Post 1

Jackie Zou

10/28/2017

Introduction to Interactive Plots with plotly

Interactive plots are extremely powerful ways to provide yourself and your readers with plots that can respond and transform to isolate different relationships in your data, in many cases eliminating the need to create multiple views of the same graph with separate lines of code. While there are multiple strategies available to do this, for this tutorial we'll be using the package plotly, which can turn existing ggplots (that fortunately we already know how to make) into interactive ones in just minutes, as well as be used to directly make interactive plots.

Note however that one big limitation of plotly is that it is an html-output based package, so your github files will need to include the command "always_allow_html: yes" in your YAML headers, and the interactive output will not show up.

Downloading plotly

First we're going to want to download the "plotly" package, which is available in CRAN. Simply run the following commands in your R console to download the package and then load it into your file. For the purposes of knitting this document I've commented out the command that installs plotly, but if you've never used it before be sure to load it onto your device from CRAN!

```
#install.packages("plotly")
library(plotly)
```

```
## Loading required package: ggplot2
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

```
#We'll also load these following packages which we'll be using in parts of this tutoria
l.
library(ggplot2)
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
```

A ratty first example

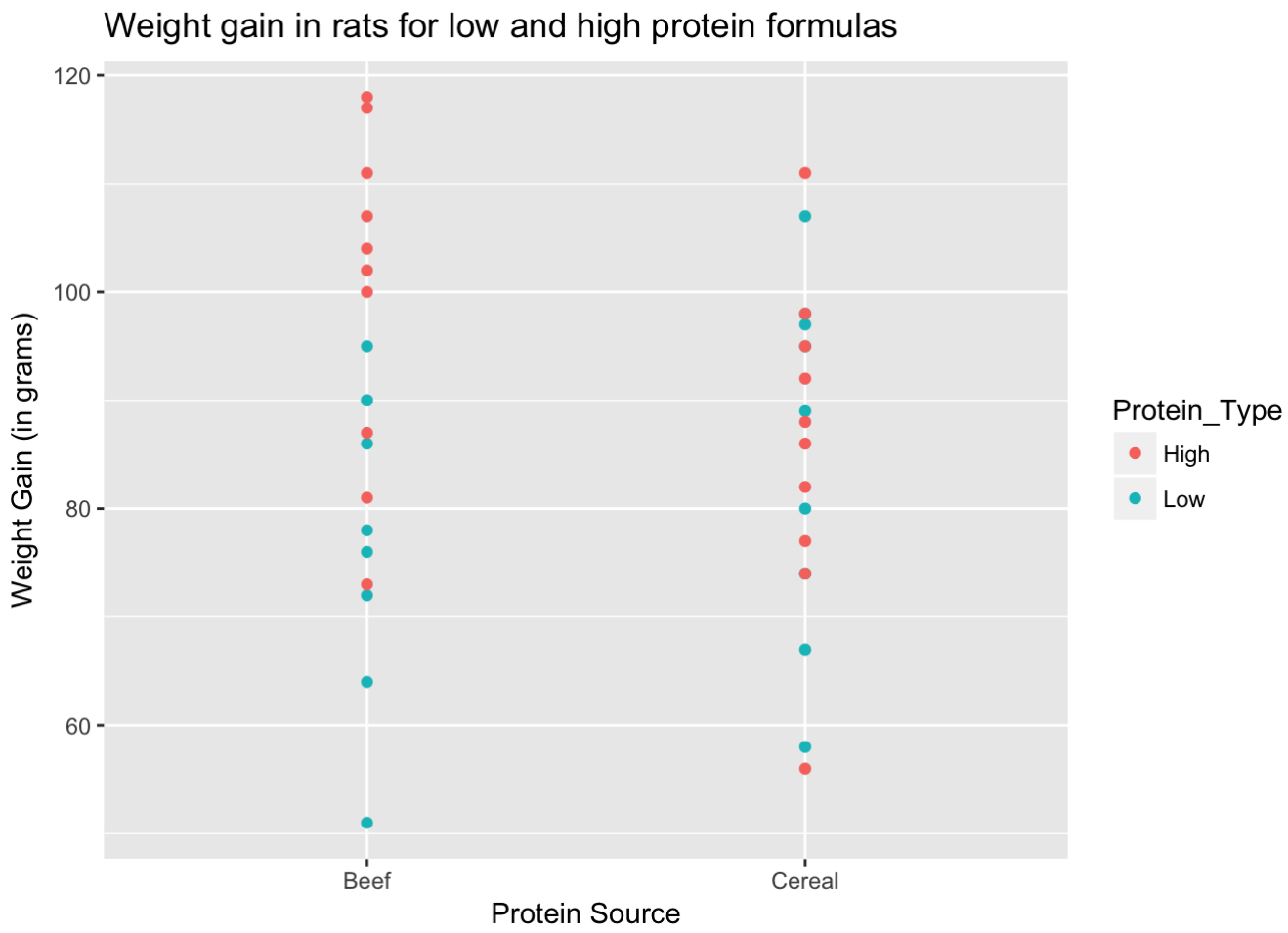
We've been talking a lot about basketball players lately so how about we try something new? For this example, let's look at this random data set I found online titled "Rat Weight." From the documentation, it seems that this data set describes an experiment wherein the weight gain of rats were tracked after being fed either protein isolated from beef or from cereal.

```
#Reading the csv file
rats <- read.csv(file = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/
master/csv/HSAUR/weightgain.csv", col.names = c("Rat_Number", "Protein_Source", "Protein
_Type", "Weight_Gain"))

head(rats)
```

```
##   Rat_Number Protein_Source Protein_Type Weight_Gain
## 1          1           Beef          Low          90
## 2          2           Beef          Low          76
## 3          3           Beef          Low          90
## 4          4           Beef          Low          64
## 5          5           Beef          Low          86
## 6          6           Beef          Low          51
```

First, let's create a basic plot by using ggplot in a way that we already are familiar with. In the following code we're going to make a basic scatterplot to visualize the relationship between protein type and weapon.

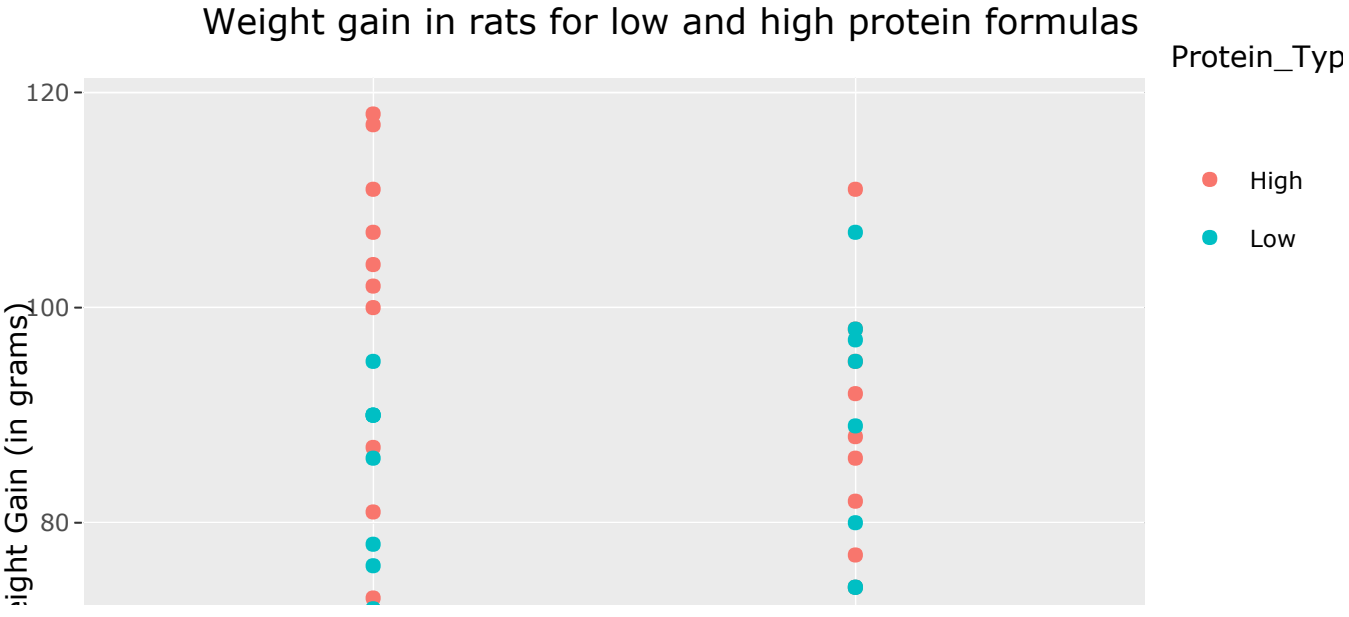
```
#Creating our basic graph with ggplot
rat_plot <- ggplot(data = rats, aes(x = Protein_Source, y = Weight_Gain)) + geom_point(a
es(color = Protein_Type)) + labs(title = "Weight gain in rats for low and high protein f
ormulas", x = "Protein Source", y = "Weight Gain (in grams)")
rat_plot
```



Now let's make our ggplot interactive! All that's required is passing in the plot we just made into the ggplotly function.

```
#Making our rat plot interactive
ggplotly(rat_plot)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```



Changing point labels

Hover your mouse over any of the points in that previous graph and you'll see that you have access to the specific data of that point. It will default to whatever aspects are passed into the graph, but we can customize these hovers pretty simply.

First, we're going to create a text vector with all of the information we want to include. We'll call it "rat_info," and we'll pare it down to tell us just what protein source it is, how much weight the rat gained, and the phrase "Great Rat!" to tell us what kind of rat it was (a great one).

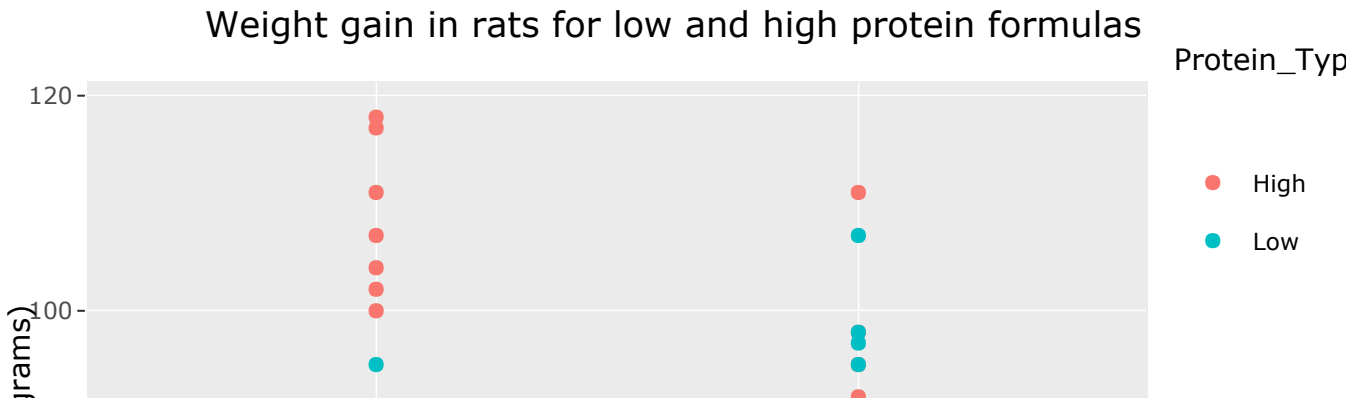
```
#We create a text vector using the paste function, which concatenates vectors. The "\n"
 allows us to separate our data in the hovers onto different lines, and we designate an
 empty string for our separator because we have those already factored into the strings
 we passed in.
rat_info <- paste("Protein Source = " , rats$Protein_Source, "\n", "Weight Gain = ", rat
s$Weight_Gain, "\n", "Great rat!", sep = "")
```

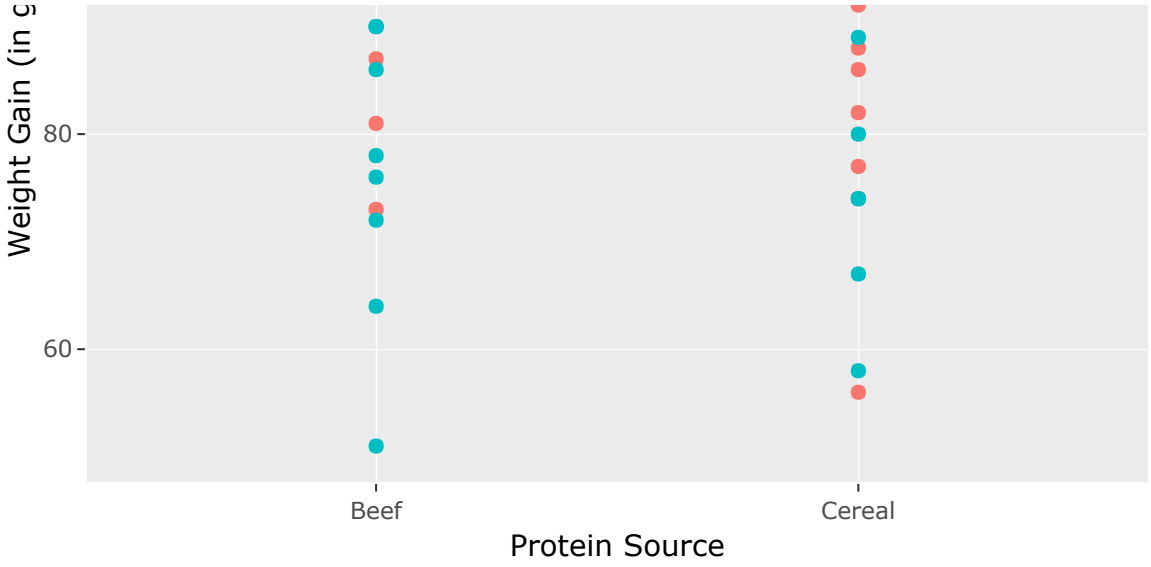
Next, we create the relabeled plot using the text vector we just created, and use the function style() and pass in a numeric vector with the numbers of the traces we want to modify, in this case a vector from 1 to 2 (for our 2 categories, Beef and Cereal)

```
#Creating a copy of the graph we made previously
rat_plot_relabeled <- ggplotly(rat_plot)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```

```
#Adding new labels to the new graph we created
style(rat_plot_relabeled, text = rat_info, traces = c(1, 2))
```





Interactive Tools in a plotly graph

Let's take a moment to appreciate the interactive tools in plotly! Hover over your graph and you'll see some clickable icons on the top on your graph. Below is a brief overview of all the features and their respective functions:

- **Screenshot** - Downloads plot as png
- **Zoom** - Zooms to the square selection you drag out on the graph
- **Pan** - Allows you to pan across the graph when you're zoomed in
- **Box Select** - Greys out all data except for those in the selected box
- **Lasso Select** - Greys out all data except for those in freely formed selection
- **Zoom In** - Zooms in to the graph
- **Zoom Out** - Zooms out of the graph
- **Autoscale** - Scales view to fit all data in frame
- **Reset Axes** - Returns graph to original scale
- **Toggle Spike Lines** Shows trace lines for each point to x and y axes when hovered
- **Show closest data on hover** Shows information for closest data when toggled (versus comparison)
- **Compare data on hover** Shows a comparison of data between two points when toggled (versus closest data)

You should see now how powerful this can be for your own analysis of the data and for your readers! Interactive plots have the ability to be several plots at once, and are great for both exploration and presentation.

A girly example

For our next couple of examples, let's load another data set that we'll call "womensrights". The data documents the answers of randomly selected men and women of varying years of education to the question:

> "Agree or disagree: women should take care of running their homes and leave running the country up to men."

Note here that agreeing would indicate that the subject affirmed traditional gender roles and that disagreeing would indicate the opposite.

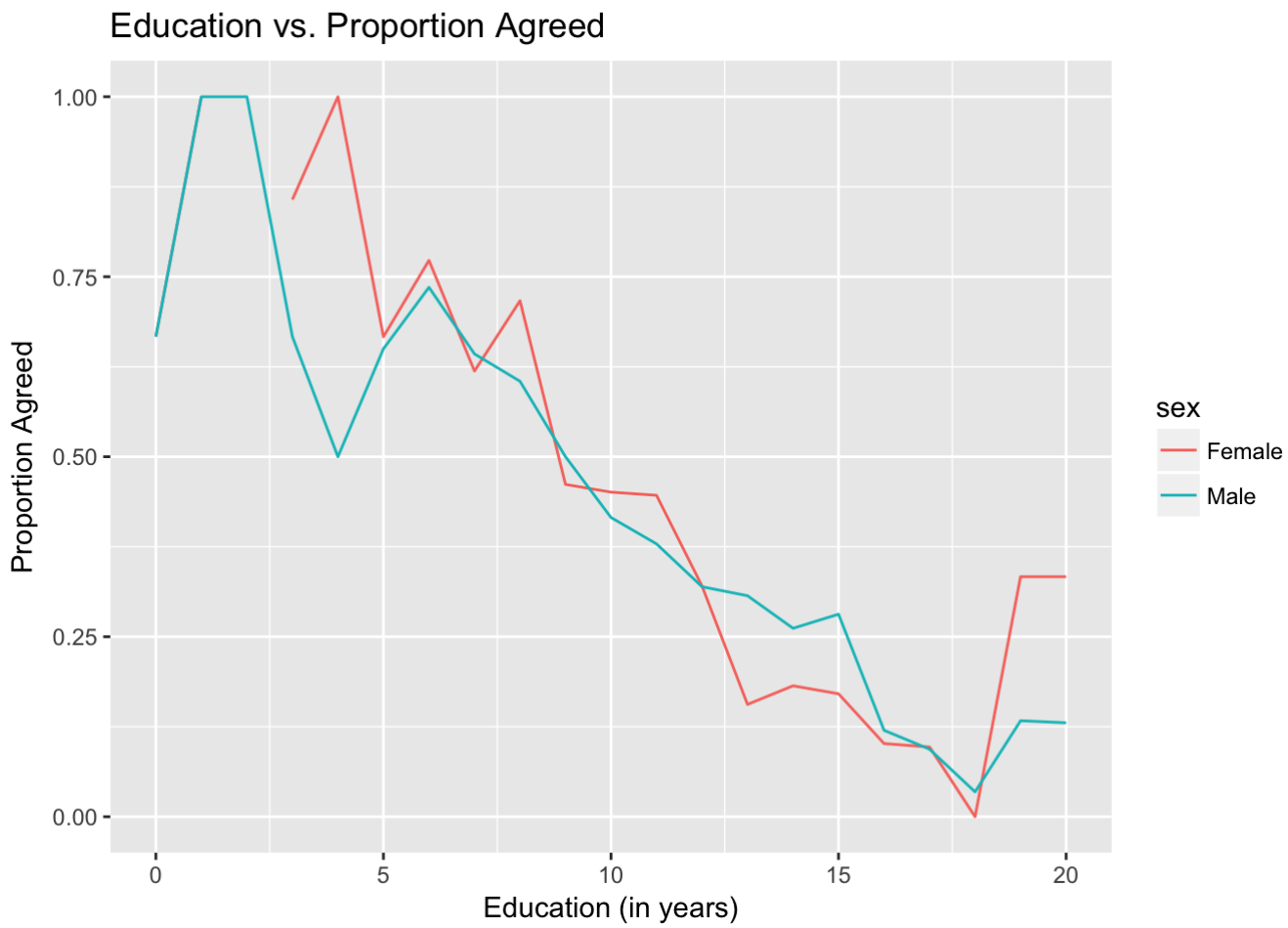
```
#Making the womensrights data frame
women <- read.csv(file = "https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/master/csv/HSAUR/womensrole.csv")

#Adding a variable "proportion" to the data frame since the number of men and women at each education level differ.
womensrights <- cbind(women, "proportion" = (women$agree / (women$agree + women$disagree)))

head(womensrights)
```

```
##   X education  sex agree disagree proportion
## 1 1         0 Male     4        2  0.6666667
## 2 2         1 Male     2        0  1.0000000
## 3 3         2 Male     4        0  1.0000000
## 4 4         3 Male     6        3  0.6666667
## 5 5         4 Male     5        5  0.5000000
## 6 6         5 Male    13        7  0.6500000
```

A Line Plot

Any plot that you can make with ggplot you can make interactive! Let's start with an example of a line plot.

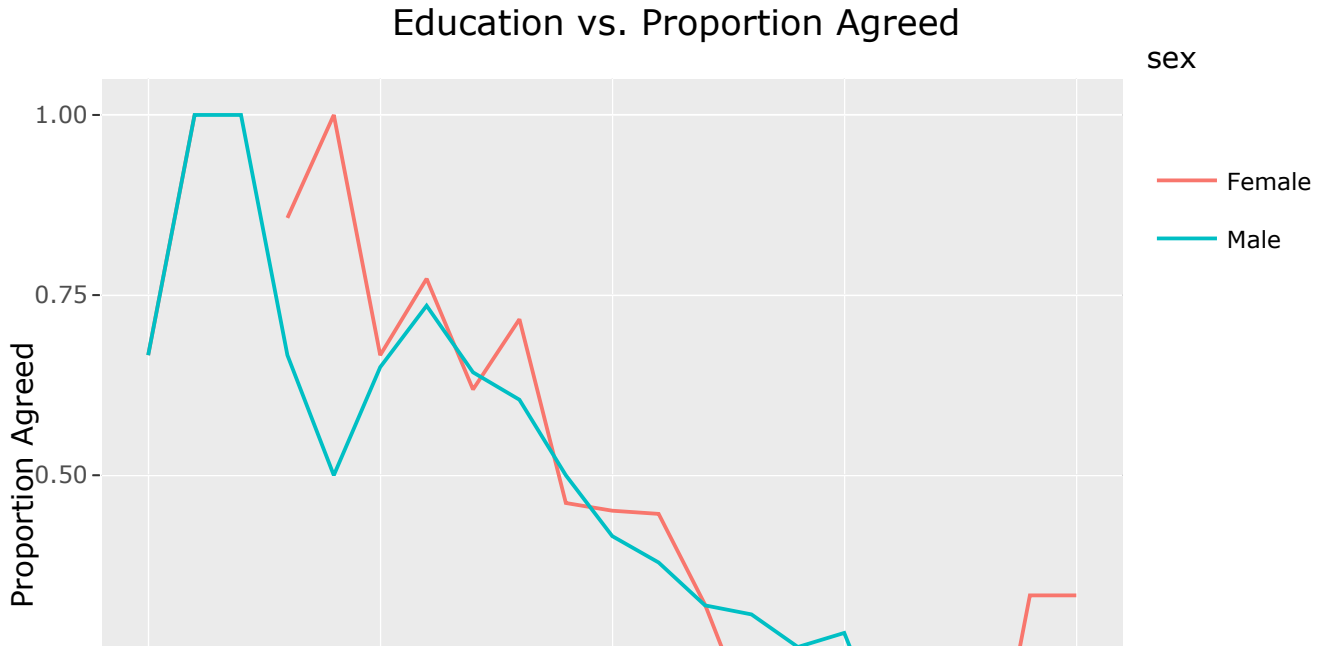
```
#Creating a line plot to compare years of education and that proportion that agreed with the statement that "Women should take care of running their homes and leave running the country up to men."

women_line <- ggplot(data = womensrights, aes(x = education, y = proportion, color = sex)) + geom_line(aes(group = sex)) + labs(title = "Education vs. Proportion Agreed", x = "Education (in years)", y = "Proportion Agreed")

women_line
```

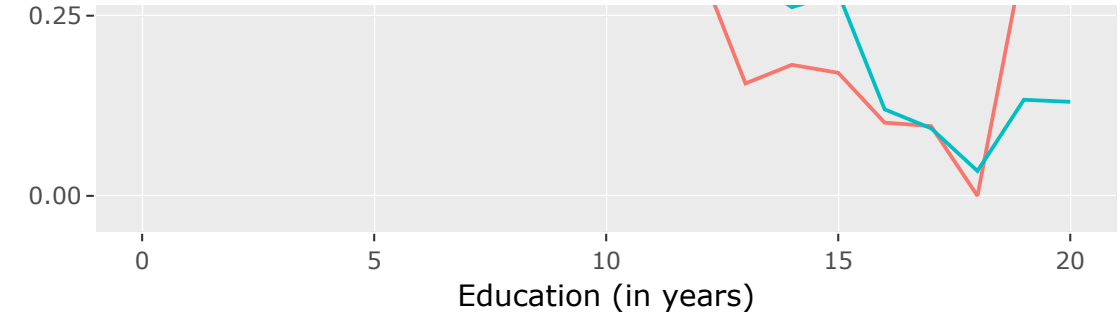


Again, let's make it interactive with out package plotly!

```
#Making the graph interactive
ggplotly(women_line)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```





Here we see that our “compare data on hover” function really shines! With an interactive line plot we can compare corresponding points a lot easier than if we had to look up the data in the table afterwords, making it an extremely powerful tool for forming your own conclusions about the data as well as showcasing those differences to your readers.

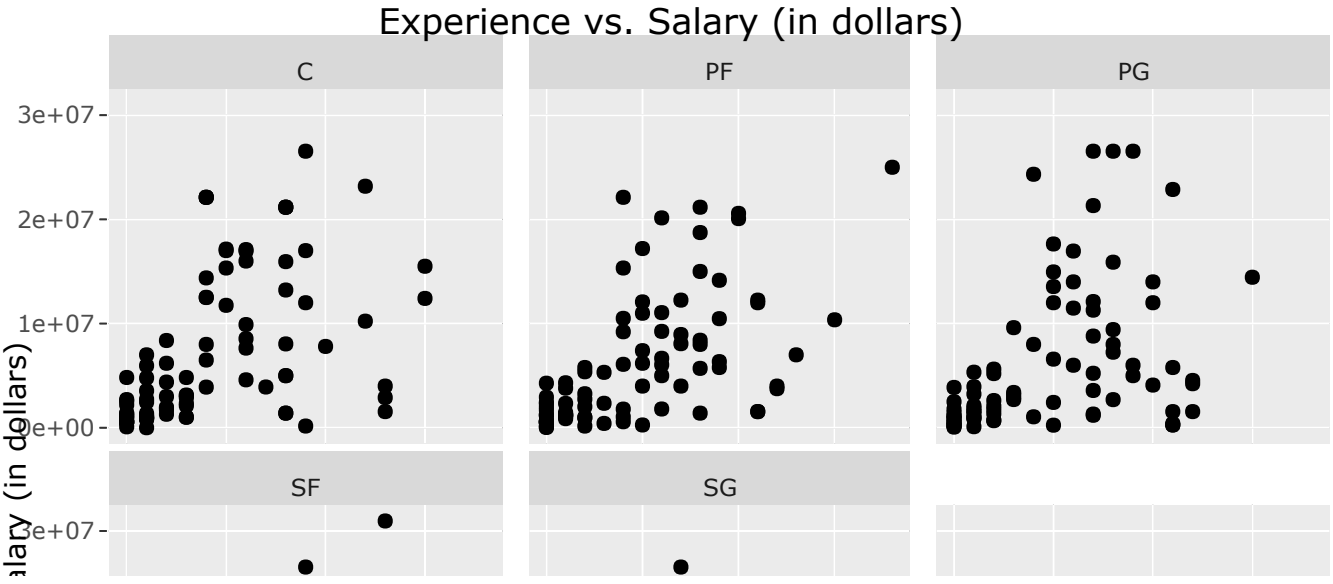
Wrapped graphs

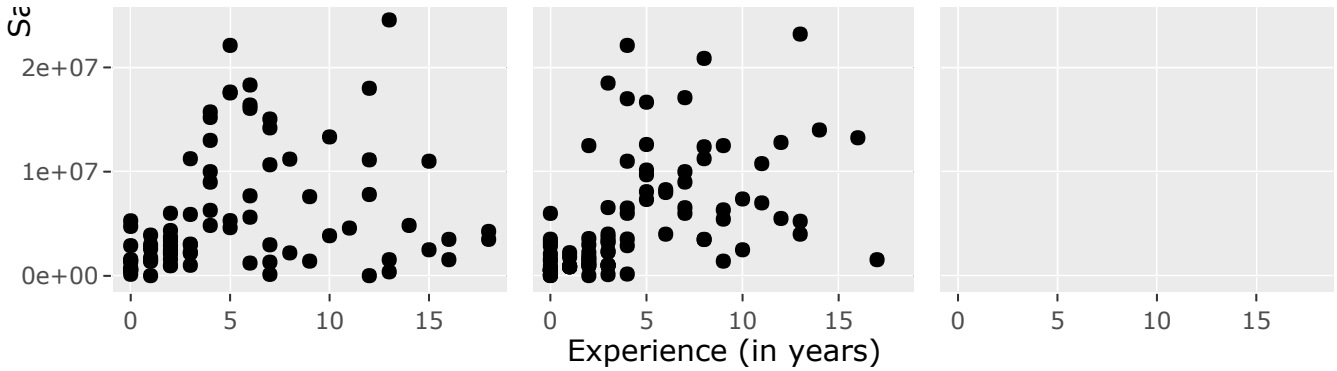
Want to make multiple wrapped graphs interactive too? We do the same thing that we’ve done, which is make the plot in ggplotly and pass it into ggplotly to make it interactive.

I’m feeling nostalgic for our ol’ NBA data so let’s use it for this example.

```
#Loading nba data
nba <- read.csv(file = "https://raw.githubusercontent.com/ucb-stat133/stat133-fall-2017/
master/data/nba2017-roster.csv")

#Creating the facet-wrapped ggplot
nba_wrap <- ggplot(data = nba, aes(x = experience, y = salary)) +
  geom_point() +
  facet_wrap(~ position) + labs(title = "Experience vs. Salary (in dollars)", x = "Exper
ience (in years)", y = "Salary (in dollars)")

#Making that plot interactive
ggplotly(nba_wrap)
```

```
## We recommend that you use the dev version of ggplot2 with `ggplotly()`
## Install it with: `devtools::install_github('hadley/ggplot2')`
```





We see that the graphs here all have interactive functionality, and all the interactive tools work for individual graphs. Neat!

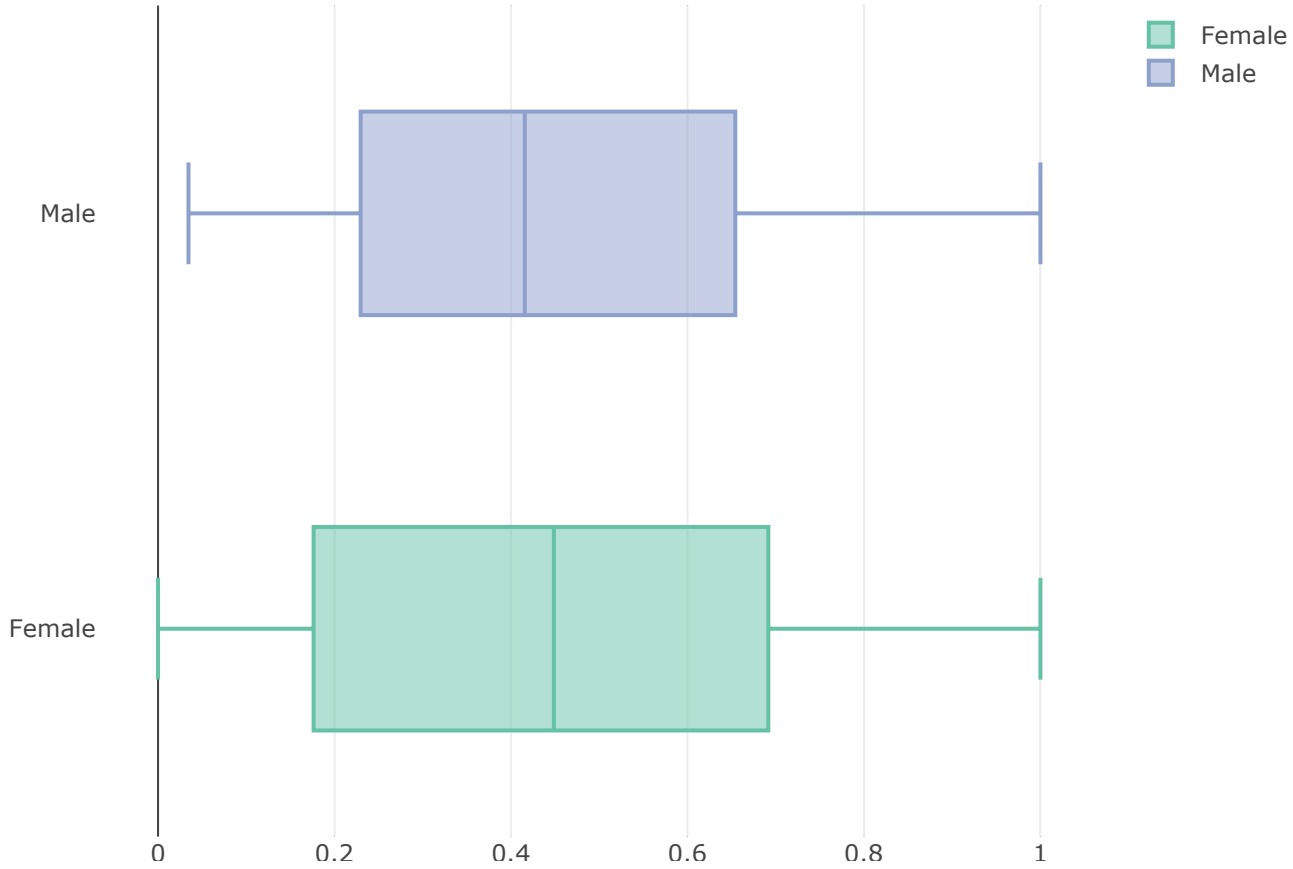
Making plots purely in plotly

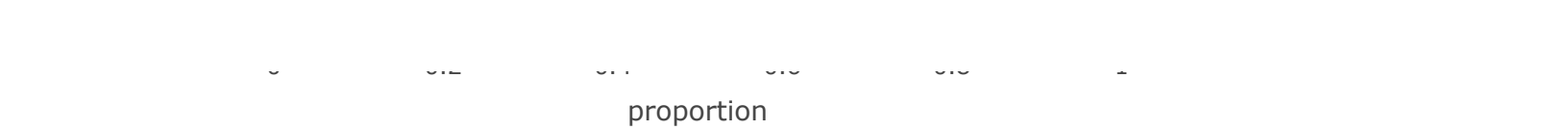
So far we've only used plotly to convert existing ggplots into interactive ones, but we can also make them directly with the command plot_ly()! Let's make a box plot on our womensrights data using the plot_ly package.

```
#Making the interactive plot with plot_ly directly
women_box <- plot_ly(womensrights, x = ~proportion, color = ~sex, type = "box")
women_box
```

```
## Warning: Ignoring 1 observations
```

```
## Warning in RColorBrewer::brewer.pal(N, "Set2"): minimal value for n is 3, returning r
equested palette with 3 different levels
```





Notice that the hover interactions for different types of plots are different! Here with the box plot, hovering gives us the values for upper and lower bounds, quartiles, etc, instead of the data for individual points like it does for scatter and line plots. Each type of graph is going to give you different options to interact with the data based on what the graph type is meant to show, so take a minute to explore different examples of graphs in the links provided below!

Want more information on interactive plots?

For more information on the different types of interactive plots you can make in plotly (which include interactive maps and 3D interactive surface plots): http://www.r-graph-gallery.com/portfolio/interactive-r-graphics/ (http://www.r-graph-gallery.com/portfolio/interactive-r-graphics/)

For more information on layout and styling: https://plot.ly/r/reference/#Layout_and_layout_style_objects (https://plot.ly/r/reference/#Layout_and_layout_style_objects)

For interesing data sets: https://vincentarelbundock.github.io/Rdatasets/datasets.html (https://vincentarelbundock.github.io/Rdatasets/datasets.html)

Thanks for reading, have fun making all your plots interactive!

Other Sources:

https://plot.ly/r/getting-started/#installation (https://plot.ly/r/getting-started/#installation)

http://www.r-graph-gallery.com/126-add-segment-to-a-plotly-graphic/ (http://www.r-graph-gallery.com/126-add-segment-to-a-plotly-graphic/)

http://ouzor.github.io/blog/2014/11/21/interactive-visualizations.html (http://ouzor.github.io/blog/2014/11/21/interactive-visualizations.html)

http://www.r-graph-gallery.com/108-multi-scatterplot-plotly/ (http://www.r-graph-gallery.com/108-multi-scatterplot-plotly/)

https://moderndata.plot.ly/interactive-r-visualizations-with-d3-ggplot2-rstudio/ (https://moderndata.plot.ly/interactive-r-visualizations-with-d3-ggplot2-rstudio/)

http://www.rebeccabarter.com/blog/2017-04-20-interactive/ (http://www.rebeccabarter.com/blog/2017-04-20-interactive/)

http://www.r-graph-gallery.com/portfolio/interactive-r-graphics/ (http://www.r-graph-gallery.com/portfolio/interactive-r-graphics/)