Lab 3 – Exploratory Graphics with GGplot2

Questions? montwe@ualberta.ca or isaacren@ualberta.ca

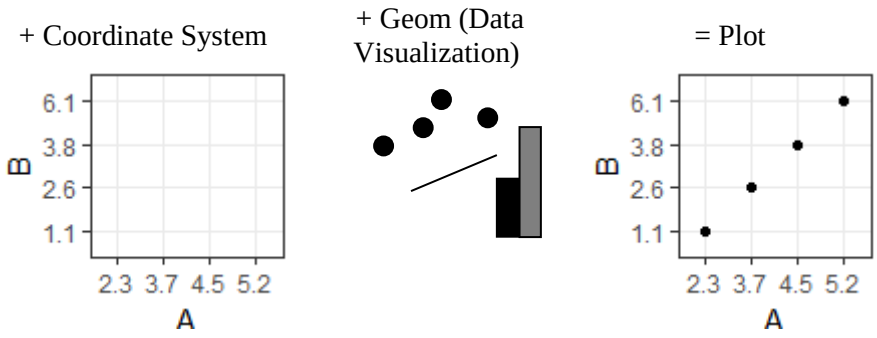
1. Introduction

In this lab, we will learn how to do basic exploratory graphs with the ggplot2 package. The "gg" stands for "grammar of graphics" because the package was developed to produce figures that are visually appealing and conform to sound graphical principles.

As with all graphs, the same three components are required: a data set, a coordinate system (x,y-coordinates) and a visual representation of data points (so called geoms in ggplot2). Here is a simple illustration of these three components:

Data

A	B	C	D
5.2	6.1	1.5	X
4.5	3.8	2.0	X
3.7	2.6	1.4	Y
2.3	1.1	0.8	Z



For plotting with the ggplot2 package, data should be in wide format if you have two variables (e.g. to produce a simple scatterplot), but long format is usually best if you have 3 or more variables (e.g. for more complex or multivariate plots).

The coordinate system and geoms must be specified in ggplot2-specific syntax. This 'grammar' is usually more efficient than the basic graphic functions of R when creating complicated figures because several key components are developed automatically and elegantly. For example, legends are produced automatically in ggplot2 if the data is in the appropriate format.

Ggplot2 syntax also requires a + symbol after each line of code before another line of code. This signals to the package that more code will follow. Therefore, all of the lines of code to produce a figure have a + symbol except the last line. Forgetting the + symbol is a common source of error.

Bringing these three principles together, the basic syntax therefore looks like this (*italics* mean user input, **bold** indicates mandatory commands that specify data, coordinate system and geom):

fig1 = **ggplot(data=*data*, mapping = aes(*mappings*)*) +**
geom_*function*(stat = *stat*, position = *position* **) +**
coordinate_*function*() +
facet_*function*() +
scale_*function*() +
theme_*function*()
fig1

* aesthetics (aes) can be specified globally in the ggplot() command or specified in geom_function command. The aesthetics mapping function specifies what and how data is displayed.

Great resources to learn about the various ggplot2 functions can be found here:

- http://docs.ggplot2.org/current/
- http://www.cookbook-r.com/Graphs/
- https://www.rstudio.com/wp-content/uploads/2016/11/ggplot2-cheatsheet-2.1.pdf.

If you are interested in learning more about the philosophy behind ggplot2, read Hadley Wickam's publication *A layered grammar of graphics*:

- http://vita.had.co.nz/papers/layered-grammar.pdf.

In this lab, you will learn about three types of graphs that can be used to visualize raw data and check for errors, outliers, and the nature of distributions and correlations: histograms, scatter plots, and boxplots. In the final presentation and discussion of quantitative data in your project, publication, or thesis, you will rarely use these tools. Instead, you will generate similar graphs based on summary statistics. However, the summary statistics tend to hide errors, outliers, and odd distributions or relationships among variables. So, it's good to start any data exploration with these three graphs.

To produce these graphs, we will first install the new package: ggplot2.

```
install.packages("ggplot2")
```

The library function activates the package:

```
library(ggplot2)
```

R conveniently comes with several datasets that you can use to practice statistical analyses or to practice graphical illustrations. We will work with the *iris* dataset in this lab. Iris is the genus of a flowering plant and the dataset includes measurement of several features of the flower: Petal length and width and sepal length and width. In addition, there is also a categorical variable of naming three species. To move the dataset into your environment, type:

```
iris=iris
head(iris)
```

Now we will produce a histogram of one variable, Petal.Length. To see how it is constructed, first create a plot with ggplot()

```
fig1 = ggplot()
fig1
```

When you call plot, it is empty because we have not yet specified a coordinate system or the data we want to plot. By referring to your data source in the ggplot() function, ggplot2 estimates a meaningful coordinate system automatically:

```
fig1 = ggplot(data = iris, mapping= aes(x = Petal.Length))
fig1
```

The plot is still empty, but now it has an axis – the start of a coordinate system. Now let's start adding some data using different functions for data visualization.

2. Histograms

We use the + sign to add a first geom (representation of the data). We will look at a histogram first to understand the data frequencies:

```
fig1 =ggplot(data = iris, mapping = aes(x = Petal.Length))+
  geom_histogram(bins = 30) # you can modify the number of bins
fig1
```

Based on this histogram, what can you tell about the distribution of Petal.Length?

By telling ggplot2 to shade the bars by the Species-column, we can illustrate why the distribution is scattered this way:

```
fig1=ggplot(data = iris, mapping = aes(x = Petal.Length, fill = Species))+
      geom_histogram(bins=30)
fig1
```

If we want to plot the distribution of each species separately, we can subset in the ggplot2 function as per usual:

```
fig1=ggplot (data = iris[iris$Species == "versicolor",], mapping = aes(x =
      Petal.Length, fill=Species)) +
      geom_histogram(bins=30)
fig1
```

Instead of “fill” you can also define “col”, this will only change the color of the frame of the bars. You will notice that ggplot2 automatically created a legend for us. This means we are all done here. If you have time, produce the same figures for Petal.Width, Sepal.Width and Sepal.Length.

3. Scatter plots

With scatter plots you can visually check the relationships among variables. Are they linear or curvilinear? Outliers are also easily visible.

Let's produce a scatter plot. Sepal length and width will serve as x-y-coordinates. Basically, we plot x as a function of y. We will add a geom_point-layer to show the data as dots:

```
fig1= ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))+
  geom_point()
fig1
```

There might be points plotted over each other, because they share the same x-y coordinates. We can add random noise to the data with geom_jitter() so that we can better see all the data at once:

```
fig1= ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))+
  geom_jitter()
fig1
```

Another way is to visualize all the data at once is to add transparency to the points with the alpha-command:

```
fig1= ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))+
  geom_point(alpha=0.2)
fig1
```

To identify certain data points, we can add labels with geom_text(). Hjust and vjust allows controlling the placement of labels. First, we have to create an ID-column:

```
iris$ID =rownames(iris)

fig1= ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))+
  geom_point() +
  geom_text(aes(label=ID),hjust=0, vjust=0)
fig1
```

The points in the plot can be shaded by species, just as we did with the bars in the histograms:

```
fig1= ggplot(data=iris, aes(x=Sepal.Length, y=Sepal.Width, col=Species))+
  geom_jitter()
fig1
```

If you have time: Try replacing col=Species with shape=Species. Create plots for the other variables.

In case of scatter plots, an R-base function offers an efficient alternative:

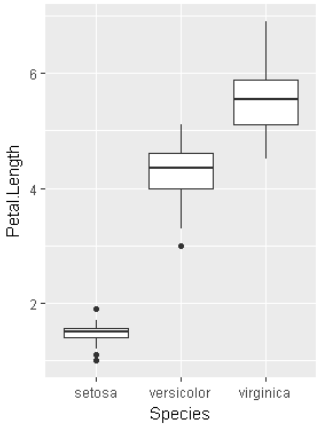
```
pairs(iris[,1:4])
```

The shading of points by Species can be set with the col= command.

```
pairs(iris[,1:4], col=rainbow(3)[iris$Species])
```

4. Box plots

Boxplots were invented by John Tukey (1915-2000). He was a bit of a rebel against the establishment in the field of statistics. He started the whole idea of visual data analysis as an alternative to inferential statistics at the time when computers were just powerful enough to make this possible.



A boxplots is based on quartiles (intervals). Quartiles are ¼ intervals and percentiles are $^{1}/_{100}$ intervals. The 50th percentile means that 50% of the data are above and 50% below this value, also called the median and displayed as a line through the box. The first quartile (Q1, 25%) and the third quartile (Q3, 75%) make up the box. The whiskers of the boxplot indicate the range of the observations excluding outliers.

Tukey subjectively defines “moderate outliers” as values that are beyond 1.5*(Q3-Q1) from the upper or lower quartile (box). “Far outliers” are defined as 3*(Q3-Q1) beyond the interquartile range. The outliers are also described according to “Tukey’s Inner Fence” (1.5*) or “Outer Fence” (3*) criteria.

The code for drawing the basic boxplot is simple:

```
fig2 =ggplot(iris, aes(x = Species,y = Petal.Length))+
  geom_boxplot()
fig2
```

Let's say we want to show the entire *iris*-dataset in one boxplot figure. Before we can do this, we have to re-arrange the data frame so that all the measurements are in one column. To do so, we first have to install a new package:

```
install.packages("tidyr")
library(tidyr)
```

This package has clean and straight-forward functions to convert from wide to long and back. Together with ggplot2, it is part of the so-called "tidyverse", an attempt to standardize the R-syntax.

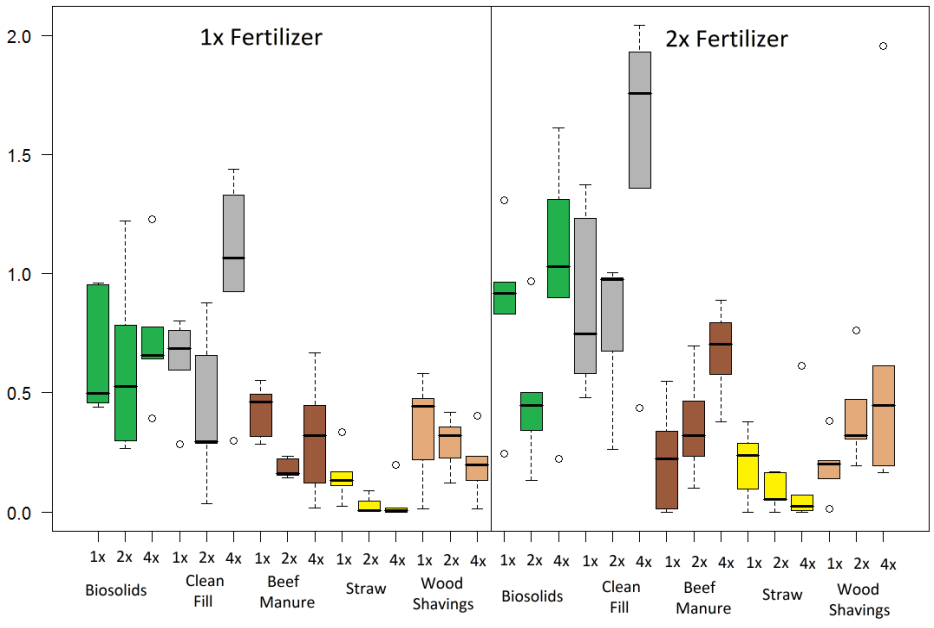
```
iris_long = gather(iris, # our dataset
  FlowerPart,  # a new column containing the old column names
  measurement, # a new column for the measurements
  Sepal.Length:Petal.Width) # the columns to gather
head(iris_long)
```

Now we can plot this new data frame:

```
fig2 = ggplot(iris_long, aes(x = FlowerPart, y = measurement))+
  geom_boxplot(aes(fill = Species))
fig2
```

What conclusions can you draw from this figure?

Below is a different example for inspiration if your own data is based on a factorial sampling or experimental design. This factorial dataset from a reclamation study (Fertilizer Level, Amendment, Amendment Level) is an excellent example for obtaining a complete understanding for the nature of a complicated dataset: (1) treatment effects, (2) interactions, (3) outliers, (4) normality, and (5) homogeneity of variances. Nothing in the subsequent analysis can surprise you anymore, and if it does, the analysis is likely wrong or violates important assumptions.



Informal assignment:

With your lentil dataset from lab 2:
1) create a boxplot to visualize the differences in the varieties and interactions
2) create a histogram

Report what you found out about the lentil dataset to me or the TA in class. This does not need to be turned in.