**Simple Linear Regression Assignment**

### 2 - Delivery\_time -> Predict delivery time using sorting time

> # Load delivery\_time.csv dataset

> library(readr)

> delivery\_time <- read.csv("E:/Data Science Asignments/Simple regression/delivery\_time.csv")

>

> View(delivery\_time)

> # Exploratory data analysis

> summary(delivery\_time)

Delivery.Time Sorting.Time

Min. : 8.00 Min. : 2.00

1st Qu.:13.50 1st Qu.: 4.00

Median :17.83 Median : 6.00

Mean :16.79 Mean : 6.19

3rd Qu.:19.75 3rd Qu.: 8.00

Max. :29.00 Max. :10.00

>

> var(delivery\_time$Delivery.Time)

[1] 25.75462

> sd(delivery\_time$Delivery.Time)

[1] 5.074901

> var(delivery\_time$Sorting.Time)

[1] 6.461905

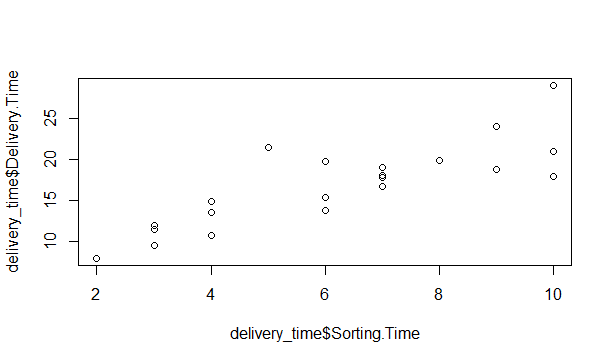
> sd(delivery\_time$Sorting.Time)

[1] 2.542028

#Scatter plot

plot(delivery\_time$Sorting.Time, delivery\_time$Delivery.Time) # plot(X,Y)

?plot



> attach(delivery\_time)

> #Correlation Coefficient (r)

> cor(Sorting.Time,Delivery.Time) # cor(X,Y)

[1] 0.8259973

> > attach(delivery\_time)

The following objects are masked from delivery\_time (pos = 3):

Delivery.Time, Sorting.Time

The following objects are masked from delivery\_time (pos = 4):

Delivery.Time, Sorting.Time

The following objects are masked from delivery\_time (pos = 5):

Delivery.Time, Sorting.Time

The following objects are masked from delivery\_time (pos = 6):

Delivery.Time, Sorting.Time

The following objects are masked from delivery\_time (pos = 7):

Delivery.Time, Sorting.Time

The following objects are masked from delivery\_time (pos = 8):

Delivery.Time, Sorting.Time

> #Correlation Coefficient (r)

> cor(Sorting.Time,Delivery.Time) # cor(X,Y)

[1] 0.8259973

>

> # Simple Linear Regression model

> reg <- lm(Delivery.Time ~Sorting.Time) # lm(Y ~ X)

>

> summary(reg)

Call:

lm(formula = Delivery.Time ~ Sorting.Time)

Residuals:

Min 1Q Median 3Q Max

-5.1729 -2.0298 -0.0298 0.8741 6.6722

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 6.5827 1.7217 3.823 0.00115 \*\*

Sorting.Time 1.6490 0.2582 6.387 3.98e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.935 on 19 degrees of freedom

Multiple R-squared: 0.6823, Adjusted R-squared: 0.6655

F-statistic: 40.8 on 1 and 19 DF, p-value: 3.983e-06

>

> pred <- predict(reg)

>

> reg$Sorting.Time

NULL

> sum(reg$Sorting.Time)

[1] 0

>

> mean(reg$Sorting.Time)

[1] NA

Warning message:

In mean.default(reg$Sorting.Time) :

argument is not numeric or logical: returning NA

> sqrt(sum(reg$Sorting.Time^2)/nrow(Delivery.Time)) #RMSE

numeric(0)

>

> sqrt(mean(reg$Sorting.Time^2))

[1] NaN

>

> confint(reg,level=0.95)

2.5 % 97.5 %

(Intercept) 2.979134 10.186334

Sorting.Time 1.108673 2.189367

> predict(reg,interval="predict")

fit lwr upr

1 23.072933 16.457161 29.68870

2 13.178814 6.780993 19.57663

3 16.476853 10.188630 22.76508

4 21.423913 14.955850 27.89198

5 23.072933 16.457161 29.68870

6 16.476853 10.188630 22.76508

7 18.125873 11.823294 24.42845

8 11.529794 5.010345 18.04924

9 23.072933 16.457161 29.68870

10 21.423913 14.955850 27.89198

11 19.774893 13.411938 26.13785

12 13.178814 6.780993 19.57663

13 18.125873 11.823294 24.42845

14 11.529794 5.010345 18.04924

15 11.529794 5.010345 18.04924

16 13.178814 6.780993 19.57663

17 16.476853 10.188630 22.76508

18 18.125873 11.823294 24.42845

19 9.880774 3.198090 16.56346

20 18.125873 11.823294 24.42845

21 14.827833 8.507631 21.14804

Warning message:

In predict.lm(reg, interval = "predict") :

predictions on current data refer to \_future\_ responses

>

> # ggplot for adding regresion line for data

> library(ggplot2)

>

> ?ggplot2

>

> ggplot(data = delivery\_time, aes(x = Sorting.Time, y = Delivery.Time)) +

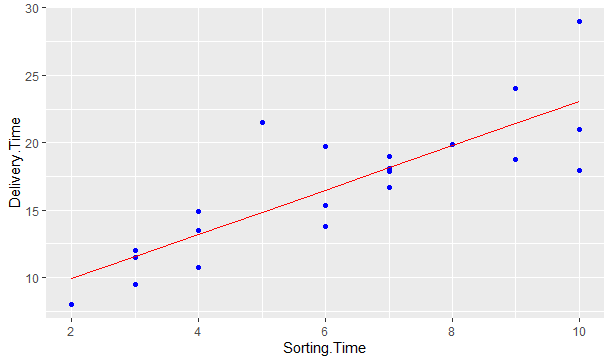
+ geom\_point(color='blue') +

+ geom\_line(color='red',data = delivery\_time, aes(x=Sorting.Time, y=pred))

>

> ?ggplot2

>



# A simple ggplot code for directly showing the line

>

> # ggplot(delivery\_time,aes(Sorting.Time,Delivery.Time))+stat\_summary(fun.data=mean\_cl\_normal) + geom\_smooth(method='lm')

>

> ####################

>

> # Logarithmic Model

>

> # x = log(Sorting.Time); y = Delivery.Time

>

> plot(log(Sorting.Time), Delivery.Time)

> cor(log(Sorting.Time), Delivery.Time)

[1] 0.8339325

>

> reg\_log <- lm(Delivery.Time ~ log(Sorting.Time)) # lm(Y ~ log(X))

>

> summary(reg\_log)

Call:

lm(formula = Delivery.Time ~ log(Sorting.Time))

Residuals:

Min 1Q Median 3Q Max

-4.0829 -2.0133 -0.1965 0.9351 7.0171

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1.160 2.455 0.472 0.642

log(Sorting.Time) 9.043 1.373 6.587 2.64e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.873 on 19 degrees of freedom

Multiple R-squared: 0.6954, Adjusted R-squared: 0.6794

F-statistic: 43.39 on 1 and 19 DF, p-value: 2.642e-06

> predict(reg\_log)

1 2 3 4 5 6 7 8 9

21.98291 13.69652 17.36331 21.03009 21.98291 17.36331 18.75735 11.09489 21.98291

10 11 12 13 14 15 16 17 18

21.03009 19.96493 13.69652 18.75735 11.09489 11.09489 13.69652 17.36331 18.75735

19 20 21

7.42810 18.75735 15.71450

>

> reg\_log$Sorting.Time

NULL

> sqrt(sum(reg\_log$Sorting.Time^2)/nrow(delivery\_time)) #RMSE

[1] 0

>

> confint(reg\_log,level=0.95)

2.5 % 97.5 %

(Intercept) -3.97778 6.297147

log(Sorting.Time) 6.16977 11.917057

> predict(reg\_log,interval="confidence")

fit lwr upr

1 21.98291 19.874779 24.09105

2 13.69652 12.056627 15.33641

3 17.36331 16.038369 18.68824

4 21.03009 19.149435 22.91075

5 21.98291 19.874779 24.09105

6 17.36331 16.038369 18.68824

7 18.75735 17.303803 20.21090

8 11.09489 8.859172 13.33061

9 21.98291 19.874779 24.09105

10 21.03009 19.149435 22.91075

11 19.96493 18.309764 21.62010

12 13.69652 12.056627 15.33641

13 18.75735 17.303803 20.21090

14 11.09489 8.859172 13.33061

15 11.09489 8.859172 13.33061

16 13.69652 12.056627 15.33641

17 17.36331 16.038369 18.68824

18 18.75735 17.303803 20.21090

19 7.42810 4.176349 10.67985

20 18.75735 17.303803 20.21090

21 15.71450 14.358258 17.07073

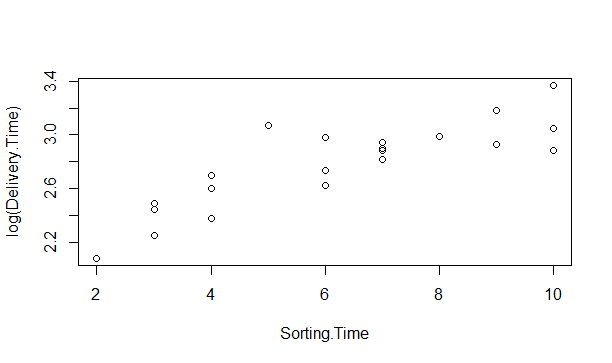
>

> ######################

######################

# Exponential Model

# x = Sorting.Time and y = log(Delivery.Time)



> cor(Sorting.Time, log(Delivery.Time))

[1] 0.8431773

>

> reg\_exp <- lm(log(Delivery.Time) ~ Sorting.Time) #lm(log(Y) ~ X)

>

> summary(reg\_exp)

Call:

lm(formula = log(Delivery.Time) ~ Sorting.Time)

Residuals:

Min 1Q Median 3Q Max

-0.29209 -0.13364 0.02065 0.08421 0.41892

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 2.12137 0.10297 20.601 1.86e-14 \*\*\*

Sorting.Time 0.10555 0.01544 6.836 1.59e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1755 on 19 degrees of freedom

Multiple R-squared: 0.7109, Adjusted R-squared: 0.6957

F-statistic: 46.73 on 1 and 19 DF, p-value: 1.593e-06

>

> reg\_exp$residuals

1 2 3 4 5 6

-0.132365397 0.059111439 0.228472049 0.106717594 0.190407996 -0.023565969

7 8 9 10 11 12

0.084205939 -0.186734850 -0.292087121 -0.140142484 0.021411304 -0.168672492

13 14 15 16 17 18

-0.046022644 0.004320387 0.049376881 0.156439783 -0.133642618 0.036231231

19 20 21

-0.253033509 0.020649391 0.418923091

>

> sqrt(mean(reg\_exp$residuals^2))

[1] 0.1669628

>

> logat <- predict(reg\_exp)

> at <- exp(logat)

> error =delivery\_time $Delivery.Time - at

> error

1 2 3 4 5 6 7

-2.97203240 0.77487677 4.03396635 2.42929301 5.02796760 -0.36603365 1.53440335

8 9 10 11 12 13 14

-1.95042273 -6.07203240 -2.82070699 0.42007296 -1.97512323 -0.78559665 0.04957727

15 16 17 18 19 20 21

0.57957727 2.15487677 -1.96603365 0.64440335 -2.30341147 0.36440335 7.35827221

>

> sqrt(sum(error^2)/nrow(delivery\_time)) #RMSE

[1] 2.94025

>

> confint(reg\_exp,level=0.95)

2.5 % 97.5 %

(Intercept) 1.90584807 2.3368956

Sorting.Time 0.07323457 0.1378686

> predict(reg\_exp,interval="confidence")

fit lwr upr

1 3.176888 3.029973 3.323803

2 2.543578 2.436627 2.650529

3 2.754681 2.674275 2.835088

4 3.071336 2.950212 3.192461

5 3.176888 3.029973 3.323803

6 2.754681 2.674275 2.835088

7 2.860233 2.775902 2.944565

8 2.438027 2.307419 2.568634

9 3.176888 3.029973 3.323803

10 3.071336 2.950212 3.192461

11 2.965785 2.866552 3.065017

12 2.543578 2.436627 2.650529

13 2.860233 2.775902 2.944565

14 2.438027 2.307419 2.568634

15 2.438027 2.307419 2.568634

16 2.543578 2.436627 2.650529

17 2.754681 2.674275 2.835088

18 2.860233 2.775902 2.944565

19 2.332475 2.175100 2.489850

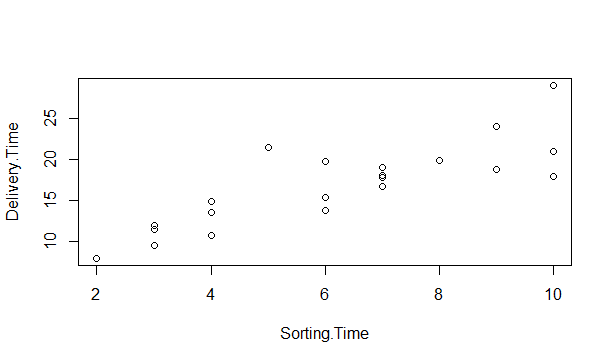
20 2.860233 2.775902 2.944565

21 2.649130 2.560206 2.738054

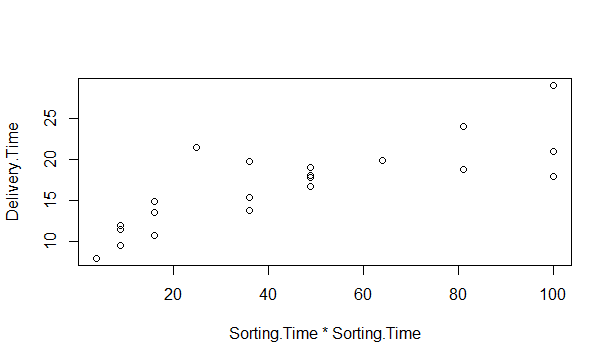
> ##############################

> # Polynomial model with 2 degree (quadratic model)

> plot(Sorting.Time,Delivery.Time)



plot(Sorting.Time\*Sorting.Time, Delivery.Time)

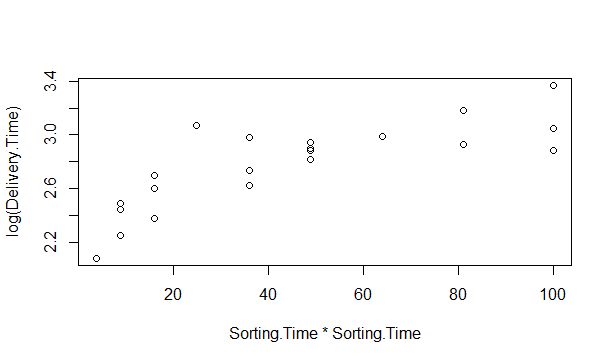


cor(Sorting.Time\*Sorting.Time, Delivery.Time)

[1] 0.7939063

>

> plot(Sorting.Time\*Sorting.Time, log(Delivery.Time))

>

cor(Sorting.Time, log(Delivery.Time))

[1] 0.8431773

> cor(Sorting.Time\*Sorting.Time, log(Delivery.Time))

[1] 0.7882452

>

> # lm(Y ~ X + I(X\*X) +...+ I(X\*X\*X...))

>

> reg2degree <- lm(log(Delivery.Time) ~ Sorting.Time + I(Sorting.Time\*Sorting.Time))

>

> summary(reg2degree)

Call:

lm(formula = log(Delivery.Time) ~ Sorting.Time + I(Sorting.Time \*

Sorting.Time))

Residuals:

Min 1Q Median 3Q Max

-0.21194 -0.11776 -0.03034 0.10550 0.35975

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1.69970 0.22843 7.441 6.77e-07 \*\*\*

Sorting.Time 0.26592 0.08022 3.315 0.00385 \*\*

I(Sorting.Time \* Sorting.Time) -0.01284 0.00632 -2.032 0.05722 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.1627 on 18 degrees of freedom

Multiple R-squared: 0.7649, Adjusted R-squared: 0.7387

F-statistic: 29.28 on 2 and 18 DF, p-value: 2.197e-06

>

> logpol <- predict(reg2degree)

> expy <- exp(logpol)

>

> err = delivery\_time$Delivery.Time - expy

>

> sqrt(sum(err^2)/nrow(delivery\_time)) #RMSE

[1] 2.799042

>

> confint(reg2degree,level=0.95)

2.5 % 97.5 %

(Intercept) 1.21978583 2.1796227087

Sorting.Time 0.09738139 0.4344624477

I(Sorting.Time \* Sorting.Time) -0.02611902 0.0004378164

> predict(reg2degree,interval="confidence")

fit lwr upr

1 3.074863 2.902224 3.247503

2 2.557942 2.457361 2.658524

3 2.832974 2.722755 2.943193

4 3.052913 2.938651 3.167174

5 3.074863 2.902224 3.247503

6 2.832974 2.722755 2.943193

7 2.931968 2.824007 3.039929

8 2.381905 2.247271 2.516538

9 3.074863 2.902224 3.247503

10 3.052913 2.938651 3.167174

11 3.005281 2.904349 3.106214

12 2.557942 2.457361 2.658524

13 2.931968 2.824007 3.039929

14 2.381905 2.247271 2.516538

15 2.381905 2.247271 2.516538

16 2.557942 2.457361 2.658524

17 2.832974 2.722755 2.943193

18 2.931968 2.824007 3.039929

19 2.180186 1.965180 2.395192

20 2.931968 2.824007 3.039929

21 2.708299 2.605416 2.811182

>

> # visualization

> ggplot(data = delivery\_time, aes(x = Sorting.Time + I(Sorting.Time^2), y = log(Delivery.Time))) +

+ geom\_point(color='blue') +

+ geom\_line(color='red',data = delivery\_time, aes(x=Sorting.Time+I(Sorting.Time^2), y=logpol))

