PSET 04 - Linear regression, Poisson Regression

S&DS 361

2024-02-21

Part 1: Data acquisition and cleaning

1. Add comments to `get.flight.location.data.r` The R script `get.flight.location.data.r` contains code that scrapes flight location data from ADSBexchange.com. It is pasted below. The script contains no comments (except at the very top of the script). Please add comments to the code below everywhere there is a `##` explaining what that chunk of code does.

You don't need to write paragraphs, just a line or two for each one. Many of the functions below will look familiar, but some may be new to you and will require you to check R help or the interwebs. Feel free to run the code (by, for example, copying it into it's own R script and running line by line) to check what the outputs are before writing your comment. You'll need to create a folder called `data/flight-locations` in order for the last line of the code to run. I suggest changing the for loop to say `j in 1:2` instead of `j in 1:nrow(du)` so that you can test the code without having to wait for all the data to download.

```
## get.flight.location.data.r
## scrape data from adsbexchange
## Got URL from the May example under readsb-hist that was give here
## https://www.adsbexchange.com/data-samples/
library(tidyverse)
library(jsonlite)

## create a data frame with three columns, where 'sec' goes from 0 to 55 in increments of 5, 'min' goes
du = expand.grid(sec=seq(0,55, by=5),
                 min=0:59,
                 hr=0:23)

## select the columns 'hr', 'min', and 'sec' from the data frame and print theb first 6 rows
du = du %>%
  select(hr, min, sec)
head(du)

## pad the values in 'hr', 'min', and 'sec' with a 0 if they are less than 2 characters long
du = du %>%
  mutate(hr  = str_pad(hr , 2, pad='0'),
         min = str_pad(min, 2, pad='0'),
         sec = str_pad(sec, 2, pad='0'))

## add a column named 'url' to the data frame that contains strings concatenated by the url, the values
du = du %>%
  mutate(url = paste0('https://samples.adsbexchange.com/readsb-hist/2022/12/01/',
                      hr, min, sec,
                      'Z.json.gz'))
head(du)
```

```
## Select the rows where the values in 'min' are multiples of 10 and the values in 'sec' are 00.
du = du %>%
  filter(as.numeric(min) %% 10 ==0,
         sec=='00')

## We are going to get the data from the web directed by the urls in the 'url' column. We need to loop
for(j in 1:nrow(du)){

  cat(j, '')

  ## create a temp.url object that is the url in the 'url' column, and read the json file from the url
  temp.url = du$url[j]
  temp.url = url(temp.url)
  con = gzcon(temp.url)
  d = fromJSON(con)

  ## select the 'aircraft' column from the data frame we crawled from the url
  d = d$aircraft

  ## find the time part of the url by regular expression and save it using 'filename'. Then save the da
  filename = gsub('.+/', '', du$url[j])
  filename = gsub('[.].+', '', filename)
  filename = paste0('data/flight-locations/', filename, '.rds')
  saveRDS(d, file=filename)

}
```

2. Add comments to `clean.flight.location.data.r` The R script `clean.flight.location.data.r` contains code that clean the data obtained by `get.flight.location.data.r`. The script is pasted below. It also contains no comments. Please add comments everywhere there is a `##` explaining what that chunk of code does.

You don't need to write paragraphs, just a line or two for each one. Many of the functions below will look familiar, but some may be new to you and will require you to check R help or the interwebs. Feel free to run the code (by, for example, copying it into it's own R script) to check what the outputs are before writing your comment.

```
## clean.flight.location.data.r
## clean the flight location data obtained by using get.flight.location.data.r

## extract all files from the directory 'data/flight-locations/' and save their names to 'filenames' ob
filenames = dir('data/flight-locations/', full.names = T)
head(filenames)

## create a data frame with no rows and columns named 'df'
df = NULL

## loop over all the filenames in 'filenames'
for(filename in filenames){

  ## extract the time part of the filename and save it to 'time'. Then extract the hour and minute from
  time = gsub('.+[/]|Z.+', '', filename)
  hr = substr(time, 1, 2)
  min= substr(time, 3, 4)
```

```
## paste the values in 'hr' and 'min' together and save it to 'time'. Then convert 'hr' and 'min' to n
time = paste0(hr, ":", min)
hr = as.numeric(hr)
min= as.numeric(min)

cat(time, '')

d = readRDS(filename)

## Manipulate 'd' and store the new data frame in 'dd'
dd = d %>%

  ## unnest the lastPosition column by '.' and create different columns for each element of the list
  unnest(lastPosition, names_sep = '.') %>%

  mutate(

    ## remove white spaces from strings in fight columna
    flight = trimws(flight),

    ## substitute NA's in order to fill in missing values for lat and lon
    lat = ifelse(!is.na(lat), lat, rr_lat),
    lon = ifelse(!is.na(lon), lon, rr_lon),
    lat = ifelse(is.na(lat) & lastPosition.seen_pos<=600, lastPosition.lat, lat),
    lon = ifelse(is.na(lon) & lastPosition.seen_pos<=600, lastPosition.lon, lon),

    ## subsitute 'ground' with 0's in order to make the column all numeric and convert it to numeric v
    alt_baro = ifelse(alt_baro=='ground', 0       , alt_baro),
    alt_baro = as.numeric(alt_baro),
    alt      = ifelse(!is.na(alt_geom)  , alt_geom, alt_baro),

    ## create a new column `speed` that is gs if it is not an NA, otherwise it is tas
    speed   = ifelse(!is.na(gs), gs, tas),

    ## create a new column `heading` that is track if it is not an NA, if `track is NA and `true_headi
    heading = case_when(
      !is.na(track)                                          ~ track,
      is.na(track) & !is.na(true_heading)                    ~ true_heading,
      is.na(track) &  is.na(true_heading) & !is.na(nav_heading) ~ nav_heading,
      TRUE ~ track),

    ## create new 'hr', 'min' and 'time' columns using the extracted values from the filename
    hr  =  hr,
    min = min,
    time = time) %>%

  ## select the columns 'hex', 'type', 'flight', 'lat', 'lon', 'alt', 'speed', 'heading', 'hr', 'min'
  select(hex, type, flight, lat, lon, alt, speed, heading, hr, min, time) %>%

  ## filter out rows where 'alt' is 0
  filter(alt!=0)

## append 'dd' to 'df'
df = rbind(df, dd)
```

```
}

saveRDS(df, file='data/flight.locations.rds')
```

Part 2: Linear regression

3. Different codings for categorical variables Suppose you are predicting `log(value)` in our `branford.csv` data set. Suppose you build one model with `log(living)`, `grade`, and `style` as predictors, and for a second model you use `relevel` to choose `Mobile Home` as the reference level for the `style` variable, instead of using the default reference `Bungalow` that `lm` uses in the first model. Show that

- the predictions obtained from the two models are the same
- the CIs and PIs are the same
- adjusted R^2 is the same
- the intercept differs by a constant
- the coefficients for `style` all differ by the same constant
- the coefficients of `Mobile Home` in model 1 and `Bungalow` in model 2 are that same constant or -1*constant.
- all other coefficients (`log(living)` and `grade`) and standard errors are the same (up to 13 decimal places)

```
d = read.csv('data/branford.csv')
head(d)
```

```
##     pid  value year land living beds                          address
## 1     1 247400 1988 0.51   2194    3   66 SUNNY MEADOW RD Branford, CT
## 2   100 177200 1900 1.30   1200    3 516 LEETES ISLAND RD Branford, CT
## 3 10000 243600 1948 1.00   2473    4         30 NEWTON RD Branford, CT
## 4 10001 206100 1966 0.26   1503    3       29 MARSHALL RD Branford, CT
## 5 10003 189200 1967 0.27   1708    3       33 MARSHALL RD Branford, CT
## 6 10008 195600 1992 4.31   1291    2       17 MARSHALL PL Branford, CT
##   buildings good        style       model grade baths halfbaths heatfuel
## 1         1   87  Split-Level Residential   B -     2         1      Oil
## 2         1   78    Old Style Residential     C     1         1      Oil
## 3         1   78     Cape Cod Residential   B -     3         0      Gas
## 4         1   75     Cape Cod Residential   B -     2         0      Oil
## 5         1   75 Raised Ranch Residential     C     2         1      Oil
## 6         1   80 Raised Ranch Residential     C     1         0      Oil
##         heattype      ac nghd stories saleprice   saledate    roof bath_style
## 1      Hot Water Central   70    1.50    342000 08/30/2010 Asphalt     Modern
## 2      Hot Water    None   80    2.00         0 10/30/2009 Asphalt    Average
## 3      Hot Water    None   72    1.75         0 12/28/1992 Asphalt    Average
## 4      Hot Water Central   78    1.50    350000 11/06/2020 Asphalt     Modern
## 5      Hot Water Central   78    1.00         0 10/13/1989 Asphalt    Average
## 6 Forced Air-Duc Central   78    1.00    264000 07/23/2018 Asphalt    Average
##   kitchen_style      long      lat miles_to_coastline miles_to_highway
## 1 Above Average -72.85060 41.27678          0.6500547     0.0020418409
## 2       Average -72.74921 41.27477          0.4848638     0.0007012297
## 3       Average -72.80105 41.26494          0.2517704     0.0350288679
## 4 Above Average -72.79967 41.27061          0.4116531     0.0607908920
## 5       Average -72.79957 41.27048          0.4216232     0.0710992080
## 6       Average -72.79947 41.27010          0.4438308     0.0850672854
##   miles_to_school miles_to_hospital
## 1       0.9302452        0.03150048
```

```
## 2       2.6642665        1.44276458
## 3       1.0888478        0.33963807
## 4       0.8292984        0.72626866
## 5       0.8392923        0.72014148
## 6       0.8614687        0.69832493
```

```
lm1 = lm(log(value) ~ log(living) + grade + style, data=d)
dd = d %>%
  mutate(style = relevel(factor(style), ref = "Mobile Home"))
lm2 = lm(log(value) ~ log(living) + grade + style, data=dd)
summary(lm1)
```

```
##
## Call:
## lm(formula = log(value) ~ log(living) + grade + style, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.12318 -0.13948 -0.04088  0.07517  2.31507
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       10.93725    0.12963  84.371  < 2e-16 ***
## log(living)        0.30470    0.01593  19.122  < 2e-16 ***
## gradeA +           0.18706    0.05653   3.309 0.000944 ***
## gradeA ++          0.12199    0.11030   1.106 0.268749
## gradeB            -0.39185    0.03020 -12.977  < 2e-16 ***
## gradeB -          -0.58536    0.03018 -19.396  < 2e-16 ***
## gradeB +          -0.13935    0.03201  -4.353 1.37e-05 ***
## gradeC            -0.89387    0.03265 -27.375  < 2e-16 ***
## gradeC -          -0.92141    0.04458 -20.670  < 2e-16 ***
## gradeC +          -0.74823    0.03126 -23.933  < 2e-16 ***
## gradeD            -0.93702    0.26309  -3.562 0.000372 ***
## styleCape Cod     -0.13068    0.02848  -4.589 4.58e-06 ***
## styleColonial     -0.12242    0.02913  -4.202 2.69e-05 ***
## styleCottage      -0.45369    0.03710 -12.228  < 2e-16 ***
## styleCustom        0.47905    0.03528  13.580  < 2e-16 ***
## styleMobile Home  -2.51305    0.03425 -73.367  < 2e-16 ***
## styleOld Style    -0.03201    0.02822  -1.134 0.256716
## styleRaised Ranch -0.23733    0.03044  -7.797 7.76e-15 ***
## styleSplit-Level  -0.19746    0.03115  -6.339 2.53e-10 ***
## styleTudor        -0.29902    0.26277  -1.138 0.255202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.261 on 4773 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8644
## F-statistic:  1608 on 19 and 4773 DF,  p-value: < 2.2e-16
```

```
summary(lm2)
```

```
##
## Call:
## lm(formula = log(value) ~ log(living) + grade + style, data = dd)
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.12318 -0.13948 -0.04088  0.07517  2.31507
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        8.42420    0.12303  68.472  < 2e-16 ***
## log(living)        0.30470    0.01593  19.122  < 2e-16 ***
## gradeA +           0.18706    0.05653   3.309 0.000944 ***
## gradeA ++          0.12199    0.11030   1.106 0.268749
## gradeB            -0.39185    0.03020 -12.977  < 2e-16 ***
## gradeB -          -0.58536    0.03018 -19.396  < 2e-16 ***
## gradeB +          -0.13935    0.03201  -4.353 1.37e-05 ***
## gradeC            -0.89387    0.03265 -27.375  < 2e-16 ***
## gradeC -          -0.92141    0.04458 -20.670  < 2e-16 ***
## gradeC +          -0.74823    0.03126 -23.933  < 2e-16 ***
## gradeD            -0.93702    0.26309  -3.562 0.000372 ***
## styleBungalow      2.51305    0.03425  73.367  < 2e-16 ***
## styleCape Cod      2.38237    0.02714  87.796  < 2e-16 ***
## styleColonial      2.39063    0.02826  84.588  < 2e-16 ***
## styleCottage       2.05936    0.03320  62.031  < 2e-16 ***
## styleCustom        2.99210    0.03541  84.490  < 2e-16 ***
## styleOld Style     2.48104    0.02680  92.585  < 2e-16 ***
## styleRaised Ranch  2.27572    0.02950  77.141  < 2e-16 ***
## styleSplit-Level   2.31559    0.03018  76.724  < 2e-16 ***
## styleTudor         2.21403    0.26266   8.429  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.261 on 4773 degrees of freedom
## Multiple R-squared:  0.8649, Adjusted R-squared:  0.8644
## F-statistic:  1608 on 19 and 4773 DF,  p-value: < 2.2e-16
```

From the summary, we immediately get the R^2 values of the two models are the same. The next trunk of code shows that the log(living) and all grade coefficients of the two models are the same up to a very small number. Their standard error is also same up to a small number. The intercept is differed by a constant.

```
for (i in 1:11) {
  print(paste("lm1:", names(lm1$coefficients)[i], "lm2: ", names(lm2$coefficients)[i]))
  print(as.numeric(lm1$coefficients[i] - lm2$coefficients[i]))
  print(sqrt(diag(vcov(lm1)))[i] - sqrt(diag(vcov(lm2)))[i])
}
```

```
## [1] "lm1: (Intercept) lm2:  (Intercept)"
## [1] 2.513049
## (Intercept)
## 0.006601881
## [1] "lm1: log(living) lm2:  log(living)"
## [1] 1.032507e-14
##   log(living)
## -3.469447e-17
## [1] "lm1: gradeA + lm2:  gradeA +"
## [1] 6.994405e-15
##      gradeA +
## -2.081668e-17
```

```
## [1] "lm1: gradeA ++ lm2:  gradeA ++"
## [1] 6.800116e-15
##     gradeA ++
## -4.163336e-17
## [1] "lm1: gradeB lm2:  gradeB"
## [1] -7.271961e-15
##        gradeB
## -6.938894e-18
## [1] "lm1: gradeB - lm2:  gradeB -"
## [1] -8.881784e-15
## gradeB -
##        0
## [1] "lm1: gradeB + lm2:  gradeB +"
## [1] -1.482148e-14
## gradeB +
##        0
## [1] "lm1: gradeC lm2:  gradeC"
## [1] -3.219647e-15
## gradeC
##      0
## [1] "lm1: gradeC - lm2:  gradeC -"
## [1] -3.330669e-16
##      gradeC -
## -6.938894e-18
## [1] "lm1: gradeC + lm2:  gradeC +"
## [1] -3.663736e-15
##      gradeC +
## -6.938894e-18
## [1] "lm1: gradeD lm2:  gradeD"
## [1] -1.210143e-14
##        gradeD
## -1.110223e-16
```

The next trunk of code shows that the coefficients for style all differ by the same constant. The coefficients of Mobile Home in model 1 and Bungalow in model 2 are -1*constant.

```
for (i in 12:19) {
  if(i < 16)
    j = i + 1
  else {
    i = i + 1
    j = i
  }
  print(paste("lm1:", names(lm1$coefficients)[i], "lm2", names(lm2$coefficients)[j]))
  print(as.numeric(lm1$coefficients[i] - lm2$coefficients[j]))
}
```

```
## [1] "lm1: styleCape Cod lm2 styleCape Cod"
## [1] -2.513049
## [1] "lm1: styleColonial lm2 styleColonial"
## [1] -2.513049
## [1] "lm1: styleCottage lm2 styleCottage"
## [1] -2.513049
## [1] "lm1: styleCustom lm2 styleCustom"
## [1] -2.513049
```

```
## [1] "lm1: styleOld Style lm2 styleOld Style"
## [1] -2.513049
## [1] "lm1: styleRaised Ranch lm2 styleRaised Ranch"
## [1] -2.513049
## [1] "lm1: styleSplit-Level lm2 styleSplit-Level"
## [1] -2.513049
## [1] "lm1: styleTudor lm2 styleTudor"
## [1] -2.513049
```

```
print('lm1 mobile home / lm2 bungalow')
```

```
## [1] "lm1 mobile home / lm2 bungalow"
```

```
print(as.numeric(lm1$coefficients[16] / lm2$coefficients[12]))
```

```
## [1] -1
```

We finally check the predictions, CIs and PIs are the same. We've finished all sanity checks.

```
library(MLmetrics, warn.conflicts = FALSE)
new_data <- data.frame(value = sample(min(d$value):max(d$value), 10),
                       living = sample(min(d$living):max(d$living), 10),
                       grade = sample(d$grade, 10),
                       style = sample(d$style, 10))
error = MSE(predict(lm1, new_data), predict(lm2, new_data))
cat('MSE difference: ', error)
```

```
## MSE difference:  8.519698e-29
```

```
lm1.pred <- predict(lm1, newdata = new_data, interval = "prediction")
lm2.pred <- predict(lm2, newdata = new_data, interval = "prediction")
print('PI difference: ')
```

```
## [1] "PI difference: "
```

```
print(lm2.pred - lm1.pred)
```

```
##               fit           lwr           upr
## 1  -2.309264e-14 -2.309264e-14 -2.309264e-14
## 2  -7.105427e-15 -7.105427e-15 -7.105427e-15
## 3  -3.552714e-15 -3.552714e-15 -3.552714e-15
## 4  -8.881784e-15 -8.881784e-15 -8.881784e-15
## 5  -1.243450e-14 -1.421085e-14 -1.065814e-14
## 6   0.000000e+00  0.000000e+00  0.000000e+00
## 7  -1.776357e-15 -1.776357e-15 -1.776357e-15
## 8   1.776357e-15  1.776357e-15  1.776357e-15
## 9  -3.552714e-15 -3.552714e-15 -3.552714e-15
## 10 -1.776357e-15 -1.776357e-15 -1.776357e-15
```

```
lm1.conf <- predict(lm1, newdata = new_data, interval = "confidence")
lm2.conf <- predict(lm2, newdata = new_data, interval = "confidence")
print('CI difference: ')
```

```
## [1] "CI difference: "
```

```
print(lm2.conf - lm1.conf)
```

```
##               fit           lwr           upr
## 1  -2.309264e-14 -2.309264e-14 -2.309264e-14
## 2  -7.105427e-15 -7.105427e-15 -7.105427e-15
```

```
## 3  -3.552714e-15 -3.552714e-15 -3.552714e-15
## 4  -8.881784e-15 -8.881784e-15 -8.881784e-15
## 5  -1.243450e-14 -1.243450e-14 -1.243450e-14
## 6   0.000000e+00  0.000000e+00  0.000000e+00
## 7  -1.776357e-15 -1.776357e-15 -1.776357e-15
## 8   1.776357e-15  1.776357e-15  1.776357e-15
## 9  -3.552714e-15 -3.552714e-15 -3.552714e-15
## 10 -1.776357e-15 -1.776357e-15 -1.776357e-15
```

4. Simulation Suppose y has a linear, non-deterministic relationship with x that can be represented by the model $y = 1 + 2x + \epsilon$, where $\epsilon \sim N(0, 1)$. Simulate some data by doing the following:

a. Create a random sample of 10,000 x values on $[0, 1]$.
b. Create a random sample of 10,000 ϵ values from $N(0, 1)$.
c. Create a random sample of 10,000 y values using $y = 1 + 2x + \epsilon$.

We first create the data in the following trunk.

```
## Data creation
set.seed(1)
x = runif(10000, 0, 1)
epsilon = rnorm(10000, 0, 1)
y = 1 + 2*x + epsilon
```

Then,

d. Fit a simple linear regression model to the data and report the estimated coefficients and estimated model standard deviation. Are the estimates what you would expect? Why or why not?

```
## Fit a simple linear regression model
lm = lm(y ~ x)
summary(lm)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2844 -0.6697 -0.0133  0.6640  3.7449
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.98049    0.01967   49.86   <2e-16 ***
## x            2.00706    0.03398   59.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.989 on 9998 degrees of freedom
## Multiple R-squared:  0.2586, Adjusted R-squared:  0.2586
## F-statistic:  3488 on 1 and 9998 DF,  p-value: < 2.2e-16
```

```
print(sigma(lm))
```

```
## [1] 0.9890432
```

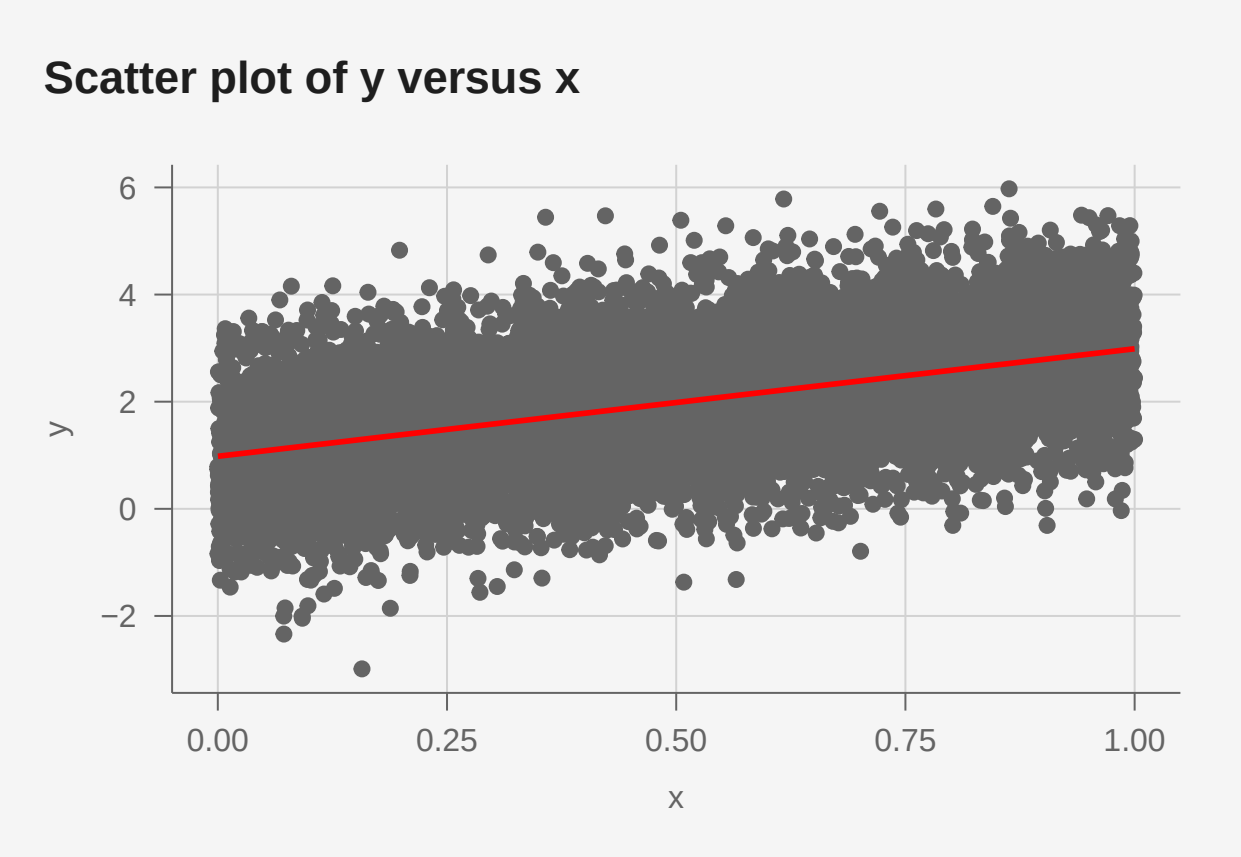
From the summary, we can see that the estimated intercept is 0.98049 and the estimated x-slope is 2.00706. The estimated model standard deviation is 0.9890432. The estimates are what we would expect. The true

model is $y = 1 + 2x + \epsilon$, where the variance of ϵ is 1, and the estimates are very close to the true values.

e. Plot the scatter plot of y versus x, and add a regression line to the plot. Does this visualization look as you would have expected? Do the linear regression assumptions appear to hold? Why or why not?

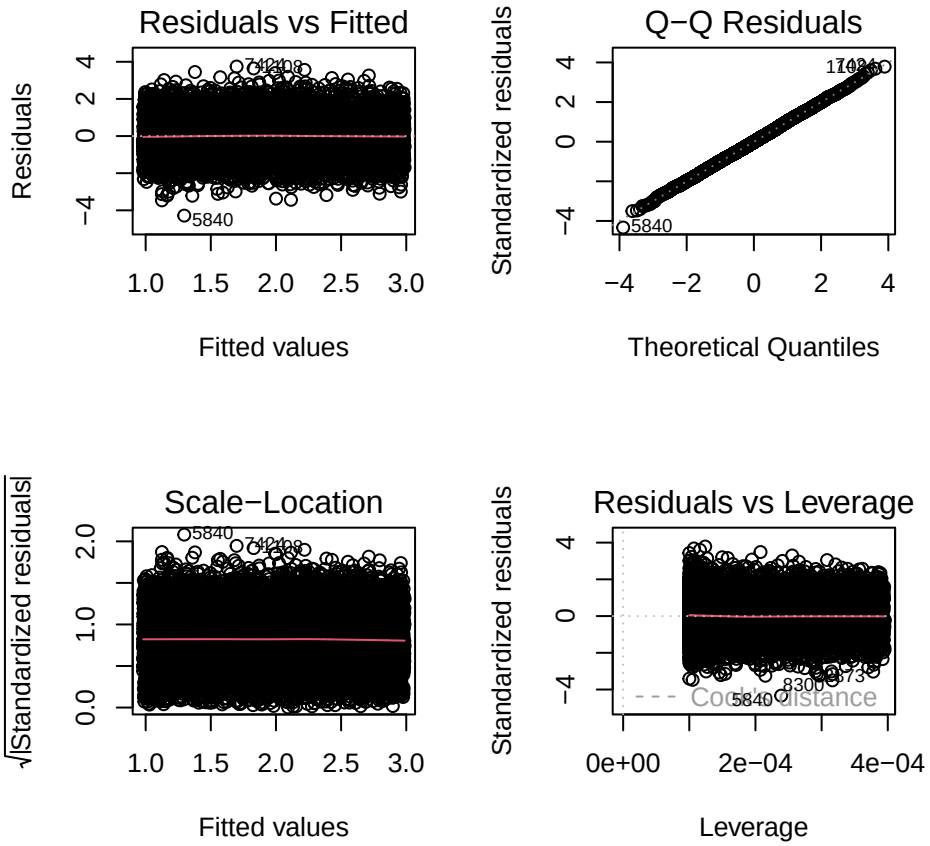
```
## Scatter plot of y versus x
ggplot(data.frame(x, y), aes(x, y)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "red") +
  ggtitle("Scatter plot of y versus x") +
  theme_pub()
```

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



The scatter plot of y versus x looks as we would have expected. We will verify the linear model assumptions in the next trunk.

```
## Check linear model assumptions
par(mfrow = c(2, 2))
plot(lm)
```



I confirmed that the linear assumptions are met because: 1. x and y tend to have a linear relationship2. the residuals are normally distributed from the Q-Q plot 3. The residuals are homoscedastic from the scale-location plot 4. The residuals are independent from the residuals versus leverage plot.

Part 3: Poisson Regression

ChatGPT

5. After asking ChatGPT "What is Poisson Regression?" I then asked "why couldn't I use linear regression instead?" Part of ChatGPT's response is below. For each bullet point, name at least one part of ChatGPT's response that isn't *quite* accurate.

- "Linear regression assumes that the response variable is continuous and normally distributed, and it is not suitable for modeling count data that is typically discrete and non-negative"

Linear regression model can be applied to non-negative data. And the normally distribution assumption is for the residuals, not the response variable.

- "Furthermore, Poisson regression models the natural logarithm of the mean of the response variable as a linear function of the predictor variables, which can handle situations where the mean and variance of the response variable are not equal."

The property of Poisson distribution is that the mean and variance are equal. For situations that mean and variance of the response variable are not equal, we should not use Poisson regression.

Tesla's Safety Score

Tesla uses data collected on each driver (using their cameras and possibly other sensors) to create a Safety Score for that driver. According to Tesla's website, "The Safety Score (Beta) is an assessment of your driving behavior based on several metrics called Safety Factors. These are combined to estimate the likelihood that

your driving could result in a future collision... The Safety Score (Beta) is intended to provide drivers transparency and feedback of their driving behaviors to encourage safer driving and potentially pay less for their insurance. The Safety Score is a value between 0 and 100, where a higher score indicates safer driving. Most drivers are expected to have a Safety Score of 80 or above."

Tesla publishes the formulas they use to compute the Safety Score. "In order to calculate your daily Safety Score, we use the Predicted Collision Frequency (PCF) formula below to predict how many collisions may occur per 1 million miles driven, based on your driving behaviors measured by your Tesla vehicle."

$$\text{Predicted Collision Frequency (PCF)} = 0.83220180 \times 1.012555104^{\text{Forward Collision Warnings per 1,000 Non-Autopilot Miles}} \times 1.1646082$$

The Safety Factors like "Hard Breaking" are described on the page linked above. They convert the PCF to the Safety Score using

$$\text{Safety Score} = 112.29263237 - 14.77121589 \cdot \text{PCF}$$

6. The PCF formula looks like it could be the result of fitting a Poisson regression model to their Safety Factors and collisions data. Discuss why a Poisson Regression model would or would not be a reasonable model for this data. Poisson regression model would be a reasonable model for this data because the response variable is the number of collisions, which is a count data. The car accidents can be interpreted as an event with a constant rate of occurance but random in time. The Poisson distribution is a good model for this kind of data.

7. Let's assume the PCF formula did in fact come from a Poisson Regression model. What was $\beta_{\text{Hard Breaking}}$, the coefficient for Hard Breaking, in that model? The coefficient for Hard Breaking in the Poisson Regression model is $\beta_{\text{Hard Breaking}} = \log(1.16460827) = 0.152$.

8. What is the interpretation of the term $1.16460827^{\text{Hard Braking}}$ in the PCF formula? The term $1.16460827^{\text{Hard Braking}}$ in the PCF formula is the multiplicative effect of a one unit increase in the Hard Braking on the predicted collision frequency. It means that for a one unit increase in the Hard Braking, the predicted collision frequency is multiplied by 1.16460827. In other words, one unit increase of Hard Braking will lead to $\log(1.164) = 0.152$ increase in the logarithm of predicted collision frequency.

EV charging stations

The following data set is census data set from earlier in the course with new columns added that have the number of Level 2 (`lev2`) and Level 3 (`lev3`) electric vehicle charging stations for each census tract (`GEOID`).

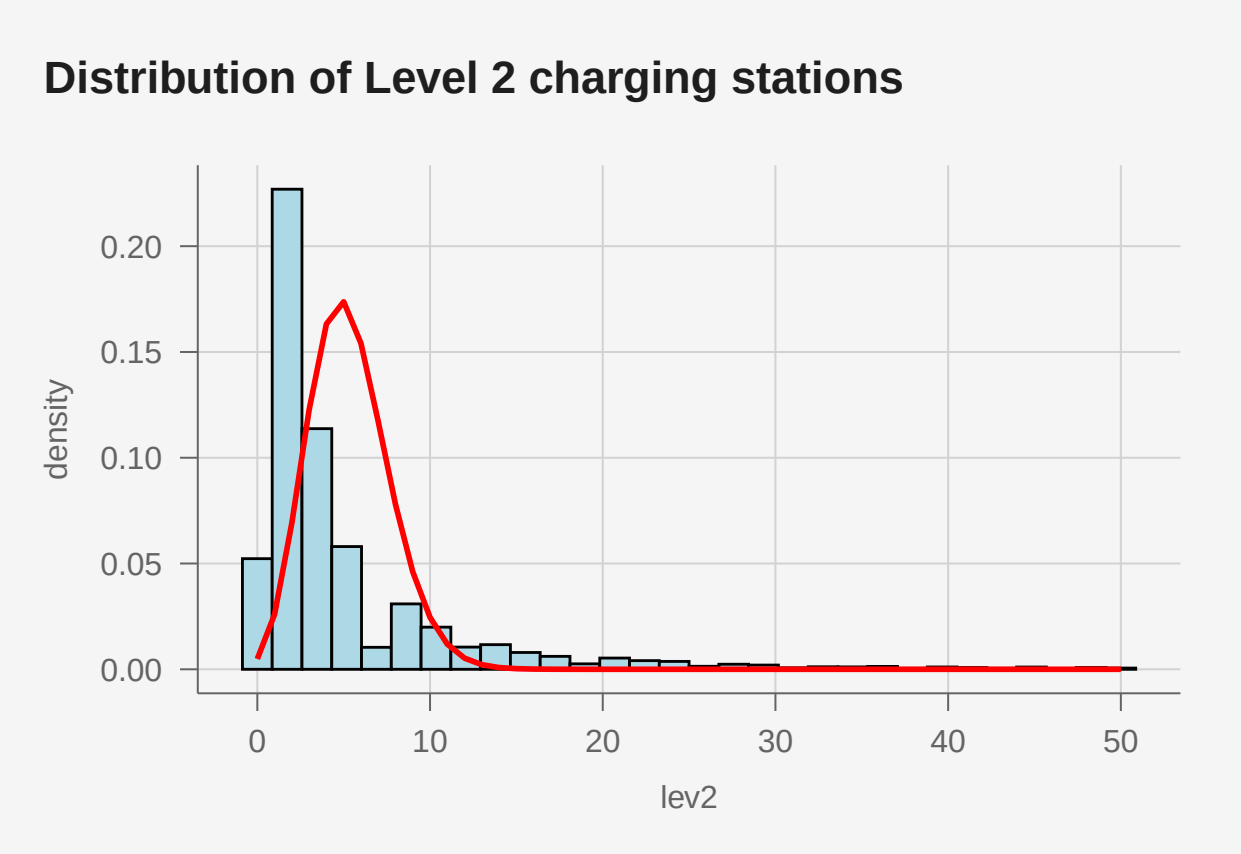
```
dc = readRDS('data/tracts.and.census.with.EV.stations.rds')
dc = dc@data ## keep just the data frame, not the polygons
dc = dc %>%

  ## Change NAs to 0s, and
  ## since charging stations seem to come in pairs, create new columns
  ## for pairs of charging stations
  mutate(lev2 = ifelse(is.na(lev2), 0, lev2),
         lev3 = ifelse(is.na(lev3), 0, lev3),
         lev2pairs = round(lev2/2),
         lev3pairs = round(lev3/2)) %>%

  ## Keep tracts with at least one charging station,
  ## and get rid of a couple of outliers
  filter((lev2!=0 | lev3!=0) & lev2 <= 50)
```

```r
# find the mean
poisson_x = min(dc$lev2):max(dc$lev2)
poisson_y = dpois(poisson_x, mean(dc$lev2))
p = data.frame(x = poisson_x, y = poisson_y)
dc %>%
  ggplot(aes(lev2)) +
  geom_histogram(aes(y=..density..), fill = "lightblue", color = "black") +
  geom_line(data = p, aes(x, y), color = "red") +
  ggtitle("Distribution of Level 2 charging stations") +
  theme_pub()
```

9. Suppose we want to quickly determine if the number of Level 2 charging stations in a census track looks roughly Poisson. (For this problem, consider all the data at once, not just the data for smaller subsets of x.) Let's compare the data to a Poisson distribution with the same mean as the data. Plot the actual distribution of the data along with this Poisson distribution in a way that allows for a quick comparison of the two. Does the number pairs of Level 2 charging stations in a census tract look Poisson distributed?

```
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
## i Please use `after_stat(density)` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The data looks roughly Poisson distributed, but the peak is left skewed than the theoretical Poisson

distribution. The number of pairs of Level 2 charging stations in a census tract looks Poisson distributed. We might use a proper mean parameter to model this.

```
## Fit a Poisson Regression model
m1 = glm(lev2 ~ house.value + age, data=dc, family=poisson)
summary(m1)
```

10. Fit a Poisson Regression model for predicting Level 2 charging stations in a census tract using `house.value` and `age`. How does this model compare to the null model (the model with only an intercept term)? Considering your observations in #9, what outputs of this model are reasonable/unreasonable?

```
##
## Call:
## glm(formula = lev2 ~ house.value + age, family = poisson, data = dc)
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.934e+00  1.667e-02  116.05   <2e-16 ***
## house.value  7.120e-07  8.887e-09   80.11   <2e-16 ***
## age         -1.389e-02  4.245e-04  -32.71   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 104687  on 17507  degrees of freedom
## Residual deviance:  98841  on 17505  degrees of freedom
##   (518 observations deleted due to missingness)
## AIC: 149761
##
## Number of Fisher Scoring iterations: 5
```

The residual deviance is much smaller than the null deviance, which indicates that the Poisson model is better than the null model. Note that all factors in the model is significant with a very low P-value, hence we are confident that we can conclude strong statistical evidence from our model. With the result from #9, we can conclude that the Poisson model is reasonable for the data if we choose house.value and age as our predictor.