Assignment 1: Time Series Analysis

March 3, 2017

1 Forecasting Apartment Prices

1.1 Plot the apartment price as a function of time

(See Figure 1)

1.2 Does the global mean value and standard deviation of the prices give a reasonable representation of the data?

The global mean (mean = 9.639204) does not give a reasonable representation of the data since there is a clear upward trend in the data. Predicting with a mean model would likely underestimate future observations.

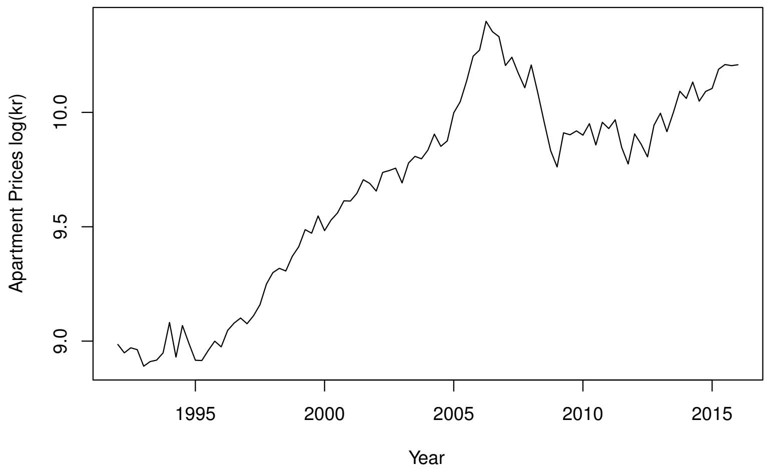

Figure 1: The log prices in DKK/m^2 are shown above as a function of time

1.3 Formulate a GLM model in form of a simple linear regression model for all the data. Estimate the model parameters and plot your fit. Would this model be useful for making predictions of future apartment prices?

A GLM would not be appropriate for predicting future apartment prices. The reason is that the regression assumes that the variance of the errors is constant and that the residuals are mutually independent, which is clearly not the case. Figure 5 shows the QQ plot for the standardized residuals. An alignment along the diagonal line would infer that the residuals are white noise (i.e. normally distributed). However, this is not the case. Figures 3 and 4 further show that the residuals are not normally distributed. As a result, a GLM is not appropriate in this case and the predictions in Figure 2 should not be trusted. We formulate the GLM as shown in the code below. We use the normal equations to solve for our coefficients. Finally, the results are cross-checked with an *lm* function in R. The information below shows the coefficients of the GLM fitted to the log of the data.

```
## 1.3: Formulate GLM: Y = X*theta + error
X <- cbind(rep(1, 92), data$time[1:92])
Xfull <- cbind(rep(1, 97), data$time[1:97])
theta_hat <- solve(t(X)%*%X, t(X)%*%Y)
Y_hat = Xfull%*%theta_hat

## check
lreg <- lm( Y ~ 0 + X )
summary(lreg)
```

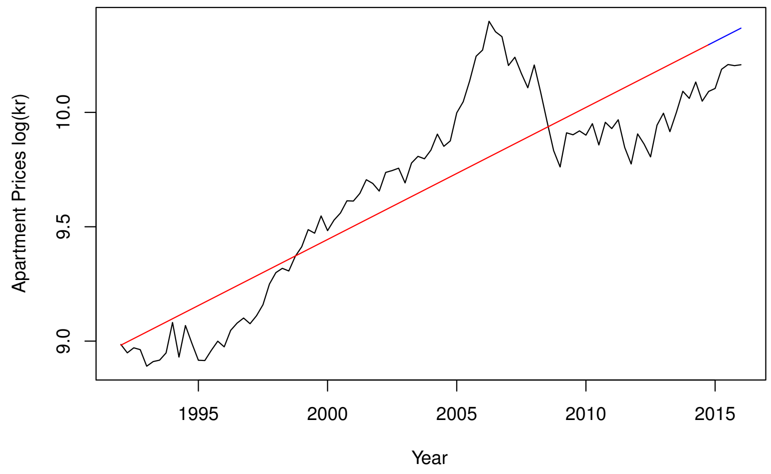

Figure 2: The fitted GLM is shown above in red. The blue extension refers to the predictions

```
Coefficients:
      Estimate Std. Error t value Pr(>|t|)
X1  -1.061e+02  6.808e+00  -15.58   <2e-16 ***
X2   5.775e-02  3.398e-03   16.99   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
```

1.4 Now we will consider methods which considers data more locally. Use simple exponential smoothing with lambda = 0.8 to predict the apartment prices for all of 2015 and 2016Q1. Plot the estimates along the entire series and make a table with the predictions. Comment on the results.

Figure 6 shows the Exponential Smoothing model along with predictions and the 95% confidence intervals. Predictions are listed in Table 1. Note that all predictions are the same; they equal the prediction 1 step ahead from the last observation. This is a feature of the exponential smoothing model. Even though the observations from 2015 onwards fall inside the 95% confidence interval, it is difficult to make reliable long-range predictions using this model. This is simply due to the aforementioned fact that the exponential smoothing model's predictions are constant in the absence of new observations. In essence, the exponential smoothing model is very close to a moving average model except that the weights differ according the value of lambda. High values of lambda will result in the model being less sensitive to recent data, or conversely, the weights across time are more evenly distributed. Therefore, in our case with a lambda of 0.8, our model has somewhat

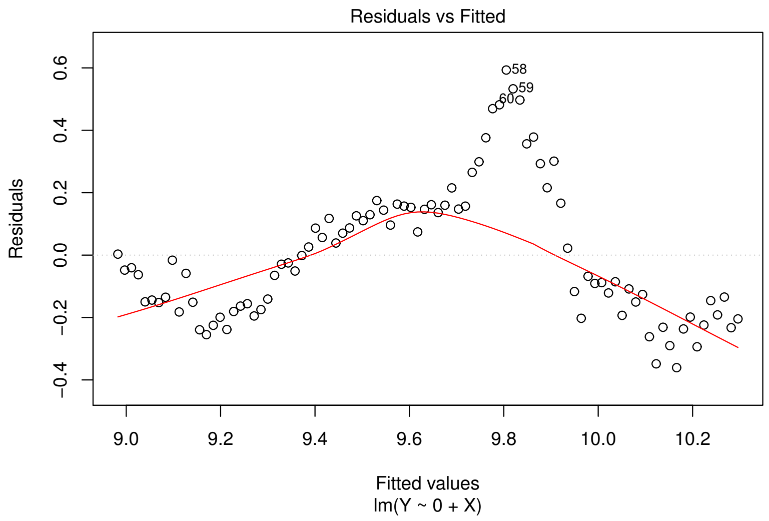

Figure 3: The residuals of the GLM are shown above. The plot shows correlation features in the residuals.

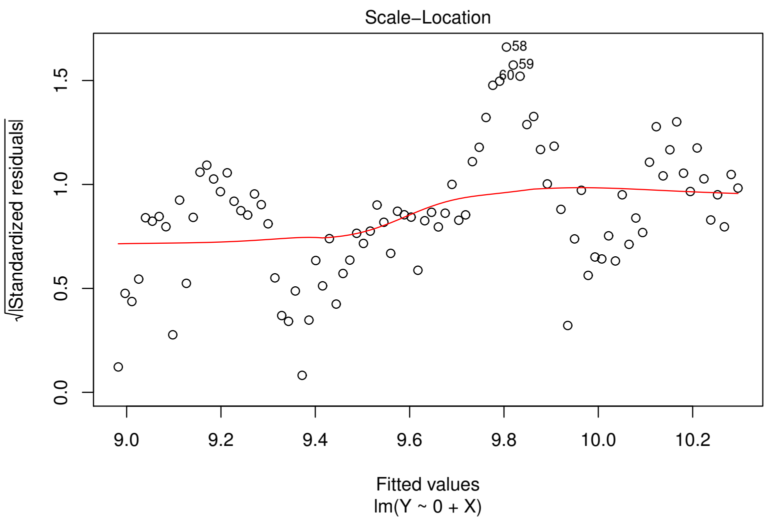

Figure 4: Same as previous figure but with stantardized residuals

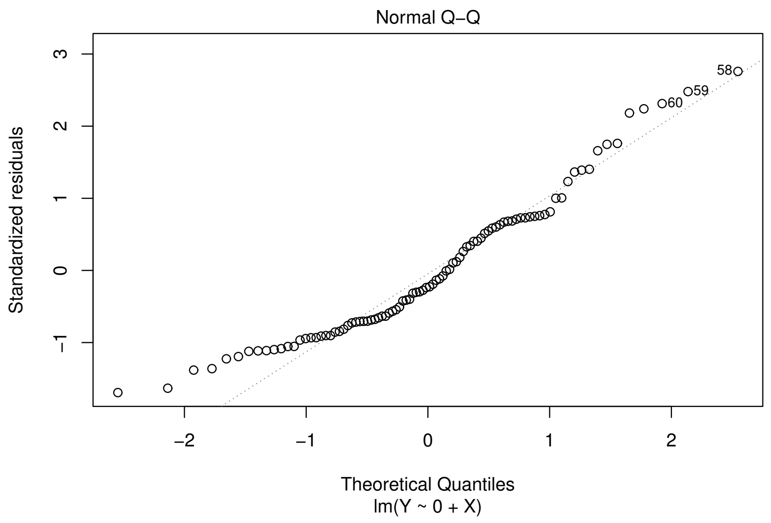

Figure 5: This QQ plot clearly shows that the residuals do not fall aligned on the diagonal, which is how white noise would be shown in the QQ plot. The residuals are not normally distrubuted.

of a ”long-term memory”, and will not be erratic or sensitive to outliers or high peaks. This can be seen in Figure 6, notice how the 1-step predictions (green dotted line) don’t overshoot during the height of the financial housing bubble. For short-term predictions, in this case it would also not be a reliable forecasting model. Notice how there’s a phase-shift in the 1-step predictions (green dotted line), as expected. In an upwards market, the exponential smoothing model is almost guaranteed to underestimate the next prediction since the prior observations have lower values, and therefore drag the prediction below the actual observation value. As can be seen in Figure 6 during the years leading to the Great Recession.

The Exponential Smoothing model was formulated as the code below shows using equations 3.74 and 3.75 from the book.

```
#We initialize the 1st and 2nd forecasts by using Y[1] as starting point (
    S0)
Y_hatExpS[1] <- Y[1]       #We use 1st observation as 1st forecast
Y_hatExpS[2] <- ( 1-lambda ) * ( Y[1] )  +  lambda * Y_hatExpS[1]

#We iterate to find the remaining forecasts using eq. 3.74 from book
for (i in 3:93) {
  Y_hatExpS[i] <-  ( 1-lambda ) * ( Y[i-1] )  +  lambda * Y_hatExpS[i-1]
}

#Predictions: Our last observation is Y[92], so we use eq. 3.75 for
```

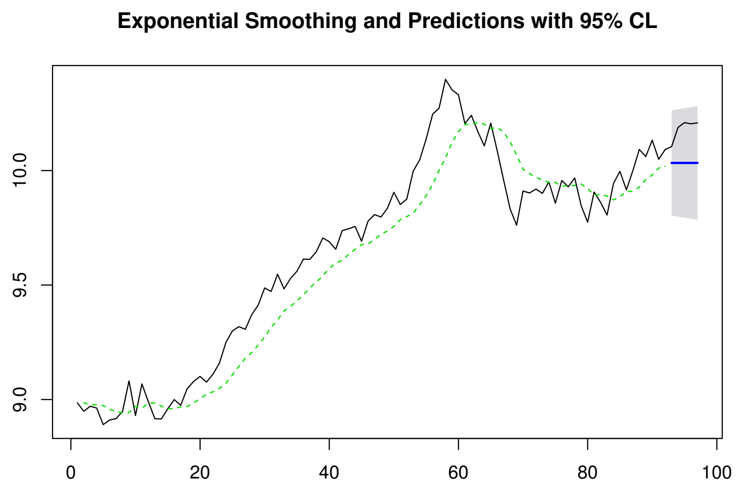

Figure 6: The exponential smoothing 1-step predictions are shown in the green dotted line. The blue line are the predictions from the last observation available. The 95% CL are shown in gray.

```
    predicting 2015 and 2016
for (i in 93:97) {
  Y_hatExpS[i] <-  ( 1-lambda ) * ( Y[92] )  +  lambda * Y_hatExpS[i-1]
}

#Check with Holtwinters function
ExSmooth <- HoltWinters( Yfull[1:92] , alpha = (1 - lambda), beta = F,
    gamma = F, l.start = Yfull[1]  )
forecastsEX <- forecast::forecast.HoltWinters( ExSmooth, h=5, level = .95 )
```

Table1: Predictions of Exponential Smoothing Model with 95% CL

Time	Prediction	Lower	Upper
2015	10.03337	9.803340	10.26340
2015.25	10.03337	9.798784	10.26796
2015.5	10.03337	9.794316	10.27242
2015.75	10.03337	9.789929	10.27681
2016	10.03337	9.785620	10.28112

1.5 Use a local linear trend model to predict the apartment prices for 2015 and 2016Q1. State the uncertainty of the predictions. Plot the estimates along the entire series and make a table with the predictions. Again use l = 0.8. Comment on the results.

Figure 7 shows the 1-step predictions of the Local Linear Trend Model along with the estimates, predictions and the 95% confidence intervals using both the local ($\sigma_{local} = 0.0998727$) and global ($\sigma_{global} = 0.0746517$) sigma estimators as stated in slide 26 of lecture 4. Notice how the observations of 2015 and onwards fall within both of the 95% confidence intervals. We make the predictions using equation 3.101 in the book. In the local linear trend model, the 1-step predictions are basically the sum of the two thetas; the estimates equal theta one.

The variance of prediction error was calculated as in equation 3.102 in the book, and as shown in the following code:

```
### UNCERTAINTY: Global and Local Sigma estimator, Variance of Prediction
    error, and 95% confidence interval:

## Global estimator of sigma2 [as in slide 26 in L#4]
residuals <- as.vector(Y[3:92] - Y_hatLT[3:92])
global.est  <- rep(NA, 90)
for (i in 1:90) {
  global.est[i] <- residuals[i]^2 / ( 1 + fgen(1) %*% solve(F_n(i+2, lambda
      )) %*% t(fgen(1)) )
}
sigma2.global <- (sum(global.est) / (90-2))
sigma.global <- sqrt(sigma2.global) #1626

## Local Estimator of sigma2 [as slide 26 in L#4]
sigma2.local<- ( t(residuals)%*%invSIGMA_n(90, lambda)%*%residuals) /(5-2)
    #/(T-p)
sigma.local <- sqrt(sigma2.local)

## Variance of Prediction Error : [as eq 3.102]
var_PredictionErrorLT.local <- rep(NA, 5)
var_PredictionErrorLT.global <- rep(NA, 5)
for (i in 92:96) {
  var_PredictionErrorLT.local[i-91] <- sigma2.local  * (1 + fgen(i-91) %*%
      solve(F_n(i, lambda)) %*% t(fgen(i-91)))
  var_PredictionErrorLT.global[i-91] <- sigma2.global  * (1 + fgen(i-91) %*
      % solve(F_n(i, lambda)) %*% t(fgen(i-91)))
}
```

Table 2: Predictions of Local Linear Model with 95% CL local est

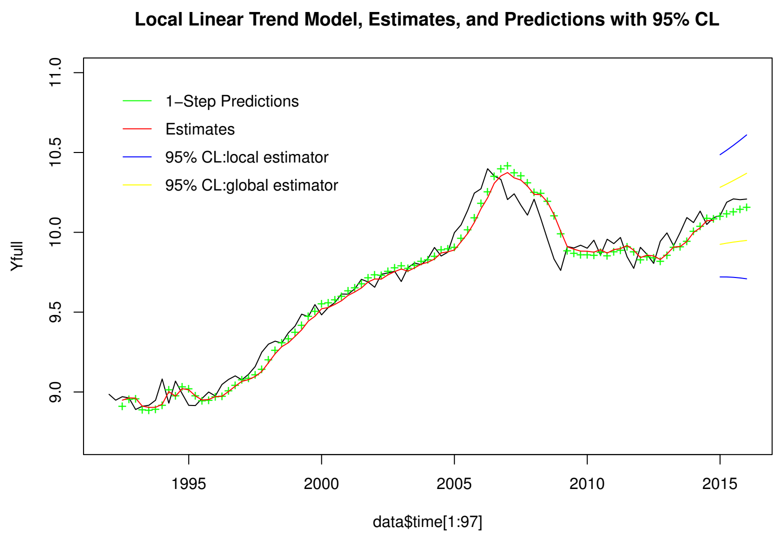

Figure 7: The local linear trend model's 1-step predictions and estimates are shown above for a lambda value of 0.8. The blue and yellow confidence intervals are calculated using the local and global sigma estimates, respectively.

Time	Prediction	σ_{local}	Lower	Upper
2015	10.10343	0.1202627	9.720696	10.48615
2015.25	10.11744	0.1247410	9.720455	10.51442
2015.5	10.13145	0.1298346	9.718256	10.54464
2015.75	10.14546	0.1354740	9.714320	10.57660
2016	10.15947	0.1415940	9.708854	10.61008

Table 3: Predictions of Local Linear Model with 95% CL global est

Time	Prediction	σ_{global}	Lower	Upper
2015	10.10343	0.08989256	9.924838	10.28201
2015.25	10.11744	0.09323995	9.932199	10.30267
2015.5	10.13145	0.09704722	9.938646	10.32425
2015.75	10.14546	0.10583706	9.944283	10.34663
2016	10.15947	0.1415940	9.949206	10.36973

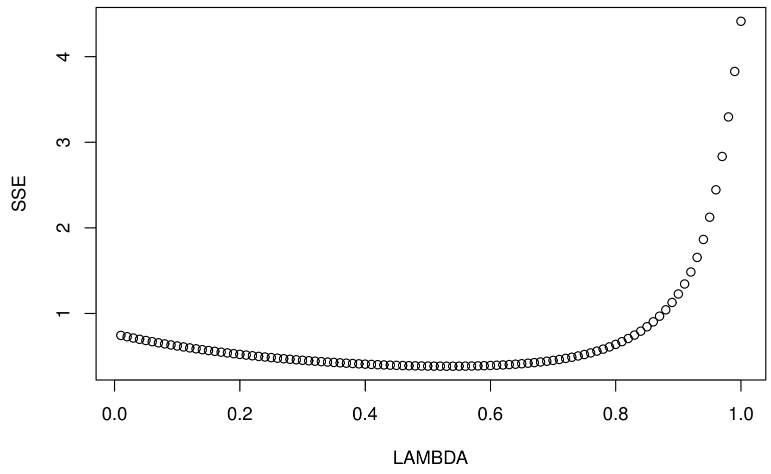

Figure 8: With a burn-in period of 20 observations, an optimal lambda value of 0.53 is found.

1.6 Find an optimal value of the forgetting factor for use in the local trend model suggested in the previous question. (Optimize 1-step predictions. And disregard the first 20 1-step predictions as burn in period.) Optional addon: What is the optimal value if only using data prior to 2005?

Using a burn-in period of 20, we find an optimal lambda of 0.53 as can be shown in Figure 8. When considering only data prior to 2005, we find an optimal lambda value of 0.69 as shown in Figure 9.

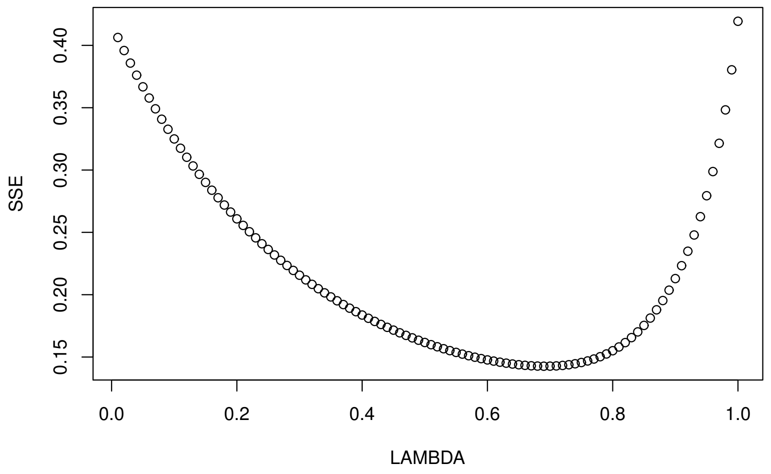

Figure 9: When considering only observations prior to 2005, an optimal lambda value of 0.69 is found.

1.7 Comment on the results. Which model do you prefer? Do you trust the forecasts? Do you have ideas for extending the forecast method?

While I would prefer the local linear trend model, none of the forecasts should be trusted. The simple caveat with our approach is that we're attempting to model a complex nonlinear system with linear methods. While we can approximate nonlinear systems with the local linear trend model, per se, it is nonetheless still an approximation. The complexity of the system is apparent especially when looking at the onset and bust of the housing bubble, which is a component that would be difficult to integrate in our linear models given our input space, which is currently only the time signal. To improve the forecasting method, I would integrate global and local signals (such as proxies for private debt per capita) that reflect the housing market and can potentially capture the surge in unreasonable residential housing prices, and can therefore better predict the housing bubble.

2 Code

```
library(expm)
library(forecast)
data <- read.table("apartment_prices.csv",header = T,sep = "")


## 1.1: Plot apartment price as function of time
Y <- ts(log(data[ 1:92, 3]))
```

```
Yfull <- ts(log(data[1:97, 3]))
plot(data$time, Yfull, pch = 16, type = "l", xlab = "Year", ylab = "
    Apartment Prices log(kr)" )

## 1.2: Does the global mean value and standard deviation of the prices
    give a reasonable representation of the data?
mean = mean(Y)


## 1.3: Formulate GLM: Y = X*theta + error
X <- cbind(rep(1, 92), data$time[1:92])
Xfull <- cbind(rep(1, 97), data$time[1:97])
theta_hat <- solve(t(X)%*%X, t(X)%*%Y)
Y_hat = Xfull%*%theta_hat

## check
lreg <- lm( Y ~ 0 + X )
summary(lreg)
plot(lreg)

#Plot GLM with predictions
plot(data$time, Yfull, type = "l", xlab = "Year", ylab = "Apartment Prices
    log(kr)" )
lines(data$time[1:92], lreg$fitted.values, type = "l", col = "red")
lines(data$time[92:97],  lreg$coefficients[2]*seq(2014.75, 2016, 0.25) +
    lreg$coefficients[1], type = "l" , col = "blue")

## NOTE: OLS is only appropriate when the variance of the errors is
    constant and
##       the residuals are mutually independent




## 1.4 : EXPONENTIAL SMOOTHING MODEL
lambda <- 0.8
Y_hatExpS <- rep(0, 93)

#We initialize the 1st and 2nd forecasts by using Y[1] as starting point (
    S0)
Y_hatExpS[1] <- Y[1]       #We use 1st observation as 1st forecast
Y_hatExpS[2] <- ( 1-lambda ) * ( Y[1] )  +  lambda * Y_hatExpS[1]

#We iterate to find the remaining forecasts using eq. 3.74 from book
for (i in 3:93) {
  Y_hatExpS[i] <-  ( 1-lambda ) * ( Y[i-1] )  +  lambda * Y_hatExpS[i-1]
}

#Predictions: Our last observation is Y[92], so we use eq. 3.75 for
    predicting 2015 and 2016
for (i in 93:97) {
  Y_hatExpS[i] <-  ( 1-lambda ) * ( Y[92] )  +  lambda * Y_hatExpS[i-1]
}

#Check with Holtwinters function
ExSmooth <- HoltWinters( Yfull[1:92] , alpha = (1 - lambda), beta = F,
    gamma = F, l.start = Yfull[1]  )
#PLOT: Exponential Smoothing Model, lambda = 0.8, with predictions and 95%
    CL
forecastsEX <- forecast::forecast.HoltWinters( ExSmooth, h=5, level = .95 )
plot(forecastsEX, main = "Exponential Smoothing and Predictions with 95% CL
    ")
lines(ExSmooth$fitted[,1], lty = 2 , col = "green")
lines(92:97, Yfull[92:97], type = "l")


### 1.5: LOCAL LINEAR TREND MODEL:
lambda <- 0.8
p <- 2

## USEFUL FUNCTIONS to set up problem
#1
x_n  <- function(N) {
  a <- matrix( ncol=2, nrow=N )
  for (i in 1:N) {
    a[i,] <- fgen(-N +i)
  }
  return(a) #returns Nx2 matrix of ones (col1) and -N going down to 0 (col2
      )
}

#2
invSIGMA_n <- function(N, lambda) {
  invS <- matrix(ncol = 1, nrow = N)
  for (i in 1:N) {
    invS[i] <- lambda^(N-i)
  }
  return( Diagonal(N , invS) ) #returns NxN diagonal matrix; it's the
      inverse of SIGMA
}

#3: returns 1x2 matrix
fgen <- function(N)  matrix( c(1,N), nrow=1 )

#4 : [as eq 3.100] : To initialize F_n
F_n <- function(N, lambda) {
  F <- matrix(data = 0, nrow = 2, ncol = 2)
  for (i in 0: (N-1) ) {
    F <- F + lambda^i * t(fgen(-i)) %*% fgen(-i)
  }
  return(F) #returns 2x2 matrix
}

#5 : [as eq 3.100] : To initialize h_n
```

```
h_n <- function(N, lambda) {
  h = matrix(data = 0, nrow = 2, ncol = 1)
  for (i in 0:(N-1) ) {
    h <- h + lambda^i * t(fgen(-i)) %*% Y[N-i]
  }
  return(h) #returns 2x1 matrix
}

#6 : Theta Estimate : [as eq 3.99]
theta.hatLT <- function(N, lambda) { solve(F_n(N, lambda)) %*% h_n(N,
    lambda) }

#7 : Predict Y.. i.e. Y_hat [as eq 3.101]
Y.hatLT <- function(N, l, lambda) { fgen(l) %*% theta.hatLT(N, lambda) }

#8 : as in slides
total_memory_n <- function(N, lambda) {
  return((1 - lambda^(N)) / (1 - lambda))
}

###  PREDICTIONS: YHAT
Y_hatLT <- rep(NA, 97)
for (i in 2:91) { #We start at 2 bc Fn[1] is singular --> no inverse to
    calculate theta
  Y_hatLT[i+1] <- Y.hatLT(i,1, lambda)
}
for (i in 92:96) { #Here we start making predictions L steps ahead from
    last observation: Y[92]
  Y_hatLT[i+1] <- Y.hatLT(92,i-91, lambda)
}

### ESTIMATES: Using F_n(0) instead of F_n(1)
EstimatesLT <- rep(NA, 92)
for (i in 2:91) {
  EstimatesLT[i+1] <- Y.hatLT(i,0, lambda)
}

### UNCERTAINTY: Global and Local Sigma estimator, Variance of Prediction
    error, and 95% confidence interval:

## Global estimator of sigma2 [as in slide 26 in L#4]
residuals <- as.vector(Y[3:92] - Y_hatLT[3:92])
global.est  <- rep(NA, 90)
for (i in 1:90) {
  global.est[i] <- residuals[i]^2 / ( 1 + fgen(1) %*% solve(F_n(i+2, lambda
      )) %*% t(fgen(1)) )
}
sigma2.global <- (sum(global.est) / (90-2))
sigma.global <- sqrt(sigma2.global) #1626

## Local Estimator of sigma2 [as slide 26 in L#4]
```

```
sigma2.local<- ( t(residuals)%*%invSIGMA_n(90, lambda)%*%residuals) /(5-2)
    #/(T-p)
sigma.local <- sqrt(sigma2.local)

## Variance of Prediction Error : [as eq 3.102]
var_PredictionErrorLT.local <- rep(NA, 5)
var_PredictionErrorLT.global <- rep(NA, 5)
for (i in 92:96) {
  var_PredictionErrorLT.local[i-91] <- sigma2.local  * (1 + fgen(i-91) %*%
      solve(F_n(i, lambda)) %*% t(fgen(i-91)))
  var_PredictionErrorLT.global[i-91] <- sigma2.global  * (1 + fgen(i-91) %*
      % solve(F_n(i, lambda)) %*% t(fgen(i-91)))
}

## 95% CL
UpperCL <- matrix(NA,nrow=5, ncol=2) #1st col for local, 2nd for global
LowerCL <- matrix(NA,nrow=5, ncol=2)
for (i in 93:97) {
  #With Local Sigma2 Estimator
  UpperCL[i-92,1] <- Y_hatLT[i]  +  qt( 0.975, df = 5 - p) * sqrt(var_
      PredictionErrorLT.local[i-92])
  LowerCL[i-92,1] <- Y_hatLT[i]  +  qt( 0.025, df = 5 - p) * sqrt(var_
      PredictionErrorLT.local[i-92])

  #With Global Sigma2 Estimator
  UpperCL[i-92,2] <- Y_hatLT[i]  +  qt( 0.975, df = 92 - p) * sqrt(var_
      PredictionErrorLT.global[i-92])
  LowerCL[i-92,2] <- Y_hatLT[i]  +  qt( 0.025, df = 92 - p) * sqrt(var_
      PredictionErrorLT.global[i-92])
}


## PLOT: Local Trend Model, lambda = 0.8, with estimates (f_n(0)), and
    predictions with 95% CL
plot(data$time[1:97], Yfull, type = "l", main = "Local Linear Trend Model,
    Estimates, and Predictions with 95% CL", ylim = c(8.7,11))
points(data$time[1:97], Y_hatLT[1:97], lty = 1 , col = "green", pch = "+")
lines(data$time[1:97], EstimatesLT[1:97], type = "l", col = "red")
lines(data$time[93:97], UpperCL[1:5,1], type = "l", col = "blue")
lines(data$time[93:97], LowerCL[1:5,1], type = "l", col = "blue")
lines(data$time[93:97], UpperCL[1:5,2], type = "l", col = "yellow")
lines(data$time[93:97], LowerCL[1:5,2], type = "l", col = "yellow")
legend( 1992,11,  c("1-Step Predictions", "Estimates", "95% CL:local
    estimator", "95% CL:global estimator"),
         lty = c(1,1,1,1), col =c("green",'red',"blue", "yellow") , bty = "n
             ")


## 1.6: FIND OPTIMAL LAMBDA VALUE: Formulate SSE as a function of lambda
#(Hint: disregard the first 20 1-step predictions as burn in period)
LAMBDA <- seq(0.01,1, by=0.01)
SSE <- matrix(0, nrow = length(LAMBDA), ncol = 1)
```

```r
Y_hat.temp <- rep(0, 92) #We dont use 2015/16 for lambda training
temp <- rep(0, 92)

for (j in 1:length(LAMBDA)) {

  #Predictions under different lambda values
  #Change range to 2:51 if you want to find optimal lambda with data prior
      to 2005, else 20:91
  for (i in 2:51) {
    Y_hat.temp[i+1] <- Y.hatLT(i,1, LAMBDA[j])
    temp[i+1]    <- (Y[i+1] - Y_hat.temp[i+1])^2  #squared residuals
  }
  SSE[j,1] <- sum(temp)
}
plot(LAMBDA, SSE)

## Optimal Lambda with 20 obs burn-in period
index <- which( min(SSE) == SSE )
lambda_opt <- LAMBDA[index] #lambda_opt = 0.53
## Optimal Lambda using data prior to 2005
lambda_opt2005 <- LAMBDA[index] #0.69
```