Predicting the fuel prize index

Assignment 1 – 02417 Time Series Analysis – Anders Hørsted (s082382)

In this report predictions for the fuel prize index (FPI) in the first half of 2005 will be made. The predictions are based on a dataset containing the observed FPI for every month from January 1979 to December 2004. The uncertainty of the predictions will also be calculated.

The data

To get an overview of the data the fuel price index is plotted as a function of time and the resulting plot is shown in figure 1. From the plot it is seen that the FPI grew rapidly from 1979 to 1985 and with small variation between months (the points almost follows a straight line). After a period with decrease between 1985-1987 a long period (1987-1997) with slow increase follows. The variation between months is larger than for the 1979-1985 period. From 2000 to mid-2001 the FPI increases rapidly followed by a short decrease from mid-2001 to start-2002. From start-2002 to the end of 2004 the FPI once more grows fast and the variation between months is larger than for any of the previous periods.

From the description of the data it should be clear that it won't make much sense to estimate a global mean and standard deviation from the data as the FPI don't seem to vary around a mean value but rather to steadily increase, only interrupted by short periods of decrease. We therefore have to rule out the constant mean model[*] and will instead start out by fitting a global linear model to the data.

General linear model

As an first attempt to make predictions it is assumed that the FPI can be modeled as a first order polynomial of time with some white noise added. The white noise represents the various random phenomenom that influences the FPI. The model is

$$Y_t = \theta_0 + \theta_1 t + \varepsilon_t$$

[*]See equation 3.63 in [1]

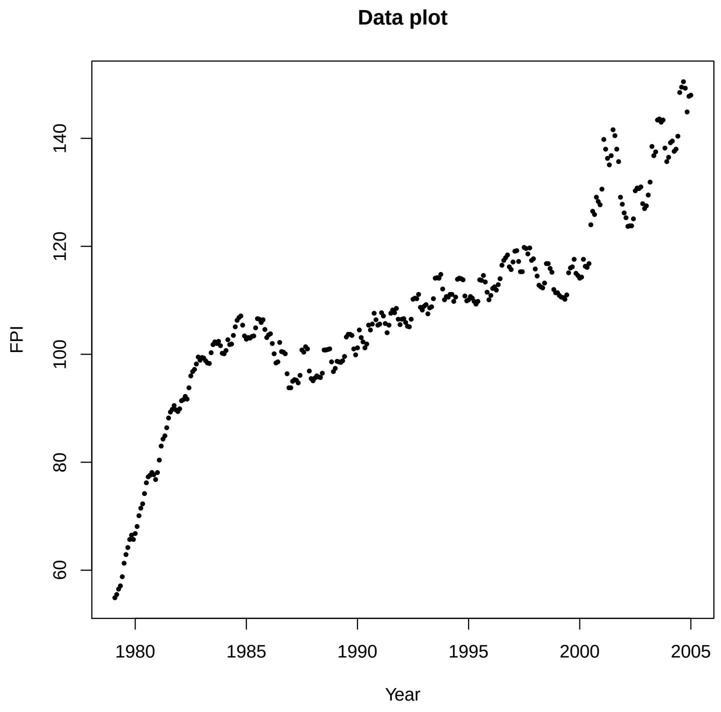

Figure 1: Plot of fuel price index as function of time

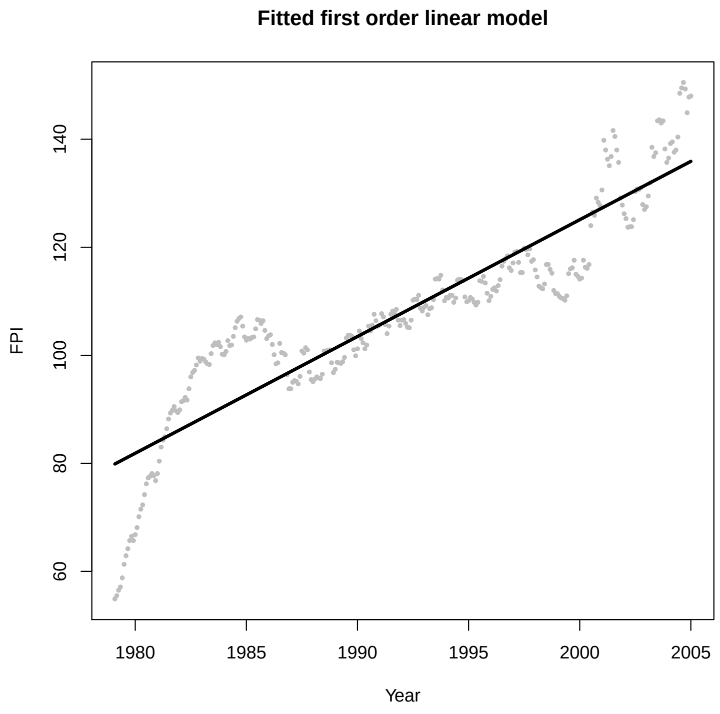

Figure 2: Plot of simple linear model

where ε_t is assumed to be i.i.d random variables with $\mathrm{E}[\varepsilon_t] = 0$ and $\mathrm{Var}[\varepsilon_t] = \sigma^2$. Using R the model parameters are estimated from the data and gives

$$Y_t = -4198 + 2.162t + \varepsilon_t$$

A plot of the model is shown in figure 2. From the plot it is seen that the model do not fit the data very well. For the early data points the model predictions are too high and for the data points near 2005 the model seems to predict lower than the observed FPI. The poor fit of the model can be further illustrated by plotting the residuals ε_t. If the model fitted the data well the residuals would be pure white noise and no structure would be seen in the residuals plot. But as seen in figure 3 a lot of structure remains in the residuals and the conclusion is that the simple linear model is not appropriate for making predictions.

Local constant mean model

One of the problems with the simple linear regression used in the previous section is that it tries to fit all the data. This means that the measured FPI in 1979 influences the predictions for 2005 as much as the measured FPI in December 2004. To counteract this we now use a local constant mean model. In this model the mean of FPI is allowed

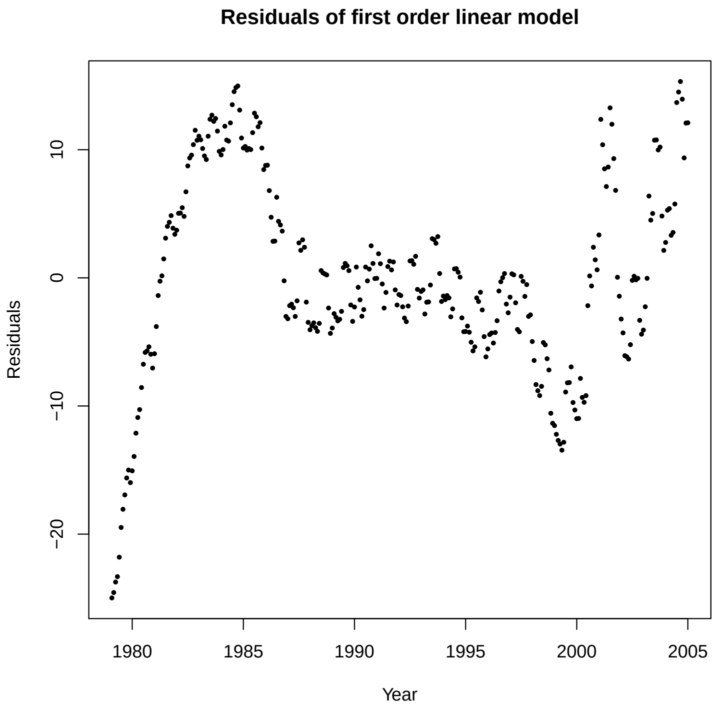

Figure 3: Plot of residuals for the simple linear model

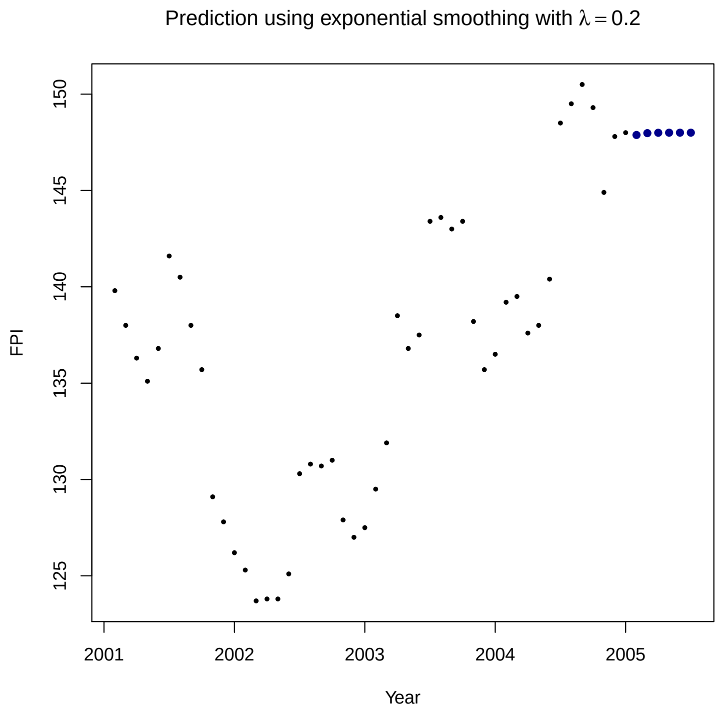

Figure 4: Plot of simple linear model

to change by giving more weight to the most recent observations (see [1, p. 50]). In theory all previous observations still influences predictions, but in practice only the few most recent observations are important. How many observations that influences the prediction can be controlled by changing the forgetting factor λ.

To predict the FPI for the first half of 2005 a local constant mean model was created for $\lambda = 0.2$ and $\lambda = 0.8$. The results are shown in figure 4 and figure 5. For $\lambda = 0.2$ all predictions are almost identical and almost equal to the newest observation of FPI. For $\lambda = 0.8$ the predictions start out a bit lower but converges toward the value of the newest observation[†].

[†]Actually I don't quite get this result. In equation 3.70 in the course textbook, it is stated that all future prediction, given a data set, should be equal. Using equation 3.75 I get predictions that are not exactly equal. Am I doing something wrong?

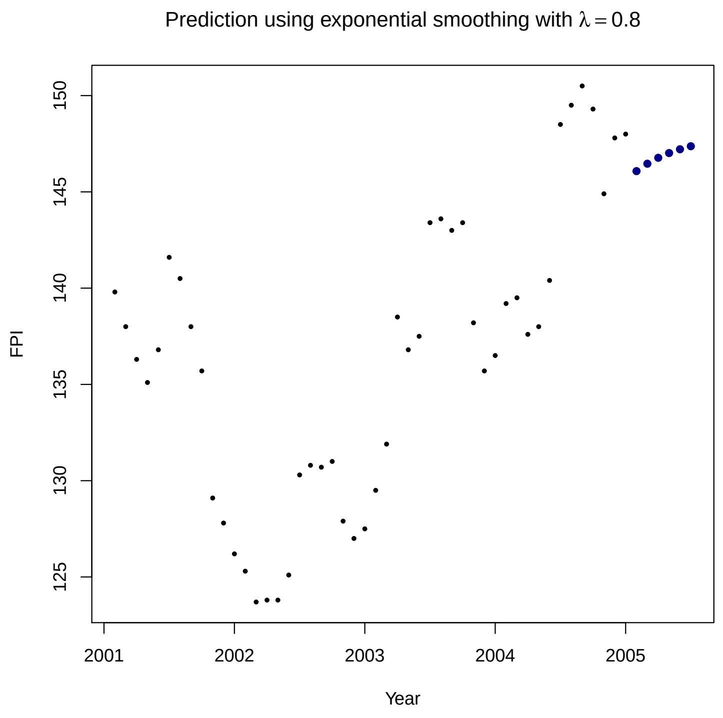

Figure 5: Plot of simple linear model

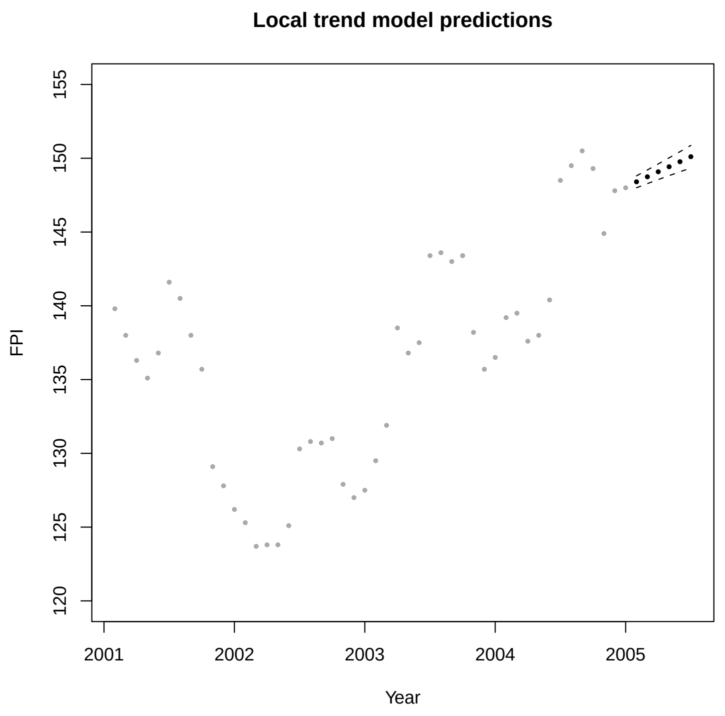

Figure 6: Plot of simple linear model

Local trend model

In the previous section the predictions was improved, compared to the simple linear model, by giving more weight to the most recent observation. But the model was simple and the predictions were almost equal which doesn't seem plausible. In this section a model that is linear in time and that gives most weight to the most recent observations is utilized. The model is called a local trend model and is just a special kind of WLS model where the weights are chosen as powers of the forgetting factor λ. Predictions and confidence intervals for the predictions is calculated (with $\lambda = 0.6$) and the result is shown in figure 6. It can be seen that the prediction for January 2005 is again very close to the value of December 2004, but unlike the previous model, predictions for the other months of 2005 are now linearly increasing.

Calculating the confidence intervals

The confidence intervals for the predictions seems very optimistic but this could be due to the fact that the covariance matrix $\mathbf{\Sigma}$ is very ill-conditioned. From a computational point of view the covariance matrix is singular which means that $\mathbf{\Sigma^{-1}}$ can't be calculated directly and therefore equation 3.44 in [1] can't be used to calculate $\widehat{\sigma}^2$. Instead $\widehat{\sigma}^2$ was

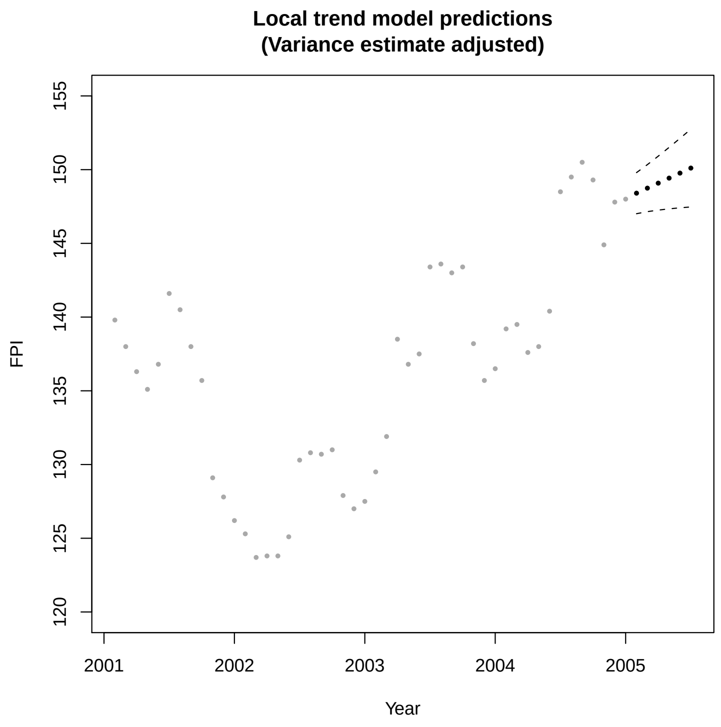

Figure 7: Plot of simple linear model

calculated by (using the notation from [1])

$$S(\widehat{\boldsymbol{\theta}}) = \sum_{j=0}^{N-1} \lambda^j (Y_{N-j} - \boldsymbol{f}^T(-j)\widehat{\boldsymbol{\theta}}_N)$$

$$\widehat{\sigma}^2 = \frac{S(\widehat{\boldsymbol{\theta}})}{N-p}$$

The problem with this approach is that for most values of j, λ^j is smaller than machine precision and therefore only a small number of the N observations are included in the calculation. But from the formula above it is seen that we still divide by $N-p$ so $\widehat{\sigma}^2$ becomes smaller than it should be. A pragmatic solution is to count for how many j's λ^j is above some threshold (e.g. 10^{-6}) and then use this number instead of N. This was implemented and a new set of confidence intervals was calculated. The result is shown in figure 7 and now the confidence interval seems much more plausible.

Selecting λ

The model seems to be useful but in the previous subsection λ was randomly chosen as 0.6. It now needs to be determined what value of λ fits the data best. Therefore the

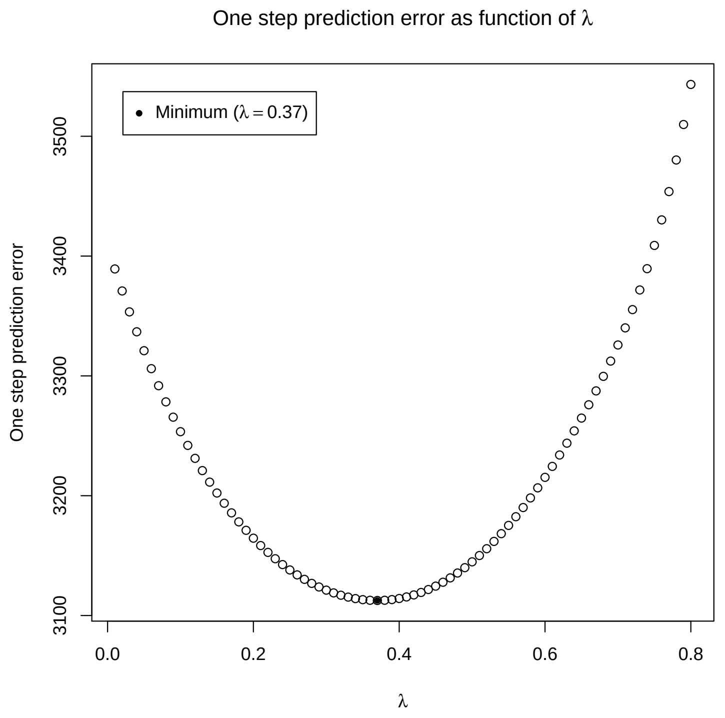

Figure 8: Plot of simple linear model

one-step prediction errors (see eq. 3.78 and 3.79 in [1]) are calculated for various lambda values and the result is shown in figure 8.

Final model

The local trend model is now estimated for $\lambda = 0.37$ and predictions and confidence intervals are calculated. The result is shown in figure **??**

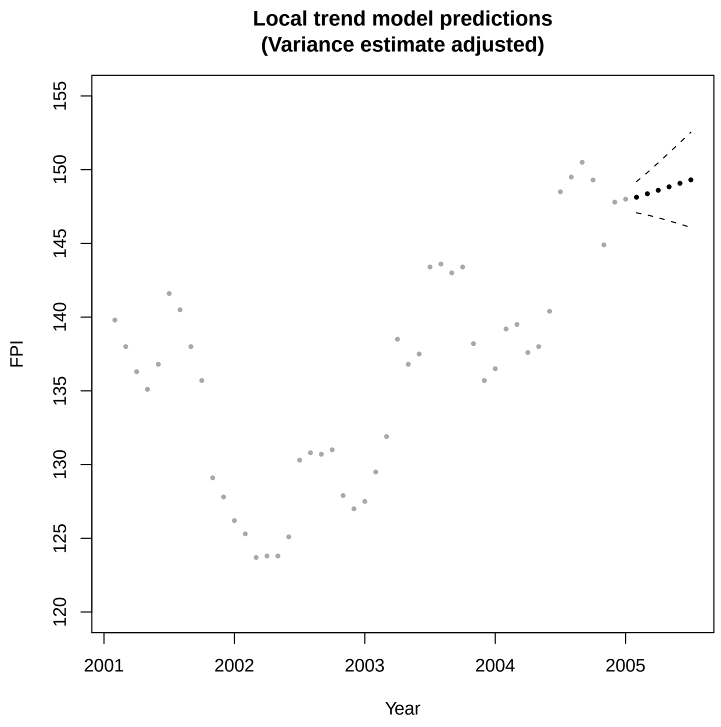

Figure 9: Plot of simple linear model

Appendices

All R code used for this assignment are included here. All source code incl. latex code for this report can be found at `http://github.com/alphabits/dtu-fall-2011/`

Loading data (loaddata.R)

```
d <- read.csv('fuel.csv')
rtime <- d[,3]
fpi <- d[,4]
N <- length(fpi)
pred.time <- seq(2005+1/12, 2005.5, length.out=6)

cex <- 0.7
col.normal <- "darkgray"
cex.pred <- 0.7
col.pred <- "black"
pch <- 20
```

Data plot (ex-1.R)

```
source('loaddata.R')

pdf('data-plot.pdf')
plot(rtime, fpi, xlab='Year', ylab='FPI', cex=cex, pch=pch,
     main="Data plot")
dev.off()
```

Simple linear model (ex-3.R)

```
# Loads data in variable "data"
source('loaddata.R')


model <- lm(fpi ~ rtime)
pdf('fitted-linear-model.pdf')
plot(rtime, fpi, pch=pch, cex=cex, xlab="Year", col="gray",
     ylab="FPI", main="First order linear model")
lines(rtime, fitted(model), lwd=3)
dev.off()
model.residuals <- resid(model)
pdf('fitted-linear-model-residuals.pdf')
plot(rtime, model.residuals, pch=pch, cex=cex,
     xlab="Year",
     ylab="Residuals",
     main="Residuals of first order linear model")
dev.off()

sink('fitted-linear-model-summary.txt')
summary(model)
sink()
```

Exponential smoothing (ex-4.R)

```
# Load data in variables rtime and fpi
source('loaddata.R')

lambda <- 0.7
N <- length(fpi)
const <- (1-lambda)
last.measure <- fpi[N]

ys <- rep(0, times=N)
ys[1] <- fpi[1]
for (i in 2:N) {
    ys[i] <- const*fpi[i] + lambda*ys[i-1]
}

num.predictions <- 6
preds <- rep(0,times=num.predictions)
preds[1] <- ys[N]
for (i in 2:num.predictions) {
    preds[i] <- const*last.measure + lambda*preds[i-1]
}

ind <- rtime > 2001
pdf(sprintf('exp-smoothing-lambda-%s.pdf', strsplit(as.character(lambda), "\\.")[[1]][2]))
plot(c(rtime[ind], pred.time), c(fpi[ind], preds), pch=pch, cex=cex,
     main=expression(paste("Prediction using exponential smoothing with ", lambda==0.7)),
     xlab="Year",
     ylab="FPI")
points(pred.time, preds, pch=pch, cex=cex.pred, col=col.pred)
dev.off()
```

Local trend model (ex-5.R)

```
source('loaddata.R')
source('local-trend-functions.R')

lambda <- 0.6
lambda.filename <- strsplit(as.character(lambda), "\\.")[[1]][2]

FN <- calc.FN(fpi, lambda)
hN <- calc.hN(fpi, lambda)
thetahat <- calc.thetahat(FN, hN)
preds <- cbind(rep(1, times=6), 1:6)%*%thetahat
ind <- rtime > 2001

for (type in c("normal", "adjusted")) {
    adjust <- (type=="adjusted")
    extra.plot.title <- ifelse(adjust, "\n(Variance estimate adjusted)", "")
    sigmahat <- calc.sigmahat(fpi, lambda, thetahat, adjust.df=adjust)
    variance <- sapply(1:6, calc.prediction.variance, sigmahat=sigmahat,
                       FN.inv=solve(FN))
    conf.interval <- qt(0.95, N-2)*sqrt(variance)
    pdf(sprintf("local-trend-predict-%s-%s.pdf", type, lambda.filename))
    plot(c(rtime[ind], pred.time), c(fpi[ind], preds), pch=pch, cex=cex,
         col=col.normal, ylim=c(120,155), ylab="FPI", xlab="Year",
         main=sprintf("Local trend model predictions%s", extra.plot.title))
    points(pred.time, preds, pch=pch, cex=cex.pred, col=col.pred)
    lines(pred.time, preds+conf.interval, lty=2)
    lines(pred.time, preds-conf.interval, lty=2)
    dev.off()
}
```

Determine optimal λ (ex-6.R)

```
source('loaddata.R')
source('local-trend-functions.R')

lambdas <- seq(0.01, 0.8, 0.01)
res <- sapply(lambdas, one.step.predict.sum, y=fpi)
pdf('prediction-error.pdf')
plot(lambdas, res, xlab=expression(lambda), ylab="One step prediction error",
     main=expression(paste("One step prediction error as function of ", lambda)))
min.index <- which.min(res)
points(lambdas[min.index], res[min.index], pch=20)
legend("topleft", inset=.05, pch=20, expression(paste("Minimum (", lambda==0.37, ")")))
dev.off()
```

Functions for local trend model (local-trend-functions.R)

```
L.inv <- matrix(c(1,-1,0,1), nrow=2)

f <- function (j) {
    return(matrix(c(1, j), nrow=2))
}

calc.FN <- function (y, lambda) {
    N <- length(y)
    FN <- matrix(rep(0, times=4), nrow=2)
    for (j in (N-1):0) {
        FN  <- FN + lambda^j*f(-j)%*%t(f(-j))
    }
    return(FN)
}

calc.hN <- function (y, lambda) {
    N <- length(y)
    hN <- matrix(rep(0, times=2), nrow=2)
    for (j in (N-1):0) {
        hN <- hN + lambda^j*f(-j)*y[N-j]
    }
    return(hN)
}

calc.thetahat <- function (FN, hN) {
    return(solve(FN)%*%hN)
}

calc.sigmahat <- function (y, lambda, thetahat, adjust.df=FALSE) {
    N <- length(y)
    adjusted.N <- sum(as.numeric(lambda^(0:(N-1)) > 1e-6))
    deg.freedoms <- ifelse(adjust.df, adjusted.N, N) - 2
    calc.sigmahat.j <- function (j) {
        return(lambda^j * (y[N-j] - t(f(-j))%*%thetahat)^2)
    }
    return(sum(sapply(0:(N-1), calc.sigmahat.j))/deg.freedoms)
}

calc.prediction.variance <- function (l, sigmahat, FN.inv) {
    sigmahat*(1 + t(f(l))%*%FN.inv%*%f(l))
}

one.step.predict.sum <- function (lambda, y) {
    FN <- calc.FN(y[1:2], lambda)
    hN <- calc.hN(y[1:2], lambda)
    errs <- 0
    for (j in 2:(N-1)) {
        thetahat <- calc.thetahat(FN, hN)
        prediction <- sum(thetahat)
        errs <- errs + (y[j+1] - prediction)^2

        FN <- update.F(FN, j, lambda)
        hN <- update.h(hN, y[j], lambda)
    }
    return(errs)
}

update.F <- function (F.old, N, lambda) {
    return(F.old + lambda^N*f(-N)%*%t(f(-N)))
}

update.h <- function (h.old, y.new, lambda) {
    lambda*L.inv%*%h.old + y.new*f(0)
}
```

References

[1] Henrik Madsen, *Time Series Analysis*. Chapman & Hall/CRC, 1st Edition, 2008.