Introduction to R

Jakob Beuschlein

Installing R

Base R
You can get it here: https://www.r-project.org/

Editor
But you will want to use an editor

- RStudio recommended
- Visual Stdio Code
- ...

You may also want to use ChatGPT (but comes at cost. . .)

R as a calculator

You can use all standard arithmetic operations

```r
2 + 2
```

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

```r
2-5/2+2^2
```

```
## [1] 3.5
```

Logical operations

R also comes along with logical operators

```r
2 == 2
```

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

```r
1 < 2 & 3 > 4
```

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

```r
(4 | 1) > 2
```

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

```r
2 %in% 1:5
```

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

Logical operators

R also comes along with logical operators

```r
2 == 2
```

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

```r
1 < 2 & 3 > 4
```

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

```r
(4 | 1) > 2
```

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

```r
2 %in% 1:5
```

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

Use ‘<-’ for assignment

```
x <- 2
x
```

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

```
x*10
```

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

Object-orientation

Everything in R is an object

- vectors
- matrices
- functions
- data frames

Data types

There are 5 main data types: double, integer, complex, logical, character

```r
typeof(1.2)
```

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

```r
typeof(1L)
```

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

```r
typeof(1+1i)
```

```
## [1] "complex"
```

```r
typeof(TRUE)
```

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

```r
typeof("Hello world")
```

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

Vectors

Combine elements of same type

```r
v <- c(1,0,0)
v
```

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

Vectors

There are different ways to do so

```
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```
rep(2,5)
```

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

```
rbind(1,2,3)
```

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

```
cbind(1,2,3)
```

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

Matrcies

Create a 3 × 3 matrix of 0s:

```
m <- matrix(0, ncol=3, nrow = 3)
m
```

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

Or combine vectors

```
v1 <- c(1,2,3)
v2 <- c(4,5,6)
m2 <- rbind(v1,v2)
m2
```

```
##    [,1] [,2] [,3]
## v1    1    2    3
## v2    4    5    6
```

Matrices

Get information about matrices

```
dim(m2)
```

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

```
nrow(m2)
```

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

```
ncol(m2)
```

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

```
length(m2)
```

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

Matrices

We can access elements of matrices

```
m2
```

```
##    [,1] [,2] [,3]
## v1    1    2    3
## v2    4    5    6
```

```
m2[1,3]
```

```
## v1
##  3
```

Lists

Lists are unstructured collections of various data types

```
l <- list("Hello", 2, FALSE)
l
```

```
## [[1]]
## [1] "Hello"
##
## [[2]]
## [1] 2
##
## [[3]]
## [1] FALSE
```

Functions

A function transforms inputs into outputs Try to avoid repetitive code by using functions

```r
f1 <- function(x,y){
  z <- x+y
  return(z)
}
f1(1,2)
```

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

```r
f1(3,4)
```

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

Functions

We can also combine them

```
f2 <- function(x,y,w){
  z <- f1(x,y)
  v <- z * w
  return(v)
}
f2(1,2,3)
```

```
## [1] 9
```

Loops

We can perform tasks iteratively by using loops

```r
for (i in 1:3){
  print(i)
}
```

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

Try to avoid loops as much as possible and use vector based operations instead to increase speed

Matrix Algebra

Let's walk through a regression model. We want to compute

$$\hat{\beta} = (X'X)^{-1}(X'Y)$$

We first create the data

```r
set.seed(101)
X <- cbind(rep(1,100), runif(100), runif(100))
Y <- 0.2*X[,1] - 5*X[,2] + 10*X[,3] + rnorm(100)
lm(Y~X)
```

```
##
## Call:
## lm(formula = Y ~ X)
##
## Coefficients:
## (Intercept)           X1           X2           X3
##    -0.07043           NA     -5.45050     10.89007
```

Matrix algebra

Now by hand:
$(X'X)$

```r
XX <- t(X) %*% X
```

Matrix algebra

$(X'X)^{-1}$

```r
XX <- t(X) %*% X
invXX <- solve(XX)
```

solve(A, b) returns the solution to $b = Ax$
with $b = 1$, $x = A^{-1}$

Matrix algebra

$(X'Y)$

```
XX <- t(X) %*% X
invXX <- solve(XX)
XY <- t(X) %*% Y
```

Matrix algebra

```
beta_hat <- invXX %*% XY
beta_hat
```

```
##             [,1]
## [1,] -0.07042626
## [2,] -5.45050463
## [3,] 10.89006649
```

```
lm(Y~X)
```

```
## 
## Call:
## lm(formula = Y ~ X)
## 
## Coefficients:
## (Intercept)           X1           X2           X3  
##    -0.07043           NA     -5.45050     10.89007
```

Using R's built in regression function

```r
ols <- lm(Y~X)
ols$coefficients[3]
```

```
##        X2
## -5.450505
```

```r
ols$residuals[1]
```

```
##         1
## 0.5948936
```

```r
ols$fitted.values[1]
```

```
##          1
## -0.7376259
```

Working with Data

Packages

A big advantage of R is the large number of packages

```
install.packages("dplyr")
```

Once installed, we will have to load them before each session

```
library(dplyr)
```

Usefule Packages

- tidyverse: data management
- data.table: alternative to tidyverse (works well with large data)
- ggplot2: generating plots
- stargazer: regression tables
- feols: high-dimensional fixed effects (reghdfe in Stata)

Start working

We may first want to set the working directory

```
setwd("/home/projects")
```

Then load the data

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

Some useful dplyer syntax

The five key dplyr functions are:

- ‘filter‘: Filter rows based on their values
- ‘arrange‘: Sort rows based on their values
- ‘select‘: Select columns based on their names
- ‘mutate‘: Create new columns
- ‘summarise‘: Collapse multiple rows into a single column

Some examples

```
df <- filter(df, var1 >= 1 | var2 ==0)
df <- select(df, var1 & var2)
df <- mutate(df, log_var = log(var1))
```

The pipe operator

We can use the pipe operator %>% to improve readability.
Instead of

```
df <- group_by(df, var1)
df <- mutate(avg_var2 = mean(var2, na.rm=TRUE))
df <- ungroup(df)
```

We can write

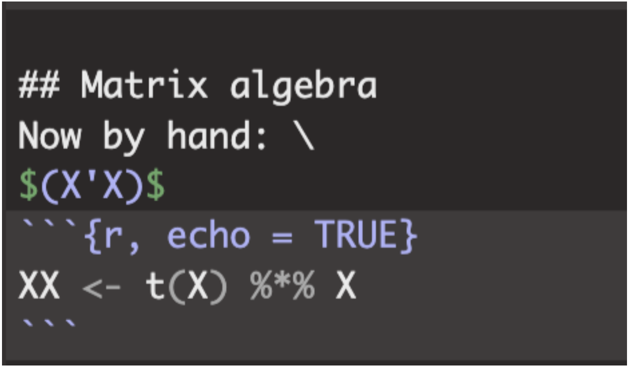
```
df <- df %>% group_by(var1) %>%
mutate(avg_var2, na.rm=TRUE) %>% ungroup()
```

See: https://www.rstudio.com/wp-content/uploads/2015/02/data-wrangling-cheatsheet.pdf

Markdown

Use can integrate your R code into a pdf file using Markdown

`New File -> R Markdown`



~~~
## Matrix algebra
Now by hand: \
$(X'X)$
```{r, echo = TRUE}
XX <- t(X) %*% X
```
~~~

Figure 1: Write code in Markdown