Lab 6

Amy Butler

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, 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.

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

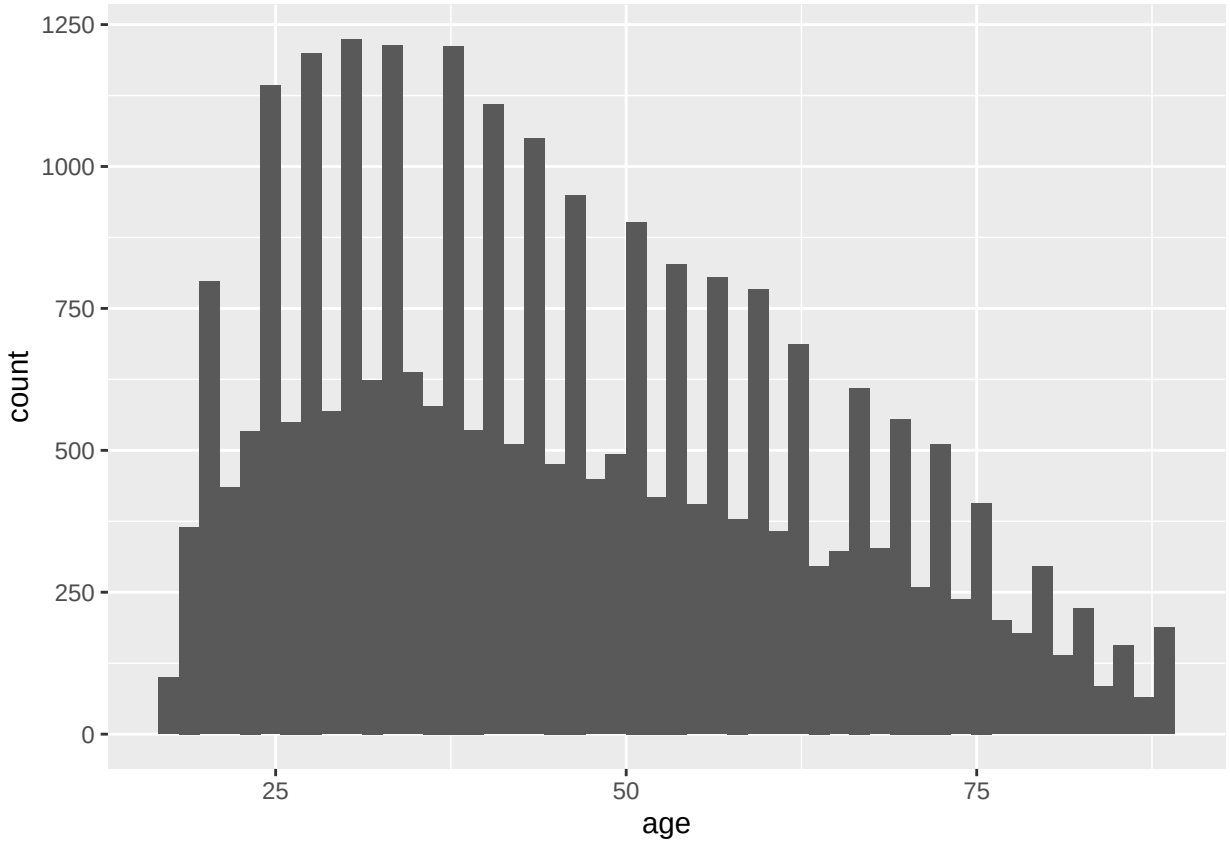
```
## starting httpd help server ... done
```

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?

There are 28867 students taking a vocab test. There are 8 features, some of which are duplicates. The response variable is scores on the vocab test. The response variable could benefit by giving more than 10 questions on the test.

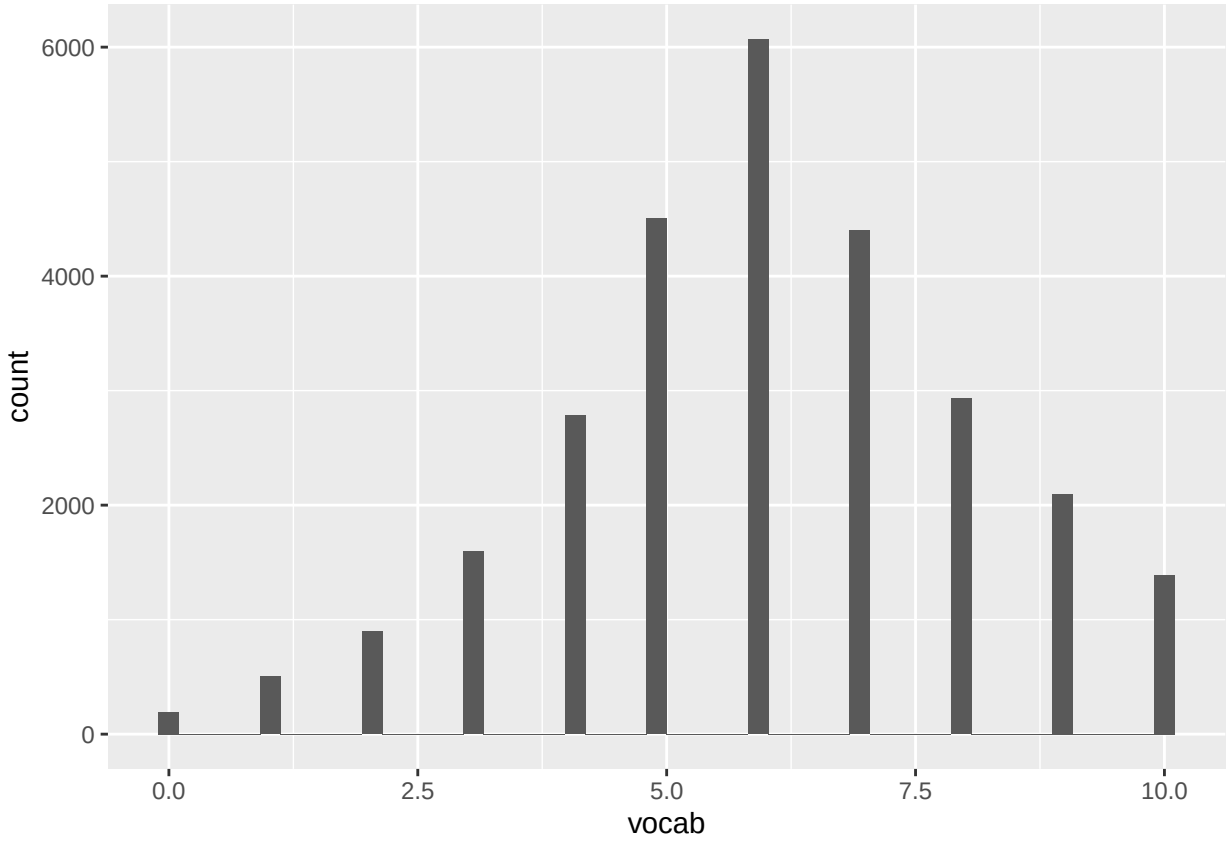
Create two different plots and identify the best-looking plot you can to examine the `age` variable. Save the best looking plot as an appropriately-named PDF.

```
#One continuous variable such as age can use a histogram:
pacman::p_load(ggplot2)
ggplot(GSSvocab)+
  aes(x=age)+
  geom_histogram(bins=50)
```

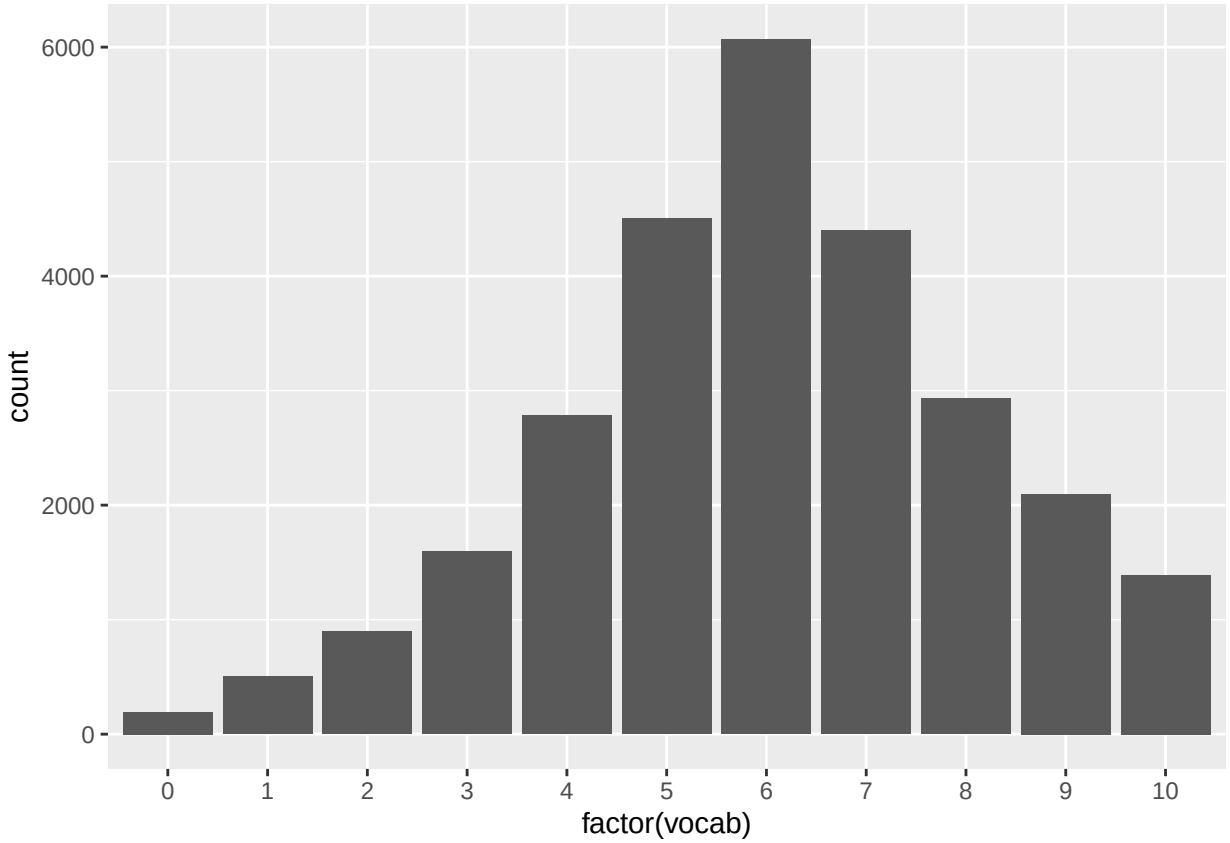


Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

```
#One categorical variable is being used here so a histogram is not the best graph to use.
ggplot(GSSvocab)+
  aes(x=vocab)+
  geom_histogram(bins=50)
```

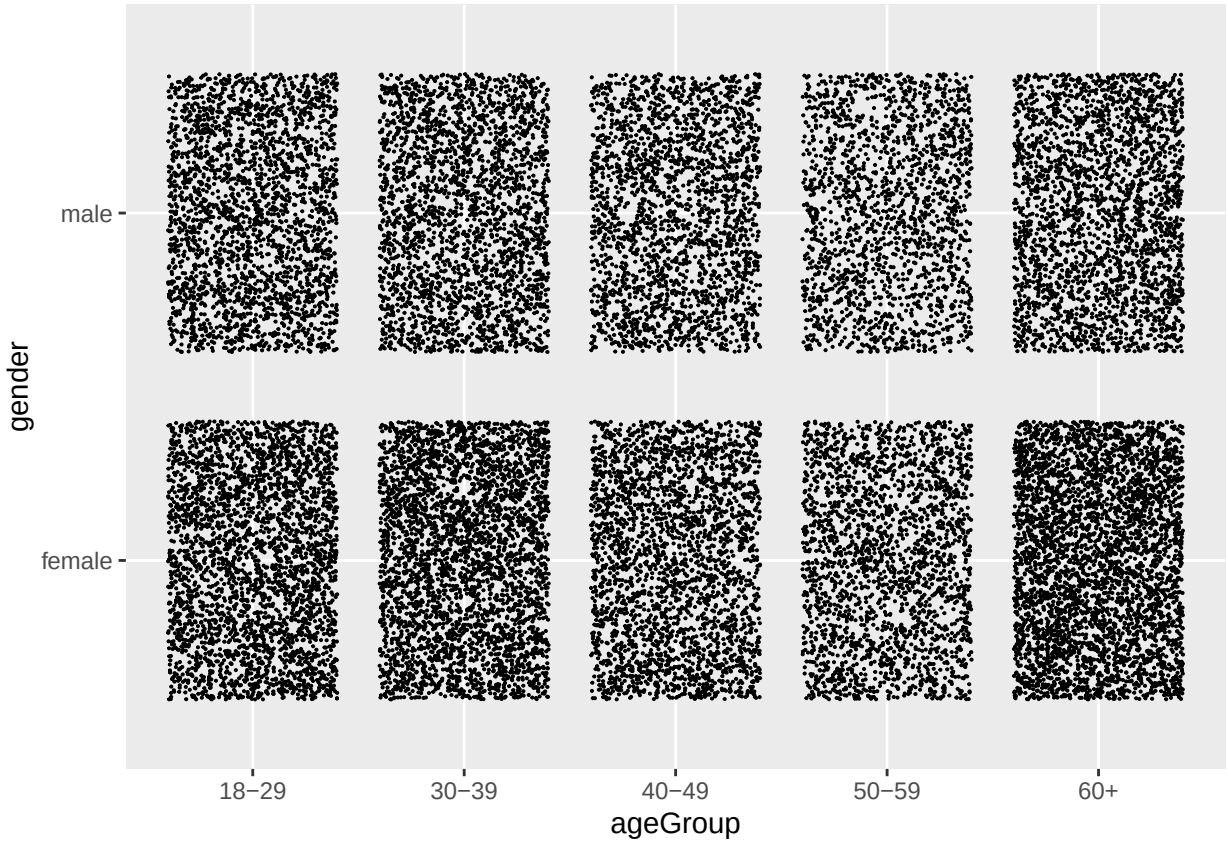



```
#It is better to use a bar graph for an ordinal variable.
ggplot(GSSvocab)+
  aes(x=factor(vocab))+
  geom_bar()
```



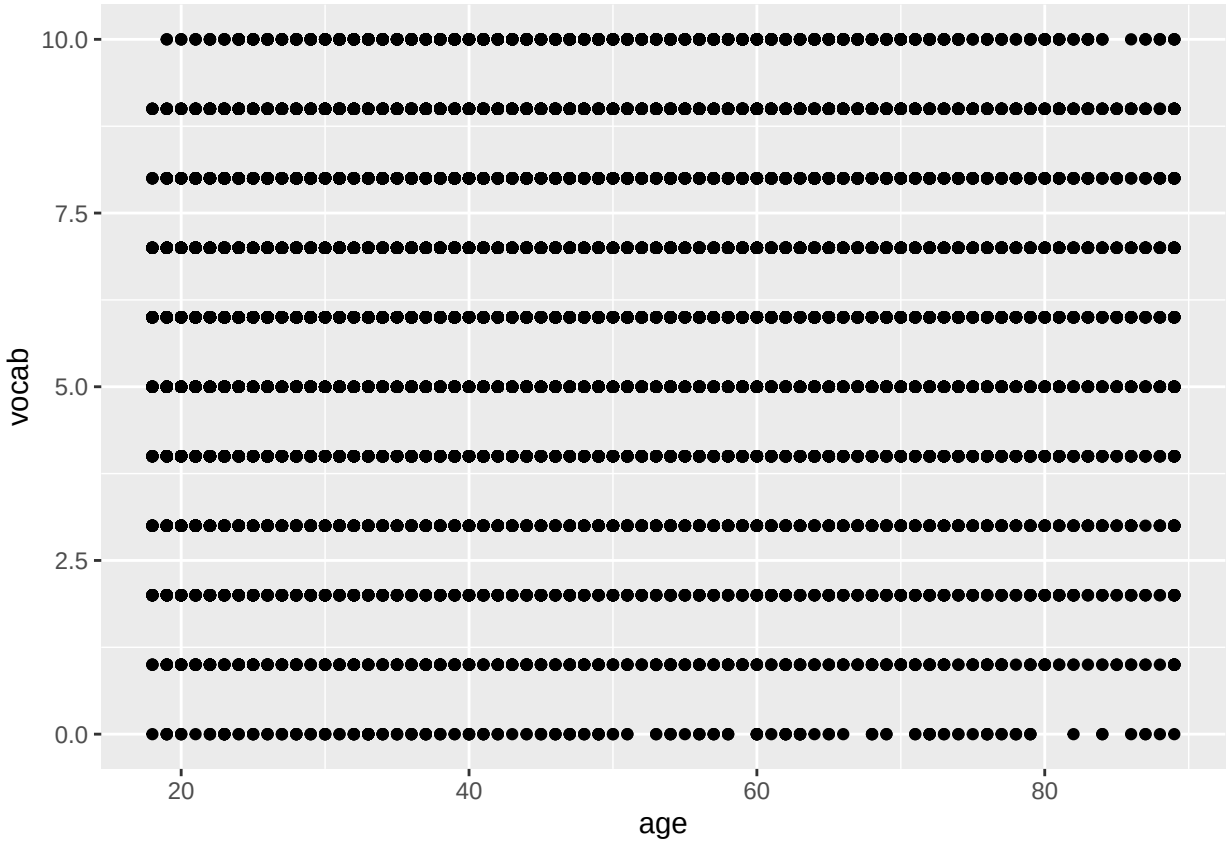
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```
#ageGroup is a categorical variable.
#gender is a binary categorical variable.
#In this case we are using two categorical variables and therefore a jitter plot should be used.
ggplot(GSSvocab)+
  aes(x=ageGroup, y=gender)+
  geom_jitter(size=.05) #Makes the size of the points smaller to make the graph more readable
```

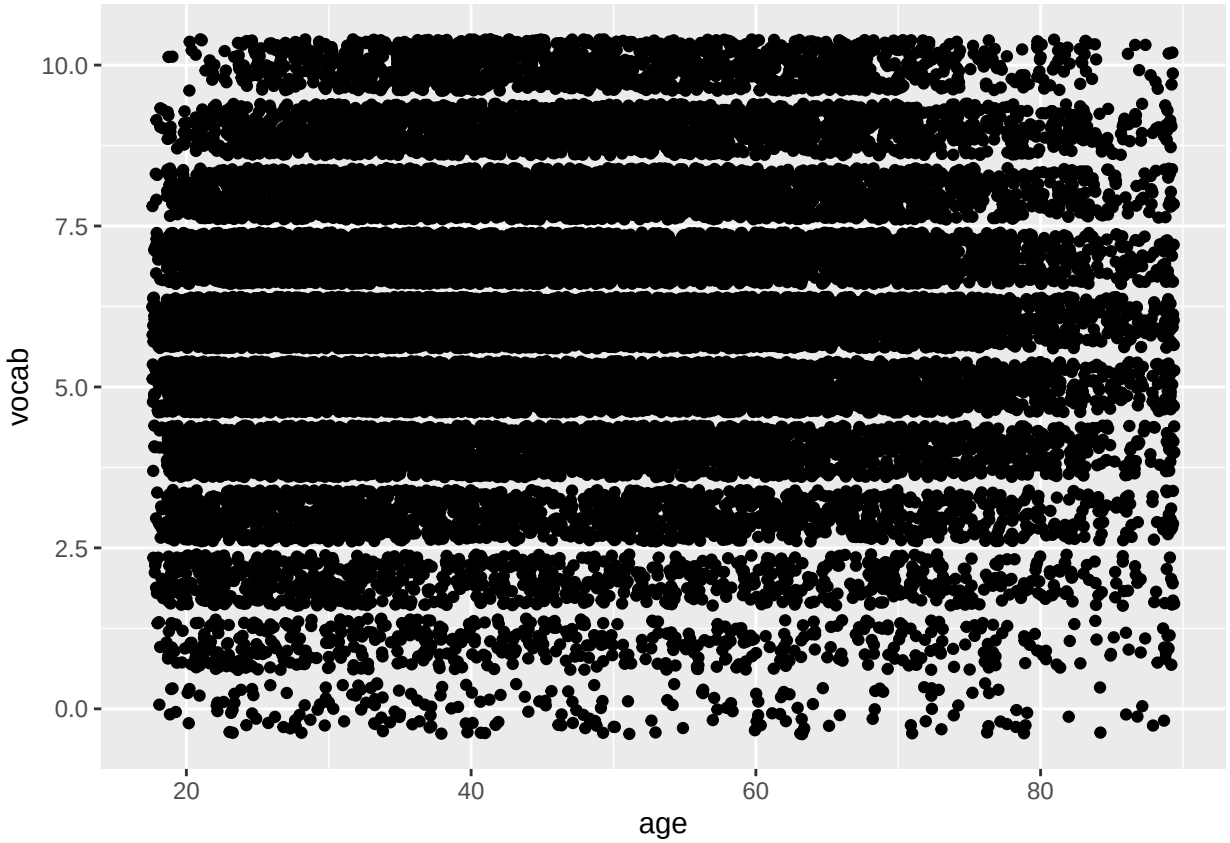


Create the best-looking plot you can to examine the `vocab` variable by `age`. Does there appear to be an association?

```
#vocab is a continuous variable
#age is a continuous variable
#With two continuous variables we can use a scatter plot. This isnt a great plot.
ggplot(GSSvocab)+
  aes(x=age, y=vocab)+
  geom_point()
```



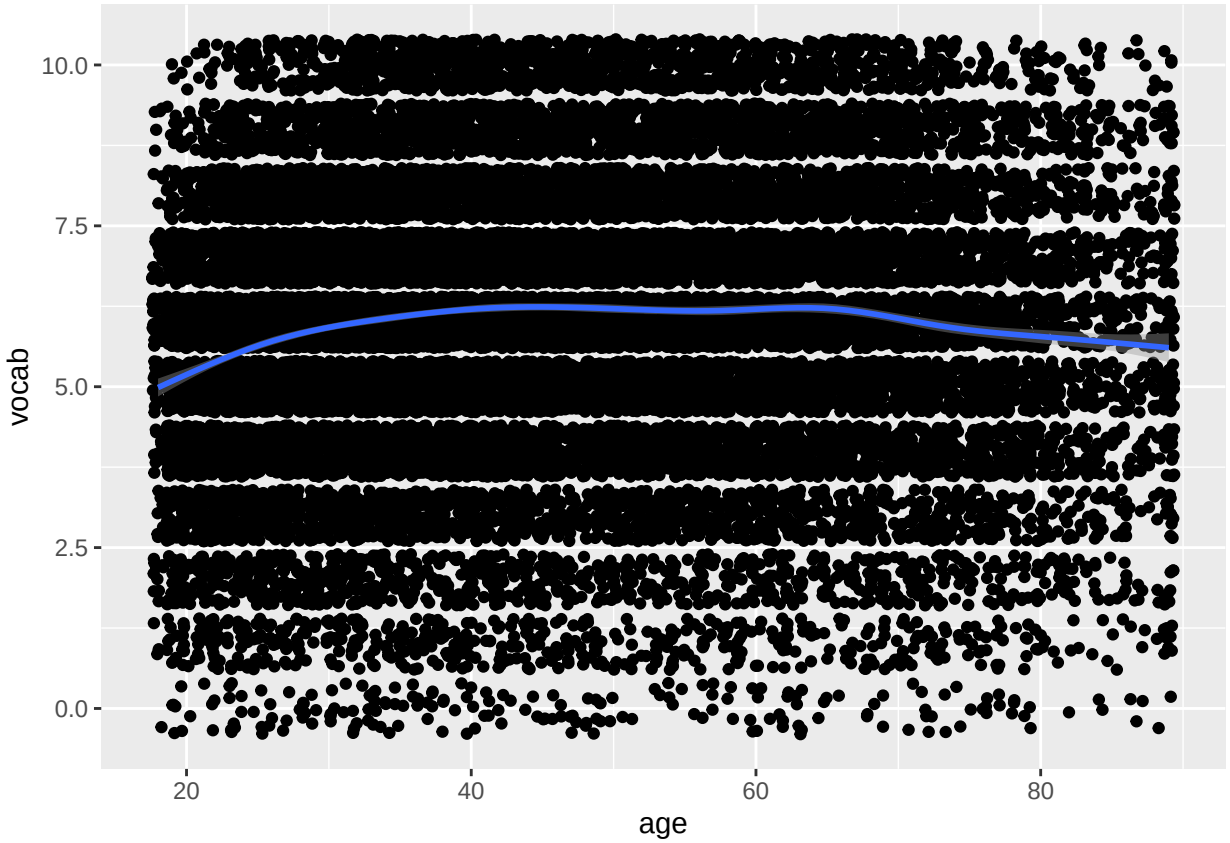

```
#We can try a jitter plot but this also isn't great.
ggplot(GSSvocab)+
  aes(x=age, y=vocab)+
  geom_jitter()
```



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")'
```

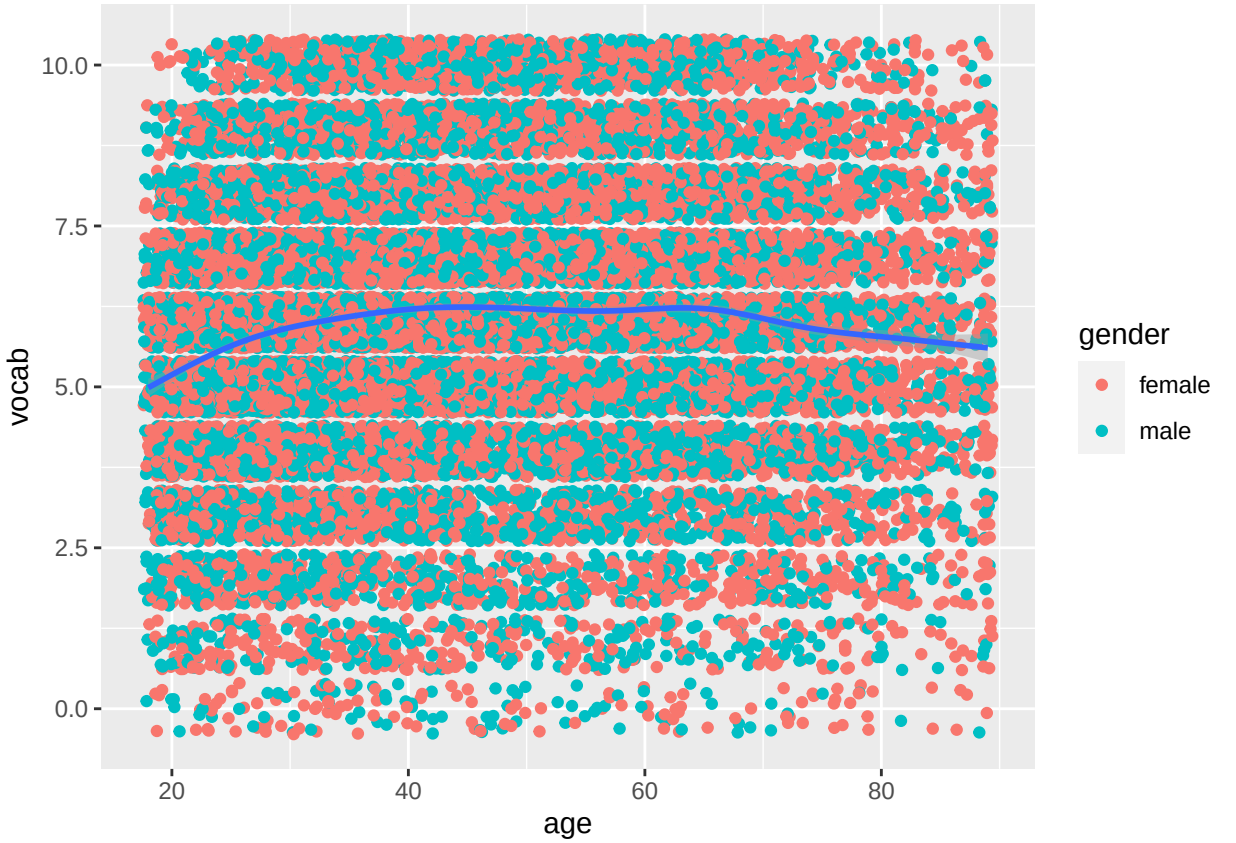



```
#The plot shows that vocabulary becomes better as you learn and grow but when getting older vocabulary
```

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`?

```
#Add a third variable to the previous graph to see if there is a distinction.
ggplot(GSSvocab)+
  aes(x=age, y=vocab)+
  geom_jitter(aes(col=gender))+
  geom_smooth()
```

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

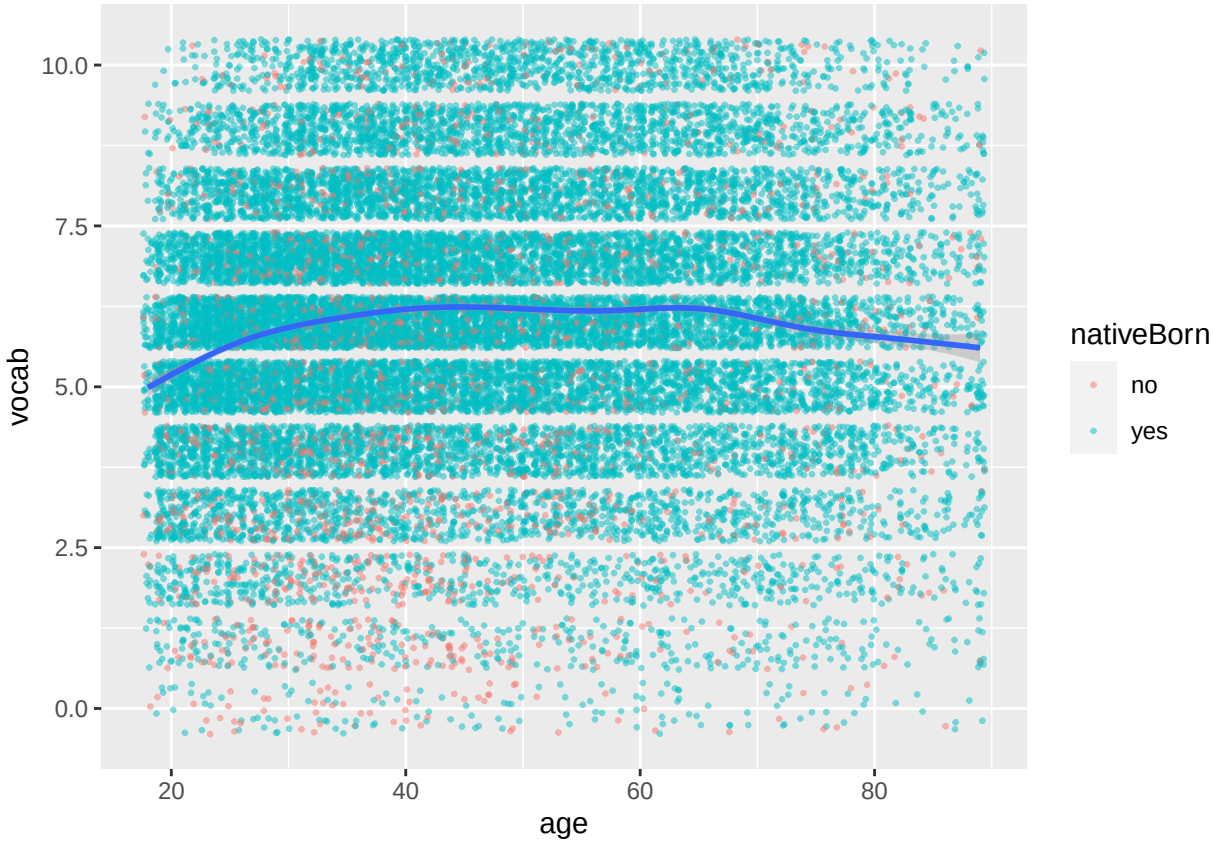



```
#There is no obvious interaction in the plot produced.
```

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`?

```
#Replace gender with ageBorn as the overloading variable in the previous graph to see if there is a dis
ggplot(GSSvocab)+
  aes(x=age, y=vocab)+
  geom_jitter(aes(col=nativeBorn), size=.5, alpha=.5)+
  geom_smooth()
```

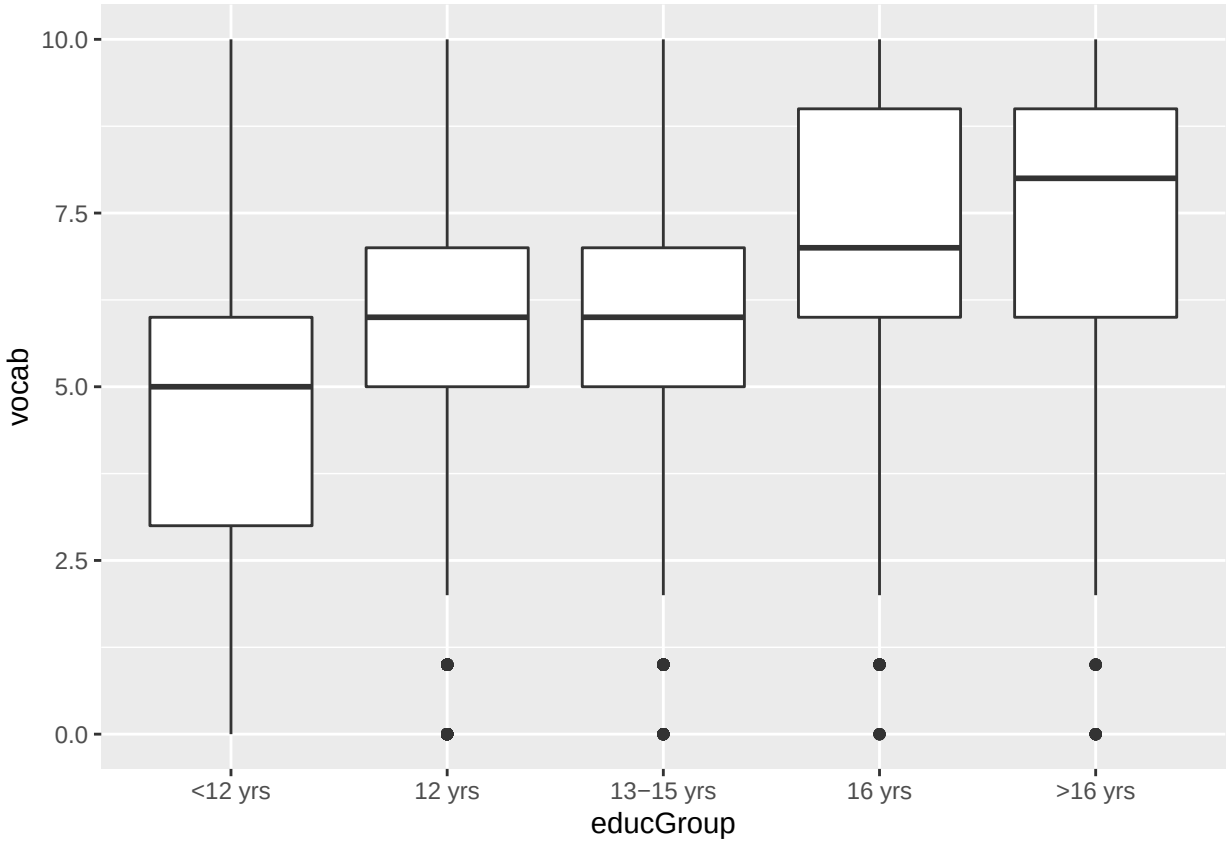
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



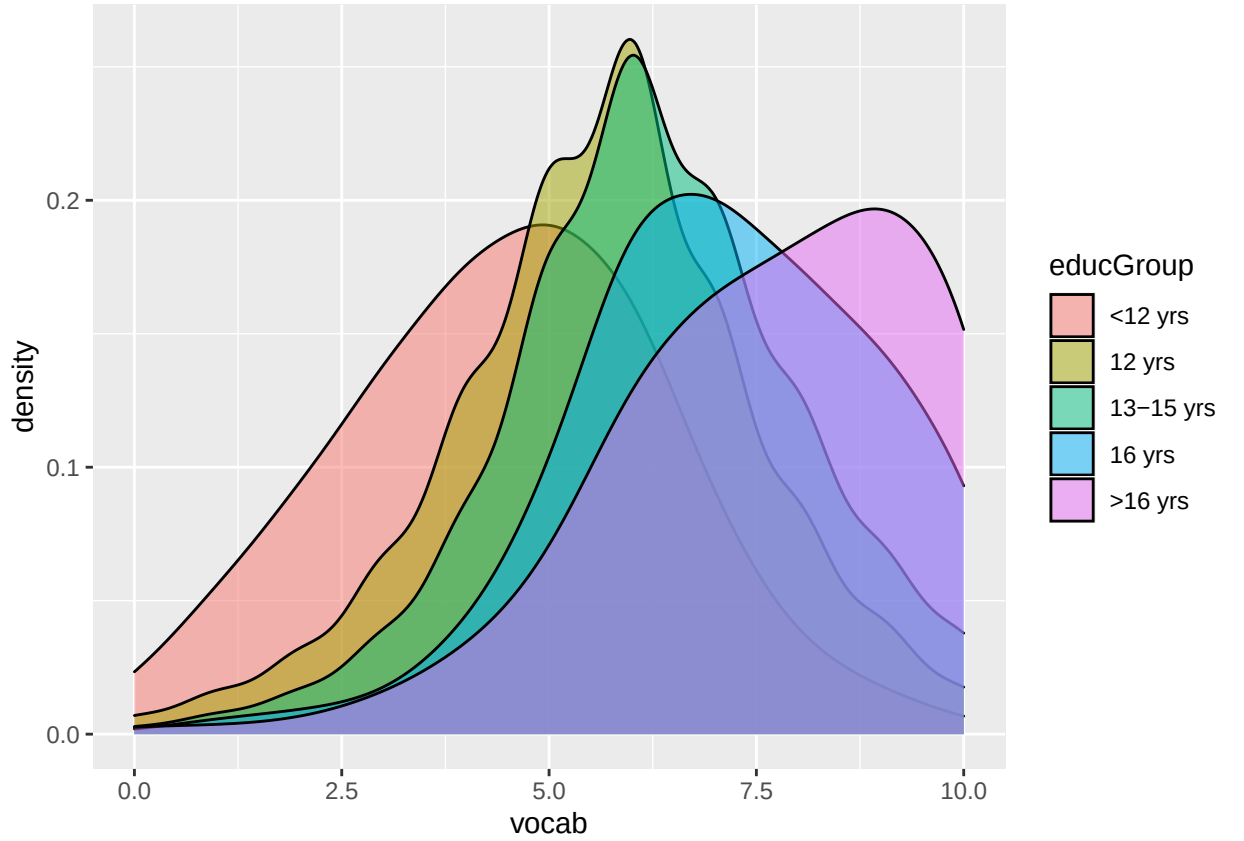

```
#The plot shows that people who are native born tend to have a higher vocabulary then people who aren't
```

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?

```
#vocab is a continuous variable (vocab can be considered categorical or continuous)
#educGroup is a categorical variable
#We can use a box plot in this scenario.
ggplot(GSSvocab)+
  aes(x=educGroup, y=vocab)+
  geom_boxplot()
```



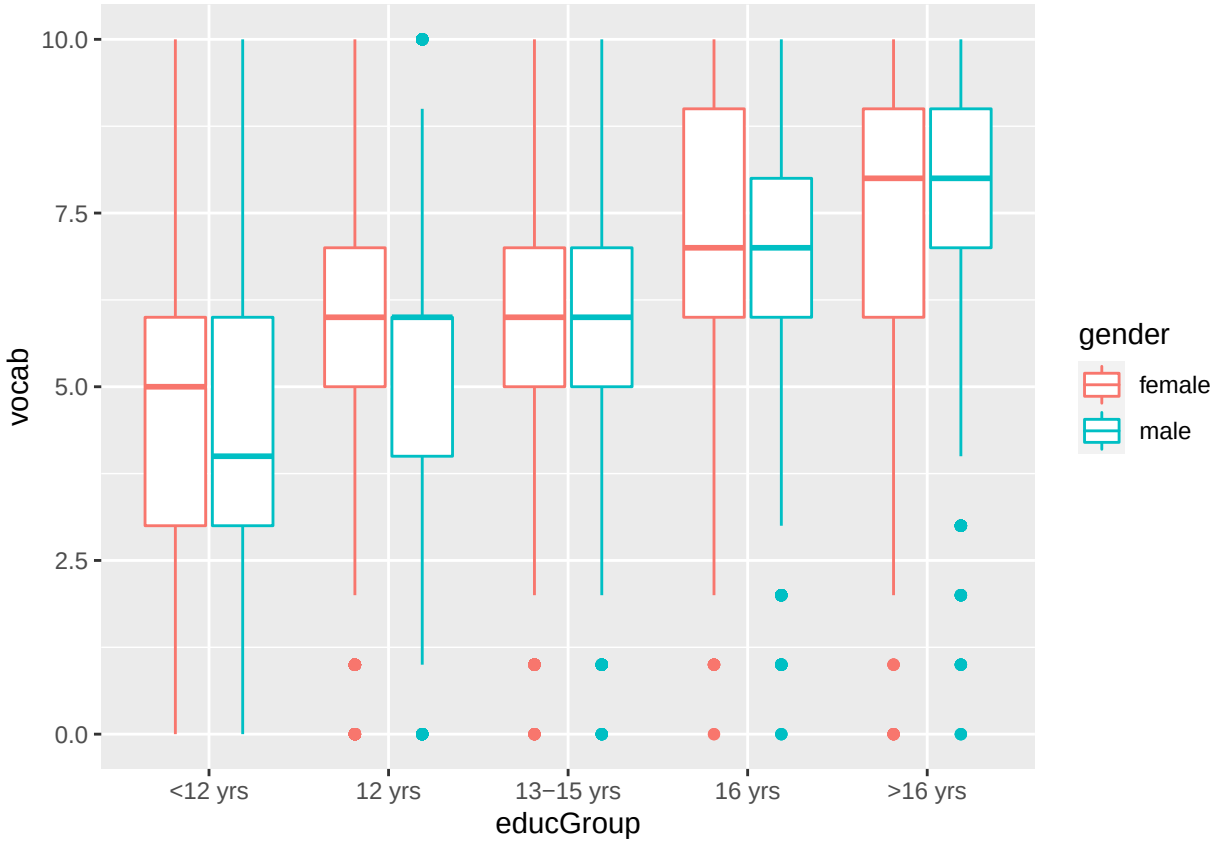

```
#The entire distribution is being shifted up. As you get older your vocab gets better.
#Another strategy is to use a density plot in this scenario.
ggplot(GSSvocab)+
  aes(x=vocab)+
  geom_density(aes(fill=educGroup),adjust=2,alpha=.5)
```




```
#The entire distribution is being shifted up. As you get older your vocab gets better until around 16 y
```

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`?

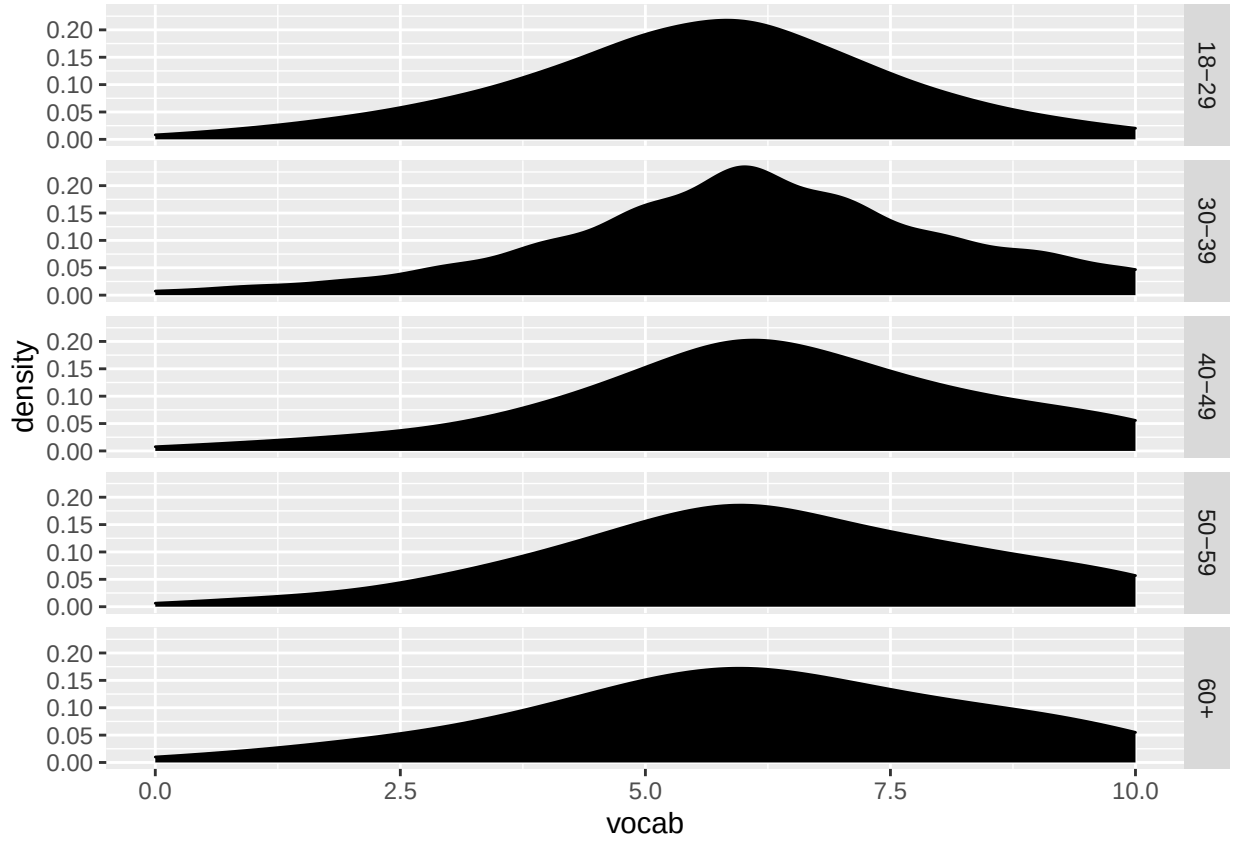
```
#Add gender to the box plot above to see if there is a distinction.
ggplot(GSSvocab)+
  aes(x=educGroup, y=vocab)+
  geom_boxplot(aes(col=gender))
```



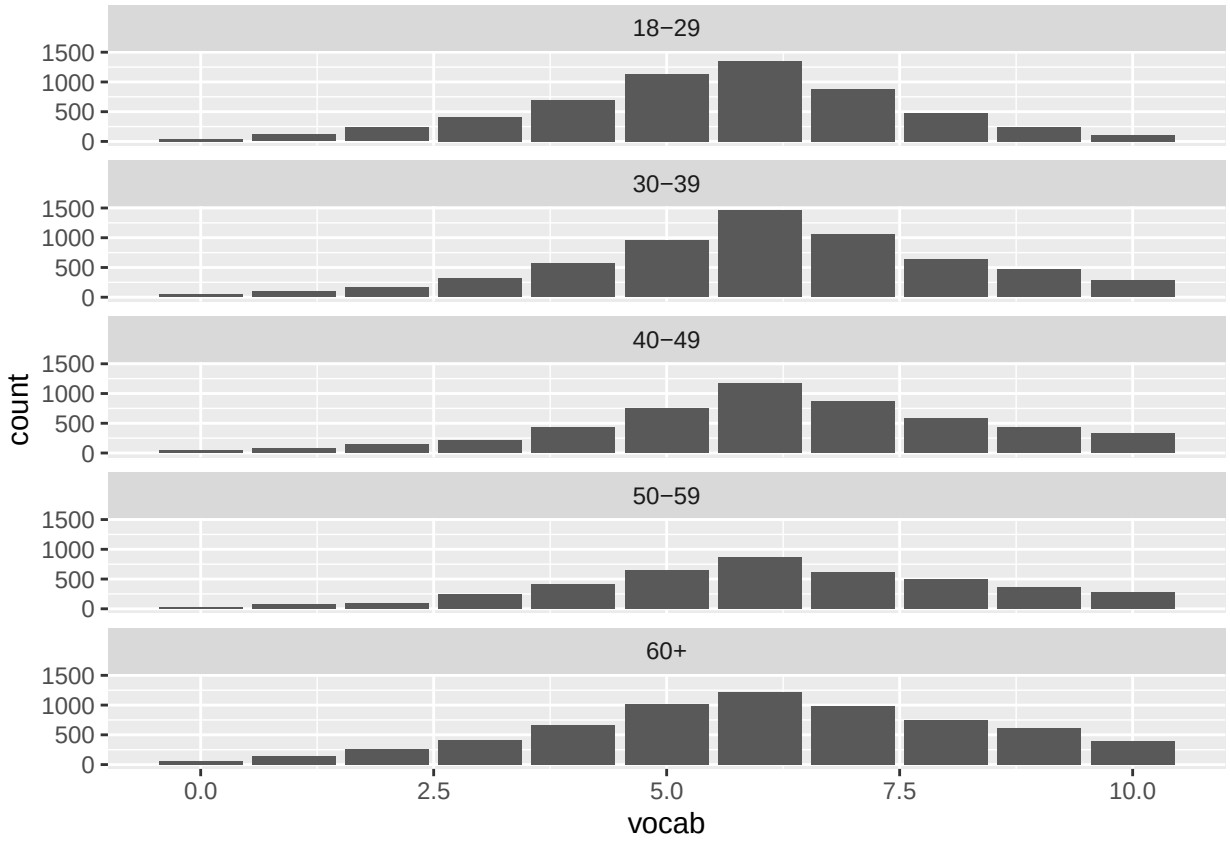

```
#Under 12 years of schooling females have a higher vocabulary and then it levels out between males and
```

Using facets, examine the relationship between `vocab` and `ageGroup`. Are we getting dumber?

```
#vocab is considered a categorical or continuous variable
#ageGroup is a categorical variable
#facets are a way to add variables or new plots
#Break up the density of vocabulary by age group.
ggplot(GSSvocab)+
  aes(x=vocab)+
  geom_density(adjust=2, fill="black")+
  facet_grid(ageGroup~.)
```




```
#We can also use a bar plot.
ggplot(GSSvocab)+
  aes(x=vocab)+
  geom_bar()+
  facet_wrap(~ageGroup, nrow=5)
```




```
#It does not seem that we are getting dumber over time.
```

#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 is a factor variable so we can cast it to numeric or you an if-else statement.
adult$income = ifelse(adult$income == ">50K",1,0)
table(adult$income)
```

```
##
##     0     1
## 22653  7508
```

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`.

```
#Part1
```

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

adult$marital_status=ifelse(adult$marital_status=="Married-AF-spouse"|adult$marital_status=="Married-ci

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

table(adult$marital_status)
```

```
##
##              Divorced               Married Married-spouse-absent
##                  4214                 14086                   370
##         Never-married             Separated               Widowed
##                  9725                   939                   827
```

```
#Part2 (Repeat part one with different variables)

adult$eductaion=as.character(adult$marital_status)

adult$education=ifelse(adult$education=="1st-4th"|adult$education=="Preschool", "<=4th", adult$educatio

adult$education=as.factor(adult$education)

table(adult$education)
```

```
##
## <=4th     1    10    11    12    13    15    16     2     3     5     6     7
##   196   820  5043   375  9840  1627   542  6678  1048   377   288   557   455
##     8     9
##  1008  1307
```

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] 100
```

```
#The categorical variables get dummified
Matrix::rankMatrix(Xmm)
```

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

```
#ncol Xmm is 100 and the rank is 94. This means it is not full rank.
```

Now tabulate and sort the variable `native_country`.

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

```
##
##         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.

Almost everyone is from the United States and there is only one Holland-Netherlands example. Yes, there are rare levels and we cannot model the difference between them. If we start cutting the data we will not see any of them and the computer won't know what to do.

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)
```

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

```
##
##           Columbia             Poland              Japan          Guatemala
##                 56                 56                 59                 63
##            Vietnam Dominican-Republic              China              Italy
##                 64                 67                 68                 68
##              South            Jamaica            England               Cuba
##                 71                 80                 86                 92
##        El-Salvador              India             Canada        Puerto-Rico
##                100                100                107                109
##            Germany        Philippines              Other             Mexico
##                128                188                486                610
##      United-States
##              27503
```

```
#Now there are no more countries with less than 50 and we have "other" with 486 that includes everythin
```

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?

```
#Compare income to workclass and occupation.
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
```

```
#ncol is 21 and the rank is 20. This is not full rank because there is duplication.
```

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` together with a ":" in between. Use the `paste` function with the `sep` argument (this casts automatically to type `character`). Then tabulate its levels and sort.

```
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tabulate = sort(table(adult$worktype))
adult = subset(adult, select = -c(occupation, workclass))
```

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$worktype = ifelse(adult$worktype %in% names(tabulate[tabulate<100]), "other", adult$worktype)
adult$worktype = as.factor(adult$worktype)
sort(table(adult$worktype))
```

```
##
##        Transport-moving:Local-gov         Protective-serv:State-gov
```

```
##                                115                                116
## Transport-moving:Self-emp-not-inc          Other-service:State-gov
##                                118                                123
##            Craft-repair:Local-gov          Priv-house-serv:Private
##                                143                                143
##       Prof-specialty:Self-emp-inc       Prof-specialty:Federal-gov
##                                157                                167
##    Other-service:Self-emp-not-inc      Exec-managerial:Federal-gov
##                                173                                179
##         Exec-managerial:State-gov          Protective-serv:Private
##                                186                                186
##           Other-service:Local-gov        Exec-managerial:Local-gov
##                                189                                212
##            Adm-clerical:State-gov           Adm-clerical:Local-gov
##                                250                                281
##               Sales:Self-emp-inc        Protective-serv:Local-gov
##                                281                                304
##          Adm-clerical:Federal-gov   Prof-specialty:Self-emp-not-inc
##                                316                                365
##            Sales:Self-emp-not-inc  Exec-managerial:Self-emp-not-inc
##                                376                                383
##      Exec-managerial:Self-emp-inc         Prof-specialty:State-gov
##                                385                                403
##  Farming-fishing:Self-emp-not-inc          Farming-fishing:Private
##                                430                                450
##     Craft-repair:Self-emp-not-inc         Prof-specialty:Local-gov
##                                523                                692
##              Tech-support:Private                              other
##                                723                               1008
##          Transport-moving:Private        Handlers-cleaners:Private
##                               1247                               1255
##         Machine-op-inspct:Private           Prof-specialty:Private
##                               1882                               2254
##           Exec-managerial:Private            Other-service:Private
##                               2647                               2665
##              Adm-clerical:Private                      Sales:Private
##                               2793                               2895
##              Craft-repair:Private
##                               3146
```

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

```
adult$relationship_status = paste(adult$relationship, adult$marital_status, sep = ":")
adult$relationship_status = as.factor(adult$relationship_status)
adult = subset(adult, select = -c(relationship, 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.

```
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.

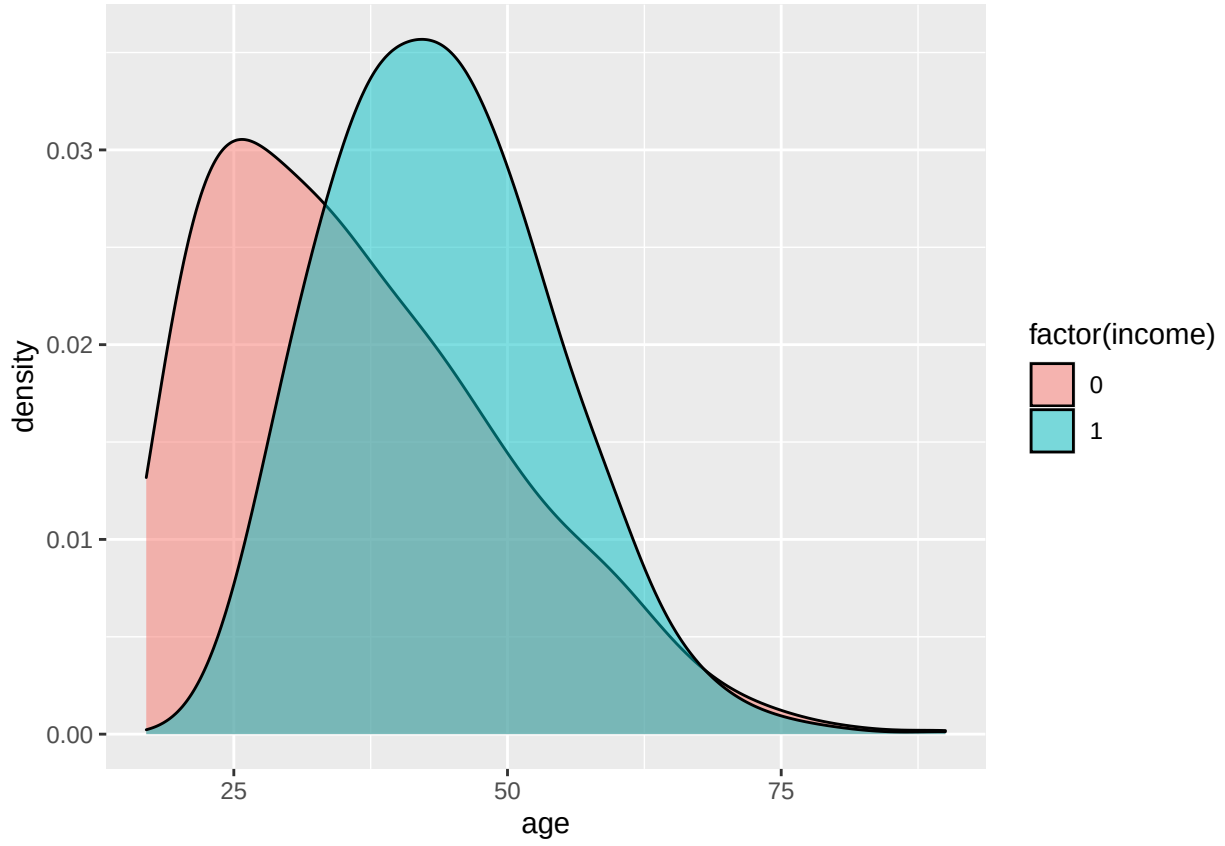
link_functions = c("logit", "probit", "cloglog", "cauchit") vanilla = income ~ . prob_est_mods = list() for(link_function in link_functions){ #prob_est_mods[[paste(link_function, "vanilla", sep = "-")]] = glm(vanilla, adult, family = binomial(link = link_function)) }

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.

```
adult$log_capital_loss = log(1 + adult$capital_loss)
adult$log_capital_gain = log(1 + adult$capital_gain)
```

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

```
ggplot(adult) +
  aes(x = age) +
  geom_density(aes(fill = factor(income)), adjust = 2, alpha = .5)
```



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

Yes this is expected.

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.

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 age_interactions = income ~ . *age for (link_function in link_functions){ prob_est_mods[[paste(link_function, "age_interactions", sep = "-")]]=glm(formula = age_interactions, data = adult_train, family = binomial(link=link_function))

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){
  p_hat=predict(prob_est_mod,X)
  mean(-(y-p_hat)^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.

```
#The logit wins because it accounts for the most relationships.
```

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

```
#There is no validation.
```

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.

K = 5 n = nrow(adult) test_indices = sample(1 : n, size = n * 1 / K) master_train_indices = setdiff(1 : n, test_indices) select_indices = sample(master_train_indices, size = n * 1 / K) subtrain_indices = setdiff(master_train_indices, select_indices) adult_subtrain = adult[subtrain_indices,] adult_select = adult[select_indices,] adult_test = adult[test_indices,] y_subtrain = adult_subtrain$income y_select = adult_select$income y_test = adult_test$income link_functions = c("logit", "probit", "cloglog", "cauchit") reran_mods = list() for (link_function in link_functions) { reran_mods[[paste(link_function, "vanilla", sep = "-")]] = glm(vanilla, adult_subtrain, family = binomial(link = link_function)) } for (link_function in link_functions) { reran_mods[[paste(link_function, "age_interactions", sep = "-")]] = glm(age_interactions, adult_subtrain, family = binomial(link = link_function)) } mods = lapply(reran_mods, brier_score, adult_select, y_select) which.max(mods) # logit vanilla is best best_model = glm(income ~ ., adult_train, family = binomial(link = "logit")) true_oos_brier_score = brier_score(best_model, adult_test, y_test) true_oos_brier_score