Live Coding Session

SCR Week 6

Manipulation of Character Variables / Objects.

This live coding script will be filled out during the lecture (and will probably be uploaded after the lecture).

paste()

Can we reproduce `Sys.time()` ourselves with the use of `paste()`?

```
current_time <- as.character(Sys.time())
year <- 2019
month <- 10
day <- 18
hours <-
minutes <-
seconds <-
YYYMMDD <-
hhmmss <-
TZ <- "CEST"
paste(year,month, day,sep = "-")
# paste these together
```

The output of `paste()` has most of (if not all) the time a character `mode`.

strsplit()

```
current_time <- as.character(Sys.time())
out <- strsplit(x = current_time, split = "-")
out2 <- strsplit(x = out[[1]], split = " ")
```

When regular expressions could bite you. . . . without knowing: beware of the `fixed = TRUE` argument!

```
fname <- "filename.txt"
nchar(fname)
strsplit(fname, ".")[[1]]
?strsplit # check FIXED = TRUE argument!
strsplit(fname, ".", fixed = TRUE)[[1]]
```

Be aware of vector recyling!

```
char_dates_leap_seconds <- as.character(.leap.seconds)
# Don't go like...
# strsplit(x = char_dates_leap_seconds, split = c(" ", ":", "-"))
# smth that we see as bonus, but that is not expected of you:
strsplit(char_dates_leap_seconds, split = "[-/[:space:]/:]") # using regex
```

gsub()

The example with gsub. . .

```
#gsub(pattern = ".", replacement = " ", x = char_dates_leap_seconds)
#gsub(pattern = "-", replacement = " ", x = char_dates_leap_seconds)
```

```
# smth that we see as bonus, but that is NOT expected of you:
gsub(pattern = "[-/:]", replacement = " ", x = char_dates_leap_seconds)
```

Create a data.frame with the columns Year, Month, Day, Hour, etc.

```
x = gsub(pattern = "[-/:]", replacement = " ", x = char_dates_leap_seconds)
x = strsplit(x,' ')
# use do.call
matrix = do.call('rbind',x)
df = as.data.frame(matrix)
names(df) <- c('y','m','d','h','m','s')
```

grep()

```
my_txt <- c(
  "This is string to show how grep can work",
  "we will ask grep to select the indices with how in it",
  "It does not select me as index 3",
  "However, it does count 'show' too,",
  "but doesn't count 'However'"
)
grep(pattern = "how", x = my_txt)
my_txt[grep(pattern = "how", x = my_txt)]
```

nchar()

```
nchar(my_txt)
# Could you program the nchar function yourself?
list_out <- strsplit(my_txt, split = '', fixed = TRUE)
sapply(list_out, length)
```

1. A Plenary Exercise on Share of Women Researchers by sectors of performance.

This is a (slightly modified) task that once was part of an assignment for grade (SCR 2016). The goal here is to download and clean Eurostat Official Statistics data on the EU women' employment such that it can be used for further data analysis and visualization. Thus, the original data is not very tidy, at the end of this task it should have the following structure:

```r
load("0_data/tidy_dat.RData")
str(tidy_dat)
```

For a descirption and its format see

> http://ec.europa.eu/eurostat/tgm/refreshTableAction.do?tab=table&plugin=1&pcode=tsc00005&language=en
>
> http://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=data/tsc00005.tsv.gz&unzip=true

We have created a tinyurl address that links to Eurostat data: http://tinyurl.com/hclx46v

You will need at least one function, i.e. `strsplit()`, for this assignment that uses `regular expressions`. Note, however, you do not have to explicitly use yourself any regular expressions. We have decomposed the whole task into sub-tasks. In case you come up with a different way to tackle the whole problem, you are welcome to do so! Then, just provide a cleary code written answer such that we can easily follow each of the steps you have chosen to take.

1.1

Download the data from the website into `R`.

Answer:

```r
data<-read.delim(url, header= TRUE, sep = '\t')
```

1.2

Focus on the first column of the data which is called `unit.sectperf.geo.time`. Each one looks something like `PC_HC,BES,AT` where each abbreviation is separated by a column. We want the sector which in this case is `BES` and the geo location which in this case is `AT`.

Use `strsplit` to extract these values for each element in the `unit.sectperf.geo.time` column of the data, and create two new variables called `sector` and `geo`. (hint, the data is currently a `factor` and `strsplit()` work on `characters`)

Answer:

```r
t_col <- as.character(data$unit.sectperf.geo.time)
t <- strsplit(t_col, split=',')
mat <- do.call(rbind, t)
sector <- mat[,2]
geo <- mat[,3]
```

1.3

Program your own function that you can apply to one of the remaining columns of the data. You may want to use the hint on how to create your function:

Hint:

```r
GiveNameThatTellsWhatThisFunctionNeedsToDo <- function(column){
  # function takes a single column as argument,
  # and returns a list with two elements.
  # The percentages of the column are stored in
  # the first element and the annotated letters
  # are strored in the second element
  return()
}
```

Please bear in mind that this can be a frustrating task. As you will most probably experience in your later career: cleaning data is very frustrating. . . .

Answer:

```
SplitCol <- function(col){
  x <- as.character(col)
  n <- length(x)
  results <- list(percentzge = numeric(n), comment = character(n))
  for (i in 1:n)
      {
      t <- strsplit(x[i], split = ' ')
      results[[1]][i]= as.numeric(t[[1]][1])
      #if (length(t[[1]]) == 2) {print(t)}
      results[[2]][i]= t[[1]][2]
      }
  return(results)
}
results <- SplitCol(data$X2007)
```

```
CleanYearPercentageColumn <- function(column){
column <- as.character(column)
col_splitted <- strsplit(column, split = " ")
prct_col <- sapply(col_splitted, "[", 1)
ter prct_col <- suppressWarnings(as.numeric(prct_col))
column annt_col <- sapply(col_splitted, "[", 2)
return(list(prct_col, annt_col)) }
```

To extract the first element from each of these character vectors:
sapply(fileNameSplit, `[`, 1) # same as
sapply(fileNameSplit, function(x) x[1])

1.4

Apply your function of 1.3 in an explicit or implicit loop at each column. Combine the results into two data frames, one that holds the `numbers`, and one that holds the `letters`.

Answer: data[2:ncol(mydat)]

```
results = lapply(data,SplitCol)
a <- do.call(rbind, results[-1])
df_number <- a[,1]
df_letter <- a[,2]
```

```
percentages <- as.data.frame(df_letter)
annotation <- as.data.frame(df_number)
```

1.5

Finally, combine the `sector`, `geo`, `percentages` and `annotation` into one data frame and make sure everything has the correct class, column names and that the variables match up with the output from above. Show that your cleaned data set is the same as `tidy_dat`.

names(annotation) <- gsub("X", "C", names(annotation))

sector <- factor(sector)

Answer:

```
tidy_dat <- data.frame('sector' = sector,
                       'geo' = geo,
                       'percentages' = percentages,
                       'annotation' = annotation)
```

```
tidy_dat
```

load("0_data/tidy_dat.RData")
all.equal(my_clean_dat, tidy_dat)