1

Key and explanations for Homework Week 3

Data Set Up

```
library(MARSS)
dat=log(grouse[,2])
```

Problem 1

Write the equations for each of these models: ARIMA(0,0,0), ARIMA(0,1,0), ARIMA(1,0,0), ARIMA(0,0,1), ARIMA(1,0,1).

In all cases, $e_t \sim N(0,\sigma^2)$ and e_t and e_{t-1} are independent. Thus the e_t are i.i.d. (independent and identically distributed).

ARIMA(0,0,0) = white noise

$$x_t = e_t$$

ARIMA(0,1,0) = differenced data is white noise

$$x_t - x_{t-1} = e_t$$

ARIMA(1,0,0) = Autoregressive lag-1. This is a mean-reverting random walk if $|b| < 1$. If $b = 1$, it is a simple random walk and the same as ARIMA(0,1,0).

$$x_t = bx_{t-1} + e_t$$

ARIMA(0,0,1) is a moving-average lag-1 model.

$$x_t = e_t + \theta e_{t-1}$$

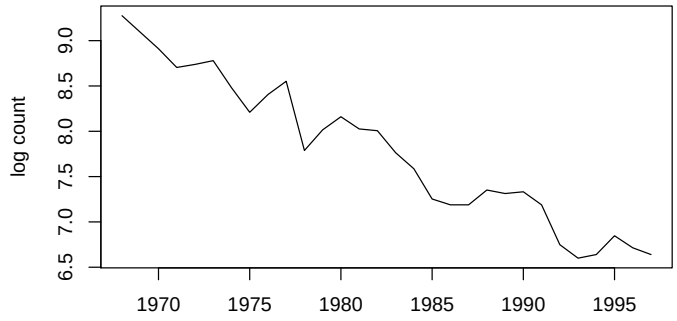
ARIMA(1,0,1) is a moving-average lag-1 model with autoregression lag-1.

$$x_t = bx_{t-1} + e_t + \theta e_{t-1}$$

Problem 2

a) Plot the data.

```
plot(grouse[,1], dat, type="l", ylab="log count", xlab="")
```



b) Fit each model using `MARSS()`.

```
mod.list1=list(
  B=matrix(1), U=matrix(0), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix("a"), tinitx=0)
fit1.marss = MARSS(dat, model=mod.list1)
mod.list2=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix("a"), tinitx=0)
fit2.marss = MARSS(dat, model=mod.list2)
```

```
coef(fit1.marss, type="vector")
```

```
      Q.q       x0.a
0.05156616 9.27537877
```

```
coef(fit2.marss, type="vector")
```

```
       U.u         Q.q        x0.a
-0.09087941  0.04358239  9.36625818
```

c) Which one appears better supported given AICc?

```
c(fit1.marss$AICc, fit2.marss$AICc)
```

```
[1]  0.6340661 -1.9336725
```

Model 2 is better supported with a $\Delta AICc$ of -2.56773865439913.

d) Load the `forecast` package. Use `auto.arima(dat)` to fit the data. Next run `auto.arima` on the data with `trace=TRUE` to see all the ARIMA models it compared.

```
library(forecast)
auto.arima(dat)
```

Let's look at all the models it tried using `trace=TRUE`.

```
auto.arima(dat, trace=TRUE)
```

```
ARIMA(2,1,2) with drift         : Inf
ARIMA(0,1,0) with drift         : -3.116841
ARIMA(1,1,0) with drift         : -1.006099
ARIMA(0,1,1) with drift         : Inf
ARIMA(0,1,0)                    : -0.5520726
ARIMA(1,1,1) with drift         : Inf

Best model: ARIMA(0,1,0) with drift

Series: dat
ARIMA(0,1,0) with drift

Coefficients:
        drift
      -0.0909
s.e.   0.0394

sigma^2 estimated as 0.0467:  log likelihood=3.79
AIC=-3.58   AICc=-3.12   BIC=-0.84
```

It picked model 2 as the best among those tested. "ARIMA(0,1,0) with drift" is model 2.

e) Is the difference in the AICc values between a random walk with and without drift comparable between MARSS() and auto.arima()?

```
fit1.arima=Arima(dat, order=c(0,1,0))
fit2.arima=Arima(dat, order=c(0,1,0), include.drift=TRUE)
fit2.arima$aicc-fit1.arima$aicc
```

```
[1] -2.564768
```

```
fit2.marss$AICc-fit1.marss$AICc
```

```
[1] -2.567739
```

Similar but not identical. BTW, to figure how to get AICc from an `Arima()` fit, I tried `names(fit1.arima)` and saw that AICc was in the element named `aicc`.

Problem 3

This produces $x_t = x_{t-1} + u + w_t$ data with $u = 0.1$ and $q = 1$.

```
dat=cumsum(rnorm(100,0.1,1))
```

a) Write out the equation for that random walk as a univariate state-space model.

$$\begin{gathered} x_t = x_{t-1} + u + w_t, w_t \sim \mathrm{N}(0,q) \\ x_0 = \mu \text{ or } x_1 = y_1 \\ y_t = x_t \end{gathered} \tag{1.1}$$

where $u = 0.1$ and $q = 1$.

b) What is the order of the **x** part of the model written as ARIMA(p, d, q)?

From question 1, you should be able to deduce it is ARIMA(0,1,0) but if you said ARIMA(1,0,0) with b=1, that's ok. That's not how `Arima()` writes $x_t = x_{t-1} + u + w_t$ but it is correct.

c) Fit that model using `Arima()` in the `forecast` package. You'll need to specify the `order` and `include.drift` term.

```
fit.arima=Arima(dat, order=c(0,1,0), include.drift=TRUE)
```

d) Fit that model with `MARSS()`.

```
mod.list=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix("mu"), tinitx=0)
fit.marss = MARSS(dat, model=mod.list)
```

or since I know that $x_1 = y_1$ from the observation model, I could use:

```
mod.list.alt=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix(dat[1]), tinitx=1)
fit.alt.marss = MARSS(dat, model=mod.list.alt, method="BFGS")
```

But hopefully you didn't try that because this makes the likelihood surface flat and you need to use lots more iterations or try a Newton method which happens to help (sometimes it doesn't or makes thing worse).

e) How are the two estimates different?

```
coef(fit.marss, type="vector")
```

```
      U.u        Q.q      x0.mu
0.1372300  0.9231202 -0.9145116
```

```
coef(fit.alt.marss, type="vector")
```

```
     U.u       Q.q
0.1372299 0.9324444
```

```
c(coef(fit.arima), s2=fit.arima$sigma2)
```

```
   drift        s2
0.1372300 0.9419594
```

MARSS() is estimating 3 parameters while Arima() is estimating 2. The u estimates are identical (or very similar) but the q estimate is different.

`coef()` is the standard function for getting estimates from fits. Try `?coef` to find the help file for `coef` applied to MARSS objects. For Arima objects, `coef()` doesn't return sigma2 (which I discovered by trying `coef(fit.arima)`). So I did `names(fit.arima)` and found it was in `fit.arima$sigma2`.

Now fit the first-differenced data:

```
diff.dat=diff(dat)
```

f) If x_t denotes a time series. What is the first difference of x? What is the second difference?

First difference `diff(x)` is $x_t - x_{t-1}$.

Second difference is `diff(diff(x))` or $(x_t - x_{t-1}) - (x_{t-1} - x_{t-2})$.

g) What is the **x** model for `diff.dat`?

$$\text{diff}(x) = (x_t - x_{t-1}) = u + w_t$$

h) Fit `diff.dat` using `Arima()`. You'll need to change `order` and `include.mean`. eeh note 2/7: I should have had you use `arima()` in the question; `Arima()` is reporting the unbiased variance estimate while MARSS and arima are reporting the straight maximum-likelihood estimate. The difference is $(n-1)/n$ where n is the length of the data, in this case the difference data. So $(n-1)/n = 98/99$.

```
fit.diff.Arima=Arima(diff.dat, order=c(0,0,0), include.mean=TRUE)
fit.diff.arima=arima(diff.dat, order=c(0,0,0), include.mean=TRUE)
```

i) Fit that model with `MARSS()`.

data (y) is now diff.dat and state-space model is

$$\begin{aligned} x_t &= u + w_t, w_t \sim \mathrm{N}(0, q) \\ x_0 &= 0 \\ y_t &= x_t \end{aligned} \tag{1.2}$$

It doesn't matter what x_0 is; it does not appear in the model, but it is important to use x_0 instead of x_1 to match `arima()`.

```
mod.list.diff.1=list(
  B=matrix(0), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix(0), tinitx=0)
fit.diff.marss.1 = MARSS(diff.dat, model=mod.list.diff.1)
```

Or we could have written it like so

$$\begin{aligned} x_t &= 0, w_t \sim \mathrm{N}(0, 0) \\ x_0 &= 0 \\ y_t &= a + v_t, v_t \sim \mathrm{N}(0, r) \end{aligned} \tag{1.3}$$

In this case, we fit this model

```
mod.list.diff.2=list(
  B=matrix(0), U=matrix(0), Q=matrix(0),
  Z=matrix(0), A=matrix("u"), R=matrix("r"),
  x0=matrix(0), tinitx=0)
fit.diff.marss.2 = MARSS(diff.dat, model=mod.list.diff.2)
```

Note, we can also fit with lm():

```
fit.diff.lm = lm(diff.dat~1)
```

Here's the parameter estimates.

```
rbind(
  marss.diff.1=coef(fit.diff.marss.1, type="vector"),
  marss.diff.2=coef(fit.diff.marss.2, type="vector"),
  arima.diff=c(coef(fit.diff.arima), s2=fit.diff.arima$sigma2),
  Arima.diff=c(coef(fit.diff.Arima), s2=(98/99)*fit.diff.Arima$sigma2),
  lm.diff=c(coef(fit.diff.lm), s2=(98/99)*summary(fit.diff.lm)$sigma^2)
)
```

```
                U.u       Q.q
marss.diff.1 0.13723 0.9324446
marss.diff.2 0.13723 0.9324446
arima.diff   0.13723 0.9324446
Arima.diff   0.13723 0.9324446
lm.diff      0.13723 0.9324446
```

They are all the same except the variances reported by `Arima()` and lm() have to be multiplied by 98/99 to be the same as MARSS and arima because the former are reporting the unbiased estimates and the latter are reporting the straight (biased) maximum-likelihood estimates.

Problem 4

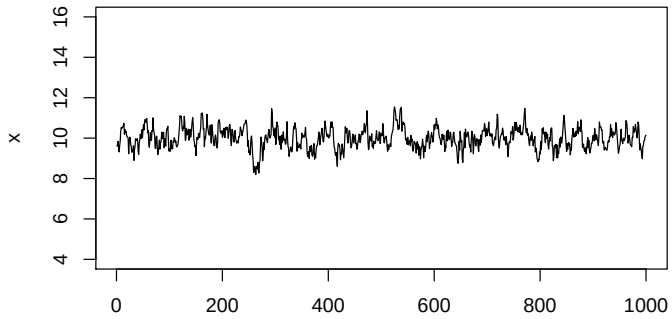
$$x_t = bx_{t-1} + u + w_t \text{ where } w_t \sim \text{N}(0, q) \tag{1.4}$$

a) Write R code to simulate Equation 1.4. Make b less than 1 and greater than 0. Set u and x_0 to whatever you want. You can use a `for` loop.

```
#set up my parameter values
b=.8; u=2; x0=10; q=0.1
nsim=1000
#set up my holder for x
x=rep(NA, nsim)
x[1]=b*x0+u+rnorm(1,0,sqrt(q))
for(t in 2:nsim) x[t]=b*x[t-1]+u+rnorm(1,0,sqrt(q))
```

b) Plot the trajectories and show that this model does not "drift" upward or downward. It fluctuates about a mean value.

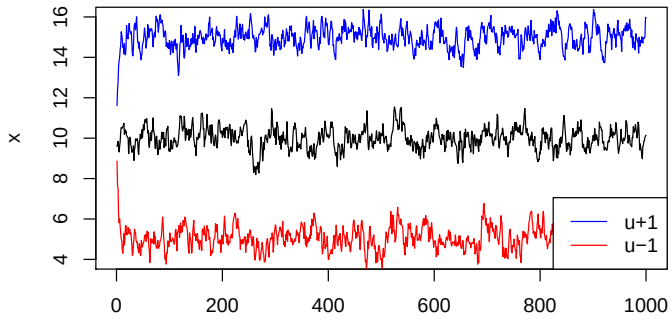
```
plot(x, type="l",xlab="", ylab="x")
```



c) Hold b constant and change u. How do the trajectories change?

```
#set up my parameter values
u2=u+1
x2=rep(NA, nsim)
```

```
x2[1]=b*x0+u2+rnorm(1,0,sqrt(q))
for(t in 2:nsim) x2[t]=b*x2[t-1]+u2+rnorm(1,0,sqrt(q))
#second u
u3=u-1
x3=rep(NA, nsim)
x3[1]=b*x0+u3+rnorm(1,0,sqrt(q))
for(t in 2:nsim) x3[t]=b*x3[t-1]+u3+rnorm(1,0,sqrt(q))
```

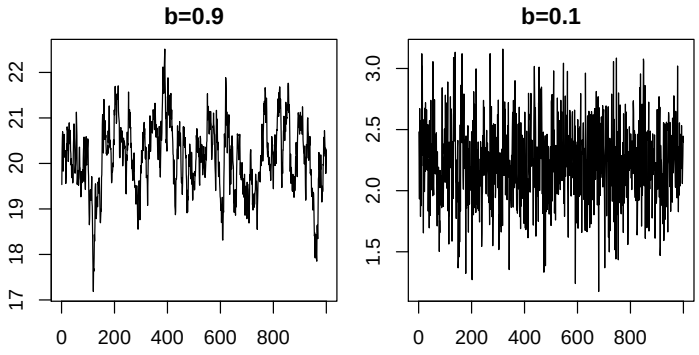


u moves the mean of the trajectories up or down.

d) Hold u constant and change b. Make sure to use a b close to 1 and another close to 0. How do the trajectories change?

```
#set up my parameter values
b1=0.9
x0=u/(1-b1)
x1=rep(NA, nsim)
x1[1]=b1*x0+u+rnorm(1,0,sqrt(q))
for(t in 2:nsim) x1[t]=b1*x1[t-1]+u+rnorm(1,0,sqrt(q))
# second b
b2=0.1
x0=u/(1-b2)
x2=rep(NA, nsim)
x2[1]=b2*x0+u+rnorm(1,0,sqrt(q))
for(t in 2:nsim) x2[t]=b2*x2[t-1]+u+rnorm(1,0,sqrt(q))
```

The one with smaller b has less auto-regression and is 'tighter' (explores less of a range of the y axis).



e) Do 2 simulations each with the same w_t. In one simulation, set $u = 1$ and in the other $u = 2$. For both simulations, set $x_1 = u/(1-b)$. You can set b to whatever you want as long as $0 < b < 1$. Plot the 2 trajectories on the same plot. What is different?

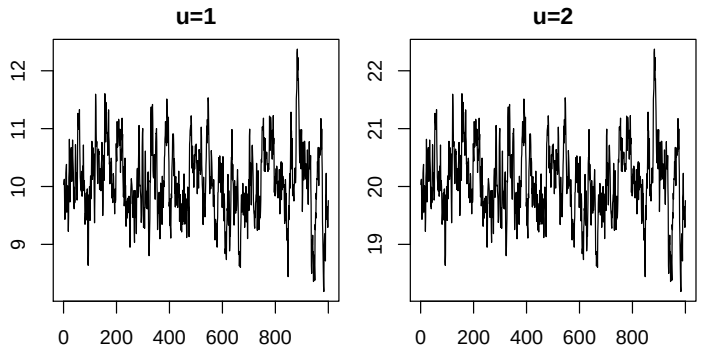
```
#set up my parameter values
b=0.9
u=1
x0=u/(1-b)
err=rnorm(nsim,0,sqrt(q))
x1=rep(NA, nsim)
x1[1]=b*x0+u+err[1]
for(t in 2:nsim) x1[t]=b*x1[t-1]+u+err[t]
# second u
u=2
x0=u/(1-b)
x2=rep(NA, nsim)
x2[1]=b*x0+u+err[1]
for(t in 2:nsim) x2[t]=b*x2[t-1]+u+err[t]
```

They are exactly the same except that the mean has changed from $1/(1-b)$ to $2/(1-b)$. The mean level in the AR-1 model $x_t = bx_{t-1} + u + w_t$ is $u/(1-b)$. For a given b, u just changes the level.

Problem 5

The MARSS package includes a data set of gray whales. Load the data to use as follows:

```
library(MARSS)
dat=log(graywhales[,2])
```



Fit a random walk with drift model observed with error to the data:

$$\begin{aligned} x_t &= x_{t-1} + u + w_t \text{ where } w_t \sim \mathrm{N}(0,q) \\ y_t &= x_t + v_t \text{ where } v_t \sim \mathrm{N}(0,r) \\ x_0 &= a \end{aligned} \tag{1.5}$$

y is the whale count in year t. x is interpreted as the 'true' unknown population size that we are trying to estimate.

a) Fit this model with `MARSS()`

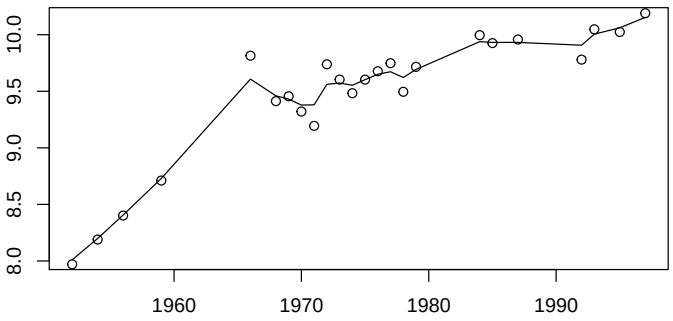
```
mod.list=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix("r"),
  x0=matrix("mu"), tinitx=0)
fit.marss = MARSS(dat, model=mod.list)
```

b) Plot the estimated x as a line with the actual counts added as points.

c) Simulate 1000 sample trajectories using the estimated u and q starting at the estimated x in 1997. You can do this with a couple `for` loops or write something terse with `cumsum` and `apply`.

```
#1997 is the 39th (last) data point
x0=fit.marss$states[1,39]
q = coef(fit.marss)$Q
u = coef(fit.marss)$U
#next question asks for pop size in 2007 so nforeward=10
nsim=1000
nforeward = 10
#each row holds a simulation
x=matrix(NA, nsim, nforeward)
```

```
par(mar=c(2,2,2,2))
plot(graywhales[,1], fit.marss$states[1,], type="l",xlab="", ylab="log count")
points(graywhales[,1], dat)
```




```
x[,1]=x0+u+rnorm(nsim,0,sqrt(q))
for(t in 2:nforeward) x[,t]=x[,t-1]+u+rnorm(nsim,0,sqrt(q))
```

d) Using this what is your estimated probability of reaching 50,000 graywhales in 2007.

The question was phrased a big vaguely. It does not specify if this means "in 2007, x=log(50000)", "at some point by or before 2007, x reaches log(50000) at least once", or "in 2007, the population is at least 50000 whales". I was thinking of the last one, but as long as you stated what you were trying to estimate, you were fine.

```
#I just want the fraction of simulations that were 50,000 or above in 2007
xthresh = log(50000)
sum(x[,10]<=xthresh)/nsim
```

```
[1] 0.586
```

e) What kind of uncertainty does that estimate NOT include? By using the point estimates of u, q and x_0, we are not including the uncertainty in those estimates in our forecasts.

Problem 6

Fit the following 3 models to the graywhales data using MARSS():

1. Process error only model with drift
2. Process error only model without drift

3. Process error with drift and observation error with observation error variance fixed = 0.05.
4. Process error with drift and observation error with observation error variance estimated.

Process error only with drift. $x_t = x_{t-1} + u + w_t$ with $y_t = x_t$.

```
mod.list=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix("mu"), tinitx=0)
fit.whales1 = MARSS(dat, model=mod.list)
```

Process error only without drift. $x_t = x_{t-1} + w_t$ with $y_t = x_t$.

```
mod.list=list(
  B=matrix(1), U=matrix(0), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0),
  x0=matrix("mu"), tinitx=0)
fit.whales2 = MARSS(dat, model=mod.list)
```

Process error only with drift. $x_t = x_{t-1} + w_t$ with $y_t = x_t + v_t, v_t \sim N(0, 0.05)$.

```
mod.list=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix(0.05),
  x0=matrix("mu"), tinitx=0)
fit.whales3 = MARSS(dat, model=mod.list)

mod.list=list(
  B=matrix(1), U=matrix("u"), Q=matrix("q"),
  Z=matrix(1), A=matrix(0), R=matrix("r"),
  x0=matrix("mu"), tinitx=0)
fit.whales4 = MARSS(dat, model=mod.list)
```

a) Compute the AICc's for each model and likelihood or deviance (-2 * log likelihood)

```
c(fit.whales1$AICc, fit.whales2$AICc,
  fit.whales3$AICc, fit.whales4$AICc)
```

```
[1] 2.131810 2.875514 3.760801 1.975372
```

```
c(fit.whales1$logLik, fit.whales2$logLik,
  fit.whales3$logLik, fit.whales4$logLik)
```

```
[1] 2.5340949 0.8479575 1.7195997 4.0649455
```

b) Calculate a table of delta-AICc values and AICc weights.

```
AICc=c(fit.whales1$AICc, fit.whales2$AICc,
       fit.whales3$AICc, fit.whales4$AICc)
delAIC = AICc-min(AICc)
relLik = exp(-0.5*delAIC)
aic.table=data.frame(
  AICc = AICc,
  delAICc = delAIC,
  relLik = relLik/sum(relLik)
)
rownames(aic.table) = c(
  "proc only with drift",
  "proc only no drift",
  "proc with drift and obs error fixed",
  "proc with drift and obs error est")
round(aic.table, digits=3)
```

```
                                    AICc delAICc relLik
proc only with drift               2.132   0.156  0.311
proc only no drift                 2.876   0.900  0.215
proc with drift and obs error fixed 3.761   1.785  0.138
proc with drift and obs error est   1.975   0.000  0.336
```

There is not much data support for including observation error with $r = 0.05$. But that is because $r = 0.05$ is too big. If we estimate r, the process error with drift and observation error model would is best.

Problem 7

Load the data to use as follows and set up so you can use the last 3 data points to validate your fits.

```
library(forecast)
dat=log(airmiles)
n=length(dat)
training.dat = dat[1:(n-3)]
test.dat = dat[(n-2):n]
```

a) Fit the following four models using `Arima()`: ARIMA(0,0,0), ARIMA(1,0,0), ARIMA(0,0,1), ARIMA(1,0,1).

```
fit.1=Arima(training.dat, order =c(0,0,0))
fit.2=Arima(training.dat, order =c(1,0,0))
fit.3=Arima(training.dat, order =c(0,0,1))
fit.4=Arima(training.dat, order =c(1,0,1))
```

b) Use `forecast()` to make 3 step ahead forecasts from each.

```
forecast.1=forecast(fit.1, h=3)
forecast.2=forecast(fit.2, h=3)
forecast.3=forecast(fit.3, h=3)
forecast.4=forecast(fit.4, h=3)
```

c) Calculate the MASE statistic for each using the `accuracy` function in the forecast package.

```
accuracy(forecast.1, test.dat)
```

```
                      ME     RMSE      MAE       MPE
Training set -4.228079e-16 1.274201 1.121923 -2.565641
Test set      1.899534e+00 1.901199 1.899534 18.526816
                 MAPE     MASE    ACF1
Training set 14.26941 5.391961 0.85469
Test set     18.52682 9.129155      NA
```

The MASE statistic we want is in the Test set row and MASE column.

```
MASEs = c(
  accuracy(forecast.1, test.dat)["Test set","MASE"],
  accuracy(forecast.2, test.dat)["Test set","MASE"],
  accuracy(forecast.3, test.dat)["Test set","MASE"],
  accuracy(forecast.4, test.dat)["Test set","MASE"]
)
```

d) Present the results in a table.

```
data.frame(
  name=paste("Arima",c("(0,0,0)","(1,0,0)","(0,0,1)","(1,0,1)"),sep=""),
  MASE=MASEs
  )
```

```
         name      MASE
1 Arima(0,0,0) 9.1291550
2 Arima(1,0,0) 0.6906049
3 Arima(0,0,1) 7.7353124
4 Arima(1,0,1) 0.5119703
```

e) Which model is best supported based on the MASE statistic?

What this table shows is that the ARMA(1,0,1) is the best, and the AR component strongly improves predictions

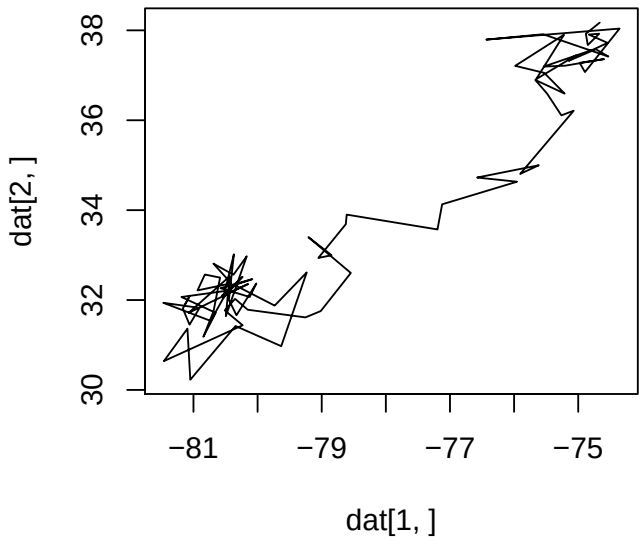
Problem 8

Set up the data

```
turtlename="MaryLee"
dat = loggerheadNoisy[which(loggerheadNoisy$turtle==turtlename),5:6]
dat = t(dat)
```

a) Plot MaryLee's locations (as a line not dots). Put the latitude locations on the y-axis and the longitude on the y-axis.

```
plot(dat[1,],dat[2,], type="l")
```



b) Analyze the data with a state-space model (movement observed with error) using

```
fit0 = MARSS(dat)
```

U_{lon} is the average velocity in N-S direction. U_{lat} is the average velocity in E-W direction. R_{diag} is the observation error variance. Q's are the movement error variances. x_0's are the estimated positions (lat/lon) at $t = 0$.

c) What assumption did the default MARSS model make about observation error and process error?

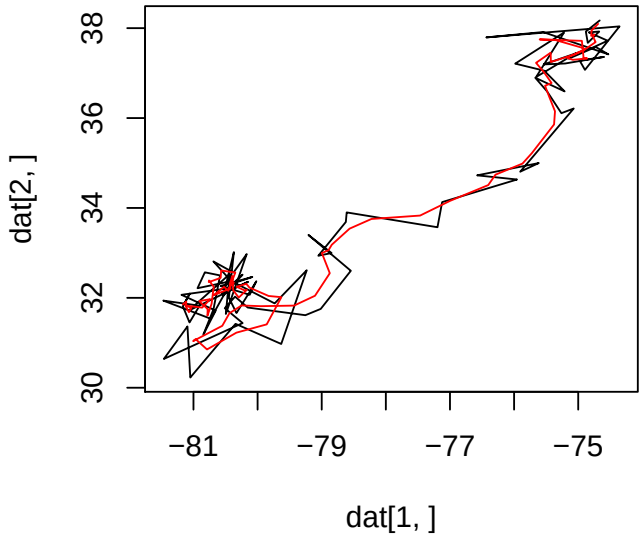
The observation errors in the lat and lon direction are independent but have identical variance. The movement errors are independent (not correlated) and allowed to have different variances. So the model doesn't allow a average NE movement; that would require correlation in the movement errors. It allows that turtles tend to move faster N-S (along the coast) than E-W (out to sea).

d) Does MaryLee move faster in the latitude direction versus longitude direction?

No. The estimated u's in the lat and lon direction are similar.

e) Add MaryLee's estimated "true" positions to your plot of her locations.

```
plot(dat[1,],dat[2,], type="l")
lines(fit0$states[1,], fit0$states[2,], col="red")
```



f) Compare the following models for these data. Movement in the lat/lon direction is (1) independent but the variance is the same, (2) is correlated and lat/lon variances are different, and (3) is correlated and the lat/lon variances are the same.

```
fit1 = MARSS(dat, model=list(Q="diagonal and equal"))
fit2 = MARSS(dat, model=list(Q="unconstrained"))
fit3 = MARSS(dat, model=list(Q="equalvarcov"))
```

```
c(fit0=fit0$AICc, fit1=fit1$AICc,
  fit2=fit2$AICc, fit3=fit3$AICc)
```

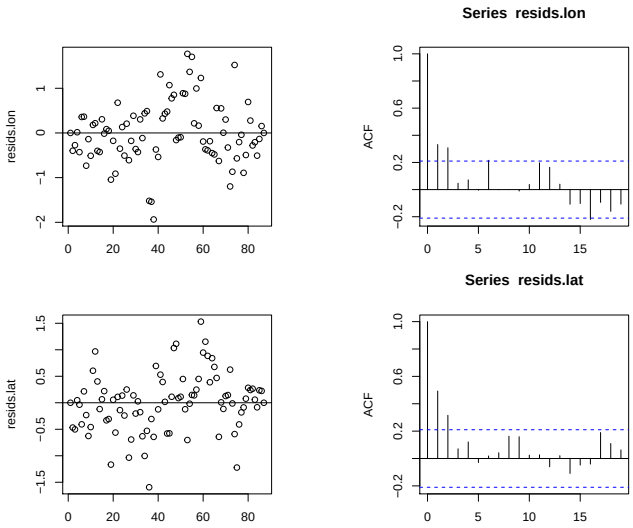
```
    fit0     fit1     fit2     fit3
267.0901 264.9996 260.0077 257.8289
```

The model with correlated movement but equal movement error variances is best supported. This suggests a tendency to move in a particular direction (probably up down the coast). However, actually this is caused by strong directional movement in the middle of the movement track.

g) Plot your state residuals (true location residuals). What are the problems? Discuss in reference to your plot of the location data. Here is how to get state residuals from MARSS:

```
par(mfrow=c(2,2),mar=c(3,5,3,5))
resids.lon = residuals(fit3)$state.residuals[1,]
plot(resids.lon, xlab=""); abline(h=0)
acf(resids.lon)
resids.lat = residuals(fit3)$state.residuals[2,]
plot(resids.lat, xlab=""); abline(h=0)
acf(resids.lat)
```

There is a period in the middle of the track where the model does not describe the movement well. We can see in the plot that the turtle has a long northward movement in the middle of the track.


Series resids.lon
resids.lon
ACF
Series resids.lat
resids.lat
ACF