Ramiro Mata (s161601)

Assignment 4: Time Series Analysis:Multivariate Processes: MARIMA and Kalman Filter Comparison

Ramiro Mata (s161601)

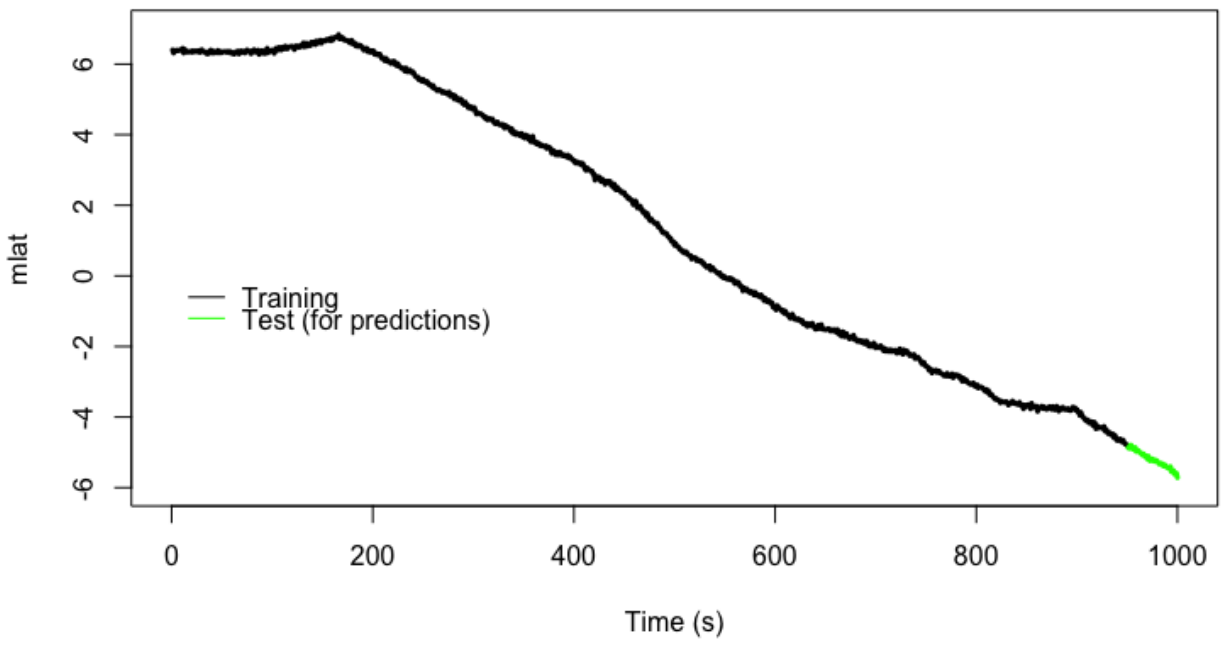

Figure 1: Mlatitude as a function of time. The black line represents the observations used to train the model while the green are the observations left out for training.

1 Presenting the data

Figures 1 and 2 show the change in latitude and longitude, respectively, over the sampled time interval. Figure 3 combines both latitude and longitude to show the actual path taken as recorded by the GPS. As can be seen by the GPS tracker, the individual walks East in almost a straight line, then turns SouthEast walking diagonally. Finally, towards the end, there are a couple of fast turn to ultimately walk South onto the last 50 observations. Notice that the last 50 observations are for evaluating our predictions and are not used for training the models.

In terms of stationarity, by definition we know that the individual is moving and so the longitude and latitude are changing constantly without tending to return to a mean value. Therefore, we say the the GPS longitude and latitude time series observations are not stationary.

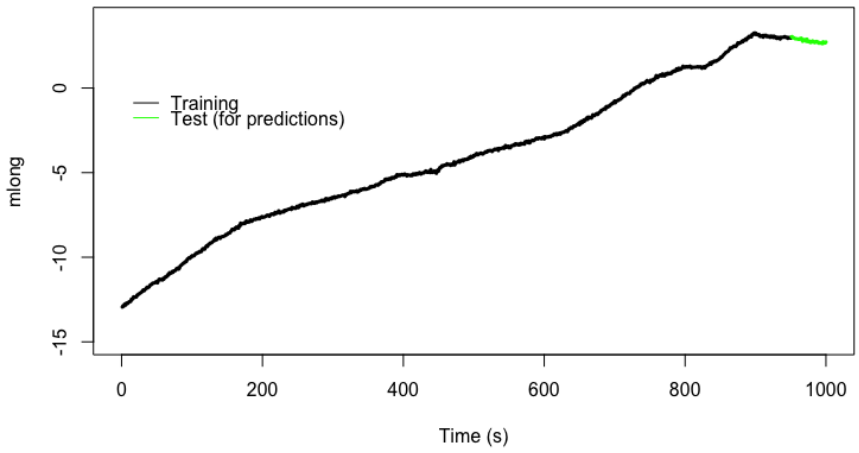

Figure 2: Mlongitude as a function of time

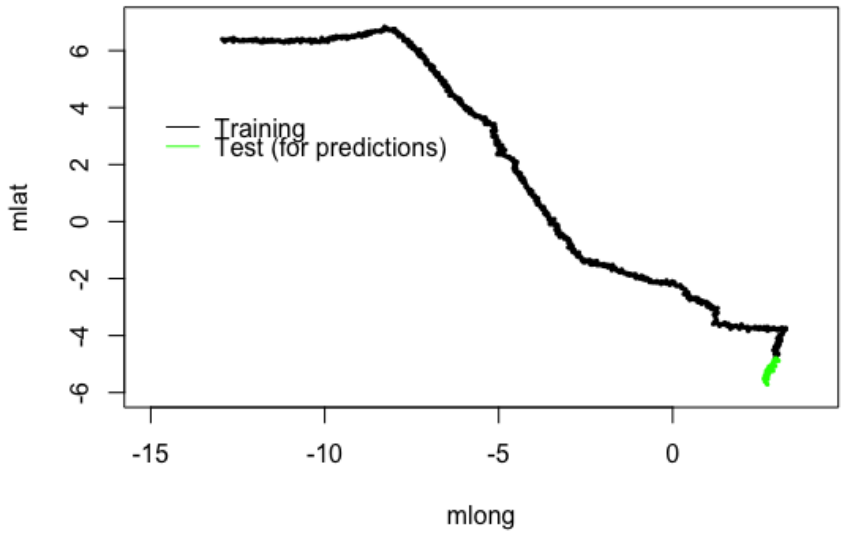

Figure 3: Mlatitude and Mlongitude showing change in direction

2 ACF, PACF, CCF, PCCF

2.1 Without Differencing the Data (Position)

As shown in figure 4, the ACF/CCF of the original time series does not exhibit the classic exponential drop to 0 displayed by clear AR processes. And as mentioned in the previous section, this time series is non-stationary (for both variables) and here the ACF/CCF is in line with our findings. This hints us to consider differencing our data.

The PACF/PCCF show one clear spike at lag 1 and another at lat 2 going above the 95% confidence intervals, possibly suggesting an AR(1) or AR(2) component for the initial MARIMA model. However, differencing the data might give better results.

2.2 Differencing the Data (Velocity)

Figure 6 shows the ACF/CCF of the differenced data, and now we have much better results. We can see that after two spikes in lags 1 and 2, the ACFs drop to 0, possibly suggesting an MA(1) or MA(2) process. However, the PACF doesn't quite exhibit an damped exponential drop behavior, and the PCCF shows lags from -5 to +5 outside the 95% confidence bounds.

2.3 Differencing the Data Twice (Acceleration)

We can see from figure 8 that the ACF and CCF of two times differenced data looks very similar to the case when it was only differenced once (fig. 6) except that now lag 3 also has a spike above the confidence bounds. In contrast, however, the PACF and PCCF now behaves as something that we can recognize from the "Golden Table." In particular, figure 9 shows how the PACF of the series after being differenced twice shows a damped exponential drop to 0 - it decays to zero as it nears lag 9. We can infer from the ACF/CCF and the PACF/PCCF plots that the initial model can contain an MA(3) process.

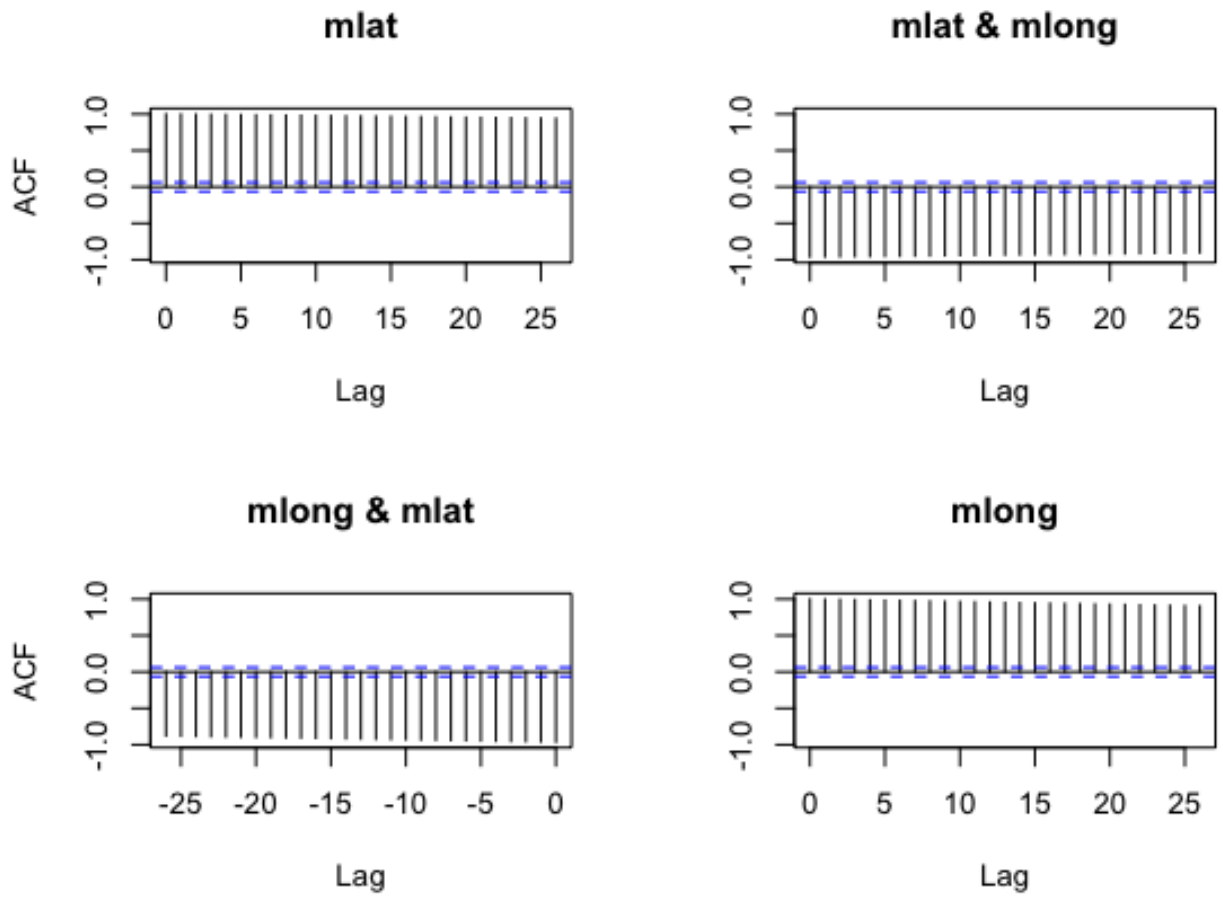

Figure 4: ACF and CCF of original time series

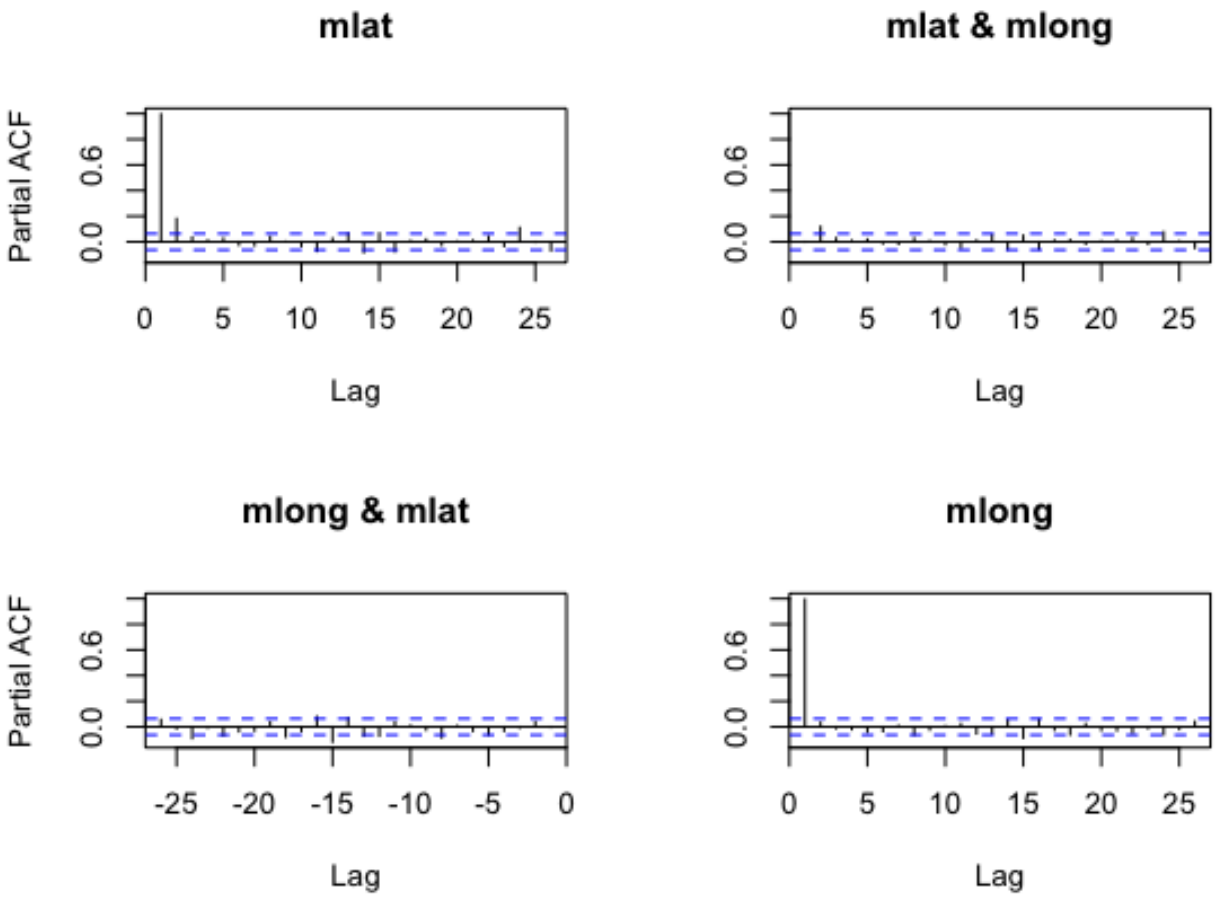

Figure 5: PACF and PCCF of original time series

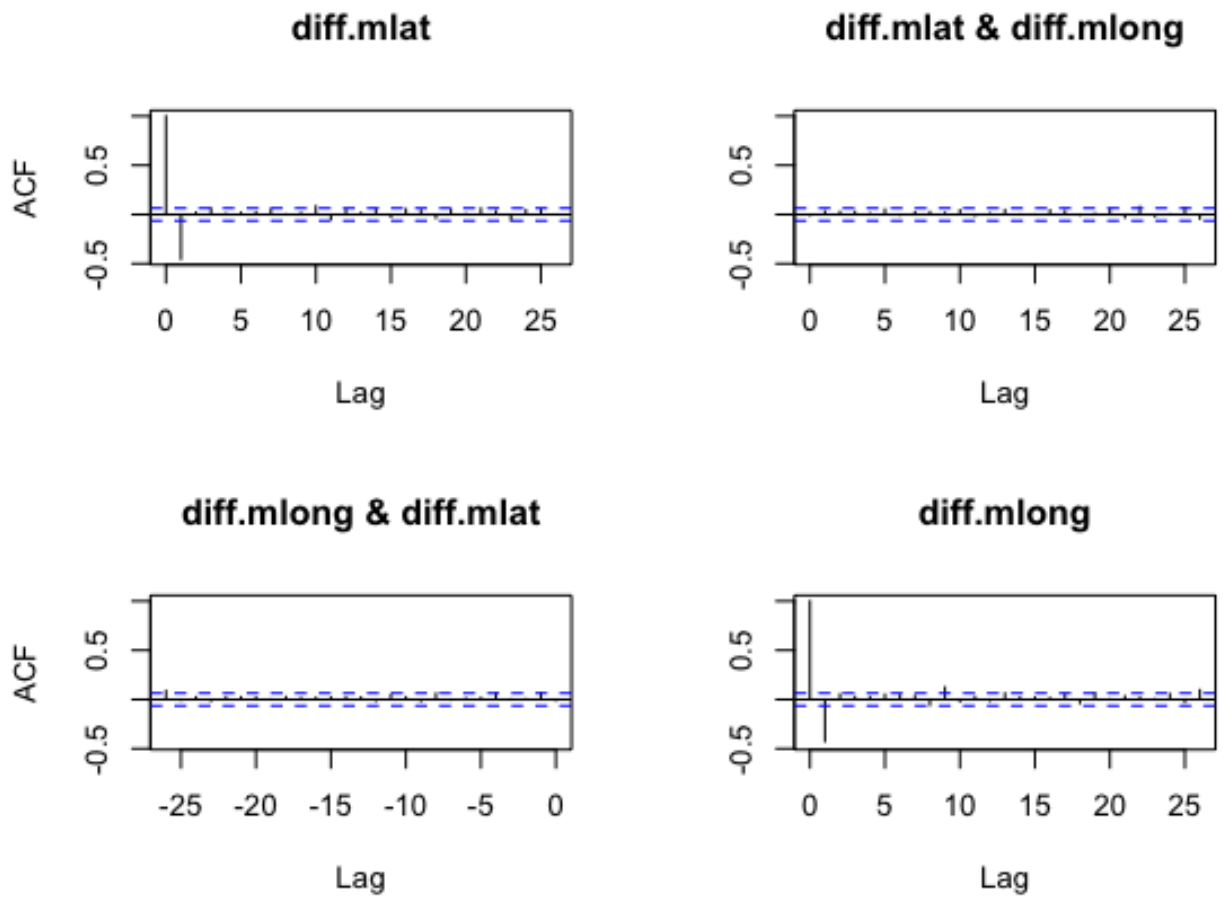

Figure 6: VELOCITY: ACF and CCF after differencing the time series

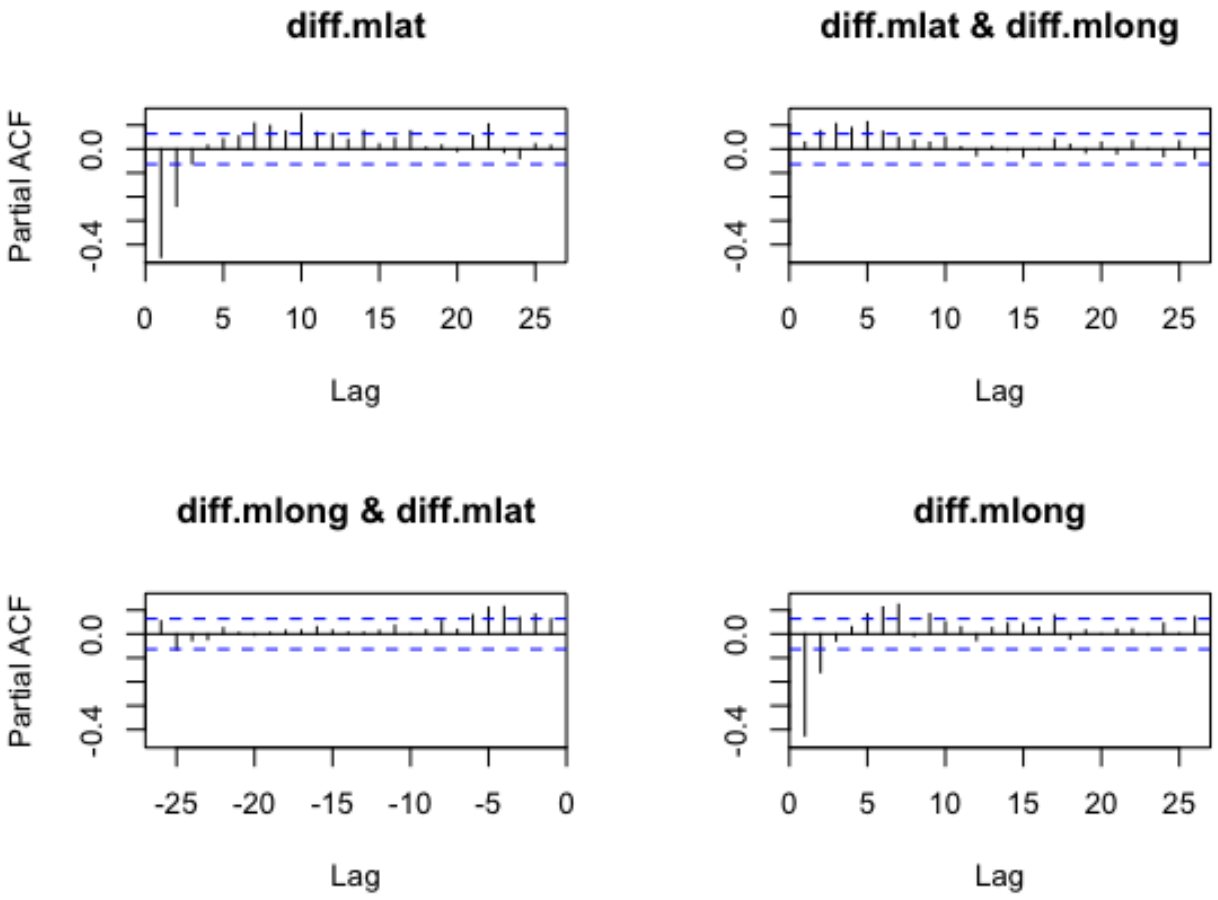

Figure 7: VELOCITY: PACF and PCCF after differencing the time series

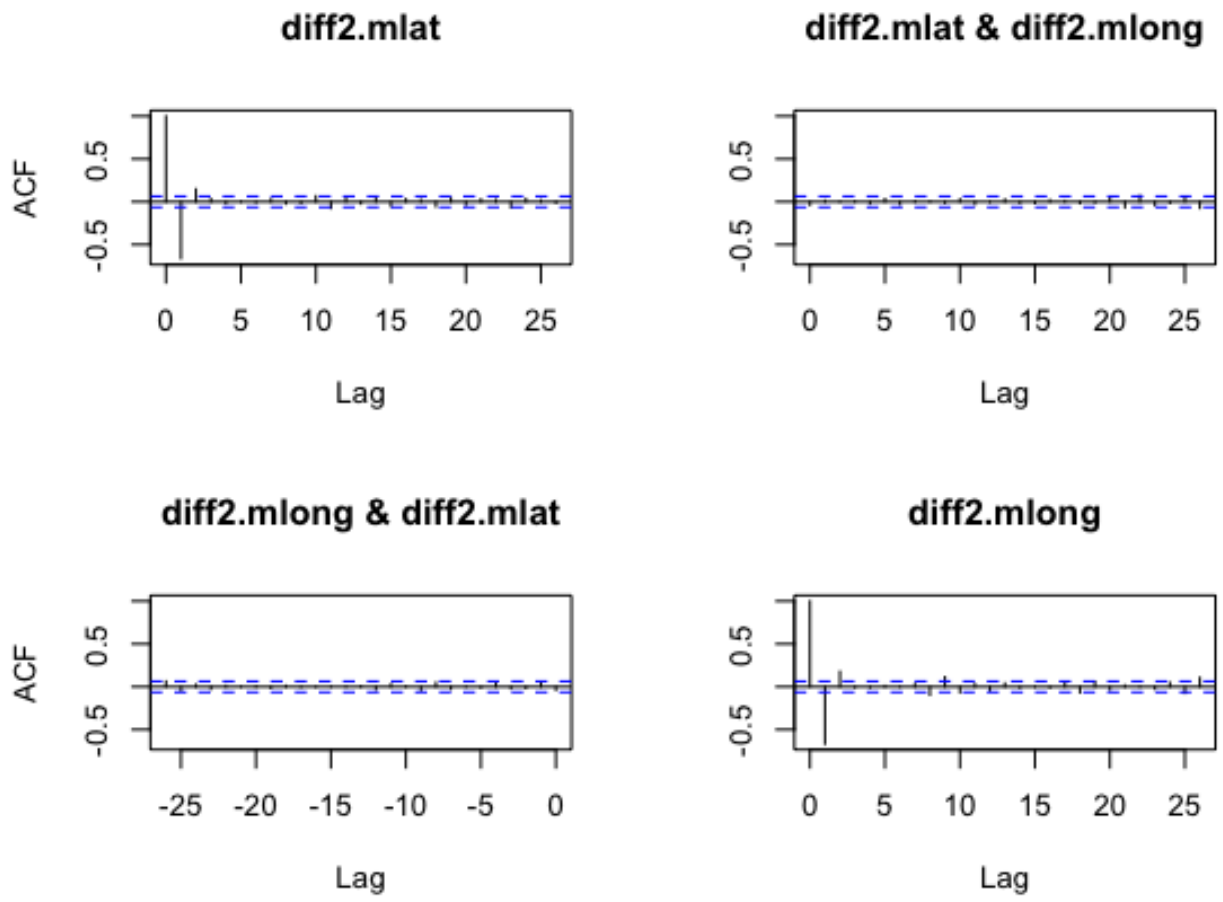

Figure 8: ACCELERATION: ACF and CCF after differencing the time series twice

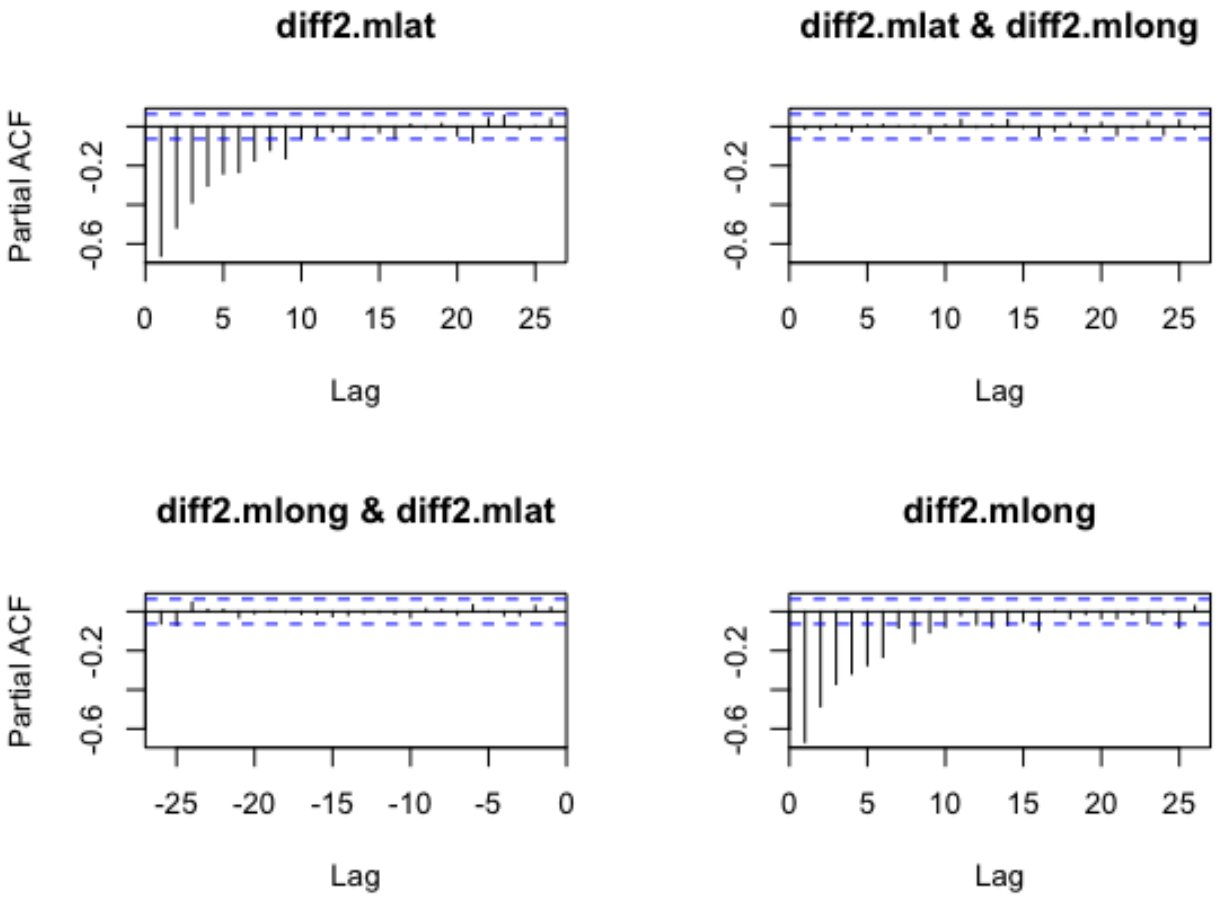

Figure 9: ACCELERATION: PACF and PCCF after differencing the time series twice

3 MARIMA Model

3.1 Model Selection Procedure

We followed the following procedure for model selection.

1. Analyze ACF/CCF and PACF/PCCF plots to infer potential AR and MA processes.
2. Update initial model to reflect ACF/PACF analysis. Compare residuals of both models to see if there was improvement (using QQ plot, cp correlogram, etc) as well as the F-values of the AR and MA components to see if they're significant.
3. Do further model validation by looking at the residuals, Ljung-Box test, and test for normality as well as look at ACF and PACF plots of residuals
4. Use the penalty parameter in marima() function to reduce the model, and check whether reduction was of value. Lastly, do a residual analysis fo the last model.

Given the analysis in the previous section, we opted for choosing an initial MARIMA model of (0,1,1). While the residuals' ACF and PACF of the initial model did not quite capture the spikes under the confidence bounds, we deemed it as a good starting point. We increased the orders of the AR and MA components in the model to obtain a better result.

We ended up with a MARIMA (1,1,2) model. While many other models of higher order were attempted, the performance as based from the ACF and PACF of their residuals as well as the F-values of their coefficients resulted in either marginal or no gains. Specifically, we noticed that increasing the order beyond this resulted in AR or MA parameter coefficients with low F-values and thus we deemed them insignificant below a certain threshold (F-values ¡ 20 in this case). Therefore, we opted for the final model of: MARIMA (1,1,2). Figure 10 shows the the ACF/CCF of the residuals of the model. While it is not perfect, we do see that the lags are out of the confidence bounds by a small margin. Similarly, figure 11 shows the PACF/PCCF and we can see that these tend to spike further out of the confidence bounds. Figure 12 shows that the model converged under 10 iterations.

We reduced the model using the penalty option (we set it to 2 and 3.84) in the marima() function as follows:

```
#Reducing MARIMA Model 3
Marima3.reduced <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model3$ar.pattern, ma.pattern=Model3$ma.pattern, Check=
                      FALSE, Plot="log.det",  penalty=2.0)
```

The f-values of the model are:

```
AR f-values (squared t-values): Lag=1
   x1=y1    x2=y2
y1 310.3038  10.3898
y2  15.2963 241.5915

MA f-values (squared t-values): Lag=2
   x1=y1 x2=y2
y1 52.03 16.90
y2 14.46 25.04
```

As can be seen most coefficients have high F-values. While there are some coefficients (AR(y1-x2) and MA(y2-x1)) that could potentially be insignificant, they are not drastically off so we decide to include them in the model.

The final model's parameters are:

```
AR definition:
, , Lag=1

   x1=y1 x2=y2
y1     1     1
y2     1     1

MA definition:
, , Lag=2

   x1=y1 x2=y2
y1     1     1
y2     1     1

AR estimates:
, , Lag=1

     x1=y1  x2=y2
y1  0.5749 -0.101
y2 -0.1328  0.507


MA estimates:
, , Lag=2

     x1=y1   x2=y2
y1 -0.2752  0.1461
y2  0.1510 -0.1851
```

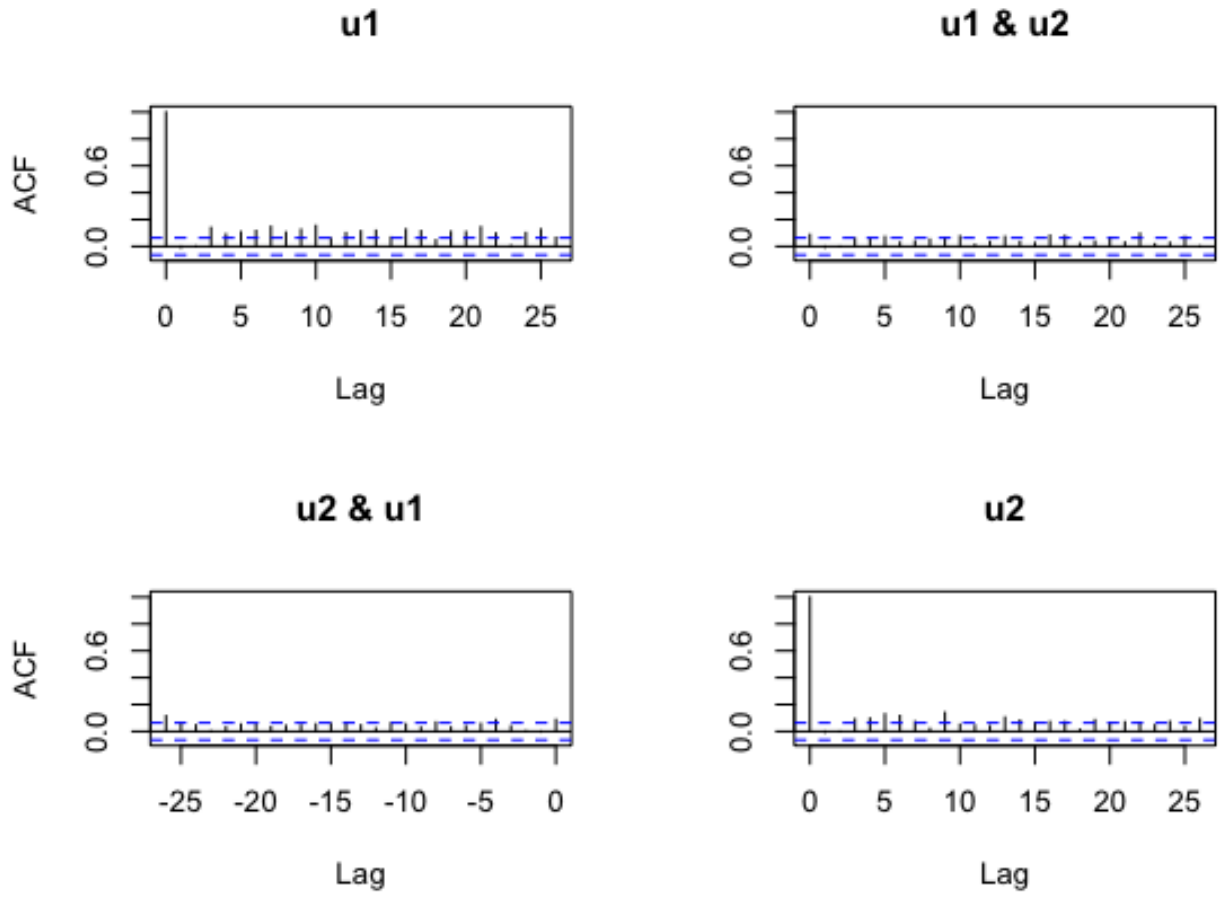

Figure 10: ACF and CCF of chosen model MARIMA(0,1,2)

4 Model Validation

Figures 10 and 11 show the ACF and PACF of the residuals of our final reduced model. As can be seen the lags don't quite perfectly stay within the confidence bounds but they remain near the upper bound in the ACF. The PACF is a bit more troublesome. Figure 13 shows the QQ plot of the residuals and it can be seen that they follow quite well the theoretical quantiles of a normal distribution. However, when we perform a CP correlogram test, we see that while for the most part they fit within the confines of the 95% confidence intervals, at lower frequencies they become out of bounds. This might potentially be due to certain turns that were tracked by the GPS. In essence, this might have caused the residuals during the turning sampling intervals to differ from the rest of the sampling space.

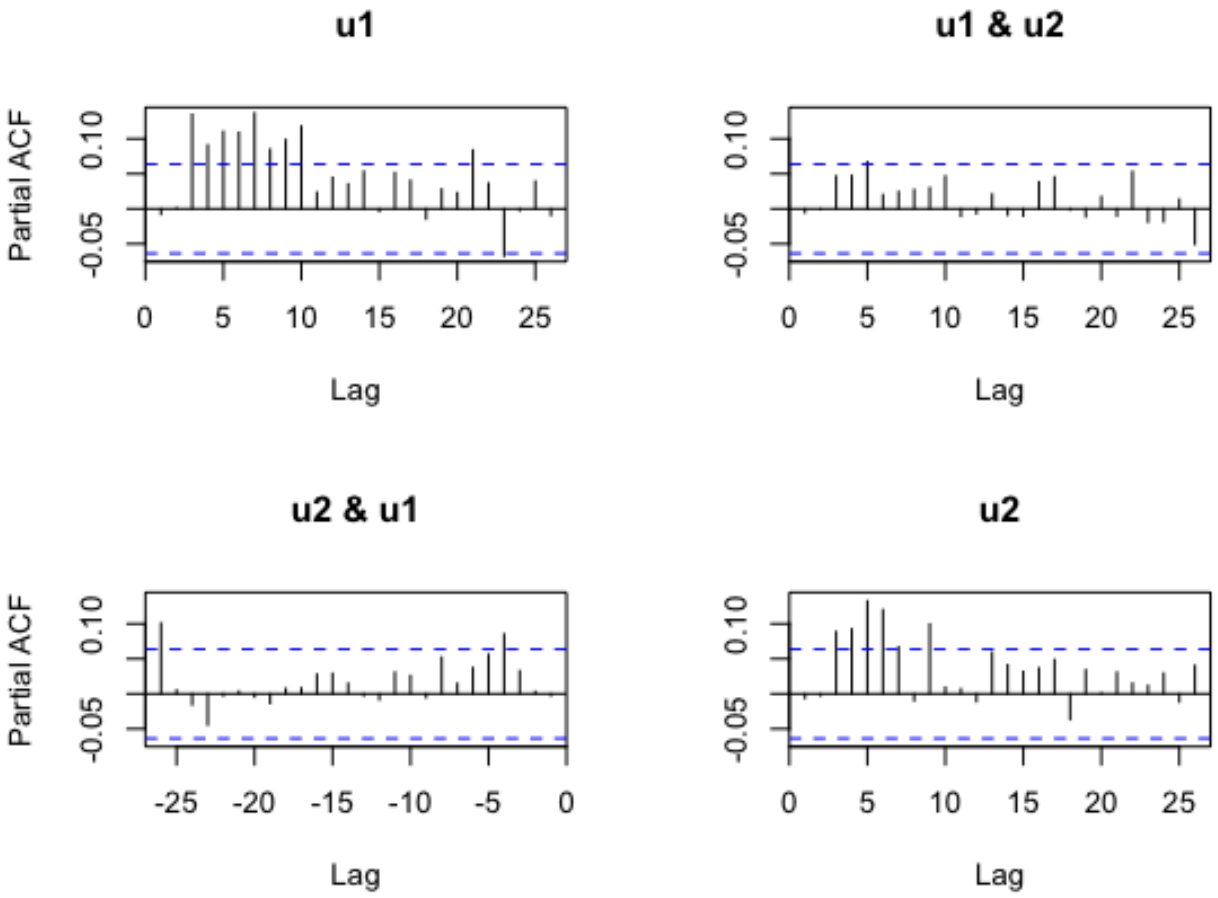

Figure 11: PACF and PCCF of chosen model MARIMA(0,1,2)

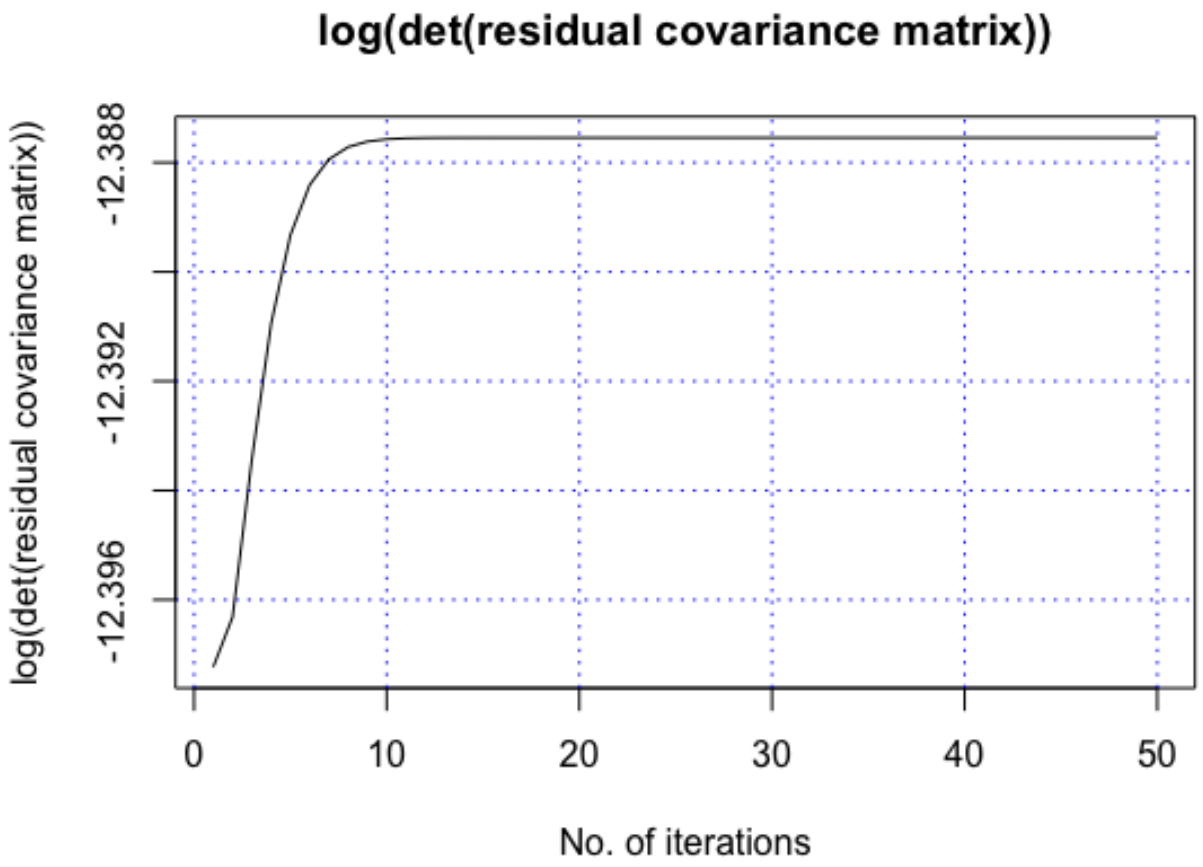

Figure 12: MARIMA response of model (0,1,2)

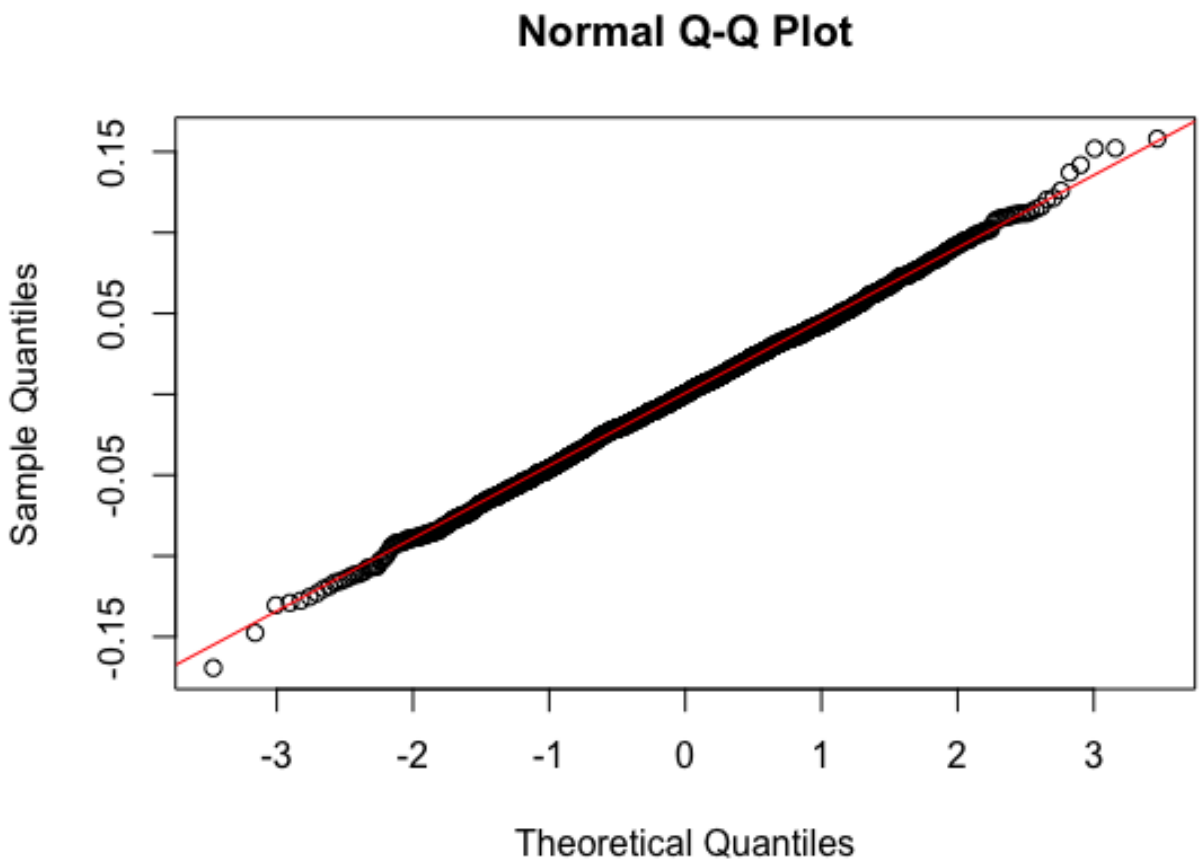

Figure 13: QQ Plot of final reduced MARIMA model

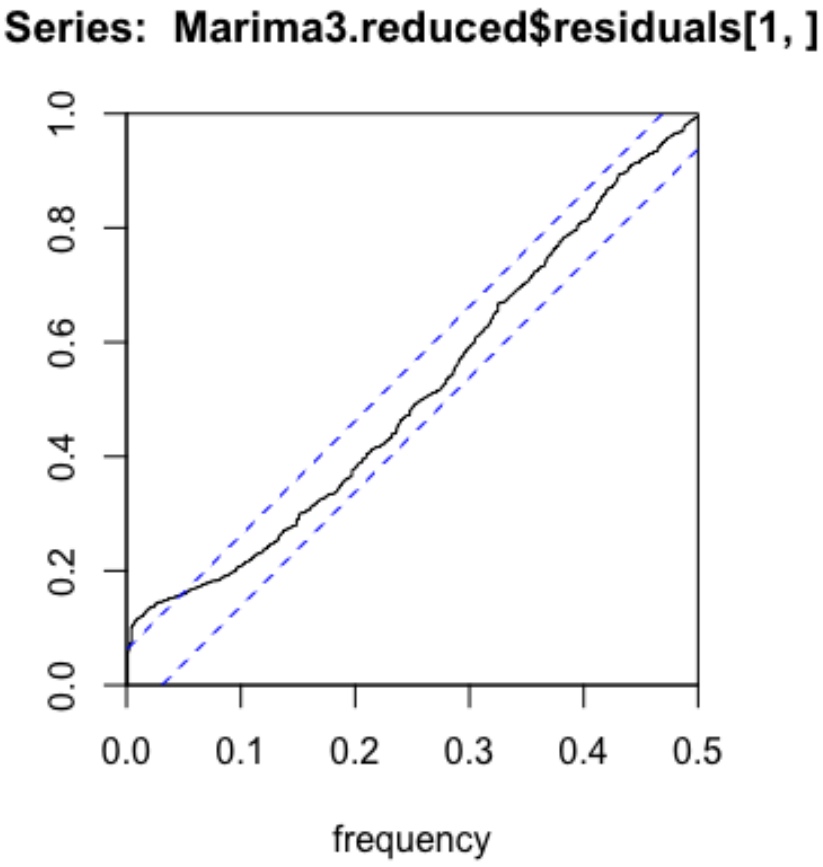

Figure 14: CP correlogram of mlatitude

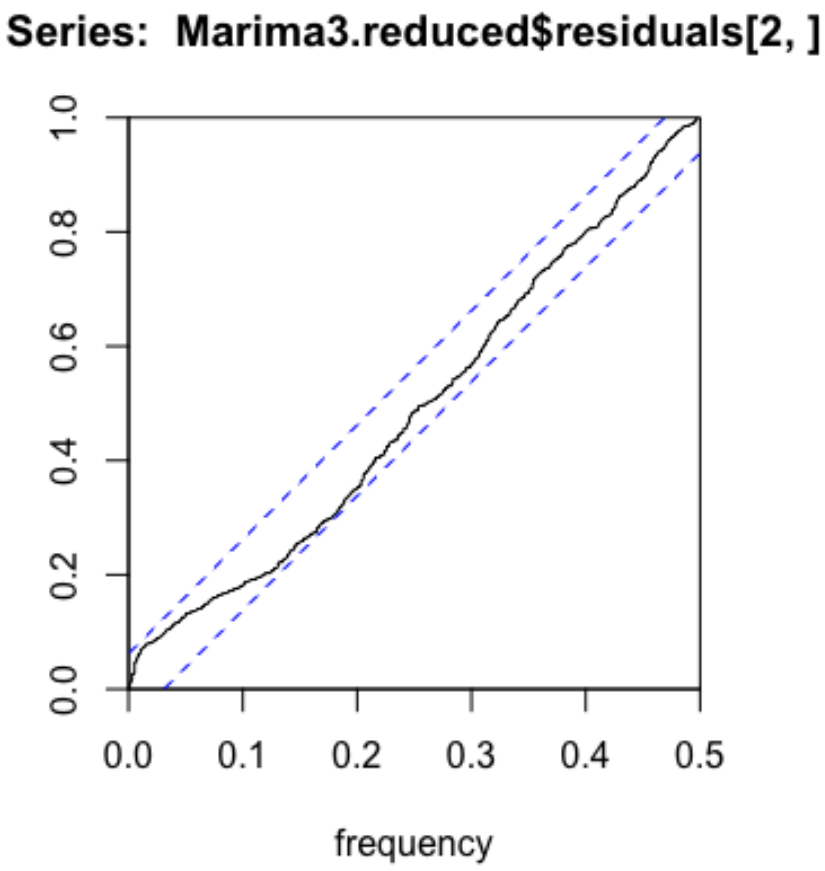

Figure 15: CP correlogram of mlongitude

5 Predicting with MARIMA Model

As shown in Figures 16 and 17, the marima predictions are quite good for mlatitude, but not for mlongitude. While it is difficult to see the long term trend from the x-axis in these figures, the figures in part 1 (Presenting the Data) show the general direction of movement for both mlatitude and mlongitude. In these, it can be seen that in mlatitude the last 50 observations follow the long term direction of the last 650 observations or so. In this case we get good predictions and the observations (test data) are just in between the predicted and the lower 95% lower bound. However, in the mlongitude we can see in figure 2 that the long term direction does not match the last 100 observations or so. That is, there is a turn at around observation 885, which potentially caused the prediction of the marima model to be comletely off as shown in figure 17. The marima model was not able to predict this change in direction.

Below we list the marima predictions:

Time	Prediction (mlat)	Observed	Sigma
1	-4.809413	-4.86796	0.04437938
10	-4.918044	-4.937415	0.07950671
25	-5.09518	-5.185779	0.1169469
50	-5.390408	-5.715376	0.1610412

Time	Prediction (mlong)	Observed	Sigma
1	2.987167	3.029237	0.04619017
10	3.134363	2.890708	0.09217591
25	3.385505	2.772647	0.1387794
50	3.804074	2.718388	0.1928691

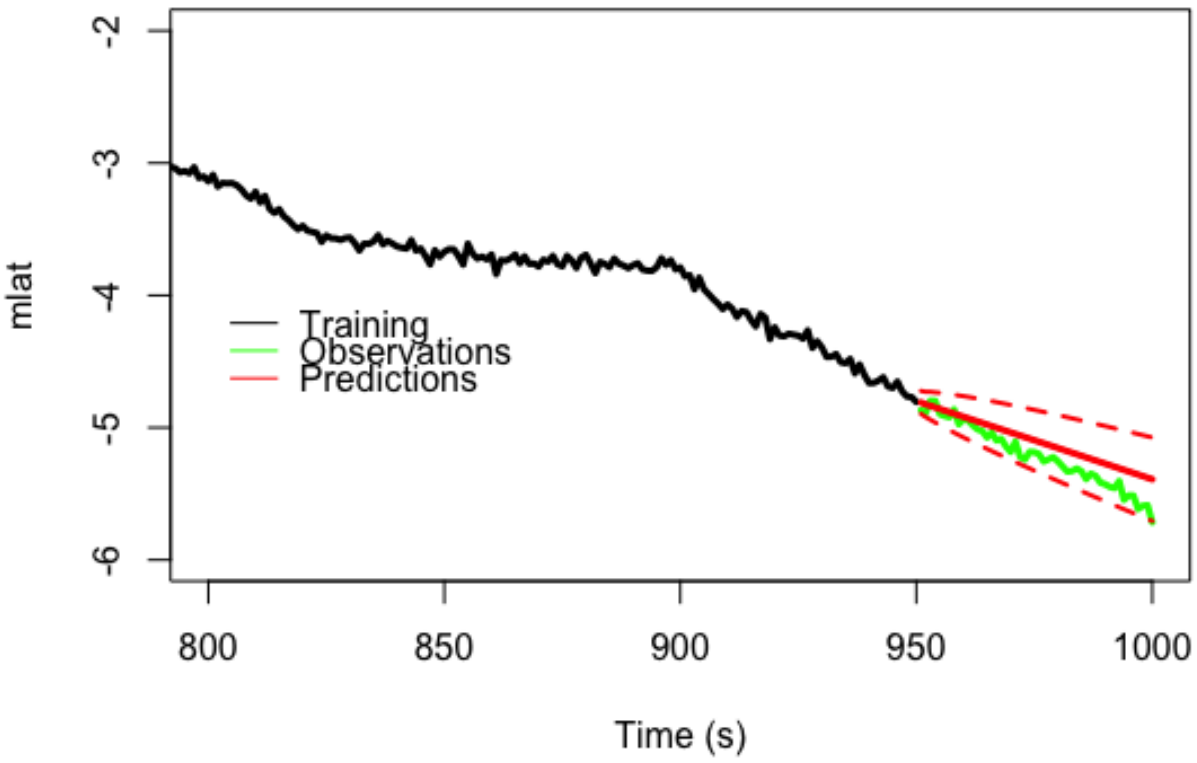

Figure 16: MARIMA prediction of mlatitude with 95% Confidence Intervals

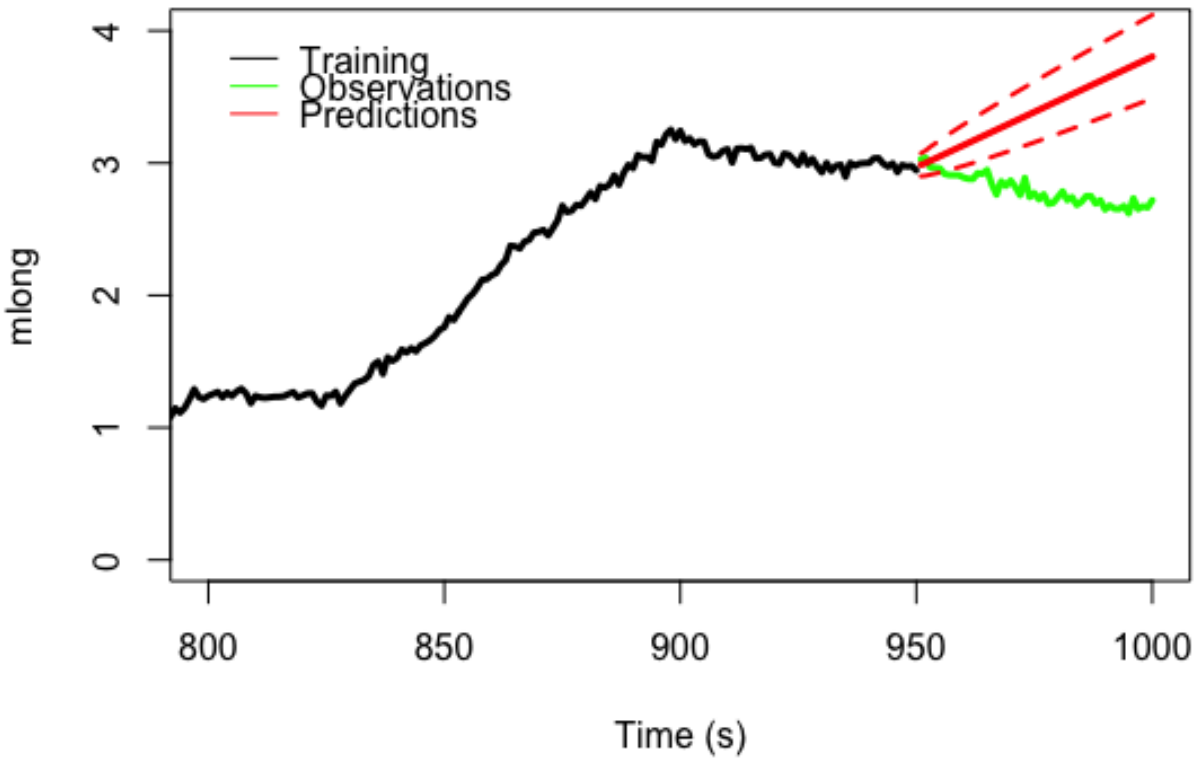

Figure 17: MARIMA prediction of mlongitude with 95% Confidence Intervals

6 Kalman Filter Formulation

We define our State Vector as:

$$X_t = \begin{bmatrix} x_t & v_{xt} & y_t & v_{yt} \end{bmatrix}$$

Our Observation Model becomes:

$$Y_t = \begin{bmatrix} mlat & mlong \end{bmatrix}$$

$$mlat = x_t + e_{1t} \tag{1}$$

$$mlong = y_t + e_{2t} \tag{2}$$

We formulate our A matrix as follows:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

And we formulate the C matrix as follows:

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

Notice that since we don't have an additional input we do not require the B and the u matrix. Finally, our State-Space Model becomes:

$$X_t = AXt - 1 + \epsilon_{1t} \tag{3}$$

$$Y_t = CXt + \epsilon_{2t} \tag{4}$$

Lastly, we formulate our Sigma matrices as follows:

$$\Sigma_1 = \begin{bmatrix} 0.0003 & 0 & 0 & 0 \\ 0 & 10^{-5} & 0 & 0 \\ 0 & 0 & 0.0003 & 0 \\ 0 & 0 & 0 & 10^{-5} \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 10^{-4} \end{bmatrix}$$

7 Kalman Filter Implementation

In this section we show and describe the code used to implement the Kalman Filter.

First, we need to initialize some variables following the same structure as the previous sections formulation.

```
# Kalman Input
Y   = data.train[,2:3]
A   = matrix( c(1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1), ncol=4)
C   = matrix( c(1,0,0,0,0,1,0,0), ncol=4)
S1  = diag(c(0.0003, 1e-5, 0.0003, 1e-5))
S2  = diag(c(1e-4,1e-4))
mu0 = c(data.train[1,2], 0, data.train[1,3], 0)
v0  = S1
steps = 50
```

Then, we show a description of the input and output variables with their dimensions and what they represent:

```
#
# INPUT
#   Y : nxp matrix witn n samples (time periods) by p observations (i.e. mlat and
    mlong) per sampling interval
#   A : sxs matrix with s being number of components in sate vector [xt vxt yt vyt
    ]
#   C : pxs matrix with p observations (i.e. mlat and mlong) by s components from
    state vector [xt vxt yt vyt]
#   S1: sxs diagonal variance matrix of system      model error
#   S2: pxp diagonal variance matrix of observation model error
#   mu0: initial estimate of state-space model at time=1 given X0.
#        an sx1 vector. s is number of states in state vector [xt vxt yt vyt]
#   v0 : initial covariance of Sxx at time=1 given Sxx0 (a 2x2 matrix)
#   steps : number of predictions after last observation in Y (a scalar)
#
#
# OUTPUT
#   REC:  list of reconstruced variables: X, Sxx, Syy, K
#   PRED: list of predicted    variables: X, Sxx, Syy
#
```

We initialize the variables according to slide 12 in Lecture 11 as follows:

```
# INITIALIZATION
Xhat <- mu0
Sxx  <- v0
Syy  <- C %*% Sxx %*% t(C) + S2
```

Also, in order to keep track of dimensions we need to set some variables that will help us in those regards:

```
N = dim(Y)[1]       #number of observations
 nstates = dim(A)[1] #number of states
```

Finally, we iterate from 1 to 1000. Notice that the reconstruction computations are inside an if statement that allow it to only be computed when within the number of observations available (i.e. training data). The predictions are outside of that if statement and hence can compute the predictions from 950 to 1000 (i.e. test data set).

```
  for (i in 1:(N+steps)) {

    if (i<N+1) {
      # RECONSTRUCTION
      K     <- Sxx %*% t(C) %*% solve(Syy)             #kalman gain
      Xhat <- Xhat + K%*%t(Y[i,]) - K%*%C%*%Xhat      #Xhat update, using kalman gain
           (reduces noise)
      Sxx   <- Sxx - K%*%Syy%*%t(K)                    #Sxx  update

      #storing
      K.all[,,i]     <- K
      X.rec[i,]      <- Xhat
      Sxx.rec[,,i]   <- Sxx
      Syy.rec[,,i]   <- Syy
    }#endif

    # PREDICTION
    Xhat <- A%*%Xhat
    Sxx   <- A%*%Sxx%*%t(A) + S1
    Syy   <- C%*%Sxx%*%t(C) + S2

    #storing
    X.pred[i,]       <- Xhat
    Sxx.pred[,,i]    <- Sxx
    Syy.pred[,,i]    <- Syy

  }#endfor

  # OUTPUT
  output   <- list(X.rec=X.rec,  Sxx.rec=Sxx.rec,  Syy.rec=Syy.rec,  K=K.all, X.pred=
      X.pred, Sxx.pred=Sxx.pred, Syy.pred=Syy.pred)
  return(output)
```

The powerful thing about the Kalman filter is that it is able to reduce noise by computing the kalman gain (variable K in the code). The kalman gain determines the influence the error will have in the next update of $Xhat$ (the state). It does this by taking into account the estimate of the current noise given the noise of the previous iteration (for both the errors of the State and the Observation models).

8 Using Kalman Filter for Reconstruction and Prediction

Below we list the kalman predictions along with the uncertainty:

Time	Prediction (mlat)	Observed	Sigma
1	-4.829242	-4.86796	0.04090321
10	-5.003964	-4.937415	0.1343099
25	-5.295168	-5.185779	0.3411213
50	-5.780507	-5.715376	0.8005788

Time	Prediction (mlong)	Observed	Sigma
1	2.940497	3.029237	0.04090321
10	2.893918	2.890708	0.1343099
25	2.816288	2.772647	0.3411213
50	2.686903	2.718388	0.8005788

As seen, the kalman predictions are much better. They follow the same direction as the observations. Had they observations taken a turn anywhere betweent t = 950-1000, then the kalman filter might have not followed the same direction since it would not have an actual (real) update of an observation from previous iterations. However, in this case there was no change in direction and the kalman filter fared much better than the marima model. Notice however, that the uncertainty as described by their standar deviations are much larger for the kalman filter than for the marima model. The farther the kalman filter predicts into the future the larger the uncertainty in the predictions becomes. This is shown in tables 1 to 4.

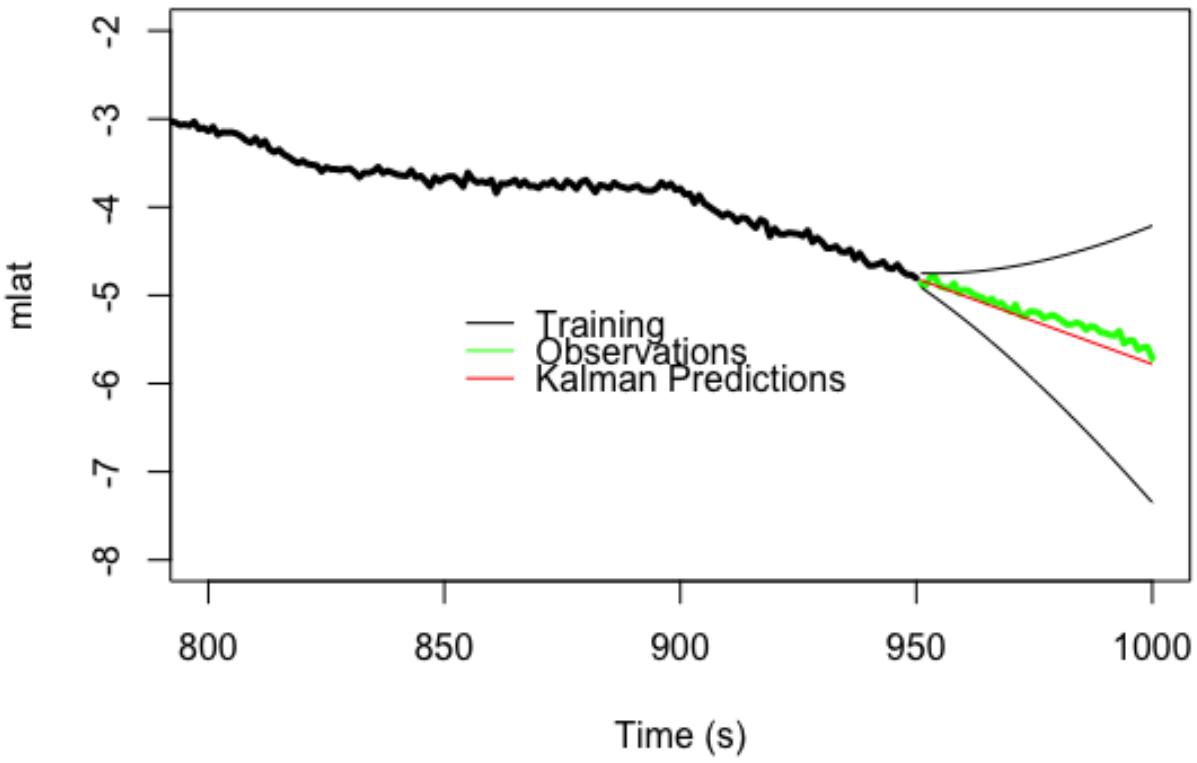

Figure 18: Kalman predictions of mlatitude

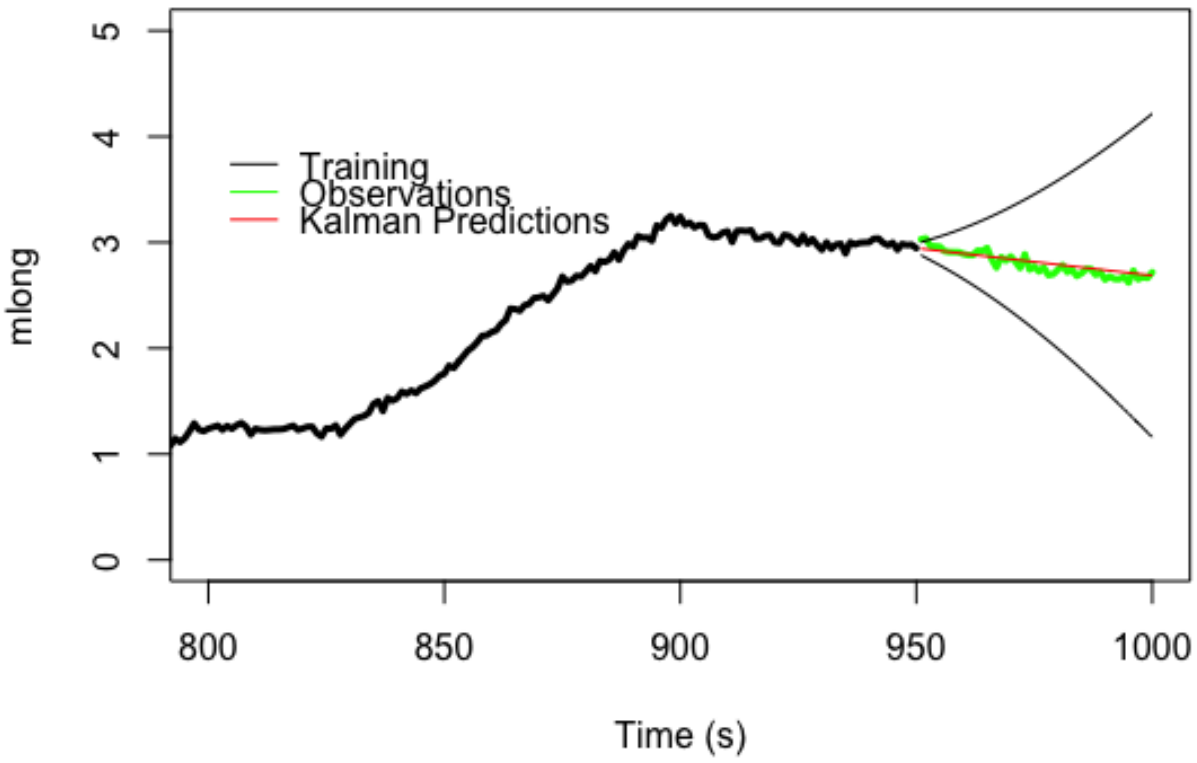

Figure 19: Kalman predictions of mlongitude

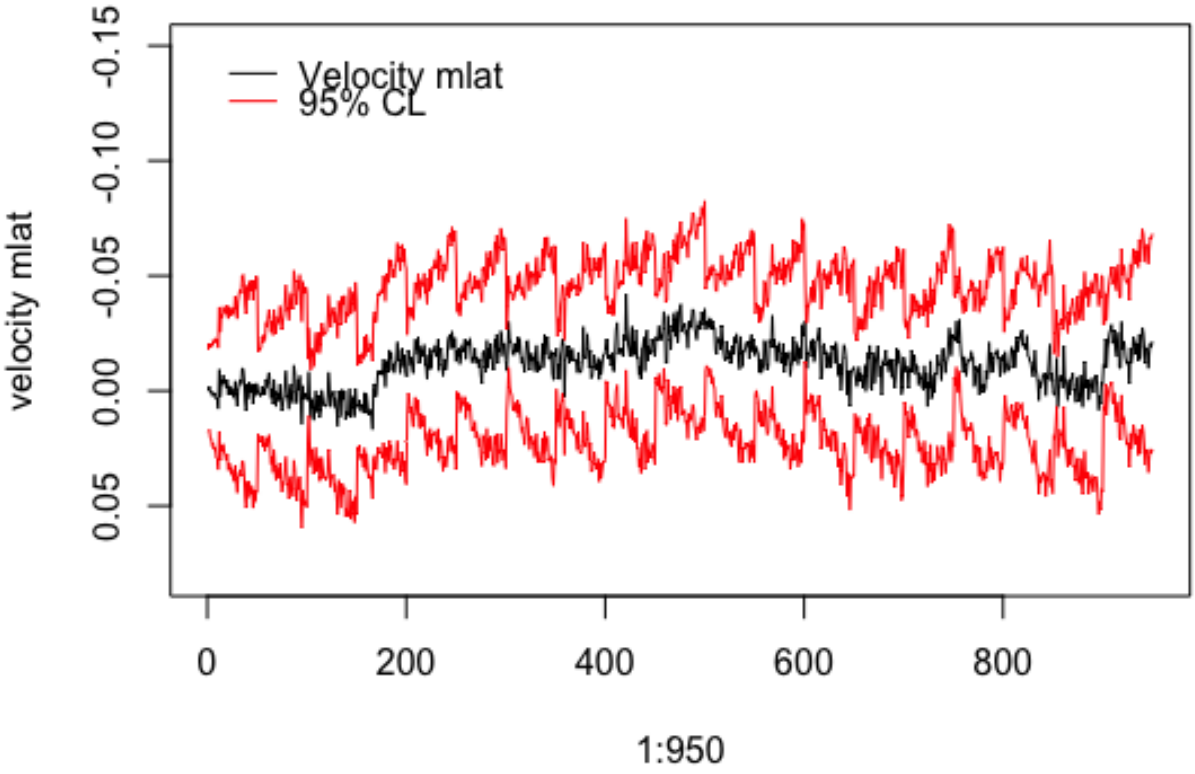

Figure 20: Kalman predictions of velocity mlatitude

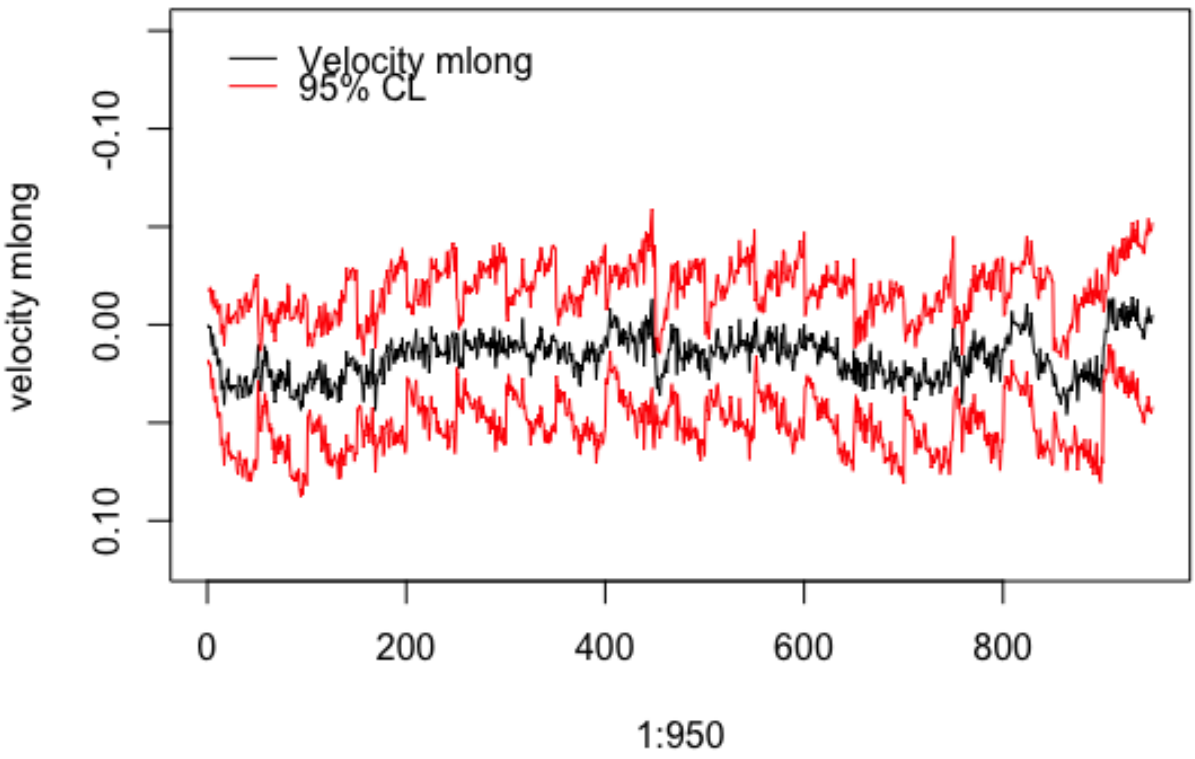

Figure 21: Kalman predictions of velocity mlongitude

9 MARIMA and Kalman Filter Comparison

As mentioned in the previous section, the marima model did not perform well when predicting mlongitude. This is possibly due the change in direction which can be seen if Figure 2. In contrast, there was no turn in mlatitude and the marima model performed adequately. In comparison to the kalman filter however, the kalman filter performed better in point prediction estimates. It was able to more closely follow the true observations. However, a drawback of the kalman filter is that the uncertainty is wider than in marima, and it grows the farther away it gets from the last observations it had available in the model. In terms of improving the kalman filter implementation, an additional input to the state-space model could improve the noise reduction by utilizing the B and the u matrices. By having an extra variable that has partial information about the state system, the kalman filter might be able to extract more certainty from the inputs and provide better estimates of the state system.

10 Kalman Filter Code

```
kalman_me <- function(Y,A,C,S1,S2,mu0,v0,steps){

  ######################## FUNCTION DESCRIPTION
     ##########################################################################
  #
  # INPUT
  #    Y : nxp matrix witn n samples (time periods) by p observations (i.e. mlat and
     mlong) per sampling interval
  #    A : sxs matrix with s being number of components in sate vector [xt vxt yt vyt
     ]
  #    C : pxs matrix with p observations (i.e. mlat and mlong) by s components from
     state vector [xt vxt yt vyt]
  #    S1: sxs diagonal variance matrix of system      model error
  #    S2: pxp diagonal variance matrix of observation model error
  #    mu0: initial estimate of state-space model at time=1 given X0.
  #         an sx1 vector. s is number of states in state vector [xt vxt yt vyt]
  #    v0 : initial covariance of Sxx at time=1 given Sxx0 (a 2x2 matrix)
  #    steps : number of predictions after last observation in Y (a scalar)
  #
  #
  # OUTPUT
  #    REC:  list of reconstruced variables: X, Sxx, Syy, K
  #    PRED: list of predicted    variables: X, Sxx, Syy
  #
  #
     ##########################################################################################
  # INITIALIZATION
  Xhat <- mu0
  Sxx  <- v0
  Syy  <- C %*% Sxx %*% t(C) + S2

  N = dim(Y)[1]       #number of observations
  nstates = dim(A)[1] #number of states

  # STORAGE
  #(For storing variables per iteration)
  X.rec    <- array(dim=c(N+steps,nstates)) #A single X is a row vector
```

```
  X.pred   <- array(dim=c(N+steps,nstates))
  Sxx.rec  <- array(dim=c(dim(Sxx), N+steps))
  Sxx.pred <- array(dim=c(dim(Sxx), N+steps))
  Syy.rec  <- array(dim=c(dim(Sxx), N+steps))
  Syy.pred <- array(dim=c(dim(Sxx), N+steps))
  K.all    <- array(dim=c(dim(Sxx %*% t(C) %*% solve(Syy)), N+steps))



  for (i in 1:(N+steps)) {

    if (i<N+1) {
      # RECONSTRUCTION
      K    <- Sxx %*% t(C) %*% solve(Syy)            #kalman gain
      Xhat <- Xhat + K%*%t(Y[i,]) - K%*%C%*%Xhat     #Xhat update, using kalman gain
          (reduces noise)
      Sxx  <- Sxx - K%*%Syy%*%t(K)                   #Sxx  update

      #storing
      K.all[,,i]    <- K
      X.rec[i,]     <- Xhat
      Sxx.rec[,,i]  <- Sxx
      Syy.rec[,,i]  <- Syy
    }#endif

    # PREDICTION
    Xhat <- A%*%Xhat
    Sxx  <- A%*%Sxx%*%t(A) + S1
    Syy  <- C%*%Sxx%*%t(C) + S2

    #storing
    X.pred[i,]      <- Xhat
    Sxx.pred[,,i]   <- Sxx
    Syy.pred[,,i]   <- Syy

  }#endfor

  # OUTPUT
  output  <- list(X.rec=X.rec,  Sxx.rec=Sxx.rec,  Syy.rec=Syy.rec,  K=K.all, X.pred=
      X.pred, Sxx.pred=Sxx.pred, Syy.pred=Syy.pred)
  return(output)

}#endfunction
```

11 All Code

```
# TSA Assignment 4 #

############################### Script Setup & Data
    ##################################################################

rm(list=ls())
setwd("~/Desktop/TSA/Assignment4")
library(expm)
library(timeSeries)
library(tseries)
library(forecast)
library(lmtest)
library(PBSmodelling)
library(marima)
```

```
#Data Partitioning
data <- read.table("A4_gps_log.csv",header = T,sep = ",")
data.train <- data[1:950,]
data.test  <- data[951:1000,]

################################  PLOTTING
    ###########################################################################

# Plot entire original series
dev.off()
par(mfrow=c(2,1))
#mlat
plot(data.train[,1], data.train[,2], ylab='mlat', xlab ="Time (s)", type = "l", xlim
    =c(0,1000), ylim=c(-6,7), lwd=3)
lines(data.test[,1], data.test[,2], col = "green", lwd=3)
legend(0,0, c("Training", "Test (for predictions)"), col = c("black", "green"), lty
    = c(1,1), bty ="n")

#mlong
plot(data.train[,1], data.train[,3], ylab='mlong', xlab ="Time (s)", type = "l",
    xlim=c(0,1000), ylim=c(-15,4), lwd=3)
lines(data.test[,1], data.test[,3], col = "green", lwd=3)
legend(0,0, c("Training", "Test (for predictions)"), col = c("black", "green"), lty
    = c(1,1), bty ="n")

#mlat and mlong
plot(data.train[,3], data.train[,2], ylab='mlat', xlab ="mlong", type = "l", ylim=c
    (-6,7), xlim=c(-15,4), lwd=3)
lines(data.test[,3], data.test[,2], col = "green", lwd=3)
legend(-15,4, c("Training", "Test (for predictions)"), col = c("black", "green"),
    lty = c(1,1), bty ="n")
dev.off()

#CrossCorrelation Plots
acf(data.train[,-1])
pacf(data.train[,-1])

#Difference Data      (1st difference)
diff.mlat <- diff(data.train[,2])
diff.mlong <- diff(data.train[,3])
diff.data <- ts(cbind(diff.mlat, diff.mlong))

#CrossCorrelation Plots of Differenced Data
acf(diff.data)
pacf(diff.data)

#Difference Data AGAIN (2nd difference)
diff2.mlat <- diff(diff(data.train[,2]))
diff2.mlong <- diff(diff(data.train[,3]))
diff2.data <- ts(cbind(diff2.mlat, diff2.mlong))

#CrossCorrelation Plots of Differenced Data
acf(diff2.data)
pacf(diff2.data)

################################ Fitting MARIMA Models
    ###############################################################

# Differencing data for marima analysis
```

```
diff_data  <- define.dif(data.train[,2:3], difference=c(1,1,2,1))
diff2_data <- define.dif(data.train[,2:3], difference=c(1,2,2,2))

#Fitting MARIMA Model 0 (0,2,3)
ar<-c(1)
ma<-c(2)
# Define the proper model:
Model0 <- define.model(kvar=2, ar=ar, ma=ma, indep = NULL)
# Difference the data
dif.y <-define.dif(t(data.train[,2:3]), c(1,2,2,2))
# Now call marima:
Marima0 <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model0$ar.pattern, ma.pattern=Model0$ma.pattern, Check=
                      FALSE, Plot="log.det", penalty=0.0)
# print AR and MA estimates
Marima0
# MA f-values are quite high; what does that mean?
#MA f-values (squared t-values): Lag=3
#     x1=y1  x2=y2
# y1  33.56 41.58
# y2  30.15 16.08

## Looking at acf and ccf of residuals
acf(t(Marima0$residuals))
pacf(t(Marima0$residuals))


#Fitting MARIMA Model 1 (0,1,1)
ar<-c(0)
ma<-c(1)
# Define the proper model:
Model1 <- define.model(kvar=2, ar=ar, ma=ma, indep = NULL)
# Difference the data
dif.y <-define.dif(t(data.train[,2:3]), c(1,1,2,1))
# Now call marima:
Marima1 <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model1$ar.pattern, ma.pattern=Model1$ma.pattern, Check=
                      FALSE, Plot="log.det", penalty=0.0)
# print AR and MA estimates
Marima1
# MA f-values are quite high; what does that mean?
#MA f-values (squared t-values): Lag=1
#     x1=y1  x2=y2
# y1 301.45  10.34
# y2  16.24 237.77

## Looking at acf and ccf of residuals
acf(t(Marima1$residuals))
pacf(t(Marima1$residuals))


#Fitting MARIMA Model 2
ar<-c(1)
ma<-c(1)
# Define the proper model:
Model2 <- define.model(kvar=2, ar=ar, ma=ma)
# Difference the data
dif.y <-define.dif(t(data.train[,2:3]), c(1,1,2,1))
# Now call marima:
Marima2 <- marima(dif.y$y.dif,means=1,
```

```
                  ar.pattern=Model2$ar.pattern, ma.pattern=Model2$ma.pattern, Check=
                      FALSE, Plot="log.det", penalty=0.0)
# print AR and MA estimates
Marima2
#AR f-values (squared t-values):Lag=1
#      x1=y1  x2=y2
# y1 4.2230 1.5879
# y2 0.4252 5.8607
#
#MA f-values (squared t-values): Lag=1
#      x1=y1 x2=y2
# y1 48.44  6.38
# y2  6.03 23.65
## Looking at acf and ccf of residuals
acf(t(Marima2$residuals[,-1]))
pacf(t(Marima2$residuals[,-1]))


#Fitting MARIMA Model 3
ar<-c(1)
ma<-c(2)
# Define the proper model:
Model3 <- define.model(kvar=2, ar=ar, ma=ma)
# Difference the data
difference = matrix(c(1,1,2,1), nrow=2)   #difference = c(1,1,2,1)
dif.y <-define.dif(t(data.train[,2:3]), difference=difference)
# Now call marima:
Marima3 <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model3$ar.pattern, ma.pattern=Model3$ma.pattern, Check=
                      FALSE, Plot="log.det", penalty=0.0)
# print AR and MA estimates
Marima3

# AR f-values (squared t-values): Lag=1
#       x1=y1    x2=y2
# y1 310.3038  10.3898
# y2  15.2963 241.5915

# MA f-values (squared t-values):Lag=2
#    x1=y1 x2=y2
# y1 52.03 16.90
# y2 14.46 25.04
## Looking at acf and ccf of residuals
acf(t(Marima3$residuals[,-1]))
pacf(t(Marima3$residuals[,-1]))


#Reducing MARIMA Model 3
Marima3.reduced <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model3$ar.pattern, ma.pattern=Model3$ma.pattern, Check=
                      FALSE, Plot="log.det",  penalty=2.0)
# print AR and MA estimates
Marima3.reduced

Marima3.reduced <- marima(dif.y$y.dif,means=1,
                          ar.pattern=Model3$ar.pattern, ma.pattern=Model3$ma.pattern
                              , Check=FALSE, Plot="log.det",  penalty=3.84)
# AR f-values (squared t-values): Lag=1
#     x1=y1    x2=y2
# y1 310.3038  10.3898
```

```
# y2  15.2963 241.5915

# MA f-values (squared t-values): Lag=2
#    x1=y1 x2=y2
# y1 52.03 16.90
# y2 14.46 25.04
## Looking at acf and ccf of residuals
acf(t(Marima3.reduced$residuals[,-1]))
pacf(t(Marima3.reduced$residuals[,-1]))


#Fitting MARIMA Model 4
ar<-c(1)
ma<-c(3)
# Define the proper model:
Model4 <- define.model(kvar=2, ar=ar, ma=ma)
# Difference the data
difference = c(1,1,2,1)
dif.y <-define.dif(t(data.train[,2:3]), difference=difference)
# Now call marima:
Marima4 <- marima(dif.y$y.dif,means=1,
                  ar.pattern=Model4$ar.pattern, ma.pattern=Model4$ma.pattern, Check=
                     FALSE, Plot="log.det", penalty=0.0)
# print AR and MA estimates
Marima4

# AR f-values (squared t-values): Lag=1
#    x1=y1    x2=y2
# y1 252.2008   0.7261
# y2   3.8175 213.6531

# MA f-values (squared t-values): Lag=3
#    x1=y1 x2=y2
# y1  9.89  2.64
# y2  0.64  4.42
acf(t(Marima4$residuals[,-1]))
pacf(t(Marima4$residuals[,-1]))



############################## MODEL VALIDATION
    ##################################################################

ks.test(Marima3.reduced$residuals[1,], y="pnorm", alternative = c("two.sided"))
ks.test(Marima3.reduced$residuals[2,], y="pnorm", alternative = c("two.sided"))


# QQ Plot
#Both variates
qqnorm(Marima3.reduced$residuals)
qqline(Marima3.reduced$residuals,col=2)

#Mlat
qqnorm(Marima3$residuals[1,])
qqline(Marima3$residuals,col=2)
#Mlong
qqnorm(Marima3$residuals[2,])
qqline(Marima3$residuals,col=2)

# Cumulative Periodogram  PLot
```

```
#     - do residuals look like white noise?

cpgram(Marima3.reduced$residuals[1,]) # not quite!
cpgram(Marima3.reduced$residuals[2,]) # not quite!



############################### PREDICTIONS
    ################################################################

## Predicting 50 years ahead
pred.data <- as.matrix(cbind(t(data.train[,2:3]), matrix(NA,nrow=2, ncol=50)))
pred <-  arma.forecast(pred.data, nstart=ncol(t(data.train[,2:3])), nstep=50, marima
    =Marima3.reduced, check=TRUE, dif.poly = dif.y$dif.poly)


#first, difference validation set
mlat.diff.test <- diff(data.test[,2])
mlong.diff.test <- diff(data.test[,3])
data.test.diff <- ts(cbind(mlat.diff.test, mlong.diff.test))



# Plot entire original series + forecasts
dev.off()
par(mfrow=c(3,1))
#mlat
plot(data.train[,1], data.train[,2], ylab='mlat', xlab ="Time (s)", type = "l", xlim
    =c(800,1000), ylim=c(-6,-2), lwd=3)
lines(data.test[,1], data.test[,2], col = "green", lwd=3)
lines(data.test[,1], pred$forecasts[1,951:1000], col = "red", lwd=3)
pred.int <- pred$forecasts[1,951:1000] + cbind(rep(0, 50), -1, 1)*qnorm(0.975)*sqrt(
    pred$pred.var[1,1,])
matlines(951:1000, pred.int, lty=c(1,2,2), col=2, lwd=2 )
legend(800,-4, c("Training", "Observations","Predictions"), col = c("black", "green"
    , "red"), lty = c(1,1,1), bty ="n")

#mlong
plot(data.train[,1], data.train[,3], ylab='mlong', xlab ="Time (s)", type = "l",
    xlim=c(800,1000), ylim=c(0,4), lwd=3)
lines(data.test[,1], data.test[,3], col = "green", lwd=3)
lines(data.test[,1], pred$forecasts[2,951:1000], col = "red", lwd=3)
pred.int <- pred$forecasts[2,951:1000] + cbind(rep(0, 50), -1, 1)*qnorm(0.975)*sqrt(
    pred$pred.var[1,1,])
matlines(951:1000, pred.int, lty=c(1,2,2), col=2, lwd=2 )
legend(800,4, c("Training", "Observations", "Predictions"), col = c("black", "green"
    , "red"), lty = c(1,1,1), bty ="n")



############################### KALMAN FILTER
    ################################################################################

# Kalman Input
Y   = data.train[,2:3]
A   = matrix( c(1,0,0,0,1,1,0,0,0,0,1,0,0,0,1,1), ncol=4)
C   = matrix( c(1,0,0,0,0,1,0,0), ncol=4)
S1  = diag(c(0.0003, 1e-5, 0.0003, 1e-5))
S2  = diag(c(1e-4,1e-4))
mu0 = c(data.train[1,2], 0, data.train[1,3], 0)
```

```
v0  = S1
steps = 50

# Kalman Implementation
k.output1 <- kalman_me( Y=Y, A=A, C=C, S1=S1, S2=S2, mu0=mu0, v0=v0, steps=steps)

# Plotting Kalman Predictions
plot(data.train[,1], data.train[,2], ylab='mlat', xlab ="Time (s)", type = "l", xlim
    =c(800,1000), ylim=c(-8,-2), lwd=3)
lines(data.test[,1], data.test[,2], col = "green", lwd=3)
lines(data.test[,1], k.output1$X.pred[951:1000,1], col="red")
cl  <- k.output1$X.pred[951:1000,1] + qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [1,1,951:1000])
cl2 <-  k.output1$X.pred[951:1000,1] - qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [1,1,951:1000])
lines(data.test[,1],cl)
lines(data.test[,1], cl2)
legend(850,-5, c("Training", "Observations", "Kalman Predictions"), col = c("black",
     "green","red"), lty = c(1,1,1), bty ="n")

plot(data.train[,1], data.train[,3], ylab='mlong', xlab ="Time (s)", type = "l",
    xlim=c(800,1000), ylim=c(0,5), lwd=3)
lines(data.test[,1], data.test[,3], col = "green", lwd=3)
lines(data.test[,1], k.output1$X.pred[951:1000,3], col="red")
cl  <- k.output1$X.pred[951:1000,3] + qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [1,1,951:1000])
cl2 <-  k.output1$X.pred[951:1000,3] - qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [1,1,951:1000])
lines(data.test[,1],cl)
lines(data.test[,1], cl2)
legend(800,4, c("Training", "Observations", "Kalman Predictions"), col = c("black",
    "green","red"), lty = c(1,1,1), bty ="n")

plot(1:950, k.output1$X.rec[1:950,2], type="l", ylab="velocity mlat", ylim=c(0.08,
    -0.15))
cl  <- k.output1$X.rec[1:950,2] + qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [2,2,951:1000])
cl2 <-  k.output1$X.rec[1:950,2] - qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [2,2,951:1000])
lines(1:950,cl, col="red")
lines(1:950, cl2, col="red")
legend(0,-0.15, c("Velocity mlat", "95% CL"), col = c("black", "red"), lty = c(1,1),
     bty ="n")

plot(1:950, k.output1$X.rec[1:950,4], type="l", ylab="velocity mlong", ylim=c(0.12,
    -0.15))
cl  <- k.output1$X.rec[1:950,4] + qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [4,4,951:1000])
cl2 <-  k.output1$X.rec[1:950,4] - qnorm(0.975)*sqrt(k.output1$Sxx.pred
    [4,4,951:1000])
lines(1:950,cl, col="red")
lines(1:950, cl2, col="red")
legend(0,-0.15, c("Velocity mlong", "95% CL"), col = c("black", "red"), lty = c(1,1)
    , bty ="n")
```