Linear Regression

Lutao DAI

Aug 20, 2019

Contents

Setups

R offers multiple packages for performing data analysis. Apart from providing an excellent interface for statistical analysis, the next best thing about R is the endless support it gets from developers and data science maestros from all over the world. The current count of downloadable packages from CRAN stands close to 7000 packages!

To do analyses in R, usually the first thing is to load some packages for their handy functions. For instance, `ggplot2` and `dplyr`, introduced in the bootcamp, are two common packages. R allows two ways of package loading: `library(package)` and `require(package)`. While `library` and `require` are almost interchangeable, they behave quite differently when tending to load a package that is not previously installed.

If you use `library(uninstalled.package)`, the program will throw an error and halt immediately. On the contrary, `require(uninstalled.package)` will give you a warning. The warning is harmless if the `uninstalled.package` is actually redundant in your code. If not, the program will carry on and crash when you try to call a function from that package. This makes troubleshooting more difficult, because the error only tells you the problem happens at, say line 500, but the actual mistake occurs at line 4 where you try to load the package. Therefore, usually `library` is preferred.

`require`, however, has one favorable feature that it returns a logical value. It returns `TRUE` if the package is installed and `FALSE` when the package is not. This property can aid programming in some situation, such as the `if` statement below.

```
library(ggplot2)
library(ggExtra)
library(dplyr)

if(!require('car')){
  install.packages('car')
  library('car')
}

if(!require('GGally')){
  install.packages('GGally')
  library('GGally')
}
```

Alternatively, you can let `pacman` to handle the package import for you. It the package is not yet installed, it will install and load the package for you.

```
require(pacman)
pacman::p_load(tidyverse, car, leaps)
```

The working directory of the R Markdown file by default is the directory of the file. The input/output (I/O) path, if not absolute, is relative to the working directory. Sometimes the input and output paths are different, even remote, from the default path. Under this scenario, it is desirable to change the working directory.

In the console, changing and showing the current working directory are as simple as `setwd(new.path)` and `getwd()` respectively. However, this two functions only change the working directory locally to the R chunk that executes them. To apply the change to all chunks, one needs the `opts_knit` function from the package `knitr`.

```
knitr::opts_knit$set(root.dir = 'path/to/your/data/file')
```

Please change the path accordingly to the folder with all the required data files.

Loading packages and setting the working directory are standard set up procedures in almost all R project. Most likely one would like to hide the set up chunks after knitting it. Do you know how to do it?

Simple Regression

This section uses the diamond dataset and we are developing a linear model that predicts the price of a diamond by its weight. First, let's load the dataset by `read_csv` and examine the dataset by `str`.

```
diamond <- read.csv('diamond_rings.dat')
str(diamond)
```

```
## 'data.frame':    48 obs. of  2 variables:
##  $ Weight..carats.          : num  0.17 0.16 0.17 0.18 0.25 0.16 0.15 0.19 0.21 0.15 ...
##  $ Price..Singapore.dollars.: int  355 328 350 325 642 342 322 485 483 323 ...
```

The dataset contains 48 observations ($n = 48$) and 2 features ($p = 2$). The names of features are too cumbersome, so let's rename the two columns as `Weight` and `Price`.

```
names(diamond) <-  c('Weight', 'Price')
```

Fitting a linear model in R is very simple. `lm` is used to fit a linear regression model. The first argument specifies the dependent variable (before the $\sim$) and independent variable(s) (after the $\sim$). R allows two ways of specifying the dataframe where variables are from: either put the dataframe as the second argument, or use '$' to indicate the affiliation.

```
model1 <- lm(Price ~ Weight, diamond) # or model1 <- lm(diamond$Price ~ diamond$Weight)
```

Some Handy Functions

This section defines three functions, whose outputs are organized in a way to mimic JMP outputs as you can see on your lecture notes. (I think JMP presents the analysis results in a more organized way, so it is good to have those functions around.) You could skip this section for now and come back to read it later if you are interested. I hope you would find the three functions helpful in your assignments and the group project.

Function 1: `par_table()`

```
par_table <- function(model1) {
  "Input: a linear model
   Output: a data frame summerises parameter estimates"

  out <- summary(model1)
  parameter <-  as.data.frame(out$coefficients)
  ci <- as.data.frame(confint.lm(model1))
  return(cbind(parameter, ci))
}
```

```
par_table(model1)
```

```
##               Estimate Std. Error   t value     Pr(>|t|)    2.5 %    97.5 %
## (Intercept) -259.6259   17.31886 -14.99094 2.523271e-19 -294.487 -224.7649
## Weight      3721.0249   81.78588  45.49715 6.751260e-40 3556.398 3885.6513
```

Function 2: `fit_table()`

```
fit_table <- function(model1, dataset.y) {
  "Input: a linear model and the numerical response of the data set
```

```
   Output: a data frame evaluates fitness of the model "

  out <- summary(model1)

  "Calculate variables"
  RSS <- sum((model1$res)^2)
  MSE <- RSS / (model1$df) #degree of freedom
  RMSE <- out$sigma
  Mean.response <- mean(dataset.y)
  count <- length(dataset.y)

  "Return Summary of Fit"
  return(t(data.frame(RSquare = out$r.squared,
                      RSquare.adj = out$adj.r.squared,
                      Root.Mean.Square.Error = RMSE,
                      Mean.of.Response = Mean.response,
                      Observations = count)))
}
```

```
fit_table(model1, diamond$Price)
```

```
##                               [,1]
## RSquare                  0.9782608
## RSquare.adj              0.9777882
## Root.Mean.Square.Error  31.8405223
## Mean.of.Response       500.0833333
## Observations            48.0000000
```

Function 3: `t_table()`

```
t_table <- function(t_test) {
  "Input: object returned by t.test()
   Output: A data frame summerises t-test statistics"

  return(data.frame(
    t.ratio = t_test$statistic,
    DF = t_test$parameter,
    'prob>|t|' = t_test$p.value,
    upper.CL.Diff = t_test$conf.int[1],
    lower.CL.Diff = t_test$conf.int[2],
    Difference = t_test$estimate[1] - t_test$estimate[2]
  ))
}
```

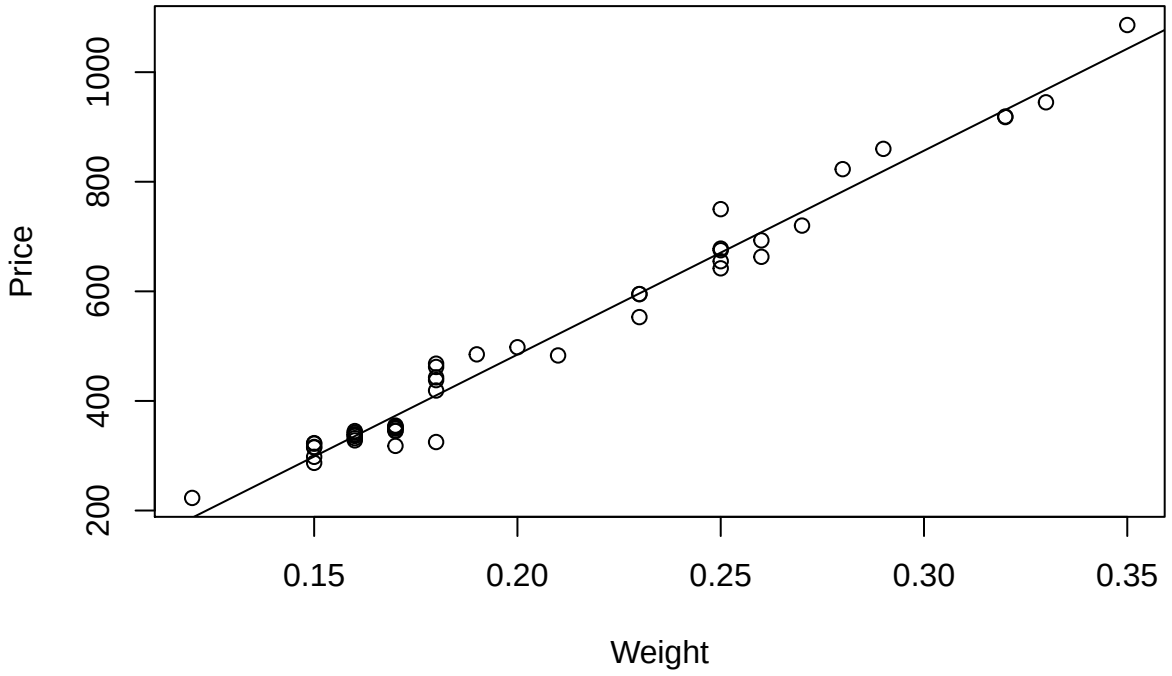
Basic Plots

Base R

The most basic plot for the linear regression is the scatter plot with regression lines. This can be easily done in both base R and `ggplot2`. Generally, `ggplot2` is a more popular choice because it gives much nicer plots. The following sections will focus on `ggplot2` as well.

In base R, `plot(x, y)` is used to generate a scatter plot, and `abline` can be used to add straight lines, which are defined by the intercept and the slope. Espetially, use `abline(v = x_0)` for vertical lines intercept x-axis at $(x_0, 0)$, and use `abline(h = y_0)` for horizontal lines intercept y-axis at $(0, y_0)$.

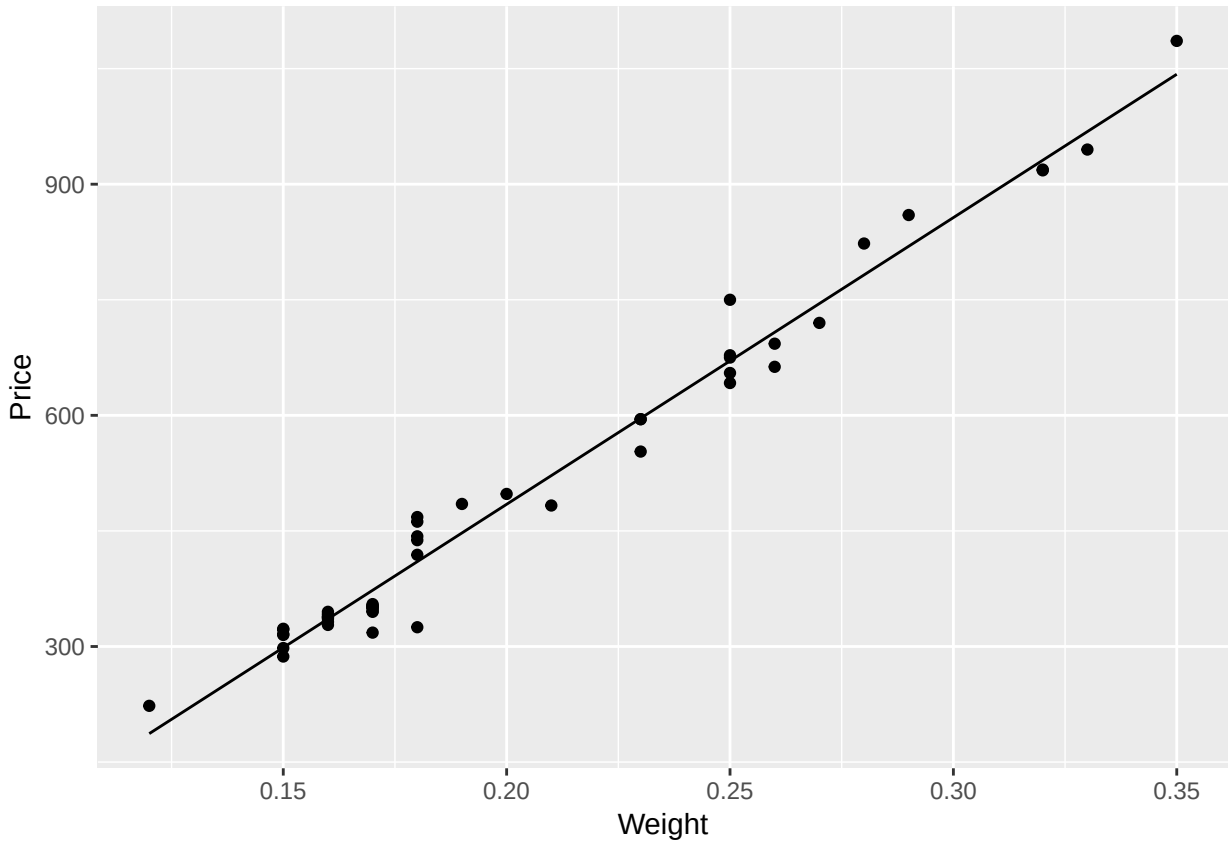
```
plot(diamond$Weight, diamond$Price, xlab = "Weight", ylab = "Price")
abline(coef(model1))
```



ggplot2

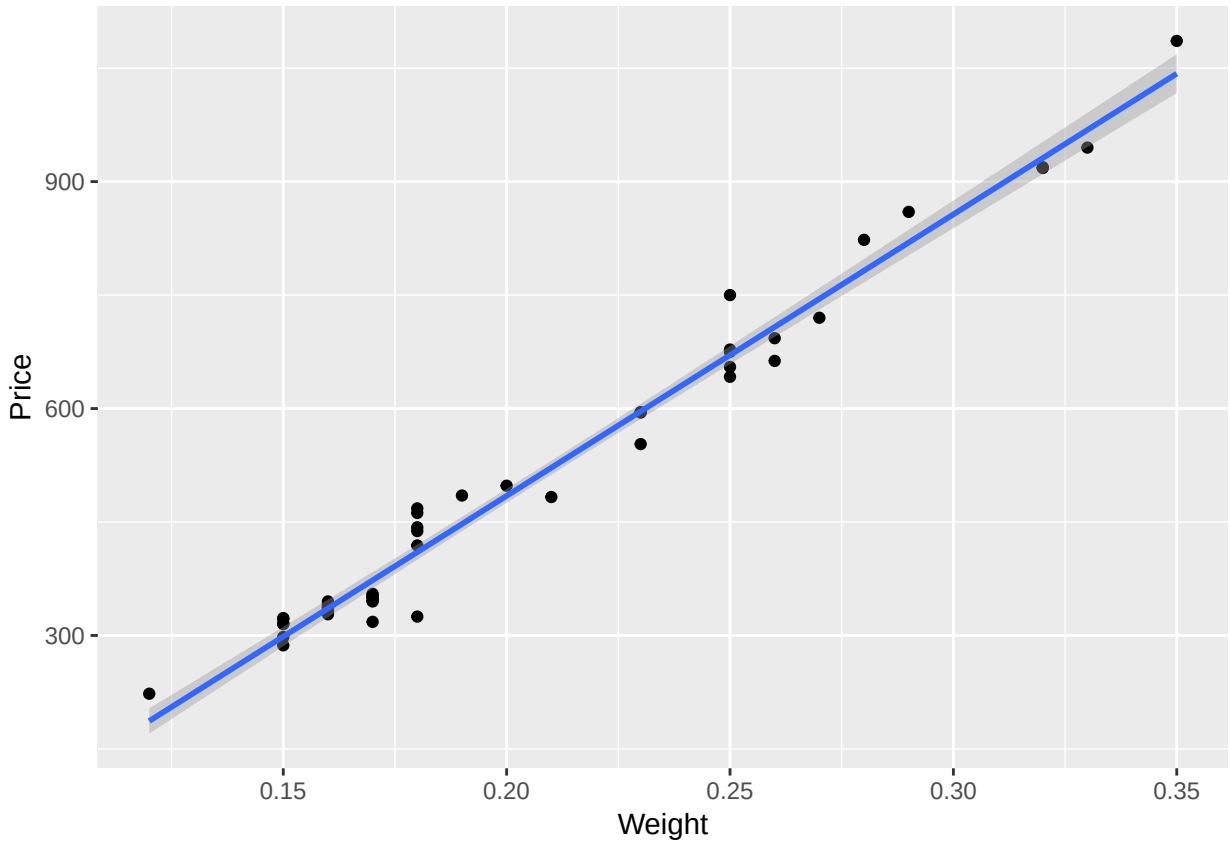
As introduced in the bootcamp, `ggplot2` generates a plot by adding the procedures. `geom_point` is the command to create scatterplots. Unlike base R that plots straight lines by coefficients, `ggplot2` connects the prediction points together, which in linear regression is a straight line.

```
diamond %>%
  ggplot(aes(x = Weight, y = Price)) +
  geom_point() +
  geom_line(aes(y = fitted(model1)))
```



`ggplot2` offers an alternative way that does not require a pre-fitted model at all. The command `geom_smooth(method = 'lm')` fit a linear model internally and visualize the regression line on the same plot. Additionally, by setting the argumment `se` to be `TRUE` (it is `TRUE` by default), `geom_smooth` also displays the confidence interval of the model. See the shaded gray area around the blue regression line.

```
diamond %>%
  ggplot(aes(x = Weight, y = Price)) +
  geom_point() +
  geom_smooth(method = 'lm', se = TRUE)
```



Model Diagnostics

Linear regression has rigid assumptions. It does not only assume the linear relationship between predictors and the response, but also require the errors to be independent, has equal variance, and normally distributed. It is critical to check if all these assumptions are satisfied, otherwise, the analysis results and conclusions are questionable, if not dangerous.

Generally speaking, linear relationships can be backed up by statistics such as R^2, significance level, CI, etc, while assumptions associated with residuals are better examined through visualizations.

Residual by Prediction Plot

The residual by prediction plot offers a good overview of the dependence of residuals on predictions.

The `ggplot2` part of codes are straightforward. It creates the scatter plot on the left. The x-axis is highlighted in red to give visual cue for us to decide the distribution of residuals, since most of the time, such judgement is quite empirical.

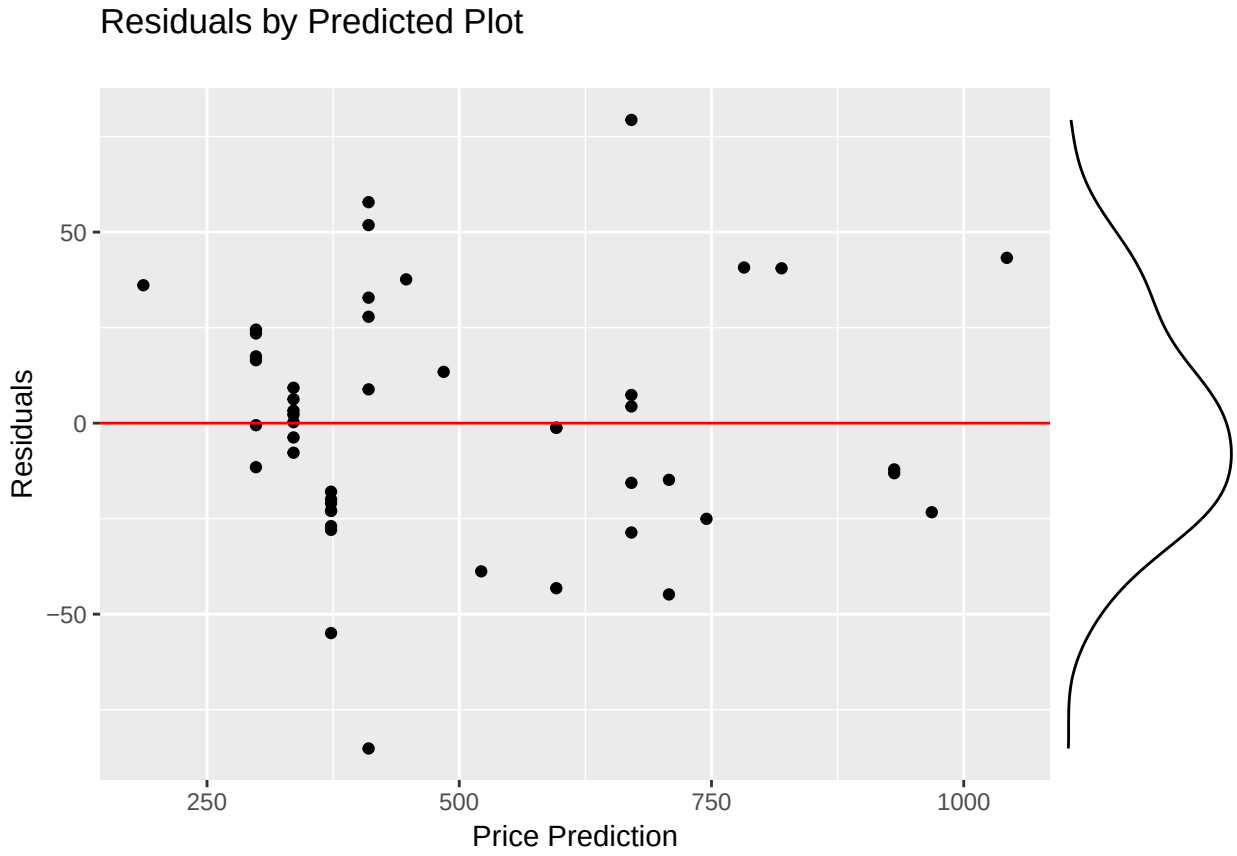
To make it less subjective, `ggMarginal`, a function from `ggExtra`, adds a density function for residuals on the right. Since the first argument of `ggMarginal` is a `ggplot` object, we need to assign the plot returned by `ggplot2` to a variable. The `type` argument speficies type of marginal plot. Apart from `density`, it can also be `histogram` and `boxplot`. `margins` determines the margins that along which to show the plots. In this example, we only care about the distribution of `residuals`, therefore, the `margins` is set to be `y`, instead of `x` or `both`.

From the density plot, we clearly observe a symmetric bell-shape curve. Even though the symmetry axis of the curve is not strictly the x-axis, it is not too much off, so the normality assumption is valid. Moreover, the

plot reveals no systematic patterns in the residuals.

The last two lines resolve a bug that make `ggMarginal` incompatible with R Markdown.

```
p <-diamond %>%
    ggplot(aes(x = fitted(model1), y = model1$residuals)) +
    geom_point() +
    geom_hline(yintercept = 0, col = 'red') +
    labs(title = 'Residuals by Predicted Plot',
         x = 'Price Prediction',
         y = 'Residuals')

p <- ggMarginal(p, type="density", margins="y")

grid::grid.newpage()
grid::grid.draw(p)
```



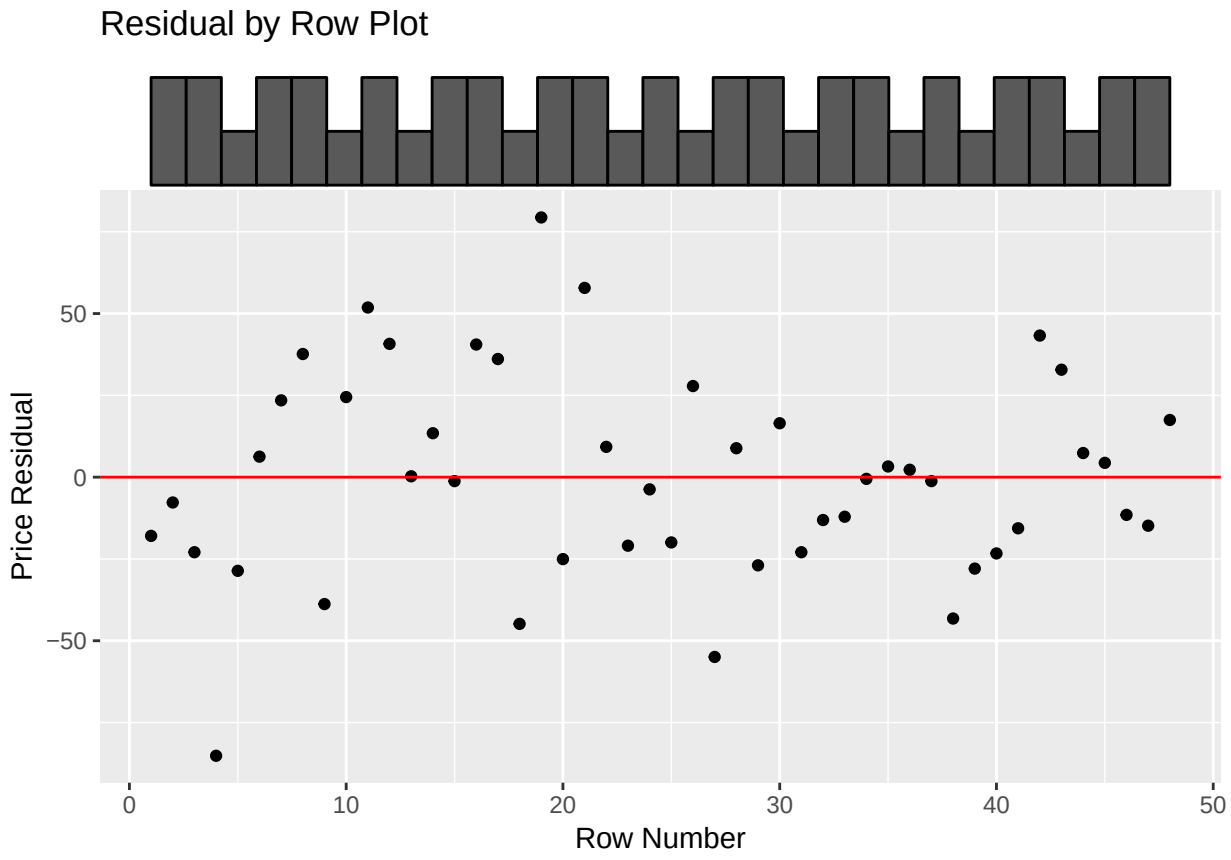
Residual by Row Plot

In the class, we discussed the seasonality of data, where residuals might vary with time. In the practice, when the time information is missing, the row numbers can be a good indicator of time. Therefore, in some cases, residual by row plot can be used to check the time dependency of residuals.

This is a bad example because the row in the Diamond dataset is not likely to follow chronological order. Nevertheless, this example gives you hints on how to do it.

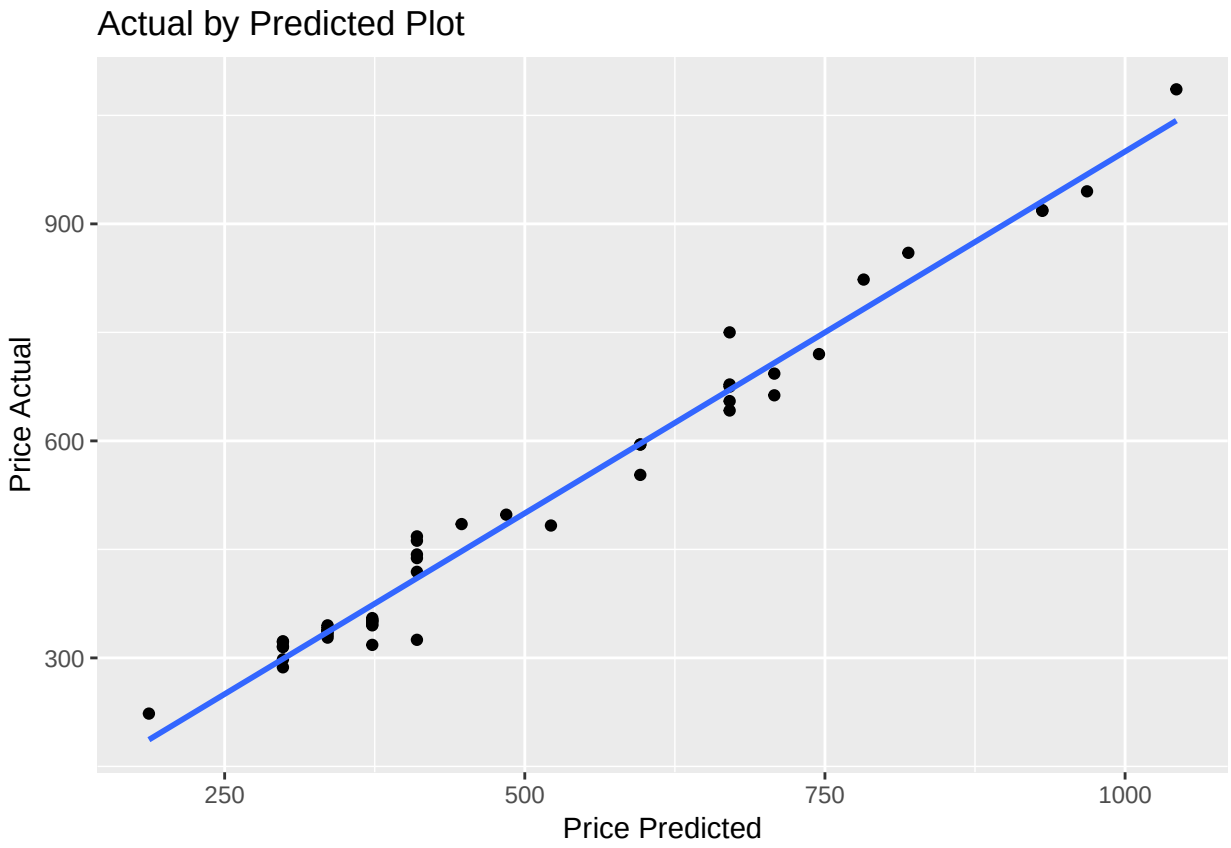
```
p <- diamond %>%
    ggplot(aes(x = seq(1, 48), y = model1$residuals)) +
```

```
    geom_point() +
    geom_hline(yintercept = 0, col = 'red') +
    labs(title = 'Residual by Row Plot',
         x = 'Row Number',
         y = 'Price Residual')

p <- ggMarginal(p, type="histogram", margins="x")

grid::grid.newpage()
grid::grid.draw(p)
```



Actual by Predicted Plot

```
diamond %>%
  ggplot(aes(x = fitted(model1), y = Price)) + geom_point() +
  geom_smooth(method = 'lm', se = FALSE) +
  labs(title = 'Actual by Predicted Plot',
       x = 'Price Predicted',
       y = 'Price Actual')
```



The estimated slope (~ 0.978) is pretty close to 1, so our linear model did a good job.

```
par_table(lm(fitted(model1) ~ diamond$Price))
```

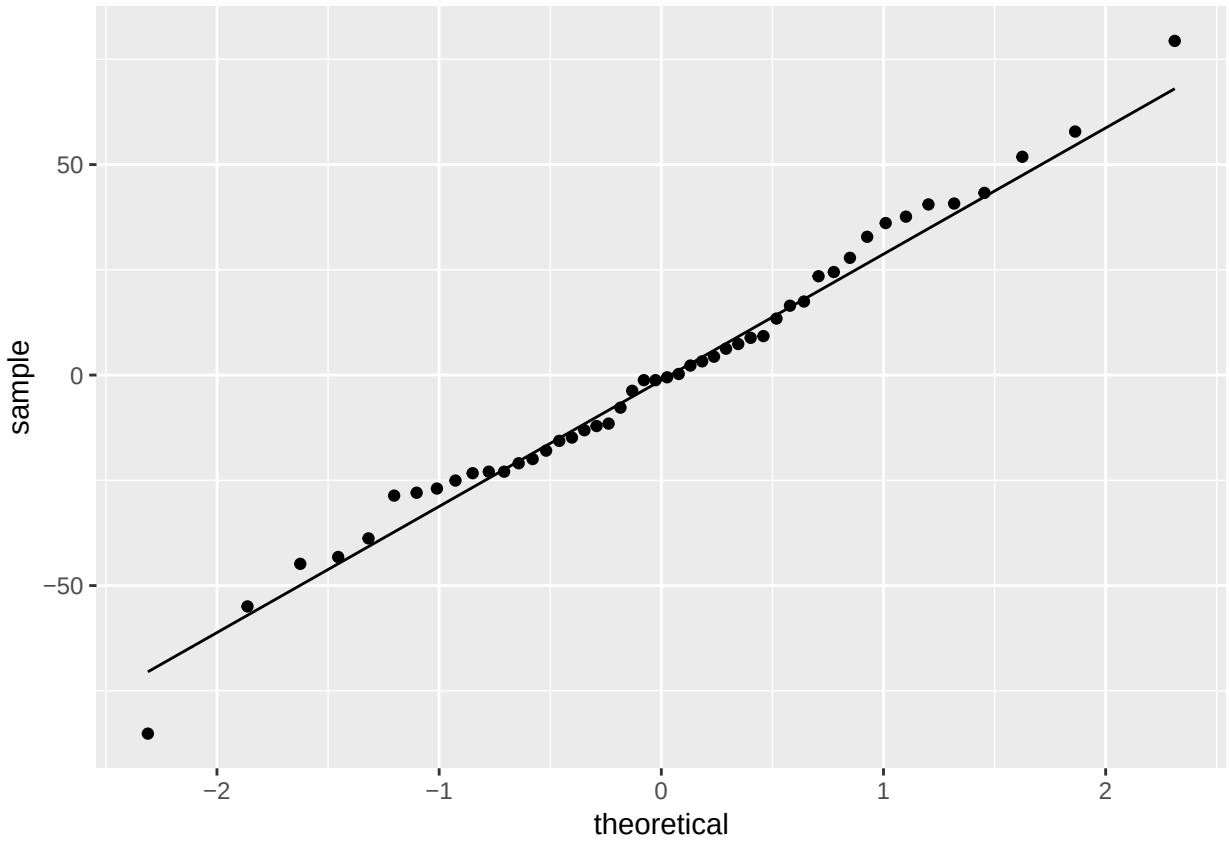
```
##                  Estimate  Std. Error    t value    Pr(>|t|)       2.5 %
## (Intercept)   10.8714226 11.67390672  0.9312583 3.56582e-01 -12.6269329
## diamond$Price  0.9782608  0.02150158 45.4971547 6.75126e-40   0.9349803
##                  97.5 %
## (Intercept)   34.369778
## diamond$Price  1.021541
```

Normal Q-Q Plot

Quantile-quantile plot (Q-Q plot), by its name, ploting quantiles of two distribution against each other. If the two distributions being compared are simular, the points in the Q-Q plot will approxmately line on the line $y = x$.

Normally, the x-axis of a Q-Q plot is the standard normal distribution. If the data to be compared is also the normal distribution (not necessarily standard), points in the plot will be closely located along the $y = x$.

```
y = model1$residuals
y = as.data.frame(y)

ggplot(y, aes(sample=y)) + stat_qq() + stat_qq_line()
```



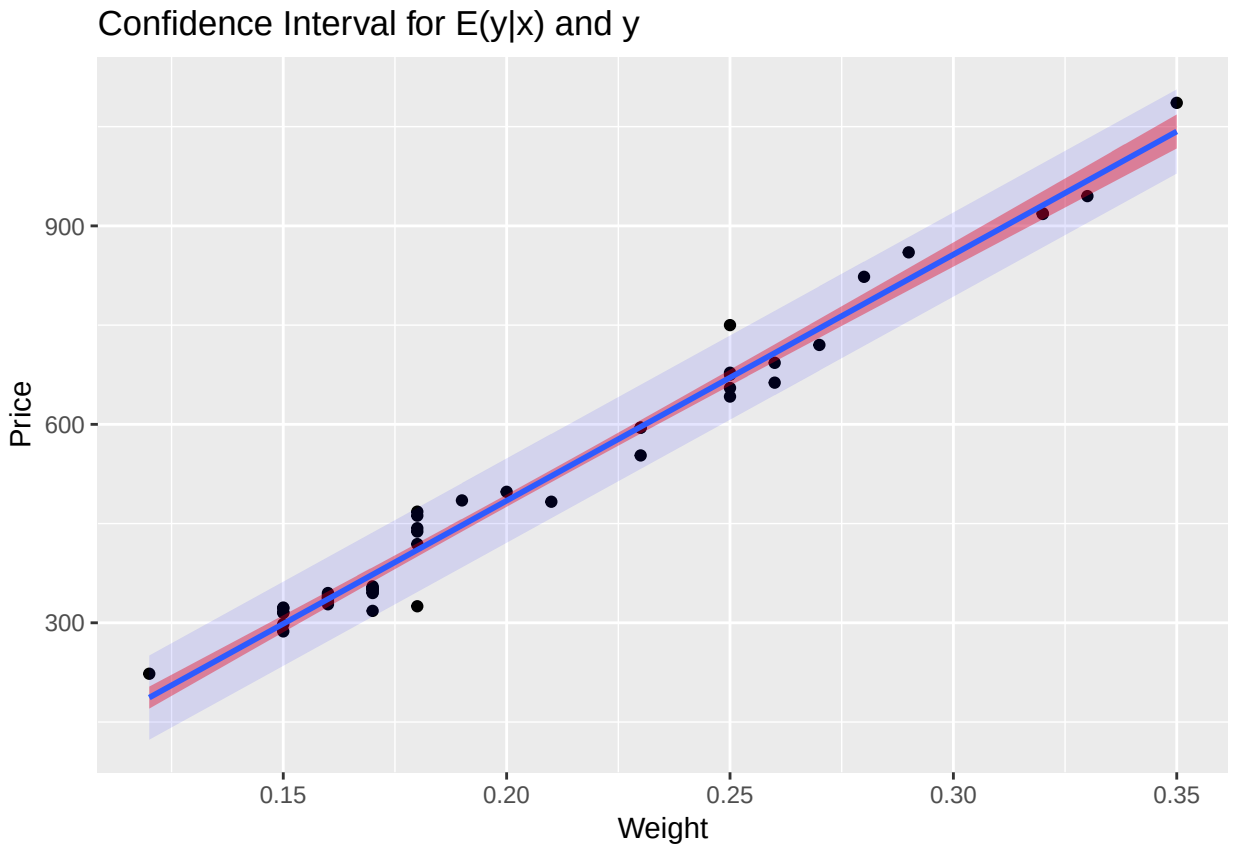
Confidence Interval

Confidence Interval for E(y|x) and y

Previously, I have introduced ploting the confidence interval for E(y|x) in `ggplot2` by setting the `se` in the `geom_smooth` function to be `TRUE`. However, to plot the confidence interval for `y`, we need to manually color some bounded regions. In this example, the region is approximated by ±2RMSE from the regression line.

The command `geom_ribbon` takes an upper bound (E(y|x)+2RMSE), a lower bound (E(y|x)-2RMSE) and colors the region in between. `fill` specifies the color to be filled, and `alpha` controls the degree of transparency.

```
RMSE = summary(model1)$sigma

diamond %>%
  ggplot(aes(x=Weight, y=Price)) +
  geom_point() +
  geom_smooth(method = "lm", fill = 'red') +
  labs(title = 'Confidence Interval for E(y|x) and y') +
  geom_ribbon(aes(ymin=fitted(model1) - 2*RMSE,
                  ymax=fitted(model1) + 2*RMSE),
              fill="blue", alpha="0.1")
```



Issues: Assumption Violation

So far we have been using the diamonds dataset. We have fitted a linear model and examined if the assumptions of simple linear regression are violated. The diamonds dataset is almost ideal but it is usually not the case in practice. This section reproduces those analyses that address the assumption violation issues in the lecture notes.

The code in this section is not much different from the previous code and the disucssion about each figure can be easily found on the lecture note, so I will skip most of explainations.

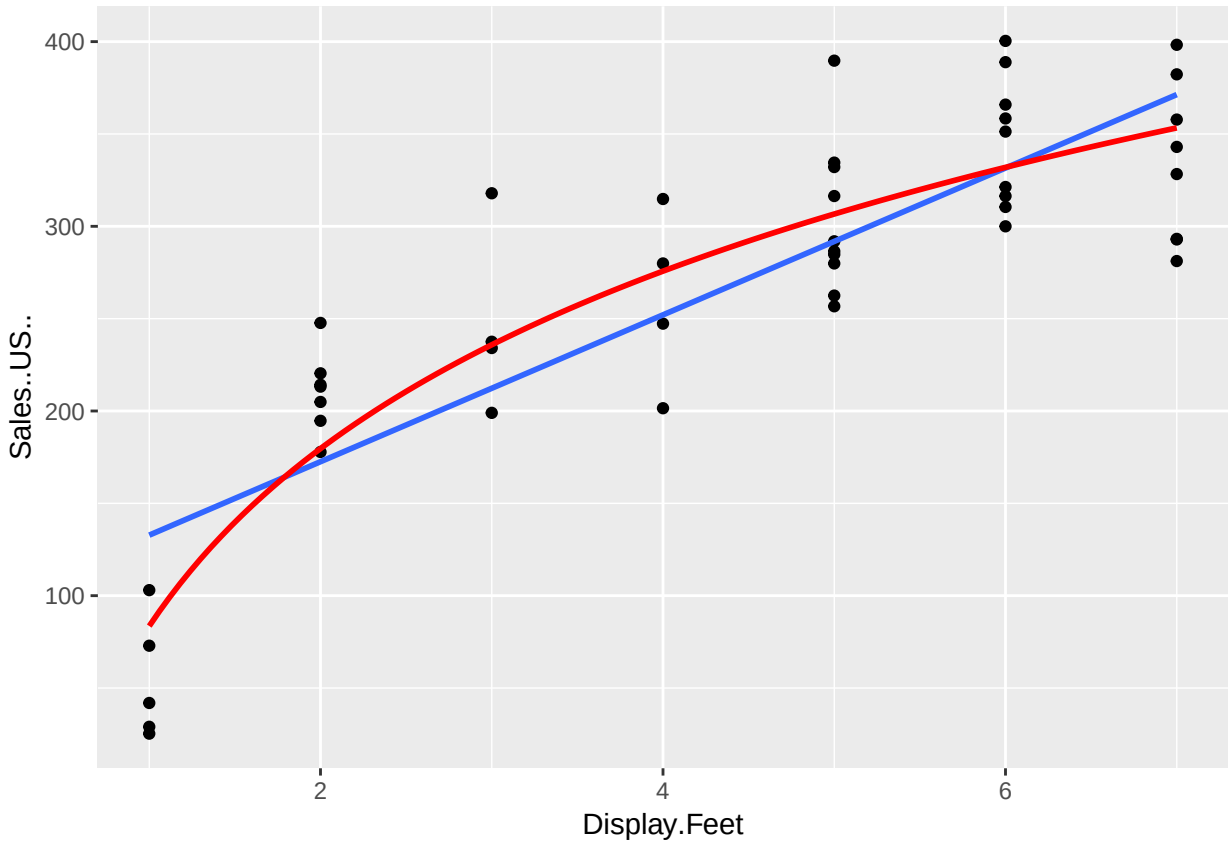
Issue 1: Non-linearity

The `Display` dataset: log transformation on the feature

```
display <- read.csv('display.dat')

model_dis1 <- lm(display$Sales..US.. ~ display$Display.Feet)
model_dis2 <- lm(display$Sales..US.. ~ log(display$Display.Feet))
```

To plot the curve in `ggplot2`, one needs to define the expression of the curve and call `stat_function`.

```
func <- function(x) log(x) *  coef(model_dis2)[2] + coef(model_dis2)[1]

display %>%
  ggplot(aes(x = Display.Feet, y = Sales..US..)) + geom_point() +
  geom_smooth(method = 'lm', se = FALSE)  +
  stat_function(fun = func, col = 'red', size = 1)
```



Compare the two models

```
fit_table(model_dis1, display$Sales..US..)
```

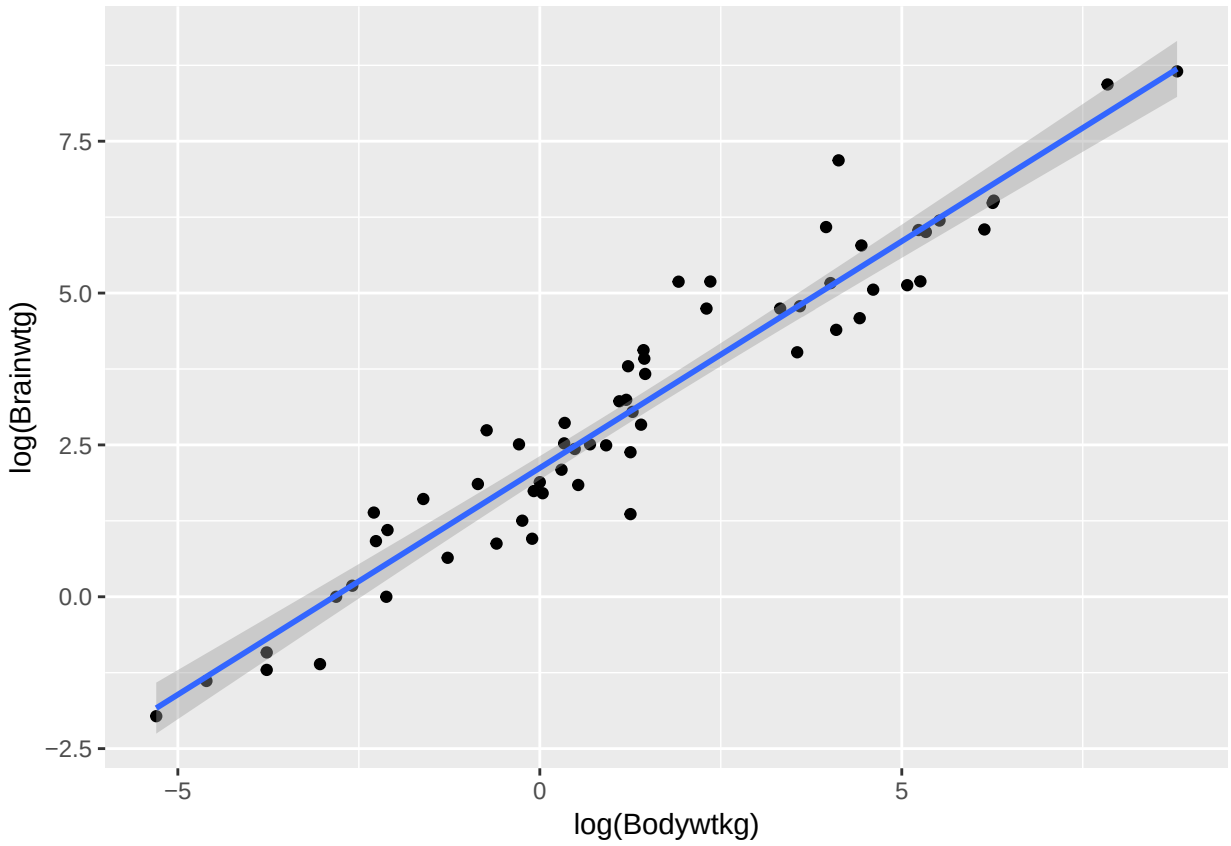
```
##                              [,1]
## RSquare                 0.7119746
## RSquare.adj             0.7055741
## Root.Mean.Square.Error 51.5912296
## Mean.of.Response       268.1300000
## Observations            47.0000000
```

```
fit_table(model_dis2, display$Sales..US..)
```

```
##                              [,1]
## RSquare                 0.8153491
## RSquare.adj             0.8112457
## Root.Mean.Square.Error 41.3081997
## Mean.of.Response       268.1300000
## Observations            47.0000000
```

The `Mammals` dataset: log transformation on both the feature and the response

```
mammals <- read.csv('mammals.dat')
model_mam <- lm(log(mammals$Brainwtg) ~ log(mammals$Bodywtkg))
mammals %>%
  ggplot(aes(x = log(Bodywtkg), y = log(Brainwtg))) +
  geom_point() +
  geom_smooth(method = 'lm')
```



Notice the second argument of the function `fit_table`!

```
fit_table(model_mam, log(mammals$Brainwtg))
```

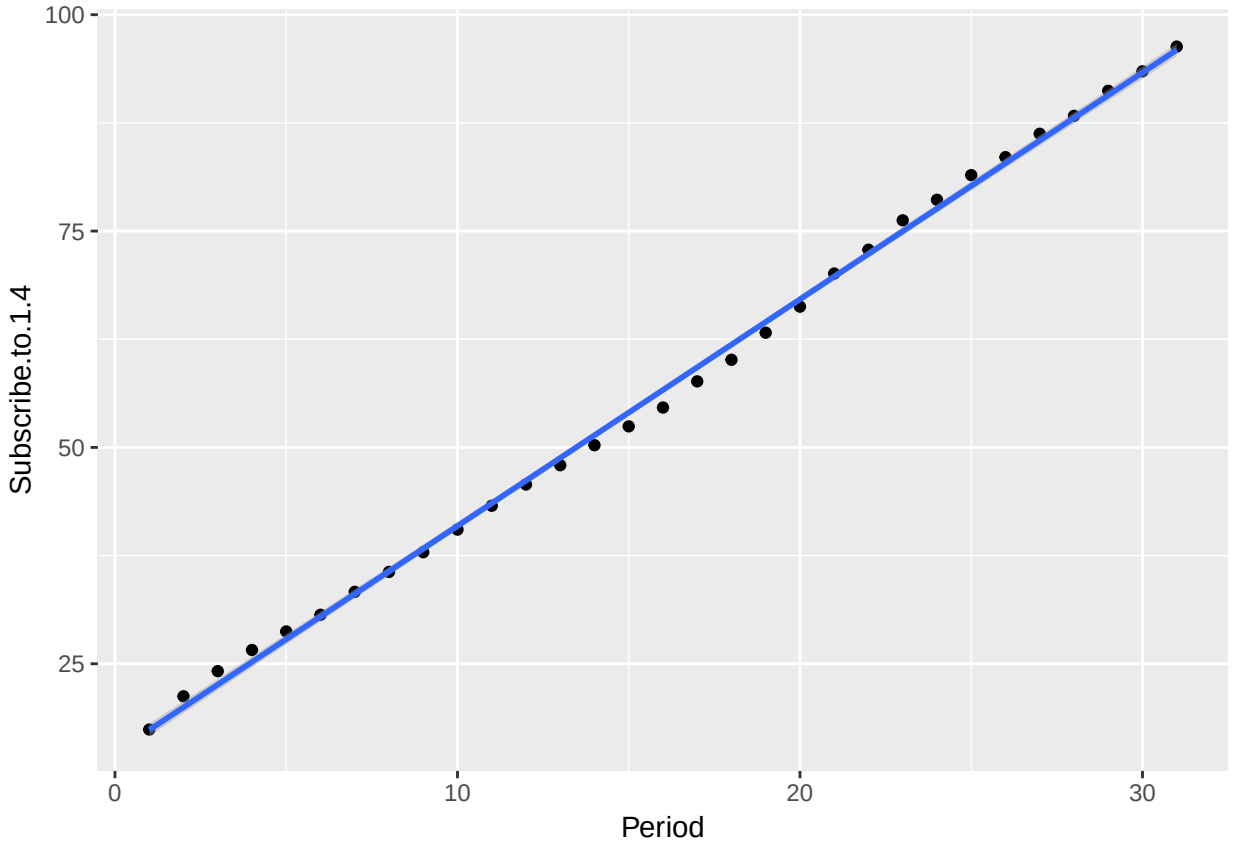
```
##                              [,1]
## RSquare                 0.9193300
## RSquare.adj             0.9179855
## Root.Mean.Square.Error  0.7006364
## Mean.of.Response        3.1401979
## Observations           62.0000000
```

Issue 2: Auto-corrected Residuals

The Cellular dataset: power transform on the response

```
cellular <- read.csv('cellular.dat')
model_cel <- lm(cellular$Subscribers^(0.25) ~ cellular$Period)
```

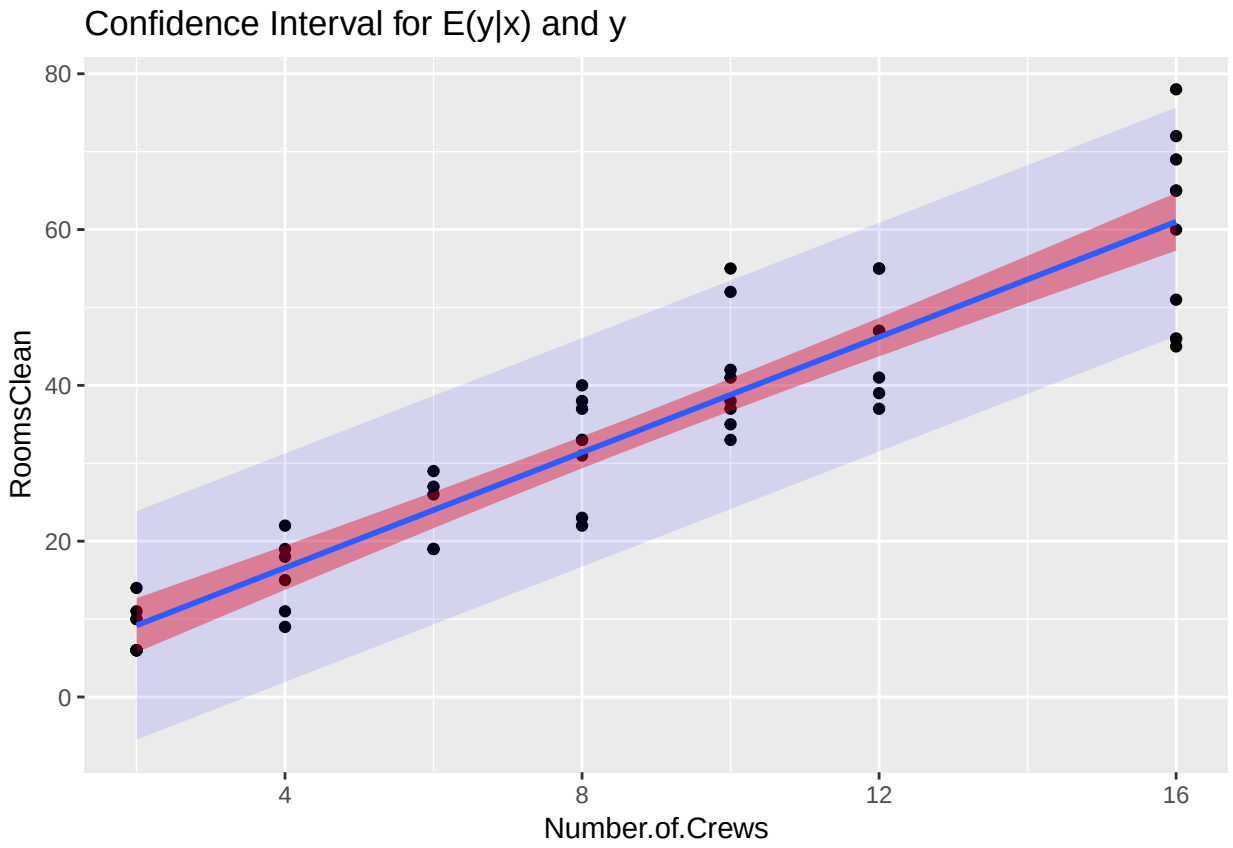
```
cellular %>%
  ggplot(aes(x = Period, y = Subscribe.to.1.4)) +
  geom_point() +
  geom_smooth(method = 'lm')
```



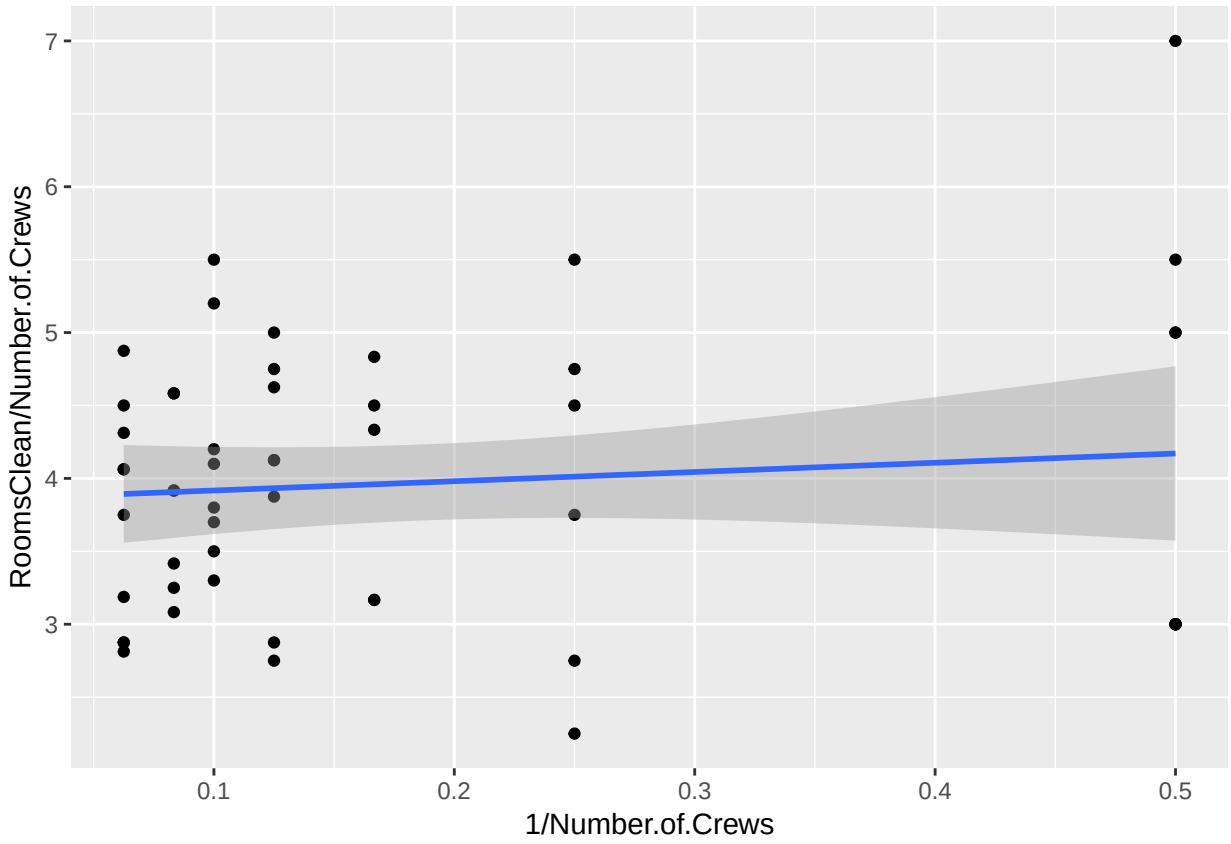
Issue 3: Non-consistant Variance, or Heteroscedasticity

The Cleaning dataset: the inverse of the feature $\text{RoomsClean} = b_0 + b_1(\text{NumCrews}) \rightarrow \text{RoomsPerCrew} = \frac{b_0}{\text{NumCrews}} + b_1$

```
cleaning <- read.csv('cleaning.dat')
model_clg <- lm(cleaning$RoomsClean ~ cleaning$Number.of.Crews)
RMSE <- fit_table(model_clg, cleaning$RoomsClean)[3]

cleaning %>%
  ggplot(aes(x=Number.of.Crews, y=RoomsClean)) +
  geom_point() +
  geom_smooth(method = "lm", fill = 'red') +
  labs(title = 'Confidence Interval for E(y|x) and y') +
  geom_ribbon(aes(ymin=fitted(model_clg) - 2*RMSE,ymax=fitted(model_clg) + 2*RMSE), fill="blue", alpha=
```




```
model_clg2 <- lm(cleaning$RoomsClean/cleaning$Number.of.Crews ~ 1/cleaning$Number.of.Crews)
cleaning %>%
  ggplot(aes(x=1/Number.of.Crews, y=RoomsClean/Number.of.Crews)) +
  geom_point() +
  geom_smooth(method = "lm")
```



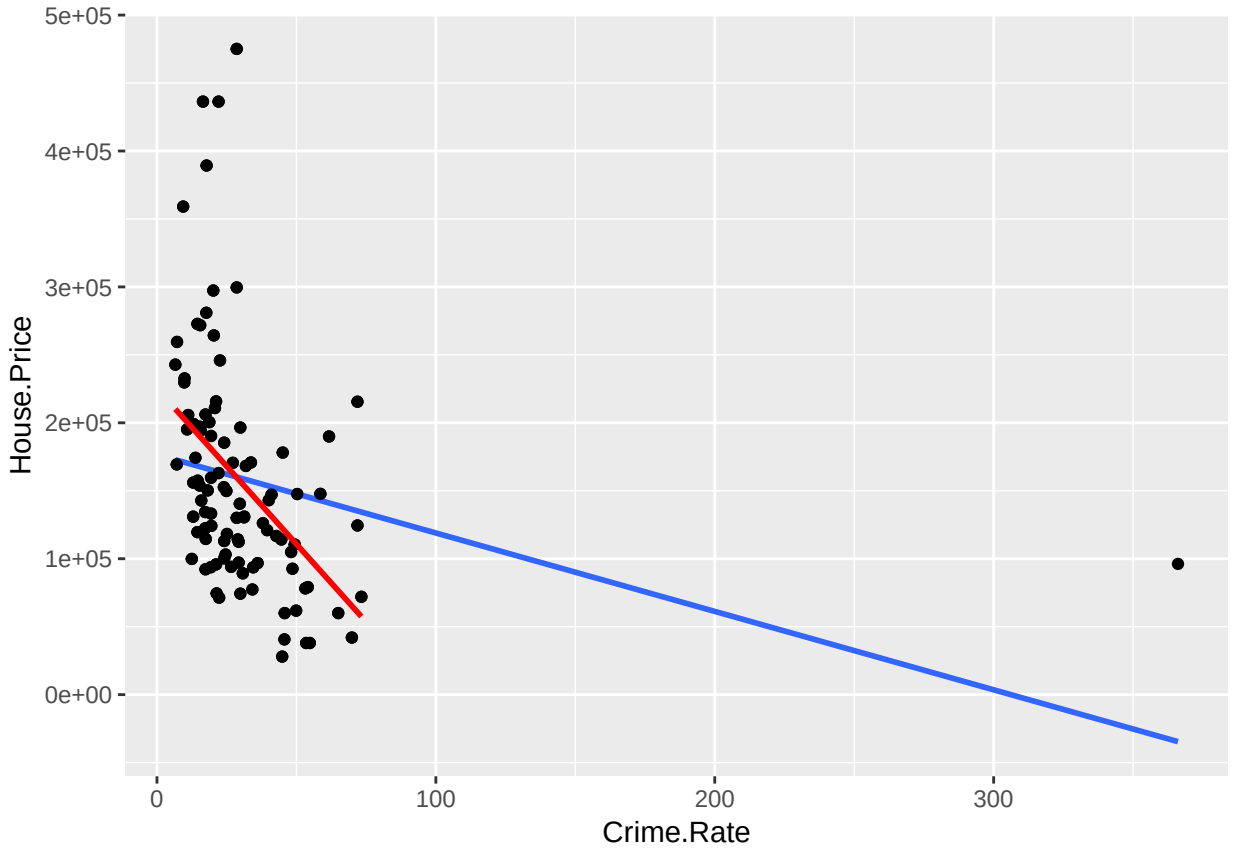
Issue 4: Outliers and Leverage Points

The Phila dataset: remove the outlier

```
phila <- read.csv('phila.dat')
```

```
# Get rid of NA values in phila
phila <- phila[complete.cases(phila[,1:2]), ]

# Get rid of the leverage point
phila2 <- phila %>%
          filter(Crime.Rate < 300)
```

```
ggplot() +
  geom_point(data = phila, aes(x = Crime.Rate, y = House.Price)) +
  geom_smooth(data = phila, aes(x = Crime.Rate, y = House.Price),
              method = 'lm', se = F) +
  geom_point(data = phila2, aes(x = Crime.Rate, y = House.Price)) +
  geom_smooth(data = phila2, aes(x = Crime.Rate, y = House.Price),
              method = 'lm', col = 'red', fill = 'red', se = F)
```




```
# options(scipen=999) #Turn scientific notation off
# options(scipen=1) #Turn scientific notation on
fit_table(lm(phila$House.Price ~ phila$Crime.Rate), phila$House.Price)
```

```
##                              [,1]
## RSquare                 6.248047e-02
## RSquare.adj             5.281532e-02
## Root.Mean.Square.Error 8.432505e+04
## Mean.of.Response        1.578356e+05
## Observations            9.900000e+01
```

```
fit_table(lm(phila2$House.Price ~ phila2$Crime.Rate), phila2$House.Price)
```

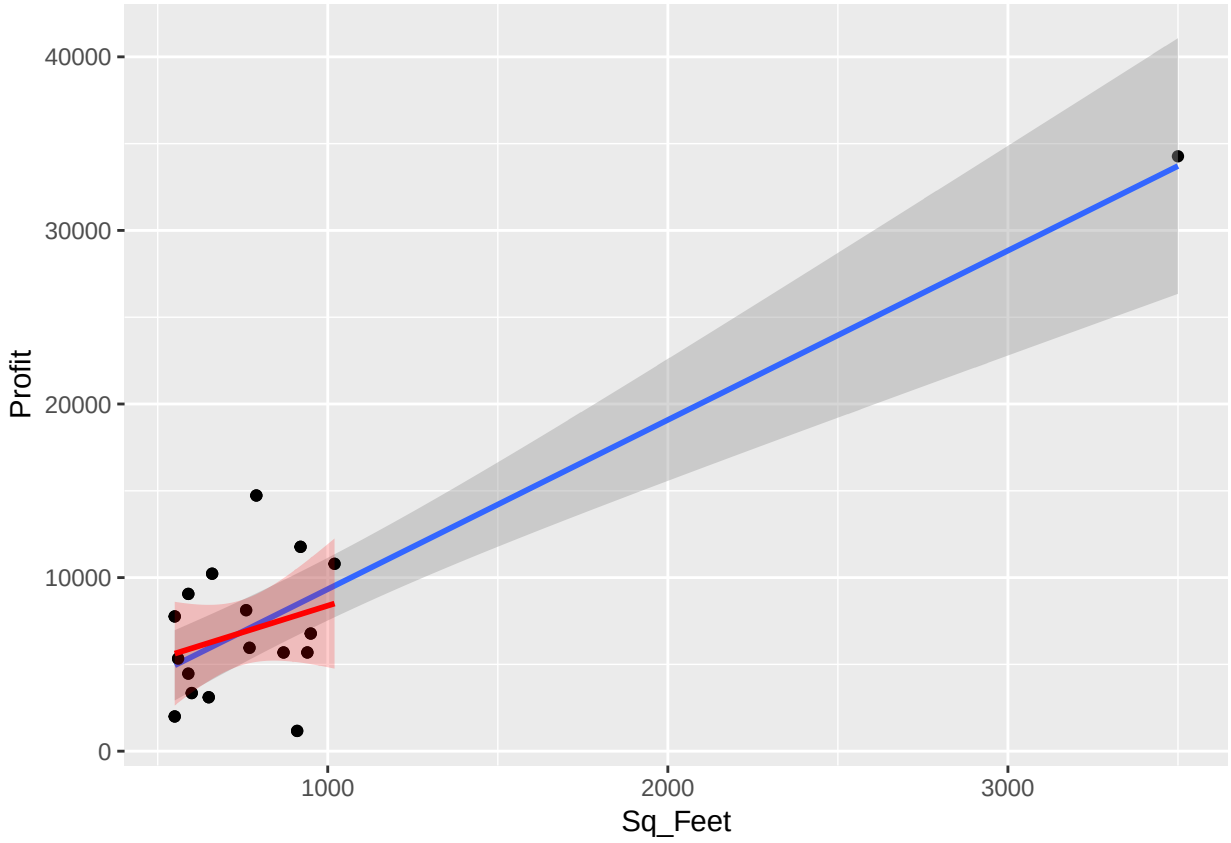
```
##                              [,1]
## RSquare                 1.842288e-01
## RSquare.adj             1.757312e-01
## Root.Mean.Square.Error 7.886153e+04
## Mean.of.Response        1.584645e+05
## Observations            9.800000e+01
```

The Cottage dataset: remove the outlier

```
cottage <- read.csv('cottages.dat')
cottage2 <- cottage %>%
  filter(Sq_Feet < 3000)
```

```
ggplot() +
  geom_point(data = cottage, aes(x = Sq_Feet, y = Profit)) +
  geom_smooth(data = cottage, aes(x = Sq_Feet, y = Profit), method = 'lm') +
```

```
  geom_point(data = cottage2, aes(x = Sq_Feet, y = Profit)) +
  geom_smooth(data = cottage2, aes(x = Sq_Feet, y = Profit), method = 'lm',
              col = 'red', fill = 'red', alpha = 0.2)
```



Multiple Regression

Multiple regression and simple regression have a lot in common, since multiple regression can be seen as an extension of simple regression. Therefore, what covered previously in this Rmd file also applies to multiple regression.

The multiple regression models are also fitted by `lm`. The features are joined by `+` and the orders of them do not matter. i.e. `multiple.regression.model = lm(y ~ x_0 + x_1 + x_2)`. Special cases involved categorical variables will be addressed later.

The lecture notes cover the following additional visualizations and statistics for multiple regression.

```
sp <- read.csv('smartphone.dat')
car04 <- read.csv('cars04.dat')
```

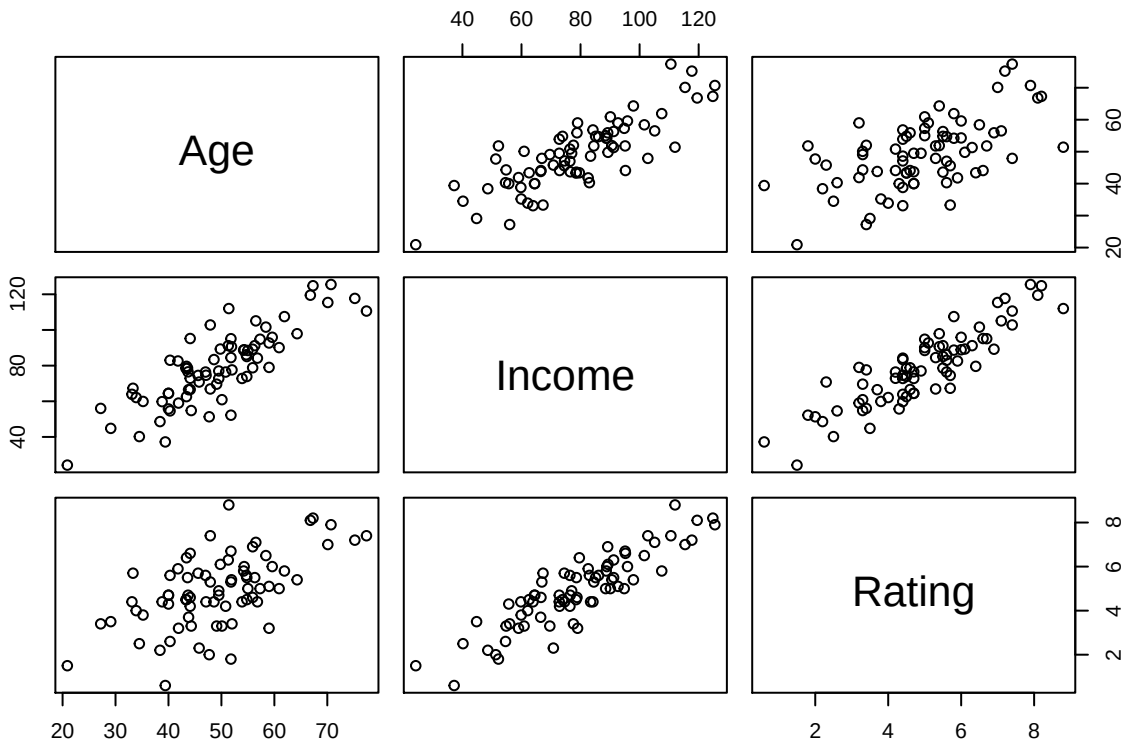
```
str(sp)
```

```
## 'data.frame':    75 obs. of  4 variables:
##  $ Age   : num  56.8 52 29.1 77.4 70.7 47.1 44.3 33.9 49.5 56.3 ...
##  $ Income: num  84.2 77.6 44.8 110.6 125.5 ...
##  $ Rating: num  4.4 3.4 3.5 7.4 7.9 4.4 3.3 4 4.9 5.5 ...
##  $ Group : Factor w/ 4 levels "","Hi","Low",..: 1 4 3 2 2 4 1 1 4 1 ...
```

Scatterplot Matrix: Exploratory Analysis

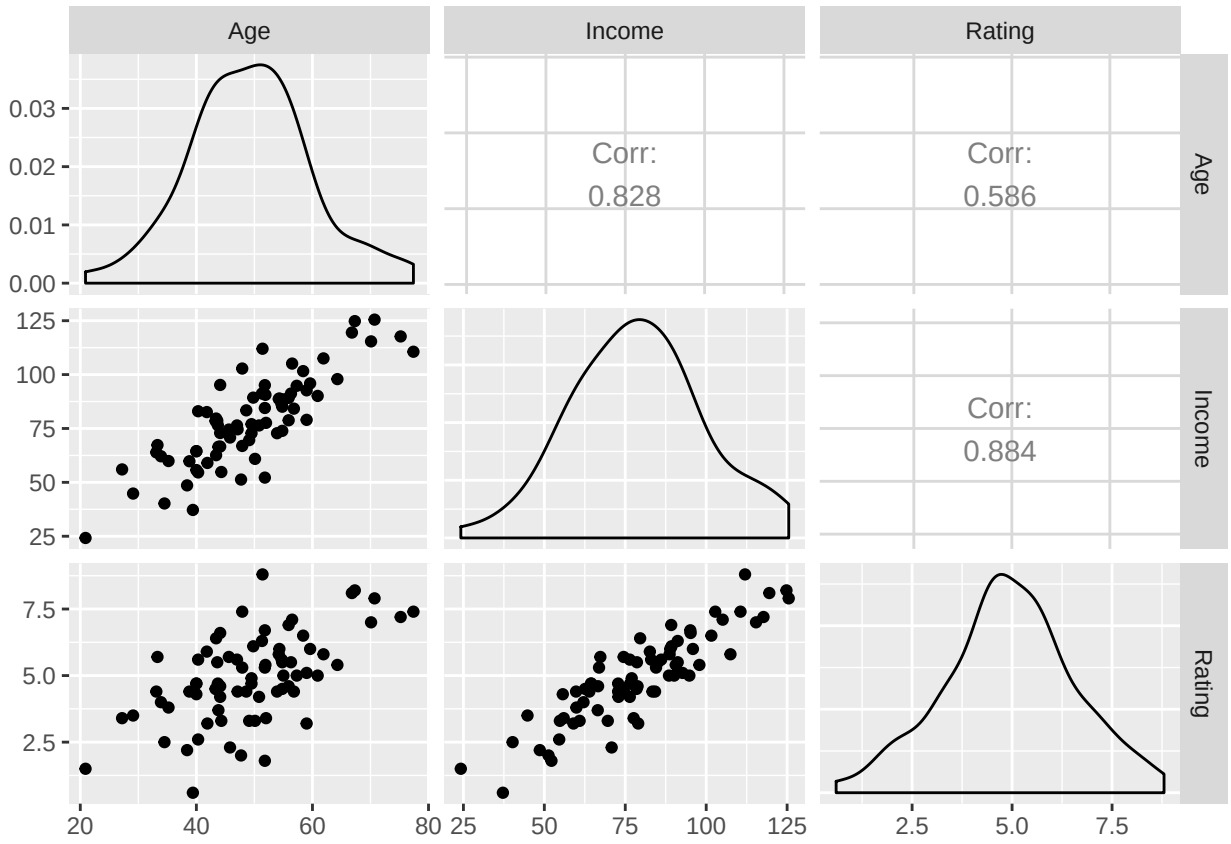
Base R

```
plot(sp[,1:3])
```



ggplot2

```
ggpairs(sp[,1:3])
```



Correlation

```
cor(sp[,1:3], method='pearson')
```

```
##              Age    Income    Rating
## Age    1.0000000 0.8284817 0.5861361
## Income 0.8284817 1.0000000 0.8835885
## Rating 0.5861361 0.8835885 1.0000000
```

Fitting and Analysing a Multiple Regression Model

```
model_sp <- lm(sp$Rating ~ sp$Age + sp$Income)
```

Variance Inflation Factor (VIF)

```
vif(model_sp)
```

```
##    sp$Age sp$Income
##  3.188591  3.188591
```

F statistics

```
anova(model_sp)
```

```
## Analysis of Variance Table
##
## Response: sp$Rating
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## sp$Age     1  68.932  68.932  163.39 < 2.2e-16 ***
## sp$Income  1 101.334 101.334  240.19 < 2.2e-16 ***
## Residuals 72  30.376   0.422
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Anova(model_sp, type=2)
```

```
## Anova Table (Type II tests)
##
## Response: sp$Rating
##            Sum Sq Df F value    Pr(>F)
## sp$Age     13.619  1   32.28 2.648e-07 ***
## sp$Income 101.334  1  240.19 < 2.2e-16 ***
## Residuals  30.376 72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Anova(model_sp, type=3)
```

```
## Anova Table (Type III tests)
##
## Response: sp$Rating
##              Sum Sq Df  F value    Pr(>F)
## (Intercept)   0.879  1   2.0831    0.1533
## sp$Age       13.619  1  32.2800 2.648e-07 ***
## sp$Income   101.334  1 240.1887 < 2.2e-16 ***
## Residuals    30.376 72
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

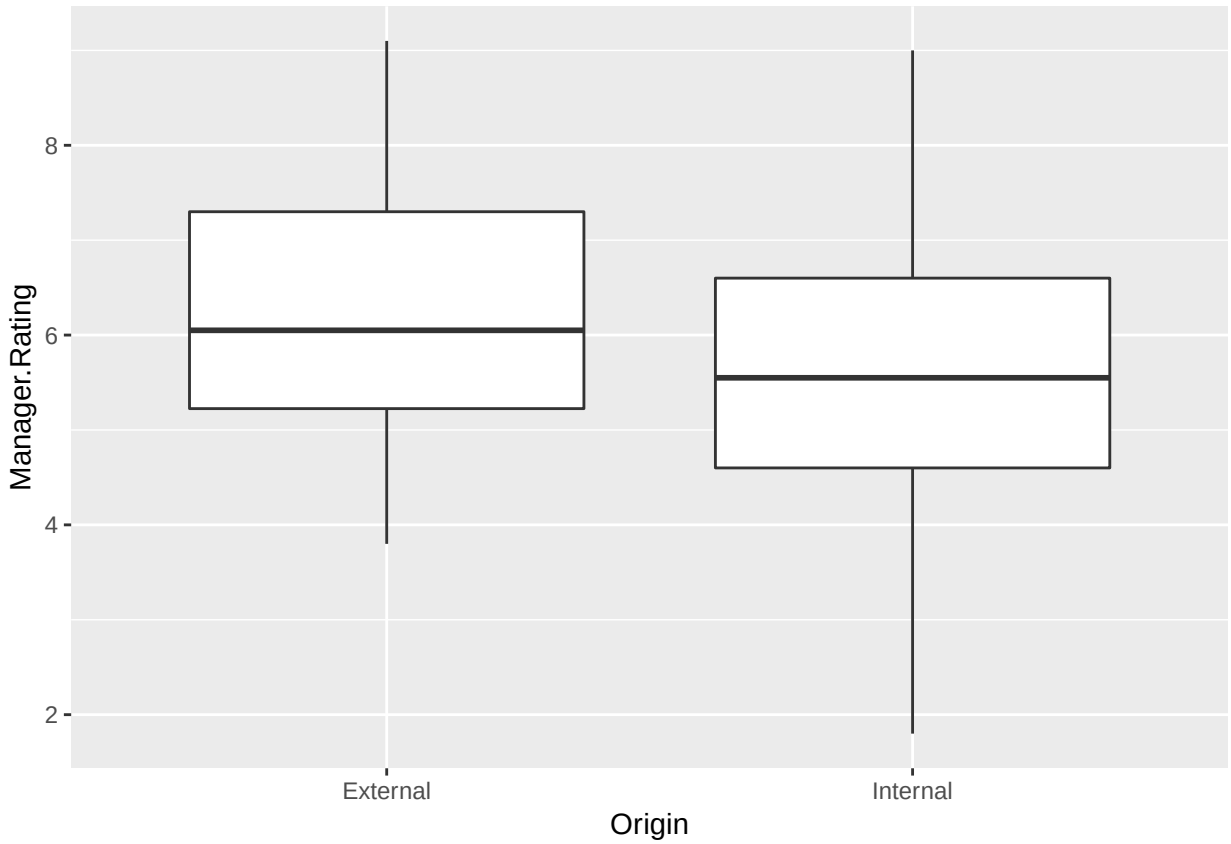
Why type 2 and type 3 results are the same?

Categorical Explanatory Variables

```
manager <- read.csv('managerrating.dat')
```

Two-Sample Comparison: Boxplot

```
ggplot(manager, aes(x=Origin, y=Manager.Rating)) +
  geom_boxplot()
```



From the boxplot, we can observe that the ratings of external managers are relatively higher than those of internal managers. However, is this difference significant? We can find out by either the t-test or comparing the confidence intervals of means of the two groups.

t-test

A t-test's statistical significance indicates whether or not the difference between two groups' averages most likely reflects a "real" difference in the population from which the groups were sampled.

```
t_test <- t.test(Manager.Rating ~ Origin, manager)
t_table(t_test)
```

```
##    t.ratio       DF    prob..t. upper.CL.Diff lower.CL.Diff Difference
## t 3.048412 140.4931 0.002749595     0.2517995      1.181045  0.7164223
```

Confidence Interval

This is how I plot the graph manually:

First, Calculating the standard error of the mean

The standard error of the mean is the expected value of the standard deviation of means of several samples, this is estimated from a single sample as:

$$\sigma_m = \frac{\sigma}{\sqrt{n}}$$

where

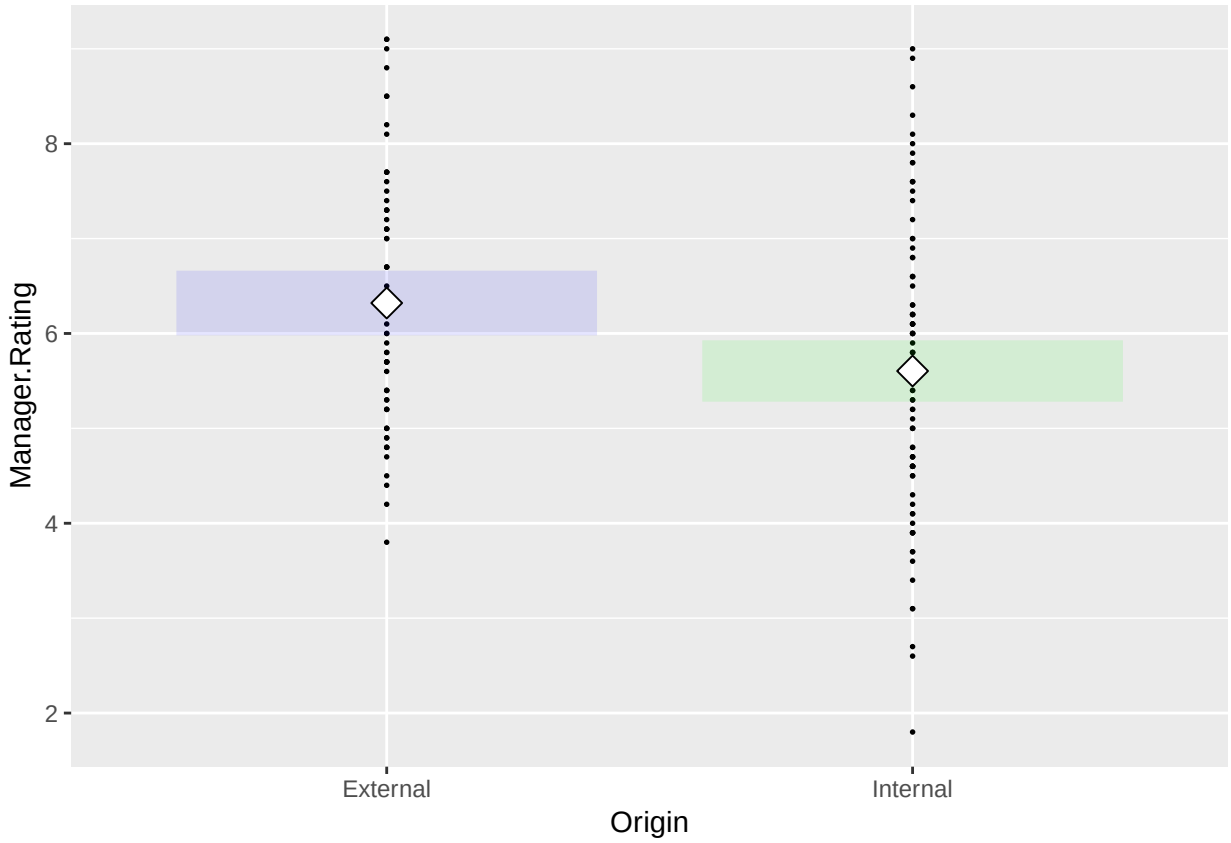
1. σ is the standard deviation of the sample

2. σ_m is the standard deviation of the sample mean
3. n is the sample size

Second, apply `geom_ribbon` to color the regions within $2\sigma_m$ of the sample mean for each group.

```
external <- manager %>%
  filter(Origin == 'External')
internal <- manager %>%
  filter(Origin == 'Internal')
```

```
mean_ex <- mean(external$Manager.Rating)
mean_in <- mean(internal$Manager.Ratin)
std_ex <- sd(external$Manager.Ratin) / sqrt(length(external$Manager.Ratin))
std_in <- sd(internal$Manager.Ratin) / sqrt(length(internal$Manager.Ratin))

ggplot(manager, aes(x=Origin, y=Manager.Rating)) +
  geom_point(size = 0.3) +
  geom_ribbon(aes(x = seq(0.6, 1.4, length.out = 150), ymin=mean_ex - 2 * std_ex,
                  ymax=mean_ex + 2 * std_ex), fill="blue", alpha="0.1")+
  geom_ribbon(aes(x = seq(1.6, 2.4, length.out = 150), ymin=mean_in - 2 * std_in,
                    ymax=mean_in + 2 * std_in), fill="green", alpha="0.1") +
  stat_summary(fun.y="mean", geom="point", shape=23, size=4, fill="white")
```



The 95% CI for the two group do not overlap, therefore we can conclude that the external managers do on average get a higher rating than the internal managers.

Scatterplot of Manager Rating vs. Salary

```
manager %>%
  ggplot(aes(x = Salary..US.000., y = Manager.Rating)) +
  geom_point(aes(col = factor(Origin)))
```

Fitting SRM Separately

```
model_ex <- lm(Manager.Rating ~ Salary..US.000., external)
model_in <- lm(Manager.Rating ~ Salary..US.000., internal)
ggplot() +
  geom_point(data = manager, aes(x = Salary..US.000.,
                                 y = Manager.Rating,
                                 col = factor(Origin))) +
  geom_line(data = external, aes(x = Salary..US.000.,
                                 y = fitted(model_ex)),
                                 col = 'red', alpha = 0.7) +
  geom_line(data = internal, aes(x = Salary..US.000.,
                                 y = fitted(model_in)),
                                 col = 'deepskyblue3', alpha = 1)
```

Review: an alternative way to plot regression lines:

```
manager %>% ggplot() +
  geom_point(aes(x = Salary..US.000., y = Manager.Rating, col = factor(Origin))) +
  geom_smooth(method = 'lm',
              aes(x = Salary..US.000., y = Manager.Rating, group = factor(Origin),
                  col = factor(Origin), fill = factor(Origin)), alpha = 0.3)
```

Fitting Regression Model with Dummy Variables

```
manager2 <- manager %>%
  mutate(Origin.External =  (Origin == 'External'))
```

Create a dummy variable Origin that

$$\text{Origin} = \begin{cases} 1 & \text{if external} \\ 0 & \text{if internal} \end{cases}$$

Same slope

Assume no interaction between the two predictor:

$$\text{Rating} = b_0 + b_{11}\text{Salary} + b_{12}\text{Origin}$$

Specifically, for internal manager, where Origin $= 0$, so that

$$\text{Rating}_{in} = b_0 + b_{11}\text{Salary}$$

and for external managers, where Origin $= 1$, so that

$$\text{Rating}_{ex} = b_0 + b_{11}\text{Salary} + b_{12} = (b_0 + b_{12}) + b_{11}\text{Salary}$$

The coefficient b_{12} is essentially the difference of intercepts between the two models.

```r
par_table(lm(Manager.Rating ~ Salary..US.000. + Origin.External, manager2))
```

```
##                         Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)          -1.5854927 0.655478620 -2.418832 1.679282e-02
## Salary..US.000.       0.1074781 0.009648748 11.139068 2.812871e-21
## Origin.ExternalTRUE -0.5149659 0.209029114 -2.463608 1.490817e-02
##                            2.5 %     97.5 %
## (Intercept)          -2.88087140 -0.2901140
## Salary..US.000.       0.08840989  0.1265462
## Origin.ExternalTRUE -0.92805616 -0.1018756
```

Different slopes

In the last example when we fitted the regression model with the categorical variable, we made an assumption that the model for each category has the same slope. The assumption may not be true. In this section, we are going to fit models that do not impose this restriction.

This time we add the interaction between the two predictor:

$$\text{Rating} = b_0 + b_{11}\text{Salary} + b_{12}\text{Origin} + b_{13}(\text{Salary} \cdot \text{Origin})$$

Specifically, for internal manager, where Origin $= 0$, so that

$$\text{Rating}_{in} = b_0 + b_{11}\text{Salary}$$

and for external managers, where Origin $= 1$, so that

$$\text{Rating}_{ex} = b_0 + b_{11}\text{Salary} + b_{12} + b_{13}\text{Salary} = (b_0 + b_{12}) + (b_{11} + b_{13})\text{Salary}$$

The coefficient b_{12} is still the difference of intercepts between the two models. Additionaly, b_{13} is the difference of slopes between the two models.

Note that to fit a model with interactions, `+` is changed to `*`

```r
model_man3 <- lm(Manager.Rating ~ Salary..US.000. *  # '*'
                 Origin.External, manager2)
par_table(model_man3)
```

```
##                                        Estimate Std. Error    t value
## (Intercept)                         -1.693524015 0.87007165 -1.9464190
## Salary..US.000.                      0.109092936 0.01289302  8.4613940
## Origin.ExternalTRUE                 -0.243417473 1.44723009 -0.1681954
## Salary..US.000.:Origin.ExternalTRUE -0.003701752 0.01952042 -0.1896348
##                                         Pr(>|t|)       2.5 %     97.5 %
## (Intercept)                         5.352450e-02 -3.41308634 0.02603831
## Salary..US.000.                     2.534952e-14  0.08361187 0.13457400
## Origin.ExternalTRUE                 8.666623e-01 -3.10364437 2.61680943
## Salary..US.000.:Origin.ExternalTRUE 8.498587e-01 -0.04228086 0.03487735
```

Principle of Marginality

Check the assumption of equal error variance

```
manager2 %>%
  ggplot(aes(x = Origin, y = model_man3$residuals)) +
  geom_point(aes(col = Origin)) +
  geom_boxplot(aes(fill = Origin))
```