Data Visualization in R

Neil S. Williams*

Data Visualization in R
September 10, 2021

Contents

*University of Georgia, snpwill@uga.edu

Getting started

The purpose of this tutorial is to show how to display and visualize data using R. We will start with plots of means and distributions, then move to regression, and finally some more advanced topics.

Please copy code from the R script used to produce this tutorial; this script can be found here.

Let's start by loading some base packages that are always useful when working in RStudio:

```
library(tidyverse)
library(dplyr)
library(ggplot2)
library(knitr)
```

-Let's start with frog weight

```
library(rio)
frog_weight.data <- rio::import("frog_weight.dta")
```

Let's check the structure of these data

Examine data

```
#Check the structure of the data with the str() command
str(frog_weight.data) #note that "Lake" iscoded as 1,2,3 or Erie, Huron, Ontario
```

```
## 'data.frame':    200 obs. of  3 variables:
##  $ frog_weight: num  35 33 39 37 31 36 36 31 41 37 ...
##   ..- attr(*, "label")= chr "weight of frog in grams"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ num_flies  : num  26 36 42 32 51 39 46 31 41 31 ...
##   ..- attr(*, "label")= chr "average number of flies near frog's habitat, 1 square km"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ lake       : num  2 2 2 2 2 2 2 3 2 2 ...
##   ..- attr(*, "label")= chr "lake of residence"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##   ..- attr(*, "labels")= Named num [1:3] 1 2 3
##   .. ..- attr(*, "names")= chr [1:3] "Erie" "Huron" "Ontario"
##  - attr(*, "label")= chr "fake frog data"
```

Much of data visualization is prepping the data to make sure it looks how you want it to in the plot. To start, I am going to recode values of the `Lake` variable so we know which number is associated with which lake.

```
#We can recode this and set the factor level becauese Lake is coded as a factor

frog_weight.data$lake[frog_weight.data$lake== 1] <- "Erie"
frog_weight.data$lake[frog_weight.data$lake== 2] <- "Huron"
frog_weight.data$lake[frog_weight.data$lake== 3] <- "Ontario"

#Save lake as a factor and relevel the variable
frog_weight.data$lake <- as.factor(frog_weight.data$lake)
frog_weight.data$lake <- relevel(frog_weight.data$lake, ref = "Erie")

summary(frog_weight.data$lake) # shows the number of cases of frogs in each lake
```

```
##    Erie   Huron Ontario
```

```
##      79      54      67
```

Showing this with the `table1` package

```
library(table1)
table1::table1(~.| lake, data = frog_weight.data) #the dot means we want to include all variables
```

	Erie	Huron	Ontario	Overall
	(N=79)	(N=54)	(N=67)	(N=200)
weight of frog in grams				
Mean (SD)	57.3 (7.27)	45.4 (7.90)	53.3 (9.41)	52.8 (9.48)
Median [Min, Max]	59.0 [37.0, 67.0]	44.0 [31.0, 59.0]	55.0 [31.0, 67.0]	54.0 [31.0, 67.0]
average number of flies near frog's habitat, 1 square km				
Mean (SD)	56.1 (9.32)	44.0 (8.34)	54.8 (10.4)	52.4 (10.7)
Median [Min, Max]	56.0 [31.0, 71.0]	46.0 [26.0, 61.0]	56.0 [26.0, 71.0]	52.0 [26.0, 71.0]

We also may just want to select a certain set of variables

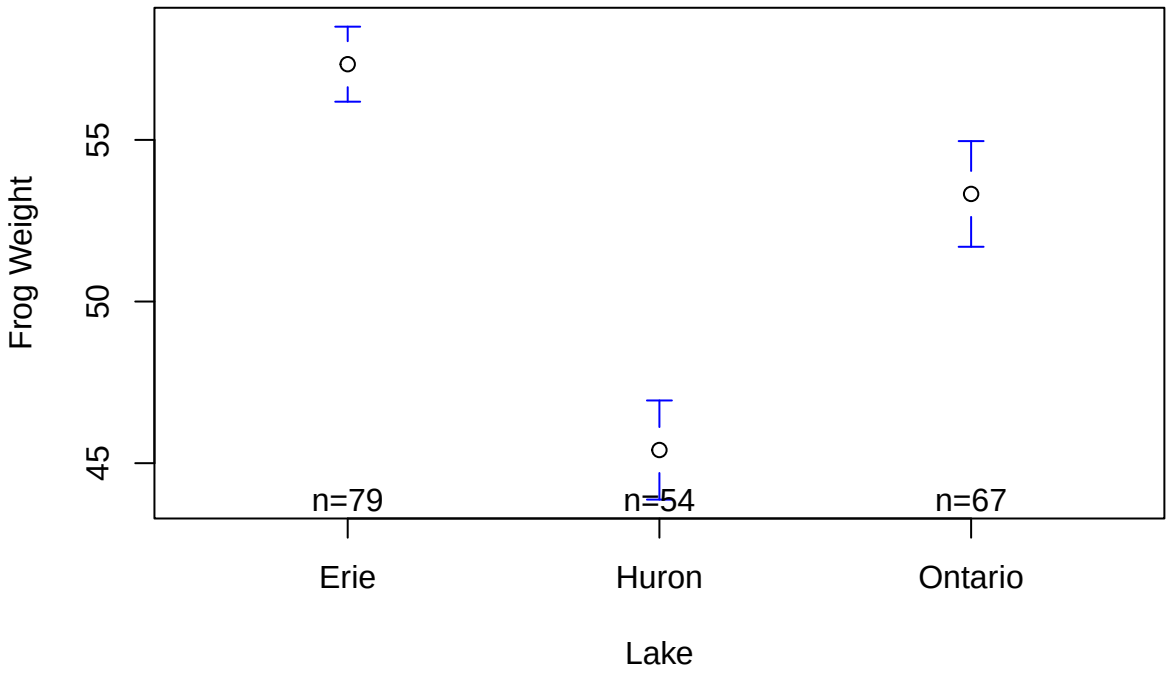
```
# we could also just select the ones we want
table1::table1(~frog_weight | lake, data = frog_weight.data)
```

	Erie	Huron	Ontario	Overall
	(N=79)	(N=54)	(N=67)	(N=200)
weight of frog in grams				
Mean (SD)	57.3 (7.27)	45.4 (7.90)	53.3 (9.41)	52.8 (9.48)
Median [Min, Max]	59.0 [37.0, 67.0]	44.0 [31.0, 59.0]	55.0 [31.0, 67.0]	54.0 [31.0, 67.0]

Plotting means

We can plot the mean for each group with

```
library(gplots)
# Plot the mean of teeth length by dose groups
plotmeans(frog_weight ~ lake, data = frog_weight.data,
          p = .84, # choose confidence level
          connect = FALSE, #whether the points should be connected
          ylab = "Frog Weight",
          xlab = "Lake")
```

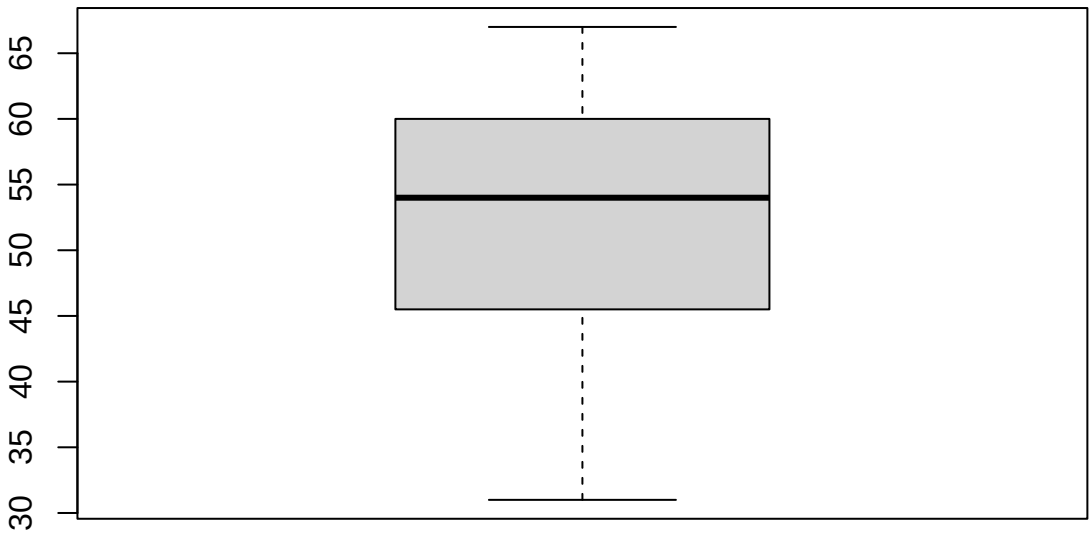


Plotting distributions

Boxplots

We can make a basic boxplot showing the range, quartile range, and mean of a variable using the `boxplot()` command

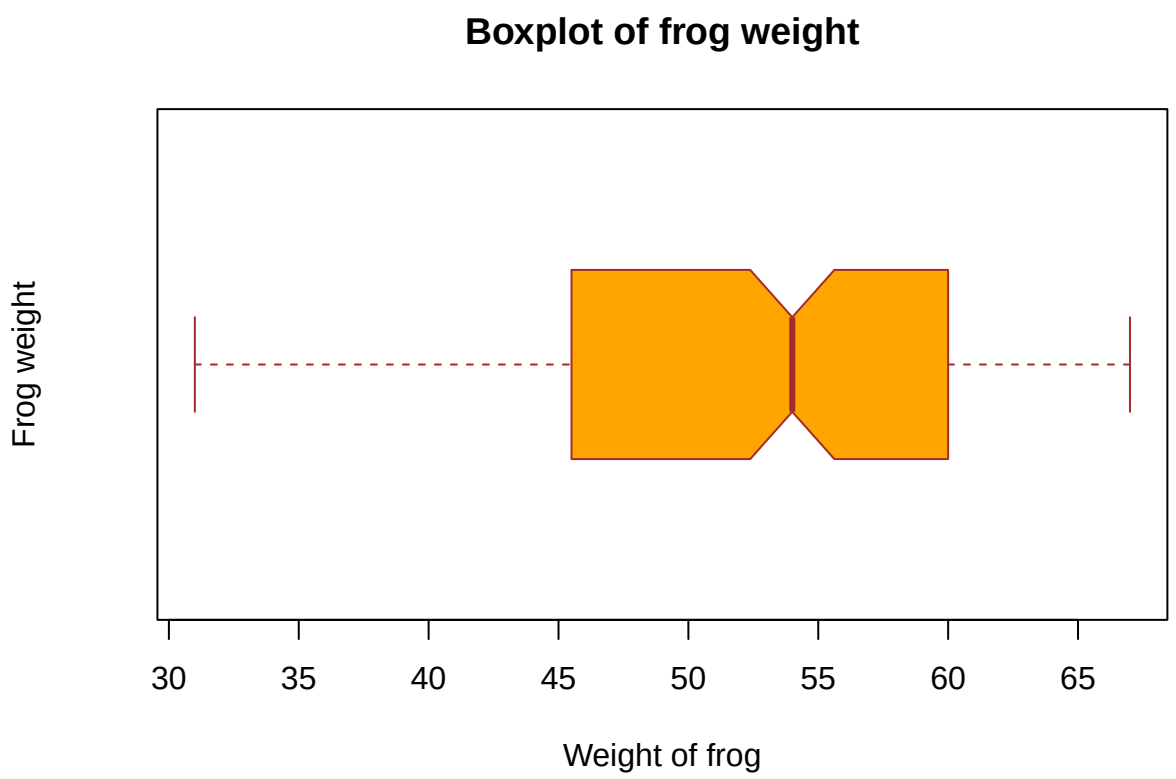
```
boxplot(frog_weight.data$frog_weight)
```



We can spruce up this figure

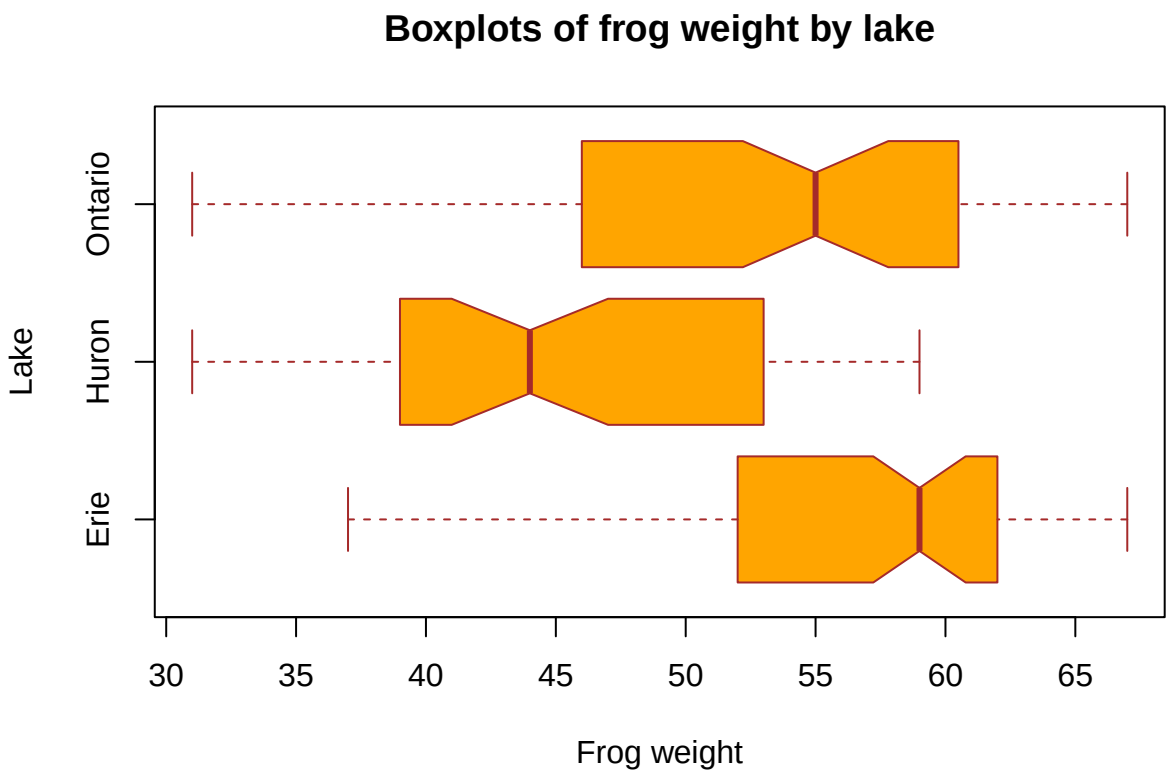
```
boxplot(frog_weight.data$frog_weight, #variable to use
main = "Boxplot of frog weight", #main title
xlab = "Weight of frog", #label on the x axis
```

```
ylab = "Frog weight", #label on the y axis
col = "orange", #fill color
border = "brown", #color of the border of the box
horizontal = TRUE, #make the plot horizontal
notch = TRUE #Creates triangular notch by mean
)
```



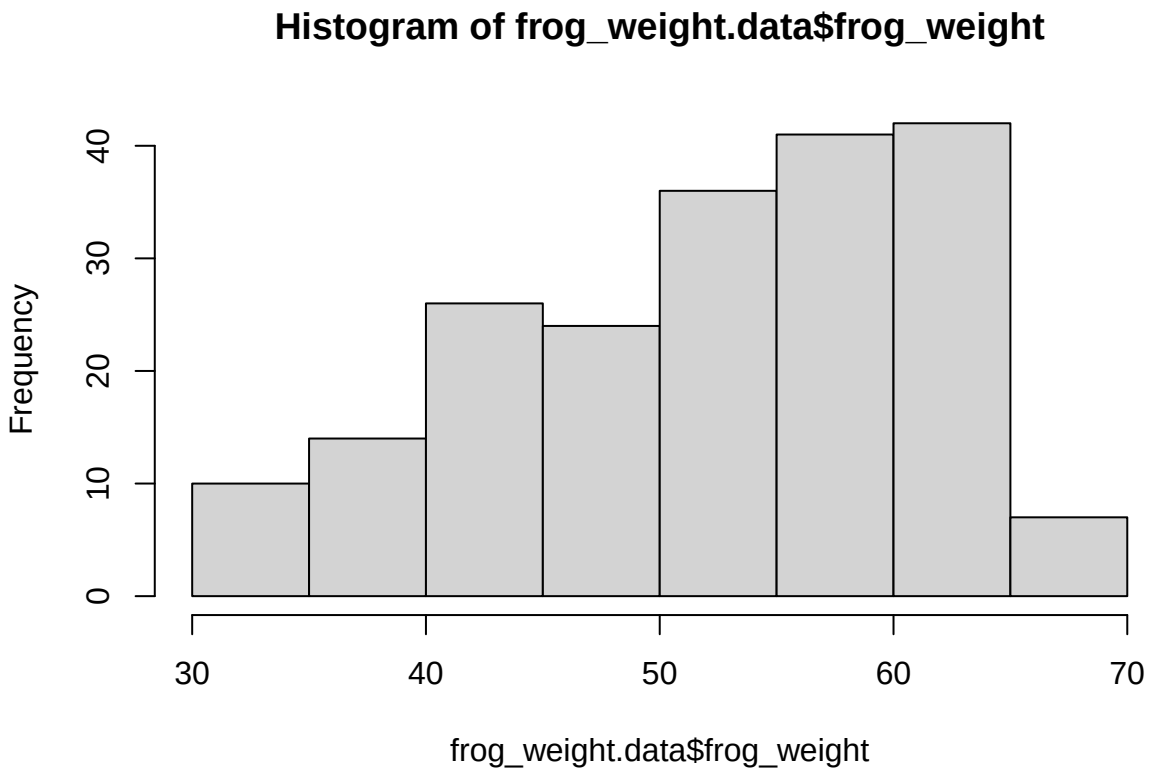
Use boxplot to compare between lakes

```
boxplot(frog_weight~lake, #frog weight plotted for each lake
data=frog_weight.data, #data
main="Boxplots of frog weight by lake",
xlab="Frog weight",
ylab="Lake",
col="orange",
border="brown",
horizontal = TRUE,
notch = TRUE
)
```

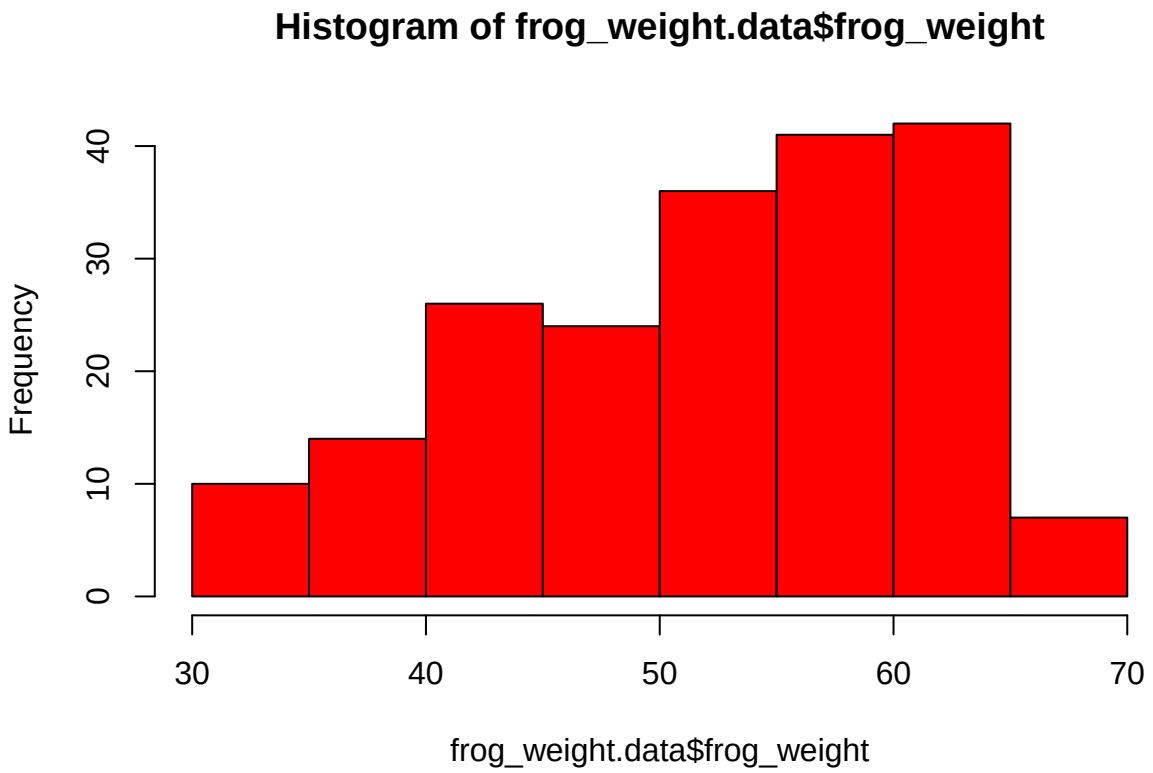


Histograms

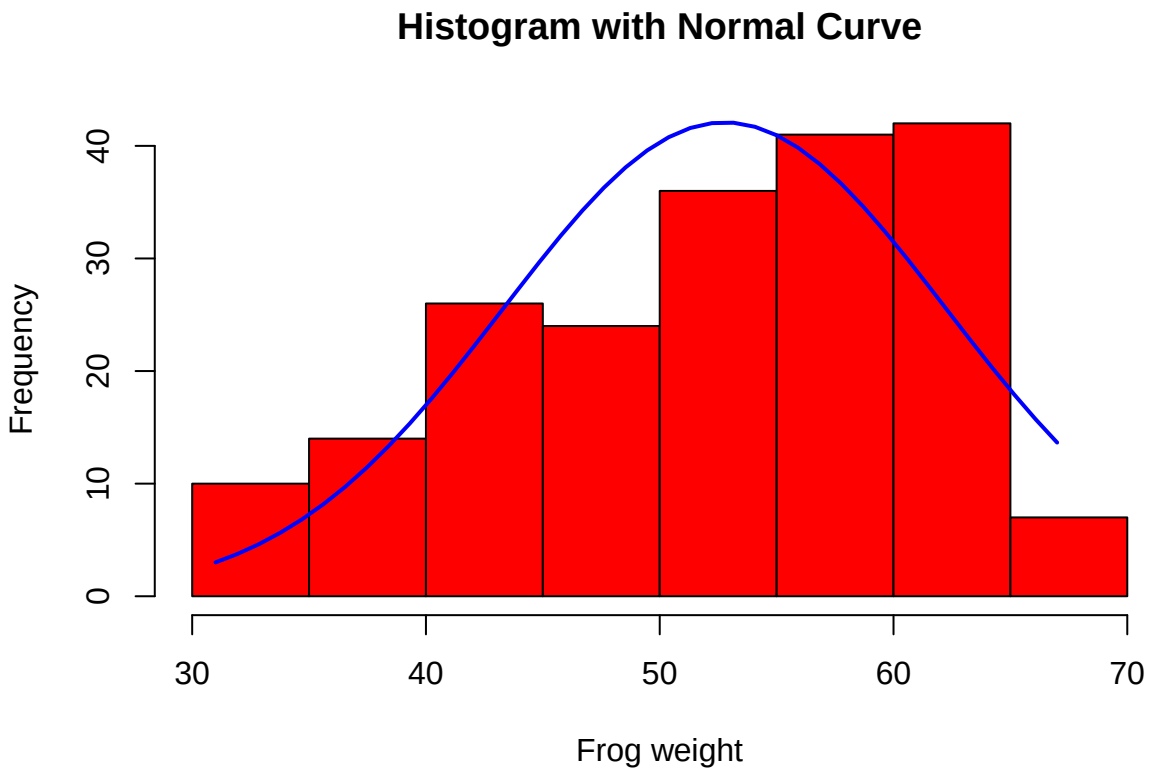
```
hist(frog_weight.data$frog_weight)
```



```
hist(frog_weight.data$frog_weight, breaks=10, col="red")
```



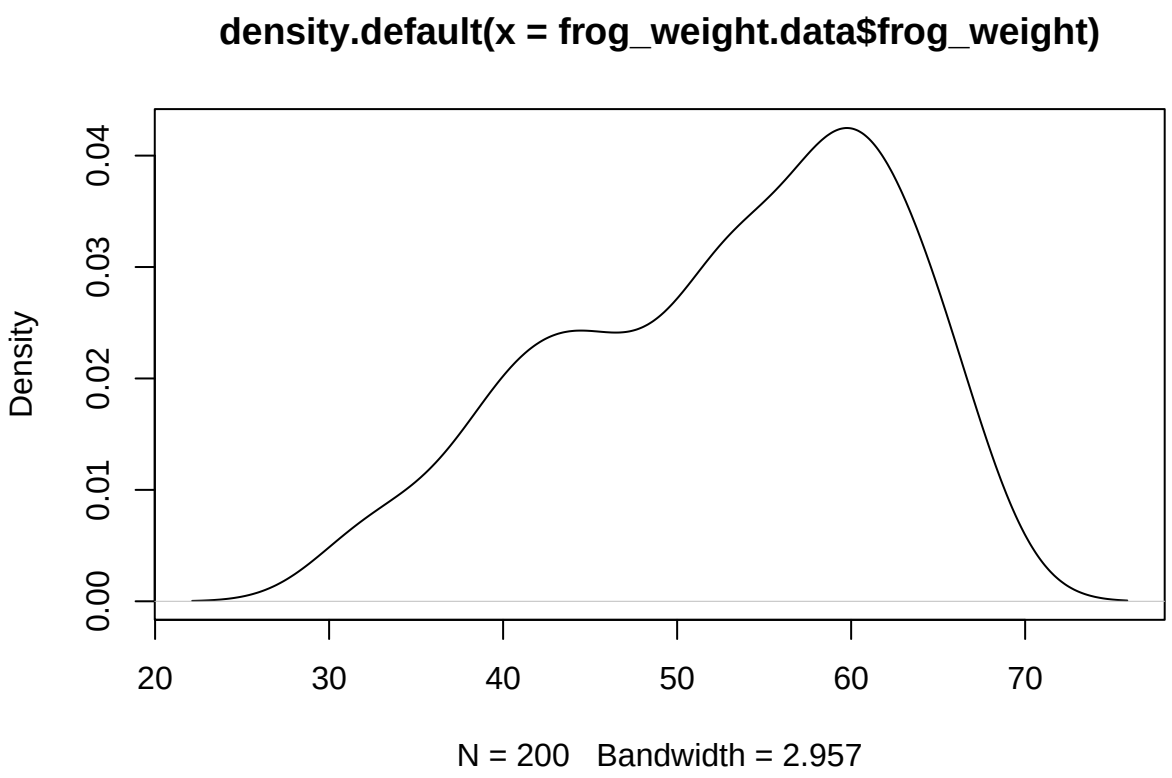

```
# Add a Normal Curve (Thanks to Peter Dalgaard)
x <- frog_weight.data$frog_weight
h<-hist(x, breaks=10, col="red", xlab="Frog weight",
   main="Histogram with Normal Curve")
xfit<-seq(min(x),max(x),length=40)
yfit<-dnorm(xfit,mean=mean(x),sd=sd(x)) #
yfit <- yfit*diff(h$mids[1:2])*length(x)
lines(xfit, yfit, col="blue", lwd=2)
```



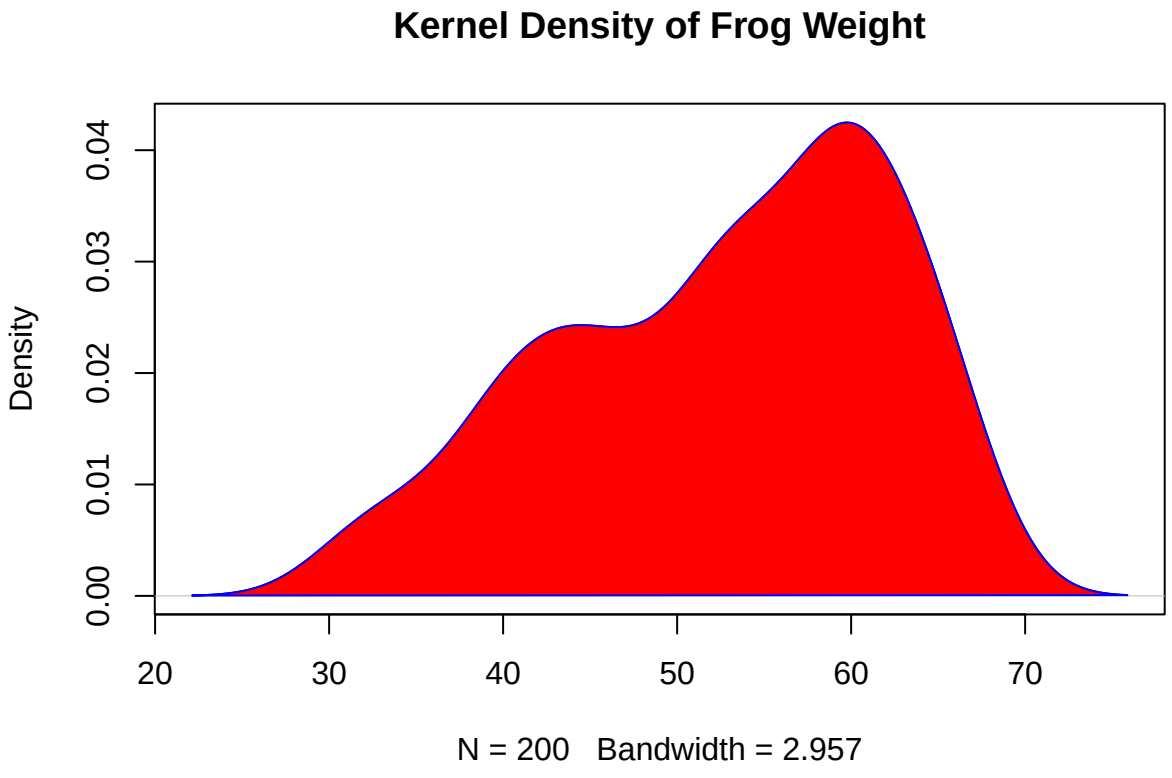
Density plots

We can also plot the density as well

```
plot(density(frog_weight.data$frog_weight))
```



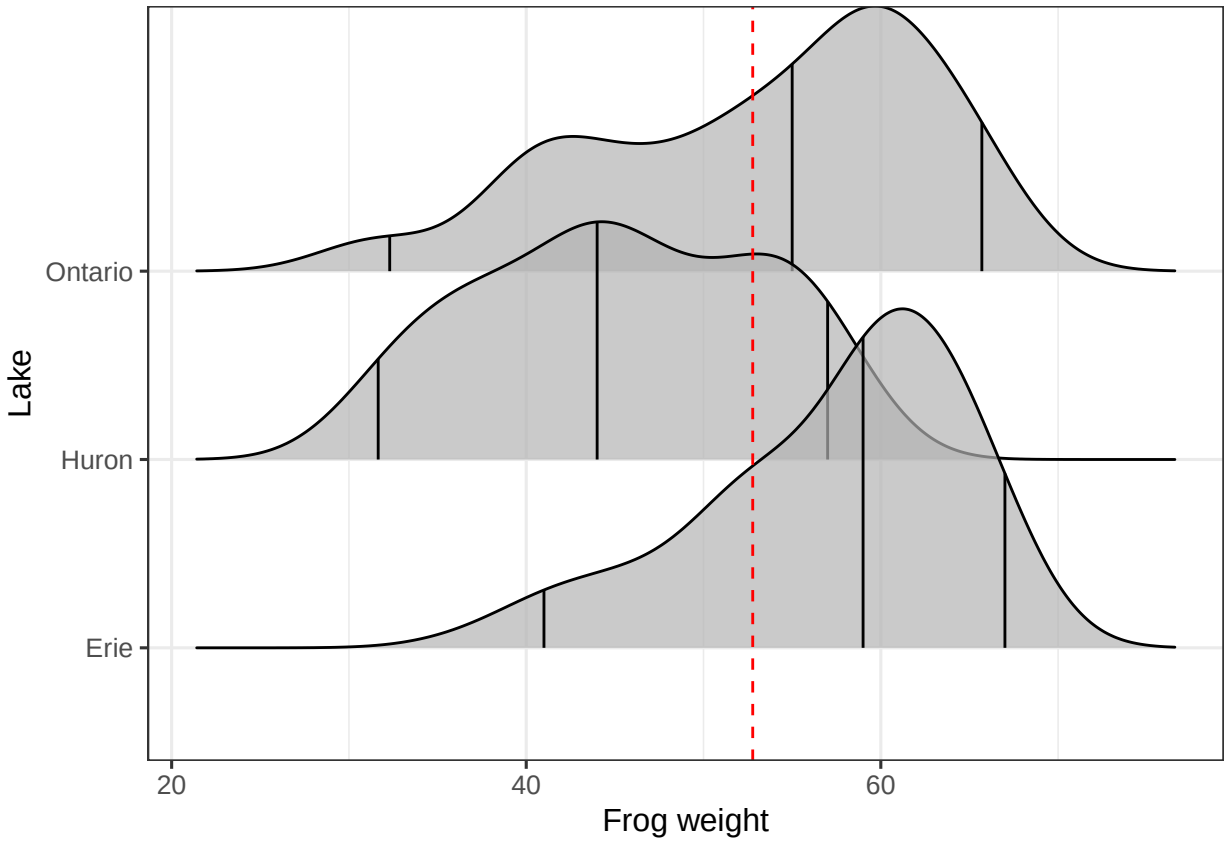

```
d <- density(frog_weight.data$frog_weight)
plot(d, main="Kernel Density of Frog Weight")
polygon(d, col="red", border="blue") # adds fill for the density plot
```



Density Ridges

Comparisons using density ridges can be even more intuitive

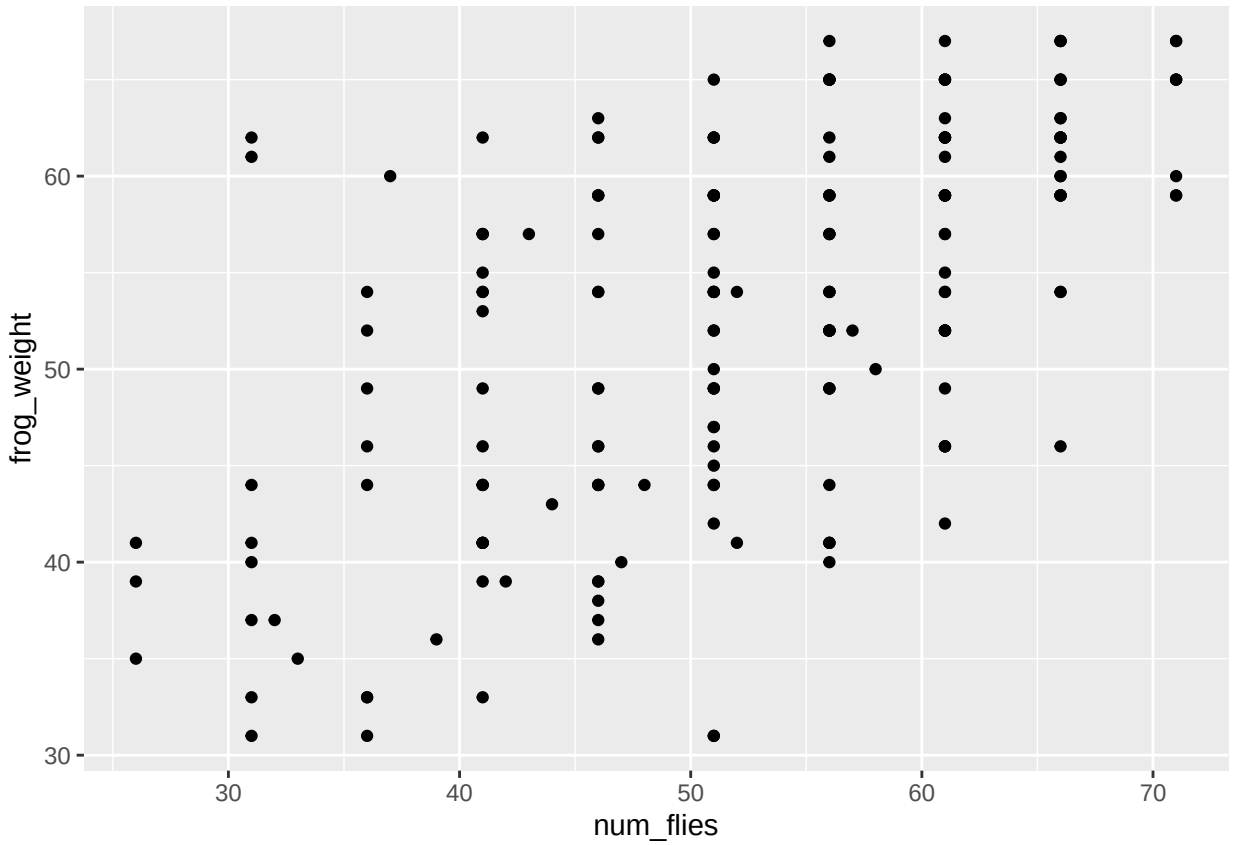
```
library(ggridges)
library(ggplot2)
ggplot(data=frog_weight.data, aes(x =frog_weight, y = lake)) + #data and base aesthetics
     theme_bw(12) +
  stat_density_ridges( quantile_lines = TRUE,
                       quantiles= c(0.025, 0.5, 0.975), alpha= 0.7) +
  geom_vline(xintercept = mean(frog_weight.data$frog_weight),
             linetype = "dashed", col = "red") +    #mean line
  xlab("Frog weight") +ylab("Lake")
```



Points and lines

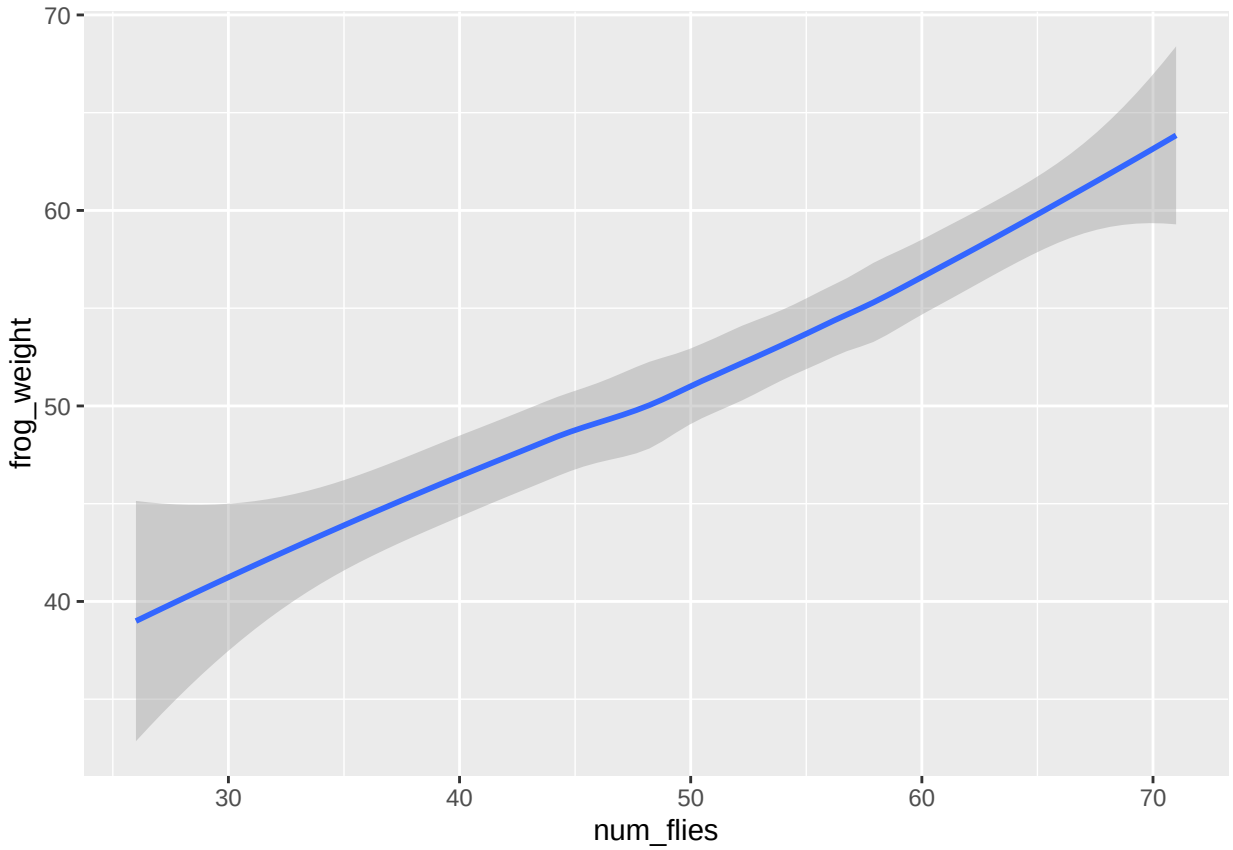
Using `geom_point` we can plot points

```
ggplot(data = frog_weight.data) +
  geom_point(mapping = aes(x =  num_flies, y = frog_weight)) # aes = aesthetics
```

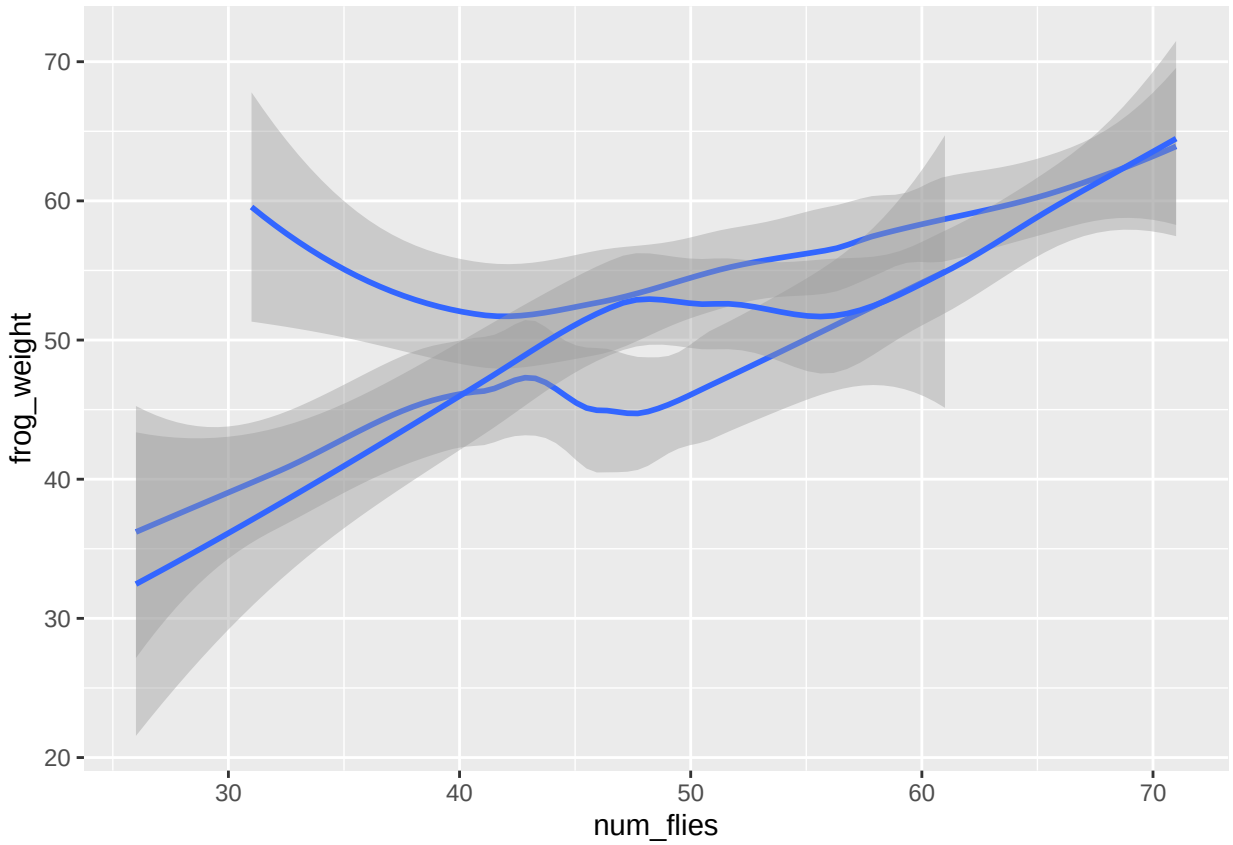


We can show a smoothed line here from the data

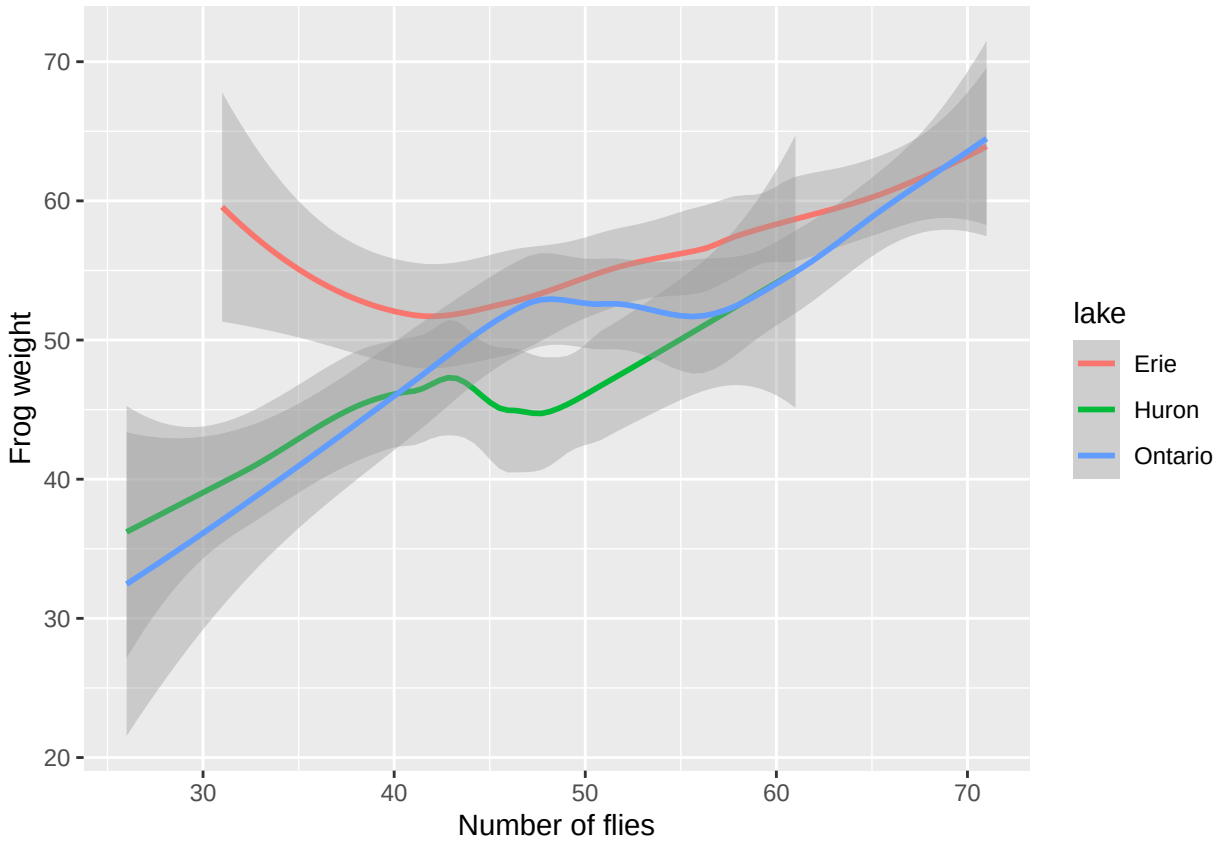
```
ggplot(data = frog_weight.data) +
  geom_smooth(mapping = aes(x =  num_flies, y = frog_weight)) # aes = aesthetics
```



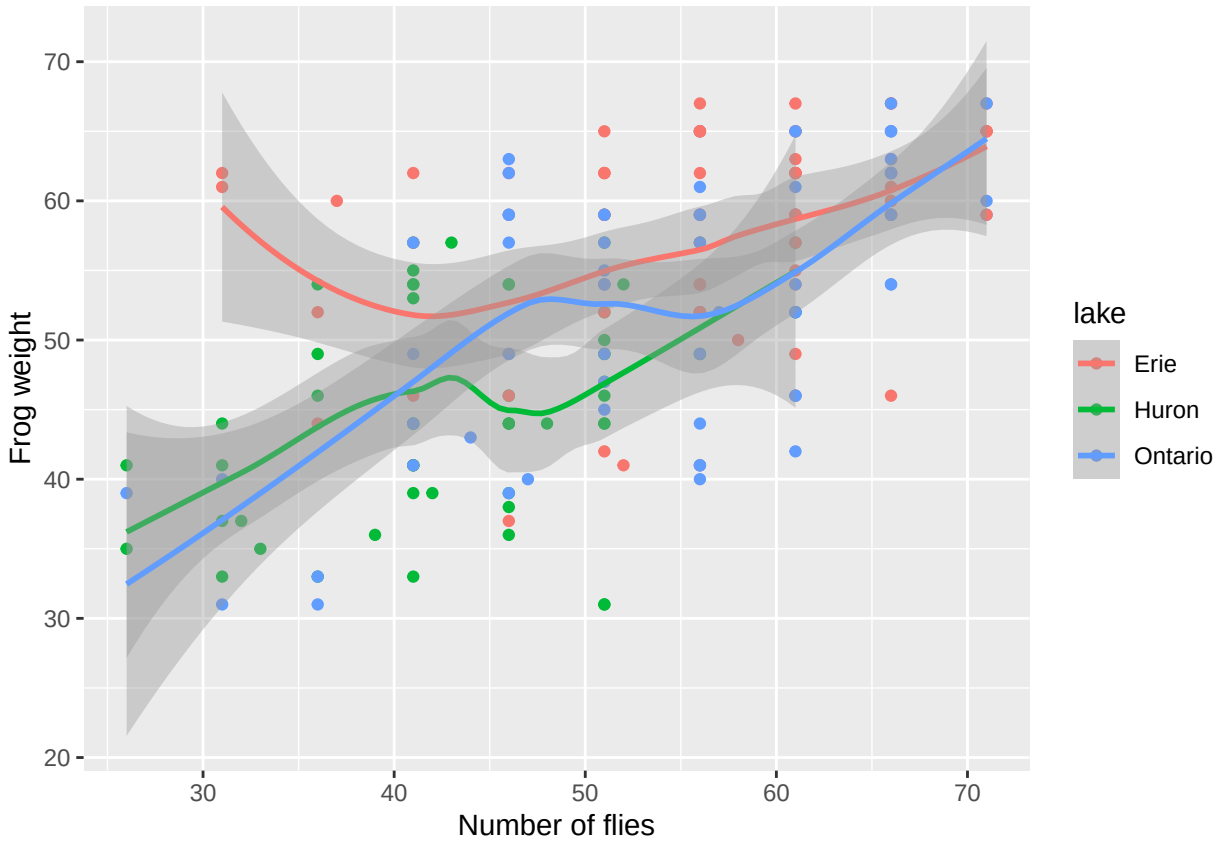

```
ggplot(data = frog_weight.data) +
  geom_smooth(mapping = aes(x =  num_flies, y = frog_weight, group = lake)) # aes = aesthetics
```




```
ggplot(data = frog_weight.data) + #our data
  geom_smooth(
    mapping = aes(x = num_flies, y = frog_weight, color = lake), # color is our grouping variable
    show.legend = TRUE # say we want legend
  ) +
  xlab("Number of flies") + #x-label
  ylab("Frog weight") #y-label
```




```
ggplot(data = frog_weight.data) + #our data
    geom_point(mapping = aes(x =  num_flies, y = frog_weight, color = lake))+ # aes = aesthetics

  geom_smooth(
    mapping = aes(x = num_flies, y = frog_weight, color = lake), # color is our grouping variable
    show.legend = TRUE # say we want legend
  ) +
  xlab("Number of flies") + #x-label
  ylab("Frog weight") #y-label
```



We can use `ggsave()` to save `ggplot` objects to a particular location

```
ggsave("flies_lakes_workshop.pdf",
       width = 6.5, height = 4.5, units = "in")
```

Regression

Estimate model

```
frog_base.mod <- lm(frog_weight ~ num_flies + lake, data = frog_weight.data)
summary(frog_base.mod)
```

```
##
## Call:
## lm(formula = frog_weight ~ num_flies + lake, data = frog_weight.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.3727  -4.3824  -0.2215   5.4703  15.2969
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 33.57093    3.13981  10.692  < 2e-16 ***
## num_flies    0.42362    0.05407   7.835 2.91e-13 ***
## lakeHuron   -6.80269    1.42743  -4.766 3.66e-06 ***
```

```
## lakeOntario -3.45934    1.19506  -2.895  0.00423 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.183 on 196 degrees of freedom
## Multiple R-squared:  0.4344, Adjusted R-squared:  0.4257
## F-statistic: 50.18 on 3 and 196 DF,  p-value: < 2.2e-16
```

If using Rmarkdown, we can use `tidy()` and `kable` to clean this output up

```
library(broom)
kable(tidy(frog_base.mod))
```

term	estimate	std.error	statistic	p.value
(Intercept)	33.5709336	3.1398112	10.692023	0.0000000
num_flies	0.4236175	0.0540691	7.834749	0.0000000
lakeHuron	-6.8026949	1.4274317	-4.765689	0.0000037
lakeOntario	-3.4593418	1.1950589	-2.894704	0.0042254

Export results to Latex

To export regression results to latex, the `stargazer` package is fantastic!

```
library(stargazer)
stargazer(frog_base.mod)
```

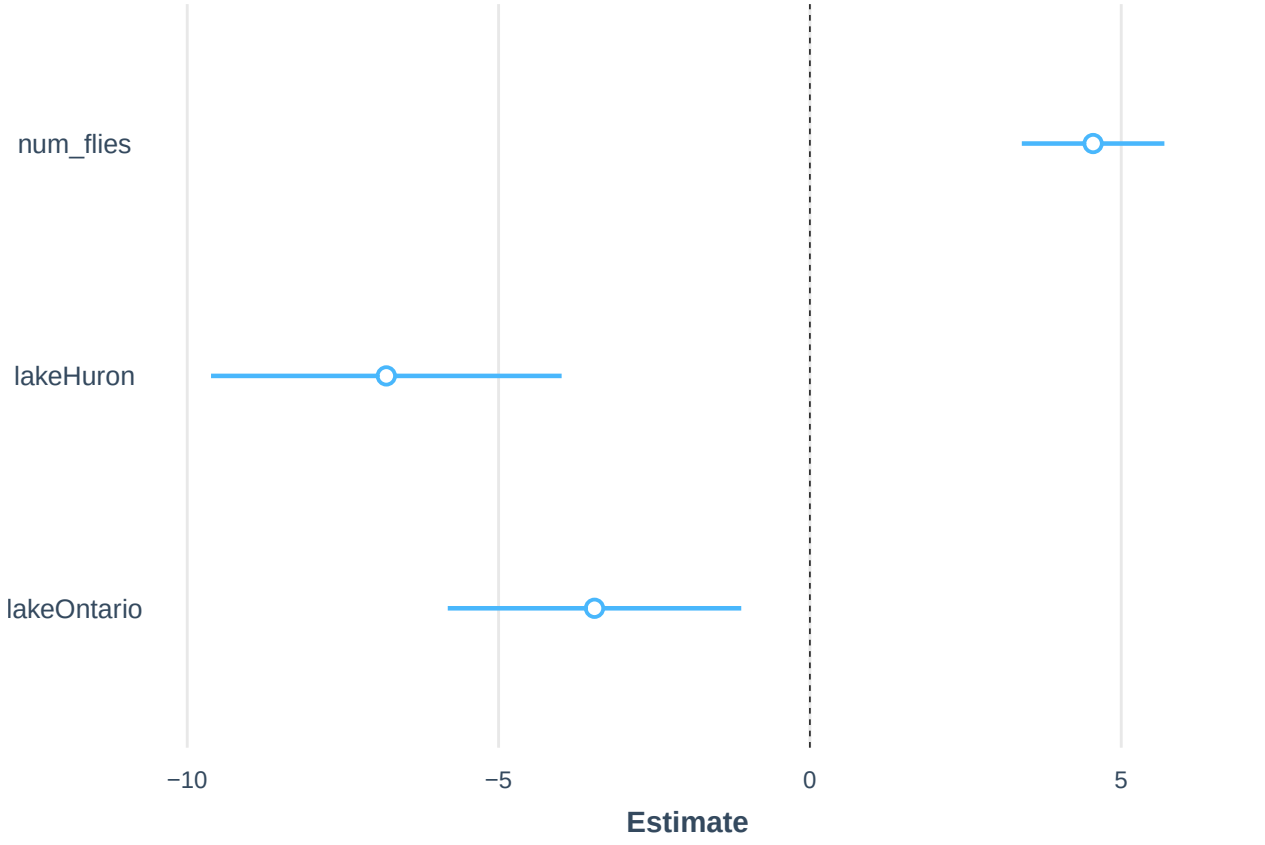
```
##
## % Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harva
## % Date and time: Fri, Sep 10, 2021 - 14:28:08
## \begin{table}[!htbp] \centering
##   \caption{}
##   \label{}
## \begin{tabular}{@{\extracolsep{5pt}}lc}
## \\[-1.8ex]\hline
## \hline \\[-1.8ex]
##  & \multicolumn{1}{c}{\textit{Dependent variable:}} \\
## \cline{2-2}
## \\[-1.8ex] & frog\_weight \\
## \hline \\[-1.8ex]
##  num\_flies & 0.424$^{***}$ \\
##   & (0.054) \\
##   & \\
##  lakeHuron & $-$6.803$^{***}$ \\
##   & (1.427) \\
##   & \\
##  lakeOntario & $-$3.459$^{***}$ \\
##   & (1.195) \\
##   & \\
##  Constant & 33.571$^{***}$ \\
##   & (3.140) \\
##   & \\
## \hline \\[-1.8ex]
## Observations & 200 \\
## R$^{2}$ & 0.434 \\
## Adjusted R$^{2}$ & 0.426 \\
```

```
## Residual Std. Error & 7.183 (df = 196) \\
## F Statistic & 50.175$^{***}$ (df = 3; 196) \\
## \hline
## \hline \\[-1.8ex]
## \textit{Note:}  & \multicolumn{1}{r}{$^{*}$p$<$0.1; $^{**}$p$<$0.05; $^{***}$p$<$0.01} \\
## \end{tabular}
## \end{table}
```

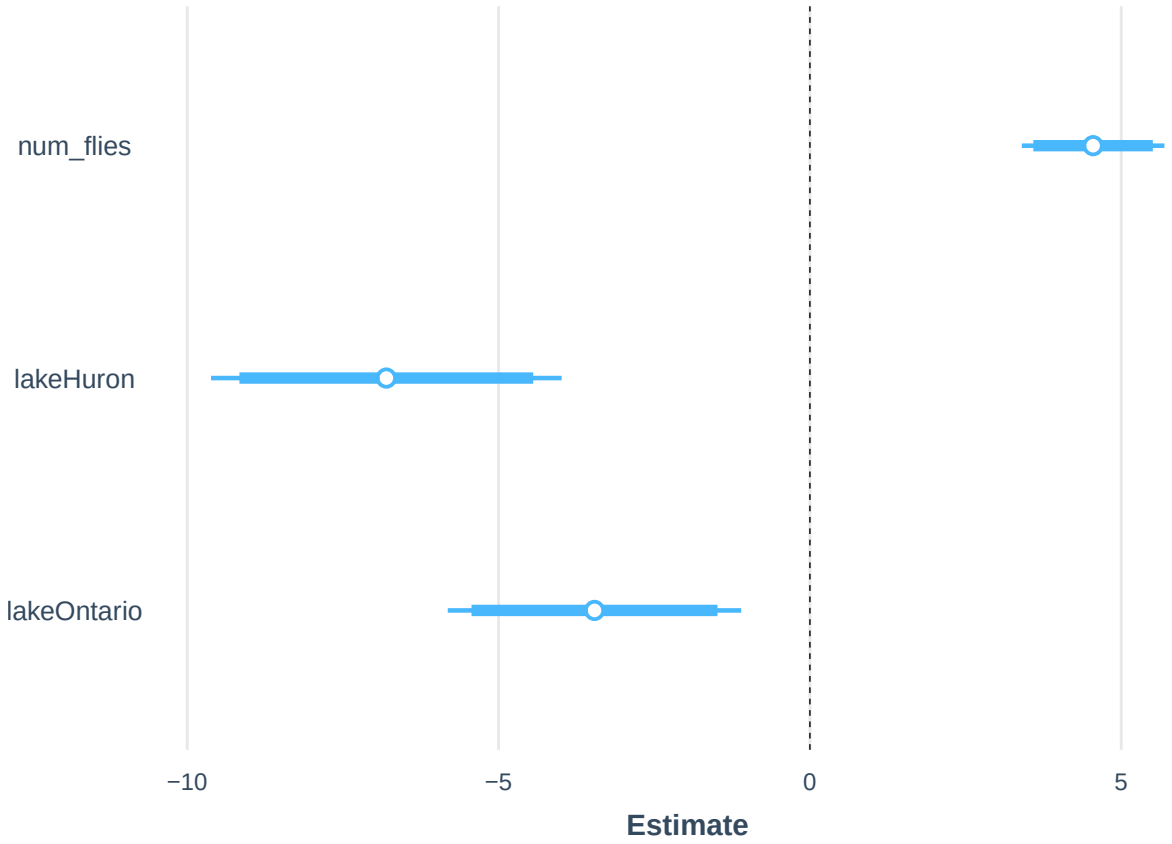
Plotting coefficients

`plot_summs` is great for coefficient plots

```
library(jtools)
plot_summs(frog_base.mod, scale = TRUE)
```

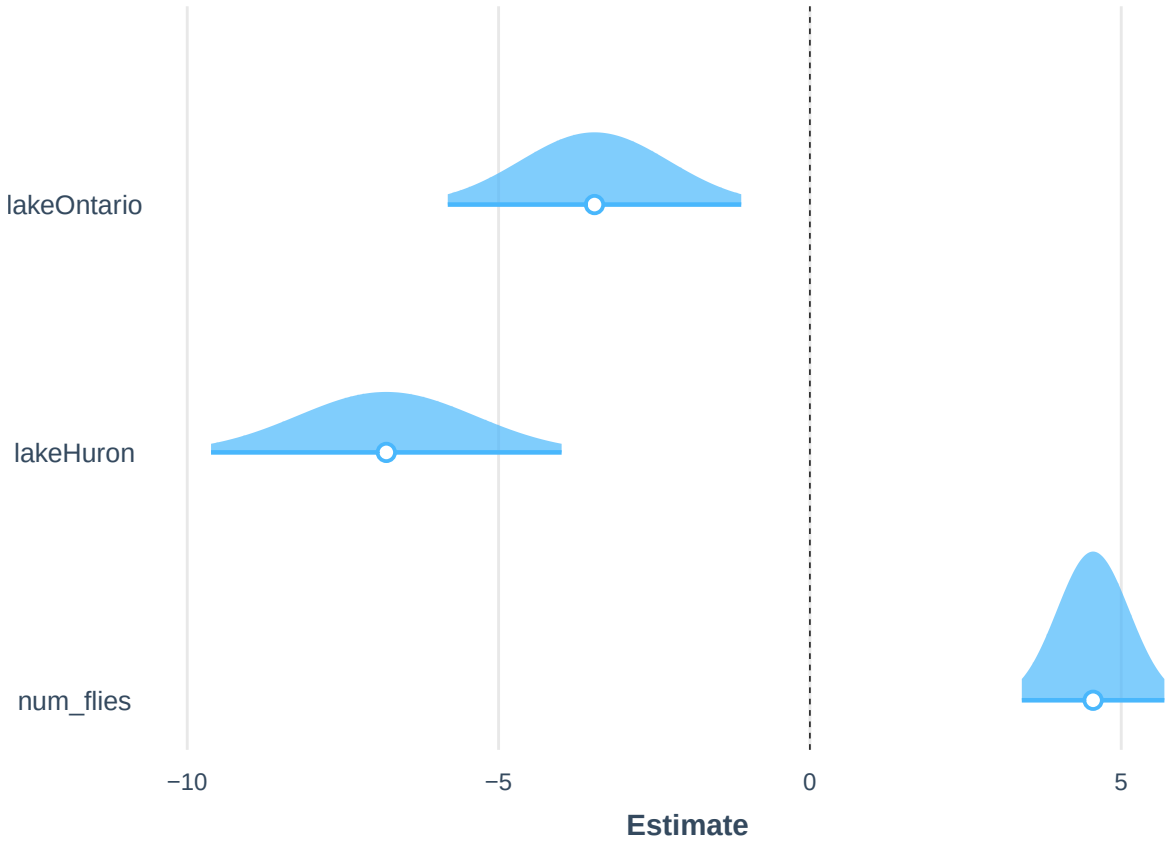



```
plot_summs(frog_base.mod, scale = TRUE, inner_ci_level = .9)
```



Plotting normal distributions can also be interesting

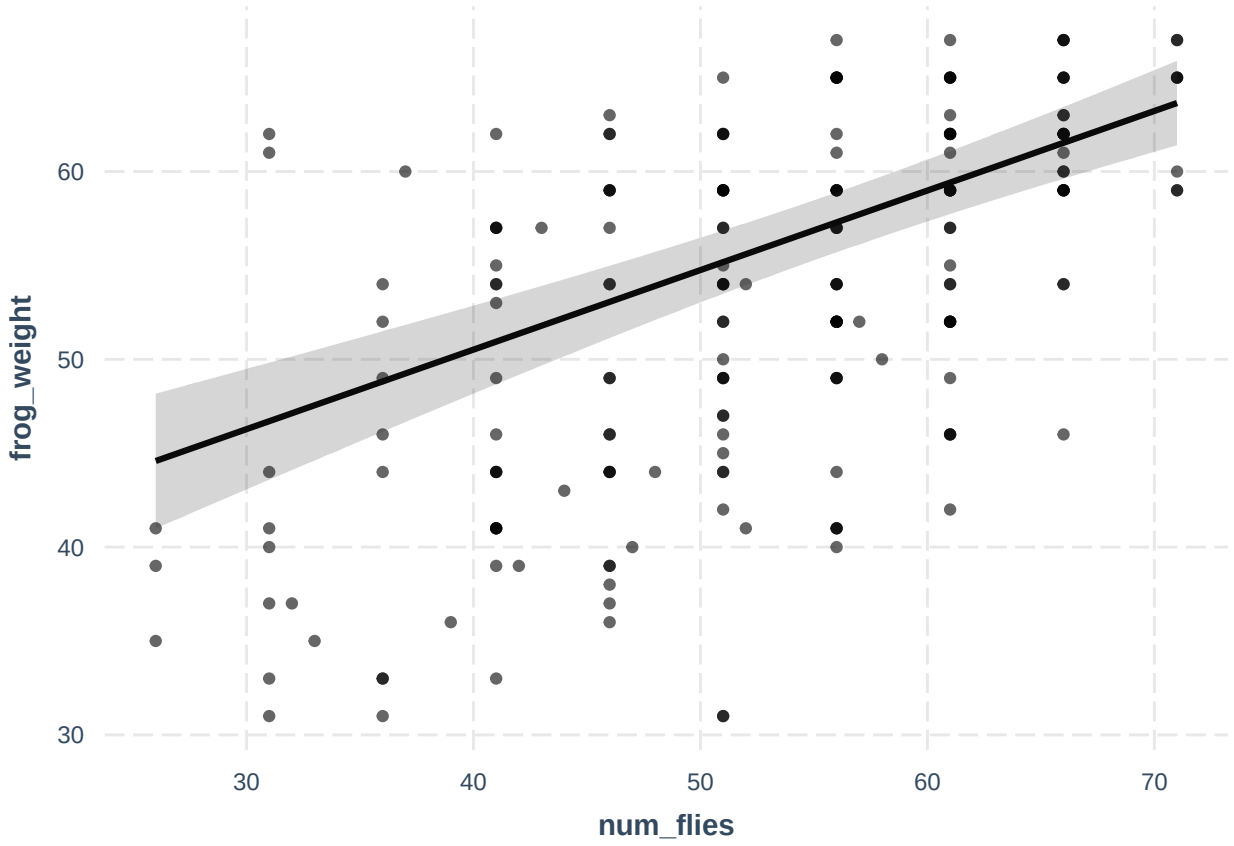
```
plot_summs(frog_base.mod,
           scale = TRUE,
           plot.distributions = TRUE) #plot normal distributions
```



Effects plots

These are useful for showing the size of the main predicted effect and can also include your points

```
library(jtools)
jtools::effect_plot(frog_base.mod,
            pred = num_flies,  #main focal predictor
            interval = TRUE, #confidence interval
            plot.points = TRUE) #plot data points
```



Interactions

Interactive models are important parts of statistical modelling in social science. However, these models are difficult to interpret by just looking at the coefficients from model output. Given this, we should try to approach results from interactions graphically!

Let's take a look at a continuous × categorical interaction with the `frog_weight.data`

Estimate interactive model

- For interactions in R all we need to do is use the `*` symbol between the two terms of interest. Using `*` will include the constituent variables from the interactions in the model as well.

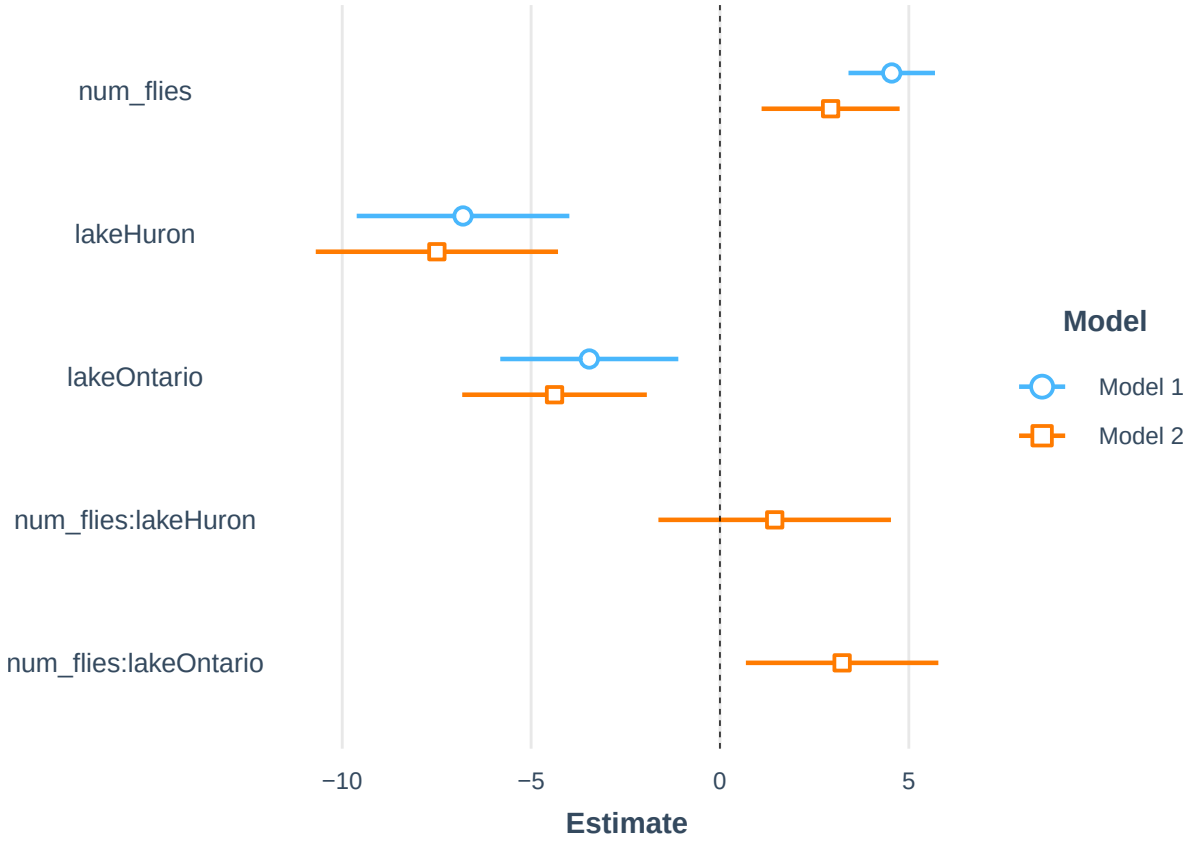
```
frog.mod <- lm(frog_weight ~ num_flies*lake, data = frog_weight.data)
summary(frog.mod)
```

```
##
## Call:
## lm(formula = frog_weight ~ num_flies * lake, data = frog_weight.data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -17.5822  -4.7273   0.3251   5.2452  14.7271
##
## Coefficients:
```

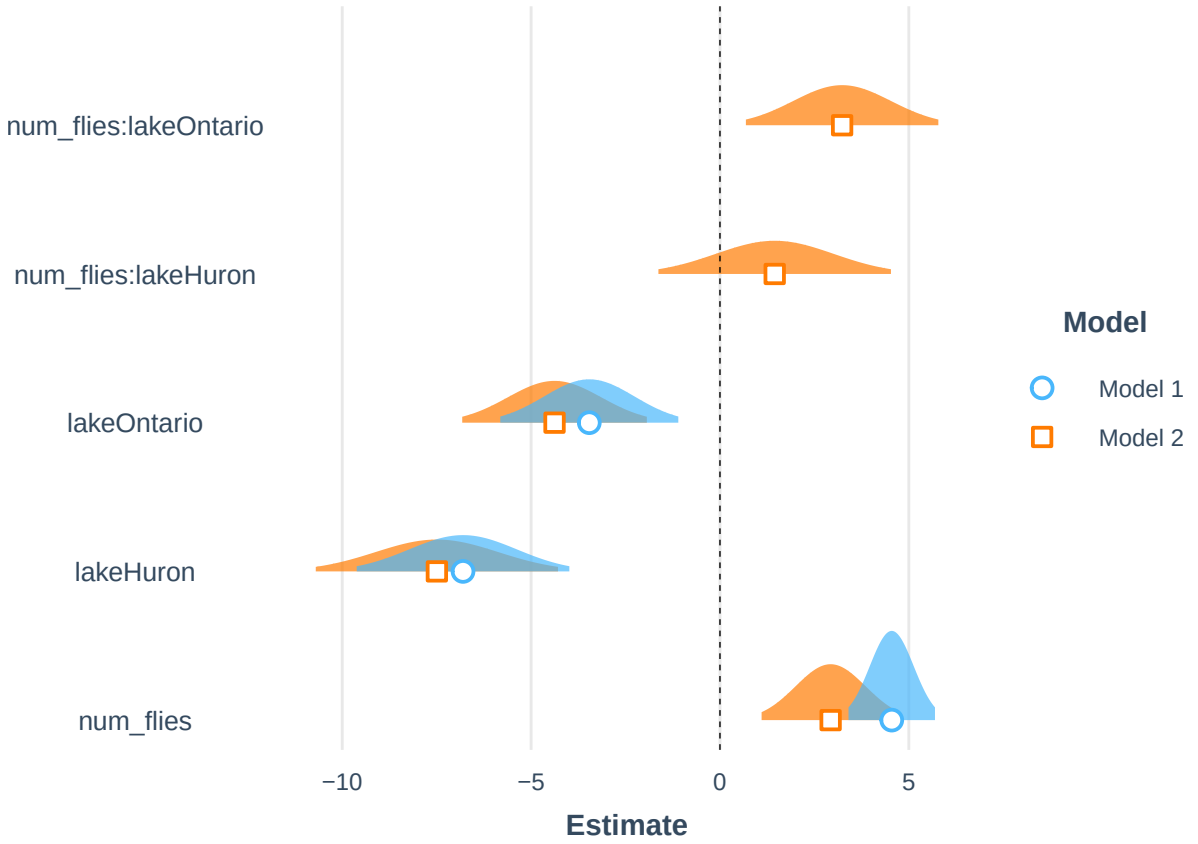
```
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              42.03102    4.90790   8.564 3.34e-15 ***
## num_flies                 0.27285    0.08629   3.162  0.00182 **
## lakeHuron               -14.56722    7.17820  -2.029  0.04379 *
## lakeOntario             -20.16644    6.77948  -2.975  0.00331 **
## num_flies:lakeHuron       0.13496    0.14539   0.928  0.35442
## num_flies:lakeOntario     0.30124    0.12033   2.504  0.01312 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.106 on 194 degrees of freedom
## Multiple R-squared:  0.4522, Adjusted R-squared:  0.438
## F-statistic: 32.02 on 5 and 194 DF,  p-value: < 2.2e-16
```

We can compare this interactive output with our output from before

```
plot_summs(frog_base.mod, frog.mod, scale = TRUE)
```

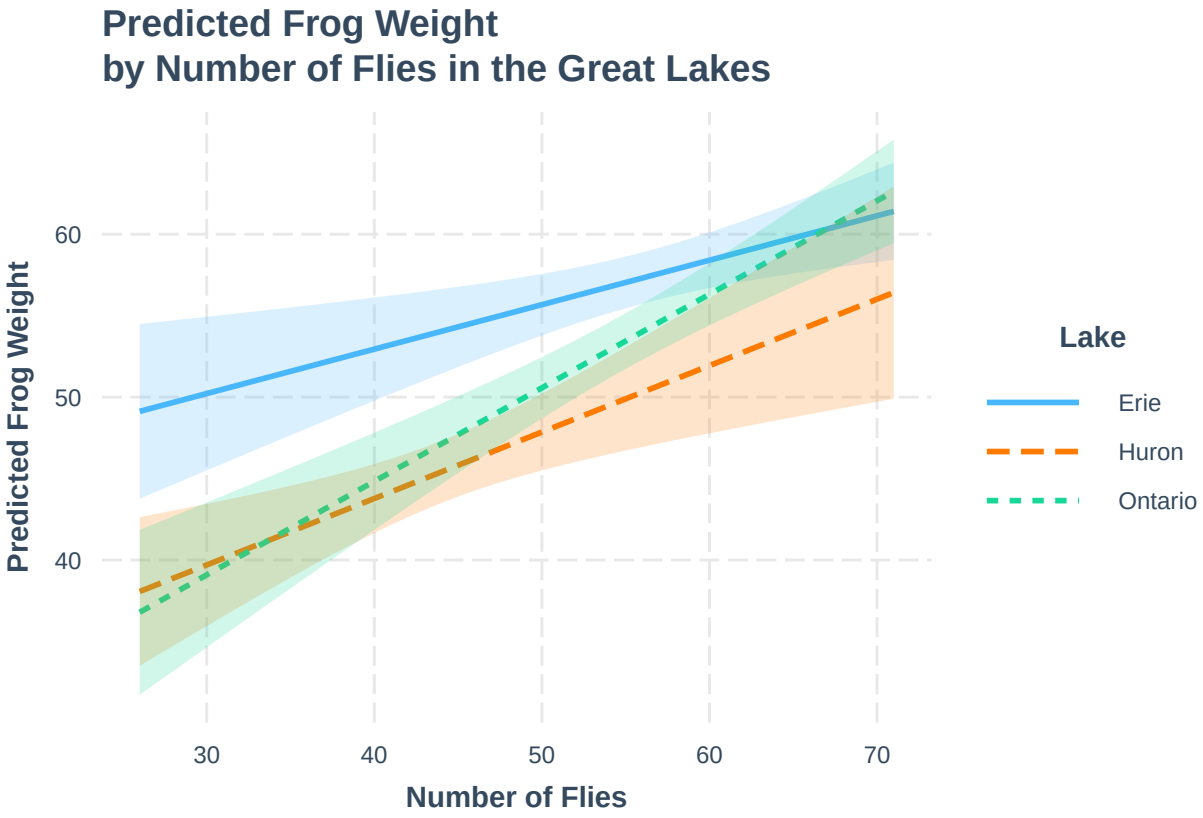



```
plot_summs(frog_base.mod, frog.mod,
           scale = TRUE,
           plot.distributions = TRUE)
```



- Now we can plot the predictions with the `interact_plot()`. More details on this command can be found here. Note that we have to set the `interval` option to `TRUE` if we want to have confidence intervals around our predictions.

```
library(jtools)
library(interactions)

interact_plot(model = frog.mod, pred = num_flies, modx = lake, #modx refers to the modifier
              interval = TRUE, #Add a confidence interval
              legend.main = "Lake", #Set the name of the legend
              x.label = "Number of Flies", #label of the x-axis
              y.label = "Predicted Frog Weight", #label of the y-axis
              main.title = "Predicted Frog Weight \nby Number of Flies in the Great Lakes")
```



Blood Pressure Analysis

Now let's move to `nhanes2`. With this, we can see more complicated interactions

```
nhanes2 <- import("nhanes2.dta")
```

Examine Data

```
str(nhanes2)
```

```
## 'data.frame':    10351 obs. of  58 variables:
##  $ sampl   : num  1400 1401 1402 1404 1405 ...
##   ..- attr(*, "label")= chr "unique case identifier"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ strata  : num  1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "stratum identifier, 1-32"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ psu     : num  1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "primary sampling unit, 1 or 2"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ region  : num  3 3 3 3 3 3 3 3 3 3 ...
##   ..- attr(*, "label")= chr "1=NE, 2=MW, 3=S, 4=W"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##   ..- attr(*, "labels")= Named num [1:4] 1 2 3 4
##   .. ..- attr(*, "names")= chr [1:4] "NE" "MW" "S" "W"
##  $ smsa    : num  2 2 1 2 1 1 1 2 2 2 ...
```

```
##   ..- attr(*, "label")= chr "1=SMSAcity,2=SMSA~city,4=~SMSA"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ location: num  1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "stand number, 1-64"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ houssiz : num  4 6 6 9 3 1 2 1 3 4 ...
##   ..- attr(*, "label")= chr "# persons in household, 1-14"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ sex     : num  1 2 2 2 2 2 2 2 2 1 ...
##   ..- attr(*, "label")= chr "1=male, 2=female"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##   ..- attr(*, "labels")= Named num [1:2] 1 2
##   .. ..- attr(*, "names")= chr [1:2] "Male" "Female"
##  $ race    : num  1 1 3 1 1 1 1 1 1 2 ...
##   ..- attr(*, "label")= chr "1=white, 2=black, 3=other"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##   ..- attr(*, "labels")= Named num [1:3] 1 2 3
##   .. ..- attr(*, "names")= chr [1:3] "White" "Black" "Other"
##  $ age     : num  54 41 21 63 64 63 67 57 68 68 ...
##   ..- attr(*, "label")= chr "age in years"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ height  : num  175 152 164 163 163 ...
##   ..- attr(*, "label")= chr "height (cm)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ weight  : num  62.5 48.8 67.2 94.5 74.3 ...
##   ..- attr(*, "label")= chr "weight (kg)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ bpsystol: num  106 108 98 180 120 180 100 154 120 104 ...
##   ..- attr(*, "label")= chr "systolic blood pressure"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ bpdiast : num  80 66 66 80 76 90 74 90 76 66 ...
##   ..- attr(*, "label")= chr "diastolic blood pressure"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ tcresult: num  226 179 137 189 311 187 187 206 244 255 ...
##   ..- attr(*, "label")= chr "serum cholesterol (mg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ tgresult: num  NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "label")= chr "serum triglycerides (mg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ hdresult: num  NA NA NA NA NA NA NA NA NA NA ...
##   ..- attr(*, "label")= chr "high density lipids (mg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ hgb     : num  13.9 10.9 13.5 13.4 15.6 ...
##   ..- attr(*, "label")= chr "hemoglobin (g/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ hct     : num  42.7 33 40.5 38.7 45.5 ...
##   ..- attr(*, "label")= chr "hematocrit (%)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ tibc    : num  377 363 353 365 320 330 312 451 375 204 ...
##   ..- attr(*, "label")= chr "total iron bind. cap. (mcg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ iron    : num  86 24 129 36 91 64 90 61 74 48 ...
##   ..- attr(*, "label")= chr "serum iron (mcg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
```

```
##  $ hlthstat: num  2 2 3 4 2 5 2 1 2 5 ...
##   ..- attr(*, "label")= chr "1=excellent,..., 5=poor"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ heartatk: num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "heart attack, 1=yes, 0=no"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ diabetes: num  0 0 0 1 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "diabetes, 1=yes, 0=no"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ sizplace: num  2 2 2 2 2 2 2 2 2 2 ...
##   ..- attr(*, "label")= chr "1=urban,..., 8=rural"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ finalwgt: num  8995 25964 8752 4310 9011 ...
##   ..- attr(*, "label")= chr "sampling weight (except lead)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ leadwt  : num  0 53543 0 0 19286 ...
##   ..- attr(*, "label")= chr "sampling weight for lead"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ corpuscl: num  89 92.4 106.3 83.4 95 ...
##   ..- attr(*, "label")= chr "mean corpuscular volume (fL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ trnsfern: num  22.8 6.6 36.5 9.9 28.4 ...
##   ..- attr(*, "label")= chr "transferrin saturation (%)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ albumin : num  4.7 4.6 4.6 4.6 4.6 ...
##   ..- attr(*, "label")= chr "serum albumin (g/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ vitaminc: num  0.4 1.2 1.3 1.1 1.3 ...
##   ..- attr(*, "label")= chr "serum vitamin C (mg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ zinc    : num  104 111 102 109 99 101 93 83 98 98 ...
##   ..- attr(*, "label")= chr "serum zinc (mcg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ copper  : num  156 139 238 132 127 164 159 130 145 175 ...
##   ..- attr(*, "label")= chr "serum copper (mcg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ porphyrn: num  38 92 48 62 44 41 43 49 52 45 ...
##   ..- attr(*, "label")= chr "erythrocyte porphyrin (mcg/dl)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ lead    : num  NA 13 NA NA 20 NA 13 NA NA 17 ...
##   ..- attr(*, "label")= chr "lead (mcg/dL)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ female  : num  0 1 1 1 1 1 1 1 1 0 ...
##   ..- attr(*, "label")= chr "1=female, 0=male"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ black   : num  0 0 0 0 0 0 0 0 0 1 ...
##   ..- attr(*, "label")= chr "1 if race=black, 0 otherwise"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ orace   : num  0 0 1 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "1 if race=other, 0 otherwise"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ fhtatk  : num  NA 0 0 0 0 0 0 0 0 NA ...
##   ..- attr(*, "label")= chr "female heart attack, 1=yes,2=no"
##   ..- attr(*, "format.stata")= chr "%8.0g"
```

```
##  $ hsizgp  : num  4 5 5 5 3 1 2 1 3 4 ...
##   ..- attr(*, "label")= chr "# in household or 5 if #>=5"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ hsiz1   : num  0 0 0 0 0 1 0 1 0 0 ...
##   ..- attr(*, "label")= chr "hsizgp==     1.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ hsiz2   : num  0 0 0 0 0 0 1 0 0 0 ...
##   ..- attr(*, "label")= chr "hsizgp==     2.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ hsiz3   : num  0 0 0 0 1 0 0 0 1 0 ...
##   ..- attr(*, "label")= chr "hsizgp==     3.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ hsiz4   : num  1 0 0 0 0 0 0 0 0 1 ...
##   ..- attr(*, "label")= chr "hsizgp==     4.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ hsiz5   : num  0 1 1 1 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "hsizgp==     5.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ region1 : num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "region==NE"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ region2 : num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "region==MW"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ region3 : num  1 1 1 1 1 1 1 1 1 1 ...
##   ..- attr(*, "label")= chr "region==S"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ region4 : num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "region==W"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ smsa1   : num  0 0 1 0 1 1 1 0 0 0 ...
##   ..- attr(*, "label")= chr "smsa==     1.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ smsa2   : num  1 1 0 1 0 0 0 1 1 1 ...
##   ..- attr(*, "label")= chr "smsa==     2.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ smsa3   : num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "smsa==     4.0000"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ rural   : num  0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "label")= chr "1=rural, 0=urban"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  $ loglead : num  NA 2.56 NA NA 3 ...
##   ..- attr(*, "label")= chr "log(lead)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ agegrp  : num  4 3 1 5 5 5 5 4 5 5 ...
##   ..- attr(*, "label")= chr "Age Group"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##   ..- attr(*, "labels")= Named num [1:6] 1 2 3 4 5 6
##   .. ..- attr(*, "names")= chr [1:6] "20-29" "30-39" "40-49" "50-59" ...
##  $ highlead: num  NA 0 NA NA 0 NA 0 NA NA 0 ...
##   ..- attr(*, "label")= chr "1 if lead >= 25, 0 otherwise"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##   ..- attr(*, "labels")= Named num [1:2] 0 1
```

```
##   .. ..- attr(*, "names")= chr [1:2] "lead<25" "lead>=25"
##  $ bmi     : num  20.5 21 25 35.7 27.9 ...
##   ..- attr(*, "label")= chr "Body Mass Index (BMI)"
##   ..- attr(*, "format.stata")= chr "%9.0g"
##  $ highbp  : num  0 0 0 1 0 1 0 1 0 0 ...
##   ..- attr(*, "label")= chr "1 if bpsystol >= 140|bpdiast >= 90, 0 otherwise"
##   ..- attr(*, "format.stata")= chr "%8.0g"
##  - attr(*, "label")= chr "Second National Health and Nutrition Examination Survey"
```

```
#note that we need to make race and region factors
nhanes2$race <- as.factor(nhanes2$race)
nhanes2$region <- as.factor(nhanes2$region)

#We would have to recode these as well but let's skip that for now

#We will recode the female variable

nhanes2$female[nhanes2$female== 0] <- "Male"
nhanes2$female[nhanes2$female== 1] <- "Female"
```

```
table1::table1(~bpsystol + race + weight + age + region + iron| female, data = nhanes2)
```

	Female	Male	Overall
	(N=5436)	(N=4915)	(N=10351)
systolic blood pressure			
Mean (SD)	129 (25.1)	133 (21.0)	131 (23.3)
Median [Min, Max]	124 [65.0, 300]	130 [70.0, 254]	128 [65.0, 300]
race			
1	4753 (87.4%)	4312 (87.7%)	9065 (87.6%)
2	586 (10.8%)	500 (10.2%)	1086 (10.5%)
3	97 (1.8%)	103 (2.1%)	200 (1.9%)
weight (kg)			
Mean (SD)	66.4 (14.7)	78.0 (13.6)	71.9 (15.4)
Median [Min, Max]	63.6 [34.9, 159]	76.9 [30.8, 176]	70.4 [30.8, 176]
age in years			
Mean (SD)	47.7 (17.3)	47.4 (17.2)	47.6 (17.2)
Median [Min, Max]	49.5 [20.0, 74.0]	49.0 [20.0, 74.0]	49.0 [20.0, 74.0]
region			
1	1078 (19.8%)	1018 (20.7%)	2096 (20.2%)
2	1464 (26.9%)	1310 (26.7%)	2774 (26.8%)
3	1521 (28.0%)	1332 (27.1%)	2853 (27.6%)
4	1373 (25.3%)	1255 (25.5%)	2628 (25.4%)
serum iron (mcg/dL)			
Mean (SD)	95.6 (33.9)	104 (33.8)	99.4 (34.1)
Median [Min, Max]	91.0 [16.0, 321]	100 [18.0, 294]	95.0 [16.0, 321]

Estimate regression

- Let's start with an interaction between `bmi` and `female`

```
blood1.reg <- lm(bpsystol ~ age + race + region + iron + bmi*female, data =nhanes2)
summary(blood1.reg)
```

```
##
## Call:
## lm(formula = bpsystol ~ age + race + region + iron + bmi * female,
##     data = nhanes2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.348 -12.737  -2.003  10.174 139.347
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    67.308429   1.508435  44.621  < 2e-16 ***
## age             0.590672   0.011303  52.257  < 2e-16 ***
## race2           2.711238   0.637696   4.252 2.14e-05 ***
## race3           2.164707   1.401985   1.544 0.122612
## region2        -0.176177   0.560818  -0.314 0.753419
## region3        -0.872894   0.563594  -1.549 0.121461
## region4        -0.690853   0.571025  -1.210 0.226365
## iron           -0.005517   0.005706  -0.967 0.333665
## bmi             1.339263   0.047925  27.945  < 2e-16 ***
## femaleMale      8.120351   2.160883   3.758 0.000172 ***
## bmi:femaleMale -0.157072   0.083354  -1.884 0.059540 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.33 on 10340 degrees of freedom
## Multiple R-squared:  0.3142, Adjusted R-squared:  0.3135
## F-statistic: 473.7 on 10 and 10340 DF,  p-value: < 2.2e-16
```

If using Rmarkdown, we can use `tidy()` and `kable` to clean this output up

```
kable(tidy(blood1.reg))
```

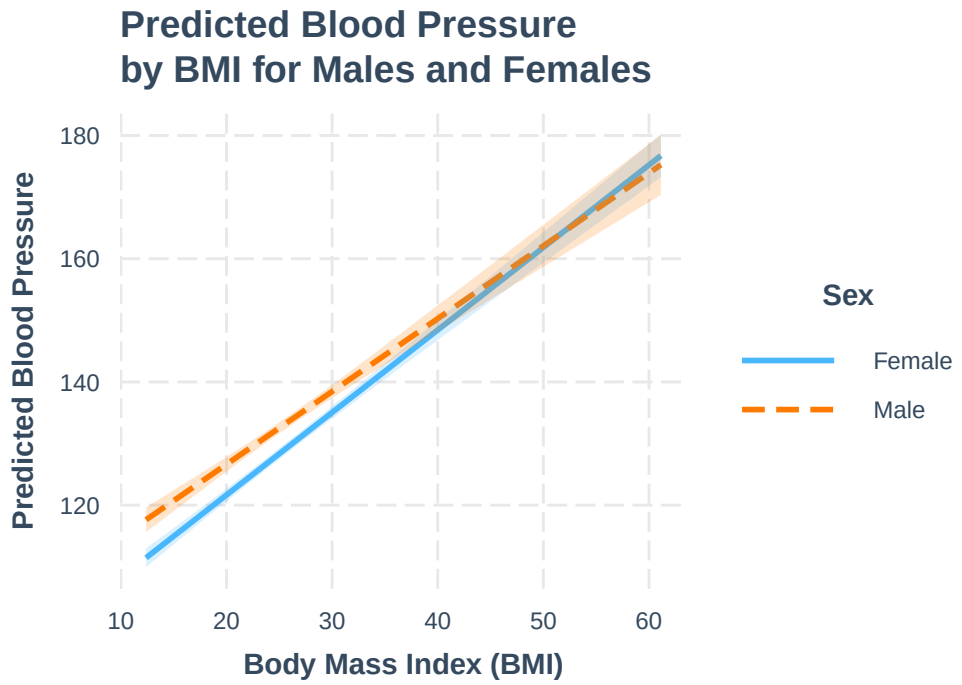
term	estimate	std.error	statistic	p.value
(Intercept)	67.3084288	1.5084354	44.6213537	0.0000000
age	0.5906725	0.0113031	52.2574140	0.0000000
race2	2.7112382	0.6376957	4.2516178	0.0000214
race3	2.1647071	1.4019853	1.5440299	0.1226117
region2	-0.1761769	0.5608183	-0.3141426	0.7534191
region3	-0.8728937	0.5635940	-1.5487986	0.1214607
region4	-0.6908532	0.5710254	-1.2098467	0.2263654
iron	-0.0055166	0.0057061	-0.9668025	0.3336654
bmi	1.3392633	0.0479249	27.9450536	0.0000000
femaleMale	8.1203512	2.1608833	3.7578851	0.0001723
bmi:femaleMale	-0.1570724	0.0833543	-1.8843942	0.0595397

Create more plots!

- Let's create a predicted values plot. Instead of showing the slopes of the two

```
interact_plot(model = blood1.reg,
              pred = bmi,
              modx = female, #modx refers to the modifier
              interval = TRUE, # confidence interval
              legend.main = "Sex",
```

```
        x.label = "Body Mass Index (BMI)",
        y.label = "Predicted Blood Pressure", #this is the dependent variable
        main.title = "Predicted Blood Pressure \nby BMI for Males and Females")
```



Estimate a continuous×continous interaction

- Now we will try a continuous×continous interaction. These are even more complicated to interpret. We will interact `bmi` with `age`

```
blood2.reg <- lm(bpsystol ~ female + race + region + iron + bmi*age, data =nhanes2)
summary(blood2.reg)
```

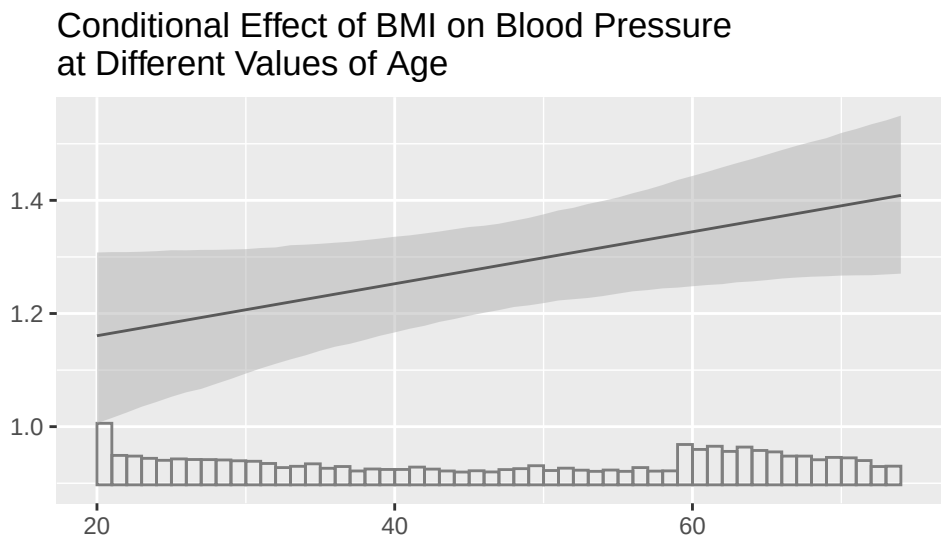
```
##
## Call:
## lm(formula = bpsystol ~ female + race + region + iron + bmi *
##     age, data = nhanes2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -65.291 -12.720  -2.014  10.216 139.103
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 73.983384   3.092982  23.920  < 2e-16 ***
## femaleMale   4.179151   0.384768  10.861  < 2e-16 ***
## race2        2.805585   0.636007   4.411 1.04e-05 ***
## race3        2.280280   1.401256   1.627   0.1037
## region2     -0.155151   0.560813  -0.277   0.7821
## region3     -0.831454   0.563696  -1.475   0.1402
## region4     -0.666471   0.571070  -1.167   0.2432
## iron        -0.006203   0.005711  -1.086   0.2774
```

```
## bmi          1.069955   0.119201   8.976  < 2e-16 ***
## age          0.477572   0.059635   8.008 1.29e-15 ***
## bmi:age      0.004543   0.002338   1.943   0.0521 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.33 on 10340 degrees of freedom
## Multiple R-squared:  0.3142, Adjusted R-squared:  0.3135
## F-statistic: 473.7 on 10 and 10340 DF,  p-value: < 2.2e-16
```

Other interaction plots!

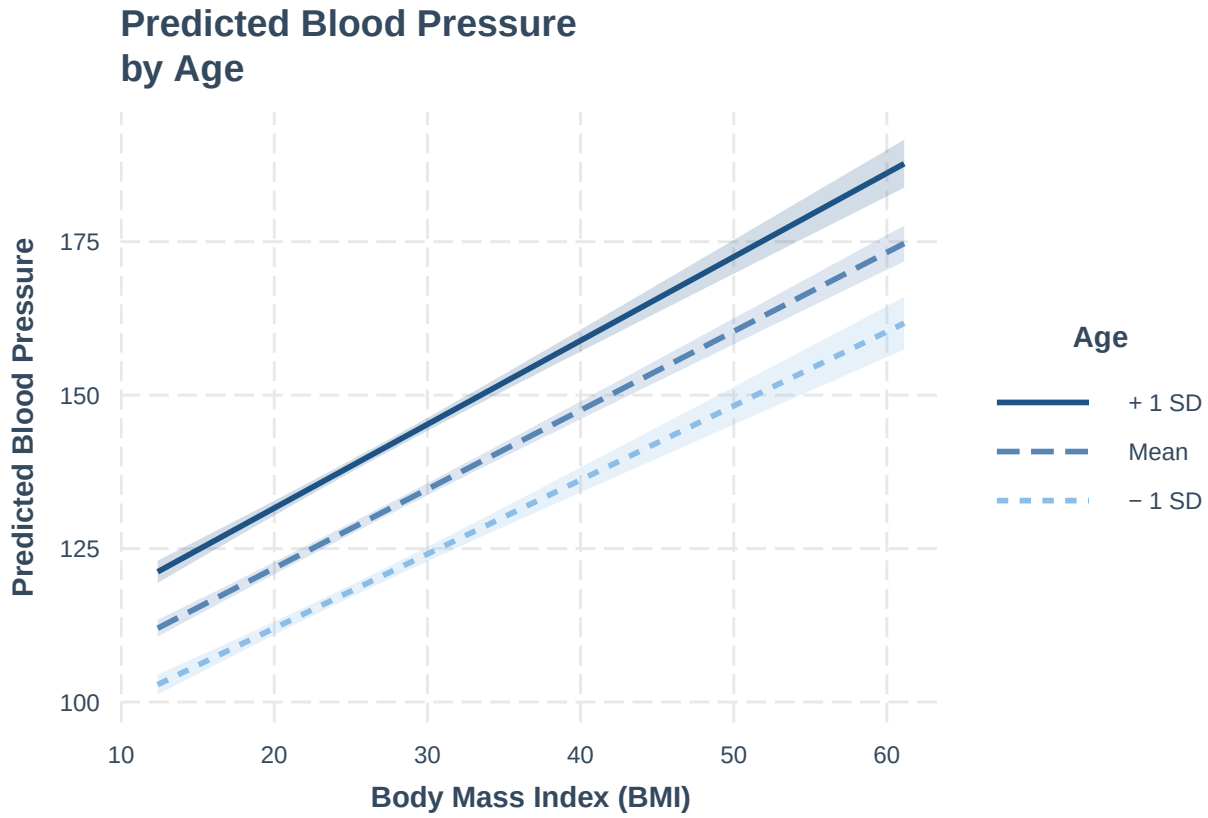
- The`interplot()` command can be especially useful for continuous interactions
- Note that we set the `hist` option to `TRUE` to include a histogram of the values of the age data.

```
library(interplot)
library(ggplot2)

interplot(m = blood2.reg, var1 = "bmi", var2 = "age", hist = TRUE) +
  ggtitle("Conditional Effect of BMI on Blood Pressure \nat Different Values of Age")
```



- Finally let's plot predictions for the interaction

```
interact_plot(model = blood2.reg,
              pred = bmi, #main predictor on the x-axis
              modx = age, #modx refers to the modifier
              interval = TRUE, #Include confidence intervals around predictions
              legend.main = "Age", #Title of the legend
              x.label = "Body Mass Index (BMI)", #x-axis label
              y.label = "Predicted Blood Pressure", #y-axis label
              main.title = "Predicted Blood Pressure \nby Age")#use the \n to break to next line
```



- As we can see, `interact_plot` will by default plot lines for the *mean* and ± *1 standard deviation either side of the mean* of the moderator, which in this case is age.

Exporting output to latex

Advanced plotting

R can do all sorts of plots and visualization that goes beyond basic regression. Here we'll look at census data and mapping.

Working with `tidycensus`

Download census data

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

-Note that you need to submit to get a census API key

- After getting the key you can call it using the below function

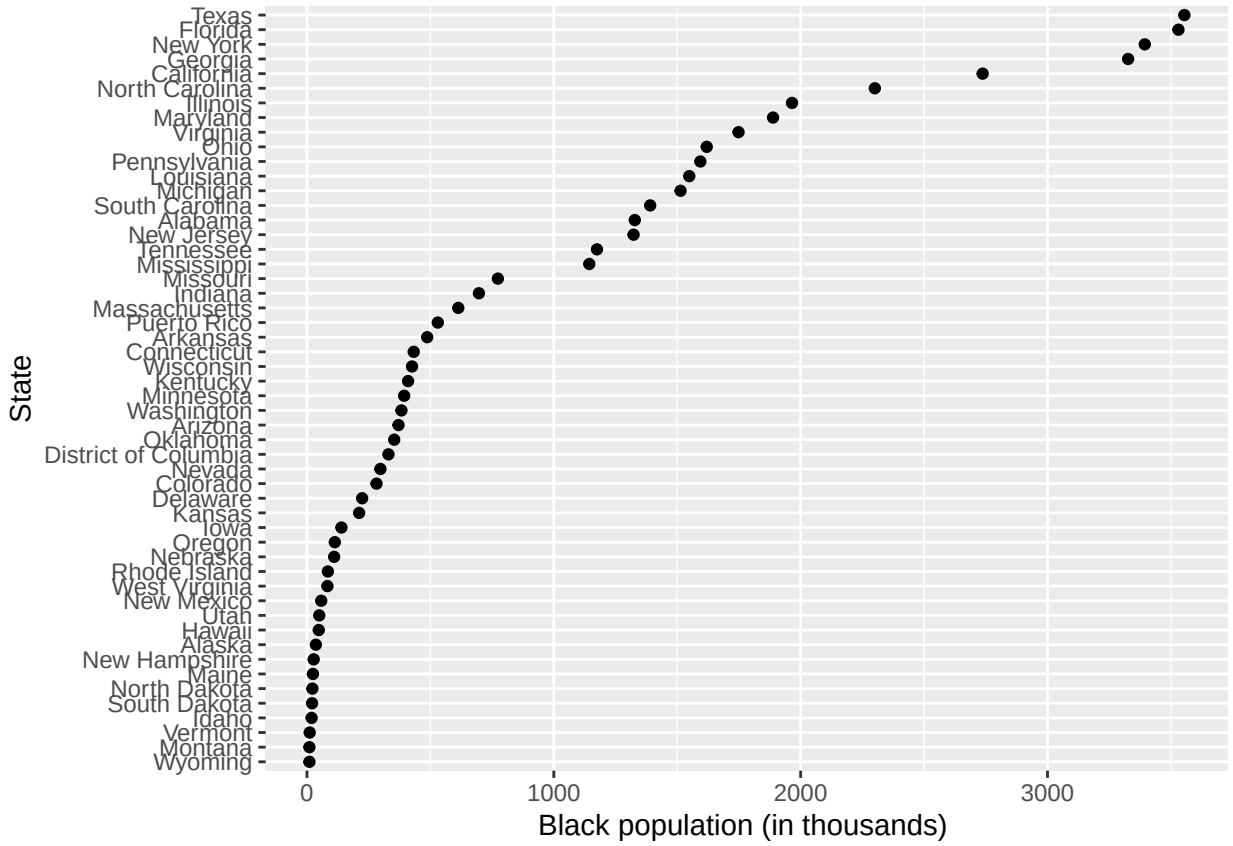
```
census_api_key("bbf9b57e8eabfb053b114216983d8348d4e24540", overwrite = FALSE, install = FALSE)


Sys.getenv("CENSUS_API_KEY")
```

```
## [1] "bbf9b57e8eabfb053b114216983d8348d4e24540"
```

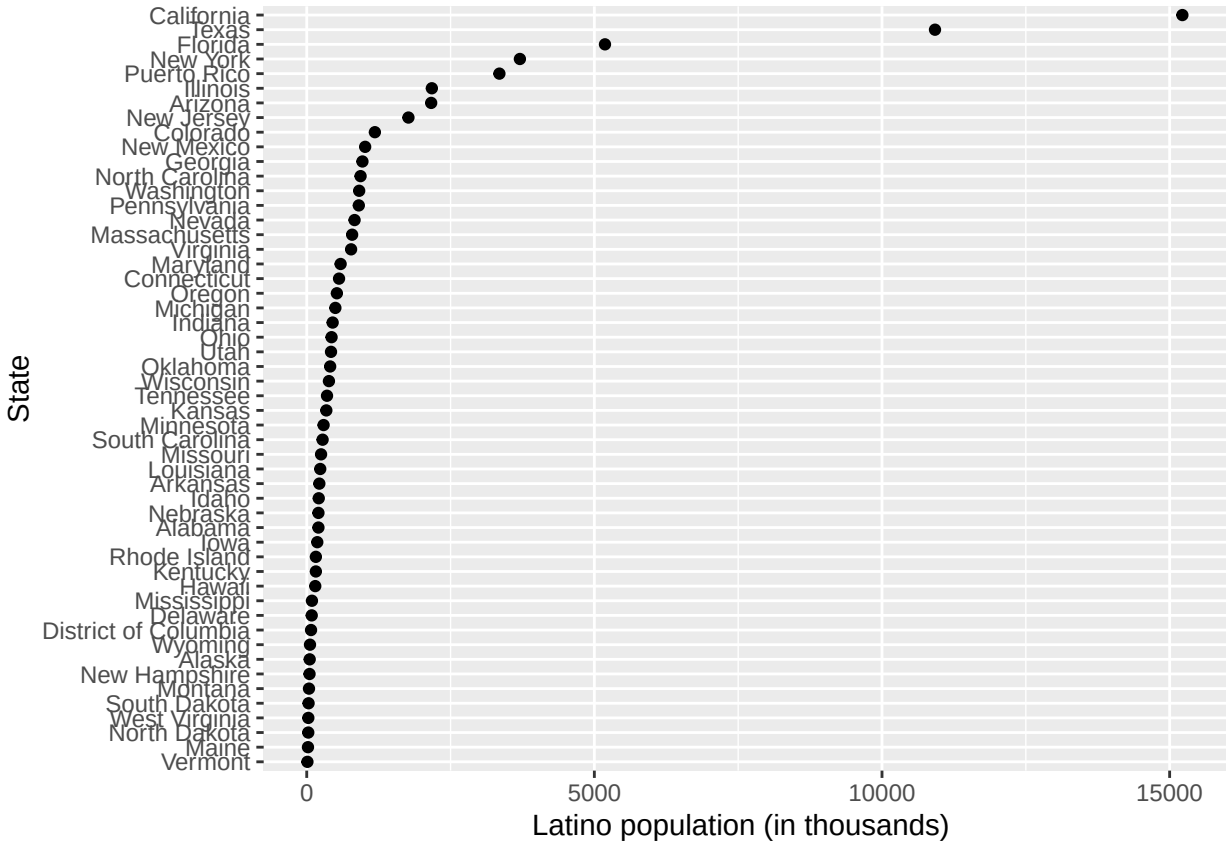
```
v17 <- load_variables(2017, "acs5", cache = TRUE)
```

```
country_2017 <- get_acs(geography = "state",
              variables = c(medincome = "B19013_001", #selecting each code and renaming it
                            Population = "B01003_001",
                            White_pop = "B02008_001",
                            Black_pop = "B02009_001",
                            Latino_pop = "B03001_003",
                            Asian_pop = "B02015_001",
                            Native_pop = "B02014_001"
                           ),
              year = 2017)

# Can also use get_decennial to get census data



#Can also get the data for 2018 and any other year

country_2018 <- get_acs(geography = "state",
              variables = c(medincome = "B19013_001",
                            Population = "B01003_001",
                            White_pop = "B02008_001",
                            Black_pop = "B02009_001",
                            Latino_pop = "B03001_003",
                            Asian_pop = "B02015_001",
                            Native_pop = "B02014_001"
                           ),
              year = 2018)
```

```
country_2017 %>%
  filter(variable == "Black_pop")%>%
  mutate(estimate = estimate/1000)%>%
  ggplot(aes(x = estimate, y = reorder(NAME, estimate))) +  #aes = aesthetics
  xlab("Black population (in thousands) ") +
  ylab("State") +

  geom_point()
```

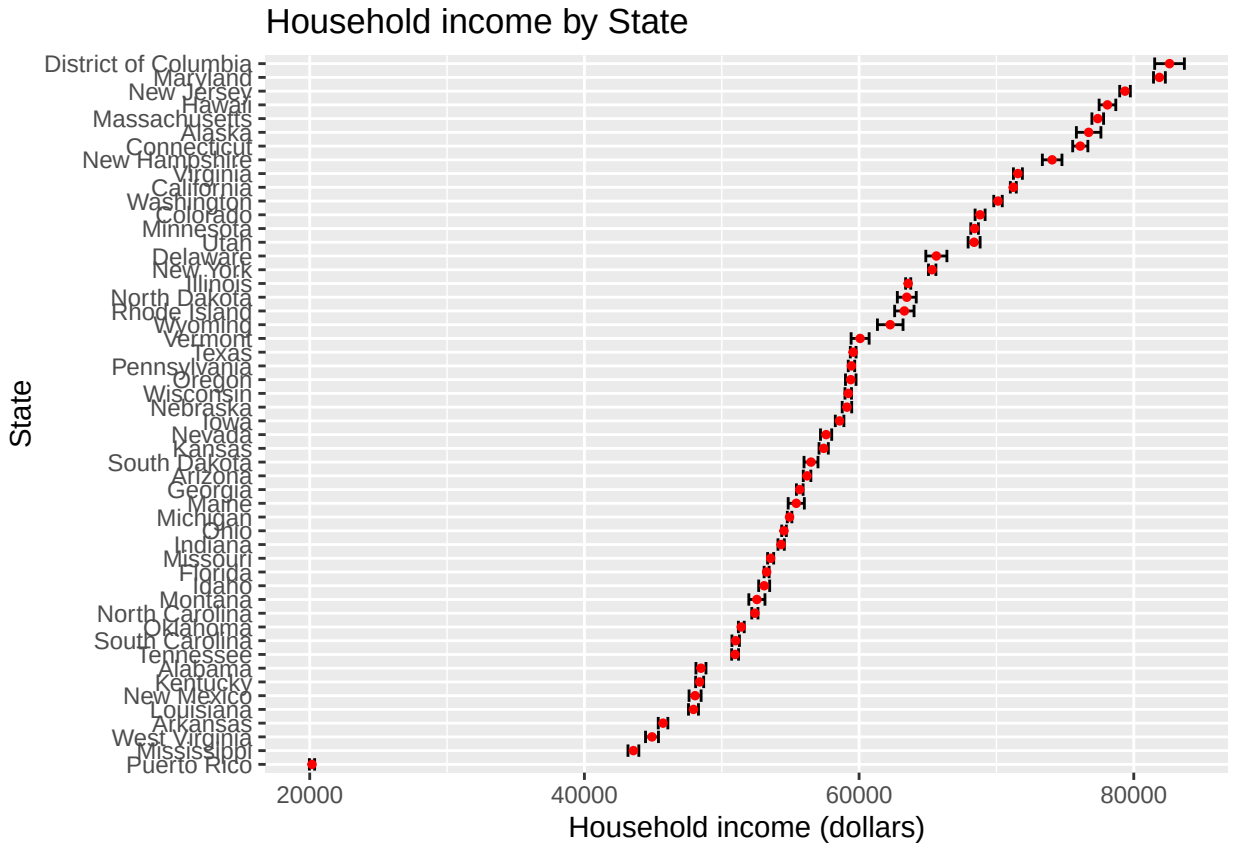



```
country_2018 %>%
  filter(variable == "Latino_pop")%>%
  mutate(estimate = estimate/1000)%>%
  ggplot(aes(x = estimate, y = reorder(NAME, estimate)))  +
  xlab("Latino population (in thousands) ") +
  ylab("State") +

  geom_point()
```



Add margin of error

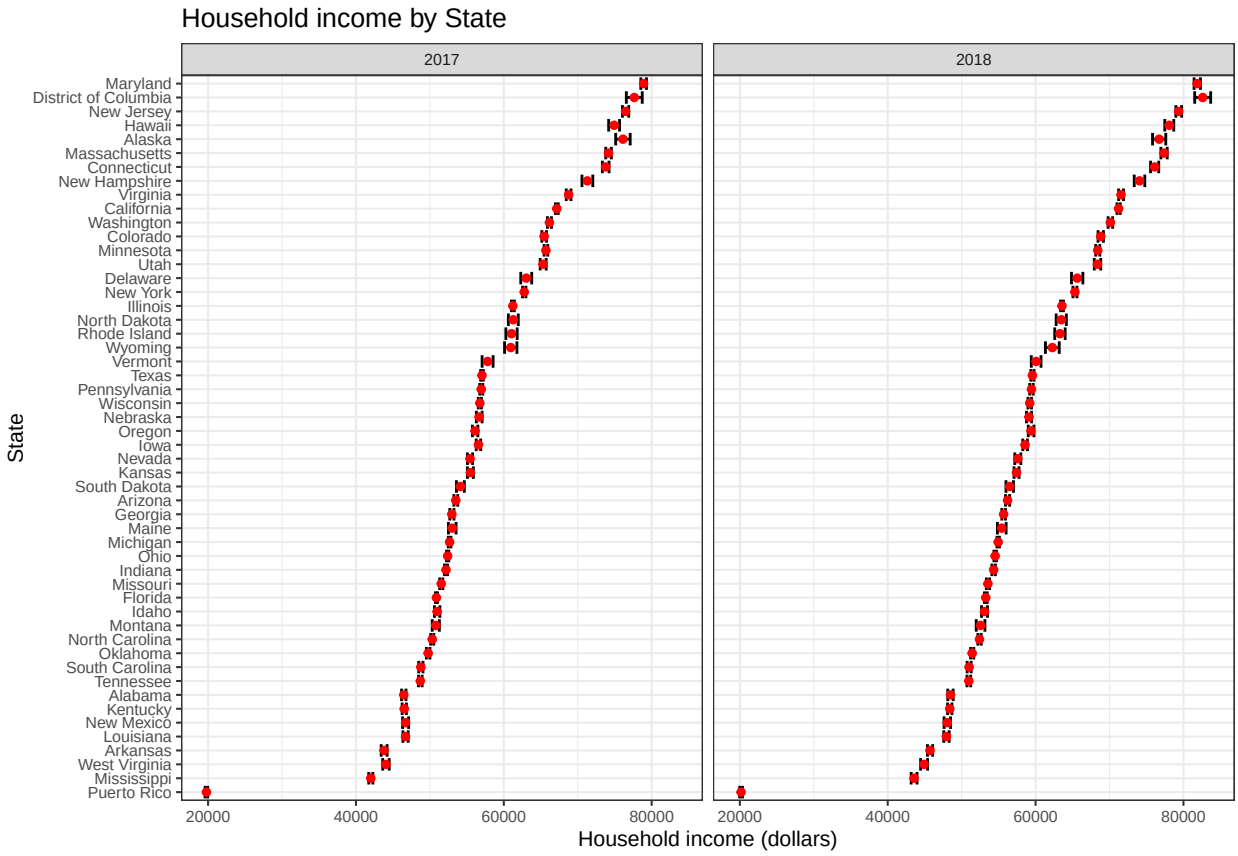
```
country_2018 %>%
  filter(variable == "medincome")%>%
  ggplot(aes(x = estimate, y = reorder(NAME, estimate))) +
  geom_errorbarh(aes(xmin = estimate - moe, xmax = estimate + moe)) +
  geom_point(color = "red", size = 1) +
  labs(title = "Household income by State",
       y = "State",
       x = "Household income (dollars)")
```



Maps with ACS and Census data

```
country_2017_year <- country_2017%>%
  mutate(Year = "2017")


country_2018_year <- country_2018%>%
  mutate(Year = "2018")

country.df <- rbind(country_2017_year,
      country_2018_year)
```

```
country.df %>%
  filter(variable == "medincome")%>%
  ggplot(aes(x = estimate, y = reorder(NAME, estimate))) +
  geom_errorbarh(aes(xmin = estimate - moe, xmax = estimate + moe)) +
      theme_bw(8) +
  geom_point(color = "red", size = 1) +
  labs(title = "Household income by State",
       y = "State",
       x = "Household income (dollars)") +
  facet_grid(.~Year)
```



Create proportions

```
library(reshape2)
country.wide <- recast(country.df, NAME + Year ~ variable, measure.var=c("estimate"))
```

```
country.prop <- country.wide%>%
group_by(NAME, Year)%>%
mutate(Black_prop = Black_pop/Population,
       Latino_prop = Latino_pop/Population,
       Asian_prop = Asian_pop/Population,
       Native_prop = Native_pop/Population,
       White_prop = White_pop/Population

       )%>%
dplyr::select(State = NAME,
              Year,
              Black = Black_prop,
              Latino = Latino_prop,
              Asian = Asian_prop,
              Native = Native_prop,
              White =  White_prop
)
```

Return back to long

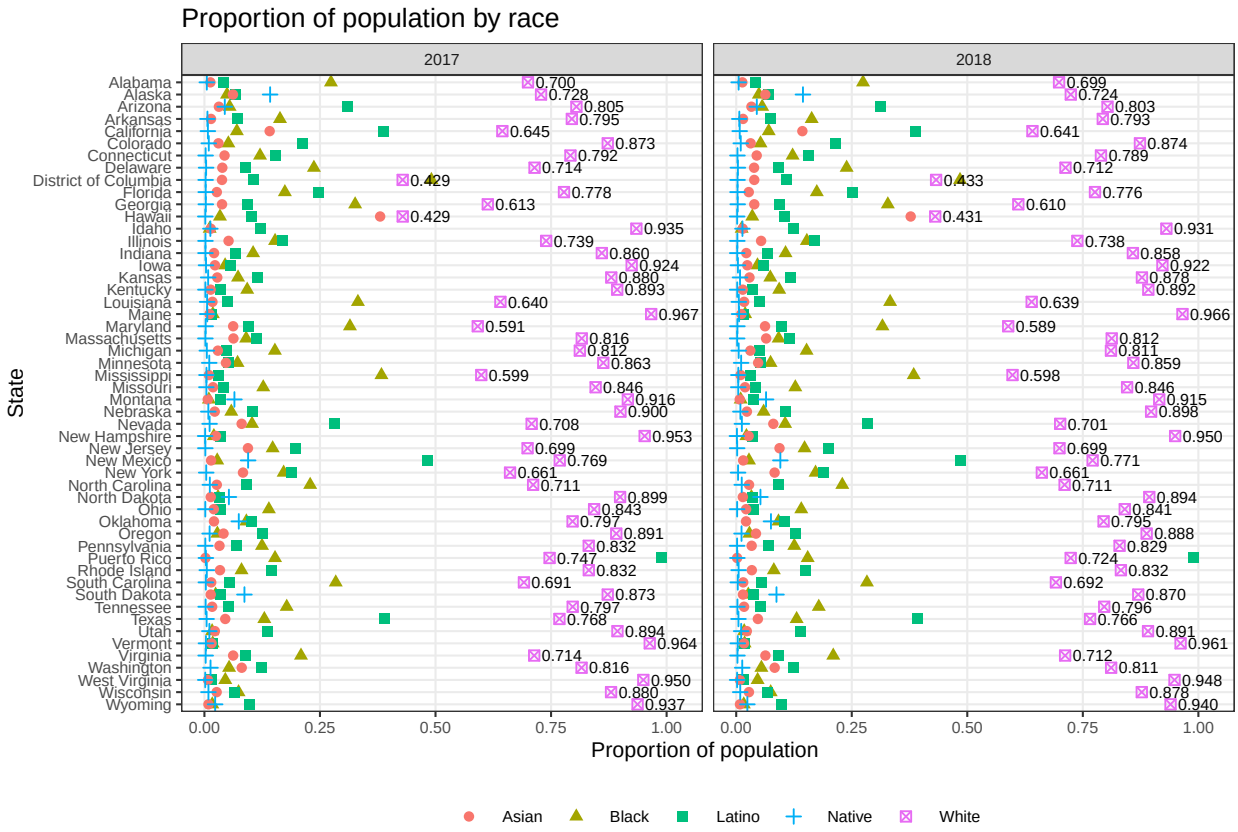
```
library(tidyr)
country_prop.long <- tidyr::gather(country.prop, #old wide data
                                   Race_prop, #variable name for the long column
                                   Proportion, #variable name for values
                                   Black:White)#variables to combine
```

```
country_prop.long %>%
  ggplot(aes(x = Proportion, y = reorder(State, desc(State)))) +
    theme_bw(8) +
  theme(legend.position="bottom",
        legend.title = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid.minor.y = element_blank()) +
  geom_point(aes(shape=Race_prop, color=Race_prop),position=position_dodge(width=0)) +
  labs(title = "Proportion of population by race",
       y = "State",
       x = "Proportion of population ") +
  facet_grid(.~Year)
```

Add conditional text

```
  ggplot(country_prop.long, aes(x = Proportion, y = reorder(State, desc(State)))) +
    theme_bw(8) +
  theme(legend.position="bottom",
        legend.title = element_blank(),
        panel.grid.minor.x = element_blank(),
        panel.grid.minor.y = element_blank()) +
  geom_point(aes(shape=Race_prop, color=Race_prop),position=position_dodge(width=0)) +
```

```
      geom_text(data=country_prop.long[country_prop.long$Race_prop=='White',],
                aes(label = sprintf("%0.3f", Proportion)),
               colour = "black",
                  size = 2,
                 position = position_nudge(x = .06)) +

labs(title = "Proportion of population by race",
     y = "State",
     x = "Proportion of population ") +

facet_grid(.~Year)
```



Maps

```
library(tmap)
library(sf)
library(raster)
library(dplyr)
library(spData)
```

US map

```
us_states_map = tm_shape(us_states, projection = 2163) + tm_polygons() +
  tm_layout(frame = FALSE)
```

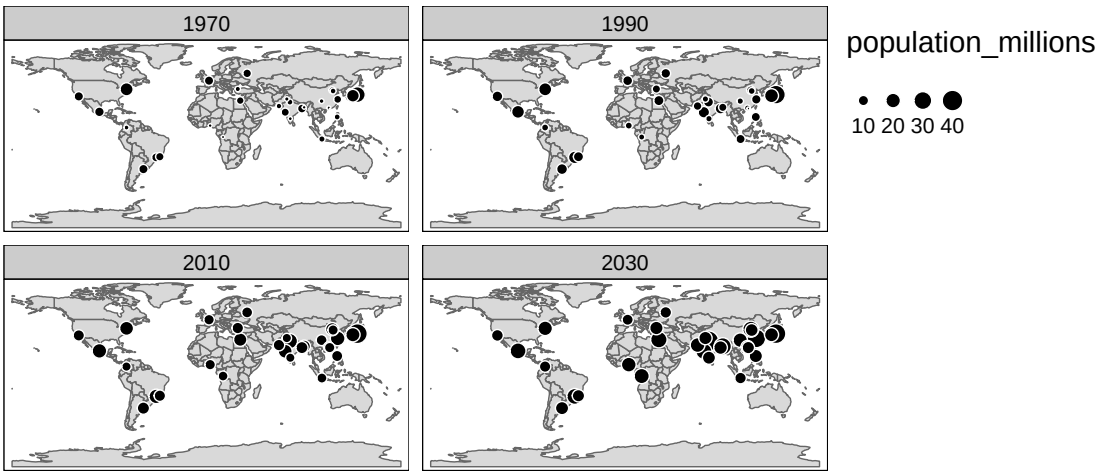
```
hawaii_map = tm_shape(hawaii) + tm_polygons() +
  tm_layout(title = "Hawaii", frame = FALSE, bg.color = NA,
            title.position = c("LEFT", "BOTTOM"))
alaska_map = tm_shape(alaska) + tm_polygons() +
  tm_layout(title = "Alaska", frame = FALSE, bg.color = NA)
```

```
us_states_map
print(hawaii_map, vp = grid::viewport(0.35, 0.1, width = 0.2, height = 0.1)) #setting the location of H
print(alaska_map, vp = grid::viewport(0.15, 0.15, width = 0.3, height = 0.3)) #setting the location of
```

We could put Alaska in the Gulf of Mexico if we wanted!!

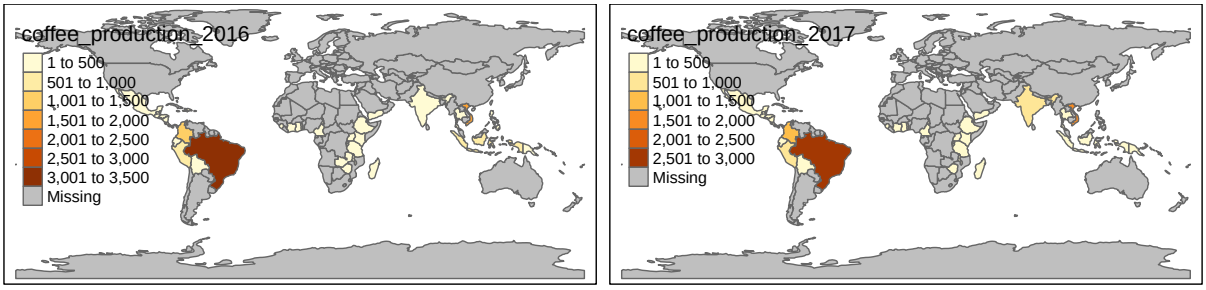
```
us_states_map
print(hawaii_map, vp = grid::viewport(0.35, 0.1, width = 0.2, height = 0.1)) #setting the location of H
print(alaska_map, vp = grid::viewport(0.6, 0.1, width = 0.3, height = 0.3)) #setting the location of al
```


```
library(tmap)


urb_1970_2030 = urban_agglomerations %>%
  filter(year %in% c(1970, 1990, 2010, 2030))

tm_shape(world) +
  tm_polygons() +
  tm_shape(urb_1970_2030) +
  tm_symbols(col = "black", border.col = "white", size = "population_millions") +
  tm_facets(by = "year", nrow = 2, free.coords = FALSE)
```



World maps

```
library(tmap)


world_coffee = left_join(world, coffee_data, by = "name_long")
facets = c("coffee_production_2016", "coffee_production_2017")
tm_shape(world_coffee) + tm_polygons(facets) +
  tm_facets(nrow = 1, sync = TRUE)
```

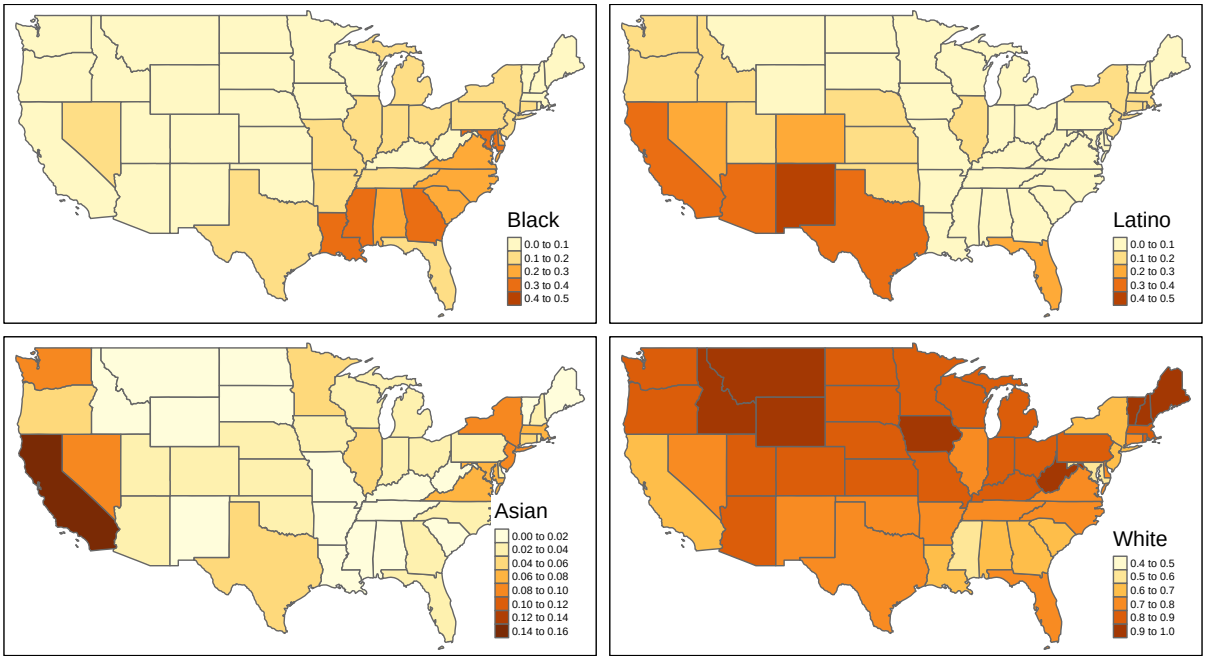


Plot our census data

```
library(tmap)

country_prop.map <- country.prop%>%
  filter(Year == 2018)%>% #only 2018
  mutate(NAME = State) #rename state to match with the map file
```

```
us_race_prop.map <- left_join(us_states, country_prop.map, by = "NAME") #combine the map and census dat


facets = c("Black", "Latino", "Asian", "White") # set facets to be included


us.census.plot <- tm_shape(us_race_prop.map) +
  tm_polygons(facets) + #what should be included-here we have four facets
tm_layout(legend.text.size = 0.5, # adjust layout
          legend.position = c("right","bottom"),
          legend.bg.color = "white",
          legend.bg.alpha = 1) +
tm_facets(nrow = 2, ncol = 2)  #number of rows and columns
```

```
print(us.census.plot)
```



- For more on this example of census mapping see here: Walkthrough of mapping census data with `tidycensus`

Conclusions

Always think: **"What information helps the reader or viewer interpret my data or results?"** A more complicated or advanced plot is not necessarily always better. In fact, these plots may confuse those who are not familiar with your method, approach, or data.

Other tips/notes:

- Consider the size of text
- Be careful with color. Are the colors too similar? Jarring? Are they visible to people with color blindness or other visual impairments?
- For the above point, combining both different colors and line/point types
- Journals may not want color in your final manuscripts

More Resources

- Data visualization in R
- Regression output visualization with `jtools`
- Vignette and examples of Interplot
- Plotting simple effects in regression models
- An introduction to the `margins` command