Assignment 3: Predicting insurance charges by age and BMI

Your name and student ID

January 17, 2022

```
BEGIN ASSIGNMENT
requirements: requirements.R
generate: true
files:
 - data
```

Run this chunk of code to load the autograder package!

Instructions

- Solutions will be released Tuesday, February 8th.
- This semester, homework assignments are for practice only and will not be turned in for marks.

Helpful hints:

- Every function you need to use was taught during lecture! So you may need to revisit the lecture code to help you along by opening the relevant files on Datahub. Alternatively, you may wish to view the code in the condensed PDFs posted on the course website. Good luck!
- Knit your file early and often to minimize knitting errors! If you copy and paste code for the slides, you are bound to get an error that is hard to diagnose. Typing out the code is the way to smooth knitting! We recommend knitting your file each time after you write a few sentences/add a new code chunk, so you can detect the source of knitting errors more easily. This will save you and the GSIs from frustration! **You must knit correctly before submitting.**
- It is good practice to not allow your code to run off the page. To avoid this, have a look at your knitted PDF and ensure all the code fits in the file. If it doesn't look right, go back to your .Rmd file and add spaces (new lines) using the return or enter key so that the code runs onto the next line.

```r
library(readr)
library(dplyr)
library(ggplot2)
library(broom)
library(forcats)
```

Predicting insurance charges by age and BMI

Problem: Medical insurance charges can vary according to the complexity of a procedure or condition that requires medical treatment. You are tasked with determining how these charges are associated with age, for patients who have a body mass index (bmi) in the "normal" range (bmi between 16 and 25) who are smokers.

Plan: You have chosen to use tools to examine relationships between two variables to address the problem. In particular, scatter plots and simple linear regression.

Data: You have access to the dataset `insurance.csv`, a claims dataset from an insurance provider.

Analysis and Conclusion: In this assignment you will perform the analysis and make a conclusion to help answer the problem statement.

1. [1 point] Type one line of code to import these data into R. Assign the data to `insure_data`. Execute the code by hitting the green arrow and ensure the dataset has been saved by looking at the environment tab and viewing the data set by clicking the table icon to the right of its name.

```
BEGIN QUESTION
name: p1
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
insure_data <- NULL
insure_data
" # END PROMPT

# BEGIN SOLUTION
insure_data <- read_csv("data/insurance.csv")
```

```
## Parsed with column specification:
## cols(
##   age = col_double(),
##   sex = col_character(),
##   bmi = col_double(),
##   children = col_double(),
##   smoker = col_character(),
##   region = col_character(),
##   charges = col_double()
## )
```

```
# END SOLUTION
```

```
## Test ##
test_that("p1a", {
  expect_true(is.data.frame(insure_data))
  print("Checking: loaded the data and saved as `insure_data`")
})
```

```
## [1] "Checking: loaded the data and saved as `insure_data`"
## Test passed
```

```
## Test ##
test_that("p1b", {
  expect_true(typeof(insure_data$sex) == "character")
  print("Checking: form of the data-reading function used: _ or .")
})
```

```
## [1] "Checking: form of the data-reading function used: _ or ."
## Test passed
```

Execute the functions below one line at a time to get to know your dataset.

```
dim(insure_data)
```

```
## [1] 1338    7
```

```
names(insure_data)
```

```
## [1] "age"      "sex"      "bmi"      "children" "smoker"   "region"   "charges"
```

```
str(insure_data)
```

```
## spec_tbl_df [1,338 x 7] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ age     : num [1:1338] 19 18 28 33 32 31 46 37 37 60 ...
##  $ sex     : chr [1:1338] "female" "male" "male" "male" ...
##  $ bmi     : num [1:1338] 27.9 33.8 33 22.7 28.9 ...
##  $ children: num [1:1338] 0 1 3 0 0 0 1 3 2 0 ...
##  $ smoker  : chr [1:1338] "yes" "no" "no" "no" ...
##  $ region  : chr [1:1338] "southwest" "southeast" "southeast" "northwest" ...
##  $ charges : num [1:1338] 16885 1726 4449 21984 3867 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   age = col_double(),
##   ..   sex = col_character(),
##   ..   bmi = col_double(),
##   ..   children = col_double(),
##   ..   smoker = col_character(),
##   ..   region = col_character(),
##   ..   charges = col_double()
##   .. )
```

```
head(insure_data)
```

```
## # A tibble: 6 x 7
##     age sex      bmi children smoker region    charges
##   <dbl> <chr>  <dbl>    <dbl> <chr>  <chr>       <dbl>
## 1    19 female  27.9        0 yes    southwest  16885.
## 2    18 male    33.8        1 no     southeast   1726.
## 3    28 male    33          3 no     southeast   4449.
## 4    33 male    22.7        0 no     northwest  21984.
## 5    32 male    28.9        0 no     northwest   3867.
## 6    31 female  25.7        0 no     southeast   3757.
```

2. [1 point] How many individuals are in the dataset? Assign this number to p2.

```
BEGIN QUESTION
name: p2
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p2 <- NULL
p2
" # END PROMPT

# BEGIN SOLUTION
p2 <- nrow(insure_data)
# END SOLUTION
```

```
## Test ##
test_that("p2a", {
  expect_true(is.numeric(p2))
  print("Checking: p2 is a number")
})
```

```
## [1] "Checking: p2 is a number"
## Test passed
```

```
## Test ##
test_that("p2b", {
  expect_true(all.equal(p2, 1338, tol = 0.01))
  print("Checking: correct number of individuals")
})
```

```
## [1] "Checking: correct number of individuals"
## Test passed
```

3. [1 point] What are the nominal variables in the dataset? Assign the names of these variables to a vector of strings, p3.

```
BEGIN QUESTION
name: p3
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p3 <- NULL
p3
" # END PROMPT

# BEGIN SOLUTION
p3 <- c("sex", "smoker", "region")
# END SOLUTION
```

```
## Test ##
test_that("p3a", {
  expect_true(is.vector(p3))
  print("Checking: p3 is a vector")
})
```

```
## [1] "Checking: p3 is a vector"
## Test passed
```

```
## Test ##
test_that("p3b", {
  expect_true(typeof(p3) == "character")
  print("Checking: p3 is a character vector")
})
```

```
## [1] "Checking: p3 is a character vector"
## Test passed
```

```
## Test ##
test_that("p3c", {
  expect_true('sex' %in% p3 && 'smoker' %in% p3 && 'region' %in% p3)
  print("Checking: variables in p3")
})
```

```
## [1] "Checking: variables in p3"
## Test passed
```

4. [1 point] How many ordinal variables are in the dataset? Assign the *number* of ordinal variables to p4.

```
BEGIN QUESTION
name: p4
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p4 <- NULL
p4
" # END PROMPT

# BEGIN SOLUTION
p4 <- 0
# END SOLUTION
```

```
## Test ##
test_that("p4a", {
  expect_true(is.numeric(p4))
  print("Checking: p4 is a number")
})
```

```
## [1] "Checking: p4 is a number"
## Test passed
```

```
## Test ##
test_that("p4b", {
  expect_true(p4 == 0)
  print("Checking: correct value of p4")
})
```

```
## [1] "Checking: correct value of p4"
## Test passed
```

5. [1 point] Are there continuous variables in the dataset? Assign the names of these variables to a vector of strings, p5.

```
BEGIN QUESTION
name: p5
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p5 <- NULL
p5
" # END PROMPT

# BEGIN SOLUTION
p5 <- c("bmi", "charges", "age")
p5 <- c("bmi", "charges") # also accepted
# END SOLUTION
```

```
## Test ##
test_that("p5a", {
  expect_true(is.vector(p5))
  print("Checking: p5 is a vector")
})
```

```
## [1] "Checking: p5 is a vector"
## Test passed
```

```
## Test ##
test_that("p5b", {
  expect_true(typeof(p5) == "character")
  print("Checking: p5 is a character vector")
})
```

```
## [1] "Checking: p5 is a character vector"
## Test passed
```

```
## Test ##
test_that("p5c", {
  expect_true('bmi' %in% p5 && 'charges' %in% p5)
  print("Checking: variables in p5")
})
```

```
## [1] "Checking: variables in p5"
## Test passed
```

6. [1 point] What are the discrete variables in the dataset? Assign the names of these variables to a vector of strings, p6.

```
BEGIN QUESTION
name: p6
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p6 <- NULL
p6
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p6 <- c("children")
p6 <- c("children", "age") # also accepted
# END SOLUTION
```

```
## Test ##
test_that("p6a", {
  expect_true(is.vector(p6))
  print("Checking: p6 is a vector")
})
```

```
## [1] "Checking: p6 is a vector"
## Test passed
```

```
## Test ##
test_that("p6b", {
  expect_true(typeof(p6) == "character")
  print("Checking: p6 is a character vector")
})
```

```
## [1] "Checking: p6 is a character vector"
## Test passed
```

```
## Test ##
test_that("p6c", {
  expect_true('children' %in% p6)
  print("Checking: variables in p6")
})
```

```
## [1] "Checking: variables in p6"
## Test passed
```

Run the following code. Remind yourself what the `mutate()` function does in general, and notice that a new function called `case_when()` is also being used.

```
insure_data <- insure_data %>%
  mutate(bmi_cat = case_when(bmi < 16 ~ "Underweight",
                             bmi >= 16 & bmi < 25 ~ "Normal",
                             bmi >= 25 & bmi < 30 ~ "Overweight",
                             bmi >= 30 ~ "Obese")
                 )
```

7. What did the code above accomplish?

```
BEGIN QUESTION
name: p7
manual: true
```

The above code created a new variable called `bmi_cat` that created four categories of BMI: underweight, normal, overweight, and obese, based on the continuous variable BMI.

8. [1 point] What type of variable is `bmi_cat`? Uncomment one of the choices below.

```
BEGIN QUESTION
name: p8
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
# p8 <- 'ordinal'
# p8 <- 'nominal'
# p8 <- 'continuous'
# p8 <- 'discrete'
" # END PROMPT

# BEGIN SOLUTION
p8 <- 'ordinal'
# END SOLUTION
```

```
## Test ##
test_that("p8a", {
  expect_true(typeof(p8) == "character")
  print("Checking: a choice was made")
})
```

```
## [1] "Checking: a choice was made"
## Test passed
```

```
## Test ##
test_that("p8b", {
  expect_true(p8 == 'ordinal')
  print("Checking: correct choice was made")
})
```

```
## [1] "Checking: correct choice was made"
## Test passed
```

9. [1 point] Read the problem statement proposed at the beginning of this exercise. Who belongs to the population of interest? Uncomment one of the choices below.

```
BEGIN QUESTION
name: p9
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
# p9 <- 'Smokers of normal BMI'
# p9 <- 'Smokers of overweight BMI'
# p9 <- 'Smokers who have abnormal BMI'
# p9 <- 'All people at risk of high medical charges'
" # END PROMPT

# BEGIN SOLUTION
p9 <- 'Smokers of normal BMI'
# END SOLUTION
```

```
## Test ##
test_that("p9a", {
  expect_true(typeof(p9) == "character")
  print("Checking: a choice was made")
})
```

```
## [1] "Checking: a choice was made"
## Test passed
```

```
## Test ##
test_that("p9b", {
  expect_true(p9 == "Smokers of normal BMI")
  print("Checking: correct choice was made")
})
```

```
## [1] "Checking: correct choice was made"
## Test passed
```

10. [1 point] Using a `dplyr` function, make a new dataset called `insure_subset` containing the population of interest.

```
BEGIN QUESTION
name: p10
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
insure_subset <- NULL
insure_subset
" # END PROMPT

# BEGIN SOLUTION
insure_subset <- insure_data %>% filter(smoker == "yes" & bmi_cat == "Normal")
# END SOLUTION
```

```r
## Test ##
test_that("p10a", {
  expect_true(is.data.frame(insure_subset))
  print("Checking: insure_subset is a dataframe")
})
```

```
## [1] "Checking: insure_subset is a dataframe"
## Test passed
```

```r
## Test ##
test_that("p10b", {
  expect_true(nrow(insure_subset) == 55)
  print("Checking: both filters were applied")
})
```

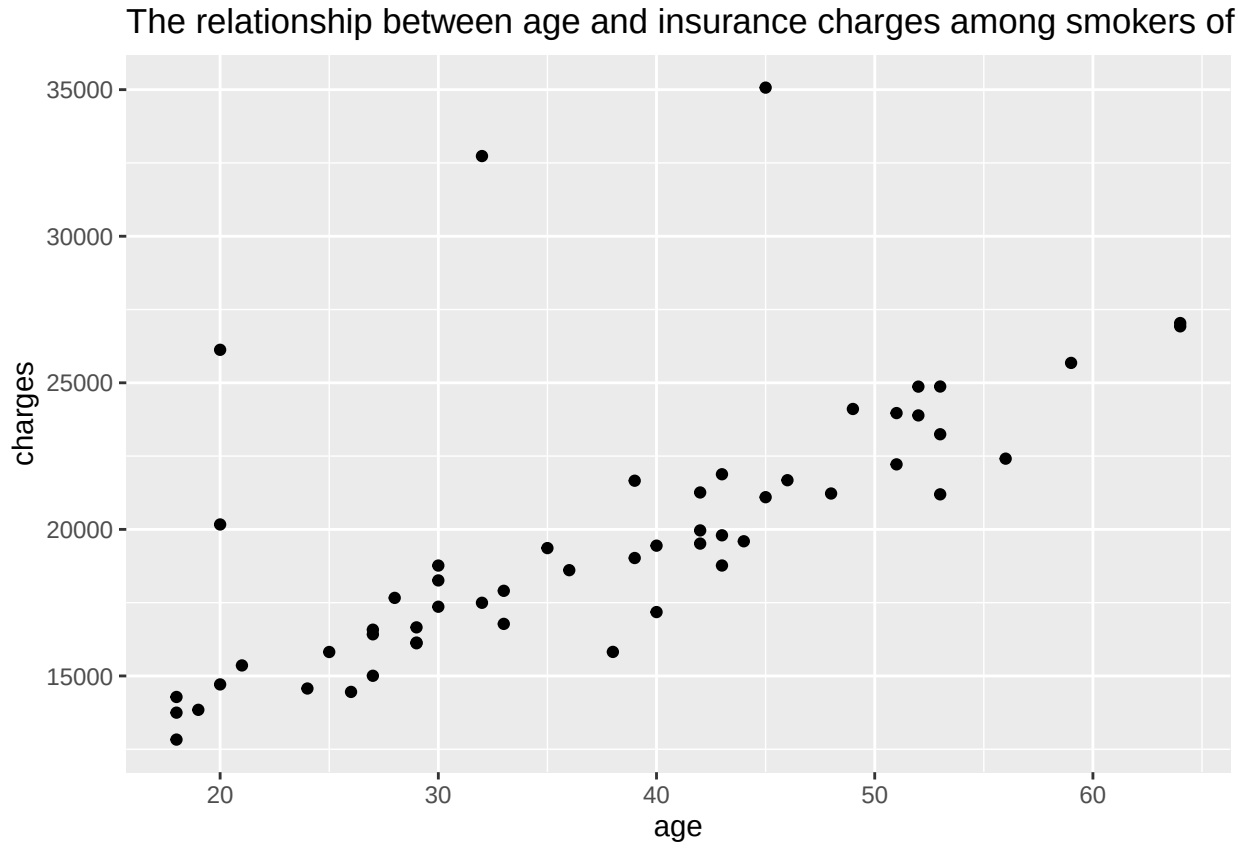
```
## [1] "Checking: both filters were applied"
## Test passed
```

11. [3 points] Make a scatter plot of the relationship between age and insurance charges for the population of interest. Give your plot an informative title.

```
BEGIN QUESTION
name: p11
manual: false
points: 3
```

```
. = " # BEGIN PROMPT
p11 <- NULL
p11
" # END PROMPT

# BEGIN SOLUTION
p11 <- ggplot(insure_subset, aes(x = age, y = charges)) +
  geom_point() +
  labs(title = "The relationship between age and insurance charges among smokers of normal BMI")
p11
```




```
# END SOLUTION
```

```
## Test ##
test_that("p11a", {
  expect_true("ggplot" %in% class(p11))
  print("Checking: p11 is a ggplot")
})
```

```
## [1] "Checking: p11 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p11b", {
  expect_true(identical(p11$data, insure_subset))
  print("Checking: Using insure_subset")
})
```

```
## [1] "Checking: Using insure_subset"
## Test passed
```

```
## Test ##
test_that("p11c", {
  expect_true(rlang::quo_get_expr(p11$mapping$x) == "age")
  print("Checking: age is on the x-axis")
})
```

```
## [1] "Checking: age is on the x-axis"
## Test passed
```

```
## Test ##
test_that("p11d", {
  expect_true(rlang::quo_get_expr(p11$mapping$y) == "charges")
  print("Checking: charges is on the y-axis")
})
```

```
## [1] "Checking: charges is on the y-axis"
## Test passed
```

```
## Test ##
test_that("p11e", {
  expect_true("GeomPoint" %in% class(p11$layers[[1]]$geom))
  print("Checking: made a scatterplot")
})
```

```
## [1] "Checking: made a scatterplot"
## Test passed
```

```
## Test ##
test_that("p11f", {
  expect_true(length(p11$labels$title) != 0)
  print("Checking: title added")
})
```

```
## [1] "Checking: title added"
## Test passed
```

12. [2 points] Run a linear regression model on the relationship between `age` and `charges`. Think about which variable is explanatory (X) and which is response (Y). Assign the regression model to the object `insure_mod` and uncomment the line of code below the model to tidy the output.

```
BEGIN QUESTION
name: p12
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
insure_model <- NULL
insure_model
# tidy(insure_model)
" # END PROMPT

# BEGIN SOLUTION
insure_model <- lm(formula = charges ~ age, data = insure_subset)
tidy(insure_model)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic       p.value
##   <chr>          <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   10656.    1471.       7.24 0.00000000184
## 2 age             246.      37.4      6.58 0.0000000217
```

```
# END SOLUTION
```

```
## Test ##
test_that("p12a", {
  expect_true("insure_subset" == insure_model$call$data)
  print("Checking: using insure_subset")
})
```

```
## [1] "Checking: using insure_subset"
## Test passed
```

```
## Test ##
test_that("p12b", {
  expect_true("age" %in% names(insure_model$model))
  print("Checking: age is in the model")
})
```

```
## [1] "Checking: age is in the model"
## Test passed
```

```
## Test ##
test_that("p12c", {
  expect_true("charges" %in% names(insure_model$model))
  print("Checking: charges is in the model")
})
```

```
## [1] "Checking: charges is in the model"
## Test passed
```

```
## Test ##
test_that("p12d", {
  expect_true(all.equal(insure_model$coefficients[[2]], 246.1367, tol = 0.01))
  print("Checking: slope value")
})
```

```
## [1] "Checking: slope value"
## Test passed
```

13. [1 point] Interpret the slope parameter in the context of this problem.

```
BEGIN QUESTION
name: p13
manual: true
```

For every year increase in age, medical charges increase by $246.14.

14. [1 point] Interpret the intercept parameter.

```
BEGIN QUESTION
name: p14
manual: true
```

The model predicts that the insurance charged would be $10,656.14 for a person of aged 0.

15. [1 point] Does the intercept make sense in this context?

```
BEGIN QUESTION
name: p15
manual: true
```

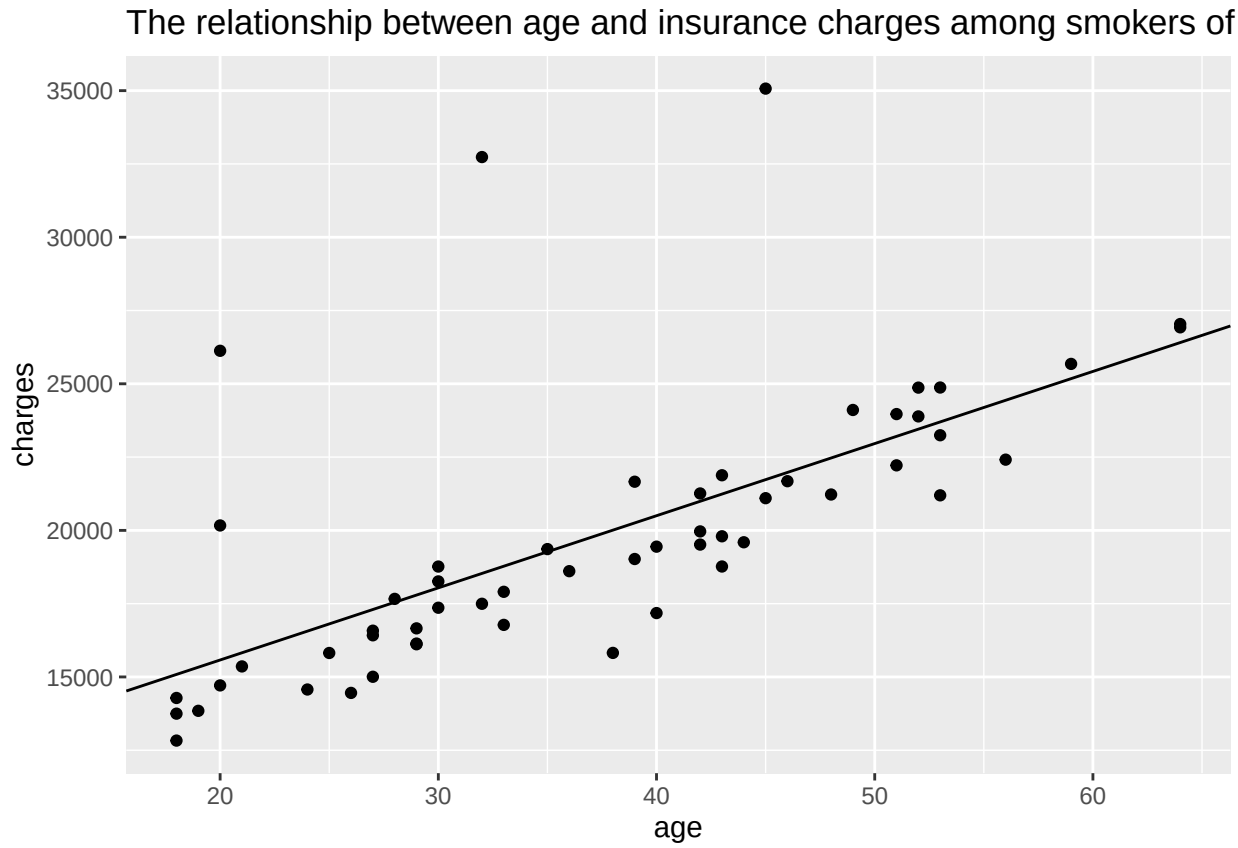
No because being 0 years old is non sensical. Further, the minimum age in the dataset is 18, so extrapolation to 0 is not supported by the data. (student can say either of these items or both.)

16. [1 point] Add the line of best fit to your scatter plot by copying and pasting the plot's code from question 11 in the chunk below and adding a `geom` that can be used to add a regression line.

```
BEGIN QUESTION
name: p16
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p16 <- NULL
p16
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p16 <- ggplot(insure_subset, aes(x = age, y = charges)) +
  geom_point() +
  labs(title = "The relationship between age and insurance charges among smokers of normal BMI") +
  geom_abline(intercept = 10656.1, slope = 246.1)
p16
```




```
# END SOLUTION
```

```
## Test ##
test_that("p16a", {
  expect_true("ggplot" %in% class(p16))
  print("Checking: p16 is a ggplot")
})
```

```
## [1] "Checking: p16 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p16b", {
  expect_true(identical(p16$data, insure_subset))
  print("Checking: using insure_subset")
})
```

```
## [1] "Checking: using insure_subset"
## Test passed
```

```
## Test ##
test_that("p16c", {
  expect_true(rlang::quo_get_expr(p16$mapping$x) == "age")
  print("Checking: age is on the x-axis")
})
```

```
## [1] "Checking: age is on the x-axis"
## Test passed
```

```
## Test ##
test_that("p16d", {
  expect_true(rlang::quo_get_expr(p16$mapping$y) == "charges")
  print("Checking: charges is on the y-axis")
})
```

```
## [1] "Checking: charges is on the y-axis"
## Test passed
```

```
## Test ##
test_that("p16e", {
  expect_true("GeomPoint" %in% class(p16$layers[[1]]$geom))
  print("Checking: made a scatterplot")
})
```

```
## [1] "Checking: made a scatterplot"
## Test passed
```

```
## Test ##
test_that("p16f", {
  expect_true(length(p16$labels$title) != 0)
  print("Checking: title added")
})
```

```
## [1] "Checking: title added"
## Test passed
```

```
## Test ##
test_that("p16g", {
  expect_true("GeomAbline" %in% class(p16$layers[[2]]$geom) | "GeomSmooth" %in% class(p16$layers[[2]]$g
  print("Checking: added a line of best fit")
})
```

```
## [1] "Checking: added a line of best fit"
## Test passed
```

17. [2 points] What do you notice about the fit of the line in terms of the proportion of points above vs. below the line? Why do you think that is?

```
BEGIN QUESTION
name: p17
manual: false
points: 2
```

The line seems high. There is a large proportion of points below the line. That's because there exists some notable outliers above the line which don't follow the linear trend of the data points.

Run the following `filter()` function in the chunk below.

```
insure_smaller_subset <- insure_subset %>%
  filter(charges < 30000 & ! (charges > 25000 & age == 20))
```

18. [2 points] How many individuals were removed? Who were they?

```
BEGIN QUESTION
name: p18
manual: true
```

Three individuals were removed. They were the "y outliers", the two people with the highest charges in the dataset and a third person who was 20 years old with a charge > $25,000.

19. [2 points] Run a regression model on `insure_smaller_subset` between `charges` and `age`. Assign the model to `insure_better_model` and analyze the output using the `tidy()` function.

```
BEGIN QUESTION
name: p19
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
insure_better_model <- NULL
# tidy(insure_better_model)
" # END PROMPT

# BEGIN SOLUTION
insure_better_model <- lm(formula = charges ~ age, data = insure_smaller_subset)
tidy(insure_better_model)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    9144.     633.       14.4 1.81e-19
## 2 age             267.      16.0      16.7 4.44e-22
```

```
# END SOLUTION
```

```
## Test ##
test_that("p19a", {
  expect_true("insure_smaller_subset" == insure_better_model$call$data)
  print("Checking: using insure_smaller_subset")
})
```

```
## [1] "Checking: using insure_smaller_subset"
## Test passed
```

```
## Test ##
test_that("p19b", {
  expect_true("age" %in% names(insure_better_model$model))
  print("Checking: age is in the model")
})
```

```
## [1] "Checking: age is in the model"
## Test passed
```

```
## Test ##
test_that("p19c", {
  expect_true("charges" %in% names(insure_better_model$model))
  print("Checking: charges is in the model")
})
```

```
## [1] "Checking: charges is in the model"
## Test passed
```

```r
## Test ##
test_that("p19d", {
  expect_true(all.equal(insure_better_model$coefficients[[2]], 266.8725, tol = 0.01))
  print("Checking: slope value")
})
```

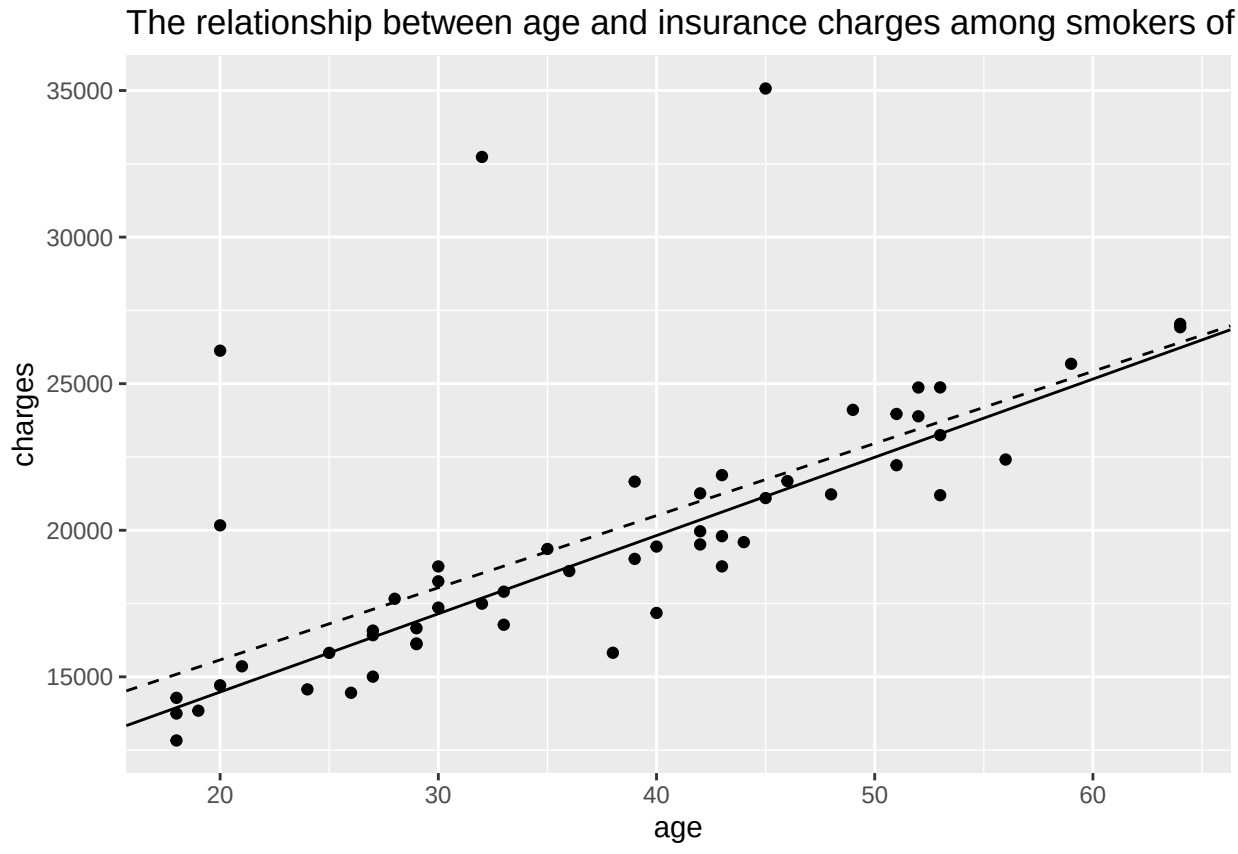
```
## [1] "Checking: slope value"
## Test passed
```

20. [2 points] Add the new regression line to your ggplot from question 16. Keep the original regression line on the plot for comparison. To distinguish the lines, change the color, line type, or line width of one of the lines.

```
BEGIN QUESTION
name: p20
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p20 <- NULL # YOUR CODE HERE
p20
" # END PROMPT

# BEGIN SOLUTION
p20 <- ggplot(insure_subset, aes(x = age, y = charges)) +
  geom_point() +
  labs(title = "The relationship between age and insurance charges among smokers of normal BMI") +
  geom_abline(intercept = 10656.1, slope = 246.1, lty = 2) +
  geom_abline(intercept = 9144.1, slope = 266.9)
p20
```




```
# END SOLUTION
```

```
## Test ##
test_that("p20a", {
  expect_true("ggplot" %in% class(p20))
  print("Checking: p20 is a ggplot")
})
```

```
## [1] "Checking: p20 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p20b", {
  expect_true(identical(p20$data, insure_subset))
  print("Checking: using insure_subset")
})
```

```
## [1] "Checking: using insure_subset"
## Test passed
```

```
## Test ##
test_that("p20c", {
  expect_true(rlang::quo_get_expr(p20$mapping$x) == "age")
  print("Checking: age is on the x-axis")
})
```

```
## [1] "Checking: age is on the x-axis"
## Test passed
```

```
## Test ##
test_that("p20d", {
  expect_true(rlang::quo_get_expr(p20$mapping$y) == "charges")
  print("Checking: charges is on the y-axis")
})
```

```
## [1] "Checking: charges is on the y-axis"
## Test passed
```

```
## Test ##
test_that("p20e", {
  expect_true("GeomPoint" %in% class(p20$layers[[1]]$geom))
  print("Checking: made a scatterplot")
})
```

```
## [1] "Checking: made a scatterplot"
## Test passed
```

```
## Test ##
test_that("p20f", {
  expect_true(length(p20$labels$title) != 0)
  print("Checking: title added")
})
```

```
## [1] "Checking: title added"
## Test passed
```

```
## Test ##
test_that("p20g", {
  expect_true("GeomAbline" %in% class(p20$layers[[2]]$geom) | "GeomSmooth" %in% class(p20$layers[[2]]$g
  print("Checking: added a line of best fit")
})
```

```
## [1] "Checking: added a line of best fit"
## Test passed
```

```
## Test ##
test_that("p20h", {
  expect_true("GeomAbline" %in% class(p20$layers[[3]]$geom) | "GeomSmooth" %in% class(p20$layers[[3]]$g
  print("Checking: added a second line of best fit")
})
```

```
## [1] "Checking: added a second line of best fit"
## Test passed
```

21. [1 point] Calculate the r-squared value for `insure_model` and assign this value to `insure_model_r2`.

```
BEGIN QUESTION
name: p21
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
insure_model_r2 <- NULL
insure_model_r2
" # END PROMPT

# BEGIN SOLUTION
insure_model_r2 <- glance(insure_model) %>% pull(r.squared)
# END SOLUTION
```

```
## Test ##
test_that("p21a", {
  expect_true(is.numeric(insure_model_r2))
  print("Checking: insure_model_r2 is a number")
})
```

```
## [1] "Checking: insure_model_r2 is a number"
## Test passed
```

```
## Test ##
test_that("p21b", {
  expect_true(all.equal(insure_model_r2, 0.449261, tol = 0.01))
  print("Checking: correct value of insure_model_r2")
})
```

```
## [1] "Checking: correct value of insure_model_r2"
## Test passed
```

22. [1 point] Calculate the r-squared value for `insure_better_model` using a function learned in class. Assign this value to `insure_better_model_r2`.

```
BEGIN QUESTION
name: p22
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
insure_better_model_r2 <- NULL
insure_better_model_r2
" # END PROMPT

# BEGIN SOLUTION
insure_better_model_r2 <- glance(insure_better_model) %>% pull(r.squared)
# END SOLUTION
```

```r
## Test ##
test_that("p22a", {
  expect_true(is.numeric(insure_better_model_r2))
  print("Checking: insure_better_model_r2 is a number")
})
```

```
## [1] "Checking: insure_better_model_r2 is a number"
## Test passed
```

```r
## Test ##
test_that("p22b", {
  expect_true(all.equal(insure_better_model_r2, 0.8477642, tol = 0.01))
  print("Checking: correct value of insure_better_model_r2")
})
```

```
## [1] "Checking: correct value of insure_better_model_r2"
## Test passed
```

23. [2 points] Calculate the correlation coefficient between age and charges using `insure_subset`. Also calculate the squared correlation coefficient. You should use `summarize()` to create a dataframe of these two values and name the two variables `corr` and `corr_sq`, respectively. What do you notice about the relationship between the correlation coefficient and r-squared values that you calculated earlier?

```
BEGIN QUESTION
name: p23
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p23 <- NULL
p23
" # END PROMPT

# BEGIN SOLUTION
p23 <- insure_subset %>% summarize(corr = cor(age, charges), corr_sq = corr^2)
p23
```

```
## # A tibble: 1 x 2
##    corr corr_sq
##   <dbl>   <dbl>
## 1 0.670   0.449
```

```
# END SOLUTION
```

```
## Test ##
test_that("p23a", {
  expect_true(is.data.frame(p23))
  print("Checking: p23 is a dataframe")
})
```

```
## [1] "Checking: p23 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p23b", {
  expect_true(nrow(p23) == 1)
  print("Checking: one row in the dataframe")
})
```

```
## [1] "Checking: one row in the dataframe"
## Test passed
```

```
## Test ##
test_that("p23c", {
  expect_true(ncol(p23) == 2)
  print("Checking: two columns in the dataframe")
})
```

```
## [1] "Checking: two columns in the dataframe"
## Test passed
```

```
## Test ##
test_that("p23d", {
  expect_true(all.equal(p23$corr[1], 0.6702694, tol = 0.01))
  print("Checking: correct correlation coefficient")
})
```

```
## [1] "Checking: correct correlation coefficient"
## Test passed
```

```
## Test ##
test_that("p23e", {
  expect_true(all.equal(p23$corr_sq[1], 0.4492611, tol = 0.01))
  print("Checking: correct correlation squared value")
})
```

```
## [1] "Checking: correct correlation squared value"
## Test passed
```

24. [2 points] Calculate the correlation coefficient between age and charges using the smaller dataset `insure_smaller_subset`. Also calculate the squared correlation coefficient. You should use `summarize()` to create a dataframe of these two values and name the two variables `corr` and `corr_sq`, respectively. What do you notice about the relationship between the correlation coefficient and r-squared values that you calculated earlier?

```
BEGIN QUESTION
name: p24
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p24 <- NULL
p24
" # END PROMPT
# BEGIN SOLUTION NO PROMPT
p24 <- insure_smaller_subset %>% summarize(corr = cor(age, charges), corr_sq = corr^2)
# END SOLUTION
```

```
## Test ##
test_that("p24a", {
  expect_true(is.data.frame(p24))
  print("Checking: p24 is a dataframe")
})
```

```
## [1] "Checking: p24 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p24b", {
  expect_true(nrow(p24) == 1)
  print("Checking: one row in the dataframe")
})
```

```
## [1] "Checking: one row in the dataframe"
## Test passed
```

```
## Test ##
test_that("p24c", {
  expect_true(ncol(p24) == 2)
  print("Checking: two columns in the dataframe")
})
```

```
## [1] "Checking: two columns in the dataframe"
## Test passed
```

```
## Test ##
test_that("p24d", {
  expect_true(all.equal(p24$corr[1], 0.9207411, tol = 0.01))
  print("Checking: correct correlation coefficient")
})
```

```
## [1] "Checking: correct correlation coefficient"
## Test passed
```

```
## Test ##
test_that("p24e", {
  expect_true(all.equal(p24$corr_sq[1], 0.8477642, tol = 0.01))
  print("Checking: correct correlation squared value")
})
```

```
## [1] "Checking: correct correlation squared value"
## Test passed
```

Your supervisor asks you to extend your analysis to consider other smokers with BMIs classified as overweight or obese. In particular, she wanted to know if the relationship between age and medical charges is different for different BMI groups. You can use data visualization coupled with your skills in linear regression to help answer this question.

25. [1 point] Make a new dataframe called `insure_smokers` that includes smokers of any BMI from the original `insure_data` dataset.

```
BEGIN QUESTION
name: p25
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
insure_smokers <- NULL
insure_smokers
" # END PROMPT

# BEGIN SOLUTION
insure_smokers <- insure_data %>% filter(smoker == "yes")
# END SOLUTION
```

```
## Test ##
test_that("p25a", {
  expect_true(is.data.frame(insure_smokers))
  print("Checking: insure_smokers is a dataframe")
})
```

```
## [1] "Checking: insure_smokers is a dataframe"
## Test passed
```

```
## Test ##
test_that("p25b", {
  expect_true(nrow(insure_smokers) == 274)
  print("Checking: filtered correctly")
})
```

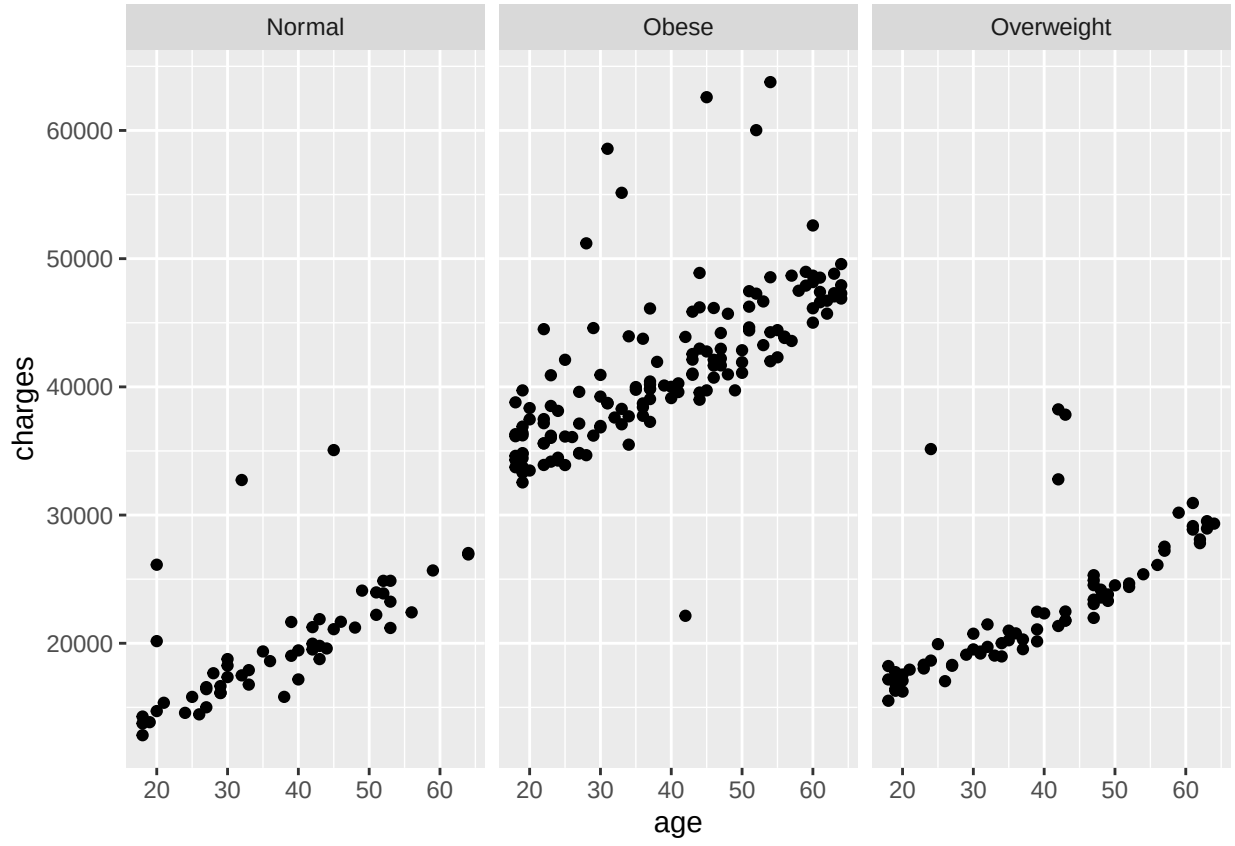
```
## [1] "Checking: filtered correctly"
## Test passed
```

26. [1 point] Make a scatterplot that examines the relationship between `age` and `charges` for normal, overweight, and obese individuals in three side by side plots. A `facet_` command may help you.

```
BEGIN QUESTION
name: p26
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p26 <- NULL
p26
" # END PROMPT

# BEGIN SOLUTION
p26 <- ggplot(insure_smokers, aes(x = age, y = charges)) +
  geom_point() +
  facet_wrap(~ bmi_cat)
p26
```




```
# END SOLUTION
```

```
## Test ##
test_that("p26a", {
  expect_true("ggplot" %in% class(p26))
```

```
  print("Checking: p26 is a ggplot")
})
```

```
## [1] "Checking: p26 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p26b", {
  expect_true(identical(p26$data, insure_smokers))
  print("Checking: using insure_smokers")
})
```

```
## [1] "Checking: using insure_smokers"
## Test passed
```

```
## Test ##
test_that("p26c", {
  expect_true(rlang::quo_get_expr(p26$mapping$x) == "age")
  print("Checking: age is on the x-axis")
})
```

```
## [1] "Checking: age is on the x-axis"
## Test passed
```

```
## Test ##
test_that("p26d", {
  expect_true(rlang::quo_get_expr(p26$mapping$y) == "charges")
  print("Checking: charges is on the y-axis")
})
```

```
## [1] "Checking: charges is on the y-axis"
## Test passed
```

```
## Test ##
test_that("p26e", {
  expect_true("GeomPoint" %in% class(p26$layers[[1]]$geom))
  print("Checking: made a scatterplot")
})
```

```
## [1] "Checking: made a scatterplot"
## Test passed
```

```
## Test ##
test_that("p26f", {
  expect_true("FacetWrap" %in% class(p26$facet))
  print("Checking: used a facet")
})
```

```
## [1] "Checking: used a facet"
## Test passed
```

The plot above automatically displays the BMI categories alphabetically. Run the chunk below to assign a different order to the values of `bmi_cat`.

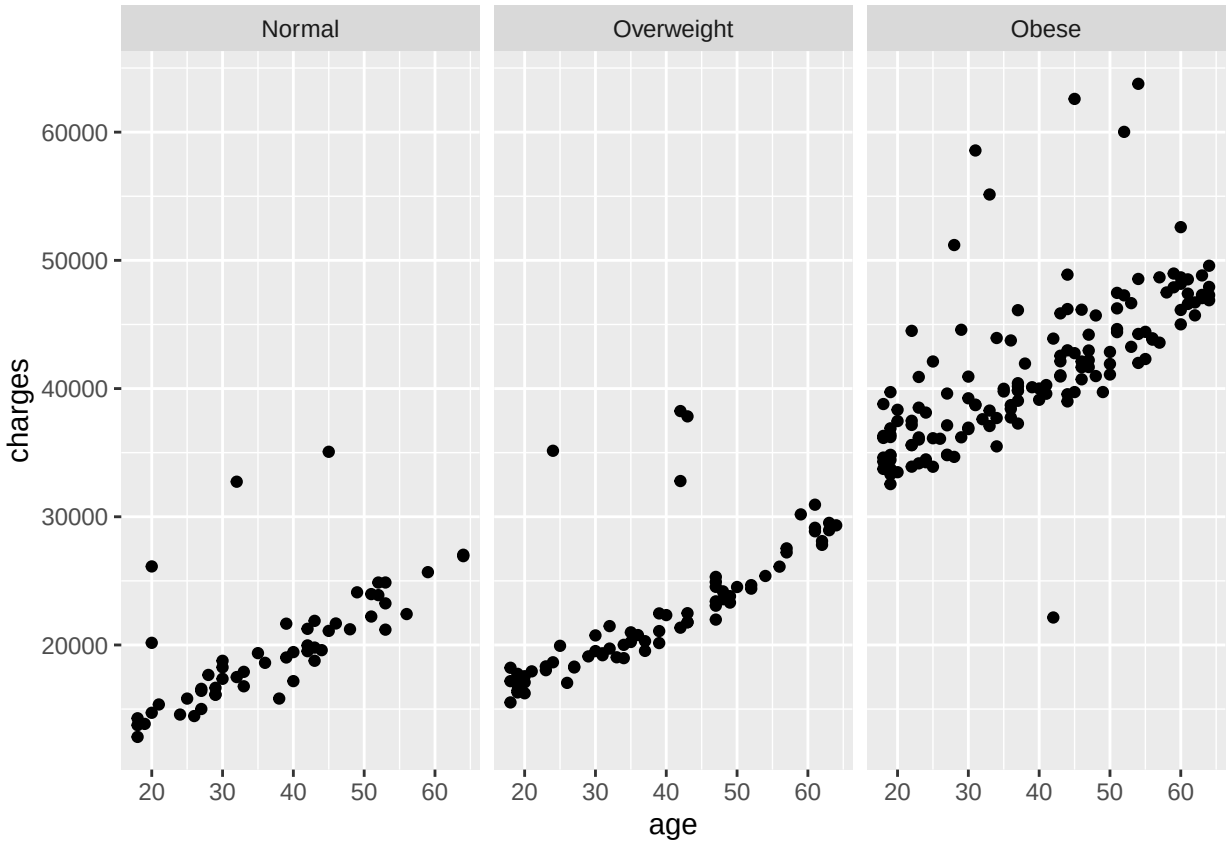
```
insure_smokers <- insure_smokers %>%
  mutate(bmi_cat_ordered = forcats::fct_relevel(bmi_cat, "Normal", "Overweight", "Obese"))
```

27. [1 point] Re-run your code from question 26, but facet using `bmi_cat_ordered`.

```
BEGIN QUESTION
name: p27
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p27 <- NULL
p27
" # END PROMPT

# BEGIN SOLUTION
p27 <- ggplot(insure_smokers, aes(x = age, y = charges)) +
  geom_point() +
  facet_wrap(~bmi_cat_ordered)
p27
```




```
# END SOLUTION
```

```
## Test ##
test_that("p27a", {
  expect_true("ggplot" %in% class(p27))
  print("Checking: p27 is a ggplot")
})
```

```
## [1] "Checking: p27 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p27b", {
  expect_true(identical(p27$data, insure_smokers))
  print("Checking: using insure_smokers")
})
```

```
## [1] "Checking: using insure_smokers"
## Test passed
```

```
## Test ##
test_that("p27c", {
  expect_true(rlang::quo_get_expr(p27$mapping$x) == "age")
  print("Checking: age is on the x-axis")
})
```

```
## [1] "Checking: age is on the x-axis"
## Test passed
```

```
## Test ##
test_that("p27d", {
  expect_true(rlang::quo_get_expr(p27$mapping$y) == "charges")
  print("Checking: charges is on the y-axis")
})
```

```
## [1] "Checking: charges is on the y-axis"
## Test passed
```

```
## Test ##
test_that("p27e", {
  expect_true("GeomPoint" %in% class(p27$layers[[1]]$geom))
  print("Checking: made a scatterplot")
})
```

```
## [1] "Checking: made a scatterplot"
## Test passed
```

```
## Test ##
test_that("p27f", {
  expect_true("FacetWrap" %in% class(p27$facet))
  print("Checking: used a facet")
})
```

```
## [1] "Checking: used a facet"
## Test passed
```

28. [3 points] Run a separate linear model for each BMI group. To do this, you will need to subset your data into the three groups of interest. Call your models `normal_mod`, `overweight_mod`, `obese_mod`. Use the `tidy()` function to display the output from each model.

```
BEGIN QUESTION
name: p28
manual: false
points: 3
```

```
. = " # BEGIN PROMPT

## subset your data here

# '<<<<YOUR CODE HERE>>>>'
# '<<<<YOUR CODE HERE>>>>'
# '<<<<YOUR CODE HERE>>>>'

## generate your models

normal_mod <- '<<<<YOUR CODE HERE>>>>'
overweight_mod <- '<<<<YOUR CODE HERE>>>>'
obese_mod <- '<<<<YOUR CODE HERE>>>>'

## tidy your models

# '<<<<YOUR CODE HERE>>>>'
# '<<<<YOUR CODE HERE>>>>'
# '<<<<YOUR CODE HERE>>>>'
" # END PROMPT

# BEGIN SOLUTION
insure_smokers_normal <- insure_smokers %>% filter(bmi_cat == "Normal")
insure_smokers_overweight <- insure_smokers %>% filter(bmi_cat == "Overweight")
insure_smokers_obese <- insure_smokers %>% filter(bmi_cat == "Obese")

normal_mod <- lm(charges ~ age, data = insure_smokers_normal)
overweight_mod <- lm(charges ~ age, data = insure_smokers_overweight)
obese_mod <- lm(charges ~ age, data = insure_smokers_obese)

tidy(normal_mod)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic       p.value
##   <chr>          <dbl>     <dbl>     <dbl>         <dbl>
## 1 (Intercept)   10656.    1471.       7.24 0.00000000184
## 2 age             246.      37.4      6.58 0.0000000217
```

```
tidy(overweight_mod)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   12400.    1176.      10.5  3.01e-16
## 2 age             264.      28.9      9.16 1.07e-13
```

```
tidy(obese_mod)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   30558.    1093.       28.0 7.96e-60
## 2 age             281.      26.2      10.7 5.05e-20
```

```
# END SOLUTION
```

```
## Test ##
test_that("p28a", {
  expect_true(class(normal_mod) == "lm")
  print("Checking: normal_mod is a linear model")
})
```

```
## [1] "Checking: normal_mod is a linear model"
## Test passed
```

```
## Test ##
test_that("p28b", {
  expect_true("age" %in% names(normal_mod$model))
  print("Checking: age is in the normal_mod")
})
```

```
## [1] "Checking: age is in the normal_mod"
## Test passed
```

```
## Test ##
test_that("p28c", {
  expect_true("charges" %in% names(normal_mod$model))
  print("Checking: charges is in the normal_mod")
})
```

```
## [1] "Checking: charges is in the normal_mod"
## Test passed
```

```
## Test ##
test_that("p28d", {
  expect_true(all.equal(normal_mod$coefficients[[2]], 246.1367, tol = 0.01))
  print("Checking: correct slope value in normal_mod")
})
```

```
## [1] "Checking: correct slope value in normal_mod"
## Test passed
```

```
## Test ##
test_that("p28e", {
  expect_true(class(overweight_mod) == "lm")
  print("Checking: overweight_mod is a linear model")
})
```

```
## [1] "Checking: overweight_mod is a linear model"
## Test passed
```

```
## Test ##
test_that("p28f", {
  expect_true("age" %in% names(overweight_mod$model))
  print("Checking: age is in the overweight_mod")
})
```

```
## [1] "Checking: age is in the overweight_mod"
## Test passed
```

```
## Test ##
test_that("p28g", {
  expect_true("charges" %in% names(overweight_mod$model))
  print("Checking: charges is in the overweight_mod")
})
```

```
## [1] "Checking: charges is in the overweight_mod"
## Test passed
```

```
## Test ##
test_that("p28h", {
  expect_true(all.equal(overweight_mod$coefficients[[2]], 264.1862, tol = 0.01))
  print("Checking: correct slope value in overweight_mod")
})
```

```
## [1] "Checking: correct slope value in overweight_mod"
## Test passed
```

```
## Test ##
test_that("p28i", {
  expect_true(class(obese_mod) == "lm")
  print("Checking: obese_mod is a linear model")
})
```

```
## [1] "Checking: obese_mod is a linear model"
## Test passed
```

```
## Test ##
test_that("p28j", {
  expect_true("age" %in% names(obese_mod$model))
  print("Checking: age is in the obese_mod")
})
```

```
## [1] "Checking: age is in the obese_mod"
## Test passed
```

```
## Test ##
test_that("p28k", {
  expect_true("charges" %in% names(obese_mod$model))
  print("Checking: charges is in the obese_mod")
})
```

```
## [1] "Checking: charges is in the obese_mod"
## Test passed
```

```
## Test ##
test_that("p28l", {
  expect_true(all.equal(obese_mod$coefficients[[2]], 281.1528, tol = 0.01))
  print("Checking: correct slope value in obese_mod")
})
```

```
## [1] "Checking: correct slope value in obese_mod"
## Test passed
```

For the next three problems, use the models to predict medical charges for a 20-year old by weight category. You don't need an R function to make these predictions, just the output from the models. Show your work for each calculation.

29. [1 point] Predict the medical charges for a 20 year old with a normal BMI.

```
BEGIN QUESTION
name: p29
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p29 <- NULL
p29
" # END PROMPT

# BEGIN SOLUTION
p29 <- 10656.1 + 246.1 * 20
# = $15578.1
# END SOLUTION
```

```
## Test ##
test_that("p29a", {
  expect_true(is.numeric(p29))
  print("Checking: p29 is a number")
})
```

```
## [1] "Checking: p29 is a number"
## Test passed
```

```
## Test ##
test_that("p29b", {
  expect_true(all.equal(p29,15578.1, tol = 0.1))
  print("Checking: correct value of p29")
})
```

```
## [1] "Checking: correct value of p29"
## Test passed
```

30. [1 point] Predict the medical charges for a 20 year old with an overweight BMI.

```
BEGIN QUESTION
name: p30
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p30 <- NULL
p30
" # END PROMPT

# BEGIN SOLUTION
p30 <- 12399.7 + 264.2 * 20
# = $17683.7
# END SOLUTION
```

```
## Test ##
test_that("p30a", {
  expect_true(is.numeric(p30))
  print("Checking: p30 is a number")
})
```

```
## [1] "Checking: p30 is a number"
## Test passed
```

```
## Test ##
test_that("p30b", {
  expect_true(all.equal(p30, 17683.7, tol = 0.1))
  print("Checking: correct value of p30")
})
```

```
## [1] "Checking: correct value of p30"
## Test passed
```

31. [1 point] Predict the medical charges for a 20 year old with an obese BMI.

```
BEGIN QUESTION
name: p31
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p31 <- NULL
p31
" # END PROMPT

# BEGIN SOLUTION
p31 <- 30558.1 + 281.2 * 20
# = $36182.1
# END SOLUTION
```

```
## Test ##
test_that("p31a", {
  expect_true(is.numeric(p31))
  print("Checking: p31 is a number")
})
```

```
## [1] "Checking: p31 is a number"
## Test passed
```

```
## Test ##
test_that("p31b", {
  expect_true(all.equal(p31, 36182.1, tol = 0.1))
  print("Checking: correct value of p31")
})
```

```
## [1] "Checking: correct value of p31"
## Test passed
```

32. [3 points] In three sentences maximum, comment on (1) the direction of the association, (2) how much the slopes vary across the BMI groups, and (3) how much the predicted medical charges for a 20-year old varies by BMI category.

```
BEGIN QUESTION
name: p32
manual: true
```

There was a positive association between age and medical charges for normal, overweight, and obese individuals. The relationship was of similar magnitude for each BMI group, though the slope increased in magnitude for overweight and obese individuals, implying that a steeper relationship for overweight individuals, and even steeper for obese individuals vs. normal BMI individuals. For a given age, obese individuals had much higher charges than overweight and normal weight individuals.

END