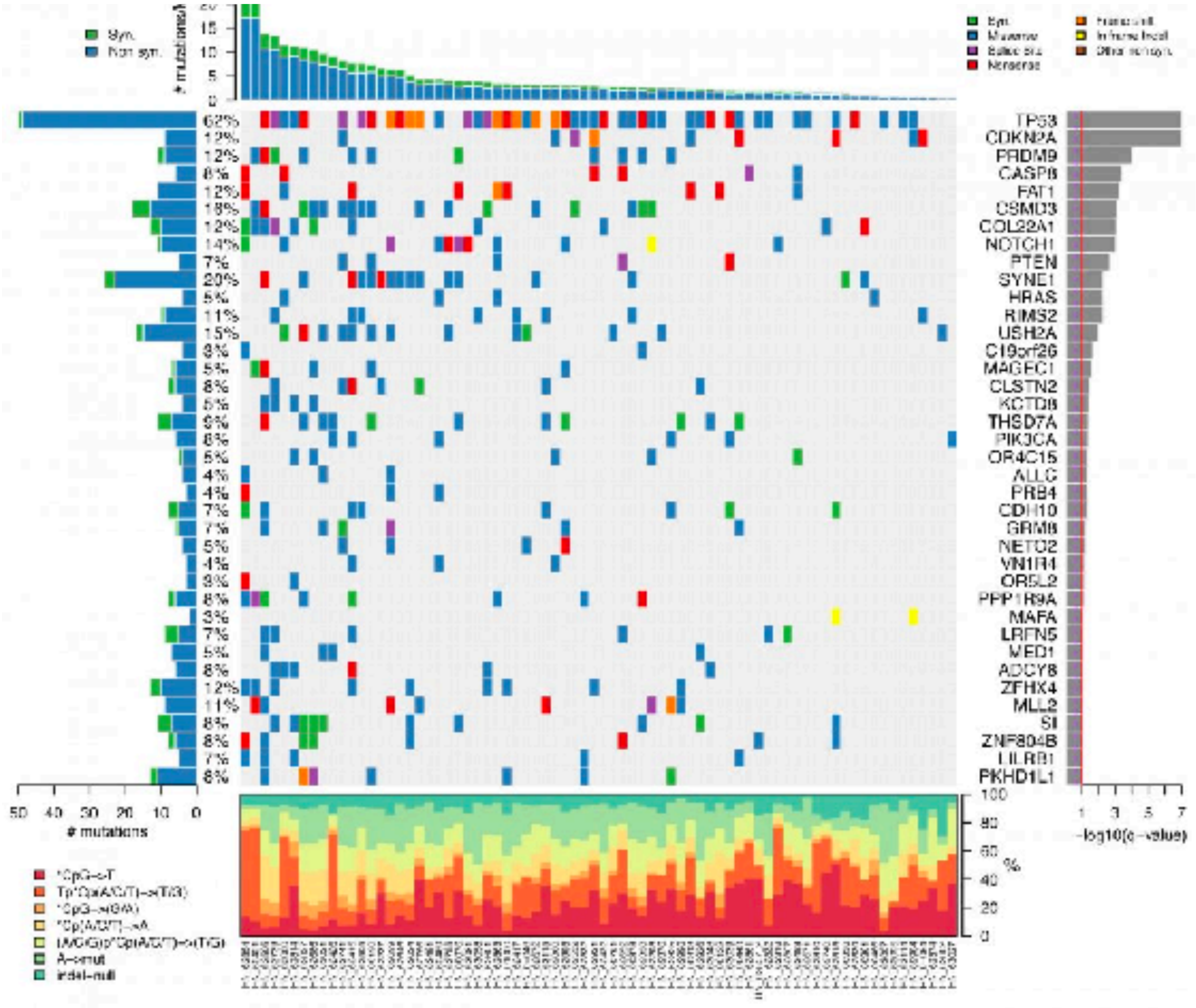
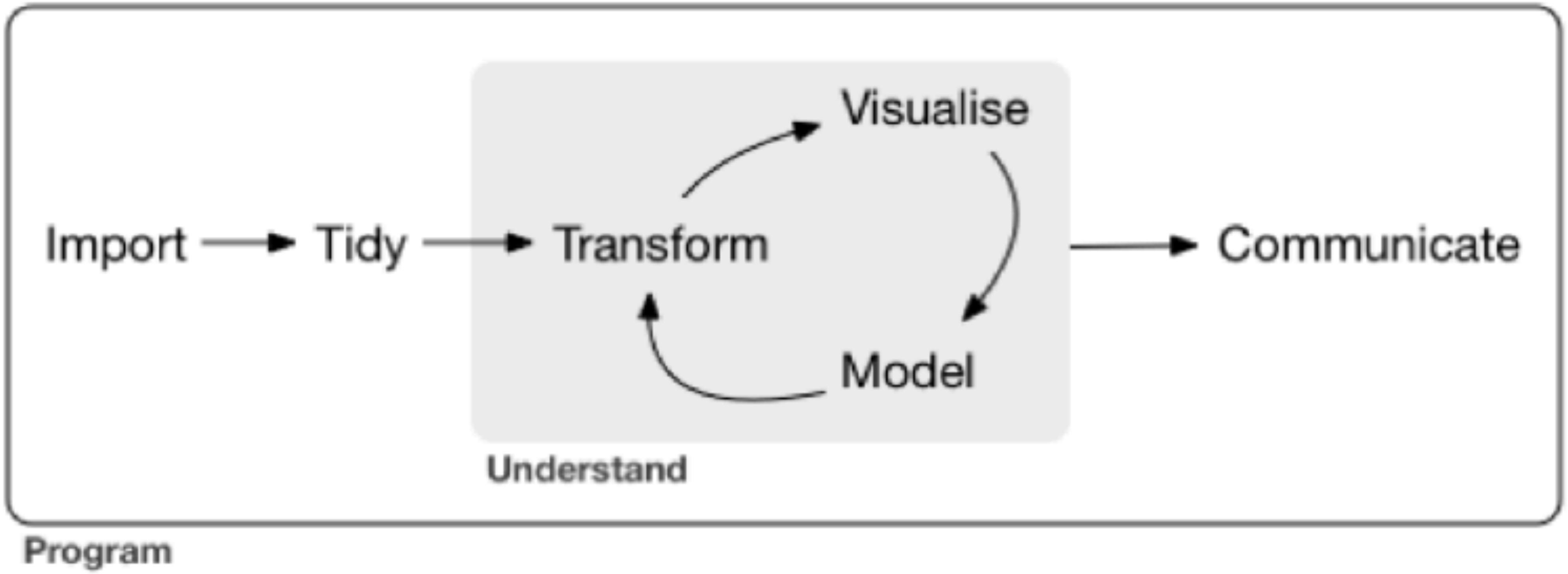
R for data transformation and visualisation



- March 2019
- Brendan Ansell

```
install.packages(c("tidyverse", "ggrepel")); library(tidyverse); library(ggrepel)
```

Job of a Data Scientist

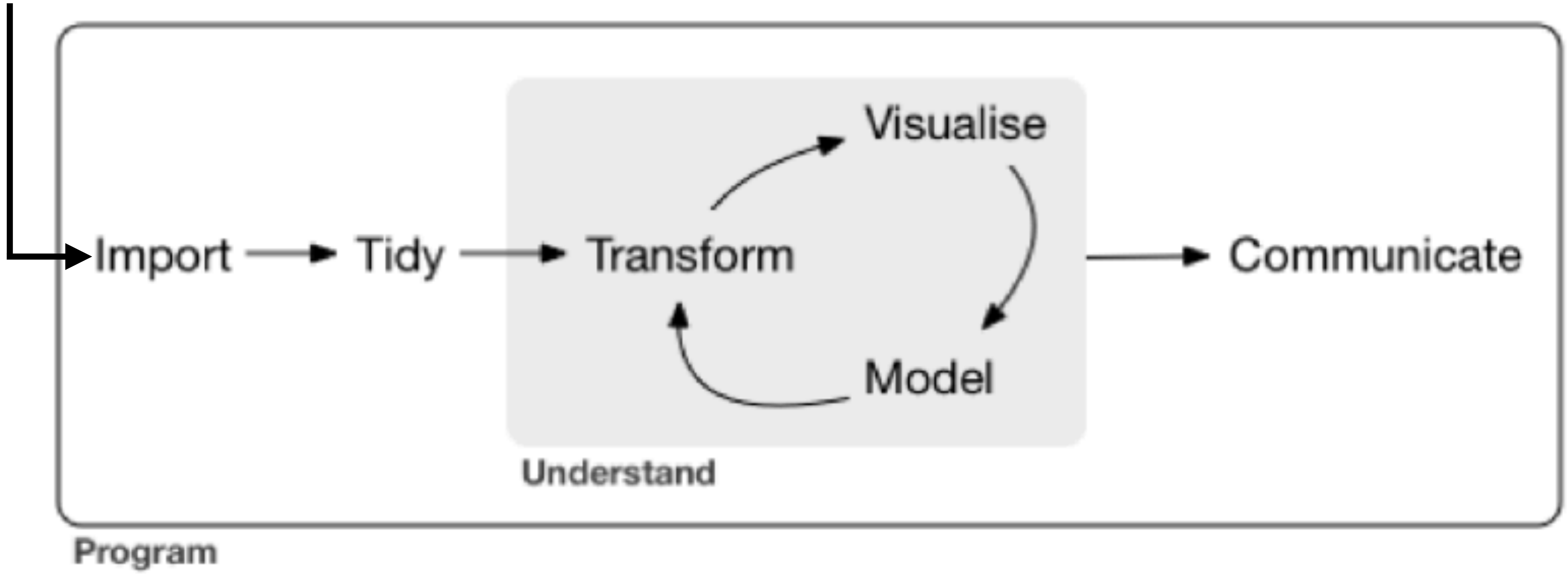


Job of a Biological Scientist

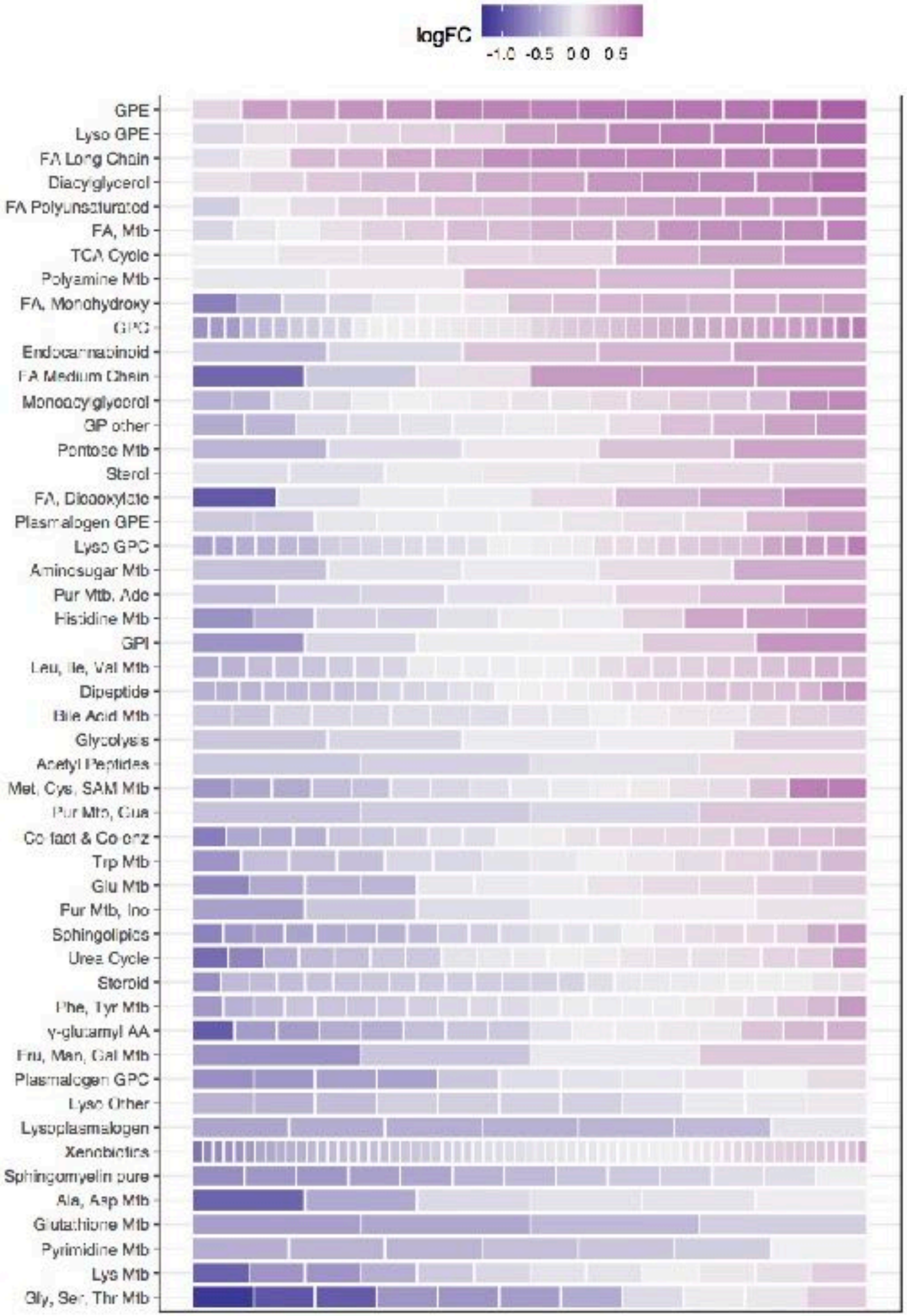
Perform experiment

↓

Generate (gigabytes of) data

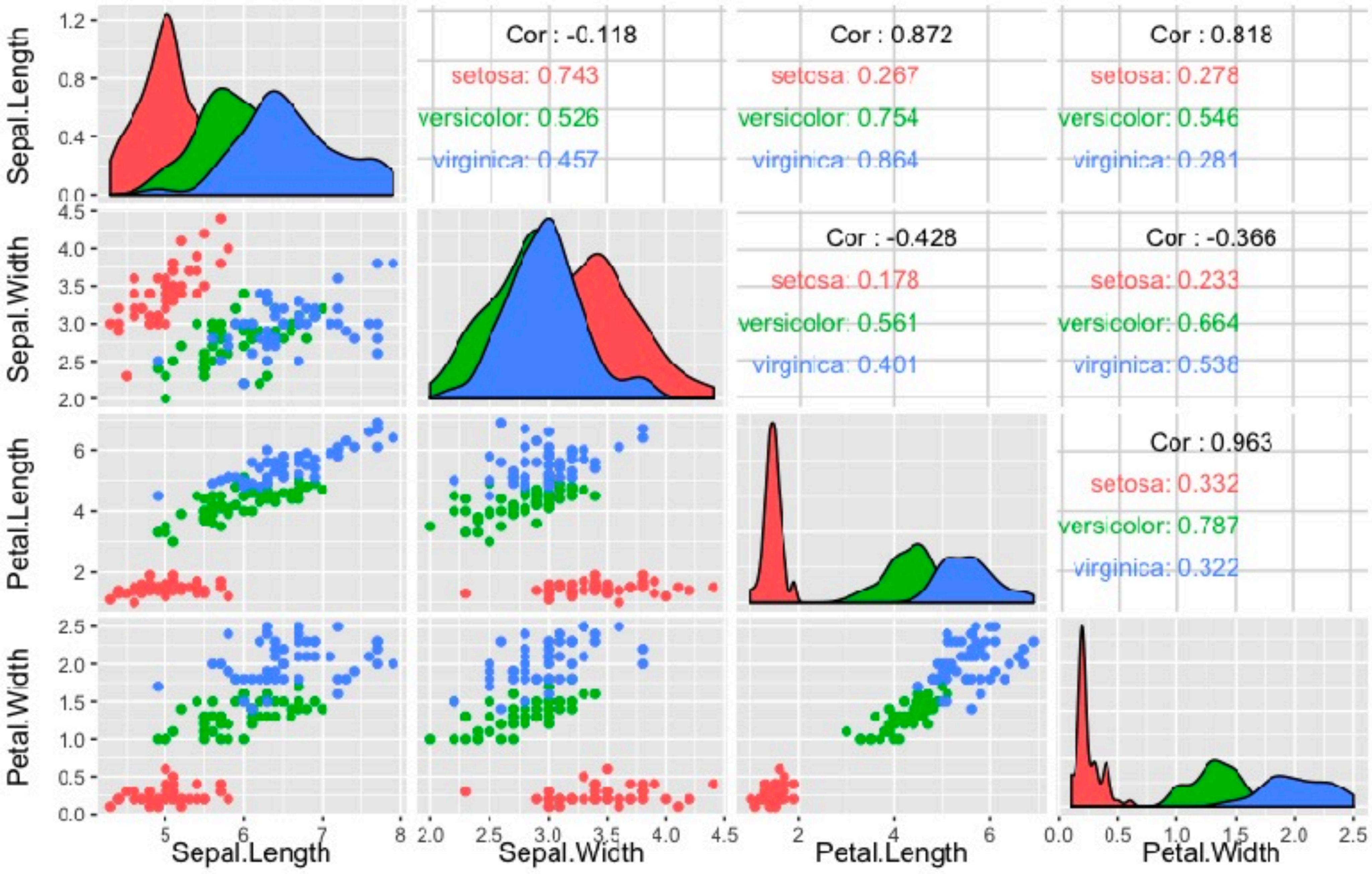


Metabolite Enrichment

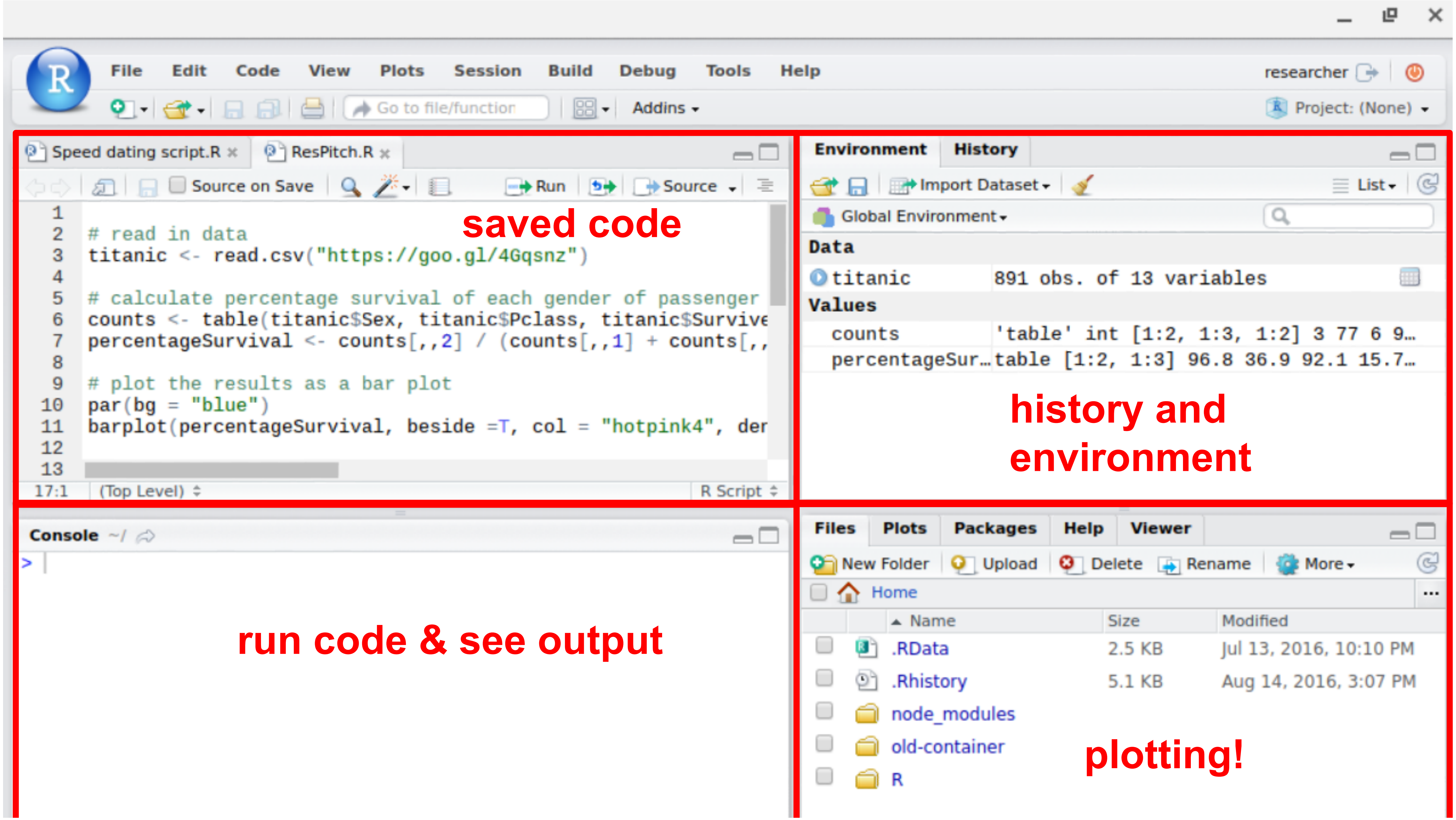


Pair-wise Correlations

Iris phenotypes



RStudio interface



saved code

```

# read in data
titanic <- read.csv("https://goo.gl/4Gqsnz")

# calculate percentage survival of each gender of passenger
counts <- table(titanic$Sex, titanic$Pclass, titanic$Survive
percentageSurvival <- counts[,,2] / (counts[,,1] + counts[,,

# plot the results as a bar plot
par(bg = "blue")
barplot(percentageSurvival, beside =T, col = "hotpink4", der


```

history and environment

Data

titanic	891 obs. of 13 variables

Values

counts	'table' int [1:2, 1:3, 1:2] 3 77 6 9…
percentageSur…	table [1:2, 1:3] 96.8 36.9 92.1 15.7…

run code & see output

```
>
```

plotting!

Name	Size	Modified
.RData	2.5 KB	Jul 13, 2016, 10:10 PM
.Rhistory	5.1 KB	Aug 14, 2016, 3:07 PM
node_modules		
old-container		
R		

Assigning a variable

opt -

example
a <- ‘apple’

Running a line of code

 cmd return

⊞ **ctrl return**

Pipe

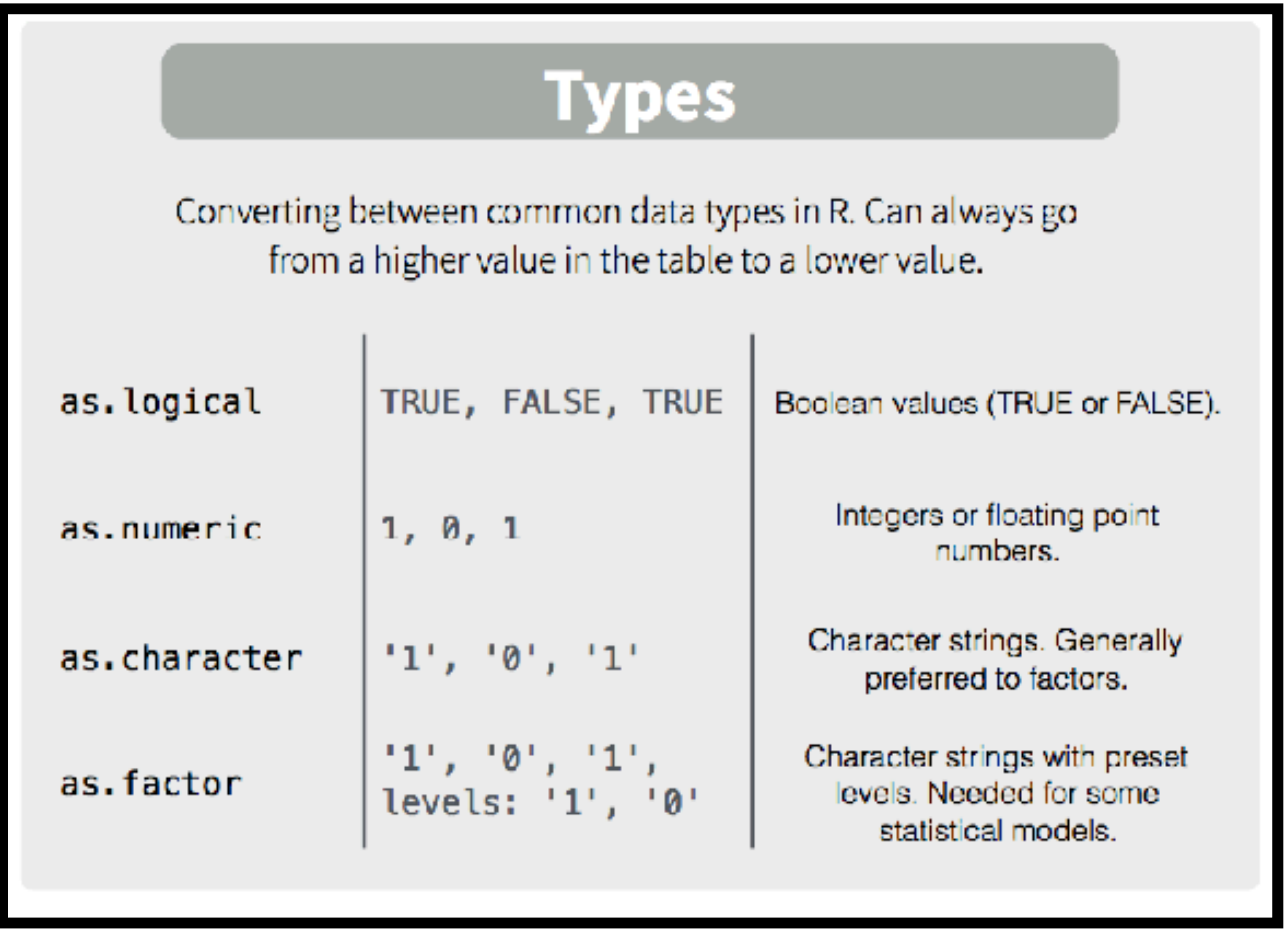
 cmd shift M

⊞ **ctrl shift M**

example
diamonds %>% str()

practice
ds **opt -** diamonds **cmd shift M** str() **cmd return**

ds <- diamonds %>% str()

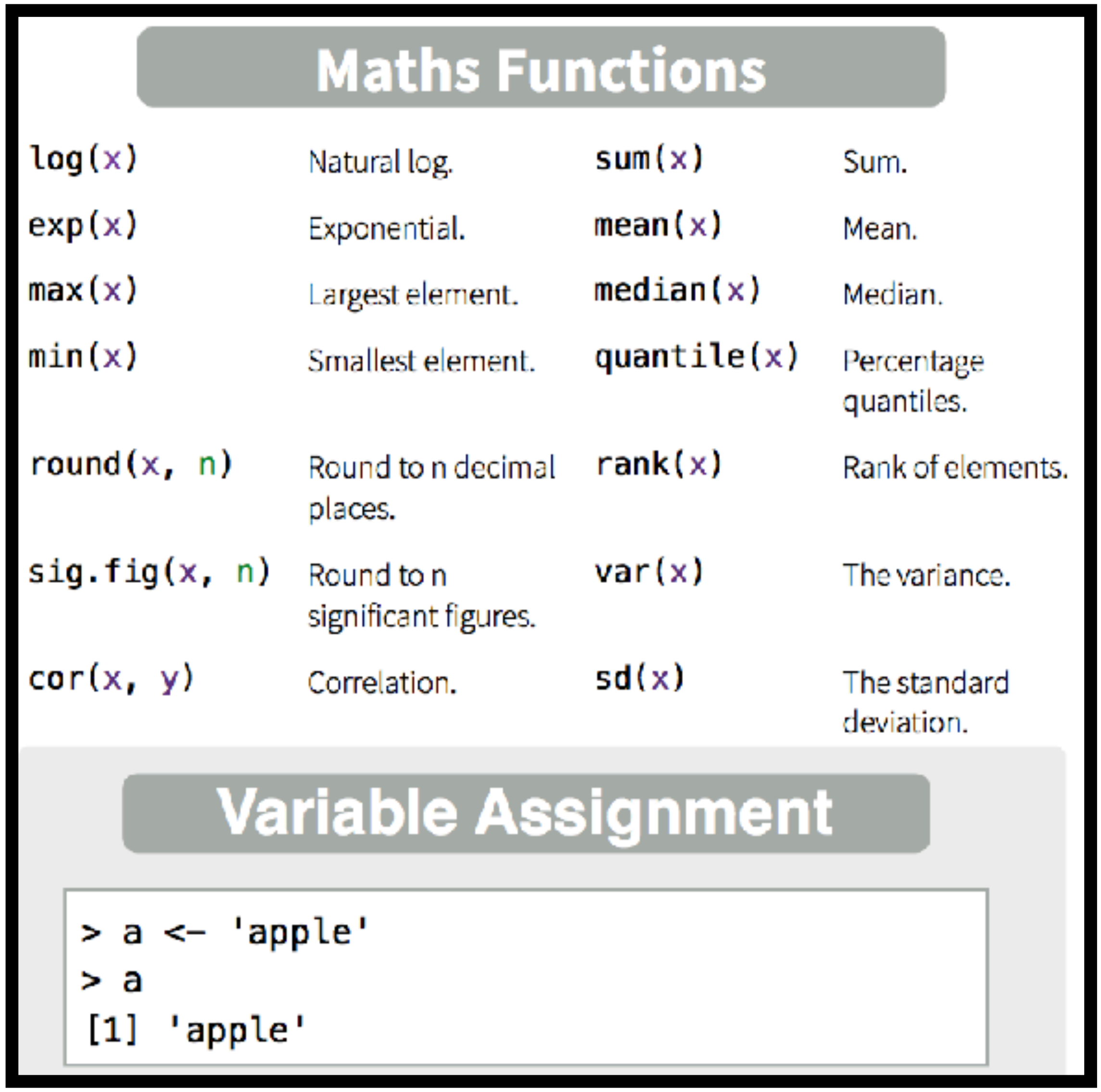


Types

Converting between common data types in R. Can always go from a higher value in the table to a lower value.

`as.logical`	`TRUE, FALSE, TRUE`	Boolean values (TRUE or FALSE).
`as.numeric`	`1, 0, 1`	Integers or floating point numbers.
`as.character`	`'1', '0', '1'`	Character strings. Generally preferred to factors.

`as.factor`	`'1', '0', '1',`
	`levels: '1', '0'`
		Character strings with preset levels. Needed for some statistical models.



Maths Functions

`log(x)`	Natural log.	`sum(x)`	Sum.
`exp(x)`	Exponential.	`mean(x)`	Mean.
`max(x)`	Largest element.	`median(x)`	Median.
`min(x)`	Smallest element.	`quantile(x)`	Percentage quantiles.
`round(x, n)`	Round to n decimal places.	`rank(x)`	Rank of elements.
`sig.fig(x, n)`	Round to n significant figures.	`var(x)`	The variance.
`cor(x, y)`	Correlation.	`sd(x)`	The standard deviation.

Variable Assignment

```
> a <- 'apple'
> a
[1] 'apple'
```

Vectors

Creating Vectors

`c(2, 4, 6)`	`2 4 6`	Join elements into a vector
`2:6`	`2 3 4 5 6`	An integer sequence
`seq(2, 3, by=0.5)`	`2.0 2.5 3.0`	A complex sequence
`rep(1:2, times=3)`	`1 2 1 2 1 2`	Repeat a vector
`rep(1:2, each=3)`	`1 1 1 2 2 2`	Repeat elements of a vector

Grammar of Graphics (ggplot2)

data

a data_frame containing values for plotting

aesthetic mappings **map to individual columns in data_frame**

columns in the data_frame that map to features on the plot
e.g. x axis, y axis, point color, point size

geometric objects **set by you**

What geometric shapes will be used?
geom_**point**() geom_**bar**() geom_**boxplot**() geom_**text**() geom_**histogram**() etc

coordinates **set by you**

control plot layout
coord_cartesian (x vs y) ; coord_map; coord_flipped; coord_polar; coord_equal etc.

faceting **map to individual columns in data_frame**

break data into sub-plots based on a particular grouping e.g. facet_wrap(~ group)

ggplots built up in layers (+)

```
                      mpg %>%
ggplot(aes(x=displ, y=cty)) +
      geom_point(col=“red”) +
           geom_line(lwd=2) +
              geom_smooth() +
             facet_wrap(~cyl)
```

map to individual column in data_frame	**No quotes**
character set by you	**Quotes**
number	**No quotes**