Vizualizacija podataka s Matplotlib bibliotekom

Tomić, Tomislav

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

TOMISLAV TOMIĆ

VIZUALIZACIJA PODATAKA S MATPLOTLIB BIBLIOTEKOM

ZAVRŠNI RAD

Dubrovnik, rujan 2021.

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

VIZUALIZACIJA PODATAKA S MATPLOTLIB BIBLIOTEKOM

ZAVRŠNI RAD

Studij: Primijenjeno/poslovno računarstvo
Kolegij: Uvod u programiranje
Mentor: izv. prof. dr. sc. Mario Miličević
Student: Tomislav Tomić

Dubrovnik, rujan 2021.

SAŽETAK

Tema ovog završnog rada jest vizualizacija podataka u znanosti o podacima. Za vizualizaciju podataka koristit ćemo biblioteku matplotlib u programskom jeziku Python. Podaci koje ćemo vizualizirati vezani su uz pandemiju COVID-19 u Europi i SAD-u. Kroz završni rad objasnit ćemo pojam znanosti o podacima, kako se razvijala u zadnjih nekoliko desetljeća, proći kroz tehnologije koje se koriste u znanosti o podacima te iznijeti prednosti i mane biblioteke matplotlib.

Praktični dio rada sastoji se od prikaza postupka instalacije potrebnih alata na različite operacijske sustave, prikaza i objašnjenja koda potrebnog za uvoz podataka, prikaza i objašnjenja koda potrebnog za prikaz podataka te konačne vizualizacije, odnosno grafa.

Ključne riječi: matplotlib, znanost podataka, Python

ABSTRACT

The topic of this paper is data visualization in data science. To visualize the data we will use matplotlib library of Python programming language. The data we are visualising is related to the COVID-19 pandemic in Europe and the United States. Troughout this paper we will define the data science, its development trough the last few decades, name the technologies used in data science and adress the advantages and drawbacks of matplotlib library.

The practical part of the paper consists of the proces of installation of needed tools on various operating systems, display and explanation of code needed for import and finally - data visualization.

Keywords: matplotlib, data science, Python

SADRŽAJ

1.Uvod

Ubrzani razvoj tehnologije i njezina sve veća dostupnost velikom broju ljudi dovela je do mogućnosti prikupljanja podataka na dosad neviđenoj razini. Velike količine prikupljenih podataka, iako mogu biti korisne pojedincima, vladinim organizacijama i korporacijama, također su donijele i nove izazove, a to su obrada i pretvaranje neobrađenih podataka u informaciju - nešto smisleno iz čega je moguće izvući korist i(li) zaključak, odnosno upotrijebiti u primjerice odlučivanju ili djelovanju.

Kroz povijest, dok su podaci bili prikupljani ručno, obrada se također mogla raditi ručno, ali s automatizacijom prikupljanja podataka nastala je i potreba za automatizacijom obrade jer bi ručna obrada velike količine podataka zahtijevala ogromnu količinu vremena te u velikoj većini slučajeva bila nemoguća i neefikasna. To je pak dovelo do stvaranja novog interdisciplinarnog područja - znanosti o podacima (*engl. Data science.)*

U ovom završnom radu proći ćemo kroz povijest znanosti podataka, objasniti kako se razvijala, spomenuti tehnologije koje su u tu svrhu korištene kroz povijest te posvetiti posebnu pozornost programskom jeziku Python i jednom od njegovih modula za prikaz podataka, matplotlib-u.

2 Znanost podataka

Znanost o podacima je interdisciplinarno polje koje koristi znanstvene metode, algoritme, procese i sisteme da izvuče znanja i zapažanja iz strukturiranih i nestrukturiranih podataka i primijeni znanja i djelotvorna zapažanja dobivenih iz podataka na širok spektar različitih djelatnosti [2].

Znanost o podacima kombinira nekoliko različitih znanstvenih disciplina, uključujući statistiku, umjetnu inteligenciju i računarstvo [3][4].

Znanost o podacima koristi širok spektar tehnologije orijentiran prema podacima, uključujući SQL, Python, R i Hadoop [5] .

2.1 Povijest znanosti podataka

Prvi spomen o znanosti o podacima sežu u 1962. godinu kada je John W. Tukey napisao članak pod imenom „The future of data analysis". U tom članku John W. Tukey predvidio je kakav će utjecaj računala imati u budućnosti na obradu podataka [6][1].

Godine 1974. danski pionir računalnih znanosti Peter Naur izdaje knjigu „Concise Survey of Computer Methods". Knjiga je pregled suvremenih metoda obrade podataka, a i sam se izraz „Znanost o podacima" se prvi put spominje upravo u njoj [1].

Godine 1977. The International Association for Statistical Computing (IASC) osnovana je kao sekcija ISI-a (Internation Statistics Institute). Misija IASC-a je „povezivanje tradicionalne statističke metodologije, moderne računalne tehnologije i znanja stručnjaka u svrhu pretvaranja podataka u informacije i znanje" [1][8] .

U rujnu 1994. američki poslovni časopis „Businessweek" izdaje članak pod nazivom „Database marketing" u kojem se navodi da tvrtke skupljaju velike količine podataka o svojim kupcima, i obrađuju te podatke u nadi da će moći predvidjeti kolika je vjerojatnost da će kupiti neki od njihovih proizvoda. Početni nalet entuzijazma izazvan širenjem naplatnih skenera na izlazu iz trgovina u osamdesetim godinama prošlog stoljeća zamijenilo je opće razočarenje nakon što se ispostavilo da je količina podataka prikupljena, prevelika da se u potpunosti obradi [1].

Godine 1996. Članovi „Internation Federation of Classification Societies" (IFCS) sastaju se u Kobeu u Japanu na konferenciji pod nazivom „Data science, classification, and related methods" koja se održava jednom u dvije godine. Prvi put izraz „Znanost o podacima" uključen je u naziv konferencije. Također 1996., Usama Fayyad, Gregory Piatetsky-Shapiro i Padhraic Smyth objavljuju članak pod nazivom „From Data Mining to Knowledge Discovery in Databases" u časopisu „AI magazine". Autori u njoj navode da je kroz povijest pojmu traženja korisnih uzoraka u podacima davano mnogo različitih naziva, uključujući dubinska analiza podataka, izvlačenje znanja, otkrivanje informacija, berba informacija, arheologija podataka, i obrada uzoraka podataka. U njihovom pogledu KDD (Knowledge Discovery in Databases) odnosi se na cijeli proces otkrivanja korisnih informacija iz podataka, a dubinska analiza podataka odnosi se na određeni korak u ovom procesu. Dubinska analiza podataka primjena je određenih algoritama za izvlačenje uzoraka iz iz podataka, dodatni koraci u KDD procesu, na primjer priprema podataka, izbor

podataka, čišćenje podataka, inkorporacija odgovarajućeg prijašnjeg znanja i pravilno tumačenje rezultata dubinske analize podataka. Ti su koraci nužni da bi se osiguralo dobivanje kvalitetnih informacija iz podataka. Primjenjivanje metoda dubinske analize podataka „na slijepo" može biti opasna aktivnost, koja lako može dovesti do otkrivanja beznačajnih i krivih uzoraka [1][7].

Godine 1997. u svom prvom predavanju na sveučilištu u Michiganu Profesor C.F. Jeff Wu poziva da se statistika preimenuje u Znanost o podacima a statističari u znanstvenike podataka. Također te godine prvi je put izdan časopis „Data Mining and Knwoledge Discovery", a poredak dva pojma u naslovu „Data Mining" i „Knowledge Discovery", reflektira činjenicu kako je dubinska analiza podataka postala popularniji naziv za izvlačenje informacija iz velikih baza podataka [1].

U prosincu 1999. Jacob Zahawi citiran je u članku „Mining Data for Nuggets of Knowledge" časopisa „Knowledge@Wharton", gdje govori da konvencionalne statističke metode rade dobro s malim skupovima podataka, ali ne sa suvremenim bazama podataka koje mogu sadržavati milijune redaka i velike brojeve stupaca. Zahawi smatra da je prilagodljivost veličini podataka veliki problem u dubinskoj analizi podataka, a da je drugi tehnički izazov razvoj modela koji će biti efikasniji u analizi podataka, otkrivanju nelinearnih veza i interakcija između elemenata te da će se morati razviti posebni alati za dubinsku analizu podataka [1][9].

Godine 2001. u članku „Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics", William S. Cleveland navodi da se planira povećati obujam tehničkog dijela rada u polju statistike te da bi takva promjena dovela do stvaranja nove grane znanosti koja bi se nazvala znanost o podacima. Cleveland smatra kako bi bilo korisno spajanje računarstva i statistike, zato što statističari nemaju potrebna tehnička znanja za automatizaciju obrade podataka, dok stručnjaci iz polja računarstva nemaju potrebna znanja iz statistike kako bi znali pristupiti obradi podataka. Spajanje te dvije znanstvene grane stvorilo bi, prema njegovu mišljenju, moćan alat za inovaciju [1][10].

Godine 2001. u članku „Statistical Modeling: The Two Cultures" Leo Breiman navodi da postoje dvije struje u korištenju statističkog modeliranja u svrhu izvlačenja nekog zaključka iz podataka. Jedna pretpostavlja da su podaci stvoreni od strane danog stohastičkog modela podataka. Druga struja koristi algoritamske modele i tretira mehanizam podataka kao nepoznanicu. Smatra da je statističko društvo predano tome da skoro isključivo koristi modele podataka i da je ta predanost dovela do nebitnih teorija, upitnih zaključaka i sprječavala je statističare da rade na širokom spektru zanimljivih problema. Breiman smatra kako se algoritmičko modeliranje i u teoriji i u praksi brzo razvilo u poljima izvan statistike te da može biti korišteno na velikim skupovima podataka kao preciznija i informativnija alternativa modeliranju podataka na manjim skupovima podataka. Autor na koncu zaključuje da se statističari moraju odmaknuti od ovisnosti o modelima podataka i prihvatiti raznovrsniji skup alata ako je cilj polja statistike da koristi podatke kako bi riješilo probleme [1].

U travnju 2002. izlazi prvo izdanje časopisa „Data Science Journal" u kojem se objavljuju znanstveni radovi o upravljanju podacima i bazama podataka. Časopis izdaje „Committee on Data for Science and Tehchnology" [1].

U siječnju 2003. izlazi prvo izdanje časopisa „Journal of Data Science". Ovaj je časopis bio namijenjen svim stručnjacima koji se bave podacima, bilo to skupljanje, analiziranje, ili najvažnije - primjena podataka i u kojem su mogli iznijeti svoje poglede i razmijeniti ideje [1].

U svibnju 2005. Thomas H. Davenport, Don Cohen, i Al Jacobson objavljuju izvješće "Competing on Analytics", u kojem opisuju novi oblik natjecanja utemeljen na opsežnom korištenju analitike, podataka i donošenju odluka utemeljenih na činjenicama. Navode da umjesto natjecanja na tradicionalnim čimbenicima, tvrtke počinju koristiti statističke i kvantitativne analize i prediktivno modeliranje kao primarne elemente natjecanja. Thomas Davenport u siječnju 2006. objavljuje ovo izvješće u časopisu „Harvard Business Review" i kasnije se s Jeanne G. Harris proširuje u knjigu „Competing on analytics: The New Science of Winning" u ožujku 2007. godine [1].

U rujnu 2005. „The National Science Board" objavljuje izvješće „Long-Lived Digital Data Collections Enabling Research and Education in the 21st Century", u kojemu navode da bi NSF u partnerstvu s upraviteljima zbirki i širom zajednicom trebao osigurati mogućnost da podatkovni znanstvenici mogu razviti uspješnu karijeru i da svaka istraživačka institucija ima dovoljan broj kvalitetnih podatkovnih znanstvenika. U izvješću definiraju podatkovne znastvenike kao, informacijske i računalne stručnjake, softverske inženjere, inženjere baza podataka, programere, disciplinske stručnjake, kuratore, arhiviste, knjižničare i sve ostale koji su ključni za uspješno vođenje digitalne kolekcije podataka [1].

Godine 2007. na Sveučilištu Fudan u Šangaju osniva se Istraživački centar za podatkologiju i znanost podataka. Godine 2009. dvojica istraživača objavljuju „Introduction to Dataology and Data Science", u kojemu navode da jepodatkologija i znanost podataka nova vrsta znanosti [1].

U srpnju 2008. JISC objavljuje zadnje izvješće studije „Skills, Role & Career Structure of Data Scientists & Curators" pod nazivom „Assessment of Current Practice & Future Needs". U izvješću definiraju znanstvenike podataka kao ljude koji rade tamo gdje se vrši istraživanje ili u slučaju osoblja podatkovnog centra, usko surađuju s tvorcima podataka i mogu biti uključeni u kreativne upite i analize, omogućujući ostalima da rade s digitalnim podacima i napredcima u tehnologiji baza podataka [1].

U siječnju 2009. objavljeno je izvješće „Harnessing the Power of Digital Data for Science and Society". U izvješću se navodi kako država treba promovirati i podržavati novu znanstvenu disciplinu, i stručnjake, misleći na znanost podataka i podatkovne znastvenike. Smatraju da znanost podataka i podatkovni znanstvenici igraju ključnu ulogu razvoju znanosti, a rijetko dobivaju priznanje za to te je samim time i razvoj njihove karijere ograničen. Također u siječnju 2009. glavni ekonomist tvrtke Google u intervjuu za „McKinsey Querterly" izjavljuje da će u idućih deset godina jedna od najprivlačnijih karijera biti karijera statističara. Kao primjer navodi stručnjake u računalnim znanostima te kako nitko devedesetih godina prošlog stoljeća ne bi vjerovao da će u budućnosti ta karijera biti toliko privlačna. Smatra kako je mogućnost uzimanja neobrađenih podataka i njihovo pretvaranje u nešto korisno, što će ljudi na rukovodećim pozicijama moći razumjeti i na temelju toga raditi poslovne odluke, biti jako bitno [1].

U ožujku 2009. Kirk D. Borne i drugi astrofizičari izdaju znanstveni rad pod nazivom „The Revolution in Astronomy Education: Data Science for the Masses". U tom radu ističu važnost obuke budućih generacija u znanosti o podacima te govore da je za uspjeh u svim poljima potrebna mogućnost dobivanja informacija i zaključaka iz prikupljenih podataka, bilo to u znanosti, javnim projektima, vođenjima poduzeća ili ekonomiji. Navode da bi vještine u polju znanosti o podacima trebalo približiti svima, bili oni znanstvenici ili ne [1].

U lipnju 2009. Nathan Yau u članku „Rise of The Data Scientist" za časopis Flowingdata se poziva na izjavu Google-ovog glavnog ekonomiste Hala Valieriana da će karijera statističara u idućih 10 godina biti jedna od najprivlačnijih karijera i nadodaje da po njegovom mišljenju karijera

statističara je već danas jedna od najprivlačnijih. Također govori da Hal Valierian pod statističare nije strogo mislio na ljude koji se bave statistikom nego na bilo koga tko ima vještine da dobije korisne informacije iz velikih skupova podataka i proslijedi ih ljudima koji nemaju te vještine ali mogu upotrijebiti dobivene informacije da proizvedu nešto korisno. Spominje disertaciju Ben Frya iz 2004. godine u kojoj zagovara stvaranje nove znanstvene discipline koja će kombinirati znanja iz polja računarstva, matematike, statistike, dubinske analize podataka i grafičkog dizajna [1][11].

U veljači 2010. Keneth Cukier piše članak „Data, Data Everywhere" za „The Economist" u kojem tvrdi da se pojavilo novo zanimanje zvano „znanstvenik podataka", koje kombinira vještine programera, statističara i umjetnika kako bi izvuklo male količine vrijednih informacija skrivenih ispod velikih količina podataka [1].

U rujnu 2011. D. J. Patil u članku za „O'Reilly Radar" govori o svom sastanku s Jeffom Hammerbacherom. Taj je sastanak po njegovu mišljenju začetak znanosti o podacima kao zasebne znanstvene discipline i profesije. Također govori kako je i sam naziv znanstvenik podataka skovan na tom sastanku. Druge ideje za naziv te profesije bile su istraživački znanstvenik, poslovni analitičar i podatkovni analitičar, ali su smatrali da znanstvenik podataka najbolje opisuje nekoga tko koristi znanost i podatke da bi stvorili nešto novo [1][12].

2.2 Tehnologije u znanosti o podacima

2.2.1 SQL i baze podataka

Podaci koji se kasnije obrađuju moraju biti negdje spremljeni, a jedan od načina spremanja podataka jesu baze podataka. No, da bi se podaci obradili nije dovoljno samo podatke imati spremljene na nekom mjestu, već i imati alat koji će nam omogućiti čitanje i manipulaciju podatcima. U tu svrhu razvijen je poseban programski jezik visoke razine – SQL (eng. Structured Query Language).

2.2.1.1 Baze podataka

Baza podataka organizirani je skup informacija ili podataka, obično spremljen na memoriju nekog računala. Bazu podataka kontrolira sustav za kontrolu baze podataka (eng. Data Base Management System – DBMS). Kombinacija podataka i sustava za kontrolu baza podataka obično se naziva sistem baze podataka ili samo baza podataka. Neki od najpoznatijih DBMS-a su:

- MySQL
- PostgreSQL
- SQLite
- Oracle Database

Unutar baze podataka podaci su logički organizirani na temelju modela podataka koji koristimo. Neki od najkorištenijih modela jesu:

- Relacijski model
- Mrežni model
- Hijerarhijski model

- Objektni model

2.2.1.2 SQL

SQL (eng. Structured Query Language) jest programski jezik koji se koristi za čitanje i manipuliranje podacima unutar baze podataka. SQL je stvoren u računalnom gigantu IBM-u ranih sedamdesetih godina prošlog stoljeća.

SQL jezik obično se dijeli u tri kategorije:

- **Jezik za opis podataka (eng. Data Description Language – DDL)** - služi za projektiranje ili administriranje baze podataka. S tim jezikom definiramo podatke i veze između podataka. Neke od naredbi vezane za DDL jesu CREATE, ALTER i DROP.

- **Jezik za manipuliranje podacima (eng. Data Manipulation Language DML) -** služi programeru za izvođenje operacija manipulacije podacima kao što su brisanje, ubacivanje i mijenjanje redaka. Neke od naredbi vezane za DML-a su INSERT, DELETE i UPDATE.

- **Jezik za postavljanje upita (eng. Query Language – QL) - s**služi za pretraživanje baze i ispisivanje podataka. Naredbe su neproceduralne, što znači da s naredbama opisujemo rezultat koji želimo dobiti, a ne postupak za dobivanje rezultata.

Ovakva podjela je zastarjela i danas se obično sva tri jezika zajedno zovu SQL [18].

```
CREATE TABLE osoba(ime VARCHAR(255), prezime VARCHAR(255), datum_rodjenja DATE);
```

Slika 1: Primjer CREATE naredbe.

```
INSERT INTO osoba(ime,prezime,datum_rodjenja) VALUES('Tomislav', 'Tomić', '12-12-1996'::DATE);
```

Slika 2: Primjer INSERT naredbe.

```
SELECT * FROM osoba;
```

Slika 3: Primjer SELECT naredbe

2.2.2 Python

Python je objektno orijentirani program visoke razine apstrakcije. Stvorio ga je Guido Van Rossum i prvi put pušten u uporabu 20. veljače 1991. godine. Python je programski jezik široke uporabe i može se koristiti između ostalog za razvoj web stranica i softvera, automatizaciju, analizu podataka i prikaz podataka.

Neke od poznatijih web stranica koje koriste Python jesu: YouTube, Instagram, Google, Reddit, Spotify, Quora i Dropbox.

Popularnost Pythona u znanosti o podacima, a i u mnogim drugim poljima, proizlazi iz njegove jednostavne sintakse, jednostavnog „debugiranja", što omogućava ljudima koji nemaju pozadinu u računarstvu i programiranju da ga brzo svladaju i koriste za obradu podataka.

Neke od prednosti Pythona jesu:

- **Jednostavna sintaksa i visoka razina apstrakcije.** Python dopušta razvojnom programeru ili bilo kome tko ga koristi da se fokusira na rješavanje programskog problema, umjesto na tehničke probleme kako što su alokacija memorije i oslobađanje memorije. Također, varijable ne moraju biti definirane ranije, i vezane za specifični tip podatka, nego se mogu pozvati bez ranijeg definiranja i u tu varijablu može se spremiti više različitih tipova podataka kroz izvedbu programa.

- **Velika zajednica razvojnih programera koji koriste Python, š**to omogućava brzo učenje i pronalazak pomoći pri rješavanju nekih programskih problema, kao i pronalazak već postojećih programskih rješenja se je potom moguće ukomponirati u svoj programski kod.

- **Velik broj biblioteka.** Postoje biblioteke za mnogo različitih polja programiranja kao što su web razvoj, razvoj video igara, vizualizaciju podataka i strojno učenje, što olakšava i ubržava rad.

- **Brz razvoj programskih rješenja.** Razvoj u Pythonu puno je brži nego u mnogim drugim programskim jezicima, što štedi vrijeme, a to u kontekstu poduzeća dovodi do smanjenja troškova razvoja i ušteda koje se mogu prenijeti na krajnjeg korisnika. Na taj način poduzeća mogu steći veću konkrentnost na tržištu.

Neke od mana Pythona jesu:

- **Sporije izvođenje.** Python ne koristi kompajler, nego interpreter, što dovodi do sporijeg izvođenja programa.

- **Problemi s threadovima.** Threading nije dobar u Pythonu zbog GIL-a (Global Interpreter Lock). GIL je mutex koji dopušta da se samo jedan thread izvodi u neko vrijeme. Zbog toga se programi pisani u Pythonu koji koriste više threadova izvode sporije nego oni koji koriste samo jedan thread.

- **Problemi s razvojem za mobilne uređaje.** Python nije prirodan operacijskim sustavima kao što su Android i iOS. Ta dva operacijska sustava ne podržavaju ga kao službeni programski jezik za razvoj na tim platformama, ali uz dodatni trud postoje načini da se Python koristi i u te svrhe.

- **Problemi s prevelikim korištenjem memorije.** Python koristi veliku količinu radne memorije i iz tog razloga nije dobar za rješavanje programskih problema koji već sami po sebi zahtijevaju veliku količinu radne memorije [14].

2.2.3 Matplotlib

Matplotlib je međuplatformska biblioteka za vizualizaciju podataka za Python, i njegovu numeričku ekstenziju NumPy. Matplotlib je besplatna, open-source alternativa MATLAB-u. Matplotlib također nudi API preko kojega grafove stvorene unutar Matplotlib-a možemo uvesti u aplikacije s grafičkim sučeljem. Matplotlib je začeo John Hunter 2002. godine prvotno kao patch za IPython, koji bi dopuštao interaktivno crtanje grafova u stilu MATLAB-a kroz gnuplot iz IPython-ovog naredbenog retka. Nakon što je Fernando Perez obavijestio Johna Huntera da patch neće moći pregledati nekoliko mjeseci John Hunter se osamostalio i tako je rođena matplotlib biblioteka za Python.

Jedna od najvećih prednosti matplotliba je njegova prilagodljivost različitim operacijskim sustavima i grafičkim pozadinama. Ta prilagodljivost proizlazi iz toga što u početku matplotlib-a, njegovi tvorci nisu pokušavali biti bolji od konkurenata, nego kroz prilagodljivost omogućiti pristup matplotlib-a što većem broju ljudi [15].

Neke od prednosti matplotlib-a su:

- **Velika kontrola nad grafovima koje crtamo.** Imamo mogućnost uređivanja skoro svakog aspekta grafa, što nam omogućuje daljnje pojašnjavanje podataka koje predstavljamo.
- **Velika zajednica razvojnih programera i drugih znanstvenika koristi matplotlib** što nam omogućuje brže učenje i pronalaženje rješenja za neki problem koji se može pojaviti dok vizualiziramo podatke.
- **Velik broj dostupnih grafova koje podržava matplotlib biblioteka.** Unutar matplotlib-a imamo skoro svaki graf koji nam može biti potreban da bi vizualizirali neki podatak.

Neke od mana matplotlib-a su:

- **Kompleksan postupak stvaranja nestandardnih grafova.**
- **Kompleksno uređivanje grafova.** Iako matplotlib nudi mogućnost uređivanja mnogo detalja vezanih za graf koji smo nacrtali, ima sučelje iznimno nisko razine apstrakcije zbog čega bilo koje odstupanje od prvotnog prikaza zahtijeva veliku količinu truda.
- **Matplotlib može prikazati samo statičke grafove** [16][19]**.**

3 Anaconda

Anaconda je distribucija programskog jezika Python i R, uglavnom korištena u programiranju vezanom za znanost podataka. Anaconda također dolazi s upraviteljem paketa „conda”, koji omogućuje brzu i jednostavnu instalaciju paketa potrebnih za razvoj u Pythonu [23].

3.1 Instalacija Anaconde i matplotlib-a

Anaconda je dostupna na operacijskim sustavima MacOS, Linux i Windows [23].

3.1.1 Instalacija na Linuxu.

Instalacija anaconde na različitim Linux distribucijama uglavnom je ista, s jedinom razlikom u tome što prije instalacije moramo zadovoljiti različite ovisnosti za Qt biblioteke za svaku od distribucija.

Na Linux distribucijama baziranim na Debianu ovisnosti instaliramo unošenjem sljedeće naredbe u naredbeni redak:

apt-get install libgl1-mesa-glx libegl1-mesa libxrandr2 libxrandr2 libxss1 libxcursor1 libxcomposite1 libasound2 libxi6 libxtst6

Na RedHat distribucijama:

yum install libXcomposite libXcursor libXi libXtst libXrandr alsa-lib mesa-libEGL libXdamage mesa-libGL libXScrnSaver

Na ArchLinux distribucijama:

pacman -Sy libxau libxi libxss libxtst libxcursor libxcomposite libxdamage libxfixes libxrandr libxrender mesa-libgl alsa-lib libglvnd

Na openSUSE/SLES distribucijama:

zypper install libXcomposite1 libXi6 libXext6 libXau6 libX11-6 libXrandr2 libXrender1 libXss1 libXtst6 libXdamage1 libXcursor1 libxcb1 libasound2 libX11-xcb1 Mesa-libGL1 Mesa-libEGL1

Na Gentoo distribucijama:

emerge x11-libs/libXau x11-libs/libxcb x11-libs/libX11 x11-libs/libXext x11-libs/libXfixes x11-libs/libXrender x11-libs/libXi x11-libs/libXcomposite x11-libs/libXrandr x11-libs/libXcursor x11-libs/libXdamage x11-libs/libXScrnSaver x11-libs/libXtst media-libs/alsa-lib media-libs/mesa

Nakon instalacija ovisnosti, drugi dio – sam proces instalacije anaconde koji ćemo opisati u nastavku, ne razlikuje se na različitim distribucijama Linuxa.

Iz pretraživača (internetskog preglednika) potrebno je preuzeti skriptu za instalaciju s Anacondine stranice. Nakon što se skripta preuzela, preporučuje se provjera integriteta preuzete skripte. Unošenjem sljedeće naredbe u naredbeni redak:

sha256sum /path/to/file

Na primjer, ako se skripta preuzela u direktorij Preuzimanja (Downloads) naredba bi bila

sha256sum ~/Preuzimanja/Anaconda3-2021.05-Linux-x86_64.sh

Dobiveni rezultat trebao bi izgledati otprilike ovako:

2751ab3d678ff0277ae80f9e8a74f218cfc70fe9a9cdc7bb1c137d7e47e33d53 Anaconda3-2021.05-Linux-x86_64.sh

Integritet preuzetog paketa možemo usporediti s točnim rezultatom koji možemo pronaći na Anacondinoj stranici, nakon što izaberemo za koji operacijski sustav i distribuciju linuxa instaliramo Anacondu.

Nakon provjere integriteta podataka, možemo započeti s instalacijom skripte.

Instalaciju započinjemo unošenjem sljedeće naredbe

bash ~/Preuzimanja/Anaconda3-2021.05-Linux-x86_64.sh

Prvi korak prilikom instalacije jest prihvaćanje licencnog ugovora te navođenje gdje želimo instalirati Anacondu.

Nakon što je je skripta završila s instalacijom, možemo ponovno pokrenuti terminal ili upisati naredbu:

source ~/.bashsrc

Nakon toga možemo pokrenuti Anaconda navigator s naredbom:

anaconda-navigator

Da bismo pokrenuli Anaconda navigator putem ove naredbe, mora biti aktivirano base okruženje. Ako želimo da svaka instanca komandne linije ima automatski aktivirano base okruženje, možemo pokrenuti naredbu:

conda config --set auto_activate_base True

Da bi se deaktiviralo base okruženje (tako da svaka instanca komandne linije nije automatski u base okruženju), možemo pokrenuti naredbu:

conda config --set auto_activate_base False

U tom slučaju, prije pokretanja Anaconda navigatora kroz naredbeni redak, morat ćemo prvo pokrenuti naredbu koja će upaliti base okruženje:

conda activate

Base okruženje možemo ugasiti naredbom conda deactivate ili ponovnim pokretanjem naredbene linije.

Na kraju nam preostaje jedino instalirati matplotlib, a to možemo napraviti uz pomoć conda upravitelja podataka tako da pokrenemo naredbu

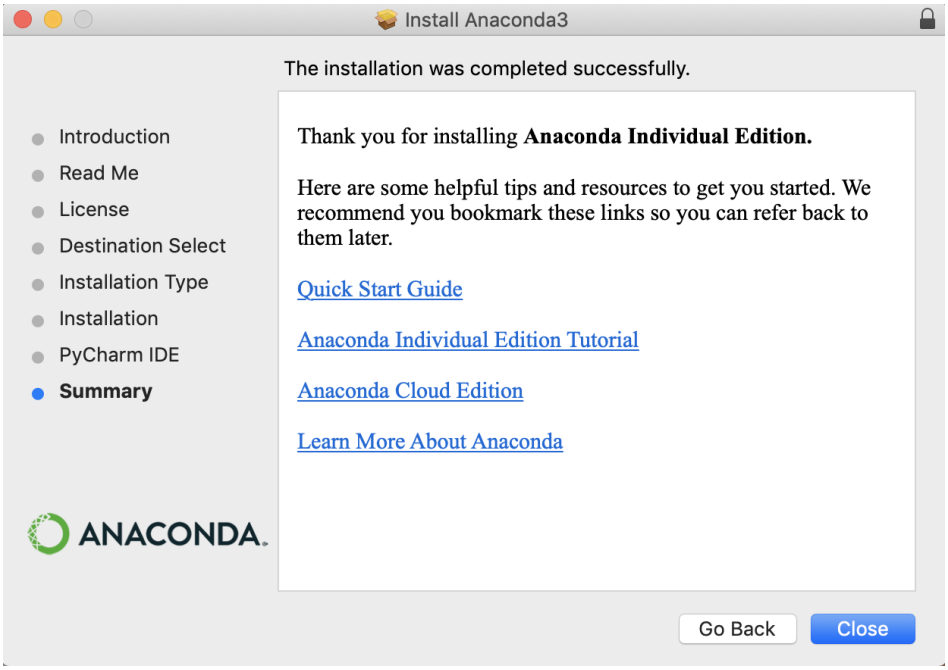
conda install -c conda-forge matplotlib [20].

3.1.2 Instalacija na MacOS-u

Instalacija na MacOS-u može se napraviti na dva načina, korištenjem grafičkog sučelja ili preko naredbenog retka.

Ako instaliramo preko grafičkog sučelja postupak je sljedeći:

- preuzimanje grafičkog instalatora s anacondine web stranice
- provjera integriteta podataka putem sha256sum
- preuzimanje instalatora i započinjanje procesa instalacije
- prihvaćanje licencnog ugovora
- potrebno je pritisnuti „Install" da bismo instalirali Anacondu u ~/opt direktorij; moguće je direktorij odabrati i ručno
- izabrati instalira li se Anaconda za sve korisnike ili samo za sebe – korisnika koji instalira
- ako je instalacija uspješno obavljena, prikazat će nam se sljedeći ekran



Slika 4: Uspješna instalacija na Mac-OS [21]

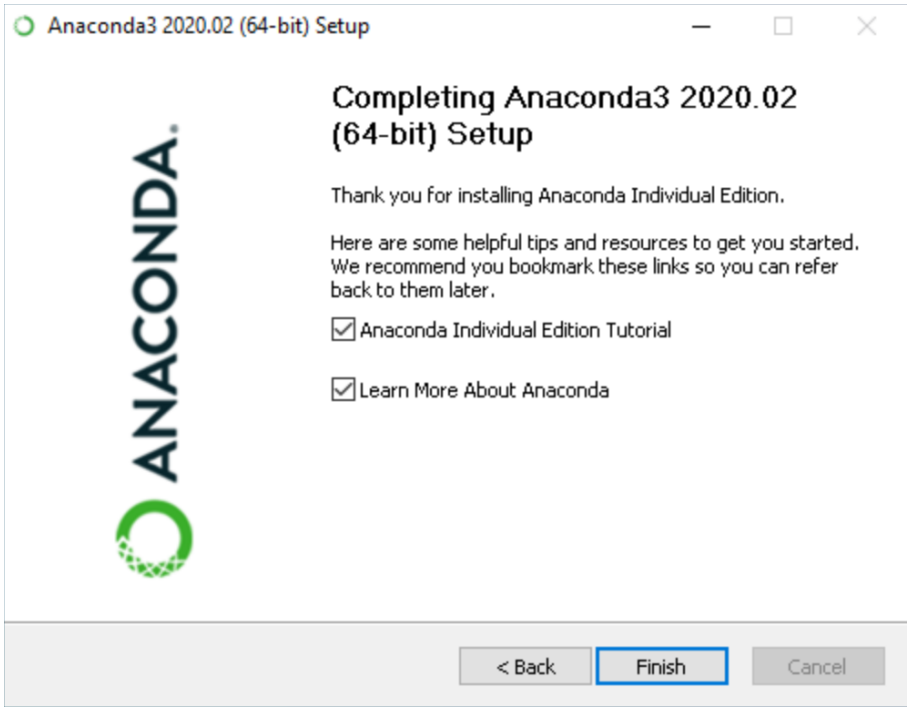
Instalacija preko naredbenog retka ima isti postupak kao i na Linuxu, jedino što na MacOS-u nije potrebno instalirati zavisnosti. Instalacija matplotlib-a također ima isti postupak kao i na Linuxu [21].

3.1.3 Instalacija na Windows

Instalacija na Windows može se napraviti jedino preko grafičkog sučelja. Postupak je sljedeći:

- preuzimanje grafičkog instalatora s anacondine web stranice

- provjera integriteta podataka sa sha256sum
- pokretanje instalacije s duplim klikom na preuzeti instalator
- prihvaćanje licencnog ugovora
- ručno izabrati lokaciju gdje ćemo instalirati Anacondu ili pritiskom na „Next" instaliramo Anacondu na zadanu lokaciju
- izabrati hoćemo li dodati Anacondu u PATH okruženje, što se ne preporučuje. Umjesto toga preporučuje se da se Anaconda Navigator i Anaconda naredbeni redak pokreće klikom na Anaconda Navigator ili Anaconda Prompt
- ako je Anaconda uspješno instalirana, trebao bi nam se pokazati sljedeći prozor



Slika 5: Uspješna instalacija Anaconde na Windows [22]

Nakon instalacije Anaconde još nam preostaje instalirati matplotlib. To možemo napraviti tako da upalimo Anaconda Prompt i pokrenemo naredbu

conda install matplotlib [22].

4 Uvoz podataka

Podatke koje želimo obraditi i prikazati prvo treba uvesti unutar programa iz nekog izvora. Postoji više načina kako se to može napraviti, a neki od njih su:

- **uvoz iz datoteke**
- **uvoz iz baze podataka**

4.1 Uvoz iz datoteke

```
import urllib.request as reader
import urlib.error as err

try:
    url = "https://opendata.ecdc.europa.eu/covid19/nationalcasedeath_eueea_daily_ei/csv"
    response = reader.urlopen(url)
except err.HTTPError as e:
    print(e.reason + " " + str(e.code))
except err.URLError as e:
    print(e.reason)
```

Slika 6: Primjer HTTP zahtjeva

Uvoz iz datoteke odnosi se na uvoz iz neke datoteke koja može biti spremljena lokalno na računalu ili se nalazi na nekom udaljenom serveru. Često, ako se podaci mijenjaju na nekoj vremenskoj bazi, koristi se uvoz s interneta da bi podaci ostali relevantni. Relevantnost podataka može se održavati i čestim ručnim preuzimanje podataka, ali ako imamo mogućnost - automatsko preuzimanje podataka nam štedi vrijeme, zato što ih računalo može preuzeti u puno bržem vremenskom roku nego što bi to ikada mogli ljudi. U slučaju da se podaci ne mijenjaju, ili ako smo u razvojnom procesu, podatke je bolje preuzeti i uvoziti ih lokalno. Na taj način postiže se brže izvođenje programa zato što se podaci ne moraju preuzeti pri svakom novom pokretanju programa, što se posebno osjeti na brzini izvođenja programa ako je skup podataka velik. Također, u nekim slučajevima postoji i ograničen broj puta koliko se neka datoteka može preuzeti, što bi nam moglo stvarati problem u razvojnom procesu, odnosno za vrijeme „debugiranja" programa.

U ovom slučaju za uvoz s interneta koristimo urlopen() metodu klase request iz biblioteke urllib, koju prvo moramo uvesti s ključnom riječi import. Kao što vidimo, kad uvozimo biblioteke ne moramo uvesti cijelu biblioteku, nego možemo uvesti samo jednu klasu iz nje. Također, s ključnom riječi „as" možemo definirati pseudonim za objekt urllib.request, tako da kad kasnije budemo pisali kod ne moramo pisati urllib.request.urlopen(), nego možemo jednostavno napisati, reader.urlopen(). Također uvozimo klasu error, u kojoj su opisane greške koje se mogu dogoditi za vrijeme HTTP zahtijeva koji metoda urlopen() radi.

Metoda urlopen() može primiti nekoliko argumenata, ali u ovom slučaju predajemo samo obavezni argument, URL. Ako urlopen() vrati kod 200 znači da je sve prošlo u redu i možemo nastaviti, a u slučaju da se dogodila neka greška dobivamo poruku s kodom greške i njezin opis.

Ako uvoz radimo s datoteke koja je spremljena lokalno, datoteku otvaramo kao i bilo koju drugu, s metodom open().

```
filename = "/home/tomislav/zavrsni/datasets/US/Provisional_COVID-19_Deaths_by_Sex_and_Age.csv"
file = open(filename,'r')
```

Slika 7: Primjer uvoza s lokalne datoteke

Funkciji open() predajemo dva argumenta: putanju do datoteke i ono što želimo raditi s datotekem, u ovom slučaju predaje se „r", što označava da želimo samo čitati iz datoteke.

Neki od najčešćih formata su:

- **JSON**

```
import json
file = json.loads(response.read())
```

Slika 8: Čitanje odgovora u JSON formatu

Za čitanje JSON datoteke preuzete HTTP zahtjevom koristimo loads() metodu iz json biblioteke.

```
import json
jf = json.load(file)
```

Slika 9: Čitanje lokalne JSON datoteke

Za čitanje JSON datoteke lokalno koristimo load() metodu iz json biblioteke.

- **CSV**

```
import csv
csvfile = [l.decode('utf-8') for l in response.readlines()]
datareader = csv.reader(csvfile)
```

Slika 10: Čitanje CSV datoteke preuzete HTTP zahtjevom s interneta

Kod uvoza iz CSV formata moramo prvo dekodirati odgovor kao UTF-8, onda otvoriti to sa reader metodom iz biblioteke CSV.

```
import csv
csvfile = csv.reader(file)
```

Slika 11: Čitanje CSV datoteke spremljene lokalno

Proces za uvoz iz lokalne datoteke je sličan, jedino što ne moramo koristiti decode() metodu

- **XLSX**

```
import pandas as pd
xlsxfile = pd.ExcelFile('/home/tomislav/Downloads/download.xlsx')
```

Slika 12: Čitanje lokalne XLSX datoteke

Za uvoz XLSX datoteke spremljene lokalno možemo koristiti ExcelFile() metodu pandas biblioteke. Ako čitamo XLSX datoteku, ne moramo prvo otvarati datoteku s funkcijom open(), nego funkciji predajemo putanju do XLSX datoteke.

```
import pandas as pd
xlsxfile = pd.ExcelFile(response.read())
```

Slika 13: Čitanje XLSX datoteke dohvaćene HTTP zahtjevom

Za uvoz datoteke preuzete HTTP zahtjevom koristimo istu ExcelFile() funkciju pandas biblioteke

4.2 Uvoz iz baze podataka

```
import psycopg2

try:
    connection = psycopg2.connect(user="tomsilav
                                  password="password"
                                  host="localhost"
                                  port="5432"
                                  database="test")

    cursor = connection.cursor()
    sql = "select * from osoba"

    cursor.execute(sql)
    res = cursor.fetchall()

    for row in res:
        print("Ime: ", row[0])
        print("Prezime: : ", row[1])
        print("Datum rođenja: ", row[2])
except(Exception, psycopg2.Error) as error:
    print("Došlo je do greške: ", error)

finally:
    if connection:
        cursor.close()
        connection.close()
        print("PostgreSQL konekcija zatvorena")
```

Slika 14: Primjer uvoza iz PostgreSQL baze podataka

Primjer uvoza iz baze podataka pokazat ćemo na primjeru uvoza iz PostgreSQL baze podataka. Za spajanje na PostgreSQL bazu podataka koristimo psycopg2 biblioteku. Prvo je potrebno stvoriti konekciju na bazu podataka, a to radimo s connect() metodom, kojoj predajemo argumente

- **User.** Odnosi se na ime korisnika unutar baze podataka.
- **Password.** Odnosi se na lozinku koju koristimo da bi pristupili bazi podataka
- **Host.** Pošto se možemo spajati i na baze podataka koje se nalaze na udaljenom serveru potrebno je predati i adresu servera. Ako se spajamo na lokalnu bazu podataka, možemo unijeti *localhost*, 127.0.0.1 ili IP adresu računala.
- **Port.** Odnosi se na port na kojem baza podataka očekuje konekciju.
- **Database.** Odnosi se na naziv baze podataka kojoj želimo pristupiti, budući da na jednom serveru može postojati nekoliko baza podataka

Nakon stvaranja konekcije potrebno je stvoriti kursor, nakon toga trebamo unijeti SQL upit i predati ga execute() metodi objekta cursor, koja vraća rezultat dohvaćen iz baze podataka. S metodom fetchall(), sve podatke dohvaćene iz baze podataka spremamo u listu kroz koju možemo proći petljom i izabrati potrebne podatke.

5 O podacima

5.1 SARS-CoV-2

Podaci koje ćemo koristiti da bismo pokazali mogućnosti matplotlib biblioteke vezani su za pandemiju virusa SARS-CoV-2. Virus je prvo zabilježen krajem 2019. i početkom 2020. godine u kineskoj provinciji Hubei u gradu Wuhanu, odakle se kroz nekoliko mjeseci proširio kroz cijeli svijet.

Do 9. rujna 2021. zabilježeno je preko 200 milijuna slučajeva infekcije virusom SARS-CoV-2 diljem svijeta i preko 4.5 milijuna smrti je pripisano bolesti Covid-19 koju virus SARS-CoV-2 izaziva. Pandemija koju je ovaj virus izazvao je doživjela medijsku do sad neviđenu medijsku popraćenost, kao i privukla pozornost mnogih grana znanosti kao što su farmacija, epidemiologija i statistika. Također je privukla veliku pozornost javnosti, svi ti elementi kombinirani su doveli do prikupljanja velike količine podataka vezanih za pandemiju [17].

5.2 ECDC-ovi Dnevni podaci o novim slučajevima zaraze u EU/EEA

European Centre for Disase Prevention and Control (ECDC) izdaje nove podatke vezane za pandemiju na tjednoj ili dnevnoj bazi. Jedan od tih skupova podataka jest broj novih slučajeva na dnevnoj bazi, kao i broj preminulih. Podaci su dostupni u XLSX, XML, JSON i CSV formatu, a osigurana je i mogućnost ručnog preuzimanja podataka, kao i API, preko kojega podatke možemo dohvatiti HTTP zahtjevom.

dateRep	day	month	year	cases	deaths	countriesAndTerritories	geoId	countryterritoryCode	popData2020	continentExp
27.08.21	27	8	2021	1535	0	Austria	AT	AUT	8901064	Europe
26.08.21	26	8	2021	1618	0	Austria	AT	AUT	8901064	Europe
25.08.21	25	8	2021	1014	6	Austria	AT	AUT	8901064	Europe
24.08.21	24	8	2021	971	1	Austria	AT	AUT	8901064	Europe
23.08.21	23	8	2021	1174	1	Austria	AT	AUT	8901064	Europe
22.08.21	22	8	2021	1309	1	Austria	AT	AUT	8901064	Europe
21.08.21	21	8	2021	1321	1	Austria	AT	AUT	8901064	Europe

Slika 15: Primjer ECDC-ovih podataka u CSV formatu

Podaci sadržani u tom skupu podataka su:

- **dateRep** – odnosi se na datum podataka
- **day** – odnosi se na redni broj dana u mjesecu
- **month** – odnosi se na redni broj mjeseca u godini
- **year** – odnosi se na godinu

- **cases** – broj novih slučajeva na taj datum
- **deaths** – broj umrlih na taj datum
- **countriesAndTerritories** – odnosi se na zemlju na koju se ostali podaci odnose
- **geold** – Alpha 2 ISO 3116 oznaka države
- **countryterritoryCode** – Alpha 3 ISO 3116 oznaka države
- **popData2020** – broj stanovnika države u 2020. godini
- **continentExp** – naziv kontinenta na kojem se država nalazi

5.3 Our World in Data skup podataka

Our World in Data ima velik skup podataka dostupan javnosti. Skup koji ćemo koristiti u ovom radu odnosi se na sveukupan broj zaraženih, umrlih, cijepljenih i ostalo od početka pandemije COVID-19.

Ovaj skup podataka sadrži previše različitih informacija da bi se prošlo i objasnilo sve, ali neki od podataka koji se nalaze u skupu su: ime države, kontinent na kojem se država nalazi, datum ažuriranja podataka za tu državu, sveukupni broj slučajeva, broj novih slučajeva na dan ažuriranja, ukupan broj preminulih, broj preminulih na dan ažuriranja, broj preminulih i zaraženih na milijun stanovnika, broj stanovnika, kao i razne podatke o cijepljenju kao što je ukupan broj cijepljenih.

iso_code	continent	location	last_updated_date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	total_cases_per_million	new_cases_per_million	new_cases_smoc
AFG	Asia	Afghanistan	2021-09-02	153306.0	46.0	69143	7127.0	4.0	4857	3848484	1155	
OWID_AFR		Africa	2021-09-02	7849133.0	27946.0	26798.0	197763.0	846.0	733571	5714751	20347	
ALB	Europe	Albania	2021-09-02	148222.0	853.0	852714	2505.0	4.0	2714	51592553	296909	296.81
DZA	Africa	Algeria	2021-09-02	196915.0	388.0	463.0	5339.0	37.0	31143	4413489	8696	
AND	Europe	Andorra	2021-09-02	15052.0	6.0	5143	130.0	0.0	0.0	194585.93	77565	
AGO	Africa	Angola	2021-09-02	48004.0	223.0	182571	1235.0	8.0	7571	1414645	6572	5.38

Slika 16: Primjer Our World in Data podataka u CSV formatu

5.4. Skup podataka američkog CDC-a o broju umrlih po spolu i dobi

Podaci u ovom skupu odnose se na broj umrlih od COVID-19, broj umrlih od upale pluća,broj umrlih od gripe i ukupan broj umrlih iz bilo kojeg razloga. Umrli su podijeljeni u dobne skupine, na spolove, a postoji i podjela na američke savezne države.

Data As Of	Start Date	End Date	Group	Year	Month	State	Sex	Age Group	COVID-19 Deaths	Total Deaths	Pneumonia Deaths	Pneumonia and COVID-19 Deaths	Influenza Deaths	Pneumonia, Influenza, or C
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	All Ages	623985	5382994	566399	308606	9251	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	Under 1 year	94	30629	328	12	22	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	0-17 years	385	54062	887	79	188	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	1-4 years	47	5650	179	9	66	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	5-14 years	131	9038	259	35	79	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	15-24 years	1131	58001	1253	447	80	
08/25/2021	01.01.20	08/21/2021	By Total			United States	All Sexes	18-29 years	2761	102121	2904	1229	148	

Slika 17: Primjer CDC-ovih podataka u CSV formatu

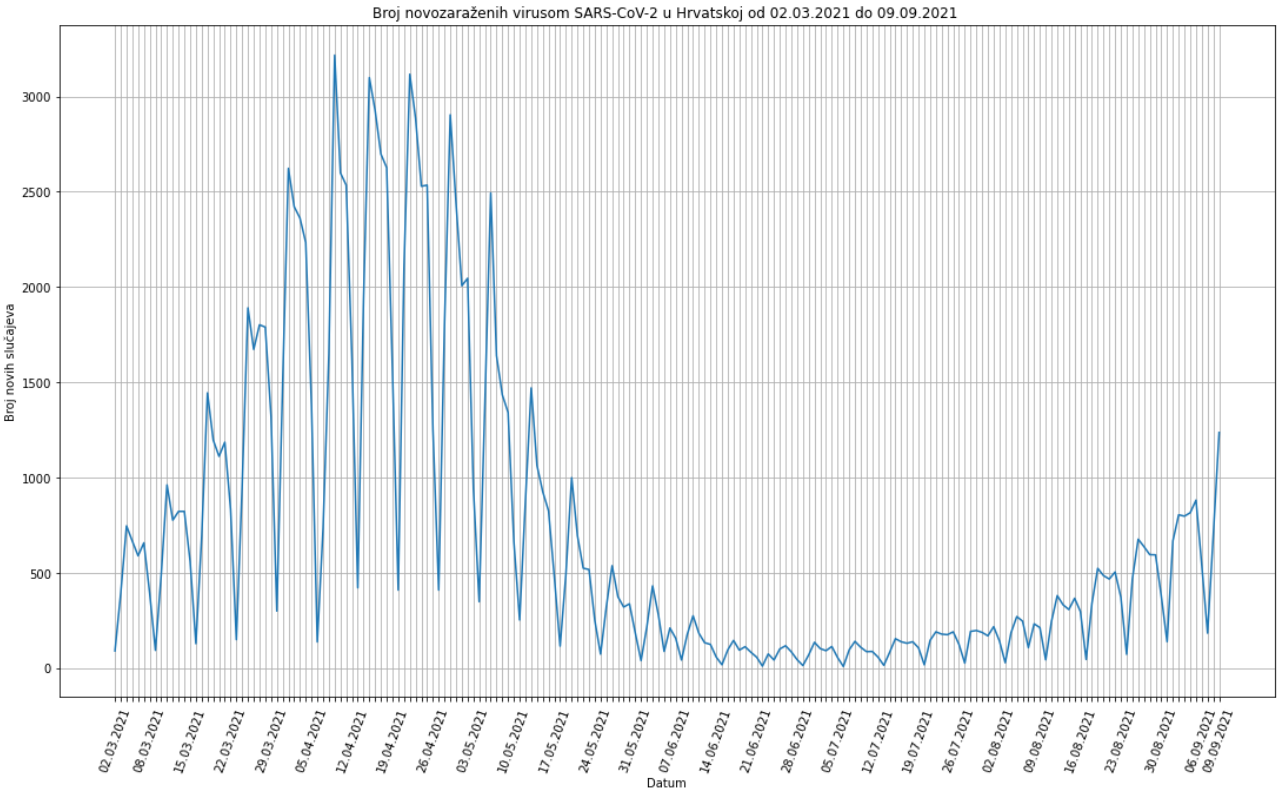
6 Prikaz podataka sa matplotlib bibliotekom

Kao što smo ranije već spomenuli, matplotlib ima mogućnost prikazivanja velikog broja grafova, te uređivanja tih grafova kako bi se bolje prezentirali neki podaci.

6.1 Linijski graf

Linijski grafovi služe kako bi se prikazala neka promjena kroz duže ili kraće vremenske periode. Također može služiti za usporedbu vremenske promjene između dvije različite stvari.

6.1.1 Broj novozaraženih u Hrvatskoj po danu



Slika 18: Promjena broja novozaraženih u hrvatskoj od 2.3.2021 do 9.9.2021

U ovom primjeru na temelju podataka ECDC-a možemo prikazati promjenu dnevnom broju novozaraženih od 2. ožujka 2021. do 9.rujna 2021.

```
import matplotlib.pyplot as plt
import csv
import datetime
import urllib.request as reader

url = "https://opendata.ecdc.europa.eu/covid19/nationalcasedeath_eueea_daily_ei/csv"
datumi = []
brojZarazenih = []
i = 0

response = reader.urlopen(url)
csvfile = [l.decode('utf-8') for l in response.readlines()]
datareader = reversed(list(csv.reader(csvfile)))
for row in datareader:
    #tražimo podatke samo za hrvatsku
    if row[6] == 'Croatia':
        #preskacemo prvi zapis zato što je tu sveukupni broj zaraženih
        #a ne broj novozaraženih za taj datum
        if i == 0:
            i+=1
            continue
        brojZarazenih.append(int(row[4]))
        datumi.append(datetime.datetime.strptime(row[0],'%d/%m/%Y').strftime('%d.%m.%Y'))

fig, ax = plt.subplots()
ax.plot(datumi,brojZarazenih)
fig.set_size_inches(18.5, 10.5)
ax.set(xlabel='Datum', ylabel='Broj novih slučajeva',
        title='Broj novozaraženih virusom SARS-CoV-2 u Hrvatskoj od '
        + datumi[0] +  ' do ' + datumi[len(datumi) - 1] + ' ')
ax.grid()

i = 0
tickCount = len(ax.xaxis.get_ticklabels())
plt.xticks(rotation=70)
for label in ax.xaxis.get_ticklabels():
    if i == 0 or datetime.datetime.strptime(datumi[i],'%d.%m.%Y').weekday() == 0 or i == tickCount - 1:
        i+=1
        label.set_visible(True)
        continue
    else:
        label.set_visible(False)
    i+=1

plt.show()
```

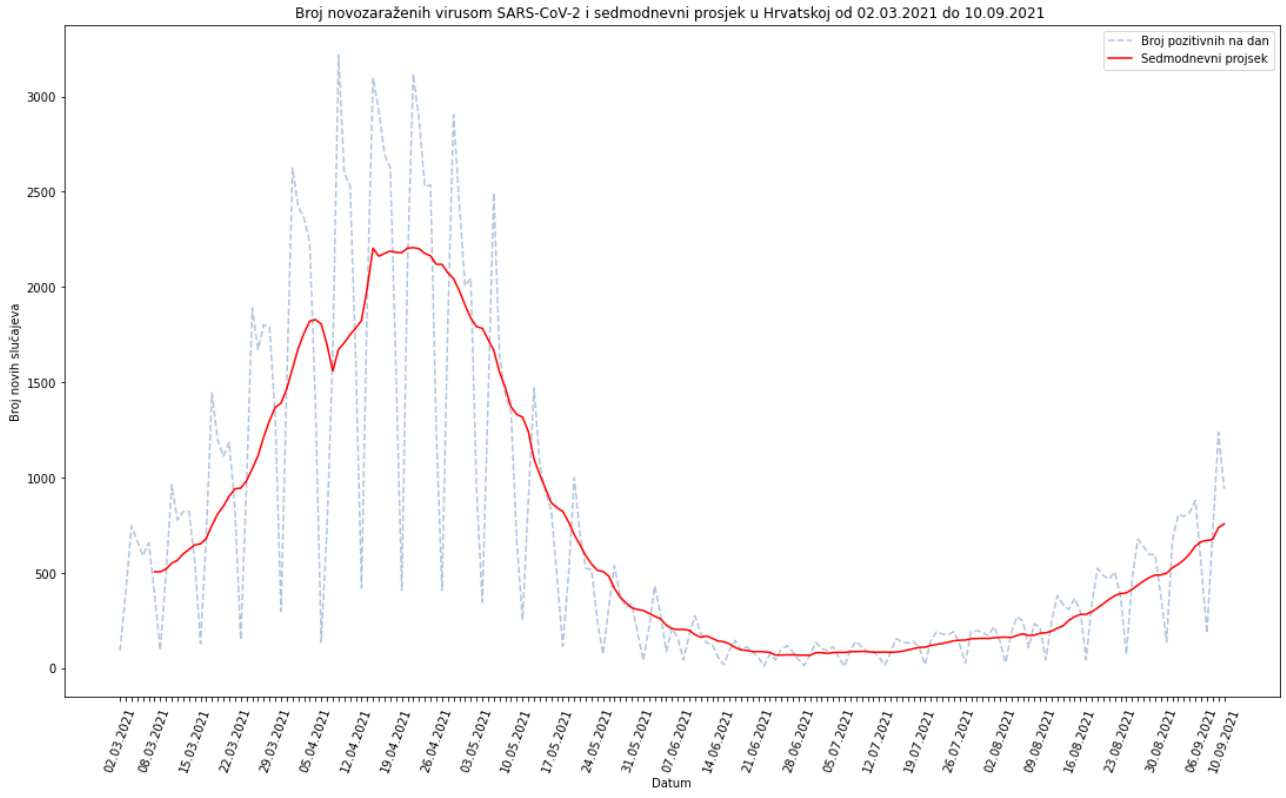
Slika 19: Kod potreban za prikaz linijskog grafa na Slici 19

Da bismo prikazali linijski graf prvo moramo uvesti matplotlib biblioteku, kao i druge datoteke koje su nam potrebne za izvođenje koda. Nakon što uvezemo CSV datoteku pomoću HTTP zahtjeva, čiji je postupak objašnjen u poglavlju **4.1**, iz datoteke moramo izabrati podatke koji su nam potrebni. Kako smo ovdje prikazivali samo promjenu novozaraženih za Hrvatsku, iz skupa podataka trebamo izvući samo podatke koji se odnose na Hrvatsku. Također podaci su zapisani obrnutim redoslijedom, to jest najraniji datum je prvi, a najkasniji zadnji, zato pročitanu CSV datoteku pretvaramo u listu sa funkcijom list() i okrećemo njen redoslijed s funkcijom reversed(). Nakon toga možemo iterirati kroz listu, u stupcu s indeksom 6 se nalazi naziv države, ako je naziv „Croatia” iz tog retka izvlačimo broj novozaraženih koji se nalazi u stupcu s indeksom 4, i dodajemo ga u listu brojZarazenih. Također, uzimamo datum koji se nalazi u stupcu s indeksom 0 i dodajemo ga u listu datumi, ali tek nakon što smo ga formatirali u format d.m.y. Formatiranje datuma obavljamo metodom strftime() iz biblioteke datetime, ali tek nakon što smo datum koji je u obliku stringa pretvorili u datetime objekt pomoću strptime() metode.

Nakon što smo smo sve podatke spremili u liste, stvaramo Figure i Axes objekte pozivom na subplots() metodu objekta pyplot. Samu liniju na grafu stvaramo metodom plot(), objekta axes, i kao argumente predajemo dvije liste, jednu s datumima, a drugu s brojevima novozaraženih.

S metodom set_size_inches() kontroliramo veličinu grafa i inčima. Metodom set() postavljamo naslove x osi, y osi i naslov grafa. U grafu datume postavljamo dinamički, zato što skup podataka preuzimamo s interneta svakim novim pokretanjem programa, pa se raspon datuma može promijeniti pri svakom novom preuzimanju podataka. Metodom grid() objekta axes, postavljamo koordinatnu mrežu, a metodom xticks() postavljamo rotaciju oznaka na x osi na 70 stupnjeva da bi ih moglo više stati nego što ih može ako su postavljene vodoravno. Čak i ako rotiramo oznake na 70 stupnjeva svejedno nema dovoljno mjesta da bi stao veliki broj datuma, zato prolazimo kroz petlju s metodom set_visible() kojoj predajemo argument True ako želimo da je odabrana oznaka vidljiva, ili false ako ne želimo da je odabrana oznaka vidljiva. U našem slučaju želimo da su prva i zadnja oznaka vidljive i svaka oznaka koja se odnosi na ponedjeljak. Je li neki datum u oznaci ponedjeljak saznajemo sa weekday() metodom datetime biblioteke koja će vratiti 0 ako je dani datum ponedjeljak. Na kraju s metodom show() prikazujemo stvoreni graf.

6.1.2 Broj novozaraženih i sedmodnevni prosjek



Slika 20: Broj novozaraženih i sedmodnevni prosjek od 2. ožujka 2021. do 10. rujna 2021.

Na linijskom grafu možemo također prikazati više linija i uređivati neka svojstva linija, kao što su boja linije i izgled linije.

Na ovom grafu smo prikazali broj novih slučajeva po danu kao svijetlu isprekidanu liniju, a sedmodnevni prosjek kao crvenu liniju.

Sedmodnevni prosjek računamo pomoću convolve() metode numpy biblioteke.

```
def moving_average(x, w):
    return np.convolve(x, np.ones(w), 'valid') / w

brojZarazenih7day = moving_average(brojZarazenih,7)
```

Slika 21: Kod za računanje sedmodnevnog prosjeka

```
for row in datareader:
    #tražimo podatke samo za hrvatsku
    if row[6] == 'Croatia':
        #preskacemo prvi zapis zato što je tu sveukupni broj zaraženih
        #a ne broj novozaraženih za taj datum
        if skip:
            skip = False
            continue
        brojZarazenih.append(int(row[4]))
        i+=1
        if i>=7:
            datumi7day.append(datetime.datetime.strptime(row[0],'%d/%m/%Y').strftime('%d.%m.%Y'))
        datumi.append(datetime.datetime.strptime(row[0],'%d/%m/%Y').strftime('%d.%m.%Y'))
```

Slika 22: Uvoz podataka kod računanja sedmodnevnog prosjeka

Uvoz podataka također se ponešto razlikuje od od prošlog grafa. Budući da su nam potrebne dvije linije, moramo imati i dva različita seta podataka za x i y osi grafa. Kad računamo sedmodnevni prosjek, prvih sedam dana ne uključuje se u prikaz, a broj podataka za x os i y os mora biti jednak, tako da stvaramo dvije različite liste za datume i u drugu listu stavljamo samo datume nakon 7 dana.

```
ax.plot(datumi,brojZarazenih, label = 'Broj pozitivnih na dan', color = 'lightsteelblue', linestyle='dashed')
ax.plot(datumi7day,brojZarazenih7day, label = 'Sedmodnevni projsek', color = 'r', linestyle='solid')
plt.legend()
```

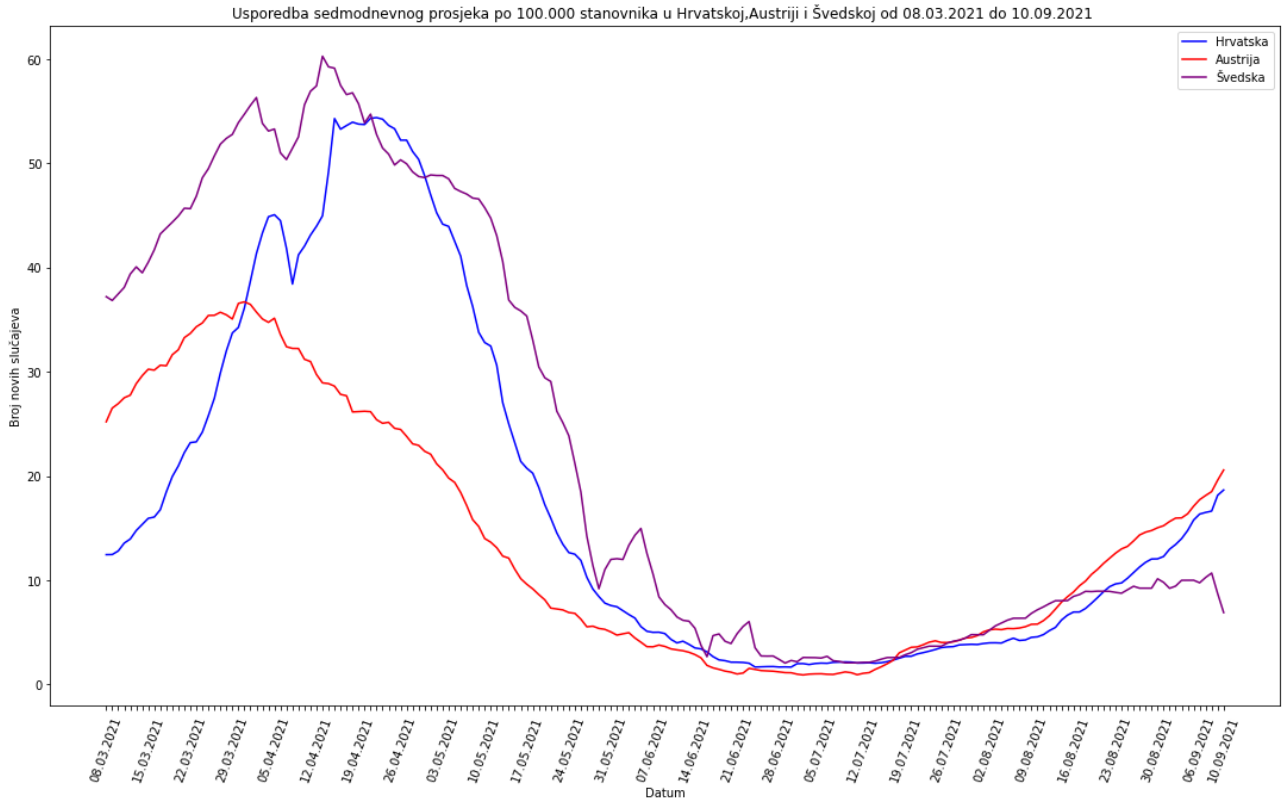
Slika 23: Stvaranje dvije različite linije na grafu

Dvije linije na grafu stvaramo tako da pozovemo plot() metodu dva puta i predamo joj podatke koje želimo prikazati. Također predajemo argumente

- **Label** – označava naziv linije
- **Color** – označava boju linije
- **Linestyle** – označava stil linije

S metodom legend() stvaramo legendu koja pokazuje na što se koja linija odnosi. Također na ovom grafu nema koordinatne linije, to je zato što nismo pozvali metodu grid().

6.1.3 Usporedba Hrvatske, Austrije i Švedske po 100.000 stanovnika



Slika 24: Usporedba sedmodnevnog prosjeka Hrvatske, Austrije i Švedske po 100.000 stanovnika

Na graf možemo dodati i više od dvije linije. Postupak je isti, jedino moramo pozvati i jednu plot() metodu kojoj predajemo treći set podataka.

Postupak uvoza podataka sličan je kao na prvom grafu, samo što u ovom slučaju uvozimo podatke i za Austriju i Švedsku, a računamo broj novozaraženih na 100.000 stanovnika tako da broj novozaraženih za taj dan podijelimo sa brojem stanovnika te države koji se nalazi u stupcu s indeksom 9, i pomnožimo sa 100.000. Kasnije tu listu predamo funkciji za računanje sedmodnevnog prosjeka.

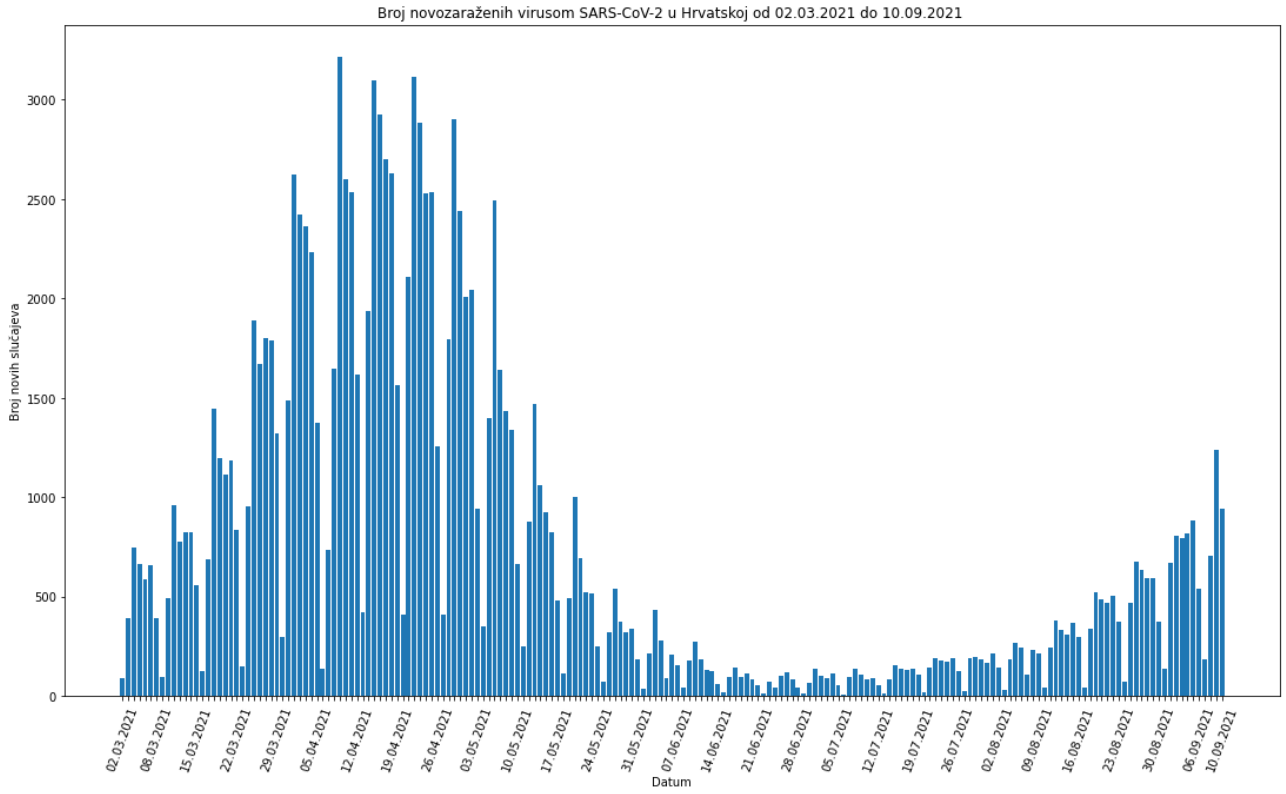
```
for row in datareader:
     if row[6] == 'Croatia':
        if popCro == 0:
            popCro = int(row[9])
            continue
        brojZarazenihHrvatska.append((int(row[4]) / popCro) * 100000)
        i+=1
        if i >= 7:
            datumi.append(datetime.datetime.strptime(row[0],'%d/%m/%Y').strftime('%d.%m.%Y'))
     elif row[6] == 'Austria':
        if popAu == 0:
            popAu = int(row[9])
            continue
        brojZarazenihAustrija.append((int(row[4]) / popAu) * 100000)

     elif row[6] == 'Sweden':
        if popSwe == 0:
            popSwe = int(row[9])
            continue
        brojZarazenihSvedska.append((int(row[4]) / popSwe) * 100000)
```

Slika 25: Uvoz podataka za Hrvatsku, Austriju i Švedsku

6.2 Stupčani graf

Stupčani graf može se koristiti kao i linijski graf za prikaz promjene kroz neki vremenski raspon, ali se može koristiti i za usporedbu razlika između nekih skupina.

6.2.1 Broj novozaraženih u Hrvatskoj po danu



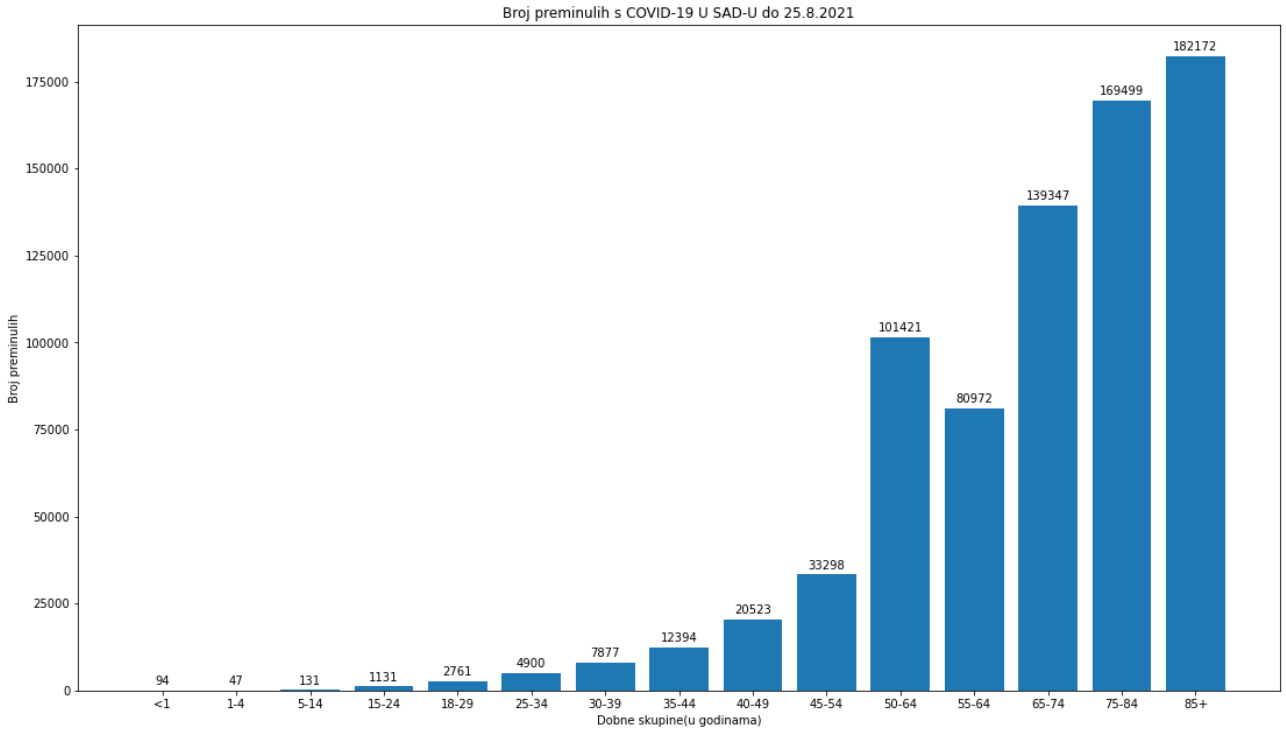
Slika 26: Prikaz broja novozaraženih u hrvatskoj po danu u obliku stupčanog grafa

Kako se stupčani graf može koristiti za prikaz promjene kroz vrijeme možemo pokazati na primjeru broja novozaraženih u Hrvatskoj. Proces prikaza broja novozaraženih u Hrvatskoj u obliku stupčanog grafa isti je kao i za prikaz sa linijskim grafom. Jedina je razlika u tome što umjesto plot() metode pozivamo bar() metodu.

```
ax.bar(datumi,brojZarazenih)
```

Slika 27: Pozivanje bar() metode

6.2.2 Umrli u SAD-u po dobnim skupinama



Slika 28: Umrli u SAD-u podijeljeni po dobnim skupinama

Kako se stupčani graf može koristiti za prikaz razlika između nekih grupa možemo pokazati na podijeli umrlih od COVID-19 po dobnoj skupini.

Podatke smo uvezli tako da smo prošli kroz preuzetu datoteku kroz petlju i koristeći indeks odabirali stupac i redak koji sadrži informacije koje su nam potrebne.

```
labels = []
brojUmrlih = []

i = 0

with open(filename,'r') as csvfile:
    datareader = csv.reader(csvfile)
    for row in datareader:
        if i == 2:
            labels.append('<1')
            brojUmrlih.append(int(row[9]))
        elif i>=4 and i<17:
            labels.append(row[8].split(' ')[0])
            brojUmrlih.append(int(row[9]))
        elif i == 17:
            labels.append('85+')
            brojUmrlih.append(int(row[9]))
        elif i>17:
            break
        i+=1
```

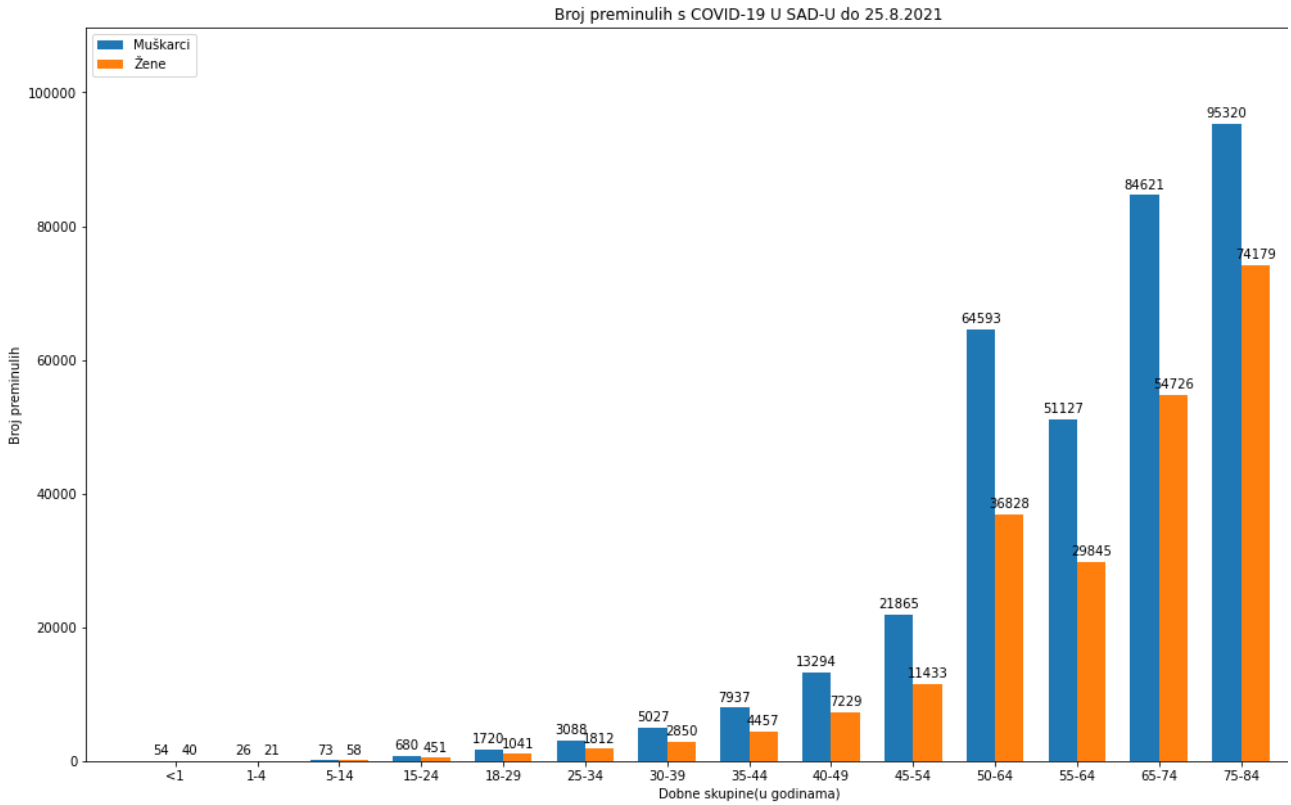
Slika 29: Proces uvoza podataka iz CDC-ovog skupa podataka.

```
rects = ax.bar(labels,brojUmrlih)
ax.bar_label(rects,padding=3)
```

Slika 30: Stvaranje oznaka iznad stupaca

U ovom grafu dodali smo oznaku na vrh svakog stupca da bismo imali bolju ideju koliko ima preminulih u svakoj skupini. To ćemo učiniti tako da kad pozivamo bar() metodu spremimo BarContainer objekt koji metoda vrati u varijablu rects i onda pozovemo bar_label() metodu kojoj predamo argumente, rects i padding.

6.2.3 Umrli u SAD-u po dobnim skupinama i spolu



Slika 31: Umrli u SAD-u podijeljeni po dobnim skupinama

Stupčane grafove također možemo grupirati, tako da za jednu oznaku na osi x imamo više različitih stupaca. To možemo prikazati tako da preminule podijelimo i na dobne skupine i po spolu.

```
x = np.arange(len(labels))
width = 0.35

ax.set_xticks(x)
ax.set_xticklabels(labels)

rects1 = ax.bar(x - width/2, brojUmrlihMuskarci, width, label='Muškarci')
rects2 = ax.bar(x + width/2, brojUmrlihZene, width, label='Žene')
```

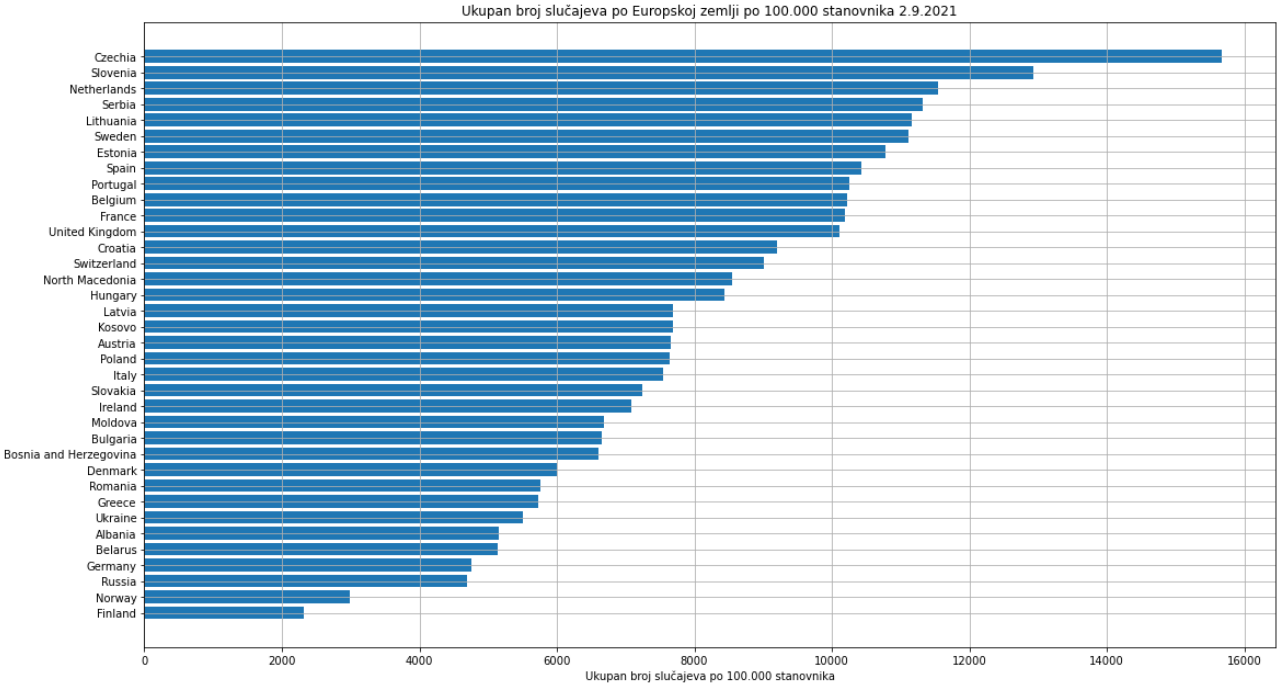
Slika 32: Kod za stvaranje grafa na slici 32

Prvo moramo pomoću arange() metode numpy biblioteke stvoriti polje cijelih brojeva dužine broja oznaka. Onda pomoću set_xticks() metode kojoj predajemo prije stvoreno polje postavljamo oznake na x os, i s metodom set_xticklabels() kojoj predajemo polje labels u kojem se nalaze nazivi dobnih skupina dajemo nazive oznakama. Poslije toga stvaramo stupce tako da dva puta pozovemo bar() metodu kojoj predajemo argumente koordinata stupca, liste s podacima o umrlima, širine stupca i naziva stupca.

6.3 Horizontalni stupčani graf

Horizontalni stupčani graf koristi se za prikaz podataka kada ima previše oznaka na osi x da bi se uredno poredale ili kad želimo bolje vizualizirati razliku između najvećeg i najmanjeg podatka.

6.3.1 Ukupan broj slučajeva po europskoj zemlji



Slika 33: Prikaz ukupnog broja zaraženih u Europi po zemlji u obliku horizontalnog stupčanog grafa

Primjer za što možemo koristiti stupčani horizontalni graf jest ukupan broj zaraženih u Europi na sto tisuća stanovnika. Kad računamo broj zaraženih na 100.000 stanovnika zemlje s malim brojem stanovnika neminovno će doći do „iskrivljavanja" statistike, stoga su zemlje koje imaju manje od milijun stanovnika isključene iz ovoga grafa.

```
filename = "/home/tomislav/zavrsni/datasets/our-world-in-data/public/data/latest/owid-covid-latest.csv"


labels = []
brojZarazenih = []
dic={}
i = 0
with open(filename,'r') as csvfile:
    datareader = csv.reader(csvfile)
    for row in datareader:
        if row[1] == 'Europe':
            try:
                if float(row[46]) > 1000000:
                    dic[row[2]]=(float(row[4])/float(row[46])) * 100000
            except ValueError:
                continue

dic = dict(sorted(dic.items(), key=lambda item: item[1]))

for key,value in dic.items():
    labels.append(key)
    brojZarazenih.append(value)

fig, ax = plt.subplots()
fig.set_size_inches(18.5, 10.5)
ax.set(xlabel='Ukupan broj slučajeva po 100.000 stanovnika'
       ,title='Ukupan broj slučajeva po Europskoj zemlji po 100.000 stanovnika 2.9.2021')

rect = ax.barh(labels,brojZarazenih)
ax.bar_label(rect, padding=3)
plt.show()
```

Slika 34: Kod za stvaranje horizontalnog stupčanog grafa

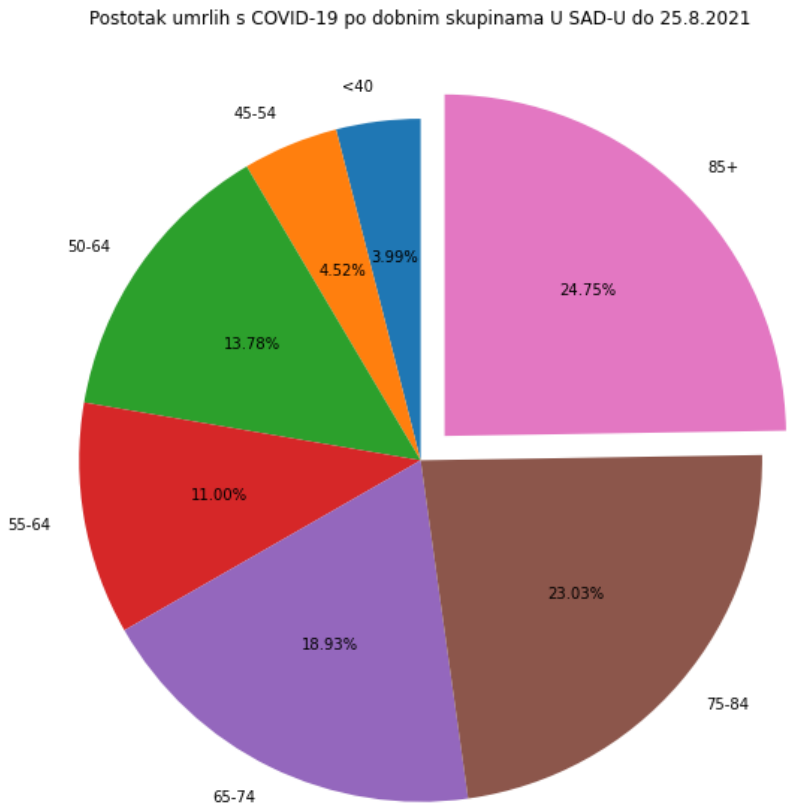
Podatke za ovaj graf uzimamo iz skupa podataka od Our World in Data. Kako CSV datoteka s podacima sadrži podatke za sve zemlje na svijetu, a nas zanima samo Europa, uzimamo u obzir samo retke u kojima stupac s indeksom 1 sadrži tekst „Europe”. Nakon toga pokušavamo vrijednost koju smo dohvatili iz stupca s indeksom 46 pretvoriti u decimalni broj i gledamo je li veći od milijun, zato što se u tom stupcu nalaze podaci o broju stanovnika. Ako je broj stanovnika manji od milijun ne uključujemo tu zemlju u ovaj graf. Ako je broj veći od milijun, uzimamo broj iz stupca s indeksom 4 i dijelimo ga s brojem stanovnika koji smo dobili iz stupca s indeksom 46 i taj broj množimo s 100.000. Dobiveni rezultat spremamo u dictionary kojem je ključ naziv države koji smo dobili. Kako su neki stupci s indeksom 4 i 46 prazni, to dovodi do pucanja programa kada pokušamo prazan string pretvoriti u decimalni broj, zato je cijeli proces provjere broje stanovnika i računanja broja zaraženih na 100.000 stanovnika stavljen u try, tako da ako dođe do greške petlja nastavlja dalje umjesto da se prekine cijelo izvođenje programa.

Nakon što smo sve podatke spremili u dictionary, sortiramo ga po veličini vrijednosti. Nakon toga prolazimo kroz petlju i ključeve spremamo u listu labels, a vrijednost u listu brojZarazenih. Same horizontalne stupce stvaramo metodom barh() kojoj predajemo listu labels i listu brojZarazenih.

6.4 Kružni Graf

Kružni graf koristi se za prikaz udjela nečega u ukupnoj populaciji. Za razliku od linijskog ili stupčanog grafa ne može se koristiti za prikaz promjene nečega kroz neko vremensko razdoblje.

6.4.1 Umrli od Covid-19 u SAD-u po dobnim skupinama



Slika 35: Umrli od Covid-19 po dobnim skupinama, prikazano kružnim grafom

Postupak uvoza podataka je sličan kao i na grafu u poglavlju **6.2.2**, ali kako broj umrlih koji su mlađi od 40. godina čini jako mali postotak umrlih, da bi graf bio jasniji svi mlađi od 40 godina su grupirani u isto dobnu skupinu.

Petljom prolazimo kroz skup podataka koji je u CSV formatu, i retke s indeksom koji nam je potreban stavljamo u određene skupine, ako je dobna skupina u tom retku zbrajamo je u varijablu manjeOd40, ako je bilo što drugo direkt tu vrijednost ubacujemo u listu brojUmrlih. Također ubacujemo u listu labels nazive dobnih skupina.

Sam kružni graf stvaramo funkcijom pie() koja u ovom slučaju prima sljedeće argument:

- **x** – Listu realnih brojeva s kojom se određuje veličina pojedinih dijelova kružnog prikaza.
- **explode** – Lista realnih brojeva s kojim jedan dio grafa možemo odvojiti od ostatka grafa, u ovom slučaju smo odvojili najveću dobnu skupinu, tako da smo koristili zeroes(), array i argmax() funkcije.
- **labels** – Prima polje s nazivima pojedinih dijelova kružnog grafa
- **autopct** – Predajemo string s kojim formatiramo postotak
- **shadow** – Označavamo je li želimo da graf ima sijenu
- **startangle** – Kut pod kojim prvi dio kružnog grafa kreće

```
labels = []
brojUmrlih = []
manjeOd40 = 0;

i = 0

with open(filename,'r') as csvfile:
    datareader = csv.reader(csvfile)
    for row in datareader:
        if i == 2:
            manjeOd40+=int(row[9])
        elif i>=4 and i<11:
            manjeOd40+=int(row[9])
        elif i>11 and i < 17:
            labels.append(row[8].split(' ')[0])
            brojUmrlih.append(int(row[9]))
        elif i == 17:
            labels.append('85+')
            brojUmrlih.append(int(row[9]))
        elif i>17:
            break
        i+=1


brojUmrlih.insert(0,manjeOd40)
labels.insert(0,'<40')

najveci = np.array(brojUmrlih)
explode = np.zeros(len(najveci))
explode[najveci.argmax()] = 0.1

fig, ax = plt.subplots()



fig.set_size_inches(18.5, 10.5)
ax.set(title='Postotak umrlih s COVID-19 po dobnim skupinama U SAD-U do 25.8.2021')

ax.pie(x=brojUmrlih, explode=explode, labels=labels, autopct='%1.2f%%',
        shadow=False, startangle=90)

plt.show()
```

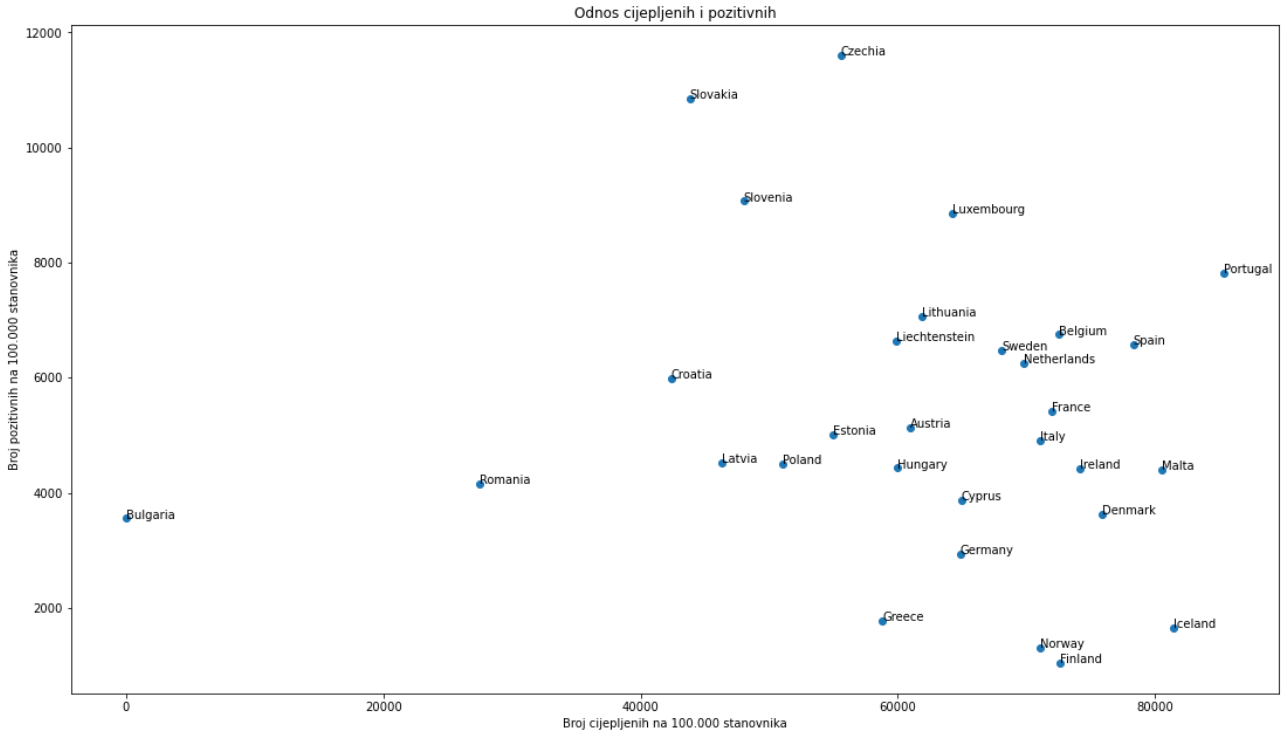
Slika 36: Kod za prikaz kružnog grafa

6.5 Raspršeni graf

Raspršeni graf se koristi da se prikaže odnos između dvije numeričke varijable. Koristeći raspršeni graf možemo vizualizirati neke podatke koji ima dvije numeričke vrijednosti, i iz njihovih pozicija na grafu iščitati korelaciju na temelju njihovog grupiranja.

6.5.1 Odnos cijepljenih i pozitivnih u EU/EEA na 100.000 stanovnika

Jedna od stvari koji možemo prikazati sa raspršenim grafom je odnos cijepljenih i pozitivnih u EU/EEA, za stvaranje ovog grafa je bilo potrebno koristiti dva različita skupa podataka, jedan od ECDC-a, a drugi od Our World in Data, iz razloga što u skupu podataka od Our World in Data imamo ukupan broj pozitivnih od početka pandemije, prikazivati taj broj ne bi bilo reprezentativno zato što je veliki broj tih slučajeva iz doba prije nego što su cjepiva bila dostupna, dok ECDC-ovi podaci se odnose na samo od početka trećeg mjeseca, kad su cjepiva već bila dostupna.



Slika 37: Raspršeni graf za prikaz odnosa cijepljenih i pozitivnih na 100.000 stanovnika

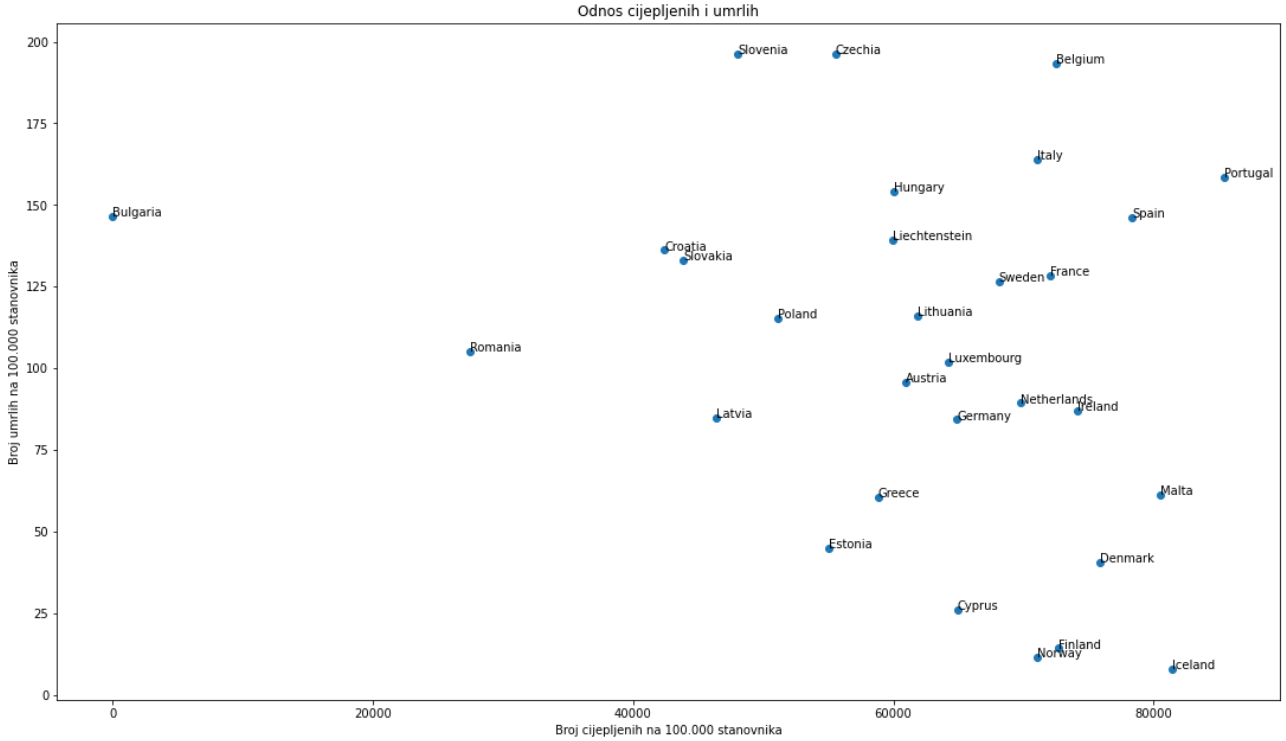
Za prikaz ovog grafa prvo smo morali stvoriti klasu country koja je primala tri podatka, naziv države, broj cijepljenih, i broj pozitivnih. Prolaskom kroz ECDC-ove podatke uzimali smo samo one retke koji su sadržavali ukupan broj pozitivnih, iz tih redaka smo ukupan broj pozitivnih i nazive država. S tim podacima smo stvarali instance objekta country, i te podatke spremali u taj objekt, a objekt u listu countries. Nakon toga smo prošli kroz skup podataka od Our World in Data i za svaku Europsku zemlju prolazili kroz listu countries i ako je ime države iz retka u skupu podataka bilo isto nazivu države spremljenom u objekt, broj cijepljenih smo spremali u taj isti objekt.

Nakon uvoza svih podataka petljom smo prošli kroz listu country, i u liste labels, vaxed i positive smo spremali vrijednosti koje su spremljene u objektu. Sam raspršeni graf se stvara s metodom scatter() u koju smo predali liste s brojem cijepljenih i pozitivnih. Da bi dali ime pojedinim točkama na grafu petljom smo prošli kroz liste vaxed i positive, i uz pomoć annotate() metode postavili oznake s nazivom države na iste koordinate gdje se na grafu nalaze točke koje predstavljaju pojedinu državu.

```
class country:
    def __init__(self,name,vaxed,positive):
        self.name = name
        self.vaxed = vaxed
        self.positive = positive

url = "https://opendata.ecdc.europa.eu/covid19/nationalcasedeath_eueea_daily_ei/csv"
filename = "/home/tomislav/zavrsni/datasets/our-world-in-data/public/data/latest/owid-covid-latest.csv"
oldname = ""

response = reader.urlopen(url)
csvfile = [l.decode('utf-8') for l in response.readlines()]
datareader = reversed(list(csv.reader(csvfile)))
countries = []
vaxed = []
positive = []
labels = []
for row in datareader:
    if oldname != row[6] and row[4]!='cases' :
        countries.append(country(row[6],0,(int(row[4]) / int(row[9])) * 100000))
        oldname = row[6]

with open(filename,'r') as csvfile:
    datareader = csv.reader(csvfile)
    for row in datareader:
        if row[1] == 'Europe':
            try:
                for country in countries:
                    if country.name == row[2]:
                        country.vaxed = (float(row[35]) / float(row[46])) * 100000
            except ValueError:
                continue

for country in countries:
    labels.append(country.name)
    vaxed.append(country.vaxed)
    positive.append(country.positive)

fig, ax = plt.subplots()

fig.set_size_inches(18.5, 10.5)
ax.set(xlabel = 'Broj cijepljenih na 100.000 stanovnika' ,
       ylabel ='Broj pozitivnih na 100.000 stanovnika',
       title='Odnos cijepljenih i pozitivnih')

ax.scatter(vaxed,positive)

i=0
for vax,pos in zip(vaxed,positive):
    ax.annotate(labels[i],xy=(vax,pos))
    i+=1

plt.show()
```

Slika 38: Kod za stvaranje raspršenog grafa

6.5.2 Odnos umrlih i cijepljenih u EU/EEA na 100.000 stanovnika



Slika 39: Raspršeni graf za prikaz odnosa cijepljenih i umrlih na 100.000 stanovnika

Postupak za stvaranje raspršenog grafa za prikaz odnosa cijepljenih i umrlih na 100.000 stanovnika se razlikuje od postupka za stvaranje grafa u poglavlju **6.5.2**, jedino po tome što u ovom slučaju iz ECDC-ovog skupa podataka uzimamo podatke o umrlima.

7 Zaključak

U ovom završnom radu objasnili smo pojam znanosti o podacima, i njen razvoj, te prikazali mogućnosti Python biblioteke matplotlib. Cilj ovog završnog rada bio je pokazati kako je kroz par zadnjih desetljeća došlo do sve veće potrebe za stvaranjem potpuno nove grane znanosti koja će ujediniti vještine iz nekoliko različitih znanstvenih disciplina da bi se iz velike količine podataka koja se skuplja na dnevnoj bazi izvukla neka informacija koju možemo upotrijebiti. Matplotlib je alat koji je proizašao iz te potrebe, on nam omogućava da uzmemo velike količine podataka i sa malom količinom koda te podatke prikažemo ljudima na njima razumljiv način. Kod Python-a i matplotlib-a je lako razumljiv i jednostavan za reproducirati, to omogućava ljudima koji nemaju iskustva u programiranju i računarstvu da ga koriste i upotrijebe znanja iz drugih znanstvenih disciplina te ih kombiniraju sa mogućnostima modernih računala i moderne tehnologije i da bi iz brda teško razumljivih podataka izvukli neku korisnu informaciju.

LITERATURA

[2] Nepoznato,Wikipedia, [Online]. "Data science", https://en.wikipedia.org/wiki/Data_science
(Pristupljeno 22. kolovoza 2021.)

[3] Nepoznato,Oracle, [Online]. "What is Data Science?", https://www.oracle.com/data-science/what-is-data-science/
(Pristupljeno 22. kolovoza 2021.)

[4] Anna Heinrich,Rasmussen University, [Online]. "Computer Science vs. Data Science: Decoding Your Ideal Career Path", https://www.rasmussen.edu/degrees/technology/blog/computer-science-vs-data-science/
(Pristupljeno 22. kolovoza 2021.)

[5] Giuliano Liguori,KDnuggets, [Online]. "Data Science History and Overview", https://www.kdnuggets.com/2020/11/data-science-history-overview.html
(Pristupljeno 22. kolovoza 2021.)

[6] UW DATA SCIENCE TEAM,University of Wisconsin, [Online]. "A Modern History of Data Science", , https://datasciencedegree.wisconsin.edu/blog/history-of-data-science/
(Pristupljeno 22. kolovoza 2021.)

[1] Gil Press,Forbes, [Online]. "A Very Short History Of Data Science", 2012, https://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science/
(Pristupljeno 22. kolovoza 2021.)

[8] ,IASC ISI, [Online]. https://iasc-isi.org/mission/
(Pristupljeno 22. kolovoza 2021.)

[7] Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P., From Data Mining to Knowledge Discovery in Databases, 1996

[9] Jacob Zahawi,Knowledge@Wharton, [Online]. "Mining Data for Nuggets of Knowledge", 1999, https://knowledge.wharton.upenn.edu/article/mining-data-for-nuggets-of-knowledge/
(Pristupljeno 22. kolovoza 2021.)

[10]: William S. Cleveland, Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics, 2001

[11] Nathan Yau,Flowing Data, [Online]. "Rise of The Data Scientist", 2009, https://flowingdata.com/2009/06/04/rise-of-the-data-scientist/
(Pristupljeno 22. kolovoza 2021.)

[12] DJ Patil,O'Reilly Radar, [Online]. "Building data science teams", 2011, http://radar.oreilly.com/2011/09/building-data-science-teams.html
(Pristupljeno 22. kolovoza 2021.)

[18] R. Manger, Baze podataka, 2003, Zagreb

[14] Krzysztof Basel,netguru, [Online]. "Python Pros and Cons", , https://www.netguru.com/blog/python-pros-and-cons
(Pristupljeno 3. rujna 2021.)

[15] Nepoznato,Active State, [Online]. "What Is Matplotlib In Python?", https://www.activestate.com/resources/quick-reads/what-is-matplotlib-in-python-how-to-use-it-for-plotting/ (Pristupljeno 5. Rujna)

[16] Natasha Sharma,Neptune Blog, [Online]. "Top Tools for Data Exploration and Visualization With Their Pros and Cons", https://neptune.ai/blog/data-exploration-and-visualization-best-tools (Pristupljeno 5. Rujna)

[19] Khuyen Tran,Towards Data Science, [Online]. "Top 6 Python Libraries for Visualization: Which one to Use?", https://towardsdatascience.com/top-6-python-libraries-for-visualization-which-one-to-use-fe43381cd658 (Pristupljeno 5. Rujna)

[23] Nepoznato,Wikipedia, [Online]. "Anaconda (Python distribution)", (Pristupljeno 28. kolovoza 2021)

[20] Nepoznato,Anaconda Inc., [Online]. "Installing on Linux", https://docs.anaconda.com/anaconda/install/linux/ (Pristupljeno 28. kolovoza 2021)

[21] Nepoznato,Anaconda Inc, [Online]. "Installing on macOS", https://docs.anaconda.com/anaconda/install/mac-os/ (Pristupljeno 28. kolovoza 2021)

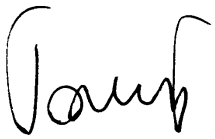
[22] Nepoznato,Anaconda Inc., [Online]. "Installing on Windows", , https://docs.anaconda.com/anaconda/install/windows/ (Pristupljeno 28. kolovoza 2021)

[17] Nepoznato,WHO, [Online]. "Coronavirus disease (COVID-19) pandemic", https://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/novel-coronavirus-2019-ncov (Pristupljeno 7.rujna 2021)

POPIS SLIKA

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam završni rad izradio samostalno,isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima podataka i uz stručno vodstvo mentora dr. sc Mario Miličevića, kojem se još jednom srdačno zahvaljujem.



Tomislav Tomić