Module 8 - Estimation and Prediction

MATH1307 Forecasting

RMIT University, School of Science, Mathematical Sciences

All the contents in this presentation are mainly based on the textbook of MATH1307 Forecasting course 'Hyndman et al., Forecasting with Exponential Smoothing: The State Space Approach. Springer, 2008.' Other resources than the textbook are cited accordingly.

Introduction

In the innovations state-space models, although there are not lots of parameters exists in the model, we need to estimate the initial (seed) states and the parameters as they are usually unknown.

This can be done using maximum likelihood estimation, based on the innovations representation of the probability density function.

These innovations can be used to compute the likelihood, which is then optimized with respect to the seed states and the parameters.

We will use numerical optimization to obtain the maximum likelihood estimates.

Thus, appropriate choice of starting values is also important.

Starting values that are as close as possible to the optimal estimates not only increase the chances of finding the true optimum but typically reduce the computational loads required during the search for the optimum solution.

We will introduce the basic methodology and discuss plausible heuristics for determining the starting values.

It is also on interest to give interval estimates for point forecasts.

For this aim, we compute the associated prediction distributions and prediction intervals for each model.

We will also focus on this problem in this module.

Maximum Likelihood Estimation

Maximum likelihood estimates (MLEs) are consistent and asymptotically efficient under reasonable conditions for the innovations state-space models.

To derive MLEs, we need to formulate the likelihood function, which is based on distributional assumptions on the series $\{Y_t\}$.

If we put smoothing parameters and damping factors into the parameter vector $\boldsymbol{\theta}$, the likelihood function is a function of $\boldsymbol{\theta}$.

It also depends on the k-vector $\boldsymbol{X}_0$ of seed states.

We will use the fact that any time series $\{Y_t\}$ governed by a linear state-space model with Normally distributed innovations has a multivariate Normal distribution.

We write the joint density of the series as the weighted product of the densities of the individual innovations as follows:

$$p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{x}_0, \sigma^2) = \prod_{t=1}^{n} p(\epsilon_t)/|r(\boldsymbol{x}_{t-1})| \tag{1}$$

The corresponding log-likelihood is written as

$$\begin{aligned} \log L(\boldsymbol{\theta}, \boldsymbol{x}_0, \sigma^2|\boldsymbol{y}) = \quad & -\tfrac{n}{2}\log(2\pi\sigma^2) - \textstyle\sum_{t=1}^{n} \log|r(\boldsymbol{x}_{t-1})| \\ & -\tfrac{1}{2}\textstyle\sum_{t=1}^{n} \epsilon_t/\sigma^2. \end{aligned} \tag{2}$$

Then taking the partial derivative with respect to σ^2 and setting it to zero gives the maximum likelihood estimate of the innovations variance σ^2 as

$$\hat{\sigma}^2 = n^{-1} \sum_{t=1}^{n} \epsilon_t^2. \tag{3}$$

We write this estimate in place in the likelihood and maximum likelihood estimates of the parameters can be obtained by minimizing twice the negative log-likelihood

$$\log L^*(\boldsymbol{\theta}, \boldsymbol{x}_0, \sigma^2 | \boldsymbol{y}) = n \log \left(\sum_{t=1}^{n} \epsilon_t^2 \right) + 2 \sum_{t=1}^{n} \log |r(\boldsymbol{x}_{t-1})|. \tag{4}$$

Equivalently, they can be obtained by minimizing the augmented sum of squared errors criterion.

In this case, we do not need to feed the likelihood with the estimate of σ^2.

$$S(\boldsymbol{\theta}, \boldsymbol{x}_0) = [exp(\log L^*(\boldsymbol{\theta}, \boldsymbol{x}_0, \sigma^2 | \boldsymbol{y}))]^{1/n} \tag{5}$$
$$= \left| \prod_{t=1}^{n} r(\boldsymbol{x}_{t-1}) \right|^{2/n} \sum_{t=1}^{n} \epsilon_t^2 .$$

In homoscedastic cases, where there is no changing variance, $r(\boldsymbol{x}_{t-1}) = 1$ reduces to the traditional sum of squared errors.

We can start the estimation process by directly specifying that the objective is to minimize $S(\boldsymbol{\theta}, \boldsymbol{x}_0)$.

This approach, known as the Augmented Least Squares (ALS) method, does not require us to make any assumptions about the distributional form of the errors.

More generally, when the MLEs are computed from (5) without any assumption of Normality, we refer to the results as quasi-maximum likelihood estimators.

Such estimators are often consistent, but the expressions for the standard errors of the estimators may be biased, even asymptotically.

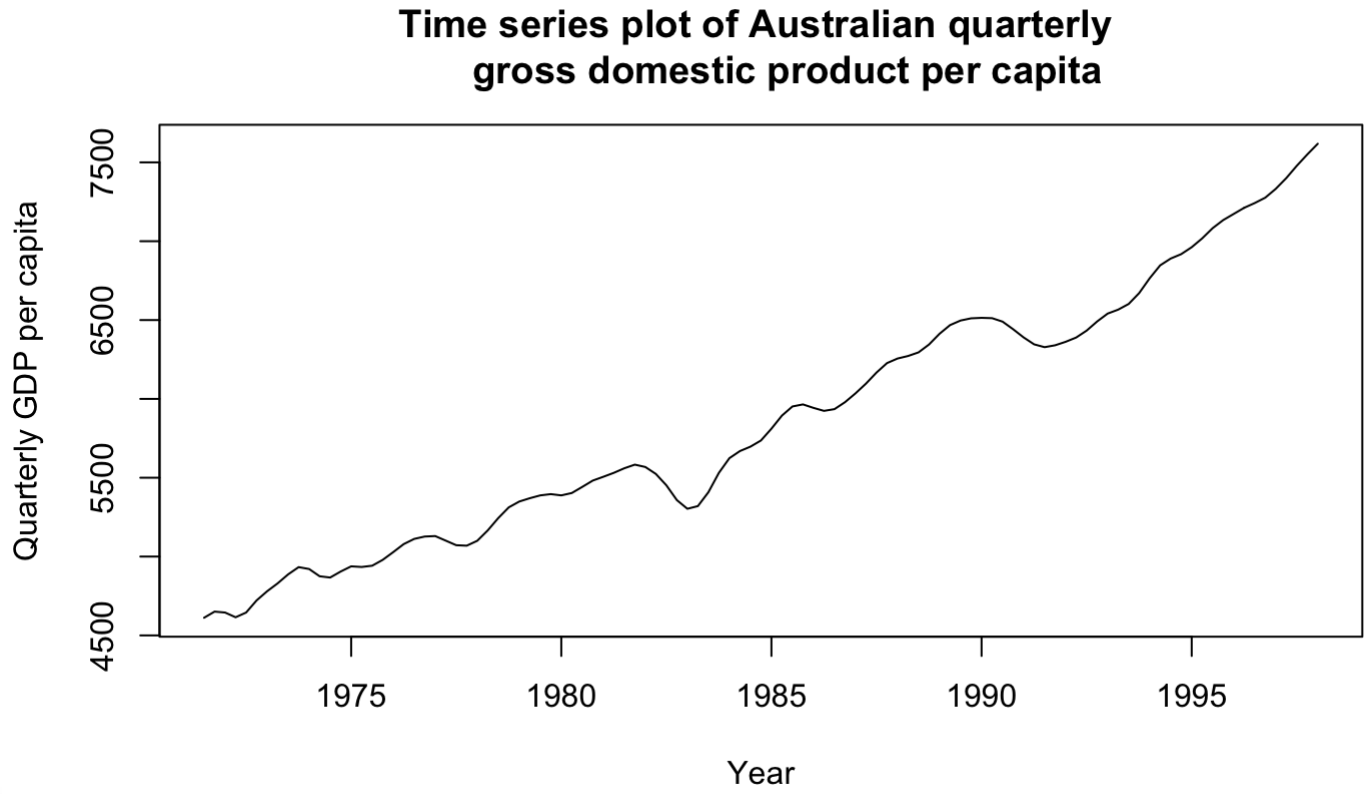
Application: Australian GDP

To illustrate the method, let's focus on the Australian quarterly real gross domestic product per capita from the September quarter of 1971 to the March quarter of 1998.

This series is available through `ausgdp` dataset of `expsmooth` package.

Time series plot of this series is as follows:

```
data("ausgdp") # Available in expsmooth package
plot(ausgdp,ylab = "Quarterly GDP per capita", xlab = "Year", main="Time series plot of Au
     gross domestic product per capita")
```



We fit a local linear trend model ETS(A,A,N) to this GDP series by minimizing Eq. (5).

We impose the constraints $\alpha \geq 0$, $\beta \geq 0$, and $2\alpha + \beta \leq 4$ to ensure stability using `upper` argument.

The parameter estimates with the conventional constraints lie on the boundary of the parameter space are as follows:

```r
fit.ausgdp.AAN.conv <- ets(ausgdp,"AAN" , damped=FALSE , upper=rep(1,4))
summary(fit.ausgdp.AAN.conv)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = ausgdp, model = "AAN", damped = FALSE, upper = rep(1,
##
##  Call:
##      4))
##
##    Smoothing parameters:
##      alpha = 1
##      beta  = 1
##
##    Initial states:
##      l = 4584.5399
##      b = 35.4444
##
##    sigma:  25.7961
##
##       AIC      AICc       BIC
## 1201.463 1202.057 1214.828
##
## Training set error measures:
##                      ME     RMSE      MAE         MPE      MAPE      MASE
## Training set 0.3042576 25.30929 20.34135 0.00672284 0.3639365 0.1403228
##                    ACF1
## Training set 0.485319
```

The estimates are within the interior of the parameter space in the following fit:

```
fit.ausgdp.AAN.stable <- ets(ausgdp,"AAN",damped=FALSE,bounds="admiss")
summary(fit.ausgdp.AAN.stable)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = ausgdp, model = "AAN", damped = FALSE, bounds = "admiss")
##
##   Smoothing parameters:
##     alpha = 0.62
##     beta  = 2.5364
##
##   Initial states:
##     l = 4561.761
##     b = 48.9696
##
##   sigma:  17.377
##
##      AIC     AICc      BIC
## 1116.918 1117.512 1130.282
##
## Training set error measures:
##                     ME     RMSE      MAE          MPE      MAPE       MASE
## Training set 0.1115701 17.04914 13.39693 0.002797773 0.2399168 0.09241737
##                     ACF1
## Training set -0.04740035
```

RMSE has considerably decreased when we relax the conventional constraints to the stability conditions by setting `bounds="admiss"`.

The MAPEs indicate that both approaches provide local linear trends with a remarkably good fit.

The lower MAPE of 0.24% for the stability approach is consistent with the RMSE results.

The optimal value of 2.55 for β in the second estimation may seem quite high.

It ensures, however, that the growth rate is very responsive to unanticipated changes in the series.

We can extract the values of ℓ_t and b_t using

`fit.ausgdp.AAN.conv$state`.

A plot of the estimated growth rates, which is obtained by

```
fit.ausgdp.AAN.conv$state[,2]
```

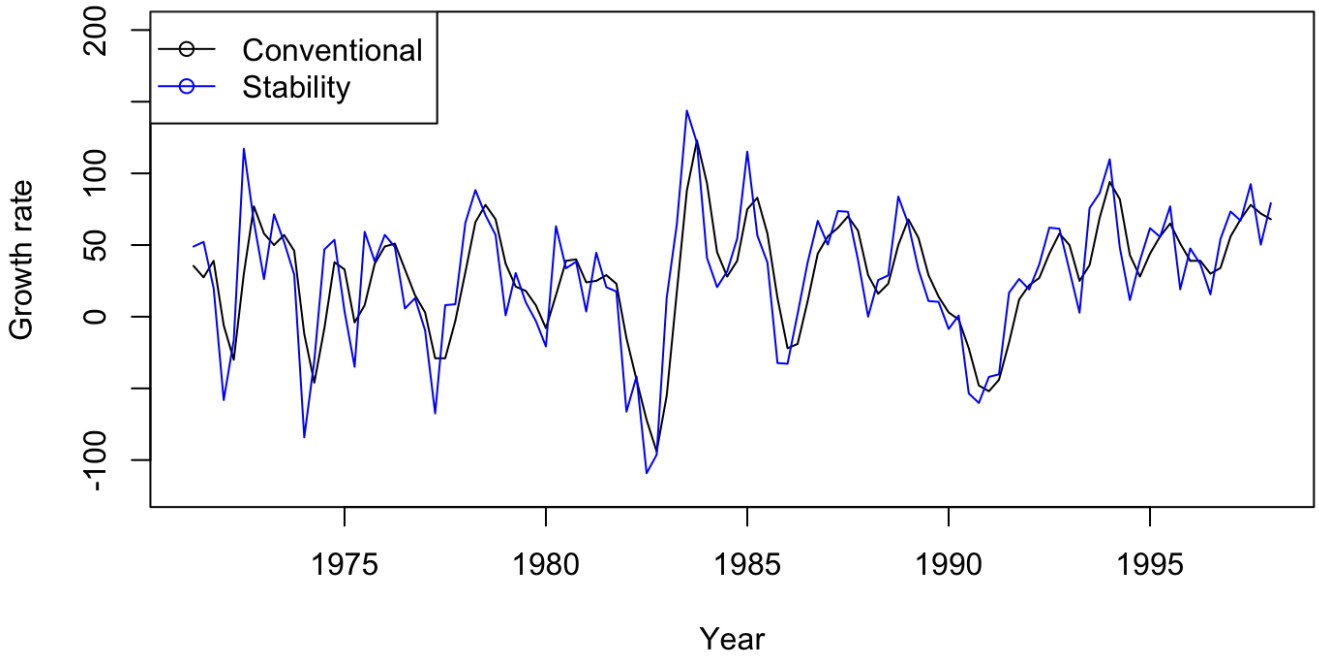
and

```
fit.ausgdp.AAN.stable$state[,2]
```

is shown in the next slide.

The effect is to ensure that the local trend adapts quickly to changes in the direction of the series values.

```
plot(fit.ausgdp.AAN.conv$state[,2],xlab="Year",ylab="Growth rate", ylim=c(-120,200))
lines(fit.ausgdp.AAN.stable$state[,2], col="blue", type="l")
legend("topleft",lty=1, pch=1, col=c("black","blue"), c("Conventional","Stability"))
```

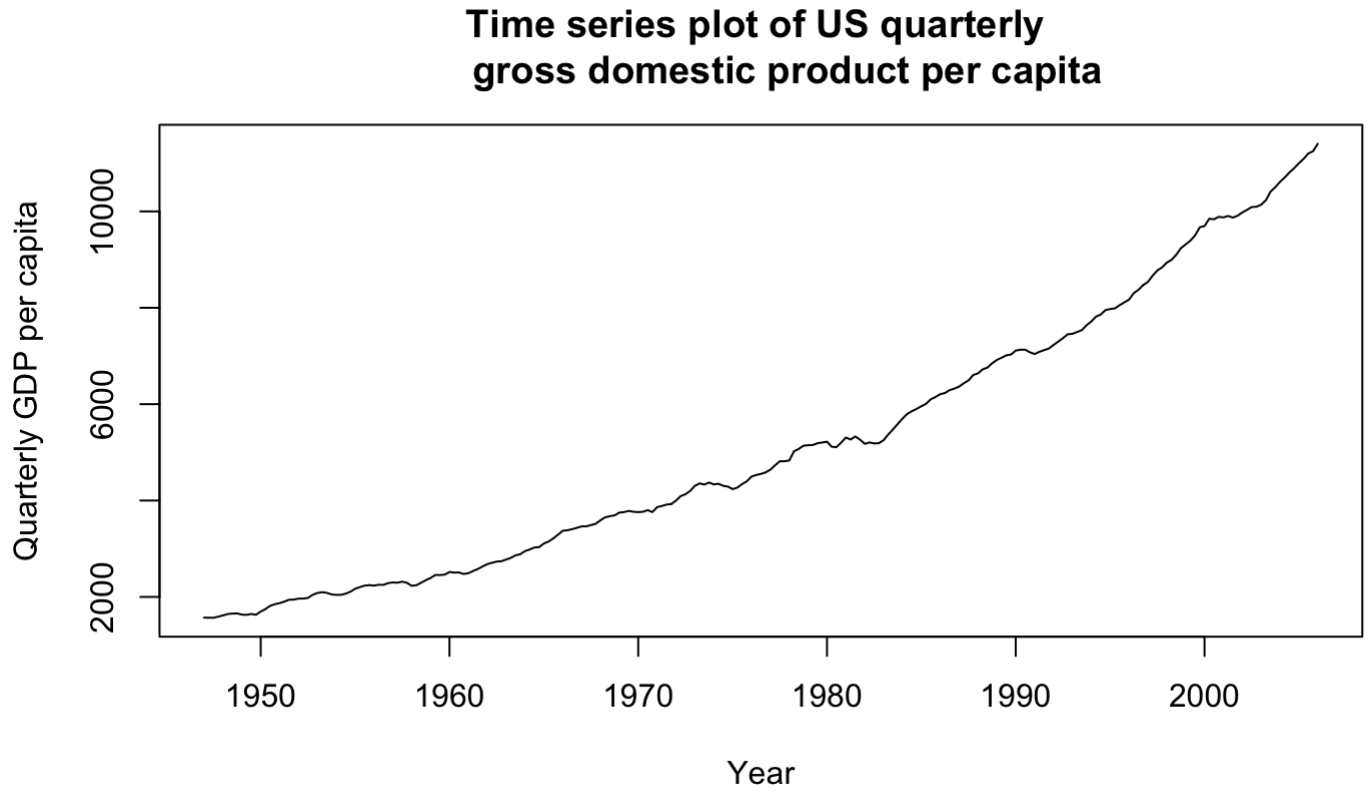


Application: United States GDP

We will consider quarterly US gross domestic product series available in the data set `usgdp`.

Time series plot of this series is as follows:

```
data("usgdp") # Available in expsmooth package
plot(usgdp,ylab = "Quarterly GDP per capita", xlab = "Year", main="Time series plot of US
     gross domestic product per capita")
```



Let’s fit the local level model ETS(A,N,N) to this series using stability conditions and minimizing the sum of squared errors criterion.

We can select the optimization criterion by

```
opt.crit=c("lik","amse","mse","sigma","mae")
```

argument.

So, it is possible to use one of `mse` (Mean Square Error), `amse` (Average MSE over first n forecast horizons), `sigma` (Standard deviation of residuals), `mae` (Mean of absolute residuals), or `lik` (Log-likelihood), which is the default.

```
fit.usgdp.ANN.mse <- ets(usgdp,"ANN" , damped=FALSE , opt.crit = "mse")
summary(fit.usgdp.ANN.mse)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = usgdp, model = "ANN", damped = FALSE, opt.crit = "mse")
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 1570.4706
##
##   sigma:  62.039
##
##      AIC     AICc      BIC
## 3256.481 3256.584 3266.885
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 41.49408 61.77664 49.6793 0.8282904 1.049518 0.2807741
##                   ACF1
## Training set 0.407214
```

```r
fit.usgdp.ANN.lik <- ets(usgdp,"ANN" , damped=FALSE)
summary(fit.usgdp.ANN.lik)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = usgdp, model = "ANN", damped = FALSE)
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 1570.4706
##
##   sigma:  62.039
##
##      AIC     AICc      BIC
## 3256.481 3256.584 3266.885
##
## Training set error measures:
##                    ME     RMSE     MAE       MPE     MAPE      MASE
## Training set 41.49408 61.77664 49.6793 0.8282904 1.049518 0.2807741
##                   ACF1
## Training set 0.407214
```

We got very similar results from both approaches.

Then we will fit the additive local level model ETS(A,N,N) with drift to the log-transformed data and the multiplicative local level model ETS(M,N,N) with drift.

```
fit.usgdp.ANN.drift <- ets(usgdp,"ANN" , beta = 0.0001,lambda = 0)
summary(fit.usgdp.ANN.drift)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = usgdp, model = "ANN", beta = 1e-04, lambda = 0)
##
##   Box-Cox transformation: lambda= 0
##
##   Smoothing parameters:
##     alpha = 0.9999
##
##   Initial states:
##     l = 7.3592
##
##   sigma:  0.013
##
##       AIC      AICc       BIC
## -759.0617 -758.9587 -748.6575
##
## Training set error measures:
##                    ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 41.49393 61.77667 49.67926 0.8282795 1.049515 0.2807739
##                    ACF1
## Training set 0.4072155
```

```
fit.usgdp.MNN.drift <- ets(usgdp,"MNN", beta = 0.0001,bounds="admiss")
summary(fit.usgdp.MNN.drift)
```

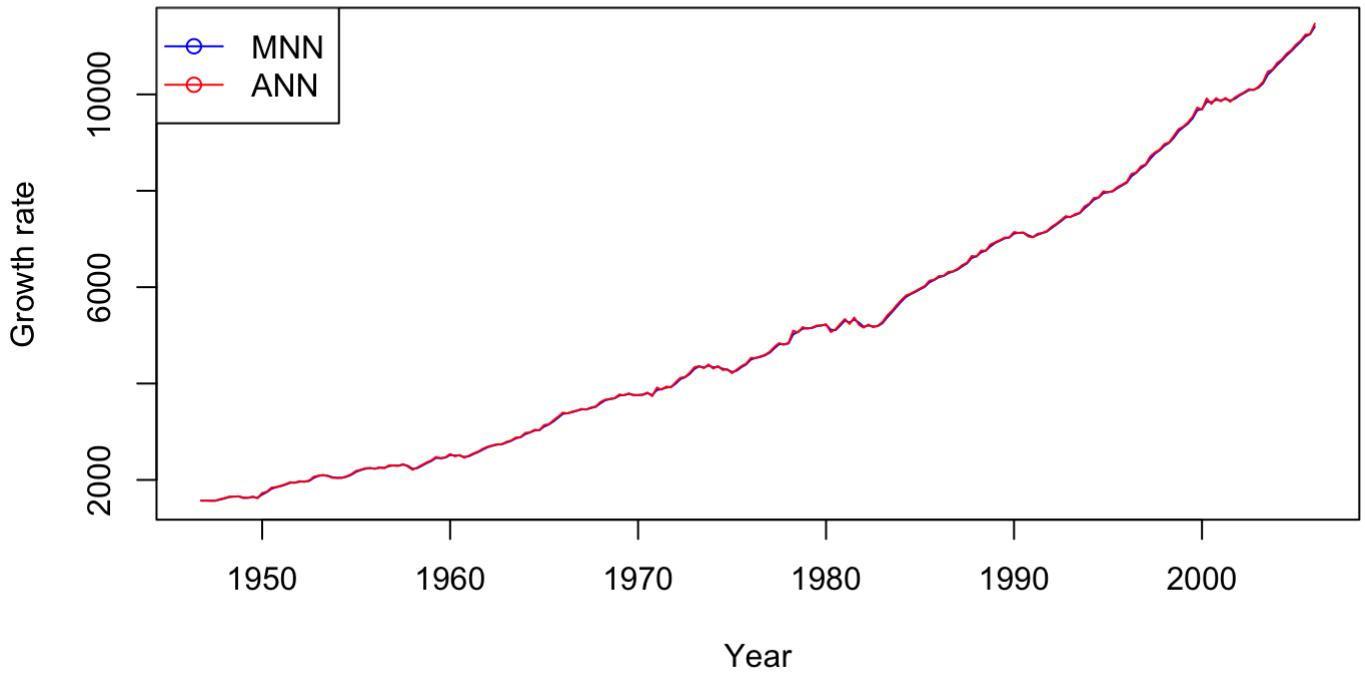
```
## ETS(M,N,N)
##
## Call:
##  ets(y = usgdp, model = "MNN", beta = 1e-04, bounds = "admiss")
##
##   Smoothing parameters:
##     alpha = 1.4162
##
##   Initial states:
##     l = 1569.7058
##
##   sigma:  0.0115
##
##      AIC     AICc      BIC
## 3157.575 3157.678 3167.979
##
## Training set error measures:
##                    ME    RMSE      MAE       MPE      MAPE      MASE
## Training set 29.4863 53.0689 41.97299 0.5887393 0.8986006 0.2372201
##                     ACF1
## Training set -0.1126863
```

According to MASE, the multiplicative local level model ETS(M,N,N) with drift performs better.

It also performs better than the additive versions of this model.

A plot of the estimated growth rates from ETS(A,N,N) with drift model is given below:

```
plot(exp(fit.usgdp.ANN.drift$state),xlab="Year",ylab="Growth rate", col = "blue")
lines(fit.usgdp.MNN.drift$state, col="red", type="l")
legend("topleft",lty=1, pch=1, col=c("blue", "red"), c("MNN","ANN"))
```



Prediction Distributions and Intervals

There are several sources of uncertainty when forecasting a future value of a time series (Chatfield 1993):

- The uncertainty in model choice; maybe another model is correct, or maybe none of the candidate models is correct.
- The uncertainty in the future innovations $\epsilon_{n+1}, \epsilon_{n+2}, \ldots, \epsilon_{n+h}$.
- The uncertainty in the parameter estimates $\alpha, \beta, \gamma, \phi$, and $\boldsymbol{X}_0$.

The prediction distribution and intervals should take all of these into account.

However, this is a difficult problem, and in most time series analysis only the uncertainty in the future innovations is taken into account.

We define the prediction distribution as the distribution of a future value of the series given the model.

The mean of the prediction distribution is called the forecast mean and is denoted by $\mu_{n+h|n} = E(Y_{n+h}|\boldsymbol{X}_n)$.

The corresponding forecast variance is given by $\nu_{n+h|n} = V(Y_{n+h}|\boldsymbol{X}_n)$.

These quantities are important in terms of the quality of our future predictions.

The most direct method of obtaining prediction distributions is to simulate many possible future sample paths from the fitted model and to estimate the distributions from the simulated data.

The drawback of this approach is that we cannot have any algebraic equations for the prediction distributions; hence, predictions will be available only numerically.

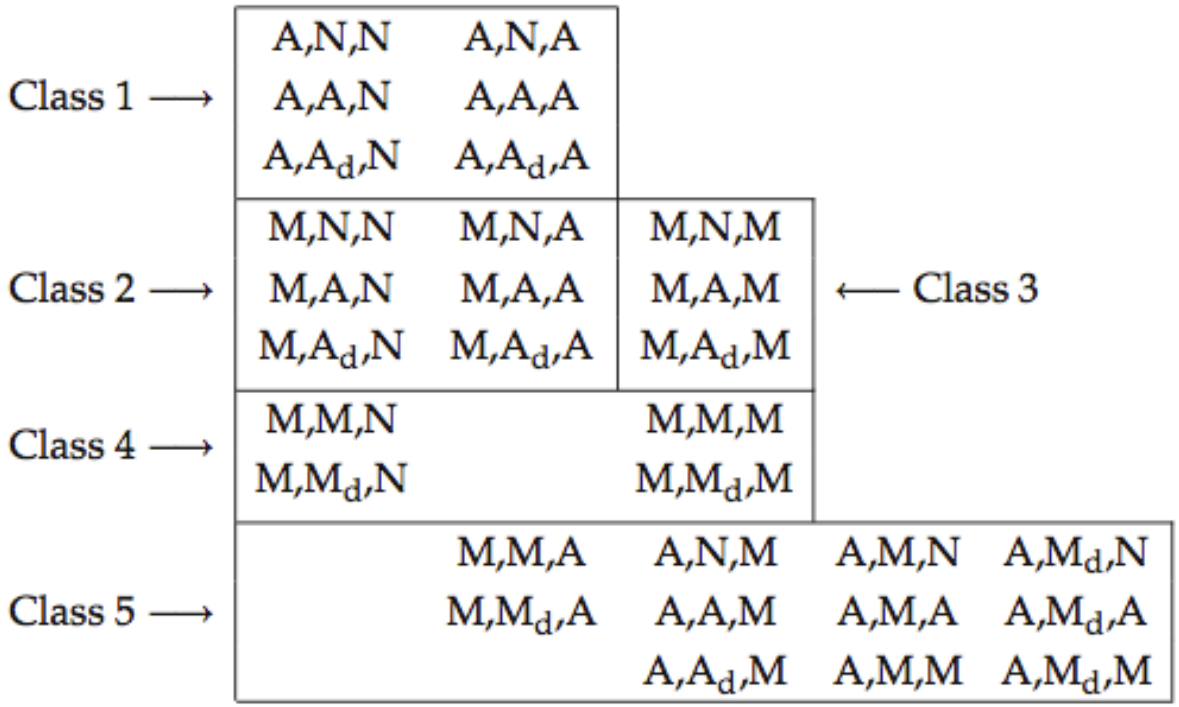
An alternative approach is to derive the distributions analytically.

Analytic results on prediction distributions can provide additional insight and can be much quicker to compute.

These results are relatively easy to derive for some models (particularly the linear models) but very difficult for others.

In this module, we will mainly focus on the use of simulations to derive prediction intervals.

However, the following classification of the innovations state-space models is very useful in term of determination of suitable models according to the characteristics of the series.



* Taken from: Hyndman et al., Forecasting with Exponential Smoothing: The State Space Approach. Springer, 2008.

The models in each class are

- Class 1: Linear homoscedastic state-space models
- Class 2: Linear heteroscedastic state-space models
- Class 3: Some nonlinear seasonal state-space models
- Class 4: Models with multiplicative errors, multiplicative trend, and either no seasonality or multiplicative seasonality
- Class 5: Renaming models

Simulated Prediction Distributions and Intervals

Recall that the general model involves a state vector $\boldsymbol{X}_t = (\ell_t, b_t, s_t, s_{t-1}, \ldots, s_{t-m+1})'$ and state space equations of the form

$$\begin{aligned} Y_t &= w(\boldsymbol{X}_{t-1}) + r(\boldsymbol{X}_{t-1})\epsilon_t, \\ \boldsymbol{X}_t &= \boldsymbol{f}(\boldsymbol{X}_{t-1}) + \boldsymbol{g}(\boldsymbol{X}_{t-1})\epsilon_t, \end{aligned} \tag{6}$$

where $\omega(\cdot)$ an $r(\cdot)$ are scalar functions and $\boldsymbol{f}(\cdot)$ and $\boldsymbol{g}(\cdot)$ are vector functions, and ϵ_t is a white noise process with variance σ^2.

One simple approach to obtain the prediction distribution is to simulate sample paths from the models, conditional on the final state $\boldsymbol{X}_n$.

We apply the following simple algorithm to run the simulations:

1. Set $i = 1$.
2. For $t = n + 1, \ldots, n + h$, generate ϵ_t from a random number generator assuming a Gaussian or other appropriate distribution.
3. Generate observations $\{Y_t^i\}$, using ϵ_t generated at the Step 2 starting with $\boldsymbol{X}_n$ over the model of interest.
4. Save the generated series to the ith column of simulation matrix $\boldsymbol{A}_{M\times(n+h)}$.
5. Set $i = i + 1$ and go to the Step 2 until the condition $i > M$ is satisfied. Here M is a large integer.

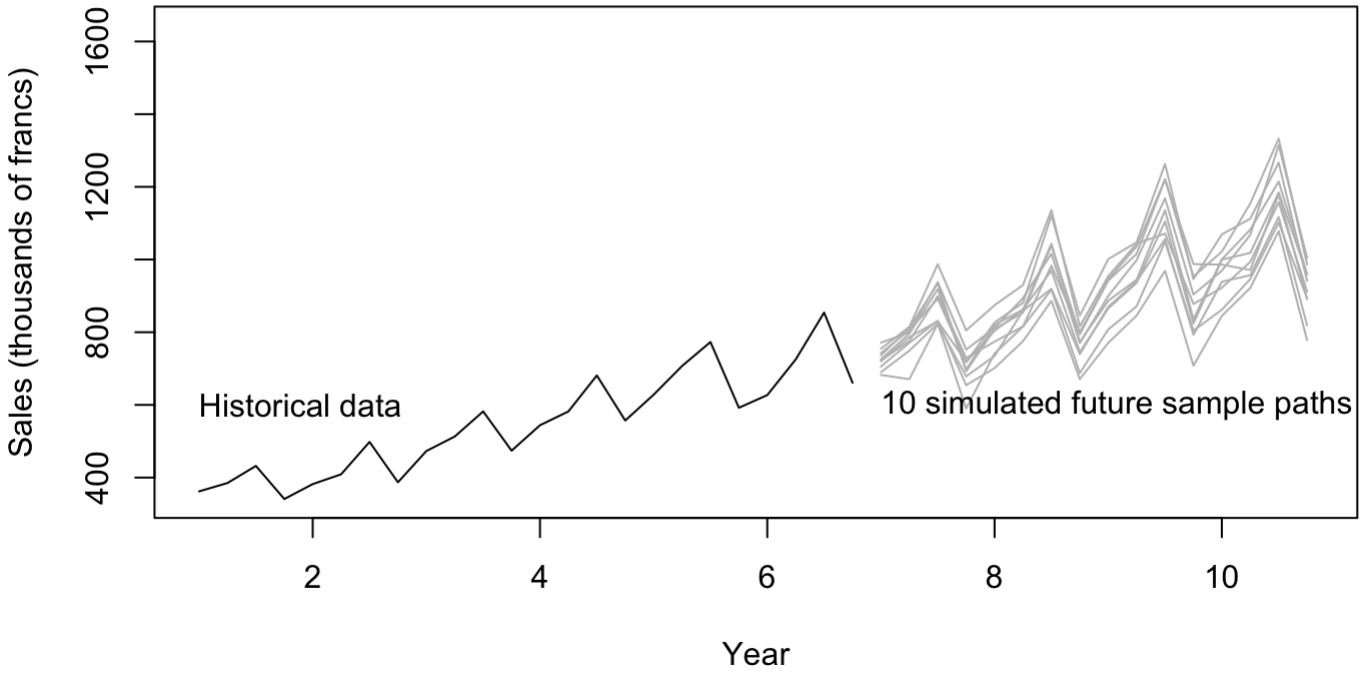
Let's explain this algorithm over the quarterly exports of a French company (in thousands of francs) series available in `frexport` dataset of `expsmooth` package.

We will use ETS(M,A,M) model in the simulation. Which means that the model of interest is ETS(M,A,M).

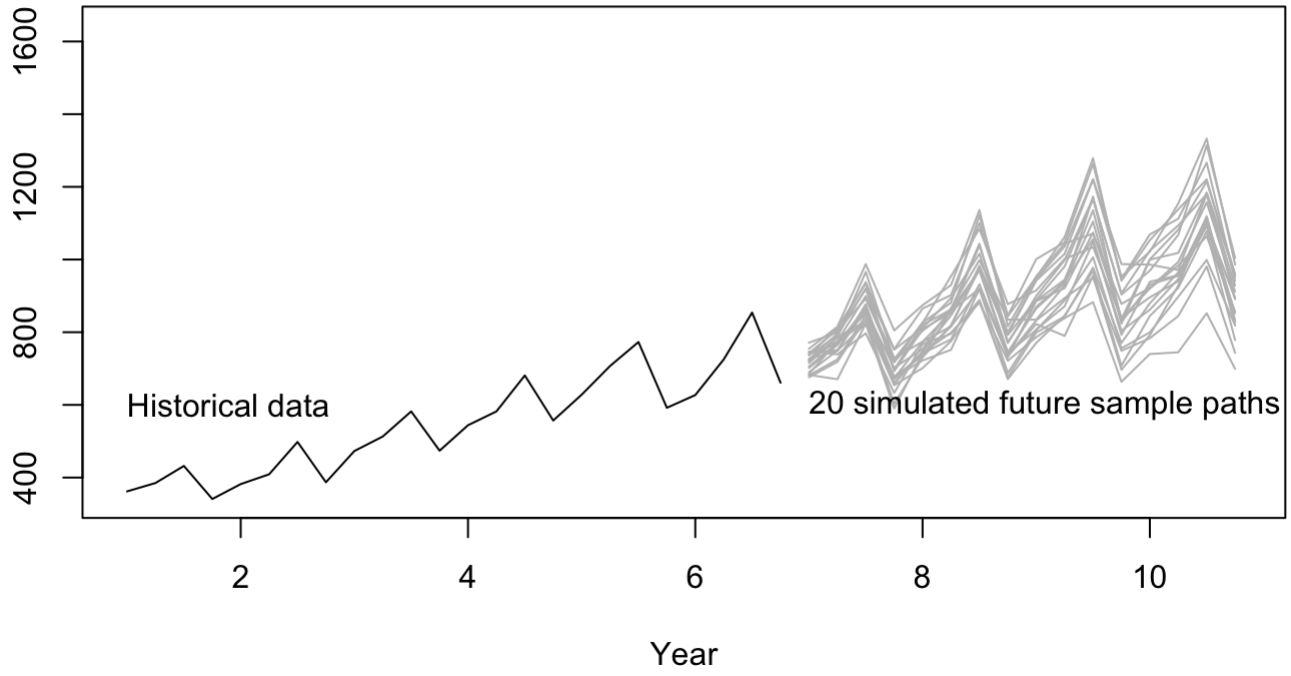
```
data("frexport") # Available in expsmooth package
A = ts(matrix(NA,16,5000),s=7,f=4)
fit.fre.MAM <- ets(frexport,model="MAM",damped=FALSE,alpha=0.8185)
n = length(frexport)
h = 10
M = 5000
for (i in 1:M){
  A[,i] = simulate(fit.fre.MAM,initstate=fit.fre.MAM$states[25,],nsim=16)
  # Generate random epsilons and apply model formulation
}
```

Then we can plot the generated forecasts.

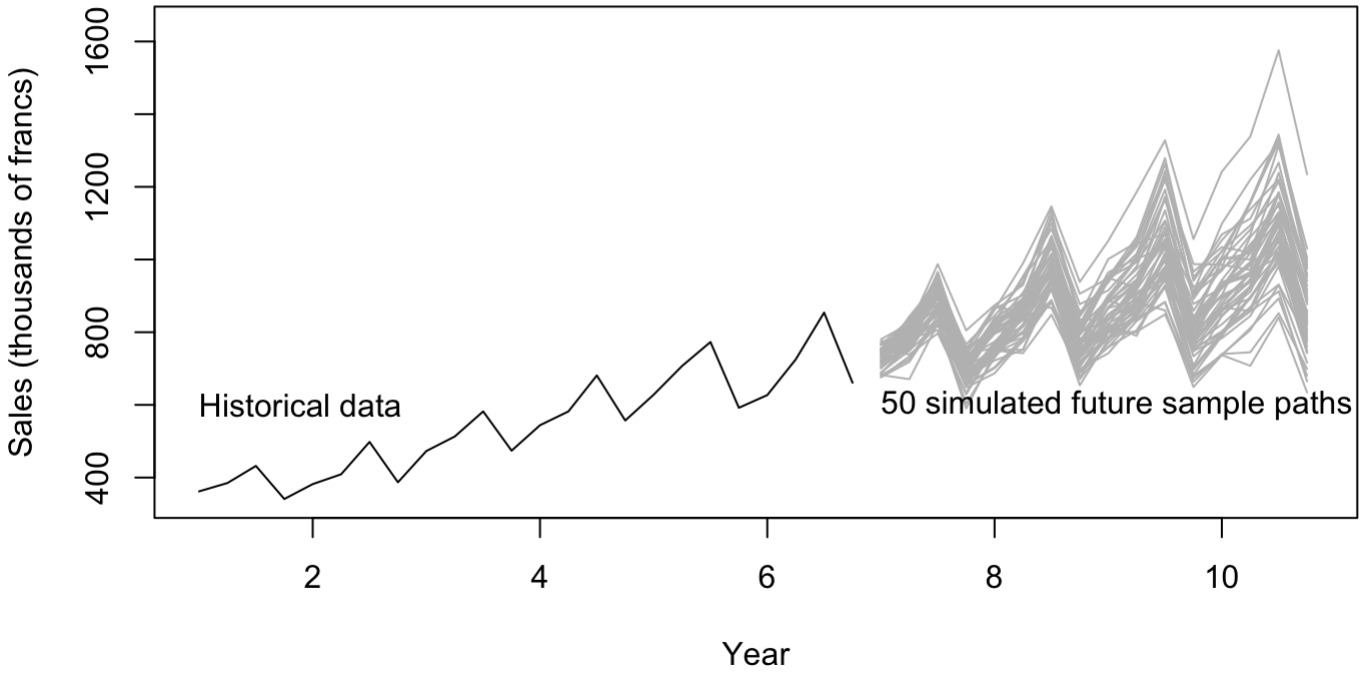
```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),ylab="Sales (thousands of francs)",xla
for(i in 1:10){
  lines(A[,i],col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"10 simulated future sample paths",adj=0)
```


Sales (thousands of francs)
400
800
1200
1600
Historical data
10 simulated future sample paths
2
4
6
8
10
Year

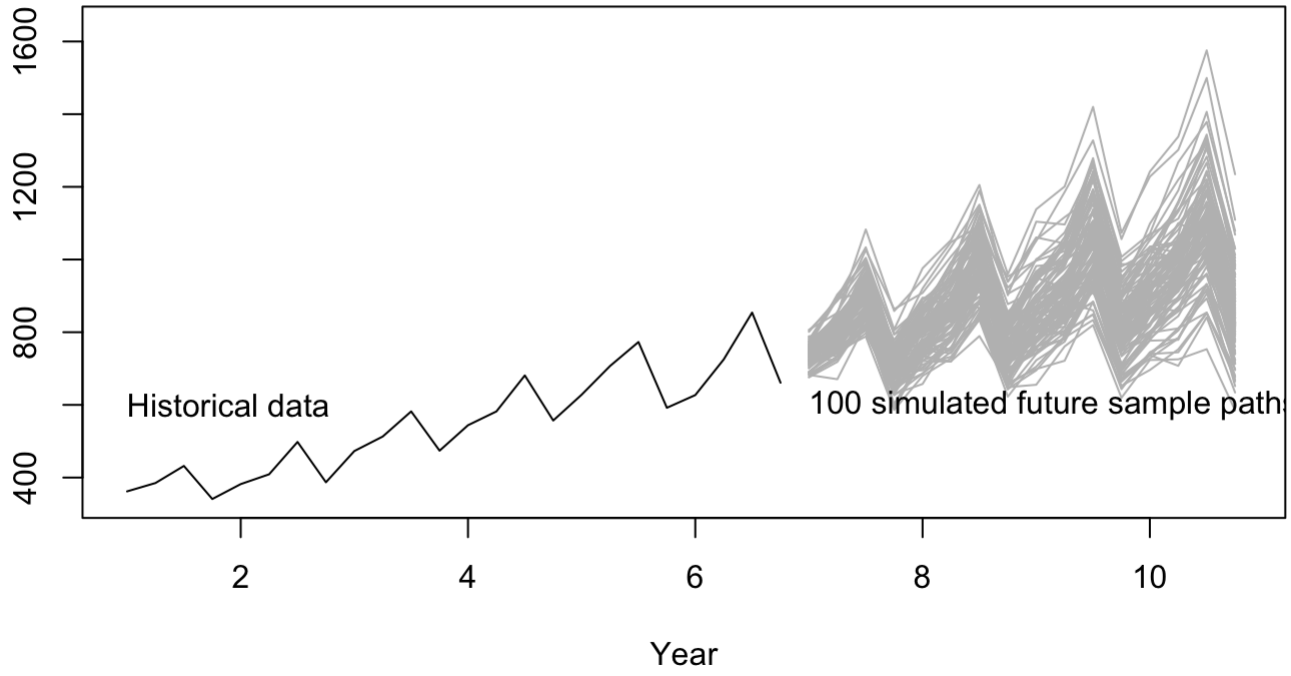
```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),ylab=NULL,xlab="Year")
for(i in 1:20){
  lines(A[,i],col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"20 simulated future sample paths",adj=0)
```


1600
1200
800
400
Historical data
20 simulated future sample paths
2
4
6
8
10
Year

```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),ylab="Sales (thousands of francs)",xla
for(i in 1:50){
  lines(A[,i],col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"50 simulated future sample paths",adj=0)
```


Sales (thousands of francs)
400
800
1200
1600
Historical data
50 simulated future sample paths
2
4
6
8
10
Year

```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),ylab=NULL,xlab="Year")
for(i in 1:100){
  lines(A[,i],col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"100 simulated future sample paths",adj=0)
```

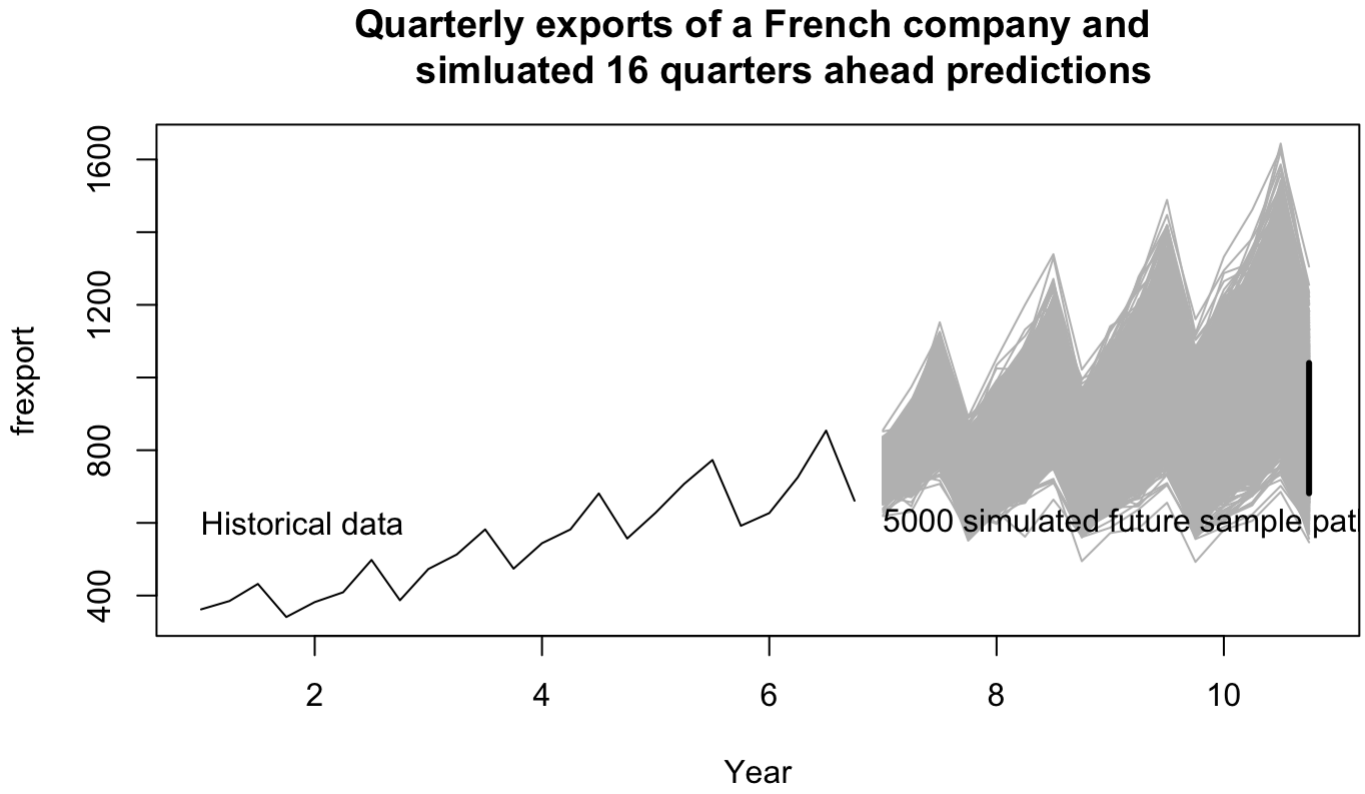

1600
1200
800
400
Historical data
100 simulated future sample path
2
4
6
8
10
Year

When we plot the generated sample paths over enough number of simulations, we eventually get a picture of the distribution of predictions on the time series plot.

The following plot displays all of the generated sample paths.

```
pi = quantile(A[16,],type=8,prob=c(.05,.95))
```

```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),xlab="Year", main="Quarterly exports c
     simluated 16 quarters ahead predictions")
for(i in 1:5000){
  lines(A[,i],col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"5000 simulated future sample paths",adj=0)
lines(c(10.75,10.75),pi,lwd=3,col=1)
```


Quarterly exports of a French company and
simluated 16 quarters ahead predictions
frexport
400
800
1200
1600
Historical data
5000 simulated future sample pat
2
4
6
8
10
Year

As expected the variation of the predictions increases when we get far from the last observation.

This gives us an insight into the dispersion of the predictions.

We can find the desired $(1-\alpha)$ confidence bounds empirically using the randomly generated sample paths.

Suppose we want to find 95% limits. We will find empirical quantiles of the generated distribution of the predictions and use them as Monte Carlo estimates of 95% confidence limits.

To find the limits for the last forecast, we use the following code chunk:

```
pi = quantile(A[16,],type=8,prob=c(.05,.95))
pi
```

```
##        5%       95%
##  681.6492 1039.9190
```

The thick line at the far right edge of the plot above shows the $[5$ limits of the last forecast.

This corresponds to a 90% confidence interval.

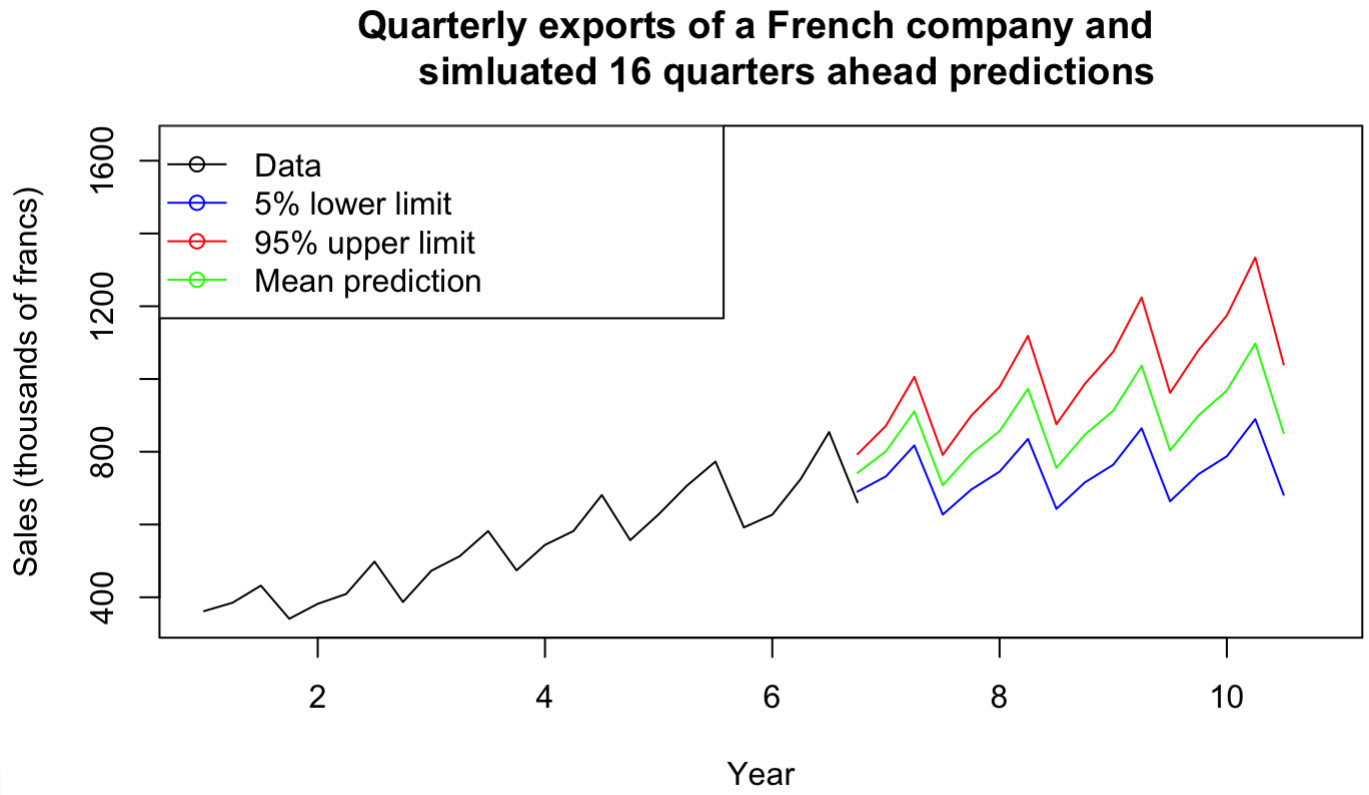
Characteristics of the prediction distribution of $Y_{n+h|n}$ can then be estimated from the simulated values at a specific forecast horizon: $Y_{n+h|n} = \{Y_{n+h}^{(1)}, \ldots, Y_{n+h}^{(M)}\}$. For example, prediction intervals can be obtained using quantiles of the simulated sample paths.

An approximate $100(1-\alpha)\%$ prediction interval for forecast horizon h is given by the $\alpha/2$ and $1-\alpha/2$ quantiles of $\boldsymbol{Y}_{n+h|n}$.

When we apply this approach for all prediction points, we produce the desired confidence limits of predictions.

```
# Define the variables to put quantiles and sample average
Pi = array(NA, dim=c(16,2))
avrg = array(NA, 16)
# Calcualte the interval estimates and mid point
for (i in 1:16){
  Pi[i,] = quantile(A[i,],type=8,prob=c(.05,.95))
  avrg[i] = mean(A[i,]) # This would be median as well
}
# Create ts objects for plotting
Pi.lb = ts(Pi[,1],start=end(frexport),f=4)
Pi.ub = ts(Pi[,2],start=end(frexport),f=4)
avrg.pred = ts(avrg,start=end(frexport),f=4)
```

```
plot(frexport,xlim=c(1,10.8),ylim=range(frexport,A),ylab="Sales (thousands of francs)",xlab="Year", m
     simluated 16 quarters ahead predictions")
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width = 4,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```

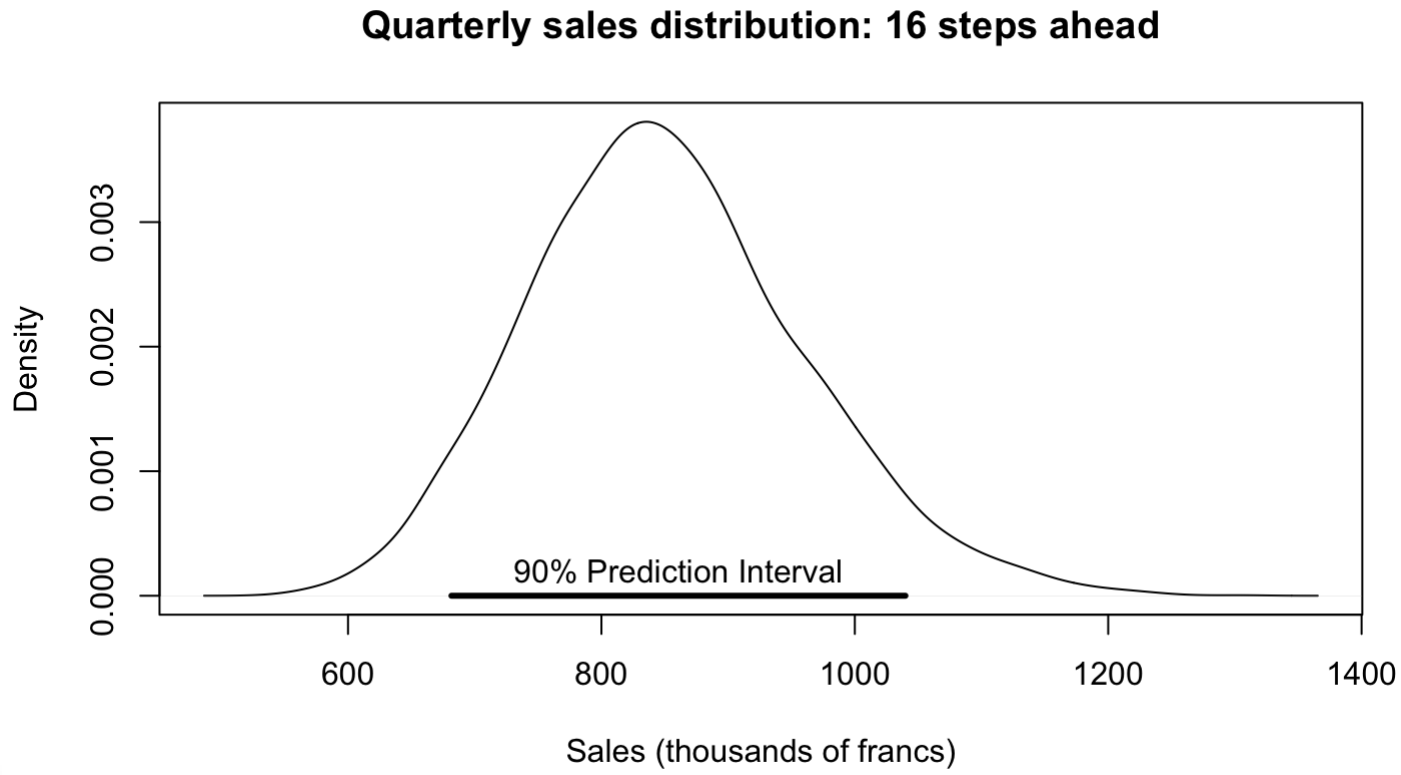


The full prediction density can be estimated using a kernel density estimator applied to $\boldsymbol{Y}_{n+h|n}$.

The function `density()` is used to apply kernel density estimation to create probability density function.

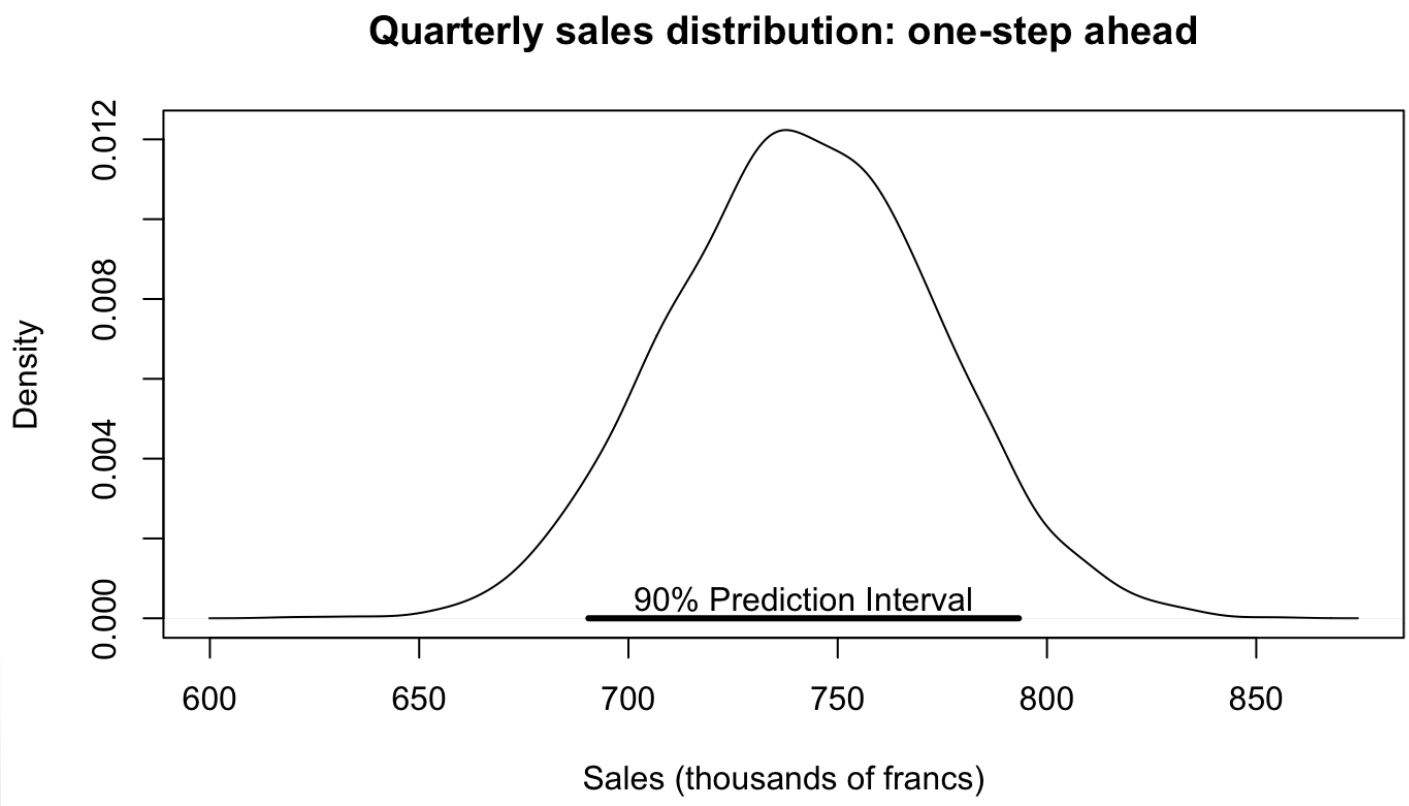
The following code chunk generates the pdf plot of the prediction density for the last forecast point.

```
plot(density(A[16,],bw="SJ"),main="Quarterly sales distribution: 16 steps ahead",xlab="Sal
lines(pi,c(0,0),lwd=3,col=1)
text(mean(pi),0.0002,"90% Prediction Interval")
```



For the prediction distribution of one-step ahead, we run the following code chunk:

```
pi.1 = quantile(A[1,],type=8,prob=c(.05,.95))
plot(density(A[1,],bw="SJ"),main="Quarterly sales distribution: one-step ahead",xlab="Sales (thousand
lines(pi.1,c(0,0),lwd=3,col=1)
text(mean(pi.1),0.0005,"90% Prediction Interval")
```



We got nearly symmetric distributions supporting the normal distribution assumption on the errors.

Lead-Time Forecasting

In inventory control, forecasts of the sum of the next h observations are often required.

These are used for determination of ordering requirements such as reorder levels, order-up-to levels and reorder quantities.

Suppose that a replenishment decision is to be made at the beginning of period $n + 1$.

Any order placed at this time is assumed to arrive a lead-time later, at the start of period $n + h + 1$.

Thus, we need to forecast the aggregate of unknown future values Y_{n+j}, defined by

$$Y_n(h) = \sum_{j=1}^{h} Y_{n+j}. \tag{7}$$

The problem is to make inferences about the distribution of $Y_n(h)$ which (in the inventory context) is known as the “lead-time demand.”

The results from the simulation of single periods give the prediction distributions and intervals for individual forecast horizons, but for re-ordering purposes, it is more useful to have the lead-time prediction distribution and interval.

Because $Y_n(h)$ involves a summation, the central limit theorem states that its distribution will tend towards Gaussianity as h increases.

However, for small to moderate h, we need to estimate the distribution.

The simulation approach can easily be used here by computing values of $Y_n(h)$ from the simulated future sample paths.

For example, to get the distribution of $Y_n(i)$ for the quarterly French exports data, we sum the first i values of the simulated future sample paths.

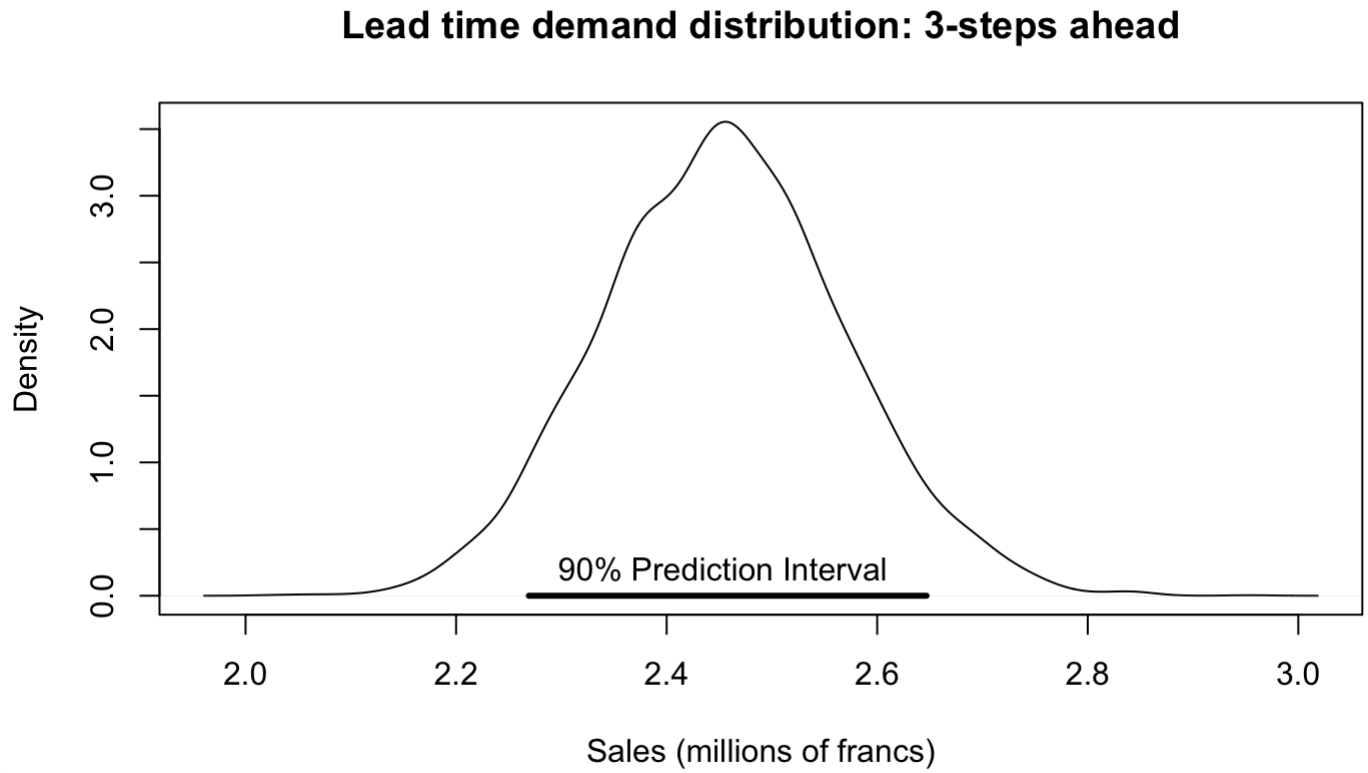
This provides us 5000 values from the distribution of $Y_n(i)$.

But notice that here we assume the model is correct.

The following figure shows the density of $Y_n(3)$ computed from these 5000 values along with a 90% prediction interval.

```r
ltd <- colSums(A[1:3,])
ltd.den <- density(ltd/1e3,bw="SJ")

plot(ltd.den,main="Lead time demand distribution: 3-steps ahead",xlab="Sales (millions of francs)")
text(median(ltd)/1e3,0.2,"90% Prediction Interval")
lines(quantile(ltd/1e3,type=8,prob=c(.05,.95)),c(0,0),lwd=3,col=1)
```



Here we have assumed that the lead-time h is fixed.

Fixed lead-times are relevant when suppliers make regular deliveries, an increasingly common situation in supply chain management.

For stochastic lead-times, we could randomly generate h from a Poisson or some other count distribution when simulating values of $Y_n(h)$.

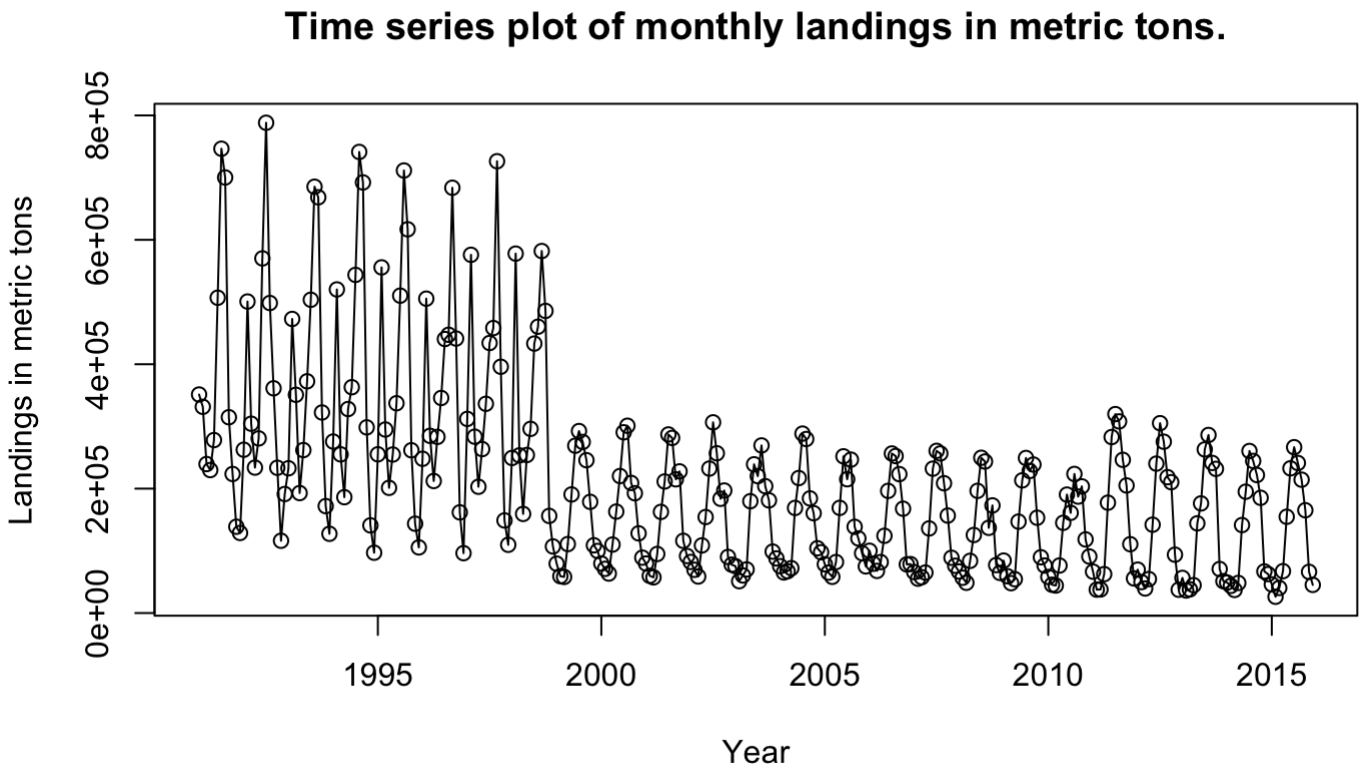
This would be used when suppliers make irregular deliveries.

Practical Application

In this practical application, we will revisit the monthly commercial landings series of US-NMFS and find prediction distributions.

```
NMFS_Landings = read.csv("~/Documents/MATH1307_Forecasting/presentations/Module 8/NMFS_Lan
NMFS_Landings.ts = matrix(NMFS_Landings$Metric_Tons, nrow = 25, ncol = 12)
NMFS_Landings.ts = as.vector(t(NMFS_Landings.ts))
NMFS_Landings.ts = ts(NMFS_Landings.ts,start=c(1991,1), end=c(2015,12), frequency=12)
```

```
plot(NMFS_Landings.ts,ylab='Landings in metric tons',xlab='Year',type='o', main = "Time se
```



There is an intervention effect in this series. This intervention creates a kind of trend in the series.

There is also changing variance and seasonality present in this series.

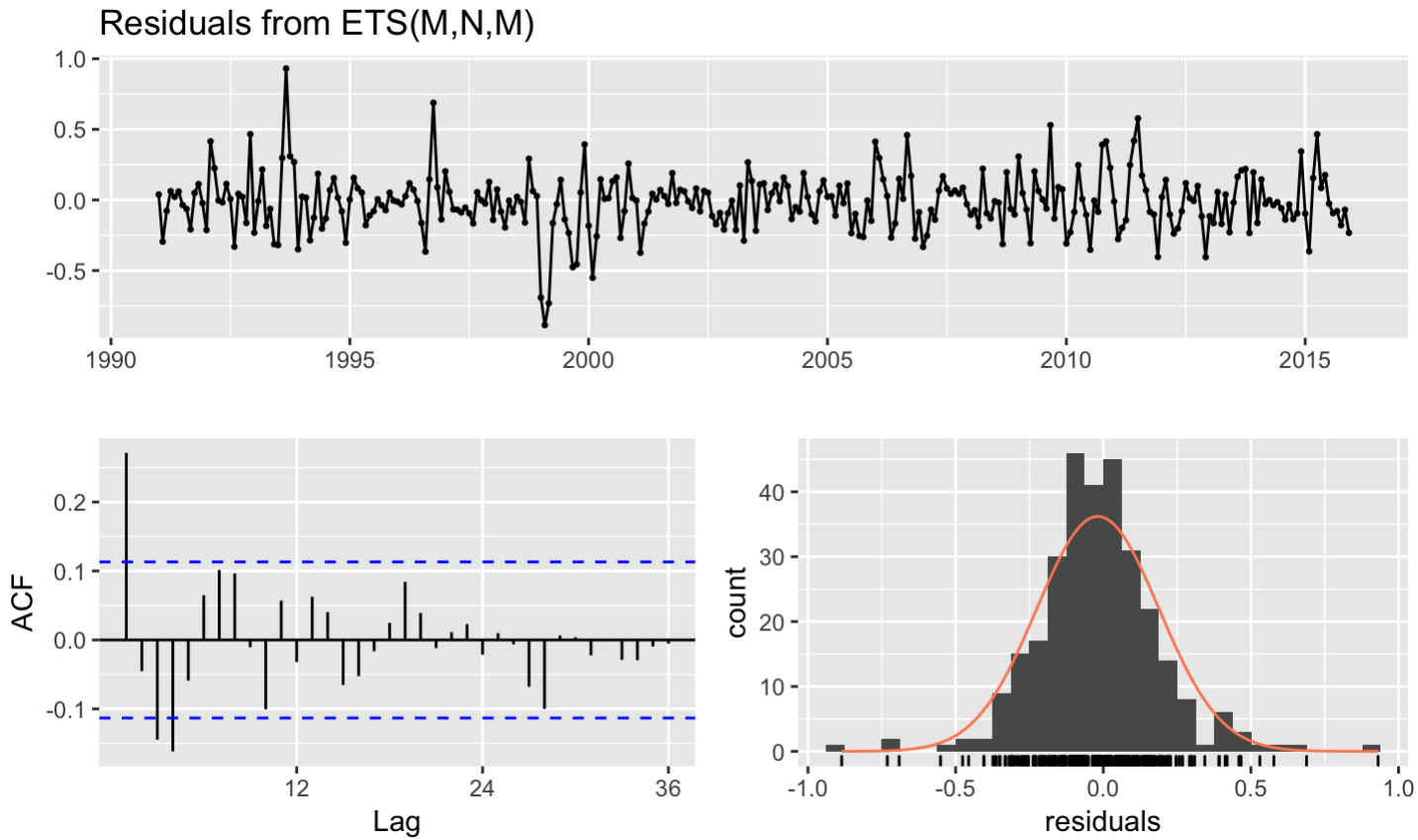
Therefore, we will go for multiplicative models and consider the existence and absence of a trend in the series.

First, we will find the best model and then apply the algorithm to find the prediction intervals.

```
fit.MNM = ets(NMFS_Landings.ts, model = "MNM")
summary(fit.MNM)
```

```
## ETS(M,N,M)
##
## Call:
##  ets(y = NMFS_Landings.ts, model = "MNM")
##
##   Smoothing parameters:
##     alpha = 0.1057
##     gamma = 0.6649
##
##   Initial states:
##     l = 377460.1978
##     s = 0.3622 0.3468 0.5954 1.0874 2.0307 2.0939
##            1.302 0.7424 0.5941 0.7073 1.2416 0.8963
##
##   sigma:  0.2126
##
##      AIC     AICc      BIC
## 8018.969 8020.659 8074.526
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -7215.472 60009.13 32691.13 -9.238586 19.62959 0.9093588
##                    ACF1
## Training set 0.2986509
```

```
checkresiduals(fit.MNM)
```

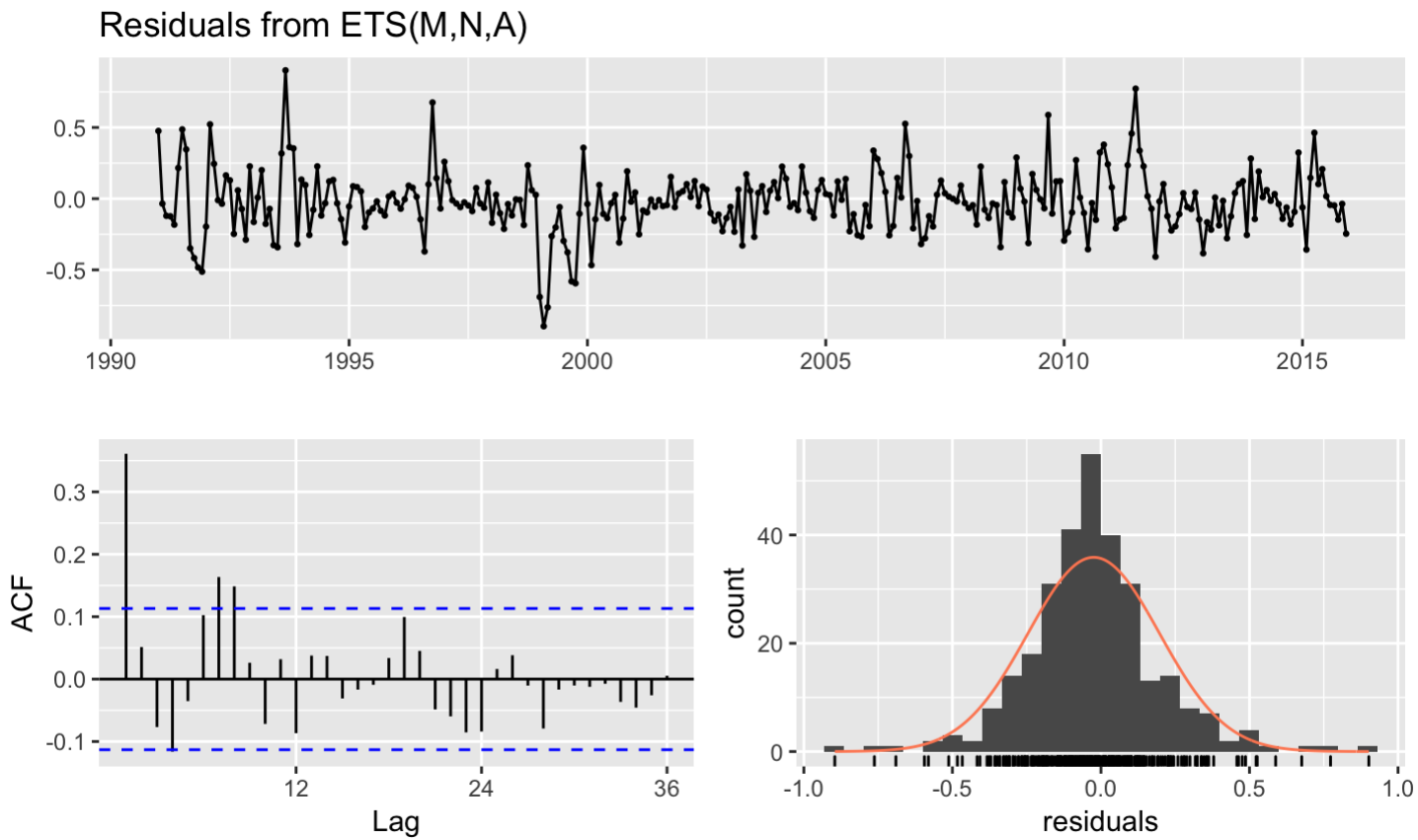



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,M)
## Q* = 57.71, df = 10, p-value = 9.804e-09
##
## Model df: 14.   Total lags used: 24
```

```
fit.MNA = ets(NMFS_Landings.ts, model = "MNA")
summary(fit.MNA)
```

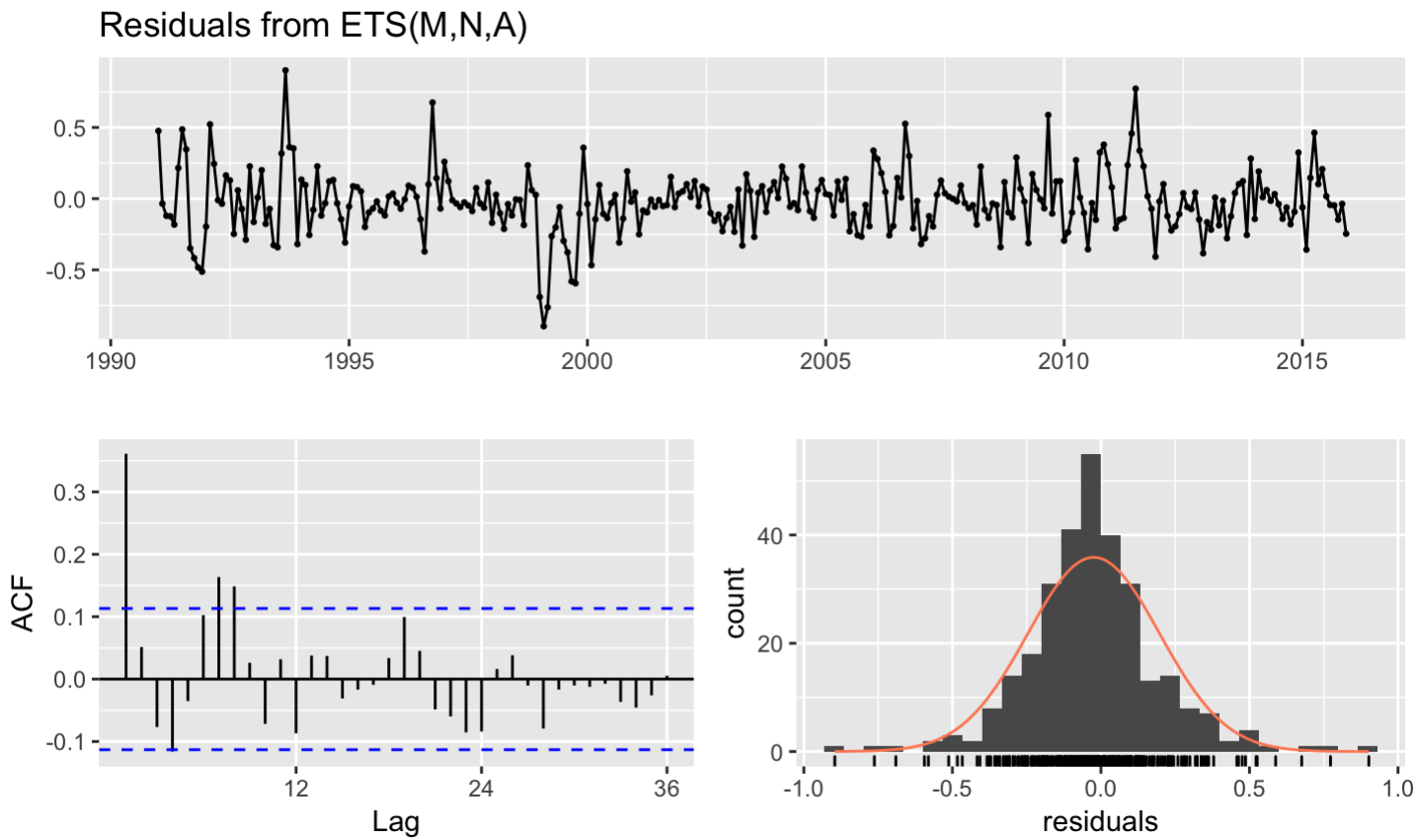
```
## ETS(M,N,A)
##
## Call:
##  ets(y = NMFS_Landings.ts, model = "MNA")
##
##   Smoothing parameters:
##     alpha = 0.0175
##     gamma = 0.797
##
##   Initial states:
##     l = 355466.9382
##     s = -91893.08 -89774.16 22300.54 119250.1 159324.5 145543.2
##            61951.35 -16046.16 -94408.25 -84984.9 -13895.79 -117367.3
##
##   sigma:  0.2285
##
##      AIC     AICc      BIC
## 8068.568 8070.259 8124.125
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE
## Training set -8946.545 69762.19 37768.98 -11.19528 21.84087 1.050608
##                    ACF1
## Training set 0.4058144
```

```
checkresiduals(fit.MNA)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,A)
## Q* = 81.612, df = 10, p-value = 2.424e-13
##
## Model df: 14.   Total lags used: 24
```

```
checkresiduals(fit.MNA)
```

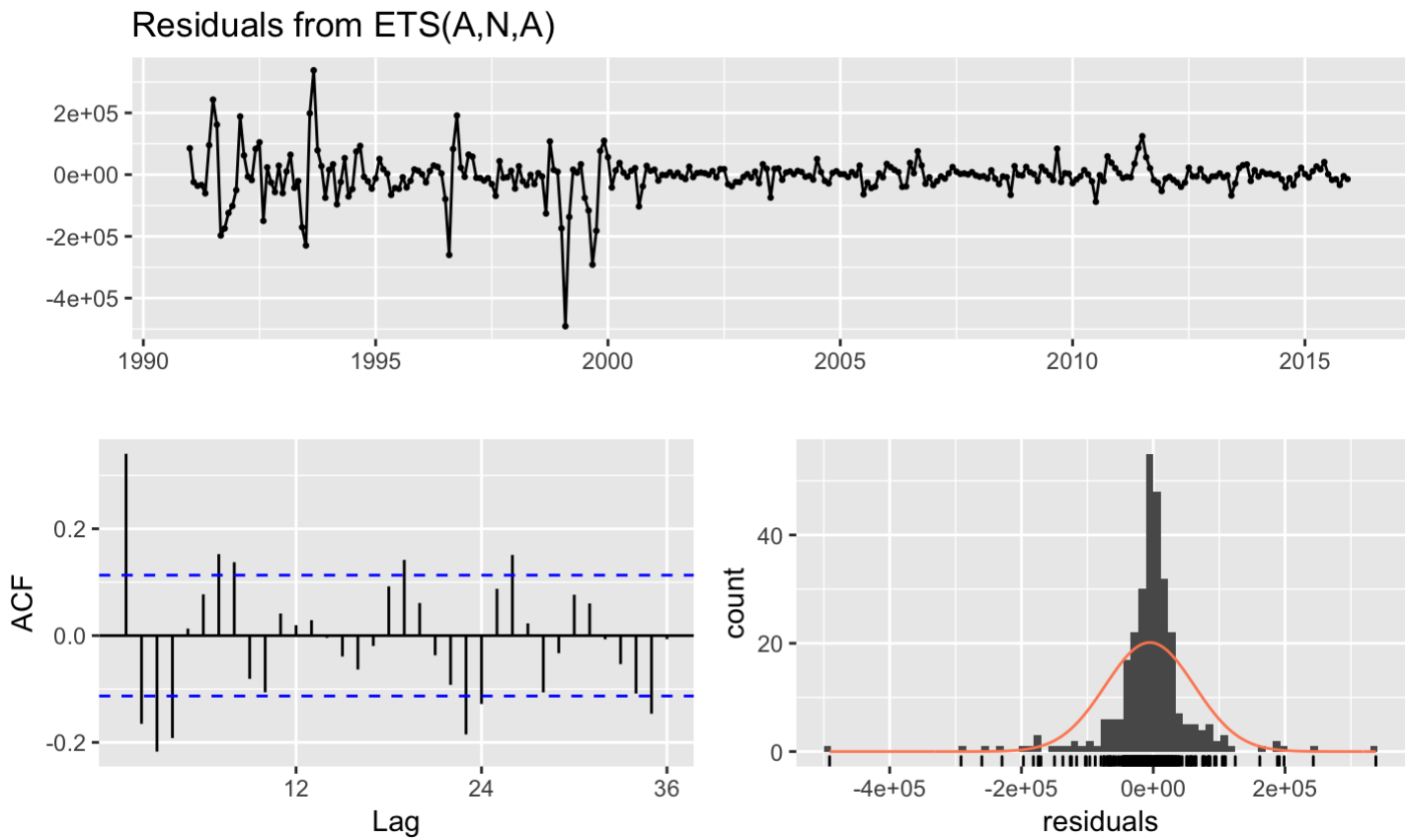



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,N,A)
## Q* = 81.612, df = 10, p-value = 2.424e-13
##
## Model df: 14.   Total lags used: 24
```

```r
fit.ANA = ets(NMFS_Landings.ts, model = "ANA")
summary(fit.ANA)
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = NMFS_Landings.ts, model = "ANA")
##
##   Smoothing parameters:
##     alpha = 0.0862
##     gamma = 0.7165
##
##   Initial states:
##     l = 355466.7851
##     s = -119503.1 -97709.24 22300.22 119250.1 159880.2 145821.3
##            61951.1 -16046.07 -94408.57 -84985.32 -7033.282 -89517.39
##
##   sigma:  69219.72
##
##      AIC     AICc      BIC
## 8413.822 8415.512 8469.379
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE     MASE
## Training set -5202.572 67585.3 37997.28 -8.132141 22.19257 1.056958
##                    ACF1
## Training set 0.340448
```

```
checkresiduals(fit.ANA)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,N,A)
## Q* = 122.59, df = 10, p-value < 2.2e-16
##
## Model df: 14.   Total lags used: 24
```

As expected due to the existence of changing variance, according to both BIC and MASES, we get better fit with MNM model, where we have multiplicative errors and multiplicative seasonality.

Additive models did not give promising results in terms of both BIC and MASE.

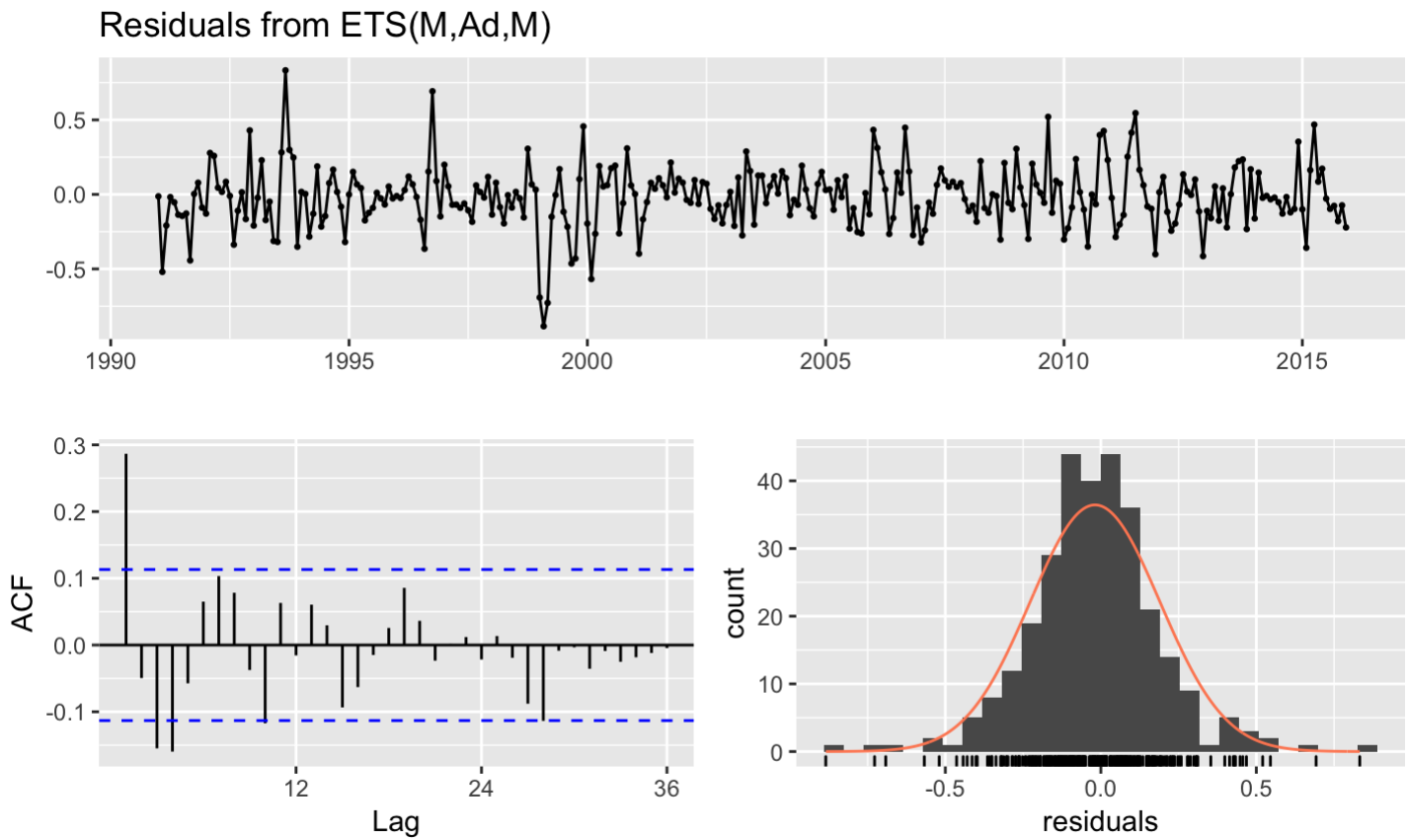
But we have three significant lag in the ACF of residuals. So there is still series correlation left in the residuals.

Now we will include the trend in the models.

```
fit.MAM = ets(NMFS_Landings.ts, model = "MAM")
summary(fit.MAM)
```

```
## ETS(M,Ad,M)
##
## Call:
##  ets(y = NMFS_Landings.ts, model = "MAM")
##
##   Smoothing parameters:
##     alpha = 0.1127
##     beta  = 0.0013
##     gamma = 0.6237
##     phi   = 0.9737
##
##   Initial states:
##     l = 466109.4749
##     b = 2111.5816
##     s = 0.3507 0.3229 0.5604 1.3525 1.8996 2.0355
##            1.3523 0.6746 0.5386 0.6822 1.4702 0.7604
##
##   sigma:  0.2155
##
##      AIC     AICc      BIC
## 8030.824 8033.258 8097.492
##
## Training set error measures:
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -9205.337 64429.85 34720.17 -9.477811 20.15859 0.9657999
##                    ACF1
## Training set 0.3188317
```

```
checkresiduals(fit.MAM)
```

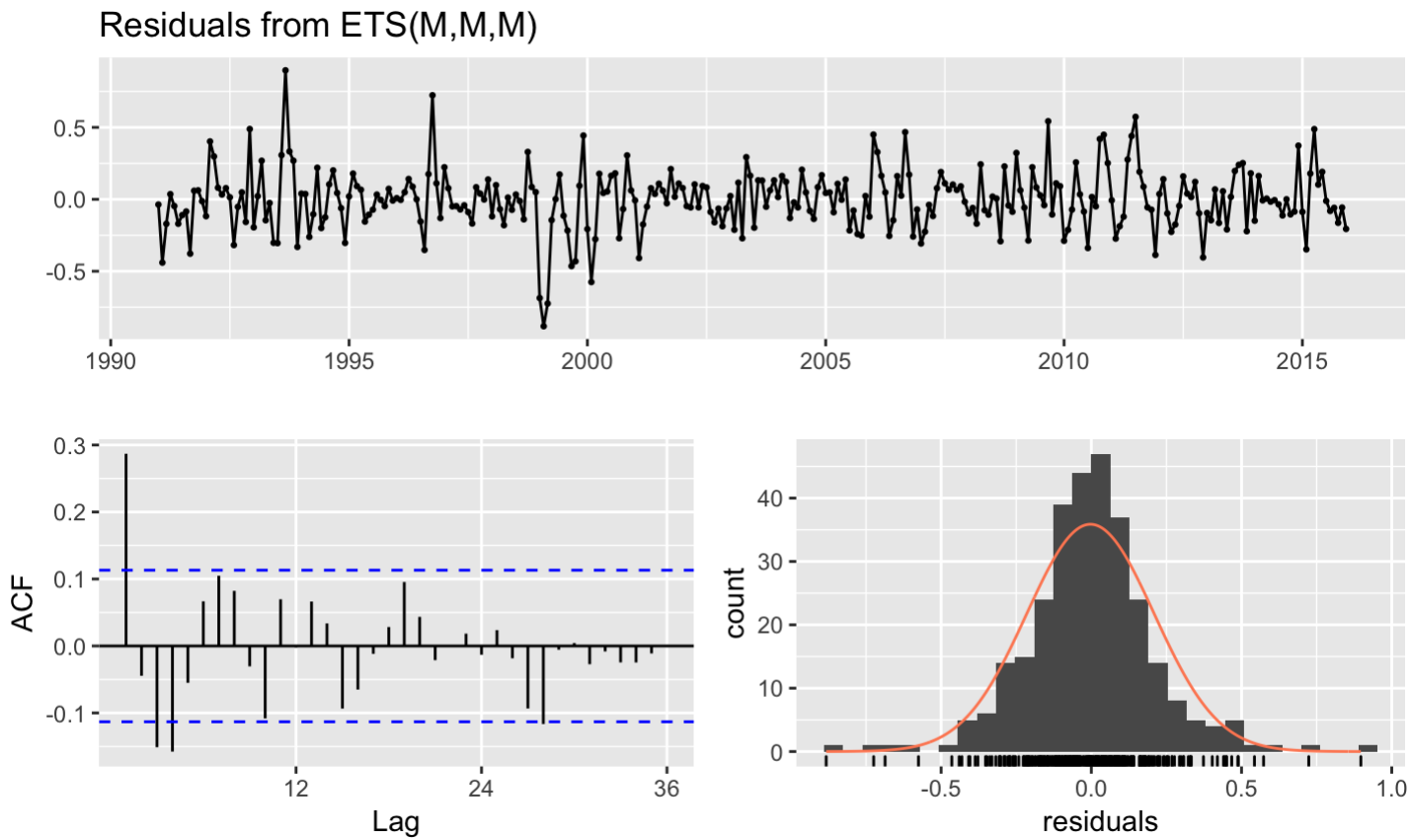



```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,Ad,M)
## Q* = 63.25, df = 7, p-value = 3.377e-11
##
## Model df: 17.   Total lags used: 24
```

```r
fit.MMM = ets(NMFS_Landings.ts, model = "MMM")
summary(fit.MMM)
```

```
## ETS(M,M,M)
##
## Call:
##  ets(y = NMFS_Landings.ts, model = "MMM")
##
##   Smoothing parameters:
##     alpha = 0.1094
##     beta  = 1e-04
##     gamma = 0.6149
##
##   Initial states:
##     l = 454827.3622
##     b = 0.997
##     s = 0.3433 0.3449 0.5589 1.2808 1.9131 2.0651
##            1.4715 0.6972 0.5307 0.6756 1.3148 0.8041
##
##   sigma:  0.2176
##
##      AIC     AICc      BIC
## 8026.037 8028.207 8089.001
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -4936.234 61476.12 33920.56 -7.706504 19.64354 0.9435574
##                    ACF1
## Training set 0.3192502
```

```
checkresiduals(fit.MMM)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,M,M)
## Q* = 63.419, df = 8, p-value = 9.906e-11
##
## Model df: 16.   Total lags used: 24
```

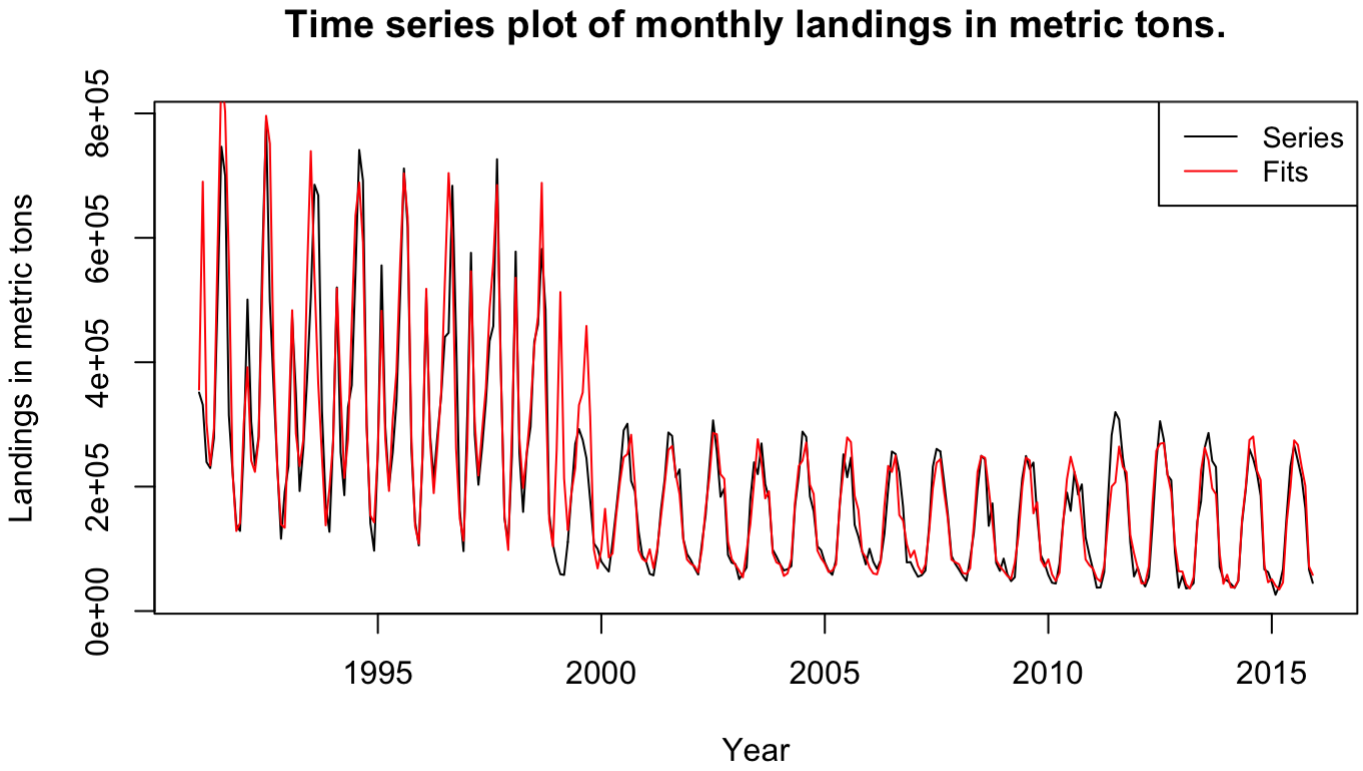
When we include the trend component in the model, although we get similar MASE values, BIC has considerably dropped.

Diagnostic check results are similar to the models with no trend.

Because MASE of MAM model is smaller than that of the MMM model, we'll go on with the MAM model for prediction distributions and intervals.

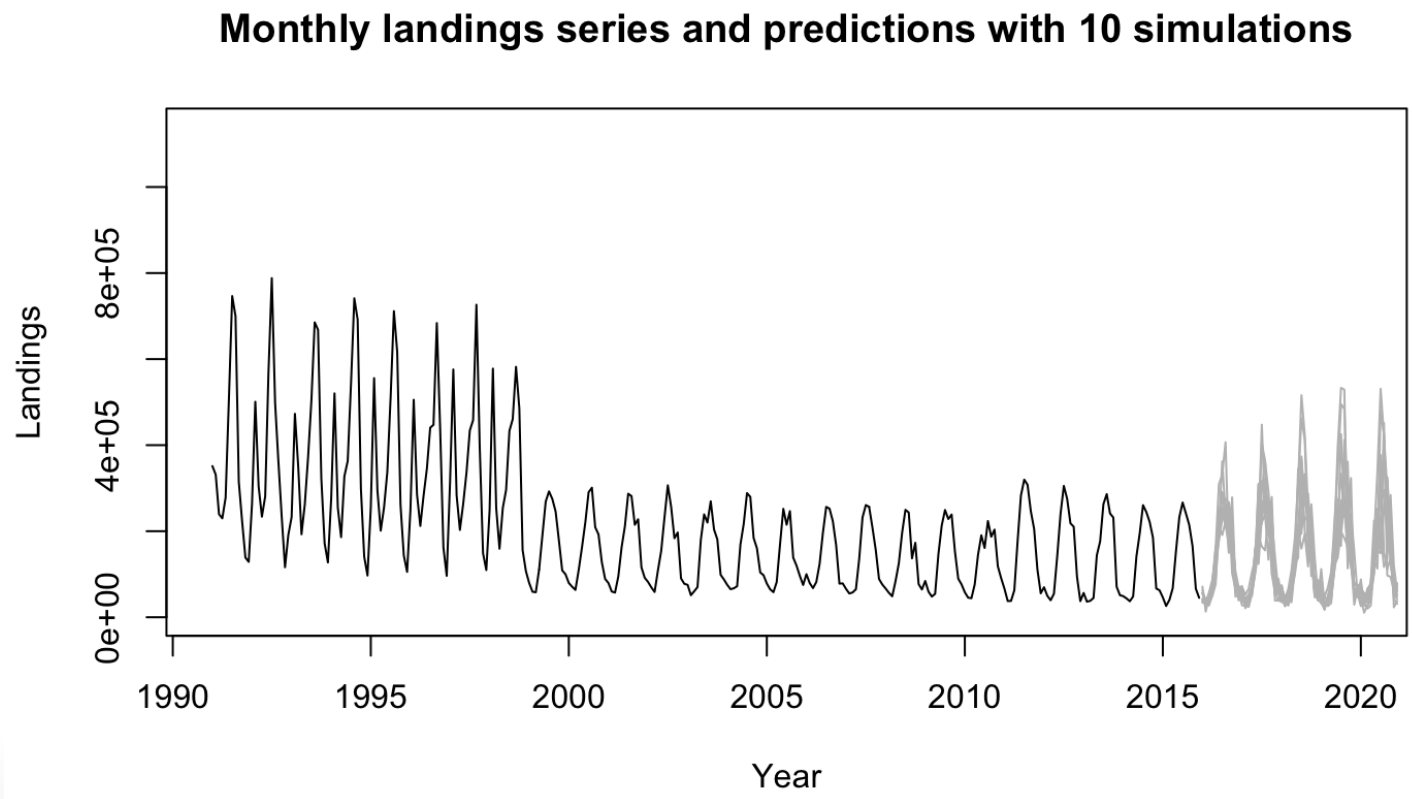
The following plot displays the original series and fits.

```
plot(NMFS_Landings.ts,ylab='Landings in metric tons',xlab='Year', main = "Time series plot of monthly
lines(fit.MAM$fitted, col="red")
legend("topright",lty=1, cex = 0.9, col=c("black","red"), c("Series", "Fits"))
```



```r
A = ts(matrix(NA,60,5000),start = c(2016,1), frequency = 12)
# A starts at the end of original series
n = length(NMFS_Landings.ts)
h = 10
M = 5000
for (i in 1:M){
  A[,i] = simulate(fit.MAM,initstate=fit.MAM$states[25,],nsim=60)
  # Generate random epsilons and apply model formulation
}
```

```r
plot(NMFS_Landings.ts, ylim=range(NMFS_Landings.ts,A),ylab="Landings",xlim=c(1991,2020), xlab="Year",
     main="Monthly landings series and predictions with 10 simulations")
for(i in 1:10){
  lines(A[,i], col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"10 simulated future sample paths",adj=0)
```



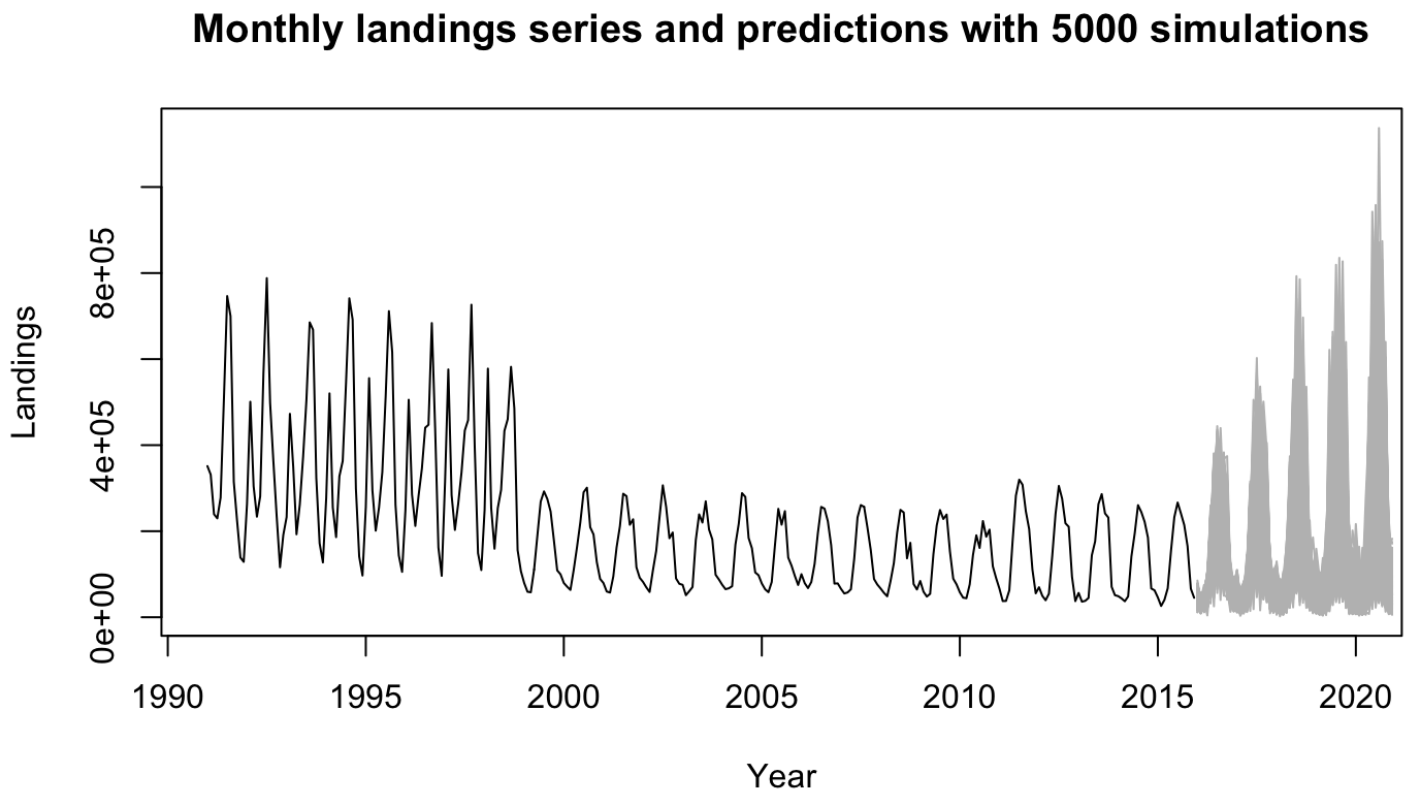
It seems that the range of predictions is so narrow.

The range of predictions also resembles both parts of landings series.

But are these interpretations reliable?

Let’s have a look at the same picture with more simulations.

```r
plot(NMFS_Landings.ts, ylim=range(NMFS_Landings.ts,A),ylab="Landings",xlim=c(1991,2020), xlab="Year",
     main="Monthly landings series and predictions with 5000 simulations")
for(i in 1:5000){
  lines(A[,i], col="gray")
}
text(1,600,"Historical data",adj=0)
text(7,600,"10 simulated future sample paths",adj=0)
```



The picture we get with 5000 simulations is quite different from the one with 10 simulations.

As expected, they are getting wider and wider.

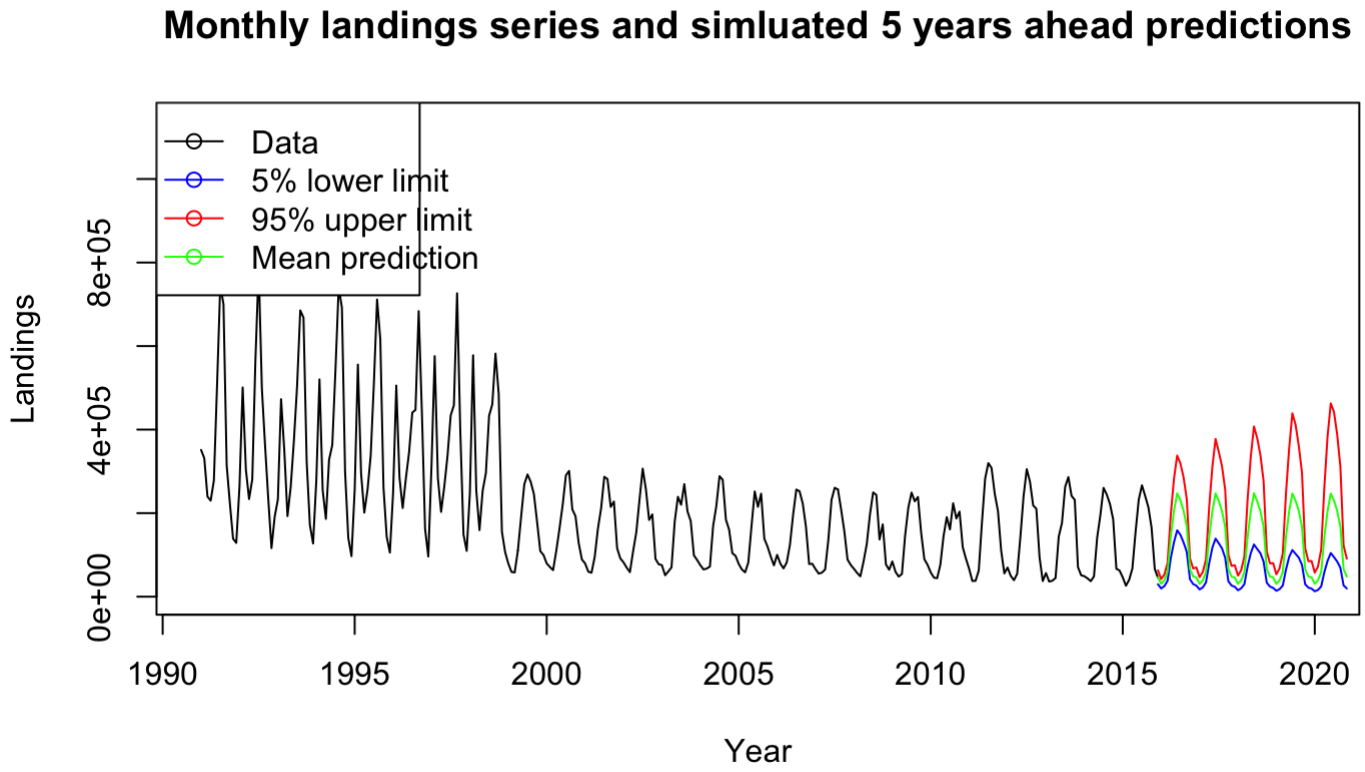
But still, they are under the effect of both parts of the series.

Let’s visualise 5%, 95% and the average of the predictions obtained by simulation.

These results are obtained over 5000 simulations.

```r
# Define the variables to put quantiles and sample average
Pi = array(NA, dim=c(60,2))
avrg = array(NA, 60)
# Calcualte the interval estimates and mid point
for (i in 1:60){
  Pi[i,] = quantile(A[i,],type=8,prob=c(.05,.95))
  avrg[i] = mean(A[i,]) # This would be median as well
}
# Create ts objects for plotting
Pi.lb = ts(Pi[,1],start=end(NMFS_Landings.ts),frequency = 12)
Pi.ub = ts(Pi[,2],start=end(NMFS_Landings.ts),frequency = 12)
avrg.pred = ts(avrg,start=end(NMFS_Landings.ts),frequency = 12)
```

```
plot(NMFS_Landings.ts,xlim=c(1991,2020),ylim=range(NMFS_Landings.ts,A),ylab="Landings",xlab="Year", m
lines(Pi.lb,col="blue", type="l")
lines(Pi.ub,col="red", type="l")
lines(avrg.pred,col="green", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red","green"), text.width = 4,
       c("Data","5% lower limit","95% upper limit","Mean prediction"))
```

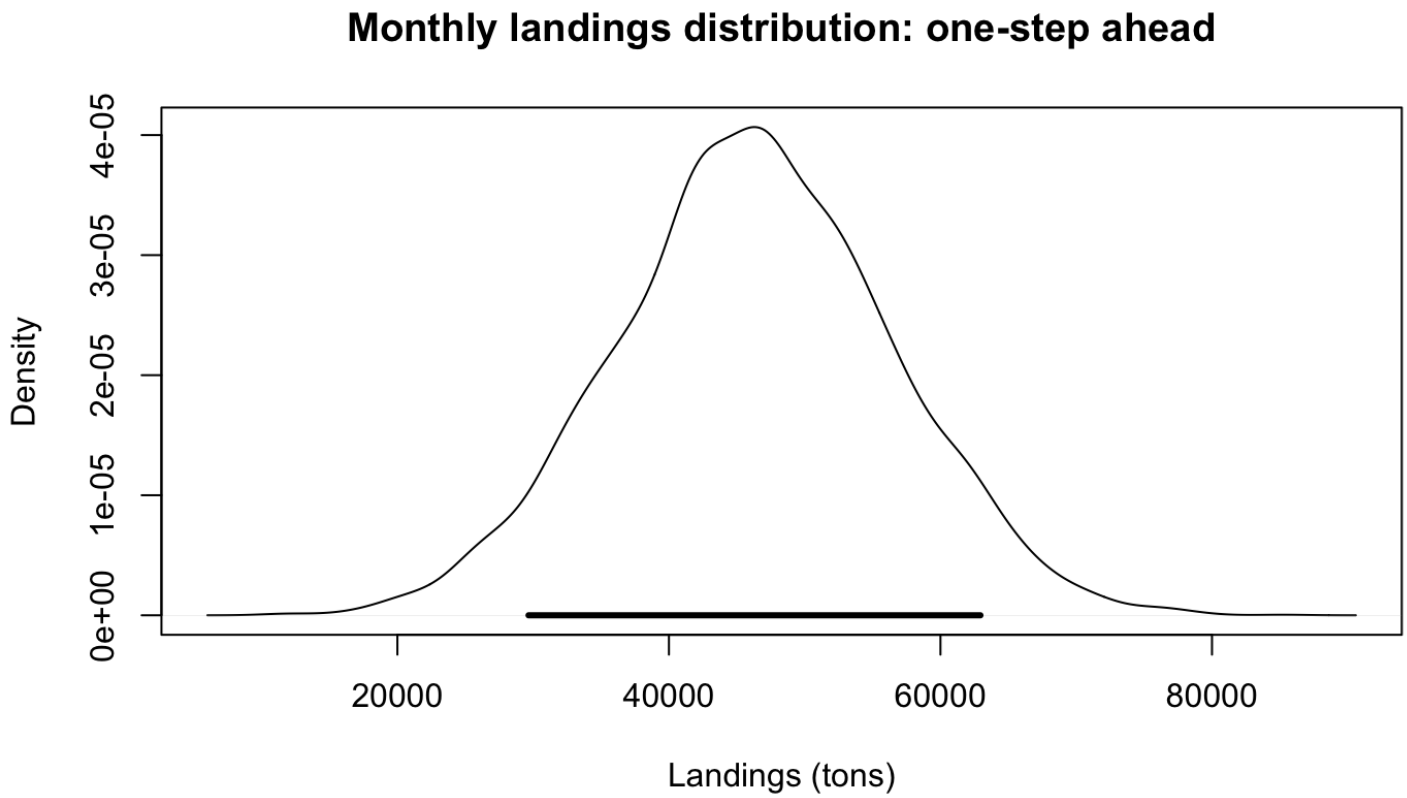


Let’s visualise 5% and 95% limits are getting wider as expected.

Also, notice that the average of simulated forecasts does not shrink upwards along with the 95% limits.

Now, we will produce an estimate of pdf of one-time ahead forecasts.

```
pi.1 = quantile(A[1,],type=8,prob=c(.05,.95))
plot(density(A[1,], bw="SJ"),main="Monthly landings distribution: one-step ahead",xlab="Landings (ton
lines(pi.1,c(0,0),lwd=3,col=1)
text(mean(pi.1),0.0005,"90% Prediction Interval")
```



See the difference between the two ends of the distribution and the limits of 90% confidence interval.

One-time ahead landings have a left-skewed pdf.

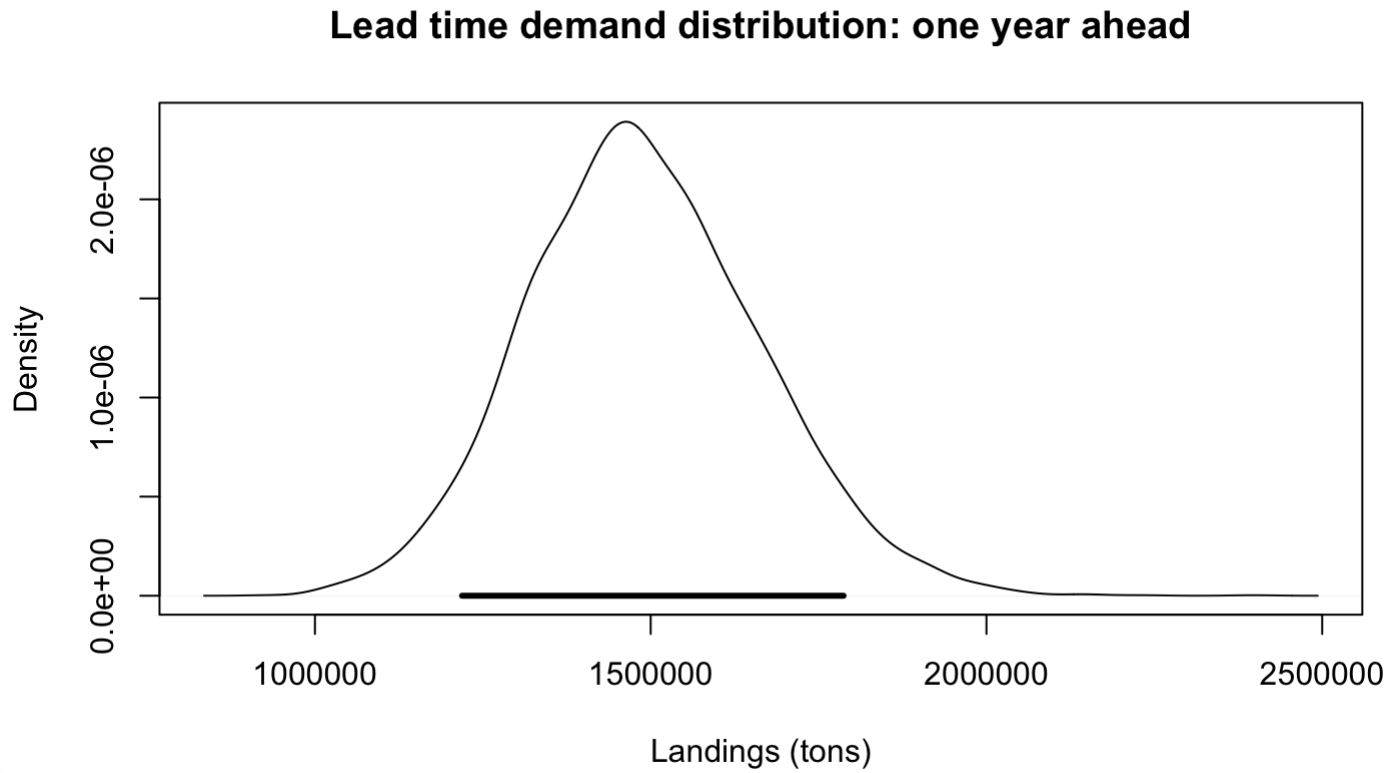
Most of the probability density is in between roughly 35000 and 65000 tons.

It is probable to have very low landings as well.

The probability of having the value of a landing greater then 80000 tons is very very low.

```
ltd <- colSums(A[1:12,])
ltd.den <- density(ltd, bw="SJ")

plot(ltd.den,main="Lead time demand distribution: one year ahead",xlab="Landings (tons)")
text(median(ltd)/1e3,0.2,"90% Prediction Interval")
lines(quantile(ltd,type=8,prob=c(.05,.95)),c(0,0),lwd=3,col=1)
```



For the distribution of one year ahead predictions, we obtained a different estimated pdf.

One year ahead landings have a right-skewed pdf.

The probability of having a landings amount in a year greater then 2M tons is very very low.

Also, the probability of having a landings amount in a year less then 1M tons is very very low.

Most of the probability density of yearly landings is around 1.5M tons.

Summary

In this module, we focused on parameter estimation in innovations state-space models from the applied point of view.

We observed that different objective function in the optimization of likelihood function could give different estimates of parameters.

Also, we need to consider the stability conditions in the estimation process.

Another important aspect of forecasting task is to get interval estimates for our future predictions.

Due to the complexity of the state space formulations of the models, it is hard to work out the prediction variances analytically.

Instead, the Monte Carlo simulation approach provides a suitable solution to this prediction problem.

We easily generate a large number of sample paths and use empirical estimates of sampling quantities like quantiles, mean, median to infer about the predictions and their variances, and confidence intervals as well.

What's next?

In the next module, we will focus on

- selection of models and
- comparing selection procedures.

References

Chatfield, C. (1993). Calculating interval forecasts, *Journal of Business & Economic Statistics*, 11, 121–135.

Hyndman, R.J., Koehler, A.B., Ord, J.K., and Snyder, R.D. (2008). Forecasting with exponential smoothing: the state space approach, Springer-Verlag.

Hyndman, R.J., Athanasopoulos, G. (2014). *Forecasting: Principles and Practice*. OTEXTS.

Thanks for your attendance! Pleas
follow the link
https://forms.gle/oE67NUczuvX2P
to give feedback!