Dec 4 Notes

Hongdou Li

12/4/2018

BIG PICTURE

Model Data

Univariate

- stationary
    - AR/MA/ARMA
    - SES
- Non-Stationary
    - Trend
        - Arima
        - DES
    - Seasonal
        - SARIMA
        - TES

Multivariate

- Exogenous
    - SARIMA
- Endogenous
    - VAR
- VARX

Choosing a Model

- Goodness-of-fit
    - AIC
    - log-lik
    - $\hat{\sigma}^2$
- Predictive Accuracy
    - RMSE
    - MAE
    - test
- Model Assumptions
    - Residual Diagnostics

Forecast

Imputing Time Series

As with any modeling endeavor, time series in practice often have missing observations. In order to fit a time series model, we require **complete** (observe everything) data. To ensure this we perform imputation.

- Many methods of imputation exist, but we need to be careful to use ones that are appropriate for time series data.
- effective methods of imputation account for the time and correlation structure of time series data. The most effective method depends on whether the data is stationary or whether it has trend and/or seasonality.

Non-TS-Specific Methods

Fill gaps with the measure of center calculated from the observed data.

- Mean imputation
- Median imputation
- Mode imputation

(obvious increasing trend) if there is a missing value, these don't work well in the presence of trend.

(when time series is typically flat or stationary) these approaches may work fine if the data is stationary.

- Random-Sample Imputation

  Draw a random observation from the existing time series and use this to fill gaps

  for the same reasons as above. This approach may be useful for stationary data, but not otherwise.

TS-Specific Methods

These exploit the strong serial correlation often exhibited by time series.

- Last observation carried forward (LOCF)
- Next observation carried backward (NOCB)

* This does not as effective for large gaps

* This doesn't do well if adjacent observations are very different. This is common in the presence of strong seasonal effects

- Interpolation (linear or polynomial)

  fit a straight line (or some higher order polynomial) across gaps in the data.

- Seasonal Adjustment + Interpolation

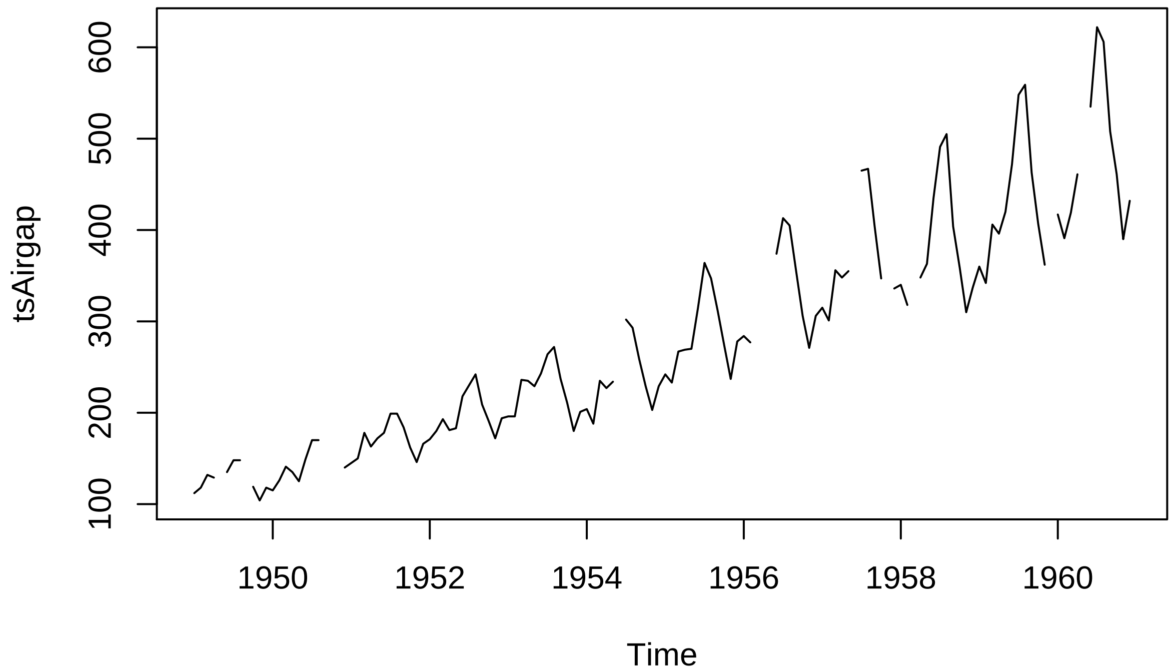
  => De-seasonalize the data (which, can be done even with missing observations)

  => Impute the missing data by interpolation

  => Once the missing data is imputed , we re-seasonalize.

```r
library(imputeTS)
```

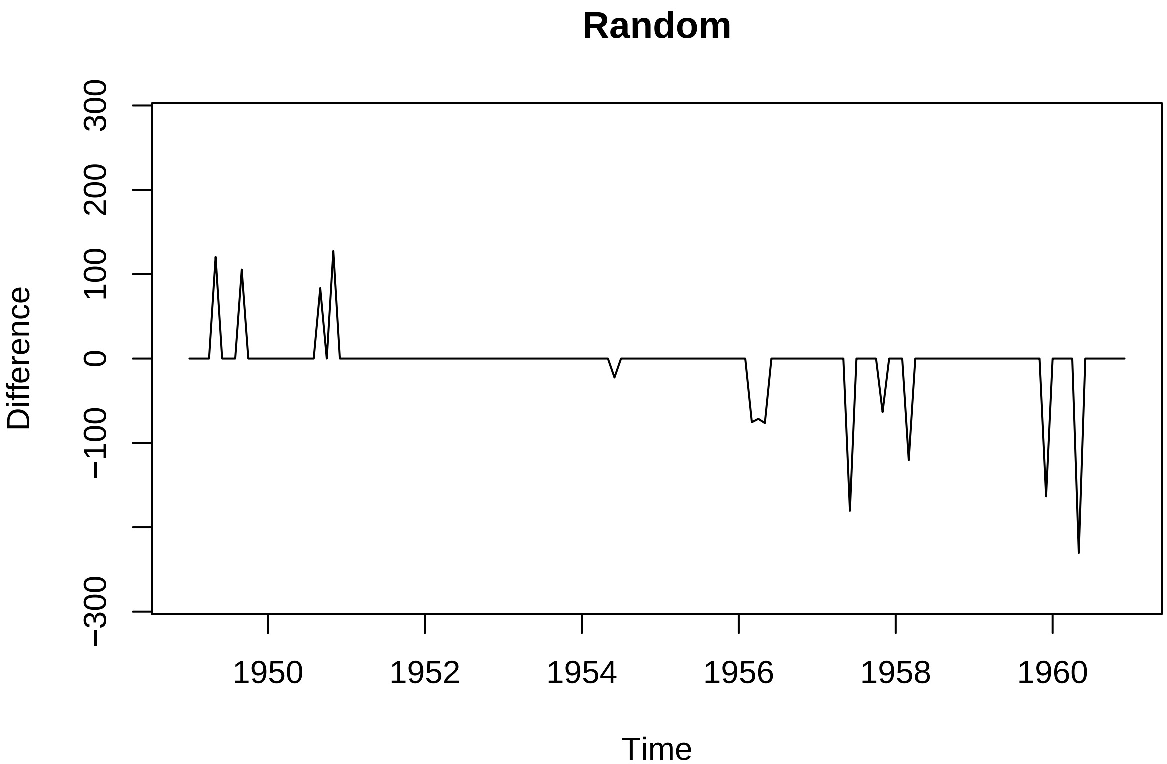
```r
plot(tsAirgap)
```



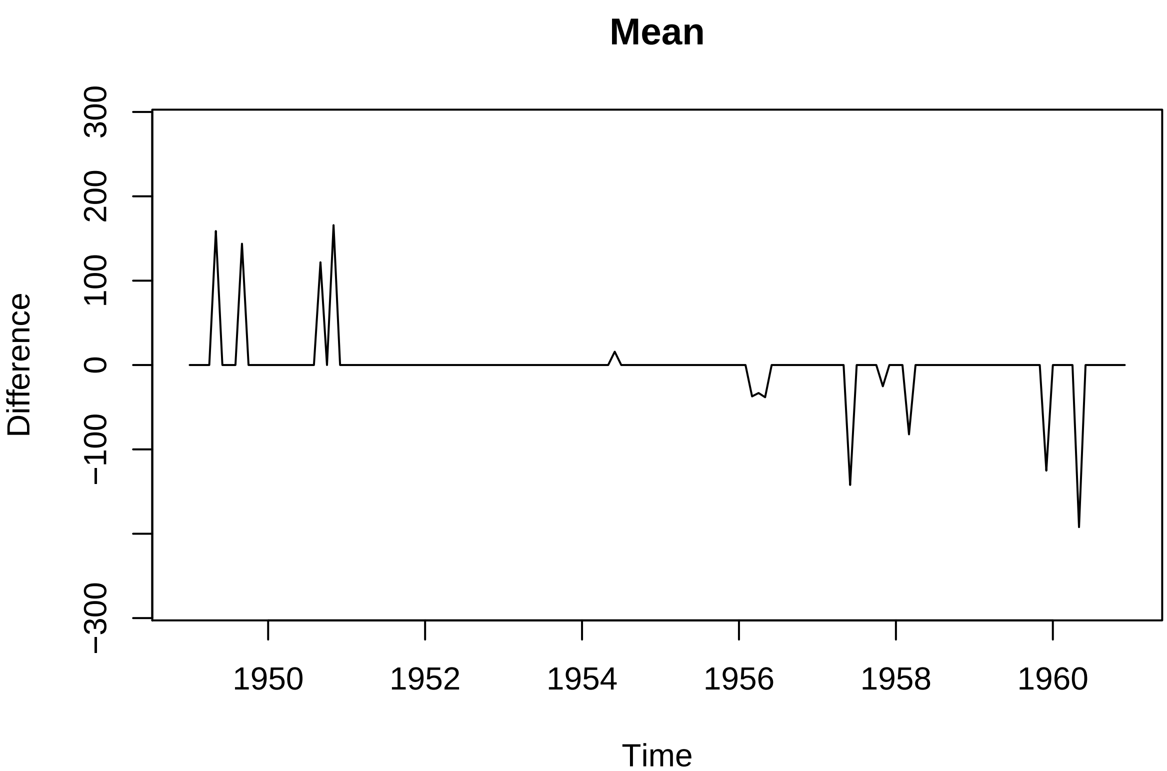

```r
statsNA(tsAirgap)
```

```
## [1] "Length of time series:"
## [1] 144
## [1] "-------------------------"
## [1] "Number of Missing Values:"
## [1] 13
## [1] "-------------------------"
## [1] "Percentage of Missing Values:"
## [1] "9.03%"
## [1] "-------------------------"
## [1] "Stats for Bins"
## [1] "  Bin 1 (36 values from 1 to 36) :      4 NAs (11.1%)"
## [1] "  Bin 2 (36 values from 37 to 72) :      1 NAs (2.78%)"
## [1] "  Bin 3 (36 values from 73 to 108) :      5 NAs (13.9%)"
## [1] "  Bin 4 (36 values from 109 to 144) :      3 NAs (8.33%)"
## [1] "-------------------------"
## [1] "Longest NA gap (series of consecutive NAs)"
## [1] "3 in a row"
## [1] "-------------------------"
## [1] "Most frequent gap size (series of consecutive NA series)"
## [1] "1 NA in a row (occuring 10 times)"
## [1] "-------------------------"
## [1] "Gap size accounting for most NAs"
## [1] "1 NA in a row (occuring 10 times, making up for overall 10 NAs)"
## [1] "-------------------------"
## [1] "Overview NA series"
## [1] "  1 NA in a row: 10 times"
## [1] "  3 NA in a row: 1 times"
```

```r
# Random Imputation
set.seed(1)
plot(na.random(tsAirgap) - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPassengers)), ylab = "I
```



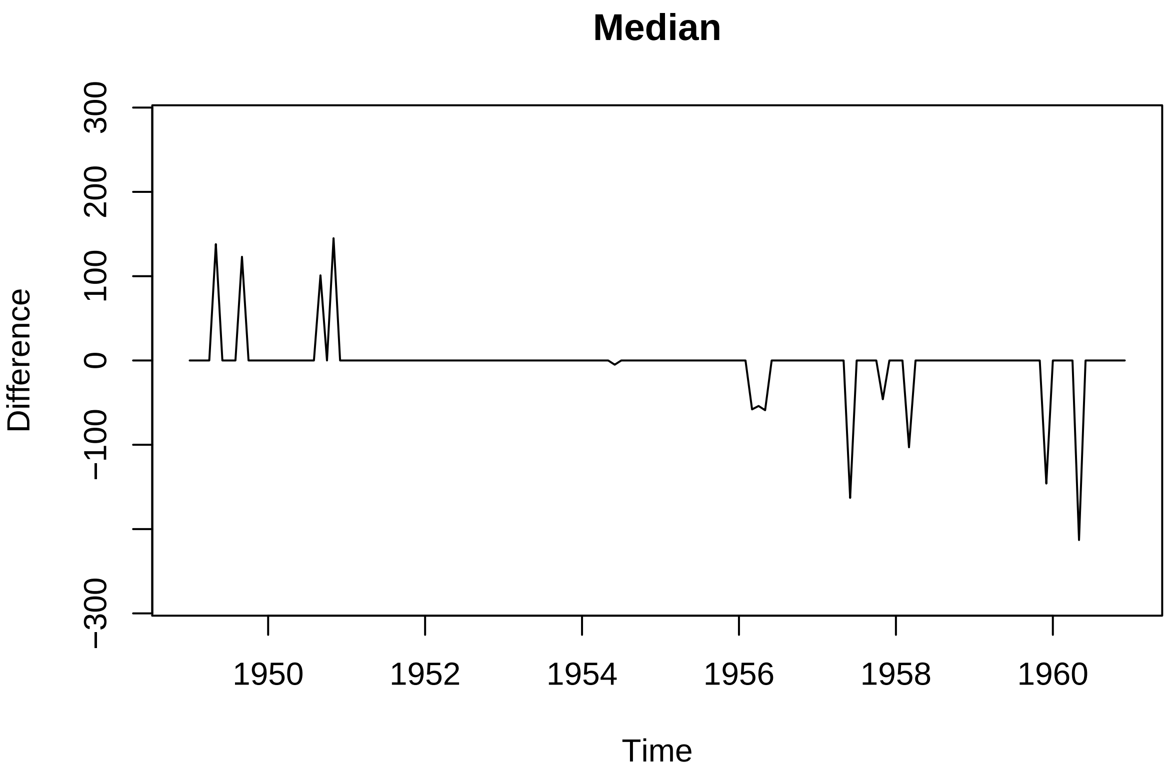

```
mean((na.random(tsAirgap) - AirPassengers)^2)

## [1] 1208.492

# Mean Imputation
plot(na.mean(tsAirgap, option = "mean") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPasseng
```



```r
mean((na.mean(tsAirgap, option = "mean") - AirPassengers)^2)
```

```
## [1] 1198.903
```

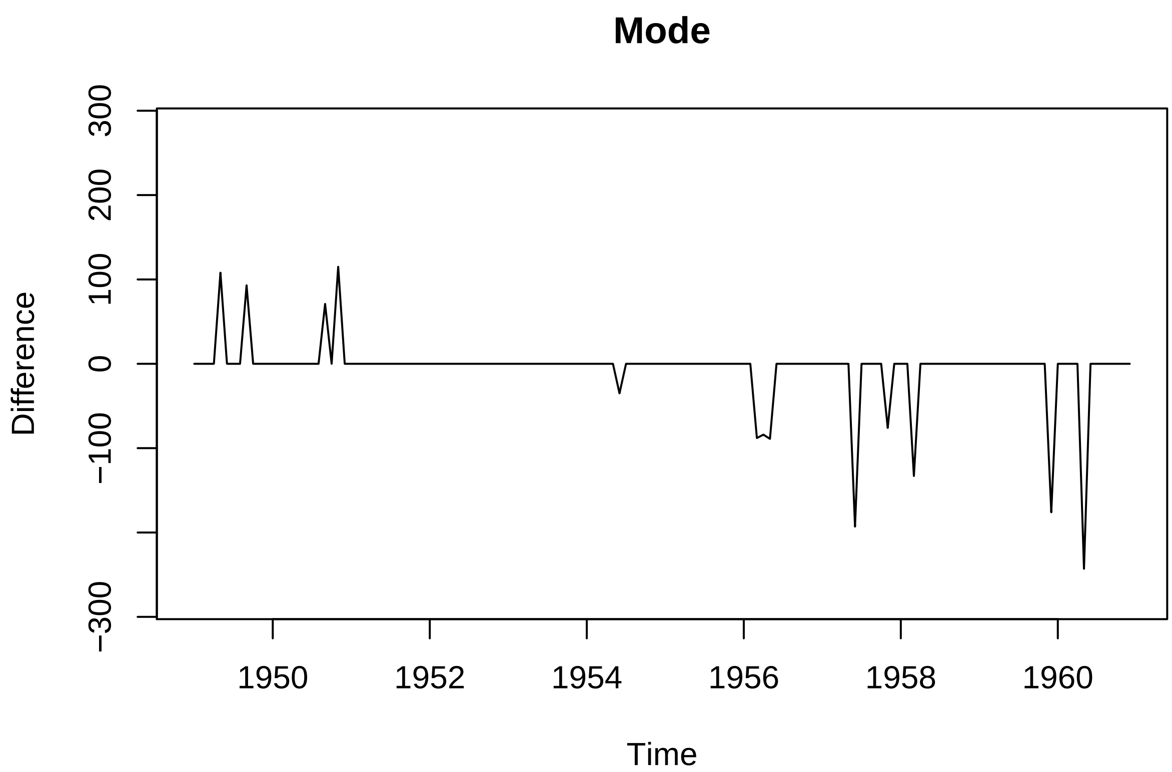
```r
# Median Imputation
plot(na.mean(tsAirgap, option = "median") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPasser
```




```r
mean((na.mean(tsAirgap, option = "median") - AirPassengers)^2)
```

```
## [1] 1258.083
```

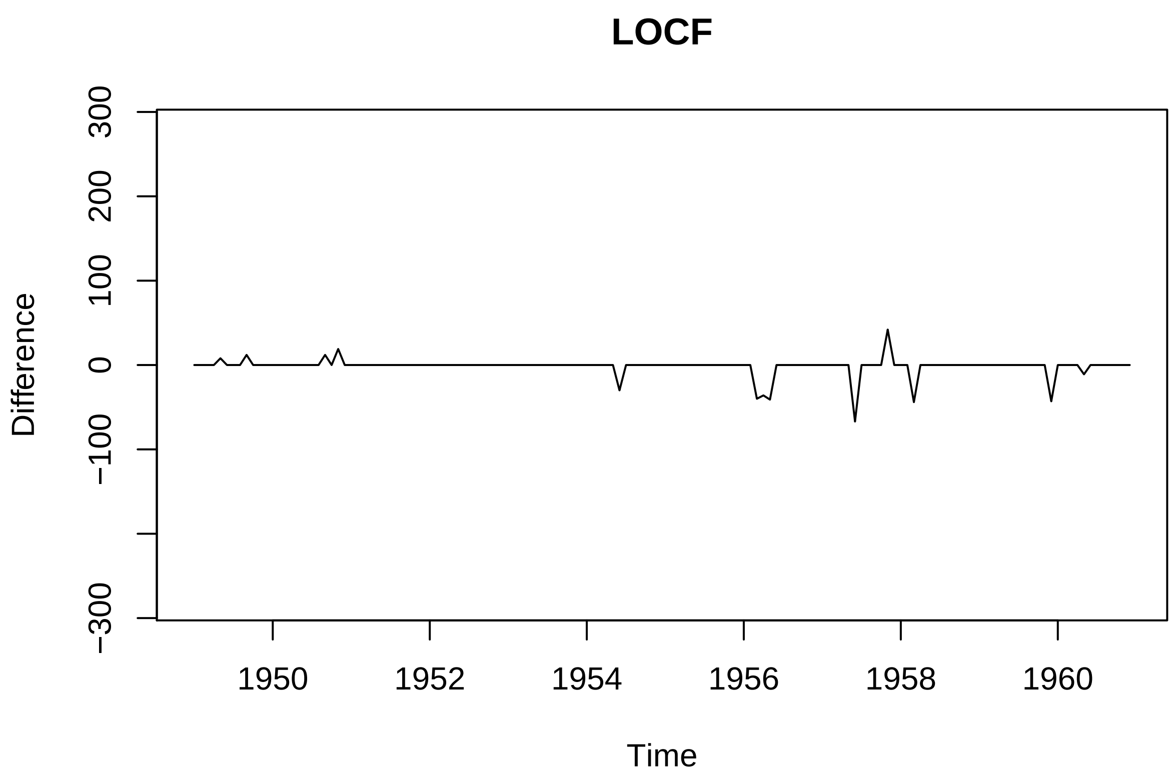
```r
# Mode Imputation
plot(na.mean(tsAirgap, option = "mode") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPasseng
```




```
mean((na.mean(tsAirgap, option = "mode") - AirPassengers)^2)
```

```
## [1] 1481
```

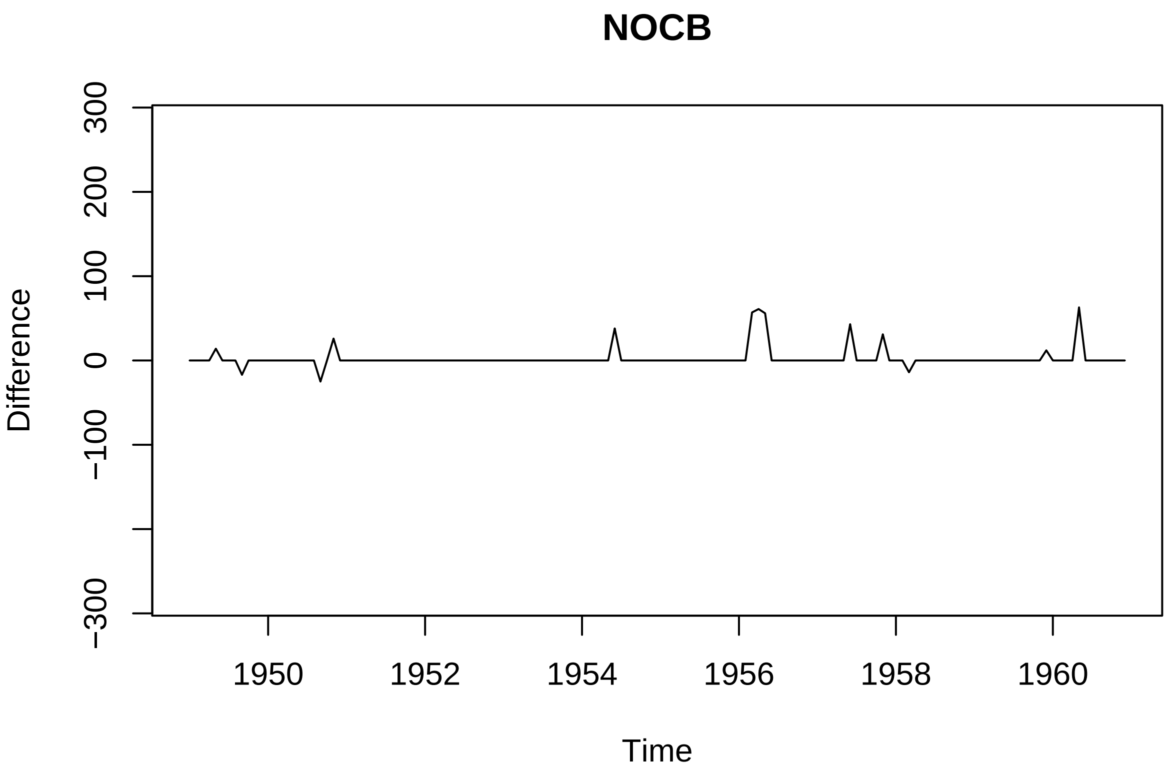
```
# Last Observartion Carried Forward
plot(na.locf(tsAirgap, option = "locf") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPasseng
```



```r
mean((na.locf(tsAirgap, option = "locf") - AirPassengers)^2)
```

```
## [1] 113.5347
```

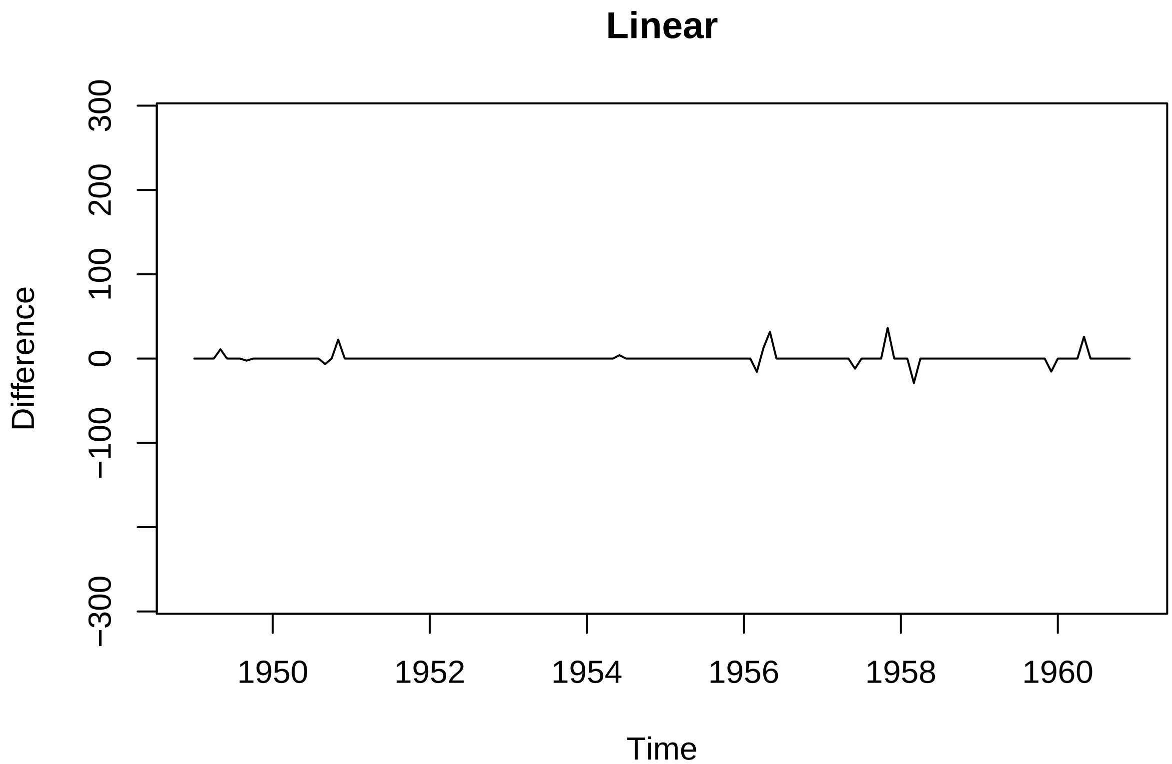
```r
# Next Observartion Carried Backward
plot(na.locf(tsAirgap, option = "nocb") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPasseng
```




```r
mean((na.locf(tsAirgap, option = "nocb") - AirPassengers)^2)
```

```
## [1] 142.0486
```

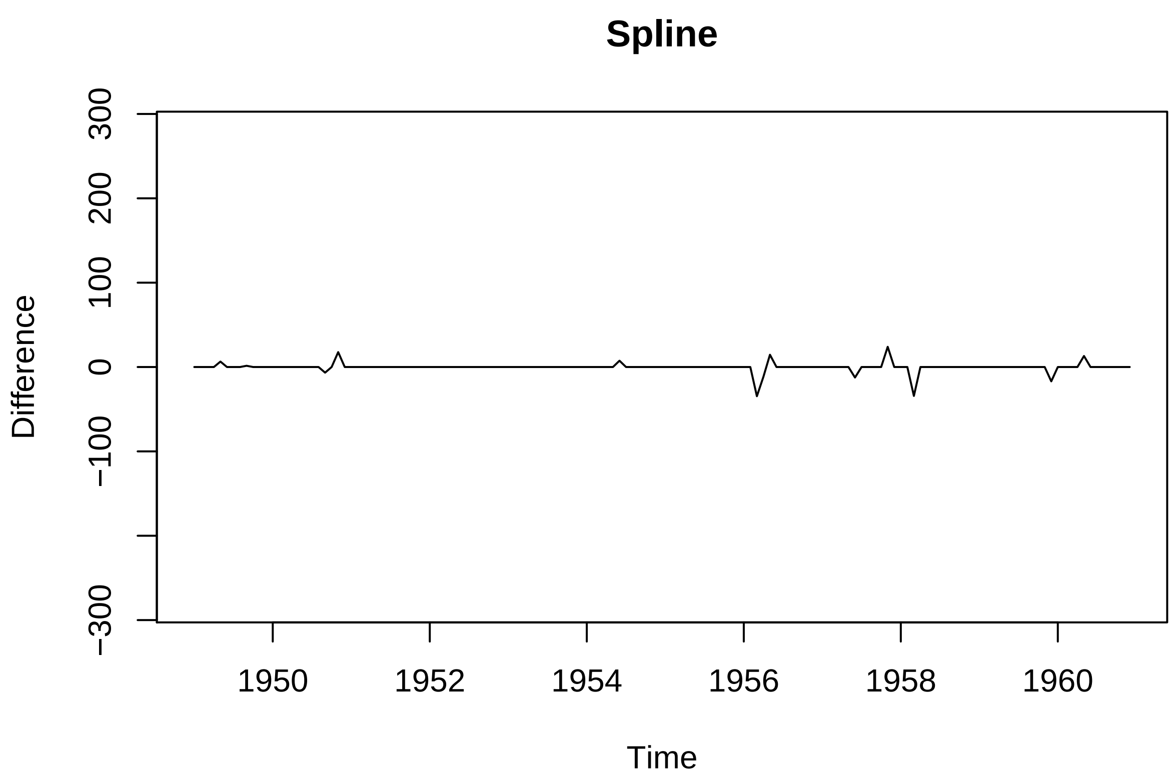
```r
# Linear Interpolation
plot(na.interpolation(tsAirgap, option = "linear") - AirPassengers, ylim = c(-mean(AirPassengers), mean
```




```r
mean((na.interpolation(tsAirgap, option = "linear") - AirPassengers)^2)
```

```
## [1] 37.06684
```

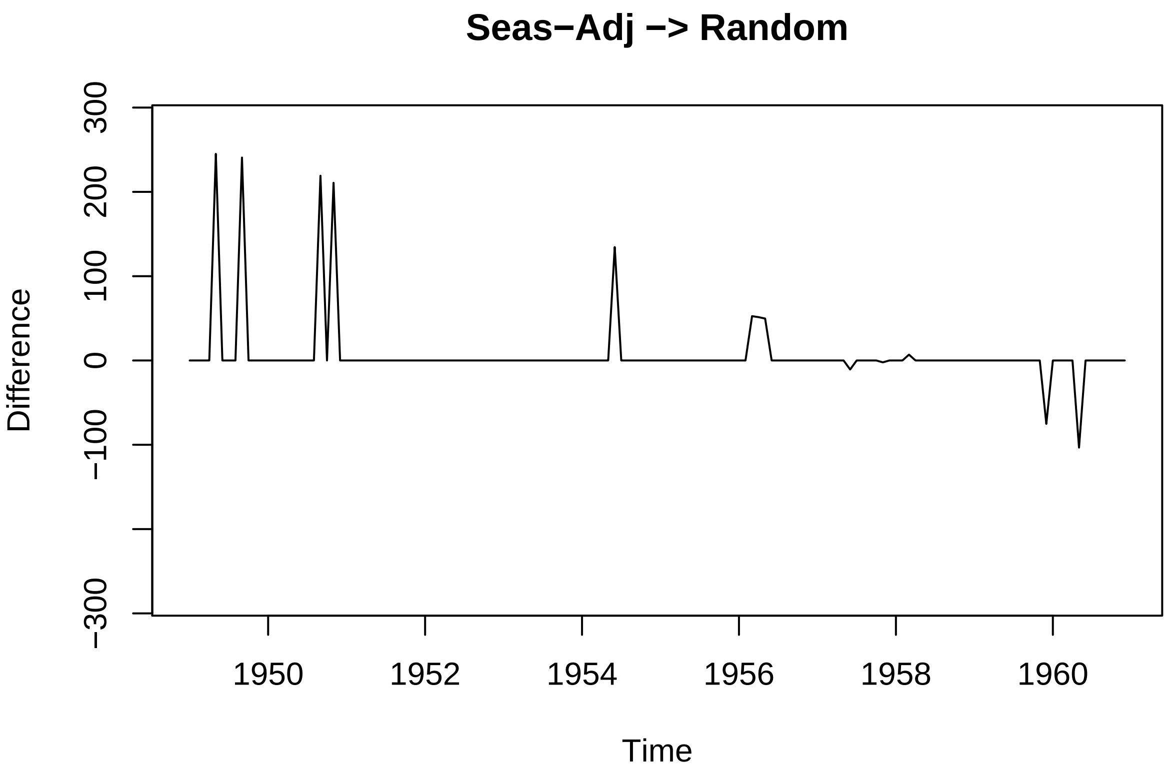
```r
# Spline Interpolation
plot(na.interpolation(tsAirgap, option = "spline") - AirPassengers, ylim = c(-mean(AirPassengers), mean
```



```
mean((na.interpolation(tsAirgap, option = "spline") - AirPassengers)^2)
```

```
## [1] 30.29405
```

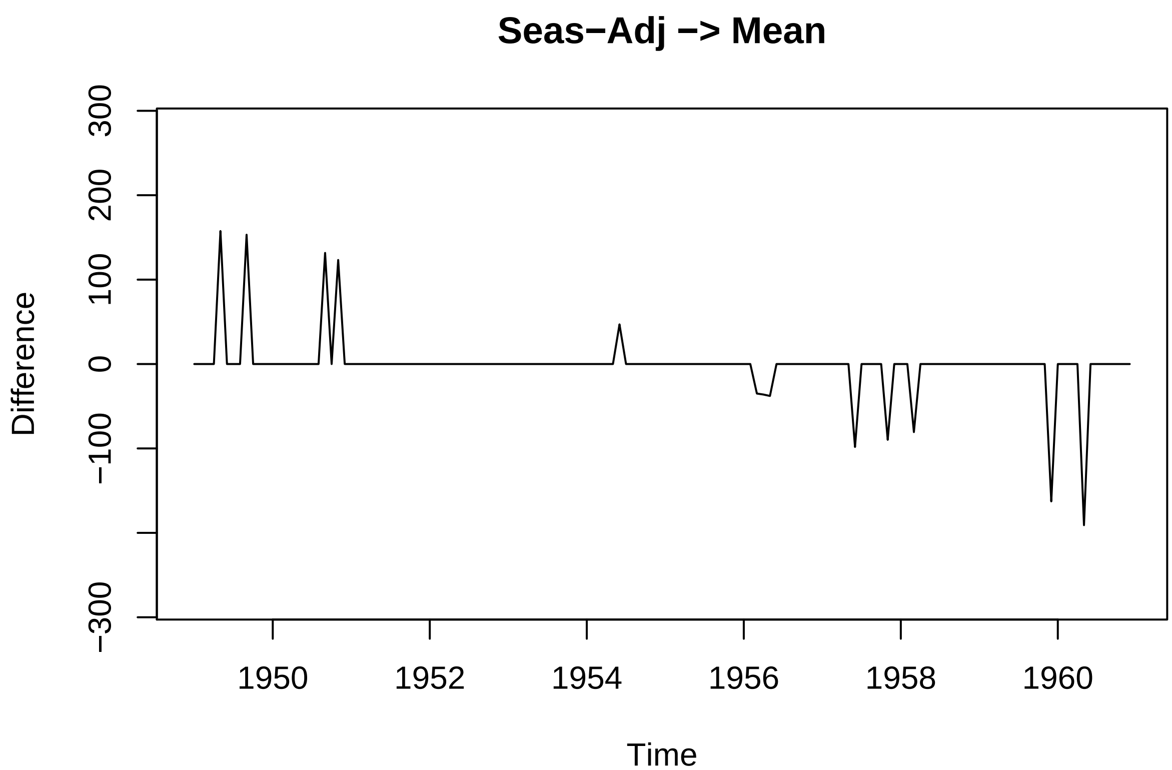
```
# Seasonal Adjustment then Random
plot(na.seadec(tsAirgap, algorithm = "random") - AirPassengers, ylim = c(-mean(AirPassengers), mean(Air
```




```
mean((na.seadec(tsAirgap, algorithm = "random") - AirPassengers)^2)
```

```
## [1] 5968.83
```

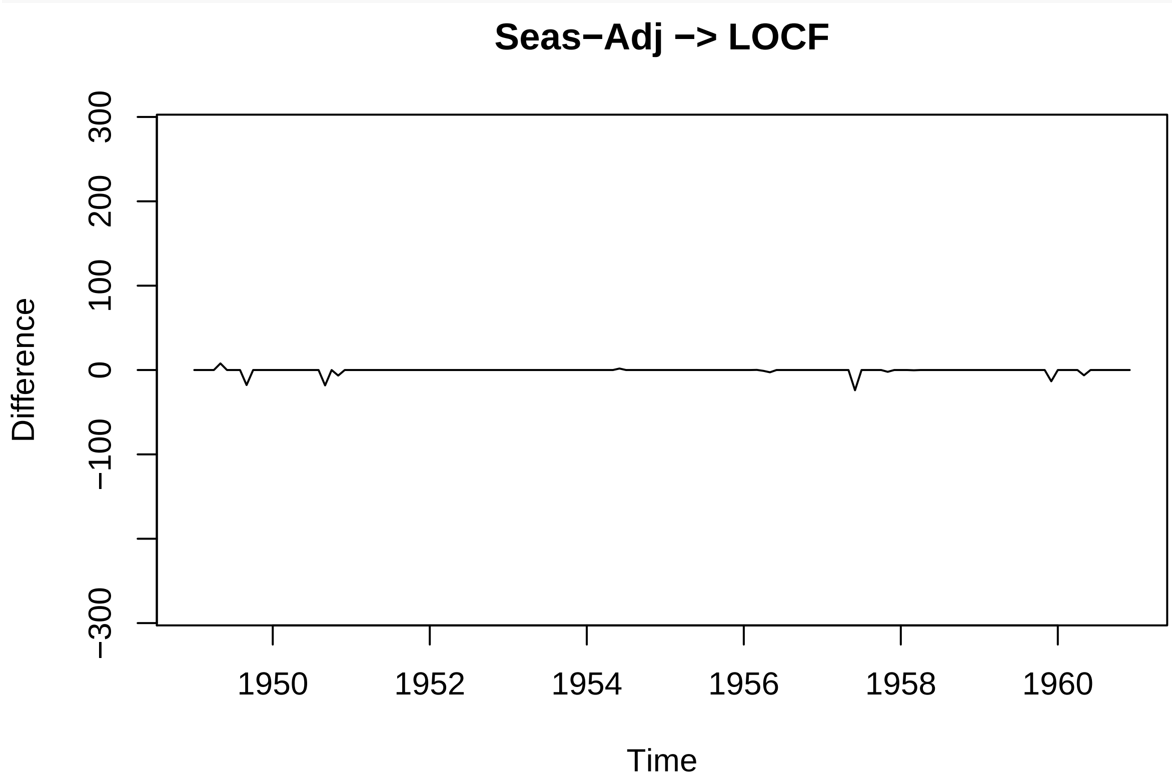
```
# Seasonal Adjustment then Mean
plot(na.seadec(tsAirgap, algorithm = "mean") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPa
```




```
mean((na.seadec(tsAirgap, algorithm = "mean") - AirPassengers)^2)
```

```
## [1] 1209.075
```

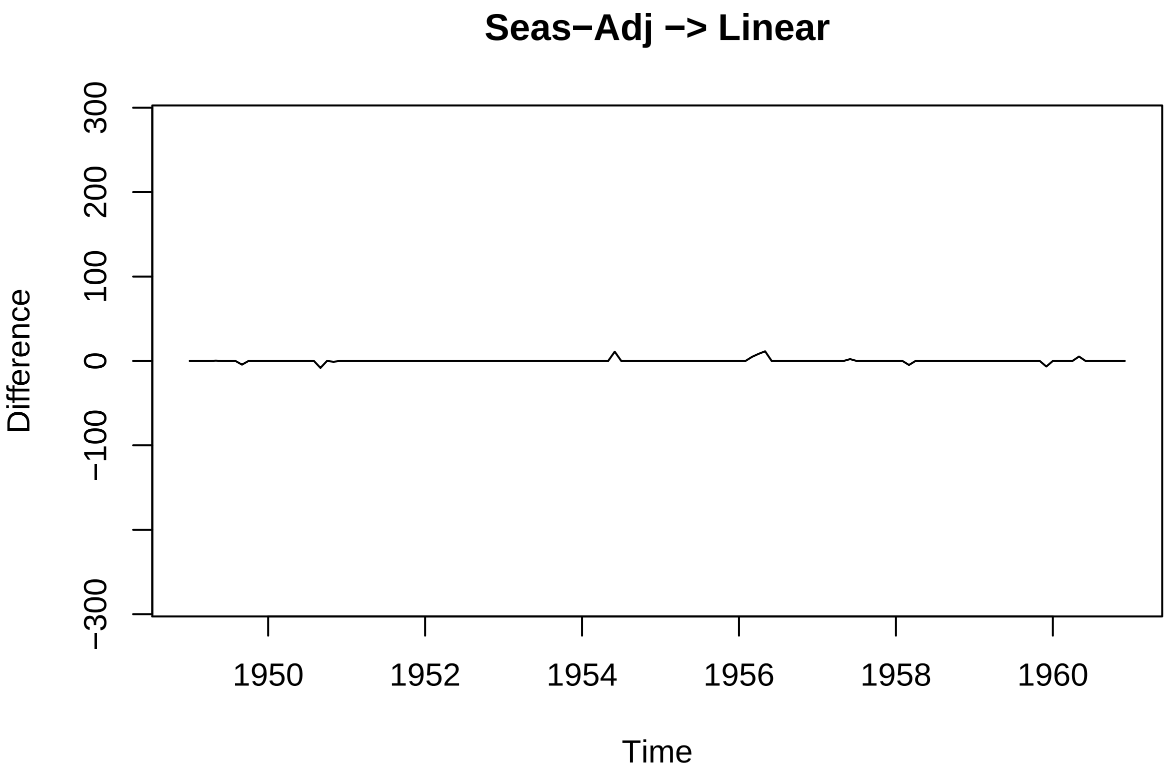
```
# Seasonal Adjustment then LOCF
plot(na.seadec(tsAirgap, algorithm = "locf") - AirPassengers, ylim = c(-mean(AirPassengers), mean(AirPas
```



```r
mean((na.seadec(tsAirgap, algorithm = "locf") - AirPassengers)^2)
```

```
## [1] 10.87818
```

```r
# Seasonal Adjustment then Linear Interpolation
plot(na.seadec(tsAirgap, algorithm = "interpolation") - AirPassengers, ylim = c(-mean(AirPassengers), me
```




```r
mean((na.seadec(tsAirgap, algorithm = "interpolation") - AirPassengers)^2)
```

```
## [1] 3.701541
```

Clustering time Series

In the context of time series, we may use clustering to identify common patterns and shapes and group entire sereis accordingly. This could be useful in identifying a group of time series to use together in a multivariate model.

Partitional clustering of time series is most often performed by the k-medoids algorithm, which behave exactly like k-means exact the centroid at any iteration is one of the time series (prototype) rather than an average of several time series.