SDS Capstone Data Report

DS-6011

Derek Banks - dmb3ey@virginia.edu

Camille Leonard – cvl7qu@virginia.edu

Shilpa Narayan – smn7ba@virginia.edu

Nick Thompson – nat3fa@virginia.edu

Overview

Open-Source Software (OSS) is computer software with its source code shared with a license in which the copyright holder provides the rights to study, change, and distribute the software to anyone and for any purpose. Examples include Linux operating system, Apache server software, and R statistical programming software.

Despite its extensive use, reliable measures of the scope and impact of OSS are scarce. The creation and use of OSS highlight an aspect of technology diffusion and flow that is not captured in science and technology indicators.

Our group focused on conducting network modeling of OSS licensed Python packages from pypi.org. These are the packages that are installed through package managers such as pip. Main areas of interest include network modeling of package dependencies and network modeling of contributors.

Specifically, we aim to create a network of package contributors where package contributor data will be used as edge weight. We will also create a network of packages which include linkages for dependencies. The measure of impact for packages will be number of downloads.

Data Collection and Audit

While the code for the PyPi packages is located throughout the pypi.org site, metadata and statistics for the packages are accessible through Google's Cloud BigQuery (GCB) API. In this paper, the PyPi package metadata table (distribution_metadata) and PyPi downloads (file_downloads) table were downloaded using the API.

To obtain project contributor information, we joined the GCB metadata table with existing GitHub data obtained from our project sponsor. These GitHub data include commits activity for all repositories with an Open-Source Initiative-approved license from January 2009-December 2019. The result of the cleaned and merged PyPi metadata contained the necessary data to begin contributor network analysis.

For package downloads, the data for this project was limited to downloads occurring in the last year, defined as 01/01/2020 to 01/01/2021. Collection and processing were executed on University of Virginia's Rivanna high-performance computing system and the sponsor's PostgreSQL server. The code for data collection and processing can be found on our project's GitHub repository.

Google Big Query Collection

The data available through the API contains two tables of interest. The first data is the distribution metadata table (referred to as distribution_metadata on GCB). This table contains information about each package such as package name, version, author, maintainer, dependencies, etc.

The second table is the file downloads table (referred to as file_downloads on GCB). It contains information about each package download for the subject time period. Our analysis aims to quantify the impact of OSS software packages. Therefore, we were interested in the number of downloads completed during the subject time period not information on each individual download. Data downloaded from the file_downloads table was aggregated by package name, version, and download country code to produce a download count. Data dictionaries can be found for each table in Appendix A.

All the data was collected from the GCB metadata table. The metadata was reduced to include only packages that were released. Preparing the data for the contributor network, the metadata table was reduced to those packages that are being hosted on GitHub with an OSS license. This was accomplished by selecting records which had a home page record pointing to a GitHub repository. We decided to limit the data to GitHub hosted projects because our primary aim was to create a contributor network based on GitHub data available from our sponsor.

name	version	author	author_email	license	home_page	dependency
RelStorage	2.1.1	[redacted]	[redacted]	ZPL 2.1	http://relstorage.readthedocs.io/	zope.interface
ae-kivy-user-prefs	0.1.21	[redacted]	[redacted]	OSI Approved :: GNU General Public License v3 ...	https://gitlab.com/ae-group/ae_kivy_user_prefs	coverage-badge ; extra == 'dev'
bio2bel-hgnc	0.3.0	[redacted]	[redacted]	MIT License	https://github.com/bio2bel/hgnc	click
amundsen-search	2.5.1	[redacted]	[redacted]	NaN	https://github.com/amundsen-io/amundsensearchl...	flask-cors (==3.0.8)
ccxt	1.42.80	[redacted]	[redacted]	MIT	https://ccxt.trade	flake8 (==3.7.9) ; extra == 'qa'
gsheetsdb	0.1.11	[redacted]	[redacted]	MIT	https://github.com/betodealmeida/gsheets-db-api	six
pidan	1.0.9.23	[redacted]	[redacted]	UNKNOWN	UNKNOWN	NaN
rexmas-users-service	0.1.17	[redacted]	[redacted]	NaN	NaN	pylint (>=2.6.0,<3.0.0)
cdk-remote-stack	0.1.63	[redacted]	[redacted]	Apache-2.0	https://github.com/pahud/cdk-remote-stack.git	aws-cdk.aws-logs (<2.0.0,>=1.62.0)
inforion	2.11.35	[redacted]	[redacted]	NaN	https://github.com/Fellow-Consulting-AG/inforion/	progressbar (==2.5)

Metadata Sample Snapshot – author identification details removed for privacy

GitHub Data (Contributor Information)

The Social and Decision Analytics Department (SDAD) of UVA Biocomplexity Institute previously gathered publicly available data on development activity of individual GitHub hosted projects and their contributors. To begin, they used the Ruby Gem Licensee to classify the LICENSE file in each repository. This enabled them to select only packages with OSS approved licenses. Then the GHOST.jl package was used to collect and track GitHub user attributes and scrape all the commit history. This data was filtered to exclude any forked, mirrored, or archived repositories spanning GitHub's creation in 2008 through the end of 2019. Finally, the data was de-duplicated and filtered to exclude known bot accounts and commits merged from previous repositories. The produced commits activity dataset totaled 3,260,612 distinct contributors and 7,628,101 distinct repositories.

Additional work was conducted to classify users into countries using GHTorrent's July 2019 user data. The SDAD team expanded the GHOST.jl package to include a function that supplemented the GHTorrent's data with email information. An algorithm was developed to standardize the self-reported

city, state, country, email, and affiliation data into ISO-2 country codes. 732,636 distinct users were classified into countries in the final dataset which represented 22.4% of the total data.

The contributors data was made available to us through the SDAD's Postgres SQL database and merged with our PyPi metadata. See the next section for further details.

slug	committed_date	login	additions	deletions
sfischer13/python-arpa	2018-12-12 13:58:58-05:00	sfischer13	7	4
cms-sw/cmssw	2009-05-25 12:24:14-04:00	null	1	1
bccp/nbodykit	2017-09-21 11:38:54-04:00	nickhand	268	118
jmathai/elodie	2016-09-07 23:51:57-04:00	jmathai	8	1
pypa/setuptools	2017-09-29 10:27:34-04:00	jaraco	12	4
stsewd/bookpy	2016-10-28 21:25:44-04:00	stsewd	2	0
aioli-framework/aioli	2019-06-02 16:40:36-04:00	rbw	1	1
cms-sw/cmssw	2012-06-06 03:31:56-04:00	null	1	1
onelogin/python-saml	2019-01-17 22:23:20-05:00	cclauss	1	1
drdoctr/doctr	2016-08-18 01:17:02-04:00	asmeurer	1	1

GitHub Data Sample Snapshot

Merged Data

The home_page column in GCB metadata table was transformed to create a new column called slug. A GitHub slug is comprised of the GitHub URL followed by the user/organization and project names.

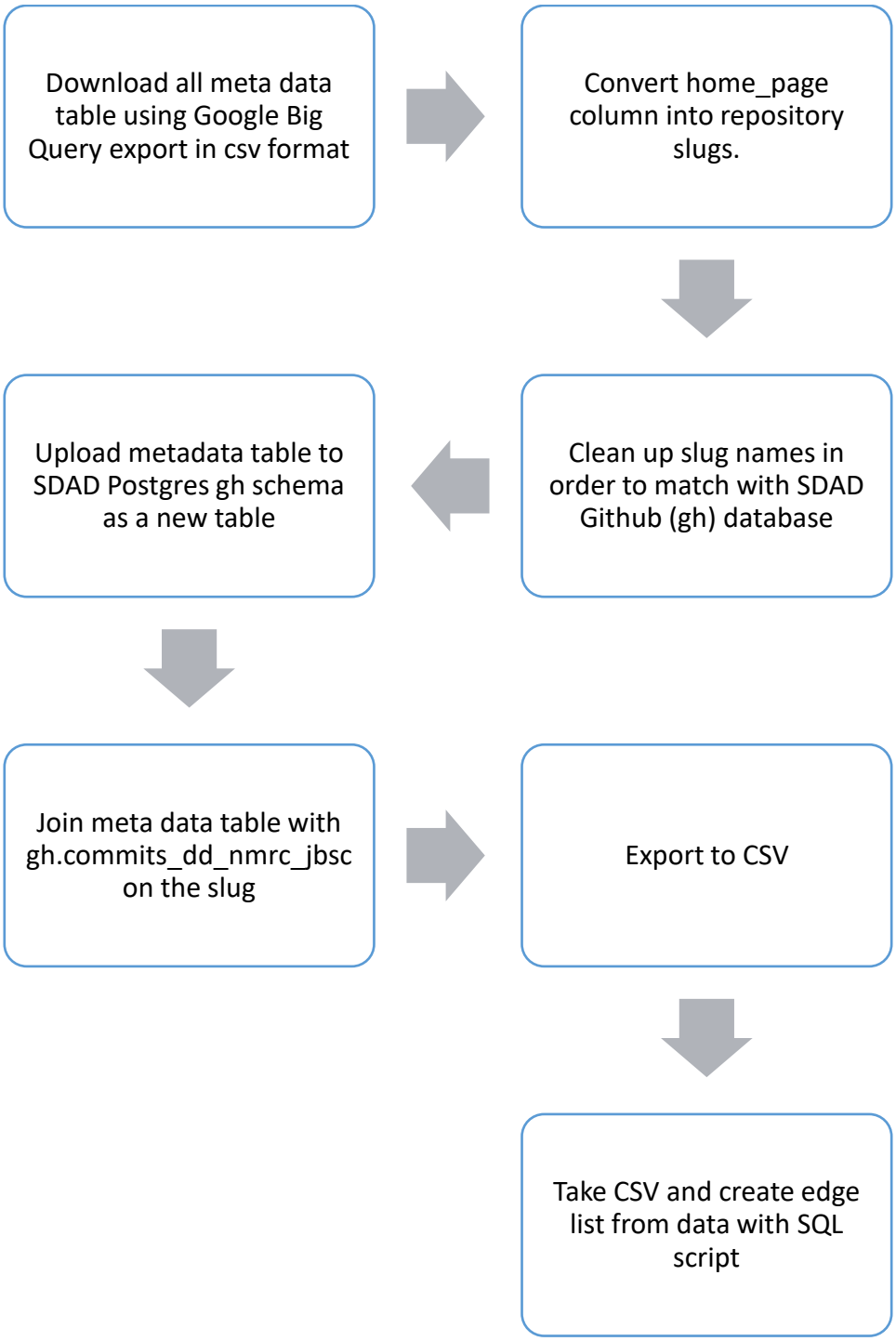
The dataset was the uploaded to the SDAD Postgres SQL server and merged with the existing GitHub data on the project slug. The result was a table with the PyPi package data and corresponding GitHub projects with contributor information. The resulting table was then saved to CSV file as the finalized dataset to create contributor edge lists.

ctr1	ctr2	repo_wts
pyup-bot	pyup-bot	1157
dependabot[bot]	dependabot[bot]	711
jaraco	jaraco	342
dependabot-preview[bot]	dependabot-preview[bot]	326
gitter-badger	gitter-badger	260
hugovk	hugovk	220
asottile	asottile	209
lnielsen	lnielsen	185
msabramo	msabramo	167
idlesign	idlesign	160

Contributor Edge list Sample Snapshot

An overview of the data cleaning and processing steps can be found below:

Contributor Data

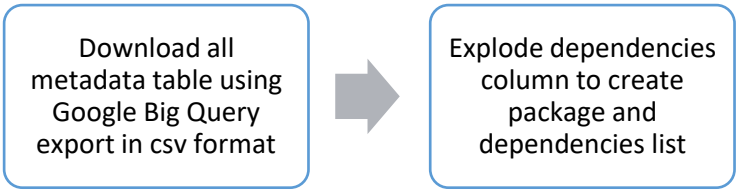


Package and Dependencies

The original metadata table was reduced to package name and its dependency with versions removed. The package name and dependency name were cleaned to have a consistent naming convention. For example, if the package name was pandas and if pandas is also used as a dependency by any other package, it was transformed consistently called pandas and not pandas-abc or pandas_123.

name	dependency_name
scaleogram	PyWavelets
rkstiff	none
gensim	Pyro4
system-query	pyudev
gemlog	sphinx
camerahub	django-autosequence
mypy-boto3	none
aegea	paramiko
bonsai-cli	websocket-client
text-diff	flake8-debugger

Dependency Edge list sample snapshot

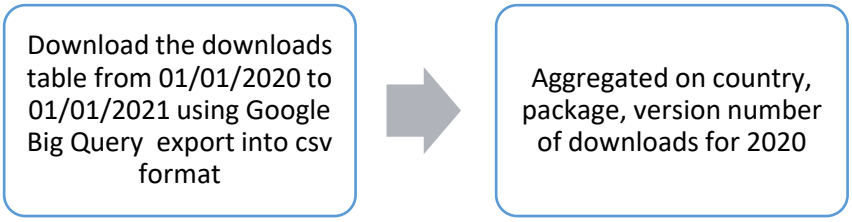


Downloads

The downloads table was queried to aggregate downloads by package name, version, and country between 01/01/2020 to 01/01/2021.

country_code	name	version	num_downloads
US	chardet	3.0.4	529606563
US	python-dateutil	2.8.1	506601529
US	six	1.15.0	425427437
US	pyasn1	0.4.8	423882729
US	s3transfer	0.3.3	413058876

File Downloads sample snapshot

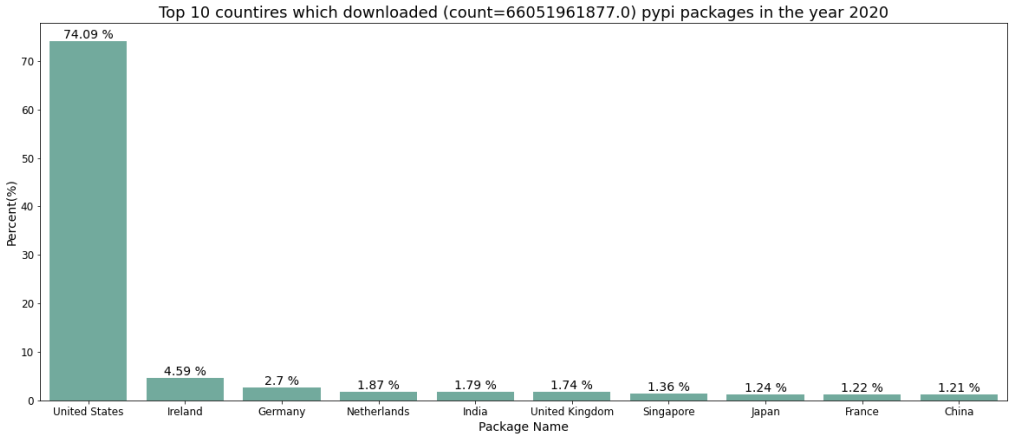


The main unit of observation was a Python package and a specific version of that package. Our finalized datasets totaled six files from data collection and processing with approximately 7GB of data.

File Name	Size	Row Count
PyPi_dependency_edgelist_112021.csv	620MB	20,640,255
PyPi_contrib_edgelist.csv	256MB	944,064
PyPi_downloads_365DAY_01012020.csv	2.9GB	85,397,267
PyPi_meta_all.csv	2.7GB	20,640,255
PyPi_meta_with_license_github_slugs.csv	1.8GB	10,687,591

Exploratory Analysis

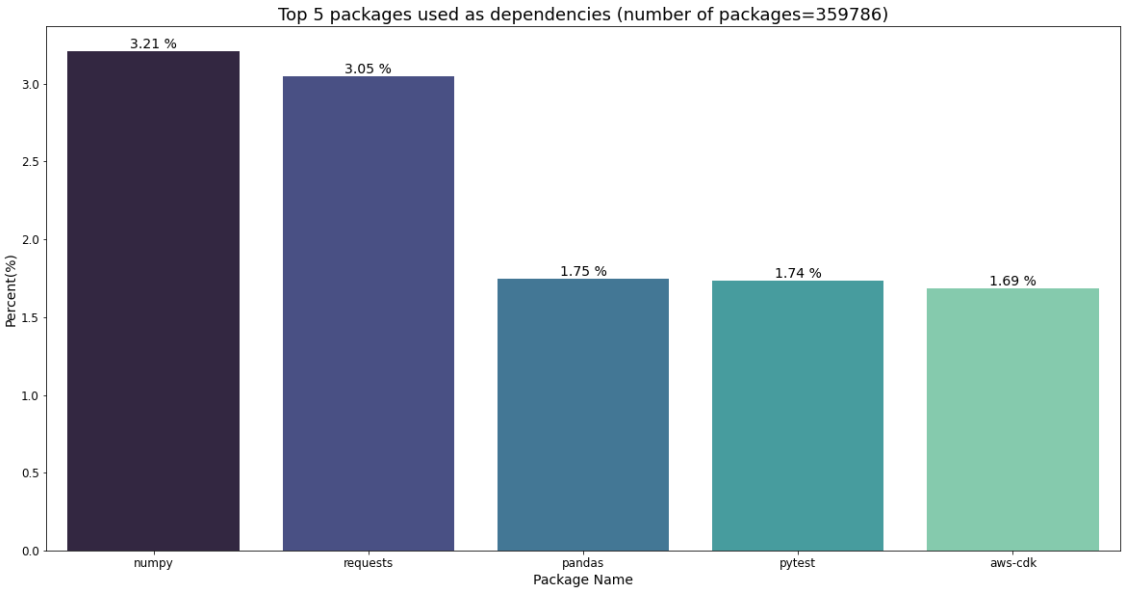
In order to better understand our transformed dataset, exploratory analysis was performed. We were interested in better understanding which countries downloaded the most Python packages, which packages were downloaded the most, and which packages were listed as a dependency the most. The downloads data was used to discover that 74.09% of package downloads occurred in the United States over the past year.



Top 10 Countries Downloading PyPi Packages

The top 5 packages which were listed as dependencies by other Python packages are:

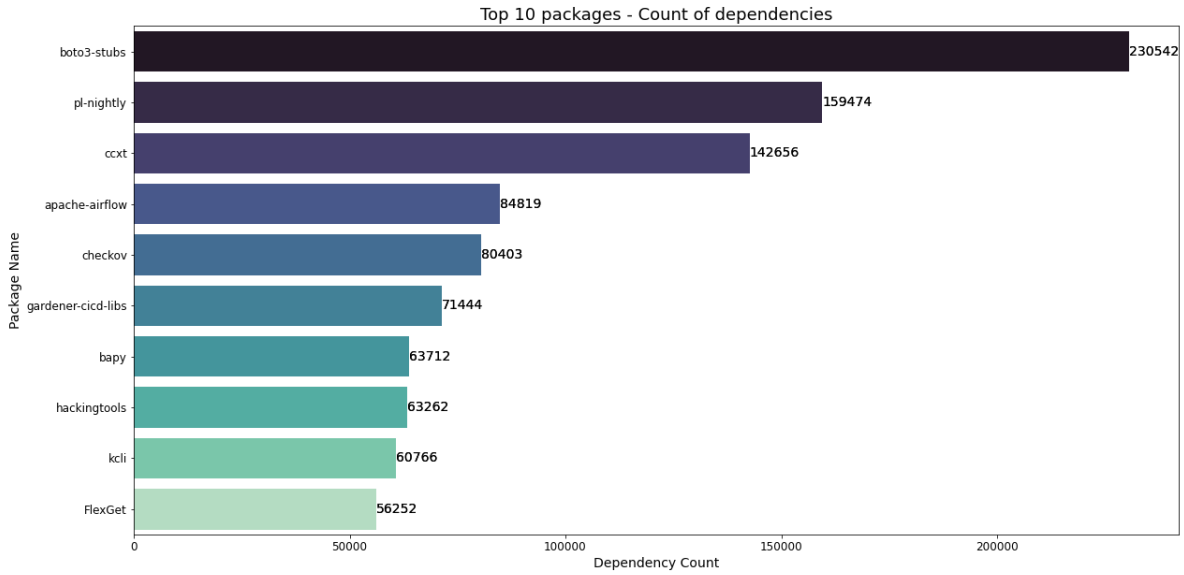
	Package Name	Description
1.	Numpy	A package for scientific computing with Python
2.	Requests	Requests is an elegant and simple HTTP library for Python, built for human beings.
3.	Pandas	Pandas is a fast, powerful, flexible and easy to use open-source data analysis and manipulation tool, built on top of the Python programming language
4.	Pytest	The pytest framework makes it easy to write small tests, yet scales to support complex functional testing for applications and libraries.
5.	Aws-cdk	The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to define your cloud application resources using familiar programming languages.



Top 5 Packages Used as Dependencies

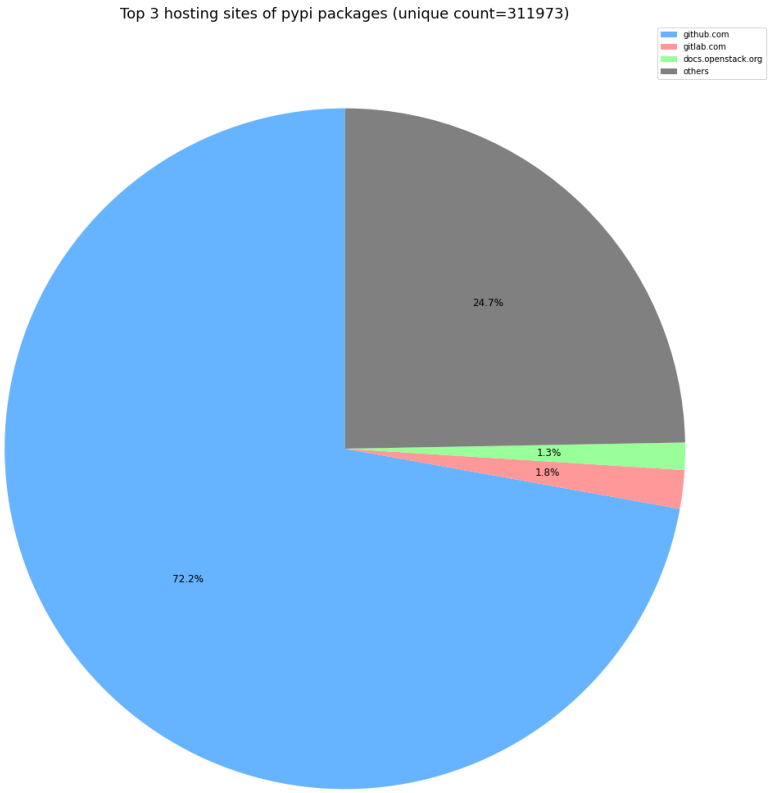
The top packages by their count of dependencies were:

	Package Name	Description
1.	Boto3-stubs	Type annotations for boto3 1.20.21 compatible with VSCode, PyCharm, Emacs, Sublime Text, mypy, pyright and other tools.
2.	Pi-nightly	No description
3.	Ccxt	A JavaScript / Python / PHP library for cryptocurrency trading and e-commerce with support for many bitcoin/ether/altcoin exchange markets and merchant APIs.
4.	Apache-airflow	Apache Airflow (or simply Airflow) is a platform to programmatically author, schedule, and monitor workflows.
5.	Checkov	Checkov is a static code analysis tool for infrastructure-as-code.
6.	Gardener-cicd-libs	No description
7.	Bapy	Bash, Ansible and Python Utils.
8.	Hakingtools	Bash, Ansible and Python Utils.
9.	Kcli	No description
10.	FlexGet	FlexGet is a multipurpose automation tool for content like torrents, nzbs, podcasts, comics, series, movies, etc. It can use different kinds of sources like RSS-feeds, html pages, csv files, search engines and there are even plugins for sites that do not provide any kind of useful feeds.

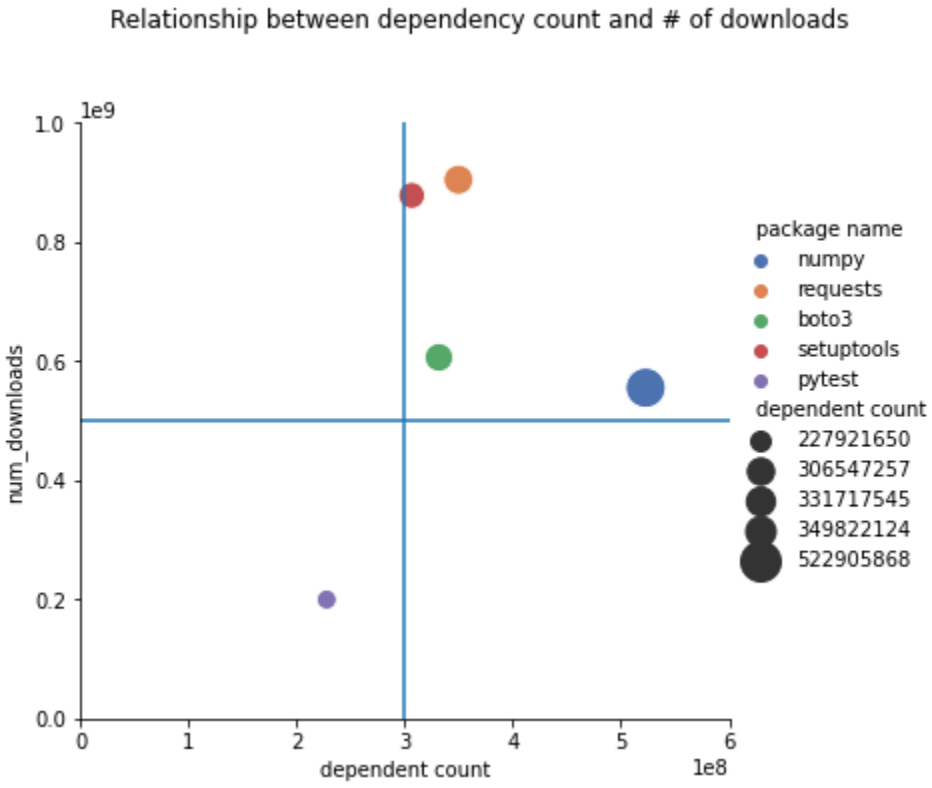


Top 10 Packages by Dependency Count

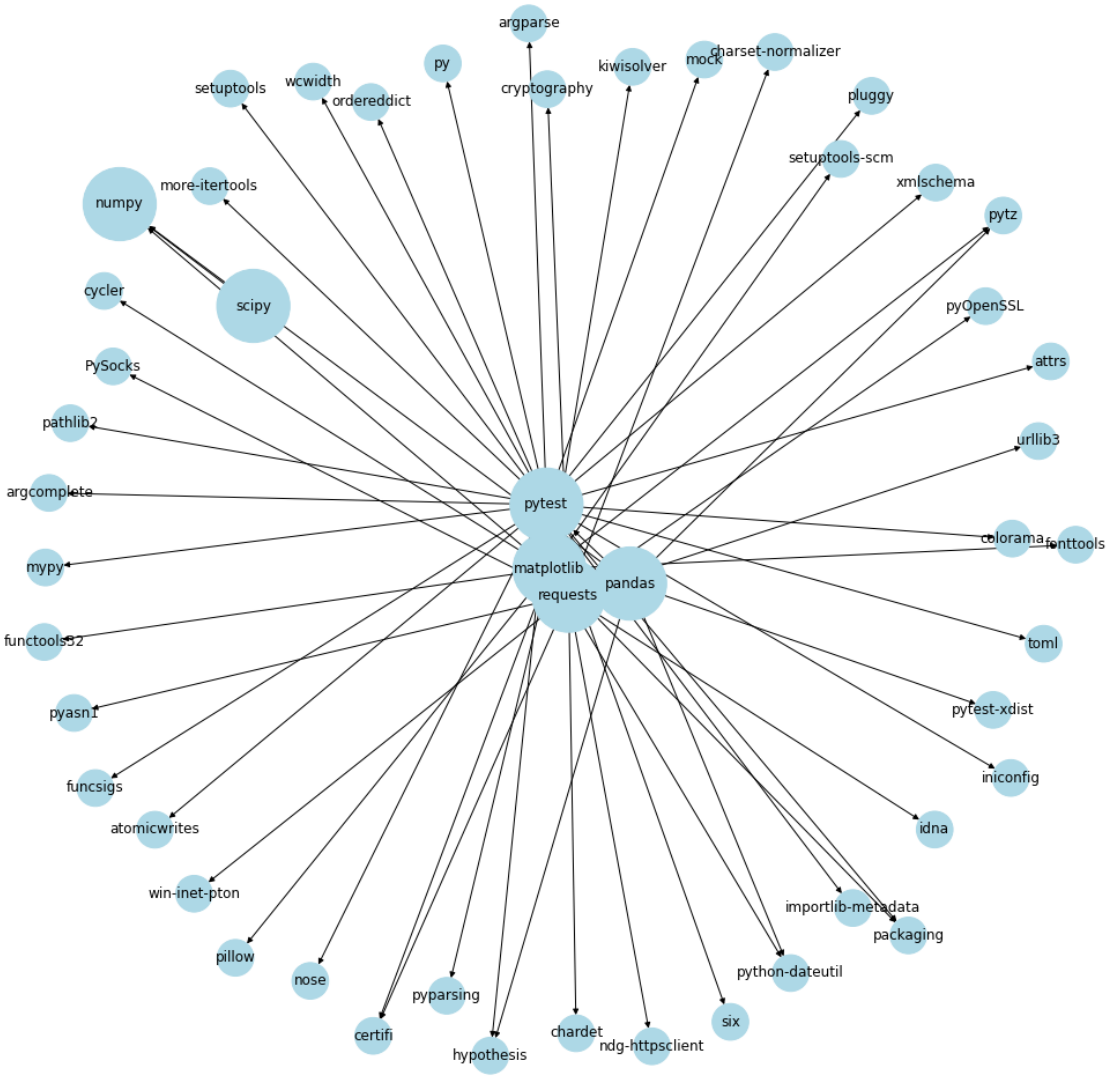
We wanted to understand how much of our original dataset was excluded by selecting GitHub hosted projects only. An analysis of the home_page field indicated that GitHub was by far the most popular code repository. GitHub was listed as the hosting site of 72% of projects on PyPi.org followed by GitLab and OpenStack. Other code hosting sites accounted for 25% of the projects.



The below graph demonstrates the impact of python packages by showing the relationship of downloads of the packages and the dependent count of those packages. The most downloaded packages are also frequently required as a dependency.



The following network graph shows the sample dependency network based on top six nodes with highest degree. The degree of a node in a network is the number of connections it has to other nodes.



Number of nodes: 50

Number of edges: 54

Average degree: 2.1600

Challenges and Roadblocks:

We faced various challenges throughout the semester learning new data systems and building our own dataset. The most challenging part was learning to use the Google Cloud BigQuery API and the method to query the data. After we learned to use the API, we found handling the dependency names was especially challenging as they were coded in PEP 508 format as an array. This field had to be exploding to single package names in order for further processing. Cleaning the data after this step was time consuming.

Additionally, we faced memory issues while querying and joining the data in various steps of the process. Refining the questions and streamlining the queries to aggregate the required data resolved this issue. We were also able to use Rivanna for higher computation power. Slurm scripts run through Rivanna enabled us to complete data querying, transformation, and aggregation that could not be completed during single Rivanna sessions.

Conclusion and Next Steps

Looking forward to next semester, we will be using the created edge lists to conduct more specific research on the impact of the open-source software. A few topics which are being considered are listed below in order of priority:

- Visualize the large networks in an effective manner
- Run network statistics such as density, centrality measures etc.
- Find patterns and path of diffusion of the packages - What are the patterns and dynamics of collaboration among OSS developers, within and across institutions, sectors (business, government, university, nonprofit), and countries?
- Conduct cost-impact analysis – lines of code as measure of cost and determine the impact a certain package has based on the cost. Eg: number of downloads, used as a dependency by other packages, who is using the package?
- Create a predictive model for a collaboration network to predict edges and observe performance improvement with different variables

Appendix A - Google Big Query PyPI API

Table 1 - Distribution Metadata

The description for each field in the PyPi metadata package and additional information can be found at the following URIs:

- https://packaging.python.org/specifications/core-metadata/
- https://www.python.org/dev/peps/pep-0621/

Field name	Type	Mode	Description	Potential Use
metadata_version	STRING	NULLABLE	Version of the file format; legal values are “1.0”, “1.1”, “1.2”, “2.1” and “2.2”. Automated tools consuming metadata SHOULD warn if metadata_version is greater than the highest version they support, and MUST fail if metadata_version has a greater major version than the highest version they support (as described in PEP 440, the major version is the value before the first dot). For broader compatibility, build tools MAY choose to produce distribution metadata using the lowest metadata version that includes all of the needed fields.	
name	STRING	REQUIRED	Name of package	
version	STRING	REQUIRED	Version of package	
summary	STRING	NULLABLE	Short description of package	
description	STRING	NULLABLE	Long description of package	
description_content_type	STRING	NULLABLE	A string stating the markup syntax (if any) used in the distribution’s description, so that tools can intelligently render the description.	
author	STRING	NULLABLE		Contributor network
author_email	STRING	NULLABLE		Contributor network
maintainer	STRING	NULLABLE	Empty when author is current maintainer, only used when someone different than author is maintainer	Contributor network
maintainer_email	STRING	NULLABLE		Contributor network

license	STRING	NULLABLE	Text indicating the license covering the distribution where the license is not a selection from the “License” Trove classifiers. See “Classifier” below. This field may also be used to specify a particular version of a license which is named via the Classifier field, or to indicate a variation or exception to such a license.	Select OSI licenses
keywords	STRING	NULLABLE	A list of additional keywords, separated by commas, to be used to assist searching for the distribution in a larger catalog.	Categories
classifiers	STRING	REPEATED	Indicates development status, can be used to find stable releases of software (List of classifiers Development Status :: 1 - Planning Development Status :: 2 - Pre-Alpha Development Status :: 3 - Alpha Development Status :: 4 - Beta Development Status :: 5 - Production/Stable Development Status :: 6 - Mature Development Status :: 7 - Inactive	Categories
platform	STRING	REPEATED	Operating systems?	
home_page	STRING	NULLABLE	A string containing the URL for the distribution’s home page.	Getting more contributor information
download_url	STRING	NULLABLE	A string containing the URL from which this version of the distribution can be downloaded.	
requires_python	STRING	NULLABLE	This field specifies the Python version(s) that the distribution is guaranteed to be compatible with. Installation tools may look at this when picking which version of a project to install.	
requires	STRING	REPEATED	Dependencies uses, PEP 508 https://www.python.org/dev/peps/pep-0508/	Dependency network
provides	STRING	REPEATED		
obsoletes	STRING	REPEATED		
requires_dist	STRING	REPEATED	Dependencies uses, PEP 508 https://www.python.org/dev/peps/pep-0508/	

			Each entry contains a string naming some other distutils project required by this distribution. The format of a requirement string contains from one to four parts: A project name, in the same format as the Name: field. The only mandatory part. A comma-separated list of ‘extra’ names. These are defined by the required project, referring to specific features which may need extra dependencies. A version specifier. Tools parsing the format should accept optional parentheses around this, but tools generating it should not use parentheses. An environment marker after a semicolon. This means that the requirement is only needed in the specified conditions. See PEP 508 for full details of the allowed format. The project names should correspond to names as found on the Python Package Index. Version specifiers must follow the rules described in Version specifiers.	
provides_dist	STRING	REPEATED	Changed in version 2.1: The field format specification was relaxed to accept the syntax used by popular publishing tools. Each entry contains a string naming a Distutils project which is contained within this distribution. This field must include the project identified in the Name field, followed by the version : Name (Version). A distribution may provide additional names, e.g. to indicate that multiple projects have been bundled together.	

			For instance, source distributions of the ZODB project have historically included the transaction project, which is now available as a separate distribution. Installing such a source distribution satisfies requirements for both ZODB and transaction. A distribution may also provide a “virtual” project name, which does not correspond to any separately-distributed project: such a name might be used to indicate an abstract capability which could be supplied by one of multiple projects. E.g., multiple projects might supply RDBMS bindings for use by a given ORM: each project might declare that it provides ORM-bindings, allowing other projects to depend only on having at most one of them installed. A version declaration may be supplied and must follow the rules described in Version specifiers. The distribution’s version number will be implied if none is specified. This field may be followed by an environment marker after a semicolon.	
obsoletes_dist	STRING	REPEATED	Changed in version 2.1: The field format specification was relaxed to accept the syntax used by popular publishing tools. Each entry contains a string describing a distutils project’s distribution which this distribution renders obsolete, meaning that the two projects should not be installed at the same time. Version declarations can be supplied. Version numbers must be in the format specified in Version specifiers. This field may be followed by an environment marker after a semicolon.	

			The most common use of this field will be in case a project name changes, e.g. Gorgon 2.3 gets subsumed into Torqued Python 1.0. When you install Torqued Python, the Gorgon distribution should be removed.	
requires_external	STRING	REPEATED	Changed in version 2.1: The field format specification was relaxed to accept the syntax used by popular publishing tools. Each entry contains a string describing some dependency in the system that the distribution is to be used. This field is intended to serve as a hint to downstream project maintainers, and has no semantics which are meaningful to the distutils distribution. The format of a requirement string is a name of an external dependency, optionally followed by a version declaration within parentheses. This field may be followed by an environment marker after a semicolon. Because they refer to non-Python software releases, version numbers for this field are not required to conform to the format specified in PEP 440: they should correspond to the version scheme used by the external dependency. Notice that there is no particular rule on the strings to be used.	
project_urls	STRING	REPEATED	Links to various things like documentation, repo, probably not useful to us	
uploaded_via	STRING	NULLABLE		
upload_time	TIMESTAMP	NULLABLE		
filename	STRING	NULLABLE		
size	INTEGER	NULLABLE		
path	STRING	NULLABLE		
python_version	STRING	NULLABLE	Python version, can be used to filter data within scope, format of this	Filtering data

			column varies (cp27, cp36, source, etc)	
packagetype	STRING	NULLABLE	Options [bdist_egg, bdist_dumb, bdist_wininst, bdist_msi, sdist, bdist_wheel, bdist_rpm, bdist_dmg]	
comment_text	STRING	NULLABLE	Mostly null	
has_signature	BOOLEAN	NULLABLE	Don’t know what this represents	
md5_digest	STRING	REQUIRED	Hash	
sha256_digest	STRING	NULLABLE	Hash	
blake2_256_digest	STRING	NULLABLE	Hash	

Caveats

In addition to the caveats listed in the background above, Linehaul suffered from a bug which caused it to significantly under-report download statistics prior to July 26, 2018. Downloads before this date are proportionally accurate (e.g. the percentage of Python 2 vs. Python 3 downloads) but total numbers are lower than actual by an order of magnitude. [Source]

Table 2 - File Downloads

Name indented according to nesting. I believe this data is in JSON format.

name	Type	Mode	Description	Potential Use
timestamp	TIMESTAMP	REQUIRED		
country_code	STRING	NULLABLE		
url	STRING	REQUIRED		
project	STRING	REQUIRED	Project name	
file	RECORD	REQUIRED		
filename	STRING	NULLABLE		
project	STRING	NULLABLE		
version	STRING	NULLABLE		

type	STRING	NULLABLE		
details	RECORD	NULLABLE		
installer	RECORD	NULLABLE		
name	STRING	NULLABLE	Name of installer	
version	STRING	NULLABLE		
python	STRING	NULLABLE		
implementation	RECORD	NULLABLE		
name	STRING	NULLABLE		
version	STRING	NULLABLE		
distro	RECORD	NULLABLE		
name	STRING	NULLABLE		
version	STRING	NULLABLE		
id	STRING	NULLABLE		
libc	RECORD	NULLABLE		
lib	STRING	NULLABLE		
version	STRING	NULLABLE		
system	RECORD	NULLABLE		

name	STRING	NULLABLE		
release	STRING	NULLABLE		
cpu	STRING	NULLABLE		
openssl_version	STRING	NULLABLE		
setuptools_version	STRING	NULLABLE		
tls_protocol	STRING	NULLABLE		
tls_cipher	STRING	NULLAB		