Practical - beginner

Aurelien Ginolhac

2nd June 2016

Project - set-up

- Create a new project in a meaningful folder name on your computer such as `R_workshop/day1-beginner`

using the project manager utility, top-right of the rstudio window.

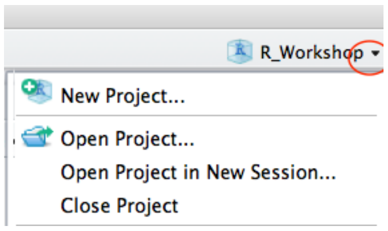

Figure 1: project menu

- Create a new folder `data` using bottom-right panel > **Files** tab > *New Folder* button

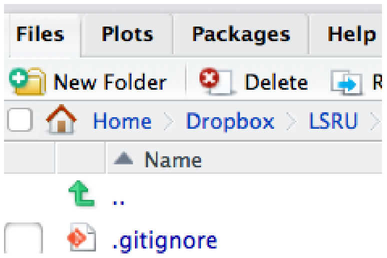

Figure 2: Files tab

- Create a new script to write and execute your `R` commands. top-left panel > **Create** icon > *New Script* entry.

Now, you have the 4 panels of the rstudio layout.

- Save the script with a relevant name `practical-beginner.R`

Reading data

Download this simple tab-separated file http://lsru.github.io/r_workshop/data/women.tsv

and save it inside the folder `R_workshop/day1-beginner/data`.

Remember, your current active rstudio project should be `day1-beginner`

load it: All paths are relative to the root which is the projects folder

```
library("readr")
df <- read_tsv("data/women.tsv", col_names = TRUE)
df
```

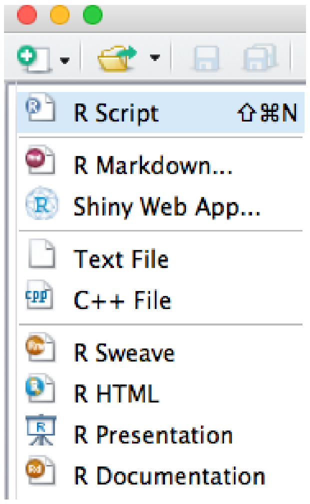

Figure 3: create menu

```
##    height weight
## 1      58    115
## 2      59    117
## 3      60    120
## 4      61    123
## 5      62    126
## 6      63    129
## 7      64    132
## 8      65    135
## 9      66    139
## 10     67    142
## 11     68    146
## 12     69    150
## 13     70    154
## 14     71    159
## 15     72    164
```

Thanks to `readr` the object `df` is already a *tibble diff* rstudio blog: tibble

Manipulate a data frame

We keep this section short, as we will focus on `dplyr` to perform tasks on `data frames`

Access to one column, display only the first elements

```
head(df$height)
```

```
## [1] 58 59 60 61 62 63
```

Using a similar syntax, apply:

- the function `mean()` to find the mean of women' height.
- the function `var()` to find the variance of women' weight.

To compute her BMI (remember `height` are inches and `weight` US pounds) the formula is:

$$BMI = \frac{weight}{height^2} * 703$$

For the first individual (^2 for square):

```
(115 / 58^2) * 703
```

```
## [1] 24.0324
```

- Compute the BMI for all individuals, save it as `bmi`
- Compute the mean and median of all BMI

plotting

First load `dplyr`. This enables the use of the `%>%` pipe operator

```
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Using `df` dataset:

- plot the `heigh` in function of the weight (`geom_point()`)
- use the previous scatterplot, but map the point' size to the `bmi`

tidying and plotting

`df` has 2 columns, both contain values. - use `gather()` from `tidyr` to get two columns + `measure` for either height or weight + `value` for actual measurements Remember that `gather` takes by default all columns. - store the result into `df_melt`

- plot the distribution as boxplots of both measures

plot densities

adding a column to a data frame

Let's add `bmi` as a third column to `df`.

```
df$bmi <- bmi
head(df)
```

```
## Source: local data frame [6 x 3]
##
##   height weight      bmi
##    (int)  (int)    (dbl)
## 1      58     115 24.03240
## 2      59     117 23.62856
## 3      60     120 23.43333
## 4      61     123 23.23811
## 5      62     126 23.04318
## 6      63     129 22.84883
```

plot densities

- tidy the 3 columns and plot all densities using different colours and set them translucent You will need to make a new `df_melt` data frame first.

The 3 distributions have very different ranges.

- Plot the same data but faceting it by `measure` (Use the appropriate **free** `scale`).

When faceting, the 3 distributions are drawn in distinct plots: mapping the colours to `measure` is useless.

- redo the plot using a `lightblue` colour for all. Be careful to **NOT** set the colour inside `aes()`.

Supplementary exercices

reading more complex file

Microarray data from the GEO dataset GSE35982.

- download this compressed file: GSE35982.tsv.gz in your `data` folder.
- read it using `read_tsv()` and store it into a data frame named `gse`. The file will be uncompressed seamlessly.
- Is the file tidy?
- Tidy the samples. Look at the `gather` help page to select columns based on characters.
- plot the distributions as boxplots
- Any obvious issues? Check the file and find out what happened.

Hint

the `locale` setting in `readr` allows to specify the decimal mark used for float numbers

- Correct the mistake by reading again the file with the adjusted relevant option and store the data into a a new object.
- Replace the wrong column in `gse` by the correct one found in the data frame you just created.
- tidy the samples again.
- plot the distributions as boxplots
- do the data appear normalised?