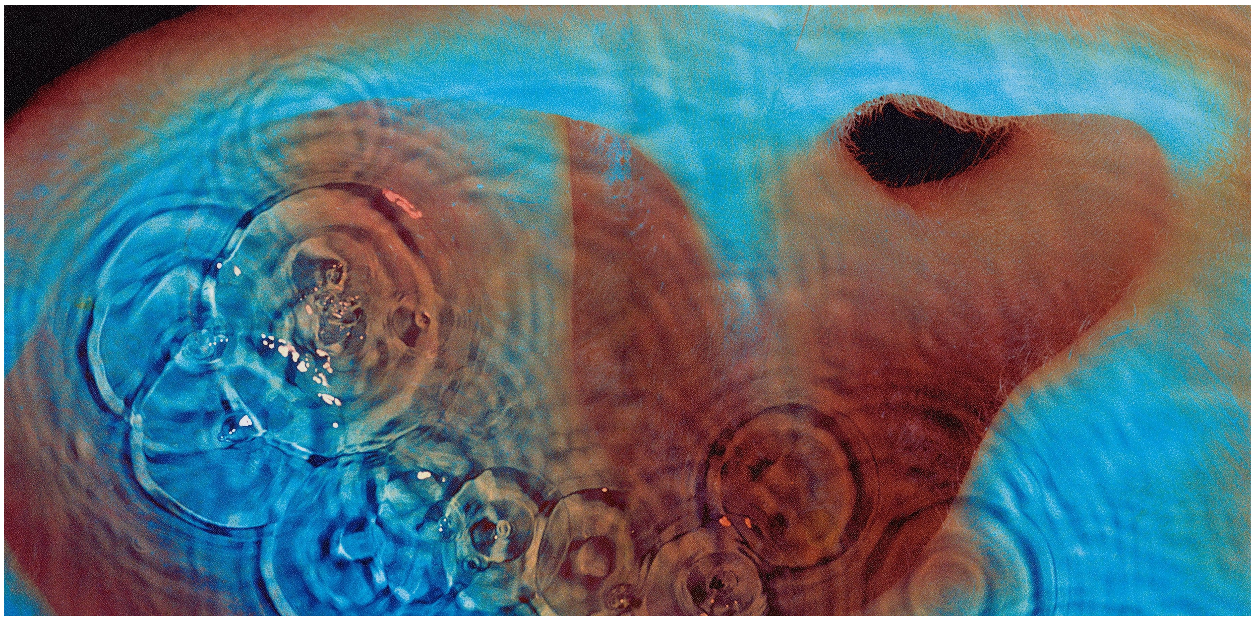
Danmarks
Tekniske
Universitet

Assignment 4: Kalman Filter

02417 - Time Series Analysis



GROUP

Pavlou, Ioannis - s212858
Blachet, Apolline - s222903
Kapakoglou, Georgios - s223001
Kozaris, Charalampos - s230224

January 20, 2024

Contents

List of Figures

List of Tables

Question 4.1: Preliminary data analysis and visualization

The aim of this project is the application of the Kalman Filter on a space-state model of a time series, while at the last section is explored how a more advanced state space model can be deduced. The dataset that is used comes from a sensor platform located in the water near Kulhuse - in the opening of the Roskilde Fjord. The sensor makes measurements about the attributes presented in Table 1 every 30 minutes. The data contains 5000 observations for each attribute, ranging between August and December 2017. Lastly, it shall be mentioned that there are 111 missing values, which has to be taken into consideration when applying the Kalman Filter.

Table 1: Data attributes

Temp	Water temperature $[°C]$
Sal	Water salinity (amount of salt) $[g/kg]$
Depth	Depth of the sensor platform $[m]$
pH	Water pH
Chl	Water chlorophyll concentration $[mg/L]$
ODOsat	Dissolved oxygen saturation $[\%]$
ODO	Dissolved oxygen $[mg/L]$
Battery	Voltage of the battery in the sensor platform $[V]$
DateTime	Time stamp $[0.5 hour]$

The data about dissolved oxygen and salinity are shown in Figures 1 and 3 respectively, along with their zoomed-in versions in Figures 2 and 4. A clear increasing trend in the dissolved oxygen is observed beginning at the end of October, which is expected as dissolved oxygen increases in the winter due to the lower water temperature. Additionally, a slight increase is observed for the water salinity beginning around the same time, although with greater fluctuations. However, this was not expected, since salinity usually decreases in the colder months and increases during the summer period due to the evaporation of water and decreased fresh water flows to the sea. Thus, it is necessary to examine more in depth the process that describes properly the water salinity.

Another important preliminary insight that shall be noted is the existence of potential outliers for both dissolved oxygen and water salinity. When outliers are present in the dataset, they can cause the constructed model to fit the data poorly, resulting in over-fitting or under-fitting. Outliers can also affect the estimates of model parameters, such as coefficients, which can lead to biased predictions. In addition, outliers can distort the relationship between variables, making it difficult to draw meaningful conclusions about the data. Therefore, it is important to identify and handle the outliers appropriately in the modeling process, by filtering the data.

At this point, it shall be mentioned that extra plots regarding the preliminary data visualization are included in the last section of the report, where the project's script material is also provided.

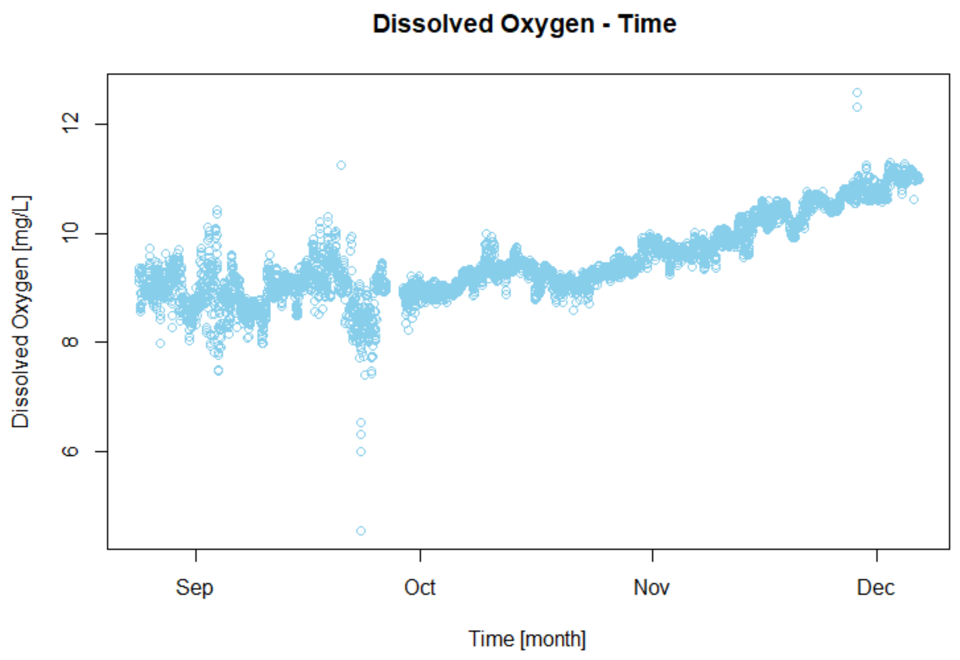


Figure 1: Dissolved oxygen visualization over time

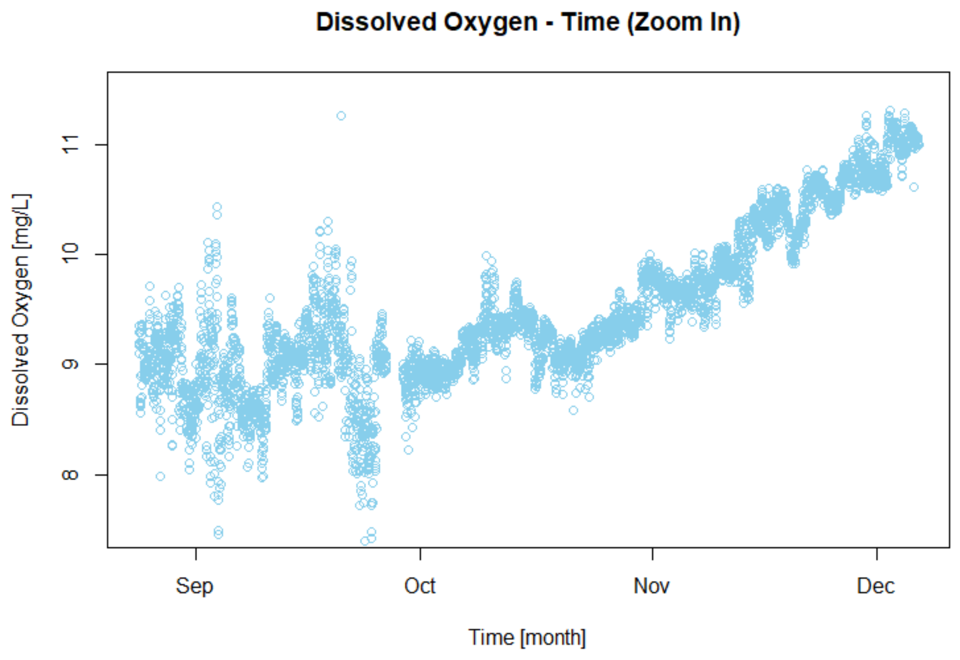


Figure 2: Dissolved oxygen visualization over time (Zoom In)

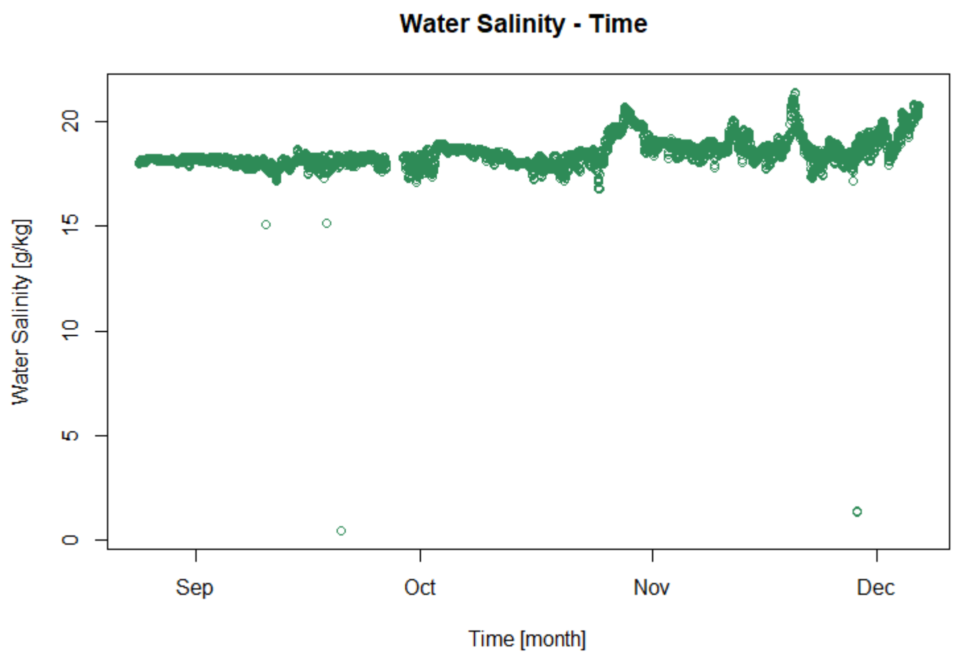


Figure 3: Water salinity visualization over time

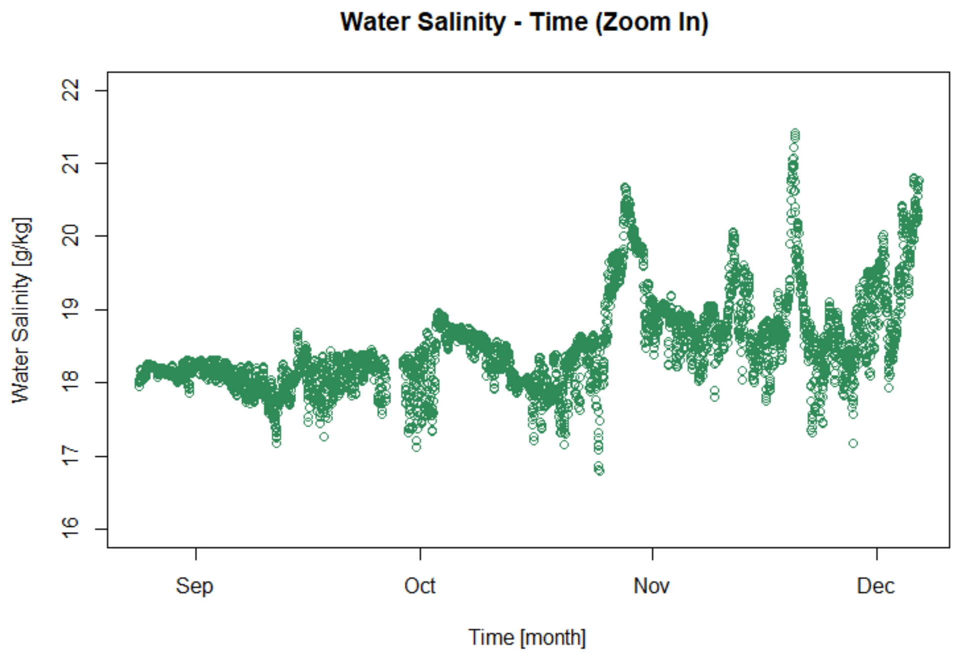


Figure 4: Water salinity visualization over time (Zoom In)

Question 4.2: Random walk state-space model of salinity

Initially, it is assumed that the water salinity can be described by a random walk process, which is observed at equidistant time points. Hence, we consider the following linear state space model formulation:

$$X_t = A_t X_{t-1} + B_t u_{t-1} + e_{1,t} \tag{1}$$
$$Y_t = C_t X_t + e_{2,t}, \tag{2}$$

where X_t is an m-dimensional stochastic state vector, u_t is a deterministic input vector, Y_t is a vector of observable stochastic outputs, and A_t, B_t and C_t are known matrices of suitable dimensions, considered constant in time. Lastly, $e_{1,t}$ and $e_{2,t}$ are the system and the observation noise respectively, which are mutually independent white noise each with their own variance.

Since the examined problem is one-dimensional, all the required matrices will be defined as one-dimensional as well. To begin with, A defines how X_t depends on the previous discrete time step state value X_{t-1} and we assume that $A = [1]$. Regarding B, we are aware that in random processes there is no input and thus $B = [0]$. Furthermore, C describes how Y_t depends on X_t, and, once again we assume that $C = [1]$. Lastly, regarding the variance of the system and the observation noise, Σ_1 and Σ_2 respectively, both are going to be one-dimensional matrices as well.

Thus, the random walk state-space model which describes the water salinity has the following formulation:

$$X_t = X_{t-1} + e_{1,t} \tag{3}$$
$$Y_t = X_t + e_{2,t} \tag{4}$$

Question 4.3: Pure Kalman filter

In this section the salinity data are filtered with a Kalman filter and one-step predictions are generated using the first observation as initial value and the system variance as initial variance. Moreover, the system variance is fixed to $\Sigma_1 = 0.01$ and the observation variance to $\Sigma_2 = 0.005$.

The one step predictions along with the observation data are illustrated in Figure 5. The Kalman filter successfully captures the trend and fluctuations in the salinity data, providing a good estimation of the underlying behavior. The inclusion of the prediction intervals enhances the reliability of the predictions (a zoomed in version is provided in the last section of the report). However, we can see that the Kalman filter is heavily affected by the presence of the outliers in the data. Overall, the Kalman filter appears to perform well in capturing the dynamics of the water salinity data with the exception of the one-step predictions after the outliers.

The respective plot zooming into observations 800 to 950 is illustrated in Figure 6. It can be observed that the Kalman filter tries to catch the outlier, with a result of leaving out data that provide better information for our time series. More specifically, the outlier appearing at time t has an impact on the one-step prediction at time t+1. Since the specific outlier is much smaller than the rest of the data, the derived one-step prediction will underestimate a lot the actual water salinity value.

The standardized one-step prediction errors are illustrated in Figure 7. In most of the periods the prediction errors are close to zero, indicating that the model is performing well and accurately predicting the outcome. However, it can be noted that the residuals corresponding to the outliers have significantly greater values. By zooming in into observations 800 to 950, as illustrated in Figure 8, it can be observed that the residuals are affected for more than one time step by the presence of the outlier and thus decreasing even more the accuracy of the one-step predictions on that segment of the time series.

To conclude, in Table 2 are illustrated the values that define the final state of the Kalman filter.

Table 2: Values that define the final state of the filter (at observation 5000)

kf1	$rec	$pred	$K	$Sigma.xx.rec	$Sigma.yy.rec	$Sigma.xx.pred	$Sigma.yy.pred
Value	20.75815	20.72578	0.7320508	0.003660254	0.01866025	0.01366025	0.01866025

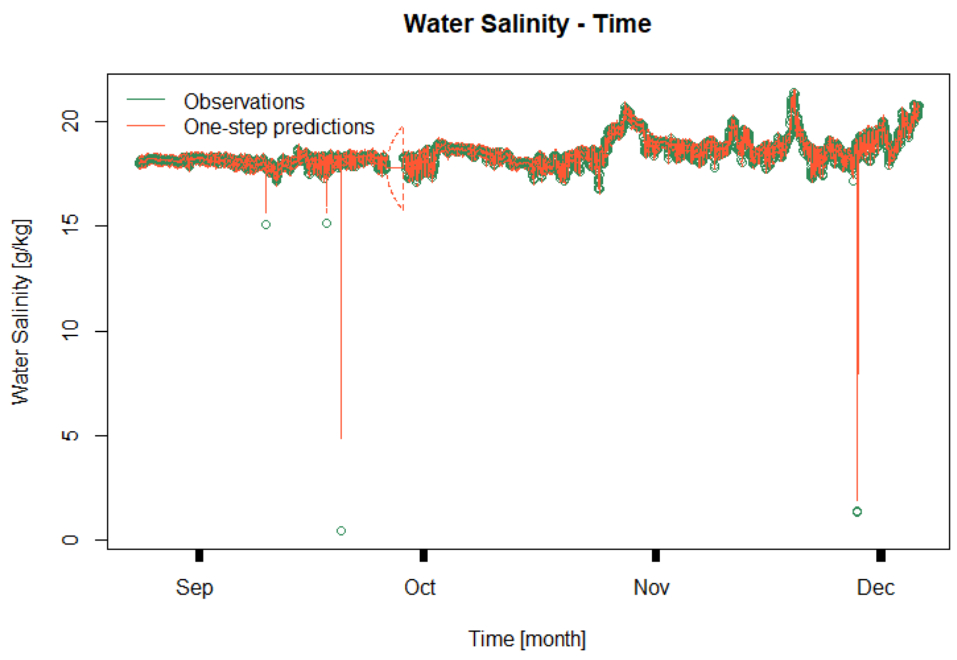


Figure 5: One-step predictions with 95% PI

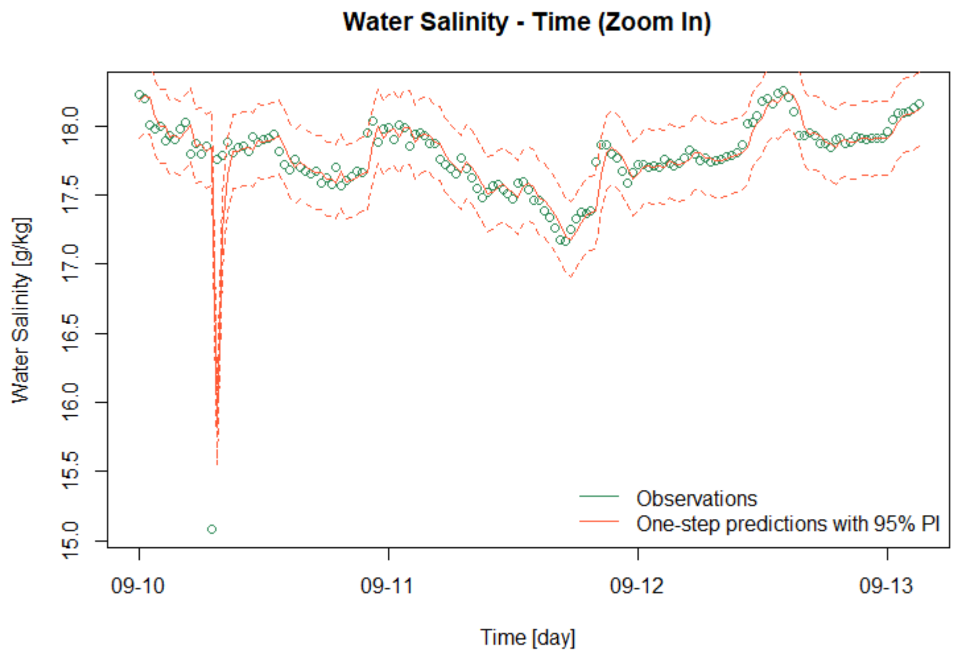


Figure 6: One-step predictions with 95% PI (Zoom In)

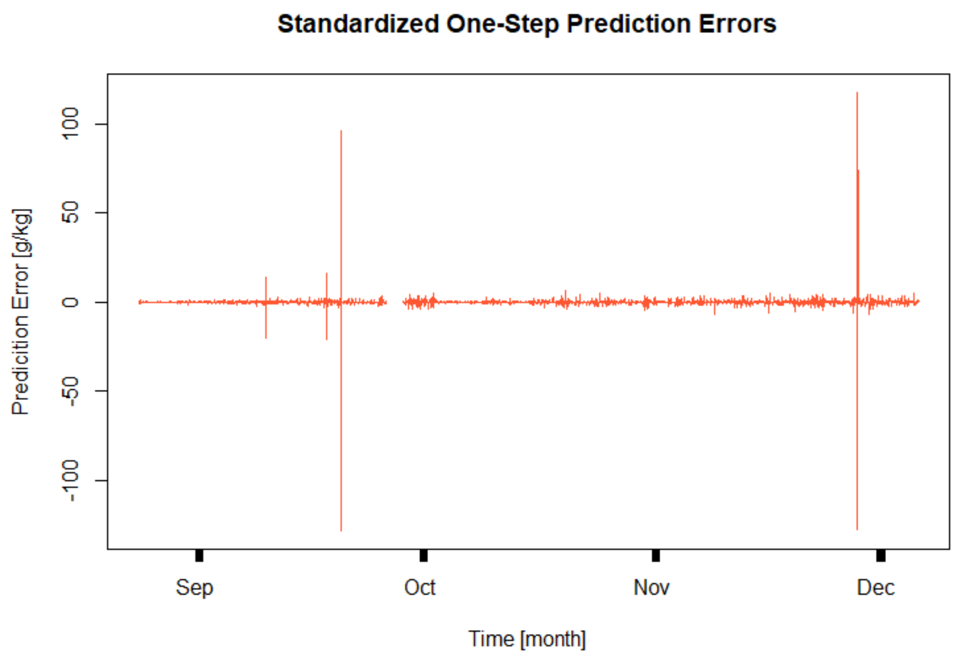


Figure 7: Standardized one-step prediction errors

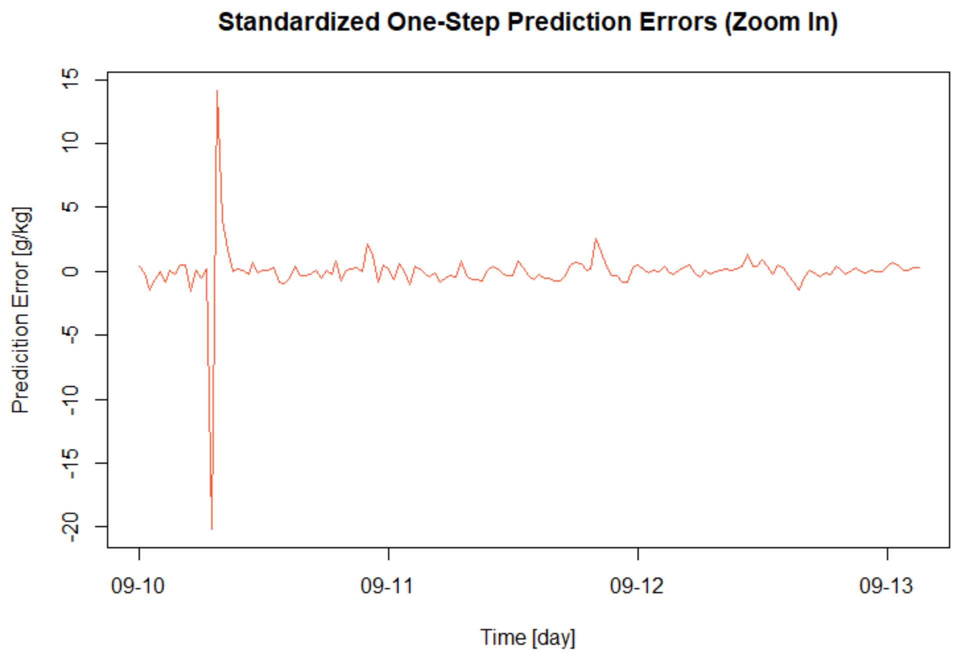


Figure 8: Standardized one-step prediction errors (Zoom In)

Question 4.4: Skipping outliers when filtering

In this section the Kalman filter is modified to not take outliers into account. This means that observations that are more than six standard deviations away from the predictions are treated as missing and a reconstruction step is not performed for them. The filter is implemented on the data and the 1-step predictions for index 800 to 950 are presented in Figure 9:

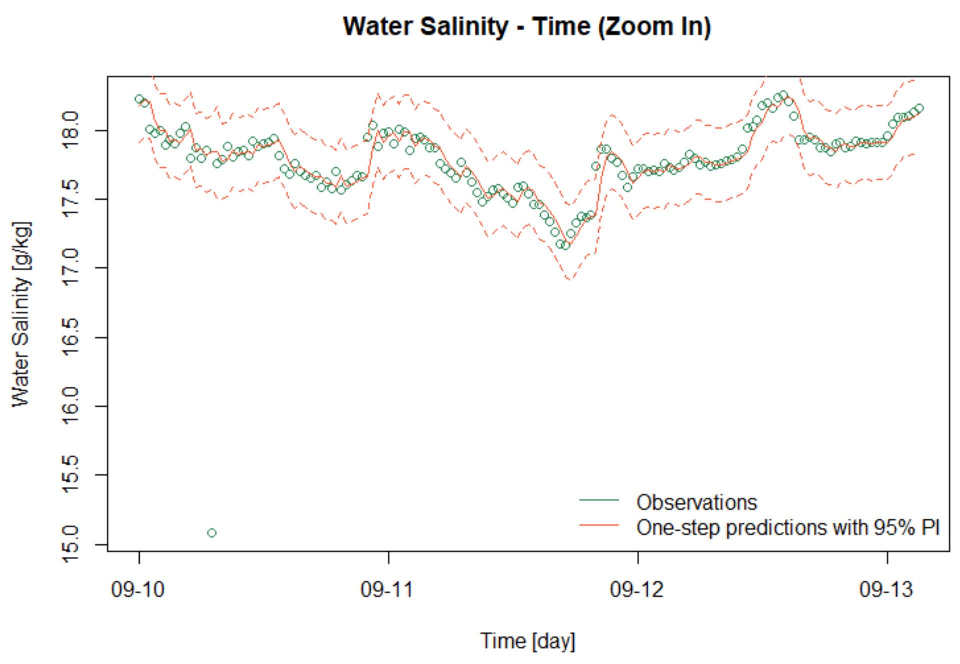


Figure 9: One-step predictions with 95% PI (Zoom In)

It can be observed that the filter successfully removes the influence of the outliers on the one-step prediction, resulting in a more accurate and reliable estimation of the system state. The absence of the outliers' impact on the predictions suggests that the new filter has better filtering performance and is more robust to outliers. This indicates that the new filter has achieved a more accurate and reliable estimate of the underlying system, by better distinguishing between the true signal and the noise or outliers. Overall, Figure 9 demonstrates the effectiveness of the new filtering approach in producing a more accurate and reliable prediction of the water salinity values.

The total number of observations that are skipped is 10 and the indexes of the first five outliers are presented in Table 3. Lastly, the values that define the final state of the filter at observation 5000 are presented in Table 4:

Table 3: Indexes of the first five detected outliers

Outliers	**1**	**2**	**3**	**4**	**5**
Index	814	1203	1296	2733	3692

Table 4: Values that define the final state of the filter (at observation 5000)

kf2	**$rec**	**$pred**	**$K**	**$Sigma.xx.rec**	**$Sigma.yy.rec**	**$Sigma.xx.pred**	**$Sigma.yy.pred**
Value	20.75815	20.72578	0.7320508	0.003660254	0.01866025	0.01366025	0.01866025

Question 4.5: Optimizing the variances

In this section the two variance parameters are optimized by a maximum likelihood estimation, MLE, of their values. A sensible lower bound for the observation variance should take into account the accuracy of the sensor used to measure salinity values. Since sensors are not perfect, measurements have an error of 0.01 with only two decimal digits. Therefore, a small lower bound is desirable to reflect the high precision of the measurements.

In our experiments, we observed that the optimization algorithm, ”L-BFGS-B”, fails to converge when using a bound that is too small. Therefore, we have chosen the lower bound for the variance of the observation error, $\Sigma 1$, to be 8E-4, and the value of 4E-4 for the variance of the system error, $\Sigma 2$, has been selected based on the same rationale. Furthermore, 0.01 and 0.005 have been chosen as initial values respectively for $\Sigma 1$ and $\Sigma 2$. Lastly, the first 800 observations have been used for the optimization process and the results are illustrated in Table 5:

Table 5: Maximum likelihood estimates of the two variance parameters

MLE $\Sigma 1$	**MLE $\Sigma 2$**	**Log-L Value**
0.001767287	0.0004	2081.459

The filter with the optimized parameters was implemented in the data and the results are presented in Figure 6. It can be observed that the updated model produces narrower prediction intervals than before and therefore the predictions can be considered more accurate in general. On the other hand, it can also be seen that some of the observations lie outside the prediction intervals, indicating that they have been treated as outliers and thus decreasing a lot the accuracy of the one step predictions on that segment of the time series. Hence, there is a trade-off between model accuracy and model sensitivity to observations with a relative drastic change that end up outside the prediction intervals. As a consequence, the model becomes less flexible and less robust due to the relatively small applied uncertainty.

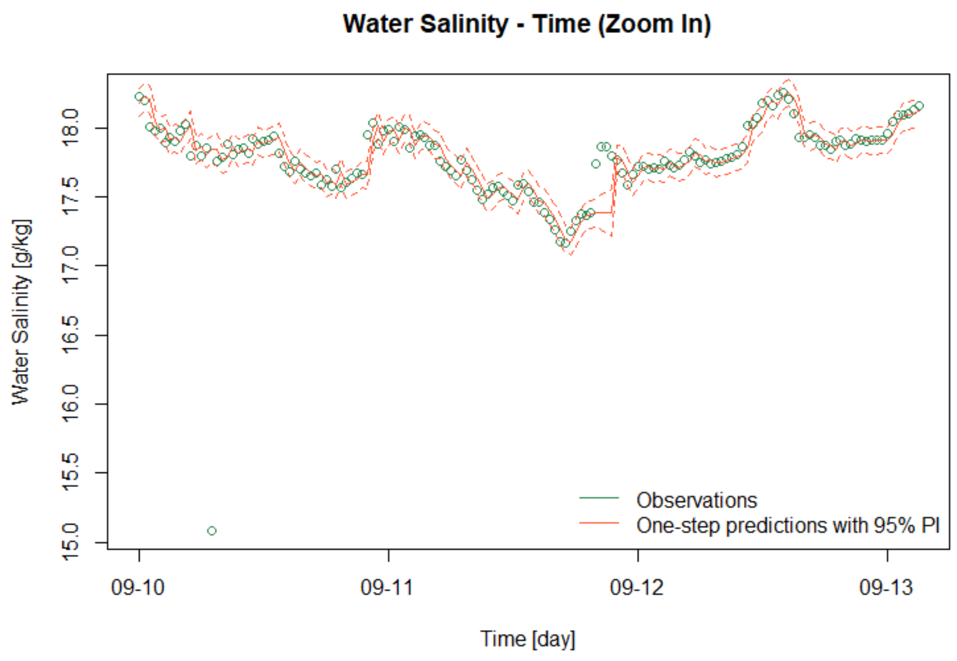


Figure 10: One-step predictions with 95% PI (Zoom In)

Finally,the values that define the final state of the filter are presented in Table 6:

Table 6: Values that define the final state of the filter (at observation 5000)

kf3	**$rec**	**$pred**	**$K**	**$Sigma.xx.rec**	**$Sigma.yy.rec**	**$Sigma.xx.pred**	**$Sigma.yy.pred**
Value	20.76403	20.73264	0.8402156	0.0003360862	0.002503374	0.002103374	0.002503374

Question 4.6: Model for dissolved oxygen

In the final section of the project, a state-space model for dissolved oxygen is formulated. The model incorporates primary production as a linear function of sun intensity and the exchange of oxygen with the atmosphere. Primary production is defined as the process of oxygen production from chlorophyll by absorbing solar energy, whereas respiration refers to the oxygen consumption from all biological matter. The exchange of oxygen with the atmosphere happens when the dissolved oxygen concentration approaches the saturation concentration, which varies based on water salinity and temperature. Additionally, the model incorporates a random walk process for respiration to identify the amount of biomass. Hence, the model is constructed as follows:

$$X_t = AX_{t-1} + Bu_{t-1} + e_{1,t} \tag{5}$$

$$Y_t = CX_t + e_{2,t} \tag{6}$$

with :

$$X_t = \begin{bmatrix} DO_t \\ DO_{sat_t} \\ R_t \end{bmatrix} \tag{7}$$

$$u_t = \begin{bmatrix} I_t \\ T_t \\ Sal_t \end{bmatrix} \tag{8}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

$$B = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & b_{23} \\ 0 & 0 & 0 \end{bmatrix} \tag{10}$$

$$C = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \tag{11}$$

$$Y_t = \begin{bmatrix} DO_t \end{bmatrix} \tag{12}$$

The **system equation** can be explained as follows:

We consider the intensity of sunlight I_t, temperature T_t, and salinity Sal_t as known inputs, denoted as u_t. The variables we aim to model are dissolved oxygen, saturation concentration of dissolved oxygen, and respiration, denoted as X_t.

We know that the model should include respiration as a random walk, which is reflected in line 3 of matrices A and B. Additionally, the saturation concentration of dissolved oxygen depends on the temperature and salinity, as shown in line 2 of matrices A and B. Furthermore, the dissolved oxygen concentration varies during the day due to sunlight, as shown in line 1 of matrix B and it may also depend on water temperature and salinity. Finally, the dissolved oxygen concentration varies during the day due to respiration, and we assume it also depends on the saturation concentration of dissolved oxygen and the previous dissolved oxygen concentration, as shown in line 1 of matrix A.

In terms of the **observation equation**, we only observe the dissolved oxygen concentration.

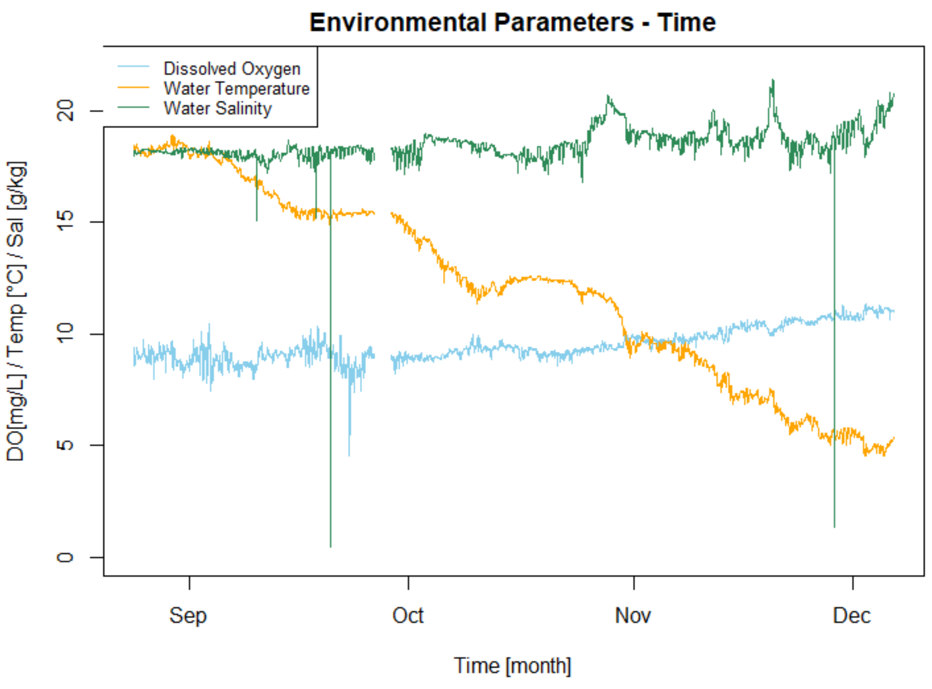


Figure 11: Environmental parameters over time

Code

Listing 1: Question 4.1 - Preliminary data analysis

```r
# Load data
data <- read.csv("A4_Kulhuse.csv")
str(data)

# Check for Nan values
(num_missing <- colSums(is.na(data))) # 111 missing values per
   column
# Convert Datetime from char to datetime object
data$DateTime <- as.POSIXct(data$DateTime, format="%Y-%m-%d %H:%M
   :%S")
str(data)
```

Listing 2: Question 4.1 - Preliminary data visualization

```r
## Dissolved Oxygen over Time
par(mfrow=c(1,1))
plot(data$DateTime, data$ODO, col='skyblue',
     xlab='Time [month]', ylab='Dissolved Oxygen [mg/L]',
     main = 'Dissolved Oxygen - Time ')
abline(h=median(data$ODO), col='darkred', lty=2)

## Dissolved Oxygen over Time - Zoom In
plot(data$DateTime, data$ODO, col='skyblue', ylim=c(7.5,11.5),
     xlab='Time [month]', ylab='Dissolved Oxygen [mg/L]',
     main = 'Dissolved Oxygen - Time (Zoom In)')
abline(h=median(data$ODO), col='darkred', lty=2)

## Salinity over Time
par(mfrow=c(1,1))
plot(data$DateTime, data$Sal, col='seagreen',
     xlab='Time [month]', ylab='Water Salinity [g/kg]',
     main = 'Water Salinity - Time ')
abline(h=median(data$Sal), col='darkred', lty=2)

## Salinity over Time - Zoom In
plot(data$DateTime, data$Sal, col='seagreen', ylim=c(16,22),
     xlab='Time [month]', ylab='Water Salinity [g/kg]',
     main = 'Water Salinity - Time (Zoom In) ')
abline(h=median(data$Sal), col='darkred', lty=2)
```

Listing 3: Question 4.2 - Random walk state-space model of salinity

```
# random walk model
x0 <- data$Sal[1]
A <- matrix(1)
B <- matrix(0)
C <- matrix(1)
# values of sigma do not have to be specified at this question
Sigma1 <- matrix(0.01)
Sigma2 <- matrix(0.005)
```

Listing 4: Question 4.3 - Pure Kalman filter

```
# Kalman filtering using manual implementation
kf1 <-  kalman(data$Sal, A= A, B=B, C=C, Sigma.1=Sigma1, Sigma.2=
    Sigma2,
                 V0=Sigma1, Xhat0=matrix(x0),n.ahead=1,verbose=TRUE)
str(kf1)

# One-step predictions plot
par(mfrow=c(1,1))
plot(data$Sal, col='seagreen',
     xlab='Time [month]', ylab='Water Salinity [g/kg]',
     main = 'Water Salinity - Time', xaxt="n")
with(kf1, matlines(pred[,1]+sqrt(Sigma.yy.pred[1,1,])%*%cbind
    (0,-1.96,1.96),
                   type="l", lty=c(1,2,2), col='#FF5733'))
legend('topleft',
       legend = c('Observations', 'One-step predictions'),
       col = c('seagreen', '#FF5733'),
       lty = c(1, 1),
       bty = 'n')

# Get indices corresponding to the first day of each month
first_day_indices <- which(format(data$DateTime, "%d") == "01")

# Set x-axis labels to month abbreviations
month_labels <- format(data$DateTime[first_day_indices], "%b")
axis(1, at=first_day_indices, labels=month_labels)

# Standatrdized one-step predictions plot
par(mfrow=c(1,1))
plot((data$Sal[-1]-kf1$pred[2:5000])/sqrt(kf1$Sigma.yy.pred
    [1,1,2:5000]), type='l', col='#FF5733',
     xlab='Time [month]', ylab='Predicition Error [g/kg]',
    main='Standardized One-Step Prediction Errors',xaxt="n") # ylim=c(-2,2)
axis(1, at=first_day_indices, labels=month_labels)

# Zoom in (One-step predictions)
par(mfrow=c(1,1))
plot(data$Sal[800:950], col='seagreen',
    xlab='Time [day]', ylab='Water Salinity [g/kg]',
    main = 'Water Salinity - Time (Zoom In)', xaxt="n")
with(kf1, matlines(pred[800:950,1]+sqrt(Sigma.yy.pred[1,1,800:950])%*%cbind(0,-1.96,1.96),
                        type="l", lty=c(1,2,2), col='#FF5733'))
legend('bottomright',
       legend = c('Observations', 'One-step predictions with 95% PI'),
       col = c('seagreen', '#FF5733'),
       lty = c(1, 1),
       bty = 'n')
axis(1, at=seq(1, 150, by=48), labels=format(data$DateTime[seq(800, 950, by=48)], '%m-%d'))

# Zoom in (Standardized errors)
par(mfrow=c(1,1))
plot((data$Sal[800:950]-kf1$pred[800:950])/sqrt(kf1$Sigma.yy.pred[1,1,800:950]), type='l', col='#FF5733',
    xlab='Time [day]', ylab='Predicition Error [g/kg]',
    main='Standardized One-Step Prediction Errors (Zoom In)',xaxt="n")
axis(1, at=seq(1, 150, by=48), labels=format(data$DateTime[seq(800, 950, by=48)], '%m-%d'))
```

Listing 5: Question 4.4 - Skipping outliers when filtering

```
# Skip outliers
sd(na.omit(data$Sal))*6

# Kalman filtering using manual implementation
kf2 <- kalman_outliers2(data$Sal, A= A, B=B, C=C, Sigma.1=Sigma1, Sigma.2=Sigma2,
                V0=Sigma1, Xhat0=matrix(x0),n.ahead=1,verbose=TRUE)
str(kf2)

# Zoom in (One-step predictions)
par(mfrow=c(1,1))
```

```r
plot(data$Sal[800:950], col='seagreen', #ylim=c(15,19),
     xlab='Time [day]', ylab='Water Salinity [g/kg]',
     main = 'Water Salinity - Time (Zoom In)', xaxt="n")
with(kf2, matlines(pred[800:950,1]+sqrt(Sigma.yy.pred
   [1,1,800:950])%*%cbind(0,-1.96,1.96),
                         type="l", lty=c(1,2,2), col='#FF5733'))
legend('bottomright',
        legend = c('Observations', 'One-step predictions with 95%
            PI'),
        col = c('seagreen', '#FF5733'),
        lty = c(1, 1),
        bty = 'n')
axis(1, at=seq(1, 150, by=48), labels=format(data$DateTime[seq
   (800, 950, by=48)], '%m-%d'))

kf2$NB.outsider
kf2$Outsider.index

# Calculating the log-likelihood
neps1 <- (data$Sal[-1]-kf1$pred[2:5000,1])^2 / kf1$Sigma.yy.pred
   [1,1,2:5000]
-0.5 * sum(neps1 + log(kf1$Sigma.yy.pred[1,1,2:5000]),na.rm = TRUE
   )

neps1 <- (data$Sal[-1]-kf2$pred[2:5000,1])^2 / kf2$Sigma.yy.pred
   [1,1,2:5000]
-0.5 * sum(neps1 + log(kf2$Sigma.yy.pred[1,1,2:5000]),na.rm = TRUE
   )
```

Listing 6: Question 4.5 - Optimizing the variances

```r
# Find the ML estimates of the two parameters using the first 800
   observations.
Ypart <- data$Sal[1:800]
my.obj.part <- function(par){
  Kro <- kalman_outliers2(Ypart, A= A, B=B, C=C, Sigma.1=matrix(
      par[1]), Sigma.2=matrix(par[2]),
                    V0=matrix(par[1]), Xhat0=matrix(x0),n.ahead=1,
                         verbose=TRUE)
  nepso <- (Ypart[-1]-Kro$pred[2:800,1])^2 / Kro$Sigma.yy.pred
      [1,1,2:800]
  return(0.5 * sum(nepso + log(Kro$Sigma.yy.pred[1,1,2:800]), na.
      rm = TRUE))
}
```

```r

(Kmopt.part <- optim(c(0.01,0.005),my.obj.part,method = "L-BFGS-B"
    , lower = c(8e-4,4e-4)))

# Filter the data with the optimal parameters and plot
 Sigma1 <- matrix(Kmopt.part$par[1])
 Sigma2 <- matrix(Kmopt.part$par[2])

# Kalman filtering using manual implementation
kf3 <-  kalman_outliers2(data$Sal, A= A, B=B, C=C, Sigma.1=Sigma1,
     Sigma.2=Sigma2,
                 V0=Sigma1,  Xhat0=matrix(x0),n.ahead=1,verbose=TRUE)
str(kf3)

# Zoom in (One-step predictions)
par(mfrow=c(1,1))
plot(data$Sal[800:950], col='seagreen', #ylim=c(15,19),
     xlab='Time [day]', ylab='Water Salinity [g/kg]',
     main = 'Water Salinity - Time (Zoom In)', xaxt="n")
with(kf3, matlines(pred[800:950,1]+sqrt(Sigma.yy.pred
    [1,1,800:950])%*%cbind(0,-1.96,1.96),
                          type="l", lty=c(1,2,2), col='#FF5733'))
legend('bottomright',
         legend = c('Observations', 'One-step predictions with 95%
             PI'),
         col = c('seagreen', '#FF5733'),
         lty = c(1, 1),
         bty = 'n')
axis(1, at=seq(1, 150, by=48), labels=format(data$DateTime[seq
    (800, 950, by=48)], '%m-%d'))

kf3$NB.outsider
kf3$Outsider.index

# Calculating the log-likelihood
neps1 <- (data$Sal[-1]-kf3$pred[2:5000,1])^2 / kf3$Sigma.yy.pred
    [1,1,2:5000]
-0.5 * sum(neps1 + log(kf3$Sigma.yy.pred[1,1,2:5000]),na.rm = TRUE
    )
```

Listing 7: Question 4.3 - Kalman Filter

```r
kalman <- function(Y,A,B=NULL,u=NULL,C,Sigma.1=NULL,Sigma.2=NULL,
    debug=FALSE,V0=Sigma.1,Xhat0=NULL,n.ahead=1,skip=0,verbose=
    FALSE){

  ## predictions through data are one-step predictions. n.ahead
     means
  ## how long we must keep predict after data. These are of course
  ## predictions of different step lengths.

  ## Y has to be columns.
  if(class(Y)=="numeric"){
    dim.Y <- c(length(Y),1)
    Y <- matrix(Y,ncol=1)
  } ## else {
  dim.Y <- dim(Y)
  ##  }

  ## Definition of default variables
  ## A and C must be supplied
  nstates <- dim(A)[1]

  ## these default values don't make much sense
  if(is.null(Sigma.1)){
    Sigma.1 <- diag(rep(1,nstates))
  }
  if(is.null(Sigma.2)){
    Sigma.2 <- diag(rep(1,dim.Y[2]))
  }

  if(is.null(B)){
    B <- matrix(rep(0,nstates),ncol=1)
  }
  if(is.null(u)){
    u <- matrix(rep(0,dim.Y[1]+n.ahead),ncol=1)
  }
  if(is.null(V0)){
    V0 <- Sigma.1
  }
  if(is.null(Xhat0)){
    Xhat0 <- matrix(rep(0,nstates),ncol=1)
  }


  ## i stedet for (10.79)
  X.hat <- Xhat0
  ## (10.80)
  Sigma.xx <- V0
  ## (10.78) (8.78)
  Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

  ## for saving reconstruction
  X.rec <- array(dim=c(dim.Y[1]+n.ahead,nstates))
  X.pred <- array(dim=c(dim.Y[1]+n.ahead,nstates))

  ## for saving K, Sigmas.
  if(verbose){
        K.out <- array(dim=c(dim(Sigma.xx%*%t(C)%*%solve(Sigma.yy)),dim.Y[1]))
        Sigma.xx.rec <- array(dim=c(dim(Sigma.xx),dim.Y[1]))
        Sigma.yy.rec <- array(dim=c(dim(Sigma.yy),dim.Y[1]))
        Sigma.xx.pred <- array(dim=c(dim(Sigma.xx),dim.Y[1]+n.ahead))
        Sigma.yy.pred <- array(dim=c(dim(Sigma.yy),dim.Y[1]+n.ahead))

  }


  for(tt in (skip+1):dim.Y[1]){
    ## (10.75) (8.75)
    K <- Sigma.xx%*%t(C)%*%solve(Sigma.yy)

    ## (10.73) (8.73) - reconstruction
    if(!any(is.na(Y[tt,]))){ # At first everything is thrown away if one is missing
    X.hat <- X.hat+K%*%(t(Y[tt,])-C %*% as.matrix(X.hat))
    X.rec[tt,] <- X.hat
    ## (10.74) (8.74)
    Sigma.xx <- Sigma.xx-K%*%C%*%Sigma.xx
    }

    if(verbose){
        Sigma.xx.rec[,,tt] <- Sigma.xx
        Sigma.yy.rec[,,tt] <- Sigma.yy
    }

    ##(10.76) (8.76) - prediction
    X.hat <- A%*%X.hat + B%*%t(matrix(as.numeric(u[tt,]),nrow=1))
    X.pred[tt+1,] <- X.hat

    ##(10.77) (8.77)
    Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
    ##(10.78) (8.78)
    Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

    if(verbose){
        K.out[,,tt] <- K
#### these are the prediction error variance-covariances
        Sigma.xx.pred[,,tt+1] <- Sigma.xx
        Sigma.yy.pred[,,tt+1] <- Sigma.yy
    }

  }

if(n.ahead>1){
    for(tt in dim.Y[1]+(1:(n.ahead-1))){
      X.hat <- A%*%X.hat + B%*%t(matrix(u[tt,],nrow=1))
      X.pred[tt+1,] <- X.hat
      Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
      Sigma.xx.pred[,,tt+1] <- Sigma.xx
      Sigma.yy.pred[,,tt+1] <- C%*%Sigma.xx%*%t(C)+Sigma.2
    }
  }
  if(verbose){
      out <- list(rec=X.rec,pred=X.pred,K=K.out,Sigma.xx.rec=Sigma
        .xx.rec,Sigma.yy.rec=Sigma.yy.rec,Sigma.xx.pred=Sigma.xx.
        pred,Sigma.yy.pred=Sigma.yy.pred)
  } else {
      out <- list(rec=X.rec,pred=X.pred)
  }
  return(out)
}
```

Listing 8: Question 4.4 - Kalman Filter (Skipping Outliers)

```r
kalman_outliers2 <- function(Y,A,B=NULL,u=NULL,C,Sigma.1=NULL,
    Sigma.2=NULL,debug=FALSE,V0=Sigma.1,Xhat0=NULL,n.ahead=1,skip
    =0,verbose=FALSE){

  ## predictions through data are one-step predictions. n.ahead
     means
  ## how long we must keep predict after data. These are of course
  ## predictions of different step lengths.

  ## Y has to be columns.
  if(class(Y)=="numeric"){
    dim.Y <- c(length(Y),1)
    Y <- matrix(Y,ncol=1)
  } ## else {
  dim.Y <- dim(Y)
  ##  }

  ## Definition of default variables
  ## A and C must be supplied
  nstates <- dim(A)[1]

  ## these default values don't make much sense
  if(is.null(Sigma.1)){
    Sigma.1 <- diag(rep(1,nstates))
  }
  if(is.null(Sigma.2)){
    Sigma.2 <- diag(rep(1,dim.Y[2]))
  }

  if(is.null(B)){
    B <- matrix(rep(0,nstates),ncol=1)
  }
  if(is.null(u)){
    u <- matrix(rep(0,dim.Y[1]+n.ahead),ncol=1)
  }
  if(is.null(V0)){
    V0 <- Sigma.1
  }
  if(is.null(Xhat0)){
    Xhat0 <- matrix(rep(0,nstates),ncol=1)
  }


  ## i stedet for (10.79)
  X.hat <- Xhat0
  ## (10.80)
  Sigma.xx <- V0
  ## (10.78) (8.78)
  Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

  ## for saving reconstruction
  X.rec <- array(dim=c(dim.Y[1]+n.ahead,nstates))
  X.pred <- array(dim=c(dim.Y[1]+n.ahead,nstates))

  ## for saving K, Sigmas.
  if(verbose){
      K.out <- array(dim=c(dim(Sigma.xx%*%t(C)%*%solve(Sigma.yy)),
        dim.Y[1]))
      Sigma.xx.rec <- array(dim=c(dim(Sigma.xx),dim.Y[1]))
      Sigma.yy.rec <- array(dim=c(dim(Sigma.yy),dim.Y[1]))
      Sigma.xx.pred <- array(dim=c(dim(Sigma.xx),dim.Y[1]+n.ahead)
        )
      Sigma.yy.pred <- array(dim=c(dim(Sigma.yy),dim.Y[1]+n.ahead)
        )

  }


  noutsider <- 0
  outsider_index <- NA


  for(tt in (skip+1):dim.Y[1]){
    ## (10.75) (8.75)
    K <- Sigma.xx%*%t(C)%*%solve(Sigma.yy)

    ##########################################################
    # compute predictions for observations checking if obs is more
        than six standard deviations away from the prediction

    ## (10.73) (8.73) - reconstruction
    if(!any(is.na(Y[tt,]))){ # At first everything is thrown away
      if one is missing
    X.hat.test <- X.hat+K%*%(t(Y[tt,])-C %*% as.matrix(X.hat))
    ## (10.74) (8.74)
    Sigma.xx.test <- Sigma.xx-K%*%C%*%Sigma.xx
```

Technical University of Denmark DTU

```
    }

    ##(10.76) (8.76) - prediction
    X.hat.test <- A%*%X.hat.test + B%*%t(matrix(as.numeric(u[tt,]),nrow=1))
    ##(10.77) (8.77)
    Sigma.xx.test <- A%*%Sigma.xx.test%*%t(A)+Sigma.1
    ##(10.78) (8.78)
    Sigma.yy.test <- C%*%Sigma.xx.test%*%t(C)+Sigma.2

    ###############################################################

    # If obs is not NA AND error is too large
    if (tt<length(Y) && !any(is.na(Y[tt+1,])) && abs(X.hat.test-Y[tt+1]) > 6*sqrt(Sigma.yy.test)) {
        #print(X.hat.test-Y[tt])
        #print(6*sqrt(Sigma.yy.test))
        Y[tt+1] <- NA # put obs to NA
        noutsider <- noutsider+1 # count number of outsider
        outsider_index <- c(outsider_index,tt+1) # store index
    }

    ###############################################################

    # run loop as usual

    ## (10.73) (8.73) - reconstruction
    if(!any(is.na(Y[tt,]))){ # At first everything is thrown away if one is missing
    X.hat <- X.hat+K%*%(t(Y[tt,])-C %*% as.matrix(X.hat))
    X.rec[tt,] <- X.hat
    ## (10.74) (8.74)
    Sigma.xx <- Sigma.xx-K%*%C%*%Sigma.xx
    }

    if(verbose){
        Sigma.xx.rec[,,tt] <- Sigma.xx
        Sigma.yy.rec[,,tt] <- Sigma.yy
    }

    ##(10.76) (8.76) - prediction
    X.hat <- A%*%X.hat + B%*%t(matrix(as.numeric(u[tt,]),nrow=1))
    X.pred[tt+1,] <- X.hat

```

```
      ##(10.77) (8.77)
      Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
      ##(10.78) (8.78)
      Sigma.yy <- C%*%Sigma.xx%*%t(C)+Sigma.2

      if(verbose){
          K.out[,,tt] <- K
#### these are the prediction error variance-covariances
          Sigma.xx.pred[,,tt+1] <- Sigma.xx
          Sigma.yy.pred[,,tt+1] <- Sigma.yy
      }

   }

if(n.ahead>1){
      for(tt in dim.Y[1]+(1:(n.ahead-1))){
        X.hat <- A%*%X.hat + B%*%t(matrix(u[tt,],nrow=1))
        X.pred[tt+1,] <- X.hat
        Sigma.xx <- A%*%Sigma.xx%*%t(A)+Sigma.1
        Sigma.xx.pred[,,tt+1] <- Sigma.xx
        Sigma.yy.pred[,,tt+1] <- C%*%Sigma.xx%*%t(C)+Sigma.2
      }
   }
   if(verbose){
        out <- list(rec=X.rec,pred=X.pred,K=K.out,Sigma.xx.rec=Sigma
            .xx.rec,Sigma.yy.rec=Sigma.yy.rec,Sigma.xx.pred=Sigma.xx.
            pred,Sigma.yy.pred=Sigma.yy.pred,NB.outsider=noutsider,
            Outsider.index=outsider_index[2:length(outsider_index)])
   } else {
        out <- list(rec=X.rec,pred=X.pred)
   }
   return(out)
}
```

Extra Plots

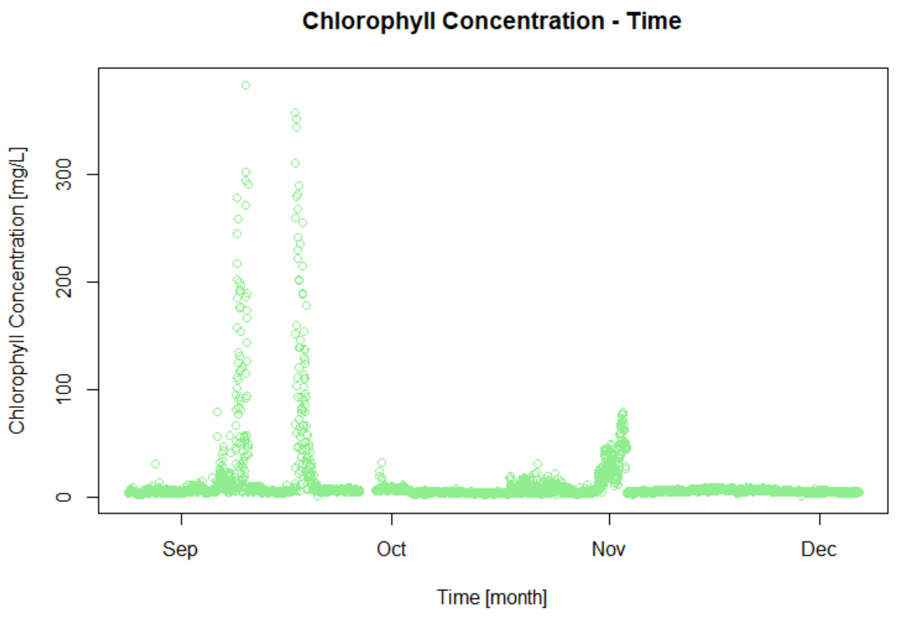


Figure 12: Q.4.1 - Chlorophyll concentration over time

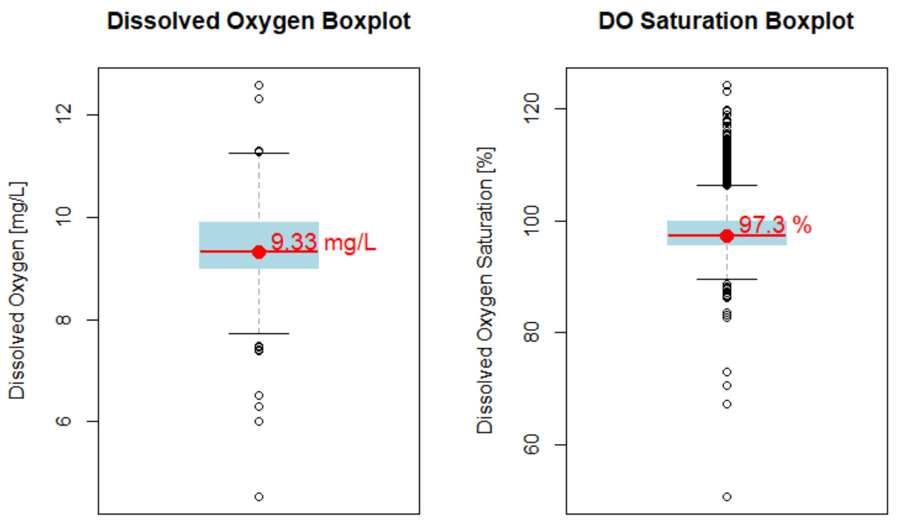


Figure 13: Q.4.1 - Dissolved oxygen boxplots

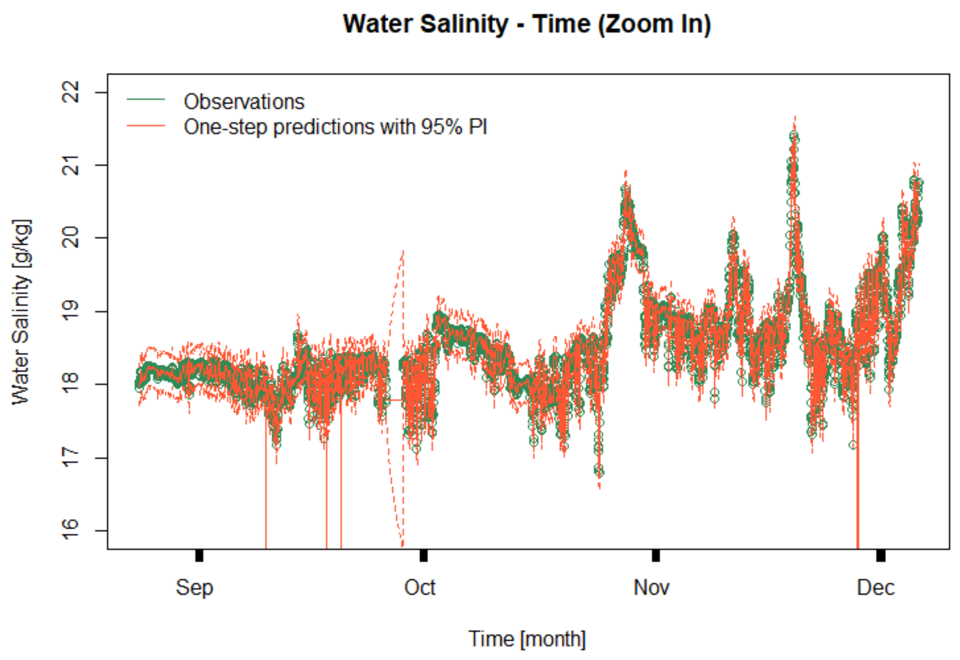


Figure 14: Q.4.3.1 - One-step predictions with 95% PI (Zoom In)

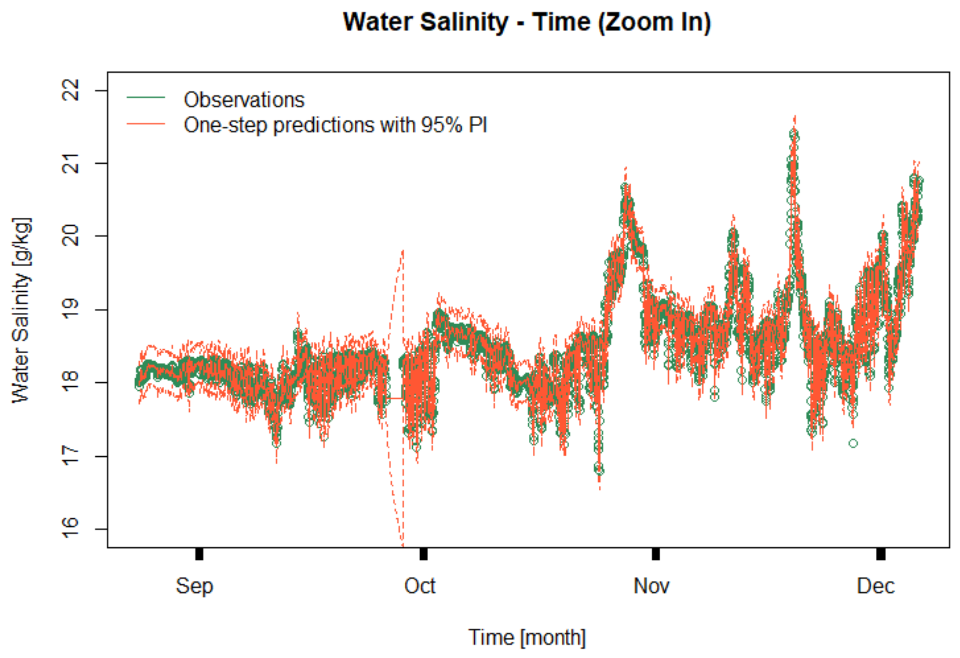


Figure 15: Q.4.4 - One-step predictions with 95% PI (Skipping Outliers)