Julia For Data Science | Cheat Sheet
DataFrames.jl Essentials

Created by: Camilo Chacón Sartori | www.camilochacon.com
September 2021
Julia version: v1.6.2 - Package versions: [DataFrames=v1.2.2;CSV=v0.8.5;Query=v1.0.0]

What is DataFrames.jl?

DataFrames.jl is a library used to work with tabular data that for practical ends, its design and functionality is similar to pandas in python.

Below how to install it:

```
julia> using Pkg
julia> Pkg.add("DataFrames")
julia> using DataFrames
```

DataFrames Constructors

To create a two-dimensional labeled data structure with columns of expected different data types.

```
julia> DataFrame(A=4:6, B=["a", "b", "c"],
                 states=[true,false,false], fixed=3.14)
```

	A	B	states	fixed
	Int64	Int64	Bool	Float64
1	4	a	1	3.14
2	5	b	0	3.14
3	6	c	1	3.14

```
julia> dict = Dict("first name" => ["Nuri", "Jaime", "Bog"],
                   "country"    => ["Chile", "Spain", "Romania"],
                   "age"        => [52, 89, 90])

julia> df = DataFrame(dict)
```

	age	country	first name
	Int64	String	String
1	52	Chile	Nuri
2	89	Spain	Jaime
3	90	Romania	Bog

I/O

As a prerequisite we need to have installed CSV.jl:

```
julia> using Pkg
julia> Pkg.add("CSV")
```

Reading data from CSV

To read a file:

```
julia> using CSV
julia> df1 = CSV.read("example.csv", DataFrame)
```

To read multiple files using a vector:

```
julia> dfs = DataFrame.(CSV.File.(["example.csv", "example2.csv"]))
julia> dfs[1]
julia> dfs[2]
```

Save a data frame as CSV

```
julia> CSV.write("new_example.csv", df1)
```

Accessing Basic Information About On a Data Frame

```
julia> size(df) # size.
julia> nrow(df) # the number of rows.
julia> ncol(df) # the number of columns.

julia> describe(df) # basic statistics of the data in your df.

julia> show(df) # print your df.
julia> show(df, allrows=true) # print all rows even if they do not fit on the screen.
julia> show(df, allcols=true) # the same for columns.

julia> first(df, 2) # get the first two rows.
julia> last(df, 2) # get the last two columns.
```

Selection

Getting Columns

```
julia> select(df, "age") # select a single column.
julia> select(df, ["age", "first name"]) # select multiples columns.
```

Getting Rows by Indexing

```
julia> df[2, :] # get a df with the second row and all columns.
julia> df[2, "age"] # get a df with the second row but only the column specified in
the code.
julia> df[1:2, ["age", "country"]] # get a df selecting from first to seconds rows
alongside the two columns.

julia> df[:, ["age"]] # get a df selecting a column.
julia> df[!, "age"] # get a vector selecting a column.
```

Sorting

```
julia> sort(df, order("age", rev=true)) # sort by descenting column.
julia> sort(df, [order("country", by=length),
                 order("age", rev=true)]) # sorting of multiple columns.
```

Filtering

```
julia> filter(row -> row.country == "Chile", df) # filter a column by a specific
value.
julia> filter(row -> row.age > 15 && row."first name" == "José", df) # filter by
multiple column values.
```

Deleting

```
julia> delete!(df, 2) # delete the second row.
julia> delete!(df, [1, 3]) # delete specific rows.

julia> select!(df, Not("age")) # delete the age column.
julia> select!(df, Not(["age", "country"])) # delete specific columns.
```

Query

In many cases ,we need to perfom simple or complex queries on the data frame. To do this, we can use the package: Query.jl.

Below how to install it:

```
julia> using Pkg

julia> Pkg.add("Query")

julia> using Query
```

How to perform a query?

```
julia>

      q1 = @from i in df begin
           @where i.age >= 10 && i.age <= 60
           @select {i.age, i.country}
           @collect DataFrame
      end
```

=>

	age	country
	Int64	String
1	52	Chile

Reshaping

```
julia>  df1 = DataFrame(A=1:5, B=11:15, C=-5:-1)
```

	A	B	C
	Int64	Int64	Int64
1	1	11	-5
2	2	12	-4
3	3	13	-3
4	4	14	-2
5	5	15	-1

```
julia> dfr = stack(df1, [:A, :B], :C) # reshaping of wide to long.
```

	C	variable	value
	Int64	String	Int64
1	-5	A	1
2	-4	A	2
3	-3	A	3
4	-2	A	4
5	-1	A	5
6	-5	B	11
7	-4	B	12
8	-3	B	13
9	-2	B	14
10	-1	B	15

Grouping

```
julia> groupby(dfr, "variable") # return two data frames grouped by a column.
```

Transforming

```
julia> combine(dfr, :, ["C", "value"] => (x, y) -> x + y, renamecols=false)
       # add a transformation into a new column.
```

	C	variable	value	C_value
	Int64	String	Int64	Int64
1	-5	A	1	-4
2	-4	A	2	-2
3	-3	A	3	0
...				