Warm up for model fitting

Fang Wang

3 Apr 2015

1 We start from the very beginning

1.1 Experiment design

This will not be included in our course but you could always discuss in your practise time or model, especially with Pedro, about your experiments.

Read this first: experiment design.

1.2 Data collection

Probably all of you went through this step.

Reading .csv

I saved same data from the previous practise to a windows csv and a MS-dos one. Both are working by the following chunk.

I show here the commented lines here to tell you, that if these lines are run, .Rmd can't be compiled. You can run the first line, which tells you what is the current working directory in the console, and compare the value returned to the value you get in the .pdf file generated by this script. We use `.df` as a name ending to help us remind that the value returned by `read.csv` and `read.csv2` is a `data.frame`, but this is just our own convention, `R` does not care as long as the name is valid.

```
getwd()
```

```
## [1] "C:/Users/aphalo/Documents/Rteaching/r-course-2015/Fang/work-flow-data-analysis"
```

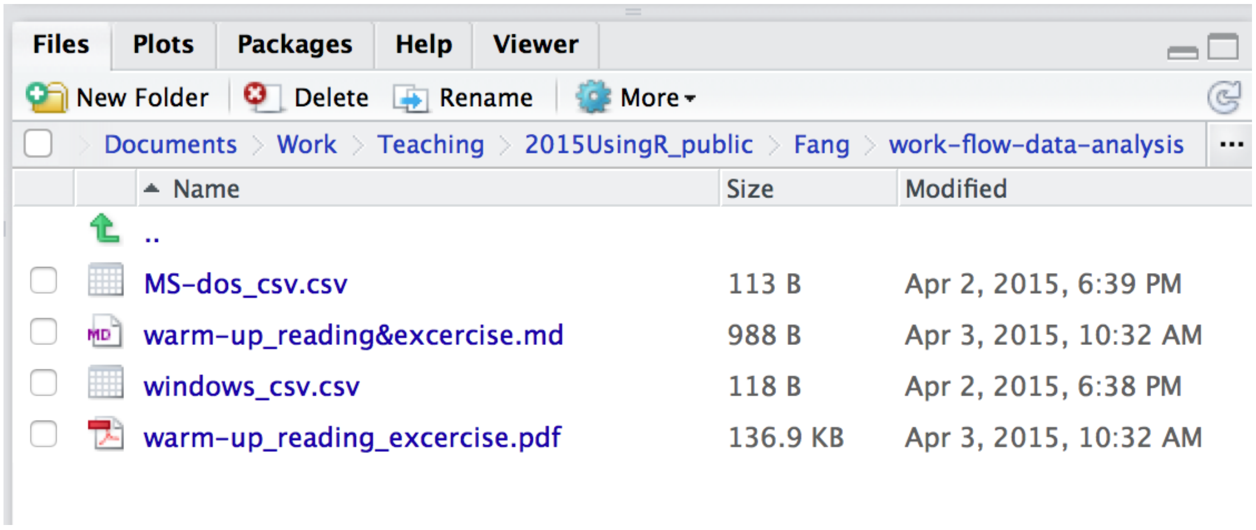
```
# root_wd <- getwd()
# setwd("./Fang/work-flow-data-analysis")

dos.csv.df <- read.csv2("MS-dos_csv.csv")
# windows.csv <- read.csv2("windows_csv.csv")

# View(dos.csv.df)
```

If you want to see further info on **CSV** file format, check this web page.

In class, last Wednesday, the reason why file reading was not working is that Rmd compiling directory is where .Rmd file is located. It has nothing to do with `.Rdata` file, which is where `R` stores the "environment" when a session is closed. It is possible to use a nested folder or subdirectory but it can be bit confusing if you are not very familiar with the operating system. Now if you understand this, then explain the reason I said in the last chunk if the comment lines are uncommented, this .Rmd file can't be compiled.

If this is too difficult for you to follow, then just keep this in your mind: **Writing in a R script (.R), then just keep all scripts and data files in the same directory where .Rdata / .Rproj are located**. So if you run the last chunk directly in console, does it work? Why?

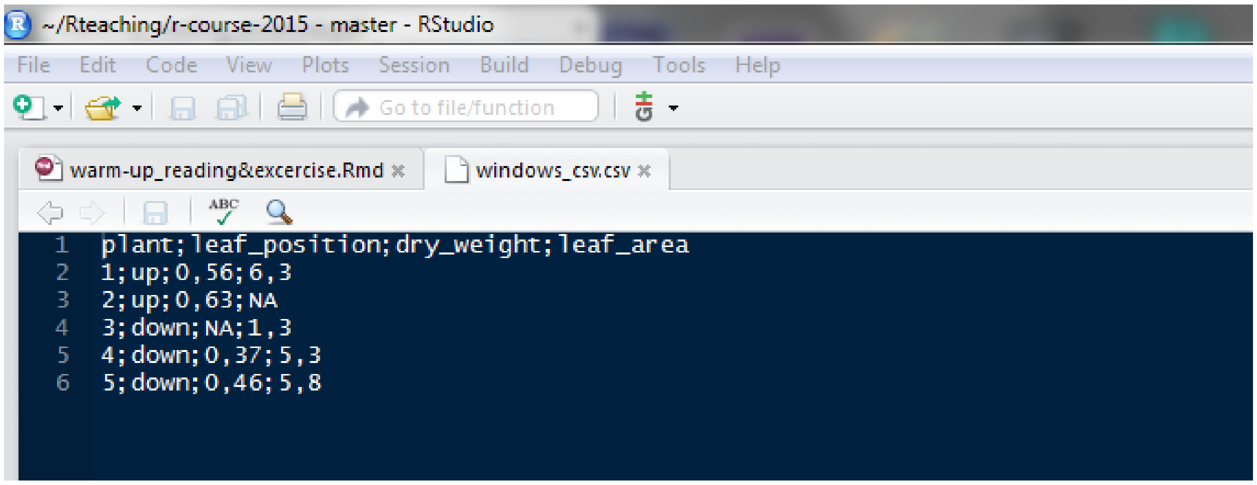
NB: If you click .csv in the file browser, which looks as



	Name	Size	Modified
	..		
	MS-dos_csv.csv	113 B	Apr 2, 2015, 6:39 PM
	warm-up_reading&excercise.md	988 B	Apr 3, 2015, 10:32 AM
	windows_csv.csv	118 B	Apr 2, 2015, 6:38 PM
	warm-up_reading_excercise.pdf	136.9 KB	Apr 3, 2015, 10:32 AM

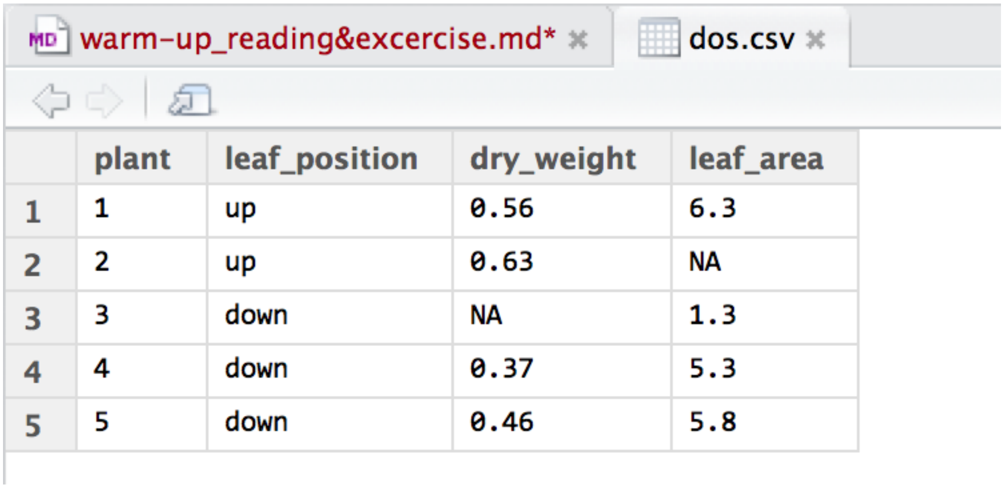
Files
Plots
Packages
Help
Viewer
New Folder
Delete
Rename
More
Documents > Work > Teaching > 2015UsingR_public > Fang > work-flow-data-analysis

the file shows what is the text inside. Notice here everything is semicolon separated.



```
plant;leaf_position;dry_weight;leaf_area
1;up;0,56;6,3
2;up;0,63;NA
3;down;NA;1,3
4;down;0,37;5,3
5;down;0,46;5,8
```

But if you view data frame (using `View()`), you see the data frame.



	plant	leaf_position	dry_weight	leaf_area
1	1	up	0.56	6.3
2	2	up	0.63	NA
3	3	down	NA	1.3
4	4	down	0.37	5.3
5	5	down	0.46	5.8

These small things just need time to practise. You will notice yourself step by step.

1.3 Explore data

- [ ] Check the structure of the data `str()`
- [ ] Check all the variable names `names()`

More to check here. Apply these functions to *dos.scv*. If console complains, why? (Added by Pedro: The page in the link is not perfect, but shows you how one works with `R`: one frequently searches for a suitable example and adapts it to ones own needs and tastes. For example I do not see any point in using a "pie chart" and I would use `ggplot` instead of the base R functions for plotting... so after applying the code as is to the data preapred by Fang, do try variations of the code that you thing would help. Two functions not used in the example, that would be worthwhile trying are `head()` and `tail()`.)

1.4 Plotting

You can try the simplest `plot()` function. Or use ggplot2 package.

When begin to use a package, the first thing is to load the package. Then you can freely use all the functions.

```
library(ggplot2)
```

This is easily forgotten.

Then, try to think the syntax again Pedro told in the lecture.

```
# this line is easy for me to check the names of the variables.
# Especially R is case sensitive. It is always good to check if you right spell everything.
names(dos.csv.df)
```

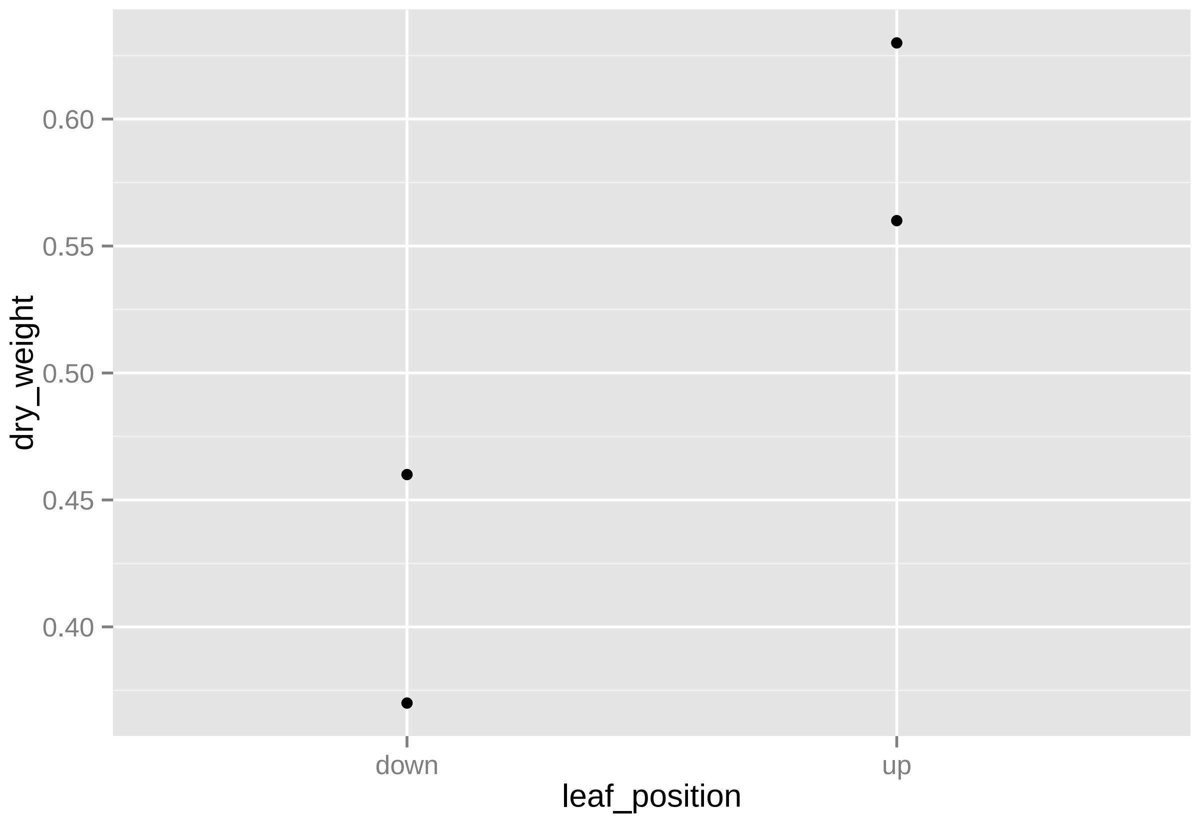
```
## [1] "plant"          "leaf_position" "dry_weight"    "leaf_area"
```

```
# assign the plot to an object (here is "explore.fig"). It is easier to call this figure again.
explore.fig <-
  ggplot(dos.csv.df,
         aes(x=leaf_position, y=dry_weight)) +
  geom_point()
```

If you want to see the figure, you have to print this figure object which in the console happens implicitly by entering its name.

```
explore.fig
```

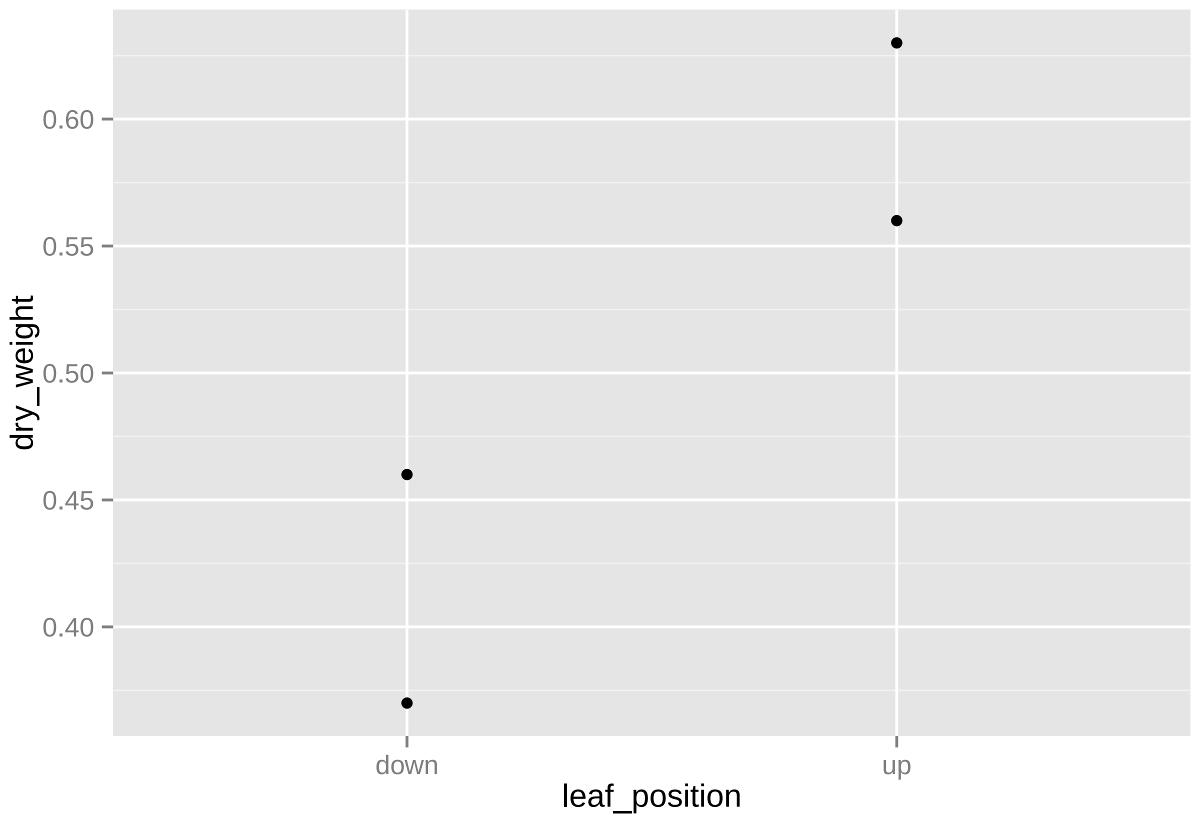
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Or print it explicitly which always works, even when sourcing an R script.

```
print(explore.fig)
```

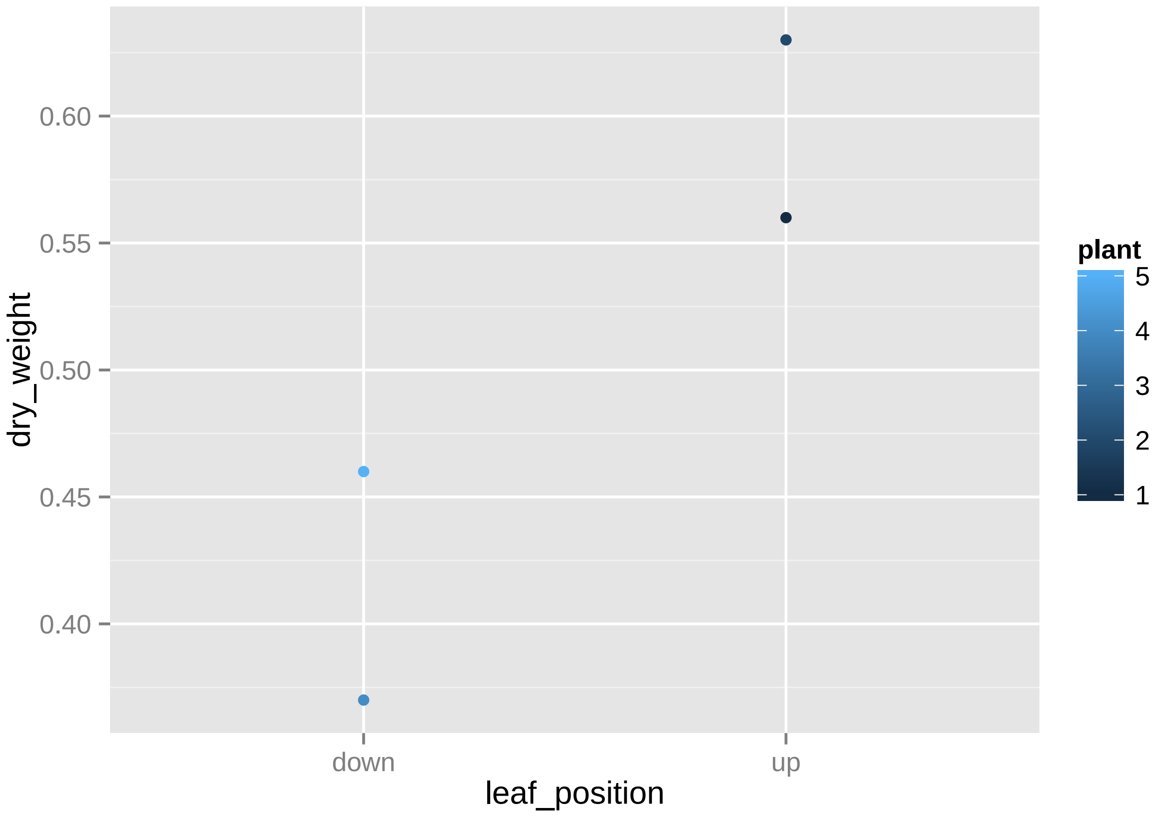
```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Now I want to mark the points with colours and each plant has own colour. Firstly, I have to assign **colour** to **aes()** setting. And tell how the colour is assigned by putting an another *aesthetic* (inside `aes()`). As `colour` is `numeric` the colour scale is continuous, given by default by a gradient between blue and black.

```
explore.fig <-
  ggplot(dos.csv.df,
         aes(x=leaf_position, y=dry_weight, colour=plant)) +
  geom_point()

explore.fig
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

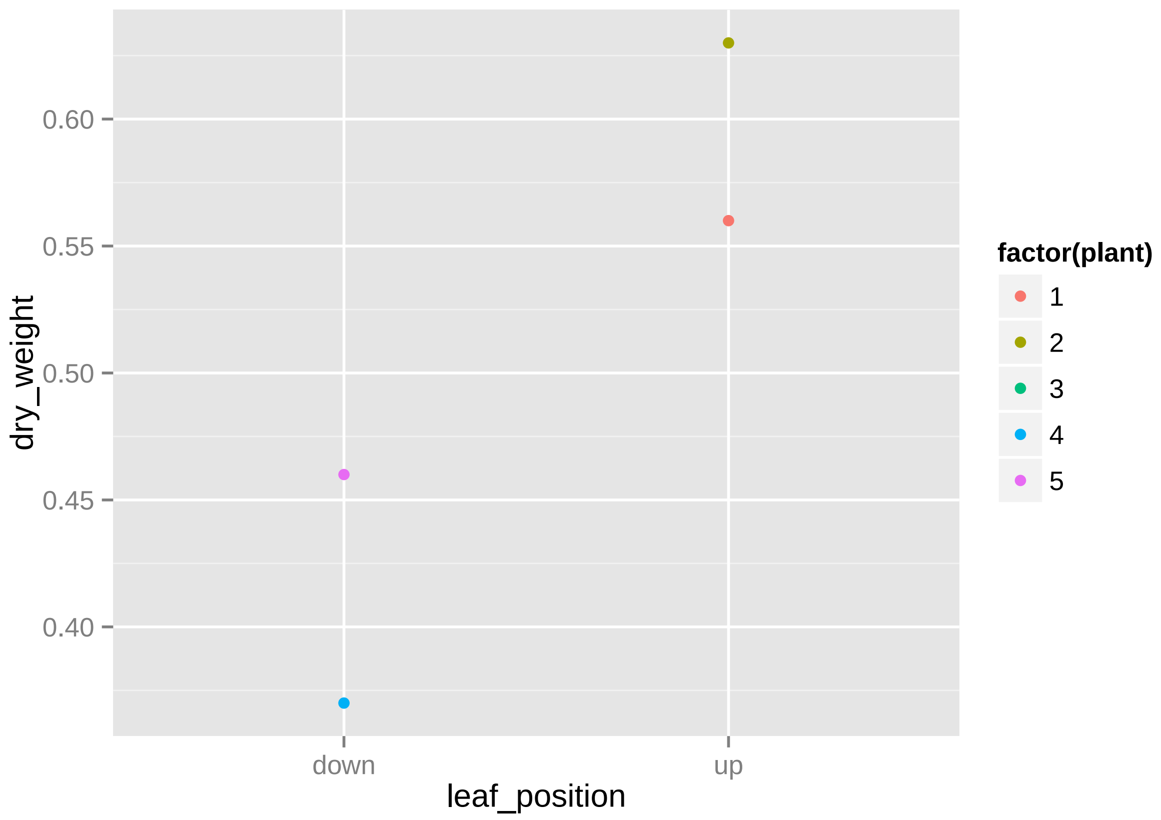


```
# compare what I changed, and how image changes.
```

However, `plant` is not really a continuous variable, the numbers are just *labels* given to individual plants, so they are simply *categorical* and *unordered* values, that should be represented with a `factor` instead. We can do this on the fly.

```
explore_factor.fig <-
  ggplot(dos.csv.df,
        aes(x=leaf_position, y=dry_weight, colour=factor(plant))) +
  geom_point()

explore_factor.fig
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



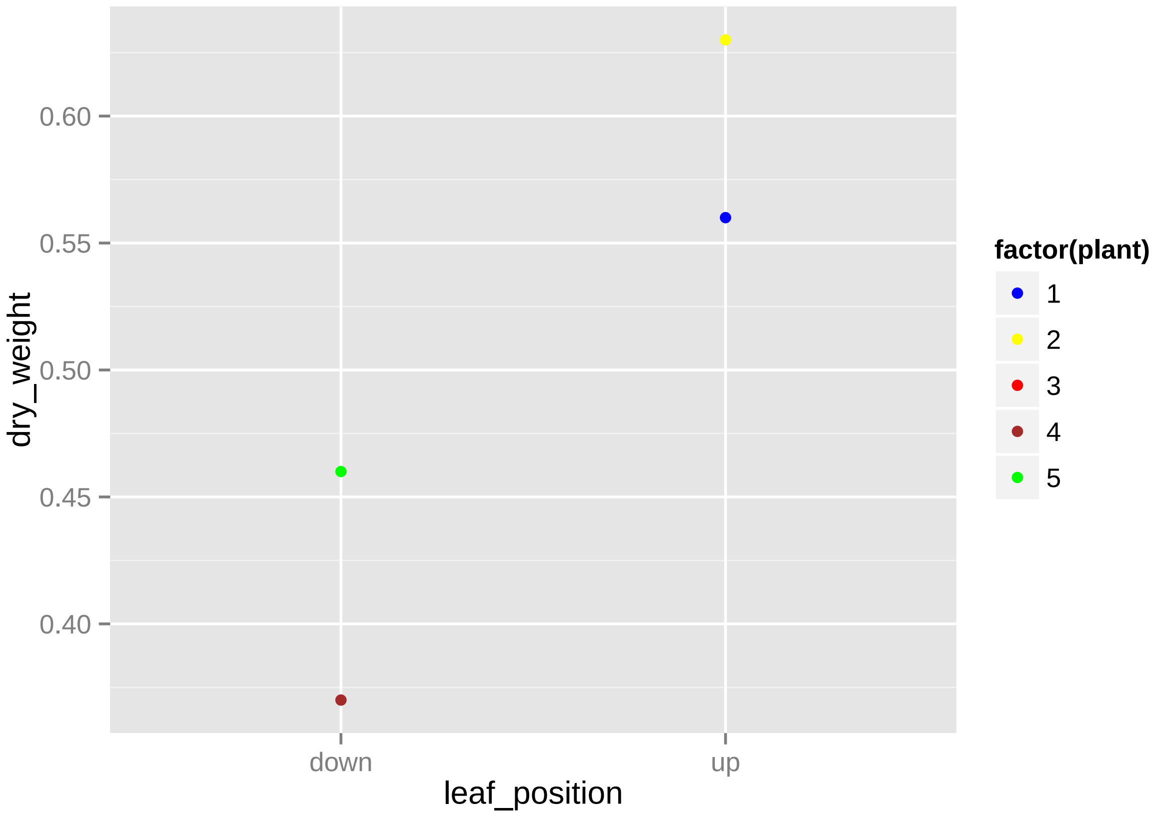
```
# compare what I changed, and how image changes.
```

I am not happy with the colour. So I want to manually assign the colours to the different plants. I have to put a new scale (a scale is not a layer, but it changes how the layer already in the figure is displayed). I already told ggplot that *aesthetic* colour is given according `factor(plant)`.

Now the colours are different.

```
explore_factor.fig <- explore_factor.fig +
  scale_colour_manual(values=c("blue","yellow","red","brown","green"))
explore_factor.fig
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



The scale used above, is a *discrete* scale, so had we apply it to an *aesthetic* assigned to a `numeric` variable as in our first attempt to plot these data, what would happen?

```
explore.fig <- explore.fig +
  scale_colour_manual(values=c("blue","yellow","red","brown","green"))
explore.fig
```

If you run this in console now, what it complains?

So, as we saw above, although you would still get a valid plot, you have to use a `factor` to be able to use a *discrete scale*. A factor has discrete levels. You can understand it in a way for your experiments. Your treatment is a factor and different treatments are different levels (unless the treatments are different amounts, or doses, of the same "thing"). And levels have own labels. If you write your treatments in data recording as "drug1", "drug2", "drug3", "drug4" and "placebo". How many levels of this factor? 5. What are the labels? What are in the "". You can change the order of the levels (I think the easiest way. More to check here.) or the labels names. **NB** just changing labels' names won't change levels' order.

Above, we converted the `numeric` variable into a `factor` on-the-fly within `aes()` but this can easily be a source of mistakes, as the property of a variable being discrete is a *permanent* property. It is much better to convert the variable `plant` into a `factor` and store it as such in the `data.frame`.

```
dos.csv.df$plant <- factor(dos.csv.df$plant)
```

Now we can check levels and labels of `doc.csv$plant`.

```
levels(dos.csv.df$plant)
```

```
## [1] "1" "2" "3" "4" "5"
```

```
labels(dos.csv.df$plant)
```

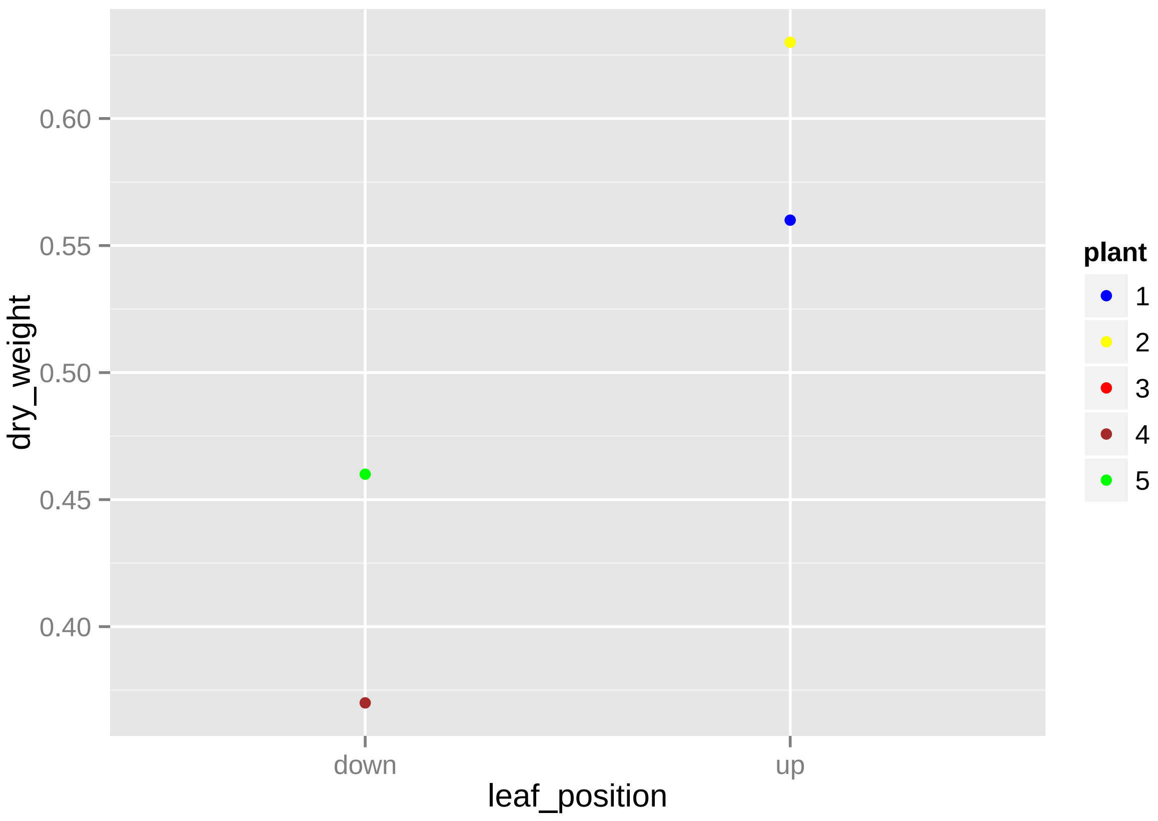
```
## [1] "1" "2" "3" "4" "5"
```

/fb{Maybe you can try yourself to change the order of the levels and the labels?}

Since we changed the data frame, ggplot has to run from the beginning.

```
explore.fig <-
  ggplot(dos.csv.df,
         aes(x=leaf_position, y=dry_weight, colour=plant)) +
  geom_point()

explore.fig <- explore.fig +
  scale_colour_manual(values=c("blue","yellow","red","brown","green"))
explore.fig
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



Understand bit more of ggplot2? Then try other types of plots instead of point. Check the cheatsheet!! Now it is the time!

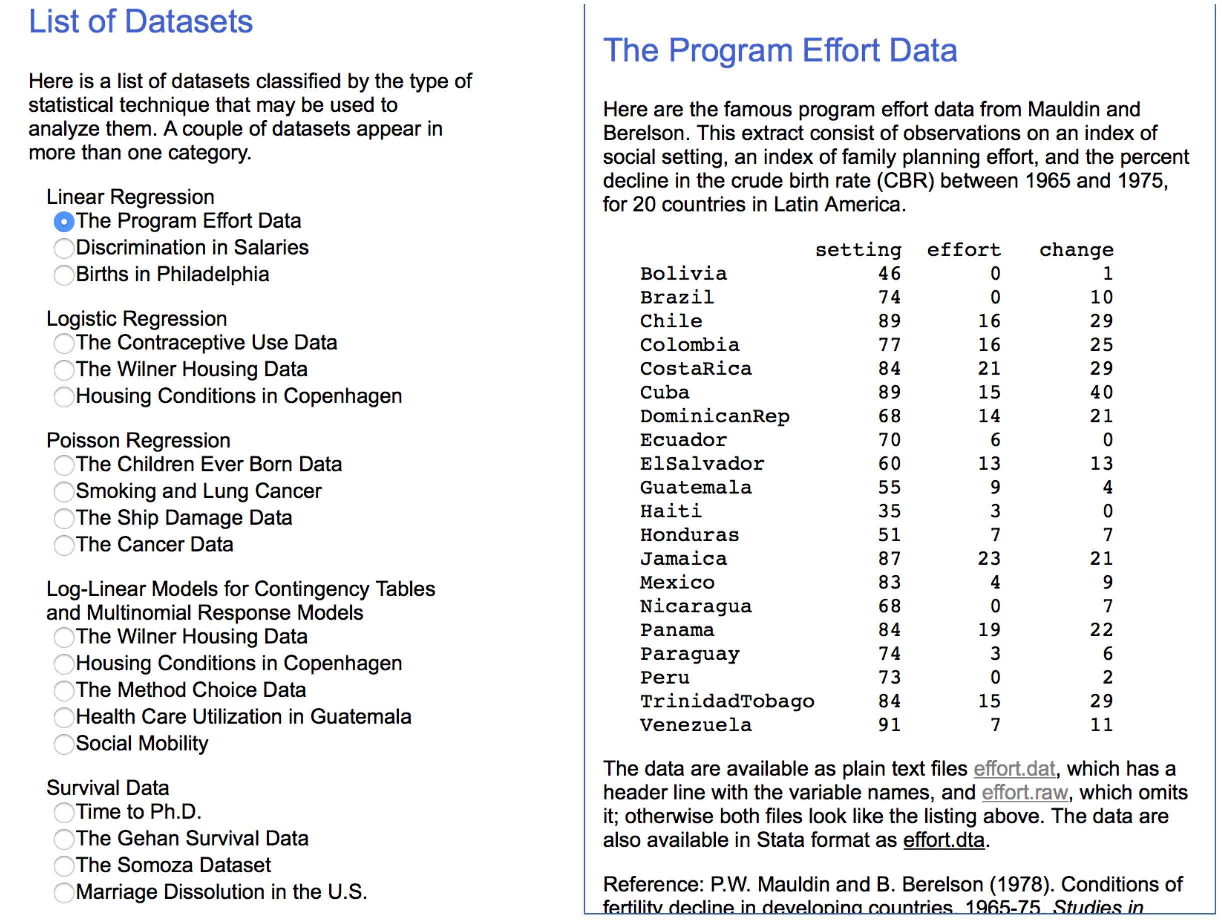
1.5 Go back to experiment design.

If I tell you now, dos.csv.df data is from an experiment that we measured 6 plants of one genotype, leaf's dry weight and leaf area. Each leaf has own position, either up or down in the individual plant. What are the replicates here? What you would like to compare?

Think your own experiment. Again, how you arrange your experiment and where the random error comes from? Check experiment design again if you want. Write down your own answer.

2 Assignment

Write down the answers in an .Rmd file. Do the following in the same .Rmd file: 1. Read your own data into R or from online database by `read.table("url_link")`. Princeton University has a very nice way to show simple linear modelling by `R`. Worth your time to read. They offer some datasets. Click .dat down in the box,



List of Datasets

Here is a list of datasets classified by the type of statistical technique that may be used to analyze them. A couple of datasets appear in more than one category.

Linear Regression
- [✓ The Program Effort Data]
- Discrimination in Salaries
- Births in Philadelphia

Logistic Regression
- The Contraceptive Use Data
- The Wilner Housing Data
- Housing Conditions in Copenhagen

Poisson Regression
- The Children Ever Born Data
- Smoking and Lung Cancer
- The Ship Damage Data
- The Cancer Data

Log-Linear Models for Contingency Tables and Multinomial Response Models
- The Wilner Housing Data
- Housing Conditions in Copenhagen
- The Method Choice Data
- Health Care Utilization in Guatemala
- Social Mobility

Survival Data
- Time to Ph.D.
- The Gehan Survival Data
- The Somoza Dataset
- Marriage Dissolution in the U.S.

The Program Effort Data

Here are the famous program effort data from Mauldin and Berelson. This extract consist of observations on an index of social setting, an index of family planning effort, and the percent decline in the crude birth rate (CBR) between 1965 and 1975, for 20 countries in Latin America.

	setting	effort	change
Bolivia	46	0	1
Brazil	74	0	10
Chile	89	16	29
Colombia	77	16	25
CostaRica	84	21	29
Cuba	89	15	40
DominicanRep	68	14	21
Ecuador	70	6	0
ElSalvador	60	13	13
Guatemala	55	9	4
Haiti	35	3	0
Honduras	51	7	7
Jamaica	87	23	21
Mexico	83	4	9
Nicaragua	68	0	7
Panama	84	19	22
Paraguay	74	3	6
Peru	73	0	2
TrinidadTobago	84	15	29
Venezuela	91	7	11

The data are available as plain text files effort.dat, which has a header line with the variable names, and effort.raw, which omits it; otherwise both files look like the listing above. The data are also available in Stata format as effort.dta.

Reference: P.W. Mauldin and B. Berelson (1978). Conditions of fertility decline in developing countries, 1965-75. *Studies in*

you could get the data link. Read the link and you can import it into your `R` space.

For example:

```r
program_effect.data <- read.table("http://data.princeton.edu/wws509/datasets/effort.dat")
View(program_effect.data)
```

It is convenient.

2. Check your data structure, names of variables, changes strings into factors (**only when they need to be factor!** Don't change whatever into factor. It could be trouble.).
3. Plot. Use `ggplot2`. Even maybe it is not necessary but it is good chance for you to practise.