You must have an idea of what scripts are from the shell lesson.
Like: grep, sort, find, wc
Script:

- Brings those individual commands together
- Bigger more complex task
- Recall
- Comment

Typical script

- Read
- Analyze
- Plot
- Save

W: who are programmers, what do you program?

Challenge – identify variables and operations needed

First script in RStudio

- Layout
- Identify where to write scripts
- **getwd()**
- New Script – myScript.R
- Ctrl-Enter
- TITLE: computes the average GDP for Albania

```
# This script computes the average GDP per capita for Albania using the gapminder data

# location of the data
fileName <- 'data/gapminderData.csv'   10 different places

# read the data file
gapminder <- read.csv(fileName)
```

meaning of the dot

show gapminder$country and so on 13:24

```
# select the rows where the country is Albania and store it in albaniaData
albaniaData <- gapminder[gapminder$country == 'Albania',]

# select the column containing the GDP per capita from the Albania data
albaniaGdp <- albaniaData$gdpPercap  did not print GOOD or BAD?

# compute the average GDP per capita value
albaniaAverageGdp <- mean(albaniaGdp)

# print a message with the result of our computation
paste('The average GDP per capita of Albania is', albaniaAverageGdp)
```

Remind advantages: well comment record and change a value and re-run it

Advantages of using the RStudio editor

- Syntax highlighting
- Automatic code indentation and bracket pairing
- Variable name completion with [tab] (like bash)
- Easier debugging

source(filename)
source button
Session → set wd

autoprinting is off in scripts – saves clutter

Challenge: Write a script to add two numbers

Functions:

- canned scripts – complicated or lengthy
- built-in base. Library() loaded packages.
- Mean() in and out
- print() in, but no out
- ls() no in, only out
- Gather a sequence of operations and preserve for ongoing use
  - memorable name
  - remembering operations
  - defined inputs and outputs
  - rich connections

New script: functions-lesson.R
Let's define F to DEG C
'then

```
fahr_to_kelvin <- function(temp) {
  kelvin <- ((temp - 32) * (5/9)) + 273.15

    kelvin

}
```

Explain the various parts.
- more than 1 args
Boiling water 212F, room 72F
CHECK understanding of functions
Compositing functions (skippable)

Challenge: Explain the command: (can be omitted for time)
print(paste('absolute zero in Celsius:', kelvin_to_celsius(0)))
Use paste to write a wrapper

R Environments and scope
Go back to **myScript.R**
Edit the script to use a variable for country name, change all variable names
Convert to function with 2 arguments (which are needed, final outcome)
Call the fn (stick with one country only for this time)
Remove one of the arguments.
Call the fn
    Missing value – looked up in the functions frame, then in the frame where it was defined

Default arguments
    start and end year
Call the fn

List of countries
sapply() or for()

Don't write: show anonymous function
gapminder.org

barplot
order

Command line: send country name from the command line

LUG PLUG

```
# This script will compute the average GDP per capita for Albania across the years

# read in the data
fileName <- 'data/gapminderData.csv'
gapminder <- read.csv(fileName)

countryName <- 'United States'

# extract the gdp column for "country" and find its average GDP per capita
computeAvgGdp <- function(countryName, startYear = 1990, endYear = 2000) {
  data <- gapminder[gapminder$country==countryName, ]
  dataYears <- data[data$year>=startYear & data$year<=endYear, ]
  gdpColumn <- dataYears[ , "gdpPercap"]
  avgGdp <- mean(gdpColumn)

  return(avgGdp)
}

begin <- 1950
end <- 1960
avgGdp <- computeAvgGdp(countryName)
```

```
# print the results
print(paste("The average GDP per cap of", countryName, "is", avgGdp))

# anonymous function
countries <- c('United States', 'United Kingdom', 'Ethiopia')
countriesAvgGdp50s <- sapply(countries, function(countryName) {
  computeAvgGdp(countryName, begin, end)
})
print(countriesAvgGdp50s)
begin <- 1990
end <- 2000
countriesAvgGdp90s <- sapply(countries, function(countryName) {
  computeAvgGdp(countryName, begin, end)
})
print(countriesAvgGdp90s)

barplot(countriesAvgGdp50s)
barplot(countriesAvgGdp90s)
```

```
# Write functions to convert temperature

# a function to convert temperature in Fahr. to Celc.
convertFarhToCelc <- function(tempFahr) {
  # the body of the function goes here
  tempCelc <- (tempFahr - 32) * (5/9)

  return(tempCelc)
}

waterBPFahr <- 212
waterBPCelc <- convertFarhToCelc(waterBPFahr)

# print(tempCelc)

print(paste("The boiling point of water is", waterBPFahr, "F or", waterBPCelc, "C"))

print(paste("The room temperature is", 72, "F or", convertFarhToCelc(72), "C"))

convertFahrToKelvins <- function(tempFahr) {
  tempKelvins <- convertFarhToCelc(tempFahr) + 273.15
}

waterBPKelvin <- convertFahrToKelvins(waterBPFahr)
print(paste("The boiling point of water is", waterBPFahr, "F, or", waterBPCelc, "C, or",
waterBPKelvin, "K"))
```