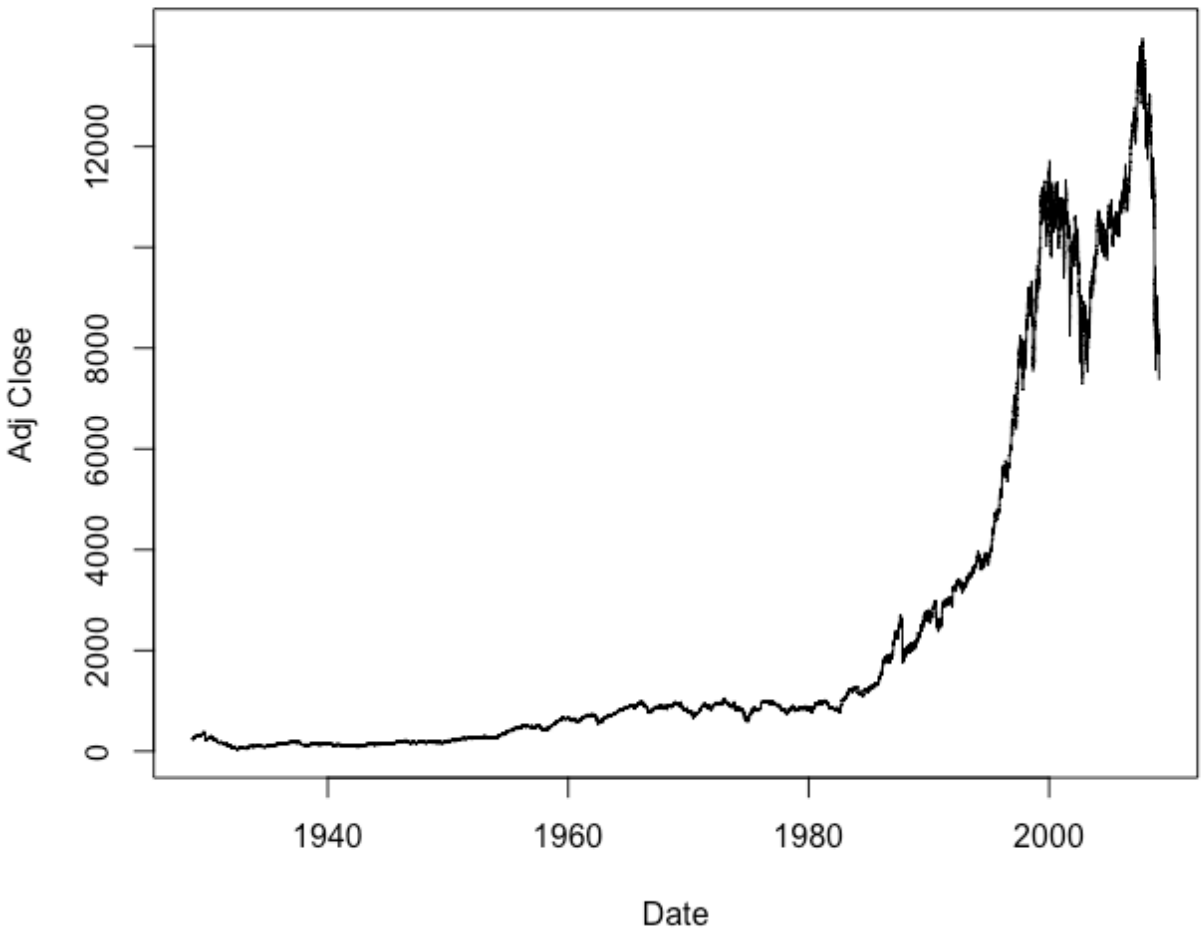
HW 8: Spline

Problem 1.

```
# ------------------------------------------------------------
# Problem 1.
# plot the time series of the adjusted closing prices.

dataPath = "/Users/woodie/Documents/Courses/ISyE 6416 Computational
Statistics (Spring 2018)/HW/ISYE-6416/hw8-spline/DJI_2009.csv"
rawdata  = read.csv(dataPath, header = TRUE)
# preprocess rawdata (including convert first field to date)
df       = rawdata
df$Date  = as.Date(df$Date, "%m/%d/%Y")
df[order(df$Date),]
# plot rawdata
plot(df$Date, df$AdjClose, xlab="Date", ylab="Adj Close", type="l")
```

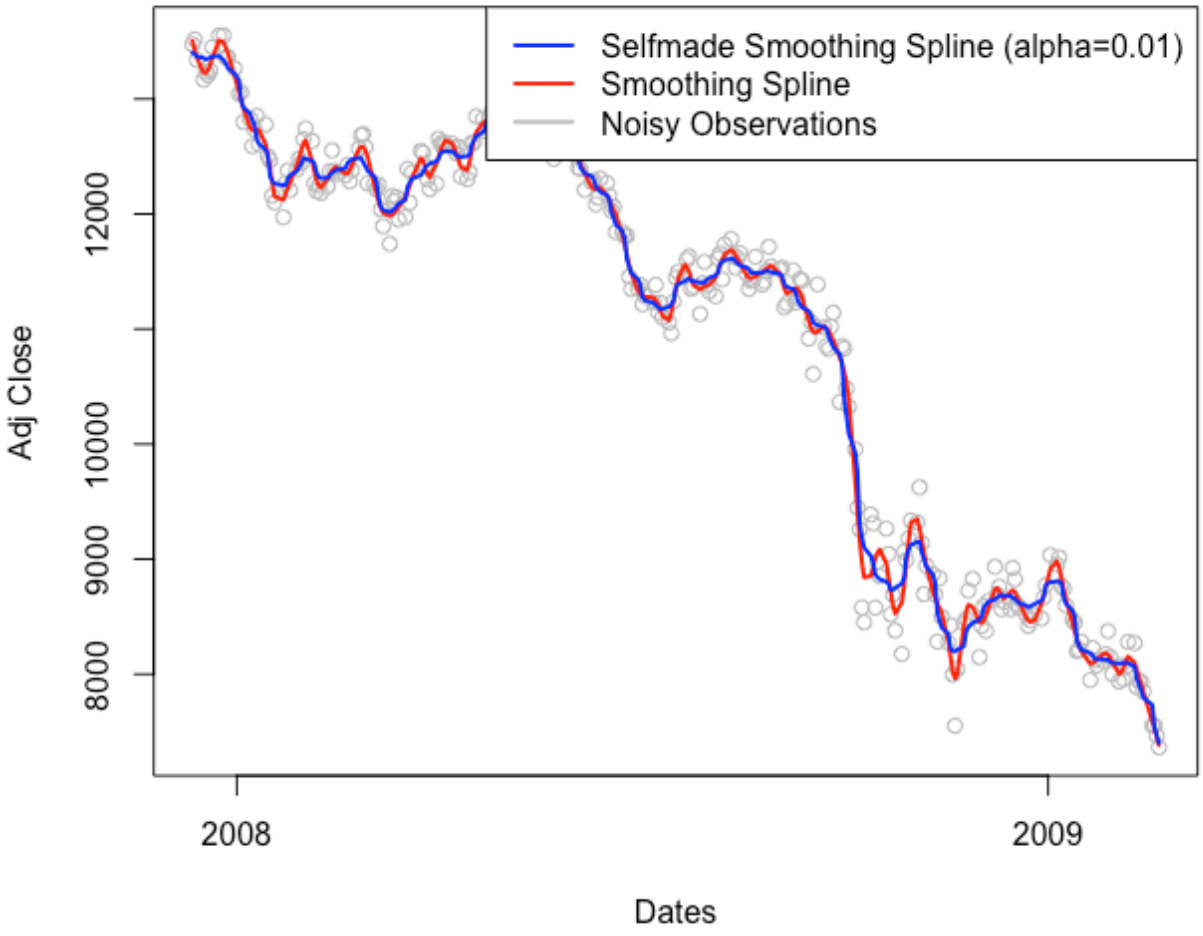


Problem 2.

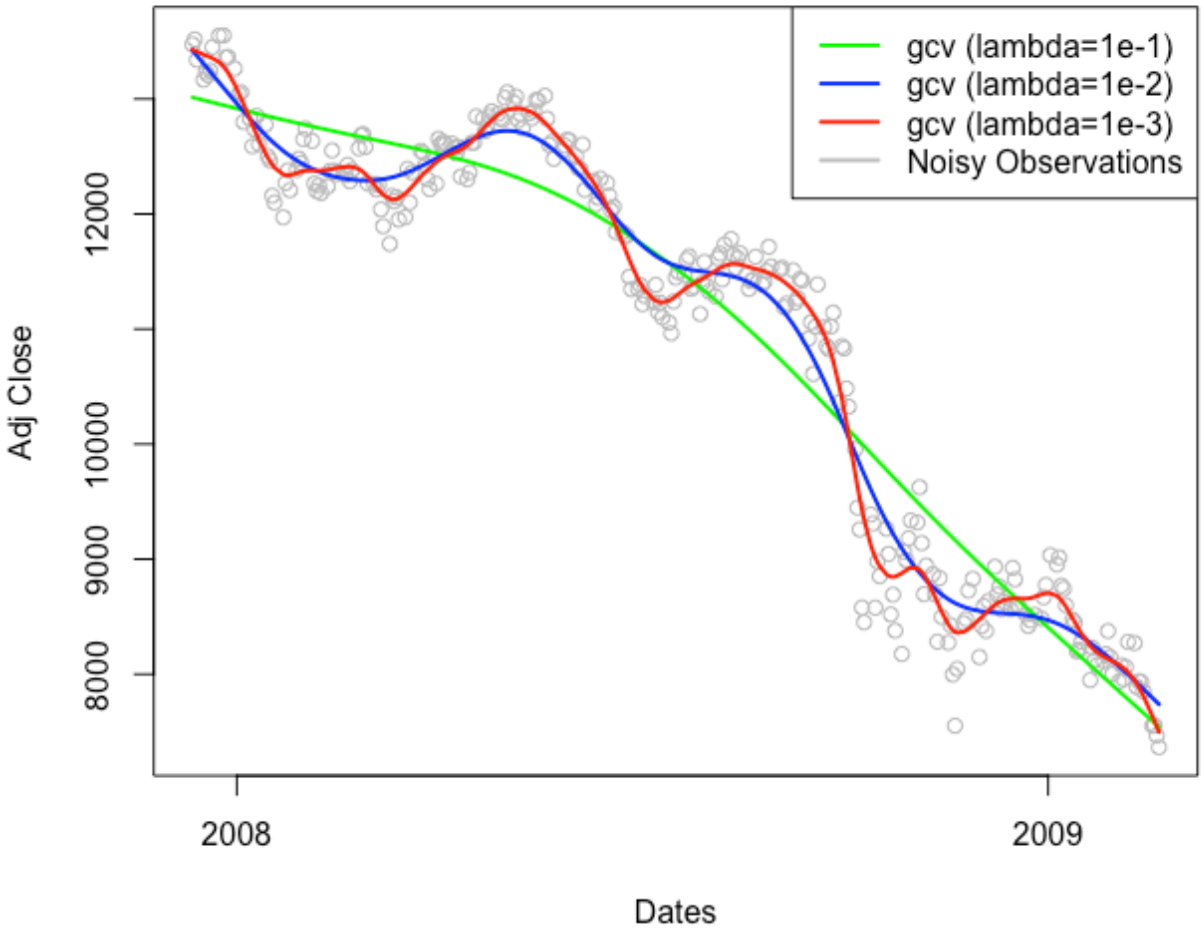
```r
# ------------------------------------------------------------------
# Problem 2.
# Take the value for the last 300 days. Fit a smoothing spline to
# these data points.
latest.df  = head(df, 300)

# call from library
fit.spline = smooth.spline(latest.df$Date, latest.df$AdjClose, cv=TRUE)
# results:
# Smoothing Parameter  spar= 0.207053  lambda= 5.170824e-08 (13 iterations)
# Equivalent Degrees of Freedom (Df): 81.09411
# Penalized Criterion (RSS): 5287446
# GCV: 21167594

# selfmade function
```

```r
my.smooth.spline = function(x, y, alpha, w=NULL){
  # configuaraion and data preparation
  x.order = order(x)
  X = x[x.order]
  Y = y[x.order]
  n = length(X)
  # parameters initialization
  if( is.null(w) ){
    w = rep(1, n)
  }
  W = diag(w)
  h = diff(X)
  Q = matrix(0, n-2, n)
  for(i in 1:(n-2)) {
    Q[i, i+(0:2)] = c(1/h[i], -1/h[i]-1/h[i+1], 1/h[i+1])
  }
  M = diag(h[-(n-1)]+h[-1])/3
  for(i in 1:(n-3)){
    M[i, i+1] = M[i+1, i] = h[i+1]/6
  }
  # solve optimal f
  # note: solve -> get inverse of input matrix (if second param is missing)
  f.hat = solve(alpha*W + (1-alpha)* t(Q) %*% solve(M) %*% Q) %*% W %*% y
*alpha

  return(as.numeric(f.hat))
}
# call from selfmade function
my.fit.spline = my.smooth.spline(as.numeric(latest.df$Date),
latest.df$AdjClose, alpha=.01, w=NULL)

# plotting both noise observation and Smoothing Splines
plot(latest.df$Date, latest.df$AdjClose, col="grey", xlab="Dates",
ylab="Adj Close")
lines(fit.spline, col="red", lwd=2)
lines(latest.df$Date, my.fit.spline, col="blue", lwd=2)
legend("topright",
       c("Selfmade Smoothing Spline (alpha=0.01)",
         "Smoothing Spline","Noisy Observations"),
       col=c("blue", "red","grey"), lwd=2)

# secondly, use the generalized cross validation criterion to determine
# the value of the algorithmic parameter.
fit.gcv.lam1 = smooth.spline(latest.df$Date, latest.df$AdjClose, cv=FALSE,
lambda=1e-1)
```

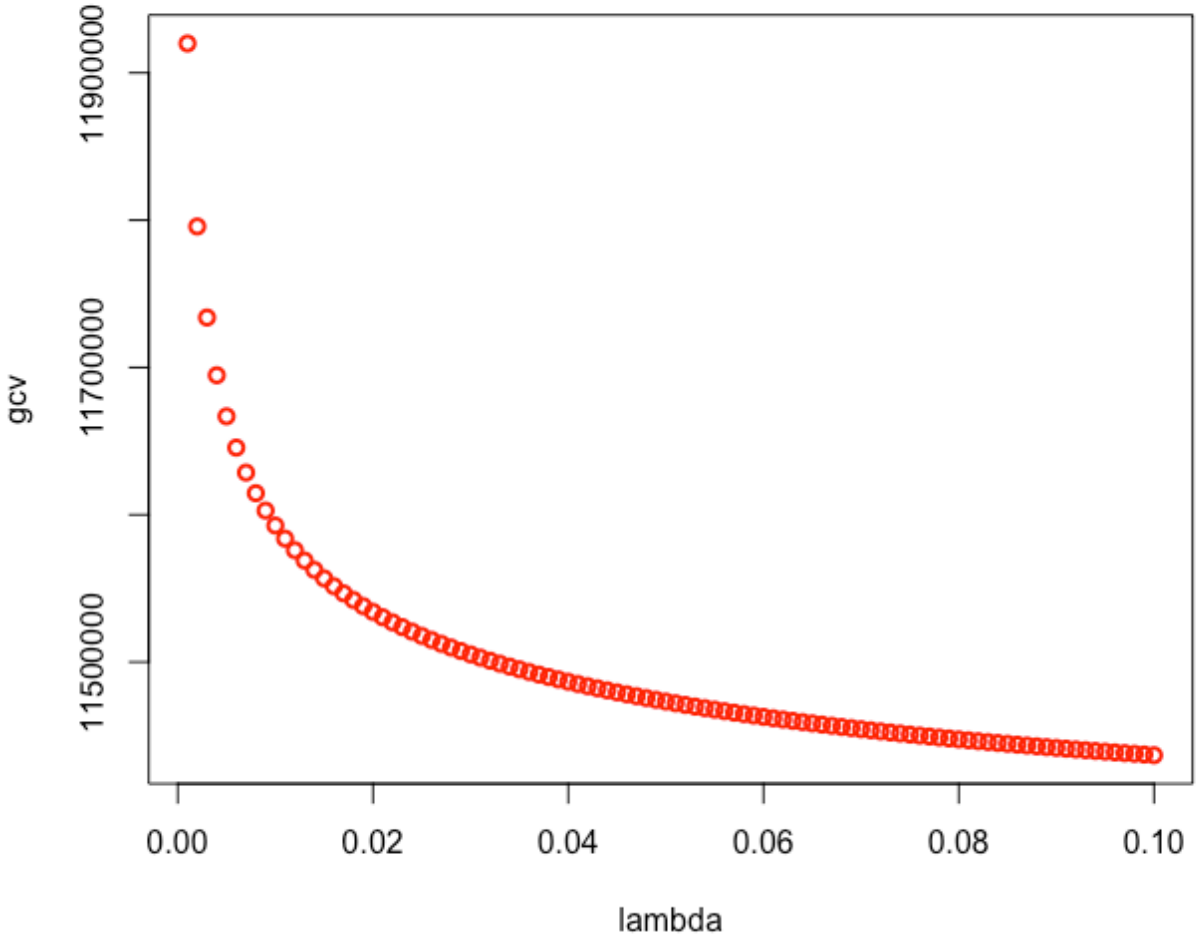
```r
fit.gcv.lam2 = smooth.spline(latest.df$Date, latest.df$AdjClose, cv=FALSE, lambda=1e-3)
fit.gcv.lam3 = smooth.spline(latest.df$Date, latest.df$AdjClose, cv=FALSE, lambda=1e-5)
lambdas  = tail(seq(0, 0.1, 1e-3), 100)
gcv.crit = sapply(lambdas,
                  function(lam) {
                    gcv = smooth.spline(latest.df$Date, latest.df$AdjClose, cv=FALSE, lambda=lam)
                    return(gcv$cv.crit)
                  })
plot(lambdas, gcv.crit, xlab="lambda", ylab="gcv", col="red", lwd=2)

# plotting both noise observation and Smoothing Splines
plot(latest.df$Date, latest.df$AdjClose, col="grey", xlab="Dates", ylab="Adj Close")
lines(fit.gcv.lam1, col="green", lwd=2)
lines(fit.gcv.lam2, col="blue", lwd=2)
lines(fit.gcv.lam3, col="red", lwd=2)
legend("topright",
       c("gcv (lambda=1e-1)", "gcv (lambda=1e-2)", "gcv (lambda=1e-3)", "Noisy Observations"),
       col=c("green", "blue", "red", "grey"), lwd=2)
```



Results of Smoothing Spline



> Results of GCV with different λ

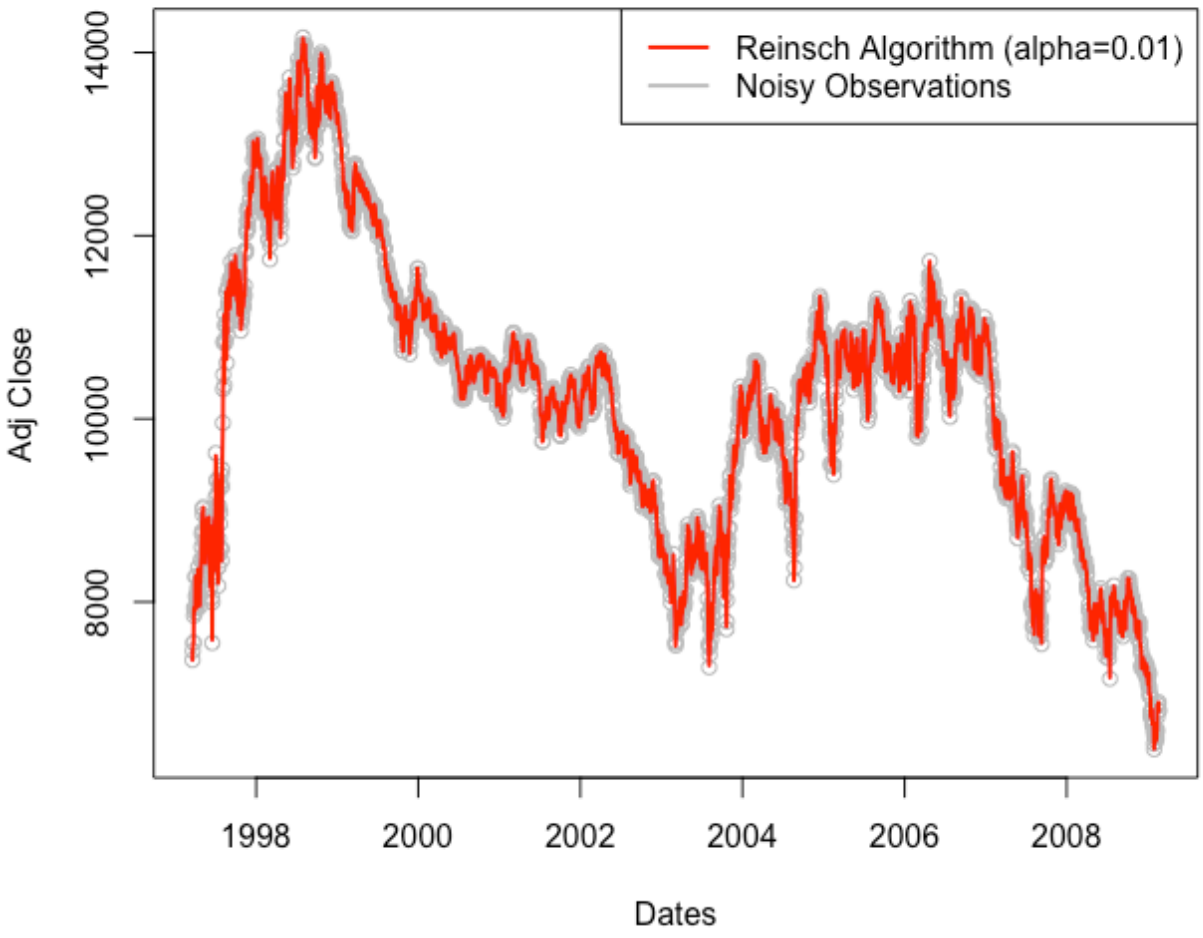


> Results of curve of GCV under different λ

Problem 3.

```r
# ------------------------------------------------------------
# Problem 3.
# Write a more efficient implementation.
first.3k.df = head(df, 3000)
my.reinsch.algo = function (x, y, lambda) {
  # configuaraion and data preparation
  x.order = order(x)
  X = x[x.order]
  Y = y[x.order]
  n = length(X)
  h = diff(X)
  Q = matrix(0, n-2, n)
```

```
  for(i in 1:(n-2)) {
    Q[i, i+(0:2)] = c(1/h[i], -1/h[i]-1/h[i+1], 1/h[i+1])
  }
  M = diag(h[-(n-1)]+h[-1])/3
  for(i in 1:(n-3)){
    M[i, i+1] = M[i+1, i] = h[i+1]/6
  }
  # solve optimal f
  delta.hat = solve(M + lambda * Q %*% t(Q)) %*% Q %*% Y
  f.hat     = Y - lambda * t(Q) %*% delta.hat
  return(f.hat)
}

fit.fast = my.reinsch.algo(as.numeric(first.3k.df$Date),
first.3k.df$AdjClose, lambda=1e-2)

# plotting both noise observation and Smoothing Splines
plot(first.3k.df$Date, first.3k.df$AdjClose, col="grey", xlab="Dates",
ylab="Adj Close")
lines(first.3k.df$Date, fit.fast, col="red", lwd=2)
legend("topright",
       c("Reinsch Algorithm (alpha=0.01)",
         "Noisy Observations"),
       col=c("red", "grey"), lwd=2)
```


Reinsch Algorithm (alpha=0.01)
Noisy Observations
Adj Close
8000
10000
12000
14000
1998
2000
2002
2004
2006
2008
Dates