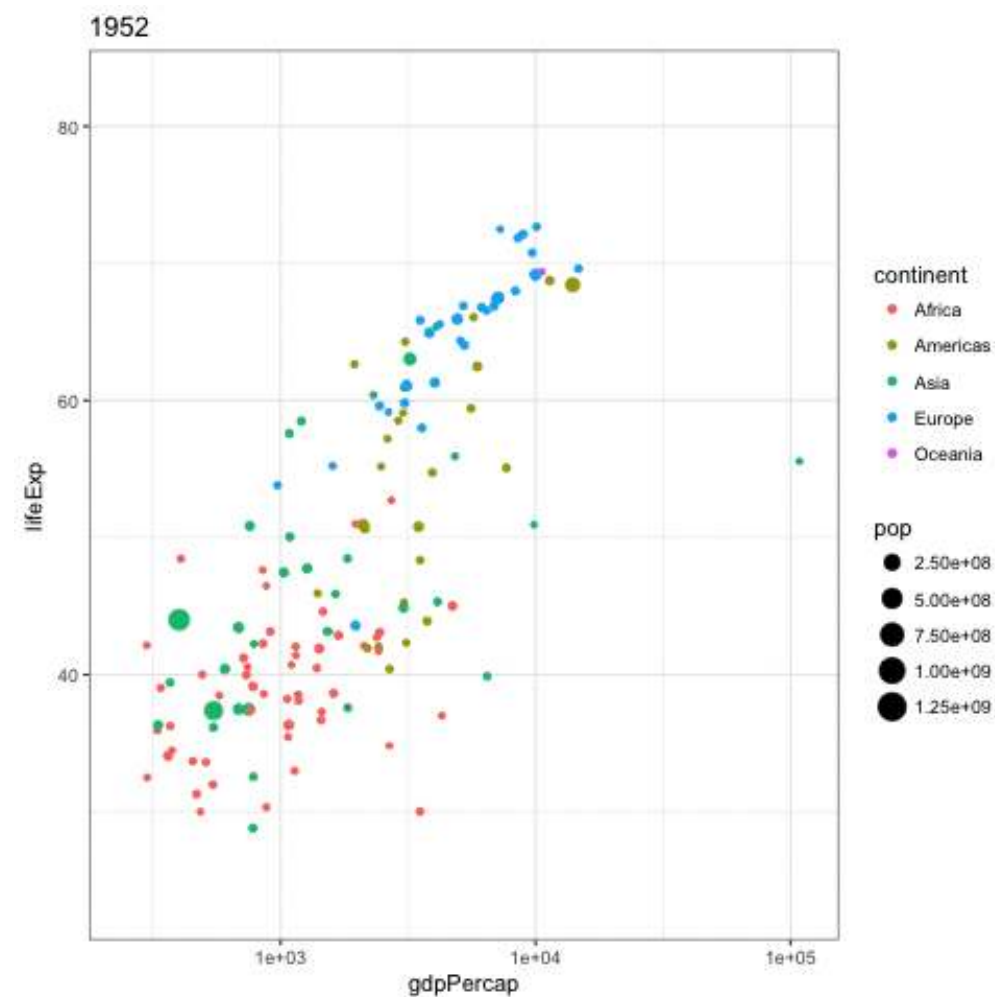
COMICS

<https://github.com/y-popov/COMICS>

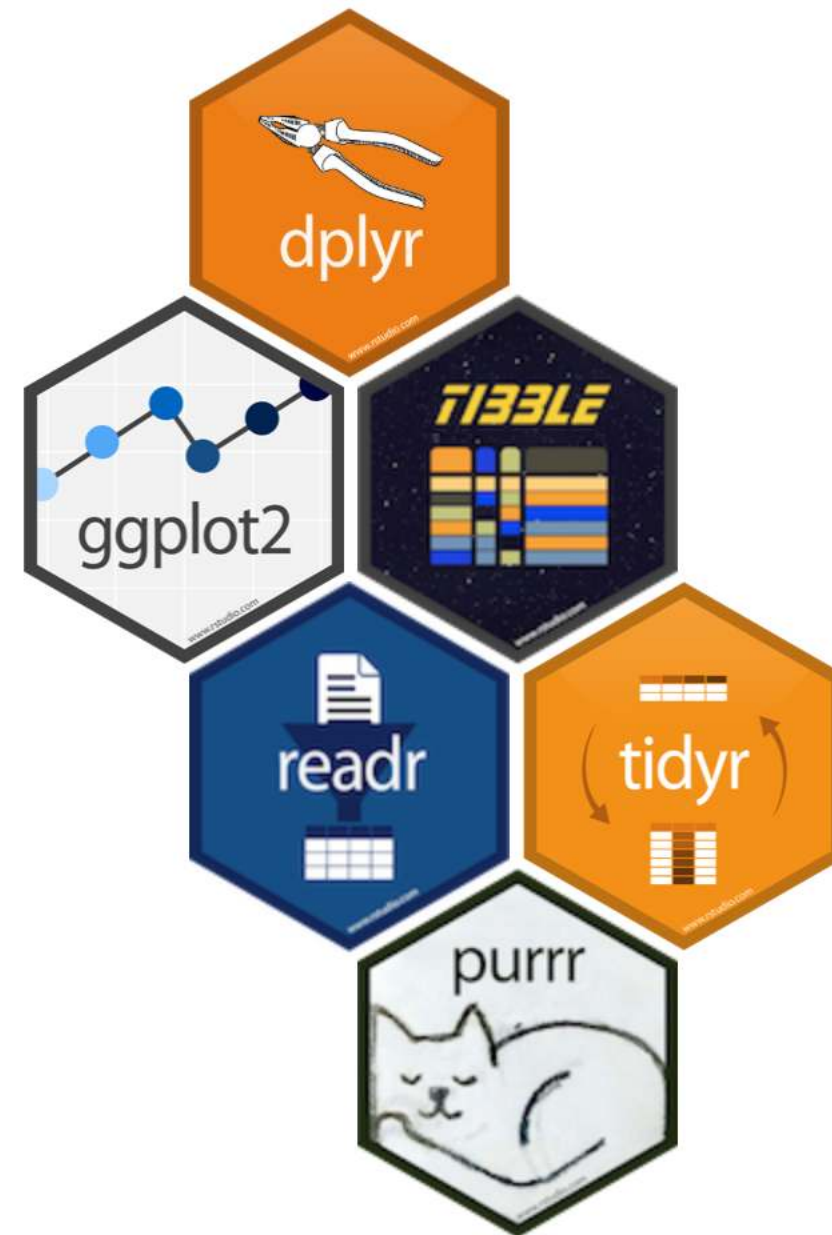


Animation / Interactivity

Indicator Name	2011	2012	2013	2014	2015	2016	Average	Improvement
Prevalence of Obesity	19.1	23.6	23.3	20.5	24.0	23.2	22.28	-21.47
Prevalence of Tobacco Use	17.4	15.0	15.3	12.2	16.6	16.7	15.53	4.02
Prevalence of Cardiovascular Disease	5.0	4.9	1.5	4.4	4.9	6.2	4.48	-24.00
Prevalence of Diabetes	8.0	7.2	9.3	7.2	7.5	10.4	8.27	-30.00



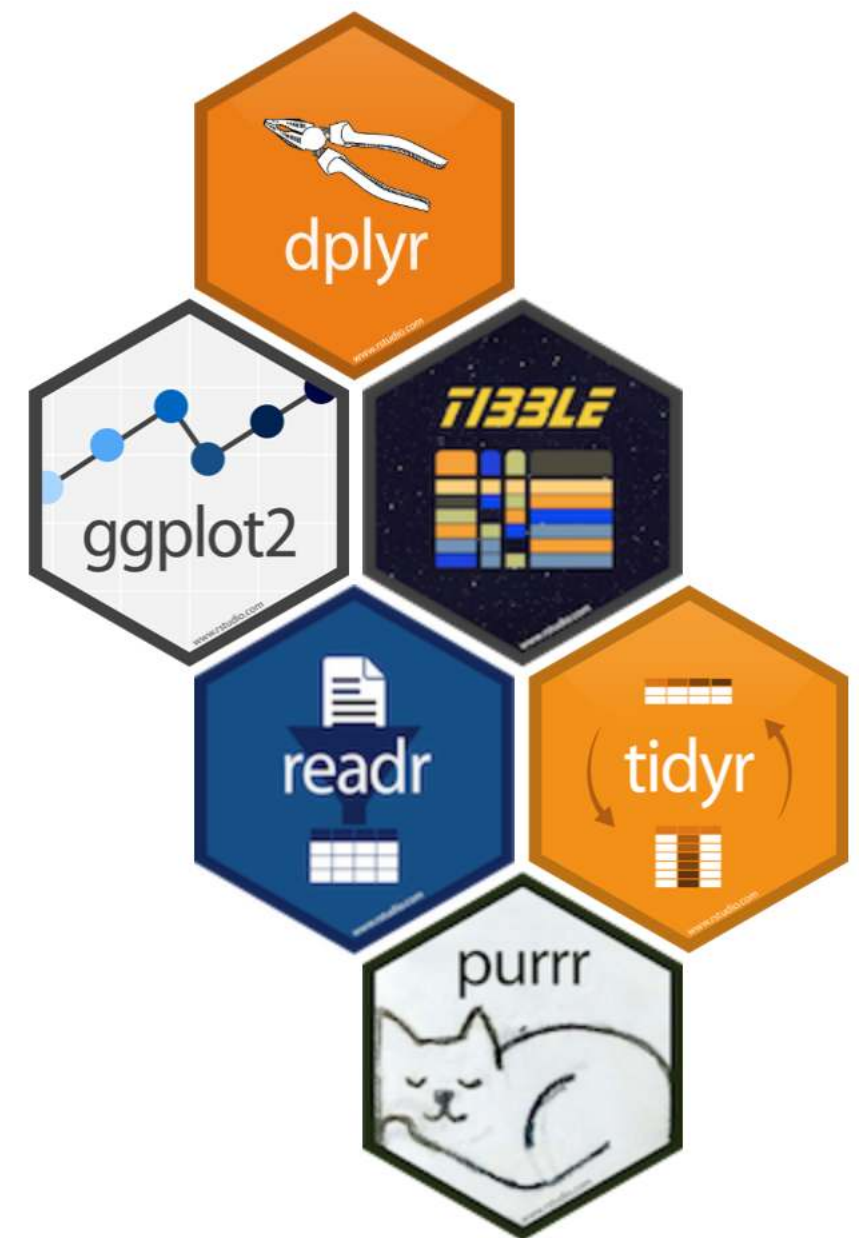
Tidyverse



What is the tidyverse

The tidyverse is an opinionated **collection of R packages** designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- ggplot2: a system for creating graphics (**G**rammar of **G**raphics)
- dplyr: a system for data manipulation
- tidyr: functions for tidying data
- readr: a system for reading in data
- purrr: functional programming in R
- stringr: system for working with strings
- forcats: system for working with factors (categorical



ggplot

Step 1: Initiate your data and aesthetics

```
ggplot(mtcars, aes(x=hp, y=mpg))
```

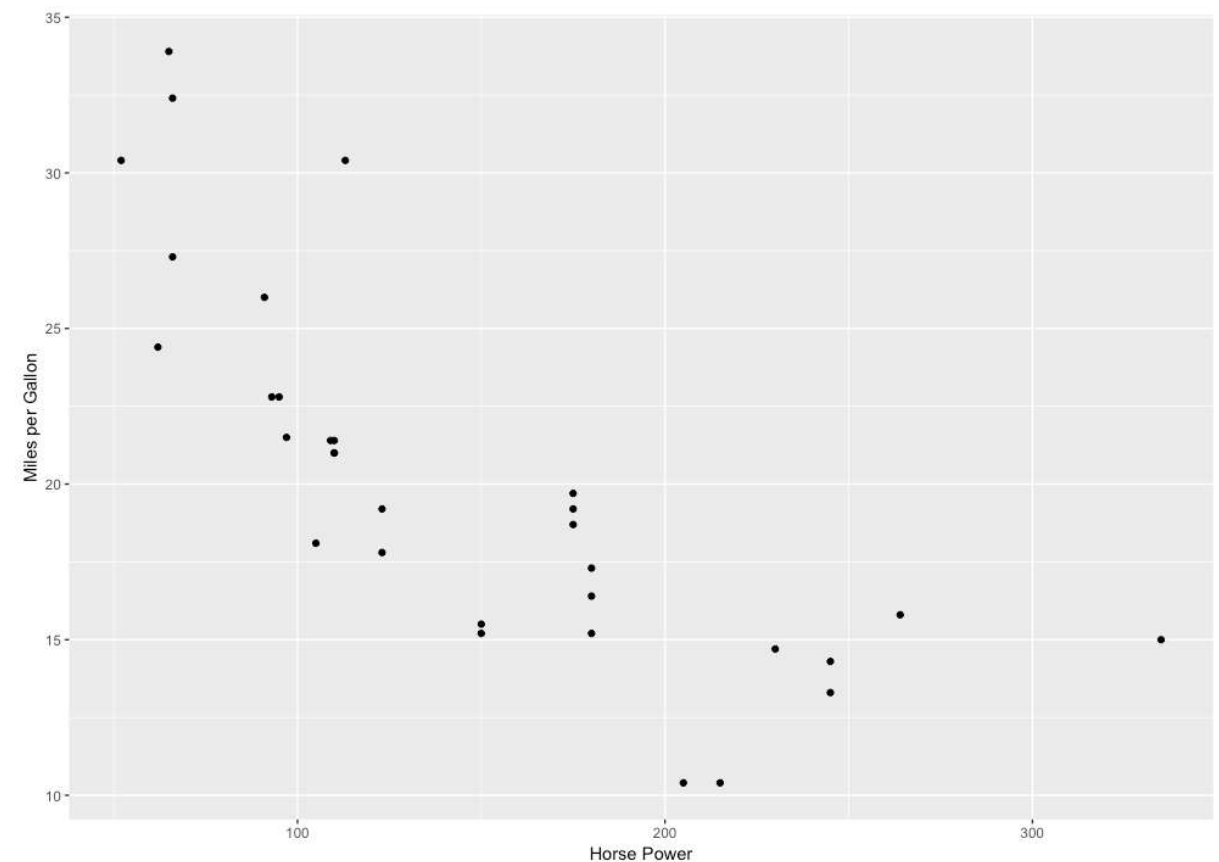
Step 2: Apply geometry

```
geom_point()
```

Step 2: Refine

```
labs(y="Miles per Gallon", x = "Horse Power")
```

```
ggplot(mtcars, aes(x=hp, y=mpg)) +  
  geom_point() +  
  labs(y="Miles per Gallon", x = "Horse Power")
```



dplyr

- **mutate()** adds new variables that are functions of existing variables
- **select()** picks variables based on their names
- **filter()** picks cases based on their values
- **summarise()** reduces multiple values down to a single summary
- **arrange()** changes the ordering of the rows

dplyr::mutate

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

mutate

mutate(mtcars, wt_mpg = wt/mpg)

```
> head(mutate(mtcars, wt_mpg = wt/mpg))
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	wt_mpg
1	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4	0.1247619
2	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4	0.1369048
3	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1	0.1017544
4	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	0.1502336
5	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2	0.1839572
6	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1	0.1911602

dplyr::select

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

select

select(mtcars, mpg, wt)

	mpg	wt
Mazda RX4	21.0	2.620
Mazda RX4 Wag	21.0	2.875
Datsun 710	22.8	2.320
Hornet 4 Drive	21.4	3.215
Hornet Sportabout	18.7	3.440
Valiant	18.1	3.460

select(mtcars, starts_with(cyl))

	cyl	carb
Mazda RX4	6	4
Mazda RX4 Wag	6	4
Datsun 710	4	1
Hornet 4 Drive	6	1
Hornet Sportabout	8	2
Valiant	6	1

dplyr::filter

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

filter



filter(mtcars, mpg > 20 & carb == 4)

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21	6	160	110	3.9	2.620	16.46	0	1	4	4
2	21	6	160	110	3.9	2.875	17.02	0	1	4	4

dplyr::summarise

```
> head(mtcars)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225	105	2.76	3.460	20.22	1	0	3	1

summarise

summarise(mtcars, mean = mean(dis))

	mean
1	230.7219

mtcars %>%
group_by(cyl) %>%
summarise(mean = mean(dis))

	cyl	mean
	<dbl>	<dbl>
1	4	105.
2	6	183.
3	8	353.

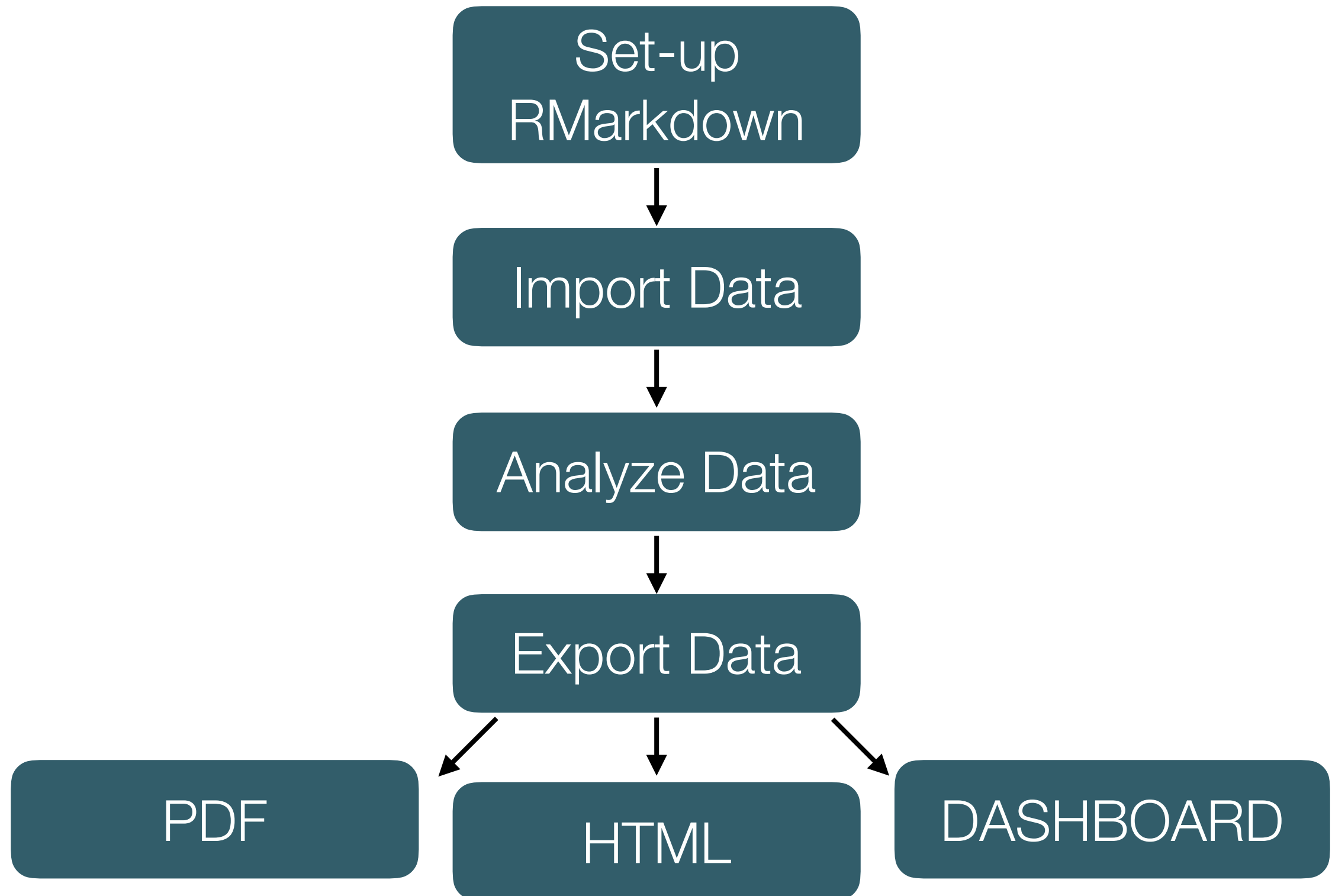
tidyverse example

- Starwars data set

```
> head(starwars)
# A tibble: 6 x 13
  name          height  mass hair_color skin_color eye_color birth_year gender homeworld species films      vehicles  starships
  <chr>         <int> <dbl> <chr>      <chr>      <chr>      <dbl> <chr>  <chr>    <chr> <list>    <list>    <list>
1 Luke Skywalker   172    77 blond      fair        blue        19   male  Tatooine Human  <chr [5]> <chr [2]> <chr [2]>
2 C-3PO            167    75 NA        gold        yellow     112   NA    Tatooine Droid  <chr [6]> <chr [0]> <chr [0]>
3 R2-D2            96     32 NA        white, blue red         33   NA    Naboo    Droid  <chr [7]> <chr [0]> <chr [0]>
4 Darth Vader      202   136 none      white       yellow     41.9 male  Tatooine Human  <chr [4]> <chr [0]> <chr [1]>
5 Leia Organa      150    49 brown     light      brown       19  female Alderaan Human  <chr [5]> <chr [1]> <chr [0]>
6 Owen Lars        178   120 brown, grey light      blue        52   male  Tatooine Human  <chr [3]> <chr [0]> <chr [0]>
```

Basic Data Science Workflow with R and RStudio

R Analysis Workflow Overview



RMarkdown

YAML

Yet Another Markdown Language

```
1 ---
2 title: "Untitled"
3 author: "Scott A. Handley"
4 date: "10/7/2018"
5 output: html_document
6 ---
```

TEXT

```
12 ## R Markdown
13
14 This is an R Markdown document. Markdown is a simple formatting syntax for
    authoring HTML, PDF, and MS Word documents. For more details on using R
    Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button a document will be generated that includes
    both content as well as the output of any embedded R code chunks within the
    document. You can embed an R code chunk like this:
```

CHUNK

```
```{r title}
contents
```
```

```
18 ```{r starwars}
19 head(starwars)
20
21 # Exploratory data analysis
22 ggplot(starwars, aes(x=mass, y=height, color = homeworld)) +
23   geom_point()
24
25 ```
26
```

CHUNK

TEXT

...

EXAMPLE: Star Wars Data