Simple Regression

MSBA7002: Business Statistics

Contents

Introduction

Readings:

- ISLR, Chapter 2/3

Objectives:

1. Case Study: Billy Beane
2. Exploratory Data Analysis (EDA)
3. Simple Regression (a quick review)
    - Model specification
    - LS estimates and properties
    - R-squared and RSE
    - Confidence intervals for coefficients
    - Prediction intervals
    - Model diagnoses
    - Caution about reverse regression
    - Violations of model assumptions
4. Appendices

Note: We hide all the outputs of R chunks by setting not to evaluate them in the global setting chunk (line 19-23). You can set `eval = TRUE` in the global setting to evaluate and output all the R chunk results.

Case Study: Billy Beane

Will a team perform better when they are paid more? Is Billy Beane (Oakland A's GM) worth 12 million dollars as predicted?

Watch movie trailer: MoneyBall https://youtu.be/-4QPVo0UIzc

Read an article: Billion dollar Billy Beane http://fivethirtyeight.com/features/billion-dollar-billy-beane/ (FiveThirtyEight)

- Data: `MLPayData_Total.csv`, consisting of winning records and payrolls of all 30 Major League teams from 1998 to 2014. There are 162 games in each season.
- payroll: total pay from 2000 to 2014 in **billion dollars**
- avgwin: average winning percentage for the span of 2000 to 2014
- p2014: total pay in 2014 in million
- X2014: number of games won in 2014
- X2014.pct: winning percentage in 2014

Goal:

1. How does the payroll relate to the performance (avgwin)?
2. Is Billy Beane worth 12 million dollars as argued in the article?
3. Given a payroll of .84, what would be the **mean** winning percentage; what would we expect the meaning percentage to be for such **A** team?

   (Oakland A's: payroll=.84, avgwin=.54; Red Sox: payroll=1.97, avgwin=.55)

Exploratory Data Analysis (EDA)

Read data

One of the most important aspects of using R is to know how to import data. Most of the data is available in a table form as a .csv file already. So the simplest way to do so is to use read.csv or read.table if the data is available in a table format.

What is the current working directory? R will find files or save files to the working directory.

```
# getwd()
# dir <- "/Users/haipeng/Dropbox/MSBA7002/Data"  # my data folder, replace with yours
# setwd(dir)   #same as  setwd("/Users/haipeng/Dropbox/MSBA7002/Data")
# getwd()
```

I put the data in the same folder as the markdown .Rmd file, and use relative path to read data.

```
datapay <- read.csv("MLPayData_Total.csv", header=T)

# You can also use the whole path to read in the data. In my case,
# datapay <- read.csv("/Users/haipeng/Dropbox/MSBA7002/Data/MLPayData_Total.csv", header=T)
```

Before doing any analysis, it is **A MUST** that we take a look at the data:

- Understand the variables: unit, format, unusual values, etc.
- Summary statistics: mean, sd, cor, etc.
- Displaying the data: histogram, scatter plot, pairwise scatter plots, etc.

Find the structure of the data and have a quick summary

```
class(datapay)
str(datapay) # make sure the variables are correctly defined.
summary(datapay) # a quick summary of each variable
```

Get a table view

This command will show the data frame in a RStudio window for easy viewing.

```
View(datapay)
```

Look at the first six rows or first few rows

```
head(datapay) # or tail(datapay)
head(datapay, 2) # first two rows
```

Find the size of the data or the numbers or rows and columns

```
dim(datapay)
```

Get the variable names

```
names(datapay)
```

Work with a subset which includes relevant variables

Payroll's first row

```r
datapay[1, 1] # first: row; second: var's
datapay[1, "payroll"]
```

Get the payroll column in two ways.

1. Base `R`

```r
datapay$payroll # base R
```

2. `dplyr`

```r
datapay %>% select(payroll) # dplyr
```

Get the first x number of columns.

```r
datapay[, 1] # Base R
datapay %>% select(1) #DPLYR
```

```r
datapay[, c(1:3)] # first three columns
datapay %>% select(1:3) #DPLYR
datapay %>% select(payroll:Team.name.2014) #DPLYR
```

Rename Variables

Two ways to do this, Base `R` or `dplyr`, both shown below

```r
datapay <- datapay %>% rename(team = Team.name.2014) # dplyr
names(datapay)[3] <- "team" # Base R
```

Check number of NAs!

First we use Base `R` to get the number of NAs:

```r
sum(is.na(datapay))
```

Next, the `dplyr` way to find out the number of rows with NAs.

```r
datapay %>%
  filter(!complete.cases(.)) %>% summarize(n())
```

Are they the same?

Lastly, we show another way that breaks it down by variable name. The function `sapply` takes a data.frame and function as its inputs. It then maps the input function to each column of the data.frame.

```r
sapply(datapay, function(x) sum(is.na(x)))
```

!!! The number of the NAs might be less than the total number of missing data. Because the missing ones might not be coded as "NA", e.g. they can be space.

Create Subset

We want to create a subset of the original dataset without X1998 to X2014 and with the rest of the col names sorted.

Base `R` way

```r
data1 <- datapay[, -(21:37)] # take X1998 to X2014 out
data2 <- data1[, sort(names(data1))] # sort colnames
data2.1 <- data1[, order(names(data1))] # alternative way
```

```
names(data2)
names(data2.1)
```

dplyr way

```
data1 <- datapay %>% select(-21:-37)
data2 <- data1 %>% select(order(names(data1)))
names(data2)
```

Descriptive statistics

We will concentrate on three variables: * payroll * avgwin * team

Base R

```
mean(datapay$payroll)
sd(datapay$payroll)
quantile(datapay$payroll)
max(datapay$payroll)
```

dplyr

```
datapay %>% select(payroll) %>%
  summarise(
    mean = mean(payroll),
    sd   = sd(payroll),
    max = max(payroll),
    "0%" = quantile(payroll)[1],
    "25%" = quantile(payroll)[2],
    "50%" = quantile(payroll)[3],
    "75%" = quantile(payroll)[4],
    "100%" = quantile(payroll)[5]
  )
```

Displaying variables

Base R plots

```
hist(datapay$payroll, breaks=5)
hist(datapay$payroll, breaks=10, col="blue") # make a larger number of classes to see the details
```

ggplot plots

```
ggplot(datapay) + geom_histogram(aes(x = payroll), bins = 5) +
  labs( title = "Histogram of Payroll", x = "Payroll" , y = "Frequency")

ggplot(datapay) + geom_histogram(aes(x = payroll), bins = 10, fill = "blue") +
  labs( title = "Histogram of Payroll", x = "Payroll" , y = "Frequency")
```

Find the team with the max payroll

Base R

```
datapay$Team.name.2014[which.max(datapay$payroll)]
datapay$Team.name.2014[which.min(datapay$payroll)]
boxplot(datapay$payroll)
```

```
# To rank teams by payroll
datapay[order(datapay$payroll, decreasing = T), "team"]
```

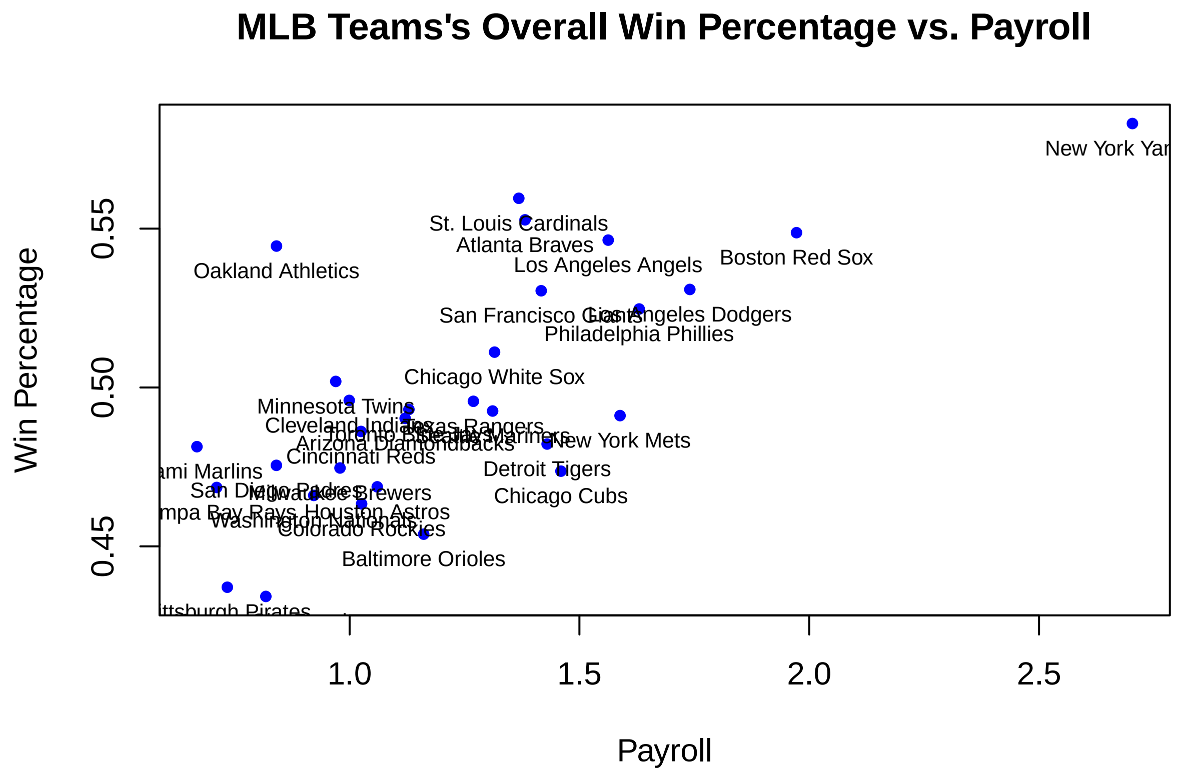
dplyr

```
datapay %>% select(team,payroll) %>% filter(payroll == max(payroll))

datapay %>% select(team,payroll) %>% filter(payroll == min(payroll))

ggplot(datapay) +
  geom_boxplot(aes(x="", y=payroll)) +
  labs(title="Boxplot of Payroll", x="")
```

Explore the relationship between `payroll`, x, and `avgwin`, y.

Make a scatter plot Base `R` way

```
plot(datapay$payroll, datapay$avgwin,
   pch  = 16,
   cex  = 0.8,
   col  = "blue",
   xlab = "Payroll",
   ylab = "Win Percentage",
   main = "MLB Teams's Overall Win Percentage vs. Payroll")
text(datapay$payroll, datapay$avgwin, labels=datapay$team, cex=0.7, pos=1) # label all points
```



`ggplot` way

```
ggplot(datapay) +
  geom_point(aes(x = payroll, y = avgwin), color = "blue") +
  # geom_text(aes(x = payroll, y = avgwin, label=team)) + # with team names shown
  labs(title = "MLB Teams's Overall Win Percentage vs. Payroll", x = "Payroll", y = "Win Percentage")
```

`ggplot` with the least squares (LS) regression line added

```
ggplot(datapay, aes(x = payroll, y = avgwin)) +
  geom_point() +
  geom_smooth(method="lm", se = F,color = "red") +
  geom_hline(aes(yintercept = mean(avgwin)), color = "blue") +
  annotate(geom="text", x=0.8409340, y=0.5445067, label="Oakland A's",color="green", vjust = -1) +
  annotate(geom="text", x=1.9723587, y=0.5487172, label="Red Sox",color="red", vjust = -1)
```

Simple Linear Regression

Let the response y_i be the `avgwin` and the explanatory variable x_i be `payroll` ($i = 1, \dots, n = 30$).

Linear model assumptions

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \qquad \epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

To be specific,

- Linearity:

$$\mathbf{E}(y_i | x_i) = \beta_0 + \beta_1 x_i$$

- Homoscedasticity:

$$\mathbf{Var}(y_i | x_i) = \sigma^2$$

- Independence:

$$\epsilon_i \quad \text{independent from each other}$$

- Normality:

$$\epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Indpendence + Normality suggest that

$$\epsilon_i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$$

or

$$y_i \overset{iid}{\sim} \mathcal{N}(\beta_0 + \beta_1 x_i, \sigma^2)$$

- Parameters of interest
  - intercept: β_0
  - slope: β_1
  - variance: σ^2

OLS estimates

- The ordinary least squares (OLS) estimators $\hat{\beta}_0$ and $\hat{\beta}_1$ are obtained by minimizing the sum of squared errors (SSE):

$$\min_{b_0,\, b_1} \sum_{i=1}^{n} (y_i - b_0 - b_1 x_i)^2.$$

The expressions are

$$\hat{\beta}_1 = r \frac{s_y}{s_x}, \qquad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \cdot \bar{x}.$$

The function `lm()` will be used extensively. This function solves the minimization problem above.

Below we use `avgwin` as the dependent variable y and `payroll` as our x.

As we can see from the following output, this function outputs a list of many statistics. We will define these statistics later.

```
myfit0 <- lm(avgwin ~ payroll, data=datapay)
names(myfit0)
```

```
##  [1] "coefficients"  "residuals"     "effects"       "rank"
##  [5] "fitted.values" "assign"        "qr"            "df.residual"
##  [9] "xlevels"       "call"          "terms"         "model"
```

We can also view a summary of the `lm()` output by using the `summary()` command.

```
summary(myfit0)   # it is another object that is often used
results <- summary(myfit0)
names(results)
```

Its also useful to notice the outputs of `myfit0` and `summary(myfit0)` are different

```
myfit0
```

```
##
## Call:
## lm(formula = avgwin ~ payroll, data = datapay)
##
## Coefficients:
## (Intercept)      payroll
##     0.42260      0.06137
```

As the OLS estimates, $\hat{\beta}_0 = 0.4225976$ and $\hat{\beta}_1 = 0.0613684$. It follows that

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i = 0.42260 + 0.06137 \cdot x_i.$$

To estimate the `avgwin` of the Oakland Athletics whose `payroll` is 0.840934,

$$\hat{y}_{\text{Oakland Athletics}} = 0.42260 + 0.06137 \times 0.840934 = 0.4742044.$$

The residual for the estimate is

$$\hat{\epsilon}_i = y_i - \hat{y}_i = y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i).$$

For the Oakland Athletics, the real `avgwin` is 0.840934. So the residual is

$$\hat{\epsilon}_{\text{Oakland Athletics}} = y_{\text{Oakland Athletics}} - \hat{y}_{\text{Oakland Athletics}} = 0.5445067 - 0.4742044 = 0.0703023.$$

Scatter plot with the LS line added

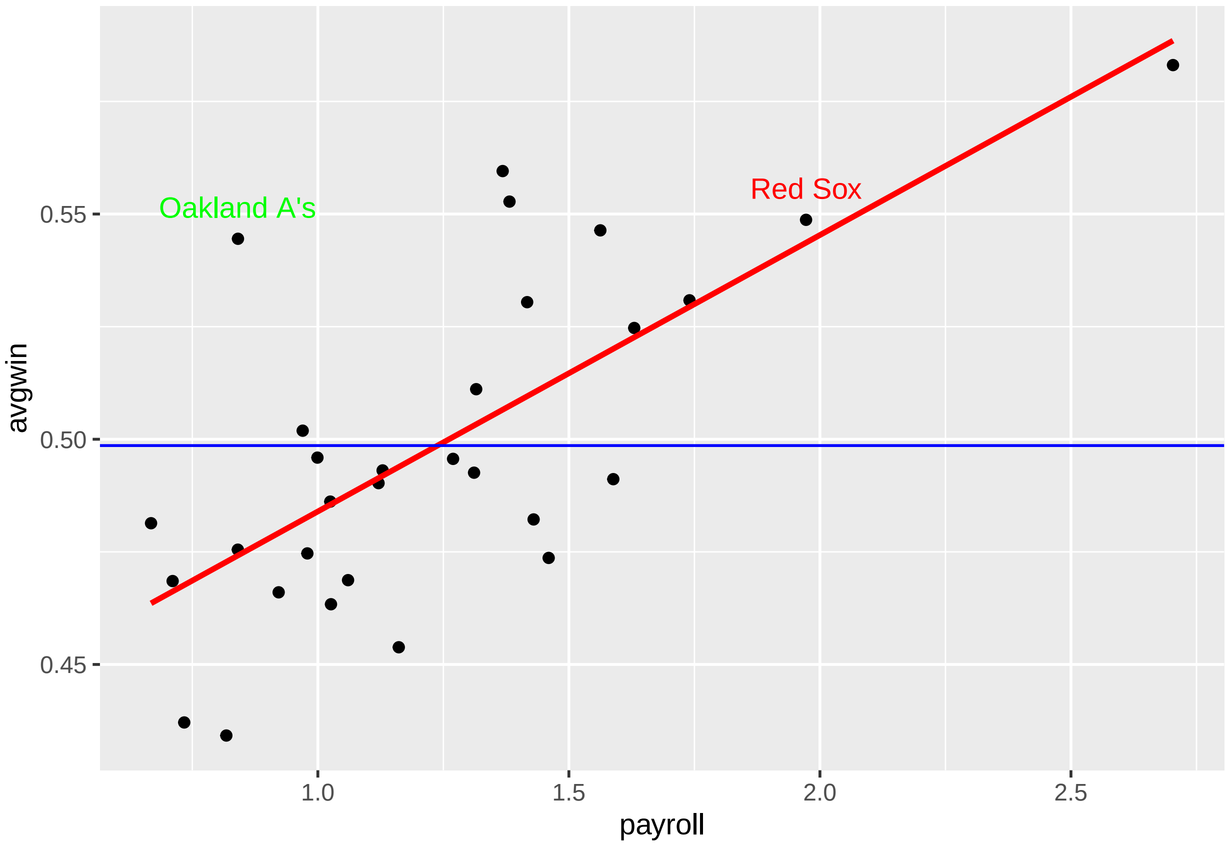
Base `R`

```
plot(datapay$payroll, datapay$avgwin,
     pch  = 16,
     xlab = "Payroll",
     ylab = "Win Percentage",
     main = "MLB Teams's Overall Win Percentage vs. Payroll")
abline(myfit0, col="red", lwd=4)         # many other ways.
abline(h=mean(datapay$avgwin), lwd=5, col="blue") # add a horizontal line, y=mean(y)
```

`ggplot`

```r
ggplot(datapay, aes(x = payroll, y = avgwin)) +
  geom_point() +
  geom_smooth(method="lm", se = F,color = "red") +
  geom_hline(aes(yintercept = mean(avgwin)), color = "blue") +
  annotate(geom="text", x=0.8409340, y=0.5445067, label="Oakland A's",color="green", vjust = -1) +
  annotate(geom="text", x=1.9723587, y=0.5487172, label="Red Sox",color="red", vjust = -1)
```

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



Here are a few rows that show the fitted values from our model:

```r
data.frame(datapay$team, datapay$payroll, datapay$avgwin, myfit0$fitted,
           myfit0$res)[15:25, ] # show a few rows
```

HERE is how the article concludes that Beane is worth as much as the GM in the Red Sox. By looking at the above plot, the Oakland A's win pct is more or less the same as that of the Red Sox, so based on the LS equation, the team should have paid 2 billion. Do you agree?????

Goodness of Fit: R^2

How well does the linear model fit the data? We need the following statistics from lm().

Residual Sum of Squares (RSS) The least squares approach chooses $\hat{\beta}_0$ and $\hat{\beta}_1$ to minimize the RSS, which is defined as:

$$RSS = \sum_{i=1}^{n} \hat{\epsilon}_i^2 = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2$$

```
  myfit0 <- lm(avgwin~payroll, data=datapay)
  RSS <- sum((myfit0$res)^2) # residual sum of squares
  RSS
```

Mean Squared Error (MSE), Residual Standard Error (RSE) i.e. Root Mean Squared Error (RMSE) The error variance σ^2 is estimated by the Mean Sum of Squares (MSE). For simple regression, MSE is defined as:

$$MSE = \frac{RSS}{n-2}.$$

The error standard deviation σ is estimated by the Residual Standard Error (RSE) or Root Mean Squared Error (RMSE). For simple regression, RSE is defined as:

$$RSE = \sqrt{MSE} = \sqrt{\frac{RSS}{n-2}}.$$

```
sqrt(RSS/myfit0$df)
```

```
summary(myfit0)$sigma
```

Total Sum of Squares (TSS) TSS measures the total variance in the response Y, and can be thought of as the amount of variability inherent in the response before the regression is performed. In contrast, RSS measures the amount of variability that is left unexplained after performing the regression.

$$TSS = \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

```
  TSS <- sum((datapay$avgwin-mean(datapay$avgwin))^2) # total sum of sqs
  TSS
```

R^2 R^2 measures the proportion of variability in Y that can be explained using X. An R^2 statistic that is close to 1 indicates that a large proportion of the variability in the response has been explained by the regression.

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

```
(TSS-RSS)/TSS     # Percentage reduction of the total errors
```

$(1 - R^2)s_y^2 \approx MSE$.

$R^2 = r^2$ with r being the correlation between X and Y.

```
(cor(datapay$avgwin, myfit0$fit))^2 # Square of the correlation between response and fitted values
```

```
summary(myfit0)$r.squared
```

R^2 captures the usefulness of using x to predict y, and is often used as a measure of the "effectiveness" of a regression.

- Proportion of idiosyncratic variation: $1 - R^2$.

How large does R^2 need to be so that you are comfortable to use the linear model?

Advice: Resist the temptation to think of R^2 in absolute terms

- The value of a regression depends on the value of the information provided.

R^2 is not useful for deciding between regressions when

- The response variables y are different due to transformation.
- The data points are different due to the removal of outliers.

Inference for the coefficients: β_1 and β_0

Most often we care about whether $\beta_1 = 0$. Why?

Under the model assumptions:

1. y_i independent, normally distributed
2. The mean of y given x is linear
3. The variance of y does not depend on x,

the LS estimates $\hat\beta_1$ and $\hat\beta_0$ have the following properties:

1. Unbiasedness:

$$\mathbf{E}(\hat\beta_1) = \beta_1, \qquad \mathbf{E}(\hat\beta_0) = \beta_0;$$

2. Normality

$$\hat\beta_1 \sim \mathcal{N}(\beta_1, \mathbf{Var}(\hat\beta_1)), \qquad \hat\beta_0 \sim \mathcal{N}(\beta_0, \mathbf{Var}(\hat\beta_0))$$

where

$$\mathbf{Var}(\hat\beta_1) = \sigma^2 \frac{1}{\sum_{i=1}^n (x_i - \bar x)^2},$$

and

$$\mathbf{Var}(\hat\beta_0) = \sigma^2 [\frac{1}{n} + \frac{\bar x^2}{\sum_{i=1}^n (x_i - \bar x)^2}].$$

Tests and CI for the coefficients t-interval and t-test can be constructed using the above results.

For example, the 95% confidence interval (CI) for β approximately takes the form

$$\hat\beta \pm 2 \cdot SE(\hat\beta).$$

We can see below that the $SE(\beta)$ is included in the summary output.

```
summary(myfit0)
```

The `confint()` function returns the confidence interval (95% by default).

```r
confint(myfit0)
confint(myfit0, level = 0.99)
```

Confidence and Prediction Intervals for the mean reponse and individual response

Confidence Interval We use a confidence interval to quantify the uncertainty surrounding the mean of the response, i.e., `avgwin`. For example, for teams like the Oakland A's whose `payroll`=.841, a $100 * (1-\alpha)\%$ Confidence Interval for the mean of response `avgwin` is

$$\hat{y}_{|x=.841} \pm t_{(\alpha/2,n-2)} \times \sqrt{MSE \times \left(\frac{1}{n} + \frac{(.841-\bar{x})^2}{\sum(x_i-\bar{x})^2}\right)}.$$

```r
new <- data.frame(payroll=c(.841))  #new <- data.frame(payroll=c(1.24))
CImean <- predict(myfit0, new, interval="confidence", se.fit=TRUE)
CImean
```

Prediction Interval A prediction interval can be used to quantify the uncertainty surrounding `avgwin` for a **particular** team.

$$\hat{y}_{|x} \pm t_{(\alpha/2,n-2)} \times \sqrt{MSE \times \left(1 + \frac{1}{n} + \frac{(x-\bar{x})^2}{\sum(x_i-\bar{x})^2}\right)}$$

We now do 95% & 99% prediction intervals for a future y given $x = .841$:

```r
new <- data.frame(payroll=c(.841))
CIpred <- predict(myfit0, new, interval="prediction", se.fit=TRUE)
CIpred

CIpred_99 <- predict(myfit0, new, interval="prediction", se.fit=TRUE, level=.99)
CIpred_99
```

From the output above, a 95% prediction interval varies from .41 to .53 for a team like the Oakland A's. But its `avgwin` is .54. So it is somewhat unusual but not that unusual! A 99% prediction interval would contain the true Oakland winning percentage of .

Read Page 82 of ILSR to fully understand the difference between confindence and prediction intervals.

Model diagnoses

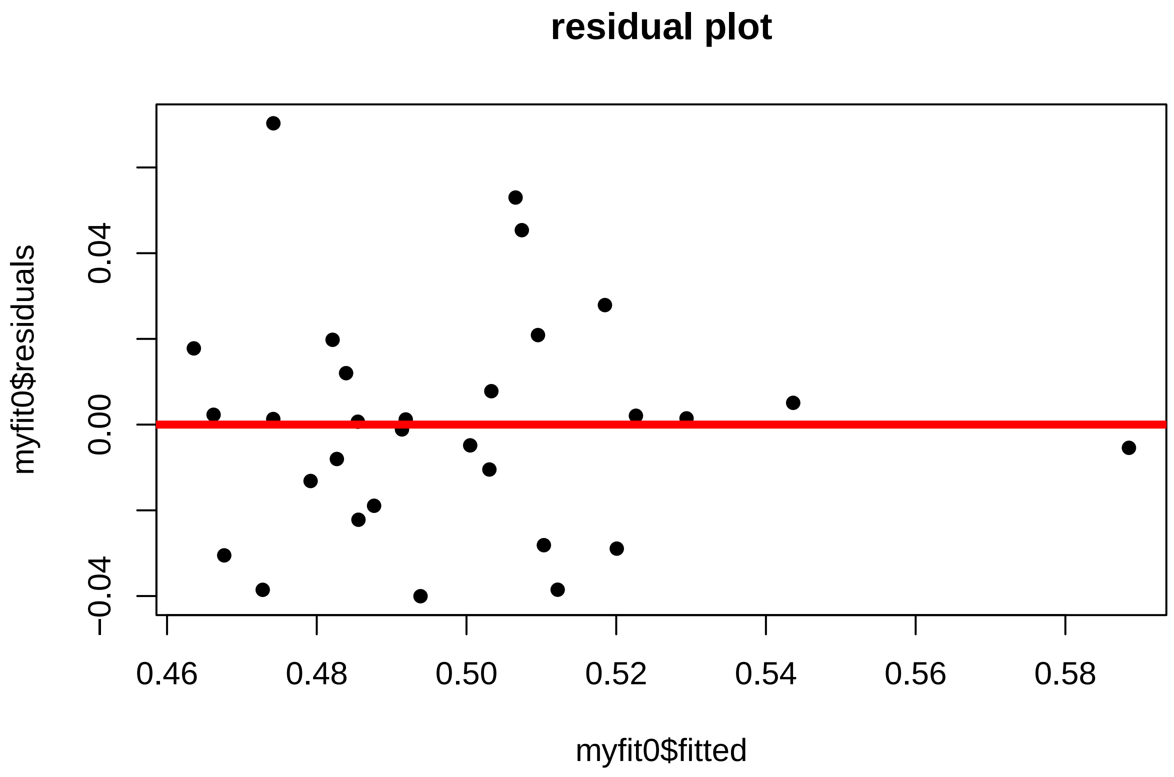
Before using the model, we need to check the model assumptions in the following steps:

1. Check **linearity** first; if linearity is satisfied, then
2. Check **homoscedasticity**; if homoscedasticity is satisfied, then
3. Check **normality**; if normality is satisfied, then
4. Check **independence**. This makes more sense when data are collected across time.

Residual plot We plot the residuals against the fitted values to

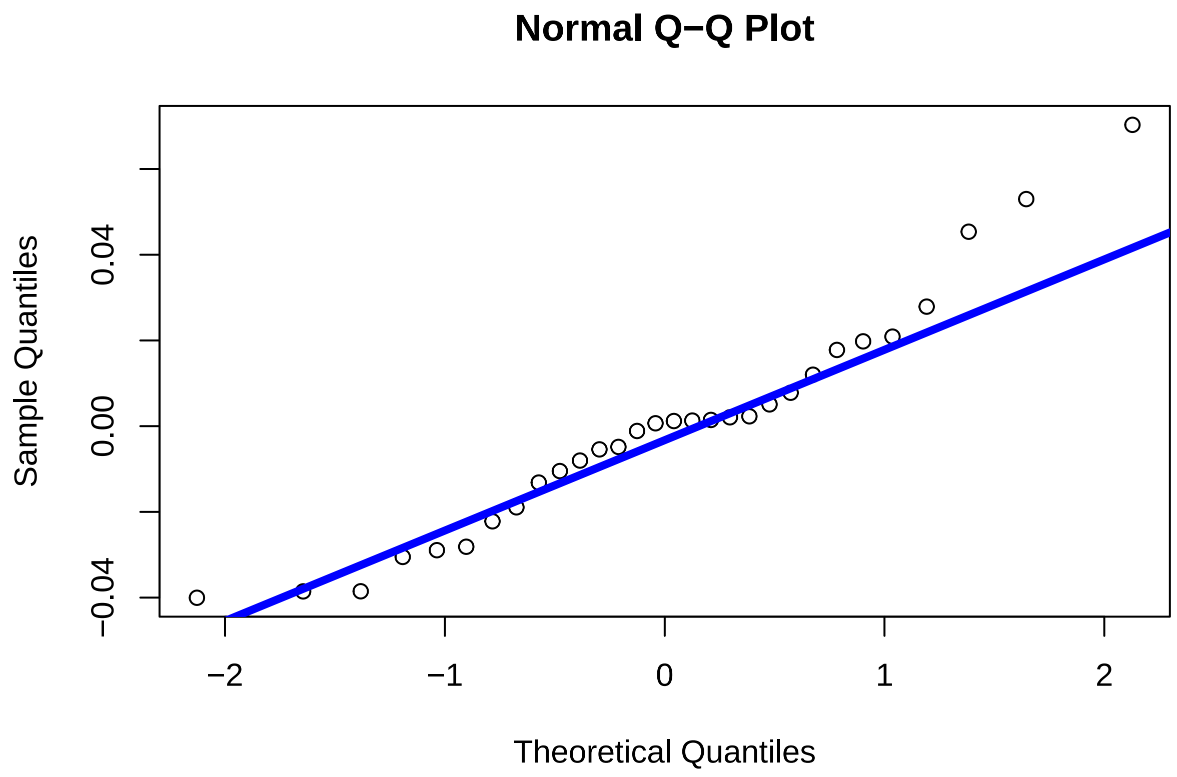
- check **linearity** by checking whether the residuals follow a symmetric pattern with respect to $h = 0$.
- check **homoscedasticity** by checking whether the residuals are evenly distributed within a band.

```
plot(myfit0$fitted, myfit0$residuals,
     pch  = 16,
     main = "residual plot")
abline(h=0, lwd=4, col="red")
```



Check normality We look at the qqplot of the residuals to check normality.

```
  qqnorm(myfit0$residuals)
  qqline(myfit0$residuals, lwd=4, col="blue")
```



Use qqPlot in library(car) to produce qqplot with envelop

Reverse Regression

Warning: one can not solve for x using the above LS equation. We will show this now.

We first plot our original model:

$$y_{avgwin} = 0.42260 + 0.06137 \cdot x_{payroll}.$$

```
par(mgp=c(1.8,.5,0), mar=c(3,3,2,1))
plot(datapay$payroll, datapay$avgwin,
     pch  = 16,
     xlab = "Payroll",
     ylab = "Win Percentage",
     main = "MLB Teams's Overall Win Percentage vs. Payroll")
abline(lm(avgwin ~ payroll, data=datapay), col="red", lwd=4)
```

Now, we reverse the regression and look at the summary output.

```
myfit1 <- lm(payroll~avgwin, data=datapay)
summary(myfit1)
```

```
##
## Call:
## lm(formula = payroll ~ avgwin, data = datapay)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.76735 -0.18705 -0.03714  0.16633  0.78427
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2.7784     0.7697  -3.610  0.00118 **
## avgwin        8.0563     1.5396   5.233 1.47e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.309 on 28 degrees of freedom
## Multiple R-squared:  0.4944, Adjusted R-squared:  0.4763
## F-statistic: 27.38 on 1 and 28 DF,  p-value: 1.469e-05
```

We can now overlay our two regressions:

$$y_{payroll} = -2.7784 + 8.0563 \cdot x_{avgwin},$$

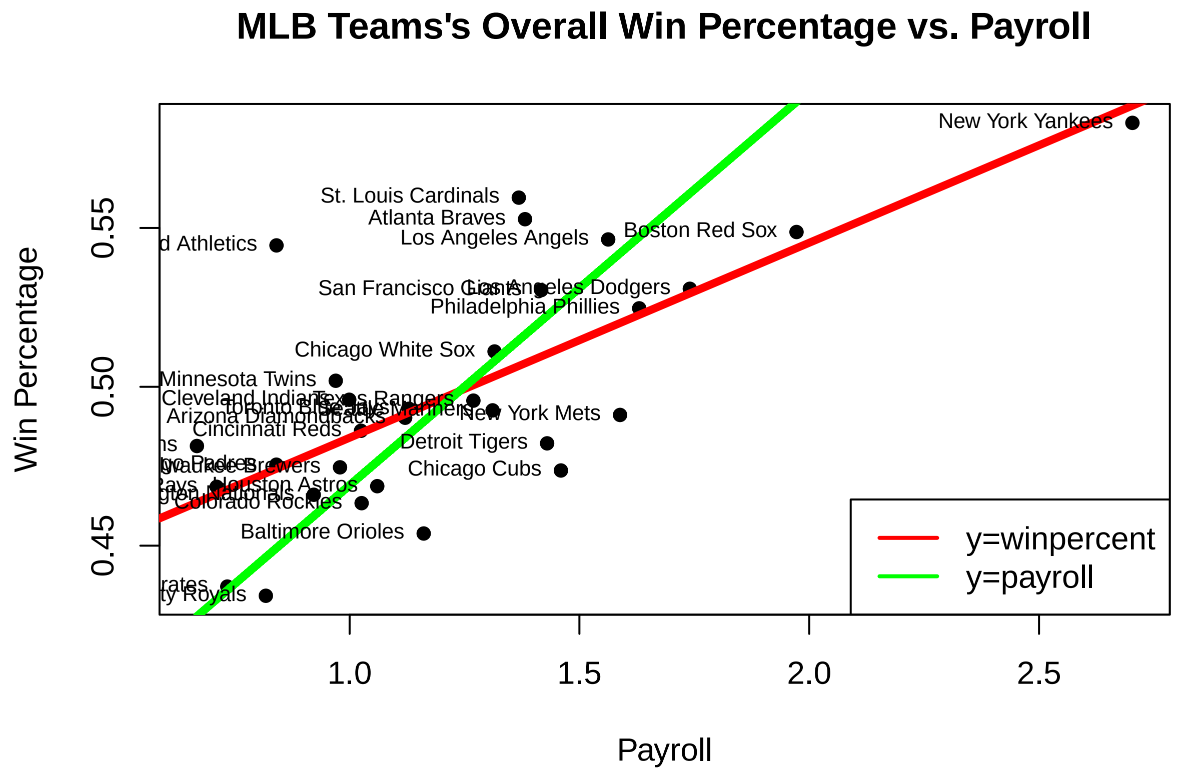
and

$$y_{avgwin} = 0.42260 + 0.06137 \cdot x_{payroll}.$$

Notice that the above is not the same as solving $x_{payroll}$ given $y_{payroll}$ from the first equation which is

$$x_{avgwin} = 1/8.0563 y_{payroll} + 2.7784/8.0563 = 0.344873 + .124 y_{payroll}.$$

```
beta0 <- myfit1$coefficients[1]
beta1 <- myfit1$coefficients[2]
avgwin2 <- (datapay$payroll - beta0) / beta1

plot(datapay$payroll, datapay$avgwin,
     pch  = 16,
     xlab = "Payroll",
     ylab = "Win Percentage",
     main = "MLB Teams's Overall Win Percentage vs. Payroll")
abline(lm(avgwin ~ payroll, data=datapay), col="red", lwd=4)
lines(datapay$payroll, avgwin2, col="green", lwd=4)
legend("bottomright", legend=c("y=winpercent", "y=payroll"),
     lty=c(1,1), lwd=c(2,2), col=c("red","green"))
text(datapay$payroll, datapay$avgwin, labels=datapay$team, cex=0.7, pos=2) # label teams
```



Conclusion: The two lines are not the same!!!!! Explain why.

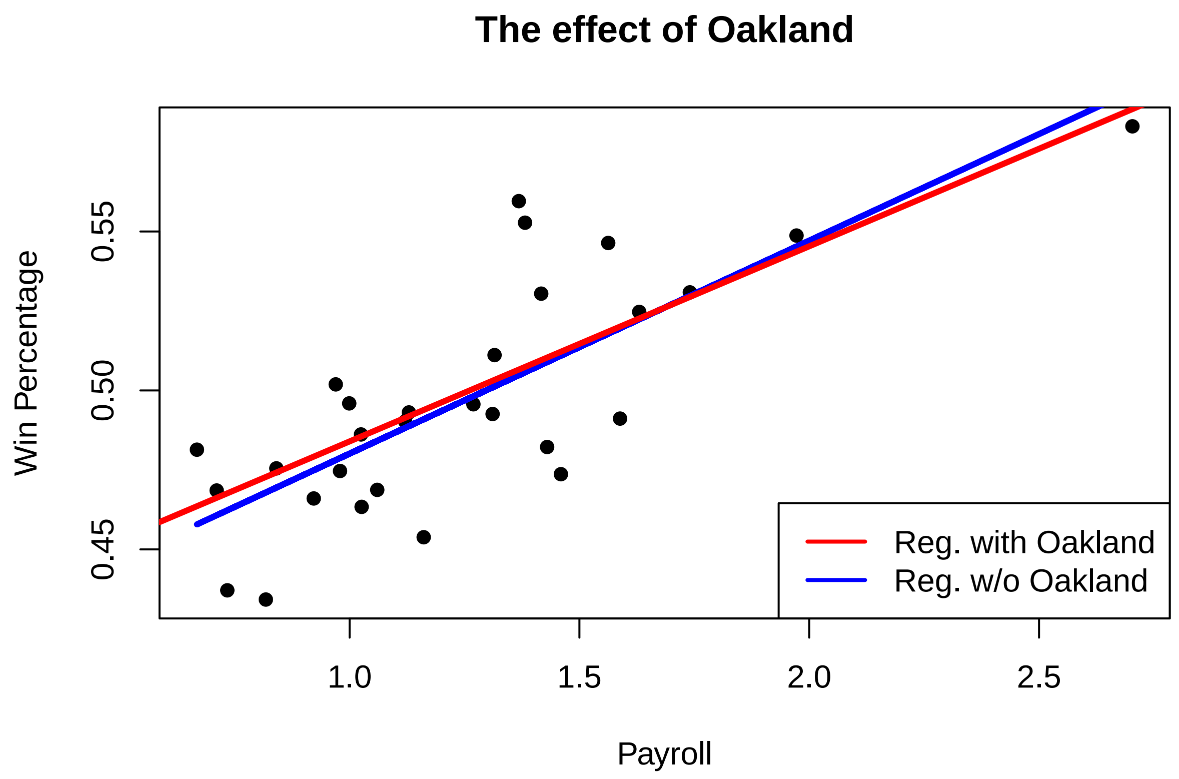
We may also want to get the LS equation w/o Oakland first.

```
subdata <- datapay[-20, ]
myfit2 <- lm(avgwin ~ payroll, data=subdata)
summary(myfit2)
```

```
##
## Call:
## lm(formula = avgwin ~ payroll, data = subdata)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.037273 -0.015430  0.002040  0.009863  0.054755
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.41311    0.01379  29.950  < 2e-16 ***
## payroll      0.06701    0.01044   6.417 7.11e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02364 on 27 degrees of freedom
## Multiple R-squared:  0.604,  Adjusted R-squared:  0.5893
## F-statistic: 41.18 on 1 and 27 DF,  p-value: 7.108e-07
```

The plot below shows the effect of Oakland on our linear model.

```
plot(subdata$payroll, subdata$avgwin,
     pch  = 16,
     xlab = "Payroll", ylab="Win Percentage",
     main = "The effect of Oakland")
lines(subdata$payroll, predict(myfit2), col="blue", lwd=3)
abline(myfit0, col="red", lwd=3)
legend("bottomright", legend=c("Reg. with Oakland", "Reg. w/o Oakland"),
       lty=c(1,1), lwd=c(2,2), col=c("red","blue"))
```



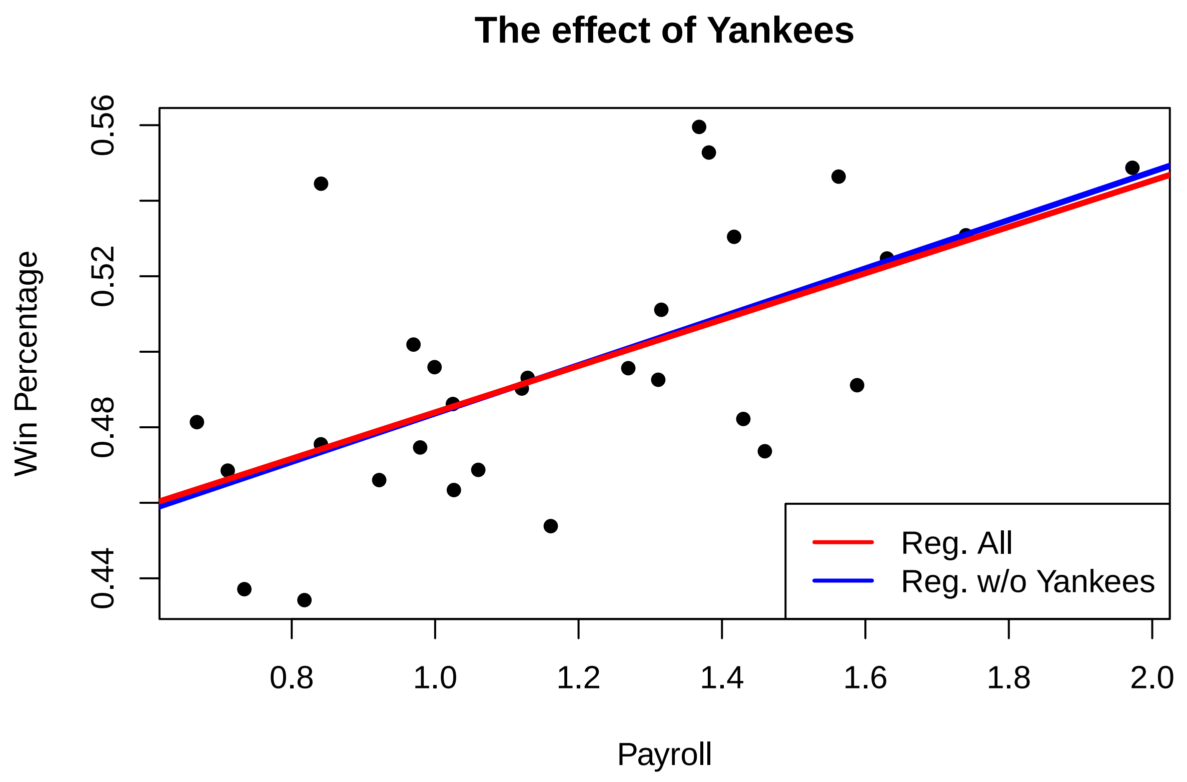
We may also want to check the effect of the Yankees (2.70, .58).

```
subdata1 <- datapay[-19, ]
myfit3 <- lm(avgwin ~ payroll, data=subdata1)
summary(myfit3)
```

```
##
## Call:
## lm(formula = avgwin ~ payroll, data = subdata1)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.040150 -0.018782  0.000696  0.012325  0.071044
##
## Coefficients:
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.41960    0.01929   21.75  < 2e-16 ***
## payroll       0.06405    0.01566    4.09 0.000349 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.02743 on 27 degrees of freedom
## Multiple R-squared:  0.3825, Adjusted R-squared:  0.3597
## F-statistic: 16.73 on 1 and 27 DF,  p-value: 0.0003489
```

```
plot(subdata1$payroll, subdata1$avgwin,
     pch  = 16,
     xlab = "Payroll",
     ylab = "Win Percentage",
     main = "The effect of Yankees")
abline(myfit3, col="blue", lwd=3)
abline(myfit0, col="red", lwd=3)
legend("bottomright", legend=c("Reg. All", "Reg. w/o Yankees"),
       lty=c(1,1), lwd=c(2,2), col=c("red","blue"))
```



Appendix

Appendix 1 - t-distribution vs z-distribution

Difference between z and t with $df = n$. The distribution of z is similar to that t when df is large, say 30.

```r
z <- rnorm(1000)
hist(z, freq=FALSE, col="red", breaks=30, xlim=c(-5,5))
```

Check Normality

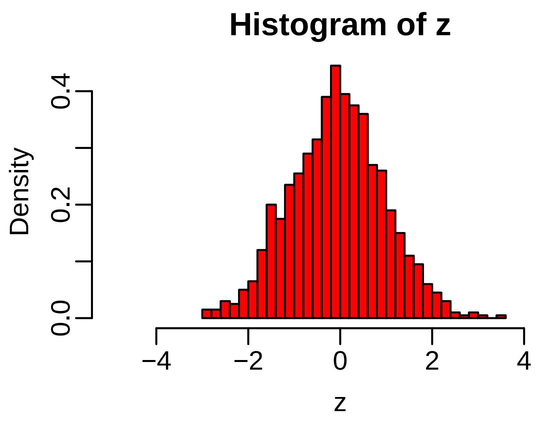
```r
qqnorm(z, pch=16, col="blue", cex=0.7)
qqline(z)   #shift-command-c  to add or to suppress comment
```

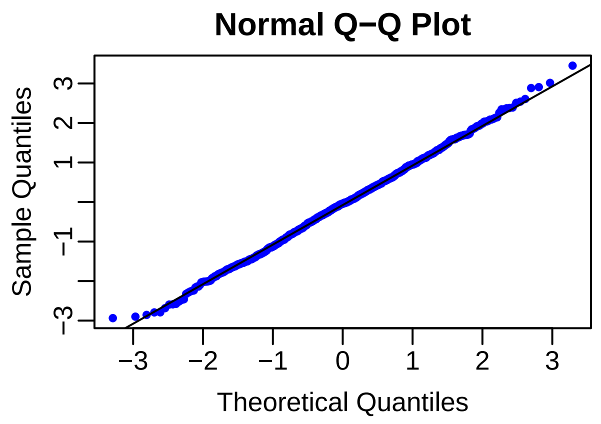
See what a t variable looks like with a degrees of freedom at 10

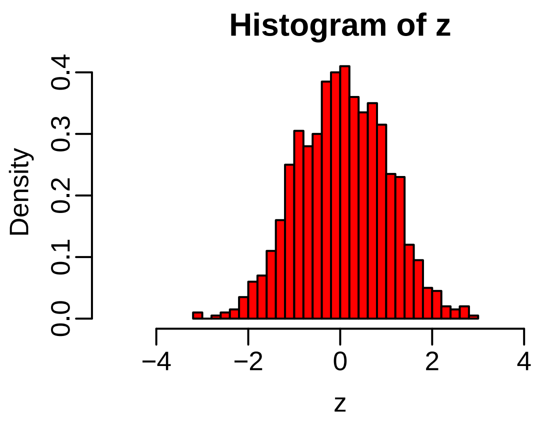
```r
df <- 10 # you may change df
t <- rt(1000, df)   #
hist(t, freq=FALSE, col="blue", breaks=50, xlim=c(-5,5),
     main=paste("Hist of t with df=",df))
```

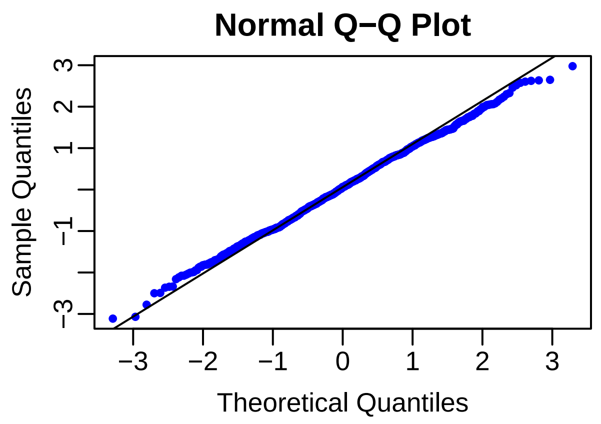
Make a small set of graphs to see the variability of sample from sample

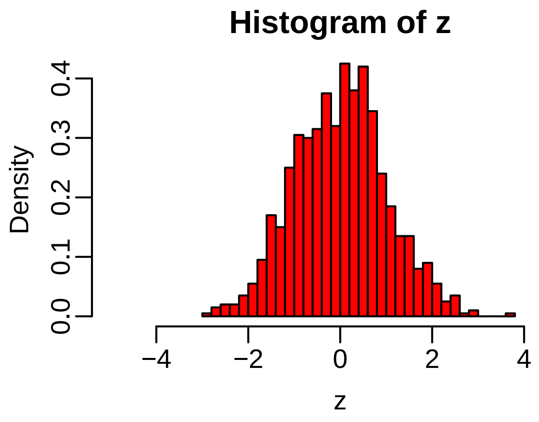
```r
# Put graphs into 2 rows and 2 cols
par(mfrow=c(2,2))

for (i in 1:4){
  z <- rnorm(1000)
  par(mgp=c(1.8,.5,0), mar=c(3,3,2,1))
  hist(z, freq=FALSE, col="red", breaks=30, xlim=c(-5,5))

  qqnorm(z, pch=16, col="blue", cex=0.7)    # check normality
  qqline(z)
}
```

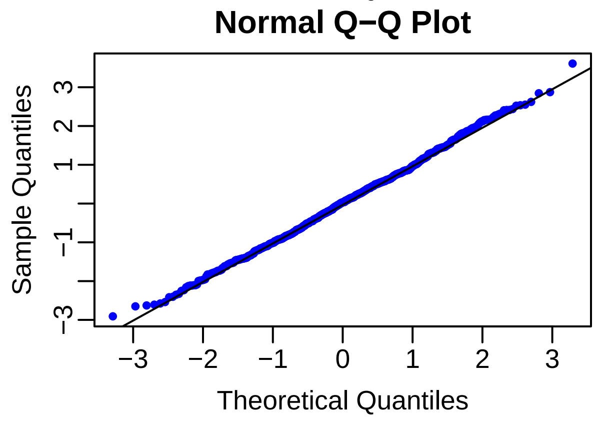

Histogram of z
Density
0.0
0.2
0.4
−4
−2
0
2
4
z

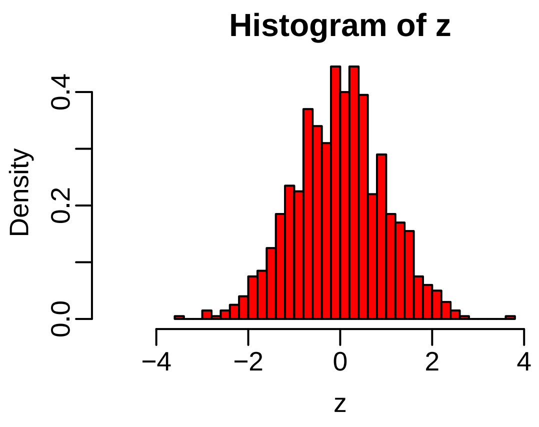

Normal Q−Q Plot
Sample Quantiles
−3
−1
1
2
3
−3
−2
−1
0
1
2
3
Theoretical Quantiles

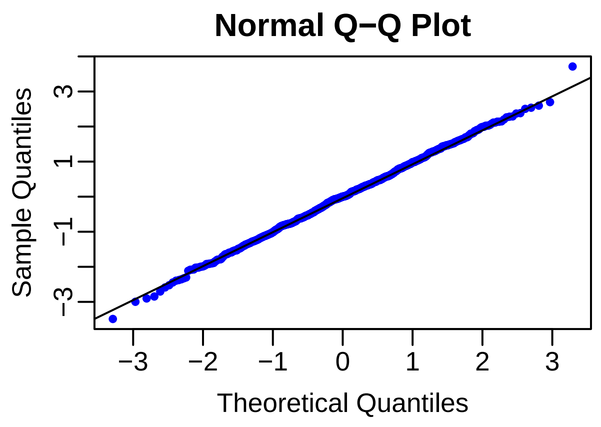

Histogram of z
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
z


Normal Q−Q Plot
Sample Quantiles
−3
−1
1
2
3
−3
−2
−1
0
1
2
3
Theoretical Quantiles


Histogram of z
Density
0.0
0.1
0.2
0.3
0.4
−4
−2
0
2
4
z


Normal Q−Q Plot
Sample Quantiles
−3
−1
1
2
3
−3
−2
−1
0
1
2
3
Theoretical Quantiles

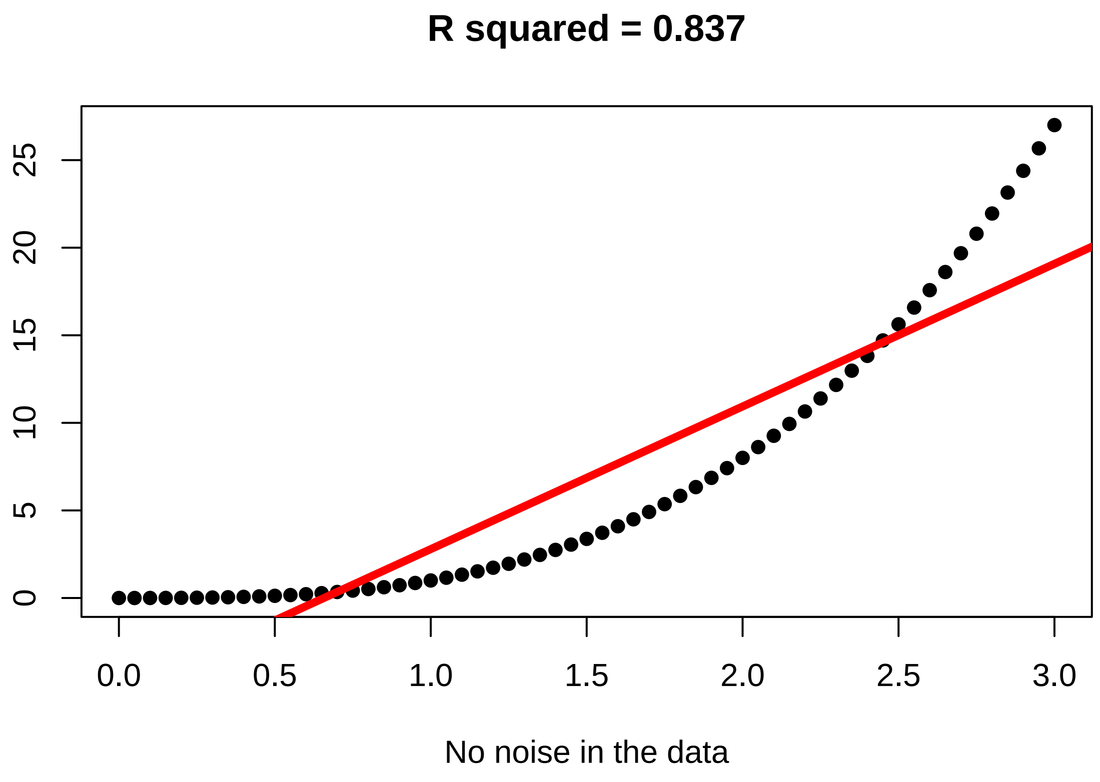

Histogram of z
Density
0.0
0.2
0.4
−4
−2
0
2
4
z


Normal Q−Q Plot
Sample Quantiles
−3
−1
1
3
−3
−2
−1
0
1
2
3
Theoretical Quantiles

Appendix 2 - Investigate R-Squared

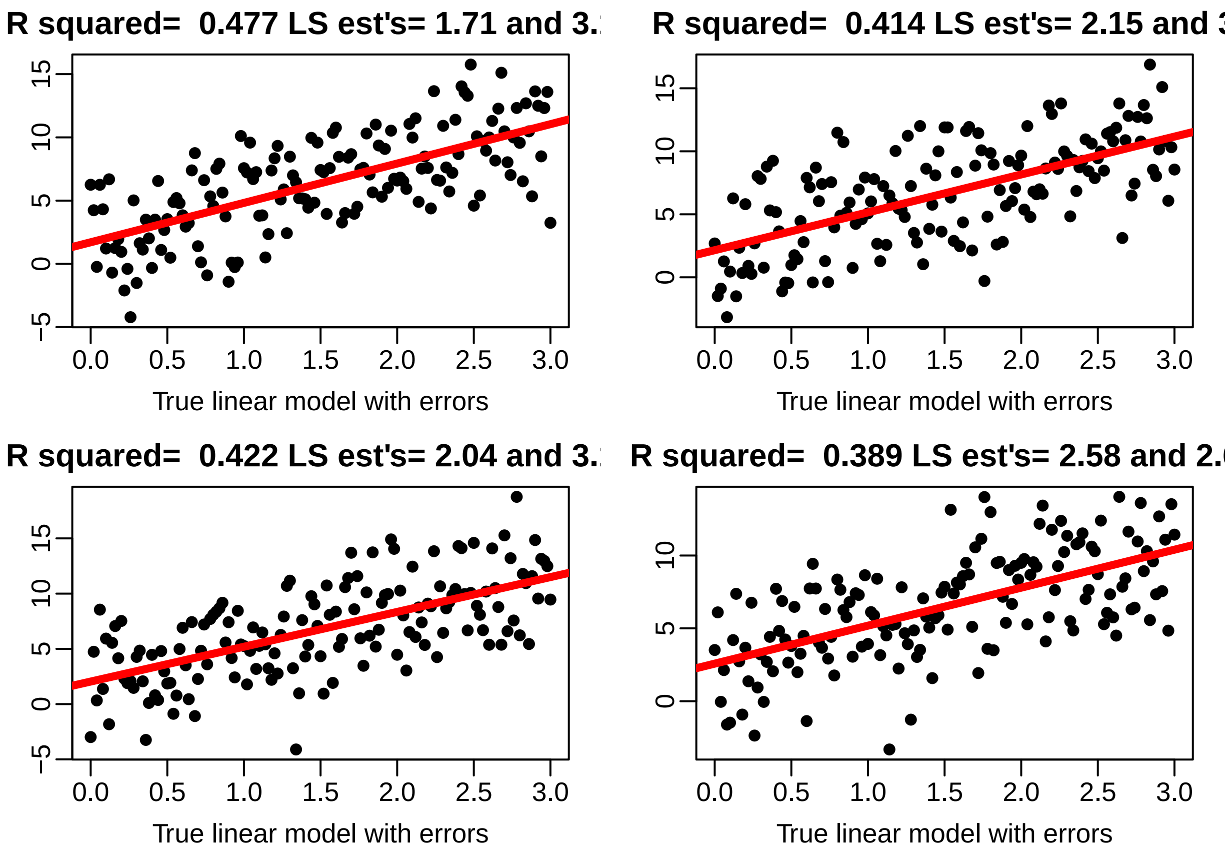
Case 1 A perfect model between X and Y, except that it is not linear: $y = x^3$ with no noise! $R^2 = .837$.

```r
x <- seq(0, 3, by=.05) # or x=seq(0, 3, length.out=61)
y <- x^3 # no variability

myfit <- lm(y ~ x)
myfit.out <- summary(myfit)
rsquared <- myfit.out$r.squared

plot(x, y, pch=16, ylab="",
   xlab = "No noise in the data",
   main = paste("R squared = ", round(rsquared, 3), sep=""))
abline(myfit, col="red", lwd=4)
```



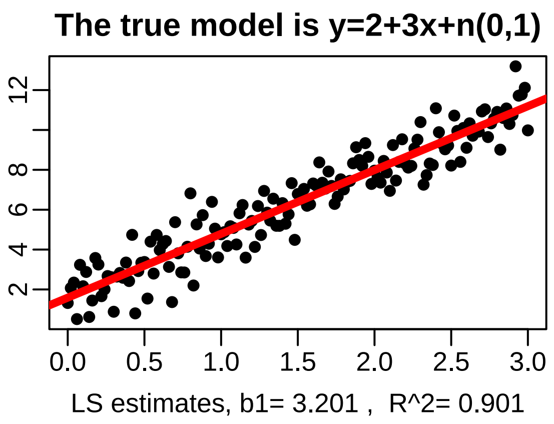
Case II A perfect linear model between X and Y but with noise: $y = 2 + 3x + \epsilon$, where ϵ is iid N(0, var=9). Run this repeatedly.

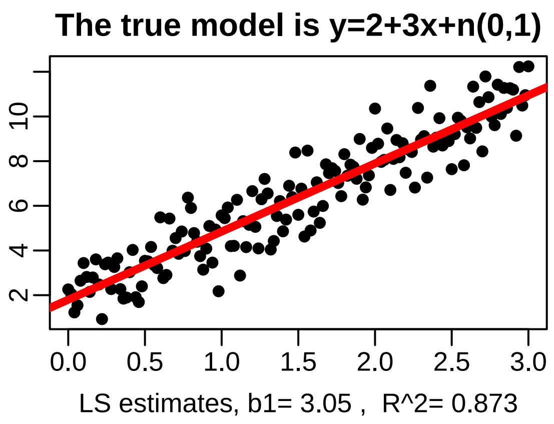
```r
par(mfrow = c(2,2))

for(i in 1:4){
  x <- seq(0, 3, by=.02)
  e <- 3*rnorm(length(x))    # Normal random errors with mean 0 and sigma=3
  y <- 2 + 3*x + 3*rnorm(length(x))

  myfit <- lm(y ~ x)
  myfit.out <- summary(myfit)
  rsquared <- round(myfit.out$r.squared,3)
  hat_beta_0 <- round(myfit$coe[1], 2)
  hat_beta_1 <- round(myfit$coe[2], 2)
  par(mgp=c(1.8,.5,0), mar=c(3,3,2,1))
  plot(x, y, pch=16, ylab="",
       xlab = "True linear model with errors",
       main = paste("R squared= ", rsquared,
                    "LS est's=", hat_beta_0, "and", hat_beta_1))

  abline(myfit, col="red", lwd=4)
}
```

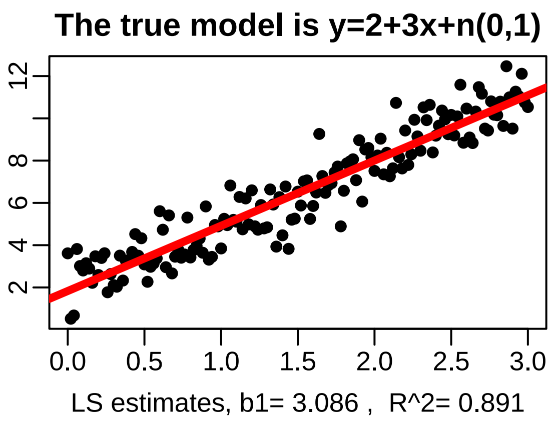


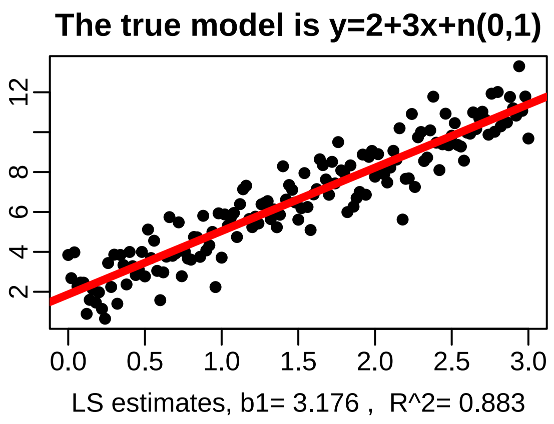
Case III Same as that in Case II, but lower the error standar deviation to $\sigma = 1$.

```
par(mfrow = c(2,2))

for(i in 1:4){
  x <- seq(0, 3, by=.02)
  e <- 3*rnorm(length(x))    # Normal random errors with mean 0 and sigma=3
  y <- 2 + 3*x + 1*rnorm(length(x))

  myfit <- lm(y ~ x)
  myfit.out <- summary(myfit)
  rsquared <- round(myfit.out$r.squared, 3)
  b1 <- round(myfit.out$coe[2], 3)
  par(mgp=c(1.8,.5,0), mar=c(3,3,2,1))
  plot(x, y, pch=16,
       ylab = "",
       xlab = paste("LS estimates, b1=", b1, ",  R^2=", rsquared),
      main = "The true model is y=2+3x+n(0,1)")
  abline(myfit, col="red", lwd=4)
}
```









Appendix 3 - More on Model Diagnoses

What do we expect to see even when all the model assumptions are met?

- a) Variability of the LS estimates $\hat{\beta}$'s
- b) Variability of the R^2's
- c) Model diagnoses: through residuals.

We demonstrate this through a simulation.

Here is a case that all the linear model assumptions are met. Once again everything can be checked by examining the residual plots.

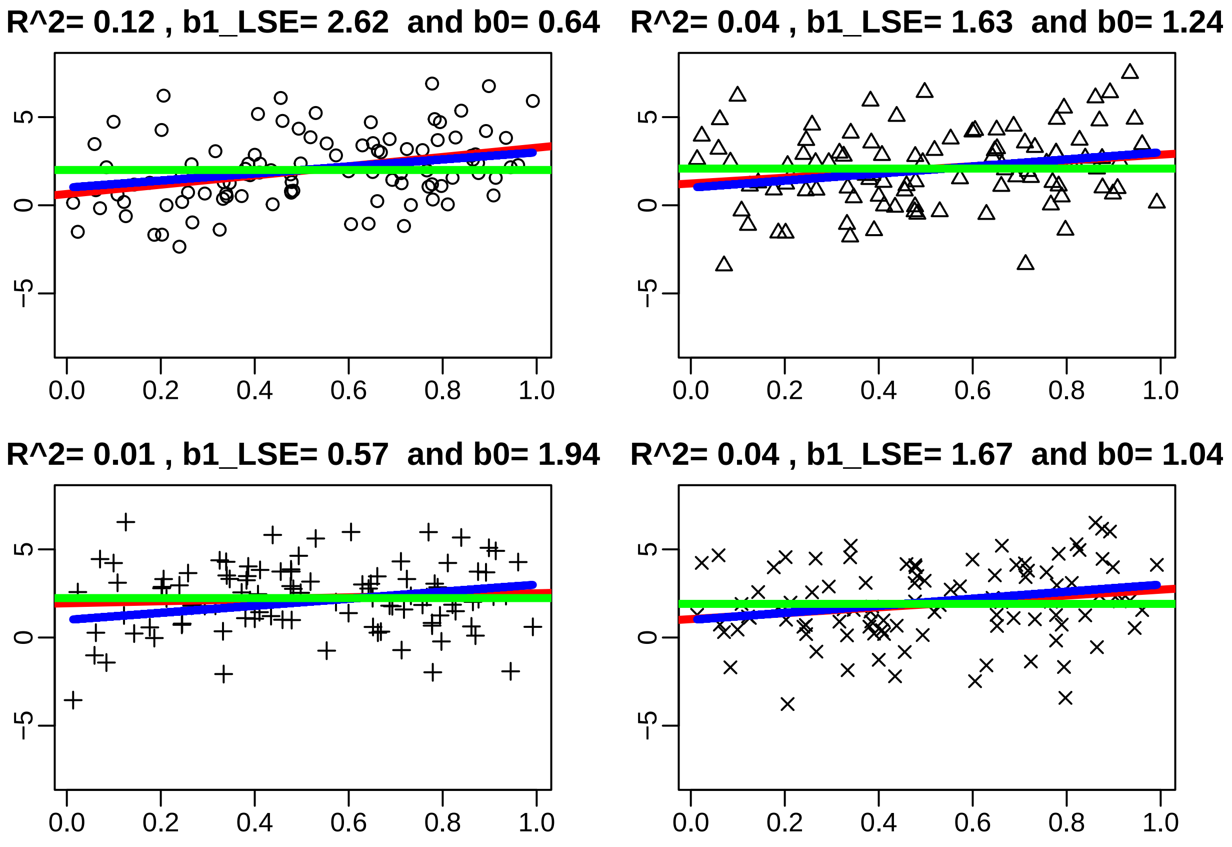
1. Randomize X

```
### Set up the simulations
set.seed(1)
x <- runif(100) # generate 100 random numbers from [0, 1]
```

2. Generate y only each time to see the variability of the LS estimates. The true y is

$$y = 1 + 2x + \mathcal{N}(0, 2^2) \quad i.e. \quad \beta_0 = 1,\ \beta_1 = 2,\ \sigma^2 = 4.$$

```r
# mar: c(bottom, left, top, right) to specify the margin on four sides
# mgp: specify the margin for axis title, axis labels and axis line
par(mfrow=c(2,2), mar=c(2,2,2,2), mgp=c(2,0.5,0))

for (i in 1:4) {
  y <- 1 + 2*x + rnorm(100, 0, 2) # generate response y's

  fit <- lm(y ~ x)
  fit.perfect <- summary(lm(y ~ x))
  rsquared <- round(fit.perfect$r.squared, 2)
  hat_beta_0 <- round(fit.perfect$coefficients[1], 2)
  hat_beta_1 <- round(fit.perfect$coefficients[2], 2)

  plot(x, y, pch=i,
       ylim = c(-8,8),
       xlab = "true mean: y=1+2x in blue, LS in red",
       main = paste("R^2=", rsquared,
                    ", b1_LSE=", hat_beta_1, " and b0=", hat_beta_0))
  abline(fit, lwd=4, col="red")
  lines(x, 1 + 2*x, lwd=4, col="blue")
  abline(h= mean(y), lwd=4, col="green")
}
```



The theory says that

$$\hat{\beta}_1 \sim \mathcal{N}(\beta_1 = 2, \, \mathbf{Var}(\hat{\beta}_1)).$$

```r
sigma <- 2
n <- length(y)
sd_b1 <- sqrt(sigma^2 /((n-1)* (sd(x))^2))  # we will estimate sigma by rse in real life.
sd_b1
```

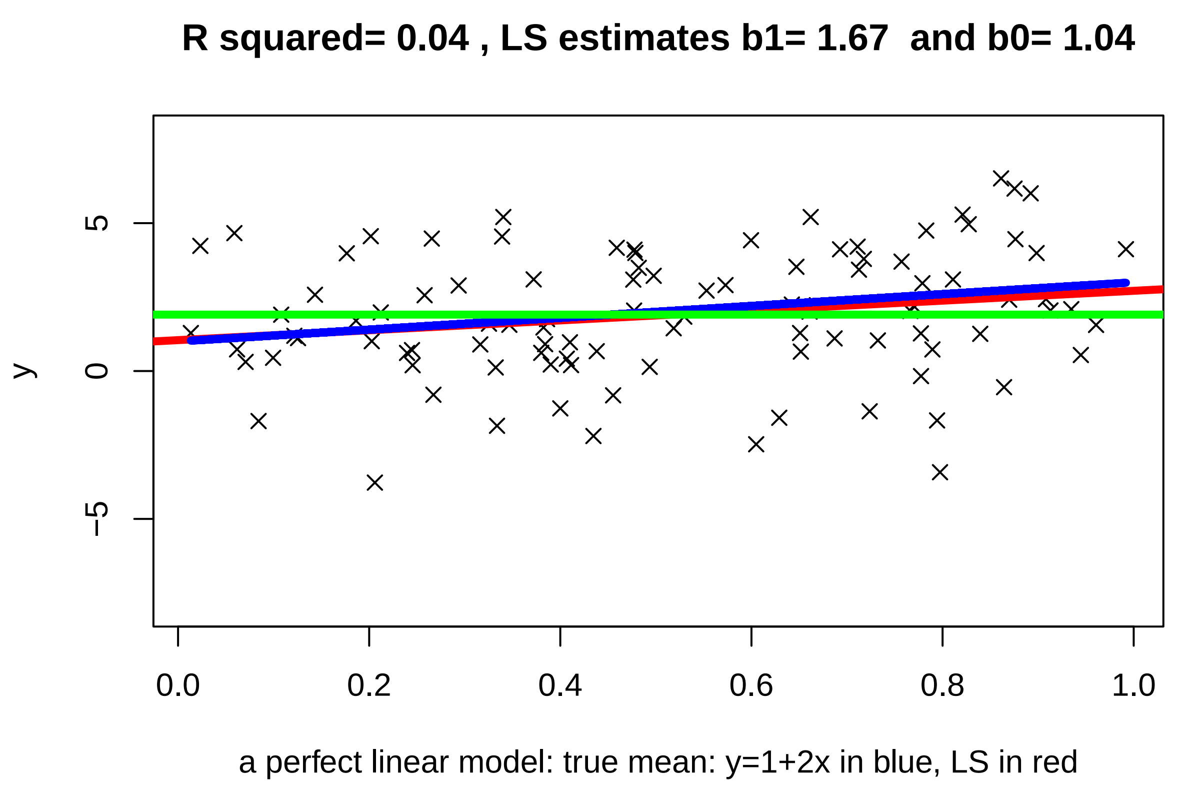
```
## [1] 0.7511921
```

```r
summary(fit)
```

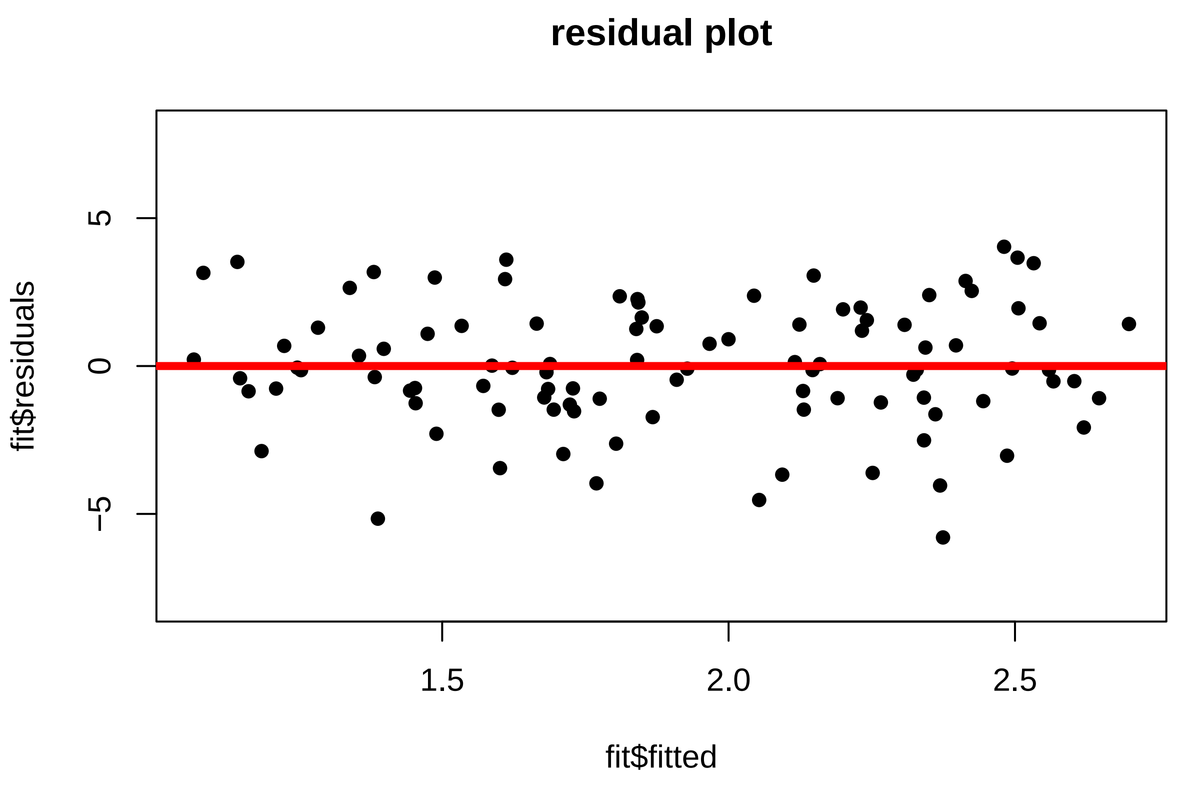
```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7958 -1.1252 -0.0861  1.4240  4.0328
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.0441     0.4607   2.266   0.0256 *
## x             1.6684     0.7913   2.108   0.0375 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.107 on 98 degrees of freedom
## Multiple R-squared:  0.04339,    Adjusted R-squared:  0.03363
## F-statistic: 4.446 on 1 and 98 DF,  p-value: 0.03754
```

Plots

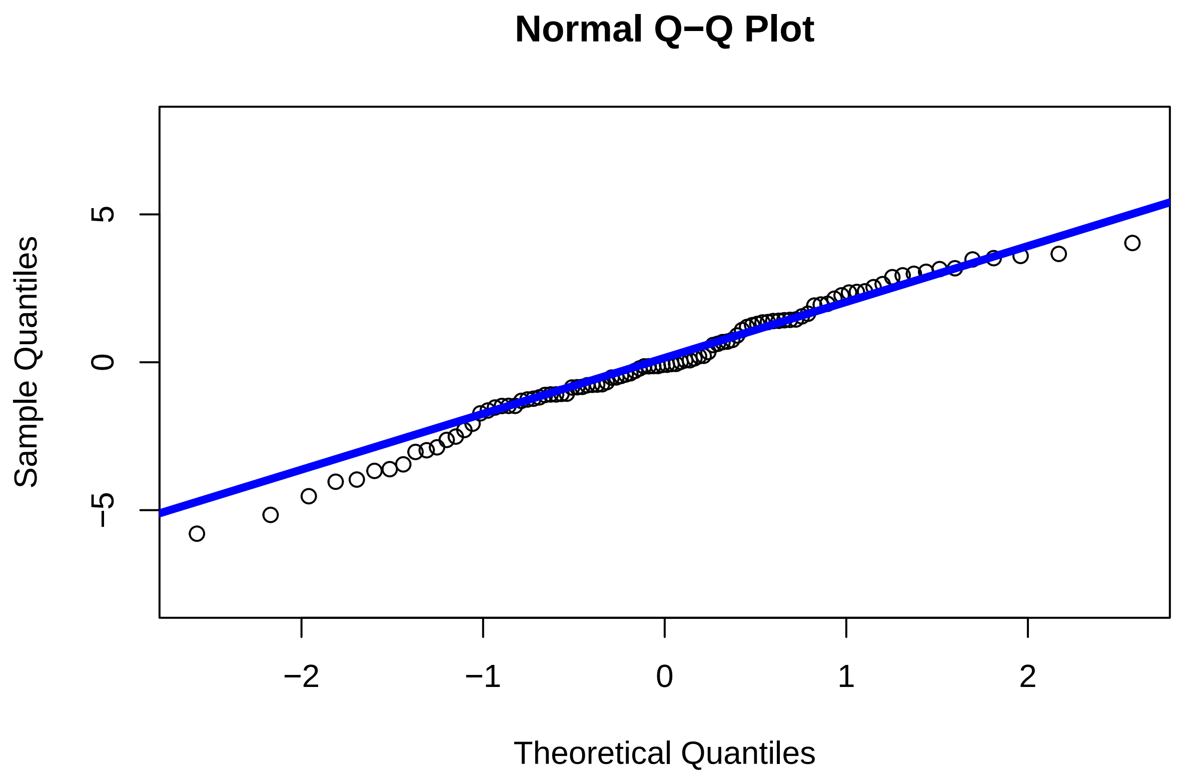
```r
plot(x, y, pch=i,
     ylim = c(-8,8),
     xlab = "a perfect linear model: true mean: y=1+2x in blue, LS in red",
     main = paste("R squared=", rsquared,
                  ", LS estimates b1=", hat_beta_1, " and b0=", hat_beta_0))
abline(fit, lwd=4, col="red")
lines(x, 1 + 2*x, lwd=4, col="blue")
abline(h= mean(y), lwd=4, col="green")
```



```
# Residuals
plot(fit$fitted, fit$residuals,
     pch  = 16,
     ylim = c(-8, 8),
     main = "residual plot")
abline(h=0, lwd=4, col="red")
```



```
# Normality
qqnorm(fit$residuals, ylim=c(-8, 8))
qqline(fit$residuals, lwd=4, col="blue")
```



Appendix 4 - Some Sample Statistics

- Sample mean:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

- Sample variance:

$$s^2 = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$$

- Sample Standard Deviation:

$$s = \sqrt{\frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}}$$

- Correlation:

$$r = \frac{1}{n-1}\sum_{i=1}^{n}(\frac{x_i - \bar{x}}{s_x}) \cdot (\frac{y_i - \bar{y}}{s_y})$$