

DS3500

Advanced Programming with Data

John Rachlin
Northeastern University



Northeastern University

Principles of Chess



The Top 35 Chess Principles



Control the Center	Doubled rooks on an open file are very strong
Develop pieces quickly	Bishops are better in open positions, knights are better in closed positions
Knights before bishops	The best way to deal with a flank attack, is with a counter attack in the center
Don't move the same piece twice in the opening	When 2 pawns can capture the same piece, capture towards the center
Don't bring your queen out too early	The king should be activated in the endgame
Castle before move 10	Rooks go behind passed pawns
Connect your rooks	2 connected passed pawns on the 6th rank will beat a rook
Rooks should go to open or half open files	Attack the base of a pawn chain
Knights on the rim are grim	Knights are usually the best piece to use to blockade a pawn
Try to avoid doubled pawns	If your position is cramped, trading pieces can help
Try to avoid isolated pawns	Trade your passive pieces for your opponent's active pieces
Try to avoid backward pawns	When ahead material, trade pieces, not pawns
Don't trade a bishop for a knight without a good reason	When behind material, trade pawns, not pieces
Avoid moving pawns in front of your castled king	Games with opposite colored bishops are dangerous in the middlegame, and drawish in the endgame
Don't open the center if your king is still there	Don't play "hope" chess
2 minor pieces are usually better than a rook and a pawn	When you see a good move, look for a better move
3 minor pieces are usually better than a queen	A really good chess player knows the right time to ignore a chess principle
Rooks are good on the 7th rank	



DS 2000: Programming with Data

What is DS 2000 about?

DS2000: Programming with Data is a first-course introduction to computer programming using the Python programming language. The course will introduce using various Python toolkits for manipulating and analyzing data in a broad range of applications including science, finance, and healthcare. The course assumes no prior programming experience.

Who should take DS 2000?

DS 2000 is aimed at students majoring in subjects outside of Computer Science or Data Science who wish to use computer programming as an essential *tool* for working with data in their respective fields. The course teaches students how to work with data algorithmically. *DS 2000 is the one computer programming course to take if you are only going to take one!*

Take DS 2000 if:

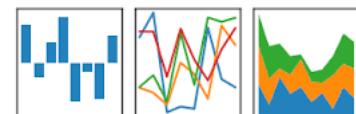
You are a DS major aiming to become a professional data scientists or you are a non-CS / non-DS major looking to gain some practical experience with computer programming as a tool for solving problems in various technical fields. Students with some prior programming skills should consider DS 2500 instead.

Alternatively, take CS 2500 if:

You are a CS major aiming to become a professional computer programmer or computer scientist. CS minors interested in the foundations of Computer Science and hoping to eventually tackle larger-scale software projects will also find CS 2500 to be a better fit.



pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of
Computer Sciences

DS 2500: Intermediate Programming with Data

What is DS 2500 about?

DS2500: Intermediate Programming with Data is a follow-on course to *DS 2000: Programming with Data*. It provides students with the opportunity to advance their computer programming skills using the Python programming language. The course will explore more advanced programming methodologies including object-oriented design and test-driven development – techniques that are useful for tackling larger-scale programming projects.

Who should take DS 2500?

DS 2500 is aimed at DS majors, DS minors, and non-CS / non-DS majors who have taken DS 2000 or have equivalent programming experience. *DS 2500 is the course to take if you are interested in taking your programming skills to the next level and your focus is practical problem solving, data analysis, and discovery in areas such as business, the sciences, and healthcare.*

Take DS 2500 if:

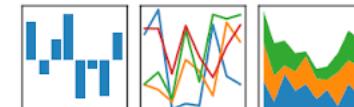
You are a DS major or DS minor, or you are a non-CS / non-DS major looking to gain greater practical experience with computer programming and data-centric problem solving. DS 2500 requires DS 2000 or equivalent programming experience.

Alternatively, take CS 2510 if:

You are a CS major or a DS major interested in the fundamentals of computer science, data structures, and algorithms. CS minors interested in the foundations of Computer Science will also find CS 2500 to be a better fit.



pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of
Computer Sciences

DS 3500: Advanced Programming with Data

What is DS 3500 about?

DS3500: Advanced Programming with Data is a follow-on course to *DS 2500: Programming with Data*. It presents a deep-dive into the design and implementation of enterprise-grade software systems with an emphasis on software architectures for more complex data-driven applications. Topics will include tools and techniques for constructing service-oriented, scalable systems for data analytics, and event-based data processing.

Who should take DS 3500?

DS 3500 is aimed at DS majors, DS minors, and non-CS / non-DS majors who have taken DS 2500 or have equivalent programming experience. *DS 3500 is the course to take if you want to become rock solid in programming for data science.*

Take DS 3500 if:

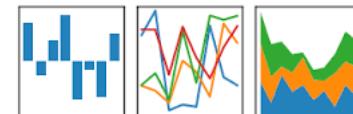
You are a DS major or DS minor, or you are a non-CS / non-DS major looking to achieve a solid foundation in computer programming and data-centric problem solving. DS 3500 requires DS 2500 or equivalent programming experience.

Alternatively, take CS 3500 if:

You are a CS major or a DS major interested in the fundamentals of computer science, data structures, and algorithms. CS minors interested in the foundations of Computer Science will also find CS 3500 to be a better fit.



pandas
 $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



Northeastern University
Khoury College of
Computer Sciences

The *Programming with Data* Track

- Northeastern's Data Science curriculum includes three courses in Python programming:
 - DS 2000: Programming with data
 - DS 2500: Intermediate programming with data
 - DS 3500: Advanced programming with data
- All three courses leverage the python software development ecosystem.
- The three-course programming track follows a logical progression from learning first principles of software development in DS2000 (with no programming experience required) to building object-oriented libraries and data applications in DS2500, to developing more complex *enterprise-grade systems* for data engineering and data science in DS3500.
- The DS Programming with Data track provides an alternative to the traditional CS-oriented *fundamentals* sequence (Fundamentals 1 and 2 and Object-Oriented Design) with special emphasis on data-centric problem solving, analytics, and engineering.



Northeastern University
Khoury College of
Computer Sciences

DS3500 Topics

Tooling

- Managing environments
- Anaconda library management
- The *git* software control system
- The JetBrains PyCharm IDE

Language Features

- Exception handling
- Object-oriented design patterns
- Test-Driven Development
- Functional programming

Modules

- Advanced visualization with *plot.ly*
- Modeling and simulation
- Web-development / User-interfaces with *flask*
- Building simple dashboards
- Working with relational and/or non-relational databases
- Optimization and evolutionary computing
- Working with streaming data
- Time-series analysis
- Multiple regression
- Data structures: Property graphs for Network Analysis
- Machine Learning: decision trees
- Machine Learning: deep learning
- Animation
- Image processing
- Concurrency with *asyncio*
- Other?



DS3500 –Programming with Data

Setting up your Python development environment



Three critical elements

- **Anaconda:** A python distribution and environment manager (or “package manager”) that comes pre-installed with a *base* environment containing hundreds of packages. (Mine currently has 278!)
- An **IDE** (Integrated Development Environment): For creating projects and writing, running, and testing code. We’ll be mostly using **Jetbrains PyCharm** in class but I might sometimes revert to spyder or Jupyter Notebooks. PyCharm readily integrates with Anaconda environments.
- **GIT:** A software control system – for working together on shared development projects. PyCharm integrates with GIT.



Python Distributions

A python *distribution* typically consists of:

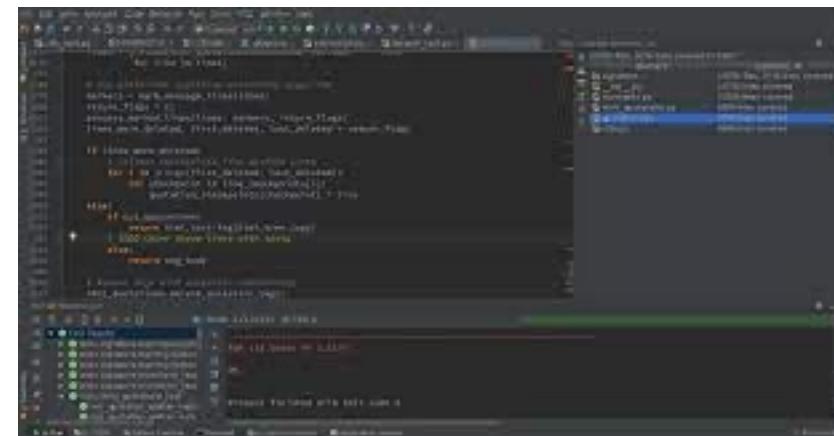
- The python programming language
- Development tools
- Pre-installed python packages (libraries) that make python more powerful
- A package manager (allows you to customize your distribution by adding more libraries or updating existing libraries)



The three pillars of your development environment

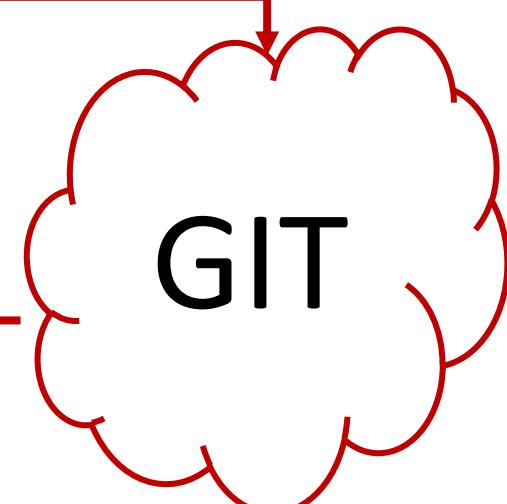


environment
(available packages)



add / commit / push

clone / pull



» PythonDistributions

» PythonDistributions

FRONTPAGE >>

RECENTCHANGES >>

FINDPAGE >>

HELPCONTENTS >>

PYTHONDISTRIBUTIONS >>

Page

» Immutable Page

» Info

» Attachments

» More Actions: ▾

User

» Login

Python Distributions

Aside from the official CPython distribution available from python.org, other distributions based on CPython include the following:

- »  [ActivePython](#) from [ActiveState](#)
- »  [Anaconda](#) from Continuum Analytics
- »  [ChinesePython Project](#): Translation of Python's keywords, internal types and classes into Chinese. Eventually allows a program to run on both English and Chinese operating systems.
- »  [Enthought's Canopy](#)
- »  [Win9xPython](#): Backport of mainline CPython 2.6/2.7 to old versions of Windows 9x/NT.
- » [IPython](#) and its [IPyKit](#) variant
- »  [PocketPython](#)
- »  [Portable Python](#): Run Python from USB device - no installation needed.
- »  [PyIMSL Studio](#)
- »  [PyPy](#): a Python implementation in Python.
- »  [Python\(x,y\)](#): Python(x,y) is a scientific-oriented Python Distribution based on Qt, Eclipse and  [Spyder](#)
- » [PythonForArmLinux](#)
- » [PythonLabsPython](#): an old name for the python.org distribution
- » [PythonwarePython](#)
- » [StacklessPython](#)
- »  [Tiny Python](#) (archived link) - not to be confused with [tinyPy](#)
- »  [WinPython](#): Another scientific-focused Python distribution, based around  [Spyder](#)

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform

[Download](#)

The open-source [Anaconda Distribution](#) is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 15 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with [Conda](#)
- Develop and train machine learning and deep learning models with [scikit-learn](#), [TensorFlow](#), and [Theano](#)
- Analyze data with scalability and performance with [Dask](#), [NumPy](#), [pandas](#), and [Numba](#)
- Visualize results with [Matplotlib](#), [Bokeh](#), [Datashader](#), and [Holoviews](#)



Anaconda installers

Windows

Python 3.7
64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (423 MB)

Python 2.7
64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)

MacOS

Python 3.7
64-Bit Graphical Installer (442 MB)

64-Bit Command Line Installer (430 MB)

Python 2.7
64-Bit Graphical Installer (637 MB)
64-Bit Command Line Installer (409 MB)

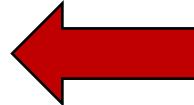
Linux

Python 3.7
64-Bit (x86) Installer (522 MB)

64-Bit (Power8 and Power9) Installer (276 MB)

Python 2.7
64-Bit (x86) Installer (477 MB)
64-Bit (Power8 and Power9) Installer (295 MB)

Make sure you get Python 3.7
You will not need Python 2.7



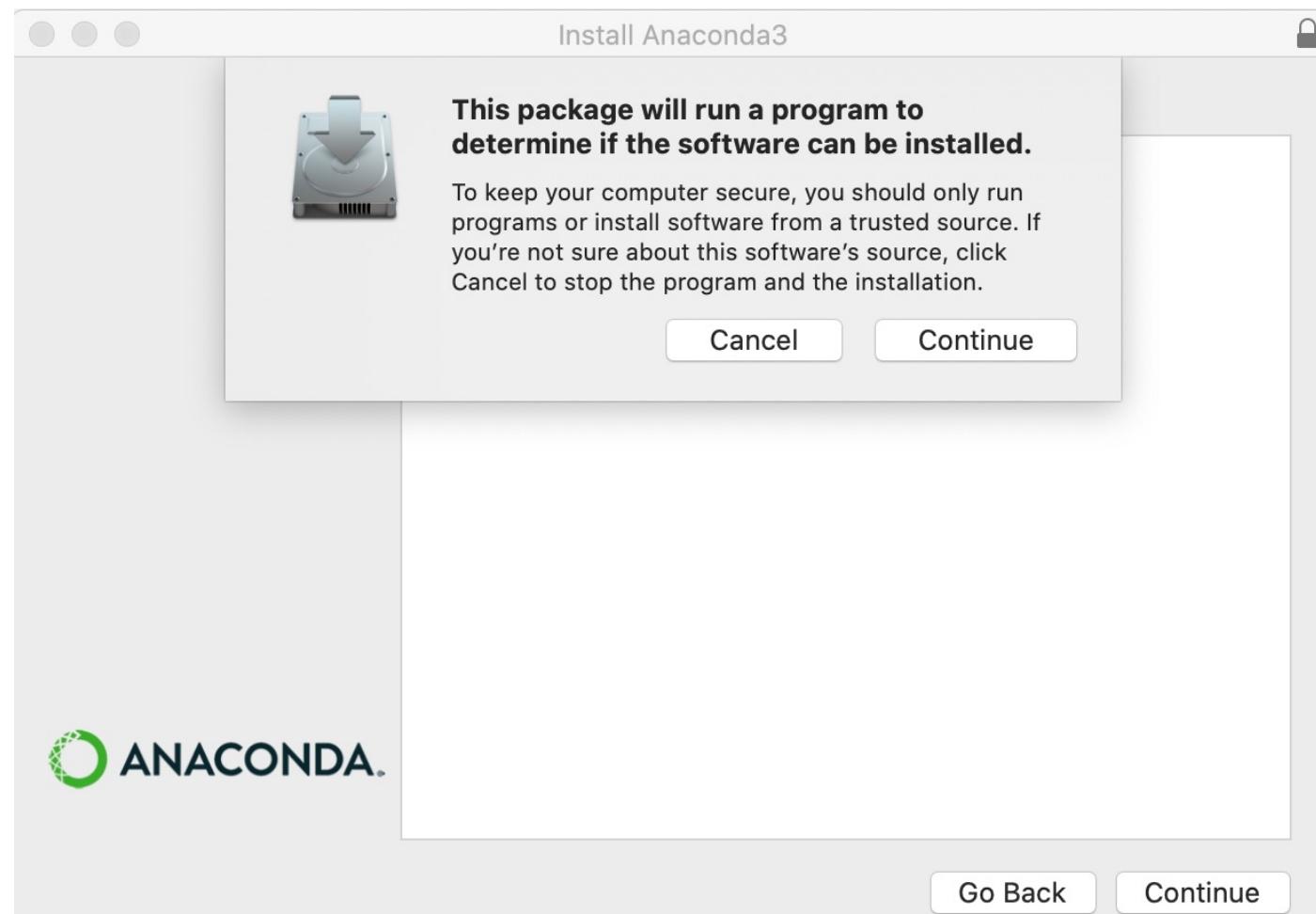
Installing Anaconda on a Mac

Double-click on the downloaded .pkg file.

Mine is called:

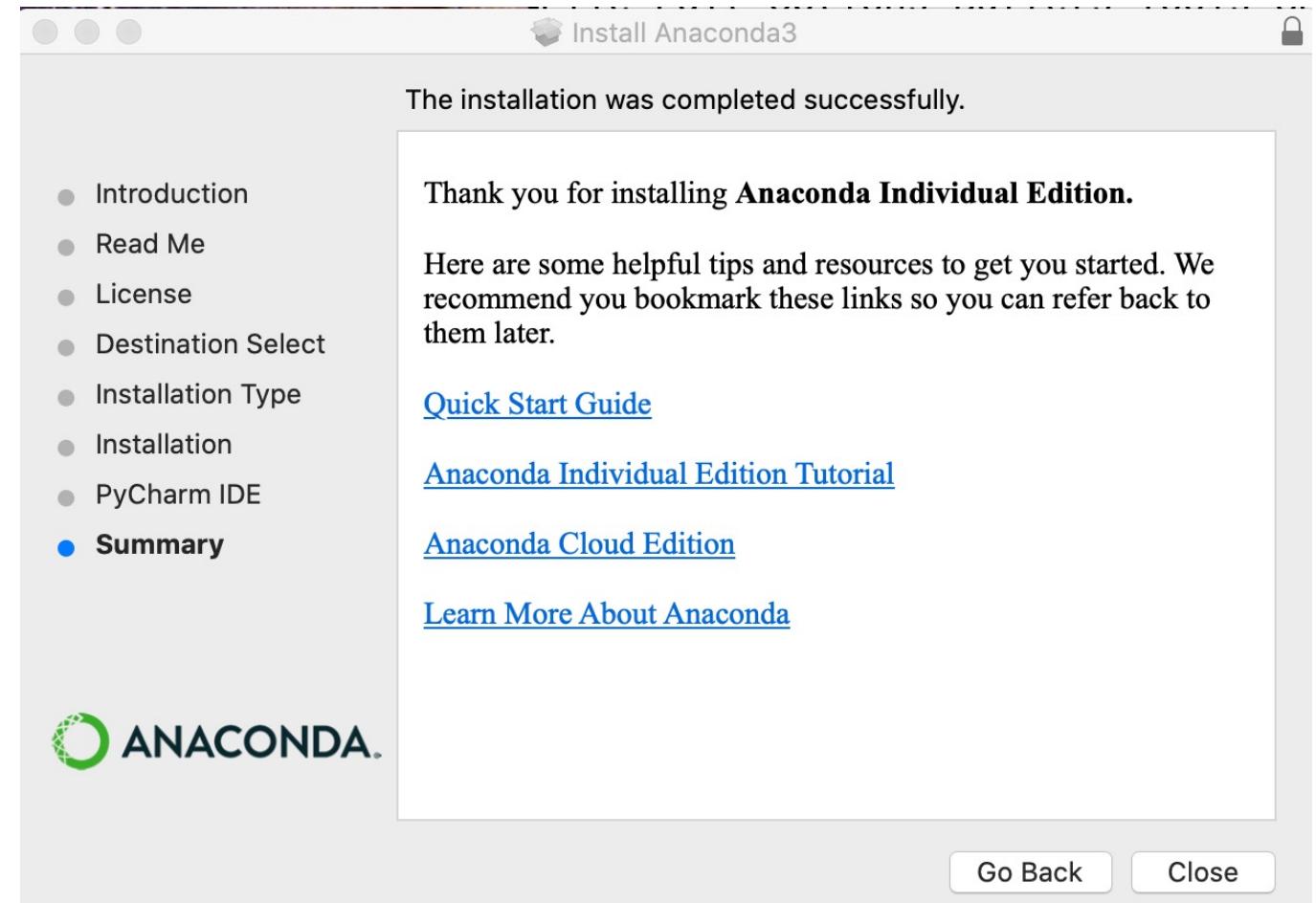
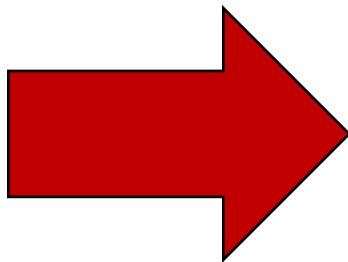
Anaconda3-2020.02-MacOSX-x86_64.pkg

Optional: specify where you want Anaconda to be installed. I installed mine in **/Users/rachlin/apps**



Installing Anaconda on a Mac

You can basically accept the defaults and keep hitting the “continue” button until you get....



Anaconda Navigator: Access to Applications

The screenshot shows the Anaconda Navigator interface. On the left is a sidebar with links for Home, Environments, Learning, Community, Documentation, and Developer Blog. The main area displays a grid of applications under the heading "Applications on base (root) Channels". The applications listed are:

- JupyterLab (1.2.6): An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture. Includes a "Launch" button.
- Jupyter Notebook (6.0.3): Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis. Includes a "Launch" button.
- Qt Console (4.6.0): PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more. Includes a "Launch" button.
- Spyder (4.0.1): Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. Includes a "Launch" button.
- Glueviz (0.15.2): Multidimensional data visualization across files. Explore relationships within and among related datasets. Includes an "Install" button.
- Orange 3 (3.23.1): Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox. Includes an "Install" button.
- RStudio (1.1.456): A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. Includes an "Install" button.

A large red arrow points upwards from the RStudio application towards the Jupyter Notebook application, highlighting it. Below the grid, the text "Especially these!" and "(Can also be invoked from the command line.)" is displayed.



Northeastern University

IPython: An enhanced REPL for Python

```
rachlin — IPython: Users/rachlin — ipython — 80x24
(base) [512 uranus:~] ipython
Python 3.7.6 (default, Jan 8 2020, 13:42:34)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.16.1 -- An enhanced Interactive Python. Type '?' for help.

[In 1]: print('hi from IPython')
hi from IPython

[In 2]: ?print
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type: builtin_function_or_method

[In 3]:
```

- Syntax highlighting
- Code completion
- Documentation



Spyder: A full-fledged Integrated Development Environment (IDE)

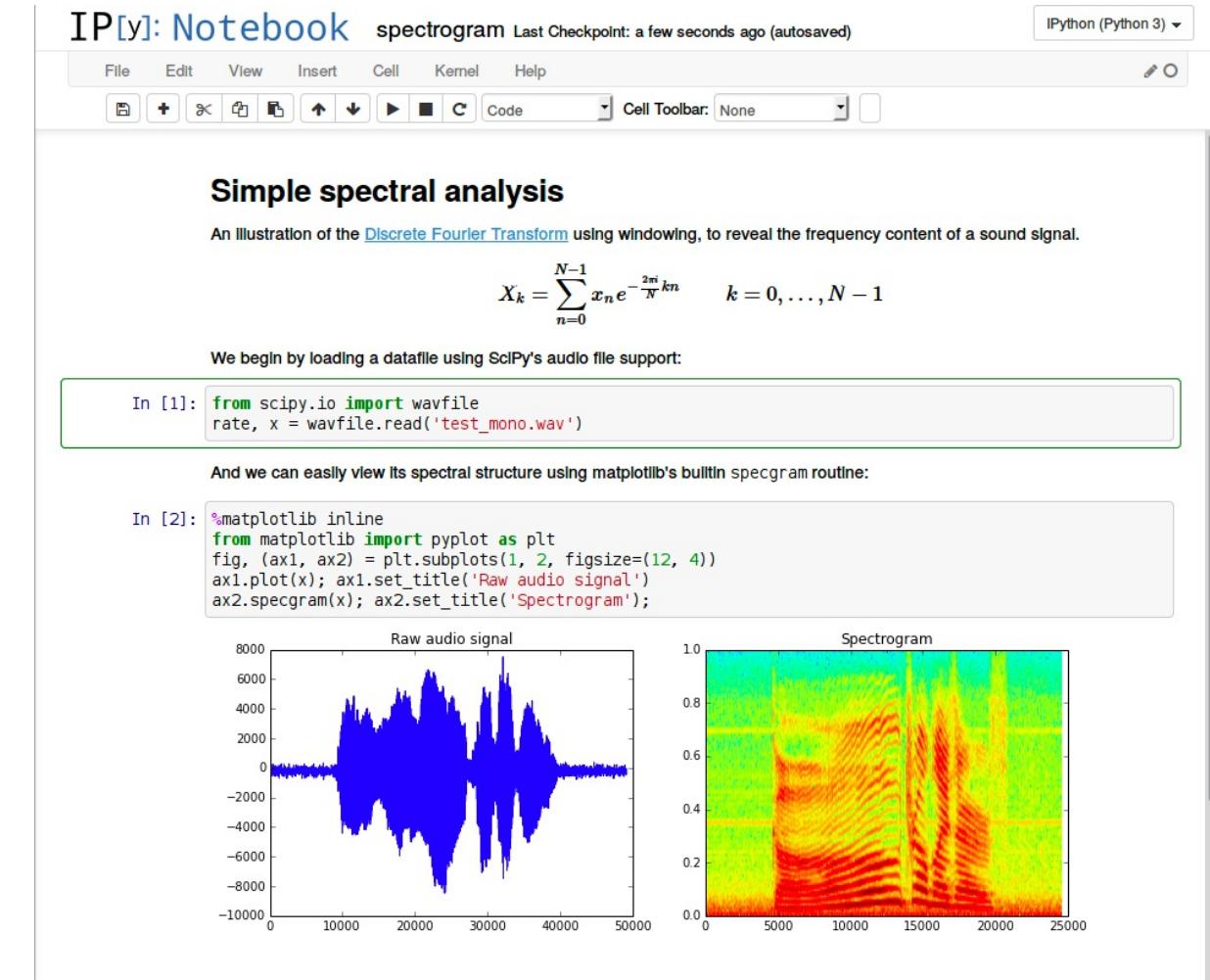
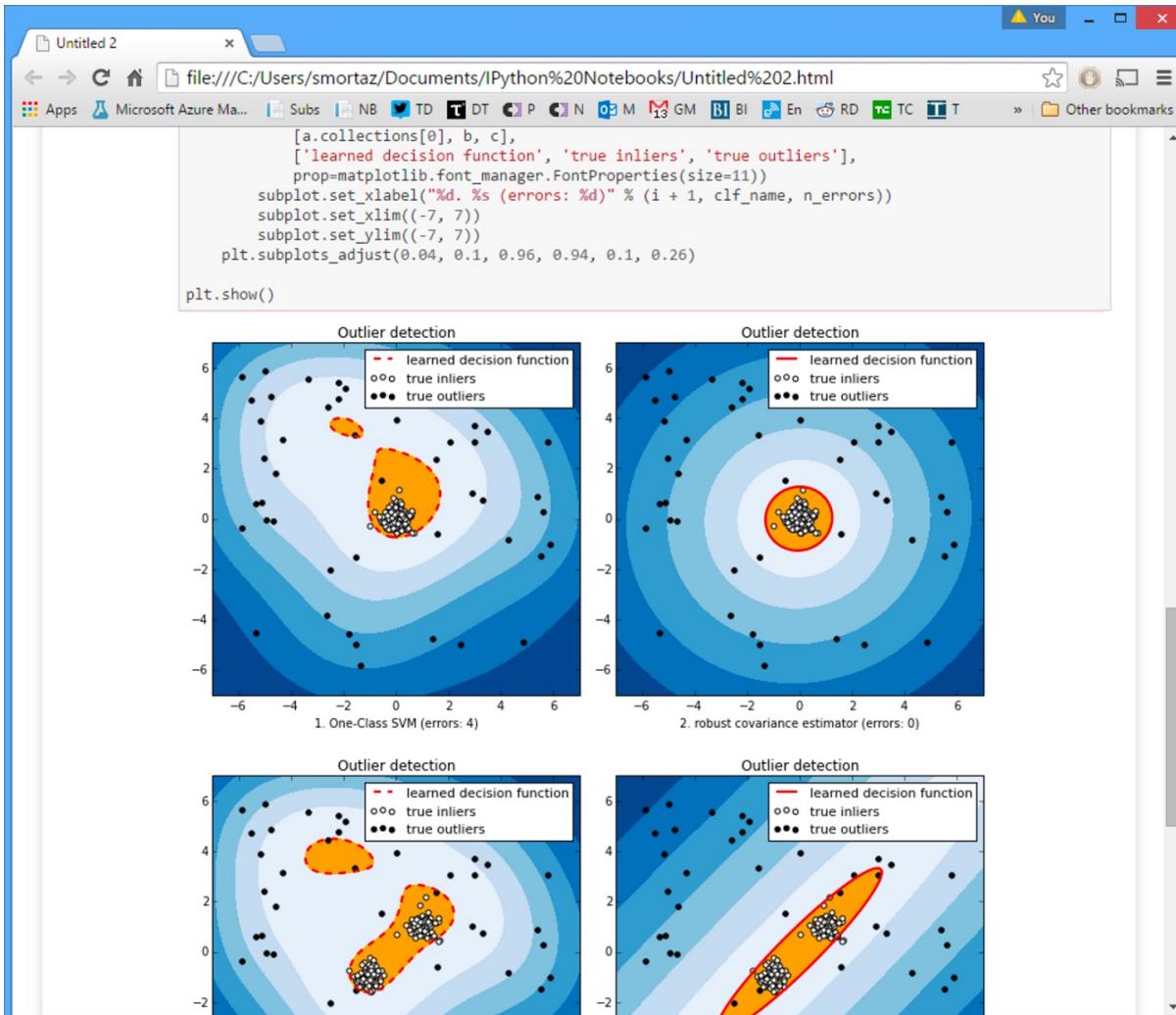
The screenshot displays the Spyder IDE interface, version Python 3.7, with the following components:

- Editor:** Shows the file `02-expressions.py*` containing Python code. The code calculates the volume of a sphere and performs some calculations related to book costs and shipping.
- Variable explorer:** A table showing the values of variables defined in the script:

Name	Type	Size	Value
n	int	1	0
pi	float	1	3.14159
r	int	1	5
volume	float	1	523.598333333332
x	str	1	hello
- IPython console:** Displays the history of commands entered and their results:

```
In [5]:  
In [6]: pi = 3.14159  
...: r = 5  
...: volume = 4/3 * pi * r**3  
...: print('volume of sphere (r=5): ', volume)  
volume of sphere (r=5): 523.598333333332  
In [7]:
```

Jupyter Notebooks: Interactive Python for Data Science



Anaconda Navigator: Access to Learning resources

The screenshot displays the Anaconda Navigator interface, a web-based application for managing and exploring data science tools. The left sidebar includes links for Home, Environments, Learning (selected), Community, Documentation, and Developer Blog, along with social media icons for Twitter, YouTube, and GitHub.

The main content area features a grid of cards representing various learning resources:

		Documentation (21)		Training (1)	Video (0)	Webinar (0)	Search	
Python	Python Tutorial	Python Reference	Anaconda Package List	pandas	Numpy Documentation	Scipy Documentation	Matplotlib Documentation	Bokeh User Guide
	Read	Read	Read	Read	Read	Read	Read	Read
Anaconda	Anaconda Cloud Documentation	Anaconda Documentation	Anaconda Navigator Documentation	Using MRO or R with Conda	Microsoft R Open: The Enhanced Distribution	The Comprehensive R Archive Network (CRAN)	The Python Package Index (PyPI)	Dask documentation
	Read	Read	Read	Read	Read	Read	Read	Read
CONDA	Conda & Conda-Build	Jupyter documentation	Spyder documentation	VSCode (python)	Orange documentation	Python Training by Anaconda		
	Read	Read	Read	Read	Read	Explore		



Anaconda Navigator: Access to Community Forums

The screenshot displays the Anaconda Navigator application window. At the top, there are three colored window control buttons (red, yellow, green) on the left, followed by the title "Anaconda Navigator" in a grey bar. On the right side of the same bar is a green button labeled "Sign in to Anaconda Cloud". Below the title bar is a navigation menu with the following items: Home, Environments, Learning, and a highlighted "Community" section. The "Community" section is further divided into Event (1), Forum (8), and Social (6). A search bar with a magnifying glass icon is located on the far right of the header.

The main content area is a grid of 16 cards, arranged in two rows of eight. Each card represents a different community resource:

- Row 1:**
 - SciPy Conferences (Icon: S logo, "Learn More" button)
 - Anaconda Forum (Icon: Anaconda logo, "Explore" button)
 - Stack Overflow: Python (Icon: orange pyramid, "Explore" button)
 - Conda Forum (Icon: Anaconda logo, "Explore" button)
 - Bokeh Forum (Icon: colorful geometric shapes, "Explore" button)
 - Blaze Dev Forum (Icon: blue cube, "Explore" button)
 - Numba Dev Forum (Icon: circular progress bar, "Explore" button)
 - Matplotlib Forum (Icon: pie chart, "Explore" button)
- Row 2:**
 - NumPy and SciPy Project Mailing Lists (Icon: S logo, "Explore" button)
 - Anaconda on Twitter (Icon: globe with network lines, "Engage" button)
 - Planet Scipy (Icon: books with S logo, "Engage" button)
 - Data Science Central (Icon: atom, "Engage" button)
 - Anaconda Company Blog (Icon: Anaconda logo, "Engage" button)
 - NUMFOCUS OPEN CODE = BETTER SCIENCE (Icon: red and blue text, "Engage" button)
 - Anaconda Developer Blog (Icon: Anaconda logo, "Engage" button)

On the far left of the main content area, there is a vertical sidebar with two sections: "Documentation" and "Developer Blog".

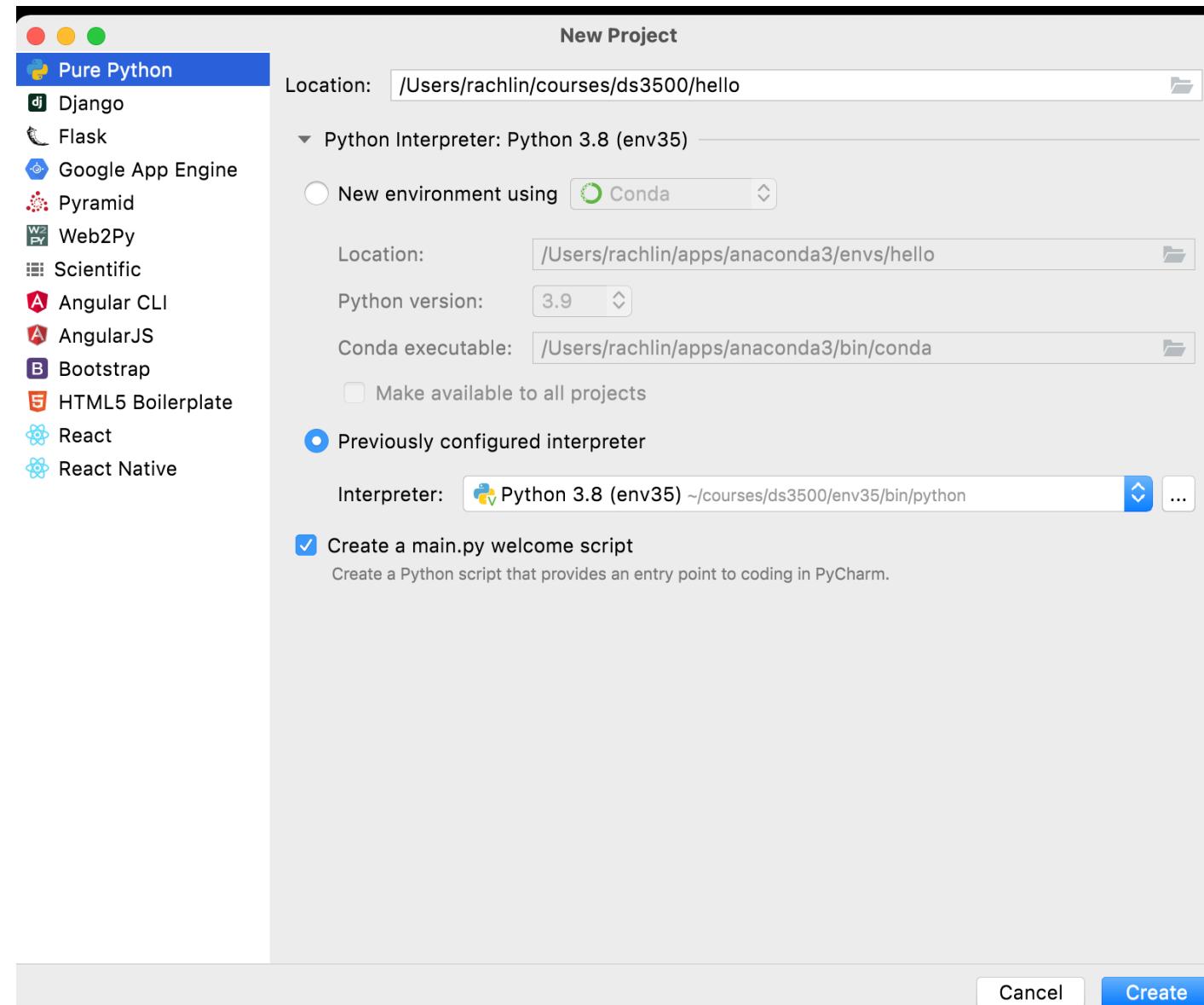


PyCharm (JetBrains): Another popular Python IDE

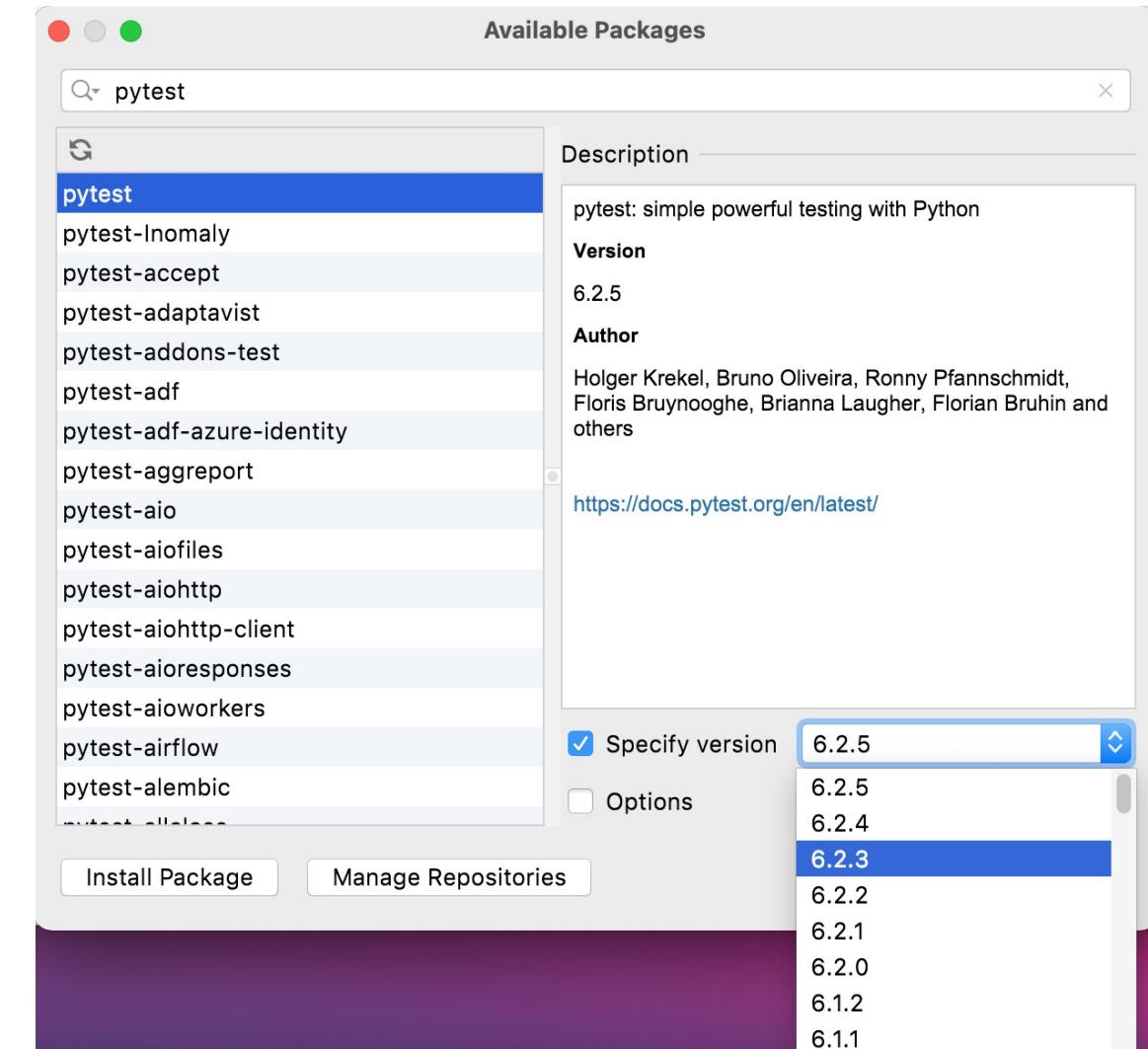
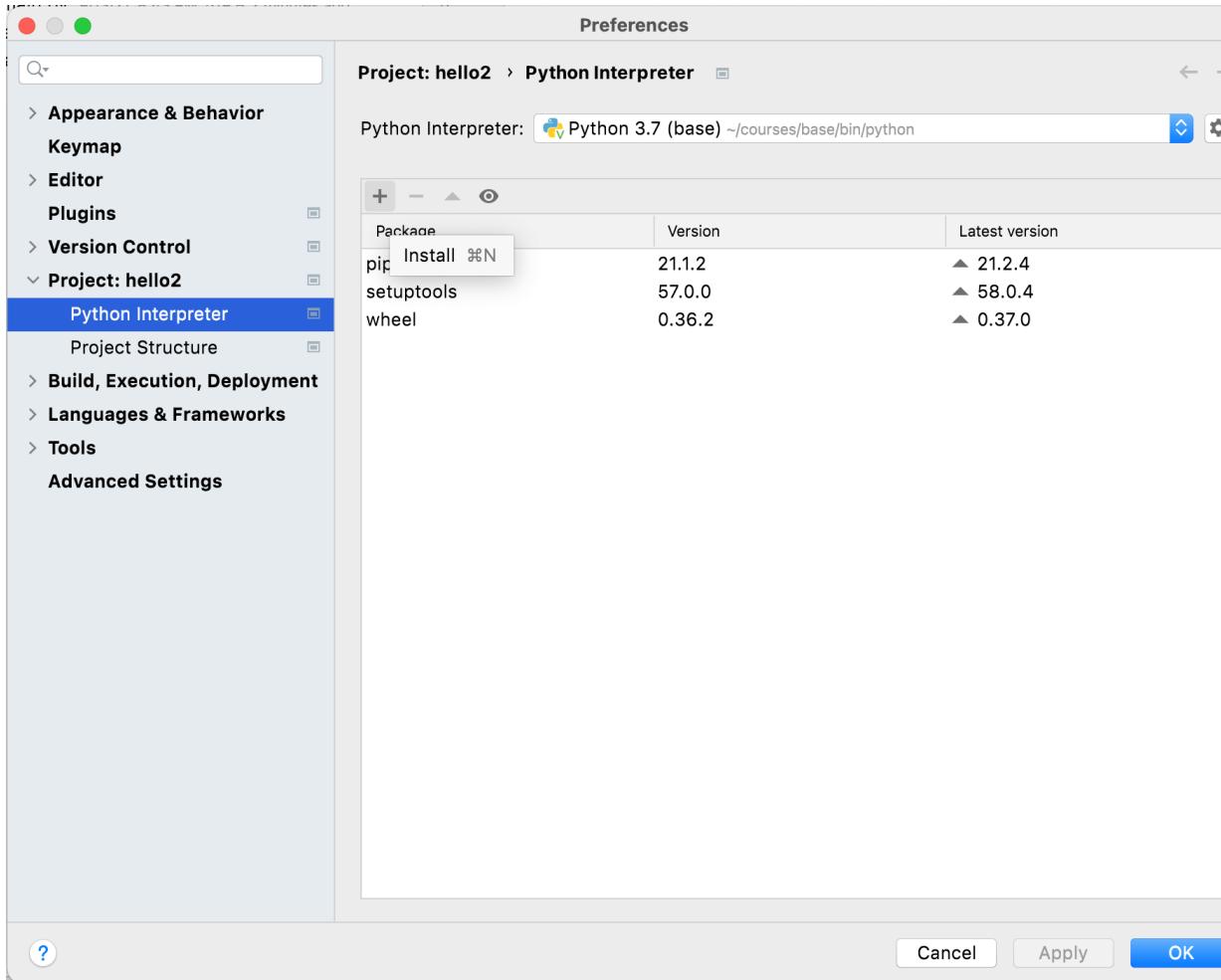
The screenshot shows the PyCharm IDE interface with the following components:

- Project View:** On the left, it shows a tree view of the project structure under the 'think' directory. Files like 11-dictionaries.py, 01-wayoftheprgram.py, and 02-expressions.py are listed.
- Code Editor:** The main area displays the content of 11-dictionaries.py. The code demonstrates various dictionary operations, including creating simple dictionaries, handling KeyError, checking length, and using values(). A section titled '#%% Gotchas' highlights common pitfalls like implied ordering and key containment.
- Python Console:** Below the code editor, the Python Console tab is active, showing the output of running the script. It includes the path to the script, the version of Python used (3.7.3), and the starting message 'PyDev console: starting.'
- Toolbars and Status Bar:** The top bar includes standard icons for file operations and Git integration. The status bar at the bottom shows the current time (13:12), encoding (UTF-8), and file type (Python 3.7).
- Right-hand Panels:** There are several panels on the right:
 - Special Variables:** A list of built-in variables like __file__, __name__, __builtins__, etc.
 - Project Configurations:** A message box indicates that configuration files can be added to Git, with options to 'View Files', 'Always Add', or 'Don't Ask Again'.
 - Event Log:** A small panel at the bottom right showing one event.

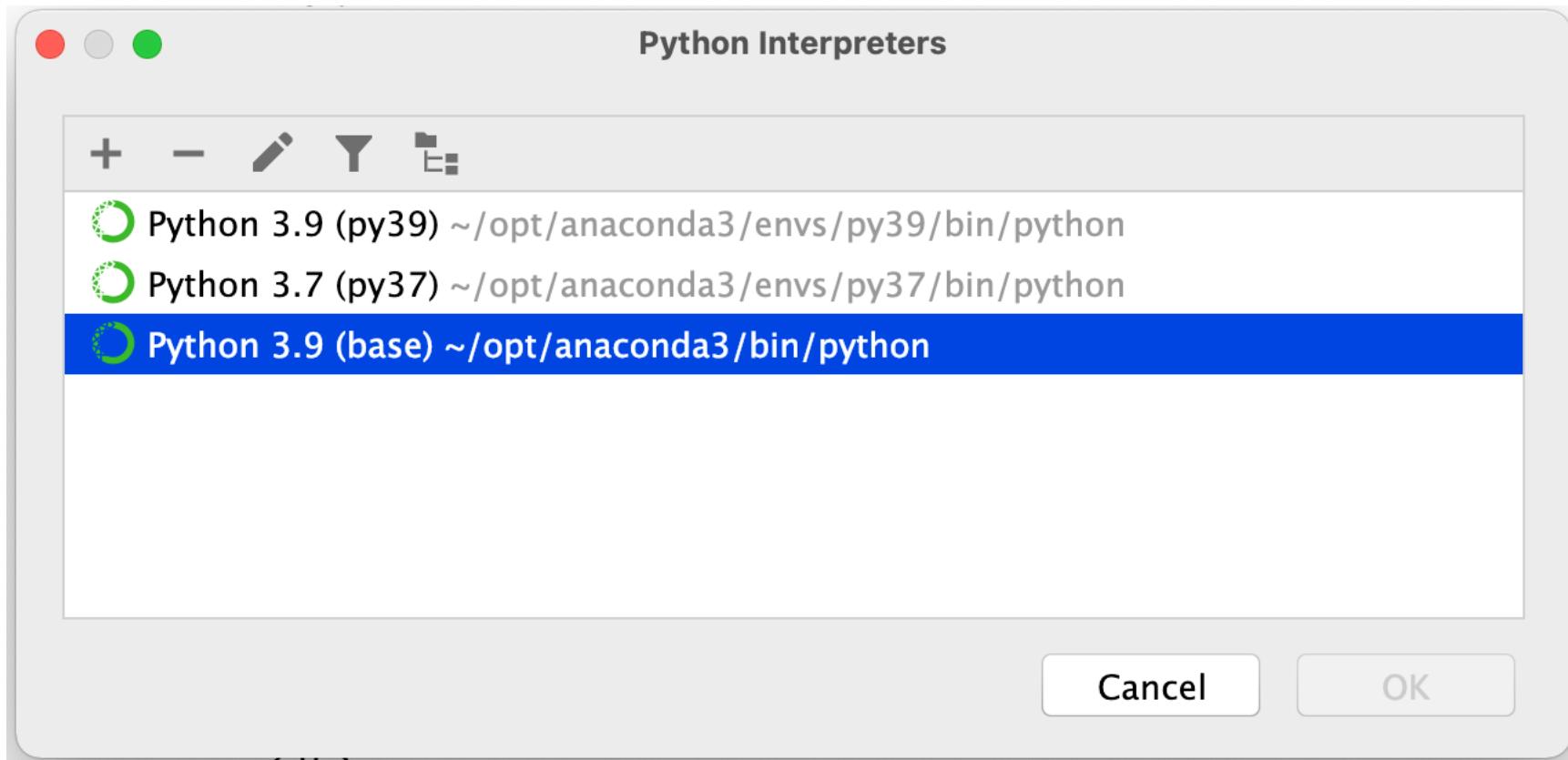
Connecting PyCharm project to an environment



Or we could create a brand-new environment



Or we can select a conda environment



PyCharm Plots

The screenshot shows the PyCharm IDE interface with the following components:

- Project View:** Shows a project named "cherry" containing a file "cherry.csv" and a script "cherry.py".
- Code Editor:** The "cherry.py" file is open, displaying Python code to read CSV data, calculate moving averages, and plot cherry blossom history.
- SciView:** A plot titled "Cherry Blossom full blossom date (Kyoto, Japan)" showing the date of full bloom from approximately 800 to 2021. The plot includes data points for individual years (pink dots) and a 30-year moving average (blue line). Annotations indicate specific years (1323, 1409, 2021) and a historical period ("The Little Ice Age").
- Python Console:** Displays the output of running the script, including the creation of a DataFrame and the command to read the CSV file.
- Status Bar:** Shows the Python interpreter is set to "Python 3.8 (base)", the file size is 596 of 2048M, and the current time is 48:5 LF.



Viewing data as a DataFrame

The screenshot shows a Jupyter Notebook interface with the following components:

- Project View:** Shows a project named "cherry" containing a file "cherry.py" and a CSV file "cherry.csv".
- Code Editor:** Displays the content of "cherry.py". The code reads a CSV file, computes moving averages, and plots the data.
- Data View:** A SciView panel titled "cherry*" displays the data from "cherry.csv" as a DataFrame. The columns are labeled "YEAR" and "FLOWERING_DOY".
- Python Console:** Shows the execution of code to read the CSV and the resulting DataFrame structure.
- Object Inspector:** A context menu for the variable "cherry" is open, with "View as DataFrame" highlighted.
- Bottom Status Bar:** Shows the Python interpreter configuration and system status.

YEAR	FLOWERING_DOY	
1216	2021	85.00000
608	1409	86.00000
435	1236	87.00000
160	961	87.00000
811	1612	87.00000
845	1646	87.00000
567	1368	88.00000
1189	1990	88.00000
154	955	89.00000
960	1761	89.00000
355	1156	90.00000
1215	2020	90.00000
980	1781	91.00000
612	1413	91.00000
1197	1998	91.00000



Python Development Tools - Summary

Tool / Platform	Strengths and Weaknesses
REPL	Great for testing small snippets of Python and for learning Python syntax. Not intended for application development. IPython is an enhanced REPL for Python that comes packaged with the Anaconda distribution.
IDLE	Start coding today! Comes with Python distribution. Fine for small, single-file, stand-alone scripts or applications. Not recommended for more complex projects.
Atom text editor	An extendible text-editor with many plugins for Python programming. See Prof. Rachlin's handout on recommended plugins.
Spyder	A full-fledged IDE installed with the Anaconda distributions. Manage multiple files. Monitor variables. Run isolated blocks of code.
PyCharm (JetBrains)	Another popular IDE for python. Has more functionality than what you will need for this class. I won't be making use of PyCharm in this class, but you are welcome to explore its capabilities and use it for your homework.
Jupyter notebooks	A <i>notebook-style</i> development environment popular with data scientists. Integrate code, text, tables, visualizations in one file. Not suited for software <i>applications</i> requiring multiple python files.
Anaconda	A Python distribution that comes with Python, popular pre-installed libraries, and a variety of development tools including Spyder, Ipython, and Jupyter notebooks. The Anaconda Navigator is a jumping-off point for applications, learning resources, and community forums. Recommended.

Git commits

H₁ Git + GitHub lecture

H₂ Git

H₃ Basics

1. `git add --all`
2. `git commit -m "Fix ..."`

COMMENT	DATE
O CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
O ENABLED CONFIG FILE PARSING	9 HOURS AGO
O MISC BUGFIXES	5 HOURS AGO
O CODE ADDITIONS/EDITS	4 HOURS AGO
O MORE CODE	4 HOURS AGO
O HERE HAVE CODE ↗	4 HOURS AGO
O AAAAAAAA	3 HOURS AGO
O ADKFJSLKDFJSOKLFJ	3 HOURS AGO
O MY HANDS ARE TYPING WORDS	2 HOURS AGO
O HAAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

[How to Write a Git Commit Message] (🔗)

If applied, this commit will *~*your subject line here*~*

3. `git pull`
4. `git push`



Create git repo: https://github.ccs.neu.edu/rachlin/ds3500_sp22

The screenshot shows a GitHub repository page for 'rachlin/ds3500_sp22'. The page includes a header with navigation links like Home, Misc, CS, Research, NEU, and Teaching. Below the header is a search bar and a navigation bar with links for Pull requests, Issues, Explore, Unwatch (1), Star (0), and Fork (0). The main content area displays the repository's code, showing four commits from user 'rachlin' with commit messages: 'Create README', 'Create README', 'Initial commit', and 'Initial commit'. The repository has 1 branch and 0 tags. On the right side, there is an 'About' section describing it as 'The class repo for DS3500 (Spring 2022)' with a 'Readme' link, and a 'Releases' section indicating 'No releases published' with a 'Create a new release' link.

github.ccs.neu.edu/rachlin/ds3500_sp22

Home Misc CS Research NEU Teaching

Enterprise Search or jump to... Pull requests Issues Explore

rachlin / ds3500_sp22 Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

rachlin Create README b069d40 1 hour ago 3 commits

misc Create README 1 hour ago

.gitignore Initial commit 1 hour ago

README.md Initial commit 1 hour ago

README.md

ds3500_sp22

The class repo for DS3500 (Spring 2022)

About

The class repo for DS3500 (Spring 2022)

Readme

Releases

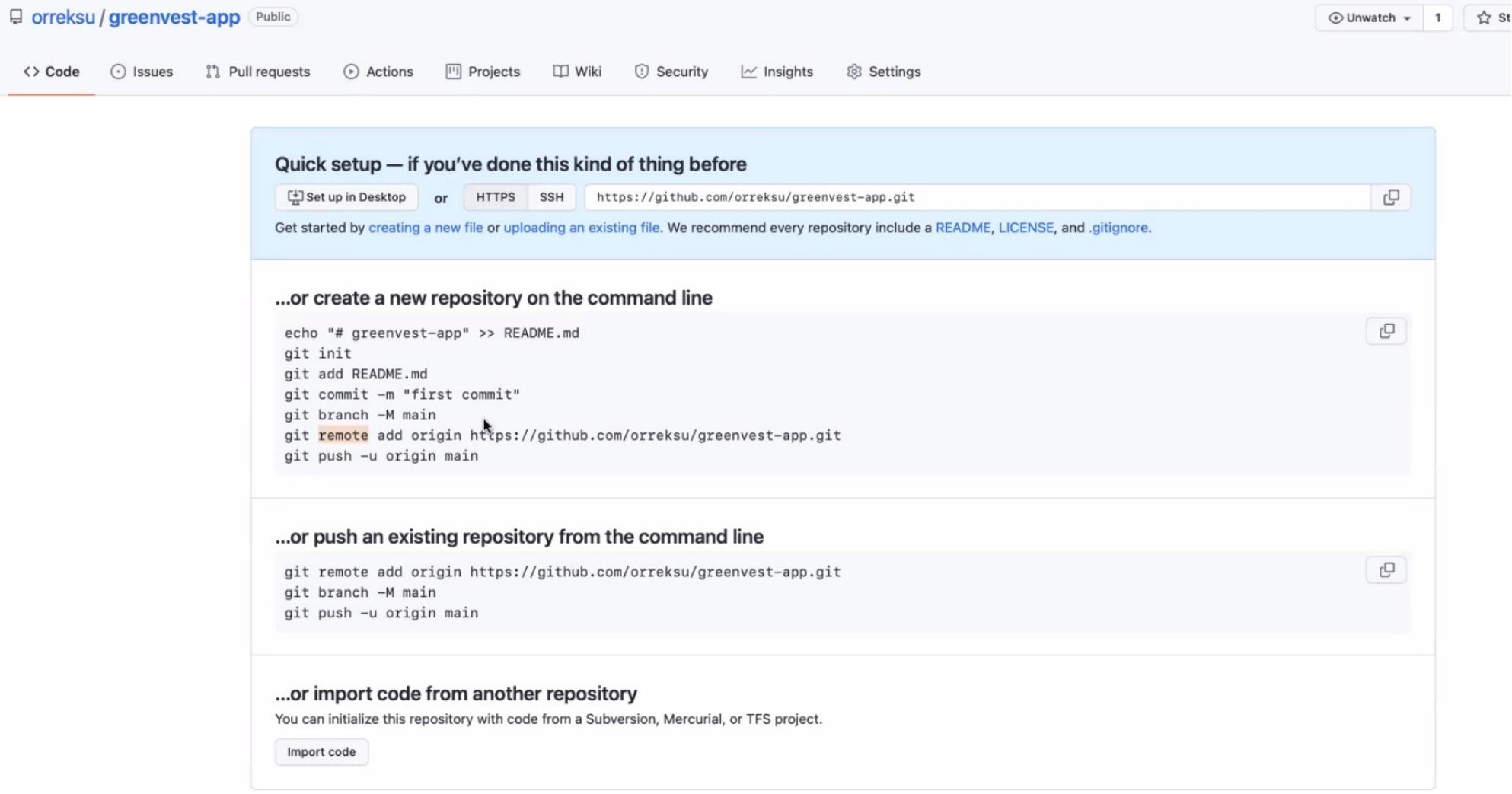
No releases published

Create a new release

https://github.ccs.neu.edu/rachlin/ds3500_sp22



Setting up Git Repositories



The screenshot shows a GitHub repository page for 'orreksu/greenvest-app'. The top navigation bar includes links for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. A 'Quick setup' section provides options to 'Set up in Desktop' or use 'HTTPS' or 'SSH' with the URL <https://github.com/orreksu/greenvest-app.git>. It also suggests creating a new file or uploading an existing one, and recommends including a README, LICENSE, and .gitignore. Below this, sections for '...or create a new repository on the command line' and '...or push an existing repository from the command line' provide git commands. The '...or import code from another repository' section allows initializing the repository with code from Subversion, Mercurial, or TFS. A 'ProTip!' at the bottom suggests using the URL for adding GitHub as a remote.

orreksu / greenvest-app Public

Unwatch 1 Star

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or HTTPS SSH https://github.com/orreksu/greenvest-app.git

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a `README`, `LICENSE`, and `.gitignore`.

...or create a new repository on the command line

```
echo "# greenvest-app" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/orreksu/greenvest-app.git  
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/orreksu/greenvest-app.git  
git branch -M main  
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

ProTip! Use the URL for this page when adding GitHub as a remote.



Viewing Git Repositories

The screenshot shows a GitHub profile page for the user 'orreksu'. The top navigation bar includes 'Search or jump to...', 'Pull requests', 'Issues', 'Marketplace', 'Explore', and a '+' button. The left sidebar lists 'Repositories' (with a search bar), including 'orreksu / orreksu.github.io', 'orreksu / python-exercises', 'racket / racket', 'orreksu / informatics-2013', 'orreksu / nush', 'orreksu / greenvest', and 'David-OConnor / pyflow'. It also shows 'Recent activity' with a link to '[Feature Request] in-json ...'.

The main content area displays 'All activity' with the following items:

- NatTuck created a repository NatTuck/local-time-elixir 6 days ago**
Sometime your app runs in only one time zone.
Updated Oct 2 [Star](#)
- GracefulLemming forked GracefulLemming/dailp-encoding from NEU-DSG/dailp-encoding 7 days ago**
Digital Archive of American Indian Languages Preservation and Perseverance
Rust ⭐ 6 Updated Oct 6 [Star](#)
- GracefulLemming starred rhasspy/larynx 10 days ago**
End to end text to speech system using gruut and onnx
Python ⭐ 304 Updated Aug 29 [Star](#)
- vmthread started following you 19 days ago**
Skal jeg munk dig i skideren
11 repositories 748 followers [Follow](#)
- GracefulLemming starred masarakki/nyaruko_lang 21 days ago**
いつもニコニコあなたの隣に這いよる混沌ニャルラトホテブ言語ですっ
Ruby ⭐ 144 Updated May 12 [Star](#)
- GracefulLemming starred lilyball/nix-env.fish 28 days ago**
lilyball/nix-env.fish [Star](#)

A 'Save the Date!' modal is open, announcing the GitHub Universe event on October 27 and 28, with a 'Learn more' button. To the right, there's an 'Explore repositories' section featuring 'zio/interop-cats' (Scala ⭐ 105), 'racket/typed-racket' (Racket ⭐ 396), and 'cjolowicz/cookiecutter-hypermodern-python' (Python ⭐ 364), along with a 'Explore more →' link.



Your new repo!

oreksu/greenvest-app Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags Go to file Add file Code

oreksu Introduce basic tests for app 8d001f7 4 minutes ago 2 commits

assets	Initial commit	6 minutes ago
data	Initial commit	6 minutes ago
.gitignore	Initial commit	6 minutes ago
Procfile	Initial commit	6 minutes ago
app.py	Initial commit	6 minutes ago
requirements.txt	Initial commit	6 minutes ago
test_app.py	Introduce basic tests for app	4 minutes ago

Help people interested in this repository understand your project by adding a README. Add a README

About Invest in green

Releases No releases published Create a new release

Packages No packages published Publish your first package

Languages Python 100.0%

© 2021 GitHub, Inc.

Terms

Privacy

Security

Status

Docs



Contact GitHub

Pricing

API

Training

Blog

About



Northeastern University

Cloning

```
~/duc-home
> ls

~/duc-home
> git clone https://github.com/orreksu/greenvest-app.git
Cloning into 'greenvest-app'...
remote: Enumerating objects: 22, done.
remote: Counting objects: 100% (22/22), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 22 (delta 1), reused 22 (delta 1), pack-reused 0
Receiving objects: 100% (22/22), 2.60 MiB | 14.60 MiB/s, done.
Resolving deltas: 100% (1/1), done.
```

```
~/duc-home
>
```



Failed commits / Auto-merging

```
greenvest_app on ✘ main
> ls
.git assets data .gitignore app.py Procfile requirements.txt test_app.py
greenvest_app on ✘ main
> code .

greenvest_app on ✘ main
> git add --all

greenvest_app on ✘ main [+]
> git commit -m "Increase size of the logo"
[main 899ded4] Increase size of the logo
 1 file changed, 1 insertion(+), 1 deletion(-)

greenvest_app on ✘ main [↑]
> git push
To https://github.com/orreksu/greenvest-app.git
 ! [rejected]      main -> main (fetch first)
error: failed to push some refs to 'https://github.com/orreksu/greenvest-app.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

greenvest_app on ✘ main [↑]
>
```

Github has changes
that I don't yet have.



When automerge fails....

e.g., two developers changing the same line of code.

```
greenvest_app on ✚ main [↑]
> git push
To https://github.com/orreksu/greenvest-app.git
! [rejected]      main -> main (fetch first)
error: failed to push some refs to 'https://github.com/orreksu/greenvest-app.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

greenvest_app on ✚ main [↑]
> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 2), reused 3 (delta 2), pack-reused 0
Unpacking objects: 100% (3/3), 309 bytes | 103.00 KiB/s, done.
From https://github.com/orreksu/greenvest-app
  839aec8..e40affa  main      -> origin/main
Auto-merging app.py
CONFLICT (content): Merge conflict in app.py
Automatic merge failed; fix conflicts and then commit the result.
greenvest_app on ✚ main (MERGING) [=]
>
```



```
11     html.Div([
2       |       html.H2('GreenVest', style={'color': '#1A9968', 'font-weight': 'bold'}),
3       |       Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session
4       |       <<<<< HEAD (Current Change)
5       |       ~~~~                                           html.H5('Invest in future', style={"marginTop": "-10px"})
6       |
7       |       =====
8       |       html.H5('Be Green!', style={"marginTop": "-10px"})
9       |       >>>>> e40affa4f48ca8a653676e18a295662fd2d4f491 (Incoming Change)
10      |       ], className="two columns"),
```



CodeTriage: Tracking projects with lots of issues

Open source projects on GitHub that need your help.

Filter list by Language: Python 

sympy  2067 ISSUES A computer algebra system written in pure Python (sympy/sympy)	youtube-dl  2044 ISSUES Command-line program to download videos from YouTube.com and other video sites (rg3/youtube-dl)	odoo  1906 ISSUES Odoo. Open Source Apps To Grow Your Business. (odoo/odoo)	pandas  1767 ISSUES Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.frame objects, statistical functions, and much more (pydata/pandas)	pandas  1669 ISSUES Flexible and powerful data analysis / manipulation library for Python, providing labeled data structures similar to R data.frame objects, statistical functions, and much more (pandas-dev/pandas)
erpnext  1571 ISSUES Open Source ERP built for the web (frappe/erpnext)	salt  1553 ISSUES Software to automate the management and configuration of any infrastructure or application at scale. Get access to the Salt software package repository here: (saltstack/salt)	youtube-dl  1533 ISSUES Command-line program to download videos from YouTube.com and other video sites (ytdd-org/youtube-dl)	scikit-learn  1383 ISSUES scikit-learn: machine learning in Python (scikit-learn/scikit-learn)	nvda  1316 ISSUES NVDA, the free and open source Screen Reader for Microsoft Windows (nvaccess/nvda)
zulip  1211 ISSUES Zulip server - powerful open source team chat (zulip/zulip)	conda  1112 ISSUES OS-agnostic, system-level binary package manager and ecosystem (conda/conda)	statsmodels  1111 ISSUES Statsmodels: statistical modeling and econometrics in Python (statsmodels/statsmodels)	scipy  1097 ISSUES Scipy library main repository (scipy/scipy)	mypy  1083 ISSUES Optional static typing for Python 2 and 3 (PEP484) (python/mypy)



Continuous Integration / Continuous Deployment

H3 Actions

H4 CI vs CD

Deliver fast in small chunk to incrementally improve product.

Commit often, test every time.

CI (Continuous Integration) helps to build, test, and merge branches.

CD (Continuous Deployment) helps automate the process of deployment to the server.

H4 Simple Actions

```
```yaml
This is a basic workflow to help you get started with Actions

name: CI

Controls when the workflow will run

on:
 # Triggers the workflow on push or pull request events but only for
 # the main branch
 push:
 branches: [main]
 pull_request:
 branches: [main]

Allows you to run this workflow manually from the Actions tab
workflow_dispatch:
```



# Workflows: “On push, run the following jobs”

orreksu/greenvest-app Public

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

greenvest-app/.github/workflows/ blank.yml in main Cancel changes Start commit ▾

<> Edit new file Preview Spaces 2 No wrap

```
1 # This is a basic workflow to help you get started with Actions
2
3 name: CI
4
5 # Controls when the workflow will run
6 on:
7 # Triggers the workflow on push or pull request events but only for the main branch
8 push:
9 branches: [main]
10 pull_request:
11 branches: [main]
12
13 # Allows you to run this workflow manually from the Actions tab
14 workflow_dispatch:
15
16 # A workflow run is made up of one or more jobs that can run sequentially or in parallel
17 jobs:
18 # This workflow contains a single job called "build"
19 build:
20 # The type of runner that the job will run on
21 runs-on: ubuntu-latest
22
23 # Steps represent a sequence of tasks that will be executed as part of the job
24 steps:
25 # Checks-out your repository under $GITHUB_WORKSPACE, so your job can access it
26 - uses: actions/checkout@v2
27
28 # Runs a single command using the runners shell
29 - name: Run a one-line script
30 run: echo Hello, world!
```

Use Control + Space to trigger autocomplete in most situations.

Marketplace Documentation

Search Marketplace for Actions

Featured Actions

- Cache** By actions 2.3k Cache artifacts like dependencies and build outputs to improve workflow execution time
- Setup Go environment** By actions 545 Setup a Go environment and add it to the PATH
- Close Stale Issues** By actions 481 Close issues and pull requests with no recent activity
- Setup .NET Core SDK** By actions 350 Used to build and publish .NET source. Set up a specific version of the .NET and authentication to private NuGet repository
- First interaction** By actions 146 Greet new contributors when they create their first issue or open their first pull request

Featured categories



# Modified workflow for python

```
1 name: Python Build
2
3 on:
4 push:
5 branches: [main]
6
7 workflow_dispatch:
8
9 jobs:
10 build:
11 runs-on: ubuntu-latest
12
13 steps:
14 - uses: actions/checkout@v2
15
16 - name: Set up Python 3.9
17 uses: actions/setup-python@v2
18 with:
19 python-version: 3.9
20
21 - name: Install dependencies
22 run:
23 - python -m pip install --upgrade pip
24 - pip install flake8 pytest
25 - if [-f requirements.txt]; then pip install -r requirements.txt; fi
26
27
28
29
```

The screenshot shows the GitHub Actions interface. On the left, there's a sidebar with 'Workflows' and 'New workflow' buttons, and a search bar for 'Python Build'. The main area is titled 'All workflows' with a subtitle 'Showing runs from all workflows'. It includes a search bar for 'Filter workflow runs' and a table header with columns for 'Event', 'Status', 'Branch', and 'Actor'. Two workflow runs are listed:

Event	Status	Branch	Actor
1 minute ago	Success	main	orreksu
7 minutes ago	Success	main	orreksu

Each row shows a green checkmark icon, the workflow name, a brief description, the branch it ran on, and the actor who triggered it.



# A workflow that uses pytest

```
jobs:
 build:
 runs-on: ubuntu-latest

 steps:
 - uses: actions/checkout@v2

 - name: Set up Python 3.9
 uses: actions/setup-python@v2
 with:
 python-version: 3.9

 - name: Install dependencies
 run: |
 python -m pip install --upgrade pip

 if [-f requirements.txt]; then pip install -r requirements.txt; fi

 test:
 needs: build
 runs-on: ubuntu-latest

 steps:
 - name: Install test tools
 pip install flake8 pytest
 - name: Test with pytest
 run: |
 pytest
```

```
1 # content of test_sample.py
2 def inc(x):
3 | return x + 1
4
5 def test_answer_1():
6 | assert inc(3) == 5
7
8 def test_answer_2():
9 | assert inc(3) == 4
```

build-test  
failed now in 52s

- >  Set up job
- >  Run actions/checkout@v2
- >  Set up Python 3.9
- >  Install dependencies
- >  Test with pytest
  - 1 ▼ Run pytest
  - 2   pytest
  - 3   shell: /usr/bin/bash -e {0}
  - 4   env:
  - 5    pythonLocation: /opt/hostedtoolcache/Python/3.9.7/x64
  - 6    LD\_LIBRARY\_PATH: /opt/hostedtoolcache/Python/3.9.7/x64/lib
  - 8 ===== test session starts =====
  - 9 platform linux -- Python 3.9.7, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
  - 10 rootdir: /home/runner/work/greenvest-app/greenvest-app
  - 11 plugins: dash-1.20.0
  - 12 collected 0 items
  - 13
  - 14 ===== no tests ran in 0.01s =====
  - 15 Error: Process completed with exit code 5.
- >  Post Run actions/checkout@v2
- >  Complete job



# Heroku

The screenshot shows the Heroku dashboard interface. At the top, there is a navigation bar with the Heroku logo, a "Jump to Favorites, Apps, Pipelines, Spaces..." button, and a "New" button. Below the navigation bar, the main content area displays a message: "You don't have any apps yet" followed by the subtext "Every app and pipeline you create or become a collaborator on will appear here". A prominent blue "Create new app" button is centered below this message. Further down, another section asks "Looking for help getting started?" and provides a link to "Choose a language guide...".



# Heroku Deployment

```
$ git init
Initialized empty Git repository in .git/
$ git add .
$ git commit -m "My first commit"
Created initial commit 5df2d09: My first commit
 44 files changed, 8393 insertions(+), 0 deletions(-)
 create mode 100644 README
 create mode 100644 Procfile
 create mode 100644 app/controllers/source_file
...
```

 Be sure to initialize the Git repository in your app's root directory. If your app is in a subdirectory of your repository, it won't run when it is pushed to Heroku.

Your app's code is now tracked in a local Git repository. It has not yet been pushed to any remote servers.

## Creating a Heroku remote

Git [remotes](#) are versions of your repository that live on other servers. You deploy your app by pushing its code to a special Heroku-hosted remote that's associated with your app.

### For a new Heroku app

The `heroku create` CLI command creates a new empty application on Heroku, along with an associated empty Git repository. If you run this command from your app's root directory, the empty Heroku Git repository is automatically set as a remote for your local repository.

With the `heroku git:remote` command, all you need is your Heroku app's name.

```
$ heroku git:remote -a thawing-inlet-61413
set git remote heroku to https://git.heroku.com/thawing-inlet-61413.git
```

### Renaming remotes

By default, the Heroku CLI names all of the Heroku remotes it creates for your app `heroku`. You can rename your remotes with the `git remote rename` command:

```
$ git remote rename heroku heroku-staging
```

Renaming your Heroku remote can be handy if you have multiple Heroku apps that use the same codebase (for example, the staging and production versions of an app). In this case, each Heroku app has its own remote in your local repository.

The remainder of this article assumes your app has a single Heroku remote that is named `heroku`.



# Creating the Heroku App

Create New App

App name

 ✓

greenvest-app is available

Choose a region

 ▼

>Add to pipeline...

Create app



# Install the Heroku Command-Line Interface (CLI)

## Deployment method



Heroku Git  
Use Heroku CLI



Github  
Connect to GitHub



Container Registry  
Use Heroku CLI

## Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

### Install the Heroku CLI

Download and install the [Heroku CLI](#).

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

### Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a greenvest-app
```

### Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```



You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions [here](#).

### Existing Git repository

For existing repositories, simply add the heroku remote

```
$ heroku git:remote -a greenvest-app
```



# Our previous test is failing....

```
✖ Test with pytest

1 ▶ Run pytest
7 ===== test session starts =====
8 platform linux -- Python 3.9.7, pytest-6.2.5, py-1.10.0, pluggy-1.0.0
9 rootdir: /home/runner/work/greenvest-app/greenvest-app
10 plugins: dash-1.20.0
11 collected 2 items
12
13 test_app.py F. [10]
14
15 ===== FAILURES =====
16 ----- test_answer_1 -----
17
18 def test_answer_1():
19 > assert inc(3) == 5
20 E assert 4 == 5
21 E + where 4 = inc(3)
22
23 test_app.py:6: AssertionError
24 ===== short test summary info =====
25 FAILED test_app.py::test_answer_1 - assert 4 == 5
26 ===== 1 failed, 1 passed in 0.05s =====
27 Error: Process completed with exit code 1.

> ✓ Post Run actions/checkout@v2
> ✓ Complete job
```

All workflows			
Showing runs from all workflows			
Event	Status	Branch	Actor
Fix tests	In progress	main	orreksu
Merge branch 'main' of https://github.com/orreksu/gr...	51s	main	orreksu
Update python_build.yml	1m 5s	main	orreksu
Update python_build.yml	1m 2s	main	orreksu
Update and rename simple_ci.yml to python_build.yml	53s	main	orreksu
Create simple_ci.yml	18s	main	orreksu

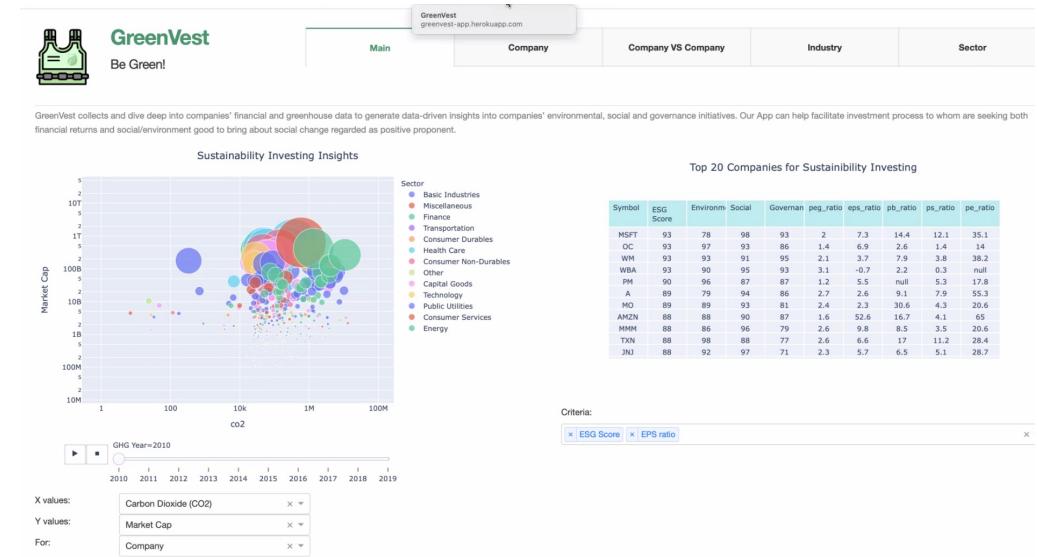
retriggering tests after fixing:



```

greenvest_app on ✚ main
> git push heroku main
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 16 threads
Compressing objects: 100% (55/55), done.
Writing objects: 100% (66/66), 2.60 MiB | 2.74 MiB/s, done.
Total 66 (delta 23), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Determining which buildpack to use for this app
remote: -----> Python app detected
remote: -----> No Python version was specified. Using the buildpack default: python-3.9.7
remote: To use a different version, see: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> Installing python-3.9.7
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: Collecting Brotli==1.0.9
remote: Downloading Brotli-1.0.9-cp39-cp39-manylinux1_x86_64.whl (257 kB)

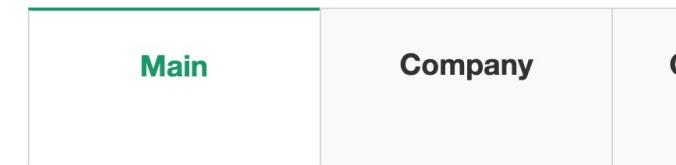
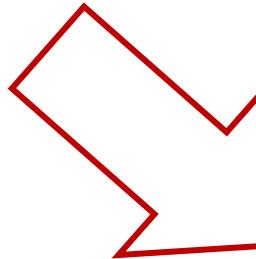
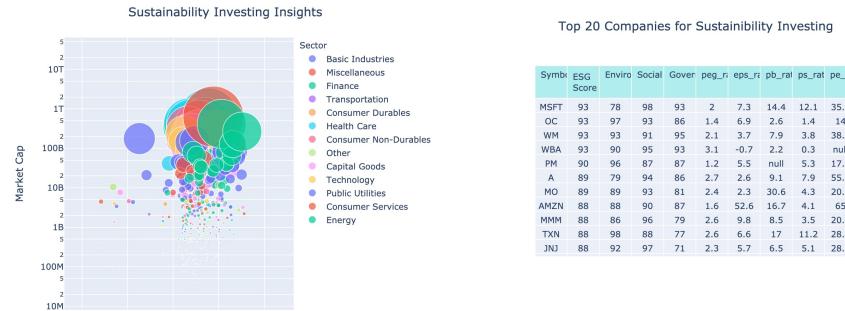
```



# Changing the app. Everything automated



GreenVest collects and dive deep into companies' financial and greenhouse data to generate data-driven insights into companies' environmental, social and governance initiatives. Our App can help facilitate investment process to whom are seeking both financial returns and social/environment good to bring about social change regarded as positive proponent.



GreenVest collects and dive deep into companies' financial and greenhouse data to generate data-driven insights into companies' environmental, social and governance initiatives. Our App can help facilitate investment process to whom are seeking both financial returns and social/environment good to bring about social change regarded as positive proponent.

Sustainability Investing Insights



Northeastern University

# Tracking your changes.

969 contributions in the last year

Contribution settings ▾



## Contribution activity

2021

October 2021

2020

Created 15 commits in 3 repositories

2019

[orreksu/greenvest-app](#) 8 commits

2018

[orreksu/greenvest](#) 6 commits

2017

[orreksu/orreksu.github.io](#) 1 commit

2016

Created 1 repository

2015

[orreksu/greenvest-app](#)

Python

Oct 8

2014

Show more activity

2013



Northeastern University

# Reliability, Scalability, and Maintainability

---



# Challenges

---

- **Reliability:** The system should continue to work *correctly* in response to hardware, software, or user errors
- **Scalability:** As the system grows (data volume, traffic, or complexity) there should be reasonable ways of dealing with that growth. The design should be *elastic*.
- **Maintainability:** Productively work on the system as people come and go. Add new features to adapt to changing requirements.



# What is reliability?

---

- **Predictable:** Application performs as expected
- **Fault-tolerant:** Application copes with human error & hardware failures.
- **Performant:** Performance is adequate under expected load and data volume
- **Secure:** System prevents unauthorized access



# Reliability

- **Faults** are a deviation in the spec of one particular component of the system. **Failures** cause the whole system to stop, requiring user intervention.
- A system is *fault-tolerant* if it anticipates and copes with certain kind of faults.
- One way to design fault-tolerant systems is to induce them deliberately! (e.g., submit bad input by the user, shut down a process, etc.)

The screenshot shows the Chaos Monkey configuration page. At the top, there is a status bar with the text "Chaos Monkey" and a checked checkbox labeled "Enabled". Below this, under the heading "Termination frequency", there are two input fields: "Mean time between terms" set to 2 days and "Minimum time between terms" set to 1 day. There are also three radio button options: "Grouping" (disabled), "App" (disabled), and "Cluster" (selected). A checked checkbox "Regions are independent" is also present. Under the heading "Exceptions", there is a table with four columns: "Account", "Region", "Stack", and "Detail". Two rows are listed: "prod" with "us-west-2" and "staging" in the respective columns, and "test" with "\*" in all columns. Each row has a delete icon at the end. At the bottom right of the exceptions section is a button labeled "+ Add Exception".

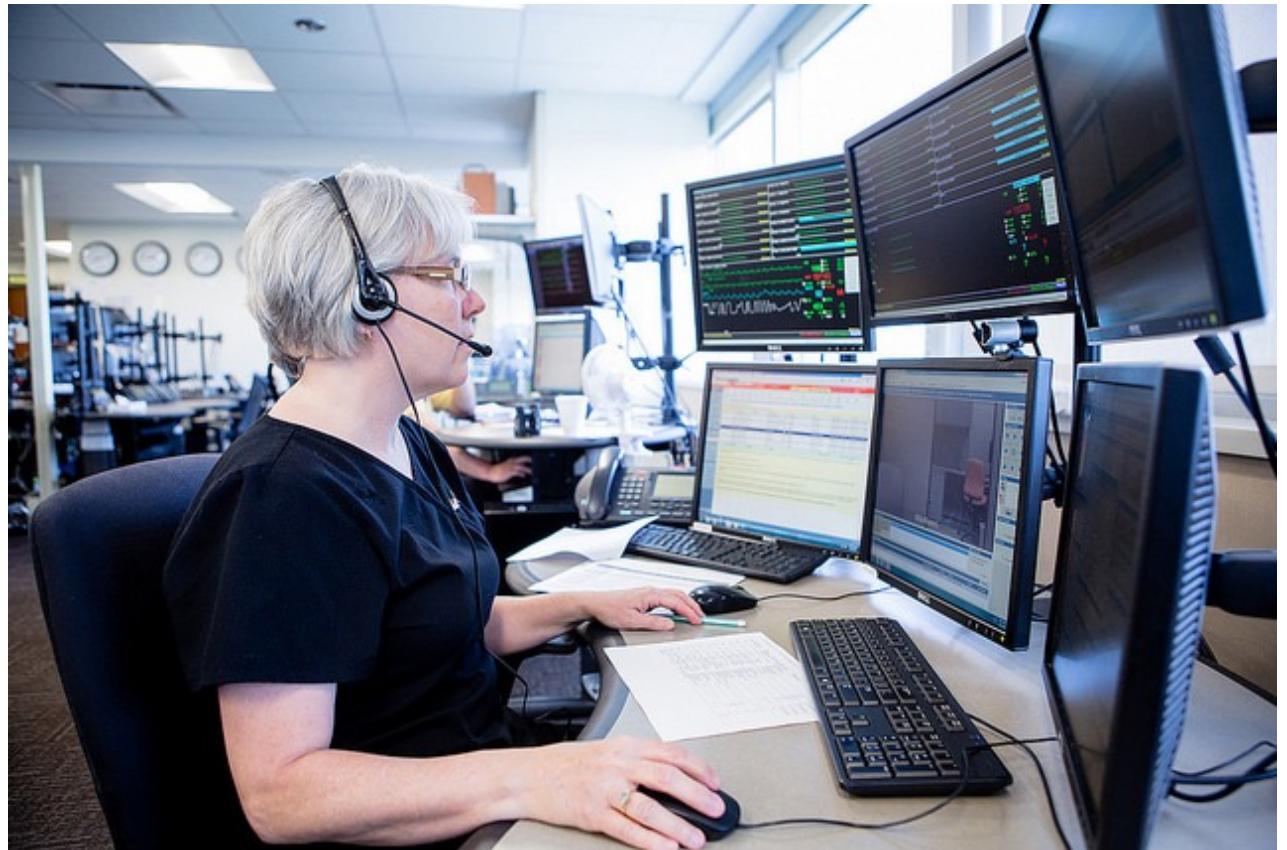
## Netflix Chaos Monkey



# Reliability: Human Errors

---

- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



# Maintainability

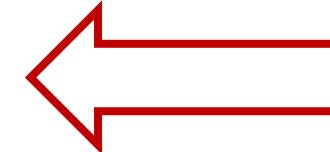
---

## Software cost breakdown:

Initial development: ~10-20%

Error correction: ~10-20%

On-going Maintenance: ~60-80%



**Operability:** Make it easy for operations teams to keep system running

**Simplicity:** Make it easy for engineers to understand

**Evolvability:** Make it easy for engineers to modify in response to changing requirements (*extensibility / plasticity*)



# Python Exception Hierarchy

```
BaseException
+-- SystemExit
+-- KeyboardInterrupt
+-- GeneratorExit
+-- Exception
 +-- StopIteration
 +-- StopAsyncIteration
 +-- ArithmeticError
 +-- FloatingPointError
 +-- OverflowError
 +-- ZeroDivisionError
 +-- AssertionError
 +-- AttributeError
 +-- BufferError
 +-- EOFError
 +-- ImportError
 +-- ModuleNotFoundError
 +-- LookupError
 +-- IndexError
 +-- KeyError
 +-- MemoryError
 +-- NameError
 +-- UnboundLocalError
 +-- OSError
 +-- BlockingIOError
 +-- ChildProcessError
 +-- ConnectionError
 +-- BrokenPipeError
 +-- ConnectionAbortedError
 +-- ConnectionRefusedError
 +-- ConnectionResetError
 +-- FileExistsError
 +-- FileNotFoundError
 +-- InterruptedError
 +-- IsADirectoryError
 +-- NotADirectoryError
 +-- PermissionError
 +-- ProcessLookupError
 +-- TimeoutError
```

```
+-- ReferenceError
+-- RuntimeError
| +-- NotImplementedError
| +-- RecursionError
+-- SyntaxError
| +-- IndentationError
| +-- TabError
+-- SystemError
+-- TypeError
+-- ValueError
| +-- UnicodeError
| +-- UnicodeDecodeError
| +-- UnicodeEncodeError
| +-- UnicodeTranslateError
+-- Warning
 +-- DeprecationWarning
 +-- PendingDeprecationWarning
 +-- RuntimeWarning
 +-- SyntaxWarning
 +-- UserWarning
 +-- FutureWarning
 +-- ImportWarning
 +-- UnicodeWarning
 +-- BytesWarning
 +-- ResourceWarning
```

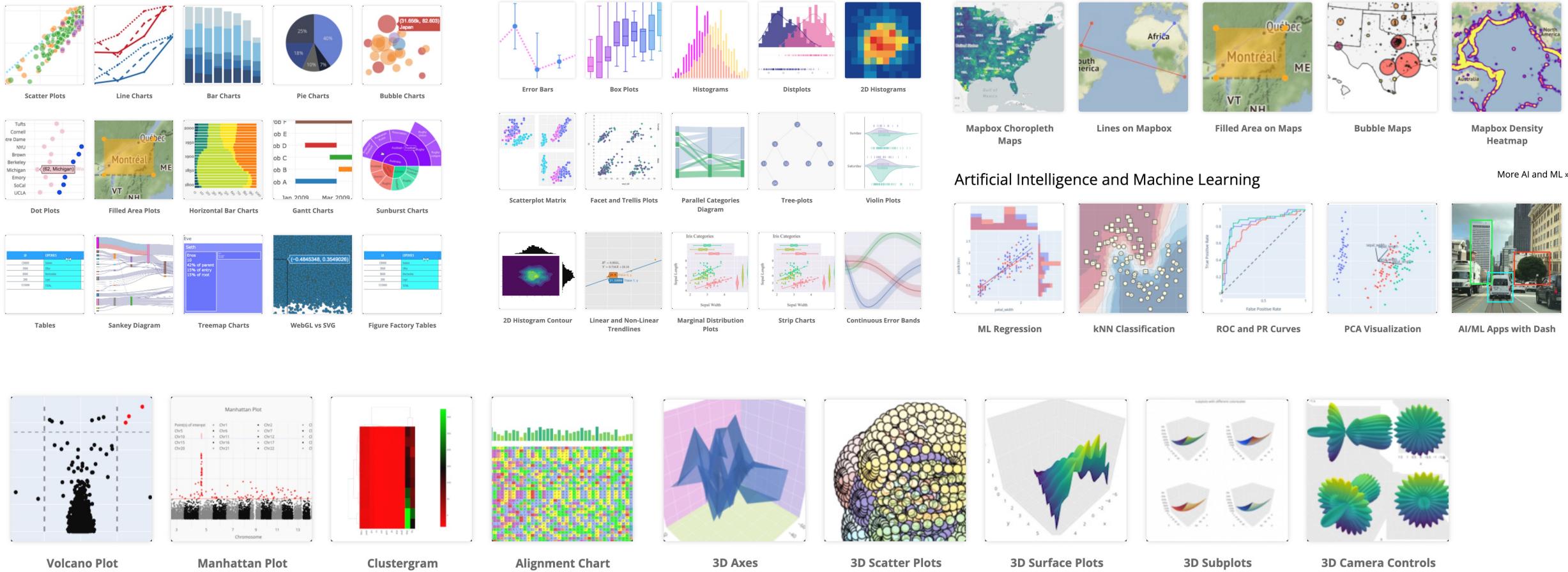
In Python, all exceptions must be instances of a class that derives from [BaseException](#). In a `try` statement with an `except` clause that mentions a particular class, that clause also handles any exception classes derived from that class (but not exception classes from which *it* is derived).

# Advanced Visualization with Plotly

---



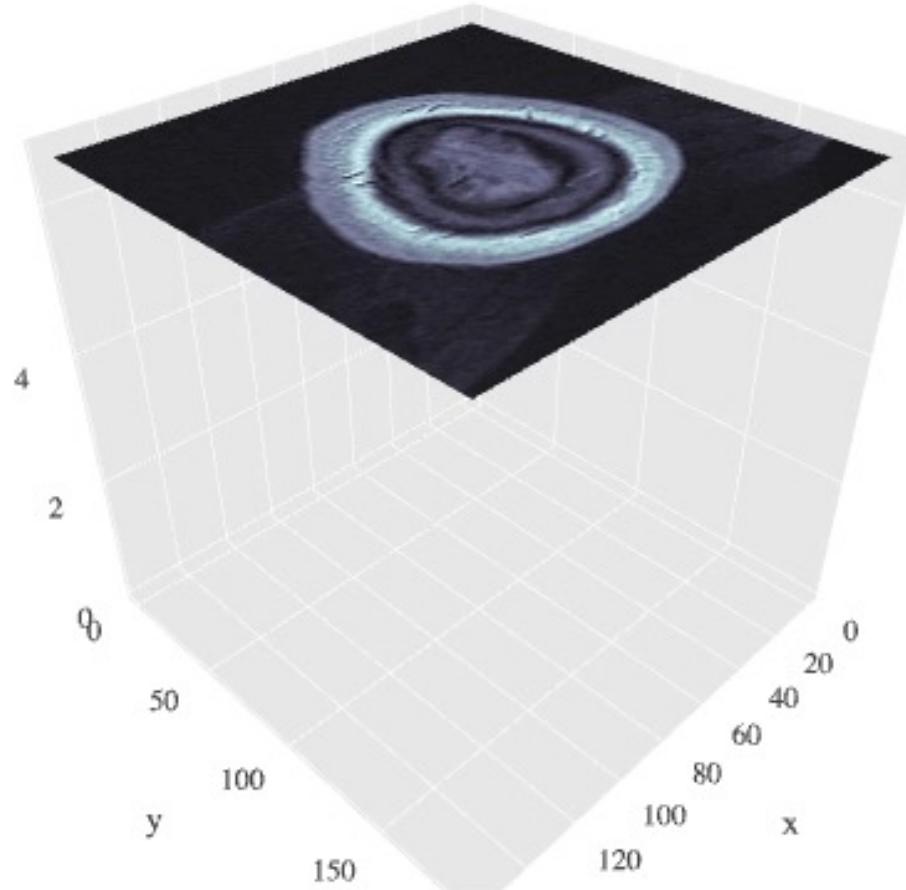
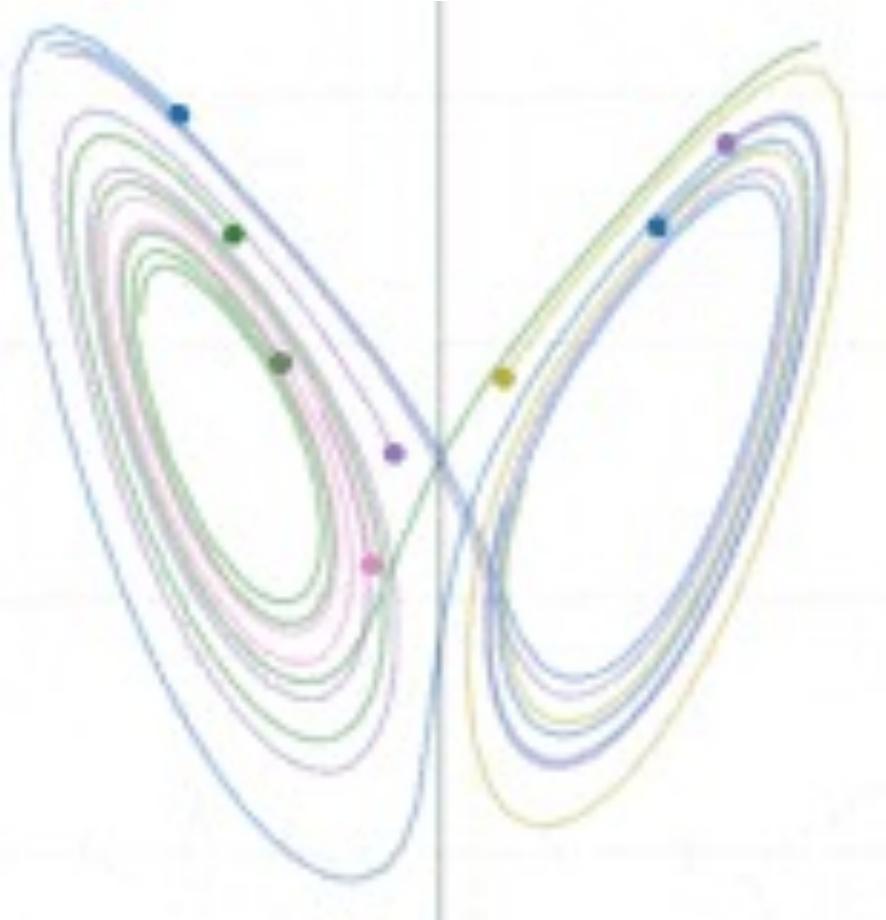
# A gallery of possibilities: <https://plotly.com/python/>



Northeastern University

# Animations!

---



## Is Plotly for Python Free?

**Yes.** Plotly for Python is free and open-source software, [licensed under the MIT license](#). It costs nothing to [install and use](#). You can view the source, report issues or contribute using [our Github repository](#).

## Can I use Plotly for Python without signing up to any service?

**Yes.** You can use Plotly for Python to make, view, and distribute charts and maps without registering for any service, obtaining any token, or creating any account. The one exception is that to view tile maps which use tiles from the Mapbox service (which is optional, as [you can use other tile servers](#)), you will need to have a Mapbox token.

## Can I use Plotly for Python offline, without being connected to the internet?

**Yes.** You can use Plotly for Python to make, view, and distribute graphics totally offline. The one exception is that to view tile maps which use tiles from a cloud-hosted service, such as Open Street Maps or Mapbox, you will need a connection to that service. You can view tile maps totally offline if you run your own local tile server and [use its tiles](#).



# Free Software: Free as in beer / Free as in speech

Free as in Beer



Free as in Speech



- The software doesn't cost you anything (monetarily) to use.
- There are still restrictions on **how** you use the software – for example you can't necessarily view, modify, or redistribute the source code.
- You are at liberty to use the software however you like.
- You can inspect the source code.
- You can redistribute or repackage the code.
- You can create your own improved versions.



# Installation

```
$ conda install -c plotly plotly=5.5.0
```

Or create a new environment  
and install it there

```
$ conda create -name py37 python=3.7
```

```
$ conda activate py37
```

```
$ conda install -c plotly plotly=5.5.0
```

(Took about 1 minute)

```
(py37) [508 uranus:~] conda install -c plotly plotly=5.5.0
Collecting package metadata (current_repodata.json): done
Solving environment: done

Package Plan

environment location: /Users/rachlin/opt/anaconda3/envs/py37

added / updated specs:
- plotly=5.5.0

The following NEW packages will be INSTALLED:

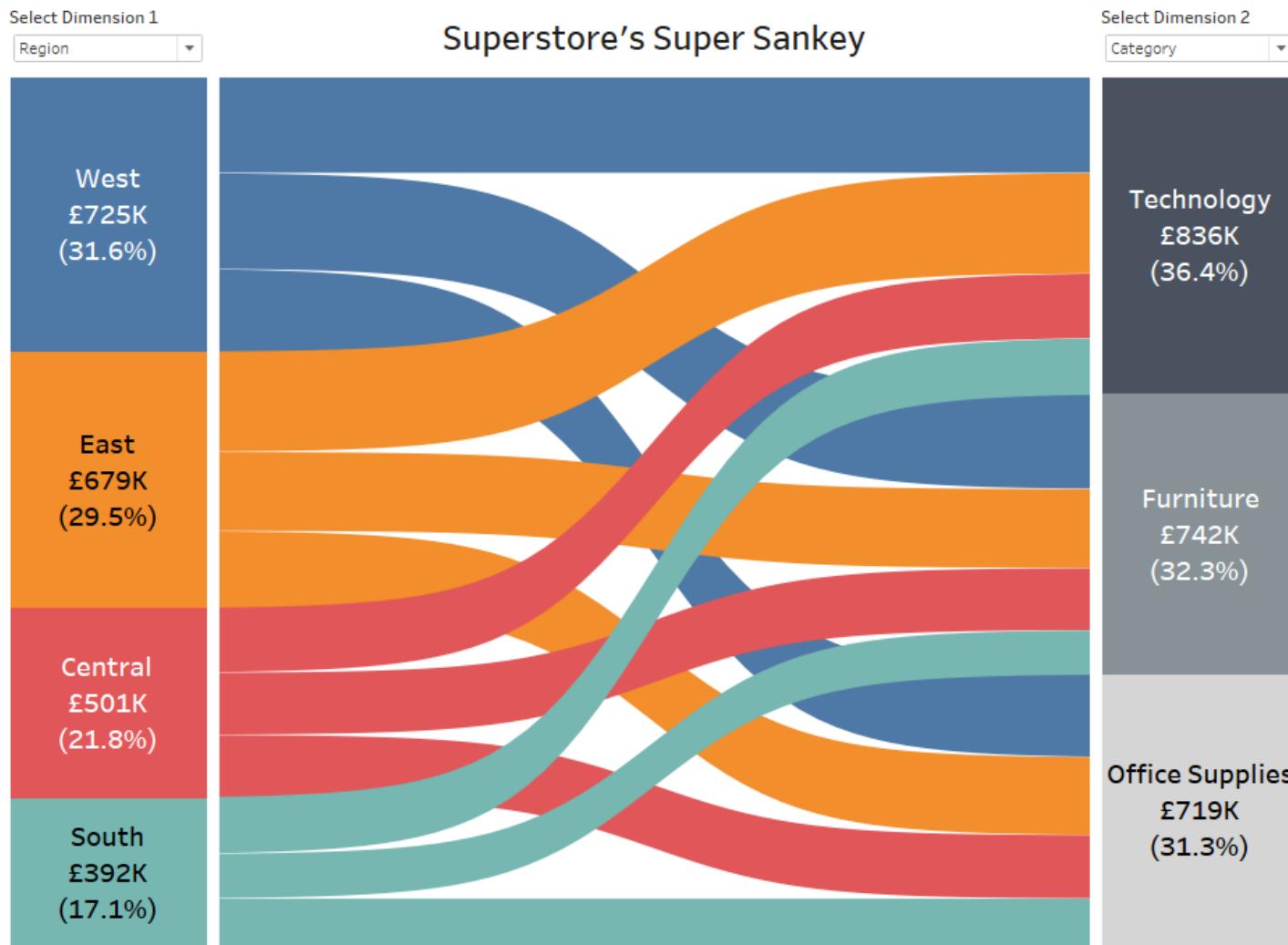
plotly plotly/noarch::plotly-5.5.0-py_0
tenacity conda-forge/noarch::tenacity-8.0.1-pyhd8ed1ab_0

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```



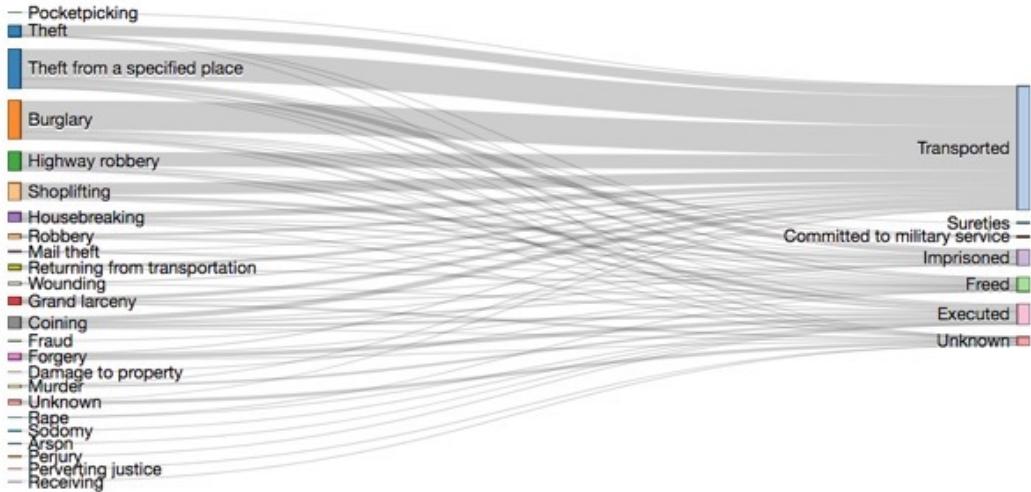
# Sankey Diagrams: Market Analysis



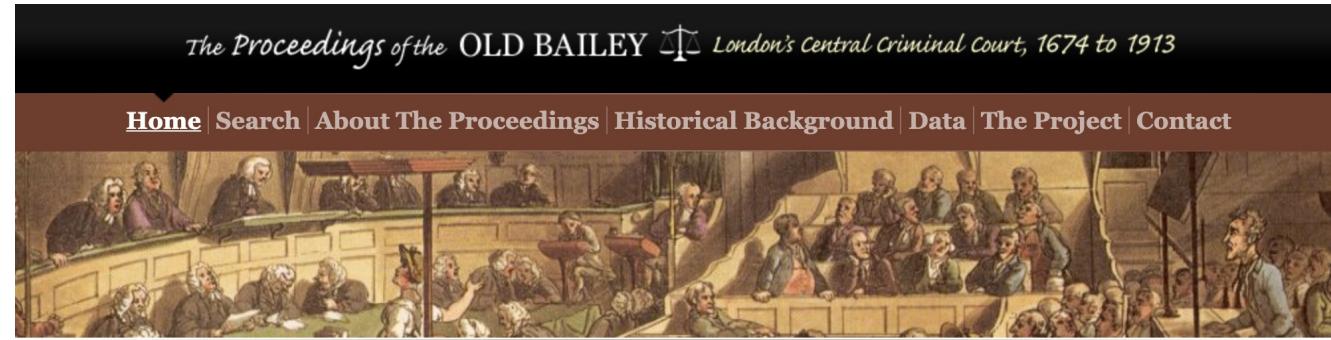
# Crime and Punishment in the Old Bailey

What actually happened to defendants sentenced to death?

763 results   [Change visualisation](#)   [View as hitlist](#)



Offence is shown on the left and **sentence outcome** on the right. All of these defendants were sentenced to death between 1810 and 1815. But many Old Bailey defendants who were sentenced to death were not actually executed. This Sankey diagram shows what actually happened to them. [See all the results](#)



[Home Page](#)

[Search](#)

[About the Proceedings](#)

[Historical Background](#)

[API](#)

[The Project](#)

[Copyright & Citation Guide](#)

[Contact](#)

ON THIS DAY IN... **1722**

James Lanman, (aged 11) and Samuel Armstrong, (aged 13) allegedly stole a silver snuffbox from Simon Hansel's shop. A month later they were sentenced to hang. [read more](#)

## The Proceedings of the Old Bailey, 1674-1913

A fully searchable edition of the largest body of texts detailing the lives of non-elite people ever published, containing 197,745 criminal trials held at London's central criminal court. If you are new to this site, you may find the [Getting Started](#) and [Guide to Searching](#) videos and tutorials helpful.

To search the Proceedings use the boxes on the right or go to the [Search Pages](#).

This site uses cookies. See our [privacy policy](#).

**February 2018 Update**

Several bugs have been fixed and tagging errors corrected. In addition, the following new features have been added:

**SEARCH**  
the Proceedings

Keyword(s)

Reference No.

Search In

<All Text>

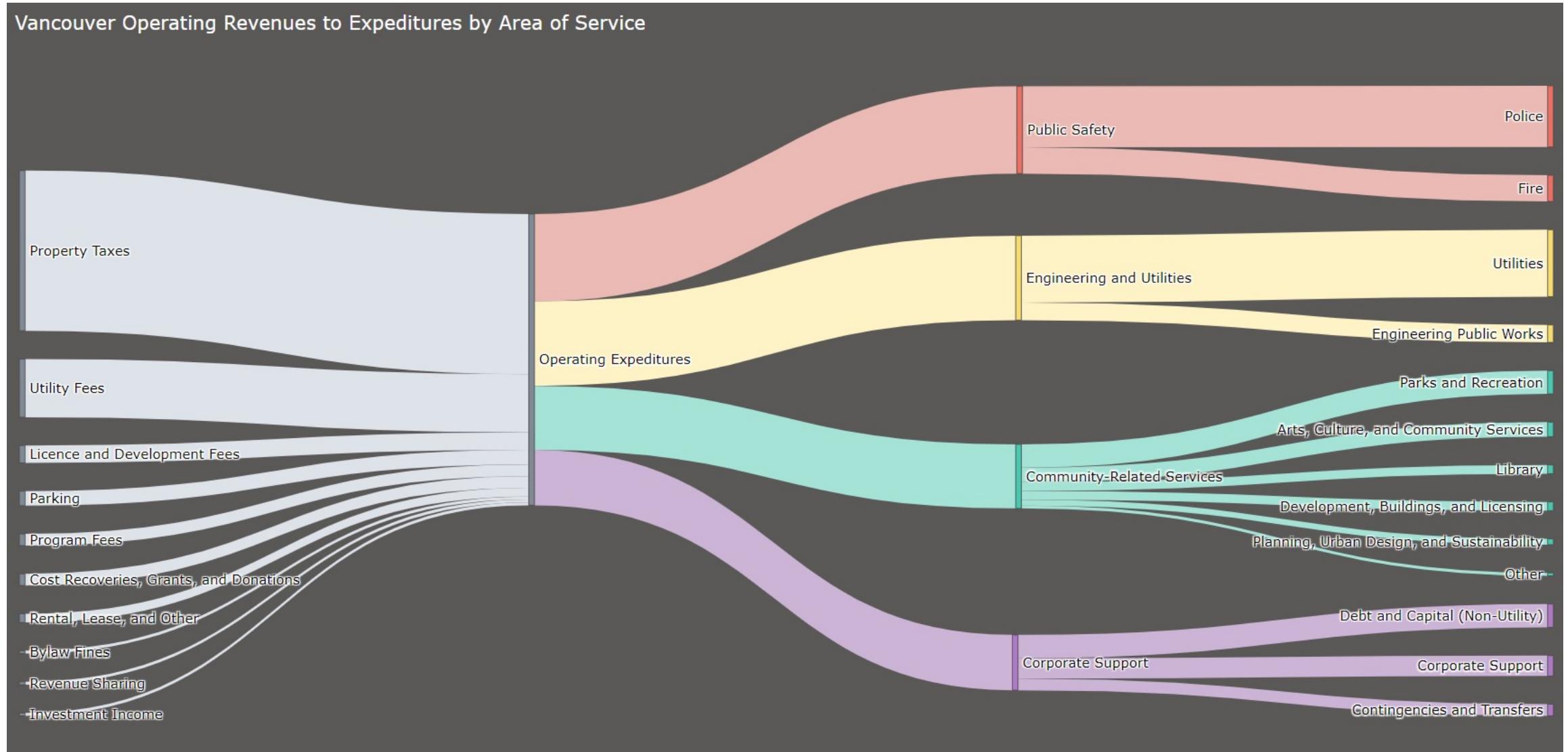
**SEARCH**

The Proceedings can also be searched in:

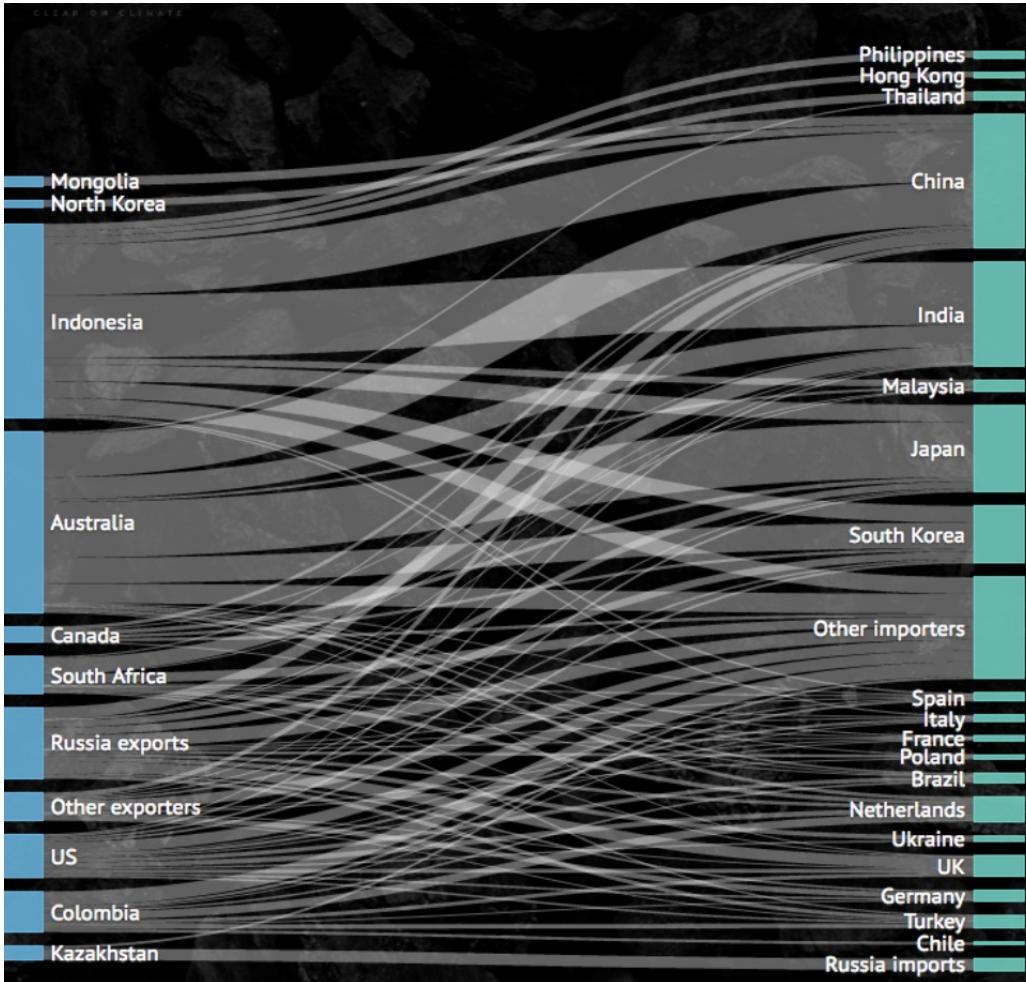


Northeastern University

# Sankey Diagrams: Analyzing Expenses



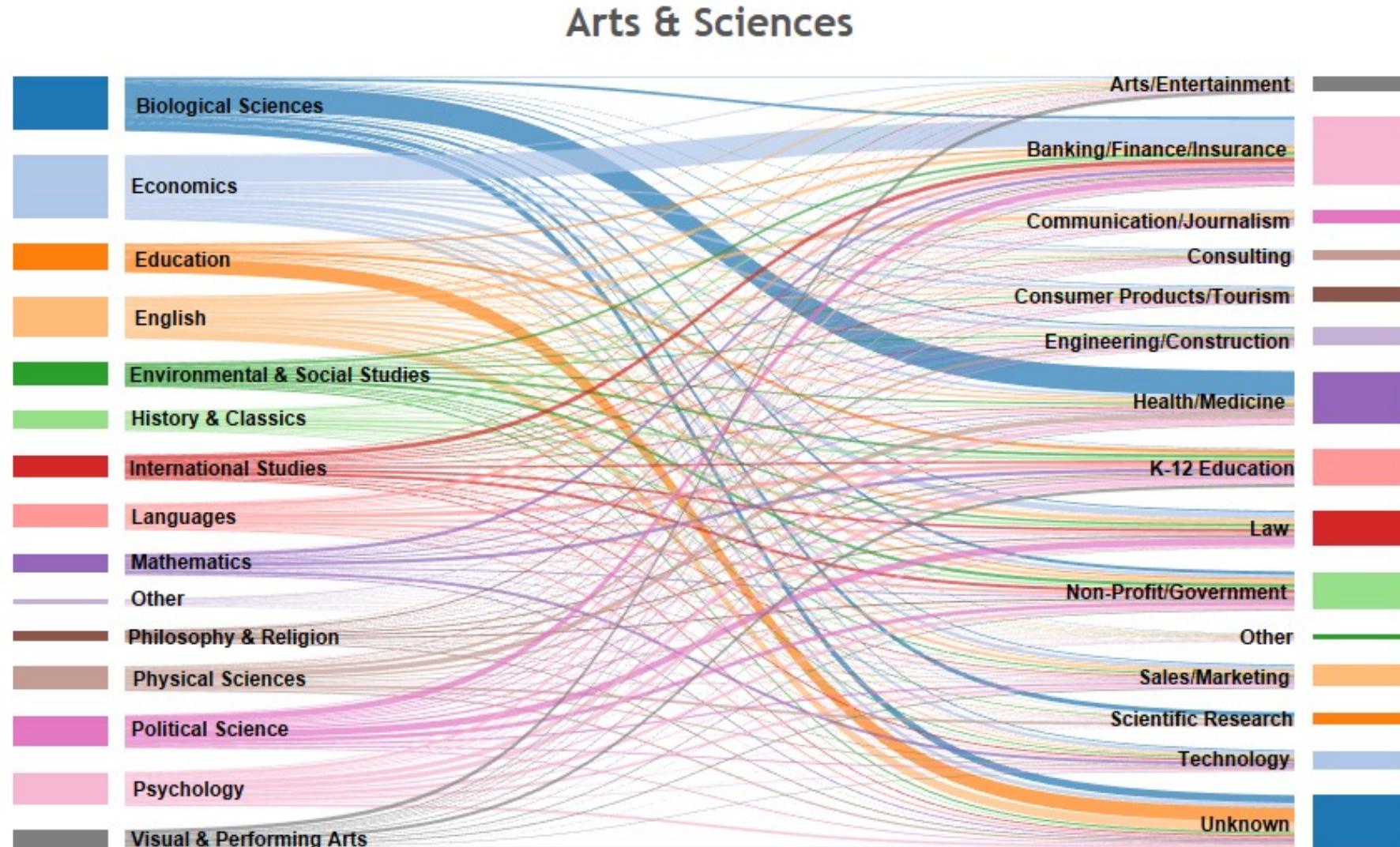
# Sankey Diagrams: Trade



## 2014 Coal Exports and Imports

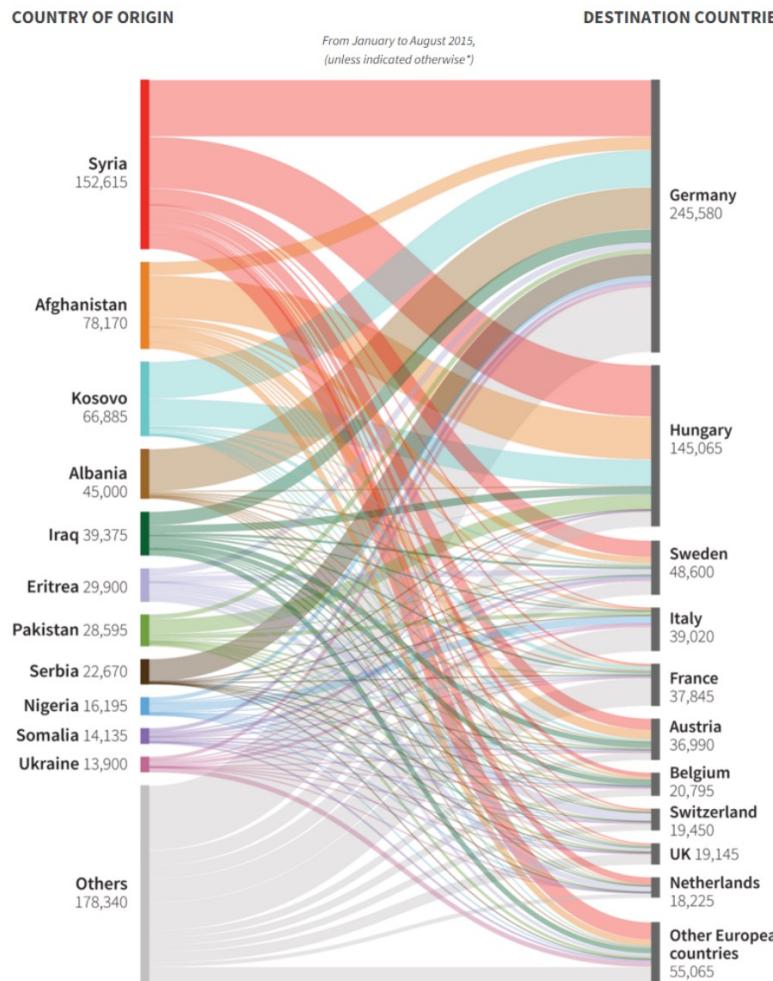


# Sankey Diagrams: The value of a liberal arts education

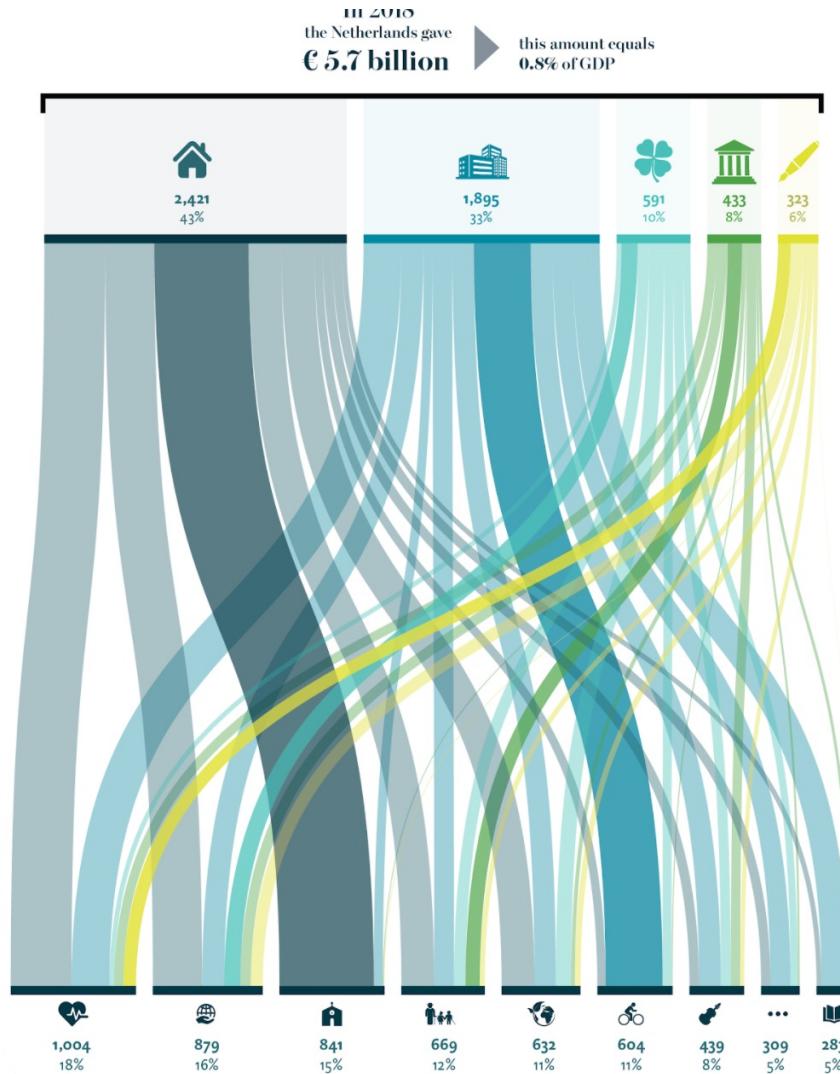


# Sankey Diagrams: Migration Patterns

Asylum seekers in Europe



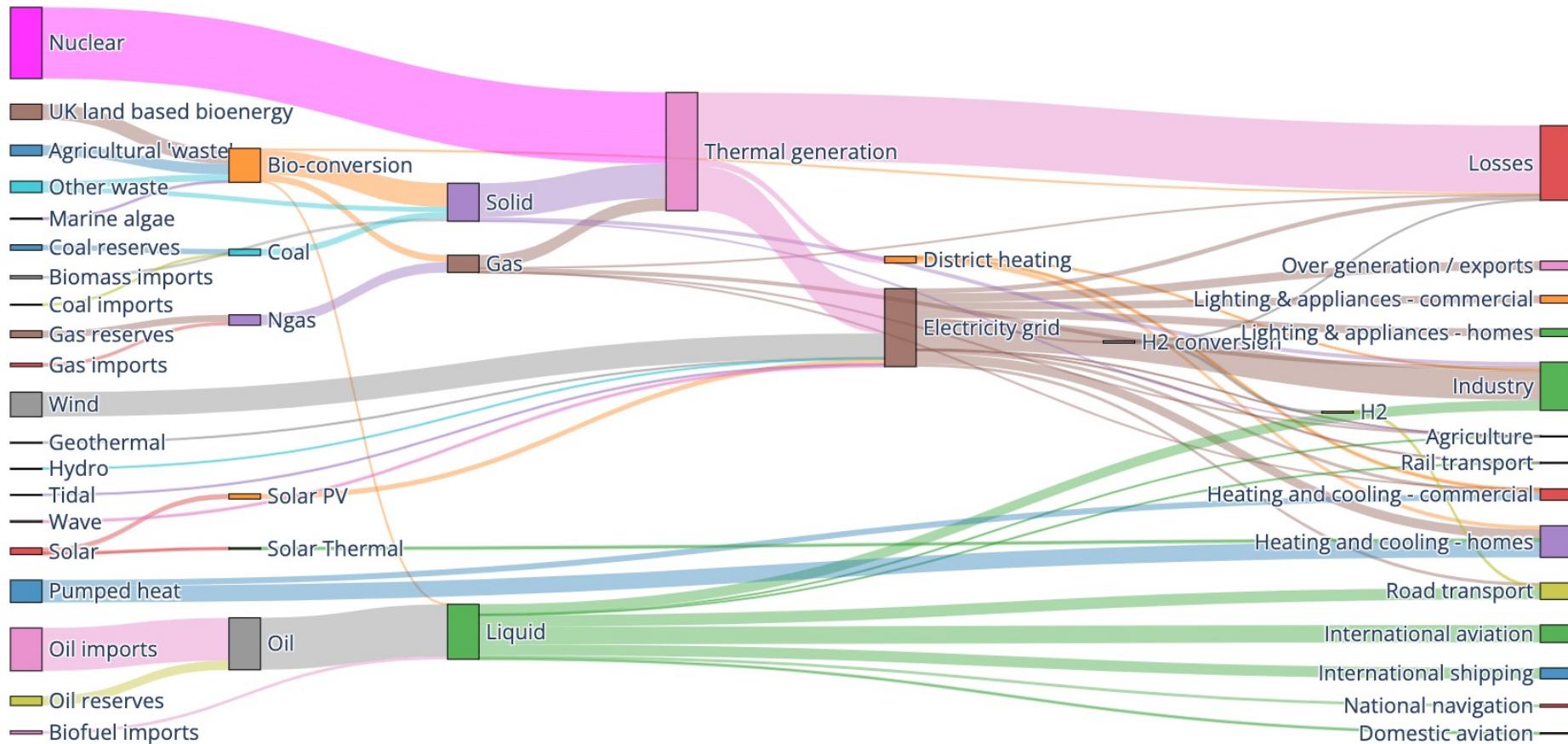
\*\*Latest data available for Czech Republic, Denmark, Estonia, Ireland, Greece, France, Italy, Latvia, Malta, Austria, Portugal, Romania, Slovakia, Finland, UK, and Iceland is for July. Latest data available for Spain, Cyprus and Liechtenstein is for the month of June. Eurostat data as of Oct. 2



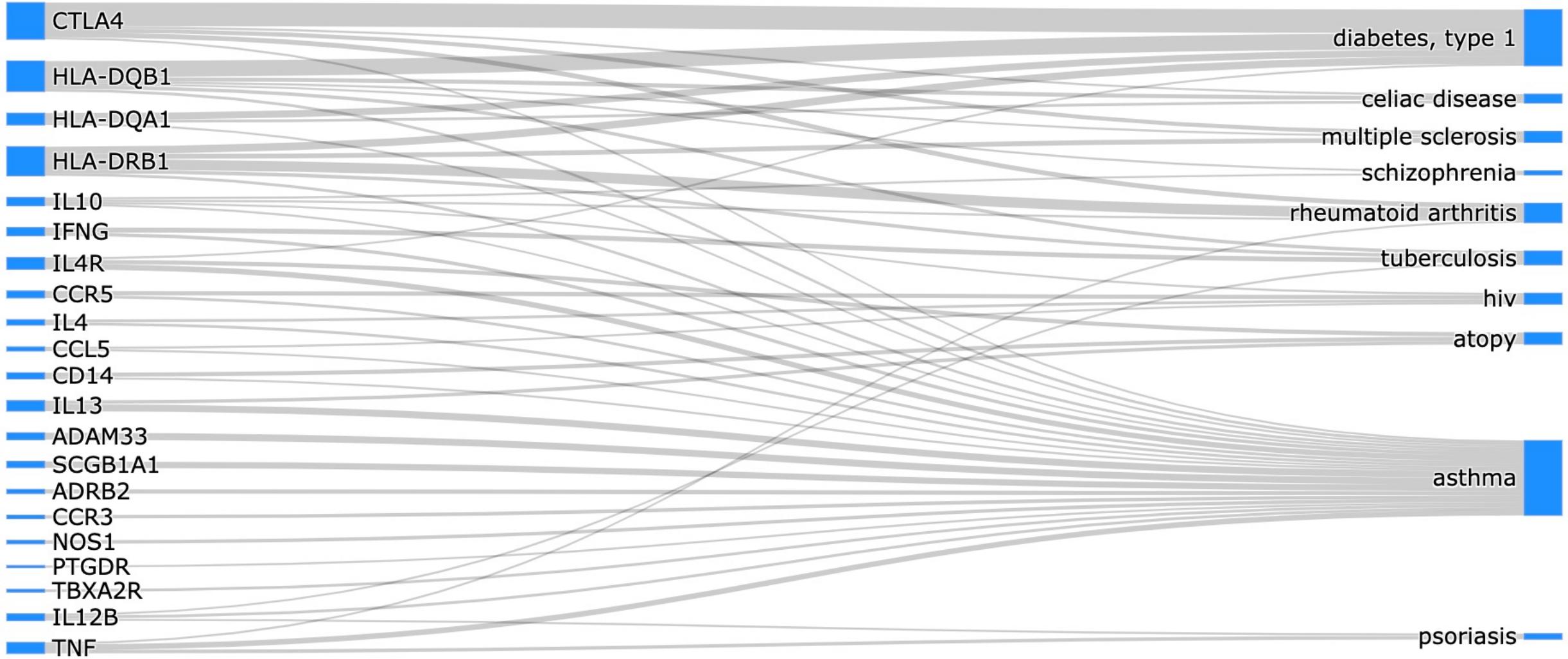
# Energy Forecasting

Energy forecast for 2050

Source: Department of Energy & Climate Change, Tom Counsell via [Mike Bostock](#)



# Sankey: The Genetic Association Database



# Links between Asthma and Type 1 Diabetes

This Issue

Views 3,824 | Citations 5 | Altmetric 40 | Comments 1



PDF



More ▾



Cite



Permissions

Original Investigation | Pediatrics



March 12, 2020

## Familial Coaggregation of Asthma and Type 1 Diabetes in Children

Awad I. Smew, MD<sup>1</sup>; Cecilia Lundholm, MSc<sup>1</sup>; Lars Sävendahl, MD, PhD<sup>2</sup>; et al

» Author Affiliations | Article Information

JAMA Netw Open. 2020;3(3):e200834. doi:10.1001/jamanetworkopen.2020.0834

### Key Points | Español | 中文 (Chinese)

**Question** Is there an association between childhood asthma and type 1 diabetes and do shared familial factors contribute to the comorbidity?

Medicine®

Wolters Kluwer

Home

Search

Submit a Manuscript

Medicine (Baltimore). 2015 Sep; 94(36): e1466.

PMCID: PMC46166

Published online 2015 Sep 11.

PMID: 263567

doi: [10.1097/MD.0000000000001466](https://doi.org/10.1097/MD.0000000000001466)

### Type 1 Diabetes and Increased Risk of Subsequent Asthma

A Nationwide Population-Based Cohort Study

Yung-Tsung Hsiao, MD, [Wen-Chien Cheng, MD](#), [Wei-Chih Liao, MD](#), [Cheng-Li Lin, MS](#), [Te-Chun Shen, MD](#), [Wei-Chun Chen, MD](#), [Chia-Hung Chen, MD](#), and [Chia-Hung Kao, MD](#)

Monitoring Editor: Zhu. Xiaolin

» Author information » Article notes » Copyright and License information [Disclaimer](#)

This article has been [cited by](#) other articles in PMC.

### Abstract

Search | Copy

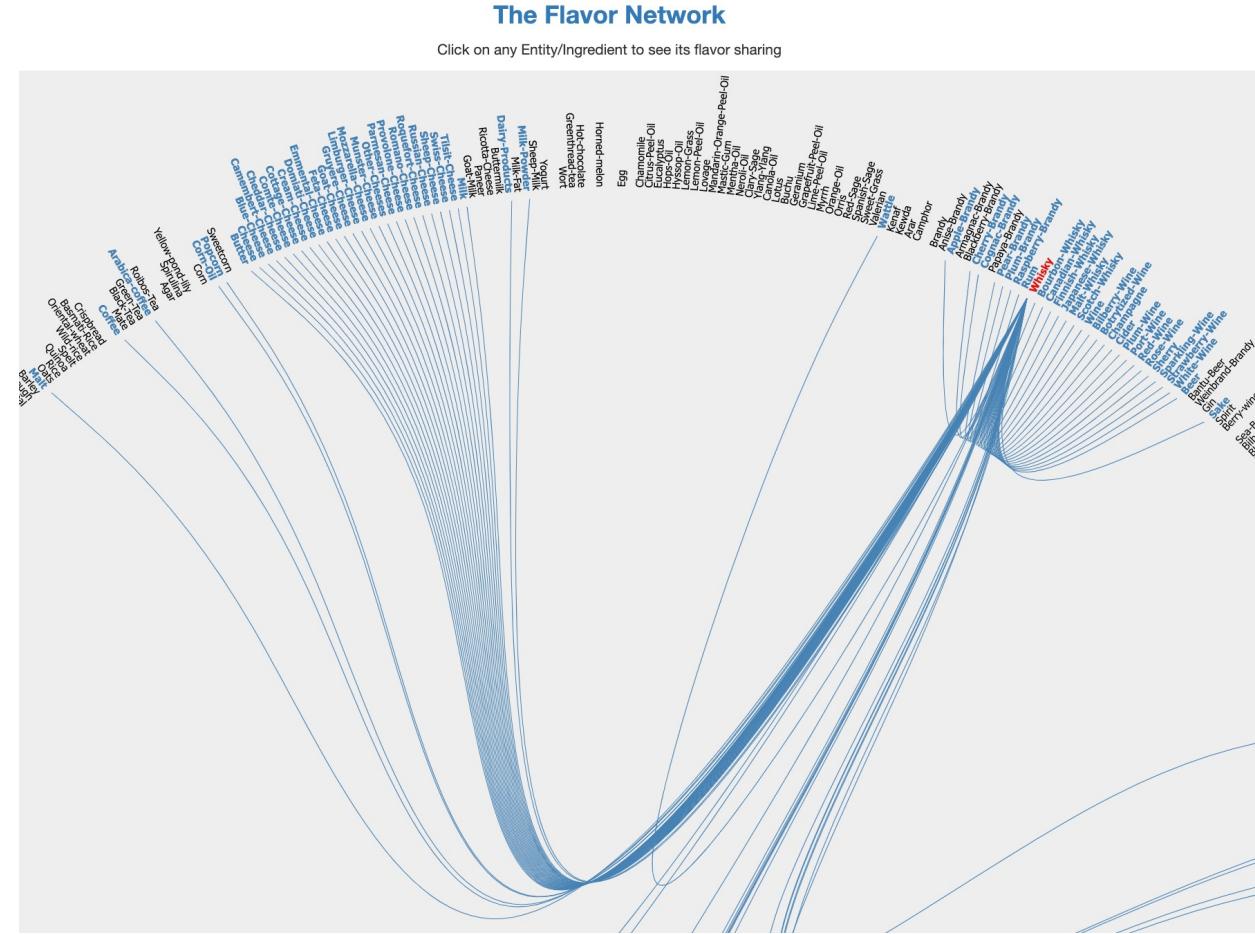
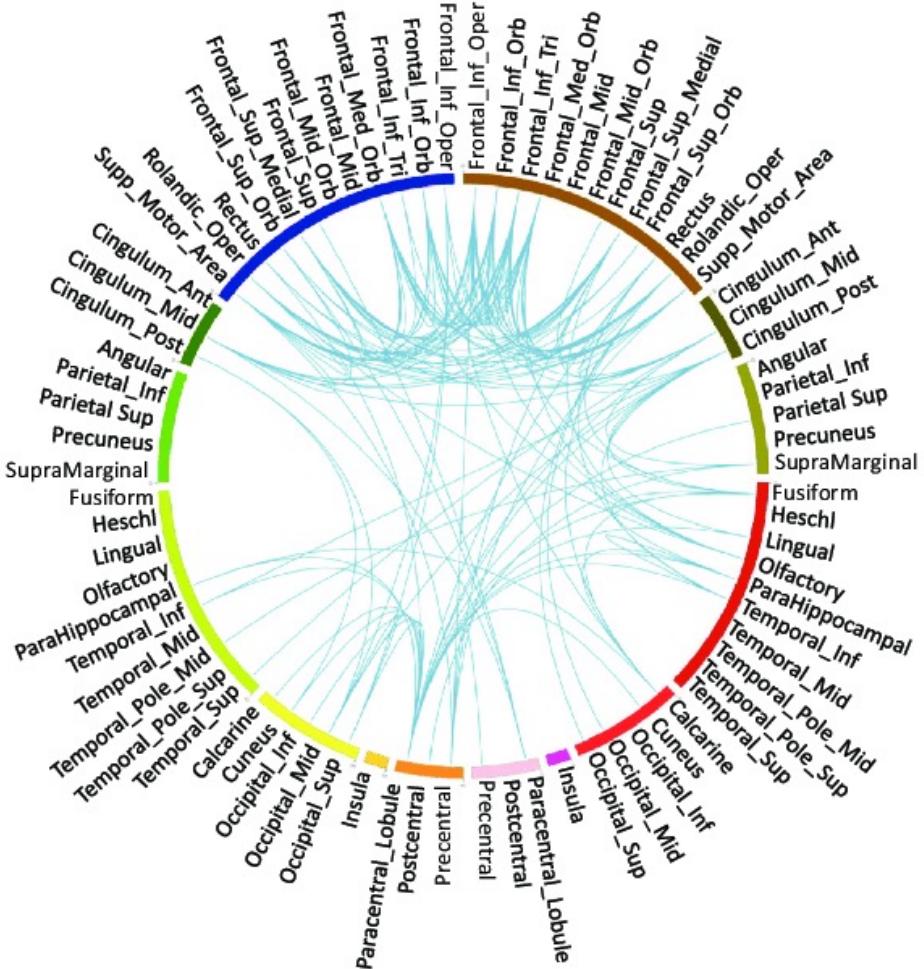
Go to:

The association between type 1 diabetes mellitus (T1DM) and asthma remains controversial and has led to new interest in these 2 disorders. The purpose of this study was to examine the associations among young people with T1DM and asthma and offer a clinical demonstration of the balance between Th1 and Th2 responses.

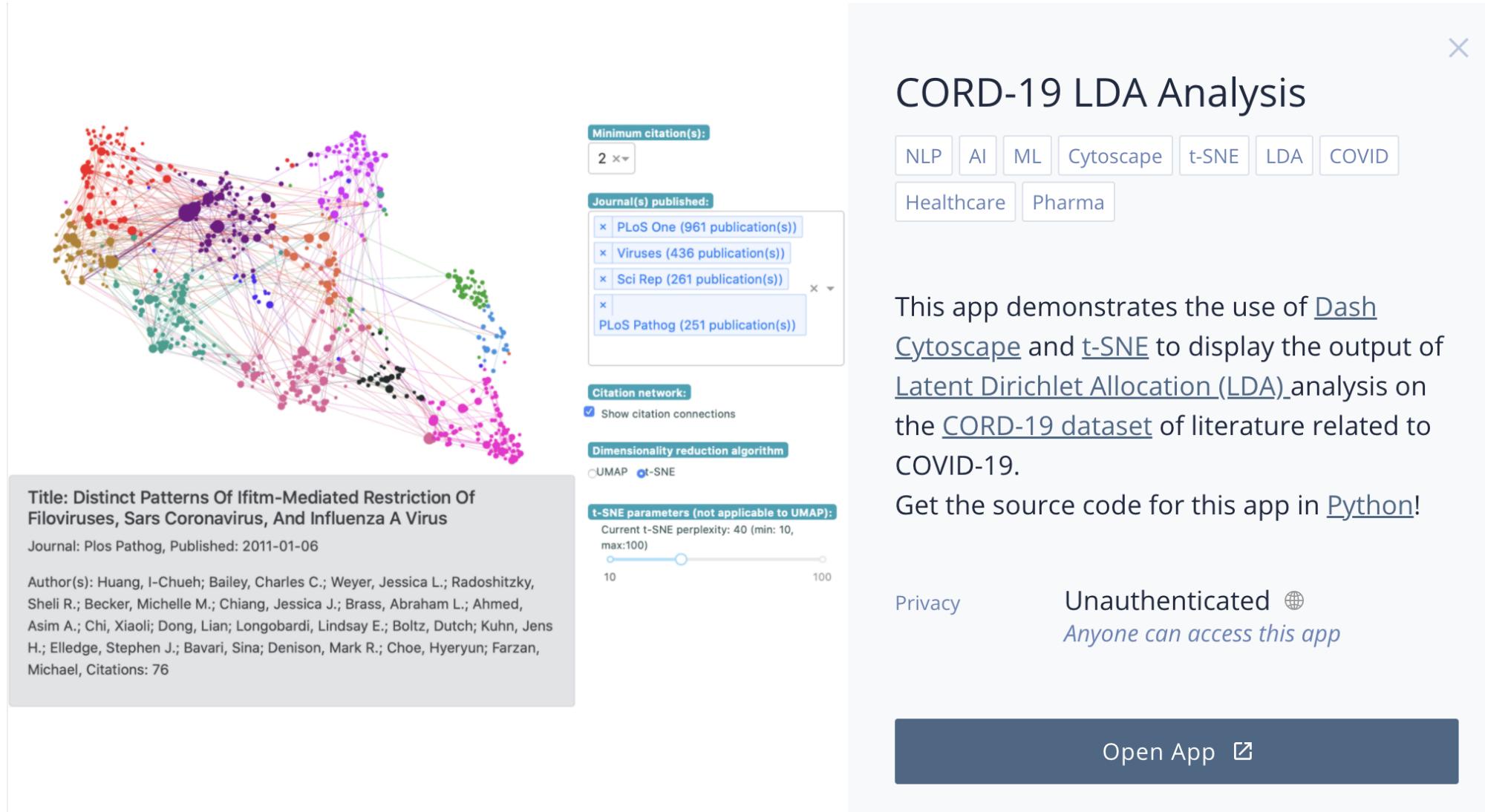


Northeastern University

# Circle Graphs: Single-Set Sankey

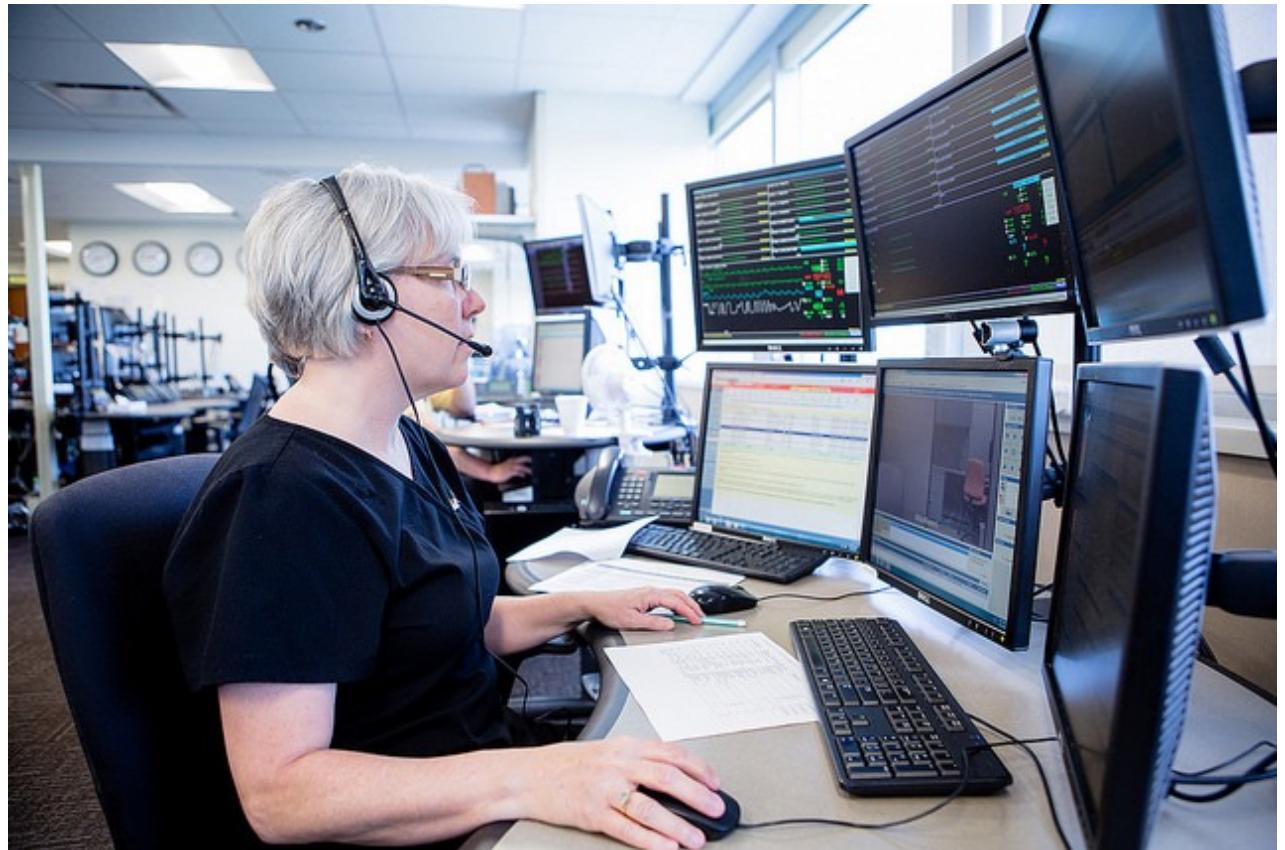


# Dashboarding for Interactive Visualization

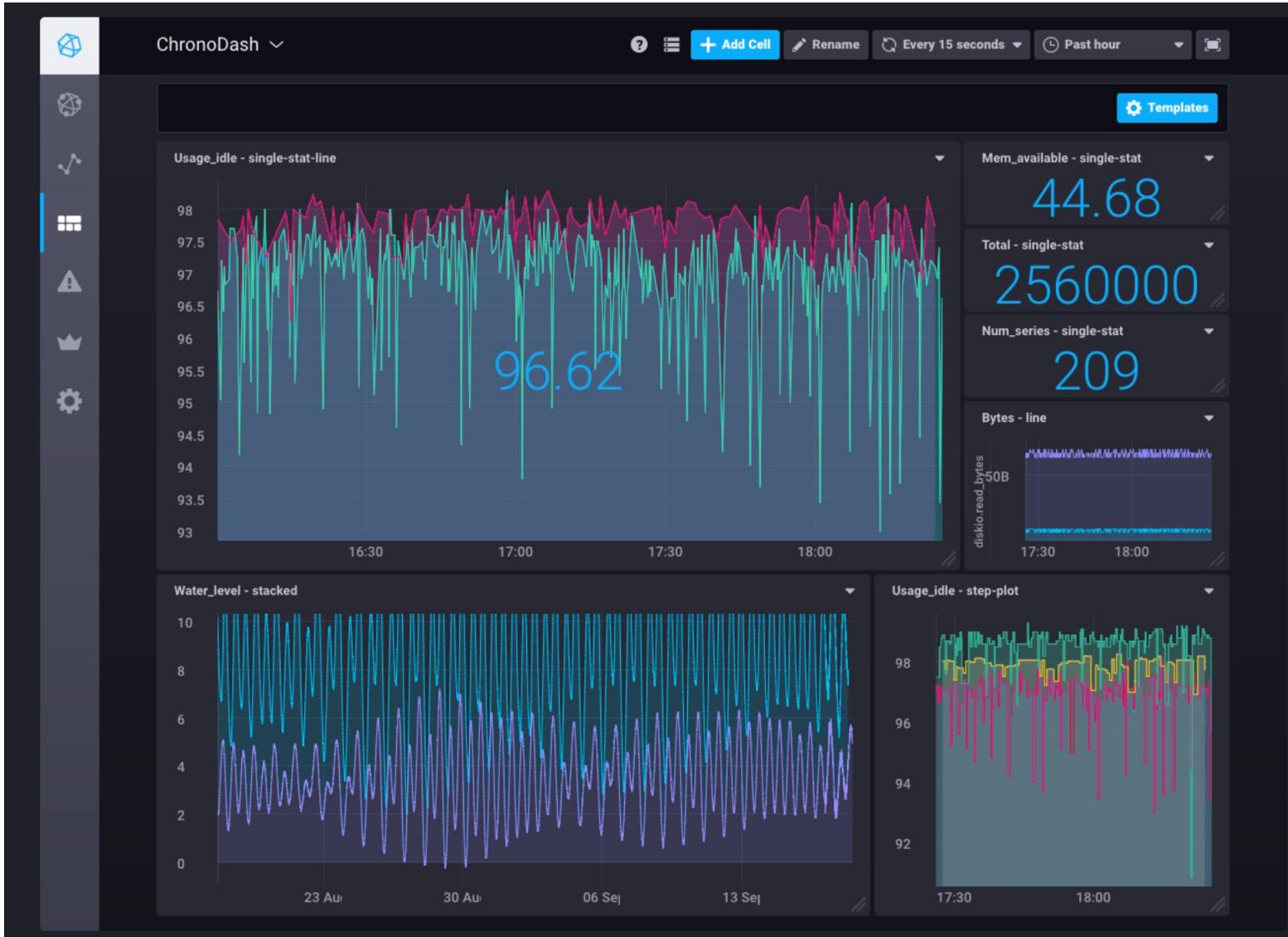


# Monitoring Data Pipelines

- Application / service / network configuration errors
- Balance flexibility with restrictions aimed at encouraging users to do the right thing
- Sandbox testing/ Automated unit testing to identify edge cases
- Automated rollout / rollback of code changes (DevOps)
- Monitoring / Dashboards / Telemetry for error detection, performance tracking



# Dashboards – cont.



```
$ conda install dash
```

The following packages will be downloaded:

package	build		
brotli-python-1.0.9	py39hfd1d529_7	736 KB	conda-forge
dash-2.3.1	pyhd8ed1ab_0	7.5 MB	conda-forge
flask-compress-1.11	pyhd8ed1ab_0	14 KB	conda-forge
Total:		8.3 MB	

The following NEW packages will be INSTALLED:

brotli-python	conda-forge/osx-64::brotli-python-1.0.9-py39hfd1d529_7
dash	conda-forge/noarch::dash-2.3.1-pyhd8ed1ab_0
flask-compress	conda-forge/noarch::flask-compress-1.11-pyhd8ed1ab_0



# \$ conda install jupyter-dash

If you prefer using Jupyter Notebooks or Jupyter Lab

The following packages will be downloaded:

package	build		
ansi2html-1.7.0	py39h6e9494a_1	39 KB	conda-forge
jupyter-dash-0.4.2	pyhd8ed1ab_1	20 KB	conda-forge
retrying-1.3.3	py_2	11 KB	conda-forge
Total:			71 KB

The following NEW packages will be INSTALLED:

ansi2html	conda-forge/osx-64::ansi2html-1.7.0-py39h6e9494a_1
jupyter-dash	conda-forge/noarch::jupyter-dash-0.4.2-pyhd8ed1ab_1
retrying	conda-forge/noarch::retrying-1.3.3-py_2



# Dash installation verification: \$ conda list dash

```
(base) [512 uranus:dash] conda list dash
packages in environment at /Users/rachlin/opt/anaconda3:
#
Name Version Build Channel
dash 2.3.1 pyhd8ed1ab_0 conda-forge
jupyter-dash 0.4.2 pyhd8ed1ab_1 conda-forge
```

Also: **conda list plotly** (plotly and plotly express already installed)

```
(base) [513 uranus:dash] condalist plotly
plotly 5.5.0 pyhd8ed1ab_0 conda-forge
plotly_express 0.4.1 py_0 conda-forge
```



# Dash HTML Components

```
from dash import html

html.Div([
 html.H1('Hello Dash'),
 html.Div([
 html.P('Dash converts Python classes into HTML'),
 html.P("This conversion happens behind the scenes by Dash's JavaScript front-end")
])
])
```

which gets converted (behind the scenes) into the following HTML in your web-app:

```
<div>
 <h1>Hello Dash</h1>
 <div>
 <p>Dash converts Python classes into HTML</p>
 <p>This conversion happens behind the scenes by Dash's JavaScript front-end</p>
 </div>
</div>
```



# Dash Core Components

## Dropdown

```
from dash import Dash, html, dcc

app = Dash(__name__)

app.layout = html.Div([
 dcc.Dropdown(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
 app.run_server(debug=True)
```

A screenshot of a Dash application's user interface. It shows a dropdown menu with three options: 'New York City', 'Montréal', and 'San Francisco'. The option 'Montréal' is currently selected, indicated by it being bolded and highlighted with a blue border. There are standard UI controls like a downward arrow icon and an 'X' button to close the menu.

# Dash Core Components → localhost:8050

## Slider

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
 dcc.Slider(-5, 10, 1, value=-3)
])

if __name__ == '__main__':
 app.run_server(debug=True)
```



# Dash Core Components

## Radio Items

```
from dash import Dash, dcc, html

app = Dash(__name__)

app.layout = html.Div([
 dcc.RadioItems(['New York City', 'Montréal', 'San Francisco'], 'Montréal')
])

if __name__ == '__main__':
 app.run_server(debug=True)
```



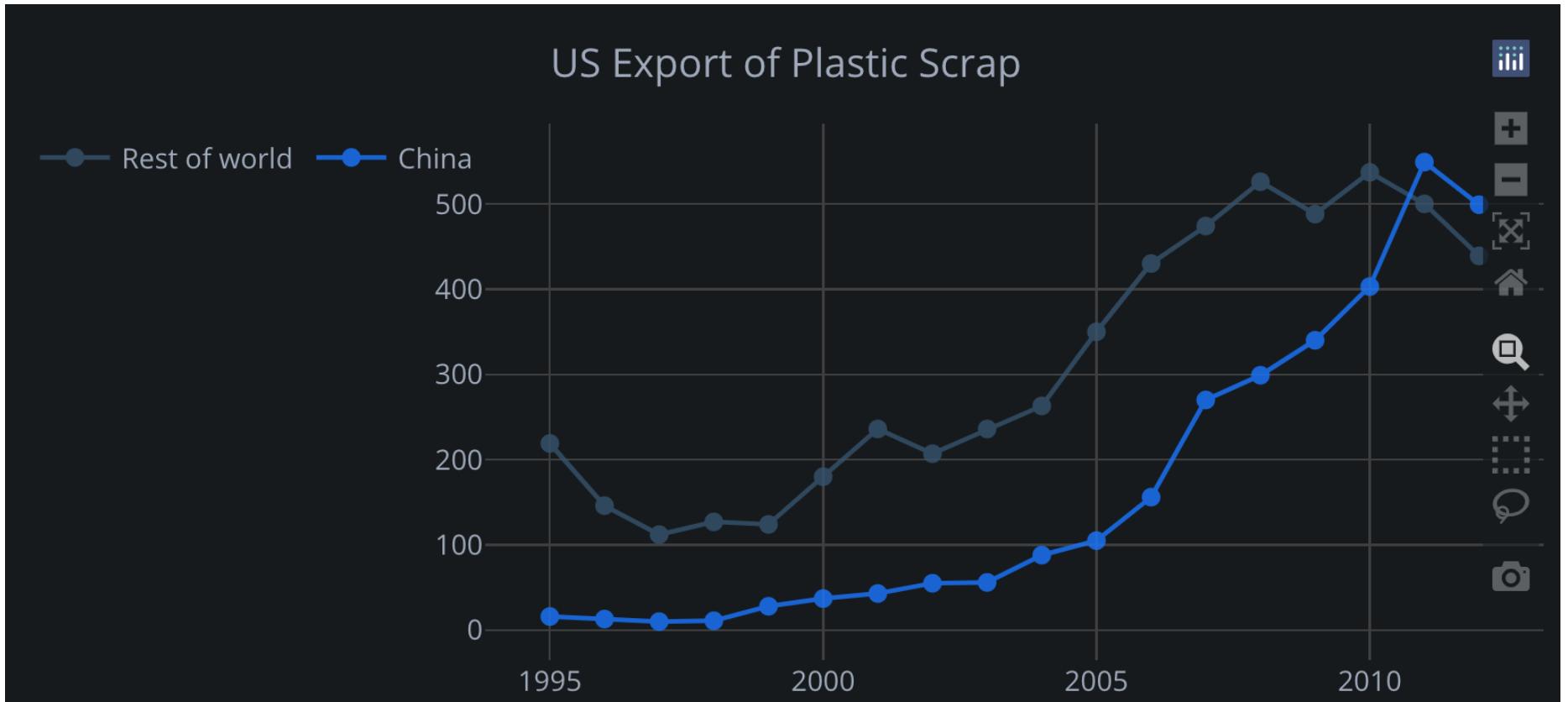
- New York City
- Montréal
- San Francisco



# Graphs! (Including our friend, the SanKey Diagram)

## Graphs

The `Graph` component shares the same syntax as the open-source `plotly.py` library. View the [plotly.py docs](#) to learn more.





## Live Updating Components

### The `dcc.Interval` component

Components in Dash usually update through user interaction: selecting a dropdown, dragging a slider, hovering over points.

If you're building an application for monitoring, you may want to update components in your application every few seconds or minutes.

The `dcc.Interval` element allows you to update components on a predefined interval. The `n_intervals` property is an integer that is automatically incremented every time `interval` milliseconds pass. You can listen to this variable inside your app's `callback` to fire the callback on a predefined interval.

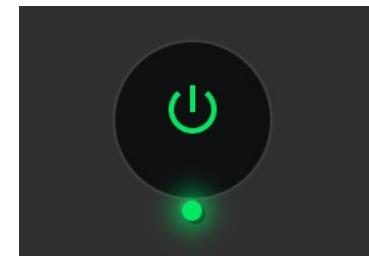
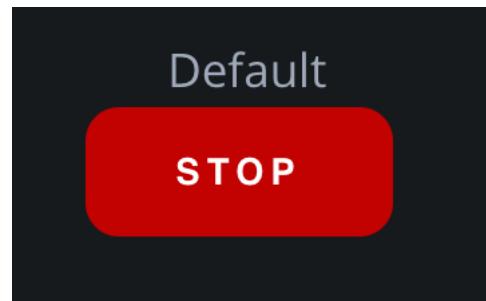
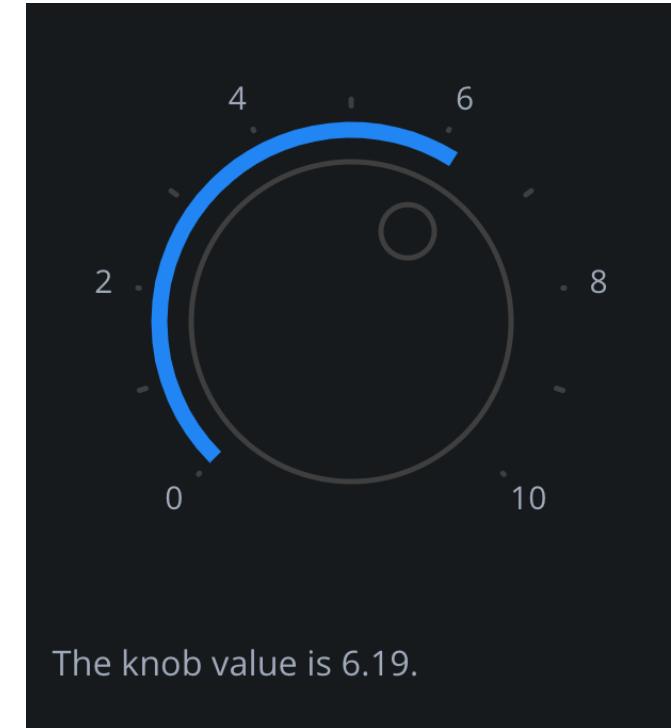
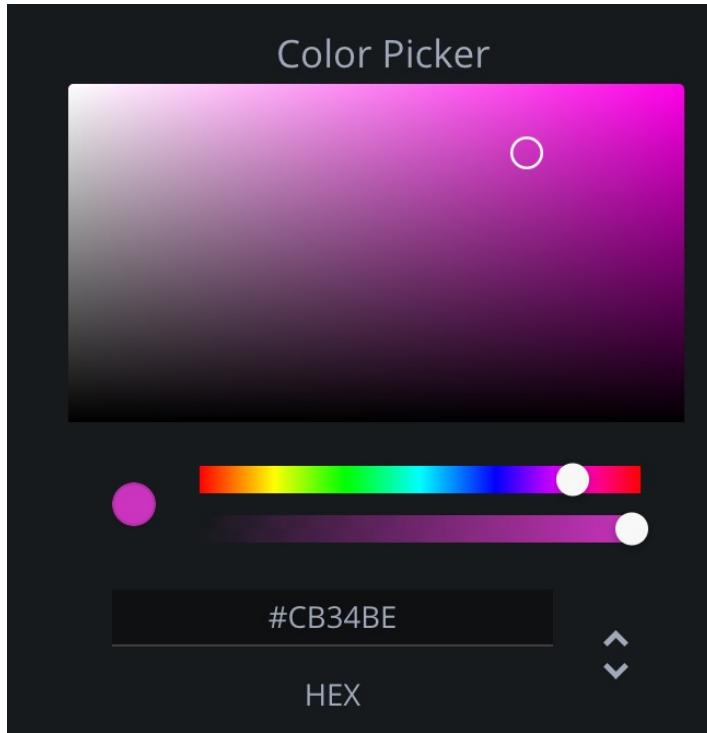
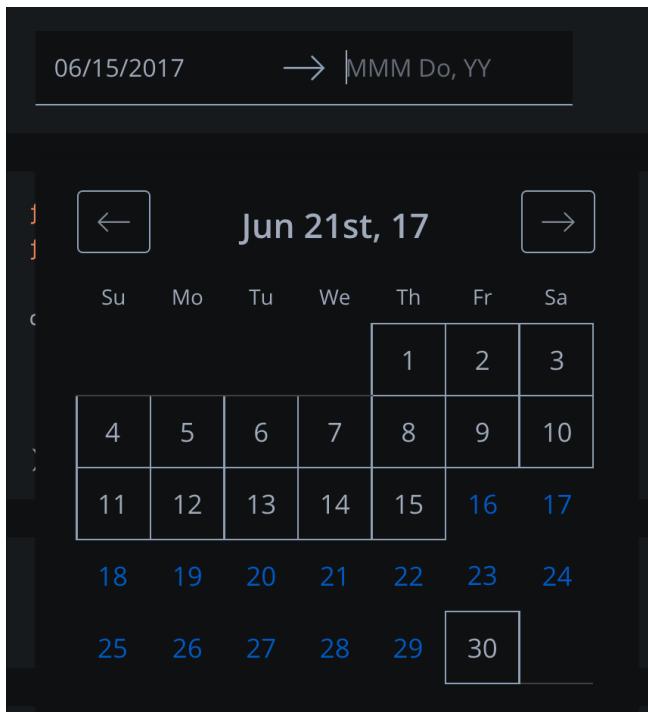


# Dash Tables

♦	index	♦	continent	♦	country	♦	gdpPercap	♦	lifeExp	♦	pop
	1		Asia		Afghanistan		974.5803384		43.828		31889923
	2		Europe		Albania		5937.029525999999		76.423		3600523
	3		Africa		Algeria		6223.367465		72.301		33333216
	4		Africa		Angola		4797.231267		42.731		12420476
	5		Americas		Argentina		12779.37964		75.32		40301927



# Dash DAQ: Web development framework



# Callbacks

Callbacks make your dashboards interactive

```
from dash import Dash, dcc, html, Input, Output

app = Dash(__name__)
app.layout = html.Div([
 dcc.Dropdown(['NYC', 'MTL', 'SF'], 'NYC', id='demo-dropdown'),
 html.Div(id='dd-output-container')
])

@app.callback(
 Output('dd-output-container', 'children'),
 Input('demo-dropdown', 'value')
)
def update_output(value):
 return f'You have selected {value}'

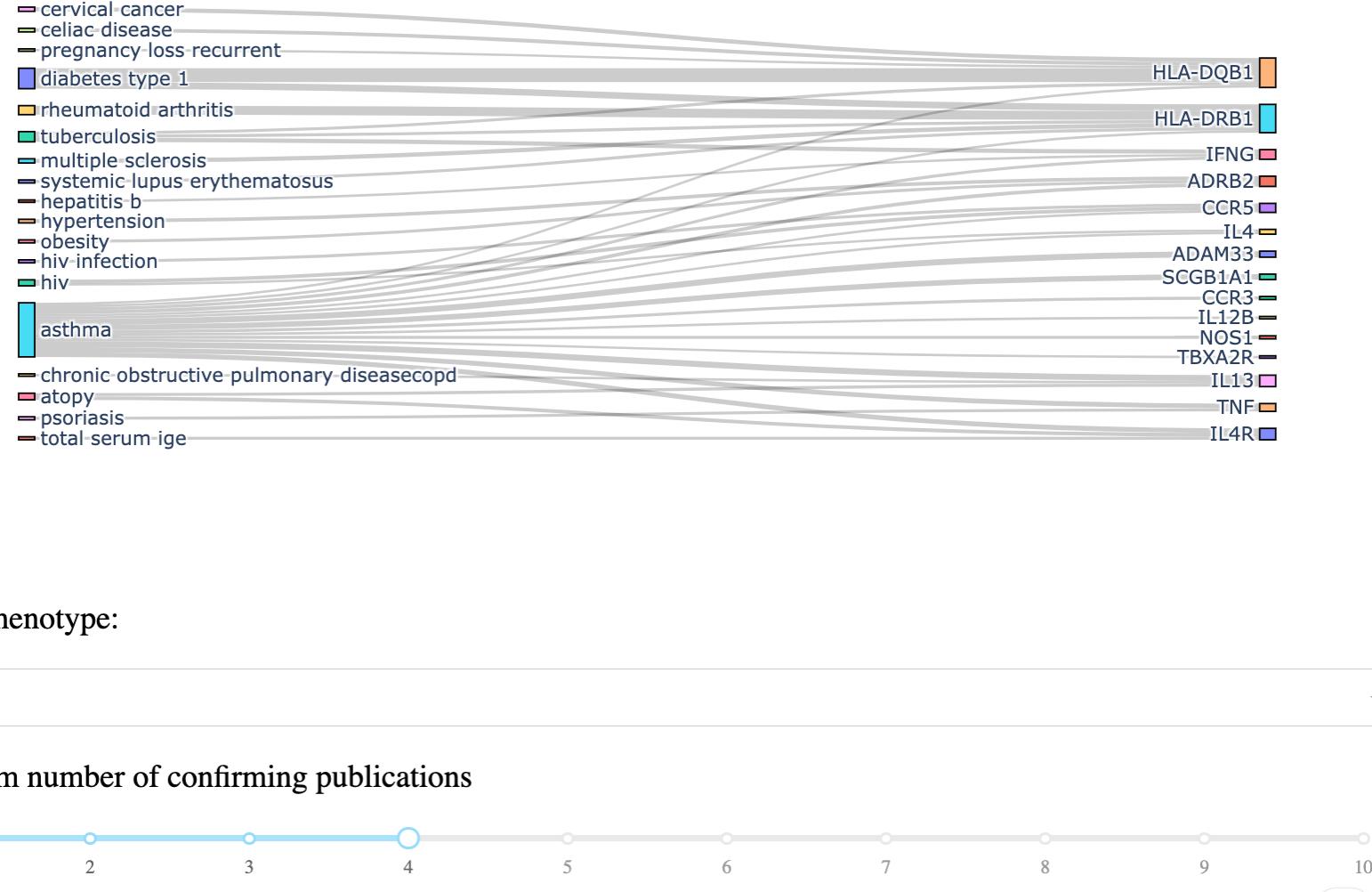
if __name__ == '__main__':
 app.run_server(debug=True)
```

NYC

You have selected NYC



# Putting it all together....



An interactive database-driven web application for visualizing gene-disease associations and identifying related phenotypes based on gene overlap.



# Array Processing with Numpy

---

(DS2500 Review)



Northeastern University

# Types of Arrays

Type	Description	Advantage	Disadvantage	
Lists	Standard built-in python list collection	Simplicity Non-homogeneous	Slower performance Limited functionality	[[1, 2, 3], [4, 5, 6]]
ndarray (NumPy)	High-performance array module for scientific applications.	High performance Flexible operations	Homogeneous values Numeric indexing only	array([1,2,3])
Series (Pandas)	Enhanced one-dimensional array	Custom indexing Missing-value support Many capabilities	Complexity	Wally 87 Eva 100 Sam 94 dtype: int64
DataFrame (Pandas)	Enhanced two-dimensional array (e.g., “tables”)	Flexible support for manipulating tabular data: filtering, sorting	Complexity	A B C 0 10 True 5.4 1 15 False 8.8 2 13 False 6.6

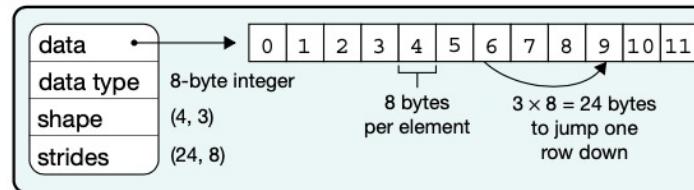


# Array operations

Nature | Vol 585 | 17 September 2020 | 357

## a Data structure

0	1	2
3	4	5
6	7	8
9	10	11



## b Indexing (view)

0	1	2
3	4	5
6	7	8
9	10	11

with slices

0	1	2
3	4	5
6	7	8
9	10	11

Slices are **start:end:step**,  
any of which can be left blank

## c Indexing (copy)

$x[1, 2] \rightarrow 5$  with scalars

$x[x > 9] \rightarrow [10, 11]$  with masks

$x[\boxed{0, 1}, \boxed{1, 2}] \rightarrow [x[0, 1], x[1, 2]] \rightarrow [1, 5]$  with arrays

$x[\boxed{1, 2}, \boxed{1, 0}] \rightarrow x[\boxed{1, 1}, \boxed{1, 0}, \boxed{2, 2}, \boxed{1, 0}] \rightarrow [4, 3]$  with arrays  
with broadcasting

## d Vectorization

0	1
3	4
6	7
9	10

+

1	1
1	1
1	1
1	1

$\rightarrow$

1	2
4	5
7	8
10	11

## e Broadcasting

0			0	0
3			3	6
6			6	12
9			9	18

$\times$

	1	2		

$\rightarrow$

0	0		
3	6		
6	12		
9	18		

## f Reduction

0	1	2		3
3	4	5		12
6	7	8		21
9	10	11		30

sum axis 1

sum axis 0

sum axis (0,1)

$\rightarrow$

18	22	26		66
----	----	----	--	----

**Fig. 1| The NumPy array incorporates several fundamental array concepts.**

**a**, The NumPy array data structure and its associated metadata fields. **b**, Indexing an array with slices and steps. These operations return a ‘view’ of the original data. **c**, Indexing an array with masks, scalar coordinates or other arrays, so that it returns a ‘copy’ of the original data. In the bottom example, an array is indexed with other arrays; this broadcasts the indexing arguments before performing the lookup. **d**, Vectorization efficiently applies operations to groups of elements. **e**, Broadcasting in the multiplication of two-dimensional arrays. **f**, Reduction operations act along one or more axes. In this example, an array is summed along select axes to produce a vector, or along two axes consecutively to produce a scalar. **g**, Example NumPy code, illustrating some of these concepts.

## g Example

In [1]: import numpy as np

In [2]: x = np.arange(12)

In [3]: x = x.reshape(4, 3)

In [4]: x

Out [4]:

array([[ 0, 1, 2],  
 [ 3, 4, 5],  
 [ 6, 7, 8],  
 [ 9, 10, 11]])

In [5]: np.mean(x, axis=0)

Out [5]: array([4.5, 5.5, 6.5])

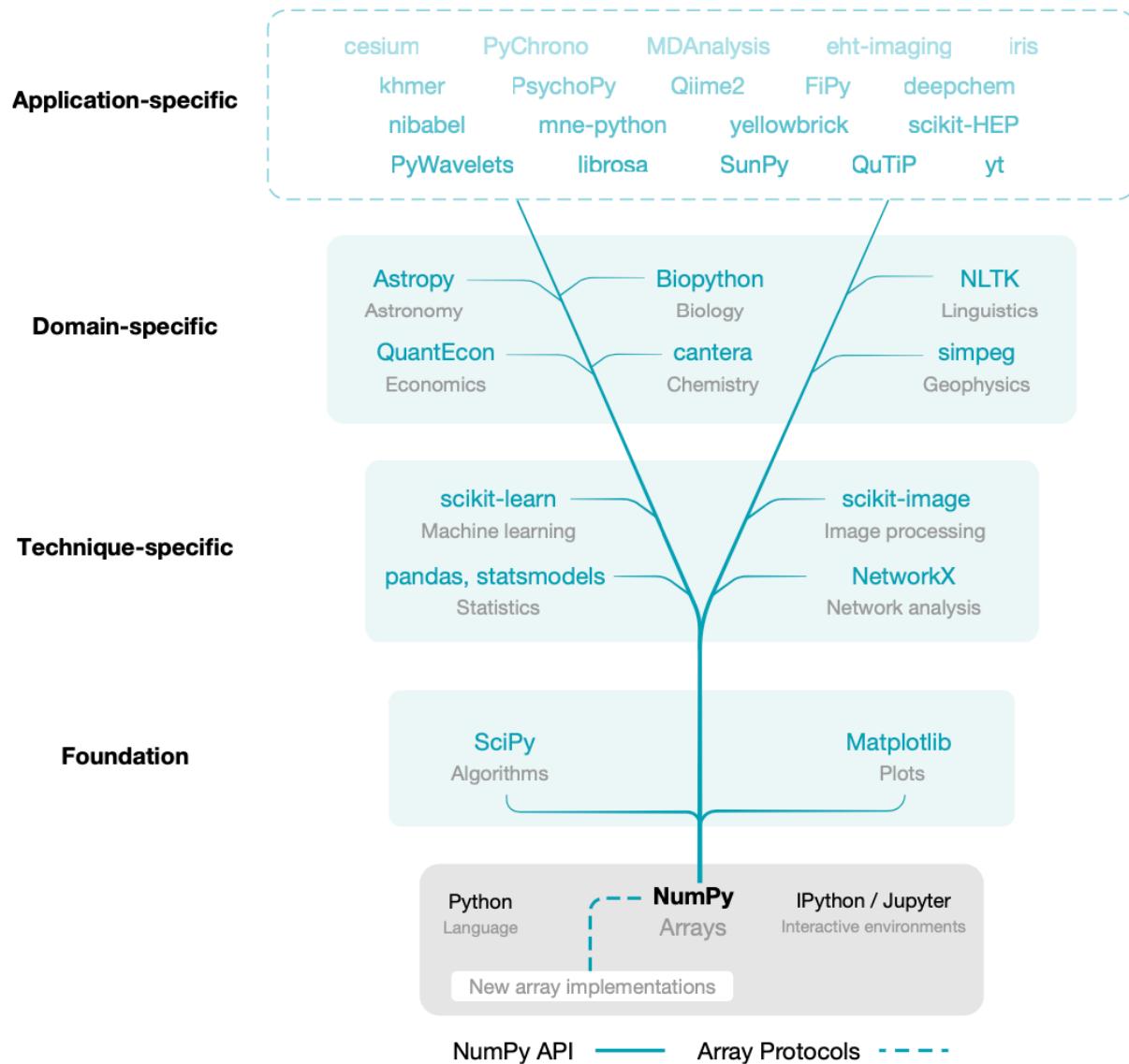
In [6]: x = x - np.mean(x, axis=0)

In [7]: x

Out [7]:

array([[-4.5, -4.5, -4.5],  
 [-1.5, -1.5, -1.5],  
 [ 1.5, 1.5, 1.5],  
 [ 4.5, 4.5, 4.5]])

# NumPy is the base of the scientific Python ecosystem



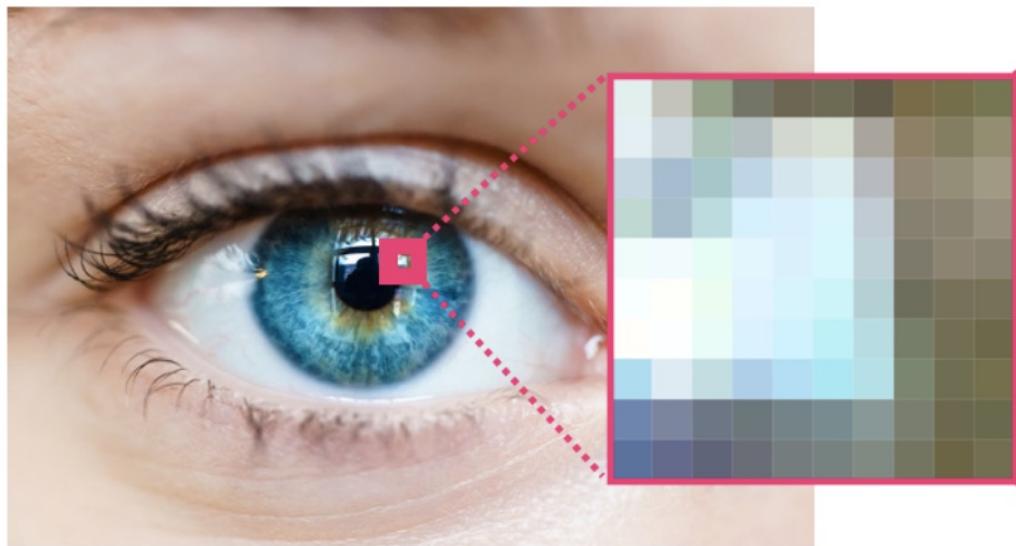
Nature | Vol 585 | 17 September 2020 | 357



**Fig. 2 | NumPy is the base of the scientific Python ecosystem.** Essential libraries and projects that depend on NumPy's API gain access to new array implementations that support NumPy's array protocols (Fig. 3).

# NumPy Image Analysis

If the image is colored, then each pixel is represented by three numbers - a value for each of red, green, and blue. In that case we need a 3rd dimension (because each cell can only contain one number). So a colored image is represented by an ndarray of dimensions: (height x width x 3).



233	188	137	96	90	95	63	73	73	82	
237	202	159	120	105	110	88	107	112	121	109
226	191	147	110	101	112	98	123	110	119	142
221	191	176	182	203	214	169	144	133	145	155
185	160	161	184	205	223	186	137	147	161	140
181	174	189	207	206	215	194	136	142	151	115
246	237	237	231	208	206	192	122	143	144	87
254	254	241	224	199	192	181	99	122	117	133
239	248	232	207	187	182	184	110	114	110	111
193	215	193	167	158	164	181	114	112	111	74
113	119	110	111	113	123	135	120	108	106	107
93	97	91	103	107	111	122	112	104	114	74
										82
										113



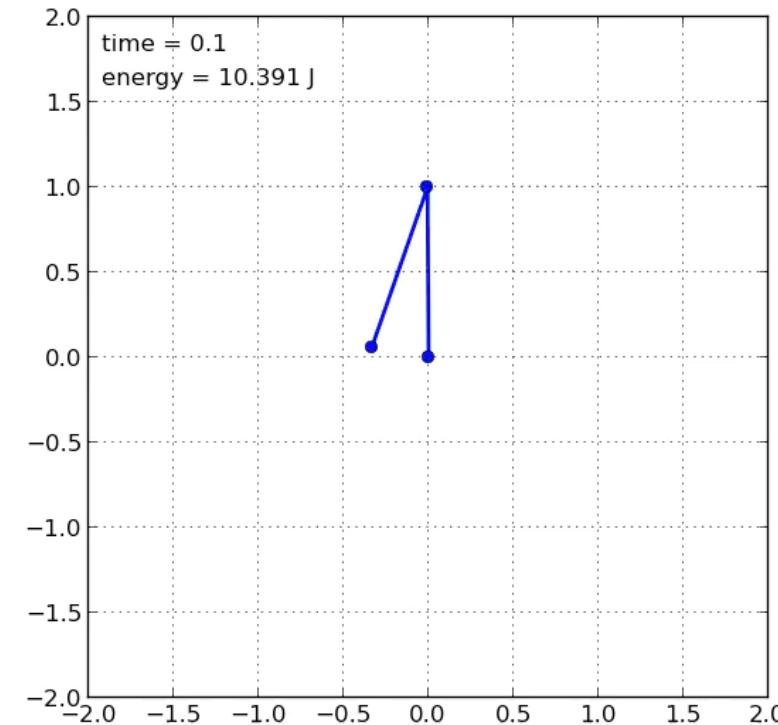
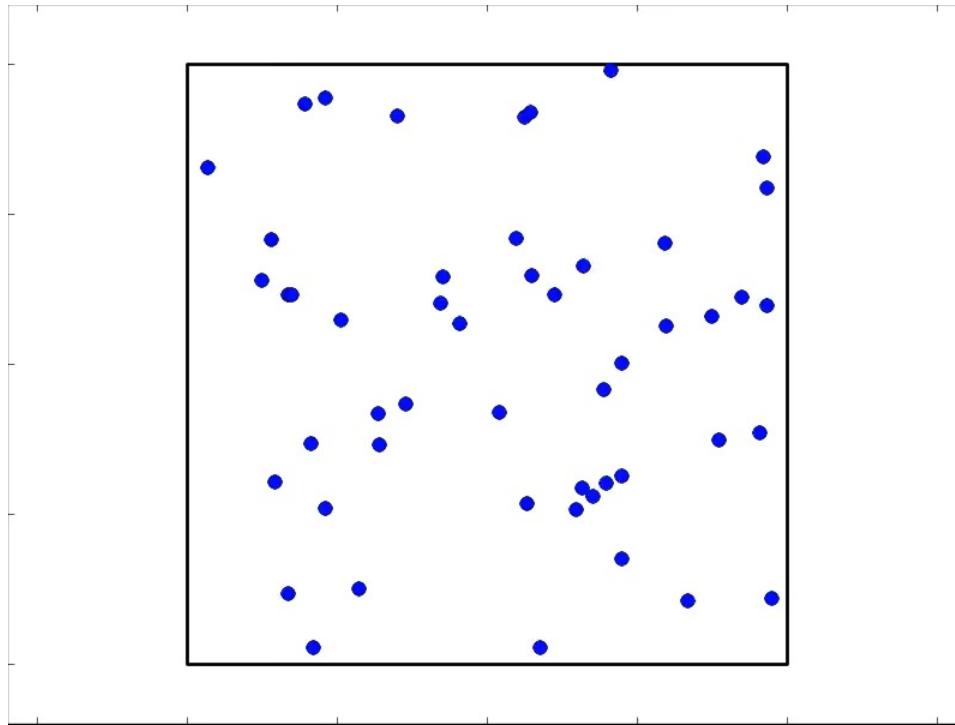
# Animation

---

Created animated visuals using Matplotlib



# Example Animations



Source: <https://jakevdp.github.io/blog/2012/08/18/matplotlib-animation-tutorial/>



Northeastern University

# FuncAnimation

---

## matplotlib.animation.FuncAnimation

```
class matplotlib.animation.FuncAnimation(fig, func, frames=None, init_func=None,
fargs=None, save_count=None, *, cache_frame_data=True, **kwargs) [source]
```

Makes an animation by repeatedly calling a function *func*.

 Note

You must store the created Animation in a variable that lives as long as the animation should run. Otherwise, the Animation object will be garbage-collected and the animation stops.

**fig** : plt.figure()

**func** : The animation function – called each frame

**frames** : The number of frames

**fargs** : Additional arguments to *func* (if needed)

**interval** : Time (ms) between frames



# Artificial Life

---

Life as it could be



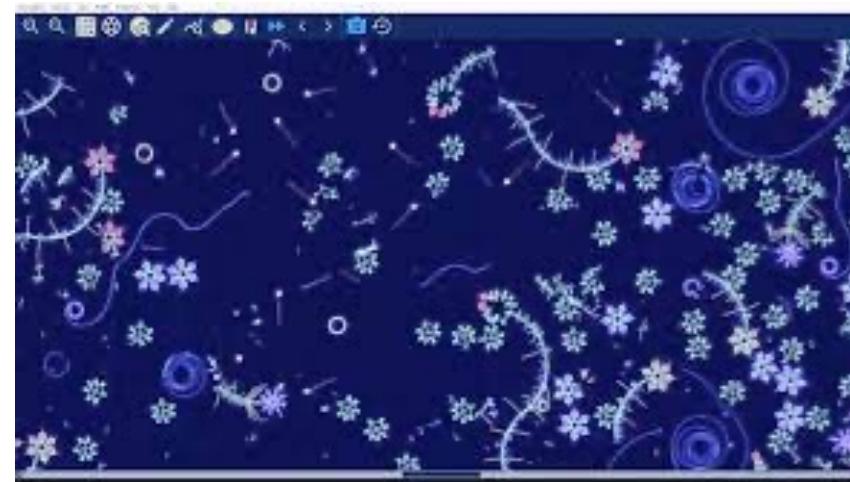
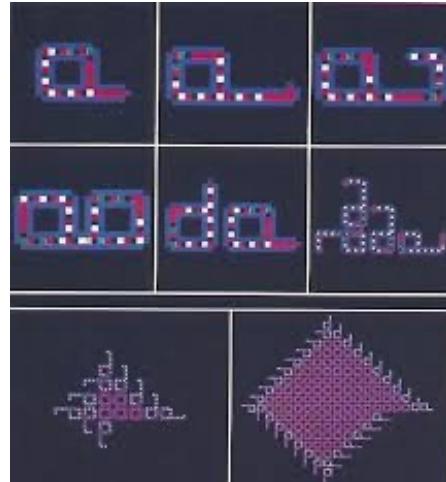
Northeastern University

# Artificial Life Defined

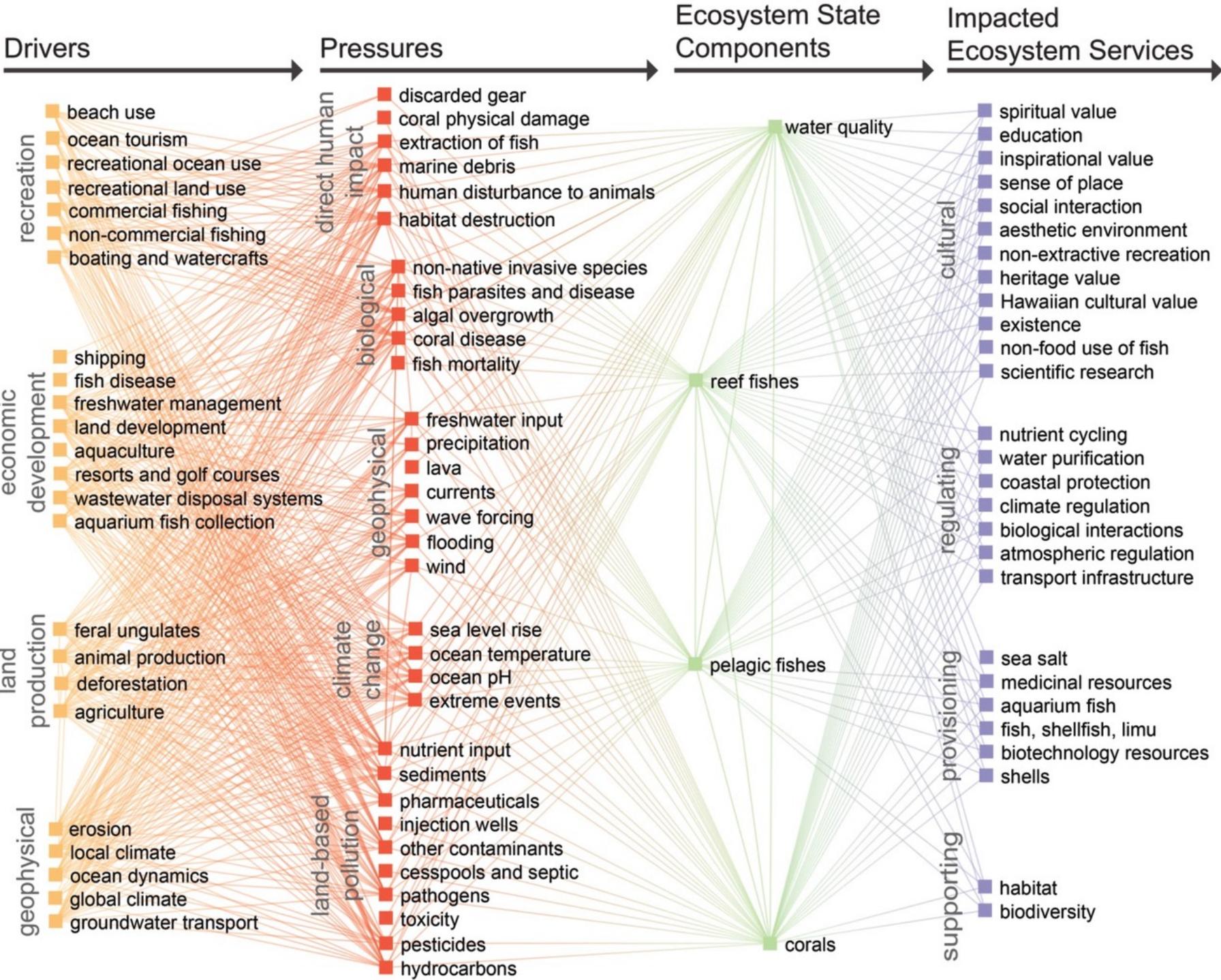
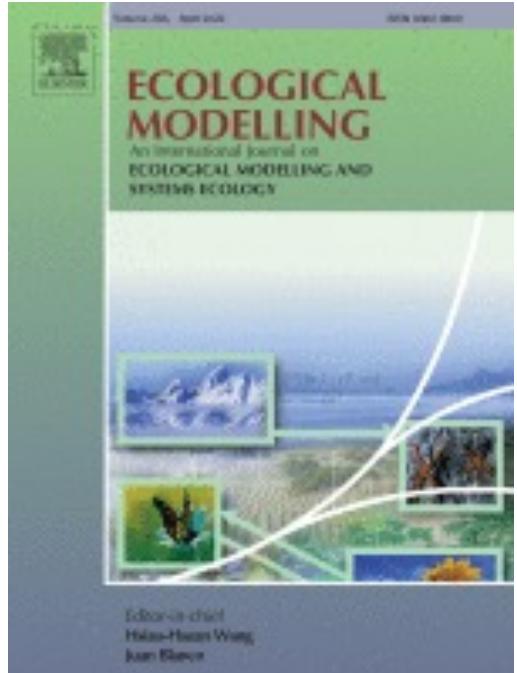
---

**Artificial life** (often abbreviated **ALife** or **A-Life**) is a field of study wherein researchers examine [systems](#) related to natural [life](#), its processes, and its evolution, through the use of [simulations](#) with [computer models](#), [robotics](#), and [biochemistry](#).

The discipline was named by [Christopher Langton](#), an American theoretical biologist, in 1986.



# Ecosystems



# Langton's Ants

Modeling ants responding to pheromone signals.

An example of how the repeated application of simple rules → complexity

**White:** Turn Right, **Red**, 1 Step.    **Red:** Turn Left, **White**, 1 Step

