Introduction to “R”
Further topics

Arndt Leininger

@a_leininger
leininger@politik.uni-mainz.de

Loops

A simple loop

Loops can speed up our work by repeating tasks while changing one thing.

```
for(var in varlist) {
  mean(var, na.rm = TRUE)  # If therese is a missing
  # in the variable and you do
  # not specify na.rm = TRUE then mean() will return NA
}
```

Will loop through a list of variables, and calculate the mean of each of them and print the result to the console.

Loops

We can perform any commands we want inside the loop, including using more loops and if conditions. What would be the outcome of this loop?

```
for(i in 1:20) {
  if(i %% 2 == 0) {
    paste(i, 'is even') %>% print
    } else {
      paste(i, 'is odd') %>% print
    }
}
```

If-else

If-else

```
if(logical statement) {
  # evaluate this code if TRUE
} else {
  # evaluate this code if FALSE
}
```

If-else

```
for(i in 1:20) {
  if(i %% 2 == 0) {
    paste(i, 'is even') %>% print
    } else {
      paste(i, 'is odd') %>% print
    }
}
```

```
## [1] "1 is odd"
## [1] "2 is even"
## [1] "3 is odd"
## [1] "4 is even"
## [1] "5 is odd"
## [1] "6 is even"
## [1] "7 is odd"
## [1] "8 is even"
## [1] "9 is odd"
## [1] "10 is even"
## [1] "11 is odd"
## [1] "12 is even"
```

Hands-on I

Hands-on

Fizz-Buzz

Count up to 100, replacing any number divisible by three with the word “fizz”, and any number divisible by five with the word “buzz”, and any number divisible by both with ”fizz-buzz”..

Hands-on (solution)

Fizz-Buzz

Count up to 100, replacing any number divisible by three with the word “fizz”, and any number divisible by five with the word “buzz”, and any number divisible by both with ”fizz-buzz”.

```
for(i in 1:100) {
  ifelse(i %% 3 == 0 & i %% 5 == 0, 'fizz-buzz',
         ifelse(i %% 3 == 0, 'fizz',
                ifelse(i %% 5 == 0, 'buzz', i)
         )
  ) %>% print
}
```

Hands-on (solution)

Fizz-Buzz

Count up to 100, replacing any number divisible by three with the word “fizz”“, and any number divisible by five with the word”buzz“”, and any number divisible by both with “fizz-buzz”.

```
# Strictly speaking you do not need a loop for this
v <- 1:100

ifelse(v %% 3 == 0 & v %% 5 == 0, 'fizz-buzz',
       ifelse(v %% 3 == 0, 'fizz',
              ifelse(v %% 5 == 0, 'buzz', v)
       )
)
```

Functions

Defining your own functions

- It is straightforward to define a function in R since it is also just an object.

Functions

```
sdp <- function(x) sqrt(sum((x - mean(x))^2) / length(x))

x <- rnorm(n = 100, mean = 0, sd = 1)

sdp(x)
```

Defining your own functions

- You can of course define multiple argument and corresponding default values

```
divide <- function(value1, value2 = 2) value1 / value2

divide(2)
```

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

```
divide(9, 3)
```

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

```
divide(value2 = 2)
```

```
## Error in divide(value2 = 2): argument "value1" is missing, wi
```

Practice project

Practice project

Reading Data

Now we want to apply some of this to a real world example.

We've found an interesting data source on the web (link).

It's an excel sheet containing various crime related data in different London boroughs over different time periods.

There's an interesting panel dataset in there but we have to work to get it.

Practice project

Reading Data

We start by downloading the data onto our computer (`download.file()`) and use `read.xls()` to load the sheet "Fear of Crime-Borough".

```
URL <- paste0('https://files.datapress.com/london/dataset/',
              'metropolitan-police-service-recorded-crime-',
              'figures-and-associated-data/',
              '2015-12-23T15:58:16/MASTER_mps-figures.xls')
download.file(URL, 'data/crime.xls')  # Download the file
# and save in data folder
# with simpler file name

d <- read.xlsx('data/crime.xls',  # read the file from the
               # hard drive (inputting URL also works)
               sheetName = 'Fear of Crime-Borough',
               rowIndex = 3:31,  # read rows 3 to 31
               colIndex = 1:33,
               header = T)  # tell R that variable
```

Practice project

```r
# remove all % signs from the variables so that we are now able
d[, 2:ncol(d)] <- d[, 2:ncol(d)] %>%
  apply(., 2, str_replace, '%', '')

# make variable numeric
d[, 2:ncol(d)] <- d[, 2:ncol(d)] %>%
  apply(., 2, as.numeric)
```

Practice project

Reading Data

Sometimes data is input incorrectly - in this case we have 2 records for September 2008. Since we don't know which is correct, it's probably safer to remove them both.

```
# identify the duplicate month and remove all obs from that mont
d <- d[-which(d$Month.Year ==
                d$Month.Year[which(duplicated(d$Month.Year))]),

# or since we know that September 2008 is double
d <- d[which(d$Month.Year != '2008-09-01'), ]
# or
d <- d %>% filter(Month.Year != '2008-09-01')
```

Practice project

Transforming data

In a standard data.frame columns are variables and rows are observations. In R there are also just referred to as columns and rows. We don't always receive data like that.

How would you reformat this data? What variables do we have?

```
d[1:12, 1:4]
```

```
##    Month.Year Barking.and.Dagenham    Barnet    Bexley
## 1  2008-06-01           0.37403766 0.3384160 0.4919643
## 2  2009-03-01           0.48311216 0.3819179 0.2975082
## 3  2009-06-01           0.39054378 0.3538071 0.2786699
## 4  2009-09-01           0.26306191 0.2866238 0.2656135
## 5  2009-12-01           0.07289901 0.2330249 0.2630928
## 6  2010-03-01           0.04276600 0.1940772 0.2075963
## 7  2010-06-01           0.34929455 0.2513858 0.1108463
## 8  2010-09-01           0.35026151 0.2691175 0.2236333
## 9  2010-12-01           0.35423293 0.2511928 0.2799398
## 10 2011-03-01           0.37985760 0.2527754 0.3170738
```

Practice project

Transforming data

Our data is too wide: we can use the `gather()` function from the `tidyr` package to switch between 'wide' and 'long' formats. While we're at it, we can set names for the new variables we create.

```
d <- tidyr::gather(d, key = 'Borough', value = 'FoC',
                   Barking.and.Dagenham:Westminster)
```

Practice project

Transforming data

```
head(d)
```

```
##   Month.Year                 Borough        FoC
## 1 2008-06-01 Barking.and.Dagenham 0.37403766
## 2 2009-03-01 Barking.and.Dagenham 0.48311216
## 3 2009-06-01 Barking.and.Dagenham 0.39054378
## 4 2009-09-01 Barking.and.Dagenham 0.26306191
## 5 2009-12-01 Barking.and.Dagenham 0.07289901
## 6 2010-03-01 Barking.and.Dagenham 0.04276600
```

Practice project

Using a loop

Now we know how to read in one sheet, we need to do the same for the others. We could write the code we just used 7 times (many many lines of code) and change all the relevant parts...

Practice project

Reading Data

Instead, we can write 4 objects, for the 4 parts of our code that change (Sheet name, name of variable extracted from the sheet, cell range, variable name) and write a loop (just a few lines of code).

```
# Here we set up the variables we need
sheets <- c("Fear of Crime-Borough", "MOPAC Priority-Borough",
            "Officer Strength-Borough",
            "Sergeant Strength-Borough",
            "Special Strength-Borough", "PCSO Strength-Borough",
            "Staff Strength-Borough")  # the names of the sheets

varnames <- c('FoC', 'MOPAC', 'Officer', 'Sergeant',
              'Special', 'PCSO', 'Staff')
# the variable names we will use for the extracted data
```

Practice project

```
rowIndexes <- list(3:31, 3:58, 5:97, 4:36, 5:97, 5:97, 5:97)

colIndexes <- list(1:33, 1:34, 1:33, 1:33, 1:33, 1:33, 1:33)

n <- length(sheets)  # the number of sheets to read
```

Practice project

Using a loop

```
data <- list()  # list to hold data.frames after looping

for(i in 1:n) {
  tmp <- read.xlsx('../data/crime.xls',
                   # read the file from the hard drive (inputting UR
                   sheetName = sheets[i],
                   # Tell R which sheet to import
                   rowIndex = rowIndexes[[i]],
                   # which rows to read
                   colIndex = colIndexes[[i]],
                   # which cols to read
                   header = T)
  # tell R that variable names are in the first row
```

Practice project

```
names(tmp)[1] <- 'Month.Year'  # makes sure that first column is
    # remove all % signs from the variables so that we are now a
  tmp[, 2:ncol(tmp)] <- tmp[, 2:ncol(tmp)] %>%
    apply(., 2, str_replace, '%', '')
  # make variable numeric
  tmp[, 2:ncol(tmp)] <- tmp[, 2:ncol(tmp)] %>%
    apply(., 2, as.numeric)
    # remove duplicates
  if(which(tmp$Month.Year ==
           tmp$Month.Year[which(duplicated(tmp$Month.Year))]) %>
     length > 0) {
    tmp <- tmp[-which(tmp$Month.Year ==
                        tmp$Month.Year[which(duplicated(tmp$Mont
  }
```

Practice project

```
# turn data into long-format
  var <- varnames[i]
  tmp <- tidyr::gather(tmp, key = 'Borough',
                       value = var, -Month.Year)
  # save the data.frames in a list
  data[[i]] <- tmp
}
```

Practice project

Reading Data

Now we can `merge()` the data.frames we extracted from the spreadsheet.

```
#First let's do a quick check whether our data seem to be in the
for(i in 1:7) data[[i]] %>% head %>% print

# Now let's merge all the data.frames using a loop

tmp <- data[[1]]
for(i in 2:n) {
  tmp <- merge(tmp, data[[i]], by = c('Borough', 'Month.Year'))
}

tmp %>% head

dat <- tmp
```

Practice Project

```r
summary(dat)  # we note a coding error in FoC, sometimes it is d
# sometimes as fractions

dat$FoC <- ifelse(dat$FoC < 1, dat$FoC * 100, dat$FoC)

write.csv(dat, 'data/crime_data.csv', row.names = F)
```

Regression

- regression models are functions that are fed an equation and data
- equation: dv ~ iv

```
lm(FoC ~ Officer, dat)
```

```
##
## Call:
## lm(formula = FoC ~ Officer, data = dat)
##
## Coefficients:
## (Intercept)      Officer
##    25.22792      0.01332
```

Presenting results with R

A Table

```
m1 <- lm(FoC ~ Officer, dat)
m2 <- lm(FoC ~ Officer + Sergeant, dat)

library(stargazer)
```

```
##
## Please cite as:

##  Hlavac, Marek (2018). stargazer: Well-Formatted Regression a

##  R package version 5.2.1. https://CRAN.R-project.org/package=
```

```
stargazer(list(m1, m2), header = F, float = F, font.size = 'tiny
          single.row = T)
```

	Dependent variable:	
	FoC	
	(1)	(2)
Officer	0.013*** (0.003)	0.027*** (0.004)