PG Diploma in Data Analytics

Course: Predictive Analytics - II

Instructor: Prof. G. Srinivasaraghavan

PGDDA-Lecture Notes/PA-II/3

Time Series Analysis

Contents

List of Figures

List of Tables

1 Learning Objectives

After this module you are expected to be able to carry out analysis of moderate complexity on time series data.

1. Build the Conceptual Foundation for Time Series Analysis, with which you must be able to:
   (a) Understand how Time-Series analysis is different from regression.
   (b) Recognize data that needs to be treated as a time series instance.
   (c) Understand the broad solution approach to time series analysis.
2. Get an overview of common classes of algorithms for time-series analysis.
   (a) Classical Time Series Decomposition & ARMA Models
      i. Additive and Multiplicative Models
      ii. Smoothing Techniques
      iii. Trend Detection — Least Squares
      iv. Seasonality and Spectral Analysis
      v. $ARMA(p,q)$ Modeling
   (b) ARIMA models for non-stationary time series modeling - Differencing; Box-Jenkins Methodology
3. Learn to work with time series data using R. We have used the features that available as part of core R and the `stats` package. However there are several packages like `forecast`, `tstm`, `tseries`, `binhf` that offer similar functions.
4. Evaluation of time series solutions.

2 Appendix: Legend and Conventions, Notation

There are several parts of the test that have been highlighted (in boxes). The document uses three kinds of boxes.

Example 0: Title

One of the running examples in the document. The reader must be able to reproduce the observations in these examples with the corresponding R code provided in Section **??**.

Title

Good to remember stuff. These boxes typically highlight the key take-aways from this document.

Title

Things that are good to be aware of. This is primarily targeted at the reader who wishes to explore further and is curious to know the underlying (sometimes rather profound) connections with other fields, alternative interpretations, proofs of some statements, etc. However one could safely skip these boxes completely if you are only interested in the primary content/message of this document.

Title

`Code snippets. Implementation Tips. Do's and Dont's.`

$\boldsymbol{x}$	A *vector* $\boldsymbol{x}$
x	A scalar quantity x
term	Refer to the glossary as a quick reference for 'term'. These are typically prerequisite concepts that the reader is expected to be familiar with.
`code`	Code snippets, actual names of library functions, etc.

3 Introduction

A Time-Series is a *time-stamped* data set in which each data point corresponds to a set of observations made at a particular time instance. Unlike the previous module where the attempt was to express a response variable in terms of some of the other attributes (explanatory variables), time-series analysis assumes the system to be a black-box — we just attempt to predict what is coming based on the past behaviour patterns. For instance treating stock prices as a time-series would mean creating a model that can predict say the stock prices over the next 1 month if we knew the prices over the past 1 year. We don't really care what factors affect stock prices — macroeconomic policies, oil prices, Brexit, ...— we just want to predict, given we know how it has changed in the past. A time-series view is often taken when we suspect a 'white-box' model based on the underlying factors will be too complex, we do not know the underlying factors that influence the response variable, or may be just that we are not as interested in understanding why something behaves the way it does, as we are in just predicting what will happen next and move on. Examples of time-series arise in diverse domains and are almost ubiquitous. An illustrative list of time-series data across a variety of domains is given below.

- Daily Stock market figures
- Sales / Revenue / Profitability figures of companies by year
- Demographic / Development data (population, birthrate, infant mortality figures, literacy, per-capita income, school enrollment figures) by year
- Blood Pressure and other body vitals by time units (in hours, minutes, ...) appropriate to the severity of the patient
- Ecology / Geology data — progression of the intensity of the tremors of an earthquake with time, pollution figures by year, annual average temperature, thickness of the glacial ice-sheet etc. by year indicating global warming, annual rainfall figures, ...)
- Epidemiology data — spread (number of people affected) of an epidemic with time
- Speech data (signal parameters evolving over time) — these need to be modeled for speech recognition and of course to build applications such as the popular Siri on iPhones
- ...

One obvious common characteristic of all these is that they have a strong temporal dependence — each of these data sets would essentially consist of a series of timestamped observations, each observation tied to a specific time instance. Most time series can be analyzed by examining some typical characteristics exhibited by such data sets:

1. **Trends**: Time varying trends (eg., increasing / decreasing linearly, ...). A typical stock market bull run is an example of an upward trend — the index moving up steadily with time. A trend is typically seen over a (relatively) long stretch of time.

2. **Seasonality**: Exhibiting a regular repeating pattern of some sort. Sales volume in a typical departmental store show regular seasonal patterns — increased volumes during weekends, increased volumes around festivals, etc. Seasonal patterns happen with a periodicity that is known and predictable.

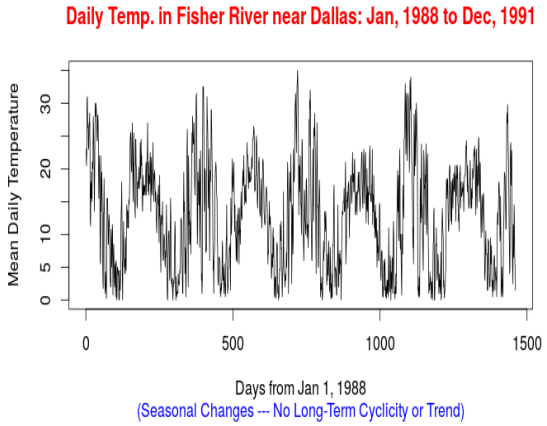
3. **Long-Term Cyclicity**: Long-term cyclic behaviour. Economic inflation-recession cycles are a typical example.

4. **Autocorrelation**: Dependence of the next value on the past. The phenomenon of some video on YouTube for instance going 'viral' is a manifestation of autocorrelation. If the number of views over the last 5 days is say 10, 11, 13, 15, 15 for one video and is 10, 100, 500, 5000, 50000 for another video. That the first one hasn't 'picked up steam' in the last 5 days is a pretty strong indication that it is not likely to be too far from say 15 even on the 10th day. The second video however is the one going 'viral'. The volumes over the last 5 days is a strong indicator of what we can expect on the 6th day. In general degree-k autocorrelation is a where the following value is highly correlated with the history over the last k time instances. We are of course assuming discrete time here — that observations are made once in (or in multiples of) a fixed units of time. In principle in almost any time series, the observations could be made at any time instance and so ideally time must be treated as a continuous variable. The approximation of a time-series by sampling at discrete equally spaced time instances, is simply an acknowledgment that sampled data will, for the most part, be discrete because of restrictions inherent in the method of collection.

5. **Stochasticity**: Almost all real time-series data is noisy due to the inherent unpredictability in the behaviour of the data sources and instruments used to collect the data. Real time-series data is therefore a combination of some expected behaviour (the deterministic, predictable ideal behaviour) and a random stochastic noisy component.

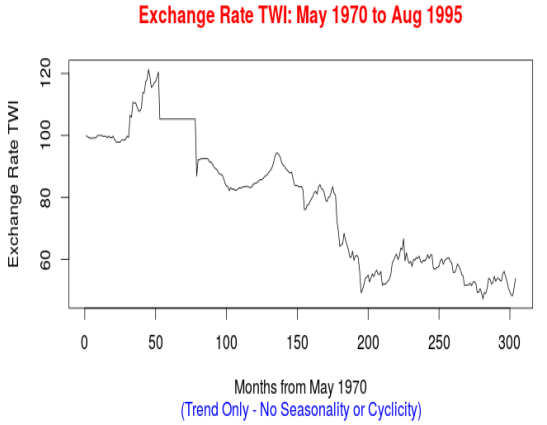
Figure 1 shows plots of time-series data of various kinds, illustrating various combinations of the characteristics described above.

A key difference between the regression methods we explored in the previous module and what we are referring to as time-series here is that in regression we almost always assume the data set consists of iid samples. The independence assumption clearly doesn't hold for time-series data. There is an obvious dependence between data points corresponding to observations at successive time-stamps. In normal regression problems we assume that the explanatory variables in any two data points are completely independent of each other and they have been sampled from the same distribution. In other words any typical regression dataset has not 'canonical' ordering of the data points in the dataset — any random shuffling of the dataset has no effect on the essential nature (the relationship it exhibits between the explanatory variables and the response variable) of the dataset. Time-series data on the other hand has one canonical ordering — ordered ascending on the time stamp — and that's the only order that is meaningful for the dataset. Successive data points in a time-series will naturally have significant correlation that cannot be ignored.

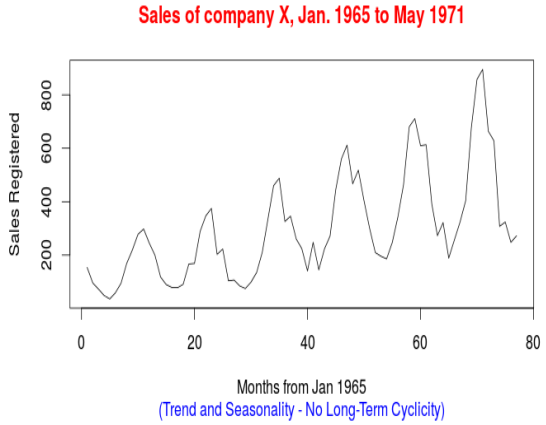


(a) Seasonal Changes in Temperature — No Long-Term Cyclicity or Trend

(b) Large Cyclicity (between 1300 and 1800) and Prominent Trend - No Seasonality



(c) Trend Only - No Seasonality or Long-Term Cyclicity



(d) Trend and Seasonality - No Long-Term Cyclicity

Figure 1: Different Kinds of Time-Series — Datasets taken from `https://datamarket.com/data/list/?q=provider:tsdl`

R Code to Generate Time Series Examples in Figure 1

```
filename <- c('real-daily-wages-in-pounds-england.csv',
              'mean-daily-temperature-fisher-river.csv',
              'exchange-rate-twi-may-1970-aug-1995.csv',
              'sales-data.csv'
             )
ylab <- c('Daily Wages',
          'Mean Daily Temperature',
          'Exchange Rate TWI',
          'Sales Registered'
          )
```

```
xlab <- c('Year',
          'Days from Jan 1, 1988',
          'Months from May 1970',
          'Months from Jan 1965'
          )
title <- c('Real Daily Wages in Pounds in England: 1260--1994',
           'Daily Temp. in Fisher River near Dallas: Jan, 1988 to Dec, 1991',
           'Exchange Rate TWI: May 1970 to Aug 1995',
           'Sales of company X, Jan. 1965 to May 1971'
           )
subtitle <- c('(Large Cyclicity and Prominent Trend - No Seasonality)',
           '(Seasonal Changes --- No Long-Term Cyclicity or Trend)',
           '(Trend Only - No Seasonality or Long-Term Cyclicity)',
           '(Trend and Seasonality - No Long-Term Cyclicity)'
)
xcol <- c(1,2,2,1)
ycol <- c(2,3,3,2)

for (example in seq(1,4)) {
    mydata <- read.csv(filename[example])
    plot(mydata[,xcol[example]], mydata[,ycol[example]],
         xlab=xlab[example], ylab=ylab[example], type='l')
    title(main=title[example], col.main="red",
          sub=subtitle[example], col.sub="blue")
}
```

Note: In the rest of this module, whenever we use any of these 4 examples for illustration, we will reuse the filename, title, column names etc. from the code above in the corresponding R code.

3.1 Generic Approach to Time Series Modeling

We describe a generic approach to time series modeling here, using the concepts such trends and seasonality. Suppose the time-series data is a series of values $(X_0, X_1, \ldots X_t, \ldots X_T)$, X_i denoting the value observed at time $t = i$.

1. The first step in any time-series modeling is to visualize the data, as a line graph and/or a scatter plot. The visualization is very useful to spot trends, seasonality and other patterns in the data, visually. This can be a very useful clue to the kind of model that we could use to model the data. Another important feature to look for in a time series visualization is the presence of outliers — data points that are way outside the pattern observed in the rest. It is often useful to either explain these outliers away and/or eliminate them (as aberrations in the data) before we carry out the modeling exercise. Outliers can sometimes significantly skew the models and consequently the forecasts made from the models.

2. We next try to *stationarize* the data. We will formally define stationarity in a time series later in this document. Informally a stationary time series is one whose statistical properties such as mean, covariances between values at different times, etc. are all constant over time. A stationary time series therefore can be extended (forecasting) by simply assuming the statistical properties observed in the historical part of the series, continue to hold in future as well. In the absence of stationarity even defining many of these statistical properties consistently becomes a challenge. It is also relatively far easier to test for stationarity in a time series than it is to directly test for other patterns. Unfortunately the characteristics such as trends and seasonality are typically are non-stationary. Autoregressive features (under some specific technical conditions) and pure noise are the ones that remain stationary. Therefore stationarizing a time series essentially involves de-trending and de-seasonalizing the data through a series of transformations. We then test for stationarity to confirm if we have correctly modeled the seasonal and other variations in the data. We will examine methods to remove trends and seasonality in the data later in this module. The modified stationary time series obtained after removing the part of the data that characterizes the trend, seasonality and cyclicity in the data is often called the *residual* series, or simply the residue. The residual series is therefore of the form:

$$(Y_0, Y_1, \ldots, Y_T) \quad \text{where} \quad Y_t = S(X_t)$$

3. Test if the residual time series is stationary. If it is not, then iterate on the earlier step to tweak the seasonal, trend and cyclicity models (the transformation S) to arrive at a stationary residual series. Model the residual series as a stationary time series using the methods we will discuss later in this module. Assume the stationary model is M where $M(t) \approx Y_t$ for every time stamp t.

4. Construct the final model by applying the inverse transformation $S^{-1}(.)$ on the stationary model.

5. Check if the final combined model models the data well. To do this we need to compute the residual series $\epsilon_t = X_t - S^{-1}(M(t)), 0 \leq t \leq T$ that remains after we eliminate the value predicted by the final model, and then verify if what remains (the residue) is 'pure' noise. Time series analysts often use the term *white noise* to describe a time series that is pure noise — it has not structured behaviour that can be modeled. This is the stochastic part of the original time series.

We first cover stationarity and the analysis of time-series data that exhibit stationarity. We will later examine some sophisticated methods to eliminate structural patterns in the data such as seasonality, cyclicity and trends. Henceforth for brevity we will denote a time series $(X_0, \ldots, X_T)$ as $\{X_t\}_0^T$ and we will omit the subscript (0) and the superscript (T) when the starting and ending timestamp indices are clear from the context or are irrelevant.

4 Stationary Time Series Analysis and ARMA Models

4.1 Stationarity

Stationarity of a time-series refers to the 'time invariant' relationships it exhibits between values seen at different time stamps. Formally a (strongly) time-invariant series is one where the joint distribution of the set of values $(X_1, \ldots, X_n)$ is identical to the joint distribution of the values $(X_{t+1}, \ldots, X_{t+n})$ for any $t, n > 0$. In other words any given sequence of values is equally likely to appear starting at any time in the overall time series. The other way to interpret the stationarity is: if a series is stationary, then the shape of the time-series plot will look identical irrespective of the time t at which you start collecting / observing the data. The time-series 'shifted' to the right or left makes little difference to the shape of the plot. This is often too a strong a criterion — most real-life time series data is not strongly stationary. Anything other than 'pure noise' (which clearly is not a very interesting time-series) will almost invariably be non-stationary, in the strong sense that we have defined here.

We often work with a weaker notion of stationarity — called *weak stationarity*. We would call a time-series weakly stationary if it is stationary in a limited sense — that the relationship between any pair of values (some time-steps, including zero, away) in the time series is independent of the timestamp of the first event. For technical reasons we often assume that the expected value (mean) of the square of the value at any time t is bounded. This would be a common assumption throughout the rest of the document, unless stated otherwise.

(Weakly) Stationary Time Series

For a time series $\{X_t\}$, let $\mu_X(t) \equiv E[X_t]$ denote the expected (mean) value the series can take at time t. **Covariance Function** for the time series is defined as

$$\sigma_X(i,j) = E\left[\left(X_i - \mu_X(i)\right)\left(X_j - \mu_X(j)\right)\right]$$

The covariance function measures how correlated the values at two different timestamps in a time series are. The time series is said to be (weakly) stationary if $E[X_t]$ and the covariance function $\sigma(t+h,t)$ for any fixed *lag* h is independent of the time t. Note that h can be both negative and positive. The **Correlation Function** is the For covariance function normalized with the individual variances.

$$\rho_X(i,j) = \frac{\sigma_X(i,j)}{\sqrt{Var(X_i).Var(X_j)}}$$

a stationary time series, because of the time independence, we can define a **Autocovariance** function $\eta(h)$ that measures the covariance between values occurring with a time *lag* of h. Formally $\eta(h) \equiv \sigma(t+h,t)$. The **Autocorrelation** function (ACF) $\rho(h)$ is a normalized autocovariance measure defined as

$$\rho(h) = \eta(h)/\eta(0)$$

As we can see autocorrelation is autocovariance normalized with the variance of an individual value in the time series. Note that $\sigma(t+h,t)$ is independent of t for all h also implies it is independent of t for $h=0$. Clearly $\sigma(t,t)$ is just the variance of X_t. So $\sigma(t,t)$ being independent of t implies the variance for all the values in the time series is the same.

Often we also use a Partial Autocorrelation Function (PACF). Notice that ACF looks at the correlation between values at time stamps that are h unit steps away — say X_t and X_{t+h}. However this correlation, measured directly as in the above formula, is a composite of the 'partial' correlations between X_{t+h} and other X_is for all $t \le i \le (t+h-1)$. It might be useful to try 'isolating' the direct correlation between just X_{t+h} and X_t without the influence of any of the intermediate values. PACF is just this. PACF is formally defined using conditioning on the intermediate values, as:

$$\pi_X(h) = \frac{Cov\left(X_{t+h}, X_t \mid X_{t+1}, \ldots, X_{t+h-1}\right)}{\sqrt{Var\left(X_{t+h} \mid X_{t+1}, \ldots, X_{t+h-1}\right).Var\left(X_t \mid X_{t+1}, \ldots, X_{t+h-1}\right)}}$$

Autocorrelation and Autocovariance are important measures of stationarity of a time series. In practise though what we really have is just time series data and we have no idea about the mean values for each X_t. How do we then check if a given time series is stationary (or close to it)? We can define some sample measures that can be computed from the given data to approximate the covariance and correlation functions of the underlying (hypothetical) process that is generating the time series.

Sample Covariance and Correlation of a Stationary Series $\{X_t\}_0^T$

Sample Mean — this approximates the mean of X_t at any t

$$\overline{X} = \frac{1}{T+1}\sum_{t=0}^{T} X_t$$

Sample Autocovariance Function — Expectation is replaced by an average over the length of the series; mean at t replaced by $\overline{X}$

$$\overline{\eta}(h) = \frac{1}{T+1}\sum_{t=0}^{T-h}(X_{t+h}-\overline{X})(X_t-\overline{X})$$

Sample ACF

$$\overline{\rho}(h) = \frac{\overline{\eta}(h)}{\overline{\eta}(0)}$$

PACF is hard to estimate directly and there are no known closed form formulas for computing PACF from a given time series dataset. One can show an equivalence between PACF values and the coefficients of an appropriate AR model (will be discussed later in this document). This is equivalence is used to iteratively estimate the PACF values (for example the *Levinson-Durbin Recursion Algorithm*). Thankfully there are implementations readily available in many of the standard statistical libraries. We will use these implementations going forward.

4.2 White Noise

Consider a time series that consists of several values all of which are sampled independently from a Gaussian distribution with unit variance. This is clearly a stationary time series with

$$\rho(h) = \eta(h) = \begin{cases} 1 & \text{if } h = 0 \\ 0 & \text{Otherwise} \end{cases}$$

Figure 2 below shows two instances of such a time series. A time series consisting of uncorrelated values each with zero mean and the same variance is also called *White Noise*. We denote white noise with variance σ^2 as $\mathcal{W}(0, \sigma^2)$. **Notice that for white noise the ACF value must be zero for all** $h > 0$. This is a crucial test for any time series analysis. Recall the last step after the modeling is done, we need to finally check if the residues after removing the predicted values out of the original series has the characteristics of white noise, to confirm if our modeling exercise is successful. We would often check for something stronger — that the white noise is in fact Gaussian. 'Pure' noise is general considered to be Gaussian — each value in the white noise is considered to be an independent sample from the same (identical) Gaussian $\mathcal{N}(0, \sigma^2)$.

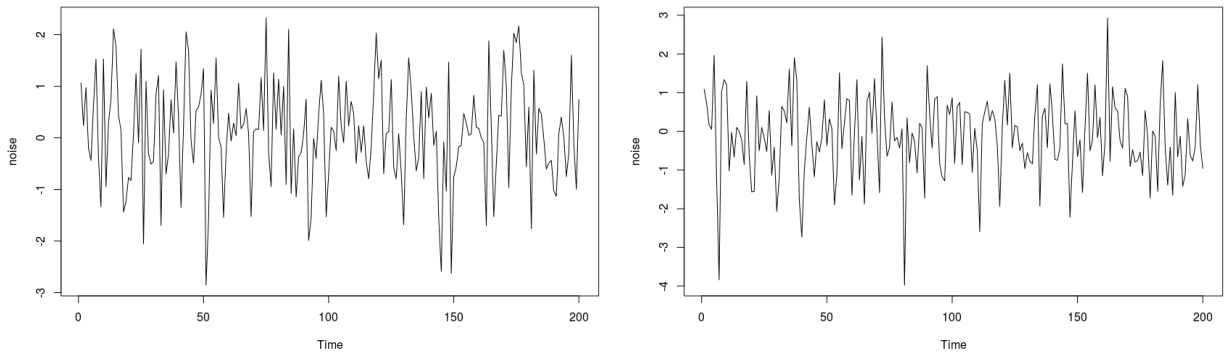


Figure 2: Two instances of Gaussian White Noise Series $\{X_t\}_0^{200}$ where $X_t = \mathcal{N}(0, 1)$

Hypothesis Testing for White Noise

Final test for the effectiveness of the modeling exercise is to check if the residue with respect to the model is just (Gaussian) white noise. We pose this as a Hypothesis test with the null hypothesis being "the residual time series consists of iid (uncorrelated) samples from a Gaussian"; the alternate hypothesis being "the time series has correlations between values observed at different times, that cannot be ignored". We often compute a statistic for this from the values in the time series and a p-value that gives the probability *under the null hypothesis (white noise), of encountering a statistic at least as extreme as that which was observed.* We use a predefined significance level (often fixed at 0.05 or 5%) α. Formally α represents the threshold for the probability that we will make an error if we rejected the null hypothesis. The null hypothesis is rejected (we decide that the residual series is not white noise) if and only if $p \leq \alpha$. Informally what we are saying is that if $p \leq \alpha$, since a small p indicates the current time series is a very unlikely instance of a white noise series, by

concluding that the series is not noise we are only making an error in the very unlikely case when a white noise process actually generated the current time series. Some commonly used tests for this are:

1. Sample Autocorrelations
2. Portmanteau Test or Ljung & Box Test
3. Using Yule-Walker Algorithm
4. Turning Point Test
5. Difference Sign Test
6. Runs Test
7. Rank Test
8. Explicit Tests for Normality

As a final test for the fitness of the model, it is recommended that we carry out at least two-three of these tests on the residual series. The first three among these have ready-made implementations in R. The last four can be carried out easily using simple R functions, the code for which is given at the end of this document in Section 8. These tests are described briefly below. Most of these tests are also available as part of the `randtests` package in R.

4.2.1 Sample Autocorrelations

Figure 3 shows a plot of the Sample ACF for the Gaussian Noise Data that we used for figure 2. It shows a plot of the the Sample ACF for each value of the lag parameter h (taken as the x / horizontal axis). The ACF plot is a very important tool in any time series analyst's toolbox. The two blue horizontal lines in the plot are lines that indicate the 95% confidence interval range (the confidence level can be specified in the `acf()` call in R — it uses a default confidence value of 95%). We can see all the ACF values (across all lag values) with just one exception lie within the range. We have already seen that the ACF for a stationary time series will be non-zero only when the lag is 0 — so the sample ACF must be close to zero for all lag values other than 0. That's exactly what we observe in the ACF plot in Figure 3. Since the blue lines show a 95% confidence interval, we expect the ACF values of at most 5% of the lag values plotted to fall outside the band. Note that we have just one barely touching the band in the plot which well below 5% of the total number of lag values plotted (40 in this case) which is 2. We reject the null hypothesis if we find more than 5% of ACFs for the lag values plotted breaching the confidence interval — in this case we conclude that there are correlations that need to be investigated further.

R Code to Generate Figures 2 and 3

```
# Generate a sequence of Gaussian random numbers and
# convert the sequence into a time-series object
```

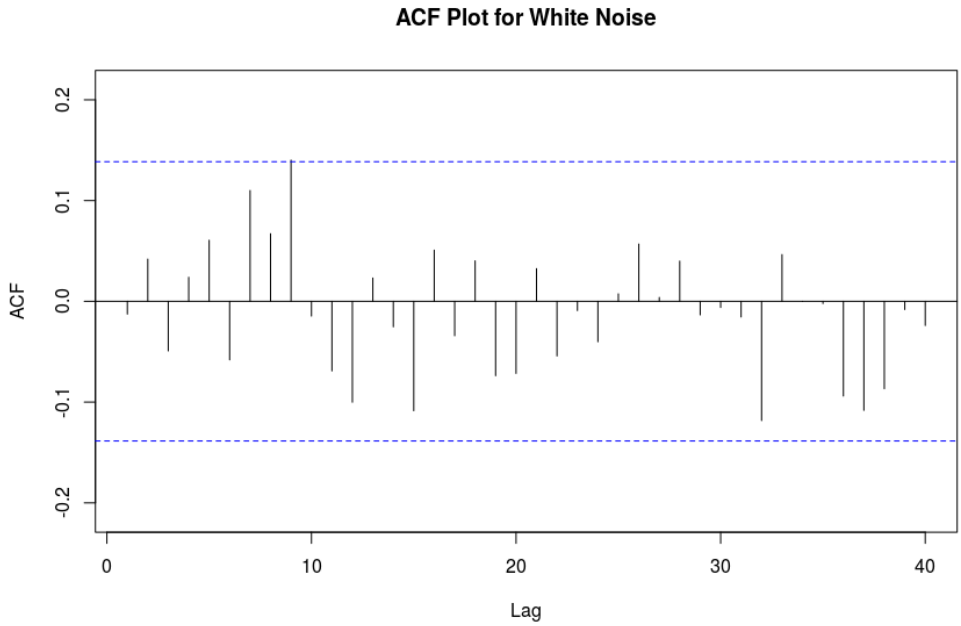


Figure 3: Sample ACF for Gaussian Noise

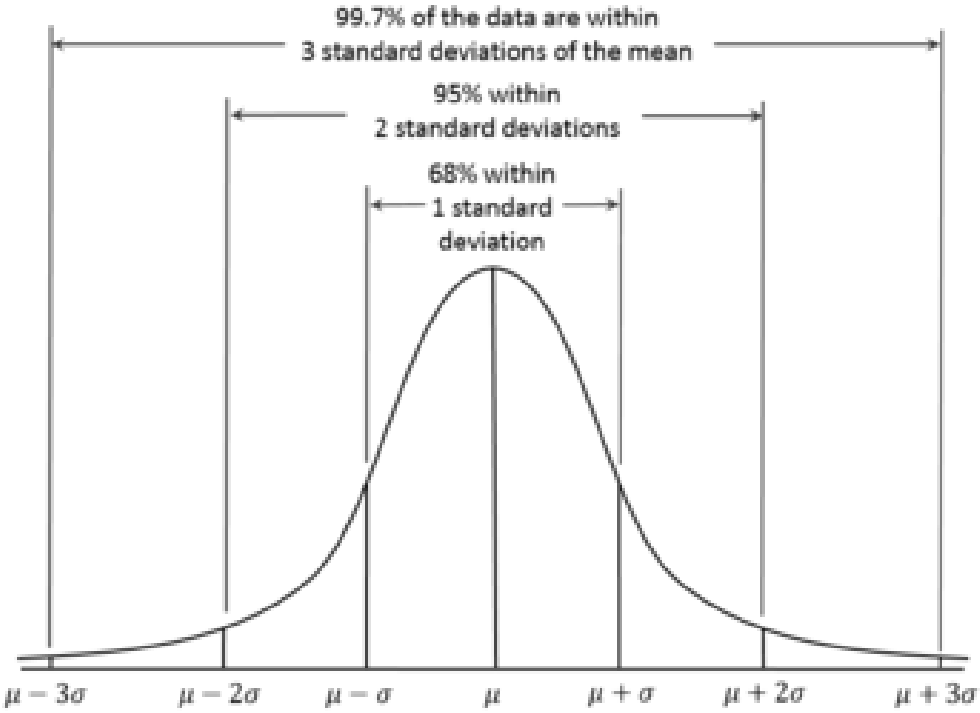


Figure 4: Gaussian Quantiles — Illustration taken from Wikipedia `https://en.wikipedia.org/wiki/Normal_distribution#Quantile_function`

```
noise <- ts(rnorm(200, mean = 0, sd = 1))
# Plot the time series
plot.ts(noise)
# Compute and plot the ACF for the time series
acf(noise, level=95, lag.max=40, main='ACF Plot for White Noise')
```

Confidence Intervals for the Gaussian and ACF

Sample Autocorrelations $\overline{\rho}(h)$ for i.i.d noise with finite variance can be shown to be approximately Gaussian with zero mean and variance $1/n$, where n is the length of the time series. For a Gaussian distribution it can be shown that with 95% probability the value of a sample from this distribution will lie within 1.96 standard deviations away from the mean of the Gaussian. In other words 95% of the area under the Gaussian distribution curve lies in the interval $\mu \pm 1.96\sigma$ where μ is the mean of the Gaussian and σ^2 is the variance. See Figure 4 for an illustration. So in the ACF plot we should expect ACF values for 95% of the lag values to lie within $\pm\left(1.96/\sqrt{n}\right)$ of the mean. In our plot the mean is zero and $n = 200$. Therefore we expect 95% of the plotted ACF values to lie in the interval $[-0.1386, +0.1386]$. The band between the blue horizontal lines in Figure 3 represents exactly this interval. If more than 5% of the ACF values plotted breach the confidence interval then we reject the null hypothesis.

4.2.2 Ljung-Box (Portmanteau) Test

In this case an aggregate statistic of the entire ACF plot is used as the test statistic. The aggregate statistic used in this case is the Q-statistic, defined as

$$Q = n(n+2)\sum_{i=1}^{h}\frac{\overline{\rho}_i^2}{n-i}$$

where n is the length of the time series and h the number of lag values being tested for. If the null hypothesis is true then Q will be a sample from a χ^2 distribution with h degrees of freedom. For a significance level α the p-value for the rejection of the null hypothesis is defined as $p(Q > \chi^2_{1-\alpha,h})$, where $\chi^2_{1-\alpha,h}$ is is the $(1-\alpha)$ quantile value of a χ^2 distribution with h degrees of freedom. A χ^2 distribution with h degrees of freedom is essentially the distribution of the sum of the squares of h random variables, each of which is a Gaussian.

Ljung-Box Test

```
> noise <- ts(rnorm(200, mean = 0, sd = 1))
> Box.test(noise, lag = 40, type = "Ljung")

Box-Ljung test

data:  noise
```

```
X-squared = 35.365, df = 40, p-value = 0.6788
```

The box above shows the result of the Ljung-Box test on a iid noise series. Notice that the p-value is well above the signifance level $\alpha = 0.05$. So we have no reason to reject the null hypothesis.

4.2.3 Turning Point Test

If the series is iid then we do expect that it will keep going up and down and it is very unlikely that there will be too many long monotonic stretches where the value keeps going up or keeps going down. A *Turning Point* is one where the direction changes. Formally for a series of values $X_1, \ldots, X_n$, X_t is a Turning Point if $X_t > X_{t-1}$ and $X_t > X_{t+1}$, or $X_t < X_{t-1}$ and $X_t < X_{t+1}$. It can be shown that for an iid series, for a large n, the number of Turning Points T roughly follows a Gaussian distribution with mean and variance given as

$$\mu_T = 2(n-2)/3, \quad \sigma_T^2 = (16n-29)/90$$

For the commonly used significance level $\alpha = 0.05$, we would therefore reject the null hypothesis of the series being iid if $|T - \mu_T| > 1.96\sigma_T$. See the code in Section 8.1.

4.2.4 Difference Sign Test

The Difference Sign count D of a series is defined as the number of time stamps t such that $X_t > X_{t-1}$. We can again show that if the time series is iid then for large n, D would follow a Gaussian distribution with mean and variance given as:

$$\mu_D = \frac{1}{2}(n-1), \quad \sigma_D^2 = (n+1)/12$$

For the commonly used significance level $\alpha = 0.05$, we would therefore reject the null hypothesis of the series being iid if $|D - \mu_D| > 1.96\sigma_D$. See the code in Section 8.2.

4.2.5 Runs Test

This is similar to the Turning Point Test. The test looks for the number of stretches r of two or more consecutive time stamps with the value continuously increasing or the value continuously decreasing in the stretch. Let the number of times the series increased be n_1 and let the number of times the value decreased be n_2. Clearly $n_1 + n_2 = n$. We can again show that if the time series is iid then for large n, r would follow a Gaussian distribution with mean and variance given as:

$$\mu_r = \frac{2n_1n_2}{n} + 1, \quad \sigma_r^2 = (\mu_r - 1)(\mu_r - 2)/(n-1)$$

For the commonly used significance level $\alpha = 0.05$, we would therefore reject the null hypothesis of the series being iid if $|r - \mu_r| > 1.96\sigma_r$. See the code in Section 8.3.

4.2.6 Rank Test

This is particularly useful for detecting linear trends in the time series data. The rank R of a series is defined as the number of pairs X_i, X_j such that $X_j > X_i$ and $j > i$. We can again show that if the time series is iid then for large n, R would follow a Gaussian distribution with mean and variance given as:

$$\mu_R = \frac{1}{4}n(n-1), \quad \sigma_R^2 = n(n-1)(2n+5)/72$$

For the commonly used significance level $\alpha = 0.05$, we would therefore reject the null hypothesis of the series being iid if $|R - \mu_R| > 1.96\sigma_R$. See the code in Section 8.4.

4.2.7 Explicit Tests for Normality

We can also test the given time series explicitly for Gaussian-ness (normality) — do the values in the time series fit into a Gaussian distribution. Note that this test is slightly different from the other tests listed above. The other tests are tests for the iid-nature of the time series. The above tests would pass even if the values at each t are not Gaussian, but independent across time stamps. The primary criterion in all the above tests is based on an aggregate computed from the entire time series and they rely on the fact that these aggregates are almost always (for large n) follow a Gaussian distribution, irrespective of the distribution of values at a given time stamp. Recall the Central Limit Theorem in basic statistics. However this is often a useful tests since 'pure' is often considered Gaussian.

One way to test for Gaussian-ness is to plot a histogram of the values and compare the histogram against a Gaussian distribution with the same mean and variance. Another way is to construct a $Q-Q$-plot. A $Q-Q$-plot is a way to compare two distributions by plotting the quantiles of the two distributions against each other. In our case we plot the quantiles of the given time series with an ideal Gaussian. If the two distributions are identical then the plot should be almost a 45% straight line starting from the origin. The R code for generating both the plots shown in Figure 5 are given below.

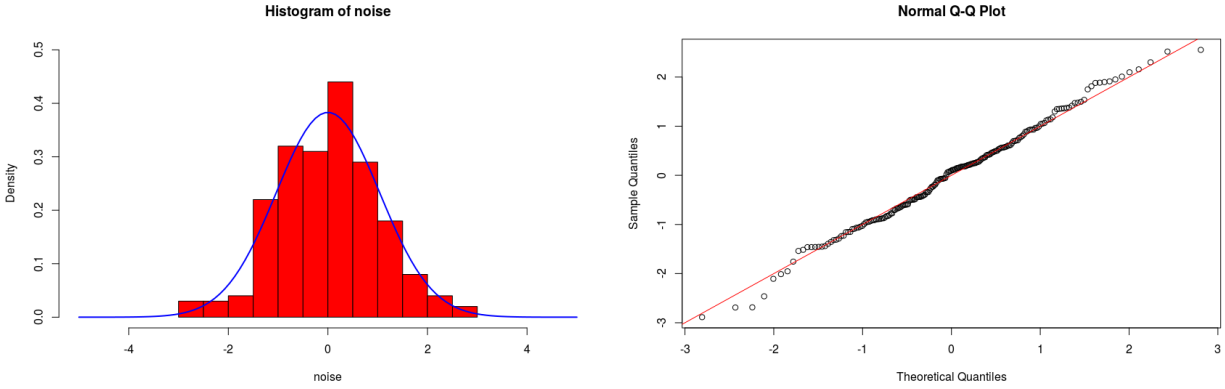


Figure 5: Histogram of the White Noise Series (Left) and the Gaussian $Q-Q$ Plot

R Code to Generate Plots in Figure 5

```
noise <- ts(rnorm(n, mean = 0, sd = 1))

# Code for the histogram
hist(noise, freq=FALSE, prob=T,ylim=c(0,0.5),xlim=c(-5,5),col="red")
mu <- mean(noise)
sigma <- sd(noise)
x<-seq(-5,5,length=100)
y<-dnorm(x,mu,sigma)
lines(x,y,lwd=2,col="blue")

# Code for the QQ Plot
qqnorm(noise)
abline(0,1, col='red')
```

A couple of other tests for which ready implementations are available can also be used to test for Gaussian-ness. These are Kolmogorov-Smirnoff test which again test a sample against a given distribution, and the Shapiro test that directly tests for Gaussian-ness. The R code using both and the output is given in the box below.

R Code for Kolmogorov-Smirnoff Test and Shapiro Test

```
noise <- ts(rnorm(n, mean = 0, sd = 1))

# Code for the KS Test
> mu <- mean(noise)
> sigma <- sd(noise)
> ks.test(noise,"pnorm",mu,sigma)

One-sample Kolmogorov-Smirnov test

data:  noise
D = 0.043187, p-value = 0.8497
alternative hypothesis: two-sided

# Code for the Shapiro Test
> shapiro.test(noise)

Shapiro-Wilk normality test

data:  noise
W = 0.99385, p-value = 0.5779
```

Note that the p-value in both cases is well above the significance threshold of 0.05, giving us no reason to reject the null hypothesis of Gaussian-ness.

We now look at two other important classes of time series that often exhibit stationary behaviour. These form the basis for the ARMA models that we are about to describe.

4.3 Moving Average (MA) Time Series

Moving Average Time Series is one where it is basically noise but the influence of the noise at some time step t carries over also to the value at $t+1$, or possibly upto $t+h$ for some fixed h. Formally a Moving Average time series $\{X_t\}$ of *Order* h (denoted $MA(h)$) is

$$X_t = \mu + Y_t + \sum_{i=1}^{h} \alpha_i Y_{t-i}$$

where $\{Y_t\} = \mathcal{W}(0, \sigma^2)$, for some constants $\mu, \alpha_i, 1 \leq i \leq h$. Note that this gives a process that is centered around the constant value μ. The value at time t in a $MA(h)$ process is therefore the noise at the current time t superimposed on the cumulative weighted influence of the noise at h previous time stamps. We can show that a $MA(h)$ process is a stationary for any fixed h. As for the name, from Box & Jenkins themselves, "The name 'moving average' is somewhat misleading ... However, this nomenclature is in common use, and therefore we employ it."

Stationarity of $MA(h)$ Processes

To prove the stationarity of any $MA(h)$ process we need to show that the covariance function for any such process is independent of the time t. This is easily seen from the following derivation, using the simple observation as a consequence of the white noise assumption that $E[Y_t] = \mu$ and $E[Y_t^2] = \sigma^2$ for any t:

$$\begin{aligned}
\forall t: \quad E[X_t] &= \mu + E[Y_t] + \sum_{i=1}^{h} \alpha_i E[Y_{t-i}] = \mu \\
\forall t, i \neq j: E[Y_i Y_j] &= E[Y_i].E[Y_j] = 0 \qquad (Y_i \text{ and } Y_j \text{ are uncorrelated}) \\
\sigma(t+k, t) &= E\left[(X_{t+k} - E[X_{t+k}])\,(X_t - E[X_t])\right] \\
&= E[X_{t+k} X_t] = \mu^2 + E\left[\left(Y_{t+k} + \sum_{i=1}^{h} \alpha_i Y_{t+k-i}\right)\left(Y_t + \sum_{i=1}^{h} \alpha_i Y_{t-i}\right)\right] \\
&= \mu^2 + E\left[Y_{t+k} Y_t + \sum_{i=1}^{h} \alpha_i Y_t Y_{t+k-i} + \sum_{i=1}^{h} \alpha_i Y_{t+k} Y_{t-i} + \sum_{i=1}^{h} \sum_{j=1}^{h} \alpha_i \alpha_j Y_{t-i} Y_{t+k-j}\right] \\
&= \mu^2 + (E[Y_{t+k} Y_t]) + \left(\sum_{i=1}^{h} \alpha_i E[Y_t Y_{t+k-i}]\right) + \left(\sum_{i=1}^{h} \alpha_i E[Y_{t+k} Y_{t-i}]\right) \\
&\quad + \left(\sum_{i=1}^{h} \sum_{j=1}^{h} \alpha_i \alpha_j E\left[Y_{t-i} Y_{t+k-j}\right]\right) \\
&= \mu^2 + E_1 + E_2 + E_3 + E_4
\end{aligned}$$

where E_1, E_2, E_3, E_4 refer to the 4 expectation summation terms in the previous expression (shown in brackets) in that order. We can now check easily the following:

- If $k = 0$, $E_1 = \sigma^2$, $E_2 = E_3 = 0$, and $E_4 = \sigma^2 \left(\sum_{i=1}^{h} \alpha_i^2\right)$.
- If $1 \leq k < h$, then $E_1 = E_3 = 0$, $E_2 = \alpha_k \sigma^2$ and $E_4 = \sigma^2 \sum_{i=1}^{h-k} \alpha_i \alpha_{i+k}$.
- If $k = h$ then $E_1 = E_3 = E_4 = 0$ and $E_2 = \alpha_h \sigma^2$.
- If $k > h$ then $E_1 = E_2 = E_3 = E_4 = 0$.

Therefore we get

$$\eta(k) = \sigma(t+k, t) = \begin{cases} \mu^2 + \sigma^2 \left(1 + \sum_{i=1}^{h} \alpha_i^2\right) & \text{if } k = 0 \\ \mu^2 + \sigma^2 \left(\alpha_k + \sum_{i=1}^{h-k} \alpha_i \alpha_{i+k}\right) & \text{if } 1 \leq k < h \\ \mu^2 + \sigma^2 \alpha_h & \text{if } k = h \\ \mu^2 & \text{if } k > h \end{cases}$$

We can see that the covariance function is independent of t thus establishing the claim that $MA(h)$ is stationary. We can also explicitly write out the formulas for the autocorrelation function (to simplify, for the case $\mu = 0$):

$$\rho(k) = \frac{\eta(k)}{\eta(0)} = \begin{cases} 1 & \text{if } k = 0 \\ \left(\alpha_k + \sum_{i=1}^{h-k} \alpha_i \alpha_{i+k}\right) \div \left(1 + \sum_{i=1}^{h} \alpha_i^2\right) & \text{if } 1 \leq k < h \\ \alpha_h \div \left(1 + \sum_{i=1}^{h} \alpha_i^2\right) & \text{if } k = h \\ 0 & \text{if } k > h \end{cases}$$

Figure 6 shows the raw values and its ACF plot for a synthetically generated $MA(5)$ time series. The key observation here is that the ACF plot does not fit into the 95% confidence band any more — there are too many exceptions than what one can accept as part of a hypothesis that claims the series to be white noise. However the intial part of the ACF plot shows a declining pattern where the ACF value roughly halves with every unit increase in the lag. Notice from the code used to generate this series, we have used a α_1 value of 0.5. Also the ACF value remains within the significance band (close to zero) for all $h \geq 5$. This is an indication that we need a $MA(h)$ for some $h \approx 5$. The ACF plot therefore does give several clues about the nature of the model that we could use for capturing the time series. We will explore how we can extend this intuition more systematically later in this module.

R Code for generating Figure 6

```
# Simulate a MA(5) Process
ma5 <- arima.sim(model=list(ma=c(0.5, 0.3, 0.2, 0.1, 0.01)),n=n)
plot.ts(ma5)
acf(ma5,type="correlation", level=95, lag.max=40, plot=T,
    main='ACF Plot for MA(5)')
```

4.4 Auto Regressive (AR) Time Series

Auto-regressive time series is one where the value at time t depends on the values at times $(t-1), \ldots, (t-h)$ superimposed on a white noise term. We define a auto-regressive time series

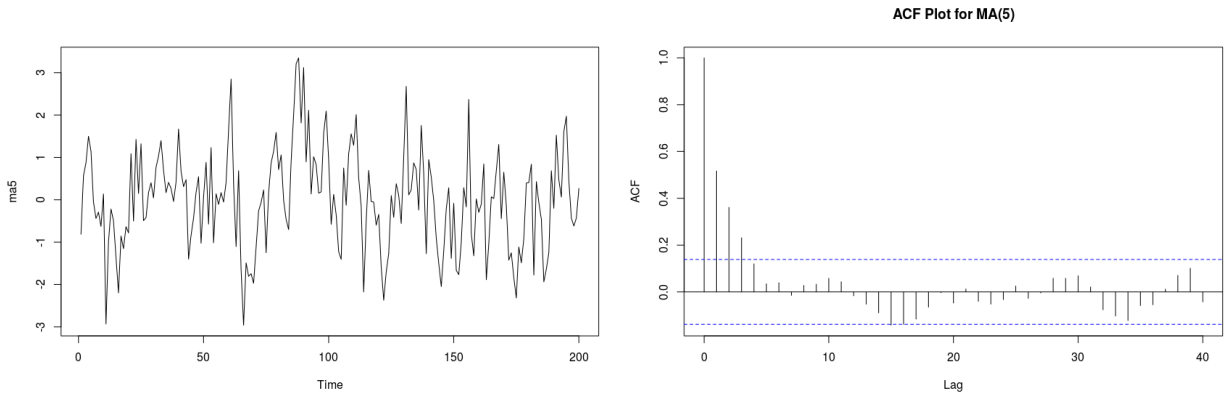


Figure 6: MA(5) Process — the time series and its ACF plot

$AR(h)$ of *order* h as a series $\{X_t\}_0^T$ where

$$X_t = \mu + \sum_{i=1}^{h} \alpha_i X_{t-i} + Z_t, \quad Z_t = \mathcal{W}(0, \sigma^2)$$

for some constants μ and $\alpha_i, 1 \leq i \leq h$. The coefficient α_i represents the influence (weight) of the value of the time series i-steps in the past, on the current value. Unlike the $MA(h)$ models, an $AR(h)$ process is not guaranteed to be stationary. However it is possible to show that under certain conditions (these conditions are too technical to describe here; for example when $h = 1$ and $|\alpha_1| < 1$) such models do admit a unique stationary solution. Also for $AR(h)$ processes that are *assumed* to be stationary, there exist algorithms to estimate the coefficients α_i from the time series data.

Assuming a given $AR(1)$ time series is stationary, we can show that its autocorrelation function $\rho(h) = \phi^{|h|}$ where the time series is of the form

$$X_t = \mu + \phi.X_{t-1} + Z_t, \quad Z_t = \mathcal{W}(0, \sigma^2), \quad |\phi| < 1, \quad \text{and } Z_t \text{ is uncorrelated with } X_k, \text{ for } k < t$$

If the ACF values for small values of h tapers of geometrically with a ratio $\approx \phi$ and remains zero after $h > k$ then it is a strong indication that the underlying process is a $AR(k)$ process. However for a $AR(p)$ stationary series, PACF is often a stronger indicator than the ACF plot.

Autocorrelations of a Stationary $AR(1)$

From the definition of covariance $Cov(X, Y) = E\left[(X - \mu_X)(Y - \mu_Y)\right]$ it is easy to verify that for for any random variables X, Y, X and constants a, b, c

$$Cov(aX + bY + c, Z) = a.Cov(X, z) + b.Cov(Y, Z)$$

This is a formal statemnet of the fact that $Cov(.)$ is a linear function. We will use this in the

proof of our claim below.

$$X_{t+1} = \mu + \phi.X_t + Z_t \Rightarrow E[X_t] = \mu + \phi.E[X_{t-1}] + E[Z_t]$$

$$E[X_t] = \frac{\mu}{1-\phi} \quad \text{Note that because of stationarity, } E[X_t] = E[X_{t-1}]$$

$$Cov(X_t, X_{t-h}) = Cov(\phi.X_{t-1} + Z_t, X_{t-h}) = \phi.Cov(X_{t-1}, X_{t-h}) + Cov(Z_t, X_{t-h})$$

$$= \phi.Cov(Z_t, X_{t-h}) \quad \text{(Recall that } Z_t \text{ is uncorrelated with previous values of } X)$$

$$\eta(h) = \phi.\eta(h-1) \Rightarrow \eta(h) = \phi^h.\eta(0)$$

$$\rho(h) = \frac{\eta(h)}{\eta(0)} = \phi^h$$

Also for the variance at any time t

$$\eta(0) = Cov(X_t, X_t) = Cov(\mu + \phi.X_{t-1} + Z_t, \mu + \phi.X_{t-1} + Z_t)$$

$$= \mu^2 + \phi^2.Cov(X_{t-1}, X_{t-1}) + Cov(Z_t, Z_t) + 2\mu\phi.E[X_{t-1}] + 2\mu.E[Z_t]$$

$$= \mu^2 + \phi^2\eta(0) + \sigma^2 + \frac{2\phi\mu^2}{1-\phi}$$

$$\eta(0)(1-\phi^2) = \sigma^2 + \mu^2\left(\frac{1+\phi}{1-\phi}\right)$$

$$\eta(0) = \frac{\sigma^2}{1-\phi^2} + \frac{\mu^2}{(1-\phi)^2}$$

See Figures 7 and 8 for an illustration of an $AR(3)$ series along with its ACF and PACF plots.

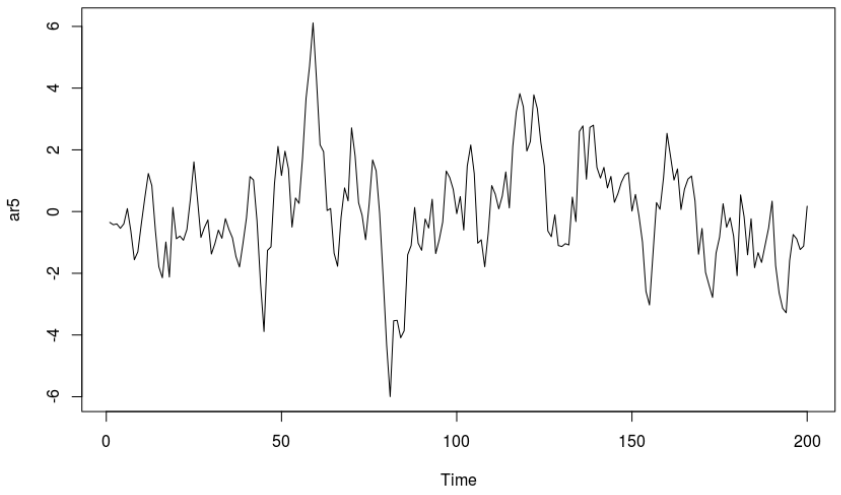


Figure 7: Simulated AR(3) Series

R Code for Figures 7 and 8

```
ar3 <- arima.sim(model=list(ar=c(.9,-.2, 0.1)),n=n)
plot.ts(ar3)
acf(ar3,type="correlation", level=95, lag.max=40, plot=T
    main='ACF Plot for AR(3)')
acf(ar3,type="partial", level=95, lag.max=40, plot=T,
    main='PACF Plot for AR(3)')
```

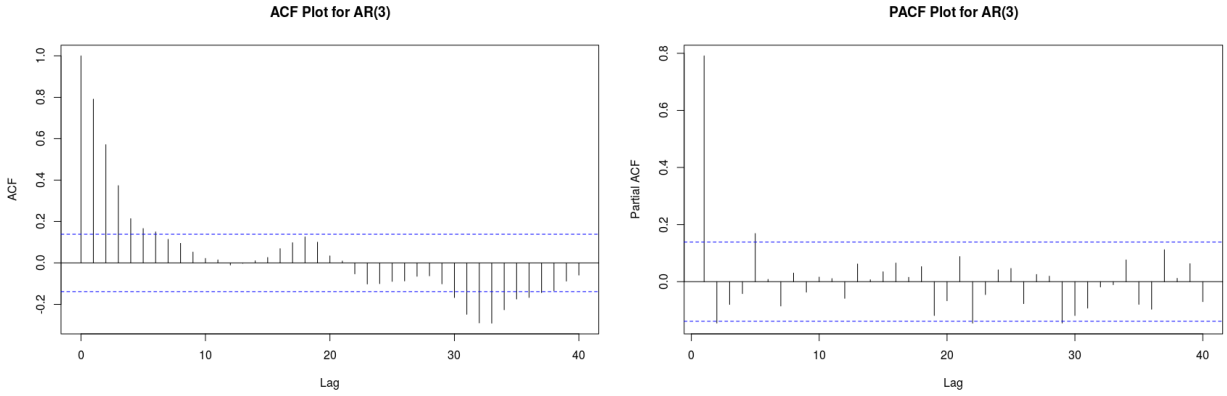


Figure 8: ACF and PACF Plots for a Simulated AR(3) Series

4.5 $ARMA(p,q)$ Models

Having looked at $MA(q)$ and $AR(p)$ separately, we can now combine the two into a composite $ARMA(p,q)$ model. A time series that exhibits characteristics of an $AR(p)$ and/or a $MA(q)$ process can be modeled using a $ARMA(p,q)$ model. The reason why we study these models is that almost any stationary series that we are likely to encounter in practise can be modeled using the $ARMA(p,q)$ family of models. As we discussed earlier in this document, our goal in time series analysis will essentially be to 'reduce' the given series to a stationary series through a series of transformations and then model the residual stationary series, finally leaving us with an error residue which we test for white noise. We will essentially model a stationary time series using the $ARMA(p,q)$ models. Formally we define an $ARMA(p,q)$ process as a series $\{X_t\}$ where

$$X_t = \mu + \sum_{i=1}^{p} \alpha_i X_{t-i} + \sum_{i=1}^{q} \beta_i Z_{t-i} + Z_t$$

There are several algorithms (Yule-Walker Estimation for Least Squares, Maximum Likelihood Methods, etc.) to fit an $ARMA(p,q)$ model to a given time series dataset the details of which go well beyond the scope of this course. There are several implementations however in R that we can make use of to forecast based on these models. It is however important to remember that all these algorithms crucially use the assumption that the series is stationary. Most of these implementations also indicate with an error message, in case the data to which you plan to fit

Process	ACF ($\rho_X(h)$)	PACF ($\pi_X(h)$)
$AR(p)$	Infinite waning exponential or sinusoidal tail	$\pi_X(h) = 0$ for $h > p$
$MA(q)$	$\rho_X(h) = 0$ for $h > q$	Infinite waning exponential or sinusoidal tail
$ARMA(p,q)$	Like $AR(p)$ for $h > q$	Like $MA(q)$ for $h > p$

Table 1: Theoretical Characteristics of the ACF and PACF plots for $ARMA(p,q)$ Processes

an $ARMA(p,q)$ model, is significantly different from what we would expect out of a stationary process. If that happens we essentially need to go back to the early stages of the modeling that try to transform the original data series into a stationary series. Another word of caution is that $ARMA(p,q)$ estimation methods typically work well for processes of small order (say $p+q<3$). So though one might obtain a fit using one of the standard implementations, a model with a higher order output by the algorithm is a typically a warning sign that the earlier stages of modeling (detrending etc.) need a relook.

To fit an $ARMA(p,q)$ model to data, as was highlighted earlier we try to get some clues about the nature of the model from the sample ACF and PACF plots of the given time series. We summarize our observations earlier about what to look for (theoretically) in the ACF/PACF plots in Table 1. There are some checks for stationarity that we can readily use. We can clearly see that the examples shown in Figure 1 are far from being stationary. See the box below for the output of these tests from one of those examples. The Dickey-Fuller test (`adf.test(.)`) works with "the data is not stationary" as the null hypothesis and therefore small p-values indicate stationarity. In this case the p=value is 0.39 which is way above the threshold of 0.05. The Kwiatkowski-Phillips-Schmidt-Shin (`kpss.test(,)`) test works with "the data is stationary" as the null hypothesis and hence small p-values indicate non-stationarity. One again see that the p-value is very small (0.01).

Stationarity Test

```
> exch <- read.csv('exchange-rate-twi-may-1970-aug-1995.csv')
> adf.test(exch$Exchange.Rate.TWI, alternative="stationary")

Augmented Dickey-Fuller Test

data:  exch$Exchange.Rate.TWI
Dickey-Fuller = -2.4404, Lag order = 6, p-value = 0.3907
alternative hypothesis: stationary

> kpss.test(exch$Exchange.Rate.TWI)

KPSS Test for Level Stationarity

data:  exch$Exchange.Rate.TWI
```

```
KPSS Level = 5.6365, Truncation lag parameter = 4, p-value = 0.01
```

Given a detrended, de-seasonalized series, to fit an $ARMA(p,q)$ model we can use the `arima` implementation in the standard R (`stats` package). We can either explicitly specify the order (p,q) of the model, try a few combinations using our guess from the ACF/PACF plots and pick the best among these or use the `auto.arima(.)` implementation that tries to automatically pick the best fitting model for the given data. However the `auto.arima(.)` will try carrying out differencing on the data to make it stationary if required and then fit an $ARMA(p,q)$ model to the differenced series. Differencing and ARIMA models are discussed in Section 7. The following piece of code tries fitting multiple ARMA models on a simulated data set in order to pick one that fits best.

Fitting ARMA Models

```
library(forecast)

# guessp, guessq - Initial guesses for p and q (from ACF/PACF plots)
# delta - window in which we want to change guessp and guessq to
#         look for alternative ARMA models
# timeseries - the one we want to analyze
tryArma <-  function(delta, guessp, guessq, timeseries) {
  df <- data.frame()
  # generate all possible ARMA models
  for (p in max(0,(guessp-delta)):(guessp+delta)) {
    for (q in max(0,(guessq-delta)):(guessq+delta)) {
      order <- c(p,0,q)
      # Fit a maximum likelihood ARMA(p,q) model
      armafit <- Arima(timeseries, order=order, method="ML")
      # Add the results to the dataframe
      df <- rbind(df, c(p, q, armafit$loglik, armafit$aic,
                        armafit$aicc, armafit$bic))
    }
  }
  names(df) <- c('p','q","log.likelihood", "AIC", "AICc", "BIC")
  return(df)
}

# Simulate a ARMA(2,2) series
arma22<-arima.sim(model=list(ar=c(.9,-.2),ma=c(-.7,.1)),n=200)
df <- tryArma(2,1,1,arma22)
> df
   p q log.likelihood      AIC     AICc      BIC
1  0 0      -293.2837 590.5674 590.6284 597.1641
2  0 1      -292.4698 590.9396 591.0621 600.8346
3  0 2      -290.0699 588.1399 588.3450 601.3331
4  0 3      -290.0697 590.1394 590.4487 606.6310
```

```
 5   1 0      -292.2228 590.4455 590.5680 600.3405
 6   1 1      -291.0014 590.0029 590.2080 603.1961
 7   1 2      -290.0695 590.1389 590.4482 606.6305
 8   1 3      -290.0699 592.1398 592.5751 611.9298
 9   2 0      -290.0443 588.0887 588.2938 601.2820
 10  2 1      -289.9375 589.8750 590.1843 606.3666
 11  2 2      -289.9372 591.8744 592.3096 611.6643
 12  2 3      -289.8280 593.6560 594.2394 616.7443
 13  3 0      -289.9481 589.8963 590.2056 606.3879
 14  3 1      -289.9374 591.8747 592.3099 611.6646
 15  3 2      -289.9374 593.8748 594.4581 616.9630
 16  3 3      -285.7924 587.5848 588.3387 613.9713
```

The sample output given has one column each for the measures we could use to compare the different models to finally pick one from them. A brief explanation of each of these measures is given below:

1. **Log Likelihood**: Log of the likelihood (probability) of the given dataset being generated by the chosen model. Higher the likelihood better the fit of the model to the data. Note that under the Gaussian assumption, the model that maximizes the likelihood also minimizes the cumulative square error. Since probabilities have to be < 1, the log-likelihoods are all negative.

2. **Akaike Information Criterion (AIC)**: This is an information theoretic measure of model appropriateness. It also takes into account the model complexity. In this case clearly as the order $(p+q)$ of the model increases, the model becomes more complex. As we have seen in the module on regression, in general more complex the model more prone it is to overfitting. We therefore try to pick a model that has as small a AIC as possible. AIC is defined as:

   $$AIC = 2k - 2\ln(L)$$

   where k is the number of model parameters and L is the likelihood of the data given the model.

3. **AIC Corrected (AICc)**: This is the AIC measure corrected for the size of the dataset, in this case the length of the time series n. AICc is defined as:

   $$AICc = AIC + \frac{2k(k+1)}{n-k-1}$$

4. **Bayes Information Criterion (BIC)**: This is a measure, similar to AICc, however arrived at using Bayesian methods. Again lower the BIC measure, better the model fit. BIC is defined as

   $$BIC = k\ln(n) - 2\ln(L)$$

Note that in all these measures the common pattern is that (i) Higher the likelihood, lower the measure, and (ii) Higher the model complexity, higher the measure. These measures are not always entirely consistent with each other and some amount of judgement (and trial) is required

in picking the right model. In our case we could possibly pick either the $(0,2)$ model or the $(1,1)$ model. Both have the same cumulative order, but order 2 for MA seems worse (longer range dependence and hence more complex) than unit order in both AR and MA. So finally we pick $(1,1)$.

One could also use the `auto.arima(.)` implementation which selects the best possible ARMA model automatically. Please find the sample code below.

ARMA Model Selection using *auto.arima*

```
> arma22<-arima.sim(model=list(ar=c(.9,-.2),ma=c(-.7,.1)),n=200)
> autoar <- auto.arima(arma22)
> autoar
Series: arma22
ARIMA(0,0,1) with zero mean

Coefficients:
         ma1
      0.2015
s.e.  0.0670

sigma^2 estimated as 1.018:  log likelihood=-285.12
AIC=574.24   AICc=574.3   BIC=580.83
```

In the following sections we discuss and carry out an end-to-end analysis of some of the examples we introduced in Figure 1.

5 End-to-End Time Series Analysis - An Introduction

We have explored stationarity in a lot of detail since that is often the most crucial part of time series anlaysis and modeling. We now look at how an end-to-end analysis is done and the two broad methodologies for the same, along with their respective pros and cons.We will explore these methodologies in detail in two separate sections below, each devoted to one of the methodologies. As we have discussed earlier, the first part of any time series analysis given some data is to try transforming it into one that is stationary.

1. **Classical Decomposition**: In this we attempt to explicitly de-trend and de-seasonalize the time series using curve fitting of the sort that we have seen in our discussion of regression. Good choice of trend and seasonal patterns will essentially imply that the residue after stripping the time series of the trend and seasonality that we have identified will be stationary. The obvious downside of this methodology is of course that we need to manually 'guess' a good trend and seasonal pattern in the data. This is often not such a big issue — on visualization most series we encounter in practice do reveal such trends rather easily.

2. **Differencing**: This approach defines a simple differencing operation on the data. There is enough mathematical and empirical evidence that repeated applications of differencing on the raw data will indeed make it stationary. So this method can easily be automated and that removes the onus of guessing an appropriate trend/seasonal pattern from the analyst. An important drawback of the differencing approach is that though the differencing eventually does remove non-stationarity, and produce a good fit, it is hard to visualize the trend and seasonality implicit in the data. An explicit visualization of such trends and seasonality can be very important for several applications.

Some sophisticated implementations that rely on a hybrid algorithm known as the Loess (Locally Weighted Scatterplot Smoothing) that relies on polynomial least-squares regression (typically linear or quadratic) on subsets data drawn using a variant of the k-nearest neighbours algorithm to build a **Loess** curve that captures the trend across the data set. A sample output using the R implementation of Loess on two of the examples in Figure 1 is shown in Figure 9.

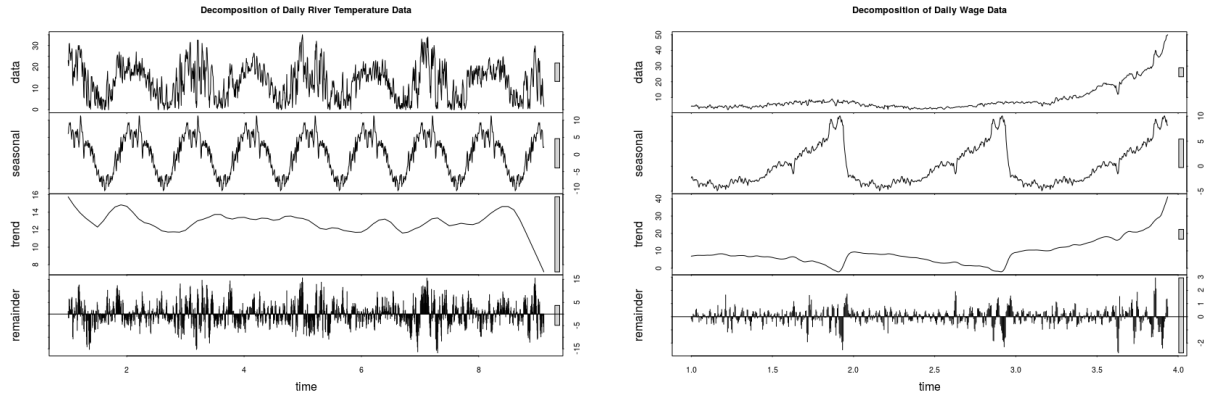


Figure 9: Decomposition using STL (Seasonal Decomposition of Time Series by LOESS)

6 Classical Decomposition Methods

In this approach we explicitly try removing the trend and seasonality from the original time series hoping to have a stationary residual series that we can model as an $ARMA(p,q)$ process. We discuss some of the details of this approach in this section.

6.1 Additive and Multiplicative Models

When we attempt to identify the trend and seasonality, we often imagine that the original time series is a 'composition' of multiple components — a trend component, seasonality and a stationary component. We could think of several reasonable ways to composing these different components into the original time series. Two common choices are (i) (**Additive Model**) to treat the original time series as a sum of these different components, (ii) (**Multiplicative Model**) as a product of these different components. To ensure that the composition is consistent we often

translate the time series so that each of the values in the time series is positive. Easy way to do this for a time series $\{X_t\}$ is to add a constant value δ to every value in the series, where

$$\delta = \begin{cases} 0 & \text{if } \min\{X_t\} > 0 \\ (1 - \min\{X_t\}) & \text{otherwise} \end{cases}$$

Let the values corresponding the trend part of the series be represented by the series $T_t\}$ and the seasonal part be the series $\{S_t\}$. Let $\{T_t\}$ represent the stationary residual series that we obtain after 'factoring out' the trend and seasonality from the original series $\{X_t\}_1^n$. Under the Additive or Multiplicative model assumptions we try finding $\{L_t\}$ and $\{S_t\}$ such that $\forall 1 \leq t \leq n$:

$$X_t = T_t + S_t + R_t \qquad \text{(Additive Model)}$$
$$X_t = T_t * S_t * R_t \qquad \text{(Multiplicative Model)}$$

As abroad thumbrule:

- When the magnitude of the seasonal pattern in the data increases as the data values increase, and decreases as the data values decrease. — **Multiplicative Model** may be a better choice. The Sales figures data used for the Bottom-Right plot in Figure 1 is a good candidate.
- When the magnitude of the seasonal pattern in the data does not directly correlate with the value of the series — **Additive Model** may be a better choice. The River Temperature, Daily Wage Data and Exchange Rate Data in Figure 1 are good candidates.

We typically work with additive models. An easy way to transform a multiplicative model to an additive model is to carry out the analysis on the log of the values in the time series.

6.2 Smoothing Techniques

One would notice from the examples of time series shown in Figure 1 that often the what we visually 'spot' as a trend or seasonality is obfuscated by the local 'spikiness' in the data; the spikiness could be due to noise or a very strong auto-regressive behaviour. Note from the earlier examples we have seen that visually auto-regressive data is hard to distinguish from from pure noisy data. Smoothing is the process of making the curve smoother by 'averaging' out the noise to make the trend and seasonality more apparent. See Figure 10 for an illustration of two of the examples in Figure 1 before and after smoothing. We will explore a few common ways of smoothing in the subsections below.

R Code for Figure 10

```
# Refer to examples in Figure 1
examples <- c(1, 3)
widths <- c(101, 51)
cols <- c('red', 'blue')
labels <- c('Raw', 'Smoothed')
for (i in seq(1,length(examples))) {
  example <- examples[i]
```

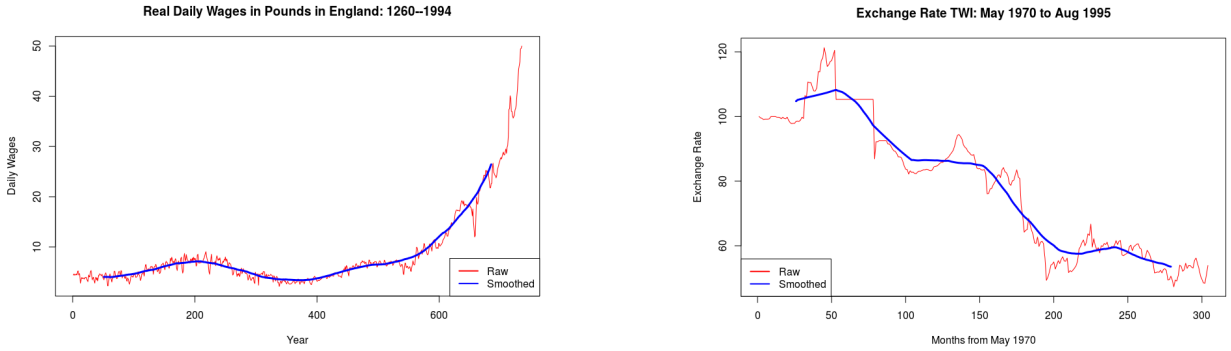


Figure 10: Moving Average Smoothing

```
  width <- widths[i]
  rawdata <- read.csv(filename[example])
  timeser <- ts(rawdata[,ycol[example]])
  plot(timeser, main=title[example], xlab=xlab[example],
       ylab=ylab[example], col=cols[1])
  smoothedseries <- filter(timeser, filter=rep(1/width, width),
                           method='convolution', sides=2)
  lines(smoothedseries, col=cols[2], lwd=2)
  legend("bottomleft", labels,  col=cols, lwd=2)
}
```

6.2.1 Moving Average Smoothing

The idea is that since the values in the series are expected to be of the form $f(t)+\epsilon$ where $f(t)$ is ideal predictable behaviour of the time series and ϵ is a zero mean noise, averaging the values of the time series over a window (with even number of values) must be something close to $f(t)$. Therefore it must be easier to 'fit' a $f(x)$ to a series obtained by replacing each value in the original series by the weighted average of the values in a time window around that value. In general, in *moving average* smoothing, from the original series $\{X_t\}$ we create a smoothed series $\{Y_t\}$ where

$$Y_t = \frac{1}{c}\sum_{i=-q}^{q} a_i.X_{t+i}$$

where $c, a_{-q}, a_{-q+1}, \ldots, a_{q-1}, a_q$ are constants with $c = \sum_{i=-q}^{q} a_i$. Such smoothing operators are also called *filters*. The moving average filters are in general *low pass* filters — they filter out the high-frequency components (the rapid changes, noisy wiggles) of the time series and let only the steady low-frequency components to pass through the filter. The simplest of moving-average filters is the one with

$$c = 2q+1, \quad a_{-q} = a_{-q+1} = \ldots = a_{q-1} = a_q = 1$$

where each value is replaced by the simple average of the values in a window of width $(2q+1)$ centered around the current value. It is possible to design filters with appropriate values for the

coefficients and the width of the window q to ensure that noise is dampened while a large class of trends pass through without distortion. For example the 15-point Spencer's Filter defined as

$$c = 320, \quad q = 7, \quad (a_0, a_1.a_2.a_3.a_4, a_5, a_6, a_7) = (74, 67, 46, 21, 3, -5, -6, -3), \quad a_i = a_{-i}$$

can be shown to let polynomials of upto degree 3 pass through without distortion. It is straightforward (although a bit laborious) to verify that

$$\text{if} \quad y_t = c_0 + c_1 t + c_2 t^2 + c_3 t^3 \quad \text{then} \quad \sum_{i=-7}^{7} a_i y_{t-i} = y_t$$

These are examples of *two sided moving averages*. One can also define **one sided** moving averages.

6.2.2 Exponential Smoothing

The other common way to carry out smoothing is what is known as *exponential smoothing*. In this scheme the current value is replaced by a weighted sum of the current value and the previous smoothed value. Exponential smoothing takes the form:

$$Y_t = \alpha.X_t + (1-\alpha)Y_{t-1}, \quad Y_1 = X_1$$

for some parameter $0 < \alpha < 1$. 'Unrolling' the recursion in the above expression we get

$$\begin{aligned} Y_t &= \alpha.X_t + (1-\alpha)Y_{t-1} \\ &= \alpha.X_t + (1-\alpha)(\alpha.X_{t-1} + (1-\alpha)Y_{t-2}) \\ &= \alpha.X_t + \alpha(1-\alpha).X_{t-1} + (1-\alpha)^2 Y_{t-2} \\ &= (1-\alpha)^{t-1} X_1 + \sum_{i=0}^{t-2} \alpha(1-\alpha)^i.X_{t-i} \end{aligned}$$

Notice that the 'smoothed value' Y_t in this case is just a weighted sum of the all the historical values in the time series, with the weight dropping exponentially as we go farther away from the current time. Small values of α result in higher levels of smoothing (may result in 'distorting' the original time series) and large values (close to 1) will not result in any smoothing at all. To see the effect of α on the smoothing achieved, refer Figure 11 for two of the examples in Figure 1.

R Code for Figure 11

```
# Refer to examples in Figure 1
for (example in c(2,3)) {
    rawdata <- read.csv(filename[example])
    timeser <- ts(rawdata[,ycol[example]])
    plot(timeser, main=title[example],
         xlab=xlab[example], ylab=ylab[example])
    cols <- c('red', 'blue', 'green', 'black')
    alphas <- c(0.02, 0.1, 0.8)
    labels <- c(paste('alpha =', alphas), 'Original')
    for (i in seq(1,length(alphas))) {
        smoothedseries <- HoltWinters(timeser, alpha=alphas[i],
```

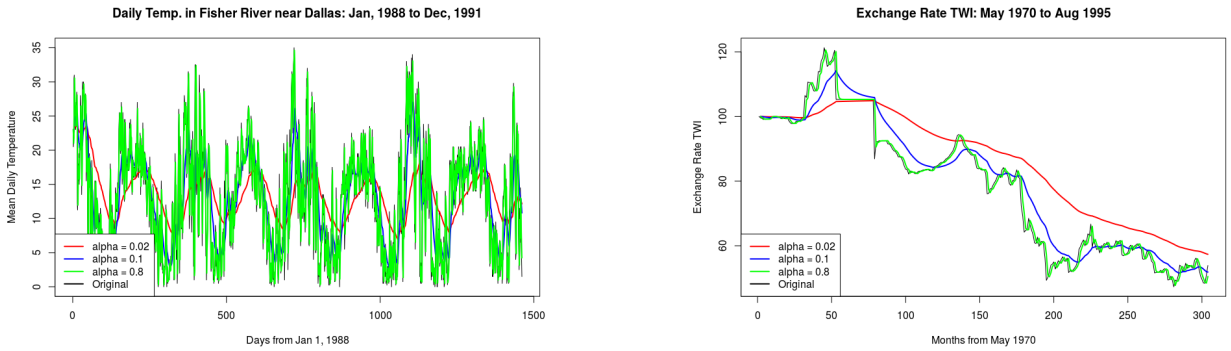


Figure 11: Exponential Smoothing

```
                                        beta=FALSE, gamma=FALSE)
        lines(fitted(smoothedseries)[,1], col=cols[i], lwd=2)
    }
    legend("bottomleft", labels,  col=cols, lwd=2)
}
```

Exponential Smoothing is also referred to *Holt-Winters* Smoothing, named after the inventors of this method. The R function implementing exponential smoothing is named `HoltWinters`.

6.3 Trend Detection

Trend detection is largely treated as a regression problem with time as the independent variable. Trend can be combined with seasonality through the choice of appropriate basis function for the regression — seasonal behaviour can be accounted for with trigonometric functions having appropriately chosen periods. Other option is to use smoothing to remove seasonality by using a moving average window of the same width as the seasonal periodicity, prior to trend fitting using regression.

6.4 Seasonality and Spectral Analysis

Seasonal behaviour is often guessed (from the scatter plot and/or domain awareness). Seasonality can either be dealt with separately or fit along with the trend as described above in the section on trend detection. If seasonal pattern is hard to guess then one could carry out a spectral analysis (decomposition of the time series treated as a signal into a weighted sum of periodic sinusoidal components) to find the dominant frequencies in the series and use that to model the seasonality explicitly.

7 ARIMA Models for Non-Stationary Series Analysis — the Box-Jenkins Methodology

7.1 Non-Seasonal ARIMA

Let's consider non-seasonal data. ARIMA models rely on a crucial observation about the time series formed by the differences between consecutive elements of a time series. Given a time series $\{X_t\}$, we call the series $\{Y_t\}$ as the differenced series if for all $t > 1$, $Y_t = X_t - X_{t-1}$. We can extend this definition to differencing to d levels or orders — starting from the original series $\{X_t\} \equiv \{X_t^{(0)}\}$ (the superscript refers to the orders of differencing from the original series), we can define $\{X_t^{(d)}\}$ inductively as the series where for each $t > 1$, $X_t^{(d)} = X_t^{(d-1)} - X_{t-1}^{(d-1)}$.

One way to think of the differencing operation on a time series is as an approximation of the slope of the time series 'curve'. Assuming the time stamps corresponding to the time series values are uniformly spaced (which the invariably the case) we can treat the time interval between consecutive time stamps as a unit interval. Consider a time series that fits a straight line as shown in Figure 12. It is easy to see that the differences between consecutive values in the time series is a constant — a line is by definition a curve with a constant slope. Another level of

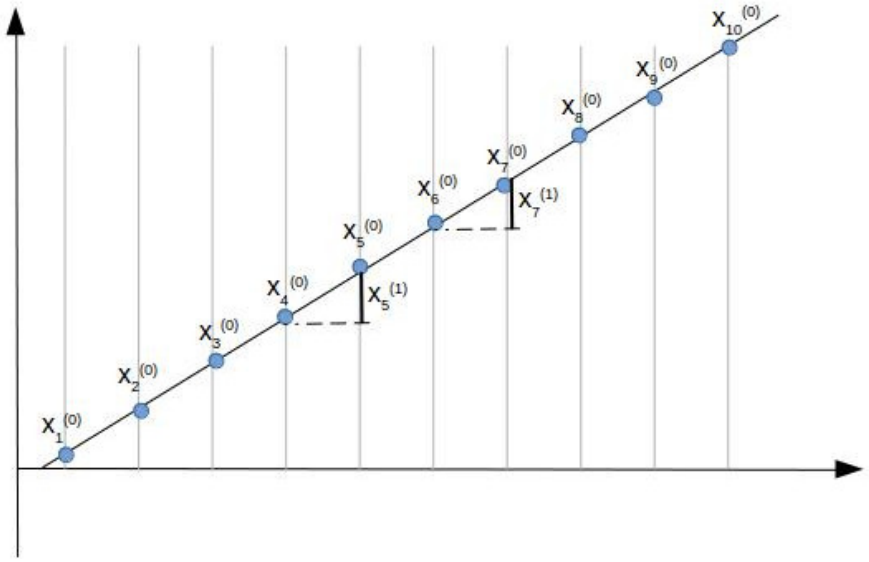


Figure 12: Differencing

differencing will result in a series that corresponds roughly to the slope of slope of the original series. For a quadratic curve, two levels of differencing will result in an approximately constant series. Extending this argument we can see that for a degree-k polynomial, we would require k levels of differencing to get a series that is stationary. In general it turns out that therefore, any time series can be reduced to a stationary time series after some finite levels of differencing and often the number of levels of differencing required is not very large. The ARIMA model therefore carries out a series of differencing operations on the time series to reduce it to a stationary series and finally models the stationary series as an ARMA model.

A more formal treatment would involve use of two *operators* — **Differencing** ∇ and **Backward Shift** β defined as:

$$\beta X_t = X_{t-1}; \quad \nabla X_t = X_t - X_{t-1} = (1-\beta)X_t$$

It turns out that though ∇ and β are operators, we can manipulate them just as we manipulate numbers. So second order differencing would become

$$\begin{aligned}\nabla^2 X_t &= \nabla X_t - \nabla X_{t-1} = X_t - 2X_{t-1} + X_{t-2} = X_t - 2\beta X_t + \beta\beta X_t = (1-2\beta+\beta^2)X_t \\ &= (1-\beta)(X_t - X_{t-1}) = (1-\beta)(1-\beta)X_t = (1-\beta)^2 X_t = (1-2\beta+\beta^2)X_t\end{aligned}$$

To see why repeated differencing will eventually lead to a stationary series, consider a quadratic series where $X_t = at^2 + bt + c + Z_t$ for some constants a, b, c and a random noise Z_t. Then

$$\begin{aligned}\nabla X_t &= (at^2 + bt + c) - \left(a(t-1)^2 + b(t-1) + c\right) + (Z_t - Z_{t-1}) \\ &= 2at + b - a + \nabla Z_t \\ \nabla^2 X_t &= (2at + b - a) - (2a(t-1) + b - a) + \nabla^2 Z_t \\ &= 2a + \nabla^2 Z_t\end{aligned}$$

Note that differencing a stationary series again results in a stationary series. We can see above that the series obtained by applying second order differencing to a series with a quadratic trend, is stationary. In general it easy to extend this argument to show that for any series $\{X_t\}$ with a degree k polynomial trend, a_k as the coefficient for the highest degree term, and a noise term Z_t

$$\nabla^k X_t = k! a_k + \nabla^k Z_t$$

a stationary series.

In practice however automatic ARIMA models do not perform as well as carefully handcrafted ones (using classical decomposition). Also repeated differencing works only for non-seasonal data.

7.2 Seasonal ARIMA

For seasonal data, the differencing 'window' will need to be the seasonal cycle length. Seasonal differencing is the process of generated a new differenced series that takes the difference between a value in the original series and a value with lag that is a multiple of the seasonal period. As observed earlier, seasonal cycles are often known apriori and hence this is not such a challenge in general.

8 Appendix A: R Code for White Noise Tests

Source for the sample R code given for the various white noise tests are taken from `http://faculty.washington.edu/dbp/s519/R-code/diagnostic-tests.R`. All these tests are written as R functions that take a time series object as the argument.

8.1 Turning Point Test in R

```
turning.point.test <- function(ts)
    {
        n <- length(ts)
        mu <- 2*(n-2)/3
        var <- (16*n-29)/90
        x <- embed(ts,3)
        test.sum <- sum((x[,2] > x[,1] & x[,2] > x[,3]) |
                        (x[,2] < x[,1] & x[,2] < x[,3]))
        test <- abs(test.sum-mu)/sqrt(var)
        p.value <- 2*(1-pnorm(test))
        structure(list(test.sum=test.sum, test=test, p.value=p.value,
                       mu=mu, var=var))
    }
```

8.2 Difference Sign Test in R

```
difference.sign.test <- function(ts)
    {
        n <- length(ts)
        mu <- (n-1)/2
        var <- (n+1)/12
        test.sum <- sum(diff(ts) > 0)
        test <- abs(test.sum-mu)/sqrt(var)
        p.value <- 2*(1-pnorm(test))
        structure(list(test.sum=test.sum, test=test, p.value=p.value, mu=mu,
                       var=var))
    }
```

8.3 Runs Test in R

```
runs.test <- function(ts)
    {
        n <- length(ts)
        n.pos <- sum(ts > 0)
        n.neg <- n - n.pos
        ts.binary <- ts
        ts.binary[ts > 0] <- 1
        ts.binary[ts < 0] <- -1
        n.runs <- sum(abs(diff(ts.binary)) > 0) + 1
        mu <- 2*n.pos*n.neg/n + 1
        var <- (mu-1)*(mu-2)/(n-1)
        sd <- sqrt(var)
```

```
        test.stat <- abs(n.runs-mu)/sd
        structure(list(n.runs=n.runs, n.pos=n.pos, n.neg=n.neg, mu=mu, var=var,
                       sd=sd, test.stat=test.stat, p.value=2*(1-pnorm(test.stat)),
                       reject.null=(test.stat > 1.96)))
    }
```

8.4 The Rank Test in R

```
rank.test <- function(ts)
    {
        n <- length(ts)
        P <- sum((matrix(rep(ts,n), ncol=n) < matrix(rep(ts,n), ncol=n, byrow=TRUE))
                 & outer(1:n, 1:n, function(x,y) x < y))
        mu <- n*(n-1)/4
        sig <- sqrt(n*(n-1)*(2*n+5)/72)
        test.stat <- abs(P-mu)/sig
        structure(list(P=P, mu=mu, var=sig^2, sd=sig, test.stat=test.stat,
                       p.value=2*(1-pnorm(test.stat)),
                       reject.null=(test.stat > 1.96)))
    }
```