An application of the R Forecast Package in the Kaggle Competition Bike Sharing Demand

John Arhin
j_arhin@yahoo.co.uk

Monday 13th October 2014

Abstract

We show how the Forecast package in R can be used to solve a Kaggle Problem predicting the use of a city bike share system.

1 Introduction

1.1 Kaggle Problem

We acknowledge that the data for this problem is attributed to [1].

For the benefit of the reader, we reproduce the details of the Bike Sharing Demand Kaggle Competition as described in the URL `http://www.kaggle.com/c/bike-sharing-demand`.

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout a city. Using these systems, people are able rent a bike from a one location and return it to a different place on an as-needed basis. Currently, there are over 500 bike-sharing programs around the world.

The data generated by these systems makes them attractive for researchers because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city. In this Kaggle competition, we are asked to combine historical usage patterns with weather data in order to forecast bike rental demand in the Capital Bikeshare program in Washington, D.C.

1.1.1 Bike Data

We are provided hourly rental data spanning two years. For this Kaggle competition, the training set is comprised of the first 19 days of each month, while the test set is the 20th to the end of the month. We predict the total count of bikes rented during each hour covered by the test set, using only information available prior to the rental period.

Variable	Value	Meaning
datetime		hourly date + timestamp
season	1	Spring
	2	Summer
	3	Fall
	4	Winter
holiday		whether the day is considered a holiday
weather	1	Clear, Few clouds, Partly cloudy, Partly cloudy
	2	Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
	3	Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
	4	Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp		temperature in Celsius
atemp		"feels like" temperature in Celsius
humidity		relative humidity
windspeed		wind speed
casual		number of non-registered user rentals initiated
registered		number of registered user rentals initiated
count		number of total rentals

Table 1: Data Fields

We show the variables of the data-set together with their meaning in Table 1.

1.1.2 R Setup

In this report, we use an an Intel i7 PC with 16GB RAM. The operating system is the Linux distribution OpenSUSE.

We use R [1] Version 3.1.1.

We use R together with Project Template [3], ggplot2 [4] and the Forecast [5] packages.

2 Exploratory Data analysis

3 Methodology

4 Results

5 Conclusion

This is a demo for using the `Sweave` command in R. To get started make a regular LaTeX file (like this one) but give it the suffix `.Rnw` instead of `.tex` and then turn it into a LaTeX file (`foo.tex`) with the (unix) command

```
R CMD Sweave foo.Rnw
```

So you can do

```
latex foo
xdvi foo
```

and so forth.

So now we have a more complicated file chain

$$\texttt{foo.Rnw} \xrightarrow{\texttt{Sweave}} \texttt{foo.tex} \xrightarrow{\texttt{latex}} \texttt{foo.dvi} \xrightarrow{\texttt{xdvi}} \text{view of document}$$

and what have we accomplished other than making it twice as annoying to the WYSIWYG crowd (having to run both `Sweave` and `latex` to get anything that looks like the document)?

Well, we can now include R in our document. Here's a simple example

```
> 2 + 2
```

```
[1] 4
```

What I actually typed in `foo.Rnw` was

```
<<two>>=
2 + 2
@
```

This is not LaTeX. It is a "code chunk" to be processed by `Sweave`. When `Sweave` hits such a thing, it processes it, runs R to get the results, and stuffs (by default) the output in the LaTeX file it is creating. The LaTeX between code chunks is copied verbatim (except for `Sexpr`, about which see below). Hence to create a Rnw document you just write plain old LaTeX interspersed with "code chunks" which are plain old R.

Plots get a little more complicated. First we make something to plot (simulate regression data).

```
> n <- 50
> x <- seq(1, n)
> a.true <- 3
> b.true <- 1.5
> y.true <- a.true + b.true * x
> s.true <- 17.3
> y <- y.true + s.true * rnorm(n)
> out1 <- lm(y ~ x)
> summary(out1)

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-48.394 -10.485   2.034  12.143  45.624

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  10.0625     4.8730   2.065   0.0444 *
x             1.3774     0.1663   8.282 8.29e-11 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.97 on 48 degrees of freedom
Multiple R-squared:  0.5883,        Adjusted R-squared:  0.5797
F-statistic: 68.59 on 1 and 48 DF,  p-value: 8.287e-11
```

(for once we won't show the code chunk itself, look at `foo.Rnw` if you want to see what the actual code chunk was).

Figure 1 (p. 5) is produced by the following code

```
> plot(x, y)
> abline(out1)
```

Note that `x`, `y`, and `out1` are remembered from the preceding code chunk. We don't have to regenerate them. All code chunks are part of one R "session".

Now this was a little tricky. We did this with two code chunks, one visible and one invisible. First we did

```
<<label=fig1plot,include=FALSE>>=
plot(x, y)
abline(out1)
@
```

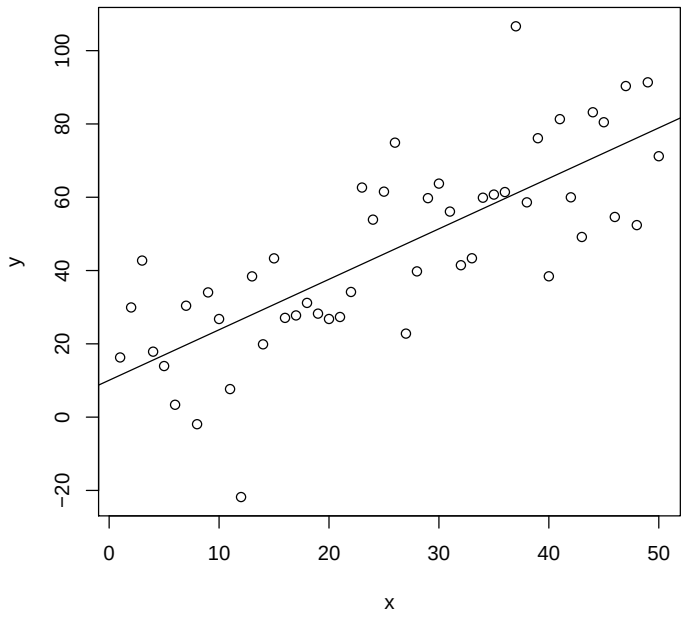

Figure 1: Scatter Plot with Regression Line

where the `include=FALSE` indicates that the output (text and graphics) should not go here (they will be some place else) and the `label=fig1plot` gives the code chunk a name (to be used later). And "later" is almost immediate. Next we did

```
\begin{figure}
\begin{center}
<<label=fig1,fig=TRUE,echo=FALSE>>=
<<fig1plot>>
@
\end{center}
\caption{Scatter Plot with Regression Line}
\label{fig:one}
\end{figure}
```

In this code chunk the `fig=TRUE` indicates that the chunk generates a figure. `Sweave` automagically makes both EPS and PDF files for the figure and automagically generates an appropriate LaTeX `\includegraphics` command to include the plot in the `figure` environment. The `echo=FALSE` in the code chunk means just what it says (we've already seen the code—it was produced by the preceding chunk—and we don't want to see it again, especially not in our figure). The `<<fig1plot>>` is an example of "code chunk reuse". It means that we reuse the code of the code chunk named `fig1plot`. It is important that we observe the DRY/SPOT rule (*don't repeat yourself* or *single point of truth*) and only have one bit of code for generating the plot. What the reader sees is guaranteed to be the code that made the plot. If we had used cut-and-paste, just repeating the code, the duplicated code might get out of sync after edits. The rest of this should be recognizable to anyone who has ever done a LaTeX figure.

So making a figure is a bit more complicated in some ways but much simpler in others. Note the following virtues

- The figure is guaranteed to be the one described by the text (at least by the R in the text).
- No messing around with sizing or rotations. It just works!

Note that if you don't care to show the R code to make the figure, it is simpler still. Figure 2 (p. 7) shows another plot. What I actually typed in `foo.Rnw` was

```
\begin{figure}
\begin{center}
<<label=fig2,fig=TRUE,echo=FALSE>>=
out3 <- lm(y ~ x + I(x^2) + I(x^3))
plot(x, y)
curve(predict(out3, newdata=data.frame(x=x)), add = TRUE)
@
```

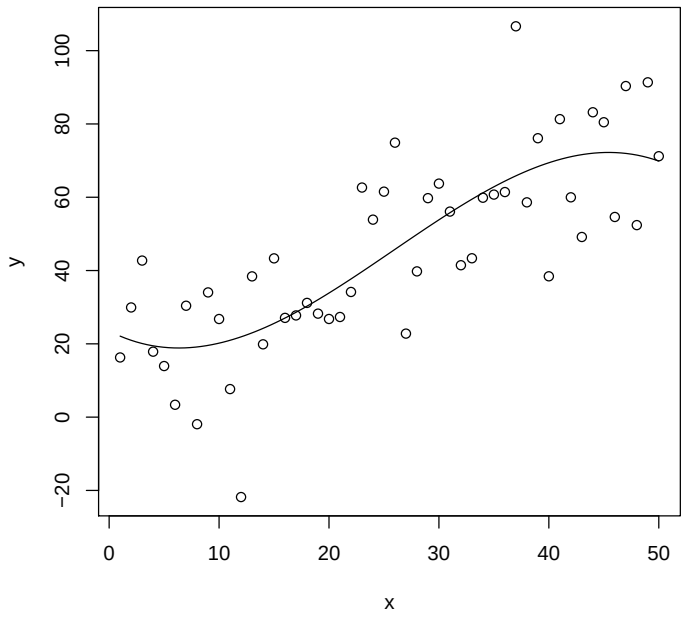

Figure 2: Scatter Plot with Cubic Regression Curve

```
\end{center}
\caption{Scatter Plot with Cubic Regression Curve}
\label{fig:two}
\end{figure}
```

Now we just included the code for the plot in the figure (with `echo=FALSE` so it doesn't show).

Also note that every time we rerun `Sweave` Figures 1 and 2 change, the latter conspicuously (because the simulated data are random). Everything just works. This should tell you the main virtue of Sweave. It's always correct. There is never a problem with stale cut-and-paste.

Simple numbers can be plugged into the text with the `\Sexpr` command, for example, the quadratic and cubic regression coefficients in the preceding regression were $\beta_2 = 0.1385$ and $\beta_3 = -0.0018$. Just magic! What I actually typed in `foo.Rnw` was

```
in the preceding regression
were $\beta_2 = \Sexpr{round(out3$coef[3], 4)}$
and $\beta_3 = \Sexpr{round(out3$coef[4], 4)}$.
```

The `xtable` command is used to make tables. (The following is the `Sweave` of another code chunk that we don't explicitly show. Look at `foo.Rnw` for details.)

```
> out2 <- lm(y ~ x + I(x^2))
> foo <- anova(out1, out2, out3)
> foo

Analysis of Variance Table

Model 1: y ~ x
Model 2: y ~ x + I(x^2)
Model 3: y ~ x + I(x^2) + I(x^3)
  Res.Df   RSS Df Sum of Sq      F  Pr(>F)
1     48 13824
2     47 13817  1      7.35 0.0261 0.87230
3     46 12934  1    882.85 3.1398 0.08302 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> class(foo)

[1] "anova"      "data.frame"

> dim(foo)

[1] 3 6

> foo <- as.matrix(foo)
> foo
```

Table 2: ANOVA Table

	Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
1	48	13824.30				
2	47	13816.95	1	7.35	0.026	0.872
3	46	12934.10	1	882.85	3.140	0.083

```
  Res.Df      RSS Df  Sum of Sq          F     Pr(>F)
1     48 13824.30 NA         NA         NA         NA
2     47 13816.95  1   7.346281 0.02612697 0.87229849
3     46 12934.10  1 882.849512 3.13984517 0.08302466
```

So now we are ready to turn the matrix `foo` into Table 2 using the R chunk

```
<<label=tab1,echo=FALSE,results=tex>>=
library(xtable)
print(xtable(foo, caption = "ANOVA Table", label = "tab:one",
    digits = c(0, 0, 2, 0, 2, 3, 3)), table.placement = "tbp",
    caption.placement = "top")
@
```

(note the difference between arguments to the `xtable` function and to the `xtable` method of the `print` function).

To summarize, `Sweave` is terrific, so important that soon we'll not be able to get along without it. It's virtues are

- The numbers and graphics you report are actually what they are claimed to be.
- Your analysis is reproducible. Even years later, when you've completely forgotten what you did, the whole write-up, every single number or pixel in a plot is reproducible.
- Your analysis actually works—at least in this particular instance. The code you show actually executes without error.
- Toward the end of your work, with the write-up almost done you discover an error. Months of rework to do? No! Just fix the error and rerun `Sweave` and `latex`. One single problem like this and you will have all the time invested in `Sweave` repaid.
- This methodology provides discipline. There's nothing that will make you clean up your code like the prospect of actually revealing it to the world.

Whether we're talking about homework, a consulting report, a textbook, or a research paper. If they involve computing and statistics, this is the way to do it.

References

[1] R Core Team (2014). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

[2] Fanaee-T, Hadi, and Gama, Joao, Event labeling combining ensemble detectors and background knowledge, Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg.

[3] ProjectTemplate: Automates the creation of new statistical analysis projects. R package; Version 0.6 http://projecttemplate.net

[4] H. Wickham. ggplot2: elegant graphics for data analysis. Springer New York, 2009. http://had.co.nz/ggplot2/book

[5] RJ Hyndman, Y Khandakar, Automatic time series for forecasting: the forecast package for R, Monash University, Department of Econometrics and Business Statistics, 2007, http://robjhyndman.com/software/forecast/