# vector of normally distributed random values around mean 20  
x = rnorm(50, mean=20)  
#real f       #error term  
y = (2 \* x + 3) + rnorm(50,sd=1)  
#hist(x)  
#hist(y)  
#convert to data frame - could be one step  
mydat = data.frame(x,y)  
#linear regression – predict y using x  
lm.fit = lm(data = mydat,formula = y ~ x)  
summary(lm.fit)  
#plot it  
plot(x,y)  
abline(lm.fit)  
# extract coefficients  
lm\_coef <- round(coef(lm.fit), 3)   
mtext(bquote(y == .(lm\_coef[2])\*x + .(lm\_coef[1])),   
      adj=1, padj=0) # display equation

#more complex model - uses powers of x and multiple terms  
lm.flexfit = lm(data=mydat, formula = y ~ x+I(x^2)+I(x^3))  
#should be a little more accurate  
summary(lm.flexfit)  
#plot it  
plot(y~x)  
abline(lm.fit)  
points(x, fitted(lm.flexfit), col='red', pch=20)  
lines(sort(x), fitted(lm.flexfit)[order(x)], col='red', type='b')   
#now, new data using same generation logic  
x = rnorm(50, mean=20)  
#real f       #error term  
y = (2 \* x + 3) + rnorm(50,sd=1)  
testdat = data.frame(x,y)  
#predict with simple model  
linear\_pred <- predict(lm.fit, newdata=testdat)  
mean(abs(linear\_pred - testdat$y))  
#predict with overfit complex model  
complex\_pred <- predict(lm.flexfit, newdata=testdat)  
mean(abs(complex\_pred - testdat$y))  
#simple model should be closer  
#install.packages(ISLR)  
lm.BostonMulti = lm(medv~lstat+age+indus+rm,data=Boston)  
summary(lm.BostonMulti)  
lm.BostonMulti = lm(medv~lstat+rm,data=Boston)  
summary(lm.BostonMulti)  
predict(lm.BostonMulti, data.frame(lstat=c(10,20,50),rm=c(6.1,6.8,7.2)),  
        interval="confidence")  
predict(lm.BostonMulti, data.frame(lstat=c(10,20,50),rm=c(6.1,6.8,7.2)),  
        interval="prediction")