Lab 6

Jaime Lin

11:59PM April 15, 2021

#Visualization with the package ggplot2

I highly recommend using the ggplot cheat sheet as a reference resource. You will see questions that say "Create the best-looking plot". Among other things you may choose to do, remember to label the axes using real English, provide a title and subtitle. You may want to pick a theme and color scheme that you like and keep that constant throughout this lab. The default is fine if you are running short of time.

Load up the `GSSvocab` dataset in package `carData` as `X` and drop all observations with missing measurements. This will be a very hard visualization exercise since there is not a good model for vocab.

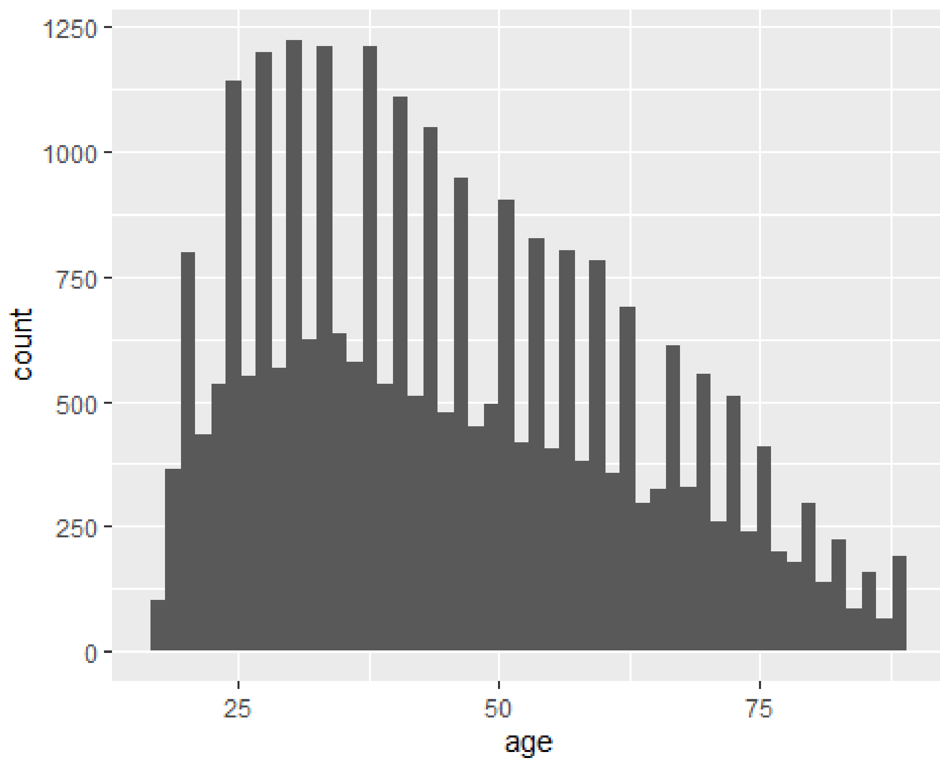
```
pacman::p_load(carData)
data(GSSvocab)
GSSvocab = na.omit(GSSvocab)
```

Briefly summarize the documentation on this dataset. What is the data type of each variable? What do you think is the response variable the collectors of this data had in mind?

This dataset will display the characteristics of the person. The data type in each variable is double. I think this will show how the characteristic of a person is mixed in the sample.

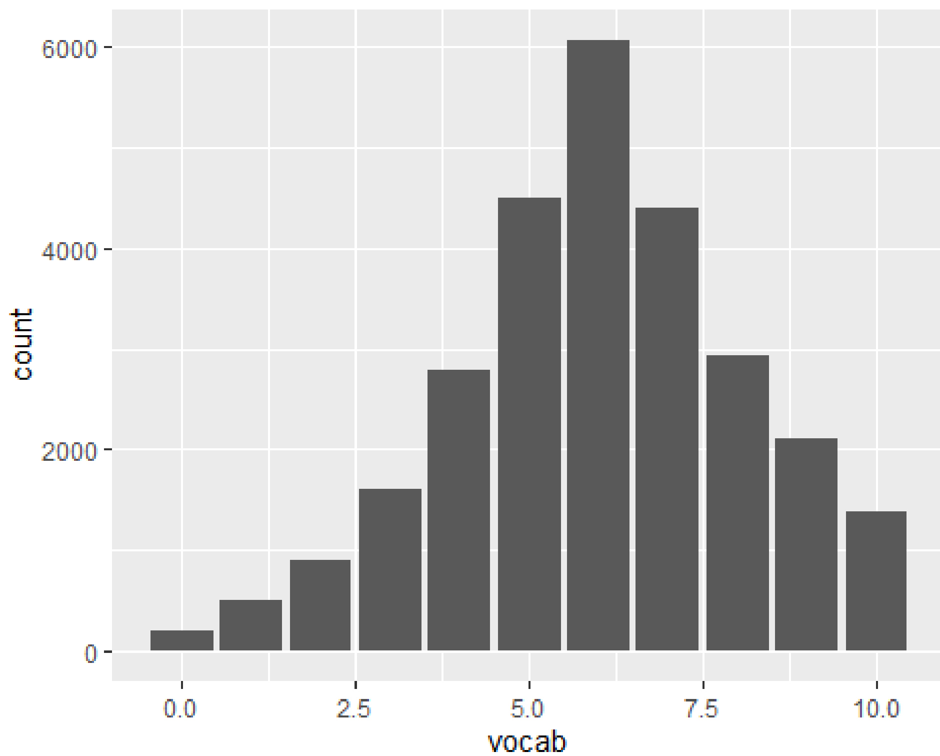
Create two different plots and identify the best-looking plot you can to examine the `age` variable. Save the best looking plot as an appropriately-named PDF.

```
pacman::p_load(ggplot2)
ggplot(GSSvocab) +
  aes(x=age) +
  geom_histogram(bins = 50)
```



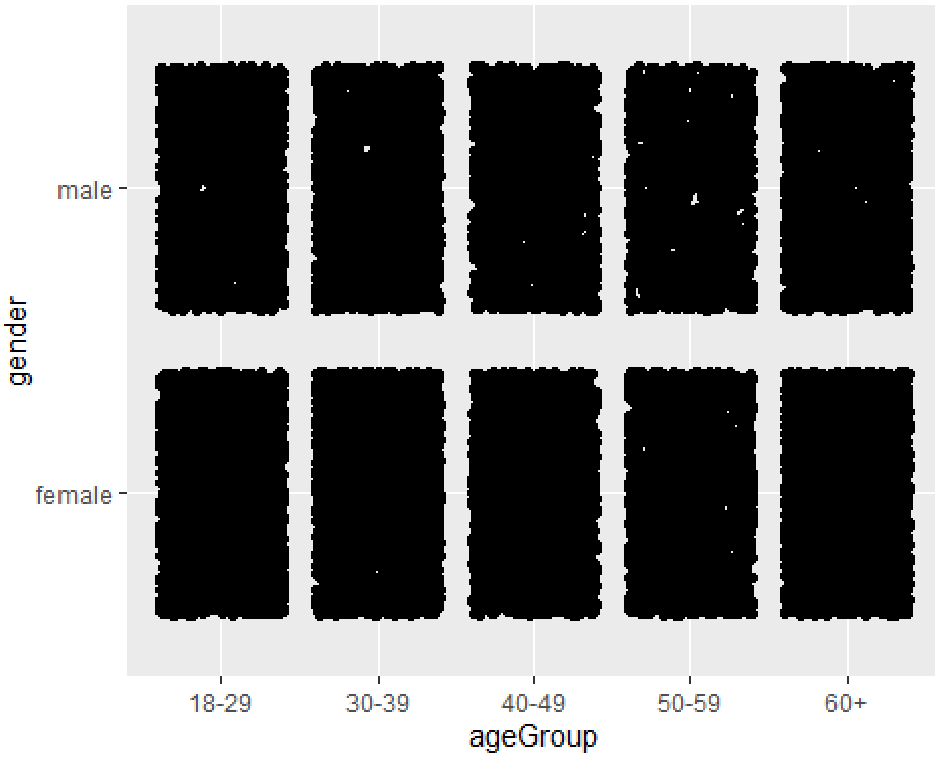
Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

```
ggplot(GSSvocab) +
  aes(x=vocab) +
  geom_bar(bins = 50)

## Warning: Ignoring unknown parameters: bins
```



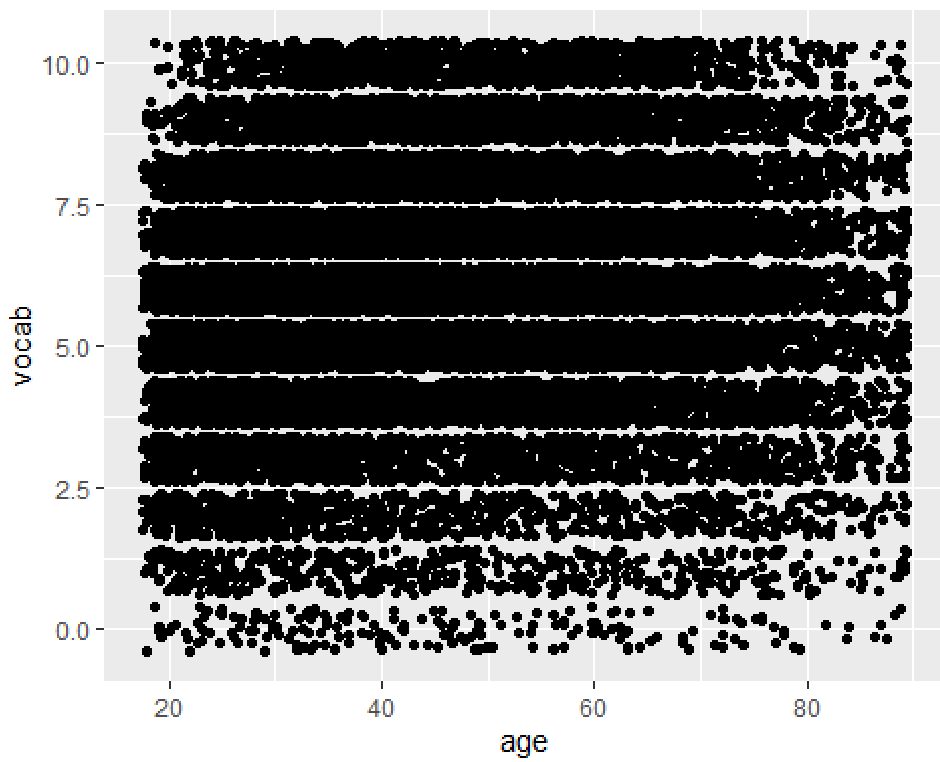
Create the best-looking plot you can to examine the ageGroup variable by gender. Does there appear to be an association? There are many ways to do this.

```
ggplot(GSSvocab) +
  aes(x = ageGroup, y= gender) +
  geom_jitter(bins = 0.5)

## Warning: Ignoring unknown parameters: bins
```



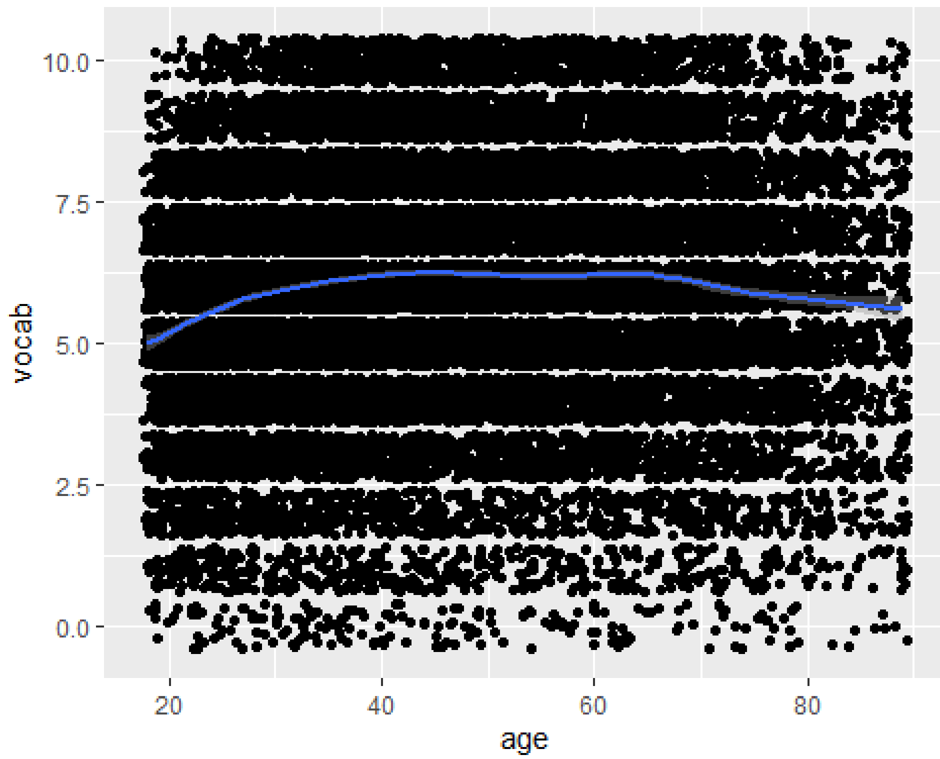
Create the best-looking plot you can to examine the vocab variable by age. Does there appear to be an association?

```
ggplot(GSSvocab) +
  aes(x = age, y= vocab) +
  geom_jitter()
```



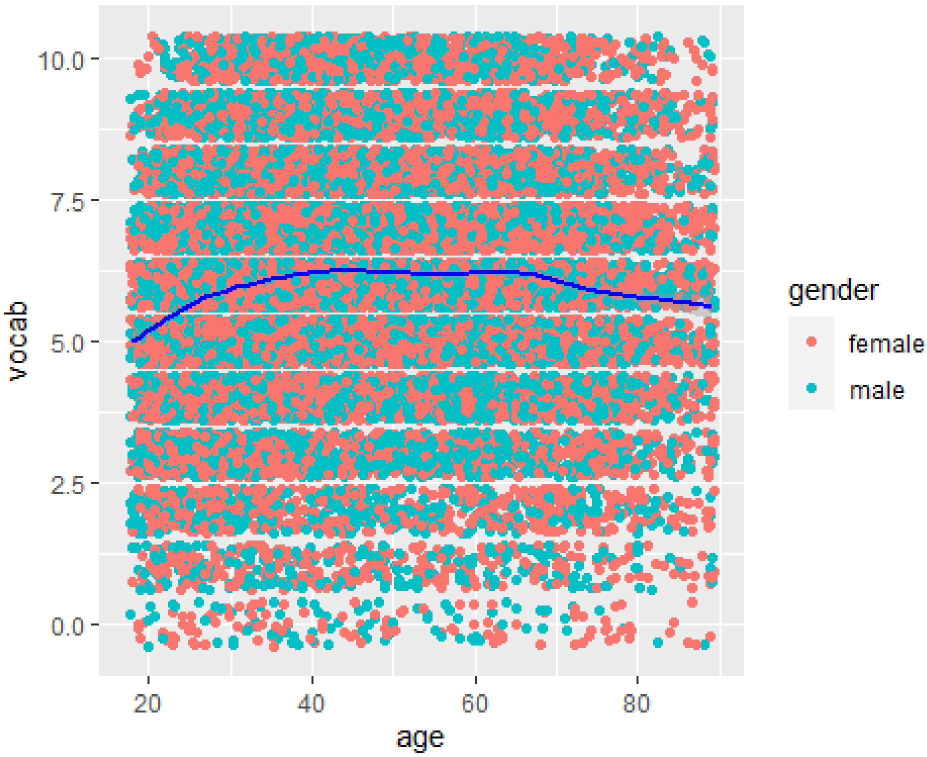
Add an estimate of $f(x)$ using the smoothing geometry to the previous plot. Does there appear to be an association now?

```
ggplot(GSSvocab) +
  aes(x = age, y= vocab) +
  geom_jitter()+
  geom_smooth()

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



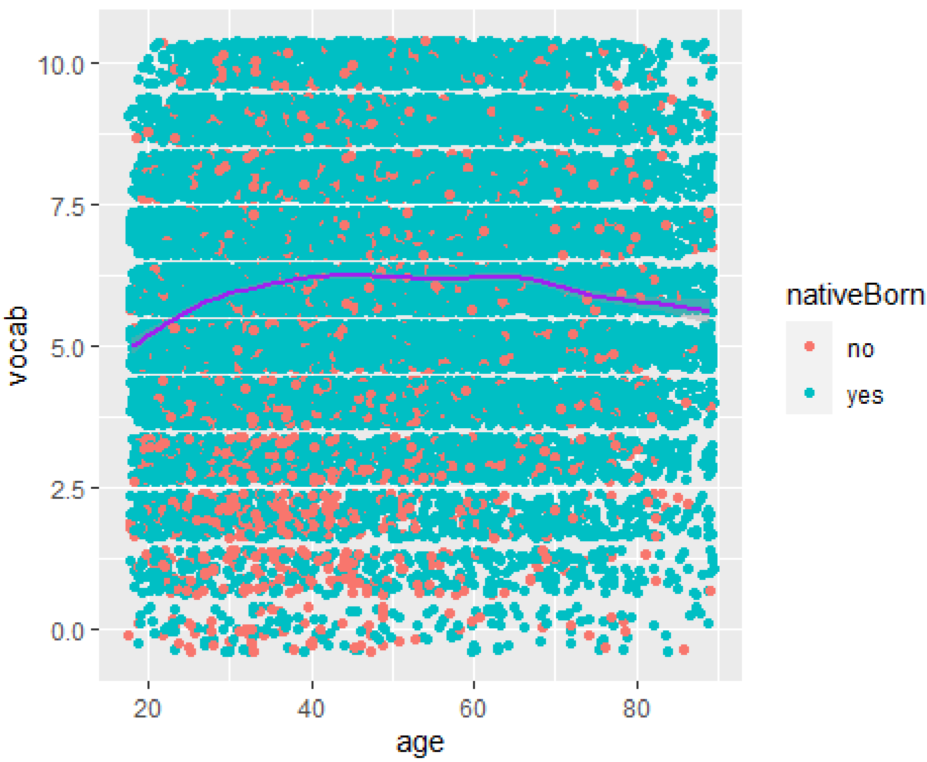
Using the plot from the previous question, create the best looking plot overloading with variable gender. Does there appear to be an interaction of gender and age?

```
ggplot(GSSvocab) +
  aes(x = age, y= vocab) +
  geom_jitter(aes(col = gender))+
  geom_smooth(col = "blue")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



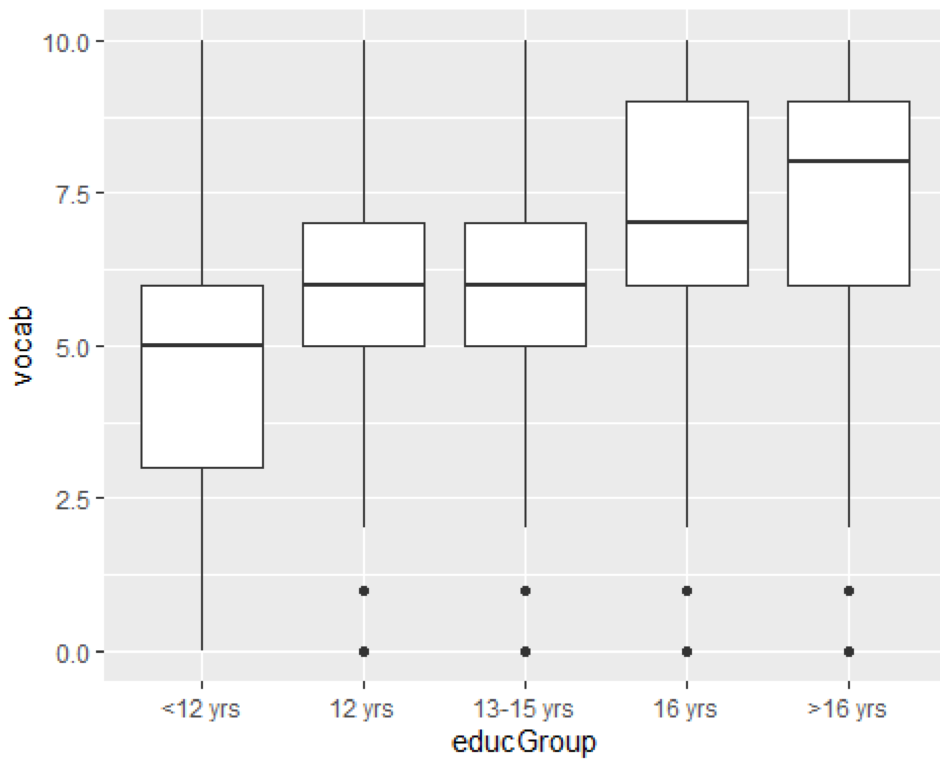
Using the plot from the previous question, create the best looking plot overloading with variable `nativeBorn`. Does there appear to be an interaction of `nativeBorn` and `age`?

```
ggplot(GSSvocab) +
  aes(x = age, y= vocab) +
  geom_jitter(aes(col = nativeBorn))+
  geom_smooth(col = "purple")

## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

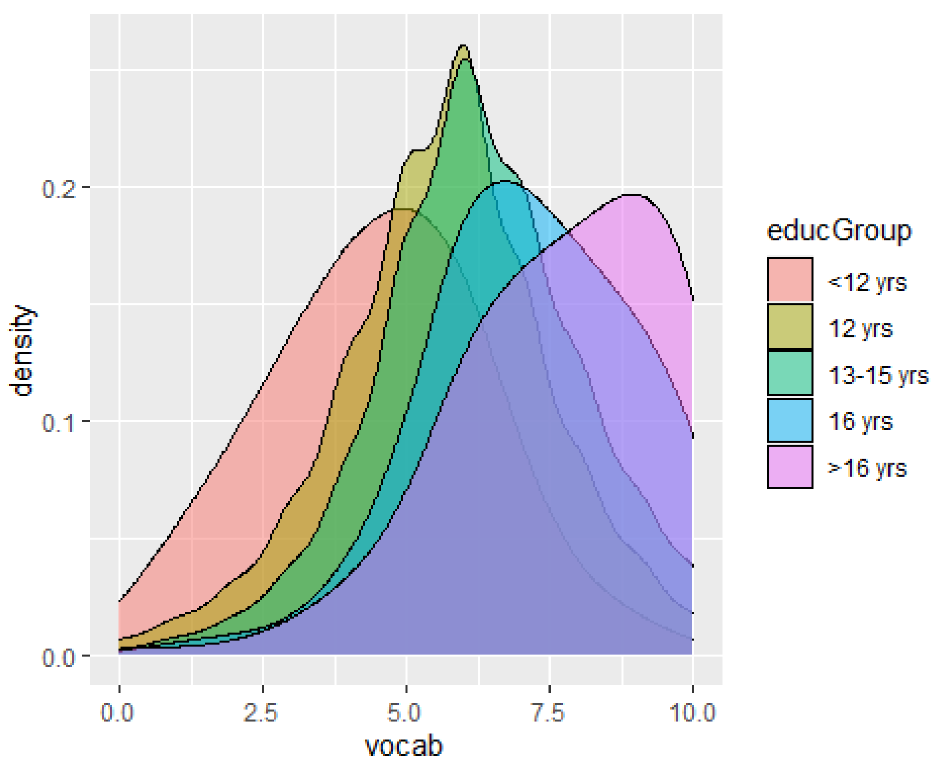


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

```
ggplot(GSSvocab) +
  aes(x = educGroup, y = vocab)+
  geom_boxplot()
```

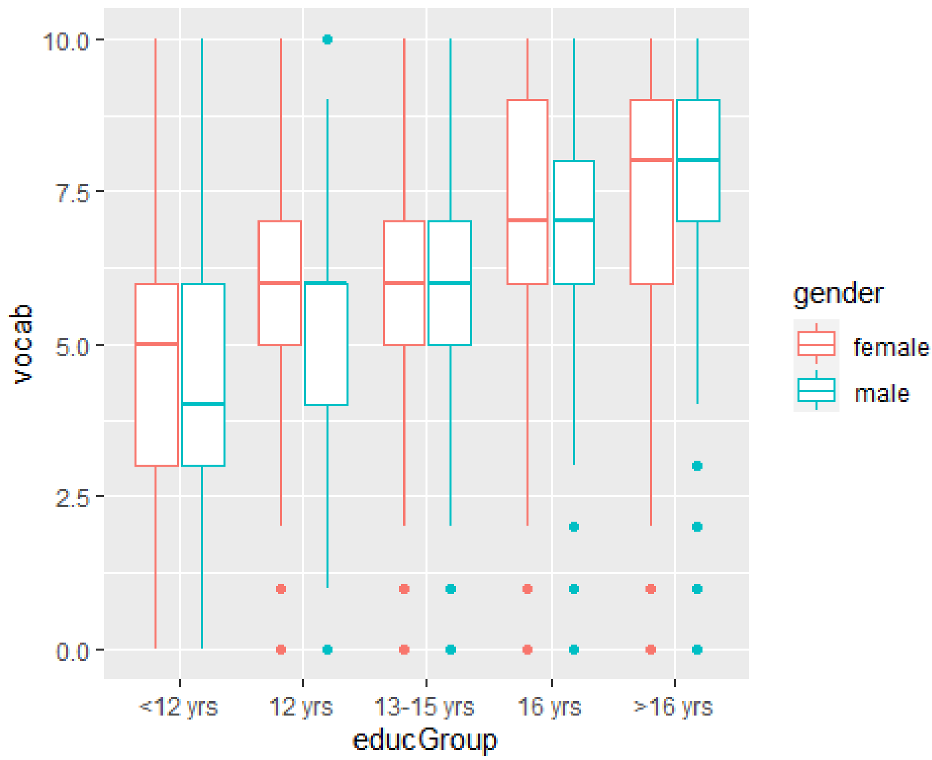



```
ggplot(GSSvocab) +
  aes(x = vocab)+
  geom_density(aes(fill = educGroup), adjust = 2, alpha = .5)
```



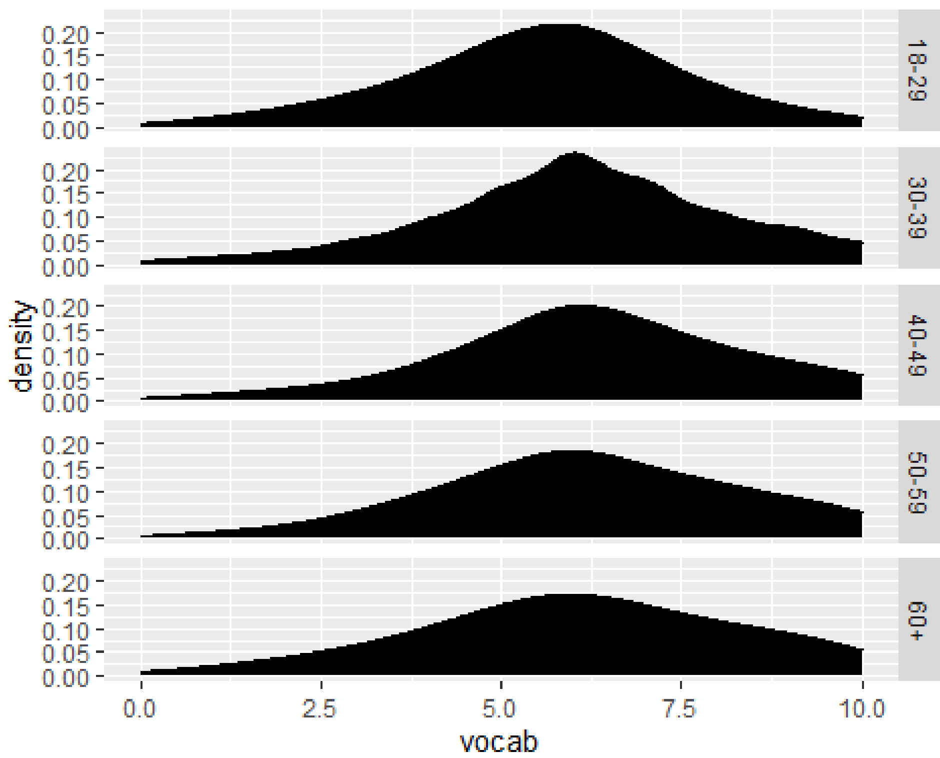
Using the best-looking plot from the previous question, create the best looking overloading with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?

```
ggplot(GSSvocab) +
  aes(x = educGroup, y = vocab)+
  geom_boxplot(aes(col = gender))
```



Using facets, examine the relationship between `vocab` and `ageGroup`. You can drop year level (`Other`). Are we getting dumber?

```
ggplot(GSSvocab) +
  aes(x = vocab)+
  geom_density(adjust = 2, fill = "black")+
  facet_grid(ageGroup~.)
```



Probability Estimation and Model Selection

Load up the `adult` in the package `ucidata` dataset and remove missingness and the variable `fnlwgt`:

```
pacman::p_load_gh("coatless/ucidata")
data(adult)
adult = na.omit(adult) #kill any observations with missingness
adult$fnlwgt = NULL
```

Cast income to binary where 1 is the `>50K` level.

```
adult$income = ifelse(adult$income == ">50k", 1, 0)
```

We are going to do some dataset cleanup now. But in every cleanup job, there's always more to clean! So don't expect this cleanup to be perfect.

Firstly, a couple of small things. In variable `marital_status` collapse the levels `Married-AF-spouse` (armed force marriage) and `Married-civ-spouse` (civilian marriage) together into one level called `Married`. Then in variable `education` collapse the levels `1st-4th` and `Preschool` together into a level called `<=4th`.

```
adult$marital_status = as.character(adult$marital_status)
adult$marital_status = ifelse(adult$marital_status == "Married-AF-spouse" |
adult$marital_status == "Married-civ-spouse", "Married",
adult$marital_status)
```

```
adult$marital_status = as.factor(adult$marital_status)

adult$education = as.character(adult$education)
adult$education = ifelse(adult$marital_status == "Married-AF-spouse" |
adult$marital_status == "Married-civ-spouse", "Married", adult$education)
adult$education = as.factor(adult$education)
```

Create a model matrix `Xmm` (for this prediction task on just the raw features) and show that it is *not* full rank (i.e. the result of `ncol` is greater than the result of `Matrix::rankMatrix`).

```
xmm = model.matrix(income~., adult)
ncol(xmm)

## [1] 96

Matrix::rankMatrix(xmm)

## [1] 94
## attr(,"method")
## [1] "tolNorm2"
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.697087e-12
```

Now tabulate and sort the variable `native_country`.

```
sort(table(adult$native_country))

##
##         Holand-Netherlands                   Scotland
##                          1                         11
##                   Honduras                    Hungary
##                         12                         13
## Outlying-US(Guam-USVI-etc)                 Yugoslavia
##                         14                         16
##                       Laos                   Thailand
##                         17                         17
##                   Cambodia            Trinadad&Tobago
##                         18                         18
##                       Hong                    Ireland
##                         19                         24
##                    Ecuador                     France
##                         27                         27
##                     Greece                       Peru
##                         29                         30
##                  Nicaragua                   Portugal
##                         33                         34
##                      Haiti                       Iran
##                         42                         42
##                     Taiwan                   Columbia
```

```
##                           42                           56
##                       Poland                        Japan
##                           56                           59
##                    Guatemala                      Vietnam
##                           63                           64
##           Dominican-Republic                        China
##                           67                           68
##                        Italy                        South
##                           68                           71
##                      Jamaica                      England
##                           80                           86
##                         Cuba                  El-Salvador
##                           92                          100
##                        India                       Canada
##                          100                          107
##                  Puerto-Rico                      Germany
##                          109                          128
##                  Philippines                       Mexico
##                          188                          610
##                United-States
##                        27503
```

Do you see rare levels in this variable? Explain why this may be a problem.

Yes, there a name and a number. These show the name of the country and the amount of people with that characteristics.

Collapse all levels that have less than 50 observations into a new level called `other`. This is a very common data science trick that will make your life much easier. If you can't hope to model rare levels, just give up and do something practical! I would recommend first casting the variable to type "character" and then do the level reduction and then recasting back to type `factor`. Tabulate and sort the variable `native_country` to make sure you did it right.

```
adult$native_country = as.character(adult$native_country)
#adult$native_country = ifelse(adult$native_country %in% names(tab[tab <
50]), "other", adult$native_country)
adult$native_country = as.factor(adult$native_country)
```

We're still not done getting this data down to full rank. Take a look at the model matrix just for `workclass` and `occupation`. Is it full rank?

```
xmm = model.matrix(income~workclass + occupation, adult)
ncol(xmm)

## [1] 21

Matrix::rankMatrix(xmm)

## [1] 20
## attr(,"method")
## [1] "tolNorm2"
```

```
## attr(,"useGrad")
## [1] FALSE
## attr(,"tol")
## [1] 6.697087e-12
```

These variables are similar and they probably should be interacted anyway eventually. Let's combine them into one factor. Create a character variable named `worktype` that is the result of concatenating `occupation` and `workclass` togetther with a ":" in between. Use the `paste` function with the `sep` argument (this casts automatically to type `character`). Then tabulate its levels and sort.

```
worktype = paste(adult$occupation:adult$workclass, sep = "GSSvocab")
```

Like the `native_country` exercise, there are a lot of rare levels. Collapse levels with less than 100 observations to type `other` and then cast this variable `worktype` as type `factor`. Recheck the tabulation to ensure you did this correct.

```
adult$native_country = as.character(adult$native_country)
#adult$native_country = ifelse(adult$native_country %in% names(tab[tab < 100]), "other", adult$native_country)
adult$native_country = as.factor(adult$native_country)
#adult$native_country = as.worktype(adult$native_country)
```

To do at home: merge the two variables `relationship` and `marital_status` together in a similar way to what we did here.

```
Merge = adult$relationship:adult$marital_status
```

We are finally ready to fit some probability estimation models for `income`! In lecture 16 we spoke about model selection using a cross-validation procedure. Let's build this up step by step. First, split the dataset into `Xtrain`, `ytrain`, `Xtest`, `ytest` using K=5.

```
set.seed(1984)
K = 5
test_prop = 1 / K
train_indices = sample(1 : nrow(adult), round((1 - test_prop) * nrow(adult)))
adult_train = adult[train_indices, ]
y_train = adult_train$income
X_train = adult_train
X_train$income = NULL
test_indices = setdiff(1 : nrow(adult), train_indices)
adult_test = adult[test_indices, ]
y_test = adult_test$income
X_test = adult_test
X_test$income = NULL
```

Create the following four models on the training data in a `list` objected named `prob_est_mods`: logit, probit, cloglog and cauchit (which we didn't do in class but might as well). For the linear component within the link function, just use the vanilla raw features using the `formula` object `vanilla`. Each model's key in the list is its link function name + "-vanilla". One for loop should do the trick here.

```
link_functions = c("logit", "probit", "cloglog", "cauchit")
vanilla = income ~ .
prob_est_mods = list()

for (link_function in link_functions) {
  prob_est_mods[[ paste(link_function, "vanilla", sep = "-")]] = glm(vanilla,
adult_train, family = binomial(link = link_function))
}
```

```
## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: algorithm did not converge
```

Now let's get fancier. Let's do some variable transforms. Add `log_capital_loss` derived from `capital_loss` and `log_capital_gain` derived from `capital_gain`. Since there are zeroes here, use log_x = log(1 + x) instead of log_x = log(x). That's always a neat trick. Just add them directly to the data frame so they'll be picked up with the `.` inside of a formula.

```
#TO-DO
```

Create a density plot that shows the age distribution by `income`.

What do you see? Is this expected using common sense?

#TO-DO

Now let's fit the same models with all link functions on a formula called `age_interactions` that uses interactions for `age` with all of the variables. Add all these models to the `prob_est_mods` list.

```
#age_interactions = class ~ #TO-DO

#TO-DO
```

Create a function called `brier_score` that takes in a probability estimation model, a dataframe `X` and its responses `y` and then calculates the brier score.

```
brier_score = function(prob_est_mod, X, y){
 phat = predict(prob_est_mod, x)
 mean(-(y-phat)^2)
}
```

Now, calculate the in-sample Brier scores for all models. You can use the function `lapply` to iterate over the list and pass in in the function `brier_score`.

```
#lapply(prob_est_mods, brier_score, x_train, y_train)
```

Now, calculate the out-of-sample Brier scores for all models. You can use the function `lapply` to iterate over the list and pass in the function `brier_score`.

```
#lapply(prob_est_mods, brier_score, x_test, y_test)
```

Which model wins in sample and which wins out of sample? Do you expect these results? Explain.

#TO-DO

What is wrong with this model selection procedure? There are a few things wrong.

#TO-DO

Run all the models again. This time do three splits: subtrain, select and test. After selecting the best model, provide a true oos Brier score for the winning model.

```
#TO-DO
```