Python and Julia for Inequality Analysis

Josep Espasa Reig

Data Scientist

Try to answer the following question

- **Should I use R, Python or Julia for the analysis of income and wealth distributions?**

- Presentation and code in GitHub repo:
  github.com/JosepER/gdansk_workshop

Briefly present…

- I use Python, R and Julia myself. I choose one or another depending on the characteristics of the task.
- Pedro has shown how to use R. I'll therefore focus on Python and Julia.

Briefly present…

- 1- Why (not) Python?
- 2- Using Python to read and process data for inequality estimates
- 3- Using Python for ML (examples)
- 4- Why (not) Julia?
- 5- Using Julia to read and process data for inequality estimates

Why (not) Python?

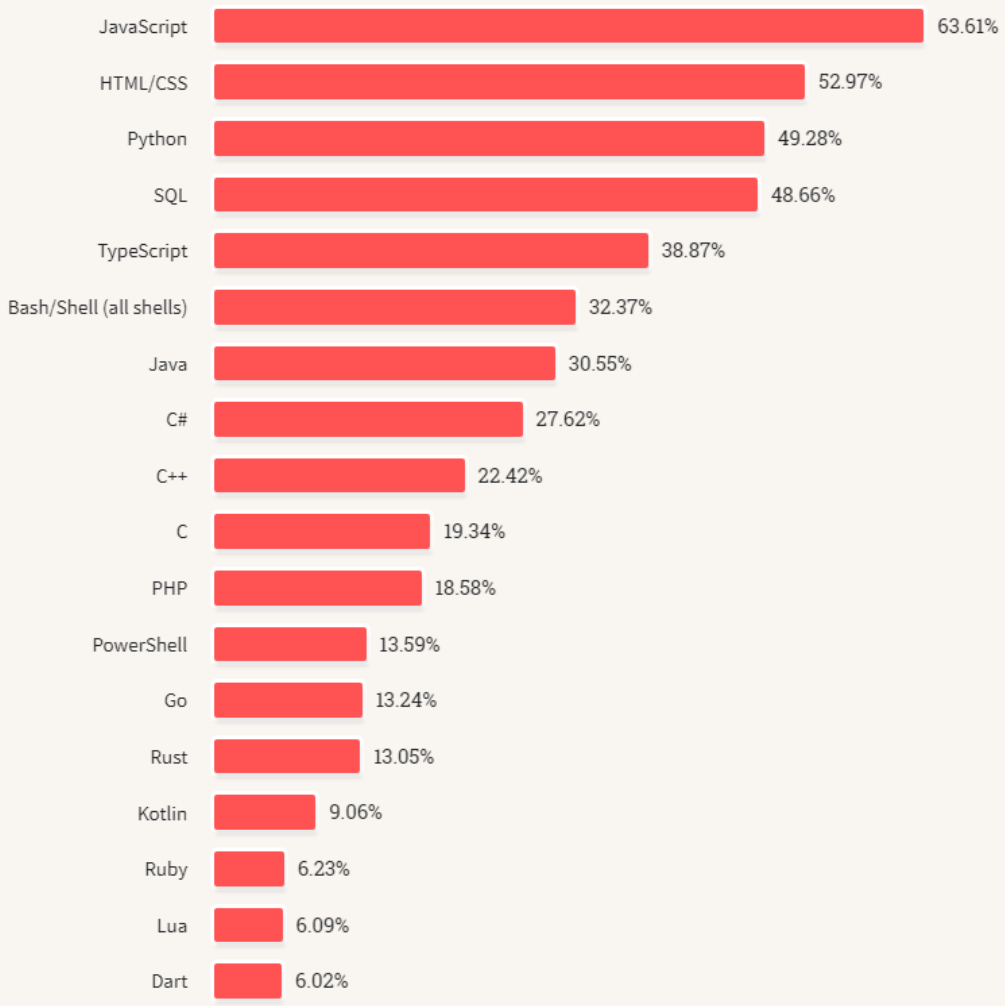
Why using Python

- "Python is the second-best language for everything" Dan Callahan at PyCon 2018
- Wide popularity. Massive community (and growing). Almost dominant in certain areas such as (ML).
- Great support by AI code generators.
- Slow but flexible (like R). Many packages are interfaces to faster (lower-level) programming languages (like C++ or Rust).
- Personal: feels more like a mature programming language than R.

From Stack Overflow 2023 survey

All Respondents | Professional Developers | Learning to Code | Other Coders

87,585 responses



Why (not) Python?

- **Why NOT using Python**
- Unlike R, it did not start with data and statistics in mind.
- It can feel 'clunky' to work with data at times (e.g. indexes for Pandas, or how certain libraries deal with missing values)
- R has some excellent libraries such as 'dplyr', 'ggplot2' and 'shiny'

Using Python to read and process data for inequality estimates (1)

01_read_process.py

```python
# .\env_gdansk\Scripts\activate
import pandas as pd

# 1- read .dta datasets in /data
file_h = pd.read_stata("data/it14ih.dta")
file_p = pd.read_stata("data/it14ip.dta")

# 2 - subset variables
file_h.columns.values  # list of variables
file_h = file_h[["hid", "hilabour", "nhhmem", "hpopwgt"]]

file_p.columns.values  # list of variables
file_p = file_p[["hid", "pid", "pilabour", "sex", "age", "marital",
                 "disabled", "educlev", "lfs", "status1", "ind1_c", "occ1_c"]]
```

Using Python to read and process data for inequality estimates (2)

```python
# 3- merge datasets
file = file_p.merge(file_h, on="hid", how="left")

# 4- Arrange the data
file = file[(file["age"] >= 30) & (file["age"] <= 60) & (file["pilabour"] > 0)]

# * print number of NAs in each variable
file.shape
file.isnull().sum()

# * delete rows with NAs
file = file.dropna()

# write file to csv
file.to_csv("clean_data/it14i.csv", index=False)
```

Using Python for ML

- Relies in external packages, like ‘sklearn’
- 3 main steps:
  - *Create instance of model*
  - *Fit the model*
  - *Predict + assess model fit*

Using Python for ML

```
58  # Display a model with a given lambda
59
60  model_with_desired_lambda = Lasso(alpha=optimal_lambda)
61
62  # Fit the model with the desired lambda
63  model_with_desired_lambda.fit(X, y)
64
65  # Get the coefficients
66  coefficients = model_with_desired_lambda.coef_
67
68  # Display the coefficients
69  for feature, coef in zip(X.columns, coefficients):
70      print(f"{feature}: {coef}")
```

- E.g. from 02_lasso.py' in github.com/JosepER/gdansk_workshop/tree/main/python_julia/py

Why (not) Julia?

- I have a previous presentation comparing R and Julia for Official Statistics (link)
- An open-source, dynamically typed language (like R and Python)
- Uses Just in time (JIT) Compilation
- Syntactically similar languages
- Julia feels more modern, easier to read and cleaner (personal opinion)
- R and Python have packages to run Julia code (and vice versa)
- Cons:
  - *the package ecosystem does not have the same maturity than the R and Python ones*

Why (not) Julia?

- Cons:
  - *The package ecosystem does not have the same maturity than the R and Python ones*
  - *Python also has ways of accelerating its code:*
    - E.g. Numba, Nuitka and (in the future) Mojo.
  - *AI code generators are currently awful at Julia.*

Syntactically similar languages

Julia

```julia
using DataFrames

df = DataFrame(a=[1,2,3], b=['x','y','z'])

df[1,1] # cell by index
df[:,1] # column by index
df[:,:b]# column by name
```

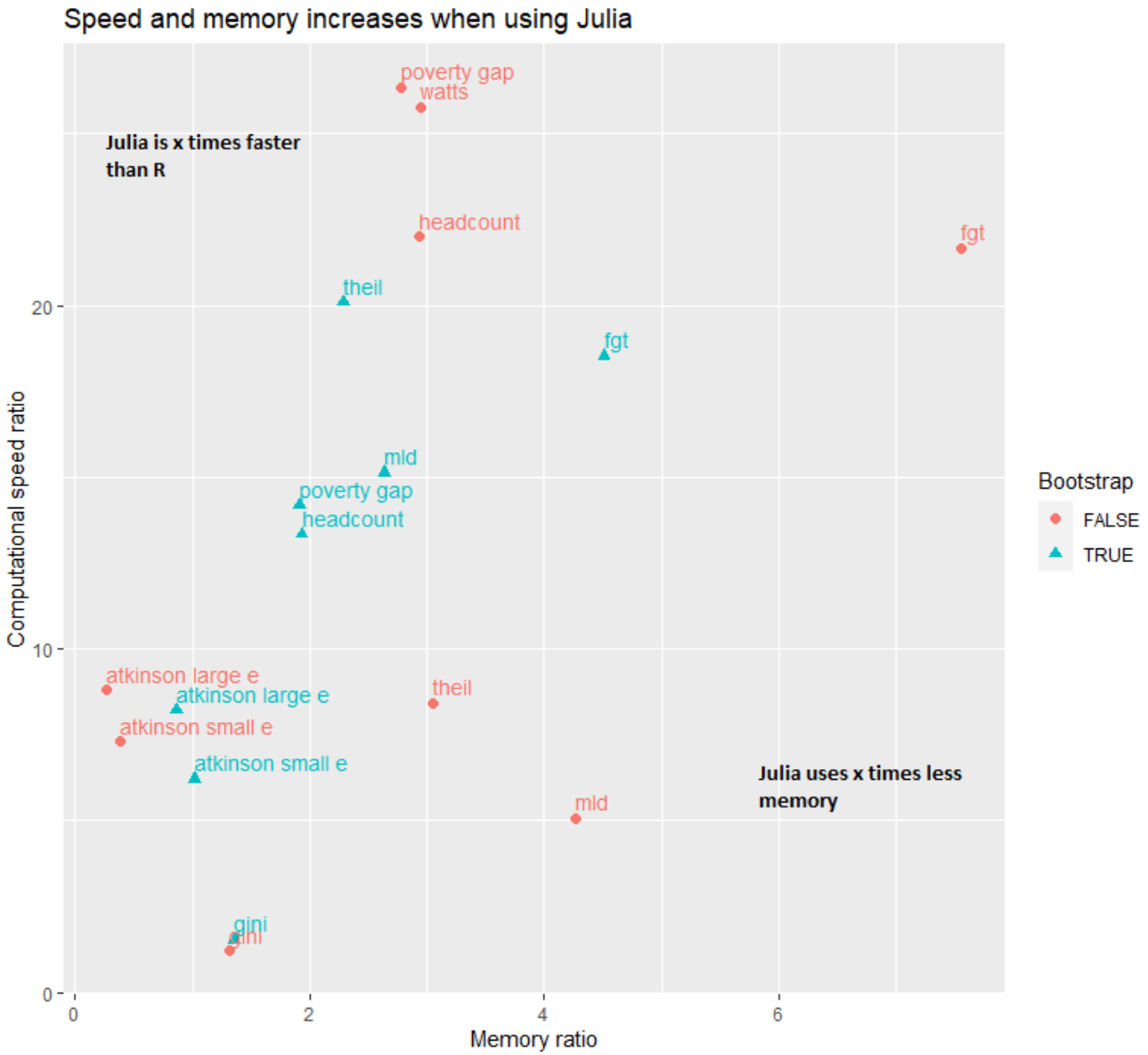
Python

```python
import pandas as pd

df = pd.DataFrame({'a':[1,2,3], 'b':['x','y','z']})

df.iloc[1,1] # cell by index
df.iloc[:,1] # column by index
df.loc[:,"b"]# column by name
```

R

```r
df <- data.frame(a=c(1,2,3), b=c('x','y','z'))

df[1,1] # cell by index
df[,1] # column by index
df[,"b"] # column by name
```

Julia is fast!



Using Julia

jl > 01_read_process.jl > ...

```julia
using DataFrames, StatFiles

# Read data
current_dir = dirname(@__FILE__)
parent_dir = dirname(current_dir)

df_h = DataFrame(load(parent_dir * "/data/it14ih.dta")) | 1000×110 DataFrame
df_p = DataFrame(load(parent_dir * "/data/it14ip.dta")) | 2384×191 DataFrame

# Subset variables
df_h = df_h[:,["hid", "hilabour", "nhhmem", "hpopwgt"]] | 1000×4 DataFrame
df_p = df_p[:,["hid", "pid", "pilabour", "sex", "age", "marital", "disabled",
    "educlev", "lfs", "status1", "ind1_c", "occ1_c"]] | 2384×12 DataFrame

# Merge df_h and df_p
df = leftjoin(df_h, df_p, on = :hid) | 2384×15 DataFrame

# Filter rows
df = df[(df[:,:age] .>= 30) .& (df[:,:age] .<= 60) .& (df[:,:pilabour] .> 0), :] | 648×15 DataFrame

# * delete rows with NAs
df = dropmissing(df)
```

Full script can be found in link

Conclusion

- My personal algorithm:
  - *R for scripting and explorations of data.*
  - *Python for building robust applications.*
  - *Julia only if I need to increase the speed of a process.*

Thank you for your attention
Questions are welcome !