**iGraph analysis**

[April 10, 2015](http://numerics.io/igraph-analysis/) [Numerics](http://numerics.io/author/numerics/)

**iGraph; a graph layout and analysis package for R**

The [iGraph package](http://igraph.org/r/) created by Gábor Csárdi is a huge library for everything graphs, graph layout and graph analysis in R. Much like any other graph library, it implies a learning curve and some math insights but it’s also refreshingly simple to use once you get through the basics. So, this article is bit of a tutorial in the right direction but also takes a pragmatic point of view by looking at concrete graph analysis situations rather than enumerating the API. In a way, one can endlessly explore graphs and their properties, there are so many values one can extract from a graph’s topology. Whether centrality, connectedness or community detection matters really depends on your (business) question.

Note that there is a textbook on [“Statistical Analysis of Network Data with R”](http://www.amazon.com/Statistical-Analysis-Network-Data-Use/dp/1493909827) wherein you can find plenty of in-depth info related to the library and that the [Nexus Network Repository](http://nexus.igraph.org) consolidates heaps of interesting networks (in R and Python format) to research. In fact, the Nexus search and load API is integrated in iGraph which makes it incredibly easy to experiment with all sorts of real-world graphs.

Let’s fetch, for instance, the ‘football’ network from Nexus;

library(igraph) # load the iGraph library

nexus.search("football") # search for networks related to football

## NEXUS 1-1/1 -- q: football

## [1] football 115/616 #11 Network of American college football games

nexus.info("football") # get info about it

## NEXUS UN-- 115 616 #11 football -- Network of American college football ga+

## + tags: sport; undirected

## + nets: 1/football

## + attr: Conference (v/n), name (v/c)

## + date: 2011-01-10

## + licence: Creative Commons by-sa 3.0

## + licence url: http://creativecommons.org/licenses/by-sa/3.0/

## + summary: Network of American football games between Division IA

## colleges during regular season Fall 2000.

## + description: The file football.gml contains the network of

## American football games between Division IA colleges during

## regular season Fall 2000, as compiled by M. Girvan and M.

## Newman. The nodes have values that indicate to which

## conferences they belong. The values are as follows:

## .

## 0 = Atlantic Coast

## 1 = Big East

## 2 = Big Ten

## 3 = Big Twelve

## 4 = Conference USA

## 5 = Independents

## 6 = Mid-American

## 7 = Mountain West

## 8 = Pacific Ten

## 9 = Southeastern

## 10 = Sun Belt

## 11 = Western Athletic

## .

## If you make use of these data, please cite M. Girvan and M. E.

## J. Newman, Community structure in social and biological

## networks, Proc. Natl. Acad. Sci. USA 99, 7821-7826 (2002).

## + formats: GraphML(4487);R-igraph(6201);Python-igraph(5138)

## + citation: M. Girvan and M. E. J. Newman, Community structure in

## social and biological networks , Proc. Natl. Acad. Sci. USA 99,

## 7821-7826 (2002).

g = nexus.get("football") # assign/create the graph

plot(g) # plot the graph

If you wish to change the layout you can simply use TAB-key to scroll through the options after typing ‘layout = layout.’;

or emphasize one area by changing the color and adding a region

utah = V(g)[name=="Utah"] #select from the vertices

utah.index = as.numeric(utah) # take the index of that vertex

utah$color = "red" # change the vertex color

wyoming = V(g)[name=="Wyoming"]

wyoming.index = as.numeric(wyoming)

wyoming$color = "red"

E(g)$color = "grey"

E(g, P=c("Wyoming", "Utah"))$color="red" # take the edge connecting the given names

V(g)$label.cex = .5 # set the font-size of all vertices

V(g)$label.cex[utah.index] = 1.5 # set the font-size of one particular vertex

V(g)$label.color[utah.index] = "Red"

## Warning in vattrs[[name]][index] <- value: number of items to replace is

## not a multiple of replacement length

V(g)$label.cex[wyoming.index] = 1.5

V(g)$label.color[wyoming.index] = "Red"

plot(g, layout=layout.auto,

mark.groups=list(c(utah.index, wyoming.index)), # make a group

mark.expand = 10, # increase the padding of the group

vertex.size = 2 # make the vertices a bit smaller

)

You can add weights to the vertices and the edges;

E(g)$weight = rnorm(ecount(g))

V(g)$weight = rnorm(vcount(g))

taking the vertices with positive weight into a subgraph and including the edge weight in the plot;

sg = induced.subgraph(g, which(V(g)$weight >0.7))

plot(sg, edge.label=round(E(sg)$weight, 3))

Sometimes you need to export the graph so it can be used elsewhere (e.g. the free [yEd graph app by yWorks](http://www.yworks.com/en/products/yfiles/yed/));

write.graph(g, "~/Desktop/myGraph.graphml", "GraphML")

A graph without loops and multiple edges between vertices is called simple and you can check that your graph is simple like so;

is.simple(sg)

## [1] TRUE

is.simple(g)

## [1] FALSE

meaning that the parent graph has a loop or multiple edges somewhere. So, which couple of vertices have multiple edges? If you would want to turn the graph into a simple graph you could in any case use the simplify command

is.simple(simplify(g))

## [1] TRUE

but that doesn’t answer our previous question. You can filter out the edges having a multiplicity

E(g)[is.multiple(g, eids=E(g))]

## Edge sequence:

##

## [52] Florida -- Auburn

## [344] Oklahoma -- KansasState

## [467] Marshall -- WesternMichigan

which reveals the list of suspicious vertices and we can use this to extract the subsgraph

mg = induced.subgraph(g, vids=c("Florida","Oklahoma","Marshall", "Auburn","KansasState","WesternMichigan"))

plot(mg)

Let’s move onto some analysis of communities and collaboration in the domain of astrophysics research by looking at the **netscience** graph from Nexus;

g = nexus.get("netscience")

V(g)$label.cex = .15

wc=walktrap.community(g)

plot(wc,g, vertex.size=.5, layout=layout.fruchterman.reingold)

A rather intricate and interesting picture can also be obtained by looking at the dendrogram;

plot(as.dendrogram(wc))

It’s clear from this plot that there are many small-scale collaborations. Let’s filter those out. There are 429 communities according to the wc output and the largest club is obtained via

x = which.max(sizes(wc))

largestGroup = induced.subgraph(g, which(membership(wc) == x))

plot(largestGroup)

By looking at the centrality of the vertices we can discover who is the most important person in the main network;

V(largestGroup)$label.cex = 0.5

plot(largestGroup, vertex.size=betweenness(largestGroup)/17)

which shows that there is a strong exponential law present;

hist(betweenness(largestGroup))

and the main person is gotten from

head(sort(betweenness(largestGroup), decreasing = TRUE),1)

## JEONG, H

## 1428.75

His degree is and immediate collaborators are

jeong = V(largestGroup)[name == "JEONG, H"]

degree(largestGroup,jeong)

## JEONG, H

## 27

neighbors = neighborhood(largestGroup, 1,nodes=c(as.numeric(jeong)))

lapply(neighbors, function(i){V(largestGroup)[i]$name})

## [[1]]

## [1] "JEONG, H" "ALBERT, R" "BARABASI, A" "VICSEK, T"

## [5] "OLTVAI, Z" "BIANCONI, G" "RAVASZ, E" "NEDA, Z"

## [9] "SCHUBERT, A" "FARKAS, I" "DERENYI, I" "GOH, K"

## [13] "KAHNG, B" "KIM, D" "OH, E" "HOLME, P"

## [17] "HUSS, M" "KIM, B" "YOON, C" "HAN, S"

## [21] "MASON, S" "TOMBOR, B" "PARK, Y" "RHO, K"

## [25] "PODANI, J" "SZATHMARY, E" "YOOK, S" "TU, Y"

<http://nexus.igraph.org/>

**High energy physics collaborations [hepcollab]**

**Description**

The collaboration network of scientists posting preprints on the high-energy theory archive at http://www.arxiv.org, 1995-1999, as compiled by M. Newman. The network is weighted, with weights assigned as described in the original papers. Please cite, as appropriate, one or more of:

M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA 98, 404-409 (2001).

M. E. J. Newman, Scientific collaboration networks: I. Network construction and fundamental results, Phys. Rev. E 64, 016131 (2001).

M. E. J. Newman, Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality, Phys. Rev. E 64, 016132 (2001).

**Metadata**

**Edge attribute 'weight'**

Edge weights, based on the number of common papers and the number of authors of these papers. See the publication(s) for the definition.

**Vertex attribute 'name'**

Author name.

**Publication(s)**

M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA 98, 404-409 (2001).

**Source**

<http://www-personal.umich.edu/~mejn/netdata/>

**Licence**

[Creative Commons by-sa 3.0](http://creativecommons.org/licenses/by-sa/3.0/)

**Astrophysics collaborations [astrocollab]**

**Description**

The collaboration network of scientists posting preprints on the astrophysics archive at http://www.arxiv.org, 1995-1999, as compiled by M. Newman. The network is weighted, with weights assigned as described in the original papers. Please cite, as appropriate, one or more of:

M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA 98, 404-409 (2001).

M. E. J. Newman, Scientific collaboration networks: I. Network construction and fundamental results, Pys. Rev. E 64, 016131 (2001).

M. E. J. Newman, Scientific collaboration networks: II. Shortest paths, weighted networks, and centrality, Phys. Rev. E 64, 016132 (2001).

**Metadata**

**Edge attribute 'weight'**

Edge weights, based on the number of common papers and the number of authors of these papers. See the publication(s) for the definition.

**Vertex attribute 'name'**

Author name.

**Publication(s)**

M. E. J. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. USA 98, 404-409 (2001).

**Source**

<http://www-personal.umich.edu/~mejn/netdata/>

**Licence**

[Creative Commons by-sa 3.0](http://creativecommons.org/licenses/by-sa/3.0/)

# Playful Reading of Learning Analytics Papers using ‘off the shelf’ R packages

[Leave a reply](http://www.laceproject.eu/tech/playful-reading-lak-2014-dataset-using-shelf-r-packages/#respond)

## Introduction

R is both a language and environment used by statisticians  that I have been tinkering with recently  for text analysis. I am no expert in statistics by any means, but I still like playing with the software because it has a [large collection of add-on packages](http://cran.r-project.org/), these packages can easily be imported in to an R environment giving the user access to techniques they may have come across in papers or on the web, they can then try the techniques on their own data.

I’ve taken some of the popular packages and techniques currently doing the rounds, modified them slightly and applied them to the the [LAK dataset](http://meco.l3s.uni-hannover.de:9080/wp2/?page_id=16) to see what sort of stuff comes out the other end. This post is a write up of my playing and is geared towards testing how possible it is to take off the shelf examples of data analysis, push our own dataset through them and get something interesting out the other side. As computer tinkerers know, it can be very difficult and time consuming to get the machine to do the things you want, particularly in an environment you are not that much of an expert in. As such this post will concentrate on the ‘is it possible?’ rather than a critique of the techniques themselves, perhaps I will save that for a part two.

I’ve tried to put in enough code examples and videos so you can play along if you like, you should be able to play along with the LAK dataset itself or with your own dataset. These are ‘off the shelf’ examples with small tweaks so I’ve also linked to the original source which you can follow up if you get stuck (or if notice that I am doing it wrong). If you do want to play along, I recommend using [R studio](http://www.rstudio.com/).

## Preparing LAK data set in R format

The basics of preparing RDF for R analysis is taken from Adam Cooper’s pre-processing script which [*can be found on his github*](https://github.com/arc12/LAK-Dataset).

The LAK dataset is available in RDF format to [download here](http://lak.linkededucation.org/), or if you know your SPARQL you can query at the end point like so: http://lak.linkededucation.org/request/lak-conference/sparql?query=[your sparql query].  The first thing I wanted to do with the data set is get it in to R in a sensible way so that I could try some of these R packages. There is an option to get the data in R format on the website, but the data you will get is the LAK 2012 dataset, which is two years out of date. Adam Cooper, who was involved in converting the LAK 2012 dataset hosts his script to convert the RDF to R dataframes on [his github](https://github.com/arc12/LAK-Dataset), but it only works with the 2012 dataset. I’ve updated the script to work with the 2014 dataset, to use it simply [download it here](https://github.com/LACE-TechBlog/LAK-Dataset/blob/master/LAK%20RDF2RData.R), place the LAK dataset in the same directory within a folder called Data. The script itself is quite straight forward, but the important bits are:

1) We import the rrdflibs and rrdf R packages, we need them to work with the LAK RDF dump, we then create a new RDF triple store object and load in the data. We can see how many triples there are with the summarize.rdf command:



|  |  |
| --- | --- |
| 1  2  3  4  5 | library(rrdflibs)  library(rrdf)  triples<-new.rdf()  triples<-load.rdf("LAK-DATASET-DUMP.rdf", "RDF/XML")  summarize.rdf(triples) |

2) We can then write SPARQL queries, run them against the triple store object and return the results as a dataframe. I have three queries all of which are slightly modified versions of Adam’s queries. I won’t go in to detail on how to write SPARQL queries here, [but there is a great guide here](https://code.google.com/p/tdwg-rdf/wiki/Beginners6SPARQL) that I often refer to. I’m not the best at SPARQL, so let me know if there are any obvious mistakes:

Creating a dataframe of people, names, location and organisation:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | people.query<- paste("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>",                       "PREFIX swrc: <http://swrc.ontoware.org/ontology#>",                       "PREFIX foaf: <http://xmlns.com/foaf/0.1/>",                       "SELECT ?person ?name ?location ?affiliation WHERE {",                       "?person rdf:type foaf:Person;",                       "foaf:name ?name .",                       "OPTIONAL{",                       "?person foaf:based\_near ?location;",                       "swrc:affiliation ?affiliation;",                       "foaf:firstName ?firstName;",                       "foaf:lastName ?lastName",                       "}",                       "}")    people<-as.data.frame(sparql.rdf(triples,people.query,rowvarname="person")) |

Creating a dataframe of paper information and contents:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | papers.query<-paste("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>",                      "PREFIX swrc: <http://swrc.ontoware.org/ontology#>",                      "PREFIX dc: <http://purl.org/dc/elements/1.1/>",                      "PREFIX swc: <http://data.semanticweb.org/ns/swc/ontology#>",                      "PREFIX bibo: <http://purl.org/ontology/bibo/>",                      "PREFIX led: <http://data.linkededucation.org/ns/linked-education.rdf#>",                      "SELECT ?paper ?origin ?month ?year ?title ?abstract ?content WHERE {",                      "?paper rdf:type swrc:InProceedings;",                      "swc:isPartOf ?origin;",                      "dc:title ?title;",                      "swrc:abstract ?abstract;",                      "swrc:month ?month;",                      "swrc:year ?year .",                      "OPTIONAL{{ ?paper led:body ?content}",                      "UNION  {?paper bibo:content ?content}}",                      "}")    papers<-as.data.frame(sparql.rdf(triples,papers.query,rowvarname="paper")) |

Creating a dataframe of authorship data:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | authorship.query<-paste("PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>",                          "PREFIX swc: <http://data.semanticweb.org/ns/swc/ontology#>",                          "PREFIX foaf: <http://xmlns.com/foaf/0.1/>",                          "SELECT ?person ?name ?paper ?origin WHERE {",                          "?person rdf:type foaf:Person;",                          "foaf:name ?name ;",                          "foaf:made ?paper .",                          "OPTIONAL{?paper swc:isPartOf ?origin}",                          "}")    authorship<-as.data.frame(sparql.rdf(triples,authorship.query)) |

You can now poke about any of the dataframes, RStudio users can click the little table next to a dataframe in the environment tab to view it. If you wish to write these objects to disk, you can using the save command (and can be loaded with the load command):



|  |  |
| --- | --- |
| 1 | save( people, papers, authorship, mtimes, file="LAK-Dataset.RData") |

Now we have all the data in R we can find some basic stats about the data, we can find the number of triples in the RDF using:



|  |  |
| --- | --- |
| 1 | summarize.rdf(triples) |

count the numbers of papers per year by simply counting the rows in the R data frame ‘papers’ with the command:



|  |  |
| --- | --- |
| 1 | nrow(papers[papers$year ==  2008,]) |

and count the mean number of words with:



|  |  |
| --- | --- |
| 1 | length(unlist(lapply(papers[papers$year == 2008,]$content, function(i) strsplit(i, ” “)[[1]])))/ nrow(papers[papers$year == 2008,]) |

Using these commands we can whip up some basic stats about the data set:

|  |  |  |
| --- | --- | --- |
| **Year** | **Number of Papers** | **Mean Number of Words** |
| 2008 | 31 | 2424 |
| 2009 | 32 | 2609 |
| 2010 | 61 | 1521 |
| 2011 | 84 | 2684 |
| 2012 | 101 | 3778 |
| 2013 | 153 | 3070 |
| 2014 | 124 | 2430 |

A side note, this seems to be quite a few papers less than the LAK data site claims there are. This could be one of two problems, the process I am using to grab the contents of the papers is dodgy or the content is missing from the data set. Exploring the dataset I do have a problem that the contents of the papers seem to have different predicates for the content of a paper. What I have been doing is using the the SPARQL UNION keyword (thanks @wilm) to grab the contents of the paper,  remove any empty papers, and then remove any duplicate ones with the following code:



|  |  |
| --- | --- |
| 1 | papers <- papers[!(is.na(papers$content) | papers$content==""), papers<-papers[!duplicated(papers),] |

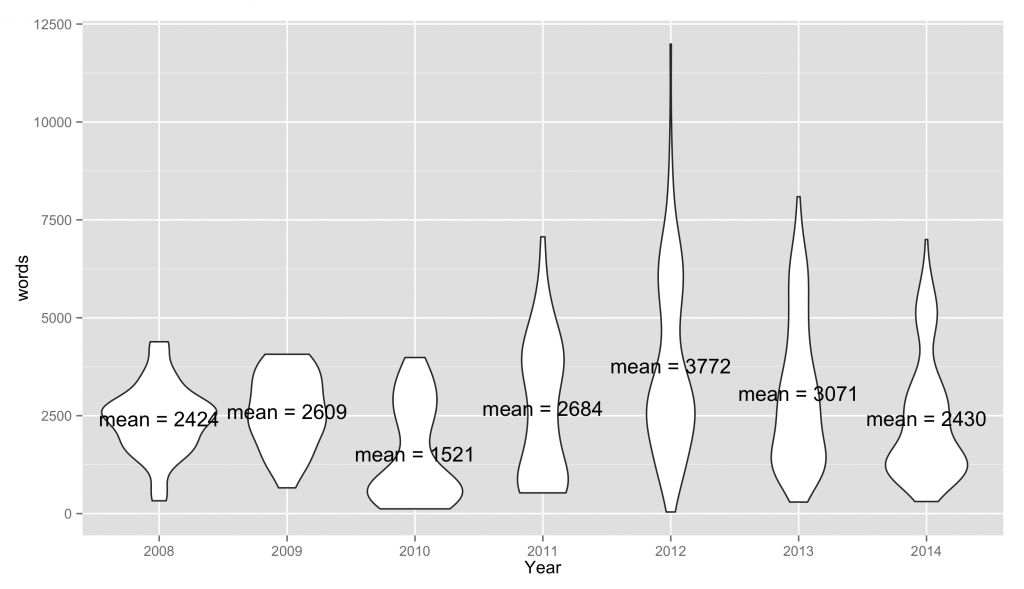
If you can spot a problem I would love an update, in the mean time I am going to contact LAK about this to explain my problem, still I think I have enough data to do an analysis, I can always come back update the post.

Ben Marwick has an [interesting way](https://github.com/benmarwick/dayofarchaeology/blob/master/003_clean_fulltext.r) to plot of the distribution of words per post by year, which I have slightly modified to work with our dataframe. The idea is that we make a dataframe where each row is a paper with columns for the number of words and the year it was published, we then use ggplot2 to  plot it



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | wordsperpaper <- data.frame(  words = unlist(lapply(papers$content, function(i) length(strsplit(i, " ")[[1]]))),  year = papers$year, stringsAsFactors = FALSE)    # To create the median labels, you can use by  meds <- round(c(by(wordsperpaper$words, wordsperpaper$year, mean)),0)      require(ggplot2)  ggplot(wordsperpaper, aes(as.factor(year), words, label=rownames(wordsperpaper))) +  geom\_violin() +  geom\_text(data = data.frame(), aes(x = names(meds) , y = meds,  label = paste("mean =", meds))) +  xlab("Year") |

results in:

[](http://www.laceproject.eu/tech/files/2014/12/averagewordsayear.png)

Plot of the distribution of words per paper by year in LAK 14 dataset

## Topic Moddeling

The way in which I approached Topic Modelling for the LAK dataset was taken from [*Ben Marwick’s reading of archeology blog posts.*](https://github.com/benmarwick/dayofarchaeology)

Now that we have the data in a format in which we can explore the data in a format we want we can do some analysis. A popular machine learning technique at present is to find ‘Topic Models’ in a collection of documents. The idea is that we can use a statistical model to find abstract “topics” that appear in the documents, then we can use that model to predict how much a document contains about each of these topics. The most common example of topic modelling currently seems to be the Latent Dirichlet allocation (LDA) model.

If you’ve been following along with this post then you should already have all the text of the documents in the papers dataframe.  There are a few examples you can follow to do this, there is an [R package you can download and use](http://www.jstatsoft.org/v40/i13/paper), but the general census seems to be that java tool [MALLET](http://mallet.cs.umass.edu/), can do it much faster, fortunately for us there is a [R wrapper for tha](http://cran.r-project.org/web/packages/mallet/mallet.pdf)t.

1. Add a new column to our papers dataframe

I started by adding a new column to the papers dataframe describing what conference the paper was from. I thought this might be useful to see if certain topics appear more at different conferences:



|  |  |
| --- | --- |
| 1  2  3 | #add a column for conference  papers$conference<-(substring(papers$origin , 57))  papers$conference<-substr(papers$conference, 1, nchar(papers$conference)-12) |

2. Import the papers in to Mallet format

First we need to load the Mallet library, then create a dataframe that contains an unique id, the paper text and the year it was published. Then we create a mallet instance out of it. Below you can see I have also loaded a file of common stop words “en.txt”, you can create your own or borrow one from somewhere (a good one comes packaged with Mallet)



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | documents <- data.frame(text = papers$content,                          id = make.unique(papers$conference),                          class = papers$year,                          stringsAsFactors=FALSE)    mallet.instances <- mallet.import( documents$text , documents$text , "en.txt", token.regexp = "\\p{L}[\\p{L}\\p{P}]+\\p{L}") |

3.Set the Topic Model parameters.

The next steps are to set the parameters, these are taken from Ben’s example. I ran this many times trying different parameters and I’m not sure there is a ‘best’ way to do it. I think the parameters will depend greatly on your data set, the number of papers you have and size of these paper. The important things you might want to change are the number of topics (n.topics <- 30) and the number of words per topics. This might take a few minutes to run, particularly if you have an old creaky machine like mine.

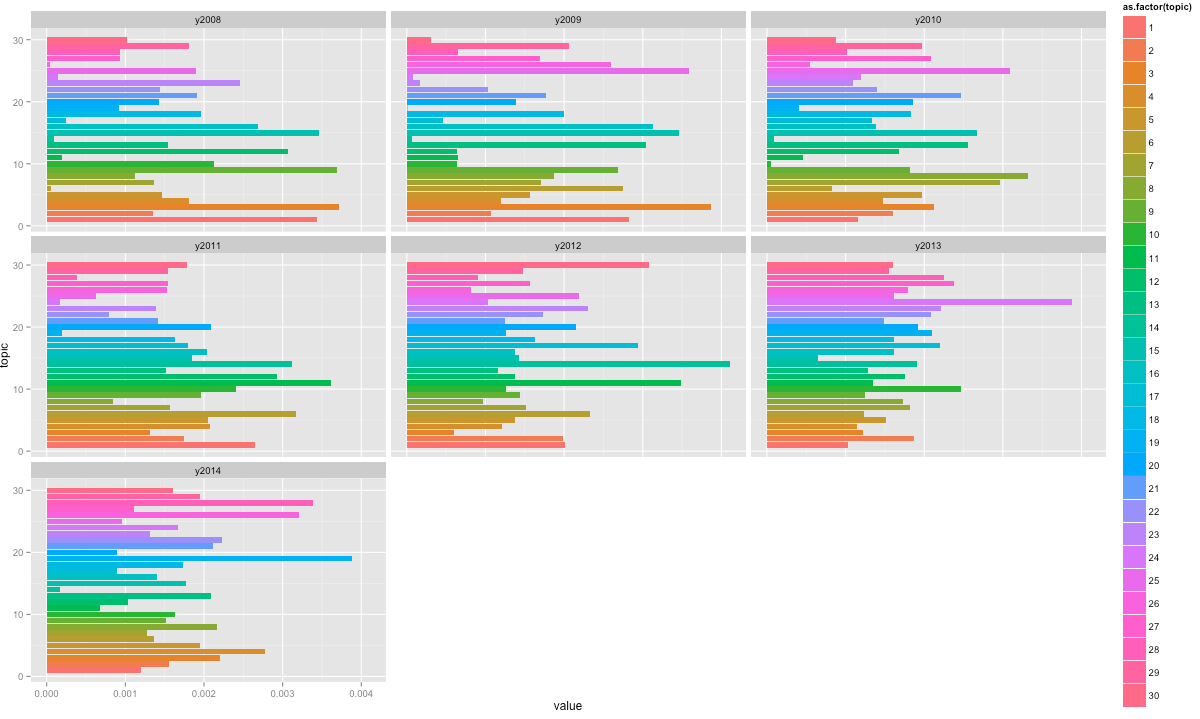


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | ## Create a topic trainer object.  n.topics <- 30  topic.model <- MalletLDA(n.topics)    #loading the documents  topic.model$loadDocuments(mallet.instances)    ## Get the vocabulary, and some statistics about word frequencies.  ## These may be useful in further curating the stopword list.  vocabulary <- topic.model$getVocabulary()  word.freqs <- mallet.word.freqs(topic.model)    ## Optimize hyperparameters every 20 iterations,  ## after 50 burn-in iterations.  topic.model$setAlphaOptimization(20, 50)    ## Now train a model. Note that hyperparameter optimization is on, by default.  ## We can specify the number of iterations. Here we'll use a large-ish round number.  topic.model$train(500)    ## NEW: run through a few iterations where we pick the best topic for each token,  ## rather than sampling from the posterior distribution.  topic.model$maximize(10)    ## Get the probability of topics in documents and the probability of words in topics.  ## By default, these functions return raw word counts. Here we want probabilities,  ## so we normalize, and add "smoothing" so that nothing has exactly 0 probability.  doc.topics <- mallet.doc.topics(topic.model, smoothed=T, normalized=T)  topic.words <- mallet.topic.words(topic.model, smoothed=T, normalized=T)    # from http://www.cs.princeton.edu/~mimno/R/clustertrees.R  ## transpose and normalize the doc topics  topic.docs <- t(doc.topics)  topic.docs <- topic.docs / rowSums(topic.docs)    ## Get a vector containing short names for the topics  topics.labels <- rep("", n.topics)  for (topic in 1:n.topics) topics.labels[topic] <- paste(mallet.top.words(topic.model, topic.words[topic,], num.top.words=5)$words, collapse=" ")  # have a look at keywords for each topic  topics.labels |

You should be left with topics and some information about how much of your documents relate to each topic. These are the 30 abstract topics in the LAK dataset that I got on my run, due to the generative nature of the model yours will be different:

 [1] “tree classifier classification classifiers tutor”  
[2] “patterns students actions sequences pattern”  
[3] “model models performance data features”  
[4] “lak data topics papers dataset”  
[5] “difficulty item time items system”  
[6] “hints state hint states game”  
[7] “student problem problems students tutor”  
[8] “number figure results work data”  
[9] “words text word writing document”  
[10] “learning analytics learners social learner”  
[11] “students data student learning behavior”  
[12] “concept dialogue concepts knowledge act”  
[13] “data student academic university institutions”  
[14] “students student study average higher”  
[15] “questions question answer code student”  
[16] “students posts topic discussion words”  
[17] “set algorithm method values section”  
[18] “data items skills item cognitive”  
[19] “network analysis social interaction group”  
[20] “cluster clusters clustering k-means algorithm”  
[21] “students scores learning features reading”  
[22] “data teachers students time tool”  
[23] “rst graph events dierent interaction”  
[24] “students time facial task participants”  
[25] “learning causal variables data condition”  
[26] “learning analysis based research system”  
[27] “rules data students attributes table”  
[28] “students video assignment lecture time”  
[29] “student skill knowledge parameters students”  
[30] “users user learning datasets objects”

While I find poking around the output very interesting I find it hard to visualise the results and would appropriate any advice. Ben’s solution was to comparing the average proportions of each topic across all documents for each year using ggplot2. For our dataset it looks like this:

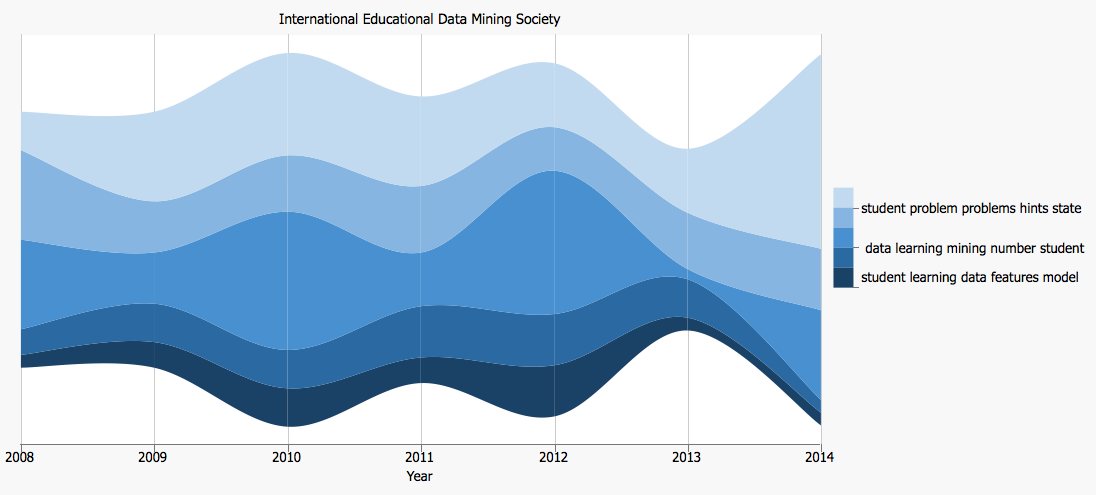
[](http://www.laceproject.eu/tech/files/2014/12/topicmodels.png)

But I think its very hard to follow, you can kind of see the shift away from topic “users user learning datasets objects” over time and the shift to “model models performance data features”.

Another thing we could do is a method I have seen on [sapping attention](http://sappingattention.blogspot.co.uk/2014/12/typical-tv-episodes-visualizing-topics.html#more), where we make a streamgraph for topics over time. Since the educational datamining conference is the only conference that seems to have papers for all 6 years I redid the topic model for that conference only and then mapped the with the y-axis  representing percent of top 5 topics in the papers, and the x-axis corresponding to the year the papers were published.

1.students learning student data features model  
2.questions student words features students question  
3.students data learning mining number student  
4.model student data models skill knowledge  
5.students student problem problems hints state

over time:

[](http://www.laceproject.eu/tech/files/2014/12/Screen-Shot-2015-01-12-at-13.25.04.png)

## Phrase Finding

Ben Schmidt runs the excellent blog [Sapping Attention](http://sappingattention.blogspot.co.uk/), in [some of my favourite posts](http://sappingattention.blogspot.co.uk/2012/02/downton-abbey-anachronisms-season.html) on the blog he mines the Google Ngram data to see if the language of TV period dramas matches the language of books from the same era. Ben doesn’t share much code, but from what I can gather his general technique to Ngram is  to be to use the [tm package](http://cran.r-project.org/web/packages/tm/index.html) to do some pre-processing of text and then use RWeka to create an Ngram tokenizer.

Since creating an R dataframe from the RDF we can access the text of the papers in papers$text, First we want to import the tm  and Rweka libraries, I also import slam so I use the [rollup](http://www.inside-r.org/packages/cran/slam/docs/rollup) function. Also I set mc.cores = 1 because of issue with forking in the tm package on Mac OS X:



|  |  |
| --- | --- |
| 1  2  3  4 | library(tm)  library("RWeka")  library("slam")  options(mc.cores=1) |

The pre-processing on the text using the tm is quite straight forward, we want to create a corpus and then remove stop words, numbers and punctuation. I then make sure all the text is lower case:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #prepare text  corpus <- Corpus(VectorSource(papers$content)) # create corpus object  corpus <- tm\_map(corpus, mc.cores=1, removePunctuation)  corpus <- tm\_map(corpus, removeNumbers, mc.cores=1)  corpus <- tm\_map(corpus, removeWords, stopwords("english"), mc.cores=1)    # convert all text to lower case  corpus <- tm\_map(corpus, tolower, mc.cores=1)    #proplem with to lower means we need to make it type of plain text document again  corpus <- tm\_map(corpus, PlainTextDocument) |

We can then create a term document matrix from this corpus:



|  |  |
| --- | --- |
| 1  2 | #make the term document matrix  tdm <- TermDocumentMatrix(corpus) |

Using the weka package we can then create a tokenizer and a dataframe of the phrases arranged by the amount of times they have been used. Below I am looking for bigrams but you can change the number of words in the phrases by editing where I have put 2:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | #tokenizer for tdm with ngrams  BigramTokenizer <- function(x) NGramTokenizer(x, Weka\_control(min = 2, max =2))  tdm <- TermDocumentMatrix(corpus, control = list(tokenize = BigramTokenizer))  #create dataframe and order by most used  rollup <- rollup(tdm, 2, na.rm=TRUE, FUN = sum)  mydata.df <- as.data.frame(inspect(rollup))  colnames(mydata.df) <- c("count")    mydata.df$ngram <- rownames(mydata.df)  newdata<-mydata.df[order(mydata.df$count, decreasing=TRUE), ] |

I created a tokenizor for phrases of length 2, 3,4 and 5 words and took out the phrases that would appear in generic papers. Many of the top phrases seem to be phrases that you would expect to see in an academic paper, for example ‘et al’ or ‘for example’, I took these phrases out and I was left with the following top phrases in the LAK dataset:

|  |  |
| --- | --- |
| **Phrase** | **Occurances** |
| learning analytics | 1097 |
| data mining | 832 |
| knowledge tracing | 437 |
| educational data | 400 |
| student performance | 353 |
| social network | 296 |
| student model | 278 |
| educational data mining | 272 |
| intelligent tutoring systems | 164 |
| social network analysis | 142 |
| national science foundation | 108 |
| bayesian knowledge tracing | 94 |
| data mining techniques | 85 |
| sequential pattern mining | 72 |
| intelligent tutoring system | 71 |
| social learning analytics | 60 |
| item response theory | 58 |
| dynamic cognitive tracing | 56 |
| logistic regression model | 56 |
| predicting student performance | 54 |
| learning management systems | 48 |
| knowledge tracing model | 47 |

I wondered how these changed over time in the dataset. To do this I created a loop that created a Corpus for each year, and then checked how many times each phrase appeared in each corpus and wrote it to a CSV. Loops are a bit controversial in the R community as they are very resource heavy, I’m sure there is a better way to do this, but rearranging data frames and the such can be quite difficult, with a bit of trial and too much room for error. The content for each year can be grabbed with: papers[papers$year == 2008,]$content, so creating a Corpus of text with papers from the LAK dataset published in 2008 looks like this:



|  |  |
| --- | --- |
| 1  2  3  4  5 | #prepare text  corpus <- Corpus(VectorSource(papers[papers$year == 2008,]$content)) # create corpus object  corpus <- tm\_map(corpus, mc.cores=1, removePunctuation)  corpus <- tm\_map(corpus, removeNumbers, mc.cores=1)  corpus <- tm\_map(corpus, removeWords, stopwords("english"), mc.cores=1) |

## [lak top phrases](http://www.laceproject.eu/tech/files/2014/12/lak-top-phrases.png)

## Creating a Network Graph

There are lots of ways to import data in to R for social network analysis, I guess each has it’s own advantages and might be easier or harder depending on the format of your data. [*My favourite resource on this can be found here*](http://www.laceproject.eu/tech/playful-reading-lak-2014-dataset-using-shelf-r-packages/%20http:/www.shizukalab.com/toolkits/sna/sna_data).

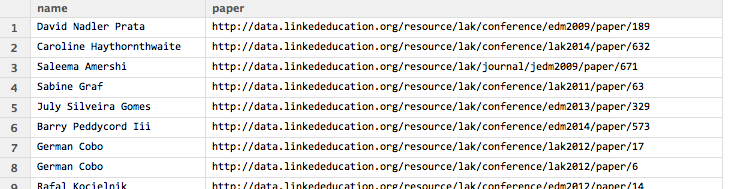
We can do simple network and graph analysis in R using the [iGraph](http://cran.r-project.org/web/packages/igraph/index.html) package and honestly, while I’m not much of network diagrams that don’t really explain themselves, I think there is something in being able to provide the data in a format that people can play with in tools such as Gephi. This is the way I do it, it seems to be the most simple way, but I lose some data along the way:

1.  Identify a dataframe that has a list of nodes, and items that might connect these nodes

In the authorship dataframe we have a list of authors and papers that they are have worked on. You can list these in R with the command:



|  |  |
| --- | --- |
| 1 | authorship[,c("name","paper")] |

[](http://www.laceproject.eu/tech/files/2014/12/names-and-papers.png)

Names and papers they worked on

2.Create an adjancy matrix

This creates a matrix of names and the number of times they appear on a paper



|  |  |
| --- | --- |
| 1 | M = as.matrix( table(authorship[,c("name","paper")])) |

This creates a matrix of the number of times users appear on the same paper together:



|  |  |
| --- | --- |
| 1 | Mrow = M %\*% t(M) |

3. Create iGraph object

We can create an Igraph object out of the adjancy matrix



|  |  |
| --- | --- |
| 1 | iMrow = graph.adjacency(Mrow, mode = "undirected") |

In this example I will remove edges and loops



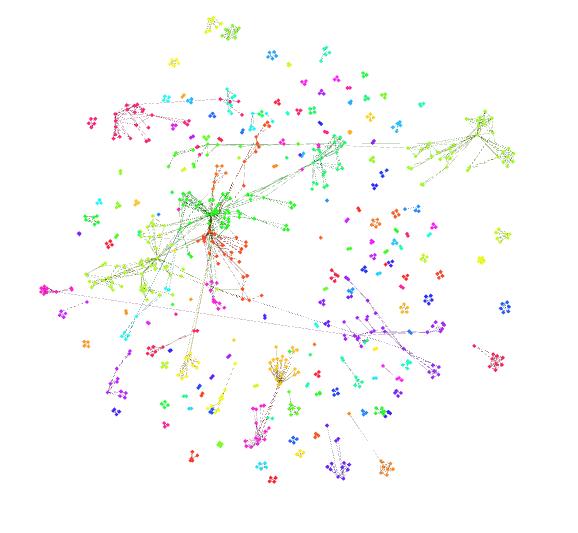
|  |  |
| --- | --- |
| 1 | iMrow <- simplify(iMrow) #remove loops and multiple edges |

3. Write to gephi file.



|  |  |
| --- | --- |
| 1 | write.graph(iMrow, file="graph.graphml", format="graphml"); |

The file can then be imported in to Gephi by going to file->open. There are lots of examples of using Gephi on the web, so I am not going to mess with the file much, I like this example of using Gephi but I recommend getting an understanding of some of the measurements elsewhere, if your library has it, skip to the the last chapter in Webb’s 3rd edition of [Statical Pattern Recognition](http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470682272.html), it is brilliant and really easy to understand. You can save as in other formats by replacing the graphml in format=”graphml” with pajek, gml, dot, etc.

[](http://www.laceproject.eu/tech/files/2014/08/gephi-lak.png)

[Click for a graphml file of authors who have co-authored papers in the LAK dataset. Coloured by communities detected with louvain modularity method.](https://www.dropbox.com/s/vo9zutjsyvv53uf/graph.graphml?dl=0)

While this is a very easy and quick method there are some drawbacks, the only attribute nodes and edges have are their label intact of the process.

**ToDo**

1. Explore other ways of doing SNA in R

2. Better ways to explain and explore topic models

## Final versions of the scripts

Will be on github once I’ve cleaned them up!

## The places I borrowed from

[Adam Cooper’s Github](https://github.com/arc12)  
[Ben Marwick’s Github](https://github.com/benmarwick/dayofarchaeology)  
[Ben Schmid’s Sapping attention](http://sappingattention.blogspot.co.uk)

This entry was posted in [LAK](http://www.laceproject.eu/tech/category/lak/) on [December 23, 2014](http://www.laceproject.eu/tech/playful-reading-lak-2014-dataset-using-shelf-r-packages/) by [David Sherlock](http://www.laceproject.eu/tech/author/david-sherlock/).

<https://datamarket.com/data/list/?q=provider:tsdl>

Time Series Data Library

* [index](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/genindex.html)
* [previous](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/installr.html) |
* [Time Series 0.2 documentation](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/index.html) »

### [Table Of Contents](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/index.html)

* [Using R for Time Series Analysis](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html)
  + [Time Series Analysis](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#time-series-analysis)
  + [Reading Time Series Data](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#reading-time-series-data)
  + [Plotting Time Series](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#plotting-time-series)
  + [Decomposing Time Series](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#decomposing-time-series)
    - [Decomposing Non-Seasonal Data](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#decomposing-non-seasonal-data)
    - [Decomposing Seasonal Data](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#decomposing-seasonal-data)
    - [Seasonally Adjusting](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#seasonally-adjusting)
  + [Forecasts using Exponential Smoothing](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#forecasts-using-exponential-smoothing)
    - [Simple Exponential Smoothing](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#simple-exponential-smoothing)
    - [Holt’s Exponential Smoothing](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#holt-s-exponential-smoothing)
    - [Holt-Winters Exponential Smoothing](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#holt-winters-exponential-smoothing)
  + [ARIMA Models](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#arima-models)
    - [Differencing a Time Series](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#differencing-a-time-series)
    - [Selecting a Candidate ARIMA Model](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#selecting-a-candidate-arima-model)
      * [Example of the Ages at Death of the Kings of England](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#example-of-the-ages-at-death-of-the-kings-of-england)
      * [Example of the Volcanic Dust Veil in the Northern Hemisphere](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#example-of-the-volcanic-dust-veil-in-the-northern-hemisphere)
    - [Forecasting Using an ARIMA Model](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#forecasting-using-an-arima-model)
      * [Example of the Ages at Death of the Kings of England](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#id2)
      * [Example of the Volcanic Dust Veil in the Northern Hemisphere](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#id3)
  + [Links and Further Reading](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#links-and-further-reading)
  + [Acknowledgements](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#acknowledgements)
  + [Contact](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#contact)
  + [License](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/timeseries.html#license)

#### Previous topic

[How to install R](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/installr.html)

### This Page

* [Show Source](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/_sources/src/timeseries.txt)

### Quick search

Top of Form



Bottom of Form

Top of Form

Bottom of Form

Enter search terms or a module, class or function name.

# Using R for Time Series Analysis

## Time Series Analysis

This booklet itells you how to use the R statistical software to carry out some simple analyses that are common in analysing time series data.

This booklet assumes that the reader has some basic knowledge of time series analysis, and the principal focus of the booklet is not to explain time series analysis, but rather to explain how to carry out these analyses using R.

If you are new to time series analysis, and want to learn more about any of the concepts presented here, I would highly recommend the Open University book “Time series” (product code M249/02), available from from [the Open University Shop](http://www.ouw.co.uk/store/).

In this booklet, I will be using time series data sets that have been kindly made available by Rob Hyndman in his Time Series Data Library at <http://robjhyndman.com/TSDL/>.

There is a pdf version of this booklet available at <https://media.readthedocs.org/pdf/a-little-book-of-r-for-time-series/latest/a-little-book-of-r-for-time-series.pdf>.

If you like this booklet, you may also like to check out my booklet on using R for biomedical statistics, <http://a-little-book-of-r-for-biomedical-statistics.readthedocs.org/>, and my booklet on using R for multivariate analysis, <http://little-book-of-r-for-multivariate-analysis.readthedocs.org/>.

## Reading Time Series Data

The first thing that you will want to do to analyse your time series data will be to read it into R, and to plot the time series. You can read data into R using the scan() function, which assumes that your data for successive time points is in a simple text file with one column.

For example, the file <http://robjhyndman.com/tsdldata/misc/kings.dat> contains data on the age of death of successive kings of England, starting with William the Conqueror (original source: Hipel and Mcleod, 1994).

The data set looks like this:

Age of Death of Successive Kings of England

#starting with William the Conqueror

#Source: McNeill, "Interactive Data Analysis"

60

43

67

50

56

42

50

65

68

43

65

34

...

Only the first few lines of the file have been shown. The first three lines contain some comment on the data, and we want to ignore this when we read the data into R. We can use this by using the “skip” parameter of the scan() function, which specifies how many lines at the top of the file to ignore. To read the file into R, ignoring the first three lines, we type:

> kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)

Read 42 items

> kings

[1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48

[26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56

In this case the age of death of 42 successive kings of England has been read into the variable ‘kings’.

Once you have read the time series data into R, the next step is to store the data in a time series object in R, so that you can use R’s many functions for analysing time series data. To store the data in a time series object, we use the ts() function in R. For example, to store the data in the variable ‘kings’ as a time series object in R, we type:

> kingstimeseries <- ts(kings)

> kingstimeseries

Time Series:

Start = 1

End = 42

Frequency = 1

[1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48

[26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56

Sometimes the time series data set that you have may have been collected at regular intervals that were less than one year, for example, monthly or quarterly. In this case, you can specify the number of times that data was collected per year by using the ‘frequency’ parameter in the ts() function. For monthly time series data, you set frequency=12, while for quarterly time series data, you set frequency=4.

You can also specify the first year that the data was collected, and the first interval in that year by using the ‘start’ parameter in the ts() function. For example, if the first data point corresponds to the second quarter of 1986, you would set start=c(1986,2).

An example is a data set of the number of births per month in New York city, from January 1946 to December 1959 (originally collected by Newton). This data is available in the file <http://robjhyndman.com/tsdldata/data/nybirths.dat> We can read the data into R, and store it as a time series object, by typing:

> births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")

Read 168 items

> birthstimeseries <- ts(births, frequency=12, start=c(1946,1))

> birthstimeseries

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1946 26.663 23.598 26.931 24.740 25.806 24.364 24.477 23.901 23.175 23.227 21.672 21.870

1947 21.439 21.089 23.709 21.669 21.752 20.761 23.479 23.824 23.105 23.110 21.759 22.073

1948 21.937 20.035 23.590 21.672 22.222 22.123 23.950 23.504 22.238 23.142 21.059 21.573

1949 21.548 20.000 22.424 20.615 21.761 22.874 24.104 23.748 23.262 22.907 21.519 22.025

1950 22.604 20.894 24.677 23.673 25.320 23.583 24.671 24.454 24.122 24.252 22.084 22.991

1951 23.287 23.049 25.076 24.037 24.430 24.667 26.451 25.618 25.014 25.110 22.964 23.981

1952 23.798 22.270 24.775 22.646 23.988 24.737 26.276 25.816 25.210 25.199 23.162 24.707

1953 24.364 22.644 25.565 24.062 25.431 24.635 27.009 26.606 26.268 26.462 25.246 25.180

1954 24.657 23.304 26.982 26.199 27.210 26.122 26.706 26.878 26.152 26.379 24.712 25.688

1955 24.990 24.239 26.721 23.475 24.767 26.219 28.361 28.599 27.914 27.784 25.693 26.881

1956 26.217 24.218 27.914 26.975 28.527 27.139 28.982 28.169 28.056 29.136 26.291 26.987

1957 26.589 24.848 27.543 26.896 28.878 27.390 28.065 28.141 29.048 28.484 26.634 27.735

1958 27.132 24.924 28.963 26.589 27.931 28.009 29.229 28.759 28.405 27.945 25.912 26.619

1959 26.076 25.286 27.660 25.951 26.398 25.565 28.865 30.000 29.261 29.012 26.992 27.897

Similarly, the file <http://robjhyndman.com/tsdldata/data/fancy.dat> contains monthly sales for a souvenir shop at a beach resort town in Queensland, Australia, for January 1987-December 1993 (original data from Wheelwright and Hyndman, 1998). We can read the data into R by typing:

> souvenir <- scan("http://robjhyndman.com/tsdldata/data/fancy.dat")

Read 84 items

> souvenirtimeseries <- ts(souvenir, frequency=12, start=c(1987,1))

> souvenirtimeseries

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1987 1664.81 2397.53 2840.71 3547.29 3752.96 3714.74 4349.61 3566.34 5021.82 6423.48 7600.60 19756.21

1988 2499.81 5198.24 7225.14 4806.03 5900.88 4951.34 6179.12 4752.15 5496.43 5835.10 12600.08 28541.72

1989 4717.02 5702.63 9957.58 5304.78 6492.43 6630.80 7349.62 8176.62 8573.17 9690.50 15151.84 34061.01

1990 5921.10 5814.58 12421.25 6369.77 7609.12 7224.75 8121.22 7979.25 8093.06 8476.70 17914.66 30114.41

1991 4826.64 6470.23 9638.77 8821.17 8722.37 10209.48 11276.55 12552.22 11637.39 13606.89 21822.11 45060.69

1992 7615.03 9849.69 14558.40 11587.33 9332.56 13082.09 16732.78 19888.61 23933.38 25391.35 36024.80 80721.71

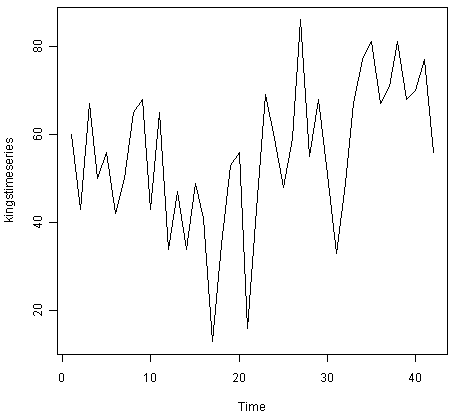
1993 10243.24 11266.88 21826.84 17357.33 15997.79 18601.53 26155.15 28586.52 30505.41 30821.33 46634.38 104660.67

## Plotting Time Series

Once you have read a time series into R, the next step is usually to make a plot of the time series data, which you can do with the plot.ts() function in R.

For example, to plot the time series of the age of death of 42 successive kings of England, we type:

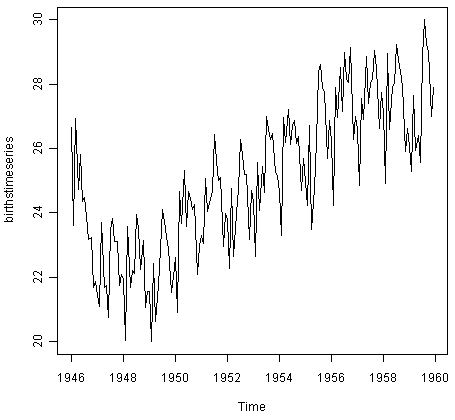
> plot.ts(kingstimeseries)



We can see from the time plot that this time series could probably be described using an additive model, since the random fluctuations in the data are roughly constant in size over time.

Likewise, to plot the time series of the number of births per month in New York city, we type:

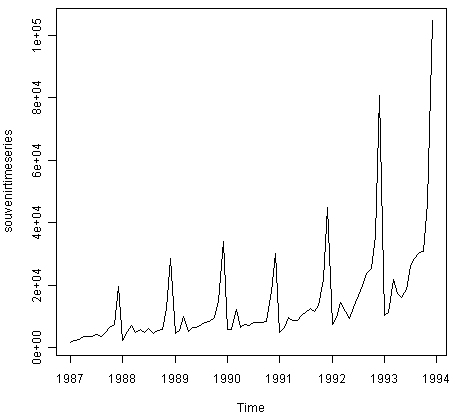
> plot.ts(birthstimeseries)



We can see from this time series that there seems to be seasonal variation in the number of births per month: there is a peak every summer, and a trough every winter. Again, it seems that this time series could probably be described using an additive model, as the seasonal fluctuations are roughly constant in size over time and do not seem to depend on the level of the time series, and the random fluctuations also seem to be roughly constant in size over time.

Similarly, to plot the time series of the monthly sales for the souvenir shop at a beach resort town in Queensland, Australia, we type:

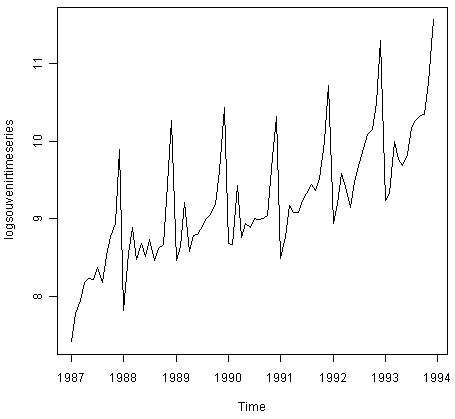
> plot.ts(souvenirtimeseries)



In this case, it appears that an additive model is not appropriate for describing this time series, since the size of the seasonal fluctuations and random fluctuations seem to increase with the level of the time series. Thus, we may need to transform the time series in order to get a transformed time series that can be described using an additive model. For example, we can transform the time series by calculating the natural log of the original data:

> logsouvenirtimeseries <- log(souvenirtimeseries)

> plot.ts(logsouvenirtimeseries)



Here we can see that the size of the seasonal fluctuations and random fluctuations in the log-transformed time series seem to be roughly constant over time, and do not depend on the level of the time series. Thus, the log-transformed time series can probably be described using an additive model.

## Decomposing Time Series

Decomposing a time series means separating it into its constituent components, which are usually a trend component and an irregular component, and if it is a seasonal time series, a seasonal component.

### Decomposing Non-Seasonal Data

A non-seasonal time series consists of a trend component and an irregular component. Decomposing the time series involves trying to separate the time series into these components, that is, estimating the the trend component and the irregular component.

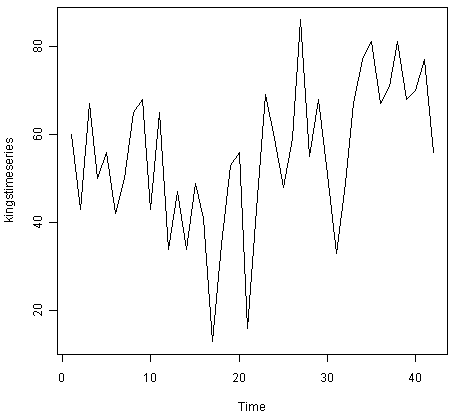
To estimate the trend component of a non-seasonal time series that can be described using an additive model, it is common to use a smoothing method, such as calculating the simple moving average of the time series.

The SMA() function in the “TTR” R package can be used to smooth time series data using a simple moving average. To use this function, we first need to install the “TTR” R package (for instructions on how to install an R package, see [How to install an R package](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/installr.html#how-to-install-an-r-package)). Once you have installed the “TTR” R package, you can load the “TTR” R package by typing:

> library("TTR")

You can then use the “SMA()” function to smooth time series data. To use the SMA() function, you need to specify the order (span) of the simple moving average, using the parameter “n”. For example, to calculate a simple moving average of order 5, we set n=5 in the SMA() function.

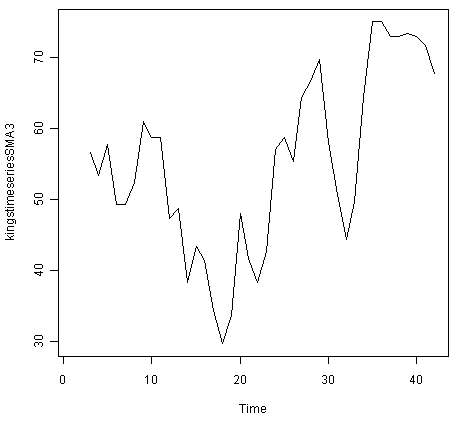
For example, as discussed above, the time series of the age of death of 42 successive kings of England appears is non-seasonal, and can probably be described using an additive model, since the random fluctuations in the data are roughly constant in size over time:



Thus, we can try to estimate the trend component of this time series by smoothing using a simple moving average. To smooth the time series using a simple moving average of order 3, and plot the smoothed time series data, we type:

> kingstimeseriesSMA3 <- SMA(kingstimeseries,n=3)

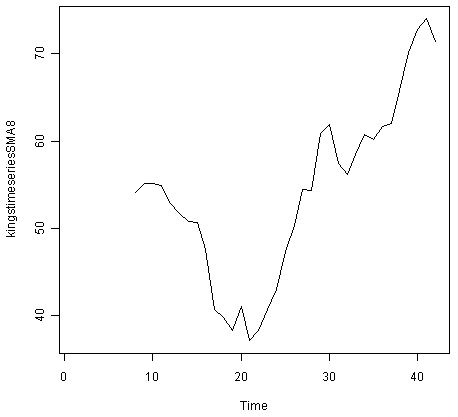
> plot.ts(kingstimeseriesSMA3)



There still appears to be quite a lot of random fluctuations in the time series smoothed using a simple moving average of order 3. Thus, to estimate the trend component more accurately, we might want to try smoothing the data with a simple moving average of a higher order. This takes a little bit of trial-and-error, to find the right amount of smoothing. For example, we can try using a simple moving average of order 8:

> kingstimeseriesSMA8 <- SMA(kingstimeseries,n=8)

> plot.ts(kingstimeseriesSMA8)



The data smoothed with a simple moving average of order 8 gives a clearer picture of the trend component, and we can see that the age of death of the English kings seems to have decreased from about 55 years old to about 38 years old during the reign of the first 20 kings, and then increased after that to about 73 years old by the end of the reign of the 40th king in the time series.

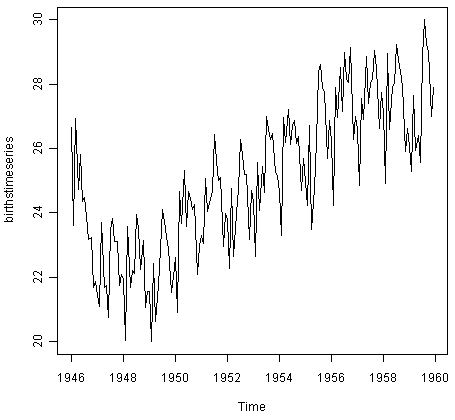
### Decomposing Seasonal Data

A seasonal time series consists of a trend component, a seasonal component and an irregular component. Decomposing the time series means separating the time series into these three components: that is, estimating these three components.

To estimate the trend component and seasonal component of a seasonal time series that can be described using an additive model, we can use the “decompose()” function in R. This function estimates the trend, seasonal, and irregular components of a time series that can be described using an additive model.

The function “decompose()” returns a list object as its result, where the estimates of the seasonal component, trend component and irregular component are stored in named elements of that list objects, called “seasonal”, “trend”, and “random” respectively.

For example, as discussed above, the time series of the number of births per month in New York city is seasonal with a peak every summer and trough every winter, and can probably be described using an additive model since the seasonal and random fluctuations seem to be roughly constant in size over time:



To estimate the trend, seasonal and irregular components of this time series, we type:

> birthstimeseriescomponents <- decompose(birthstimeseries)

The estimated values of the seasonal, trend and irregular components are now stored in variables birthstimeseriescomponents$seasonal, birthstimeseriescomponents$trend and birthstimeseriescomponents$random. For example, we can print out the estimated values of the seasonal component by typing:

> birthstimeseriescomponents$seasonal # get the estimated values of the seasonal component

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1946 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1947 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1948 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1949 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1950 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1951 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1952 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1953 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1954 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1955 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1956 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1957 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

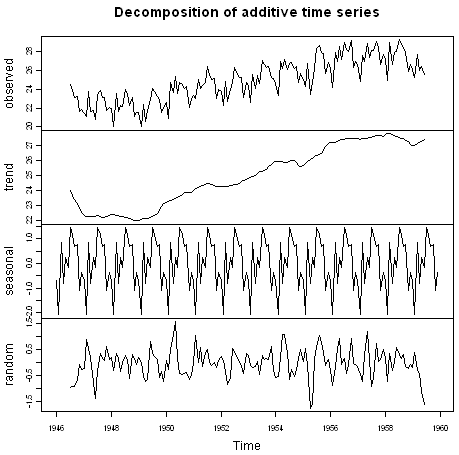
1958 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

1959 -0.6771947 -2.0829607 0.8625232 -0.8016787 0.2516514 -0.1532556 1.4560457 1.1645938 0.6916162 0.7752444 -1.1097652 -0.3768197

The estimated seasonal factors are given for the months January-December, and are the same for each year. The largest seasonal factor is for July (about 1.46), and the lowest is for February (about -2.08), indicating that there seems to be a peak in births in July and a trough in births in February each year.

We can plot the estimated trend, seasonal, and irregular components of the time series by using the “plot()” function, for example:

> plot(birthstimeseriescomponents)



The plot above shows the original time series (top), the estimated trend component (second from top), the estimated seasonal component (third from top), and the estimated irregular component (bottom). We see that the estimated trend component shows a small decrease from about 24 in 1947 to about 22 in 1948, followed by a steady increase from then on to about 27 in 1959.

### Seasonally Adjusting

If you have a seasonal time series that can be described using an additive model, you can seasonally adjust the time series by estimating the seasonal component, and subtracting the estimated seasonal component from the original time series. We can do this using the estimate of the seasonal component calculated by the “decompose()” function.

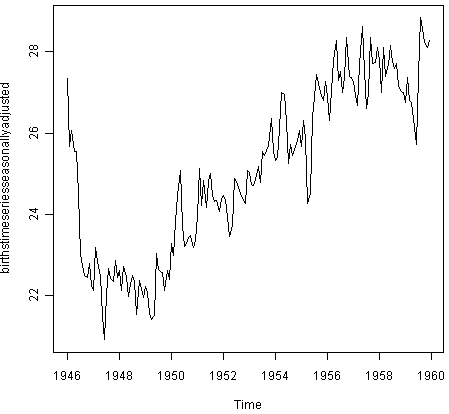
For example, to seasonally adjust the time series of the number of births per month in New York city, we can estimate the seasonal component using “decompose()”, and then subtract the seasonal component from the original time series:

> birthstimeseriescomponents <- decompose(birthstimeseries)

> birthstimeseriesseasonallyadjusted <- birthstimeseries - birthstimeseriescomponents$seasonal

We can then plot the seasonally adjusted time series using the “plot()” function, by typing:

> plot(birthstimeseriesseasonallyadjusted)



You can see that the seasonal variation has been removed from the seasonally adjusted time series. The seasonally adjusted time series now just contains the trend component and an irregular component.

## Forecasts using Exponential Smoothing

Exponential smoothing can be used to make short-term forecasts for time series data.

### Simple Exponential Smoothing

If you have a time series that can be described using an additive model with constant level and no seasonality, you can use simple exponential smoothing to make short-term forecasts.

The simple exponential smoothing method provides a way of estimating the level at the current time point. Smoothing is controlled by the parameter alpha; for the estimate of the level at the current time point. The value of alpha; lies between 0 and 1. Values of alpha that are close to 0 mean that little weight is placed on the most recent observations when making forecasts of future values.

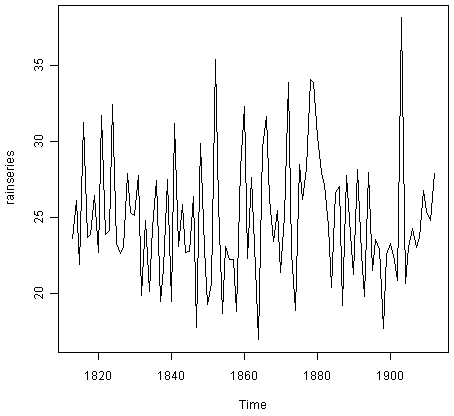
For example, the file <http://robjhyndman.com/tsdldata/hurst/precip1.dat> contains total annual rainfall in inches for London, from 1813-1912 (original data from Hipel and McLeod, 1994). We can read the data into R and plot it by typing:

> rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)

Read 100 items

> rainseries <- ts(rain,start=c(1813))

> plot.ts(rainseries)



You can see from the plot that there is roughly constant level (the mean stays constant at about 25 inches). The random fluctuations in the time series seem to be roughly constant in size over time, so it is probably appropriate to describe the data using an additive model. Thus, we can make forecasts using simple exponential smoothing.

To make forecasts using simple exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the “HoltWinters()” function in R. To use HoltWinters() for simple exponential smoothing, we need to set the parameters beta=FALSE and gamma=FALSE in the HoltWinters() function (the beta and gamma parameters are used for Holt’s exponential smoothing, or Holt-Winters exponential smoothing, as described below).

The HoltWinters() function returns a list variable, that contains several named elements.

For example, to use simple exponential smoothing to make forecasts for the time series of annual rainfall in London, we type:

> rainseriesforecasts <- HoltWinters(rainseries, beta=FALSE, gamma=FALSE)

> rainseriesforecasts

Smoothing parameters:

alpha: 0.02412151

beta : FALSE

gamma: FALSE

Coefficients:

[,1]

a 24.67819

The output of HoltWinters() tells us that the estimated value of the alpha parameter is about 0.024. This is very close to zero, telling us that the forecasts are based on both recent and less recent observations (although somewhat more weight is placed on recent observations).

By default, HoltWinters() just makes forecasts for the same time period covered by our original time series. In this case, our original time series included rainfall for London from 1813-1912, so the forecasts are also for 1813-1912.

In the example above, we have stored the output of the HoltWinters() function in the list variable “rainseriesforecasts”. The forecasts made by HoltWinters() are stored in a named element of this list variable called “fitted”, so we can get their values by typing:

> rainseriesforecasts$fitted

Time Series:

Start = 1814

End = 1912

Frequency = 1

xhat level

1814 23.56000 23.56000

1815 23.62054 23.62054

1816 23.57808 23.57808

1817 23.76290 23.76290

1818 23.76017 23.76017

1819 23.76306 23.76306

1820 23.82691 23.82691

...

1905 24.62852 24.62852

1906 24.58852 24.58852

1907 24.58059 24.58059

1908 24.54271 24.54271

1909 24.52166 24.52166

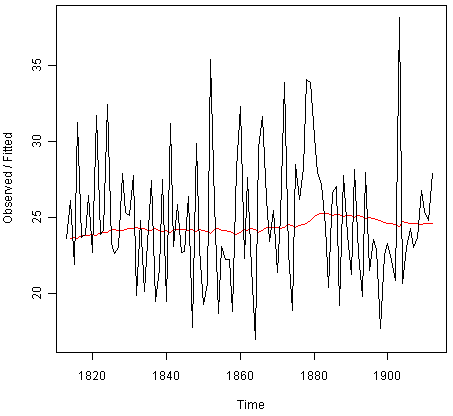
1910 24.57541 24.57541

1911 24.59433 24.59433

1912 24.59905 24.59905

We can plot the original time series against the forecasts by typing:

> plot(rainseriesforecasts)



The plot shows the original time series in black, and the forecasts as a red line. The time series of forecasts is much smoother than the time series of the original data here.

As a measure of the accuracy of the forecasts, we can calculate the sum of squared errors for the in-sample forecast errors, that is, the forecast errors for the time period covered by our original time series. The sum-of-squared-errors is stored in a named element of the list variable “rainseriesforecasts” called “SSE”, so we can get its value by typing:

> rainseriesforecasts$SSE

[1] 1828.855

That is, here the sum-of-squared-errors is 1828.855.

It is common in simple exponential smoothing to use the first value in the time series as the initial value for the level. For example, in the time series for rainfall in London, the first value is 23.56 (inches) for rainfall in 1813. You can specify the initial value for the level in the HoltWinters() function by using the “l.start” parameter. For example, to make forecasts with the initial value of the level set to 23.56, we type:

> HoltWinters(rainseries, beta=FALSE, gamma=FALSE, l.start=23.56)

As explained above, by default HoltWinters() just makes forecasts for the time period covered by the original data, which is 1813-1912 for the rainfall time series. We can make forecasts for further time points by using the “forecast.HoltWinters()” function in the R “forecast” package. To use the forecast.HoltWinters() function, we first need to install the “forecast” R package (for instructions on how to install an R package, see [How to install an R package](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/installr.html#how-to-install-an-r-package)).

Once you have installed the “forecast” R package, you can load the “forecast” R package by typing:

> library("forecast")

When using the forecast.HoltWinters() function, as its first argument (input), you pass it the predictive model that you have already fitted using the HoltWinters() function. For example, in the case of the rainfall time series, we stored the predictive model made using HoltWinters() in the variable “rainseriesforecasts”. You specify how many further time points you want to make forecasts for by using the “h” parameter in forecast.HoltWinters(). For example, to make a forecast of rainfall for the years 1814-1820 (8 more years) using forecast.HoltWinters(), we type:

> rainseriesforecasts2 <- forecast.HoltWinters(rainseriesforecasts, h=8)

> rainseriesforecasts2

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

1913 24.67819 19.17493 30.18145 16.26169 33.09470

1914 24.67819 19.17333 30.18305 16.25924 33.09715

1915 24.67819 19.17173 30.18465 16.25679 33.09960

1916 24.67819 19.17013 30.18625 16.25434 33.10204

1917 24.67819 19.16853 30.18785 16.25190 33.10449

1918 24.67819 19.16694 30.18945 16.24945 33.10694

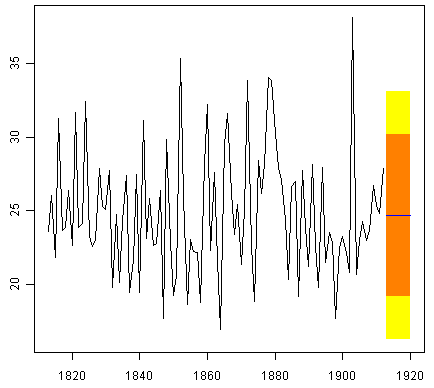
1919 24.67819 19.16534 30.19105 16.24701 33.10938

1920 24.67819 19.16374 30.19265 16.24456 33.11182

The forecast.HoltWinters() function gives you the forecast for a year, a 80% prediction interval for the forecast, and a 95% prediction interval for the forecast. For example, the forecasted rainfall for 1920 is about 24.68 inches, with a 95% prediction interval of (16.24, 33.11).

To plot the predictions made by forecast.HoltWinters(), we can use the “plot.forecast()” function:

> plot.forecast(rainseriesforecasts2)



Here the forecasts for 1913-1920 are plotted as a blue line, the 80% prediction interval as an orange shaded area, and the 95% prediction interval as a yellow shaded area.

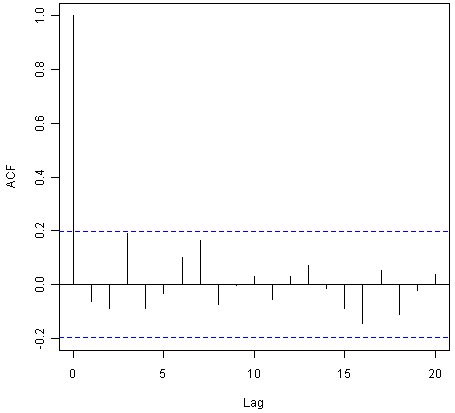
The ‘forecast errors’ are calculated as the observed values minus predicted values, for each time point. We can only calculate the forecast errors for the time period covered by our original time series, which is 1813-1912 for the rainfall data. As mentioned above, one measure of the accuracy of the predictive model is the sum-of-squared-errors (SSE) for the in-sample forecast errors.

The in-sample forecast errors are stored in the named element “residuals” of the list variable returned by forecast.HoltWinters(). If the predictive model cannot be improved upon, there should be no correlations between forecast errors for successive predictions. In other words, if there are correlations between forecast errors for successive predictions, it is likely that the simple exponential smoothing forecasts could be improved upon by another forecasting technique.

To figure out whether this is the case, we can obtain a correlogram of the in-sample forecast errors for lags 1-20. We can calculate a correlogram of the forecast errors using the “acf()” function in R. To specify the maximum lag that we want to look at, we use the “lag.max” parameter in acf().

For example, to calculate a correlogram of the in-sample forecast errors for the London rainfall data for lags 1-20, we type:

> acf(rainseriesforecasts2$residuals, lag.max=20)



You can see from the sample correlogram that the autocorrelation at lag 3 is just touching the significance bounds. To test whether there is significant evidence for non-zero correlations at lags 1-20, we can carry out a Ljung-Box test. This can be done in R using the “Box.test()”, function. The maximum lag that we want to look at is specified using the “lag” parameter in the Box.test() function. For example, to test whether there are non-zero autocorrelations at lags 1-20, for the in-sample forecast errors for London rainfall data, we type:

> Box.test(rainseriesforecasts2$residuals, lag=20, type="Ljung-Box")

Box-Ljung test

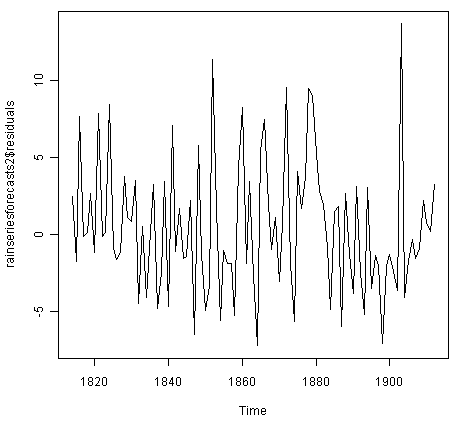
data: rainseriesforecasts2$residuals

X-squared = 17.4008, df = 20, p-value = 0.6268

Here the Ljung-Box test statistic is 17.4, and the p-value is 0.6, so there is little evidence of non-zero autocorrelations in the in-sample forecast errors at lags 1-20.

To be sure that the predictive model cannot be improved upon, it is also a good idea to check whether the forecast errors are normally distributed with mean zero and constant variance. To check whether the forecast errors have constant variance, we can make a time plot of the in-sample forecast errors:

> plot.ts(rainseriesforecasts2$residuals)



The plot shows that the in-sample forecast errors seem to have roughly constant variance over time, although the size of the fluctuations in the start of the time series (1820-1830) may be slightly less than that at later dates (eg. 1840-1850).

To check whether the forecast errors are normally distributed with mean zero, we can plot a histogram of the forecast errors, with an overlaid normal curve that has mean zero and the same standard deviation as the distribution of forecast errors. To do this, we can define an R function “plotForecastErrors()”, below:

> plotForecastErrors <- function(forecasterrors)

{

# make a histogram of the forecast errors:

mybinsize <- IQR(forecasterrors)/4

mysd <- sd(forecasterrors)

mymin <- min(forecasterrors) - mysd\*5

mymax <- max(forecasterrors) + mysd\*3

# generate normally distributed data with mean 0 and standard deviation mysd

mynorm <- rnorm(10000, mean=0, sd=mysd)

mymin2 <- min(mynorm)

mymax2 <- max(mynorm)

if (mymin2 < mymin) { mymin <- mymin2 }

if (mymax2 > mymax) { mymax <- mymax2 }

# make a red histogram of the forecast errors, with the normally distributed data overlaid:

mybins <- seq(mymin, mymax, mybinsize)

hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)

# freq=FALSE ensures the area under the histogram = 1

# generate normally distributed data with mean 0 and standard deviation mysd

myhist <- hist(mynorm, plot=FALSE, breaks=mybins)

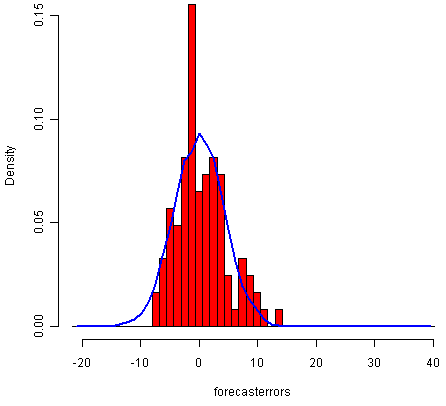
# plot the normal curve as a blue line on top of the histogram of forecast errors:

points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)

}

You will have to copy the function above into R in order to use it. You can then use plotForecastErrors() to plot a histogram (with overlaid normal curve) of the forecast errors for the rainfall predictions:

> plotForecastErrors(rainseriesforecasts2$residuals)



The plot shows that the distribution of forecast errors is roughly centred on zero, and is more or less normally distributed, although it seems to be slightly skewed to the right compared to a normal curve. However, the right skew is relatively small, and so it is plausible that the forecast errors are normally distributed with mean zero.

The Ljung-Box test showed that there is little evidence of non-zero autocorrelations in the in-sample forecast errors, and the distribution of forecast errors seems to be normally distributed with mean zero. This suggests that the simple exponential smoothing method provides an adequate predictive model for London rainfall, which probably cannot be improved upon. Furthermore, the assumptions that the 80% and 95% predictions intervals were based upon (that there are no autocorrelations in the forecast errors, and the forecast errors are normally distributed with mean zero and constant variance) are probably valid.

### Holt’s Exponential Smoothing

If you have a time series that can be described using an additive model with increasing or decreasing trend and no seasonality, you can use Holt’s exponential smoothing to make short-term forecasts.

Holt’s exponential smoothing estimates the level and slope at the current time point. Smoothing is controlled by two parameters, alpha, for the estimate of the level at the current time point, and beta for the estimate of the slope b of the trend component at the current time point. As with simple exponential smoothing, the paramters alpha and beta have values between 0 and 1, and values that are close to 0 mean that little weight is placed on the most recent observations when making forecasts of future values.

An example of a time series that can probably be described using an additive model with a trend and no seasonality is the time series of the annual diameter of women’s skirts at the hem, from 1866 to 1911. The data is available in the file <http://robjhyndman.com/tsdldata/roberts/skirts.dat> (original data from Hipel and McLeod, 1994).

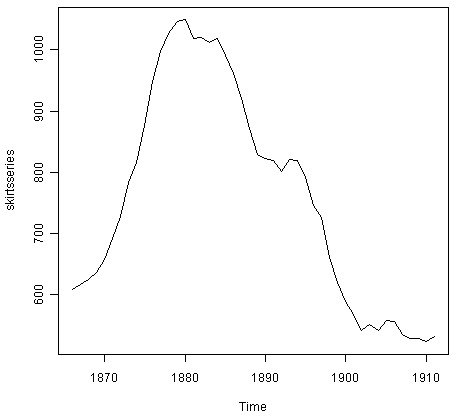
We can read in and plot the data in R by typing:

> skirts <- scan("http://robjhyndman.com/tsdldata/roberts/skirts.dat",skip=5)

Read 46 items

> skirtsseries <- ts(skirts,start=c(1866))

> plot.ts(skirtsseries)



We can see from the plot that there was an increase in hem diameter from about 600 in 1866 to about 1050 in 1880, and that afterwards the hem diameter decreased to about 520 in 1911.

To make forecasts, we can fit a predictive model using the HoltWinters() function in R. To use HoltWinters() for Holt’s exponential smoothing, we need to set the parameter gamma=FALSE (the gamma parameter is used for Holt-Winters exponential smoothing, as described below).

For example, to use Holt’s exponential smoothing to fit a predictive model for skirt hem diameter, we type:

> skirtsseriesforecasts <- HoltWinters(skirtsseries, gamma=FALSE)

> skirtsseriesforecasts

Smoothing parameters:

alpha: 0.8383481

beta : 1

gamma: FALSE

Coefficients:

[,1]

a 529.308585

b 5.690464

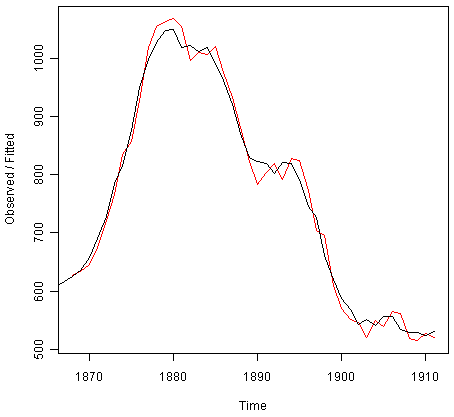
> skirtsseriesforecasts$SSE

[1] 16954.18

The estimated value of alpha is 0.84, and of beta is 1.00. These are both high, telling us that both the estimate of the current value of the level, and of the slope b of the trend component, are based mostly upon very recent observations in the time series. This makes good intuitive sense, since the level and the slope of the time series both change quite a lot over time. The value of the sum-of-squared-errors for the in-sample forecast errors is 16954.

We can plot the original time series as a black line, with the forecasted values as a red line on top of that, by typing:

> plot(skirtsseriesforecasts)



We can see from the picture that the in-sample forecasts agree pretty well with the observed values, although they tend to lag behind the observed values a little bit.

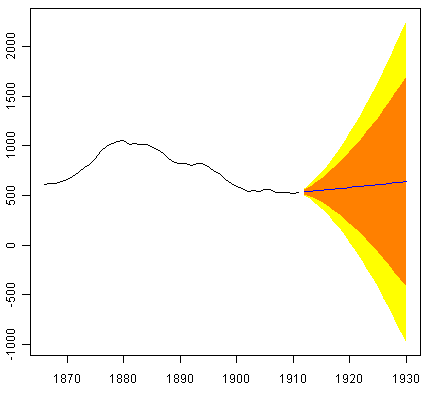
If you wish, you can specify the initial values of the level and the slope b of the trend component by using the “l.start” and “b.start” arguments for the HoltWinters() function. It is common to set the initial value of the level to the first value in the time series (608 for the skirts data), and the initial value of the slope to the second value minus the first value (9 for the skirts data). For example, to fit a predictive model to the skirt hem data using Holt’s exponential smoothing, with initial values of 608 for the level and 9 for the slope b of the trend component, we type:

> HoltWinters(skirtsseries, gamma=FALSE, l.start=608, b.start=9)

As for simple exponential smoothing, we can make forecasts for future times not covered by the original time series by using the forecast.HoltWinters() function in the “forecast” package. For example, our time series data for skirt hems was for 1866 to 1911, so we can make predictions for 1912 to 1930 (19 more data points), and plot them, by typing:

> skirtsseriesforecasts2 <- forecast.HoltWinters(skirtsseriesforecasts, h=19)

> plot.forecast(skirtsseriesforecasts2)



The forecasts are shown as a blue line, with the 80% prediction intervals as an orange shaded area, and the 95% prediction intervals as a yellow shaded area.

As for simple exponential smoothing, we can check whether the predictive model could be improved upon by checking whether the in-sample forecast errors show non-zero autocorrelations at lags 1-20. For example, for the skirt hem data, we can make a correlogram, and carry out the Ljung-Box test, by typing:

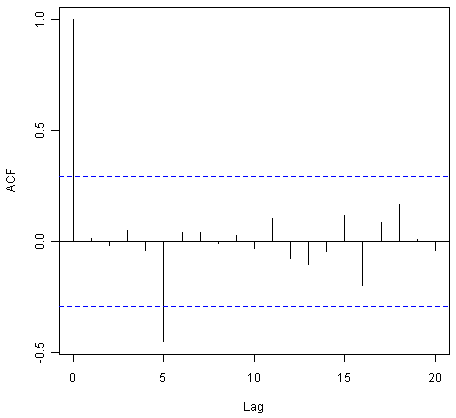
> acf(skirtsseriesforecasts2$residuals, lag.max=20)

> Box.test(skirtsseriesforecasts2$residuals, lag=20, type="Ljung-Box")

Box-Ljung test

data: skirtsseriesforecasts2$residuals

X-squared = 19.7312, df = 20, p-value = 0.4749

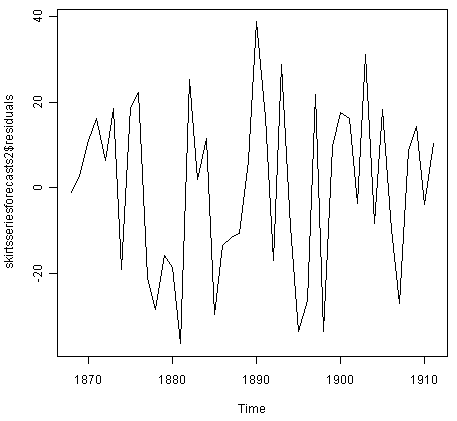


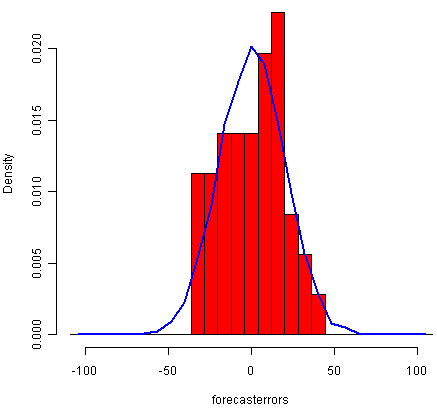
Here the correlogram shows that the sample autocorrelation for the in-sample forecast errors at lag 5 exceeds the significance bounds. However, we would expect one in 20 of the autocorrelations for the first twenty lags to exceed the 95% significance bounds by chance alone. Indeed, when we carry out the Ljung-Box test, the p-value is 0.47, indicating that there is little evidence of non-zero autocorrelations in the in-sample forecast errors at lags 1-20.

As for simple exponential smoothing, we should also check that the forecast errors have constant variance over time, and are normally distributed with mean zero. We can do this by making a time plot of forecast errors, and a histogram of the distribution of forecast errors with an overlaid normal curve:

> plot.ts(skirtsseriesforecasts2$residuals) # make a time plot

> plotForecastErrors(skirtsseriesforecasts2$residuals) # make a histogram





The time plot of forecast errors shows that the forecast errors have roughly constant variance over time. The histogram of forecast errors show that it is plausible that the forecast errors are normally distributed with mean zero and constant variance.

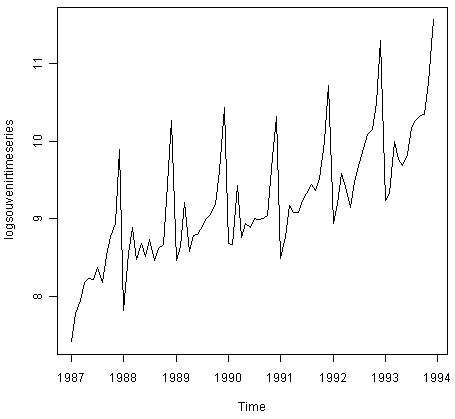
Thus, the Ljung-Box test shows that there is little evidence of autocorrelations in the forecast errors, while the time plot and histogram of forecast errors show that it is plausible that the forecast errors are normally distributed with mean zero and constant variance. Therefore, we can conclude that Holt’s exponential smoothing provides an adequate predictive model for skirt hem diameters, which probably cannot be improved upon. In addition, it means that the assumptions that the 80% and 95% predictions intervals were based upon are probably valid.

### Holt-Winters Exponential Smoothing

If you have a time series that can be described using an additive model with increasing or decreasing trend and seasonality, you can use Holt-Winters exponential smoothing to make short-term forecasts.

Holt-Winters exponential smoothing estimates the level, slope and seasonal component at the current time point. Smoothing is controlled by three parameters: alpha, beta, and gamma, for the estimates of the level, slope b of the trend component, and the seasonal component, respectively, at the current time point. The parameters alpha, beta and gamma all have values between 0 and 1, and values that are close to 0 mean that relatively little weight is placed on the most recent observations when making forecasts of future values.

An example of a time series that can probably be described using an additive model with a trend and seasonality is the time series of the log of monthly sales for the souvenir shop at a beach resort town in Queensland, Australia (discussed above):



To make forecasts, we can fit a predictive model using the HoltWinters() function. For example, to fit a predictive model for the log of the monthly sales in the souvenir shop, we type:

> logsouvenirtimeseries <- log(souvenirtimeseries)

> souvenirtimeseriesforecasts <- HoltWinters(logsouvenirtimeseries)

> souvenirtimeseriesforecasts

Holt-Winters exponential smoothing with trend and additive seasonal component.

Smoothing parameters:

alpha: 0.413418

beta : 0

gamma: 0.9561275

Coefficients:

[,1]

a 10.37661961

b 0.02996319

s1 -0.80952063

s2 -0.60576477

s3 0.01103238

s4 -0.24160551

s5 -0.35933517

s6 -0.18076683

s7 0.07788605

s8 0.10147055

s9 0.09649353

s10 0.05197826

s11 0.41793637

s12 1.18088423

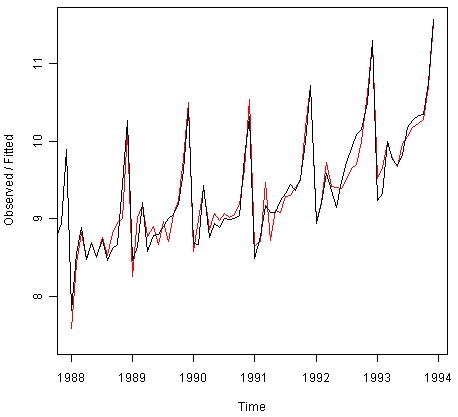
> souvenirtimeseriesforecasts$SSE

2.011491

The estimated values of alpha, beta and gamma are 0.41, 0.00, and 0.96, respectively. The value of alpha (0.41) is relatively low, indicating that the estimate of the level at the current time point is based upon both recent observations and some observations in the more distant past. The value of beta is 0.00, indicating that the estimate of the slope b of the trend component is not updated over the time series, and instead is set equal to its initial value. This makes good intuitive sense, as the level changes quite a bit over the time series, but the slope b of the trend component remains roughly the same. In contrast, the value of gamma (0.96) is high, indicating that the estimate of the seasonal component at the current time point is just based upon very recent observations.

As for simple exponential smoothing and Holt’s exponential smoothing, we can plot the original time series as a black line, with the forecasted values as a red line on top of that:

> plot(souvenirtimeseriesforecasts)

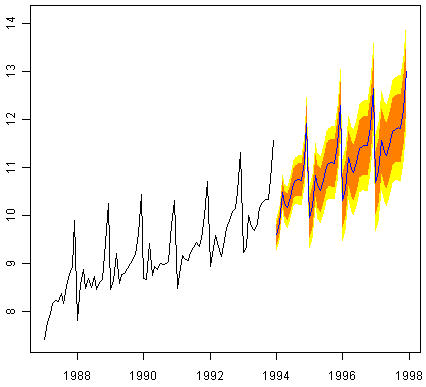


We see from the plot that the Holt-Winters exponential method is very successful in predicting the seasonal peaks, which occur roughly in November every year.

To make forecasts for future times not included in the original time series, we use the “forecast.HoltWinters()” function in the “forecast” package. For example, the original data for the souvenir sales is from January 1987 to December 1993. If we wanted to make forecasts for January 1994 to December 1998 (48 more months), and plot the forecasts, we would type:

> souvenirtimeseriesforecasts2 <- forecast.HoltWinters(souvenirtimeseriesforecasts, h=48)

> plot.forecast(souvenirtimeseriesforecasts2)



The forecasts are shown as a blue line, and the orange and yellow shaded areas show 80% and 95% prediction intervals, respectively.

We can investigate whether the predictive model can be improved upon by checking whether the in-sample forecast errors show non-zero autocorrelations at lags 1-20, by making a correlogram and carrying out the Ljung-Box test:

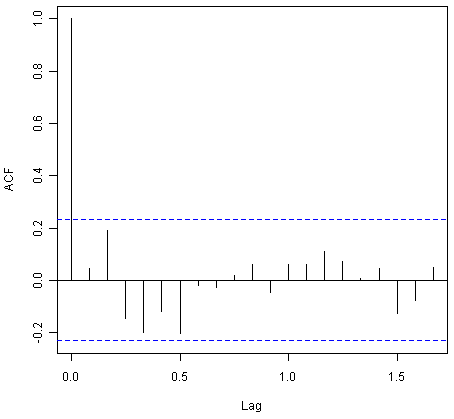
> acf(souvenirtimeseriesforecasts2$residuals, lag.max=20)

> Box.test(souvenirtimeseriesforecasts2$residuals, lag=20, type="Ljung-Box")

Box-Ljung test

data: souvenirtimeseriesforecasts2$residuals

X-squared = 17.5304, df = 20, p-value = 0.6183

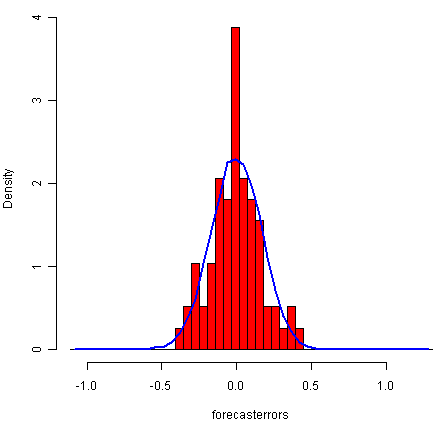
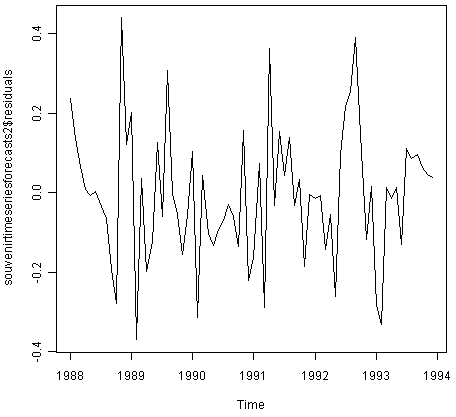


The correlogram shows that the autocorrelations for the in-sample forecast errors do not exceed the significance bounds for lags 1-20. Furthermore, the p-value for Ljung-Box test is 0.6, indicating that there is little evidence of non-zero autocorrelations at lags 1-20.

We can check whether the forecast errors have constant variance over time, and are normally distributed with mean zero, by making a time plot of the forecast errors and a histogram (with overlaid normal curve):

> plot.ts(souvenirtimeseriesforecasts2$residuals) # make a time plot

> plotForecastErrors(souvenirtimeseriesforecasts2$residuals) # make a histogram



From the time plot, it appears plausible that the forecast errors have constant variance over time. From the histogram of forecast errors, it seems plausible that the forecast errors are normally distributed with mean zero.

Thus,there is little evidence of autocorrelation at lags 1-20 for the forecast errors, and the forecast errors appear to be normally distributed with mean zero and constant variance over time. This suggests that Holt-Winters exponential smoothing provides an adequate predictive model of the log of sales at the souvenir shop, which probably cannot be improved upon. Furthermore, the assumptions upon which the prediction intervals were based are probably valid.

## ARIMA Models

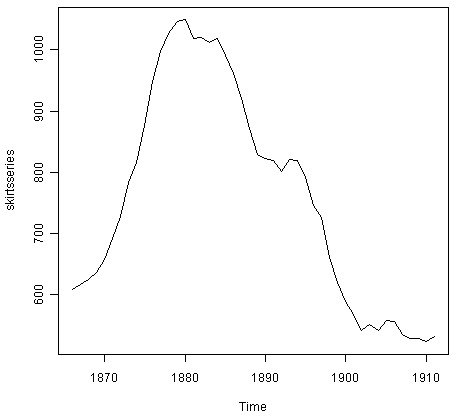
Exponential smoothing methods are useful for making forecasts, and make no assumptions about the correlations between successive values of the time series. However, if you want to make prediction intervals for forecasts made using exponential smoothing methods, the prediction intervals require that the forecast errors are uncorrelated and are normally distributed with mean zero and constant variance.

While exponential smoothing methods do not make any assumptions about correlations between successive values of the time series, in some cases you can make a better predictive model by taking correlations in the data into account. Autoregressive Integrated Moving Average (ARIMA) models include an explicit statistical model for the irregular component of a time series, that allows for non-zero autocorrelations in the irregular component.

### Differencing a Time Series

ARIMA models are defined for stationary time series. Therefore, if you start off with a non-stationary time series, you will first need to ‘difference’ the time series until you obtain a stationary time series. If you have to difference the time series d times to obtain a stationary series, then you have an ARIMA(p,d,q) model, where d is the order of differencing used.

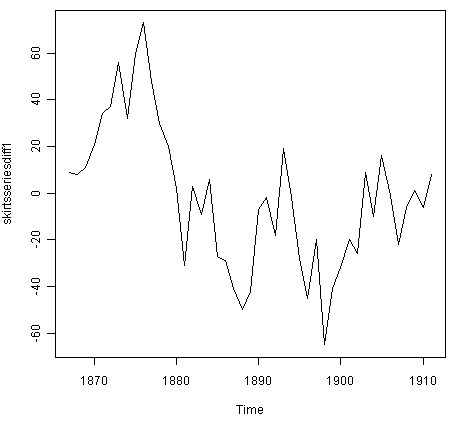
You can difference a time series using the “diff()” function in R. For example, the time series of the annual diameter of women’s skirts at the hem, from 1866 to 1911 is not stationary in mean, as the level changes a lot over time:



We can difference the time series (which we stored in “skirtsseries”, see above) once, and plot the differenced series, by typing:

> skirtsseriesdiff1 <- diff(skirtsseries, differences=1)

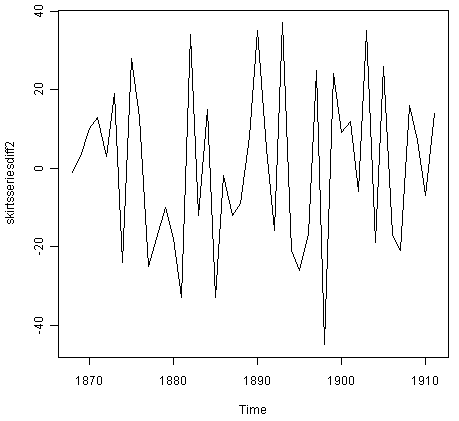
> plot.ts(skirtsseriesdiff1)



The resulting time series of first differences (above) does not appear to be stationary in mean. Therefore, we can difference the time series twice, to see if that gives us a stationary time series:

> skirtsseriesdiff2 <- diff(skirtsseries, differences=2)

> plot.ts(skirtsseriesdiff2)



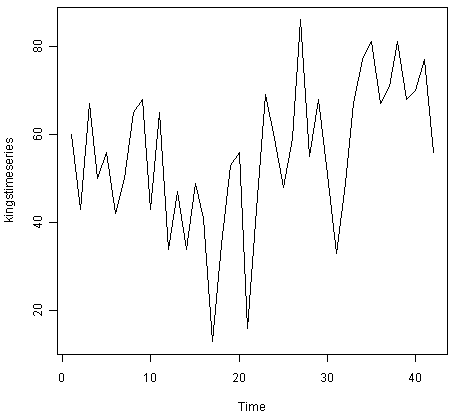
Formal tests for stationarity

Formal tests for stationarity called “unit root tests” are available in the fUnitRoots package, available on CRAN, but will not be discussed here.

The time series of second differences (above) does appear to be stationary in mean and variance, as the level of the series stays roughly constant over time, and the variance of the series appears roughly constant over time. Thus, it appears that we need to difference the time series of the diameter of skirts twice in order to achieve a stationary series.

If you need to difference your original time series data d times in order to obtain a stationary time series, this means that you can use an ARIMA(p,d,q) model for your time series, where d is the order of differencing used. For example, for the time series of the diameter of women’s skirts, we had to difference the time series twice, and so the order of differencing (d) is 2. This means that you can use an ARIMA(p,2,q) model for your time series. The next step is to figure out the values of p and q for the ARIMA model.

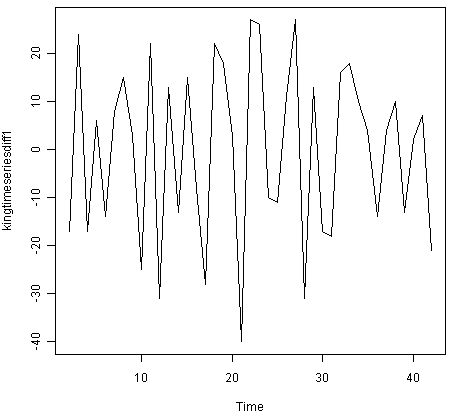
Another example is the time series of the age of death of the successive kings of England (see above):



From the time plot (above), we can see that the time series is not stationary in mean. To calculate the time series of first differences, and plot it, we type:

> kingtimeseriesdiff1 <- diff(kingstimeseries, differences=1)

> plot.ts(kingtimeseriesdiff1)



The time series of first differences appears to be stationary in mean and variance, and so an ARIMA(p,1,q) model is probably appropriate for the time series of the age of death of the kings of England. By taking the time series of first differences, we have removed the trend component of the time series of the ages at death of the kings, and are left with an irregular component. We can now examine whether there are correlations between successive terms of this irregular component; if so, this could help us to make a predictive model for the ages at death of the kings.

### Selecting a Candidate ARIMA Model

If your time series is stationary, or if you have transformed it to a stationary time series by differencing d times, the next step is to select the appropriate ARIMA model, which means finding the values of most appropriate values of p and q for an ARIMA(p,d,q) model. To do this, you usually need to examine the correlogram and partial correlogram of the stationary time series.

To plot a correlogram and partial correlogram, we can use the “acf()” and “pacf()” functions in R, respectively. To get the actual values of the autocorrelations and partial autocorrelations, we set “plot=FALSE” in the “acf()” and “pacf()” functions.

#### Example of the Ages at Death of the Kings of England

For example, to plot the correlogram for lags 1-20 of the once differenced time series of the ages at death of the kings of England, and to get the values of the autocorrelations, we type:

> acf(kingtimeseriesdiff1, lag.max=20) # plot a correlogram

> acf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # get the autocorrelation values

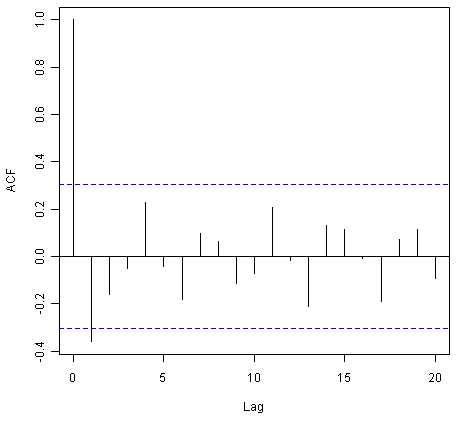
Autocorrelations of series 'kingtimeseriesdiff1', by lag

0 1 2 3 4 5 6 7 8 9 10

1.000 -0.360 -0.162 -0.050 0.227 -0.042 -0.181 0.095 0.064 -0.116 -0.071

11 12 13 14 15 16 17 18 19 20

0.206 -0.017 -0.212 0.130 0.114 -0.009 -0.192 0.072 0.113 -0.093



We see from the correlogram that the autocorrelation at lag 1 (-0.360) exceeds the significance bounds, but all other autocorrelations between lags 1-20 do not exceed the significance bounds.

To plot the partial correlogram for lags 1-20 for the once differenced time series of the ages at death of the English kings, and get the values of the partial autocorrelations, we use the “pacf()” function, by typing:

> pacf(kingtimeseriesdiff1, lag.max=20) # plot a partial correlogram

> pacf(kingtimeseriesdiff1, lag.max=20, plot=FALSE) # get the partial autocorrelation values

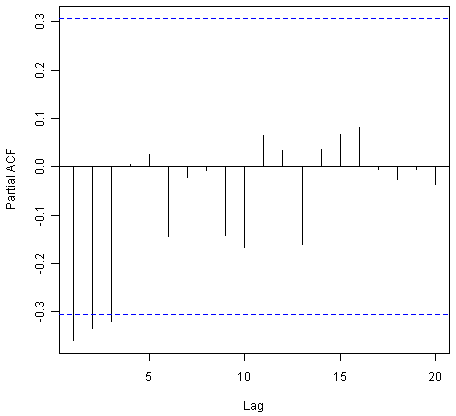
Partial autocorrelations of series 'kingtimeseriesdiff1', by lag

1 2 3 4 5 6 7 8 9 10 11

-0.360 -0.335 -0.321 0.005 0.025 -0.144 -0.022 -0.007 -0.143 -0.167 0.065

12 13 14 15 16 17 18 19 20

0.034 -0.161 0.036 0.066 0.081 -0.005 -0.027 -0.006 -0.037



The partial correlogram shows that the partial autocorrelations at lags 1, 2 and 3 exceed the significance bounds, are negative, and are slowly decreasing in magnitude with increasing lag (lag 1: -0.360, lag 2: -0.335, lag 3:-0.321). The partial autocorrelations tail off to zero after lag 3.

Since the correlogram is zero after lag 1, and the partial correlogram tails off to zero after lag 3, this means that the following ARMA (autoregressive moving average) models are possible for the time series of first differences:

* an ARMA(3,0) model, that is, an autoregressive model of order p=3, since the partial autocorrelogram is zero after lag 3, and the autocorrelogram tails off to zero (although perhaps too abruptly for this model to be appropriate)
* an ARMA(0,1) model, that is, a moving average model of order q=1, since the autocorrelogram is zero after lag 1 and the partial autocorrelogram tails off to zero
* an ARMA(p,q) model, that is, a mixed model with p and q greater than 0, since the autocorrelogram and partial correlogram tail off to zero (although the correlogram probably tails off to zero too abruptly for this model to be appropriate)

We use the principle of parsimony to decide which model is best: that is, we assume that the model with the fewest parameters is best. The ARMA(3,0) model has 3 parameters, the ARMA(0,1) model has 1 parameter, and the ARMA(p,q) model has at least 2 parameters. Therefore, the ARMA(0,1) model is taken as the best model.

An ARMA(0,1) model is a moving average model of order 1, or MA(1) model. This model can be written as: X\_t - mu = Z\_t - (theta \* Z\_t-1), where X\_t is the stationary time series we are studying (the first differenced series of ages at death of English kings), mu is the mean of time series X\_t, Z\_t is white noise with mean zero and constant variance, and theta is a parameter that can be estimated.

A MA (moving average) model is usually used to model a time series that shows short-term dependencies between successive observations. Intuitively, it makes good sense that a MA model can be used to describe the irregular component in the time series of ages at death of English kings, as we might expect the age at death of a particular English king to have some effect on the ages at death of the next king or two, but not much effect on the ages at death of kings that reign much longer after that.

Shortcut: the auto.arima() function

The auto.arima() function can be used to find the appropriate ARIMA model, eg., type “library(forecast)”, then “auto.arima(kings)”. The output says an appropriate model is ARIMA(0,1,1).

Since an ARMA(0,1) model (with p=0, q=1) is taken to be the best candidate model for the time series of first differences of the ages at death of English kings, then the original time series of the ages of death can be modelled using an ARIMA(0,1,1) model (with p=0, d=1, q=1, where d is the order of differencing required).

#### Example of the Volcanic Dust Veil in the Northern Hemisphere

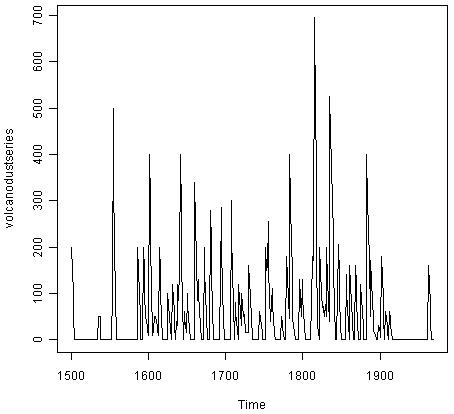
Let’s take another example of selecting an appropriate ARIMA model. The file file <http://robjhyndman.com/tsdldata/annual/dvi.dat> contains data on the volcanic dust veil index in the northern hemisphere, from 1500-1969 (original data from Hipel and Mcleod, 1994). This is a measure of the impact of volcanic eruptions’ release of dust and aerosols into the environment. We can read it into R and make a time plot by typing:

> volcanodust <- scan("http://robjhyndman.com/tsdldata/annual/dvi.dat", skip=1)

Read 470 items

> volcanodustseries <- ts(volcanodust,start=c(1500))

> plot.ts(volcanodustseries)



From the time plot, it appears that the random fluctuations in the time series are roughly constant in size over time, so an additive model is probably appropriate for describing this time series.

Furthermore, the time series appears to be stationary in mean and variance, as its level and variance appear to be roughly constant over time. Therefore, we do not need to difference this series in order to fit an ARIMA model, but can fit an ARIMA model to the original series (the order of differencing required, d, is zero here).

We can now plot a correlogram and partial correlogram for lags 1-20 to investigate what ARIMA model to use:

> acf(volcanodustseries, lag.max=20) # plot a correlogram

> acf(volcanodustseries, lag.max=20, plot=FALSE) # get the values of the autocorrelations

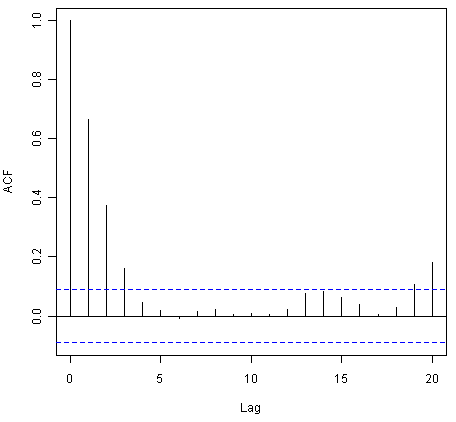
Autocorrelations of series 'volcanodustseries', by lag

0 1 2 3 4 5 6 7 8 9 10

1.000 0.666 0.374 0.162 0.046 0.017 -0.007 0.016 0.021 0.006 0.010

11 12 13 14 15 16 17 18 19 20

0.004 0.024 0.075 0.082 0.064 0.039 0.005 0.028 0.108 0.182



We see from the correlogram that the autocorrelations for lags 1, 2 and 3 exceed the significance bounds, and that the autocorrelations tail off to zero after lag 3. The autocorrelations for lags 1, 2, 3 are positive, and decrease in magnitude with increasing lag (lag 1: 0.666, lag 2: 0.374, lag 3: 0.162).

The autocorrelation for lags 19 and 20 exceed the significance bounds too, but it is likely that this is due to chance, since they just exceed the significance bounds (especially for lag 19), the autocorrelations for lags 4-18 do not exceed the signifiance bounds, and we would expect 1 in 20 lags to exceed the 95% significance bounds by chance alone.

> pacf(volcanodustseries, lag.max=20)

> pacf(volcanodustseries, lag.max=20, plot=FALSE)

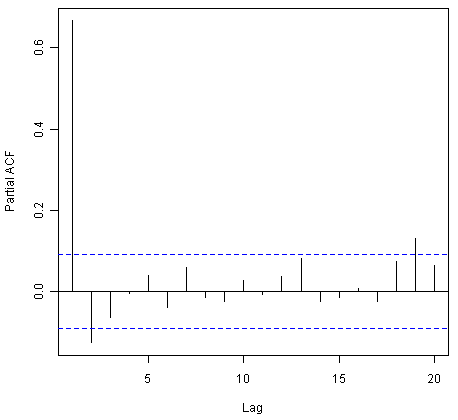
Partial autocorrelations of series 'volcanodustseries', by lag

1 2 3 4 5 6 7 8 9 10 11

0.666 -0.126 -0.064 -0.005 0.040 -0.039 0.058 -0.016 -0.025 0.028 -0.008

12 13 14 15 16 17 18 19 20

0.036 0.082 -0.025 -0.014 0.008 -0.025 0.073 0.131 0.063



From the partial autocorrelogram, we see that the partial autocorrelation at lag 1 is positive and exceeds the significance bounds (0.666), while the partial autocorrelation at lag 2 is negative and also exceeds the significance bounds (-0.126). The partial autocorrelations tail off to zero after lag 2.

Since the correlogram tails off to zero after lag 3, and the partial correlogram is zero after lag 2, the following ARMA models are possible for the time series:

* an ARMA(2,0) model, since the partial autocorrelogram is zero after lag 2, and the correlogram tails off to zero after lag 3, and the partial correlogram is zero after lag 2
* an ARMA(0,3) model, since the autocorrelogram is zero after lag 3, and the partial correlogram tails off to zero (although perhaps too abruptly for this model to be appropriate)
* an ARMA(p,q) mixed model, since the correlogram and partial correlogram tail off to zero (although the partial correlogram perhaps tails off too abruptly for this model to be appropriate)

Shortcut: the auto.arima() function

Again, we can use auto.arima() to find an appropriate model, by typing “auto.arima(volcanodust)”, which gives us ARIMA(1,0,2), which has 3 parameters. However, different criteria can be used to select a model (see auto.arima() help page). If we use the “bic” criterion, which penalises the number of parameters, we get ARIMA(2,0,0), which is ARMA(2,0): “auto.arima(volcanodust,ic=”bic”)”.

The ARMA(2,0) model has 2 parameters, the ARMA(0,3) model has 3 parameters, and the ARMA(p,q) model has at least 2 parameters. Therefore, using the principle of parsimony, the ARMA(2,0) model and ARMA(p,q) model are equally good candidate models.

An ARMA(2,0) model is an autoregressive model of order 2, or AR(2) model. This model can be written as: X\_t - mu = (Beta1 \* (X\_t-1 - mu)) + (Beta2 \* (Xt-2 - mu)) + Z\_t, where X\_t is the stationary time series we are studying (the time series of volcanic dust veil index), mu is the mean of time series X\_t, Beta1 and Beta2 are parameters to be estimated, and Z\_t is white noise with mean zero and constant variance.

An AR (autoregressive) model is usually used to model a time series which shows longer term dependencies between successive observations. Intuitively, it makes sense that an AR model could be used to describe the time series of volcanic dust veil index, as we would expect volcanic dust and aerosol levels in one year to affect those in much later years, since the dust and aerosols are unlikely to disappear quickly.

If an ARMA(2,0) model (with p=2, q=0) is used to model the time series of volcanic dust veil index, it would mean that an ARIMA(2,0,0) model can be used (with p=2, d=0, q=0, where d is the order of differencing required). Similarly, if an ARMA(p,q) mixed model is used, where p and q are both greater than zero, than an ARIMA(p,0,q) model can be used.

### Forecasting Using an ARIMA Model

Once you have selected the best candidate ARIMA(p,d,q) model for your time series data, you can estimate the parameters of that ARIMA model, and use that as a predictive model for making forecasts for future values of your time series.

You can estimate the parameters of an ARIMA(p,d,q) model using the “arima()” function in R.

#### Example of the Ages at Death of the Kings of England

For example, we discussed above that an ARIMA(0,1,1) model seems a plausible model for the ages at deaths of the kings of England. You can specify the values of p, d and q in the ARIMA model by using the “order” argument of the “arima()” function in R. To fit an ARIMA(p,d,q) model to this time series (which we stored in the variable “kingstimeseries”, see above), we type:

> kingstimeseriesarima <- arima(kingstimeseries, order=c(0,1,1)) # fit an ARIMA(0,1,1) model

> kingstimeseriesarima

ARIMA(0,1,1)

Coefficients:

ma1

-0.7218

s.e. 0.1208

sigma^2 estimated as 230.4: log likelihood = -170.06

AIC = 344.13 AICc = 344.44 BIC = 347.56

As mentioned above, if we are fitting an ARIMA(0,1,1) model to our time series, it means we are fitting an an ARMA(0,1) model to the time series of first differences. An ARMA(0,1) model can be written X\_t - mu = Z\_t - (theta \* Z\_t-1), where theta is a parameter to be estimated. From the output of the “arima()” R function (above), the estimated value of theta (given as ‘ma1’ in the R output) is -0.7218 in the case of the ARIMA(0,1,1) model fitted to the time series of ages at death of kings.

Specifying the confidence level for prediction intervals

You can specify the confidence level for prediction intervals in forecast.Arima() by using the “level” argument. For example, to get a 99.5% prediction interval, we would type “forecast.Arima(kingstimeseriesarima, h=5, level=c(99.5))”.

We can then use the ARIMA model to make forecasts for future values of the time series, using the “forecast.Arima()” function in the “forecast” R package. For example, to forecast the ages at death of the next five English kings, we type:

> library("forecast") # load the "forecast" R library

> kingstimeseriesforecasts <- forecast.Arima(kingstimeseriesarima, h=5)

> kingstimeseriesforecasts

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

43 67.75063 48.29647 87.20479 37.99806 97.50319

44 67.75063 47.55748 87.94377 36.86788 98.63338

45 67.75063 46.84460 88.65665 35.77762 99.72363

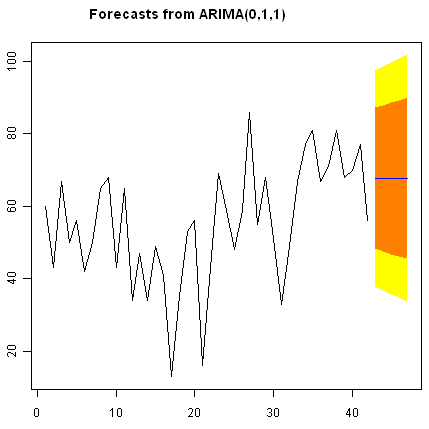
46 67.75063 46.15524 89.34601 34.72333 100.77792

47 67.75063 45.48722 90.01404 33.70168 101.79958

The original time series for the English kings includes the ages at death of 42 English kings. The forecast.Arima() function gives us a forecast of the age of death of the next five English kings (kings 43-47), as well as 80% and 95% prediction intervals for those predictions. The age of death of the 42nd English king was 56 years (the last observed value in our time series), and the ARIMA model gives the forecasted age at death of the next five kings as 67.8 years.

We can plot the observed ages of death for the first 42 kings, as well as the ages that would be predicted for these 42 kings and for the next 5 kings using our ARIMA(0,1,1) model, by typing:

> plot.forecast(kingstimeseriesforecasts)



As in the case of exponential smoothing models, it is a good idea to investigate whether the forecast errors of an ARIMA model are normally distributed with mean zero and constant variance, and whether the are correlations between successive forecast errors.

For example, we can make a correlogram of the forecast errors for our ARIMA(0,1,1) model for the ages at death of kings, and perform the Ljung-Box test for lags 1-20, by typing:

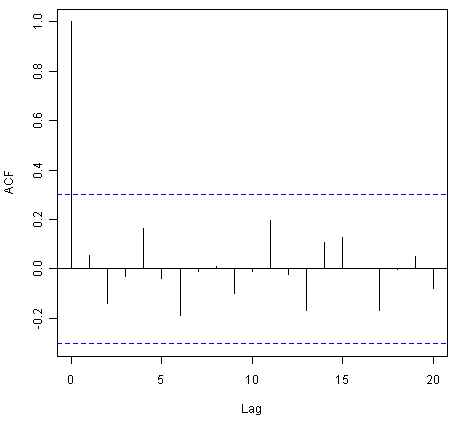
> acf(kingstimeseriesforecasts$residuals, lag.max=20)

> Box.test(kingstimeseriesforecasts$residuals, lag=20, type="Ljung-Box")

Box-Ljung test

data: kingstimeseriesforecasts$residuals

X-squared = 13.5844, df = 20, p-value = 0.851

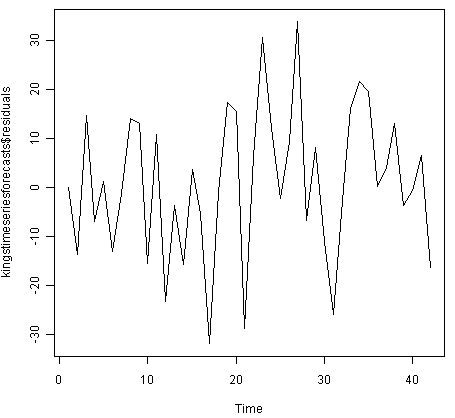


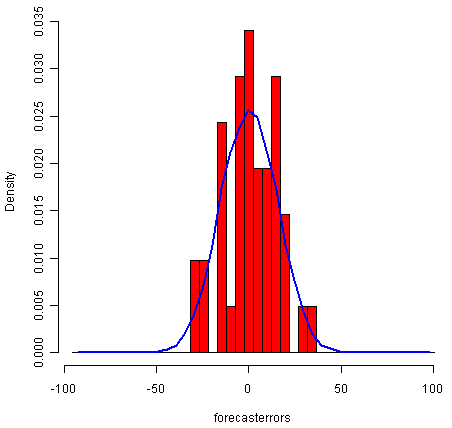
Since the correlogram shows that none of the sample autocorrelations for lags 1-20 exceed the significance bounds, and the p-value for the Ljung-Box test is 0.9, we can conclude that there is very little evidence for non-zero autocorrelations in the forecast errors at lags 1-20.

To investigate whether the forecast errors are normally distributed with mean zero and constant variance, we can make a time plot and histogram (with overlaid normal curve) of the forecast errors:

> plot.ts(kingstimeseriesforecasts$residuals) # make time plot of forecast errors

> plotForecastErrors(kingstimeseriesforecasts$residuals) # make a histogram





The time plot of the in-sample forecast errors shows that the variance of the forecast errors seems to be roughly constant over time (though perhaps there is slightly higher variance for the second half of the time series). The histogram of the time series shows that the forecast errors are roughly normally distributed and the mean seems to be close to zero. Therefore, it is plausible that the forecast errors are normally distributed with mean zero and constant variance.

Since successive forecast errors do not seem to be correlated, and the forecast errors seem to be normally distributed with mean zero and constant variance, the ARIMA(0,1,1) does seem to provide an adequate predictive model for the ages at death of English kings.

#### Example of the Volcanic Dust Veil in the Northern Hemisphere

We discussed above that an appropriate ARIMA model for the time series of volcanic dust veil index may be an ARIMA(2,0,0) model. To fit an ARIMA(2,0,0) model to this time series, we can type:

> volcanodustseriesarima <- arima(volcanodustseries, order=c(2,0,0))

> volcanodustseriesarima

ARIMA(2,0,0) with non-zero mean

Coefficients:

ar1 ar2 intercept

0.7533 -0.1268 57.5274

s.e. 0.0457 0.0458 8.5958

sigma^2 estimated as 4870: log likelihood = -2662.54

AIC = 5333.09 AICc = 5333.17 BIC = 5349.7

As mentioned above, an ARIMA(2,0,0) model can be written as: written as: X\_t - mu = (Beta1 \* (X\_t-1 - mu)) + (Beta2 \* (Xt-2 - mu)) + Z\_t, where Beta1 and Beta2 are parameters to be estimated. The output of the arima() function tells us that Beta1 and Beta2 are estimated as 0.7533 and -0.1268 here (given as ar1 and ar2 in the output of arima()).

Now we have fitted the ARIMA(2,0,0) model, we can use the “forecast.ARIMA()” model to predict future values of the volcanic dust veil index. The original data includes the years 1500-1969. To make predictions for the years 1970-2000 (31 more years), we type:

> volcanodustseriesforecasts <- forecast.Arima(volcanodustseriesarima, h=31)

> volcanodustseriesforecasts

Point Forecast Lo 80 Hi 80 Lo 95 Hi 95

1970 21.48131 -67.94860 110.9112 -115.2899 158.2526

1971 37.66419 -74.30305 149.6314 -133.5749 208.9033

1972 47.13261 -71.57070 165.8359 -134.4084 228.6737

1973 52.21432 -68.35951 172.7881 -132.1874 236.6161

1974 54.84241 -66.22681 175.9116 -130.3170 240.0018

1975 56.17814 -65.01872 177.3750 -129.1765 241.5327

1976 56.85128 -64.37798 178.0805 -128.5529 242.2554

1977 57.18907 -64.04834 178.4265 -128.2276 242.6057

1978 57.35822 -63.88124 178.5977 -128.0615 242.7780

1979 57.44283 -63.79714 178.6828 -127.9777 242.8634

1980 57.48513 -63.75497 178.7252 -127.9356 242.9059

1981 57.50627 -63.73386 178.7464 -127.9145 242.9271

1982 57.51684 -63.72330 178.7570 -127.9040 242.9376

1983 57.52212 -63.71802 178.7623 -127.8987 242.9429

1984 57.52476 -63.71538 178.7649 -127.8960 242.9456

1985 57.52607 -63.71407 178.7662 -127.8947 242.9469

1986 57.52673 -63.71341 178.7669 -127.8941 242.9475

1987 57.52706 -63.71308 178.7672 -127.8937 242.9479

1988 57.52723 -63.71291 178.7674 -127.8936 242.9480

1989 57.52731 -63.71283 178.7674 -127.8935 242.9481

1990 57.52735 -63.71279 178.7675 -127.8934 242.9481

1991 57.52737 -63.71277 178.7675 -127.8934 242.9482

1992 57.52738 -63.71276 178.7675 -127.8934 242.9482

1993 57.52739 -63.71275 178.7675 -127.8934 242.9482

1994 57.52739 -63.71275 178.7675 -127.8934 242.9482

1995 57.52739 -63.71275 178.7675 -127.8934 242.9482

1996 57.52739 -63.71275 178.7675 -127.8934 242.9482

1997 57.52739 -63.71275 178.7675 -127.8934 242.9482

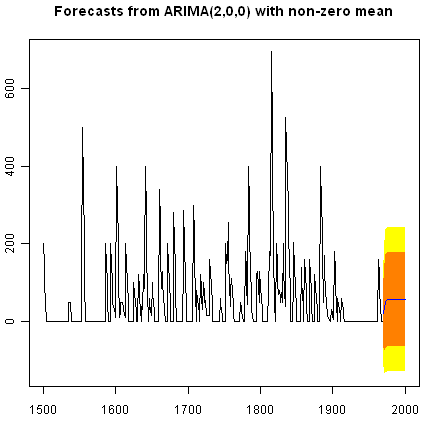
1998 57.52739 -63.71275 178.7675 -127.8934 242.9482

1999 57.52739 -63.71275 178.7675 -127.8934 242.9482

2000 57.52739 -63.71275 178.7675 -127.8934 242.9482

We can plot the original time series, and the forecasted values, by typing:

> plot.forecast(volcanodustseriesforecasts)



One worrying thing is that the model has predicted negative values for the volcanic dust veil index, but this variable can only have positive values! The reason is that the arima() and forecast.Arima() functions don’t know that the variable can only take positive values. Clearly, this is not a very desirable feature of our current predictive model.

Again, we should investigate whether the forecast errors seem to be correlated, and whether they are normally distributed with mean zero and constant variance. To check for correlations between successive forecast errors, we can make a correlogram and use the Ljung-Box test:

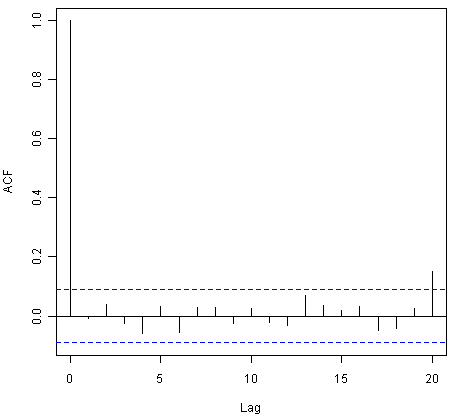
> acf(volcanodustseriesforecasts$residuals, lag.max=20)

> Box.test(volcanodustseriesforecasts$residuals, lag=20, type="Ljung-Box")

Box-Ljung test

data: volcanodustseriesforecasts$residuals

X-squared = 24.3642, df = 20, p-value = 0.2268

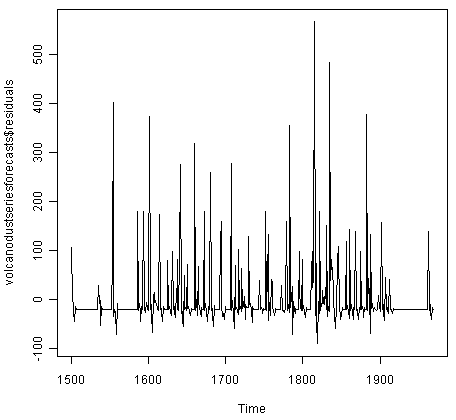


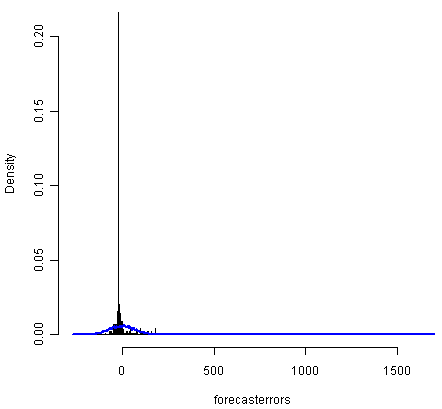
The correlogram shows that the sample autocorrelation at lag 20 exceeds the significance bounds. However, this is probably due to chance, since we would expect one out of 20 sample autocorrelations to exceed the 95% significance bounds. Furthermore, the p-value for the Ljung-Box test is 0.2, indicating that there is little evidence for non-zero autocorrelations in the forecast errors for lags 1-20.

To check whether the forecast errors are normally distributed with mean zero and constant variance, we make a time plot of the forecast errors, and a histogram:

> plot.ts(volcanodustseriesforecasts$residuals) # make time plot of forecast errors

> plotForecastErrors(volcanodustseriesforecasts$residuals) # make a histogram





The time plot of forecast errors shows that the forecast errors seem to have roughly constant variance over time. However, the time series of forecast errors seems to have a negative mean, rather than a zero mean. We can confirm this by calculating the mean forecast error, which turns out to be about -0.22:

> mean(volcanodustseriesforecasts$residuals)

-0.2205417

The histogram of forecast errors (above) shows that although the mean value of the forecast errors is negative, the distribution of forecast errors is skewed to the right compared to a normal curve. Therefore, it seems that we cannot comfortably conclude that the forecast errors are normally distributed with mean zero and constant variance! Thus, it is likely that our ARIMA(2,0,0) model for the time series of volcanic dust veil index is not the best model that we could make, and could almost definitely be improved upon!

## Links and Further Reading

Here are some links for further reading.

For a more in-depth introduction to R, a good online tutorial is available on the “Kickstarting R” website, [cran.r-project.org/doc/contrib/Lemon-kickstart](http://cran.r-project.org/doc/contrib/Lemon-kickstart/).

There is another nice (slightly more in-depth) tutorial to R available on the “Introduction to R” website, [cran.r-project.org/doc/manuals/R-intro.html](http://cran.r-project.org/doc/manuals/R-intro.html).

You can find a list of R packages for analysing time series data on the [CRAN Time Series Task View webpage](http://cran.r-project.org/web/views/TimeSeries.html).

To learn about time series analysis, I would highly recommend the book “Time series” (product code M249/02) by the Open University, available from [the Open University Shop](http://www.ouw.co.uk/store/).

There are two books available in the “Use R!” series on using R for time series analyses, the first is [Introductory Time Series with R](http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-88697-8) by Cowpertwait and Metcalfe, and the second is [Analysis of Integrated and Cointegrated Time Series with R](http://www.springer.com/statistics/statistical+theory+and+methods/book/978-0-387-75966-1) by Pfaff.

## Acknowledgements

I am grateful to [Professor Rob Hyndman](http://robjhyndman.com/), for kindly allowing me to use the time series data sets from his [Time Series Data Library (TSDL)](http://robjhyndman.com/TSDL/) in the examples in this booklet.

Many of the examples in this booklet are inspired by examples in the excellent Open University book, “Time series” (product code M249/02), available from [the Open University Shop](http://www.ouw.co.uk/store/).

Thank you to Ravi Aranke for bringing auto.arima() to my attention, and Maurice Omane-Adjepong for bringing unit root tests to my attention, and Christian Seubert for noticing a small bug in plotForecastErrors(). Thank you for other comments to Antoine Binard and Bill Johnston.

## Contact

I will be grateful if you will send me (Avril Coghlan) corrections or suggestions for improvements to my email address [alc@sanger.ac.uk](mailto:alc%40sanger.ac.uk)

## License

The content in this book is licensed under a [Creative Commons Attribution 3.0 License](http://creativecommons.org/licenses/by/3.0/).

* [index](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/genindex.html)
* [previous](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/src/installr.html) |
* [Time Series 0.2 documentation](http://a-little-book-of-r-for-time-series.readthedocs.org/en/latest/index.html) »

© Copyright 2010, Avril Coghlan. Created using [Sphinx](http://sphinx-doc.org/) 1.3.1.

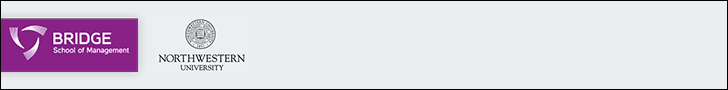
  v: latest

# A Complete Tutorial on Time Series Modeling in R

[Business Analytics](http://www.analyticsvidhya.com/blog/category/business-analytics/)

SHARE

http://2.gravatar.com/avatar/51bf7a2a1f2a0dd3a1012fa4cf5a9e80?s=48&d=mm&r=g[Tavish Srivastava](http://www.analyticsvidhya.com/blog/author/tavish1/) , December 16, 2015 / [16](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comments)

[](http://admissions.bridgesom.com/pba-new/?utm_source=AV&utm_medium=Banner&utm_campaign=AVBanner)

## Introduction

‘Time’ is the most important factor which ensures success in a business. It’s difficult to keep up with the pace of time.  But, technology has developed some powerful methods using which we can ‘see things’ ahead of time. Don’t worry, I am not talking about Time Machine. Let’s be realistic here!

I’m talking about the methods of prediction & forecasting. One such method, which deals with time based data is **Time Series Modeling**. As the name suggests, it involves working on time (years, days, hours, minutes) based data, to derive hidden insights to make informed decision making.

Time series models are very useful models when you have serially correlated data. Most of business houses work on time series data to analyze sales number for the next year, website traffic, competition position and much more. However, it is also one of the areas, which many analysts do not understand.

So, if you aren’t sure about complete process of time series modeling, this guide would introduce you to various levels of time series modeling and its related techniques.

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/12/tsm1.jpg)

The following topics are covered in this tutorial as shown below:

## Table of Contents

1. Basics – Time Series Modeling
2. Exploration of Time Series Data in R
3. Introduction to ARMA Time Series Modeling
4. Framework and Application of ARIMA Time Series Modeling

Time to get started!

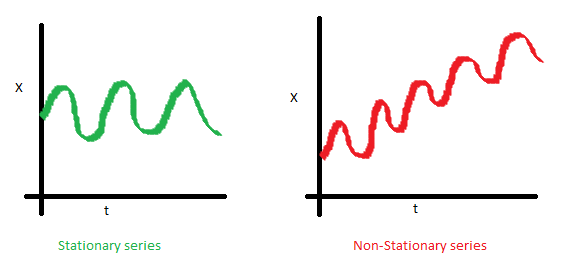
## 1. Basics – Time Series Modeling

Let’s begin from basics.  This includes stationary series, random walks , Rho Coefficient, Dickey Fuller Test of Stationarity. If these terms are already scaring you, don’t worry – they will become clear in a bit and I bet you will start enjoying the subject as I explain it.

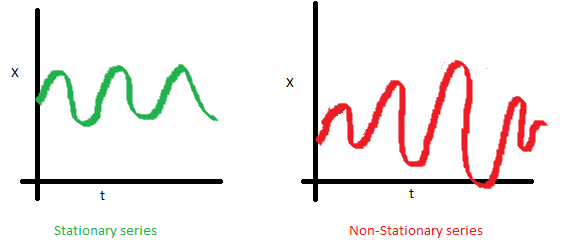
### Stationary Series

There are three basic criterion for a series to be classified as stationary series :

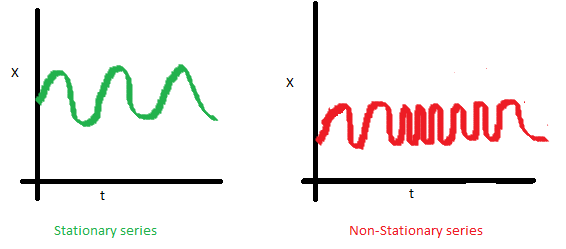
1. The mean of the series should not be a function of time rather should be a constant. The image below has the left hand graph satisfying the condition whereas the graph in red has a time dependent mean.

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Mean_nonstationary.png)

2. The variance of the series should not a be a function of time. This property is known as homoscedasticity. Following graph depicts what is and what is not a stationary series. (Notice the varying spread of distribution in the right hand graph)

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Var_nonstationary.png)

3. The covariance of the i th term and the (i + m) th term should not be a function of time. In the following graph, you will notice the spread becomes closer as the time increases. Hence, the covariance is not constant with time for the ‘red series’.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Cov_nonstationary.png)

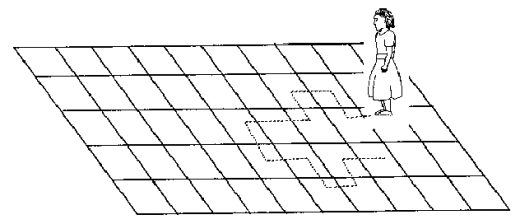
### Why do I care about ‘stationarity’ of a time series?

The reason I took up this section first was that until unless your time series is stationary, you cannot build a time series model. In cases where the stationary criterion are violated, the first requisite becomes to stationarize the time series and then try stochastic models to predict this time series. There are multiple ways of bringing this stationarity. Some of them are Detrending, Differencing etc.

### Random Walk

This is the most basic concept of the time series. You might know the concept well. But, I found many people in the industry who interprets random walk as a stationary process. In this section with the help of some mathematics, I will make this concept crystal clear for ever. Let’s take an example.

**Example:** Imagine a girl moving randomly on a giant chess board. In this case, next position of the girl is only dependent on the last position.

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/RandomWalk.gif)

(Source: http://scifun.chem.wisc.edu/WOP/RandomWalk.html )

Now imagine, you are sitting in another room and are not able to see the girl. You want to predict the position of the girl with time. How accurate will you be? Of course you will become more and more inaccurate as the position of the girl changes. At t=0 you exactly know where the girl is. Next time, she can only move to 8 squares and hence your probability dips to 1/8 instead of 1 and it keeps on going down. Now let’s try to formulate this series :

X(t) = X(t-1) + Er(t)

where Er(t) is the error at time point t. This is the randomness the girl brings at every point in time.

Now, if we recursively fit in all the Xs, we will finally end up to the following equation :

X(t) = X(0) + Sum(Er(1),Er(2),Er(3).....Er(t))

Now, lets try validating our assumptions of stationary series on this random walk formulation:

**1. Is the Mean constant ?**

E[X(t)] = E[X(0)] + Sum(E[Er(1)],E[Er(2)],E[Er(3)].....E[Er(t)])

We know that Expectation of any Error will be zero as it is random.

Hence we get E[X(t)] = E[X(0)] = Constant.

**2. Is the Variance constant?**

Var[X(t)] = Var[X(0)] + Sum(Var[Er(1)],Var[Er(2)],Var[Er(3)].....Var[Er(t)])

Var[X(t)] = t \* Var(Error) = Time dependent.

Hence, we infer that the random walk is not a stationary process as it has a time variant variance. Also, if we check the covariance, we see that too is dependent on time.

### Let’s spice up things a bit,

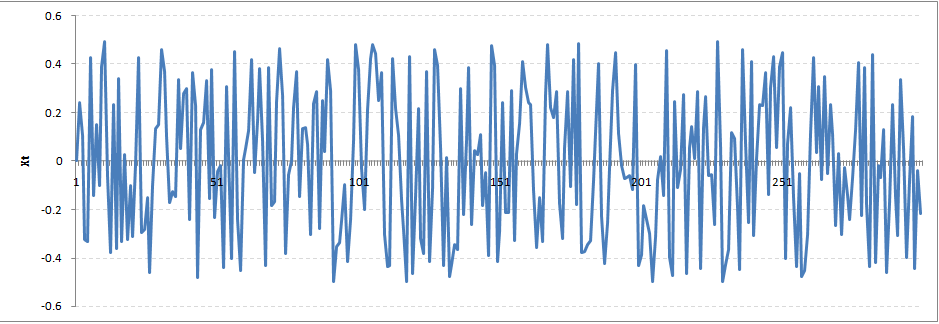
We already know that a random walk is a non-stationary process. Let us introduce a new coefficient in the equation to see if we can make the formulation stationary.

**Introduced coefficient : Rho**

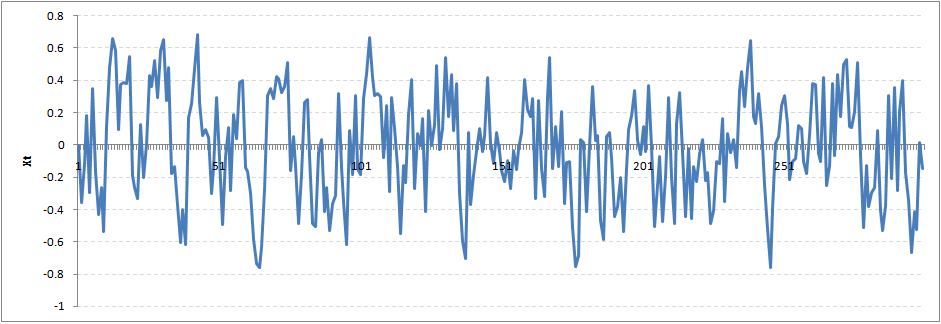
X(t) = Rho \* X(t-1) + Er(t)

Now, we will vary the value of Rho to see if we can make the series stationary. Here we will interpret the scatter visually and not do any test to check stationarity.

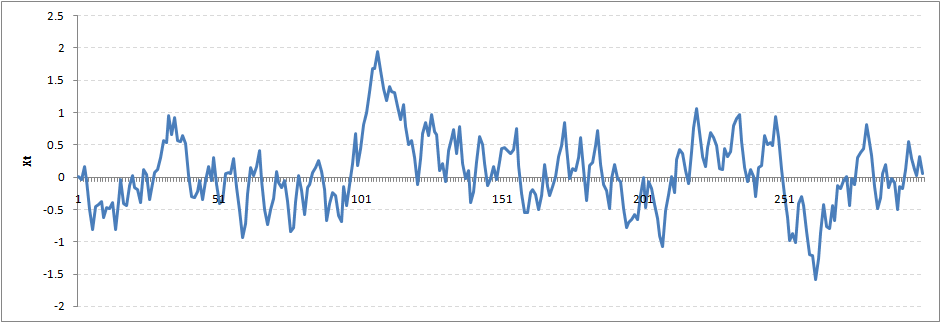
Let’s start with a perfectly stationary series with Rho = 0 . Here is the plot for the time series :

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/rho0.png)

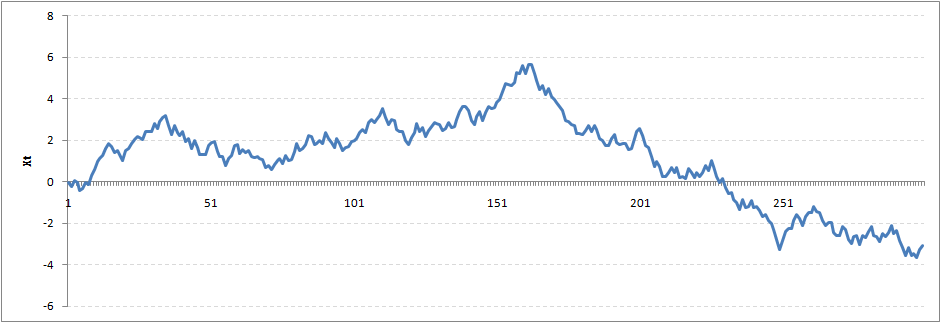
Increase the value of Rho to 0.5 gives us following graph :

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/rho5.png)

You might notice that our cycles have become broader but essentially there does not seem to be a serious violation of stationary assumptions. Let’s now take a more extreme case of Rho = 0.9

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/rho9.png)

We still see that the X returns back from extreme values to zero after some intervals. This series also is not violating non-stationarity significantly. Now, let’s take a look at the random walk with rho = 1.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/rho1.png)

This obviously is an violation to stationary conditions. What makes rho = 1 a special case which comes out badly in stationary test? We will find the mathematical reason to this.

Let’s take expectation on each side of the equation  “X(t) = Rho \* X(t-1) + Er(t)”

E[X(t)] = Rho \*E[ X(t-1)]

This equation is very insightful. The next X (or at time point t) is being pulled down to Rho \* Last value of X.

For instance, if X(t – 1 ) = 1, E[X(t)] = 0.5 ( for Rho = 0.5) . Now, if X moves to any direction from zero, it is pulled back to zero in next step. The only component which can drive it even further is the error term. Error term is equally probable to go in either direction. What happens when the Rho becomes 1? No force can pull the X down in the next step.

### Dickey Fuller Test of Stationarity

What you just learnt in the last section is formally known as Dickey Fuller test. Here is a small tweak which is made for our equation to convert it to a Dickey Fuller test:

X(t) = Rho \* X(t-1) + Er(t)

=>  X(t) - X(t-1) = (Rho - 1) X(t - 1) + Er(t)

We have to test if Rho – 1 is significantly different than zero or not. If the null hypothesis gets rejected, we’ll get a stationary time series.

Stationary testing and converting a series into a stationary series are the most critical processes in a time series modelling. You need to memorize each and every detail of this concept to move on to the next step of time series modelling.

Let’s now consider an example to show you what a time series looks like.

## 2. Exploration of Time Series Data in R

Here we’ll learn to handle time series data on R. Our scope will be restricted to data exploring in a time series type of data set and not go to building time series models.

I have used an inbuilt data set of R called AirPassengers. The dataset consists of monthly totals of international airline passengers, 1949 to 1960.

### Loading the Data Set

Following is the code which will help you load the data set and spill out a few top level metrics.

> data(AirPassengers)

> class(AirPassengers)

[1] "ts"

#This tells you that the data series is in a time series format

> start(AirPassengers)

[1] 1949 1

#This is the start of the time series

> end(AirPassengers)

[1] 1960 12

#This is the end of the time series

> frequency(AirPassengers)

[1] 12

#The cycle of this time series is 12months in a year

> summary(AirPassengers)

Min. 1st Qu. Median Mean 3rd Qu. Max.

104.0 180.0 265.5 280.3 360.5 622.0

### Detailed Metrics

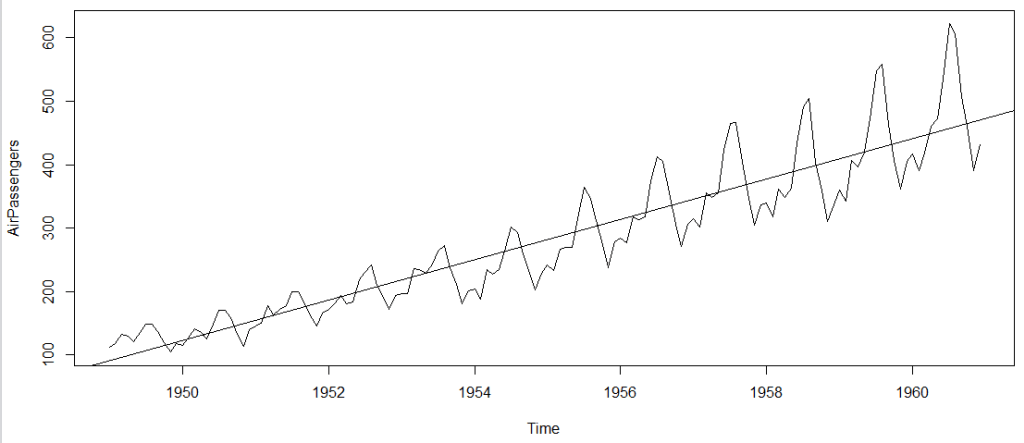
#The number of passengers are distributed across the spectrum

> plot(AirPassengers)

#This will plot the time series

>abline(reg=lm(AirPassengers~time(AirPassengers)))

# This will fit in a line

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_AP1.png)

Here are a few more operations you can do:

> cycle(AirPassengers)

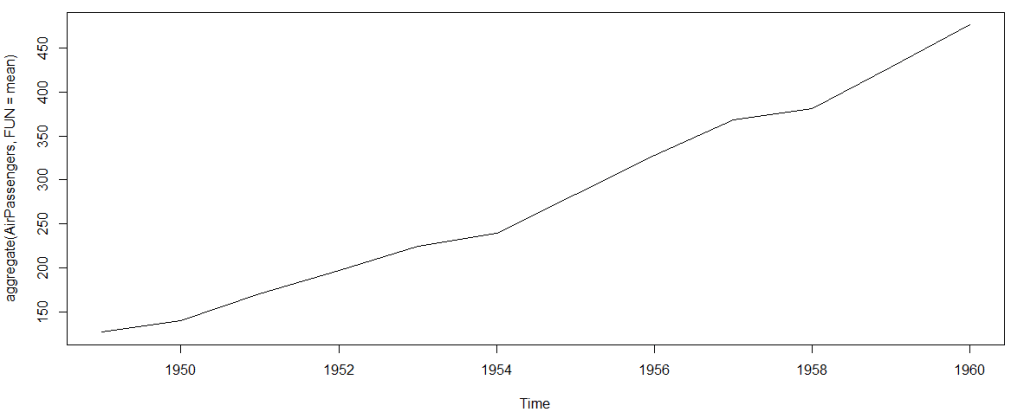
#This will print the cycle across years.

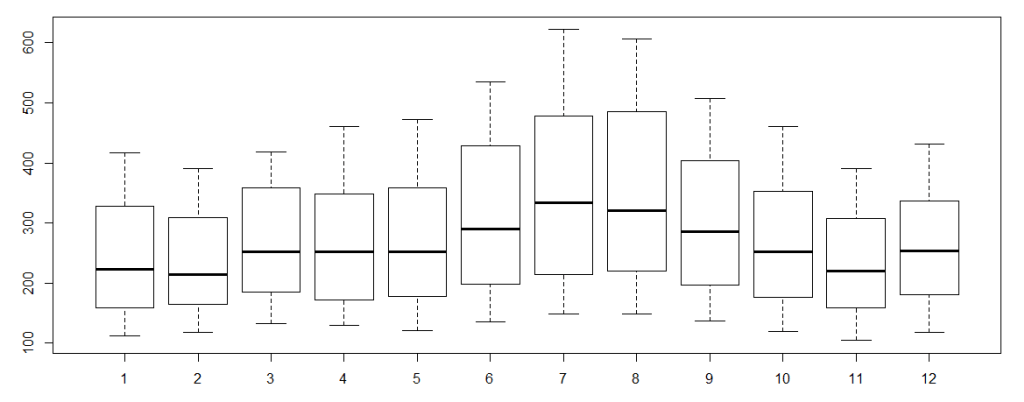
>plot(aggregate(AirPassengers,FUN=mean))

#This will aggregate the cycles and display a year on year trend

> boxplot(AirPassengers~cycle(AirPassengers))

#Box plot across months will give us a sense on seasonal effect

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_aggregate.png)

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_month_wise.png)

### Important Inferences

1. The year on year trend clearly shows that the #passengers have been increasing without fail.
2. The variance and the mean value in July and August is much higher than rest of the months.
3. Even though the mean value of each month is quite different their variance is small. Hence, we have strong seasonal effect with a cycle of 12 months or less.

Exploring data becomes most important in a time series model – without this exploration, you will not know whether a series is stationary or not. As in this case we already know many details about the kind of model we are looking out for.

Let’s now take up a few time series models and their characteristics. We will also take this problem forward and make a few predictions.

## 3. Introduction to ARMA Time Series Modeling

ARMA models are commonly used in time series modeling. In ARMA model, AR stands for auto-regression and MA stands for moving average. If these words sound intimidating to you, worry not – I’ll simplify these concepts in next few minutes for you!

We will now develop a knack for these terms and understand the characteristics associated with these models. **But before we start, you should remember, AR or MA are not applicable on non-stationary series**.

In case you get a non stationary series, you first need to stationarize the series (by taking difference / transformation) and then choose from the available time series models.

First, I’ll explain each of these two models (AR & MA) individually. Next, we will look at the characteristics of these models.

### Auto-Regressive Time Series Model

Let’s understanding AR models using the case below:

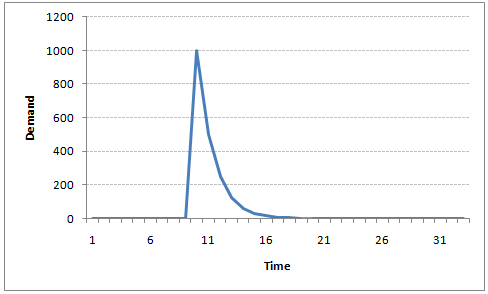
The current GDP of a country say x(t) is dependent on the last year’s GDP i.e. x(t – 1). The hypothesis being that the total cost of production of products & services in a country in a fiscal year (known as GDP) is dependent on the set up of manufacturing plants / services in the previous year and the newly set up industries / plants / services in the current year. But the primary component of the GDP is the former one.

Hence, we can formally write the equation of GDP as:

**x(t) = alpha \*  x(t – 1) + error (t)**

This equation is known as AR(1) formulation. The numeral one (1) denotes that the next instance is solely dependent on the previous instance.  The alpha is a coefficient which we seek so as to minimize the error function. Notice that x(t- 1) is indeed linked to x(t-2) in the same fashion. Hence, any shock to x(t) will gradually fade off in future.

For instance, let’s say x(t) is the number of juice bottles sold in a city on a particular day. During winters, very few vendors purchased juice bottles. Suddenly, on a particular day, the temperature rose and the demand of juice bottles soared to 1000. However, after a few days, the climate became cold again. But, knowing that the people got used to drinking juice during the hot days, there were 50% of the people still drinking juice during the cold days. In following days, the proportion went down to 25% (50% of 50%) and then gradually to a small number after significant number of days. The following graph explains the inertia property of AR series:

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/AR1.png)

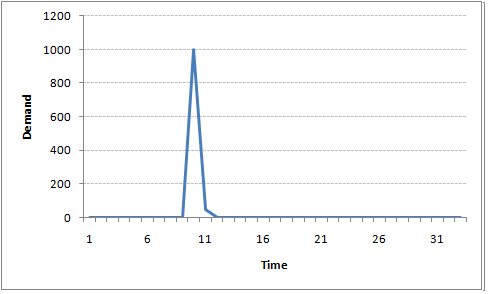
### Moving Average Time Series Model

Let’s take another case to understand Moving average time series model.

A manufacturer produces a certain type of bag, which was readily available in the market. Being a competitive market, the sale of the bag stood at zero for many days. So, one day he did some experiment with the design and produced a different type of bag. This type of bag was not available anywhere in the market. Thus, he was able to sell the entire stock of 1000 bags (lets call this as x(t) ). The demand got so high that the bag ran out of stock. As a result, some 100 odd customers couldn’t purchase this bag. Lets call this gap as the error at that time point. With time, the bag had lost its woo factor. But still few customers were left who went empty handed the previous day. Following is a simple formulation to depict the scenario :

**x(t) = beta \*  error(t-1) + error (t)**

If we try plotting this graph, it will look something like this :

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/MA1.png)

Did you notice the difference between MA and AR model? In MA model, noise / shock quickly vanishes with time. The AR model has a much lasting effect of the shock.

### Difference between AR and MA models

The primary difference between an AR and MA model is based on the correlation between time series objects at different time points. The correlation between x(t) and x(t-n) for n > order of MA is always zero. This directly flows from the fact that covariance between x(t) and x(t-n) is zero for MA models (something which we refer from the example taken in the previous section). However, the correlation of x(t) and x(t-n) gradually declines with n becoming larger in the AR model. This difference gets exploited irrespective of having the AR model or MA model. The correlation plot can give us the order of MA model.

### Exploiting ACF and PACF plots

Once we have got the stationary time series, we must answer two primary questions:

Q1. Is it an AR or MA process?

Q2. What order of AR or MA process do we need to use?

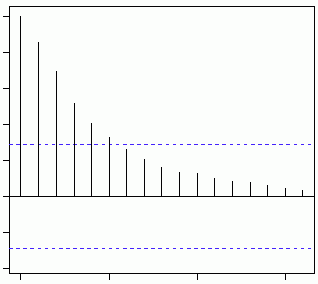
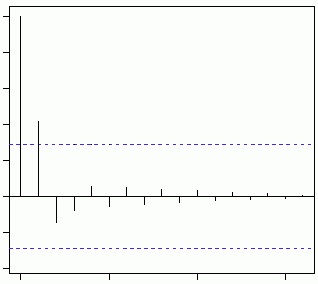
The trick to solve these questions is available in the previous section. Didn’t you notice?

The first question can be answered using Total Correlation Chart (also known as Auto – correlation Function / ACF). ACF is a plot of total correlation between different lag functions. For instance, in GDP problem, the GDP at time point t is x(t). We are interested in the correlation of x(t) with x(t-1) , x(t-2) and so on. Now let’s reflect on what we have learnt above.

In a moving average series of lag n, we will not get any correlation between x(t) and x(t – n -1) . Hence, the total correlation chart cuts off at nth lag. So it becomes simple to find the lag for a MA series. For an AR series this correlation will gradually go down without any cut off value. So what do we do if it is an AR series?

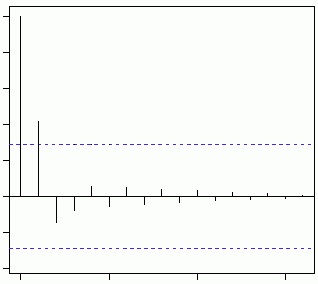
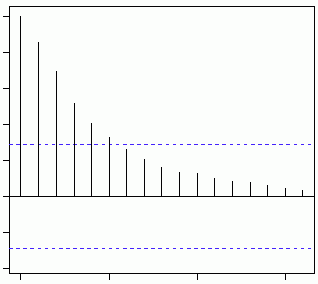
Here is the second trick. If we find out the partial correlation of each lag, it will cut off after the degree of AR series. For instance,if we have a AR(1) series,  if we exclude the effect of 1st lag (x (t-1) ), our 2nd lag (x (t-2) ) is independent of x(t). Hence, the partial correlation function (PACF) will drop sharply after the 1st lag. Following are the examples which will clarify any doubts you have on this concept :

**ACF**                                                                      **PACF**

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Gradual-decline.gif)[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/cut-off.gif)

The blue line above shows significantly different values than zero. Clearly, the graph above has a cut off on PACF curve after 2nd lag which means this is mostly an AR(2) process.

**ACF                                                                 PACF**

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/cut-off.gif)[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Gradual-decline.gif)

Clearly, the graph above has a cut off on ACF curve after 2nd lag which means this is mostly a MA(2) process.

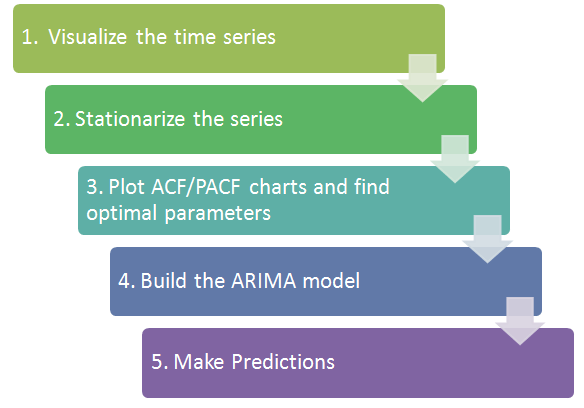
Till now, we have covered on how to identify the type of stationary series using ACF & PACF plots. Now, I’ll introduce you to a comprehensive framework to build a time series model.  In addition, we’ll also discuss about the practical applications of time series modelling.

## 4. Framework and Application of ARIMA Time Series Modeling

A quick revision, Till here we’ve learnt basics of time series modeling, time series in R and ARMA modeling. Now is the time to join these pieces and make an interesting story.

### Overview of the Framework

This framework(shown below) specifies the step by step approach on ‘**How to do a Time Series Analysis**‘:

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/flowchart.png)

As you would be aware, the first three steps have already been discussed above. Nevertheless, the same has been delineated briefly below:

### Step 1: Visualize the Time Series

It is essential to analyze the trends prior to building any kind of time series model. The details we are interested in pertains to any kind of trend, seasonality or random behaviour in the series. We have covered this part in the second part of this series.

### Step 2: Stationarize the Series

Once we know the patterns, trends, cycles and seasonality , we can check if the series is stationary or not. Dickey – Fuller is one of the popular test to check the same. We have covered this test in the [first part](http://www.analyticsvidhya.com/blog/2015/02/step-step-guide-learn-time-series/) of this article series. This doesn’t ends here! What if the series is found to be non-stationary?

There are three commonly used technique to make a time series stationary:

1.  **Detrending** : Here, we simply remove the trend component from the time series. For instance, the equation of my time series is:

**x(t) = (mean + trend \* t) + error**

We’ll simply remove the part in the parentheses and build model for the rest.

2. **Differencing** : This is the commonly used technique to remove non-stationarity. Here we try to model the differences of the terms and not the actual term. For instance,

**x(t) – x(t-1) = ARMA (p ,  q)**

This differencing is called as the Integration part in AR(I)MA. Now, we have three parameters

**p : AR**

**d : I**

**q : MA**

3. **Seasonality** : Seasonality can easily be incorporated in the ARIMA model directly. More on this has been discussed in the applications part below.

### Step 3: Find Optimal Parameters

The parameters p,d,q can be found using  [ACF and PACF plots](http://www.analyticsvidhya.com/blog/2015/03/introduction-auto-regression-moving-average-time-series/). An addition to this approach is can be, if both ACF and PACF decreases gradually, it indicates that we need to make the time series stationary and introduce a value to “d”.

### Step 4: Build ARIMA Model

With the parameters in hand, we can now try to build ARIMA model. The value found in the previous section might be an approximate estimate and we need to explore more (p,d,q) combinations. The one with the lowest BIC and AIC should be our choice. We can also try some models with a seasonal component. Just in case, we notice any seasonality in ACF/PACF plots.

### Step 5: Make Predictions

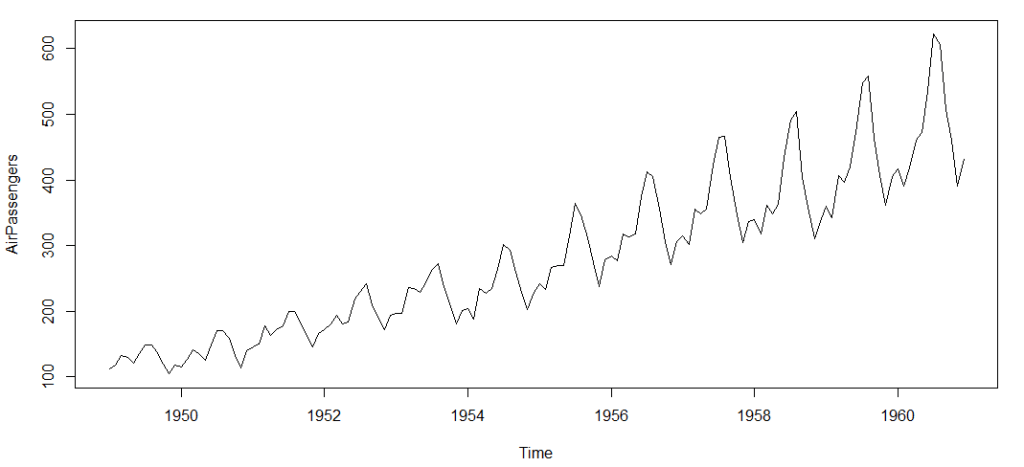
Once we have the final ARIMA model, we are now ready to make predictions on the future time points. We can also visualize the trends to cross validate if the model works fine.

### Applications of Time Series Model

Now, we’ll use the same example that we have used above. Then, using time series, we’ll make future predictions. We recommend you to check out the example before proceeding further.

### Where did we start ?

Following is the plot of the number of passengers with years. Try and make observations on this plot before moving further in the article.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/plot_AP.png)

Here are my observations :

1. There is a trend component which grows the passenger year by year.

2. There looks to be a seasonal component which has a cycle less than 12 months.

3. The variance in the data keeps on increasing with time.

We know that we need to address two issues before we test stationary series. One, we need to remove unequal variances. We do this using log of the series. Two, we need to address the trend component. We do this by taking difference of the series. Now, let’s test the resultant series.

adf.test(diff(log(AirPassengers)), alternative="stationary", k=0)

Augmented Dickey-Fuller Test

data: diff(log(AirPassengers))

Dickey-Fuller = -9.6003, Lag order = 0,

p-value = 0.01

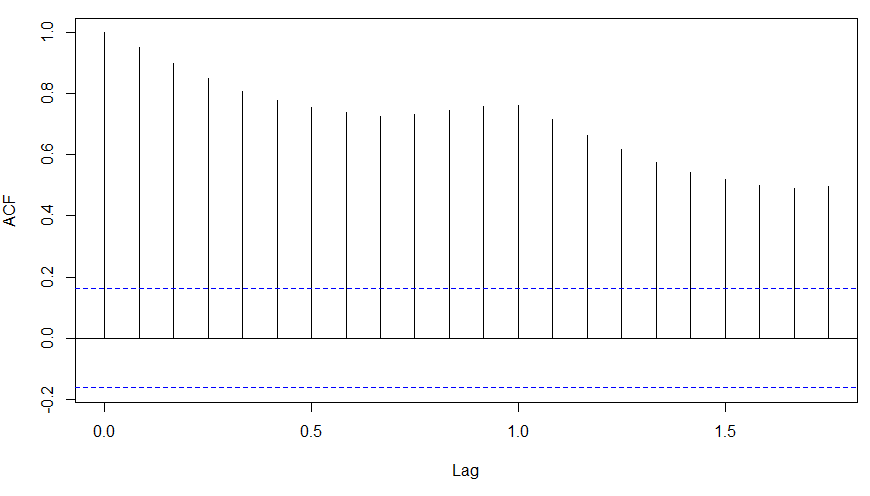
alternative hypothesis: stationary

We see that the series is stationary enough to do any kind of time series modelling.

Next step is to find the right parameters to be used in the ARIMA model. We already know that the ‘d’ component is 1 as we need 1 difference to make the series stationary. We do this using the Correlation plots. Following are the ACF plots for the series :

#ACF Plots

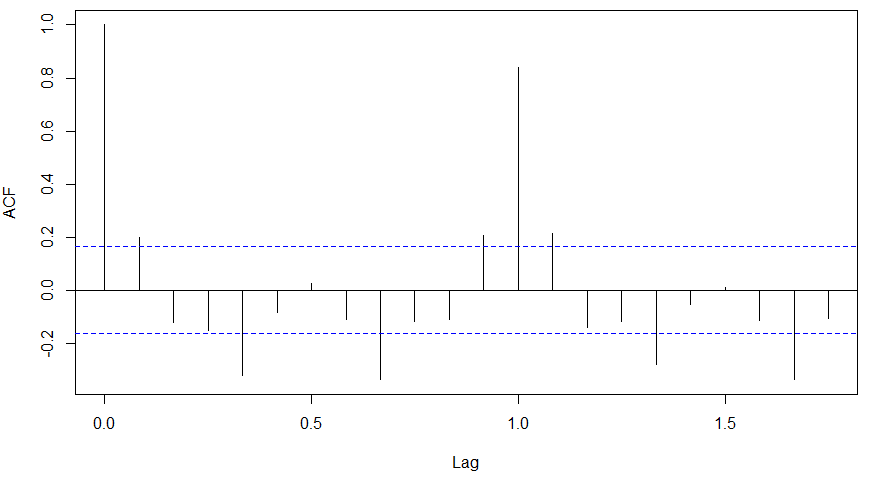
acf(log(AirPassengers))

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/ACF_original.png)

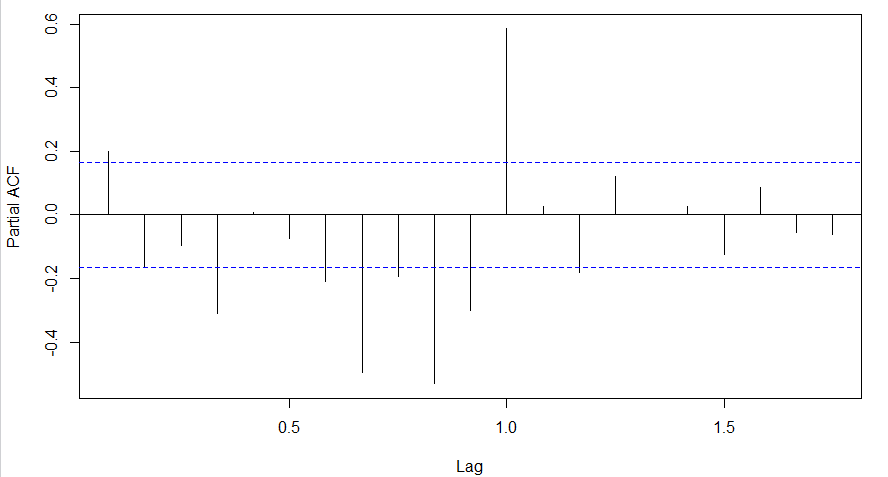
### What do you see in the chart shown above?

Clearly, the decay of ACF chart is very slow, which means that the population is not stationary. We have already discussed above that we now intend to regress on the difference of logs rather than log directly. Let’s see how ACF and PACF curve come out after regressing on the difference.

acf(diff(log(AirPassengers)))

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/ACF-diff.png)

pacf(diff(log(AirPassengers)))

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/PACF-diff.png)

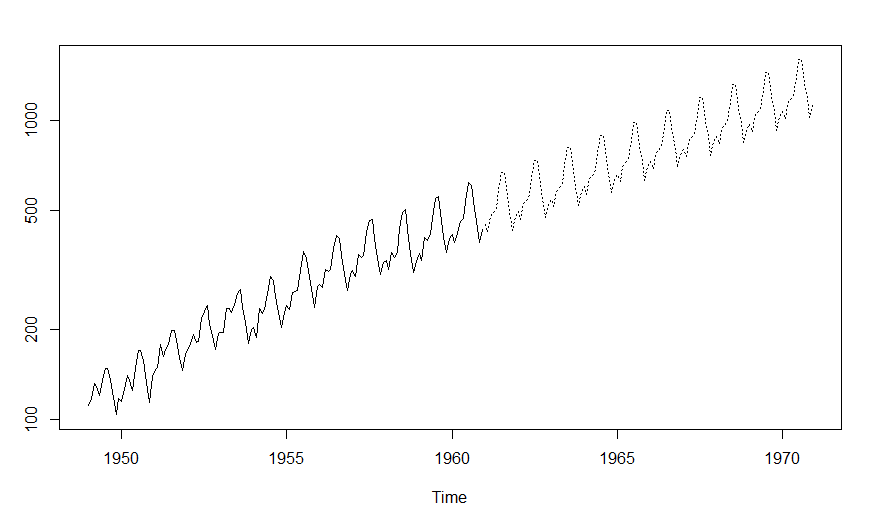
Clearly, ACF plot cuts off after the first lag. Hence, we understood that value of p should be 0 as the ACF is the curve getting a cut off. While value of q should be 1 or 2. After a few iterations, we found that (0,1,1) as (p,d,q) comes out to be the combination with least AIC and BIC.

Let’s fit an ARIMA model and predict the future 10 years. Also, we will try fitting in a seasonal component in the ARIMA formulation. Then, we will visualize the prediction along with the training data. You can use the following code to do the same :

(fit <- arima(log(AirPassengers), c(0, 1, 1),seasonal = list(order = c(0, 1, 1), period = 12)))

pred <- predict(fit, n.ahead = 10\*12)

ts.plot(AirPassengers,2.718^pred$pred, log = "y", lty = c(1,3))

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/predictions.png)

## End Notes

With this, we come to this end of tutorial on Time Series Modeling. I hope this will help you to improve your knowledge to work on time based data. To reap maximum benefits out of this tutorial, I’d suggest you to practice these R codes side by side and check your progress.

Did you find the article useful? Share with us if you have done similar kind of analysis before. Do let us know your thoughts about this article in the box below.

### ****If you like what you just read & want to continue your analytics learning,****[subscribe to our emails](http://feedburner.google.com/fb/a/mailverify?uri=analyticsvidhya)****,****[follow us on twitter](http://twitter.com/analyticsvidhya)****or like our****[facebook page](http://facebook.com/analyticsvidhya)****.****

### Share this:

* [Click to share on LinkedIn (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=linkedin&nb=1)
* [Share on Facebook (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=facebook&nb=1)
* [Click to share on Google+ (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=google-plus-1&nb=1)
* [Click to share on Twitter (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=twitter&nb=1)
* [Click to share on Pocket (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=pocket&nb=1)
* [Click to share on Reddit (Opens in new window)](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?share=reddit&nb=1)

### Related

[](http://www.analyticsvidhya.com/blog/2015/03/framework-application-build-arima-model/)

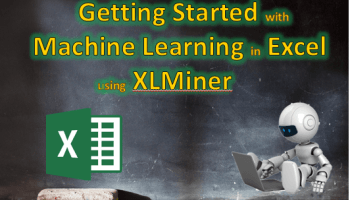
#### [Framework and Applications of ARIMA time series models](http://www.analyticsvidhya.com/blog/2015/03/framework-application-build-arima-model/)

In "Business Analytics"

[](http://www.analyticsvidhya.com/blog/2013/06/business-analytics-spectrum/)

#### [What is Business Analytics and which tools are used for analysis?](http://www.analyticsvidhya.com/blog/2013/06/business-analytics-spectrum/)

In "Big data"

[](http://www.analyticsvidhya.com/blog/2015/11/started-machine-learning-ms-excel-xl-miner/)

#### [Getting started with Machine Learning in MS Excel using XLMiner](http://www.analyticsvidhya.com/blog/2015/11/started-machine-learning-ms-excel-xl-miner/)

In "Business Analytics"

Tags: [ACF Plots](http://www.analyticsvidhya.com/blog/tag/acf-plots/), [ARIMA](http://www.analyticsvidhya.com/blog/tag/arima/), [ARIMA modeling](http://www.analyticsvidhya.com/blog/tag/arima-modeling/), [ARMA modeling](http://www.analyticsvidhya.com/blog/tag/arma-modeling/), [Dickey-Fuller test](http://www.analyticsvidhya.com/blog/tag/dickey-fuller-test/), [forecasting](http://www.analyticsvidhya.com/blog/tag/forecasting/), [Time Series](http://www.analyticsvidhya.com/blog/tag/time-series/)

[Next Article](http://www.analyticsvidhya.com/blog/2015/12/top-business-analytics-programs-india-2015-2016/)

[Top Business Analytics Programs in India (2015 – 16)](http://www.analyticsvidhya.com/blog/2015/12/top-business-analytics-programs-india-2015-2016/)

[Previous Article](http://www.analyticsvidhya.com/blog/2015/12/10-machine-learning-algorithms-explained-army-soldier/)

[10 Machine Learning Algorithms Explained to an ‘Army Soldier’](http://www.analyticsvidhya.com/blog/2015/12/10-machine-learning-algorithms-explained-army-soldier/)

[](http://www.analyticsvidhya.com/blog/author/tavish1/)

Author

[Tavish Srivastava](http://www.analyticsvidhya.com/blog/author/tavish1/)

I am Tavish Srivastava, a post graduate from IIT Madras in Mechanical Engineering. I have more than two years of work experience in Analytics. My experience ranges from hands on analytics in a developing country like India to convince banking partners with analytical solution in matured market like US. For last two and a half years I have contributed to various sales strategies, marketing strategies and Recruitment strategies in both Insurance and Banking industry.

### 16 Comments

* Dr.D.K.Samuel says:

[December 16, 2015 at 5:27 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102002)

Really useful. Please also write on how to make weather data into a times series for further analysis in R

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102002#respond)

* [*Dr Sahul Bharti*](http://alphabetastats.com) says:

[December 16, 2015 at 5:56 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102005)

Hi  
I am a medical specialist (MD Pediatrics) with further training in research and statistics (Panjab University, Chandigarh). In our medical settings, time series data are often seen in ICU and anesthesia related research where patients are continuously monitored for days or even weeks generating such data. Frankly speaking, your article has clearly decoded this arcane process of time series analysis with quite wonderful insight into its practical relevance. Fabulous article Mr Tavish, kindly write more about ARIMA modelling.  
Thanks a lot  
Dr Sahul Bharti

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102005#respond)

* baseer says:

[December 16, 2015 at 6:37 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102006)

great article to start with timeseries mod

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102006#respond)

* Shivam Bansal says:

[December 16, 2015 at 6:52 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102008)

Awesome Tutorial.

Big fan of you Tavish, your articles are really great. Explanations in beautiful manner.

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102008#respond)

* Ankur Bhargava says:

[December 16, 2015 at 7:19 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102011)

Please elucidate on PACF part of MA series.  
Thanks

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102011#respond)

* + Tavish Srivastava says:

[December 16, 2015 at 9:04 am](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102014)

PACF is not really required for MA models, as the degree of MA can be found from ACF directly.

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102014#respond)

* Hugo says:

[December 16, 2015 at 1:02 pm](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102019)

Hi Tavish.  
First off all, congratulations on your work around here. It’s been very useful. Thank you  
I a doubt and i hope that you can help me

I performed a Dickey-Fuller test on both series ; AirPassengers and diff( log(AirPassengers))

Here the results:

Augmented Dickey-Fuller Test  
data: diff(log(AirPassengers))  
Dickey-Fuller = -9.6003, Lag order = 0, p-value = 0.01  
alternative hypothesis: stationary

and

Augmented Dickey-Fuller Test

data: diff(log(AirPassengers))  
Dickey-Fuller = -9.6003, Lag order = 0, p-value = 0.01  
alternative hypothesis: stationary

In both tests i got a small p-value that allows me to reject the non stationary hypothesis. Am I right?

If so, the first series is already stationary??  
This means that if i had performed a stationary test on the original series had move on to the next step.

Thank you in advance.

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102019#respond)

* + Hugo says:

[December 16, 2015 at 1:05 pm](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102020)

Now with the right results .

Augmented Dickey-Fuller Test

data: AirPassengers  
Dickey-Fuller = -4.6392, Lag order = 0, p-value = 0.01  
alternative hypothesis: stationary

Augmented Dickey-Fuller Test

data: diff(log(AirPassengers))  
Dickey-Fuller = -9.6003, Lag order = 0, p-value = 0.01  
alternative hypothesis: stationary

[Reply](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/?replytocom=102020#respond)

* + - Ram says:

[December 18, 2015 at 10:45 pm](http://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/#comment-102163)

@Hugo,

Yes, the adf.test(AirPassengers) indicates that the series is stationary. This is a bit misleading.

Reason: This test first does a de-trend on the series, (ie., removes the trend component), then checks for stationarity. Hence it flags the series as stationary.

There is another test in package fUnitRoots. Please try this code:

## Start  
install.packages(“fUnitRoots”) # If you already have installed this package, you can omit this line  
library(fUnitRoots);  
adfTest(AirPassengers);  
adfTest(log(AirPassengers));  
adfTest(diff(AirPassengers));  
## End

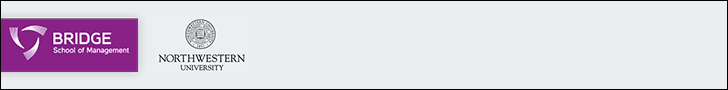
Hope this helps..

# A Comprehensive guide to Data Exploration

[Business Analytics](http://www.analyticsvidhya.com/blog/category/business-analytics/)

SHARE

[Sunil Ray](http://www.analyticsvidhya.com/blog/author/sunil-ray/) , January 10, 2016 / [17](http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#comments)

[](http://admissions.bridgesom.com/pba-new/?utm_source=AV&utm_medium=Banner&utm_campaign=AVBanner)

## Introduction

There are no shortcuts for data exploration. If you are in a state of mind, that machine learning can sail you away from every data storm, trust me, it won’t. After some point of time, you’ll realize that you are struggling at improving model’s accuracy. In such situation, data exploration techniques will come to your rescue.

I can confidently say this, because I’ve been through such situations, a lot.

I have been a Business Analytics professional for close to three years now. In my initial days, one of my mentor suggested me to spend significant time on exploration and analyzing data. Following his advice has served me well.

I’ve created this tutorial to help you understand the underlying techniques of data exploration. As always, I’ve tried my best to explain these concepts in the simplest manner. For better understanding, I’ve taken up few examples to demonstrate the complicated concepts.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2016/01/de.jpg)

## Table of Contents

1. [**Steps of Data Exploration and Preparation**](http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#one)
2. [**Missing Value Treatment**](http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#two)
   * Why missing value treatment is required ?
   * Why data has missing values?
   * Which are the methods to treat missing value ?
3. [**Techniques of Outlier Detection and Treatment**](http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#three)
   * What is an outlier?
   * What are the types of outliers ?
   * What are the causes of outliers ?
   * What is the impact of outliers on dataset ?
   * How to detect outlier ?
   * How to remove outlier ?
4. [**The Art of Feature Engineering**](http://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/#four)
   * What is Feature Engineering ?
   * What is the process of Feature Engineering ?
   * What is Variable Transformation ?
   * When should we use variable transformation ?
   * What are the common methods of variable transformation ?
   * What is feature variable creation and its benefits ?

### Let’s get started.

## 1. Steps of Data Exploration and Preparation

Remember the quality of your inputs decide the quality of your output. So, once you have got your business hypothesis ready, it makes sense to spend lot of time and efforts here. With my personal estimate, data exploration, cleaning and preparation can take up to 70% of your total project time.

Below are the steps involved to understand, clean and prepare your data for building your predictive model:

1. Variable Identification
2. Univariate Analysis
3. Bi-variate Analysis
4. Missing values treatment
5. Outlier treatment
6. Variable transformation
7. Variable creation

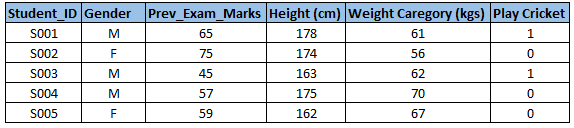
Finally, we will need to iterate over steps 4 – 7 multiple times before we come up with our refined model.

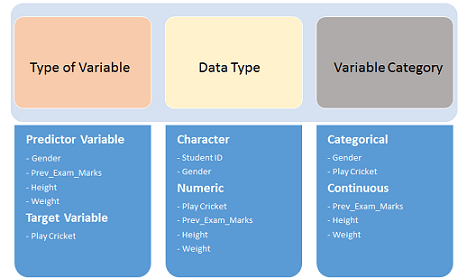
Let’s now study each stage in  detail:-

### Variable Identification

First, identify **Predictor** (Input) and **Target** (output) variables. Next, identify the data type and category of the variables.

Let’s understand this step more clearly by taking an example.

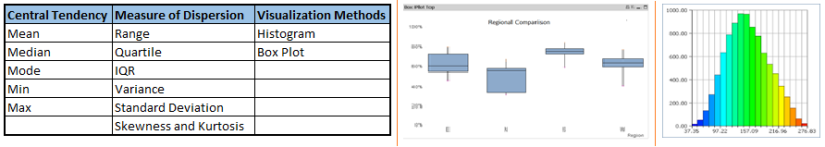
Example:- Suppose, we want to predict, whether the students will play cricket or not (refer below data set). Here you need to identify predictor variables, target variable, data type of variables and category of variables.[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_11.png)Below, the variables have been defined in different category:

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_2.png)

### Univariate Analysis

At this stage, we explore variables one by one. Method to perform uni-variate analysis will depend on whether the variable type is categorical or continuous. Let’s look at these methods and statistical measures for categorical and continuous variables individually:

**Continuous Variables:-** In case of continuous variables, we need to understand the central tendency and spread of the variable. These are measured using various statistical metrics visualization methods as shown below:

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_31.png)**Note:** Univariate analysis is also used to highlight missing and outlier values. In the upcoming part of this series, we will look at methods to handle missing and outlier values. To know more about these methods, you can refer course [descriptive statistics from Udacity](https://www.udacity.com/course/ud827).

**Categorical Variables:-** For categorical variables, we’ll use frequency table to understand distribution of each category. We can also read as percentage of values under each category. It can be be measured using two metrics, **Count** and **Count%** against each category. Bar chart can be used as visualization.

### Bi-variate Analysis

Bi-variate Analysis finds out the relationship between two variables. Here, we look for association and disassociation between variables at a pre-defined significance level. We can perform bi-variate analysis for any combination of categorical and continuous variables. The combination can be: Categorical & Categorical, Categorical & Continuous and Continuous & Continuous. Different methods are used to tackle these combinations during analysis process.

Let’s understand the possible combinations in detail:

**Continuous & Continuous:** While doing bi-variate analysis between two continuous variables, we should look at scatter plot. It is a nifty way to find out the relationship between two variables. The pattern of scatter plot indicates the relationship between variables. The relationship can be linear or non-linear.

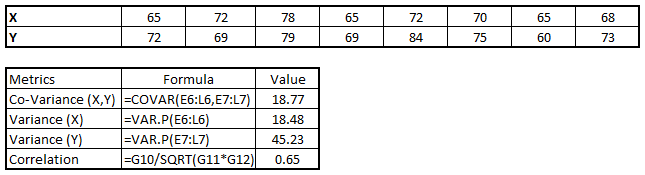
Scatter plot shows the relationship between two variable but does not indicates the strength of relationship amongst them. To find the strength of the relationship, we use Correlation. Correlation varies between -1 and +1.

* -1: perfect negative linear correlation
* +1:perfect positive linear correlation and
* 0: No correlation

Correlation can be derived using following formula:

**Correlation = Covariance(X,Y) / SQRT( Var(X)\* Var(Y))**

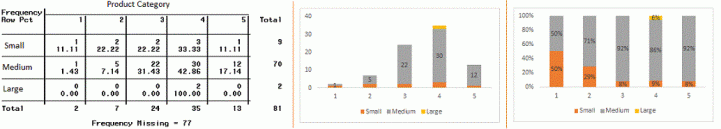
Various tools have function or functionality to identify correlation between variables. In Excel, function CORREL() is used to return the correlation between two variables and SAS uses procedure PROC CORR to identify the correlation. These function returns Pearson Correlation value to identify the relationship between two variables:

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_51.png)

In above example, we have good positive relationship(0.65) between two variables X and Y.

**Categorical & Categorical:**To find the relationship between two categorical variables, we can use following methods:

* **Two-way table:** We can start analyzing the relationship by creating a two-way table of count and count%. The rows represents the category of one variable and the columns represent the categories of the other variable. We show count or count% of observations available in each combination of row and column categories.
* **Stacked Column Chart:** This method is more of a visual form of Two-way table.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_6.gif)

* **Chi-Square Test:** This test is used to derive the statistical significance of relationship between the variables. Also, it tests whether the evidence in the sample is strong enough to generalize that the relationship for a larger population as well. Chi-square is based on the difference between the expected and observed frequencies in one or more categories in the two-way table. It returns probability for the computed chi-square distribution with the degree of freedom.

Probability of 0: It indicates that both categorical variable are dependent

Probability of 1: It shows that both variables are independent.

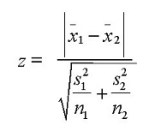
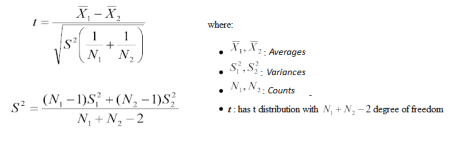
Probability less than 0.05: It indicates that the relationship between the variables is significant at 95% confidence. The chi-square test statistic for a test of independence of two categorical variables is found by:

[Data Exploration, Chi Square, Business Analytics](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_7.png)where *O* represents the observed frequency. *E* is the expected frequency under the null hypothesis and computed by:  
[Data Exploration, Chi Square, Business Analytics](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_exploration_8.png)  
From previous two-way table, the expected count for product category 1 to be of small size is  0.22. It is derived by taking the row total for Size (9) times the column total for Product category (2) then dividing by the sample size (81). This is procedure is conducted for each cell. Statistical Measures used to analyze the power of relationship are:

* Cramer’s V for Nominal Categorical Variable
* Mantel-Haenszed Chi-Square for ordinal categorical variable.

Different data science language and tools have specific methods to perform chi-square test. In SAS, we can use **Chisq** as an option with **Proc freq** to perform this test.

**Categorical & Continuous:** While exploring relation between categorical and continuous variables, we can draw box plots for each level of categorical variables. If levels are small in number, it will not show the statistical significance. To look at the statistical significance we can perform Z-test, T-test or ANOVA.

* **Z-Test/ T-Test:-** Either test assess whether mean of two groups are statistically different from each other or not.[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/ztestformula1.jpg)If the probability of Z is small then the difference of two averages is more significant. The T-test is very similar to Z-test but it is used when number of observation for both categories is less than 30.  
  [](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/ttest.png)
* **ANOVA:-** It assesses whether the average of more than two groups is statistically different.

**Example:** Suppose, we want to test the effect of five different exercises. For this, we recruit 20 men and assign one type of exercise to 4 men (5 groups). Their weights are recorded after a few weeks. We need to find out whether the effect of these exercises on them is significantly different or not. This can be done by comparing the weights of the 5 groups of 4 men each.

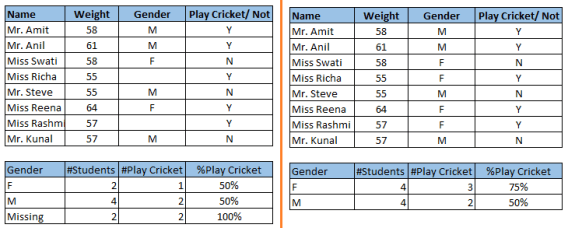
Till here, we have understood the first three stages of Data Exploration, Variable Identification, Uni-Variate and Bi-Variate analysis. We also looked at various statistical and visual methods to identify the relationship between variables.

Now, we will look at the methods of Missing values Treatment. More importantly, we will also look at why missing values occur in our data and why treating them is necessary.

## 2. Missing Value Treatment

### Why missing values treatment is required?

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behavior and relationship with other variables correctly. It can lead to wrong prediction or classification.

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_Exploration_2_11.png)

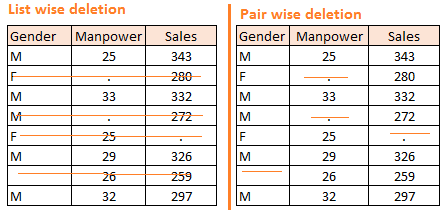
Notice the missing values in the image shown above: In the left scenario, we have not treated missing values. The inference from this data set is that the chances of playing cricket by males is higher than females. On the other hand, if you look at the second table, which shows data after treatment of missing values (based on gender), we can see that females have higher chances of playing cricket compared to males.

### Why my data has missing values?

We looked at the importance of treatment of missing values in a dataset. Now, let’s identify the reasons for occurrence of these missing values. They may occur at two stages:

1. **Data Extraction**: It is possible that there are problems with extraction process. In such cases, we should double-check for correct data with data guardians. Some hashing procedures can also be used to make sure data extraction is correct. Errors at data extraction stage are typically easy to find and can be corrected easily as well.
2. **Data collection**: These errors occur at time of data collection and are harder to correct. They can be categorized in four types:
   * **Missing completely at random:** This is a case when the probability of missing variable is same for all observations. For example: respondents of data collection process decide that they will declare their earning after tossing a fair coin. If an head occurs, respondent declares his / her earnings & vice versa. Here each observation has equal chance of missing value.
   * **Missing at random:** This is a case when variable is missing at random and missing ratio varies for different values / level of other input variables. For example: We are collecting data for age and female has higher missing value compare to male.
   * **Missing that depends on unobserved predictors:** This is a case when the missing values are not random and are related to the unobserved input variable. For example: In a medical study, if a particular diagnostic causes discomfort, then there is higher chance of drop out from the study. This missing value is not at random unless we have included “discomfort” as an input variable for all patients.
   * **Missing that depends on the missing value itself:** This is a case when the probability of missing value is directly correlated with missing value itself. For example: People with higher or lower income are likely to provide non-response to their earning.

### Which are the methods to treat missing values ?

1. **Deletion:** It is of two types: List Wise Deletion and Pair Wise Deletion.
   * In list wise deletion, we delete observations where any of the variable is missing. Simplicity is one of the major advantage of this method, but this method reduces the power of model because it reduces the sample size.
   * In pair wise deletion, we perform analysis with all cases in which the variables of interest are present. Advantage of this method is, it keeps as many cases available for analysis. One of the disadvantage of this method, it uses different sample size for different variables.  
     [](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Data_Exploration_2_2.png)
   * Deletion methods are used when the nature of missing data is “**Missing completely at random**” else non random missing values can bias the model output.
2. **Mean/ Mode/ Median Imputation:**Imputation is a method to fill in the missing values with estimated ones. The objective is to employ known relationships that can be identified in the valid values of the data set to assist in estimating the missing values. Mean / Mode / Median imputation is one of the most frequently used methods. It consists of replacing the missing data for a given attribute by the mean or median (quantitative attribute) or mode (qualitative attribute) of all known values of that variable. It can be of two types:-
   * **Generalized Imputation:** In this case, we calculate the mean or median for all non missing values of that variable then replace missing value with mean or median. Like in above table, variable “**Manpower”** is missing so we take average of all non missing values of “**Manpower”**  (**28.33**) and then replace missing value with it.
   * **Similar case Imputation:** In this case, we calculate average for gender “**Male”**(29.75) and “**Female**” (25) individually of non missing values then replace the missing value based on gender. For “**Male**“, we will replace missing values of manpower with 29.75 and for “**Female**” with 25.
3. **Prediction Model**:  Prediction model is one of the sophisticated method for handling missing data. Here, we create a predictive model to estimate values that will substitute the missing data.  In this case, we divide our data set into two sets: One set with no missing values for the variable and another one with missing values. First data set become training data set of the model while second data set with missing values is test data set and variable with missing values is treated as target variable. Next, we create a model to predict target variable based on other attributes of the training data set and populate missing values of test data set.We can use regression, ANOVA, Logistic regression and various modeling technique to perform this. There are 2 drawbacks for this approach:
   * The model estimated values are usually more well-behaved than the true values
   * If there are no relationships with attributes in the data set and the attribute with missing values, then the model will not be precise for estimating missing values.
4. **KNN Imputation:** In this method of imputation, the missing values of an attribute are imputed using the given number of attributes that are most similar to the attribute whose values are missing. The similarity of two attributes is determined using a distance function. It is also known to have certain advantage & disadvantages.
   * **Advantages:**
     + k-nearest neighbour can predict both qualitative & quantitative attributes
     + Creation of predictive model for each attribute with missing data is not required
     + Attributes with multiple missing values can be easily treated
     + Correlation structure of the data is taken into consideration
   * **Disadvantage:**
     + KNN algorithm is very time-consuming in analyzing large database. It searches through all the dataset looking for the most similar instances.
     + Choice of k-value is very critical. Higher value of k would include attributes which are significantly different from what we need whereas lower value of k implies missing out of significant attributes.

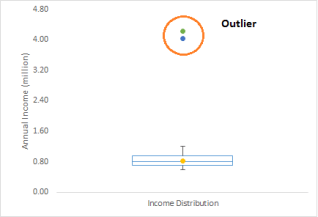
After dealing with missing values, the next task is to deal with outliers. Often, we tend to neglect outliers while building models. This is a discouraging practice. Outliers tend to make your data skewed and reduces accuracy. Let’s learn more about outlier treatment.

## 3. Techniques of Outlier Detection and Treatment

### What is an Outlier?

Outlier is a commonly used terminology by analysts and data scientists as it needs close attention else it can result in wildly wrong estimations. Simply speaking, Outlier is an observation that appears far away and diverges from an overall pattern in a sample.

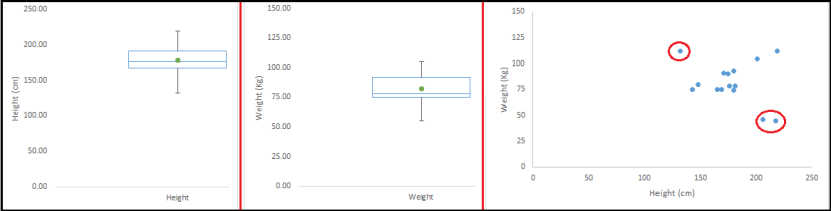
Let’s take an example, we do customer profiling and find out that the average annual income of customers is $0.8 million. But, there are two customers having annual income of $4 and $4.2 million. These two customers annual income is much higher than rest of the population. These two observations will be seen as Outliers.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier.png)

### What are the types of Outliers?

Outlier can be of two types: **Univariate** and **Multivariate**. Above, we have discussed the example of univariate outlier. These outliers can be found when we look at distribution of a single variable. Multi-variate outliers are outliers in an n-dimensional space. In order to find them, you have to look at distributions in multi-dimensions.

Let us understand this with an example. Let us say we are understanding the relationship between height and weight. Below, we have univariate and bivariate distribution for Height, Weight. Take a look at the box plot. We do not have any outlier (above and below 1.5\*IQR, most common method). Now look at the scatter plot. Here, we have two values below and one above the average in a specific segment of weight and height.

[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier_21.png)

### What causes Outliers?

Whenever we come across outliers, the ideal way to tackle them is to find out the reason of having these outliers. The method to deal with them would then depend on the reason of their occurrence. Causes of outliers can be classified in two broad categories:

1. **Artificial (Error) / Non-natural**
2. **Natural**.

Let’s understand various types of outliers in more detail:

* **Data Entry Errors:-** Human errors such as errors caused during data collection, recording, or entry can cause outliers in data. For example: Annual income of a customer is $100,000. Accidentally, the data entry operator puts an additional zero in the figure. Now the income becomes $1,000,000 which is 10 times higher. Evidently, this will be the outlier value when compared with rest of the population.
* **Measurement Error:**It is the most common source of outliers. This is caused when the measurement instrument used turns out to be faulty. For example: There are 10 weighing machines. 9 of them are correct, 1 is faulty. Weight measured by people on the faulty machine will be higher / lower than the rest of people in the group. The weights measured on faulty machine can lead to outliers.
* **Experimental Error:** Another cause of outliers is experimental error. For example: In a 100m sprint of 7 runners, one runner missed out on concentrating on the ‘Go’ call which caused him to start late. Hence, this caused the runner’s run time to be more than other runners. His total run time can be an outlier.
* **Intentional Outlier*:***This is commonly found in self-reported measures that involves sensitive data. For example: Teens would typically under report the amount of alcohol that they consume. Only a fraction of them would report actual value. Here actual values might look like outliers because rest of the teens are under reporting the consumption.
* **Data Processing Error:** Whenever we perform data mining, we extract data from multiple sources. It is possible that some manipulation or extraction errors may lead to outliers in the dataset.
* **Sampling error:** For instance, we have to measure the height of athletes. By mistake, we include a few basketball players in the sample. This inclusion is likely to cause outliers in the dataset.
* **Natural Outlier:**When an outlier is not artificial (due to error), it is a natural outlier. For instance: In my last assignment with one of the renowned insurance company, I noticed that the performance of top 50 financial advisors was far higher than rest of the population. Surprisingly, it was not due to any error. Hence, whenever we perform any data mining activity with advisors, we used to treat this segment separately.

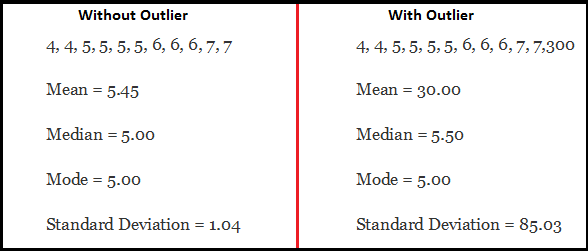
### What is the impact of Outliers on a dataset?

Outliers can drastically change the results of the data analysis and statistical modeling. There are numerous unfavourable impacts of outliers in the data set:

* It increases the error variance and reduces the power of statistical tests
* If the outliers are non-randomly distributed, they can decrease normality
* They can bias or influence estimates that may be of substantive interest
* They can also impact the basic assumption of Regression, ANOVA and other statistical model assumptions.

To understand the impact deeply, let’s take an example to check what happens to a data set with and without outliers in the data set.

**Example:**

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Outlier_31.png)

As you can see, data set with outliers has significantly different mean and standard deviation. In the first scenario, we will say that average is 5.45. But with the outlier, average soars to 30. This would change the estimate completely.

### How to detect Outliers?

Most commonly used method to detect outliers is visualization. We use various visualization methods, like **Box-plot**, **Histogram**, **Scatter Plot** (above, we have used box plot and scatter plot for visualization). Some analysts also various thumb rules to detect outliers. Some of them are:

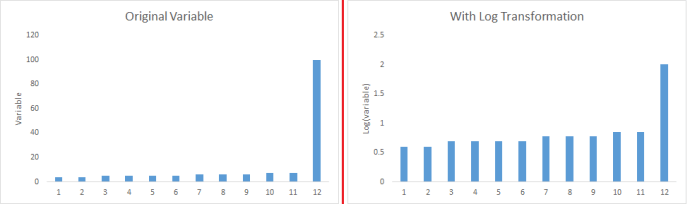
* Any value, which is beyond the range of -1.5 x IQR to 1.5 x IQR
* Use capping methods. Any value which out of range of 5th and 95th percentile can be considered as outlier
* Data points, three or more standard deviation away from mean are considered outlier
* Outlier detection is merely a special case of the examination of data for influential data points and it also depends on the business understanding
* Bivariate and multivariate outliers are typically measured using either an index of influence or leverage, or distance. Popular indices such as Mahalanobis’ distance and Cook’s *D* are frequently used to detect outliers.
* In SAS, we can use PROC Univariate, PROC SGPLOT. To identify outliers and influential observation, we also look at statistical measure like STUDENT, COOKD, RSTUDENT and others.

### How to remove Outliers?

Most of the ways to deal with outliers are similar to the methods of missing values like deleting observations, transforming them, binning them, treat them as a separate group, imputing values and other statistical methods. Here, we will discuss the common techniques used to deal with outliers:

**Deleting observations:**We delete outlier values if it is due to data entry error, data processing error or outlier observations are very small in numbers. We can also use trimming at both ends to remove outliers.

**Transforming and binning values:**Transforming variables can also eliminate outliers. Natural log of a value reduces the variation caused by extreme values. Binning is also a form of variable transformation. Decision Tree algorithm allows to deal with outliers well due to binning of variable. We can also use the process of assigning weights to different observations.

[](http://i2.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/02/Transformation_1.png)

**Imputing:**Like [imputation of missing values](http://www.analyticsvidhya.com/blog/2015/02/7-steps-data-exploration-preparation-building-model-part-2/), we can also impute outliers. We can use mean, median, mode imputation methods. Before imputing values, we should analyse if it is natural outlier or artificial. If it is artificial, we can go with imputing values. We can also use statistical model to predict values of outlier observation and after that we can impute it with predicted values.

**Treat separately:**If there are significant number of outliers, we should treat them separately in the statistical model. One of the approach is to treat both groups as two different groups and build individual model for both groups and then combine the output.

Till here, we have learnt about steps of data exploration, missing value treatment and techniques of outlier detection and treatment. These 3 stages will make your raw data better in terms of information availability and accuracy. Let’s now proceed to the final stage of data exploration. It is Feature Engineering.

## 4. The Art of Feature Engineering

### What is Feature Engineering?

Feature engineering is the science (and art) of extracting more information from existing data. You are not adding any new data here, but you are actually making the data you already have more useful.

For example, let’s say you are trying to predict foot fall in a shopping mall based on dates. If you try and use the dates directly, you may not be able to extract meaningful insights from the data. This is because the foot fall is less affected by the day of the month than it is by the day of the week. Now this information about day of week is implicit in your data. You need to bring it out to make your model better.

This exercising of bringing out information from data in known as feature engineering.

### What is the process of Feature Engineering ?

You perform feature engineering once you have completed the first 5 steps in data exploration – [Variable Identification, Univariate, Bivariate Analysis](http://www.analyticsvidhya.com/blog/2015/02/data-exploration-preparation-model/), [Missing Values Imputation](http://www.analyticsvidhya.com/blog/2015/02/7-steps-data-exploration-preparation-building-model-part-2/) and [Outliers Treatment](http://www.analyticsvidhya.com/blog/2015/02/outliers-detection-treatment-dataset/). Feature engineering itself can be divided in 2 steps:

* Variable transformation.
* Variable / Feature creation.

These two techniques are vital in data exploration and have a remarkable impact on the power of prediction. Let’s understand each of this step in more details.

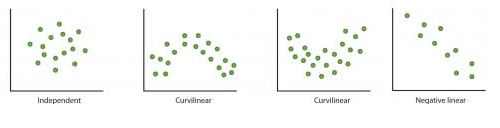
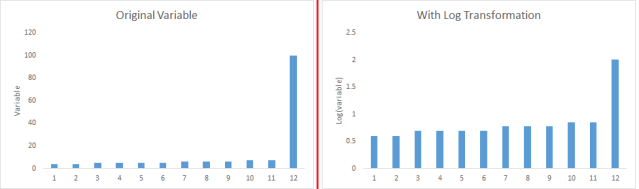
### What is Variable Transformation?

In data modelling, transformation refers to the replacement of a variable by a function. For instance, replacing a variable x by the square / cube root or logarithm x is a transformation. In other words, transformation is a process that changes the distribution or relationship of a variable with others.

Let’s look at the situations when  variable transformation is useful.

### When should we use Variable Transformation?

Below are the situations where variable transformation is a requisite:

* When we want to **change the scale** of a variable or standardize the values of a variable for better understanding. While this transformation is a must if you have data in different scales, this transformation does not change the shape of the variable distribution
* When we can **transform complex non-linear relationships into linear relationships**. Existence of a linear relationship between variables is easier to comprehend compared to a non-linear or curved relation. Transformation helps us to convert a non-linear relation into linear relation. Scatter plot can be used to find the relationship between two continuous variables. These transformations also improve the prediction. Log transformation is one of the commonly used transformation technique used in these situations.
* [](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/03/Relation.png)**Symmetric distribution is preferred over skewed distribution** as it is easier to interpret and generate inferences. Some modeling techniques requires normal distribution of variables. So, whenever we have a skewed distribution, we can use transformations which reduce skewness. For right skewed distribution, we take square / cube root or logarithm of variable and for left skewed, we take square / cube or exponential of variables.[](http://i0.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/03/Transformation_1.png)
* Variable Transformation is also done from an **implementation point of view** (Human involvement). Let’s understand it more clearly. In one of my project on employee performance, I found that age has direct correlation with performance of the employee i.e. higher the age, better the performance. From an implementation stand point, launching age based progamme might present implementation challenge. However, categorizing the sales agents in three age group buckets of <30 years, 30-45 years and >45  and then formulating three different strategies for each group is a judicious approach. This categorization technique is known as Binning of Variables.

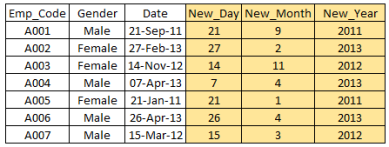
### What are the common methods of Variable Transformation?

There are various methods used to transform variables. As discussed, some of them include square root, cube root, logarithmic, binning, reciprocal and many others. Let’s look at these methods in detail by highlighting the pros and cons of these transformation methods.

* **Logarithm:** Log of a variable is a common transformation method used to change the shape of distribution of the variable on a distribution plot. It is generally used for reducing right skewness of variables. Though, It can’t be applied to zero or negative values as well.
* **Square / Cube root:** The square and cube root of a variable has a sound effect on variable distribution. However, it is not as significant as logarithmic transformation. Cube root has its own advantage. It can be applied to negative values including zero. Square root can be applied to positive values including zero.
* **Binning:** It is used to categorize variables. It is performed on original values, percentile or frequency. Decision of categorization technique is based on business understanding. For example, we can categorize income in three categories, namely: High, Average and Low.We can also perform co-variate binning which depends on the value of more than one variables.

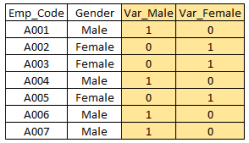
### What is Feature / Variable Creation & its Benefits?

Feature / Variable creation is a process to generate a new variables / features based on existing variable(s). For example, say, we have date(dd-mm-yy) as an input variable in a data set. We can generate new variables like day, month, year, week, weekday that may have better relationship with target variable. This step is used to highlight the hidden relationship in a variable:



There are various techniques to create new features. Let’s look at the some of the commonly used methods:

* **Creating derived variables:** This refers to creating new variables from existing variable(s) using set of functions or different methods. Let’s look at it through “[**Titanic – Kaggle competition**](https://www.kaggle.com/c/titanic-gettingStarted/data)”. In this data set, variable age has missing values. To predict missing values, we used the salutation (Master, Mr, Miss, Mrs) of name as a new variable. How do we decide which variable to create? Honestly, this depends on business understanding of the analyst, his curiosity and the set of hypothesis he might have about the problem. Methods such as taking log of variables, binning variables and other methods of variable transformation can also be used to create new variables.
* **Creating dummy variables:** One of the most common application of dummy variable is to convert categorical variable into numerical variables. Dummy variables are also called Indicator Variables. It is useful to take categorical variable as a predictor in statistical models.  Categorical variable can take values 0 and 1. Let’s take a variable ‘gender’. We can produce two variables, namely, “**Var\_Male**” with values 1 (Male) and 0 (No male) and “**Var\_Female**” with values 1 (Female) and 0 (No Female). We can also create dummy variables for more than two classes of a categorical variables with n or n-1 dummy variables.

[](http://i1.wp.com/www.analyticsvidhya.com/wp-content/uploads/2015/03/Dummy.png)

#### For further read, here is a [list of transformation / creation ideas](http://www.analyticsvidhya.com/blog/2013/11/simple-manipulations-extract-data/) which can be applied to your data.

## End Notes

As mentioned in the beginning, quality and efforts invested in data exploration differentiates a good model from a bad model.

This ends our guide on data exploration and preparation. In this comprehensive guide, we looked at the seven steps of data exploration in detail. The aim of this series was to provide an in depth and step by step guide to an extremely important process in data science.