Lab 6

Enoch Kim

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)
```

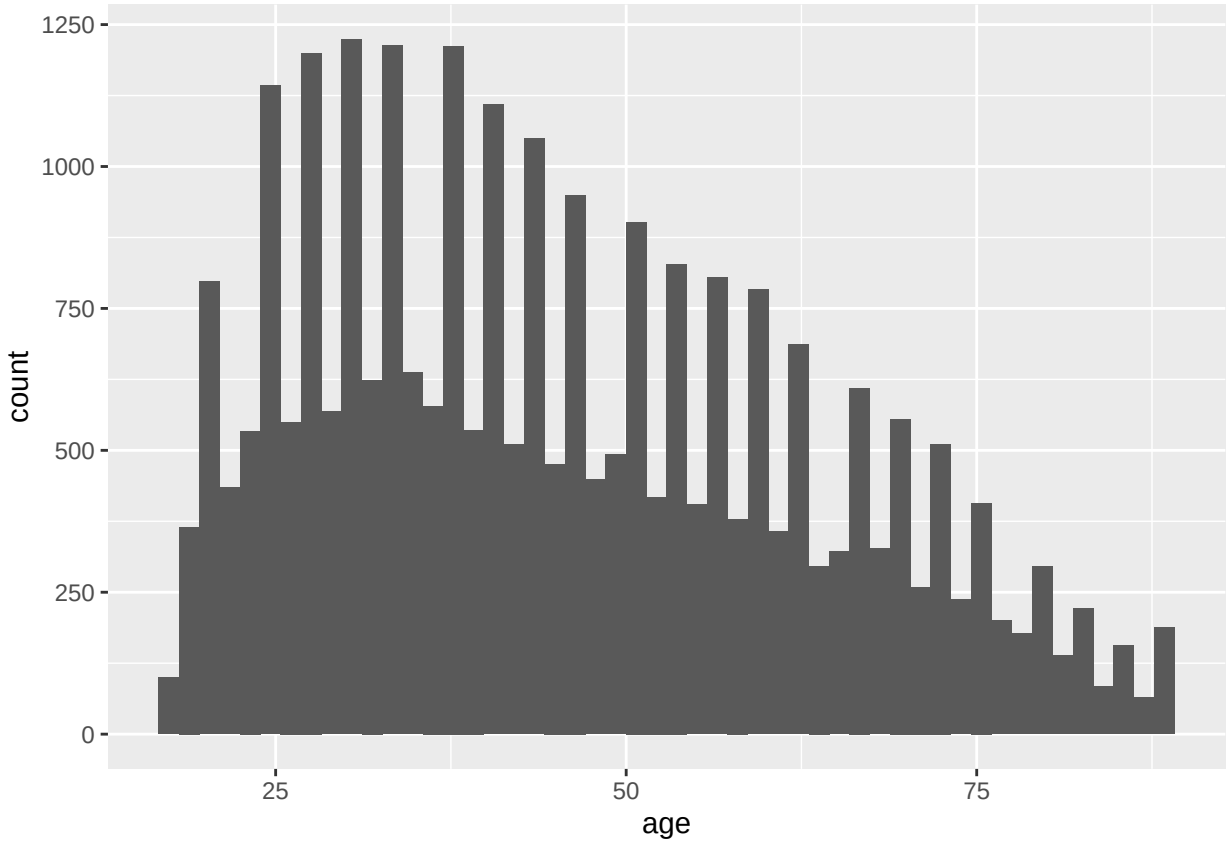
Briefly summarize the documentation on this dataset. What is the data type of each variable? What do you think is the response variable the collectors of this data had in mind?

GSSvocab has 28867 observations with 8 variables. These 8 variables are

1. year, the year the GSS was taken by the respondent, this is a continuous variable
2. gender, the respondent is either female or male, this is a factor/categorical variable
3. nativeborn, was the respondent born in the US, this is a factor/categorical variable
4. ageGroup, the group of age of the respondent, this is a factor/categorical variable
5. educGroup, the respondent education group, this is a factor/categorical variable
6. vocab, the amount corrected answers out of 10 on the test, this is a continuous variable
7. age, the respondents age in years, this is a continuous variable
8. educ, years of education of the respondent, this is a continuous 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.

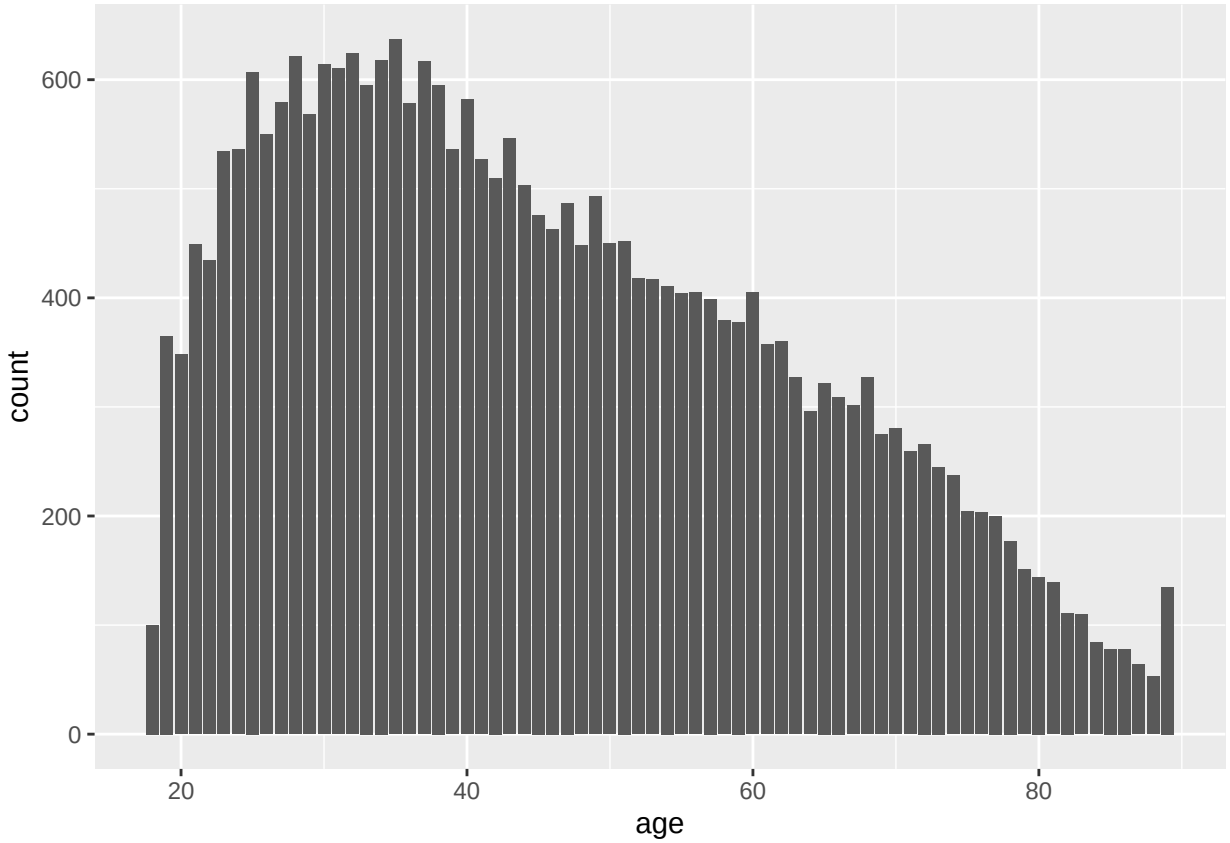
```
pacman::p_load(ggplot2)

agePlot = ggplot(GSSvocab) +
  aes(x = age) +
  geom_histogram(bins = 50)

agePlot
```




```
ggsave("agePlot.pdf")
```

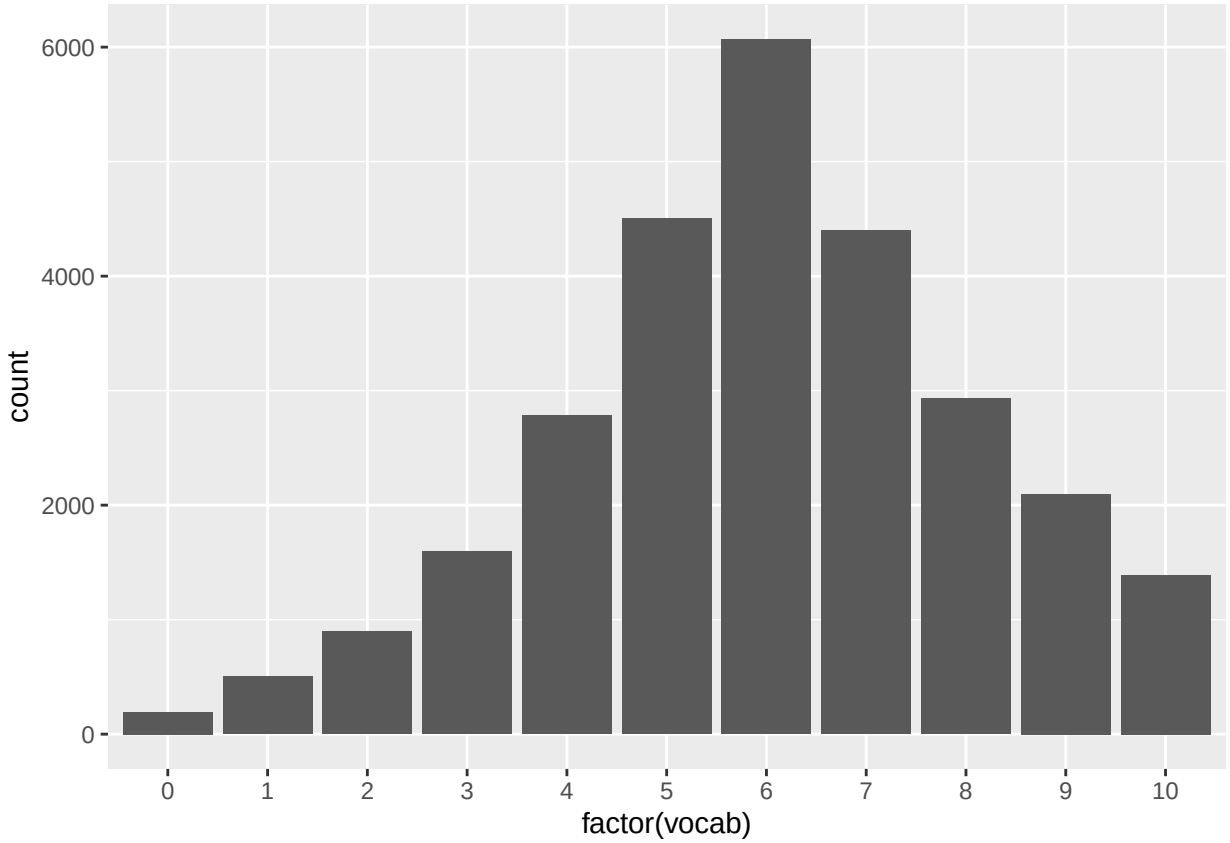
```
## Saving 6.5 x 4.5 in image
```

```
ggplot(GSSvocab) +
  aes(x = age) +
  geom_bar()
```



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.

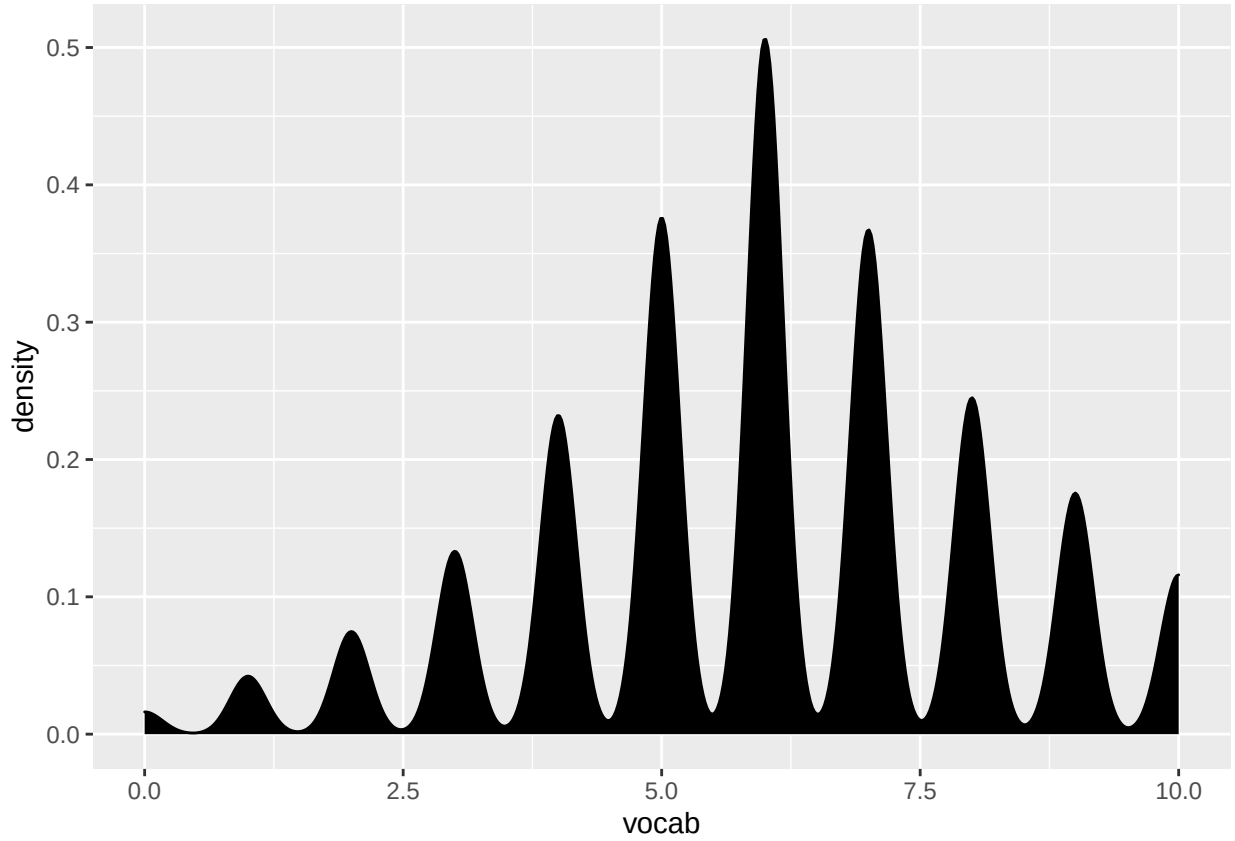
```
vocabPlot = ggplot(GSSvocab) +
  aes(x = factor(vocab)) +
  geom_bar()

vocabPlot
```




```
ggsave("vocabPlot.pdf")
```

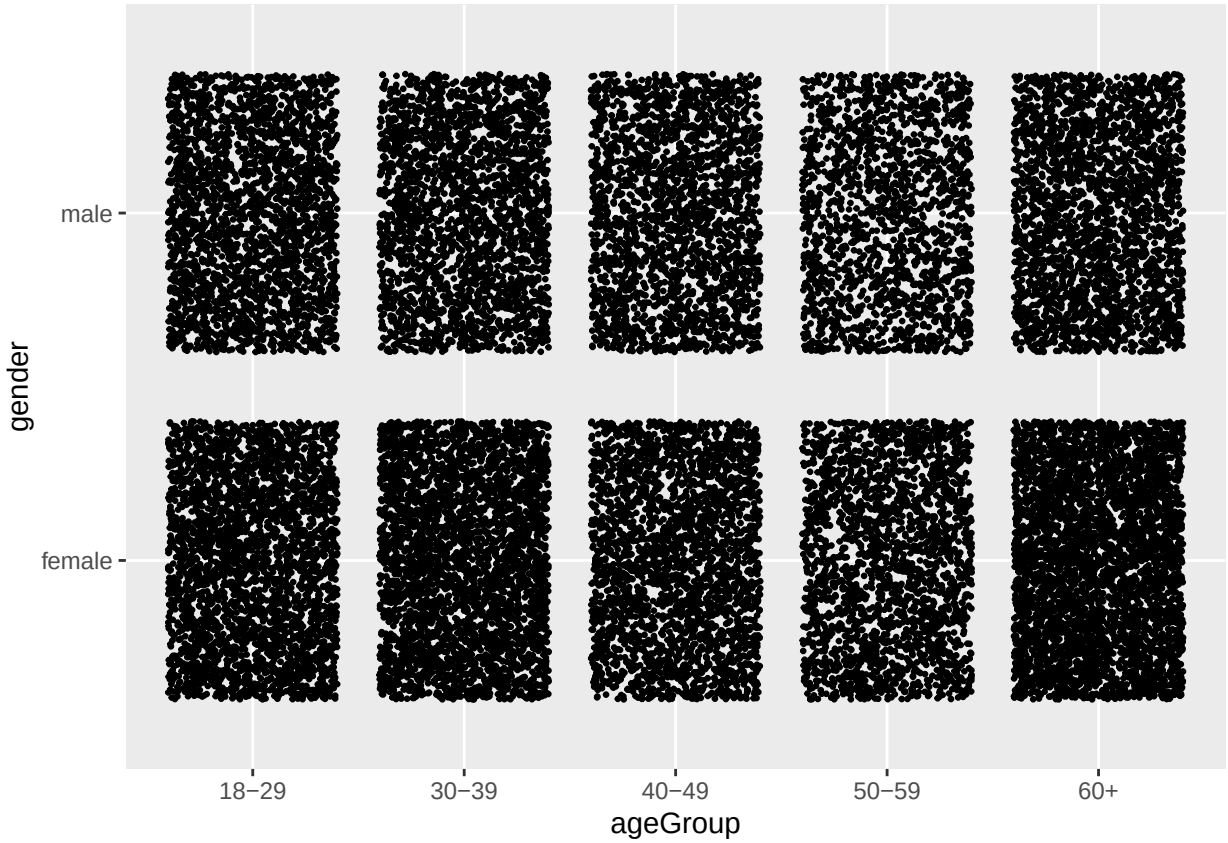
```
## Saving 6.5 x 4.5 in image
```

```
ggplot(GSSvocab) +
  aes(x = vocab) +
  geom_density(fill = "black")
```



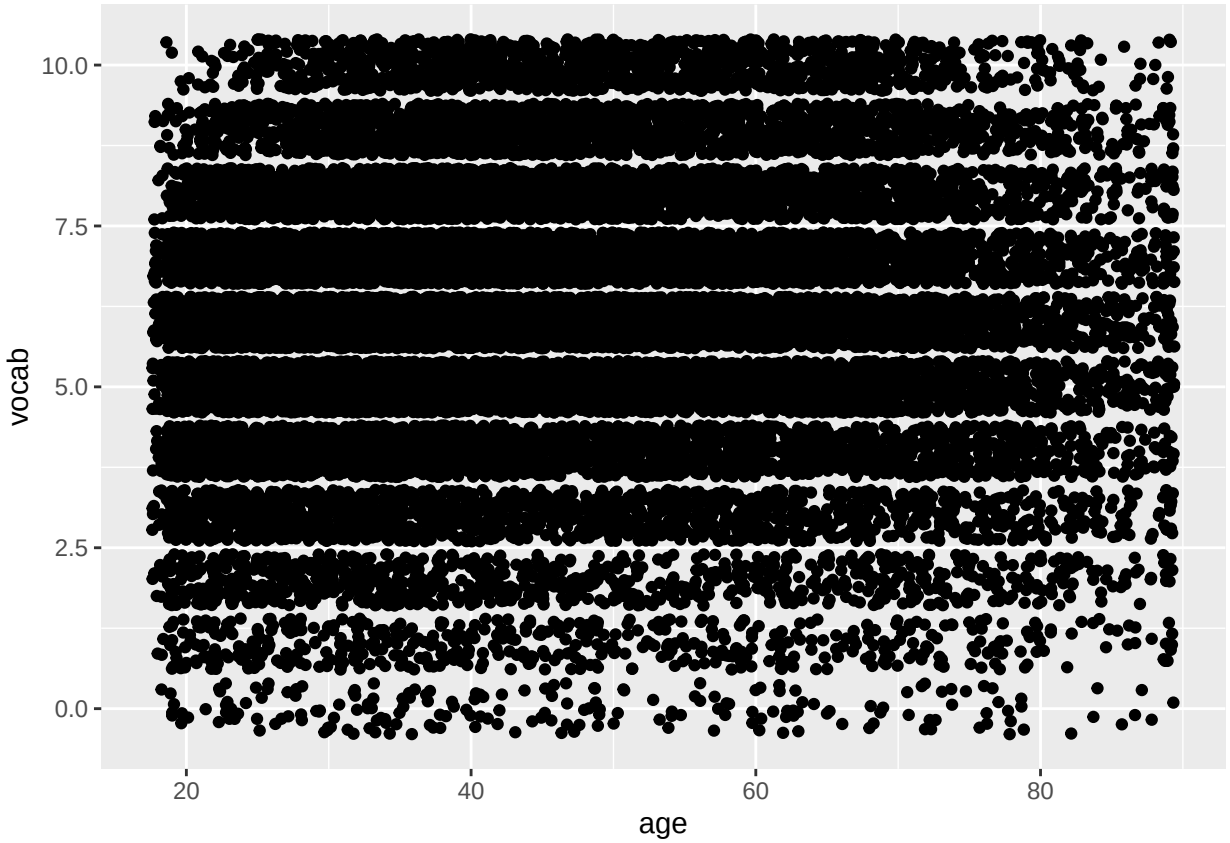
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

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



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

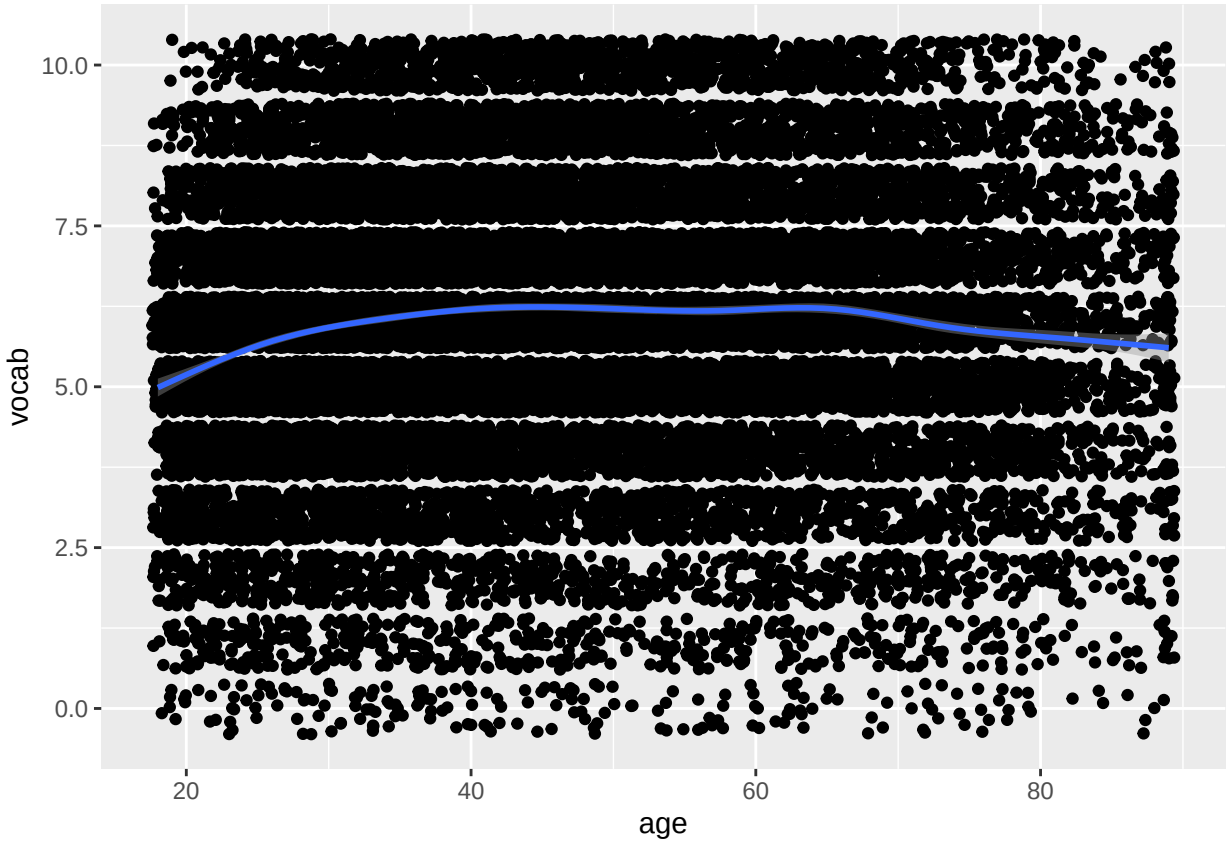
```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter()
```



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()
```

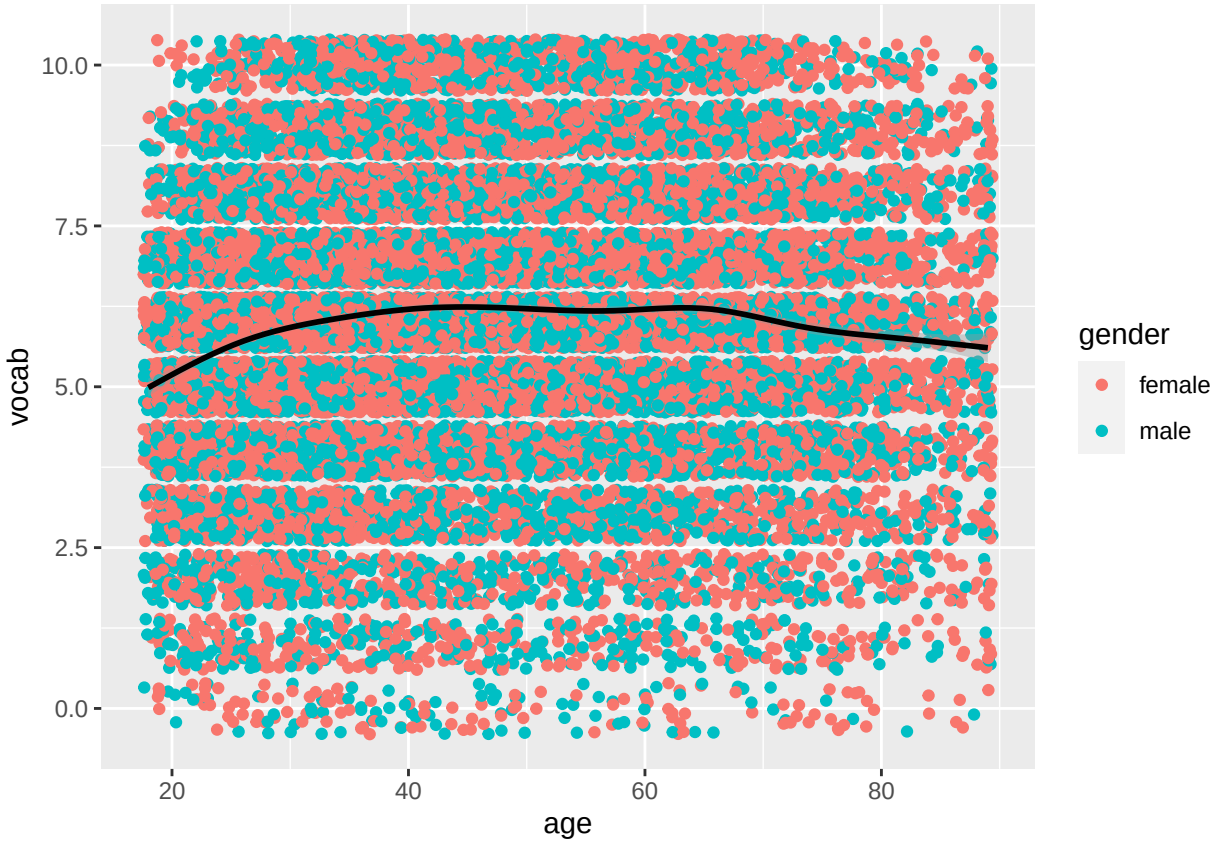
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Using the plot from the previous question, create the best looking 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 = "black")
```

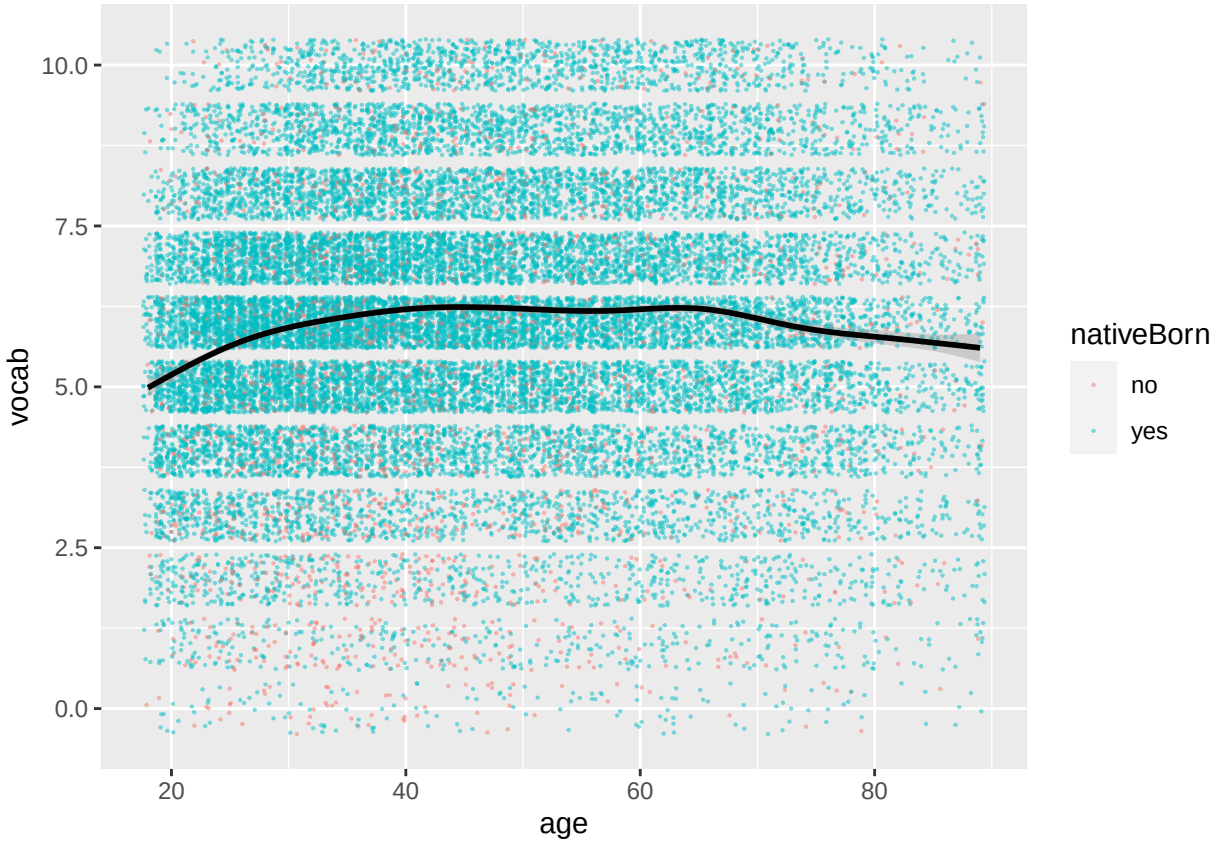
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



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

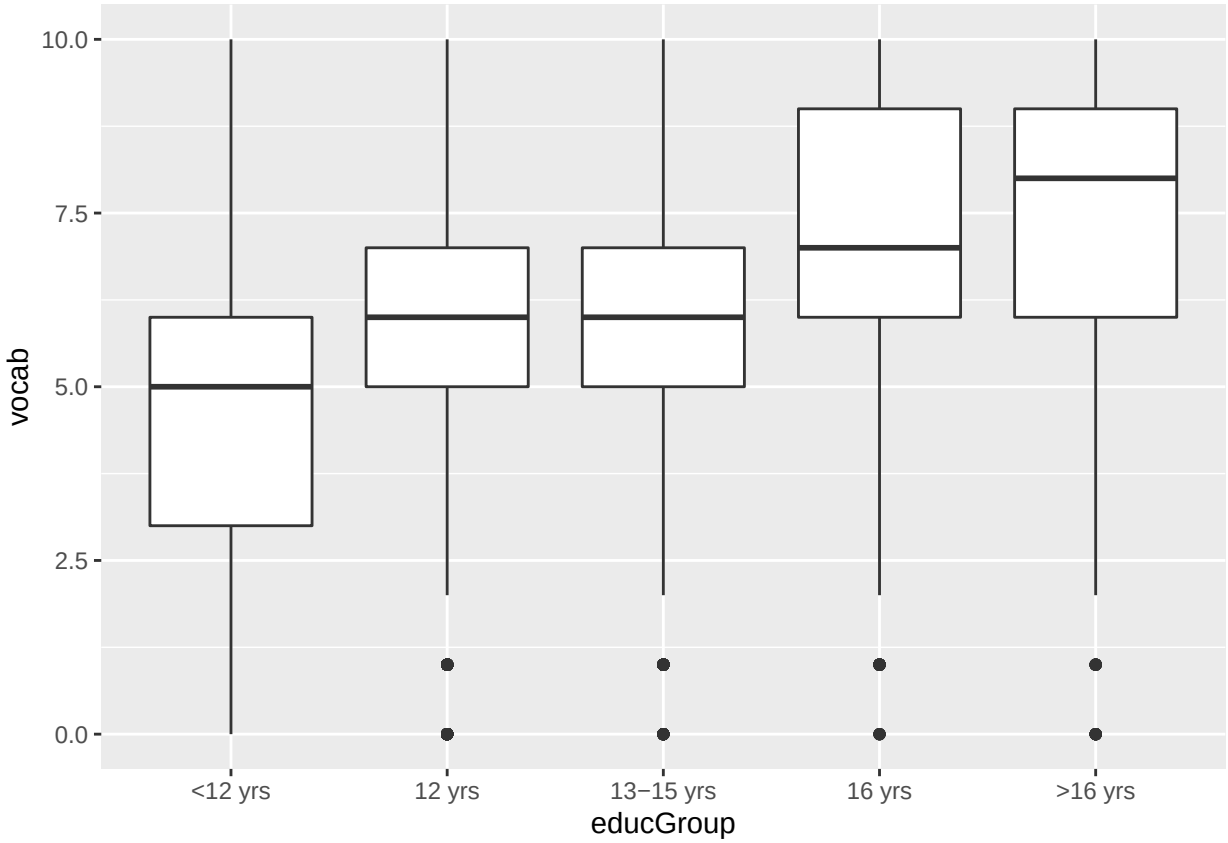
```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter(aes(col = nativeBorn), size = .1, alpha = 0.5) +
  geom_smooth(col = "black")
```

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

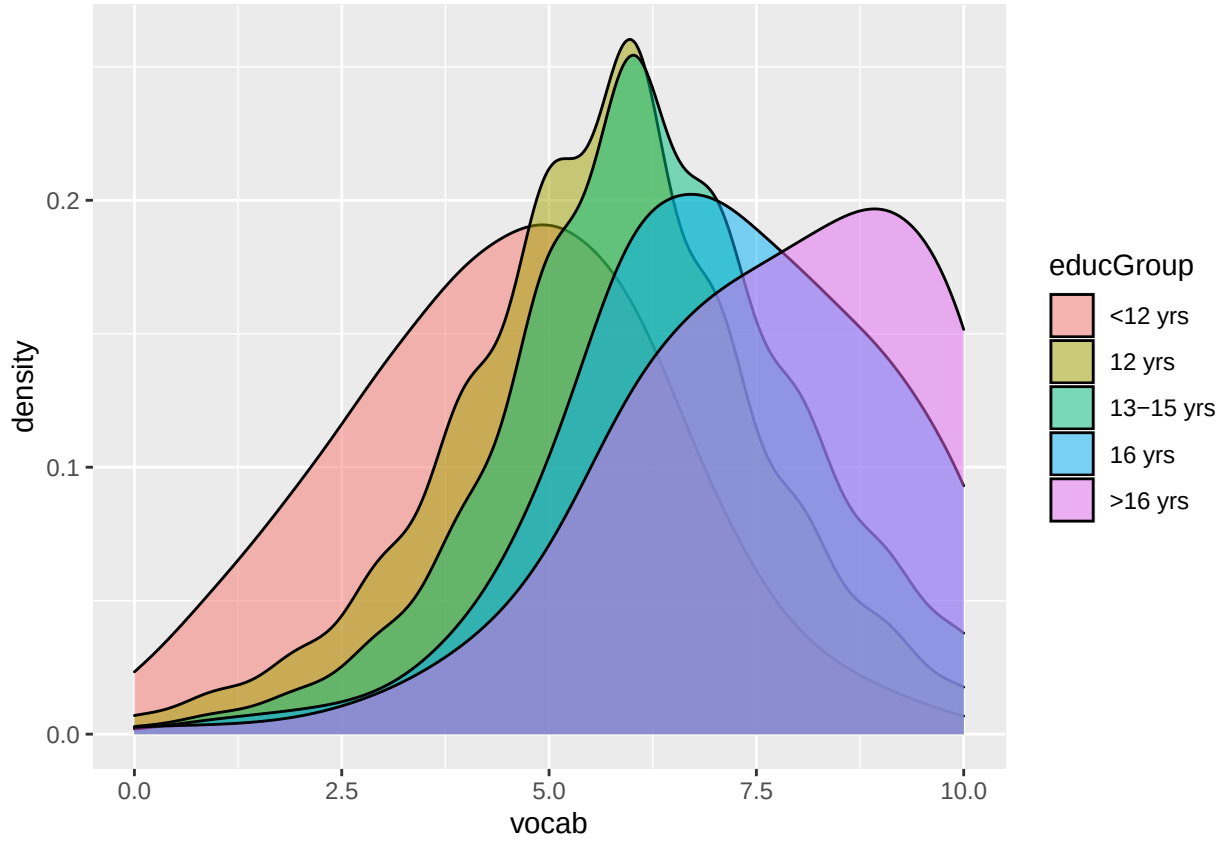


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

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

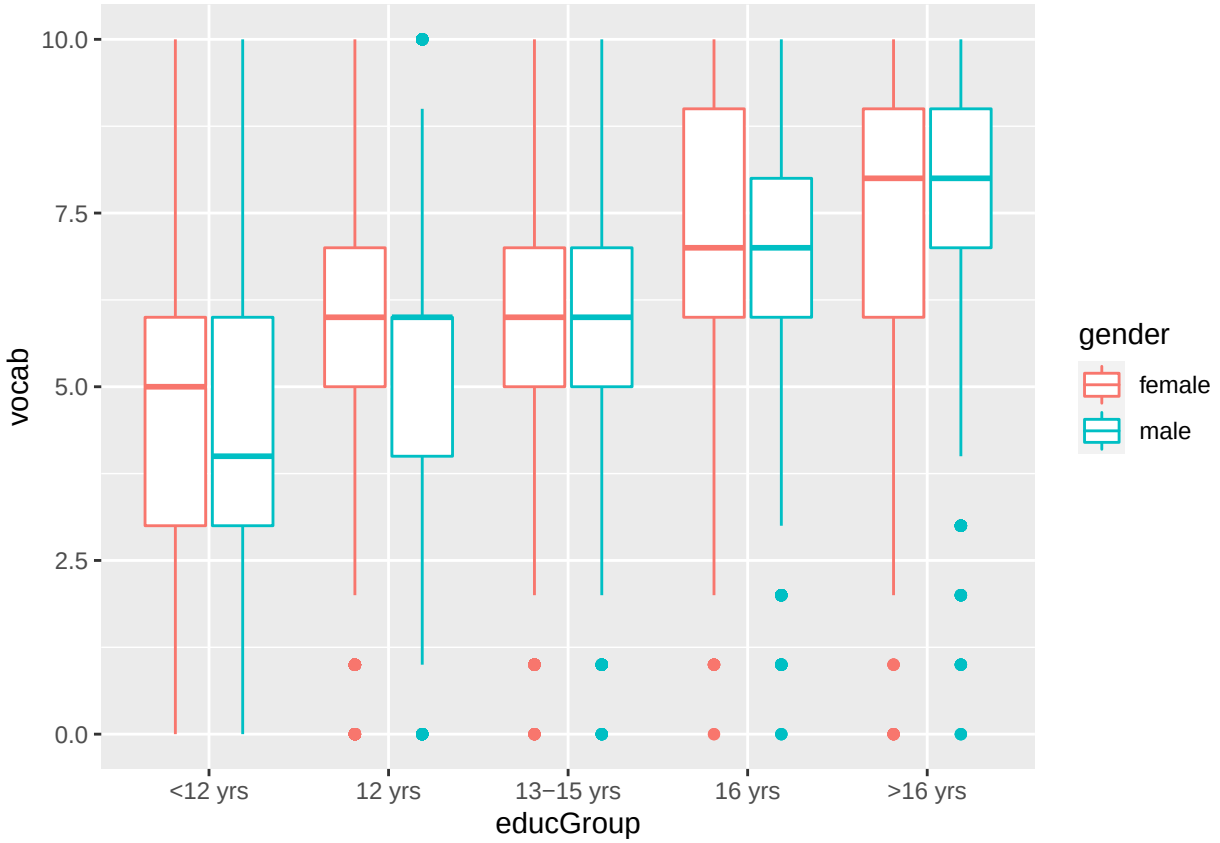



```
ggplot(GSSvocab) +
  aes(x = vocab) +
  geom_density(aes(fill = educGroup), adjust = 2, alpha = .5)
```



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`?

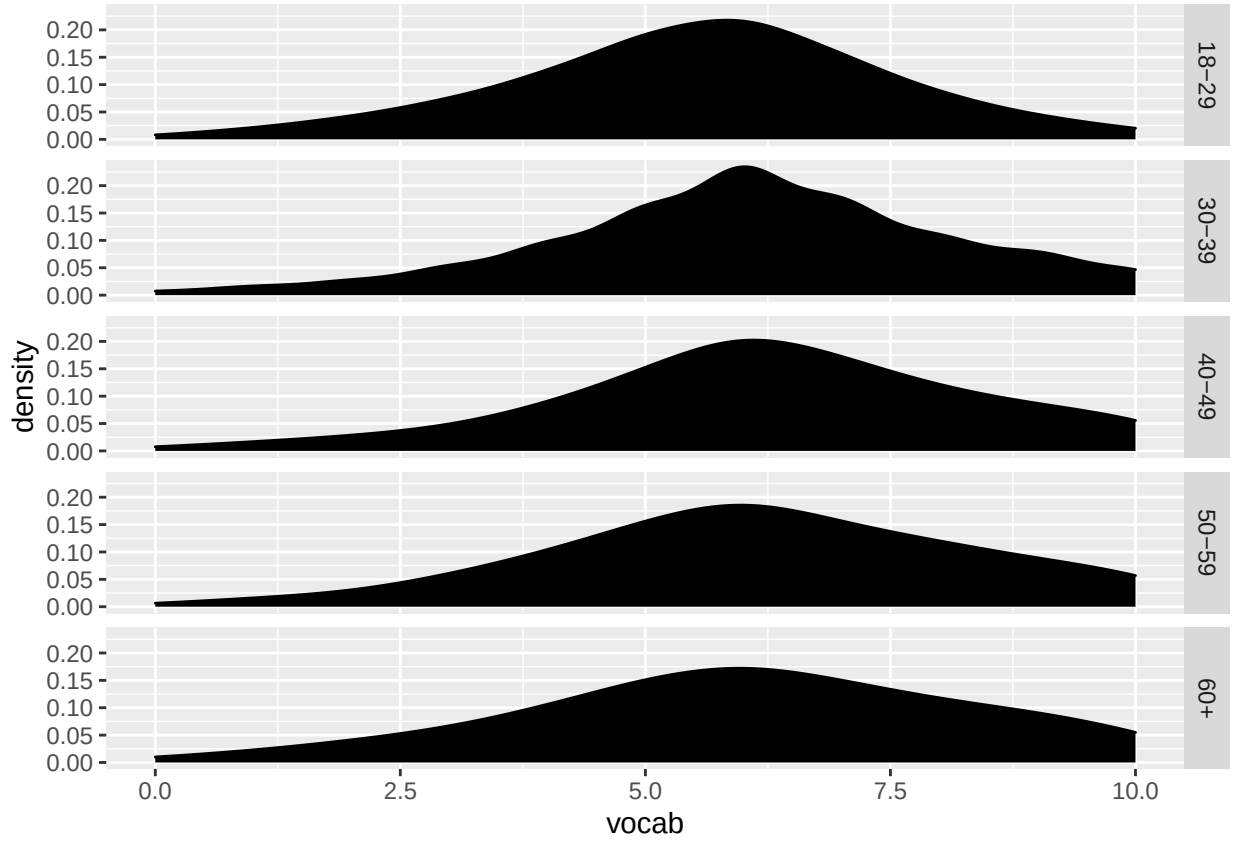
```
ggplot(GSSvocab) +
  aes(x = educGroup, y = vocab) +
  geom_boxplot(aes(col = gender))
```



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

No we are not 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)
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`.

```
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)

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

```
##
##          <=4         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
```

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

```
##
##          <=4         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 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] 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[tab < 50]
```

```
##
##         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
##                         42
```

```
names(tab[tab < 50])
```

```
##  [1] "Holand-Netherlands"         "Scotland"
##  [3] "Honduras"                   "Hungary"
##  [5] "Outlying-US(Guam-USVI-etc)" "Yugoslavia"
##  [7] "Laos"                       "Thailand"
##  [9] "Cambodia"                   "Trinadad&Tobago"
## [11] "Hong"                       "Ireland"
## [13] "Ecuador"                    "France"
## [15] "Greece"                     "Peru"
## [17] "Nicaragua"                  "Portugal"
## [19] "Haiti"                      "Iran"
## [21] "Taiwan"
```

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

Yes, this may be a problem because some countries features are very small that they don't really contribute to legit information for their own models, which potentially may lead to overfitting.

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_countr

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
```

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` 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$occupation = as.character(adult$occupation)
adult$workclass = as.character(adult$workclass)
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tabulate_worktype = sort(table(adult$worktype))
adult$occupation = NULL
adult$workclass = NULL
tabulate_worktype
```

```
##
##           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_worktype[tabulate_worktype < 100]), "Others"
adult$worktype = as.factor(adult$worktype)
table(adult$worktype)
```

```
##
##          Adm-clerical:Federal-gov            Adm-clerical:Local-gov
##                               316                               281
##             Adm-clerical:Private            Adm-clerical:State-gov
##                              2793                               250
##           Craft-repair:Local-gov             Craft-repair:Private
##                               143                              3146
##     Craft-repair:Self-emp-not-inc       Exec-managerial:Federal-gov
##                               523                               179
##         Exec-managerial:Local-gov           Exec-managerial:Private
##                               212                              2647
##      Exec-managerial:Self-emp-inc  Exec-managerial:Self-emp-not-inc
##                               385                               383
##         Exec-managerial:State-gov           Farming-fishing:Private
##                               186                               450
##  Farming-fishing:Self-emp-not-inc         Handlers-cleaners:Private
##                               430                              1255
##         Machine-op-inspct:Private           Other-service:Local-gov
##                              1882                               189
##            Other-service:Private    Other-service:Self-emp-not-inc
##                              2665                               173
```

```
##           Other-service:State-gov                           Others
##                               123                             1008
##           Priv-house-serv:Private       Prof-specialty:Federal-gov
##                               143                              167
##          Prof-specialty:Local-gov          Prof-specialty:Private
##                               692                             2254
##       Prof-specialty:Self-emp-inc  Prof-specialty:Self-emp-not-inc
##                               157                              365
##          Prof-specialty:State-gov        Protective-serv:Local-gov
##                               403                              304
##           Protective-serv:Private        Protective-serv:State-gov
##                               186                              116
##                     Sales:Private              Sales:Self-emp-inc
##                              2895                              281
##            Sales:Self-emp-not-inc             Tech-support:Private
##                               376                              723
##        Transport-moving:Local-gov         Transport-moving:Private
##                               115                             1247
## Transport-moving:Self-emp-not-inc
##                               118
```

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

```
adult$relationship = as.character(adult$relationship)
adult$marital_status = as.character(adult$marital_status)
adult$newRelationship = paste(adult$marital_status, adult$relationship, sep = ":")
tabulate_relationship = sort(table(adult$newRelationship))
adult$newRelationship = ifelse(adult$newRelationship %in% names(tabulate_relationship[tabulate_relations
tabulate2_relationship = sort(table(adult$newRelationship))
adult$relationship = NULL
adult$marital_status = NULL
tabulate2_relationship
```

```
##
##             Divorced:Other-relative              married:Other-relative
##                                 103                                 119
##     Married-spouse-absent:Unmarried Married-spouse-absent:Not-in-family
##                                 120                                 181
##                 Divorced:Own-child                   Widowed:Unmarried
##                                 308                                 343
##                             Others             Separated:Not-in-family
##                                 362                                 383
##                Separated:Unmarried               Widowed:Not-in-family
##                                 413                                 432
##        Never-married:Other-relative             Never-married:Unmarried
##                                 548                                 801
##                       married:Wife                  Divorced:Unmarried
##                                1406                                1535
##             Divorced:Not-in-family             Never-married:Own-child
##                                2268                                3929
##         Never-married:Not-in-family                    married:Husband
##                                4447                               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.

```
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(li
}
```

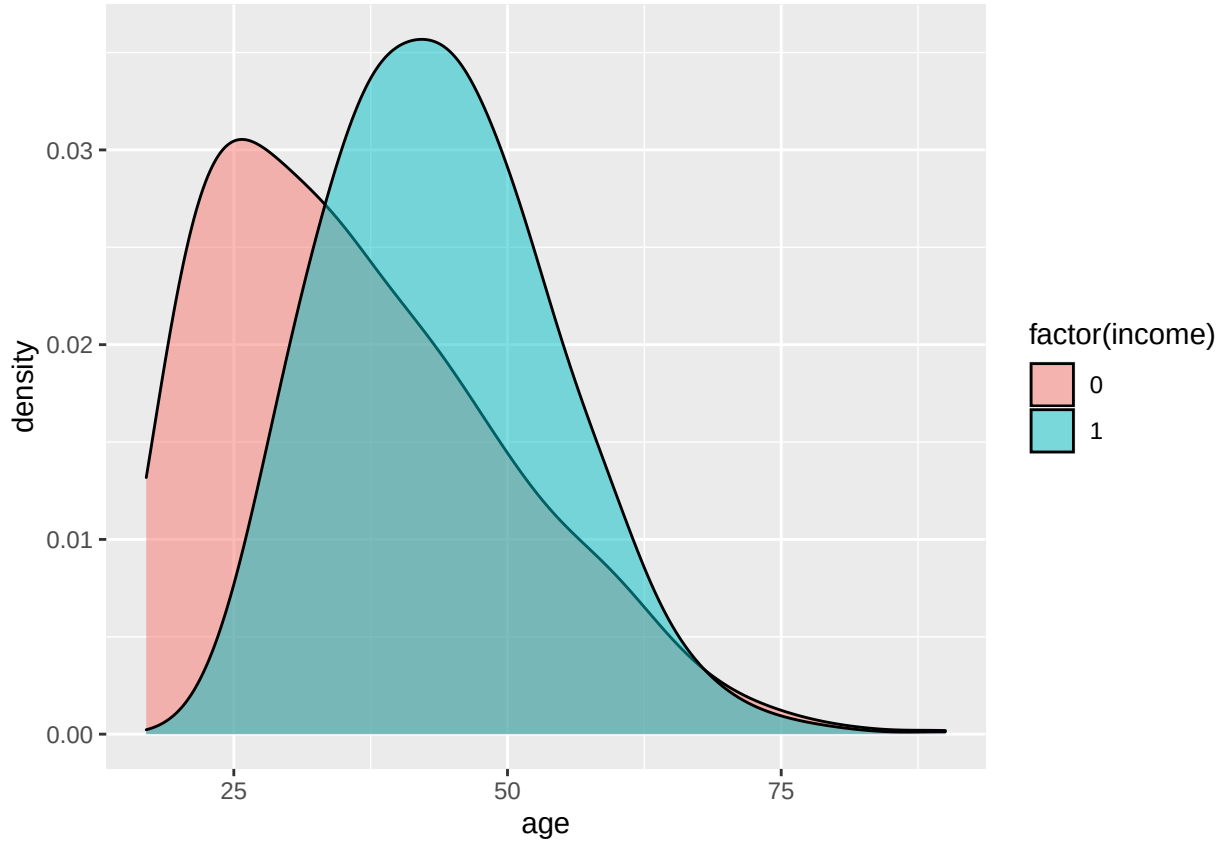
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## 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$log_capital_loss = log(1 + adult$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 because, as an adult your income tends to increase into your 20s, 30s and 40s. When people start getting into there late 50s and early 60s, the income tends to decrease because that is around the retirement age for most.

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, 
}
```

```
## 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
```

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] -0.1029524
##
## $`probit-vanilla`
## [1] -0.1030002
##
## $`cloglog-vanilla`
## [1] -0.1037007
##
## $`cauchit-vanilla`
## [1] -0.1045145
##
## $`logit-age_interactions`
## [1] -0.09966427
##
## $`probit-age_interactions`
## [1] -0.3626756
##
## $`cloglog-age_interactions`
## [1] -0.2509428
##
## $`cauchit-age_interactions`
## [1] -0.1006681
```

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] -0.1039165
##
## $`probit-vanilla`
## [1] -0.104026
##
## $`cloglog-vanilla`
## [1] -0.1056871
##
## $`cauchit-vanilla`
## [1] -0.1055944
##
## $`logit-age_interactions`
## [1] -0.1024649
##
## $`probit-age_interactions`
## [1] -0.3622347
##
## $`cloglog-age_interactions`
## [1] -0.2491711
##
## $`cauchit-age_interactions`
## [1] -0.1043539
```

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

The model that won the in-sample was logit and the model that won the out of sample was also logit because the complexity accounts for more relationships between the feature in logit.

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

The few problems with this model selection procedure is that, there is only one data test and the variance can be very high because we are not conducting the method of cross validation/K-fold split

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.

```
n = nrow(adult)
K = 5
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, ]
y_subtrain = adult_subtrain$income
adult_select = adult[select_indices, ]
y_select = adult_select$income
adult_test = adult[test_indices, ]
y_test = adult_test$income
```

```
mods = list()

for (link_function in link_functions){
  mods[[paste(link_function, "vanilla", sep = "-")]]=glm(formula = vanilla, data = adult_subtrain, fami
}
```

```
## 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
```

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

```
## 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
```

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

```
## logit-age_interactions
##                      5
```

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

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

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

```
## [1] -0.1015678
```