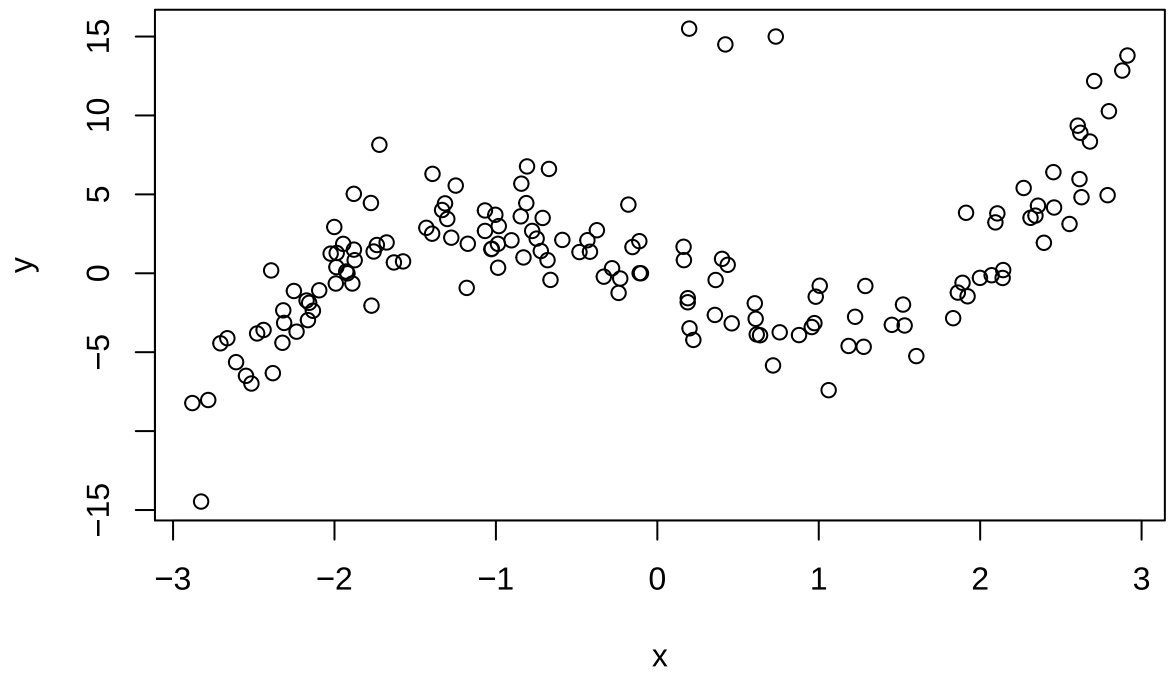
3.(a)

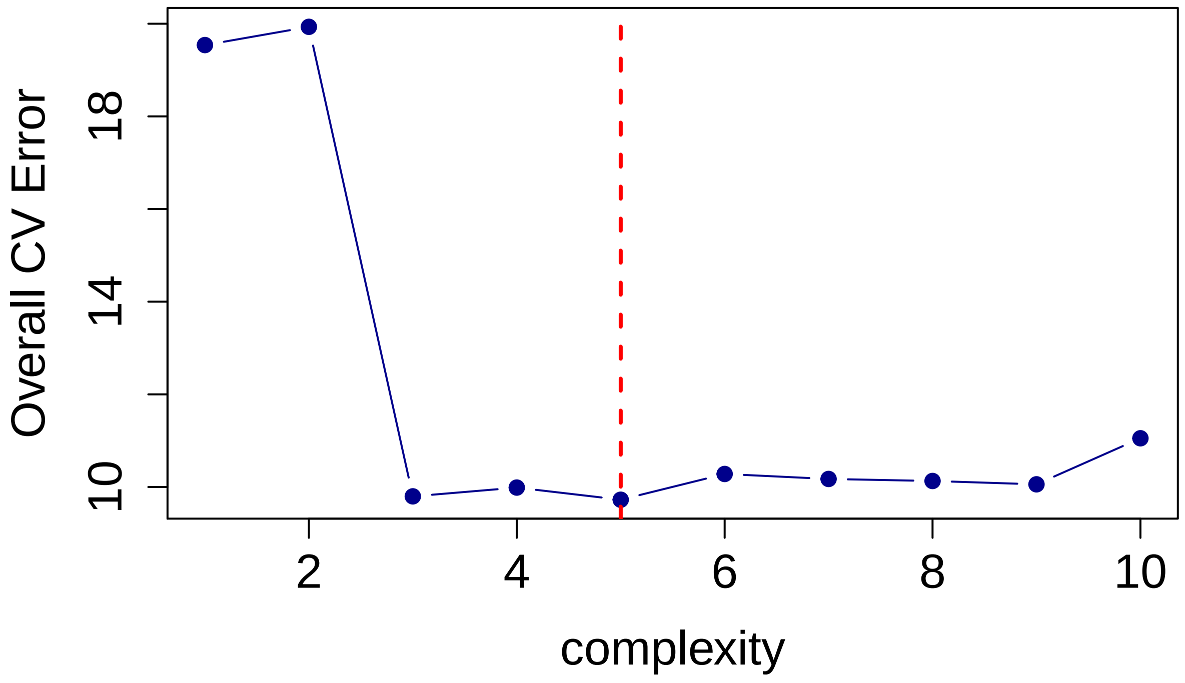
```
set.seed(444)
x = runif(150,-3,3)
y = (x+2)*(x)*(x-2) + rnorm(150,sd=2)
Extreme.Indx = c(19 , 45 , 124)
y[Extreme.Indx] = c(15.5 , 14.5 , 15)
plot(x, y)
```



3.(b)

```r
data = data.frame('x' = x, 'y' = y)
set.seed(444)
for(i in 1:length(complexity)){
  glm.fit = glm(y ~ poly(x, complexity[i]))
  cv.err[i] = cv.glm(data, glm.fit, K = 10)$delta[1]
}
```

```r
plot(complexity, cv.err, pch=19, col="darkblue", type="b",
     cex.axis = 1.5, cex.lab=1.5, ylab="Overall CV Error")
indx = which.min(cv.err)
abline(v=indx, lty=2, lwd=2, col='red')
```



The CV errors are:

```r
cv.err
```

```
##  [1] 19.539397 19.932891  9.798028  9.987752  9.725404 10.281030 10.173550
##  [8] 10.129751 10.057638 11.051842
```

We choose the model with complexity 5.

The parameter estimates are:

```r
glm(y ~ poly(x, 5))$coefficient
```

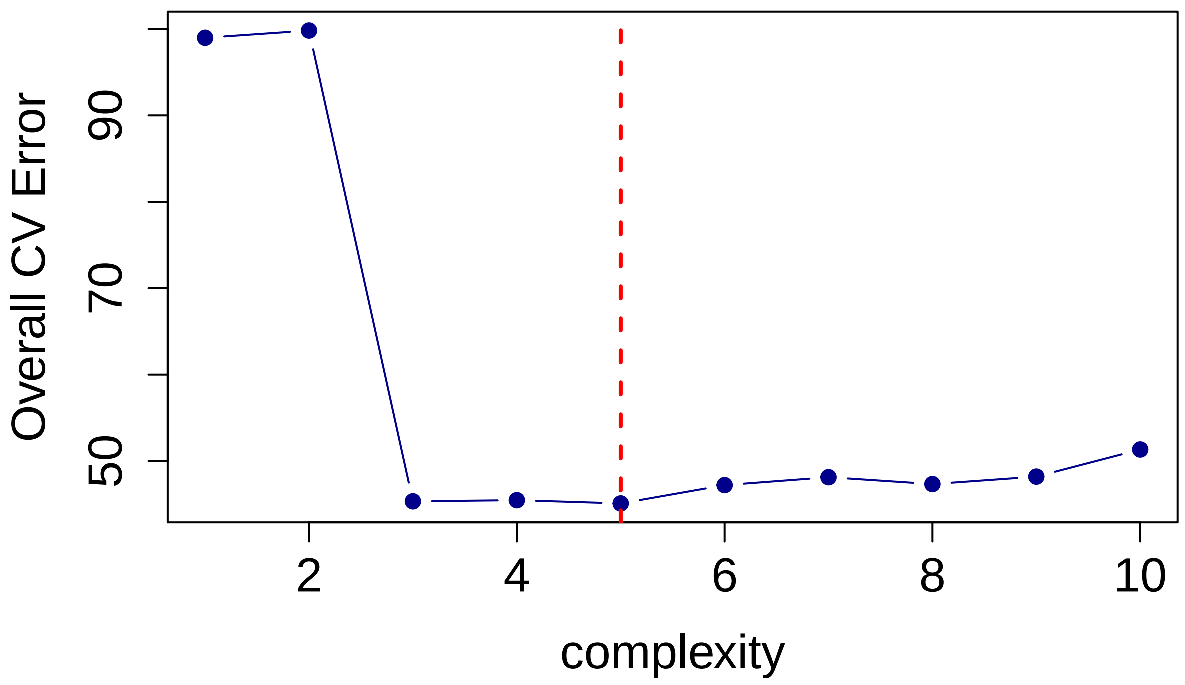
```
## (Intercept) poly(x, 5)1 poly(x, 5)2 poly(x, 5)3 poly(x, 5)4 poly(x, 5)5
##   0.8849621  19.7095868  -0.3411478  38.4914044   3.1915946   0.9058450
```

3.(c)

```r
cost = function(y, yhat) {
  r = y - yhat
  sum = 0
  for(i in 1:length(r)) {
    if(abs(r[i]) <= 3) {
      sum = sum + 1 / 2 * r[i] ^ 2
    } else {
      sum = sum + 3 * (abs(r[i]) - 1.5)
    }
  }
  return(sum)
}

data = data.frame('x' = x, 'y' = y)
set.seed(444)
for(i in 1:length(complexity)){
  glm.fit = glm(y ~ poly(x, complexity[i]))
  cv.err[i] = cv.glm(data, glm.fit, cost, K = 10)$delta[1]
}
```

```r
plot(complexity, cv.err, pch=19, col="darkblue", type="b",
     cex.axis = 1.5, cex.lab=1.5, ylab="Overall CV Error")
indx = which.min(cv.err)
abline(v=indx, lty=2, lwd=2, col='red')
```



The cv errors are:

```
cv.err
```

```
##  [1] 98.98340 99.81865 45.33607 45.46929 45.09229 47.21182 48.12844
##  [8] 47.33174 48.19553 51.33455
```

We choose the model with complexity 5.

The parameter estimates are:

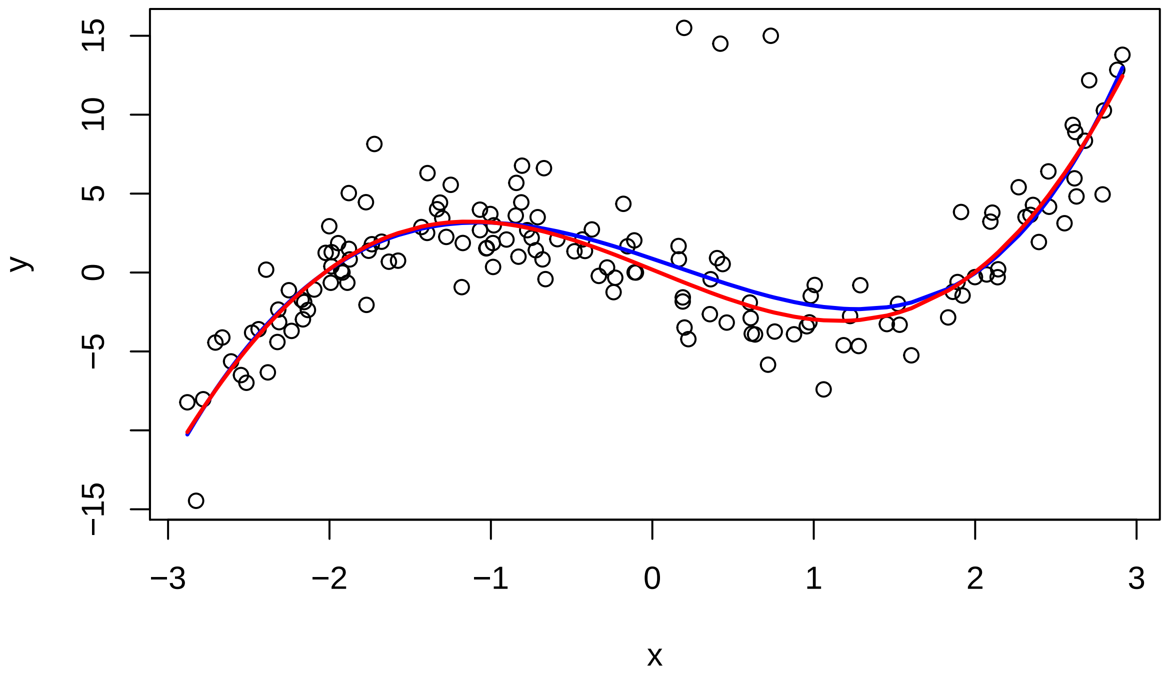
```
library(MASS)

huberfn = function (u, k = 3) {
  return(pmin(1, k/abs(u)))
}
rlm(y ~ poly(x, 5), psi = huberfn)$coefficient
```

```
## (Intercept) poly(x, 5)1 poly(x, 5)2 poly(x, 5)3 poly(x, 5)4 poly(x, 5)5
##   0.6684345  18.6787377   2.4116282  40.5387255   1.4500114  -1.5255481
```

3.(d)

```
xorder = order(x)
plot(x, y)
lines(x[xorder], predict(lm(y ~ poly(x, 5)))[xorder], type="l", col="blue", lwd=2)
lines(x[xorder], predict(rlm(y ~ poly(x, 5), psi = huberfn))[xorder], type="l", col="red", lwd=2)
```



Both lines are reasonable close to each other except the range (-1, 1). Since robust fit is less sensitive to the outliers.