**Describing graph**

**Describing interpretation of x  
  
  
Variance**

variance in R is sample variance  
 **Correlation**

cor(var1, var2)

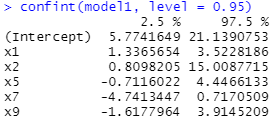
**R2 &**

* cor(var1,var2)2
* summary(model1)
  + Multiple R-squared : Normal R-squared
  + Adjusted R-squared

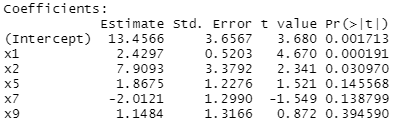


**Confidence interval**

* confint(model, ”df$colx”, level = 0.95)



* summary(model)

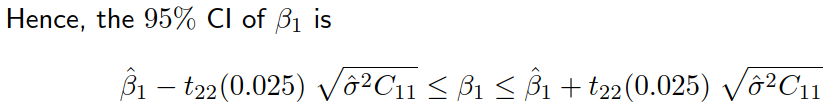


**Prediction Interval**

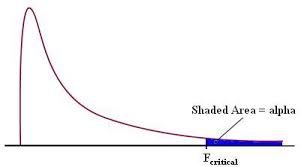
predict(model, data.frame(x=c(2)), interval = “confidence”, level = 0.95)

**Hypothesis test**

1. Test for model significance
2. Test for single regressor significance, given all other regressors included ( T-test )
3. Testing for multiple regressor significance given all other regressors included (F-test)  
   [ Extra sum of Squares method ]  
   q Regressors kept as last q variables in lm(Y ~ x1 + … Xq1 + Xq1 + .. Xq)

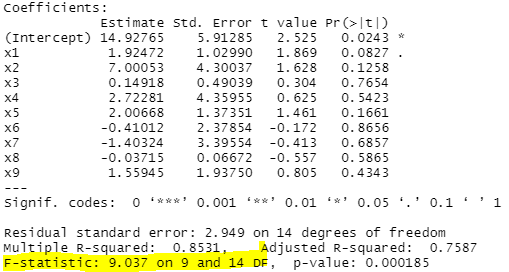
**Se(Bj)**  
Generated by summary(model) is sufficient for use without multiplying for hypothesis test/ confidence interval  


**F-Test**



F test is one sided ( significance level not halved) , to the right side

qf(p, df1, df2, **lower.tail = FALSE** )

**Testing Significance of Model**Using F – test  
anova(model)

**F-statistic** & **p-value** of f statistic

**Indicator Variables**

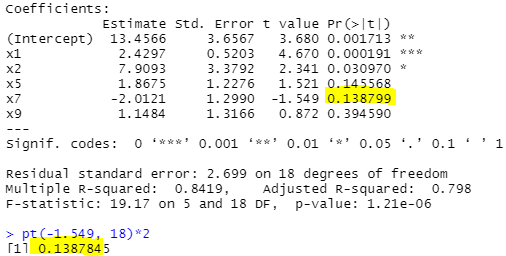
df$colx <- as.factor(df$colx)  
lm(y ~ x + colx)

**Interaction terms**

lm(y ~ x1 + x2 \* x3)

**Model Residuals**

model$residuals

**Testing significance of individual Fitted Values j**Assuming all other regressors are included in the model  


* P-value obtained using pt
* P-value as indicated, is for a **2 sided test.**

For testing individual regressors for T-test, *not required* *to set target variable X1* at the end. (sequence of variables does not matter, summary(model) will give same stats.

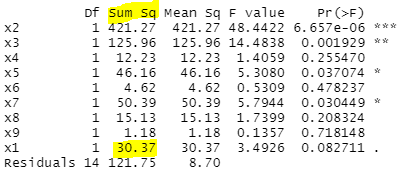
**Testing significance of a group of Fitted Values j**

Extra sum of squares  
Target variables Xi and Xk to put at the end

anova(model)

**Calculating contribution of variable to sum of squared *regression* of model 1, given that all other regressors as included**

model2 <- lm( y ~ x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + **x1**, data = hseprice)  
anova(model2)

   
Set target variable to be at the very end.

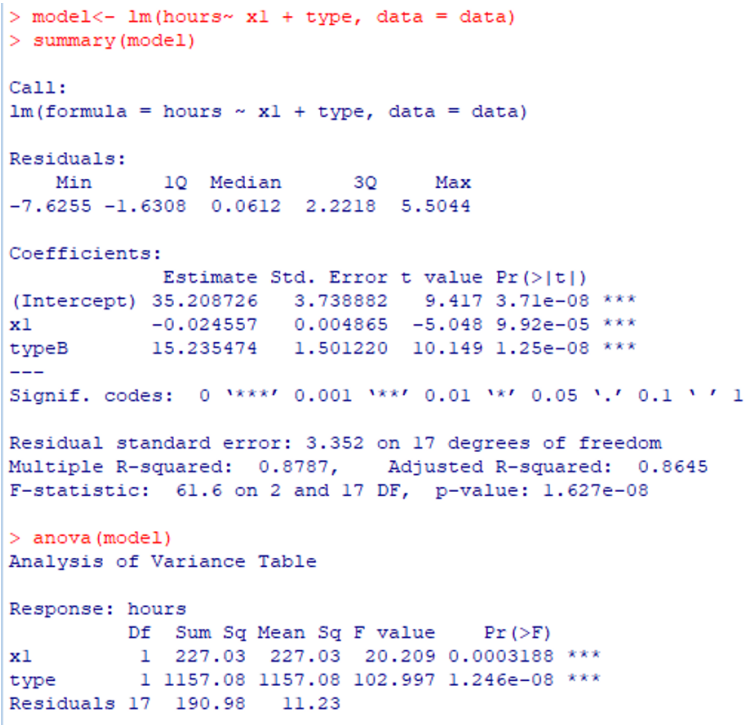
**Indicator Variable**

df$colx <- as.factor(df$colx)

lm(y ~ colx, data = df)

**Used for:**

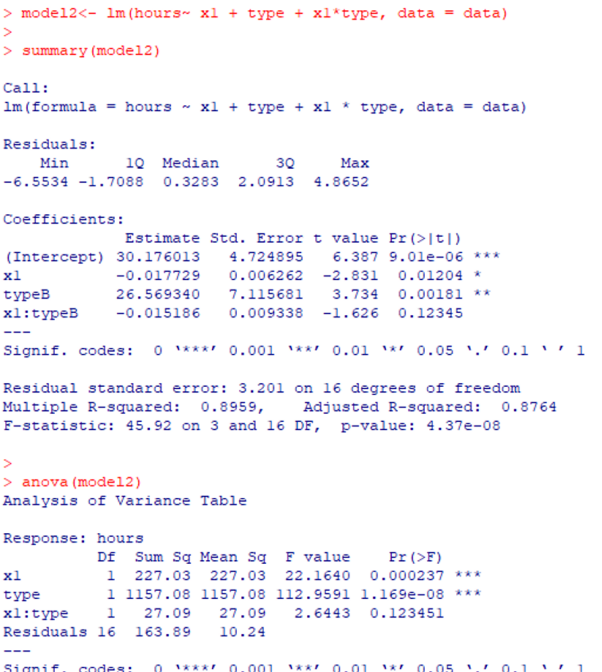
Confidence interval calculation



log(FEV) = -1.94293 + 0.02339 \* Age + 0.04278 \* Hgt + 0.029248 \* **I(Sex = 1)** - 0.04602 \* **I(Smoke =1 )**

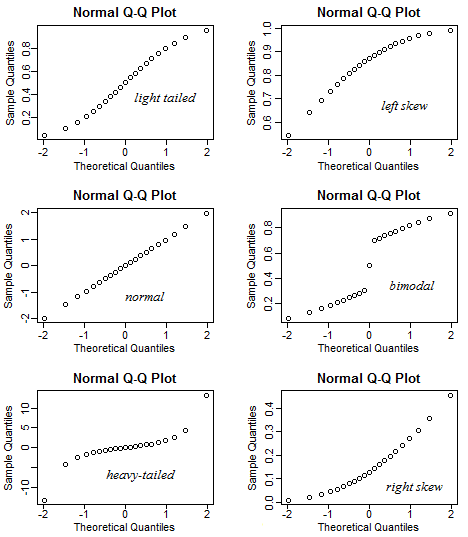
**Interaction Terms**

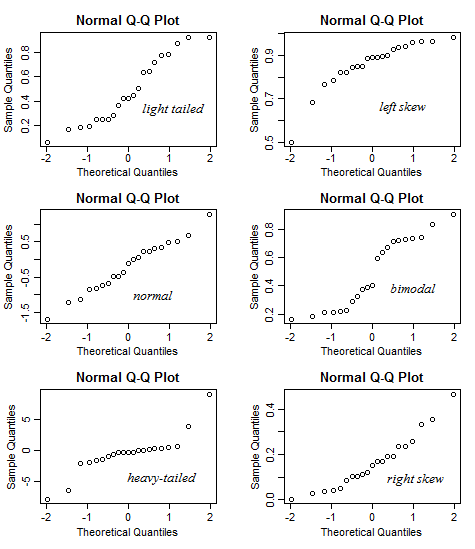
Used for:  
Anova to check model accuracy

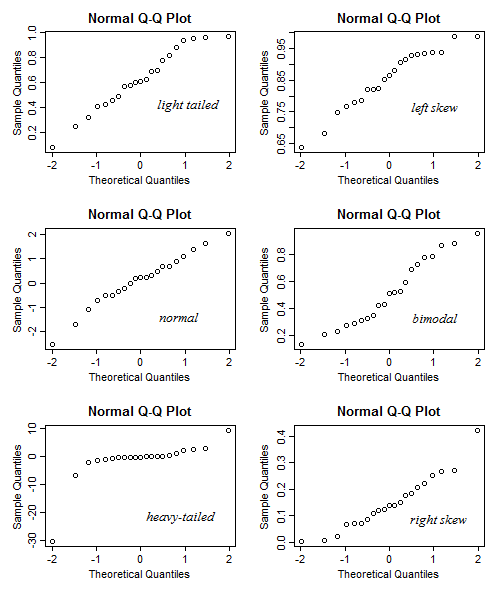


**QQplot**

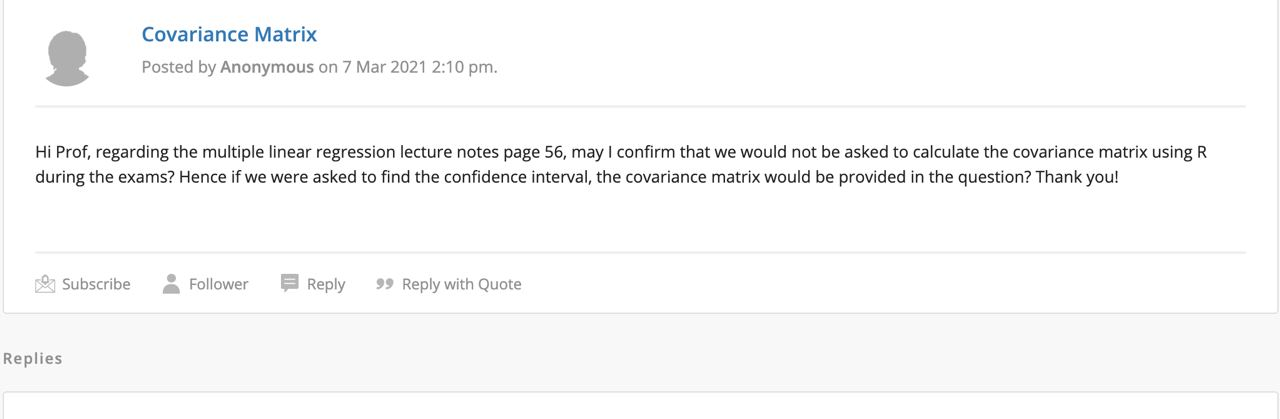
qqnorm(rstandard(model))  
qqline(rstandard(model))

**Interpretation of QQplot**  






Interpreting QQplots : left-skew or right-skew  
left tail below qqline -> values are lower than expected, left tail is longer than expected (heavier)  
right tail above qqline -> values are higher than expected, right tail is longer than expected (heavier)



data<- read.table("Delivery\_Time.csv", sep= ",", header = TRUE)

names(data)[2] = 'time'  
names(data)[3] = 'cases'  
names(data)[4] = 'distance'

attach(data)  
n = length(time) # sample size

x = cbind(c(rep(1,n)), cases, distance) #this is the matrix **X** in slide 16

(y = time )  
beta = solve(t(x) %\*%x)%\*%t(x)%\*%time #formula in slide 18 for beta^

Beta # the estimated coefficients for the fitted model

Matrix C = (X'X)^(-1) is easy to be done in R:

C = solve(t(x) %\*%x)

**Hat Matrix**

x <- cbind(**rep(1,n)**, col1, col2)  
hat <- x %\*% solve(t(x) %\*% x) %\*% t(x) # hat matrix  
diag(hat)

**Influential points**

diag(hat)[diag(hat) > 2\*p/n]

**Cooks Distance**

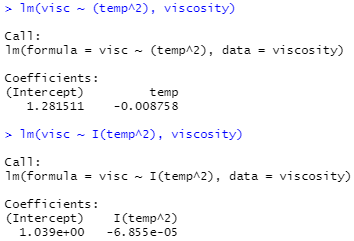
cookd <- cooks.distance(model)  
which(cookd > 1)

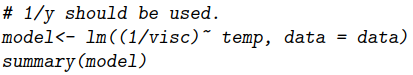
**MATRIX operations**

* Transpose, :
  + t(X)
* Find inverse, :
  + solve(X)
* Matrix Multiplication, X●X :
  + X %\*% X

For **transformed regressors**, required to add I()

* Not required for transformations on **Y**





**Box – Cox Method**

library(MASS)

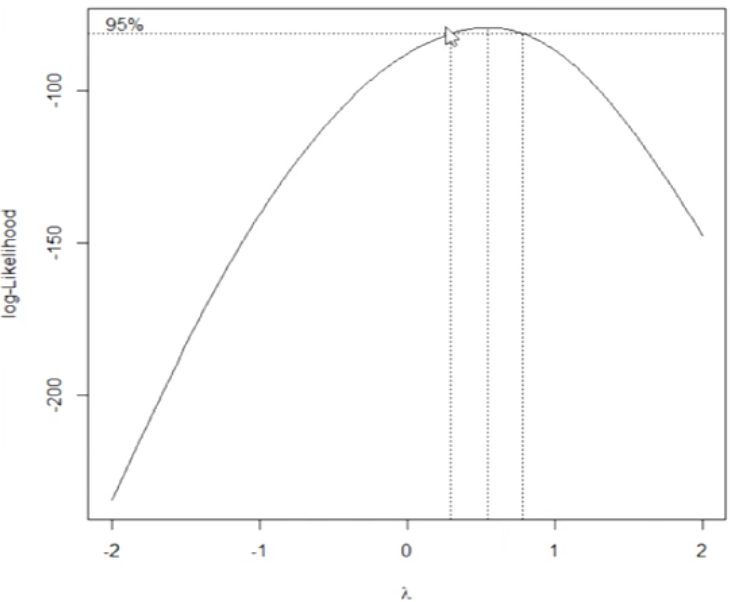
model <- lm(y ~ x, data)

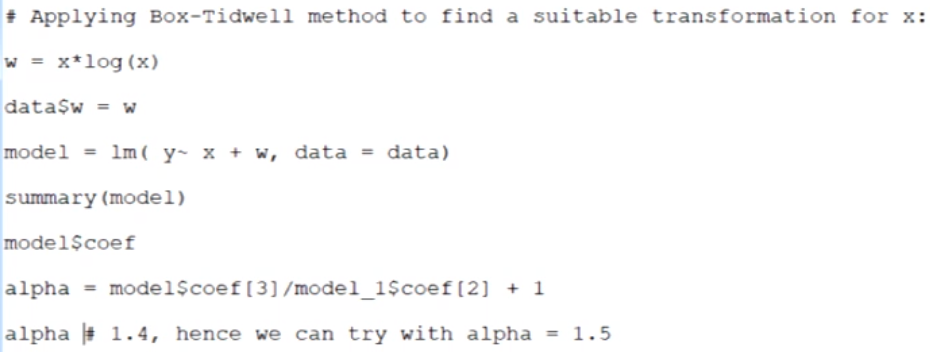
boxcox(model, lambda = seq(-2, 2, by = 0.5), optimize = TRUE, plotit = TRUE)  
seq(-2,2, by = 0.1)

, adjust by = 0.1, to be sth smaller

value of lambda, is for transformation on y,

if lambda = 0, transform y to log(y)





**To find VIF:**

r <- cor(x) r – correlation x

c <- solve(r)

VIF = diag(c)

VIF > 5 or 10 : indicate multicollinearity

**Eigensystem Analysis**

regressor matrix  
x <- cbind(df$x, df$x^2)

# take correlation of regression matrix

r <- cor(x)

# take inv of corr matrix

corInvX <- solve(r)

eigen(r)$values

condNum <- max(eigen(r)$values) / min(eigen(r)$values)

condNum <100 : no serious multicollinearity  
100< condNum <100 : moderate to strong multicollinearity

condNum > 1000: strong multicollinearity

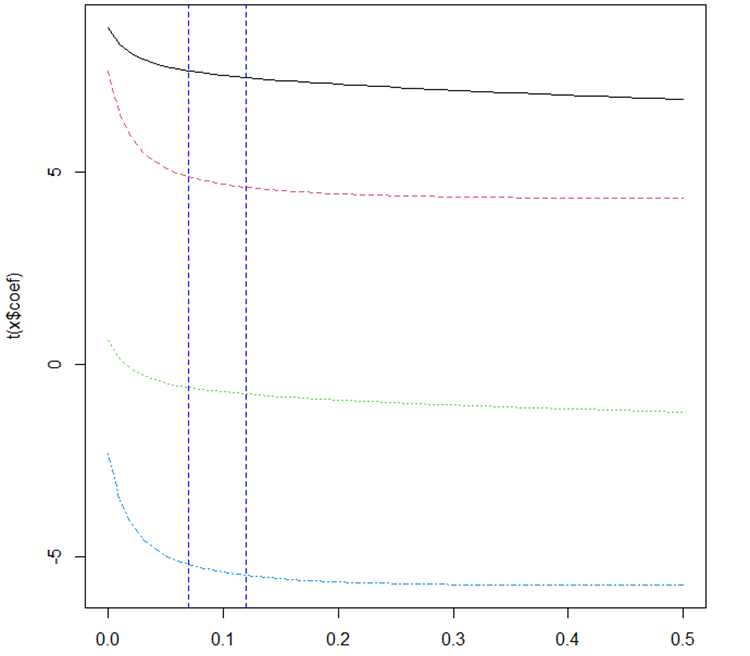
library(MASS)

lm.ridge(y ~ x1 + x2 + x3, data, lambda = 0.01)

Plot to find best lambda value

plot(lm.ridge(y ~ x1 + x2 + x3, data, lambda = seq(0, 0.5, 0.01))

(must use lm.ridge not lmridge from lmridge library, as plot will be different & cannot deduce optimal lambda value)



Choose lambda value where gradient stable and lambda is small(0.08-0.12)

**For predictions( or SSres )**

pred1 <- as.matrix(cbind(1,haldCement$x1, haldCement$x2, haldCement$x3, haldCement$x4)) %\*% rbind(81.8251068,1.3301867,0.3137801,-0.1160452,-0.3357076)

t(pred1 - haldCement$y) %\*% (pred1 - haldCement$y)

Call **model for coefficients**, **DO NOT USE model$coef**, it is not the same

**recall (include the intercept weight and regressor)**

* include intercept column

cbind(1, x)

* include X0 intercept weight

c(w0, model$coef)

SSt = var(y) \* (NROW(y) – 1)

= 1 – SSres/SSt

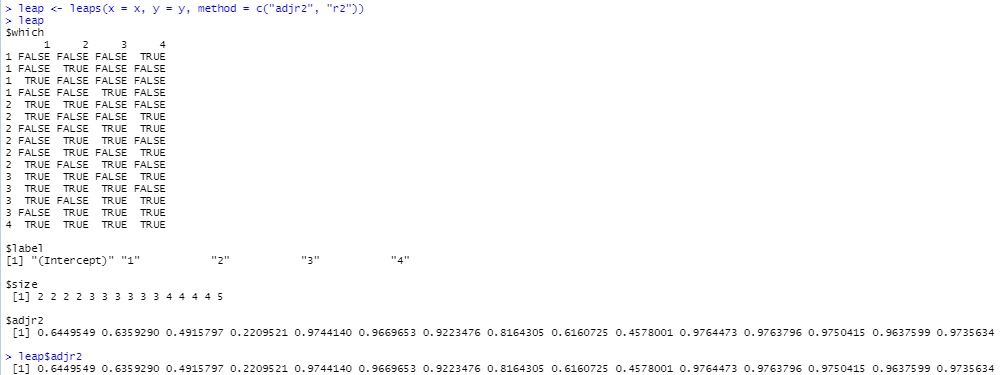
install.packages('leaps')

scope = full model of k regressors

library(leaps)

x <- cbind(x\_1, x\_2, x\_3, x\_4)

leap <- leaps(x = x, y = y, method = c("adjr2", "r2"))  
method = c("Cp", "adjr2", "r2”)



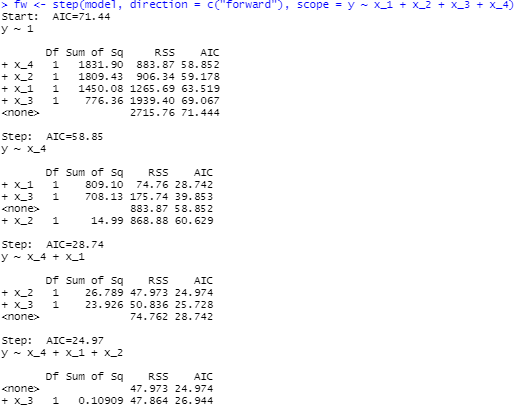
which.max(leap$adjr2)

obtain the index, and find the index-th row and choose the regressors accordingly(row number may not tally with index)

**Forward**

model <- lm(y~ 1, haldCement)

fw <- step(model, direction = c("forward"), **scope = y ~ x\_1 + x\_2 + x\_3 + x\_4**)

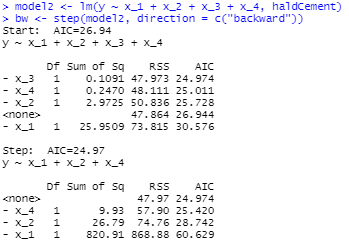


criteria chosen is by **lowest AIC**

**Backward**

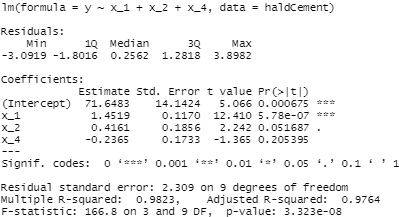
model2 <- lm(y ~ x\_1 + x\_2 + x\_3 + x\_4, haldCement)

bw <- step(model2, direction = c("backward"))



(left column minus because removing, compared to forward)

summary(bw)



sw <- step(model2, direction = c("both"))