Assignment 3: Predicting insurance charges by age and BMI

Your name and student ID

February 09, 2021

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

Instructions

- Solutions will be released on Tuesday, February 9
- 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.

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

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] Please type one line of code below to import these data into R. Assign the data to `insure_data`. Execute the code by hitting the green arrow and ensure the data set 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
```

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

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   age = col_double(),
##   sex = col_character(),
##   bmi = col_double(),
##   children = col_double(),
##   smoker = col_character(),
##   region = col_character(),
##   charges = col_double()
## )
```

```
## Test ##
test_that("p1a", {
  expect_true(is.data.frame(insure_data))
  print("Checking: you've loaded the data into insure_data")
})
```

```
## [1] "Checking: you’ve loaded the data into insure_data"
## Test passed
```

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

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

Use the space below to use the functions from lecture to get to know your dataset. Execute these functions line by line so you can look at their output, and examine these data.

```
dim(insure_data)
```

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

```
names(insure_data)
```

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

```
str(insure_data)
```

```
## tibble [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
```

```r
p2 <- nrow(insure_data) # SOLUTION
```

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

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

```r
## Test ##
test_that("p2b", {
  expect_true(p2 == 1338)
  print("Checking: which form of the data-reading function was used ( _ or .)")
})
```

```
## [1] "Checking: which form of the data-reading function was used ( _ or .)"
## 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
```

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

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

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

```r
## 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
```

```r
## 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
```

```r
p4 <- 0 # SOLUTION
```

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

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

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

```
## [1] "Checking: 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
```

```r
. = " # BEGIN PROMPT
p5 <- NULL # YOUR CODE HERE
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p5 <- c("bmi", "charges", "age")
p5 <- c("bmi", "charges") # also accepted
# END SOLUTION
```

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

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

```r
## 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
```

```r
## 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
```

```r
. = " # BEGIN PROMPT
p6 <- NULL # YOUR CODE HERE
" # 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 by removing the eval = F from the chunk header and executing the code chunk. Remind yourself what the `mutate()` function does in general, and notice that a new function `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. [1 point] What did the above code achieve?:

```
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 NO PROMPT
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: which choice was made")
})
```

```
## [1] "Checking: which 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 NO PROMPT
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: which choice was made")
})
```

```
## [1] "Checking: which 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
```

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

```
## 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
```

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

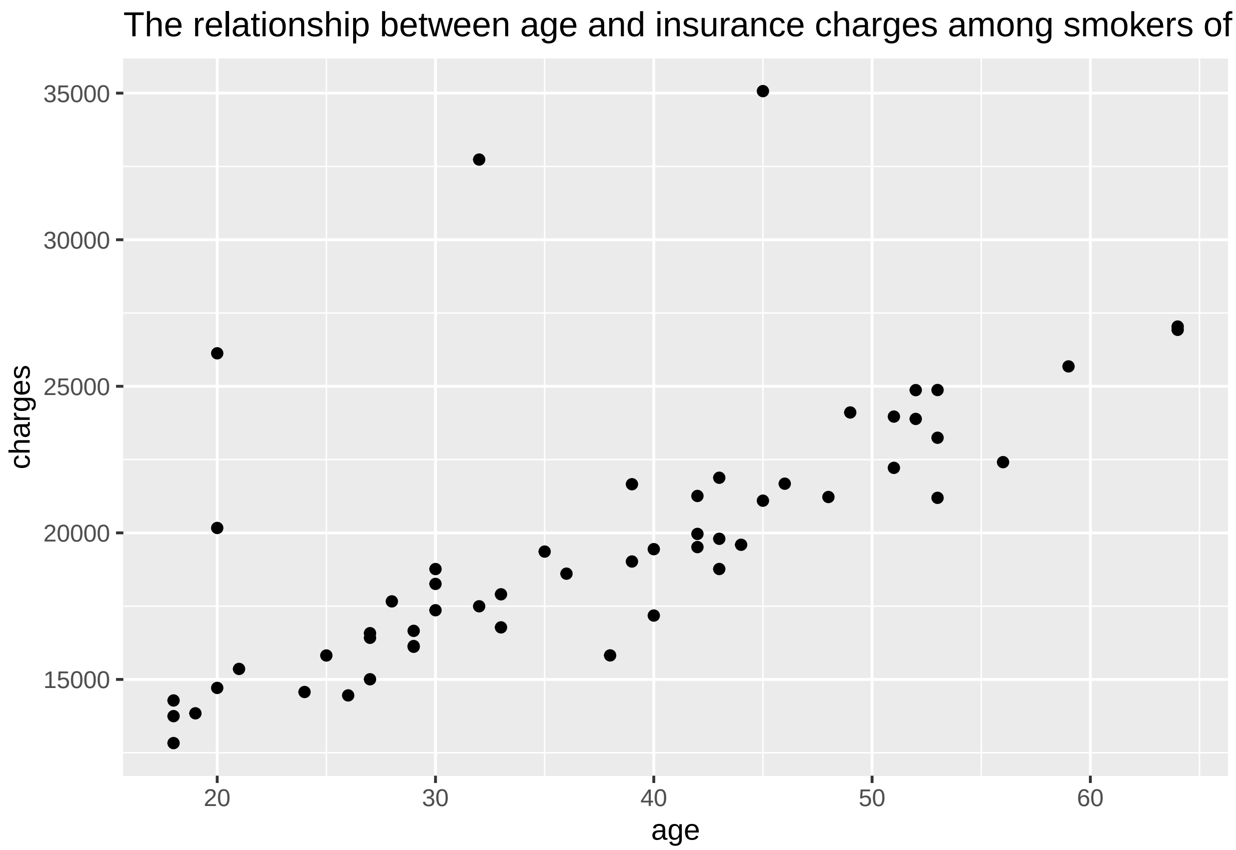
```
## [1] "Checking: if both filters were applied"
## Test passed
```

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 # YOUR CODE HERE
p11
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
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
```

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

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

```r
## 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
```

```r
## 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
```

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

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

```r
## 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 name `insure_mod`. Then type `tidy(insure_mod)` below the model's code and execute both lines.**

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

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

# BEGIN SOLUTION NO PROMPT
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: insure_subset is the dataset")
})
```

```
## [1] "Checking: insure_subset is the dataset"
## 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
```

13a. [1 point] Interpret the slope parameter:

```
BEGIN QUESTION
name: p13a
manual: true
```

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

13b. [1 point] Interpret the intercept parameter:

```
BEGIN QUESTION
name: p13b
manual: true
```

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

13c. [1 point] Does the intercept make sense in this context?:

```
BEGIN QUESTION
name: p13c
manual: true
```

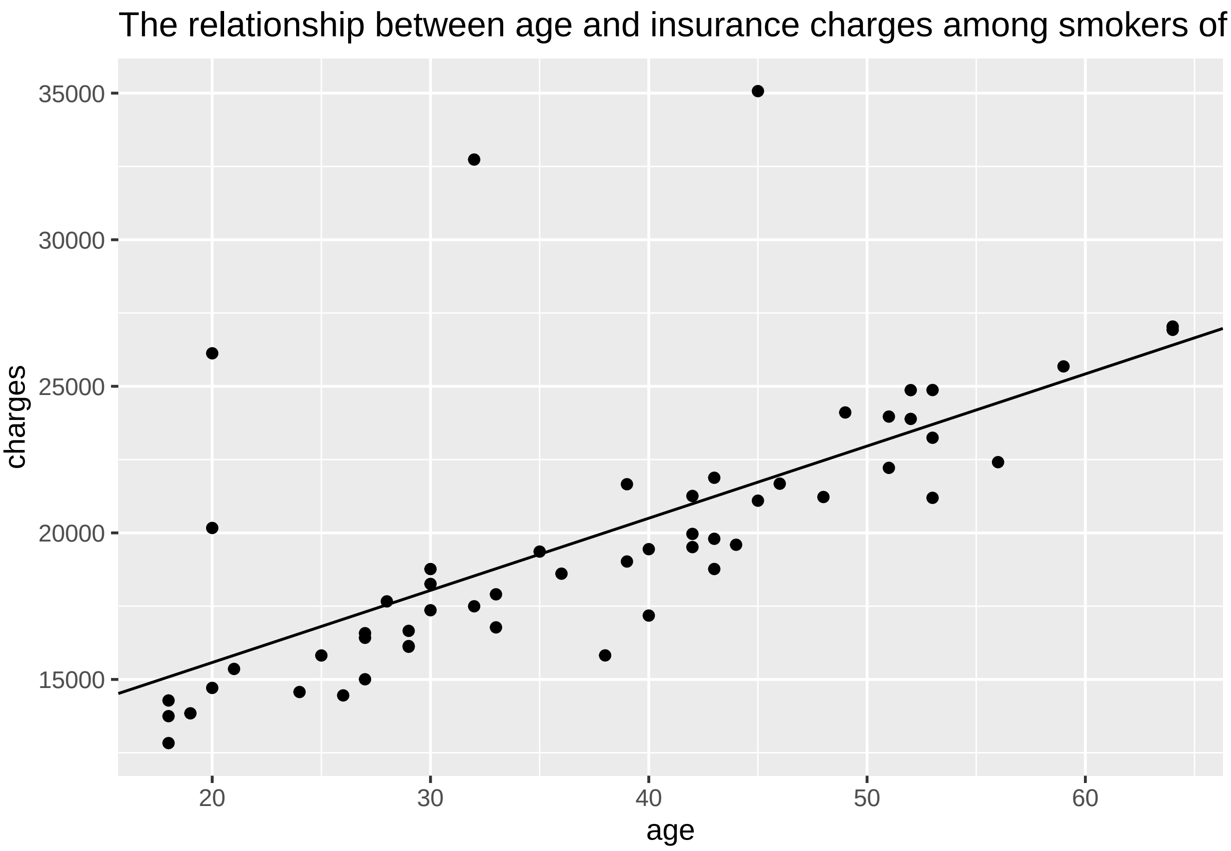
No because being 0 years old is non sensical. Further, the minimum age in the dataset is 18, so extrapolation to 0 is not supported by the data. (student can say either of these items or both.)

14. [1 point] Add the line of best fit to your scatterplot by copying and pasting the plot's code from Problem 11 into the chunk below and adding a `geom` that can be used to add a regression line:

```
BEGIN QUESTION
name: p14
manual: false
points: 1
```

```r
. = " # BEGIN PROMPT
p14 <- NULL # YOUR CODE HERE
p14
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p14 <- 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)
p14
```



```r
# END SOLUTION
```

```r
## Test ##
test_that("p14a", {
  expect_true("ggplot" %in% class(p14))
  print("Checking: p14 is a ggplot")
})
```

```
## [1] "Checking: p14 is a ggplot"
## Test passed
```

```r
## Test ##
test_that("p14b", {
  expect_true(identical(p14$data, insure_subset))
  print("Checking: Using insure_subset")
})
```

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

```r
## Test ##
test_that("p14c", {
  expect_true(rlang::quo_get_expr(p14$mapping$x) == "age")
  print("Checking: age is on the x axis")
})
```

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

```r
## Test ##
test_that("p14d", {
  expect_true(rlang::quo_get_expr(p14$mapping$y) == "charges")
  print("Checking: charges is on the y axis")
})
```

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

```r
## Test ##
test_that("p14e", {
  expect_true("GeomPoint" %in% class(p14$layers[[1]]$geom))
  print("Checking: A scatterplot was made")
})
```

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

```r
## Test ##
test_that("p14f", {
  expect_true(length(p14$labels$title) != 0)
  print("Checking: Title added")
})
```

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

```
## Test ##
test_that("p14g", {
  expect_true("GeomAbline" %in% class(p14$layers[[2]]$geom) | "GeomSmooth" %in% class(p14$layers[[2]]$ge
  print("Checking: A line of best fit has been added")
})
```

```
## [1] "Checking: A line of best fit has been added"
## Test passed
```

15. [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: p15
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 by removing eval = F from the chunk header and executing the code chunk.

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

16. [**2 points**] **How many individuals were removed? Who were they?:**

```
BEGIN QUESTION
name: p16
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.

17. [2 points] Run a regression model on `insure_smaller_subset` between `charges` and `age`. Assign it to insure_better_model and look at the output using the `tidy()` function, as was done with the previous linear model.

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

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

# BEGIN SOLUTION NO PROMPT
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("p17a", {
  expect_true("insure_smaller_subset" == insure_better_model$call$data)
  print("Checking: insure_smaller_subset is the dataset")
})
```

```
## [1] "Checking: insure_smaller_subset is the dataset"
## Test passed
```

```
## Test ##
test_that("p17b", {
  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("p17c", {
  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("p17d", {
  expect_true(all.equal(insure_better_model$coefficients[[2]], 266.8725, tol = 0.01))
  print("Checking: slope value")
})
```

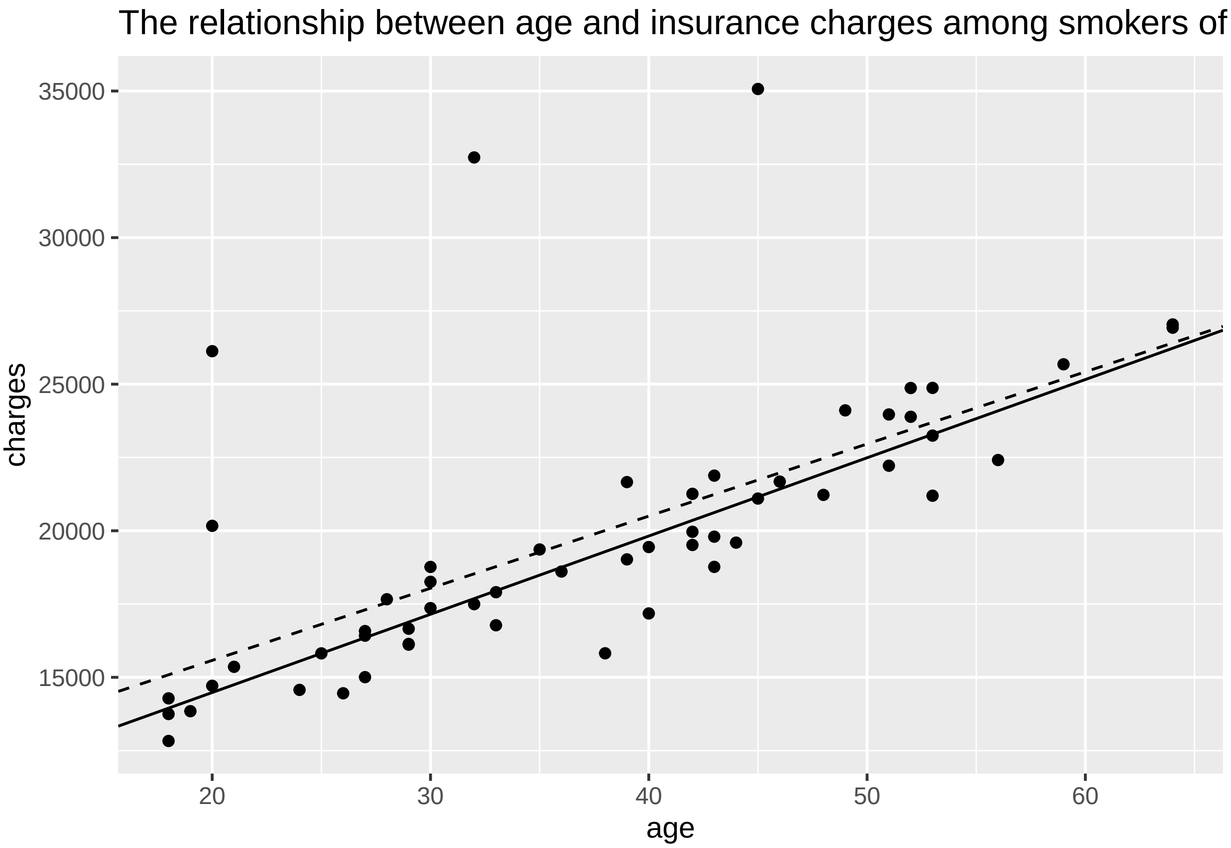
```
## [1] "Checking: slope value"
## Test passed
```

18. [2 points] Add the new regression line to your ggplot from Problem 14. Keep the older regression line on the plot for comparison. To distinguish them, change the color, line type, or line width of the newly-added line.

```
BEGIN QUESTION
name: p18
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p18 <- NULL # YOUR CODE HERE
p18
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p18 <- 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)
p18
```



```
# END SOLUTION
```

```r
## Test ##
test_that("p18a", {
  expect_true("ggplot" %in% class(p18))
  print("Checking: p18 is a ggplot")
})
```

```
## [1] "Checking: p18 is a ggplot"
## Test passed
```

```r
## Test ##
test_that("p18b", {
  expect_true(identical(p18$data, insure_subset))
  print("Checking: Using insure_subset")
})
```

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

```r
## Test ##
test_that("p18c", {
  expect_true(rlang::quo_get_expr(p18$mapping$x) == "age")
  print("Checking: age is on the x axis")
})
```

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

```r
## Test ##
test_that("p18d", {
  expect_true(rlang::quo_get_expr(p18$mapping$y) == "charges")
  print("Checking: charges is on the y axis")
})
```

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

```r
## Test ##
test_that("p18e", {
  expect_true("GeomPoint" %in% class(p18$layers[[1]]$geom))
  print("Checking: A scatterplot was made")
})
```

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

```r
## Test ##
test_that("p18f", {
  expect_true(length(p18$labels$title) != 0)
  print("Checking: Title added")
})
```

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

```
## Test ##
test_that("p18g", {
  expect_true("GeomAbline" %in% class(p18$layers[[2]]$geom) | "GeomSmooth" %in% class(p18$layers[[2]]$ge
  print("Checking: A line of best fit has been added")
})
```

```
## [1] "Checking: A line of best fit has been added"
## Test passed
```

```
## Test ##
test_that("p18h", {
  expect_true("GeomAbline" %in% class(p18$layers[[3]]$geom) | "GeomSmooth" %in% class(p18$layers[[3]]$ge
  print("Checking: A second line of best fit has been added")
})
```

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

19. [1 point] Calculate the r-squared value for `insure_model` using a function learned in class. Assign this value to insure_model_r2.

```
BEGIN QUESTION
name: p19
manual: false
points: 1
```

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

# BEGIN SOLUTION NO PROMPT
insure_model_r2 <- glance(insure_model) %>% pull(r.squared)
# END SOLUTION
```

```
## Test ##
test_that("p19a", {
  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("p19b", {
  expect_true(all.equal(insure_model_r2, 0.449261, tol = 0.01))
  print("Checking: value of insure_model_r2 rounded to 2 decimals")
})
```

```
## [1] "Checking: value of insure_model_r2 rounded to 2 decimals"
## Test passed
```

20. [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: p20
manual: false
points: 1
```

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

# BEGIN SOLUTION NO PROMPT
insure_better_model_r2 <- glance(insure_better_model) %>% pull(r.squared)

# END SOLUTION
```

```
## Test ##
test_that("p20a", {
  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
```

```
## Test ##
test_that("p20b", {
  expect_true(all.equal(insure_better_model_r2, 0.8477642, tol = 0.01))
  print("Checking: value of insure_better_model_r2 rounded to 2 decimals")
})
```

```
## [1] "Checking: value of insure_better_model_r2 rounded to 2 decimals"
## Test passed
```

21. [2 points] **Calculate the correlation between `age` and `charges` using the subset `insure_subset`. Also calculate correlation squared. You should use `summarize()` and name the two new columns `corr` and `corr_sq`. What do you notice about the relationship between the correlation and r-squared values that you calculated earlier?**

```
BEGIN QUESTION
name: p21
manual: false
points: 2
```

```
. = " # BEGIN PROMPT
p21 <- NULL # YOUR CODE HERE
p21
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p21 <- insure_subset %>% summarize(corr = cor(age, charges), corr_sq = corr^2)
p21
```

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

```
# END SOLUTION
```

```
## Test ##
test_that("p21a", {
  expect_true(is.data.frame(p21))
  print("Checking: p21 is a dataframe")
})
```

```
## [1] "Checking: p21 is a dataframe"
## Test passed
```

```
## Test ##
test_that("p21b", {
  expect_true(nrow(p21) == 1)
  print("Checking: summarized used correctly")
})
```

```
## [1] "Checking: summarized used correctly"
## Test passed
```

```
## Test ##
test_that("p21c", {
  expect_true(ncol(p21) == 2)
  print("Checking: two columns in dataframe")
})
```

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

```r
## Test ##
test_that("p21d", {
  expect_true(all.equal(p21$corr[1], 0.6702694, tol = 0.01))
  print("Checking: value of corr column in p21")
})
```

```
## [1] "Checking: value of corr column in p21"
## Test passed
```

```r
## Test ##
test_that("p21e", {
  expect_true(all.equal(p21$corr_sq[1], 0.4492611, tol = 0.01))
  print("Checking: value of corr_sq column in p21")
})
```

```
## [1] "Checking: value of corr_sq column in p21"
## Test passed
```

22. [2 points] Calculate the correlation between `age` and `charges` using the smaller dataset `insure_smaller_subset`. Also calculate correlation squared. You should use `summarize()` and name the two new columns `corr` and `corr_sq`. What do you notice about the relationship between the correlation and r-squared values that you calculated earlier?

```
BEGIN QUESTION
name: p22
manual: false
points: 2
```

```r
. = " # BEGIN PROMPT
p22 <- NULL # YOUR CODE HERE
p22
" # END PROMPT
# BEGIN SOLUTION NO PROMPT
p22 <- insure_smaller_subset %>% summarize(corr = cor(age, charges), corr_sq = corr^2)
# END SOLUTION
```

```r
## Test ##
test_that("p22a", {
  expect_true(is.data.frame(p22))
  print("Checking: p22 is a dataframe")
})
```

```
## [1] "Checking: p22 is a dataframe"
## Test passed
```

```r
## Test ##
test_that("p22b", {
  expect_true(nrow(p22) == 1)
  print("Checking: summarized used correctly")
})
```

```
## [1] "Checking: summarized used correctly"
## Test passed
```

```
## Test ##
test_that("p22c", {
  expect_true(ncol(p22) == 2)
  print("Checking: two columns in dataframe")
})
```

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

```
## Test ##
test_that("p22d", {
  expect_true(all.equal(p22$corr[1], 0.9207411, tol = 0.01))
  print("Checking: value of corr column in p22")
})
```

```
## [1] "Checking: value of corr column in p22"
## Test passed
```

```
## Test ##
test_that("p22e", {
  expect_true(all.equal(p22$corr_sq[1], 0.8477642, tol = 0.01))
  print("Checking: value of corr_sq column in p22")
})
```

```
## [1] "Checking: value of corr_sq column in p22"
## Test passed
```

PART B

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.

23. [1 point] Make a new dataframe called `insure_smokers` that includes smokers of any BMI from the original `insure_data`.

```
BEGIN QUESTION
name: p23
manual: false
points: 1
```

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

```
## Test ##
test_that("p23a", {
  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("p23b", {
  expect_true(nrow(insure_smokers) == 274)
  print("Checking: filtered correctly")
})
```

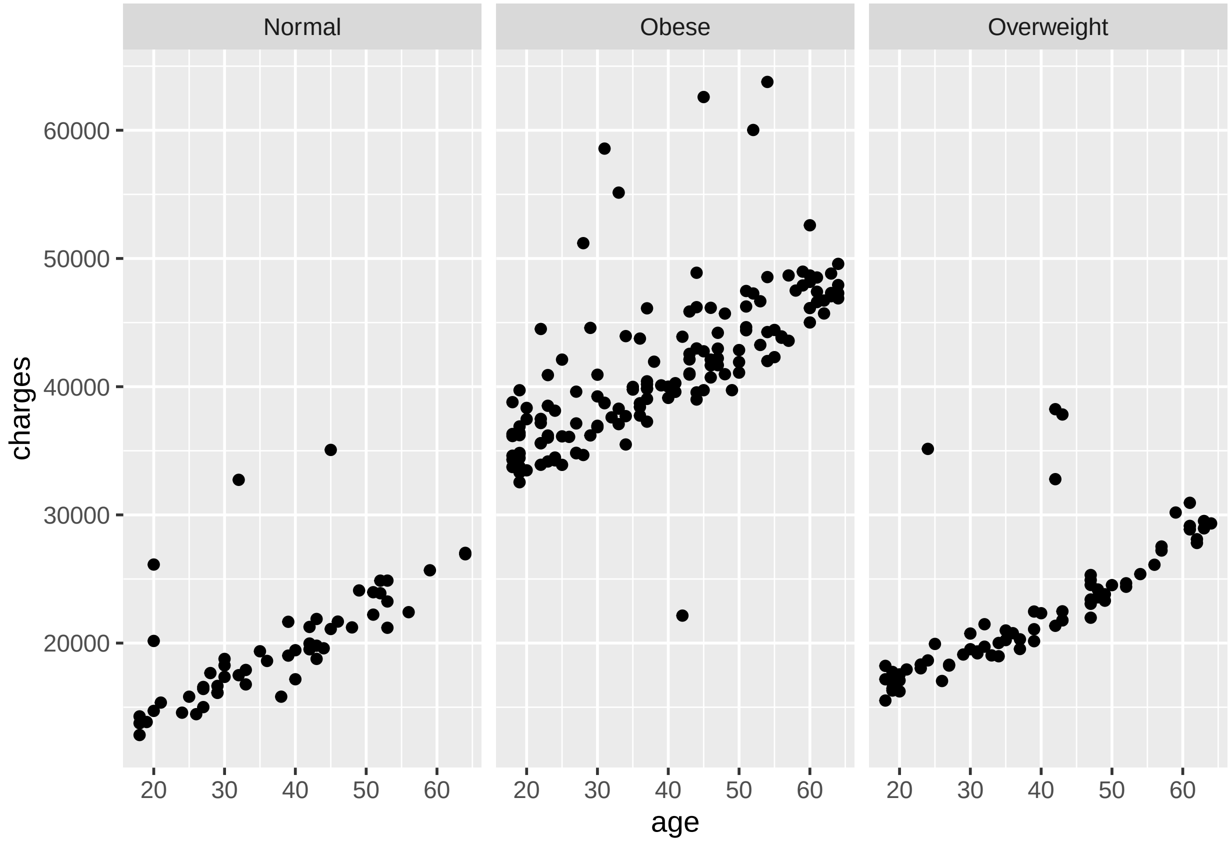
```
## [1] "Checking: filtered correctly"
## Test passed
```

24. [1 point] Make a scatter plot that examines the relationship between `age` and `charges` separately for normal, overweight, and obese individuals. A `facet_` command may help you.

```
BEGIN QUESTION
name: p24
manual: false
points: 1
```

```
. = " # BEGIN PROMPT
p24 <- NULL # YOUR CODE HERE
p24
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p24 <- ggplot(insure_smokers, aes(x = age, y = charges)) +
  geom_point() +
  facet_wrap(~ bmi_cat)
p24
```



```
# END SOLUTION
```

```
## Test ##
test_that("p24a", {
  expect_true("ggplot" %in% class(p24))
  print("Checking: p24 is a ggplot")
})
```

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

```r
## Test ##
test_that("p24b", {
  expect_true(identical(p24$data, insure_smokers))
  print("Checking: Using insure_smokers")
})
```

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

```r
## Test ##
test_that("p24c", {
  expect_true(rlang::quo_get_expr(p24$mapping$x) == "age")
  print("Checking: age is on the x axis")
})
```

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

```r
## Test ##
test_that("p24d", {
  expect_true(rlang::quo_get_expr(p24$mapping$y) == "charges")
  print("Checking: charges is on the y axis")
})
```

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

```r
## Test ##
test_that("p24e", {
  expect_true("GeomPoint" %in% class(p24$layers[[1]]$geom))
  print("Checking: A scatterplot was made")
})
```

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

```r
## Test ##
test_that("p24f", {
  expect_true("FacetWrap" %in% class(p24$facet))
  print("Checking: facet was used")
})
```

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

Is there something out of order with your plot you just made? The issue is that the plot is automatically displayed by listing the BMI categories alphabetically. Remove the eval = F from the code chunk and run the following code chunk to assign an ordering to the values of `bmi_cat`.:

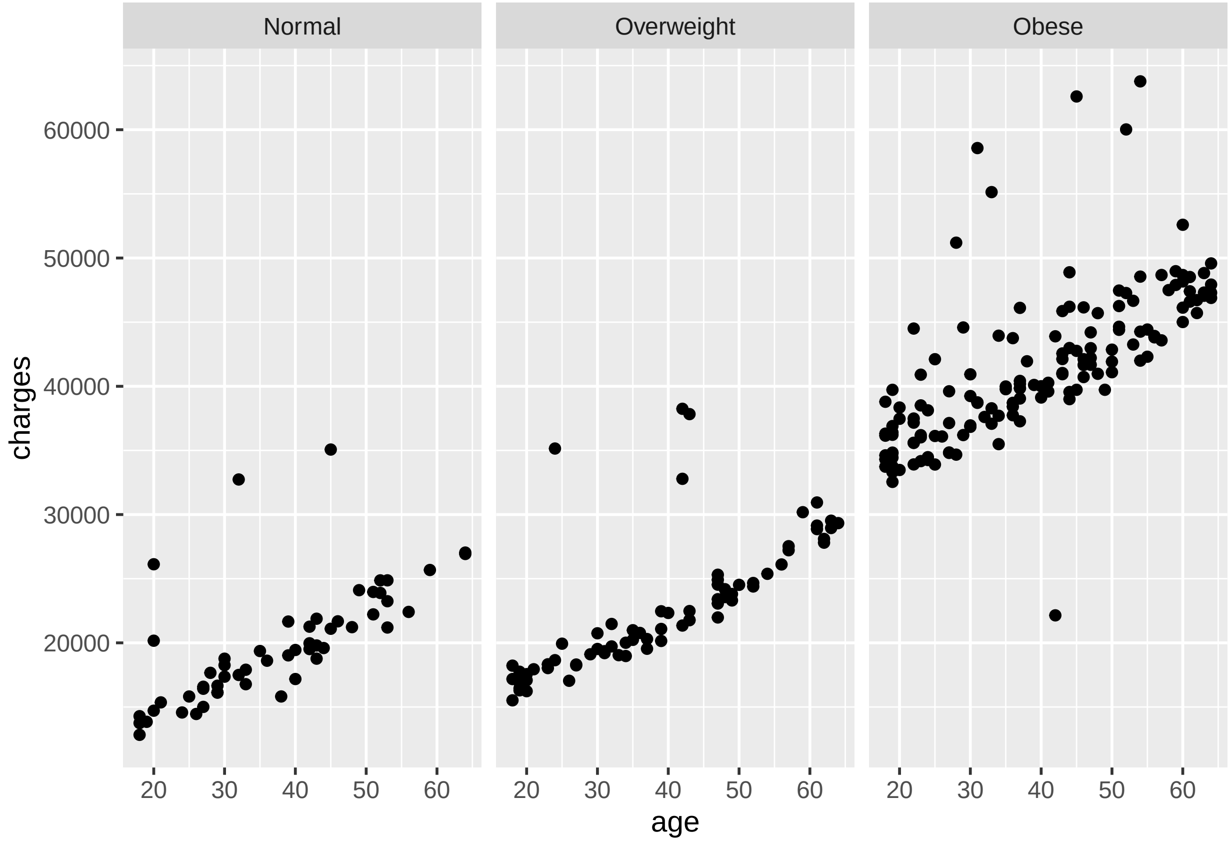
```
insure_smokers <- insure_smokers %>%
  mutate(bmi_cat_ordered = forcats::fct_relevel(bmi_cat, "Normal", "Overweight", "Obese"))
```

25. [1 point] **Re-run your plot code, but this time, facet using `bmi_cat_ordered`.**

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

```
. = " # BEGIN PROMPT
p25 <- NULL # YOUR CODE HERE
p25
" # END PROMPT

# BEGIN SOLUTION NO PROMPT
p25 <- ggplot(insure_smokers, aes(x = age, y = charges)) +
  geom_point() +
  facet_wrap(~ bmi_cat_ordered)
p25
```



```
# END SOLUTION
```

```
## Test ##
test_that("p25a", {
  expect_true("ggplot" %in% class(p25))
  print("Checking: p25 is a ggplot")
})
```

```
## [1] "Checking: p25 is a ggplot"
## Test passed
```

```
## Test ##
test_that("p25b", {
  expect_true(identical(p25$data, insure_smokers))
  print("Checking: Using insure_smokers")
})
```

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

```
## Test ##
test_that("p25c", {
  expect_true(rlang::quo_get_expr(p25$mapping$x) == "age")
  print("Checking: age is on the x axis")
})
```

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

```
## Test ##
test_that("p25d", {
  expect_true(rlang::quo_get_expr(p25$mapping$y) == "charges")
  print("Checking: charges is on the y axis")
})
```

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

```
## Test ##
test_that("p25e", {
  expect_true("GeomPoint" %in% class(p25$layers[[1]]$geom))
  print("Checking: A scatterplot was made")
})
```

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

```
## Test ##
test_that("p25f", {
  expect_true("FacetWrap" %in% class(p25$facet))
  print("Checking: facet was used")
})
```

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

26. [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 first. Call your models `normal_mod`, `overweight_mod`, `obese_mod`. Use the `tidy()` function to display the output from each model.

```
BEGIN QUESTION
name: p26
manual: false
points: 3
```

```r
. = " # 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 NO PROMPT
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
```

```r
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("p26a", {
  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("p26b", {
  expect_true("age" %in% names(normal_mod$model))
  print("Checking: age is in the overweight_mod")
})
```

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

```
## Test ##
test_that("p26c", {
  expect_true("charges" %in% names(normal_mod$model))
  print("Checking: charges is in the overweight_mod")
})
```

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

```
## Test ##
test_that("p26d", {
  expect_true(all.equal(normal_mod$coefficients[[2]], 246.1367, tol = 0.01))
  print("Checking: slope value in normal_mod")
})
```

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

```
## Test ##
test_that("p26e", {
  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("p26f", {
  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("p26g", {
  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("p26h", {
  expect_true(all.equal(overweight_mod$coefficients[[2]], 264.1862, tol = 0.01))
  print("Checking: slope value in overweight_mod")
})
```

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

```
## Test ##
test_that("p26i", {
  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("p26j", {
  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("p26k", {
  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("p26l", {
  expect_true(all.equal(obese_mod$coefficients[[2]], 281.1528, tol = 0.01))
  print("Checking: slope value in obese_mod")
})
```

```
## [1] "Checking: 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 model. Show your work for each calculation by writing the mathematical expression in and round to the nearest dollar.

27. [1 point] **. . . among normal BMI group:**

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

```
p27 <- 10656.1 + 246.1 * 20 # = $15578.1 # SOLUTION
```

```
## Test ##
test_that("p27a", {
  expect_true(is.numeric(p27))
  print("Checking: p27 is a number")
})
```

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

```
## Test ##
test_that("p27b", {
  expect_true(all.equal(p27,15578.1, tol = 0.1))
  print("Checking: value of p27")
})
```

```
## [1] "Checking: value of p27"
## Test passed
```

28. [1 point] **. . . among overweight BMI group:**

```
BEGIN QUESTION
name: p28
manual: false
points: 1
```

```
p28 <- 12399.7 + 264.2 * 20 # = $17683.7 # SOLUTION
```

```
## Test ##
test_that("p28a", {
  expect_true(is.numeric(p28))
  print("Checking: p28 is a number")
})
```

```
## [1] "Checking: p28 is a number"
## Test passed
```

```r
## Test ##
test_that("p28b", {
  expect_true(all.equal(p28, 17683.7, tol = 0.1))
  print("Checking: value of p28")
})
```

```
## [1] "Checking: value of p28"
## Test passed
```

29. [1 point] . . . among obese BMI group:

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

```r
p29 <- 30558.1 + 281.2 * 20 # = $36182.1 # SOLUTION
```

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

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

```r
## Test ##
test_that("p29b", {
  expect_true(all.equal(p29, 36182.1, tol = 0.1))
  print("Checking: value of p29")
})
```

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

30. [3 points] In three sentences maximum, (1) comment on the direction of the association, (2) comment on how much the slopes vary across the BMI groups, and (3) how much the prediction for a 20-year old varies.

```
BEGIN QUESTION
name: p30
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.