Julia-Data-Query Documentation

Release latest

May 01, 2017

Contents

In data analysis, one of the challenges faced by statisticians/data scientists/researchers is the data cleaning. And for most of us coming from R and Python, we enjoyed the vast resources of the documentations of our favorite packages, such as R's dplyr and Python's Pandas. And most of our problems often have answers already on forums such as Stack Overflow. Unfortunately, this is not the case for Julia. Luckily, we have an active community on places such as Github, Gitter and Julia Discourse all aiming to fill the gaps. Of course we like to invite more users and grow the community even bigger. This is the motivation of the project, by giving newcomers a good place to start. The project aims to provide compilation of examples for data queries in Julia. The benefit of doing this is that we can track the limitations of the current data query packages and further improve it. I encourage readers to contribute their examples for references. We'll feature this at the Contributed section of the two data query packages, namely the DataFramesMeta.jl and the Query.jl.

The documentation proceeds by reproducing the examples in RStudio's dplyr documentation using both DataFrames-Meta.jl and Query.jl.

CHAPTER 1

Contents

DataFramesMeta.jl

I'm going to use R's dplyr as basis for demonstrating the use of DataFramesMeta.jl. We'll try to reproduce the example in this site.

Data: nycflights13

The data is available in the repository and we'll import this using the Request.jl.

That is,

```
using DataFrames
using DataFramesMeta
using Lazy
using Requests

url = "https://raw.githubusercontent.com/alstat/Julia-Data-Query/master/data-raw/
↪flights.csv"

flights = readtable(get_streaming(url));
delete!(flights, :x);
```

We need to delete the column `:x` in the data frame since this is the index and not a column of the original data. So that,

```
size(flights)
# (336776,20)
head(flights)
# 6×20 DataFrames.DataFrame
# | Row | x | year | month | day | dep_time | sched_dep_time | dep_delay |
# ----------------------------------------------
# | 1   | 1 | 2013 | 1     | 1   | 517      | 515            | 2         |
# | 2   | 2 | 2013 | 1     | 1   | 533      | 529            | 4         |
```

```
# | 3   | 3 | 2013 | 1     | 1   | 542      | 540            | 2         |
# | 4   | 4 | 2013 | 1     | 1   | 544      | 545            | -1        |
# | 5   | 5 | 2013 | 1     | 1   | 554      | 600            | -6        |
# | 6   | 6 | 2013 | 1     | 1   | 554      | 558            | -4        |
#
# | Row | arr_time | sched_arr_time | arr_delay | carrier | flight | tailnum  |
# ------------------------------------------------
# | 1   | 830      | 819            | 11        | "UA"    | 1545   | "N14228" |
# | 2   | 850      | 830            | 20        | "UA"    | 1714   | "N24211" |
# | 3   | 923      | 850            | 33        | "AA"    | 1141   | "N619AA" |
# | 4   | 1004     | 1022           | -18       | "B6"    | 725    | "N804JB" |
# | 5   | 812      | 837            | -25       | "DL"    | 461    | "N668DN" |
# | 6   | 740      | 728            | 12        | "UA"    | 1696   | "N39463" |
#
# | Row | origin | dest  | air_time | distance | hour | minute |
# ------------------------------------
# | 1   | "EWR"  | "IAH" | 227      | 1400     | 5    | 15     |
# | 2   | "LGA"  | "IAH" | 227      | 1416     | 5    | 29     |
# | 3   | "JFK"  | "MIA" | 160      | 1089     | 5    | 40     |
# | 4   | "JFK"  | "BQN" | 183      | 1576     | 5    | 45     |
# | 5   | "LGA"  | "ATL" | 116      | 762      | 6    | 0      |
# | 6   | "EWR"  | "ORD" | 150      | 719      | 5    | 58     |
#
# | Row | time_hour             |
# ------------------
# | 1   | "2013-01-01 05:00:00" |
# | 2   | "2013-01-01 05:00:00" |
# | 3   | "2013-01-01 05:00:00" |
# | 4   | "2013-01-01 05:00:00" |
# | 5   | "2013-01-01 06:00:00" |
# | 6   | "2013-01-01 05:00:00" |
```

Single Table Verbs

The following are the equivalen of R dplyr's functions:

DataFrames.jl and **DataFramesMeta.jl**	**R dplyr**
`@where`	`filter()` (and `slice()`)
`@orderby`	`arrange()`
`@select` (and `rename()`, `rename!()`)	`select()` (and `rename()`)
`unique()` and `unique!()`	`distinct()`
`@transform`	`mutate()` (and `transmute()`)
`groupby()`	`group_by()`
`@based_on`	`summarise()`
possible but no function yet	`sample_n()` (and `sample_frac()`)

Filter Rows: @where

Subsetting the data according to some conditions is done using the macro `@where`. The first argument is the DataFrame and the succeeding arguments are the conditions set on the rows of the columns of the DataFrame.

```
@where flights (:month .== 1) (:day .== 1) # or @where(flights, :month .== 1, :day .
↪== 1)
# 842×19 DataFrames.DataFrame
```

```
# | Row | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
# ------------------------------------------------------------------------------
# | 1   | 2013 | 1     | 1   | 517      | 515            | 2         | 830      |
# | 2   | 2013 | 1     | 1   | 533      | 529            | 4         | 850      |
# | 3   | 2013 | 1     | 1   | 542      | 540            | 2         | 923      |
# | 4   | 2013 | 1     | 1   | 544      | 545            | -1        | 1004     |
# | 5   | 2013 | 1     | 1   | 554      | 600            | -6        | 812      |
# | 6   | 2013 | 1     | 1   | 554      | 558            | -4        | 740      |
# | 7   | 2013 | 1     | 1   | 555      | 600            | -5        | 913      |
# | 8   | 2013 | 1     | 1   | 557      | 600            | -3        | 709      |
#
# | 834 | 2013 | 1     | 1   | 2327     | 2250           | 37        | 32       |
# | 835 | 2013 | 1     | 1   | 2343     | 1724           | 379       | 314      |
# | 836 | 2013 | 1     | 1   | 2353     | 2359           | -6        | 425      |
# | 837 | 2013 | 1     | 1   | 2353     | 2359           | -6        | 418      |
# | 838 | 2013 | 1     | 1   | 2356     | 2359           | -3        | 425      |
# | 839 | 2013 | 1     | 1   | NA       | 1630           | NA        | NA       |
# | 840 | 2013 | 1     | 1   | NA       | 1935           | NA        | NA       |
# | 841 | 2013 | 1     | 1   | NA       | 1500           | NA        | NA       |
# | 842 | 2013 | 1     | 1   | NA       | 600            | NA        | NA       |
# more columns ...
```

This is equivalent to the more verbose code in base Julia's DataFrame:

```
flights[(flights[:month] .== 1) & (flights[:day] .== 1), :]
```

You can also use other boolean operators:

```
@where flights ((:month .== 1) | (:month .== 2)) # or @where(flights, (:month .== 1)␣
↪| (:month .== 2))
# 51955×19 DataFrames.DataFrame
# | Row   | year | month | day | dep_time | sched_dep_time | dep_delay |
# ----------------------------------------------------------------------
# | 1     | 2013 | 1     | 1   | 517      | 515            | 2         |
# | 2     | 2013 | 1     | 1   | 533      | 529            | 4         |
# | 3     | 2013 | 1     | 1   | 542      | 540            | 2         |
# | 4     | 2013 | 1     | 1   | 544      | 545            | -1        |
# | 5     | 2013 | 1     | 1   | 554      | 600            | -6        |
# | 6     | 2013 | 1     | 1   | 554      | 558            | -4        |
# | 7     | 2013 | 1     | 1   | 555      | 600            | -5        |
# | 8     | 2013 | 1     | 1   | 557      | 600            | -3        |
#
# | 51947 | 2013 | 2     | 28  | NA       | 1820           | NA        |
# | 51948 | 2013 | 2     | 28  | NA       | 1154           | NA        |
# | 51949 | 2013 | 2     | 28  | NA       | 900            | NA        |
# | 51950 | 2013 | 2     | 28  | NA       | 605            | NA        |
# | 51951 | 2013 | 2     | 28  | NA       | 850            | NA        |
# | 51952 | 2013 | 2     | 28  | NA       | 905            | NA        |
# | 51953 | 2013 | 2     | 28  | NA       | 1115           | NA        |
# | 51954 | 2013 | 2     | 28  | NA       | 830            | NA        |
# | 51955 | 2013 | 2     | 28  | NA       | 840            | NA        |
# more columns ...
```

To select rows by position, simply do the following:

```
@where flights 1:10
# 10×19 DataFrames.DataFrame
# | Row | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
```

```
# ------------------------------------------------
# | 1    | 2013 | 1     | 1   | 517      | 515           | 2         | 830      |
# | 2    | 2013 | 1     | 1   | 533      | 529           | 4         | 850      |
# | 3    | 2013 | 1     | 1   | 542      | 540           | 2         | 923      |
# | 4    | 2013 | 1     | 1   | 544      | 545           | -1        | 1004     |
# | 5    | 2013 | 1     | 1   | 554      | 600           | -6        | 812      |
# | 6    | 2013 | 1     | 1   | 554      | 558           | -4        | 740      |
# | 7    | 2013 | 1     | 1   | 555      | 600           | -5        | 913      |
# | 8    | 2013 | 1     | 1   | 557      | 600           | -3        | 709      |
# | 9    | 2013 | 1     | 1   | 557      | 600           | -3        | 838      |
# | 10   | 2013 | 1     | 1   | 558      | 600           | -2        | 753      |
# more columns ...
```

or select using boolean index, simply run

```
@where flights find([true, true])
```

Arrange Rows: @orderby

```
@orderby flights :year :month :day
# 336776×19 DataFrames.DataFrame
# | Row    | year | month | day | dep_time | sched_dep_time | dep_delay |
# ----------------------------------------------
# | 1      | 2013 | 1     | 1   | 517      | 515           | 2         |
# | 2      | 2013 | 1     | 1   | 533      | 529           | 4         |
# | 3      | 2013 | 1     | 1   | 542      | 540           | 2         |
# | 4      | 2013 | 1     | 1   | 544      | 545           | -1        |
# | 5      | 2013 | 1     | 1   | 554      | 600           | -6        |
# | 6      | 2013 | 1     | 1   | 554      | 558           | -4        |
# | 7      | 2013 | 1     | 1   | 555      | 600           | -5        |
# | 8      | 2013 | 1     | 1   | 557      | 600           | -3        |
#
# | 336768 | 2013 | 12    | 31  | NA       | 2000          | NA        |
# | 336769 | 2013 | 12    | 31  | NA       | 1500          | NA        |
# | 336770 | 2013 | 12    | 31  | NA       | 1430          | NA        |
# | 336771 | 2013 | 12    | 31  | NA       | 855           | NA        |
# | 336772 | 2013 | 12    | 31  | NA       | 705           | NA        |
# | 336773 | 2013 | 12    | 31  | NA       | 825           | NA        |
# | 336774 | 2013 | 12    | 31  | NA       | 1615          | NA        |
# | 336775 | 2013 | 12    | 31  | NA       | 600           | NA        |
# | 336776 | 2013 | 12    | 31  | NA       | 830           | NA        |
# more columns ...
```

There is a problem with descending order. The code below is my attempt and the result is not equivalent with that in dplyr.

```
@orderby flights sort(:arr_delay, rev = false)
```

Select Columns: @select

To select a specific column use `@select` macro.

```
 @select flights :year :month :day # or @select(flights, :year, :month, :day)
 # 336776×3 DataFrames.DataFrame
```

```
# | Row    | year | month | day |
# -----------------
# | 1      | 2013 | 1     | 1   |
# | 2      | 2013 | 1     | 1   |
# | 3      | 2013 | 1     | 1   |
# | 4      | 2013 | 1     | 1   |
# | 5      | 2013 | 1     | 1   |
# | 6      | 2013 | 1     | 1   |
# | 7      | 2013 | 1     | 1   |
# | 8      | 2013 | 1     | 1   |
#
# | 336768 | 2013 | 9     | 30  |
# | 336769 | 2013 | 9     | 30  |
# | 336770 | 2013 | 9     | 30  |
# | 336771 | 2013 | 9     | 30  |
# | 336772 | 2013 | 9     | 30  |
# | 336773 | 2013 | 9     | 30  |
# | 336774 | 2013 | 9     | 30  |
# | 336775 | 2013 | 9     | 30  |
# | 336776 | 2013 | 9     | 30  |

Select all columns except those from year to day (inclusive)
```

```
cols = setdiff(names(flights), [:year, :month, :day]);
@select flights cols
# 336776×16 DataFrames.DataFrame
# | Row    | dep_time | sched_dep_time | dep_delay | arr_time | sched_arr_time |
# ------------------------------------------------
# | 1      | 517      | 515            | 2         | 830      | 819            |
# | 2      | 533      | 529            | 4         | 850      | 830            |
# | 3      | 542      | 540            | 2         | 923      | 850            |
# | 4      | 544      | 545            | -1        | 1004     | 1022           |
# | 5      | 554      | 600            | -6        | 812      | 837            |
# | 6      | 554      | 558            | -4        | 740      | 728            |
# | 7      | 555      | 600            | -5        | 913      | 854            |
# | 8      | 557      | 600            | -3        | 709      | 723            |
#
# | 336768 | 2241     | 2246           | -5        | 2345     | 1              |
# | 336769 | 2307     | 2255           | 12        | 2359     | 2358           |
# | 336770 | 2349     | 2359           | -10       | 325      | 350            |
# | 336771 | NA       | 1842           | NA        | NA       | 2019           |
# | 336772 | NA       | 1455           | NA        | NA       | 1634           |
# | 336773 | NA       | 2200           | NA        | NA       | 2312           |
# | 336774 | NA       | 1210           | NA        | NA       | 1330           |
# | 336775 | NA       | 1159           | NA        | NA       | 1344           |
# | 336776 | NA       | 840            | NA        | NA       | 1020           |
# more columns ...
```

You can rename variables with `@select` by using named arguments:

```
@select(flights, tail_num = :tailnum)
# 336776×1 DataFrames.DataFrame
# | Row    | tail_num |
# ------------
# | 1      | "N14228" |
# | 2      | "N24211" |
# | 3      | "N619AA" |
# | 4      | "N804JB" |
```

```
# | 5       | "N668DN" |
# | 6       | "N39463" |
# | 7       | "N516JB" |
# | 8       | "N829AS" |
#
# | 336768 | "N346JB" |
# | 336769 | "N565JB" |
# | 336770 | "N516JB" |
# | 336771 | "N740EV" |
# | 336772 | NA       |
# | 336773 | NA       |
# | 336774 | "N535MQ" |
# | 336775 | "N511MQ" |
# | 336776 | "N839MQ" |
```

But because `@select` drops all the variables not explicitly mentioned, it's not that useful. Instead, use `@rename`:

```
rename(flights, :tailnum, :tail_num)
# 336776×19 DataFrames.DataFrame
# | Row     | year | month | day | dep_time | sched_dep_time | dep_delay |
# ------------------------------------------
# | 1       | 2013 | 1     | 1   | 517      | 515            | 2         |
# | 2       | 2013 | 1     | 1   | 533      | 529            | 4         |
# | 3       | 2013 | 1     | 1   | 542      | 540            | 2         |
# | 4       | 2013 | 1     | 1   | 544      | 545            | -1        |
# | 5       | 2013 | 1     | 1   | 554      | 600            | -6        |
# | 6       | 2013 | 1     | 1   | 554      | 558            | -4        |
# | 7       | 2013 | 1     | 1   | 555      | 600            | -5        |
# | 8       | 2013 | 1     | 1   | 557      | 600            | -3        |
#
# | 336768 | 2013 | 9     | 30  | 2241     | 2246           | -5        |
# | 336769 | 2013 | 9     | 30  | 2307     | 2255           | 12        |
# | 336770 | 2013 | 9     | 30  | 2349     | 2359           | -10       |
# | 336771 | 2013 | 9     | 30  | NA       | 1842           | NA        |
# | 336772 | 2013 | 9     | 30  | NA       | 1455           | NA        |
# | 336773 | 2013 | 9     | 30  | NA       | 2200           | NA        |
# | 336774 | 2013 | 9     | 30  | NA       | 1210           | NA        |
# | 336775 | 2013 | 9     | 30  | NA       | 1159           | NA        |
# | 336776 | 2013 | 9     | 30  | NA       | 840            | NA        |
# more columns ...
```

Extract Distinct Rows: unique()

```
unique(flights, :tailnum)
# 4044×19 DataFrames.DataFrame
# | Row  | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
# ---------------------------------------------
# | 1    | 2013 | 1     | 1   | 517      | 515            | 2         | 830      |
# | 2    | 2013 | 1     | 1   | 533      | 529            | 4         | 850      |
# | 3    | 2013 | 1     | 1   | 542      | 540            | 2         | 923      |
# | 4    | 2013 | 1     | 1   | 544      | 545            | -1        | 1004     |
# | 5    | 2013 | 1     | 1   | 554      | 600            | -6        | 812      |
# | 6    | 2013 | 1     | 1   | 554      | 558            | -4        | 740      |
# | 7    | 2013 | 1     | 1   | 555      | 600            | -5        | 913      |
# | 8    | 2013 | 1     | 1   | 557      | 600            | -3        | 709      |
#
```

```
# | 4036 | 2013 | 9     | 17  | 1755     | 1805           | -10       | 1937     |
# | 4037 | 2013 | 9     | 18  | 1237     | 1240           | -3        | 1515     |
# | 4038 | 2013 | 9     | 18  | 1754     | 1805           | -11       | 1927     |
# | 4039 | 2013 | 9     | 19  | 1759     | 1805           | -6        | 1946     |
# | 4040 | 2013 | 9     | 20  | 1758     | 1805           | -7        | 1929     |
# | 4041 | 2013 | 9     | 22  | 1759     | 1805           | -6        | 1945     |
# | 4042 | 2013 | 9     | 23  | 1759     | 1805           | -6        | 1935     |
# | 4043 | 2013 | 9     | 24  | 1751     | 1805           | -14       | 1937     |
# | 4044 | 2013 | 9     | 28  | 712      | 720            | -8        | 955      |
# more columns ...

unique(flights, [:origin, :dest])
# 224×19 DataFrames.DataFrame
# | Row | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
# ------------------------------------------------
# | 1   | 2013 | 1     | 1   | 517      | 515            | 2         | 830      |
# | 2   | 2013 | 1     | 1   | 533      | 529            | 4         | 850      |
# | 3   | 2013 | 1     | 1   | 542      | 540            | 2         | 923      |
# | 4   | 2013 | 1     | 1   | 544      | 545            | -1        | 1004     |
# | 5   | 2013 | 1     | 1   | 554      | 600            | -6        | 812      |
# | 6   | 2013 | 1     | 1   | 554      | 558            | -4        | 740      |
# | 7   | 2013 | 1     | 1   | 555      | 600            | -5        | 913      |
# | 8   | 2013 | 1     | 1   | 557      | 600            | -3        | 709      |
#
# | 216 | 2013 | 3     | 1   | 2046     | 2045           | 1         | 2206     |
# | 217 | 2013 | 3     | 2   | 1606     | 1610           | -4        | 1723     |
# | 218 | 2013 | 4     | 14  | 937      | 945            | -8        | 1130     |
# | 219 | 2013 | 5     | 25  | 1007     | 1000           | 7         | 1129     |
# | 220 | 2013 | 6     | 14  | 1756     | 1800           | -4        | 2004     |
# | 221 | 2013 | 6     | 15  | 1517     | 1520           | -3        | 1656     |
# | 222 | 2013 | 7     | 5   | 1355     | 1400           | -5        | 1550     |
# | 223 | 2013 | 7     | 6   | 1629     | 1615           | 14        | 1954     |
# | 224 | 2013 | 7     | 27  | NA       | 106            | NA        | NA       |
# more columns ...
```

Add New Column: @transform

Besides selecting sets of existing columns, it's often useful to add new columns that are functions of existing columns. This is the job of @transform:

```
@transform(flights,
  gain = :arr_delay .- :dep_delay,
  speed = :distance ./ :air_time * 60)
# 336776×21 DataFrames.DataFrame
# | Row    | year | month | day | dep_time | sched_dep_time | dep_delay |
# ----------------------------------------
# | 1      | 2013 | 1     | 1   | 517      | 515            | 2         |
# | 2      | 2013 | 1     | 1   | 533      | 529            | 4         |
# | 3      | 2013 | 1     | 1   | 542      | 540            | 2         |
# | 4      | 2013 | 1     | 1   | 544      | 545            | -1        |
# | 5      | 2013 | 1     | 1   | 554      | 600            | -6        |
# | 6      | 2013 | 1     | 1   | 554      | 558            | -4        |
# | 7      | 2013 | 1     | 1   | 555      | 600            | -5        |
# | 8      | 2013 | 1     | 1   | 557      | 600            | -3        |
#
# | 336768 | 2013 | 9     | 30  | 2241     | 2246           | -5        |
```

```
# | 336769 | 2013 | 9     | 30  | 2307     | 2255          | 12        |
# | 336770 | 2013 | 9     | 30  | 2349     | 2359          | -10       |
# | 336771 | 2013 | 9     | 30  | NA       | 1842          | NA        |
# | 336772 | 2013 | 9     | 30  | NA       | 1455          | NA        |
# | 336773 | 2013 | 9     | 30  | NA       | 2200          | NA        |
# | 336774 | 2013 | 9     | 30  | NA       | 1210          | NA        |
# | 336775 | 2013 | 9     | 30  | NA       | 1159          | NA        |
# | 336776 | 2013 | 9     | 30  | NA       | 840           | NA        |
# more columns ...
```

There is no option to refer to columns that was created, and so my solution is to use another `@transform` function.

```
old_flights = @transform(flights,
  gain = :arr_delay .- :dep_delay,
  speed = :distance ./ :air_time * 60)

new_flights = @transform(old_flights,
  gain_per_hour = :gain ./ (:air_time / 60))
# 336776×22 DataFrames.DataFrame
# | Row    | year | month | day | dep_time | sched_dep_time | dep_delay |
# ----------------------------------------
# | 1      | 2013 | 1     | 1   | 517      | 515            | 2         |
# | 2      | 2013 | 1     | 1   | 533      | 529            | 4         |
# | 3      | 2013 | 1     | 1   | 542      | 540            | 2         |
# | 4      | 2013 | 1     | 1   | 544      | 545            | -1        |
# | 5      | 2013 | 1     | 1   | 554      | 600            | -6        |
# | 6      | 2013 | 1     | 1   | 554      | 558            | -4        |
# | 7      | 2013 | 1     | 1   | 555      | 600            | -5        |
# | 8      | 2013 | 1     | 1   | 557      | 600            | -3        |
#
# | 336768 | 2013 | 9     | 30  | 2241     | 2246           | -5        |
# | 336769 | 2013 | 9     | 30  | 2307     | 2255           | 12        |
# | 336770 | 2013 | 9     | 30  | 2349     | 2359           | -10       |
# | 336771 | 2013 | 9     | 30  | NA       | 1842           | NA        |
# | 336772 | 2013 | 9     | 30  | NA       | 1455           | NA        |
# | 336773 | 2013 | 9     | 30  | NA       | 2200           | NA        |
# | 336774 | 2013 | 9     | 30  | NA       | 1210           | NA        |
# | 336775 | 2013 | 9     | 30  | NA       | 1159           | NA        |
# | 336776 | 2013 | 9     | 30  | NA       | 840            | NA        |
# more columns ...
```

Summarise Values: @with

```
@with(flights, mean(dropna(:dep_delay)))
# 12.639070257304708
```

Randomly Sample Rows

```
@where flights sample(1:nrow(flights), 10, replace = false)
# 10×19 DataFrames.DataFrame
# | Row | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
# ------------------------------------------------
# | 1   | 2013 | 7     | 1   | 812      | 806            | 6         | 1049     |
# | 2   | 2013 | 11    | 10  | 1356     | 1400           | -4        | 1544     |
```

```
# | 3    | 2013 | 5     | 13  | 905      | 900           | 5         | 1016     |
# | 4    | 2013 | 11    | 27  | 816      | 805           | 11        | 1006     |
# | 5    | 2013 | 5     | 23  | 1749     | 1607          | 102       | NA       |
# | 6    | 2013 | 11    | 5   | 1254     | 1300          | -6        | 1513     |
# | 7    | 2013 | 3     | 15  | 601      | 600           | 1         | 723      |
# | 8    | 2013 | 6     | 9   | 813      | 820           | -7        | 939      |
# | 9    | 2013 | 6     | 19  | 952      | 909           | 43        | 1205     |
# | 10   | 2013 | 5     | 30  | 1554     | 1559          | -5        | 1745     |
# more columns ...

@where flights sample(1:nrow(flights), convert(Int64, ceil(nrow(flights) * .01)),␣
↪replace = false)
# 3368×19 DataFrames.DataFrame
# | Row  | year | month | day | dep_time | sched_dep_time | dep_delay | arr_time |
# ------------------------------------------------
# | 1    | 2013 | 11    | 29  | 1224     | 1230           | -6        | 1401     |
# | 2    | 2013 | 4     | 7   | 1753     | 1755           | -2        | 2116     |
# | 3    | 2013 | 5     | 29  | 1351     | 1345           | 6         | 1653     |
# | 4    | 2013 | 2     | 26  | 942      | 955            | -13       | 1240     |
# | 5    | 2013 | 10    | 21  | 811      | 755            | 16        | 929      |
# | 6    | 2013 | 10    | 5   | 1416     | 1415           | 1         | 1538     |
# | 7    | 2013 | 5     | 12  | 926      | 930            | -4        | 1216     |
# | 8    | 2013 | 5     | 25  | 1244     | 1250           | -6        | 1419     |
#
# | 3360 | 2013 | 4     | 23  | 1743     | 1555           | 108       | 1946     |
# | 3361 | 2013 | 10    | 9   | 658      | 700            | -2        | 1044     |
# | 3362 | 2013 | 4     | 28  | 1455     | 1038           | 257       | 1606     |
# | 3363 | 2013 | 11    | 25  | 706      | 700            | 6         | 829      |
# | 3364 | 2013 | 7     | 7   | 1626     | 1555           | 31        | 1858     |
# | 3365 | 2013 | 6     | 24  | 839      | 836            | 3         | 942      |
# | 3366 | 2013 | 7     | 11  | 1441     | 1440           | 1         | 1633     |
# | 3367 | 2013 | 10    | 23  | 842      | 848            | -6        | 1116     |
# | 3368 | 2013 | 4     | 30  | 1048     | 1030           | 18        | 1351     |
#  more columns ...
```

Grouped Operations

```
by_tailnum = groupby(flights, :tailnum)
delay = @based_on(by_tailnum,
  count = length(:tailnum),
  dist = mean(dropna(:distance)),
  delay = mean(dropna(:arr_delay)))
delay = @where(delay, :count .> 20, :dist .< 2000)
# 2962×4 DataFrames.DataFrame
# | Row  | tailnum  | count | dist     | delay     |
# ----------------------------
# | 1    | NA       | 2512  | 710.258 | NaN       |
# | 2    | "N0EGMQ" | 371   | 676.189 | 9.98295   |
# | 3    | "N10156" | 153   | 757.948 | 12.7172   |
# | 4    | "N102UW" | 48    | 535.875 | 2.9375    |
# | 5    | "N103US" | 46    | 535.196 | -6.93478  |
# | 6    | "N104UW" | 47    | 535.255 | 1.80435   |
# | 7    | "N10575" | 289   | 519.702 | 20.6914   |
# | 8    | "N105UW" | 45    | 524.844 | -0.266667 |
#
# | 2954 | "N995DL" | 57    | 883.579 | 1.92982   |
```

```
# | 2955 | "N996AT" | 29    | 673.897 | 6.53846   |
# | 2956 | "N996DL" | 102   | 897.304 | 0.524752  |
# | 2957 | "N997AT" | 44    | 679.045 | 16.3023   |
# | 2958 | "N997DL" | 63    | 867.762 | 4.90323   |
# | 2959 | "N998AT" | 26    | 593.538 | 29.96     |
# | 2960 | "N998DL" | 77    | 857.818 | 16.3947   |
# | 2961 | "N999DN" | 61    | 895.459 | 14.3115   |
# | 2962 | "N9EAMQ" | 248   | 674.665 | 9.23529   |
```

We could use these to find the number of planes and the number of flights that go to each possible destination:

```
destinations = groupby(flights, :dest)
@based_on(destinations,
  planes = length(unique(:tailnum)),
  flights = length(:dest)
)
# 105×3 DataFrames.DataFrame
# | Row | dest  | planes | flights |
# -------------------
# | 1   | "ABQ" | 108    | 254     |
# | 2   | "ACK" | 58     | 265     |
# | 3   | "ALB" | 172    | 439     |
# | 4   | "ANC" | 6      | 8       |
# | 5   | "ATL" | 1180   | 17215   |
# | 6   | "AUS" | 993    | 2439    |
# | 7   | "AVL" | 159    | 275     |
# | 8   | "BDL" | 186    | 443     |
#
# | 97  | "SRQ" | 373    | 1211    |
# | 98  | "STL" | 960    | 4339    |
# | 99  | "STT" | 87     | 522     |
# | 100 | "SYR" | 383    | 1761    |
# | 101 | "TPA" | 1126   | 7466    |
# | 102 | "TUL" | 105    | 315     |
# | 103 | "TVC" | 60     | 101     |
# | 104 | "TYS" | 273    | 631     |
# | 105 | "XNA" | 176    | 1036    |
```

Example of progressively rolling-up a dataset:

```
daily = groupby(flights, [:year, :month, :day])
per_day = @based_on(daily, flights = length(:day))
# 365×4 DataFrames.DataFrame
# | Row | year | month | day | flights |
# ---------------------
# | 1   | 2013 | 1     | 1   | 842     |
# | 2   | 2013 | 1     | 2   | 943     |
# | 3   | 2013 | 1     | 3   | 914     |
# | 4   | 2013 | 1     | 4   | 915     |
# | 5   | 2013 | 1     | 5   | 720     |
# | 6   | 2013 | 1     | 6   | 832     |
# | 7   | 2013 | 1     | 7   | 933     |
# | 8   | 2013 | 1     | 8   | 899     |
#
# | 357 | 2013 | 12    | 23  | 985     |
# | 358 | 2013 | 12    | 24  | 761     |
# | 359 | 2013 | 12    | 25  | 719     |
# | 360 | 2013 | 12    | 26  | 936     |
```

```
# | 361 | 2013 | 12    | 27  | 963     |
# | 362 | 2013 | 12    | 28  | 814     |
# | 363 | 2013 | 12    | 29  | 888     |
# | 364 | 2013 | 12    | 30  | 968     |
# | 365 | 2013 | 12    | 31  | 776     |

per_month = @based_on(groupby(per_day, [:year, :month]), flights = sum(:flights))
# 12×3 DataFrames.DataFrame
# | Row | year | month | flights |
# ------------------
# | 1   | 2013 | 1     | 27004   |
# | 2   | 2013 | 2     | 24951   |
# | 3   | 2013 | 3     | 28834   |
# | 4   | 2013 | 4     | 28330   |
# | 5   | 2013 | 5     | 28796   |
# | 6   | 2013 | 6     | 28243   |
# | 7   | 2013 | 7     | 29425   |
# | 8   | 2013 | 8     | 29327   |
# | 9   | 2013 | 9     | 27574   |
# | 10  | 2013 | 10    | 28889   |
# | 11  | 2013 | 11    | 27268   |
# | 12  | 2013 | 12    | 28135   |

per_year = @based_on(groupby(per_day, [:year]), flights = sum(:flights))
# 1×2 DataFrames.DataFrame
# | Row | year | flights |
# -------------
# | 1   | 2013 | 336776  |
```

Chaining Operations

What we've done above so far is not based on chaining, in R the popular operator for doing this is the magrittr's pipe operator `%>%`. In Julia, this is similar to `|>`. Consider the following example from dplyr's website.

```
a1 = @select flights :year :month :day :arr_delay :dep_delay
a2 = groupby(a1, [:year, :month, :day])
a3 = @based_on(a2,
  arr = mean(dropna(:arr_delay)),
  dep = mean(dropna(:dep_delay)))
a4 = @where(a3, (:arr .> 30) | (:dep .> 30))
```

We can do chaining using `|>` with the help of `@linq` macro. So that,

```
@linq flights |>
  @select(:year, :month, :day, :arr_delay, :dep_delay) |>
  groupby([:year, :month, :day]) |>
  based_on(
    arr = mean(dropna(:arr_delay)),
    dep = mean(dropna(:dep_delay))) |>
  @where((:arr .> 30) | (:dep .> 30))
```

Another alternative, which for me personally is a lot cleaner is to use the pipe macro `@>` from the Lazy.jl package. The above codes is equivalent to

```
using Lazy: @>
```

```
@> begin
  flights
  @select :year :month :day :arr_delay :dep_delay
  groupby([:year, :month, :day])
  @based_on(
    arr = mean(dropna(:arr_delay)),
    dep = mean(dropna(:dep_delay))
  )
  @where ((:arr .> 30) | (:dep .> 30))
end
```

The three approaches above returns the same result give below:

```
# 49×5 DataFrames.DataFrame
# | Row | year | month | day | arr      | dep      |
# ----------------------------
# | 1   | 2013 | 1     | 16  | 34.2474 | 24.6129 |
# | 2   | 2013 | 1     | 31  | 32.6029 | 28.6584 |
# | 3   | 2013 | 2     | 11  | 36.2901 | 39.0736 |
# | 4   | 2013 | 2     | 27  | 31.2525 | 37.7633 |
# | 5   | 2013 | 3     | 8   | 85.8622 | 83.5369 |
# | 6   | 2013 | 3     | 18  | 41.2919 | 30.118  |
# | 7   | 2013 | 4     | 10  | 38.4123 | 33.0237 |
# | 8   | 2013 | 4     | 12  | 36.0481 | 34.8384 |
#
# | 41  | 2013 | 10    | 7   | 39.0173 | 39.1467 |
# | 42  | 2013 | 10    | 11  | 18.923  | 31.2318 |
# | 43  | 2013 | 12    | 5   | 51.6663 | 52.328  |
# | 44  | 2013 | 12    | 8   | 36.9118 | 21.5153 |
# | 45  | 2013 | 12    | 9   | 42.5756 | 34.8002 |
# | 46  | 2013 | 12    | 10  | 44.5088 | 26.4655 |
# | 47  | 2013 | 12    | 14  | 46.3975 | 28.3616 |
# | 48  | 2013 | 12    | 17  | 55.8719 | 40.7056 |
# | 49  | 2013 | 12    | 23  | 32.226  | 32.2541 |
```

Contributed Examples

The following contains examples contributed by the community

This section contains examples using DataFramesMeta.jl package.

Transform

1. This example is available in this link. We want to reproduce the following R codes:

```
library(dplyr)

women_new <- rbind(women, c(NA, 1), c(NA, NA))
women_new %>%
  filter(height %>% complete.cases) %>%
  mutate(sector = character(n()),
         sector = replace(sector, height >= 0 & height <= 60, "1"),
         sector = replace(sector, height >= 61 & height <= 67, "2"),
         sector = replace(sector, height >= 68 & height <= 72, "3"))
```

Solution using DataFramesMeta.jl:

```
using DataFrames
using DataFramesMeta
using Lazy: @>
using RDatasets

women = dataset("datasets", "women");
women_new = vcat(
              women,
              DataFrame(Height = [NA; NA], Weight = @data [1; NA])
            )

@> begin
    women_new
    @where !isna(:Height)
    @transform(
        Class = @> begin
            function (x)
                 0 <= x <= 60 ?  1 :
                61 <= x <= 67 ?  2 :
                68 <= x <= 72 ?  3 :
                NA
            end
            map(:Height)
        end
    )
end
```

Without removing the `NA`:

```
@> begin
    women_new
    @transform(
        Class = @> begin
            function (x)
                isna(x)       ? NA :
                 0 <= x <= 60 ?  1 :
                61 <= x <= 67 ?  2 :
                68 <= x <= 72 ?  3 :
                NA
            end
            map(:Height)
        end
    )
end
```

Solution using Query.jl:

```
using DataFrames
using Query
using RDatasets

women = dataset("datasets", "women");
women_new = vcat(
              women,
              DataFrame(Height = [NA; NA], Weight = @data [1; NA])
            )
```

```
@from i in women_new begin
    @where !isnull(i.Height)
    @select {
        i.Height, i.Weight,
        class = 0 <= i.Height <= 60 ?  1 :
                61 <= i.Height <= 67 ?  2 :
                68 <= i.Height <= 72 ?  3 :
                 0
    }
    @collect DataFrame
end
```

Without removing the `NA`:

```
@from i in women_new begin
    @select {
        i.Height, i.Weight,
        class = 0 <= i.Height <= 60 ?  1 :
                61 <= i.Height <= 67 ?  2 :
                68 <= i.Height <= 72 ?  3 :
                 0
    }
    @collect DataFrame
end
```

Filter

Summarize

Join

CHAPTER 2

Contributors

Prepared and maintain by Al-Ahmadgaid B. Asaad (twitter, blog) and Other Contributors.