Section 1: Week 1: Evaluate Tools for Statistical Applications

Nate Bachmeier

TIM-7101: Statistics for Technology Leaders

August 9, 2020

Northcentral University

# Evaluate Tools for Statistical Applications

When a project requires significant investments into statistical modeling, it can quickly become prohibitively expensive and tedious to perform these calculations by hand (Denis, 2015). Instead, the analyst must defer to software solutions to transform raw data values into business intelligence. Luckily there is a vast ecosystem of tooling that can specialize in different scenarios, such as interactive ‘slicing and dicing’ versus offline batch processing. An organization like NCU-C also must consider how the various products align with existing data platforms. For instance, a business that relies on traditional relational data stores might have more flexibility than another institution that requires graph technologies. Just as remodeling a kitchen necessitates hammers, screwdrivers, and tape measures—statistical applications can involve multiple tools. Consider the situation where data begins life in unstructured data lakes, and through an extract transform and load (ETL) process, becomes a geographical map. This situation might call for Python scripts to parse records into a tabular format. Next, confirming the dataset is complete can call upon programs like Microsoft Excel, IBM SPSS, or Tableau to visually create pivot tables and charts. Finally, importing the geospatial data into software like QGIS provides a canvas for further domain-specific explorations.

# Programming Interfaces

## Static versus Dynamic

While any programming language can perform statistical calculations, there are inherent advantages that make one a more natural choice over another in specific-contexts. For instance, when a project begins, the requirements are more nebulous, and this shifts the focus to developer efficiencies over runtime performance (see Table 1). During this initial period, scripting languages like R and Python allow the analyst to experiment and find the best modeling technique. Later, porting these scripts to C++ and Fortran increases the cost for additional modifications but provides hardware-native execution speeds. These low-level languages also enable controls over memory management, remote procedure calls, and local thread scheduling. Engineering teams can use these primitives to scale-out systems to enormous datasets and related High-Performance Computing (HPC) tasks.

Table 1: Programming Languages

|  |  |  |  |
| --- | --- | --- | --- |
| Optimization Goal | Language | Type | Vector Support |
| Rapid Prototyping | Python (NumPy, 2020) | General-Purpose | NumPy and Pandas |
| R (R-Project, 2020) | Statistical language | First-class citizen |
| Matlab  (MathWorks, 2020) | Linear algebra DSL | First-class citizen |
| Octave  (Eaton, 2020) | Open Source Matlab | First-class citizen |
| Runtime Performance | C++ | General-Purpose | Open-Source Libraries |
| Fortran | General-purpose | Open-Source Libraries |

## Matrix versus Scalar Models

Performing data transformations across large quantities of data will require significant computational resources. These calculations must happen efficiently through matrix and vector processing, not general-purpose loop-constructs. For instance, iterating through a list of objects can encounter much overhead in accounting for index offsets. Meanwhile, data frames libraries are specifically optimized to apply manipulations on high-dimensional data (NumPy, 2020). These optimizations enable packages to make several updates at the same time or scale-out distinct partitions across distinct processing units, such as Central Processing Units (CPU), General Purpose Graphics Processing Units (GPGU/GPU), Application-specific Integrated Circuits (ASIC), and Field-Programmable Gate Arrays (FPGA).

# Interactive Software Packages

## General Purpose

Analysts can import data into commercial software programs like Microsoft Excel, IBM SPSS, and SAS to model statistical functions (see Table 2). These tools offer domain-specific languages to reduce the complexity of common use-cases. For instance, all three toolsets support built-in support standard distribution measurements (e.g., standard deviation and means), along with more sophisticated operations like correlation and error rates. Users can also generate charts from these statistical outputs to compare trends visually.

Table 2: Software Packages

|  |  |  |
| --- | --- | --- |
| Licensing | Package | Audience |
| Closed Source | Microsoft Excel | General Analyst |
| IBM SPSS | Intermediate Audience |
| SAS | Advanced Modeling |
| Tableau | Dashboarding/Exploration |
| Open Source | Jupyter Notebooks | Programmer |
| QGIS | Geospatial analysis |
| Gephi | Graph analyst |

## Specialized Visualizers

It can be challenging to represent all trends and relationships within data as a simple line, bar, pie, or scatter graph. For these more complex situations, the organization might seek dashboarding technologies like Tableau, Microsoft PowerBI, and Amazon QuickSight. These products can connect multiple representations into a single view, enabling a mixture of macro and micro perspectives. For example, the business sales dashboard enumerate Key Performance Indicators (KPI) and then decompose those figures into supporting evidence. A business analyst can use this information to understand potential causalities and uncover new insights (Hawking, 2012). However, these platforms tend to support a finite list of statistical operations. If the user wants to deviate from this list, it can require calculating values out-of-band through additional ETL steps. Depending on the organizational makeup, these additional actions could require scheduling engineering tasks that must compete with other business commitments.

## Custom Solutions

When the business has a particular need, then it might choose to extend open-source software packages. For instance, researchers can use Jupyter Notebooks to host custom graphs, proprietary calculations, and notes. One of the critical advantages of this approach is that results are reproducible inline (Jupyter, 2020). While many plugins and extensions are freely available, others require commercial licensing. Purchasing specific extensions can reduce the time to market of value differentiating functionality by removing the need to reinvent wheels. Other open-source products like d3js and HighCharts offer a language for describing data presentation but lack statistical functions. Since these different products have unique specializations, business analysts might need to glue together multiple products as part of a broader statistical application strategy.

# Conclusion

Some situations require a hammer, while others need a screwdriver or wrench. Similarly, statistical modeling relies on various tools for approaching the contextually sensitive problem at hand. This context must consider the intended audience and potential data volume that must be analyzed. The audience is likely to evolve with the requirements of the statistical application, such as initially focusing on data analyst and rapid prototyping. As the solution matures, the external customers become crucial, and the emphasis aligns more with high-performance programming languages, such as C++. During this journey, the organization also needs tools that enable interactive exploration of descriptive statistics and pivot tables. These solutions come from a collection of open and closed source providers, each with its unique quirks.

# References

Denis, D. (2015). *Applied Univariate, Bivariate, and Multivariate Statistics* (1st ed.). John Wiley & Sons, Incorporated.

Eaton, W. (2020). *Table of Contents*. Retrieved from GNU Octave: https://octave.org/doc/v5.2.0/

Hawking, P. (2012). Business Intelligence Excellence: A Company's Journey to Business Intelligence Maturity. *International Journal of Technology, Knowledge, and Society, 8*(2), 91-99. doi:https://doi.org/10.18848/1832-3669/CGP/v08i02/56274

Jupyter. (2020). *Documentation*. Retrieved from Jupyter: https://jupyter.org/documentation

MathWorks. (2020). *Matlab language fundamentals*. Retrieved from MathWorks: https://www.mathworks.com/help/matlab/language-fundamentals.html

NumPy. (2020). *NumPy v1.19 Manual*. Retrieved from NumPy: https://numpy.org/doc/stable/

R-Project. (2020). *What is R*. Retrieved from The R-Project: https://www.r-project.org/about.html