ReadingFilesFromFolder

John Park

July 3, 2019

Examples using rainfall data from Barro Colorado Island

We will explore dealing with loading and using datasets in mulitple occasions. In today's tutorial, we will explore how to read files by files and to make it easier to access each file's information with labels.

First, the data we will be using is a time series of daily rainfall data collected at the clearing ('El Claro'), the original meteorological station maintained by STRI. See https://biogeodb.stri.si.edu/physical_monitoring/research/barrocolorado

List of the functions we will be using today are R basic functions (which means we don't need to load any packages):

```
base::getwd;
```

```
## function ()
## .Internal(getwd())
## <bytecode: 0x7f8235c5d8d0>
## <environment: namespace:base>
```

```
base::paste0;
```

```
## function (..., collapse = NULL)
## .Internal(paste0(list(...), collapse))
## <bytecode: 0x7f823192fc08>
## <environment: namespace:base>
```

```
base::list.files;
```

```
## function (path = ".", pattern = NULL, all.files = FALSE, full.names = FALSE,
##     recursive = FALSE, ignore.case = FALSE, include.dirs = FALSE,
##     no.. = FALSE)
## .Internal(list.files(path, pattern, all.files, full.names, recursive,
##     ignore.case, include.dirs, no..))
## <bytecode: 0x7f8235910ee0>
## <environment: namespace:base>
```

```
base::list;
```

```
## function (...)  .Primitive("list")
```

```
utils::read.csv;
```

```
## function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
##     fill = TRUE, comment.char = "", ...)
## read.table(file = file, header = header, sep = sep, quote = quote,
##     dec = dec, fill = fill, comment.char = comment.char, ...)
## <bytecode: 0x7f82341bc958>
## <environment: namespace:utils>
```

```
base::names;
```

```
## function (x)  .Primitive("names")
```

We start with setting file path for reading the files with precipitation data.

```
DIR_wd=getwd() #get working directory
DIR_path=paste0(DIR_wd,"/data/yearly_rainfall_BCI/") #set path of data location
```

```
ls_filenames<-list.files(path=DIR_path,pattern='.csv$')
```

Function base::list.files() returns vectors of names of files and / or directoties in the designated path. Note that I used an argument `pattern="\.csv$"` in this case, to only retrieve file names ending with ".csv".

Please note that characters `$` and \ are "regular expressions". What are regular expressions? Let's say they are set of grammers to define search patterns, for now.

Brielfy speaking, character `$` is one of the regular expression syntax incidating the end of the string, and character \ is a syntex to declare that we are using the following character as a literal. (\ is also called an escape).

For more details in regular expressions, please refer to other sources such as [link].

Now lets see what ls_filenames object contains:

```
print(ls_filenames)
```

```
##  [1] "BCI_1971.csv" "BCI_1972.csv" "BCI_1973.csv" "BCI_1974.csv"
##  [5] "BCI_1975.csv" "BCI_1976.csv" "BCI_1977.csv" "BCI_1978.csv"
##  [9] "BCI_1979.csv" "BCI_1980.csv" "BCI_1981.csv" "BCI_1982.csv"
## [13] "BCI_1983.csv" "BCI_1984.csv" "BCI_1985.csv" "BCI_1986.csv"
## [17] "BCI_1987.csv" "BCI_1988.csv" "BCI_1989.csv" "BCI_1990.csv"
## [21] "BCI_1991.csv" "BCI_1992.csv" "BCI_1993.csv" "BCI_1994.csv"
## [25] "BCI_1995.csv" "BCI_1996.csv" "BCI_1997.csv" "BCI_1998.csv"
## [29] "BCI_1999.csv" "BCI_2000.csv" "BCI_2001.csv" "BCI_2002.csv"
## [33] "BCI_2003.csv" "BCI_2004.csv" "BCI_2005.csv" "BCI_2006.csv"
## [37] "BCI_2007.csv" "BCI_2008.csv" "BCI_2009.csv" "BCI_2010.csv"
## [41] "BCI_2011.csv" "BCI_2012.csv" "BCI_2013.csv" "BCI_2014.csv"
## [45] "BCI_2015.csv" "BCI_2016.csv" "BCI_2017.csv" "BCI_2018.csv"
## [49] "BCI_2019.csv"
```

Perfect, since we have

readfiles #R #regex