Holt and Holt-Winters Models – Model Evaluation

FISH 550 – Applied Time Series Analysis

Eli Holmes

15 April 2025

Exponential Smoothing Models

ETS Models in Forecasting: Principles and Practice

Exponential smoothing models are a family of forecasting tools that help us make smarter predictions by giving **more weight to recent observations** while not completely ignoring the past.

Why "exponential"? The weights assigned to past observations decrease exponentially — meaning the most recent data gets the most attention, and older data fades out gradually.

Simple Level Model

My prediction for today is an exponentially weighted average of past values.

$$\hat{y}_{t+1} = \ell_t$$

$$\ell_t = \alpha y_t + \alpha(1-\alpha)y_{t-1} + \alpha(1-\alpha)^2 y_{t-2} + \dots$$

Simple Trend + Level Model (Holt)

My prediction includes both an exponentially weighted average of past values **and** a trend.

“trend” = tendency to increase or decrease.

$$\hat{y}_{t+1} = \ell_t + b_t$$

Where:

- ℓ_t: exponentially weighted average of past values
- b_t: exponentially weighted average of past trends ($\ell_t - \ell_{t-1}$)

Expanded Form

Level:

$$\ell_t = \alpha y_t + \alpha(1-\alpha)(y_{t-1} + b_{t-1}) + \alpha(1-\alpha)^2(y_{t-2} + b_{t-2}) + \dots$$

Trend:

$$b_t = \beta(\ell_t - \ell_{t-1}) + \beta(1-\beta)(\ell_{t-1} - \ell_{t-2}) + (1-\beta)^2 b_{t-2}$$

Holt-Winters Model

My forecast includes a weighted average of past values, a trend, **and** a seasonal component:

$$\hat{y}_{t+1} = \ell_t + b_t + s_{t-m}$$

Where: - ℓ_t: smoothed level (local average) - b_t: smoothed trend (local slope) - s_{t-m}: seasonal adjustment from one full cycle ago

Expanded form of seasonality:

$$
\begin{aligned}
s_t &= \gamma(y_t - \ell_t) + (1-\gamma)s_{t-m} \\
&= \gamma(y_t - \ell_t) + \gamma(1-\gamma)(y_{t-m} - \ell_{t-m}) \\
&\quad + \gamma(1-\gamma)^2(y_{t-2m} - \ell_{t-2m}) + \gamma(1-\gamma)^3(y_{t-3m} - \ell_{t-3m}) + \dots
\end{aligned}
$$

This lets the **seasonal pattern** evolve gradually over time.

Fitting Holt and Holt-Winters

The forecast package provides:

- `holt(dat)`, `ets(dat, model="AAN")` Holt
- `hw(dat)`, `ets(dat, model="AAA")` Holt-Winters
- `ets(dat, model="ZZZ")` others

Fit Holt model

Evolving level and trend.

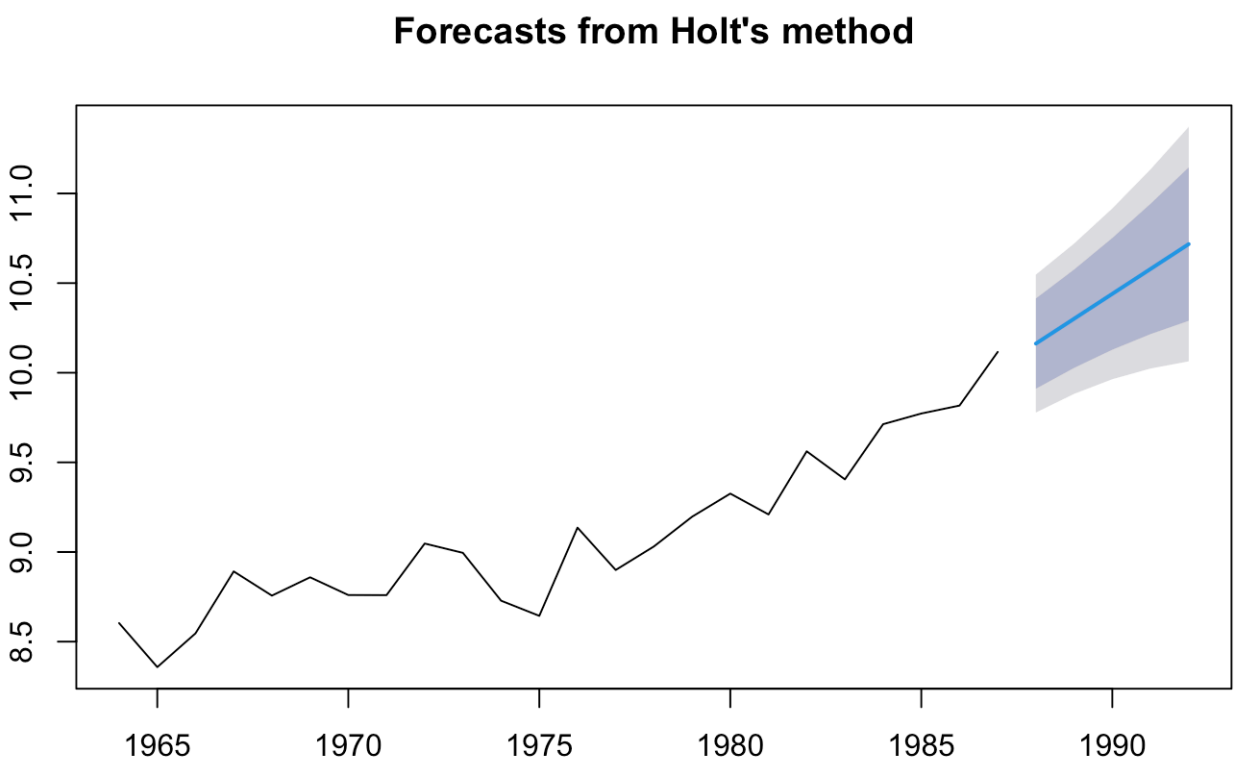
Load the data and **forecast** package.

```
data(greeklandings, package="atsalibrary")
anchovy <- subset(greeklandings,
                  Species=="Anchovy" & Year<=1987)$log.metric.tons
anchovy <- ts(anchovy, start=1964)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
fit <- forecast::holt(anchovy)
fr <- forecast(fit, h=5)
```

```
plot(fr)
```


Forecasts from Holt's method
8.5
9.0
9.5
10.0
10.5
11.0
1965
1970
1975
1980
1985
1990

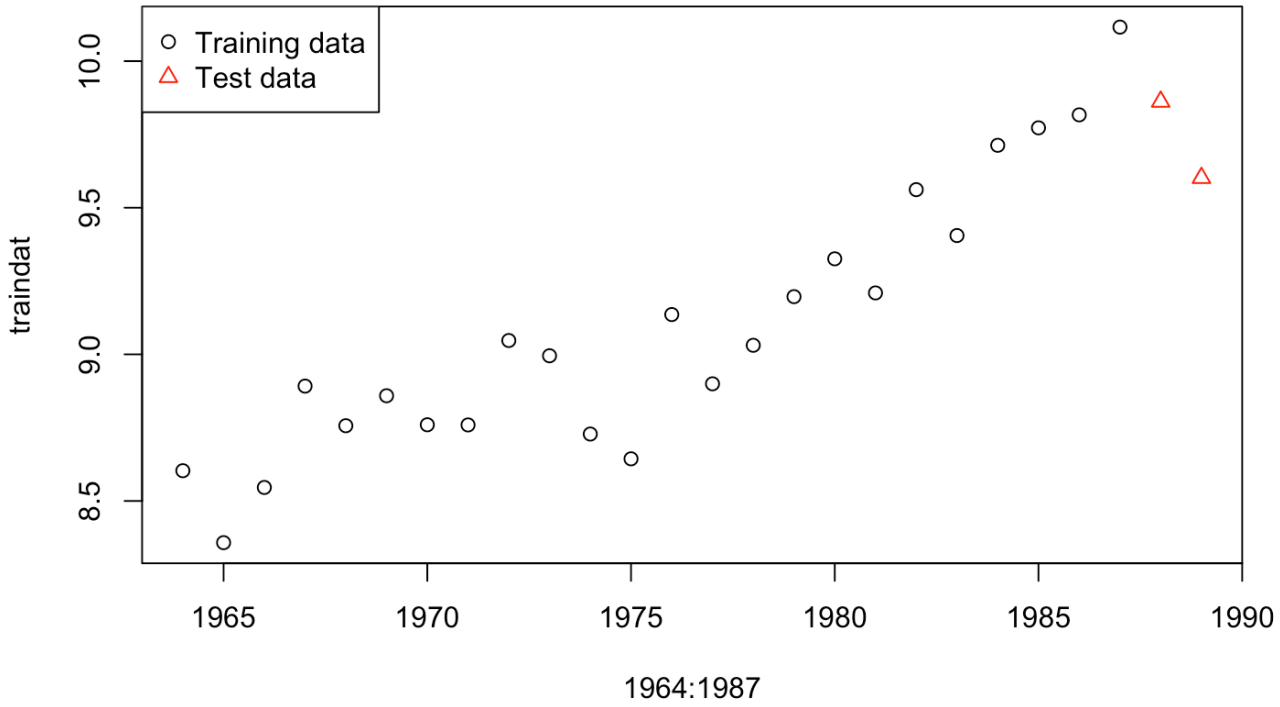
Validation

Once we have a model(s), we will want to evaluate the performance of the model. We'll see examples using the anchovy landings.

```
spp <- "Anchovy"
training = subset(greeklandings, Year <= 1987)
test = subset(greeklandings, Year >= 1988 & Year <= 1989)
traindat <- subset(training, Species==spp)$log.metric.tons
testdat <- subset(test, Species==spp)$log.metric.tons
```

Measures of forecast fit

To measure the forecast fit, we fit a model to training data and test a forecast against data in a test set. We 'held out' the test data and did not use it for fitting.



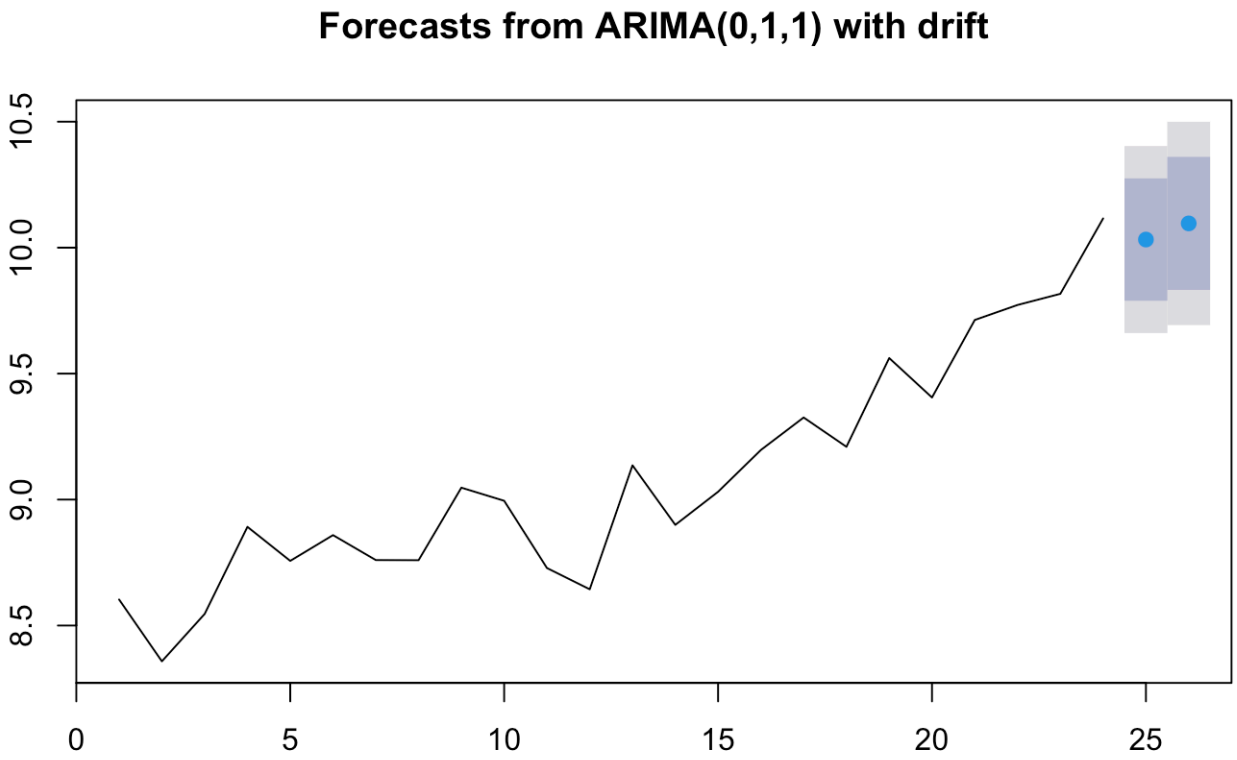
We will fit to the training data and make a forecast.

```
fit_arima <- auto.arima(traindat)
fit_arima
```

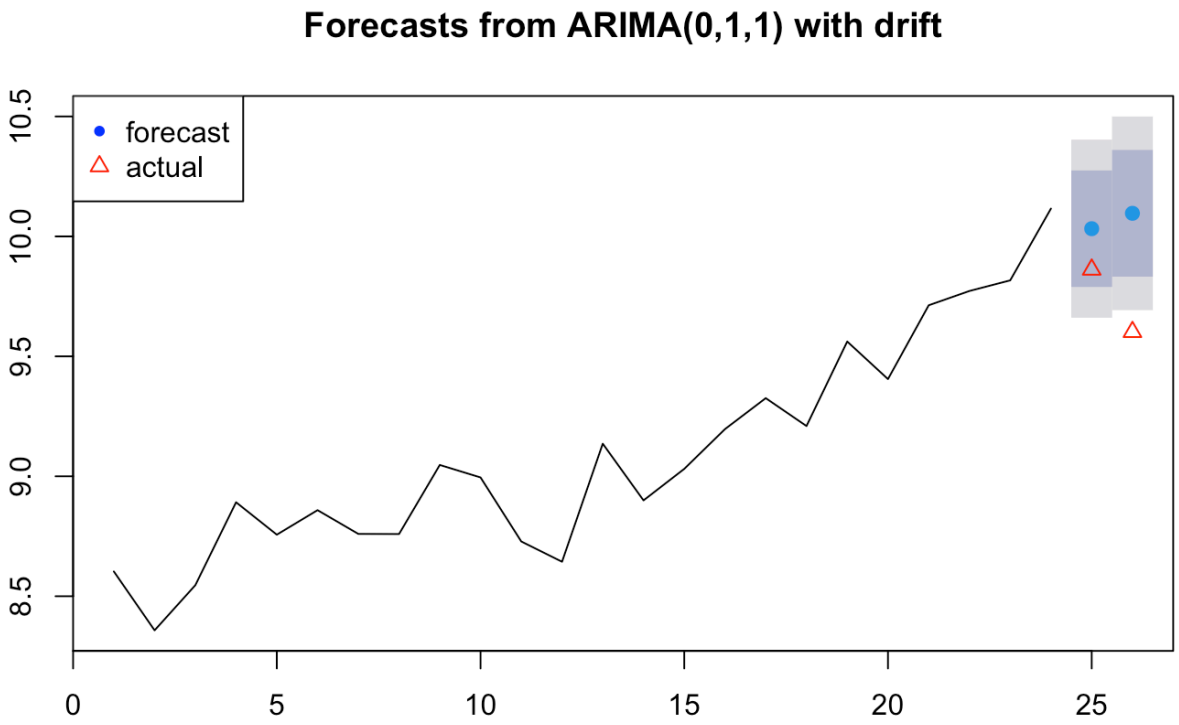
```
## Series: traindat
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.5731  0.0641
## s.e.   0.1610  0.0173
##
## sigma^2 = 0.03583:  log likelihood = 6.5
## AIC=-6.99   AICc=-5.73   BIC=-3.58
```

Forecast

```
fr <- forecast(fit_arima, h=2)
plot(fr)
```



Compare to test



How do we quantify the difference between the forecast and the actual values?

```
fr.err <- testdat - fr$mean
fr.err
```

```
## Time Series:
## Start = 25
## End = 26
## Frequency = 1
## [1] -0.1704302 -0.4944778
```

There are many metrics. The `accuracy()` function in forecast provides many different metrics: mean error, root mean square error, mean absolute error, mean percentage error, mean absolute percentage error.

ME Mean error

```
me <- mean(fr.err); me
```

```
## [1] -0.332454
```

RMSE Root mean squared error

```
rmse <- sqrt(mean(fr.err^2)); rmse
```

```
## [1] 0.3698342
```

MAE Mean absolute error

```
mae <- mean(abs(fr.err)); mae
```

```
## [1] 0.332454
```

MPE Mean percentage error

```
fr.pe <- 100*fr.err/testdat
mpe <- mean(fr.pe); mpe
```

```
## [1] -3.439028
```

MAPE Mean absolute percentage error

```
mape <- mean(abs(fr.pe)); mape
```

```
## [1] 3.439028
```

```
accuracy(fr, testdat)[,1:5]
```

```
##                       ME      RMSE       MAE        MPE     MAPE
## Training set -0.00473511 0.1770653 0.1438523 -0.1102259 1.588409
## Test set     -0.33245398 0.3698342 0.3324540 -3.4390277 3.439028
```

```
c(me, rmse, mae, mpe, mape)
```

```
## [1] -0.3324540  0.3698342  0.3324540 -3.4390277  3.4390277
```

Test all the models in your candidate set

Compute metrics for all the models in your candidate set.

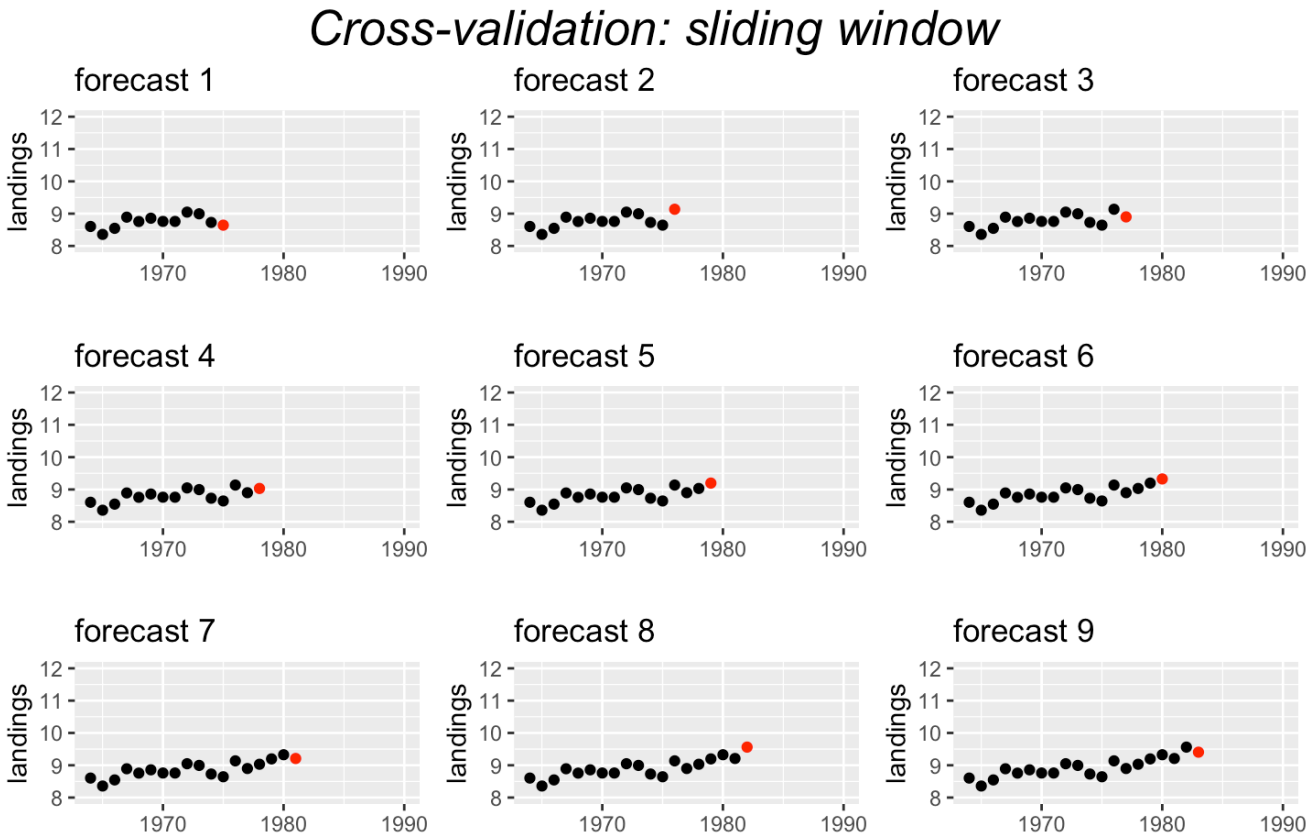
```
# The model picked by auto.arima
fit1 <- Arima(traindat, order=c(0,1,1))
fr1 <- forecast(fit1, h=2)
test1 <- accuracy(fr1, testdat)[2,1:5]

# AR-1
fit2 <- Arima(traindat, order=c(1,1,0))
fr2 <- forecast(fit2, h=2)
test2 <- accuracy(fr2, testdat)[2,1:5]

# Holt
fit3 <- holt(traindat)
fr3 <- forecast(fit3, h=2)
test3 <- accuracy(fr3, testdat)[2,1:5]
```

Show a summary

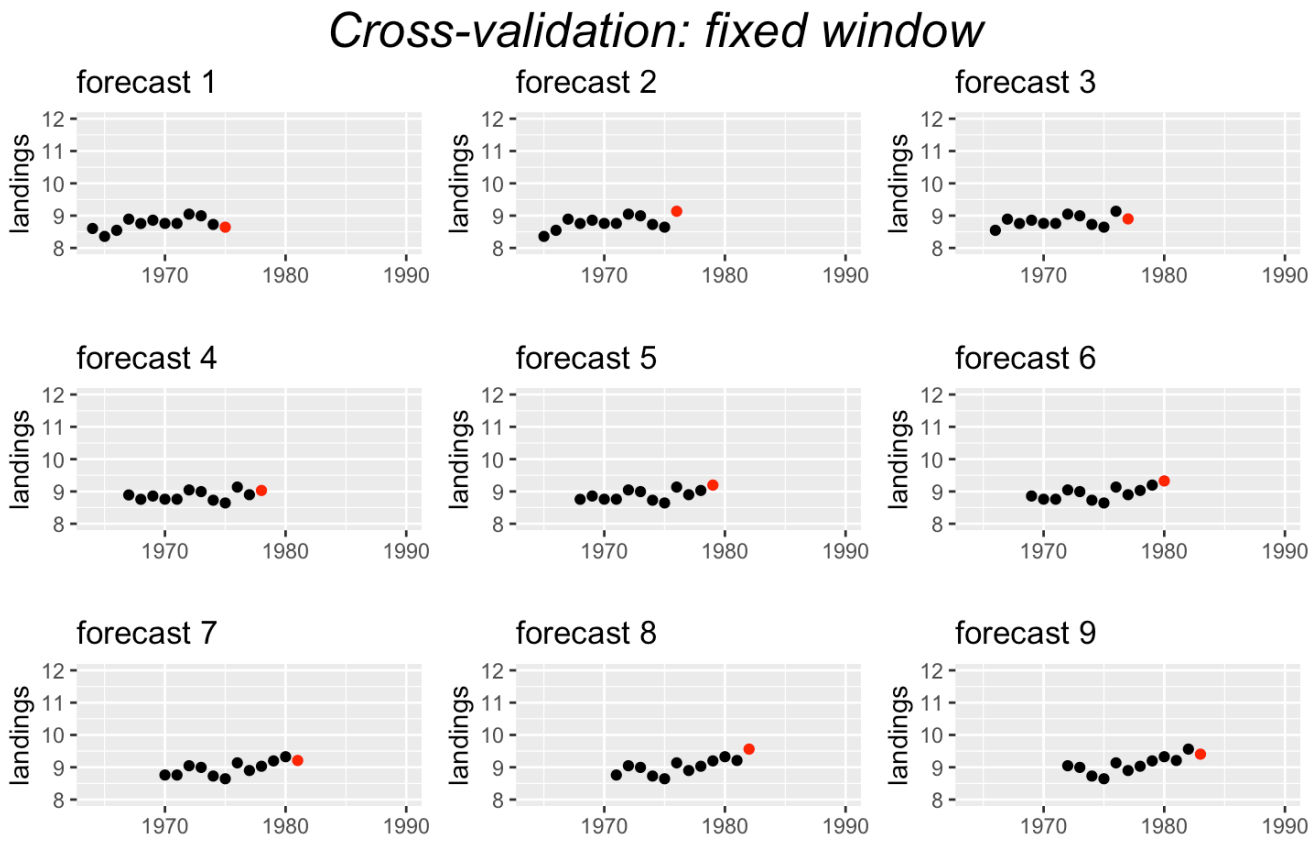
	ME	RMSE	MAE	MPE	MAPE
(0,1,1)	-0.293	0.320	0.293	-3.024	3.024
(1,1,0)	-0.309	0.341	0.309	-3.200	3.200
Holt	-0.500	0.538	0.500	-5.168	5.168

Cross-validation

An alternate approach to testing a model's forecast accuracy is to use windows.



Another approach uses a fixed window. For example, a 10-year window.



Cross-validation with the forecast package

```
far2 <- function(x, h, order){
  forecast(Arima(x, order=order), h=h)
  }
e <- tsCV(traindat, far2, h=1, order=c(0,1,1))
tscv1 <- c(ME=mean(e, na.rm=TRUE), RMSE=sqrt(mean(e^2, na.rm=TRUE)),
           MAE=mean(abs(e), na.rm=TRUE))
tscv1
```

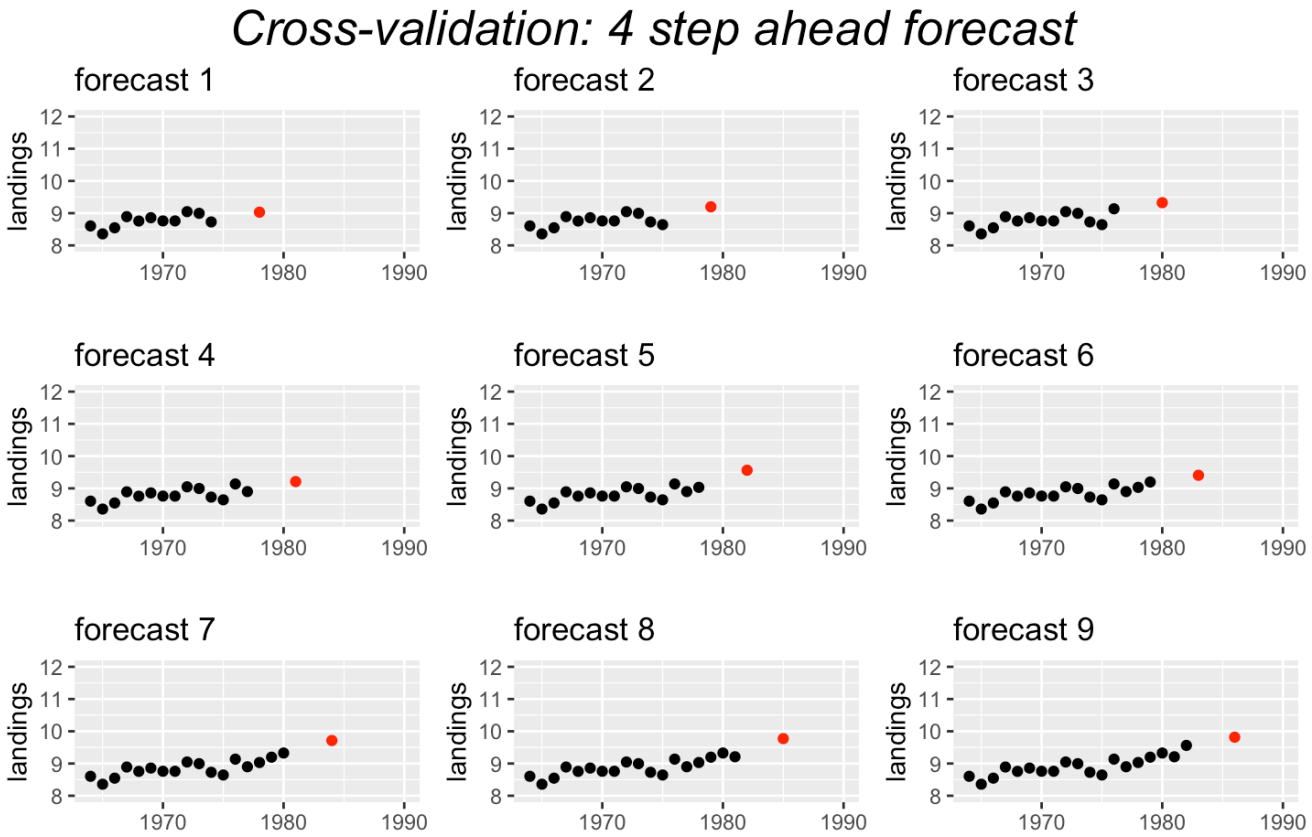
```
##        ME      RMSE       MAE
## 0.1128788 0.2261706 0.1880392
```

Compare to RMSE from just the 2 test data points.

```
test1[c("ME","RMSE","MAE")]
```

```
##         ME       RMSE        MAE
## -0.2925326  0.3201093  0.2925326
```

Cross-validation farther in future



Compare accuracy of forecasts 1 year out versus 4 years out. If h is greater than 1, then the errors are returned as a matrix with each h in a column. Column 4 is the forecast, 4 years out.

```
e <- tsCV(traindat, far2, h=4, order=c(0,1,1))[,4]
#RMSE
tscv4 <- c(ME=mean(e, na.rm=TRUE), RMSE=sqrt(mean(e^2, na.rm=TRUE)),
           MAE=mean(abs(e), na.rm=TRUE))
rbind(tscv1, tscv4)
```

```
##               ME      RMSE       MAE
## tscv1 0.1128788 0.2261706 0.1880392
## tscv4 0.2839064 0.3812815 0.3359689
```

Compare accuracy of forecasts with a fixed 10-year window and 1-year out forecasts.

```
e <- tsCV(traindat, far2, h=1, order=c(0,1,1), window=10)
#RMSE
tscvf1 <- c(ME=mean(e, na.rm=TRUE), RMSE=sqrt(mean(e^2, na.rm=TRUE)),
            MAE=mean(abs(e), na.rm=TRUE))
tscvf1
```

```
##        ME      RMSE       MAE
## 0.1387670 0.2286572 0.1942840
```

```
comp.tab <- rbind(test1=test1[c("ME","RMSE","MAE")],
      slide1=tscv1,
      slide4=tscv4,
      fixed1=tscvf1)
knitr::kable(comp.tab, format="html")
```

	ME	RMSE	MAE
test1	-0.2925326	0.3201093	0.2925326
slide1	0.1128788	0.2261706	0.1880392
slide4	0.2839064	0.3812815	0.3359689
fixed1	0.1387670	0.2286572	0.1942840

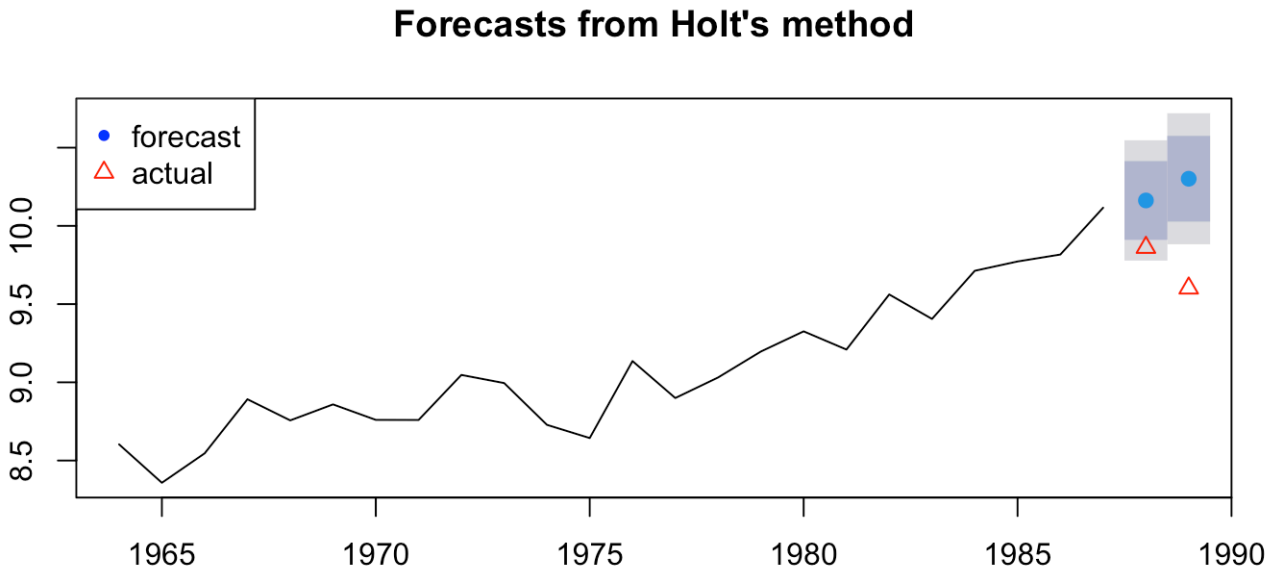
Summary forecast performance

We can evaluate the forecast performance with forecasts of our test data or we can use all the data and use time-series cross-validation.

Let’s start with the former.

Test forecast performance against a test data set

We will fit Holt model to the training data and make a forecast for the years that we ‘held out’.



We can calculate a variety of forecast error metrics with

```
accuracy(fr, testdat)
```

```
##                      ME      RMSE       MAE        MPE     MAPE      MASE
## Training set  0.0155550 0.1788989 0.1442749  0.1272736 1.600569 0.7721001
## Test set     -0.5001404 0.5384071 0.5001404 -5.1675444 5.167544 2.6765467
##                     ACF1 Theil's U
## Training set -0.008467932        NA
## Test set     -0.500000000  2.690789
```

We would now repeat this for all the models in our candidate set and choose the model with the best forecast performance.

Forecast performance with time-series cross-validation

We can use `tsCV()` as we did for ARIMA models. We will redefine `traindat` as all our Anchovy data.

tsCV() function

We will use the `tsCV()` function. We need to define a function that returns a forecast.

```
far2 <- function(x, h){
  fit <- holt(x)
  forecast(fit, h=h)
  }
```

Now we can use `tsCV()` to run our `far2()` function to a series of training data sets. We will specify that a NEW ets model be estimated for each training set. We are not using the weighting estimated for the whole data set but estimating the weighting new for each set.

The `e` are our forecast errors for all the forecasts that we did with the data.

```
e <- tsCV(traindat, far2, h=1)
c(ME=mean(e, na.rm=TRUE), RMSE=sqrt(mean(e^2, na.rm=TRUE)),
          MAE=mean(abs(e), na.rm=TRUE))
```

```
##         ME       RMSE        MAE
## 0.03393316 0.26161402 0.22468116
```

Seasonal Holt-Winters

```
data("chinook", package="atsalibrary")
head(chinook.month)
```

Year	Month	Species	State	log.metric.tons	metric.tons	value.usd
1990	Jan	Chinook	WA	3.26	26.1	108685
1990	Feb	Chinook	WA	3.78	43.7	309196
1990	Mar	Chinook	WA	3.47	32.1	201279
1990	Apr	Chinook	WA	4.23	69	433238
1990	May	Chinook	WA	5.09	162	876329
1990	Jun	Chinook	WA	4.28	71.9	421885

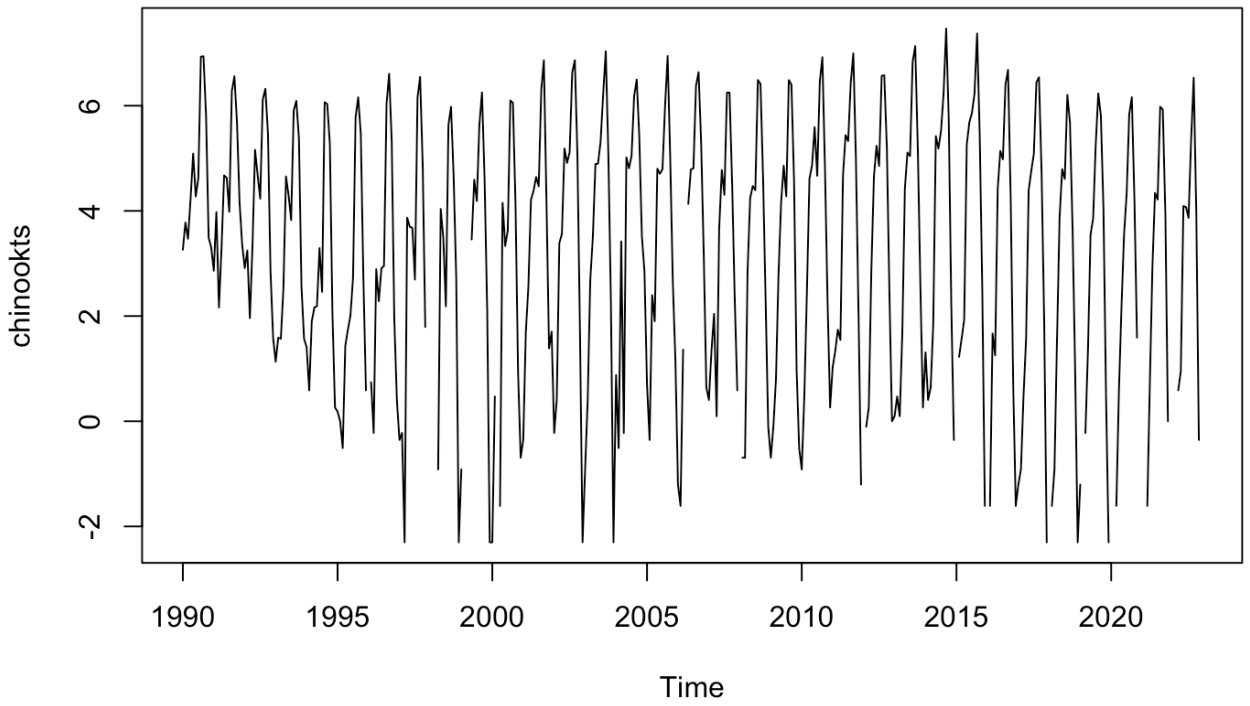
The data are monthly and start in January 1990. To make this into a ts object do

```
df <- chinook.month %>% subset(State == "WA")
chinookts <- ts(df$log.metric.tons, start=c(1990,1), frequency=12)
```

`start` is the year and month and frequency is the number of months in the year.

Plot seasonal data

```
plot(chinookts)
```

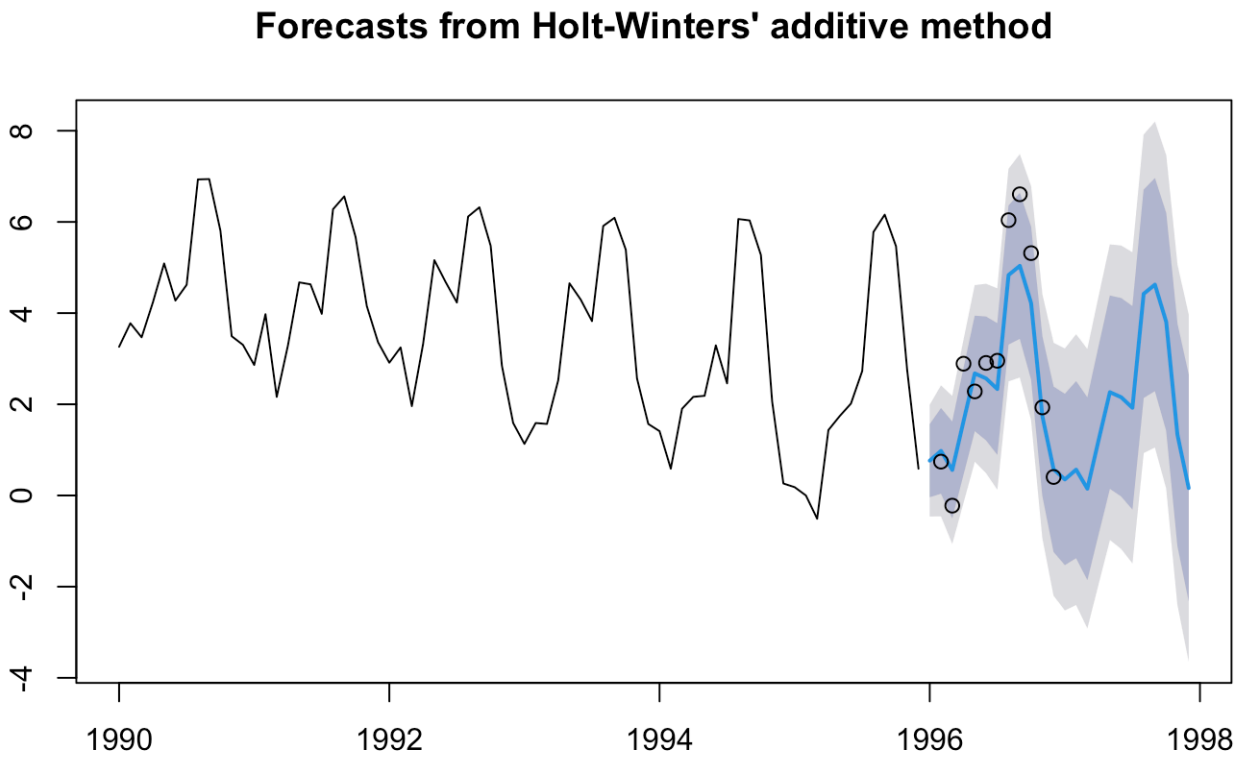


Holt-Winters

```
library(forecast)
traindat <- window(chinookts, c(1990,1), c(1999,12))
fit <- forecast::hw(traindat)
```

```
## Warning in ets(x, "AAA", alpha = alpha, beta = beta, gamma = gamma, phi = phi,
## : Missing values encountered. Using longest contiguous portion of time series
```

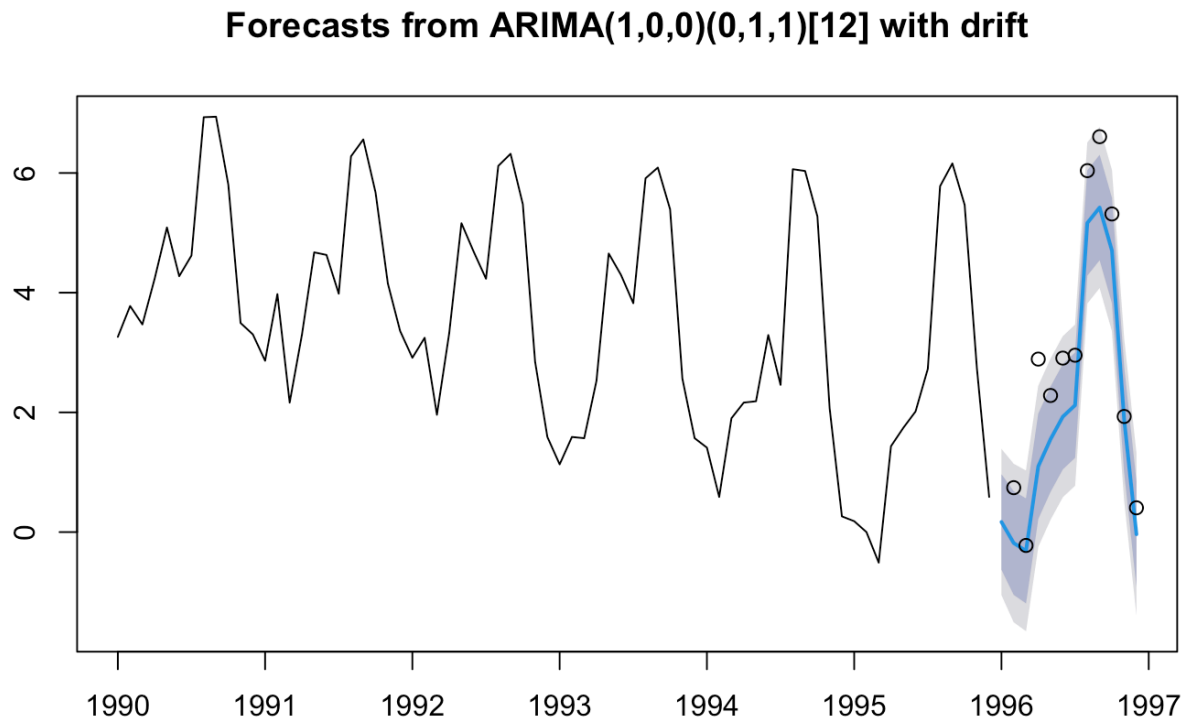
```
fr <- forecast(fit, h=24)
plot(fr)
points(window(chinookts, c(1996,1), c(1996,12)))
```



Compare to a seasonal ARIMA model

`auto.arima()` will recognize that our data has season and fit a seasonal ARIMA model to our data. Let's use the data that `ets()` used. This is shorter than our training data. The data used by `hw()` is returned in `fit$x`.

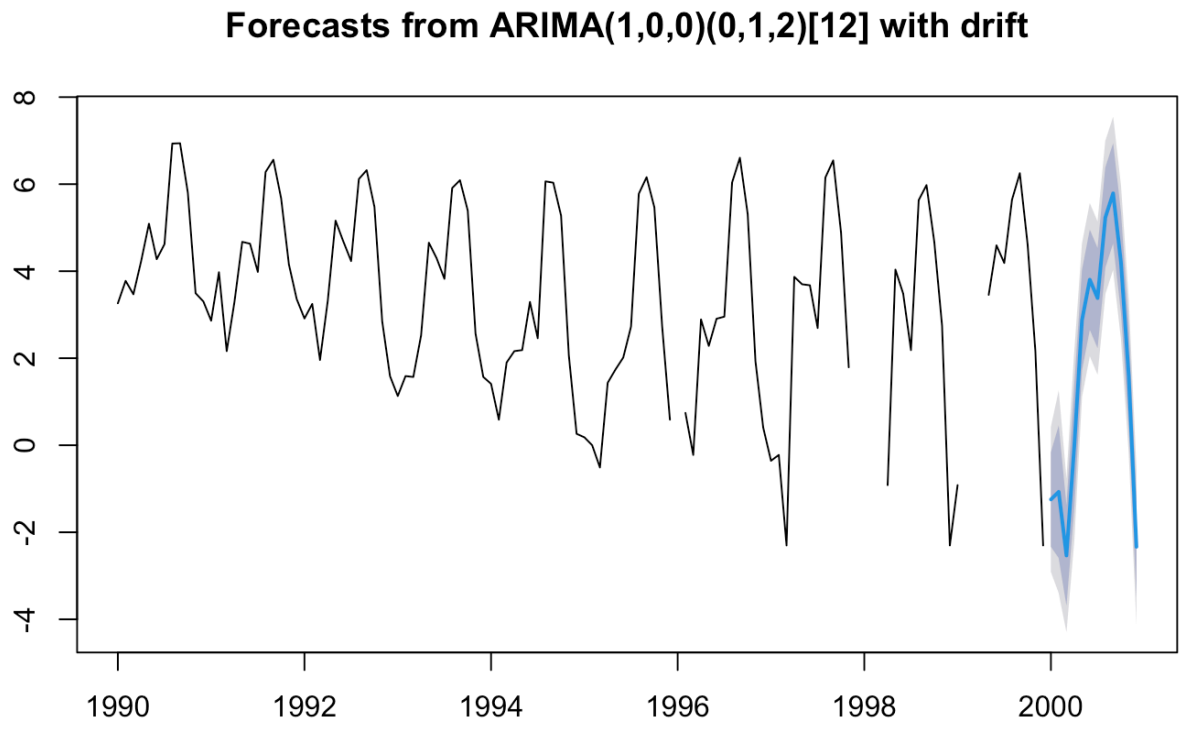
```
no_miss_dat <- fit$x
fit <- auto.arima(no_miss_dat)
fr <- forecast(fit, h=12)
plot(fr)
points(window(chinookts, c(1996,1), c(1996,12)))
```



Missing values

Missing values are ok when fitting a seasonal ARIMA model, unlike an ETS model.

```
fit <- auto.arima(traindat)
fr <- forecast(fit, h=12)
plot(fr)
```



Forecast evaluation

We can compute the forecast performance metrics as usual.

```
fit <- hw(traindat)
```

```
## Warning in ets(x, "AAA", alpha = alpha, beta = beta, gamma = gamma, phi = phi,
## : Missing values encountered. Using longest contiguous portion of time series
```

```
fr <- forecast(fit, h=12)
```

Look at the forecast so you know what years and months to include in your test data. Pull those 12 months out of your data using the `window()` function.

```
testdat <- window(traindat, c(1996,1), c(1996,12))
```

Use `accuracy()` to get the forecast error metrics.

```
accuracy(fr, testdat)
```

```
##                       ME      RMSE       MAE      MPE    MAPE      MASE
## Training set 0.0009781278 0.5506796 0.4238807     -Inf     Inf 0.7095807
## Test set     0.4269572812 0.8579622 0.7150331 37.32735 53.8648 1.1969730
##                    ACF1 Theil's U
## Training set 0.04312778        NA
## Test set     0.15488159 0.4105402
```

We can do the same for the ARIMA model.

```
no_miss_dat <- fit$x
fit <- auto.arima(no_miss_dat)
fr <- forecast(fit, h=12)
accuracy(fr, testdat)
```

```
##                       ME      RMSE       MAE     MPE     MAPE      MASE
## Training set 0.008756236 0.5535951 0.3948974    -Inf      Inf 0.6610624
## Test set     0.774509391 0.9045662 0.7745094 35.9946 43.38318 1.2965369
##                    ACF1 Theil's U
## Training set -0.03293199        NA
## Test set     -0.21056349 0.5722577
```