Hello Julia

An Interactive Look at My First Experience with the MIT Incubated Open Source Programming Language

Richard Bryant
PHUSE Hoboken SDE
18 JULY 2019

Health Analytics Collective

Real world data. Real world insights. ™

Health Analytics Collective

- A pioneering group that strategically uses RWE in drug development
- Combined teams bring proven regulatory expertise, strong processes, and the latest technology to provide data-driven insights

GOAL

Shortened development timelines to new treatments for unmet needs.

What is Julia?

- High performance open-sourced computer language for data science, machine learning and scientific computing
- Origins go back to 2009 and released publicly in 2012
- Julia Computing founded in 2015, by the following:
  - Julia language creators Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah
  - Keno Fischer and Deepak Vinchhi

What's so special about Julia?

Pros

- Fast, dynamic (multiple dispatch), compiled – not interpreted, enables asynchronous I/O, package management and more…
- Easily adaptable for dynamic Real World Evidence
- Easy to learn
- Growing user community (with excellent support)
  - (New!) PHUSE – Julia Initiative for Standard Analysis Scripts
    - Chris Hurley and Hanming Tu

Cons

- Need more Julia based visualization packages
- Relatively new user community (but growing)

Stats Programmer's Perspective

This presentation is for you

- Julia programming
  - Code examples for familiar tasks
  - Reading SAS datasets
  - Developing macros and functions
- Processing large data quickly
- Data visualizations

Julia computing

HelloJuliaJupyterDemo.mp4

Related code and tools

- Packages
- Jupyter notebooks ("Julia"-"Python"-"R")
- GitHub
- StackOverflow, Google and others

Global Macros and Functions

Power to share between users repeatable functions, making code more powerful, and faster

Global Macros and Functions ← What's the difference?

In [1]:
```
# Macro to determine how many records are coming from each file
# UserMacro prtnumrecs
macro prtnumrecs(dsname)
    sd = :(size($dsname,1))
    nr = eval(sd)
    println("Number of records in $dsname", " is ", nr)
end
```

Out[1]:
```
@prtnumrecs (macro with 1 method)
```

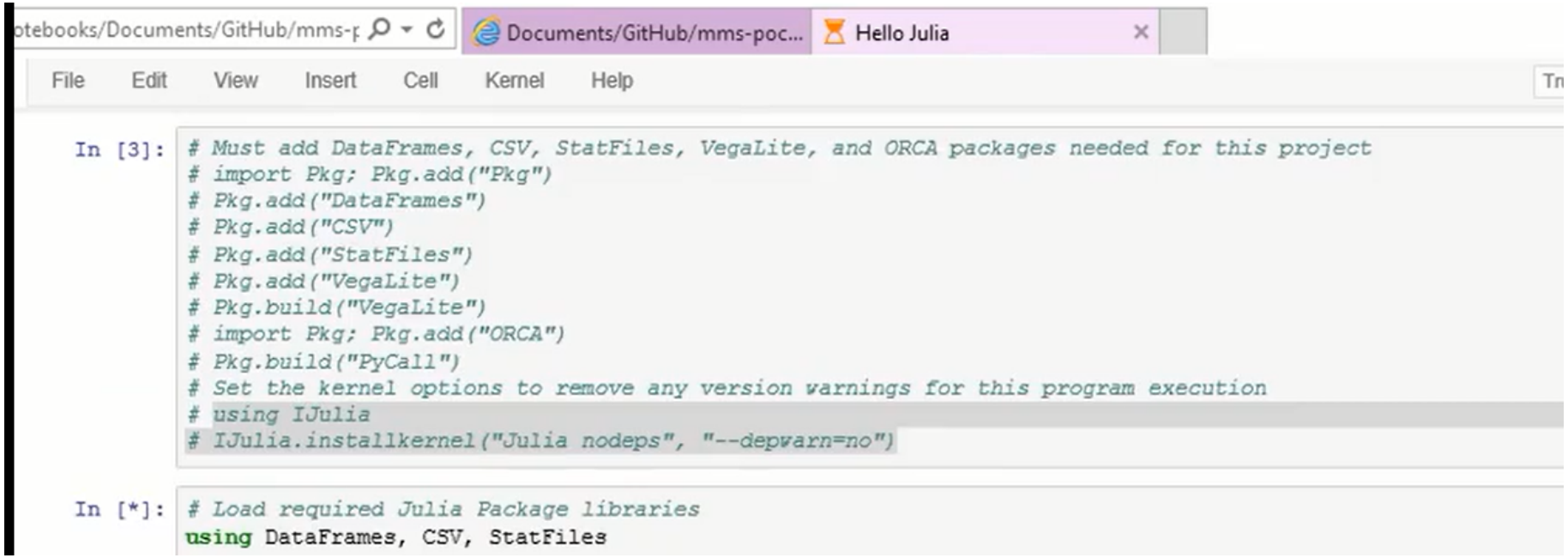
In [2]:
```
# Function to show a user specified number of lines from a user specified dataframe
# UserFunction displayrecs
function displayrecs(dfname, numlines)
    withenv("LINES" => numlines) do
        display(dfname)
    end
end
```

Out[2]:
```
displayrecs (generic function with 1 method)
```

Extract Transform Load (ETL)

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets



ETL

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets

```
# Read tables on Datacise using the updated CSV.jl package v0.5.3
demo17q2 = CSV.read("C:\\Users\\churley\\data\\DEMO17Q2.txt"; header = 1, delim = '$')
drug17q2 = CSV.read("C:\\Users\\churley\\data\\DRUG17Q2.txt"; header = 1, delim = '$')
indi17q2 = CSV.read("C:\\Users\\churley\\data\\INDI17Q2.txt"; header = 1, delim = '$')
outc17q2 = CSV.read("C:\\Users\\churley\\data\\OUTC17Q2.txt"; header = 1, delim = '$')
reac17q2 = CSV.read("C:\\Users\\churley\\data\\REAC17Q2.txt"; header = 1, delim = '$')
rpsr17q2 = CSV.read("C:\\Users\\churley\\data\\RPSR17Q2.txt"; header = 1, delim = '$')
ther17q2 = CSV.read("C:\\Users\\churley\\data\\THER17Q2.txt"; header = 1, delim = '$')
```

Filter Data

```
In [8]: drug17q2.exp_dt[1127910:1127912]  #The values look fine

Out[8]: 3-element Array{Union{Missing, Int64},1}:
                missing
         20190303
                missing
```

ETL

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets

```
In [*]: # Read SAS dictionary data using a function from the StatFiles package and load into a dataframe
        using ReadStat, DataFrames, StatFiles

        # meddra20 = load from Datacise
        m20 = read_sas7bdat("C:\\Users\\churley\\data\\meddra20_0.sas7bdat")

        # meddra20 = load from desktop
        # m20 = read_sas7bdat("C:\\Users\\hurle\\Anaconda3\\Juliaprog\\Q2Data\\meddra20_0.sas7bdat")
        meddra20 = DataFrame(m20.data, m20.headers)
```

ETL

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets

```
In [10]: # Read SAS dictionary data using a function from the StatFiles package and load into a dataframe
         using ReadStat, DataFrames, StatFiles

         # meddra20 = load from Datacise
         m20 = read_sas7bdat("C:\\Users\\churley\\data\\meddra20_0.sas7bdat")

         # meddra20 = load from desktop
         # m20 = read_sas7bdat("C:\\Users\\hurle\\Anaconda3\\Juliaprog\\Q2Data\\meddra20_0.sas7bdat")
         meddra20 = DataFrame(m20.data, m20.headers)
```

_code	hlgt_code	soc_code	pt_name	hlt_name	hlgt_name	soc_name	soc_abbrev	pt_soc_code	primary_soc_fg	ll
/alu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu...	DataValu..
277e7	1.00017e7	1.00214e7	"Alveolitis allergic"	"Allergic conditions NEC"	"Allergic conditions"	"Immune system disorders"	"Immun"	1.00387e7	"N"	"\"VENTILATION\ PNEUMONITIS
025e7	1.0025e7	1.00387e7	"Alveolitis allergic"	"Lower respiratory tract inflammatory and immunologic conditions"	"Lower respiratory tract disorders (excl obstruction and infection)"	"Respiratory, thoracic and mediastinal disorders"	"Resp"	1.00387e7	"Y"	"\"VENTILATION\ PNEUMONITIS

ETL

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets

```
In [11]: # Use the size function within println to show number of records in dataframe
         println("Records in meddra20: ", size(meddra20, 1))

         Records in meddra20: 120267
```

ETL

Ingest data

- Packages (bundle of functions) made more powerful by shared development
- Using packages (CSV.read) (read_sas7bdat)
- Data frames instead of datasets

```
# Concatenate previous table with report source information
demo_reac_out_source = join(
    demo_reac_out, rpsr17q2[:, [:primaryid, :rpsr_cod]], on = :primaryid, kind = :left
)
@prtnumrecs(demo_reac_out_source)

# Lastly, join the (Drug/Therapy/Indication) and (Demographic/Reaction/Outcome/Report Source)
# combined sets into one set with all this information in one place.
faers17 = join(drug_ind_ther, deletecols!(demo_reac_out_source, :caseid), on = :primaryid, kind = :left);
@prtnumrecs(faers17)
```

```
Number of records in drug_ind is 1189407
Number of records in drug_ind_ther is 1189460
Number of records in demo_reac is 987728
Number of records in demo_reac_out is 1243921
Number of records in demo_reac_out_source is 1244199
Number of records in faers17 is 6943850
```

ETL

Transform data

- Joins, filter, subset
- Mutations of variables is possible
- Describe() function like a proc contents

```
In [7]: describe(drug17q2)
```

Out[7]: 20 rows × 8 columns

	variable	mean	min	median	max	nunique	nmissing	eltype
	Symbol	Union...	Any	Union...	Any	Union...	Int64	DataType
1	primaryid	1.49595e8	38853455	1.34975e8	1357411410		0	Int64
2	caseid	1.32713e7	3885345	1.34911e7	13792410		0	Int64
3	drug_seq	6.13917	1	3.0	209		0	Int64
4	role_cod		C		SS	4	0	String
5	drugname		LOSARTIN		zyloric	47359	0	String
6	prod_ai		(1-743)-(1638-2332)-BLOOD-COAGULATION FACTOR VIII (SYNTHETIC HUMAN) FUSION PROTEIN WITH IMMUNOGLOBULIN G1 (SYNTHETIC HUMAN FC DOMAIN FRAGMENT), (1444-6')-(1447-9')-BIS(DISULFIDE) WITH IMMUNOGLOBULIN G1 (SYNTHETIC HUMAN FC DOMAIN FRAGMENT)		ZUCLOPENTHIXOL HYDROCHLORIDE	5085	25429	String
7	val_vbm	1.02109	1	1.0	2		0	Int64
8	route		Buccal		Vaginal	62	347905	String
9	dose_vbm		13.75 MG, UNK		¼ OF ONE PILL	106458	603062	String

Algorithms & Analytics

- Data frames prepared upfront and then passed to plot as functions
- Backends customizable (interactive JavaScript plots or PyPlot graphics)
- Different plotting libraries are available (VegaLite)
- Very flexible and operates through a “grammar of graphics” framework

In [26]:

```
#Julia visualization using VegaLite v0.6.0
using VegaLite

piegen |> @vlplot(:bar,
    x = {"gender:n",
    axis = {title = "Gender"}},
    title = "Adverse Events by Gender - Atacand 2017Q2",
    y = {:gen_pct, axis = {title = "Percent of AEs"}},
    color = {"gender:n", scale = {range = ["gray", "#e377c2", "#1f77b4"]}},
    height = 400,
    width = 600)
```

Conclusion

Julia has many more capabilities

- Machine Learning and AI
- High-level support for parallelism and cloud computing
- May be run from the command line like Python
- Can run Python or R commands using PyCall and RCall packages
- See Chris Hurley at the break if you are interested in the working group