Seminar notebook

Anastasia Chanbour

08/10/2021

Loading the required packages

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.5     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(gapminder)
library(broom) #broom package takes the messy output of built-in functions in R, such as lm,#nls, or t.
theme_set(theme_minimal(base_size = 10))   #theme_set overides default ggplot theme
library(gapminder)
library(plotly)
```

```
##
## Attaching package: 'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
## The following object is masked from 'package:graphics':
##
##     layout
```

Linear regression

Simple linear regression

Estimation

Task 1. Filter year=2007 from the gapminder data and call the data frame object gm2007. 2. Create a regression object called model_lm that regresses lifeEXp on gdpPercap for 2007 data 3. Output the estimated coefficients, intercept and slope, only

```r
gm2007 <- gapminder %>% filter(year == 2007)
model_lm <- lm(data = gm2007, lifeExp ~ gdpPercap)
coef(model_lm)
```

```
##  (Intercept)    gdpPercap
## 5.956565e+01 6.371341e-04
```

Demo dplyr::summarise function

Summarise function displays the coefficient estimates in the regression model

- hat_beta1 = `cor(x,y) * sd(y) / sd(x)`
- hat_beta0 = mean(y) - hat_beta1 * mean(x)
- Regression line passes through `(mean(x), mean(y))`

```r
#output: correlation, sd, mean(x), mean(y), estimated intercept and slope coefficient using formula
gm2007 %>%
  summarize(cor_xy=cor(gdpPercap, lifeExp),
            sd_x = sd(gdpPercap),
            sd_y = sd(lifeExp),
            mean_x = mean(gdpPercap),
            mean_y = mean(lifeExp),
            hat_beta1 = cor_xy/sd_x*sd_y,
            hat_beta0 = mean_y - hat_beta1*mean_x)
```

```
## # A tibble: 1 x 7
##   cor_xy   sd_x  sd_y mean_x mean_y hat_beta1 hat_beta0
##    <dbl>  <dbl> <dbl>  <dbl>  <dbl>     <dbl>     <dbl>
## 1  0.679 12860.  12.1 11680.   67.0  0.000637      59.6
```

Inference

```r
#Use summary() to obtain basic output of regression
summary(model_lm)
```

```
##
## Call:
## lm(formula = lifeExp ~ gdpPercap, data = gm2007)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -22.828  -6.316   1.922   6.898  13.128
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 5.957e+01  1.010e+00   58.95   <2e-16 ***
## gdpPercap   6.371e-04  5.827e-05   10.93   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.899 on 140 degrees of freedom
## Multiple R-squared:  0.4606, Adjusted R-squared:  0.4567
## F-statistic: 119.5 on 1 and 140 DF,  p-value: < 2.2e-16
```

(ISLR eq 3.8) p.66

`se(beta hat) = sigma / sqrt(sum((x - mean(x))^2))`

Estimated by:

`se(beta hat) = sigma hat / sqrt(sum((x - mean(x))^2))`

where (ISLR 3.15) p.69:

`sigma hat = RSE = sqrt( RSS / (n-2) )`

```
#Using augment() to extract residuals, compute the estimated se of the slope parameter by
#obtaining 1. RSS 2. MSE = (RSS/(n-p)) 3. estimated S.E of hat beta1
augment(model_lm) %>%
  summarize(RSS = sum(.resid^2),
            MSE = RSS /(n()-2),
            SE = sqrt(MSE)/sqrt(sum((gdpPercap - mean(gdpPercap))^2)))
```

```
## # A tibble: 1 x 3
##      RSS   MSE        SE
##    <dbl> <dbl>     <dbl>
## 1 11086.  79.2 0.0000583
```

Model diagnostics

```
#glance: Construct a single row summary "glance" of a model, fit, or other object
#CODE
glance(model_lm)
```

```
## # 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     0.461         0.457  8.90      120. 1.69e-20     1  -511. 1028. 1037.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

"Portion of variance in outcome **explained** by simple linear regression model"

$$R^2 = \text{cor}(x, y)^2$$

```
#compute correlation squared between gdpPercap and lifeExp
#CODE
cor(gm2007$gdpPercap, gm2007$lifeExp)^2
```

```
## [1] 0.4605827
```

$$R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$

```
#using residuals from augment(), obtain values of RSS, TSS, and R2
#CODE
augment(model_lm) %>%
  summarize(RSS = sum(.resid^2),
            TSS = sum((lifeExp - mean(lifeExp))^2),
            R2 = 1 - RSS/TSS)
```

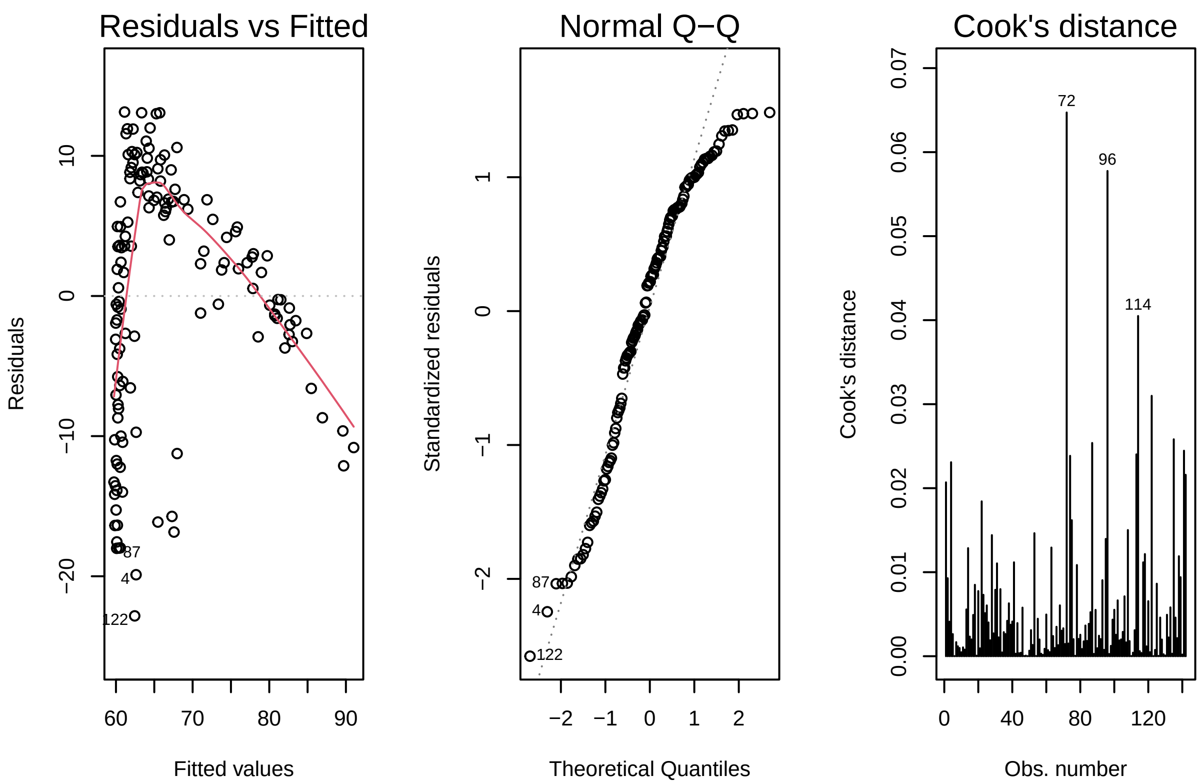
```
## # A tibble: 1 x 3
##      RSS    TSS    R2
##    <dbl>  <dbl> <dbl>
## 1 11086. 20552. 0.461
```

Diagnostic plots

Idea: look for patterns in residuals, which could indicate systematic errors (bias)

Outliers vs influential points: influential points affect the slope of the model line

```r
#Plot residuals vs gdpPercap, using residuals from augment()
#CODE
par(mfrow=c(1,3))
plot(model_lm, which = c(1,2,4)) #plot.lm is plot on a regression generates 6 plots
```



There is a pattern in residuals. Suggets that the relationship is non-linear

Other diagnostics:

- Checking for (approximate) normality with quantile-quantile plot
- Checking for influential observations

Cook's distance, `cooksd` in the plots, measures how much the predictions for all other observations change if we leave out one observation

Point with high `cooksd` values