Session 3: Data Visualization

Stephanie Simpson

July 14, 2020

Contents

Load packages

Download the following packages in order to run this script.

As a reminder, these packages need to be installed already before they can be added here. If they haven't been installed yet, you can do it by uncommenting and running this line of code. This will allow you to install multiple packages at once.

```
# install.packages(c("readr", "tidyverse", "ggthemes", "cowplot", "gridExtra"))
```

Load in data

Load in the data by using the read_csv function (which calls on readr). To run this script, we are going to use Nichole's single item recognition data (in long format). You will need to *change this directory* to match

the one on your personal computer (i.e., you need to tell R where it should go and grab your data csv file).

```
# you need to change this path to match your personal directory of where you saved the csv file
# this allows us to load in our data and call it "sirdat"
sirdat <-read_csv('~/Dropbox/Baycrest_Rworkshop/2020/SngItmRec_Long.csv')
```

```
## Parsed with column specification:
## cols(
##   subid = col_double(),
##   condition = col_character(),
##   stimulus = col_character(),
##   hit_minus_fa = col_double(),
##   group = col_character()
## )
```

View data

Make sure the data (which we called sirdat) looks correct before continuing.

```
View(sirdat)
```

```
## Warning in system2("/usr/bin/otool", c("-L", shQuote(DSO)), stdout = TRUE):
## running command ''/usr/bin/otool' -L '/Library/Frameworks/R.framework/
## Resources/modules/R_de.so'' had status 1
```

What is ggplot2?

ggplot2 is a system for creating graphics in R and is one of the core packages in the tidyverse.

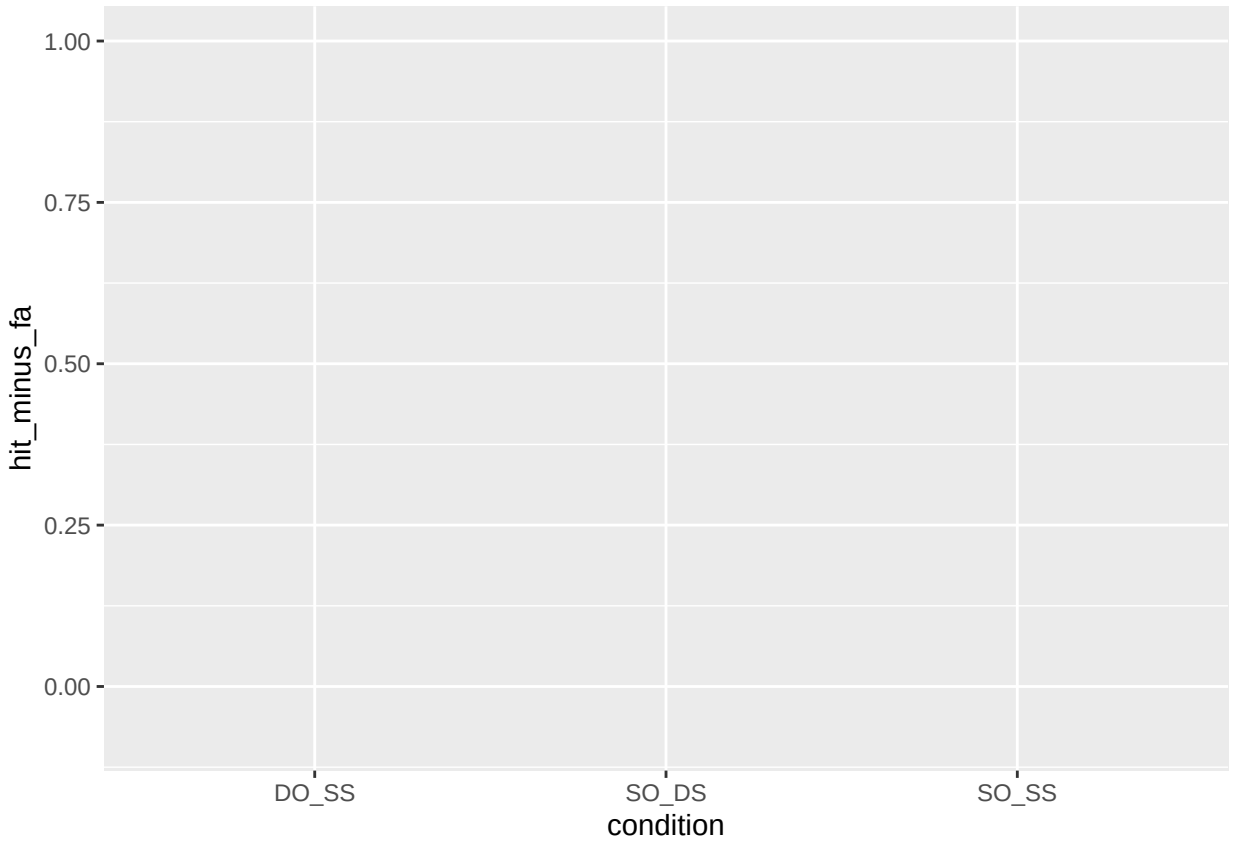
ggplot2 is all about layering! The idea is that you build your graph by literally adding different components to your figure, one line at a time. We first tell the function ggplot() which data we would like to visualize and then add layers like:
- aesthetic properties to represent variables - aes(),
- geoms to represent data points - e.g., geom_point(),
- scales - e.g., scale_colour_brewer()

See https://ggplot2.tidyverse.org/ for more information.

Plot #1: Bar Plot

Make the base

```
sirdat %>%
  ggplot(aes(x = condition, # tells R what variable will be plotted on the x axis
             y = hit_minus_fa, # tells R what variable will be plotted on the y axis
             fill = group)) # tells R how to assign colour to our bars (in this case, by the group vari
```

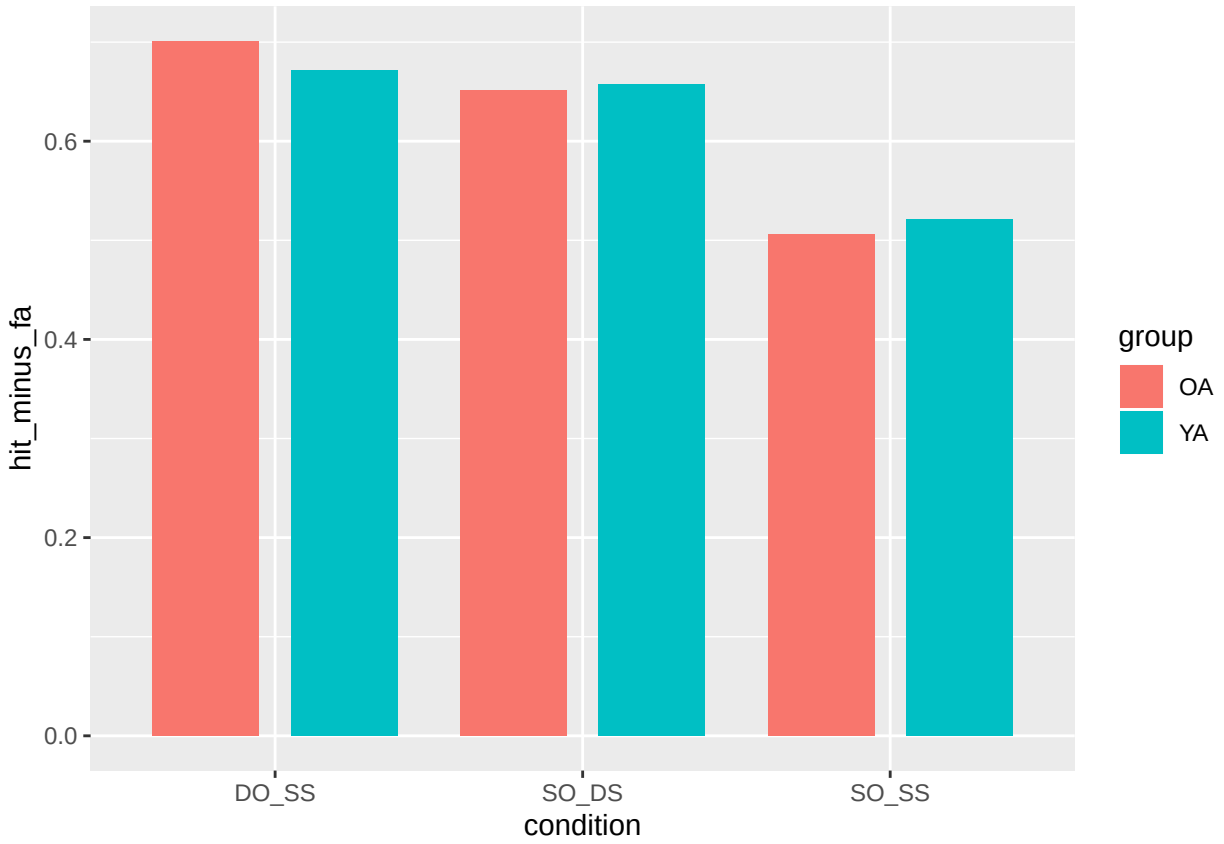


This shows us the "skeleton" of our plot, but we haven't added any of the actual data points to it yet. We can do this by adding stat_summary. stat_XX is an alternative way to build layers - it will add transformations of the original data set.

Add some features

Let's also give the plot a name so it's easier to add more features.

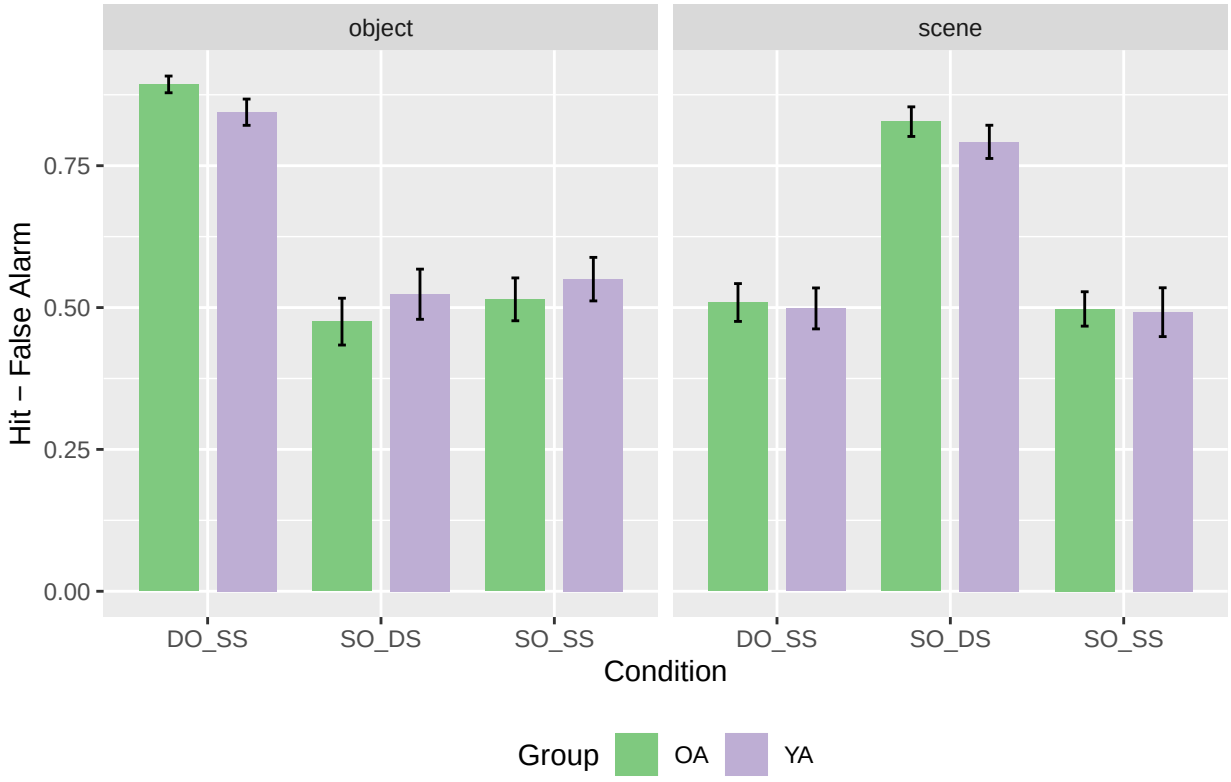
```
bar_plot <- sirdat %>% # first, name the plot to make it easier to add featurs later on
  ggplot(aes(x = condition,
             y = hit_minus_fa,
             fill = group)) +
  # remember to use fill = for bar graphs!
  stat_summary(fun.y = mean, # this  function will output the mean values of your y variable
               geom = "bar", # rather than adding a separate geom, we can just place it within stat_sum
               width = 0.69, # change the width of your individual bars
               position = position_dodge(width = .9)) # changes how close the bars are positioned next
                                                      # (smaller # = closer together)

bar_plot
```



To make our graph "manuscript-ready", we need to add some measure of effect size, labels, and change the colours. It would also be nice to display the accuracy rate by stimulus types. To do this, we can apply a facet wrap. This allows you to separate your data into subsets and then plot them all together (http://www.cookbook-r.com/Graphs/Facets_(ggplot2)/).

```
bar_plot +
  stat_summary(fun.data = mean_se, # this function will be applied to all of the data
                       geom = "errorbar", # tells ggplot we want to add errorbars
                       width = 0.09, # width of "ticks"
                       position = position_dodge(width = 0.9)) +
                       # this needs to match the witdth of your bars so that the errorbars are aligned
  facet_wrap(~stimulus) + # facets will split up your figure by different variables
  # scale_fill_manual("Group", values = c("darkgreen", "darkolivegreen3")) +
  scale_fill_brewer("Group", palette = "Accent") + # change the colour
  labs(title = "Figure 1. Bar Graph", # add an overall title
       x = "Condition", # add a title to the x axis
       y = "Hit - False Alarm") + # add a title to the y axis
  theme(legend.position = "bottom") # move the legend to the bottom
```

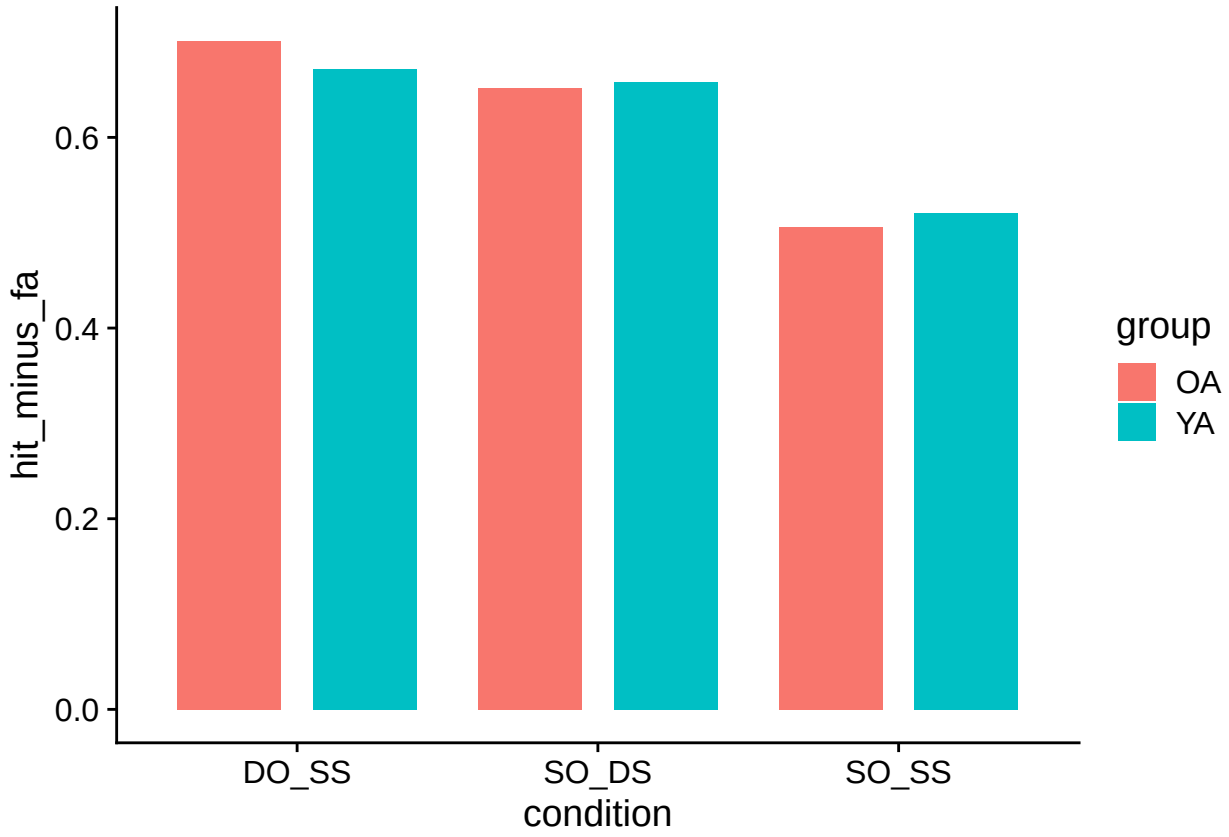
Figure 1. Bar Graph



Themes

I'm currently using the built-in gpplot2 themes, but we can change this as well! For instance, we can two other packages called cowplot (https://cran.r-project.org/web/packages/cowplot/vignettes/introduction.html) or ggthemes (https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/). Check out their websites for more great ideas.

```
bar_plot +
  # theme_bw()
  # or try something with ggthemes!
  # theme_few()
  theme_cowplot()
```

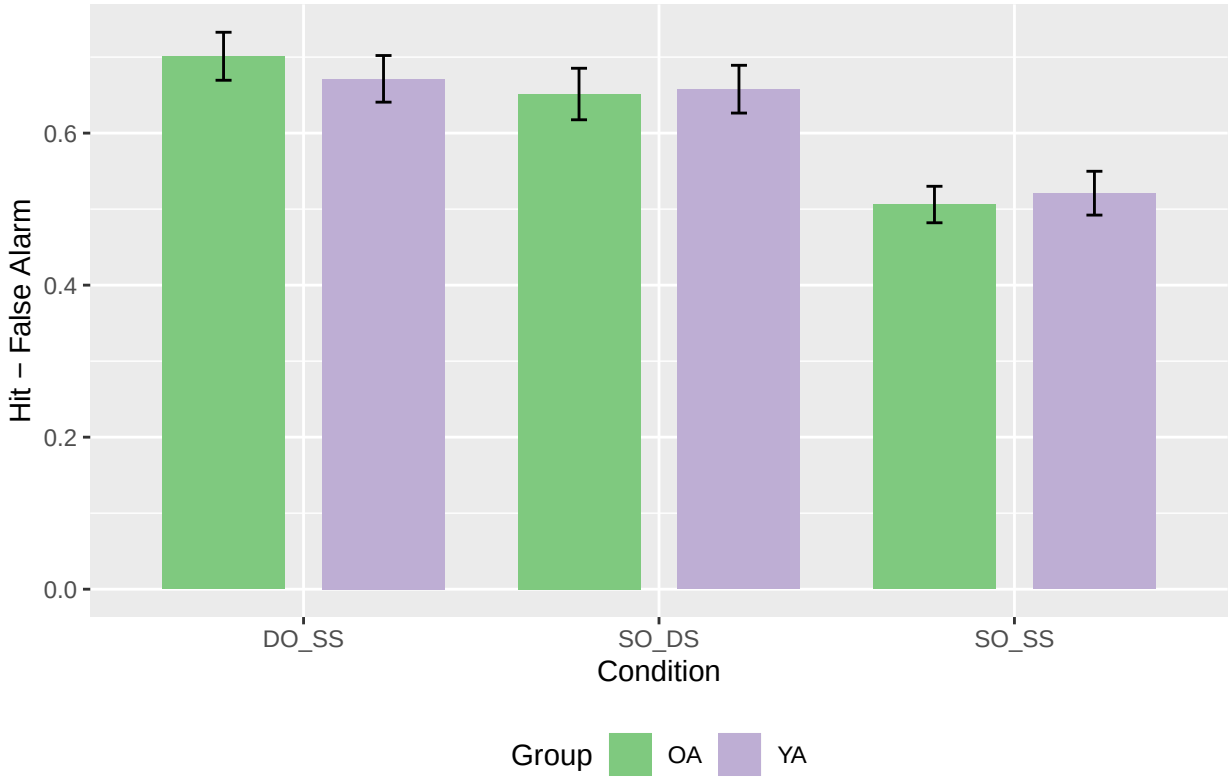


Override

Notice here that R is displaying the older version of the barplot. If we want to see our more polished version, we need to override the original bar_plot.

```
bar_plot <- bar_plot +
  stat_summary(fun.data = mean_se,
               geom = "errorbar",
               width = 0.09,
               position = position_dodge(width = 0.9)) +
  # facet_wrap(~stimulus) +
  # scale_fill_manual("Group", values = c("darkgreen", "darkolivegreen3")) +
  scale_fill_brewer("Group", palette = "Accent") +
  labs(title = "Figure 1. Bar Graph",
       x = "Condition",
       y = "Hit - False Alarm") +
  theme(legend.position = "bottom")

bar_plot
```

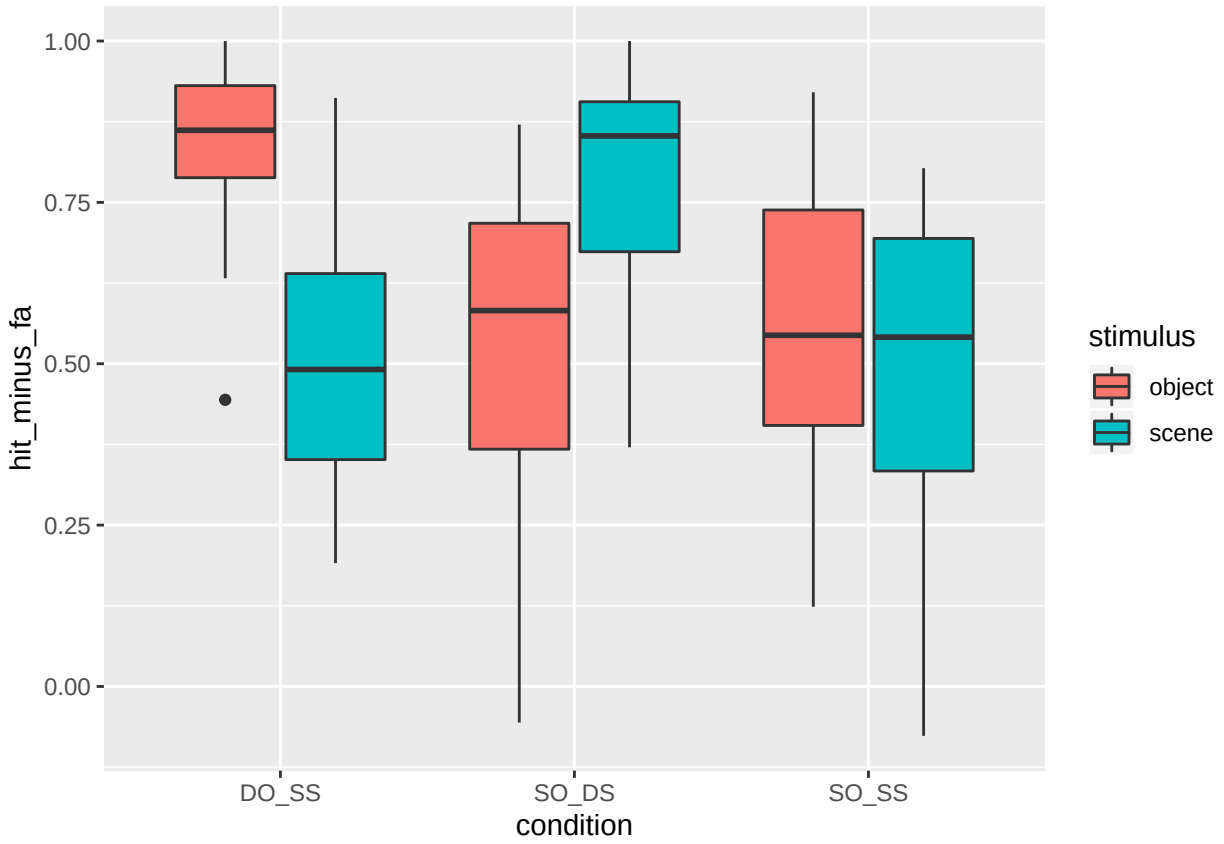
Figure 1. Bar Graph



Plot #2: Box Plot

Base of the plot

```
box_plot <- sirdat %>%
  # drop_na(hit_minus_fa) %>%
  filter(group == "YA") %>% # this selects only the YA subjects in the group factor
  ggplot(aes(x = condition, # variable that is plotted on x axis
             y = hit_minus_fa, # variable that is plotted on y axis
             fill = stimulus)) + # how to colour (or fill in) the boxes
  geom_boxplot(varwidth = TRUE) # adding a box plot geom
  # when varwidth = T, boxes are drawn with widths proportional to the square-root of the # of observat

box_plot
```



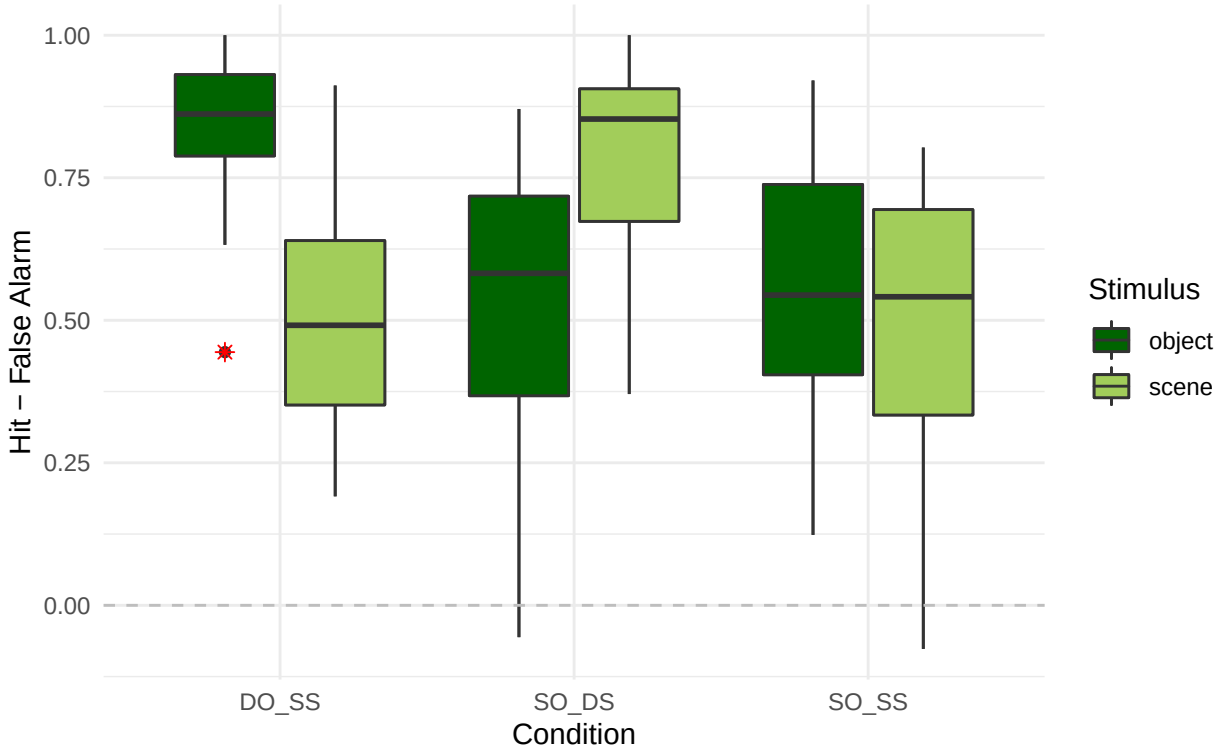
Customizations

Again, let's spice this up! This line of code will: - modify the appearance of outliers - add a horizontal line - create labels - change colours - label and reposition the legend

```
box_plot <- box_plot +
  geom_boxplot(outlier.colour = "red", # modify the appearance of outliers
               outlier.shape = 8,
               outlier.size = 2) +
  # scale_y_discrete(limits = c(0, 0.5, 1, 1.5)) +
  geom_hline(yintercept=0, # placement
             linetype="dashed", # changes the style of the line
             color = "gray") + # add a horizontal line
  labs(title = "Figure 2. Box Plot",
       subtitle = "Dots represent potential outliers", # add subtitles
       x = "Condition",
       y = "Hit - False Alarm") +
  scale_fill_manual("Stimulus", # label the legend
                    values = c("darkgreen", "darkolivegreen3")) + # manually change colours
  theme(legend.position = "bottom") + # move legend
  theme_minimal() # add a general theme

box_plot
```

Figure 2. Box Plot

Dots represent potential outliers



Plot #3: Scatterplot

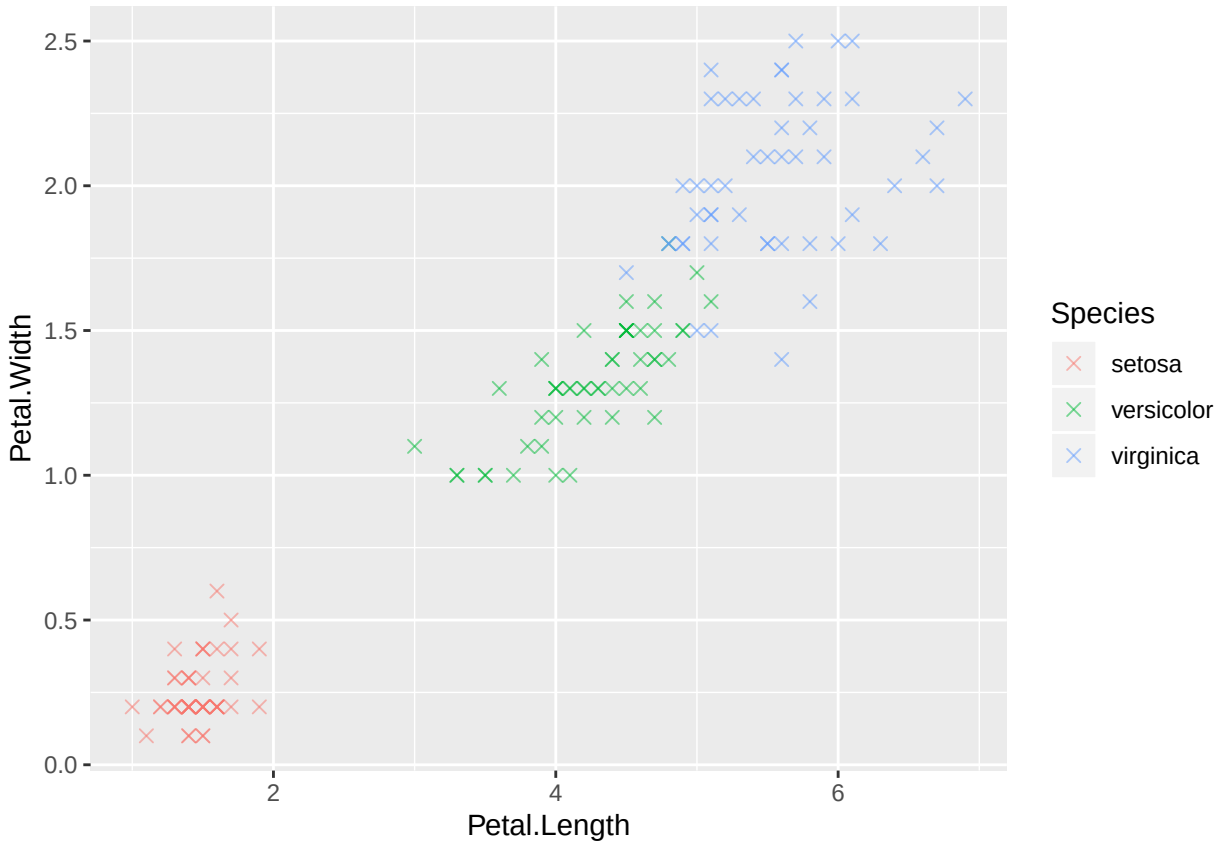
Iris dataset

Let's try one of R's built-in datasets called iris.

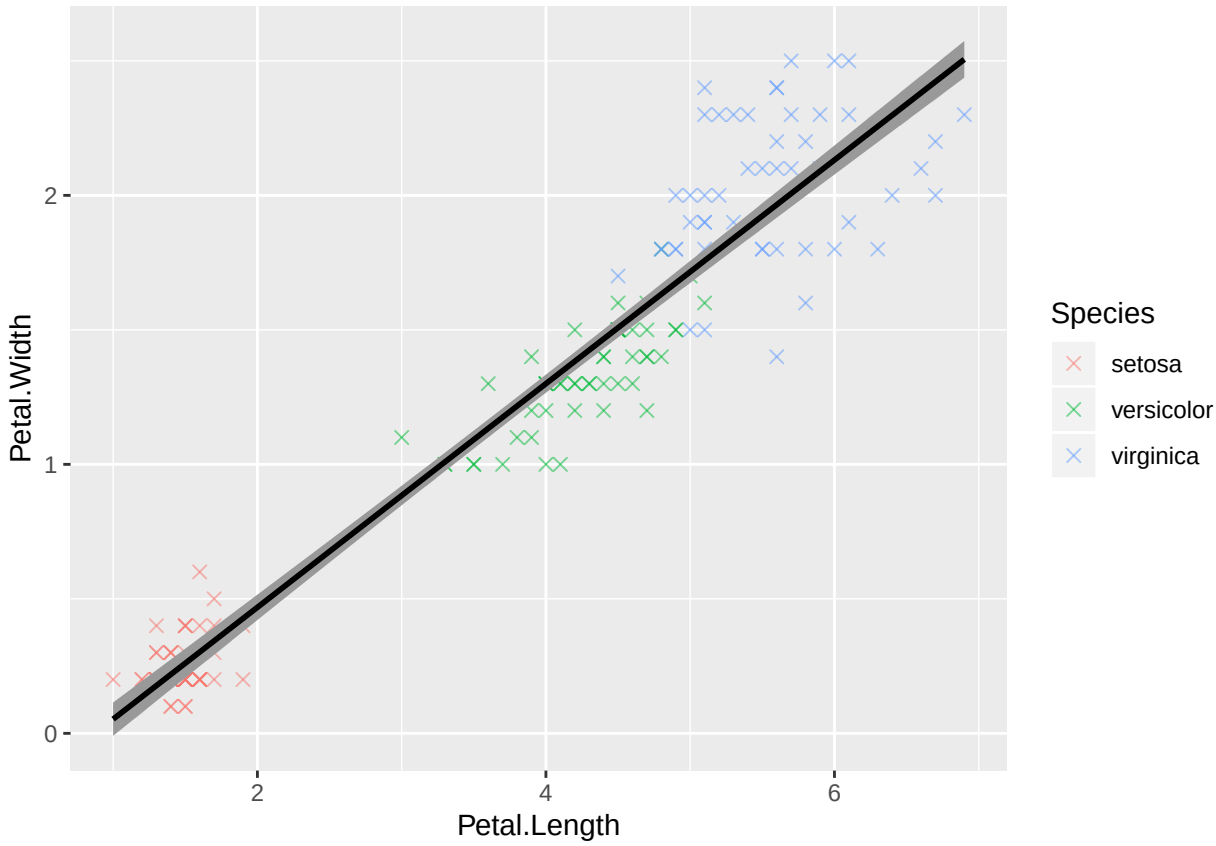
```
?iris
```

Scatterplot A

```
# scatterplot base
scatterplot_base <- iris %>%
  ggplot(aes(x = Petal.Length, # variable plotted on x axis
             y = Petal.Width, # variable on y axis
             color = Species)) + # change colour based on species of flower
  geom_point(alpha = .5, # alpha will adjust how translucent your points are (lower value = faded colou
             size = 2, # change the size of your points
             shape = 4) # change the shape, there are many different kinds to choose from

scatterplot_base
```

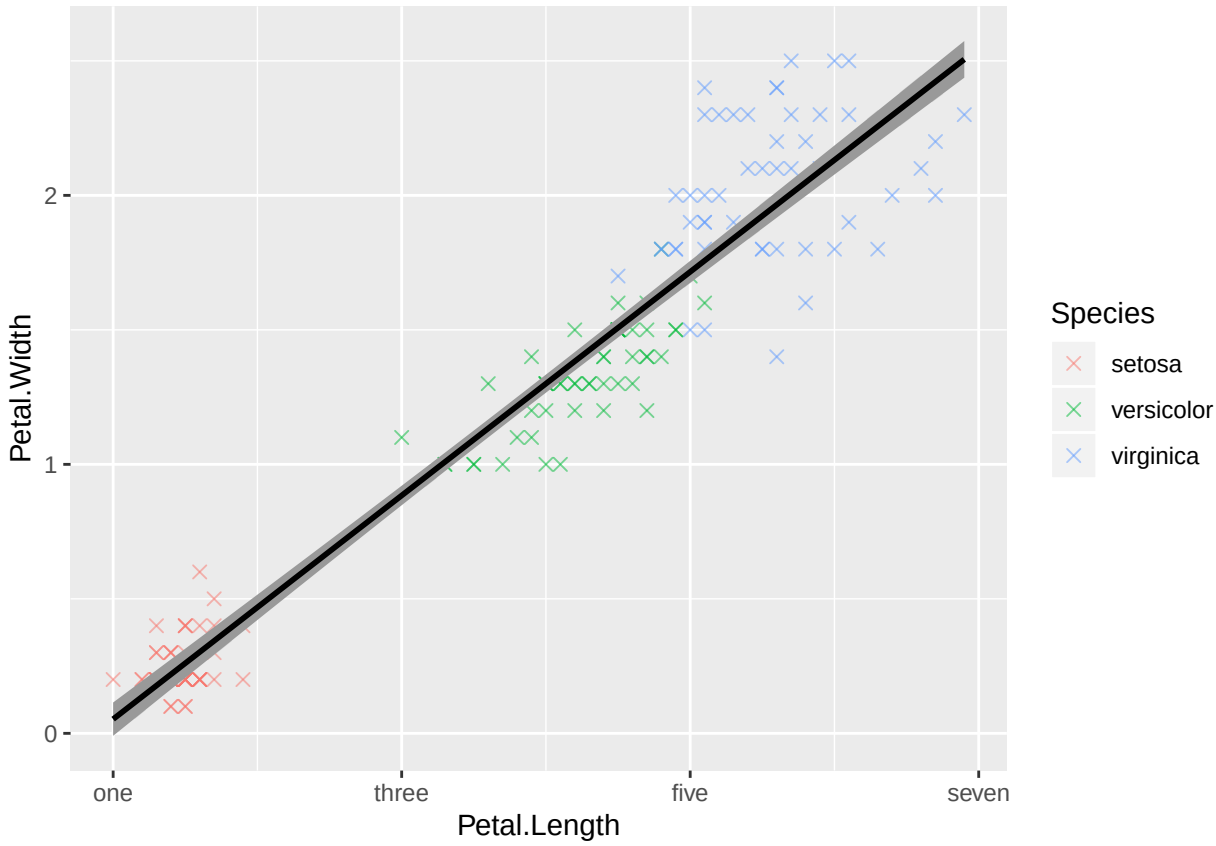



```
# add a single linear trend line to this plot with geom_smooth
# this helps better visualize the pattern in your data

scatterplot <- scatterplot_base +
  geom_smooth(method = "lm", # this will change how the line is smoothed (e.g., glm, loess, gam)
              size = 1, # thickness of line
              # aes(fill=Species),
              color = "black",
              show.legend = FALSE, # removes the line legend from the display
              se = TRUE,# adds SE to line
              alpha = 1) # changes transparency of the line

scatterplot
```



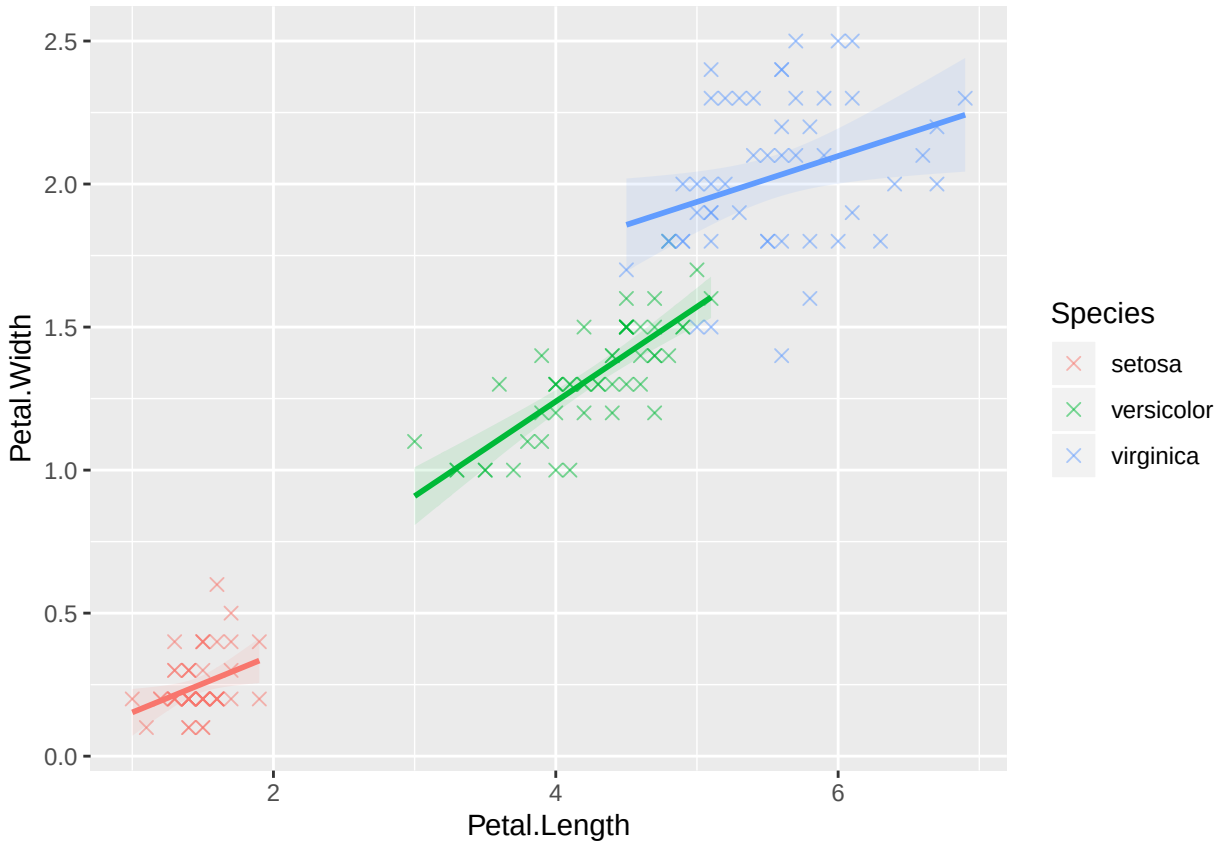
Change the axis labels

```
scatterplot +
  scale_x_continuous(breaks = c(1, 3, 5, 7), # how to break up values displayed on the x axis
                     labels = c("one", "three", "five", "seven")) # what will actually be displayed
```



Scatterplot B

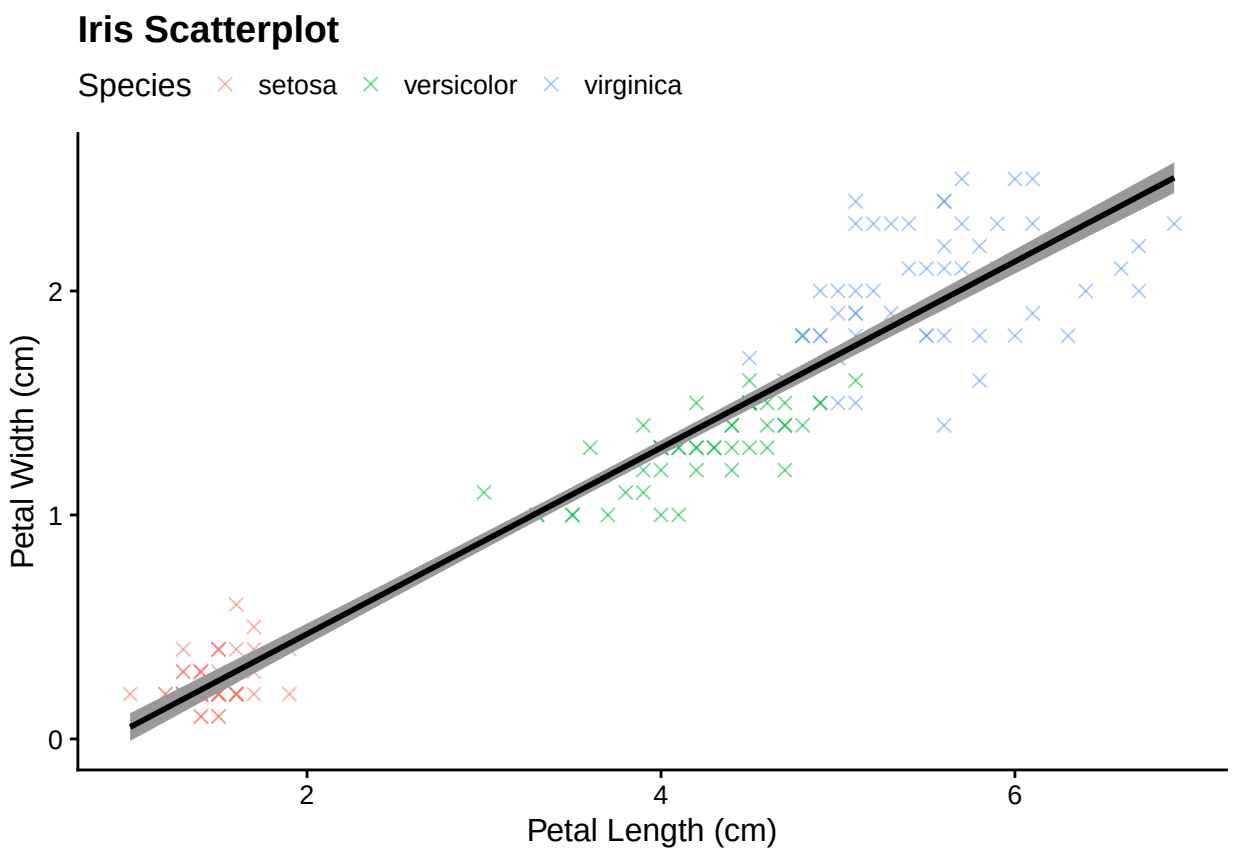
```
# same code as above, but if I add an aesthetic, then it will affect how the lines are displayed
scatterplot_base +
  geom_smooth(method = "lm",
              alpha = 0.1,
              size = 1,
              aes(fill=Species), # this adds a line to each subset of data (by Species)
              # color = "black",
              show.legend = FALSE,
              se = TRUE)
```



Try yourself

How would you add some labels? How would you change the theme to cowplot? How would you reposition the legend to the top of your figure?

```
scatterplot +
  labs(title = "Iris Scatterplot",
       y = "Petal Width (cm)",
       x = "Petal Length (cm)") +
  theme_cowplot(font_size = 12) +
  theme(legend.position = "top")
```



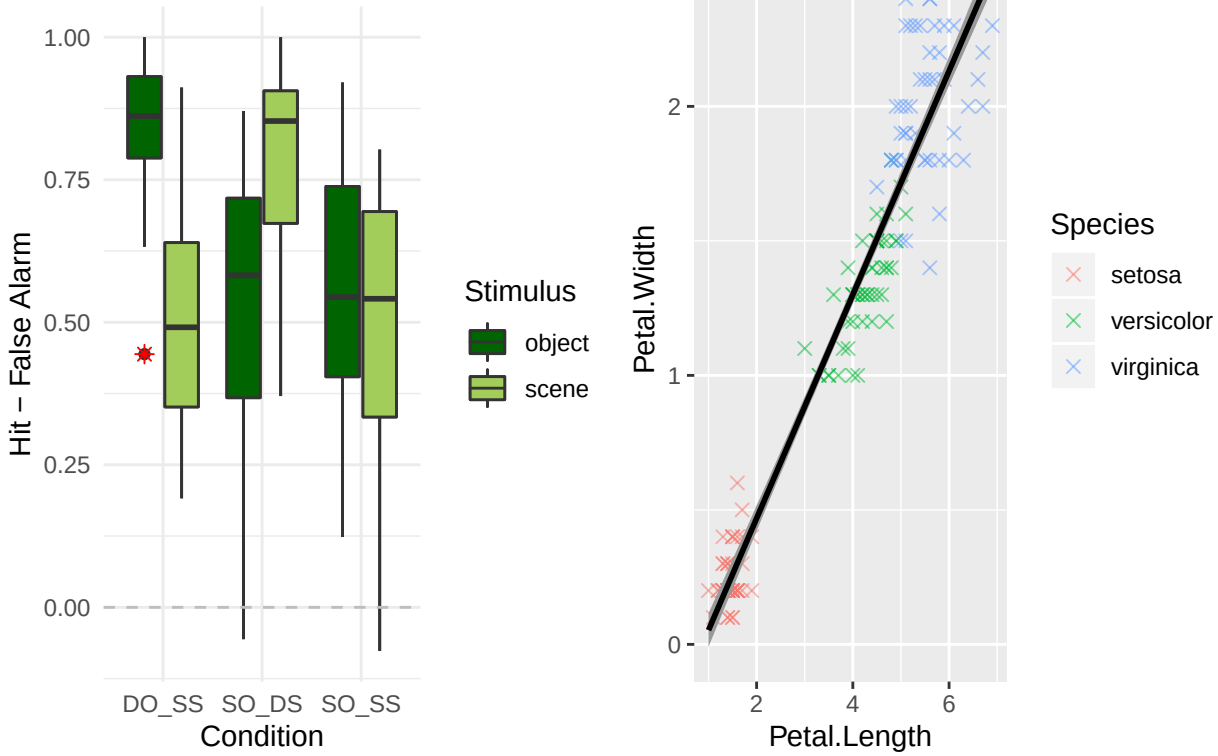
Other tips

Combine plots

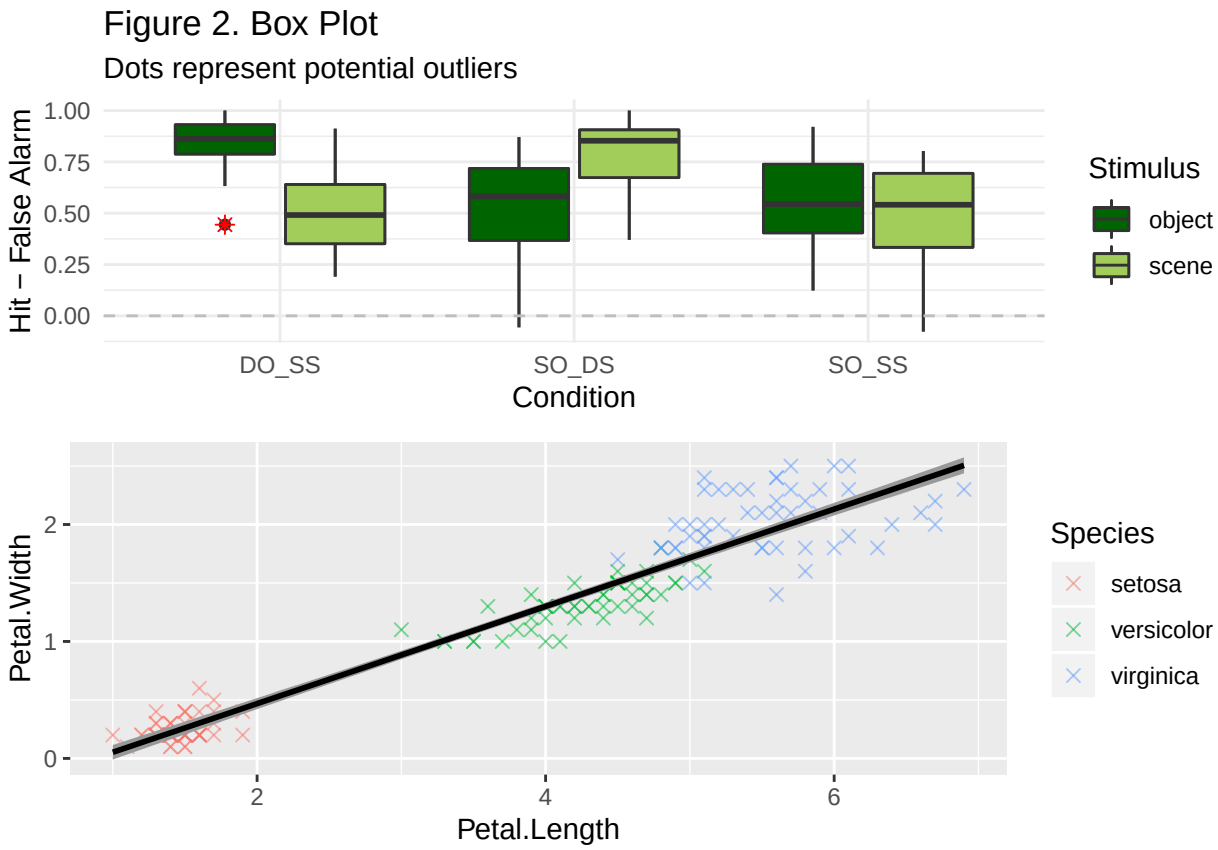
```
# using the :: allows you to call the package without downloading it (though it must be installed)

# side by side orientation
gridExtra::grid.arrange(box_plot, # left plot
                        scatterplot, # right plot to join
                        ncol = 2) # side by side orientation
```

Figure 2. Box Plot

Dots represent potential outliers




```
gridExtra::grid.arrange(box_plot,
                        scatterplot,
                        ncol = 1) # vertical orientation
```



Save your figures

This is key to adding your figures to any manuscript!

```
# if you don't specify the plot, then it will automatically save the last plot you loaded

ggsave("figure1.png", # title of saved image
       plot = bar_plot, # if you don't specify the plot, then it will automatically save the last plot
       path = "~/Dropbox/Baycrest_Rworkshop/2020/part3/figures", # change this path for your personal u
       dpi = 300, # clarity of image
       width = 6, # image width
       height = 6, # image height
       units = "in") # in = inches
```

Resources

- Change the shape of your points = https://www.datanovia.com/en/blog/ggplot-point-shapes-best-tips/
- Change the colour = http://www.sthda.com/english/wiki/ggplot2-colors-how-to-change-colors-automatically-and-manua
  use-wes-anderson-color-palettes
- R color sheet = http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf
- ggthemes = https://yutannihilation.github.io/allYourFigureAreBelongToUs/ggthemes/
- ggplot2 cheatsheet = https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf