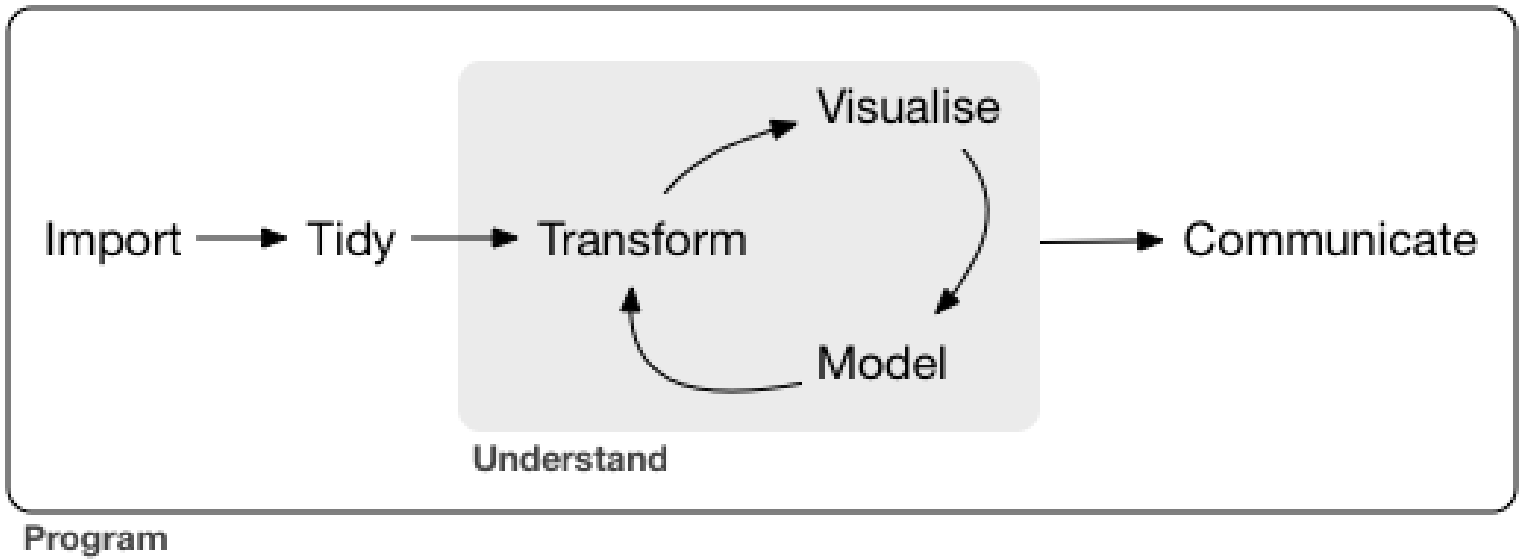
Introduction to R for social researchers

Introduction to R

Data analysis framework



Why use R?

Open source

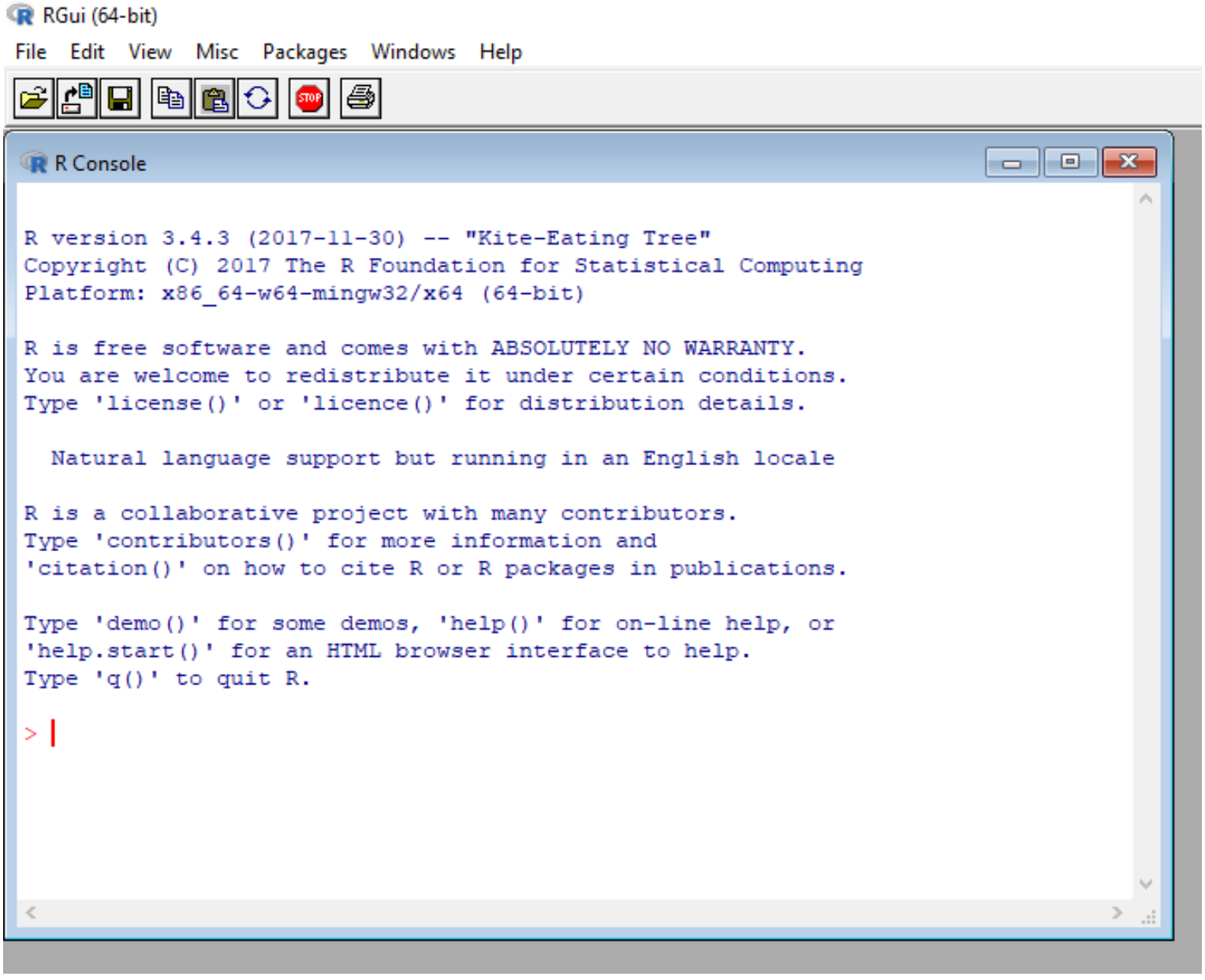
Flexible programming

Wide range of statistical methods

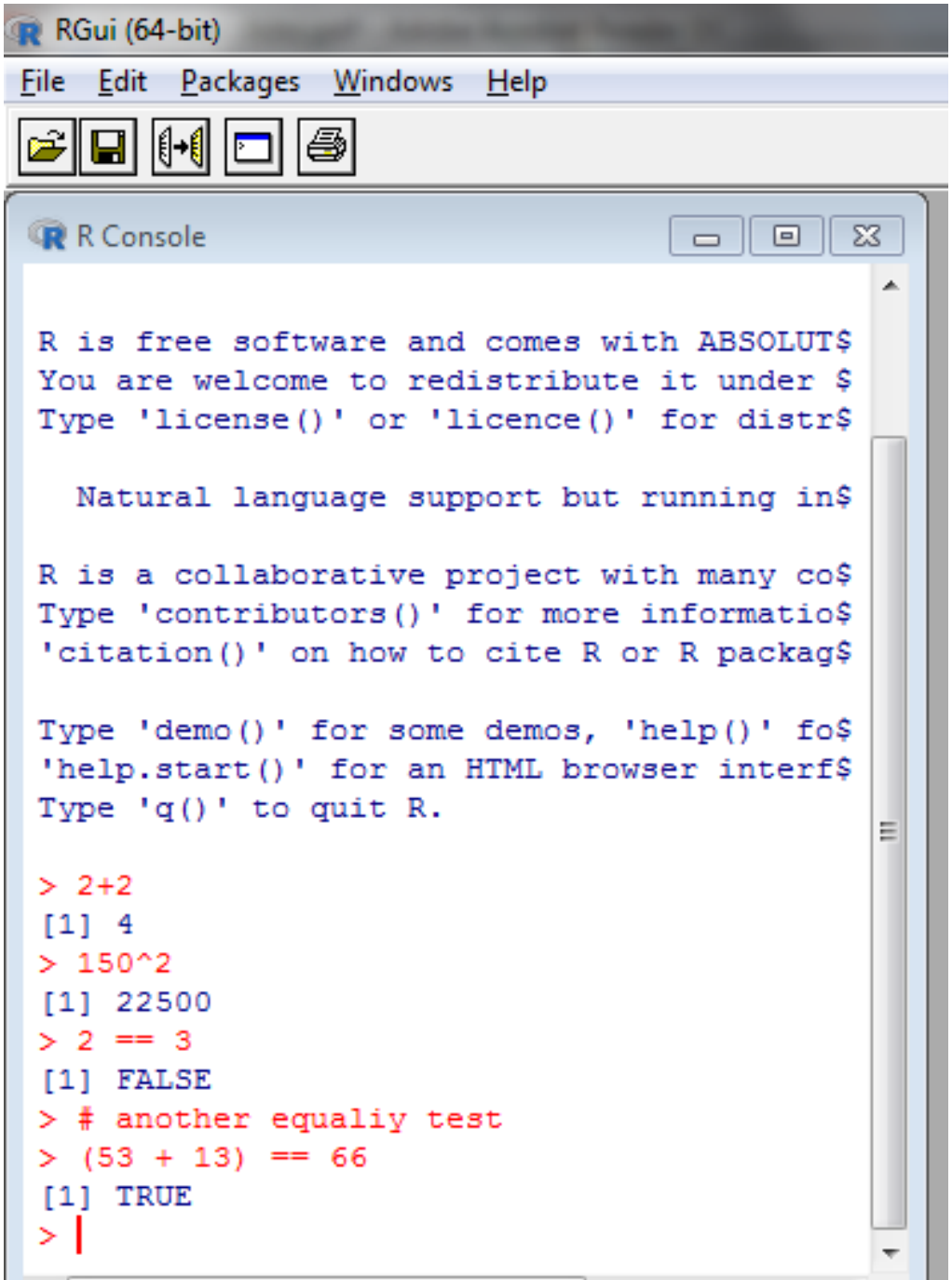
Beautiful graphs

Basics of R

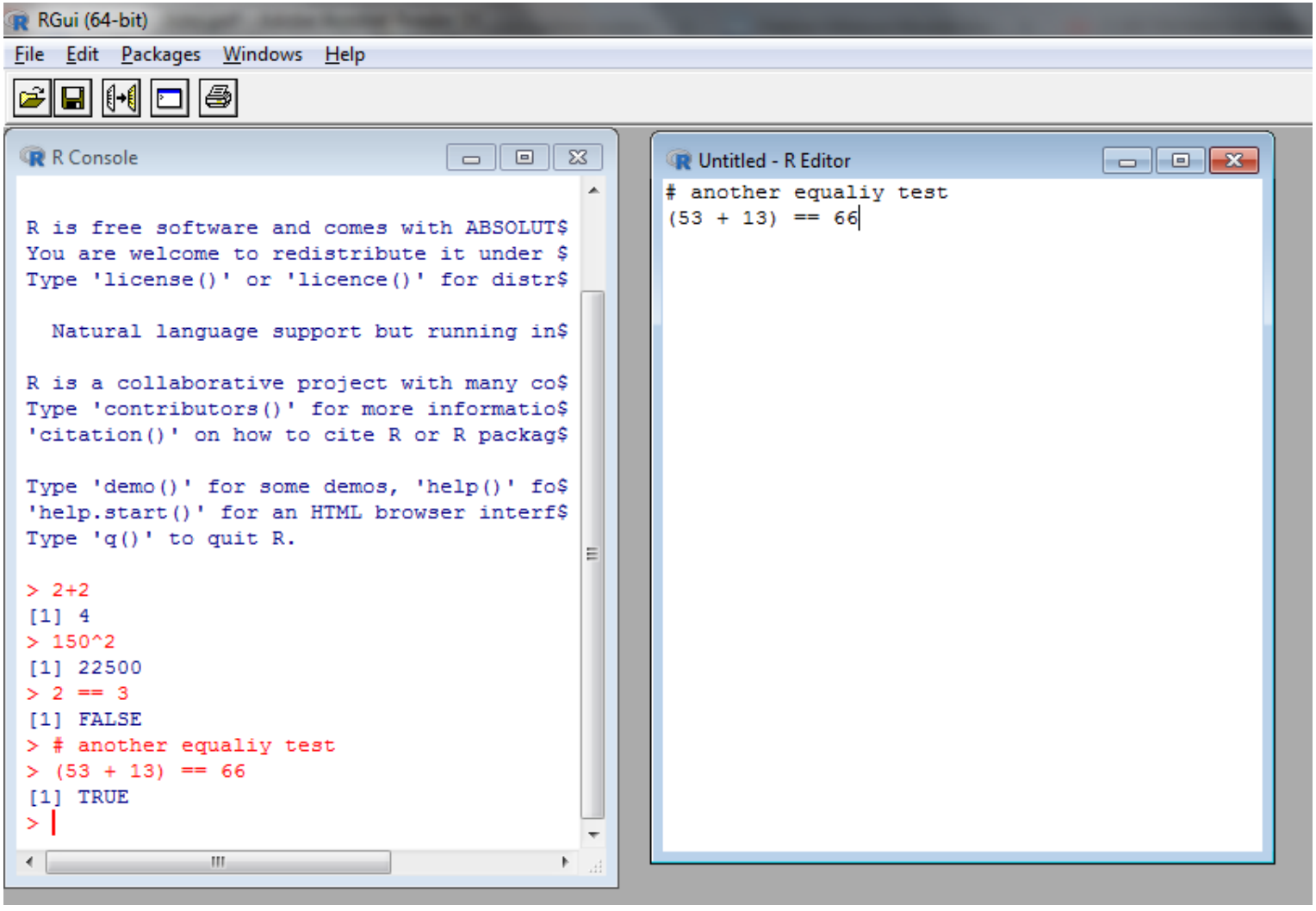
R console


RGui (64-bit)
File Edit View Misc Packages Windows Help
R Console
R version 3.4.3 (2017-11-30) -- "Kite-Eating Tree"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (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 console interactive



R console and script

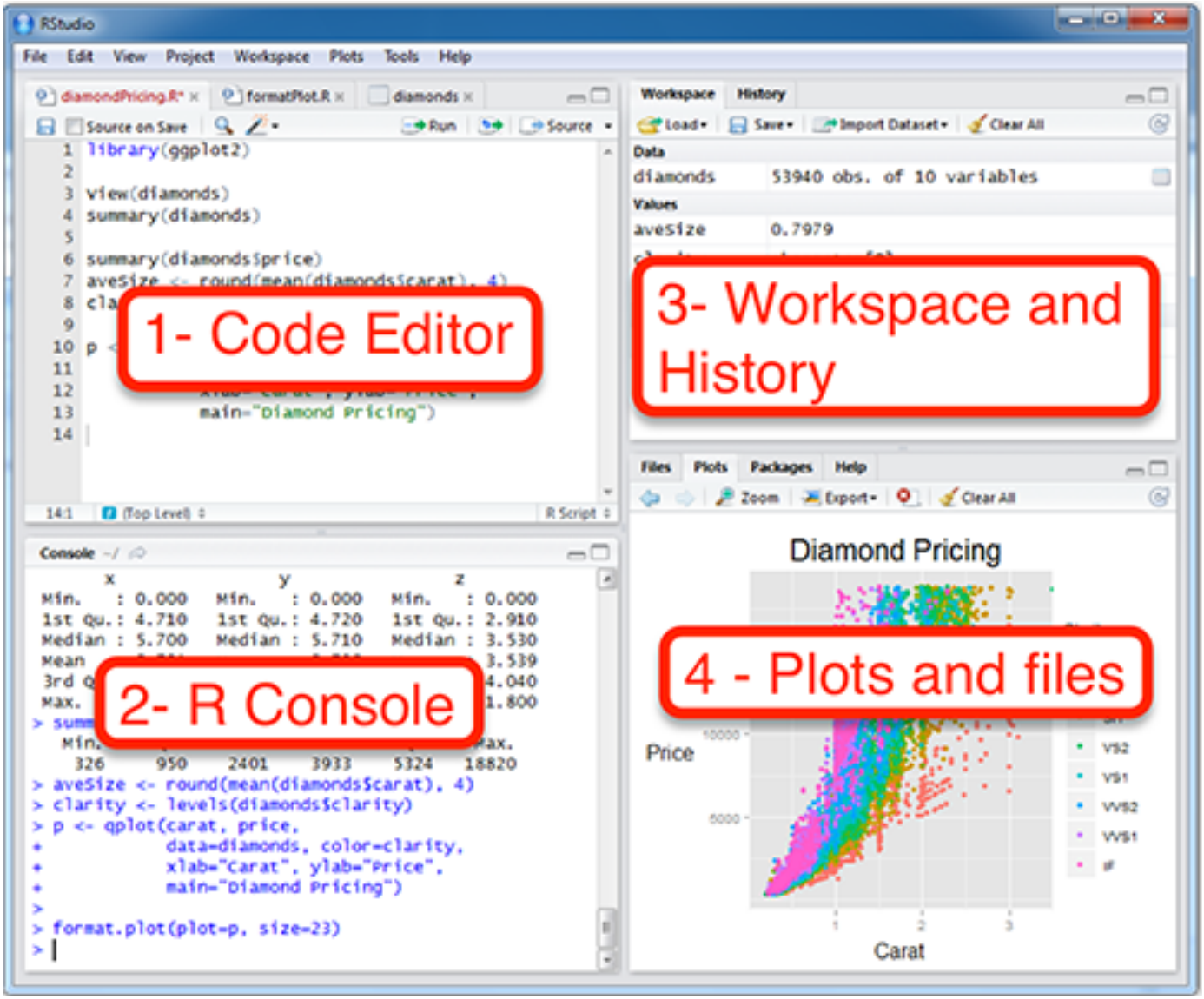


RStudio

Integrated development environment (IDE)

It includes:
console
syntax-highlighting
direct code execution
history
debugging
work-space management

Rstudio interface



Packages

Collections of functions and objects

Thousands free of user-written packages

Can significantly extend the capabilities of R

Package install and load

```r
# install package
install.packages("tidyverse")

# load package
library(tidyverse)
```

Basic work-flow

1. Make new project (optional)
or set up working directory

Basic work-flow

1. Make new project (optional)
or set up working directory

2. Make new R script/syntax
Run syntax using "Ctrl + Enter"

Basic work-flow

1. Make new project (optional)
or set up working directory

2. Make new R script/syntax
Run syntax using "Ctrl + Enter"

3. Save syntax and data for later use

Style guide

Good writing practice is essential
for **future proofing**
and **collaboration**

Main rules:
- comment your script well
- be consistent in writing style

https://style.tidyverse.org/

Main types of objects in R

Creating and using objects

Creating and using objects

```
# giving a value to an object
number <- 3

# call object
number
```

```
## [1] 3
```

```
# using the object in other tasks
number + 5
```

```
## [1] 8
```

Classification of objects

	Homogeneous	**Heterogeneous**
1d	Atomic vector	List
2d	Matrix	Data frame
nd	Array	

Atomic vectors

There are four common ones:

Logical

Integer

Double (numeric)

Character

Double vectors

```
dbl_var <- c(1, 2.5, 4.5)

dbl_var
```

```
## [1] 1.0 2.5 4.5
```

Integer vectors

```
# With the L suffix, you get an integer
int_var <- c(1L, 6L, 10L)

int_var
```

```
## [1]  1  6 10
```

Logical vectors

```
# Use TRUE and FALSE (or T and F) for logical vectors
log_var <- c(TRUE, FALSE, T, F)

log_var
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

Character vectors

```
chr_var <- c("these are", "some strings")

chr_var
```

```
## [1] "these are"    "some strings"
```

Missing data in R

```
miss_var <- c(NA, NA, NA)

miss_var
```

```
## [1] NA NA NA
```

Checking the type of vectors

Checking the type of vectors

```
typeof(int_var)
```

```
## [1] "integer"
```

```
is.integer(int_var)
```

```
## [1] TRUE
```

```
is.character(int_var)
```

```
## [1] FALSE
```

Coercion

```
# when combined they coerce to the most flexible type
c("a", 1)
```

```
## [1] "a" "1"
```

```
typeof(c("a", 1))
```

```
## [1] "character"
```

Coercion

```
log_var
```

```
## [1]  TRUE FALSE  TRUE FALSE
```

```
# logical is coerced to numeric
as.numeric(log_var)
```

```
## [1] 1 0 1 0
```

```
# we can also use in calculations
sum(log_var)
```

```
## [1] 2
```

```
mean(log_var)
```

```
## [1] 0.5
```

Lists

```r
x <- list(1:3, "a", c(TRUE, FALSE, TRUE), c(2.3, 5.9))

str(x)
```

```
## List of 4
##  $ : int [1:3] 1 2 3
##  $ : chr "a"
##  $ : logi [1:3] TRUE FALSE TRUE
##  $ : num [1:2] 2.3 5.9
```

Lists for more complex data structures

Often used as outputs of statistical models

Or when doing programming

Names

Names

```r
x <- c(a = 1, b = 2, c = 3)

# or

x <- 1:3
names(x)
```

```
## NULL
```

```r
names(x) <- c("a", "b", "c")

x
```

```
## a b c
## 1 2 3
```

Factors

Is used to store categorical data

Can contain only predefined values

Factors

```
x <- factor(c("a", "b", "b", "a"))
x
```

```
## [1] a b b a
## Levels: a b
```

```
class(x)
```

```
## [1] "factor"
```

```
levels(x)
```

```
## [1] "a" "b"
```

Factors

```r
sex_char <- c("m", "m", "m")
sex_factor <- factor(sex_char, levels = c("m", "f"))

table(sex_char)
```

```
## sex_char
## m
## 3
```

```r
table(sex_factor)
```

```
## sex_factor
## m f
## 3 0
```

Matrices and data frames

Matrices

A matrix is a type of vector
with two dimensions

Matrices are essential for statistical models

Creating matrices

```r
# Two scalar arguments to specify rows and columns
mat1 <- matrix(1:6, ncol = 3, nrow = 2)
mat1
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Matrix information

```
length(mat1)
```

```
## [1] 6
```

```
nrow(mat1)
```

```
## [1] 2
```

```
ncol(mat1)
```

```
## [1] 3
```

```
rownames(mat1) <- c("A", "B")
colnames(mat1) <- c("a", "b", "c")
mat1
```

```
##   a b c
## A 1 3 5
## B 2 4 6
```

Data frames

Useful way of storing data

List of equal-length vectors (2 dim)

Shares characteristics with matrices and lists

Data frame creation

```r
df <- data.frame(x = 1:3, y = c("a", "b", "c"))

df
```

```
##   x y
## 1 1 a
## 2 2 b
## 3 3 c
```

```r
str(df)
```

```
## 'data.frame':    3 obs. of  2 variables:
##  $ x: int  1 2 3
##  $ y: chr  "a" "b" "c"
```

Combining data frames

```r
# add columns
cbind(df, data.frame(z = 3:1))
```

```
##   x y z
## 1 1 a 3
## 2 2 b 2
## 3 3 c 1
```

```r
# add rows
rbind(df, data.frame(x = 10, y = "z"))
```

```
##    x y
## 1  1 a
## 2  2 b
## 3  3 c
## 4 10 z
```

Subsetting

Positive integers - selection

```
x <- c(2.1, 4.2, 3.3, 5.4)

# select second value
x[2]
```

```
## [1] 4.2
```

```
# select 3rd and 1st value
x[c(3, 1)]
```

```
## [1] 3.3 2.1
```

Positive integers - order

```
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
# order
x[order(x)]
```

```
## [1] 2.1 3.3 4.2 5.4
```

```
# order decreasing
x[order(x, decreasing = T)]
```

```
## [1] 5.4 4.2 3.3 2.1
```

Other subsetting rules

```
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
# Duplicated indices yield duplicated values
x[c(1, 1)]
```

```
## [1] 2.1 2.1
```

```
# Real numbers are silently truncated to integers
x[c(2.1, 2.9)]
```

```
## [1] 4.2 4.2
```

Negative integers

```
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
# exclude third and first values
x[-c(3, 1)]
```

```
## [1] 4.2 5.4
```

```
# can't mix positive and negative
x[c(-1, 2)]
```

Logical vectors

```
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
# TRUE selects cases while FALSE eliminates them
x[c(TRUE, TRUE, FALSE, FALSE)]
```

```
## [1] 2.1 4.2
```

```
# we can also use logical conditions
x > 3
```

```
## [1] FALSE  TRUE  TRUE  TRUE
```

```
x[x > 3]
```

```
## [1] 4.2 3.3 5.4
```

Recycling vectors

```
x
```

```
## [1] 2.1 4.2 3.3 5.4
```

```
# when shorter vector is used it's recycled
x[c(TRUE, FALSE)]
```

```
## [1] 2.1 3.3
```

```
# Equivalent to
x[c(TRUE, FALSE, TRUE, FALSE)]
```

```
## [1] 2.1 3.3
```

If names are present they can be used

```
(y <- setNames(x, letters[1:4]))
```

```
##   a   b   c   d
## 2.1 4.2 3.3 5.4
```

```
y[c("d", "c", "a")]
```

```
##   d   c   a
## 5.4 3.3 2.1
```

```
# They work like numeric indices
y[c("a", "a", "a")]
```

```
##   a   a   a
## 2.1 2.1 2.1
```

Selection in matrices

Matrices

```r
a <- matrix(1:9, nrow = 3)
colnames(a) <- c("A", "B", "C")
a
```

```
##      A B C
## [1,] 1 4 7
## [2,] 2 5 8
## [3,] 3 6 9
```

```r
# select the first 2 rows
a[1:2, ]
```

```
##      A B C
## [1,] 1 4 7
## [2,] 2 5 8
```

Matrices

```r
# select some rows and columns
a[c(TRUE, FALSE, TRUE), c("B", "A")]
```

```
##      B A
## [1,] 4 1
## [2,] 6 3
```

Data frames - select coloumns

```r
# make data
(df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3]))
```

```
##   x y z
## 1 1 3 a
## 2 2 2 b
## 3 3 1 c
```

```r
# select variable using "$" + name
df$z
```

```
## [1] "a" "b" "c"
```

Data frames - select rows

```r
# can select rows and columns like in matrices
df[c(1, 3), ]
```

```
##   x y z
## 1 1 3 a
## 3 3 1 c
```

```r
# can also test conditions
df$x == 2
```

```
## [1] FALSE  TRUE FALSE
```

```r
df[df$x == 2, ]
```

```
##   x y z
## 2 2 2 b
```

Data frames - select variables using names

```
df[c("x", "z")]
```

```
##   x z
## 1 1 a
## 2 2 b
## 3 3 c
```

```
df[, c("x", "z")]
```

```
##   x z
## 1 1 a
## 2 2 b
## 3 3 c
```

Subsetting and assignment

Subsetting and assignment

```
x <- 1:5
x[c(1, 2)]
```

```
## [1] 1 2
```

```
x[c(1, 2)] <- 2:3
x
```

```
## [1] 2 3 3 4 5
```

```
# The length of the LHS needs to match the RHS
x[-1] <- 4:1
x
```

```
## [1] 2 4 3 2 1
```

Conditional assignment and missing

```
df <- data.frame(a = c(1, 10, NA))
df$a < 5
```

```
## [1]  TRUE FALSE    NA
```

```
df$a[df$a < 5] <- 0
df$a
```

```
## [1]  0 10 NA
```

Practical 1

Some applications

Ordering

```
x <- c("b", "c", "a")
order(x)
```

```
## [1] 3 1 2
```

```
x[order(x)]
```

```
## [1] "a" "b" "c"
```

```
x[order(x, decreasing = T)]
```

```
## [1] "c" "b" "a"
```

Ordering df

```
# create a new variable
df$b <- 3:1
df
```

```
##    a b
## 1  0 3
## 2 10 2
## 3 NA 1
```

```
# ordering rows by column b
df[order(df$b), ]
```

```
##    a b
## 3 NA 1
## 2 10 2
## 1  0 3
```

Removing columns from data frames

```r
df <- data.frame(x = 1:3, y = 3:1, z = letters[1:3])

df$z <- NULL

# or

df[c("x", "y")]
```

Selecting rows based on a condition

```r
mtcars[mtcars$gear == 5, ]

mtcars[mtcars$gear == 5 & mtcars$cyl == 4, ]

# or

subset(mtcars, gear == 5)
```

Importing and exporting data

Data types

Multiple types of data

- Comma Separated Values - csv (delimited)

- Statistical software specific - sav/dat/sas

- eXtensible Markup Language - XML

- Structured Query Language - SQL

R specific data types

.Rdata - can be used to save multiple objects

.rds - efficient way of saving one dataset

Examples saving and importing RDS

```r
# save RDS data
saveRDS(mtcars, "mtcars.RDS")

# import RDS data
mtcars2 <- readRDS("mtcars.RDS")

# check data
mtcars2
```

```
##                    mpg cyl  disp  hp drat    wt  qsec
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90
```

Examples saving and importing Rdata

```r
# save RData
save(mtcars, cars, file = "mtcars.RData")

# import RData
load("mtcars.RData")

# check dataset
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

You can also save all objects using this format

```
# save all objects in memory
save.image("all_objects.RData")

# load RData
load("all_objects.RData")
```

Importing and exporting csv data

```r
# write a csv file
write.csv(mtcars, "mtcars.csv")

# read csv file
mtcars3 <- read.csv("mtcars.csv")

# check data
head(mtcars3)
```

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

Importing other types of data

```
# load heaven (already installed)
library(haven)

# read Stata data from a subfolder
ess9_raw <- read_dta("./data/ESS9e01_2.dta")

# check data
head(ess9_raw)
```

```
## # A tibble: 6 x 492
##   name    essround edition proddate   idno cntry nwspol
##   <chr>      <dbl> <chr>   <chr>     <dbl> <chr> <dbl+>
## 1 ESS9e0~        9 1.2     30.01.20~    17 AT         60
## 2 ESS9e0~        9 1.2     30.01.20~    31 AT         60
## 3 ESS9e0~        9 1.2     30.01.20~    58 AT         60
## 4 ESS9e0~        9 1.2     30.01.20~    69 AT         20
## 5 ESS9e0~        9 1.2     30.01.20~    98 AT         10
## 6 ESS9e0~        9 1.2     30.01.20~   145 AT         45
## # ... with 483 more variables: pplstrst <dbl+lbl>, pplfair
```

Describing data

Selecting variables of interst

```r
# object with variable names
vars <- c("idno", "cntry", "gndr", "eduyrs",
          "vote", "marsts", "ppltrst")

# select variables we want and make smaller data
ess9 <- ess9_raw[vars]
```

Get to know the data 1

```
# dimensions
dim(ess9)
```

```
## [1] 36015     7
```

```
# top of the data
head(ess9)
```

```
## # A tibble: 6 x 7
##    idno cntry     gndr  eduyrs     vote
##   <dbl> <chr> <dbl+lb> <dbl+l> <dbl+lb>
## 1    17 AT    1 [Male]      12  1 [Yes]      4 [Legally c
## 2    31 AT    2 [Fema~       8  1 [Yes]      5 [Widowed/C
## 3    58 AT    1 [Male]      11  1 [Yes] NA(a) [Not appli
## 4    69 AT    2 [Fema~      12  2 [No]       6 [None of t
## 5    98 AT    1 [Male]      12  1 [Yes] NA(a) [Not appli
## 6   145 AT    1 [Male]      11  1 [Yes] NA(a) [Not appli
```

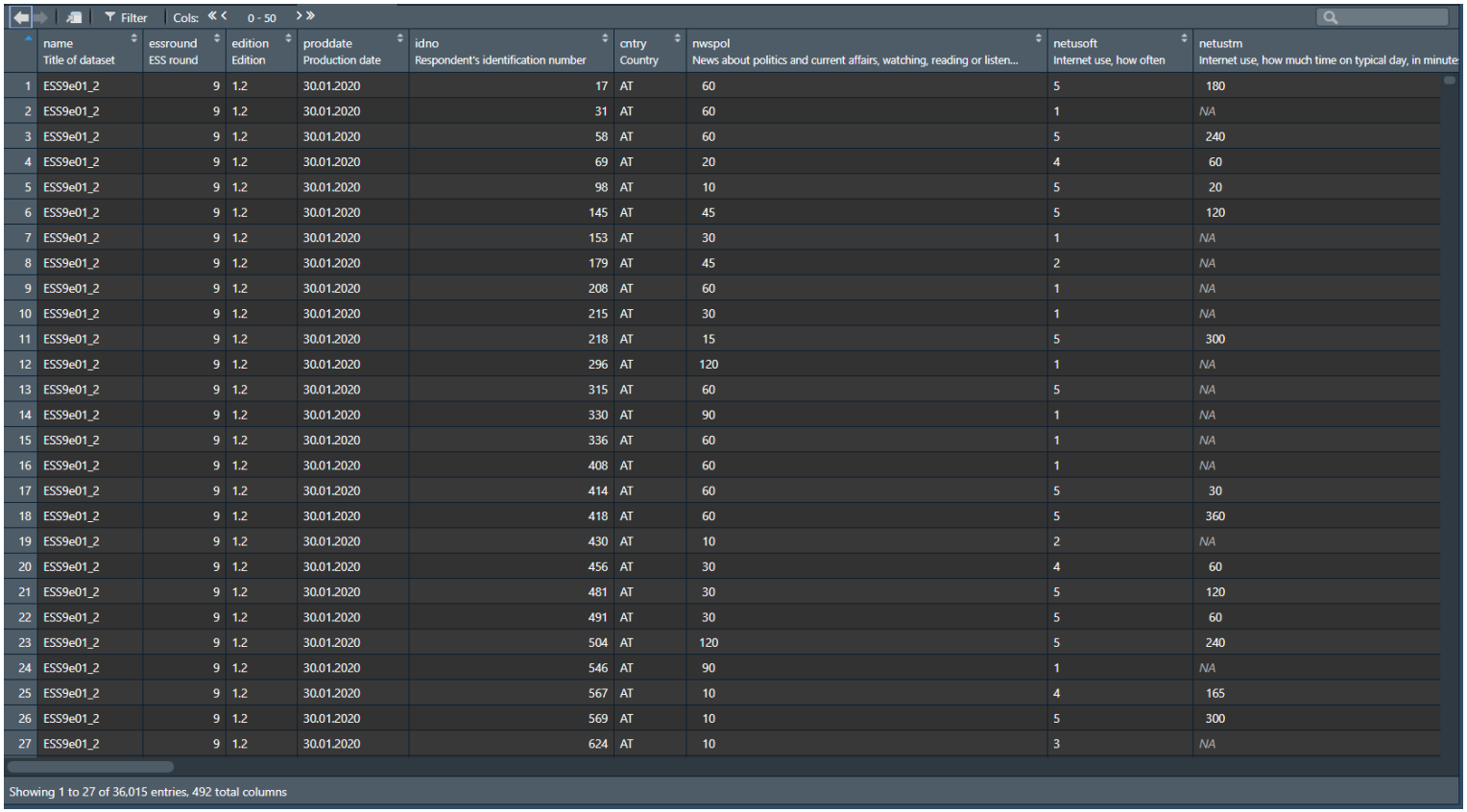
Get to know the data 2

```
# last rows of the data
tail(ess9)
```

```
## # A tibble: 6 x 7
##     idno cntry     gndr  eduyrs        vote
##    <dbl> <chr> <dbl+lb> <dbl+l>   <dbl+lbl>
## 1 48235 SI    2 [Fema~       8 1 [Yes]         5 [Widowe
## 2 48329 SI    1 [Male]       9 3 [Not eli~     6 [None o
## 3 48484 SI    1 [Male]      12 1 [Yes]     NA(a) [Not ap
## 4 48535 SI    2 [Fema~       8 1 [Yes]         5 [Widowe
## 5 48546 SI    1 [Male]      13 1 [Yes]         6 [None o
## 6 48554 SI    2 [Fema~      14 1 [Yes]     NA(a) [Not ap
```

Get to know the data 3

```
# to see it like in excel
View(ess9)
```



Filter | Cols: « < 0 - 50 > »

	name Title of dataset	essround ESS round	edition Edition	proddate Production date	idno Respondent's identification number	cntry Country	nwspol News about politics and current affairs, watching, reading or listen...	netusoft Internet use, how often	netustm Internet use, how much time on typical day, in minute
1	ESS9e01_2	9	1.2	30.01.2020	17	AT	60	5	180
2	ESS9e01_2	9	1.2	30.01.2020	31	AT	60	1	NA
3	ESS9e01_2	9	1.2	30.01.2020	58	AT	60	5	240
4	ESS9e01_2	9	1.2	30.01.2020	69	AT	20	4	60
5	ESS9e01_2	9	1.2	30.01.2020	98	AT	10	5	20
6	ESS9e01_2	9	1.2	30.01.2020	145	AT	45	5	120
7	ESS9e01_2	9	1.2	30.01.2020	153	AT	30	1	NA
8	ESS9e01_2	9	1.2	30.01.2020	179	AT	45	2	NA
9	ESS9e01_2	9	1.2	30.01.2020	208	AT	60	1	NA
10	ESS9e01_2	9	1.2	30.01.2020	215	AT	30	1	NA
11	ESS9e01_2	9	1.2	30.01.2020	218	AT	15	5	300
12	ESS9e01_2	9	1.2	30.01.2020	296	AT	120	1	NA
13	ESS9e01_2	9	1.2	30.01.2020	315	AT	60	5	NA
14	ESS9e01_2	9	1.2	30.01.2020	330	AT	90	1	NA
15	ESS9e01_2	9	1.2	30.01.2020	336	AT	60	1	NA
16	ESS9e01_2	9	1.2	30.01.2020	408	AT	60	1	NA
17	ESS9e01_2	9	1.2	30.01.2020	414	AT	60	5	30
18	ESS9e01_2	9	1.2	30.01.2020	418	AT	60	5	360
19	ESS9e01_2	9	1.2	30.01.2020	430	AT	10	2	NA
20	ESS9e01_2	9	1.2	30.01.2020	456	AT	30	4	60
21	ESS9e01_2	9	1.2	30.01.2020	481	AT	30	5	120
22	ESS9e01_2	9	1.2	30.01.2020	491	AT	30	5	60
23	ESS9e01_2	9	1.2	30.01.2020	504	AT	120	5	240
24	ESS9e01_2	9	1.2	30.01.2020	546	AT	90	1	NA
25	ESS9e01_2	9	1.2	30.01.2020	567	AT	10	4	165
26	ESS9e01_2	9	1.2	30.01.2020	569	AT	10	5	300
27	ESS9e01_2	9	1.2	30.01.2020	624	AT	10	3	NA

Showing 1 to 27 of 36,015 entries, 492 total columns

Get to know the data 4

```
colnames(ess9)
```

```
## [1] "idno"    "cntry"   "gndr"    "eduyrs"  "vote"    "n
```

Get to know the data 5

```
# some data come with attributes
attributes(ess9$ppltrst)
```

```
## $label
## [1] "Most people can be trusted or you can't be too careful"
##
## $format.stata
## [1] "%26.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##   You can't be too careful                          1
##                          0                          1
##                          2                          3
##                          2                          3
##                          4                          5
##                          4                          5
##                          6                          7
##                          6                          7
##                          8                          9
##                          8                          9
## Most people can be trusted                    Refusal
##                         10                         NA
##                 Don't know                 No answer
##                         NA                         NA
```

Summarise the data

```
summary(ess9)
```

```
##       idno          cntry                gndr          eduyrs
##  Min.   :    1   Length:36015       Min.   :1.000   Min.   : 0.00
##  1st Qu.:12281   Class :character   1st Qu.:1.000   1st Qu.:11.00
##  Median :24378   Mode  :character   Median :2.000   Median :12.00
##  Mean   :24375                      Mean   :1.528   Mean   :13.01
##  3rd Qu.:36471                      3rd Qu.:2.000   3rd Qu.:16.00
##  Max.   :48642                      Max.   :2.000   Max.   :51.00
##                                                     NA's   :505
##       vote           marsts         ppltrst
##  Min.   :1.000   Min.   :1.000   Min.   : 0.000
##  1st Qu.:1.000   1st Qu.:5.000   1st Qu.: 3.000
##  Median :1.000   Median :6.000   Median : 5.000
##  Mean   :1.368   Mean   :5.163   Mean   : 5.094
##  3rd Qu.:2.000   3rd Qu.:6.000   3rd Qu.: 7.000
##  Max.   :3.000   Max.   :6.000   Max.   :10.000
##  NA's   :412     NA's   :17209   NA's   :109
```

Do your own summary statistics

```
# when you have missing the result is missing
mean(ess9$ppltrst)
```

```
## [1] NA
```

```
# use "na.rm" option to ignore missing
mean(ess9$ppltrst, na.rm = T)
```

```
## [1] 5.094051
```

```
median(ess9$ppltrst, na.rm = T)
```

```
## [1] 5
```

```
sd(ess9$ppltrst, na.rm = T)
```

```
## [1] 2.447792
```

```
var(ess9$ppltrst, na.rm = T)
```

```
## [1] 5.991683
```

Doing tables

```
# table of voting
table(ess9$vote)
```

```
##
##     1     2     3
## 25447  7208  2948
```

```
# find out what the numbers mean
attributes(ess9$vote)
```

```
## $label
## [1] "Voted last national election"
##
## $format.stata
## [1] "%20.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##                  Yes                   No Not eligible to vote
##                    1                    2                    3
##              Refusal           Don't know           No answer
##                   NA                   NA                   NA
```

Cross-tables

```r
# table of voting by country
tab1 <- table(ess9$cntry, ess9$vote)

tab1
```

```
##
##          1    2    3
##    AT 1996  344  145
##    BE 1349  180  231
##    BG 1632  413   86
##    CH  736  382  367
##    CY  538  181   58
##    CZ 1459  847   82
##    DE 1860  259  238
##    EE 1166  467  266
##    FI 1381  207  154
##    FR 1173  573  223
##    GB 1663  419  116
##    HU 1183  420   78
##    IE 1637  392  178
##    IT 1976  506  202
##    NL 1291  247  130
##    NO 1156  124  125
##    PL  967  382  130
##    RS 1454  448   74
##    SI  830  417   65
```

Get proportions in tables

```r
tab2 <- prop.table(tab1, 1) # proportion by row

round(tab2, 2) # print and round to 2 decimals
```

```
##
##          1    2    3
##    AT 0.80 0.14 0.06
##    BE 0.77 0.10 0.13
##    BG 0.77 0.19 0.04
##    CH 0.50 0.26 0.25
##    CY 0.69 0.23 0.07
##    CZ 0.61 0.35 0.03
##    DE 0.79 0.11 0.10
##    EE 0.61 0.25 0.14
##    FI 0.79 0.12 0.09
##    FR 0.60 0.29 0.11
##    GB 0.76 0.19 0.05
##    HU 0.70 0.25 0.05
##    IE 0.74 0.18 0.08
##    IT 0.74 0.19 0.08
##    NL 0.77 0.15 0.08
##    NO 0.82 0.09 0.09
##    PL 0.65 0.26 0.09
##    RS 0.74 0.23 0.04
##    SI 0.63 0.32 0.05
```

Modifying variables

Create new variable

```
table(ess9$gndr)
```

```
##
##     1     2
## 16982 19033
```

```
attributes(ess9$gndr)
```

```
## $label
## [1] "Gender"
##
## $format.stata
## [1] "%12.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##      Male    Female No answer
##         1         2        NA
```

Create new variable

```
ess9$female <- ess9$gndr - 1

table(ess9$gndr, ess9$female)
```

```
##
##          0     1
##    1 16982     0
##    2     0 19033
```

Create new variable - factor

```
ess9$gndr_fct <- factor(ess9$gndr,
                        labels = c("Male", "Female"))

table(ess9$gndr, ess9$gndr_fct)
```

```
##
##       Male Female
##    1 16982      0
##    2     0  19033
```

```
table(ess9$gndr_fct, ess9$female)
```

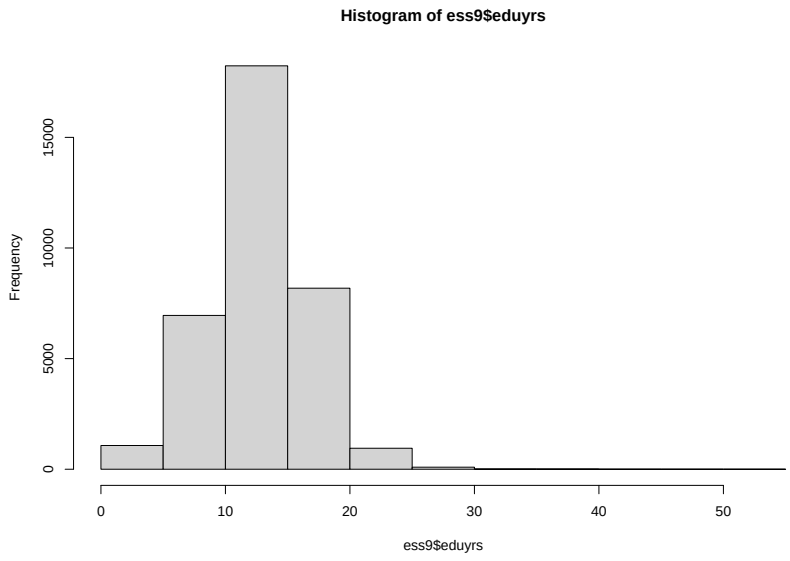
```
##
##                0     1
##    Male    16982     0
##    Female      0 19033
```

Helper function - ifelse

```
summary(ess9$eduyrs)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   11.00   12.00   13.01   16.00   51.00     505
```

```
hist(ess9$eduyrs)
```



Helper function - ifelse

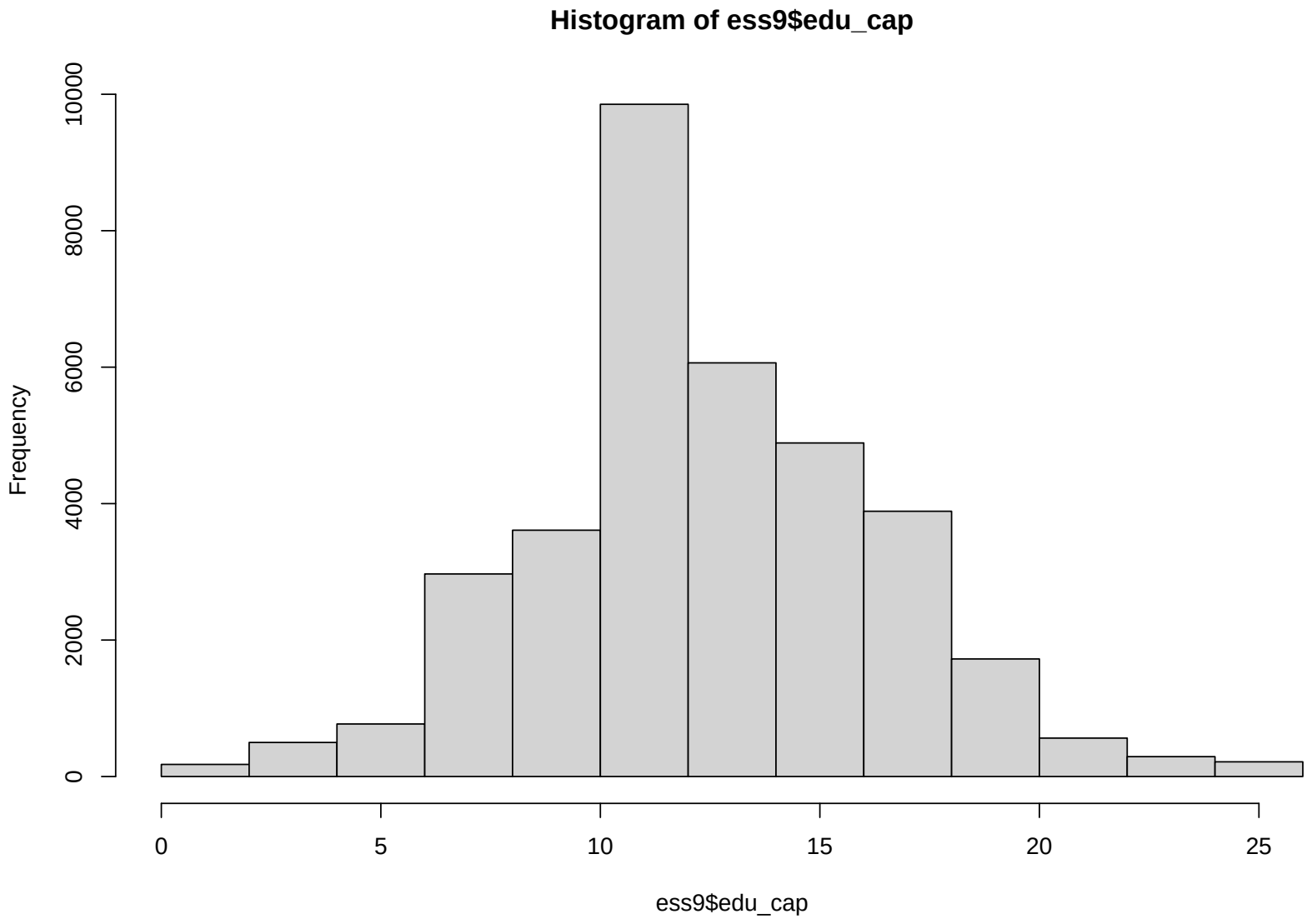
```
# if eduyrs is more than 25 change it to 25
ess9$edu_cap <- ifelse(ess9$eduyrs > 25,
                       25,
                       ess9$eduyrs)

summary(ess9$edu_cap)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.00   11.00   12.00   12.99   16.00   25.00     505
```

Helper function - ifelse

```
hist(ess9$edu_cap)
```



Helper function - ifelse

```
table(ess9$vote)
```

```
##
##     1     2     3
## 25447  7208  2948
```

```
attributes(ess9$vote)
```

```
## $label
## [1] "Voted last national election"
##
## $format.stata
## [1] "%20.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##                  Yes                   No Not eligible to vote
##                    1                    2                    3
##              Refusal           Don't know           No answer
##                   NA                   NA                   NA
```

Helper function - ifelse

```r
# code to 0 all values except 1
ess9$vote2 <- ifelse(ess9$vote == 1, 1, 0)

# code as missing if value is 3
ess9$vote2 <- ifelse(ess9$vote == 3, NA, ess9$vote2)

# check result (print also missing cases)
table(ess9$vote, ess9$vote2, useNA = "always")
```

```
##
##              0     1  <NA>
##    1         0 25447     0
##    2      7208     0     0
##    3         0     0  2948
##    <NA>      0     0   412
```

Helper function - %in%

```
# check if elements on the left side are in the right
"a" %in% c("a", "c", "d", "a")
```

```
## [1] TRUE
```

Helper function - %in%

```
# let's recode marital status
attributes(ess9$marsts)
```

```
## $label
## [1] "Legal marital status"
##
## $format.stata
## [1] "%66.0g"
##
## $class
## [1] "haven_labelled" "vctrs_vctr"     "double"
##
## $labels
##                                                    Legally married
##                                                                  1
##                                  In a legally registered civil union
##                                                                  2
##                                                 Legally separated
##                                                                  3
##                               Legally divorced/Civil union dissolved
##                                                                  4
##                                             Widowed/Civil partner died
##                                                                  5
## None of these (NEVER married or in legally registered civil union)
##                                                                  6
##                                                       Not applicable
##                                                                 NA
```

Helper function - %in%

```
ess9$single <- ifelse(ess9$marsts %in% c(3, 4, 6),
                      1, 0)

# check result
table(ess9$marsts, ess9$single)
```

```
##
##          0     1
##    1   867     0
##    2   101     0
##    3     0   460
##    4     0  3122
##    5  3380     0
##    6     0 10876
```

Other ways to do conditional change

```
# everyone gets value 0
ess9$partner <- 0

# we change to 1 if they have 1 or 2 on marsts
ess9$partner[ess9$marsts < 3] <- 1

# check result
table(ess9$marsts, ess9$partner, useNA = "always")
```

```
##
##             0     1  <NA>
##    1        0   867     0
##    2        0   101     0
##    3      460     0     0
##    4     3122     0     0
##    5     3380     0     0
##    6    10876     0     0
##    <NA> 17209     0     0
```

Code missing cases as well

```r
ess9$partner[is.na(ess9$marsts)] <- NA

# check result
table(ess9$marsts, ess9$partner, useNA = "always")
```

```
##
##                0     1  <NA>
##    1           0   867     0
##    2           0   101     0
##    3         460     0     0
##    4        3122     0     0
##    5        3380     0     0
##    6       10876     0     0
##    <NA>        0     0 17209
```

Practical 2

Summary slides

Main topics from today

Data analysis and management framework

Introduction to R and Rstudio

Objects in R

Atomic, matrix, lists and data frames

Atomic objects can be:
- logical
- numeric (integer/double)
- character

Names in R

Names can be added to different objects

Factors can be used for categorical data

Using matrices and data frames

Matrices: 2d but homogeneous

Data frames: 2d but heterogeneous

Subsetting objects in R

Using "[" with:
- positive integers
- negative integers
- logical vectors
- character vectors

Using "$" and "[[" for lists

Working with data

Importing/exporting data in R

Describing data

Modifying data