Lab 7

Tziporah Horowitz

Revised

NOTE: I revised this lab because I did not have a computer to run R when I originally did this lab

Run three OLS models on the boston housing data using all available features:

(1) where the response is medv,
(2) where the response is the log base 10 of medv and
(3) where the response is the square root of medv.

Compare the two models on oos se of the residuals. Use K = 5 to create a training-test split. Which model is better?

```
pacman::p_load(MASS)
boston <- Boston

n <- nrow(boston)
K <- 5

test_indices <- sample(1 : n, 1 / K * n)
train_indices <- setdiff(1 : n, test_indices)

X <- boston[, 1:13]
y <- boston$medv

X_train <- X[train_indices, ]
y_train <- y[train_indices]
X_test <- X[test_indices, ]
y_test <- y[test_indices]

mod1 <- lm(y_train ~ ., X_train)
mod2 <- lm(log10(y_train) ~ ., X_train)
mod3 <- lm(sqrt(y_train) ~ ., X_train)

yhat_oos1 <- predict(mod1, X_test)
yhat_oos2 <- predict(mod2, X_test)
yhat_oos3 <- predict(mod3, X_test)

oos_residuals1 = y_test - yhat_oos1
oos_residuals2 = y_test - yhat_oos2
oos_residuals3 = y_test - yhat_oos3

sd(oos_residuals1)
```

```
## [1] 3.838264
```

```
sd(oos_residuals2)
```

```
## [1] 7.757975
```

```
sd(oos_residuals3)
```

```
## [1] 7.157177
```

When evaluating the models out of sample, did you ever extrapolate? Which predictions specifically in your test set were extrapolations? How "bad" were the exrapolations?

```
for (i in 1:13){
  if (max(X_train[, i]) < max(X_test[, i]) | min(X_train[, i]) > min(X_test[, i])){
    extrapolate = TRUE
  }
  else extrapolate = FALSE
  if (extrapolate == TRUE){
    print(colnames(boston)[i])
  }
}
```

```
## [1] "age"
```

Regardless of the model that came out better, lets consider the response to be raw medv i.e. without taking a transformation. Run a model that includes all squared features (except `chas` which is binary). Does this model do better than vanilla OLS from question 1?

```
colnames(boston)
```

```
##  [1] "crim"    "zn"      "indus"   "chas"    "nox"     "rm"      "age"
##  [8] "dis"     "rad"     "tax"     "ptratio" "black"   "lstat"   "medv"
```

```
mod4 <- lm(medv ~ poly(crim, 2) + poly(zn, 2) + poly(indus, 2) + chas + poly(nox, 2) +
             poly(rm, 2) + poly(age, 2) + poly(dis, 2) + poly(rad, 2) + poly(tax, 2) +
             poly(ptratio, 2) + poly(black, 2) + poly(lstat, 2), boston)

summary(mod4)$sigma
```

```
## [1] 3.875402
```

```
summary(mod4)$r.squared
```

```
## [1] 0.8312355
```

Run a model that includes all polynomial functions of degree 3 of all features (except `chas` which is binary). Does this model do better than the degree 2 polynomial function of the previous question?

```
mod5 <- lm(medv ~ poly(crim, 3) + poly(zn, 3) + poly(indus, 3) + chas + poly(nox, 3) +
             poly(rm, 3) + poly(age, 3) + poly(dis, 3) + poly(rad, 3) + poly(tax, 3) +
             poly(ptratio, 3) + poly(black, 3) + poly(lstat, 3), boston)

summary(mod5)$sigma
```

```
## [1] 3.757808
```

```
summary(mod5)$r.squared
```

```
## [1] 0.8452889
```

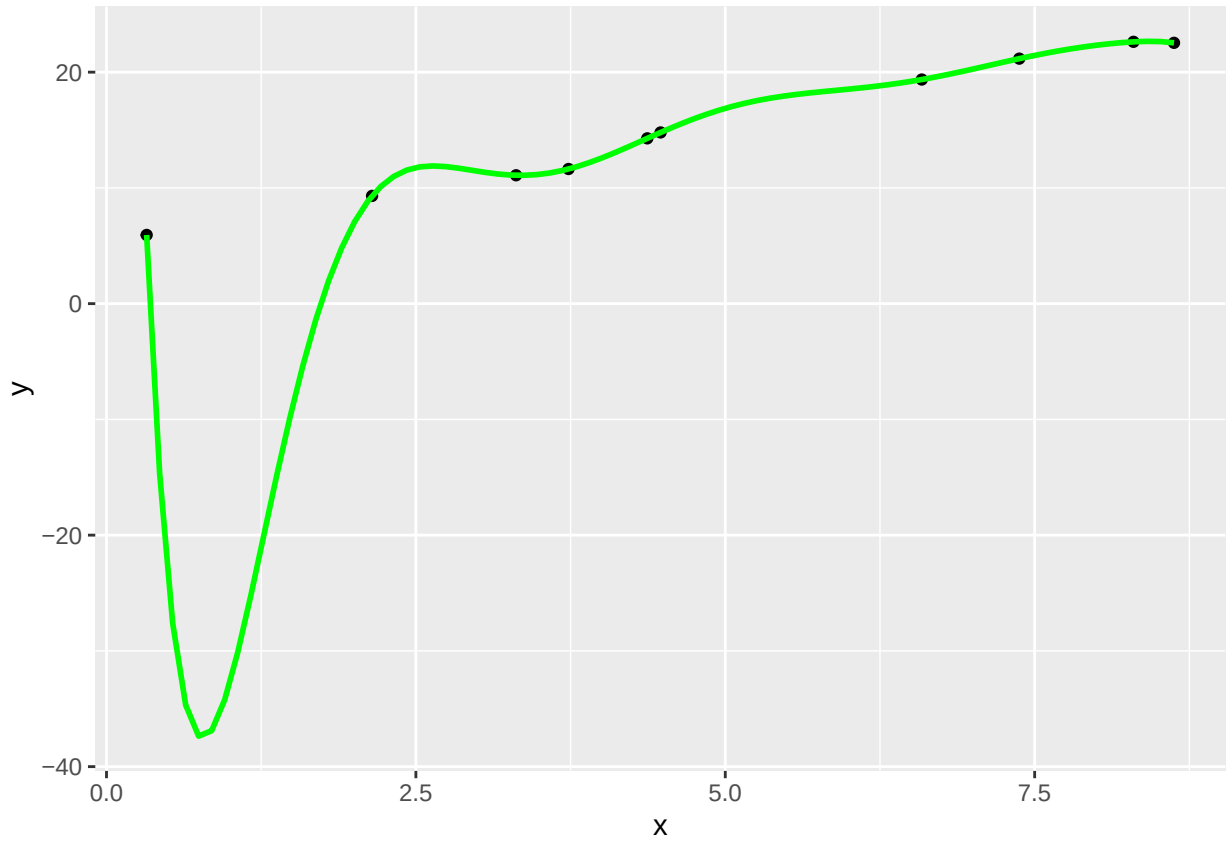
Use polynomial regression to perfectly fitting the following data:

```
n = 10
set.seed(1984)
x = runif(n, 0, 10)
y = 5 + 2 * x + rnorm(n)

mod6 <- y ~ poly(x, 9)
summary(lm(mod6))
```

```
##
## Call:
## lm(formula = mod6)
##
## Residuals:
## ALL 10 residuals are 0: no residual degrees of freedom!
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.27341         NA      NA       NA
## poly(x, 9)1 17.59202         NA      NA       NA
## poly(x, 9)2  0.25427         NA      NA       NA
## poly(x, 9)3 -1.38251         NA      NA       NA
## poly(x, 9)4 -0.36392         NA      NA       NA
## poly(x, 9)5  0.38060         NA      NA       NA
## poly(x, 9)6 -0.88106         NA      NA       NA
## poly(x, 9)7 -0.01316         NA      NA       NA
## poly(x, 9)8  0.59918         NA      NA       NA
## poly(x, 9)9 -0.02908         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:    NaN
## F-statistic:   NaN on 9 and 0 DF,  p-value: NA
```

Illustrate Runge's phenomenon in this model by scatterplotting the data with $g(x)$ overlaid in green.

```
pacman::p_load(ggplot2)

ggplot(, aes(x = x, y = y)) + geom_point() + geom_smooth(method = lm, formula = mod6, col = "green")
```



For the rest of this assignment, 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. Briefly summarize the documentation on this dataset. What is the data type of each variable? What is the response variable?

```
pacman::p_load(ggthemes)
pacman::p_load(carData)

X <- na.omit(GSSvocab)

str(X)
```

```
## 'data.frame':    27360 obs. of  8 variables:
##  $ year      : Factor w/ 20 levels "1978","1982",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ gender    : Factor w/ 2 levels "female","male": 1 1 2 1 1 2 2 2 1 2 ...
##  $ nativeBorn: Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
##  $ ageGroup  : Factor w/ 5 levels "18-29","30-39",..: 4 5 2 4 3 1 1 4 3 1 ...
##  $ educGroup : Factor w/ 5 levels "<12 yrs","12 yrs",..: 2 1 1 2 2 2 2 2 4 2 ...
##  $ vocab     : num  10 6 4 9 6 6 4 7 8 3 ...
##  $ age       : num  52 74 35 50 41 19 19 59 49 21 ...
##  $ educ      : num  12 9 10 12 12 12 12 12 16 12 ...
```

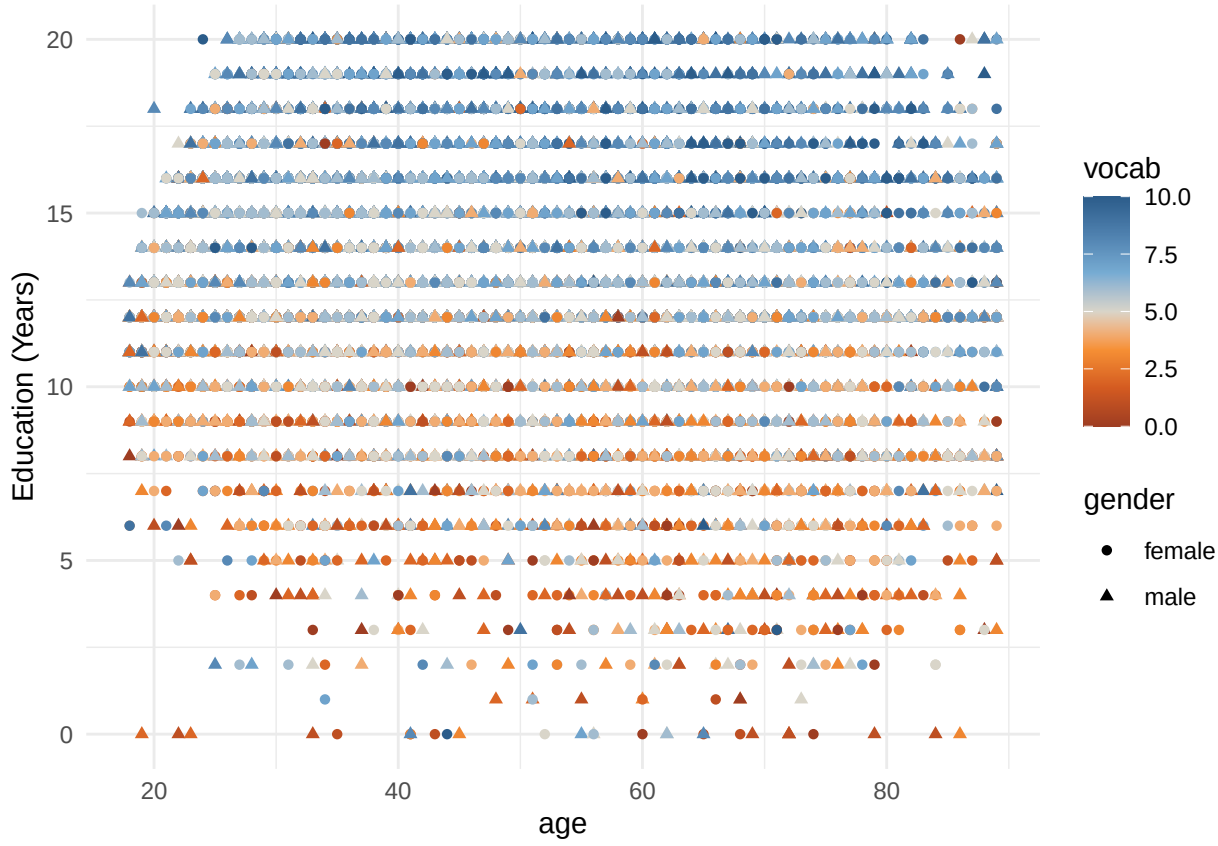
```
##  - attr(*, "na.action")= 'omit' Named int  179 193 258 329 346 389 390 408 429 451 ...
##   ..- attr(*, "names")= chr  "1978.179" "1978.193" "1978.258" "1978.329" ...
```

```
summary(X)
```

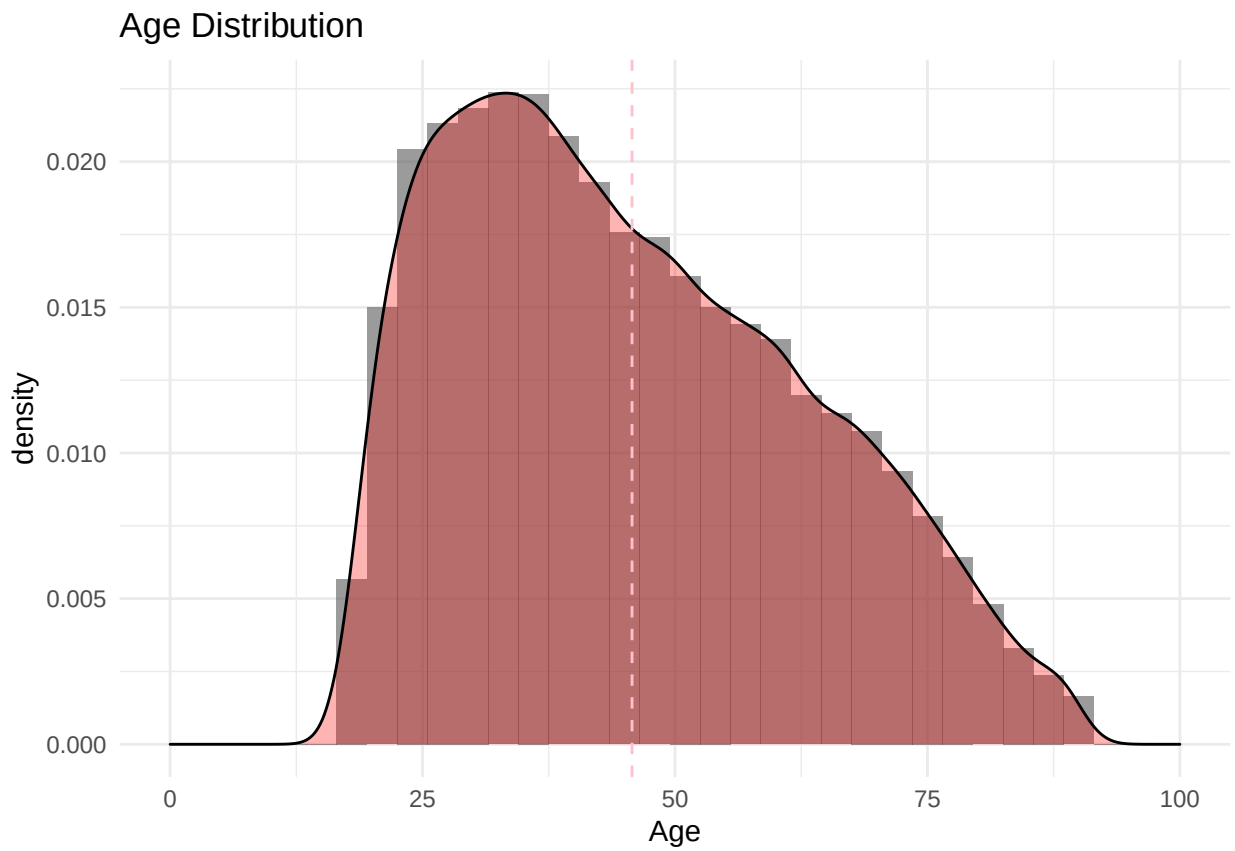
```
##       year          gender      nativeBorn   ageGroup        educGroup
##  2016   : 1856   female:15512   no : 2342   18-29:5691   <12 yrs  :5264
##  1996   : 1855   male  :11848   yes:25018   30-39:6024   12 yrs   :8259
##  1994   : 1842                               40-49:5035   13-15 yrs:6942
##  1982   : 1714                               50-59:4113   16 yrs   :3814
##  1987   : 1659                               60+  :6497   >16 yrs  :3081
##  2014   : 1650
##  (Other):16784
##      vocab          age             educ
##  Min.   : 0   Min.   :18.00   Min.   : 0.00
##  1st Qu.: 5   1st Qu.:31.00   1st Qu.:12.00
##  Median : 6   Median :43.00   Median :13.00
##  Mean   : 6   Mean   :45.75   Mean   :13.16
##  3rd Qu.: 7   3rd Qu.:59.00   3rd Qu.:16.00
##  Max.   :10   Max.   :89.00   Max.   :20.00
##
```

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.

```
ggplot(X) + geom_point(aes(x = age, y = educ, col = vocab, shape = gender)) +
  ylab("Education (Years)") +
  scale_color_gradient2_tableau() +
  theme_minimal()
```




```
ggplot(X, aes(x = age)) +
  geom_histogram(aes(y=..density..), alpha = .6, binwidth = 3, na.rm = TRUE) +
  geom_density(alpha = .3, fill = "red") +
  xlim(0, 100) +
  xlab("Age") +
  ggtitle("Age Distribution") +
  geom_vline(xintercept = mean(X$age), col = "pink", linetype = "dashed") +
  theme_minimal()
```

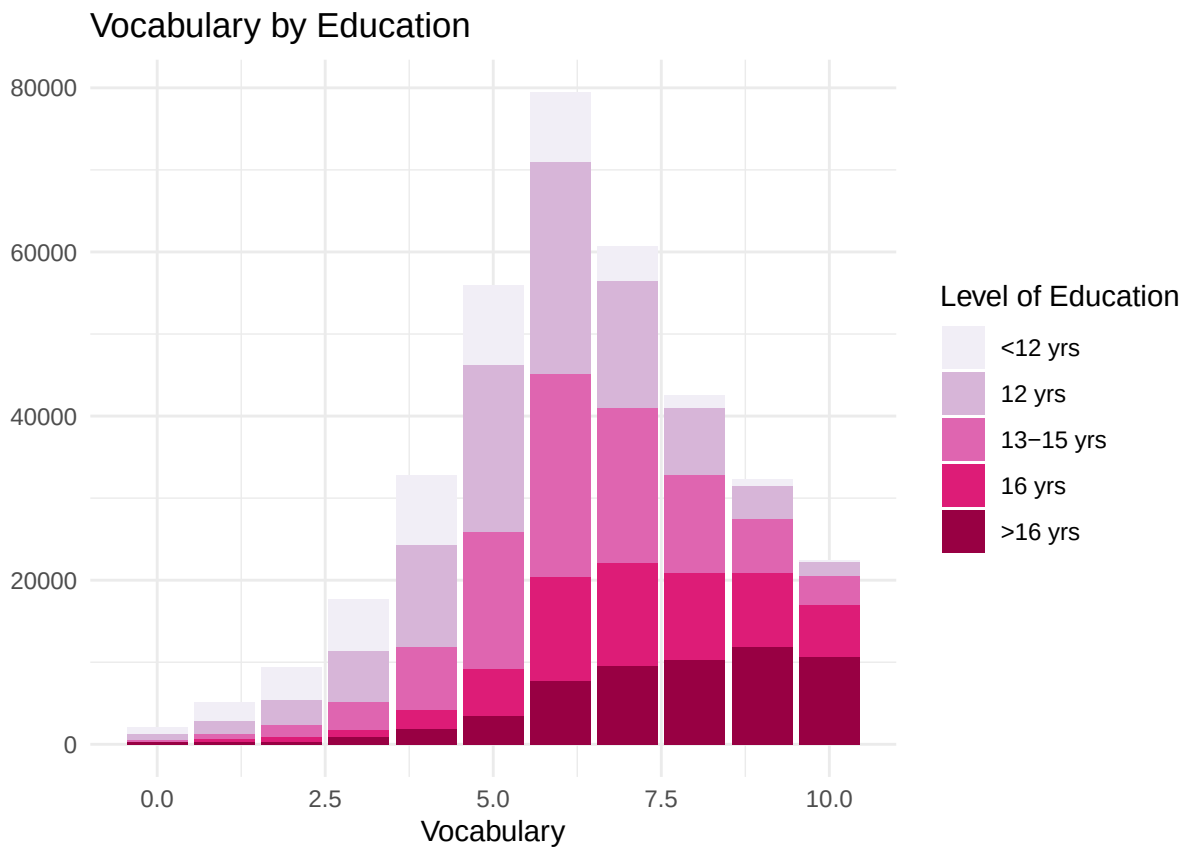



```
ggsave("age.pdf")
```

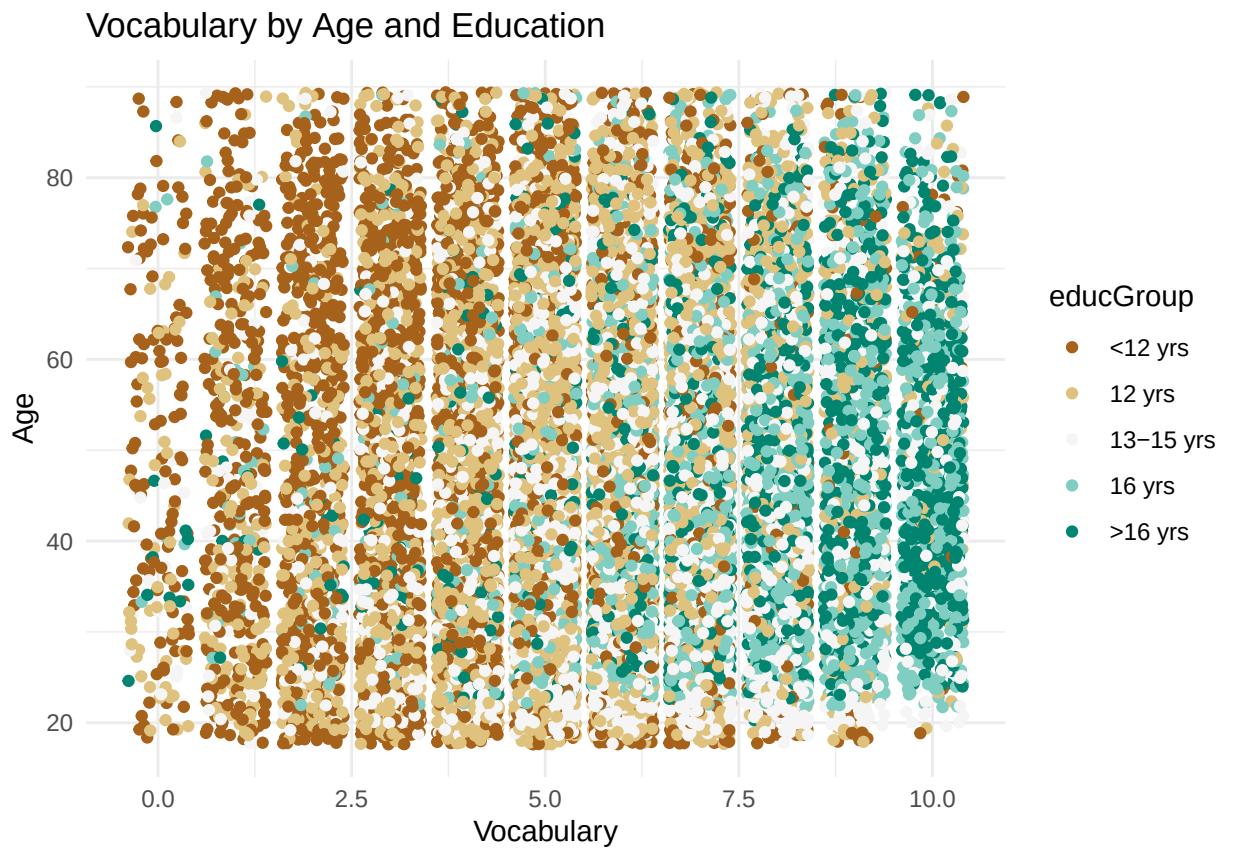
```
## Saving 6.5 x 4.5 in image
```

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.

```
ggplot(X) + geom_col(aes(x = vocab, y = educ, fill = educGroup)) +
  xlab("Vocabulary") +
  ylab("") +
  ggtitle("Vocabulary by Education") +
  labs(fill = "Level of Education") +
  scale_fill_brewer(palette = "PuRd") +
  theme_minimal()
```




```
ggplot(X) +
  geom_jitter(aes(x = vocab, y = age, col = educGroup)) +
  xlab("Vocabulary") +
  ylab("Age") +
  ggtitle("Vocabulary by Age and Education") +
  scale_color_brewer(palette = "BrBG") +
  theme_minimal()
```

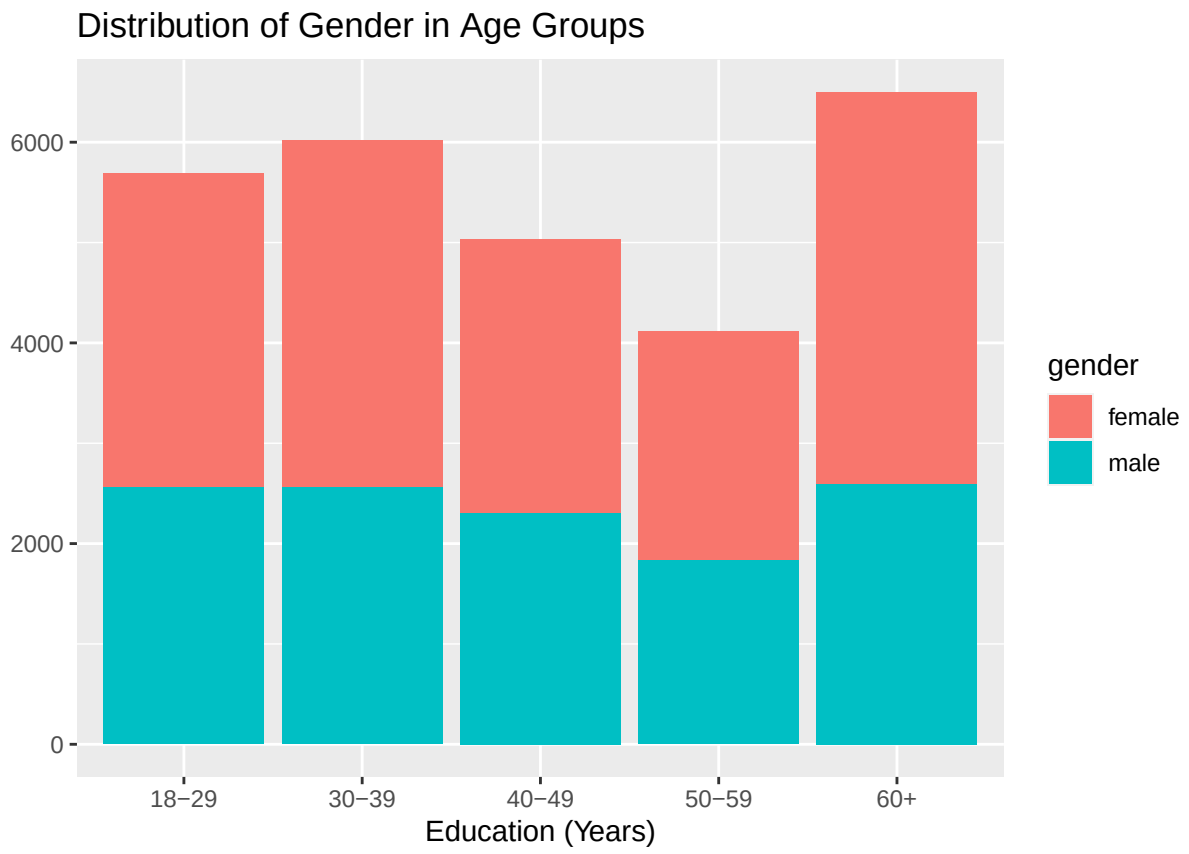



```
ggsave("vocab.pdf")
```

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

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) + geom_bar(aes(x = ageGroup,  fill = gender)) +
  xlab("Education (Years)") +
  ylab("") +
  ggtitle("Distribution of Gender in Age Groups")
```




```
  theme_minimal()
```

```
## List of 93
##  $ line                        :List of 6
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ lineend      : chr "butt"
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ rect                        :List of 5
##   ..$ fill         : chr "white"
##   ..$ colour       : chr "black"
##   ..$ size         : num 0.5
##   ..$ linetype     : num 1
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_rect" "element"
##  $ text                        :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
```

```
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                     : NULL
##  $ aspect.ratio              : NULL
##  $ axis.title                : NULL
##  $ axis.title.x              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.75pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.top          :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 2.75pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.x.bottom       : NULL
##  $ axis.title.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 2.75pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
```

```
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title.y.left         : NULL
##  $ axis.title.y.right        :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 0pt 2.75pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text                 :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey30"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 2.2pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.top           :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : num 0
```

```
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 2.2pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.x.bottom        : NULL
##  $ axis.text.y               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 1
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 2.2pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.text.y.left          : NULL
##  $ axis.text.y.right         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 0pt 2.2pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.ticks                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.x              : NULL
##  $ axis.ticks.x.top          : NULL
##  $ axis.ticks.x.bottom       : NULL
##  $ axis.ticks.y              : NULL
##  $ axis.ticks.y.left         : NULL
##  $ axis.ticks.y.right        : NULL
##  $ axis.ticks.length         : 'unit' num 2.75pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
```

```
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ axis.line                 : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.line.x               : NULL
##  $ axis.line.x.top           : NULL
##  $ axis.line.x.bottom        : NULL
##  $ axis.line.y               : NULL
##  $ axis.line.y.left          : NULL
##  $ axis.line.y.right         : NULL
##  $ legend.background         : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.margin             : 'margin' num [1:4] 5.5pt 5.5pt 5.5pt 5.5pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ legend.spacing            : 'unit' num 11pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'unit' num 1.2lines
##   ..- attr(*, "valid.unit")= int 3
##   ..- attr(*, "unit")= chr "lines"
##  $ legend.key.height         : NULL
##  $ legend.key.width          : NULL
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.text.align         : NULL
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
```

```
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title.align        : NULL
##  $ legend.position           : chr "right"
##  $ legend.direction          : NULL
##  $ legend.justification      : chr "center"
##  $ legend.box                : NULL
##  $ legend.box.just           : NULL
##  $ legend.box.margin         : 'margin' num [1:4] 0cm 0cm 0cm 0cm
##   ..- attr(*, "valid.unit")= int 1
##   ..- attr(*, "unit")= chr "cm"
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'unit' num 11pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ panel.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'unit' num 5.5pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ size         : NULL
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major          : NULL
##  $ panel.grid.minor          :List of 6
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.5
##   ..$ linetype     : NULL
##   ..$ lineend      : NULL
##   ..$ arrow        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_line" "element"
##  $ panel.grid.major.x        : NULL
##  $ panel.grid.major.y        : NULL
##  $ panel.grid.minor.x        : NULL
##  $ panel.grid.minor.y        : NULL
##  $ panel.ontop               : logi FALSE
##  $ plot.background           : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
```

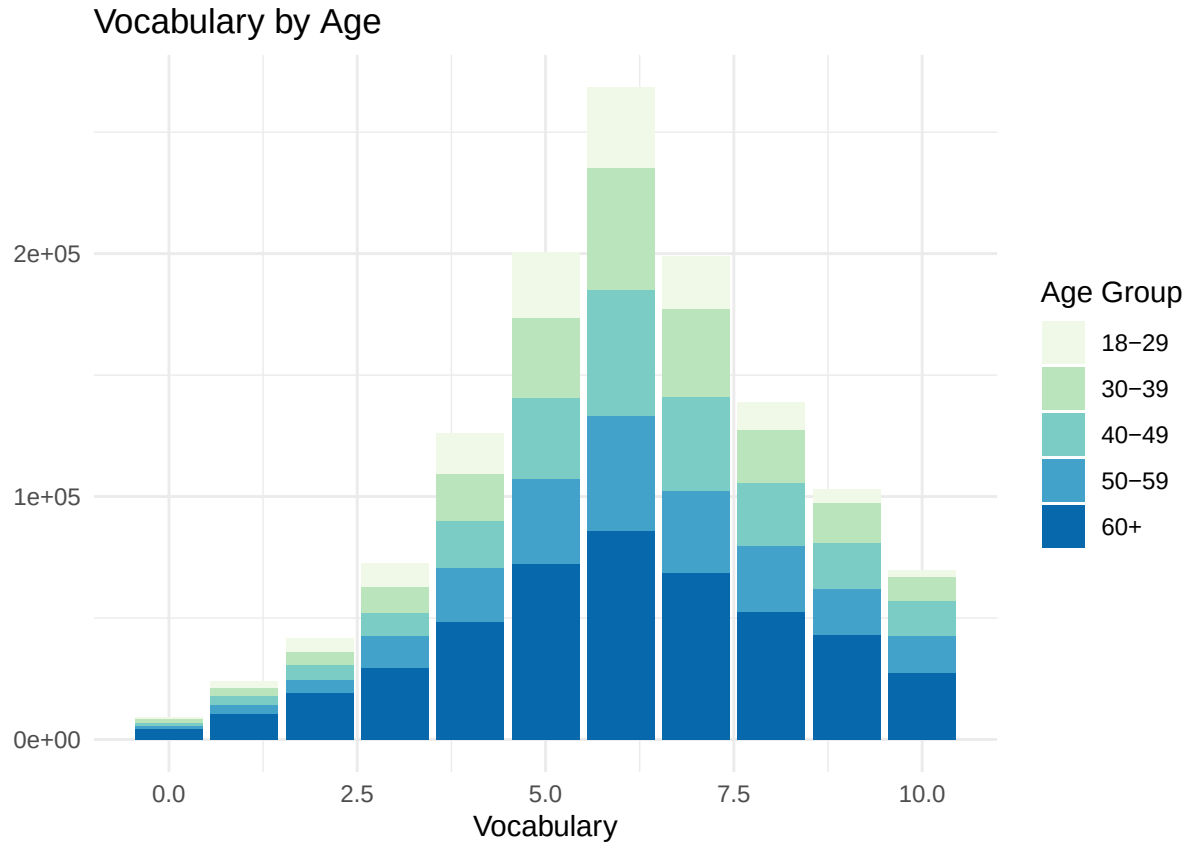
```
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 5.5pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.title.position       : chr "panel"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 5.5pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption.position     : chr "panel"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
```

```
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 5.5pt 5.5pt 5.5pt 5.5pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.placement           : chr "inside"
##  $ strip.text                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : chr "grey10"
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 4.4pt 4.4pt 4.4pt 4.4pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.y              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num -90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'unit' num 2.75pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ strip.switch.pad.wrap     : 'unit' num 2.75pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ strip.text.y.left         :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : num 90
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
```

```
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

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

```
ggplot(X) + geom_col(aes(x = vocab, y = age, fill = ageGroup)) +
  xlab("Vocabulary") +
  ylab("") +
  ggtitle("Vocabulary by Age") +
  labs(fill = "Age Group") +
  scale_fill_brewer(palette = "GnBu") +
  theme_minimal()
```

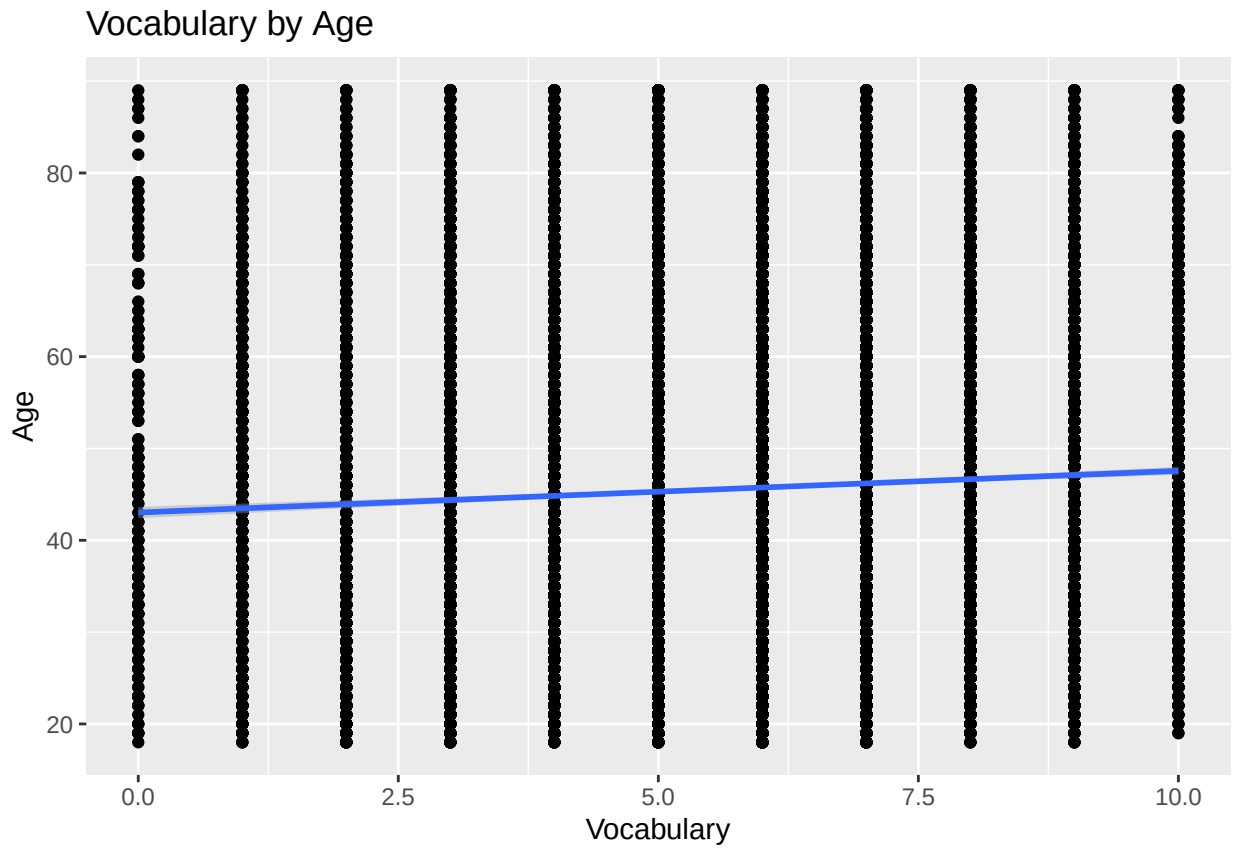



```
ageplot <- ggplot(X, aes(x = vocab, y = age)) +
  geom_point() +
  xlab("Vocabulary") +
  ylab("Age") +
  ggtitle("Vocabulary by Age")
ageplot
```

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

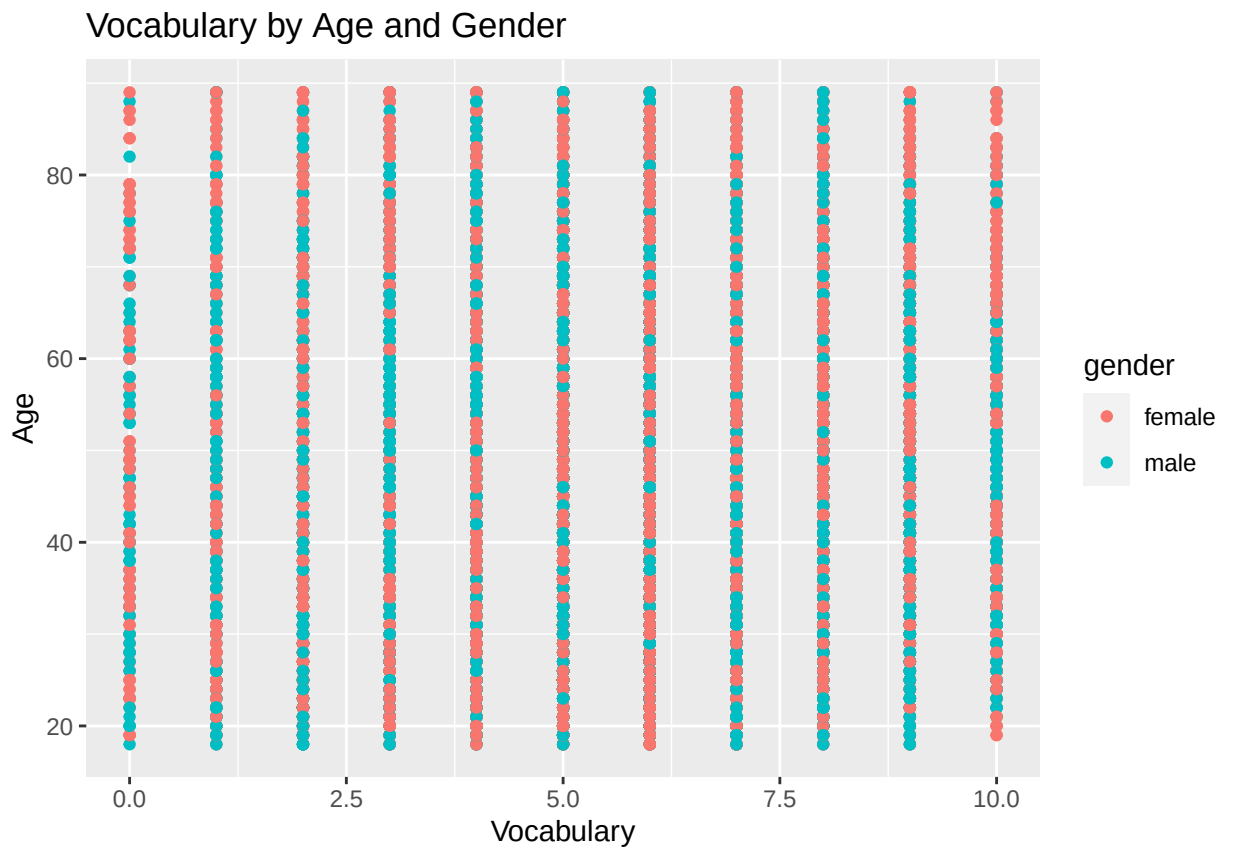
```
ageplot + geom_smooth(method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



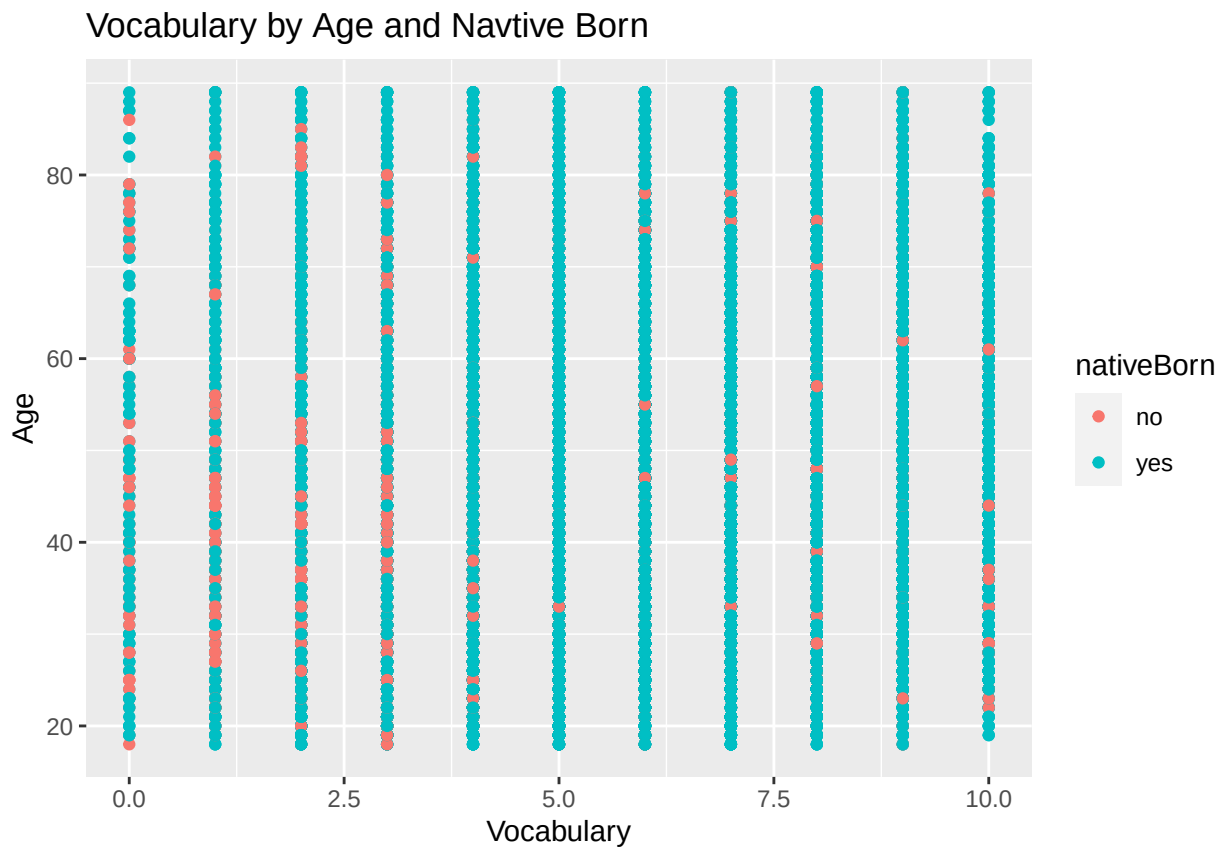
Using the plot from the previous question, create the best looking overloading with variable `gender`. Does there appear to be an interaction of `gender` and `age`?

```
ageplot + aes(col = gender) + ggtitle("Vocabulary by Age and Gender")
```



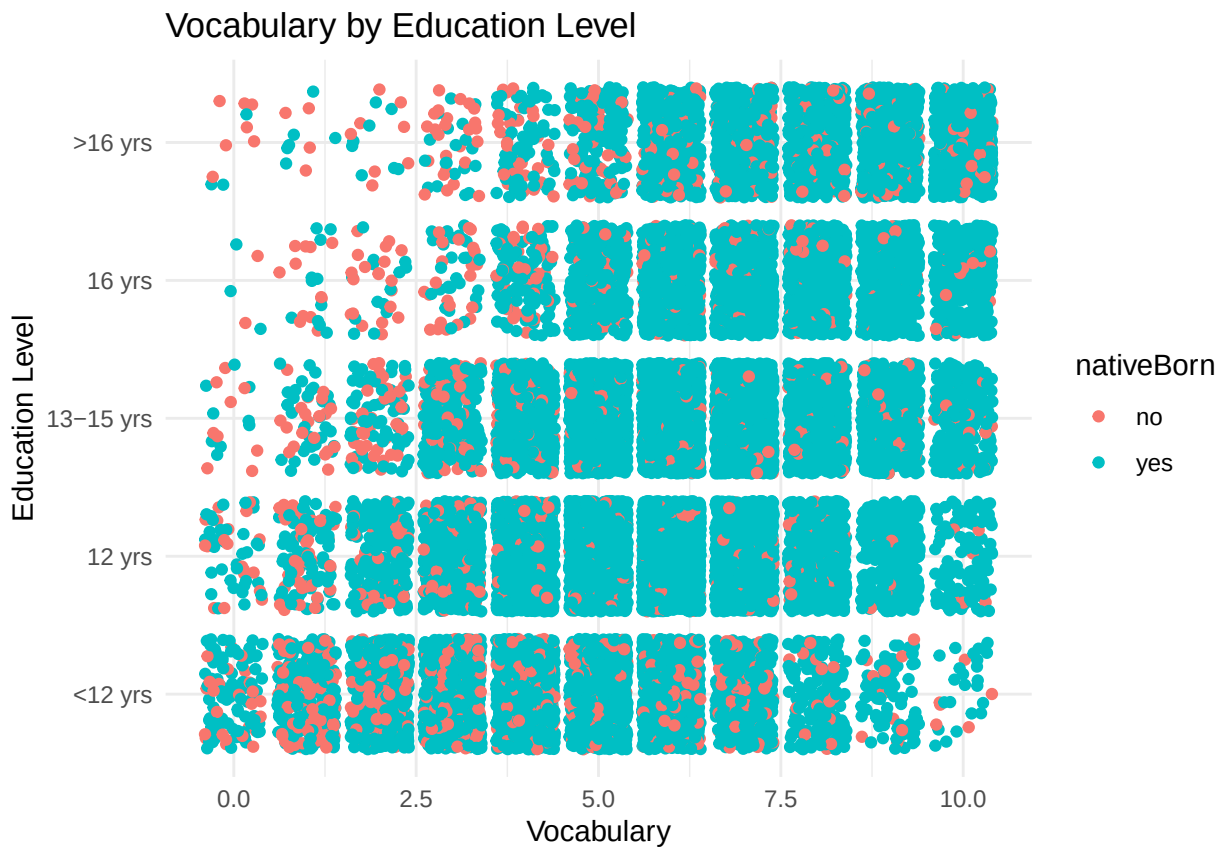
Using the plot from the previous question, create the best looking overloading with variable `nativeBorn`. Does there appear to be an interaction of `nativeBorn` and `age`?

```
ageplot + aes(col = nativeBorn) + ggtitle("Vocabulary by Age and Navtive Born")
```

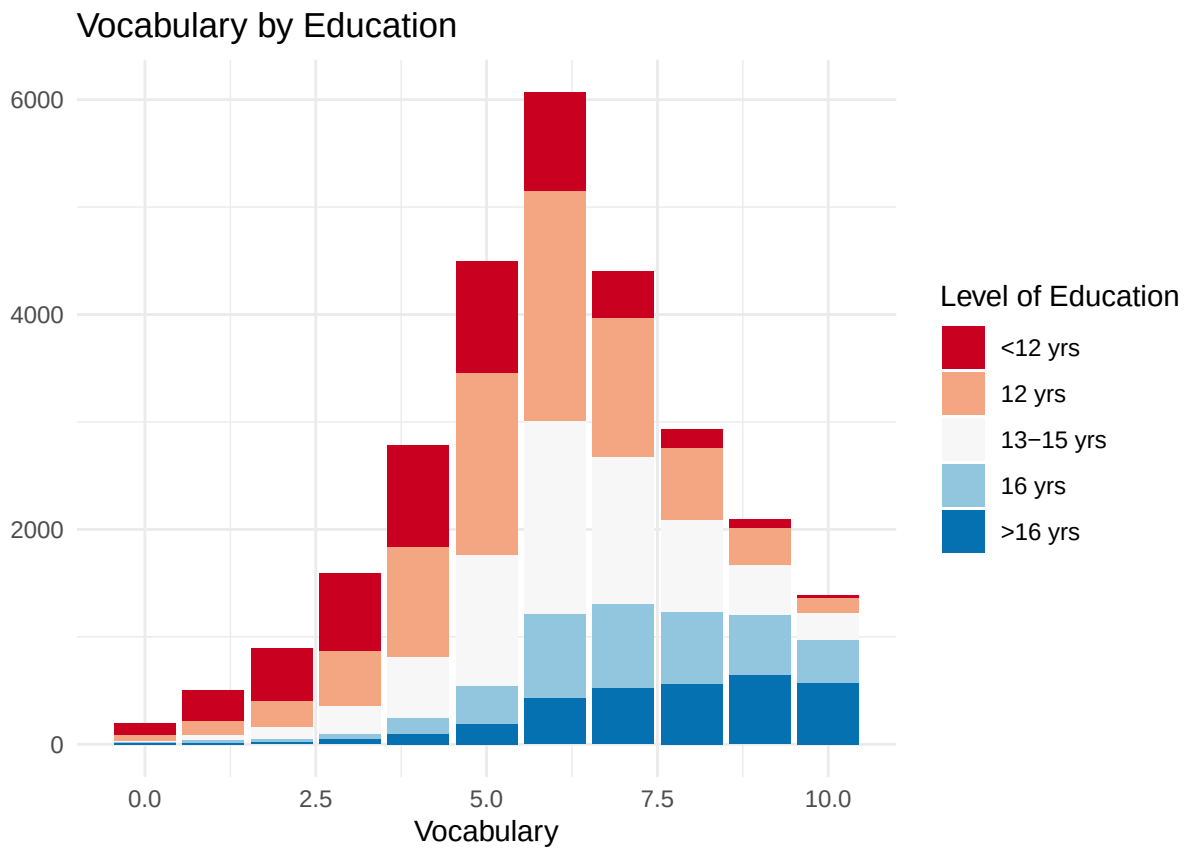


Create two different plots and identify the best-looking plot you can to examine the `vocab` variable by `educGroup`. Does there appear to be an association?

```
ggplot(X) +
  geom_jitter(aes(x = vocab, y = educGroup, col = nativeBorn)) +
  xlab("Vocabulary") +
  ylab("Education Level") +
  ggtitle("Vocabulary by Education Level") +
  theme_minimal()
```

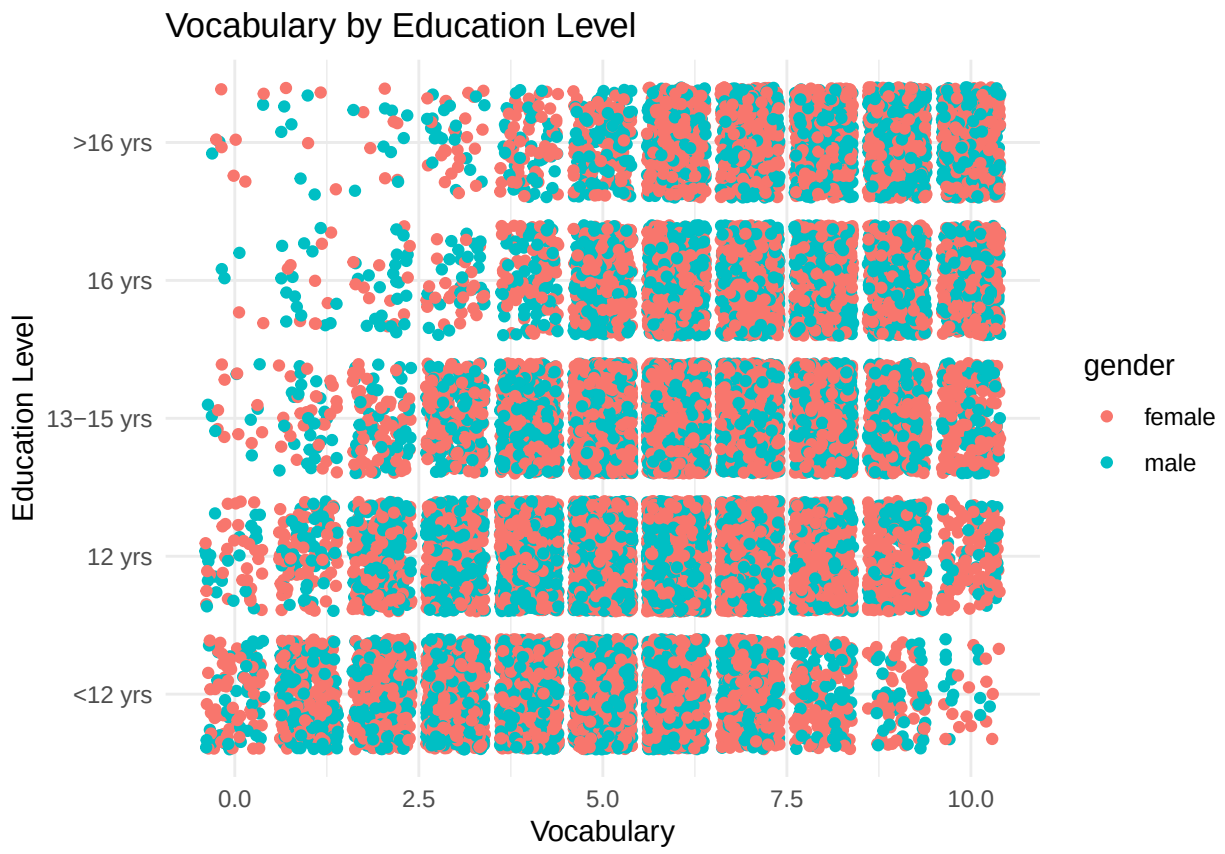



```
ggplot(X) + geom_bar(aes(x = vocab, fill = educGroup)) +
  xlab("Vocabulary") +
  ylab("") +
  ggtitle("Vocabulary by Education") +
  labs(fill = "Level of Education") +
  scale_fill_brewer(palette = "RdBu") +
  theme_minimal()
```



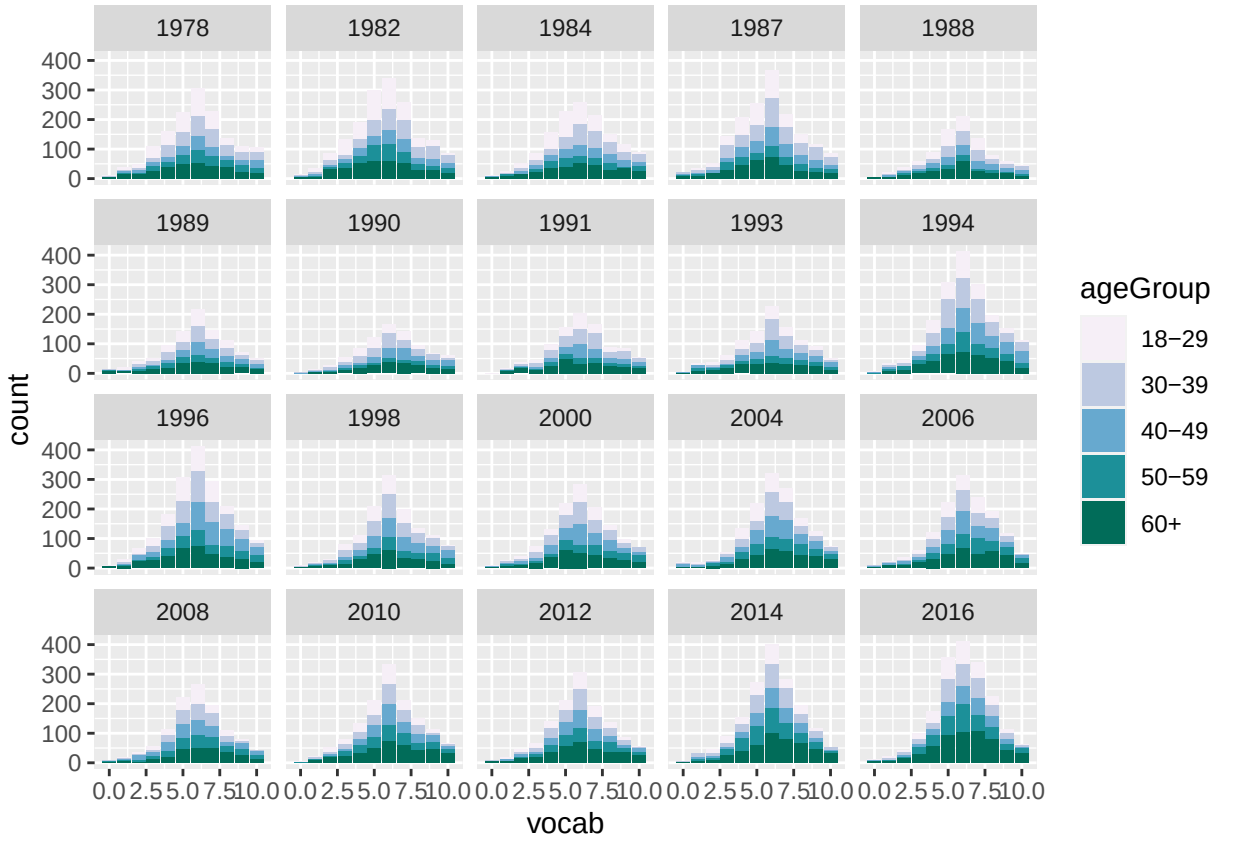
Using the best-looking plot from the previous question, create the best looking overloading with variable `gender`. Does there appear to be an interaction of `gender` and `educGroup`?

```
ggplot(X) +
  geom_jitter(aes(x = vocab, y = educGroup, col = gender)) +
  xlab("Vocabulary") +
  ylab("Education Level") +
  ggtitle("Vocabulary by Education Level") +
  theme_minimal()
```



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

```
ggplot(X) +
  geom_bar(aes(x = vocab, fill = ageGroup)) +
  facet_wrap(~ year) +
  scale_fill_brewer(palette = "PuBuGn")
```



We will now be getting some experience with speeding up R code using C++ via the `Rcpp` package.

First, clear the workspace and load the `Rcpp` package.

```
rm(list = ls())
pacman::p_load(Rcpp)
```

Create a variable `n` to be 10 and a variable `Nvec` to be 100 initially. Create a random vector via `rnorm` `Nvec` times and load it into a `Nvec` x `n` dimensional matrix.

```
n <- 10
Nvec <- 100
X <- c()
for (i in 1:n){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}

dim(X)
```

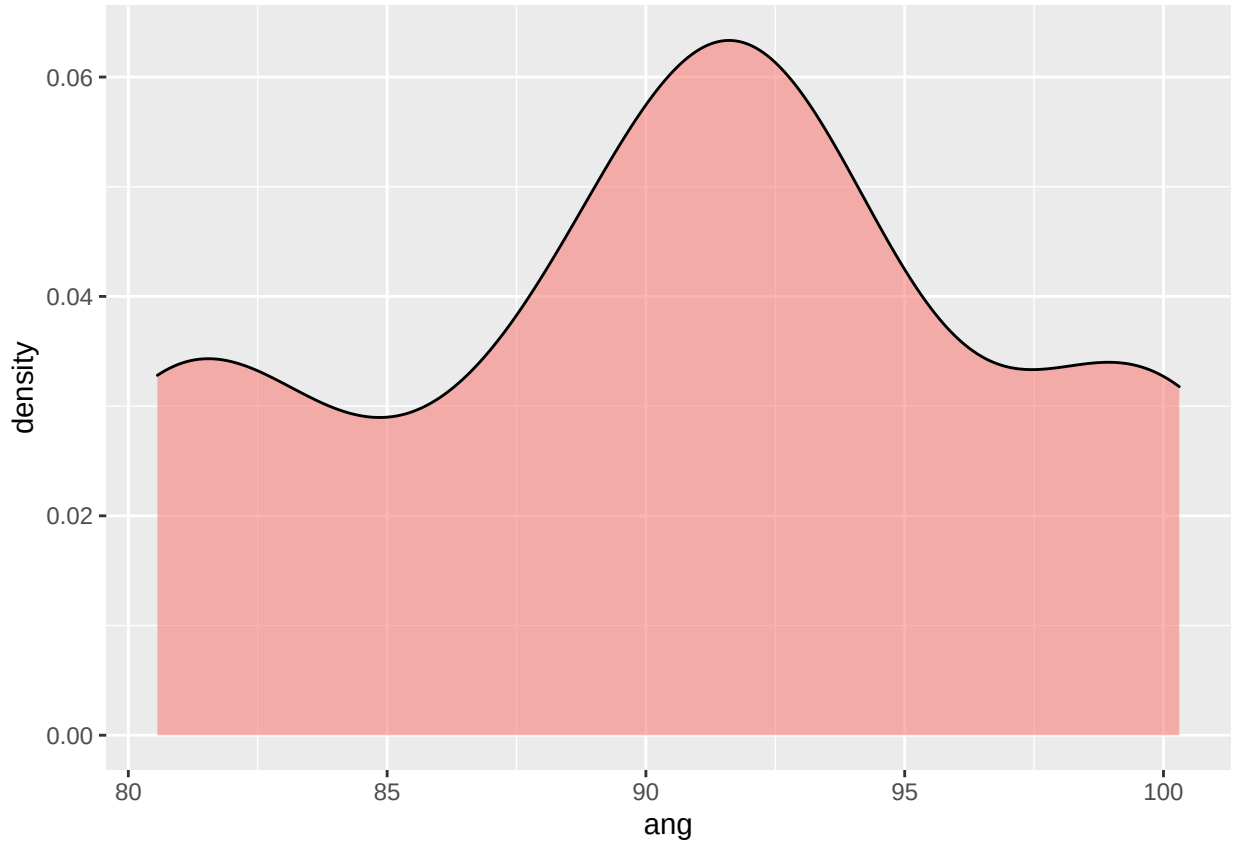
```
## [1] 100  10
```

Write a function `all_angles` that measures the angle between each of the pairs of vectors. You should measure the vector on a scale of 0 to 180 degrees with negative angles coerced to be positive.

```
all_angles <- function(X, n){
  angle <- c()

  for (i in 1:(n-1)){
    theta_in_rad <- acos((X[ , i] %*% X[ , i+1]) / (sqrt(sum(X[ , i]^2)) * sqrt(sum(X[ , i+1]^2))))
    angle <- c(angle, theta_in_rad * 180 / pi)
  }
  angle
}

ang <- all_angles(X, n)
```

Plot the density of these angles.

```
pacman::p_load(ggplot2)

ggplot() + geom_density(aes(x = ang, fill = "red", alpha = .7)) +
  theme(legend.position = "none")
```



Write an Rcpp function `all_angles_cpp` that does the same thing. Use an IDE if you want, but write it below in-line.

```
cppFunction(
'NumericMatrix mmult(const NumericMatrix& m1, const NumericMatrix& m2){
  if (m1.ncol() != m2.nrow()){
```

```
    stop ("Incompatible matrix dimensions");
  }
  NumericMatrix out(m1.nrow(), m2.ncol());
  NumericVector rm1, cm2;
  for (size_t i = 0; i < m1.nrow(); ++i) {
    rm1 = m1(i,_);
    for (size_t j = 0; j < m2.ncol(); ++j) {
      cm2 = m2(_, j);
      out(i,j) = std::inner_product(rm1.begin(), rm1.end(), cm2.begin(), 0.);
    }
  }
  return out;
}'
)
## I'm not sure how function pointers work in Rcpp
cppFunction(
'NumericVector all_angles_cpp(NumericMatrix& X, int n, Function f){
  NumericVector angle = {0};
  int sum1 = 0, sum2 = 0;
  double pi = 3.141592654, theta_in_rad;
  for (int i = 0; i < (n-1); i++){
    for (int j = 0; j < X.nrow(); j++){
      sum1 += X(j, i) * X(j, i);
      sum2 += X(j, i + 1) * X(j, i + 1);
    }
    theta_in_rad = acos( f(X(_, i), X(_, i + 1)) / (sqrt(sum1) * sqrt(sum2)) );
    angle[i] = theta_in_rad * 180 / pi;
  }
  return angle;
}'
)
```

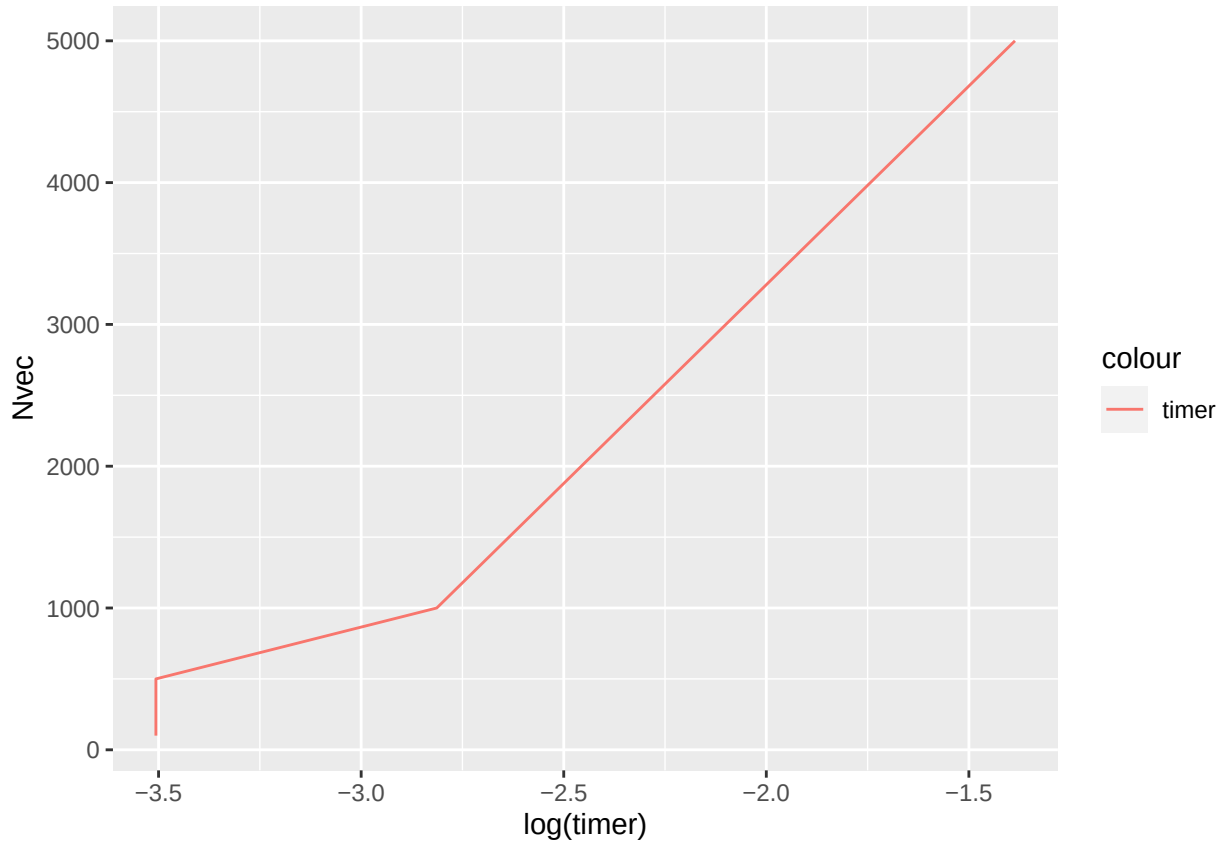
Test the time difference between these functions for `n = 1000` and `Nvec = 100, 500, 1000, 5000`. Store the results in a matrix.

```
n <- 1000
Nvec <- c(100, 500, 1000, 5000)
timer <- c()
timecpp <- c()
for (i in 1:length(Nvec)){
  X <- c()
  for (j in 1:n){
    x <- rnorm(Nvec[i])
    X <- cbind(X, x)
  }
  timer <- c(timer, unname(system.time({
    anglesr = all_angles(X, n)
  }))[3])
  # timecpp <- c(timecpp, unname(system.time({
  #   anglescpp = all_angles_cpp(X, n, mmult)
  # }))[3])
}
```

Plot the divergence of performance (in log seconds) over n using a line geometry. Use two different colors

for the R and CPP functions. Make sure there's a color legend on your plot.

```
pacman::p_load(ggplot2)
ggplot() +
  geom_line(aes(y = Nvec, x = log(timer), col = "timer")) #+
```

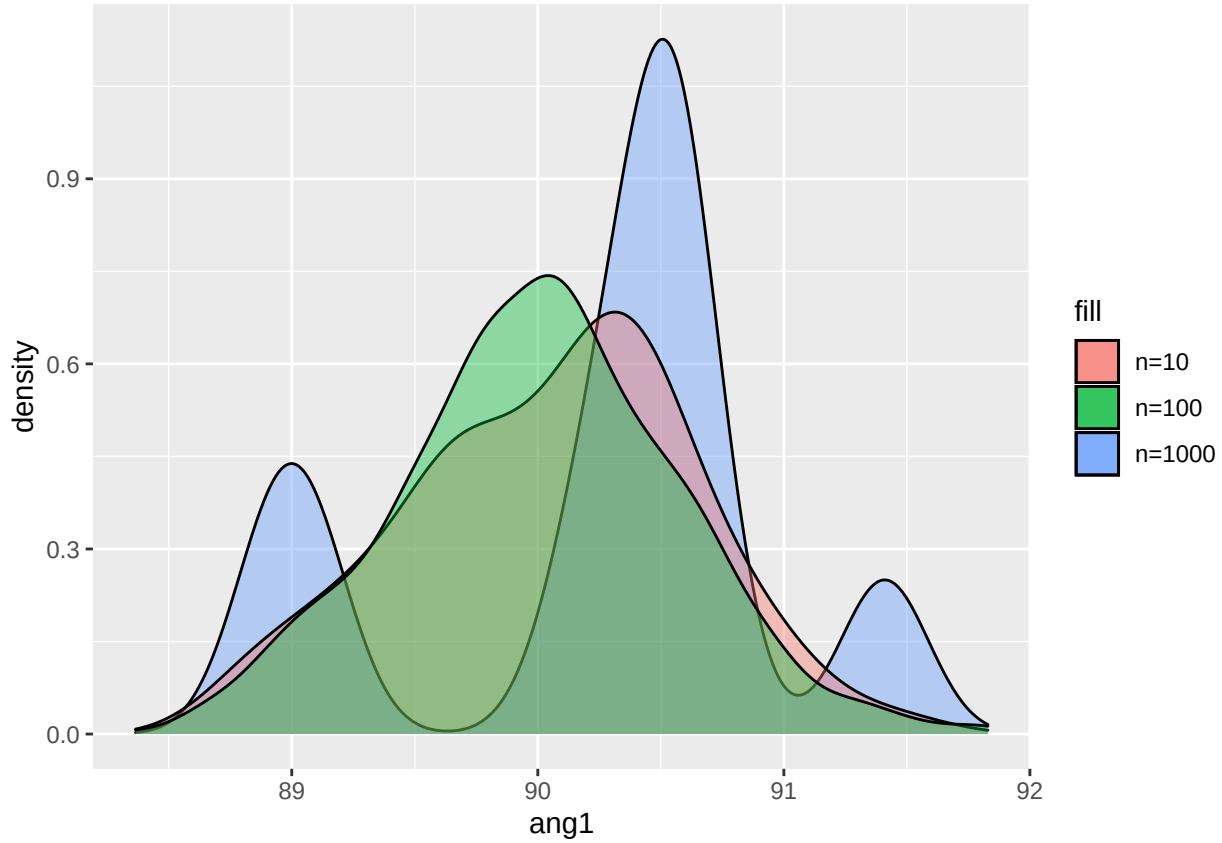



```
  # geom_line(aes(y = Nvec, x = log(timecpp), col = "timecpp"))
```

Let `Nvec = 10000` and vary `n` to be 10, 100, 1000. Plot the density of angles for all three values of `n` on one plot using color to signify `n`. Make sure you have a color legend. This is not easy.

```
Nvec = 10000
X <- c()
for (i in 1:10){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang1 <- all_angles(X, 10)
X <- c()
for (i in 1:100){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang2 <- all_angles(X, 100)
X <- c()
```

```r
for (i in 1:1000){
  x <- rnorm(Nvec)
  X <- cbind(X, x)
}
ang3 <- all_angles(X, 1000)

ggplot() +
  geom_density(aes(x = ang1, fill = "red"), alpha = .4) +
  geom_density(aes(x = ang2, fill = "blue"), alpha = .4) +
  geom_density(aes(x = ang3, fill = "green"), alpha = .4) +
  scale_fill_discrete(labels = c("n=10", "n=100", "n=1000"))
```



Write an R function `nth_fibonnaci` that finds the nth Fibonnaci number via recursion but allows you to specify the starting number. For instance, if the sequence started at 1, you get the familiar 1, 1, 2, 3, 5, etc. But if it started at 0.01, you would get 0.01, 0.01, 0.02, 0.03, 0.05, etc.

```r
nth_fibonacci <- function(n, start){
  if (n == 1 | n == 2) return(start)
  else return(nth_fibonacci(n-1, start) + nth_fibonacci(n-2, start))
}
```

Write an Rcpp function `nth_fibonnaci_cpp` that does the same thing. Use an IDE if ou want, but write it below in-line.

```
cppFunction(
  'double nth_fibonacci_cpp(int n, double start){
    if (n == 1 || n == 2) return start;
    else return (nth_fibonacci_cpp(n-1, start) + nth_fibonacci_cpp(n-2, start));
  }'
)
```

Time the difference in these functions for n = 100, 200, ..., 1500 while starting the sequence at the smallest possible floating point value in R. Store the results in a matrix.

```
n <- c(1:15) * 100
timer <- c()
timecpp <- c()

# this will make r blow up
for (i in 1:length(n)){
  timer <- c(timer, unname(system.time(
    {fibr = nth_fibonacci(n[i], .Machine$double.xmin)}
  )[3]))
  timecpp <- c(timecpp, unname(system.time(
    {fibcpp = nth_fibonacci_cpp(n[i], .Machine$double.xmin)}
  )[3]))
}
```

Plot the divergence of performance (in log seconds) over n using a line geometry. Use two different colors for the R and CPP functions. Make sure there's a color legend on your plot.

```
ggplot() +
  geom_line(aes(y = n, x = log(timer), col = "timer")) +
  geom_line(aes(y = n, x = log(timecpp), col = "timecpp"))
```