The Scientific Research Portfolio

Section 1 Planning

Milestone	*Timeframe*	*Date Completed*
Decide research area	End of Term 4	Nov 22, 2021
Shortlist relevant articles	End of Term 4	Nov 22, 2021
Develop preliminary Scientific Research Question & Hypothesis	End of Term 4	Nov 23, 2021
Write annotated bibliography	Term 1 Week 2	Nov 23, 2021
Refine and justified Scientific Research Question & Scientific Hypothesis	Term 1 Week 4	Dec 9, 2022
Develop methodology	Term 1 Week 6	Feb 4, 2022
Write Literature Review	Term 1 Week 4	Feb 25, 2022
Collect data	Term 1 Week 10	April 28, 2022
Analyze data	Term 2 Week 3	May 29, 2022
Write scientific research report	Term 2 Week 6	June 9, 2022

Working Reference List

Danny S 2018, ‘How Google autocomplete works in Search’, Google: The Keyword, accessed 22 Feb 2022, <https://blog.google/products/search/how-google-autocomplete-works-search/>

Chloe K, Aniko H, David L, Christo W 2015, ‘Location, Location, Location: The Impact of Geolocation on Web Search Personalization’, *ResearchGate,* accessed 22 Feb 2022, <https://www.researchgate.net/publication/301417602_Location_Location_Location>

Peng W, Xianghang M, Xiaojing L, XiaoFeng W, Kan Y, Feng Q, Raheem B 2018, ‘Game of Missuggestions: Semantic Analysis of Search-Autocomplete Manipulations’, accessed 22 Feb 2022, <https://homes.luddy.indiana.edu/xw7/papers/peng18ndss.pdf>

Chris C, Theo L, Ute S 2021, ‘What you see depends on where you sit: The effect of geographical location on web-searching for systematic reviews: A case study’, Research Synthesis Methods, Volume 12 Issue 4, P. 557-570, accessed 22 Feb 2022, <https://onlinelibrary.wiley.com/doi/10.1002/jrsm.1485>

Summaries & annotations extracts of articles

ADM+S Center & Algorithm Watch - Australian Search Experience Project

https://www.admscentre.org.au/searchexperience/

Summary/Analysis	This study is in progress however some initial findings have been presented. ● Different platforms had different stability patterns. ○ Google search largely stable ○ Google news very fast-moving ○ Google Video quite static ○ Youtube stable for top 5 results then highly changeable. ● Limited evidence of personalization. ○ Personalization in google search driven by location ○ Critical search topics appear manually curated ○ Some differences based on browser type ○ No data as of current for other platforms.
Limitations	Limited sample size of 1013 participants. Demographics of citizen scientist cohort unrepresentative. Some participant attrition.
Further Research	Per platform and per-browser analysis. Evaluation of the result quality. New search terms that reflect emerging topics such as real world upcoming events.
Other	Significant information here on the text samples (stimulus) to use for testing. Source: https://aw-datenspende-bucket.s3.us-east-2.amazonaws.com/inject.1.1.4.5.json "Liberal Party", "National Party", "Labor Party", "Greens", "One Nation", "Scott Morrison", "Barnaby Joyce", "Anthony Albanese", "Adam Bandt", "Pauline Hanson", "COVID", "Vaccine", "Travel rules", "Quarantine", "Lockdown", "Home loan", "Mortgage broker", "Cash loan", "Cash advance", "Superannuation", "Critical Race Theory", "Feminism", "Black Lives Matter", "eating healthy can prevent what diseases", "what kind of fat to avoid", "knee pain what to do", "why are some parents concerned about vaccines", "should i get vaccinated",

https://dataskop.net/erste-ergebnisse/

Conclusions	The study found that a particular german-language news TV channel was the most popular Youtube News news source however it's prominence was not well explained. This study has also not had its full findings published yet.

Limitations	Sample size of 5000 participants. People who donated data are not necessarily representative of the larger cohort. There was significant participant attrition that affects the data in a currently non-discussed way.
Further Research	This study only spoke about further analysis within the study and didn't really add much to possible research.

Google Scholar's Ranking Algorithm: An Introductory Overview

https://www.issi-society.org/proceedings/issi_2009/ISSI2009-proc-vol1_Aug2009_batch2-paper-1.pdf

Conclusions	This study performed an analysis of the ranking engine for academic articles on Google Scholar. Analysis showed that citation count was the highest weighted factor. Next was article title, while full article text was weighted significantly less.
Limitations	Authors said that more sample data was needed, and came up with 6 specific conclusions with lots unprovable or untested.
Further Research	The different ranking algorithms used for keyword search, related articles and cited by. The weight of an author's or general's reputation on the ranking.

Game of Missuggestions: Semantic Analysis of Search-Autocomplete Manipulations

https://homes.luddy.indiana.edu/xw7/papers/peng18ndss.pdf
https://www.ndss-symposium.org/wp-content/uploads/2018/03/ndss2018_07A-1_Wang_Slides.pdf

Conclusions	There is significant poisoning of the autocomplete results for different kinds of samples, up to 4.13% for Lending Products.
Limitations	Due to the use of Machine learning, detection can be evaded by making results more closely mimic benign ones. Difficult to test completely if results are manipulations. Limitation from lack of ground truth and manual evaluation.
Further Research	Develop similar systems for e-commerce platforms.
Analysis	"Game of Missugestions: Semantic Analysis of Search-Autocomplete Manipulations" (Wang 2018) has solved this through the combination of a NLP and machine learning system that achieves a 96.23% precision and 95.63% (Wang 2018, p. 1) recall on terms, all without passing to the Google API except for scraping ancillary data about the page returned by the term for analysis. This technological approach was very interesting, although requiring advanced skills in machine learning and computer science to replicate. This combined with the lack of any published source code means that the paper is of little use to researchers wishing to replicate the study. However, the study's innovative methods worked extremely well, producing results on over 100 million search terms which is a scale that would be impossible

	by using a screen-scraping or API based approach. This inspired further research into a more technological approach, and to explore a wider range of data collection methods.

What you see depends on where you sit: The effect of geographical location on web-searching for systematic reviews: A case study

https://pubmed.ncbi.nlm.nih.gov/33713573/

Conclusions	● Researchers must be careful to report geographical location and other search data to ensure web searches are repeatable. ● Search returns and page rank vary by location.
Limitations	● Low sample size of only 43 items between 12 locations.
Further Research	● Required manual analysis by researchers. Perhaps we can look at autocomplete results instead which are more easily categorized.
Analysis	“What you see depends on where you sit: The effect of geographical location on web-searching for systematic reviews: A case study” (Chris Cooper 2021), a paper which is looking to determine whether the location of researchers around the world could have any impact on systematic reviews. In (Chris Cooper 2021, p. 2) a single computer is used with Cookies cleared between searches on the chrome browser. However, this method does not prevent persistent fingerprinting via FLoC, Javascript-based fingerprinting and device fingerprinting which while on their own present no risk to identification when combined can be used to identify the user even after cookie and site data is cleared. The lower sample size of the paper also brings into question its findings.

Location, Location, Location: The Impact of Geolocation on Web Search Personalization

https://dl.acm.org/doi/10.1145/2815675.2815714

Conclusions	● Highly dependent on what the user searches for ● Queries for establishments vary 4-5 results per page ● Queries for more general terms exhibit little to no personalization by location.
Limitations	● Use of the mobile version of Google ● Only 94% identical results to desktop
Further Research	● Google search personalizes based on search results in the past 10 minutes - A. Hannak, P. Sapiezy´nski, A. M. Kakhki, B. Krishnamurthy, D. Lazer, A. Mislove, and C. Wilson. Measuring Personalization of Web Search. WWW, 2013.

Analysis	“Location, Location, Location: The Impact of Geolocation on Web Search Personalization” (Chloe Kliman-Silver 2015) which analyzed search results by location in the context of personalization and the “Filter Bubble Effect”, where personalization algorithms prevent users from seeing different perspectives by marking information as not important. This is affected by personalization because demographic traits such as race, income, educational attainment and political affiliation can be inferred from a location. The paper tested a variety of terms including political, controversial and local topics. The paper is more technically sound than the previous paper, providing evidence that their method prevents personalization on many factors. However, they do not provide any evidence that the mobile and desktop version of Google provide the same results. Additionally, their validation study measuring IP address-based personalization on the mobile browser only showed 94% identical results which should be studied further to come to a statistically significant conclusion. The paper also had an important point which was to ensure that all queries to google are sent to the same data center.

Log of the sequential development of the scientific research process

Project Idea - First Proposal - 28th Nov 2021

For the **Science Extension Major Research Project**, I Intend to study the difference in autocomplete results at different geographical areas around the world. This study will be conducted using automated software in order to sample a large number of autocompletes from each location around the world.

Ashburn, USA	Atlanta, USA	Boston, USA	Charlotte, USA	Chicago, USA
Cincinnati, USA	Dallas, USA	Denver, USA	Houston, USA	Las Vegas, USA
Los Angeles, USA	Miami, USA	New Orleans, USA	New York, USA	Phoenix, USA
San Jose, USA	Seattle, USA	Dubai, UAE	Taipei, Taiwan	Zurich, Switzerland
Stockholm, Sweden	Madrid, Spain	Johannesburg, SA	Ljubljana, Slovenia	Bratislava, Slovakia
Singapore, Singapore	Belgrade, Serbia	Bucharest, Romania	Lisbon, Portugal	Warsaw, Poland
Lima, Peru	Oslo, Norway	Auckland, New Zealand	Amsterdam, Netherlands	Chisinau, Moldova
Guadalajara, Mexico	KL, Malaysia	Luxembourg, Luxembourg	Riga, Latvia	Seoul, Korea
Tokyo, Japan	Milan, Italy	Tel Aviv, Israel	Dublin, Ireland	Mumbai, India

New Delhi, India	Reykjavik, Iceland	Budapest, Hungary	Athens, Greece	Frankfurt, Germany
Bordeaux, France	Paris, France	Helsinki, Finland	Tallinn, Estonia	Copenhagen, Denmark
Prague, Czechia	Zagreb, Croatia	San Jose, Costa Rica	Bogota, Colmbia	Santiago, Chile
Montreal, Canada	Toronto, Canada	Vancouver, Canada	Sofia, Bulgaria	São Paulo, Brazil
Brussels, Belgium	Vienna, Austria	Adelaide, Australia	Brisbane, Australia	Melbourne, Australia
Perth, Australia	Sydney, Australia	Buenos Aires, Argentina	Tirana, Albania	Birmingham, UK
Glasgow, UK	London, UK	Manchester, UK		

As of just having copied out that list from my VPN provider, I may be inclined to limit it some. In order to connect to these servers and limit variables, I will be using a VMWare EXSi hosted cluster and scripting to automatically spin up, set up, perform the experiment and then shutdown the instances of Windows.

I am also still looking into the exact words or terms that I would like to test. These will likely be decided upon a better investigation into the literature below.

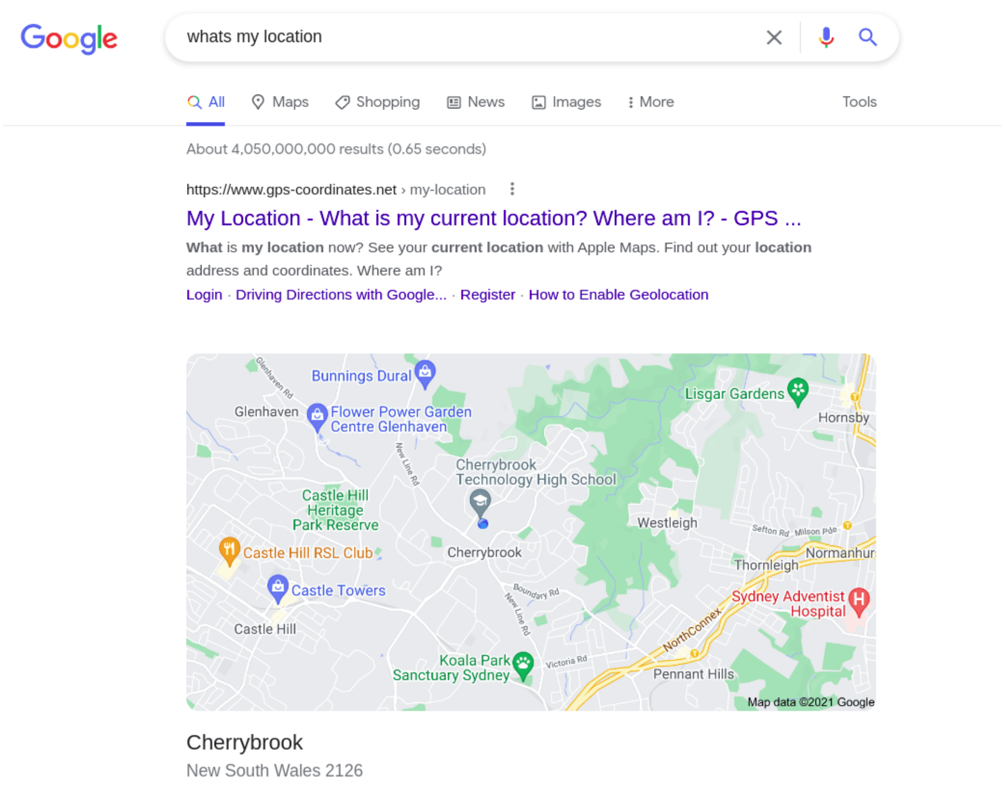
Independent Variables	Dependent Variables	Controlled Variables
Location (As from list above),	Characters after text sample, first 10 autocompletes kept in order.	Operating system, web browser, computer specs, account in use, text samples

Response #1 - 7th Dec 2021

Very interesting! How will you know that VPN will trick google's search and be a valid way to location spoof? Let us get a small scale trial going and see the feasibility for this study.

Response #2 - 9th Dec 2021

The location spoofing will be confirmed by seeing the results of googling "Whats my location". See attached image.



whats my location

All · Maps · Shopping · News · Images · More · Tools

About 4,050,000,000 results (0.65 seconds)

https://www.gps-coordinates.net › my-location

My Location - What is my current location? Where am I? - GPS ...

What is **my location** now? See your **current location** with Apple Maps. Find out your **location** address and coordinates. Where am I?

Login · Driving Directions with Google... · Register · How to Enable Geolocation

Cherrybrook

New South Wales 2126

I have begun working on the feasibility study. The steps taken so far include: - Investigated libraries needed to communicate with infrastructure: Listed in LIBRARIES.md - Prepared operating system for livessh-type booting: Steps taken in IMAGEPREP.md - Coding in-built library classes

Steps to complete include: - Investigate asynchronous programming - Prepare Database for result storage - Write code for experiment engine (server) with DB - Test EXSi Cluster - Write client code with automatic VPN connection, automated experimentation and result collection - Run test experiment with small sample

Technical notes - 13th Dec 2021

IP address space of 10.137.0.0/24

```
Address:   10.137.1.0           00001010.10001001 .00000001.00000000
Netmask:   255.255.0.0 = 16     11111111.11111111 .00000000.00000000
Wildcard:  0.0.255.255          00000000.00000000 .11111111.11111111
=>
Network:   10.137.0.0/16        00001010.10001001 .00000000.00000000 (Class A)
Broadcast: 10.137.255.255       00001010.10001001 .11111111.11111111
HostMin:   10.137.0.1           00001010.10001001 .00000000.00000001
HostMax:   10.137.255.254       00001010.10001001 .11111111.11111110
Hosts/Net: 65534                (Private Internet)
```

Ideas with Mr An - 14th Dec 2021

- Rank which result is first related to covid-19.

Test from Las Vegas 1. testosterone 2. test now 3. Testal Mexican Kitchen 4. Restaurant. Phoenix. AZ 5. testicular cancer 6. testicular torsion 7. testosterone booster 8. test now covid 9. testout 10. trrinetarearces reanlarcarssazan?¢ thararmé

Test from Los Angeles 1. testosterone 2. test internet speed 3. test my internet speed 4. testicular cancer 5. testicular torsion 6. testimony 7. test statistic calculator 8. testosterone booster 9. ’ Testament

Test from UK, London 1. Testing for AI 2. test and trace 3. testing for schools 4. test internet speed 5. testosterone 6. testicular torsion 7. test my internet speed 8. testbase 9. tectir ular cancer

Test from Sydney, Australia 1. covid cases nsw 2. covid nsw 3. covid testing near me 4. covid testing sydney 5. covid testing parramatta 6. covid restrictions nsw 7. covid cases nsw today 8. covid 19 nsw 9. covid symptoms 10. covid cases victoria

Update - 15th Dec 2021

First try at automatically setting up the VPN connection today. Seems easy enough working well. Simply calling the openvpn process while experiment is running.

Update - 16th Dec 2021

The client works fully and now I am working on introducing the socket handling mechanics. This will require the socket library and a server that provides connection handles, managing the order and execution of tasks.

Update - 23rd Dec 2021

First attempt to make a system to handle the injection of the experiment into the final node. Using pysftp, we upload the folder recursively.

Update - 31st Dec 2021

The project is now called MAS or Magma Automated Science. Today I have been working on documentating the steps required in order to fully inject and run the experiment. This process can be summarised as: - VM creation - VM startup - Client code injection - Client code execution - VPN activation - Firefox startup - Data scraping - VPN disconnection - VM shutdown - VM destruction

I am making progress on all parts of this process. Additionally, I have switched from using a socket system to a http based API which allows the design to become asynchronous. Additionally, I have taken a lease from REST-type apis, using the model to design a read-edit-destroy API that makes it easy to implement the system handling the DB end as well as multiple client processors to handle EXSi, injection etc.

Additionally, I wrote the client code to handle the requests to the HTTP api as well. At this stage all VM actions are manual but I am working to use the pyvmomi api to handle this side.

Update - 1st Jan 2022

I have started working on the VM creation step and have a semi-functioning VM Creator that needs a few more features such as virtual CD addition. The client has come along a bit as well, now featuring proper timing and delays to prevent API overload.

Update - 3rd Jan 2022

The EXSi API now handles deletion of VMs, as well as shutdown.

Update - 18th Jan 2022

The EXSi API portion now successfully handles the creation of VMs including virtual CD Drives and network adapters. Additionally, I have added code to allow fetching of the VMs ips from EXSi which will be useful later in the injection code.

Update - 19 Jan 2022

I am now working on the SSH injector that will actually run the code using the paramikro and fabric libraries. Some serious issues here with allowing my code to continue to run after closing the SSH Connection. Solutions to be found, hopefully.

Update - 20 Jan 2022

I have removed my VENV as the dependencies need to be installed fresh on the VM anyway, and windows to linux incompatibilities in using the same virtual environment are causing me issues.

Update - 21 Jan 2022

Today I have succesfully run the code fully for the first time successfully. It has built a VM from scratch, injected it and ran a test code.

Update - 26 Jan 2022

I can't believe it is actually working fully today. I had some more issues with the fresh install including with a newer version of pytesseract and pip updating to a version incompatible with the version of python on the system. By fixing these steps I am nearly ready to run a multi-sample gatherer test.

Update - 27 Jan 2022

I have run into performance issues with the pysftp library and so am transition to using fabric for both upload and run steps. Additionally, I may look into using over the wire compression to decrease the file count.

Update - 28 Jan 2022

Today I started the first big run, collecting results from 48+ countries 5 times with the stimulus “test”

Update - 29 Jan 2022

After removing non-english results, I have been left with 210 results form 41 countries. This project is now a success and I have a feasibility testing data set as well. From here the project needs to find a strong footing, with a research question to test.

Update - 3 Feb 2022

I talked to Mr An more about the Feasibility study– He is glad it has worked but we need to figure out our analysis steps. I was unable to find a data scientist to help us with the project so I will be on my own.

At this stage, it is looking like a Chi-Squared analysis, comparing covid-19 case numbers to the prevalence of the word “covid” in auto-complete results may be the best way to continue with the project. I plan to meet with Mr An to discuss this tomorrow during a double free period.

Update - 4 Feb 2022

As of current I still agree whit my previous Chi-Squared analysis idea.

The hypothesis will be: ### An increased prevalance of Covid-19 infections will correlate with a higher level of Covid-19 related terms in autocomplete results.

My only real question at this point is how to make an expected value for the Chi-Squared analysis. I have a datasource for automatically receiving up-to-date covid-19 infection rates world wide thanks to the John Hopkins University Covid-19 Datasource but I am unsure exactly how to calculate them into a comparable scale or term. More discussion of this is necessary with Mr An.

Update - 4 Feb 2022 (Mr An)

Chi Squared test of independence comparing the number of covid-19 related auto-complete results and the covid-19 case results.

Research Question/Hypothesis: - Include variables

Compare to number of people getting covid per 100,000

Update - 28 April 2022

The look now is to find out what terms to search for. For this, I will write a program that goes down the database and records the number of terms to find the most popular ones.

Total results: 1574. 1516 Compliant. 60 Removed as part of Data Cleansing We have a list: testosterone: 178 test Internet speed: 97 test my internet speed: 94 testicular cancer: 67

testicular torsion: 45 testing sites near me: 42 testing near me: 39 test: 37 testout: 26 test speed: 25 test cricket: 24 test internet speed: 24 testing near me: 24 test match: 20 testing: 20 test wizard: 20 test grade calculator: 19 testosterone booster: 17 test de embarazo: 15 test ranking: 15 testing sites nyc: 14 testimony: 13 test my Internet speed: 13 testing: 11 test pcr: 11 testflight: 10 test netconnection: 10 testament: 10 test score: 10 test rapldo covld: 10 test de personalldad: 10 test match llve: 10 testbeforeyougo: 10 test antlgeno covld: 9 test match score: 9 test calculator: 9 testing sites houston: 9 testout: 9 testamur: 9 test isolation payment: 8 testis: 8 test covld: 8 testing sites brooklyn: 8 testing clinics near me: 7 testicular cancer: 7 test my internet speed: 7 testosterone booster: 7 testing sites katy: 7 testing los angeles: 7 testing for covid: 7 testing texas: 7 testing for covid near me: 7 testdisk: 6 test de antlgenos: 6 test: 6 test Iq: 6 test webcam: 6 testing clinics sydney: 5 test and isolate payment: 5 test4free: 5 testing covid near me: 5 testing covid adelaide: 5 testing covid port adelaide: 5 test isolation payment: 5 testing stations near me: 5 testing stations auckland: 5 testing stations manukau: 5 testing stations: 5 testimony meaning: 5 testing sites chicago: 5 test now: 5 testnav: 5 testing colorado: 5 testing denver: 5 test and protect cincinnati: 5 test and protect: 5 test kit covld: 5 test kit: 5 test kit saliva: 5 test calculator: 5 testing center byui: 5 testosterone supplements: 5 testing clinic near me: 5 testing clinic brisbane: 5 testing clinic gold coast: 5 testing pasadena: 5 test rapid covld: 5 teste auto: 5 test de personalidad: 5 test vocacional: 5 testosterona: 5 test de Iq: 5 test de velocidad de internet: 5 test de las 16 personalldades: 5 testing site near me: 5 testimonial: 4 testing sites newark nj: 4 testing sites union nj: 4 testing sites new jersey: 4 test Isolatlon payment: 4 testing california: 4 testing sites melbourne: 4 test Internet speed: 4 test mbti: 4 testdisk: 4 testis: 3 testing sites norwalk ct: 3 test internet speed: 3 testing clinics liverpool: 3 testseala bs: 3 testing colorado: 3 test covld chlslnau: 3 test pcr covld chlslnau: 3 testwizard: 3 testudo times: 3 test covld: 3 testing site mlaml: 3 testing site mlaml beach: 3 test covld near me: 3 test cps: 3 testing sites new york: 3 test de velocidad: 3 testing for covid near me: 2 test and go thalland: 2 testimonial: 2 testpaperfree: 2 test match llve: 2 testing sltes nyc: 2 testing sltes norwalk ct: 2 testing las vegas: 2 testing sites nsw: 2 test and isolate payment: 2 testudo times: 2 testnav: 2 test covld Chisinau: 2 test pcr covld Chisinau: 2 testing sites geelong: 2 testing sites: 2 testing sites Jersey clty: 2 testing for covid greensboro nc: 2 testimony: 2 testing callfornia: 2 testing site miami: 2 testing site miami beach: 2 test covid near me: 2 test tube: 1 testing clinics liverpool: 1 tested positive for covid 19: 1 testing sites rhode island: 1 testing sites taunton ma: 1 testing callfornia: 1 test match llve: 1 testing for covld near me: 1 test mbtl: 1 testicles: 1 test tube: 1 testosterone meaning: 1 test de antigenos: 1 test antigeno covld: 1 testing clinics parramatta: 1 test my speed: 1 test mbti: 1 testing st louis: 1 testbook logln: 1 test my speed: 1

And a manually selected list of covid-related terms.

testing sites near me tesitng near me testing near me testing testing sites nyc test pcr test antlgeno covld test isolation payment testing sites houston test covid testing sites brooklyn testing clinics near me testing sites katy testing los angeles testing for covid testing texas testing for covid near me testing clinics sydney test and isolate payment test isolation payment test4free testing covid near me testing covid adelaide testing covid port adelaide testing stations near me testing stations auckland testing stations manukau testing stations testing sites chicago testing center byui testing colorado testing denver testing sites chicago test now test and protect test kit covld test kit test kit saliva test calculator testing clinic near me testing clinic brisbane testing clinic gold coast testing pasadena test rapid covld testing site near me testing sites newark nj tesitng sites union nj testing sites new jersey test

Isolation payment testing california testing sites melbourne testing colorado testing site miami testing site miami beach test covid near me testing sites new york tested positive for covid 19 tested positive

Refining into key terms

tested positive testing isolation covid, covld clinic pcr

Anything with these I will consider related.

19th May 2022

I have coded the KeyWord processor, and now have got the final_result.json file. Now to find the Covid-19 case load for Jan 28th 2022.

25th May 2022

I have finished coding up the code that reads in the CSV data from John Hopkins report, and exports an excel compatible CSV file for processing using statistical analysis.

29th May 2022

I have now performed the ANOVA and Pearson's analysis on the data. As for the ANOVA, we have found statistical significance below P=0.01. Unfortunately, with a Pearsons (R) value of 0.037, there is no good correlation between the variables studied.

Section 2 Evidence

Work Sample #1 - Data Collection

```
# ©Macauley Lim 2021 -- File Licensed Under The GNU GPLv3. See The Full Notice In
License.md For Binding Terms.
import logging, subprocess, time, pytesseract, cv2, numpy, os, re
import MASsettings
from PIL import Image
try:
   import pyautogui
except Exception as e:
   logging.error("No PyAutoGUI Loaded | Exception: "+str(e))

class SciExClientAutomaticExecution:
   def __init__(self, nodeid, aptpackagelist, pippackagelist):
       self.nodeid = nodeid
       self.settings = MASsettings.Settings()
       logging.basicConfig(level=logging.DEBUG, filename="client.log", filemode="w")
       logging.getLogger().addHandler(logging.StreamHandler())
       self.log = logging.getLogger()
       try:
           self.sw, self.sh = pyautogui.size()
       except Exception as e:
           logging.debug(str(e))
       logging.info("SciExClientAutomaticExecution Initalized")

   def install(self):
       try:
           dns = subprocess.Popen(["echo", "'nameserver 1.1.1.1'", ">>",
"/etc/resolv.conf"])
           dns.wait()
           aptupdate = subprocess.Popen(["apt", "update"])
           aptupdate.wait()
           aptcommand = ["apt", "install", "-yq"]
           aptcommand.extend(aptpackagelist)
           gui = subprocess.Popen(aptcommand, stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
           gui.wait()
           output = gui.stdout.read().decode("utf-8").split("\n")
           pipcommand = ["pip", "install"]
           aptcopipcommandmmand.extend(pippackagelist)
```

```
            pip = subprocess.Popen(aptcommand, stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
            pip.wait()
            pipoutout = pip.stdout.read().decode("utf-8").split("\n")
            for line in output:
                logging.debug("APT Install: "+str(line))
            for line in pipoutout:
                logging.debug("PIP Install: "+str(line))
        except Exception as e:
            logging.error("INSTALLGUI: "+str(e))

    def startx(self):
        try:
            x = subprocess.Popen()
            gui = subprocess.Popen(["startx"], stdout=subprocess.PIPE,
stderr=subprocess.STDOUT)
            gui.wait()
            output = gui.stdout.read().decode("utf-8").split("\n")
            for line in output:
                logging.debug("StartX: "+str(line))
        except Exception as e:
            logging.error("StartX: "+str(e))

    def launchFirefox(self):
        try:
            self.firefox = subprocess.Popen(["firefox", "https://google.com"])
            time.sleep(8)
        except Exception as e:
            logging.error("Firefox launch error :"+str(e))

    def runAutoCompleteTest(self, stimulus):
        pyautogui.moveTo(0.50*self.sw, 0.5*self.sh)
        pyautogui.click()
        pyautogui.write(stimulus)
        time.sleep(1)
        im = pyautogui.screenshot()
        imc = im.crop((0.14*self.sw, 0.58*self.sh, 0.7*self.sw, 0.99*self.sh))
        imc_cv = numpy.array(imc)
        resized = cv2.resize(imc_cv, (0,0), fx = 4, fy = 4)
        gray = cv2.cvtColor(resized, cv2.COLOR_BGR2GRAY)
        blur = cv2.GaussianBlur(gray, (3,3), 0)
        thresh = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
        kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3,3))
```

```python
        opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=1)
        cv2.imwrite("test.png", opening)
        text = pytesseract.image_to_string(opening, config="--psm 6")
        text.strip("\n\n")
        logging.info(text)
        return text

    def closeFirefox(self):
        try:
            time.sleep(2)
            self.firefox.terminate()
        except Exception as e:
            logging.error("Firefox close error"+str(e))

    def joinVPN(self, region):
        vpnspath = os.getcwd() + "/vpn_connections/"
        servers = "|".join(os.listdir(os.getcwd() + "/vpn_connections/"))
        server = re.search(region+'-([abc]\S\S)', servers).group(1)
        vpnpath = vpnspath + "ipvanish-" +region+"-"+server+".ovpn"
        credsfile = os.getcwd() + "/" + self.settings.read("SETTINGS.VPN.FILE")
        try:
            self.vpn = subprocess.Popen(["openvpn", "--auth-nocache","--config", vpnpath,
"--auth-user-pass", credsfile], cwd=vpnspath)
            time.sleep(10)
            logging.info("VPN Returned:" + str(self.vpn.returncode))
        except Exception:
            logging.error("VPN Launch Error")

    def leaveVPN(self):
        try:
            self.vpn.terminate()
            a = subprocess.Popen(["killall", "openvpn"])
            time.sleep(3)
            a.terminate()
        except Exception:
            logging.error("Failed to close VPN")

if __name__ == "__main__":
    e = SciExClientAutomaticExecution(9999, " ", " ")
    e.launchFirefox()
    e.runAutoCompleteTest("test")
    e.closeFirefox()
```

```
# ©Macauley Lim 2021 -- File Licensed Under The GNU GPLv3. See The Full Notice In
License.md For Binding Terms.
import requests, MASexperiment, os, time, sys

url = "http://$$$.$$$.$$$.$$$:5000"
aptrequest = requests.get(url+"/aptpackages")
piprequest = requests.get(url+"/pippackages")

nodeid = sys.argv[1]
aptpackagelist = aptrequest.text.split(",")
pippackagelist = piprequest.text.split(",")

experiment = MASexperiment.SciExClientAutomaticExecution(nodeid, aptpackagelist,
pippackagelist)
continueprocessing = True
httpheaders = {"X-Node-ID":str(nodeid)}

while continueprocessing:
   commandrequest = requests.get(url+"/commands", headers=httpheaders)
   commands = commandrequest.text.split(",")
   command = str(commands[0])
   print("Command Received: "+command)
   if command=="Install":
       print("Installing")
       experiment.install()
       removecommandrequest = requests.post(url+"/commands",
"Install".encode('utf-8'),headers=httpheaders)
   if command=="StartX":
       print("StartX")
       experiment.startx()
       removecommandrequest = requests.post(url+"/commands",
"StartX".encode('utf-8'),headers=httpheaders)
   if command=="JoinVPN":
       print("JoinVPN")
       vpnrequest = requests.get(url+"/vpnlocation", headers=httpheaders)
       vpn = vpnrequest.text
       experiment.joinVPN(vpn)
       removecommandrequest = requests.post(url+"/commands",
"JoinVPN".encode('utf-8'),headers=httpheaders)
   if command=="LaunchFirefox":
       print("LaunchFirefox")
       experiment.launchFirefox()
       removecommandrequest = requests.post(url+"/commands",
"LaunchFirefox".encode('utf-8'),headers=httpheaders)
   if command=="RunAutoCompleteTest":
       print("RunAutoCompleteTest")
       stimulusrequest = requests.get(url+"/stimulus", headers=httpheaders)
       stimulus = stimulusrequest.text
       text = experiment.runAutoCompleteTest(stimulus)
       resultrequest = requests.post(url+"/results",text.encode('utf-8'),
headers=httpheaders)
       removecommandrequest = requests.post(url+"/commands",
"RunAutoCompleteTest".encode('utf-8'),headers=httpheaders)
   if command=="CloseFirefox":
       print("CloseFirefox")
       experiment.closeFirefox()
       removecommandrequest = requests.post(url+"/commands",
"CloseFirefox".encode('utf-8'),headers=httpheaders)
   if command=="LeaveVPN":
       print("LeaveVPN")
       experiment.leaveVPN()
       removecommandrequest = requests.post(url+"/commands",
"LeaveVPN".encode('utf-8'),headers=httpheaders)
   if command=="ShutdownLinux":
       print("ShutdownLinux")
       removecommandrequest = requests.post(url+"/commands",
"ShutdownLinux".encode('utf-8'),headers=httpheaders)
       continueprocessing = False
       os.system("shutdown now")
   time.sleep(5)
```

Work Sample #2 - Experimental Preparation

```
# ©Macauley Lim 2021 -- File Licensed Under The GNU GPLv3. See The Full Notice In
License.md For Binding Terms.
import sys, MASsettings, argparse, time
from tools import pchelper, service_instance
#VMware imports
from pyVmomi import vim
from pyVim.task import WaitForTask
from tools.tasks import wait_for_tasks
```

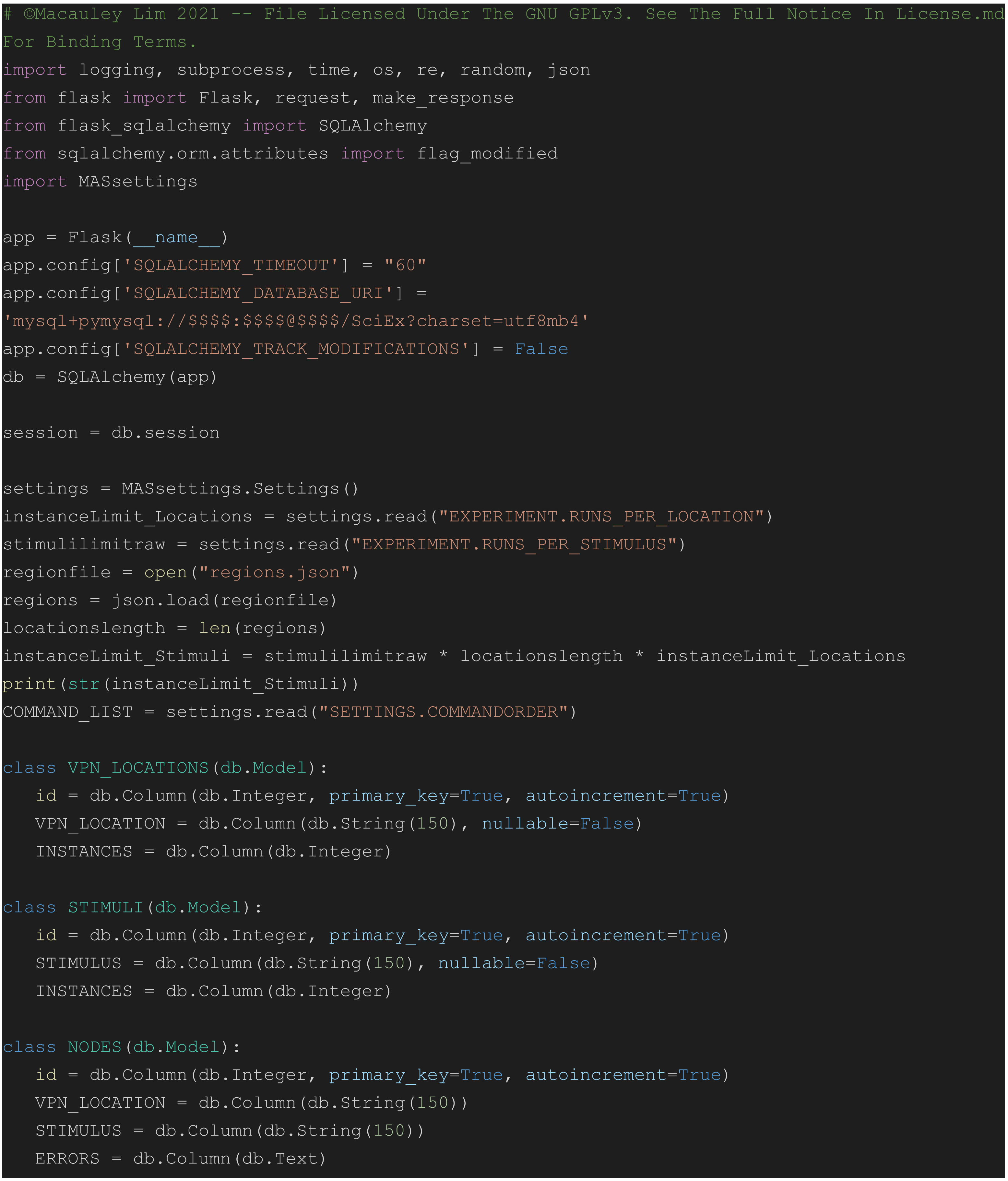
```python
settings = MASsettings.Settings()

VMPREFIX = settings.read("SETTINGS.EXSI.VMPREFIX")
VMRAM = settings.read("SETTINGS.EXSI.RAM")
VMCPU = settings.read("SETTINGS.EXSI.CPU")
VMDATA = settings.read("SETTINGS.EXSI.DATASTORE")
VMNET = settings.read("SETTINGS.EXSI.NETWORK")
VMDC = settings.read("SETTINGS.EXSI.DATACENTER")
VMISO = settings.read("SETTINGS.EXSI.ISOPATH")

args = argparse.ArgumentParser()
args.host = settings.read("SETTINGS.EXSI.IP")
args.user = settings.read("SETTINGS.EXSI.USER")
args.password = settings.read("SETTINGS.EXSI.PASS")
args.port = 443
args.disable_ssl_verification = True

si = service_instance.connect(args)

def create_config_spec(datastore_name, name):
    config = vim.vm.ConfigSpec()
    config.annotation = "Science Extension Project Virtual Machine. Do Not Touch 100% Automated"
    config.memoryMB = int(VMRAM)
    config.guestId = "ubuntu64Guest"
    config.name = name
    config.numCPUs = int(VMCPU)
    files = vim.vm.FileInfo()
    files.vmPathName = "["+datastore_name+"]"
    config.files = files
    return config

def GenerateVM(id):
    content = si.RetrieveContent()
    destination_host = pchelper.get_obj(content, [vim.HostSystem], "exsi1.local.HOME.NET")
    source_pool = destination_host.parent.resourcePool
    global VMDATA
    if VMDATA is None:
        VMDATA = destination_host.datastore[0].name
    config = create_config_spec(datastore_name=VMDATA, name=VMPREFIX+str(id))
    for child in content.rootFolder.childEntity:
        if child.name == VMDC:
            vm_folder = child.vmFolder
            break
        else:
            print("Datacenter %s not found!" % VMDC)
            sys.exit(1)
    try:
        WaitForTask(vm_folder.CreateVm(config, pool=source_pool, host=destination_host))
        print("VM created: %s" % id)
    except vim.fault.DuplicateName:
        print("VM duplicate name: %s" % id, file=sys.stderr)
    except vim.fault.AlreadyExists:
        print("VM name %s already exists." % id, file=sys.stderr)

def AddNIC(id):
    """
    :param si: Service Instance
    :param vm: Virtual Machine Object
    :param network_name: Name of the Virtual Network
    """
    network_name = VMNET
    content = si.RetrieveContent()
    search_index = si.content.searchIndex
    for child in content.rootFolder.childEntity:
        if child.name == VMDC:
            dc = child.vmFolder  # child is a datacenter
            break
        else:
            print("Datacenter %s not found!" % VMDC)
            sys.exit(1)

    vm = search_index.FindChild(dc, VMPREFIX+str(id))

    spec = vim.vm.ConfigSpec()
    nic_changes = []

    nic_spec = vim.vm.device.VirtualDeviceSpec()
    nic_spec.operation = vim.vm.device.VirtualDeviceSpec.Operation.add

    nic_spec.device = vim.vm.device.VirtualE1000()

    nic_spec.device.deviceInfo = vim.Description()
    nic_spec.device.deviceInfo.summary = 'MASnet'
    network = pchelper.get_obj(content, [vim.Network], network_name)
    if isinstance(network, vim.OpaqueNetwork):
        nic_spec.device.backing = \
            vim.vm.device.VirtualEthernetCard.OpaqueNetworkBackingInfo()
        nic_spec.device.backing.opaqueNetworkType = \
            network.summary.opaqueNetworkType
        nic_spec.device.backing.opaqueNetworkId = \
            network.summary.opaqueNetworkId
    else:
        nic_spec.device.backing = \
            vim.vm.device.VirtualEthernetCard.NetworkBackingInfo()
        nic_spec.device.backing.useAutoDetect = False
        nic_spec.device.backing.deviceName = network_name

    nic_spec.device.connectable = vim.vm.device.VirtualDevice.ConnectInfo()
    nic_spec.device.connectable.startConnected = True
    nic_spec.device.connectable.allowGuestControl = True
    nic_spec.device.connectable.connected = False
    nic_spec.device.connectable.status = 'untried'
    nic_spec.device.wakeOnLanEnabled = True
    nic_spec.device.addressType = 'assigned'
    nic_spec.device.addressType = vim.vm.device.VirtualEthernetCardOption.MacTypes.generated

    nic_changes.append(nic_spec)
    spec.deviceChange = nic_changes
    try:
        WaitForTask(vm.ReconfigVM_Task(spec=spec))
    except Exception as e:
        print(e)

    print("NIC CARD ADDED")

def new_cdrom_spec(controller_key, backing):
    connectable = vim.vm.device.VirtualDevice.ConnectInfo()
    connectable.allowGuestControl = True
    connectable.startConnected = True

    cdrom = vim.vm.device.VirtualCdrom()
    cdrom.controllerKey = controller_key
    cdrom.key = -1
    cdrom.connectable = connectable
    cdrom.backing = backing
    return cdrom
```

```python
def find_free_ide_controller(vm):
    for dev in vm.config.hardware.device:
        if isinstance(dev, vim.vm.device.VirtualIDEController):
            # If there are less than 2 devices attached, we can use it.
            if len(dev.device) < 2:
                return dev
    return None

def find_device(vm, device_type):
    result = []
    for dev in vm.config.hardware.device:
        if isinstance(dev, device_type):
            result.append(dev)
    return result

def AddCD(id):
    iso = VMISO
    content = si.RetrieveContent()
    search_index = si.content.searchIndex
    for child in content.rootFolder.childEntity:
        if child.name == "ha-datacenter":
            dc = child.vmFolder  # child is a datacenter
            break
        else:
            print("Datacenter %s not found!" % "ha-datacenter")
            sys.exit(1)
    vm = search_index.FindChild(dc, VMPREFIX+str(id))
    cdrom_operation = vim.vm.device.VirtualDeviceSpec.Operation
    if iso is not None:
        device_spec = vim.vm.device.VirtualDeviceSpec()
        controller = find_free_ide_controller(vm)
        if controller is None:
            raise Exception('Failed to find a free slot on the IDE controller')
        backing = vim.vm.device.VirtualCdrom.IsoBackingInfo(fileName=iso)
        cdrom = new_cdrom_spec(controller.key, backing)
        device_spec.operation = cdrom_operation.add
        device_spec.device = cdrom
        config_spec = vim.vm.ConfigSpec(deviceChange=[device_spec])
        WaitForTask(vm.Reconfigure(config_spec))

        cdroms = find_device(vm, vim.vm.device.VirtualCdrom)
        cdrom = next(filter(lambda x: type(x.backing) == type(backing) and
                    x.backing.fileName == iso, cdroms))
        print("Added CD Rom")
    else:
        print('Skipping ISO test as no iso provided.')

def PowerOnVM(id):
    vmnames = [VMPREFIX+str(id)]
    content = si.RetrieveContent()
    obj_view = content.viewManager.CreateContainerView(content.rootFolder,
                                                       [vim.VirtualMachine],
                                                       True)
    vm_list = obj_view.view
    obj_view.Destroy()

    tasks = [vm.PowerOn() for vm in vm_list if vm.name in vmnames]

    wait_for_tasks(si, tasks)

def GetVMIPS(id):
    content = si.RetrieveContent()
    search_index = si.content.searchIndex
    for child in content.rootFolder.childEntity:
        if child.name == VMDC:
            dc = child.vmFolder  # child is a datacenter
            break
        else:
            print("Datacenter %s not found!" % VMDC)
            sys.exit(1)
    vm = search_index.FindChild(dc, VMPREFIX+str(id))
    ips = []
    for nic in vm.guest.net:
        addresses = nic.ipConfig.ipAddress
        for adr in addresses:
            ips.append(adr.ipAddress)
    return ips

def DeleteVM(id):
    print("Deleted VM")
    content = si.RetrieveContent()
    VM = pchelper.get_obj(content, [vim.VirtualMachine], VMPREFIX+str(id))
    wait_for_tasks(si, [VM.PowerOff()])
    time.sleep(3)
    wait_for_tasks(si, [VM.Destroy_Task()])
```

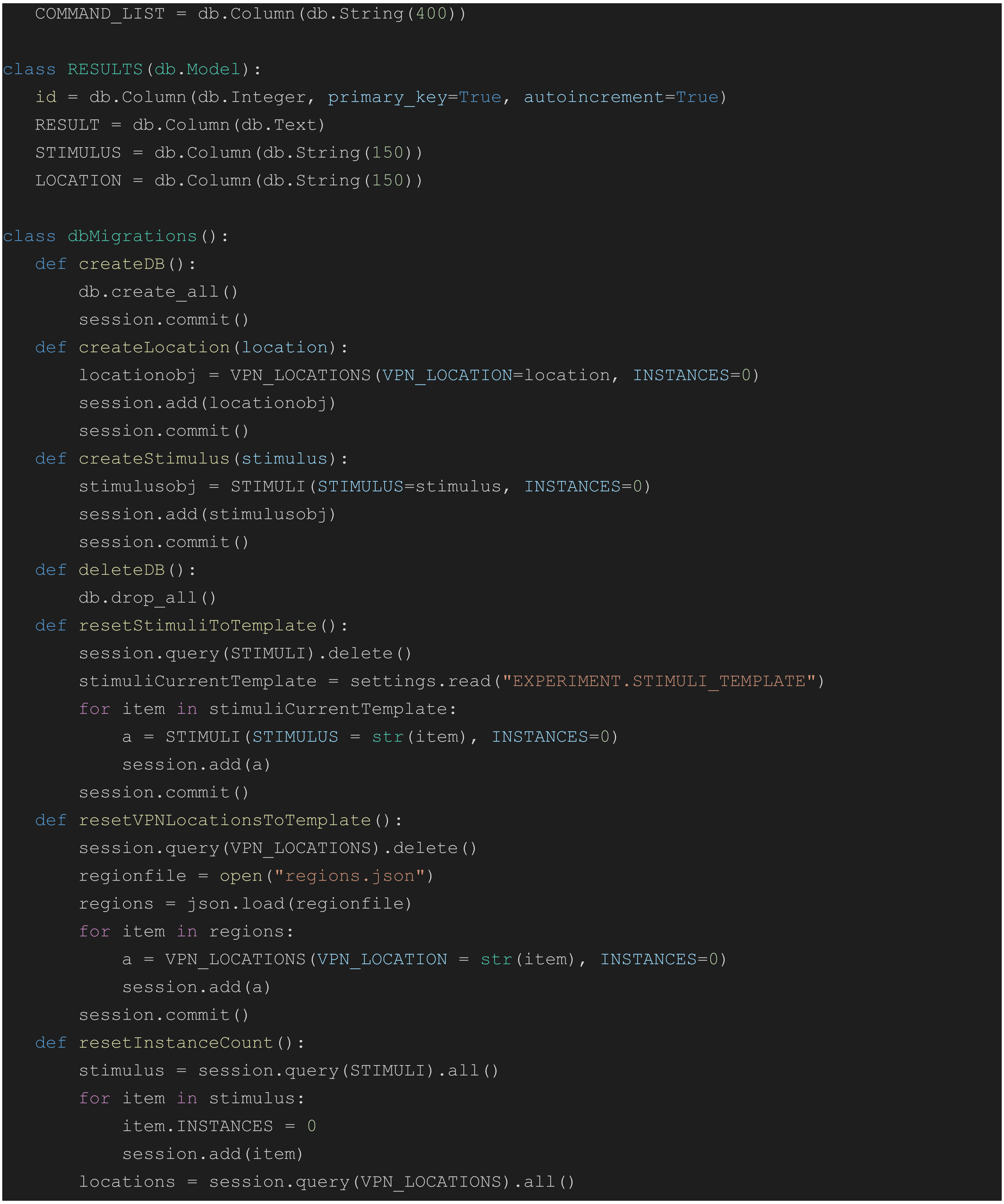
```python
if __name__ == '__main__':
   #GenerateVM(1)
   AddNIC(1)
   AddCD(1)
   PowerOnVM(1)
```

```python
# ©Macauley Lim 2021 -- File Licensed Under The GNU GPLv3. See The Full Notice In
License.md For Binding Terms.
import os, time
from fabric import Connection, Config
from MASsettings import Settings
from MASexsi import GetVMIPS

settings = Settings()
sshconfig = Config(overrides={'sudo': {'password': ""}})

def PushPayload(ID):
   IP = GetVMIPS(ID)[0]
   print("Pushing Payload for ID: "+str(ID)+" @ IP: "+str(IP))
   ssh = Connection(IP, user=settings.read("SETTINGS.VM.USER"),
connect_kwargs={"key_filename":settings.read("SETTINGS.VM.SSH_KEY")}, config=sshconfig)
   ssh.put("/home/mac/SciEx-AutoCompleteByLocation/client.tar", "/home/ubuntu/")
   try:
       ssh.run("mkdir MASclient")
   except Exception:
       print("Folder already Exists")
   ssh.run("tar -xvf client.tar -C MASclient")
   ssh.run("echo 'nameserver 10.1.1.1' | sudo tee -a /etc/resolv.conf")
   ssh.run('cd '+settings.read("SETTINGS.VM.CLIENT_PATH"))
   time.sleep(10)
   ssh.run("sudo apt install ca-certificates")
   try:
       ssh.run('sudo apt update')
   except Exception:
       try:
           time.sleep(10)
           ssh.run('sudo apt update')
       except Exception:
           try:
               time.sleep(10)
               ssh.run('sudo apt update')
           except Exception:
               try:
                   time.sleep(10)
                   ssh.run('sudo apt update')
               except Exception:
                   try:
                       time.sleep(10)
                       ssh.run('sudo apt update')
                   except Exception:
                       time.sleep(10)
                       ssh.run('sudo apt update')
   ssh.run('sudo apt install -yq python3 dbus-x11 screen python3-pip python3-tk python3-dev
tesseract-ocr libtesseract-dev firefox xorg openbox openvpn')
   try:
       ssh.run('sudo rm get-pip.py')
   except Exception:
       print("Did not need to remove get-pip.py")
   ssh.run('wget https://bootstrap.pypa.io/pip/3.5/get-pip.py')
   ssh.run('python3 get-pip.py')
   ssh.run('sudo pip3 install --upgrade pip==20.3.4')
   ssh.run('sudo pip3 install pytesseract==0.2.2 numpy==1.18.5 requests six opencv-python
tesseract pyautogui --no-cache-dir')
   try:
       ssh.run('sudo killall startx && sudo killall openbox')
   except Exception:
       print("Startx and openbox kill not needed.")
   time.sleep(1)
   ssh.run('sudo startx 1>/dev/null 2>/dev/null &')
   ssh.run('sudo sysctl -w net.ipv6.conf.all.disable_ipv6=1')
   ssh.run('sudo sysctl -w net.ipv6.conf.default.disable_ipv6=1')
   ssh.run('sudo touch /root/.Xauthority')
   time.sleep(3)
   try:
       ssh.run('export DISPLAY=:0 && export HOME="/root/" && cd /home/ubuntu/MASclient &&
sudo python3 MASclient.py '+str(ID), pty=True)
   except Exception:
       return
   #ssh.run('export DISPLAY=:0 && export HOME="/root/" && cd /home/ubuntu/MASclient && sudo
python3 MASclient.py '+str(ID)+' 1>/dev/null 2>/dev/null &', pty=True)

if __name__ == "__main__":
   PushPayload(1)
```

Work Sample #3 - Organizing Result Collection



```
# ©Macauley Lim 2021 -- File Licensed Under The GNU GPLv3. See The Full Notice In License.md
For Binding Terms.
import logging, subprocess, time, os, re, random, json
from flask import Flask, request, make_response
from flask_sqlalchemy import SQLAlchemy
from sqlalchemy.orm.attributes import flag_modified
import MASsettings

app = Flask(__name__)
app.config['SQLALCHEMY_TIMEOUT'] = "60"
app.config['SQLALCHEMY_DATABASE_URI'] =
'mysql+pymysql://$$$$:$$$$@$$$$/SciEx?charset=utf8mb4'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

session = db.session

settings = MASsettings.Settings()
instanceLimit_Locations = settings.read("EXPERIMENT.RUNS_PER_LOCATION")
stimulilimitraw = settings.read("EXPERIMENT.RUNS_PER_STIMULUS")
regionfile = open("regions.json")
regions = json.load(regionfile)
locationslength = len(regions)
instanceLimit_Stimuli = stimulilimitraw * locationslength * instanceLimit_Locations
print(str(instanceLimit_Stimuli))
COMMAND_LIST = settings.read("SETTINGS.COMMANDORDER")

class VPN_LOCATIONS(db.Model):
   id = db.Column(db.Integer, primary_key=True, autoincrement=True)
   VPN_LOCATION = db.Column(db.String(150), nullable=False)
   INSTANCES = db.Column(db.Integer)

class STIMULI(db.Model):
   id = db.Column(db.Integer, primary_key=True, autoincrement=True)
   STIMULUS = db.Column(db.String(150), nullable=False)
   INSTANCES = db.Column(db.Integer)

class NODES(db.Model):
   id = db.Column(db.Integer, primary_key=True, autoincrement=True)
   VPN_LOCATION = db.Column(db.String(150))
   STIMULUS = db.Column(db.String(150))
   ERRORS = db.Column(db.Text)
```



```python
    COMMAND_LIST = db.Column(db.String(400))

class RESULTS(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=True)
    RESULT = db.Column(db.Text)
    STIMULUS = db.Column(db.String(150))
    LOCATION = db.Column(db.String(150))

class dbMigrations():
    def createDB():
        db.create_all()
        session.commit()
    def createLocation(location):
        locationobj = VPN_LOCATIONS(VPN_LOCATION=location, INSTANCES=0)
        session.add(locationobj)
        session.commit()
    def createStimulus(stimulus):
        stimulusobj = STIMULI(STIMULUS=stimulus, INSTANCES=0)
        session.add(stimulusobj)
        session.commit()
    def deleteDB():
        db.drop_all()
    def resetStimuliToTemplate():
        session.query(STIMULI).delete()
        stimuliCurrentTemplate = settings.read("EXPERIMENT.STIMULI_TEMPLATE")
        for item in stimuliCurrentTemplate:
            a = STIMULI(STIMULUS = str(item), INSTANCES=0)
            session.add(a)
        session.commit()
    def resetVPNLocationsToTemplate():
        session.query(VPN_LOCATIONS).delete()
        regionfile = open("regions.json")
        regions = json.load(regionfile)
        for item in regions:
            a = VPN_LOCATIONS(VPN_LOCATION = str(item), INSTANCES=0)
            session.add(a)
        session.commit()
    def resetInstanceCount():
        stimulus = session.query(STIMULI).all()
        for item in stimulus:
            item.INSTANCES = 0
            session.add(item)
        locations = session.query(VPN_LOCATIONS).all()
```

```
       for item in locations:
           item.INSTANCES = 0
           session.add(item)
       session.commit()

#admin routes
@app.route("/admin/createDB")
def handleAdminNewDB():
   dbMigrations.createDB()
   dbMigrations.resetStimuliToTemplate()
   dbMigrations.resetVPNLocationsToTemplate()
   dbMigrations.resetInstanceCount()
   return "OK"

@app.route("/admin/createLocation", methods=['POST'])
def handleAdminNewLocation():
   dbMigrations.createLocation(request.get_data().decode('utf-8'))
   return "OK"

@app.route("/admin/createStimulus", methods=['POST'])
def handleAdminNewStimulus():
   dbMigrations.createStimulus(request.get_data().decode('utf-8'))
   return "OK"

@app.route("/admin/deleteDB")
def handleAdminDeleteDB():
   dbMigrations.deleteDB()
   return "OK"

@app.route("/admin/resetStimuliToTemplate")
def handleAdminResetStimuliToTemplate():
   dbMigrations.resetStimuliToTemplate()
   return "OK"

@app.route("/admin/resetVPNLocationsToTemplate")
def handleAdminResetVPNToTemplate():
   dbMigrations.resetVPNLocationsToTemplate()
   return "OK"

@app.route("/admin/resetInstanceCount")
def handleAdminDeleteResetInstanceCount():
   dbMigrations.resetInstanceCount()
   return "OK"
```

```python
#MASexsi routes



#MASclient routes
@app.route("/node")
def handleNewNode():
    vpn_locations = session.query(VPN_LOCATIONS).all()
    location = ""
    while True:
        location = random.choice(vpn_locations)
        if location.INSTANCES < instanceLimit_Locations:
            location.INSTANCES = location.INSTANCES + 1
            session.add(location)
            session.commit()
            location = location.VPN_LOCATION
            break
    stimuli = session.query(STIMULI).all()
    stimulus = ""
    while True:
        stimulus = random.choice(stimuli)
        if stimulus.INSTANCES < instanceLimit_Stimuli:
            stimulus.INSTANCES = stimulus.INSTANCES + 1
            session.add(stimulus)
            session.commit()
            stimulus = stimulus.STIMULUS
            break
    nodeCommand_List = COMMAND_LIST
    nodedbi = NODES(VPN_LOCATION = location, STIMULUS = stimulus, ERRORS = "", COMMAND_LIST = nodeCommand_List)
    session.add(nodedbi)
    session.commit()
    nodeid = nodedbi.id
    return str(nodeid)

@app.route("/results", methods=["POST"])
def handleSentResults():
    nodeid = request.headers.get("X-NODE-ID")
    result = request.get_data().decode('utf-8')
    nodedetails = session.query(NODES).filter(NODES.id == nodeid).first()
    stimulus = nodedetails.STIMULUS
    location = nodedetails.VPN_LOCATION
    dbobj = RESULTS(id = nodeid, RESULT = result, STIMULUS = stimulus, LOCATION = location)
    session.add(dbobj)
    session.commit()
    return "OK"

@app.route("/aptpackages")
def handleGetPackagesApt():
    rtext = ",".join(settings.read("SETTINGS.VM.APTPACKAGES"))
    response = make_response(rtext, 200)
    response.mimetype = "text/plain"
    return response

@app.route("/pippackages")
def handleGetPackagesPip():
    rtext = ",".join(settings.read("SETTINGS.VM.PIPPACKAGES"))
    response = make_response(rtext, 200)
    response.mimetype = "text/plain"
    return response

@app.route("/error", methods=["POST"])
def handleError():
    nodeid = request.headers.get("X-NODE-ID")
    nodedetails = session.query(NODES).filter(NODES.id == nodeid).first()
    nodedetails.ERRORS = nodedetails.ERRORS + ", "+request.get_data().decode('utf-8')
    session.add(nodedetails)
    session.commit()
    return "OK"

@app.route("/commands", methods=["GET","POST"])
def handleGetCommands():
    session.commit()
    if request.method == "GET":
        nodeid = request.headers.get("X-NODE-ID")
        print(nodeid)
        nodedetails = session.query(NODES).filter(NODES.id == nodeid).first()
        session.rollback()
        return nodedetails.COMMAND_LIST
    if request.method == "POST":
        nodeid = request.headers.get("X-NODE-ID")
        print(nodeid)
        nodedetails = session.query(NODES).filter_by(id=nodeid).first()
        commandlist = nodedetails.COMMAND_LIST.split(",")
        commandlist.remove(request.get_data().decode('utf-8'))
        nodedetails.COMMAND_LIST = ",".join(commandlist)
        print(nodedetails.COMMAND_LIST)
        session.add(nodedetails)
        session.commit()
        return "OK"

 @app.route("/stimulus")
 def handleGetStimulus():
    nodeid = request.headers.get("X-NODE-ID")
    nodedetails = session.query(NODES).filter(NODES.id == nodeid).first()
    return nodedetails.STIMULUS

 @app.route("/vpnlocation")
 def handleGetLocation():
    nodeid = request.headers.get("X-NODE-ID")
    nodedetails = session.query(NODES).filter(NODES.id == nodeid).first()
    return nodedetails.VPN_LOCATION
```

Work Sample #4 - Result Processing

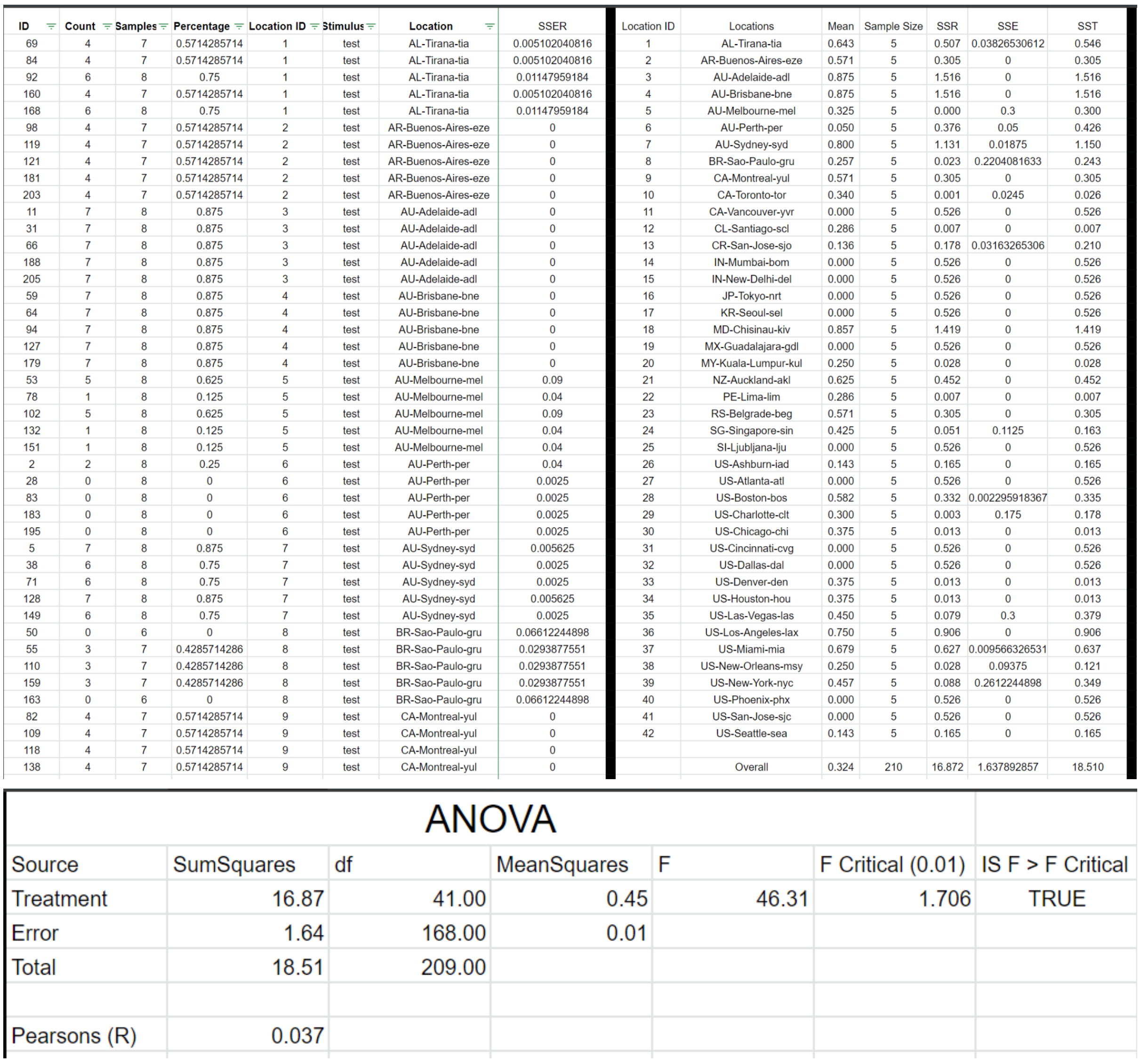
```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, Text, MetaData
import sqlalchemy, json
import re
import csv, time
engine = create_engine("mysql+pymysql://$$$$$:$$$$@$$$/SciEx?charset=utf8mb4", echo=True)
sessionm = sessionmaker(bind=engine)
base = declarative_base()

class VPN_LOCATIONS(base):
    __tablename__ = "VPN_LOCATIONS"

    id = Column(Integer, primary_key=True, autoincrement=True)
    VPN_LOCATION = Column(String(150), nullable=False)
    INSTANCES = Column(Integer)

class STIMULI(base):
    __tablename__ = "STIMULI"

    id = Column(Integer, primary_key=True, autoincrement=True)
    STIMULUS = Column(String(150), nullable=False)
    INSTANCES = Column(Integer)
```

```python
class NODES(base):
    __tablename__ = "NODES"

    id = Column(Integer, primary_key=True, autoincrement=True)
    VPN_LOCATION = Column(String(150))
    STIMULUS = Column(String(150))
    ERRORS = Column(Text)
    COMMAND_LIST = Column(String(400))

class RESULTS(base):
    __tablename__ = "RESULTS"

    id = Column(Integer, primary_key=True, autoincrement=True)
    RESULT = Column(Text)
    STIMULUS = Column(String(150))
    LOCATION = Column(String(150))

class COUNTRESULT(base):
    __tablename__ = "LINERESULTS"

    id = Column(Integer, primary_key=True, autoincrement=True)
    RESULTID = Column(Integer)
    KEYCOUNT = Column(Integer)
    LINES = Column(Integer)
    STIMULUS = Column(String(150))
    LOCATION = Column(String(150))

base.metadata.create_all(engine)

keyterms = ["tested", "positive", "testing", "isolation", "covid", "covld", "clinic",
"pcr"]
locationcount = {}
locationcountall = {}

session = sessionm()
results = session.query(RESULTS).all()

for item in results:
    itemcount = 0
    itemlines = 0
    itemlines += len(re.findall("(.+)", item.RESULT))
    try:
        locationcountall[item.LOCATION] = locationcountall[item.LOCATION] +
len(re.findall("(.+)", item.RESULT))
    except KeyError:
        locationcountall[item.LOCATION] = 1
    for string in keyterms:
        if string in str(item.RESULT):
            itemcount += len(re.findall(string, item.RESULT))
            try:
                locationcount[item.LOCATION] = locationcount[item.LOCATION] +
len(re.findall(string, item.RESULT))
            except KeyError:
                locationcount[item.LOCATION] = 1
    temp = COUNTRESULT(RESULTID = item.id, KEYCOUNT = itemcount, LINES = itemlines,
STIMULUS = item.STIMULUS, LOCATION = item.LOCATION)
    session.add(temp)
    session.commit()

print(f"{len(results)} Autocomplete Searches processed.")

results = {}
country_mapper = {
    "AE-Dubai-dxb": ["","","United Arab Emirates"],
    "AL-Tirana-tia": ["","","Albania"],
    "AR-Buenos-Aires-eze": ["","", "Argentina"],
    "AU-Adelaide-adl": ["","South Australia", "Australia"],
    "AU-Brisbane-bne": ["","Queensland", "Australia"],
    "AU-Melbourne-mel": ["","Victoria", "Australia"],
    "AU-Perth-per": ["","Western Australia", "Australia"],
    "AU-Sydney-syd": ["","New South Wales", "Australia"],
    "BR-Sao-Paulo-gru": ["","Sao Paulo", "Brazil"],
    "CA-Montreal-yul": ["","Quebec", "Canada"],
    "CA-Toronto-tor": ["","Ontario", "Canada"],
    "CA-Vancouver-yvr": ["","British Columbia", "Canada"],
    "CH-Zurich-zrh": ["","","Switzerland"],
    "CL-Santiago-scl": ["","Metropolitana", "Chile"],
    "CR-San-Jose-sjo": ["","", "Costa Rica"],
    "FI-Helsinki-hel": ["","", "Finland"],
    "HR-Zagreb-zag": ["","", "Croatia"],
    "IN-Mumbai-bom": ["","Maharashtra", "India"],
    "IN-New-Delhi-del": ["","Delhi", "India"],
    "IS-Reykjavik-rkv": ["","", "Iceland"],
    "JP-Tokyo-nrt": ["","Tokyo", "Japan"],
    "KR-Seoul-sel": ["","", "Korea, South"],
    "MD-Chisinau-kiv": ["","", "Moldova"],
    "MX-Guadalajara-gdl": ["","Jalisco", "Mexico"],
    "MY-Kuala-Lumpur-kul": ["","W.P. Kuala Lumpur", "Malaysia"],
    "NZ-Auckland-akl": ["","", "New Zealand"],
    "PE-Lima-lim": ["","Lima", "Peru"],
    "RO-Bucharest-otp": ["","","Romania"],
    "RS-Belgrade-beg": ["","", "Serbia"],
    "SG-Singapore-sin": ["","","Singapore"],
    "SI-Ljubljana-lju": ["","", "Slovenia"],
    "US-Ashburn-iad": ["Loudoun", "Virginia", "US"],
    "US-Atlanta-atl": ["Fulton", "Georgia", "US"],
    "US-Boston-bos": ["Suffolk", "Massachusetts", "US"],
    "US-Charlotte-clt": ["Mecklenburg", "North Carolina", "US"],
    "US-Chicago-chi": ["Cook", "Illinois", "US"],
    "US-Cincinnati-cvg": ["Hamilton", "Ohio", "US"],
    "US-Dallas-dal": ["Dallas", "Texas", "US"],
    "US-Denver-den": ["Denver", "Colorado", "US"],
    "US-Houston-hou": ["Harris", "Texas", "US"],
    "US-Las-Vegas-las": ["Clark", "Nevada", "US"],
    "US-Los-Angeles-lax": ["Los Angeles", "California", "US"],
    "US-Miami-mia": ["Miami-Dade", "Florida", "US"],
    "US-New-Orleans-msy": ["Orleans", "Louisiana", "US"],
    "US-New-York-nyc": ["New York", "New York", "US"],
    "US-Phoenix-phx": ["Maricopa", "Arizona", "US"],
    "US-San-Jose-sjc": ["Santa Clara", "California", "US"],
    "US-Seattle-sea": ["King", "Washington", "US"],
}

covidFile = open("01-28-2022.csv", "r")
covidData = csv.reader(covidFile)
locList = []
outputFile = open("correlation.csv", "w", newline='')
outputCSV = csv.writer(outputFile, delimiter= ",", quoting=csv.QUOTE_MINIMAL)
outputCSV.writerow(["Location", "Count", "Samples", "Percentage", "InfectionRate"])

for row in covidData:
    locList.append(row)

for key in sorted(locationcount, reverse=True):
    value = locationcount[key]
    totalcount = locationcountall[key]
    percentage = value/totalcount
    admin2 = country_mapper[key][0]
```

```python
    province_state = country_mapper[key][1]
    country = country_mapper[key][2]
    row_final = None
    #print(f"Country: {country}; Province/State: {province_state}; LGA/Locality: {admin2}")
    for row in locList:
        if row[3] == country:
            if  row[1] == admin2:
                if  row[2] == province_state:
                    row_final = row
    #print(f"{key}: {value} out of {totalcount}: {percentage}")
    outputCSV.writerow([key, value, totalcount, percentage, row_final[12]])

lineresult = session.query(COUNTRESULT).all()
outputFile.close()

rawFile = open("raw.csv", "w", newline='')
rawCSV = csv.writer(rawFile, delimiter= ",", quoting=csv.QUOTE_MINIMAL)
rawCSV.writerow(["ID", "Count", "Samples", "Stimulus", "Location"])

for item in lineresult:
    rawCSV.writerow([item.id, item.KEYCOUNT, item.LINES, item.STIMULUS, item.LOCATION])

time.sleep(5)
rawFile.close()
```

```python
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import Column, Integer, String, Text
import sqlalchemy, json
import re
engine = create_engine("mysql+pymysql://$$$$$:$$$$@$$$/SciEx?charset=utf8mb4", echo=True)
sessionm = sessionmaker(bind=engine)
base = declarative_base()

class VPN_LOCATIONS(base):
    __tablename__ = "VPN_LOCATIONS"

    id = Column(Integer, primary_key=True, autoincrement=True)
    VPN_LOCATION = Column(String(150), nullable=False)
    INSTANCES = Column(Integer)

class STIMULI(base):
    __tablename__ = "STIMULI"

    id = Column(Integer, primary_key=True, autoincrement=True)
    STIMULUS = Column(String(150), nullable=False)
    INSTANCES = Column(Integer)

class NODES(base):
    __tablename__ = "NODES"

    id = Column(Integer, primary_key=True, autoincrement=True)
    VPN_LOCATION = Column(String(150))
    STIMULUS = Column(String(150))
    ERRORS = Column(Text)
    COMMAND_LIST = Column(String(400))

class RESULTS(base):
    __tablename__ = "RESULTS"

    id = Column(Integer, primary_key=True, autoincrement=True)
    RESULT = Column(Text)
    STIMULUS = Column(String(150))
    LOCATION = Column(String(150))

countdict = {}
totalcount = 0

session = sessionm()

results = session.query(RESULTS).all()
for item in results:
    a = re.findall("(.+)", item.RESULT)
    for match in a:
        if "test" in match:
            totalcount = totalcount + 1
            try:
                countdict[match] = countdict[match] + 1
            except KeyError:
                countdict[match] = 1

print(totalcount)

for item in sorted(countdict, key=countdict.get, reverse=True):
    text = str(countdict[item])
    item = item.strip("\r")
    print(item + ": " + text)
```

Work Sample #5 - Data Analysis



ID	Count	Samples	Percentage	Location ID	Stimulus	Location	SSER
69	4	7	0.5714285714	1	test	AL-Tirana-tia	0.005102040816
84	4	7	0.5714285714	1	test	AL-Tirana-tia	0.005102040816
92	6	8	0.75	1	test	AL-Tirana-tia	0.01147959184
160	4	7	0.5714285714	1	test	AL-Tirana-tia	0.005102040816
168	6	8	0.75	1	test	AL-Tirana-tia	0.01147959184
98	4	7	0.5714285714	2	test	AR-Buenos-Aires-eze	0
119	4	7	0.5714285714	2	test	AR-Buenos-Aires-eze	0
121	4	7	0.5714285714	2	test	AR-Buenos-Aires-eze	0
181	4	7	0.5714285714	2	test	AR-Buenos-Aires-eze	0
203	4	7	0.5714285714	2	test	AR-Buenos-Aires-eze	0
11	7	8	0.875	3	test	AU-Adelaide-adl	0
31	7	8	0.875	3	test	AU-Adelaide-adl	0
66	7	8	0.875	3	test	AU-Adelaide-adl	0
188	7	8	0.875	3	test	AU-Adelaide-adl	0
205	7	8	0.875	3	test	AU-Adelaide-adl	0
59	7	8	0.875	4	test	AU-Brisbane-bne	0
64	7	8	0.875	4	test	AU-Brisbane-bne	0
94	7	8	0.875	4	test	AU-Brisbane-bne	0
127	7	8	0.875	4	test	AU-Brisbane-bne	0
179	7	8	0.875	4	test	AU-Brisbane-bne	0
53	5	8	0.625	5	test	AU-Melbourne-mel	0.09
78	1	8	0.125	5	test	AU-Melbourne-mel	0.04
102	5	8	0.625	5	test	AU-Melbourne-mel	0.09
132	1	8	0.125	5	test	AU-Melbourne-mel	0.04
151	1	8	0.125	5	test	AU-Melbourne-mel	0.04
2	2	8	0.25	6	test	AU-Perth-per	0.04
28	0	8	0	6	test	AU-Perth-per	0.0025
83	0	8	0	6	test	AU-Perth-per	0.0025
183	0	8	0	6	test	AU-Perth-per	0.0025
195	0	8	0	6	test	AU-Perth-per	0.0025
5	7	8	0.875	7	test	AU-Sydney-syd	0.005625
38	6	8	0.75	7	test	AU-Sydney-syd	0.0025
71	6	8	0.75	7	test	AU-Sydney-syd	0.0025
128	7	8	0.875	7	test	AU-Sydney-syd	0.005625
149	6	8	0.75	7	test	AU-Sydney-syd	0.0025
50	0	6	0	8	test	BR-Sao-Paulo-gru	0.06612244898
55	3	7	0.4285714286	8	test	BR-Sao-Paulo-gru	0.0293877551
110	3	7	0.4285714286	8	test	BR-Sao-Paulo-gru	0.0293877551
159	3	7	0.4285714286	8	test	BR-Sao-Paulo-gru	0.0293877551
163	0	6	0	8	test	BR-Sao-Paulo-gru	0.06612244898
82	4	7	0.5714285714	9	test	CA-Montreal-yul	0
109	4	7	0.5714285714	9	test	CA-Montreal-yul	0
118	4	7	0.5714285714	9	test	CA-Montreal-yul	0
138	4	7	0.5714285714	9	test	CA-Montreal-yul	0

Location ID	Locations	Mean	Sample Size	SSR	SSE	SST
1	AL-Tirana-tia	0.643	5	0.507	0.03826530612	0.546
2	AR-Buenos-Aires-eze	0.571	5	0.305	0	0.305
3	AU-Adelaide-adl	0.875	5	1.516	0	1.516
4	AU-Brisbane-bne	0.875	5	1.516	0	1.516
5	AU-Melbourne-mel	0.325	5	0.000	0.3	0.300
6	AU-Perth-per	0.050	5	0.376	0.05	0.426
7	AU-Sydney-syd	0.800	5	1.131	0.01875	1.150
8	BR-Sao-Paulo-gru	0.257	5	0.023	0.2204081633	0.243
9	CA-Montreal-yul	0.571	5	0.305	0	0.305
10	CA-Toronto-tor	0.340	5	0.001	0.0245	0.026
11	CA-Vancouver-yvr	0.000	5	0.526	0	0.526
12	CL-Santiago-scl	0.286	5	0.007	0	0.007
13	CR-San-Jose-sjo	0.136	5	0.178	0.03163265306	0.210
14	IN-Mumbai-bom	0.000	5	0.526	0	0.526
15	IN-New-Delhi-del	0.000	5	0.526	0	0.526
16	JP-Tokyo-nrt	0.000	5	0.526	0	0.526
17	KR-Seoul-sel	0.000	5	0.526	0	0.526
18	MD-Chisinau-kiv	0.857	5	1.419	0	1.419
19	MX-Guadalajara-gdl	0.000	5	0.526	0	0.526
20	MY-Kuala-Lumpur-kul	0.250	5	0.028	0	0.028
21	NZ-Auckland-akl	0.625	5	0.452	0	0.452
22	PE-Lima-lim	0.286	5	0.007	0	0.007
23	RS-Belgrade-beg	0.571	5	0.305	0	0.305
24	SG-Singapore-sin	0.425	5	0.051	0.1125	0.163
25	SI-Ljubljana-lju	0.000	5	0.526	0	0.526
26	US-Ashburn-iad	0.143	5	0.165	0	0.165
27	US-Atlanta-atl	0.000	5	0.526	0	0.526
28	US-Boston-bos	0.582	5	0.332	0.002295918367	0.335
29	US-Charlotte-clt	0.300	5	0.003	0.175	0.178
30	US-Chicago-chi	0.375	5	0.013	0	0.013
31	US-Cincinnati-cvg	0.000	5	0.526	0	0.526
32	US-Dallas-dal	0.000	5	0.526	0	0.526
33	US-Denver-den	0.375	5	0.013	0	0.013
34	US-Houston-hou	0.375	5	0.013	0	0.013
35	US-Las-Vegas-las	0.450	5	0.079	0.3	0.379
36	US-Los-Angeles-lax	0.750	5	0.906	0	0.906
37	US-Miami-mia	0.679	5	0.627	0.009566326531	0.637
38	US-New-Orleans-msy	0.250	5	0.028	0.09375	0.121
39	US-New-York-nyc	0.457	5	0.088	0.2612244898	0.349
40	US-Phoenix-phx	0.000	5	0.526	0	0.526
41	US-San-Jose-sjc	0.000	5	0.526	0	0.526
42	US-Seattle-sea	0.143	5	0.165	0	0.165
	Overall	0.324	210	16.872	1.637892857	18.510

ANOVA

Source	SumSquares	df	MeanSquares	F	F Critical (0.01)	IS F > F Critical
Treatment	16.87	41.00	0.45	46.31	1.706	TRUE
Error	1.64	168.00	0.01			
Total	18.51	209.00				
Pearsons (R)	0.037					

Section 3 Reflection

Throughout the project, numerous difficulties have been brought across the project. From choosing the appropriate inferential statistics, to working out how to collect 100s of autocomplete results in a small time and finding what to do as a project; challenges have been found, conquered and surpassed. While the best effort has been made to find the right questions and find valid answers, there are ways the project can be improved.

In terms of the methodology, the MAS system is a good start however it needs more work in order to make it more capable for other researchers. For example, a GUI control system and generalized experimental script (MASexperiment) would help researchers perform these kinds of experiments more easily. Installation is also difficult, and could be simplified using Docker or a VMWare virtual appliance container. If more time was available, then these more complicated implementation details could be solved.

As for the COVID-19 key term experiment, the main improvements could be made in finding more direct variables to control. For example, search results themselves could be processed to gauge the prevalence in online media which could then be related.

Overall however the process was fun and enjoyable, demonstrating the full scientific process at work in the context of a high-school science experiment.

Examples of Use Of Scientific Language

In the conclusion paragraph, statistically significant is used as a technical scientific term to demonstrate that the P-value is below the confidence interval specified by the paper.

Pearson's R is a scientific language that demonstrates the use of non-inferential statistics to find correlations between variables.

P-values are scientific language referring to the probability that a group's differences to another group are the product of probability rather than significant difference.

Independent and dependent variables are classifications of variables used to describe what is being tested and measured. Independent variables are the variables changed by the study in the different groups (e.g. control or experimental). Dependent variables are measured in order to determine the effect of this change between groups. Controlled variables are a special kind of variable that are purposefully maintained the same between groups to ensure that only one kind of change occurs between the groups (the independent variable).

The Null and Alternative Hypothesis are terms used to describe the effect of statistical hypothesis testing. The null hypothesis represents no effect or relationship and is categorized by a p-value above the confidence interval required by the paper. Otherwise, the alternative hypothesis is accepted and the findings considered statistically significant. The alternative hypothesis will contain scientific language that states a cause and effect relationship between the independent and dependent variables.

Confidence interval is a scientific term used to describe the probability that two groups of results are within the same larger group of results. Scientists set the confidence interval to state the minimum stand for a dataset that confirms the alternate hypothesis. 95% (P-value 0.05) and 99% (P-value 0.01) are common values used in scientific reports.

These are all examples of scientific language at work in the research project report.