Lab 6

Loyd Flores

#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 and 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. This will be a very hard visualization exercise since there is not a good model for vocabulary.

```
pacman::p_load(carData)

X=carData::GSSvocab
X=na.omit(X)

tinytex::reinstall_tinytex(repository = "illinois")
```

```
## If reinstallation fails, try install_tinytex() again. Then install the following packages:
##
## tinytex::tlmgr_install(c("amscls", "amsfonts", "amsmath", "atbegshi", "atveryend", "auxhook", "babel

## The directory C:\Users\lenovo\AppData\Roaming\TinyTeX/texmf-local is not empty. It will be backed up

## tlmgr install everyshi

## tlmgr --repository http://www.preining.info/tlgpg/ install tlgpg

## tlmgr option repository "https://ctan.math.illinois.edu/systems/texlive/tlnet"

## tlmgr update --list
```

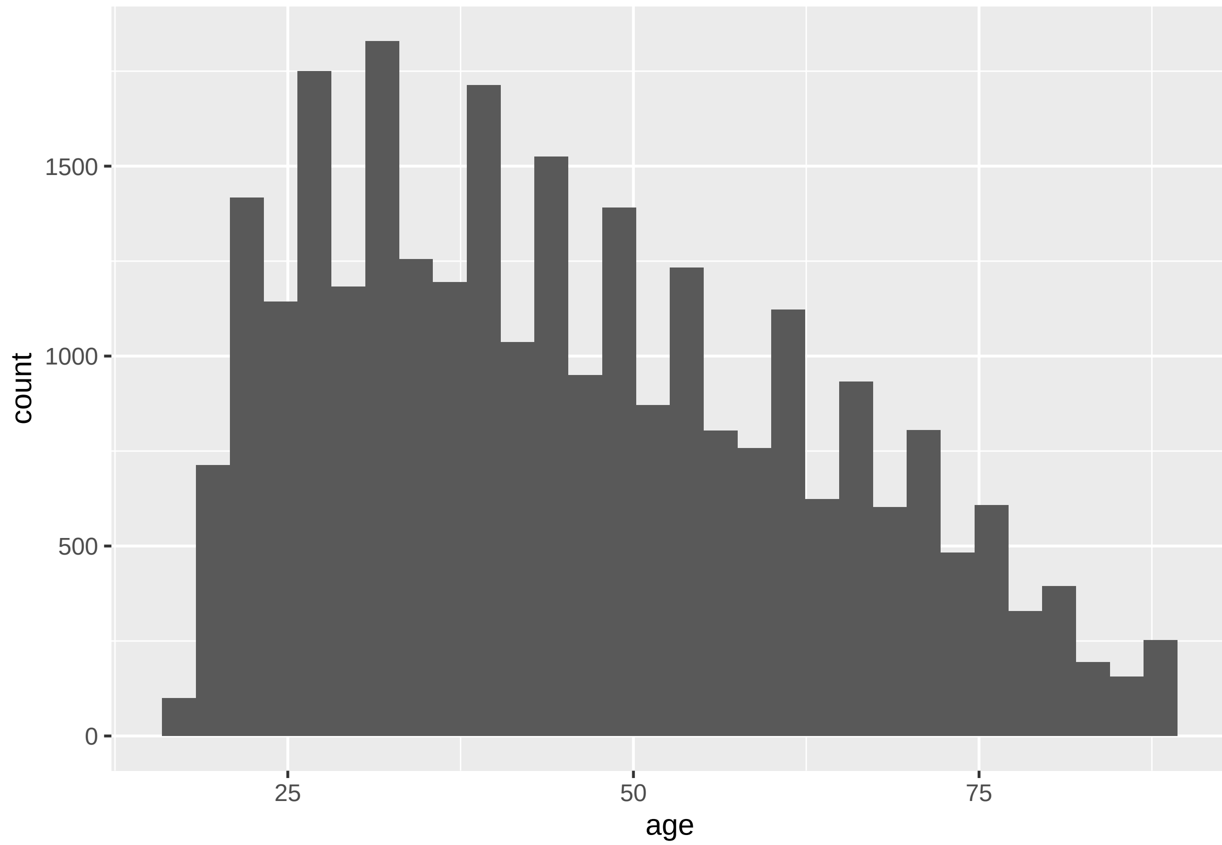
Briefly summarize the documentation on this dataset. What is the data type of each variable? What do you think is the response variable the collectors of this data had in mind?

#TO-DO

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

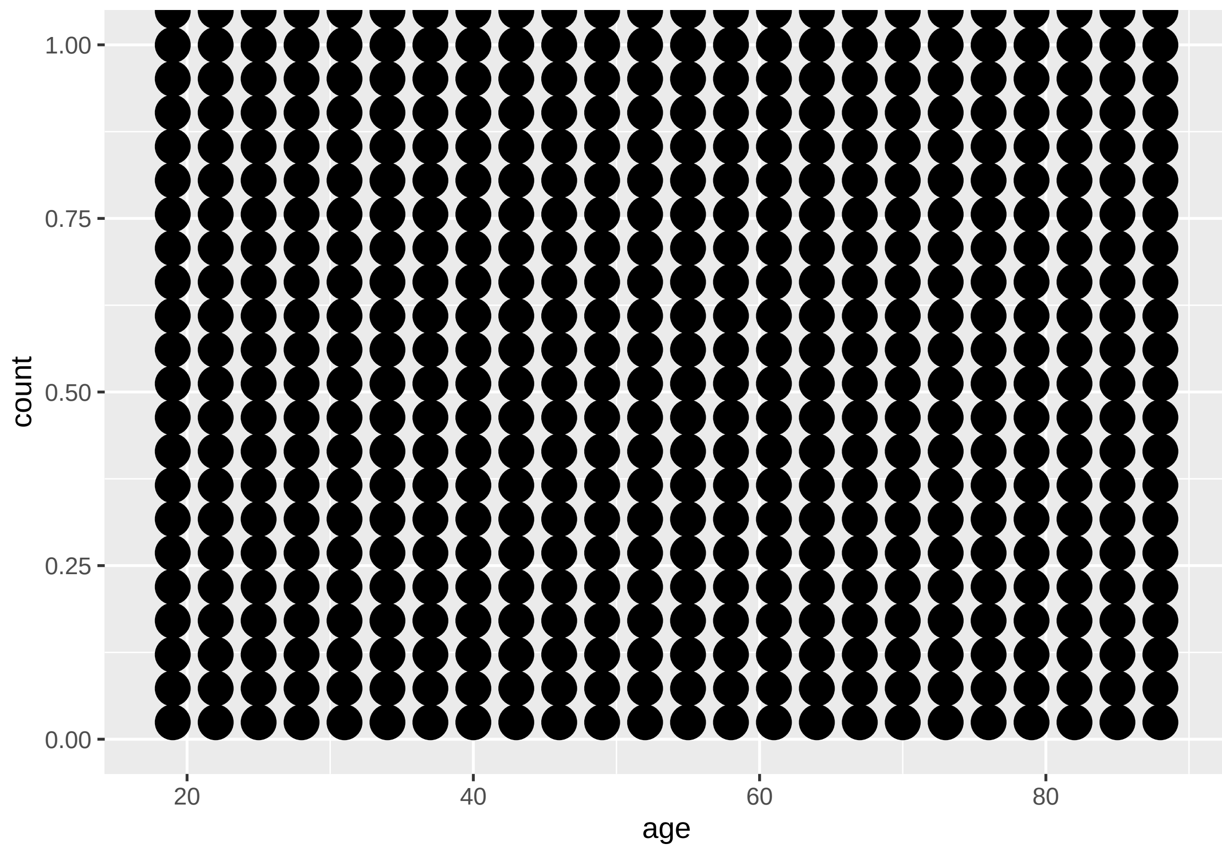
```
pacman::p_load(ggplot2)
plot_age=ggplot(X) +
  aes(age)
plot_age + geom_histogram()
```

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

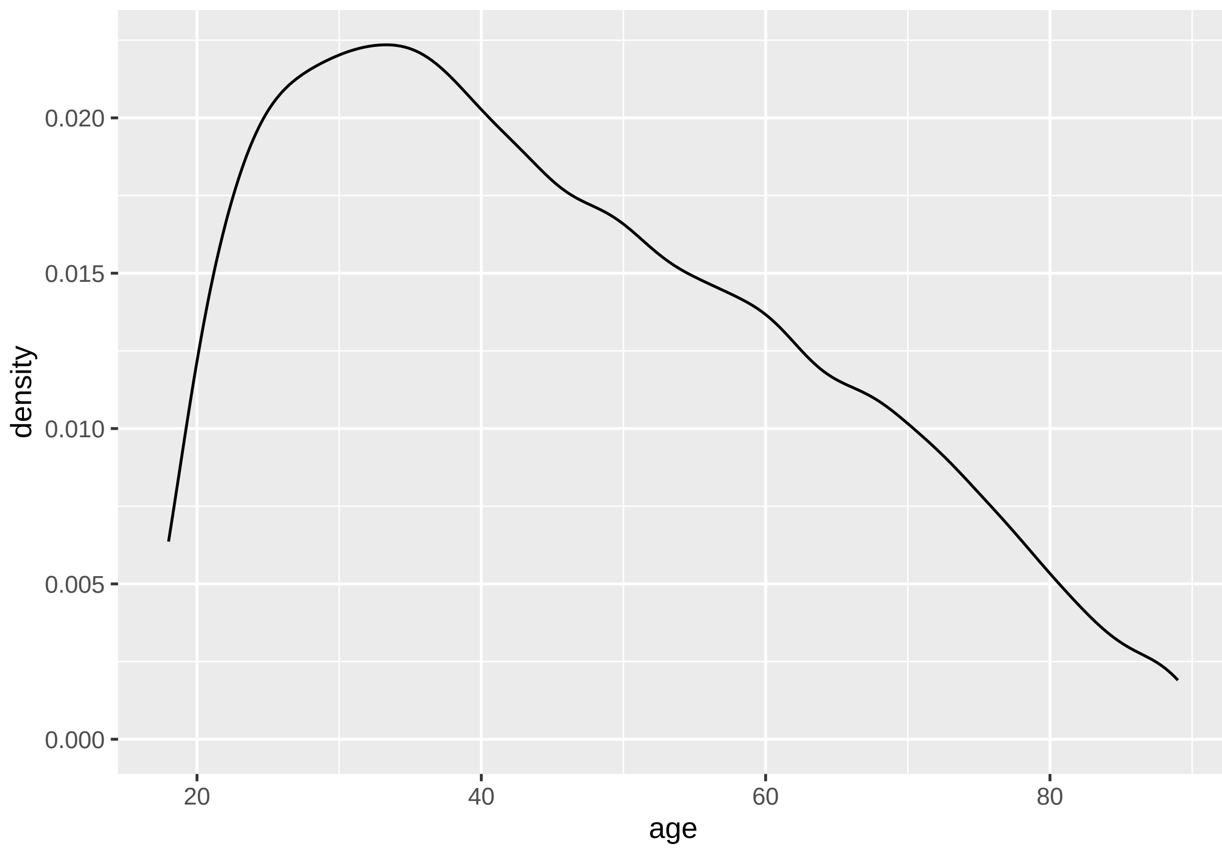



```r
plot_age + geom_dotplot()
```

```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.
```



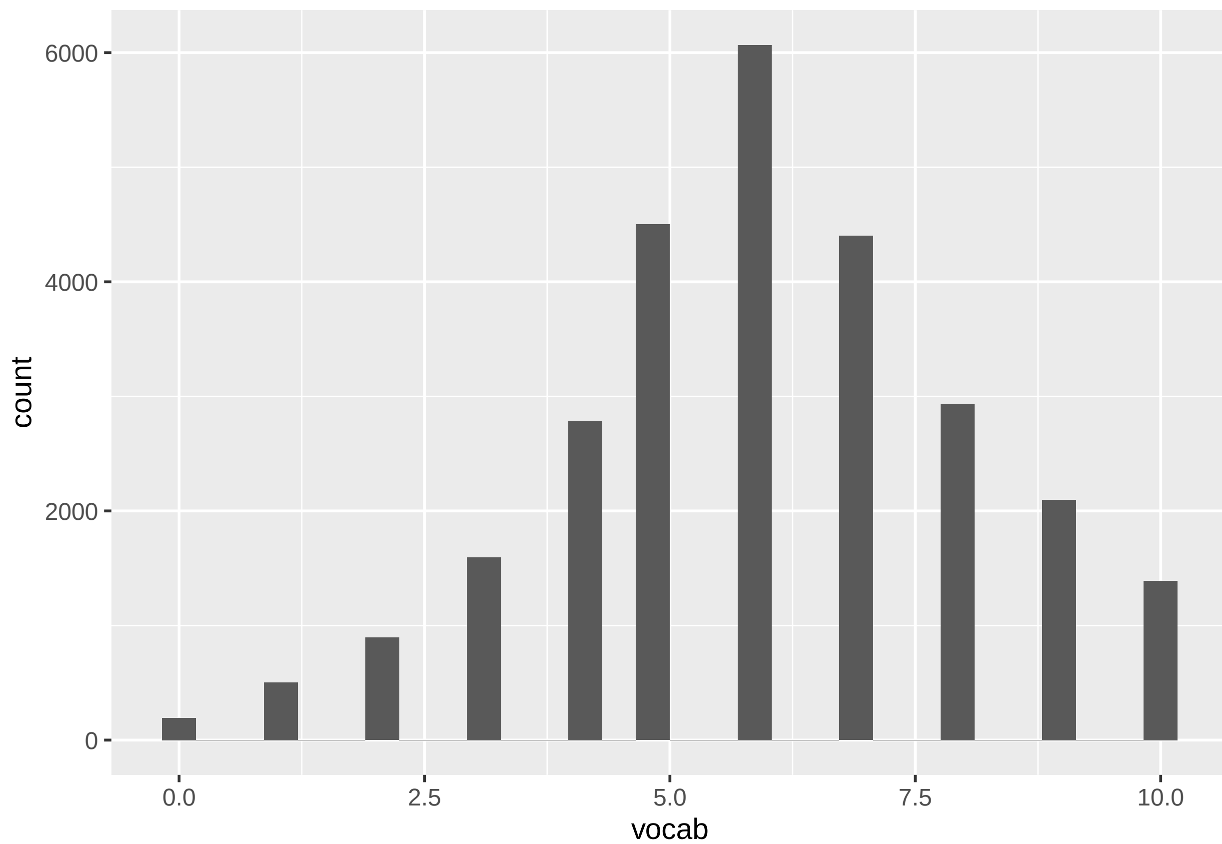

```
plot_age + geom_density()
```



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.

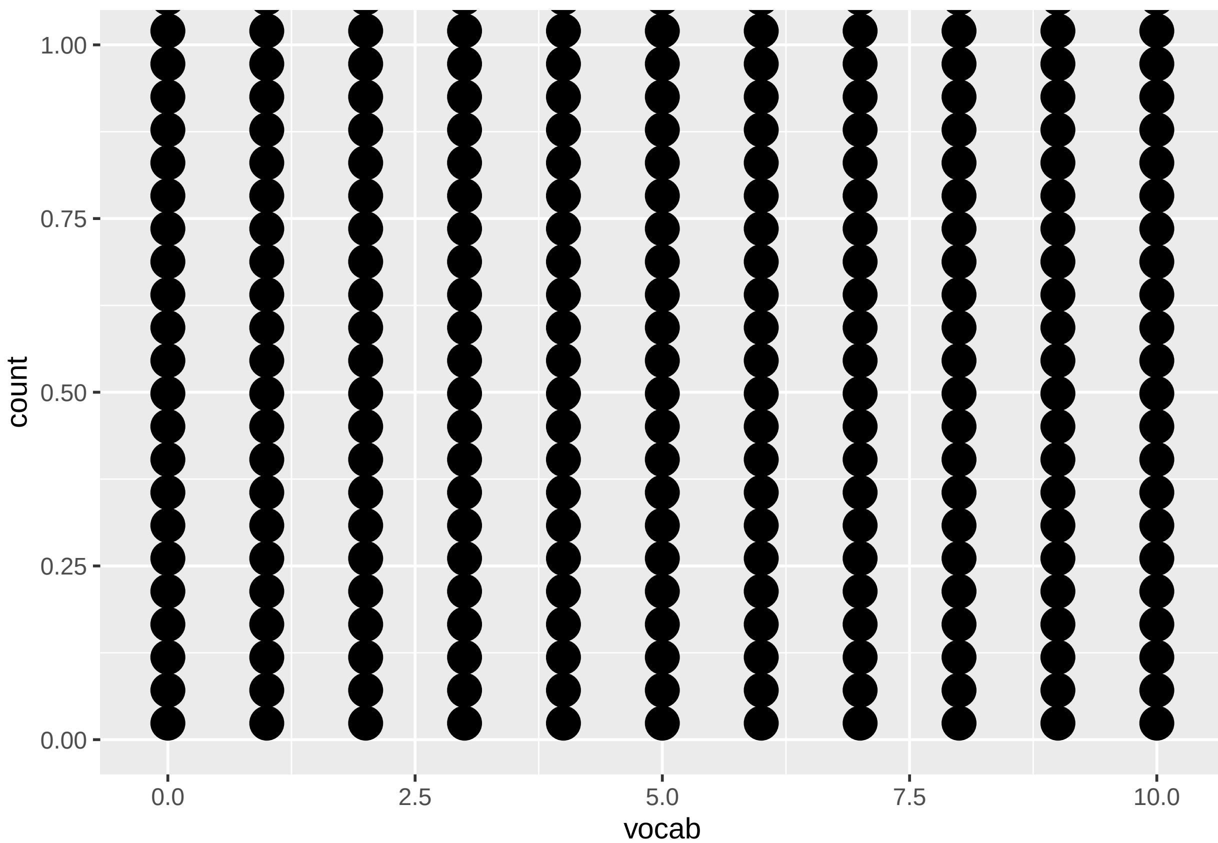
```
pacman::p_load(ggplot2)
plot_age=ggplot(X) +
  aes(vocab)
plot_age + geom_histogram()
```

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

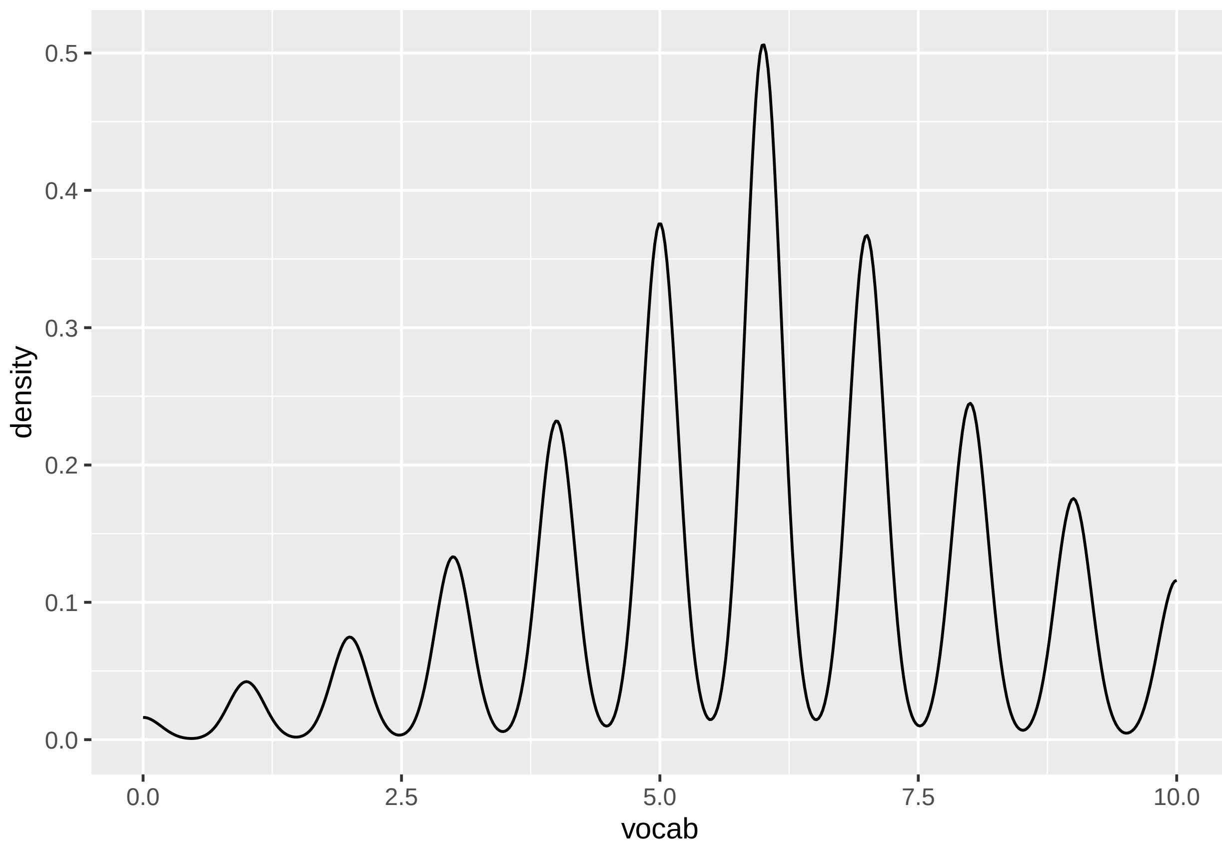



```r
plot_age + geom_dotplot()
```

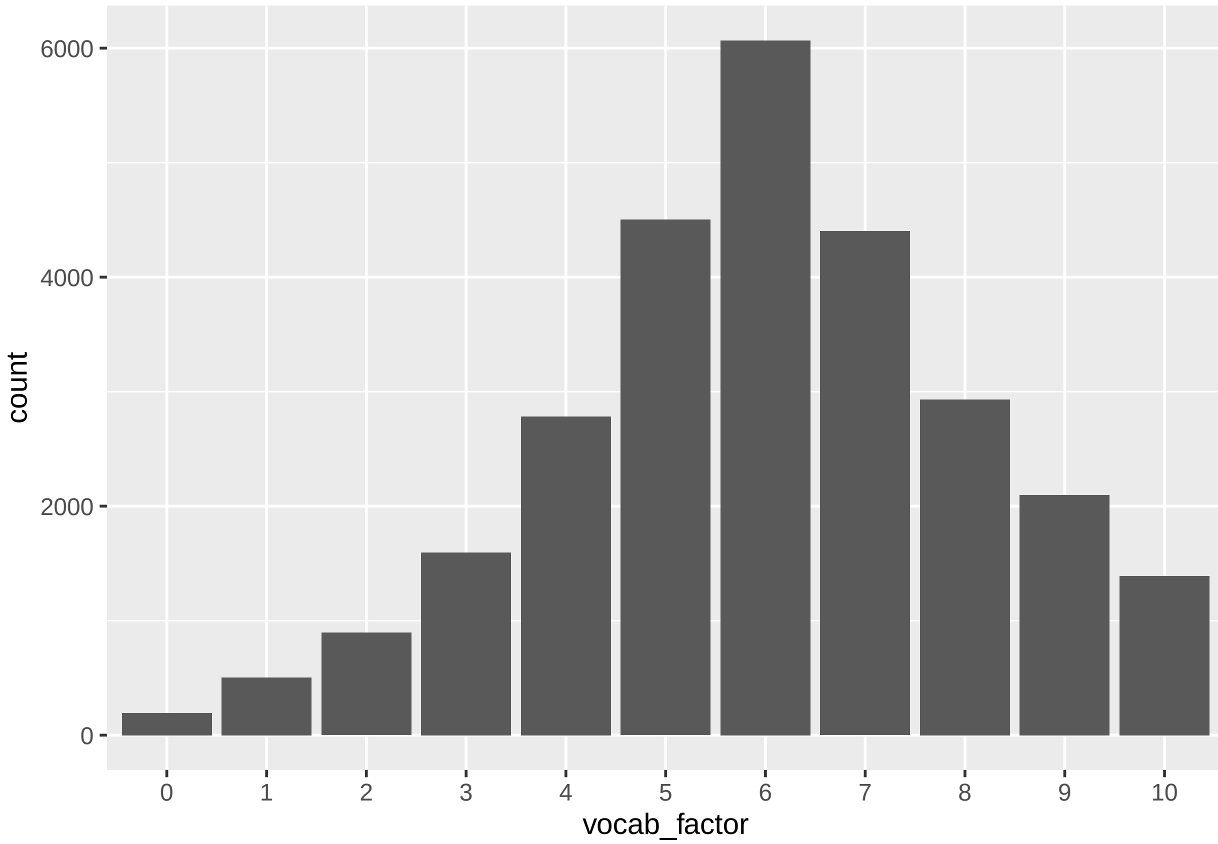
```
## Bin width defaults to 1/30 of the range of the data. Pick better value with
## `binwidth`.
```




```r
plot_age + geom_density()
```

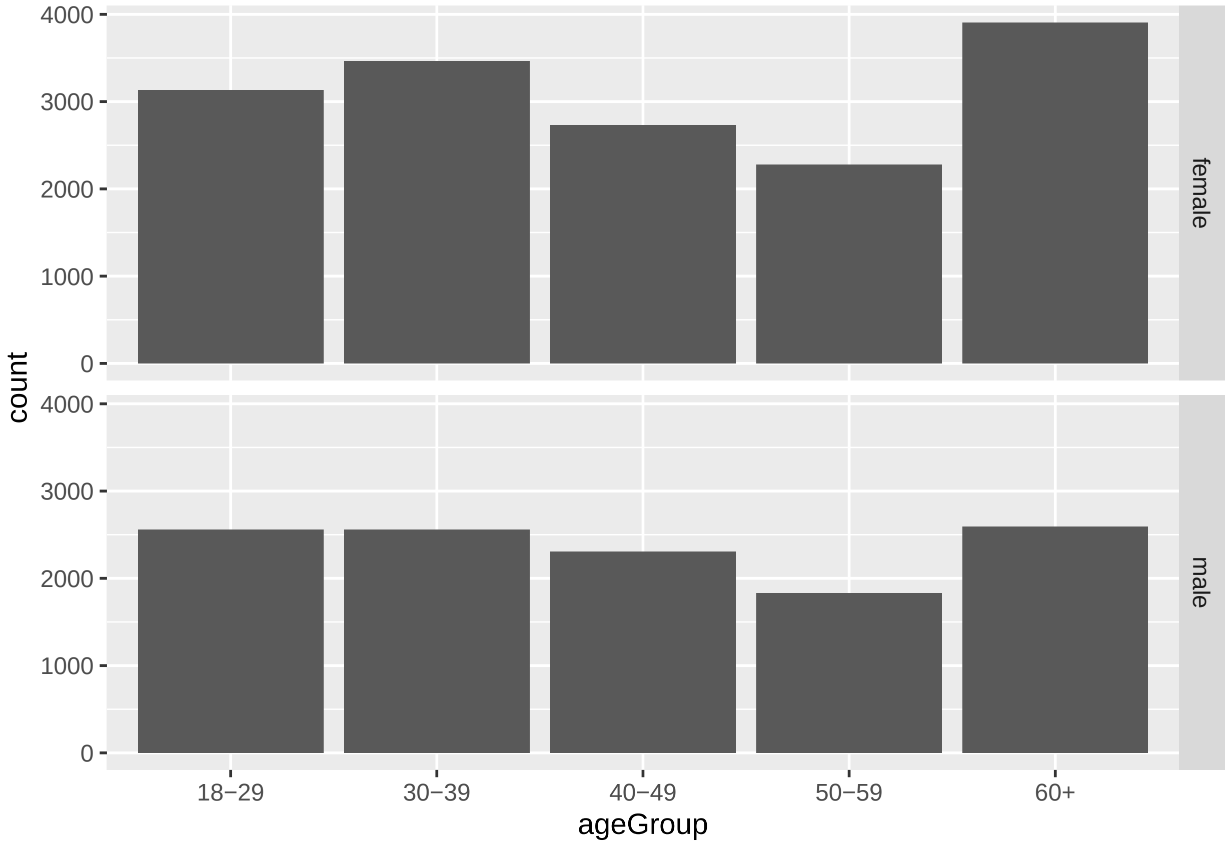



```
X$vocab_factor=factor(X$vocab)
ggplot(X) + aes(vocab_factor) + geom_bar()
```



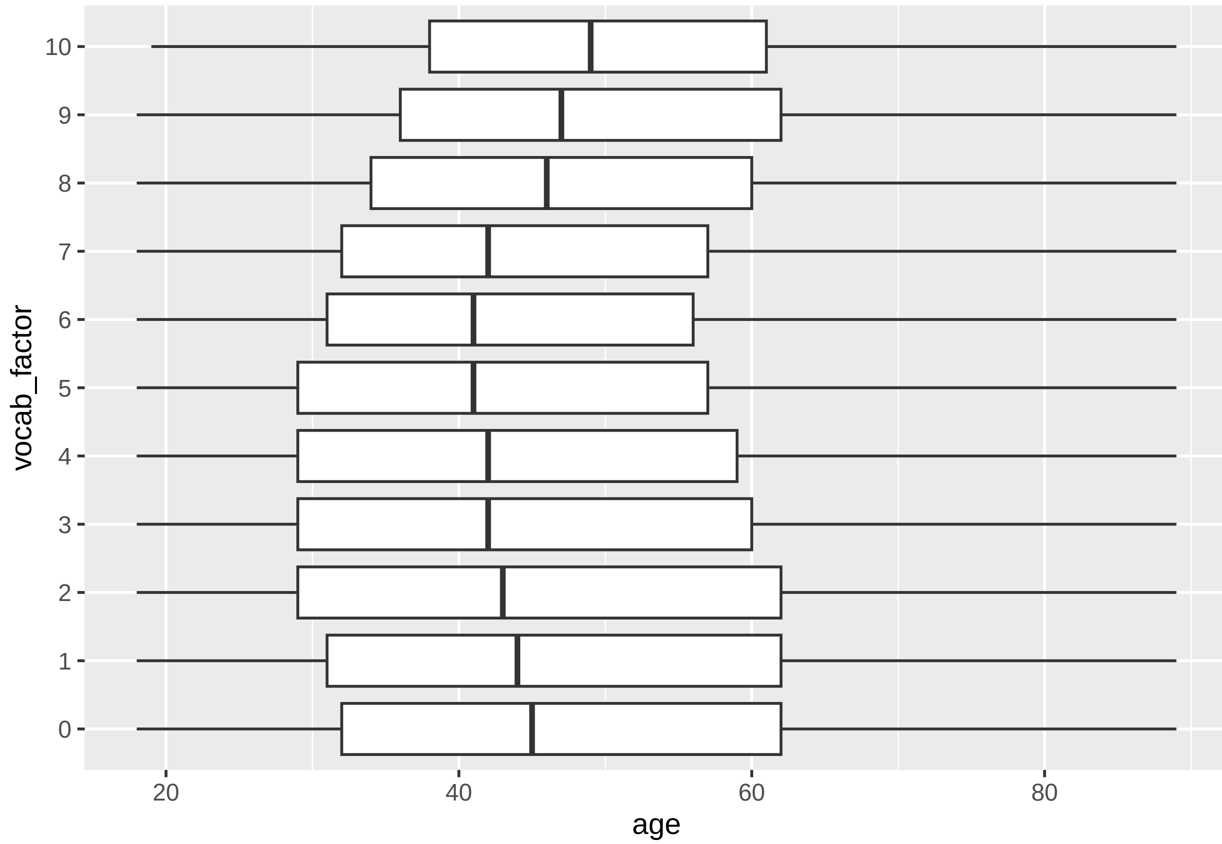
Create the best-looking plot you can to examine the `ageGroup` variable by `gender`. Does there appear to be an association? There are many ways to do this.

```
ggplot(X) + aes(x = ageGroup) + geom_bar() + facet_grid(gender~ .)
```



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

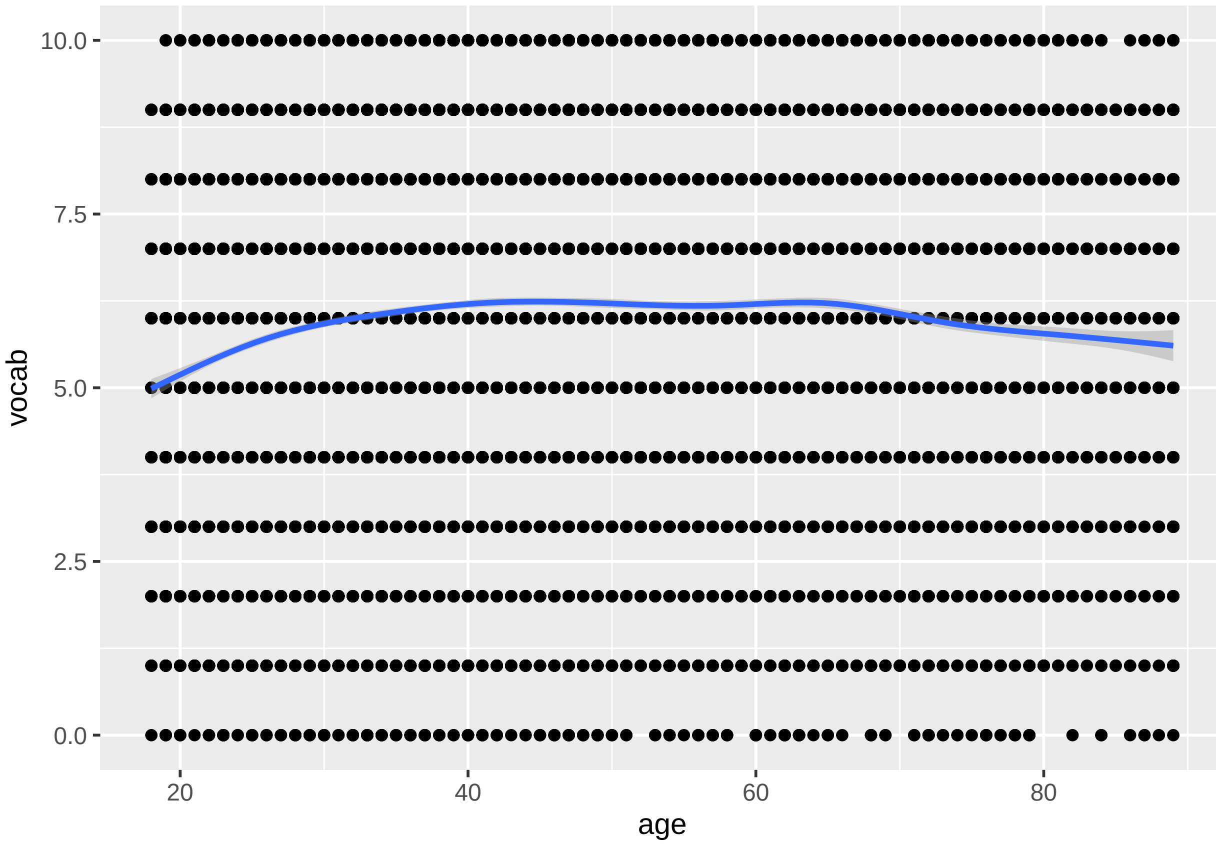
```
ggplot(X) + aes(x = age, y = vocab_factor) + geom_boxplot()
```



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

```
ggplot(X) + aes(x = age, y = vocab) + geom_point() + geom_smooth()
```

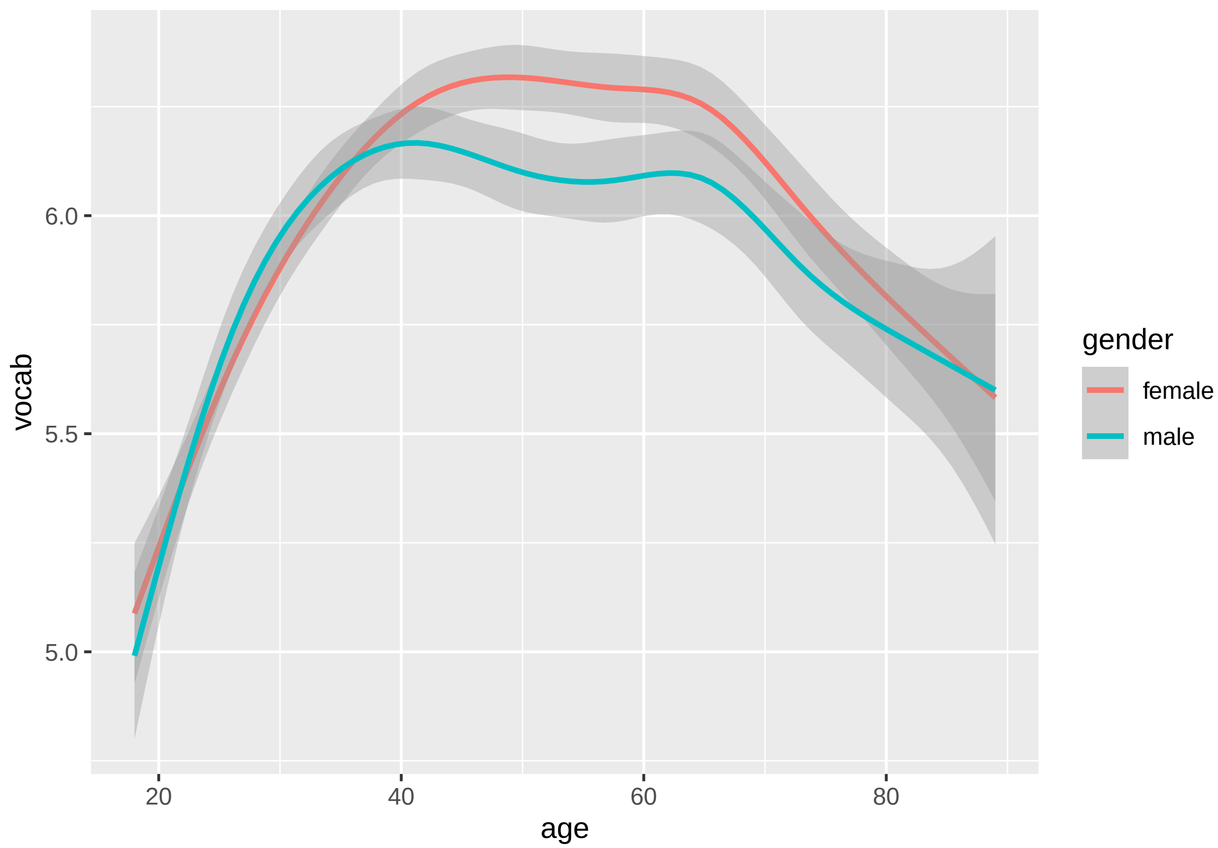
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



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(X) + aes(x = age, y = vocab, color = gender) + geom_smooth()
```

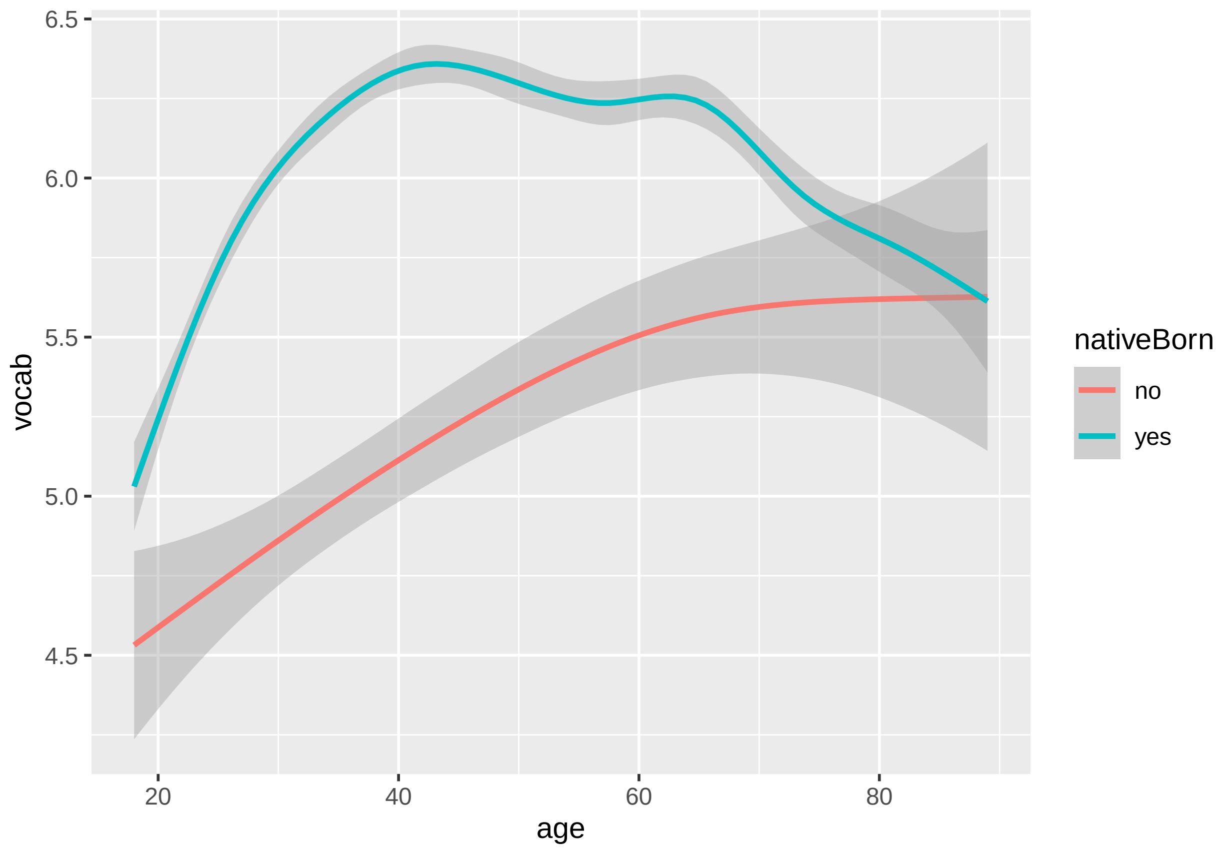
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



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`?

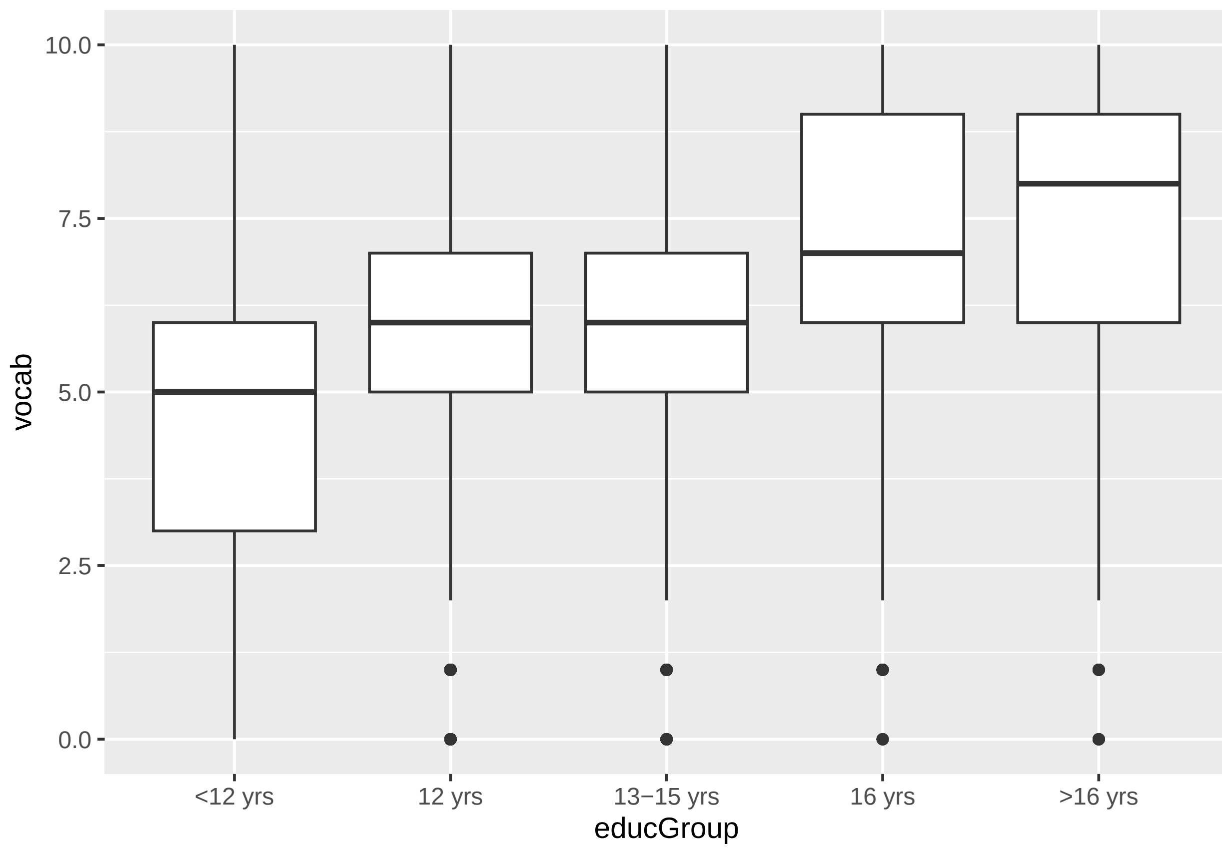
```
ggplot(X) + aes(x = age, y = vocab, color = nativeBorn) + geom_smooth()
```

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



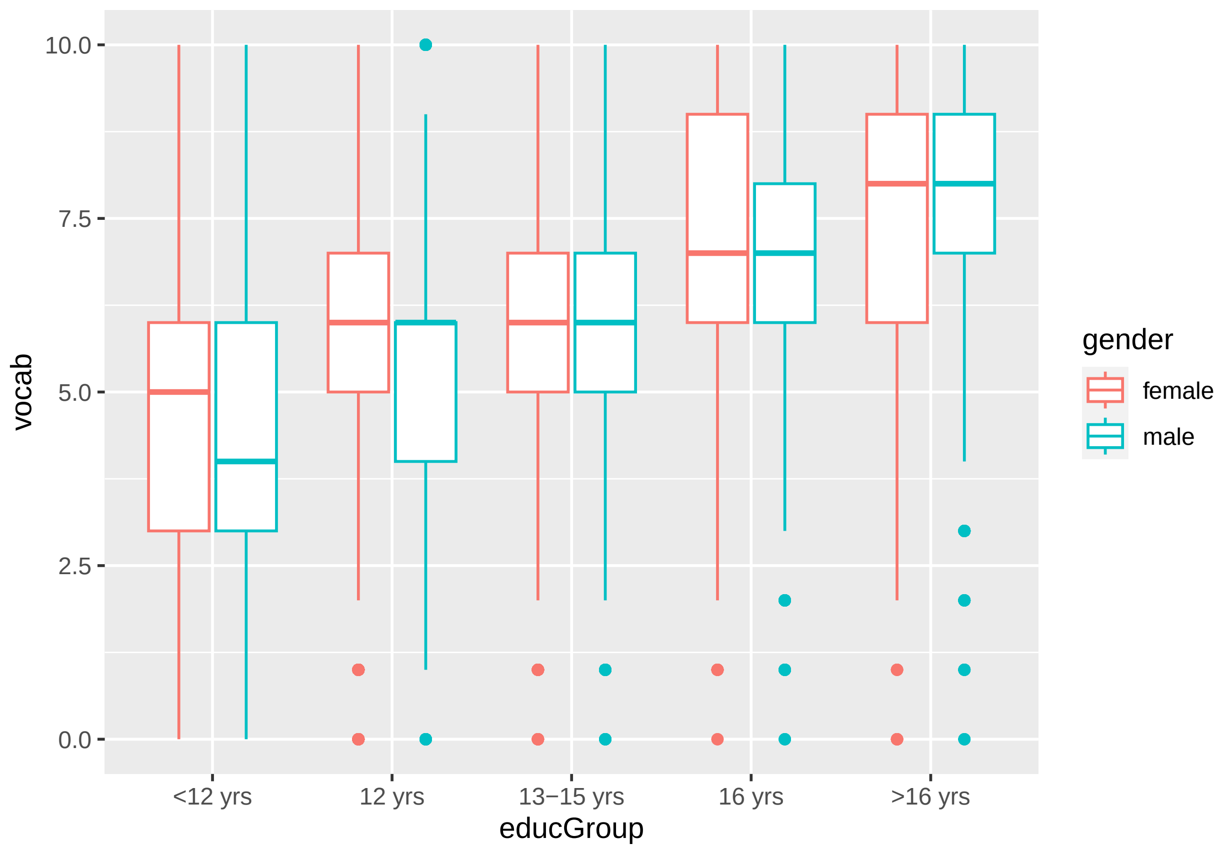
Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

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



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`?

```
#TO-DO
ggplot(X) + aes(x = educGroup, y = vocab, color = gender) + geom_boxplot()
```



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

```
# Based off of the graphs there seems to be a continuous increase as we age, which makes sense because
```

#Logistic Regression

Let's consider the Pima Indians Diabetes dataset from 1988:

```
?MASS::Pima.tr2
```

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

```
pima = na.omit(MASS::Pima.tr2)
skimr::skim(pima)
```

Table 1: Data summary

Name	pima
Number of rows	200
Number of columns	8
Column type frequency:	
factor	1

numeric	7
Group variables	None

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
type	0	1	FALSE	2	No: 132, Yes: 68

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
npreg	0	1	3.57	3.37	0.00	1.00	2.00	6.00	14.00	
glu	0	1	123.97	31.67	56.00	100.00	120.50	144.00	199.00	
bp	0	1	71.26	11.48	38.00	64.00	70.00	78.00	110.00	
skin	0	1	29.22	11.72	7.00	20.75	29.00	36.00	99.00	
bmi	0	1	32.31	6.13	18.20	27.58	32.80	36.50	47.90	
ped	0	1	0.46	0.31	0.09	0.25	0.37	0.62	2.29	
age	0	1	32.11	10.98	21.00	23.00	28.00	39.25	63.00	

```r
y = ifelse(pima$type == "Yes", 1, 0)
X = cbind(1, pima[, 1 : 7])
```

Note the missing data. We will learn about how to handle missing data towards the end of the course. For now, replace, the missing data in the design matrix X with the average of the feature x_dot,j. You can check that this worked with the table commands at the end of the chunk:

```r
table(X$bp, useNA = "always")
```

```
##
##   38   40   48   50   52   54   55   56   58   60   62   64   65   66   68   70
##    1    1    3    2    3    4    1    3    7   13   10    9    1   12   15   17
##   72   74   75   76   78   80   82   84   85   86   88   90   92   94   95  102
##   10   14    1   12   13   10   11    6    2    3    4    5    1    2    1    1
##  106  110 <NA>
##    1    1    0
```

```r
table(X$skin, useNA = "always")
```

```
##
##    7    8   10   11   12   13   14   15   16   17   18   19   20   21   22   23
##    1    1    2    3    6    4    3    6    2    8    5    4    5    4    5    7
##   24   25   26   27   28   29   30   31   32   33   34   35   36   37   38   39
##    1    6    6    9    8    8    9    7   10    7    4    5    5    3    1    4
##   40   41   42   43   44   45   46   48   49   50   52   60   99 <NA>
##    8    5    3    6    2    3    5    2    2    2    1    1    1    0
```

```r
table(X$bmi, useNA = "always")
```

```
##
## 18.2 19.3 20.4   21 21.1 22.1 22.2 22.5 22.9 23.1 23.2 23.4 23.8   24 24.1 24.2
##    2    1    2    1    1    2    1    1    1    1    1    1    1    1    1    3
## 24.4 24.6 24.7 24.8   25 25.1 25.2 25.3 25.4 25.5 25.6 25.9 26.1 26.2 26.3 26.4
##    1    1    1    1    1    1    2    1    4    1    1    2    1    2    1    1
## 26.5 27.2 27.3 27.4 27.5 27.6 27.8 28.4 28.5 28.6 28.7 29.3 29.7 30.1 30.2 30.4
##    2    2    1    1    1    2    3    3    1    2    2    3    2    2    1    1
## 30.8 30.9 31.1 31.2 31.3 31.6   32 32.2 32.4 32.5 32.8 33.1 33.2 33.3 33.6 33.7
##    4    4    1    1    1    4    6    1    3    1    4    1    3    2    1    3
## 33.8 33.9   34 34.1 34.2 34.3 34.4 34.5 34.6 34.7 34.9   35 35.2 35.3 35.4 35.5
##    1    1    1    2    6    3    1    1    3    1    2    1    1    1    2    1
## 35.6 35.8 35.9 36.1 36.2 36.4 36.5 36.6 36.8 36.9   37 37.4 37.6 37.7 37.8 37.9
##    1    2    3    1    1    1    2    2    1    1    1    3    3    2    1    1
## 38.1 38.2 38.4 38.5 38.7 38.8 38.9 39.1 39.2 39.4 40.5 40.6 41.3 41.8 42.1 42.6
##    2    1    1    1    1    1    1    2    1    5    1    1    2    1    2    1
## 42.9 43.1 43.3 43.5 46.1 46.3 46.8 47.9 <NA>
##    2    1    1    1    2    1    1    1    0
```

Now let's fit a log-odds linear model of y=1 (type is "diabetic") on just the `glu` variable. Use `optim` to fit the model.

```r
x = pima$glu
log_logistic_prob = function(w){
  -sum(-y*log(1+exp(-w[1]-w[2]*x))-(1-y)*log(1+exp(w[1]+w[2]*x)))
}
optim(c(0, 0), log_logistic_prob)$par
```

```
## [1] -5.50249591  0.03777468
```

Run a logistic regression of y=1 (type is "diabetic") on just the `glu` variable using R's built-in function and report b_0, b_1.

```r
b = coef(glm(y~x, family = "binomial"))
b
```

```
## (Intercept)           x
## -5.50363574  0.03778372
```

Comment on how close the results from R's built-in function was and your optimization call.

#TO-DO

Interpret the value of b_1 from R's built-in function.

a one unit increase in x results in a .04 increase in log odds of having diabetes

Interpret the value of b_0 from R's built-in function. When all the variables are 0 the log odds are -5.504 or 0.004

#TO-DO

Plot the probability of y=1 from the minimum value of `glu` to the maximum value of `glu`.

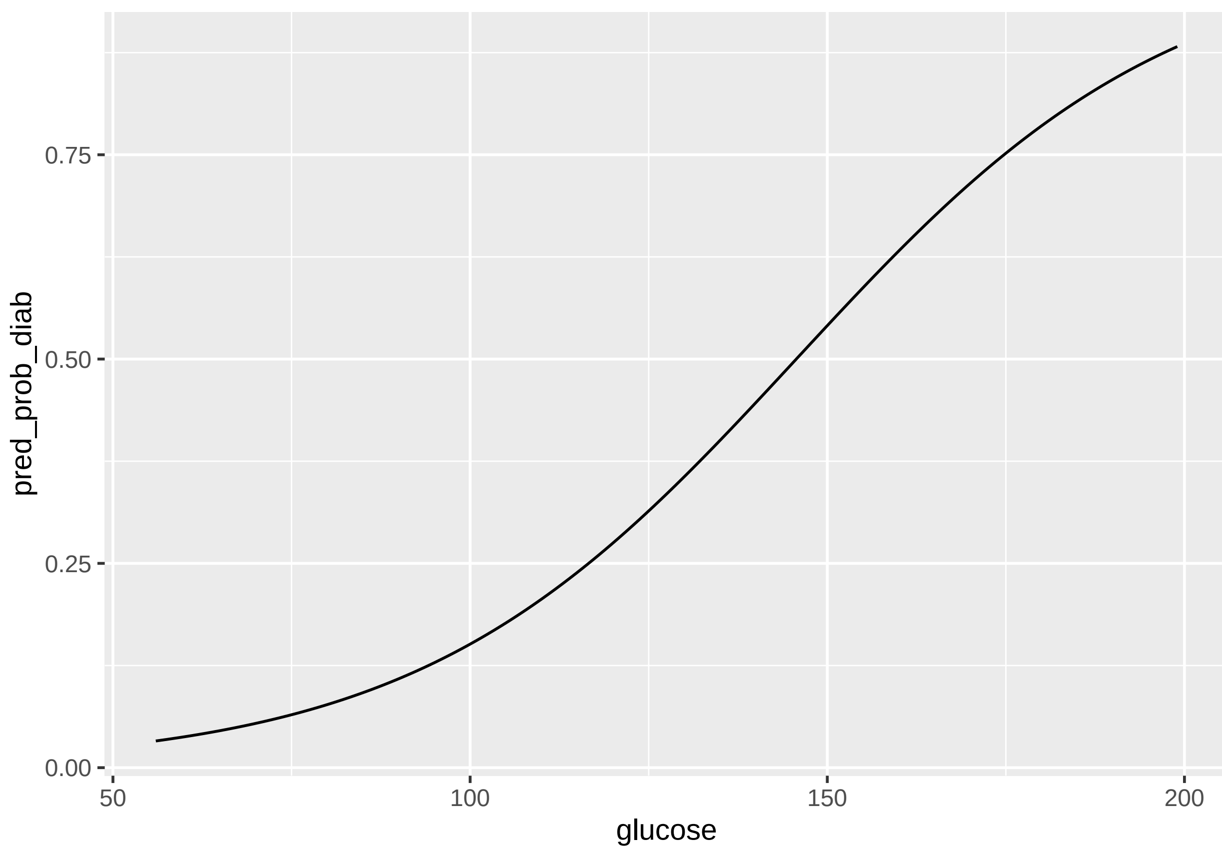
```r
min(x)
```

```
## [1] 56
```

```r
max(x)
```

```
## [1] 199
```

```r
res = .1
x_stars = seq(from = min(x), to = max(x), by = res)
log_odds_hat = cbind(1, x_stars)%*%b
p_hat = 1/(1+exp(-log_odds_hat))
pacman::p_load(ggplot2)
ggplot(data.frame(glucose = x_stars, pred_prob_diab = p_hat)) +
  aes(x = glucose, y = pred_prob_diab) +
  geom_line()
```



Run a logistic regression of y=1 (type is "diabetic") on all variables using R's built-in function and report the b vector.

```r
coef(glm(y ~ X[, "glu"], family = "binomial"))
```

```
## (Intercept)  X[, "glu"]
## -5.50363574  0.03778372
```

Predict the probability of diabetes for someone with a blood sugar of 150.

```
#TO-DO
model = glm(y ~ X[, "glu"], data = pima, family = "binomial")
b_of_model = coef(model)
predict_150 = predict(model, newdata = data.frame(glu = 150), type='response')
```

```
## Warning: 'newdata' had 1 row but variables found have 200 rows
```

```
print(predict_150)
```

```
##          1          2          3          4          5          6          7
## 0.09498471 0.86578450 0.06950685 0.67495399 0.18834838 0.13721517 0.08567797
##          8          9         10         11         12         13         14
## 0.85675823 0.46546813 0.33909724 0.41890592 0.57811911 0.83719271 0.11634224
##         15         16         17         18         19         20         21
## 0.09498471 0.14640906 0.20017348 0.47488016 0.53149283 0.43740539 0.14640906
##         22         23         24         25         26         27         28
## 0.15119438 0.08567797 0.15610751 0.09828277 0.66661042 0.14640906 0.44672421
##         29         30         31         32         33         34         35
## 0.19419262 0.20629139 0.07455608 0.52207428 0.28255850 0.61448200 0.17706719
##         36         37         38         39         40         41         42
## 0.49375118 0.07455608 0.05620031 0.16114998 0.26749422 0.75883421 0.13721517
##         43         44         45         46         47         48         49
## 0.34761574 0.13721517 0.09498471 0.31417530 0.29812585 0.11634224 0.72259695
##         50         51         52         53         54         55         56
## 0.88239666 0.24587694 0.08567797 0.57811911 0.23213553 0.18264015 0.33068169
##         57         58         59         60         61         62         63
## 0.30609179 0.20017348 0.29812585 0.69130881 0.80977802 0.13280306 0.34761574
##         64         65         66         67         68         69         70
## 0.11634224 0.20017348 0.43740539 0.39159385 0.18264015 0.36494415 0.40063169
##         71         72         73         74         75         76         77
## 0.61448200 0.21893914 0.79170232 0.28255850 0.69931308 0.48431008 0.15610751
##         78         79         80         81         82         83         84
## 0.13280306 0.18834838 0.28255850 0.15119438 0.57811911 0.31417530 0.31417530
##         85         86         87         88         89         90         91
## 0.29028050 0.23213553 0.23893847 0.23213553 0.23893847 0.35623288 0.07455608
##         92         93         94         95         96         97         98
## 0.21893914 0.54088898 0.11251377 0.15119438 0.44672421 0.20629139 0.12433912
##         99        100        101        102        103        104        105
## 0.08868455 0.52207428 0.03920982 0.25294992 0.14640906 0.07720533 0.57811911
##        106        107        108        109        110        111        112
## 0.16632322 0.21254659 0.30609179 0.47488016 0.07994059 0.83719271 0.24587694
##        113        114        115        116        117        118        119
## 0.16632322 0.30609179 0.05620031 0.41890592 0.21893914 0.52207428 0.40973764
##        120        121        122        123        124        125        126
## 0.49375118 0.12028332 0.18834838 0.55025616 0.13721517 0.48431008 0.21893914
##        127        128        129        130        131        132        133
## 0.14640906 0.20017348 0.27496232 0.82662863 0.34761574 0.77896582 0.07720533
##        134        135        136        137        138        139        140
## 0.17706719 0.84723204 0.12851170 0.14640906 0.41890592 0.13721517 0.15119438
##        141        142        143        144        145        146        147
## 0.69130881 0.78540274 0.29028050 0.10879578 0.27496232 0.57811911 0.03268059
```

```
##        148        149        150        151        152        153        154
## 0.76568104 0.30609179 0.09178605 0.10168248 0.55958783 0.87841873 0.83197691
##        155        156        157        158        159        160        161
## 0.43740539 0.69931308 0.87432542 0.46546813 0.32237311 0.61448200 0.35623288
##        162        163        164        165        166        167        168
## 0.15119438 0.66661042 0.12851170 0.29028050 0.09178605 0.55025616 0.48431008
##        169        170        171        172        173        174        175
## 0.21254659 0.18834838 0.23893847 0.17706719 0.86133298 0.80977802 0.12851170
##        176        177        178        179        180        181        182
## 0.30609179 0.21254659 0.41890592 0.03389635 0.60549346 0.12851170 0.44672421
##        183        184        185        186        187        188        189
## 0.25294992 0.15119438 0.29812585 0.42813054 0.15119438 0.75185215 0.06252401
##        190        191        192        193        194        195        196
## 0.38262973 0.26749422 0.58730625 0.33909724 0.21893914 0.44672421 0.45608062
##        197        198        199        200
## 0.34761574 0.18264015 0.26015618 0.58730625
```

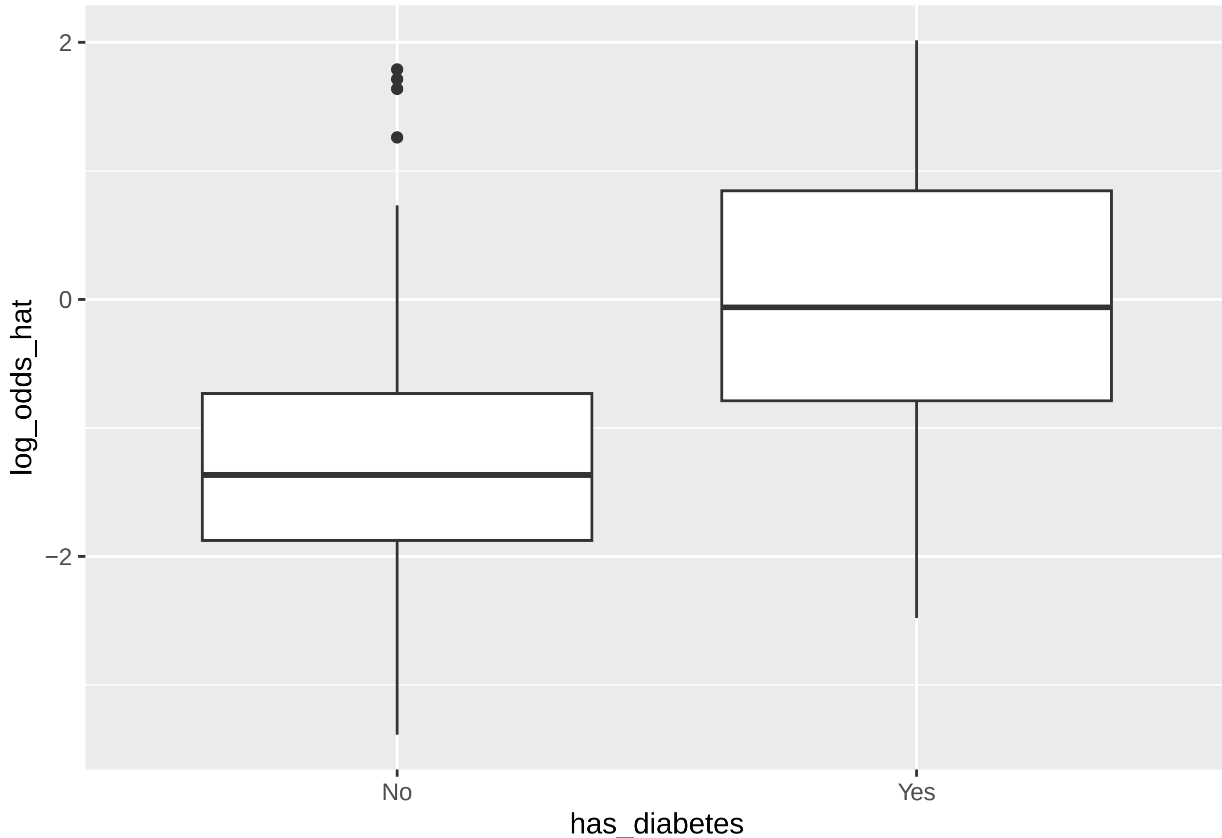
For 100 people with blood sugar of 150, what is the probability more than 75 of them have diabetes? (You may need to review 241 to do this problem).

```r
#TO-DO
# Number of trials
n = 100

# Probability of success (probability of having diabetes for one individual with blood sugar of 150)
p = predict_150

# Calculate the probability of having less than or equal to 75 successes
prob_less_than_or_equal_to_75 = pbinom(75, size = n, prob = p)

# Probability of having more than 75 successes
prob_more_than_75 = 1 - prob_less_than_or_equal_to_75

# Print the result
print(mean(prob_more_than_75))
```

```
## [1] 0.086975
```

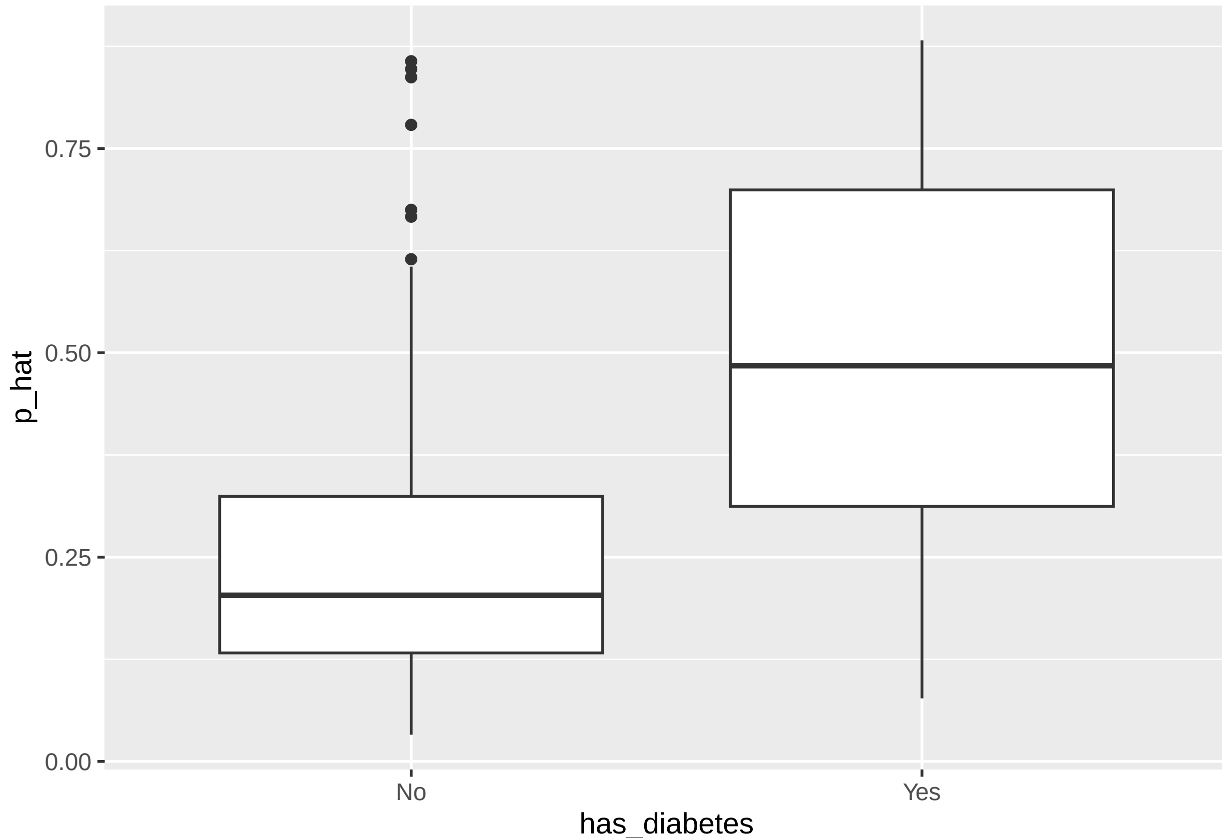
Plot the in-sample log-odds predictions (y-axis) versus the real response values (x-axis).

```r
#TO-DO
p_hat=glm(y ~ X[, "glu"], family = "binomial")$fitted.values
log_odds_hat=log(p_hat/(1-p_hat))
ggplot(data.frame(log_odds_hat=log_odds_hat, has_diabetes=pima$type))+
  aes(x=has_diabetes, y=log_odds_hat)+
  geom_boxplot()
```



Plot the in-sample probability predictions (y-axis) versus the real response values (x-axis).

```
#TO-DO
ggplot(data.frame(p_hat=p_hat, has_diabetes=pima$type))+
  aes(x=has_diabetes, y=p_hat)+
  geom_boxplot()
```



Comment on how well you think the logistic regression performed in-sample.

It did pretty bad because for the ones it predicted yes it is at .50 which is really bad. For the ones it says no to it's about .22 which is better but still bad.

Calculate the in-sample Brier score.

```r
print(mean((y - p_hat)^2))
```

```
## [1] 0.1720724
```

Calculate the in-sample log-scoring rule.

```r
mean(y * log(p_hat) + (1 - y) * log(1 - p_hat))
```

```
## [1] -0.5184318
```

Run a probit regression of y=1 (type is "diabetic") on all variables using R's built-in function and report the b vector.

```r
probit_model = glm(y ~ ., data = pima, family = binomial(link = "probit"))
b_vector_probit = coef(probit_model)

b_vector_probit
```

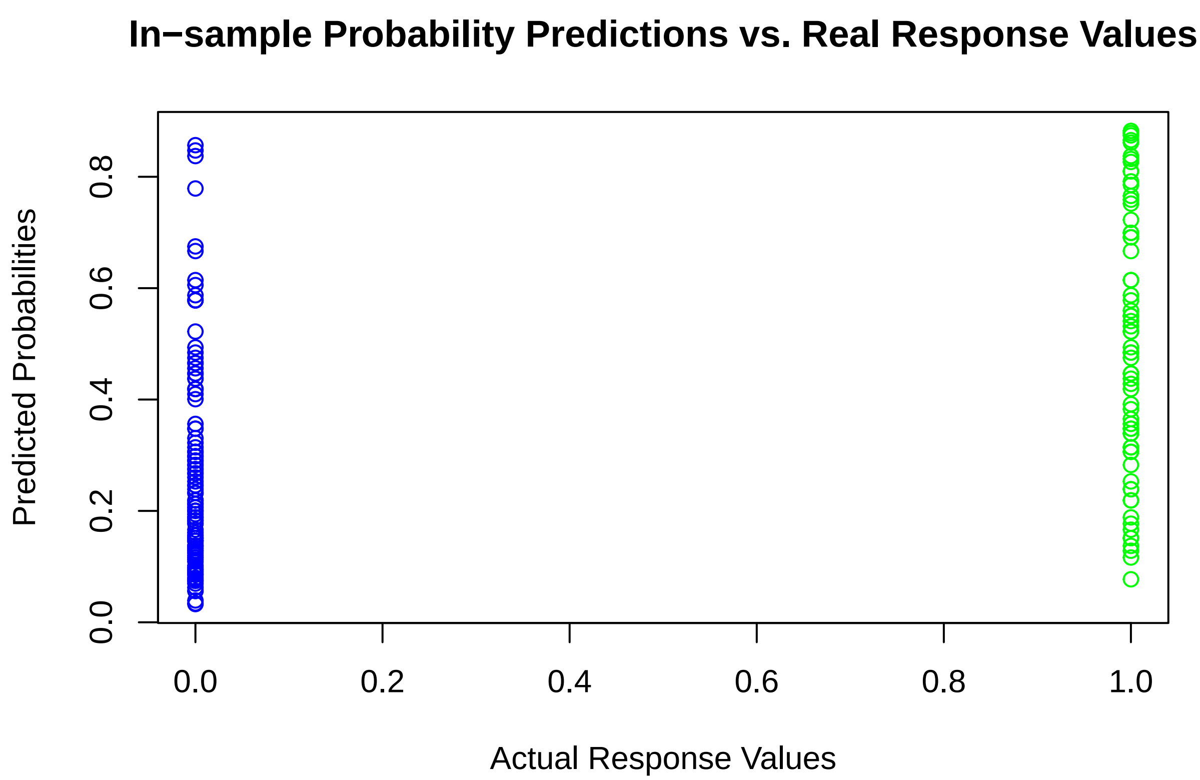
```
##   (Intercept)         npreg           glu            bp          skin
## -6.991223e+00  8.238242e-12 -1.073294e-13  2.665216e-13 -5.426212e-13
##           bmi           ped           age       typeYes
##  1.033327e-12  1.320862e-11 -1.711495e-12  1.398244e+01
```

Does the weight estimates here in the probit fit have different signs than the weight estimates in the logistic fit? What does that mean?

#TO-DO

Plot the in-sample probability predictions (y-axis) versus the real response values (x-axis).

```r
# Plot with different colors for actual and predicted values
plot(y, p_hat, xlab = "Actual Response Values", ylab = "Predicted Probabilities", main = "In-sample Prol
```



Calculate the in-sample Brier score.

```r
#TO-DO
p_hat_probit=glm(y ~ X[, "glu"], family = "binomial"(link= "probit"))$fitted.values
print(mean((y - p_hat_probit)^2))
```

```
## [1] 0.1720422
```

Calculate the in-sample log-scoring rule.

```r
#TO-DO
mean(y * log(p_hat_probit) + (1 - y) * log(1 - p_hat_probit))
```

```
## [1] -0.5180971
```

Which model did better in-sample?

They performed about the same

Compare both model oos using the Brier score and a test set with 1/3 of the data.

```r
set.seed(123)

# Split the data into training (2/3) and test (1/3) sets
train_indices = sample(1:nrow(pima), size = round(2/3 * nrow(pima)))
train_data = pima[train_indices, ]
test_data = pima[-train_indices, ]

# Fit logistic regression model on the training set
fit_logistic = glm(y ~ ., data = pima, family = binomial)
```

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

```r
p_hat_logistic = predict(fit_logistic, newdata = test_data, type = "response")

# Calculate Brier score for logistic regression model
brier_score_logistic = mean((test_data$y - p_hat_logistic)^2)

# Fit probit regression model on the training set
fit_probit <- glm(y ~ ., data = pima, family = binomial(link = "probit"))
p_hat_probit <- predict(fit_probit, newdata = test_data, type = "response")
brier_score_probit <- mean((test_data$y - p_hat_probit)^2)

# Print the Brier scores for both models
print(paste("Brier score for logistic regression model:", brier_score_logistic))
```

```
## [1] "Brier score for logistic regression model: NaN"
```

```r
print(paste("Brier score for probit regression model:", brier_score_probit))
```

```
## [1] "Brier score for probit regression model: NaN"
```

Which model did better oos?

#TO-DO I'm assuming they perform about the same because their in-sample was the exact same.