Simple data mining in R with data.table and ggplot2

Pierre Tocquin

3/8/2017

Reading CSV (or any text tabulated) data

`data.table` is a package that extends the data.frame object of base R. It comes with its own functions to read (`fread`) and write (`fwrite`) tabulated data. Those replacement functions for `read.table` and `write.table` are incredibly easy to use and time/memory efficient.

```
library(data.table)
data <- fread(input = "student2/gravi.csv")
head(data)
```

```
##      Plante Position  NPA time   angle
## 1: Plante 1   Courbe Avec    0 168.527
## 2: Plante 2   Courbe Avec    0 173.283
## 3: Plante 3   Courbe Avec    0 177.797
## 4: Plante 4   Courbe Avec    0 169.992
## 5: Plante 5   Courbe Avec    0 175.236
## 6: Plante 6   Courbe Avec    0 177.780
```

`fread` is able (most of the time) to automatically identify the type and the layout of your data (separators, headers, ...).

Filtering and manipulating data

The `data.table` object comes with built-in aggregative capabilities (*via* the `by` keyword). A picture is worth a thousand words...

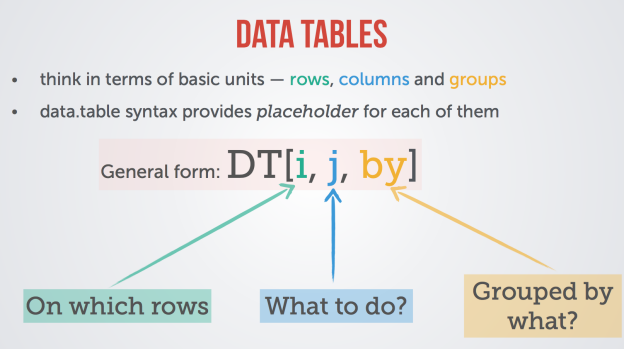

Figure 1: The data.table general form

Row filtering

The basic *row* filtering is similar to the data.frame method, **except that column names can be used as *variables*** .

```r
# The data.frame version
# head(data[data$Position == "Courbe",])

# The data.table version
head(data[Position == "Courbe"])
```

```
##      Plante Position  NPA time   angle
## 1: Plante 1   Courbe Avec    0 168.527
## 2: Plante 2   Courbe Avec    0 173.283
## 3: Plante 3   Courbe Avec    0 177.797
## 4: Plante 4   Courbe Avec    0 169.992
## 5: Plante 5   Courbe Avec    0 175.236
## 6: Plante 6   Courbe Avec    0 177.780
```

Note that with `data.table` the ',' delimiting the x,y (or i,j) dimensions of the table can be omitted when j is empty (meaning you want to return all the columns without further manipulation).

Data manipulation

At the j placeholder, direct manipulation of data can be perform by using column names as variables.

```r
data[Position == "Courbe", mean(angle)]
```

```
## [1] 162.6024
```

The j placeholder accepts a **list** of arguments, making possible multiple and simultaneous data manipulations.

```r
# data[Position == "Courbe", list(mean(angle), sd(angle))]
# The same with direct renaming of the new columns
data[Position == "Courbe", list(mean=mean(angle), sd=sd(angle))]
```

```
##        mean       sd
## 1: 162.6024 19.63851
```

Data grouping

The third placeholder, *by*, is used to aggregate data before the manipulation by the j placeholder is performed. *by* accepts as argument a **list** of one or more column names.

```r
data[Position == "Courbe", list(mean=mean(angle), sd=sd(angle)), by=list(NPA, time)]
```

```
##     NPA time     mean        sd
## 1: Avec    0 175.0569  4.272599
## 2: Sans    0 172.7916  4.996510
## 3: Avec    1 164.8701 10.752290
## 4: Sans    1 137.6909 23.009854
```

Go-beyond example

The dataset used in the previous examples was made by students who measured the response of *Arabidopsis thaliana* plantlets to a gravitropic stimulus: the plants were either submitted to the gravitropic stimulus (`Position == "Courbe"`) or not (`Position == "Debout"`) and the role of the phytohormone *auxin* in this response was evaluated by submitting some of the plants to an auxin transport inhibitor (NPA, `NPA == "Avec"` *vs* `NPA == "Sans"`). The quantification of the response was made by measuring the re-orientation of the hypocotyl one day after the onset of the stimulus, expressed as the difference between the angle (the

direction of the growth) measured after (day 1, `time == 1`) and before (day 0, `time == 0`) the stimulus. The question was to calculate the mean angle and the standard deviation for each group of experimental conditions.

Here is the way to the one-liner `data.table` solution.

1. For each plantlet in each condition, calculate the 'angle difference' between day 1 and day 0

```
step1 <- data[, list(diff=angle[time == 1]-angle[time == 0]),
              by=list(Position, NPA, Plante)]
head(step1)
```

```
##    Position  NPA   Plante    diff
## 1:   Courbe Avec Plante 1 -13.452
## 2:   Courbe Avec Plante 2 -26.212
## 3:   Courbe Avec Plante 3  -0.741
## 4:   Courbe Avec Plante 4   1.363
## 5:   Courbe Avec Plante 5   1.803
## 6:   Courbe Avec Plante 6   1.050
```

2. Calculate the mean and standard deviation for each (Position, NPA) combination of this new data table

```
step2 <- step1[, list(mean=mean(diff), sd=sd(diff)), by=list(Position, NPA)]
step2
```

```
##    Position  NPA     mean        sd
## 1:   Courbe Avec -10.1868 10.518551
## 2:   Debout Avec  -0.1395  7.567384
## 3:   Courbe Sans -35.1007 25.521659
## 4:   Debout Sans  -4.3573 13.185171
```

Both steps can be combined in one line:

```
my_summary <- data[, list(diff=angle[time == 1]-angle[time == 0]),
                   by=list(Position, NPA, Plante)][, list(mean=mean(diff), sd=sd(diff)),
                                                   by=list(Position, NPA)]
```

and eventually produce the figure...

```
library(ggplot2)
# Main parameters of the plot: What are my x ? My y ?
# On which parameter do I distinguish my data series ?

g  <- ggplot(data = my_summary, mapping = aes(x = Position, y = mean, fill = NPA))

# A 'basic' barplot + error bars
g.basic <- g +
  geom_bar(stat = "identity", position = position_dodge()) +
  geom_errorbar(width=.25, aes(ymin=mean-sd, ymax=mean), position = position_dodge(0.9))

g.basic
```

```
# An improved plot
g.improved <- g.basic +
  scale_fill_manual(values = c("red","blue")) + # changing the colors
  scale_x_discrete(labels=c("Gravitropic stimulus",
```

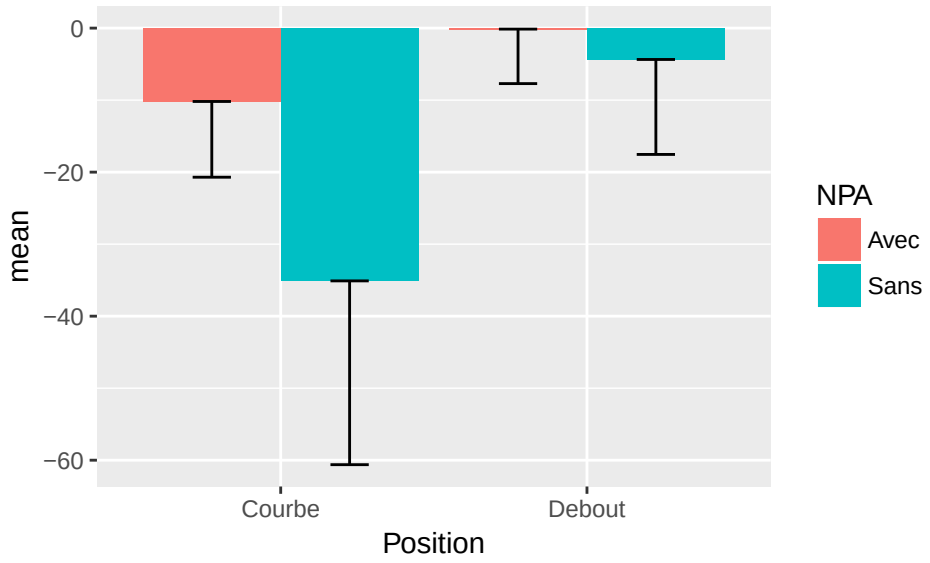

Figure 2: **My basic plot**.

```
                          "No stimulus")) + # changing the x tick labels
  ylab("Gravitopric induced curvature (°)") + # The x axis title
  xlab("") + # The x axis title
  guides(fill=FALSE) # Hide the legend

g.improved
```

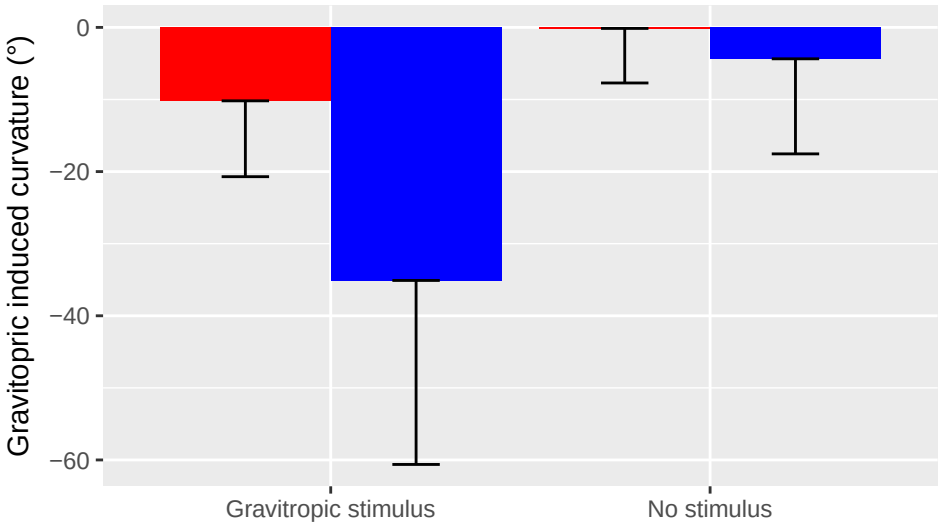

Figure 3: **My improved plot**.