Software Systems for Policymakers and Data Scientists

Harnessing Big Data for National Statistics and Evidence-Based Policymaking

18-19 August, 2025
Blantyre, MW

NSO | UC Berkeley | World Bank

Overview

- Computing and software choices
- Statistical Computing/Data Science
  - Python
  - R
  - Stata
- Mechanics of running code in Python and R
- Further resources

Goal is not for you to memorize these slides, but to understand the landscape of choices and have this document as a reference.

Computing choices

- Open source versus closed source
- Paid vs free
- Cloud vs local

Open source vs closed source

Open Source: The software's source code is publicly available, allowing anyone to view, modify, and distribute it. Often community-driven, it emphasizes transparency and collaboration.

Closed Source: The source code is kept secret and is proprietary to the company or developer. Users are given a license to use the software but cannot access or change its internal workings.

Paid vs free

Paid: Users must pay a fee to use the software, either as a one-time purchase or a recurring subscription. Payment typically grants access to features, updates, and dedicated support.

Free (Freeware/Gratis): The software can be downloaded and used at no financial cost. However, it may have limitations, such as restricted features, ads, or a lack of professional support.

Local vs cloud

Local (On-Premise): The software is installed and runs directly on a user's computer or a company's private servers. It does not require an internet connection for core functionality but may be less scalable and require manual updates.

Cloud: The software is hosted on remote servers and accessed via the internet. This model offers flexibility, as users can access their applications and data from any device with an internet connection.

Statistical Computing/ Data Science

Programming languages for statistical computing

Main choices:

- Python
- R
- Stata

Other choices:

- SAS
- SPSS
- Julia
- JMP

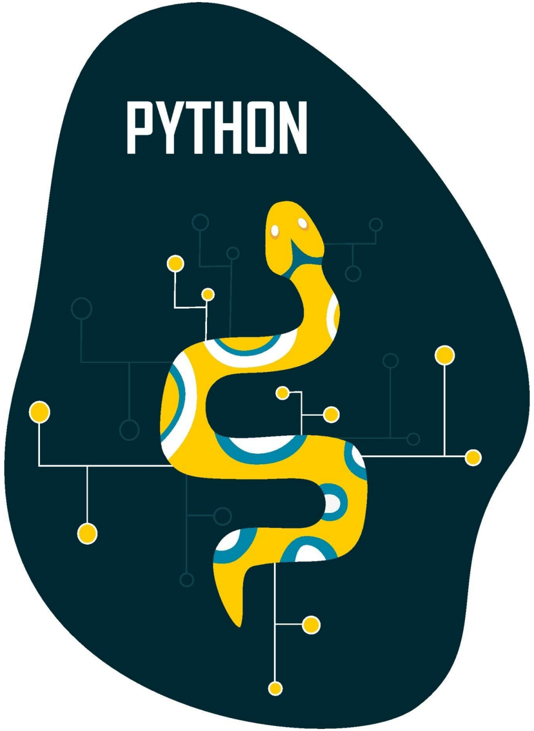
Python

Versatility: A general-purpose programming language. It is a great choice for end-to-end projects that require more than just statistical analysis, such as web scraping, automation, and machine learning.

Key Libraries: Statistical/numerical computing is powered by specialized libraries. **Pandas** and **NumPy** are foundational for data manipulation and numerical operations. **SciPy** and **Statsmodels** provide a comprehensive range of statistical tests and models. **Scikit-learn** is the standard for machine learning.

Learning Curve: The learning curve is moderate. Python's clean syntax is beginner-friendly, but mastering the full data science ecosystem requires learning multiple libraries.

Community & Use: Has a massive, diverse user base in data science, software engineering, and academia. Widely used in the industry for production environments.



numpy, scipy

NumPy (Numerical Python) is the foundational library for scientific computing in Python. It provides a powerful **N-dimensional array object** and functions for working with these arrays. It's the backbone for most other data science libraries.

SciPy (Scientific Python) is a library of algorithms and functions built on top of NumPy. It offers a wide range of modules for scientific and technical computing, including optimization, linear algebra, signal processing, and statistics.

pandas

A powerful and flexible library for **data manipulation and analysis**.

It introduces two key data structures: `DataFrame` for 2D tabular data (like a spreadsheet or SQL table) and `Series` for 1D data.

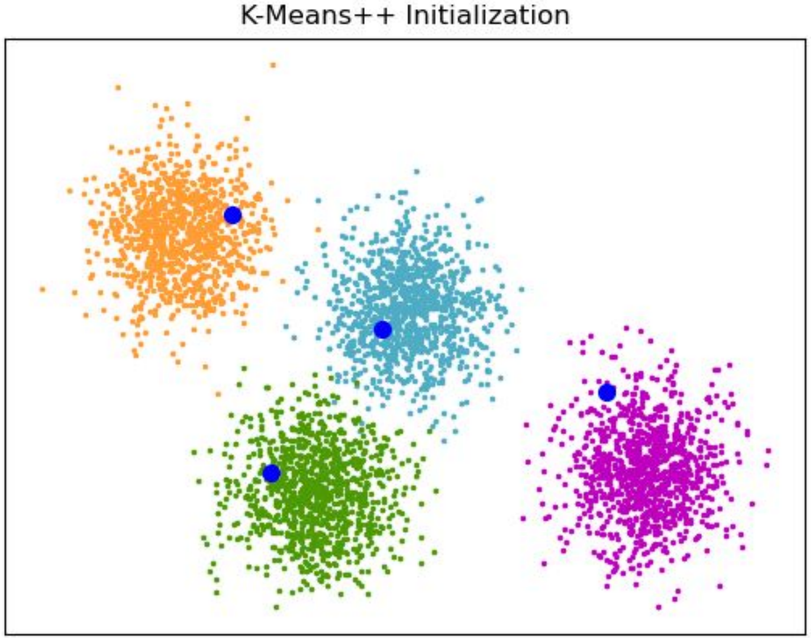
Pandas makes it easy to handle messy, real-world data by providing functions for tasks like cleaning, merging, and reshaping data.

scikit-learn

The de facto standard library for **classical machine learning** in Python.

It provides a consistent and simple interface to a vast collection of algorithms for classification, regression, clustering, dimensionality reduction, and model selection.

It's built on NumPy, SciPy, and matplotlib, making it a key component of the scientific Python stack.



tensorflow/pytorch

Tensorflow (Google) and Pytorch (Meta) are frameworks for machine learning and deep learning.

They are focused on building and training **neural networks** and are highly optimized for performance on various hardware (CPUs, GPUs, TPUs).

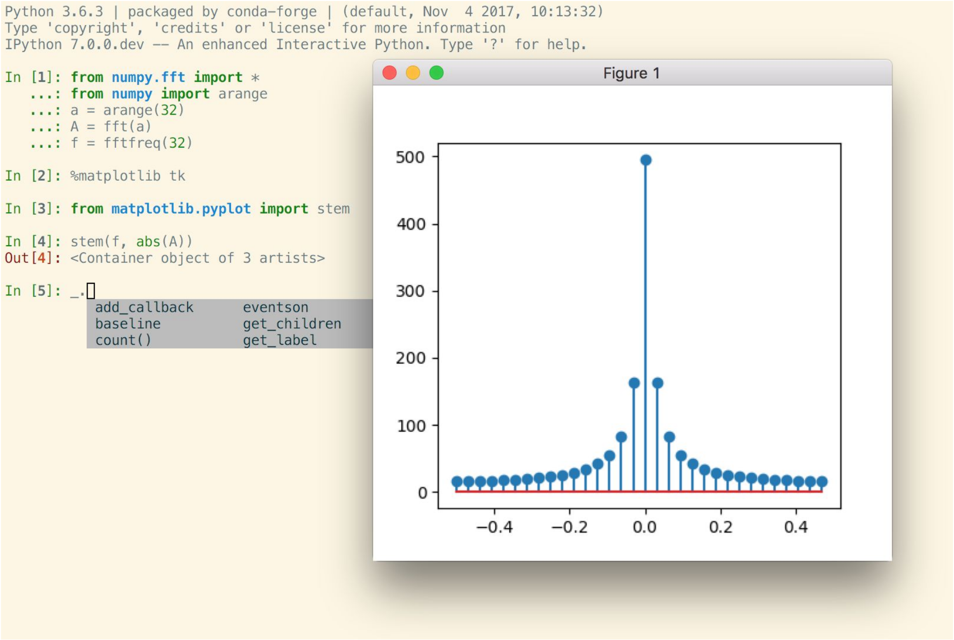
While a steeper learning curve than Scikit-learn, they offers the flexibility and power needed for cutting-edge research and large-scale applications.

IPython

An interactive command shell that provides a significant upgrade to the standard Python interpreter.

It features powerful additions like **tab completion, rich media display (e.g., inline plots), and command history**, making it an essential tool for exploratory data analysis.

IPython is the kernel that powers the popular Jupyter Notebook.



```
Python 3.6.3 | packaged by conda-forge | (default, Nov  4 2017, 10:13:32)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.0.0.dev -- An enhanced Interactive Python. Type '?' for help.

In [1]: from numpy.fft import *
   ...: from numpy import arange
   ...: a = arange(32)
   ...: A = fft(a)
   ...: f = fftfreq(32)

In [2]: %matplotlib tk

In [3]: from matplotlib.pyplot import stem

In [4]: stem(f, abs(A))
Out[4]: <Container object of 3 artists>

In [5]: _.
          add_callback     eventson
          baseline         get_children
          count()          get_label
```

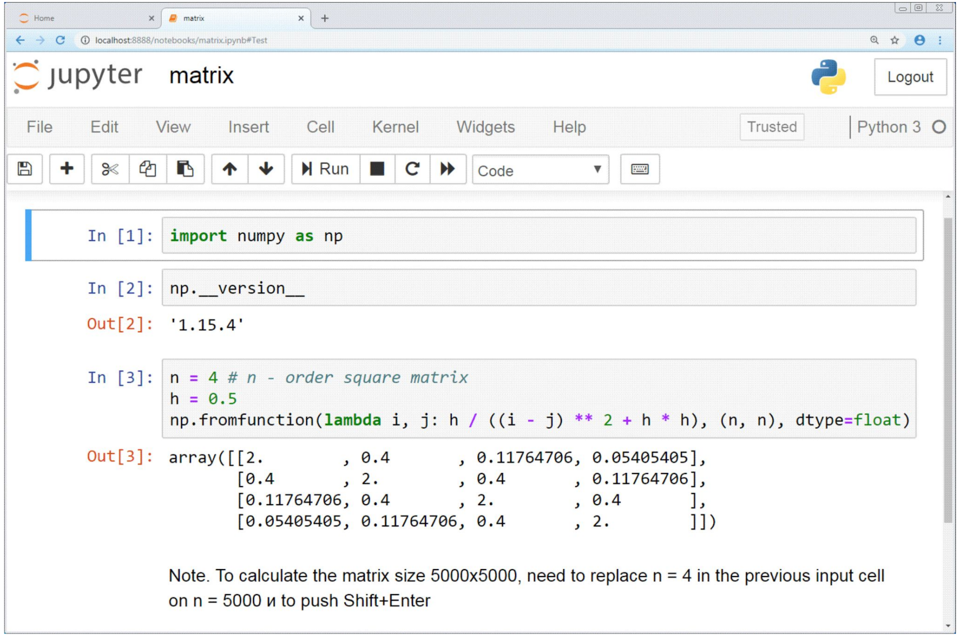
Jupyter Notebook

An interactive, web-based environment that allows users to create and share documents containing **live code, equations, visualizations, and narrative text**.

They are structured into **cells**, which can be either code or Markdown text. This enables a "computational narrative" where you can interleave explanations and analysis with the code and its output.

Jupyter Notebooks are an indispensable tool for data science because they provide a highly **reproducible and shareable workflow** for exploratory data analysis, data cleaning, and model prototyping.

Also runs Julia and R code.



```
In [1]: import numpy as np

In [2]: np.__version__

Out[2]: '1.15.4'

In [3]: n = 4 # n - order square matrix
        h = 0.5
        np.fromfunction(lambda i, j: h / ((i - j) ** 2 + h * h), (n, n), dtype=float)

Out[3]: array([[2.        , 0.4       , 0.11764706, 0.05405405],
               [0.4       , 2.        , 0.4       , 0.11764706],
               [0.11764706, 0.4       , 2.        , 0.4       ],
               [0.05405405, 0.11764706, 0.4       , 2.        ]])
```

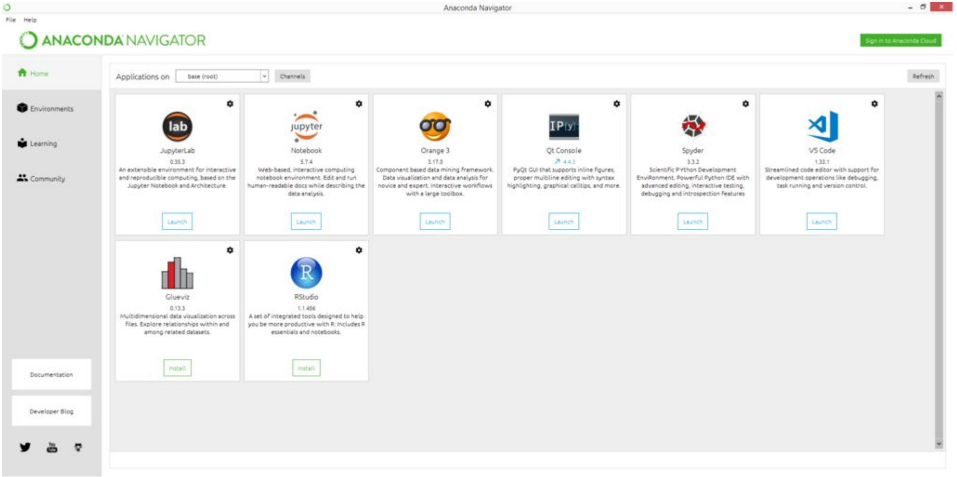
Note. To calculate the matrix size 5000x5000, need to replace n = 4 in the previous input cell on n = 5000 и to push Shift+Enter

Anaconda

A free and open-source **distribution** of Python and R for scientific computing.

It simplifies the setup process by coming pre-packaged with over 7,500 data science packages, including all the libraries mentioned above.

Its key feature is **Conda**, a powerful package and environment manager that helps you create isolated, conflict-free environments for different projects.



An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Streamlined code editor with support for development operations like debugging, task running and version control.

Multidimensional data visualization across files. Explore relationships within and among related datasets.

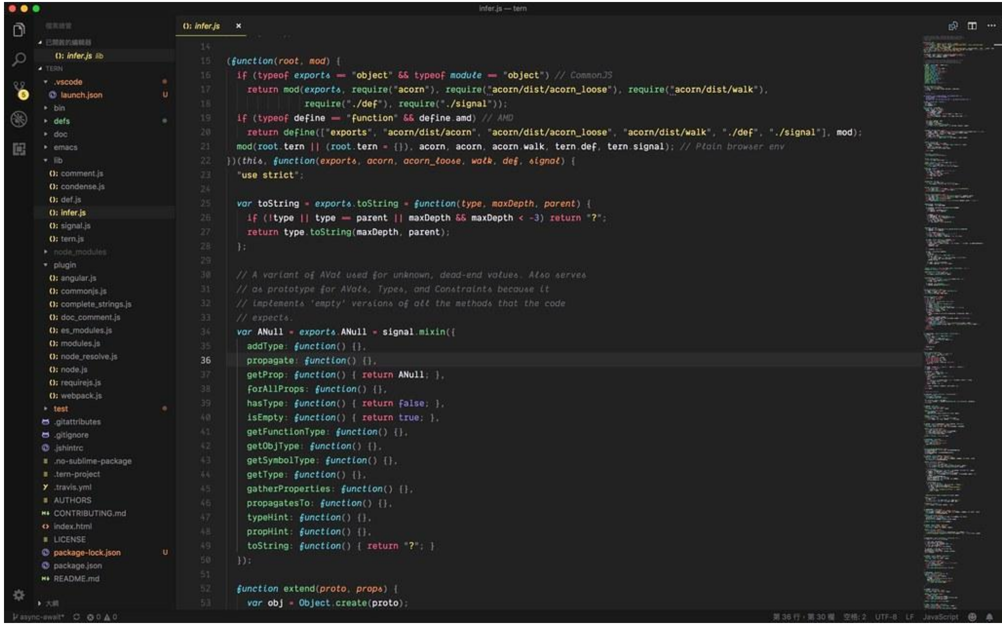
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

VSCode

A free, lightweight, and highly customizable **code editor** from Microsoft.

With its extensive marketplace of extensions, it can be transformed into a powerful integrated development environment (IDE) for Python, data science, and many other programming languages.

It provides excellent support for debugging, Git integration, working with Jupyter Notebooks, and LLMs (like ChatGPT).



```
(function(root, mod) {
  if (typeof exports == "object" && typeof module == "object") // CommonJS
    return mod(exports, require("acorn"), require("acorn/dist/acorn_loose"), require("acorn/dist/walk"),
               require("./def"), require("./signal"));
  if (typeof define == "function" && define.amd) // AMD
    return define(["exports", "acorn/dist/acorn", "acorn/dist/acorn_loose", "acorn/dist/walk", "./def", "./signal"], mod);
  mod(root.tern || (root.tern = {}), acorn, acorn, acorn.walk, tern.def, tern.signal); // Plain browser env
})(this, function(exports, acorn, acorn_loose, walk, def, signal) {
  "use strict";

  var toString = exports.toString = function(type, maxDepth, parent) {
    if (!type || type == parent || maxDepth && maxDepth < -3) return "?";
    return type.toString(maxDepth, parent);
  };

  // A variant of AVal used for unknown, dead-end values. Also serves
  // as prototype for AVals, Types, and Constraints because it
  // implements 'empty' versions of all the methods that the code
  // expects.
  var ANull = exports.ANull = signal.mixin({
    addType: function() {},
    propagate: function() {},
    getProp: function() { return ANull; },
    forAllProps: function() {},
    hasType: function() { return false; },
    isEmpty: function() { return true; },
    getFunctionType: function() {},
    getObjType: function() {},
    getSymbolType: function() {},
    getType: function() {},
    gatherProperties: function() {},
    propagatesTo: function() {},
    typeHint: function() {},
    propHint: function() {},
    toString: function() { return "?"; }
  });

  function extend(proto, props) {
    var obj = Object.create(proto);
```

R

Purpose-Built: Created by statisticians for statistical computing and graphics. It is unparalleled for its depth and breadth of statistical packages.

Key Packages: The **tidyverse** is a popular collection of packages (**dplyr**, **ggplot2**, **tidyr**) that provide a consistent and intuitive approach to data manipulation and visualization. The **CRAN** (Comprehensive R Archive Network) repository hosts over 18,000 user-contributed packages for virtually any statistical need.

Learning Curve: Can have a steeper initial learning curve due to its unique syntax, but it's exceptionally efficient for statistical tasks once mastered.

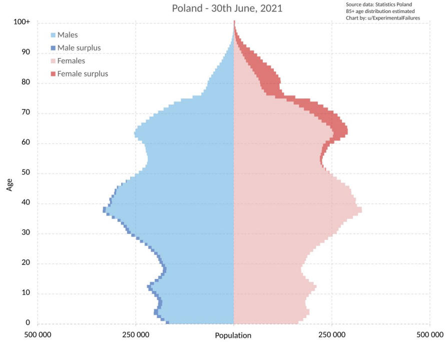
Community & Use: The primary language for statisticians and academic researchers. It's often favored for its state-of-the-art statistical methods and publication-quality graphics.

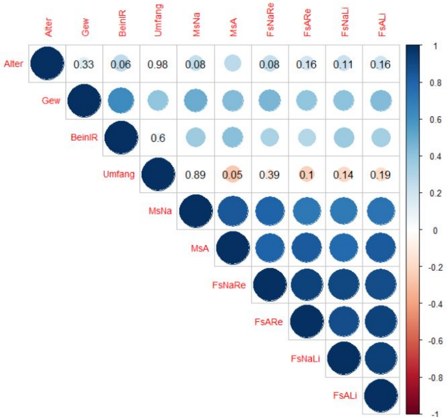
tidyverse

A collection of packages in R designed with a **consistent philosophy and grammar** to make data manipulation and visualization more intuitive.

It operates on the principle of **"tidy data,"** where each variable is a column, each observation is a row, and each type of observational unit is a table.

Key packages include `dplyr` for data manipulation, `tidyr` for tidying data, and `ggplot2` for plotting. The consistent syntax and "piping" operator (`%>%`) streamline a user's workflow.





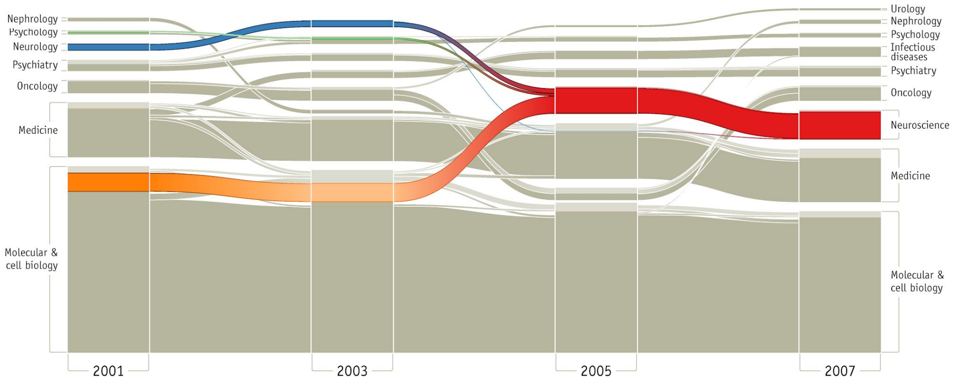
Alter
0.13 Gew
0.26 0.63 BeinIR
0.00 0.40 0.07 Umfang
0.23 0.50 0.36 -0.02 MsNa
0.27 0.44 0.41 -0.26 0.85 MsA
0.24 0.46 0.32 -0.12 0.80 0.80 FsNaRe
0.19 0.40 0.29 -0.22 0.72 0.83 0.93 FsARe
0.22 0.40 0.37 -0.20 0.71 0.78 0.91 0.88 FsNaLi
0.19 0.43 0.33 -0.18 0.74 0.83 0.89 0.93 0.93 FsALi
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

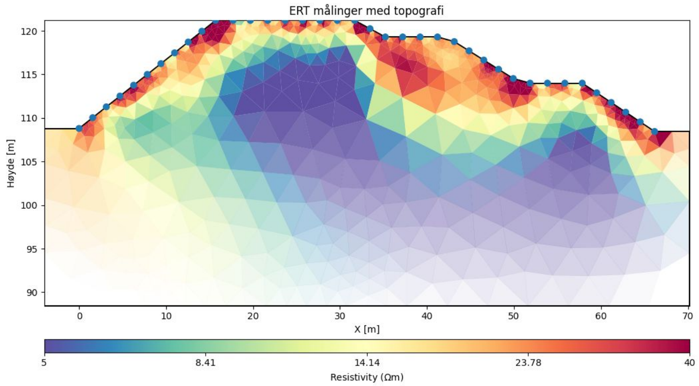
ggplot2

A powerful and widely used R package for creating high-quality **data visualizations**. It is a core component of the Tidyverse.

It is based on the **"Grammar of Graphics,"** a conceptual framework that allows you to build complex plots by combining simple components like data, geometric objects (points, lines), and aesthetic mappings (color, size, shape).

Unlike other plotting libraries, `ggplot2` enables you to create sophisticated and aesthetically pleasing graphics with minimal code, making it a go-to for exploratory data analysis and publication-ready plots.





glm

A fundamental function in the base R installation for fitting **generalized linear models**.

GLMs are a flexible extension of ordinary linear regression that allows for **response variables with different error distributions**, such as binomial (for logistic regression), Poisson (for count data), and gamma.

The `glm()` function provides a single, consistent interface for fitting a variety of models, making it a cornerstone for a wide range of statistical analyses beyond standard linear models.

Machine learning libraries in R

caret (Classification and Regression Training):
Provides a unified interface for training and evaluating various machine learning models, simplifying the workflow for tasks like classification and regression.

randomForest:
Implements the Random Forest algorithm, an ensemble method effective for both classification and regression.

xgboost:
A highly efficient and popular library for gradient boosting, known for its performance in competitive machine learning scenarios.

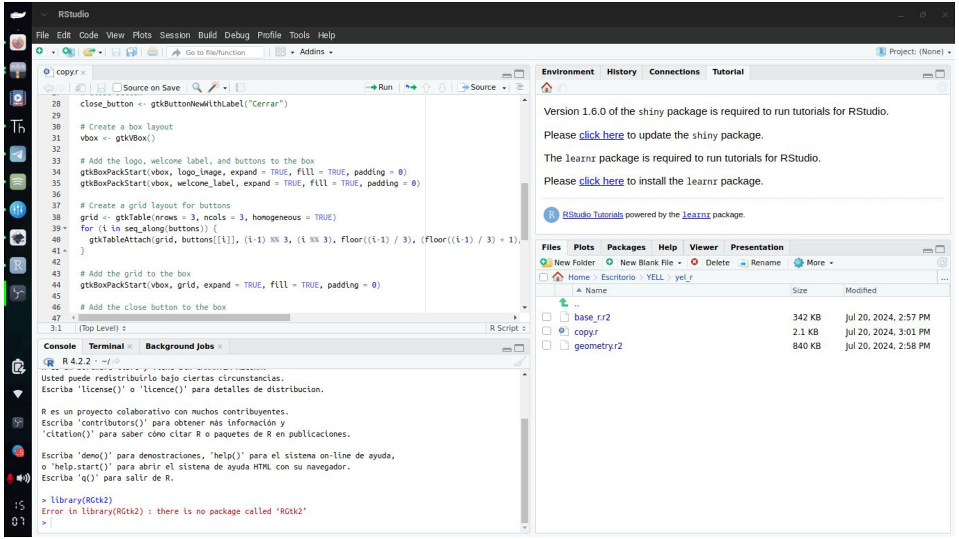
e1071:
Contains implementations of various algorithms, including Support Vector Machines (SVMs) and Naive Bayes classifiers.

RStudio

A popular and comprehensive **integrated development environment (IDE)** specifically for R.

It simplifies the process of coding in R by providing a structured layout with a console, a script editor, a history pane, a file browser, and a "plots" pane for viewing visualizations.

RStudio is highly valued by R users for its powerful features like debugging tools, project management, and seamless integration with packages like the Tidyverse, making it the **standard environment for R programming**.



```r
close_button <- gtkButtonNewWithLabel("Cerrar")

# Create a box layout
vbox <- gtkVBox()

# Add the logo, welcome label, and buttons to the box
gtkBoxPackStart(vbox, logo_image, expand = TRUE, fill = TRUE, padding = 0)
gtkBoxPackStart(vbox, welcome_label, expand = TRUE, fill = TRUE, padding = 0)

# Create a grid layout for buttons
grid <- gtkTable(nrows = 3, ncols = 3, homogeneous = TRUE)
for (i in seq_along(buttons)) {
  gtkTableAttach(grid, buttons[[i]], (i-1) %% 3, (i %% 3), floor((i-1) / 3), (floor((i-1) / 3) + 1),
}

# Add the grid to the box
gtkBoxPackStart(vbox, grid, expand = TRUE, fill = TRUE, padding = 0)

# Add the close button to the box
```

```
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

> library(RGtk2)
Error in library(RGtk2) : there is no package called ‘RGtk2’
>
```

Version 1.6.0 of the `shiny` package is required to run tutorials for RStudio.

Please click here to update the `shiny` package.

The `learnr` package is required to run tutorials for RStudio.

Please click here to install the `learnr` package.

RStudio Tutorials powered by the `learnr` package.

Name	Size	Modified
..		
base_r.r2	342 KB	Jul 20, 2024, 2:57 PM
copy.r	2.1 KB	Jul 20, 2024, 3:01 PM
geometry.r2	840 KB	Jul 20, 2024, 2:58 PM

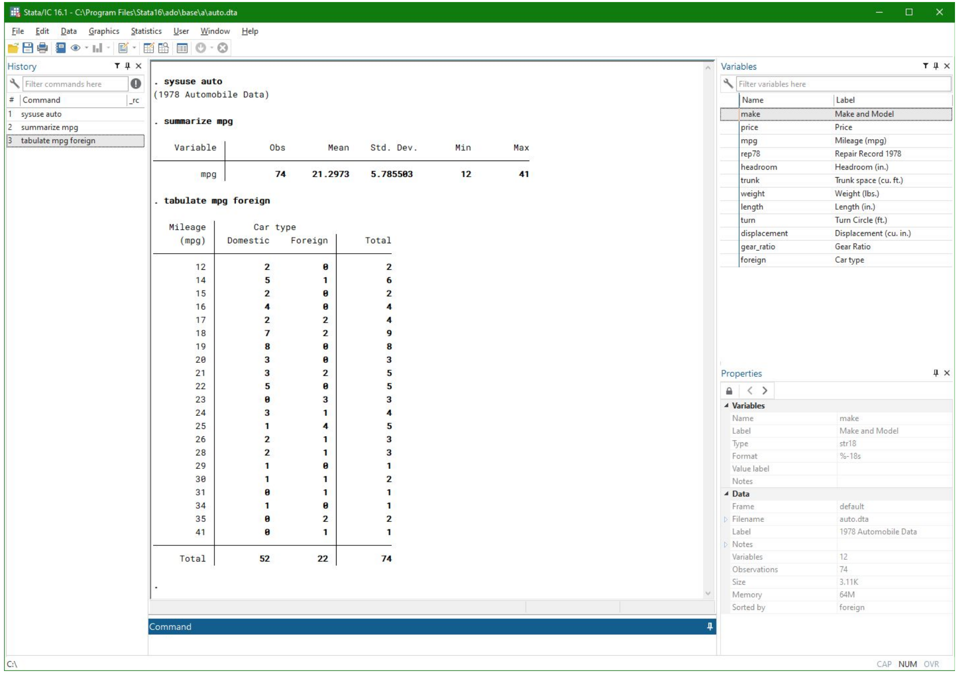
Stata

Proprietary & Specialized: A commercial, user-friendly statistical software package particularly strong in econometrics, biostatistics, and panel data analysis.

Strengths: Offers a straightforward, command-driven syntax and a comprehensive set of built-in commands. It excels at data management and producing reproducible research through "do-files" that document all steps. It also includes an easy-to-use graphical user interface (GUI).

Learning Curve: Generally considered the easiest to learn for those with a statistics background who are not strong programmers, thanks to its intuitive commands and strong documentation.

Community & Use: Widely used in economics and other social sciences, as well as in public health research. Its key advantage is a reliable, all-in-one platform with consistent features.



#	Command	_rc
1	sysuse auto	
2	summarize mpg	
3	tabulate mpg foreign	

```
. sysuse auto
(1978 Automobile Data)

. summarize mpg

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
         mpg |         74     21.2973    5.785503         12         41

. tabulate mpg foreign

   Mileage |       Car type
     (mpg) |  Domestic    Foreign |     Total
-----------+----------------------+----------
        12 |         2          0 |         2
        14 |         5          1 |         6
        15 |         2          0 |         2
        16 |         4          0 |         4
        17 |         2          2 |         4
        18 |         7          2 |         9
        19 |         8          0 |         8
        20 |         3          0 |         3
        21 |         3          2 |         5
        22 |         5          0 |         5
        23 |         0          3 |         3
        24 |         3          1 |         4
        25 |         1          4 |         5
        26 |         2          1 |         3
        28 |         2          1 |         3
        29 |         1          0 |         1
        30 |         1          1 |         2
        31 |         0          1 |         1
        34 |         1          0 |         1
        35 |         0          2 |         2
        41 |         0          1 |         1
-----------+----------------------+----------
     Total |        52         22 |        74

.
```

Name	Label
make	Make and Model
price	Price
mpg	Mileage (mpg)
rep78	Repair Record 1978
headroom	Headroom (in.)
trunk	Trunk space (cu. ft.)
weight	Weight (lbs.)
length	Length (in.)
turn	Turn Circle (ft.)
displacement	Displacement (cu. in.)
gear_ratio	Gear Ratio
foreign	Car type

Variables	
Name	make
Label	Make and Model
Type	str18
Format	%-18s
Value label	
Notes	
Data	
Frame	default
Filename	auto.dta
Label	1978 Automobile Data
Notes	
Variables	12
Observations	74
Size	3.11K
Memory	64M
Sorted by	foreign

Example: Python vs R vs Stata

Let’s look at how to generate synthetic data, and run multiple regression in Python, R, and Stata.

Example: Python vs R vs Stata

```
# Import necessary libraries
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression

# Set seed for reproducibility
np.random.seed(12345)

# Create a dictionary of variables
df = pd.DataFrame(data={f'x{i}': np.random.normal(size=200) for i in range(1, 5)})

# Generate the dependent variable y
df['y'] = 1.5 + 0.8 * df['x1'] - 0.7 * df['x2'] + 1.1 * df['x3'] - 1.7 * df['x4']
+ np.random.normal(size=200)
```

Example: Python vs R vs Stata

```python
# Define the independent variables (X) and dependent variable (y)
X, y = df[['x1', 'x2', 'x3', 'x4']], df['y']

# Create a LinearRegression model object and fit the model to the data
model = LinearRegression()
model.fit(X, y)

# Print the model coefficients and intercept
print("Coefficients:", model.coef_)
print("Intercept:", model.intercept_)
```

Example: Python vs R vs Stata

```r
set.seed(12345)

# Create a data frame with 200 observations
df <- data.frame(
  x1 = rnorm(200),
  x2 = rnorm(200),
  x3 = rnorm(200),
  x4 = rnorm(200)
)

# Generate the dependent variable y
df$y <- 1.5 + 0.8 * df$x1 - 0.7 * df$x2 + 1.1 * df$x3 - 1.7 * df$x4 + rnorm(200)

# Run the regression
model <- lm(y ~ x1 + x2 + x3 + x4, data = df)

# Print the summary of the regression results
summary(model)
```

Example: Python vs R vs Stata

```
clear
set seed 12345
set obs 200
forv i=1/4 {
    gen x`i'=rnormal()
}
gen y=1.5+0.8*x1-0.7*x2+1.1*x3-1.7*x4+rnormal()

reg y x*
```

Jupyter notebooks and RStudio

Further resources

Python

Python for Data Analysis by Wes McKinney: Written by the creator of the pandas library, this book is great for understanding how to work with large datasets.

Python Data Science Handbook by Jake VanderPlas: This comprehensive book provides a guide to essential Python data science tools like NumPy, pandas, and Matplotlib.

Data Science from Scratch by Joel Grus: This book covers the fundamental concepts of data science by having you implement the algorithms from scratch using Python.

Learn Pandas by Hernan Rojas: Comprehensive guide to how to use pandas.

R

R for Data Science by Hadley Wickham and Garrett Grolemund: Written by key contributors to the R ecosystem, this book focuses on using the tidyverse for data analysis.

An Introduction to Statistical Learning: With Applications in R by Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani: A classic resource for statistical learning with practical examples in R.

swirl: An R package that teaches you R from within the R console itself, offering an interactive, hands-on learning experience.

RStudio Community: A forum where you can ask questions and get help from other R users.

The R Bloggers website: A great aggregator of blogs from the R community.

Stata

Statalist: The official forum for Stata users to engage in discussions about statistics and Stata.

Stata Blog and Video Tutorials: The official Stata website offers short, focused video tutorials and blog posts on specific features and techniques.

Stata Documentation and Web Resources: The official documentation is comprehensive, and there are web resources that provide step-by-step instructions.