# Lab 5 – Customized Scientific Graphs

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

In lab 3, you learned about exploratory graphics to assess underlying relationships and outliers in data. In lab 4, you learned more about managing data in R and Excel. In this lab, we'll cover how to create customized scatterplots with a few additional commands (customizing boxplots and line graphs are in the next lab). Combined, these labs can be used to develop high-quality figures that can be used in publications and in the RenR 711 course project.

## 5.1. Import data and prepare a scatter plot

Scatterplots were covered in the exploratory graphics lab, but can also be used to present results.

To get started, create a "Lab 5" folder. Enter from below or download the scatter dataset from the class website (scatter.csv), set your working directory, import your data into R and name it "dat1":

| ID | SPEC | ECOSYS | DBH | VOL | DENSITY | AGE |
|---|---|---|---|---|---|---|
| 1 | Spec1 | Ecosys3 | 11.5 | 1.09 | 0.55 | 23 |
| 2 | Spec2 | Ecosys1 | 5.5 | 0.52 | 0.74 | 24 |
| 3 | Spec1 | Ecosys3 | 11 | 1.05 | 0.56 | 27 |
| 4 | Spec2 | Ecosys2 | 7.6 | 0.71 | 0.71 | 23 |
| 5 | Spec2 | Ecosys3 | 10 | 0.95 | 0.63 | 22 |
| 6 | Spec1 | Ecosys2 | 8.4 | 0.78 | 0.63 | 29 |
| 7 | Spec2 | Ecosys2 | 8.4 | 0.77 | 0.64 | 21 |
| 8 | Spec1 | Ecosys2 | 9 | 0.87 | 0.6 | 27 |

```
setwd("C:/YourPathHere/Lab5Folder")
dat1 = read.csv("scatter.csv")
head(dat1)
```

To create a basic scatterplot in R, you can use basic R functionality:

```
plot(dat1$DENSITY ~ dat1$VOL)
```

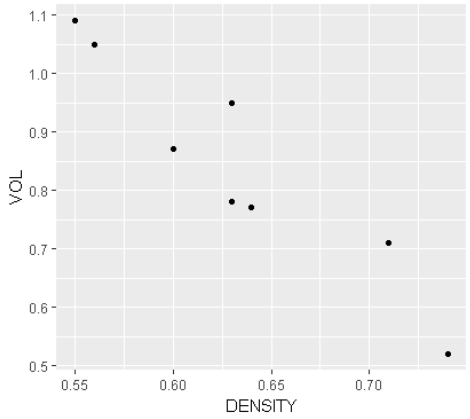
The output is ok for a quick check of the data, but customizing the plot requires extensive coding that may not be efficient. As we discussed in lecture and saw in lab 3, there is an easily-customizable alternative available offered through ggplot2. In lab 3, we used two geoms to create scatterplots:

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

Fig1_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point()
Fig1_Scatter

Fig2_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_jitter()
Fig2_Scatter
```

Here is the output of the scatter plot. You can see that the two variables have a negative relationship.



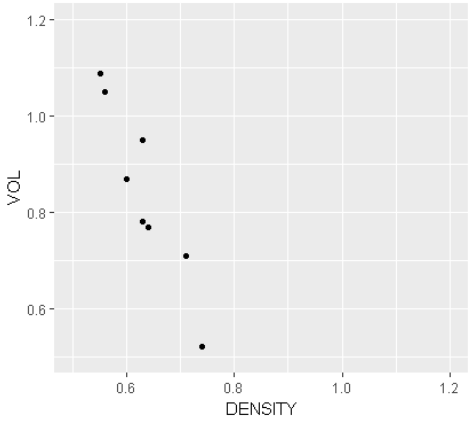

## 5.2 Modifying Axes and High Ink-to-Data Ratios

ggplot2 does a good job at automatically adjusting the range of the coordinate system as well as font and dot size according to the defined width and height of the plot (the base R functions do not provide these automated defaults). However, we still might want to modify features of the graph for publication. For example, the x-axis is a little bit odd, going from 0.55 to 0.70.

Let's therefore start by modifying the axes using ggplot2 syntax. To modify the range of the coordinate system in the display, you can use `coord_cartesian()`. For example:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,1.2))
Fig3_Scatter
```

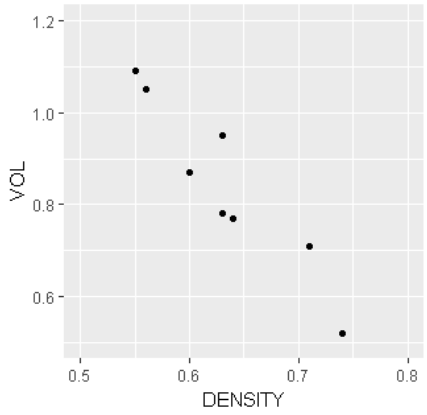
Below is the new output. As you can see, the axes have been modified to the specifications in the third line of code above. The code worked well, but the defined axis specifications are not great. There is a lot of empty space in the plot now. No journal has space to publish figures with such low data density. Always aim for high ink-to-data ratios.




Instead, let's aim for a nice clean axis that makes sense and provides high data density.

Try:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8))
Fig3_Scatter
```




Now data density and axes look reasonable. You might also want to modify the tick marks of the axes. If you would like to modify where the axes are, you need a new function which depends on the data type. Here, both scales are continuous, so we use `scale_x_continuous()` and `scale_y_continuous()` instead of `scale_x_discrete()` or `scale_y_discrete()`.

The function `seq()` creates a sequence of numbers between a user-specified minimum and max by a defined break. This is useful because we don't have to type all the values.

```
Fig4_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.3))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.2))
Fig4_Scatter
```

Depending on taste, the axes may now look a bit too sparse. Can you find a setting that works better?

The next thing to modify are the axis-labels, and we can do this with the `lab()` functions:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")
Fig3_Scatter
```

## 5.3 Modifying Titles

Next, we might want a title:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
```

```
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density")
Fig3_Scatter
```

To center the title, we need to change the `hjust` setting in the `theme()` function: hjust stands for horizontal adjustment.

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5))
Fig3_Scatter
```

If the title is too long, it can be split by inserting \n command into the text string. Change your code to: `ggtitle("Tradeoff between volume\n and wood density")`

The theme-function can also be used to modify the font size of the title:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))
Fig3_Scatter
```

## 5.4 Annotating the Plot

The annotate function is useful for including sample size or $R^2$-values in your plot. Please try to find a better spot for the annotation by modifying the x and y values.

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

## 5.5 Changing Colours

Next, we will change colors of the points. If you want to change the color of *all* points, you can modify the `geom_point()` function.

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point(col="red") +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

R understands many colors by their common name. Try blue, green, yellow, gold and darkgrey (must be in quotation marks). More colour names can be found here:
http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

R also recognizes 'hexadecimal' colours, which represent blends of red, green and blue. Red, green and blue colours are each represented by two digits composed of either a number or letter, where 0 represents nothing and F represents full saturation. For example, "#FF00FF" will represent the full value for red, no green and full value for blue (=pink).

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point(col="#FF00FF") +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

You can usually find the Hex colour in Illustrator, Photoshop, Paint or even just on Google. However, here is another handy resource:
https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf

Color Brewer is another fantastic resource. It provides good colour schemes that can accommodate colour-blindness. This is a good thing to keep in mind when publishing graphs and maps.
http://colorbrewer2.org/#type=sequential&scheme=BuGn&n=3

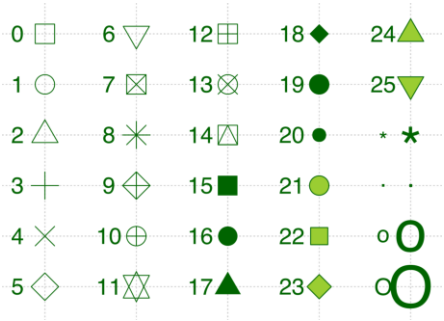
## 5.6 Changing Point Size and Shape

The shape of the points can also be changed. Each shape is associated with a number in R. Some symbols are solid in color (e.g., see numbers 15-20 in figure below). The colour of solid symbols is set with the `col` command. Other symbols have an outline and a fill. For these symbols (e.g. 21-25), `col` specifies the outline, and `fill` specifies the inside color.

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
```

```
  geom_point(col="red", fill="black", shape=21) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

Try out some more:




The size of the symbols can also be modified:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL))+
  geom_point(col="red",fill="black",shape=21,size=5) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

To increase the data-to-ink ratio, the colours or symbology can be used to specify another variable:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS, shape=SPEC))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.6, y = 0.9, label = "n=8")
Fig3_Scatter
```

## 5.7 Fully Customized Point Shape, Colour and Sizes

Now that you have specified that the colour and shape should illustrate a variable, you can further customize the colour and shape according to your tastes with the `scale_color_manual()` and `scale_shape_manual()` commands.

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS,
shape=SPEC,size=AGE))+
  geom_point() +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  ggtitle("Tradeoff between volume and wood density") +
  theme(plot.title = element_text(hjust = 0.5,size=10))+
  annotate("text", x = 0.5, y = 1.2, label = "n=8")+
  scale_color_manual(values=c("#999999", "#E69F00", "#56B4E9"))+
  scale_shape_manual(values=c(15, 16))
Fig3_Scatter
```

This figure is demonstrating some of the customizations possible with ggplot2. However, it also looks messy now. Let's clean it up a bit:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS))+
  geom_point(size=4) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")
Fig3_Scatter
```

## 5.8 Legend Placement

The legend can be moved anywhere you like. We have to specify this in the `theme()` function. Let's begin with specifying "bottom", "top", "left", or "right".

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS))+
  geom_point(size=4) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  theme(legend.position = "bottom")
Fig3_Scatter
```

Another way is to x-y coordinates:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS))+
  geom_point(size=4) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
```

```
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  theme(legend.position = c(0.8, 0.7))
```

## 5.9 ggplot2 Themes

You may not like the standard ggplot2 theme with its grey background. Because of this, the package comes with several different themes that can be added to the plot as per usual:

```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS))+
  geom_point(size=4) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  theme_bw()+
  theme(legend.position = c(0.8, 0.7))
Fig3_Scatter
```

Note that `theme(legend.position = c(0.8, 0.7))` must come after the `theme_bw()` command because it will otherwise overwrite the `theme()` specifications.

Also try: `theme_classic()`, `theme_minimal()`. There is also an additional package with even more different themes: "ggthemes". If you have time, install the package and explore the options.

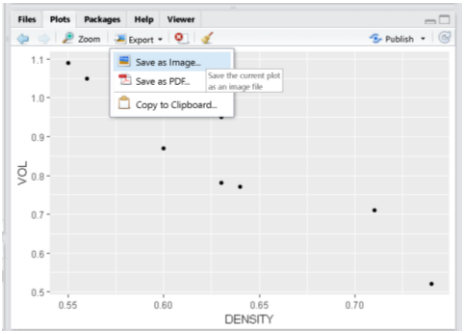
## 5.10 Flipping Coordinate Systems

It's sometimes useful to flip the plot. This can be done with the `coord_flip()functio`

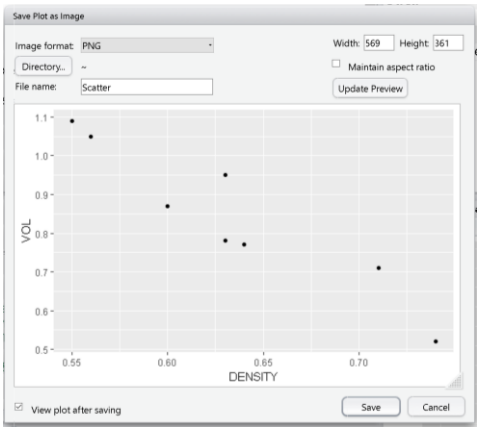
```
Fig3_Scatter = ggplot(dat1, aes(x=DENSITY,y=VOL,col=ECOSYS))+
  geom_point(size=4) +
  coord_cartesian(ylim=c(0.5,1.2),xlim=c(0.5,0.8)) +
  scale_y_continuous(breaks=seq(0.5, 1.2, 0.2))+
  scale_x_continuous(breaks=seq(0.5, 0.8, 0.1))+
  xlab("Wood density (kg/m³)")+
  ylab("Volume (m³)")+
  theme_bw()+
  theme(legend.position = c(0.8, 0.7)) +
  coord_flip()
Fig3_Scatter
```

## 5.11 Exporting Figures

There are several ways to export a graph. The probably easiest way is to click on export – save as image in the Plot tab of RStudio:




In the next dialog, you can choose an image format, adjust the width and height of the figure, define a file name and save the file. We will cover the advantages and disadvantages of image file types in another lab.




You can also right-click on the graph and choose "copy file" then paste it directly into Word or Powerpoint.

A better way to save a graph is to use the `ggsave()` function, which allows you to specify the width and height of the final plot in inches:

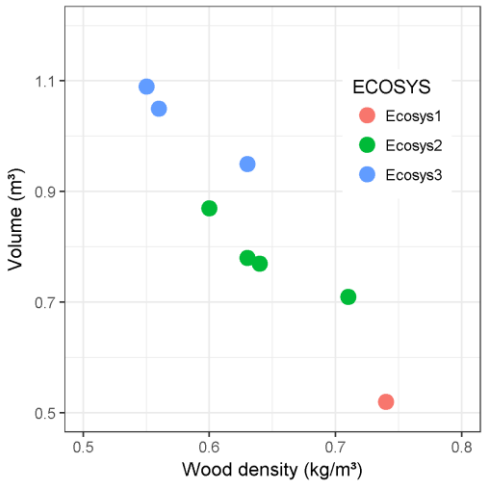
```
ggsave(Fig3_Scatter, file="Fig3.png", width=4, height=4)
```

The file should show up in your folder. You can also save the file in different formats, e.g. pdf:

```
ggsave(Fig3_Scatter, file="Fig3.pdf", width=4, height=4, useDingbats=F)
```

The "`useDingbats=F`" specification is important if you want to edit the pdf in a vector editor such as Inkscape or Adobe Illustrator, because otherwise the symbols might get imported as letters or numbers.

End result on next page.


ECOSYS
Ecosys1
Ecosys2
Ecosys3
Volume (m³)
1.1
0.9
0.7
0.5
0.5
0.6
0.7
0.8
Wood density (kg/m³)