The University Of Sheffield.

# Introduction to Data Handling in R (BMS 109) Week6

The University Of Sheffield.

# Today you will learn

- Box plot with ggplot using iris data
- Potting blood data charts with ggplot
  - Scatter plot
  - Box plot
  - Bar Chart
  - Histograms
- Representing basic statistics with plotting
- Final layout of figures

The University Of Sheffield.

# Setting up

- Set you working directory
- Open a new script, name it and save it
- Prepare your work space to use ggplot2 with library()
- Bring the iris data set in your work space with data()
- Read the “Blood_osmosity .csv” data and assign it to a name. Use read.csv()

The University Of Sheffield.

# Recap of 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

what the package **ggplot2** is based of this concepts of layers to build plots.
We add layers to the basic template with the **+**

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

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

The University Of Sheffield.

# Box Plots

Box plots summarise the distributions of a variable using quartiles

Each boxplot shows us:

- the group median (horizontal bold line)
- the interquartile range ( the size of the box ")
- the range of the rest of the data (the vertical lines: "whiskers")
- the outliers

We can use the geom_ function geom_boxplot() to draw them in ggplot

```
ggplot_iris<-ggplot(iris, aes(x = Species, y = Petal.Length/Petal.Width,
color = Species)) +
  geom_boxplot() +
  labs(x = "Species", y = "Petal Eccentricty", color= "Species")

ggplot_iris
```

Change *color* with *fill* what happens?
What if we add theme_classic() as extra layer?

The University Of Sheffield.

# Splitting the print area

In ggplot we cannot use the classic approach with par() . We need to split the area using another package called *cowplot*

Install the package  install.packages("cowplot")
then run *library(cowplot)*

**Exercise:**

Calculate the eccentricity of Petals as before and of Sepals. Plot the two boxplots side by side using

```
plot_grid(ggplot_iris1, ggplot_iris2, labels = c("a)", "b)"))
```

The University Of Sheffield.

# Splitting the print area

```
ggplot_iris1<-ggplot(iris, aes(x = Species, y = Petal.Length/
Petal.Width, fill = Species)) +
  geom_boxplot() +
  labs(x = "Species", y = "Petal Eccentricty", fill= "Species") +
  theme_classic()

ggplot_iris2<-ggplot(iris, aes(x = Species, y = Sepal.Length/
Sepal.Width, fill=Species)) +
  geom_boxplot() +
  labs(x = "Species", y = "Sepal Eccentricty") +
  theme_classic()

plot_grid(ggplot_iris1, ggplot_iris2, labels = c("a)", "b)"))
```

You can use the same template to look at the blood data, with y=sample.A in one plot and y=sample.B in the other.

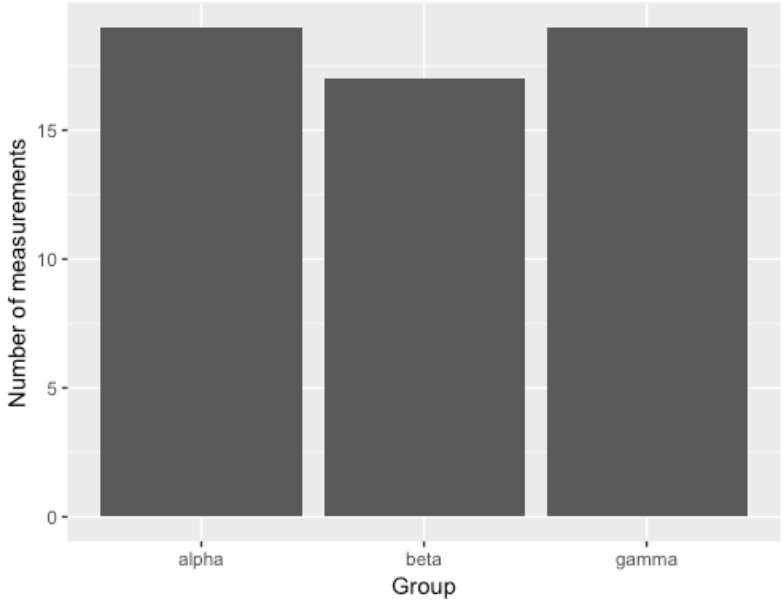
The University Of Sheffield.

# Bar plots

When we need to to summarise differences in summary statistics (like the mean) among groups, we use bar plots. They are particularly effective in grouped data.

In ggplot we can use geom_bar to make barplots. Note: it we use geom_bar on full data it will display the the number of observations in each group

```
ggplot(blood_data, aes(x = factor(Group))) +
  geom_bar() +
  labs(x = "Group", y = "Number of measurements")
```



The University Of Sheffield.

# Bar plots with group means

To display the means of the groups we first need to calculate them. We can do it in R using classic approach with :

```
sampleA.mean <- summarise(group_by(blood_data, Group),

mean.SampleA = mean(sample.A))
sampleB.mean <- summarise(group_by(blood_data, Group),

mean.SampleB = mean(sample.B))

mean.blood_data <- data.frame(blood_data$Group,
           sampleA.mean$mean.SampleA, sampleB.mean$mean.SampleB)
mean.blood_data
```

There is another way without using nested function. This is done with the *dplyr* methods that uses the concept of *chaining or pipes*

library(dplyr)
If not install.packages("dplyr")

The University Of Sheffield.

# Using dplyr

The dplyr package includes a special operator, called *the pipe*. To use this operator you need to combine % and > like this: **%>%.**

This allows us to avoid storing intermediate results or nesting functions.

If you remember the first example for the rounding a number after sqrt() we did:

```
x<- 5
x<-sqrt(5)
x<- round(x,2)

x<-round(sqrt(5),2)

5 %>% sqrt() %>% round(2)
```

The University Of Sheffield.

# Bar plots with group means - dplyr

```
# step 1 calculate the means
bdA_stats <-
  blood_data %>%
  group_by(Group) %>%
  summarise(mean_bdA = mean(sample.A))

# step 2 plot the data

bd_sA<-ggplot(bdA_stats, aes(x = Group, y = mean_bdA)) +
  geom_col(fill="lightblue") +
  labs(y = "Blood Osmolarity (mg)", title = "Sample A")+
  theme(plot.title = element_text(hjust = 0.5))+
  theme_classic()
```

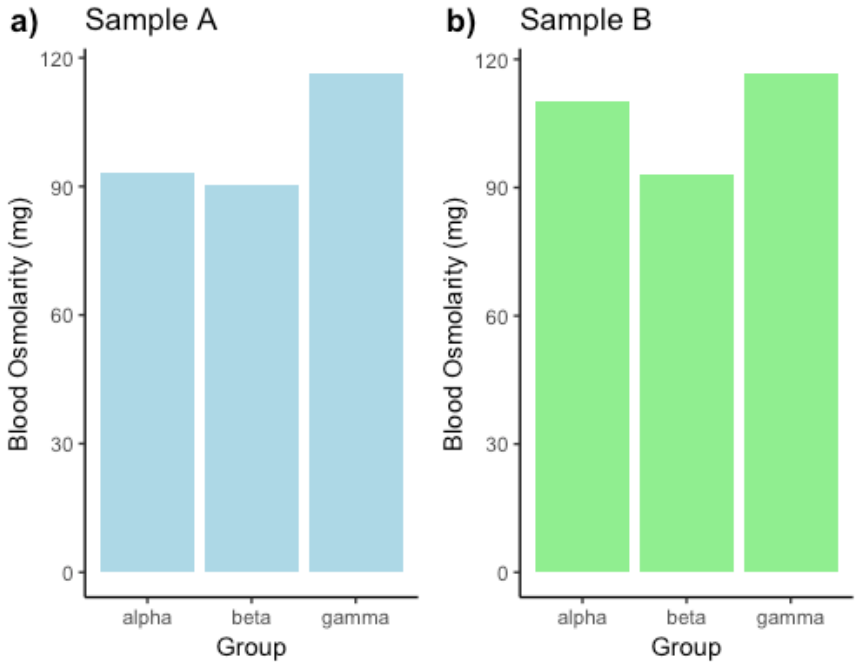
**Exercise:**
**do the same with sample B an use plot them side by side**

# Bar plots with group means – dplyr (cont…)

```
bdB_stats <-
  blood_data %>%
  group_by(Group) %>%
  summarise(mean_bdB = mean(sample.B))

# step 2
bd_sB<-ggplot(bdB_stats, aes(x = Group, y = mean_bdB)) +
  geom_col(fill="lightgreen") +
  labs(y = "Blood Osmolarity (mg)", title = "Sample B")+
  theme_classic()

plot_grid(bd_sA, bd_sB, labels = c("a)", "b)"))
```



The University Of Sheffield.

# Adding Error bars to plots

When we describe data using descriptive statistics like the mean, we would always need to show how these data is dispersed around the mean.
This is by adding the Standard error to the plots.

The standard error is one option here:

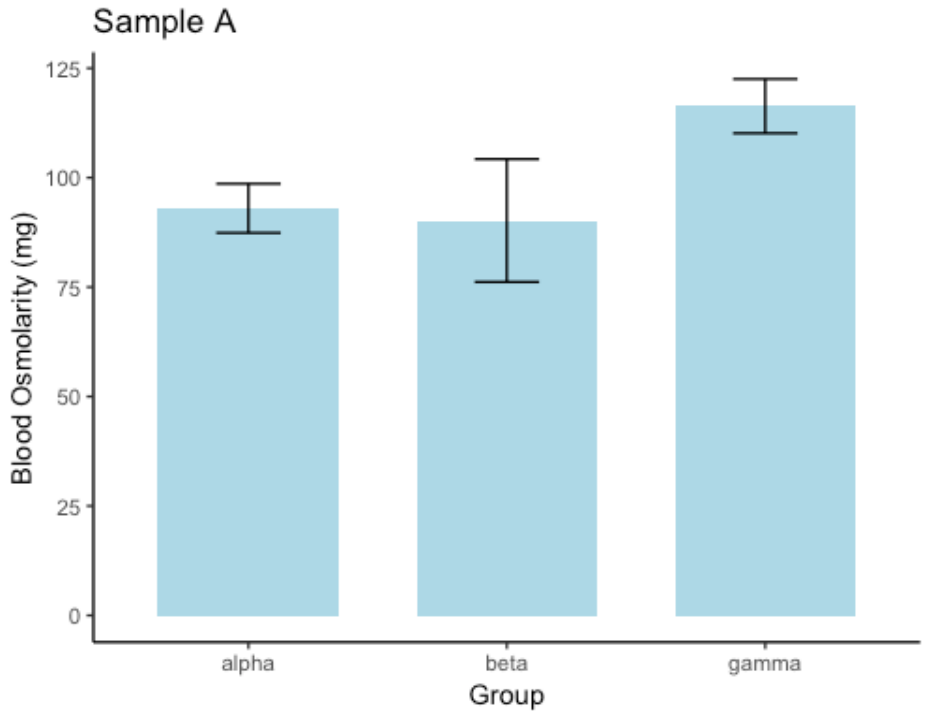
$$\text{Standard Error} = \frac{\text{Standard Deviation}}{\sqrt{\text{Sample Size}}}$$

We need to include a calculation of the standard errors along with the means in our chain:

```
bdA_stats <-
  blood_data %>%
  group_by(Group) %>%
  summarise(mean_bdA = mean(sample.A),
            se = sd(sample.A) / sqrt(n())) # <- New calculation
```

# Adding Error bars to plots (cont…)

```
bd_sA<-ggplot(bdA_stats,
       aes(x = Group, y = mean_bdA,
           ymin = mean_bdA - se, ymax = mean_bdA + se)) +
  geom_col(fill = "lightblue", width = 0.7) +
  geom_errorbar(width = 0.25) +
  labs(y = "Blood Osmolarity (mg)", title = "Sample A")+
  theme_classic()
```



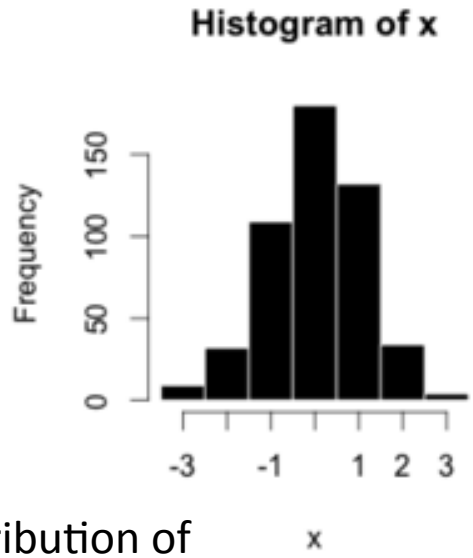

**Exercise:**
**Repeat the previous exercise WITH SAMPLE B, add the error bars to the plots and print side by side.**

The University Of Sheffield.

# Histograms

It is an estimate of the *probability distribution* of a continuous variable (quantitative variable) and was first introduced by Karl Pearson

To estimate this **distribution** we proceed in the same way as the bar chart but we first grouping the observations. This consists in choosing a set of contiguous non-overlapping intervals, called class intervals (or bins), the observations can be grouped to form a discrete variable from the continuous variable.




Histograms give a rough sense of the *density* of the underlying distribution of the data and often are used as density estimators ( to estimate *the probability density function* of the underlying variable)

The University Of Sheffield.

# Plotting histograms for blood data

```
######### histograms with density ############
# semi transparent SAMPLEA
pA<-ggplot(blood_data, aes(x=sample.A, fill=Group, color=Group)) +
  geom_histogram(aes(y= ..density..), alpha=0.5, binwidth = 5)+
  theme(axis.text.x = element_text(size= 10, angle = 90))+
  geom_density(alpha=.2, fill="#FF6666")+  # Overlay with transparent
density plot
  labs(x="Osmolarity", title="Sample A")+
  facet_grid(Group ~ .)

pB<-ggplot(blood_data, aes(x=sample.B, fill=Group, color=Group)) +
  geom_histogram(aes(y= ..density..), alpha=0.5, binwidth = 5)+
  theme(axis.text.x = element_text(size= 10, angle = 90))+
  geom_density(alpha=.2, fill="#FF6666")+  # Overlay with transparent
density plot
  labs(x="Osmolarity", title="Sample B")+
  theme(axis.text.x = element_text(angle = 90))+
  facet_grid(Group ~ .)

plot_grid(pA, pB, labels = c("a)", "b)"))
```

The University Of Sheffield.

# Dots and lines plots

Some times it is effective to be able to see a trend in a time series data. This can be achieved using a dot-line plot

```
data(ChickWeight)

## Calculate the mean and standard errors for each diet at each time point
pltdata<- group_by(ChickWeight, Time, Diet) %>%
  summarise(mn = mean(weight), se = sd(weight)/sqrt(n()))

## Plot the means over time - remembering to colour by the diet
plta <- ggplot(pltdata, aes(x=Time, y = mn, colour = Diet)) +
  geom_point() +
  geom_line() + ## function for adding lines to our plot
  theme_classic() +
  labs(y = "Mean weight (g)", x = "Time (days)")
plta
```

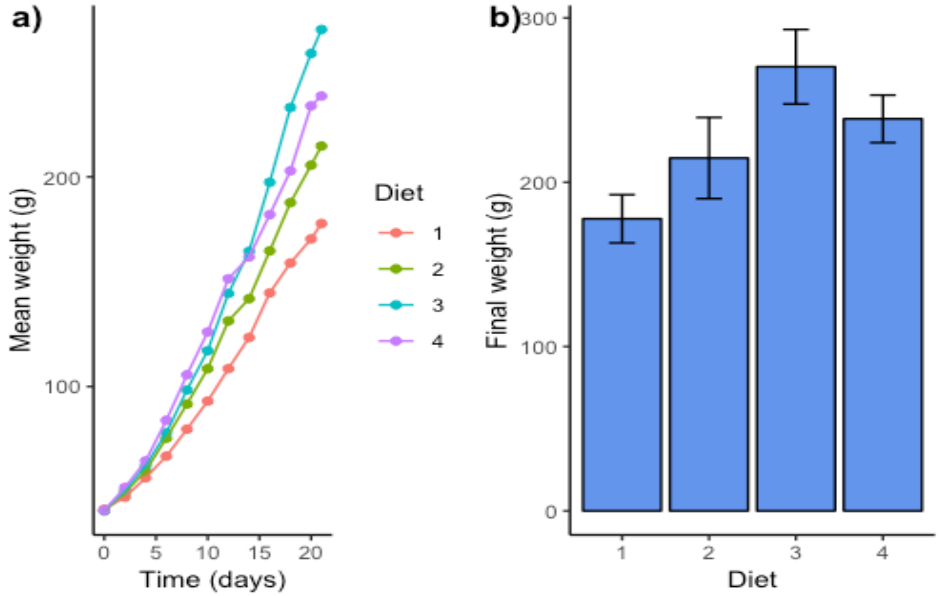
```
## Filter the summary data to only include the final weights
pltdata2 <- ungroup(pltdata) %>%
  filter(Time==max(Time))
pltdata2

## Make a bar plot of the means and standard errors
pltb <- ggplot(pltdata2, aes(x=Diet, y = mn, ymin = mn-se, ymax = mn+se)) +
  geom_col(fill = 'cornflowerblue', colour = "black") +
  geom_errorbar(width = 0.3) +
  labs(y = "Final weight (g)") +
  theme_classic()
pltb
```

**Exercise**

**Print everything side by side and discuss it**

The University Of Sheffield.




- Each plot helps to visualise different aspect of the data
- A good plot depends on good data
- Interpreting well your plot depends on how well you know the underlying concepts of the plots you used
- Choose the correct plot not just the one that looks better

**WELL DONE FOR GETTING HERE !!!**