



R news and tutorials contributed by (573) R bloggers

- [Home](#)
- [About](#)
- [RSS](#)
- [add your blog!](#)
- [Learn R](#)
- [R jobs](#)
- [Contact us](#)

Welcome!

Follow @rbloggers { 30.9K }

Here you will find daily **news and tutorials about R**, contributed by over 573 bloggers. There are many ways to **follow us** - [By e-mail:](#)

28725 readers
BY FEEDBURNER

[On Facebook:](#)

R blogg..
32 K mentions J'

Soyez le premier de vos amis à aimer ça.

If you are an R blogger yourself you are invited to [add your own R content feed to this site](#) (Non-English R bloggers should add themselves- [here](#))

[Jobs for R-users](#)

- [Data Scientist for ARMUS @ California](#)
- [Data Services Adjunct Librarian: Adjunct Quantitative Data Analysis Specialist](#)
- [Data Scientist @ England](#)
- [Advanced Analytics Consultant @ Maryland](#)
- [Research Associate \(Bioinformatician\) @ London](#)

Search & Hit Enter

Popular Searches

- [web scraping](#)
- [heatmap](#)
- [maps](#)
- [twitter](#)
- [time series](#)
- [boxplot](#)
- [animation](#)
- [shiny](#)
- [hadoop](#)
- [how to import image file to R](#)
- [ggplot2](#)
- [ggplot](#)
- [trading](#)
- [latex](#)
- [finance](#)
- [excel](#)
- [eclipse](#)
- [rattle](#)
- [googlevis](#)
- [pca](#)
- [SQL](#)
- [RSTUDIO](#)
- [quantmod](#)
- [knitr](#)
- [market research](#)
- [Regression](#)
- [tutorial](#)
- [coplot](#)
- [rcmdr](#)
- [map](#)

Recent Posts

- [First Thoughts on Detecting Motorsport Safety Car Periods from Laptimes](#)
- [New R package for K-S goodness-of-fit tests](#)
- [When Documents Become Databases – Tabulizer R Wrapper for Tabula PDF Table Extractor](#)
- [Zooming](#)
- [New features in imager 0.20](#)
- [Pining for the fjoRds & monitoring SSL/TLS certificate expiration in R with flexdashboard](#)
- [New nasadata R package](#)
- [Register now for Hadley Wickham's Master R in Amsterdam](#)
- [Lattice exercises – part 1](#)
- [Base R Nostalgia – by, tapply, ave, ...](#)
- [Identify, describe, plot, and remove the outliers from the dataset](#)
- [First Thoughts on Automatically Generating Accessible Text Descriptions of ggplot Charts in R](#)

- [Tufte-style graphics in R](#)
- [Cross-Validation: Estimating Prediction Error](#)
- [testthat 1.0.0](#)

Other sites

- [Statistics of Israel](#)
- [Jobs for R-users](#)
- [SAS blogs](#)

Integrating Python and R Part III: An Extended Example

December 18, 2015

By [Mango Blogger](#)

Like Share 8 Tweet Share 9

(This article was first published on [Mango Solutions](#), and kindly contributed to [R-bloggers](#))

By Chris Musselle

This is the third post in a three part series where I have explored the options available for including both R and Python in a data analysis pipeline. See [post one](#) for some reasons on why you may wish to do this, and details of a general strategy involving flat files. [Post two](#) expands on this by showing how R or Python processes can call each other and parse arguments between them.

In this post I will be sharing a longer example using these approaches in analysis we carried out at Mango as a proof of concept to cluster news articles. The pipeline involved the use of both R and Python at different stages, with a Python script being called from R to fetch the data, and the exploratory analysis piece being conducted in R.

Full implementation details can be found in the repository [on github here](#), though for brevity this article will focus on the core concepts with the most relevant parts to R and Python integration discussed below.

Document Clustering

We were interested in the problem of document clustering of live published news articles, and specifically, wished to investigate times when multiple news websites were talking about the same content. As a first step towards this, we looked at sourcing live articles via RSS feeds, and used text mining methods to preprocess and cluster the articles based on their content.

Sourcing News Articles From RSS Feeds

There are some great Python tools out there for scraping and sourcing web data, and so for this task we used a combination of `feedparser`, `requests`, and `BeautifulSoup` to process the RSS feeds, fetch web content, and extract the parts we were interested in. though the general code structure was as follows:

```
# fetch_RSS_feed.py

def get_articles(feed_url, json_filename='articles.json'):
    """Update JSON file with articles from RSS feed"""
    #
    # See github link for full function script
    #

if __name__ == '__main__':

    # Pass Arguments
    args = sys.argv[1:]
    feed_url = args[0]
    filepath = args[1]

    # Get the latest articles and append to the JSON file given
    get_articles(feed_url, filepath)
```

Here we can see that the `get_articles` function is defined to perform the bulk of the data sourcing tasks, and that the parameters passed to it

are the positional arguments from the command line. Within `get_articles`, the url link, publication date, title and text contents, were then extracted for each article in the RSS feed and stored in a JSON file. For each article, the text content was made up of all HTML paragraph tags within the news article.

Sidenote: The `if __name__ == "__main__":` line may look strange to non-Python programmers, but this is a common way in Python scripts to control the sections of the code that are run when the whole script is executed, vs when the script is imported by another Python script. If the script is executed directly (as is the case when it is called from R later), the if statement evaluates to true and all code is run. If however, at some point in the future I wanted to reuse `get_articles` in another Python script, I could now import that function from this script without triggering the code within the if statement.

The above Python script was then executed from within R by defining the utility function shown below. Note that by using `stdout=TRUE`, any messages printed to stdout with `print()` in the Python code, can be captured and displaced in the R console.

```
fetch_articles <- function(url, filepath) {
  command = "python"
  path2script="fetch_RSS_feed.py"

  args = c(url, filepath)
  allArgs = c(path2script, args)

  output = system2(command, args=allArgs, stdout=TRUE)
  print(output)
}
```

Loading Data into R

Once the data had been written to a JSON file, the next job was to get it into R to be used with the `tm` package for text mining. This proved a little trickier than first expected however, as the `tm` package is mainly geared around reading in documents from raw text files, or directories containing multiple text files. To convert the JSON file into the expected `VCorpus` object for `tm` I used the following:

```
load_json_file <- function(filepath) {

  # Load data from JSON
  json_file <- file(filepath, "rb", encoding = "UTF-8")
  json_obj <- fromJSON(json_file)
  close(json_file)

  # Convert to VCorpus
  bbc_texts <- lapply(json_obj, FUN = function(x) x$text )
  df = as.data.frame(bbc_texts)
  df = t(df)
  articles = VCorpus(DataframeSource(df))
  articles
}
```

Unicode Woes

One potential problem when manipulating text data from a variety of sources and passing it between languages, is that you can easily get tripped up by character encoding errors on route. We found that by default Python was able to read in, process and write out the article content from the HTML sources, but R was struggling to decode certain characters that were written out to the resulting JSON file. This is due to the languages using or expecting a different character encoding by default.

To remedy this, you should be explicit in the encoding you are using when writing and reading a file, by specifying it when opening a file connection. This meant using the following in Python when writing out to a JSON file,

```
# Write updated file.
with open(json_filename, 'w', encoding='utf-8') as json_file:
    json.dump(JSON_articles, json_file, indent=4)
```

and on the R side opening the file connection was as follows:

```
# Load data from JSON
json_file <- file(filepath, "rb", encoding = "UTF-8")
json_obj <- fromJSON(json_file)
close(json_file)
```

Here “UTF-8” Unicode is chosen as it is a good default encoding to use, and is the most popular one used in [HTML documents worldwide](http://www.w3.org/International/questions/qa-unicode-universal).

For more details on Unicode and ways of handling it in Python 2 and 3 [see Ned Batchelder's PyCon talk here](#).

Summary of Text Preprocessing and Analysis

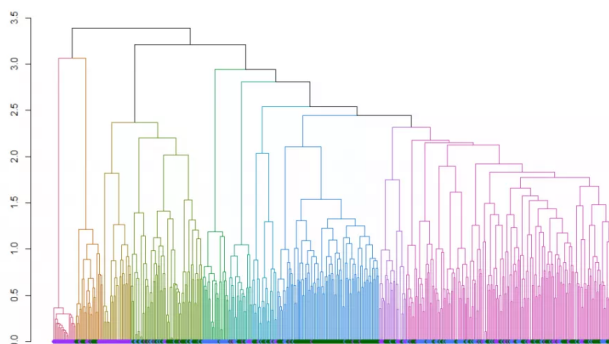
The text preprocessing part of the analysis consisted of the following steps, which were all carried out using the `tm` package in R:

- Tokenisation – Splitting text into words.
- Punctuation and whitespace removal.
- Conversion to lowercase.
- Stemming – to consolidate different word endings.
- Stopword removal – to ignore the most common and therefore least informative words.

Once cleaned and processed, the [Term Frequency-Inverse Document Frequency \(TF-IDF\)](#) statistic was calculated for the collection of articles. This statistic aims to provide a measure of how important each word is for a particular document, across a collection of documents. It is more sophisticated than just using the word frequencies themselves, as it takes into account that some words may naturally occur more frequently than others across all documents.

Finally a distance matrix was constructed based on the TF-IDF values and hierarchical clustering was performed. The results were then visualised as a dendrogram using the [dendextend](#) package in R.

An example of the clusters formed from 475 articles published over the last 4 days is shown below where the leaf nodes are coloured according to their source, with blue corresponding to BBC News, green to The Guardian, and indigo to The Independent.



It is interesting here to see articles from the same news websites occasionally forming groups, suggesting that news websites often post multiple articles with similar content, which is plausible considering how news stories unfold over time.

What's more interesting is finding clusters where multiple news websites are talking about similar things. Below is one such cluster with the article headlines displayed, which mostly relate to the recent flooding in Cumbria.



Hierarchical clustering is often a useful step in exploratory data analysis, and this work gives some insight into what is possible with news article clustering from live RSS feeds. Future work will look to evaluate different clustering approaches in more detail by examining the quality of the clusters they produce.

Other Approaches

In this series we have focused on describing the simplest approach of using flat files as an intermediate storage medium between the two languages. However it is worth briefly mentioning several other options that are available, such as:

- Using a database, such as [sqlite](#), as a medium of storage instead of flat files.
- Passing the results of a script execution in memory instead of writing to an intermediate file.
- Running two persistent R and Python processes at once, and passing data between them. Libraries such as [rpy2](#) and [rPython](#) provide one such way of doing this.

Each of these methods brings with it some additional pros and cons, and so the question of which is most suitable is often dependent on the application itself. As a first port of call though, using common flat file formats is a good place to start.

Summary

This post gave an extended example of how Mango have been using both Python and R to perform exploratory analysis around clustering news articles. We used the flat file air gap strategy described in [the first post](#) in this series, and then automated the calling of Python from R by spawning a separate subprocess (described in the [second post](#)). As can be seen with a bit of care around character encodings, this provides a straight forward approach to “bridging the language gap”, and allows multiple skillsets to be utilised when performing a piece of analysis.


Like Share { 8 } Tweet [Share](#) 9

Related

[Integrating Python and R Part II – Executing R from Python and Vice Versa](#)

By Chris Musselle In a previous article we went over why you might want to integrate both R and Python into In "R bloggers"

[Integrating Python and R into a Data Analysis Pipeline – Part I](#)
For a conference in the R language, EARL London 2015 saw a surprising number of discussions about In "R bloggers"

[Stepping up to Big Data with R and Python: A Mind Map of All the Packages You Will Ever Need](#)
Stepping up to Big Data with R and Python: A Mind Map of All the Packages You Will Ever Need
In "R bloggers"

{ 8 } Tweet [Share](#) 9
Like

To **leave a comment** for the author, please follow the link and comment on their blog: [Mango Solutions](#).

[R-bloggers.com](#) offers [daily e-mail updates](#) about [R](#) news and [tutorials](#) on topics such as: [Data science](#), [Big Data](#), [R jobs](#), visualization ([ggplot2](#), [Boxplots](#), [maps](#), [animation](#)), programming ([RStudio](#), [Sweave](#), [LaTeX](#), [SQL](#), [Eclipse](#), [git](#), [hadoop](#), [Web Scraping](#)) statistics ([regression](#), [PCA](#), [time series](#), [trading](#)) and more...

If you got this far, why not **subscribe for updates** from the site?
Choose your flavor: [e-mail](#), [twitter](#), [RSS](#), or [facebook](#)...

Like Share { 8 } Tweet [Share](#) 9

Comments are closed.

Search & Hit Enter

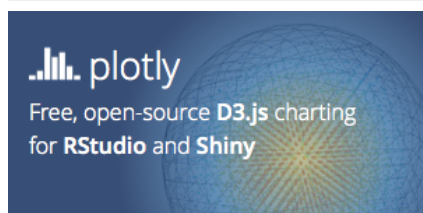
Recent popular posts

- [New nasadata R package](#)
- [How to write the first for loop in R](#)
- [R tutorials](#)
- [Identify, describe, plot, and remove the outliers from the dataset](#)

Most visited articles of the week

1. [How to write the first for loop in R](#)
2. [The one machine learning concept you need to know](#)
3. [Installing R packages](#)
4. [A Data Scientist's Perspective on Microsoft R](#)
5. [R tutorials](#)
6. [How to perform a Logistic Regression in R](#)
7. [In-depth introduction to machine learning in 15 hours of expert videos](#)
8. [Computing and visualizing PCA in R](#)
9. [Using apply, sapply, lapply in R](#)

Sponsors



[Plotly: collaborative, publication-quality graphing.](#)



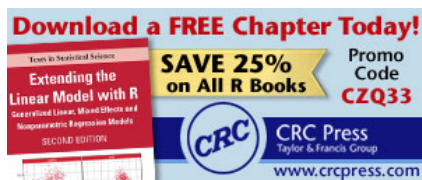
Quantide: statistical consulting and training



Werden Sie zum Expe[R]ten mit der
R-Akademie von



Beratung | Software
Training | Lösungen



[Contact us](#) if you wish to help support R-

bloggers, and place your banner here.

[Jobs for R users](#)

- [Data Scientist for ARMUS @ California](#)
- [Data Services Adjunct Librarian: Adjunct Quantitative Data Analysis Specialist](#)
- [Data Scientist @ England](#)
- [Advanced Analytics Consultant @ Maryland](#)
- [Research Associate \(Bioinformatician\) @ London](#)
- [Senior Process Engineer/Data Scientist @ Maryland](#)
- [Summer 2016 intern for Distributed Data Structures in R](#)

[Full list of contributing R-bloggers](#)

[R-bloggers](#) was founded by [Tal Galili](#), with gratitude to the [R](#) community.

Is powered by [WordPress](#) using a [bavotasan.com](#) design.

Copyright © 2016 **R-bloggers**. All Rights Reserved. [Terms and Conditions](#) for this website