Social Sciences Intro to Statistics

Week 7.2 Introduction to Bivariate Regression (continued)

Week 7: Learning goal - Demonstrate estimation and prediction of bivariate regression analysis in R

Introduction

Lecture overview:

-Estimation -Population mean -Regression -Writing out models

-Regression in R - introducing tidymodels - understanding object created by regression

- Prediction

Load packages:

```
library(tidyverse)
library(ggplot2)
library(haven)

load(url('https://raw.githubusercontent.com/bcl96/Social-Sciences-Stats/main/data/els/output_

# ELS data frames
els <- df_els_stu_allobs_fac
```

Conceptually, the next step in learning about regression is understanding how we choose estimates of β_0 and β_1 using sample data? This step is called "estimation"

Before providing a conceptual explanation of estimation, we will teach you about running regression in *R*. Then, we'll use output from these regression models to aid our conceptual explanation of estimation

Run models using `lm()`

We use the `lm()` function to run linear regression models

- Description
  - "`lm` is used to fit linear models"
- syntax (including default values)
  - `lm(formula, data, subset, weights, na.action, method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE, contrasts = NULL, offset, ...)`
- selected arguments
  - `formula`: "an object of class 'formula' " that provides "a symbolic description of the model to be fitted."
    - follows form `formula = y_var ~ x_var1 + x_var2+ x_var3`
    - where `y_var` is the outcome variable and `x_var`s are independent variables
  - `data`: data frame that contains variables named in `formula`
  - `subset`: "an optional vector specifying a subset of observations to be used in the fitting process."
  - `na.action`: "a function which indicates what should happen when the data contain NAs"
    - by default, an observation will be excluded from the model if itcontains missing values for any variable used in the model
- Value (object created or "returned" by the `lm()` function)
  - "lm returns an object of class 'lm' "
  - "An object of class 'lm' is a list containing at least the following components"

Run model of the relationship between reading test score (X) and math test score (Y)

- However, the output printed just by running the `lm()` function usually doesn't contain everything we need

```
lm(formula = bytxmstd ~ bytxrstd, data = els)
#>
#> Call:
#> lm(formula = bytxmstd ~ bytxrstd, data = els)
#>
#> Coefficients:
#> (Intercept)     bytxrstd
#>       7.592        0.850
```

When you run the `lm()` function, *R* creates an object that contains lots of information about the model we run. We can store this object by assigning it a name using `<-`

- let's store the object

```
# create object
mod1 <- lm(formula = bytxmstd ~ bytxrstd, data = els)

# simple print of mod1 object
mod1
#>
#> Call:
#> lm(formula = bytxmstd ~ bytxrstd, data = els)
#>
#> Coefficients:
#> (Intercept)     bytxrstd
#>       7.592        0.850
```

Let's investigate the contents of the `mod1` object using the `str()` function

- Below output looks rather nasty! let me highlight a few points:
- the underlying data type of `mod1` is a "list"; data frames are also list
- `mod1` is a list of 12 elements
  - each of these 12 elements has a "name", so `mod1` is a "named list"
- we can refer to elements the object `mod1` using the syntax: `object_name$element_name`
  - this is the same as when we referred to variables in a data frame using the syntax: `dataframe_name$variable_name`!
- The first element of `mod1` is named "coefficients", so we could refer to this element by typing `mod1$coefficients`:
  - 7.592331, 0.8500363

```
# investigate contents of mod1 object using str() function
mod1 %>% str()
#> List of 12
#>  $ coefficients : Named num [1:2] 7.59 0.85
#>   ..- attr(*, "names")= chr [1:2] "(Intercept)" "bytxrstd"
#>  $ residuals    : dbl+lbl [1:16197]  -6.085,   1.861,   4.054,  -4.301,   4.476,   2.924.
#>    ..@ label       : chr "Math test standardized score"
#>    ..@ format.stata: chr "%12.0g"
#>    ..@ labels      : Named num -8
```

```
#>    .. ..- attr(*, "names")= chr "Survey component legitimate skip/NA"
#>  $ effects      : dbl+lbl [1:16197] -6313.051, -1364.813,     4.084,    -4.252,     4.54.
#>    ..@ label       : chr "Math test standardized score"
#>    ..@ format.stata: chr "%12.0g"
#>    ..@ labels      : Named num -8
#>    .. ..- attr(*, "names")= chr "Survey component legitimate skip/NA"
#>  $ rank         : int 2
#>  $ fitted.values: Named num [1:16197] 58.2 55.8 62.4 49 36.1 ...
#>   ..- attr(*, "names")= chr [1:16197] "1" "2" "3" "4" ...
#>  $ assign       : int [1:2] 0 1
#>  $ qr           :List of 5
#>   ..$ qr   : num [1:16197, 1:2] -1.27e+02 7.86e-03 7.86e-03 7.86e-03 7.86e-03 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   .. .. ..$ : chr [1:16197] "1" "2" "3" "4" ...
#>   .. .. ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   .. ..- attr(*, "assign")= int [1:2] 0 1
#>   ..$ qraux: num [1:2] 1.01 1
#>   ..$ pivot: int [1:2] 1 2
#>   ..$ tol  : num 1e-07
#>   ..$ rank : int 2
#>   ..- attr(*, "class")= chr "qr"
#>  $ df.residual  : int 16195
#>  $ xlevels      : Named list()
#>  $ call         : language lm(formula = bytxmstd ~ bytxrstd, data = els)
#>  $ terms        :Classes 'terms', 'formula'  language bytxmstd ~ bytxrstd
#>   .. ..- attr(*, "variables")= language list(bytxmstd, bytxrstd)
#>   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:2] "bytxmstd" "bytxrstd"
#>   .. .. .. ..$ : chr "bytxrstd"
#>   .. ..- attr(*, "term.labels")= chr "bytxrstd"
#>   .. ..- attr(*, "order")= int 1
#>   .. ..- attr(*, "intercept")= int 1
#>   .. ..- attr(*, "response")= int 1
#>   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
#>   .. ..- attr(*, "predvars")= language list(bytxmstd, bytxrstd)
#>   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
#>   .. .. ..- attr(*, "names")= chr [1:2] "bytxmstd" "bytxrstd"
#>  $ model        :'data.frame':   16197 obs. of  2 variables:
#>   ..$ bytxmstd: dbl+lbl [1:16197] 52.1, 57.6, 66.4, 44.7, 40.6, 35.0, 50.7, 66.2, 3...
#>    .. ..@ label       : chr "Math test standardized score"
#>    .. ..@ format.stata: chr "%12.0g"
```

```
#>    .. ..@ labels      : Named num -8
#>    .. .. ..- attr(*, "names")= chr "Survey component legitimate skip/NA"
#>   ..$ bytxrstd: dbl+lbl [1:16197] 59.5, 56.7, 64.5, 48.7, 33.5, 28.9, 40.8, 68.3, 4...
#>    .. ..@ label       : chr "Reading test standardized score"
#>    .. ..@ format.stata: chr "%12.0g"
#>    .. ..@ labels      : Named num -8
#>    .. .. ..- attr(*, "names")= chr "Survey component legitimate skip/NA"
#>   ..- attr(*, "terms")=Classes 'terms', 'formula'  language bytxmstd ~ bytxrstd
#>   .. .. ..- attr(*, "variables")= language list(bytxmstd, bytxrstd)
#>   .. .. ..- attr(*, "factors")= int [1:2, 1] 0 1
#>   .. .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. .. ..$ : chr [1:2] "bytxmstd" "bytxrstd"
#>   .. .. .. .. ..$ : chr "bytxrstd"
#>   .. .. ..- attr(*, "term.labels")= chr "bytxrstd"
#>   .. .. ..- attr(*, "order")= int 1
#>   .. .. ..- attr(*, "intercept")= int 1
#>   .. .. ..- attr(*, "response")= int 1
#>   .. .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
#>   .. .. ..- attr(*, "predvars")= language list(bytxmstd, bytxrstd)
#>   .. .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
#>   .. .. .. ..- attr(*, "names")= chr [1:2] "bytxmstd" "bytxrstd"
#>  - attr(*, "class")= chr "lm"
```

Let's play around with the element named `coefficients` within the object `mod1` using the syntax `object_name$element_name`

```
# names of elements in object mod1
names(mod1)
#>  [1] "coefficients"  "residuals"     "effects"       "rank"
#>  [5] "fitted.values" "assign"        "qr"            "df.residual"
#>  [9] "xlevels"       "call"          "terms"         "model"

# mod1$coefficients

  #print element
    mod1$coefficients
#> (Intercept)    bytxrstd
#>   7.5923310   0.8500363

  #investigate structure of element
    # a (named) numeric vector
    mod1$coefficients %>% str()
```

```
#>  Named num [1:2] 7.59 0.85
#>  - attr(*, "names")= chr [1:2] "(Intercept)" "bytxrstd"
    str(mod1$coefficients) # same same
#>  Named num [1:2] 7.59 0.85
#>  - attr(*, "names")= chr [1:2] "(Intercept)" "bytxrstd"

  # names of element
    mod1$coefficients %>% names()
#> [1] "(Intercept)" "bytxrstd"
    names(mod1$coefficients) # same same
#> [1] "(Intercept)" "bytxrstd"

  # length of element
    mod1$coefficients %>% length()
#> [1] 2
    length(mod1$coefficients) # same same
#> [1] 2

# so the mod1$coefficients is an object itself; it is a numeric vector of length=2

  # print the first element of mod1$coefficients
  mod1$coefficients[1]
#> (Intercept)
#>    7.592331

  # print the second element of mod1$coefficients
  mod1$coefficients[2]
#>  bytxrstd
#> 0.8500363
```

A key takeaway from above code chunk

- Population linear regression model
  - $Y_i = \beta_0 + \beta_1 X_i + u_i$
- Our estimate of β_0 can be found by typing `mod1$coefficients[1]`
  - 7.592331
- Our estimate of β_1 can be found by typing `mod1$coefficients[2]`
  - 0.8500363

`summary()` function

Use the `summary()` after use running `lm()` to produce easier-to-read summary of regression results

- Description
  - "summary is a generic function used to produce result summaries of the results of various model fitting functions. The function invokes particular methods which depend on the class of the first argumen"
- syntax
  - `summary(object, ...)`
  - examples (these give the same result)
    - `summary(object = mod1)`
    - `summary(mod1)`
- Value (object created or "returned" by the `summary()` function)
  - object returned by the summary function is different depending on the kind of model you are summarizing

Run `summary()` function after running regression using `lm()`

```
mod1 <- lm(formula = bytxmstd ~ bytxrstd, data = els)

# priting output from summary(mod1)
  #summary(object = mod1)
  summary(mod1)
#>
#> Call:
#> lm(formula = bytxmstd ~ bytxrstd, data = els)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -26.703  -4.434  -0.071   4.144  39.084
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 7.592331   0.213320   35.59   <2e-16 ***
#> bytxrstd    0.850036   0.004182  203.26   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
```

```
#> Residual standard error: 6.715 on 16195 degrees of freedom
#> Multiple R-squared:  0.7184, Adjusted R-squared:  0.7184
#> F-statistic: 4.131e+04 on 1 and 16195 DF,  p-value: < 2.2e-16
```

Using `str()` to investigate object created by `summary()` function

- again, the Below output looks rather nasty! let me highlight a few points:
- the underlying data type of the object created by `summary(mod1)` is a "list";
  - data frames are also list; `mod1` is also a list
- `summary(mod1)` is a list of 11 elements
  - each of these 12 elements has a "name", so `summary(mod1)` is a "named list"
- we can refer to elements the object `summary(mod1)` using the syntax: `object_name$element_name`
  - e.g., `summary(mod1)$coefficients`

```
# investigating object created by summary(mod1)
summary(mod1) %>% str()
#> List of 11
#>  $ call         : language lm(formula = bytxmstd ~ bytxrstd, data = els)
#>  $ terms        :Classes 'terms', 'formula'  language bytxmstd ~ bytxrstd
#>   .. ..- attr(*, "variables")= language list(bytxmstd, bytxrstd)
#>   .. ..- attr(*, "factors")= int [1:2, 1] 0 1
#>   .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. ..$ : chr [1:2] "bytxmstd" "bytxrstd"
#>   .. .. .. ..$ : chr "bytxrstd"
#>   .. ..- attr(*, "term.labels")= chr "bytxrstd"
#>   .. ..- attr(*, "order")= int 1
#>   .. ..- attr(*, "intercept")= int 1
#>   .. ..- attr(*, "response")= int 1
#>   .. ..- attr(*, ".Environment")=<environment: R_GlobalEnv>
#>   .. ..- attr(*, "predvars")= language list(bytxmstd, bytxrstd)
#>   .. ..- attr(*, "dataClasses")= Named chr [1:2] "numeric" "numeric"
#>   .. .. ..- attr(*, "names")= chr [1:2] "bytxmstd" "bytxrstd"
#>  $ residuals    : dbl+lbl [1:16197]  -6.085,   1.861,   4.054,  -4.301,   4.476,   2.924.
#>    ..@ label       : chr "Math test standardized score"
#>    ..@ format.stata: chr "%12.0g"
#>    ..@ labels      : Named num -8
#>    .. ..- attr(*, "names")= chr "Survey component legitimate skip/NA"
#>  $ coefficients : num [1:2, 1:4] 7.59233 0.85004 0.21332 0.00418 35.59134 ...
#>   ..- attr(*, "dimnames")=List of 2
```

```
#>   .. ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   .. ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
#>  $ aliased      : Named logi [1:2] FALSE FALSE
#>   ..- attr(*, "names")= chr [1:2] "(Intercept)" "bytxrstd"
#>  $ sigma        : num 6.71
#>  $ df           : int [1:3] 2 16195 2
#>  $ r.squared    : num 0.718
#>  $ adj.r.squared: num 0.718
#>  $ fstatistic   : Named num [1:3] 41315 1 16195
#>   ..- attr(*, "names")= chr [1:3] "value" "numdf" "dendf"
#>  $ cov.unscaled : num [1:2, 1:2] 1.01e-03 -1.92e-05 -1.92e-05 3.88e-07
#>   ..- attr(*, "dimnames")=List of 2
#>   .. ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   .. ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>  - attr(*, "class")= chr "summary.lm"

summary(mod1)$coefficients
#>              Estimate  Std. Error   t value      Pr(>|t|)
#> (Intercept) 7.5923310 0.213319614  35.59134 3.345643e-267
#> bytxrstd    0.8500363 0.004182012 203.26012  0.000000e+00

summary(mod1)$coefficients %>% str()
#>  num [1:2, 1:4] 7.59233 0.85004 0.21332 0.00418 35.59134 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"
```

Grabbing individual elements from `summary(mod1)$coefficients`

- the element "coefficients" created by summary() is a "matrix"
- more specifically, it is a 2 X 4 matrix, that is a matrix with 2 rows and 4 columns
- we can grab an individual cell within the matrix using this syntax: `summary(mod1)$coefficients[<row_nu`
  - e.g., type `summary(mod1)$coefficients[2,1]` to print estimate of β_1
  - like this: 0.8500363

```
# the element "coefficients" created by summary() is a "matrix"
summary(mod1)$coefficients %>% class()
#> [1] "matrix" "array"

# more specifically, it is a 2 X 4 matrix, that is a matrix with 2 rows and 4 columns
summary(mod1)$coefficients %>% str()
```

```
#>  num [1:2, 1:4] 7.59233 0.85004 0.21332 0.00418 35.59134 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"

# we can grab an individual cell within the matrix using this syntax:
  # summary(mod1)$coefficients[<row_num>,<col_num]
  summary(mod1)$coefficients[2,1] # this is the estimate for Beta_1
#> [1] 0.8500363
```

Estimation

General things we do in regression analysis

1. **Estimation**
    - How do we choose estimates of β_0 and β_1 using sample data?
2. **Prediction**
    - What is the predicted value of Y for someone with a particular value of X?
3. **Hypothesis testing** [focus of the rest of the semester]
    - Hypothesis testing and confidence intervals about β_1

Step 1 of regression is to estimate parameters of the population linear regression model

- $Y_i = \beta_0 + \beta_1 X_i + u_i$

Goal of estimation: use sample data to estimate population parameters:

- Previous lecture: we used sample data on variable Y to estimate population mean, μ_Y
    - $\bar{Y}$ (sample mean of Y) is an estimate of μ_Y
- This lecture: use sample data to estimate the population intercept, β_0, and the population regression coefficient, β_1
    - $\hat{\beta_0}$ is an estimate of β_0
    - $\hat{\beta_1}$ is an estimate of β_1

Review of terms

- *point estimate (also called "estimate")*:
    - a single value that is our guess of the value of the population parameter

  - e.g., sample mean institution-level student debt $\bar{Y} = 50$ is our best guess of population mean math test score μ_Y

- *estimator*:
  - a method or formula for calculating an estimate of a population parameter based on sample data
  - e.g., $\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$ is estimator (formula) for calculating an estimate of population mean, μ_Y

- *estimation*:
  - the process of using an estimator to calculate point estimates of population parameters, based on sample data

Estimation problem:

- we write out the population linear regression model, $Y_i = \beta_0 + \beta_1 X_i + u_i$
- Need to develop a method for choosing values of $\widehat{\beta}_0$ and $\widehat{\beta}_1$
  - that is we need to develop an estimator that yields the "best" point estimates of $\widehat{\beta}_0$ and $\widehat{\beta}_1$

Estimation (population mean)

We faced a similar estimation problem when we were trying to estimate population mean, μ_Y!

- Using sample data, we want to calculate the "best" estimate of the population mean, μ_Y
- We decided sample mean, $\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$, was the "best" estimate

How did statisticians decide that $\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$ was "best" estimate of μ_Y? What criteria to make this decision?:

Imagine that m is all potential estimates for μ_Y

- e.g., $\bar{Y}$ is one potential estimate of μ_Y; the sample median is another potential estimate

Criteria statisticians used to decide which m is the "best" estimate of μ_Y:

- choose the value, m , that minimizes the "sum of squares"
- Sum of squares = $\sum_{i=1}^{n}(Y_i - m)^2$
- Sum of squares in words
  - for each observation: measures how far away an individual observation Y_i is from estimate m; and then square this distance (to remove negative values)

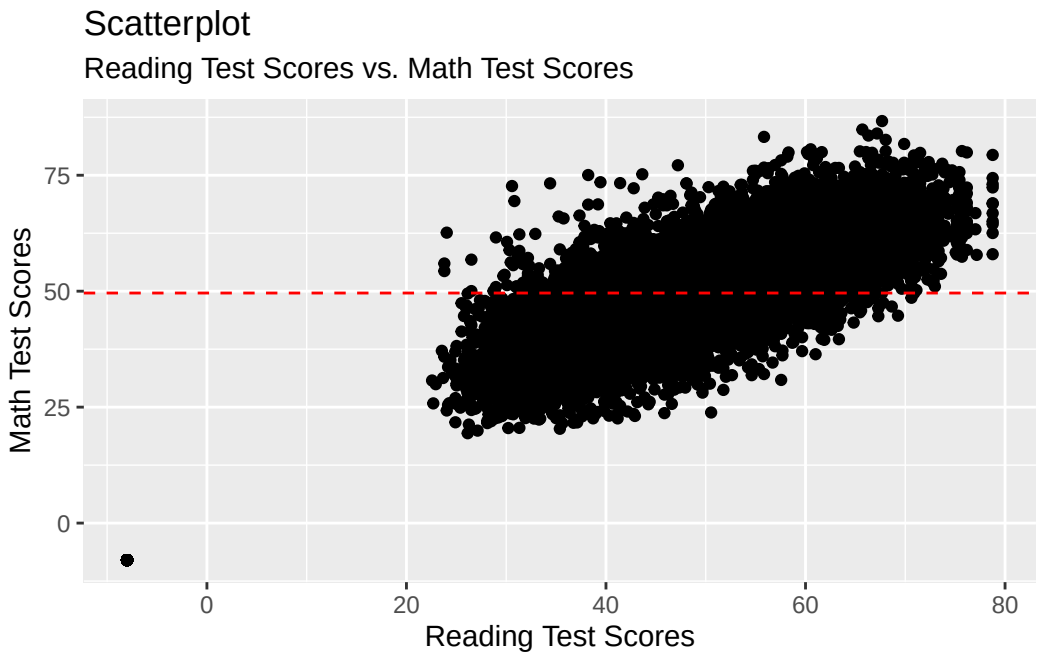
  - then sum this squared distance across all observations, Y_i

- Conceptually, in choosing "minimize sum of squares" as our criteria for selecting the best estimate of μ_Y, we are "how far off" in total (across all observations) if we can only use a single value to be our guess of the value of each individual observation

Below, we graph scatterplot of reading test score (X) and math test score (Y axis), and add a horizontal line for the mean value of math test score, $\bar{Y} = 50$

- let's say that $\bar{Y} = 50$ is our "predicted value" for each observation in the scatterplot
- For each observation i, the "residual" $u_i = Y_i - \bar{Y}$ represents how far off our predicted value is from the actual value for that observation
- $\bar{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$ is the choice for m that minimizes the sum of squared errors, $\sum_{i=1}^{n}(Y_i - m)^2$
- in other words, $\bar{Y}$ is the value that minimizes how far off our prediction is in total (across all observations)

```
els %>% ggplot(aes(x = bytxrstd, y = bytxmstd)) +
  geom_point() +
  geom_hline(yintercept=mean(els$bytxmstd, na.rm = TRUE), linetype="dashed", color = "red") +
    labs(subtitle="Reading Test Scores vs. Math Test Scores",
       y="Math Test Scores",
       x="Reading Test Scores",
       title="Scatterplot")
```



Regression in R

Population linear regression model

- $Y_i = \beta_0 + \beta_1 X_i + u_i$

The estimation problem in regression

- Need to develop method for selecting the "best" estimate of $\hat{\beta}_0$ and $\hat{\beta}_1$
- Solution: do the same thing we did for population mean! Choose the estimates $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize the sum of squares

First some terminology:

- Y_i is the actual observed value of Y for observation i
- $\hat{\beta}_0$ is an estimate of β_0, population average value of Y when $X = 0$
- $\hat{\beta}_1$ is an estimate of β_1, the average change in the value of Y associated with a one-unit increase in X
- $\hat{Y}_i$ is the predicted value of Y_i, based on sample data!
  - $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$
  - Example: $\hat{\beta}_0 = 10,000$ and $\hat{\beta}_1 = .75$, then for a university with cost of attendance $= 30,000$
    - $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i = 10,000 + .75 \times 30,000 = 32,500$
- Estimated residual, $\hat{u}_i$ is the difference between actual Y_i and predicted $\hat{Y}_i$
  - $Y_i - \hat{Y}_i = \hat{u}_i$
  - $Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i) = \hat{u}_i$
  - Residuals are often called "errors"

Criteria for choosing "best" estimate of $\hat{\beta}_0$ and $\hat{\beta}_1$

- Select values that minimize "sum of squared residuals"
- that is, the "best" estimates of $\hat{\beta}_0$ and $\hat{\beta}_1$ are those that any other alternatives would result in a higher sum of squared residuals

Sum of squared residuals (or sometimes called "sum of squared errors"):

- $\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$
- $\sum_{i=1}^{n} (Y_i - (\hat{\beta}_0 + \hat{\beta}_1 X_i))^2$
- $\sum_{i=1}^{n} (u_i)^2$

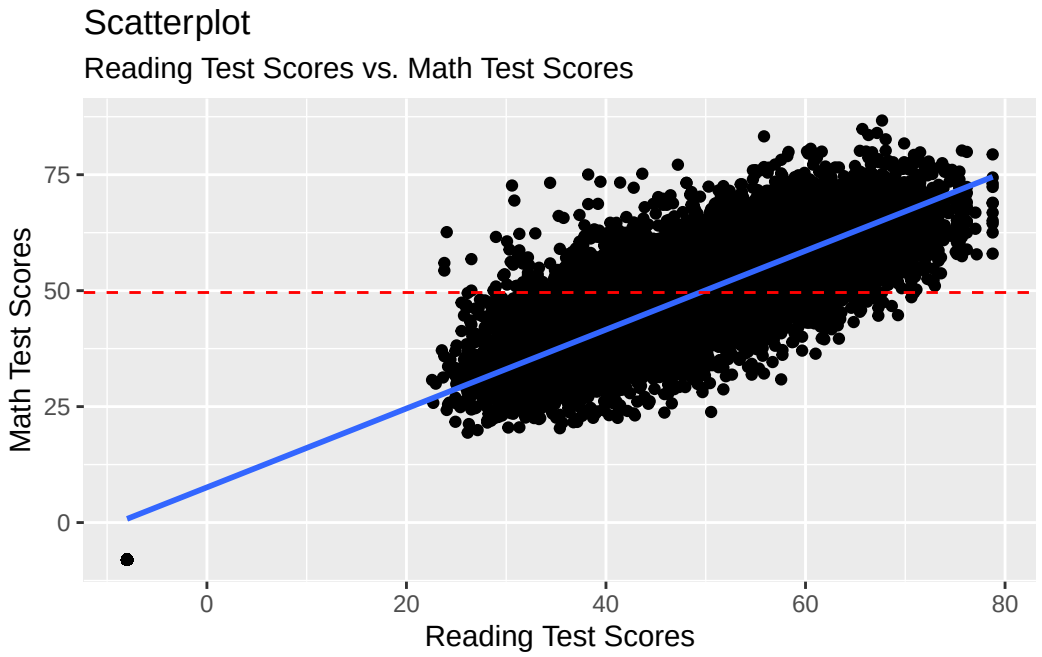
Conceptual explanation for use "sum of squared residuals" as the criteria for choosing values for $\hat{\beta}_0$ and $\hat{\beta}_1$

- Having a y-intercept ($\hat{\beta}_0$) and a slope ($\hat{\beta}_1$) allows us to draw a straight line, our "prediction line"
- by choosing the values of $\hat{\beta}_0$ and $\hat{\beta}_1$ that minimize the sum of squared residuals, we are creating a linear prediction line that minimizes how far away (in total, across all observations) the predicted values are from the actual values

Scatterplot of reading test score (X) and math test score (Y)

- red dotted horizontal line for mean of institution-level student debt
- blue line is the OLS prediction line, associated with values of $\hat{\beta}_0$ and $\hat{\beta}_1$ calculated by R, we will cover more in later weeks
- visually you can see that sum of squared errors would be much larger if we used the mean of institution-level student debt (red line) as the predicted value for every observation

```
els %>% ggplot(aes(x = bytxrstd, y = bytxmstd)) + geom_point() +
  stat_smooth(method = 'lm') +
  geom_hline(yintercept=mean(els$bytxmstd, na.rm = TRUE), linetype="dashed", color = "red") +
    labs(subtitle="Reading Test Scores vs. Math Test Scores",
       y="Math Test Scores",
       x="Reading Test Scores",
       title="Scatterplot")
```



Prediction

RQ: What is the effect of reading test score (X) on math test score (Y)?

Population Linear Regression Model

- $Y_i = \beta_0 + \beta_1 X_i + u_i$

OLS Prediction Line or "OLS Regression Line" (based on sample data)

- $\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$

Let's run regression in R using `lm()` to calculate values for point estimates $\hat{\beta}_0$ and $\hat{\beta}_1$

- $\hat{\beta}_0 = 7.59$
- $\hat{\beta}_1 = 0.85$

```
mod1 <- lm(formula = bytxmstd ~ bytxrstd, data = els)

summary(mod1)
#>
#> Call:
#> lm(formula = bytxmstd ~ bytxrstd, data = els)
#>
#> Residuals:
#>     Min      1Q  Median      3Q     Max
#> -26.703  -4.434  -0.071   4.144  39.084
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept) 7.592331   0.213320   35.59   <2e-16 ***
#> bytxrstd    0.850036   0.004182  203.26   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 6.715 on 16195 degrees of freedom
#> Multiple R-squared:  0.7184, Adjusted R-squared:  0.7184
#> F-statistic: 4.131e+04 on 1 and 16195 DF,  p-value: < 2.2e-16

summary(mod1)$coefficients %>% class()
#> [1] "matrix" "array"

# more specifically, it is a 2 X 4 matrix, that is a matrix with 2 rows and 4 columns
summary(mod1)$coefficients %>% str()
```

```
#>  num [1:2, 1:4] 7.59233 0.85004 0.21332 0.00418 35.59134 ...
#>  - attr(*, "dimnames")=List of 2
#>   ..$ : chr [1:2] "(Intercept)" "bytxrstd"
#>   ..$ : chr [1:4] "Estimate" "Std. Error" "t value" "Pr(>|t|)"

# we can grab an individual cell within the matrix using this syntax:
  # summary(mod1)$coefficients[<row_num>,<col_num]
  summary(mod1)$coefficients[1,1] # row 1, column 1; this is the estimate for Beta_0
#> [1] 7.592331
  summary(mod1)$coefficients[2,1] # row 2, column 1; this is the estimate for Beta_1
#> [1] 0.8500363
```

Writing out models

When we run a regression model, write out regression models like this (we will be doing this the rest of the quarter):

- Write out the population linear regression model
  - Label symbols; label what the variables X and Y actually represent
- OLS prediction line
  - write out OLS prediction line without estimate values
  - write out OLS prediction line with estimate values
- And usually a good idea to:
  - write out interpretation of$\widehat{\beta_0}$ in words
  - write out interpretation of$\widehat{\beta_1}$ in words
- **Population Linear Regression Model**
  - $Y_i = \beta_0 + \beta_1 X_i + u_i$
  - where:
    - subscript i refers to universities
    - Y_i: institution-level student debt (in dollars) at university i
    - X_i: annual cost of attendance (in dollars) for full-time resident graduate students at university i
    - β_0: population intercept, average value of Y when $X = 0$
    - β_1: population regression coefficient, the average change in the value of Y associated with a one-unit increase in X
  - **OLS Prediction Line or "OLS Regression Line" (based on sample data)**

- $\widehat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i$
- $\widehat{Y}_i = 7.59 + 0.85 \times X_i$

Once you write it out like this, easy to answer questions about your regression model

- Predict the the expected math test score for a student who has a reading test score of 40? That is, what is $E(\widehat{Y}_i|X = 40)$?
  - $E(\widehat{Y}_i|X = 40) = \widehat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i = 7.59 + 0.85 \times 40 = 42$
- Predict the the expected math test score for a student who has a reading test score of 70? That is, what is $E(\widehat{Y}_i|X = 70)$?
  - $E(\widehat{Y}_i|X = 70) = \widehat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 X_i = 7.59 + 0.85 \times 70 = 67$

Interpreting $\hat{\beta}_0$ and $\hat{\beta}_1$ in words

- $\hat{\beta}_0 = 7.59$ is the OLS estimate of the population intercept β_0, which represents average value of Y (math test score) for a student with X=0 (reading test score = 0)
  - interpretation of $\hat{\beta}_0 = 7.59$: the predicted math test score for a student with a reading test score of `0` is 7.59
- $\hat{\beta}_1 = 0.85$ is the OLS estimate of the population intercept β_1, which represents average effect of a one-unit increase in X on the value of Y
  - General formula for interpretation of $\hat{\beta}_1$:
    - "On average, a one-unit increase in X is associated with a $\hat{\beta}_1$ increase (or decrease) in the value of Y
    - when applying this formula replace "one unit increase in X" with "one test score increase in reading test score"; do the same sort of thing for Y
  - interpretation of $\hat{\beta}_1 = 0.85$: a test score increase in reading test score is associated with a 0.85 a test score increase in math test score