Assignment 1
Forecasting temperature anomalies

Authors:
Grzegorz Wojciech Zaba, s213035
Andreas With Aspe, s174197
Evangelos Kalimantzalis Lianas, s210260
Francesco Maria D'Antiga, s212354

May 27, 2022

1 Student contributions

All students have participated actively in discussing and calculating the results presented in the sections of this report. The table below shows which students took the main responsibility of each question.

Question 1.1	**Question 1.2**	**Question 1.3**	**Question 1.4**	**Question 1.5**
Francesco	Andreas	Grzegorz	Evangelos	Francesco

2 Question 1.1

Plot below represents temperature anomalies in northern hemisphere development over the time. As it can bee seen, the number of anomalies is increasing with a time. At the beginning mean value was following linear trend, but about 1930 it started to increase.

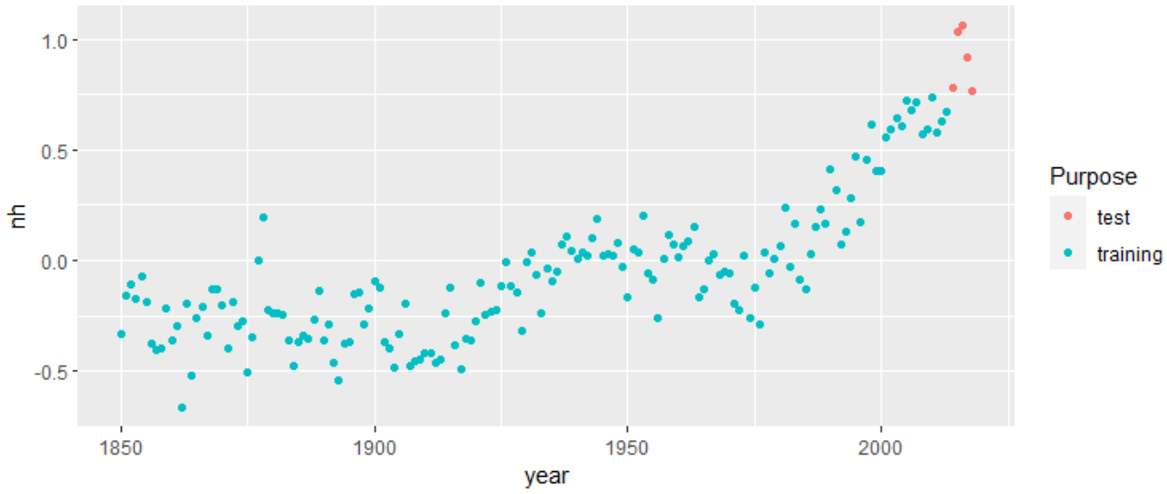


Figure 1: Northern hemisphere temperature anomalies in function of time.

3 Question 1.2

A linear trend model is described by the following function:

$$\mathbf{f}(j) = \begin{pmatrix} 1 \\ j \end{pmatrix} \tag{1}$$

and thus we need to estimate two parameters - θ_0 and θ_1. The global trend model is defined by the below:

$$\mathbf{Y_N} = \mathbf{x_N}\theta_\mathbf{n} + \epsilon \tag{2}$$

where θ is constant in time and each observation has same weight. The same equation is shown in an extended version at (3.87) in the book.

We want to do one step predictions for every point in time. Therefore in each step we update the parameters with the following equations:

$$\mathbf{F}_{N+1} = \mathbf{F}_N + \mathbf{f}(-N)\mathbf{f}^T(-N) \tag{3}$$

and

$$\mathbf{h}_{N+1} = \mathbf{L}^{-1}\mathbf{h}_N + \mathbf{f}(0)Y_{N+1} \tag{4}$$

where $\mathbf{L}$ for linear models are given by

$$\mathbf{L} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{5}$$

Then the new estimate of θ can be found by

$$\hat{\theta}_{N+1} = \mathbf{F}_{N+1}^{-1}\mathbf{h}_{N+1} \tag{6}$$

With this new value for θ, the one step prediction can now be estimated as

$$\hat{\mathbf{Y}}_{N+1|N} = \mathbf{f}^T(1)\hat{\theta}_N \tag{7}$$

In this task the first 3 data points are used to find the first estimate of $\mathbf{F}$ and $\mathbf{h}$.

Since we are concerned about one step prediction error, we can calculate the variance of the prediction error as

$$V[e_N(1)] = \sigma^2[1 + \mathbf{f}^T(1)\mathbf{F}_N^T\mathbf{f}(1)] \tag{8}$$

where

$$e_N(1) = Y_{N+1} - \hat{Y}_{N+1|N} \tag{9}$$

This can then be used to find a prediction interval:

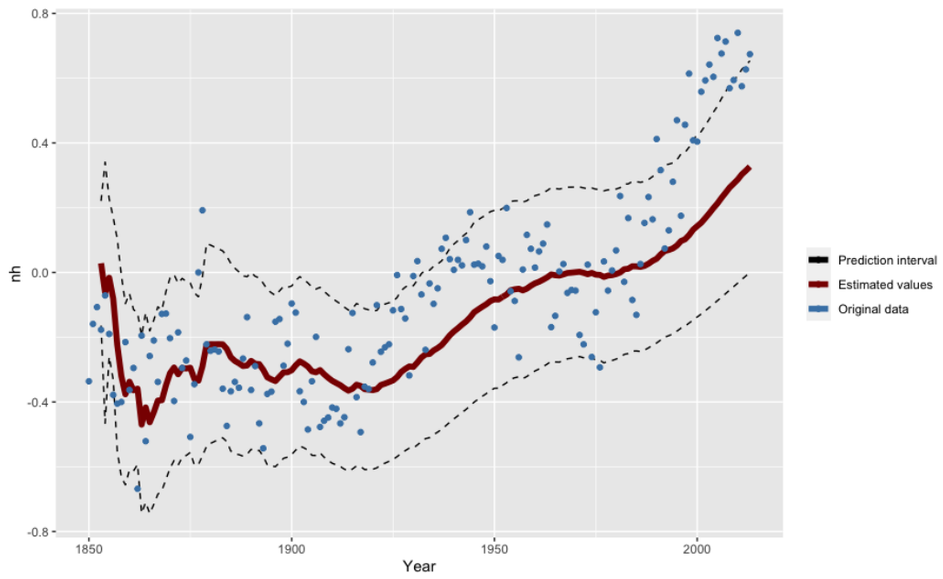
$$\hat{Y}_{N+1|N} \pm t_{\alpha/2}(N-p)\sqrt{V[e_N(1)]} \tag{10}$$

N is the total number of data points used to define the model and p is the number of parameters - in this case $p = 2$, because we are looking at linear trend models.

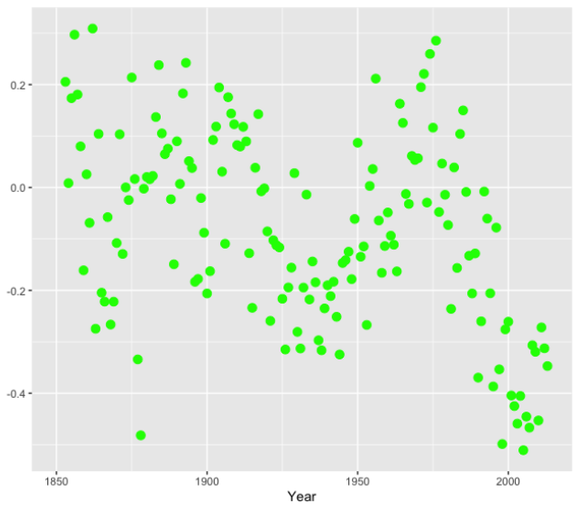
For global trend models an estimate σ^2 is

$$\hat{\sigma}^2 = \epsilon^{\mathbf{T}}\epsilon/(N-p) \tag{11}$$

Then using the above methods, the following plot for the training data can be obtained:



(a) One step prediction values and 95% prediction interval



(b) Prediction errors

Figure 2: Plots for training data

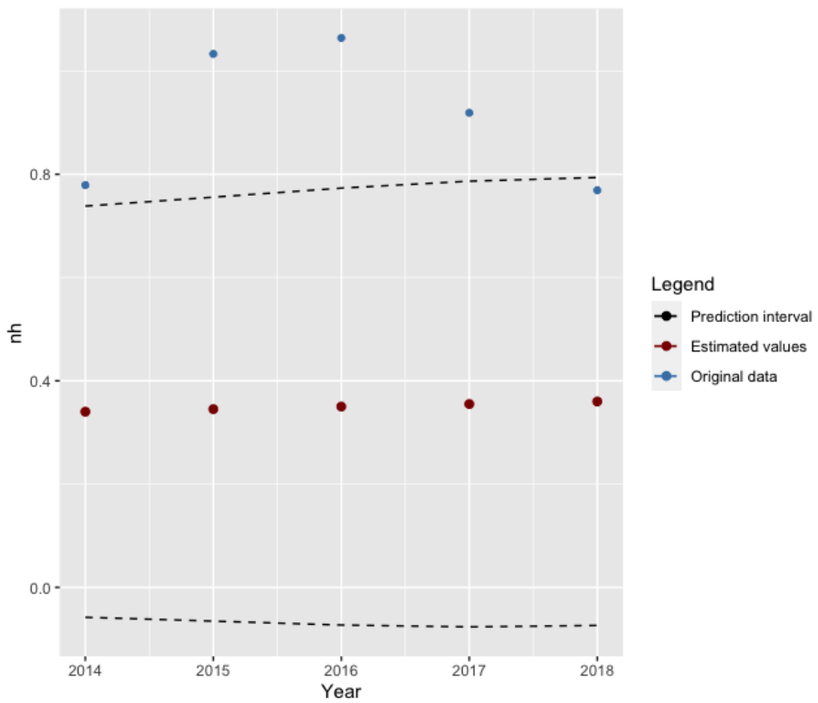
Predictions can now be made for 2014-2018. This is done by using one, two, three, four and five step predictions. This is done by using Equation 7, but substituted with for instance a

number l, if one wants to see an l step prediction. The prediction values and prediction interval is shown in the table below:

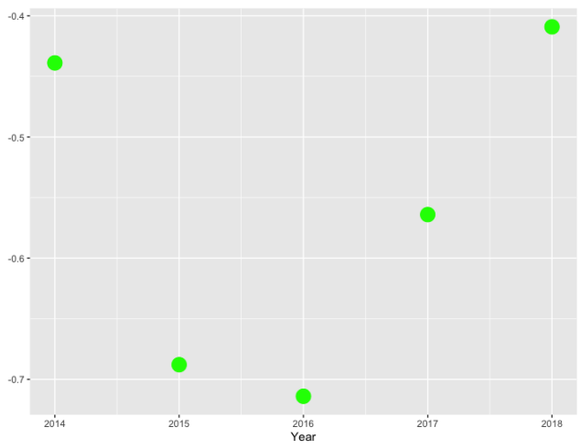
Table 1: Predicted values and corresponding prediction intervals for test data.

Year	Test data	Estimated value	Prediction interval
2014	0.779	0.340	$[-5.80 \cdot 10^{-2}, 0.738]$
2015	1.033	0.345	$[-6.54 \cdot 10^{-2}, 0.756]$
2016	1.064	0.350	$[-7.30 \cdot 10^{-2}, 0.773]$
2017	0.919	0.355	$[-7.66 \cdot 10^{-2}, 0.787]$
2018	0.769	0.360	$[-7.94 \cdot 10^{-2}, 0.794]$

The results can also be plotted. A plot for the predicted values, the original values and prediction interval for test data is shown below:



(a) One step prediction values and 95% prediction interval



(b) Prediction errors

Figure 3: Plots for test data

It is clear from the last two figures that the predicted values does not fit that well to the original data. Almost all the original data points are above the confidence interval of the predictions. This could be a sign of the fact that a global model is not suitable in this case. In next section the linear model is still considered, but now for the local case.

4 Question 1.3

In this part local linear trend model was used to estimate data values and forecasts. Calculations in this problem were initiated with the same x and Y matrices as in previous part. First difference is in initial F and h matrices calculations. Here diagonal matrix Σ of lambdas were taken into account. Σ, F_3 and h_3 were calcualted as follows:

$$\Sigma = \begin{vmatrix} \frac{1}{\lambda^2} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{vmatrix} \tag{12a}$$

$$F_3 = x^{\mathrm{T}} \Sigma^{-1} x \tag{12b}$$

$$h_3 = x^{\mathrm{T}} \Sigma^{-1} Y \tag{12c}$$

Then vector θ_3 was estimated as in previous part and so was predicted value of y. In iterative calculation of time series prediction process, λ coefficient was considered, what led to following recursion:

$$F_{\mathrm{N+1}} = F_{\mathrm{N}} + \lambda^N f(-N) f^{\mathrm{T}}(-N) \tag{13a}$$

$$h_{\mathrm{N+1}} = \lambda \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix} h_{\mathrm{N}} + f(0) Y_{N+1} \tag{13b}$$

Local variance was estimated for every prediction according to equations:

$$\widehat{\mathrm{Var}}_{\mathrm{Local},N} = \hat{\sigma}_N^2 \cdot \left(1 + f^{\mathrm{T}}(1) F_N^{-1} f(1)\right) \tag{14}$$

where

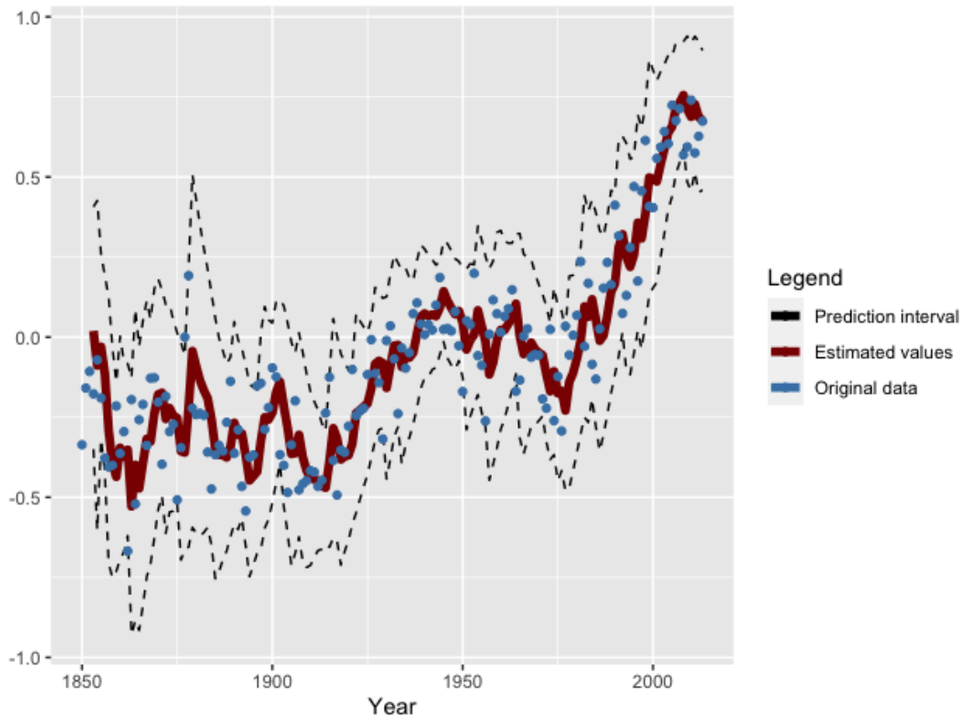
$$\hat{\sigma}_N^2 = \frac{\left(Y_N - x_N \theta_N\right) \Sigma_N^{-1} \left(Y_N - x_N \theta_N\right)}{T_N - 2} \tag{15a}$$

$$T_N = \sum_{j=0}^{N-1} \lambda^j \tag{15b}$$

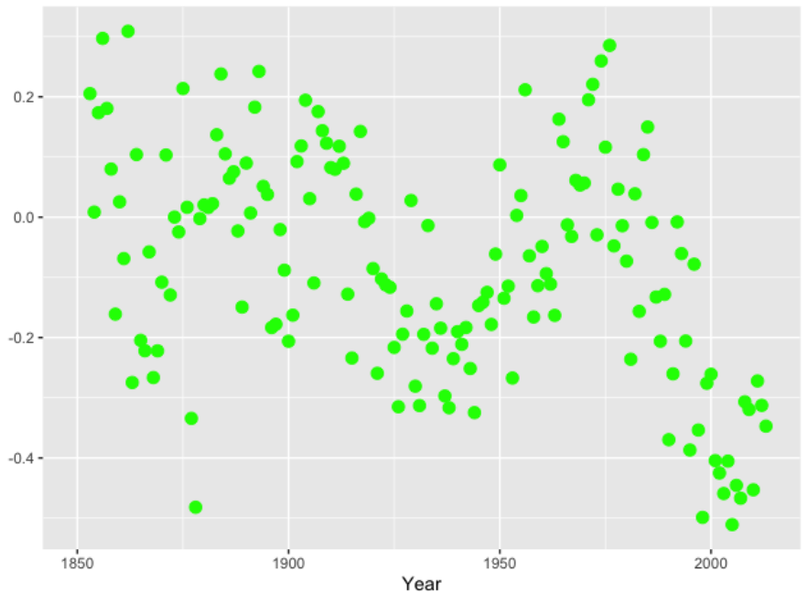
and

$$\Sigma_N = \mathrm{diag}\left(\frac{1}{\lambda^{N-1}}, \frac{1}{\lambda^{N-2}}, \frac{1}{\lambda^{N-3}}, ..., \frac{1}{\lambda}, 1\right) \tag{16}$$

The Σ matrix is updated in every iteration.
Prediction interval was calculated the same way as in Question 1.2.
First of all plots for the training data are shown:



(a) One step prediction values and 95% prediction interval



(b) Prediction errors

Figure 4: Plots for training data

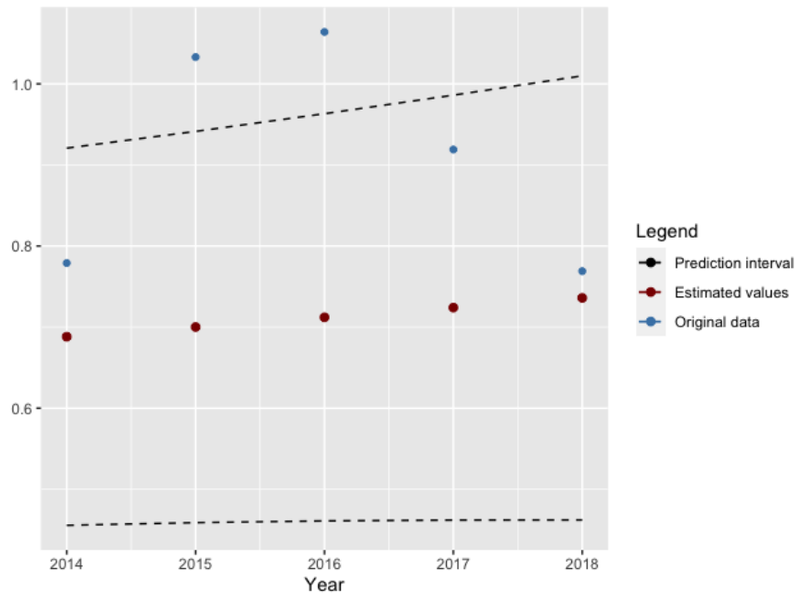
It can be seen that it seems to be a pretty nice fit.

Now the test data is analysed. When making predictions for 2014-2018 with this local linear trend model, one will get the following data:

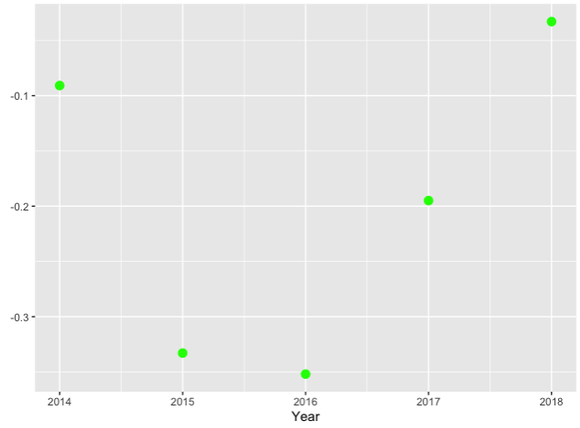
Table 2: Predicted values and corresponding prediction intervals for test data.

Year	Test data	Estimated value	Prediction interval
2014	0.779	0.688	[0.456, 0.921]
2015	1.033	0.700	[0.459, 0.941]
2016	1.064	0.712	[0.461, 0.963]
2017	0.919	0.724	[0.462, 0.986]
2018	0.769	0.736	[0.462, 1.010]

This can also be plotted.



(a) One step prediction values and 95% prediction interval



(b) Prediction errors

Figure 5: Plots for test data

From the table and plots it can be seen that the predicted values are much closer to the original data now compared to the results in question 1.2. However, the estimated values still vary a bit from ordinary test data. For some data points the original data is even above confidence interval for our predictions. This may be due to linear model selection, wrong lambda parameter, inprecise variance measure or the fact, that some test data values (years 2015, 2016 and 2017) are much greater than other data in analysed set. Also prediction interval is very wide, but this may be result of huge variance in data and again using linear model.

As one last test another measure of the variance were taken into account. We still want to calculate the variance of the prediction error as in Equation 14, but now another estimate of $\hat{\sigma}_N^2$ will be used. The estimate in Equation 15a and Equation 15b is a local estimate. This is a good estimator if there is a big difference in the size of prediction errors. But in our case all the data points are fairly dense and located close to each other. Therefore the variance of the prediction error will be close to constant. We therefore try the global estimator, which is better for that case. It is given by

$$\hat{\sigma}_N^2 = \frac{(Y_N - x_N\theta_N)\,\Sigma_N^{-1}\,(Y_N - x_N\theta_N)}{N-n} \tag{17}$$

where N is the total number of training points and n represents the first n predictions that will be left out of the calculation to stabilize the estimator. In our case we chose $n = 3$. Now Σ_N is given by

$$\Sigma_N = \text{diag}(1 + \mathbf{f}^T(1)\mathbf{F}_n^{-1}\mathbf{f}(1), 1 + \mathbf{f}^T(1)\mathbf{F}_{n+1}^{-1}\mathbf{f}(1), \ldots, 1 + \mathbf{f}^T(1)\mathbf{F}_{N-1}^{-1}\mathbf{f}(1)) \tag{18}$$

When using this equation we can get another prediction interval for the last five test points:

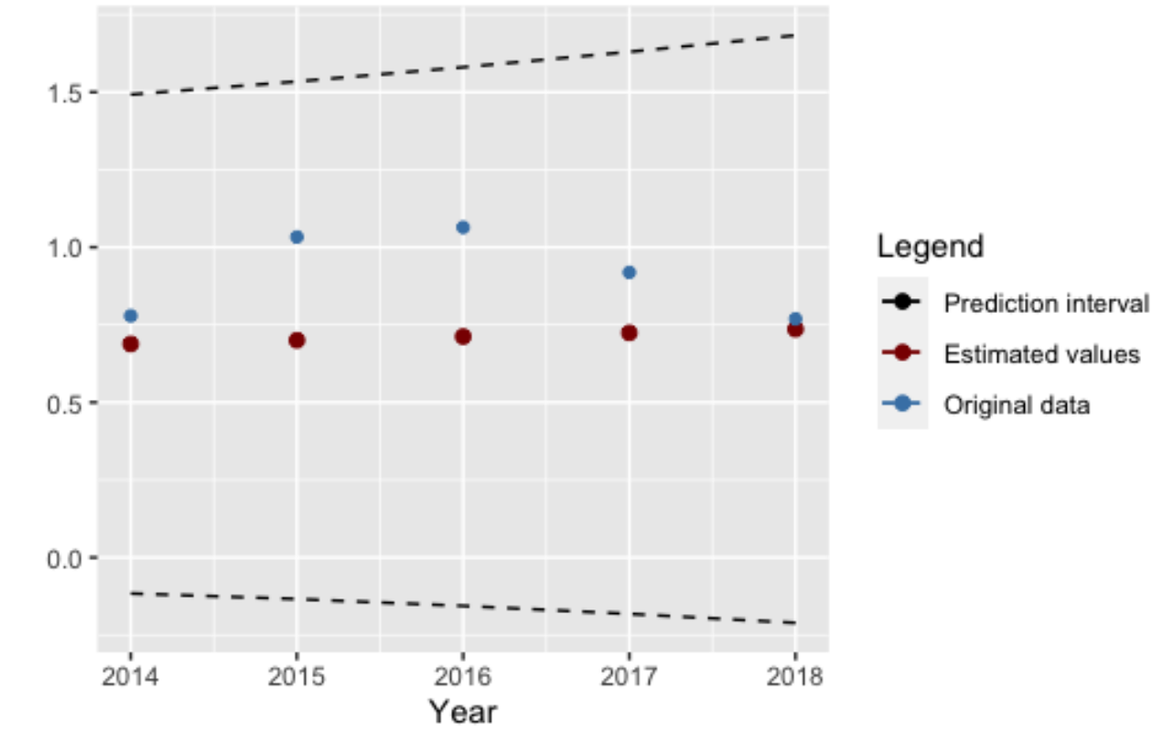


Figure 6: Prediction values and interval with the global measure of $\hat{\sigma}_N^2$

It can be seen that the prediction interval now actually encapsulates the data points, so this might be a better measure.

5 Question 1.4

Using the same Equations As 1.3 we are calculating the optimal forgetting factor λ that minimizes the sum of the errors. We have tried repeatedly possible values between 0.05 and 0.95 with a step of 0.025 to find a value that is close enough to our result (which was 0.85), and after we restricted the range to be from 0.83 to 0.85 with a step of 0.005 in order to have a more precise value. From that we concluded that the minimum error of sums is related to the forgetting factor $\lambda = 0.845$.

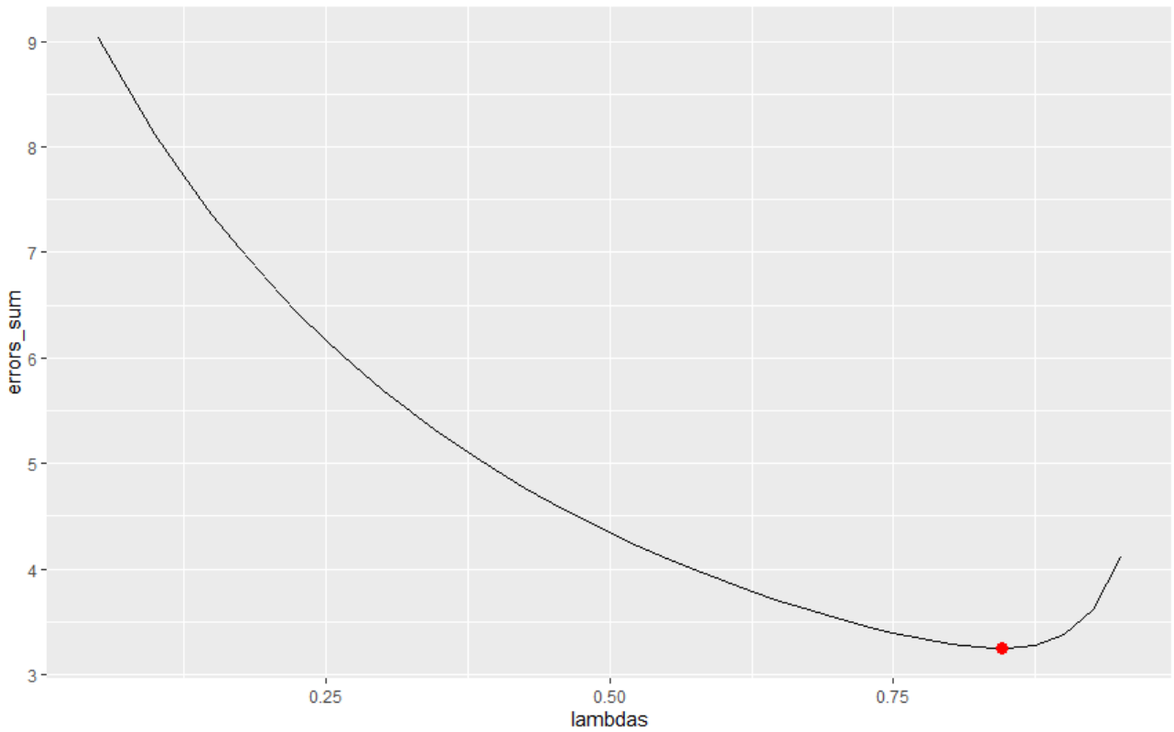


Figure 7: Computing the error sum for each forgetting factor. With red dot is marked the minimum error sum

6 Question 1.5

Taking all results into consideration the local trend model showed better prediction than the global one. It may have been caused by the fact that a linear model was used with a data distribution in time that didn't seem to be overall linear. Even though the local model performed better its forecast wasn't very accurate, so it shouldn't be used to make future predictions. Generically the growth trend was captured correctly by both models, but in each cases their estimated values at specific times differ from the original ones. Some values from the test set aren't even in range of the confidence interval. A possible way to improve the model would be to consider a different, and more elaborate, trend function, for example by including quadratic or exponential term to $f(j)$. This suggestion can also be supported by analyzing the distribution of all the prediction error plots. If we assume the model to have taken into account all the underlying trends, then the prediction error should resemble a white noise distribution, but for instance the plot in Figure 4b not only doesn't resemble white noise, but suggests some harmonic pattern instead. This is a clear sign that the model is not well suited to explain, and predict, the focus of our study, and one should therefore try different trend models. Moreover, in local trend model different forgetting factor could be used, slightly bigger that 0.8 (what analysis in Question 1.4 showed).

A Code

Code used for this assignments is as follows:

```
# ---------- Settings ----------

training_years <- c(1850, 2013)
test_years <- c(2014, 2018)
lambda <- 0.8

# ---------- Imports ----------

library(ggplot2)
library(rstudioapi)
pathA=dirname(rstudioapi::getSourceEditorContext()$path)
raw_data <- read.csv(paste(pathA, '/A1_annual.txt', sep=''),
    header=TRUE, sep='\t')
raw_data <- raw_data[c('year', 'nh')]
training_set <- raw_data[which(raw_data$year==training_years[1])
    :which(raw_data$year==training_years[2]),]
test_set <- raw_data[which(raw_data$year==test_years[1])
    :which(raw_data$year==test_years[2]),]
raw_data['Purpose'] <- vector()
raw_data[which(raw_data$year==training_years[1]):
    which(raw_data$year==training_years[2]),'Purpose']<-'training'
raw_data[which(raw_data$year==test_years[1]):
    which(raw_data$year==test_years[2]),'Purpose']<-'test'

# ---------- Question 1.1 ----------

ggplot(raw_data, aes(x=year, y=nh, color=Purpose)) + geom_point()

# ---------- Question 1.2 calculations ----------

# Defining linear function and L matrix
f <- function(j) rbind(1, j)
L=matrix(c(1,1,0,1), ncol=2, nrow=2)
LInv=solve(L)

# Vector for Time Series estimated values, confidence interval and errors
yhat <- matrix(NA, nrow=nrow(raw_data))
errors <- matrix(NA, nrow=nrow(raw_data))
theta.all <- matrix(NA,ncol=2, nrow=nrow(raw_data))
pred_int <- matrix(NA, ncol=2, nrow=nrow(raw_data))

# Creating x and Y matrix for first time series iteration
x <- matrix(c(1, 1, 1, -2, -1, 0), ncol=2, nrow=3)
Y <- training_set[1:3, 2]

# F and h matrices for first iteration
F <- t(x) %*% x
h <- t(x) %*% Y

# First solution
theta <- solve(F, h)
theta.all[3,] <- theta
yhat[4] <- t(f(1)) %*% theta
errors[4] <- yhat[4]-raw_data[4,2]
epsilon = Y - x %*% theta
sigma2 = (t(epsilon)%*%epsilon)/(4-2)
variance <- sigma2*(1+(t(f(1)) %*% solve(F) %*% f(1)))
#variance <- (t(Y - x %*% theta) %*% (Y - x %*% theta)/(4 - 2))
```

```
    *(1+(t(f(1)) %*% solve(F) %*% f(1)))
pred_int[4,] <- c(yhat[4]-qt(p=0.95,df=4-2)*sqrt(variance),
    yhat[4]+qt(p=0.95,df=4-2)*sqrt(variance))


# Loop for getting all solutions for training set
for (i in 4:nrow(training_set)){
  Y <- training_set[1:i,2]
  x <- rbind(c(1,-i+1), x)
  F <- F + f(-(i-1)) %*% t(f(-(i-1)))
  h <- LInv %*% h + f(0)*training_set[i,2]
  theta <- solve(F, h)
  theta.all[i,] <- theta
    if (i<nrow(training_set)){
      yhat[i+1] <- t(f(1)) %*% theta
      errors[i+1] <- yhat[i+1]-raw_data[i+1,2]
      epsilon = Y - x %*% theta
      sigma2 = (t(epsilon)%*%epsilon)/(i-2)
      variance <- sigma2*(1+(t(f(1)) %*% solve(F) %*% f(1)))
      pred_int[i+1,] <- c(yhat[i+1]-qt(p=0.95,df=i-2)*sqrt(variance)
        ,yhat[i+1]+qt(p=0.95,df=i-2)*sqrt(variance))
    }
}

# Calculating values for test data
init <- which(raw_data$year==test_years[1])
for (i in 1:nrow(test_set)){
  yhat[init-1+i] <- t(f(i)) %*% theta
  errors[init-1+i] <- yhat[init-1+i]-test_set[i,2]
  epsilon = raw_data[1:which(raw_data$year==training_years[2])+i,2]
    - x %*% theta
  sigma2 = (t(epsilon) %*%epsilon)/(nrow(training_set) - 2)
  variance <- sigma2*(1+(t(f(i)) %*% solve(F) %*% f(i)))
  pred_int[init-1+i,] <- c(yhat[init+i-1]-qt(p=0.975,
    df=nrow(training_set)-2)*sqrt(variance),yhat[init+i-1]
        +qt(p=0.975,df=nrow(training_set)-2)*sqrt(variance))
}

# Joining all data into 1 dataframe for plots
raw_data$yhat <- yhat
raw_data$errors <- errors
raw_data <- cbind(raw_data, pred_int)
names(raw_data)[6:7]<-c('lower', 'upper')

# ---------- Question 1.2 plots ----------

train_data <- raw_data[1:which(raw_data$year==training_years[2]),]
colors <- c('Prediction interval'='black', 'Estimated values'='darkred'
    , 'Original data'='steelblue')
ggplot(train_data, aes(x=year,y=nh)) +
  geom_line(aes(y=yhat, color='Estimated values'), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed") +
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed") +
  #geom_line(aes(y=errors, color='Errors')) +
  geom_point(aes(y=nh, color='Original data')) +
  labs(x = "Year",
       y = "nh",
       color = "") +
  scale_color_manual(values = colors)

ggplot(train_data,aes(x=year)) +
```

```
  geom_point(aes(y=errors), colour="green", size = 3) +
  labs(x = "Year",
     y = "",
     color = "")

test_data <- raw_data[which(raw_data$year==test_years[1])
    :which(raw_data$year==test_years[2]),]
ggplot(test_data, aes(x=year,y=nh)) +
  geom_point(aes(y=yhat, color='Estimated values'), size=2) +
  geom_line(aes(y=lower, color='Prediction interval'), linetype = "dashed") +
  geom_line(aes(y=upper, color='Prediction interval'), linetype = "dashed") +
  geom_point(aes(y=nh, color='Original data')) +
  labs(x = "Year",
       y = "nh",
       color = "Legend") +
  scale_color_manual(values = colors)

ggplot(test_data,aes(x=year)) +
  geom_point(aes(y=errors), colour="green", size = 6)+
  labs(x = "Year",
       y = "",
       color = "")


# ---------- Question 1.3 calculations ----------

# Vector for Time Series estimated values, confidence interval and errors
yhat_loc <- matrix(NA, nrow=nrow(raw_data))
errors_loc <- matrix(NA, nrow=nrow(raw_data))
pred_int_loc <- matrix(NA, ncol=2, nrow=nrow(raw_data))
pred_int_loc_new <- matrix(NA, ncol=2, nrow=nrow(test_set))

# Creating x and Y matrix for first time series iteration
x <- matrix(c(1, 1, 1, -2, -1, 0), ncol=2, nrow=3)
Y <- training_set[1:3, 2]
E <- diag(c(1/(lambda^2), 1/lambda, 1), 3,3)

# F and h matrices for first iteration
F <- t(x) %*% solve(E) %*% x
h <- t(x) %*% solve(E) %*% Y

# First solution
theta <- solve(F, h)
yhat_loc[4] <- t(f(1)) %*% theta
errors_loc[4] <- yhat_loc[4]-raw_data[4,2]
T <- 1 + lambda + lambda^2
diagonal <- c(1/lambda^2, 1/lambda^1, 1/lambda^0)
sigma <- diag(diagonal, 3, 3)
variance <- (t(Y - x %*% theta) %*% solve(sigma) %*% (Y - x %*% theta)
    /(T - 2))*(1+(t(f(1)) %*% solve(F) %*% f(1)))
pred_int_loc[4,] <- c(yhat[4]-qt(p=0.95,df=4-2)*sqrt(variance)
    ,yhat[4]+qt(p=0.95,df=4-2)*sqrt(variance))

#Allocate memory
N<-length(training_set[,2])
n<-3
sigma_new <- diag(N-n)

train_idx = nrow(training_set)

# Loop for getting all solutions for training set
```

```
for (i in 4:train_idx){
  sigma_new[i-n,i-n] <- 1+t(f(1))%*%solve(F)%*%f(1)
  F <- F + lambda^(i-1) * f(-(i-1)) %*% t(f(-(i-1)))
  h <- lambda * LInv %*% h + f(0)*training_set[i,2]
  x <- rbind(c(1,-i+1), x)
  theta <- solve(F, h)
  T <- T + lambda^(i-1)
  diagonal <- c(1/lambda^(i-1), diagonal)
  sigmainv <- diag(1/diagonal, i, i)
  if (i<nrow(training_set)){
    yhat_loc[i+1] <- t(f(1)) %*% theta
    errors_loc[i+1] <- yhat_loc[i+1]-raw_data[i+1,2]
    variance <- (t(training_set[1:i,2] - x %*% theta) %*% sigmainv
        %*% (training_set[1:i,2] - x %*% theta)/(T - 2))*(1+(t(f(1)) %*%
        solve(F) %*% f(1))) pred_int_loc[i+1,] <-
        c(yhat_loc[i+1]-qt(p=0.95,df=i-2)*sqrt(variance)
        ,yhat_loc[i+1]+qt(p=0.95,df=i-2)*sqrt(variance))
  }
}

sigma_new_inv <- solve(sigma_new)

init <- which(raw_data$year==test_years[1])
for (i in 1:nrow(test_set)){
  yhat_loc[init-1+i] <- t(f(i)) %*% theta
  errors_loc[init-1+i] <- yhat_loc[init-1+i]-test_set[i,2]
  variance <- (t(training_set[1:train_idx,2] - x %*% theta) %*% sigmainv
    %*% (training_set[1:train_idx,2] - x %*% theta)
        /(T - 2))*(1+(t(f(i)) %*% solve(F) %*% f(i)))
  variance_new <- (t(training_set[4:train_idx,2] - x[4:train_idx,]
    %*% theta) %*% sigma_new_inv %*% (training_set[4:train_idx,2]
    - x[4:train_idx,] %*% theta)
        /(N - n))*(1+(t(f(i)) %*% solve(F) %*% f(i)))
  pred_int_loc[init-1+i,] <- c(yhat_loc[init+i-1]
    -qt(p=0.975,df=nrow(training_set)-2)*sqrt(variance)
    ,yhat_loc[init+i-1]+qt(p=0.975,df=nrow(training_set)-2)
    *sqrt(variance))
  pred_int_loc_new[i,] <- c(yhat_loc[init+i-1]
    -qt(p=0.975,df=nrow(training_set)-2)*sqrt(variance_new)
    ,yhat_loc[init+i-1]+qt(p=0.975,df=nrow(training_set)-2)
    *sqrt(variance_new))
}

# ---------- Question 1.3 plots ----------

# Joining all data together
raw_data$yhat_loc <- yhat_loc
raw_data$errors_loc <- errors_loc
raw_data <- cbind(raw_data, pred_int_loc)
names(raw_data)[10:11]<-c('lower_loc', 'upper_loc')

train_data <- raw_data[1:which(raw_data$year==training_years[2]),]
ggplot(train_data, aes(x=year)) +
  geom_line(aes(y=yhat_loc, color='Estimated values'), size=2) +
  geom_line(aes(y=lower_loc, color='Prediction interval')
    , linetype = "dashed") +
  geom_line(aes(y=upper_loc, color='Prediction interval')
    , linetype = "dashed") +
  geom_point(aes(y=nh, color='Original data')) +
  labs(x = "Year",
       y = "",
```

```
        color = "Legend") +
  scale_color_manual(values = colors)

ggplot(train_data,aes(x=year)) +
  geom_point(aes(y=errors), colour="green", size = 3) +
  labs(x = "Year",
       y = "",
       color = "")

test_data <- raw_data[which(raw_data$year==test_years[1])
    :which(raw_data$year==test_years[2]),]
ggplot(test_data, aes(x=year)) +
  geom_point(aes(y=yhat_loc, color='Estimated values'), size=2) +
  geom_line(aes(y=lower_loc, color='Prediction interval')
    , linetype = "dashed") +
  geom_line(aes(y=upper_loc, color='Prediction interval')
    , linetype = "dashed") +
  geom_point(aes(y=nh, color='Original data')) +
  labs(x = "Year",
       y = "",
       color = "Legend") +
  scale_color_manual(values = colors)

ggplot(test_data,aes(x=year)) +
  geom_point(aes(y=errors_loc), colour="green", size = 3) +
  labs(x = "Year",
       y = "",
       color = "")

test_data <- raw_data[which(raw_data$year==test_years[1])
    :which(raw_data$year==test_years[2]),]
ggplot(test_data, aes(x=year)) +
  geom_point(aes(y=yhat_loc, color='Estimated values'), size=2) +
  geom_line(aes(y=pred_int_loc_new[,1], color='Prediction interval')
    , linetype = "dashed") +
  geom_line(aes(y=pred_int_loc_new[,2], color='Prediction interval')
    , linetype = "dashed") +
  geom_point(aes(y=nh, color='Original data')) +
  labs(x = "Year",
       y = "",
       color = "Legend") +
  scale_color_manual(values = colors)
# ---------- Question 1.4 ----------

seq1 <- seq(from=0.05, to=0.825, by=0.025)
seq2 <- seq(from=0.83, to=0.85, by=0.005)
seq3 <- seq(from=0.875, to=0.95, by=0.025)


lambdas <- c(seq1,seq2,seq3)
lambdas
output_data <- data.frame(lambdas)
output_data$errors_sum <- matrix(NA, nrow=nrow(output_data))

for (i in 1:length(lambdas)){
  # Initiating values
  x <- matrix(c(1, 1, 1, -2, -1, 0), ncol=2, nrow=3)
  Y <- training_set[1:3, 2]
  T <- 1 + lambdas[i] + lambdas[i]^2
  diagonal <- c(1/lambdas[i]^2, 1/lambdas[i]^1, 1/lambdas[i]^0)
  sigma <- diag(diagonal, 3, 3)
```

```
  errors_temp <- 0

  # F and h matrices for first iteration
  F <- t(x) %*% solve(sigma) %*% x
  h <- t(x) %*% solve(sigma) %*% Y

  # First solution
  theta <- solve(F, h)
  yhat_temp <- t(f(1)) %*% theta
  errors_temp <- errors_temp + (yhat_temp-raw_data[4,2])^2

  # Loop for further estimates
  for (j in 4:nrow(training_set-1)){
    F <- F + lambdas[i]^(j-1) * f(-(j-1)) %*% t(f(-(j-1)))
    h <- lambdas[i] * LInv %*% h + f(0)*training_set[j,2]
    x <- rbind(c(1,-j+1), x)
    theta <- solve(F, h)
    yhat_temp <- t(f(1)) %*% theta
    errors_temp <- errors_temp + (yhat_temp-raw_data[j+1,2])^2
  }
  output_data[i, 2] <- errors_temp
}


minimum <- output_data[which.min(output_data$errors_sum),]
minimum

min_DF = data.frame(c(1,2),c(minimum$lambdas,minimum$errors_sum))

# Plot
ggplot(output_data, aes(x=lambdas, y=errors_sum))+ geom_line()+
geom_point(x=minimum$lambdas,y=minimum$errors_sum[1,1],color="red",size=3)
```