Lab 10: Proportions and Inference for Regression

Nimita Gaggar (3039677409)

Today's date

Run this chunk of code to load the autograder package!

Instructions

- Due date: Monday, August 7th, at 10:00pm PST with 2 hour grace period.
- Late penalty: 50% late penalty if submitted within 24 hours of due date, no marks for assignments submitted thereafter.
- This assignment is graded on **correct completion**, all or nothing. You must pass all public tests and submit the assignment for credit.
- Submission process: Follow the submission instructions on the final page. Make sure you do not remove any `\newpage` tags or rename this file, as this will break the submission.

Boston Data on Median Household Value and Distance to Employment Centers

We are examining a dataset used to predict housing prices in the area around Boston (Harrison, D. and Rubinfeld, 1978). We wish to specifically examine the association of the measure of housing price (`medv`, median value of owner-occupied homes in the $1000s) and a measure of adjacency to employment (a weighted distance, `dis`, in miles). The data frame (Boston) is contained in another package (`MASS`), which we load into the object `boston2` below.

```
### NOTE: All of the code is to get you started on the lab. You do not need to
### understand any functions below that you have not seen before.

# Load library with data
library(MASS)

### NOTE: This package has a function `select()` that can be confused with
### dplyr's select. To overcome this, we first import the data we need and then
### detach the library before loading dplyr.

# List variables
boston2 <-
  read.csv("data/Boston.csv")

# Variable definition - take a quick look at the variables in the data frame
# help(Boston)
detach(package:MASS)
library(broom)
library(dplyr)
library(ggplot2)
```

```r
library(tidyr)
library(patchwork)
library(testthat)

### Normally when we are doing inference, we take a random sample to make an inference about the popula
### If you have time after the lab, take a random sample of 50 rows from the data and perform the analy
```

Section 1: Inference for Regression

1. [1 point] Use the `boston2` data to perform a linear regression of `medv` (median value of owner-occupied homes in $1000s) versus `dis` (weighted mean of distances to five Boston employment centers) and tidy the results. Assign the *linear model* to `p1`.

```
p1 <- lm(medv~dis, data = boston2)

p1
```

```
##
## Call:
## lm(formula = medv ~ dis, data = boston2)
##
## Coefficients:
## (Intercept)          dis
##      18.390        1.092
```

```
tidy(p1)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    18.4      0.817     22.5  4.01e-78
## 2 dis             1.09     0.188      5.79 1.21e- 8
```

```
. = ottr::check("tests/p1.R")
```

```
##
## All tests passed!
```

2. Interpret the slope parameter and the p-value from the output above. What null and alternative hypotheses does this p-value refer to?

A one unit increase in weight mean distance is associated with an estimated $1092 increase in the median home value. Because our p-value is very close to zero we reject the null . The null being that Beta is Zero(there is no association between the distance and prize). Alternative is association is different from zero

3. [1 point] Derive a 95% CI for this slope parameter and assign the lower and upper bounds to p3. Round your bounds to 4 decimal places. In your opinion, would you expect the direction of this relationship to hold if the data were collected today?

```
lower_bound<- round(1.091613     - qt(0.975, df = 504)* 0.1883784   , 4)
upper_bound <- round(1.091613    + qt(0.975, df = 504)* 0.1883784   , 4)
p3 <- c(lower_bound, upper_bound)
p3
```

```
## [1] 0.7215 1.4617
```

May be today, we don't care much about the distance.

```
. = ottr::check("tests/p3.R")
```

```
##
## All tests passed!
```

4. [1 point] Use a function to look at the r-squared value for this model. Assign this value to `p4` and round to two decimal places. Does `dis` explain a lot of the variance in median household value? Would you expect it to?

```
glance(lm(medv~dis, data = boston2))
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic      p.value    df logLik   AIC   BIC
##       <dbl>         <dbl> <dbl>     <dbl>        <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1    0.0625        0.0606  8.91      33.6 0.0000000121     1 -1824. 3654. 3667.
## # i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
p4<- round(0.06060418   ,2)
p4
```

```
## [1] 0.06
```

No the distance variable explains only 6% of the variance in the prize

```
. = ottr::check("tests/p4.R")
```

```
##
## All tests passed!
```

5. [2 points] Make a plot with the raw data points, use `geom_smooth()` to add the line of best fit from the simple linear regression model (containing `medv` and `dis`), and use `geom_abline()` to add a horizontal line with a slope of 0 that crosses the y-axis at the average value of `medv` to vertically bisect the data points. Store your plot as the object `p5`.

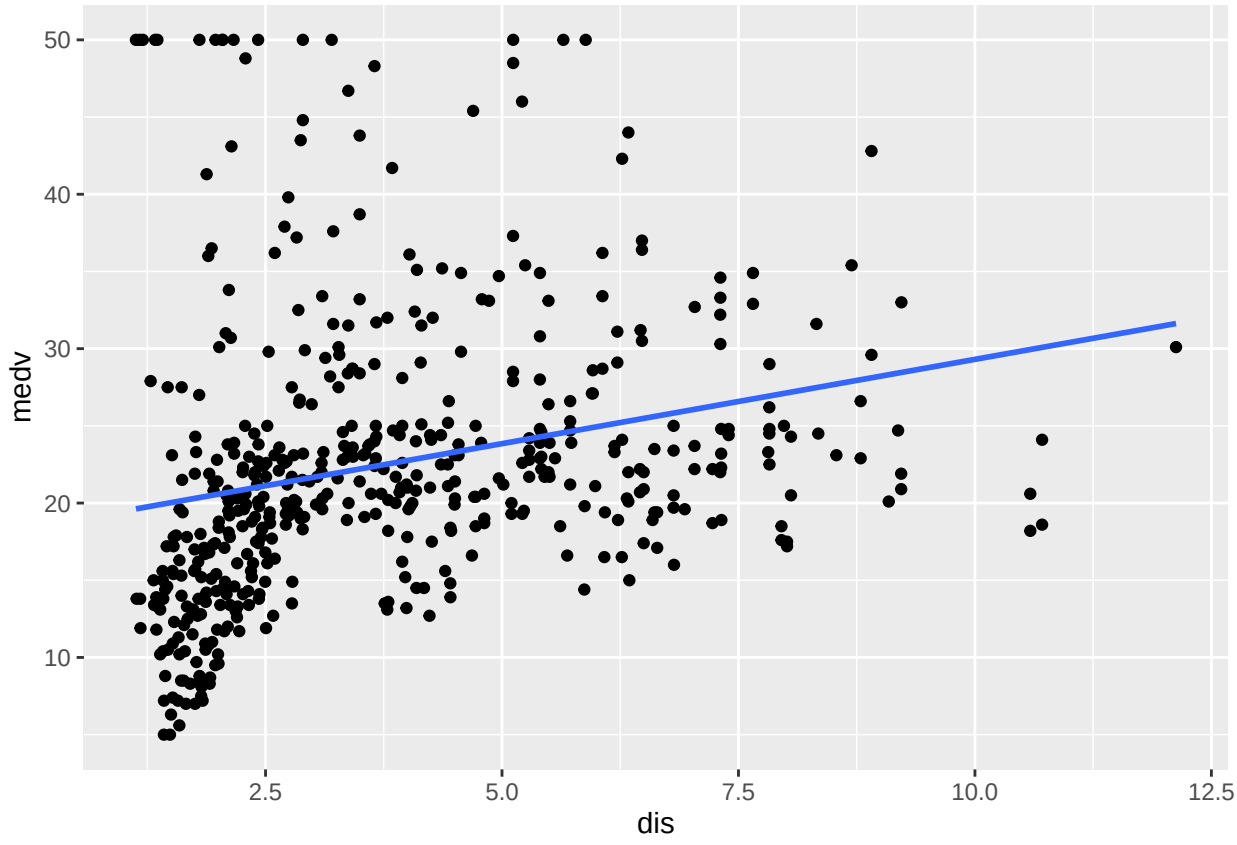
```
p5 <- ggplot(boston2,aes( x = dis,y = medv)) + geom_point() +
      geom_smooth(method = "lm", se = F)+
      geom_abline(slope=0, intercept = mean(boston2))
```

```
## Warning in mean.default(boston2): argument is not numeric or logical: returning
## NA
```

```
p5
```

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

```
## Warning: Removed 1 rows containing missing values (`geom_abline()`).
```




```
. = ottr::check("tests/p5.R")
```

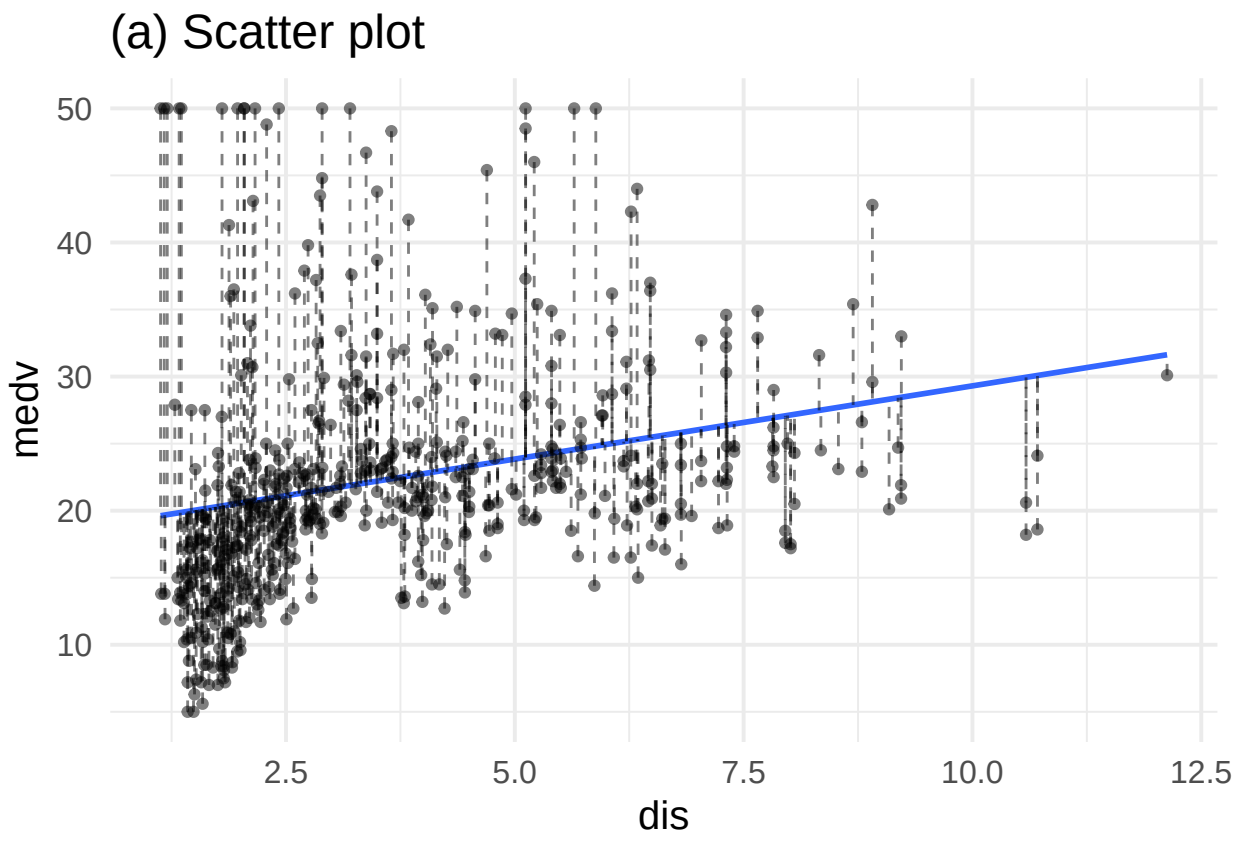
```
##
## All tests passed!
```

6. Create plots to check the assumptions required for using simple linear models. You will first need to fit the linear model and then use the `augment()` function from the `broom` package to store the residuals and fitted values into a new data frame. Do the plots raise any concerns about the assumptions of the linear regression you just performed?

```
data_with_prediction <- augment(lm(medv~dis, data = boston2))

# first plot
plot1 <- ggplot(data_with_prediction, aes(x = dis,y = medv)) +
geom_smooth(method = "lm", se = F) +
geom_point(alpha = 0.5) +
geom_segment(aes(xend = dis, yend = .fitted), lty = 2, alpha = 0.5) +
theme_minimal(base_size = 15) +
labs(title = "(a) Scatter plot")
plot1
```

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




```
# second plot
plot2 <- ggplot(data_with_prediction, aes(sample = .resid)) +
geom_qq() + geom_qq_line()
theme_minimal(base_size = 15) +
labs(x = "Theoritical Quantiles", y = "Residuals", title = "b (QQplot)")
```

```
## List of 99
##  $ line                        :List of 6
```

```
##   ..$ colour       : chr "black"
##   ..$ linewidth    : num 0.682
##   ..$ 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"
##   ..$ linewidth    : num 0.682
##   ..$ 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 15
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ title                     : chr "b (QQplot)"
##  $ 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] 3.75points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 0points 3.75points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 3.75points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 0points 0points 3.75points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 3points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 0points 3points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 3points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 0points 0points 3points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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         : 'simpleUnit' num 3.75points
##   ..- attr(*, "unit")= int 8
##  $ 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] 7.5points 7.5points 7.5points 7.5points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing            : 'simpleUnit' num 15points
##   ..- attr(*, "unit")= int 8
##  $ legend.spacing.x          : NULL
##  $ legend.spacing.y          : NULL
##  $ legend.key                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.key.size           : 'simpleUnit' num 1.2lines
##   ..- attr(*, "unit")= int 3
##  $ 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
```

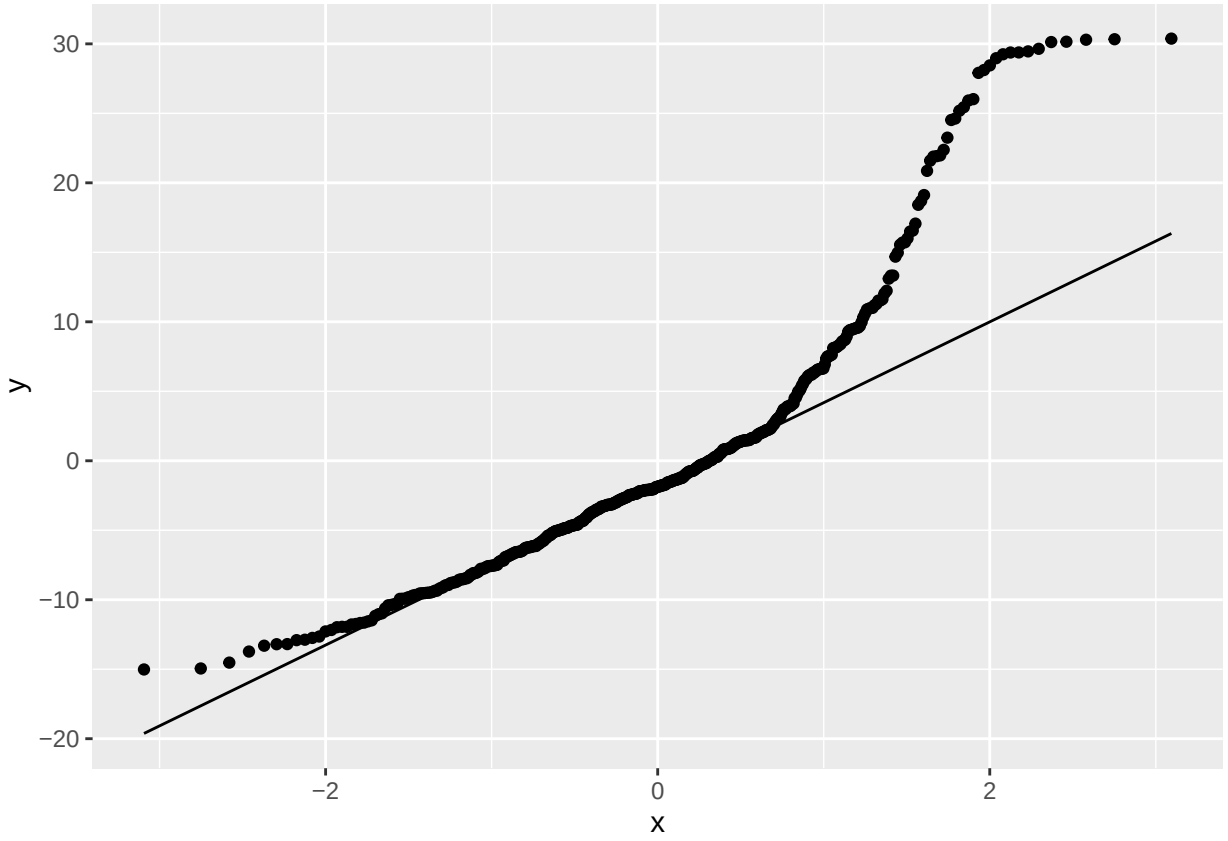
```
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ 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(*, "unit")= int 1
##  $ legend.box.background     : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ legend.box.spacing        : 'simpleUnit' num 15points
##   ..- attr(*, "unit")= int 8
##  $ panel.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.border              : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ panel.spacing             : 'simpleUnit' num 7.5points
##   ..- attr(*, "unit")= int 8
##  $ panel.spacing.x           : NULL
##  $ panel.spacing.y           : NULL
##  $ panel.grid                :List of 6
##   ..$ colour       : chr "grey92"
##   ..$ linewidth    : 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
##   ..$ linewidth    : '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] 0points 0points 7.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 0points 0points 7.5points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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] 7.5points 0points 0points 0points
##   .. ..- attr(*, "unit")= int 8
##   ..$ 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
```

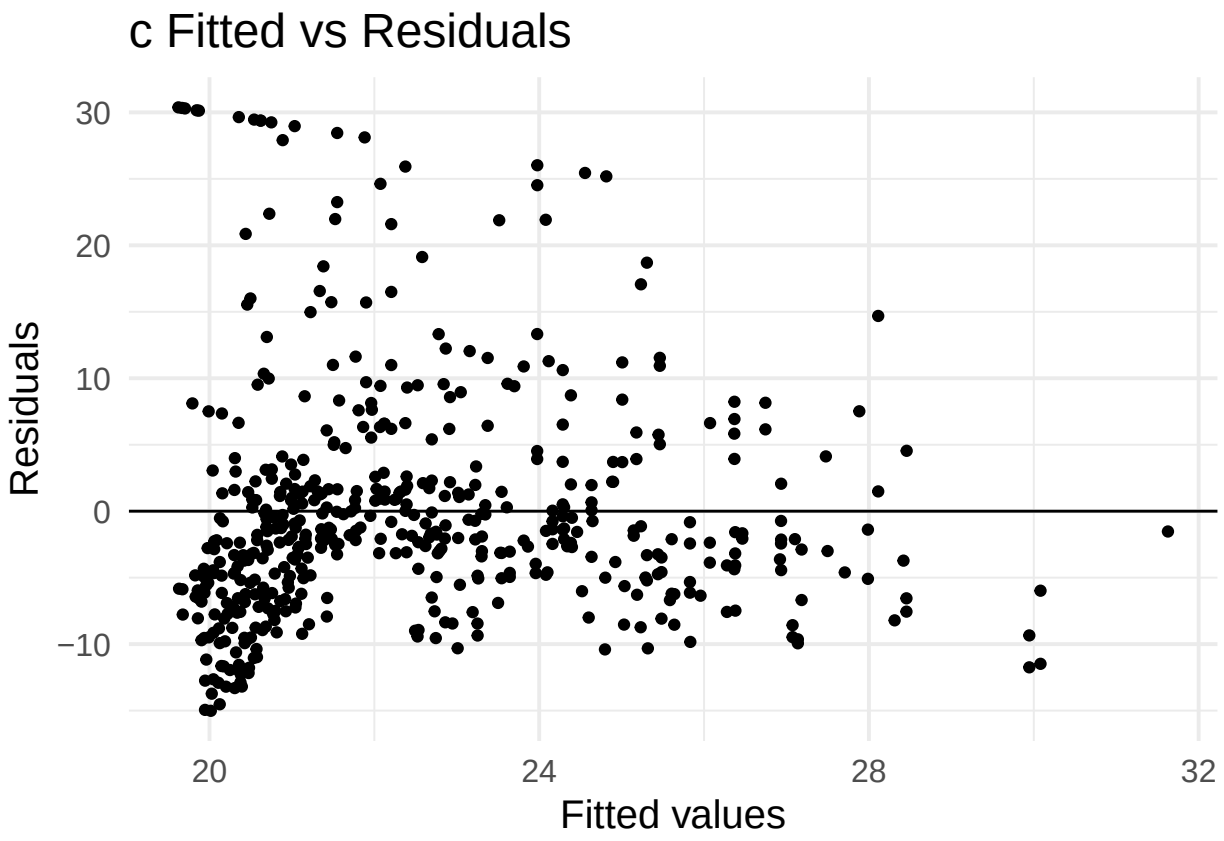
```
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'margin' num [1:4] 7.5points 7.5points 7.5points 7.5points
##   ..- attr(*, "unit")= int 8
##  $ strip.background          : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ strip.background.x        : NULL
##  $ strip.background.y        : NULL
##  $ strip.clip                : chr "inherit"
##  $ 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] 6points 6points 6points 6points
##   .. ..- attr(*, "unit")= int 8
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.text.x              : NULL
##  $ strip.text.x.bottom       : NULL
##  $ strip.text.x.top          : 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.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"
```

```
##  $ strip.text.y.right        : NULL
##  $ strip.switch.pad.grid     : 'simpleUnit' num 3.75points
##   ..- attr(*, "unit")= int 8
##  $ strip.switch.pad.wrap     : 'simpleUnit' num 3.75points
##   ..- attr(*, "unit")= int 8
##  $ x                         : chr "Theoritical Quantiles"
##  $ y                         : chr "Residuals"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

```
plot2
```




```
# third plot
plot3 <- ggplot(data_with_prediction, aes(y = .resid, x = .fitted)) +
geom_point() +
theme_minimal(base_size = 15) +
geom_hline(aes(yintercept = 0)) +
  labs(x = "Fitted values", y = "Residuals", title = "c Fitted vs Residuals")
plot3
```

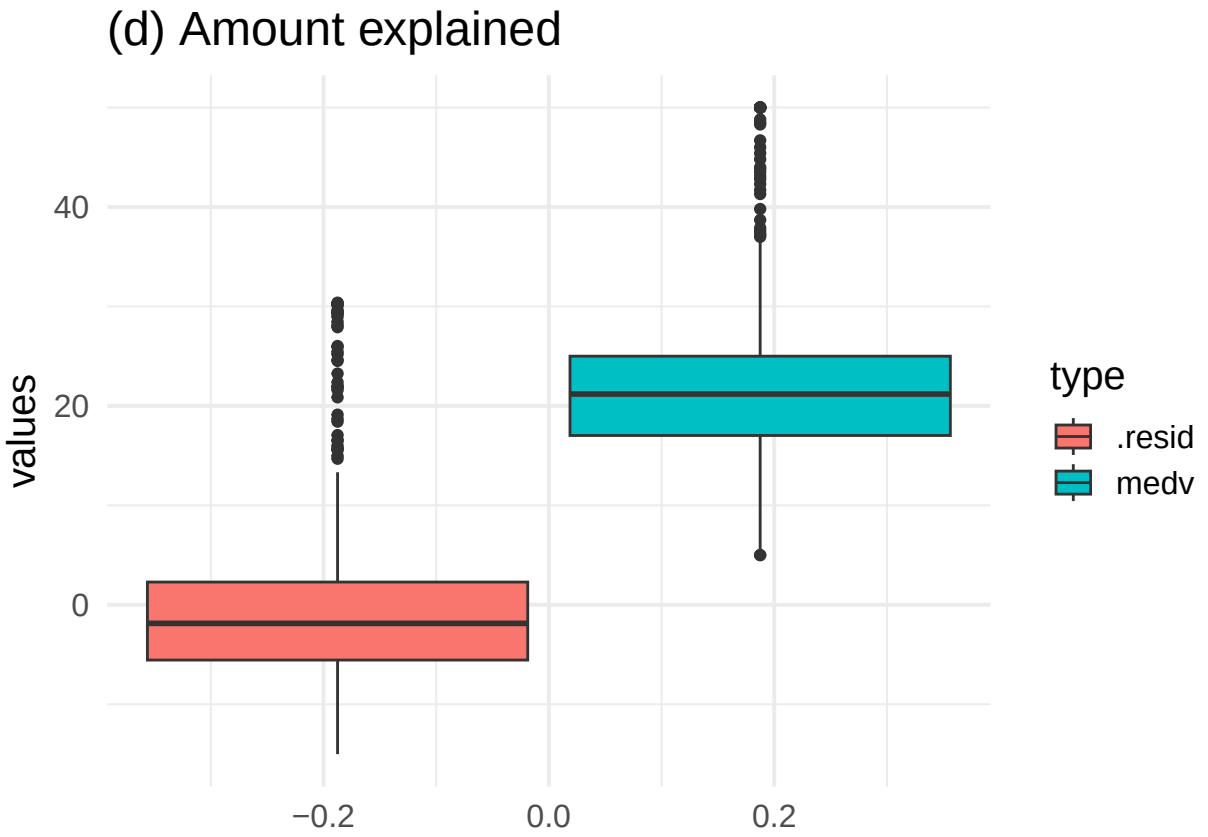



```
plot4 <- predictions_reshaped <- data_with_prediction %>% select( medv, .resid)%>%
  gather(key = "type", value = "values", medv, .resid )

plot4
```

```
## # A tibble: 1,012 x 2
##    type  values
##    <chr>  <dbl>
##  1 medv    24
##  2 medv    21.6
##  3 medv    34.7
##  4 medv    33.4
##  5 medv    36.2
##  6 medv    28.7
##  7 medv    22.9
##  8 medv    27.1
##  9 medv    16.5
## 10 medv    18.9
## # i 1,002 more rows
```

```
plot5<- ggplot(predictions_reshaped, aes(y = values)) +
        geom_boxplot(aes(fill = type)) +
       theme_minimal(base_size = 15) +
        labs(title = "(d) Amount explained")
plot5
```



Based on the graphs, we would have concerns regarding linearity of the model, the normality of the residuals, equal variance of the residuals

Regardless of your answer, we will continue using the model to make inferences about the relationship between `med` and `dis`.

Pointwise Confidence Intervals and Multiple Testing

As you learned in lecture, there are two types of confidence intervals applicable to estimating a point on the plot which are related to whether one is predicting the population average among individuals with $X = x$ (**mean response**) or whether one is predicting the actual Y for a particular individual (**single observation**). For this assignment, we will concentrate on the confidence interval for the mean response. We do so because it is rare to use statistical models in public health as forecasting models (predicting an individual's health in the future) and is more common to use them to estimate population-level changes (how the mean of a health variable changes in a population as we change exposure). However, as precision medicine becomes more of a reality and models more accurately predict health (i.e., have high R^2's), then statistical forecasting may become more common in our field.

7. [1 point] Calculate four 95% confidence intervals for the mean response, one at each `dis` value: 2.5, 5.0, 7.5, and 10.0 miles. Create a vector of the lower bounds for each confidence interval rounded to two decimal places and assign it to `p12`.

Hint: Use the `predict()` function and be sure to specify interval = "confidence"

OPTIONAL: If time allows, add the four CIs to a scatter plot of the data (along with the line of best fit).

```
ci_dataframe <- data.frame(dis = c(2.5, 5.0, 7.5, 10))
ci_dataframe
```

```
##    dis
## 1  2.5
## 2  5.0
## 3  7.5
## 4 10.0
```

```
predict(p1, newdata = ci_dataframe, interval = "confidence" ) %>%
      cbind(ci_dataframe)
```

```
##        fit      lwr      upr  dis
## 1 21.11912 20.20485 22.03339  2.5
## 2 23.84815 22.95092 24.74539  5.0
## 3 26.57719 25.00035 28.15402  7.5
## 4 29.30622 26.88135 31.73108 10.0
```

```
p7 <- c (20.20, 22.95, 25.00, 26.88)
```

```
. = ottr::check("tests/p7.R")
```

```
##
## All tests passed!
```

8. Interpret the point wise 95% confidence interval of the median house price when the distance = 10.

If we use the same sampling method and take 100 samples of houses in Boston with a distance of 10 miles to the Boston employment center, we would expect the confidence interval of 95% of those samples to contain the true neighborhood median home value to something between 26.88 and 31.73(in thousands of dollar)

9. Do the CI's differ in length for different values of `dis`? Why or why not?

CI,s do differ. They tend to be narrower around the mean.

Section 2: Inference for Proportions

The rest of the lab assignment is for practice only. You are responsible for understanding the material but these questions will not be graded.

Tests of Changes in Sex Ratios Based on a Single Sample

There is a long literature studying changes in sex-ratios of births due to stressful events, such as 9/11. In today's lab, we consider a relatively small study that recorded biomarkers of stress on pregnancy. In the group of subjects that had the highest markers of stress (based on cortisol), there were 14 births to males out of a total of 38.

In this lab, we will compare the four methods we learned to calculate CIs for proportions. Recall that two of these methods involved hand calculations (though we can treat R as if it were a calculator) and two of the methods used built-in R functions.

10. Use the Normal approximation to construct a 95% confidence interval in this high stress group. We also called this specific method of constructing the CI the "large sample method". Assign the object `large_sample_ci` to a vector of the lower bound and upper bound rounded to 4 decimal places.

```
p.hat <- 14/38
se <- sqrt(p.hat * (1-p.hat) / 38)

large_sample_ci <- c(round(p.hat- 1.96 * se, 4),
                     round(p.hat+ 1.96 * se, 4) )
large_sample_ci
```

```
## [1] 0.2150 0.5218
```

Type your answer here, replacing this text.

```
. = ottr::check("tests/p10.R")
```

```
## Test p10 - 1 passed
```

11. Create the 95% CI again, this time using the R function that implements the Wilson Score method with a continuity correction. Round your answer to 4 decimal places.

```
wilson_score_ci <- c (round(0.2229295, 4), round (0.5400424, 4))
wilson_score_ci
```

```
## [1] 0.2229 0.5400
```

Type your answer here, replacing this text.

```
. = ottr::check("tests/p11.R")
```

```
## Test p11 - 1 passed
```

12. Create the 95% CI again, this time using the Plus 4 method. Round your answer to 4 decimal places.

```
p_tilde <- (14+2) / (38+4)
p_tilde
```

```
## [1] 0.3809524
```

```
se <- sqrt(p_tilde * (1- p_tilde)/ 42)
plus_4_ci <- c(round(p_tilde - 1.96 * se, 4), round(p_tilde + 1.96 * se, 4))
plus_4_ci
```

```
## [1] 0.2341 0.5278
```

Type your answer here, replacing this text.

```
. = ottr::check("tests/p12.R")
```

```
## Test p12 - 1 passed
```

13. Create the 95% CI again, this time using the R function that implements the Clopper Pearson (Exact) method. Round your answer to 4 decimal places.

```
binom.test (14, 38, p = 0.5)
```

```
##
##  Exact binomial test
##
## data:  14 and 38
## number of successes = 14, number of trials = 38, p-value = 0.1433
## alternative hypothesis: true probability of success is not equal to 0.5
## 95 percent confidence interval:
##  0.2181250 0.5400572
## sample estimates:
## probability of success
##              0.3684211
```

```
exact_method_ci <- c(round( 0.2181250 , 4), round( 0.5400572, 4))
exact_method_ci
```

```
## [1] 0.2181 0.5401
```

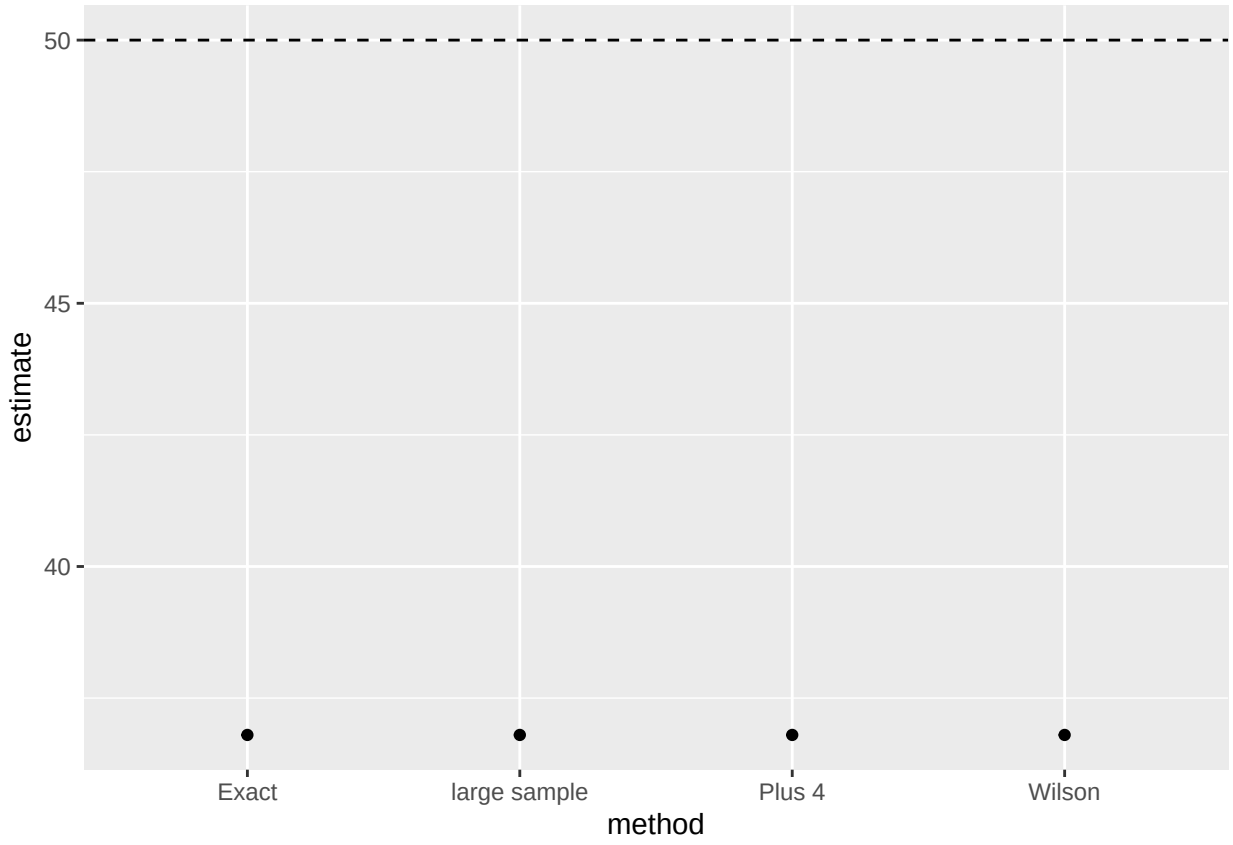
Type your answer here, replacing this text.

```
. = ottr::check("tests/p13.R")
```

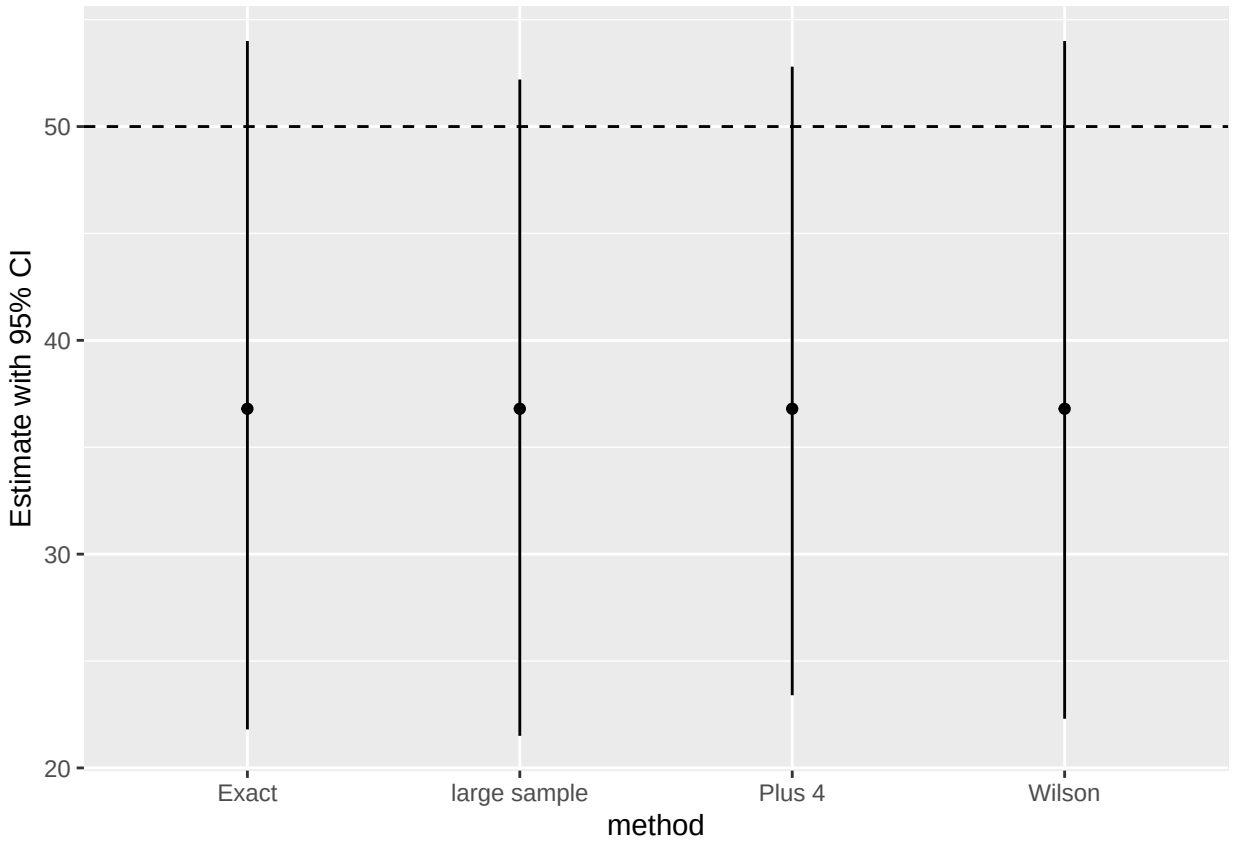
```
## Test p13 - 1 passed
```

14. Here is a code template to help you to graphically present these estimates. Graphical presentations of estimates and their CIs is very useful for assessing whether the CIs overlap the null hypothesized value and tends to be better than presenting tables of estimates to readers of your research. Fill in the code below with your estimates for each confidence interval method, assign `p14` to the plot of the confidence intervals, then answer the question below.

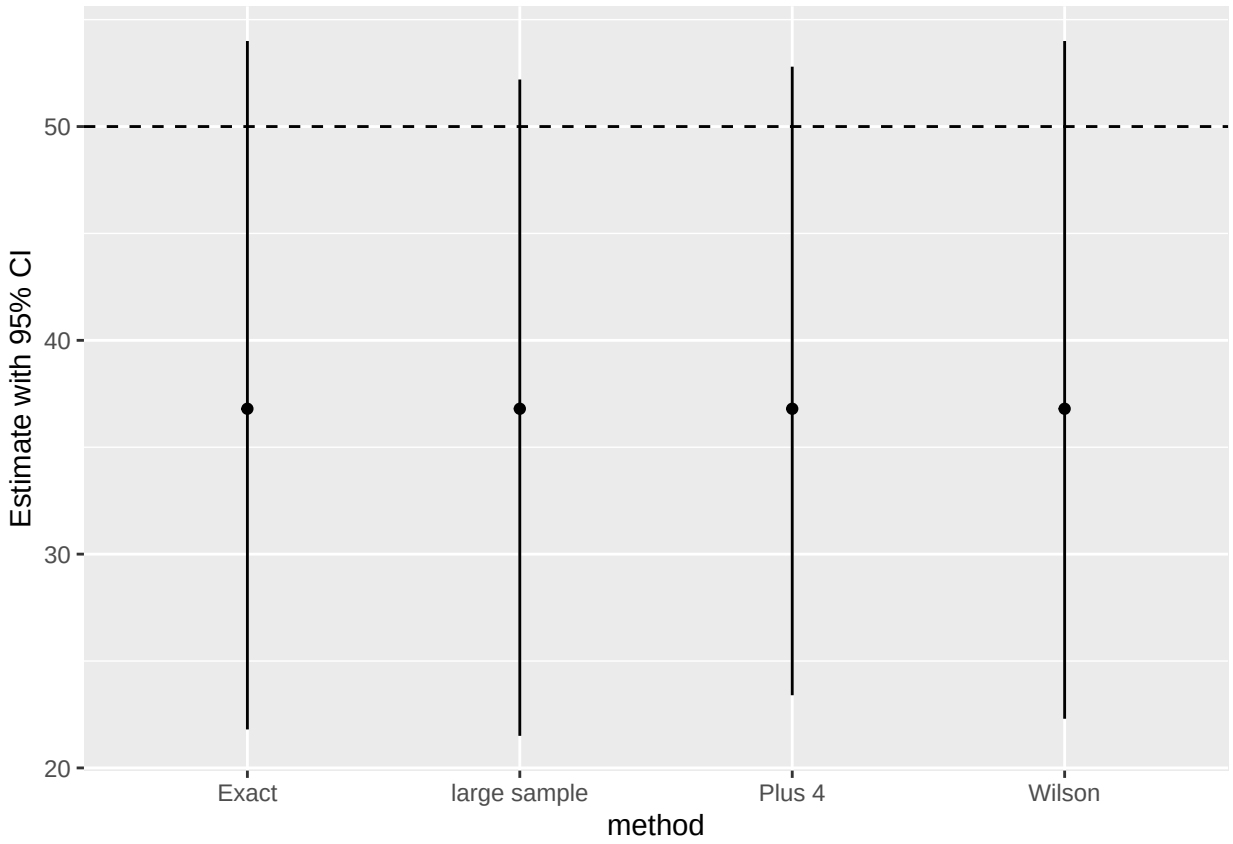
```
# First make a tibble (an easy way to make a data frame) with the data about
# each confidence interval. To do this, replace each instance of 0.00 with the
# estimate from your calculations above.
sex_CIs <- tibble(method   = c("large sample", "Exact", "Wilson", "Plus 4"),
                  lower_CI = c(21.5           , 21.8    ,  22.3    ,  23.4),
                  upper_CI = c(52.2           , 54.0    ,  54.0    ,  52.8),
                  estimate = c(36.8           , 36.8    ,  36.8    ,  36.8)
                  )
# Build the ggplot incrementally, to understand how it works.
# Step 1: (why do we put a horizontal line at 50?)
ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2)
```




```
# Step 2:
ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2) +
  geom_segment(aes(x = method, xend = method, y = lower_CI, yend = upper_CI)) +
  labs(y = "Estimate with 95% CI")
```




```
p14 <- ggplot(data = sex_CIs, aes(x = method, y = estimate)) +
  geom_point() +
  geom_hline(aes(yintercept = 50), lty = 2) +
  geom_segment(aes(x = method, xend = method, y = lower_CI, yend = upper_CI)) +
  labs(y = "Estimate with 95% CI")
p14
```



What does `geom_segment()` do? In particular, what do `x`, `xend`, `y` and `yend` specify in this case?

It creates the graphical confidence intervals using the upper and lower bounds.

```
. = ottr::check("tests/p14.R")
```

```
## Test p14 - 1 passed
```

15. Based on this plot, what can you say about the confidence intervals for the sex ratio in the high stress group?

Because the null is contained in the 4 confidence interval, we fail to reject the null that there is no relationship between the stress level and the predicted sex of the body.

Submission

For assignments in this class, you'll be submitting using the **Terminal** tab in the pane below. In order for the submission to work properly, make sure that:

1. Any image files you add that are needed to knit the file are in the `src` folder and file paths are specified accordingly.
2. You **have not changed the file name** of the assignment.
3. The file knits properly.

Once you have checked these items, you can proceed to submit your assignment.

1. Click on the **Terminal** tab in the pane below.
2. Copy-paste the following line of code into the terminal and press enter.

cd; cd ph142-su23/lab/lab10; python3 turn_in.py

3. Follow the prompts to enter your Gradescope username and password.
4. If the submission is successful, you should see "Submission successful!" appear as the output. **Check your submission on the Gradescope website to ensure that the autograder worked properly and you received credit for your correct answers. If you think the autograder is incorrectly grading your work, please post on Ed!**
5. If the submission fails, try to diagnose the issue using the error messages–if you have problems, post on Ed under the post "Datahub Issues".

The late policy will be strictly enforced, **no matter the reason**, including submission issues, so be sure to submit early enough to have time to diagnose issues if problems arise.

END