Intro to Programming and Data Analysis in R

Dr. Iulia Cioroianu

Institute for Policy Research, University of Bath

TOPIC 1. Introduction, installing and setting up R and RStudio

https://cran.r-project.org/

https://www.rstudio.com/products/rstudio/download/

Downloading and installing R:

Depending on your operating system, follow the instruction below.

a. Windows

Visit https://cran.r-project.org/bin/windows/base/ and click on "Download R 3.5.1 for Windows" in the upper left corner. Save the R-3.5.1.exe installer in your Downloads directory (or any other location of your choice on your computer). Once the download has finished, double click the file. If asked if you want to run the software and/or if you are allowing the app to make changes to your computer click "Run" and/or "Yes". Select the language to be used in the installation (English is the default). Read the information provided and keep clicking "Next" on the following screens to keep all the default settings. Wait for the installation to finish. Proceed to the "Install RStudio" section below.

b. Mac OS X

Visit https://cran.r-project.org/bin/macosx/ and click on "R-3.5.1.pkg" on the left side of the page. Wait for the installer to download, then click on it and click the "Continue" button until prompted to agree to the terms and conditions. Read the terms and conditions and click the "Agree" button. Then click the "Install" button. If prompted to log in to complete the installation, insert your computer username and password, and click "Install Software". When installation finishes, proceed to the "Install RStudio" section below.

c. Linux

Use the following command to install R-Base:

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

typing "y" if prompted along the way. Wait for the installer to finish and proceed to the Install RStudio section below.

Starting an R session

Windows: Rgui.exe.
MAC: R.app.
UNIX: you enter the command R (the path must include the R binaries).

Writing scripts

You can write anything in the console, but it is much more convenient to write a script that you can then run (in chunks or in full).
> Script/program - collection of expressions to be evaluated sequentially.

You can write scripts in the R GUI, but it is much more convenient to use a text editor such as Emacs, VIM, Eclipse, OR…

The amazing IDE (integrated development environment) for R: **RStudio**.

Downloading and installing RStudio:

Visit https://www.rstudio.com/products/rstudio/download/#download and click the installer that corresponds to your operating system to download RStudio Desktop. Wait for the installer to download, then click on it to proceed with the installation, clicking "Next"/"Keep" and "Install" along the way if prompted. Wait for the installer to finish.

`Testing`

You should now have both R and RStudio installed among your programs/applications. To test if the installation was successful, start RStudio by either clicking on the shortcut created or searching for it in your programs/applications list/search box.

In the "Console" window, type:

```
1+sqrt(4)
```

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

And click enter. R will compute and display the answer for you. In this case, 3.

Setting the working directory

What is the current working directory?

```
getwd()
```

```
## [1] "C:/Rintro"
```

Change the working directory to the folder we created at the beginning:

```
setwd("C:/Rintro")
```

What is in the working directory?

```
list.files()
```

```
## [1] "Data_analysis_R.R" "example_data.csv"  "Intro_to_r.pdf"
## [4] "Intro_to_r.Rmd"    "Rplot.png"         "saved_data.csv"
## [7] "table_data.csv"
```

List the folders and sub-folders:

```
list.dirs()
```

```
## [1] "."
```

Getting help

In RStudio: the Help panel in the lower-right window. Try it out! In general, in R, find out more about an R command or function `x`, `help(x)` or `?x`.

```
help(median)
```

```
## starting httpd help server ... done
```

```
?lm
```

Search the titles, names, aliases, and keyword entries of the available help files for a keyword or phrase:

```
help.search("regular expression")
```

```
help.start()  #HTML help interface
```

```
## If nothing happens, you should open
## 'http://127.0.0.1:14973/doc/html/index.html' yourself
```

TOPIC 2: Basic computing and programming in R

This section is based on"Introduction to Scientific Programming and Simmulation Using R"by Jones, Millardet and Robinson, which is a great resource if you are serious about using R in your work. Available at: https://www.crcpress.com/Introduction-to-Scientific-Programming-and-Simulation-Using-R-Second-Edition/Jones-Maillardet-Robinson/9781466569997

Arithmetic operations

```r
((1 + 2/3)*3)^2
```

```
## [1] 25
```

```r
sqrt(25)   # square root
```

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

```r
exp(1)   #exponential
```

```
## [1] 2.718282
```

```r
options(digits = 20)   #set number of digits to display
```

```r
exp(1)
```

```
## [1] 2.7182818284590451
```

```r
options(digits = 6)
```

```r
log(2.7183)   #logarithm
```

```
## [1] 1.00001
```

Functions

Multiple built-in functions were used above. A function takes one or more arguments (or inputs) and produces one or more outputs (or return values). You write the name of the function followed by the argument in round brackets.

```r
pi   #itself a function in R, without an argument
```

```
## [1] 3.14159
```

```r
sin(pi/6)   #sine function
```

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

```r
round(pi)   #round to nearest integer
```

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

```r
seq(from = 1, to = 7, by = 2)   #produces sequences of numbers from 1 to 9, every 2
```

```
## [1] 1 3 5 7
```

```r
seq(from = 1, to = 7)   #the default value for the third argument is 1
```

```
## [1] 1 2 3 4 5 6 7
```

If you do not specify the names of the arguments (from, to, by) then the order you list them in in the function is assumed to be the default one.

```
seq(1, 9, 3)
```

```
## [1] 1 4 7
```

```
?seq
```

Variables

`"Expression"` - a phrase of code that can be executed. `"Assignment"` - saving the evaluation of an expression, assigning it to a variable.

```
a=1   #variables are created the first time you assign a value to them.
```

```
a   #display the value of variable a
```

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

```
A <- 2   #names are case sensitive.
```

```
print(A)   #display the value of variable A
```

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

```
a+A/10   #evaluate value of this expression
```

```
## [1] 1.2
```

```
a <- a+1   #you can have a variable both on the left and the right
```

```
a
```

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

Using <- or = ?

Can use either `=` or `<-` for assignment. They do the same thing. Some prefer one, some prefer the other. In this tutorial I will use `<-` because it is more widely used. I prefer `=` because it is easier to use and similar to other programming languages that I use. To get `<-` in RStudio, use Alt- as a shortcut. Note that it includes spaces around it. It is good practice to have these spaces, although they are not (always) necessary. It is also good practice to give informative names to your variables.

Vectors

A vector (or array) is an indexed list of variables. Created the first time a value is assigned to them, just like variables.
A variable is a vector with length 1.
Vectors are created by using functions such as `c()`-combine, `seq(from, to, by)`-sequence, `rep(x, times)`-repetition. Let's create some vectors. Check the "Environment" panel in the left to see them.

```
v1 <- c(1, 2, 3, 4)
```

```
v2 <- seq(1,7,2)
```

```
v3 <- rep(2,3)
```

```
v4 <- c(v2,v3)   #we can combine vectors as well
```

To refer to element i of vector x, we use x[i].

```r
v1[1]
```

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

```r
v2[2:3]    #Get elementf "from:to"
```

```
## [1] 3 5
```

```r
length(v3)  #the number of elements of vector v3
```

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

Algebraic operations on each element separately (elementwise):

```r
v1+v2
```

```
## [1]  2  5  8 11
```

```r
v1*v2
```

```
## [1]  1  6 15 28
```

```r
v1*v3
```

```
## Warning in v1 * v3: longer object length is not a multiple of shorter
## object length
```

```
## [1] 2 4 6 8
```

Note the warning message. The two vectors need to either be the same length or one be a multiple of the other. If not, R duplicates the shorter one, but produces the warning.

```r
2+v3   #So if one of them is of length 1, this works.
```

```
## [1] 4 4 4
```

Applying functions to vectors

Note that some functions applied to vectors act on the whole vector:

```r
sum(v1)
```

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

```r
mean(v1)
```

```
## [1] 2.5
```

```r
var(v1)
```

```
## [1] 1.66667
```

```r
prod(v1)
```

```
## [1] 24
```

```r
min(v1)
```

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

```r
max(v1)
```

```
## [1] 4
```

While other functions act elementwise:

```r
sqrt(v1)
```

```
## [1] 1.00000 1.41421 1.73205 2.00000
```

```r
sort(v1, decreasing=TRUE)
```

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

Logical expressions: True or False

Comparison operators <, >, <=, >=, == (equal to), and != (not equal to). Logical operators & (and), | (or), and ! (not).

```r
v1==v2 #0 and 1 can be used to mean true or false
```

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

```r
c(0,1,1) | c(1,1,0)    #True if either one of them true
```

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

```r
xor(c(0,1,1), c(1,1,0))    #exclusive disjunction -
```

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

```r
                           #either one or the other but not both
```

Selecting a subset of a vector for which a certain condition is true:

```r
v1 > 2
```

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

```r
v1[v1 > 2]
```

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

Finding the position of elements which meet a certain condition:

```r
which(v4>=5)
```

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

Missing data

```r
m <- NA   # assign NA to variable a
```

```r
is.na(m)    #is it missing?
```

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

```r
m <- c(1,NA,3)    #NA in a vector
```

```r
is.na(m)    #is it missing?
```

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

```r
mean(m)    # NAs can propagate
```

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

```r
mean(m, na.rm = TRUE)    # NAs can be removed
```

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

```
length(m[!is.na(m)]) #how many non-missing elements are in m?
```

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

Matrices

Created from a vector using the function `matrix(data, nrow = 1, ncol = 1, byrow = FALSE)`. If `length(data)<nrow*ncol`, then data is reused as many times as is needed.

```
matrix(0, 2, 2)
```

```
##      [,1] [,2]
## [1,]    0    0
## [2,]    0    0
```

```
matrix(c(1,2), 2, 2)
```

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

```
matrix(c(1:6), 2, 3)
```

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

```
matrix(1:6, 2, 3, TRUE)
```

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

To create a diagonal matrix:

```
diag(1:3)
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    2    0
## [3,]    0    0    3
```

We refer to the elements of a matrix using two indices: the first one for row, the second for column

```
A <- matrix(1:6, 2, 3)  #create a new matrix
```

```
A
```

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

```
A[1, 3] <- 0   #assign the valie 0 to the element on row 1, column 3
```

```
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    0
## [2,]    2    4    6
```

```
A[, 2:3]   #display all rows, but only columns from 2 to 3.
```

```
##      [,1] [,2]
## [1,]    3    0
## [2,]    4    6
```

```r
A[1,]
```

```
## [1] 1 3 0
```

Get the number of rows and columns of the matrix:

```r
nrow(A)
```

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

```r
ncol(A)
```

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

To join matrices with rows of the same length (stacking vertically):

```r
rbind(A, c(1:3))
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    0
## [2,]    2    4    6
## [3,]    1    2    3
```

To join matrices with columns of the same length (stacking horizontally) use cbind(. . .).

```r
cbind(A, c(1:2))
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    0    1
## [2,]    2    4    6    2
```

Algebraic operations act elementwise on matrices:

```r
A+A   # addition
```

```
##      [,1] [,2] [,3]
## [1,]    2    6    0
## [2,]    4    8   12
```

```r
A*A   #element-wise product
```

```
##      [,1] [,2] [,3]
## [1,]    1    9    0
## [2,]    4   16   36
```

For matrix multiplication, use %*%:

```r
A%*%t(A)   # multiply A by its transpose
```

```
##      [,1] [,2]
## [1,]   10   14
## [2,]   14   56
```

```r
solve(A%*%t(A))   #get the inverse of the above square matrix
```

```
##            [,1]       [,2]
## [1,]  0.1538462 -0.0384615
## [2,] -0.0384615  0.0274725
```

From vector to matrix and the other way around

```r
is.vector(A)   #test if A is a vector
```

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

```r
is.matrix(A)   #test if A is a matrix
```

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

```r
B <- as.matrix(v1)  # create a matrix of 1 column out of a vector
```

```r
B
```

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

```r
is.matrix(B)
```

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

```r
x <- as.vector(A)  #create a vector from the columns of a matrix (stored columnwise)
```

```r
x
```

```
## [1] 1 2 3 4 0 6
```

```r
?is.vector
```

Control structures

- *if*

```r
if (A[1,1]==1) {
  print("First element of first column is 1")}
```

```
## [1] "First element of first column is 1"
```

- if, else statement

```r
if (A[1,1]==2) {
  print("First element of first column is 2")
} else {
  print("First element of first column is not 2")
}
```

```
## [1] "First element of first column is not 2"
```

- *ifelse*

```r
v <- c(1,2,3,4,5,6)
ifelse(v %% 2 == 0,"even","odd")
```

```
## [1] "odd"  "even" "odd"  "even" "odd"  "even"
```

- *for* loop

```r
for(i in v2) {
  print(i)
}
```

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

```
## [1] 3
## [1] 5
## [1] 7
```

Note that the value of i gets updated in the loop.

```
print(i)
```

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

- *while*

```
j <- 1
while (j < 5) {
  print(j)
  j = j+1
}
```

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

Note that it is not efficient or advisable to use loops in R.

Writing your own functions

```
x=1
```

```
y=100
```

```
result=0
```

```
myfunction <- function(x, y) {
    #function that raises x to the power y
  result <- x^y
  print(paste(x," raised to the power ",y," is ",result))
}
```

Note that you need to be careful when you write functions not to re-write any that already exist. For example, if we names our function above `sum` then that woule replace the sum function in base R. If that ever happens, you can revert back to the original function with `rm(sum)` or `sum <- base::sum`.

Calling the function:

```
myfunction(2,5)
```

```
## [1] "2  raised to the power  5  is  32"
```

TOPIC 3: Working with dataframes

We have the following vectors:

```
a <- c(78, 89.5, 82, 36.5, 74, 39)   #numeric vector
b <- c("one","two","three","six","five","four")   #character vector
c <- c(TRUE,TRUE,TRUE,FALSE,TRUE,FALSE)  #logical vector
d <- c("red","blue","yellow", NA, NA, NA)   #character vector
```

Create a dataset out of the vectors above

```r
mydata <- data.frame(a,b,c,d)
```

Give names to the variables (the vectors)

```r
names(mydata) <- c("score","exam_seat", "passed","color")   #variable names
```

Identify and inspect the elements of a data frame:

```r
mydata   #prints the data
```

```
##   score exam_seat passed  color
## 1  78.0       one   TRUE    red
## 2  89.5       two   TRUE   blue
## 3  82.0     three   TRUE yellow
## 4  36.5       six  FALSE   <NA>
## 5  74.0      five   TRUE   <NA>
## 6  39.0      four  FALSE   <NA>
```

```r
View(mydata)   #opens the dataframe in a new tab. Note the capital V.
```

```r
names(mydata)   #print the names of the variables
```

```
## [1] "score"     "exam_seat" "passed"    "color"
```

```r
nrow(mydata)   #the number of observations in the dataset (number of rows)
```

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

```r
ncol(mydata)   #the number of variables in the dataset (number of columns)
```

```
## [1] 4
```

```r
mydata[2:4]    # prints columns 2,3,4 of data frame
```

```
##   exam_seat passed  color
## 1       one   TRUE    red
## 2       two   TRUE   blue
## 3     three   TRUE yellow
## 4       six  FALSE   <NA>
## 5      five   TRUE   <NA>
## 6      four  FALSE   <NA>
```

```r
mydata[c("score","color")]   #prints columns id and color from data frame
```

```
##   score  color
## 1  78.0    red
## 2  89.5   blue
## 3  82.0 yellow
## 4  36.5   <NA>
## 5  74.0   <NA>
## 6  39.0   <NA>
```

```r
mydata$passed   #prints variable 'passed' from the data frame
```

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

```r
mydata["passed"]
```

```
##   passed
## 1   TRUE
## 2   TRUE
```

```
## 3   TRUE
## 4  FALSE
## 5   TRUE
## 6  FALSE
```

Subsetting - selecting only certain observations and/or certain variables.

Select only observations with score less than 80, put them in a new object called 'newdata'

```
newdata1 <- subset(mydata, score<=80)
newdata1
```

```
##   score exam_seat passed color
## 1  78.0       one   TRUE   red
## 4  36.5       six  FALSE  <NA>
## 5  74.0      five   TRUE  <NA>
## 6  39.0      four  FALSE  <NA>
```

Select only observations with acore less than 80 and which passed (so passed==TRUE).

```
newdata2 <- subset(mydata, score<80 & passed==TRUE)
newdata2
```

```
##   score exam_seat passed color
## 1    78       one   TRUE   red
## 5    74      five   TRUE  <NA>
```

Select only observations which passed AND keep only the variables 'score' and 'passed' in the new dataset.

```
newdata3 <- subset(mydata, passed==TRUE, select=c(score,passed))
```

```
newdata3
```

```
##   score passed
## 1  78.0   TRUE
## 2  89.5   TRUE
## 3  82.0   TRUE
## 5  74.0   TRUE
```

Aggregate/collapse data

```
#Get the mean score for the group that failed and the one that passed
aggregated_data=aggregate(cbind(score)~passed, data=mydata, mean)
```

More generally, you can do: `aggregate(cbind(var1, var2, var3)~cat_var1+cat_var2, data=mydata, mean)` to get the means of `var1`, `var2` and `var3` BY the categories of categorical variables `cat_var1` and `cat_var2`. Instead of mean you can also get sum, median, min, max.

You can assign the result of the aggregation to a new dataframe

```
collapsed_data <- aggregate(cbind(score)~passed, data=mydata, mean)
```

Generate and modify variables

To create a new variable in a dataset you just have to declare it.

```
mydata$newvar <- NA   #generate a new empty variable (full of NAs).
```

```
ncol(mydata)
```

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

Warning!!! If you declare any variable in the dataset again, it gets overwritten without any warning. So be careful with this!

```
mydata$newvar <- 0   #the already existing variable 'newvar' is changed to all 0s.
```

It's always good practice to copy a variable that you want to make changes into a new variable, and modify the new one.

```
mydata$newvar <- mydata$score   #replace the values of 'newvar' with the values of 'score'
```

Assign values to a variable, conditional on something:

```
mydata$newvar[mydata$score!=78] <- 0   #if score different from 78, newvar is 0.
```

```
mydata$newvar[mydata$score==78] <- 1  # if score is equal to 78, newvar is 1.
  #(You can also use <, <=, >, >=)
```

```
mydata
```

```
##   score exam_seat passed  color newvar
## 1  78.0       one   TRUE    red      1
## 2  89.5       two   TRUE   blue      0
## 3  82.0     three   TRUE yellow      0
## 4  36.5       six  FALSE   <NA>      0
## 5  74.0      five   TRUE   <NA>      0
## 6  39.0      four  FALSE   <NA>      0
```

```
mydata$newvar[mydata$score<=80 & mydata$passed==TRUE] <- 1
#if score<=80 AND passed is true then newvar becomes 1
```

```
mydata$newvar[is.na(mydata$color)==TRUE | mydata$score==1 | mydata$exam_seat=="three"] <- 2
```

Code above says: if color is missing OR score is 1 OR exam_seat is 'three' then replace the value of newvar with 2.

```
mydata
```

```
##   score exam_seat passed  color newvar
## 1  78.0       one   TRUE    red      1
## 2  89.5       two   TRUE   blue      0
## 3  82.0     three   TRUE yellow      2
## 4  36.5       six  FALSE   <NA>      2
## 5  74.0      five   TRUE   <NA>      2
## 6  39.0      four  FALSE   <NA>      2
```

NOTE: You only need quotation marks in the condition for string variables (variables whose values are words). Do NOT use quotation marks for number variables (whose values are either only numbers or a combination of numbers and missing data(NA)), and do NOT use quotation marks for boolean - TRUE/FALSE - variables.

```
A=TRUE
```

```
typeof(A)
```

```
## [1] "logical"
```

Create a dummy variable (a variable with only two categories)

```
mydata$newvar2 <- ifelse(is.na(mydata$color)==TRUE, 1, 0)
mydata
```

```
##   score exam_seat passed  color newvar newvar2
## 1  78.0       one   TRUE    red      1       0
## 2  89.5       two   TRUE   blue      0       0
## 3  82.0     three   TRUE yellow      2       0
## 4  36.5       six  FALSE   <NA>      2       1
## 5  74.0      five   TRUE   <NA>      2       1
## 6  39.0      four  FALSE   <NA>      2       1
```

Create a variable `newvar2` which takes the value 1 if color is not missing, and 0 if color is missing.

What type of variable it newvar2?

```
typeof(mydata$newvar2)
```

```
## [1] "double"
```

Change it to integer:

```
mydata$newvar2 <- as.integer(mydata$newvar2)
```

TOPIC 4: Importing and exporting data

Check out the "Import dataset" button in the Environment window. What formats are available for import? Make sure you give the imported data frame a short name. Check 'Yes' for heading if the variable names are on the first row.

OR: Set the working directory to be the folder where you have the data:

```
setwd("C:/Rintro")
```

Now you can read the data directly from the folder. Everything that you save will also go to that folder.

```
data <- read.csv("example_data.csv")
```

Let's have a look at the data.

```
View(data)
```

Turning off scientific notation

```
options(scipen=999)
```

Export the data to a new file, called `saved_data.csv`.

```
write.csv(data, file="C://Rintro/saved_data.csv")
```

Exercise:

Replace the "missing" values of the variables in our dataset with NA.

```
 typeof(data$reelected)
```

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

```
data[data=="missing"] <- NA
```

Turn variable to numeric

```
data$reelected=as.numeric(data$reelected)
```

TOPIC 5: Descriptive statistics

Get the mean, median, 25th and 75th quartiles, min, max, number of NA (if no NA given then no NAs exist for that variable).

```
summary(data$spent) #  for a single variable
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       0  172196  256307  351340  393587 3489592
```

```
summary(data)  # for ALL the variables in the dataset.
```

```
##        id              region       party      chamber     spent
##  Min.   :  1   East      :108   Centre:179   H:434   Min.   :      0
##  1st Qu.:136   North     :108   Left  :180   S:105   1st Qu.: 172196
##  Median :270   North-East:107   Right :180           Median : 256307
##  Mean   :270   South     :108                        Mean   : 351340
##  3rd Qu.:404   West      :108                        3rd Qu.: 393587
##  Max.   :539                                         Max.   :3489592
##
##      raised          reelected      run_again         female
##  Min.   :      0   Min.   :1.00   Min.   :0.000   Min.   :0.000
##  1st Qu.: 276093   1st Qu.:2.00   1st Qu.:0.000   1st Qu.:0.000
##  Median : 455861   Median :3.00   Median :1.000   Median :0.000
##  Mean   : 717580   Mean   :2.96   Mean   :0.614   Mean   :0.315
##  3rd Qu.: 755435   3rd Qu.:4.00   3rd Qu.:1.000   3rd Qu.:1.000
##  Max.   :9790929   Max.   :6.00   Max.   :1.000   Max.   :1.000
##                    NA's   :2
##    support      incumbent        tweets        income
##  High  :310   Min.   :0.00   Min.   : 20   Min.   :  58372
##  Low   : 68   1st Qu.:0.00   1st Qu.:553   1st Qu.:  97574
##  Medium:161   Median :1.00   Median :646   Median : 126581
##               Mean   :0.57   Mean   :600   Mean   : 165333
##               3rd Qu.:1.00   3rd Qu.:706   3rd Qu.: 165242
##               Max.   :1.00   Max.   :980   Max.   :1107015
##
##      committee
##  Economy  : 63
##  Education: 61
##  Health   : 56
##  Other    :359
##
##
##
```

One measure at a time

```
length(na.omit(data$reelected)) #  The number of non-missing observations
```

```
## [1] 537
```

```
mean(data$reelected, na.rm=T)    #mean
```

```
## [1] 2.96089
```

```
sd(data$reelected, na.rm=T)      #standard deviation
```

```
## [1] 1.56583
```

```r
min(data$reelected, na.rm=T)    #minimum value
```

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

```r
max(data$reelected, na.rm=T)    #maximum value
```

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

Set the CRAN mirror. This is where we will install packages from.

```r
r <- getOption("repos")
r["CRAN"] <- "http://www.stats.bris.ac.uk/R/"
options(repos = r)
```

Install the package **psych** if it is not already installed. You only do this once.

```r
if(!require(psych)){
  install.packages('psych')
}
```

```
## Loading required package: psych
```

Load the package into your R session. For every new session.

```r
library(psych)
```

Use the `describe` function on the `psych` package to get descriptive statistics.

```r
describe(data)
```

```
##              vars   n      mean        sd median   trimmed       mad   min
## id              1 539    270.00    155.74    270    270.00    200.15     1
## region*         2 539      3.00      1.42      3      3.00      1.48     1
## party*          3 539      2.00      0.82      2      2.00      1.48     1
## chamber*        4 539      1.19      0.40      1      1.12      0.00     1
## spent           5 539 351340.45 344042.64 256307 288132.41 149300.79     0
## raised          6 539 717579.71 949969.55 455861 529747.58 322852.46     0
## reelected       7 537      2.96      1.57      3      2.85      1.48     1
## run_again       8 539      0.61      0.49      1      0.64      0.00     0
## female          9 539      0.32      0.47      0      0.27      0.00     0
## support*       10 539      1.72      0.89      1      1.66      0.00     1
## incumbent      11 539      0.57      0.50      1      0.59      0.00     0
## tweets         12 539    600.40    159.95    646    625.27    105.26    20
## income         13 539 165332.86 123305.44 126581 138906.98  44172.58 58372
## committee*     14 539      3.32      1.07      4      3.52      0.00     1
##                  max   range  skew kurtosis       se
## id               539     538  0.00    -1.21     6.71
## region*            5       4  0.00    -1.31     0.06
## party*             3       2  0.00    -1.50     0.04
## chamber*           2       1  1.54     0.36     0.02
## spent        3489592 3489592  3.61    19.70 14818.97
## raised       9790929 9790929  4.43    26.96 40918.09
## reelected          6       5  0.33    -0.93     0.07
## run_again          1       1 -0.47    -1.78     0.02
## female             1       1  0.79    -1.37     0.02
## support*           3       2  0.57    -1.51     0.04
## incumbent          1       1 -0.28    -1.93     0.02
## tweets           980     960 -1.52     2.34     6.89
## income       1107015 1048643  3.47    15.94  5311.14
```

```
## committee*       4       3 -1.23    -0.06     0.05
```

TOPIC 6: Frequency and proportions tables

Frequency table:

```
mytable <- table(data$party,data$reelected)
```

var1 will be on the rows, var2 will be on the columns.

```
mytable   # print table
```

```
##
##           1  2  3  4  5  6
##   Centre 43 33 36 37 15 14
##   Left   42 32 36 37 16 16
##   Right  45 35 30 41 15 14
```

Relative frequency (or proportions) table. Note that this is a table of the frequency table defined above.

```
prop.table(mytable)   #cell percentages
```

```
##
##                  1         2         3         4         5         6
##   Centre 0.0800745 0.0614525 0.0670391 0.0689013 0.0279330 0.0260708
##   Left   0.0782123 0.0595903 0.0670391 0.0689013 0.0297952 0.0297952
##   Right  0.0837989 0.0651769 0.0558659 0.0763501 0.0279330 0.0260708
```

```
prop.table(mytable, 1)   #row percentages
```

```
##
##                  1         2         3         4         5         6
##   Centre 0.2415730 0.1853933 0.2022472 0.2078652 0.0842697 0.0786517
##   Left   0.2346369 0.1787709 0.2011173 0.2067039 0.0893855 0.0893855
##   Right  0.2500000 0.1944444 0.1666667 0.2277778 0.0833333 0.0777778
```

```
prop.table(mytable, 2)   #column percentages
```

```
##
##                 1        2        3        4        5        6
##   Centre 0.330769 0.330000 0.352941 0.321739 0.326087 0.318182
##   Left   0.323077 0.320000 0.352941 0.321739 0.347826 0.363636
##   Right  0.346154 0.350000 0.294118 0.356522 0.326087 0.318182
```

```
table_data=data.frame(mytable)
```

Write the table out to a comma separated file:

```
write.table(table_data, file = 'table_data.csv', sep=',')
```

Topic 7: Simple plots

General syntax: `PlotType(data_to_plot, option1, option2, option3, etc)` where:

PlotType can be: plot(does a scatterplot), barplot, pie (for piechart), hist(for histogram), boxplot, etc.

data_to_plot can be a variable, a table, data coming from some model, etc.

options can be: 'main' and 'sub' for main title and sub-title; 'col' for changing colors; 'ylab' and 'xlab' for labeling the axes; legend; names.arg for labeling plot elements etc.

Let's first attach the dataset so we can refer to variable names directly.

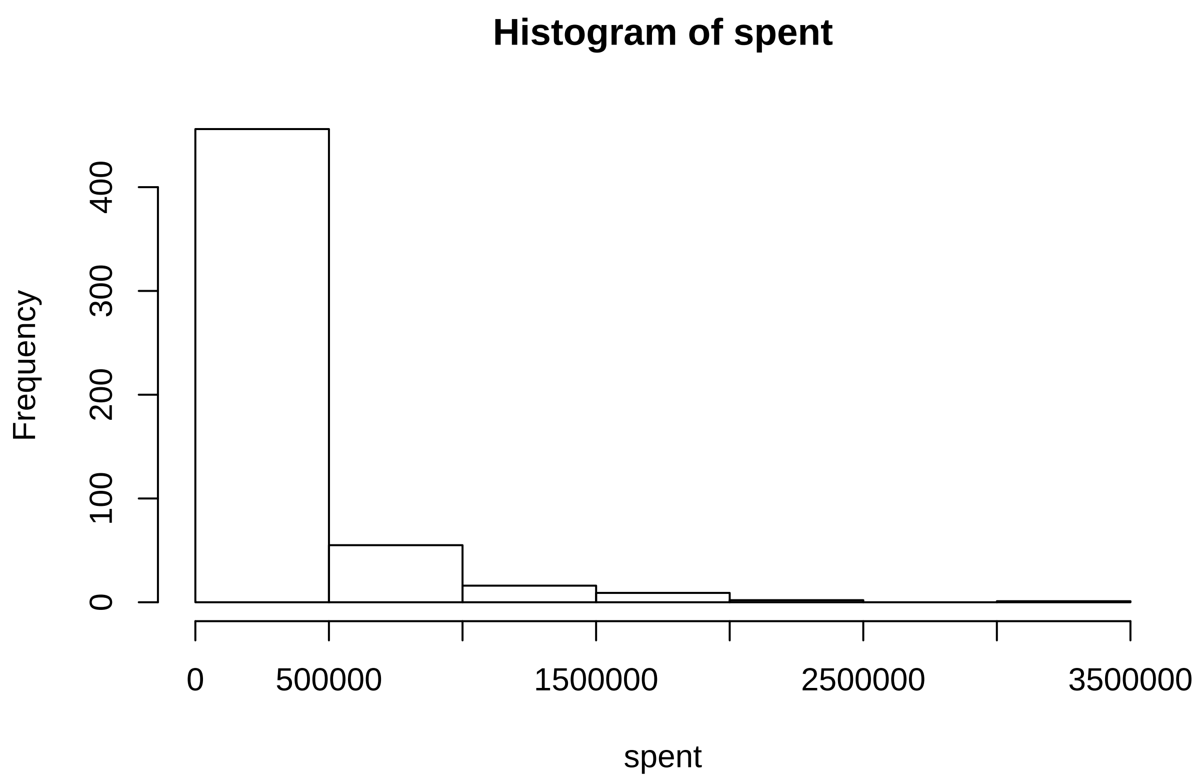
```r
attach(data)
```

```
## The following object is masked from package:psych:
##
##     income
```

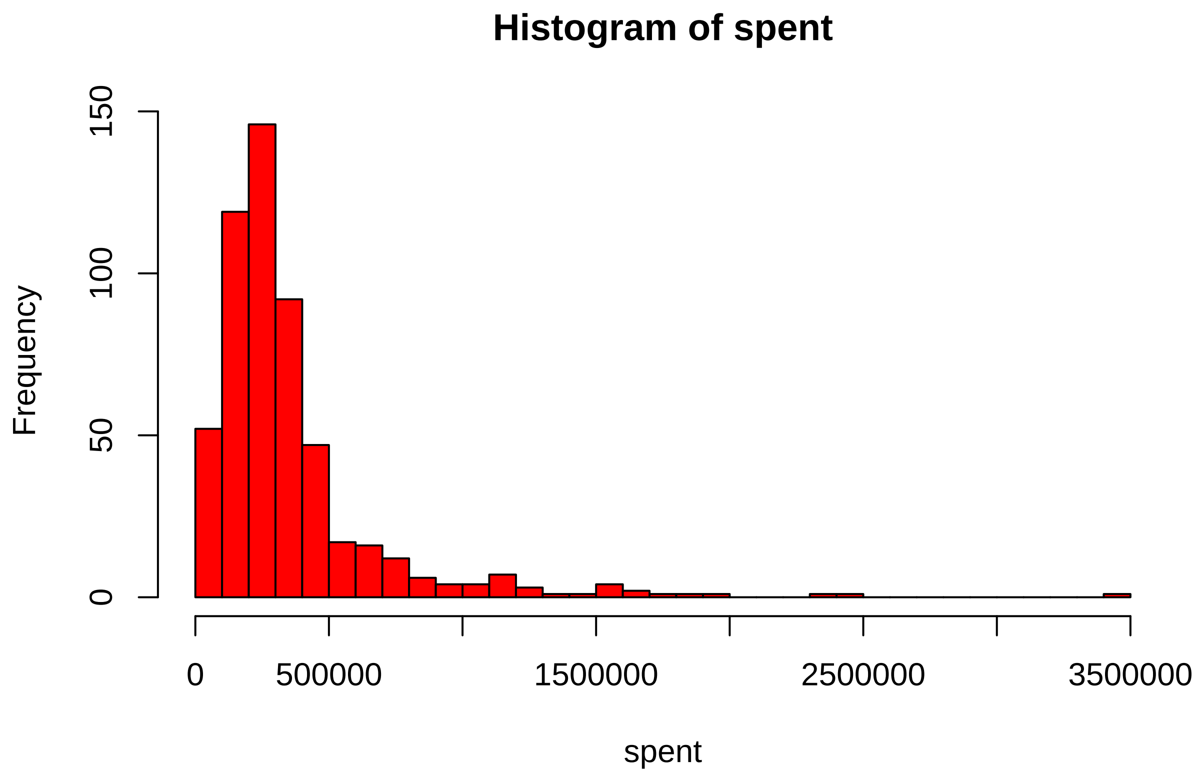
Examples:

- Histogram:

```r
hist(spent)  # simple histogram
```

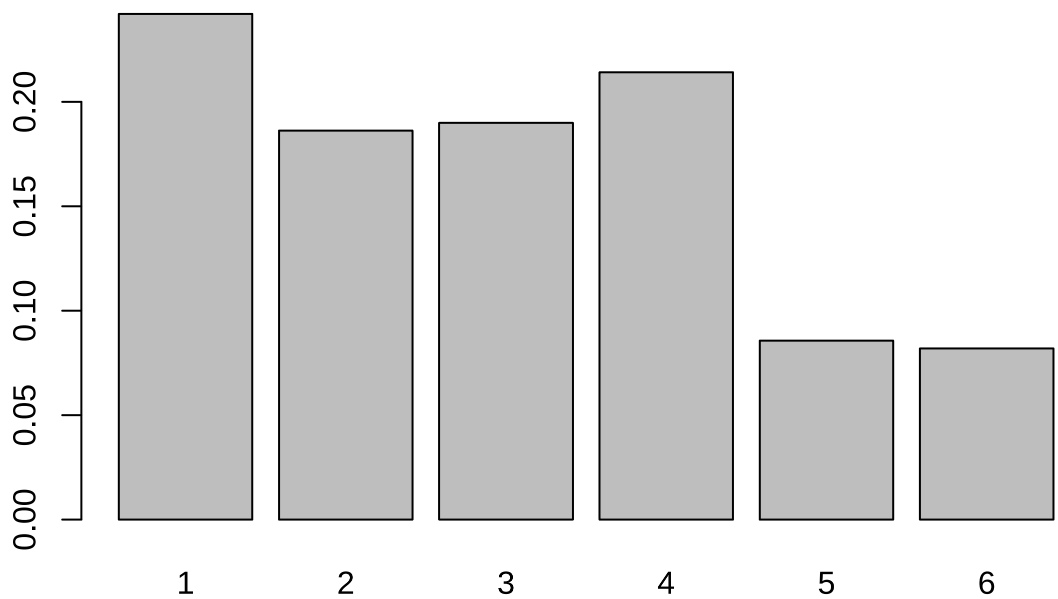



```r
hist(spent, breaks=30, col="red") #  red histogram with different number of bins
```



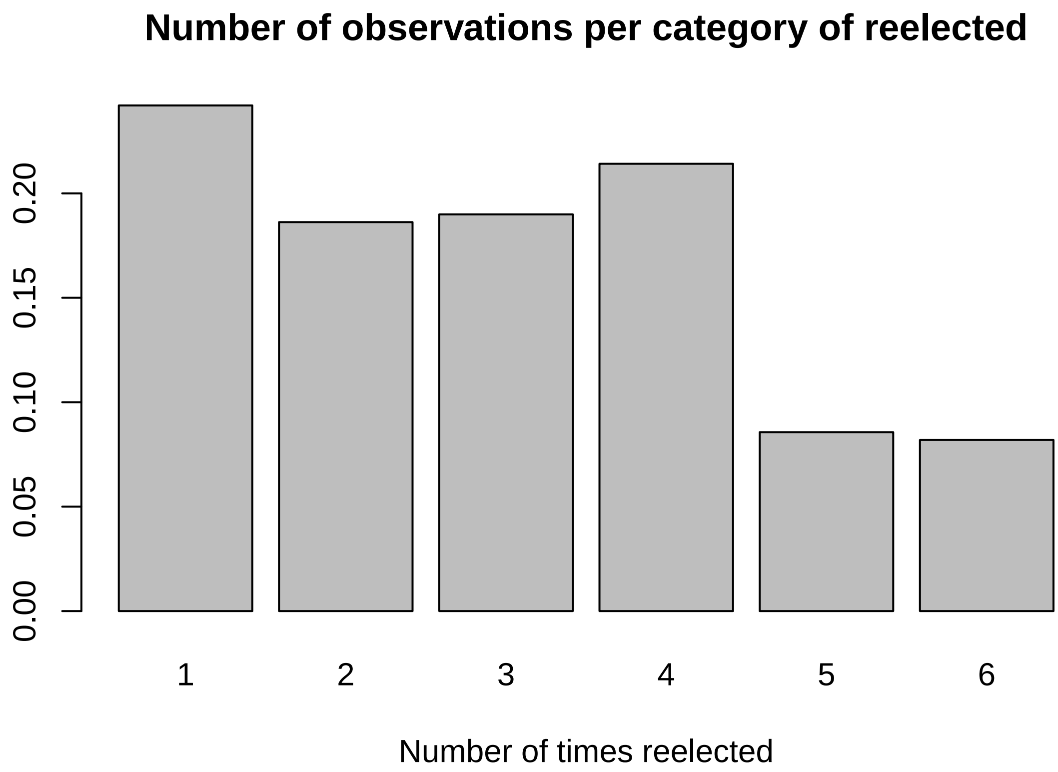
- Barplots: Simple bar plot of a a frequency table for a variable (an ordinal or binary variable):

```
barplot(prop.table(table(reelected)))
```



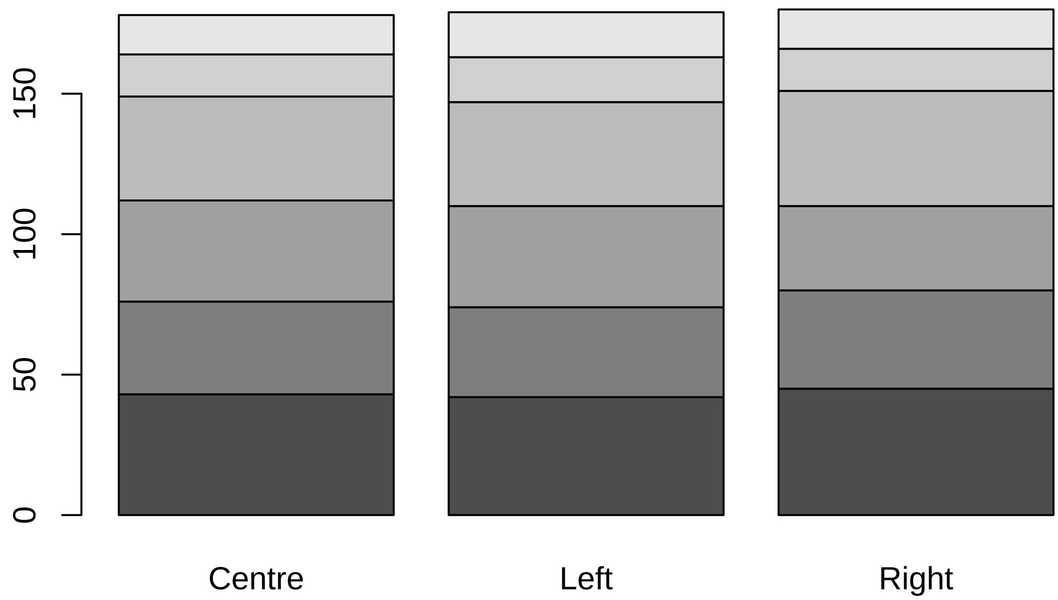
Same plot with title and labeled x axis:

```
barplot(prop.table(table(reelected)), main="Number of observations per category of reelected",
          xlab="Number of times reelected")
```



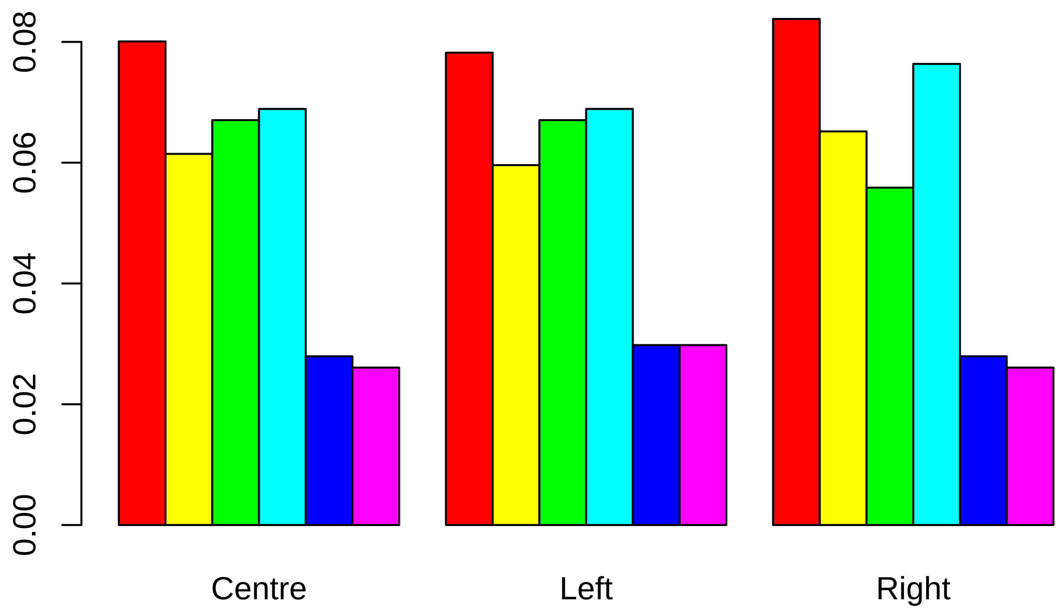
Stacked bar plot of the number of observations by group of var1 and var2

```
barplot(table(reelected,party))    # but so ugly!
```



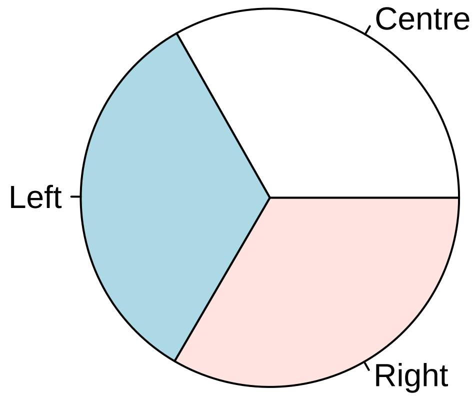
Same as above, but grouped bar plot (not stacked but aside) and letting R choose the colours:

```
barplot(prop.table(table(reelected,party)), col=rainbow(6), beside=TRUE)
```



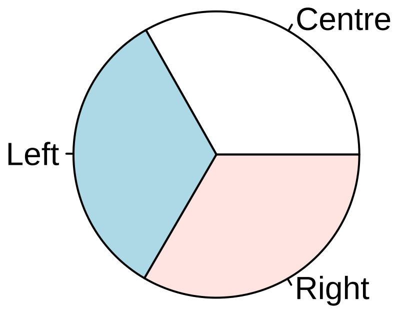
- Piecharts

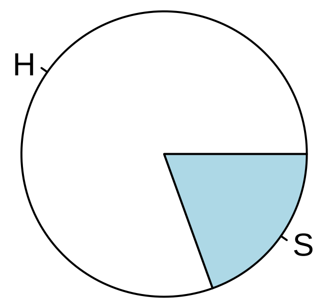
```
pie(table(party))    #Simple pie chart of var1
```



Put two graphs in the same plot, one next to the other:

```
par(mfrow=c(1,2))   #This divides the plotting space into two vertical cells.
pie(table(party))   #First plot shows on the left.
pie(table(chamber))   #Second plot shows on the right.
```

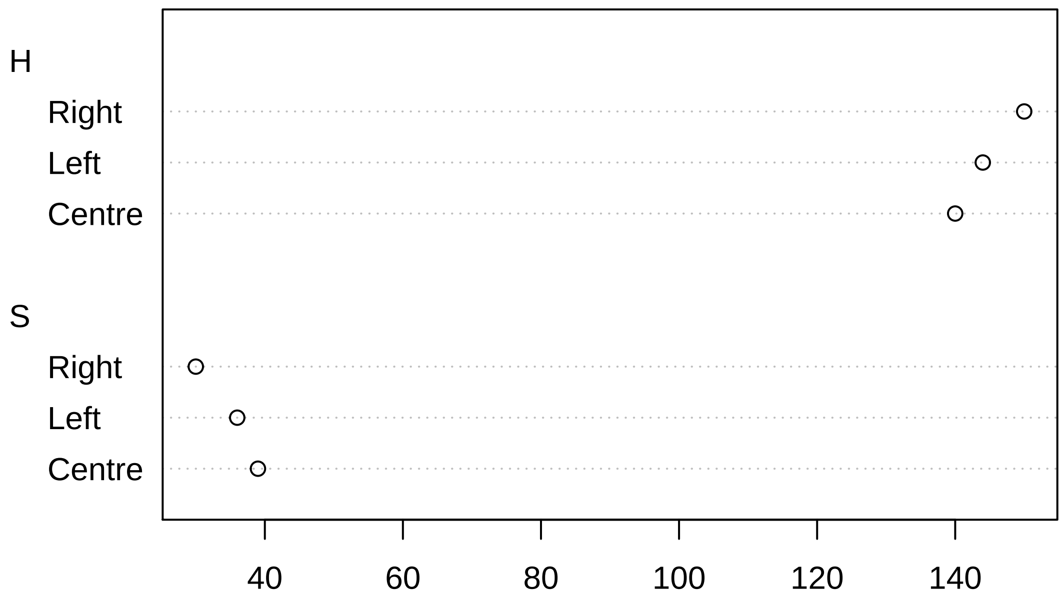





```
par(mfrow=c(1,1))    #This turns it back to a single cell
```

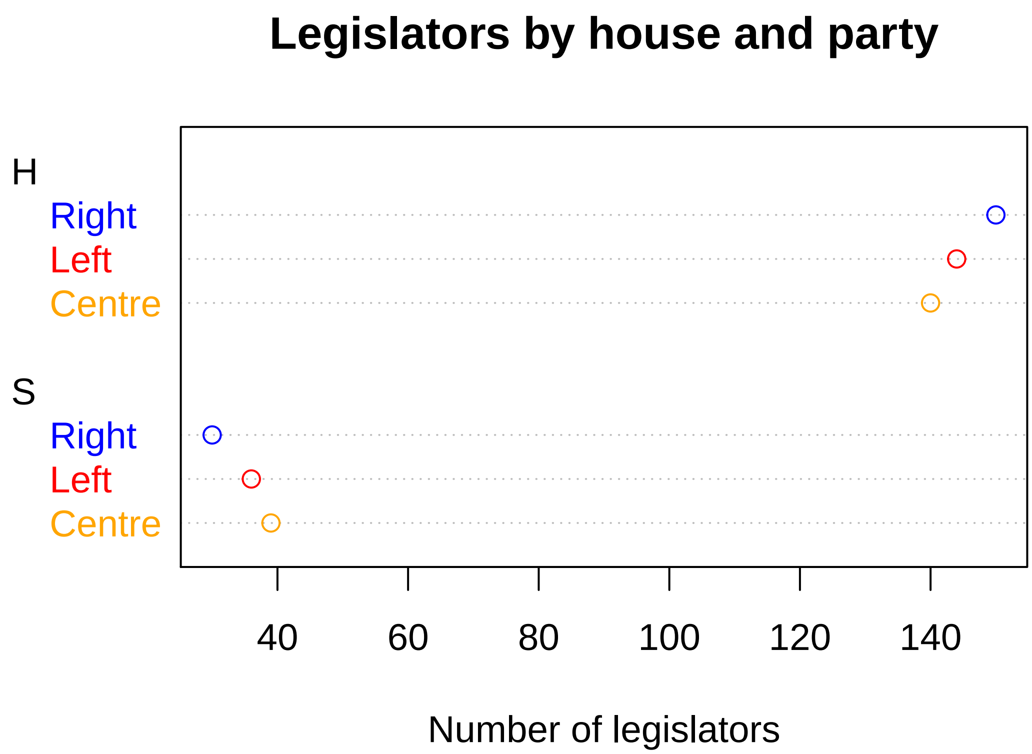
- Dotcharts

```
dotchart(table(data$party, data$chamber))    #number of legislators by party and chamber
```



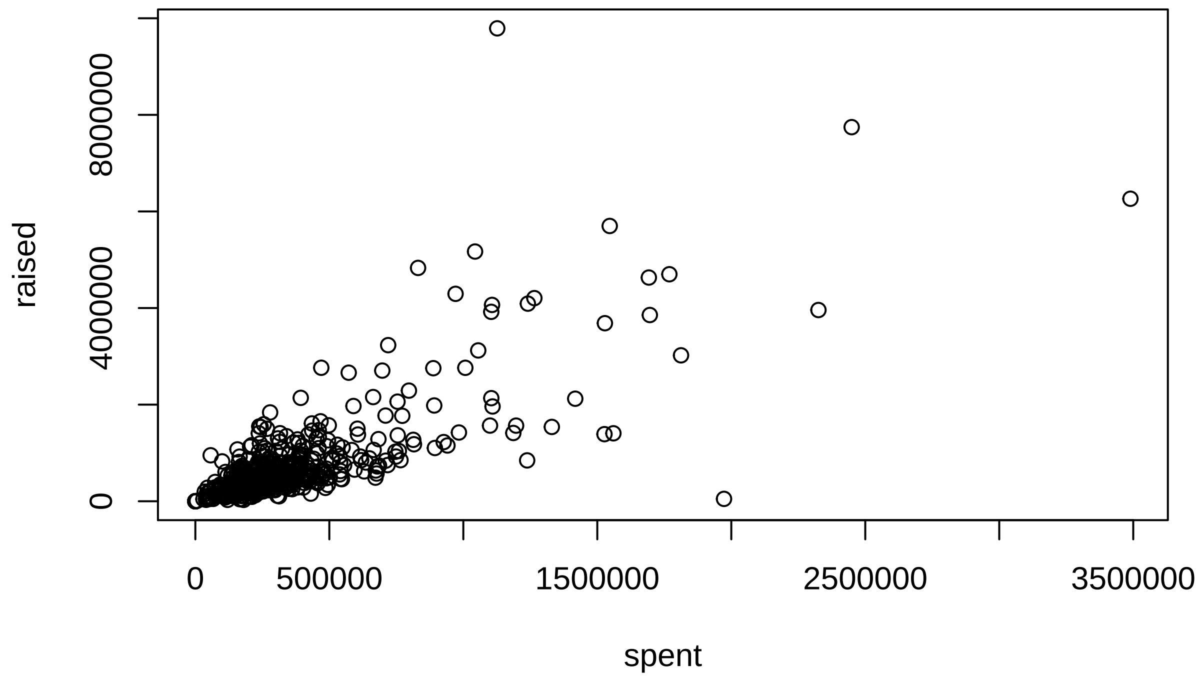
Now make it pretty:

```
dotchart(table(data$party, data$chamber), cex=1.2, color=c("orange", "red", "blue"),
         main="Legislators by house and party",
         xlab="Number of legislators")
```



- Scatterplots

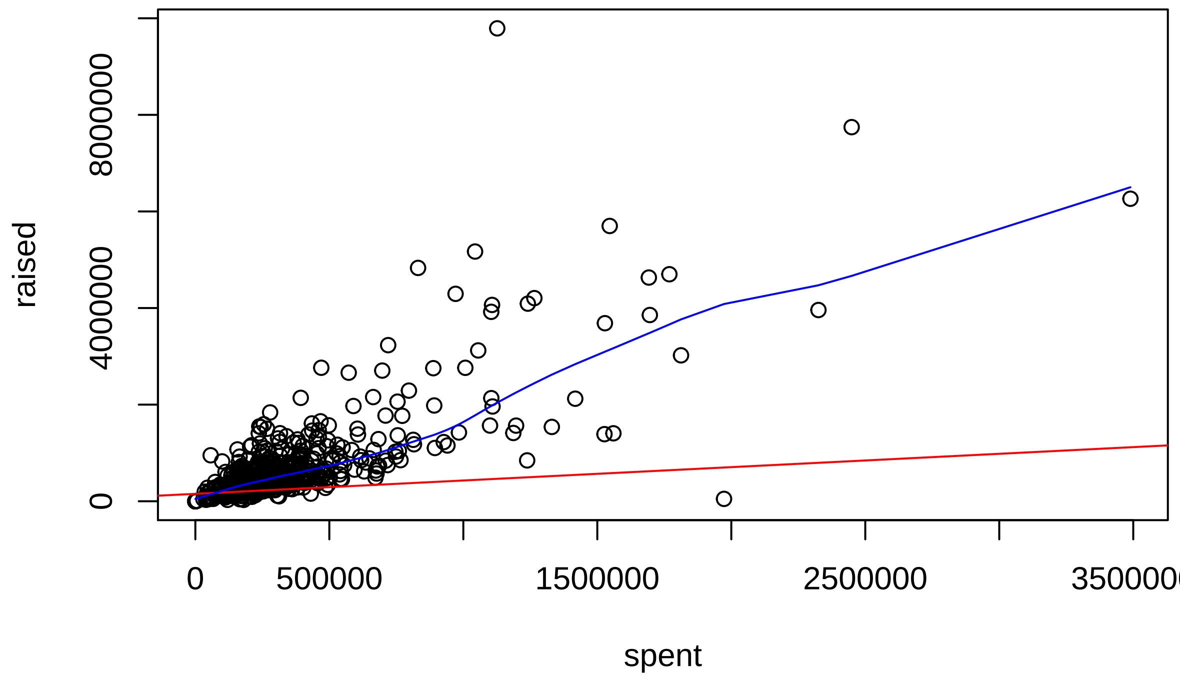
```r
plot(spent, raised)  # Simple scatterplot of variables x and y.
```



This order puts x on the horizontal axis and y on the vertical one.

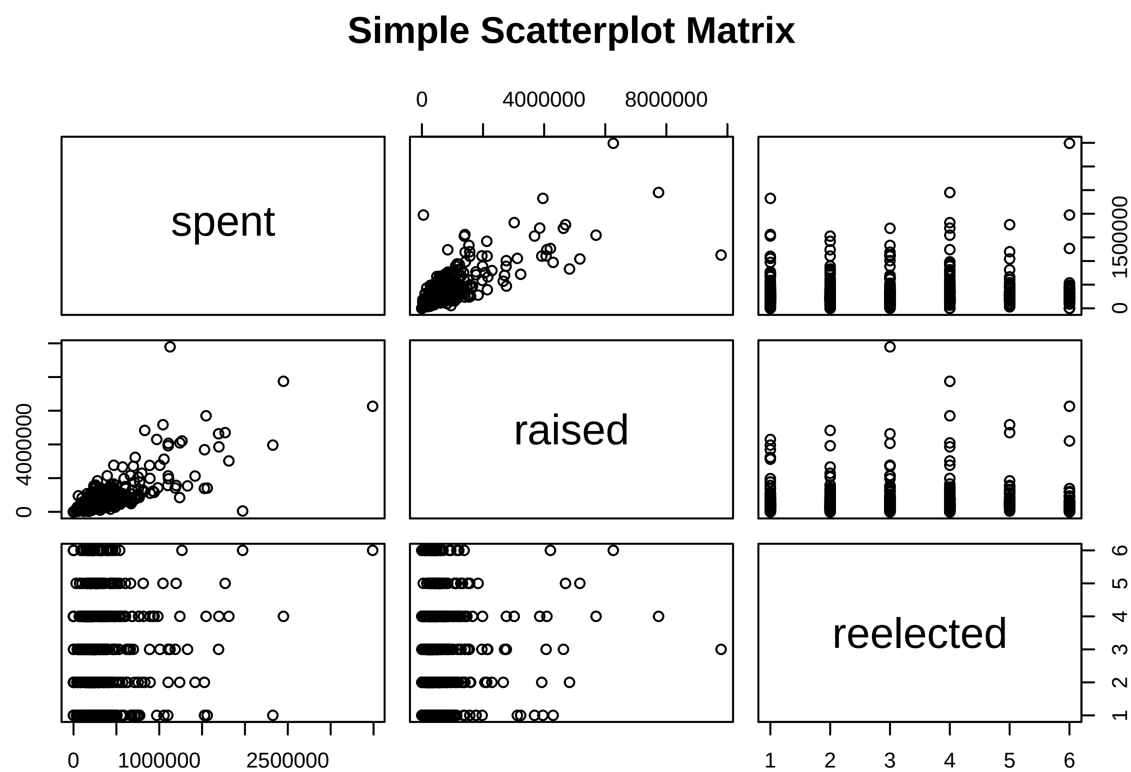
Note that the order is particularly important if you want to add a linear fit line:

```
plot(spent, raised)  # Simple scatterplot of variables x and y.
abline(lm(spent~raised), col="red")  # add a regression line (y~x)
lines(lowess(spent, raised), col="blue") # add a lowess line
```



Matrices of scatterplots, for multiple variables (two by two)

```
pairs(~spent+raised+reelected, data=data, main="Simple Scatterplot Matrix")
```



Topic 8: Correlations and covariance analysis

Correlations (What kind of correlation?)

```
?cor
```

```r
cor(data$spent, data$raised)
```

```
## [1] 0.763208
```

```r
#correlation between var1 and var2. var1 and var2 need to be numeric.
```

```r
cor(data$spent, data$raised, use="complete.obs", method="pearson")
```

```
## [1] 0.763208
```

```r
#only use complete cases if there is any missing data in var1 and var2.
```

```r
cor.test(data$spent, data$raised, use="complete.obs")  # test significance
```

```
##
##  Pearson's product-moment correlation
##
## data:  data$spent and data$raised
## t = 27.37, df = 537, p-value <0.0000000000000002
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.725517 0.796334
## sample estimates:
```

```
##      cor
## 0.763208
```

Covariance

```r
cov(data$spent, data$raised, use="complete.obs")
```

```
## [1] 249439228564
```

Analysis of variance (ANOVA)

```r
anova_model1 <- aov(spent ~ chamber, data=data)
#one factor
summary(anova_model1)
```

```
##              Df         Sum Sq       Mean Sq F value    Pr(>F)
## chamber       1  2490982272237 2490982272237    21.9 0.0000037 ***
## Residuals   537 61189570595049  113947058836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

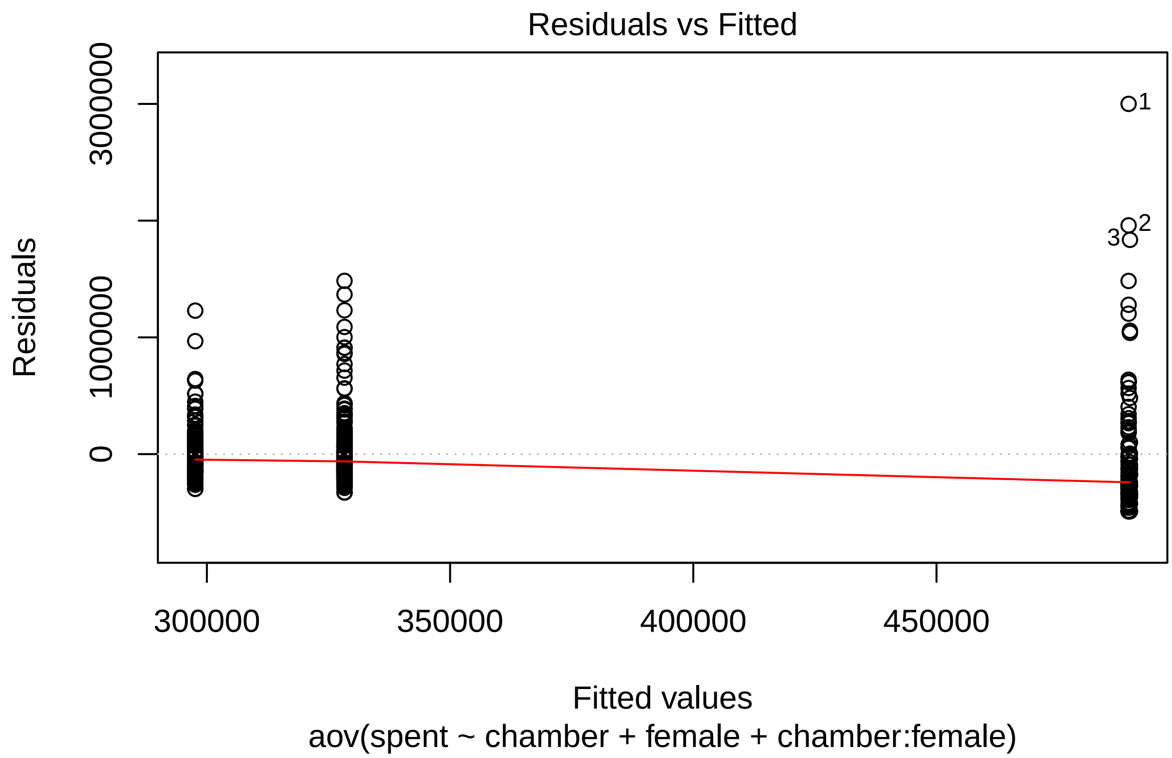
```r
anova_model2 <- aov(spent ~ chamber + female, data=data)
#two factors
summary(anova_model2)
```

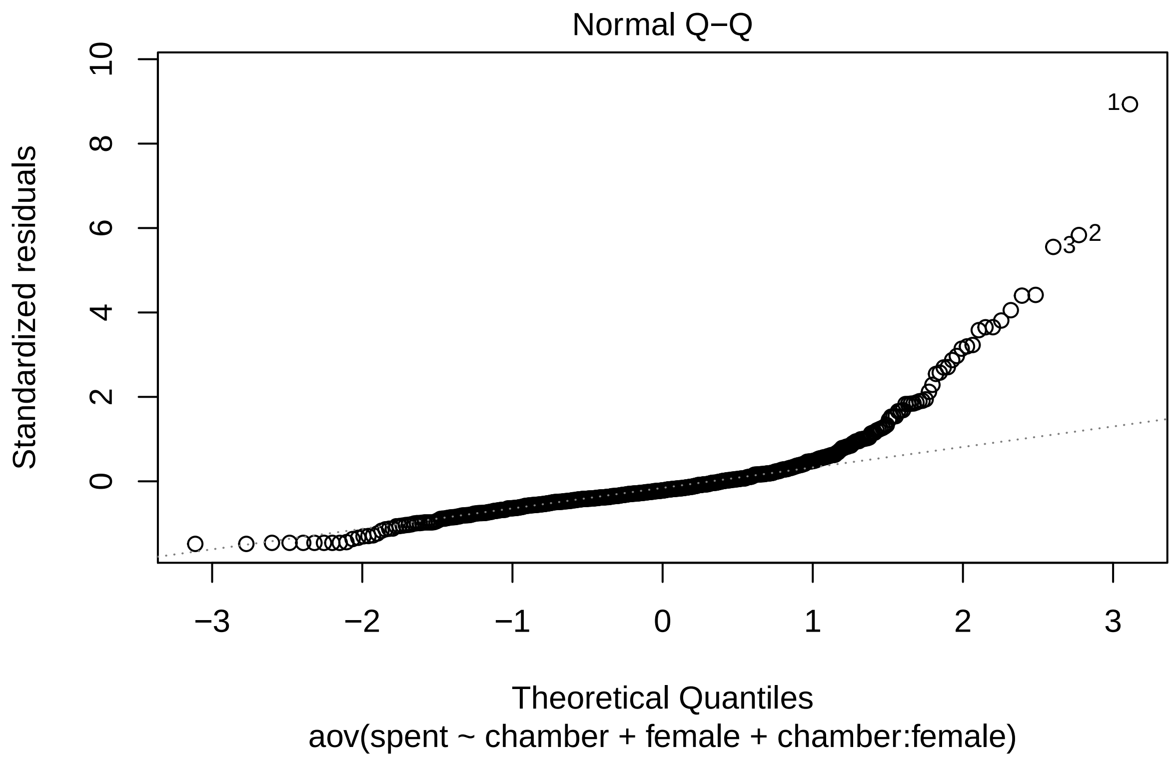
```
##              Df         Sum Sq       Mean Sq F value    Pr(>F)
## chamber       1  2490982272237 2490982272237   21.85 0.0000037 ***
## female        1    76974169397   76974169397    0.68      0.41
## Residuals   536 61112596425652  114016038108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

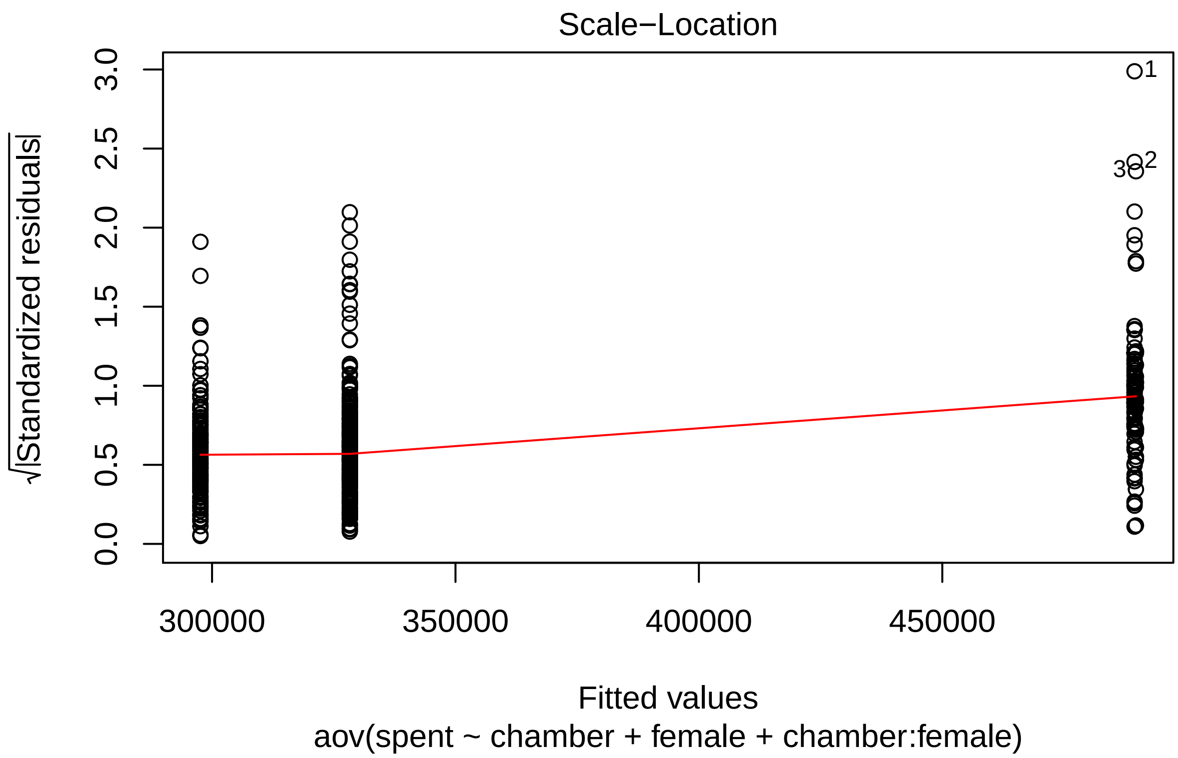
```r
anova_model3 <- aov(spent ~ chamber + female +chamber:female , data=data)
#factorial design
summary(anova_model3)
```

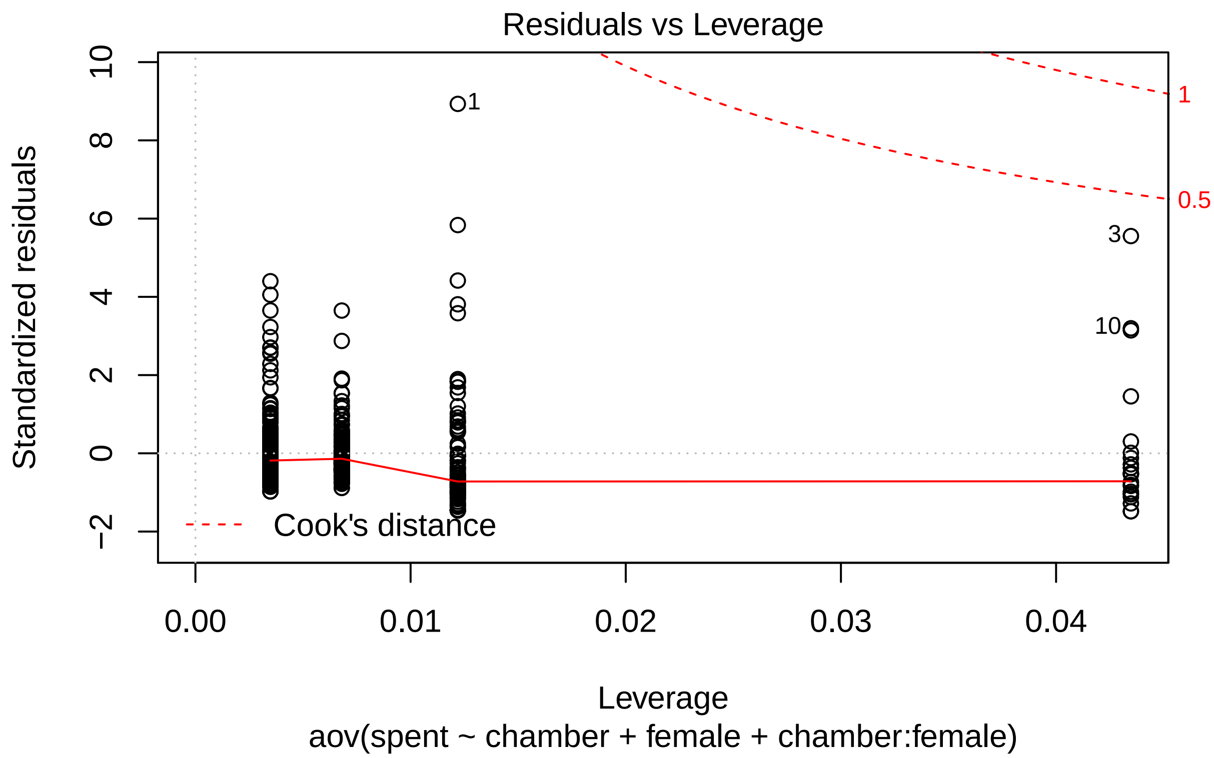
```
##                 Df         Sum Sq       Mean Sq F value    Pr(>F)
## chamber          1  2490982272237 2490982272237   21.81 0.0000038 ***
## female           1    76974169397   76974169397    0.67      0.41
## chamber:female   1    14536171577   14536171577    0.13      0.72
## Residuals      535 61098060254075  114201981783
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
plot(anova_model3)
```


Residuals vs Fitted
Residuals
3000000
1000000
0
300000
350000
400000
450000
Fitted values
aov(spent ~ chamber + female + chamber:female)
1
2
3

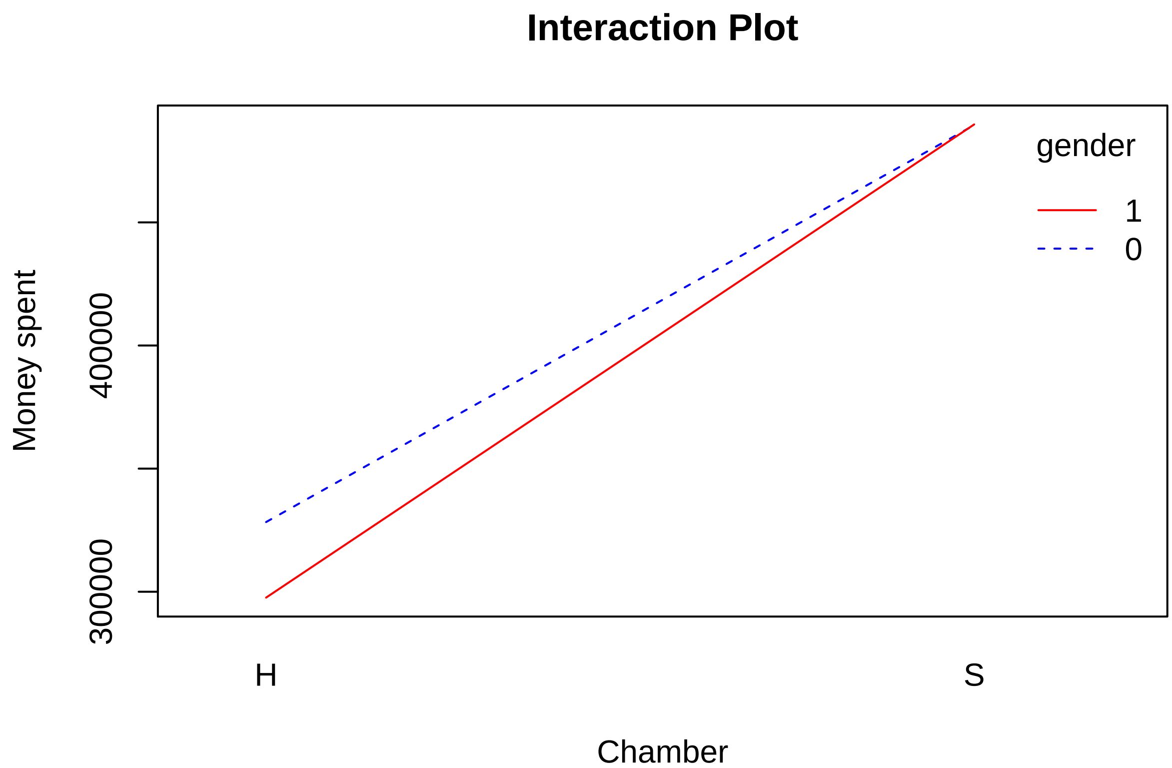

Normal Q−Q
Standardized residuals
Theoretical Quantiles
aov(spent ~ chamber + female + chamber:female)
−3
−2
−1
0
1
2
3
0
2
4
6
8
10
1
2
3


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
300000
350000
400000
450000
Fitted values
aov(spent ~ chamber + female + chamber:female)
1
2
3



Interaction plot for two-way factorial design:

```
house <- factor(data$chamber)
gender <- factor(data$female)
interaction.plot(house, gender, data$spent,
                 col=c("blue", "red"),
                 xlab="Chamber",
                 ylab="Money spent",
                 main="Interaction Plot")
```



TOPIC 9: Hypothesis testing

Independent 2-group t-test.

`Test if the means for two groups are equal.`

Is the average money raised the same for House and Senate? The t-test tests if the difference between the two groups' averages is unlikely to have occurred because of random chance in sample selection.

H0: There is no difference in means between the two groups. The difference is 0.

HA: The difference in means is not equal to 0.

```
t.test(data$spent~data$chamber)
```

```
##
##  Welch Two Sample t-test
##
## data:  data$spent by data$chamber
## t = -2.988, df = 113.5, p-value = 0.00345
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -285469.3  -57827.9
## sample estimates:
## mean in group H mean in group S
##          317902          489551
```

The p-value is the probability of observing a greater absolute value of the t-statistic under the null hypothesis. If the p-value associated with the test-statistic is smaller than 0.05, then there is evidence that the mean is different from the reference value. If the p-value associated with the t-test is not small ($p > 0.05$), then the null hypothesis cannot be rejected.

To change the required confidence level:

```r
t.test(data$spent~data$chamber, conf.level=0.99)
```

```
##
##  Welch Two Sample t-test
##
## data:  data$spent by data$chamber
## t = -2.988, df = 113.5, p-value = 0.00345
## alternative hypothesis: true difference in means is not equal to 0
## 99 percent confidence interval:
##  -322168.0  -21129.1
## sample estimates:
## mean in group H mean in group S
##          317902          489551
```

By default, R assumes unequal variances, so it applies a df correction. To get the t-test without the correction:

```r
t.test(data$spent~data$chamber, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  data$spent by data$chamber
## t = -4.676, df = 537, p-value = 0.00000371
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -243765.0  -99532.1
## sample estimates:
## mean in group H mean in group S
##          317902          489551
```

Also by default, a two-tailed test is done. To do a one-tailed test use options "less" or "greater":

```r
t.test(data$spent~data$chamber, alternative="less", var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  data$spent by data$chamber
## t = -4.676, df = 537, p-value = 0.00000186
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##     -Inf -111159
## sample estimates:
## mean in group H mean in group S
##          317902          489551
```

What is the alternative hypothesis in this case? Note that tested difference is first group (H) minus second group (S).

TOPIC 10: Regression analysis

Simple Linear Regression Example

```r
model1 <- lm(data$spent~data$raised)
summary(model1) #show results
```

```
##
## Call:
## lm(formula = data$spent ~ data$raised)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1732607  -106432   -30715    52748  1806346
##
## Coefficients:
##                Estimate  Std. Error t value            Pr(>|t|)
## (Intercept) 152998.0702  12015.3037    12.7 <0.0000000000000002 ***
## data$raised      0.2764      0.0101    27.4 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 223000 on 537 degrees of freedom
## Multiple R-squared:  0.582,  Adjusted R-squared:  0.582
## F-statistic:  749 on 1 and 537 DF,  p-value: <0.0000000000000002
```

The same thing as:

```r
model2 <- lm(spent~raised, data=data)
summary(model2)
```

```
##
## Call:
## lm(formula = spent ~ raised, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1732607  -106432   -30715    52748  1806346
##
## Coefficients:
##                Estimate  Std. Error t value            Pr(>|t|)
## (Intercept) 152998.0702  12015.3037    12.7 <0.0000000000000002 ***
## raised           0.2764      0.0101    27.4 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 223000 on 537 degrees of freedom
## Multiple R-squared:  0.582,  Adjusted R-squared:  0.582
## F-statistic:  749 on 1 and 537 DF,  p-value: <0.0000000000000002
```

Multiple Linear Regression Example Generate a new variable, 'house' which is 1 of chamber==house and 0 if not.

```r
data$house <- ifelse(data$chamber=="H", 1, 0)
```

Although R understands that chamber is a binary variable and you can introduce it in the model just as it is, along with other covariates.

```r
model3 <- lm(spent~raised+house+incumbent+reelected+female+income, data=data)
summary(model3)
```

```
##
## Call:
## lm(formula = spent ~ raised + house + incumbent + reelected +
##     female + income, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1673052  -106344   -23993    61200  1621689
##
## Coefficients:
##                Estimate  Std. Error t value             Pr(>|t|)
## (Intercept) 117073.8826  32031.1760    3.65              0.00028 ***
## raised           0.2680      0.0104   25.72 < 0.0000000000000002 ***
## house       -19937.3182  24844.3036   -0.80              0.42263
## incumbent     8124.7961  23799.8962    0.34              0.73295
## reelected     1050.7943   7502.6134    0.14              0.88867
## female       -9488.7648  20596.6394   -0.46              0.64521
## income           0.3228      0.0784    4.12             0.000044 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 220000 on 530 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.597,  Adjusted R-squared:  0.592
## F-statistic:  131 on 6 and 530 DF,  p-value: <0.0000000000000002
```

Useful functions for getting info from the model

```
coefficients(model3)  # model coefficients
```

```
##   (Intercept)        raised         house     incumbent     reelected
## 117073.882573      0.267993 -19937.318190   8124.796144   1050.794349
##        female        income
##  -9488.764836      0.322755
```

```
confint(model3, level=0.95)  # CIs for model parameters
```

```
##                     2.5 %        97.5 %
## (Intercept)  54150.237911 179997.527234
## raised           0.247526      0.288459
## house       -68742.711303  28868.074923
## incumbent   -38628.910668  54878.502956
## reelected   -13687.714889  15789.303586
## female      -49949.833772  30972.304101
## income           0.168825      0.476685
```

```
fitted_values <- fitted(model3) #  predicted values
```

```
model_residual <- residuals(model3)  # residuals
```

TOPIC 11: Regression diagnostics

Great package for regression analysis:

```
install.packages("car")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
```

```
## (as 'lib' is unspecified)
```

```
## package 'car' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

```r
library("car")
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

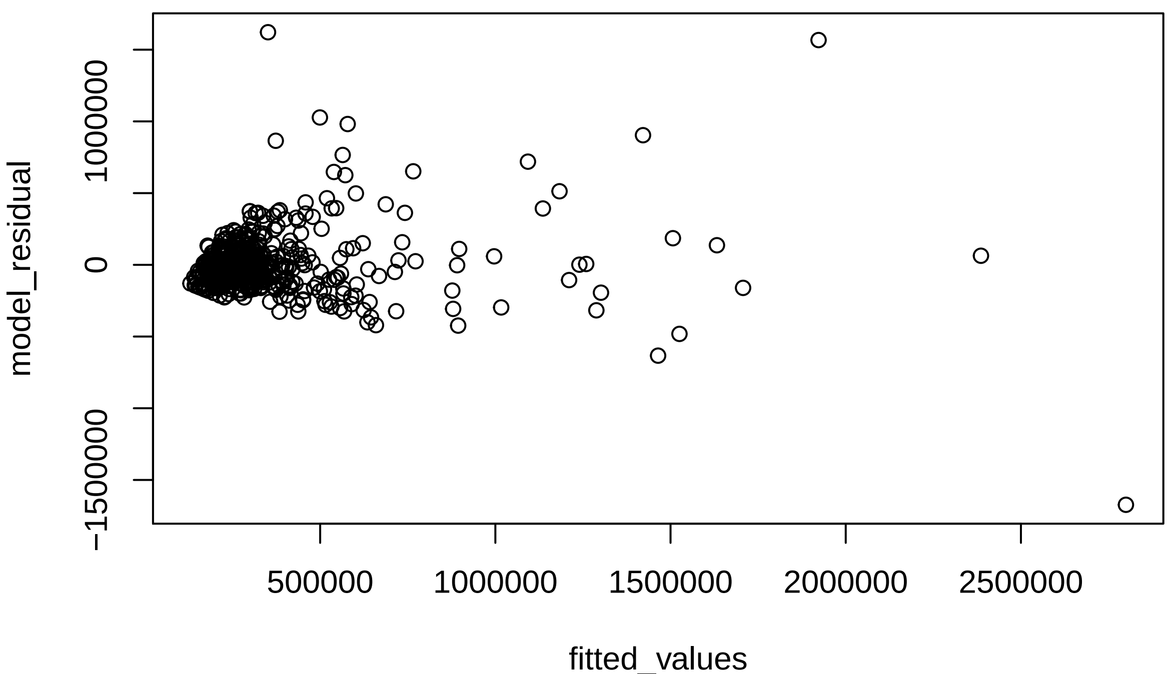
```
## The following object is masked from 'package:psych':
##
##     logit
```

Most commonly used: plot residuals vs fitted values.

- Linearity. The residuals "bounce randomly" around the 0 line. This suggests that the assumption that the relationship is linear is reasonable.
- Homoskedasticity (equal variances of the error terms). The residuals roughly form a "horizontal band" around the 0 line.
- No outliers. No residual "stands out" from the basic random pattern of residuals.

Remember we saved the fitted values and residuals from model 3.

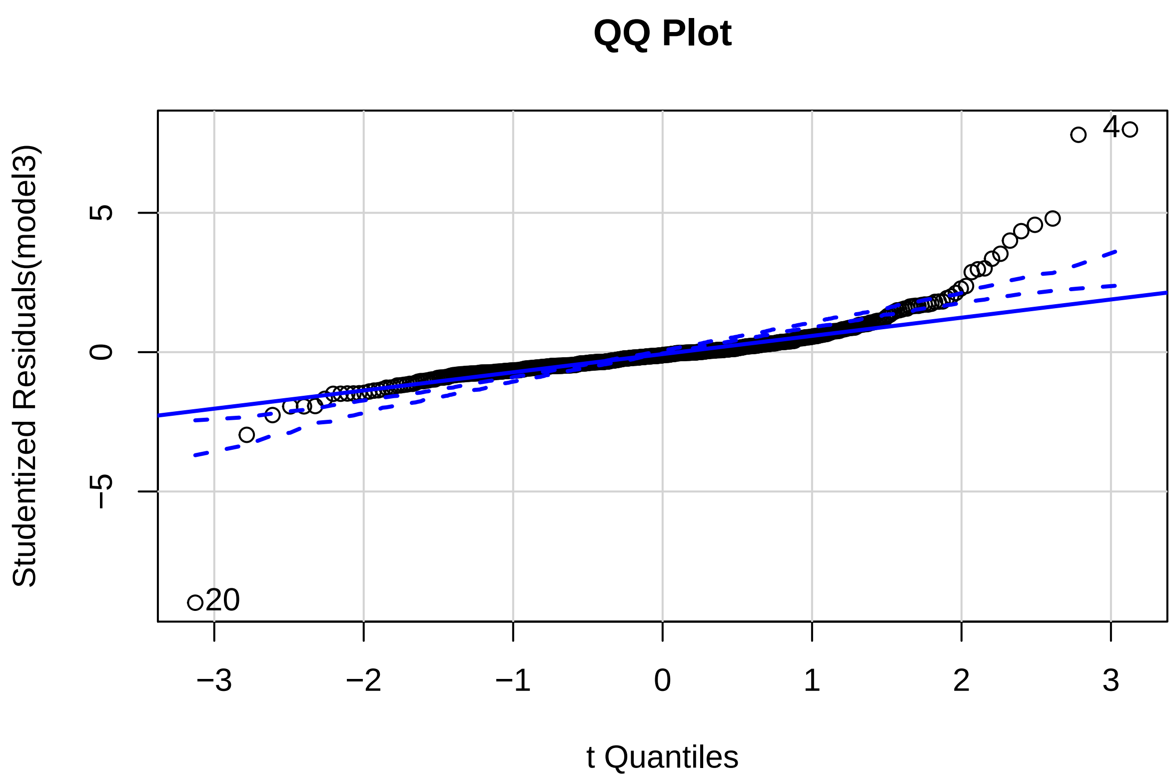
```r
plot(fitted_values, model_residual) # a fitted values vs. residuals plot
```



What problems do you see?

Another useful plot for assessing outliers and normality of residuals:

```r
qqPlot(model3, main= "QQ Plot")
```




```
## [1]  4 20
```

What are the outliers?

```r
outlierTest(model3) #  Bonferonni p-value for most extreme obs
```

```
##    rstudent        unadjusted p-value          Bonferonni p
## 20 -8.99084 0.0000000000000000043543 0.0000000000000023383
## 4   7.98976 0.0000000000000085151000 0.0000000000045726000
## 1   7.80087 0.0000000000000330050000 0.0000000000177240000
## 12  4.79597 0.0000021079000000000002 0.0011318999999999999
## 9   4.56949 0.0000060908000000000002 0.0032707999999999999
## 3   4.34148 0.0000169639999999999989 0.0091094000000000001
## 17  4.00259 0.0000716019999999999965 0.0384509999999999991
```

Test for Autocorrelated Errors

```r
durbinWatsonTest(model3)
```

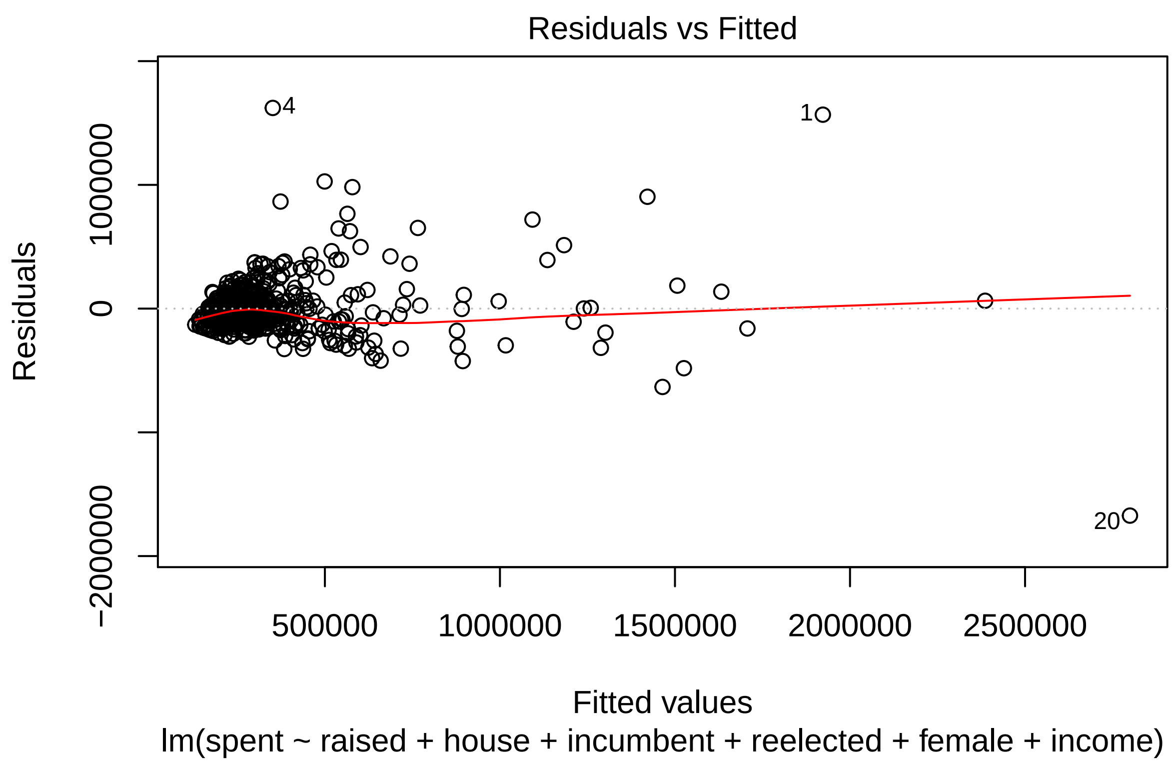
```
##  lag Autocorrelation D-W Statistic p-value
##    1        0.289885       1.32337       0
##  Alternative hypothesis: rho != 0
```

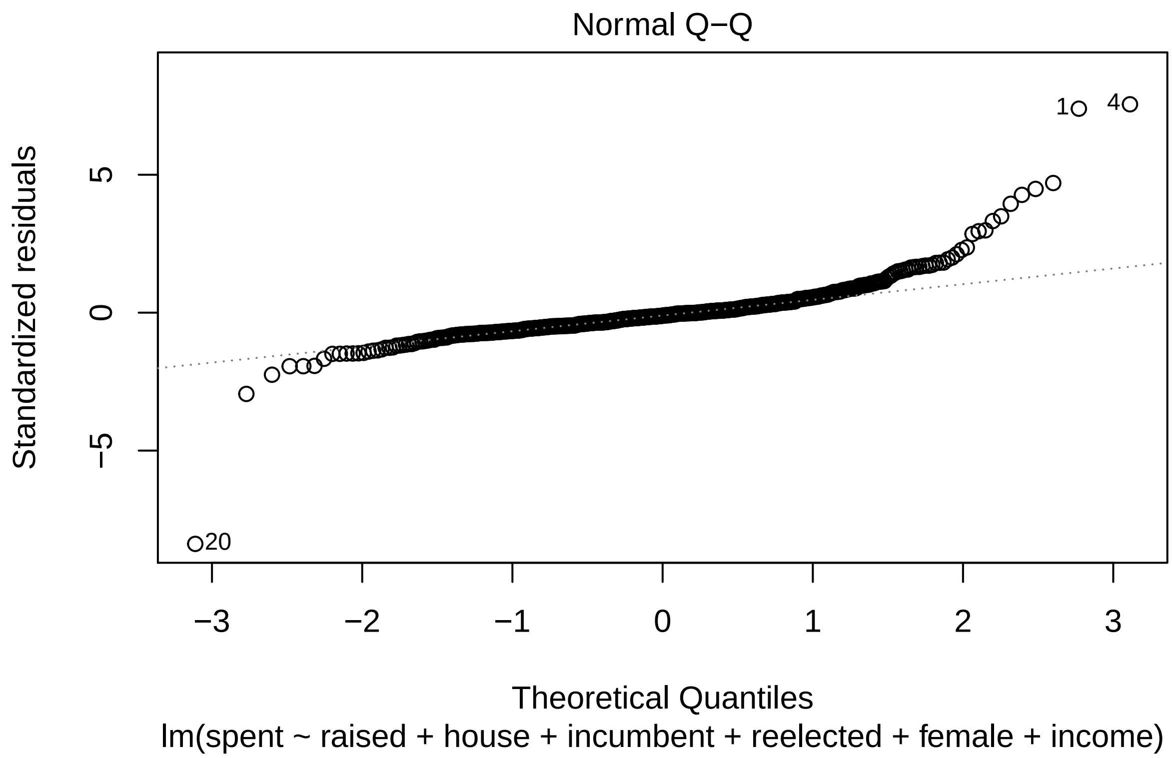
The null hypothesis is that there is no correlation among residuals, i.e., they are independent. Can we reject the null?

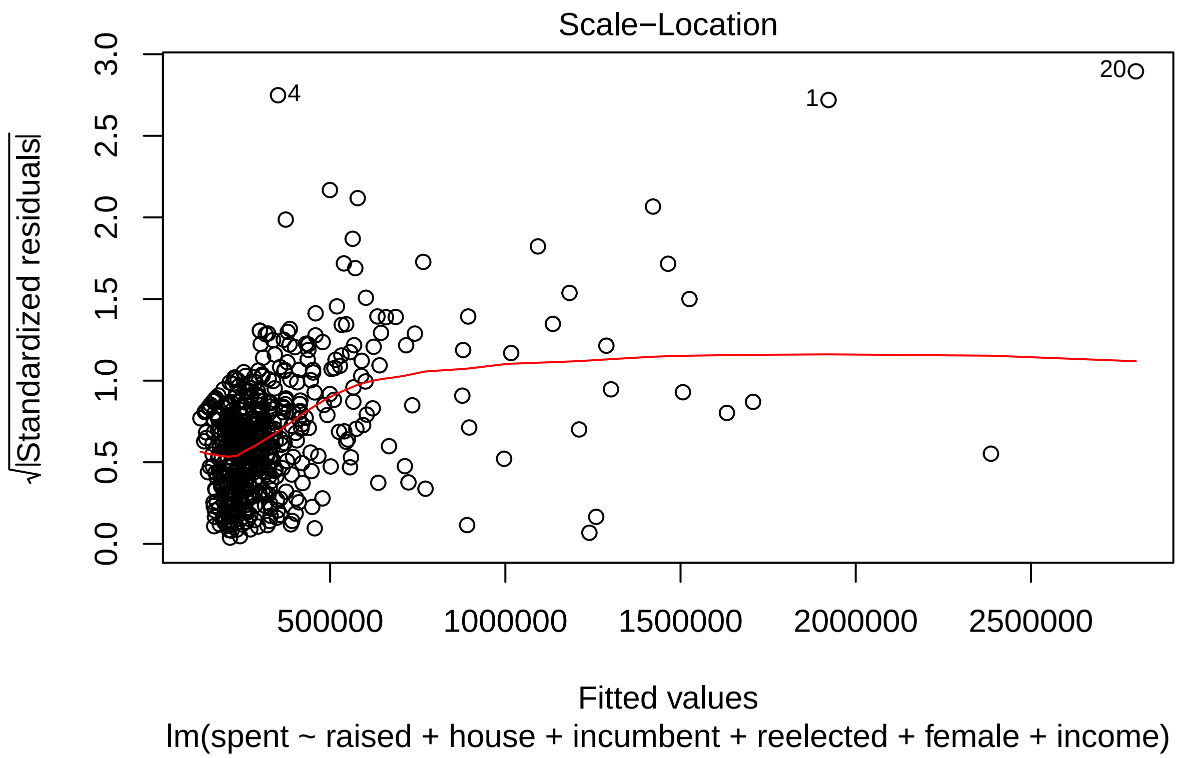
Combined diagnostic plots

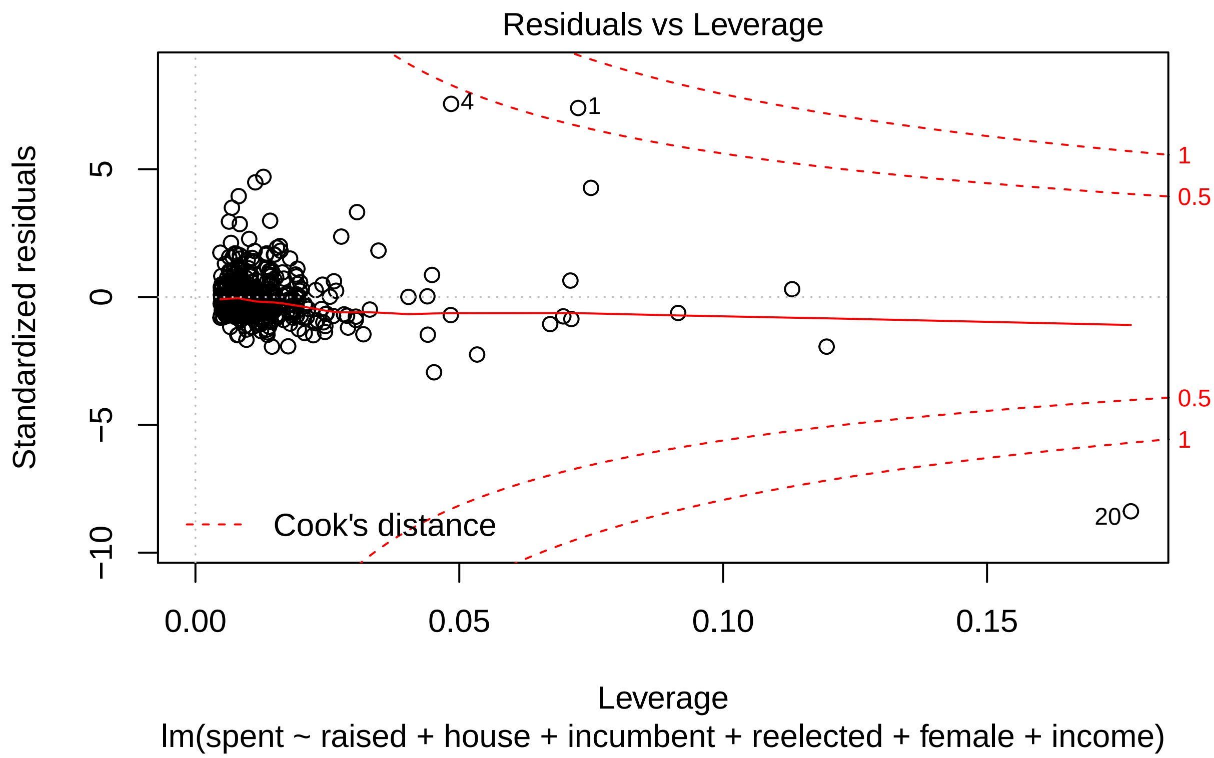
```
layout(matrix(c(1,2,3,4),2,2))  # optional 4 graphs/page
```

```
plot(model3)
```




Normal Q−Q
Standardized residuals
5
0
−5
1
4
20
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(spent ~ raised + house + incumbent + reelected + female + income)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
3.0
4
1
20
500000
1000000
1500000
2000000
2500000
Fitted values
lm(spent ~ raised + house + incumbent + reelected + female + income)



The Scale-Location plot should look random, with no patterns. Cook's distance plot tells us which points have the greatest influence on the regression (leverage points).

```
par(mfrow=c(1,1))    # This plot area back to a single cell
```

TOPIC 12: Transformations and interaction terms

```
attach(data)
```

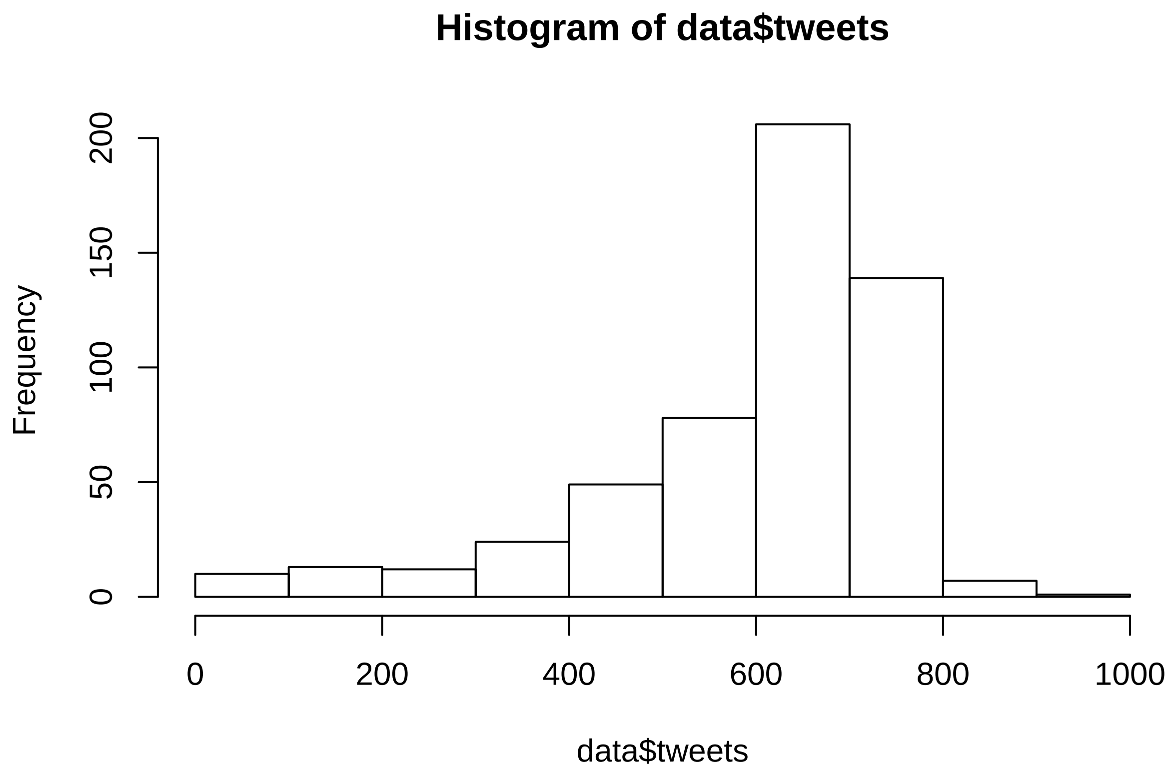
```
## The following object is masked _by_ .GlobalEnv:
##
##     house
```

```
## The following objects are masked from data (pos = 5):
##
##     chamber, committee, female, id, income, incumbent, party,
##     raised, reelected, region, run_again, spent, support, tweets
```

```
## The following object is masked from package:psych:
##
##     income
```

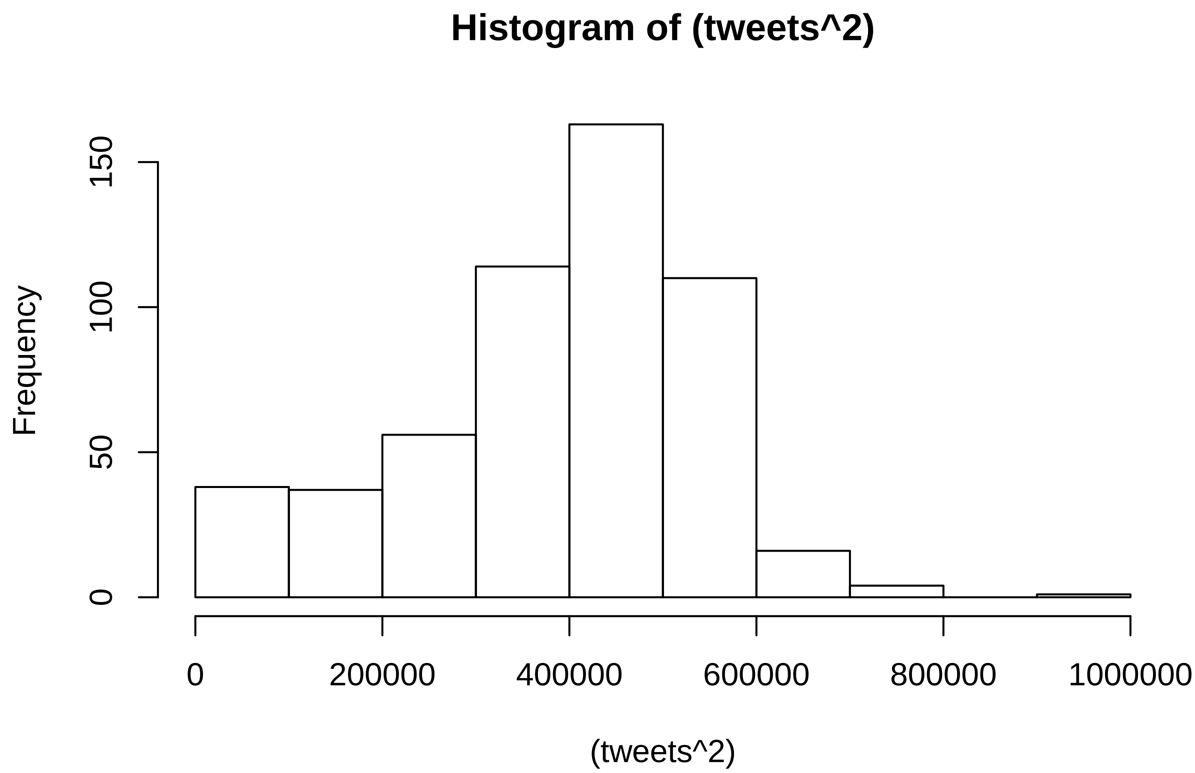
Transformations imply replacing a variable by a function of itself, to change the shape of a distribution or relation. Left skewness example:

```
hist(data$tweets)
```



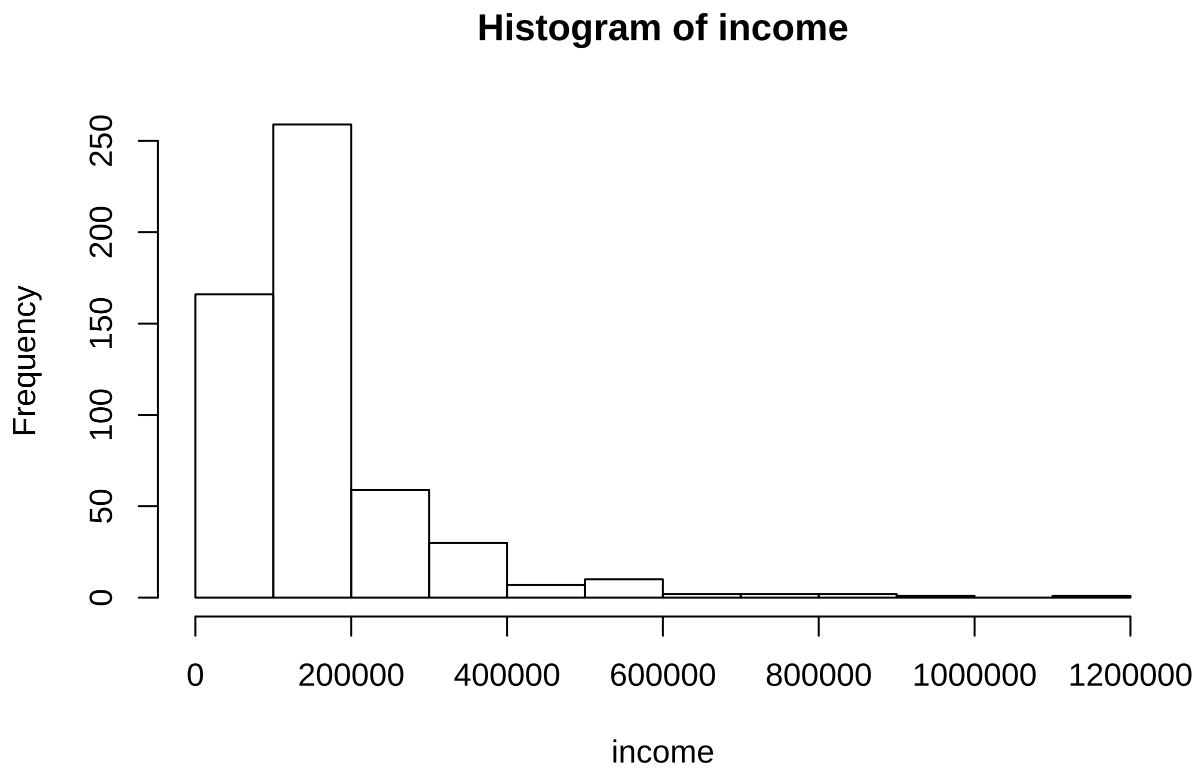
To reduce left skewness we can use polynomial (square, cube or higher order) transformations.

```
hist((tweets^2))
```



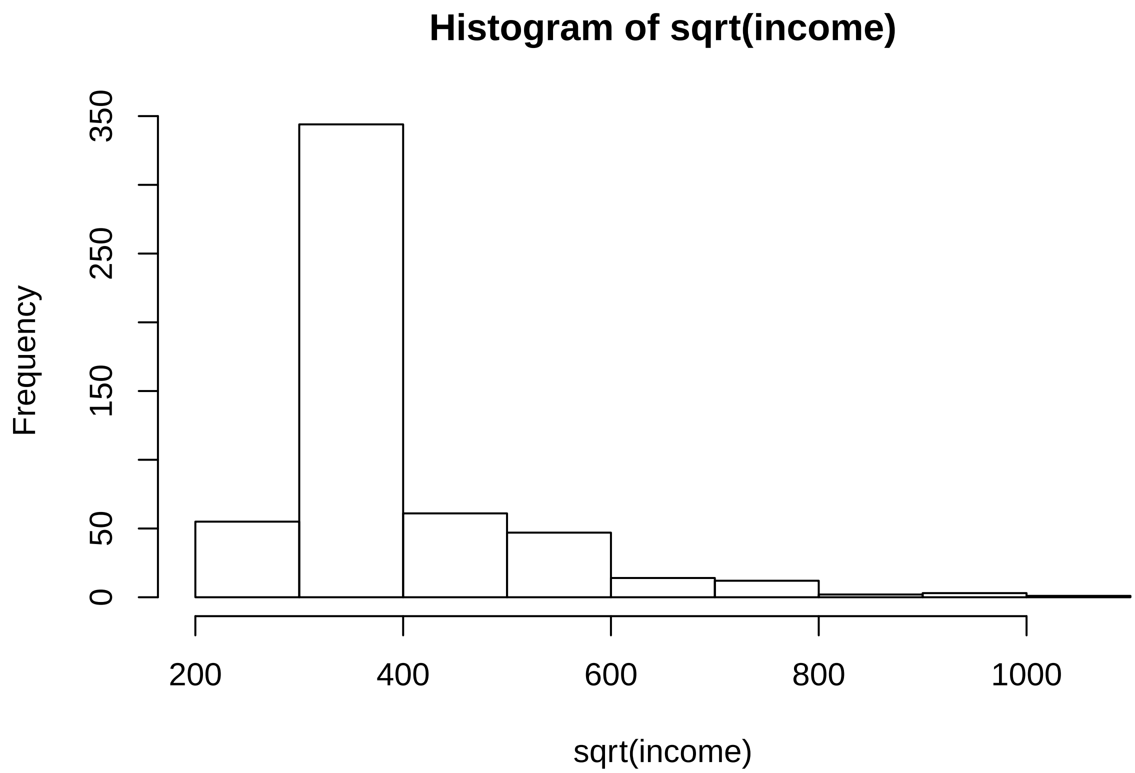
Right skewness example:

```
hist(income)
```

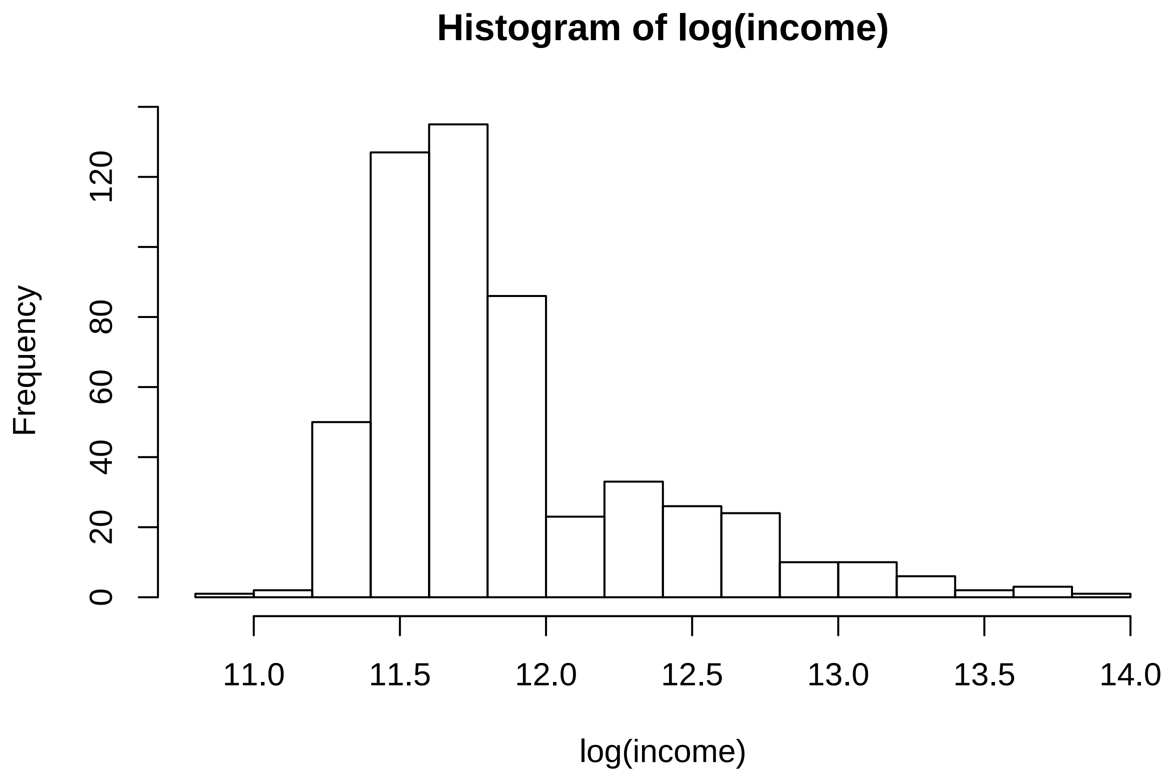


To reduce right skewness we can use root, log or reciprocal transformations.

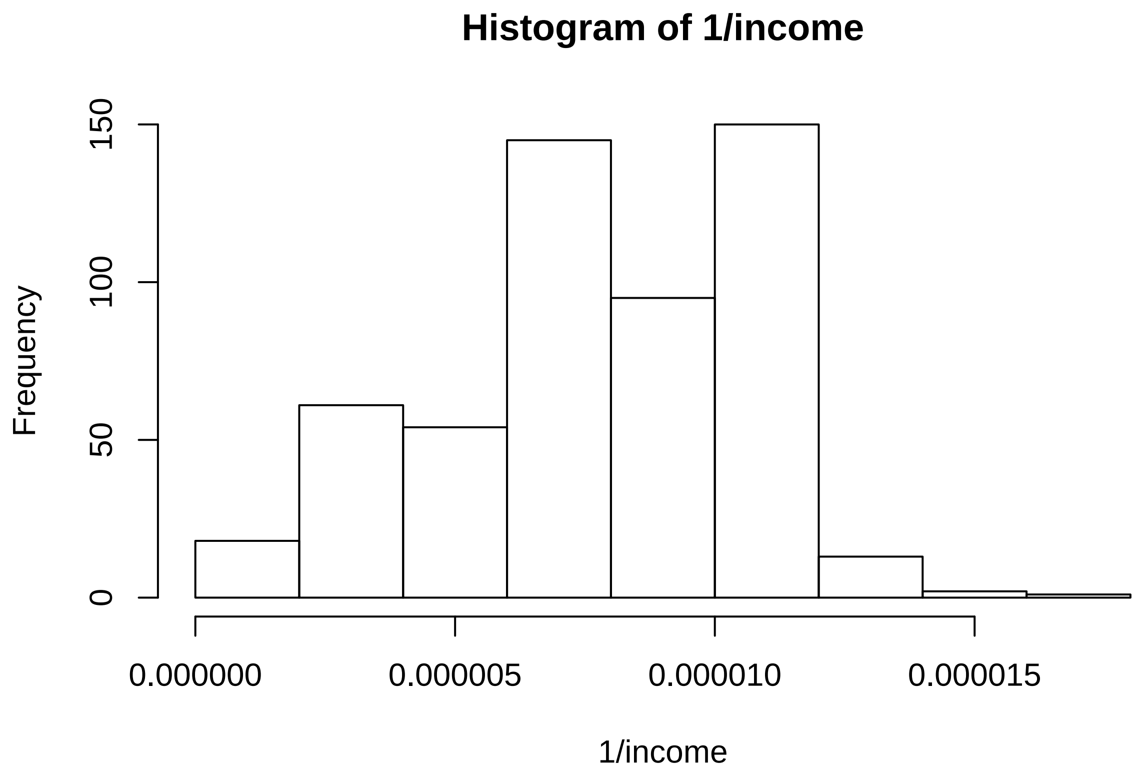
```
hist(sqrt(income))
```




```r
hist(log(income))
```




```
hist(1/income)
```



Adding interaction terms to regression analysis is easy:

```
model_int <- lm(spent~raised+house+raised*house, data=data)
summary(model_int)
```

```
##
## Call:
## lm(formula = spent ~ raised + house + raised * house, data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1565179  -102729   -23774    64503  1765467
##
## Coefficients:
##                  Estimate  Std. Error t value             Pr(>|t|)
## (Intercept)  194923.7830  25810.2080    7.55     0.00000000000019 ***
## raised            0.2550      0.0123   20.79 < 0.0000000000000002 ***
## house        -72923.3403  30220.8398   -2.41                0.016 *
## raised:house      0.0652      0.0226    2.89                0.004 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 221000 on 535 degrees of freedom
## Multiple R-squared:  0.59,   Adjusted R-squared:  0.587
## F-statistic:  256 on 3 and 535 DF,  p-value: <0.0000000000000002
```

TOPIC 13: Plotting regression analysis results

```
install.packages("jtools")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## package 'jtools' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

```
install.packages("broom")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## package 'broom' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

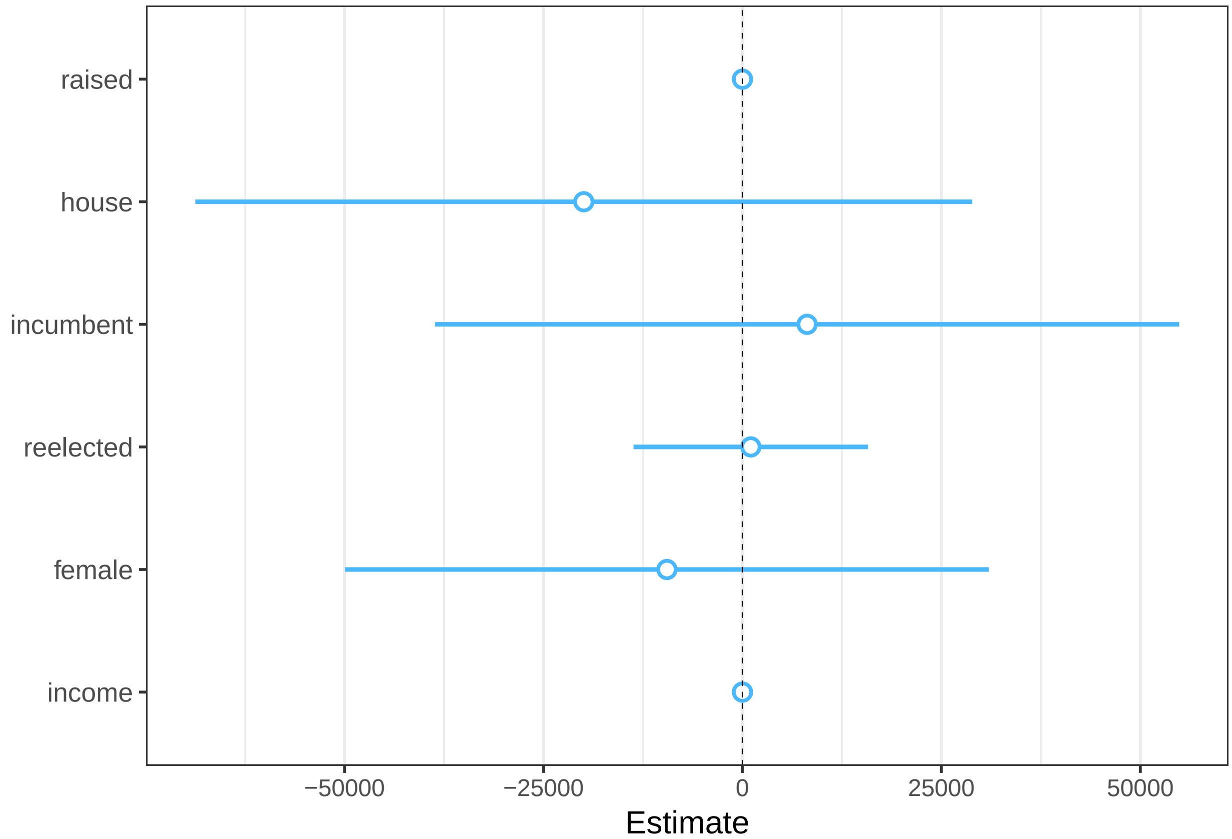
```
install.packages("ggstance")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

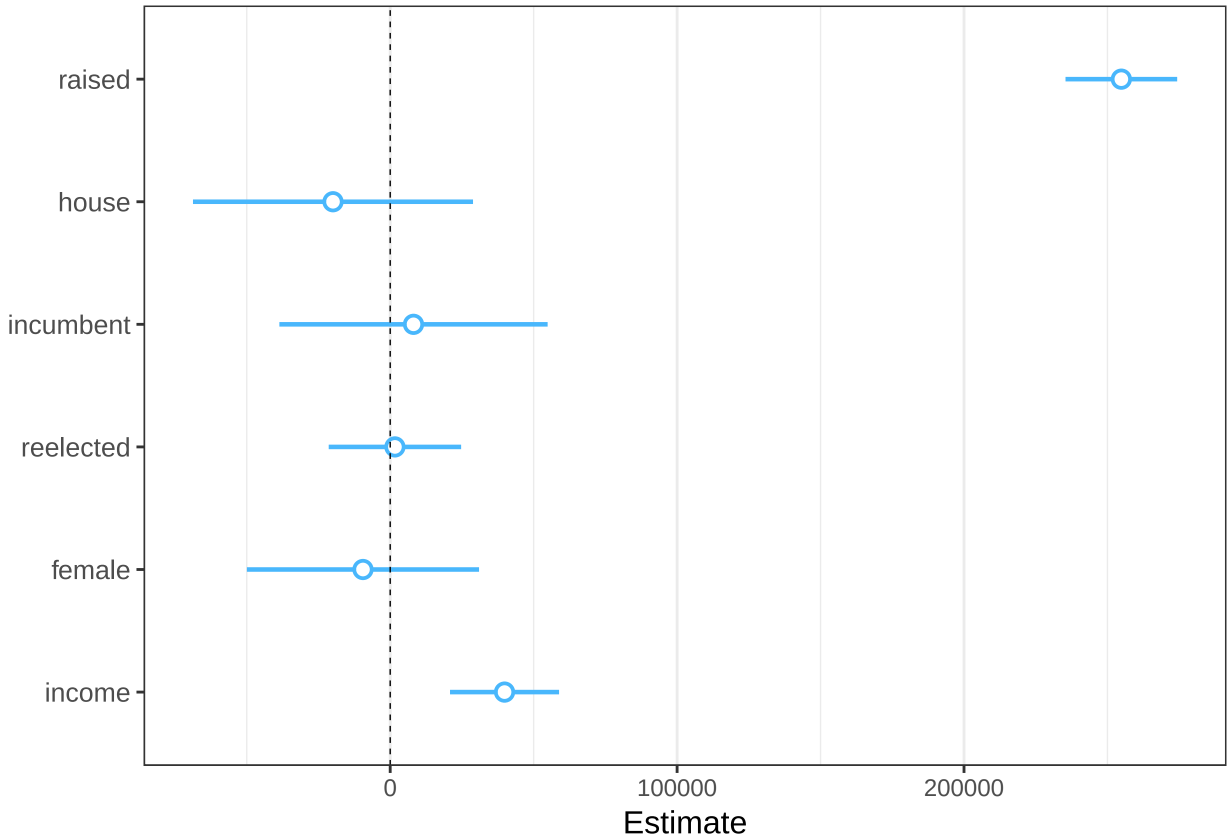
```
## package 'ggstance' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

```
library(jtools)
```

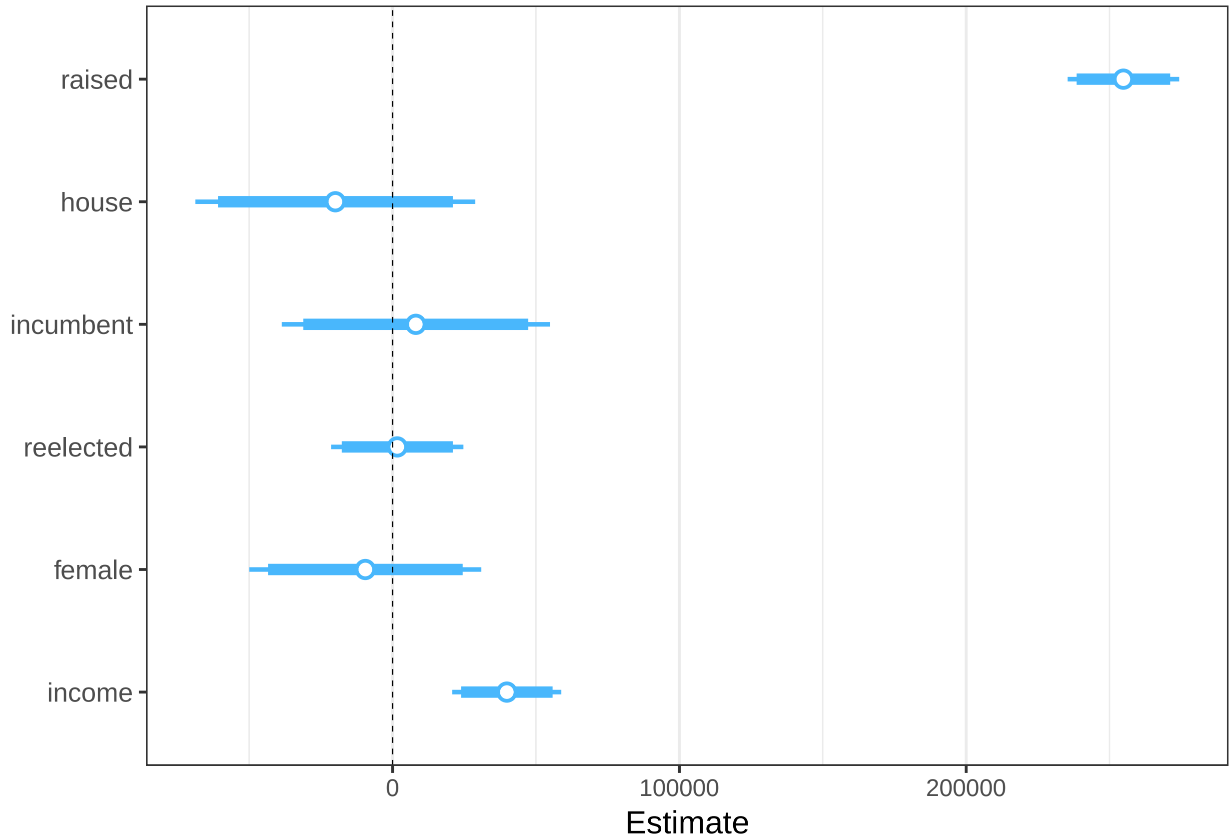
```
plot_summs(model3)   #Plot coefficients with 95% confidence intervals.
```



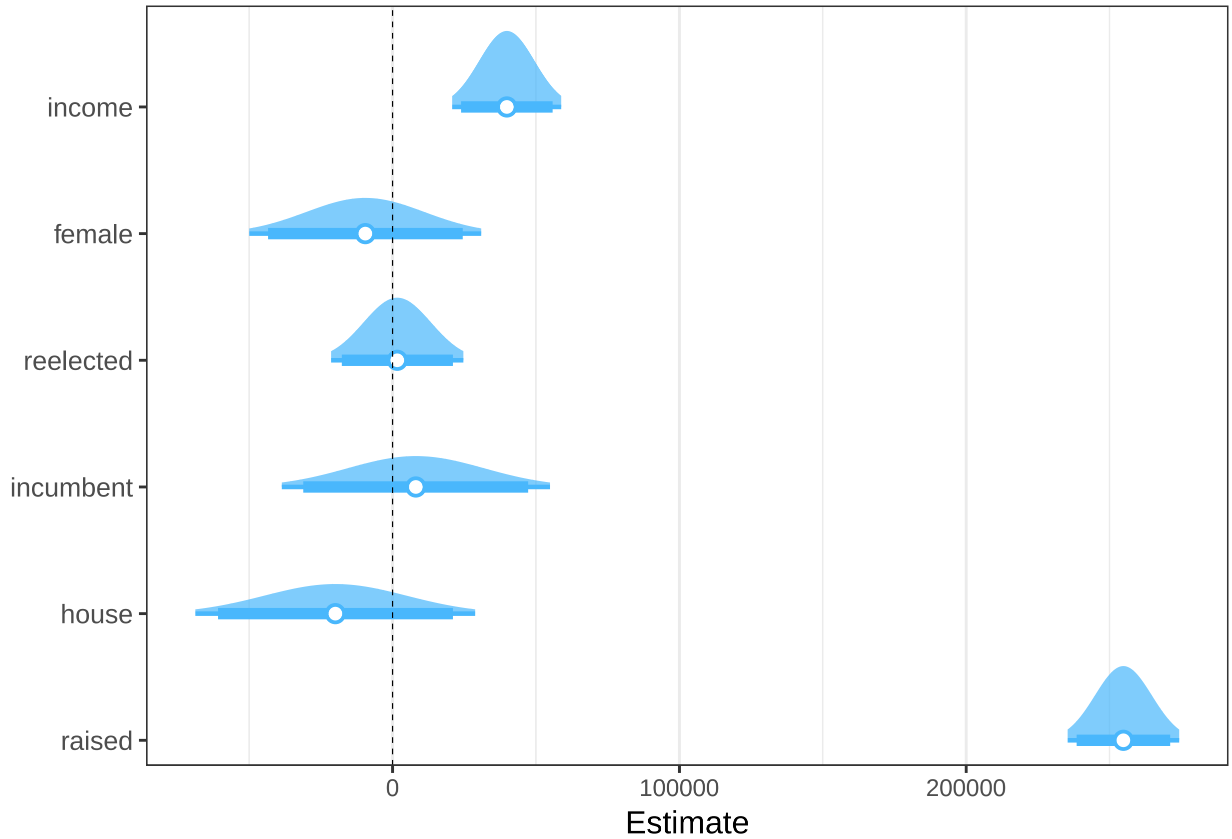

```
plot_summs(model3, scale = TRUE)  # Standardized coefficients.
```




```r
plot_summs(model3, scale = TRUE, inner_ci_level = .9)    #Add 90% confidence intervals
```




```
plot_summs(model3, scale = TRUE, inner_ci_level = .9,
           plot.distributions = TRUE)    #Plot coefficient uncertainty as distribution
```

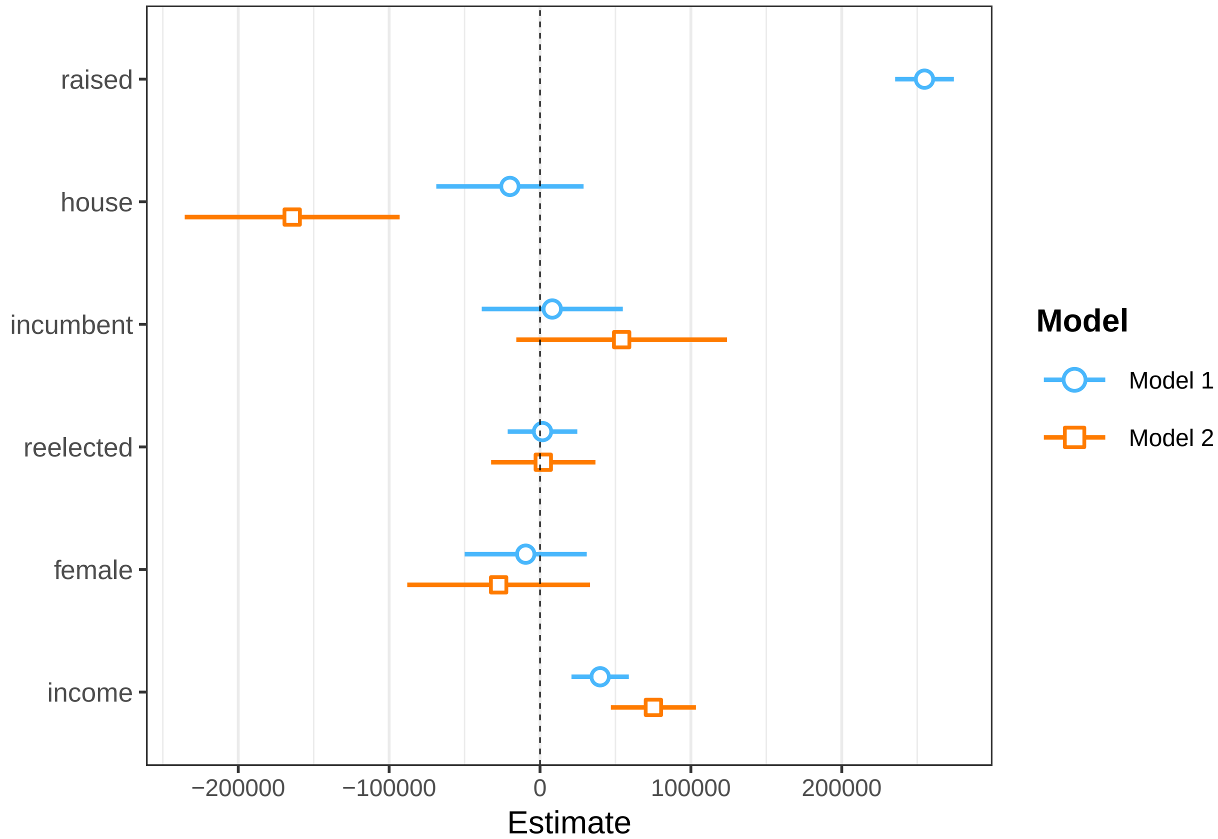


Comparing coefficients across models (note, same DV):

```
model4 <- lm(spent~house+incumbent+reelected+female+income, data=data)
```

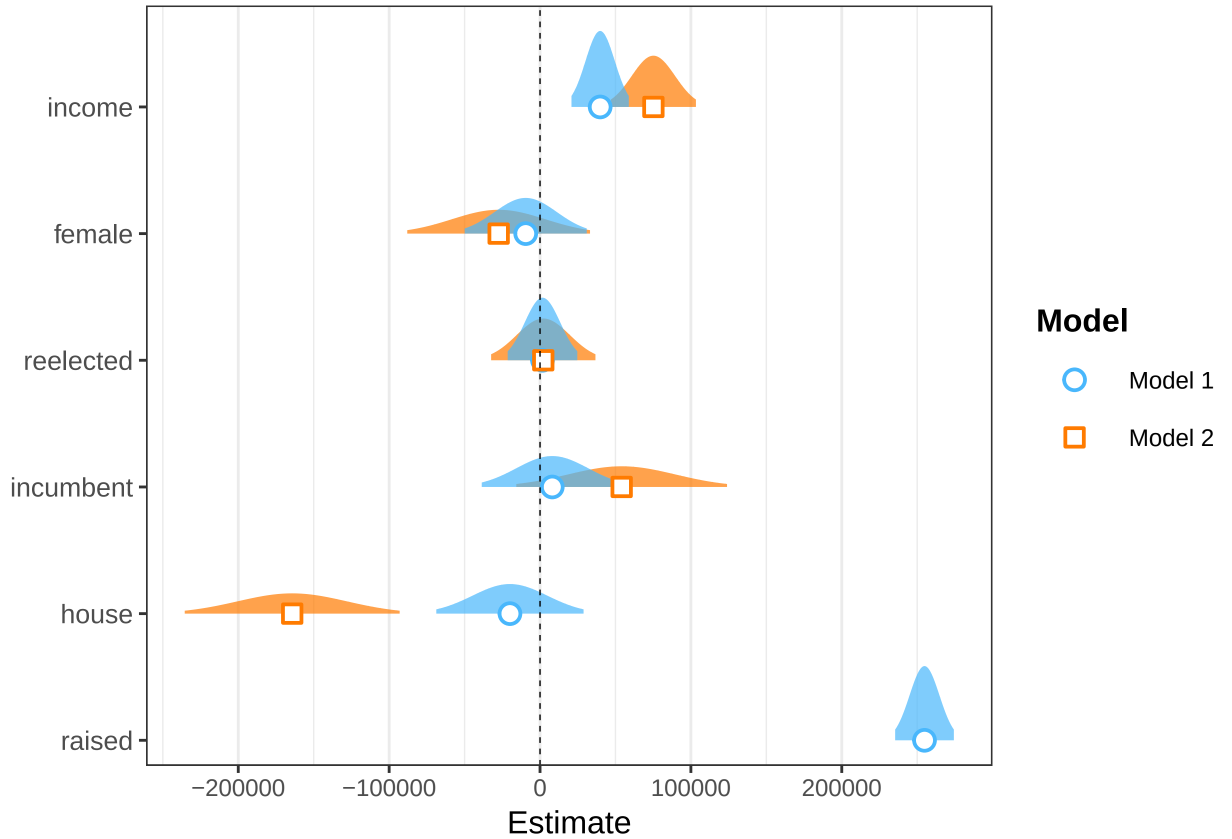
Standardized coefficients with 95% CIs:

```
plot_summs(model3, model4, scale = TRUE)
```



Standardized coefficient distributions:

```
plot_summs(model3, model4, scale = TRUE, plot.distributions = TRUE)
```



Now let's plot the effects of the interaction model:

```r
install.packages("sjPlot")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## package 'sjPlot' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

```r
install.packages("sjmisc")
```

```
## Installing package into 'C:/Users/iulia/Documents/R/win-library/3.5'
## (as 'lib' is unspecified)
```

```
## package 'sjmisc' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\iulia\AppData\Local\Temp\Rtmp0iLnJs\downloaded_packages
```

```r
library(sjPlot)
```

```
## #refugeeswelcome
```

```r
library(sjmisc)
```

```
##
## Attaching package: 'sjmisc'
```

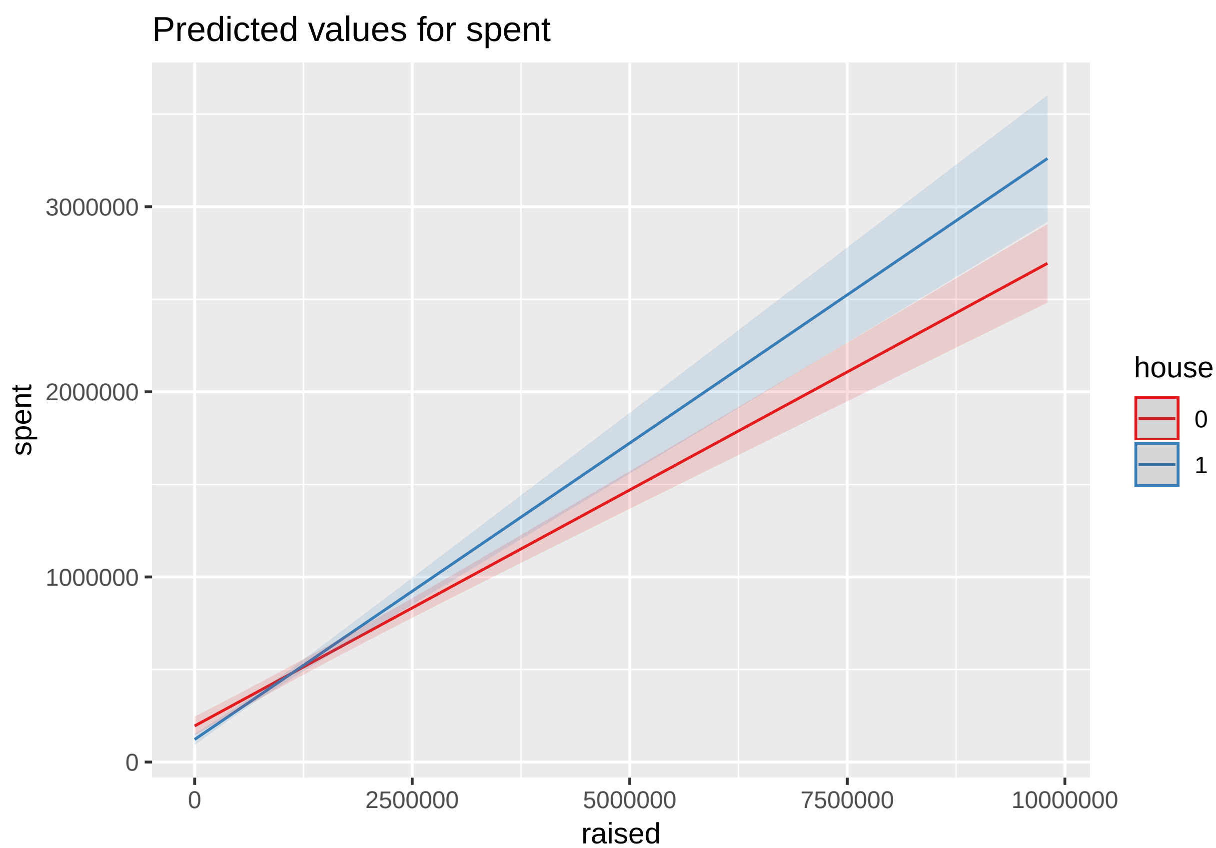
```
## The following object is masked from 'package:jtools':
##
##     center
```

```r
library(ggplot2)
```

```
##
## Attaching package: 'ggplot2'

## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

```r
plot_model(model_int, type = "pred", terms = c("raised", "house"))
```

```
## Argument `include.non.labelled` is deprecated. Please use `non.labelled` instead.

## Argument `include.values` is deprecated. Please use `values` instead.

## Argument `include.non.labelled` is deprecated. Please use `non.labelled` instead.
```



TOPIC 14: Advanced methods

Overview of the variables used in these models.

```r
View(data)
```

Logistic Regression The dependent variable is a binary factor.

```r
model5 <- glm(run_again~raised,data=data,family=binomial)
```

```r
summary(model5)   # display results
```

```
##
## Call:
## glm(formula = run_again ~ raised, family = binomial, data = data)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -2.497  -1.323   0.926   1.025   1.086
##
## Coefficients:
##                Estimate  Std. Error z value Pr(>|z|)
## (Intercept) 0.219660887 0.121184489    1.81    0.070 .
## raised      0.000000368 0.000000134    2.75    0.006 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 718.90  on 538  degrees of freedom
## Residual deviance: 708.88  on 537  degrees of freedom
## AIC: 712.9
##
## Number of Fisher Scoring iterations: 4
```

For every one unit increase in money raised, the log odds of running again (versus not running) increase by 0.000000368.

```r
confint(model5)  # 95% CI for the coefficients
```

```
## Waiting for profiling to be done...
```

```
##                       2.5 %         97.5 %
## (Intercept) -0.021059408520 0.454614076917
## raised       0.000000128333 0.000000655392
```

```r
exp(coef(model5))  # exponentiated coefficients
```

```
## (Intercept)      raised
##     1.24565     1.00000
```

Coefficients as odds ratios. For a one unit increase in raised, the odds of running again (versus not running again) increase by a factor of 1.00001.

```r
exp(confint(model5))  # 95% CI for exponentiated coefficients
```

```
## Waiting for profiling to be done...
```

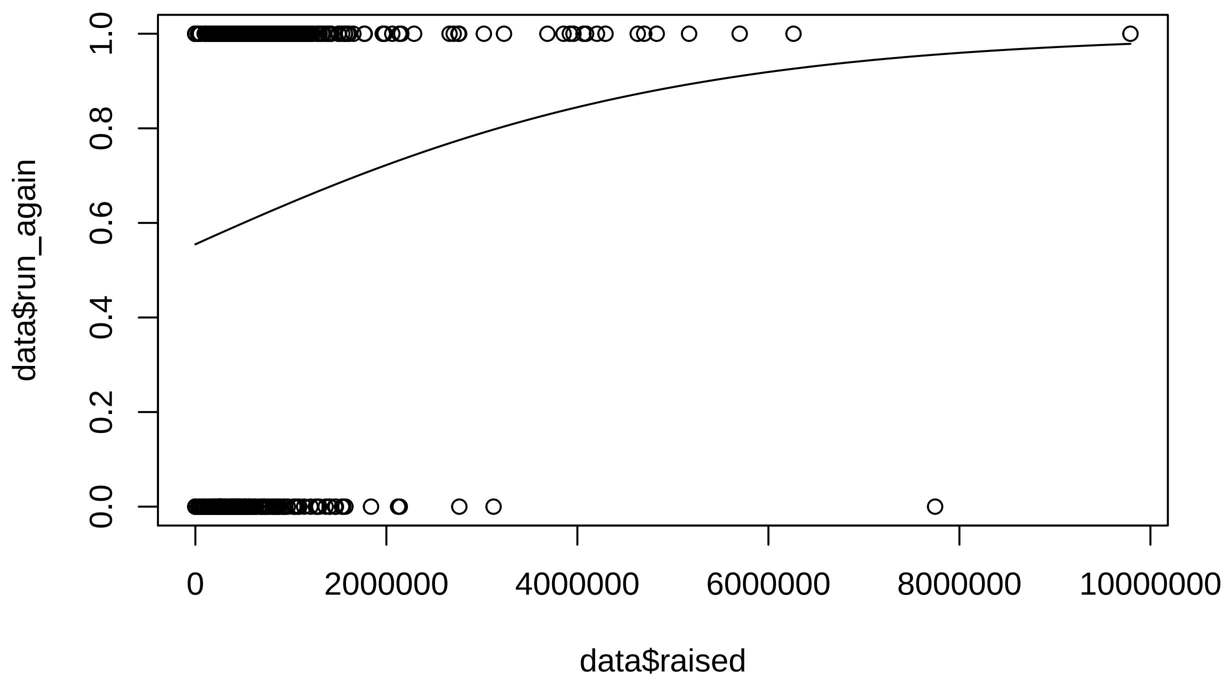
```
##                2.5 %  97.5 %
## (Intercept) 0.979161 1.57557
## raised      1.000000 1.00000
```

```r
predicted_logit <- predict(model5, type="response")  # predicted values
```

```r
residuals_logi <- residuals(model5, type="deviance") #  residuals
```

Plotting predicted probabilities

```r
plot(data$raised, data$run_again)
curve(predict(model5,data.frame(raised=x),type="response"),add=TRUE)
```



Multinomial logistic regression

```r
library(nnet)
```

```r
model6 <- multinom(committee ~ raised+female+income, data=data)
```

```
## # weights:  20 (12 variable)
## initial  value 747.212661
## iter  10 value 482.843121
## iter  20 value 456.504546
## final  value 456.504040
## converged
```

```r
summary(model6)
```

```
## Call:
## multinom(formula = committee ~ raised + female + income, data = data)
##
## Coefficients:
##           (Intercept)           raised     female       income
## Education    -3.04579  0.0000000206246  0.0594853 0.0000285634
## Health       -3.81656  0.0000001642592  0.3012381 0.0000328887
## Other        -4.52618 -0.0000001299842 -0.0762076 0.0000545466
##
## Std. Errors:
```

```
##                   (Intercept)         raised              female
## Education 0.0000000000198015 0.000000219505 0.00000000000665312
## Health    0.0000000000190210 0.000000204087 0.00000000000733658
## Other     0.0000000000195588 0.000000194698 0.00000000000585331
##                   income
## Education 0.00000222971
## Health    0.00000222612
## Other     0.00000191041
##
## Residual Deviance: 913.008
## AIC: 937.008
```

By default, the first category is the reference one. But we can change it to another one:

```r
data$committee=relevel(data$committee, ref="Other")
```

```r
model6 <- multinom(committee ~ raised+income+female+incumbent, data=data)
```

```
## # weights:  24 (15 variable)
## initial  value 747.212661
## iter  10 value 544.126312
## iter  20 value 463.216592
## iter  30 value 455.610244
## final  value 455.609114
## converged
```

```r
summary(model6)
```

```
## Call:
## multinom(formula = committee ~ raised + income + female + incumbent,
##     data = data)
##
## Coefficients:
##           (Intercept)          raised        income   female  incumbent
## Economy      4.666698 0.000000140503 -0.0000543673 0.060246 -0.2945837
## Education    1.507514 0.000000153008 -0.0000259358 0.136903 -0.0596368
## Health       0.601076 0.000000289680 -0.0000218128 0.379648  0.2087604
##
## Std. Errors:
##                   (Intercept)         raised        income
## Economy   0.0000000000178800 0.000000194803 0.00000191236
## Education 0.0000000000127563 0.000000169045 0.00000158895
## Health    0.0000000000121390 0.000000143152 0.00000155349
##                         female           incumbent
## Economy   0.00000000000583352 0.00000000000895707
## Education 0.00000000000416362 0.00000000000714900
## Health    0.00000000000462694 0.00000000000756463
##
## Residual Deviance: 911.218
## AIC: 941.218
```

These are the logit coefficients relative to the reference category. Switching from male to female increases the logit coefficient for "Health" committee relative to "Other" committees 0.37 units. Coefficients as relative risk ratios:

```r
exp_coefs=exp(coef(model6))
```

```r
exp_coefs
```

```
##           (Intercept) raised   income  female incumbent
## Economy     106.34602      1 0.999946 1.06210  0.744842
## Education     4.51549      1 0.999974 1.14672  0.942107
## Health        1.82408      1 0.999978 1.46177  1.232150
```

Keeping all other variables constant, switching from male to female means that you are 1.46 times more likely to be on the "Health" committee vs "Other" committees. So the risk (or odds) are 46% higher.

Models for counts: Poisson regression Poisson Regression The dependent variable is a count.

```r
model6 <- glm(reelected ~ raised + income, data=data, family=poisson())
```

```r
summary(model6)   #display results
```

```
##
## Call:
## glm(formula = reelected ~ raised + income, family = poisson(),
##     data = data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6579  -0.6900   0.0308   0.6046   1.6135
##
## Coefficients:
##                 Estimate   Std. Error z value            Pr(>|z|)
## (Intercept) 1.0298952516 0.0432830665   23.79 <0.0000000000000002 ***
## raised      0.0000000233 0.0000000252    0.93                0.35
## income      0.0000002303 0.0000001948    1.18                0.24
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 459.53  on 536  degrees of freedom
## Residual deviance: 456.96  on 534  degrees of freedom
##   (2 observations deleted due to missingness)
## AIC: 1986
##
## Number of Fisher Scoring iterations: 4
```

Poisson regression models the log of the expected count as a function of the predictor variables. For a one unit change in the independent variable, the difference in the logs of expected counts is expected to change by Beta.

TOPIC 15. Extra resources

Useful resources for more advanced topics

- Time series and forecast analysis:

https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html

- Panel data:

https://www.princeton.edu/~otorres/Panel101R.pdf

- Event and survival analysis:

https://rviews.rstudio.com/2017/09/25/survival-analysis-with-r/

- Multilevel modelling:

https://cran.r-project.org/doc/contrib/Bliese_Multilevel.pdf

- Text analysis:

http://kenbenoit.net/pdfs/text_analysis_in_R.pdf

Other intro to R resources

Today's workshop relied heavily on "Introduction to Scientific Programming and Simmulation Using R" by Jones, Millardet and Robinson

https://www.crcpress.com/Introduction-to-Scientific-Programming-and-Simulation-Using-R-Second-Edition/Jones-Maillardet-Robinson/9781466569997.

There are multiple excellent introductions to R online: http://www.r-tutor.com/r-introduction http://tryr.codeschool.com/ http://www.statmethods.net/

Very good resource for more advanced programming: Hadley Wickham's Advanced R (which is not THAT advanced) http://adv-r.had.co.nz/

Contact info

For any questions about this tutorial, please email me at i.cioroianu@bath.ac.uk.