02 Time series with trend and seasonality components

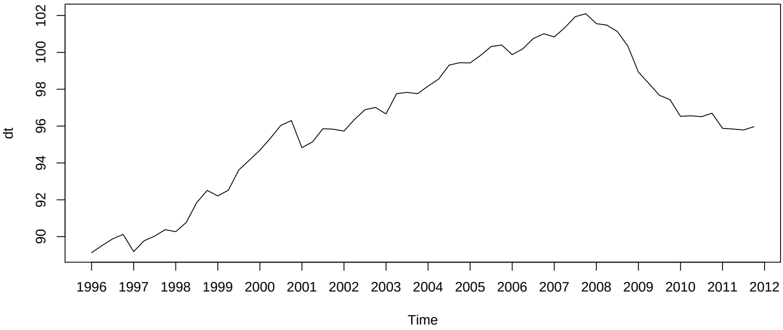
Andrius Buteikis, andrius.buteikis@mif.vu.lt
http://web.vu.lt/mif/a.buteikis/

Time series: Backcasting

We may sometimes want to forecast a series back in time - this is known as *backcasting*.

Let our dataset be of a quarterly retail trade index in the Euro area (17 countries), 1996-2011, covering wholesale and retail trade, and repair of motor vehicles and motorcycles. (Index: 2005 = 100).

```
dt <- fpp::euretail
plot.ts(dt, xaxt = "n")
axis(side = 1, at = c(1996:2012), labels = 1996:2012)
```



Assume that we want to create a forecast for $1995 : Q1 - 1995 : Q4$. To do this, we note that while the **ordering** of a time series is important, the **direction** of a series is less critical.

If our series is modelled via an $ARMA(1,1)$ process:

$$Y_t = \phi_1 Y_{t-1} + \epsilon_t + \theta_1 \epsilon_{t-1}$$

which we can express for Y_{t-1} as:

$$Y_{t-1} = \frac{1}{\phi_1} Y_t - \frac{1}{\phi_1}\epsilon_t - \frac{\theta_1}{\phi_1}\epsilon_{t-1}$$

i.e.:

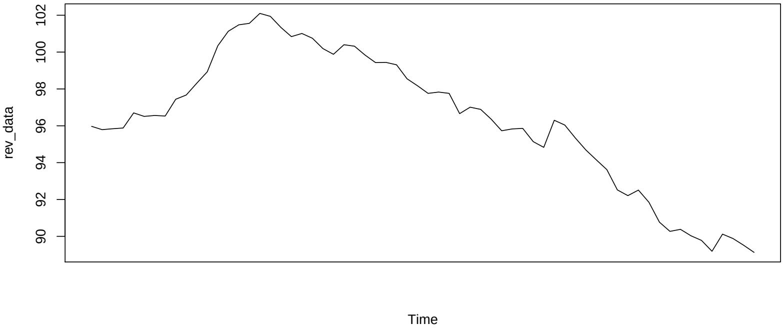
$$Y_t = 1/\phi_1 Y_{t+1} - 1/\phi_1 \epsilon_{t+1} - \theta_1/\phi_1 \epsilon_t$$

Similarly, think back to the $AR(1)$ case wiwth $|\phi| > 1$ - we had no trouble expressing Y_t as a function of *future* value Y_{t+1} and ϵ_{t+1}.

While the above expression is not useful in **forecasting the future**, it is quite useful if we want to **forecast backwards**.

To see why, we can reverse the series:

```
rev_data <- rev(dt)
plot.ts(rev_data, xaxt = "n")
```

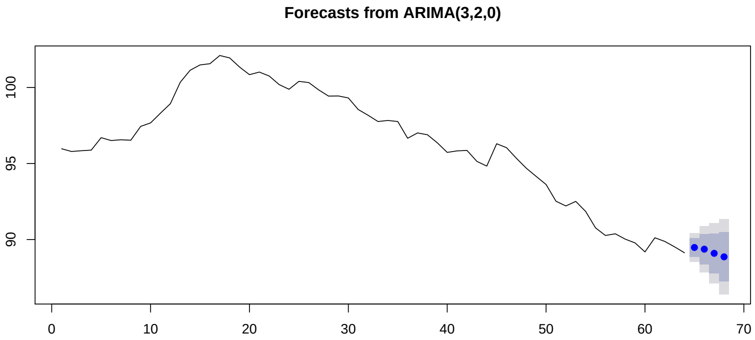


Estimate a model:

```
suppressPackageStartupMessages({
  library(forecast)
  library(fma)
})
mdl <- auto.arima(rev_data)
```

And calculate the forecasts:

```
dt_forc <- forecast(mdl, h = 4)
plot(dt_forc)
```



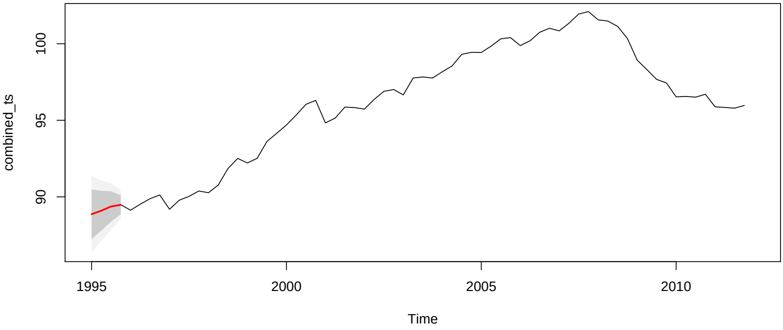
Note: We will learn about $ARMA(p, d, q)$ models in later lectures.

So, for the initial (i.e. non-reversed) series, the predictions are:

```
dt_forc_mean <- ts(rev(dt_forc$mean), start = c(1995, 1), frequency = 4)
combined_ts <- ts(c(dt_forc_mean, dt),
                  start = start(dt_forc_mean), frequency = 4)
```

So, for the initial series, the predictions are:

```
plot.ts(combined_ts, xlim = c(1995, 2012),
        ylim = c(min(dt_forc$lower[, "95%"], dt), 102))
polygon(c(time(dt_forc_mean), rev(time(dt_forc_mean))),
        c(rev(dt_forc$lower[, "95%"]), dt_forc$upper[, "95%"]),
        col = "grey95", border = NA)
polygon(c(time(dt_forc_mean), rev(time(dt_forc_mean))),
        c(rev(dt_forc$lower[, "80%"]), dt_forc$upper[, "80%"]),
        col = "grey80", border = NA)
lines(dt_forc_mean, col = "red", lwd = 2)
```

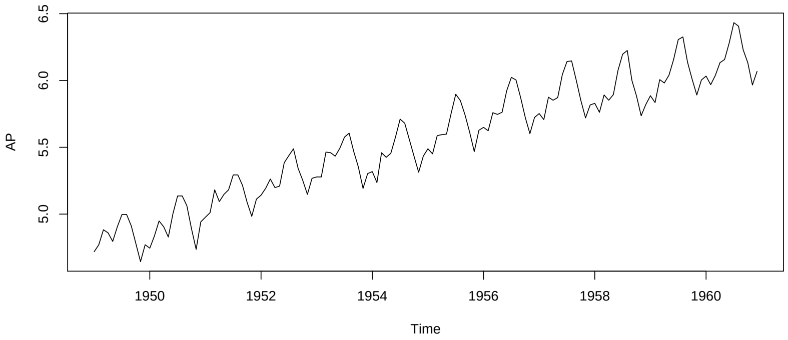


Note that we need to *reverse* the series with `rev()` - both the mean forecast, as well as the lower an upper confidence intervals. Note that to shade the area we have used `polygon()`, where the ordering of the data needed to be specified `c(time(dt_forc_mean), rev(time(dt_forc_mean)))` (the upepr and lower intervals).

Time Series: Forecasting and Backcasting for missing data

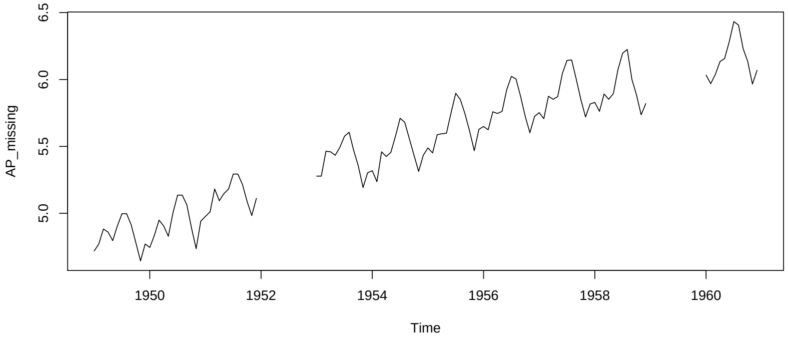
Assume that we have the following **logarithms** of a *monthly* series, where we see that a couple of *years* of data are missing:

```
data(airpass)
AP <- log(airpass)
AP <- ts(AP, start = c(1949, 1), freq = 12)
plot.ts(AP)
```



For interests sake, assume that some of the data is missing - some near the middle, and some near the end:

```
AP_missing <- AP
AP_missing[(12*3 + 1):(12*4)] <- NA
AP_missing[(12*10 + 1):(12*11)] <- NA
plot.ts(AP_missing)
```



We see that we are missing some data - what do we do now?

- If we try to only use the most recent available data - we only have around a year's worth.
- If we try to only use the data in the middle - we do not have the most recent data.

Solution: Combine backcasting and forecasting!

We see that there a number of ways that we can begin - as we know from cross-sectional data - the more data we have, the more accurate our models (and their predictions). Furthermore, one we have the forecasts - we have "bridged" the gap between our historical data with the expected value and can now use more data for prediction. Hence we *could* employ a number of different strategies:

- Strategy 1: Use the middle data to **backcast** and **forecast** the missing series;
- Strategy 2: Use the middle data to **backcast** the missing series. Then use the available historical data, **including the predictions**, to **forecast** the missing series;
- Strategy 3: Use the middle data to **forecast** the missing series. Then use the available historical data, **including the predictions**, to **backcast** the missing series.

We will use the `auto.arima()`, the exponential smoothing `ets()` methods, as well as a regression via OLS and compare the results for **Streategy 1**. Other strategies may provide a similar fit for the missing data in terms of accuracy (**you should try the other methods and verify this!**).

Strategy 1: via auto.arima()

We take the middle of the data, which is not missing and create an additional reverse of the series:

```
dt_middle <- ts(AP_missing[(12*4+1):(12*10)],
                    start = time(AP_missing)[(12*4+1)],
                    frequency = frequency(AP_missing))
dt_middle_rev <- ts(rev(dt_middle), start = start(dt_middle),
                        frequency = frequency(dt_middle))
```

It is very important that you specify the seasonal frequency correctly. we then estimate two different models for the series and its reverse:

```
mdl_arima_middle_frw <- auto.arima(dt_middle)
mdl_arima_middle_bck <- auto.arima(dt_middle_rev)
```

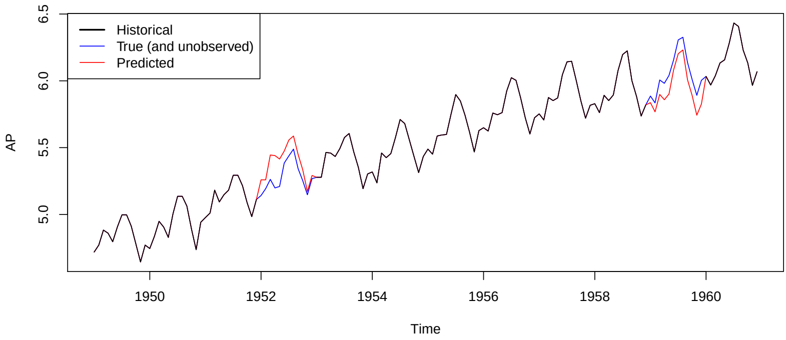
and calculate the forecasts:

```
middle_forc_arima_frw <- forecast(mdl_arima_middle_frw, h = 12)
middle_forc_arima_bck <- forecast(mdl_arima_middle_bck, h = 12)
```

To make plotting a bit easier, we will fill in the missing values with the forecasted mean values:

```
AP_filled_arima <- AP_missing
AP_filled_arima[(12*10 + 1):(12*11)] <- middle_forc_arima_frw$mean
AP_filled_arima[(12*3 + 1):(12*4)]  <- rev(middle_forc_arima_bck$mean)
```

```r
plot(AP, col = "blue")
lines(AP_filled_arima, col = "red")
lines(AP_missing, col = "black")
legend("topleft",
       legend = c("Historical", "True (and unobserved)", "Predicted"),
       col = c("black", "blue", "red"), lwd = c(2, 1, 1))
```



Strategy 1: via `ets()`

we use the exponential smoothing methods on the two different models for the series and its reverse:

```
mdl_ets_middle_frw <- ets(dt_middle)
mdl_ets_middle_bck <- ets(dt_middle_rev)
```

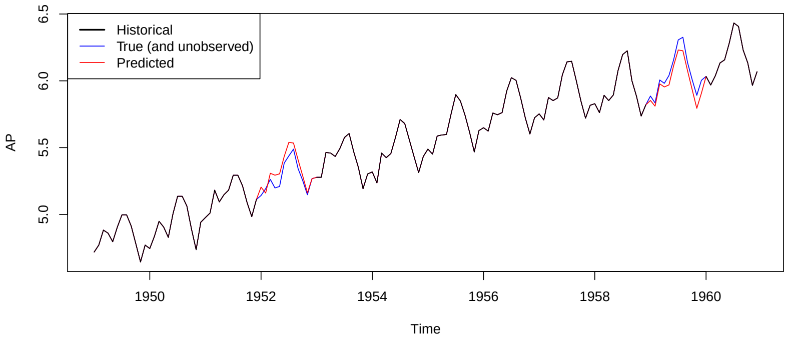
and calculate the forecasts:

```
middle_forc_ets_frw <- forecast(mdl_ets_middle_frw, h = 12)
middle_forc_ets_bck <- forecast(mdl_ets_middle_bck, h = 12)
```

Finally, we fill the forecasts in place of the missing observations:

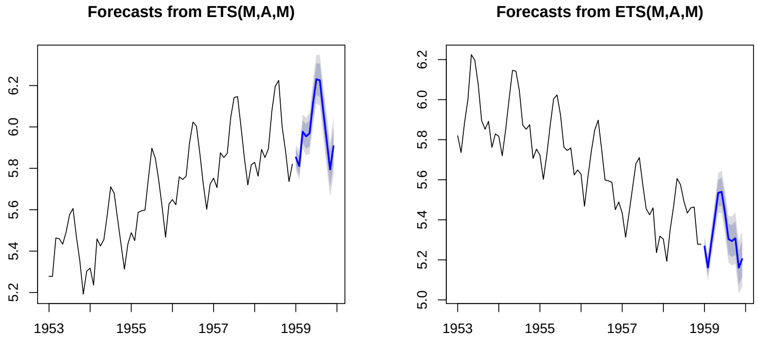
```
AP_filled_ets <- AP_missing
AP_filled_ets[(12*10 + 1):(12*11)] <- middle_forc_ets_frw$mean
AP_filled_ets[(12*3 + 1):(12*4)]   <- rev(middle_forc_ets_bck$mean)
```

```r
plot(AP, col = "blue")
lines(AP_filled_ets, col = "red")
lines(AP_missing, col = "black")
legend("topleft",
       legend = c("Historical", "True (and unobserved)", "Predicted"),
       col = c("black", "blue", "red"), lwd = c(2, 1, 1))
```



The forecasts for the two separate parts are:

```
par(mfrow = c(1, 2))
plot(middle_forc_ets_frw)
plot(middle_forc_ets_bck)
```



Note that the date is the same only for plotting convenience purposes.

The letters are for the **E**rror-**T**rend-**s**easonality smoothing type. It appears that the best model would be a **M**ultiplicative for the error, an **A**dditive for the trend, and a **M**ultiplicative for the seasonality.

What would happen if we **forget to specify the frequency of our data?**

```r
print(head(dt_middle, 6))
```

```
##            Jan      Feb      Mar      Apr      May      Jun
## 1953 5.278115 5.278115 5.463832 5.459586 5.433722 5.493061
```

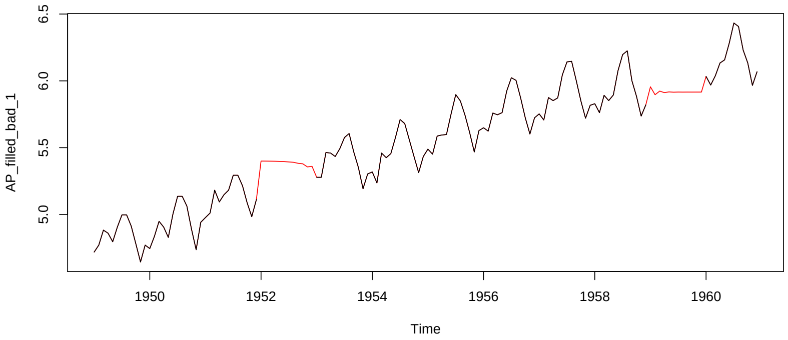
```r
dt_middle_no_freq <- ts(c(dt_middle))
print(head(dt_middle_no_freq, 6))
```

```
## Time Series:
## Start = 1
## End = 6
## Frequency = 1
## [1] 5.278115 5.278115 5.463832 5.459586 5.433722 5.493061
```

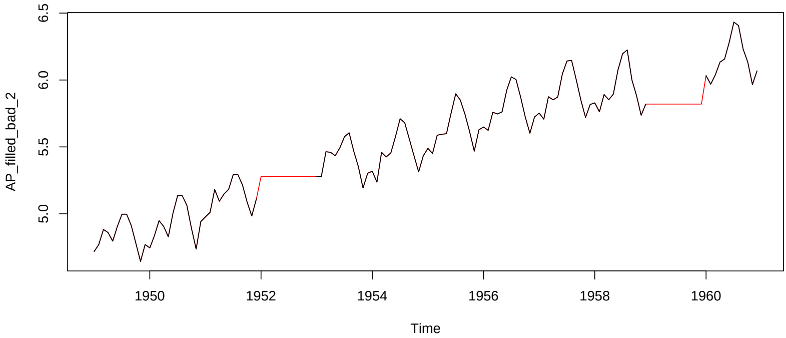
```r
bad_middle_forc_arima_frw <- forecast(auto.arima(dt_middle_no_freq), h = 12)
bad_middle_forc_arima_bck <- forecast(auto.arima(rev(dt_middle_no_freq)),h=12)
#
bad_middle_forc_ets_frw <- forecast(ets(dt_middle_no_freq), h = 12)
bad_middle_forc_ets_bck <- forecast(ets(rev(dt_middle_no_freq)), h = 12)
```

```r
AP_filled_bad_1 <- AP_missing
AP_filled_bad_1[(12*10 + 1):(12*11)] <- bad_middle_forc_arima_frw$mean
AP_filled_bad_1[(12*3 + 1):(12*4)]   <- rev(bad_middle_forc_arima_bck$mean)
#
AP_filled_bad_2 <- AP_missing
AP_filled_bad_2[(12*10 + 1):(12*11)] <- bad_middle_forc_ets_frw$mean
AP_filled_bad_2[(12*3 + 1):(12*4)]   <- rev(bad_middle_forc_ets_bck$mean)
```

```r
plot(AP_filled_bad_1, col = "red")
lines(AP_missing, col = "black")
```



```r
plot(AP_filled_bad_2, col = "red")
lines(AP_missing, col = "black")
```



To reiterate: **do NOT forget to specify the frequency of our data!** Otherwise, it some cases it may seem that neither *ARIMA*, nor exponential smoothing are adequate, when all we needed to do was specify the correct data frequency!

Strategy 1: via `lm()`

Firstly, we create the seasonal dummy variables:

```
tmp <- seasonaldummy(AP_missing)
print(head(tmp, 4))
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
## [1,]   1   0   0   0   0   0   0   0   0   0   0
## [2,]   0   1   0   0   0   0   0   0   0   0   0
## [3,]   0   0   1   0   0   0   0   0   0   0   0
## [4,]   0   0   0   1   0   0   0   0   0   0   0
```

Then the trend:

```
t = time(AP_missing)
print(head(t, 4))
```

```
##           Jan      Feb      Mar      Apr
## 1949 1949.000 1949.083 1949.167 1949.250
```

and combine into a single dataset:

```
dt <- data.frame(cbind(AP_missing, t, tmp))
colnames(dt) <- c("log_Y", "t", paste0("dm_", 1:ncol(tmp)))
print(head(dt, 4))
```

```
##      log_Y        t dm_1 dm_2 dm_3 dm_4 dm_5 dm_6 dm_7 dm_8 dm_9 dm_10 dm_11
## 1 4.718499 1949.000    1    0    0    0    0    0    0    0    0     0     0
## 2 4.770685 1949.083    0    1    0    0    0    0    0    0    0     0     0
## 3 4.882802 1949.167    0    0    1    0    0    0    0    0    0     0     0
## 4 4.859812 1949.250    0    0    0    1    0    0    0    0    0     0     0
```

Note that in `lm()` by default rows with `NA` values in the dependent and/or the explanatory variables, are removed. This would mean that our beginning and end of the series would be included.

To keep in line with the data used in `auto.arima` and `ets`, we will only use the middle of the series.

From the previous lecture, we know that the logarithm of `airpass` for the dependent variable, a quadratic trend and dummy variables provide the best model fit.

```
dt_middle <- dt[(12*4+1):(12*10), ]
```

```
mdl_ols_middle_frw <- lm(log_Y ~ t + I(t^2)+.,
                         dt_middle[1:nrow(dt_middle), ])
mdl_ols_middle_bck <- lm(log_Y ~ t + I(t^2)+.,
                         dt_middle[rev(1:nrow(dt_middle)), ])
```

Note that for the regression case - we can predict in both directions!

Furthermore, since the data ordering does not matter for the regression - both of our models are exactly the same!

```r
round(coef(summary(mdl_ols_middle_frw)), 4)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19843.4959  8000.9644 -2.4801   0.0161
## t               20.1803     8.1811  2.4667   0.0166
## I(t^2)          -0.0051     0.0021 -2.4525   0.0172
## dm_1             0.0378     0.0276  1.3673   0.1768
## dm_2            -0.0150     0.0276 -0.5446   0.5881
## dm_3             0.1379     0.0276  5.0013   0.0000
## dm_4             0.1104     0.0276  4.0048   0.0002
## dm_5             0.1149     0.0275  4.1739   0.0001
## dm_6             0.2469     0.0275  8.9722   0.0000
## dm_7             0.3505     0.0275 12.7425   0.0000
## dm_8             0.3350     0.0275 12.1857   0.0000
## dm_9             0.1810     0.0275  6.5852   0.0000
## dm_10            0.0402     0.0275  1.4643   0.1485
## dm_11           -0.1061     0.0275 -3.8617   0.0003
```

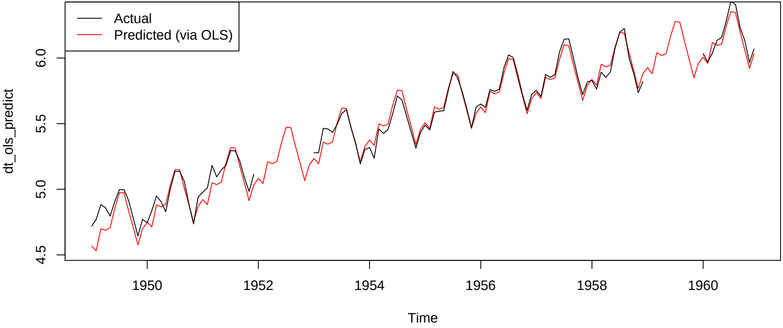
```r
round(coef(summary(mdl_ols_middle_bck)), 4)
```

```
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -19843.4959  8000.9644 -2.4801   0.0161
## t               20.1803     8.1811  2.4667   0.0166
## I(t^2)          -0.0051     0.0021 -2.4525   0.0172
## dm_1             0.0378     0.0276  1.3673   0.1768
## dm_2            -0.0150     0.0276 -0.5446   0.5881
## dm_3             0.1379     0.0276  5.0013   0.0000
## dm_4             0.1104     0.0276  4.0048   0.0002
## dm_5             0.1149     0.0275  4.1739   0.0001
## dm_6             0.2469     0.0275  8.9722   0.0000
## dm_7             0.3505     0.0275 12.7425   0.0000
## dm_8             0.3350     0.0275 12.1857   0.0000
## dm_9             0.1810     0.0275  6.5852   0.0000
## dm_10            0.0402     0.0275  1.4643   0.1485
## dm_11           -0.1061     0.0275 -3.8617   0.0003
```

An upside is that in order to calculate the predictions, we do not need to create a new data matrix for the trend and seasonality!

```
dt_ols_predict <- predict(mdl_ols_middle_frw, newdata = dt)
dt_ols_predict <- ts(dt_ols_predict, start = start(AP_missing),
                     frequency = frequency(AP_missing))
```

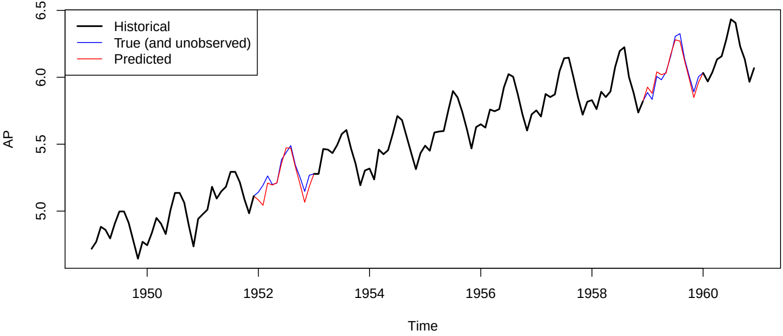
```
plot(dt_ols_predict, col = "red")
lines(AP_missing)
legend("topleft", legend = c("Actual", "Predicted (via OLS)"),
       lty = 1, col = c("black", "red"))
```



Or, keeping only the missing data predictions:

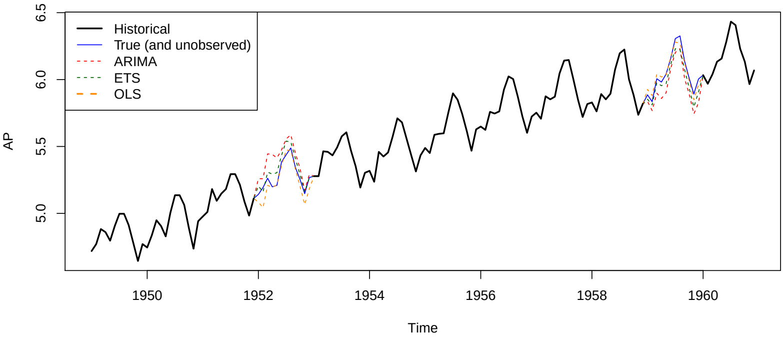
```
AP_filled_ols <- AP_missing
AP_filled_ols[is.na(AP_filled_ols)] <- dt_ols_predict[is.na(AP_filled_ols)]
```

```
plot(AP, col = "blue")
lines(AP_filled_ols, col = "red")
lines(AP_missing, lwd = 2)
legend("topleft",
       legend = c("Historical", "True (and unobserved)", "Predicted"),
       col = c("black", "blue", "red"), lwd = c(2, 1, 1))
```



Comparing the results

```
plot(AP, col = "blue")
lines(AP_filled_arima, col = "red", lty = 2)
lines(AP_filled_ets, col = "darkgreen", lty = 2)
lines(AP_filled_ols, col = "darkorange", lty = 2)
lines(AP_missing, lwd = 2)
legend("topleft",
       legend = c("Historical", "True (and unobserved)", "ARIMA", "ETS", "OLS"),
       col = c("black", "blue", "red", "darkgreen", "darkorange"),
       lwd = c(2, 1, 1, 1), lty = c(1, 1, 2, 2, 2))
```



Visually, it is difficult to say which method is superior.

The errors for the missing data:

```
err_arima <- (AP - AP_filled_arima)[is.na(AP_missing)]
err_ets <- (AP - AP_filled_ets)[is.na(AP_missing)]
err_ols <- (AP - AP_filled_ols)[is.na(AP_missing)]
```

Can be used to calculate the:

- Mean absolute error (MAE):

$$MAE = \frac{1}{T}\sum_{t=1}^{T} |Y_t - \widehat{Y}_t|$$

- Mean absolute percentage error (MAPE):

$$MAPE = \frac{1}{T}\sum_{t=1}^{T} \left|\frac{Y_t - \widehat{Y}_t}{Y_t}\right|$$

- Mean absolute scaled error (MASE):

$$MASE = \frac{1}{T}\sum_{t=1}^{T} \frac{|Y_t - \widehat{Y}_t|}{\frac{1}{T-1}\sum_{t=2}^{T} |Y_t - Y_{t-1}|}$$

```r
MAE <- data.frame(ARIMA = mean(abs(err_arima)), ETS = mean(abs(err_ets)),
                  OLS = mean(abs(err_ols)))
print(MAE)
```

```
##       ARIMA        ETS        OLS
## 1 0.1121146 0.05741011 0.04000328
```

```r
MAPE <- data.frame(ARIMA = mean(abs(err_arima / AP[is.na(AP_missing)])),
                   ETS = mean(abs(err_ets / AP[is.na(AP_missing)])),
                   OLS = mean(abs(err_ols / AP[is.na(AP_missing)])))
print(MAPE)
```

```
##        ARIMA        ETS        OLS
## 1 0.01991718 0.01010996 0.00723825
```

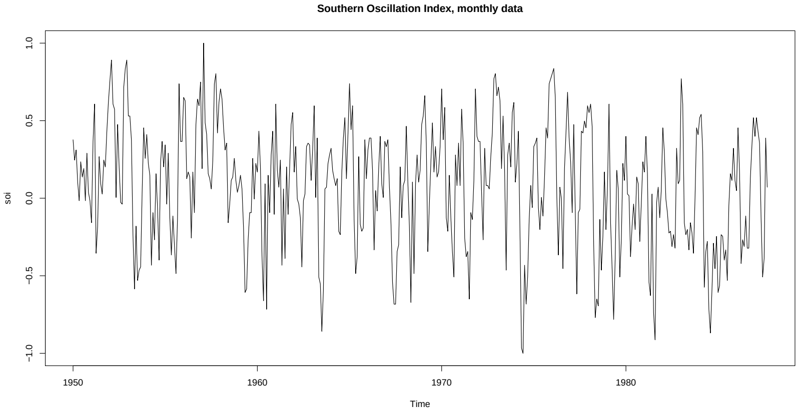
```r
denom <- mean(abs(diff(AP[is.na(AP_missing)])))
MASE <- data.frame(ARIMA = mean(abs(err_arima) / denom),
                   ETS = mean(abs(err_ets) / denom),
                   OLS = mean(abs(err_ols) / denom))
print(MASE)
```

```
##       ARIMA       ETS       OLS
## 1 0.9568328 0.4899616 0.3414046
```

From the above **accuracy measures** we see that the *OLS* model provides a better fit **for the missing data** (we can also see a simialr result from the plot, though it is not quite as clear).

Question: Would the previous methods give the same results in terms of accuracy if there are no clear seasonal patterns and the random shocks have a more pronounced effect (and some of the data would be missing)?

```r
soi <- astsa::soi
plot(soi, main = "Southern Oscillation Index, monthly data")
```



Answer: It is likely that a *SARMA* model, in combination with OLS or *ETS*, may prove useful, if the shocks are autocorrelated. Other methods for replacing missing data include EM and MICE and others.

Takeaway

The key takeaway is:

- There are no universal methods, which would be best suited to **all problems**;
- It is vital to know the frequency of the series - the automated procedures may fail in case of seasonality effects;
- OLS can be succesfully applied to time series data and in some cases be superior to *ARIMA* and exponential smoothing methods, or at least when compared to the automated model selection procedures.