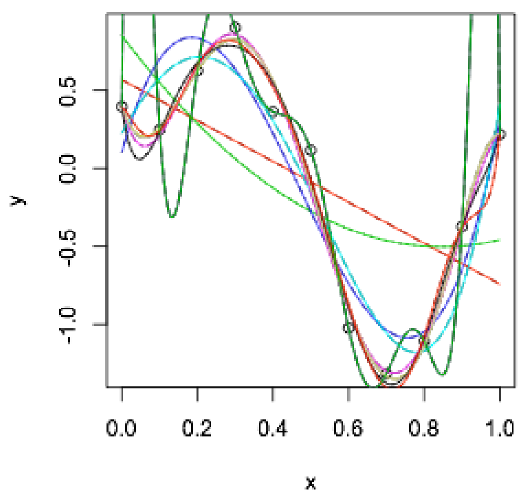
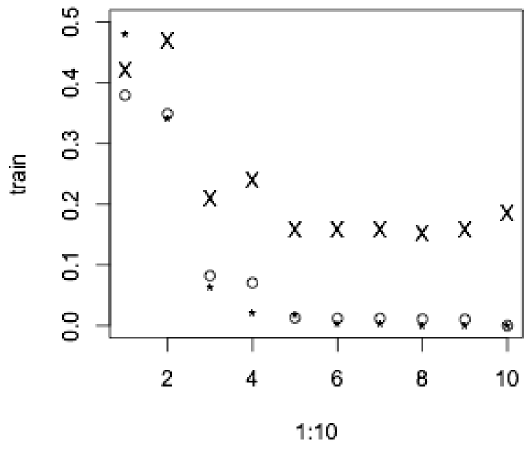
```
## R code for fitting various polynomial regressions

## generate some data
x = seq(0,1,length=11)
y = sin(2*pi*x) + rnorm(11, sd=0.3)

## plot it
plot(x,y)

## fit a linear model
lm1 = lm(y~x)

## you can look at the output with, e.g.
summary(lm1)

## now fit everything
lm10 = lm(y~x
+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)+I(x^9)+I(x^10))
lm9 = lm(y~x
+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8)+I(x^9))
lm8 = lm(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7)+I(x^8))
lm7 = lm(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6)+I(x^7))
lm6 = lm(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5)+I(x^6))
lm5 = lm(y~x+I(x^2)+I(x^3)+I(x^4)+I(x^5))
lm4 = lm(y~x+I(x^2)+I(x^3)+I(x^4))
lm3= lm(y~x+I(x^2)+I(x^3))
lm2 = lm(y~x+I(x^2))

## I want to plot nice smooth curves, so will evaluate these lms
## at a finer x-grid

xplot=seq(0,1,length=200)

# the next command adds to the current plot
lines(xplot,predict(lm3,newdata=data.frame(x=xplot)), col="blue")

# and so on
lines(xplot,predict(lm2,newdata=data.frame(x=xplot)), col="red")
lines(xplot,predict(lm5,newdata=data.frame(x=xplot)), col="green")
lines(xplot,predict(lm10,newdata=data.frame(x=xplot)),
col="purple")

# Here is a better version, taken from Notes I found on-line
# the reference is http://www.stat.cmu.edu/~cshalizi/350/, Lecture
```

```
for(degree in 1:10){
   fm = lm(y~poly(x,degree))
   assign(paste("y",degree,sep="."),fm) # this isn't needed, but
handy
   lines(xplot,predict(fm,data.frame(x=xplot)), col=degree)
   }



#truth
lines(xplot,sin(2*pi*xplot),col="gray")

#getting the test and training errors

lms = list(lm1,lm2,lm3,lm4,lm5,lm6,lm7,lm8,lm9,lm10)

# set up two null vectors for the errors

train = test = rep(0,10)  # train = vector(length=10) also works

# this is SSE/10

for(i in 1:10){train[i]=var(lms[i][[1]]$residuals)}

# now compute similarly scaled prediction error for test data

newy = sin(2*pi*x)+rnorm(11,sd=.3)  # note we used the same x's

for(j in 1:10){test[j] = (1/10)*sum((newy-predict(lms[j]
[[1]],newdata=data.frame(x)))^2)}

plot(1:10,train, ylim=c(0,.5))
points(1:10,test,pch="X")
```



```
cbind(train,test)
```


train
1:10