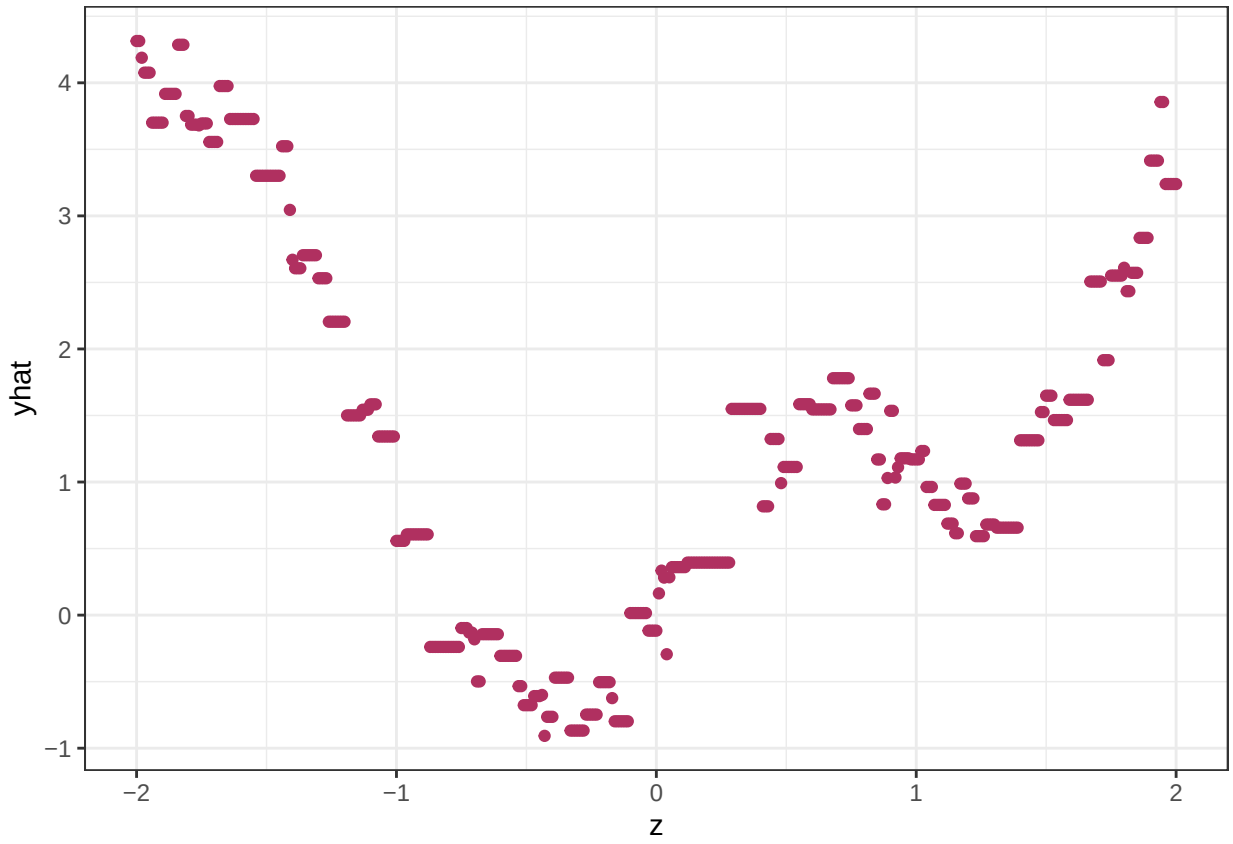
Assignment 7 Coding Part

Kaicheng Luo

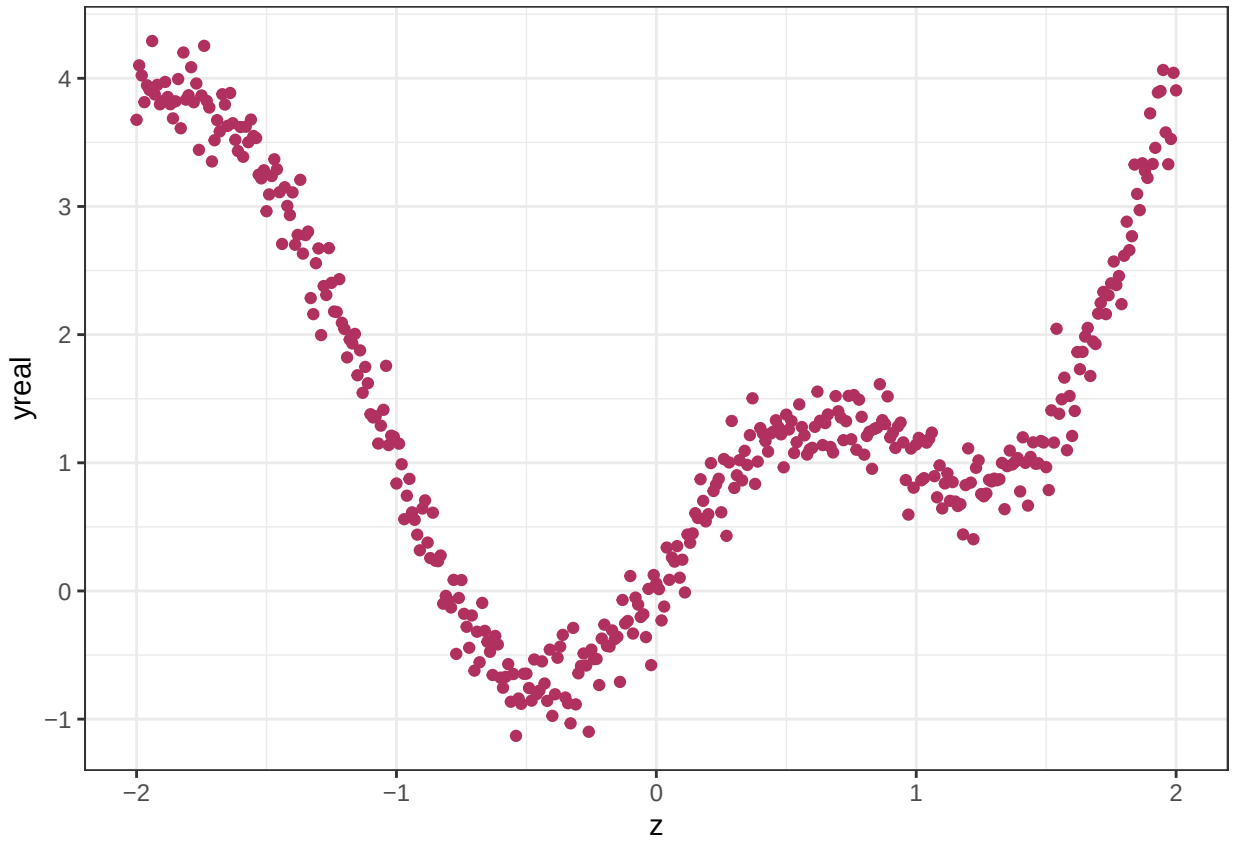
2019/10/29

```
d <- function(x,z,p=2){
  return((abs(x-z)^p)^(1/p))
}

kNNW <- function(x,y,z,k,p){
  result = c()
  for (query in 1:length(z))
  {
    # First, find the nearest k points
    indices <- which(rank(d(x,z[query],p))<=k)
    # Access the value and feature of those k points
    label <- y[indices]
    feature <- x[indices]
    pred = 0
    denominator = 0
    # Calculate the weights
    for (i in 1:length(label)){
      denominator = denominator + exp(-d(feature[i], z[query], p))
    }
    for (i in 1:length(label)){
      pred = pred + label[i]*exp(-d(feature[i], z[query], p)) / denominator
    }
    result = c(result, pred)
  }
  return(result)
}

# Note that in the simple scenario, L1 norm = L2 norm
set.seed(12345)
x <- runif(-2, 2, n = 100)
f <- function(u){
  return(sin(pi*u) + u^2)
}
y <- f(x) + rnorm(length(x), sd = 0.2)
# # one query point, k = 10 neighbors, manhattan distance
# yhat = kNNW(x, y, z = 0, k = 10, p = 1)
# # one query point, k = 50 neighbors, manhattan distance
# yhat = kNNW(x, y, z = 0, k = 50, p = 1)
# # various query points, k = 50 neighbors, manhattan distance
# yhat = kNNW(x, y, z = c(-0.5, 0, 0.5), k = 50, p = 1)
```

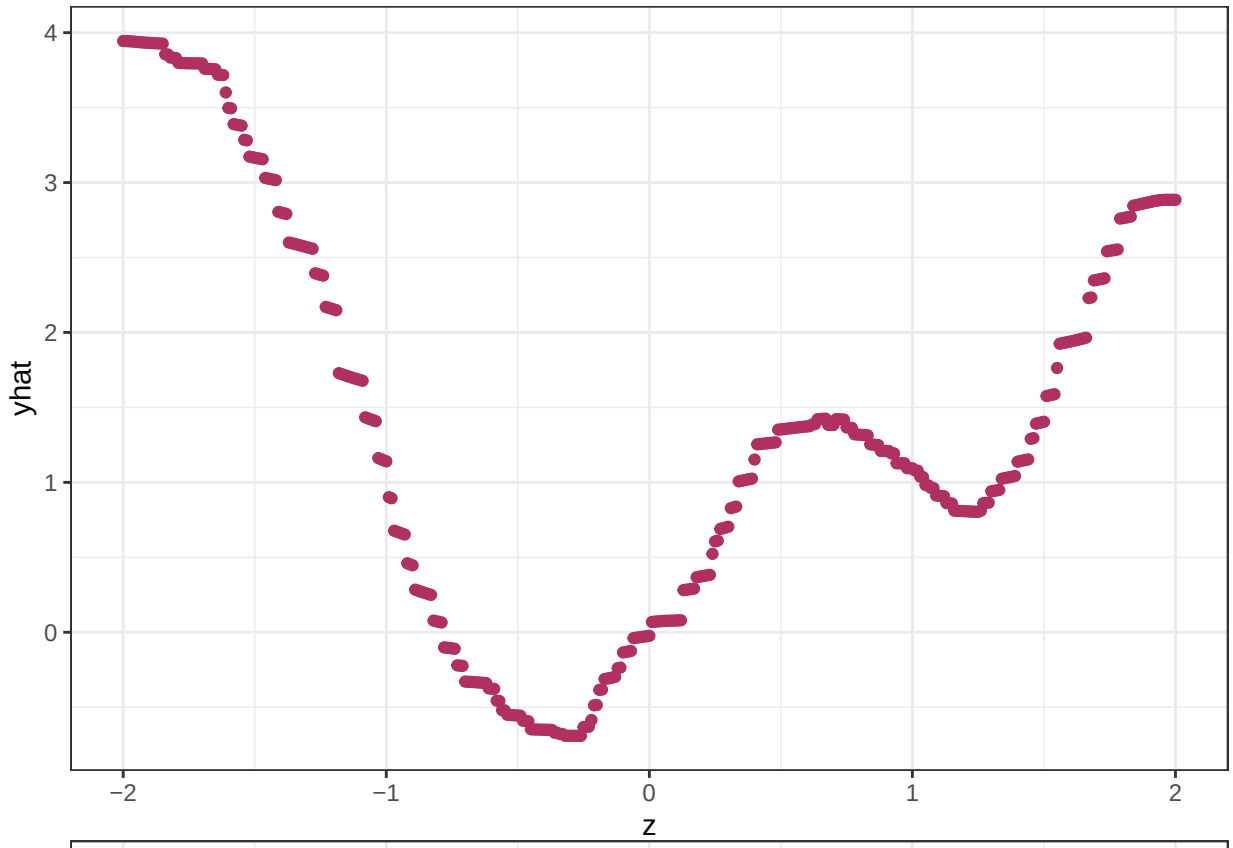
```
# Generate some test data
z = seq(-2, 2, 0.01)
yhat = kNNW(x, y, z = z, k = 1, p = 1)
yreal <- f(z) + rnorm(length(z), sd = 0.2)
ggplot() + theme_bw() +
  geom_point(aes(x = z, y = yhat), color = "maroon")
```

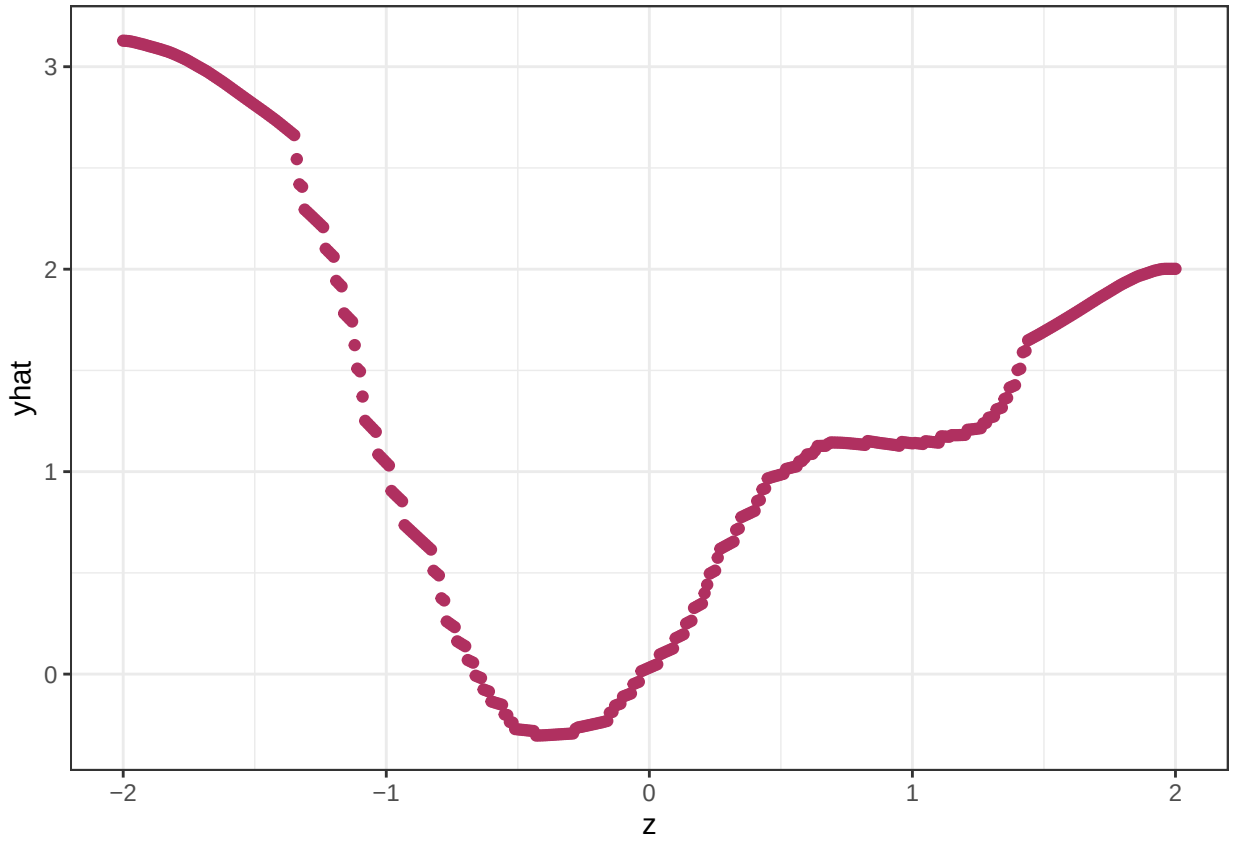


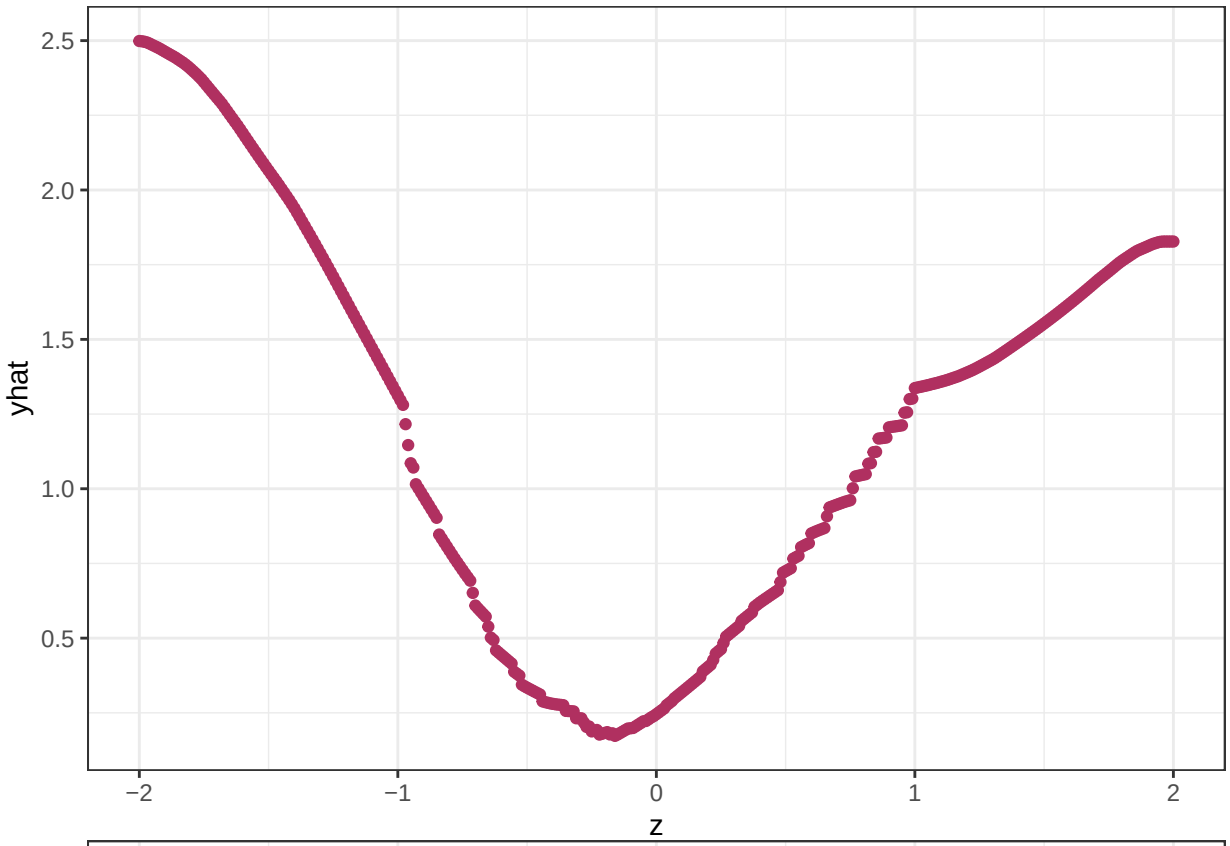

```
ggplot() + theme_bw() +
  geom_point(aes(x = z, y = yreal), color = "maroon")
```

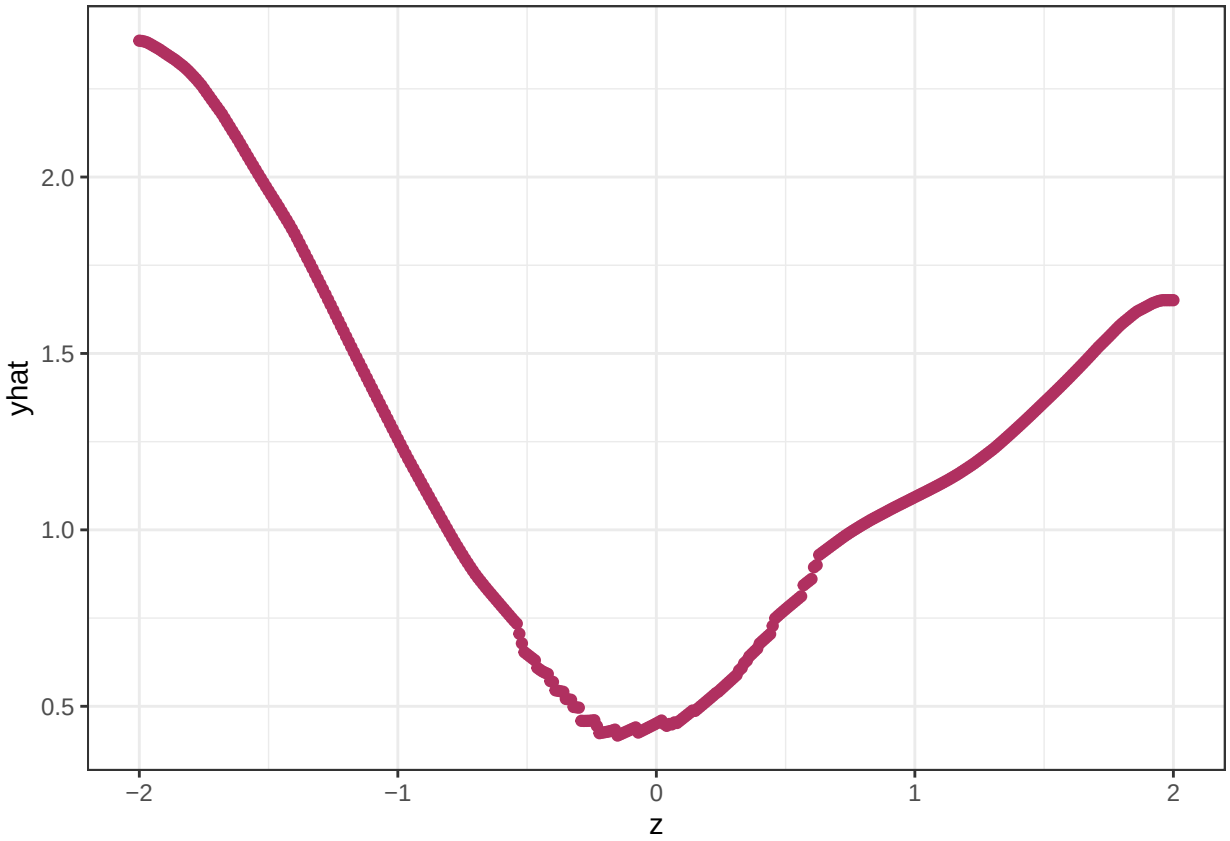


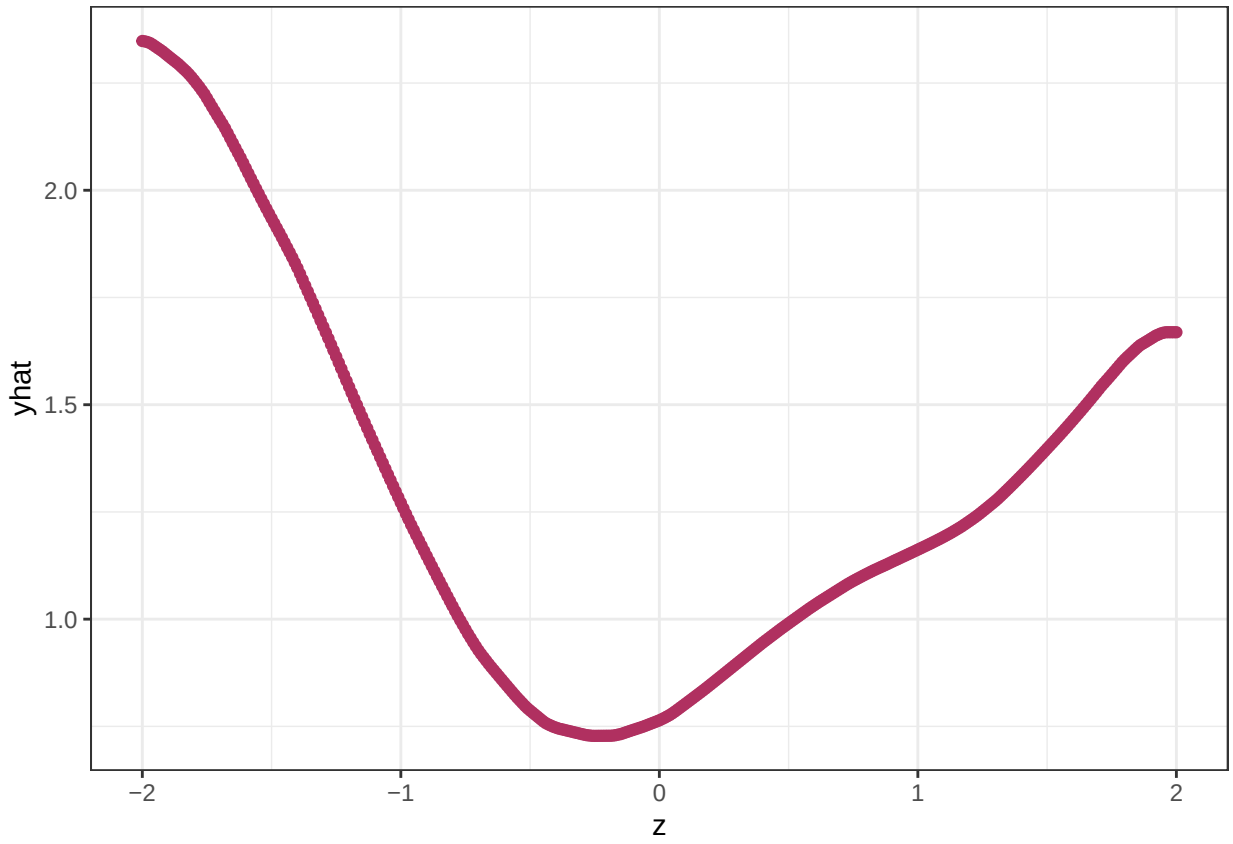

```
for (i in c(10,30,50,70,100)){
  yhat = kNNW(x, y, z = z, k = i, p = 1)
  print(ggplot() + theme_bw() +
    geom_point(aes(x = z, y = yhat), color = "maroon"))
}
```


yhat
4
3
2
1
0
−2
−1
0
1
2
z


yhat
3
2
1
0
−2
−1
0
1
2
z

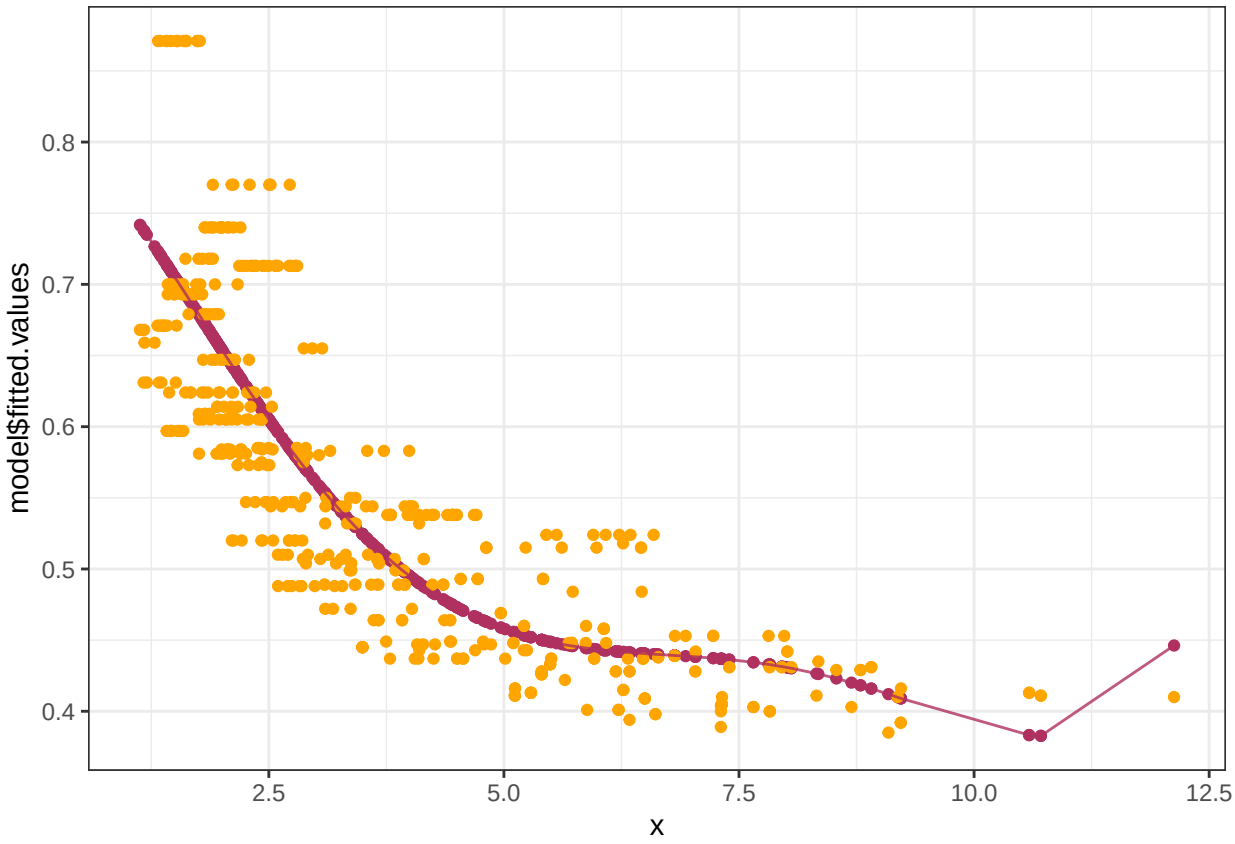

2.5
2.0
1.5
1.0
0.5
yhat
−2
−1
0
1
2
z


2.0
1.5
1.0
0.5
yhat
−2
−1
0
1
2
z

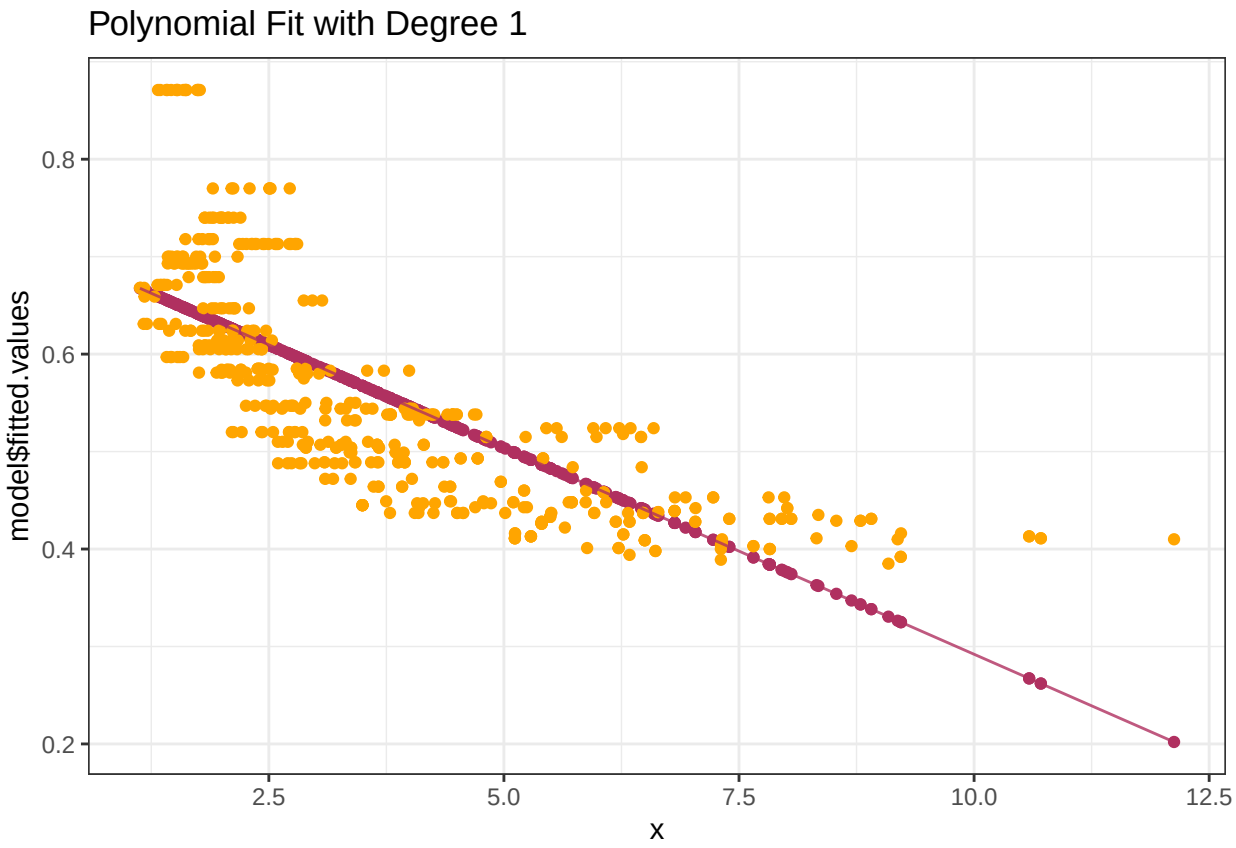



```
# Problem 6
data(Boston)
trainingSet <- Boston %>% dplyr::select(dis, nox)
x = trainingSet$dis
y = trainingSet$nox
```

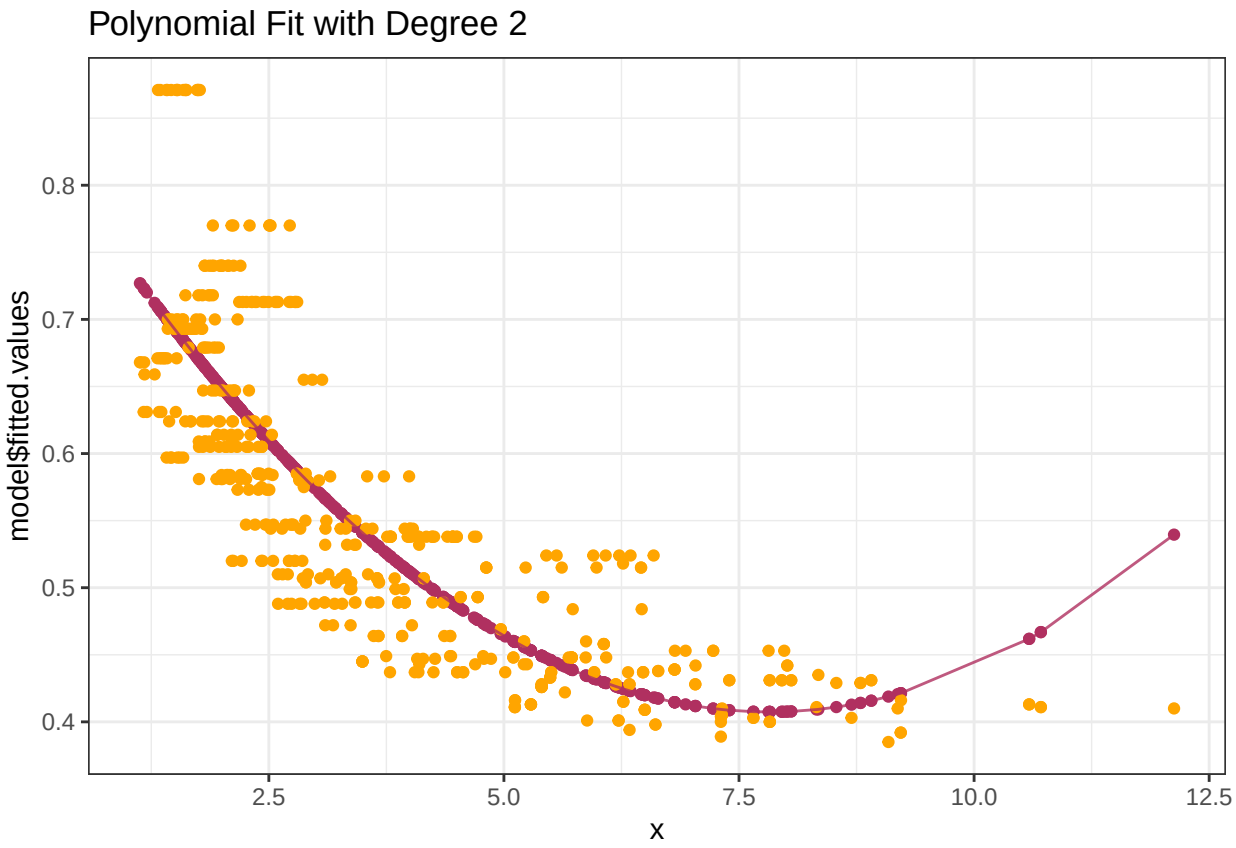
```
# Q1: Fit a model with, say, degree 5
model <- lm(y~poly(x, degree = 5, raw = T))
# stargazer(model)
ggplot() + theme_bw() + geom_point(aes(x = x, y = model$fitted.values), color = "maroon") + geom_point(a
```



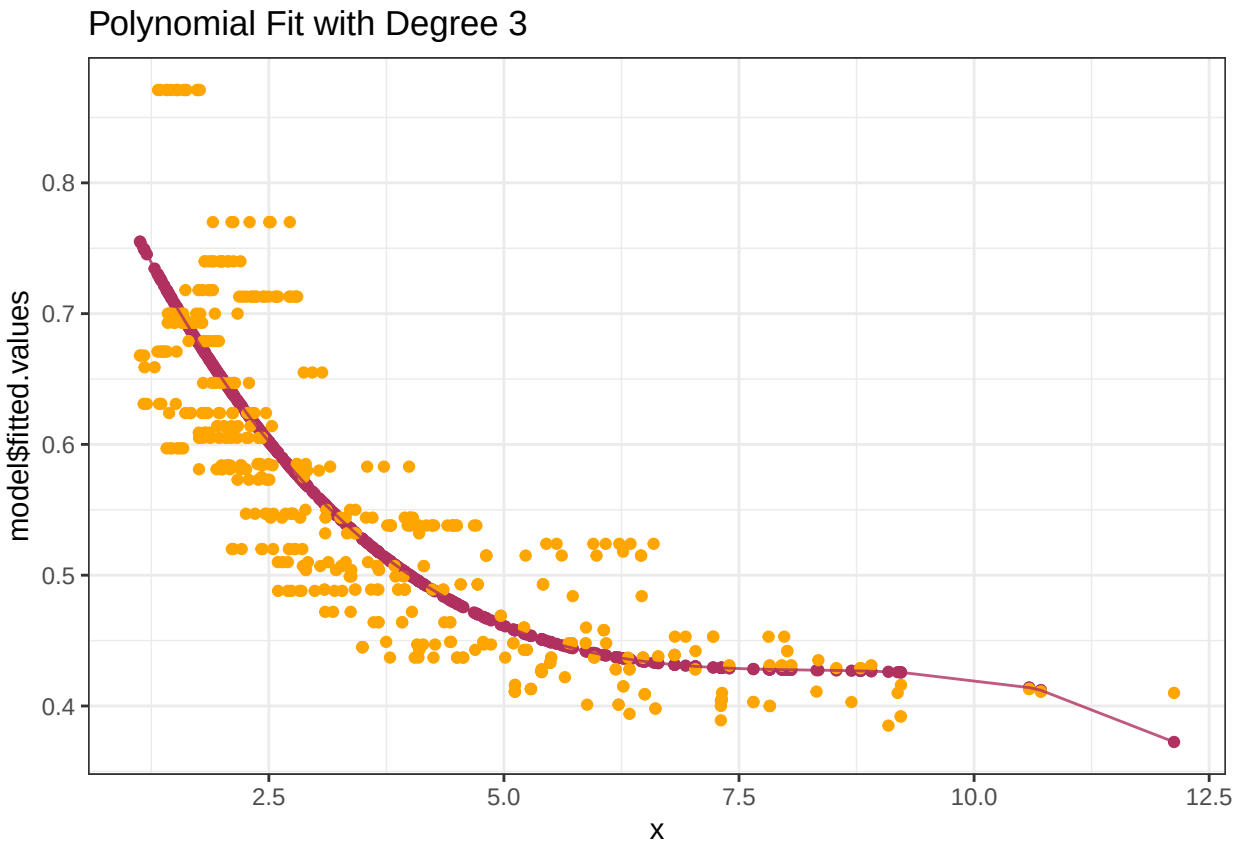

```
# Q2: Plot more!
RSS = c()
for (i in 1:10){
  model <- lm(y~poly(x, degree = i, raw = T))
  print(ggplot() + theme_bw() + geom_point(aes(x = x, y = model$fitted.values), color = "maroon") + geo
  RSS = c(RSS, sum(model$residuals^2))
  print(paste("The RSS of the model with degrees ", i, " is ", RSS[i],".", sep = ""))
}
```



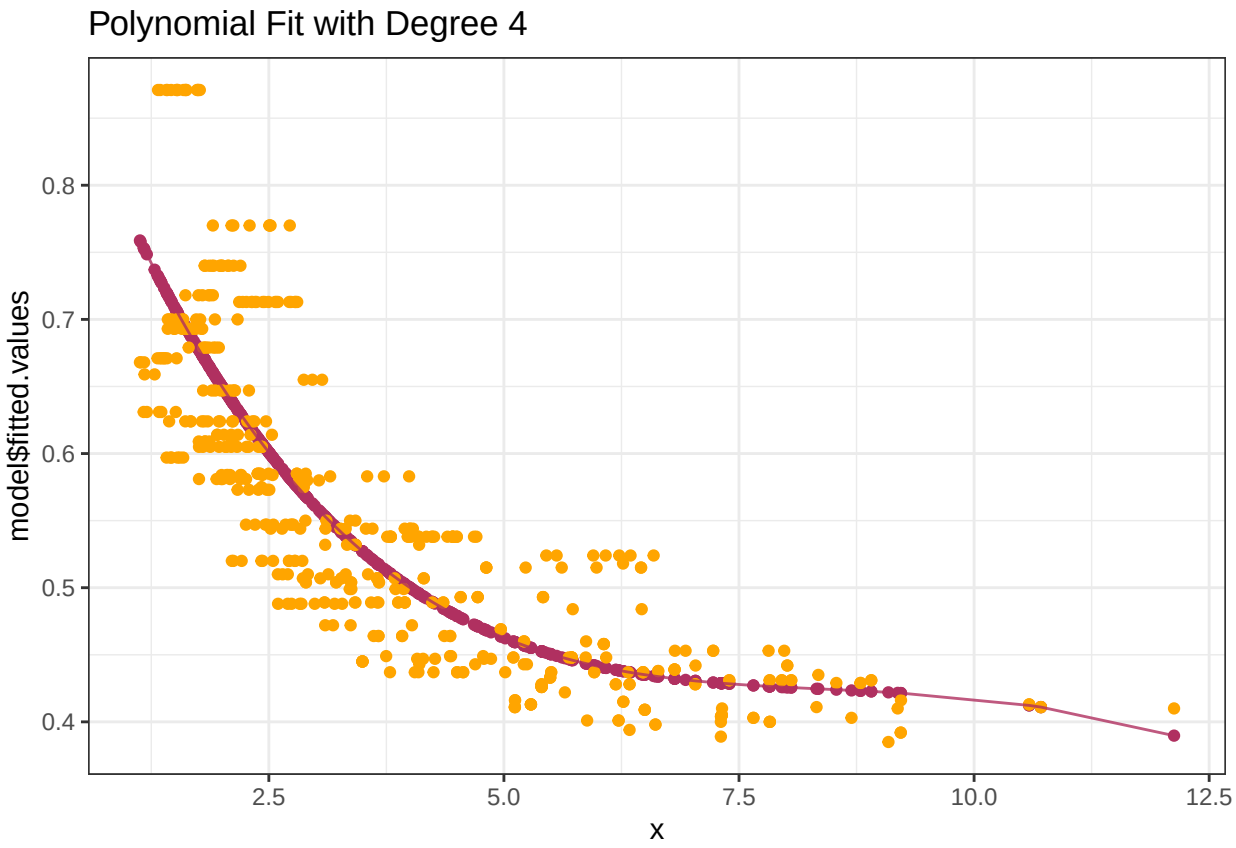

```
## [1] "The RSS of the model with degrees 1 is 2.76856285896928."
```



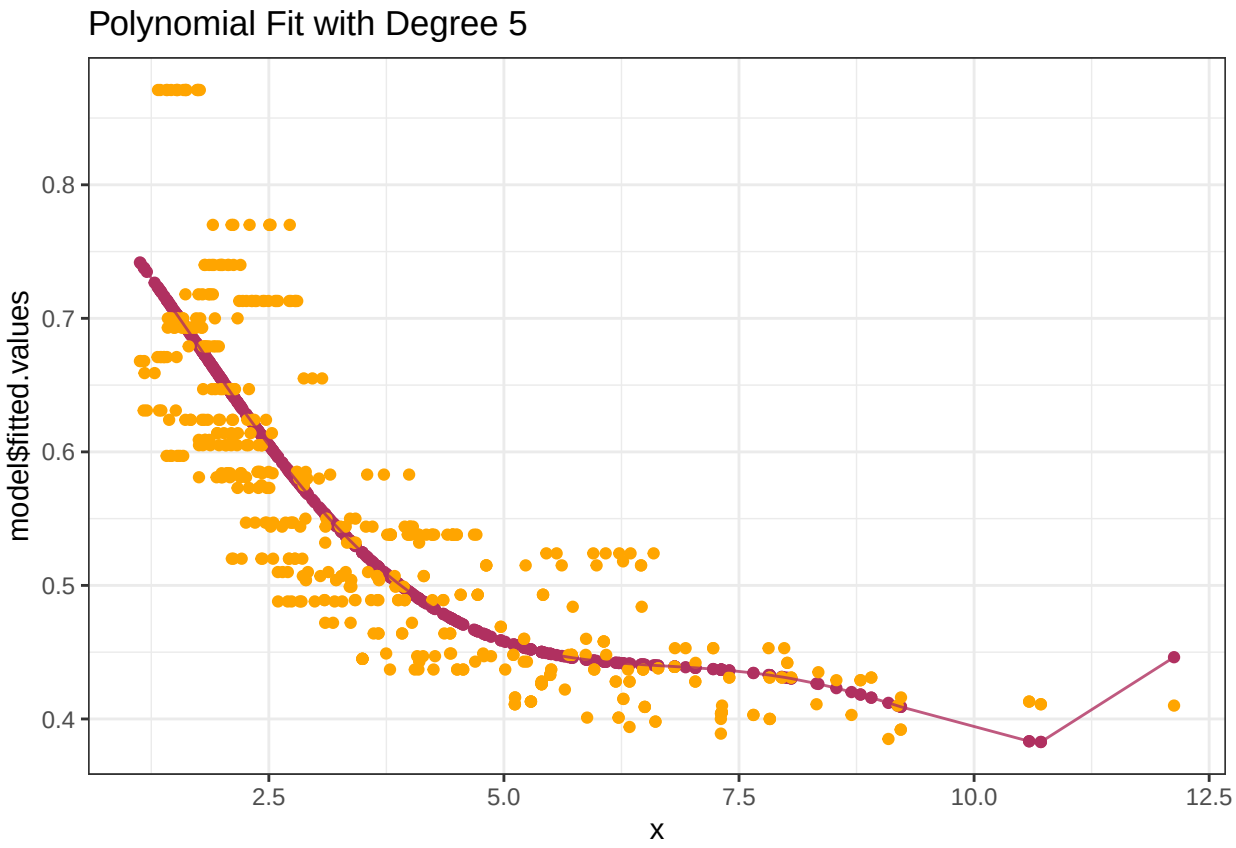

```
## [1] "The RSS of the model with degrees 2 is 2.03526186893526."
```



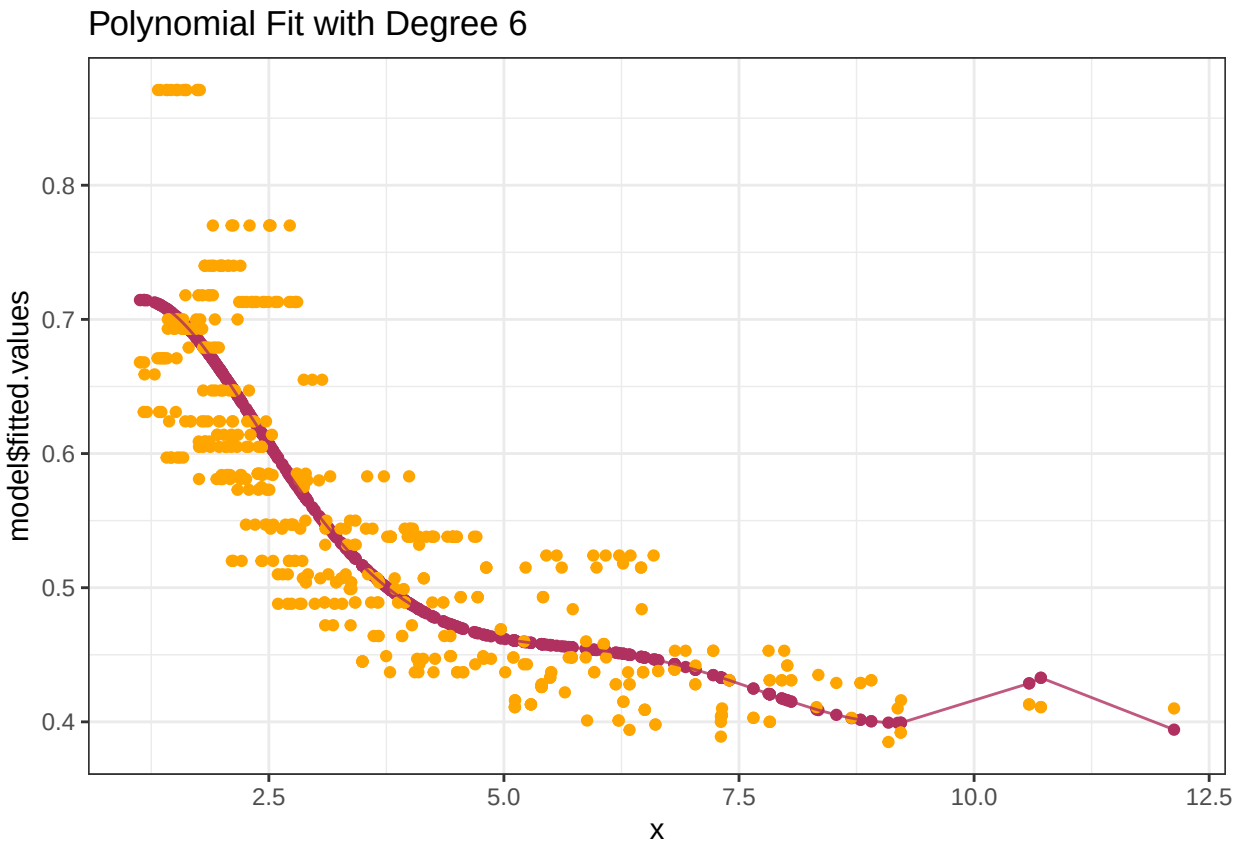

```
## [1] "The RSS of the model with degrees 3 is 1.93410670717907."
```



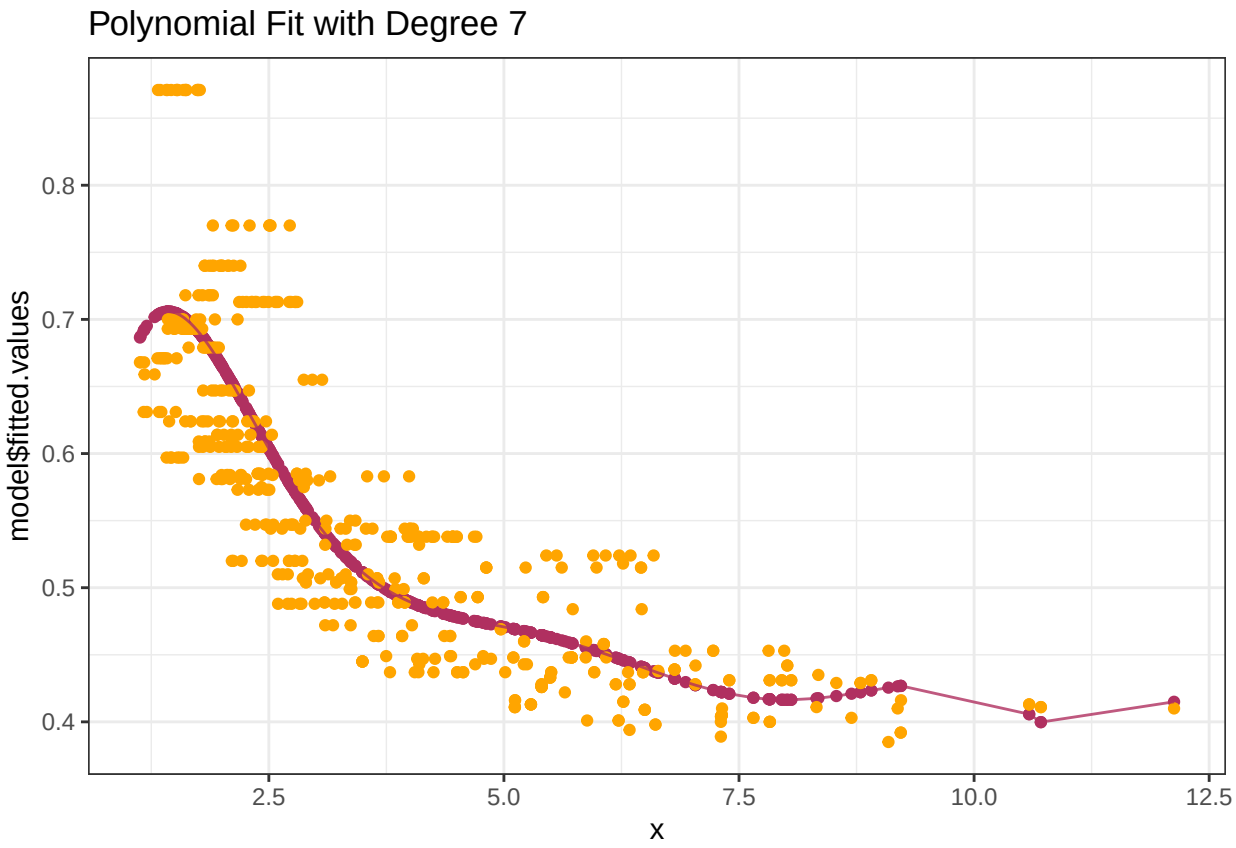

```
## [1] "The RSS of the model with degrees 4 is 1.93298132729859."
```



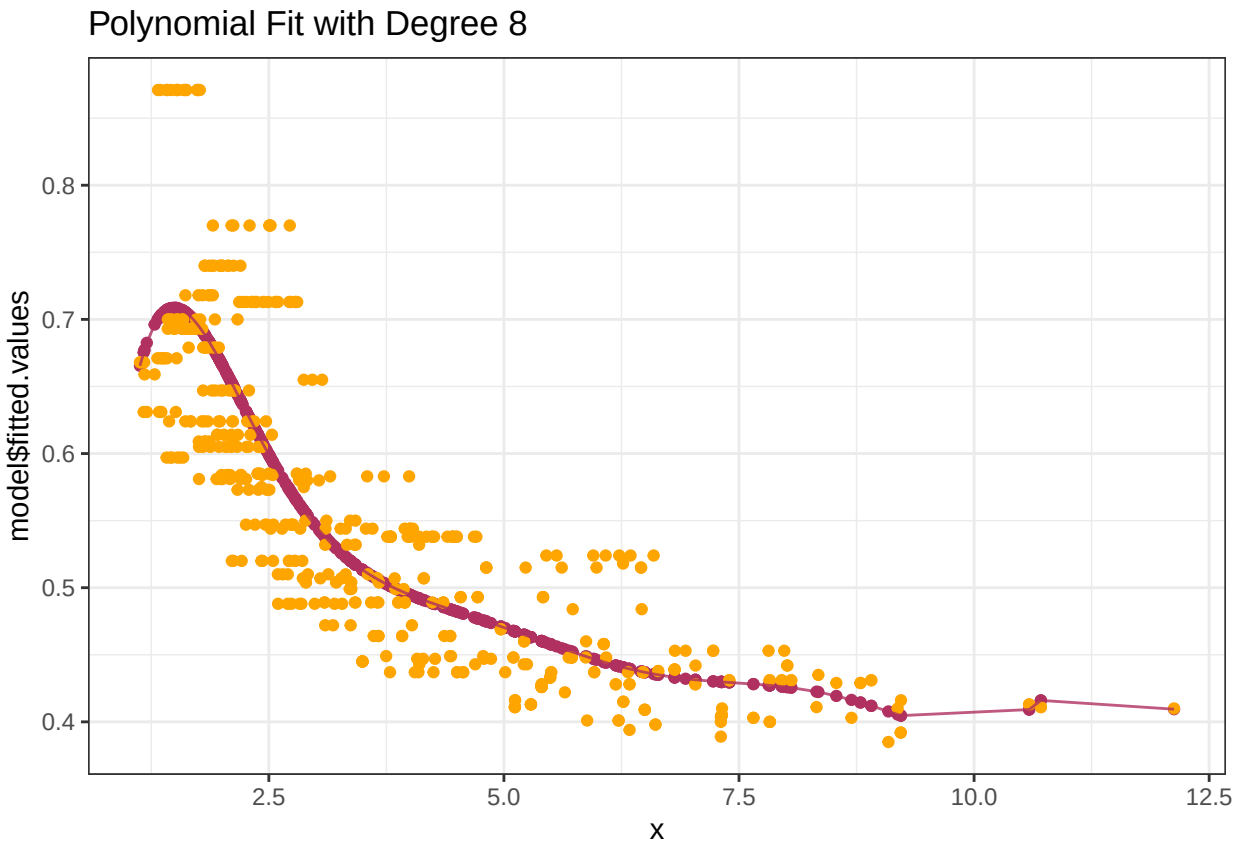

```
## [1] "The RSS of the model with degrees 5 is 1.9152899610843."
```



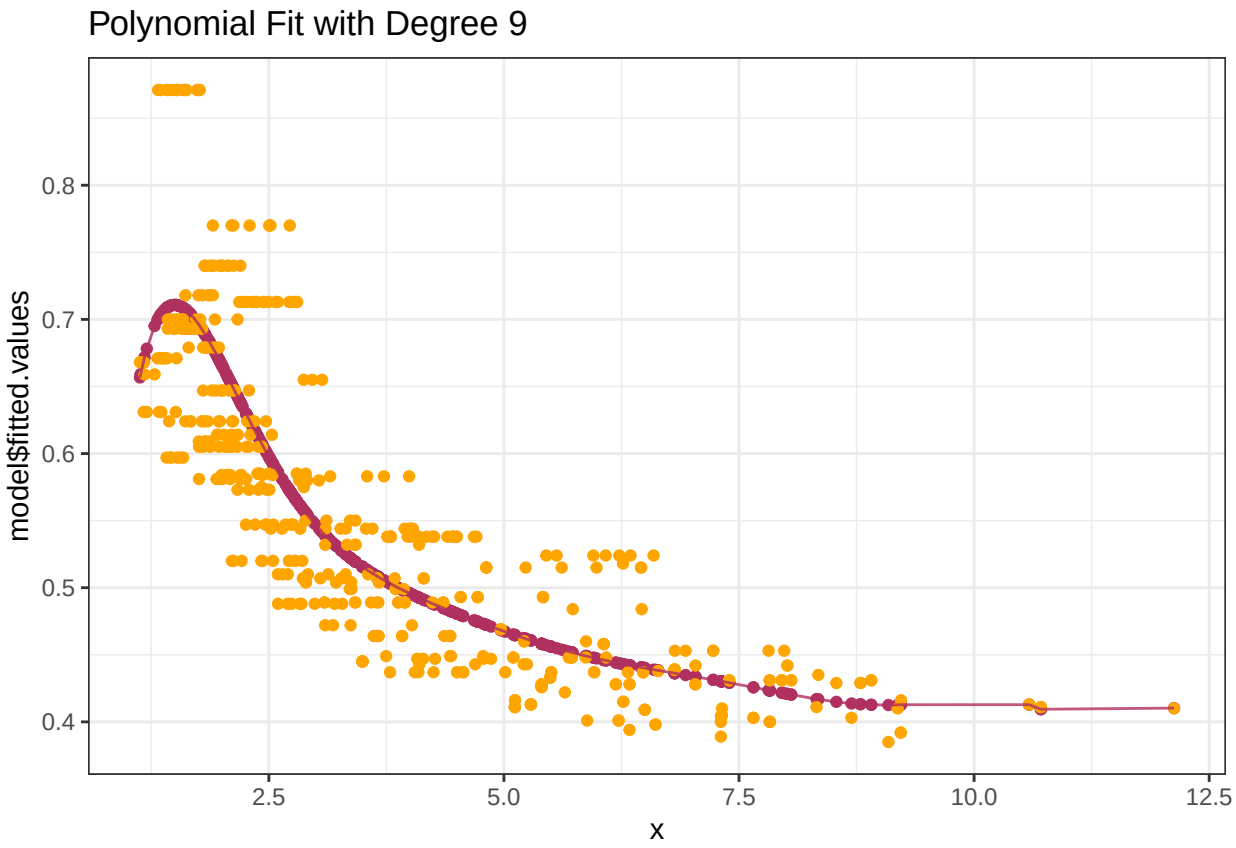

```
## [1] "The RSS of the model with degrees 6 is 1.87825729850817."
```



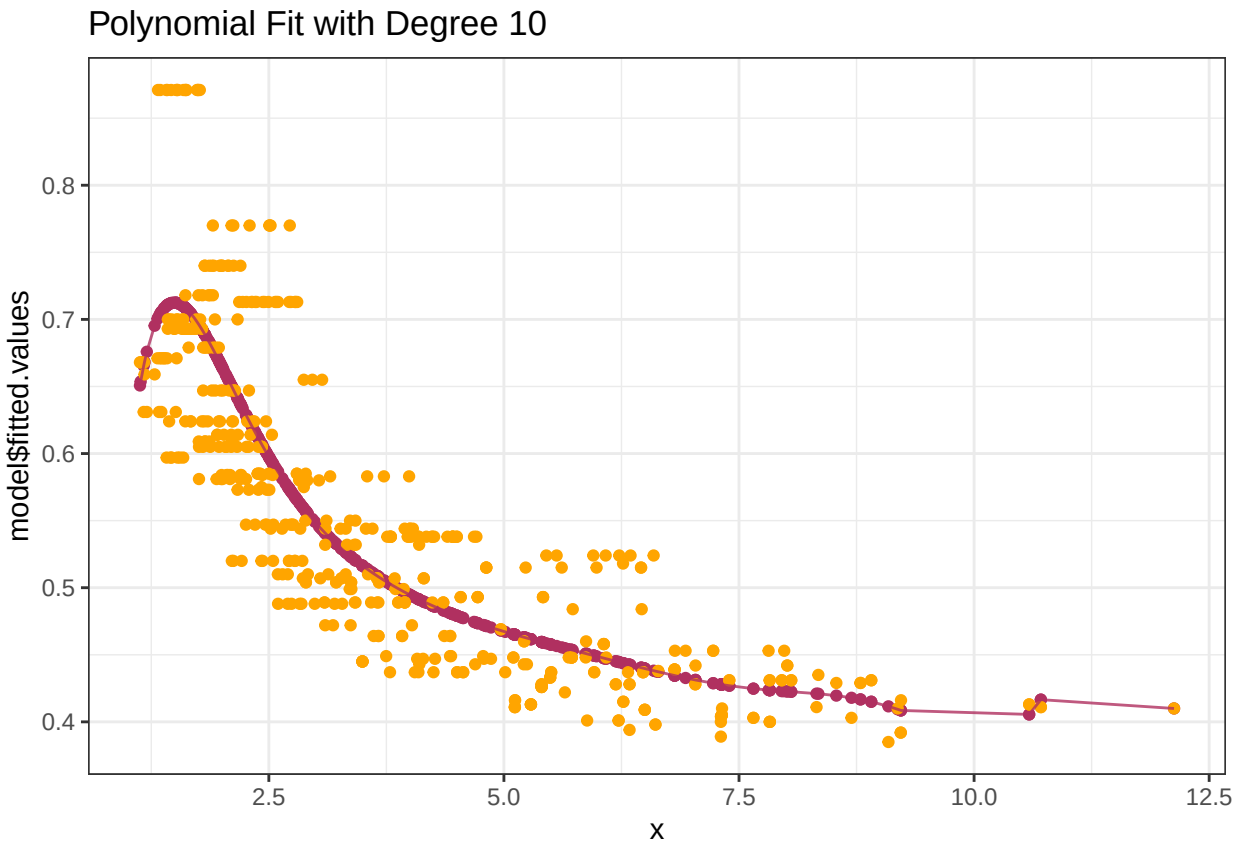

```
## [1] "The RSS of the model with degrees 7 is 1.84948361458297."
```




```
## [1] "The RSS of the model with degrees 8 is 1.83562968906761."
```

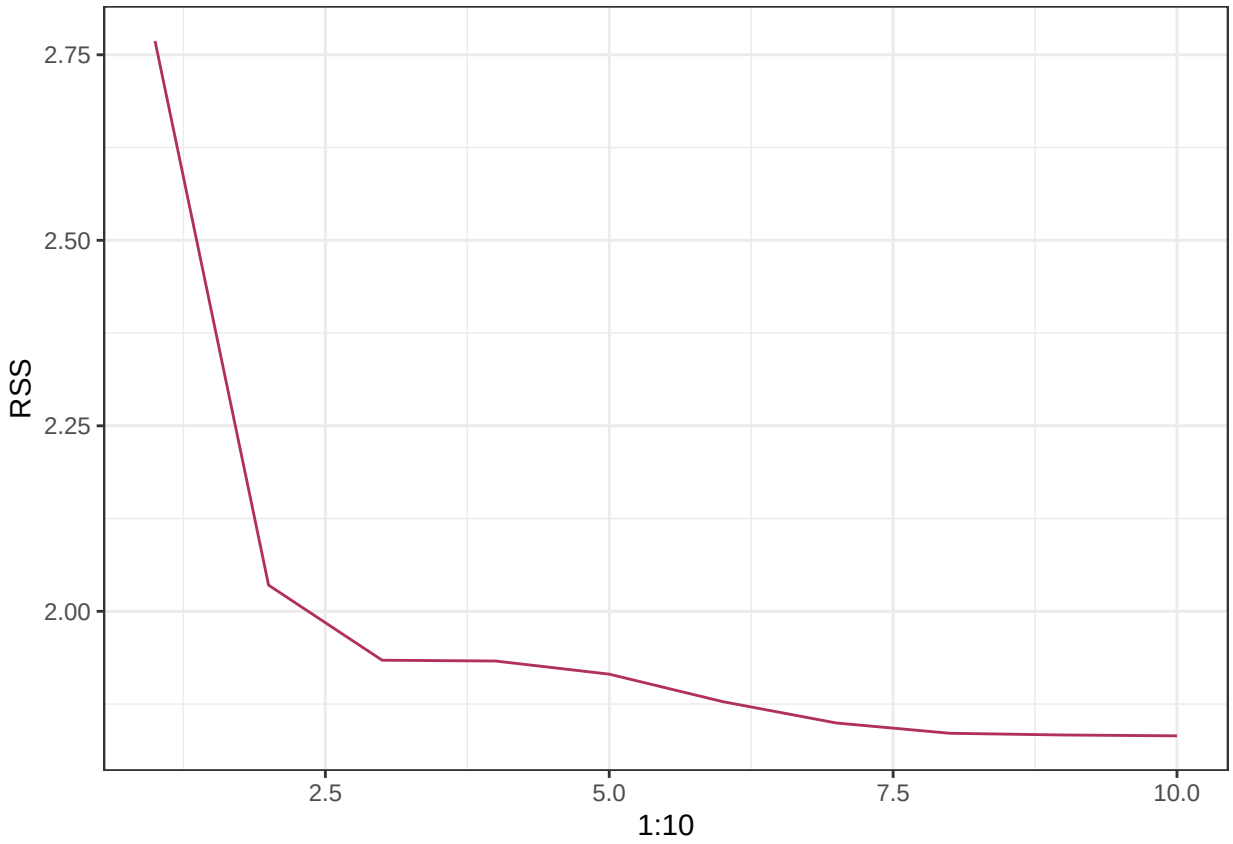



```
## [1] "The RSS of the model with degrees 9 is 1.83333080449161."
```




```
## [1] "The RSS of the model with degrees 10 is 1.83217112393014."
```

```
ggplot() + theme_bw() + geom_line(aes(x = 1:10, y = RSS), color = "maroon")
```



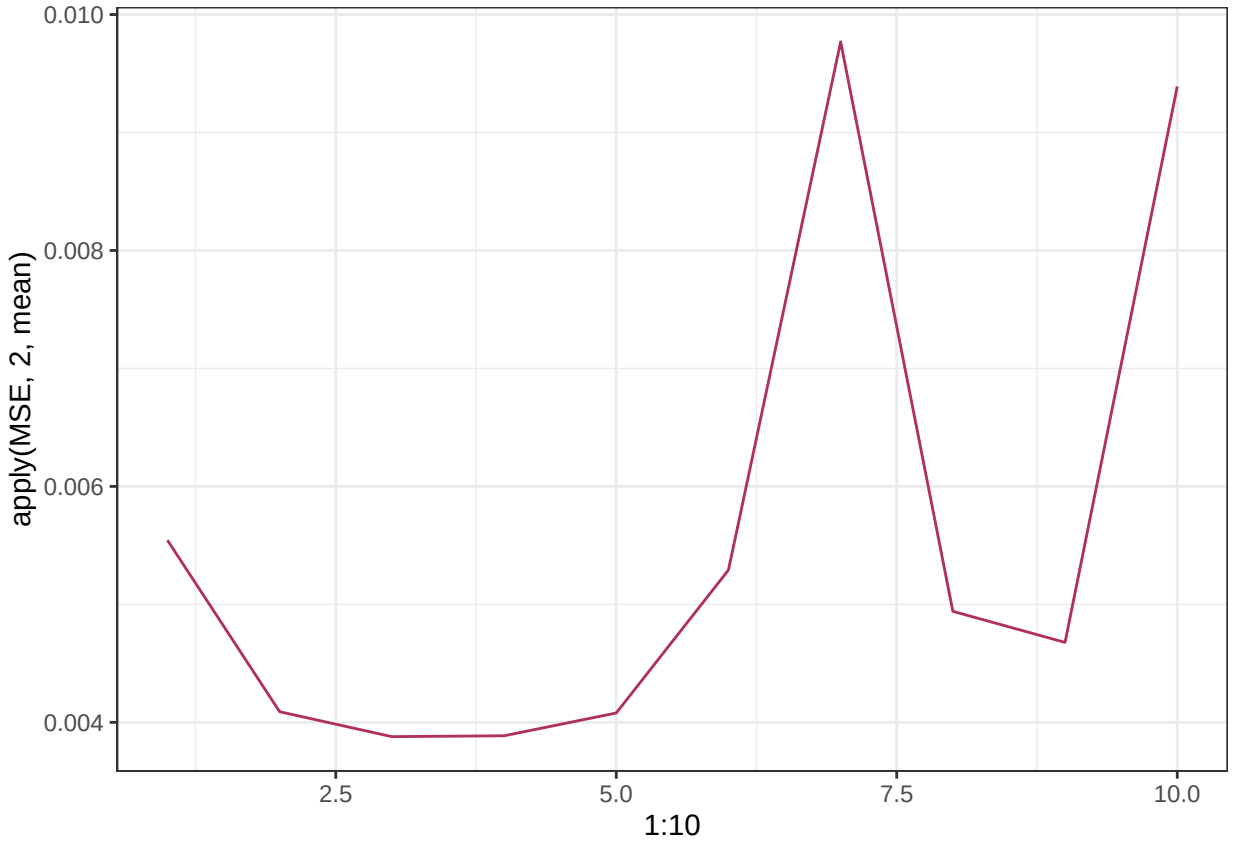

```
set.seed(11111)
# Use Cross-Validation
trainingSet <- trainingSet %>%
  mutate(instant = 1:nrow(trainingSet), fold = 0)
tempdata <- trainingSet

for (i in 1:5){
  num = 100
  if (i == 5){num = 106}
  temp <-  sample_n(tempdata, num)
  tempdata <- tempdata %>%
    filter(!(instant %in% temp$instant))
  trainingSet[trainingSet$instant %in% temp$instant,] <- trainingSet %>%
    filter(instant %in% temp$instant) %>%
    mutate(fold = i)
}
```

```
MSE = matrix(0, nrow = 5, ncol = 10)
for (i in 1:5){
  train = trainingSet %>% filter(fold != i)
  test = trainingSet %>% filter(fold == i)
  for (j in 1:10){
    model = lm(nox ~ poly(dis, degree = j, raw = T), data = train)
    yfit = cbind(1, poly(test$dis, degree = j, raw = T)) %*% model$coefficients
    MSE[i,j] = mean((test$nox - yfit)^2)
  }
}
```

```r
which.min(apply(MSE, 2, mean))
```

```
## [1] 3
```

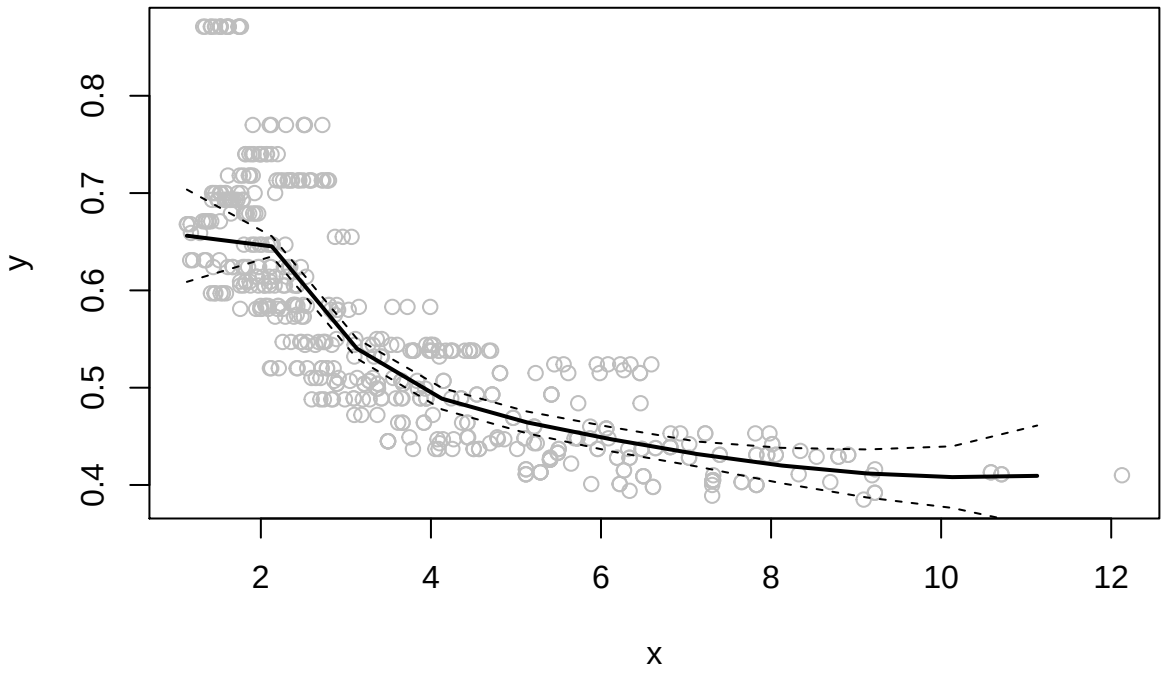
```r
# The best model is the 3rd degree polynomial
ggplot() + geom_line(aes(x = 1:10, y = apply(MSE, 2, mean)), color = "maroon") + theme_bw()
```



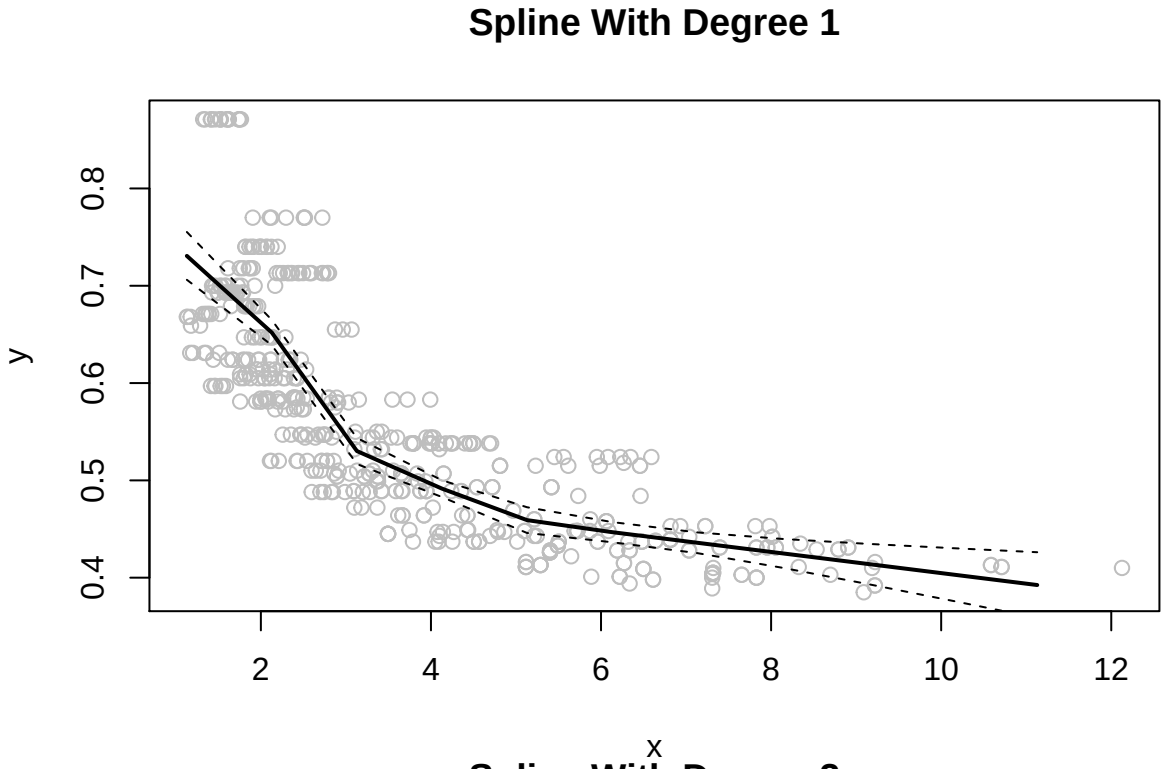

```r
# The knots are chosen by the quantiles of the data
quantile(x)
```

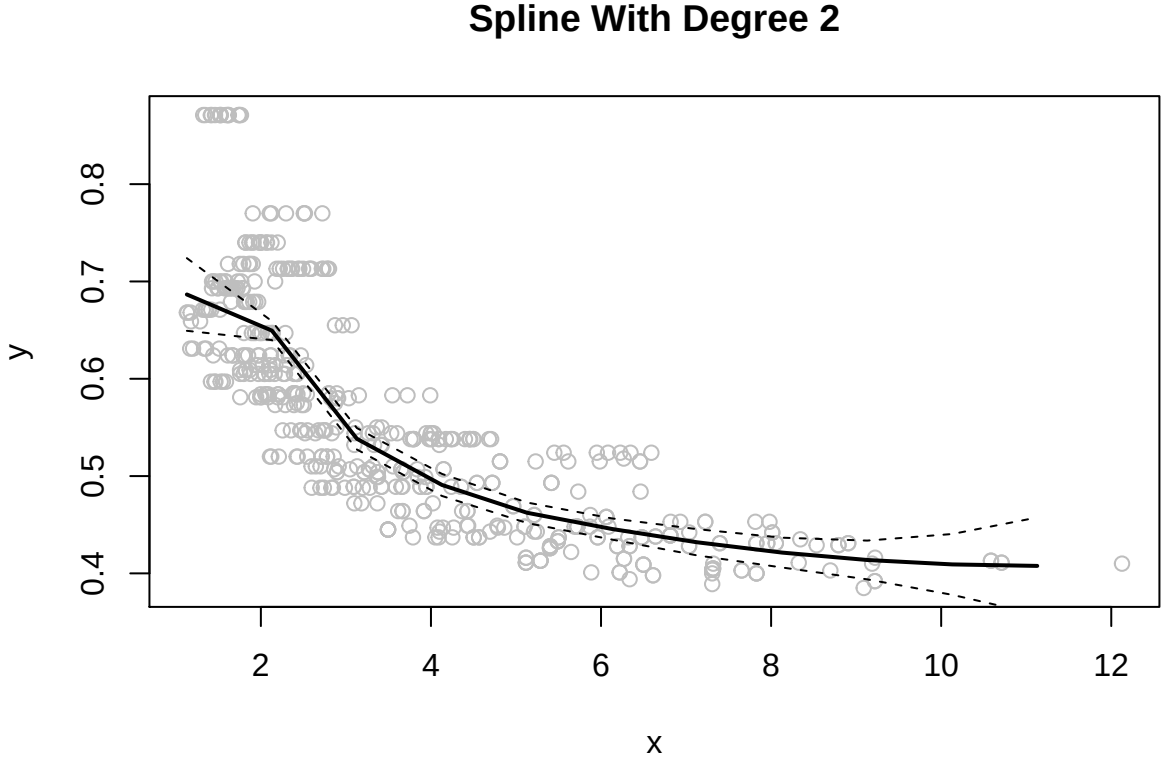
```
##        0%       25%       50%       75%      100%
##  1.129600  2.100175  3.207450  5.188425 12.126500
```

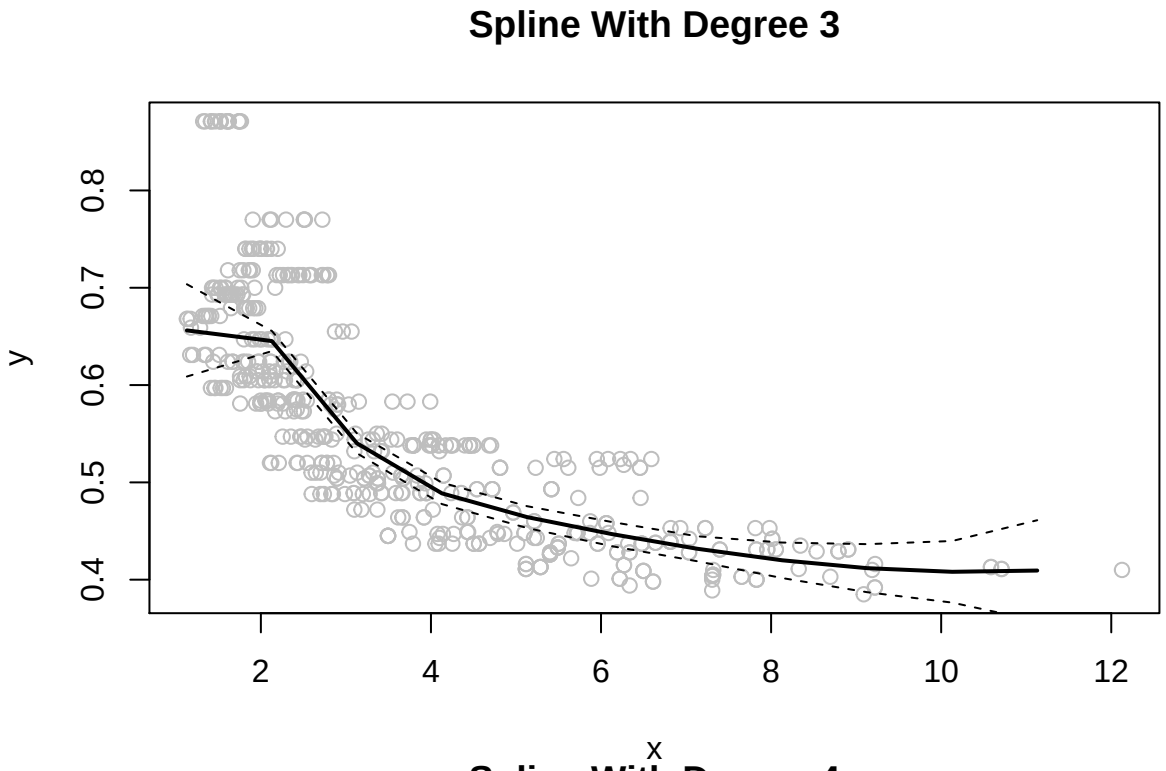
```r
model <- lm(nox~bs(dis,knots=c(2.1,3.2,5.2)),data=trainingSet)

attach(trainingSet)
preds=predict(model,newdata=list(dis=seq(from = min(dis), to = max(dis))),se=TRUE)

plot(x, y, col="gray")
lines(seq(from = min(dis), to = max(dis)),preds$fit,lwd=2)
lines(seq(from = min(dis), to = max(dis)),(preds$fit +2*preds$se) ,lty="dashed")
lines(seq(from = min(dis), to = max(dis)),(preds$fit -2*preds$se) ,lty="dashed")
```

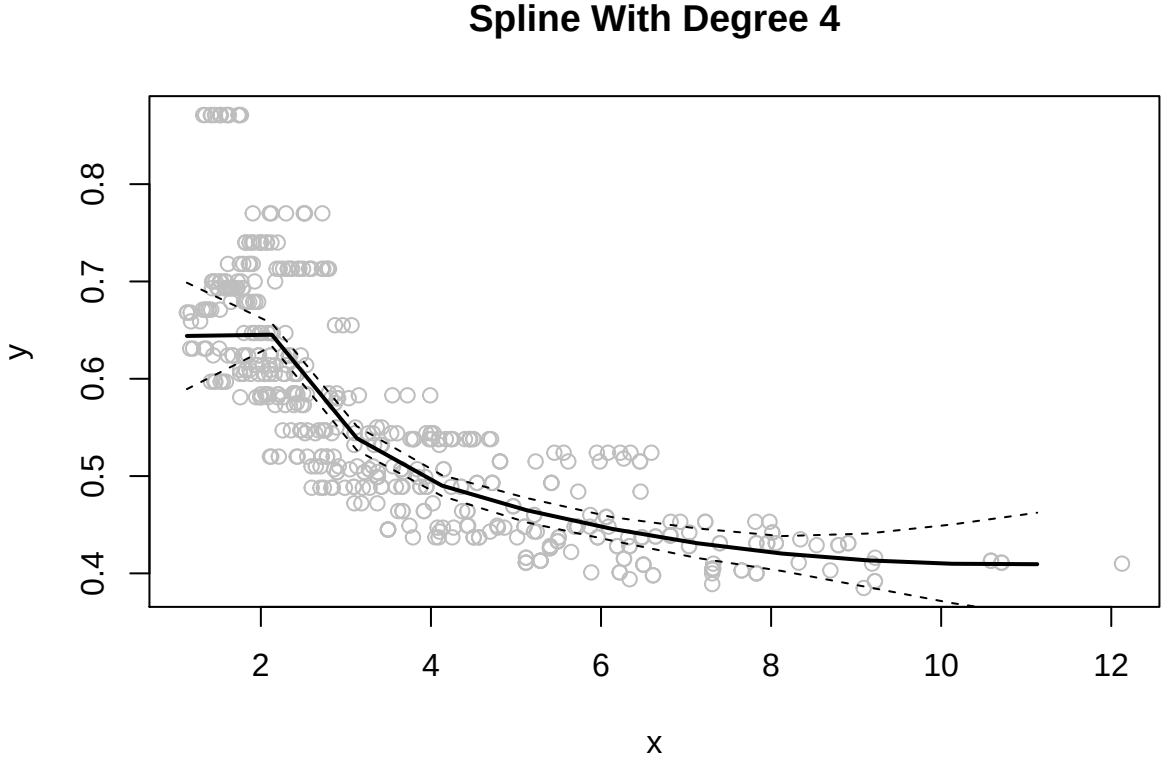


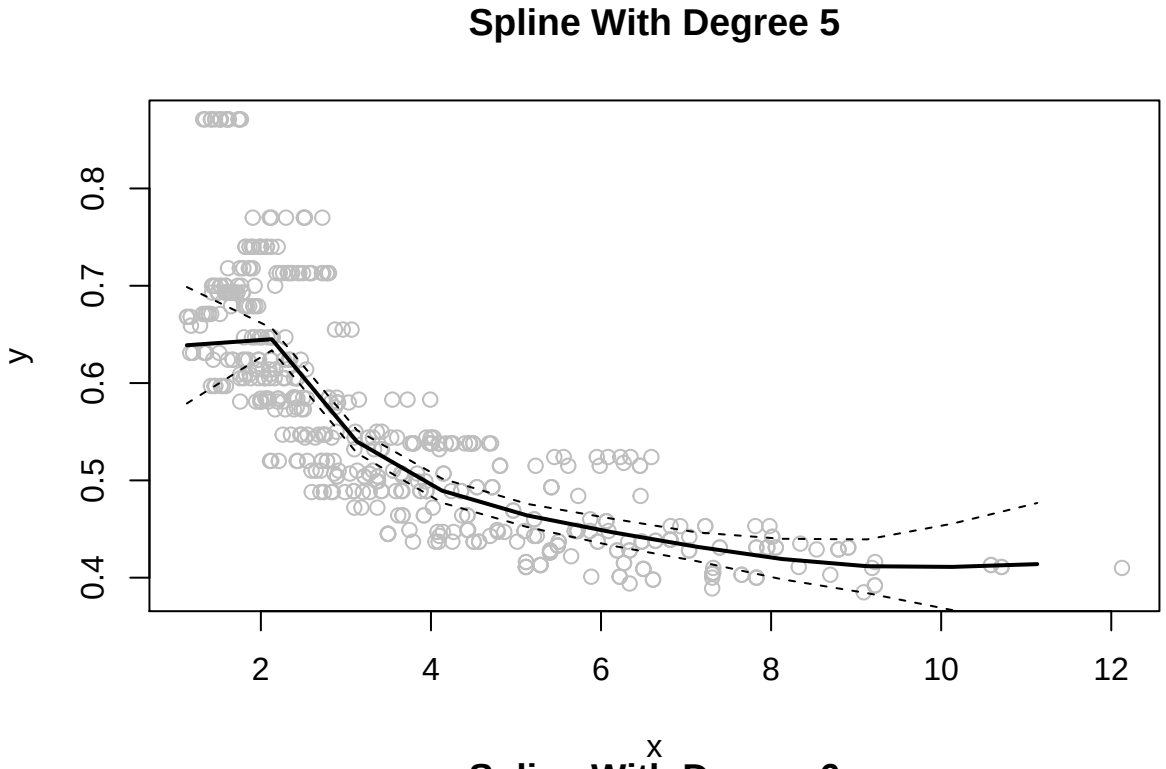

```r
# Note that in the plot we only choose some sample points,
# presenting a broken line instead of a curve
for (i in 1:10){
  model <- lm(nox~bs(dis,knots=c(2.1,3.2,5.2), degree = i),data=trainingSet)
  preds=predict(model,newdata=list(dis=seq(from = min(dis), to = max(dis))),se=TRUE)

  plot(x, y, col="gray")
  lines(seq(from = min(dis), to = max(dis)),preds$fit,lwd=2)
  lines(seq(from = min(dis), to = max(dis)),(preds$fit +2*preds$se) ,lty="dashed")
  lines(seq(from = min(dis), to = max(dis)),(preds$fit -2*preds$se) ,lty="dashed")
  title(paste("Spline With Degree ", i, sep = ""))
}
```

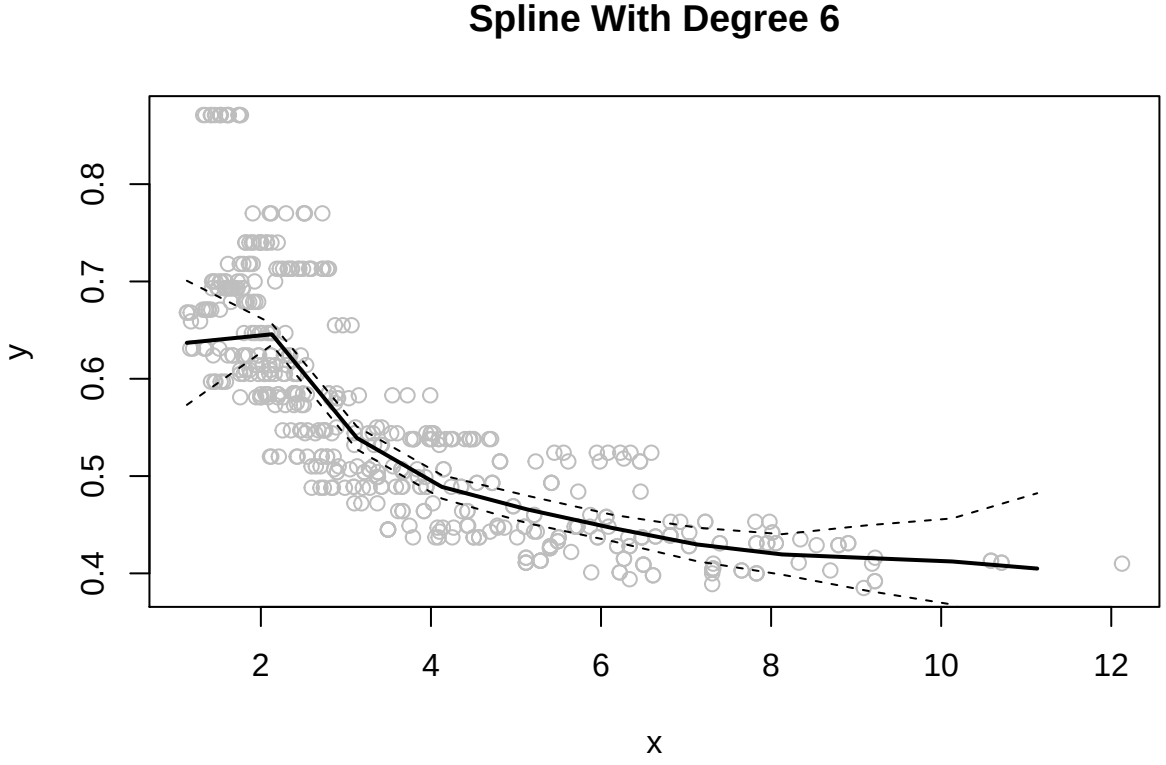

Spline With Degree 1
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x

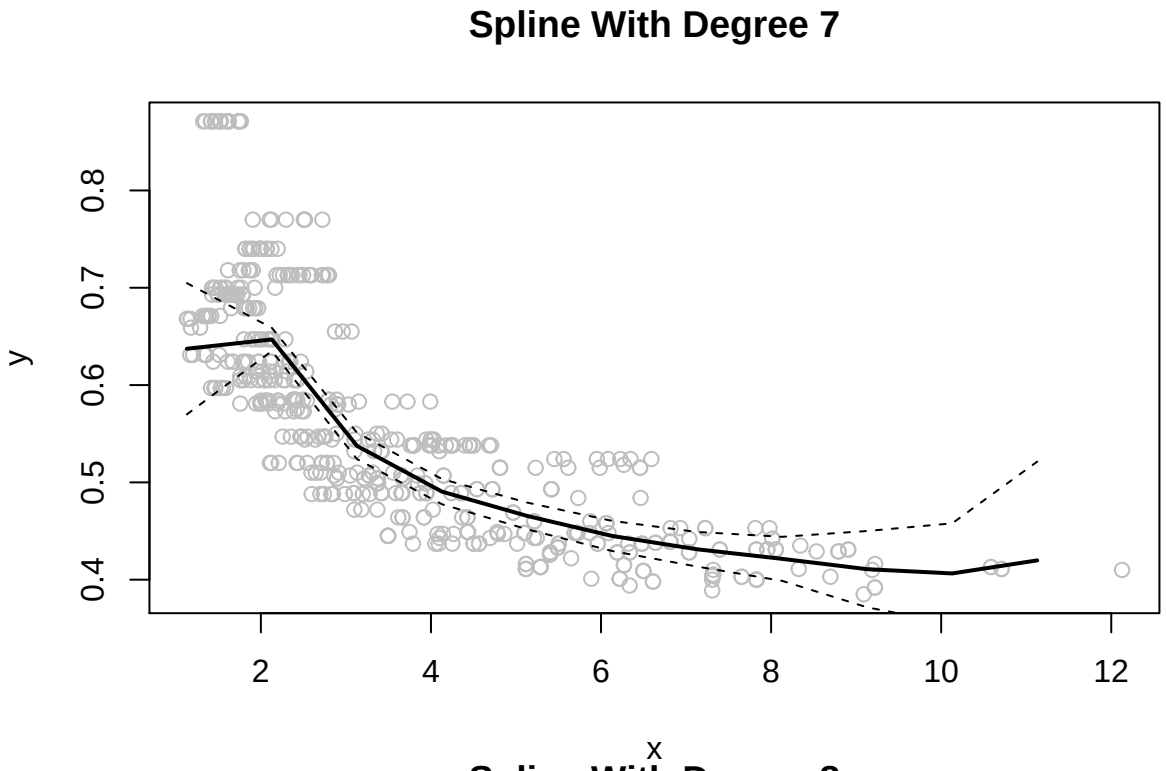

Spline With Degree 2
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x

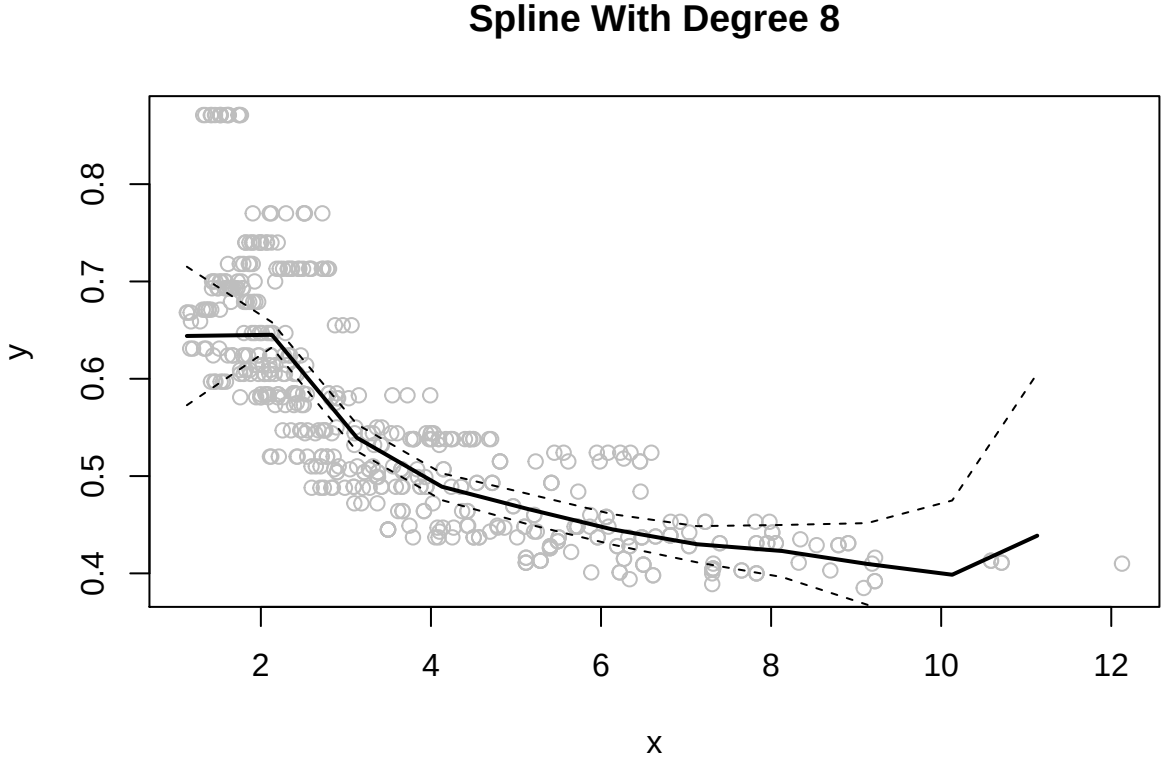

Spline With Degree 3
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x

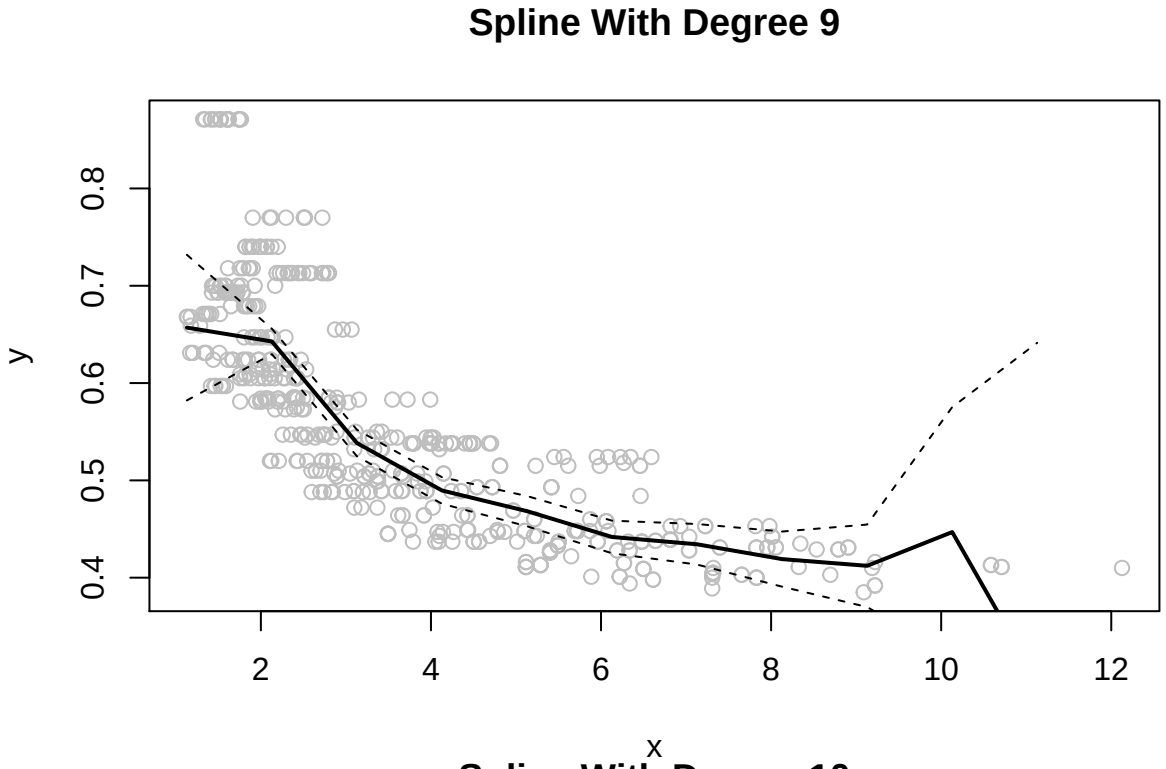

Spline With Degree 4
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x

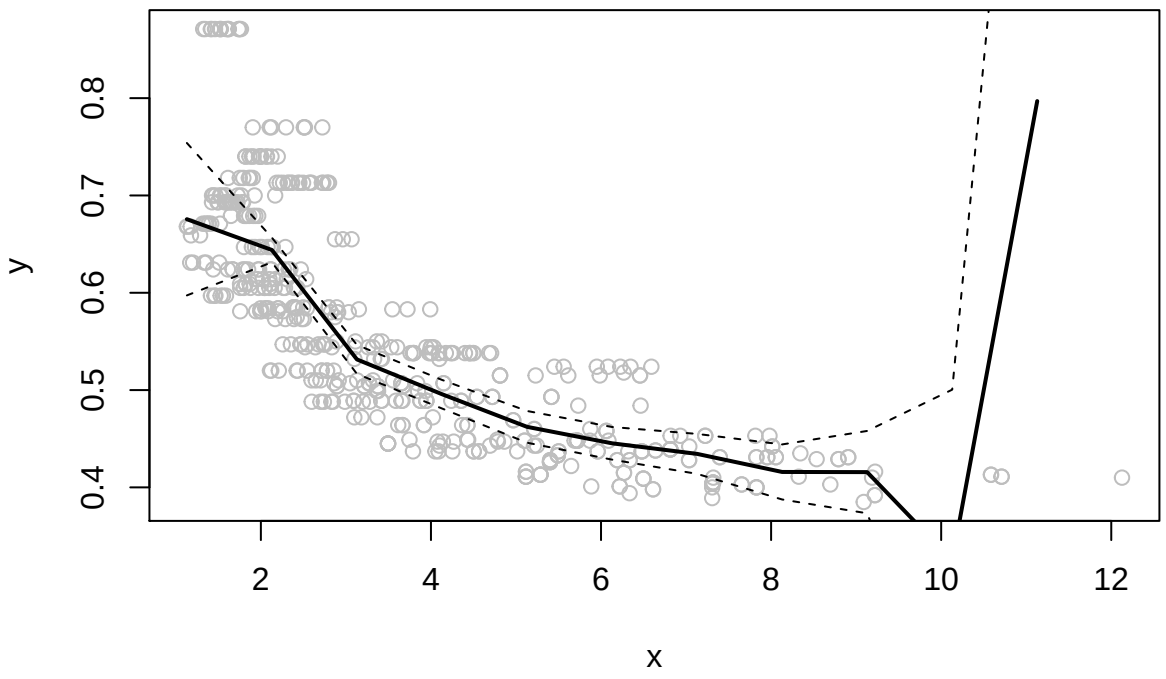

Spline With Degree 5
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x


Spline With Degree 6
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x


Spline With Degree 7
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x


Spline With Degree 8
y
0.4
0.5
0.6
0.7
0.8
2
4
6
8
10
12
x






```
model <- smooth.spline(x = train$dis, y = train$nox)
temp <- predict(model, test$dis)
# Cross-validation
MSE = matrix(0, nrow = 5, ncol = 9)
for (i in 1:5){
```

```
  train = trainingSet %>% filter(fold != i)
  test = trainingSet %>% filter(fold == i)
  for (j in 2:10){
    model <- smooth.spline(x = train$dis, y = train$nox, df = j)
    yfit = predict(model, test$dis)$y
    MSE[i,j-1] = mean((test$nox - yfit)^2)
  }
}
which.min(apply(MSE, 2, mean)) + 1
```

```
## [1] 10
```

```
# The best model is the 10th degree smooth-spline
ggplot() + geom_line(aes(x = 2:10, y = apply(MSE, 2, mean)), color = "maroon") + theme_bw()
```