Week 8 - 6103 Handout

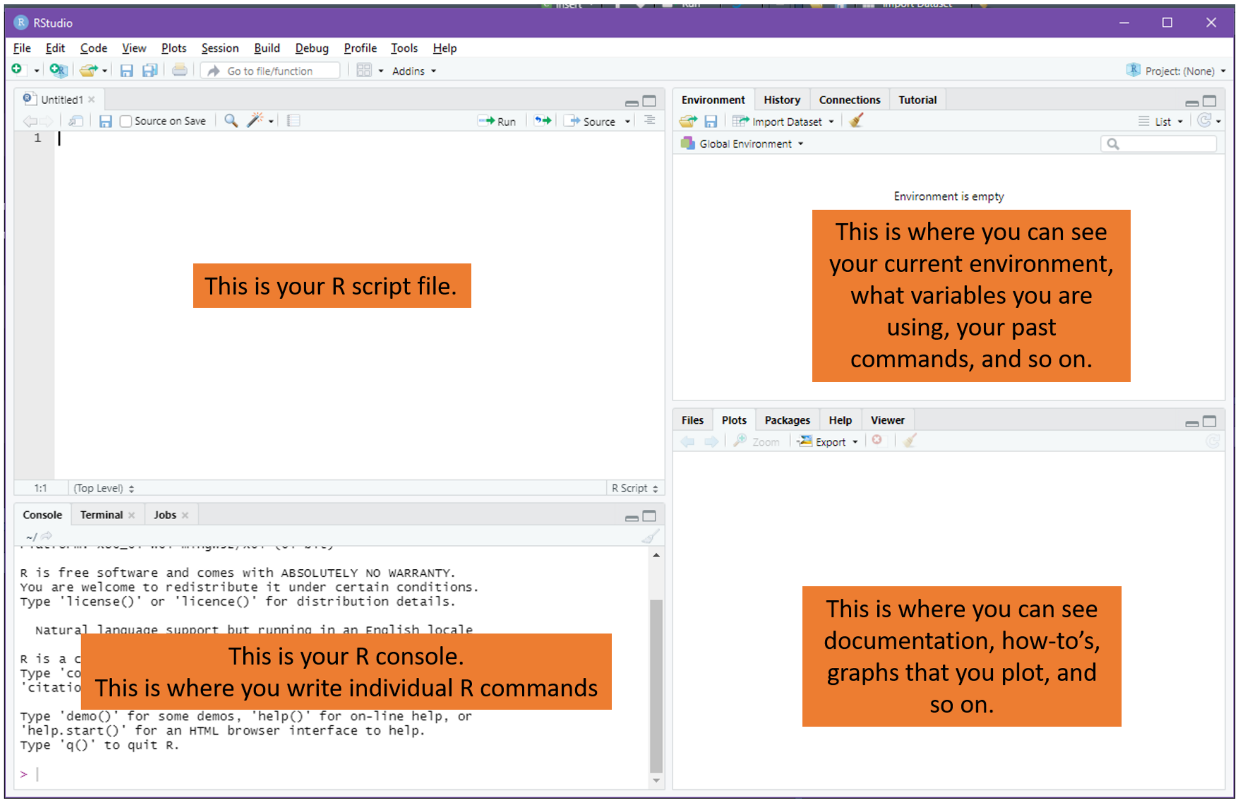
Subhayan Mukerjee

10/3/2021

Welcome to R!

This document is an RMarkdown document. It allows you to write text as well as code (and their outputs) in a single convenient file, and is often used for tutorial purposes. Most of what you will be doing in R, however, will be done in a R script file. An R Script file is essentially a text file, with the .R extension. You can create an R Script file, by opening RStudio and then clicking on File -> New File -> R Script. Alternatively, on Windows you can simple open RStudio and press Ctrl + Shift + N.

When you open RStudio you will see there are four quadrants as follows (Figure 1)



This is your R script file.

This is where you can see your current environment, what variables you are using, your past commands, and so on.

This is where you can see documentation, how-to's, graphs that you plot, and so on.

This is your R console.
This is where you write individual R commands

```
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a c
Type 'co
'citatio

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Figure 1: RStudio

Let's type our first R command. Take your cursor to the console (bottom left) and type the following (what you see in the shaded section). This command simply tells R to display the text "Hello World!". You should see the output that follows.

```
print("Hello World!")
```

```
## [1] "Hello World!"
```

In this document, like in all RMarkdown documents, the R commands appear on a shaded background. And the output from the R command appears right after the '##'. The [1] in the output refers to the first line of output.

You can do basic arithmetic in the console as well. For example

```
1+1
```

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

```
5-2
```

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

```
22*45/56
```

```
## [1] 17.67857
```

```
sqrt(98)
```

```
## [1] 9.899495
```

```
5^3
```

```
## [1] 125
```

However, as you will notice, you can only write single line commands in the console. As most analysis that you will do, will require multiple commands, it is wiser to use the R Script (top left)

Type out all the commands that you just typed in the console, in the R script, select the entire text and click on "Run" (Figure 2). The output will appear in the console.

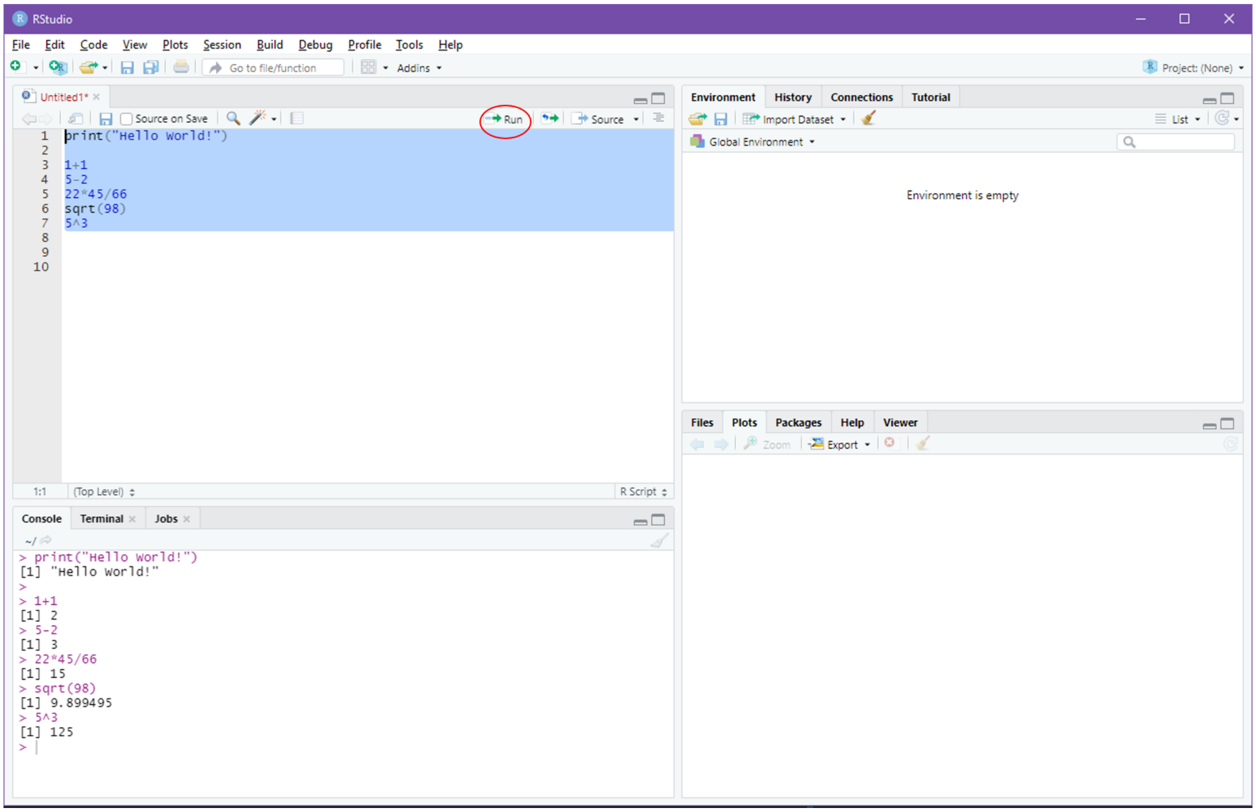
You can also select each line individually and click on "Run" to run just that line.

From now on, when you encounter an R command in this file, type it out in the R script and run it from there, instead of from the console.

Variables

Now, as you know we can't keep dealing with numbers. We will need to store them in variables. These variables are not conceptually the same as social science variables we have discussed so far, but they can be used as a proxy. You can use R variables to store actual variables that you have collected for your research.

Creating variables in R is easy. For example, the next command defines a variable called 'a' and assigns to it, the value of 5. You can use both, the "=" as well as the "<-" symbols to assign values to a variable.



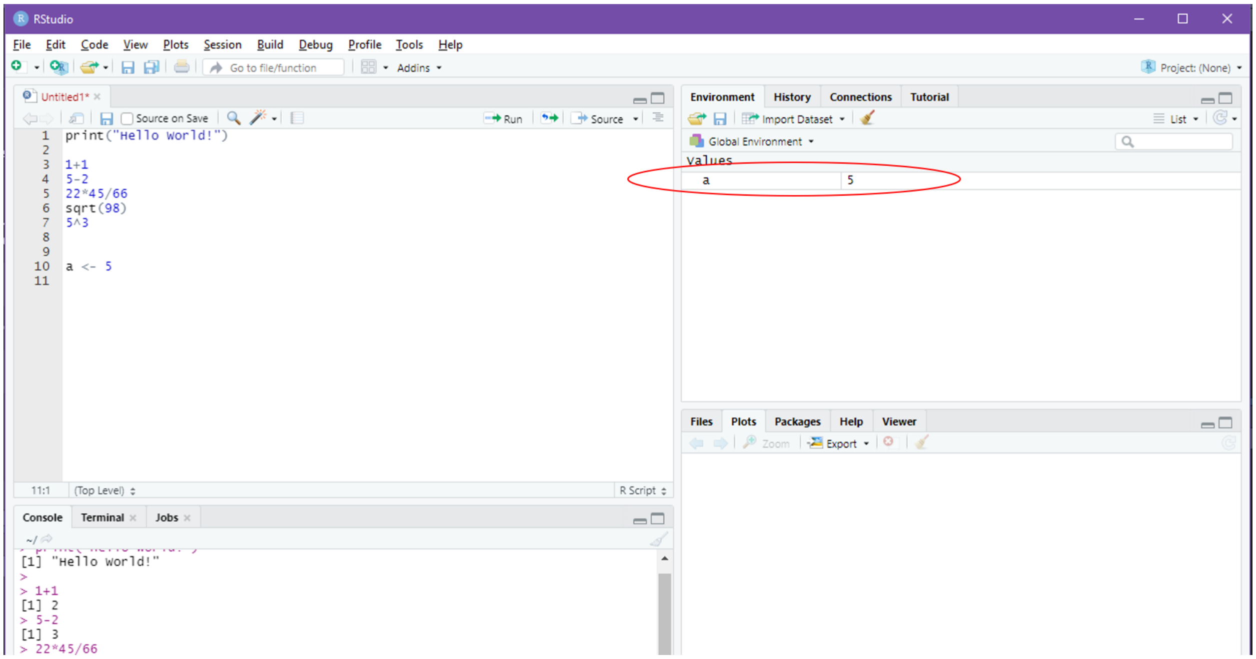
```r
print("Hello World!")

1+1
5-2
22*45/66
sqrt(98)
5^3
```

```
> print("Hello World!")
[1] "Hello World!"
>
> 1+1
[1] 2
> 5-2
[1] 3
> 22*45/66
[1] 15
> sqrt(98)
[1] 9.899495
> 5^3
[1] 125
>
```

Figure 2: R script in R Studio

```r
a <- 5
```

Once a variable has been assigned a value, it will appear in your environment panel of RStudio (top right) (Figure 3).



```r
print("Hello World!")

1+1
5-2
22*45/66
sqrt(98)
5^3


a <- 5

```

```
Values
  a                     5
```

```
[1] "Hello World!"
>
> 1+1
[1] 2
> 5-2
[1] 3
> 22*45/66
```

Figure 3: R script in R Studio

you can see what the value is by doing:

```r
print(a)
```

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

Variables are just like numbers. You can do arithmetic on them.

```r
a <- 6
b <- 4
c <- 5

d <- (a + b) / c

print(d)
```

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

Vectors

Vectors lets you combine multiple values into a single variable. Say, for instance, we have the heights of 5 individuals (in cm.). We could have a single variable for each of them (`h1`, `h2`, `h3`, `h4`, `h5`) but that can be very confusing to keep track of. Instead you can define a vector called heights as follows:

```
heights <- c(140, 176, 154, 143, 166, 180, 148, 178)

print(heights)
```

```
## [1] 140 176 154 143 166 180 148 178
```

The `c` in the above command stands for "combine". It instructs R to combine the five values into one vector.

Remember that vectors are also "variables". Anything that can take a value is a variable. Vectors are just variables that are made up other variables.

We can access individual elements of a vector using square brackets. The following two commands gives you the first and third height respectively.

```
print(heights[1])
```

```
## [1] 140
```

```
print(heights[3])
```

```
## [1] 154
```

The number within the square brackets is called the index. These individual elements are similar to the variables (`a`, `b`, `c`) we used earlier. Indeed, you can combine existing variables into a vector as well like this:

```
random_vector <- c(a, b, c, d)

print(random_vector)
```

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

Character variables

Similar to numeric variables you can also define textual variables. These are called `character` variables. You can have vectors of these variables as well.

```
text_var <- "abcdef"
text_var2 <- "xyz"

print(text_var)
```

```
## [1] "abcdef"
```

```
print(text_var2)
```

```
## [1] "xyz"
```

```r
persons <- c("Michael", "Alice", "Sanjay", "Claudia", "Bob", "Jack", "Mary", "Jill")

print(persons[4])
```

```
## [1] "Claudia"
```

```r
print(persons[2])
```

```
## [1] "Alice"
```

What happens when you try printing the 10th person's name (that clearly doesn't exist?)

```r
print(persons[10])
```

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

Logical variables

`Logical` variables can either be `TRUE` or `FALSE`. They are used to know whether a certain variable is equal to (`==` NOTE the double equals), less than (`<`), or greater (`>`) than a certain value. These can also be combined as "less than or equal to" ('<=') or "greater than or equal to" ('>=') Let's look at an example:

```r
print(a)
```

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

```r
print(b)
```

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

```r
print(c)
```

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

```r
print(a == 6) # is a equal to 6?
```

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

```r
print(b > 3) # is b greater than 3?
```

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

```r
print(c <= 5) # is c less than or equal to 5?
```

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

What you see on the right of each command, after the # is called a comment. They are used to describe what that command does, and is ignored by R. They are largely written to make the script understandable to someone reading the file.

Another useful operator is the != which means "not equal to". It can be used as follows

```r
print((a+b) != 10) # is a+b not equal to 10?
```

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

Logical variables are useful in many cases. Let's look at the heights vector again.

```r
print(heights)
```

```
## [1] 140 176 154 143 166 180 148 178
```

We want to know which of the heights are greater than 160?

```r
print(heights > 160)
```

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

This returns a vector of logicals. If you want to know what those heights values are (corresponding to TRUE), you can simply use the output of the above command as the index for the vector. The way R works, it only shows those values which correspond to TRUE.

```r
print(heights[heights > 160])
```

```
## [1] 176 166 180 178
```

This command essentially prints all the heights that are above 160. What does the next command do?

```r
print(heights[heights == 170])
```

```
## numeric(0)
```

Dataframe

A dataframe is a table, comprising of rows and columns of data. They are the most widely used data structures in R. Let's put the two vectors we have into a dataframe. To do that, we will first use the cbind or the column bind command to put the two vectors "next to each other" as two columns, and then use the data.frame command to create a dataframe.

Finally, we will assign the dataframe to a variable called df (the name can be anything, obviously).

This gives us a dataframe with two columns and 8 rows. Each row corresponds to one person.

```
df <- data.frame(cbind(persons, heights))
print(df)
```

```
##   persons heights
## 1 Michael     140
## 2   Alice     176
## 3  Sanjay     154
## 4 Claudia     143
## 5     Bob     166
## 6    Jack     180
## 7    Mary     148
## 8    Jill     178
```

We can access the individual columns using the `$` command:

```
print(df$persons)
```

```
## [1] "Michael" "Alice"   "Sanjay"  "Claudia" "Bob"     "Jack"    "Mary"
## [8] "Jill"
```

```
print(df$heights)
```

```
## [1] "140" "176" "154" "143" "166" "180" "148" "178"
```

What each of these two commands gives you are the original vectors that you used to make the dataframe in the first place.

However, if you notice closely, you will see that `df$heights` shows you the height values but within double quotes. This is because when you used `cbind` it converted both vectors into a common format. In other words, the heights were converted from numeric to character. If you're not sure what I mean by this use the `class` command. The `class` command tells you the type of the data stored in that variable. So, run the following two commands:

```
class(heights)
```

```
## [1] "numeric"
```

```
class(df$heights)
```

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

`heights` is `numeric` but `df$heights` is a `character`.

So we will need to reconvert them to numeric. So use the `as.numeric` command as shown below, and re-assign the re-converted heights to the same column.

```
df$heights <- as.numeric(df$heights)
```

Dataframes are very powerful as they allow you to do a variety of analysis. Crucial is knowing how to index dataframes, for retrieving the information you need.

Just like vectors are indexed using square brackets, so are dataframes. But unlike vectors they need two numbers, one for the row, and one for the column.

For example

```
print(df[2,1])
```

```
## [1] "Alice"
```

```
print(df[6,2])
```

```
## [1] 180
```

Again, what happens when you index incorrectly?

```
print(df[3,4])
```

```
## NULL
```

Now these indices can be logicals. This lets us ask questions like, "who are the persons with heights greater than 160?"

Let's answer this step-by-step.

First, we have

```
print(df$heights > 160)
```

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

Next, we use this as the first index for our dataframe (we want the rows where heights > 160). For the second index, we keep it blank, so that we can see all (both) columns.

```
print(df[df$heights > 160,])
```

```
##   persons heights
## 2   Alice     176
## 5     Bob     166
## 6    Jack     180
## 8    Jill     178
```

Remember that the previous output is a dataframe itself. So, if we just want the heights, we can do

```
print(df[df$heights > 160,]$heights)
```

```
## [1] 176 166 180 178
```

You can also sort the dataframe in ascending or descending order of height using the `order` command:

```r
print(df[order(heights),]) # increasing (ascending) order
```

```
##   persons heights
## 1 Michael     140
## 4 Claudia     143
## 7    Mary     148
## 3  Sanjay     154
## 5     Bob     166
## 2   Alice     176
## 8    Jill     178
## 6    Jack     180
```

```r
print(df[order(-heights),]) # decreasing (descending) order
```

```
##   persons heights
## 6    Jack     180
## 8    Jill     178
## 2   Alice     176
## 5     Bob     166
## 3  Sanjay     154
## 7    Mary     148
## 4 Claudia     143
## 1 Michael     140
```

Descriptive Statistics

Now let's look at some statistics of the heights

```r
print(mean(df$heights)) # this gives the mean height
```

```
## [1] 160.625
```

```r
print(median(df$heights)) # this gives the median height
```

```
## [1] 160
```

```r
print(sd(df$heights)) # this gives the standard deviation of the heights
```

```
## [1] 16.39632
```

Other commands you can use are `min` (for minimum) or `max` for maximum. You can see a statistical summary of a variable using the `summary` command:

```r
summary(df$heights)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   140.0   146.8   160.0   160.6   176.5   180.0 
```

Let's add another column to the dataframe, let's call it gender.

```
df$gender <- c("M", "F", "M", "F", "M", "M", "F", "F")
print(df)
```

```
##   persons heights gender
## 1 Michael     140      M
## 2   Alice     176      F
## 3  Sanjay     154      M
## 4 Claudia     143      F
## 5     Bob     166      M
## 6    Jack     180      M
## 7    Mary     148      F
## 8    Jill     178      F
```

Let's now find the average heights of Males

Step 1: select only those rows where gender == "M" (Remember the double equals "=="!)

```
print(df[df$gender == "M",])
```

```
##   persons heights gender
## 1 Michael     140      M
## 3  Sanjay     154      M
## 5     Bob     166      M
## 6    Jack     180      M
```

Step 2: get their heights

```
print(df[df$gender == "M",]$heights)
```

```
## [1] 140 154 166 180
```

Step 3: Calculate their mean

```
print(mean(df[df$gender == "M",]$heights))
```

```
## [1] 160
```

Repeat for females

```
print(mean(df[df$gender == "F",]$heights))
```

```
## [1] 161.25
```

You can inspect a dataframe anytime by using the `str` or structure command.

```
str(df)
```

```
## 'data.frame':    8 obs. of  3 variables:
##  $ persons: chr  "Michael" "Alice" "Sanjay" "Claudia" ...
##  $ heights: num  140 176 154 143 166 180 148 178
##  $ gender : chr  "M" "F" "M" "F" ...
```

Now let's save the data that we generated in a CSV (comma separated values) file

```
write.csv(df, "heights.csv")
```

Where has the file been saved? In your current working directory. Type the command `getwd()` to locate it.

Let's work on an actual dataset!

Download the `gapminder.csv` file from luminus. Let's first read in the file into R.

```
gapminder_df <- read.csv("data/gapminder.csv") # replace this path with where you downloaded the file o
```

We want to look at the data, but this dataset has thousands of rows, so we don't want to see all of them. If you do `print(gapminder_df)` R will only show you the first 1000 rows in the console. You can try that, but it's not really very helpful. Let's look at the structure instead:

```
str(gapminder_df)
```

```
## 'data.frame':    1704 obs. of  6 variables:
##  $ country  : chr  "Afghanistan" "Afghanistan" "Afghanistan" "Afghanistan" ...
##  $ continent: chr  "Asia" "Asia" "Asia" "Asia" ...
##  $ year     : int  1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 ...
##  $ lifeExp  : num  28.8 30.3 32 34 36.1 ...
##  $ pop      : int  8425333 9240934 10267083 11537966 13079460 14880372 12881816 13867957 16317921 222
##  $ gdpPercap: num  779 821 853 836 740 ...
```

So we have 6 columns: country, continent, year, life expectancy, population, GDP per capita.

Look at their types. Some are `chr` meaning they are characters. Others are `int` meaning they are integers, or `numeric` meaning they can also be fractions. For all practical purposes in R, integers and fractions can be used interchangeably, unlike in many languages.

Another option is to use the `head` or `tail` command to look at just the top few or bottom few rows of a datset.

```
head(gapminder_df)
```

```
##       country continent year lifeExp      pop gdpPercap
## 1 Afghanistan      Asia 1952  28.801  8425333  779.4453
## 2 Afghanistan      Asia 1957  30.332  9240934  820.8530
## 3 Afghanistan      Asia 1962  31.997 10267083  853.1007
## 4 Afghanistan      Asia 1967  34.020 11537966  836.1971
## 5 Afghanistan      Asia 1972  36.088 13079460  739.9811
## 6 Afghanistan      Asia 1977  38.438 14880372  786.1134
```

```
tail(gapminder_df)
```

```
##       country continent year lifeExp      pop gdpPercap
## 1699 Zimbabwe    Africa 1982  60.363  7636524  788.8550
## 1700 Zimbabwe    Africa 1987  62.351  9216418  706.1573
## 1701 Zimbabwe    Africa 1992  60.377 10704340  693.4208
## 1702 Zimbabwe    Africa 1997  46.809 11404948  792.4500
## 1703 Zimbabwe    Africa 2002  39.989 11926563  672.0386
## 1704 Zimbabwe    Africa 2007  43.487 12311143  469.7093
```

Yet another option to “see” the data is the following:

```
View(gapminder_df)
```

The output isn’t shown in the console but is instead shown in a separate tab in RStudio. Helpful, right?

Let’s look at the data for a single year only, viz. 2007.

```
gapminder_2007df <- gapminder_df[gapminder_df$year == 2007,]
```

Let’s look at this much more manageable dataset

```
print(gapminder_2007df)
```

```
##                       country continent year lifeExp        pop  gdpPercap
## 12                Afghanistan      Asia 2007  43.828   31889923   974.5803
## 24                    Albania    Europe 2007  76.423    3600523  5937.0295
## 36                    Algeria    Africa 2007  72.301   33333216  6223.3675
## 48                     Angola    Africa 2007  42.731   12420476  4797.2313
## 60                  Argentina  Americas 2007  75.320   40301927 12779.3796
## 72                  Australia   Oceania 2007  81.235   20434176 34435.3674
## 84                    Austria    Europe 2007  79.829    8199783 36126.4927
## 96                    Bahrain      Asia 2007  75.635     708573 29796.0483
## 108                Bangladesh      Asia 2007  64.062  150448339  1391.2538
## 120                   Belgium    Europe 2007  79.441   10392226 33692.6051
## 132                     Benin    Africa 2007  56.728    8078314  1441.2849
## 144                   Bolivia  Americas 2007  65.554    9119152  3822.1371
## 156    Bosnia and Herzegovina    Europe 2007  74.852    4552198  7446.2988
## 168                  Botswana    Africa 2007  50.728    1639131 12569.8518
## 180                    Brazil  Americas 2007  72.390  190010647  9065.8008
## 192                  Bulgaria    Europe 2007  73.005    7322858 10680.7928
## 204              Burkina Faso    Africa 2007  52.295   14326203  1217.0330
## 216                   Burundi    Africa 2007  49.580    8390505   430.0707
## 228                  Cambodia      Asia 2007  59.723   14131858  1713.7787
## 240                  Cameroon    Africa 2007  50.430   17696293  2042.0952
## 252                    Canada  Americas 2007  80.653   33390141 36319.2350
## 264  Central African Republic    Africa 2007  44.741    4369038   706.0165
## 276                      Chad    Africa 2007  50.651   10238807  1704.0637
## 288                     Chile  Americas 2007  78.553   16284741 13171.6388
## 300                     China      Asia 2007  72.961 1318683096  4959.1149
## 312                  Colombia  Americas 2007  72.889   44227550  7006.5804
## 324                   Comoros    Africa 2007  65.152     710960   986.1479
## 336          Congo, Dem. Rep.    Africa 2007  46.462   64606759   277.5519
## 348               Congo, Rep.    Africa 2007  55.322    3800610  3632.5578
## 360                Costa Rica  Americas 2007  78.782    4133884  9645.0614
## 372             Cote d’Ivoire    Africa 2007  48.328   18013409  1544.7501
## 384                   Croatia    Europe 2007  75.748    4493312 14619.2227
## 396                      Cuba  Americas 2007  78.273   11416987  8948.1029
## 408            Czech Republic    Europe 2007  76.486   10228744 22833.3085
## 420                   Denmark    Europe 2007  78.332    5468120 35278.4187
## 432                  Djibouti    Africa 2007  54.791     496374  2082.4816
## 444        Dominican Republic  Americas 2007  72.235    9319622  6025.3748
## 456                   Ecuador  Americas 2007  74.994   13755680  6873.2623
## 468                     Egypt    Africa 2007  71.338   80264543  5581.1810
## 480               El Salvador  Americas 2007  71.878    6939688  5728.3535
## 492         Equatorial Guinea    Africa 2007  51.579     551201 12154.0897
## 504                   Eritrea    Africa 2007  58.040    4906585   641.3695
## 516                  Ethiopia    Africa 2007  52.947   76511887   690.8056
## 528                   Finland    Europe 2007  79.313    5238460 33207.0844
## 540                    France    Europe 2007  80.657   61083916 30470.0167
## 552                     Gabon    Africa 2007  56.735    1454867 13206.4845
## 564                    Gambia    Africa 2007  59.448    1688359   752.7497
## 576                   Germany    Europe 2007  79.406   82400996 32170.3744
## 588                     Ghana    Africa 2007  60.022   22873338  1327.6089
## 600                    Greece    Europe 2007  79.483   10706290 27538.4119
## 612                 Guatemala  Americas 2007  70.259   12572928  5186.0500
## 624                    Guinea    Africa 2007  56.007    9947814   942.6542
## 636             Guinea-Bissau    Africa 2007  46.388    1472041   579.2317
## 648                     Haiti  Americas 2007  60.916    8502814  1201.6372
## 660                  Honduras  Americas 2007  70.198    7483763  3548.3308
## 672          Hong Kong, China      Asia 2007  82.208    6980412 39724.9787
## 684                   Hungary    Europe 2007  73.338    9956108 18008.9444
## 696                   Iceland    Europe 2007  81.757     301931 36180.7892
## 708                     India      Asia 2007  64.698 1110396331  2452.2104
## 720                 Indonesia      Asia 2007  70.650  223547000  3540.6516
## 732                      Iran      Asia 2007  70.964   69453570 11605.7145
## 744                      Iraq      Asia 2007  59.545   27499638  4471.0619
## 756                   Ireland    Europe 2007  78.885    4109086 40675.9964
## 768                    Israel      Asia 2007  80.745    6426679 25523.2771
## 780                     Italy    Europe 2007  80.546   58147733 28569.7197
## 792                   Jamaica  Americas 2007  72.567    2780132  7320.8803
## 804                     Japan      Asia 2007  82.603  127467972 31656.0681
## 816                    Jordan      Asia 2007  72.535    6053193  4519.4612
## 828                     Kenya    Africa 2007  54.110   35610177  1463.2493
## 840          Korea, Dem. Rep.      Asia 2007  67.297   23301725  1593.0655
## 852               Korea, Rep.      Asia 2007  78.623   49044790 23348.1397
## 864                    Kuwait      Asia 2007  77.588    2505559 47306.9898
## 876                   Lebanon      Asia 2007  71.993    3921278 10461.0587
## 888                   Lesotho    Africa 2007  42.592    2012649  1569.3314
## 900                   Liberia    Africa 2007  45.678    3193942   414.5073
## 912                     Libya    Africa 2007  73.952    6036914 12057.4993
## 924                Madagascar    Africa 2007  59.443   19167654  1044.7701
## 936                    Malawi    Africa 2007  48.303   13327079   759.3499
## 948                  Malaysia      Asia 2007  74.241   24821286 12451.6558
## 960                      Mali    Africa 2007  54.467   12031795  1042.5816
## 972                Mauritania    Africa 2007  64.164    3270065  1803.1515
## 984                 Mauritius    Africa 2007  72.801    1250882 10956.9911
## 996                    Mexico  Americas 2007  76.195  108700891 11977.5750
## 1008                 Mongolia      Asia 2007  66.803    2874127  3095.7723
## 1020               Montenegro    Europe 2007  74.543     684736  9253.8961
## 1032                  Morocco    Africa 2007  71.164   33757175  3820.1752
## 1044               Mozambique    Africa 2007  42.082   19951656   823.6856
## 1056                  Myanmar     Asia 2007  62.069  47761980   944.0000
## 1068                  Namibia   Africa 2007  52.906   2055080  4811.0604
## 1080                    Nepal     Asia 2007  63.785  28901790  1091.3598
## 1092              Netherlands   Europe 2007  79.762  16570613 36797.9333
## 1104              New Zealand  Oceania 2007  80.204   4115771 25185.0091
## 1116                Nicaragua Americas 2007  72.899   5675356  2749.3210
## 1128                    Niger   Africa 2007  56.867  12894865   619.6769
## 1140                  Nigeria   Africa 2007  46.859 135031164  2013.9773
## 1152                   Norway   Europe 2007  80.196   4627926 49357.1902
## 1164                     Oman     Asia 2007  75.640   3204897 22316.1929
## 1176                 Pakistan     Asia 2007  65.483 169270617  2605.9476
## 1188                   Panama Americas 2007  75.537   3242173  9809.1856
## 1200                 Paraguay Americas 2007  71.752   6667147  4172.8385
## 1212                     Peru Americas 2007  71.421  28674757  7408.9056
## 1224              Philippines     Asia 2007  71.688  91077287  3190.4810
## 1236                   Poland   Europe 2007  75.563  38518241 15389.9247
## 1248                 Portugal   Europe 2007  78.098  10642836 20509.6478
## 1260              Puerto Rico Americas 2007  78.746   3942491 19328.7090
## 1272                  Reunion   Africa 2007  76.442    798094  7670.1226
## 1284                  Romania   Europe 2007  72.476  22276056 10808.4756
## 1296                   Rwanda   Africa 2007  46.242   8860588   863.0885
## 1308    Sao Tome and Principe   Africa 2007  65.528    199579  1598.4351
## 1320             Saudi Arabia     Asia 2007  72.777  27601038 21654.8319
## 1332                  Senegal   Africa 2007  63.062  12267493  1712.4721
## 1344                   Serbia   Europe 2007  74.002  10150265  9786.5347
## 1356             Sierra Leone   Africa 2007  42.568   6144562   862.5408
## 1368                Singapore     Asia 2007  79.972   4553009 47143.1796
## 1380          Slovak Republic   Europe 2007  74.663   5447502 18678.3144
## 1392                 Slovenia   Europe 2007  77.926   2009245 25768.2576
## 1404                  Somalia   Africa 2007  48.159   9118773   926.1411
## 1416             South Africa   Africa 2007  49.339  43997828  9269.6578
## 1428                    Spain   Europe 2007  80.941  40448191 28821.0637
## 1440                Sri Lanka     Asia 2007  72.396  20378239  3970.0954
## 1452                    Sudan   Africa 2007  58.556  42292929  2602.3950
## 1464                Swaziland   Africa 2007  39.613   1133066  4513.4806
## 1476                   Sweden   Europe 2007  80.884   9031088 33859.7484
## 1488              Switzerland   Europe 2007  81.701   7554661 37506.4191
## 1500                    Syria     Asia 2007  74.143  19314747  4184.5481
## 1512                   Taiwan     Asia 2007  78.400  23174294 28718.2768
## 1524                 Tanzania   Africa 2007  52.517  38139640  1107.4822
## 1536                 Thailand     Asia 2007  70.616  65068149  7458.3963
## 1548                     Togo   Africa 2007  58.420   5701579   882.9699
## 1560      Trinidad and Tobago Americas 2007  69.819   1056608 18008.5092
## 1572                  Tunisia   Africa 2007  73.923  10276158  7092.9230
## 1584                   Turkey   Europe 2007  71.777  71158647  8458.2764
## 1596                   Uganda   Africa 2007  51.542  29170398  1056.3801
## 1608           United Kingdom   Europe 2007  79.425  60776238 33203.2613
## 1620            United States Americas 2007  78.242 301139947 42951.6531
## 1632                  Uruguay Americas 2007  76.384   3447496 10611.4630
## 1644                Venezuela Americas 2007  73.747  26084662 11415.8057
## 1656                  Vietnam     Asia 2007  74.249  85262356  2441.5764
## 1668        West Bank and Gaza     Asia 2007  73.422   4018332  3025.3498
## 1680              Yemen, Rep.     Asia 2007  62.698  22211743  2280.7699
## 1692                   Zambia   Africa 2007  42.384  11746035  1271.2116
## 1704                 Zimbabwe    Africa 2007  43.487   12311143   469.7093
```

Now let's find the average life expectancy (in 2007) of countries in the various continents

Step 1. Filter by each continent. Let's start with Asia.

```
print(gapminder_2007df[gapminder_2007df$continent == "Asia",])
```

```
##                 country continent year lifeExp        pop  gdpPercap
## 12          Afghanistan      Asia 2007  43.828   31889923   974.5803
## 96              Bahrain      Asia 2007  75.635     708573 29796.0483
## 108          Bangladesh      Asia 2007  64.062  150448339  1391.2538
## 228            Cambodia      Asia 2007  59.723   14131858  1713.7787
## 300               China      Asia 2007  72.961 1318683096  4959.1149
## 672    Hong Kong, China      Asia 2007  82.208    6980412 39724.9787
## 708               India      Asia 2007  64.698 1110396331  2452.2104
## 720           Indonesia      Asia 2007  70.650  223547000  3540.6516
## 732                Iran      Asia 2007  70.964   69453570 11605.7145
## 744                Iraq      Asia 2007  59.545   27499638  4471.0619
## 768              Israel      Asia 2007  80.745    6426679 25523.2771
## 804               Japan      Asia 2007  82.603  127467972 31656.0681
## 816              Jordan      Asia 2007  72.535    6053193  4519.4612
## 840    Korea, Dem. Rep.      Asia 2007  67.297   23301725  1593.0655
## 852         Korea, Rep.      Asia 2007  78.623   49044790 23348.1397
## 864              Kuwait      Asia 2007  77.588    2505559 47306.9898
## 876             Lebanon      Asia 2007  71.993    3921278 10461.0587
## 948            Malaysia      Asia 2007  74.241   24821286 12451.6558
## 1008           Mongolia      Asia 2007  66.803    2874127  3095.7723
## 1056            Myanmar      Asia 2007  62.069   47761980   944.0000
## 1080              Nepal      Asia 2007  63.785   28901790  1091.3598
## 1164               Oman      Asia 2007  75.640    3204897 22316.1929
## 1176           Pakistan      Asia 2007  65.483  169270617  2605.9476
## 1224        Philippines      Asia 2007  71.688   91077287  3190.4810
## 1320       Saudi Arabia      Asia 2007  72.777   27601038 21654.8319
## 1368          Singapore      Asia 2007  79.972    4553009 47143.1796
## 1440          Sri Lanka      Asia 2007  72.396   20378239  3970.0954
## 1500              Syria      Asia 2007  74.143   19314747  4184.5481
## 1512             Taiwan      Asia 2007  78.400   23174294 28718.2768
## 1536           Thailand      Asia 2007  70.616   65068149  7458.3963
## 1656            Vietnam      Asia 2007  74.249   85262356  2441.5764
## 1668 West Bank and Gaza      Asia 2007  73.422    4018332  3025.3498
## 1680        Yemen, Rep.      Asia 2007  62.698   22211743  2280.7699
```

Here, as you see, only those countries that are in Asia are shown.

Step 2. Fetch the life expectancy of Asian countries in 2007.

```
print(gapminder_2007df[gapminder_2007df$continent == "Asia",]$lifeExp)
```

```
##  [1] 43.828 75.635 64.062 59.723 72.961 82.208 64.698 70.650 70.964 59.545
## [11] 80.745 82.603 72.535 67.297 78.623 77.588 71.993 74.241 66.803 62.069
## [21] 63.785 75.640 65.483 71.688 72.777 79.972 72.396 74.143 78.400 70.616
## [31] 74.249 73.422 62.698
```

Step 3. Calculate the mean.

```
print(mean(gapminder_2007df[gapminder_2007df$continent == "Asia",]$lifeExp))
```

```
## [1] 70.72848
```

Repeat for the other continents. Which continent had the highest average life expectancy in 2007?

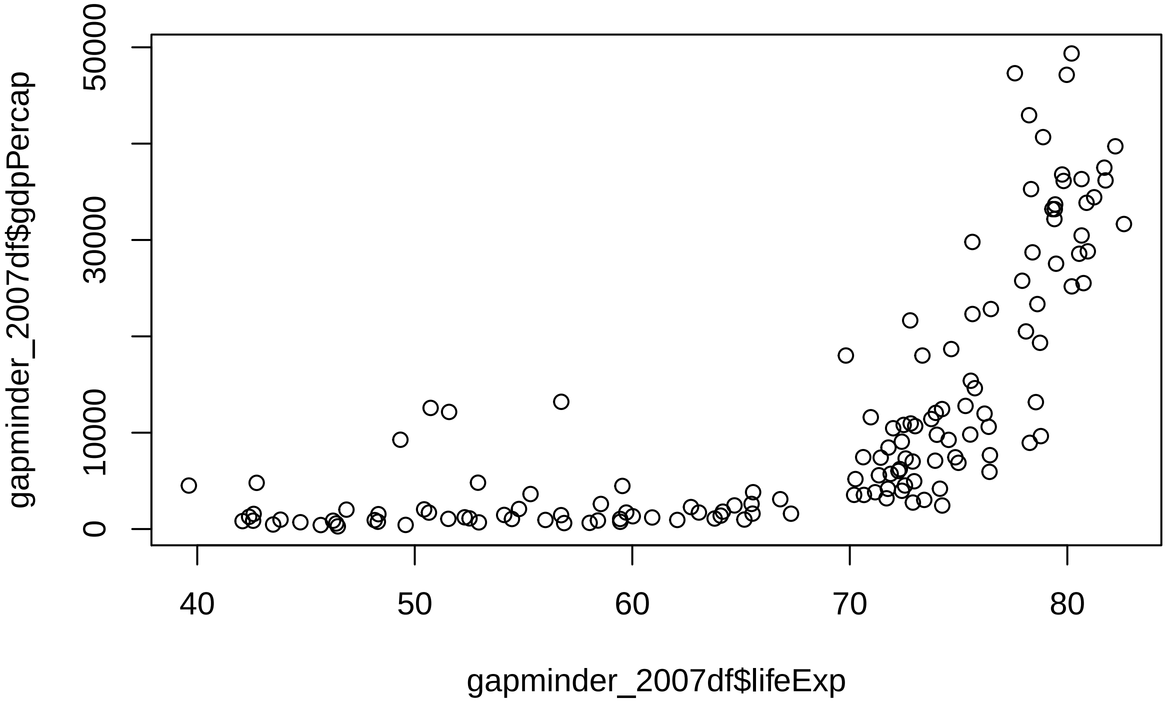
Hint: you can find the names of all the continents using the `unique` command like this:

```
print(unique(gapminder_2007df$continent))
```

```
## [1] "Asia"     "Europe"   "Africa"   "Americas" "Oceania"
```

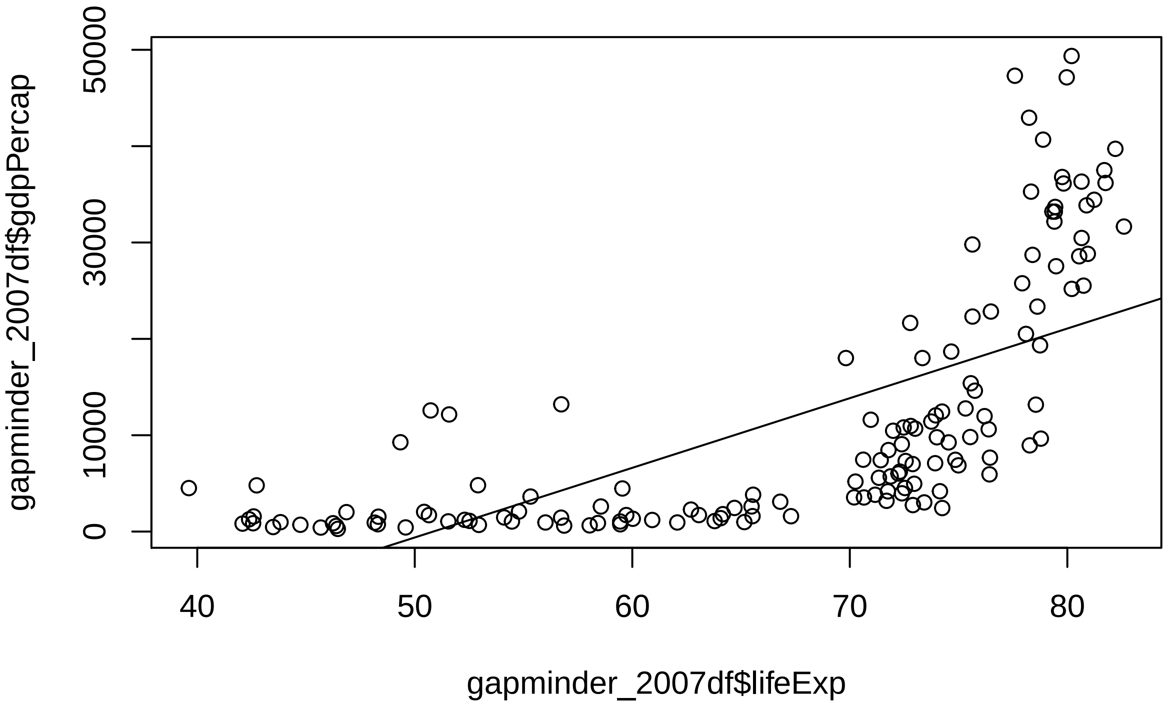
Finally, let's draw our first R graph. Let's have "life expectancy on X axis" and "GDP per capita" on the y axis.

```
plot(gapminder_2007df$lifeExp, gapminder_2007df$gdpPercap)
```



Now, let's draw a "regression line" between these two variables: life expectancy and GDP per capita. To do that, first plot the graph, and then draw the line using the `abline` command. Notice the `lm` command and the use of the tilde `~`. This is instructing R to fit a "linear model" (lm) between GDP per capita and life expectancy.

```r
plot(gapminder_2007df$lifeExp, gapminder_2007df$gdpPercap)
abline(lm(gapminder_2007df$gdpPercap ~ gapminder_2007df$lifeExp))
```



Exercises:

Using what you have learned so far, answer the following questions using the gapminder dataset.

1. Which country had the 5th highest population in 1957? (Hint: use the `order` command)
2. Which continent had the lowest average population in 1982?
3. Which country had the highest per capita GDP in 1952?