# Responsive Web Shop

# Overview

The analytics in this notebook covers a web shop that was fused from Desktop and Mobile variants into Responsive design, which accommodates for any device. The underlying hypothesis is, that the Responsive design had an effect on conversion rates (i.e. revenues) and or new visitors to the site and their transactions. To challenge this hypothesis, the null hypothesis holds that the new web shop does not affect conversion rates or transactions of new visitors. To prove the validity of the hypotheses, data was downloaded from Google Analytics (GA), to form the basis of the underlying analytics. It should be noted, that the value of the analytics increase by including additional information, such as non-transactional data. Especially data about bouncers (visitors who have left the site without purchasing) and detail about them and / or their sessions but also influencing details like marketing campaigns (content, reach, etc.) can help to weigh the outcome of these analytics. Such inclusions were not possible due to time contraints but should be considered for further analytics.

The analytics were based on two logical sets of data, to perform analytics from two perspectives:

1. The first set of data contains revenues figures and were downloaded from the “conversions” -> “ecommerce” -> “sales performance” reports in Google Analytics. This set of data covers all data since the very first month and it was used to assess potential impact on revenues. Rather than just comparing absolute actual figures, forecasts were made on the basis of different scenarios.

To allow for detail level analytics (as opposed to high level / aggregated analyses), revenue data was downloaded from GA (<https://www.quora.com/Is-there-any-way-to-export-Google-Analytics-data-into-a-CSV-file>) along with all relevant attributes (“secondary dimensions”). Revenue data are provided on a transaction level (lowest grain). Most other data in GA come aggregated only and, thus, are not suited for detail data driven analytics below.

1. The second set of data features some details not addressed in the first data set. It goes beyond the sessions that lead to transactions and covers information like bounce rates, distinction between new or returning visitors, etc.

Like above, csv exports were created - in this case via the “audience” -> “user explorer” reports. As opposed to the above, this data does cover transaction detail level. Rather, it is based on the so called client-id, which is in essence an approximation to identifying individuals on the grounds of certain session details and cookies. Depending on the user intentional or unintentional masking of his/her identity, a client-id does not necessarily equate to one individual. Rather one person may be depicted by various client-ids. This potentially distorts outcome - different client-ids may show similar behaviour but only because they may be one and the same person (as opposed to different individuals acting alike in web shop).

The data retrieval mentioned above is only one way of downloading GA data and it provides data quickly and easily. However, the process of downloading many details (in this case resulting in downloading several hundred csv export files), as required for this task, ultimately renders this approach inefficient considering the effort and time it takes to download all details. For repeated detail level downloading, Google has provided other means like for instance an R API, that allows for integration in automated processes (<http://code.markedmondson.me/googleAnalyticsR/>).

The files downloaded, were then used via External Tables created im an Oracle database (<http://allthingsoracle.com/external-tables-an-introduction/>). Tables ease data handling like when defining or filtering data sets, as well as facilitate complimentary querying. It also eases repeated processing of downloads and supports the inclusion of additional downloads to an existing data set.

The csv format provided through the GA downloads, cannot be used without modfications. Reasons include special characters in data blocks as well as column names (like slashes, parantheses, currency sumbols, quotation marks), reserved database field names (like DATE), etc. In addition each file contains header lines as well as extensive “appendix information” that had to be filtered, before the file content can be put to database use.

Logically, the downloads separate into the following logical parts (the four letter acronyms are used in the R and SQL programming below, as well):

* MDBC: Mobile shop data before change of web site to Responsive design (up to 3 weeks of data)
* DDBC: Desktop shop data before change of web site to Responsive design (up to 3 weeks of data)
* ADET: Responsive shop data for the entire time available (from the very first month to June current year), i.e. before and after the change of the web shop to Responsive design

The following analytics are divided into blocks (which in turn are subdivided in several steps):

I - Data Preparation (preparing GA download files and their content for later usage in database and R)

II - Data Exploration, Experimental Design and Hypothesis Testing (finding if there is statistical relevance to approach)

III - Analytics (applying models to the data sets for forecasting and to find correlations, so they exist)

# I - Data Preparation - Revenue Forecast

## Step 1 - Preparing Google Analytics export files

Summary: - each file name is shortened to its containing secondary dimension (attribute) - the content of each file is stripped of unnecessary lines - only one header line and data to be processed remain - header line is stripped of all blanks so the database table field names can be derived from this line - the cleansed content is written to new files in a separate directory

library("stringr")

#General parameters defining type of data sets and respectively their location - To be changed for each run accordingly

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#path for csv files to be processed, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/input/before\_change/Desktop/"

sInputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/input/Conversions/all\_data/"

#path for processed csv files, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/output/before\_change/Desktop/"

sOutputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/output/all\_data/"

#abbreviation of the data set block (see intro above), i.e.: "DDBC\_"

sBlock <- "ADET\_"

#Reading file names

files <- list.files( sInputPath )

#the marker is part of all file names and delineates the first part of the file name from the secondary dimension name

sFileNameMarker <- " # "

#Looping through the files

for( sFileName in files )

{

print( paste( "Processing File:", sFileName ) ) #show processing status on file level

ilength <- str\_length( sFileName ) #length of file name, used for substr later on

#finding the beginning of the string for the new file name in old name

sn <- str\_locate( sFileName, sFileNameMarker )

#building the name of the file in which to copy the valid lines

sNewFilename <- paste( sOutputPath, sBlock, substr( sFileName, sn[,2] + 1, ilength ), sep = "" )

sFilePathName <- paste( sInputPath, sFileName, sep = "" ) #adding path to file name

conn <- file( sFilePathName, open = "r" ) #open file connector

linn <- readLines( conn ) #read line by line

iLoop <- 0 #loop counter for the exit condition

sBailOutLine <- grep( ",,", linn ) #to identify the row, where reading stops => ,,"$

for( iLine in 1:length( linn ) ) #looping through the lines per file

{

if( (iLoop > 5) & (iLoop < sBailOutLine - 1) ) #start after lines containg export metadata and stop before footer

{

if( iLoop == 6 ) #this is the header line, containing field names, which need to be stripped of blanks

{

#appending header line

sObjectTmpName <- str\_split\_fixed( linn[iLine], ",", 2 )

sNewHeader <- paste( "SD\_", sObjectTmpName[1,2], sep = "" )

sNewHeader <- gsub( "[()/ ]", "\_", paste( sObjectTmpName[1,1], ",", sNewHeader, sep = "" ) )

write( sNewHeader, file = sNewFilename, append = TRUE )

}

else

{

#replace all quotation marks to make string replacement easier

#temp <- str\_replace\_all( linn[iLine], pattern = '"', replacement = '#')

temp <- gsub( "\"", "##", linn[iLine] )

#split line in 3 parts to isolate problematic number in the middle, containing double quotation marks and comma

tempsplit <- str\_split\_fixed(temp, "##", 3)

sBeginning <- gsub( "[$#]", "", tempsplit[1,1] ) #clear the first part of currency sign and hash

sEnd <- gsub( "[$#]", "", tempsplit[1,3] ) #clear the last part of currency sign and hash

sMiddle <- gsub( "[$,]", "", tempsplit[1,2]) #clear the middle part of currency sign and comma

sNewLine <- paste( sBeginning, sMiddle, sEnd, sep = "" ) #piece all parts together in one line

write( sNewLine, file = sNewFilename, append = TRUE ) #appending each line to new file

}

}

iLoop <- iLoop + 1 #incrementing loop counter

}

close( conn ) #close file connector

}

The above output lists the files processed in step 1 of the data preparation. It represents one set of data processed, like for instance from the Responsive shop (“ADET”), as defined in the parameters at the beginning of the Chunk.

## Step 2 - Generating CREATE TABLE statements for External Tables based on processed download files

Summary: - Form the DB object name based on the file name - Eliminate any special characters from header line, so the content can be used as column names for the tables - Build the Create Table statement - Write all Create Tables statements to a SQL script file

library("stringr")

library("stringi")

#General parameters defining type of data sets and respectively their location - To be changed for each run accordingly

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#path for the processed csv files from step 1, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/output/before\_change\_Desktop/"

sInputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/output/all\_data/"

#path for the SQL file containing the DDL statements, i.e.: "/home/oracle/Desktop/MSDS692/rws/SQL/output/"

sOutputPath <- "/home/oracle/Desktop/MSDS692/rws/SQL/output/"

#database path for data files (processed csv files) of External Tables, i.e.: "rws\_out\_ddbc"

sDBDirSource <- "rws\_out\_adet"

#database path for error / log / discard files created by external table reads, i.e.: "rws\_out\_eld"

sDBDirTarget <- "rws\_out\_eld"

#prefix for the External Tables, according to the abbreviation of the data set block (see intro above), i.e.: "DDBC\_"

sTablePrefix <- "ADET###"

#Reading file names

files <- list.files( sInputPath )

#file containing DDL statements for creating External Tables, i.e.: "CreateADET.sql"

sOutFile <- paste( sOutputPath, "Create", sTablePrefix, ".sql", sep = "" )

temp <- files

#Looping through the files

for( sFileName in files )

{

#reading file to extract all lines and to prepare renaming of secondary dimension by adding sd\_ prefix

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ilength <- str\_length( sFileName )

fileName <- paste( sInputPath, sFileName, sep = "" ) #adding path to file name

fConnInFiles <- file( fileName, open = "r" ) #open file connector

linn <- readLines( fConnInFiles ) #read file line by line

close( fConnInFiles ) #close file connector

sHeader <- linn[1] #reading header, needed to get field name for secondary dimension

tempsplit <- str\_split\_fixed( sHeader, ",", 2 ) #first step in isolating field name

sDimension <- str\_split\_fixed( tempsplit[1,2], ",", 2 ) #isolating secondary dimension

#building the dynamic content for the create statement

sField2 <- sDimension[1,1]

#Initializing sFileCount to satisfy condition for generating statements

sFileCount <- 1

#stripping suffix of the file name

sFilePrefix <- gsub( ".csv", "", sFileName )

#Making column names in header line compliant with table column/field name for the database

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#stripping group name of secondary dimension to avoid DB object names > 30 characterss

sObjectTmpName <- str\_split\_fixed( sFilePrefix, " ", 2 ) #isolating second part of dimension name

#replacing blanks and parantheses with underscores

sObjectName <- paste( sTablePrefix

, "\_"

, substr( gsub( "[()/ 0123456789]"

, ""

, sObjectTmpName[1,2]

)

, 1

, str\_length( sObjectTmpName[1,2] )

)

, sep = ""

)

#handling multiple files - since downloads in GA are limited to 5000 rows, the following brings them together logically

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#getting the vector index of the file names that correspond to the current one for multiple download file handling

iaFiles <- grep( gsub("[0123456789]", "", sFileName), gsub("[0123456789]", "", files) )

print( paste( "Processing File:", sFileName ) )

#generating list of source files, based on number of files per secondary dimension download

iNumSourceFiles <- 0 #number of downloaded file per secondary dimension download

iaETFiles <- NULL

for( iFileHit in iaFiles ) #looping through the vector referencing the files with the same beginning as the current one

{

if( str\_length( files[iFileHit] ) == str\_length( sFileName ) )

{

if( iNumSourceFiles == 1)

{

iaETFiles <- files[iFileHit] #only one file belonging to the download

}

else

{

iaETFiles <- c( iaETFiles, files[iFileHit] ) #list of files belonging to the download

}

iNumSourceFiles <- iNumSourceFiles + 1

}

}

#the following will only work up to 99 download files for secondary dimension exports / downloads

#checking for the first of multiple files for one secondary dimension - no need to loop for the remaining multiple files

iCurrentFileCounter <- "1"

if( (iNumSourceFiles > 1) & (iNumSourceFiles < 10) ) #less than 10 files in download

{

iCurrentFileCounter <- gsub( " ", "", substr( sFilePrefix, str\_length( sFilePrefix ) - 1, str\_length( sFilePrefix ) ) )

}

if( (iNumSourceFiles > 1) & (iNumSourceFiles > 9) ) #10 or more files in the download

{

iCurrentFileCounter <- gsub( " ", "", substr( sFilePrefix, str\_length( sFilePrefix ) - 2, str\_length( sFilePrefix ) ) )

}

sSourceFiles <- NULL

#for the first of multiple download files, the list of source files does not start with a comma

if( (iCurrentFileCounter == "1") | (iCurrentFileCounter == "01" ) | ( iNumSourceFiles == 1 ) )

{

sSourceFiles <- paste( "'", sFileName, "'", sep = "" )

#concatenate list of files for multiple files download

if( iNumSourceFiles > 1 )

{

#add the remaining files to the first one

i <- 2

while( i <= iNumSourceFiles )

{

sSourceFiles <- paste( sSourceFiles

, ", '"

, files[iaFiles[i]]

, "'"

, sep = ""

)

i <- i + 1

}

}

}

#DDL statements are only created for first file of multiple files or a single file

if( (iCurrentFileCounter == "1") | (iCurrentFileCounter == "01" ) | ( iNumSourceFiles == 1 ) )

{

#log / bad / discard file names

sLogFile <- paste( sObjectName, ".log", sep = "" )

sBadFile <- paste( sObjectName, ".bad", sep = "" )

sDiscFile <- paste( sObjectName, ".disc", sep = "" )

#Lines to be written to DML file

sLine01 <- paste( "DROP TABLE ", sObjectName, ";", sep = "" )

sLine02 <- paste( "CREATE TABLE", sObjectName )

sLine03 <- "( transaction\_id VARCHAR2(25)"

sLine04 <- paste( ", ", sField2, " VARCHAR2(4000)")

sLine05 <- ", revenue NUMBER"

sLine06 <- ", tax NUMBER"

sLine07 <- ", shipping NUMBER"

sLine08 <- ", refund\_amount NUMBER"

sLine09 <- ", quantity NUMBER"

sLine10 <- ")"

sLine11 <- "ORGANIZATION EXTERNAL"

sLine12 <- "( TYPE ORACLE\_LOADER"

sLine13 <- paste( " DEFAULT DIRECTORY", sDBDirSource )

sLine14 <- " ACCESS PARAMETERS( RECORDS DELIMITED BY NEWLINE"

sLine15 <- paste( " LOGFILE ", sDBDirTarget, ": '", sObjectName, ".log'", sep = "" )

sLine16 <- paste( " BADFILE ", sDBDirTarget, ": '", sObjectName, ".bad'", sep = "" )

sLine17 <- paste( " DISCARDFILE ", sDBDirTarget, ": '", sObjectName, ".disc'", sep = "" )

sLine18 <- " FIELDS TERMINATED BY ','"

sLine19 <- " MISSING FIELD VALUES ARE NULL"

sLine20 <- " )"

sLine21 <- " LOCATION"

sLine22 <- " ("

sLine23 <- paste( " ", sDBDirSource, ": ", sSourceFiles, sep = "" )

sLine24 <- " )"

sLine25 <- ")"

sLine26 <- "REJECT LIMIT UNLIMITED"

sLine27 <- ";"

#writing above lines to DML file

cat( sLine01

, "\n", sLine02

, "\n", sLine03

, "\n", sLine04

, "\n", sLine05

, "\n", sLine06

, "\n", sLine07

, "\n", sLine08

, "\n", sLine09

, "\n", sLine10

, "\n", sLine11

, "\n", sLine12

, "\n", sLine13

, "\n", sLine14

, "\n", sLine15

, "\n", sLine16

, "\n", sLine17

, "\n", sLine18

, "\n", sLine19

, "\n", sLine20

, "\n", sLine21

, "\n", sLine22

, "\n", sLine23

, "\n", sLine24

, "\n", sLine25

, "\n", sLine26

, "\n", sLine27, "\n"

, "\n"

, file = sOutFile

, append = TRUE

)

}

}

The above output lists the files processed in step 2 of the data preparation. It represents one set of data processed, like for instance from the Responsive shop (“ADET”), as defined in the parameters at the beginning of the Chunk.

## Step 3 - Establish database connection for all data retrieval

Establish database connection for all data based operations and R functions to follow

#install.packages("ROracle") #native Oracle Call Interface access to Oracle RDBMS

library( ROracle )

## Loading required package: DBI

drv <- dbDriver( "Oracle" ) #this driver is used for connecting to DB

host <- "localhost" #connection details: server name of Oracle instance

port <- 1521 #connection details: port, the instance can be reached under

sid <- "orcl" #connection details: name of the DB instance

connect.string <- paste(

"(DESCRIPTION=",

"(ADDRESS=(PROTOCOL=tcp) (HOST=", host, ") (PORT=", port, "))",

"(CONNECT\_DATA=(SID=", sid, ")))", sep = "") #connection details: entire connect string with

con <- dbConnect( drv, username = "rws", password = "welcome1", dbname=connect.string) #connection handler

The connection object created above can not be used in any SELECT statement in any Chunk following.

# II - Data Exploration - Revenue Forecast

## Checking when time series starts as well as for potentially missing months

If any time period was missing, the time series would be “broken”, which may cause problems when calculating seasonality, etc.

SELECT DISTINCT( substr( sd\_date, 1, 6 ) )

FROM v\_adet

ORDER BY substr( sd\_date, 1, 6 )

Displaying records 1 - 10

| (SUBSTR(SD\_DATE,1,6)) |
| --- |
| xxxx01 |
| xxxx02 |
| xxxx03 |
| xxxx04 |
| xxxx05 |
| xxxx06 |
| xxxx07 |
| xxxx08 |
| xxxx09 |
| xxxx10 |

The results above show, that the first month of data is January xxxx. Every month following until current June is filled with data (otherwise certain months would be missing, as the list displays only months in which transactions occured). In short, then, the data supports time series analyses on the grounds that there are no missing months.

# III - Analytics - Revenue Forecast

## Step 1 - fetching data for time series analytics over all months

The following Chunk sums up each month’s revenue from the first month to the last one downloaded.

#every month except current June is fetched because June data is incomplete and would potentially distort the forecast

adts <- dbSendQuery( con

, "SELECT Sum( revenue )

FROM v\_adet

WHERE sd\_date < 'xxxx0601'

GROUP BY substr( sd\_date, 1, 6 )

ORDER BY substr( sd\_date, 1, 6 )"

)

revbymonth <- fetch( adts )

head( revbymonth )

## SUM(REVENUE)

## 1 removed

## 2 removed

## 3 removed

## 4 removed

## 5 removed

## 6 removed

Months from very first month to current July

## Step 2 - turning fetched data into an R time series for all months

The “ts” function (<https://stat.ethz.ch/R-manual/R-devel/library/stats/html/ts.html>) stores the monthly revenue totals in a time series object. The starting point of the time series is the initial month identified by the SELECT statement in step 1 (very first month). As the time series object will be covering months, the frequency is defined to be “12”. The time series object is needed later on for the HoltWinters forecast function.

revtimeseries <- ts( revbymonth, frequency=12, start=c(year1,1) ) #build time series object

revtimeseries #display time series

## Jan Feb Mar Apr May Jun Jul

## year1 removed removed removed removed removed removed removed

## year2 removed removed removed removed removed removed removed

## year3 removed removed removed removed removed removed removed

## year4 removed removed removed removed removed removed removed

## year5 removed removed removed removed removed

## Aug Sep Oct Nov Dec

## year1 removed removed removed removed removed

## year2 removed removed removed removed removed

## year3 removed removed removed removed removed

## year4 removed removed removed removed removed

## year5

Revenues in time series format.

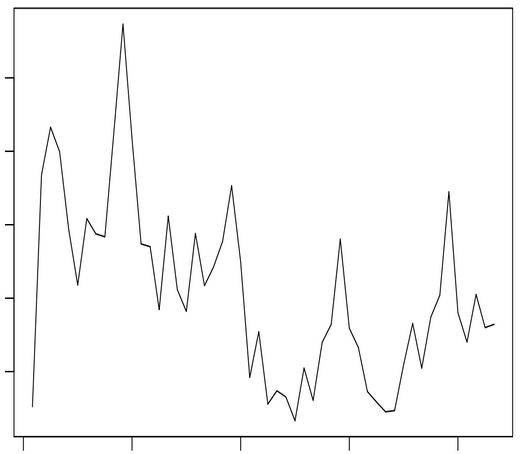
## Step 3 - plotting the time series

plot.ts( revtimeseries

, ylab = "Revenues"

, main = "Actual Revenues up to current June"

)



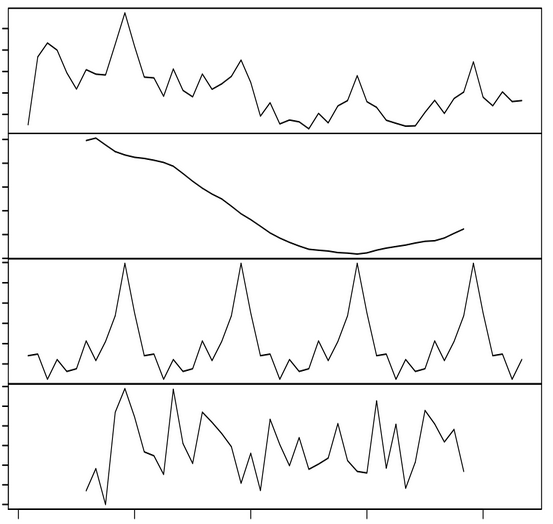
Monthly revenues for all months downloaded. The spikes around the end of each year indicate seasonal data. Therefore, forecasting needs to account for seasonality.

## Step 4 - decomposing seasonal data

Due to the seasonality derived from the plot above, the data needs to be decomposed into 3 parts - the trend component, the seasonal component and the irregular component. Decomposition aids assessing each individual component and its respective influence on forecasting. (<http://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html>).

revtimeseriescomponents <- decompose( revtimeseries ) #decompose time series

plot( revtimeseriescomponents )



Time series separation in: original component (“observed” → top), trend component (“trend” → second from top), seasonal component (“seasonal” → third from top) and estimated irregular component (“random” → bottom).

## Step 5 - seasonal adjustment

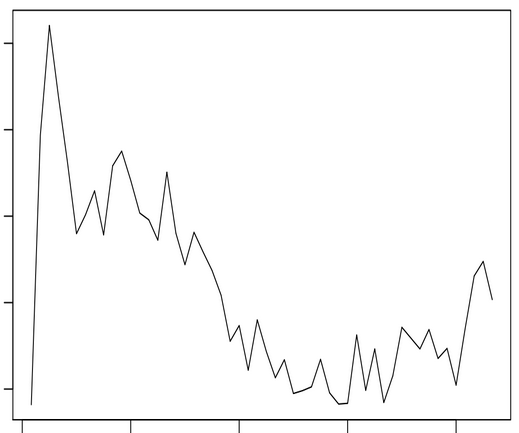
To show the effect of seasonal influence more clearly, the seasonal component (which is part of the decomposed time series object) is subtracted from the original time series as follows:

revtimeseriesseasonallyadjusted <- revtimeseries - revtimeseriescomponents$seasonal #subtract seasonality

plot( revtimeseriesseasonallyadjusted

, main = "Seasonally Adjusted Actual Revenues up to current June"

)



Seasonally adjusted revenues from very first month to current June

## Step 6 - HoltWinters forecasting

The forecast is derived with the HoltWinters funtion:

revtimeseriesforecasts <- HoltWinters( revtimeseries ) #applying forecast function on revenue time series

revtimeseriesforecasts #displaying result of forecast

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

##

## Call:

## HoltWinters(x = revtimeseries)

##

## Smoothing parameters:

## alpha: 0.248331

## beta : 0.257106

## gamma: 0.8504031

##

## Coefficients:

## [,1]

## a 304741.68

## b 11891.10

## s1 -56404.57

## s2 -20697.43

## s3 31702.89

## s4 -30108.15

## s5 33244.42

## s6 59831.31

## s7 189784.05

## s8 38384.47

## s9 -3591.81

## s10 26460.17

## s11 -33681.22

## s12 -43215.35

The above output shows, among other details, the alpha (0.248331), beta (0.257106) and gamma (0.8504031) values. The relatively low alpha value indicates that the current estimate (current point in time) is based on recent observations as well as some historic observations. The relatively low beta value indicates that the estimate of the slope of the trend is updated / adjusted slightly over the time series. Finally, the relatively high gamma value indicates that the estimate of the seasonal components currently (current point in time) is based upon very recent observations (potentially referring to the effect of the change to Responsive design).

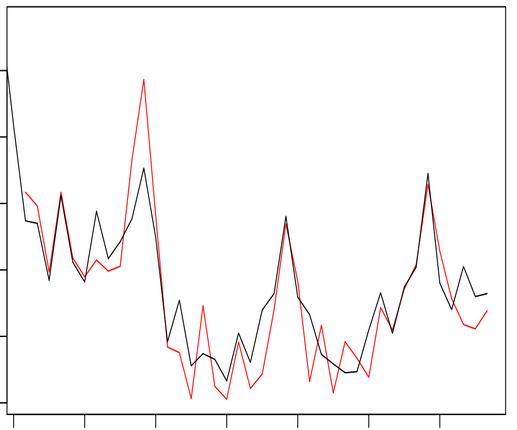
## Step 7 - plotting forecast on historic data

The forecast derived sofar is a calculation for past data. In other words, no prediction has taken place so far. Calculating a forecast on historic revenues helps assessing the accuracy / applicability of the approach:

plot( revtimeseriesforecasts

, main = "Actual Revenues and (historic) Forecasted Revenues"

)



Historic revenue figures (black line) together with the forecast of months in the past (red line). Diagram validates the approach taken so far is sound, as actuals and forecast “follow a close pattern”.

## Step 8 - forecasting (predicting) revenues

To forecast revenues for time periods beyond the actuals in the underlying data set (i.e. predicting), the forecast function of the HoltWinters method is used. The forecast is derived for 6 months:

library(forecast)

## Loading required package: zoo

##

## Attaching package: 'zoo'

## The following objects are masked from 'package:base':

##

## as.Date, as.Date.numeric

## Loading required package: timeDate

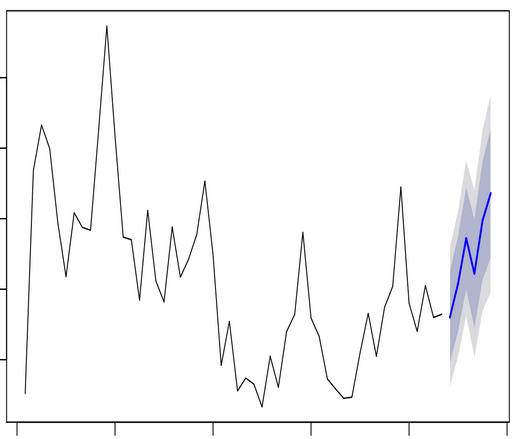
## This is forecast 7.3

revtimeseriesforecastspredict <- forecast.HoltWinters( revtimeseriesforecasts, h=6 )

plot.forecast( revtimeseriesforecastspredict

, main = "Predicted Forecast current June to current November"

)



The blue line in the above plot shows the predicted revenue. The shaded areas in dark grey and light grey represent the 80% and 95% prediction intervals, respectively. In other words, 80% of all future values are predicted to fall in the dark grey area and 95% are predicted not to be outside the light grey area (<https://en.wikipedia.org/wiki/Prediction_interval>).

The above forecast can be checked against forecasts in the past for instance by controllers to assess its validity and challenge accuracy. Depending on the result, it can be used as a baseline for comparing with current forecasts or even as a guideline for deriving coming forecasts.

## Step 9 - forecast based on all revenues before the change to Responsive design

To challenge the results of the forecasting above and especially the hypothesis, that the Responsive design affects revenues, the forecast is repeated with all data, except the part covering revenues after the change of the web shop. In case Responsive design did affect revenues, revenue predictions should differ. The following Chunk executes all prior steps but rather than running each steps in a dedicated Chunk as above, steps are now combined into Chunk. To accommodate for the 25 days of Reponsive web shop revenues included in the above forecast but not in this one, this forecast adds one additional month to the prediction, to arrive at the same target month (current November).

library(forecast)

library(zoo)

#fetching 25 days before change of web shop to Responsive design

adbc <- dbSendQuery( con

, "SELECT Sum( revenue )

FROM v\_adet

WHERE sd\_date < removed

GROUP BY substr( sd\_date, 1, 6 )

ORDER BY substr( sd\_date, 1, 6 )"

)

revbymonthbc <- fetch( adbc )

head( revbymonthbc )

## SUM(REVENUE)

## 1 removed

## 2 removed

## 3 removed

## 4 removed

## 5 removed

## 6 removed

#turning all fetched data before change into an R time series

revtimeseriesbc <- ts( revbymonthbc, frequency=12, start=c(year1,1) ) #build time series object

revtimeseriesbc

## Jan Feb Mar Apr May Jun Jul

## year1 removed removed removed removed removed removed removed

## year2 removed removed removed removed removed removed removed

## year3 removed removed removed removed removed removed removed

## year4 removed removed removed removed removed removed removed

## year5 removed removed removed removed removed

## Aug Sep Oct Nov Dec

## year1 removed removed removed removed removed

## year2 removed removed removed removed removed

## year3 removed removed removed removed removed

## year4 removed removed removed removed removed

## year5

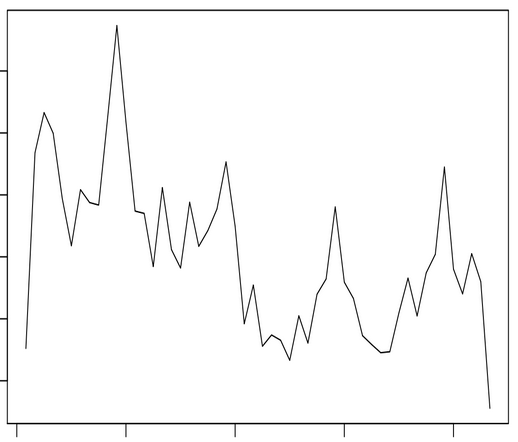
#plotting the time series

plot.ts( revtimeseriesbc

, ylab = "Revenues"

, main = "Actual Revenues up to current May"

)



#perform HoltWinters forecasting

revtimeseriesforecastsbc <- HoltWinters( revtimeseriesbc )

revtimeseriesforecastsbc

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

##

## Call:

## HoltWinters(x = revtimeseriesbc)

##

## Smoothing parameters:

## alpha: 0.1532087

## beta : 0.4051263

## gamma: 0.9060263

##

## Coefficients:

## [,1]

## a 254232.6326

## b -302.7918

## s1 -69306.4905

## s2 -24200.9555

## s3 28181.9359

## s4 -33652.3358

## s5 34803.4284

## s6 63987.6782

## s7 198714.8054

## s8 41075.4811

## s9 2495.6796

## s10 43915.8151

## s11 -13491.4305

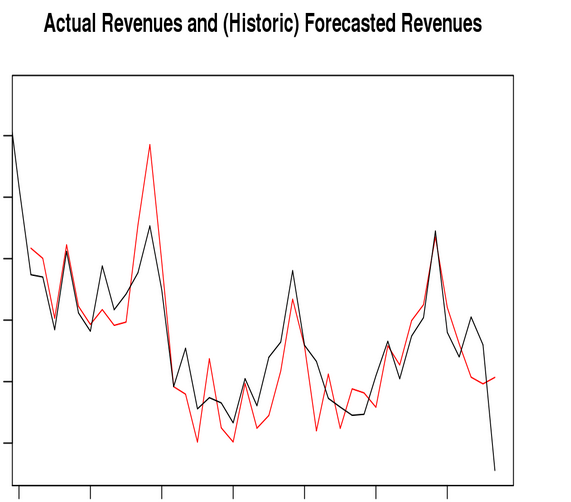
## s12 -186560.3031

#plotting historic forecast

plot( revtimeseriesforecastsbc

, main = "Actual Revenues and (Historic) Forecasted Revenues"

)



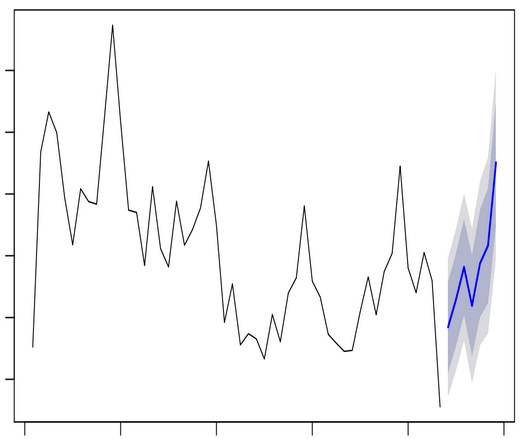
#predicting 7 months (to be par with the prior forecast) and plot

revtimeseriesforecastspredictbc <- forecast.HoltWinters( revtimeseriesforecastsbc, h=7 )

plot.forecast( revtimeseriesforecastspredictbc

, main = "Predicted Revenues current May to current November"

)



## Step 10 - interpretation of both long term predictions

The comparison of the predicted revenues including June’s actual figures in plot 8 (after 25 days of Responsive design being live) with the prediction ending with May’s actuals in plot 9 (i.e. before the Responsive web shop went live) reveal differing revenue developments. While plot in step 8 shows a more continuous and relatively steady incline, the plot in step 9 shows a rather sharp incline in October, starting from a lower base. Ultimately, both forecasts arrive at the same forecast level in November but with a different “growth pattern”. To further analyze the difference in the initial months after the change of the web shop, a short term forecast follows

## Step 11 - short term forecast based on 25 days before the change to Responsive design

To validate the existence of a potential short term effect of the change of the web shop design on revenues, this Chunk predicts revenues for 25 days following the change date (May xx +1 to June zz), based on the 25 days preceding the change (April xx to May yy):

library(forecast)

library(zoo)

#fetching 25 days before change of web shop to Responsive design

bcts <- dbSendQuery( con

, "SELECT Sum( revenue )

FROM v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

GROUP BY sd\_date

ORDER BY sd\_date"

)

revbcbymonth <- fetch( bcts )

head( revbcbymonth )

## SUM(REVENUE)

## 1 removed

## 2 removed

## 3 removed

## 4 removed

## 5 removed

## 6 removed

#defining start data and end date for the time series

indbc <- seq( as.Date(removed), as.Date(removed), by = "day" )

#the zoo function makes daily time series easier to handle since it eliminates the need to estimate the number of each day within the year

zoobc <- zoo( revbcbymonth, indbc)

#plotting the time series

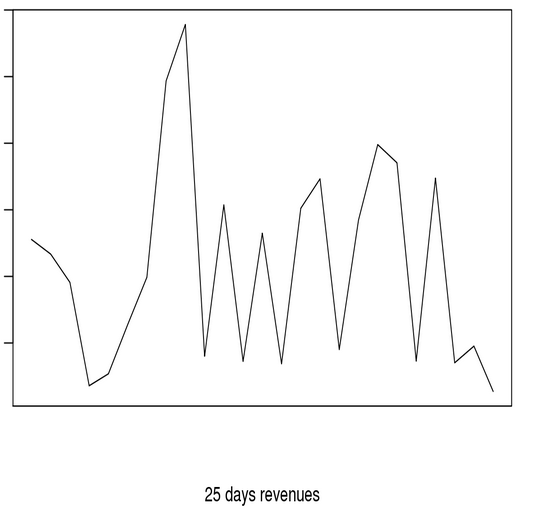
plot.ts( zoobc

, xaxt = "n"

, xlab = "25 days revenues"

, ylab = "Revenues Before Change to Responsive Design"

)



#forecasting historic data

revbctimeseriesforecasts <- HoltWinters( zoobc, gamma=FALSE )

revbctimeseriesforecasts

## Holt-Winters exponential smoothing with trend and without seasonal component.

##

## Call:

## HoltWinters(x = zoobc, gamma = FALSE)

##

## Smoothing parameters:

## alpha: 0.3840719

## beta : 0.004462037

## gamma: FALSE

##

## Coefficients:

## [,1]

## a 5493.528

## b -410.599

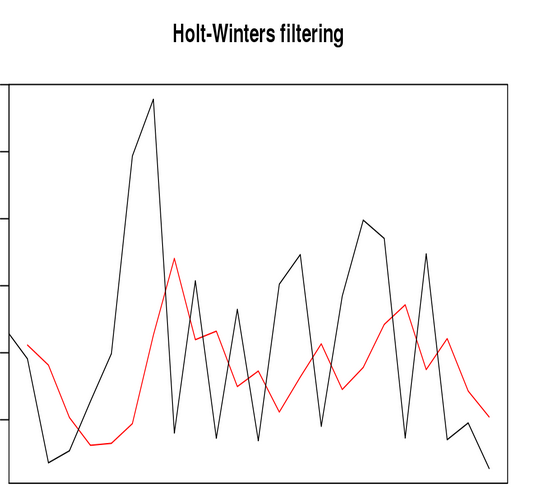
#plotting historic forecast

plot( revbctimeseriesforecasts

, xlab = "Actuals and Forecast 25 days Before Change to Responsive Design"

, xaxt = "n"

)



#predicting 25 days

revbctimeseriesforecastspredict <- forecast.HoltWinters(revbctimeseriesforecasts, h=25)

#plotting prediction

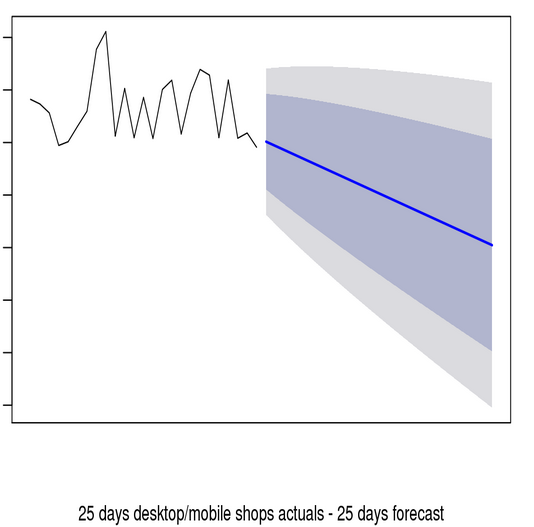
plot.forecast( revbctimeseriesforecastspredict

, xaxt = "n"

, xlab = "25 days desktop/mobile shops actuals - 25 days forecast"

, main = "Predicted Revenues - Actuals: xx Apr to zz May, Forecast: xx May to zz Jun"

)

Results of short term forecast based on 25 days before the change to Responsive design

## Step 12 - short term forecast based on 25 days after the change to Responsive design

As opposed to the step above, in this step the first 25 days of Responsive web shop revenues form the the basis for a short term forecast:

library(forecast)

library(zoo)

#fetching 25 days after change of web shop to Responsive design

acts <- dbSendQuery( con

, "SELECT Sum( revenue )

FROM v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

GROUP BY sd\_date

ORDER BY sd\_date"

)

revacbymonth <- fetch( acts )

head( revacbymonth )

## SUM(REVENUE)

## 1 removed

## 2 removed

## 3 removed

## 4 removed

## 5 removed

## 6 removed

#defining start data and end date for the time series

indac <- seq( as.Date(removed), as.Date(removed), by = "day" )

#the zoo function makes daily time series easier to handle since it eliminates the need to estimate the number of each day within the year

zooac <- zoo( revacbymonth, indac)

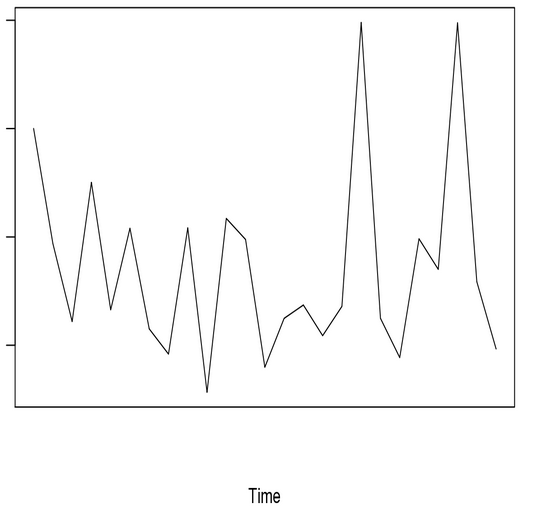
#plotting the time series

plot.ts( zooac

, xaxt = "n"

, ylab = "Revenues"

)



#forecasting historic data

revactimeseriesforecasts <- HoltWinters( zooac, gamma=FALSE )

revactimeseriesforecasts

## Holt-Winters exponential smoothing with trend and without seasonal component.

##

## Call:

## HoltWinters(x = zooac, gamma = FALSE)

##

## Smoothing parameters:

## alpha: 0.3254507

## beta : 0.6158097

## gamma: FALSE

##

## Coefficients:

## [,1]

## a 10327.6126

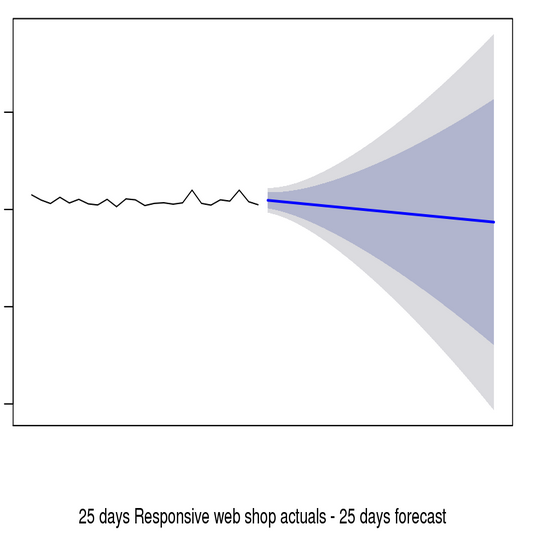
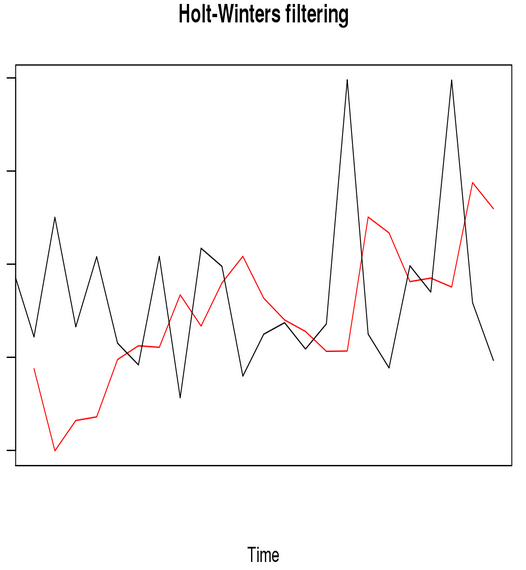
## b -934.4436

#plotting historic forecast

plot( revactimeseriesforecasts

, xaxt = "n"

)



#predicting 25 days

revactimeseriesforecastspredict <- forecast.HoltWinters(revactimeseriesforecasts, h=25)

#plotting prediction

plot.forecast( revactimeseriesforecastspredict

, xaxt = "n"

, xlab = "25 days Responsive web shop actuals - 25 days forecast"

, main = "Predicted Revenues - Actuals: xx May to zz Jun, Forecast: xx to zz Jun" )

Results of short term forecast based on 25 days after the change to Responsive design

## Forecast conclusion

The predicted increase in revenues reflects seasonal impact as well as recent events, such as the change of the web shop to a Responsive design. Considering the overall development of revenues, which have reached lows in the middle of years 3 and 4, revenues seem to recover slightly (see plot in step 4), in general.

The level of increase in the plot in step 8 appears much steeper compared to the revenue trend indicated in the plot of step 3. Considering the final month of the prediction (November), in spite of a different “growth pattern”, predicted revenues arrive at the same level. This indicates, that impact may be more short term, which, at least in part, could be attributable to the seasonal impact observed around the turn of the years (potential Christmas effects reflecting positive on revenues in November).

For verification, the short term forecast depicted in the plots in step 11 (before the change to Responsive design) and step 12 (after the change to Responsive design) show a favorable development after the change. Even though both short term forecasts depict a decline in revenues, the revenues forecasted on the basis of transactions made after the change to Responsive design (plot in step 12) decrease only “marginally”. This also, in short, speaks in favor of a positive short term effect of Responsive design.

In essence, the results above indicate a correlation between the change of web shop and predicted development of revenues (short term). For further confirmation on correlation, another set of analytics targets the impact on “the audience”.

# I - Data Preparation - Audience Data

## Step 1 - Preparing Google Analytics export files

Summary: - each file name is shortened to its containing day and audience group (new/returning) - the content of each file is stripped of the header lines - the cleansed content is written to new files in a separate directory

library("stringr")

#General parameters defining type of data sets and respectively their location - To be changed for each run accordingly

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#path for csv files to be processed, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/input/before\_change/Desktop/"

sInputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/input/Audience/"

#path for processed csv files, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/output/before\_change/Desktop/"

sOutputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/output/Audience/"

#abbreviation of the data set block (see intro above), i.e.: "AUDIENCE"

sBlock <- "AUDIENCE"

#Reading file names

files <- list.files( sInputPath )

#the marker is part of all file names and delineates the first part of the file name from the secondary dimension name

sFileNameMarker <- " # "

#Looping through the files

for( sFileName in files )

{

print( paste( "Processing File:", sFileName ) ) #show processing status on file level

ilength <- str\_length( sFileName ) #length of file name, used for substr later on

#finding the beginning of the string for the new file name in old name

sn <- str\_locate( sFileName, sFileNameMarker ) #i.e.: 57 59

#extracting date from file name, i.e.: xxxxxxxx

sDate <- substr( sFileName, sn[,1] - 8, sn[,1] - 1 )

#extracting user group from file name, i.e.: returning\_users

sUserGroup <- gsub( " ", "\_", substr( sFileName, sn[,2] + 1, ilength - 4 ) )

#building the name of the file in which to copy the valid lines

#i.e.: AUDIENCE\_xxxxxxxx\_returning users.csv

sNewFilename <- paste( sOutputPath, sBlock, "\_", sDate, "\_", substr( sFileName, sn[,2] + 1, ilength ), sep = "" )

#i.e.: /home/oracle/Desktop/MSDS692/rws/datasets/input/Audience/Analytics MFM - All Data User Explorer xxxxxxxx-zzzzzzzz # returning users.csv

sFilePathName <- paste( sInputPath, sFileName, sep = "" ) #adding path to file name

conn <- file( sFilePathName, open = "r" ) #open file connector

linn <- readLines( conn ) #read line by line - contains all lines in file read

iLoop <- 0 #loop counter for the exit condition

for( iLine in 1:length( linn ) ) #looping through the lines per file

{

if( (iLoop > 6) ) #start after header

{

#replace all quotation marks to make string replacement easier

#i.e.: xxxxxxxxxx.yyyyyyyyyy,1,00:06:11,0.00%,$0.00,0,100.00%

sCurrentLine <- gsub( "\"", "##", linn[iLine] )

#split line in 3 parts to isolate problematic number in the middle, containing double quotation marks and comma

#i.e.: "xxxxxxxxxx.yyyyyyyyyy,1,00:06:11,0.00%,$0.00,0,100.00%" "" ""

sLineSplit <- str\_split\_fixed(sCurrentLine, "##", 3)

sBeginning <- gsub( "[$#%<]", "", sLineSplit[1,1] ) #clear the first part of currency sign and hash

sEnd <- gsub( "[$#%<]", "", sLineSplit[1,3] ) #clear the last part of currency sign and hash

sMiddle <- gsub( "[$,%<]", "", sLineSplit[1,2]) #clear the middle part of currency sign and comma

sEnd <- paste( sEnd, ",", sDate, ",", sUserGroup, sep = "" ) #adding date to every data line, i.e.: ,xxxxxxxx,returning\_users

sNewLine <- paste( sBeginning, sMiddle, sEnd, sep = "" ) #piece all parts together in one line

write( sNewLine, file = sNewFilename, append = TRUE ) #appending each line to new file

}

iLoop <- iLoop + 1 #incrementing loop counter

}

close( conn ) #close file connector

}

The above output lists the files processed in step 1 of the data preparation.

## Step 2 - Generating CREATE TABLE statements for External Tables based on processed download files

* Add all downloaded files to the list of files that will be the source of the External Table
* Build the Create Table statement
* Write the Create Table statement to a SQL script

library("stringr")

library("stringi")

#General parameters defining type of data sets and respectively their location - To be changed for each run accordingly

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#path for the processed csv files from step 1, i.e.: "/home/oracle/Desktop/MSDS692/rws/datasets/output/before\_change\_Desktop/"

sInputPath <- "/home/oracle/Desktop/MSDS692/rws/datasets/output/Audience/"

#path for the SQL file containing the DDL statements, i.e.: "/home/oracle/Desktop/MSDS692/rws/SQL/output/"

sOutputPath <- "/home/oracle/Desktop/MSDS692/rws/SQL/output/"

#database path for data files (processed csv files) of External Tables, i.e.: "rws\_out\_ddbc"

sDBDirSource <- "rws\_out\_audience"

#database path for error / log / discard files created by external table reads, i.e.: "rws\_out\_eld"

sDBDirTarget <- "rws\_out\_eld"

#prefix for the External Tables, according to the abbreviation of the data set block (see intro above), i.e.: "DDBC\_"

sTablePrefix <- "AUDIENCE"

#Reading file names

files <- list.files( sInputPath )

#file containing DDL statements for creating External Tables, i.e.: "CreateDDBC.sql"

sOutFile <- paste( sOutputPath, "Create", sTablePrefix, ".sql", sep = "" )

sObjectName <- sTablePrefix

#Looping through the files

sSourceFiles <- NULL

i <- 1

for( sFileName in files )

{

#adding name of each input file to list, which will specify the data source for the Ecternal Table

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

if( i == 1 ) #i.e. first file in list has no leading comma

{

sSourceFiles <- paste( "'"

, files[i]

, "'"

, sep = ""

)

}

else #i.e. every file name added to the list is preceded by a comma

{

sSourceFiles <- paste( sSourceFiles

, ", '"

, files[i]

, "'"

, sep = ""

)

}

i <- i + 1

}

#log / bad / discard file names

sLogFile <- paste( sObjectName, ".log", sep = "" )

sBadFile <- paste( sObjectName, ".bad", sep = "" )

sDiscFile <- paste( sObjectName, ".disc", sep = "" )

#Lines to be written to DML file

sLine01 <- paste( "DROP TABLE ", sObjectName, ";", sep = "" )

sLine02 <- paste( "CREATE TABLE", sObjectName )

sLine03 <- "( client\_id VARCHAR2(50)"

sLine04 <- ", sd\_sessions NUMBER"

sLine05 <- ", avg\_session\_duration VARCHAR(8)"

sLine06 <- ", bounce\_rate NUMBER"

sLine07 <- ", revenue NUMBER"

sLine08 <- ", transactions NUMBER"

sLine09 <- ", goal\_conversion\_rate NUMBER"

sLine10 <- ", sd\_date VARCHAR2(8)"

sLine11 <- ", user\_group VARCHAR2(15)"

sLine12 <- ")"

sLine13 <- "ORGANIZATION EXTERNAL"

sLine14 <- "( TYPE ORACLE\_LOADER"

sLine15 <- paste( " DEFAULT DIRECTORY", sDBDirSource )

sLine16 <- " ACCESS PARAMETERS( RECORDS DELIMITED BY NEWLINE"

sLine17 <- paste( " LOGFILE ", sDBDirTarget, ": '", sObjectName, ".log'", sep = "" )

sLine18 <- paste( " BADFILE ", sDBDirTarget, ": '", sObjectName, ".bad'", sep = "" )

sLine19 <- paste( " DISCARDFILE ", sDBDirTarget, ": '", sObjectName, ".disc'", sep = "" )

sLine20 <- " FIELDS TERMINATED BY ','"

sLine21 <- " MISSING FIELD VALUES ARE NULL"

sLine22 <- " )"

sLine23 <- " LOCATION"

sLine24 <- " ("

sLine25 <- paste( " ", sDBDirSource, ": ", sSourceFiles, sep = "" )

sLine26 <- " )"

sLine27 <- ")"

sLine28 <- "REJECT LIMIT UNLIMITED"

sLine29 <- ";"

#writing above lines to DML file

cat( sLine01

, "\n", sLine02

, "\n", sLine03

, "\n", sLine04

, "\n", sLine05

, "\n", sLine06

, "\n", sLine07

, "\n", sLine08

, "\n", sLine09

, "\n", sLine10

, "\n", sLine11

, "\n", sLine12

, "\n", sLine13

, "\n", sLine14

, "\n", sLine15

, "\n", sLine16

, "\n", sLine17

, "\n", sLine18

, "\n", sLine19

, "\n", sLine20

, "\n", sLine21

, "\n", sLine22

, "\n", sLine23

, "\n", sLine24

, "\n", sLine25

, "\n", sLine26

, "\n", sLine27

, "\n", sLine28

, "\n", sLine29, "\n"

, "\n"

, file = sOutFile

, append = TRUE

)

The above list shows the input files processed into the CREATE TABLE statement

# II - Experimental Design and Hypothesis Testing - Audience Data

## Step 1 - Determine sample size needed for a statistically significant ANOVA test

To prepare initial analyses, the downloaded data sets need to be checked to make sure they are of statistical significance before performing an ANOVA test. While the ANOVA test in principle checks to see of there is a meaningful difference in the data sets to compare (Desktop / Mobile shops compared to Responsive shop), thereby justifying further analyses, the Power Analysis at this stage calculates how much data is needed for the ANOVA test.

Summary: Part a): retrieving data sets from database to compare revenues before and after change of web shop Part b): calculating effect size - a parameter needed in part c Part c): calculating the number of sample records needed using function “pwr.anova.test”

Detailing part c): This function pwr.anova.test allows us to determine how large a sample has to be if any effect is to be found, given a definable degree of confidence (<http://www.statmethods.net/stats/power.html>).

Explanation of the function / parameters (part “c” in the R chunck, below): k = number of groups => value is “3”, since mobile shop data are compared with desktop shop data and Responsive shop data n = common number of samples in each group (assuming groups of equal size) => left blank / to be calculated f = expected effect on size (0.1 for small, 0.25 for medium, .4 for large, according to Cohen definition) => effect size quantifies the size of the difference between groups (<https://www.leeds.ac.uk/educol/documents/00002182.htm>). An analysis of the group would be needed for an optimal f value. With respect to the impact of the responsive design on revenue, further information about the groups would be required, to include products and customers involved in the transactions. In absense of such details, the size of the difference between the groups is calculated based on the revenues observed (first part of the Power Analysis R Chunck below) sig.level = level of significance (typically 5%) power = power of test (typically 90% or larger)

library(lsr)

library(pwr)

# a) retrieving datasets from database to calculate effect size

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#retrieving mobile and desktop shop data before change

bdbc <- dbSendQuery( con

, "SELECT client\_id --transaction\_id

, revenue

FROM audience --v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND rownum < 120"

)

#load the records read to a data frame

desktopmobile\_bc <- fetch( bdbc )

#retrieving responsive shop data after change

rdac <- dbSendQuery( con

, "SELECT client\_id --transaction\_id

, revenue

FROM audience --v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND rownum < 120"

)

#load the records read to a data frame

responsive\_ac <- fetch( rdac )

# b) calculating effect size for the power test "f" parameter below

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

set.seed(119) #makes the result reproducible

x <- rnorm(desktopmobile\_bc[,2])

y <- rnorm(responsive\_ac[,2])

print("calculated effect size:")

## [1] "calculated effect size:"

cohensD( x, y )

## [1] 0.2532056

The above calculation arrives on the effect size, based on an equally sized sample of data from before and after the change of the web shop

The following Power test pwr.anov.test function is used to calculate the number of sample records needed. Meaning of the parameters: k -> number of groups n -> number of observations per sample / to be calculated f -> effect size (the standardised “diﬀerence” between treatment groups) sig.level -> significance level (Type I error probability, i.e. the incorrect rejection of a true null hypothesis / a “false positive”) power -> power of test (1 minus Type II error probability, i.e. incorrectly retaining a false null hypothesis / a “false negative”)

# c) calculating the number of sample records needed, using above calculated effect size

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

pwr.anova.test(k = 2, n = , f = 0.2532056, sig.level = 0.05, power = 0.9)

##

## Balanced one-way analysis of variance power calculation

##

## k = 2

## n = 82.91666

## f = 0.2532056

## sig.level = 0.05

## power = 0.9

##

## NOTE: n is number in each group

As shown above, at least 83 records (“n”) are required in each group for a statistically significant ANOVA test.

## Step 2 - Determine existing data set sizes, to establish whether there are enough records for ANOVA testing

The data set downloaded for comparison covers a maximum of 25 days after the go live of the Responsive web shop (May xx - June zz). For comparison purposes, the data set before the go-live will be kept to 25 days as well (April xx - May zz). The xx of May has been kept out deliberatedly to avoid a potential mix of mobile/desktop shop and responsive shop revenues.

Checking how many transactions are available and total revenue for Desktop / Mobile shop data for the given time frame (“Days” refer to days with business transactions)

SELECT Count( 'Client\_ID' ) AS Number\_of\_Clients

, Sum( revenue ) AS Total\_Revenue

, Count( Distinct( sd\_date ) ) AS Days

FROM audience --v\_ddbc

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

1 records

| NUMBER\_OF\_CLIENTS | TOTAL\_REVENUE | DAYS |
| --- | --- | --- |
| removed | removed | 25 |

Checking how many transactions are available and total revenue for Responsive shop data for the given time frame (“Days” refer to days with business transactions)

SELECT Count( 'Client\_ID' ) AS Number\_of\_Clients

, Sum( revenue ) AS Total\_Revenue

, Count( Distinct( sd\_date ) ) AS Days

FROM audience --v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

1 records

| NUMBER\_OF\_CLIENTS | TOTAL\_REVENUE | DAYS |
| --- | --- | --- |
| removed | removed | 25 |

As shown above, Desktop + Mobile / Responsive shop records exceed the minimum of 83 records (number of clients) each for ANOVA testing, so the analysis can be continued.

## Step 3 - summary check if there are differences to be observed in the number of transactions and revenues

Checking the differences between the combined Deskop / Mobile shop data and the Responsive shop data

SELECT d.transactions + m.transactions AS transactions\_before\_change

, d.revenue + m.revenue AS revenue\_before\_change

, d.quantity + m.quantity AS quantity\_before\_change

, r.transactions AS transactions\_after\_change

, r.revenue AS revenue\_after\_change

, r.quantity AS quantity\_after\_change

, r.transactions

- (d.transactions + m.transactions) AS change\_in\_transactions

, r.revenue

- (d.revenue + m.revenue) AS change\_in\_revenue

, r.quantity

- (d.quantity + m.quantity) AS change\_in\_quantity

, ( r.transactions

- ( d.transactions + m.transactions )

) / ( d.transactions + m.transactions )

\* 100 AS percentage\_change\_transactions

, ( r.revenue

- ( d.revenue + m.revenue )

) / ( d.revenue + m.revenue )

\* 100 AS percentage\_change\_revenue

, ( r.quantity

- ( d.quantity + m.quantity )

) / ( d.quantity + m.quantity )

\* 100 AS percentage\_change\_quantity

FROM ( SELECT Count( 1 ) AS transactions

, Sum( revenue ) AS revenue

, Sum( quantity ) AS quantity

FROM v\_ddbc

WHERE sd\_date >= removed

AND sd\_date <= removed

) d

, ( SELECT Count( 1 ) AS transactions

, Sum( revenue ) AS revenue

, Sum( quantity ) AS quantity

FROM v\_mdbc

WHERE sd\_date >= removed

AND sd\_date <= removed

) m

, ( SELECT Count( 1 ) AS transactions

, Sum( revenue ) AS revenue

, Sum( quantity ) AS quantity

FROM v\_adet

WHERE sd\_date >= removed

AND sd\_date <= removed

) r

1 records

| TRANSACTIONS\_BEFORE\_CHANGE | REVENUE\_BEFORE\_CHANGE | QUANTITY\_BEFORE\_CHANGE | TRANSACTIONS\_AFTER\_CHANGE | REVENUE\_AFTER\_CHANGE | QUANTITY\_AFTER\_CHANGE | CHANGE\_IN\_TRANSACTIONS | CHANGE\_IN\_REVENUE | CHANGE\_IN\_QUANTITY | PERCENTAGE\_CHANGE\_TRANSACTIONS | PERCENTAGE\_CHANGE\_REVENUE | PERCENTAGE\_CHANGE\_QUANTITY |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| removed | removed | removed | removed | removed | removed | -5747 | 15469.59 | 644 | -91.77579 | 7.727855 | 8.701527 |

The results above show that the number of transactions has decreased after the go live of the Responsive shop, while revenues and quantities have increased. Provided the ANOVA test leads to a rejection of the null hypothesis (i.e. there are statistically meaningful differences in the data sets), further analyses should be performed.

## Step 4 - Reading data sets for ANOVA testing

The following datasets are based on the filtering shown above, to allow for groups of similar size

#retrieve a data set for all shops for ANOVA testing

mdr <- dbSendQuery( con

, "SELECT 'Desktop/Mobile' AS shop

, client\_id

, sd\_date

, revenue

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

UNION

SELECT 'Responsive' AS shop

, client\_id

, sd\_date

, revenue

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'"

)

#load the records read to a data frame

allsample <- fetch( mdr )

allsample

## SHOP CLIENT\_ID SD\_DATE

## 1 Desktop/Mobile xxxxxxxxx1.yyyyyyyyy1 xxxxxxx1

## 2 Desktop/Mobile xxxxxxxxx2.yyyyyyyyy2 xxxxxxx2

## ……….. Desktop/Mobile ………………….…………………………….. ……………...

## REVENUE

## 1 removed

## 2 removed

## ... …………...

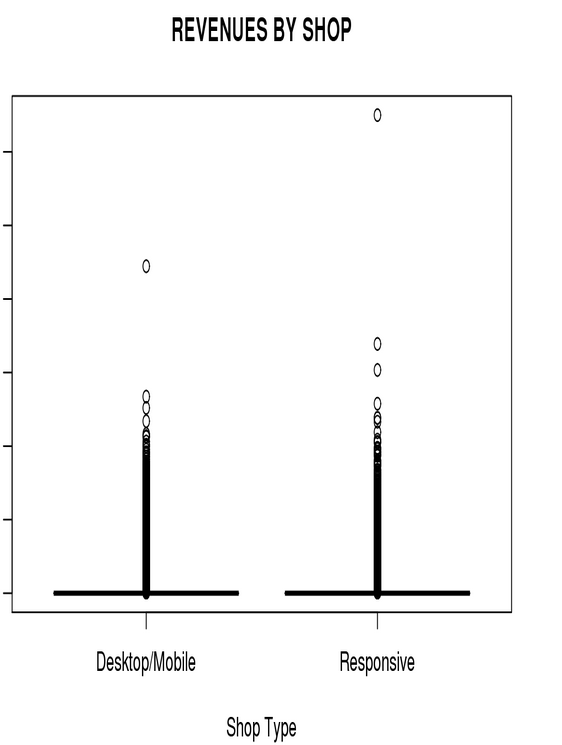
## [ reached getOption("max.print") -- omitted xx rows ]

Samples data for ANOVA test.

## Step 5 - Plotting ANOVA sample data set

Plotting to check visually if revenues are of equal nature between Mobile / Desktop / Responsive shops

boxplot( allsample[,4]~allsample[,1], data=allsample, main=toupper("Revenues by Shop"), xlab="Shop Type", ylab="Revenue", col="skyblue" )



The above boxplot graphically displays statistical keyfigures of the three data sets (<http://www.physics.csbsju.edu/stats/box2.html>). The midline (median revenue) of Responsive shop revenues is close to the one of the Desktop / Mobile shops, as are the upper and lower limits of the Resonsive box (the third and first quartile / 75th and 25th percentile of the revenues), which are not dicernable from the median due to their proximity. The “circles” (outlier revenues) do show different patterns, though.

In essence, then, Responsive shop revenues do follow the same “pattern” as to the other two shops for the 50 days observed. The box plot is only intended to indicate the function to be used in the following experimental testing.

## Step 6 - Testing for homoscedasticity

The following test is intended to identify the “nature” of the data sets. The closer their content is (homoscedasticity), the higher the indication to use the aov function for ANOVA testing. The test is also known as the Bartlett’s test (<http://www.itl.nist.gov/div898/handbook/eda/section3/eda357.htm>).

bartlett.test( allsample$REVENUE ~allsample$SHOP, allsample )

##

## Bartlett test of homogeneity of variances

##

## data: allsample$REVENUE by allsample$SHOP

## Bartlett's K-squared = 282.54, df = 1, p-value < 2.2e-16

With a p-value of 0,0000000000000022 being far below 0.05, the Null Hypothesis that the variances are equal between the groups can be rejected. The above further indicates homoscedasticity.

As a result, a one way ANOVA test with the aov function is appropriate (in case of heterocedasticity, the “oneway” function would be used for ANOVA testing).

## Step 7 - ANOVA testing

The following Analysis of Variance (ANOVA) test helps establish if potential differences between the groups / data sets are greater than expected (i.e. worth further analysis) or perhaps are caused by chance, like a sampling error. In the latter case, further analysis would not be indicated (<https://www.edanzediting.com/blogs/statistics-anova-explained?utm_expid>=.AOHD7hLYSMyfsa41SGKulA.0&utm\_referrer=https%3A%2F%2Fwww.google.de%2F).

Parameters: dependent variable -> revenues independent variable -> shop dataset -> new users, 25 days before and after go-live of Responsive shop

#oneway.test( allsample[,4]~allsample[,1], allsample )

cliid.aov <- aov( allsample$REVENUE ~ allsample$SHOP, allsample )

summary( cliid.aov)

## Df Sum Sq Mean Sq F value Pr(>F)

## allsample$SHOP 1 removed removed 27.93 1.26e-07 \*\*\*

## Residuals removed removed 93

## ---

## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

The above listed output shows an f-value far greater than 1 (f-value of 1 stands for a variance between groups that one would expect by chance) and a p-value of 0.000000126. P-values lower than 0.05 (usually) indicate a statistically meaningfull difference between groups (compared to the statistical variances within each group). The null hypothesis (i.e. no statistically meaningful differences between the groups) can, hence, be rejected, which in turn speaks in favour of further analyses.

## Step 8 - querying how many new users to the web shop have made a purchase on their first visit

The audience data sets available for download, allowed for a comparison of period 50 days prior to the change of the web shop and 25 days after the change. The average of buying new users was taken for both 25 day windows prior to the change.

First 25 days window:

SELECT nub.new\_users\_buying + nunb.new\_users\_not\_buying AS new\_users

, nub.new\_users\_buying

, nunb.new\_users\_not\_buying

, nub.new\_users\_buying

/ nunb.new\_users\_not\_buying

\* 100 AS percentage\_new\_users\_buying

FROM ( SELECT Count(\*) AS new\_users\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions > 0

) nub

, ( SELECT Count(\*) AS new\_users\_not\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions = 0

) nunb

1 records

| NEW\_USERS | NEW\_USERS\_BUYING | NEW\_USERS\_NOT\_BUYING | PERCENTAGE\_NEW\_USERS\_BUYING |
| --- | --- | --- | --- |
| Removed | removed | removed | removed |

Second 25 days window:

SELECT nub.new\_users\_buying + nunb.new\_users\_not\_buying AS new\_users

, nub.new\_users\_buying

, nunb.new\_users\_not\_buying

, nub.new\_users\_buying

/ nunb.new\_users\_not\_buying

\* 100 AS percentage\_new\_users\_buying

FROM ( SELECT Count(\*) AS new\_users\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions > 0

) nub

, ( SELECT Count(\*) AS new\_users\_not\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions = 0

) nunb

1 records

| NEW\_USERS | NEW\_USERS\_BUYING | NEW\_USERS\_NOT\_BUYING | PERCENTAGE\_NEW\_USERS\_BUYING |
| --- | --- | --- | --- |
| removed | removed | removed | 4.627063 |

Average of both windows:

SELECT removed + removed AS new\_users

, removed + removed AS new\_users\_buying

, removed + removed AS new\_users\_not\_buying

, ( 0.0336993563044301400984475577432790609618

+ 0.0462706322898872082630977566096862873757

)

/ 2 AS average\_perc\_new\_users\_buying

FROM dual

1 records

| NEW\_USERS | NEW\_USERS\_BUYING | NEW\_USERS\_NOT\_BUYING | AVERAGE\_PERC\_NEW\_USERS\_BUYING |
| --- | --- | --- | --- |
| removed | removed | removed | 0.039985 |

On average, 3.99 % of new users made a purchase before the change of the web shop.

Third 25 days window:

SELECT nub.new\_users\_buying + nunb.new\_users\_not\_buying AS new\_users

, nub.new\_users\_buying

, nunb.new\_users\_not\_buying

, nub.new\_users\_buying

/ nunb.new\_users\_not\_buying

\* 100 AS percentage\_new\_users\_buying

FROM ( SELECT Count(\*) AS new\_users\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions > 0

) nub

, ( SELECT Count(\*) AS new\_users\_not\_buying

FROM audience

WHERE sd\_date >= removed

AND sd\_date <= removed

AND user\_group = 'new\_users'

AND transactions = 0

) nunb

1 records

| NEW\_USERS | NEW\_USERS\_BUYING | NEW\_USERS\_NOT\_BUYING | PERCENTAGE\_NEW\_USERS\_BUYING |
| --- | --- | --- | --- |
| removed | removed | removed | 4.228936 |

On average, 4.27% of new users made a purchase after the change of the web shop.

# III - Analytics - Audience Data

## Step 1 - calculating required sample size for A/B test

Calculate the power of the test comparing the proportions in two groups parameters: n -> size of sample needed / to be calculated p1 -> probability 1 (calculated above - probability that new users make a purchase in old shops) p2 -> probability 2 (calculated above - probability that new users make a purchase in new shop) sig.level -> significance level (Type I error probability, i.e. the incorrect rejection of a true null hypothesis / a “false positive”) power -> power of test (1 minus Type II error probability, i.e. incorrectly retaining a false null hypothesis / a “false negative”)

power.prop.test(n=, p1=0.03998499, p2=0.04268766, sig.level=0.05, power=0.8)

##

## Two-sample comparison of proportions power calculation

##

## n = 85161.59

## p1 = 0.03998499

## p2 = 0.04268766

## sig.level = 0.05

## power = 0.8

## alternative = two.sided

##

## NOTE: n is number in \*each\* group

Solving for the required number of visitors (“n”), at least 85162 need to be checked in both data sets.

## Step 2 - A/B test

Testing the rate of new users buying - comparing: removed new visitors, of which removed are buying in the old shops to removed new visitors, of which removed are buying in the old shop

prop.test(c(removed, removed), c(removed, removed))

##

## 2-sample test for equality of proportions with continuity

## correction

##

## data: c(removed, removed) out of c(removed, removed)

## X-squared = 9.8332, df = 1, p-value = 0.001714

## alternative hypothesis: two.sided

## 95 percent confidence interval:

## 0.001104292 0.004797778

## sample estimates:

## prop 1 prop 2

## 0.04389107 0.04094003

## A/B Test conclusion

The p-value is less than 0.05, so the hypothesis that the rates of new users buying are equal can be rejected and it can be assumed that the rate of new visitors to the Responsive web shop buying is higher.

In short, the A/B test shows an impact of the Responsive shop on new visitors buying on their first visit.

# Overall conclusion

As summarized above, Responsive design has a positive impact on revenues forecasted and new visitors making a purchase on their first visit.

At this is based on a relatively small time frame of 25 days, this tendency should be evaluated covering a longer time frame if meaningful deductions are to be made.

Additional details on users, transactions, marketing campaigns and other external influencing factors should also help identify causation. The current data is not sufficient to isolate causing factors, like device used with respect to product purchased, preceding campaigns and the like.

# Sources Consulted

Avril Coghlan, Using R for Time Series Analysis. Retrieved from [http://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html](https://email.regis.edu/owa/redir.aspx?C=-Q6oY-XPzr7PnWbZXypD23VU-XFLgjO4ju37Ha8RaiwaV2XyIcTUCA..&URL=http%3A%2F%2Fa-little-book-of-r-for-time-series.readthedocs.io%2Fen%2Flatest%2Fsrc%2Ftimeseries.html)  
Box Plot: Display of Distribution, Retrieved from [http://www.physics.csbsju.edu/stats/box2.html](https://email.regis.edu/owa/redir.aspx?C=DhiA479ODhAA6nW34Kr0pu3z_FkTHxcA-AA214vQdPkaV2XyIcTUCA..&URL=http%3A%2F%2Fwww.physics.csbsju.edu%2Fstats%2Fbox2.html)  
Deborah J. Rumsey. What a P-Value Tells You About Statistical Data. Retrieved from [http://www.dummies.com/education/math/statistics/what-a-p-value-tells-you-about-statistical-data](https://email.regis.edu/owa/redir.aspx?C=L2YDAoNY6VktuUtlr5xDHSdIw9mVJUcG7zCm4kWWd_oaV2XyIcTUCA..&URL=http%3A%2F%2Fwww.dummies.com%2Feducation%2Fmath%2Fstatistics%2Fwhat-a-p-value-tells-you-about-statistical-data)/  
Department of Commerce [(2013](tel:2013), October). Bartlett's Test. Retrieved from [http://www.itl.nist.gov/div898/handbook/eda/section3/eda357.htm](https://email.regis.edu/owa/redir.aspx?C=IeBRzG3hu5UqJKm1-3L359voRASiDL6cUh5PzqU1TNIaV2XyIcTUCA..&URL=http%3A%2F%2Fwww.itl.nist.gov%2Fdiv898%2Fhandbook%2Feda%2Fsection3%2Feda357.htm)  
Kamil Bartocha [(2014](tel:2014)). Multiple A/B/n Tests in Marketing with ANOVA and R. Retrieved from [http://www.marketingdistillery.com/2014/08/10/multiple-abn-tests-in-marketing-with-anova-and-r](https://email.regis.edu/owa/redir.aspx?C=IZG9f06nGY8Jh7limrdTQ-6pToBxJljbAE84mUUC7j4aV2XyIcTUCA..&URL=http%3A%2F%2Fwww.marketingdistillery.com%2F2014%2F08%2F10%2Fmultiple-abn-tests-in-marketing-with-anova-and-r)/  
Noah Lorang [(2011](tel:2011)). A/B Testing Tech Note: determining sample size. Retrieved from [https://signalvnoise.com/posts/3004-ab-testing-tech-note-determining-sample-size](https://email.regis.edu/owa/redir.aspx?C=5t-bbVhLYxlUBl36FhjQ-6A1OYsVxAxxLjnpTylYWd0aV2XyIcTUCA..&URL=https%3A%2F%2Fsignalvnoise.com%2Fposts%2F3004-ab-testing-tech-note-determining-sample-size)  
Robert Coe, It's the Effect Size, Stupid [(2002](tel:2002)). Retrieved from [https://www.leeds.ac.uk/educol/documents/00002182.htm](https://email.regis.edu/owa/redir.aspx?C=c84NmP-fFe6mInnkJdMNp0sPppT9p48M4vOErnHXaU4aV2XyIcTUCA..&URL=https%3A%2F%2Fwww.leeds.ac.uk%2Feducol%2Fdocuments%2F00002182.htm)  
Robert I. Kabacoff, Ph.D [(2017](tel:2017)). Quick-R,Power Analysis. Retrieved from [http://www.statmethods.net/stats/power.html](https://email.regis.edu/owa/redir.aspx?C=w5RFEtSH39t2GGQ_yimuMYNn3nwgQNPCofLCw83EZ6kaV2XyIcTUCA..&URL=http%3A%2F%2Fwww.statmethods.net%2Fstats%2Fpower.html)

Script