Time Series Analysis: Basic Forecasting.

As published in Benchmarks RSS Matters, April 2015

http://web3.unt.edu/benchmarks/issues/2015/04/rss-matters

Jon Starkweather, PhD

Jon Starkweather, PhD
`jonathan.starkweather@unt.edu`
Consultant
Research and **S**tatistical Support

http://www.unt.edu

http://www.unt.edu/rss

RSS hosts a number of "Short Courses".
A list of them is available at:
http://www.unt.edu/rss/Instructional.htm

Those interested in learning more about R, or how to use it, can find information here:
`http://www.unt.edu/rss/class/Jon/R_SC`

Time Series Analysis: Basic Forecasting.

This month's article will provide a very gentle introduction to basic time series analysis. The primary reference for this article is Hyndman and Athanasopoulos (2015) and it is highly recommended, not least because it is completely free[1] and regularly updated at OTexts. If you are unfamiliar, there is a growing group of academics and researchers using `www.OTexts.org` [Online, Open Access Textbooks] to remove barriers to learning - a most honorable endeavor. The book by Hyndman and Athanasopoulos also has a companion R package; 'fpp' (Hyndman, 2013) which, obviously, makes working through the examples presented in the book much easier.

As with any introduction, this one includes some necessary notation and terms which must be defined prior to actually learning any of the data analysis techniques. Say we have a vector of time series data, y, and there are nine values in this time series ($t = 9$). The most recent value is referred to as y_t and the last value as y_{t-8}. Continuing the notation, y_{t+1} is used when referring to a forecast value (i.e. the predicted next value of the time series). Next, there are a few terms worth noting. The term *trend* refers to a general pattern (e.g. increase or decrease) in the time series over the course of the series. Hyndman and Athanasopoulos (2015) define a trend as the following; "a trend exists when there is a long-term increase or decrease in the data" (p. 28). The term *seasonal* refers to patterns in the series which occur at regular intervals (e.g. season of the year, semesters of an academic year, days of the week, or even times of a day). Another term widely used is *cycle*, which "occurs when the data exhibit rises and falls that are not of a fixed period" (Hyndman and Athanasopoulos, p. 28). Basic time series are conceptually composed of either an additive model:

$$y_t = S_t + T_t + E_t \tag{1}$$

or a multiplicative model:

$$y_t = S_t * T_t * E_t \tag{2}$$

In both models, the y_t is the data at period t, S_t refers to the seasonal component at time t, the T_t refers to the trend (or cycle) component at time t, and the E_t refers to everything else (i.e. error) at time t. Hyndman and Athanasopoulos (2015) state that "the additive model is most appropriate if the magnitude of the seasonal fluctuations or the variation around the trend (or cycle) does not vary with the level of the time series" (p. 147). Alternatively, Hyndman and Athanasopoulos state that "when the variation in the seasonal pattern, or the variation around the trend (or cycle), appears to be proportional to the level of the time series, then the multiplicative model is more appropriate" (p. 147 – 148).

Below we will be using R Commander (package: 'Rcmdr') and the epack R Commander plugin (package: 'RcmdrPlugin.epack'), as well as one of the main time series packages (package: 'tseries'). Keep in mind, the R Commander package has several dependent packages; as does the epack plugin (including package: 'tseries'). The examples will be using monthly average stock price for BP PLC from January 1st, 2010 until January 1st, 2011. If you are wondering why only these 12 data points were chosen, please see[2]. Below we load R Commander and the epack plugin so we can import the data from Yahoo Finance. The function used to retrieve the data is the 'histprice2' function from the epack plugin.

---

[1] `https://www.otexts.org/book/fpp`
[2] `http://en.wikipedia.org/wiki/Deepwater_Horizon_oil_spill`

```
R Console (64-bit)                                                    [_][□][✕]
File  Edit  Misc  Packages  Windows  Help

> library(Rcmdr)
Loading required package: splines
Loading required package: RcmdrMisc
Loading required package: car
Loading required package: sandwich

Rcmdr Version 2.1-6

> library(RcmdrPlugin.epack)
Loading required package: TeachingDemos
Loading required package: tseries

    'tseries' version: 0.10-32

    'tseries' is a package for time series analysis and computational
    finance.

    See 'library(help="tseries")' for details.

Loading required package: abind
Loading required package: MASS
Loading required package: xts
Loading required package: zoo

Attaching package: 'zoo'

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

    as.Date, as.Date.numeric

Loading required package: forecast
Loading required package: timeDate
This is forecast 5.8

> bp_close <- histprice2(inst1 = "BP", quot1 = "Close", start1 = "2010-01-01",
+                        end1 = "2011-01-01")
trying URL 'http://chart.yahoo.com/table.csv?s=BP&a=0&b=01&c=2010&d=0&e=01&f=2011&g=m&q=q&y=0&z=BP$
Content type 'text/csv' length unknown
opened URL
downloaded 637 bytes

time series starts 2010-01-04
time series ends   2010-12-01
> |
```

It is worth noting that the 'histprice2' function is actually calling a function from the 'tseries' package called 'get.hist.quote' which; by default, uses Yahoo Finance data. This is important because although we are using monthly stock prices, the 'get.hist.quote' function is capable of retrieving daily historical data. As a quick example, consider the data imported below which contains the daily closing price of the S&P 500 from January 1964 until January 2014.

```
R Console (64-bit)                                                    [_][□][✕]
File  Edit  Misc  Packages  Windows  Help

> spc <- get.hist.quote(instrument = "^gspc", start = "1964-01-01",
+                        end = "2014-01-01", quote = "Close")
trying URL 'http://chart.yahoo.com/table.csv?s=^gspc&a=0&b=01&c=1964&d=0&e=01&f=2014&g=d&q=q&y=0&z$
Content type 'text/csv' length unknown
opened URL
downloaded 689 Kb

time series starts 1964-01-02
time series ends   2013-12-31
> length(spc)
[1] 12587
> |
```
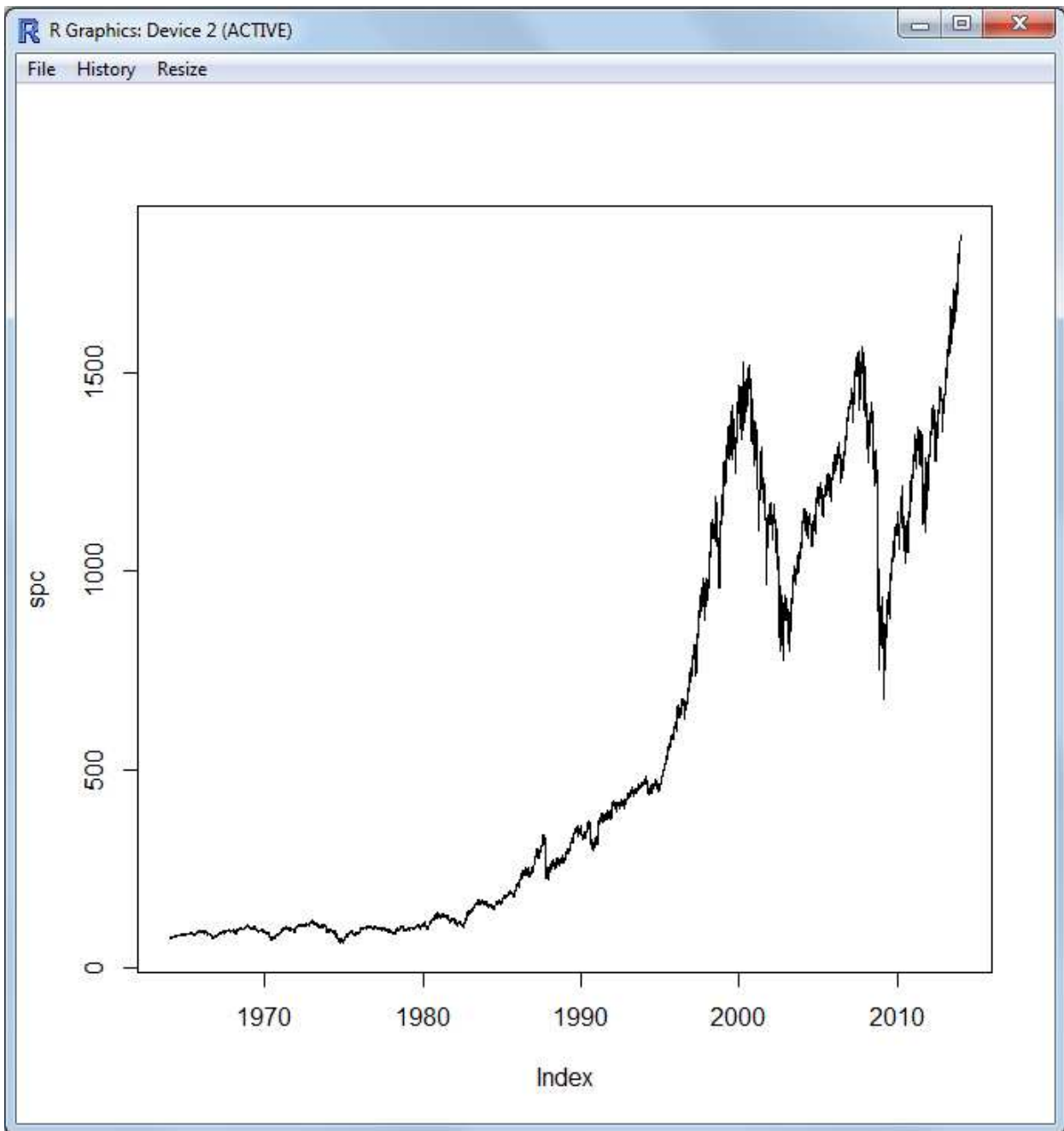
It is generally a good idea to begin with a graph of the data, while keeping in mind those terms from above (e.g. trend, seasonality, cycle). First, take a look at the S&P 500 data.

```
R Console (64-bit)                                                    [_][□][✕]
File  Edit  Misc  Packages  Windows  Help

> plot(spc)
> |
```

Now imagine you were attempting to forecast where the S&P 500 would be if you were in 1980. You would likely have very narrow prediction intervals (similar to confidence intervals). Contrast those imagined intervals with the intervals you would imagine based on the complete data in the graph, with those two ominous bunny ear spikes...very foreboding. Do you see any trends, seasonality, or cycles in the series? One way to get a better idea of those types of patterns is to plot segments of the data, only one decade at a time perhaps using four decades.

```
> par(mfrow = c(2,2))
> plot(spc[1485:4010])
> plot(spc[4010:6538])
> plot(spc[6538:9066])
> plot(spc[9066:11581])
> 
```
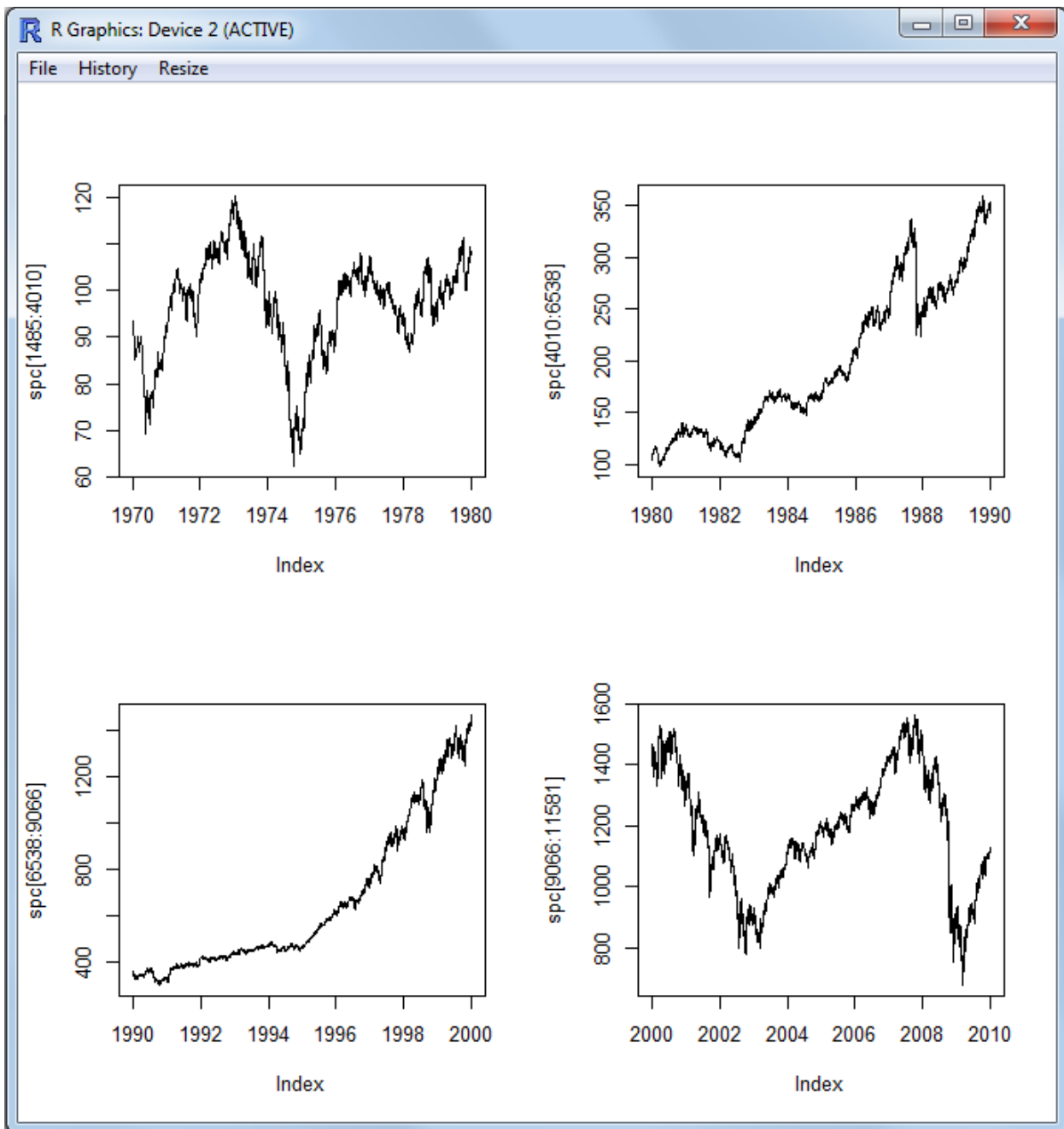
As you can see above, there does not seem to be any discernible patterns among the four segments. Of course, part of the problem above is that each of the four panels has a different y-axis scale; which means they are not directly comparable in terms of the variance in each series displayed. We could remedy that by forcing each graph to have the same scale, but let's turn our attention to a much smaller series of data; BP PLC average closing stock price for each month in 2010.
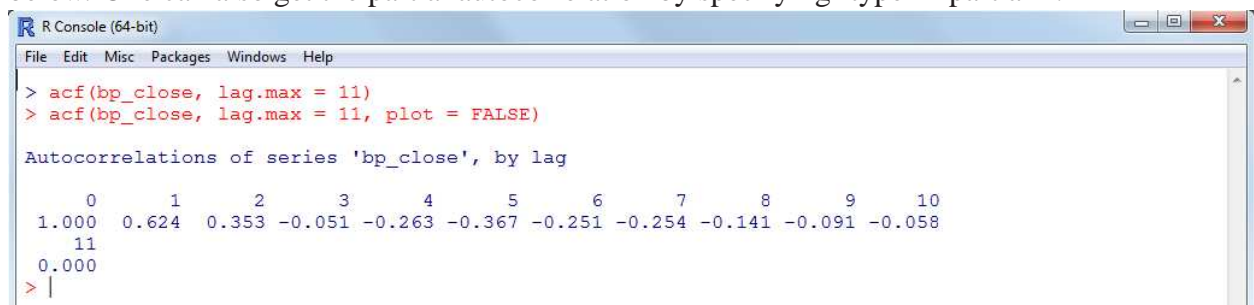


```
> graphics.off()
> plot(bp_close)
>
```
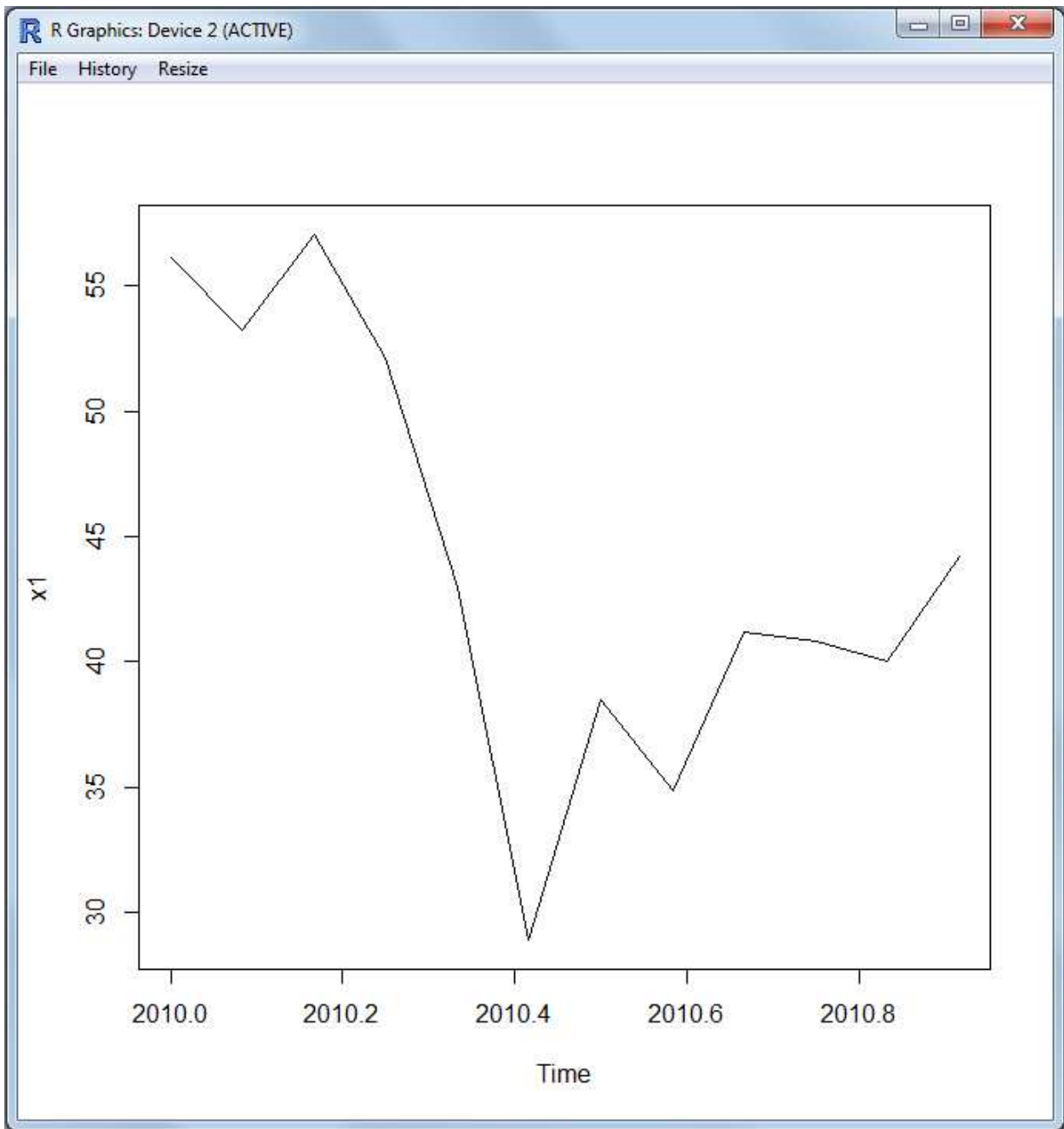
In the above graph we can see the stock price dropped nearly half its value between March and June. Most analysts likely would have predicted BP's stock to stay between $50 and $60 throughout 2010 – even when using complex multivariate models which will not be covered here. However, on April 20th, 2010 the Deepwater Horizon oil rig exploded and thus began one of the worst petroleum related oceanic environmental disasters. The point in highlighting this particular data set is to remind all of us that no matter how sophisticated the model, there is always uncertainty. Statistics is a tool for helping to make informed decisions in the presence of uncertainty; but models are not reality and no model is perfect. However, the more data available and analyzed; the less uncertainty one is likely to have in estimates resulting from a model. As the current article is to be an introduction, let's return to some of the more basic concepts of time series analysis.

1 Autocorrelation

Autocorrelation can be considered a measure of the momentum of a time series. In most time series, it is reasonable to suspect that the most recent data points are likely to contribute most influence on the next (i.e. future) data point. Autocorrelation is a type of correlation statistic specifically for correlating the most recent data point to other data points in the series. Recall, the most recent point is notated y_t and subsequently older points labeled y_{t-1}, y_{t-2}, y_{t-3} ... y_{t-k} (where $k = t - 1$). The maximum number of autocorrelations calculated is one minus the number of data points (i.e. k). Each autocorrelation represents a different lagged value – which refers to the number of points between the most recent data and the older data. Our BP data contains 12 values and therefore we can compute 11 autocorrelations (r_1, r_2, r_3, ... r_{11}). Graphing is generally the preferred method of inspecting the autocorrelations of a time series. The function used is simply 'acf' and by default it produces the desired graph. However, we can simply print the autocorrelations by changing 'plot = TRUE' (the default) to 'plot = FALSE' as seen below. One can also get the partial autocorrelation by specifying 'type = 'partial''.
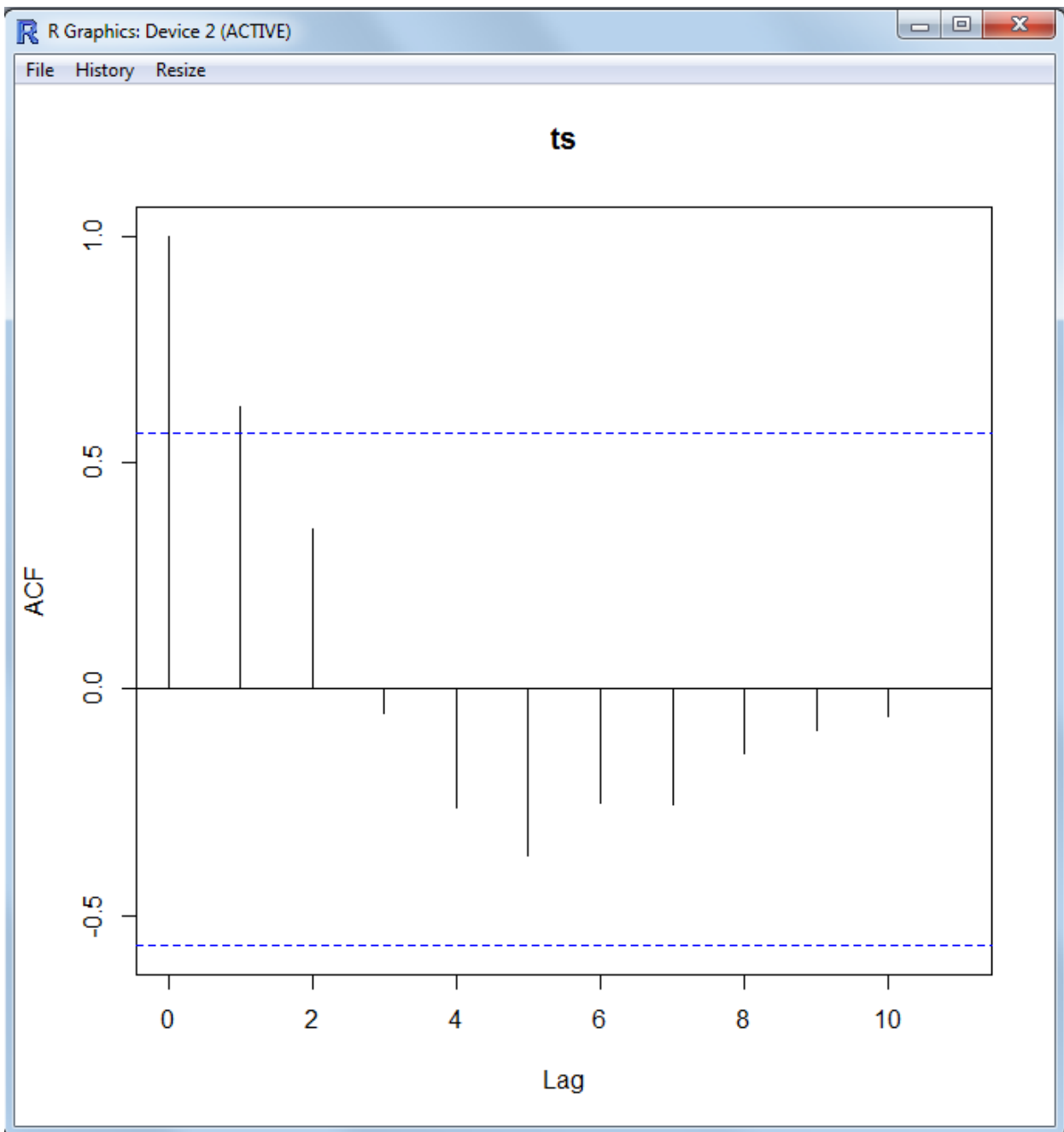
```
R Console (64-bit)

File  Edit  Misc  Packages  Windows  Help

> acf(bp_close, lag.max = 11)
> acf(bp_close, lag.max = 11, plot = FALSE)

Autocorrelations of series 'bp_close', by lag

     0      1      2      3      4      5      6      7      8      9     10
 1.000  0.624  0.353 -0.051 -0.263 -0.367 -0.251 -0.254 -0.141 -0.091 -0.058
    11
 0.000
>
```
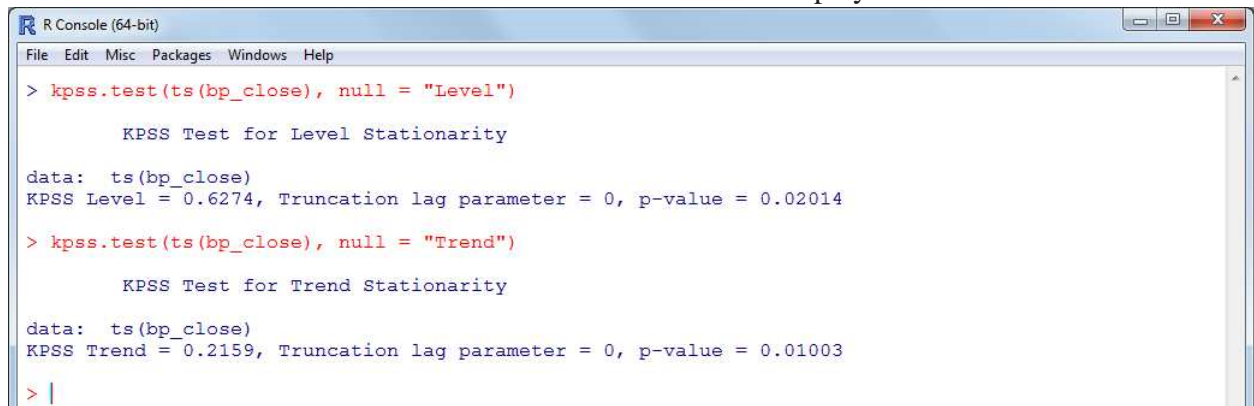
The blue dotted lines, in the plot above, represent a default confidence interval of 95% (i.e. 'ci = 0.95') which can be changed with the 'ci' argument (e.g. 'ci = 0.80' for 80%).

2 Stationarity and Non-Stationarity

An important concept in time series analysis is stationarity and particularly the recognition of non-stationarity in a particular time series. Stationarity refers to the idea that the time series fluctuates around a constant mean and the variance around that mean remains constant. As one might expect, most time series exhibit non-stationarity; in other words, most time series do not fluctuate around a fixed mean and they do not fluctuate uniformly. Fortunately, there is a function available in R to test for stationarity; the Kwiatkowski, Phillips, Schmidt, and Shin (KPSS; 1992) test for the null hypothesis that the time series

is level or trend stationary. Using the 'kpss.test' function actually requires running the function twice; once to test if the series is level and once to test if the series displays trend.

```
R R Console (64-bit)

File  Edit  Misc  Packages  Windows  Help

> kpss.test(ts(bp_close), null = "Level")

        KPSS Test for Level Stationarity

data:  ts(bp_close)
KPSS Level = 0.6274, Truncation lag parameter = 0, p-value = 0.02014

> kpss.test(ts(bp_close), null = "Trend")

        KPSS Test for Trend Stationarity

data:  ts(bp_close)
KPSS Trend = 0.2159, Truncation lag parameter = 0, p-value = 0.01003

> |
```

Both tests above indicate our time series is non-stationary (i.e. both *p*-values are less than 0.05); which indicates our series does not vary uniformly and does not vary around a constant mean. When a series is non-stationary (as the tests above indicate is the case with our example), then forecasting is much more difficult. However, a *differencing operation* can be used to transform a non-stationary series into a series where the assumptions of stationarity are met. For example, with the above result we would apply a $d = 1$ differencing operation first and then apply the KPSS tests to the resulting differences. The differencing operation simply takes the difference between each time datum ($d = 1$), or the difference between each other datum ($d = 2$), etc. If the resulting differenced time series shows stationarity (i.e. $p > 0.05$) then the models discussed below (e.g. ARIMA) are appropriate. The differencing operation is not used here due to the space and scope constraints of this document.

3   Auto-Regressive Model

The Auto-Regressive model (AR) is nothing more complex than a linear regression model for time series. The AR model essentially assumes the current time point datum is linearly related most to the previous point and subsequently less to each previous time point as specified in the model. When specifying an AR model, you must specify the *order* which indicates how many lags to use. The function for fitting an AR model in R is simply 'ar(data)', as can be seen below with no order specified – by default the order is 1 (i.e. AR-1). In the example below, the 'names' function is used to display the named objects which are returned by the 'ar' function. Of particular use is the 'partialacf' which returns the partial autocorrelations (simply type: "`ar(bp_close)$partialacf`" into the console to return these values). Also typically informative are the residuals of the model (simply type: "`ar(bp_close)$resid`" into the console to return these values); larger residuals indicate poorer fit.

```
R Console (64-bit)                                                          [_][□][X]
File  Edit  Misc  Packages  Windows  Help

> ar(bp_close, aic = TRUE, method = "yule-walker")

Call:
ar(x = bp_close, aic = TRUE, method = "yule-walker")

Coefficients:
      1
0.6241

Order selected 1  sigma^2 estimated as   51.73
> names(ar(bp_close))
 [1] "order"        "ar"            "var.pred"      "x.mean"        "aic"
 [6] "n.used"       "order.max"     "partialacf"    "resid"         "method"
[11] "series"       "frequency"     "call"          "asy.var.coef"
> |
```

4   Autoregressive-Moving-Average Models

Autoregressive-moving-average (ARMA) models are based on two polynomial functions; one for the autoregression (AR) and a second for the moving-average (MA). The basic function for fitting an ARMA model to a univariate time series is simply the 'arma' function as demonstrated below. Again, the function defaults to specify AR-1 and MA-1; order one (AR) and order two (MA) which reflect the lag periods for each component of the model.

```
R Console (64-bit)                                                          [_][□][X]
File  Edit  Misc  Packages  Windows  Help

> arma(bp_close)

Call:
arma(x = bp_close)

Coefficient(s):
      ar1        ma1    intercept
  0.65763   -0.09556   13.97203

> names(arma(bp_close))
 [1] "coef"              "css"               "n.used"
 [4] "residuals"         "fitted.values"     "series"
 [7] "frequency"         "call"              "vcov"
[10] "lag"               "convergence"       "include.intercept"
> summary(arma(bp_close))

Call:
arma(x = bp_close)

Model:
ARMA(1,1)

Residuals:
    Min      1Q  Median      3Q     Max
-13.832  -2.416   1.443   3.862   8.329

Coefficient(s):
          Estimate  Std. Error  t value  Pr(>|t|)
ar1        0.65763     0.21792    3.018   0.00255 **
ma1       -0.09556     0.41602   -0.230   0.81832
intercept 13.97203     9.88450    1.414   0.15750
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Fit:
sigma^2 estimated as 35.93,  Conditional Sum-of-Squares = 359.35,  AIC = 83.03

> |
```
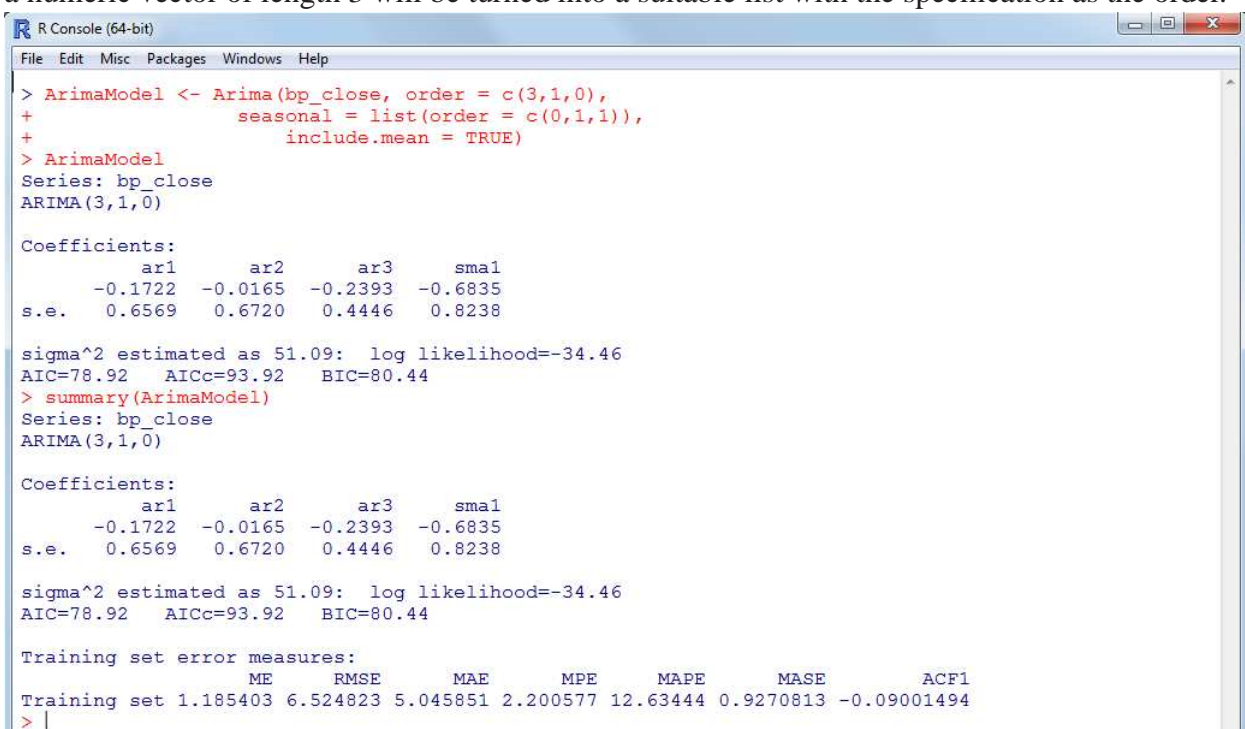
As can be seen in the summary above, only the autoregressive portion of the ARMA model is significant. As should be expected; the autoregressive coefficient is fairly close to what was observed in the AR model from the previous section.

5 Autoregressive Integrated Moving Average Models

The autoregressive integrated moving average (ARIMA) model is an appropriate choice (over the ARMA model) when non-stationarity is suspected or observed in the time series. The application of the 'Arima' function below has default values specified for the arguments. The 'Arima' function is supplied the time series data and two other components can be specified. The 'order' - which is the non-seasonal part of the ARIMA model, the three components (p, d, q) are the AR order, the degree of differencing (to correct for non-stationarity – as discussed at the beginning of this document), and the MA order. The 'seasonal' argument allows one to specify the seasonal part of the ARIMA model, plus the period (which defaults to `frequency(x)`). This should be a list with components order and period, but a specification of just a numeric vector of length 3 will be turned into a suitable list with the specification as the order.

```
R Console (64-bit)
File  Edit  Misc  Packages  Windows  Help

> ArimaModel <- Arima(bp_close, order = c(3,1,0),
+                   seasonal = list(order = c(0,1,1)),
+                   include.mean = TRUE)
> ArimaModel
Series: bp_close
ARIMA(3,1,0)

Coefficients:
          ar1      ar2      ar3     sma1
       -0.1722  -0.0165  -0.2393  -0.6835
s.e.    0.6569   0.6720   0.4446   0.8238

sigma^2 estimated as 51.09:  log likelihood=-34.46
AIC=78.92   AICc=93.92   BIC=80.44
> summary(ArimaModel)
Series: bp_close
ARIMA(3,1,0)

Coefficients:
          ar1      ar2      ar3     sma1
       -0.1722  -0.0165  -0.2393  -0.6835
s.e.    0.6569   0.6720   0.4446   0.8238

sigma^2 estimated as 51.09:  log likelihood=-34.46
AIC=78.92   AICc=93.92   BIC=80.44

Training set error measures:
                    ME      RMSE      MAE      MPE     MAPE      MASE        ACF1
Training set 1.185403 6.524823 5.045851 2.200577 12.63444 0.9270813 -0.09001494
>
```
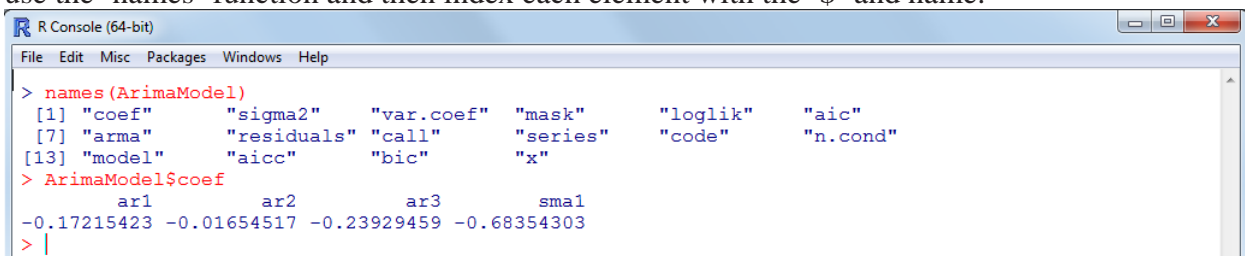
The output provided by simply listing the output object or retrieving a summary of the output object provides the basic information; primarily the coefficients (and their standard errors [s.e.]). To see all the elements of the output, in case one wanted to extract specific parts of it for further computation; we can use the 'names' function and then index each element with the '$' and name.

```
R Console (64-bit)
File  Edit  Misc  Packages  Windows  Help

> names(ArimaModel)
 [1] "coef"      "sigma2"    "var.coef"  "mask"      "loglik"    "aic"
 [7] "arma"      "residuals" "call"      "series"    "code"      "n.cond"
[13] "model"     "aicc"      "bic"       "x"
> ArimaModel$coef
        ar1         ar2         ar3        sma1
-0.17215423 -0.01654517 -0.23929459 -0.68354303
>
```
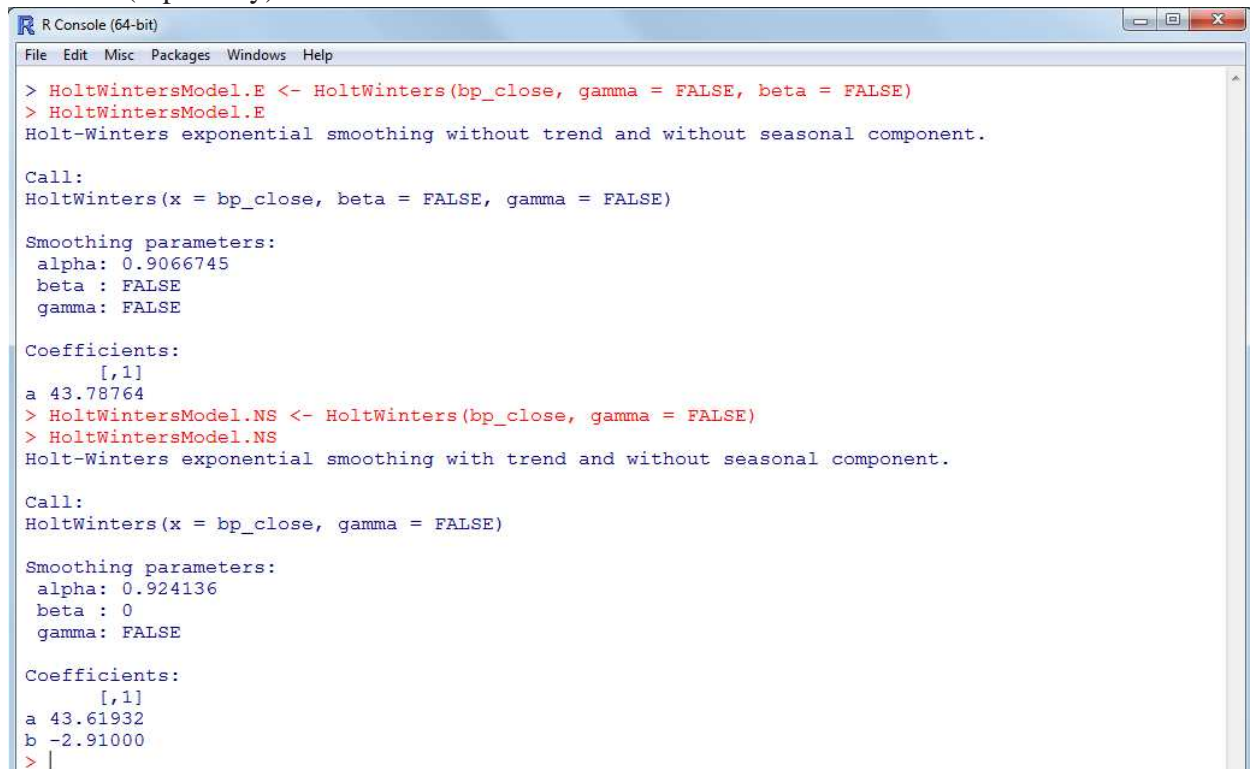
We can also apply some functions to use our ARIMA model to forecast predictions providing an estimate of the expected next time series point(s) and show that prediction graphically. The 'forecast' function provides 80% and 95% confidence intervals as well as a point estimate. Both functions used below are requesting only one time point ahead predictions; however, both functions are capable of forecasting multiple future time points.
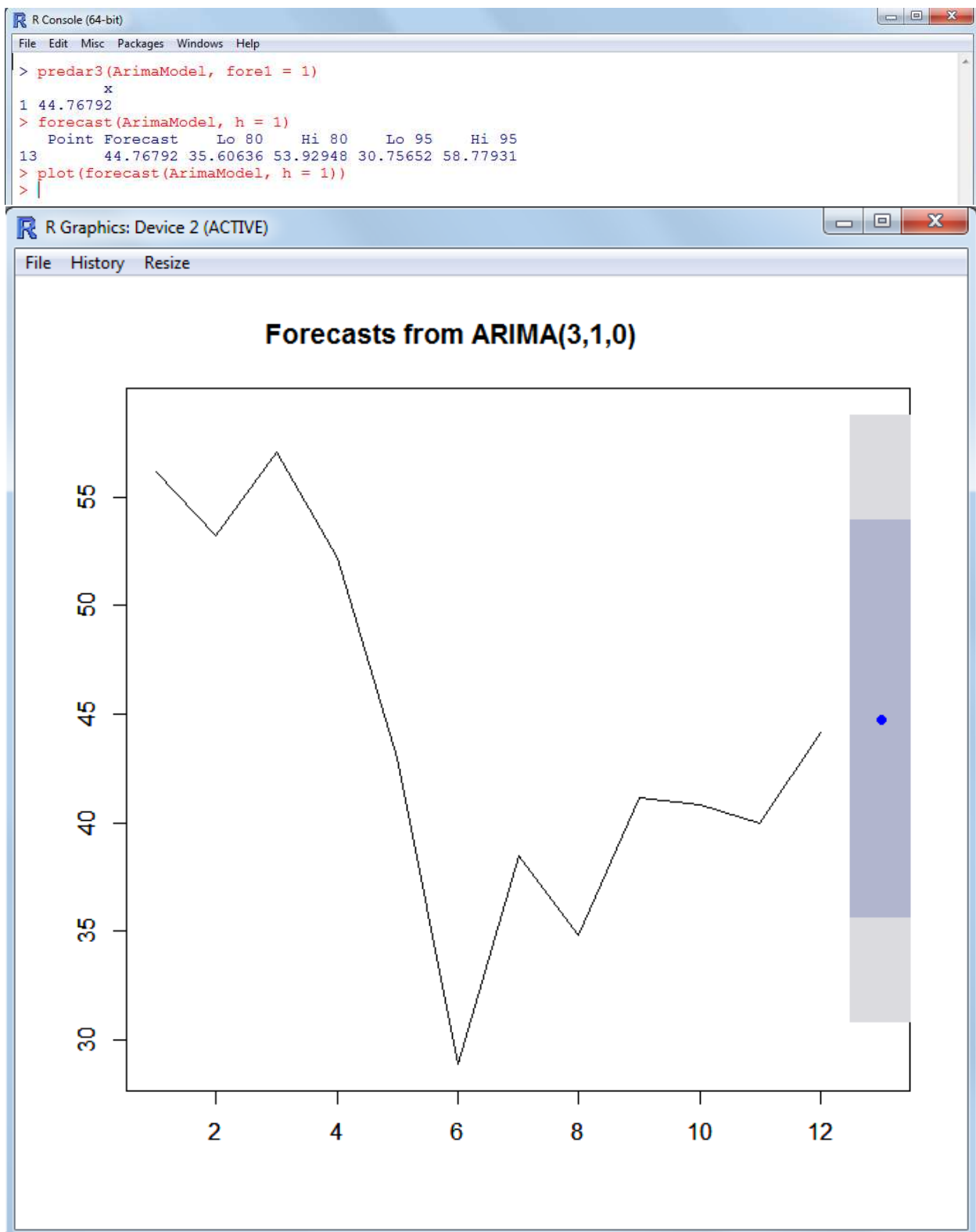
The rather large interval (shaded area) around the point estimate is reasonable given the rather large fluctuation of the existing series (i.e. the drastic decrease in price once the oil spill was made public). In the graph above, the blue shading represents the 80% confidence interval, while the gray shading represents the 95% interval.

There also ways of filtering the time series. Below the series is filtered with exponential smoothing

and then (separately) filtered as a non-seasonal model.

```
R Console (64-bit)

File  Edit  Misc  Packages  Windows  Help

> HoltWintersModel.E <- HoltWinters(bp_close, gamma = FALSE, beta = FALSE)
> HoltWintersModel.E
Holt-Winters exponential smoothing without trend and without seasonal component.

Call:
HoltWinters(x = bp_close, beta = FALSE, gamma = FALSE)

Smoothing parameters:
 alpha: 0.9066745
 beta : FALSE
 gamma: FALSE

Coefficients:
      [,1]
a 43.78764
> HoltWintersModel.NS <- HoltWinters(bp_close, gamma = FALSE)
> HoltWintersModel.NS
Holt-Winters exponential smoothing with trend and without seasonal component.

Call:
HoltWinters(x = bp_close, gamma = FALSE)

Smoothing parameters:
 alpha: 0.924136
 beta : 0
 gamma: FALSE

Coefficients:
      [,1]
a 43.61932
b -2.91000
> |
```

Lastly, much of the above has been covered on the RSS Do-it-yourself Introduction to R web site[3] and specifically in Module 10[4].

Until next time, *why are you wearing that stupid man suit?*

6   References and Resources

Fox, J. (2005). The R Commander: A basic statistics graphical user interface to R. *Journal of Statistical Software, 14*(9), 1 - 42. Documentation available at CRAN:
http://cran.r-project.org/web/packages/Rcmdr/index.html

Hodgess, E. (2012). Package RcmdrPlugin.epack. Documentation available at CRAN:
http://cran.r-project.org/web/packages/RcmdrPlugin.epack/index.html

Hyndman, R. J. (2013). Package fpp. Documentation available CRAN:
http://cran.r-project.org/web/packages/fpp/index.html

Hyndman, R. J., & Athanasopoulos, G. (2015). *Forecasting: principles and practice*. Freely available online at:
https://www.otexts.org/fpp

Hyndman, R. J. (2014). CRAN Task View: Time Series Analysis. Available at CRAN:

---

[3]http://www.unt.edu/rss/class/Jon/R_SC/
[4]http://www.unt.edu/rss/class/Jon/R_SC/Module9/BP_TimeSeries.R

http://cran.r-project.org/web/views/TimeSeries.html

Hyndman, Koehler, Ord, & Snyder (2008) *Forecasting with exponential smoothing* (3rd ed.). Wiley.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., & Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root. *Journal of Econometrics, 54*(1), 159 - 178. doi:10.1016/0304-4076(92)90104-Y. A free pdf of the 1992 paper is available at: http://www.deu.edu.tr/userweb/onder.hanedar/dosyalar/kpss.pdf

Makridakis, Wheelwright, & Hyndman (1998) *Forecasting: methods and applications*. Springer.

Trapletti, A., & Hornik, K. (2015). Package tseries. Documentation available at CRAN: http://cran.r-project.org/web/packages/tseries/index.html

This article was last updated on March 30, 2015.

This document was created using LaTeX