The University Of Sheffield.

Introduction to Data Handling in R (BMS 109) Week5

The University Of Sheffield.

Today you will learn

- Checking your imported data
- Potting blood data charts with basic plotting
  - Scatter plot
  - Box plot
- Introduction to ggplot
  - Setting the workflow for ggplot
- Preparation to plotting with ggplot
  - Use iris data
  - Use your imported data
  - Comapring ggplot with basic plotting commands

The University Of Sheffield.

Recap from last week

The working directory is a default location where R looks for files you want to use. It is just a folder on your computer.

Setting your working directory is essential to find your data and your scripts. It **is Good Practice to do so everytime you start Rstudio.**

We can import/export data into/from our work space using CSV files and the functions *read.csv()* and *write.csv()*

We will use CSV (Comma Delimited) type when save files from Excel to be imported in R. Once read into the work space they will be of type *data.frame*

Similarly we will export *data.frame* objects in files .csv

The University Of Sheffield.

Export data

In R we can export data in a csv format after analysis using a command *write.csv()*

All data needs to be in data frame before exporting. Also make sure you have clear column names (explore *colnames()*) and row names (explore *rownames()*)

Let assume that our blood_data is ready to be exported:

```
write.csv(blood_data, file="Blod_data1.csv")
```

Your data will be save in the current working directory

The University Of Sheffield.

Setting up

- Set you working directory
- Open a new script, name it and save it
- Read the “Blood_osmosity .csv” data in your work space

```
blood_data <- read.csv("Blood_osmosity.csv",
 stringsAsFactors = TRUE) #we need Groups as factors for unclass()

blood_data
```

- Check the data, using the function str()

```
> str(blood_data)
```

- Comment briefly what you find

The University Of Sheffield.

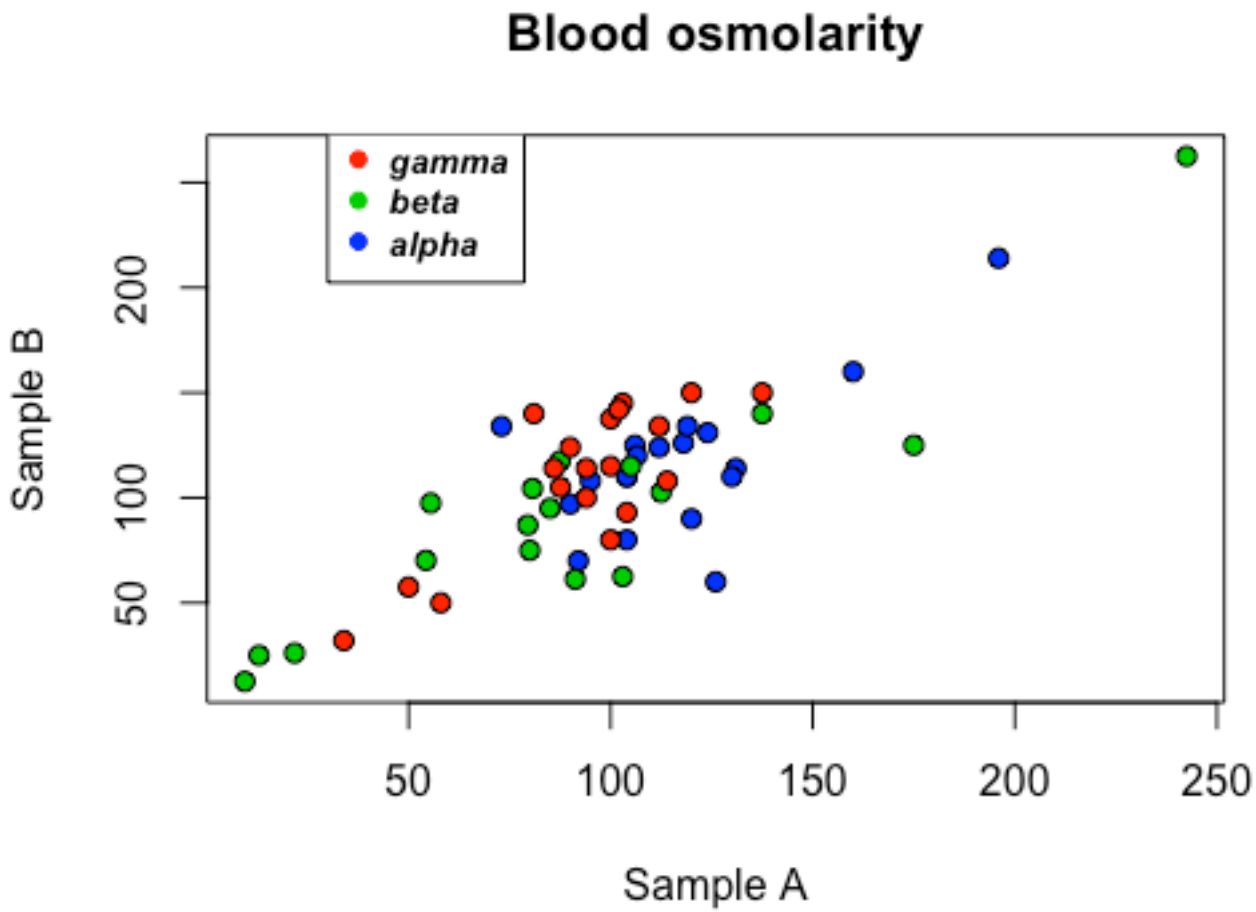
Plotting Blood Data

Exercise:
Plot the data you have imported using plot() as last week.
sampleA vs sampleB and use the color to distinguish the groups. Add the legend.

```
# scatter plots
plot(blood_data$sampleA, blood_data$sampleB, pch=21,
     bg=c("red","green3","blue")[unclass(blood_data$Group)],
     xlab="Sample A", ylab="Sample B", main="Blood osmolarity")

legend(30, 300,legend=unique(blood_data$Group),
col=c("red","green3","blue"), pch = 19,
       cex=0.8,title="Species", text.font=4)
```

The University Of Sheffield.

Blood Data scatter plot



The University Of Sheffield.

Blood Data box plot

A **box plot** is a graphical representation of groups of numerical data through their quartiles (25%, 50%, 75% of the full population) .

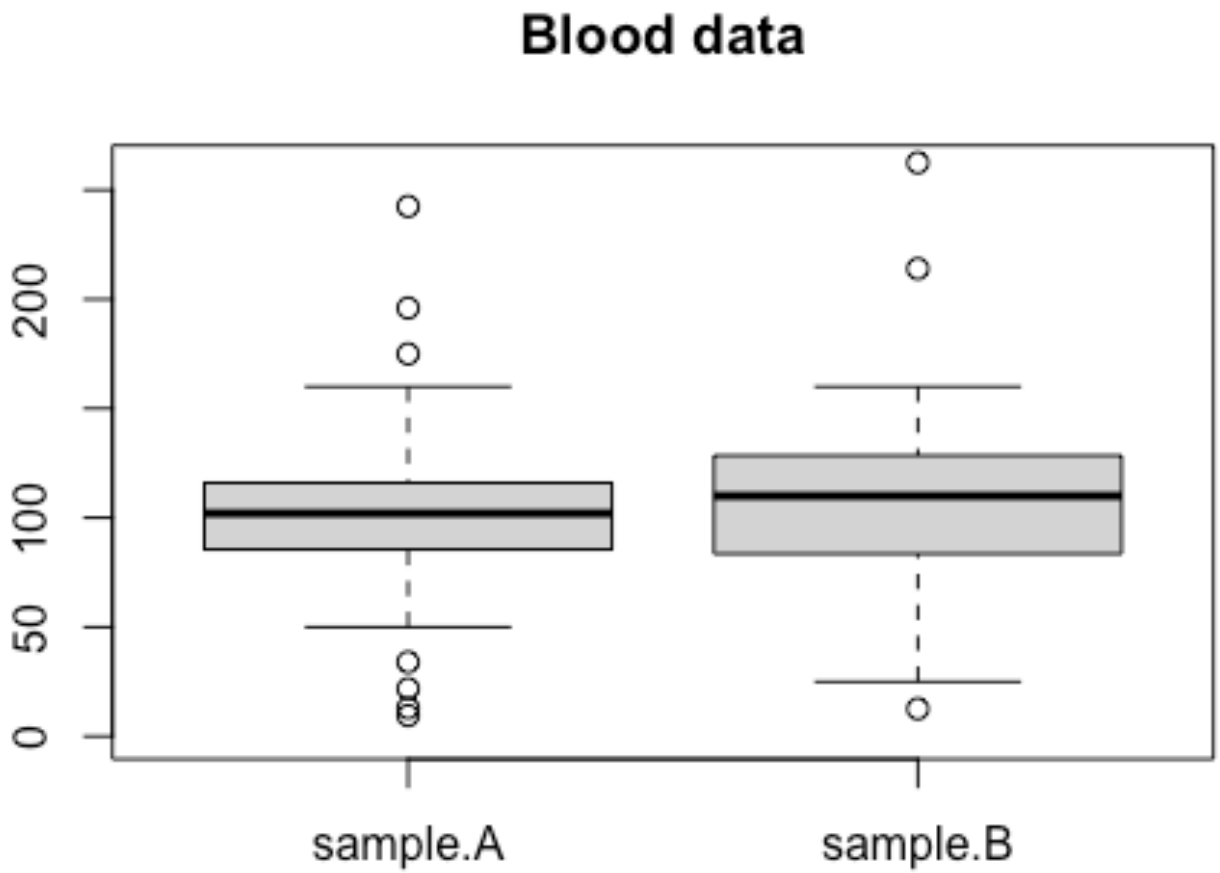
Box plots may also have lines extending vertically from the boxes (whiskers) indicating variability outside the upper (75%) and lower (25%) quartiles.

The bold line represents the median of the samples. Outliers may be plotted as individual points.

We can draw them using the function boxplot(). It wants in input a “data table”, therefore we need to use the data frame as a table (remember use of [,] for data frames.

The University Of Sheffield.

Blood Data box plot (cont…)



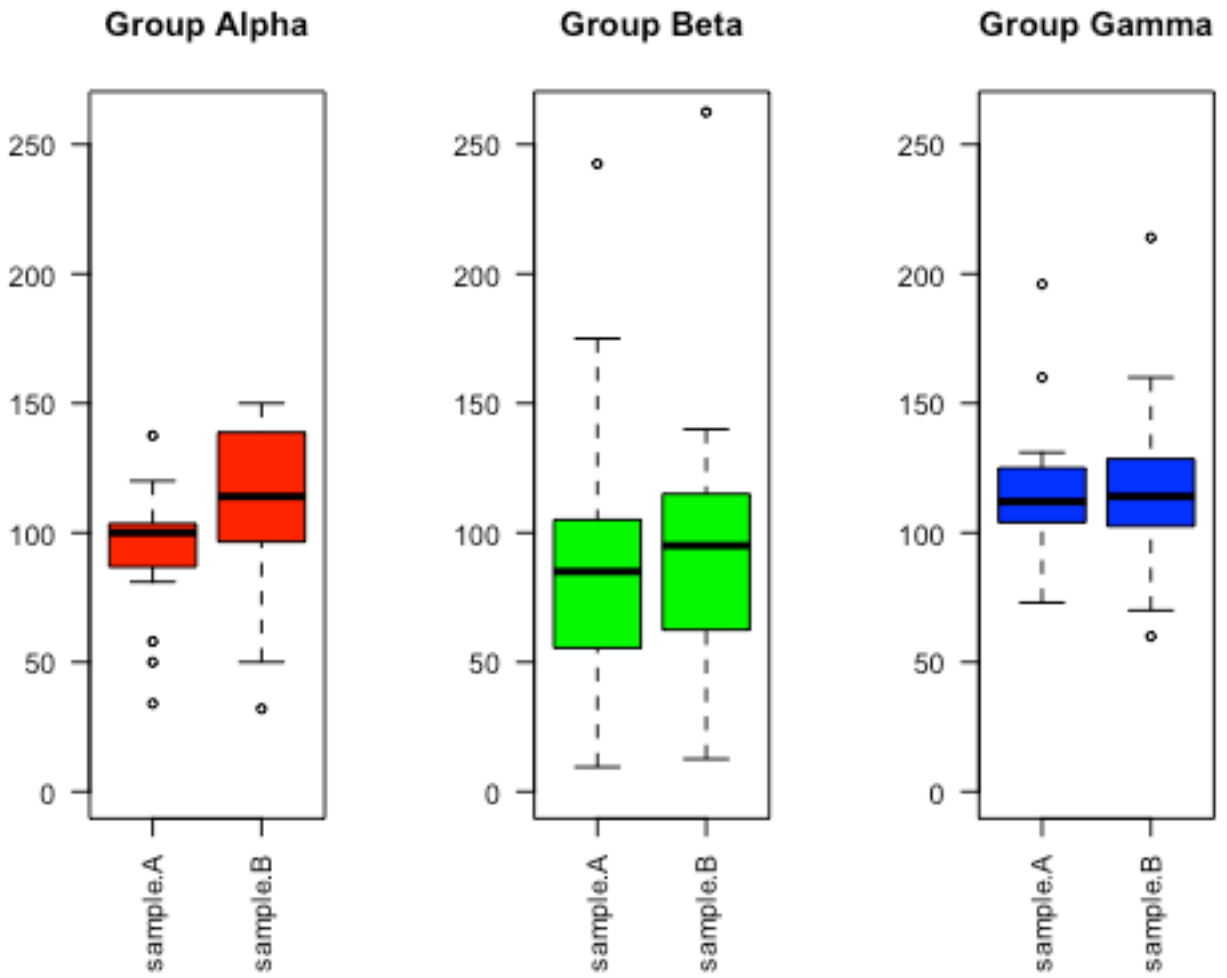
The University Of Sheffield.

Blood Data box plot in groups

```
par(mfrow=c(1,3))

boxplot(blood_data[blood_data$Group=="alpha",2:3],
        col='red', ylim=c(0,260),
        main="Group Alpha")

boxplot(blood_data[blood_data$Group=="beta",2:3],
        col='green', ylim=c(0,260),
        main="Group Beta")

boxplot(blood_data[blood_data$Group=="gamma",2:3],
        col='blue',ylim=c(0,260),
        main="Group Gamma")
```

The University Of Sheffield.

Blood Data box plot in groups



The University Of Sheffield.

Plotting figures

When building the plot we have paid attention to three main aspects:

- Data (DATA)
- Visual marks to represents the data points (MAPPING)
- Coordinate system (GEOMETRY FUNCTIONS)
  - Lines/curves
  - Box
  - Points

The data frame is always something that we cannot change and the markers often depends on the type of data and the size of the data frame.
We can add then another layer. This is how we represent the data with different shapes plots.

There is another way to build the plots, based of this concepts of layers.
This is what the package **ggplot2** does.

The University Of Sheffield.

Plotting with ggplot2

ggplot2 is a plotting package that simplifies to create complex plots from data in a data frame. It provides a way for specifying what variables to plot, how they are displayed and general visual properties.

Why ggplot2? It helps to create high quality plots with minimum amount of adjustments and tweaking. It makes it easier to change type of plots (i.e scatter plots or box plot) on same data.

The data needs to be in the a column for every variable and a row for every observation.

Disadvantages of using ggplot2

- You have to learn "the grammar" to use it well
- Vast package, can be intimidating
- More than one way to do things

The University Of Sheffield.

ggplot2 grammar

ggplot graphics are built step by step by adding new elements, like layers.

To build a ggplot, we will use the following basic template :

ggplot(data = <DATA>, mapping = aes(<MAPPINGS>)) + <GEOM_FUNCTION>()

The three main steps are:

1. Choose the data

2. Define a mapping (using the **aesthetic** (aes) function), by selecting the variables to be plotted and specifying how to present them in the graph, e.g. as x/y positions or characteristics such as size, shape, color, etc.

3. Add 'geoms' – graphical representations of the data in the plot (points, lines, bars).
   `geom_point()` for scatter plots, dot plots, etc.
   `geom_boxplot()` for, well, boxplots!
   `geom_line()` for trend lines, time series, etc.

To add a geom to the plot use the **+** operator.

The University Of Sheffield.

ggplot Example

```
library(ggplot2) # if not working use install.packages("ggplot2")
data("iris")

# set up the main structure
ggplot_iris <- ggplot(iris, aes(x = iris$Sepal.Length, y = iris
$Petal.Length))

# Add a layer using the point 'geom'…
ggplot_iris <- ggplot_iris + geom_point()


#Show the plot—just 'print' the object to the console
ggplot_iris
```

What about the colors of the Species?
Modify the aes adding the parameter color

```
ggplot_iris <- ggplot(iris, aes(x = iris$Sepal.Length, y = iris
$Petal.Length, color=iris$Species))
```

The University Of Sheffield.

ggplot Example

What about changing labels for the axis and legend title?

Add a new layer +labs()

```
Ggplot_iris <- ggplot_iris + geom_point() +
labs(x= "Sepal Length", y="Petal Length", color="Species")
```

Add a plot title

```
labs(x= "Sepal Length", y="Petal Length",
color="Species", title=" Iris Data")
```

```
labs(x= "Sepal Length", y="Petal Length",
color="Species", title=" Iris Data") +
theme(plot.title = element_text(hjust = 0.5))
```

Excerise

Repeat the same for the width of Sepal and Petal

The University Of Sheffield.

Before you go

Comment well your script.

Save your .R file

Close Rstudio

Log off