Lab 6

Amir ElTabakh

11:59PM April 15, 2021

#Visualization with the package ggplot2

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

Load up the `GSSvocab` dataset in package `carData` as `X` and drop all observations with missing measurements.

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

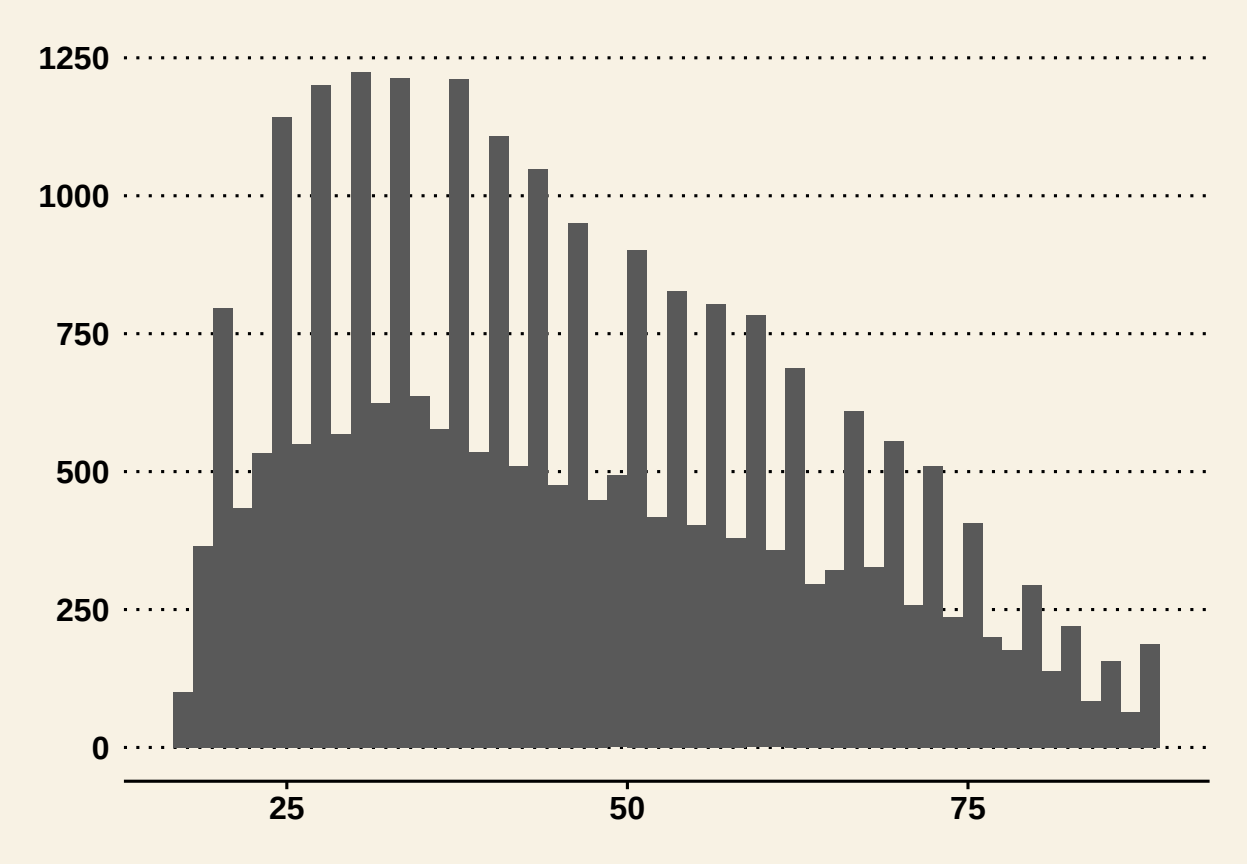
```
## starting httpd help server ... done
```

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

There are 29 thousand observations, with 8 variables. The data set illustrates a students characteristics with features such as age, nativeBorn, years of education, and year they took the test. The vocab feature specifies the number of words correct out of 10 on a vocabulary test. There is an age feature that represents the age of the respondent in years.

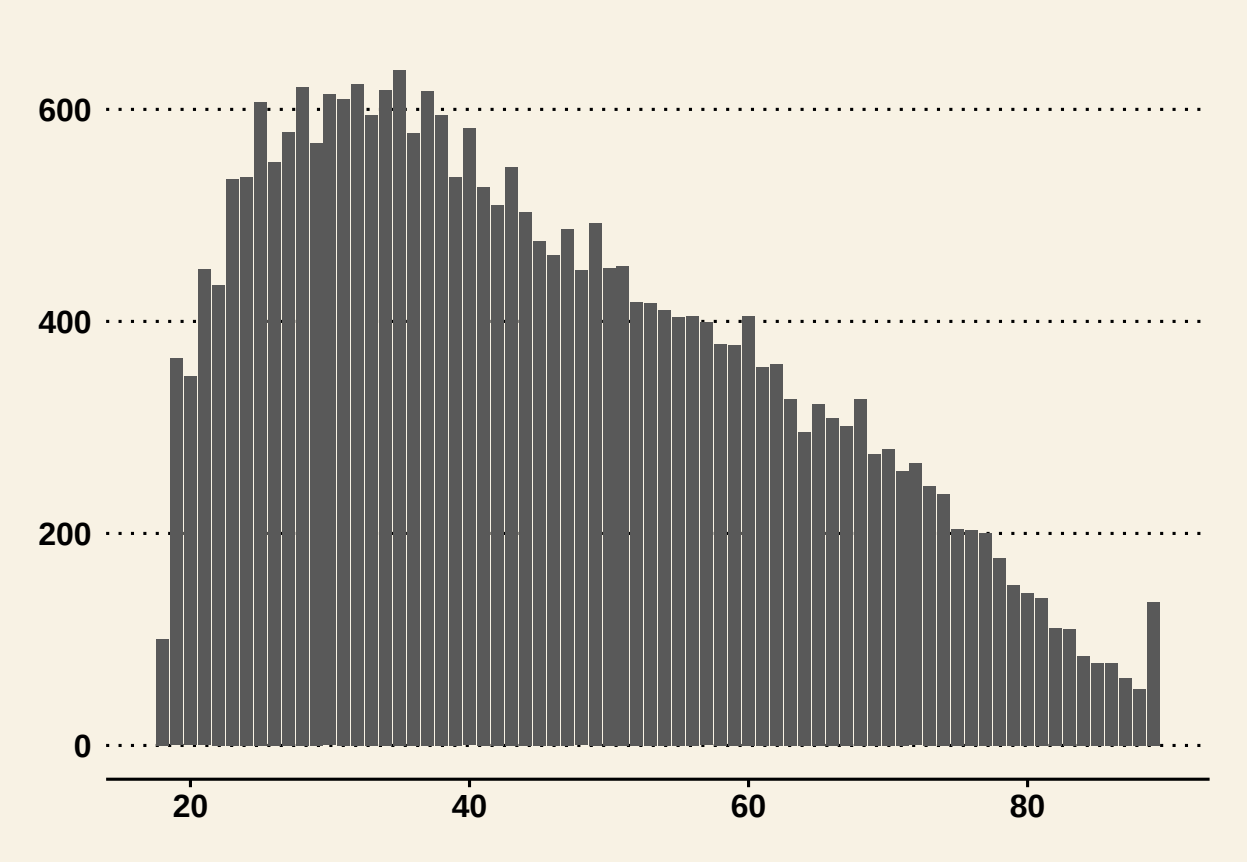
Create two different plots and identify the best-looking plot you can to examine the `age` variable. Save the best looking plot as an appropriately-named PDF.

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




```
ggplot(GSSvocab) +
  aes(x = age) +
  geom_bar(bins = 50)+
  ggthemes::theme_wsj()
```

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




```
plot = ggplot(GSSvocab) +
  aes(x = age) +
  geom_bar(bins = 50)+
  ggthemes::theme_wsj()
```

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

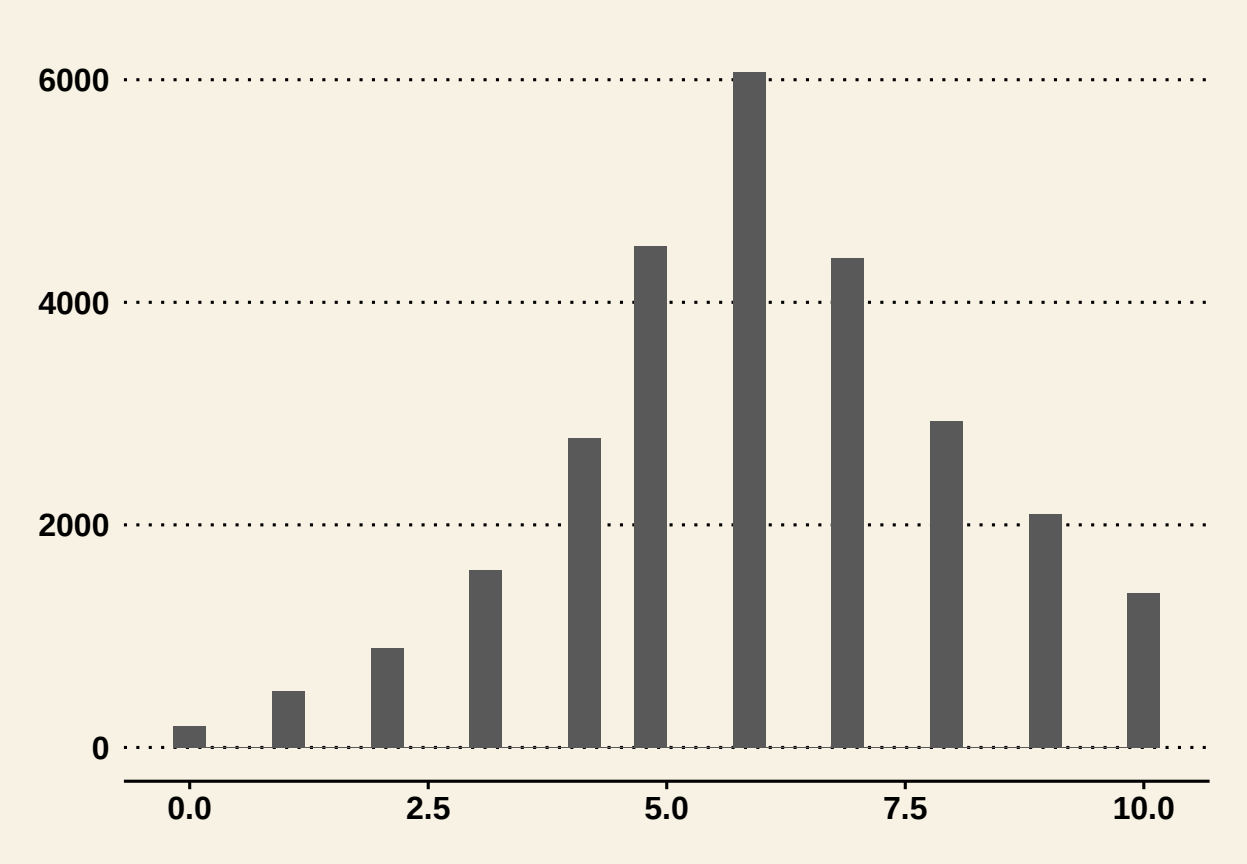
```
ggsave("lab_06_plot.pdf", plot = plot)
```

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

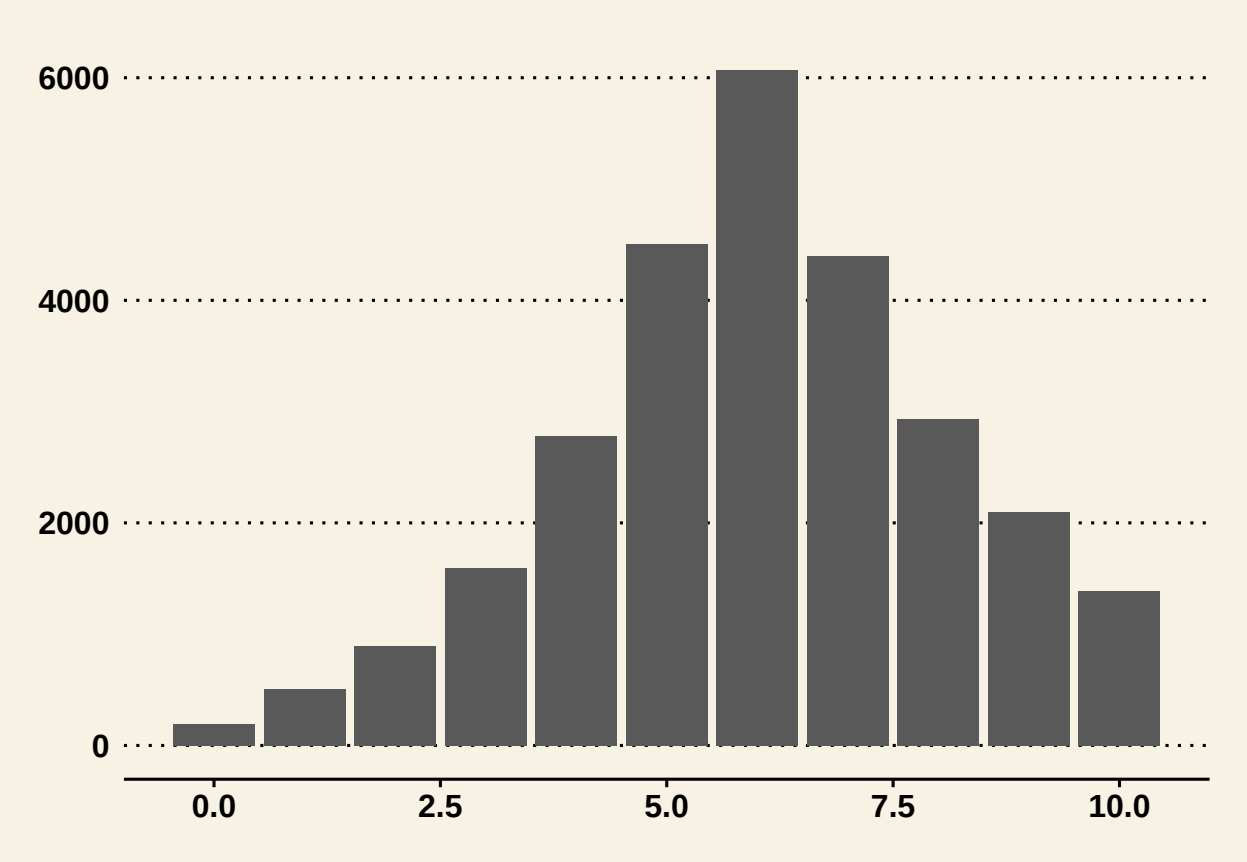
Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

```
# Histogram
ggplot(GSSvocab) +
  aes(x = vocab) +
  geom_histogram() +
  ggthemes::theme_wsj()
```

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

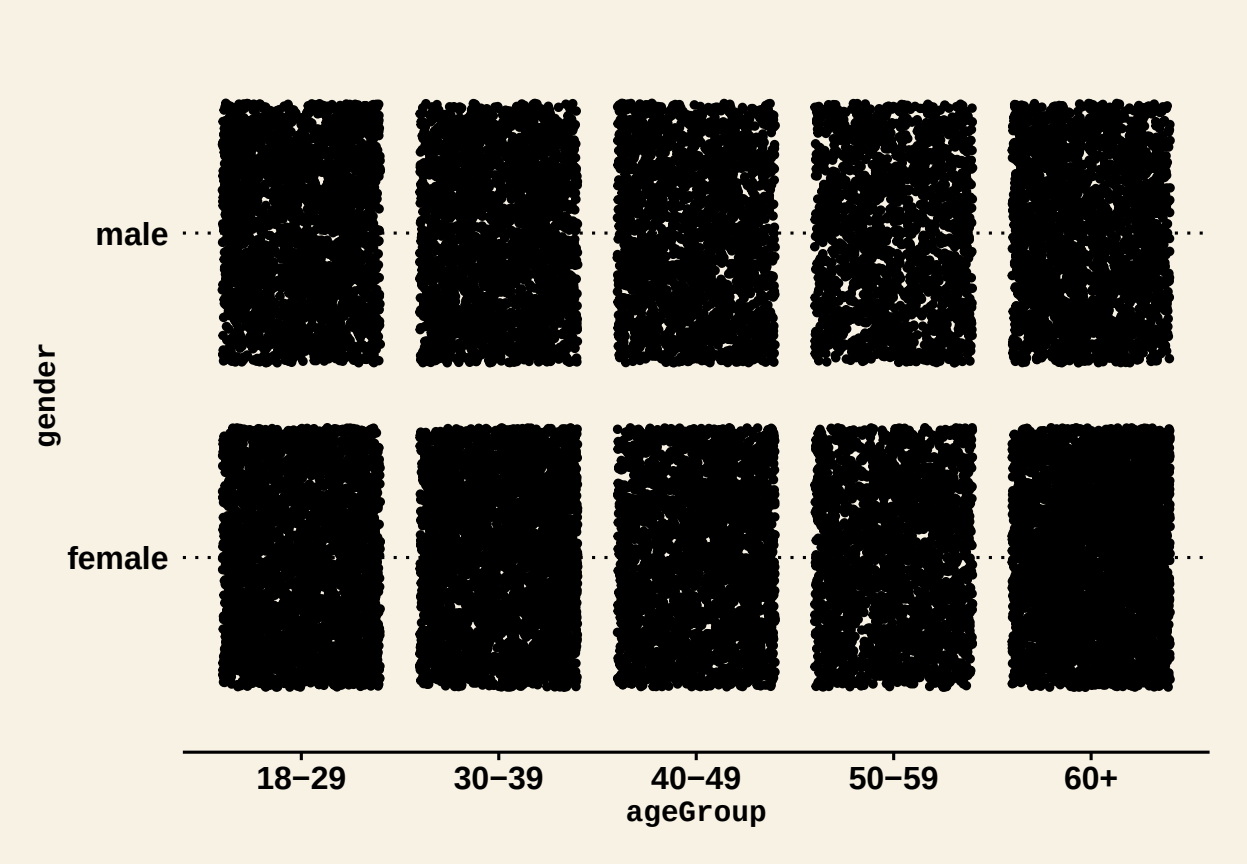



```
# Bar chart
ggplot(GSSvocab) +
  aes(x = vocab) +
  geom_bar() +
  ggthemes::theme_wsj()
```

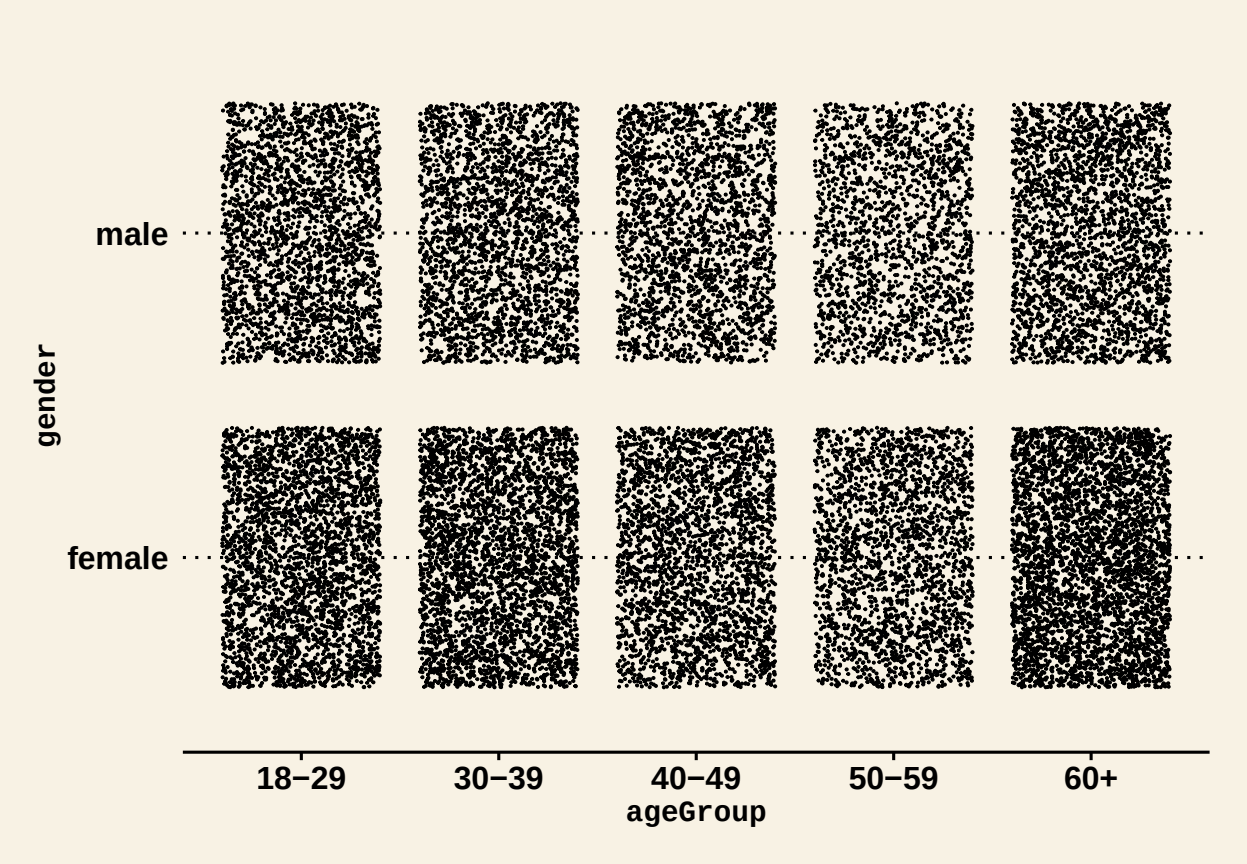


Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

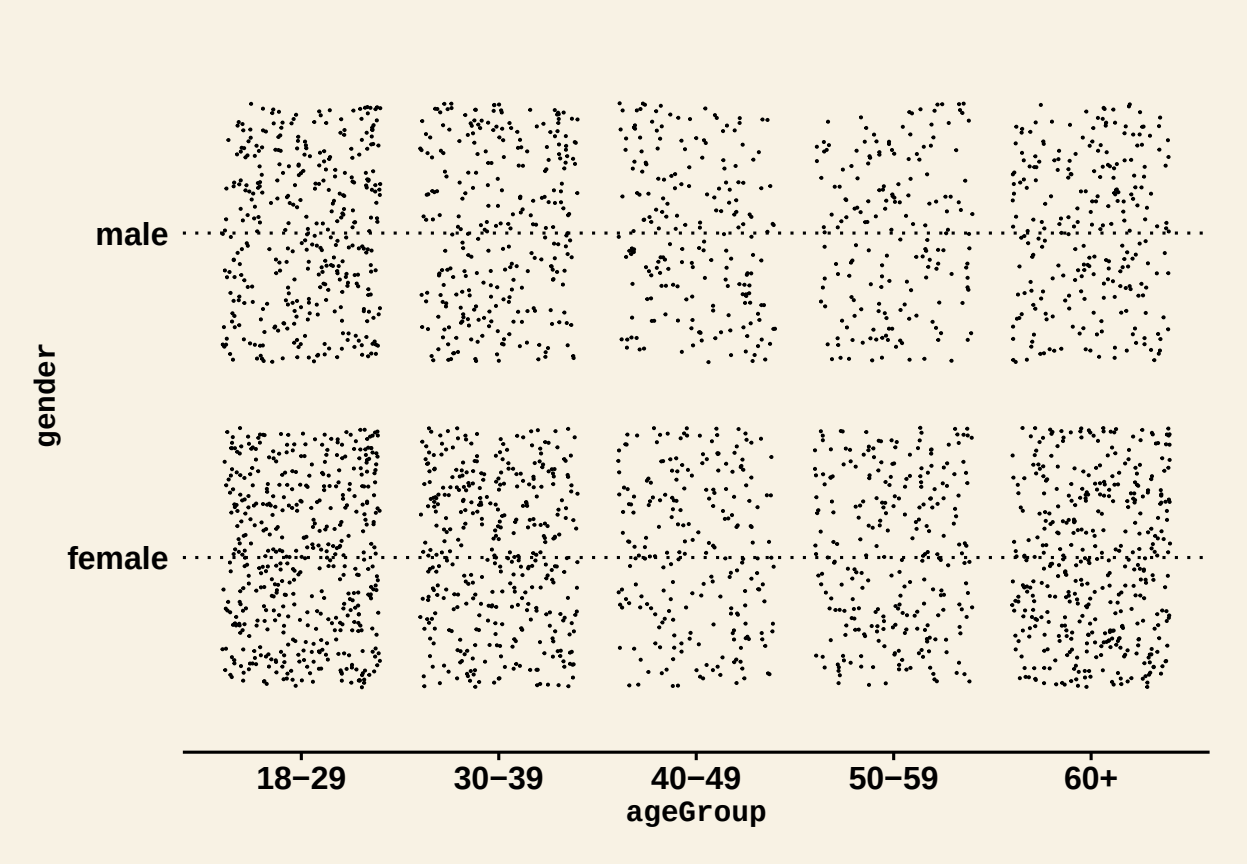
```
ggplot(GSSvocab, aes(x = ageGroup, y = gender)) +
    geom_jitter(size = 1) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```




```
# The dots are very coarse, so we have trouble differentiating the density of different parts of the pl

ggplot(GSSvocab, aes(x = ageGroup, y = gender)) +
    geom_jitter(size = 0.05) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

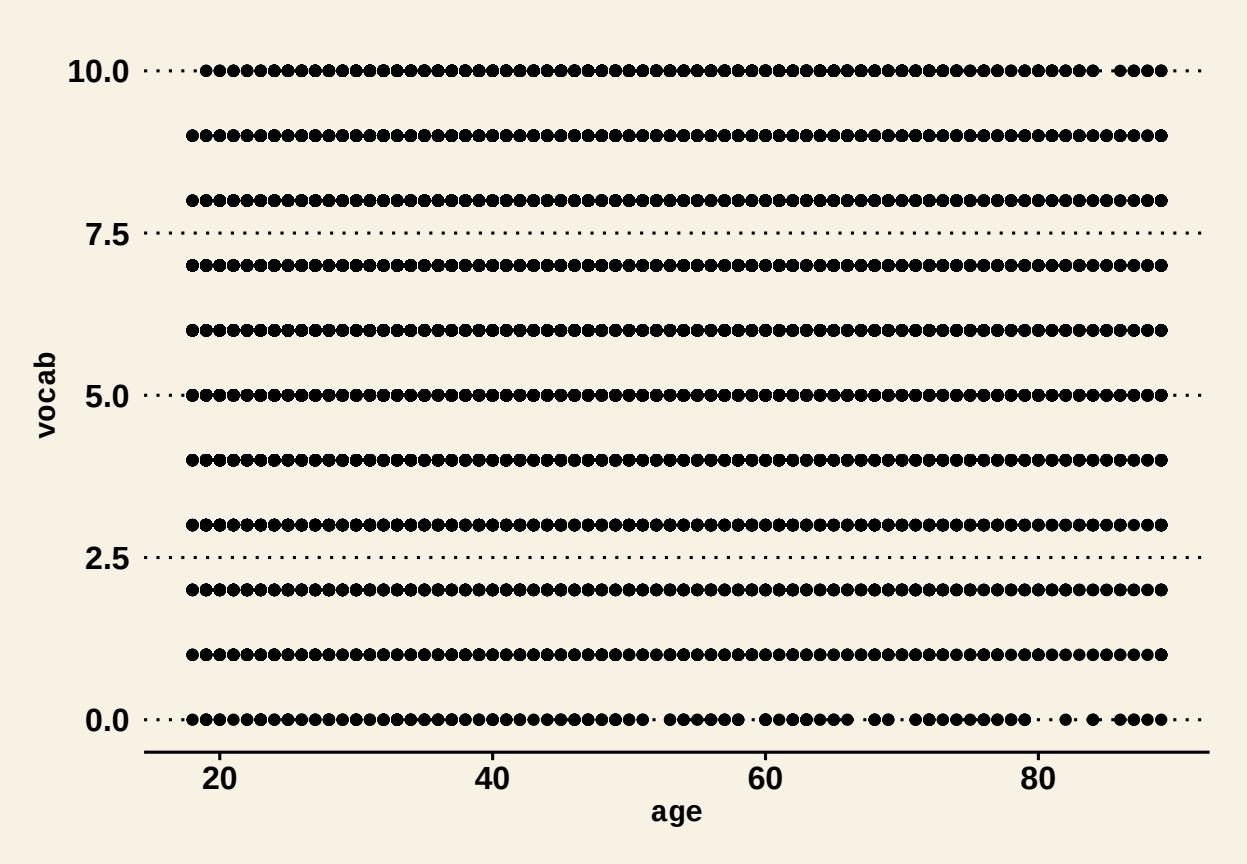



```
# The plot is still too dense, so lets sample the GSSvocab dataset, and then repeat.

ggplot(GSSvocab[1:3000,], aes(x = ageGroup, y = gender)) +
    geom_jitter(size = 0.05) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

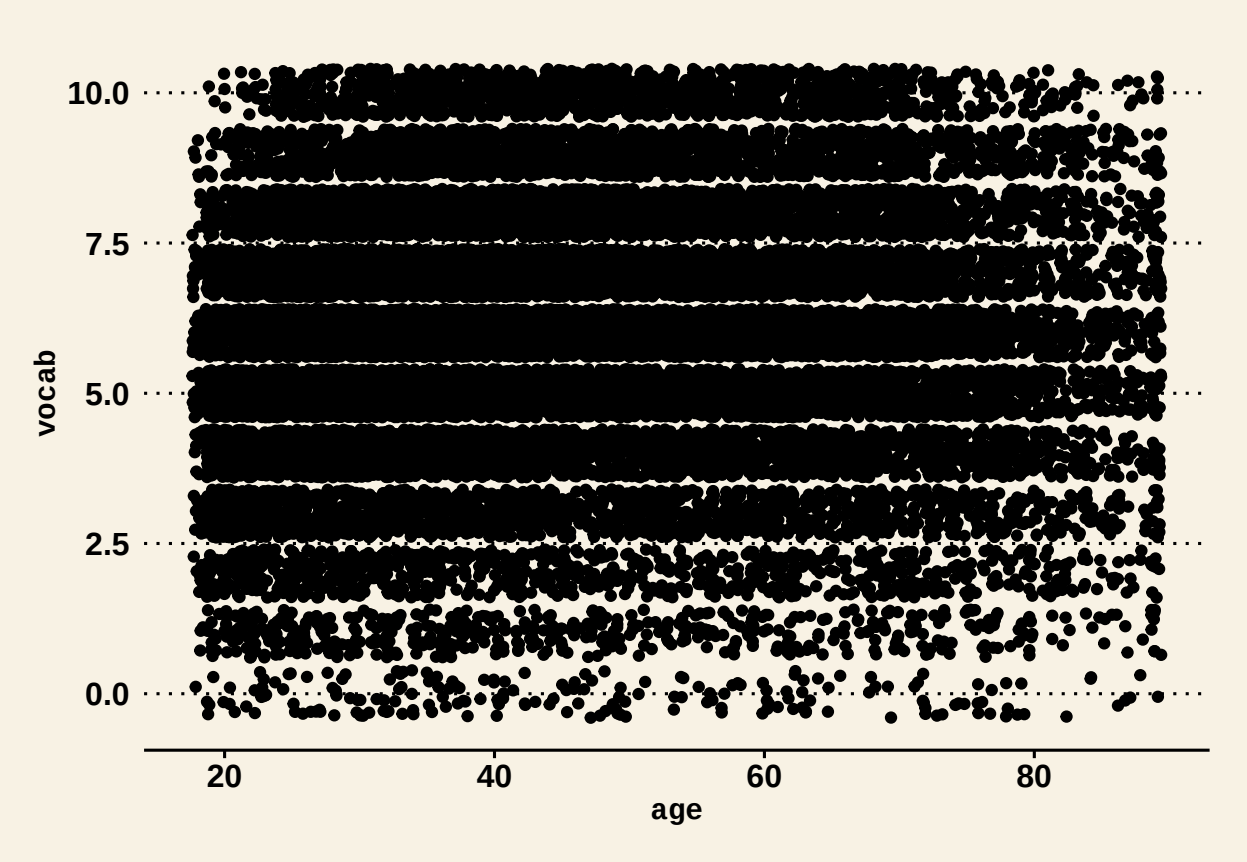


Create the best-looking plot you can to examine the `vocab` variable by `age`. Does there appear to be an association?

```
ggplot(GSSvocab, aes(x = age, y = vocab)) +
  geom_point() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```




```
# This plot does not seem too informative, lets try a jitter plot.

ggplot(GSSvocab, aes(x = age, y = vocab)) +
  geom_jitter() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

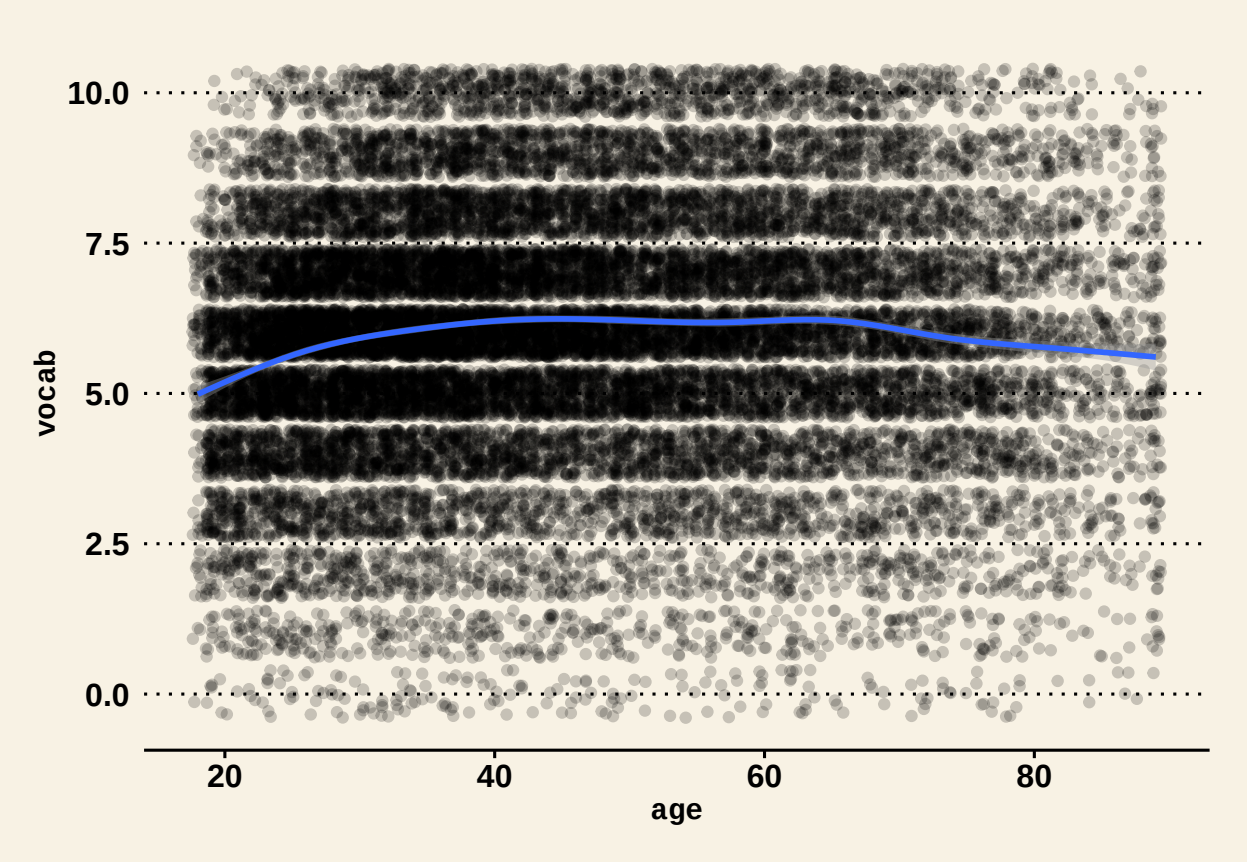



```
# We can kind of see an association, but we can do better.
```

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(alpha = 0.2) +
  geom_smooth() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

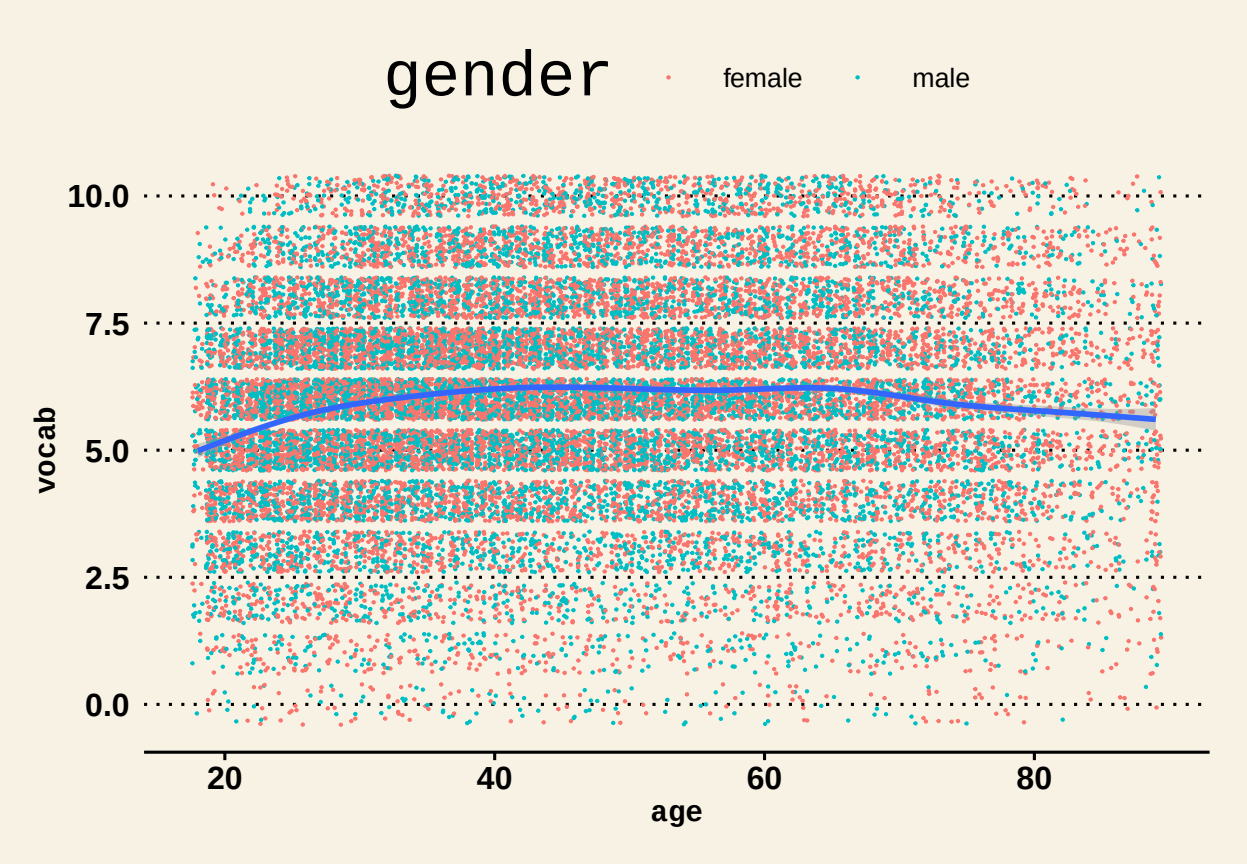
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



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), size = 0.1) +
  geom_smooth() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

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

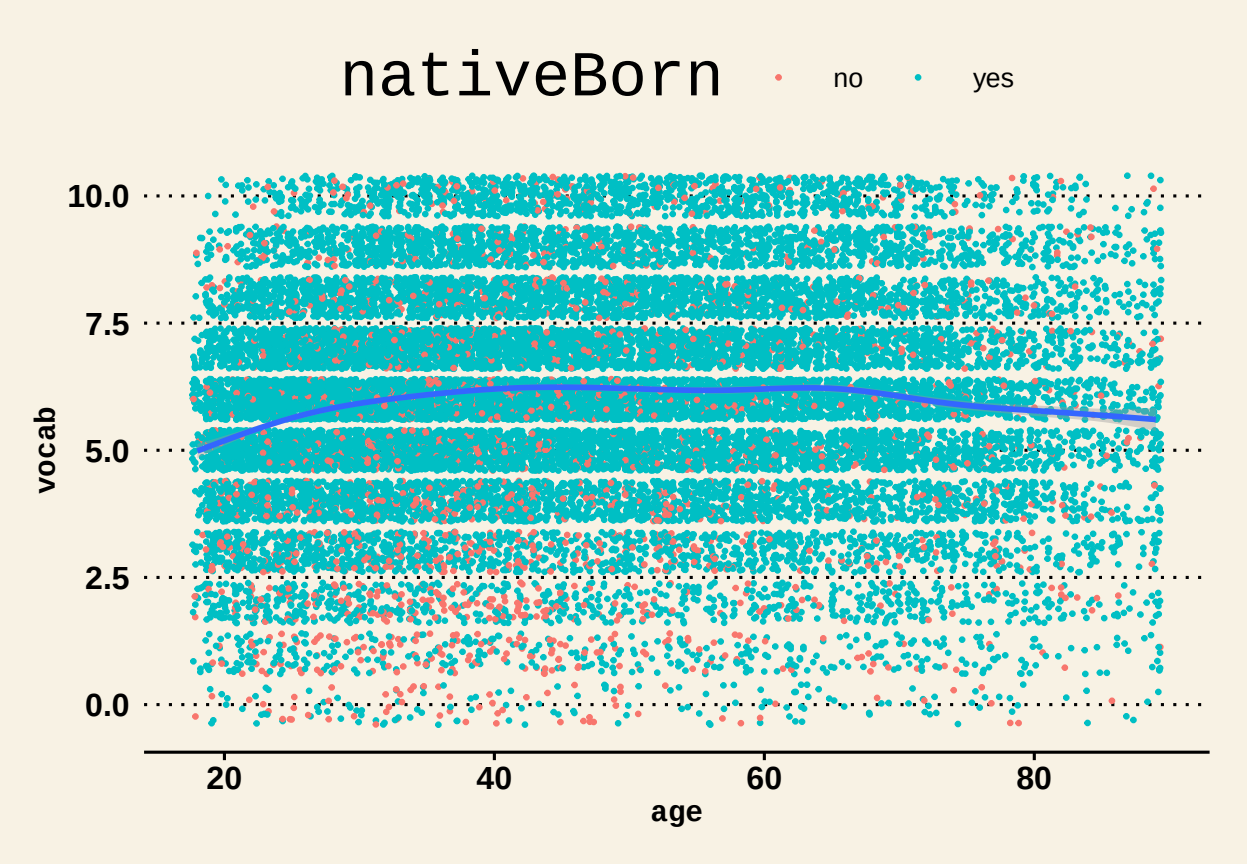



```
# This plot still is not too informative because the data is so varied. We still cannot find a strong a
```

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`?

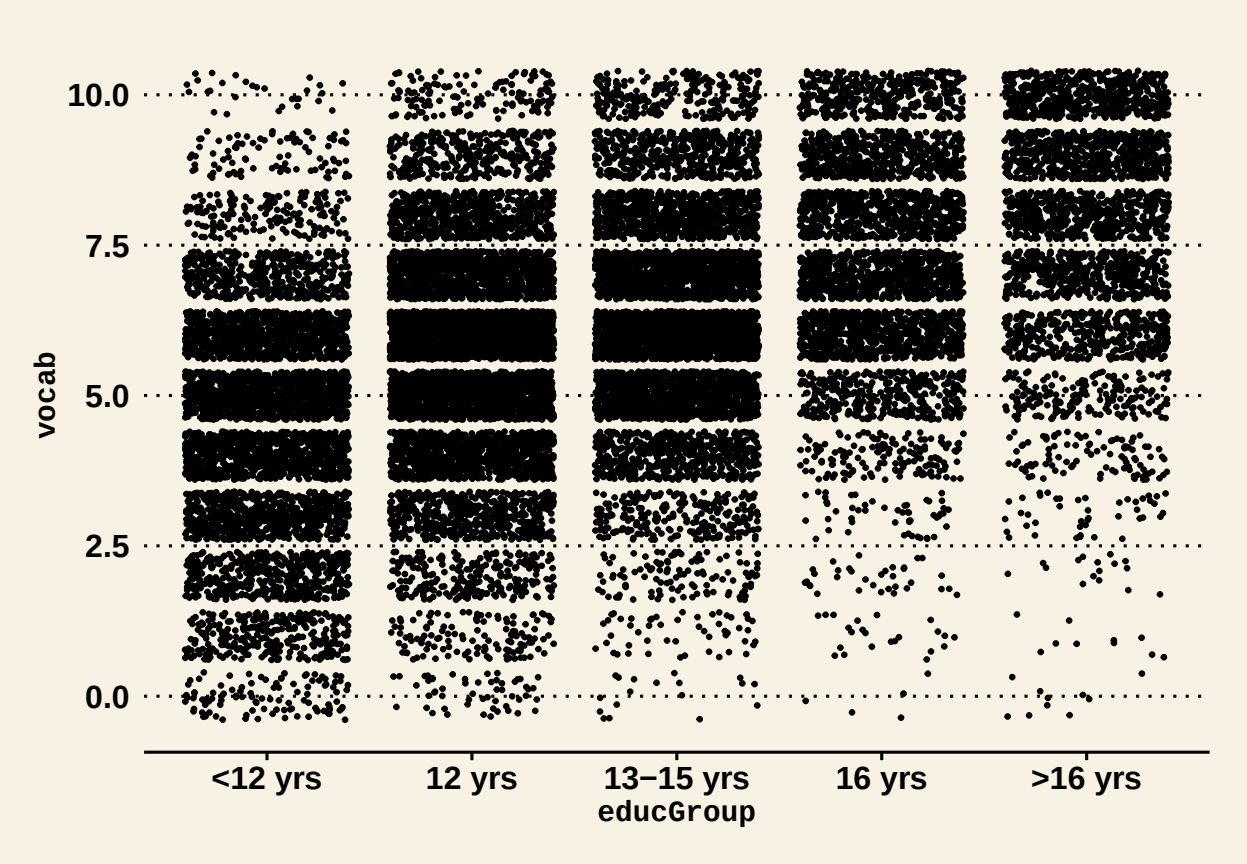
```
ggplot(GSSvocab, aes(x = age, y = vocab)) +
  geom_jitter(aes(col = nativeBorn), size = 0.5) +
  geom_smooth() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```

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

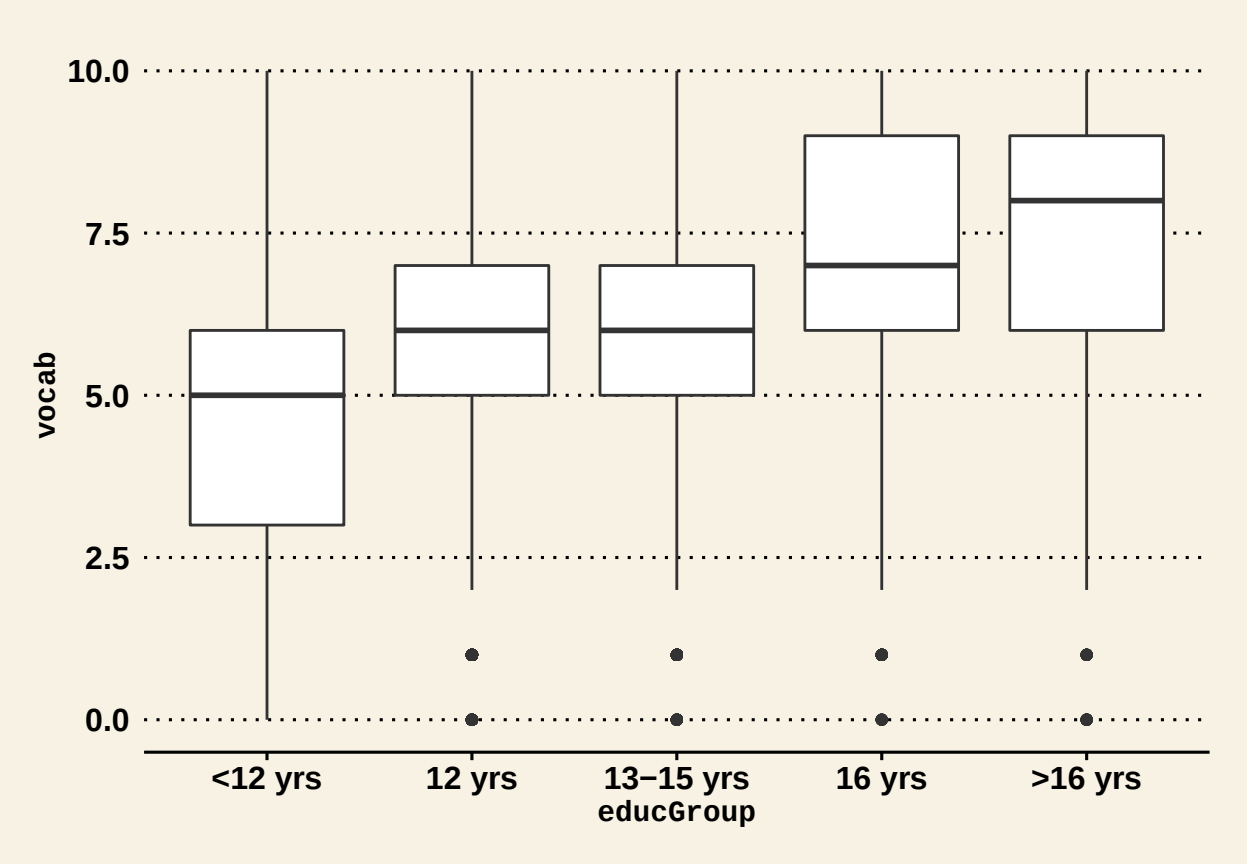


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

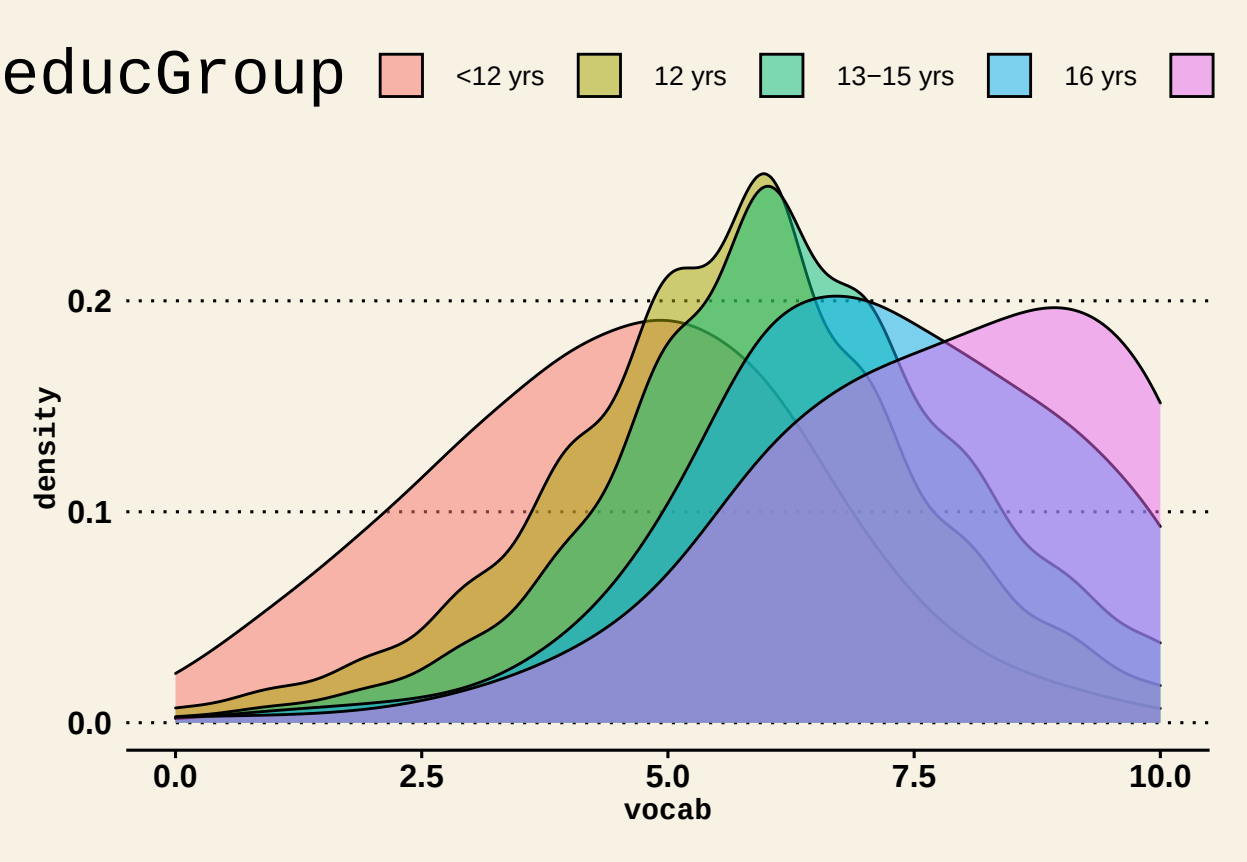
```
# Jitter plot
ggplot(GSSvocab, aes(x = educGroup, y = vocab)) +
  geom_jitter(size = 0.5) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```



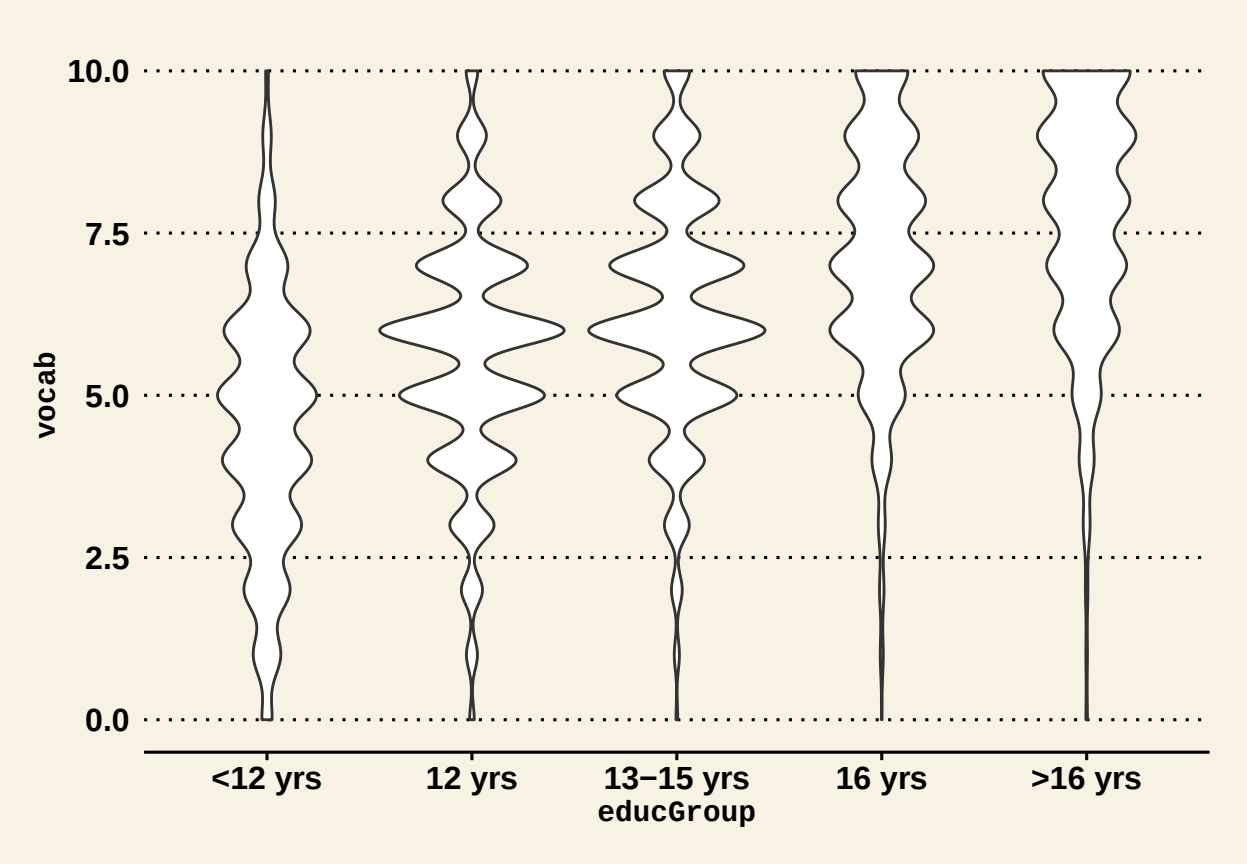

```
# Box plot
ggplot(GSSvocab, aes(x = educGroup, y = vocab)) +
  geom_boxplot() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```




```
# The box plot returns a very informative chart. We can see that those groups with higher education per

#Desnity plot but with colors!
ggplot(GSSvocab, aes(x = vocab)) +
  geom_density(aes(fill = educGroup), adjust = 2, alpha = 0.5) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```



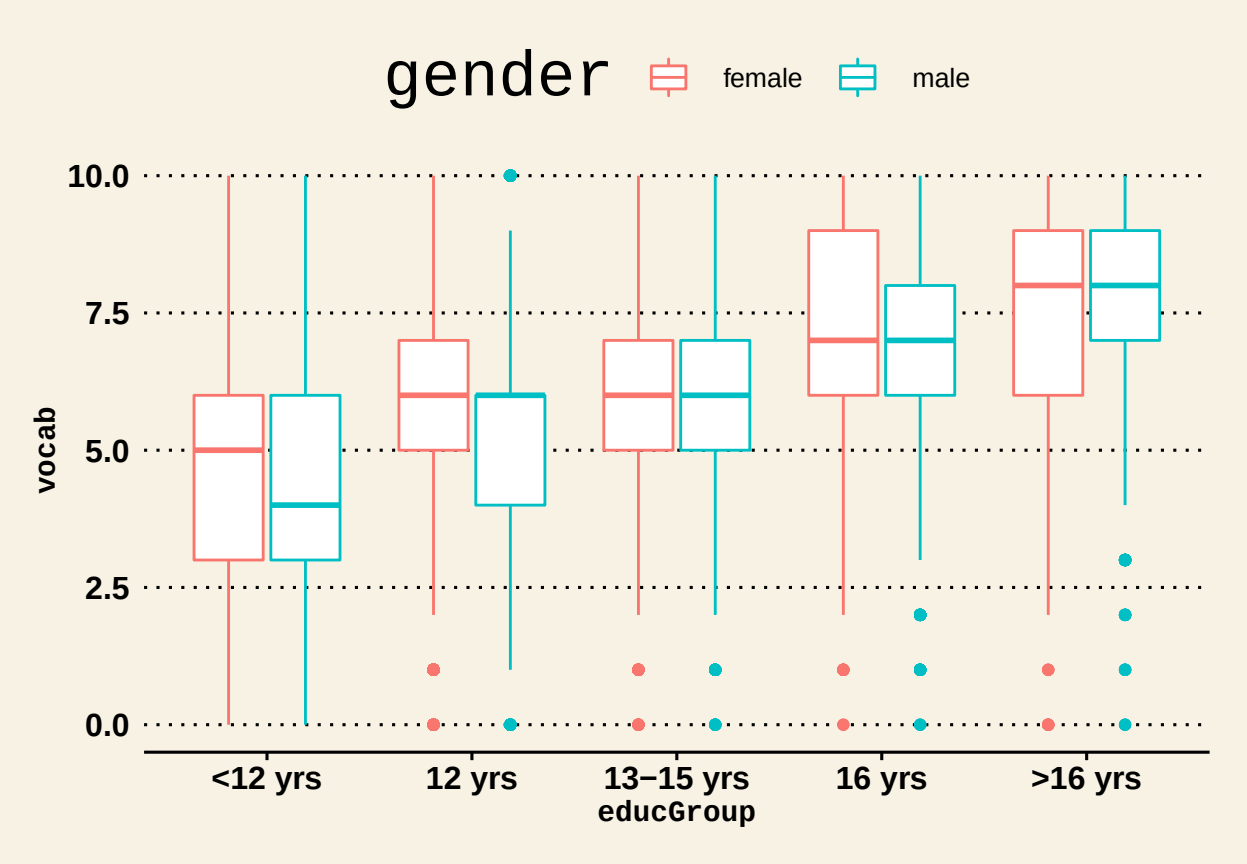

```
# GOOOORGEOUS

# Violin
ggplot(GSSvocab, aes(x = educGroup, y = vocab)) +
  geom_violin() +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```




```
# Not very smooth because the vocab feature is discrete
```

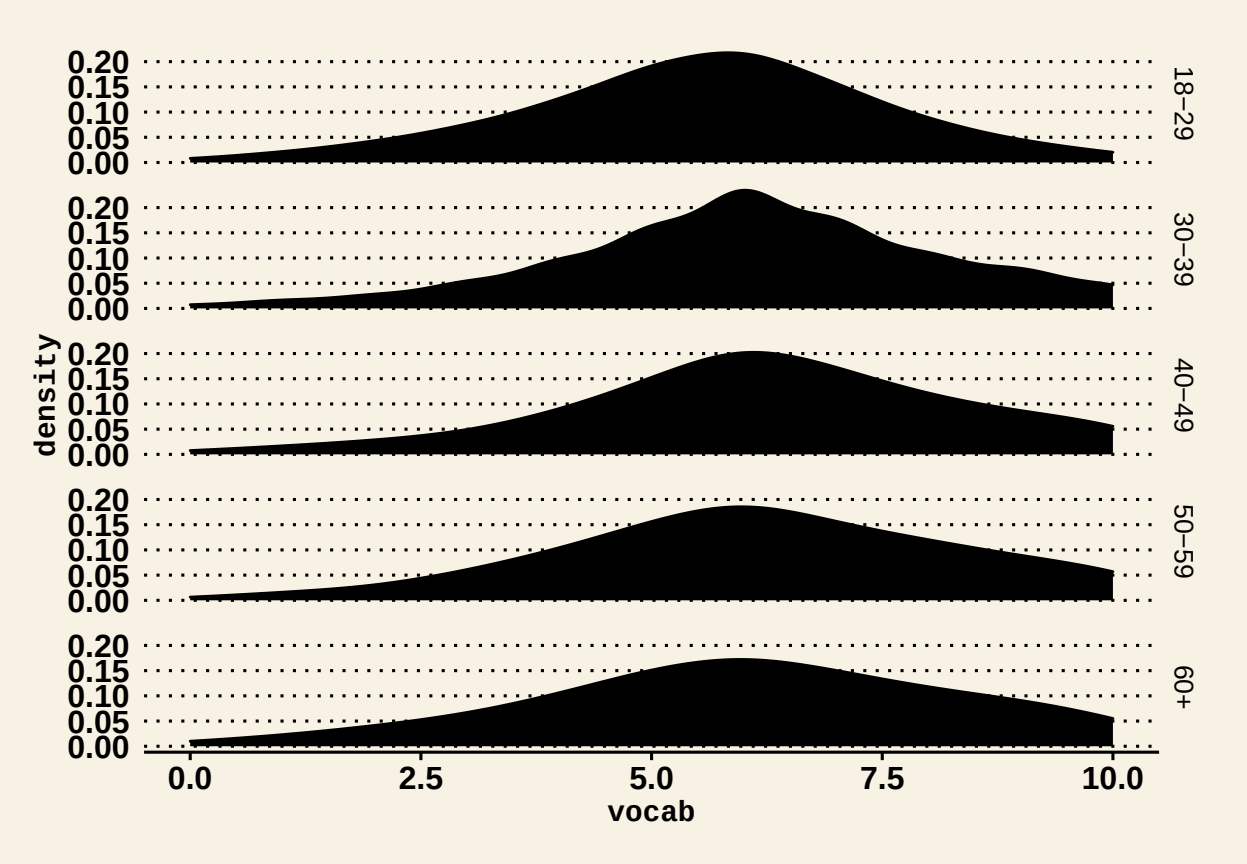
Using the best-looking plot from the previous question, create the best looking overloading with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?

```
ggplot(GSSvocab, aes(x = educGroup, y = vocab)) +
  geom_boxplot(aes(col = gender)) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```



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

```
ggplot(GSSvocab) +
  aes(x=vocab) +
  geom_density(adjust=2, fill= "black") +
  facet_grid(ageGroup~.) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```



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
?adult
```

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 forces 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$education == "Preschool", "<=4th", adult$
adult$education = as.factor(adult$education)
```

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 I do see rare levels! For instance, there is only one indivudual from the Netherlands, only 12 from Honduras, etc. This might be a problem because if we attempt to run an OLS model, then the data might not be full rank.

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?

No, this is not full rank, as per the code below.

```
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 = ":")

adult$occupation = NULL
adult$workclass = NULL

tab_worktype = sort(table(adult$worktype))
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                              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(tab_worktype[tab_worktype < 100]), "Other", adult$work
adult$worktype = as.factor(adult$worktype)
tab_worktype2 = sort(table(adult$worktype))
tab_worktype2
```

```
##
##        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$relationship = as.character(adult$relationship)
adult$marital_status = as.character(adult$marital_status)
adult$relationship_marital_status = paste(adult$marital_status, adult$relationship, sep = ":")
tab_relationship_marital_status = sort(table(adult$relationship_marital_status))

adult$relationship = NULL
adult$marital_status = NULL

tab_relationship_marital_status
```

```
##
##                     Widowed:Own-child                married:Not-in-family
##                                    12                                   14
## Married-spouse-absent:Other-relative              Widowed:Other-relative
##                                    26                                   40
##      Married-spouse-absent:Own-child            Separated:Other-relative
##                                    43                                   53
##                     married:Own-child                Separated:Own-child
##                                    84                                   90
##              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
##              Separated:Not-in-family                Separated:Unmarried
##                                   383                                  413
##                Widowed:Not-in-family         Never-married:Other-relative
##                                   432                                  548
##              Never-married:Unmarried                         married:Wife
##                                   801                                 1406
##                    Divorced:Unmarried              Divorced:Not-in-family
##                                  1535                                 2268
##              Never-married:Own-child           Never-married:Not-in-family
##                                  3929                                 4447
##                       married:Husband
##                                 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_train, family=binomia
}
```

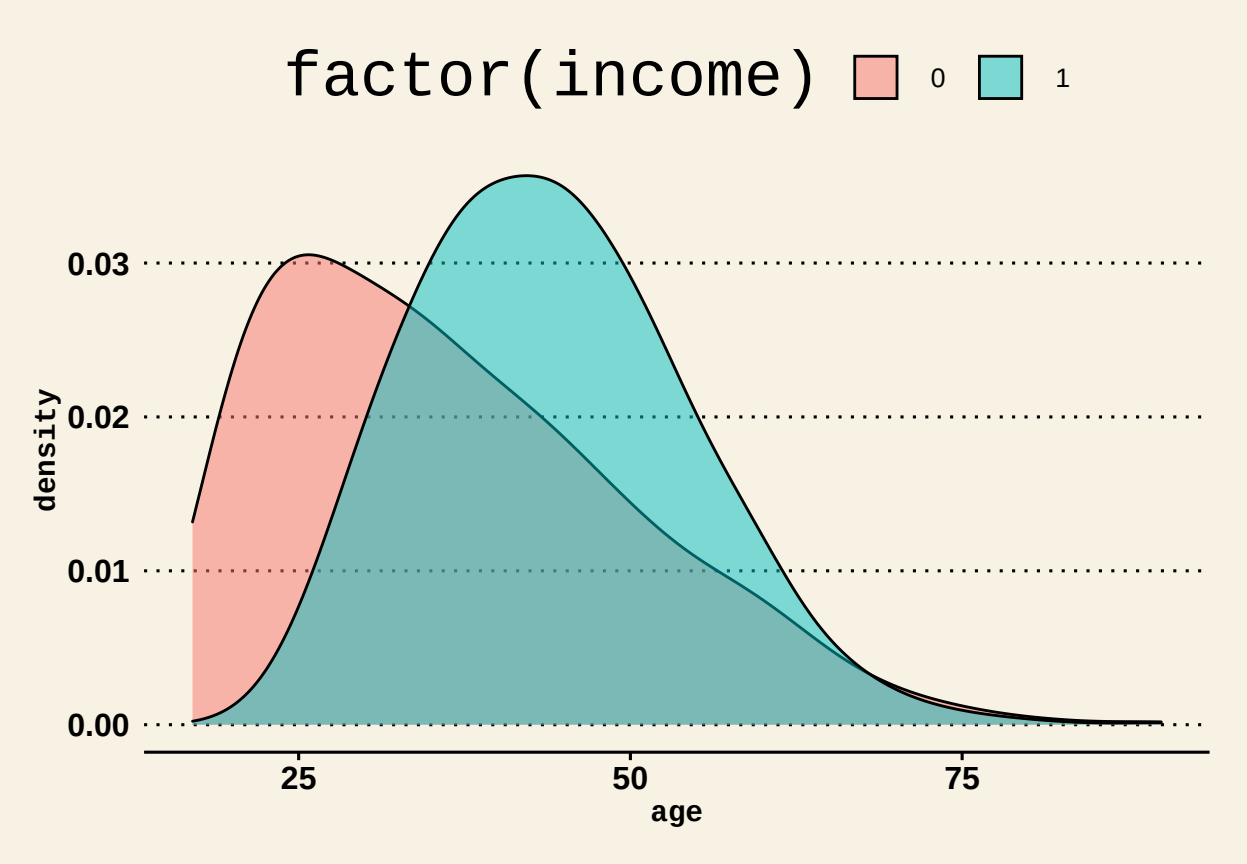
```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## 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 = 0.5) +
  ggthemes::theme_wsj() +
  ggplot2::theme(
    axis.title.x = ggplot2::element_text(size = 11, face = "bold"),
    axis.title.y = ggplot2::element_text(size = 11, face = "bold")
  )
```



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

Earlier we cast the income variable to a binary variable, 1 for values greater than 50k, and 0 for values less than or equal to 50k. It makes sense to see the mode age of the individuals with income greater than 50k be of a greater than the mode age of the individuals with income less than 50k. Observe how the curve of individuals with values less than 50k decreases at around age 25, this is likely due to the population getting better paying careers at this age.

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: 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] -0.1033971
##
## $`probit-vanilla`
## [1] -0.1034124
##
## $`cloglog-vanilla`
## [1] -0.1042804
##
## $`cauchit-vanilla`
## [1] -0.1049043
##
## $`logit-age_interactions`
## [1] -0.09971602
##
## $`probit-age_interactions`
## [1] -0.09969004
##
## $`cloglog-age_interactions`
## [1] -0.1004294
##
## $`cauchit-age_interactions`
## [1] -0.1004961
```

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] -0.102545
##
## $`probit-vanilla`
## [1] -0.1027367
##
## $`cloglog-vanilla`
## [1] -0.1040528
##
## $`cauchit-vanilla`
## [1] -0.1045966
##
## $`logit-age_interactions`
## [1] -0.1008248
##
## $`probit-age_interactions`
## [1] -0.1008646
##
## $`cloglog-age_interactions`
## [1] -0.1018692
##
## $`cauchit-age_interactions`
## [1] -0.1037456
```

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

For in-sample metrics, the vanilla logit and probit models performed just as well. This conclusion goes for age-interactions. For out of sample metrics, the logit and probit models again performed just as well, and this conclusion goes for age-interactions as well. Its important to note that the logit models outperformed the probit models by a relatively very small margin in all cases.

What is wrong with this model selection procedure? There are a few things wrong. The train-test split was done pseudo-randomly, and it would be better practice to adopt a better model selection technique, such as nested resampling.

#TO-DO

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

```
n = nrow(adult)
K = 5
test_indices = sample(1 : n, size = n * 1 / K)
master_train_indices = setdiff(1 : n, test_indices) ##overall train
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, famil
}
```

```
## 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

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

```
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)
briers
```

```
## $`logit-vanilla`
## [1] -0.103504
##
## $`probit-vanilla`
## [1] -0.1038314
##
## $`cloglog-vanilla`
## [1] -0.1048344
##
```

```
## $`cauchit-vanilla`
## [1] -0.1810345
##
## $`logit-age_interactions`
## [1] -0.1039674
##
## $`probit-age_interactions`
## [1] -0.1695955
##
## $`cloglog-age_interactions`
## [1] -0.104636
##
## $`cauchit-age_interactions`
## [1] -0.1065006
```

```
which_final = which.max(briers)
which_final
```

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

```
g_final = glm(income ~., data = 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_test)
```

```
## [1] -0.255305
```