Live Monitoring and Analysis of PyPI Downloads

Nitin Reddy, Ravi Vasani, Xuan Wang
Instructor : Prof. James Abello
Computer Science Department
Rutgers University
Piscataway, NJ, USA

***Abstract*— Considering the popularity and increasing usage of python packages this project shows live monitoring and analysis of package downloads. The project is useful for developers , researchers and even business entity to see what the world is using or what is the trend. The dashboard contains statistics about the live data and also it shows analysis on historical data.**

I. Project Description

Since, The Python ecosystem is vast and far-reaching in both scope and depth. In this project we came up with an idea of analyzing the python package downloads on all over the world. This project involves data collection, data analysis, visual interaction and UI implementation for perfect dashboard. Here we intended to develop a interactive dashboard. For some time now PyPI have archived all of the logs generated by users. And not many have analyzed this logs.

The motive for the project is to build a dashboard that shows live streaming of the python package downloads from PyPI and monitor the trend or shift of packages that are currently being downloaded. The dashboard can be broadly divided into 2 categories - Live data and Historical Data.

Live Data : This part of dashboard demonstrated the streaming data and various metric such as number of downloads in particular region, rate of download and top packages being downloaded.

Historical Data : In addition to that analysis of historical data helps to get trends, information that reflects insights on the data.

This dashboard basically answers following questions

- How many packages are being downloaded in any Country X?
- What is the rate of download ?
- How many packages are being downloaded?
- What is the top downloaded package now and their distribution?
- How does the rate of download look like over time?
- For any period, any country what are the top packages downloaded?
- How does download of one package compare to other over time?

The prospective information that can be gained from the dashboard can help developers, maintainers or decision makers that uses python. They can see the trends, demands and understand the progress of technology across these categories-

- NLP
- Scientific
- GUI
- Networking
- Machine Learning
- Visualization
- Data Analysis
- Development
- Deep learning and many more.

The user can get information about usage, popularity and demand of different python frameworks.

The stumbling block for this project is the scale of data. Each data entry shows particular package downloaded at certain point of time in a region with system specification on which the request is made each second. To handle enormous amount of data we have used google big query API to fetch data that is updated every second on the servers.

To get insights from the data we had to try and come up with different visualisation that could represent the data better. Trying various visualisations helped us in finding some interesting insights.

The project has four stages: Gathering, Design, Infrastructure Implementation, and User Interface.

A. Stage1 - The Requirement Gathering Stage.

- The general Project description:

  The project is based on live monitoring and analysis of PyPI downloads and the purpose is to analyze python package downloads all over the world based on interactive dashboard that involves live and historical data. We have seen several important developments in python frameworks/packages and a dashboard like this can help a new user researcher or open source developer understand those trends and changes.

To make a visually attractive and interactive dashboard and to solve a problem we used data from python's software foundations that is linked with Google Big Query. Since PyPI stat attempts to operate within free tier of its hosted services. So, eventually aggregate data is only retained for 180 days.

To get a data for brief period of time and make some analysis we used API for Google Big Query to get aggregate data for packages.

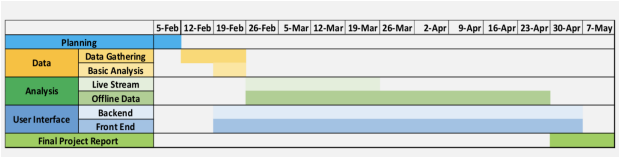
The dashboard was developed keeping the following users in mind -

- User Type 1 - Student:
  - Scenario 1 description: This user is student who typically use python for study.They can see live data streaming and get information about usage and popularity for packages. Even in our historical data analysis user can check shifts trends of approx. 100 packages.
    After adding date range and packages name The dashboard will show trend lines with important information with the plots.
- User Type 2 - Researcher:
  - Scenario 2 description: Other type of user can be a researcher that want to see a package in demand for a particular category and might want to shift according the popularity of the package.
- User Type 3 - Open Source Developer :
  - Scenario 3 description: Based on the analysis provided by by the dashboard developers can decide to contribute to the projects that are gaining popularity and have acceptance regionally.
- User Type 4 - Industry Application :
  - Scenario 4 description: Even this dash board can have business perspective based on the python package downloaded with particular system requirement in a region may give an idea of providing system compatibility to python version in a region.

In all of the above cases user will be provided with a login ID and password For all use cases data is fetched from Google Big Query.

- Project Time line
  - Week 1 - Planning the problem formalization and setting up the development environments. Did research on related problems and determine the corresponding user type and scenarios.
  - Week 2/3 - Basic Analysis , Requirement Analysis and Data gathering.
  - Week 3/4/5 - Implemented Live streaming and did historical with small chunk of data and implement the user interface.
  - Week 6/7/8/9/ - Performed analysis for live streaming, offline data and In addition to that we parallelly worked in building our UI for the dashboard.

Here is the detailed project time-line



For the division of labor, For the most part each of these tasks are undertaken in a collective fashion, from Nitin handling UI, plots and Xuan, Ravi supporting with data and plots for analysis and documentation.

B. Stage2 - The Design Stage.

- Short Textual Project Description.

In a requirement gathering part we tried to make two tabs in a dashboard that involves two parts

- Live Streaming Data
- Historical Data

One tab for live streaming would show live downloads in different regions and download rate etc. And Historical Data tab user can set date range and package name and see the trend lines for that packages. But when we grew monitoring over data and gained insights from the data, we extracted historical tab into two tabs

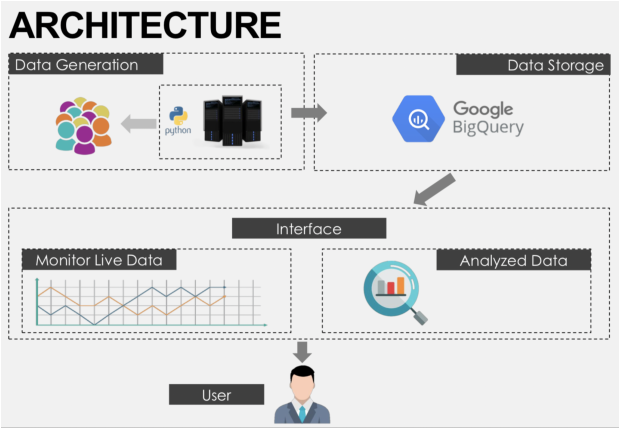
- One for specifically countries
- One for package trend lines

So finally, dashboard contained two tabs

- Live Data
- Countrywise Analysis
- Package Trendlines

- Conceptual Diagram

Conceptually, after data is generated from different end points all over the world , it is stored into Google big query and the interface shows live monitoring and analysis of data as show in architecture. The architecture of the application can be found below -



In our dashboard we have used tree map to build a plot that shows top 10 packages downloads per second.

- High Level Pseudo Code for the tree map

The algorithm recursively computes children's layout of the treemap rectangle. If adding a rectangle to the current row will improve layout contained by the row then the rectangle will be added to the current row else fix current row and start new row in the remaining subrectangle. The criteria of whether there is a improvement is the highest aspect ratio of listed rectangles(e.g.max($height/width$,$width/height$)) surpassing the old one which does not contain the processing rectangle.

Algorithm 1 Squarified Treemap Algorithm

Input:
$children$, A list of children for one rectangle;
row, A list of rectangles that is currently being laid out;
w, rectangle sizes in treemap;

Output:
Rectangles layout;
{$worst()$ function gives the highest aspect ratio of a list of rectangles; $++$ gives concatenate operation; $layoutrow()$ adds a new row of children to the rectangle; $width()$ returns the length of the shortest side of the remaining subrectangle in which the current row is placed;}

$c = head(children)$;
if $worst(row, w) \leq worst(row + +[c], w)$ **then**
  $squarify(tail(children), row + +[c], w)$
else
  $layoutrow(row)$;
  $squarify(children, [], width())$;
end if

C. Stage3 - The Implementation Stage.

For live monitoring and analysis, we have used approx. 57TB of data that involves 50-100 million entries in data set every day.

To accomplish this project we have used

- Python
- Dash
- Plotly
- Flask
- D3.js
- HTML
- CSS
- Javascript
- SQL Scripting
- Google BigQuery

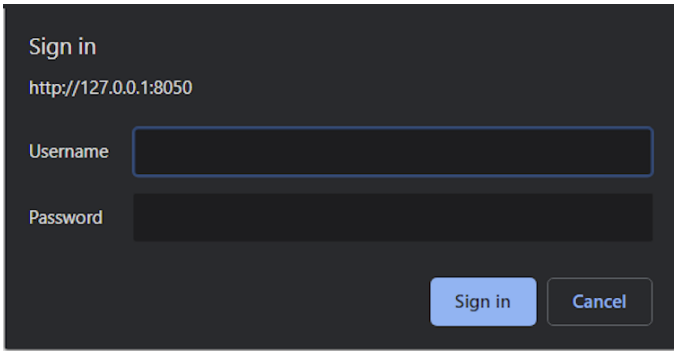
Also, we got some take away from Visual Analytics master book to use parallel coordinate plots for spatio temporal data when space and time combined can lead to insights.

- Data Description

Time Stamp	timestamp of download	2019-02-12 23:57:50 UTC
Country Code	Country	US
fileName	package name with version	mozfile-2.0.0-py2.py3-none-any.whl
fileproject	name of package	mozfile
file version	package version	2.0.0
installer name	Details of Installer	pip
installer version	Details of Installer version	19.0.2
Python Version	Details of python version	3.6
implementation name	Details of Implementation name	Cpython
Implementation version	Details of Implementation version	3.6
System Name	Details of System	Linux
System version	Version of system	18.04
CPU Details	CPU Details	x86

D. Stage4 - User Interface.

The user interface of this system is implemented on a Dash Python with Plotly. The users are supposed to easily straightforward login into dash board with credentials and start interacting with the dashboard.

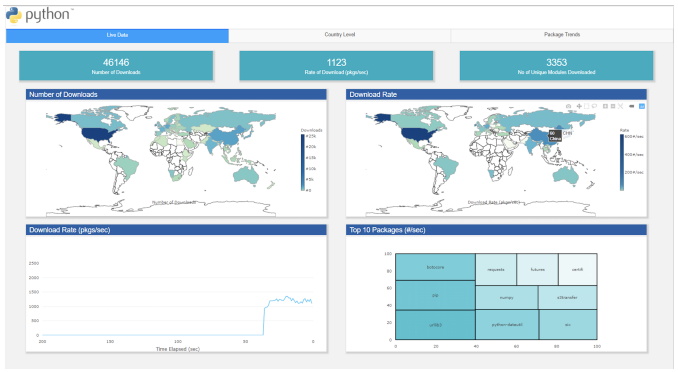


Different types of users share the same operation process by simply entering username and password getting a result image shown here.

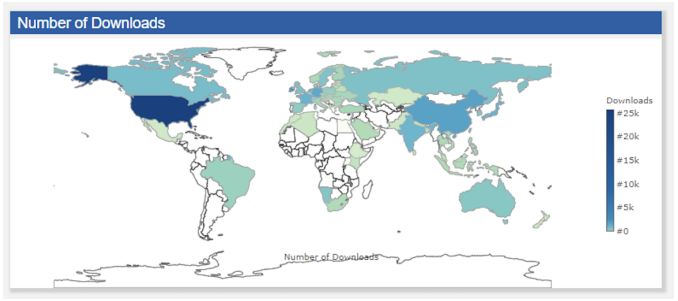
That includes live counts for

- Number Of Downloads
- Rate of Downloads (Packages per Second)
- Number of Unique Modules Downloaded

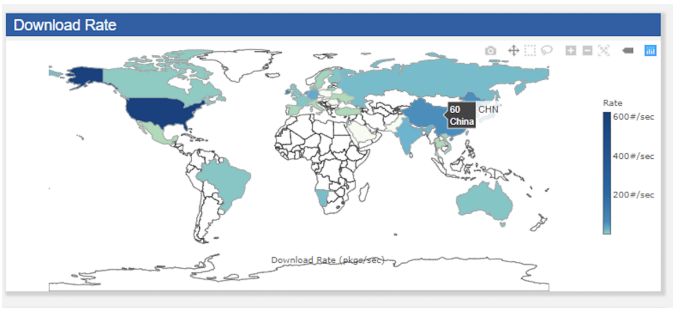
This page is over view of Live Streaming of Data



Choropleth for Number of Downloads



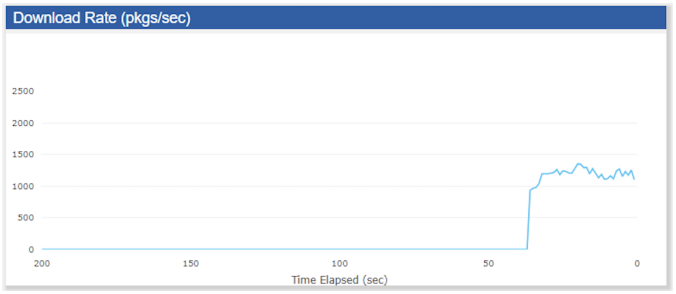
This plot show the current downloads in each and every country. So total number of downloads in region is showed based on density. The scale in the right of the plot shows the range of number of current downloads in each country.



Choropleth for Rate of Downloads

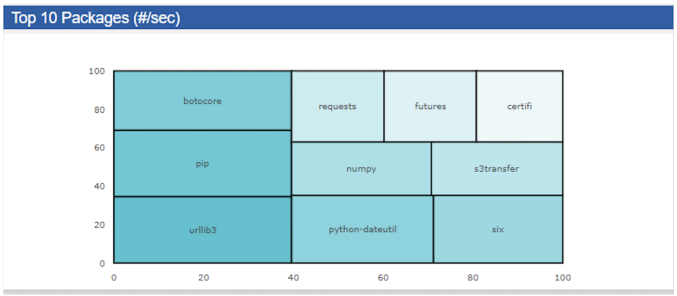
This plot show the current download rate in each and every country. So total number of downloads per second in region is showed based on density. The scale in the right of the plot shows the range of number of current download rate in each country.

Download Rate - Time Elapsed



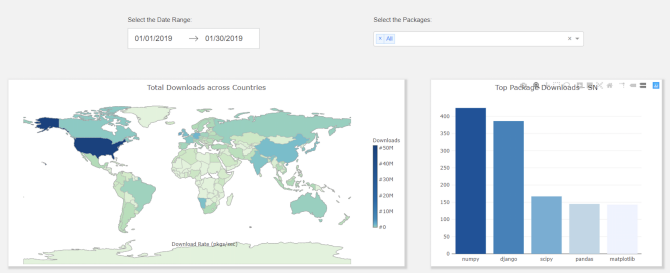
This plot show the current download rate with time elapsed in each and every country. Here, x axis shows time and y axis shows number of downloads at that particular time.

Tree Map - Top 10 Packages



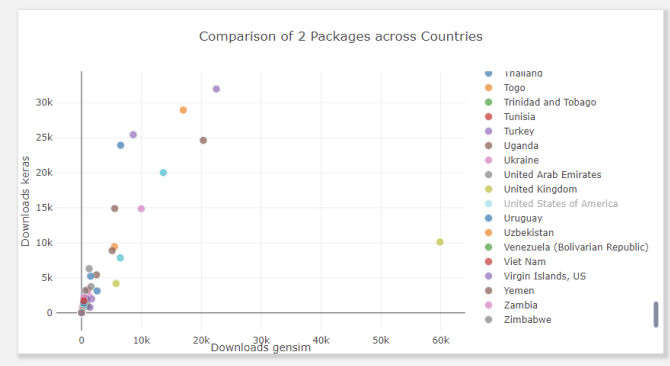
This plot shows Number of downloads per second of top 10 packages at that particular time. The algorithm for tree map is shown earlier in this report. Tree map is based on square of side 100.The size of each rectangle in the tree map shows the number of package in download per second. So, Size of the rectangle represents counts of the number of download per second and it updates every second.

Country level Analysis of Data is the second tab and it looks as below.



In this plot user can select Date range and the number select the packages from list that they want to analyze. In the left , the choropleth shows the total downloads of python packages in each country. Total number of downloads are adjusted to color scale in the choropleth. In addition to that while user hover over specific country the bar chart in the right displays number of top 5 package downloaded in that country.

Scattered Plot - For Comparisons



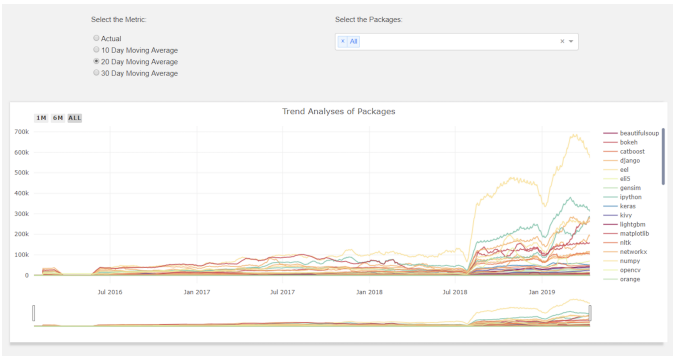
Above scatter plot is intended to show comparison of 2 packages across countries. Basically, User can select two packages from the list of packages in country wise analysis page. So, In this picture all of the countries are listed and user can see the comparison for usage and popularity of package in specific country.

Package Trends

In Package Trends tab user can select the package from package list and in the plot it can see the trend analysis over years.

In addition to that ,we also tried to include different metric to get insights from the data of relatively longer period of time.

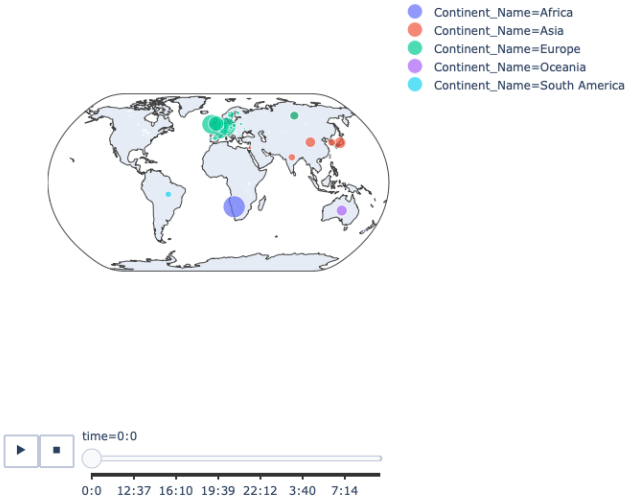
User can select a metric using radio button and the results are displayed according to the metric selected.



- Actual
- 10 Day Moving Average
- 20 Day Moving Average
- 30 Day Moving Average

Plot also gives user to select a time frame from bottom and select a packages to compare.

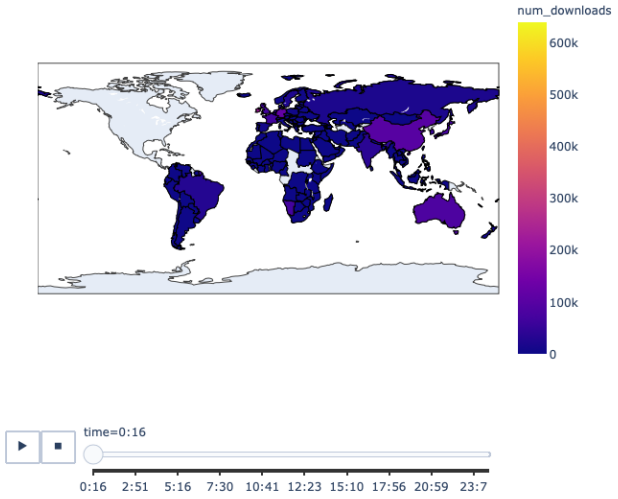
Also, using we tried to get some insights on a daily basis for package downloads. For that we have implemented geo-scatter plot and choropleth that shows number of downloads over a period of day in different countries.



- Findings

As shown in figure 1, A constant dip occurred once in week and then rises for rest of the week for all the packages.The reason behind this can be Some of python servers could have possible downtime in week or Application might have setup to update their packages every week.

Figure 2 shows that US undoubtedly dominated the downloads for any package.Reason could be most of the



application using python might have setup servers in US or when downloading show their location as US.

UK is more inclined towards Natural Language Processing than deep learning when compared with rest of the world.Reason -Since a part of EU, required to work with different languages spoken in EU.That is shown in figure 3.

According to figure 4 ,Seaborn was most preferred until March 2018, then plot.ly surpassed in total downloaded post that.

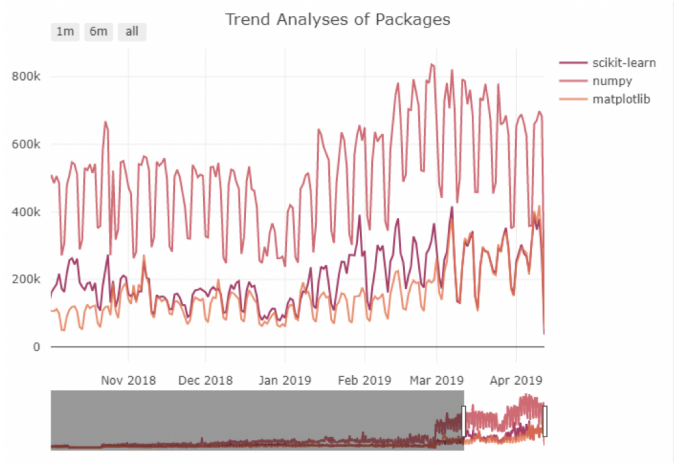


Fig. 1.

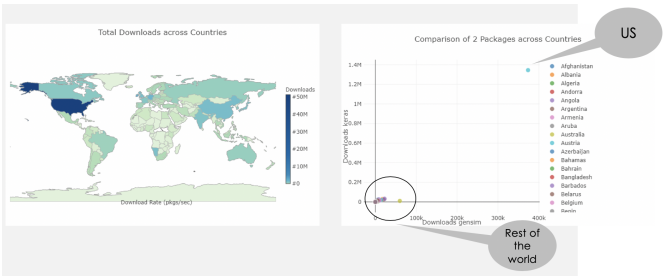


Fig. 2.

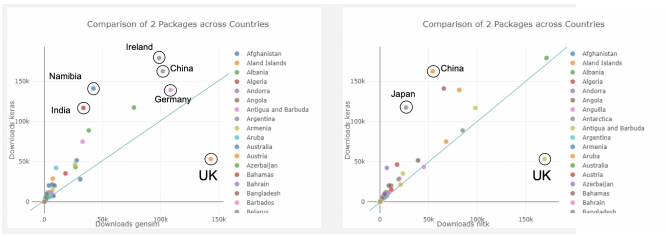


Fig. 3.

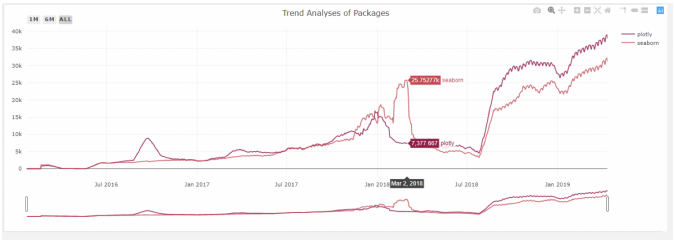


Fig. 4.

REFERENCES

[1]Data : https://bigquery.cloud.google.com/table/the-psf:pypi.downloads20190212?tab=preview

[2]API, https://pypistats.org/

[3]Dash, https://dash.plot.ly/

[4]Master Book, http://www.vismaster.eu/wp-content/uploads/2010/11/VisMaster-book-lowres.pdf

[5]NumFocus, https://numfocus.org/wp-content/uploads/2018/07/NumFOCUS-Corporate-Sponsorship-Brochure.pdf

[6] Data Visualization: A Successful Design Process â by Andy Kirk

[7] Guidelines for Using Multiple Views in Information Visualization - http://courses.ischool.berkeley.edu/i247/f05/readings/Baldonado_MultipleViews_AVI00.pdf

[8]Mark Bruls and Kees Huizing and Jarke van Wijk,'Squarified Treemaps', In Proceedings of the Joint Eurographics and IEEE TCVG Symposium on Visualization, press, 1999 pp. 33-42