Introduction to ggplot2

Dave Millar

dave.millar@uwyo.edu

R for Ecologists

10/25/17

This R notebook contains a brief overview of ggplot2, using several examples with sample data.

First of all, let's clear everything from memory and load the relevant libraries. See the tidyverse for additional information.

```r
rm(list=ls())

library(ggplot2)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Now, let's create some data to work with. For data visualization using ggplot2, your data needs to be in a *data frame* that is in *long format*. This will become relevant in the examples below where we plot multiple datasets on the same graph.

The first examples here are applicable for line/scatter/time series type plots.

Creating some data. . . .

Here is a simple function for calculating percent loss conductance (PLC) for various woody species using xylem water potential (MPa) as a driving variable.

```r
PLC_func <- function (psi,c,d) {

  # psi = xylem water potential
  # c,d = regression coefficients

  PLC <- 100*(1-exp(-1*(-psi/d)^c))

  return(PLC)

}
```

Next we'll create a range xylem water potentials to serve as model input.

```r
psi_input <- seq(-14,0,0.25)
```

Here, we pull some c and d parameter estimates for four species from the following citation:

Sperry, J. S., F. R. Adler, G. S. Campbell, and J. P. Comstock (1998), Limitation of plant water use by rhizosphere and xylem conductance: results from a model, Plant, Cell Environ., 21, 347-359.

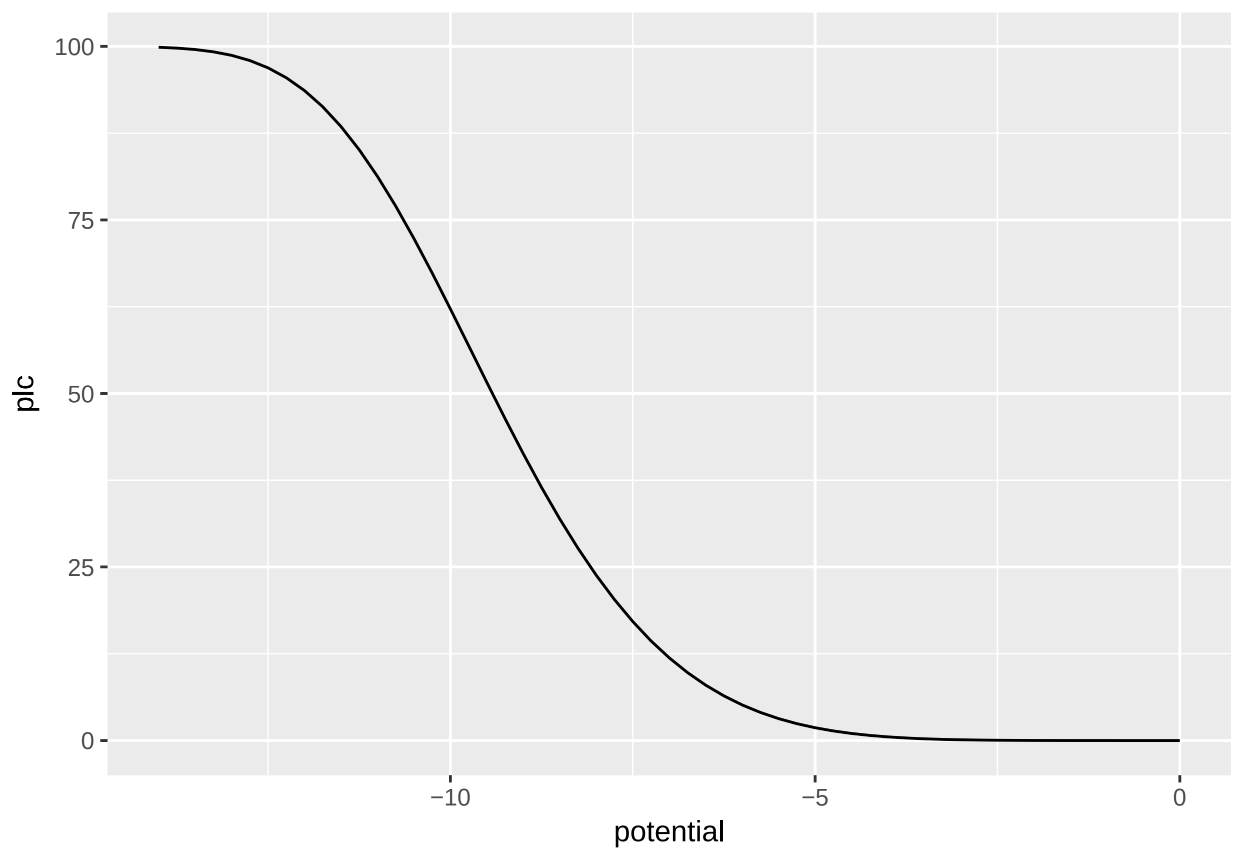
```r
d_ceanothus <- 10.05
c_ceanothus <- 5.71

d_sage <- 3.54
c_sage <- 2.64

d_boxelder <- 2.15
c_boxelder <- 3.43

d_birch <- 1.28
c_birch <- 9.53
```

Run the model for each species using the same driving variable (xylem water potential).

```r
ceanothus_data <- PLC_func(psi_input,c_ceanothus,d_ceanothus)
sage_data <- PLC_func(psi_input,c_sage,d_sage)
boxelder_data <- PLC_func(psi_input,c_boxelder,d_boxelder)
birch_data <- PLC_func(psi_input,c_birch,d_birch)
```

Create a line graph for the Ceanothus PLC curve.

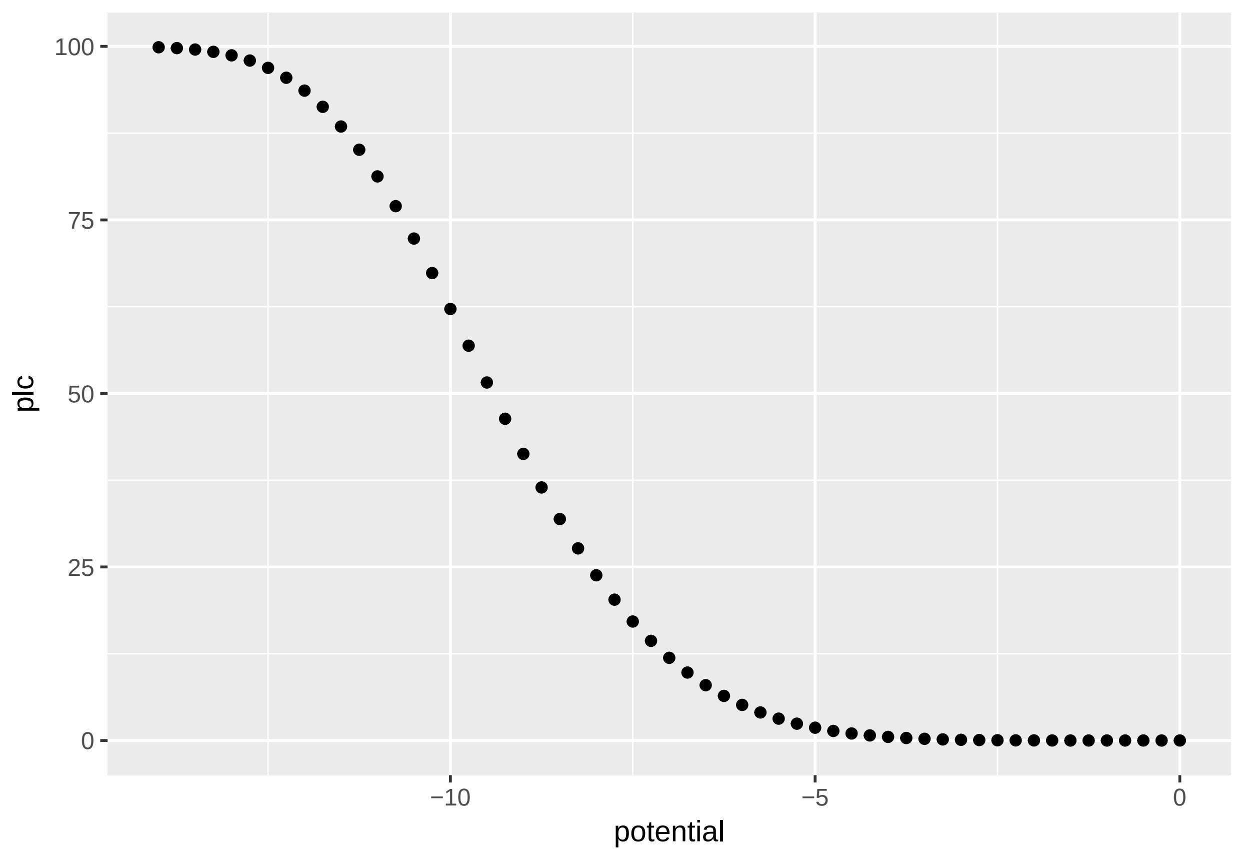
```r
cean_plot_data <- cbind.data.frame(potential=psi_input,plc=ceanothus_data)

ceanothus_plot <- ggplot(cean_plot_data, aes(x=potential,y=plc)) +
                     geom_line()

ceanothus_plot
```



In the above example, we use the **ggplot** function to get the ball rolling with our first plot. The first piece of this function is the data frame we want to plot, in this case *cean_plot_data*. After placing a comma, this is followed by the **aes** function which is where we are doing the aesthetic mapping. Here, we're saying that the x variables for this plot are in the *potential* column, and the y variables are in the *plc* column. For more complex graphs, like the ones described further below, there are other aesthetics that can be mapped for additional types of variables (color, fill, size, etc.).

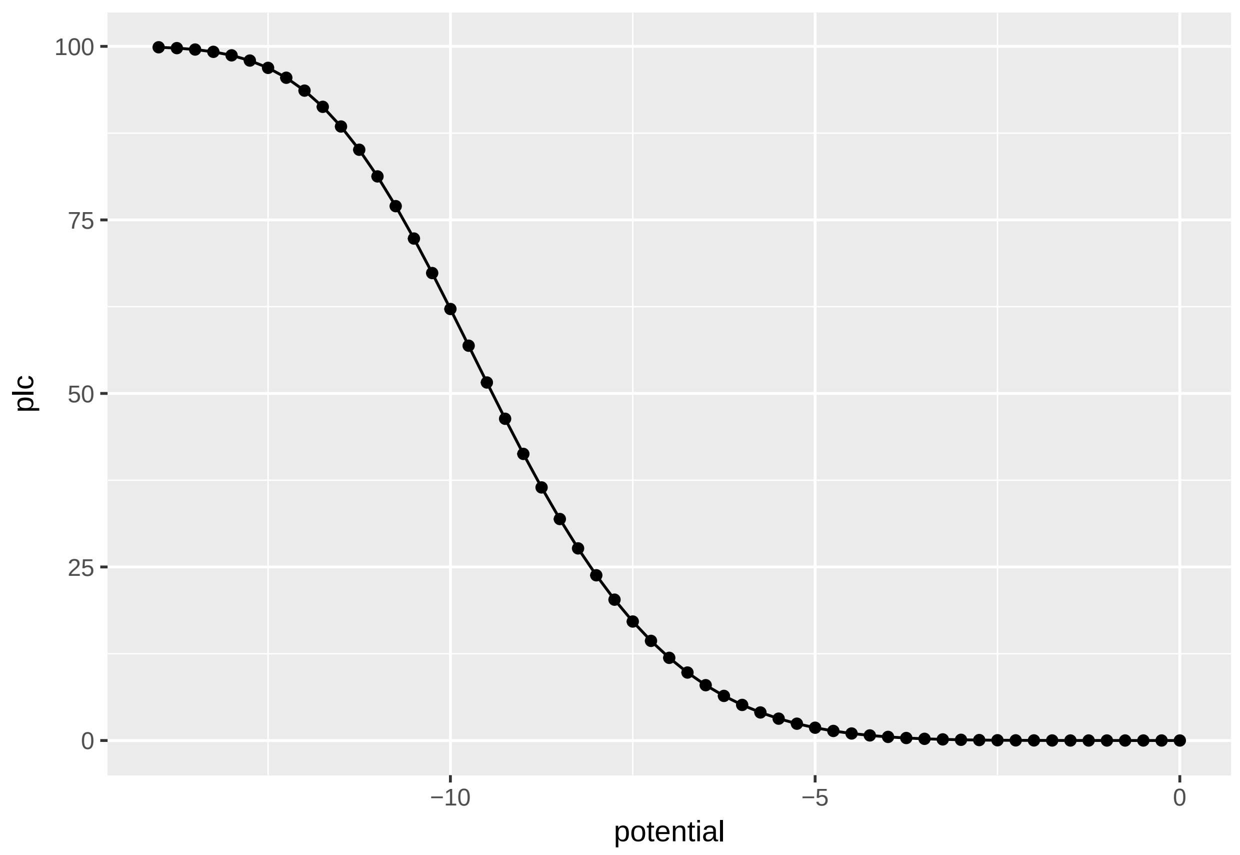
Create a point plot for the same data.

```
ceanothus_plot <- ggplot(cean_plot_data, aes(x=potential,y=plc)) +
                          geom_point()

ceanothus_plot
```



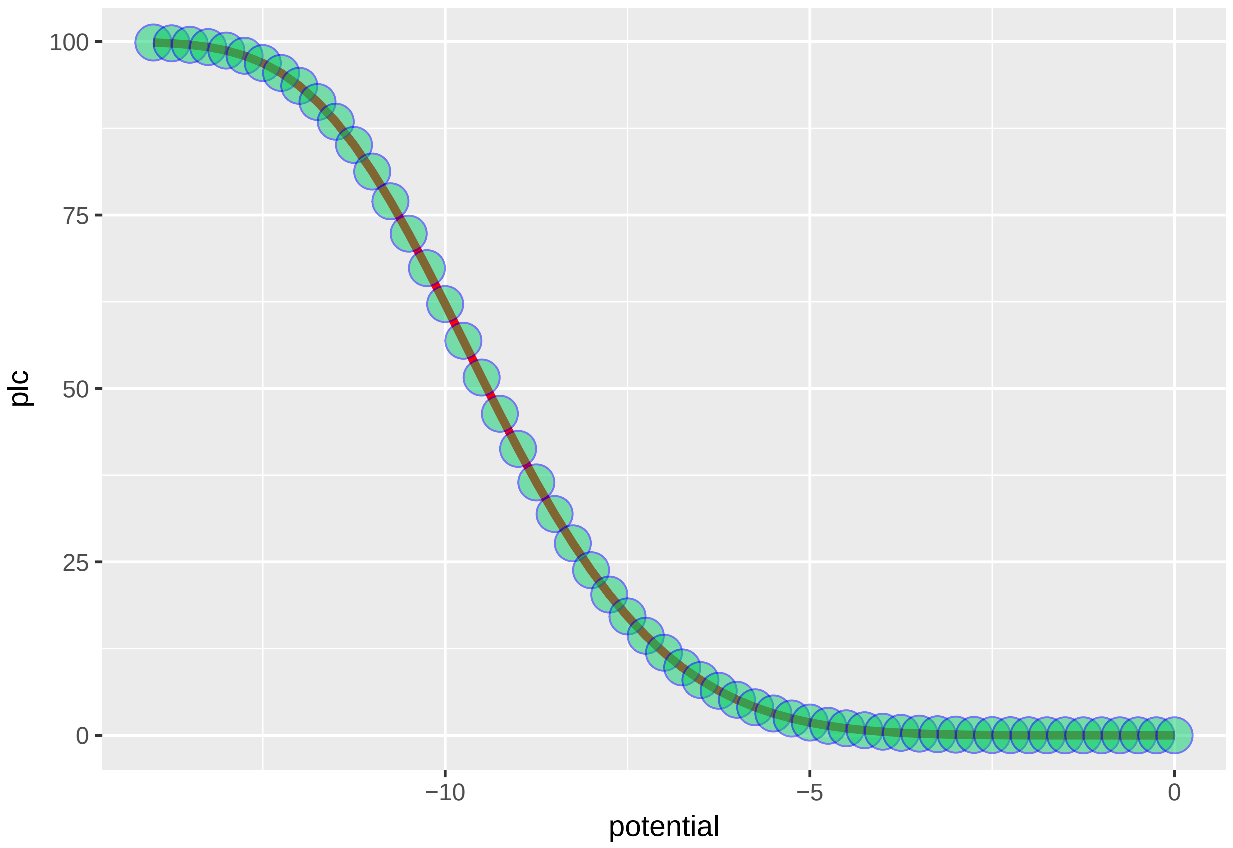
Produce a plot with points and a line.

```
ceanothus_plot <- ggplot(cean_plot_data, aes(x=potential,y=plc)) +
                         geom_line() +
                         geom_point()

ceanothus_plot
```



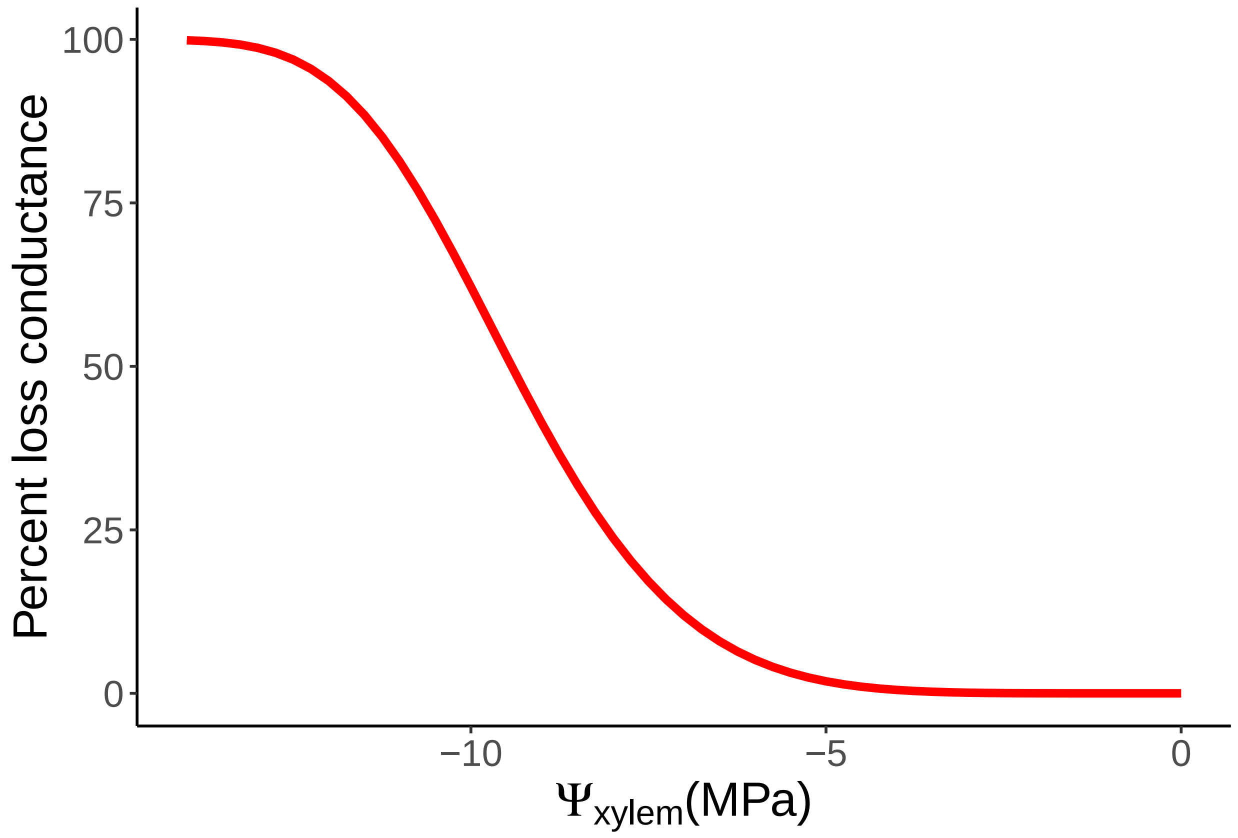
Produce a plot with points and a line, with some additional customization.

```
ceanothus_plot <- ggplot(cean_plot_data, aes(x=potential,y=plc)) +
                            geom_line(color="red",size=1.5) +
                            geom_point(shape=21,size=6,color="blue",fill="springgreen3",alpha=0.5)

ceanothus_plot
```



These plots cover some basic functionality of ggplot2, but they still have that aweful grey background, the axes aren't labeled properly, and the font is too small. We can sort these things out using axes labels and **theme**.

```
ceanothus_plot <- ggplot(cean_plot_data, aes(x=potential,y=plc)) +
                         geom_line(color="red",size=1.5) +
                         xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                         ylab("Percent loss conductance") +
                         theme_classic() +
                         theme(
                           text=element_text(size=18)
                         )

ceanothus_plot
```



Often, the data visualization associated with your current ecological research will be a little or a lot more complex than simply plotting one set of y values against one set of x values. Below we'll visualize all four species on the same plot, and then create *faceted* plots, one per species.

As a first step, we'll need to do a bit of data wrangling. Below, we'll combine all of our species y-axis data (PLC) with the one set of x-axis data (xylem water potential) into one data frame, then we'll use **gather** from the **tidyr** package to get everything into long format.

```
all_PLC_data <- cbind.data.frame(potential=psi_input,
                                 ceanothus=ceanothus_data,
                                 sage=sage_data,
                                 boxelder=boxelder_data,
                                 birch=birch_data)
head(all_PLC_data)
```

```
##   potential ceanothus sage boxelder birch
## 1    -14.00  99.86901  100      100   100
## 2    -13.75  99.74933  100      100   100
## 3    -13.50  99.54521  100      100   100
## 4    -13.25  99.21491  100      100   100
## 5    -13.00  98.70620  100      100   100
## 6    -12.75  97.95813  100      100   100
```

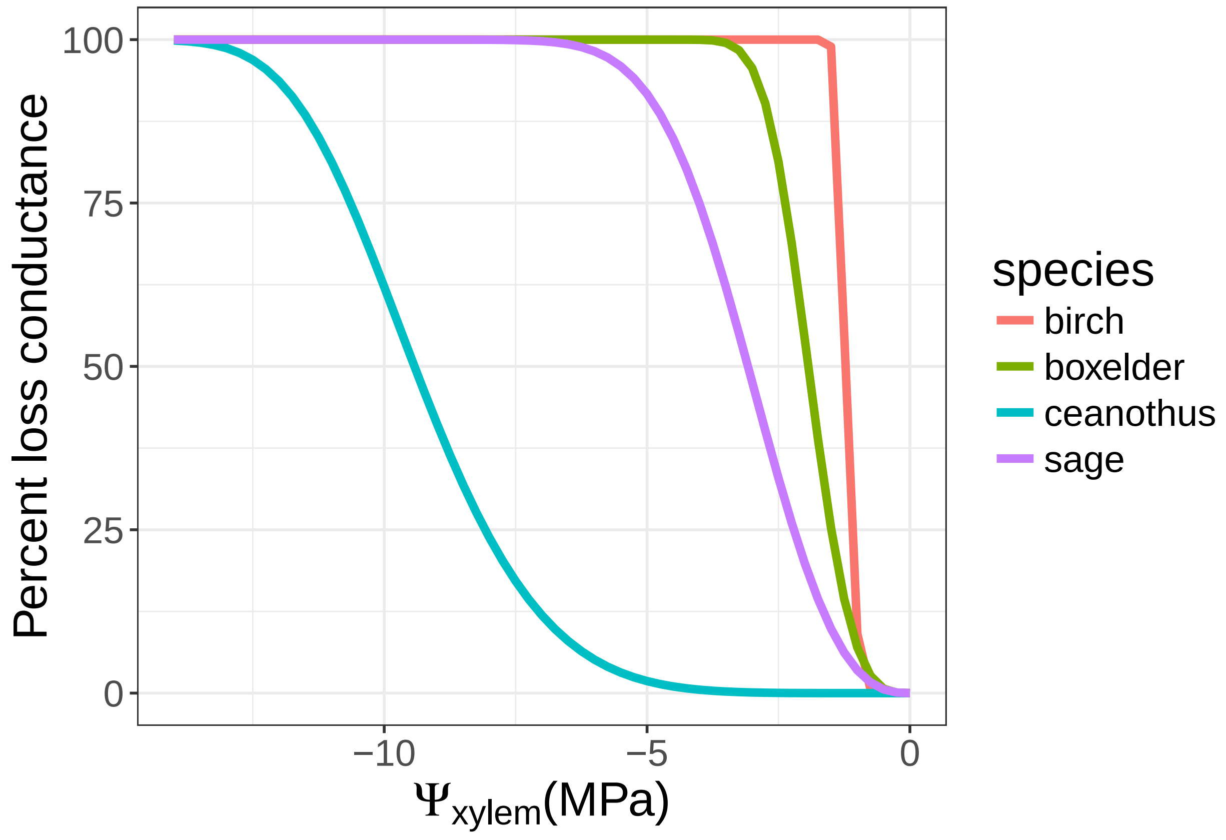
In the above data frame, we can see the data in *wide format*, so we use **gather** next. I sometimes add "_gg" as a suffix to the newly created *long format* data frame, just to help me keep track of things in case it makes sense to have the data stored in two formats.

```
all_PLC_data_gg <- all_PLC_data %>% gather("ceanothus", "sage","boxelder","birch",
                                             key = "species", value = "plc")
head(all_PLC_data_gg)
```

```
##   potential   species      plc
## 1    -14.00 ceanothus 99.86901
## 2    -13.75 ceanothus 99.74933
## 3    -13.50 ceanothus 99.54521
## 4    -13.25 ceanothus 99.21491
## 5    -13.00 ceanothus 98.70620
## 6    -12.75 ceanothus 97.95813
```

Create a line plot with all four species.

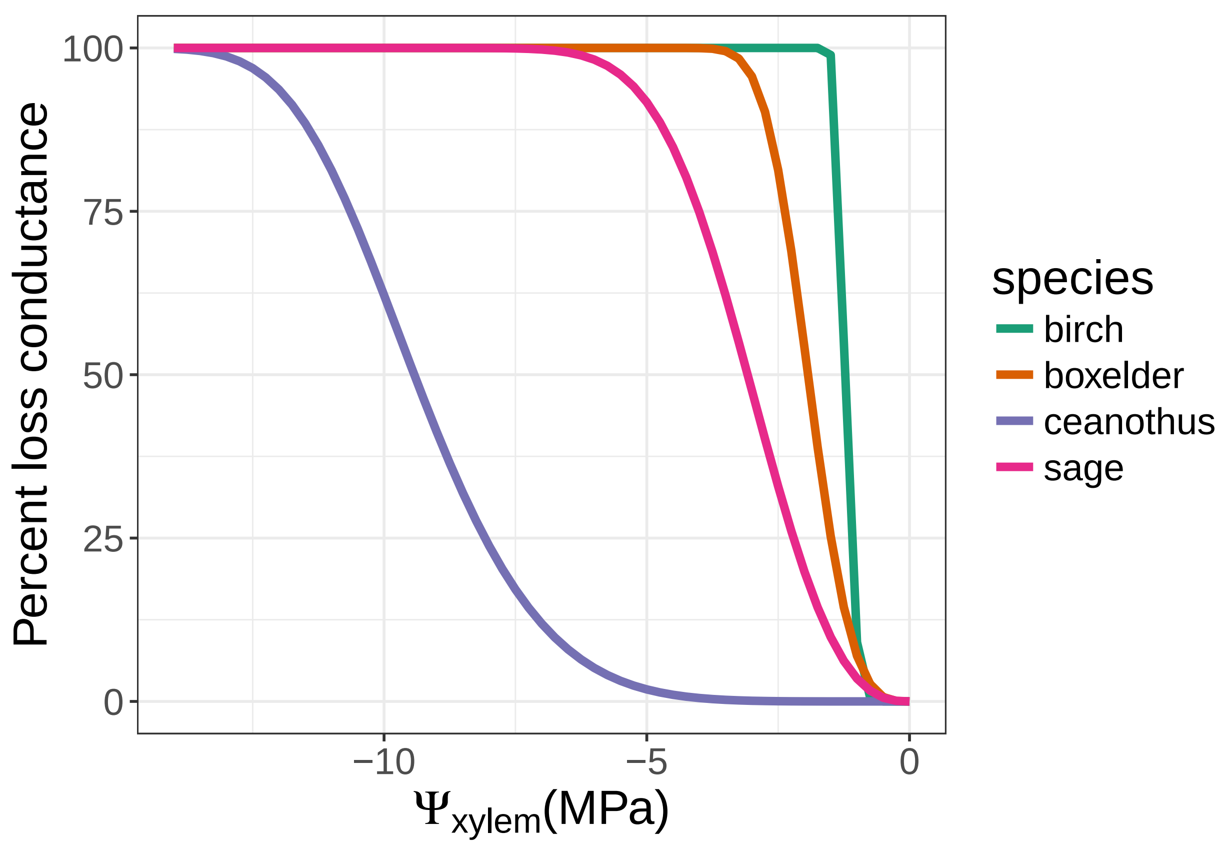
```
fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc,color=species)) +
                         geom_line(size=1.5) +
                         xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                         ylab("Percent loss conductance") +
                         theme_bw() +
                         theme(
                           text=element_text(size=18)
                         )

fourSpecies_plot
```



So here, we've got all four species visualized on the same plot. However, we can touch this up a bit more. Since we didn't provide further specification, ggplot went ahead and used its default color scheme to separate out each species and decided on how to set up the legend, which uses the column header text from the data frame to identify each species. There are a number of options for changing the color scheme, including manually adjusting each color.

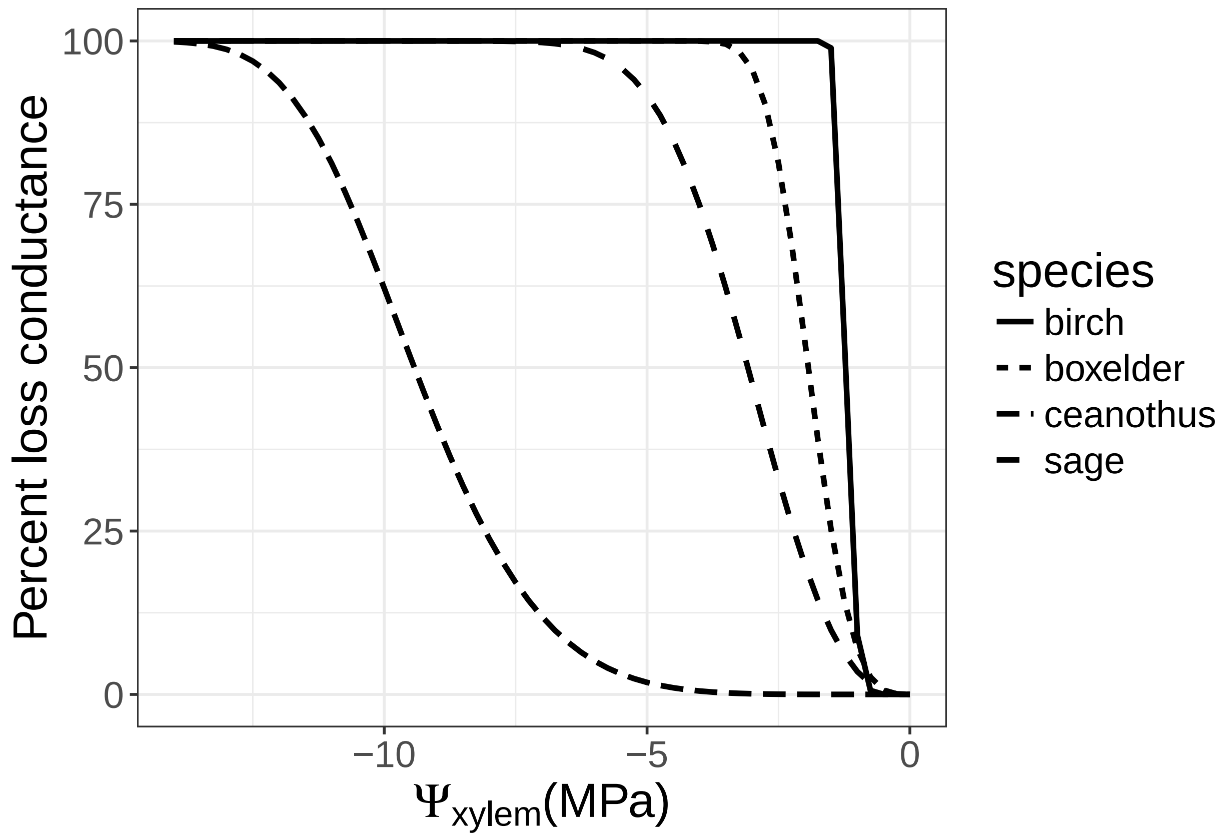
In this example, let's use the *Brewer Color Palette* to map different colors to each species type. Additional info on this can be found here: Brewer Color Palette Info.

```
fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc,color=species)) +
                    geom_line(size=1.5) +
                    scale_color_brewer(palette="Dark2") +
                    xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                    ylab("Percent loss conductance") +
                    theme_bw() +
                    theme(
                          text=element_text(size=18)
                    )

fourSpecies_plot
```



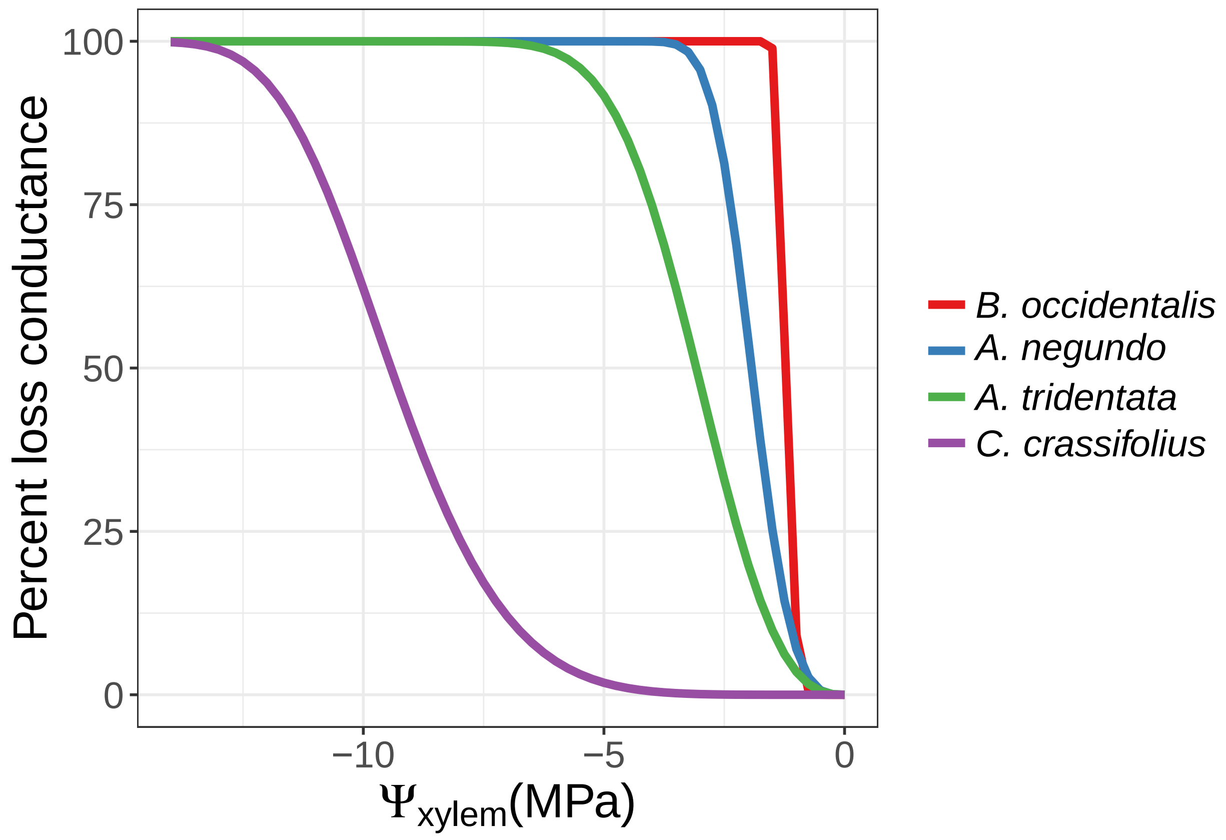
If we wanted to produce a black and white plot, we could map the *linetype* aesthetic to species instead of color.

```
fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc,linetype=species)) +
                    geom_line(size=1) +
                    xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                    ylab("Percent loss conductance") +
                    theme_bw() +
                    theme(
                                   text=element_text(size=18)
                                 )

fourSpecies_plot
```



We can also customize the order in which plotted layers appear (notice that depending on the ordering thus far, birch could blocking the other curves along the top of the plot) and manually assign labels to each variable in our *color* aesthetic group, to improve how each species appears in plot and the legend.

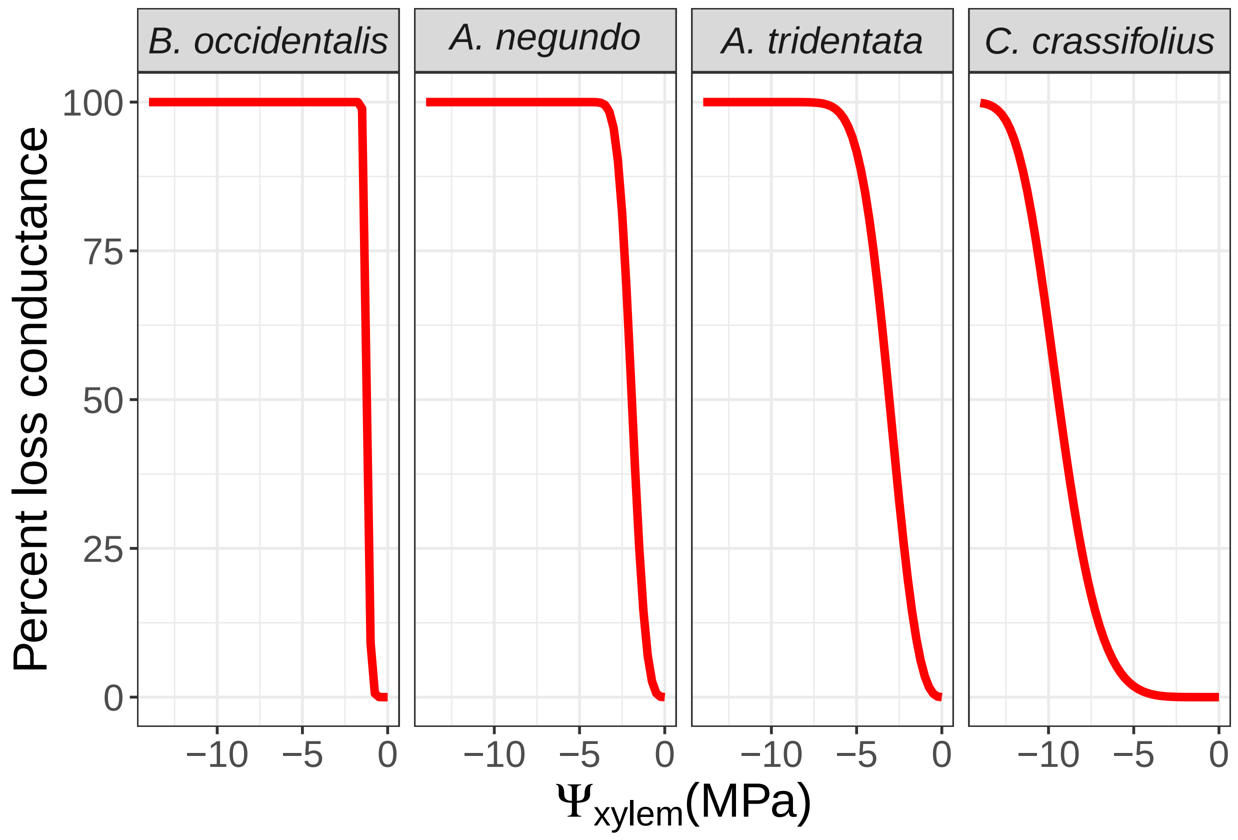
```
all_PLC_data_gg$species <- factor(all_PLC_data_gg$species,
                                  levels=c("birch","boxelder","sage","ceanothus"))


fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc,color=species)) +
                          geom_line(size=1.5) +
                          scale_color_brewer(palette="Set1",
                                             labels = c(expression(italic("B. occidentalis")),
                                                        expression(italic("A. negundo")),
                                                        expression(italic("A. tridentata")),
                                                        expression(italic("C. crassifolius"))
                                                        )) +
                          xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                          ylab("Percent loss conductance") +
                          theme_bw() +
                          theme(
                            legend.title=element_blank(),
                            legend.text.align = 0,
                            text=element_text(size=18)
                          )

 fourSpecies_plot
```



We can also use facetting to plot each species separately on adjacent plots.

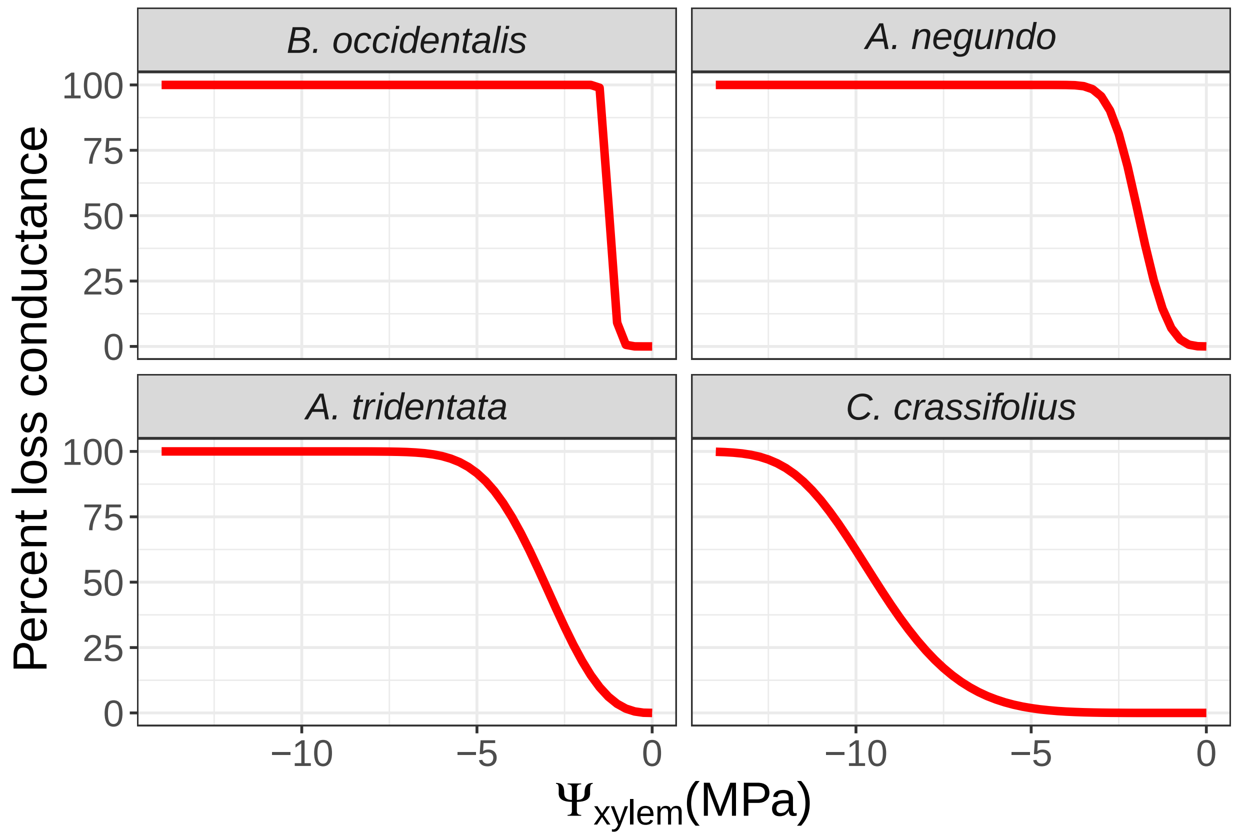
```
all_PLC_data_gg$species <- factor(all_PLC_data_gg$species,
                                  labels=c(expression(italic("B. occidentalis")),
                                           expression(italic("A. negundo")),
                                           expression(italic("A. tridentata")),
                                           expression(italic("C. crassifolius"))
                                           ))

fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc)) +
                         geom_line(size=1.5, color="red") +
                         facet_grid( ~ species,scales = "fixed", labeller = label_parsed) +
                         xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                         ylab("Percent loss conductance") +
                         theme_bw() +
                         theme(
                           legend.title=element_blank(),
                           legend.text.align = 0,
                           text=element_text(size=18)
                         )
```

```
fourSpecies_plot
```



The dimensions of the facets can be customized as well.

```
all_PLC_data_gg$species <- factor(all_PLC_data_gg$species,
                                  labels=c(expression(italic("B. occidentalis")),
                                           expression(italic("A. negundo")),
                                           expression(italic("A. tridentata")),
                                           expression(italic("C. crassifolius"))
                                           ))

fourSpecies_plot <- ggplot(all_PLC_data_gg, aes(x=potential,y=plc)) +
                         geom_line(size=1.5, color="red") +
                         facet_wrap( ~ species, scales = "fixed", labeller = label_parsed,ncol = 2) +
                         xlab(expression(paste(Psi[xylem],"(MPa)"))) +
                         ylab("Percent loss conductance") +
                         theme_bw() +
                         theme(
                           legend.title=element_blank(),
                           legend.text.align = 0,
                           text=element_text(size=18)
                         )

fourSpecies_plot
```



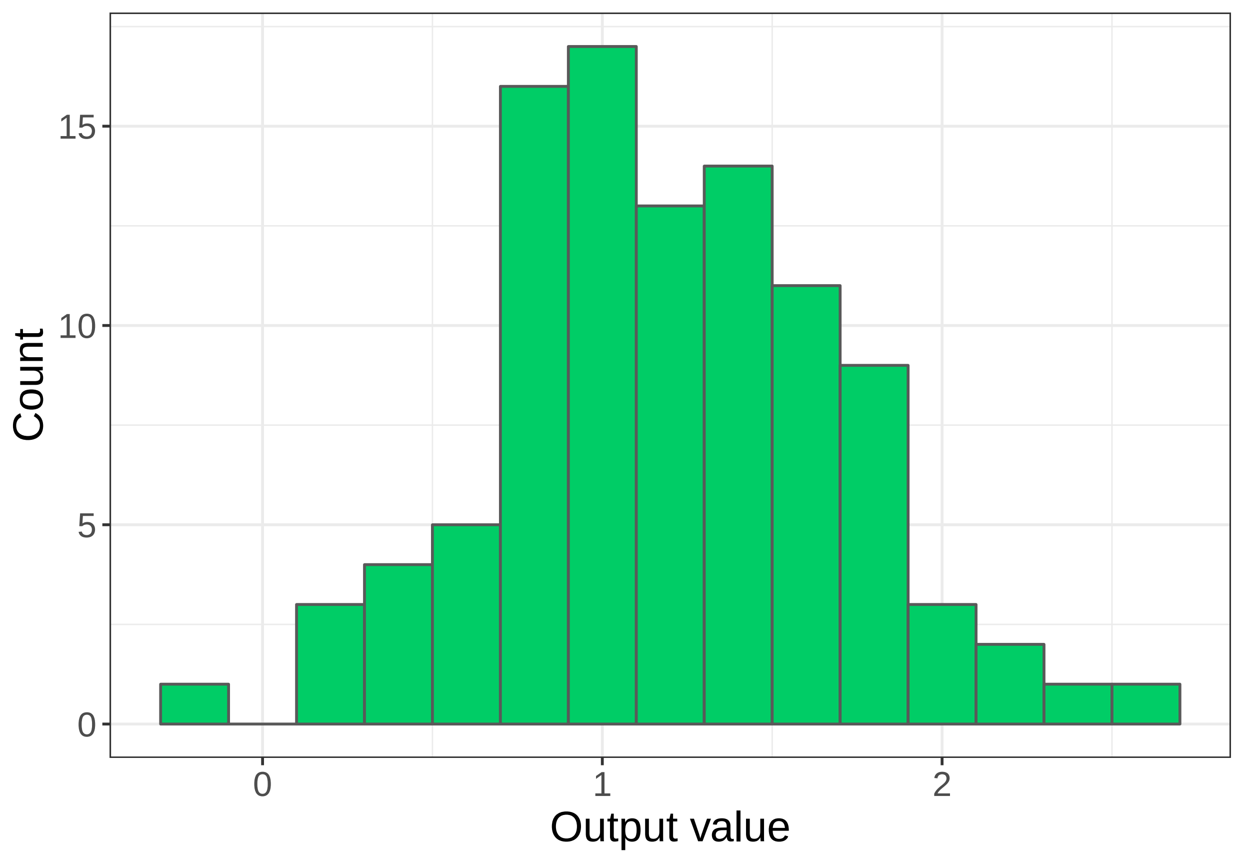
Ok, so now lets create some categorical data sets, and create a few plots to visualize them.

Here, we'll just create some random data using different normal distributions, for Groups A-E, and get them into a wide and long format data frame.

```
groupA <- rnorm(100,5.3,1.2)
groupB <- rnorm(100,6.1,1.7)
groupC <- rnorm(100,9.5,1.4)
groupD <- rnorm(100,1.2,0.5)
groupE <- rnorm(100,8.5,1.2)

cat_data <- cbind.data.frame(groupA, groupB, groupC, groupD, groupE)

cat_data_gg <- cat_data %>% gather("groupA", "groupB", "groupC", "groupD", "groupE",
                                   key = "group", value = "value")
```

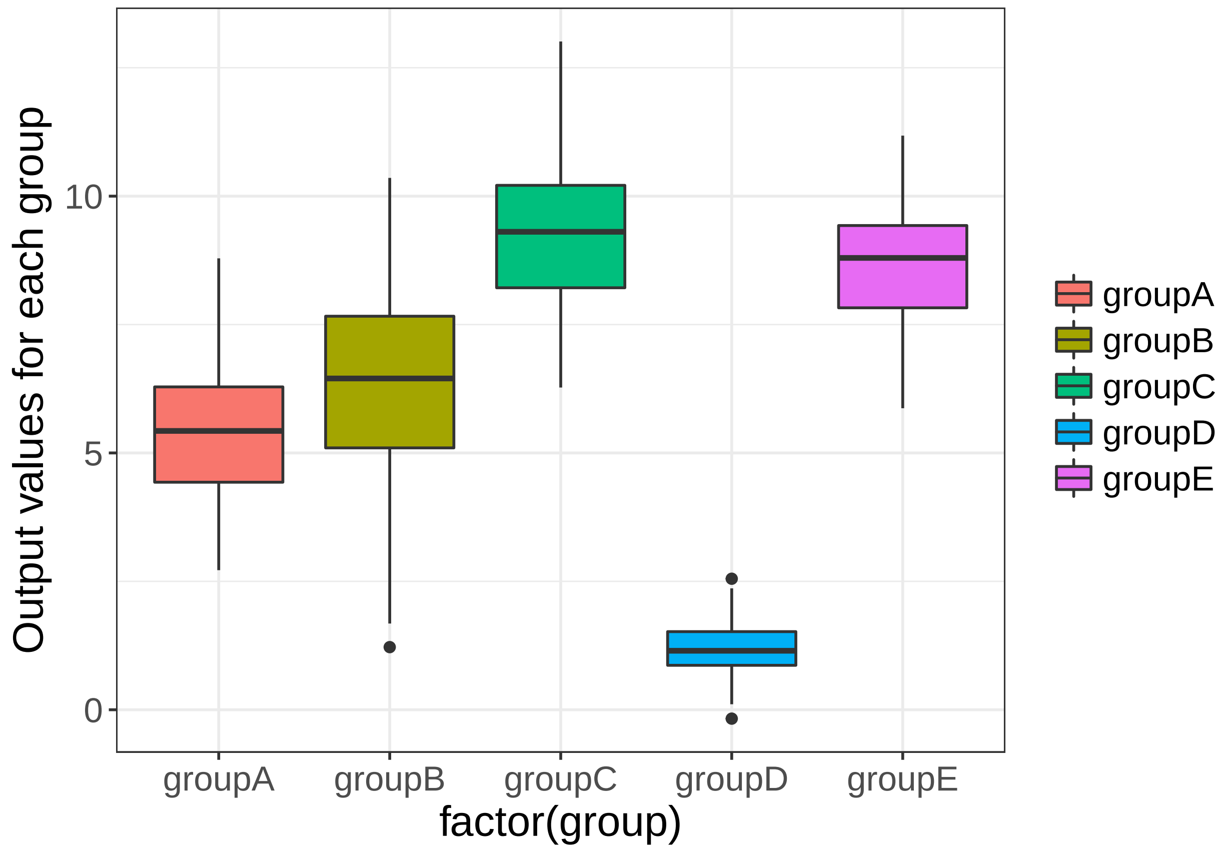
Create a histogram with data from Group A. Note that we're using the wide format data frame here.

```
hist_plot <- ggplot(cat_data, aes(groupD)) +
  geom_histogram(binwidth = 0.2,fill="springgreen3",color="grey35") +
  xlab("Output value") +
  ylab("Count") +
  theme_bw() +
  theme(text=element_text(size=16),
        legend.title=element_blank()
        )

hist_plot
```



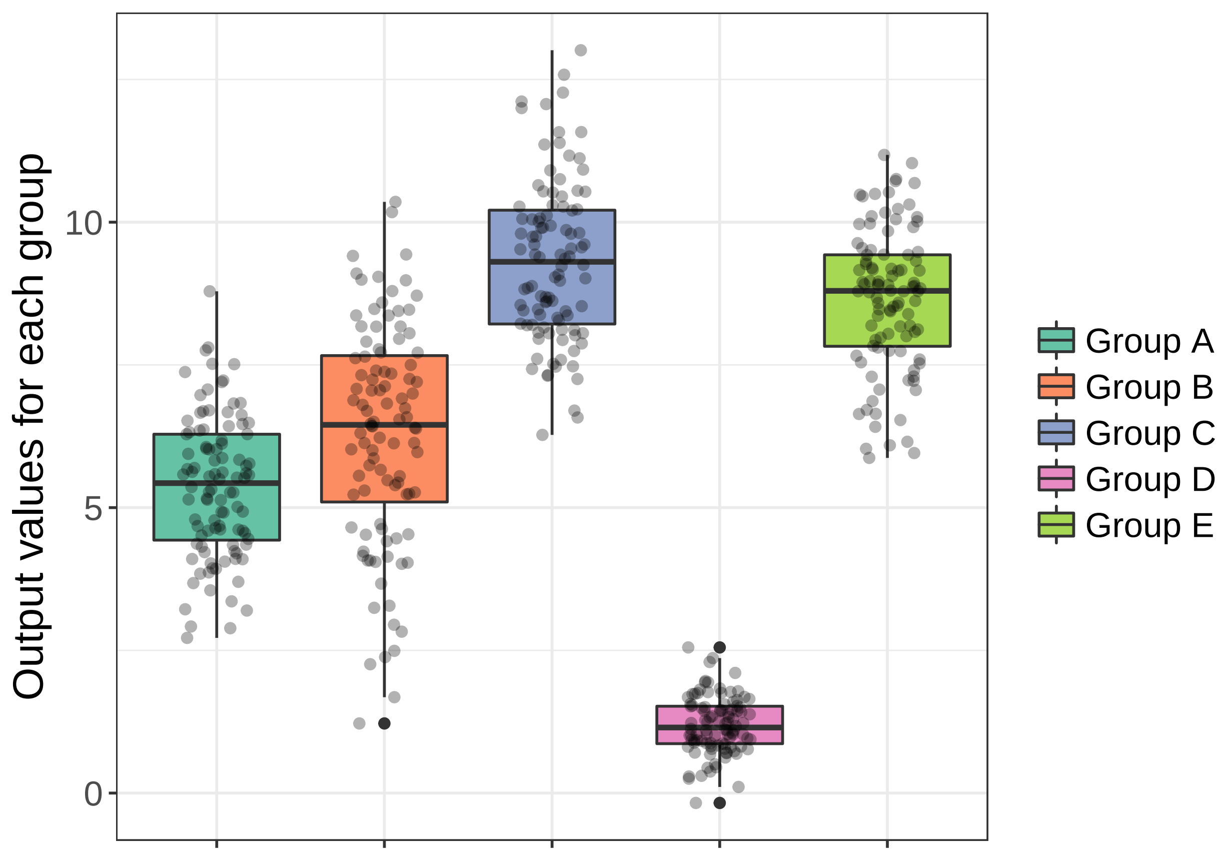
Create some boxplots covering all group. Here, we're using the long format data frame.

```
box_plot <- ggplot(cat_data_gg, aes(factor(group), value)) +
  geom_boxplot(aes(fill = factor(group))) +
  ylab("Output values for each group") +
  theme_bw() +
  theme(text=element_text(size=16),
        legend.title=element_blank()
        )

box_plot
```



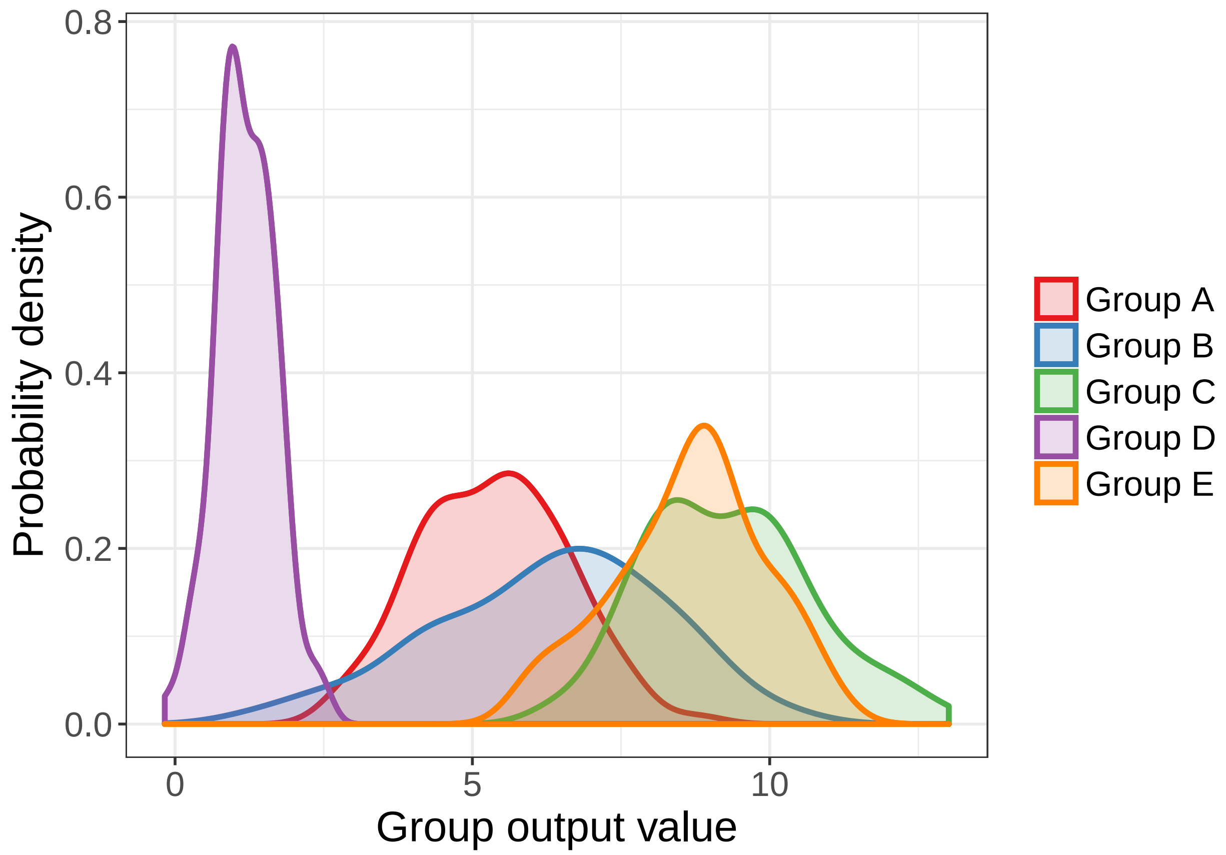
Let's add jittered point on top of the boxplots and spruce things up a bit.

```
box_plot <- ggplot(cat_data_gg, aes(factor(group), value)) +
  geom_boxplot(aes(fill = factor(group))) +
  scale_fill_brewer(palette="Set2",
                    labels = c("Group A",
                               "Group B",
                               "Group C",
                               "Group D",
                               "Group E")) +
  geom_jitter(width = 0.2,alpha=0.3) +
  ylab("Output values for each group") +
  theme_bw() +
  theme(text=element_text(size=16),
        legend.title=element_blank(),
        axis.text.x=element_blank(),
        axis.title.x=element_blank()
        )

box_plot
```



Present the data as a density plot.

```
density_plot <- ggplot(cat_data_gg, aes(x=value,color=group,fill=group)) +
  geom_density(alpha=1,size=1,fill="NA") +
  geom_density(alpha=.2,size=1) +
  scale_color_brewer(palette="Set1",
                     labels = c("Group A",
                                "Group B",
                                "Group C",
                                "Group D",
                                "Group E")) +
  scale_fill_brewer(palette="Set1",
                     labels = c("Group A",
                                "Group B",
                                "Group C",
                                "Group D",
                                "Group E")) +
  xlab("Group output value") +
  ylab("Probability density") +
  theme_bw() +
  theme(text=element_text(size=16),
        legend.title=element_blank()
        )

density_plot
```

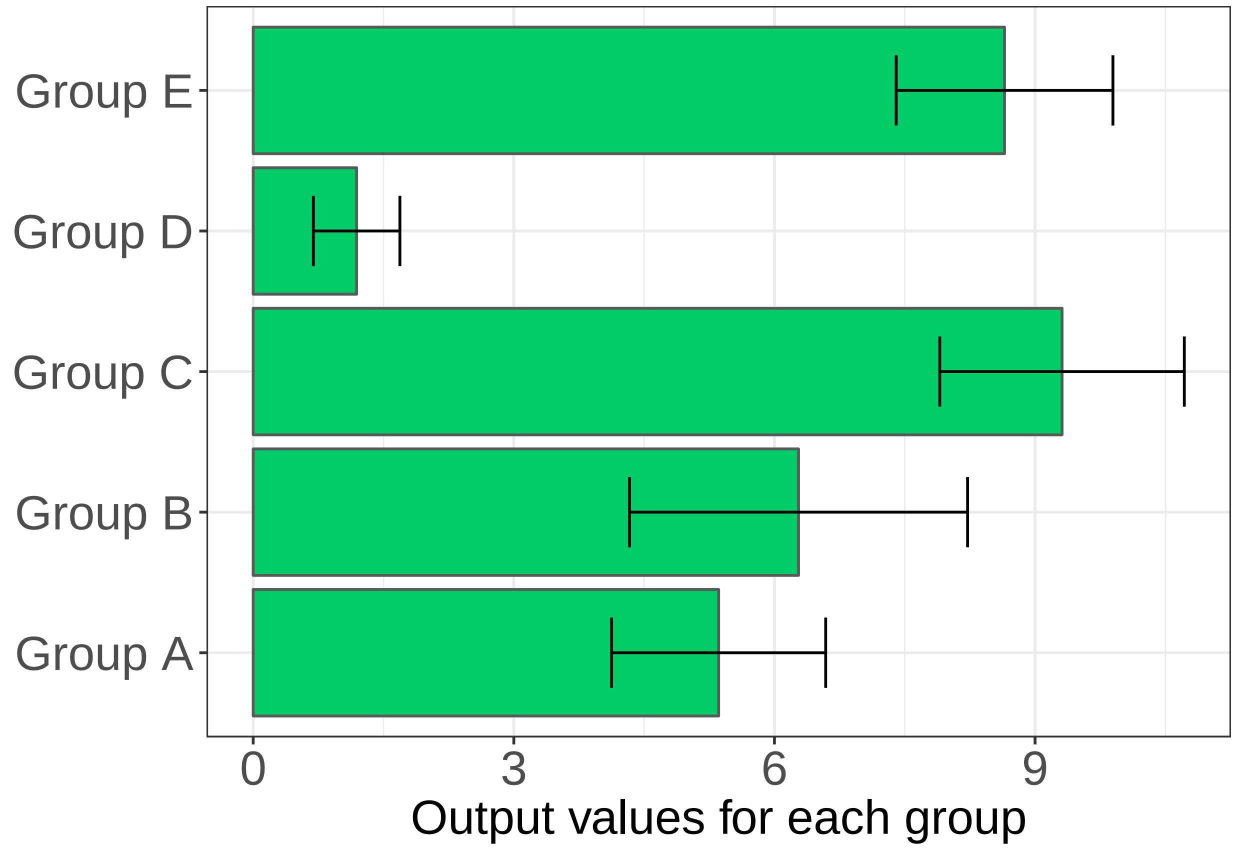


Now, let's create a column plot where the columns represent the means for each group and the error bars represent standard deviation. Note that here we're going to pretend we don't already know the mean and standard deviation for each group, and use the summarize function.

```
group_stats_gg <- cat_data_gg %>%
               group_by(group) %>%
               summarize(mean = mean(value),sd = sd(value))

group_stats_gg$group <- factor(group_stats_gg$group,
                                   labels = c("Group A",
                                              "Group B",
                                              "Group C",
                                              "Group D",
                                              "Group E"))
group_stats_gg
```

```
## # A tibble: 5 x 3
##     group     mean        sd
##    <fctr>    <dbl>     <dbl>
## 1 Group A 5.357571 1.2322412
## 2 Group B 6.277367 1.9457252
## 3 Group C 9.310551 1.4075501
## 4 Group D 1.190531 0.4981305
## 5 Group E 8.648545 1.2474963
```

```
column_plot <- ggplot(group_stats_gg, aes(x=group,y=mean)) +
  geom_col(fill="springgreen3",color="grey35",size=0.5) +
  geom_errorbar(aes(ymin=mean-sd, ymax=mean+sd), size=0.5, width=0.5) +
  coord_flip() +
  ylab("Output values for each group") +
  theme_bw() +
  theme(text=element_text(size=18),
        axis.text=element_text(size=18),
        legend.title=element_blank(),
        axis.title.y=element_blank()
        )
column_plot
```



Use **ggsave** to save your plots as .tiff, .pdf, etc. file types.

```
ggsave("column_plot.pdf",column_plot,width=6,height=4,units='in',dpi=300)
```

Hopefully if you're just starting out with ggplot2, you've found these examples useful. That said, there is a plethora of additional features and types of analyses available in ggplot2, and additional information can be found here: ggplot2 homepage. Also, R Studio has a handy ggplot2 cheatsheet R Studio Cheatsheet.