Lab 7.1 Non-linear regression

quan le

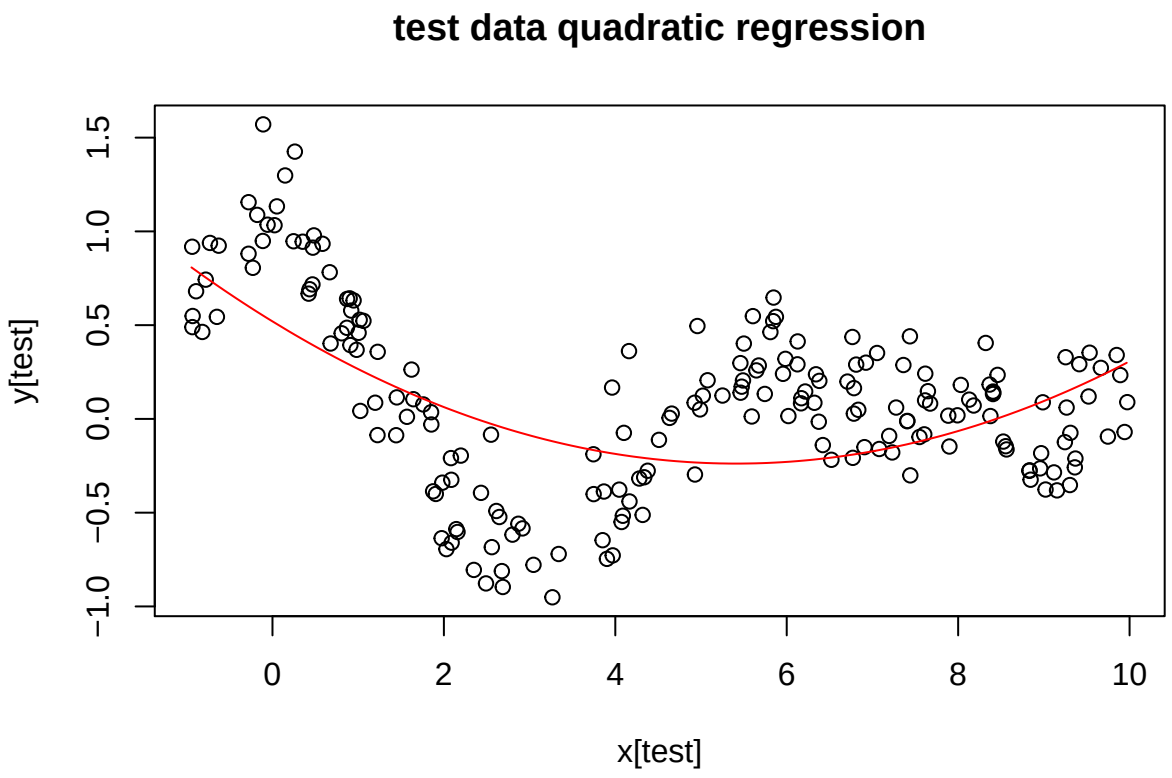
2022-09-29

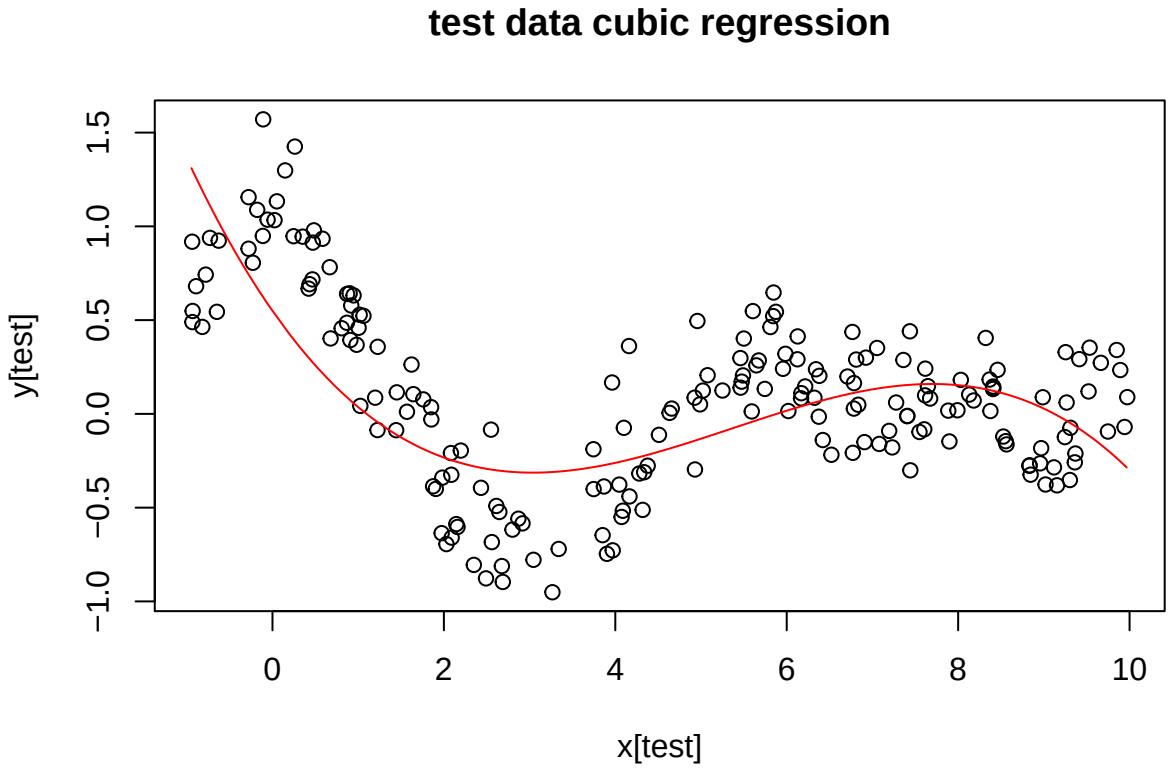
Intro to Non-linear Regression

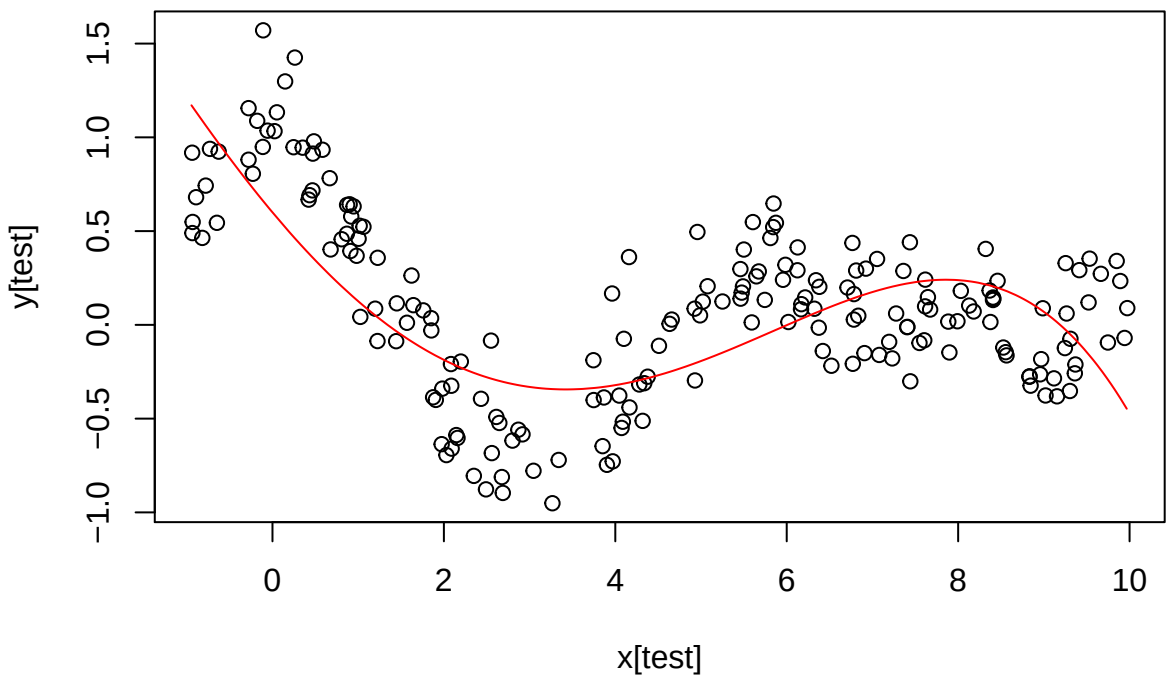
```
# generates gaussian kernel from training data Z
# k(x,z) = exp(-(x-z)^2)
gaussk = function(X, Z, gamma=1) {
  return(exp(-gamma * (X%*%t(rep(1,length(Z))) - rep(1,length(X))%*%t(Z))^2))
}

# generates simple polynomial kernel from training data Z
polyk = function(X, Z, d, r=1, gamma=1) {
  return((r + gamma * (X%*%t(Z)))^d)
}

# kernel ridge regression function
kreg = function(K, y, lambdas) {
  alphas = matrix(NA, nrow = ncol(K), ncol = length(lambdas))
  fit = matrix(NA, nrow = length(y), ncol = length(lambdas))
  lambda.min = lambdas[1]
  min.err = Inf
  mse = rep(0, length(lambdas))

  for (i in 1:length(lambdas)) {
    alphas[,i] = solve(K + lambdas[i]*diag(nrow(K)), y)
    fit[,i] = K %*% alphas[,i]
    mse[i] = mean((y - fit[,i])^2)
    if (mse[i] < min.err) {
      min.err = mse[i]
      lambda.min = lambdas[i]
    }
  }
  return(list("fit"=fit, "alphas"=alphas, "mse"=mse, "lambda.min"=lambda.min))
}
```

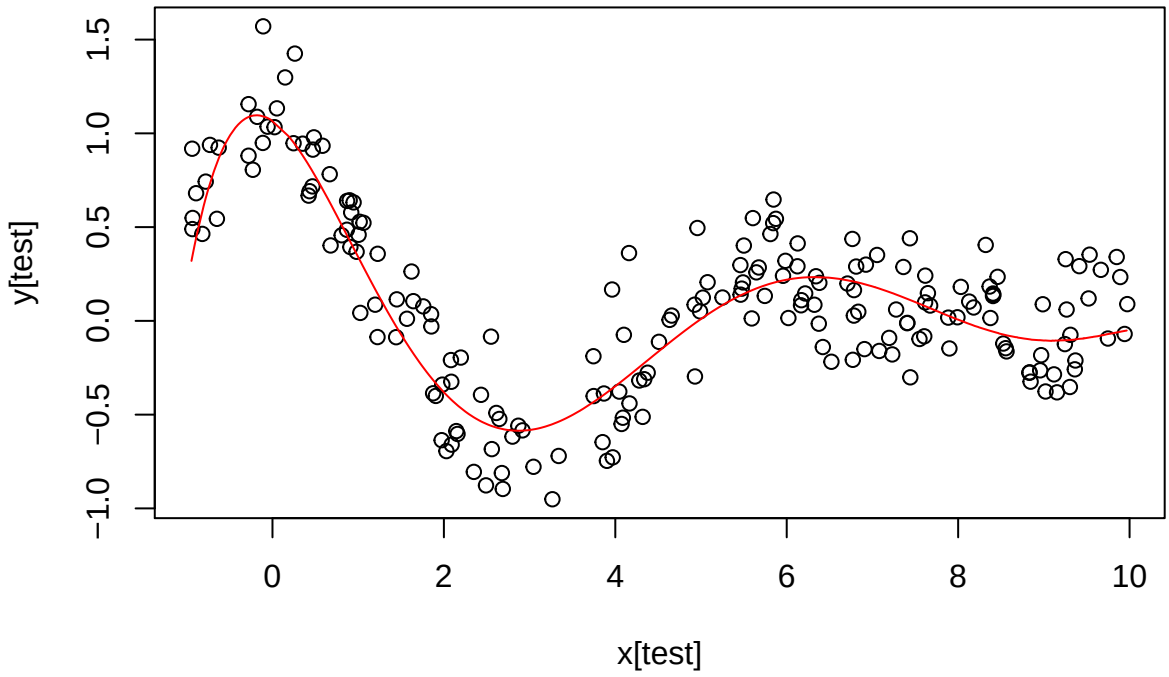
```
# generate stimulated data
n = 600
x = sort(runif(n, min = -1, max = 10))
# exponentially damped sine function + error
y = exp(-(x/5)^2)*cos(x) + rnorm(n, mean=0, sd=0.2)

grid = exp(1)^seq(10,-1,length=10) # the sequence of lambda

test = sort(sample(1:length(x), length(x)/3))
```

```r
train = (-test)

X = x[train]

polynames = c("quadratic", "cubic", "quartic", "sextic")
degrees = c(2,3,4,6)

train.mse = c()
test.mse = c()
for (i in 1:length(degrees)) {
  reg = kreg(polyk(X, X, degrees[i]), y[train], grid)
  plot(x[test], y[test], main=paste("test data", polynames[i], "regression"))
  lines(x[train], reg$fit[,match(reg$lambda.min, grid)], col="red")
  train.mse[i] = min(reg$mse)
  test.mse[i] = mean((y[test] - polyk(x[test], X, degrees[i]) %*% reg$alphas[,match(reg$lambda.min, gri
}
```




test data cubic regression
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10

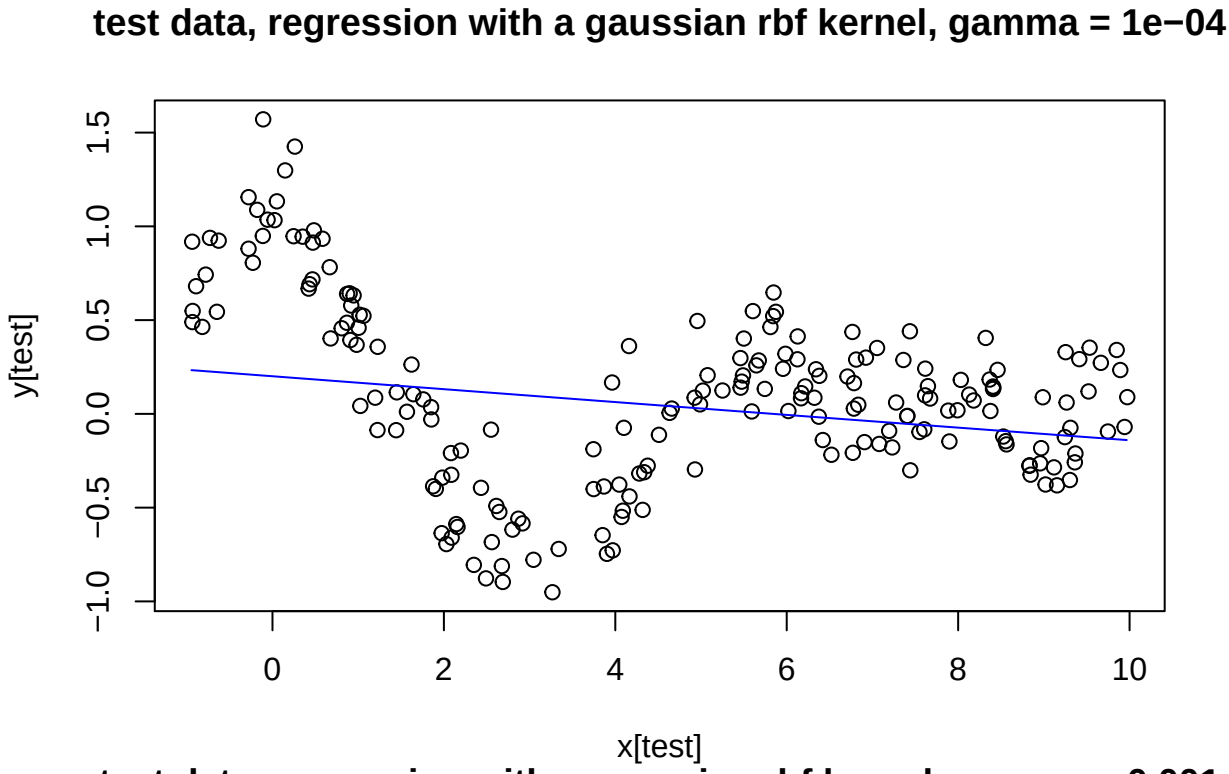

test data quartic regression
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10

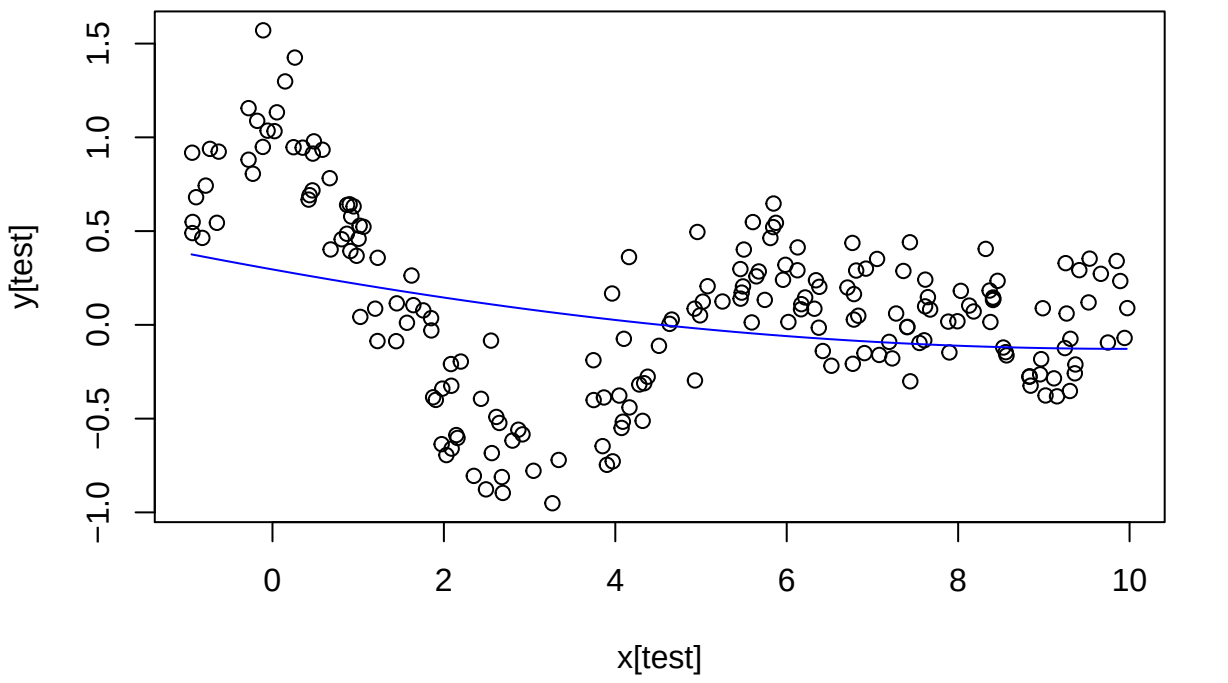


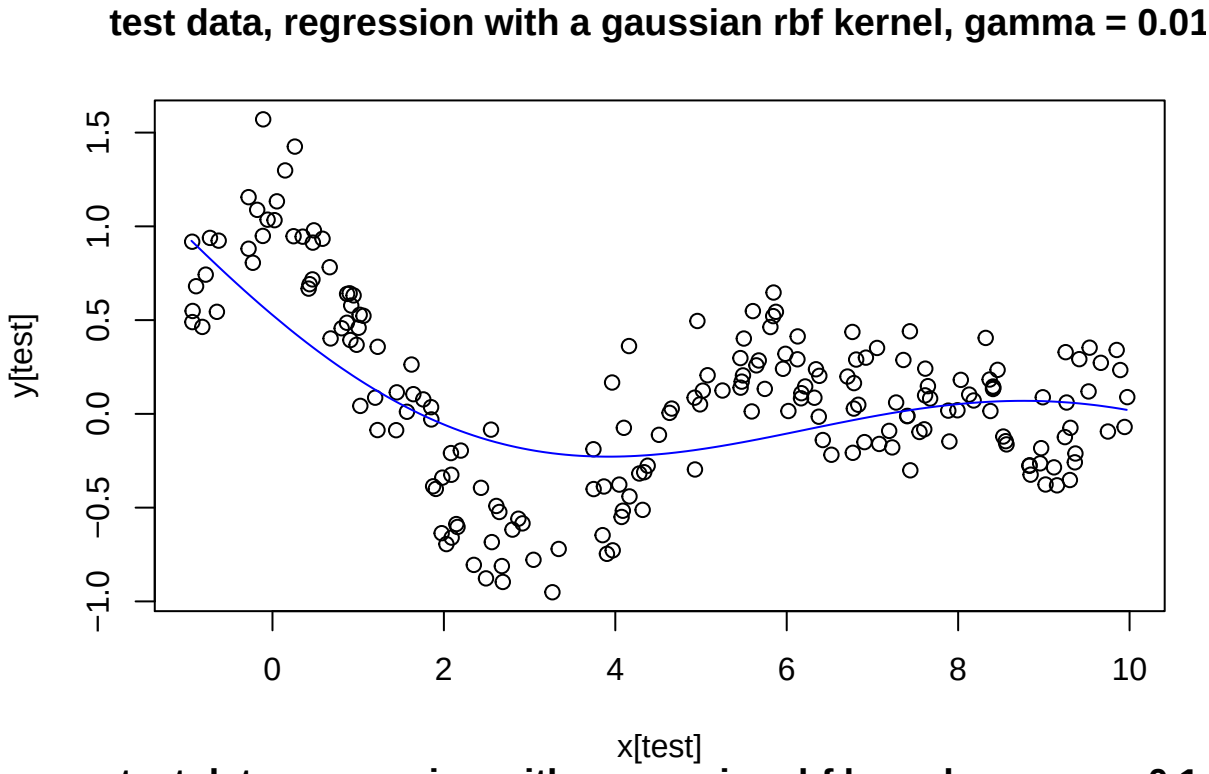

```
poly.mse = data.frame(regression = polynames,
                      train.mse = train.mse,
                      test.mse = test.mse)
print(poly.mse)
```

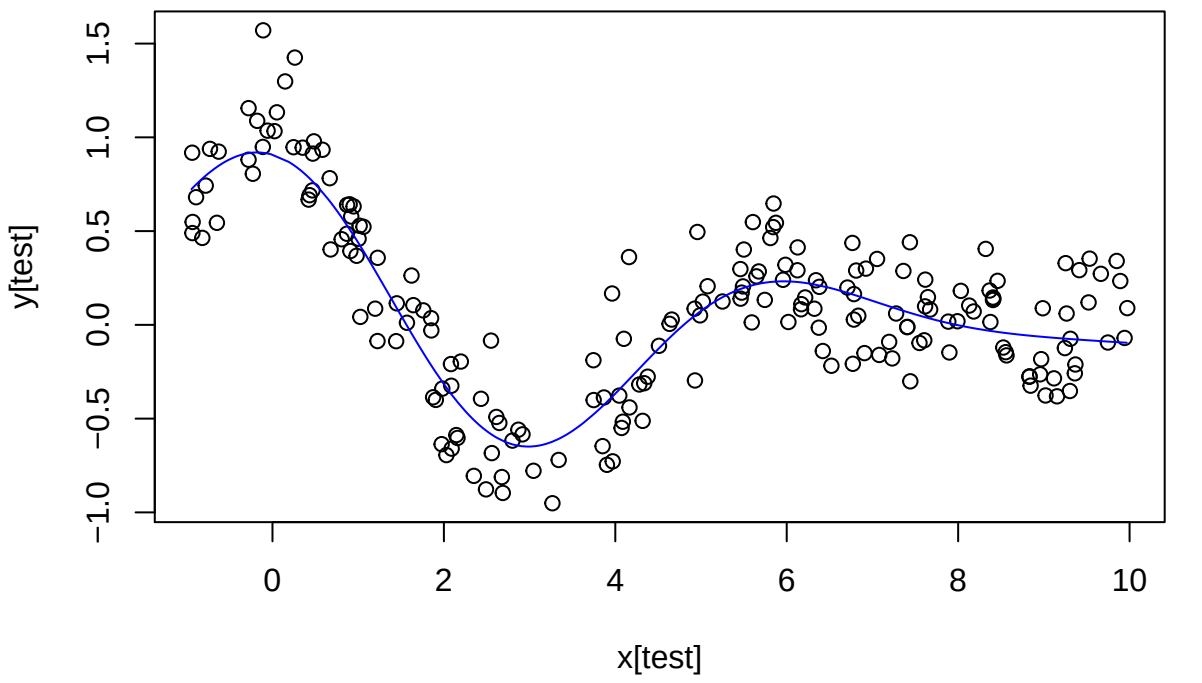
```
##   regression  train.mse   test.mse
## 1  quadratic 0.17054138 0.16248528
## 2      cubic 0.11147648 0.12751214
## 3    quartic 0.10792610 0.12260511
## 4     sextic 0.04890807 0.05089887
```

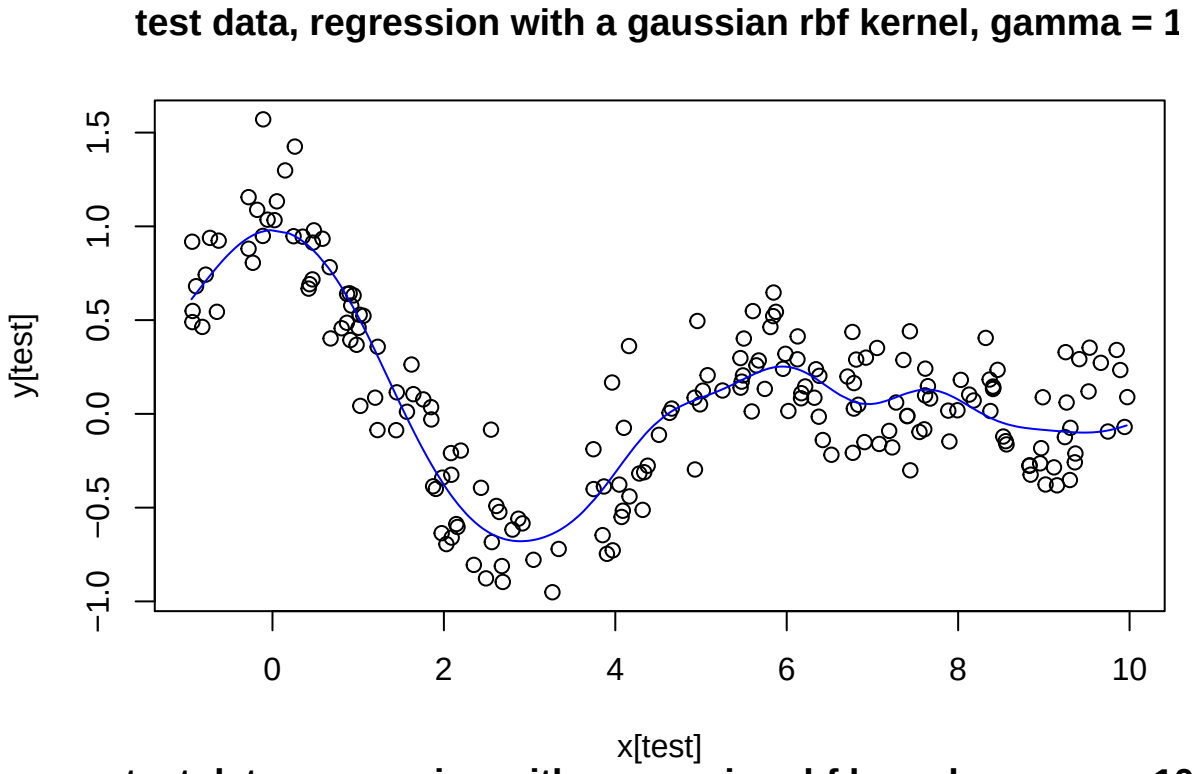
```
gamma = 10^(-4:4)
train.mse = c()
test.mse = c()
for (i in 1:length(gamma)) {
  reg = kreg(gaussk(X, X, gamma[i]), y[train], grid)
  plot(x[test], y[test], main=paste("test data, regression with a gaussian rbf kernel, gamma =",gamma[i]
  lines(x[train], reg$fit[,match(reg$lambda.min, grid)], col="blue")
  train.mse[i] = min(reg$mse)
  test.mse[i] = mean((y[test] - gaussk(x[test], X, gamma[i]) %*% reg$alphas[,match(reg$lambda.min, grid
}
```

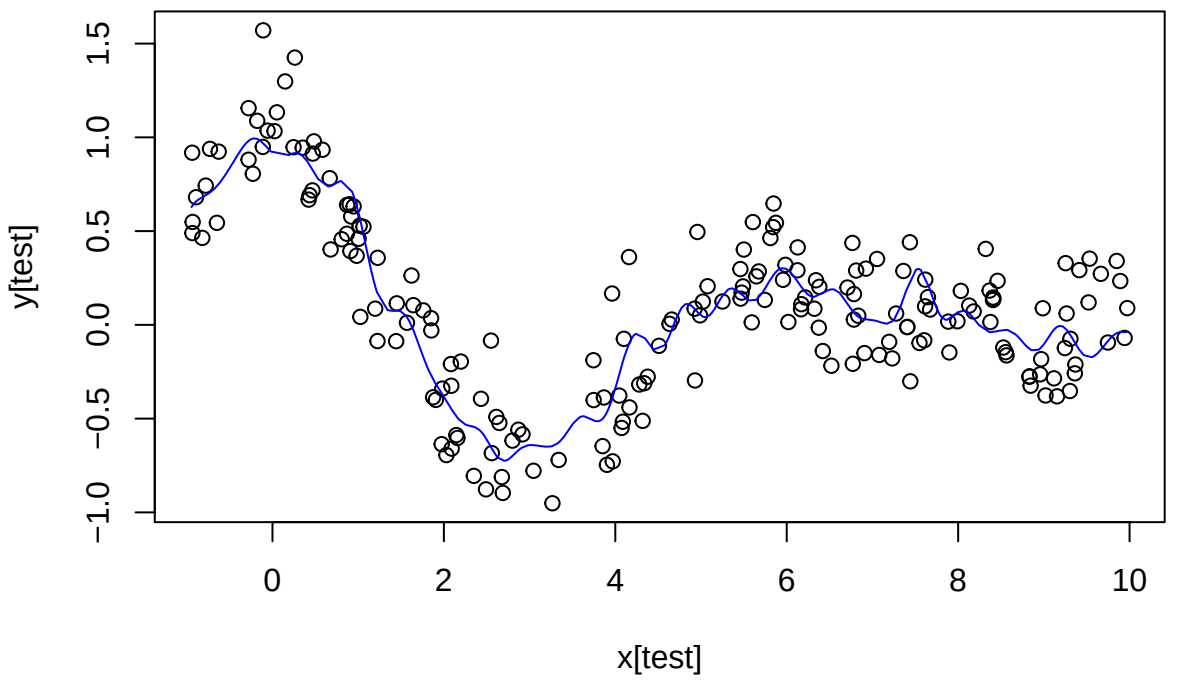

test data, regression with a gaussian rbf kernel, gamma = 1e−04
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10

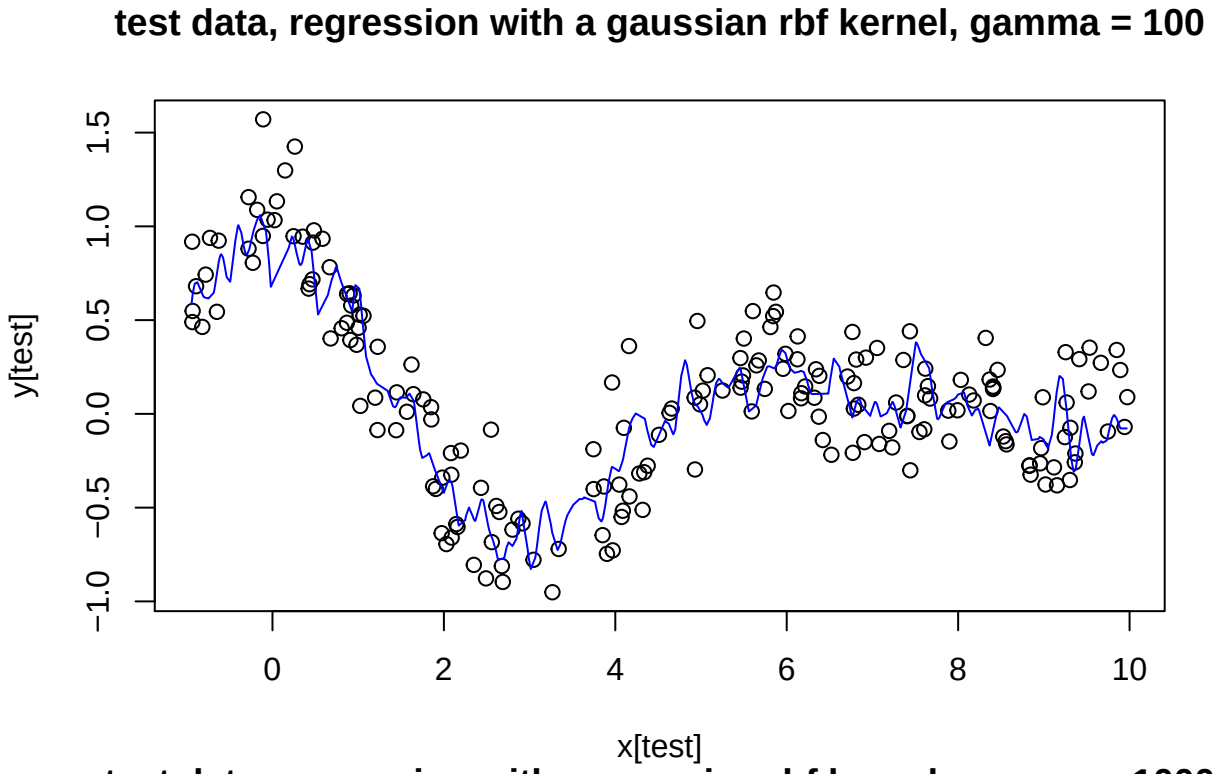

test data, regression with a gaussian rbf kernel, gamma = 0.001
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10

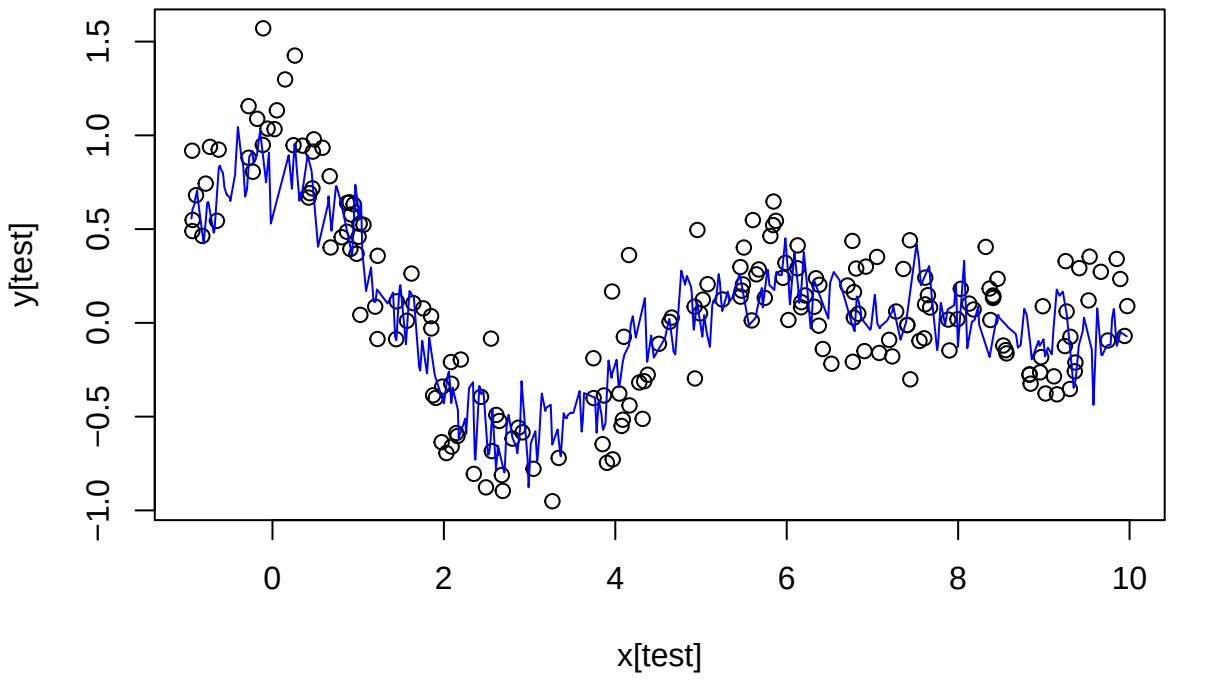

test data, regression with a gaussian rbf kernel, gamma = 0.01
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10

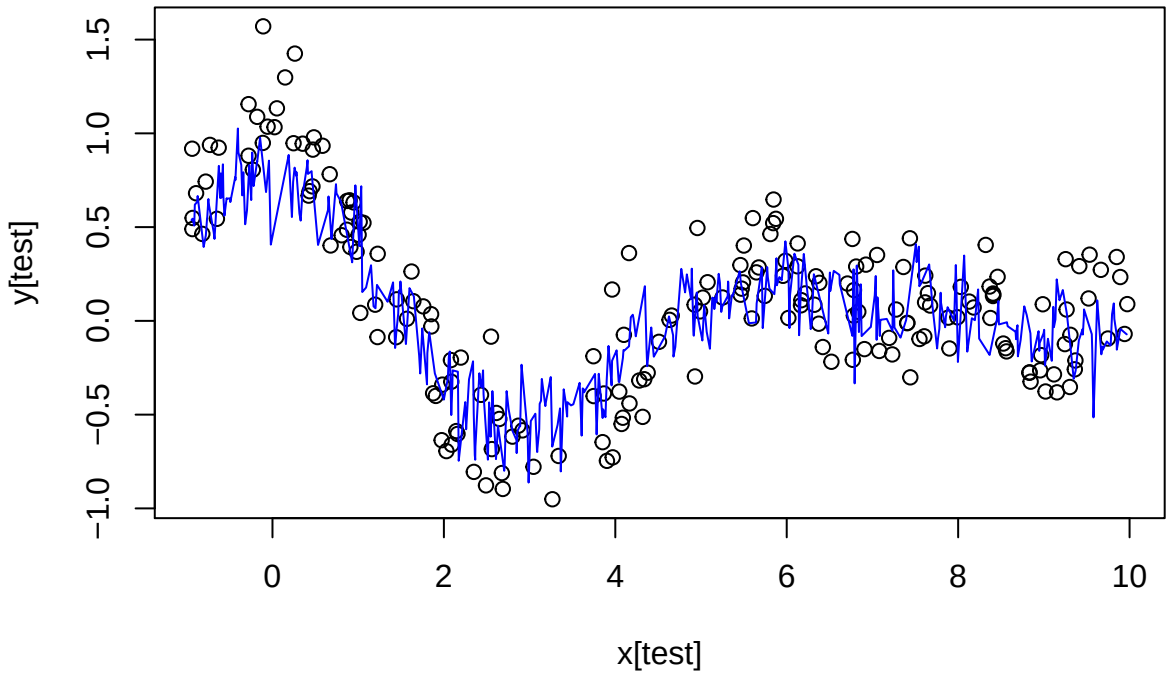

test data, regression with a gaussian rbf kernel, gamma = 0.1
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10


test data, regression with a gaussian rbf kernel, gamma = 1
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10


test data, regression with a gaussian rbf kernel, gamma = 10
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10


test data, regression with a gaussian rbf kernel, gamma = 100
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10


test data, regression with a gaussian rbf kernel, gamma = 1000
y[test]
x[test]
−1.0
−0.5
0.0
0.5
1.0
1.5
0
2
4
6
8
10




```
rbf.mse = data.frame(gamma = gamma,
                     train.mse = train.mse,
                     test.mse = test.mse)
print(rbf.mse)
```

```
##   gamma  train.mse   test.mse
## 1 1e-04 0.23413852 0.21242345
## 2 1e-03 0.21435734 0.19638939
## 3 1e-02 0.12628038 0.12842429
## 4 1e-01 0.04385791 0.04879949
## 5 1e+00 0.04046446 0.04733622
## 6 1e+01 0.03672060 0.05176061
## 7 1e+02 0.02919868 0.06323308
## 8 1e+03 0.02051877 0.09590642
## 9 1e+04 0.01752274 0.16292500
```