Beyond Google Analytics.

Using C# to integrate Google Analytics with R statistical server

Juan Antonio Breña Moral

Abstract

Commercial Web sites are designed, planned and launched in many cases by Marketing Departments or Communication Agencies. Web Traffic is a variable very important to measure the impact of different iniciatives, and this measure is a mirror of economics results.

Google Analytics is a powerful tool to analyze/store Web Traffic data, but the information that you can see with their reports can be improved with others systems as R (Statistical Engine).

This paper has been divided in 2 parts. Firt part has been written to show how to use data exported in Xml format from Google Analytics to make forecasts using Time Series techniques with R. Second part is dedicated to show how to integrate C# with R to run your own Scripts.

Make Forecasts with Data exported from Google Analytics

In many articles in Internet, I have seen that somebody says that Google Analytics doesn't have features to integrate with other services. I have written this section to show how to integrate with a wonderful statistical engine as R using datasources in xml format that I have downloaded from Google analytics. R allow to improve the analysis from Data stored in Google Analytics.

Data Sources

In this article, I will use Web traffic stored from the web site: http://www.juanantonio.info in Google Analytics. To start to analyze this data, it is necessary to export this data. Google analytics allows you to export in several formats, but the best format is Xml with integration purpose.

To export only data from visit activity in any web site, the path in Google Analytics is the following:

Google Analytics > Users > User Trends > Visits

If you download data from main panel, you will download a heavy Xml file. It is a good practice download only data that you are going to use.

Figure 1: Visit Panel in Google Analytics

Select a period of time to analyze. I have choosen the following period of time:

Figure 2: Visit Panel in Google Analytics

When you export in Xml format you would see this window in your web Browser.

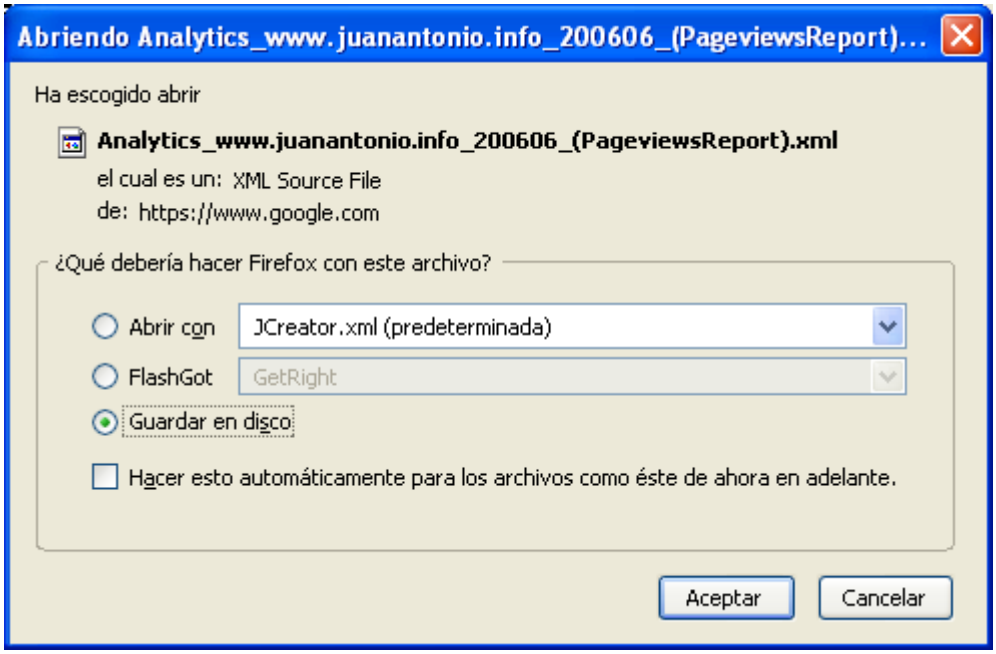


Figure 3: Visit Panel in Google Analytics

Once you have stored your data into your Computer, then it is time to start to analyze data with R

R Stat Engine

R is a language and environment for statistical computing and graphics. It is a GNU project which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R.

R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible. The S language is often the vehicle of choice for research in statistical methodology, and R provides an Open Source route to participation in that activity.

One of R's strengths is the ease with which well-designed publication-quality plots can be produced, including mathematical symbols and formulae where needed. Great care has been taken over the defaults for the minor design choices in graphics, but the user retains full control.

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form. It compiles and runs on a wide variety of UNIX platforms and similar systems (including FreeBSD and Linux), Windows and MacOS.

I usually use JGR as GUI for R.. In 2005, JGR won Chambers Awards, http://www.statcomputing.org/awards/jmc/winners.html

Figure 4: Launching JGR

Once you have downloaded your data from Google Analytics, it is time to analyze!

To process your Web Traffic Data with R, you will need to be installed into your R instance the following Libraries:

1. Normal Analysis
    a. library("nortest");
2. Time Series Analysis
    a. library("dyn");
    b. library("ArDec");
    c. library("forecast");
    d. library("fBasics");
    e. library("fCalendar");
    f. library("fSeries");
    g. library("tseries");
    h. XML Processing
3. library("XML");

The first step to analyze any data is get data in right format. I convert data in vectors.

```
##########################
# SETTINGS FOR ANALYSIS #
##########################

#Set Working Directory
WD <- "C:/DATOS/ESTADISTICA/R/SCRIPTS/ANALYTICS/";
JAB.ANALYTICS.CONFIG(WD);

#############
# LOAD XML #
#############

XML_FILE_PATH <- "DATA/Analytics_www.juanantonio.info_200606_(PageviewsReport).xml";
DATA <- JAB.ANALYTICS.LOAD_XML(XML_FILE_PATH);

################
# PROCESS XML #
################

VISITS <- JAB.ANALYTICS.GET_VISIT_ARRAY(DATA);
DATES <- JAB.ANALYTICS.GET_DATES_ARRAY(DATA);
```

Note: With new Google Analytics release, Xml protocol changed too. I had to update my R script.

Now you have in R your data. It is the moment to analyze it using Time Series Techniques.

Forecast your visits using R

The first step toforecast data is see the shape of your data to know if your data is stationary or not.

```
######################
# GENERATING TS DATA #
######################

#Generando una serie temporal
DATOS.TS <- ts(VISITS,start=5, frequency = frequency(VISITS), names=DATE_TRAFFIC);

JAB.TS.FIRST_VIEW(DATOS.TS);
```

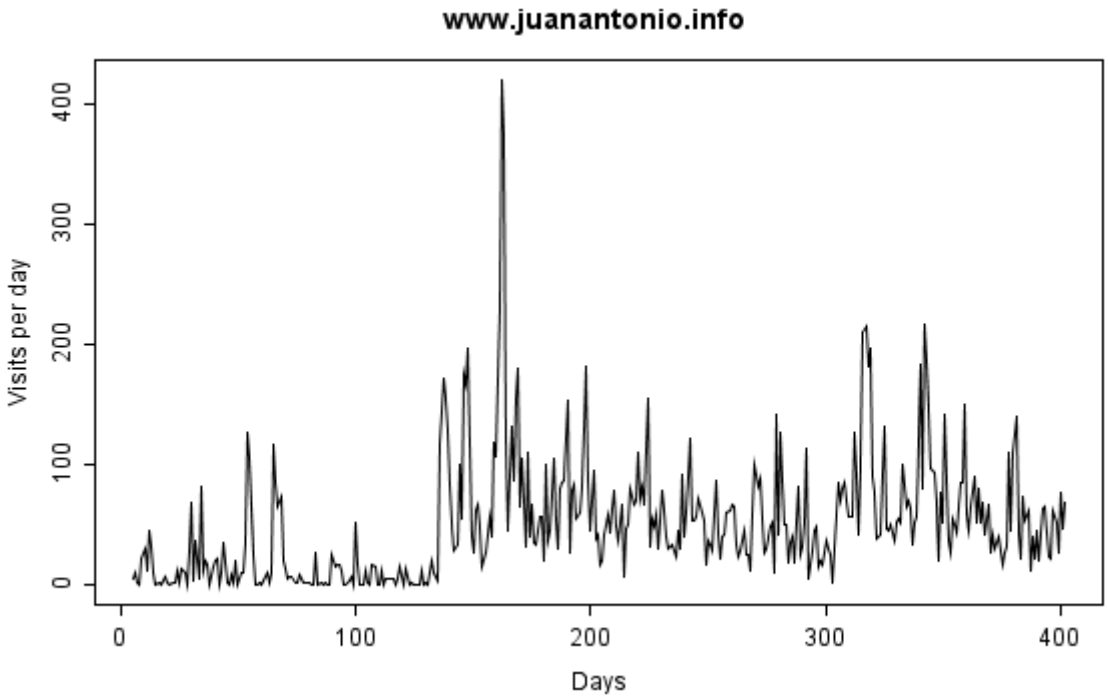


Figure 5: Web Traffic Evolution in juanantonio.info Web Site
period: 01/06/2006 - 30/06/2007

You will notice that this data is not stationary then we have to make data transformation to do this data stationary.

Besides I observe that Web traffic increased in October of 2006. I think that data used to develop a time series model should use data from 1st of October 2006- 30th of June 2007.

```
#SUBSET
VISITS.SUBSET <- VISITS[123:395];
VISITS.SUBSET;
DATOS.SUBSET.TS <- ts(VISITS.SUBSET,start=5, frequency = frequency(VISITS.SUBSET),
names=DATE_TRAFFIC);
JAB.TS.FIRST_VIEW(DATOS.SUBSET.TS);
```

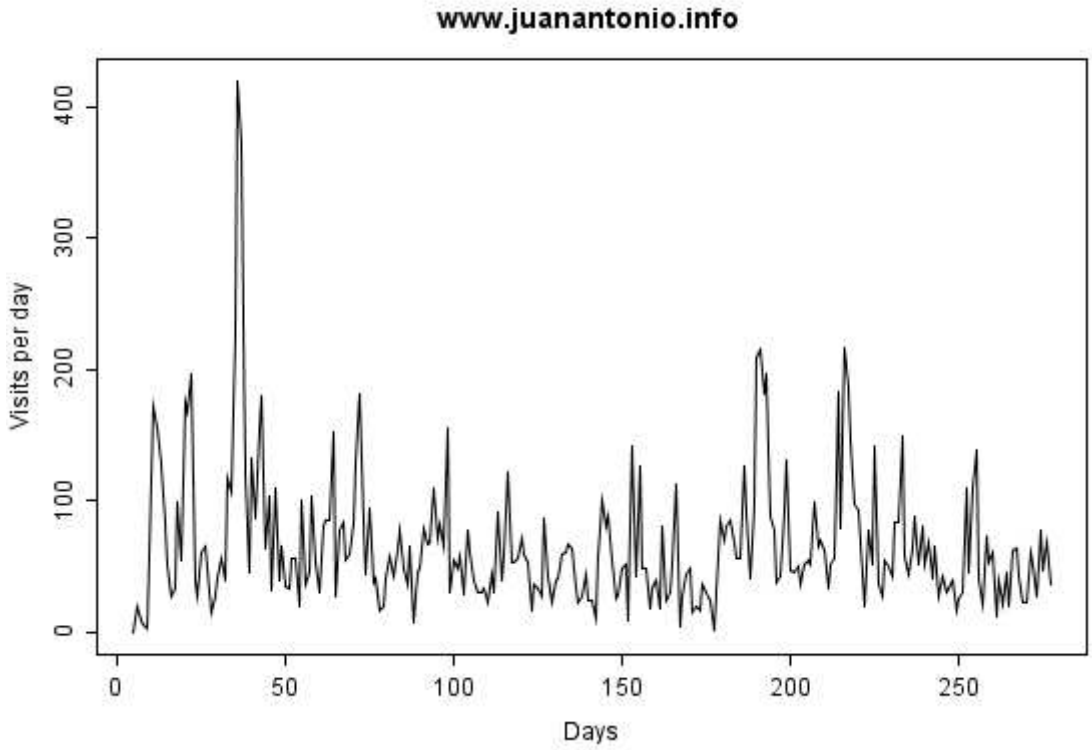


Figure 6: Web Traffic Evolution in juanantonio.info Web Site
period: 01/10/2006 - 30/06/2007

If you notice, data is stationary.

To determinate what is the best model for your data, it is neccesary analyze the following Graphics:

1. ACF Graph
2. PACF Graph

The Autocorrelation Functions
The techniques of model identification which are most commonly used were propounded originally by Box and Jenkins (1972). Their basic tools were the sample autocorrelation function and the partial autocorrelation function. We shall describe these functions and their use separately from the spectral density function which ought, perhaps, to be used more often in selecting models. The fact that spectral density function is often overlooked is probably due to an unfamiliarity with frequency-domain analysis on the part of many model builders.

Autocorrelation function (ACF).
Given a sample y0; y1; : : : ; yT¡1 of T observations, we de¯ne the sample autocorrelation function to be the sequence of values

$$r_\tau = c_\tau / c_0, \quad \tau = 0, 1, \ldots, T-1,$$

where in

$$c_\tau = \frac{1}{T}\sum_{t=\tau}^{T-1}(y_t - \bar{y})(y_{t-\tau} - \bar{y})$$

is the empirical autocovariance at lag *t* and *c0* is the sample variance. One should note that, as the value of the lag increases, the number of observations comprised in the empirical autocovariance diminishes until thenal element cT¡1 = T¡1(y0 ¡ ¹y)(yT¡1 ¡ ¹y) is reached which comprises only the rst and last mean-adjusted observations. In plotting the sequence {*rt*}, we shall omit the value of *r0* which is invariably unity. Moreover, in interpreting the plot, one should be wary of giving too much credence to the empirical autocorrelations at lag values which are signicantly high in relation to the size of the sample.

Partial autocorrelation function (PACF).

The sample partial autocorrelation *pt* at lag *t* is simply the correlation between the two sets of residuals obtained from regressing the elements yt and yt¡¿ on the set of intervening values y1; y2; : : : ; yt-1. The partial autocorrelation measures the dependence between yt and yt-1 after the effect of the intervening values has been removed. The sample partial autocorrelation pt is virtually the same quantity as the estimated coeffcient of lag *t* obtained by tting an autoregressive model of order ¿ to the data. Indeed, the diference between the two quantities vanishes as the sample size increases. The Durbin-Levinson algorithm provides an effcient way of computing the sequence *{pt}* of partial autocorrelations from the sequence of *{ct}* of autocovariances. It can be seen, in view of this algorithm, that the information in *{ct}* is equivalent to the information contained jointly in *{pt}* and c0. Therefore the sample autocorrelation function frtg and the sample partial autocorrelation function fptg are equivalent in terms of their information content.

```
JAB.TS.ACFS(DATOS.SUBSET.TS);
```

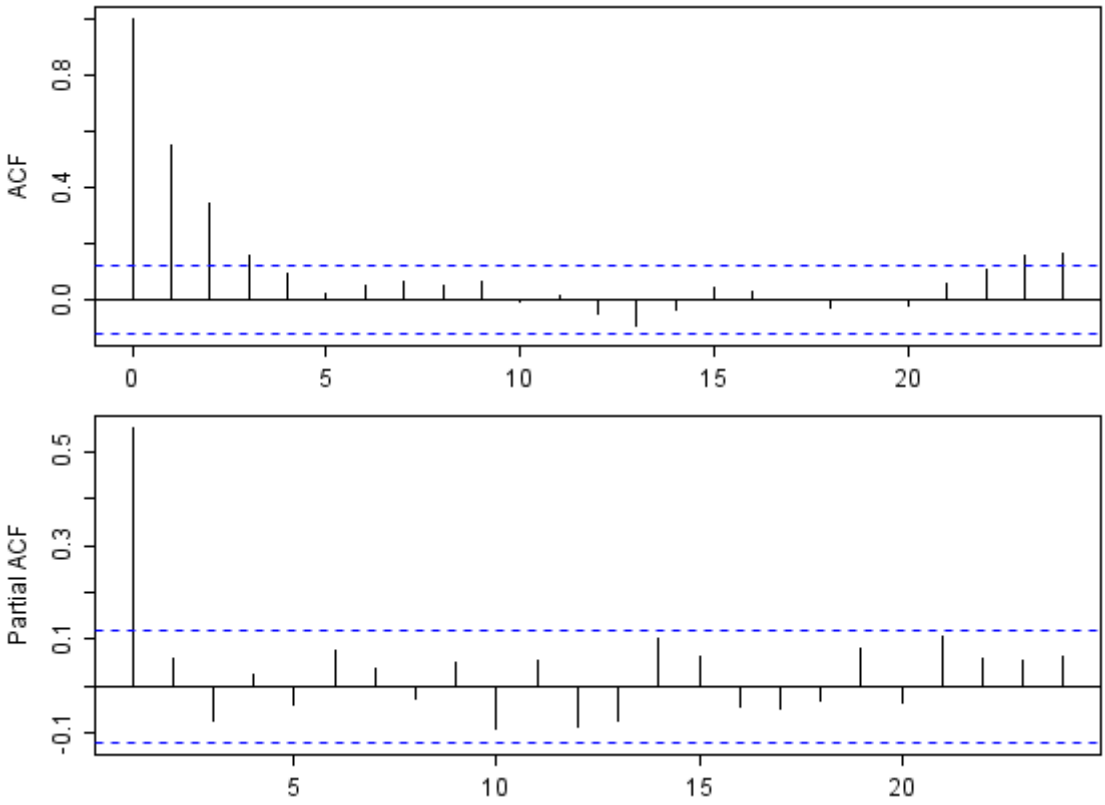


Figure 7: ACF & PAFC graphs

In previous graph, you notice that data is stationary. With ACF and PACF you reach to the same conclussion. I am going to confirm this hypothesis using a method to detect if data needs some difference:

```
JAB.TS.DIFFERENCES(DATOS.TS);
```

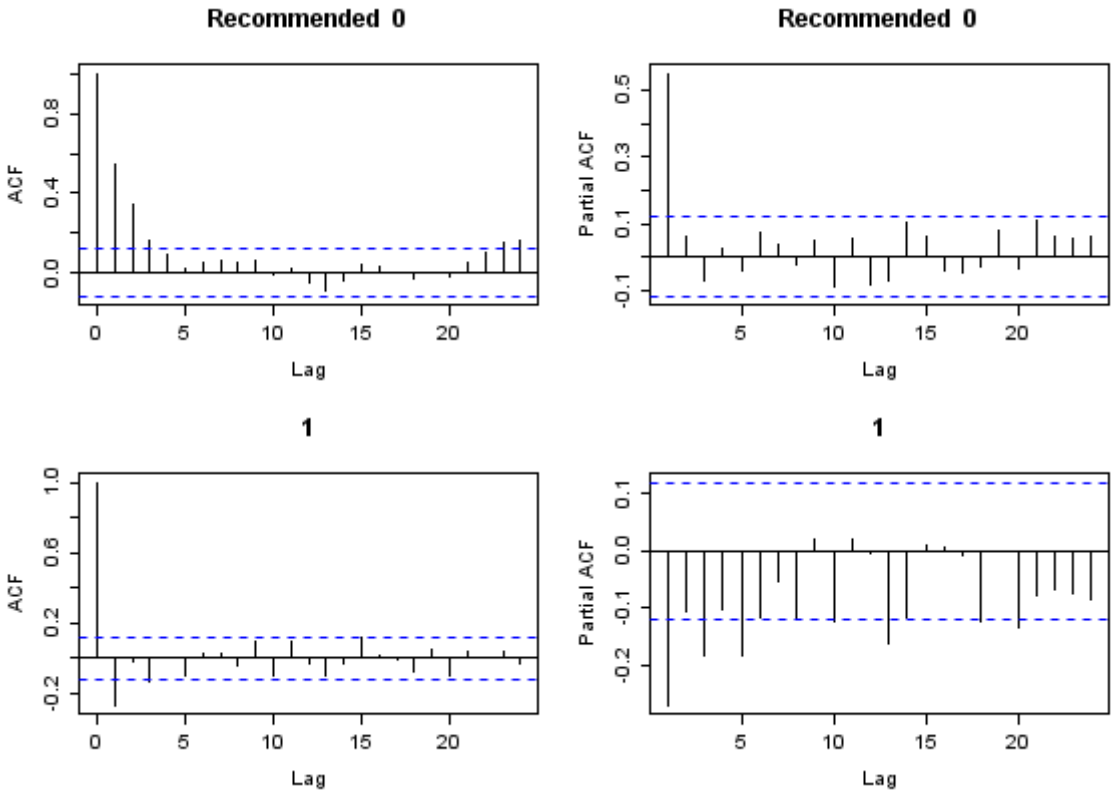


Figure 8: How to know the right number of differences

If you notice, data say you the right model to predict data. In this case, Web Traffic data will model using a AR model. This conclusion is due to analyze previous graphs because ACF has a many coeficcients not nulls but if you see PAFC you can see that has many coeficcients not nulls, then the best model to forecast is AR

The following Table is very useful to reach conclusions using previous graphs.

MODEL	ACF	FACF
AR(p)	Many coefficients are not null	First coefficients are not null
MA(q)	First coefficients are not null	Many coefficients are not null
ARMA(p,q)	Many coefficients are not null	Many coefficients are not null

The hypothesi to model data are the following:

1. AR(3)
2. AR(4)
3. ARIMA(?,?,?) *In R exist a function to calculate the best Arima Model*

AR, Autoregressive Models

The autoregressive model is one of a group of linear prediction formulas that attempt to predict an output of a system based on the previous outputs and inputs. The autoregressive (AR) models are used in time series analysis to describe stationary time series.

The current value of the series is a linear combination of the p most recent past values of itself plus an error term, which incorporates everything new in the series at time t that is not explained by the past values. This is like a multiple regressions model but is regressed not on independent variables, but on past values; hence the term "Autoregressive" is used.

An important guide to the properties of a time series is provided by a series of quantities called sample autocorrelation coefficients or serial correlation coefficient, which measures the correlation between observations at different distances apart. These coefficients often provide insight into the probability model which generated the data. The sample autocorrelation coefficient is similar to the ordinary correlation coefficient between two variables (x) and (y), except that it is applied to a single time series to see if successive observations are correlated.

Given (N) observations on discrete time series we can form (N - 1) pairs of observations. Regarding the first observation in each pair as one variable, and the second observation as a second variable, the correlation coefficient is called autocorrelation coefficient of order one.

A useful aid in interpreting a set of autocorrelation coefficients is a graph called a correlogram, and it is plotted against the lag(k); where is the autocorrelation coefficient at lag(k). A correlogram can be used to get a general understanding on the following aspects of our time series:

1. A random series: if a time series is completely random then for Large (N), will be approximately zero for all non-zero values of (k).
2. Short-term correlation: stationary series often exhibit short-term correlation characterized by a fairly large value of 2 or 3 more correlation coefficients which, while significantly greater than zero, tend to get successively smaller.
3. Non-stationary series: If a time series contains a trend, then the values of will not come to zero except for very large values of the lag.
4. Seasonal fluctuations: Common autoregressive models with seasonal fluctuations, of periods.

AR model is defined using the following ecuations:

$$y_k = \sum_{j=1}^{p} a_j x_{k+j} \quad k = p..1$$

$$y_k = \sum_{j=1}^{p} a_j x_{k-j} \quad k = p+1..N$$

In these equations x is the data series of length N and a is the autoregressive parameter array of order p. AutoSignal uses the positive sign (linear prediction) convention for the AR coefficients. The model is defined as reverse prediction for the first p values, and forward prediction for the remaining N-p values. This definition is used for all AR fit statistics, although it is not the model fitted in any of the AR linear least-squares procedures.

Selection Criteria: Several criteria may be specified for choosing a model format, given the simple and partial autocorrelation correlogram for a series:

1. If none of the simple autocorrelations is significantly different from zero, the series is essentially a random number or white-noise series, which is not amenable to autoregressive modeling.
2. If the simple autocorrelations decrease linearly, passing through zero to become negative, or if the simple autocorrelations exhibit a wave-like cyclical pattern, passing through zero several times, the series is not stationary; it must be differenced one or more times before it may be modeled with an autoregressive process.
3. If the simple autocorrelations exhibit seasonality; i.e., there are autocorrelation peaks every dozen or so (in monthly data) lags, the series is not stationary; it must be differenced with a gap approximately equal to the seasonal interval before further modeling.
4. If the simple autocorrelations decrease exponentially but approach zero gradually, while the partial autocorrelations are significantly non-zero through some small number of lags beyond which they are not significantly different from zero, the series should be modeled with an autoregressive process.
5. If the partial autocorrelations decrease exponentially but approach zero gradually, while the simple autocorrelations are significantly non-zero through some small number of lags beyond which they are not significantly different from zero, the series should be modeled with a moving average process.
6. If the partial and simple autocorrelations both converge upon zero for successively longer lags, but neither actually reaches zero after any particular lag, the series may be modeled by a combination of autoregressive and moving average process.

ARMA Model

Taking the AR model and the MA model, we get the ARMA model. The notation ARMA(p, q) refers to a model with p autoregressive terms and q moving average terms.

ARMA models are a method for modeling univariate time series.

$$X_t = \varepsilon_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}.$$

The power of ARMA models is that they can incorporate both autoregressive terms and moving average terms. The use of ARMA models was popularized by Box and Jenkins. Although both AR and MA models were previously known and used, Box and Jenkins provided a systematic approach for modeling both AR and MA terms in the model. ARMA models are also commonly known as Box-Jenkins models or ARIMA models.

ARMA models assume that the data are stationary, i.e. the data have constant location and scale. Trend can often be removed from a nonstationary series to achieve stationarity. Differencing is a common approach for removing trend. The first difference is defined as X(t) - X(t-1). In most cases, a single differencing is sufficient. However,

more than one differencing can be applied if necessary. You can also fit a linear or non-linear model to remove trend.

ARMA models can also incorporate seasonal terms (and seasonal differencing). See Box and Jenkins for the complete mathematical description of this model.

ARMA models typically require fairly long series (at least 50 points is recommended by some authors). Also, if the series is dominated by trend and seasonal components, a trend/seasonality/residual decomposition method may be preferred. Dataplot supports a SEASONAL LOWESS command for this type of decomposition.

ARIMA(p,d,q): ARIMA models are, in theory, the most general class of models for forecasting a time series which can be stationarized by transformations such as differencing and logging. In fact, the easiest way to think of ARIMA models is as fine-tuned versions of random-walk and random-trend models: the fine-tuning consists of adding lags of the differenced series and/or lags of the forecast errors to the prediction equation, as needed to remove any last traces of autocorrelation from the forecast errors.

Arima Model

The acronym ARIMA stands for "Auto-Regressive Integrated Moving Average." Lags of the differenced series appearing in the forecasting equation are called "auto-regressive" terms, lags of the forecast errors are called "moving average" terms, and a time series which needs to be differenced to be made stationary is said to be an "integrated" version of a stationary series. Random-walk and random-trend models, autoregressive models, and exponential smoothing models (i.e., exponential weighted moving averages) are all special cases of ARIMA models.

A nonseasonal ARIMA model is classified as an "ARIMA(p,d,q)" model, where:

- p is the number of autoregressive terms,
- d is the number of nonseasonal differences, and
- q is the number of lagged forecast errors in the prediction equation.

To identify the appropriate ARIMA model for a time series, you begin by identifying the order(s) of differencing needing to stationarize the series and remove the gross features of seasonality, perhaps in conjunction with a variance-stabilizing transformation such as logging or deflating. If you stop at this point and predict that the differenced series is constant, you have merely fitted a random walk or random trend model. (Recall that the random walk model predicts the first difference of the series to be constant, the seasonal random walk model predicts the seasonal difference to be constant, and the seasonal random trend model predicts the first difference of the seasonal difference to be constant--usually zero.) However, the best random walk or random trend model may still have autocorrelated errors, suggesting that additional factors of some kind are needed in the prediction equation.

To determinate what is the best model, I will use Akaike criterion.

Akaike information criterion

Akaike's An information criterion (AIC), developed by Hirotsugu Akaike in 1971 and proposed in Akaike (1974), is a measure of the goodness of fit of an estimated statistical

model. It is grounded in the concept of entropy. The AIC is an operational way of trading off the complexity of an estimated model against how well the model fits the data.

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

where k is the number of parameters, and L is the likelihood function.

Hypothesis Analysis

The hypothesis to model data are using AR, MA and ARIMA models.

Hypothesis 1: AR (3)

```
#Hypothesis 1: AR(3)
DATOS.TS.AR <- ar(DATOS.SUBSET.TS, aic = TRUE, order =3);
DATOS.TS.AR
DATOS.TS.AR$aic


Call:
ar(x = DATOS.SUBSET.TS, aic = TRUE, order.max = 3)

Coefficients:
     1
0.5485

Order selected 1  sigma^2 estimated as  1914
> DATOS.TS.AR$aic
         0          1          2          3
95.6879071  0.0000000  0.9978798  1.5177618
```

AIC value: 1.52

Hypothesis 2: AR(4)

```
#Hypothesis 2: AR(4)
DATOS.TS.AR <- ar(DATOS.SUBSET.TS, aic = TRUE, order =4);
DATOS.TS.AR
DATOS.TS.AR$aic

Call:
ar(x = DATOS.SUBSET.TS, aic = TRUE, order.max = 4)

Coefficients:
     1
0.5485

Order selected 1  sigma^2 estimated as  1914
> DATOS.TS.AR$aic
         0          1          2          3          4
95.6879071  0.0000000  0.9978798  1.5177618  3.3738844
```

AIC value: 3.37

Hypothesis 3: ARIMA(?,?,?)

```
DATOS.TS.BEST_ARIMA <- best.arima(DATOS.SUBSET.TS);
DATOS.TS.BEST_ARIMA;
```

```
Series: DATOS.SUBSET.TS
ARIMA(1,0,0) model

Coefficients:
         ar1  intercept
      0.5505    66.3907
s.e.  0.0505     5.8341

sigma^2 estimated as 1894:    log likelihood = -1417.66,    aic = 2841.32
```

AIC = 2841.32

If you notice, all hypothesis says that best way to model this data is using a AR(1)

Forecast your model

If I want to see the forecast for next 3 months (July, August & September) the results are:

```
JAB.TS.ARIMA.SHOW.FORECAST(DATOS.SUBSET.TS,DATOS.TS.BEST_ARIMA,90);
```

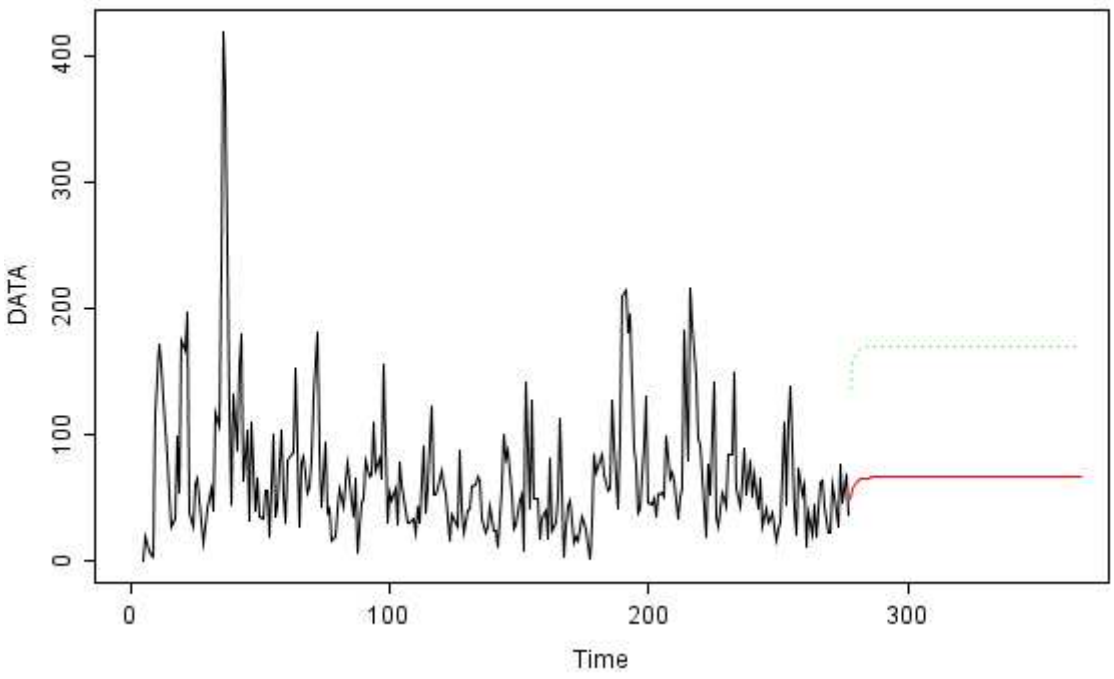


Figure 9: Forecast graph using Time Series Model

If i want to know the values:

```
LH.pred<-predict(DATOS.TS.BEST_ARIMA,n.ahead=90)
LH.pred

$pred
Time Series:
Start = 278
End = 369
Frequency = 1
 [1] 50.21254 57.48540 61.48875 63.69240 64.90541 65.57311 65.94064 66.14295
 [9] 66.25432 66.31561 66.34936 66.36793 66.37815 66.38378 66.38688 66.38859
```

```
[17] 66.38952 66.39004 66.39032 66.39048 66.39057 66.39061 66.39064 66.39066
[25] 66.39066 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[33] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[41] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[49] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[57] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[65] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[73] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[81] 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067 66.39067
[89] 66.39067 66.39067 66.39067 66.39067
```

Improve Web Analysis Process using R

When any manager has to increase the results in his Web Channel, the first Graph to analyze that he should see is a evolution graph in a interval of time and a cumulative plot. Currenly in main panel in Google Analytics we see a evolution graph but We dont have any cumulative plot to observe *growth rate*:

```
##################
# BUSINESS VIEW #
##################

JAB.ANALYTICS.VIEW(VISITS);
```

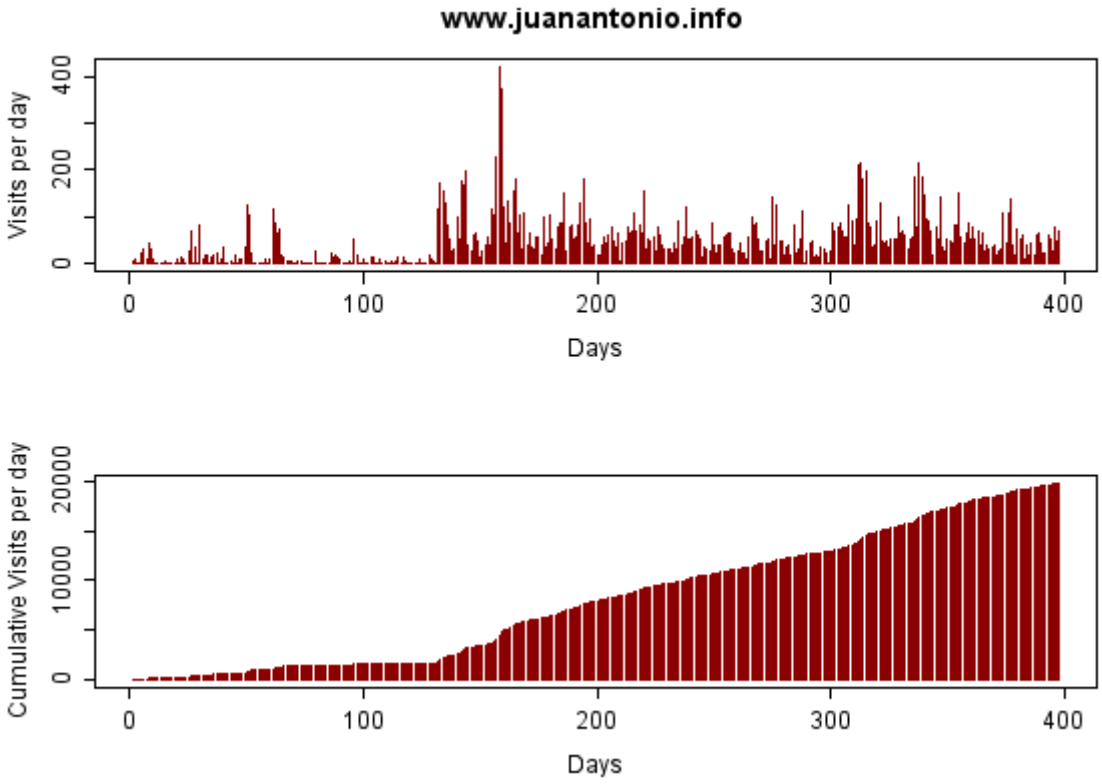


Figure 10: Esmeta Web Channel Business View

Integrating data into R, we can generate other graphics. If we want to generate a comparative between 2 years ,to know the growth rate, the solution is easy:

```
VISITS.MONTH <- JAB.ANALYTICS.GET_DATA_PER_MONTH(VISITS,6,13);
```

```
VISITS.MONTH.2006 <- VISITS.MONTH[1:12];
VISITS.MONTH.2007 <- VISITS.MONTH[13];
MONTHS <-
c("June","July","August","September","October","November","December","January","February
","March","April","May");
names(VISITS.MONTH.2006) <- MONTHS;
ACTUAL <- c(0,0,0,0,0,0,0,0,0,0,0,0);
VISITS.MONTH.2007 <- c(VISITS.MONTH.2007,ACTUAL);

#Ejemplo
TITLE = "www.juanantonio.info Comparative 2006-2007";
Y_LAB = "Visits per Month";
X_LAB = "Months";

plot(VISITS.MONTH.2006, type = "l", col = "red", main=TITLE, xlab= X_LAB, ylab= Y_LAB,
xlim=c(1,12))
lines(VISITS.MONTH.2007)
```

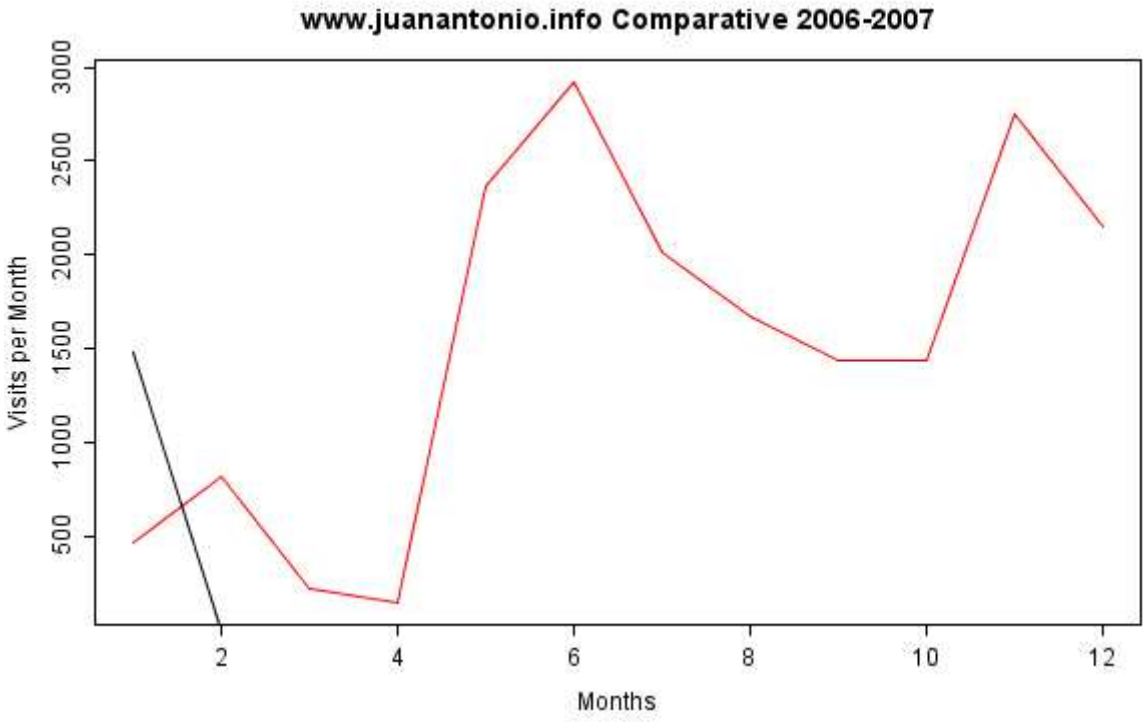


Figure 11: Comparative View between 2 years.

The growth rate for June 2007 is:

```
GROWTH.RATE <- ((VISITS.MONTH.2006[1] / VISITS.MONTH.2007[1]) * 100);
GROWTH.RATE

    June
31.67116
```

In previous sections, I made the forecast for 3 months (July, August, September). Now, I will add this forecast to the previous graph:

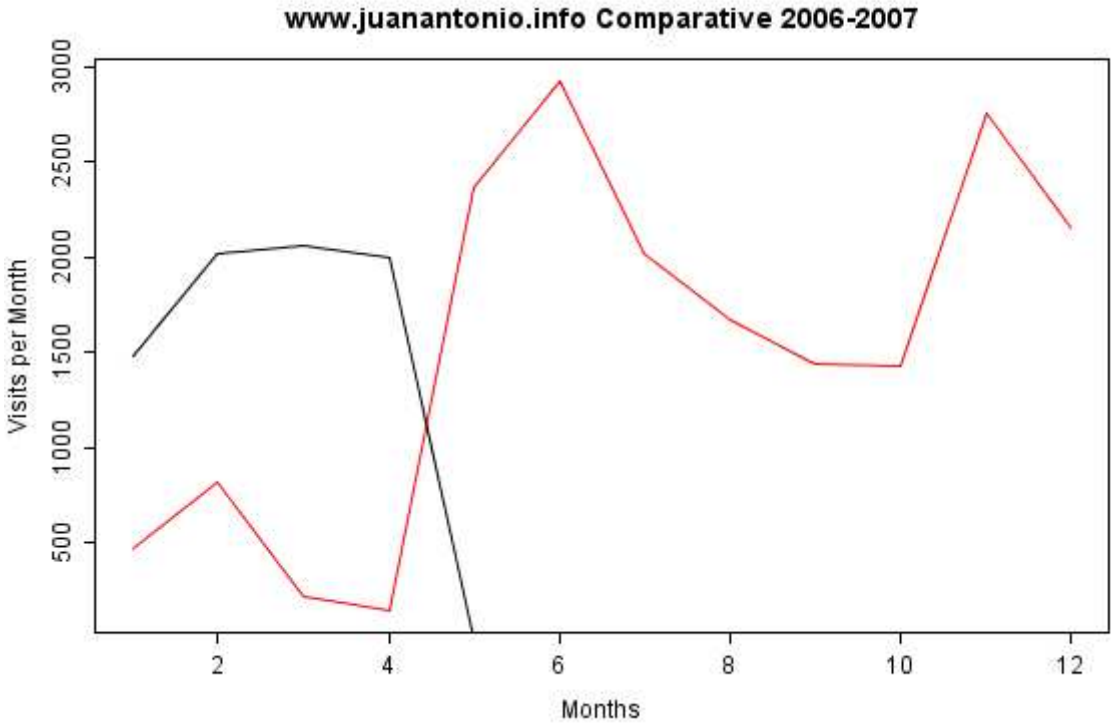


Figure 12: Comparative View between 2 years with Forecast data

How to integrate Google Analytics with R using C#

It is possible to integrate R with your .NET application developed with C# or with other .NET languaje using R-(D)COM.

R-(D)COM is a programmation interface to COM and DCOM (ex ActiveX; Microsoft distributed object interface) to access the R calculation engine. As such, it runs only under the Windows environnement.

R can be integrated with others technologies for example PHP. If you are intested in this technology visit this link:

To install R-(D)COM visit http://cran.r-project.org/contrib/extra/dcom/ . Once you have installed R-(D)COM, it is neccesary to config it.

Config R-(D)COM

First step is execute the command dcomcnfg in console window, then this command will launch component manager. Search StatConnectorSrv in DCOM components. When you find this component, see the properties

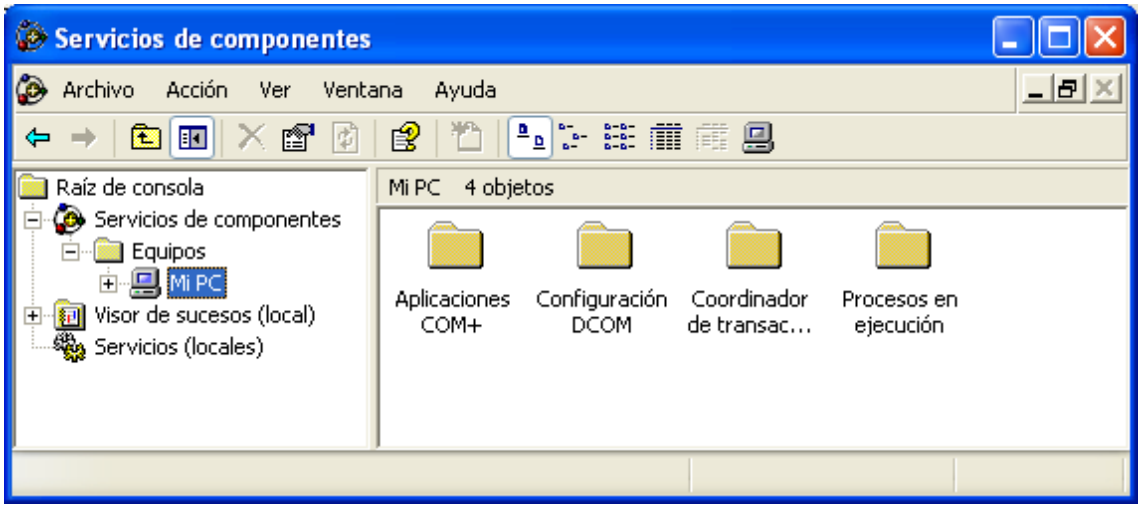


Figure 13: Component Manager

Last step to use StatConnectorSrv in your .NET applications is simply update permissions give the webiste account the launch permission (the account name depends on the configuration of your webserver, but mostly this is eihter 'NETWORK SERVICE', ASP or IUSR_machinename)

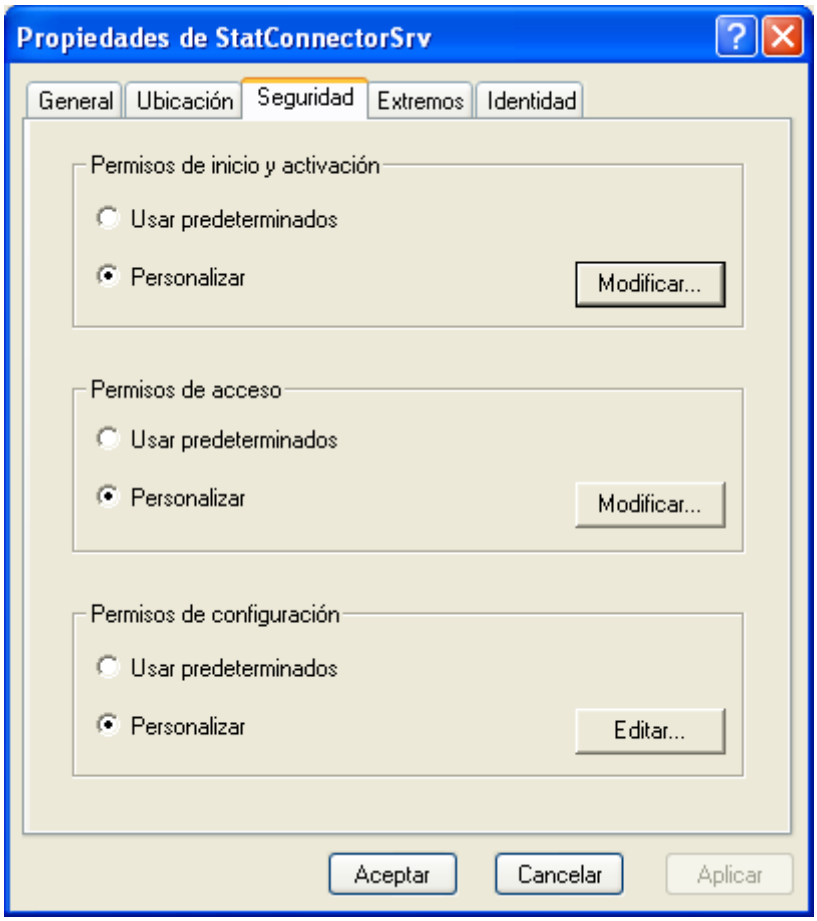


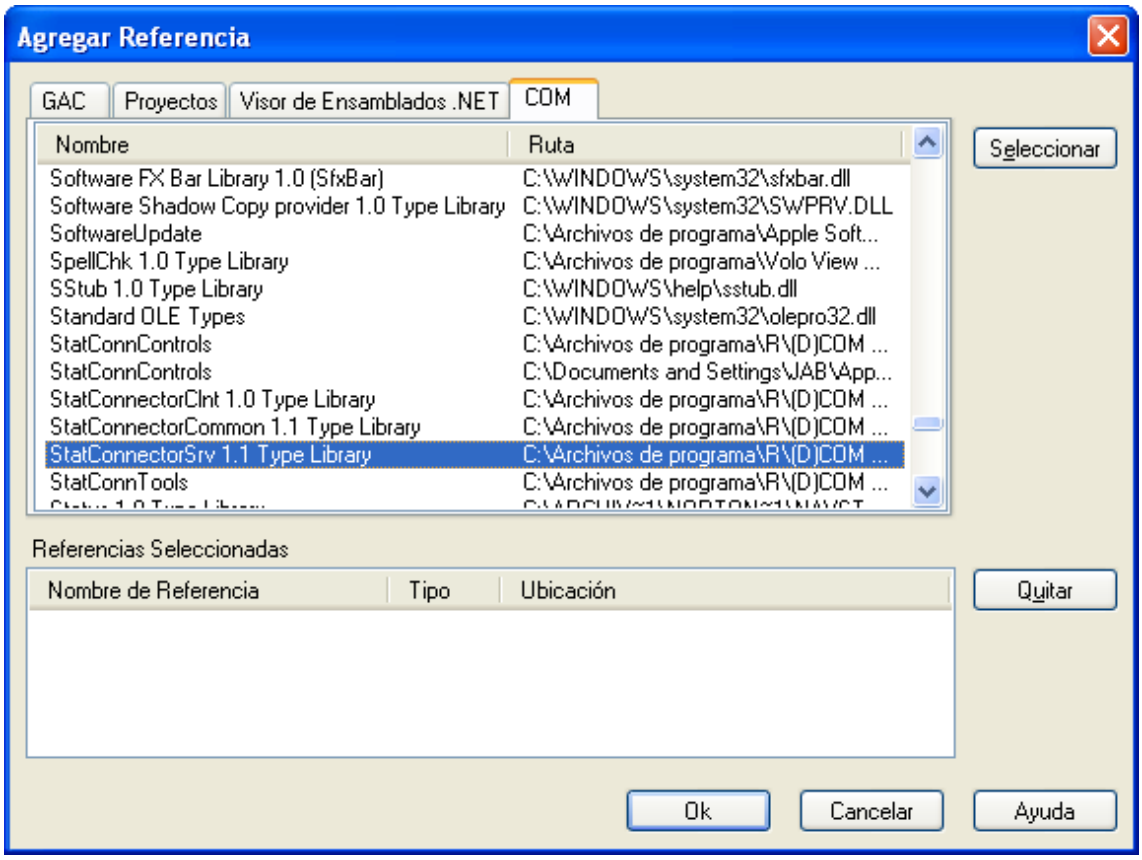
Figure 14: StatConnectorSrv properties

If you have followed this steps, you will develop applications with communication with R stat engine.

“Hello World” with C# and R

To develop this example, I will use a Open Source C# editor, Sharp Develop. I will create a Console application to show how to execute R scripts using C# as interface.

It is very important to import the following Reference in your .NET project:




```
using System;
using STATCONNECTORSRVLib;

namespace GOOGLEANALYTICS
{
      class MainClass
      {
            public static void Main(string[] args)
            {
                  Console.WriteLine("Hello World!");
             StatConnectorClass rObj = new
STATCONNECTORSRVLib.StatConnectorClass();
              rObj.Init("R");
              string cmdVars = "a <-1;";
              rObj.EvaluateNoReturn(cmdVars);
              cmdVars = "b <-1;";
              rObj.Evaluate(cmdVars);
              cmdVars = "a + b;";
              object result = rObj.Evaluate(cmdVars);
              Console.WriteLine((double)result);
```

```
            Console.ReadLine();
            rObj.Close();
          }
      }
}
```

In this example, the class connect with R and send 3 commands:

a <- 1;
b <- 1;
a + b;

Then the class receive the result of the third command, simple sum between 2 variables stored in R, the result: 2.

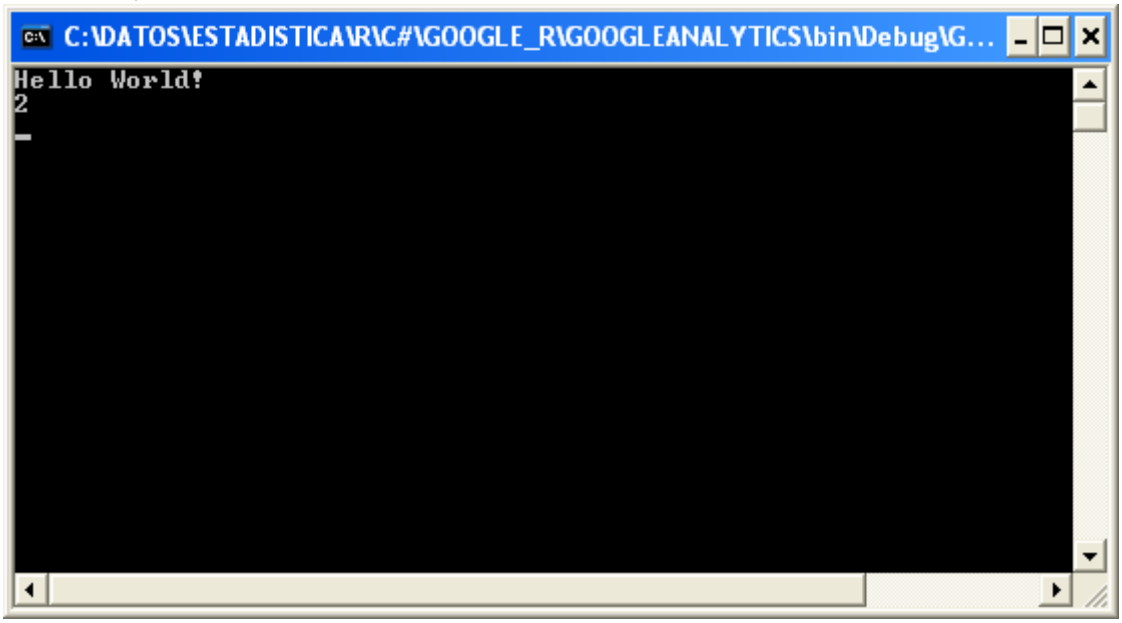


Figure 15: Execution of the example

Showing Graphs from R in your .NET applications.

I am going to show how to show graphs from R in yor .NET application developed with C#. Imagine that you need to plot graphs uning a data sample.

```
plot(cars);
```

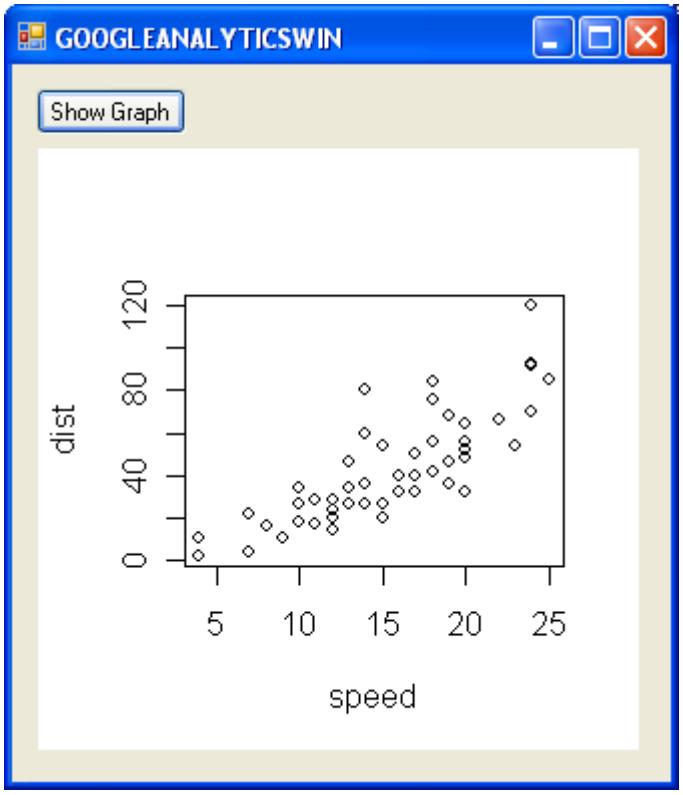


Figure 16: ..NET Winform showing a R graph

To get this result, it is necessary to save R graph in your file system. To use .NET object, Picturebox.

The script in R is:

```
bmp(file="c:/R_GUI_TEMP/345345.bmp", width=300, height=300);
plot(cars);
dev.off();
```

The source code of this example is:

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;

using STATCONNECTORSRVLib;

namespace GOOGLEANALYTICSWIN
{
	/// <summary>
	/// Description of MainForm.
	/// </summary>
	public partial class MainForm
	{
		[STAThread]
		public static void Main(string[] args)
		{
			Application.EnableVisualStyles();
			Application.SetCompatibleTextRenderingDefault(false);
```

```
                Application.Run(new MainForm());
            }

            public MainForm()
            {
                InitializeComponent();
            }

            void Button1Click(object sender, System.EventArgs e)
            {
                StatConnectorClass rObj = new
STATCONNECTORSRVLib.StatConnectorClass();
            rObj.Init("R");

            string R_SCRIPT  ="";

            Random random = new Random();
            int num = random.Next(100000000);

            string PATH = "c:/R_GUI_TEMP/" +  num.ToString() + ".bmp";

            R_SCRIPT = string.Format("bmp(file=\"{0}\", width={1},
height={1});", PATH, 300);
            rObj.EvaluateNoReturn(R_SCRIPT);
            R_SCRIPT = "plot(cars);";
            rObj.EvaluateNoReturn(R_SCRIPT);
            R_SCRIPT = "dev.off();";
            rObj.EvaluateNoReturn(R_SCRIPT);

                rObj.Close();

                this.Rgraph.Image = new Bitmap(PATH);


            }
        }
}
```

Conclussions

Google Analytics is a great product but it is possible to improve the information that it gives if you integrate with others systems as R. .NET is a alternative to integrate Google Analytics with R.

Bibliography

R
Statistical Graphs with R,
http://cran.r-project.org/doc/contrib/grafi3.pdf
Using R for Data Analysis and Graphics,
http://cran.r-project.org/doc/contrib/usingR-2.pdf
Introduction to JGR,
http://rosuda.org/JGR/JGR.pdf
R GUI projects,

http://www.sciviews.org/_rgui/
R packages,
http://cran.r-project.org/src/contrib/PACKAGES.html
Nortest Package,
http://cran.r-project.org/src/contrib/Descriptions/nortest.html
Dyn Package,
http://cran.r-project.org/src/contrib/Descriptions/dyn.html
Ardec Package,
http://cran.r-project.org/src/contrib/Descriptions/ArDec.html
Forecast Package,
http://cran.r-project.org/src/contrib/Descriptions/forecasting.html
Fbasis Package,
http://cran.r-project.org/src/contrib/Descriptions/fBasics.html
FCalendar Package,
http://cran.r-project.org/src/contrib/Descriptions/fCalendar.html
FSeries Package,
http://cran.r-project.org/src/contrib/Descriptions/fSeries.html
TSeries Package,
http://cran.r-project.org/src/contrib/Descriptions/tseries.html
XML Package,
http://cran.r-project.org/src/contrib/Descriptions/XML.html

R & Time Series
Time Series & R,
http://www.statoek.wiso.uni-goettingen.de/veranstaltungen/zeitreihen/sommer03/ts_r_intro.pdf
Time Series Reference Card with R,
http://cran.r-project.org/doc/contrib/Ricci-refcard-ts.pdf
Econometrics with R,
http://cran.r-project.org/doc/contrib/Farnsworth-EconometricsInR.pdf
Time Series Analysis and Its Applications,
http://lib.stat.cmu.edu/general/tsa2/

C#
RnetWeb,
http://www.msbi.nl/dnn/Default.aspx?tabid=159
Sharp Develop,
http://www.icsharpcode.net/OpenSource/SD/
RCOM-I
http://mailman.csd.univie.ac.at/mailman/listinfo/rcom-l