Names: Ibrahim, Basil; Zaman, Saad

Instructor: Yulia Gel

Course: STAT930

Semester: Spring 2009

Results from Modelling Toronto Temperature

Prediction

July 20: 22.33403
Interval at 95% CI: [19.02834, 25.70585]

Methodology

Preliminary Steps:

Step 1: Remove leap days (29 Feb 2000, 2004, 2008)

Step 2: Create column called t (starting from 1)

Step 3: Calculate standard deviation of each day of the year, relative to the same year in all subsequent years (e.g.**SDV =stdev(Jan1 1999,Jan1 2000,…,Jan 1 2009)**)

Step 4: Divide temperature of each day with standard deviation → Standardizing temperatures

Observations of Data:

- ➔ Looking at the actual data, the same day across the years can have a large temperature interval (the largest was 27, the average was 13.3).
- ➔ Winters have more volatile temperatures than summers.

Best Model:

1) Taking the original temperature **temp**.

2) Fitting a sinusoidal component **S=A1+B1cos(wt)+C1sin(wt)**, with **w=2π/365**.

3) Taking residuals from S, **resid11=temp – S**, and dividing them by their standard deviation. **resid11/SDV**

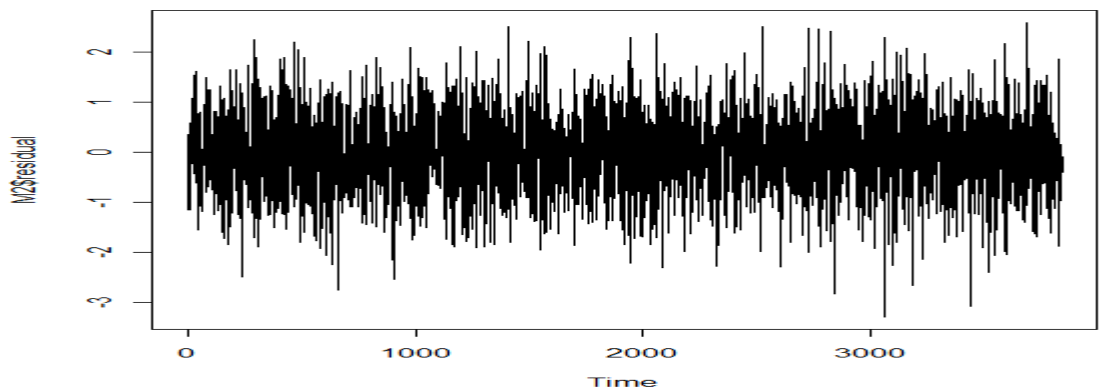
4) Fitting an **ARMA(4,1)** model (Used a spectrum fit and found AR(4) is best, then found adding an MA(1) component has a lower AIC value).

arima0(x = resid11, order = c(4, 0, 1))

Coefficients:
ar1 ar2 ar3 ar4 ma1 intercept

1.6881 -0.9111 0.2477 -0.0489 -0.9065 0
s.e. 0.0160 0.0389 0.0312 0.0173 0.0741 NaN

sigma^2 estimated as 0.5491: log likelihood = -4294.66, aic = 8603.33
Warning message:
In sqrt(diag(x$var.coef)) : NaNs produced



Diagnostic Tests

Error terms do not seem to follow a normal distribution. Therefore, for obtaining prediction intervals, we used a bootstrapping method (500 iterations).

Testing Our Predictions

Point Prediction {14 Day Prediction from Jan 15th to July6th}:

Sum of errors (Actual-Predicted): -57.86658→ Overestimate temperature more often
Absolute sum of errors: 600.0487
Worst difference: 12.50583
Best difference: 0.03122916
Mean deviation (in absolute value): 3.46849
Standard deviation of deviation (in absolute value): 2.687968

Interval Prediction {14 Day Prediction from June 15th to June24th}:

Due to high computational complexity of bootstrapping, we were only able to bootstrap (500 iterations) 10 days. The results are very positive. 9 out 10 had the actual temperatures lay within the predicted 95% confidence intervals.

Names: Ibrahim, Basil; Zaman, Saad Course: STAT930

Instructor: Yulia Gel Semester: Spring 2009

Another Candidate Model Originally Considered:

1) Taking the standardized temperature **stdzd_temp**.

2) Differencing it seasonally (subtract every day by its previous year).

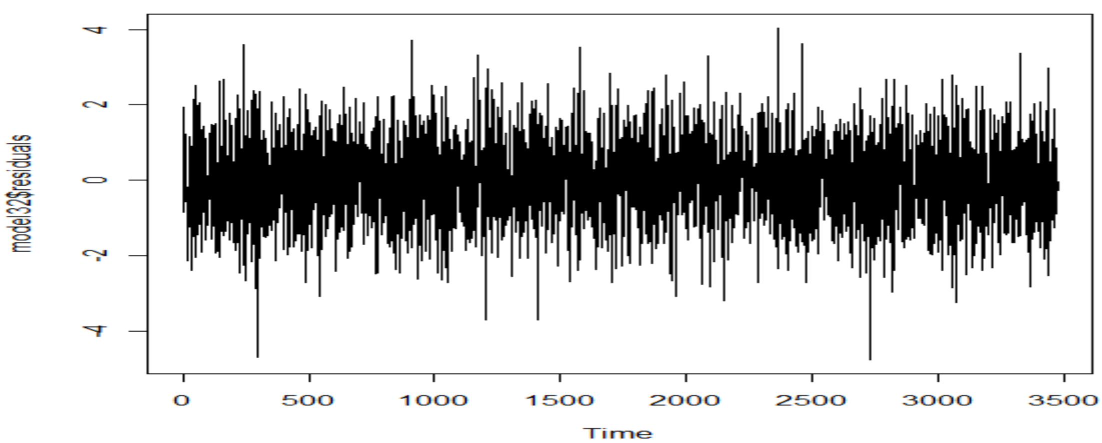
3) Fitting an **ARMA(6,1)** model.

arima0(x = diff_stdzd_temp, order = c(6, 0, 1))
Coefficients:

	ar1	ar2	ar3	ar4	ar5	ar6	ma1	intercept
	1.6567	-0.8795	0.2591	-0.0506	-0.0556	0.0433	-0.8847	-0.0423
s.e.	0.0170	0.0445	0.0367	0.0362	0.0329	0.0177	0.1823	0.0754

sigma^2 estimated as 1.051: log likelihood = -5012.54, aic = 10043.08

Residuals look like this:



Diagnostic Tests

1) Unit root test:

Augmented Dickey-Fuller Test

data: model32$residual
Dickey-Fuller = -14.7332, Lag order = 15, p-value = 0.01
alternative hypothesis: stationary

No unit roots

Names: Ibrahim, Basil; Zaman, Saad

Course: STAT930

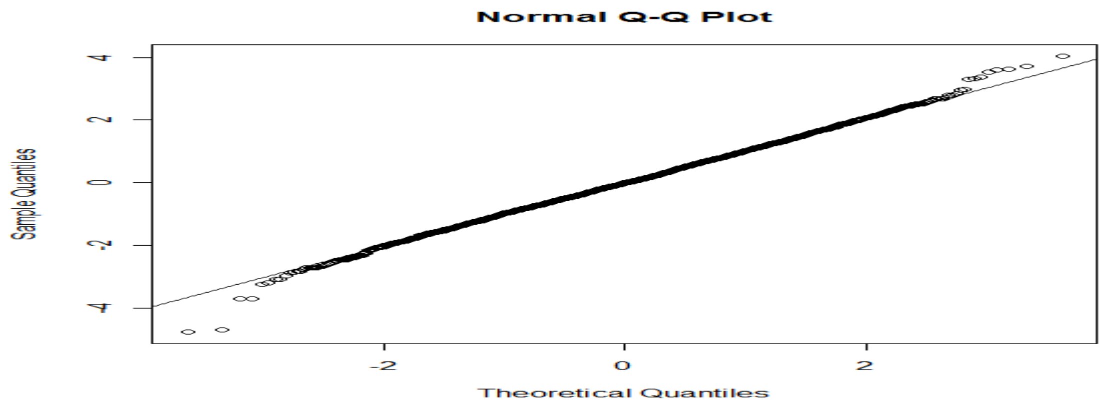
Instructor: Yulia Gel

Semester: Spring 2009

2) a) Normality (without Removing Outliers)

Shapiro Wilk: Poor fit, probably due to outliers

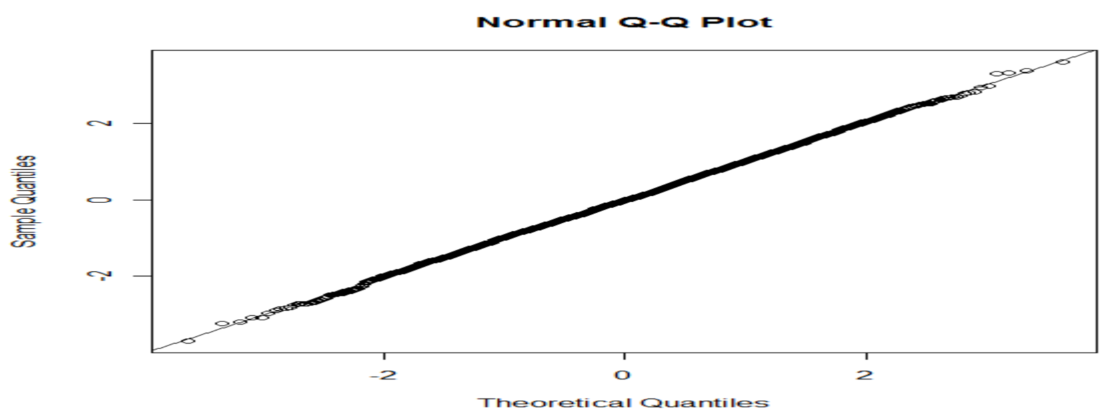
Shapiro-Wilk: p-value = 0.002409



2) b) Normality (Removing 7 outliers)

Shapiro-Wilk: Very high fit → Error terms seem to be normal

Shapiro-Wilk: p-value = 0.89



3) Heteroskedasticity Test:

Levene (Data divided into 3 groups: 50 in group 1, 60 in group 2, the rest in group 3):
Using means and group centres: p-value = 0.9474 → Do not reject homoskedastic hypothesis
Using medians as group centres: p-value=0.9466 → Do not reject homoskedastic hypothesis

→Error terms seem to be homoskedastic

Names: Ibrahim, Basil; Zaman, Saad | Course: STAT930

Instructor: Yulia Gel | Semester: Spring 2009

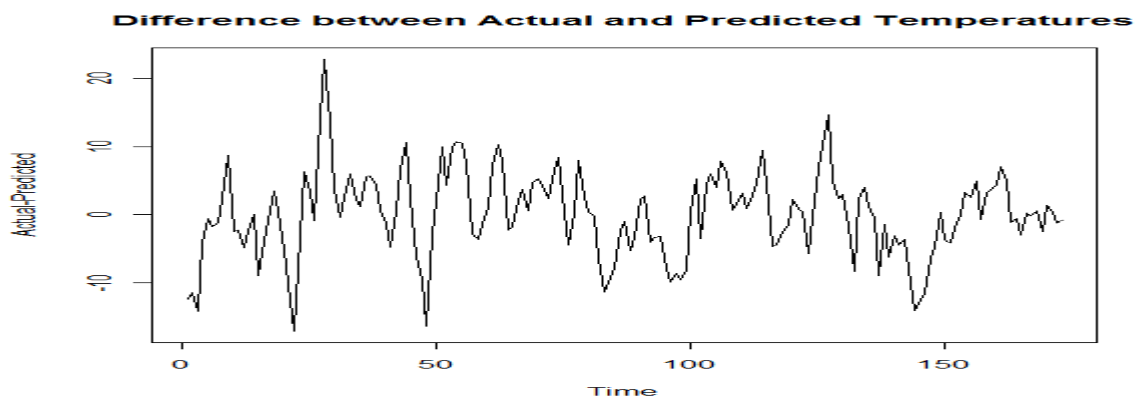
Prediction

July 20: 19.75824
Interval at 95% CI: [14.54769, 24.96879] (greater range than our chosen model)

Testing Our Predictions {14 Day Prediction from Jan 1st to June 22nd }

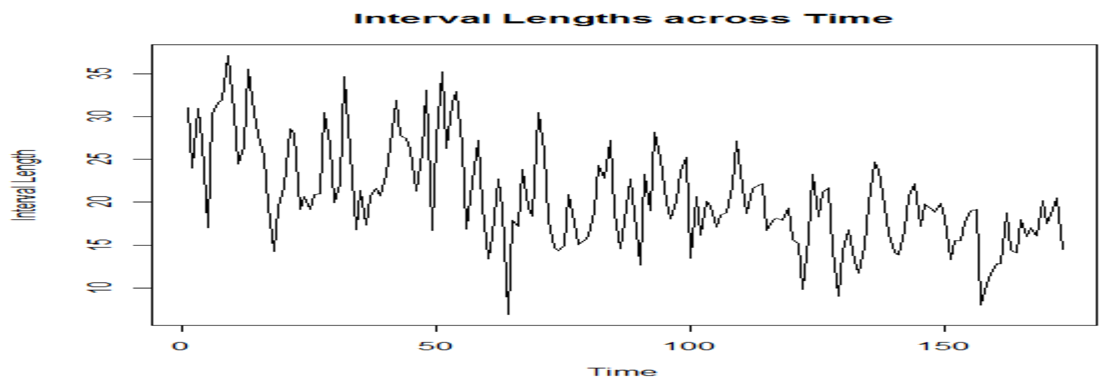
Point Prediction:

Sum of errors (Actual-Predicted):7.865395 → Underestimate temperature more often
Absolute sum of errors: 824.0057 (greater than our chosen model)
Worst difference: 22.90181 (greater than first model)
Best difference: Nearly zero
Mean deviation (in absolute value): 4.763039
Standard deviation of deviation (in absolute value): 4.050973(greater than first model)



Interval Prediction:

Since the error terms follow normal, we can calculate the intervals by:
PredictedJuly20±1.96*(SE(July20))^0.5
Percentage of actual temperature being in estimated intervals: 92.48555% (due to large intervals)
Largest interval: 37.14708 (very large)
Smallest interval: 6.89344
Mean interval: 20.63781
Standard deviation of interval: 5.816204



Concluding Remarks on Why Our First Model is Better

- ➔ First model has a lower AIC than second model implying a better goodness of fit
- ➔ Although the second model's residuals seem to be iid normal, making bootstrapping an option rather than a necessity (as was the case in the first model), the prediction intervals are very large relative to the first model as opposed to the second model.
- ➔ Although the first model overestimates the temperatures more often than the second model underestimates it, it is more often closer to the actual temperatures.

Names: Ibrahim, Basil; Zaman, Saad Course: STAT930

Instructor: Yulia Gel Semester: Spring 2009

R-Code:

```
#Time Series Project
#Note: We have removed all leap days from Excel file for convenience purposes
library(RODBC) #This is for opening and reading from the Excel file
chan=odbcConnectExcel("TempData")#Reading from the Excel Sheet "TempData"
data=sqlFetch(chan,"VAR")#Putting what is in Sheet VAR into data
close(chan)

t=data$t[1:(length(data$t)-1)] #time (Excluding leap days)
temp=data$Temp[1:(length(data$Temp)-1)] #Mean Temperature
SDV=data$SDV[1:(length(data$SDV)-1)]#Standard deviation of each day
stdzd_temp=temp/SDV #Standardized Temperature


#1) Removing Sinusoidal Component
w=2*pi/365
x1=lm(temp~cos(w*t)+sin(w*t))#Regressing using sinusoidal components
A1=x1$coefficients[1]#Intercept
B1=x1$coefficients[2]#Coefficient of coswt
C1=x1$coefficients[3]#Coefficient of sinwt
temp_hat1=A1+B1*cos(w*t)+C1*sin(w*t)#Estimate using sinusoidal components
resid11=temp-temp_hat1#Removing sinusoidal component

#2)Destandardizing the Data
resid11=resid11/SDV#Dividing the residual from the standard deviation of each day

#3) Modelling ARMA(4,0) and Checking Accuracy of Model
M1=arima0(resid11,order=c(4,0,0))
M1
July22=31+28+31+30+31+22#Time index number of July 22nd (year not important)
Dec31=365*10#December 31st 2008 index number
Test=rep(0,July22)#Predicted results of test (will be used to store predictions from Jan1 to Jul22)
diff_tempAR4=rep(0,July22)#Absolute differences
#Predicting from January 15th till July 6th
for(i in 1:July22)
{
 X=resid11[1:(Dec31+i)]#Starting point December 31st

 t_pred=length(X)+14#Time index of prediction (starting point: January 15th 2009)
 X_pred=predict(M1,n.ahead=14)$pred[14]#Get predicted value 14 days ahead

 #Now we need to remove seasonal difference and unstandardize the predicted value
 #a)Unstandardizing
 X_pred=X_pred*SDV[14+i]
 #b)Adding back sinusoidal component
```

```
 X_pred=X_pred+A1+B1*cos(w*(14+i))+C1*sin(w*(14+i))
 Test[i]=X_pred
 diff_tempAR4[i]=temp[Dec31+i+14]-Test[i]#Difference between original and predicted
}
```

#4) Modelling ARMA(4,1) and Checking Accuracy of Model

```
M2=arima0(resid11,order=c(4,0,1))
M2
July22=31+28+31+30+31+22#Time index number of July 22nd (year not important)
Dec31=365*10#December 31st 2008 index number
Test=rep(0,July22)#Predicted results of test (will be used to store predictions from Jan1 to Jul22)
diff_tempAR41=rep(0,July22)#Absolute differences
for(i in 1:July22)
{
 X=resid11[1:(Dec31+i)]#Starting point December 31st
 M2=arima0(resid11,order=c(4,0,1))
 t_pred=length(X)+14#Time index of prediction (starting point: January 15th 2009)
 X_pred=predict(M2,n.ahead=14)$pred[14]#Get predicted value 14 days ahead

 #Now we need to remove seasonal difference and unstandardize the predicted value
 #a)Unstandardizing
 X_pred=X_pred*SDV[14+i]
 #b)Adding back sinusoidal component
 X_pred=X_pred+A1+B1*cos(w*(14+i))+C1*sin(w*(14+i))
 Test[i]=X_pred
 diff_tempAR41[i]=temp[Dec31+i+14]-Test[i]#Difference between original and predicted
}
```

#5) Bootstrapping

#a) Bootstrap Function

```
# The function takes input parameters as following
# x = Data to bootstrap
# k = order of the Max AR model that we want to fit. If k= 0, then it is automatically chosen to be n-1 (i.e.
no effect)
# H = number of step ahead prediction desired. Thus, if  H = 3, method will return a vector of size 3.
Sieve.Boot<-function(X,k,H)
{
   n <- length(X)
   T <- length(X)      # used in Step 8
   X_star_Th <- rep(0,H)    # For forecasting h step ahead.

   if(k==0)    #<- if they don't specify the desired order of AR model desired, then we select the AR model
Based on AIC criterion.
   {
     k = n-1
```

```
  }

# Step 1  - find the AR model and the correct order
  a<-ar(X, aic=TRUE, method="mle")
# Step 2  - Get the Ar coefficients using the Yule-Walker Estimates
  p <- a$order
  psi_hat <- a$ar
  if(p < 1)
  { print("Order is 0 of the model in step 2")
    a <-ar(X, aic=FALSE, order=1, method="mle")
    p <- a$order
    psi_hat <- a$ar
  }
# Step 3  - Compute the residuals
  eps_hat <- rep(0, n-p-1)
  for(t in (p+1):n)
  {
    eps_hat[t-p] <- X[t] - mean(X)     #<- for j = 0
    for(j in 1:p)
    { eps_hat[t-p] = eps_hat[t-p] + psi_hat[j]*(X[t-j] - mean(X))
    }
  }
# Step 4 - Define the Emperical Distribution
  eps_gamma <- sum(eps_hat)/(n-p)
  eps_t <- rep(0,length(eps_hat))
  for(i in 1:length(eps_hat))
  {
    eps_t[i] <- eps_hat[i] - eps_gamma
  }
# Step 5 - Draw a random sample for eps_t
 eps_star <- sample(eps_t, size=n, replace=TRUE)
 future_eps_star <- sample(eps_t, size=H, replace=TRUE)
# Step 6 - Calculate X_star_t by recursion
 X_star <- rep(0,n)
 X_star[1:p] <- mean(X)
 SumVec <- rep(0,p)

 eps_star_star <- rep(0,n)
 for(j in 1:p)
 { SumVec[j] <- psi_hat[j]*(mean(X))
 }
 eps_star_star = eps_star + sum(SumVec)
 X_star = arima.sim(list(order = c(a$order,0,0), ar = a$ar), n = n, innov=eps_star_star)
# Step 7  - Get the Ar coefficients using the Yule-Walker Estimates just like we did in Stpe 2
  a_star<-ar(X_star,aic=TRUE, method="mle")
  p_star <- a_star$order
  psi_hat_star <- a_star$ar
```

```
  if(p_star < 1)
  { print("Order is 0")
   a_star<-ar(X_star, aic=FALSE, order=1, method="mle")
   p_star <- a_star$order
   psi_hat_star <- a_star$ar
  }
 # Step 8 - Compute the Bootstrap Observations by recusion
   for(h in 1:H)
  { X_star_Th[h] = mean(X) + future_eps_star[h]
   for(j in 1:p_star)
   {
    if((T+h-j) <= T)
    {  X_star_Th[h] = X_star_Th[h] - psi_hat_star[j]*(X[T+h-j] - mean(X))
    }else
    {
     X_star_Th[h] = X_star_Th[h] - psi_hat_star[j]*(X_star_Th[h-j] - mean(X))
    }
   }
  }  # i.e. returning h step aheads forecasts in one vector
  return(X_star_Th)
}
```

#b) Prediction Intervals for July 20th using AR(4) and ARMA(4,1)

```
iterations=500#Number of iterations for the bootstrap
Jul20=31+28+31+30+31+30+20#Time index of July 20th (year irrelevant)
Jul20Model11=rep(0,iterations) #Store bootstrap values for AR(4) model
Jul20Model12=rep(0,iterations) # Store bootstrap values for ARMA(4,1) model
for (i in 1: iterations)
{
boot.result1=Sieve.Boot(M1$residual,0,14)
boot.result2=Sieve.Boot(M2$residual,0,14)
Jul20Model11[i]=boot.result1[14]#Get predicted value of July 20th for AR(4)
Jul20Model12[i]=boot.result2[14] #Get predicted value of July 20th for ARMA(4,1)
}
#AR(4)
PointM1=predict(M1,n.ahead=14)$pred[14] #Residual point prediction of AR(4)
LowerM1=PointM1+quantile(Jul20Model11,0.025) #Residual lower bound prediction of AR(4)
UpperM1=PointM1+quantile(Jul20Model11,0.975) #Residual upper bound prediction of AR(4)
#Unstandardizing
PointM1=PointM1*SDV[Jul20] ]
LowerM1=LowerM1*SDV[Jul20]
UpperM1=UpperM1*SDV[Jul20]
#Adding sinusoidal component
PointM1=PointM1+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)
LowerM1=LowerM1+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)
UpperM1=UpperM1+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)
```

```
PointM1#22.5539
LowerM1#19.11617(bootstrap of 500)
UpperM1#26.33619(bootstrap of 500)

#ARMA(4,1) Winner Model
PointM2=predict(M2,n.ahead=14)$pred[14]#Residual point prediction of ARMA(4,1)
LowerM2=PointM2+quantile(Jul20Model12,0.025) #Residual lower interval prediction of ARMA(4,1)
UpperM2=PointM2+quantile(Jul20Model12,0.975) #Residual upper interval prediction of ARMA(4,1)
#Unstandardizing
PointM2=PointM2*SDV[Jul20]
LowerM2=LowerM2*SDV[Jul20]
UpperM2=UpperM2*SDV[Jul20]
#Adding sinusoidal component
PointM2=PointM2+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)
LowerM2=LowerM2+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)
UpperM2=UpperM2+A1+B1*cos(w*Jul20)+C1*sin(w*Jul20)

PointM2#22.33403
LowerM2#19.02834 (bootstrap of 500)
UpperM2#25.70585 (bootstrap of 500)

#c) Checking Prediction Accuracy. Last days from June 1st till July 6th for AR(4) and ARMA(4,1)
iterations=500#Number of iterations used for bootstrapping
May31=3650+31+28+31+30+31#Time Index May 31st 2009
Jun15=May31+15#Time Index May 31st 2009
Jun22=May31+22#Time Index May 31st 2009
Jul6=Jun22+14#Time Index May 31st 2009

Point1=rep(0,Jun22-May31)#Residual point prediction (22 days) for AR(4)
Upper1=rep(0,Jun22-May31)#Residual upper bound prediction (22 days) for AR(4)
Lower1=rep(0,Jun22-May31)#Residual lower bound prediction (22 days) for AR(4)

Point2=rep(0,Jun22-May31)#Residual point prediction (22 days) forARMA(4,1)
Upper2=rep(0,Jun22-May31)#Residual upper bound prediction (22 days) forARMA(4,1)
Lower2=rep(0,Jun22-May31)#Residual lower bound prediction (22 days) forARMA(4,1)

X1=rep(0,iterations)#Storing bootstrapped residuals for AR(4) for a given date
X2=rep(0,iterations)#Storing bootstrapped residuals for ARMA(4,1) for a given date

Actual=temp[Jun15:Jul6]#Actual Temperatures from Jun 15th to Jul 6th
counter1=0 #Count number of times actual temperature is in predicted 0.95 intervals of AR(4)
counter2=0 #Count number of times actual temperature is in predicted 0.95 intervals of ARMA(4,1)
for (i in 1:(Jul6-Jun15+1))#i denotes each day from June 15th to July 6th (22 days)
{
 for (j in 1:iterations)
 {
  boot.resultM1=Sieve.Boot(M1$residual[1:(May31+i)],0,14) #Bootstrap for AR(4)
```

```
 boot.resultM2=Sieve.Boot(M2$residual[1:(May31+i)],0,14) #Bootstrap for ARMA(4,1)
 X1[j]=boot.resultM1[14]#Get predicted residuals for AR(4) of date May31+i
 X2[j]=boot.resultM2[14]#Get predicted residuals for ARMA(4,1) of date of May31+i
 }
 Point1[i]=predict(M1,n.ahead=14)$pred[14] #Point prediction of date May31+i for AR(4)
 Lower1[i]=Point1[i]+quantile(X1,0.025) #Lower bound prediction of date May31+i for AR(4)
 Upper1[i]=Point1[i]+quantile(X1,0.975) #upper bound prediction of date May31+i for AR(4)

#Unstandardizing
 Point1[i]=Point1[i]*SDV[(May31+i)]
 Lower1[i]=Lower1[i]*SDV[(May31+i)]
 Upper1[i]=Upper1[i]*SDV[(May31+i)]
#Adding sinusoidal component
 Point1[i]=Point1[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
 Lower1[i]=Lower1[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
 Upper1[i]=Upper1[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
#Count if actual temperature lies between predicted interval of date May31+i
 if (Actual[i]>=Lower1[i] && Actual[i]<=Upper1[i]) counter1=counter1+1

 Point2[i]=predict(M2,n.ahead=14)$pred[14]#Point prediction of date May31+i for ARMA(4,1)
 Lower2[i]=Point2[i]+quantile(X2,0.025) #Lower bound prediction of date May31+i for ARMA(4,1)
 Upper2[i]=Point2[i]+quantile(X2,0.975) #Upper bound prediction of date May31+i for ARMA(4,1)

#Unstandardizing
 Point2[i]=Point2[i]*SDV[(May31+i)]
 Lower2[i]=Lower2[i]*SDV[(May31+i)]
 Upper2[i]=Upper2[i]*SDV[(May31+i)]
#Adding sinusoidal component
 Point2[i]=Point2[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
 Lower2[i]=Lower2[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
 Upper2[i]=Upper2[i]+A1+B1*cos(w*(May31+i))+C1*sin(w*(May31+i))
#Count if actual temperature lies between predicted interval of date May31+i
 if (Actual[i]>=Lower2[i] && Actual[i]<=Upper2[i]) counter2=counter2+1
}
counter1/22#Percentage of accuracy of AR(4)
counter2/22#Percentage of accuracy of ARMA(4,1)


#6) Standardizing and Modelling without Sinusoidal Component

ts_stdzd=ts(stdzd_temp,frequency=1,start=1,end=length(stdzd_temp))
plot(ts_stdzd,ylab="Mean Temperature",main="Plot of Standardized Temperatures")
acf(stdzd_temp,main="ACF Plot of Standardized Temperatures")
pacf(stdzd_temp,main="PACF Plot of Standardized Temperatures")

#a)Proceed to seasonal differencing
diff_stdzd_temp=diff(stdzd_temp,lag=365)#Seasonally differencing the data
```

```
ts_diff_stdzd_temp=ts(diff_stdzd_temp,frequency=1,start=1,end=length(diff_stdzd_temp))
plot(ts_diff_stdzd_temp,ylab="Residual",main="Seasonally Differenced Residuals of Standardized
Temperatures")#Improvement
acf(diff_stdzd_temp,main="ACF Plot of Seasonalized Standardized Temperatures")#Better than previous
pacf(diff_stdzd_temp,main="PACF Plot of Seasonalized Standardized Temperatures")#Much better than
previous

#b) Modelling
#Model 1: ARMA(5,1)
model61=arima0(diff_stdzd_temp,order=c(5,0,1))#AIC is 10061.69
model61
#Model 2: ARMA (6,1)#Better than MA=0 and MA>1, d>0, this is the candidate model
model62=arima0(diff_stdzd_temp,order=c(6,0,1))#AIC is 10043.08
model62
#Model 3: ARMA(7,1)
model63=arima0(diff_stdzd_temp,order=c(7,0,1))#AIC is 10044.26
model63
#Model 4: ARMA(6,2)
model64=arima0(diff_stdzd_temp,order=c(6,0,2))#AIC is 10064.51
model64
#Model 5: SARIMA(6,0,1), seasonal(2,0,0)#Does not work due to memory problems
model65=arima0(diff_stdzd_temp,order=c(6,0,1),seasonal=list(order=c(2,0,0),period=365))
model65

#c)Taking Diagnostics of Model 2: our candidate model
library(tseries)
adf.test(model62$residual)#p-value<0.01, reject Ho:there is a unit root
ts.plot(model62$residuals)#Looks homoskedastic with a few spikes
qqnorm(model62$residual)
qqline(model62$residual)#Looks normal with a few aberrational points
shapiro.test(model62$residual)#Very low p-value=0.002409, probably due to outliers
k=order(model62$residual)
outliers=c(k[1],k[2],k[3],k[length(model62$residuals)-3],k[length(model62$residuals)-
2],k[length(model62$residuals)-1],k[length(model62$residuals)])
qqnorm(model62$residual[-outliers])
qqline(model62$residual[-outliers])#Very close to normality
shapiro.test(model62$residual[-outliers])#Significant improvement, p-value=0.89

#Levene's Test: Do not reject homoskedasticity
library(lawstat)
group=c(rep(1,50),rep(2,60),rep(3,(length(model62$residual)-110)))
lev_mean=levene.test(model62$residual,group)
lev_mean#very positive p-value=0.9474
lev_median=levene.test(model62$residual,group,option="median")
lev_median#very positive p-value=0.9466

#d)Predicting 2 weeks ahead
```

```
t_Jul20=length(stdzd_temp)+14#time index value of July 20th in time series

Jul20Model62=predict(model62,n.ahead=14)$pred[14]#Get predicted value of July 20th
Jul20Model62Lower95=Jul20Model62+qnorm(0.025)*(predict(model62,n.ahead=14)$se[14])^0.5
Jul20Model62Upper95=Jul20Model62+qnorm(0.975)*(predict(model62,n.ahead=14)$se[14])^0.5

#Now we need to remove seasonal difference and unstandardize the predicted value
#Adding to July20 2008 (removing seasonal difference)
Jul20Model62=Jul20Model62+stdzd_temp[t_Jul20-365]
Jul20Model62Lower95=Jul20Model62Lower95+stdzd_temp[t_Jul20-365]
Jul20Model62Upper95=Jul20Model62Upper95+stdzd_temp[t_Jul20-365]
#Unstandardizing
Jul20Model62=Jul20Model62*SDV[31+28+31+30+31+30+20]
Jul20Model62Lower95=Jul20Model62Lower95*SDV[31+28+31+30+31+30+20]
Jul20Model62Upper95=Jul20Model62Upper95*SDV[31+28+31+30+31+30+20]

Jul20Model62#Predicted temperature is 19.75824
Jul20Model62Lower95#Lower 95% confidence level= 14.54769
Jul20Model62Upper95#Upper 95% confidence level= 24.96879

#e) Testing the Model Two Weeks Ahead from January 1st till June 22

June22=31+28+31+30+31+22
Dec31=365*10#Index of December 31th 2008
Test=rep(0,June22)#Predicted results of test
Lower95=rep(0,June22)#Predicted lower intervals at 95% confidence level
Upper95=rep(0,June22)#Predicted upper intervals at 95% confidence level
counter=0#Counter used to check how often the actual values lie in the intervals
diff_tempARMA=rep(0,June22)#Absolute differences

for(i in 1:June22)
{
 X=stdzd_temp[1:(Dec31+i)]#Starting point January 1st
 diff_X=diff(X,lag=365)#Seasonally differencing the data
 model61=arima0(diff_X,order=c(6,0,1))#Assume ARMA(6,1) is best fit
 t_pred=length(X)+14#Date of prediction (starting point: January 15th)
 X_pred=predict(model61,n.ahead=14)$pred[14]#Get predicted value 14 days ahead
 Lower95[i]=X_pred+qnorm(0.025)*(predict(model61,n.ahead=14)$se[14])^0.5
 Upper95[i]=X_pred+qnorm(0.975)*(predict(model61,n.ahead=14)$se[14])^0.5

 #Now we need to remove seasonal difference and unstandardize the predicted value
 #Removing seasonal difference
 X_pred=X_pred+X[t_pred-365]
 Lower95[i]=Lower95[i]+X[t_pred-365]
 Upper95[i]=Upper95[i]+X[t_pred-365]
 #Unstandardizing
 X_pred=X_pred*SDV[14+i]
```

```
 Lower95[i]=Lower95[i]*SDV[14+i]
 Upper95[i]=Upper95[i]*SDV[14+i]

 Test[i]=X_pred
 if (temp[Dec31+i+14]>=Lower95[i] && temp[Dec31+i+14]<=Upper95[i]) counter=counter+1

 diff_tempARMA[i]=temp[Dec31+i+14]-Test[i]#Difference between original and predicted
}

#Assessing Prediction Points
abs_diffARMA=abs(diff_tempARMA)
ts.plot(diff_tempARMA,ylab="Actual-Predicted",main="Difference between Actual and Predicted
Temperatures")
sum(diff_tempARMA)#7.865395-->Underestimates the data more often
max(abs_diffARMA)#22.90181
min(abs_diffARMA)#Nearly zero
mean(abs_diffARMA)#4.7630309
(var(abs_diffARMA))^0.5#4.050973

#Assessing Prediction Intervals
In_95interval=counter/June22#How often does the actual temperature fall in the confidence interval
In_95interval#92.48555%
diff_interval=Upper95-Lower95
ts.plot(diff_interval,ylab="Interval Length",main="Interval Lengths across Time")
max(diff_interval)#37.14704
min(diff_interval)#6.89344
mean(diff_interval)#20.6781
(var(diff_interval))^0.5#5.816204
```