Lab 6

Jia Yu 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, 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
```

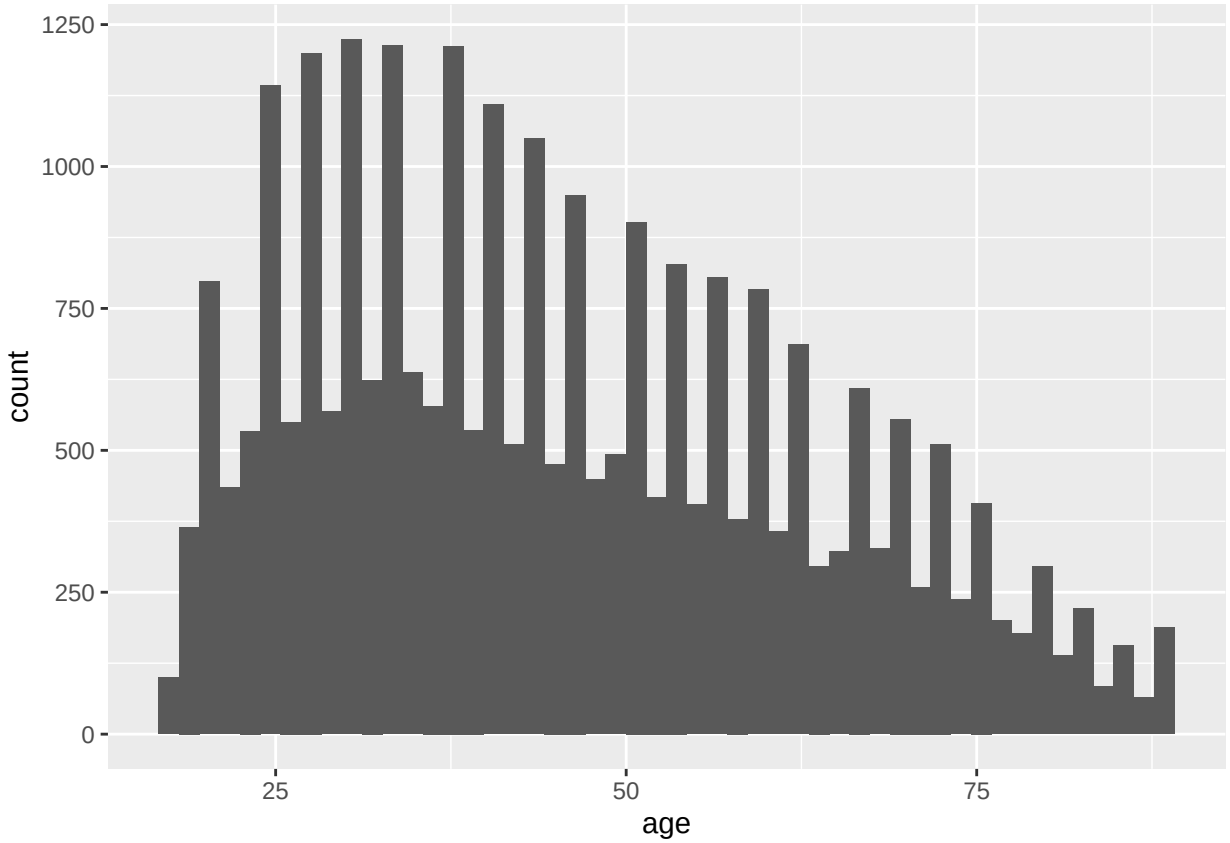
```
## starting httpd help server ... done
```

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?

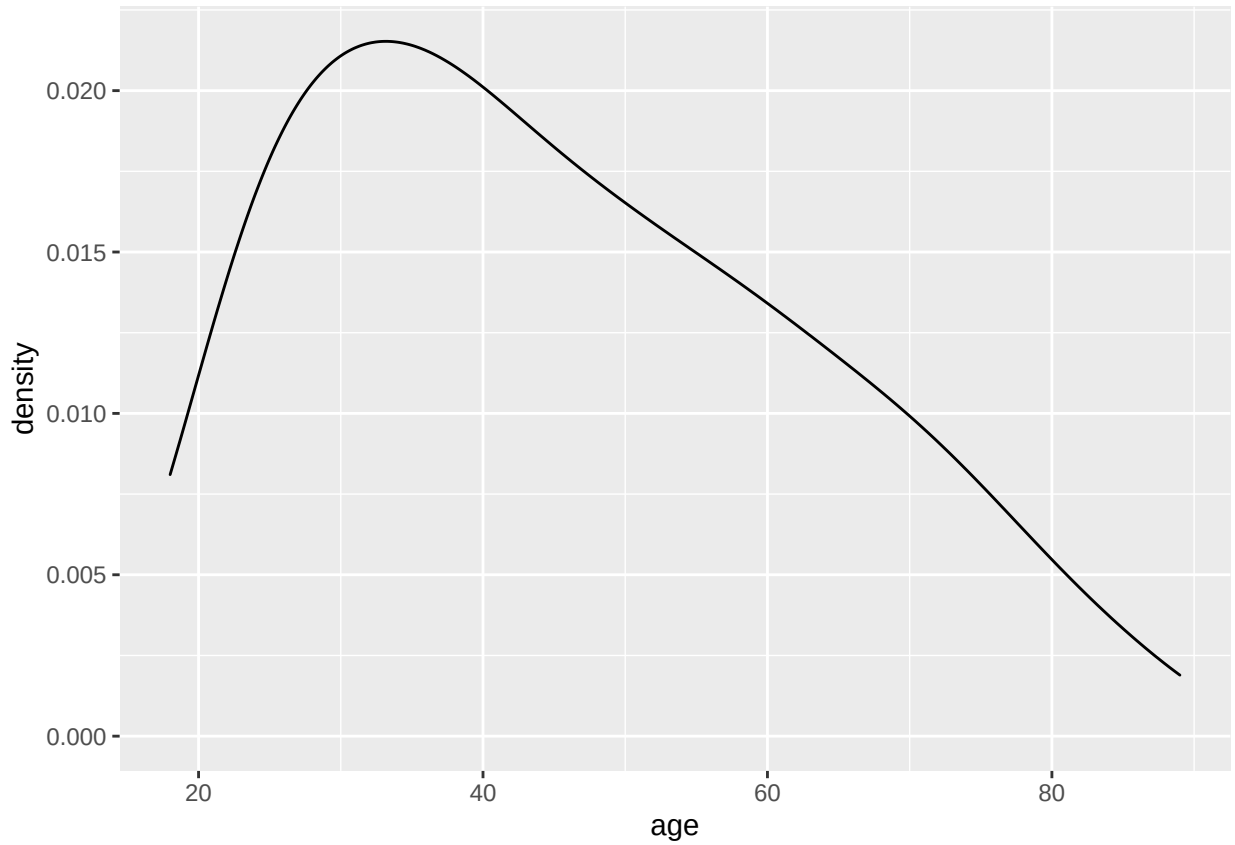
Year, gender, nativeBorn, ageGroup, and educGroup are a factor. Vocab, age, and educ are numeric. The response variable the collectors of this data had in mind is the vocab variable.

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)

#plot 1
ggplot(GSSvocab) +
  aes(x = age) +
  geom_histogram(bins = 50)
```

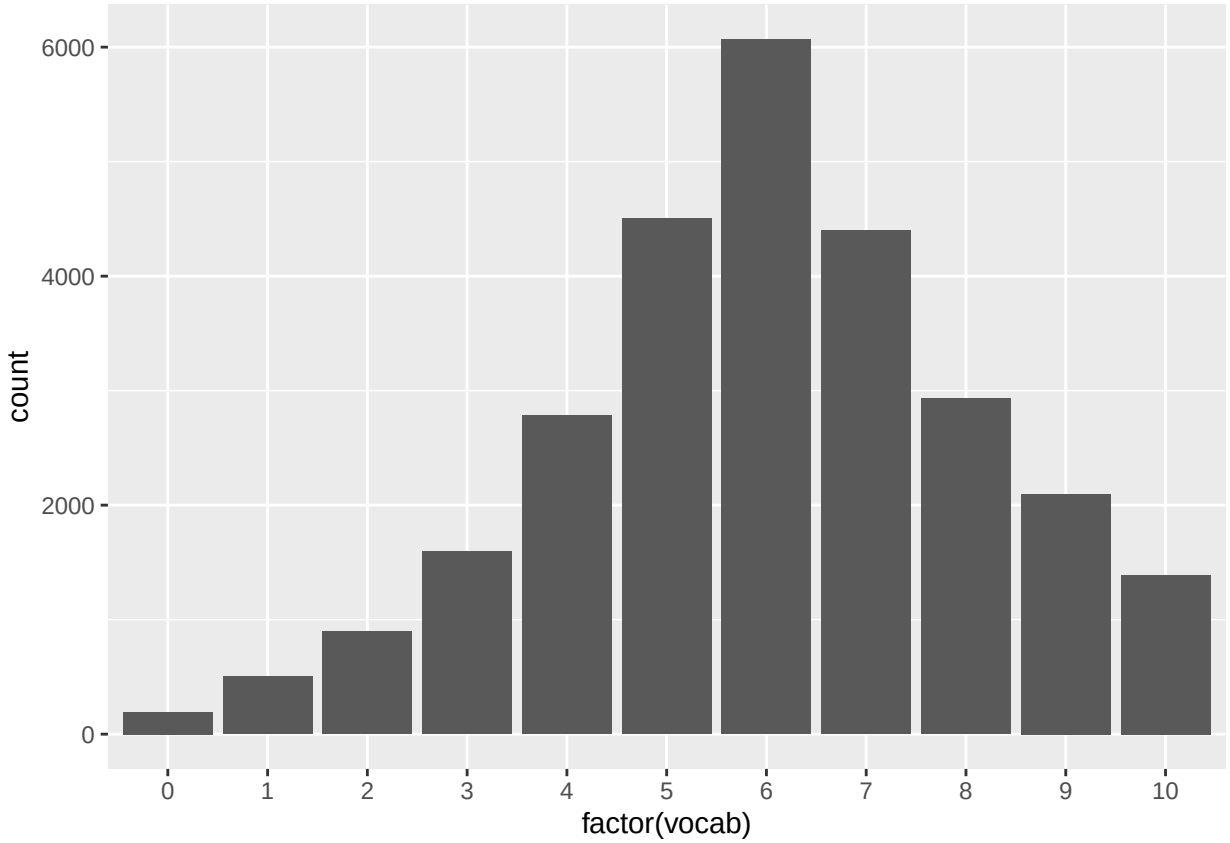



```
#plot 2 should be the better looking plot
ggplot(GSSvocab) +
  aes( x = age) +
  geom_density(adjust = 2.5)
```



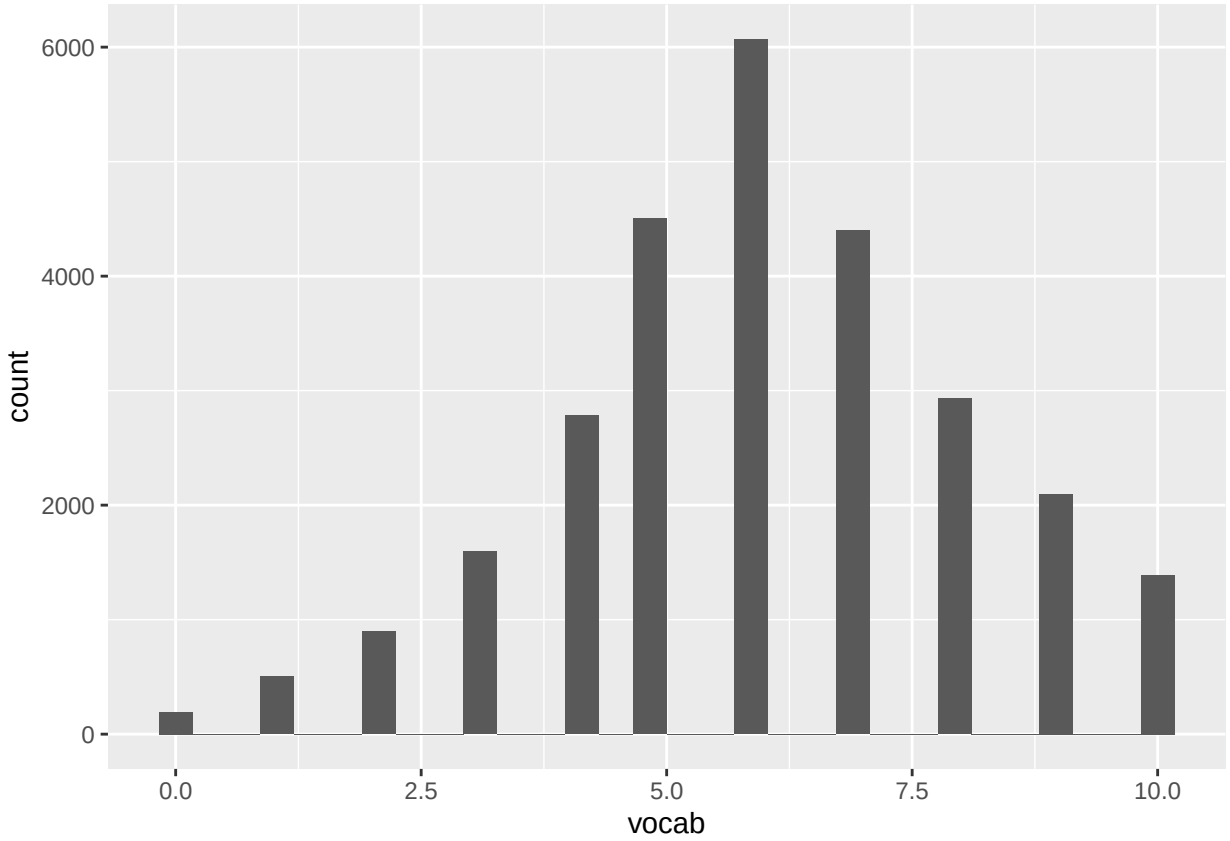
Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

```
#plot 1
ggplot(GSSvocab) +
  aes(x = factor(vocab)) +
  geom_bar()
```



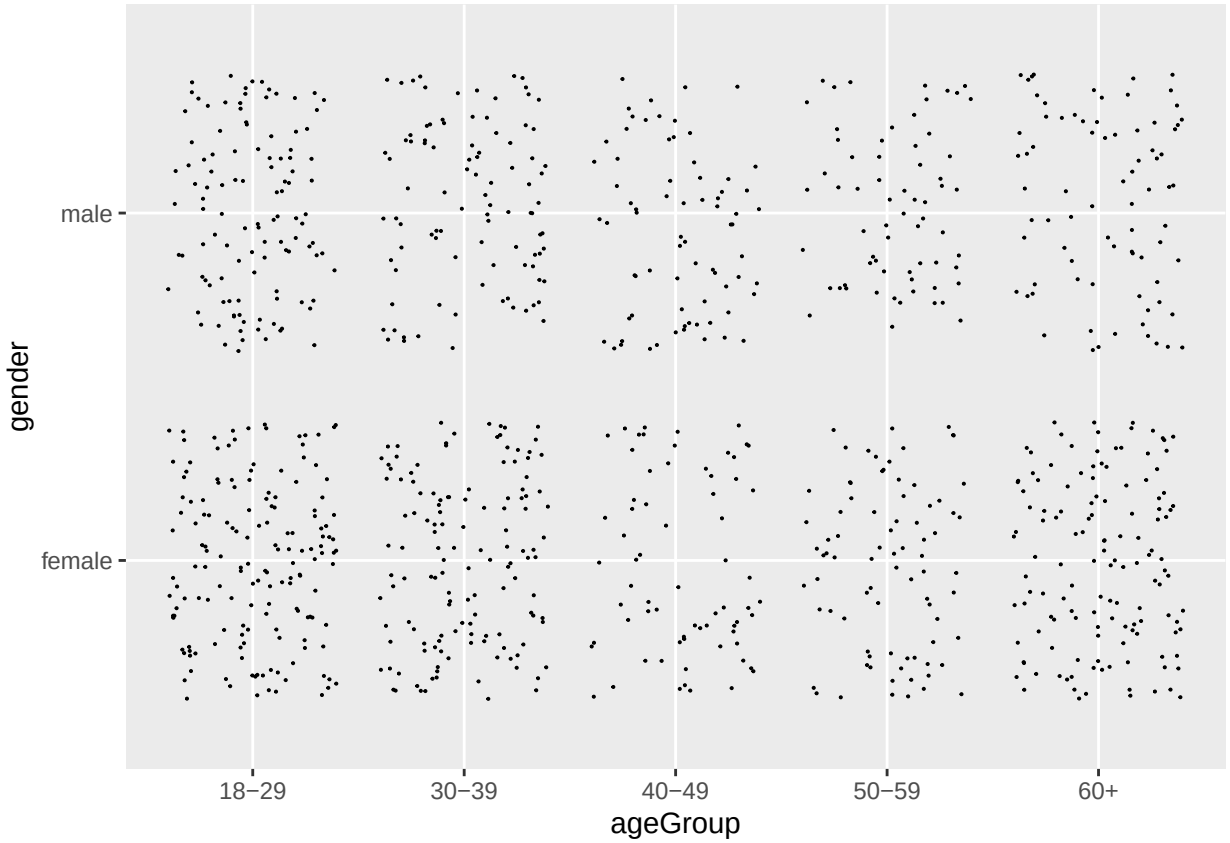

```
#plot 2 should be the better looking plot

ggplot(GSSvocab) +
  aes( x = vocab) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



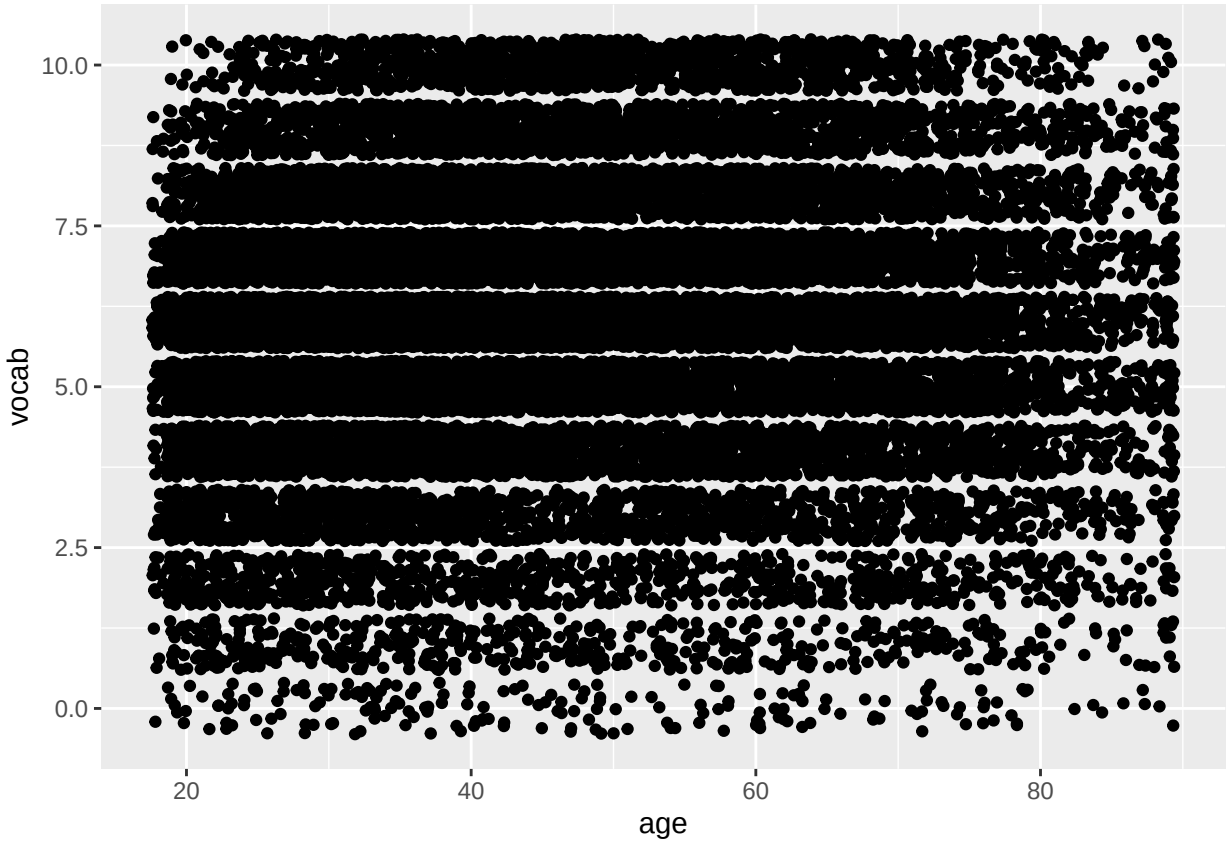
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```
ggplot(GSSvocab[1:1000, ]) +
  aes(x = ageGroup, y = gender) +
  geom_jitter(size = .05)
```



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

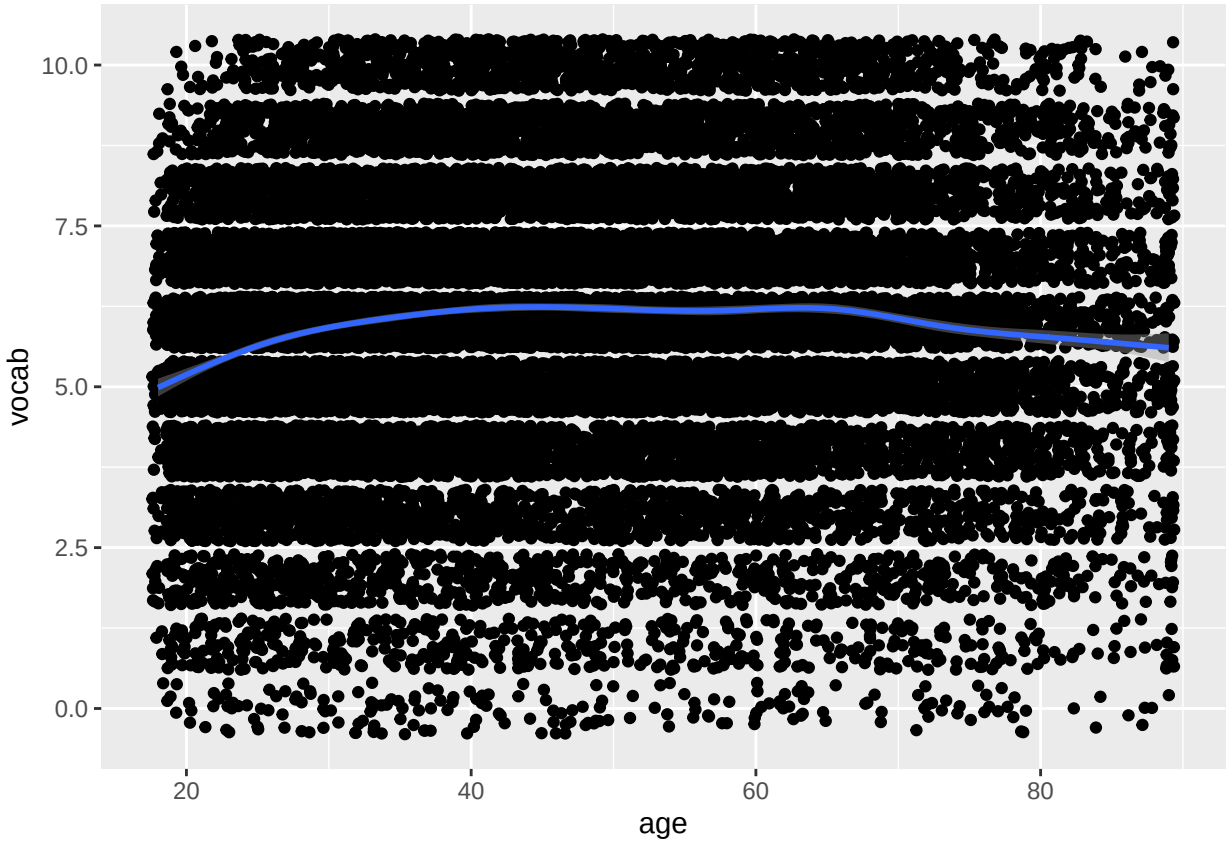
```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter()
```



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()
```

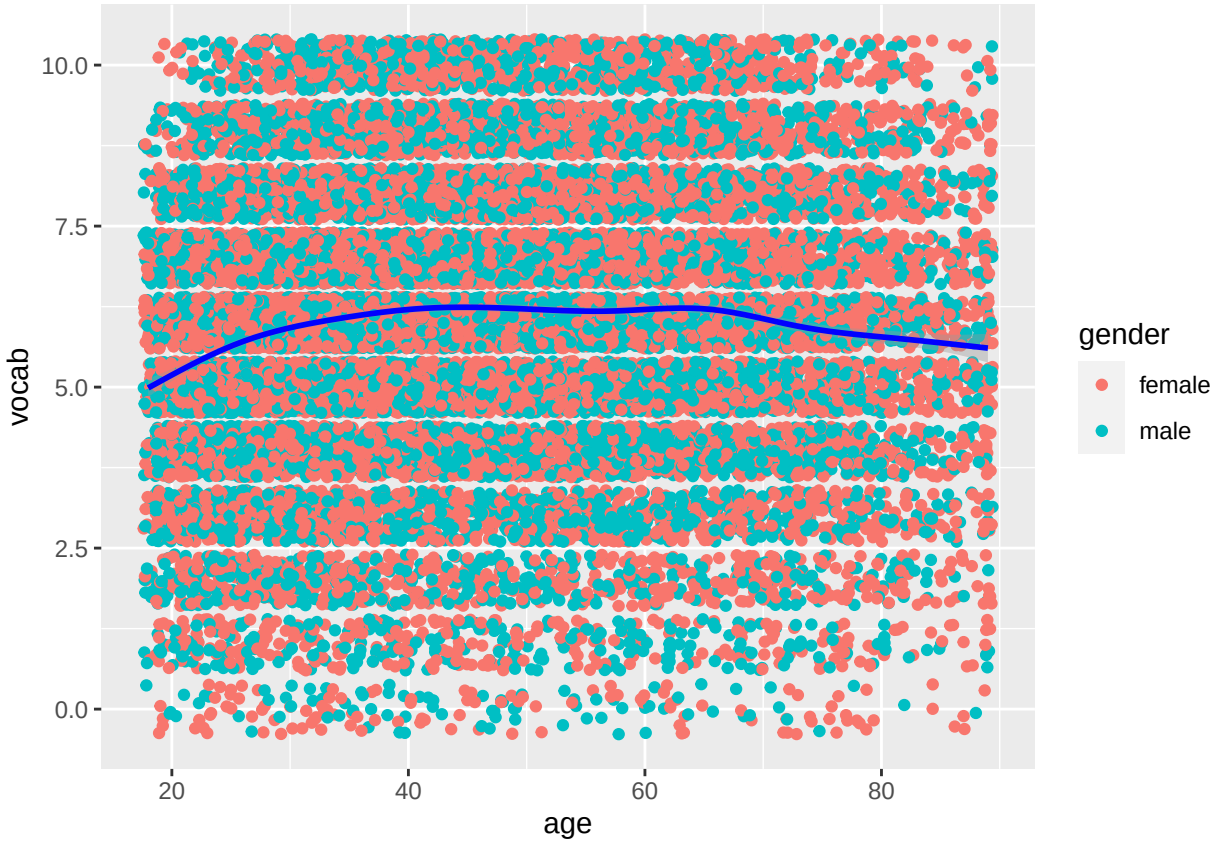
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



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")'
```

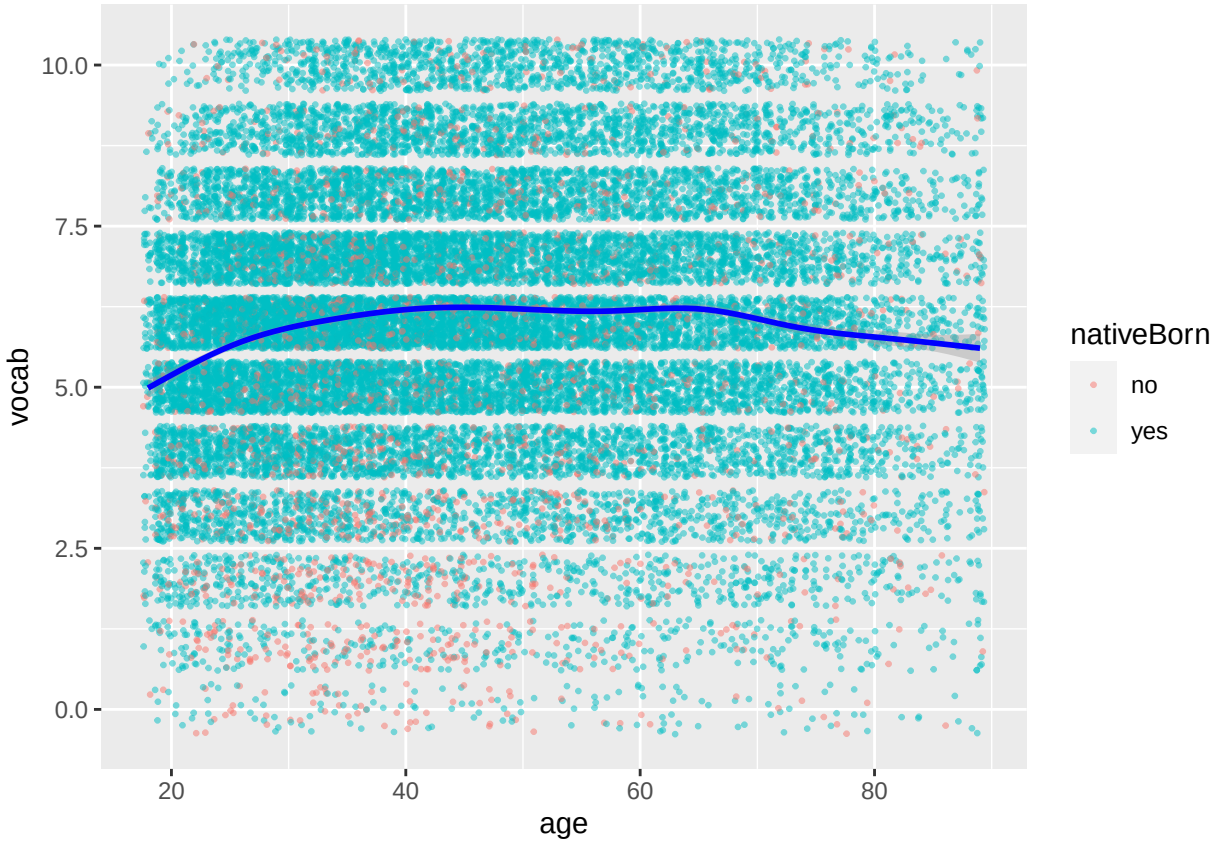



```
#not obvious that there is an interaction
```

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), size = .5, alpha = 0.5) +
  geom_smooth(col = "blue")
```

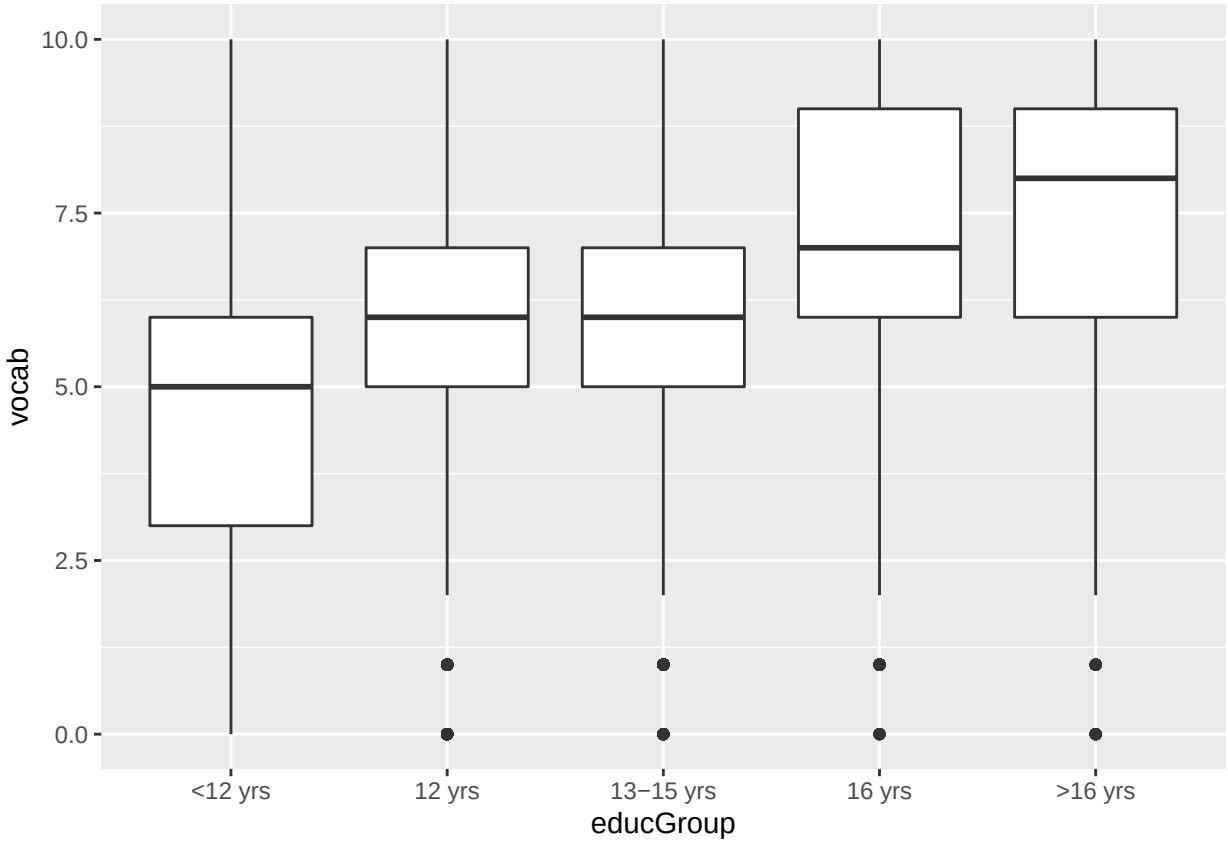
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



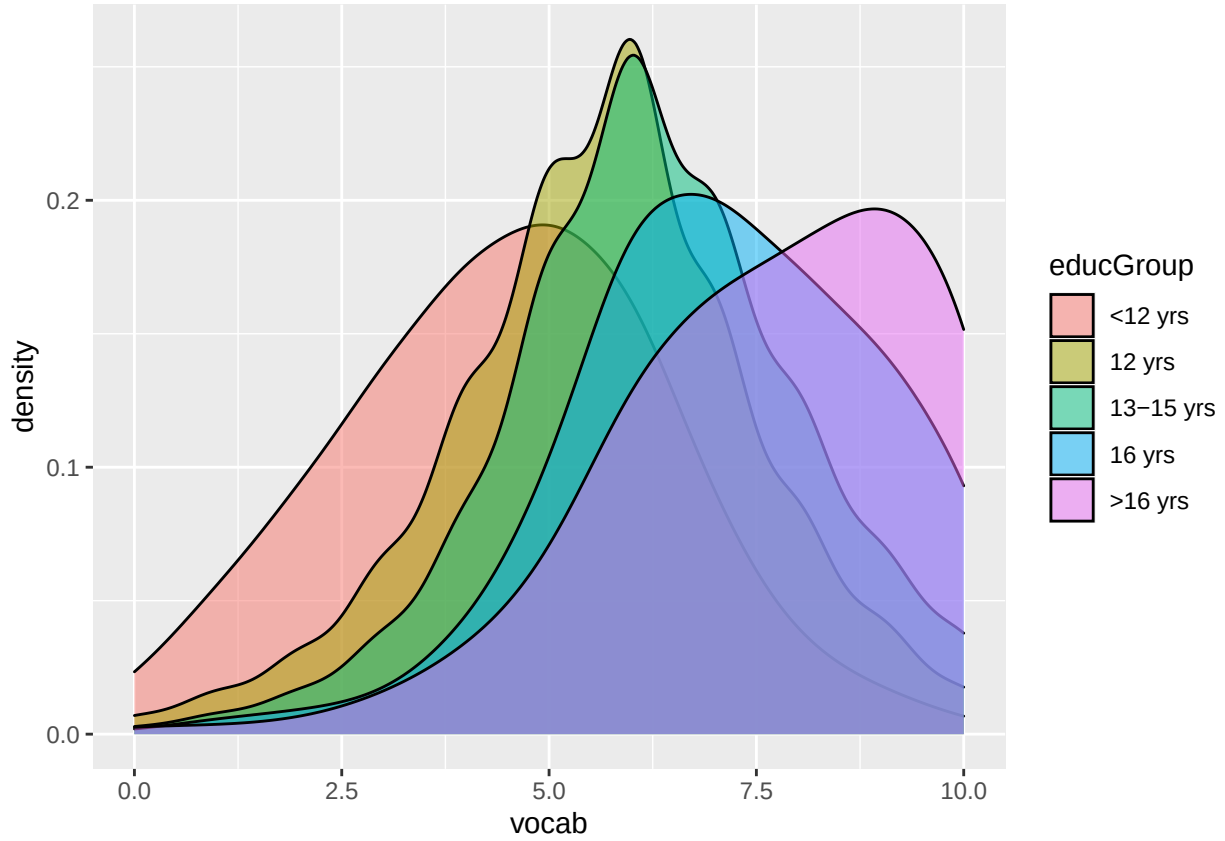

```
#Red dots density is lower down in the y access than the blue dots, therefore, native born people tend
```

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?

```
#plot 1
ggplot(GSSvocab) +
  aes(x = educGroup, y = vocab) +
  geom_boxplot()
```

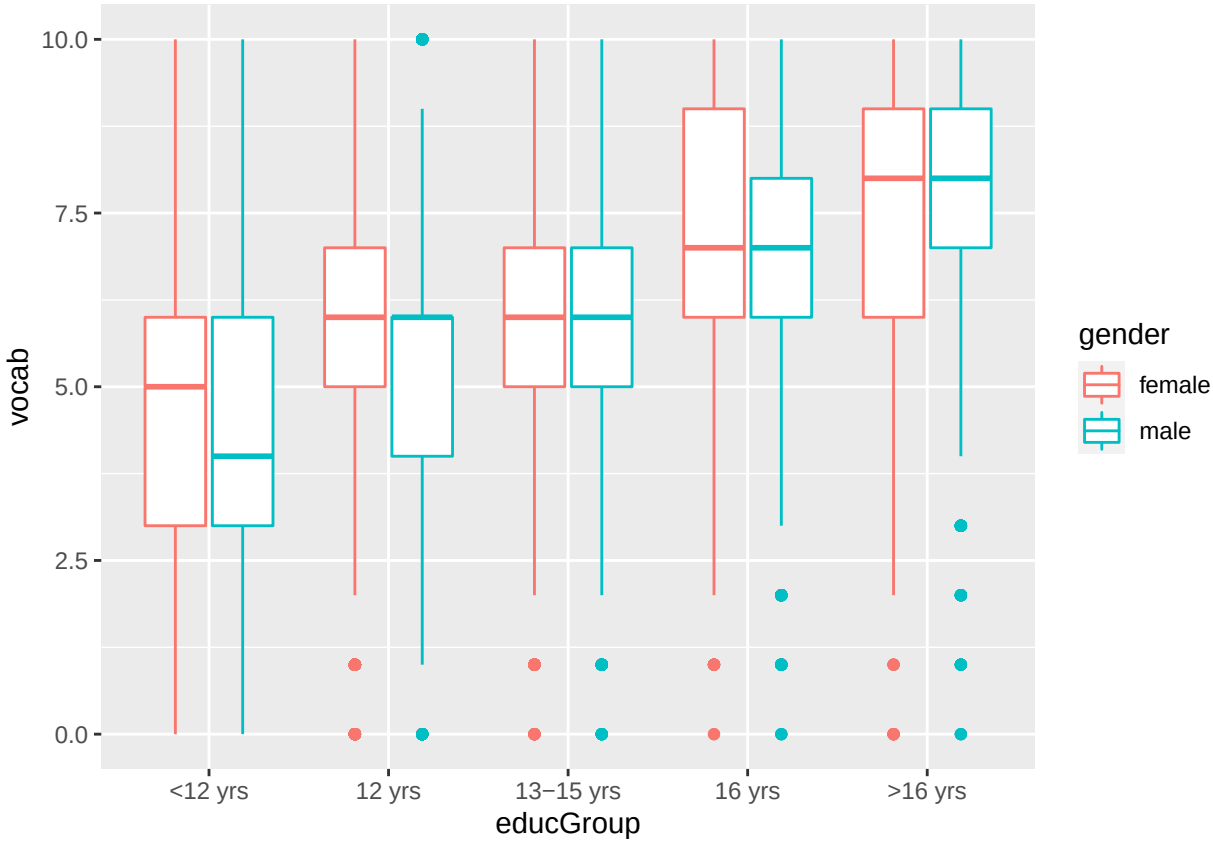



```
#With education, the middle 50 percentile is gaining vocabulary. The entire distribution is shifted up

#plot 2
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`?

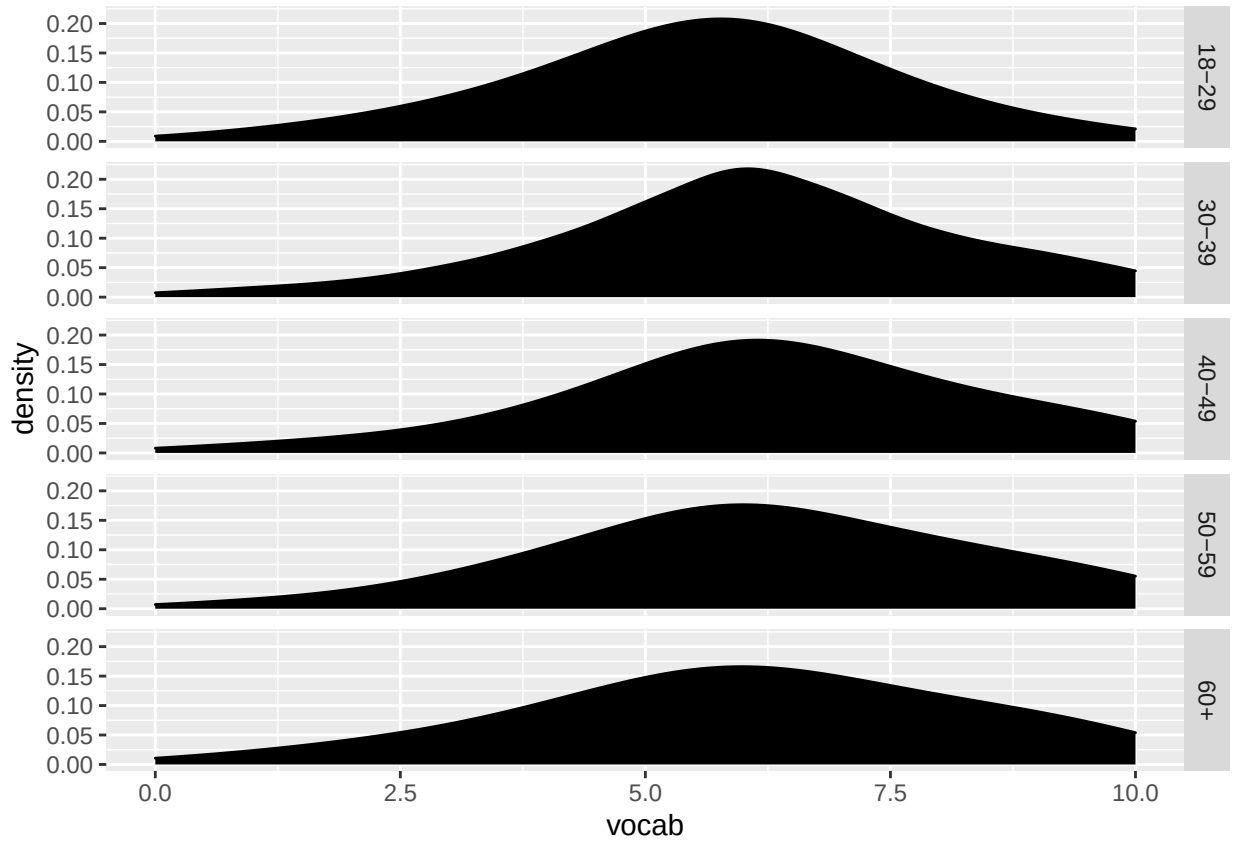
```
ggplot(GSSvocab) +
  aes(x = educGroup, y = vocab) +
  geom_boxplot(aes(col = gender))
```




```
#As female have higher vocabulary score up until highschool (13-15 years). In highschool, the male and
```

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 = 2.5, fill = "black") +
  facet_grid(ageGroup~.)
```




```
#No
```

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)

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`.

```
table(adult$marital_status)
```

```
##
##              Divorced     Married-AF-spouse    Married-civ-spouse
##                  4214                   21                 14065
## Married-spouse-absent         Never-married              Separated
##                   370                  9725                   939
##               Widowed
##                   827
```

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

```
table(adult$education)
```

```
##
##         10th         11th         12th      1st-4th      5th-6th      7th-8th
##          820         1048          377          151          288          557
##          9th   Assoc-acdm    Assoc-voc    Bachelors    Doctorate      HS-grad
##          455         1008         1307         5043          375         9840
##      Masters    Preschool  Prof-school Some-college
##         1627           45          542         6678
```

```
adult$education = as.character(adult$education)
adult$education = ifelse(adult$education == "1st-4th" | adult$education ==  "Preschool", "<=4th", adult$
adult$education = as.factor(adult$education)
table(adult$education)
```

```
##
##        <=4th         10th         11th         12th      5th-6th      7th-8th
##          196          820         1048          377          288          557
##          9th   Assoc-acdm    Assoc-voc    Bachelors    Doctorate      HS-grad
##          455         1008         1307         5043          375         9840
##      Masters  Prof-school Some-college
##         1627          542         6678
```

Create a model matrix `Xmm` (for this prediction task) 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] 95
```

```
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`.

```
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.

Yes. It may be a problem because there's not enough data points for some countries.

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$native_cou
adult$native_country = as.factor(adult$native_country)

table(adult$native_country)
```

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

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

```
## [1] 21
```

```
Matrix::rankMatrix(Xmm_work)
```

```
## [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.

```
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
adult$workclass <- NULL
adult$occupation <- NULL
tabulate = sort(table(adult$worktype))
tabulate
```

```
##
##           Craft-repair:Without-pay      Handlers-cleaners:Without-pay
##                                  1                                  1
##      Machine-op-inspct:Without-pay          Other-service:Without-pay
##                                  1                                  1
##       Transport-moving:Without-pay     Handlers-cleaners:Self-emp-inc
##                                  1                                  2
##           Adm-clerical:Without-pay          Tech-support:Self-emp-inc
##                                  3                                  3
##       Protective-serv:Self-emp-inc        Farming-fishing:Without-pay
##                                  5                                  6
##   Protective-serv:Self-emp-not-inc                   Sales:Local-gov
##                                  6                                  7
##        Farming-fishing:Federal-gov           Armed-Forces:Federal-gov
##                                  8                                  9
##        Handlers-cleaners:State-gov     Machine-op-inspct:Self-emp-inc
##                                  9                                 10
##        Machine-op-inspct:Local-gov                 Sales:State-gov
##                                 11                                 11
##        Machine-op-inspct:State-gov      Machine-op-inspct:Federal-gov
##                                 13                                 14
##                  Sales:Federal-gov          Farming-fishing:State-gov
##                                 14                                 15
## Handlers-cleaners:Self-emp-not-inc      Handlers-cleaners:Federal-gov
##                                 15                                 22
##       Transport-moving:Federal-gov      Tech-support:Self-emp-not-inc
##                                 24                                 26
##      Transport-moving:Self-emp-inc         Other-service:Self-emp-inc
##                                 26                                 27
##        Protective-serv:Federal-gov          Adm-clerical:Self-emp-inc
##                                 27                                 28
```

```
##          Farming-fishing:Local-gov          Other-service:Federal-gov
##                                 29                                 34
## Machine-op-inspct:Self-emp-not-inc            Tech-support:Local-gov
##                                 35                                 38
##         Transport-moving:State-gov       Handlers-cleaners:Local-gov
##                                 41                                 46
##      Adm-clerical:Self-emp-not-inc       Farming-fishing:Self-emp-inc
##                                 49                                 51
##            Craft-repair:State-gov            Tech-support:State-gov
##                                 55                                 56
##           Craft-repair:Federal-gov          Tech-support:Federal-gov
##                                 63                                 66
##          Craft-repair:Self-emp-inc        Transport-moving:Local-gov
##                                 99                                115
##          Protective-serv:State-gov  Transport-moving:Self-emp-not-inc
##                                116                                118
##            Other-service:State-gov            Craft-repair:Local-gov
##                                123                                143
##            Priv-house-serv:Private       Prof-specialty:Self-emp-inc
##                                143                                157
##         Prof-specialty:Federal-gov    Other-service:Self-emp-not-inc
##                                167                                173
##        Exec-managerial:Federal-gov         Exec-managerial:State-gov
##                                179                                186
##            Protective-serv:Private           Other-service:Local-gov
##                                186                                189
##          Exec-managerial:Local-gov            Adm-clerical:State-gov
##                                212                                250
##             Adm-clerical:Local-gov                Sales:Self-emp-inc
##                                281                                281
##          Protective-serv:Local-gov           Adm-clerical:Federal-gov
##                                304                                316
##    Prof-specialty:Self-emp-not-inc            Sales:Self-emp-not-inc
##                                365                                376
##   Exec-managerial:Self-emp-not-inc       Exec-managerial:Self-emp-inc
##                                383                                385
##           Prof-specialty:State-gov   Farming-fishing:Self-emp-not-inc
##                                403                                430
##            Farming-fishing:Private     Craft-repair:Self-emp-not-inc
##                                450                                523
##           Prof-specialty:Local-gov              Tech-support:Private
##                                692                                723
##           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
```

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$worktype = as.character(adult$worktype)

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 <- NULL
adult$marital_status <- NULL
tabulate = sort(table(adult$relationship_status))
tabulate
```

```
##
##                     Own-child:Widowed                Not-in-family:married
##                                    12                                   14
## Other-relative:Married-spouse-absent               Other-relative:Widowed
##                                    26                                   40
##      Own-child:Married-spouse-absent             Other-relative:Separated
##                                    43                                   53
##                    Own-child:married                 Own-child:Separated
##                                    84                                   90
##              Other-relative:Divorced                Other-relative:married
##                                   103                                  119
##      Unmarried:Married-spouse-absent  Not-in-family:Married-spouse-absent
##                                   120                                  181
##                   Own-child:Divorced                   Unmarried:Widowed
##                                   308                                  343
##              Not-in-family:Separated                 Unmarried:Separated
##                                   383                                  413
##                Not-in-family:Widowed         Other-relative:Never-married
##                                   432                                  548
##              Unmarried:Never-married                          Wife:married
##                                   801                                 1406
##                   Unmarried:Divorced               Not-in-family:Divorced
##                                  1535                                 2268
##              Own-child:Never-married          Not-in-family:Never-married
##                                  3929                                 4447
##                     Husband:married
##                                 12463
```

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 = binom
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

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.

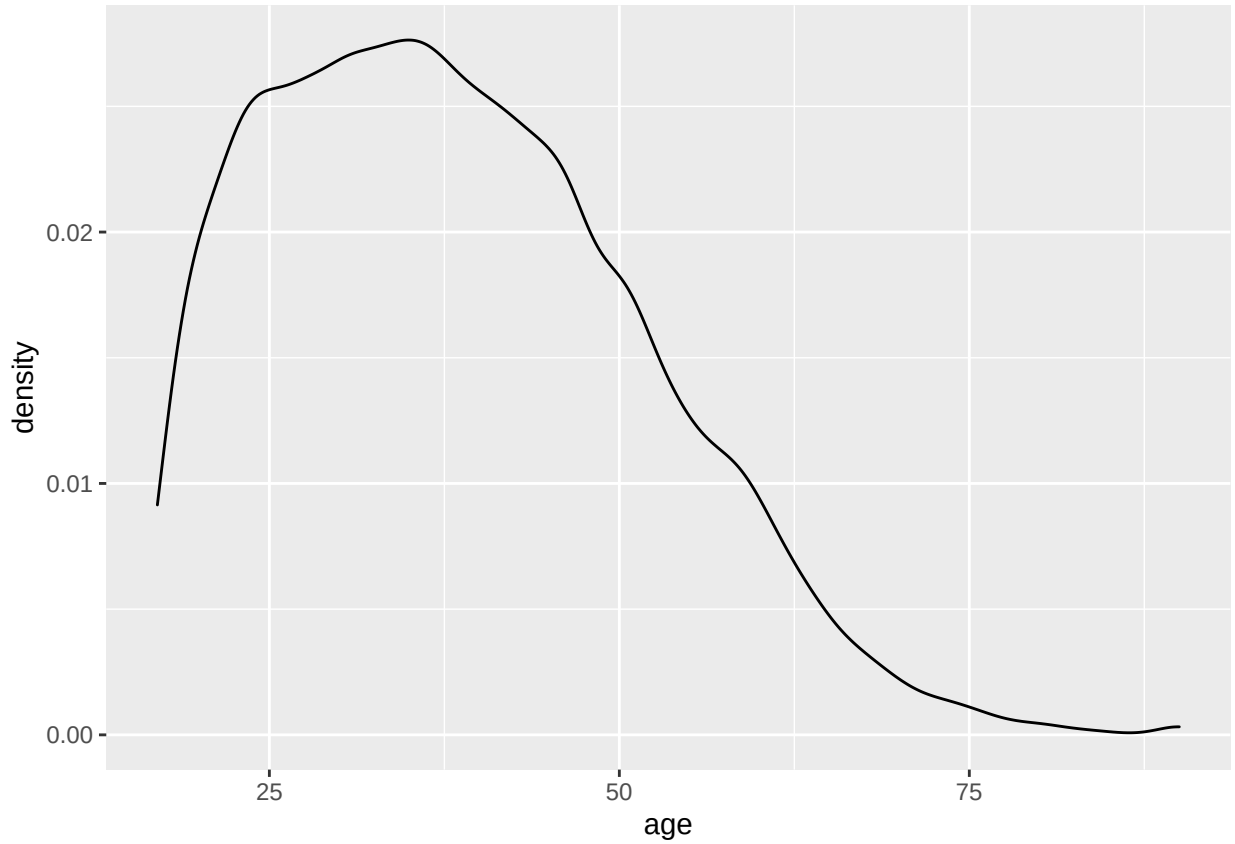
```
adult$log_capital_loss = log(adult$capital_loss + 1)
adult$log_capital_gain = log(adult$capital_gain + 1)
```

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

```
head(adult)
```

```
##   age education education_num  race    sex capital_gain capital_loss
## 1  50 Bachelors            13 White   Male            0            0
## 2  38   HS-grad             9 White   Male            0            0
## 3  53      11th             7 Black   Male            0            0
## 4  28 Bachelors            13 Black Female            0            0
## 5  37   Masters            14 White Female            0            0
## 6  49       9th             5 Black Female            0            0
##   hours_per_week native_country income                        worktype
## 1             13  United-States      0 Exec-managerial:Self-emp-not-inc
## 2             40  United-States      0        Handlers-cleaners:Private
## 3             40  United-States      0        Handlers-cleaners:Private
## 4             40           Cuba      0           Prof-specialty:Private
## 5             40  United-States      0          Exec-managerial:Private
## 6             16        Jamaica      0            Other-service:Private
##                   relationship_status log_capital_loss log_capital_gain
## 1                   Husband:married                0                0
## 2              Not-in-family:Divorced                0                0
## 3                   Husband:married                0                0
## 4                      Wife:married                0                0
## 5                      Wife:married                0                0
## 6 Not-in-family:Married-spouse-absent                0                0
```

```
ggplot(adult) +
  aes( x = age) +
  geom_density( y = "income")
```



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

Yes because when you are around 50-75, that is usually retirement age. Before 25, you are usually a student or child so chances are that you won't be working. At 25-50, most people start working.

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 = income ~ age*.

for (link_function in link_functions) {
  prob_est_mods[[paste(link_function,"age_interactions", sep = "-")]] = glm(age_interactions, adult_tra
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

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

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, type="response")
  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)
```

```
## $`logit-vanilla`
## [1] -0.1031617
##
## $`probit-vanilla`
## [1] -0.1031872
##
## $`cloglog-vanilla`
## [1] -0.1041009
##
## $`cauchit-vanilla`
## [1] -0.104773
##
## $`logit-age_interactions`
## [1] -0.1016513
##
## $`probit-age_interactions`
## [1] -0.1017761
##
## $`cloglog-age_interactions`
## [1] -0.2363546
##
## $`cauchit-age_interactions`
## [1] -0.1028667
```

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)
```

```
## $`logit-vanilla`
## [1] -0.1035148
##
## $`probit-vanilla`
## [1] -0.1035982
##
## $`cloglog-vanilla`
## [1] -0.1044773
```

```
##
## $`cauchit-vanilla`
## [1] -0.1048273
##
## $`logit-age_interactions`
## [1] -0.1029698
##
## $`probit-age_interactions`
## [1] -0.1029807
##
## $`cloglog-age_interactions`
## [1] -0.2398873
##
## $`cauchit-age_interactions`
## [1] -0.1051165
```

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

Logit-vanilla model wins for being the least negative.

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

We didn't do K-fold cross validation or nested resampling.

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
```

```
#Now, fit all models and select the best one:
mods = list()
for (link_function in link_functions) {
  mods[[paste(link_function,"vanilla", sep = "-")]] = glm(vanilla, adult_subtrain, family = binomial(li
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

```
for (link_function in link_functions) {
  mods[[paste(link_function,"age_interactions", sep = "-")]] = glm(age_interactions, adult_subtrain, fa
}
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

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

```
score = lapply(mods, brier_score, adult_select, y_select)
which_final = which.max(score)
score
```

```
## $`logit-vanilla`
## [1] -0.09977037
##
## $`probit-vanilla`
## [1] -0.1853448
##
## $`cloglog-vanilla`
## [1] -0.1008164
##
## $`cauchit-vanilla`
## [1] -0.1019644
##
## $`logit-age_interactions`
## [1] -0.1008693
##
## $`probit-age_interactions`
## [1] -0.1012003
##
## $`cloglog-age_interactions`
## [1] -0.1015431
##
## $`cauchit-age_interactions`
## [1] -0.1037994
```

```
which_final
```

```
## logit-vanilla
##             1
```

```
g_final = glm(income~., adult_train, family = binomial(link = "logit"))
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
true_oos_brier = brier_score(g_final, adult_test, y_test)
true_oos_brier
```

```
## [1] -0.1018789
```