

# Lab 6

Sara Jedwab

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
```

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

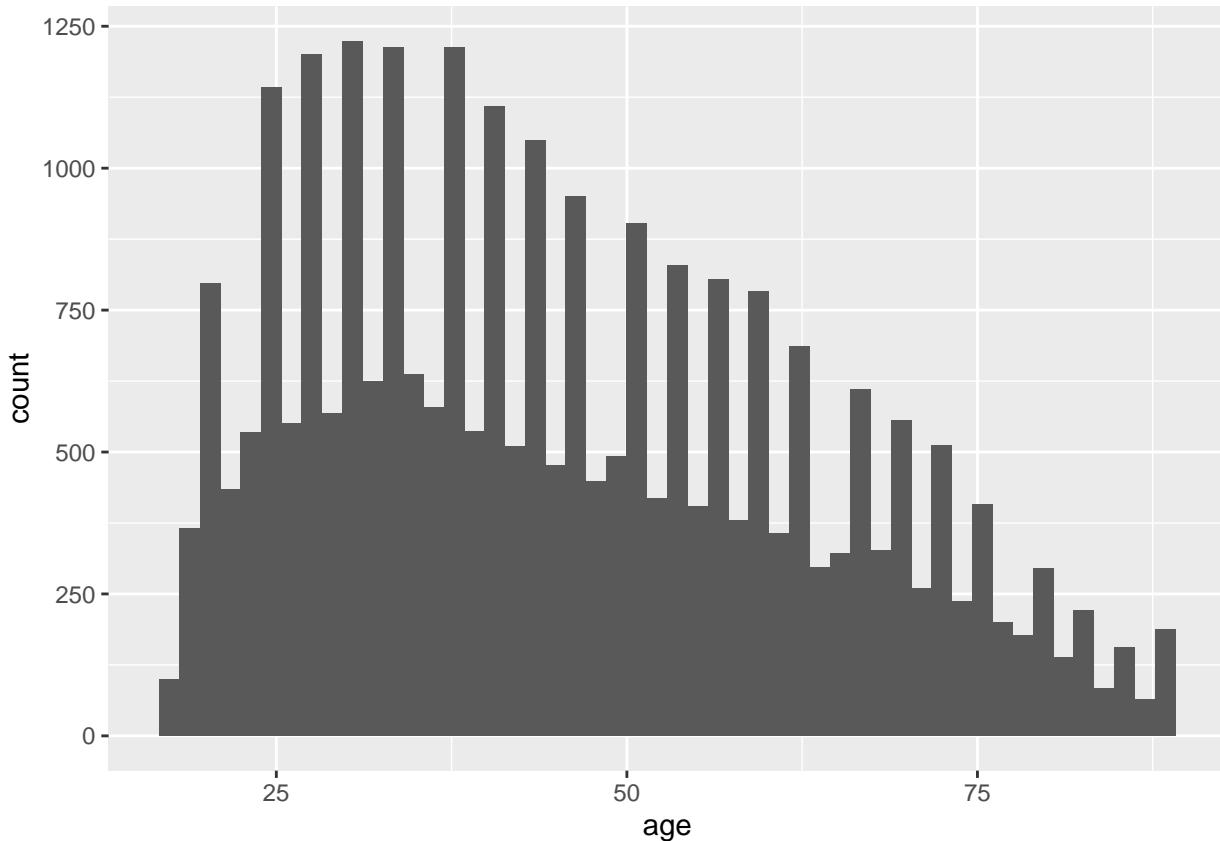
This dataset has 8 factors: (1) year - this is a continuous variable representing the year that the observation was collected (2) gender - this is a factor/categorical variable representing the respondents's gender (3) nativeBorn - this is a factor/categorical variable representing whether or not the respondent was born in the United States (4) ageGroup - this is a factor/categorical variable representing the grouped age of the respondent (5) educGroup - this is a factor/categorical variable representing the grouped education level of the respondent (starting with 12 years corresponding to high school graduate) (6) vocab - this is a continuous variable measuring the score on the given vocabulary test (out of 10) (7) age - this is a continous variable measuring the age of the respondent in years (8) educ - this is a continuous variable measuring the years of education of the respondent

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)

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

age_plot
```

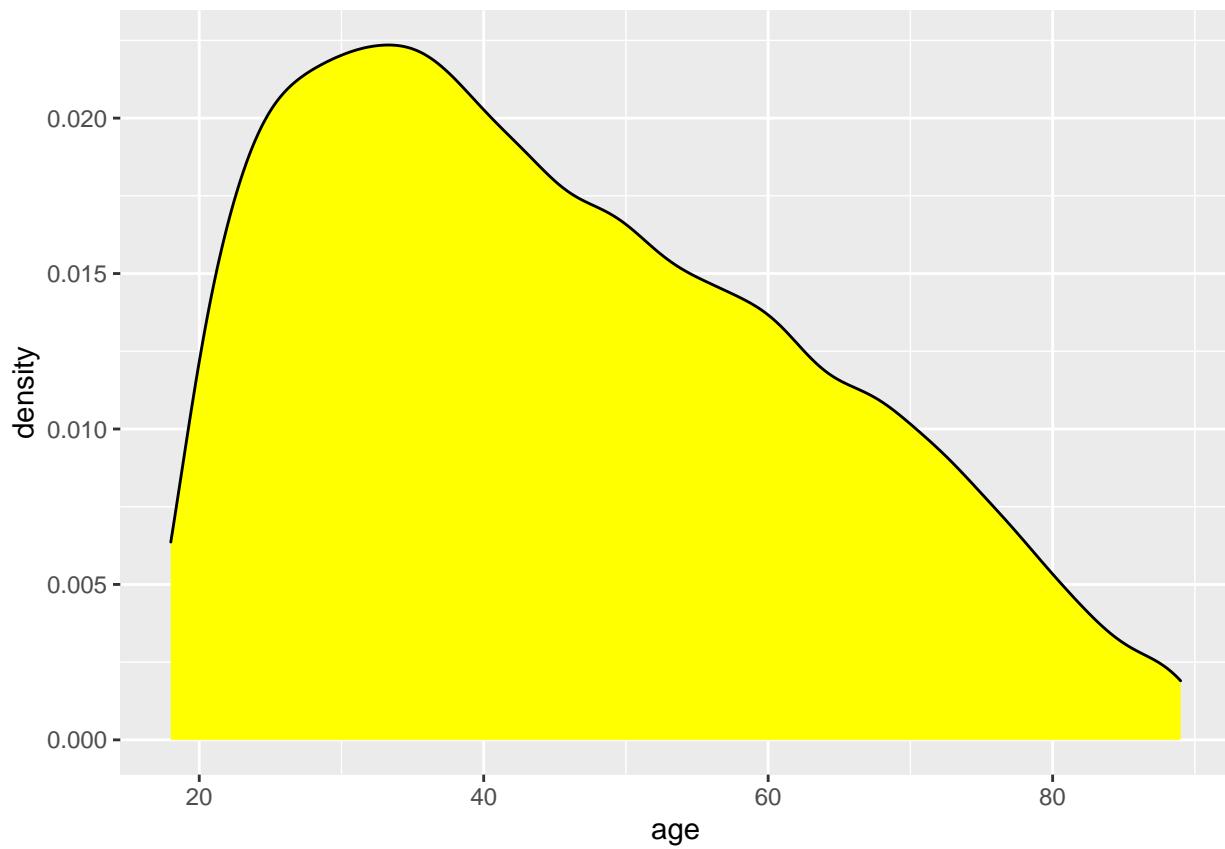


```
ggsave("age_plot.pdf")

## Saving 6.5 x 4.5 in image
#system("open age_plot.pdf")

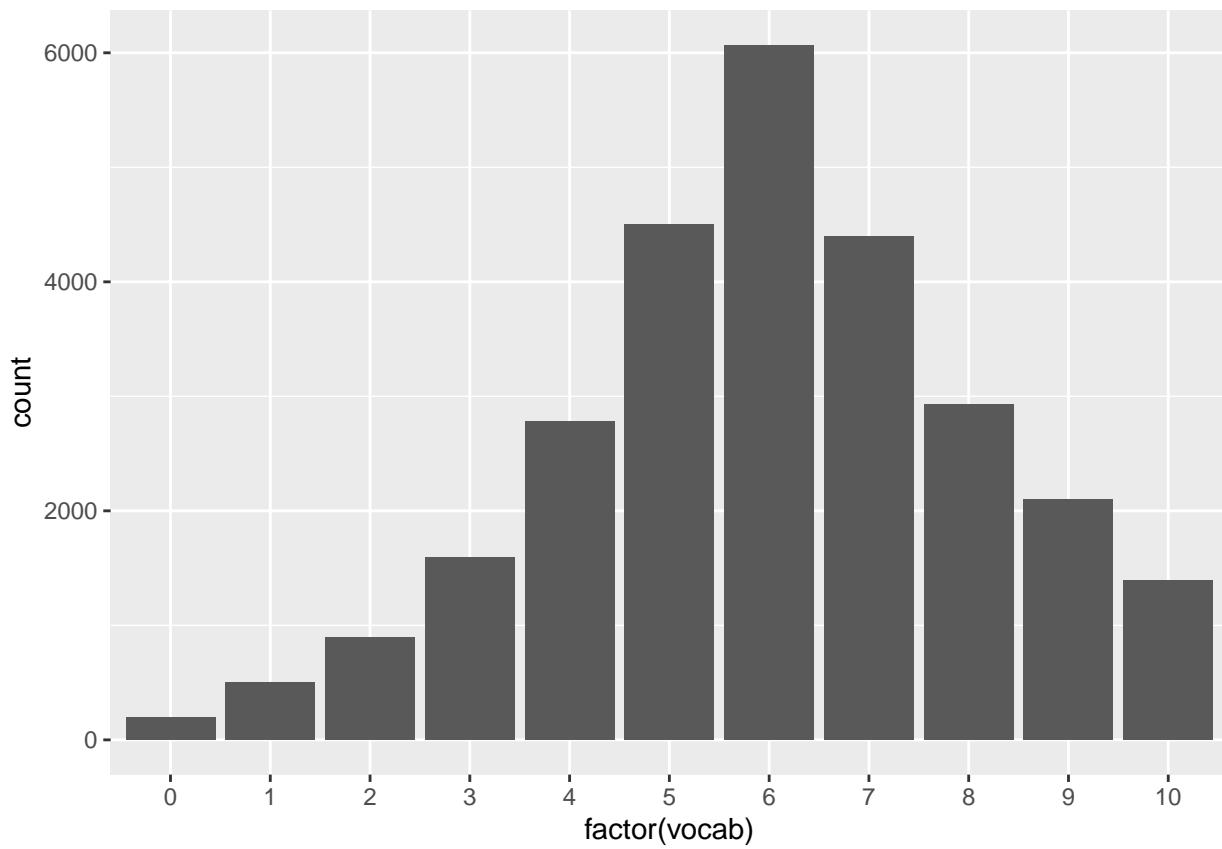
ggplot(GSSvocab) +
  aes(x = age) +
  geom_density(fill = "yellow", kernel = "guassian")

## Warning: Ignoring unknown parameters: kernel
```



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.

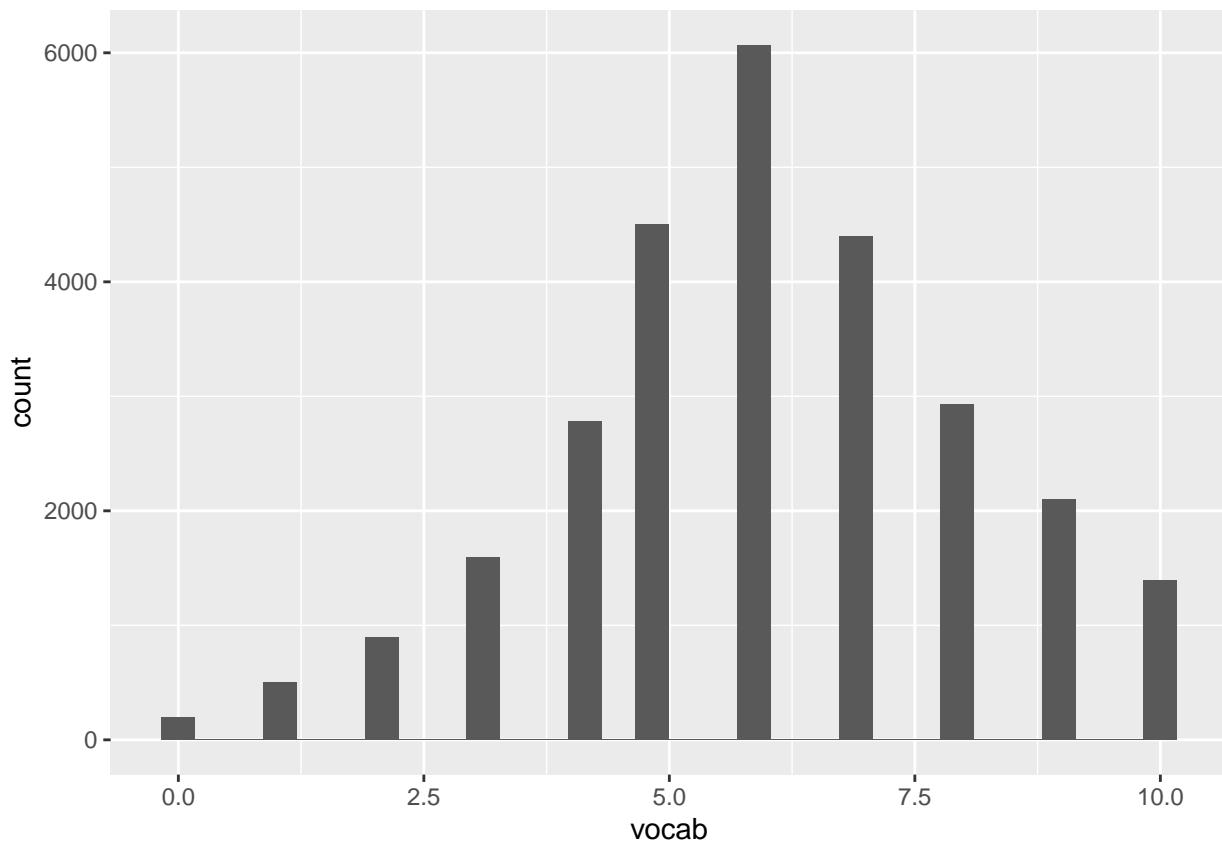
```
vocab_plot = ggplot(GSSvocab) +  
  aes(x = factor(vocab)) +  
  geom_bar()  
vocab_plot
```



```
ggsave("vocab_plot.pdf")

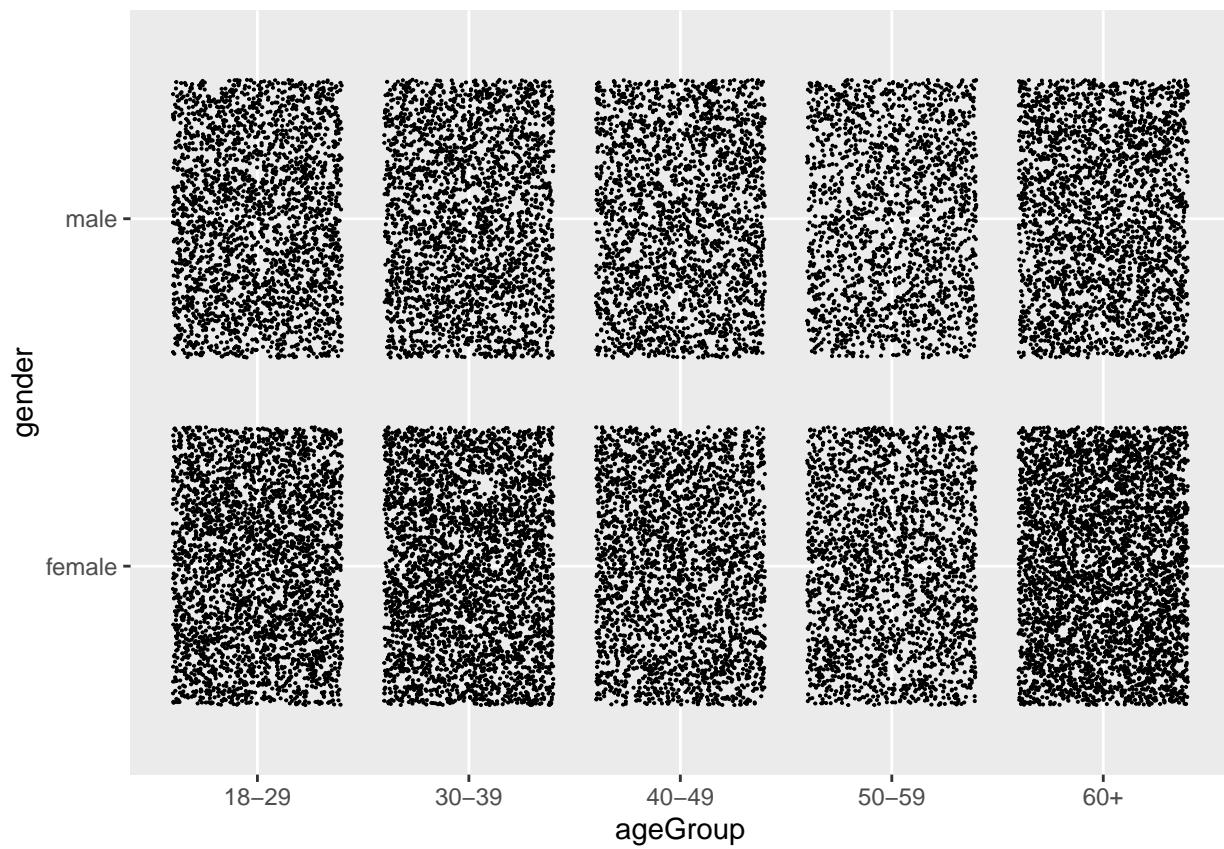
## Saving 6.5 x 4.5 in image
ggplot(GSSvocab) +
  aes(x = vocab) +
  geom_histogram()

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



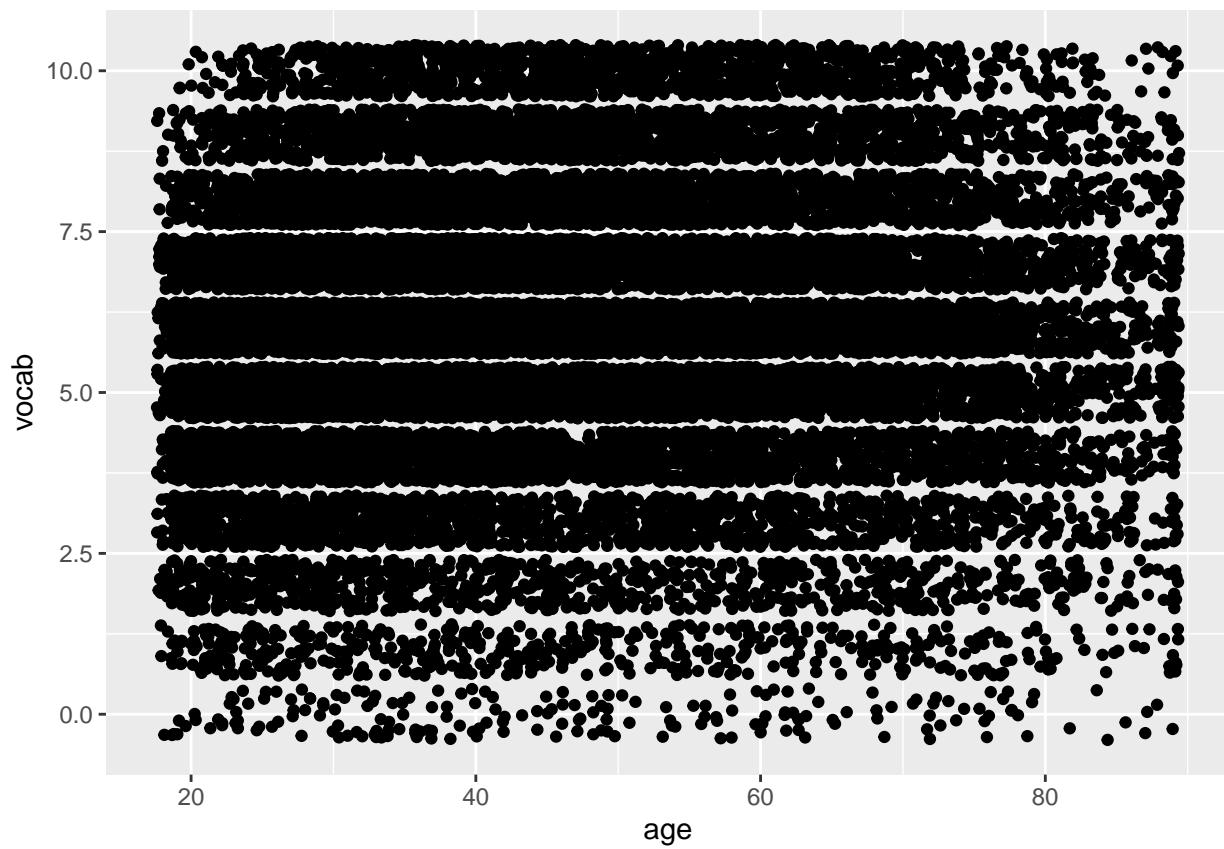
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```
ggplot(GSSvocab) +  
  aes(x = ageGroup, y = gender) +  
  geom_jitter(size = 0.05)
```



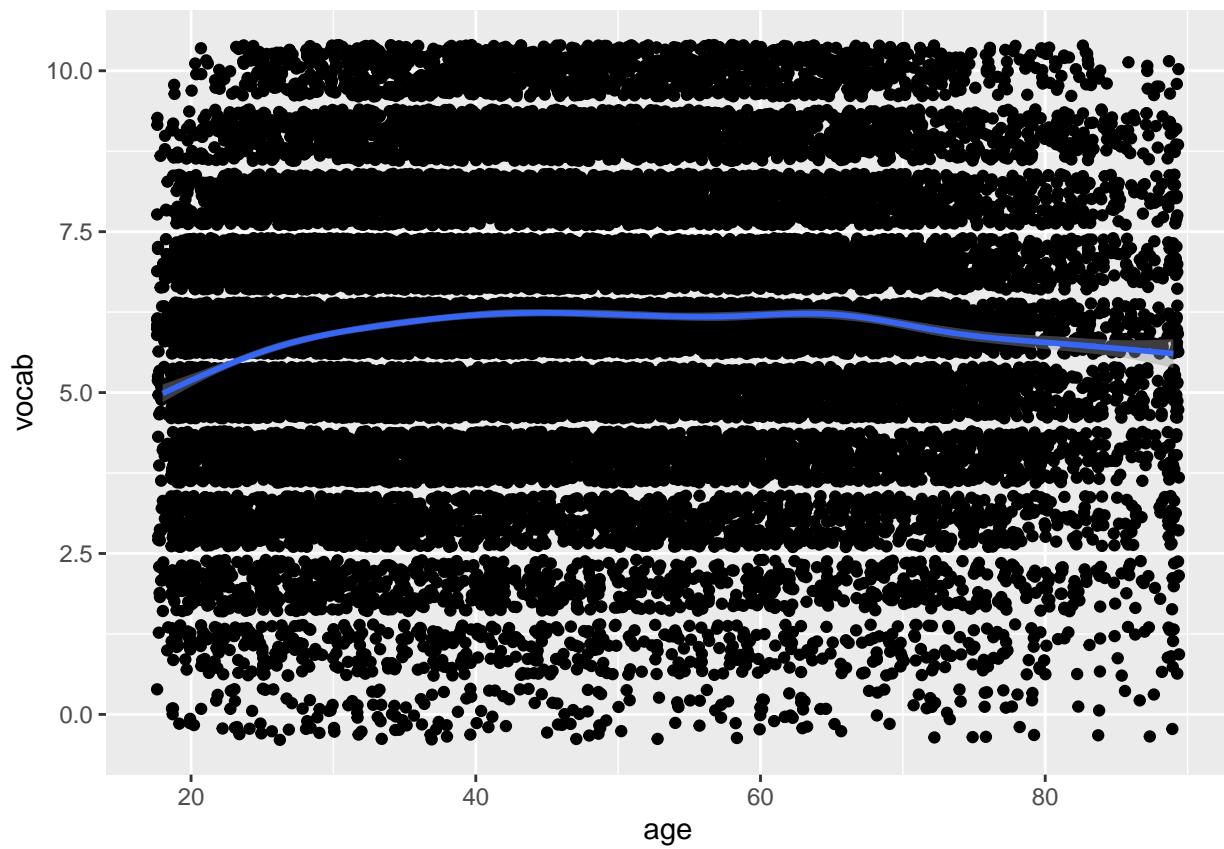
Create the best-looking plot you can to examine the `vocab` variable by `age`. Does there appear to be an association?

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



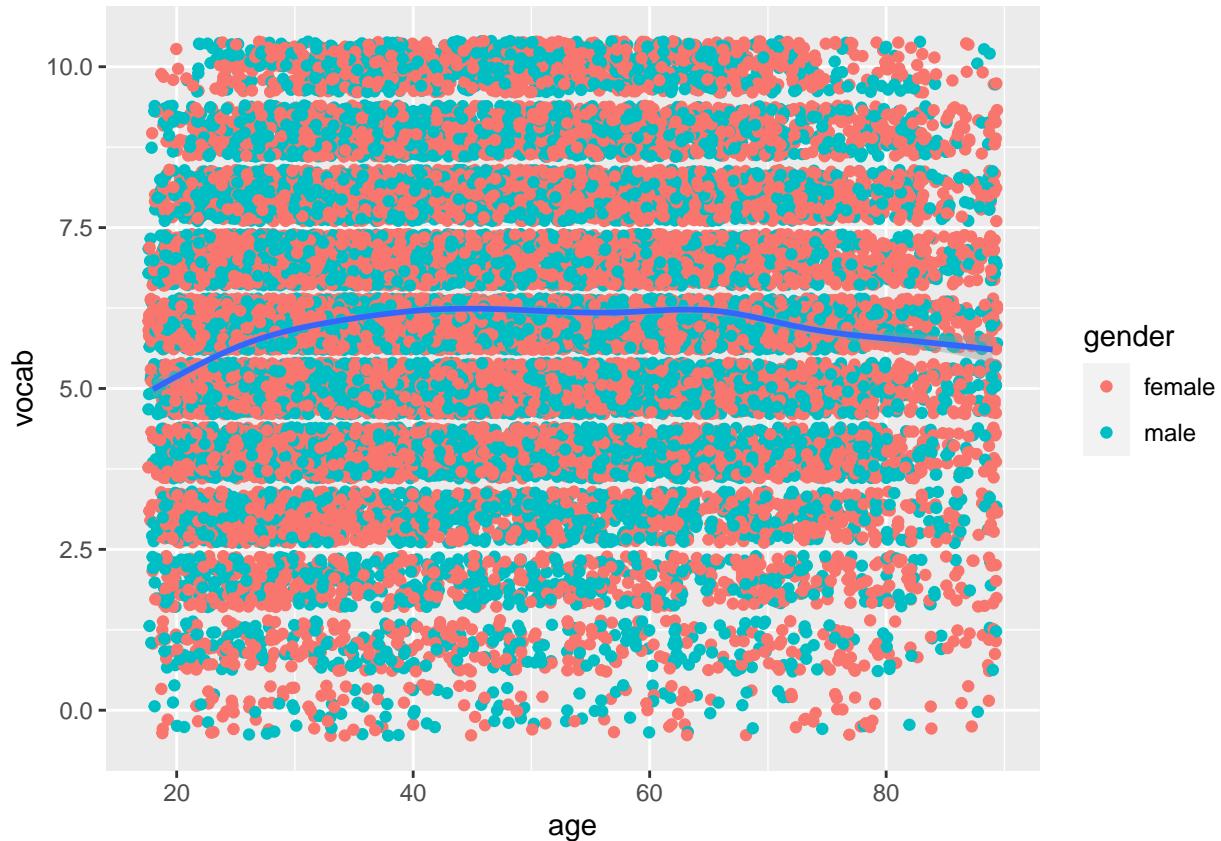
Add an estimate of  $f(x)$  using the smoothing geometry to the previous plot. Does there appear to be an association now?

```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter() + geom_smooth()
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



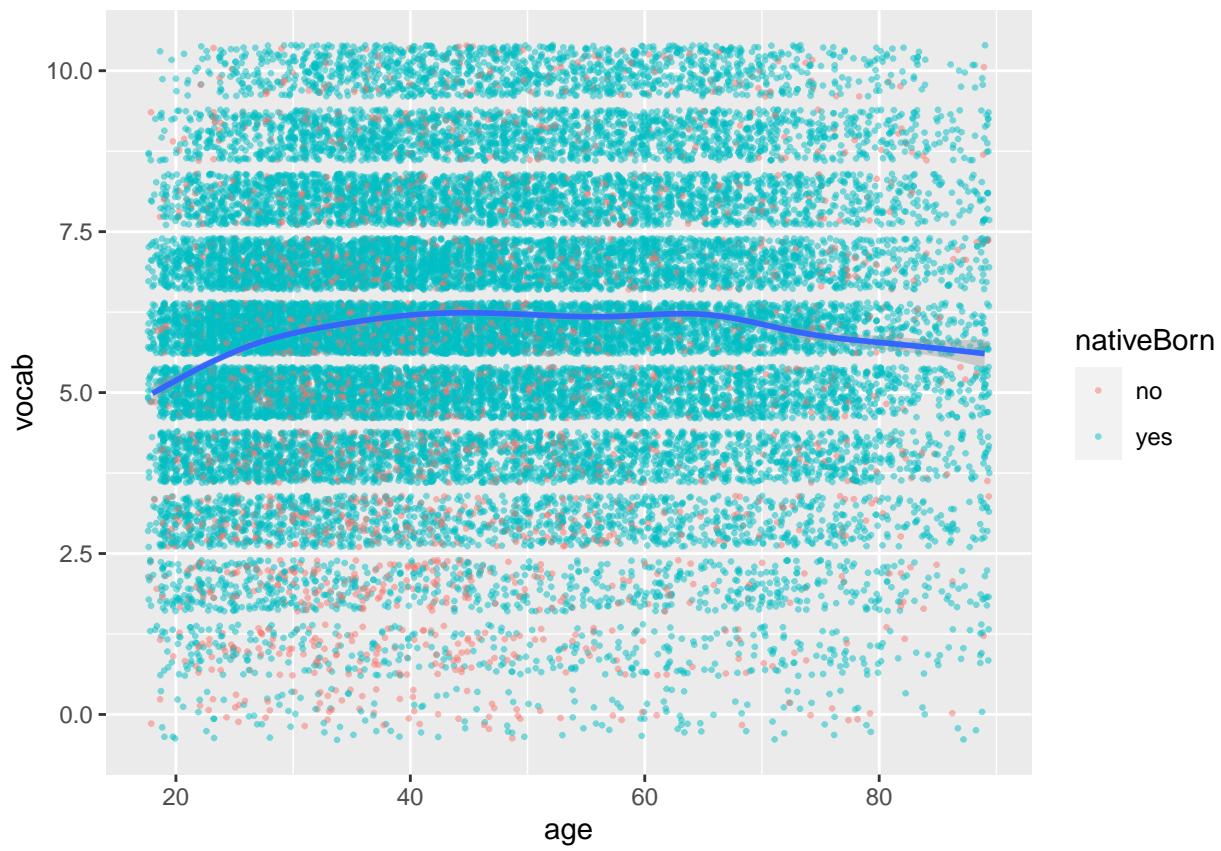
Using the plot from the previous question, create the best looking plot overloading with variable `gender`. Does there appear to be an interaction of `gender` and `age`?

```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter(aes(col = gender)) + geom_smooth()
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



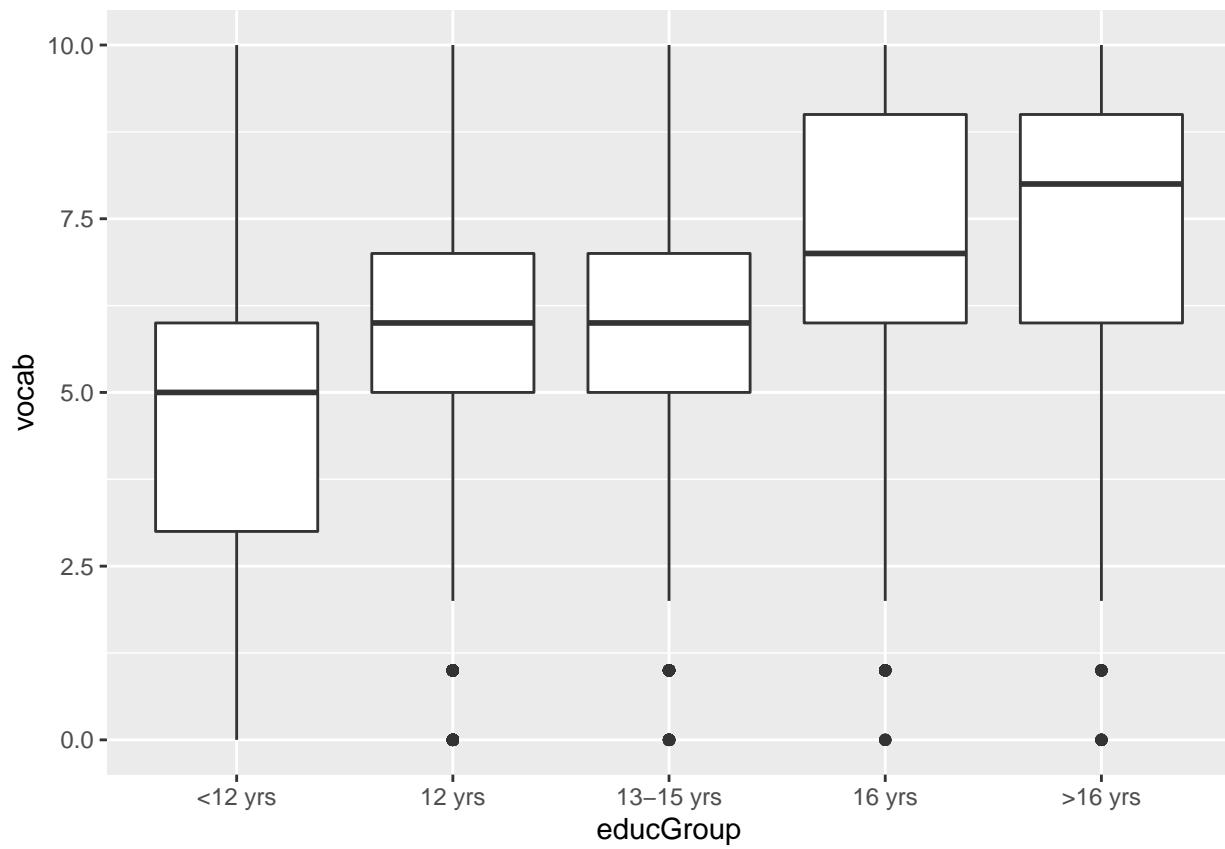
Using the plot from the previous question, create the best looking plot overloading with variable `nativeBorn`. Does there appear to be an interaction of `nativeBorn` and `age`?

```
ggplot(GSSvocab) +
  aes(x = age, y = vocab) +
  geom_jitter(aes(col = nativeBorn), size = 0.5, alpha = 0.5) + geom_smooth()
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

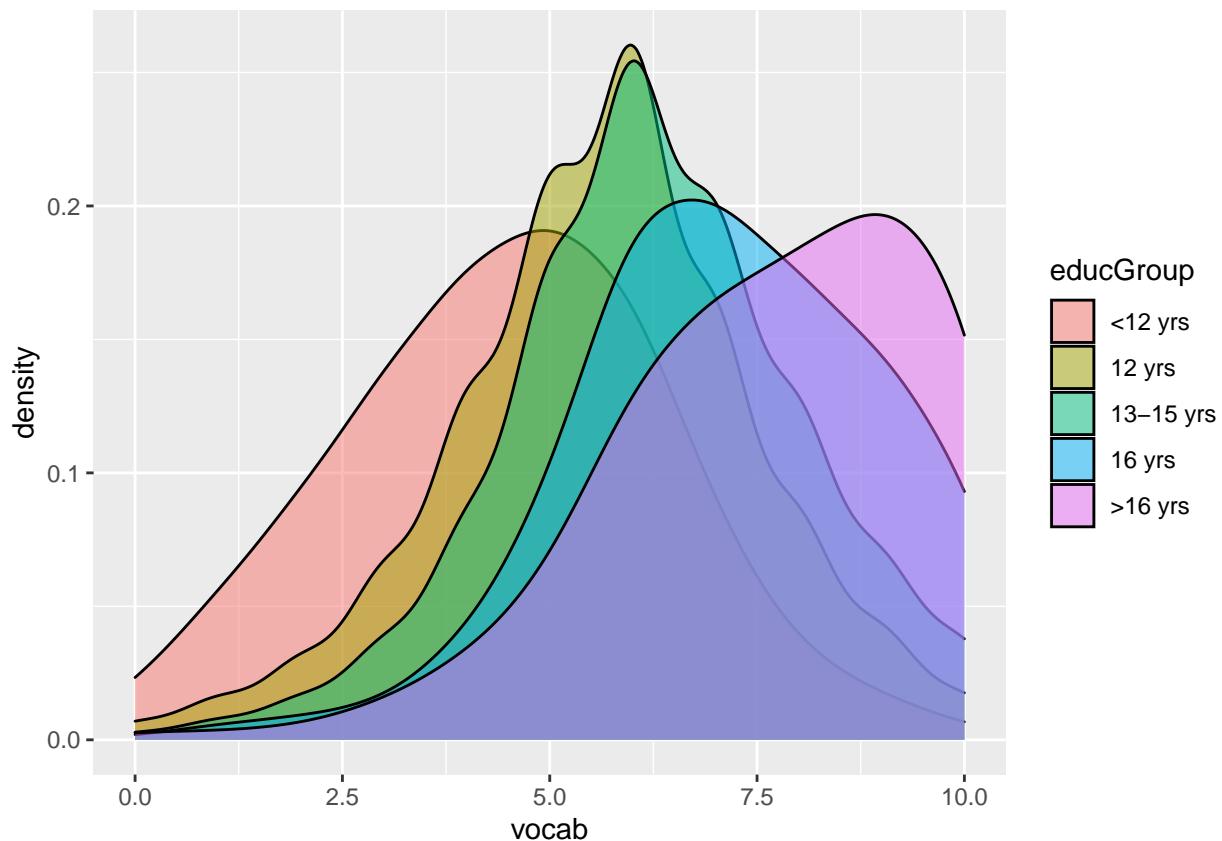


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

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

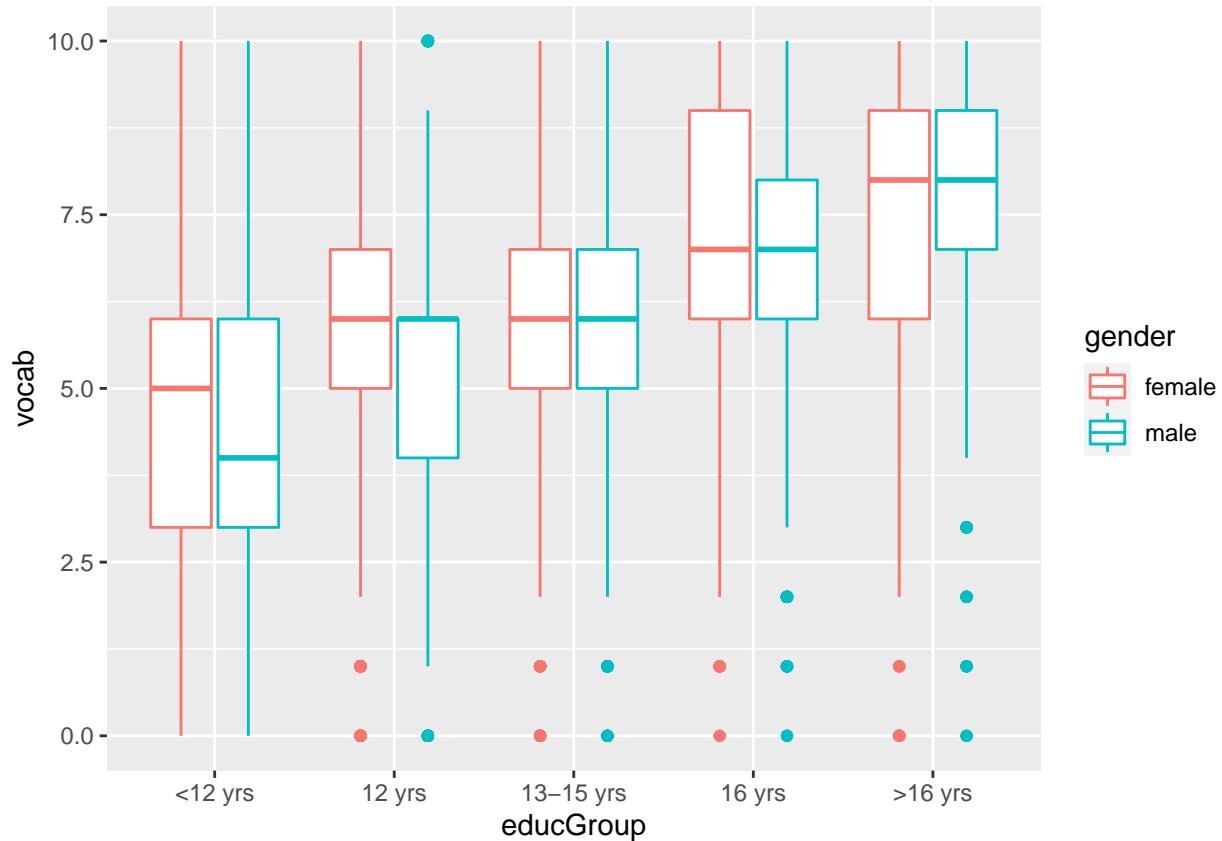


```
ggplot(GSSvocab) +  
  aes(x = vocab) +  
  geom_density(aes(fill = educGroup), adjust = 2, alpha = 0.5)
```



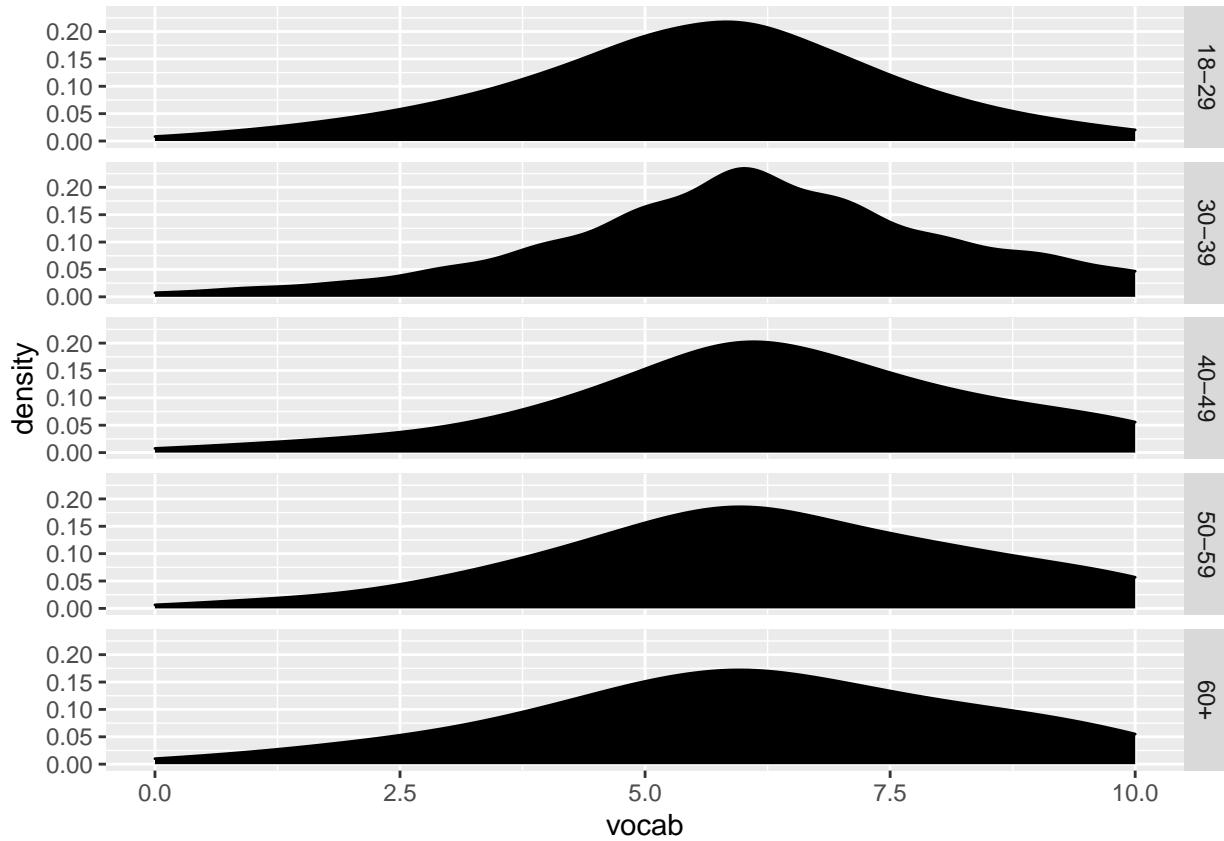
Using the best-looking plot from the previous question, create the best looking overloading with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?

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



Using facets, examine the relationship between `vocab` and `ageGroup`. Are we getting dumber? No, we're 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)
```

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

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

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

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

Create a model matrix `Xmm` (for this prediction task) on just the raw features and show that it is *not* full

rank (i.e. the result of `ncol` is greater than the result of `Matrix::rankMatrix`).

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

```
## [1] 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))
```

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

Yes you do. This is a problem because each country will each become its own feature, but they're each so small that they don't really contribute valuable information to the model on their own, so this could 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$native_country)
adult$native_country = as.factor(adult$native_country)
```

We're still not done getting this data down to full rank. Take a look at the model matrix just for `workclass` and `occupation`. Is it full rank? No, this is not 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$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tab_worktype = sort(table(adult$worktype))
adult$occupation = NULL
adult$workclass = NULL
tab_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
##      Priv-house-serv:Private      143
##                                     143
##      Prof-specialty:Federal-gov  167
##                                     167
##      Exec-managerial:Federal-gov 179
##                                     179
##      Protective-serv:Private     186
##                                     186
##      Exec-managerial:Local-gov   212
##                                     212
##      Adm-clerical:Local-gov     281
##                                     281
##      Protective-serv:Local-gov   304
##                                     304
##      Prof-specialty:Self-emp-not-inc 365
##                                     365
##      Exec-managerial:Self-emp-not-inc 383
##                                     383
##      Prof-specialty:State-gov     403
##                                     403
##      Farming-fishing:Private     450
##                                     450
##      Prof-specialty:Local-gov    692
##                                     692
##      Transport-moving:Private    1247
##                                     1247
##      Machine-op-inspct:Private   1882
##                                     1882
##      Exec-managerial:Private     2647
##                                     2647
##      Adm-clerical:Private       2793
##                                     2793
##      Craft-repair:Private       3146
##                                     3146

##                                     143
##      Prof-specialty:Self-emp-inc 157
##                                     157
##      Other-service:Self-emp-not-inc 173
##                                     173
##      Exec-managerial:State-gov   186
##                                     186
##      Other-service:Local-gov     189
##                                     189
##      Adm-clerical:State-gov     250
##                                     250
##      Sales:Self-emp-inc        281
##                                     281
##      Adm-clerical:Federal-gov   316
##                                     316
##      Sales:Self-emp-not-inc    376
##                                     376
##      Exec-managerial:Self-emp-inc 385
##                                     385
##      Farming-fishing:Self-emp-not-inc 430
##                                     430
##      Craft-repair:Self-emp-not-inc 523
##                                     523
##      Tech-support:Private       723
##                                     723
##      Handlers-cleaners:Private  1255
##                                     1255
##      Prof-specialty:Private     2254
##                                     2254
##      Other-service:Private      2665
##                                     2665
##      Sales:Private             2895
##                                     2895

```

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(tab_worktype[tab_worktype<100]), "Other", adult$worktype)
adult$worktype = as.factor(adult$worktype)
sort(table(adult$worktype))

##                                     115
##      Transport-moving:Local-gov  115
##                                     115
##      Transport-moving:Self-emp-not-inc 118
##                                     118
##      Craft-repair:Local-gov        143
##                                     143
##      Prof-specialty:Self-emp-inc   157
##                                     157
##      Other-service:Self-emp-not-inc 173
##                                     173
##      Protective-serv:State-gov    116
##                                     116
##      Other-service:State-gov      123
##                                     123
##      Priv-house-serv:Private      143
##                                     143
##      Prof-specialty:Federal-gov   167
##                                     167
##      Exec-managerial:Federal-gov  179
##                                     179

```

```

##                               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$relationship = as.character(adult$relationship)
adult$marital_status = as.character(adult$marital_status)
adult$status = paste(adult$relationship, adult$marital_status, sep = ":")
adult$status = as.character(adult$status)
tab_status = sort(table(adult$status))
adult$relationship = NULL
adult$marital_status = NULL
adult$status = as.factor(adult$status)

tab_status

##
##                               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` object 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

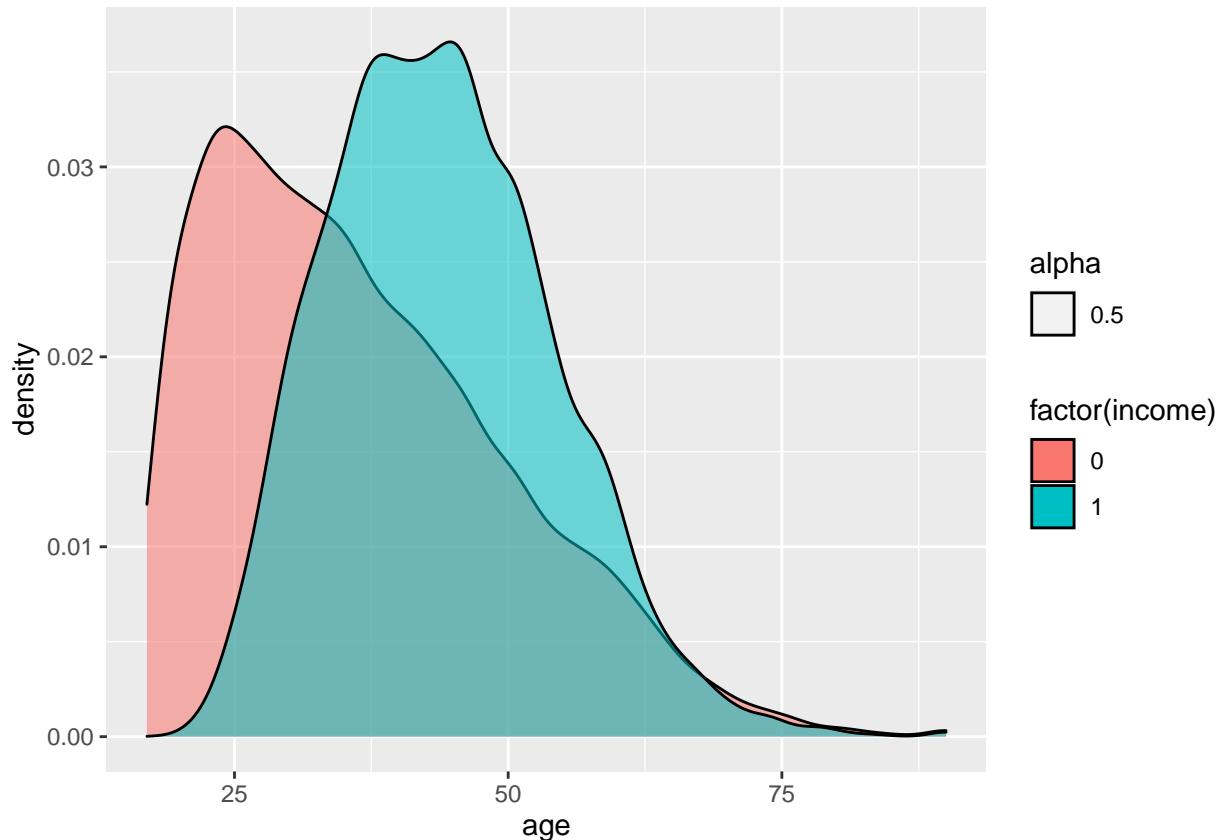
```

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$log_capital_loss = log(1 + adult$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), alpha = 0.5))
```



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

This shows a shift to the right in the distribution of income by age; it makes sense that older people tend to have a higher income than younger people.

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.

```
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

age_interactions = income ~ .*age

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

## 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
## 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] -0.1031617
##
## $`probit-vanilla`
## [1] -0.1031872
##
## $`cloglog-vanilla`
## [1] -0.1041009
##
## $`cauchit-vanilla`
## [1] -0.104773
##
## $`logit-age_interactions`
## [1] -0.09960394
##
## $`probit-age_interactions`
## [1] -0.1613826
##
## $`cloglog-age_interactions`
## [1] -0.100328
##
## $`cauchit-age_interactions`
## [1] -0.1005902

```

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] -0.1009183
##
## $`probit-age_interactions`
## [1] -0.16313
##
## $`cloglog-age_interactions`
## [1] -0.1017733
##
## $`cauchit-age_interactions`
## [1] -0.1025621
```

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

The logit and probit models are basically tied for best out of sample, this is to be expected because logit is typically expected to be best. We also expected that the age interaction model to do better than the vanilla models, since the complexity accounts for more of the feature's relationships.

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

We aren't do a K-fold split/cross validation, there is only one set of test data, so the oos Error is going to be more variable.

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
set.seed(1984)
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_train = adult[master_train_indices,]

adult_subtrain = adult[subtrain_indices, ]
y_subtrain = adult_subtrain$income

adult_select = adult[select_indices, ]
y_adult_select = adult_select$income
```

```

adult_select$income = NULL

adult_test = adult[test_indices, ]
y_adult_test = adult_test$income
adult_test$income = NULL

vanilla = income ~ .
age_interactions = income ~ .*age
link_functions = c("logit", "probit", "cloglog", "cauchit")
mods = list()

for(link_function in link_functions){
  mods[[paste(link_function, "vanilla", sep = "-")]] = glm(vanilla, adult_subtrain, family = binomial())
}

## 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
## 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, family = binomial())
}

## 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
briers = lapply(mods, brier_score, adult_select, y_adult_select)
which_final = which.max(briers)
which_final

## logit-vanilla
##           1
g_final = glm(income ~ ., adult_train, family = binomial(link = logit))

## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
brier_score(g_final, adult_test, y_adult_test)

## [1] -0.1059241

```