Lab 6

Jacob Minkin

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
```

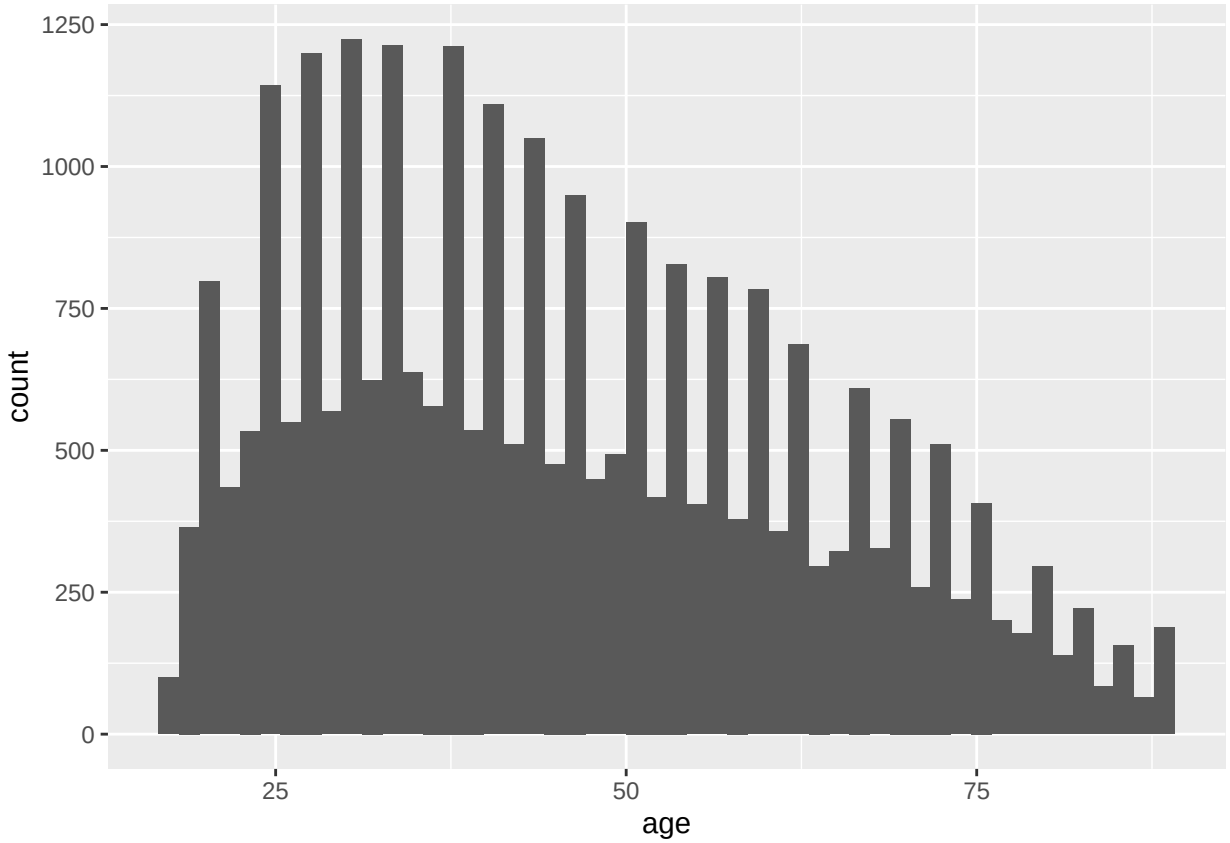
```
## starting httpd help server ... done
```

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?

In this data frame there are 8 variables. Year- discrete. gender-binary. Nativeborn-binary. ageGroup-discrete. educGroup - discrete. vocab- discrete. age - discrete. educ - discrete.

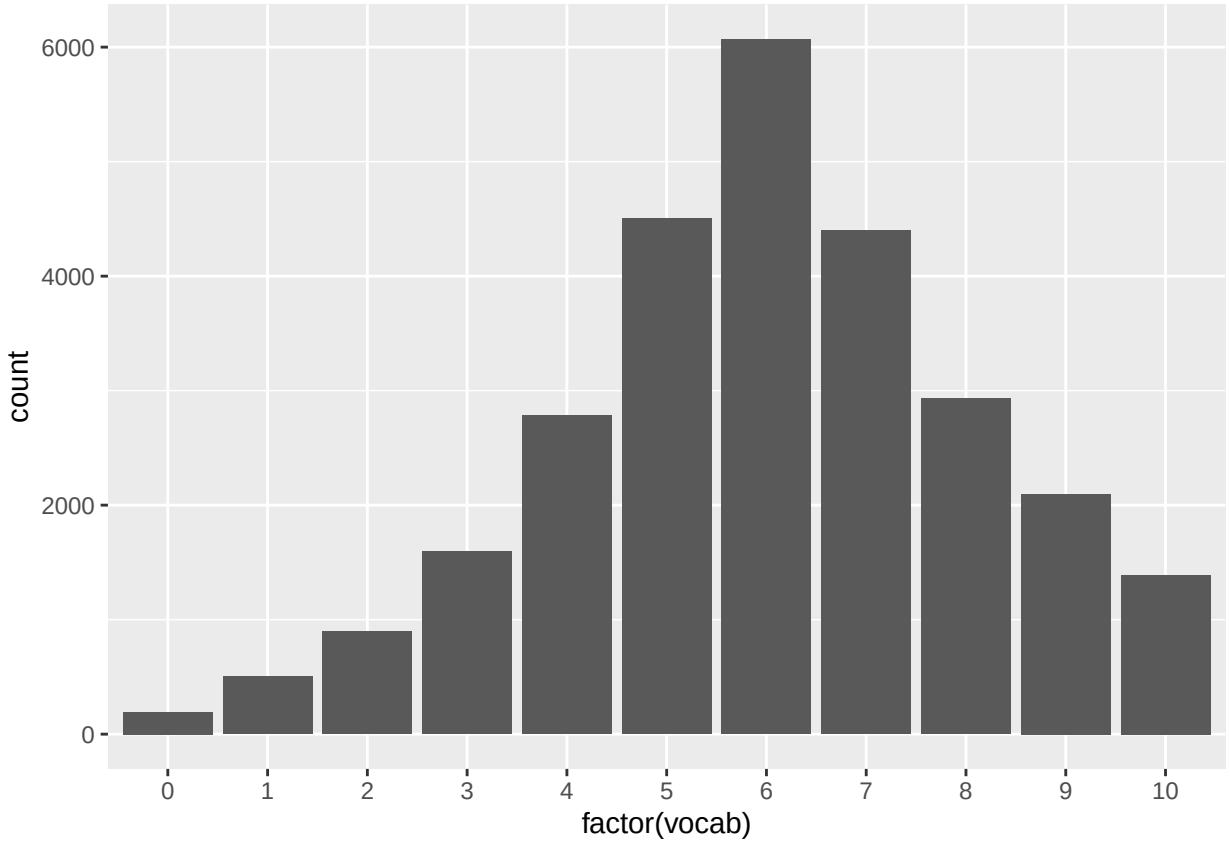
Create two different plots and identify the best-looking plot you can to examine the `age` variable. Save the best looking plot as an appropriately-named PDF.

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



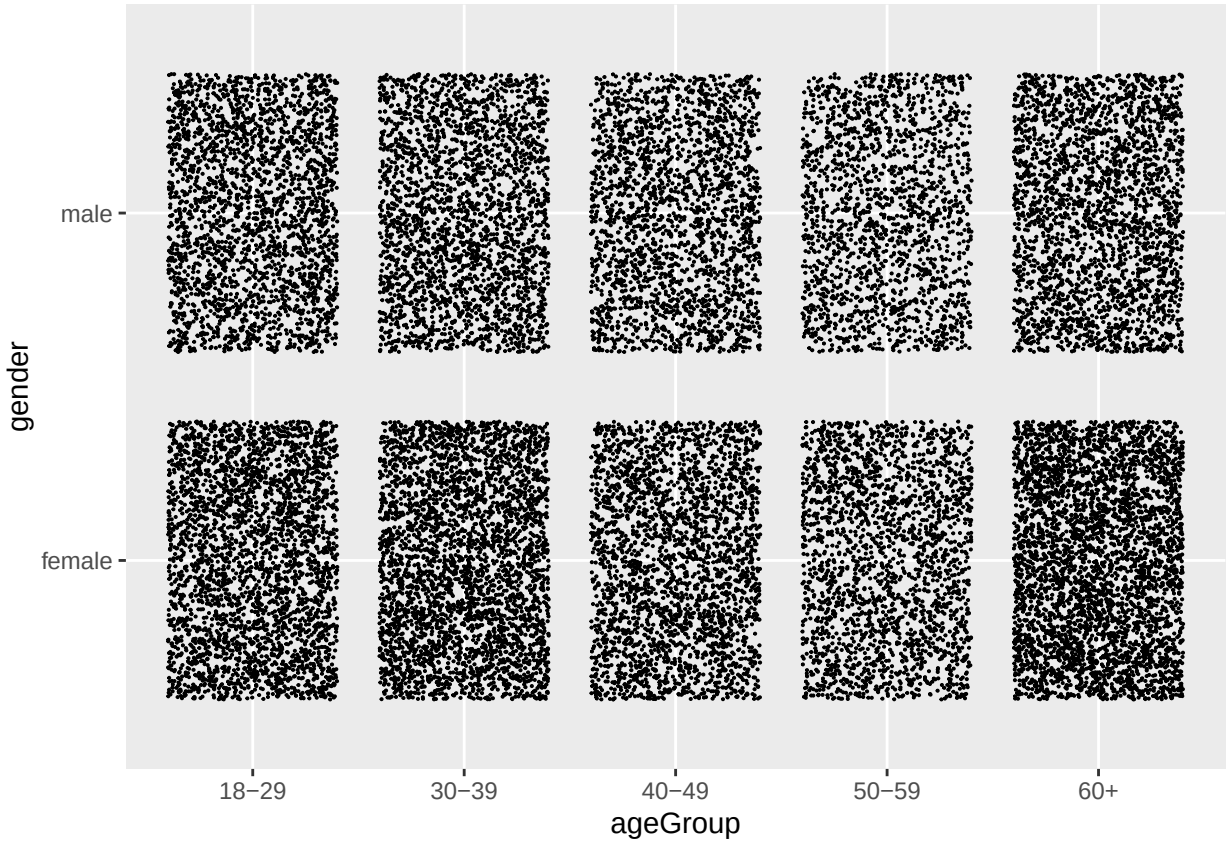
Create two different plots and identify the best looking plot you can to examine the `vocab` variable. Save the best looking plot as an appropriately-named PDF.

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



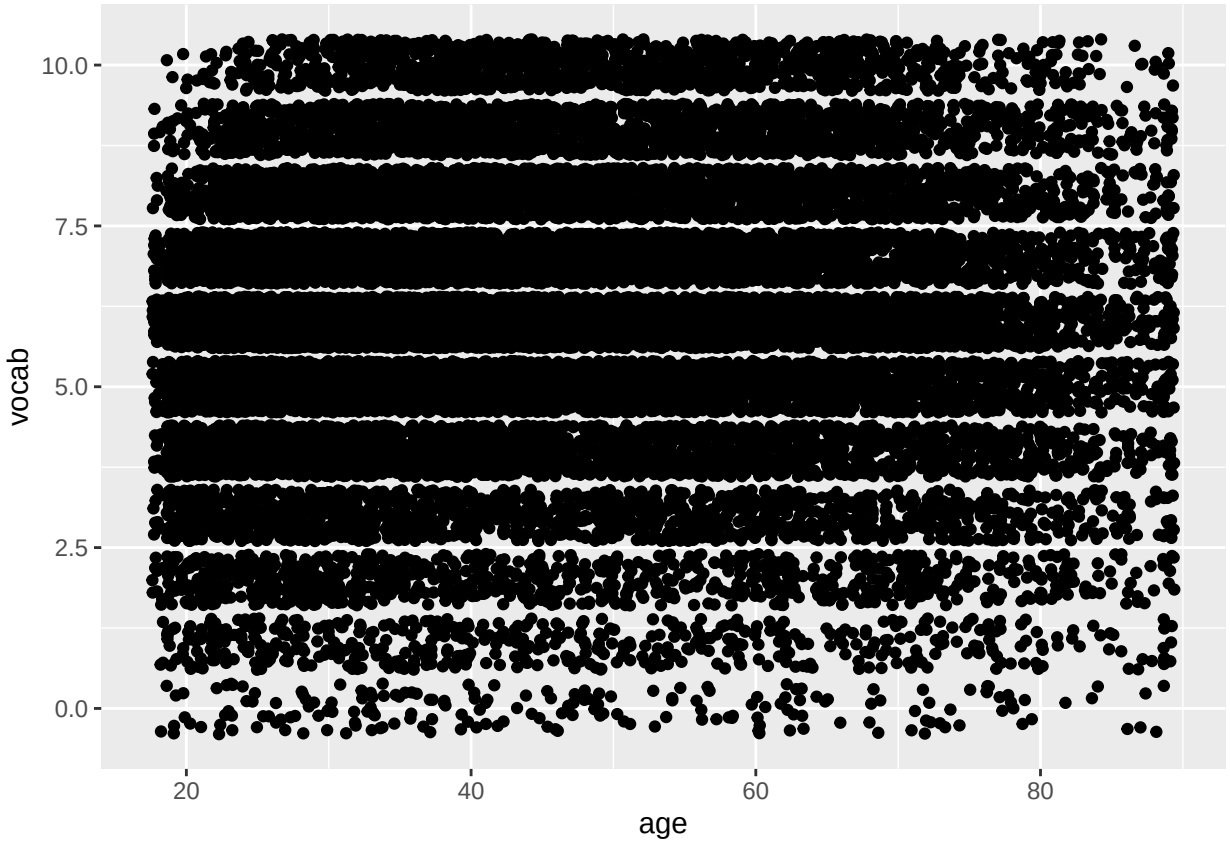
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```
ggplot(GSSvocab) +
  aes(x=ageGroup,y = gender) +
  geom_jitter(size=.05)
```



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

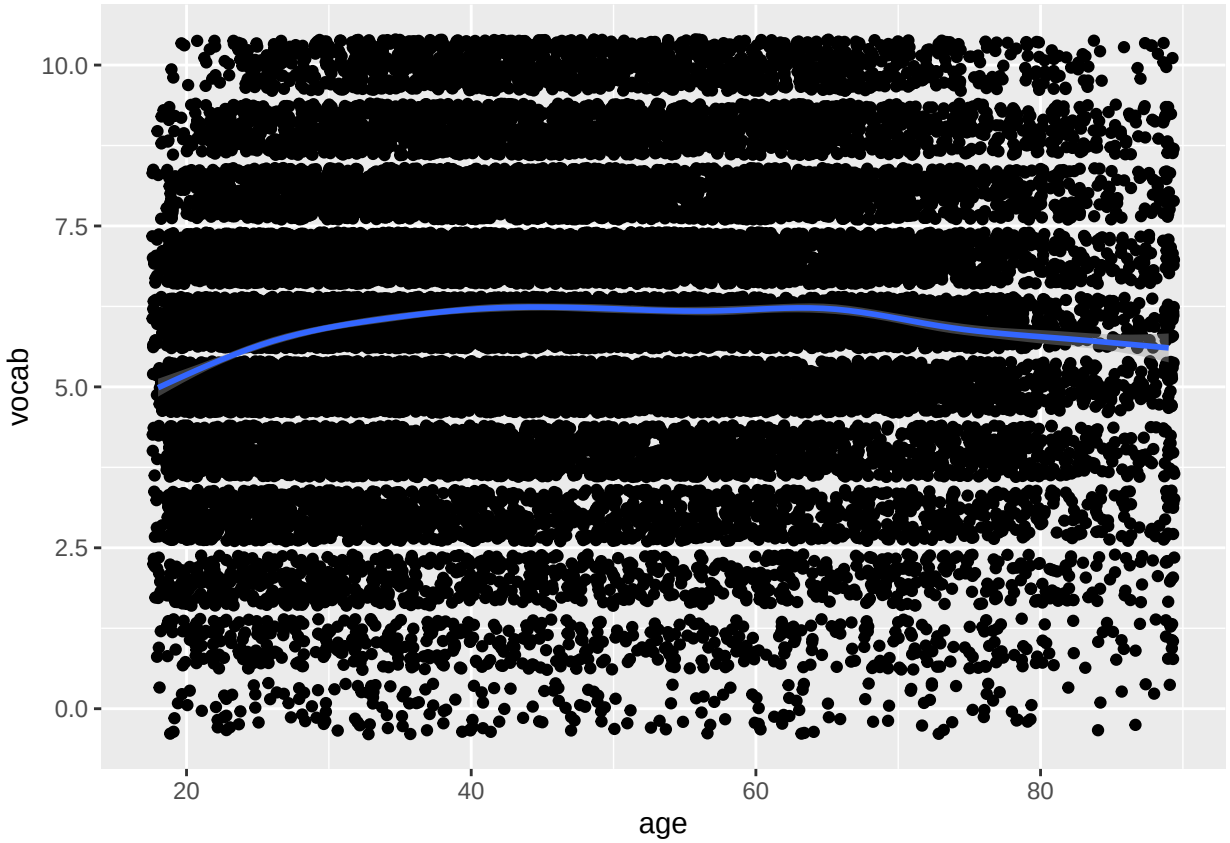
```
ggplot(GSSvocab) +
  aes(x=age,y = vocab) +
  geom_jitter()
```



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

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

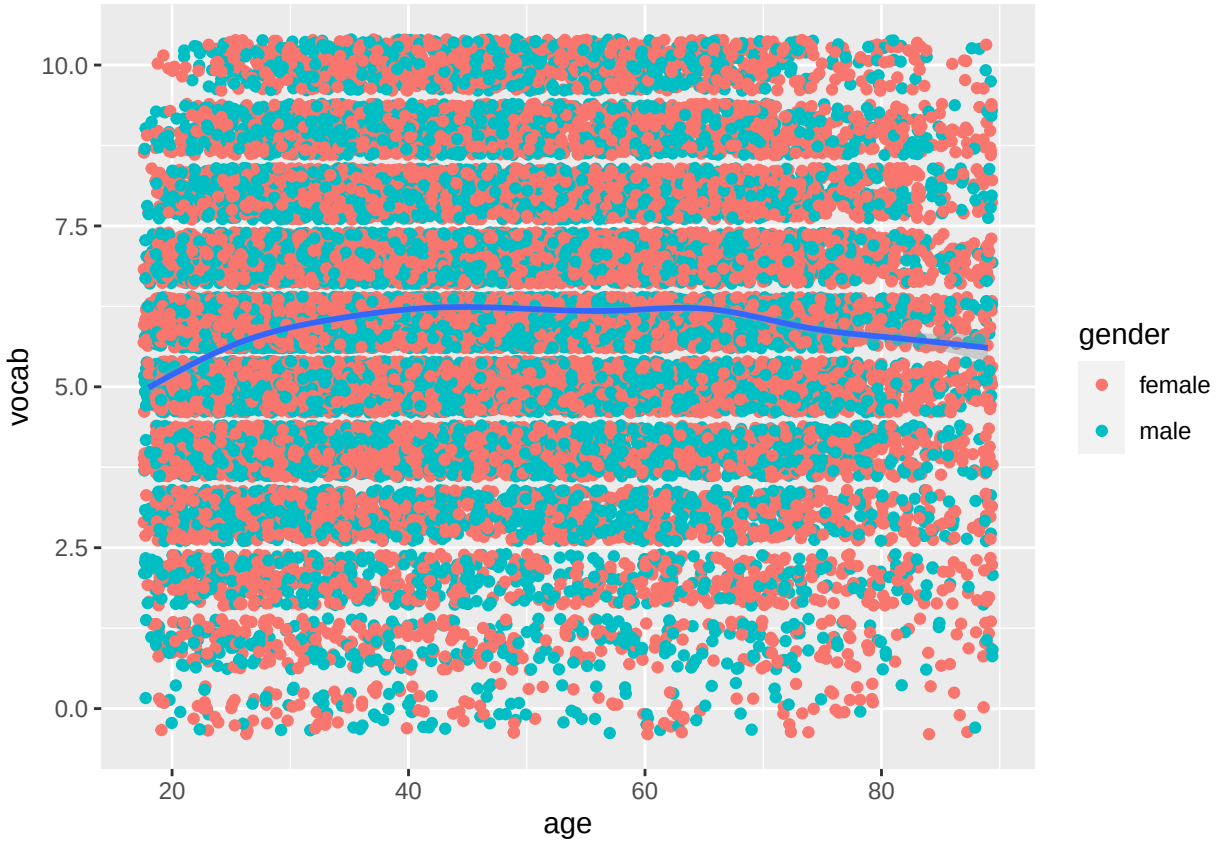
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



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

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

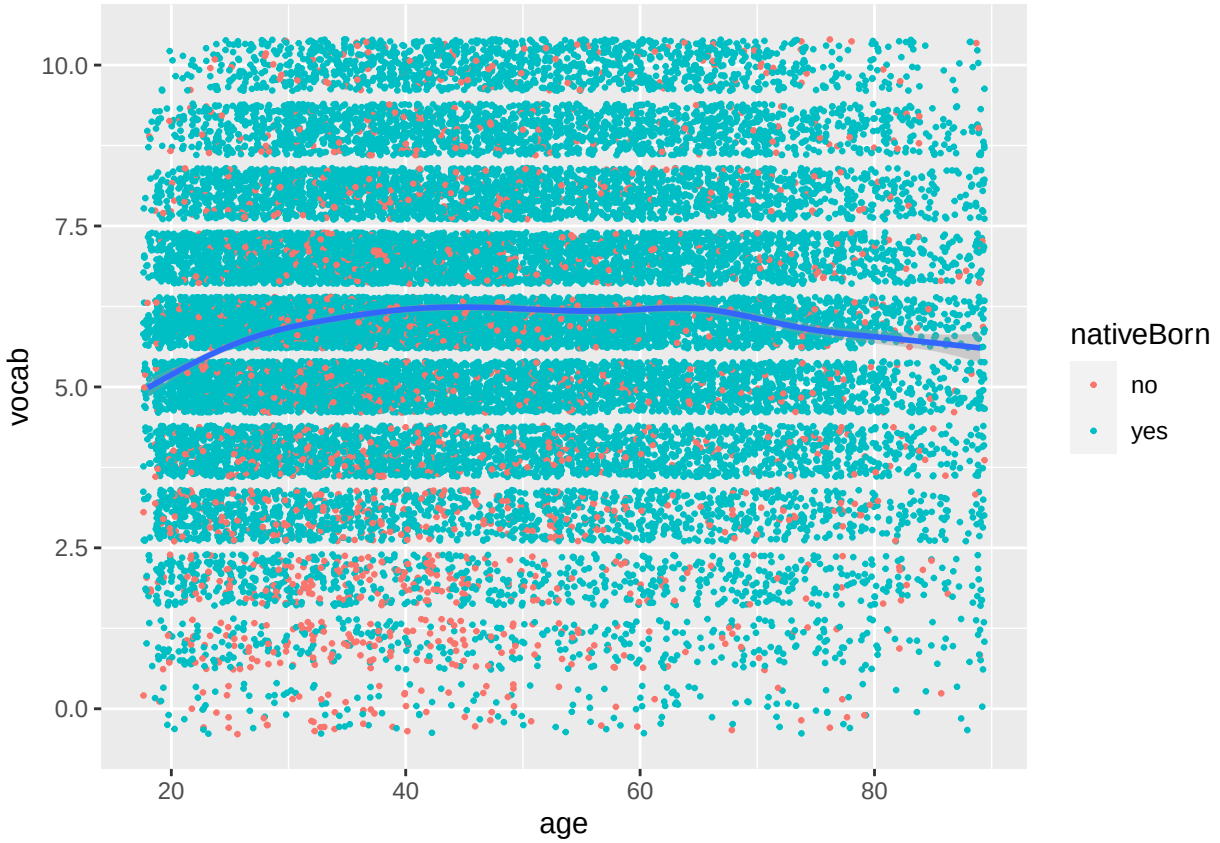
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



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`?

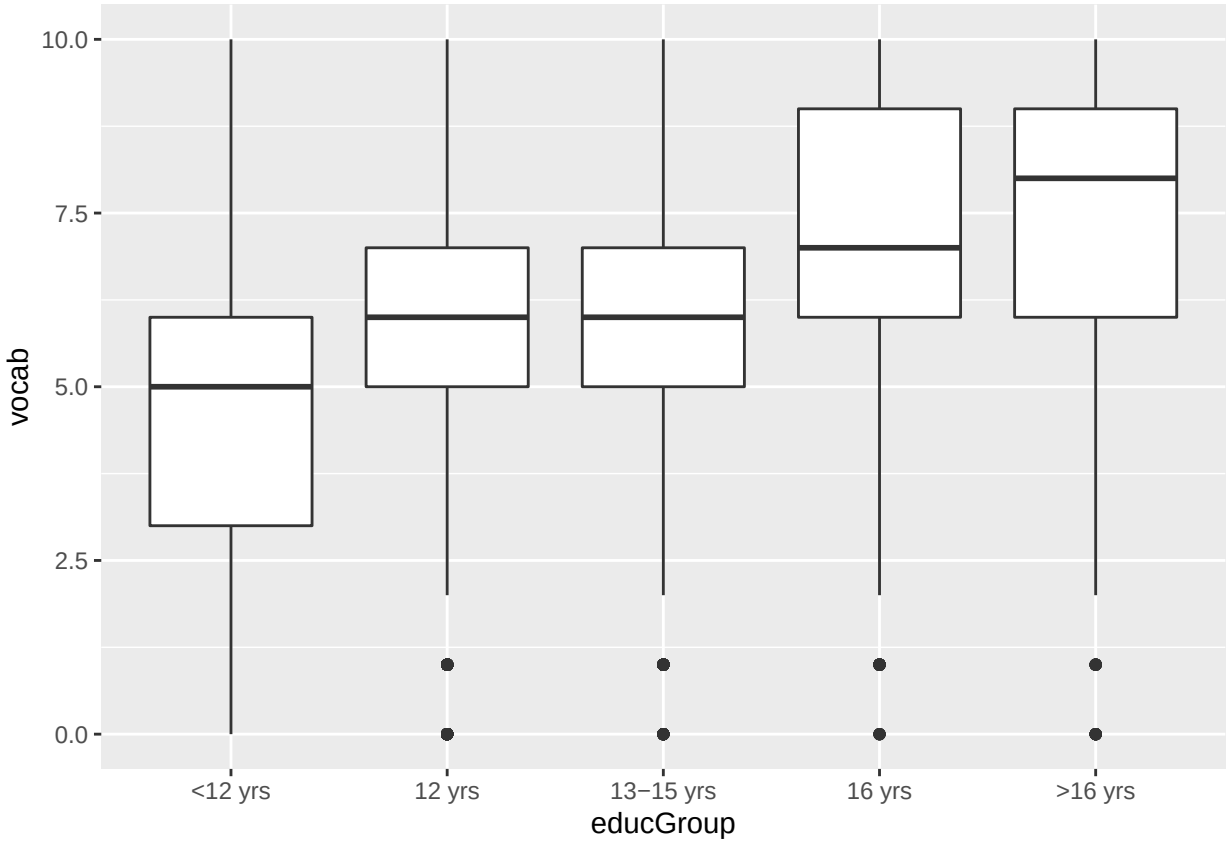
```
ggplot(GSSvocab) +
  aes(x=age,y = vocab) +
  geom_jitter(aes(col = nativeBorn), size=0.5) +
  geom_smooth()
```

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

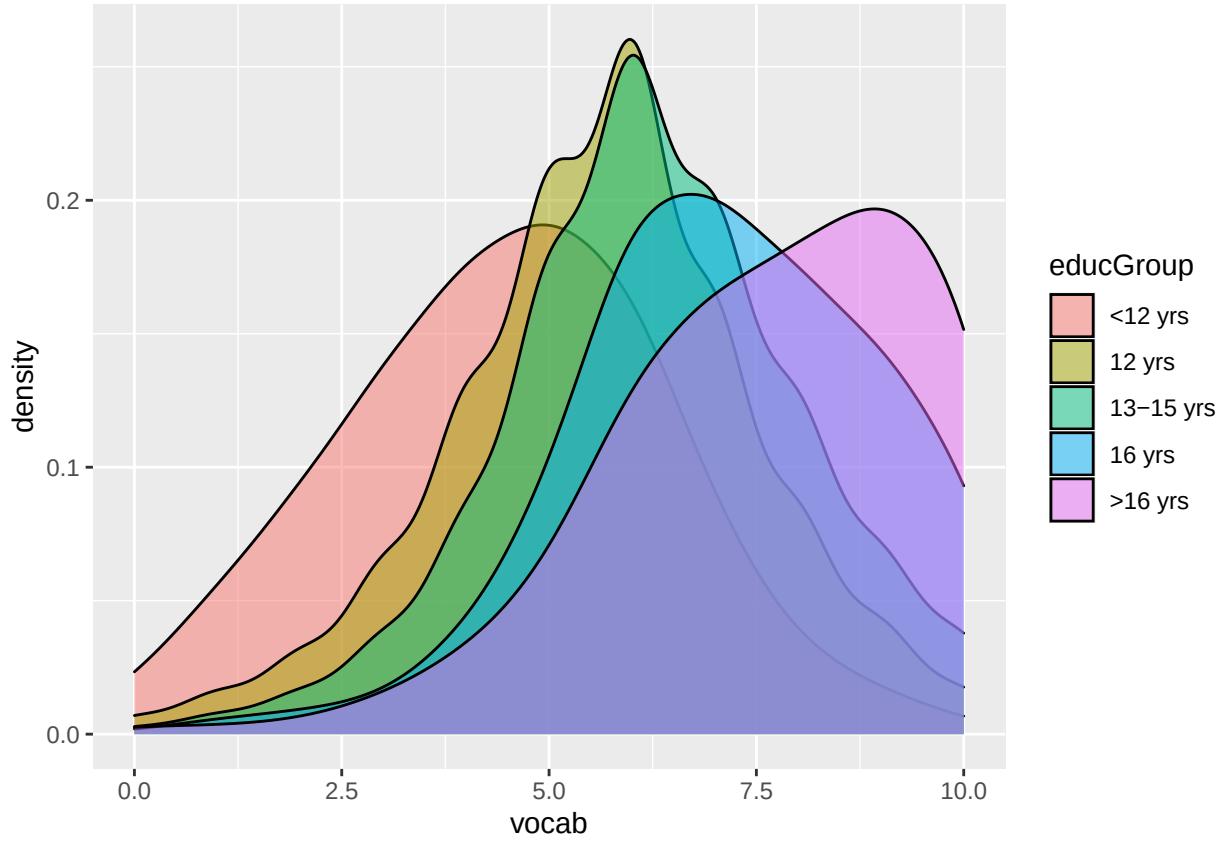


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

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

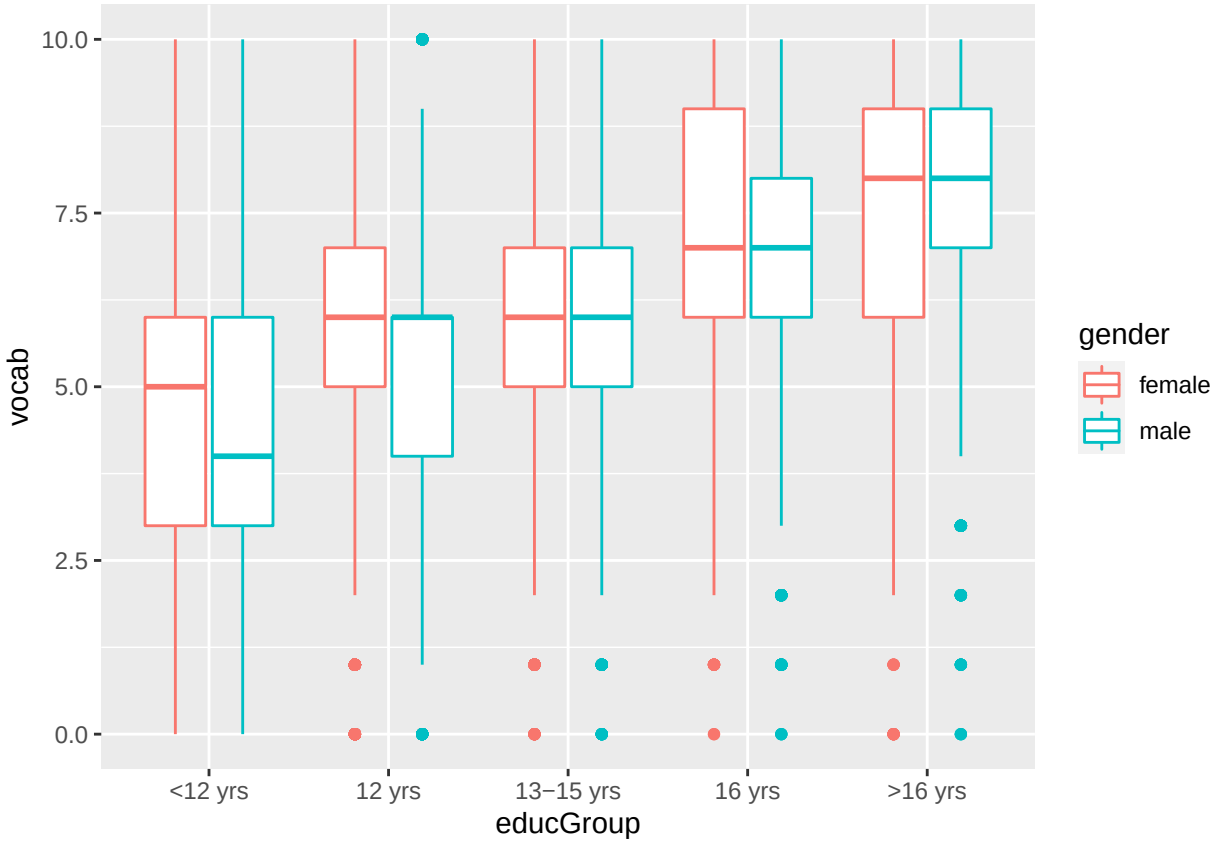



```
ggplot(GSSvocab) +
  aes(x= vocab) +
  geom_density(aes(fill = educGroup),adjust = 2,alpha=.5)
```



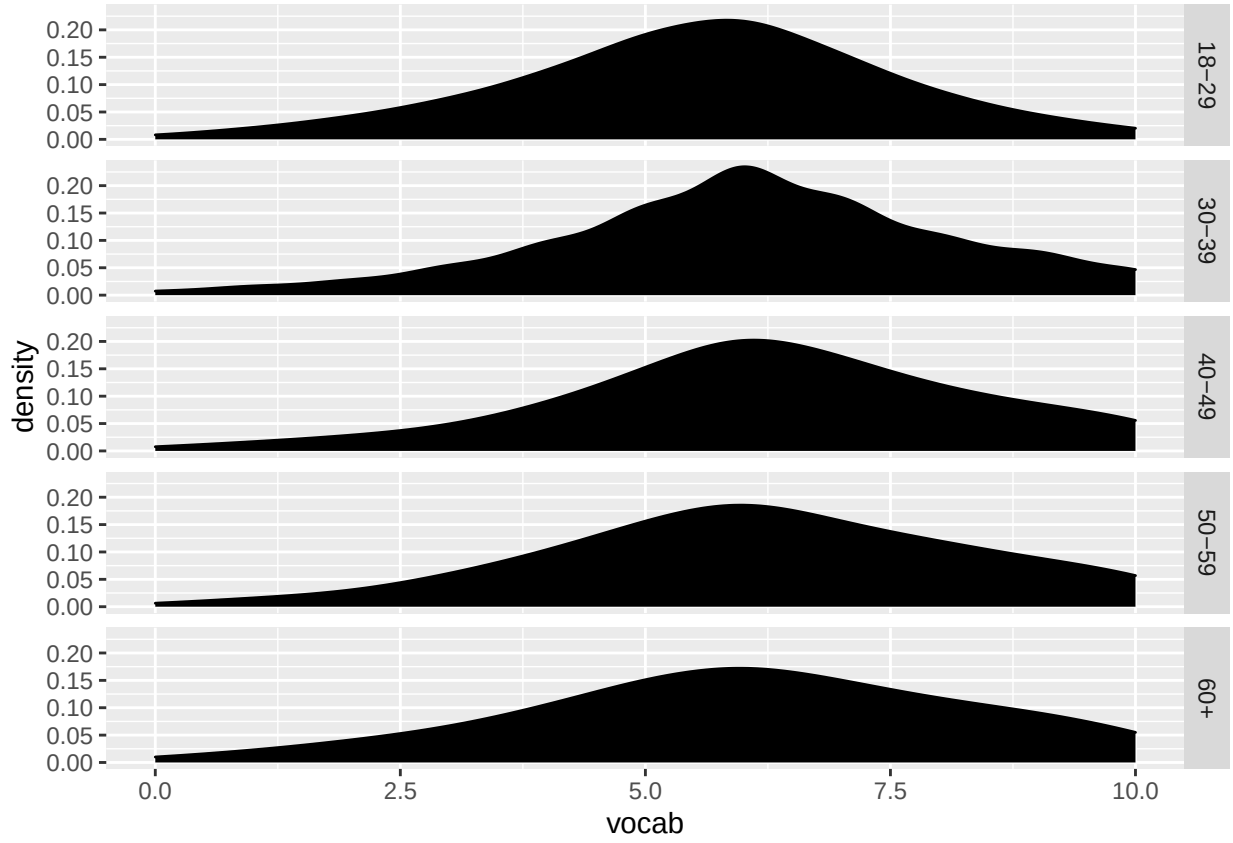
Using the best-looking plot from the previous question, create the best looking plot overloading with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?

```
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?

```
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$marital_status=="Married-
adult$marital_status = as.factor(adult$marital_status)
```

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

Holand-Netherlands has only 1 data point. This may overlap with some other feature and cause it to cancel out.

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)

tab = sort(table(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?

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

```
adult$worktype = paste(adult$occupation, adult$workclass, sep = ":")
tab = (table(adult$worktype))
tab
```

```
##
##           Adm-clerical:Federal-gov             Adm-clerical:Local-gov
##                                316                                281
##               Adm-clerical:Private          Adm-clerical:Self-emp-inc
##                               2793                                 28
##      Adm-clerical:Self-emp-not-inc             Adm-clerical:State-gov
##                                 49                                250
##           Adm-clerical:Without-pay           Armed-Forces:Federal-gov
##                                  3                                  9
##           Craft-repair:Federal-gov             Craft-repair:Local-gov
##                                 63                                143
##               Craft-repair:Private          Craft-repair:Self-emp-inc
##                               3146                                 99
##      Craft-repair:Self-emp-not-inc             Craft-repair:State-gov
##                                523                                 55
##           Craft-repair:Without-pay        Exec-managerial:Federal-gov
##                                  1                                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:Federal-gov
##                                186                                  8
##          Farming-fishing:Local-gov            Farming-fishing:Private
##                                 29                                450
##       Farming-fishing:Self-emp-inc   Farming-fishing:Self-emp-not-inc
##                                 51                                430
##          Farming-fishing:State-gov        Farming-fishing:Without-pay
##                                 15                                  6
##      Handlers-cleaners:Federal-gov        Handlers-cleaners:Local-gov
##                                 22                                 46
##          Handlers-cleaners:Private     Handlers-cleaners:Self-emp-inc
##                               1255                                  2
## Handlers-cleaners:Self-emp-not-inc        Handlers-cleaners:State-gov
##                                 15                                  9
##      Handlers-cleaners:Without-pay      Machine-op-inspct:Federal-gov
##                                  1                                 14
##        Machine-op-inspct:Local-gov          Machine-op-inspct:Private
##                                 11                               1882
##     Machine-op-inspct:Self-emp-inc Machine-op-inspct:Self-emp-not-inc
##                                 10                                 35
##        Machine-op-inspct:State-gov      Machine-op-inspct:Without-pay
##                                 13                                  1
##          Other-service:Federal-gov            Other-service:Local-gov
##                                 34                                189
##              Other-service:Private         Other-service:Self-emp-inc
##                               2665                                 27
##     Other-service:Self-emp-not-inc            Other-service:State-gov
##                                173                                123
##          Other-service:Without-pay            Priv-house-serv:Private
##                                  1                                143
##         Prof-specialty:Federal-gov           Prof-specialty:Local-gov
##                                167                                692
##             Prof-specialty:Private        Prof-specialty:Self-emp-inc
```

```
##                              2254                              157
##    Prof-specialty:Self-emp-not-inc          Prof-specialty:State-gov
##                               365                               403
##        Protective-serv:Federal-gov         Protective-serv:Local-gov
##                                27                               304
##           Protective-serv:Private       Protective-serv:Self-emp-inc
##                               186                                 5
##   Protective-serv:Self-emp-not-inc         Protective-serv:State-gov
##                                 6                               116
##                 Sales:Federal-gov                   Sales:Local-gov
##                                14                                 7
##                     Sales:Private                 Sales:Self-emp-inc
##                              2895                               281
##             Sales:Self-emp-not-inc                   Sales:State-gov
##                               376                                11
##           Tech-support:Federal-gov            Tech-support:Local-gov
##                                66                                38
##               Tech-support:Private         Tech-support:Self-emp-inc
##                               723                                 3
##      Tech-support:Self-emp-not-inc            Tech-support:State-gov
##                                26                                56
##       Transport-moving:Federal-gov         Transport-moving:Local-gov
##                                24                               115
##           Transport-moving:Private       Transport-moving:Self-emp-inc
##                              1247                                26
##  Transport-moving:Self-emp-not-inc         Transport-moving:State-gov
##                               118                                41
##       Transport-moving:Without-pay
##                                 1
```

Like the `native_country` exercise, there are a lot of rare levels. Collapse levels with less than 100 observations to type `other` and then cast this variable `worktype` as type `factor`. Recheck the tabulation to ensure you did this correct.

```
adult$worktype = ifelse(adult$worktype%in% names(tab[tab<50]),"Other",adult$worktype)
adult$worktype = as.factor(adult$worktype)
sort(table(adult$worktype))
```

```
##
##      Farming-fishing:Self-emp-inc            Craft-repair:State-gov
##                                51                                55
##            Tech-support:State-gov          Craft-repair:Federal-gov
##                                56                                63
##          Tech-support:Federal-gov         Craft-repair:Self-emp-inc
##                                66                                99
##        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                             Other
##                               523                               618
##          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
```

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

```
adult$relmarried = paste(adult$relationship, adult$marital_status, sep = ":")
tab = (table(adult$relmarried))
adult$relmarried = ifelse(adult$relmarried%in% names(tab[tab<50]),"Other",adult$relmarried)
adult$relmarried = as.factor(adult$relmarried)


(table(adult$relmarried))
```

```
##
##                      Husband:Married              Not-in-family:Divorced
##                                12463                                2268
## Not-in-family:Married-spouse-absent         Not-in-family:Never-married
##                                  181                                4447
##              Not-in-family:Separated               Not-in-family:Widowed
##                                  383                                 432
##                                Other             Other-relative:Divorced
##                                  135                                 103
##               Other-relative:Married        Other-relative:Never-married
##                                  119                                 548
##             Other-relative:Separated                 Own-child:Divorced
##                                   53                                 308
```

```
##                   Own-child:Married             Own-child:Never-married
##                                  84                                3929
##                 Own-child:Separated                  Unmarried:Divorced
##                                  90                                1535
##     Unmarried:Married-spouse-absent             Unmarried:Never-married
##                                 120                                 801
##                 Unmarried:Separated                   Unmarried:Widowed
##                                 413                                 343
##                        Wife:Married
##                                1406
```

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=binom
}
```

```
## 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
```

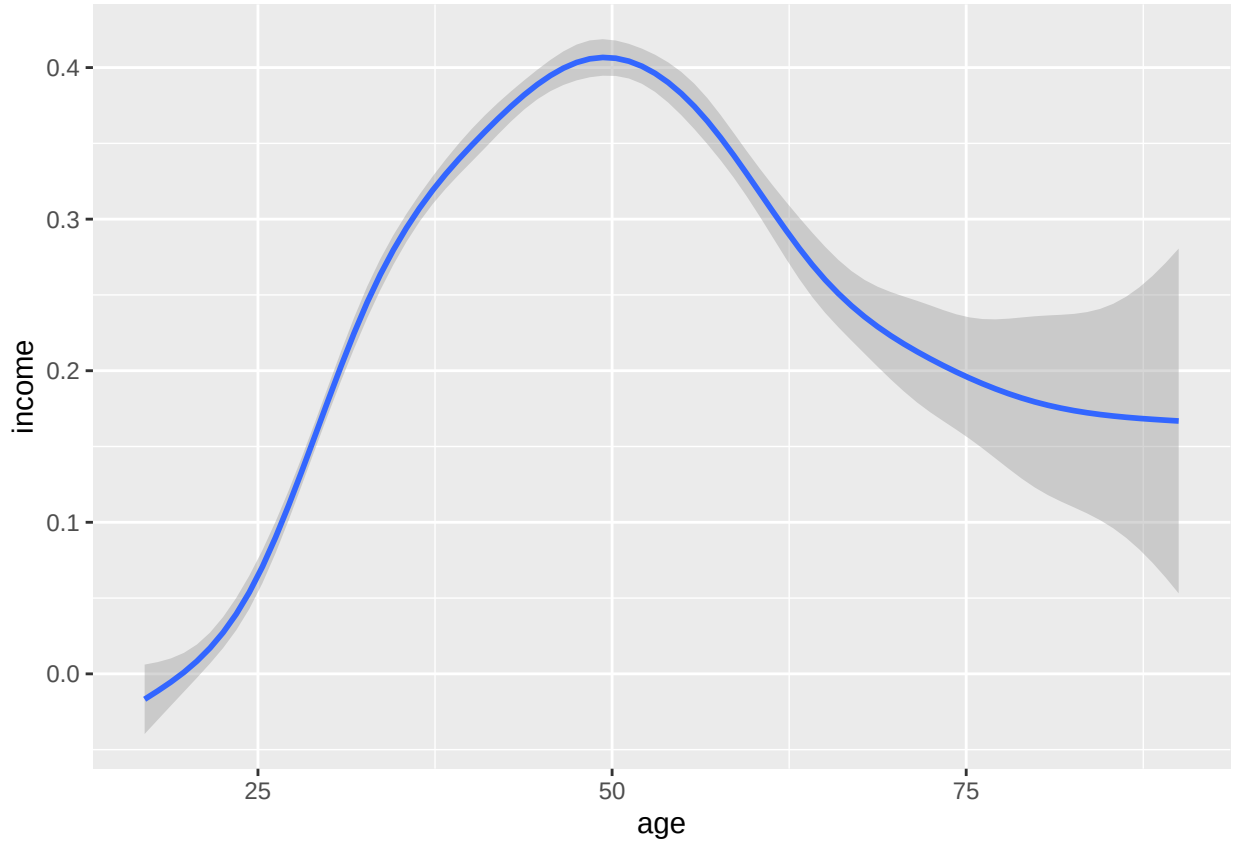
Now let's get fancier. Let's do some variable transforms. Add `log_capital_loss` derived from `capital_loss` and `log_capital_gain` derived from `capital_gain`. Since there are zeroes here, use log_x = log(1 + x) instead of log_x = log(x). That's always a neat trick. Just add them directly to the data frame so they'll be picked up with the `.` inside of a formula.

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

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

```
pacman::p_load(ggplot2)
ggplot(adult) +
  aes(x=age,y = income) +
  geom_smooth()
```

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



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

You see that income peaks at 50 and then goes down. This follows what would be expected.

Now let's fit the same models with all link functions on a formula called `age_interactions` that uses interactions for `age` with all of the variables. Add all these models to the `prob_est_mods` list.

```
#age_interactions = age ~ .
#prob_est_mods = list()

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

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)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## $`logit-vanilla`
## [1] -0.1030118
##
## $`probit-vanilla`
## [1] -0.1031256
##
## $`cloglog-vanilla`
## [1] -0.194289
##
## $`cauchit-vanilla`
## [1] -0.1045133
```

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)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## $`logit-vanilla`
## [1] -0.1028097
##
## $`probit-vanilla`
## [1] -0.1028824
##
```

```
## $`cloglog-vanilla`
## [1] -0.1957891
##
## $`cauchit-vanilla`
## [1] -0.1048611
```

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

logit wins both. That is because it is the best.

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

you are only running it once so you can get the wrong answer due to variance.

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

```
K = 2.5
test_and_select_prop = 1 / K
train_indices = sample(1 : nrow(adult), round((1 - test_and_select_prop) * nrow(adult)))
adult_train = adult[train_indices, ]
y_train = adult_train$income
X_train = adult_train
X_train$income = NULL

other_indices = setdiff(1 : nrow(adult), train_indices)

test_indices = sample (other_indices, .5* length(other_indices))
adult_test = adult[test_indices, ]
y_test = adult_test$income
X_test = adult_test
X_test$income = NULL

select_indices = setdiff(other_indices , test_indices)
adult_select = adult[select_indices, ]
y_select = adult_select$income
X_select = adult_select
X_select$income = NULL

lapply(prob_est_mods, brier_score, X_select, y_select)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## $`logit-vanilla`
## [1] -0.106131
##
## $`probit-vanilla`
## [1] -0.1060887
```

```
##
## $`cloglog-vanilla`
## [1] -0.1957891
##
## $`cauchit-vanilla`
## [1] -0.1080759
```

```
lapply(prob_est_mods, brier_score, X_test, y_test)
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading

## Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
## prediction from a rank-deficient fit may be misleading
```

```
## $`logit-vanilla`
## [1] -0.1014038
##
## $`probit-vanilla`
## [1] -0.1015502
##
## $`cloglog-vanilla`
## [1] -0.1937997
##
## $`cauchit-vanilla`
## [1] -0.1031376
```