R has extensive facilities for analysing time series data. This handout describes the loading and creation of a time series, seasonal decomposition, modeling with exponential and ARIMA models, and forecasting with the forecast package.

Loading or creating a Time Series

The **ts()** function can also convert a numeric vector into an R time series object. The format is **ts(***vector*, **start=, end=, frequency=)** where start and end are the times of the first and last observation and frequency is the number of observations per unit time (1=annual, 4=quartly, 12=monthly, etc.).

In the basic packages of R (already pre-installed), you can call the data set "AirPassengers" by the following. This data set is a time series, i.e. a series of values counted at regular time intervals (monthly in this case).

```
> data("AirPassengers")
>
> AirPassengers
     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1949 112 118 132 129 121 135 148 148 136 119 104 118
1950 115 126 141 135 125 149 170 170 158 133 114 140
1951 145 150 178 163 172 178 199 199 184 162 146 166
1952 171 180 193 181 183 218 230 242 209 191 172 194
1953 196 196 236 235 229 243 264 272 237 211 180 201
1954 204 188 235 227 234 264 302 293 259 229 203 229
1955 242 233 267 269 270 315 364 347 312 274 237 278
1956 284 277 317 313 318 374 413 405 355 306 271 306
1957 315 301 356 348 355 422 465 467 404 347 305 336
1958 340 318 362 348 363 435 491 505 404 359 310 337
1959 360 342 406 396 420 472 548 559 463 407 362 405
1960 417 391 419 461 472 535 622 606 508 461 390 432
```

decomposed using the.We can check the starting date, the end date of the object "AirPassengers" by the following commands. We can also obtain the frequency of the time series:

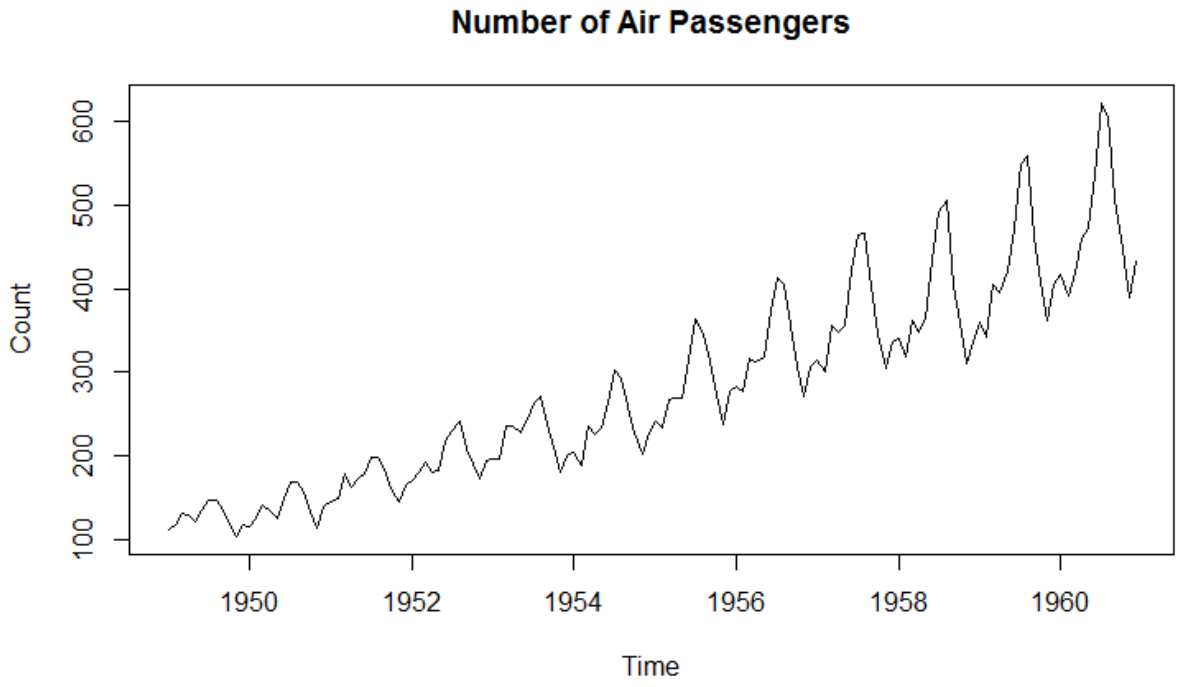
```
> start(AirPassengers)
[1] 1949    1
> end(AirPassengers)
[1] 1960   12
> frequency(AirPassengers)
[1] 12
```

We start by defining the time series first of all with the exact starting date, end date and frequency:

```
> myvector<-AirPassengers
> myts<-ts(myvector, start=c(1949,1), end=c(1960,12),frequency=12)
```

In order to plot the time series, we have:

```
> plot(myts,main="Number of Air Passengers", ylab="Count")
```



Seasonal Decomposition

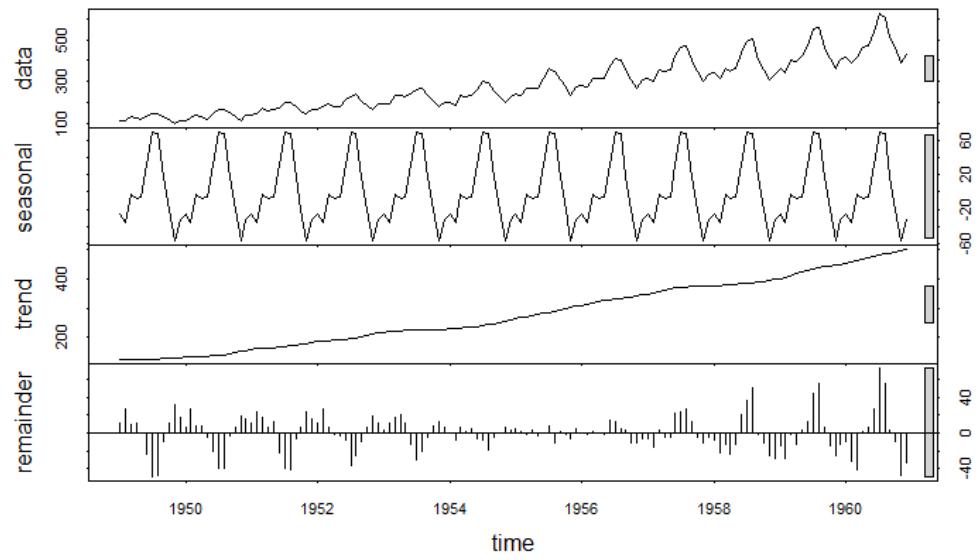
We can proceed to a decomposition of the time series before doing any modelling:

```
> fit<-stl(myts,s.window="period")
> plot(fit)
```

The first graph on the figure below will display the data (or time series value). The second graph will display the seasonal component and the third graph will display the trend of the time series. The fourth graph consists of the error term, that is, the true value minus the seasonal value and the trend value (called remainder).

Any actual value at a given time of the time series “AirPassengers” can be written as the sum of the trend value + seasonal value + an error term:

VALUE = SEASONAL VALUE + TREND VALUE + ERROR TERM



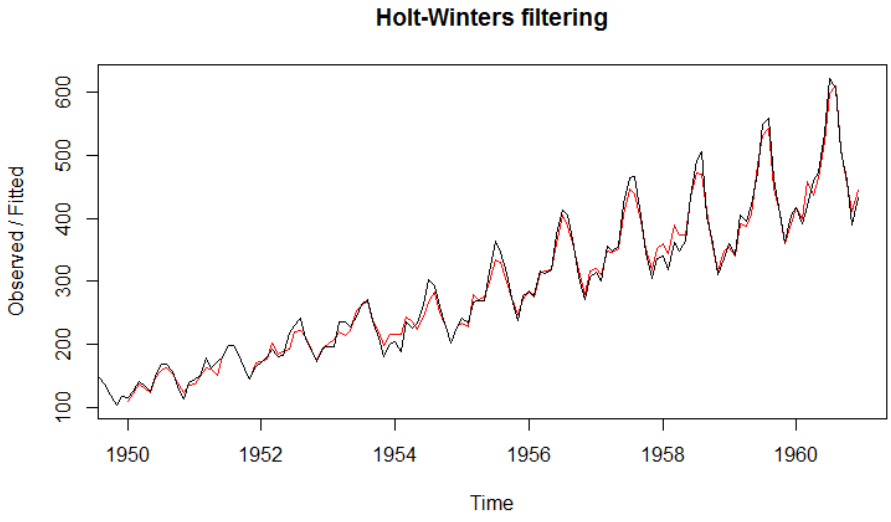
Exponential Models

Both the **HoltWinters()** function in the base installation, and the **ets()** function in the forecast package, can be used to fit exponential models.

```
> fit<-HoltWinters(myts)
> fit
Holt-Winters exponential smoothing with trend and additive seasonal component.

Call:
HoltWinters(x = myts)

Smoothing parameters:
 alpha: 0.2479595
 beta : 0.03453373
 gamma: 1
> plot(fit)
```



ARIMA Models and automated forecast

The **arima()** function can be used to fit an autoregressive integrated moving averages model. The **auto.arima()** function will fit the best ARIMA model of your time series:

```
> library(forecast)
> fit<-auto.arima(myts)
```

Testing the accuracy of the model and Forecasting

We can calculate the mean of the squared errors (MSE). The mean of the squared errors is often the number reported when checking for the "goodness of fit". For each observation of the time series (actual value), there is a predicted value of the time series (predicted value from the model). The error for that observation is then:

Error = Actual value – Predicted value

If the actual value is far from the predicted value, the resulting error will be big and therefore will increase the mean of the squared errors. If the mean of the squared errors is a big number, then we can conclude that the model is not a good fit. In R, the square root of the MSE is given by the value "RMSE" below:

```
> accuracy(fit)
                    ME     RMSE      MAE       MPE     MAPE      MASE         ACF1
Training set 1.3423 10.84619 7.86754 0.420698 2.800458 0.245628 -0.00124847
```

In this next command, we can forecast the 10 next future values (until October 1961)

```
> library(forecast)
> plot(forecast(fit,10))
```