Kezdi.jl: A data analysis package for economists

Miklós Koren[1, 2, 3, 4] and Gergely Attila Kiss[1, 2]

[1]Central European University, Vienna, Austria
[2]HUN-REN KRTK, Budapest, Hungary
[3]CEPR, London, United Kingdom
[4]CESifo, Munich, Germany

ABSTRACT

Economists overwhelmingly rely on proprietary data analysis languages such as Stata and MATLAB for their research computing needs. The transition to open-source languages like Julia presents various challenges due to differences in syntax, functionality, and best practices. We introduce `Kezdi.jl`, a data analysis package designed for economists that provides a Stata-like interface for working with data frames in Julia. The package is built on `DataFrames.jl` and related libraries, but uses a streamlined macro-based interface to eliminate common points of confusion. By emulating best practices from Stata, `Kezdi.jl` allows economists to be productive in Julia from day one. It supports a wide range of data wrangling and analysis tasks, including cleaning and transforming data, handling missing values, generating new variables, aggregating data, and running regressions.

Keywords

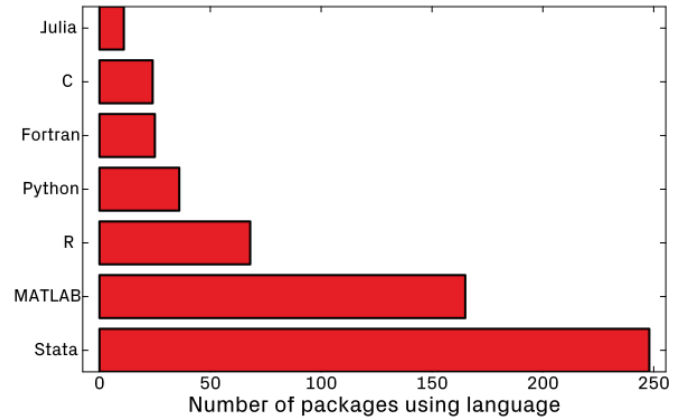Julia, Data analysis, Data wrangling, Stata

Fig. 1: Programming language usage in economics replication packages submitted to the Review of Economic Studies. A package can contain code in multiple languages.

1. Introduction

Research computing is central to the scientific workflow of economists. We studied 357 replication packages submitted to the Review of Economic Studies in recent years to understand what programming languages economists use. Figure 1 shows the number of packages that contain code in various languages. Stata is by far the most popular, used in 72% of packages, followed by MATLAB with 41%. In contrast, only 3% of replication packages contain any Julia code, which is even less prevalent than Fortran and C or C++.

Stata and MATLAB are often used together in the same replication package, with Stata focusing on data manipulation and analysis, while MATLAB handles computationally intensive tasks like model simulations. These findings highlight both the current dominance of proprietary scientific computing languages in economics and the large untapped potential for open source tools like Julia to make inroads in the field. Lowering the barriers to entry through familiar interfaces, as Kezdi.jl aims to do for Stata users, could play an important role in accelerating adoption.

Stata in particular has a dominant position in the data analysis and statistical modeling stages of the research workflow. 88% of code lines in a typical replication package are devoted to data wrangling and related programming tasks, with only 5% implementing statistical analysis [9]. This is no surprise, given that a typical empirical economics paper assembles and harmonizes data from several disparate sources.

The transition from proprietary languages like Stata or MATLAB to open-source ones like Julia presents several challenges. First, the syntax and mental model is very different. Stata is a data-centric language, where commands operate directly on columns of a dataset, and the details of the type system are hidden from the user. Second, Stata and MATLAB have had decades to build a rich library of statistical and econometric routines that are part of the core language. Third, they have developed coding styles and best practices that are ingrained in the muscle memory of economists. To name a concrete example, in Stata one can calculate a heteroskedasticity-robust standard error in a regression by simply adding the option `robust`. Achieving the same in Julia requires several more keystrokes even if the required packages are loaded.

Beyond syntax and functionality, reproducible research practices pose additional challenges when transitioning between languages. While Stata enforces a single-dataset workflow that naturally documents data transformations, Julia's more flexible approach requires careful attention to reproducibility. Kezdi.jl addresses this by providing clear audit trails of data transformations through its command syntax, similar to how Stata preserves transformation history in do-files [7].

```
1  @use "trade.dta"
2  @generate log_trade = log(trade)
3  @regress log_trade log_distance, robust
```

Fig. 2: Commands in Kezdi.jl follow Stata naming conventions and syntax. All commands start with the @ sign. Options are separated from arguments by a comma.

```
1  using Dates
2  @generate date = Date(year, month, day)
3
4  using Statistics
5  @collapse std_x = std(x), by(group)
```

Fig. 4: Functions are automatically vectorized, but only if necessary.

```
1   # Create a new variable that is 1 if x > 10, 0 otherwis
2   @generate y = cond(x > 10, 1, 0)
3
4   # Rename variables
5   @rename oldname newname
6
7   # Encode missing values
8   @mvencode y1 y2 y3 @if x < 0, mv(999)
9
10  # Collapse data by group, calculating the mean of x
11  @collapse avg_x = mean(x), by(group)
12
13  # Reshape data from wide to long format
14  @reshape long y, i(id) j(year)
```

Fig. 3: Some data wrangling commands illustrating the syntax of Kezdi.jl.

```
1   julia> using Kezdi
2   julia> df = DataFrame(x=[1, 2, missing], y=[missing, 3, 4])
3   julia> @collapse sum_x = sum(x)
4   1 1  DataFrame
5        Row      sum_x
6                 Int64?
7
8         1          3
9
10  julia> @generate z = x + y
11  3 3  DataFrame
12       Row      x         y         z
13                Int64?   Int64?   Int64?
14
15        1          1   missing   missing
16        2          2         3         5
17        3     missing       4   missing
```

Fig. 5: Missing values are handled in a logical way.

Kezdi.jl [1] is a data analysis package for economists that aims to ease the transition to Julia by providing a Stata-like user experience. It allows users to manipulate data frames and conduct statistical analysis using Stata-inspired macro commands that are both powerful and easy to read.

2. Key features of Kezdi.jl

The overarching design principle of Kezdi.jl is to provide a convenient and familiar interface for economists working with data frames. Here we highlight some key features that help achieve this.

2.1 Stata-like command syntax

Kezdi.jl has near complete coverage of Stata's data manipulation commands, including `egen`, `collapse`, `reshape` and `mvencode`. Combined with Julia's superior performance, this allows economists to easily translate their Stata workflows.

Some examples of common data wrangling tasks in Kezdi.jl:

2.2 Automatic column selection and vectorization

Variable names are automatically matched to data frame column names in Kezdi.jl. In standard Julia one would need to wrap these in : or string macros to refer to variables.

Function calls are also automatically vectorized. In the examples above, `log(trade)` is equivalent to `log.(trade)` and `cond(x > 10, 1, 0)` is equivalent to `cond.(x .> 10, 1, 0)` in base Julia.

This allows for concise and readable code that operates directly on data frame columns.

2.3 Use any Julia function

Because Kezdi.jl supports arbitrary Julia syntax within commands, user-defined and external package functions work seamlessly with data frame columns.

This allows economists to easily extend their data manipulation and modeling capabilities beyond Stata's built-in functions by leveraging Julia's package ecosystem.

2.4 Handling of missing values

Stata has special `missing` values and carefully defined rules for their propagation. For example, `sum(x)` in Stata returns the sum of non-missing values of x.

Missing value handling in Kezdi.jl follows these rules to ensure consistency with Stata:

—In logical expressions, `missing` is treated as `false`.

—Mathematical operations return `missing` if any of their arguments are missing.

—Aggregation functions skip missing values by default.

2.5 Syntactic sugar for operating row-wise

One of the most convenient features of Stata is the ability to restrict any command to a subset of rows using logical expressions with the `if` qualifier.

Kezdi.jl implements the same with the `@if` macro-call, but without an actually implemented macro. Logical expressions are automatically broadcast over rows.

2.6 Integration with Julia's package ecosystem

Kezdi.jl builds on the DataFrames.jl ecosystem [14] to provide its data manipulation features. It also integrates tightly with other packages:

---
[1] https://github.com/korenmiklos/Kezdi.jl

```
1  @replace trade = 0 @if ismissing(trade)
2  @regress log_trade log_distance @if trade > 0
3  @keep @if !ismissing(trade)
4  @collapse mean_x = mean(x) @if group ==
   "treatment"
```

Fig. 6: The @if macro can be used to limit the scope of any command.

—FreqTables.jl [1] for frequency tables via `@tabulate`

—FixedEffectModels.jl [5] for estimating regressions with many fixed effects

—Chain.jl for piping a sequence of commands with `@with`

—ReadStatTables.jl for importing Stata `.dta` files

2.7 Support for Reproducible Workflows

Kezdi.jl is designed with reproducibility as a core principle. The package follows established best practices for computational reproducibility [10] in several ways:

First, all data transformations are explicitly recorded through macro commands, creating an auditable trail similar to Stata do-files. This addresses the "hidden researcher degrees of freedom" problem identified by [6] in empirical research.

Second, the package provides a consistent environment across different machines by explicitly handling missing values and maintaining consistent random number generation seeds. This follows recommendations by [11] regarding numerical reproducibility across computing environments.

Finally, Kezdi.jl encourages organizing analysis as sequential scripts that can be executed from a main control file. This project organization pattern, recommended by [7], ensures that the full analysis can be reproduced with minimal manual intervention.

3. Comparison with related packages

Kezdi.jl is not the first attempt to provide a simplified data analysis workflow in Julia. TimeSeriesRecipes.jl [12] provides a grammar for time series operations. DataFramesMeta.jl [2] and SplitApplyCombine.jl [4] extend data frame operations.

On the R side, dplyr [15] has been a hugely successful abstraction. Tidier.jl [13] ports many ideas from dplyr to Julia. However, the dplyr "verb + data pipeline" model is very different from how economists approach data analysis in Stata.

Kezdi.jl was inspired by Douglass.jl, a work-in-progress package that also aims to provide a Stata-like user experience in Julia [3]. The two packages share many design principles, but Kezdi.jl has a more extensive feature set that covers a larger portion of a typical economics workflow.

4. Performance

Stata is often perceived as slow by its users, especially for larger datasets. A promise of moving to Julia is faster code execution.

We benchmarked selected Kezdi.jl commands against their Stata equivalents on a 1.2GB dataset with 15 million observations. The results are summarized in Table 1.

The speedup ranges from 3x for `@collapse` to 13x for `@egen`. This can provide a significant productivity boost for data-intensive research projects. The performance edge of Kezdi.jl is likely to grow as datasets get larger.

Command	Stata	Kezdi.jl	Speedup
@rename	0.23	0.03	7x
@generate	0.23	0.05	5x
@collapse	0.94	0.28	3x
@egen	5.00	0.37	13x
@regress	0.85	0.14	6x

Table 1. : Execution time (seconds) of equivalent Stata and Kezdi.jl commands. Benchmarks were run on a **is this true????** Windows 11 machine with an AMD Ryzen 7 processor and 32 GB RAM.

5. Conclusion

We introduced Kezdi.jl, a data analysis package for economists that provides a Stata-like interface to Julia's data manipulation and statistics ecosystem. Kezdi.jl aims to shorten the learning curve for economists transitioning from Stata, while also unlocking the power and expressiveness of Julia.

With Kezdi.jl, economists can manipulate data frames, calculate summary statistics, estimate regressions, and generate plots using familiar keystrokes and mental models. At the same time, the package opens the door to Julia's rich ecosystem of packages for optimization, machine learning, visualization, and high-performance computing.

Our hope is that Kezdi.jl will accelerate adoption of Julia in economics and other social sciences, and ultimately lead to better, more reproducible research. We invite researchers and developers to try out the package and share their feedback on how we can expand its capabilities to better meet their scientific computing needs.

References

6. References

[1] Pietro Alimena et al. Freqtables.jl. 2023.

[2] Hakan Arslan et al. Dataframesmeta.jl. 2023.

[3] Johannes Boehm. Douglass.jl. 2023.

[4] Tomás Pinho Davies et al. Splitapplycombine.jl. 2023.

[5] Matthieu Gomez et al. Fixedeffectmodels.jl. 2023.

[6] Nick Huntington-Klein, Andreu Arenas, Emily Beam, Marco Bertoni, Jeffrey R Bloem, Pralhad Burli, Naibin Chen, et al. The influence of hidden researcher decisions in applied microeconomics. *Economic Inquiry*, 59(3):944–960, 2021.

[7] Miklós Koren, Lars Vilhuber, Imola Csóka, Marie Connolly, and Joan Llull. Ten simple rules for creating a replication package. *Working Paper*, January 2024.

[8] Miklós Koren. Adoption of computational reproducibility in economics. *Working paper*, 2022.

[9] Miklós Koren. A typical empirical economics paper. *Working paper*, 2023.

[10] Matthew S Krafczyk, August Shi, Adhithya Bhaskar, Darko Marinov, and Victoria Stodden. Learning from reproducing computational results: introducing three principles and the reproduction package. *Philosophical Transactions of the Royal Society A*, 379(2197):20200069, 2021.

[11] Bruce D McCullough and Hrishikesh D Vinod. The numerical reliability of econometric software. *Journal of Economic Literature*, 37(2):633–665, 1999.

[12] Paweł Rybiński et al. Timeseriesrecipes. 2023.

[13] Josh Singer et al. Tidier.jl. 2022.

[14] Jacob White et al. Dataframes.jl. 2023.

[15] Hadley Wickham et al. dplyr: A grammar of data manipulation. 2023.