Regular Expressions Every R programmer Should Know

Posted on April 11, 2018 by **R on The Jumping Rivers Blog** in **R bloggers** | 0 Comments

[This article was first published on **R on The Jumping Rivers Blog**, and kindly contributed to R-bloggers]. (You can report issue about the content on this page here)

Want to share your content on R-bloggers? click here if you have a blog, or here if you don't.

Share Tweet

- Regex: The backslash, `\`
- Regex: The hat , `^`, and dollar, `$`
- Regex: Round parentheses, `()`, and the pipe, `|`
- Regex: Square parentheses, `[]`, and the asterisk, `*`

Regular expressions. How they can be cruel! Well we're here to make them a tad easier. To do so we're going to make use of the **stringr** package

```
install.packages("stringr")
library("stringr")
```

We're going to use the `str_detect()` and `str_subset()` functions. In particular the latter. These have the syntax

```
function_name(STRING, REGEX_PATTERN)
```

`str_detect()` is used to detect whether a string contains a certain pattern. At the most basic use of these functions, we can match strings of text. For instance

```
jr = c("Theo is first", "Esther is second", "Colin - third")
str_detect(jr, "Theo")
## [1]  TRUE FALSE FALSE
str_detect(jr, "is")
## [1]  TRUE  TRUE FALSE
```

So `str_detect()` will return `TRUE` when the element contains the pattern we searched for. If we want to return the actual strings that contain these patterns, we use `str_subset()`

```
str_subset(jr, "Theo")
## [1] "Theo is first"
str_subset(jr, "is")
## [1] "Theo is first"    "Esther is second"
```

To practise our regex, we'll need some text to practise on. Here we have a vector of filenames called `files`

```
files = c(
  "tmp-project.csv", "project.csv",
```

Search R-bloggers.. Go
Your e-mail here Subscribe 52793 readers
Follow @rbloggers 101K followers
R bloggers Like Page 80K likes

Most viewed posts (weekly)

Which data science skills are important ($50,000 increase in salary in 6-months)
PCA vs Autoencoders for Dimensionality Reduction
Better Sentiment Analysis with sentiment.ai
Self-documenting plots in ggplot2
5 Ways to Subset a Data Frame in R
How to write the first for loop in R
Dashboards in R Shiny

Sponsors

EARL EARLy birds: Buy your tickets now! 6-8th September 2022 Tower Hotel, London #earlconf

Learn R by doing datacamp Start For Free

COMPONENT SEARCH ENGINE Access PCB Symbols, Footprints, and 3D Models

I'm also going to give us a task. The task is to be able to grab the files that have a format "project-objects" or "project_objects". Let's say of those files we want the csv and ods files. i.e. we want to grab the files "project_cars.ods", "project-houses.csv" and "project_Trees.csv". As we introduce more regex we'll gradually tackle our task.

Regex: The backslash, `\`

Here we go! Our first regular expression. When typing regular expressions, there are a group of special characters called metacharacters that have other functions. These are:

```
.{()\^$|?*+
```

The backslash is *SUPER* important because if we want to search for any of these characters without using their built in function we must escape the character with a backslash. For example, if we wanted to extract the names of the name of all csv files then perhaps we would think to search for the string ".csv"? Then we would do

```
str_subset(files, "\.csv") # xXx error = TRUE isn't working
```

Hang on a second, what? Ah yes. The backslash is a metacharacter too! So to create a backslash for the function to search with, we need to escape the backslash!

```
str_subset(files, "\\.csv")
## [1] "tmp-project.csv"          "project.csv"
## [3] "project2-csv-specs.csv"   "project2.csv2.specs.xlsx"
## [5] "project-houses.csv"       "Project_Trees.csv"
```

Much better. With regards to our task, this is already useful, as we want csv and ods files. However, you'll notice when we searched for files contained the string ".csv", we got files of type ".xlsx" as well, just because they had ".csv" somewhere in their name or extension. Up step the hat and dollar…

Regex: The hat , `^`, and dollar, `$`

The hat and dollar are used to specify the start and end of a line respectively. For instance, all file names that start with "Proj" (take note of the capital "P"!)

```
str_subset(files, "^Proj")
## [1] "Project_Trees.csv"   "Project-final2.xlsx"
```

So what if we wanted specifically just ".csv" or ".ods" files, just like in our task? We could use the dollar to search for files ending in a specific extension

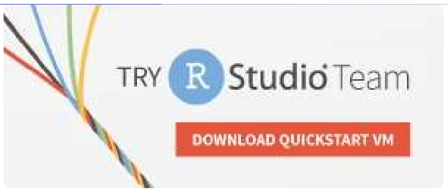
```
str_subset(files, "\\.csv$")
## [1] "tmp-project.csv"        "project.csv"
## [3] "project2-csv-specs.csv" "project-houses.csv"
## [5] "Project_Trees.csv"
str_subset(files, "\\.ods$")
## [1] "project_cars.ods"
```

Now we can search for files that end in certain patterns. That's all well and good, but we still can't search for both together. Up step round parentheses and the pipe…

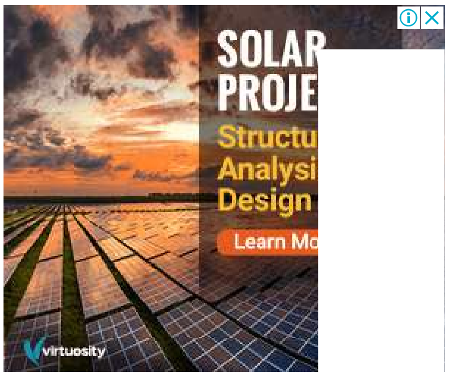
Regex: Round parentheses, `()`, and the pipe, `|`

Round parentheses and the pipe are best used in conjuction with either other. The parentheses specify a group and the pipe means "or". Now, we could search for files ending in a certain extension or another extension. For our task we need ".csv" and ".ods" files. Using the pipe

```
str_subset(files, "\\.csv$|\\.ods$")
## [1] "tmp-project.csv"        "project.csv"
## [3] "project2-csv-specs.csv" "project_cars.ods"
```



Beginner's Guide to
Spatial, Temporal and Spatial-Temporal Ecological Data Analysis with R-INLA
Zuur, Ieno, Saveliev



```
## [3] "project2-csv-specs.csv" "project_cars.ods"
## [5] "project-houses.csv"     "Project_Trees.csv"
```

Now we don't have to write surrounding expressions more than once. Of course there are other csv and ods files that we don't want to collect. Now we need a way of specifiying a block of letters. Up step the square parentheses and the asterisk…

Regex: Square parentheses, `[]`, and the asterisk, `*`

The square parentheses and asterisk. We can match a group of characters or digits using the square parentheses. Here I'm going to use a new function, `str_extract()`. This does as it says on the tin, it ***extracts*** the parts of the text that match our pattern. For instance the last lower case letter in each element of the vector, if such a thing exists

```
str_extract(files, "[a-z]$")
##  [1] "v" "v" "v" "x" "s" "v" "v" NA  "r" "s" "x"
```

Notice that one of the files ends with an upper case letter, so we get an `NA`. To include this we add "A-Z" (to add numbers we add 0-9 and to add metacharacters we write them without escaping them)

```
str_extract(files, "[a-zA-Z]$")
##  [1] "v" "v" "v" "x" "s" "v" "v" "R" "r" "s" "x"
```

Now, this is obviously useless at the moment. This is where does the asterisk comes into it. The asterisk is what is called a quantifier. There are three other quantifiers (`+`, `?` and `{}`), but won't cover them here. A quantifier ***quantifies*** how many of the characters we want to match and the asterisk means we want 0 or more characters of the same form. For instance, we could now extract all of the file extensions if we wished to

```
str_extract(files, "[a-zA-Z]*$")
##  [1] "csv"  "csv"  "csv"  "xlsx" "ods"  "csv"  "csv"  "R"    "r"    "xls"
## [11] "xlsx"
```

So we go backwards from the end of the line collecting all the characters until we hit a character that isn't a lower or upper case letter. We can now use this to grab the group letters preceeding the file extensions for our task

```
str_subset(files, "[a-zA-Z]*\\.(csv|ods)$")
## [1] "tmp-project.csv"        "project.csv"
## [3] "project2-csv-specs.csv" "project_cars.ods"
## [5] "project-houses.csv"     "Project_Trees.csv"
```

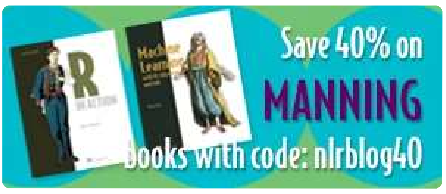
Obviously we still have some pesky files in there that we don't want. Up step the… only joking! We now actually have all the tools to complete the task. The filenames we want take the form project-objects or project_objects, so we know that preceeding that block of letters for "objects" we want either a dash or an underscore. We can use a group and pipe for this

```
str_subset(files, "(\\_|\\-)[a-zA-Z]*\\.(csv|ods)$")
## [1] "tmp-project.csv"        "project2-csv-specs.csv"
## [3] "project_cars.ods"       "project-houses.csv"
## [5] "Project_Trees.csv"
```

We still have two pesky files sneaking in there. How do those two files and the three files we want differ? Well the files we want all start with "project-" or "project_" where as the other two don't. We must also take note that the project could have a capital "P". We can combat that using a group!

```
str_subset(files, "(P|p)roject(\\_|\\-)[a-zA-Z]*\\.(csv|ods)$")
## [1] "project_cars.ods"   "project-houses.csv" "Project_Trees.csv"
```

If we had a huge file list, we'd want to stop files such as "?Project_Trees.csv" filtering in as



Learn R (for ecology)

Our ads respect your privacy. Read our Privacy Policy page to learn more.

Contact us if you wish to help support R-bloggers, and place **your banner here**.

Recent Posts

Deutschsprachiges Online Shiny Training von eoda
How to Calculate a Bootstrap Standard Error in R
Dashboards in R Shiny
Some R Conferences for 2022
Curating Your Data Science Content on RStudio Connect
Adding competing risks in survival data generation
measurement units
Confidence Intervals Explained
The E8 root polytope
extinction minus one
A zsh Helper Script For Updating macOS
RStudio Daily Electron + Quarto CLI Installs
Predictive Analytics Models in R
repoRter.nih: a convenient R interface to the NIH RePORTER Project API
Markov Chain Introduction in R
Dual axis charts – how to make them and why they can be useful

Jobs for R-users

Junior Data Scientist / Quantitative economist
Senior Quantitative Analyst
R programmer
Data Scientist – CGIAR Excellence in Agronomy (Ref No: DDG-R4D/DS/1/CG/EA/06/20)
Data Analytics Auditor, Future of Audit Lead @ London or Newcastle

COMPONENT SEARCH ENGINE
Access PCB Symbols, Footprints, and 3D Models

Regular expressions are definitely a trade worth learning. They play a big role in modern data analytics. For a good table of metacharacters, quantifiers and useful regular expressions, see this microsoft page. Remember, in R you have to double escape metacharacters!

That's all for now. Cheers for reading!

Related

Natural Language Processing in R: Strings and Regular Expressions.
April 9, 2019
In "R bloggers"

Regular expressions in R vs RStudio
The 'regex' family of languages and commands is used for manipulating text strings. More specifically, regular
April 14, 2014
In "R bloggers"

R talk on regular expressions (regex)
Regular expressions are a powerful in any language to manipulate, search, etc. data. For example:> fruit <- c("apple", "banana",
October 6, 2011
In "R bloggers"

Share Tweet

To **leave a comment** for the author, please follow the link and comment on their blog: **R on The Jumping Rivers Blog**.

R-bloggers.com offers **daily e-mail updates** about R news and tutorials about learning R and many other topics. Click here if you're looking to post or find an R/data-science job.

Want to share your content on R-bloggers? click here if you have a blog, or here if you don't.

← **Previous post** **Next post** →

Copyright © 2022 | MH Corporate basic by MH Themes

Object Oriented Programming in Python – What and Why?
Dunn Index for K-Means Clustering Evaluation
Installing Python and Tensorflow with Jupyter Notebook Configurations
How to Get Twitter Data using Python
Visualizations with Altair
Spelling Corrector Program in Python

Full list of contributing R-bloggers

Archives

Select Month

Other sites

SAS blogs
Jobs for R-users

COMPONENT SEARCH ENGINE
Access PCB Symbols, Footprints, and 3D Models