

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA

IZDELAVA SPLETNE APLIKACIJE SHINY ZA
ANALIZO PODATKOV PANDEMIJE COVID-19

LARA KOMEL

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN
INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga

Izdelava spletne aplikacije Shiny za analizo podatkov Covid-19

(Development of a Shiny web application for Covid-19 data analysis)

Ime in priimek: Lara Komel

Študijski program: Računalništvo in informatika

Mentor: doc. dr. Uroš Godnov

Koper, julij 2023

Ključna dokumentacijska informacija

Ime in PRIIMEK: Lara KOMEL

Naslov zaključne naloge: IZDELAVA SPLETNE APLIKACIJE SHINY ZA ANALIZO
PODATKOV PANDEMIJE COVID-19

Kraj: Koper

Leto: 2023

Število strani: 43 Število slik: 23

Število referenc: 22

Mentor: doc. dr. Uroš Godnov

Ključne besede: Shiny, R, spletna aplikacija, podatki, Covid-19, analiza podatkov

Izvleček: Zaključna naloga predstavlja raziskavo in izdelavo spletne aplikacije za analizo podatkov pandemije Covid-19, izdelane v programskem jeziku R in RShiny. Aplikacija omogoča vizualni pregled potrjenih primerov, smrti in cepljenj skozi čas v različnih državah, primerjavo števila le teh med več državami in prikaz na zemljevidu.

Key document information

Name and SURNAME: Lara KOMEL

Title of the final project paper: Development of a Shiny web application for Covid-19 data analysis

Place: Koper

Year: 2023

Number of pages: 43 Number of figures: 23

Number of references: 22

Mentor: doc. dr. Uroš Godnov

Keywords: Shiny, R, web application, data, Covid-19, data analysis

Abstract: This thesis focuses on research and development of a web application for analyzing data related to the Covid-19 pandemic. The application is built using the R programming language and RShiny framework. The application enables a visual overview of confirmed cases, deaths and vaccinations over time in all countries, and allows users to compare numbers between different countries, as well as view the data on a world map.

ZAHVALA

Zahvaljujem se družini in še posebej svoji mami za podporo na moji študijski poti.

Zahvaljujem se tudi svojemu mentorju doc. dr. Urošu Godnovu za podporo in nasvete od izdelavi nalogi in pomoč pri izbiri teme.

KAZALO VSEBINE

1	UVOD.....	1
2	COVID-19	2
3	GRAFIČNI PRIKAZ PODATKOV	4
	3.1 R in Rstudio.....	7
	3.2 RShiny	9
4	NABOR PODATKOV	9
	4.1 Pregled podatkov	10
	4.2 Priprava podatkov.....	10
5	SPLETNA APLIKACIJA SHINY	11
	5.1 Predpriprava	11
	5.2 Izvedba.....	11
	5.2.1 Domača stran	11
	5.2.2. Pregled po državah	18
	5.2.3. Primerjava držav.....	24
	5.2.4. Pregled po celinah	26
	5.2.5. Primerjava celin.....	28
	5.2.6. Prikaz na zemljevidu	30
	5.2.7. Podatki	33
	5.2.8. O aplikaciji	34
6	ZAKLJUČEK	34
7	LITERATURA IN VIRI.....	35

KAZALO SLIK IN GRAFIKONOV

Slika 1: Tabela	4
Slika 2: Stolpični diagram	5
Slika 3: Tortni diagram.....	5
Slika 4: Točkovni diagram	6
Slika 5: Linijski diagram	6
Slika 6: Krivuljni diagram	7
Slika 7: Bločni diagram	7
Slika 8: Delo s podatki	8
Slika 9: Prikaz glavnih podatkov	12
Slika 10: Lestvica držav	13
Slika 11: Domača stran graf	14
Slika 12: Domača stran histogram.....	16
Slika 13: Domača stran.....	18
Slika 14: Graf izbrane države	20
Slika 15: Glavni podatki države	23
Slika 16: Stran podatkov držav.....	24
Slika 17: Stran primerjava držav	26
Slika 18: Graf izbrane celine	27
Slika 19: Stran podatkov celin.....	28
Slika 20: Graf primerjave celin	29
Slika 21: Zemljevid	32
Slika 22: Prikaz podatkov.....	33
Slika 23: Stran o aplikaciji.....	34

1 UVOD

Pandemija Covid-19 je zadnji dve leti in pol zasedala članke po spletnih portalih, naslovnice časopisov, najbolj pogoste teme pogovora uporabnikov socialnih omrežij in velik del vsakodnevnih televizijskih poročil. Zdaj, ko se življenje za večino ljudi po svetu vrača vsaj nekako v normalo, se lahko ozremo na zadnji dve leti naših življenj ter podrobno analiziramo dogodek, ki nam je na novo opredelil našo normalnost.

Kako se je pandemija začela, kako se je razširila po celem svetu, kakšne so bile njene realne številke, kateri deli sveta so bili najbolj prizadeti in s katerimi lastnostmi lahko to povezujemo (razvitost, bogatost držav, politika ipd.) so le nekatera vprašanja, na katera lahko poskusimo poiskati odgovore z analizo dnevnih podatkov, ki so se zadnji dve leti nabirali po celem svetu

Namen te diplomske naloge je bil ustvariti spletno aplikacijo ter z njo raziskati globalno epidemijo.

V aplikaciji si lahko s klikanjem na različne zavihke in vnašanjem poljubnih vrednosti (npr. datumov) ogledamo vizualni prikaz Covid-19 pandemije skozi zadnjih 30 mesecev. V aplikaciji so grafično prikazane vse pomembne številke kot so porast okužb skozi čas, smrtnost, cepljenje po vseh državah sveta za daljše časovno obdobje ali vsak dan posebej, primerjava poljubnih dveh držav glede na obvladovanje pandemije in podobno.

Teoretični del naloge temelji na kratki raziskavi strokovnih člankov o Covid-19 pandemiji, bolj obsežno pa na načinih prikazovanja velikih količin podatkov, dokumentaciji in literaturi programskega okolja za statistično obdelavo in grafični prikaz podatkov R in R studio, ter dokumentaciji in literaturi R odprtokodnega paketa Shiny za gradnjo spletnih aplikacij z uporabo R programskega jezika.

Empirični del je sestavljen iz tehnične predstavitve postavitve celotne Shiny spletne aplikacije, pregleda nabora podatkov o Covid-19 pandemiji in podobnostne analize podatkov z Shiny aplikacijo.

2 COVID-19

Najprej pogledjmo kaj nekatere besede, ki jih zadnjih nekaj let slišimo vsakodnevno sploh pomenijo. Dve izmed teh sta zagotovo besedi koronavirus ter COVID-19.

Koronavirus je virus, ki povzroči okužbo v nosu, sinusih in zgornjem delu grla. Po izbruhu na Kitajskem v decembru 2019, je v začetku leta 2020 Svetovna zdravstvena organizacija-WHO identificirala SARS-CoV-2 kot nov tip koronavirusa, ki se je hitro razširil po celem svetu. SARS-CoV-2 je najverjetneje živalskega izvora in je kasneje z mutacijami povzročil bolezen tudi pri ljudeh. V preteklosti se je več nalezljivih bolezni, ki so postale nevarne za ljudi, najprej pojavilo pri pticah, prašičih, netoperjih in drugih živalih, točen razvoj novega koronavirusa se pa še raziskuje. [5]

COVID-19 je bolezen, ki jo povzroča SARS-CoV-2, povzroči pa okužbo dihalnih poti. Bolezen ima različne poteke, pri okoli 80% okuženih je le ta blag, to je z vročino, kašljem, glavobolom, bolečinami v mišicah in sklepah ter izgubo vonja in okusa. Preostalih 20% okuženih ima težji potek, približno 5% pa kritičnega, tudi z potrebami po zdravljenju na intenzivni enoti v bolnišnici. Bolezen se lahko prelevi v pljučnico ali druge zaplete, ki so lahko tudi smrtni za nekatere bolnike. Med bolj ogrožene osebe spadajo starejši od 60 let in pa tisti s pridruženimi kroničnimi boleznimi. [6]

SARS-CoV-2 se med ljudmi prenaša s kapljicami, ki jih ljudje izločajo ob kašljanju, kihanju ali pa preprosto govorjenju in dihanju. Večje kapljice padejo na površine, od kod lahko kasneje virus sami prenesemo do ust ali nosa, manjši delci pa lahko ostanejo v zraku še nekaj časa.[6]

Simptomi okužbe se lahko pojavijo od dva pa tudi do 14 dni po tem, ko smo bili virusu izpostavljeni. Oseba okužena z virusom lahko okuži druge dva dni pred prvimi znaki bolezni, in ostaja kužna naslednjih 10 do 20 dni. [7]

Okužbo s COVID-19 lahko potrdimo le z opravljenim testom, saj se veliko simptomov tega virusa ujema z prehladom in drugimi boleznimi, veliko okuženim ljudi pa znakov obolenja sploh ne kaže. Po okužbi pri večini obolelih zdravljenje v bolnišnici ni potrebno, saj lahko sami ozdravimo z počitkom in zdravili za blaženje vročine. Hujši primeri okužbe pa lahko zahtevajo hospitalizacijo, nadomestni kisik, zdravila in ventilator. [7]

Še dve zelo pogosto uporabljani besedi, ko govorimo o koronavirusu sta pandemija in epidemija. Svetovna zdravstvena organizacija je izbruh SARS-CoV-2 razglasila za pandemijo, kar pomeni, da se je nevarnost za okužbo z novim koronavirusom razširila po

skoraj celem svetu. Pri epidemiji pa je hudo nalezljiva bolezen razširjena le v neki državi ali regiji, zato jo tudi razglasijo vodilni vsake države posebej. [11]

Kriteriji, ki epidemijo spremenijo v pandemijo in jih za to uporablja Svetovna zdravstvena organizacija morajo biti izpolnjeni istočasno in so naslednji: bolezen je nova pri določenem obsegu prebivalcev, ki imajo zaradi nje večje zdravstvene težave in ki se hitro in brez ovir širi med prebivalstvom. Če se bolezen npr. pojavlja širom po svetu, ni pandemija, v kolikor ni podan pogoj hitrega širjenja med prebivalci. [11]

Pred zadnjo pandemijo koronavirusa leta 2020, se je po svetu razvilo še kar nekaj drugih pandemij. Najstarejša je pandemija kuge, ki je v 14. stoletju ubila od 75 pa do 200 milijonov ljudi po svetu. V zadnjih stoletjih pa se soočamo predvsem z pandemijami gripe, te so bile od 20. stoletja kar štiri, kot najhujša se definitivno beleži španska gripa, ki je po prvi svetovni vojni zahtevala čez 40 milijonov življenj. Pandemija novega koronavirusa pa je od 631 milijonov prijavljenih okužb zahtevala 6,6 milijonov smrti (podatek oktobra 2022). [11]

3 GRAFIČNI PRIKAZ PODATKOV

Grafični prikaz podatkov (“data visualization”) je način, kako lahko velike količine podatkov predstavimo tako, da je človeku lažje razumljivo in da lahko le v nekaj trenutkih razume bistvo nabora podatkov. Glavni cilj vizualnega prikaza podatkov je hitrejša identifikacija vzorcev, trendov in odstopanj v ogromnih količinah podatkov. Podatek pa je neka informacija, ki jo izrazimo opisno ali številčno, in omogoča, da neko stvar bolj podrobno opišemo. Nekaj primerov podatkov so na primer podatki o sestavi zraka, rojstni podatki, podatek o številu prebivalcev in podobno.

Vizualizacija podatkov je eden izmed korakov v procesu podatkovne znanosti. Prej kot lahko podatke na najprimernejši način prikažemo, pa jih moramo zbrati. To lahko naredimo z anketami, merjenjem ali pa jih poiščemo v drugih virih. Nato moramo podatke procesirati in urediti glede na naše potrebe, na primer izločiti tiste, ki nam ne podajo dovolj informacij ali združiti več naborov podatkov za bolj podroben opis. Za tem podatke analiziramo in jih (grafično) predstavimo z različnimi diagrami, ki so predstavljeni v nadaljevanju. Ostane nam še analiza dejanskih rezultatov in interpretacija le teh. Grafični prikaz podatkov je pomemben v različnih industrijah in karierah, od učiteljev, ki prikazujejo ocene na testih do vodilnih v podjetjih, ki prikazujejo rast podjetja ter prihodke. [8]

Poznamo različne načine prikazovanja podatkov, pri izbiri le teh pa moramo biti pozorni na vrsto podatkov, ki jih želimo predstaviti, saj različne podatke najbolj razumljivo prikažemo v različnih diagramih.

Le nekateri najbolj pogosto uporabljeni diagrami so:

a. Preglednica

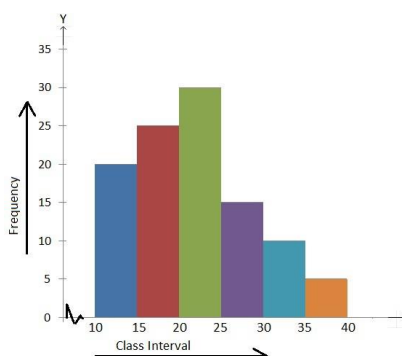
Ta način je primeren za manjše količine podatkov ali pa za pregled le nekaj lastnosti v podatkih, saj lahko postane hitro nepregledna. V preglednice lahko vpišemo številčne ali opisne podatke, ki jih uredimo po vrsticah in stolpcih.

Person	Age
Chris	38
Dennis	45
Sarah	29
Karen	47

Slika 1: Tabela [17]

b. Stolpčni diagram

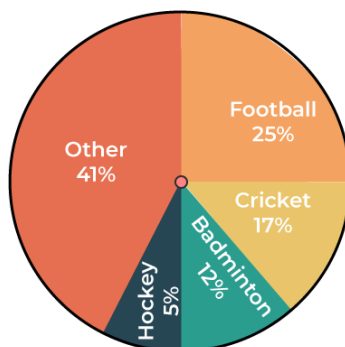
Z stolpčnim diagramom ponavadi prikazujemo podatke v obliki števil, ki jih lahko razvrstimo v neke kategorije. Sestavljen je iz vodoravne osi, na kateri so razvrščeni razredi/kategorije in navpične, ki je oštevilčena in namenjena šteju podatkov.



Slika 2: Stolpčni diagram [18]

c. Tortni diagram

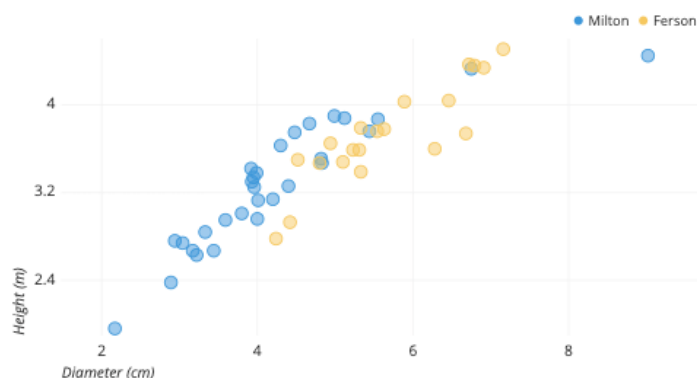
Pri tortnem diagramu prikazujemo številske podatke v krogu, ki ga razdelimo na več delov. Uporablja se takrat, ko želimo prikazati neko razmerje med različnimi skupinami/razredi v podatkih, ki pa skupaj tvorijo neko celoto.



Slika 3: Tortni diagram [19]

d. Točkovni diagram

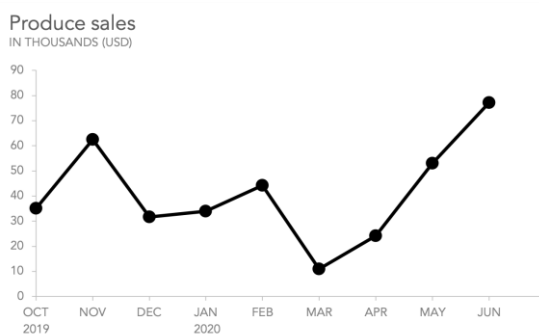
V točkovnem diagramu podatke prikazujemo s točkami v koordinatnem sistemu, prikazujejo pa relacijo med spremenljivkama na x in y osi.



Slika 4: Točkovni diagram [20]

e. Linijski diagram

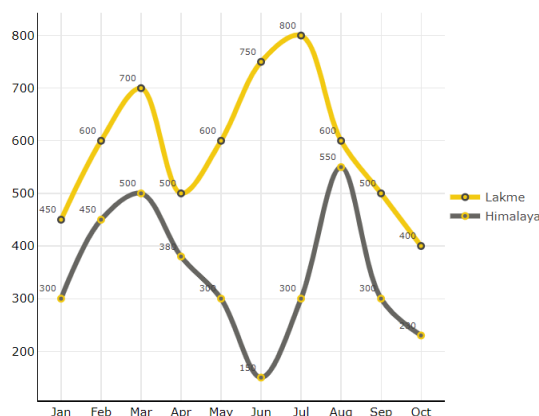
Linijski diagram je, podobno kot točkovni sestavljen iz točko v koordinatnem sistemu, ki prikazujejo odnos med x in y spremenljivkama, le da so točke v tem primeru med sabo povezane z ravnimi linijami. S tem diagramom lahko bolj nazorno prikažemo večje spremembe v relaciji z drugo spremenljivko med različnimi vrednostnimi.



Slika 5: Linijski diagram [21]

f. Krivuljni diagram

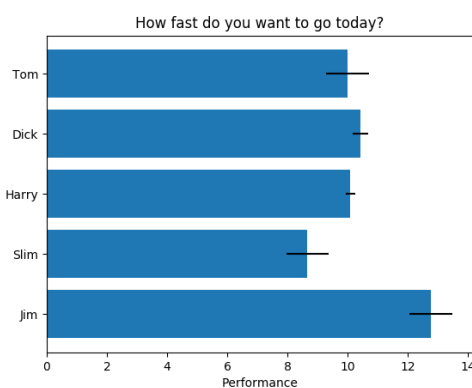
Podobno kot pri linijskem diagramu krivuljni diagram sestavljajo med sabo povezane točke v koordinatnem sistemu, le da v tem, primeru niso povezane z ravnimi linijami pač pa s krivuljami.



Slika 6: Krivuljni diagram [19]

g. Bločni diagram

Bločni diagram je v uporabi in izgledu močno podoben stolpčnem, le da so v primeru bločnega navpični stolpci nadomeščeni z vodoravnimi pravokotniki. Spremenljivka, ki prikazuje razrede/kategorije je torej na navpični osi, na vodoravno pa zapišemo številke.

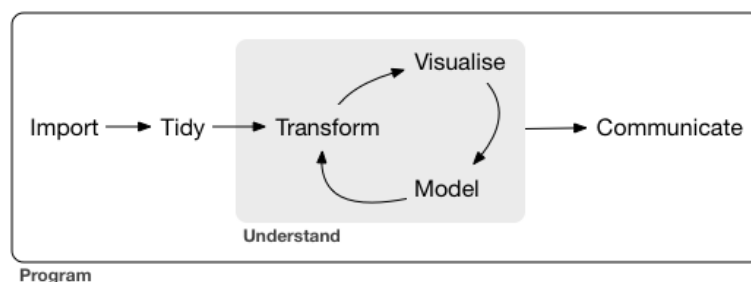


Slika 7: Bločni diagram [22]

3.1 R in Rstudio

R je programski jezik in tudi programsko orodje, ki ga uporabljamo za statistično in vizualno obdelavo podatkov. Je univerzalni jezik, kompatibilen z platformami Windows, Macintosh, UNIX in Linux. Leta 1993 sta ga razvila Ross Ihaka in Robert Gentleman na Aucklandski univerzi na Novi Zelandiji, iz prvih črk njunih imen izhaja tudi ime jezika. Uporablja pa se pri podatkovnem rudarjenju, bioinformatiki, statistiki in analizi podatkov, z njim pa lahko počistimo, analiziramo in grafično prikažemo velike količine podatkov. Uporaben je tudi za stojno učenje, z operacijami kot sta regresija in klasifikacija. Ne uporablja se zgolj med akademiki, temveč ga vsakodnevno uporabljajo tudi podjetja kot so Uber, Google, Meta, Airbnb in podobni. Implementira algoritme strojnega učenja, knjižnice jezika pa so napisane v jeziku R samem, težje funkcije pa v jezikih C, C++ in Fortran. Po rezultatih anket [9] je R najbolj uporabljen jezik pri podatkovnem rudarjenju.

Pri delu s podatki v programskem jeziku R moramo najprej podatke iz datoteke, podatkovne baze oziroma mesta, kjer so shranjeni naložiti v tako imenovan “data frame”. Za tem moramo podatke urediti, tj. odstraniti nepomembne spremenljivke ali pa vnose, ki nam ne dajo vseh želenih vrednosti. Ko imamo podatke urejene lahko začnemo z obdelavo le teh. V tem koraku lahko ohranimo le nekaj lastnosti podatkov, ki nas zanimajo, lahko iz obstoječih podatkov izračunamo nove, štejemo koliko krat se neka vrednosti pojavi v podatkih in podobno.



Slika 8: Delo s podatki

Ena glavnih lastnosti R je odprtokodnost, ki omogoča da lahko vsak (brezplačno) dostopa do kode, ki se uporablja za zagon programa in doda svojo. To pomeni, da lahko R vedno izvaja najnovejše statistične analize in hitro odpravi napake v kodi.

Vsak lahko piše svojo R kodo, zato lahko posledično vsak doda v veliko kolekcijo orodij R-ja. Programerji dodajo kodo v obliki paketov, ki so lahko splošni ali pa fokusirani na točno določen način analize. Primer je paket “pwr”, ki ga je ustvaril Stephane Champely za analizo moči. Trenutno je na voljo kar 18800 paketov različnih avtorjev objavljenih na CRAN (The Comprehensive R Archive Network), do kode katerih lahko dostopa vsak. Eden izmed razlogov, zakaj se veliko ljudi odloča za R ob obdelavi podatkov je ta, da je za razliko od bolj “tradicionalnih” metod R programski jezik. Torej imamo celoten postopek, ki smo ga nad podatki izvajali zapisan v skripti in ga lahko večkrat ponovimo, tudi nad različnimi podatki, predvsem pa ne pozabimo postopkov, ki smo jih uporabili. Te postopke lahko kasneje delimo s komerkoli, ki jih lahko brez dodatnega dela uporabi tudi sam.

Rstudio pa je brezplačno, odprtokodno integrirano razvojno okolje za R, ki ga namestimo po namestitvi R-ja in ga uporabljamo za pisanje in urejanje kode ter pregled rezultatov. Uporabniški vmesnik je oblikovan tako, da lahko uporabnik vidi programsko kodo, grafe, tabele s podatki in rezultat programa naenkrat. Omogoča tudi da uvozimo podatke iz CSV, Excel, Sas in drugih vrst datotek brez pisanja obsežne kode.

Na voljo sta dve vrsti, RStudio Desktop, ki je navadna računalniška aplikacija in pa RStudio Server, ki deluje na oddaljenem strežniku in dovoli dostop do Rstudio kar iz

spletnega brskalnika. Poleg RStudio razvojnega okolja RStudio PBC razvija in vzdržuje tudi nekaj pomembnih paketov za delo z R. Eden izmed teh je Tidyverse, ki je uporaben pri podatkovnih znanostih, torej urejanju in prikazovanju podatkov. Drugi je RMarkdown, ki omogoča mešanje kode več programskih jezikov z besedilom, kjer lahko posamezne dele kode obrazložimo tudi pisno in predstavimo delovanje oziroma analiziramo rezultate in grafe. Še eden izmed pomembnejših je Shiny. [15]

3.2 RShiny

Shiny je R paket, ustvarjen leta 2012, ki omogoča preprostejše grajenje spletnih aplikacij kar v R. To omogoča brez poznavanja tehnologij, ki so ponavadi prisotne pri ustvarjanju spletnih aplikacij kot so HTML, CSS in JavaScript, z le temi pa lahko Shiny aplikacijo nadgradimo. Posebej uporaben je za aplikacije, ki jih poganjajo podatki in vsebujejo grafični prikaz teh podatkov. Z uporabo R ustvarimo uporabniški vmesnik in strežnik, Shiny pa sestavi kodo v kombinaciji HTML/CSS/JavaScript, ki je potrebna, da aplikacija deluje na spletu. Ker je aplikacija v ozadju zgrajena v R, lahko opravlja vse funkcije in operacije, ki jih izvajamo na računalniku.

Shiny omogoča zbiranje vnosov/ukazov z spletne strani, ki jih R izvede nad podatki, in vrne kot rezultat v spletni aplikaciji. Shiny aplikacije imajo dva sestavna dela, uporabniški vmesnik in strežniško skripto, kamor pišemo kodo za preoblikovanje in urejanje nabora podatkov. [16]

4 NABOR PODATKOV

Pred začetkom izdelave aplikacije začnemo s krajšo raziskavo o različnih obstoječih naborih podatkov o pandemiji Covid-19. Ob pregledu podatkov iz različnih virov, moramo biti pozorni na nekaj ključnih stvari:

seveda, da so podatki o (čim več) državah posodobljeni redno,
da vključujejo podatke vseh držav sveta,
da vključujejo vse glavne podatke, kot so število novih primerov, smrti in cepljen
da vključujejo še čim več drugih relevantnih podatkov, na primer povprečna starost prebivalstva

Ob pregledu nekaj različnih naborov podatkov, kot so Googlov COVID-19 Open Data Repository, COVID-19 Dataset iz Kaggle.com in podobnih, sem se odločila za uporabo podatkov "Data on COVID-19 (coronavirus) by Our World in Data" [18]. Nabor podatkov je redno posodobljen (vsak dan) in vključuje vse glavne informacije, ki sem jih želela tako ali drugače prikazati v aplikaciji.

4.1 Pregled podatkov

Nabor podatkov vključuje vse kontinente in države sveta in ima 67 stolpcev. Le nekateri v tem primeru najpomembnejši so:

- *date*
- *location*
- *continent*
- *iso_code*: posebna koda za vsako državo, sestavljena iz treh črk
- *total_cases*: število vseh (do dneva vpogleda) potrjenih primerov v državi
- *new_cases*: število potrjenih primerov v izbrani državi na določen dan
- *total_cases_per_million*: število vseh (do dneva vpogleda) potrjenih primerov v državi za vsak 1.000.000 prebivalcev
- *new_cases_per_million*: število potrjenih primerov na 1.000.000 prebivalcev v izbrani državi na določen dan
- *total_deaths*: število vseh (do dneva vpogleda) potrjenih smrti zaradi Covida v državi
- *new_deaths*: število potrjenih smrti zaradi Covida v izbrani državi na določen dan
- *total_deaths_per_million*: število vseh (do dneva vpogleda) potrjenih smrti zaradi Covida v državi za vsak 1.000.000 prebivalcev
- *new_deaths_per_million*: število potrjenih smrti zaradi Covida na 1.000.000 prebivalcev v izbrani državi na določen dan
- *total_vaccinations*: število vseh (do dneva vpogleda) administriranih doz cepiva v državi
- *new_vaccinations*: število administriranih doz cepiva v izbrani državi na določen dan
- *total_vaccinations_per_million*: število vseh (do dneva vpogleda) administriranih doz cepiva v državi za vsak 1.000.000 prebivalcev
- *new_vaccinations_per_million*: število administriranih doz cepiva na 1.000.000 prebivalcev v izbrani državi na določen dan
- *population*: število prebivalcev v državi

4.2 Priprava podatkov

```
initial_data<-readr::read_csv("https://covid.ourworldindata.org/data/owid-covid-  
data.csv")  
basic_data<-initial_data[, c(2:6,8,9,11,12,14,15,37,39,41,42)]
```

Začnemo s prenosom datoteke s podatki. Datoteka se prenese vsakič, ko se program zažene, da imamo vedno najnovejše podatke vključno z današnjim dnevom. Le te podatke shranimo v spremenljivko `initial_data`, ki jo potem “reduciramo” v spremenljivki `basic_data`, da vsebuje le tiste stolpce, ki jih bomo v aplikaciji uporabili. Podatke počistimo

(odstranimo vnose, ki nimajo številke- NA) kasneje, saj se jih želelimo znebiti le pri nekaterih delih aplikacije in ne vseh.

5 SPLETNA APLIKACIJA SHINY

5.1 Predpriprava

Pred začetkom izdelave aplikacije, začnemo z razmislekom, kaj bi v njej sploh rada prikazali in pa kako bi aplikacijo oblikovali (v podstrani). Glavne dele, ki jih želimo v aplikaciji so prikaz svetovnih številke primerov in smrti, grafični prikaz primerov, smrt in cepljenj po celem svetu, prikaz številke v državah z največ primeri, grafični prikaz poljubnega kriterija v eni državi, grafična primerjava izbranih držav po nekem kriteriju in pa enako za kontinente in pa prikaz podatkov na zemljevidu.

Aplikacijo sem se odločila razdeliti na sedem podstrani:

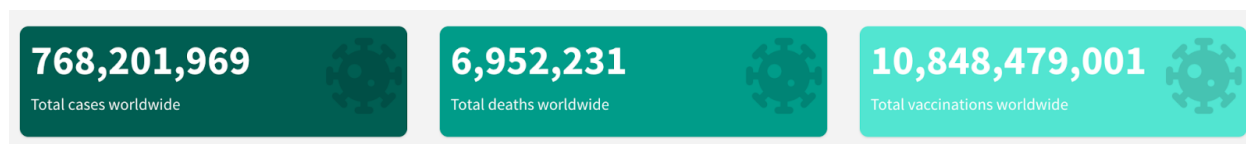
- Domača stran, kjer so predstavljeni splošni podatki o pandemiji, kot so število svetovnih primerov in primeri, smrti in cepljenje skozi čas
- Pregled po državah, kjer si lahko izberemo državo in kriterij, ki nas zanima (na primer število cepljenj), in se nam le to prikaže v obliki grafa
- Primerjava držav, kjer lahko izberemo več držav in nek kriterij, in se nam primerjava pokaže na grafu
- Pregled po celinah, kjer si lahko izberemo celino in kriterij, ki nas zanima (na primer število cepljenj), in se nam le to prikaže v obliki grafa
- Primerjava celin, kjer lahko izberemo več celin in nek kriterij, in se nam primerjava pokaže na grafu
- Prikaz na zemljevidu, kjer se nam prikaže zemljevid z obarvanimi državami, glede na izbran kriterij
- O aplikaciji, kjer le opišem z kakšnim namenom se je aplikacija izdelovala

5.2 Izvedba

5.2.1 Domača stran

Na domači strani aplikacije želimo prikazati nekaj bolj splošnih podatkov za cel svet, kot so vsi potrjeni primeri, cepljenja in smrti na svetu do sedaj, primerjava le teh tekom mesecev in prikaz nekaj držav z največ potrjenimi primeru na svetu.

Vse portjene primere, vnešena cepiva in smrti, povezane z pandemijo prikažemo za vrhu strani.



Slika 9: Prikaz glavnih podatkov

Za podatke, ki jih želimo prikazati potrebujemo seštevek vseh potrjenih primerov, smrti in pa prejetih doz cepiva po celem svetu od začetka pandemije do danes.

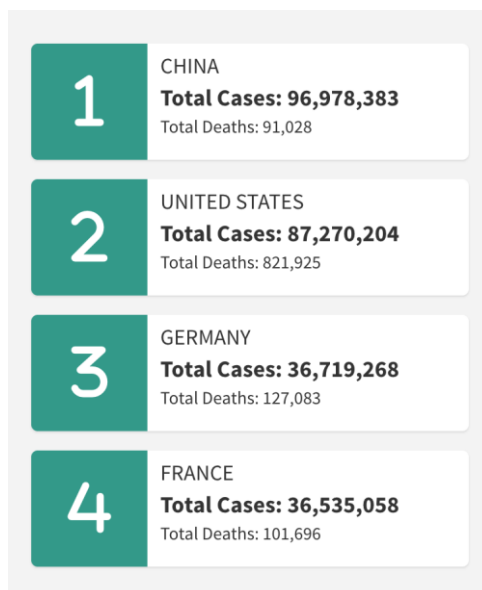
To storimo tako, da najprej iz naših osnovnih podatkov (`initial_data`) izločimo vnose, ki imajo kot lokacijo celino, saj bi nam še ti podatke podvojili. Za tem izločimo tiste stolpce, ki jih ne potrebujemo in v spremenljivko `total` shranimo le podatke, ki bodo uporabljeni za prikaz števil- `new_cases`, `new_deaths` in `new_vaccinations`. Vsak stolpec nato posebej seštejemo v svojo spremenljivko `total_cases`, `total_deaths` in `total_vaccinations`. Te spremenljivke nato prikažemo v obarvanih okencih z uporabo funkcije `valueBox`.

```
home_boxes_data <- function(){
  total <- basic_data%>%
    filter(location %!in% non_countries) %>%
    select( new_cases, new_deaths, new_vaccinations )%>%
    setNames(c( "cases", "deaths", "vaccinations"))

  total_cases <- sum((total%>%drop_na(cases))$cases)
  total_deaths <- sum((total%>%drop_na(deaths))$deaths)
  total_vacc <- sum((total%>%drop_na(vaccinations))$vaccinations)

  return(c(total_cases, total_deaths, total_vacc))
}
output$valuebox_cases <- renderValueBox({
  valueBox(
    format(
      home_boxes_data()[1], big.mark=","),
    "Total cases worldwide ",
    Icon=icon("virus-covid"),
    color="red"
  )
})
```

Pod tem, v obliki lestvice prikažemo še štiri države, v katerih so do sedaj portdili največ okužb z Covid-19.



Slika 10: Lestvica držav

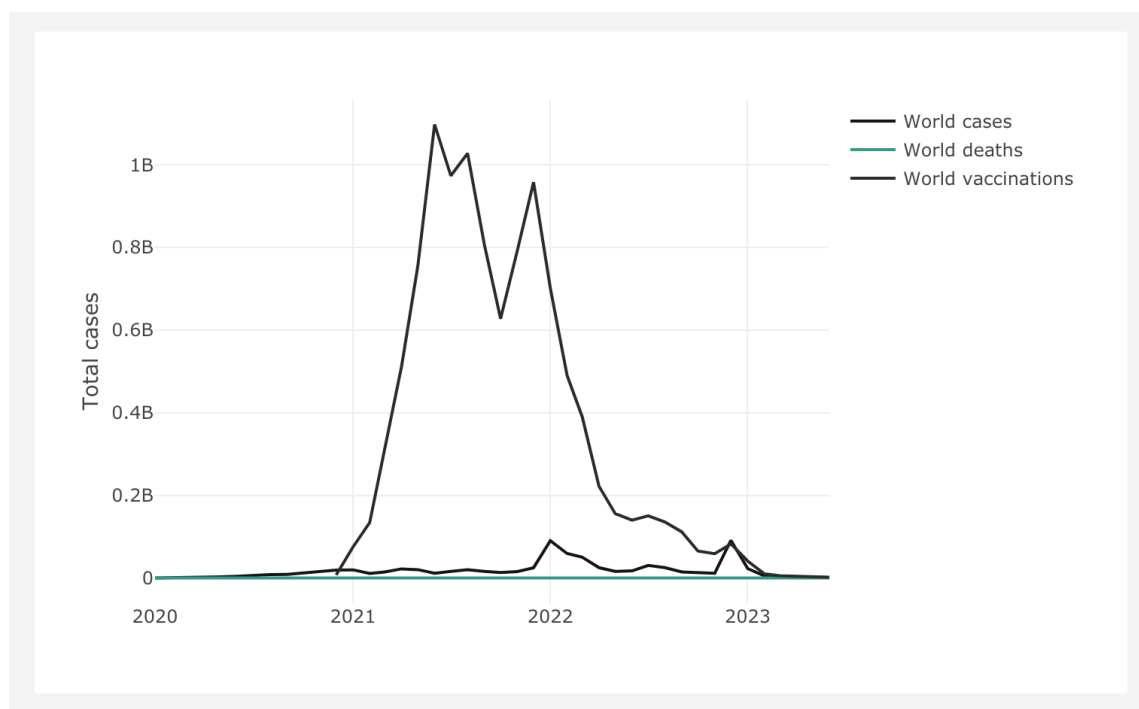
Kot v prejšnjem primeru se najprej znebimo stolpcev, ki jih ne potrebujemo in vrstic z podatki o celinah. Nato vse vrstice, v katerih so podatki za isto državo, združimo tako, da primere in smrti seštejemo. Tako imamo za vsako državo po eno vrstico z vsemi podatki od začetka pandemije do sedaj. Na koncu podatke še razvrstimo po naraščajočem vrstnem redu, in jih vstavimo v lestvico po vrsti. Tako imamo lestvico 4 držav, v katerih je bilo vsega skupaj potrjenih največ primerov, z pripisanimi smrtmi. Za vsako državo podatke prikažemo z uporabo InfoBox, v katerem izpišemo ime, število primerov, število smrti in pa številko na lestvici.

```

leaderboard_data <- function(){
  leaderboard <- basic_data %>% filter(location %!in% non_countries) %>%
    select( location, new_cases, new_deaths, new_vaccinations ) %>%
    setNames(c( "location", "cases", "deaths", "vacc" )) %>%
    group_by(location) %>% drop_na(c(cases, deaths, vacc)) %>%
    summarise(num = n(),
              cases = sum(cases),
              deaths = sum(deaths),
              vacc = sum(vacc))
  leaderboard <- leaderboard[order(leaderboard$cases, decreasing = TRUE),]
  return(leaderboard)
}
output$leader_1 <- renderInfoBox({
  infoBox(
    paste0(" ", (leaderboard_data()[1,])[1]),
    paste0("Total Cases: ", format(as.numeric((leaderboard_data()[1,])[3]),
      big.mark=",")),
    paste0("Total Deaths: ", format(as.numeric((leaderboard_data()[1,])[4]),
      big.mark=",")),
    icon = icon("1"),
    color = "light-blue",
    width = 12
  )
})

```

Zraven lestvice držav želimo prikazati še graf, kateri primerja število primerov, smrti in cepljenj po celem svetu skozi čas.



Slika 11: Domača stran graf

Začnemo z tremi različnimi spremenljivkami, v katerih imamo podatke za primere, smrti in cepljenja za cel svet in celotno pandemijo. V spremenljivkah *data_home_plot_vacc*, *data_home_plot_deaths* in *data_home_plot* imamo podatke iz začetnega nabora podatkov,

le da obdržimo le datume in stolpec, ki nas zanima, nato pa seštejemo tiste podatke, ki imajo isti mesec in leto. Pred tem uporabimo funkcijo *data_home_plot_cleanup*, ki vse datume spremeni v format leto-mesec. Tako nam na koncu ostane tabela z dvema stolpci: mesec/leto in podatek, saj v grafu ne predstavljamo vsakega posameznega dneva ampak vsak mesec pandemije. Z funkcijo *homeplot_fixdate* pa še spremenimo datum iz tipa 'string' v tip 'Date', da ga lahko uporabimo za prikaz v grafu.

```
homepage_plot_merge <- function(){
  data3 <- homeplot_deaths_fixdate(data_home_plot_vacc())
  data2 <- homeplot_deaths_fixdate(data_home_plot_deaths())
  data1 <- homeplot_fixdate(data_home_plot())

  #create graph here
}
data_home_plot_cleanup <- function(dat1){
  dat1$date <- as.Date( dat1$date)
  dat1$Month_Yr <- format(as.Date(dat1$date), "%Y-%m")
  dat1$location<- NULL
  dat1$date<- NULL

  return(dat1)
}
```

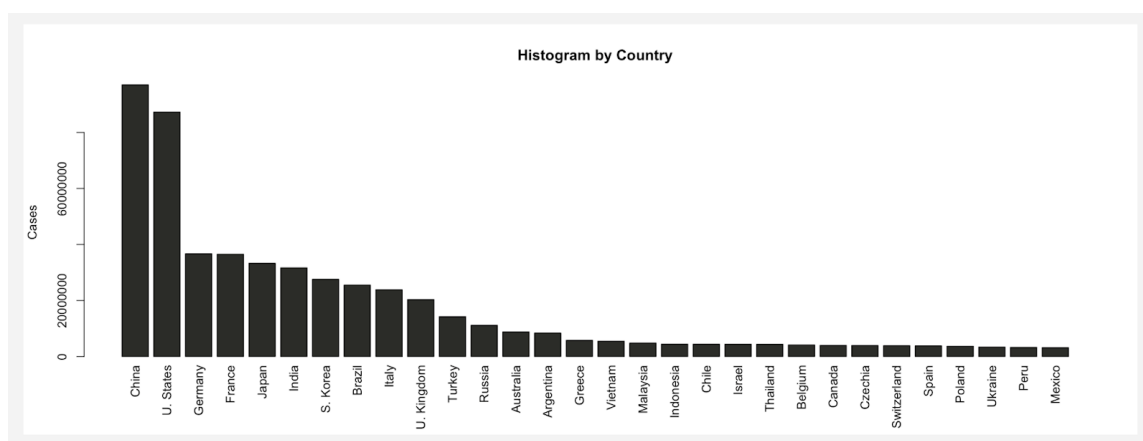
Nato ustvarimo še graf, na katerem posebej predstavimo primere, smrti in cepljenja. Ustvarimo ga z uporabo *plot_ly*, saj se tako ob postavitvi miške na mesto na grafu pojavijo podatki, ki jih ta točka na grafu predstavlja.

```

homepage_plot_merge <- function(){
  #create data here
  p <- plot_ly() %>%
    add_trace(
      x = ~data1$neki,
      y = ~data1$total_c,
      type = 'scatter',
      mode = 'lines',
      name = "World cases",
      line = list(color = '#131515', width = 2)) %>%
    add_trace(
      x = ~data2$neki,
      y = ~data2$total_d,
      type = 'scatter',
      mode = 'lines',
      name = "World deaths",
      line = list(color = '#339989', width = 2)) %>%
    add_trace(
      x = ~data3$neki,
      y = ~data3$total_v,
      type = 'scatter',
      mode = 'lines',
      name = "World vaccinations",
      line = list(color = '#2B2C28', width = 2)) %>%
    layout(
      xaxis = list(title = ""),
      yaxis = list(title = "Total cases"),
      legend = list(title = "Legend"),
      yaxis = list(tickformat = ",")
    )
  return(p)
}

```

Kot zadnji na domači strani imamo še graf, ki nam vizualno prikazuje potrjene Covid-19 primere v 30 državah z največjim številom le teh.



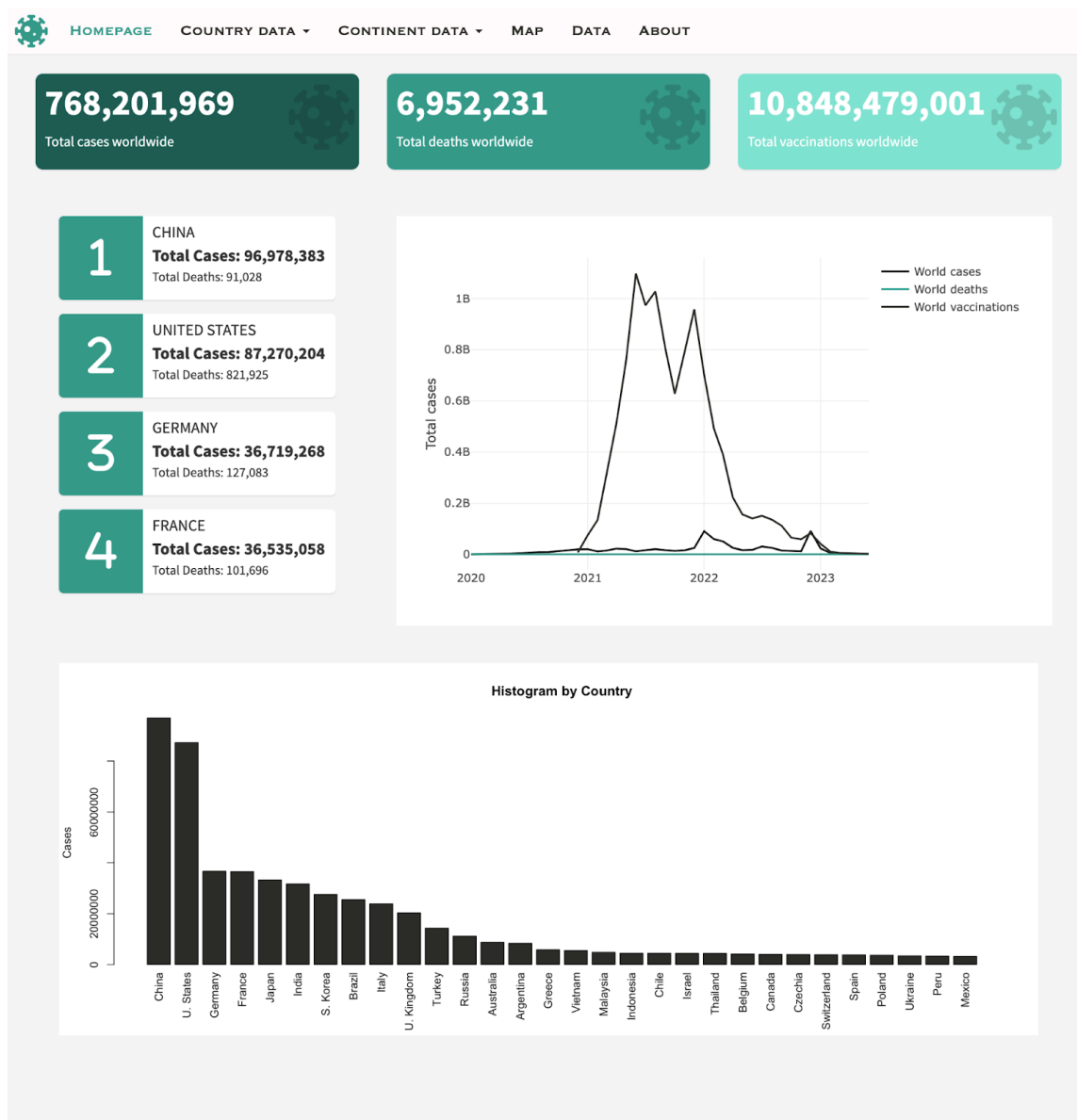
Slika 12: Domača stran histogram

Za podatke lahko uporabimo kar tiste, ki smo jih uporabili za lestvico držav z največ primeri *leaderboard_data*, le da nas v tem primeru zanima prvih 30 držav, namesto 4. Iz teh podatkov nato ustvarimo histogram, kjer en stolpec predstavlja eno državo. Če je ime

države sestavljeno iz več besed, ga še spremenimo tako, da prvo besedo skrajšamo na le začetno črko.

```
homepage_hist <- function(){  
  data <- leaderboard_data()%>%head(30)  
  options(scipen = 999)  
  bar <- barplot(  
    data$cases,  
    col = "#2B2C28",  
    border = "black",  
    ylab = "Cases",  
    main = "Histogram by Country")  
  
  modified_labels <- sapply(  
    strsplit(data$location, " "),  
    function(x) {  
      if (length(x) > 1) {  
        paste0(substr(x[1], 1, 1), ". ", paste(x[-1], collapse = " "))  
      } else {  
        x  
      }  
    })  
  text(x = bar,  
       y = par("usr")[3] - 0.02 * (par("usr")[4] - par("usr")[3]),  
       labels = modified_labels,  
       srt = 90,  
       adj = c(1, 0.5),  
       xpd = TRUE)  
  
  return(bar)  
}
```

Tako imamo vse dele, si sestavljajo domačo stran aplikacije.



Slika 13: Domača stran

5.2.2. Pregled po državah

Na naslednji podstrani želimo imeti možnost izbrati poljubno državo in pa kriterij, ki nas zanima, le ta pa se nam prikaže na grafu v tudi poljubnem časovnem intervalu. Dodati želimo še prikaz glavnih podatkov za izbrano državo v okvirčkih za hiter pregled, podobno kot na domači strani.

Začnemo z razdelkom, kjer lahko uporabnik izbere vse kriterije, po katerih želi, da se mu podatki nato prikažejo. V njem najprej želimo način izbire kriterija, ki zanima uporabnika, npr. število cepljenj, nato izbiro katerikoli države, z načinom, da uporabniku ni treba preiskati celega seznama držav, da poišče svojo. Nato želimo izbiro dveh datumov, med katerimi nas zanimajo podatki in pa še možnost za dodajo povprečja primerov v graf, pri katerem si lahko uporabnik izbere tudi za koliko dni se meri povprečje.

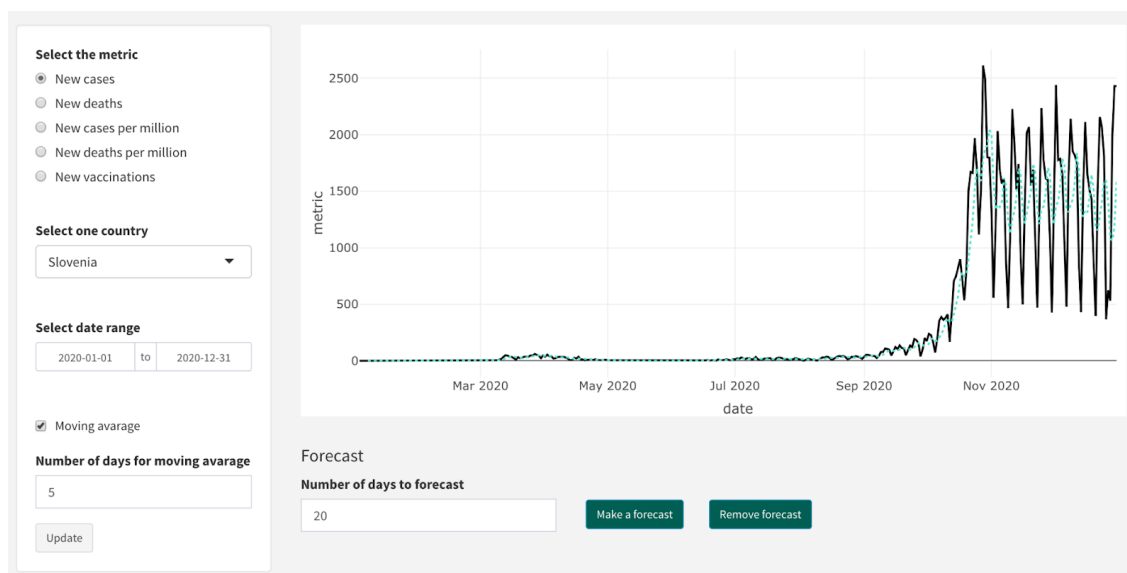
Te podatke nato uporabimo v grafu, ki se posodobi vsakič, ko spremenimo katerega koli izmed vnosov. Ustvarimo funkcijo *plot_by_country_card*, ki nam na koncu vrne graf, ki ga prikažemo na strani. Najprej preverimo, če uporabnik želi na grafu tudi linijo s povprečjem, saj glede na to izbiramo med dvema različnima podatki. V primeru, da želimo le graf s podatki in brez povprečij najprej “očistimo” podatke, kot smo jih na domači strani, le da uporabimo informacije, ki jih uporabnik vnese za izbor le ustreznih vnosov v naboru podatkov. Nato te podatke prikažemo na grafu, ki ja spet ustvarimo s pomočjo paketa Plotly.

V primeru, da želimo na grafu prikazati tudi povprečje na začetne podatke dodamo še stolpec *ma* (moving average). V tem stolpcu izračunamo mediano podatkov iz števila dni, ki ga izberemo. Nato pa te podatke uporabimo za to, da na prejšnji graf dodamo še eno linijo za povprečje.

```

plot_by_country_card <- function(){
  ifelse(
    input$moving_av_country2 ==T & !is.null(input$moving_av_days_country2),
    dat_ma <- clean_data_country_card()%>%
      group_by(location)%>%
      mutate(ma2=rollapply(metric,ma_days3(), mean,align='right',fill=NA)),
    dat_ma <- clean_data_country_card()
  )
  plt <- plot_ly (data = dat_ma, x=~date, color=~location, text = ~location)
  plt <- plt%>%add_trace(
    y=~metric,
    type='scatter',
    mode='lines',
    line = list(color = 'black'),
    hovertemplate = paste(
      paste0('<extra></extra>Country Name: %{text}\n', 'Metric:',
        name_fix(input$metric_country_card), ': %{y}\nDate: %{x} ')
    ))
  if(input$moving_av_country2==T & !is.null(input$moving_av_days_country2)){
    plt <- plt%>%add_trace(
      y=~ma2,
      type='scatter',
      mode='lines',
      line=list(dash="dot", color='#7DE2D1'), showlegend=F,
      hovertemplate = paste(
        paste0("<extra>Moving Average</extra>Country Name: %{text}\n",
          'Metric: ',name_fix(input$metric_country_card),
          ': %{y}\nDate: %{x}'))
    )
  }
  highlight(plt)
}

```



Slika 14: Graf izbrane države

Na graf lahko po želji dodamo tudi napoved podatkov za poljubno število dni naprej. To storimo tako, da ob dodajanju grafa na stran preverimo, ali si uporabnik želi napoved. Če si jo, ustvarimo prejšnji graf, v nasprotnem primeru pa kličemo funkcijo *plot_by_country_card_forecast*. V njej enako kot v prejšnjem primeru preverimo, ali želimo na graf dodati tudi povprečje in pripravimo ustrezne podatke. Nato podobno

storimo še za napoved, saj ima tudi napoved lahko povprečje. Podatke za napoved dobimo v funkciji *forecast_data_c*, katerim po potrebi dodamo še podatke za povprečje. V funkciji *forecast_data_c* združimo podatke in nove podatke o napovedih, ki jih dobimo tako, da najprej ustvarimo dodatne datume- toliko kot jih izberemo, nato pa za le te izračunamo podatke tako, da izračunamo za vsak datum mediano prejšnjih dni.

```
output$single_country_plot <- renderPlotly({  
  req(input$country_card)  
  ifelse(  
    make_forecast_c$value==0, return (plot_by_country_card()),  
    return(plot_by_country_card_forecast()))  
})
```

```

plot_by_country_card_forecast <- function(){
  ifelse(
    input$moving_av_country2 ==T & !is.null(input$moving_av_days_country2),
    dat_ma <- clean_data_country_card()%>%
      group_by(location)%>%
      mutate(ma2=rollapply(metric,ma_days3(), mean,align='right',fill=NA)),
    dat_ma <- clean_data_country_card()
  )
  ifelse(
    input$moving_av_country2 ==T & !is.null(input$moving_av_days_country2),
    forecastdata <- forecast_data_c()%>%
      group_by(location)%>%
      mutate(ma2=rollapply(metric,ma_days3(),mean,align='right',fill=NA)),
    forecastdata <- forecast_data_c()
  )

  forecasted_data2 <- rbind(forecastdata%>%
    filter(forecast==0)%>%
    filter(date==max(date)),
    forecastdata%>%filter(forecast==1))

  plt <- plot_ly (
    data = forecastdata%>%
      filter(forecast==0), x=~date, color=~location, text = ~location)

  plt <- plt%>%add_trace(
    y=~metric,
    type='scatter',
    mode='lines',
    line = list(color = 'black'),
    hovertemplate = paste(
      paste0('<extra></extra>Country Name: %{text}\n', 'Metric:',
        name_fix(input$metric_country_card), ': %{y}\nDate: %{x} ')
    ))
  if(input$moving_av_country2==T & !is.null(input$moving_av_days_country2)){
    plt <- plt %>% add_trace(
      y=~ma2,
      type='scatter',
      mode='lines',
      line=list(dash="dot", color="#7DE2D1"),
      showlegend=F,
      hovertemplate = paste(
        paste0("<extra>Moving Average</extra>Country Name: %{text}\n", 'Metric: ',
          name_fix(input$metric_country_card), ': %{y}\nDate: %{x}')) )
  }
  plt <- plt%>% add_trace(
    data = forecasted_data2 ,
    y=~metric,
    x=~date,
    color="#339989",
    showlegend = F,
    type="scatter",
    mode="lines",
    line=list(color=~location, dash="dot"),
    hovertemplate = paste(
      paste0('<extra>Forecast</extra>Country Name: %{text}\n',
        name_fix(input$metric_country), ': %{y}\nDate: %{x} ')
    ))
  highlight(plt)
}

```

Na koncu strani za hiter pregled predstavimo še glavne podatke, tudi glede na izbiro

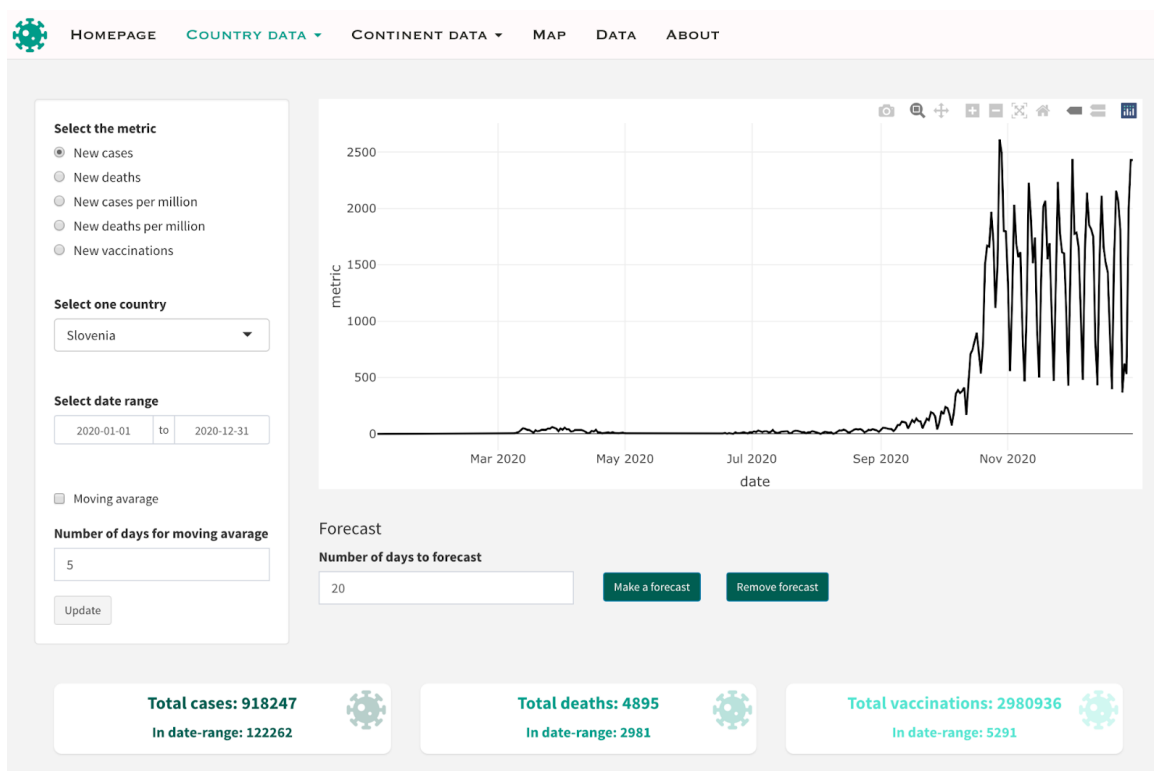
uporabnika. Le te ustvarimo na podoben način kot tiste na domači strani, le da nas tukaj poleg podatkov za državo v času celotne pandemije zanimajo tudi podatki za državo v časovnem intervalu, ki ga je izbral uporabnik. Za nabor podatkov torej, poleg začetnega uporabimo tudi tistega, katerega uporabljamo za izris grafa- torej le z podatki, ki jih je uporabnik izbral. V le teh nato seštejemo podatke za potrjene primere, cepljenja in smrti in jih prikažemo.

```
output$countryboxes <- renderUI({
  req(input$country_card)
  data2 <- func_card_info2(input$country_card, input$daterange_card[1],
    input$daterange_card[2])
  data <- leaderboard_data() %>% filter(location == input$country_card)
  fluidRow(
    div(id="count",
      valueBox(
        paste0("Total cases: ", data[3]), paste0("In date-range: ", data2[2]), icon =
          icon("virus-covid"), color = "yellow"),
    div(id="count",
      valueBox(
        paste0("Total deaths: ", data[4]), paste0("In date-range: ", data2[3]), icon =
          icon("virus-covid"), color = "blue"),
    div(id="count",
      valueBox(
        paste0("Total vaccinations: ", data[5]), paste0("In date-range: ", data2[4]),
          icon = icon("virus-covid"), color = "navy"))
  )
})
```



Slika 15: Glavni podatki države

Vsi deli nato sestavijo končno stran, na kateri si lahko pregledamo točne podatke za določene države v določenem časovnem intervalu, izračunamo povprečje in pa poskušamo napovedati primere v prihodnosti.



Slika 16: Stran podatkov držav

5.2.3. Primerjava držav

Na naslednji podstrani želimo imeti možnost primerjati več držav po izbranem kriteriju. Hkrati pa na grafu lahko, kot v prejšnjih primerih prikažemo tudi povprečje in pa napoved za izbrano število dni. Na levi strani strani imamo spet okno, v katerem lahko izberemo vse želene kriterije, le da zdaj izbiramo več držav namesto ene. Po tem ko izberemo kriterij, državo in pa datume, se nam na grafu prikažejo podatki, za vsako državo v svoji barvi. Kot v prejšnjem primeru uporabimo paket Plotly, da se nam ob postavitvi miške na določeno točko na grafu izpišejo točno podatki za tisti dan, ob izbiri prikaza povprečja in/ali napovedi, pa se tudi ti podatki prikažejo za vsako izmed izbranih držav v pravi barvi.

Na začetku spet najprej preverimo, ali želi uporabnik na grafu tudi napoved za poljubno število dni, saj glede na izbiro pokličemo dve različni funkciji, ki nam ustavarita pravi graf- `plot_by_country()`, če ne želimo napovedi in `plot_by_country_forecast` v nasprotnem primeru.

```
output$country_plot <- renderPlotly({
  req(input$country_country)
  ifelse(make_forecast$value==0,
    return(plot_by_country()),
    return(plot_data_country_forecast()))
})
```

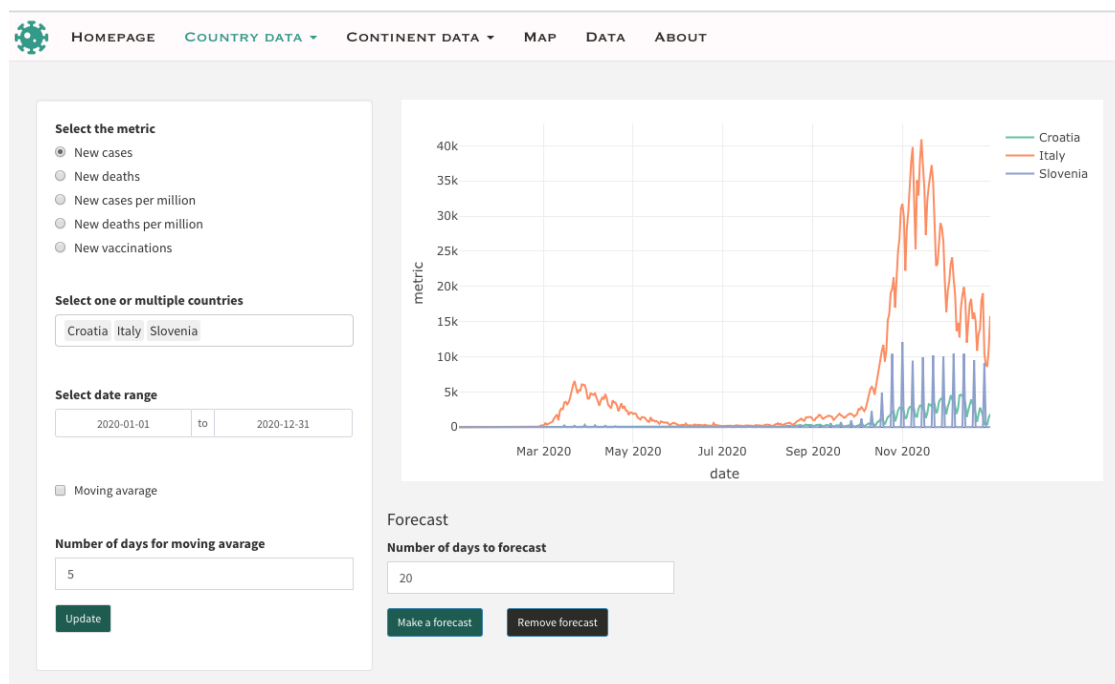
Podobno kot v prejšnjem primeru funkciji najprej preverita, ali želimo na grafu tudi prikaz

povprečja. Za primer funkcije *plot_by_country*, le ta najprej pokliče funkcijo *clean_data_country*, ki vrne nabor podatkov, v katerem so le podatki, ki uporabnika zanimajo, torej izbran kriterij in časovni interval in pa vse izbrane države. Le te pridobimo tako, da iz začetnih podatkov obdržimo le vnose, katerih država je ena izmed držav, ki jih želi uporabnik primerjati. Nato izločimo še tiste vnose, ki so izven izbranega časovnega intervala. Na le te podatke funkcija po potrebi doda še povprečje. Funkcija *plot_by_country_forecast* je skoraj identična, le da na podatke doda še napoved na enak način, kot funkcija na prejšnji podstrani. Funkcija *plot_by_country* podatke prikaže na Plotly grafu tako, da za vsako državo uporabi drugačno barvo.

```
clean_data_country <- reactive({cases_new <- basic_data %>%
  filter(location %in% input$country_country) %>%
  filter(date >= input$daterange_country[1])%>%
  filter(date <= input$daterange_country[2])%>%
  select(location, date,input$metric_country )%>%
  set_names(c("location", "date", "metric"))%>%
  arrange(date)})

plot_by_country <- function(){
  ifelse(input$moving_av_country ==T & !is.null(input$moving_av_days_country),
    dat_ma <- clean_data_country()%>%
    group_by(location)%>%
    mutate(ma2=rollapply(metric,ma_days(),mean,align='right',fill=NA)),
    dat_ma <- clean_data_country()
  )
  plt <- plot_ly (data = dat_ma, x=~date, color=~location, text = ~location)
  plt <- plt %>% add_trace(y=~metric, type='scatter', mode='lines',
    hovertemplate = paste(
      paste0('<extra></extra>Country Name: %{text}\n',
        'Metric: ',name_fix(input$metric_country),
        ': %{y}\nDate: %{x} ')
    ))
  if(input$moving_av_country==T & !is.null(input$moving_av_days_country)){
    plt <- plt %>% add_trace(
      y=~ma2, type='scatter', mode='lines', line=list(dash="dot"), showlegend=F,
      hovertemplate = paste(
        paste0("<extra>Moving Average</extra>Country Name: %{text}\n",
          'Metric: ',name_fix(input$metric_country),': %{y}\nDate: %{x}')
      )) )
  }
  highlight(plt)
}
```

Tudi na tej podstrani imamo poleg grafa še polja za vnos podatkov na levi strani in pa polje za dodajanje napovedi pod grafom. Ob grafu se prikaže tudi legenda, katera barva predstavlja katero državo.



Slika 17: Stran primerjava držav

5.2.4. Pregled po celinah

Na naslednjih dveh podstraneh, podobno kot za države podatke prikažemo še za celine. Prva izmed le teh prikazuje podatke za izbran kriterij, v izbranem časovnem intervalu, za izbrano celino. Kot v prejšnjih primerih lahko grafu dodamo tudi povprečje in pa napoved.

Spet začnemo tako, da preverimo ali želimo na grafu tudi napoved in pokličemo ustrezno funkcijo, ki nam bo vrnila graf, glede na izbran kriterij.

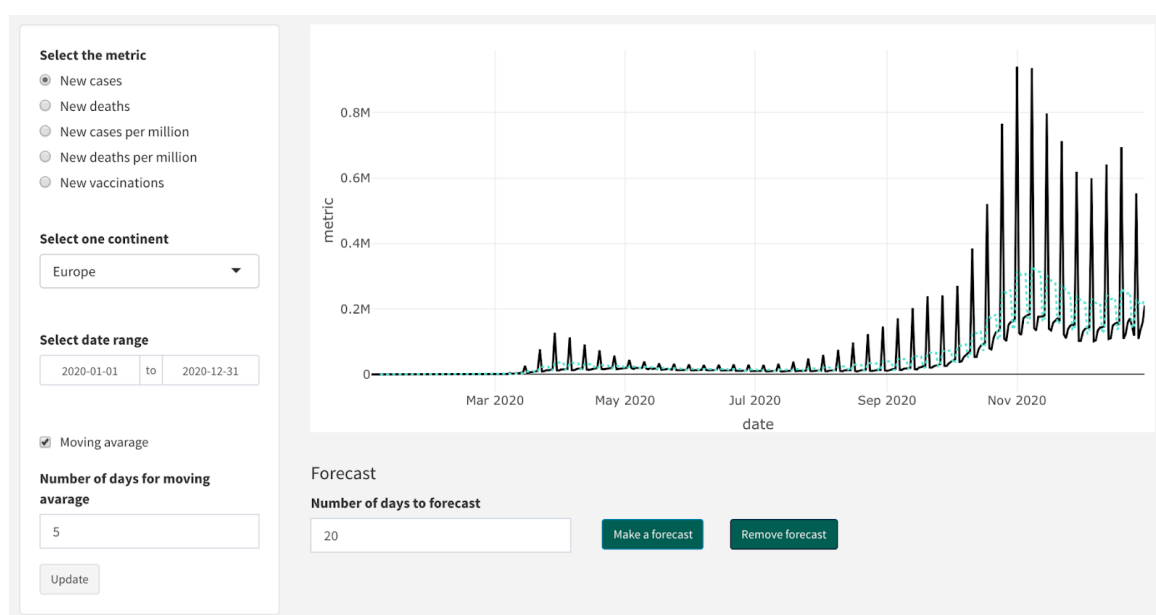
```
output$single_continent_plot <- renderPlotly({
  req(input$continent_card)
  ifelse(make_forecast_c2$value==0,
    return(plot_by_continent_card()),
    return(plot_by_continent_card_forecast()))
})
```

Funkcija *plot_by_continent_card*, ki nam vrne ustrezen graf v primeru, da napovedi ne želimo prikazane, najprej preveri ali naj v graf doda tudi povprečje. Podatke za graf pridobi z klicem funkcije *clean_data_continent_card*, katera iz izvornih podatkov izloči vse vnose, ki jih ne želimo imeti prikazane- obdrži le podatke za izbran kontinent v izbranem časovnem intervalu.

```

plot_by_continent_card <- function(){
  ifelse(input$moving_av_country3 == T & !is.null(input$moving_av_days_country3),
    dat_ma <- clean_data_continent_card() %>% group_by(location) %>%
      mutate(ma2=rollapply(metric,ma_days4(),mean,align='right',fill=NA)),
    dat_ma <- clean_data_continent_card()
  )
  plt <- plot_ly (data = dat_ma, x=~date, color=~location, text = ~location)
  plt <- plt %>% add_trace(y=~metric, type='scatter', mode='lines',
    line = list(color = 'black'),
    hovertemplate = paste(paste0(
      '<extra></extra>Continent Name: %{text}\n', 'Metric: ',
      name_fix(input$metric_continent_card),
      ': %{y}\nDate: %{x} '))
  ))
  if(input$moving_av_country3 == T & !is.null(input$moving_av_days_country3)){
    plt <- plt %>% add_trace(y=~ma2, type='scatter', mode='lines',
      line=list(dash="dot", color='#7DE2D1'), showlegend=F,
      hovertemplate = paste(paste0(
        "<extra>Moving Average</extra>Continent Name:
        %{text}\n",
        'Metric: ', name_fix(input$metric_continent_card),
        ': %{y}\nDate: %{x}'))))
  }
  highlight(plt)
}

```

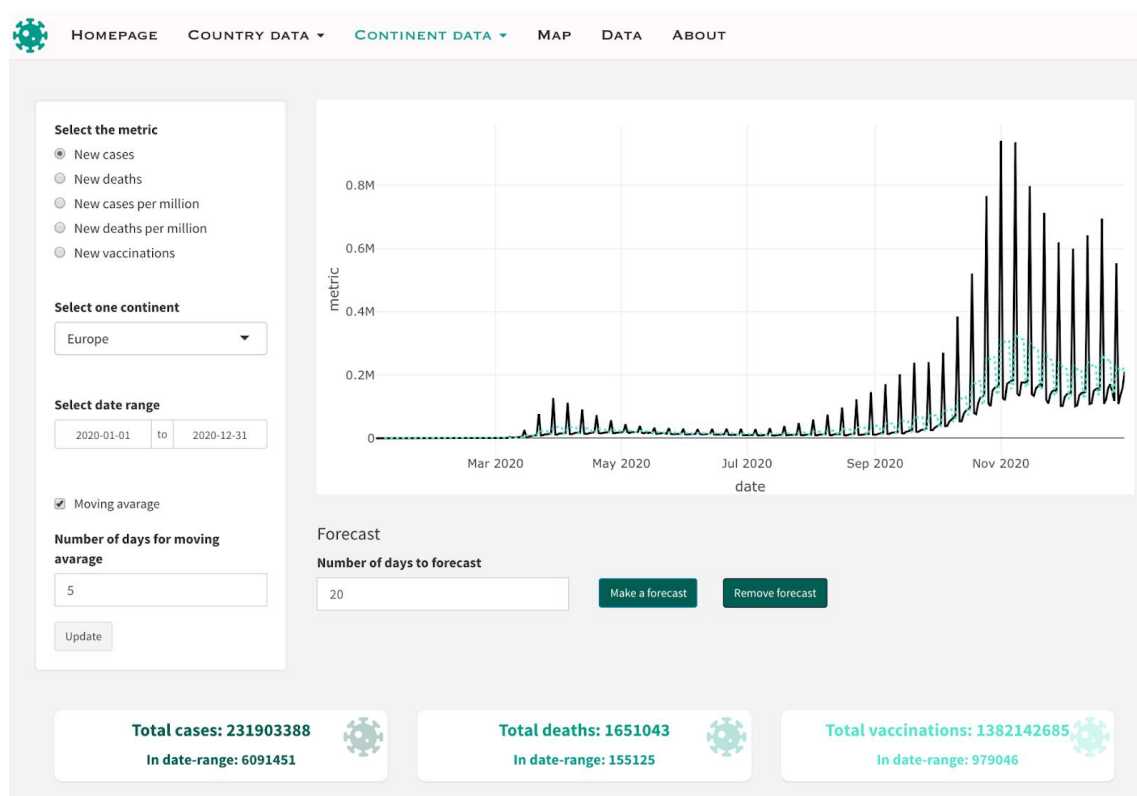


Slika 18: Graf izbrane celine

Levo od grafa imamo tudi na tej strani okno, v katerem lahko izberemo vse kriterije. V primeru prikaza podatkov za celine, moramo imeti samo te na izbiro v oknu, kjer izberemo zeleno celino. Le to dosežemo tako, da naredimo seznam continents, v katerega zapišemo imena vseh celin in podatke iz le tega imamo nato na voljo za izbiro.

Pod grafom imamo kot na preostalih straneh tudi okno za izbiro prikaza napovedi, pod le tem pa še prikaz ključnih podatkov za hiter pregled. Tu prikažemo podatke za izbrano

celino za celotno pandemijo in pa podatke za izbrani časovni interval na celini. Te izračunamo tako, da iz osnovnih podatkov izločimo vse zapise razen tistih, ki imajo v polju države zapis eno izmed celin iz prej omenjenega seznama continents, in seštejemo vse vrednosti določenega kriterija.



Slika 19: Stran podatkov celin

5.2.5. Primerjava celin

Naslednja podstran omogoča primerjavo podatkov večih celin na grafu, seveda tudi z možnostjo prikaza povprečja in napovedi. Na levi strani imamo kot na vseh ostalih okno, v katerem izberemo kriterije, desno pa graf, na katerem so prikazani podatki za več celin, vsaka v svoji barvi.

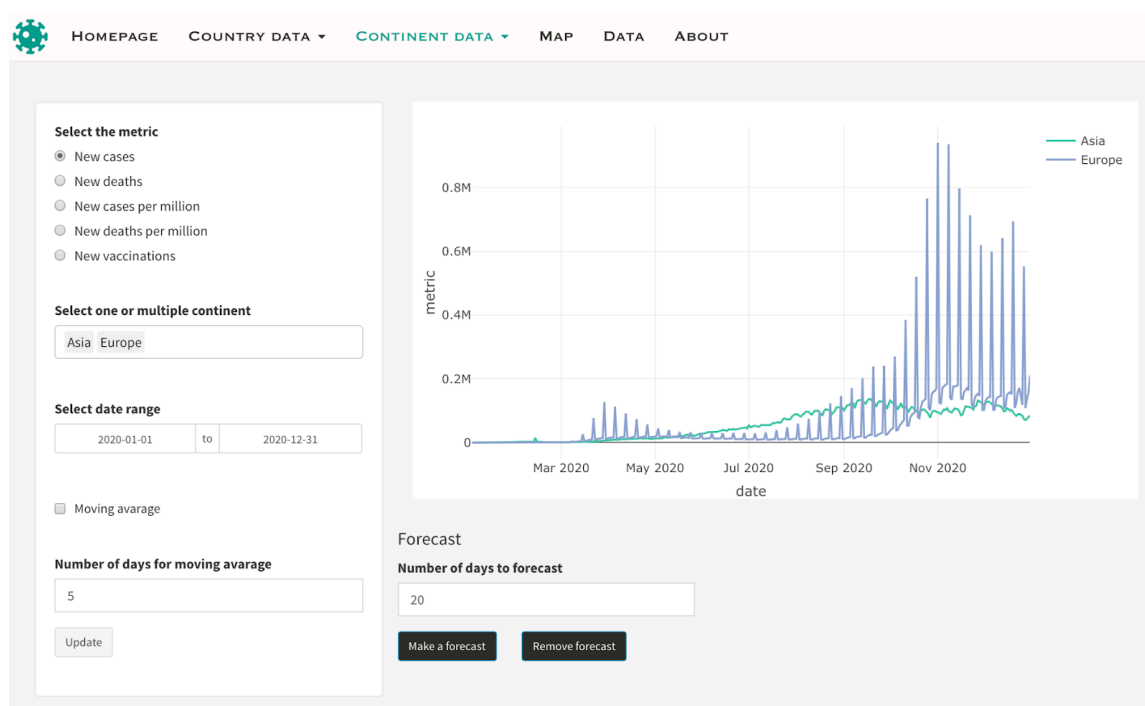
Najprej preverimo, ali želimo v grafu tudi napoved in skladno s tem izberemo funkcijo, ki vrne graf. Funkcija `plot_by_continent_card`, ki vrne graf brez napovedi, najprej preveri, ali naj na graf doda tudi povprečje. Nato ustvari graf, na katerem je predstavljen vsaka izmed izbranih celin z eno barvo, podatke pa pridobi s klicem funkcije `clean_data_continent_card`, katera iz začetnega nabora podatkov izloči vse vnose, ki se ne ujemajo z vnešenimi kriteriji.

```

output$single_continent_plot <- renderPlotly({
  req(input$continent_card)
  ifelse(make_forecast_c2$value==0,
    return(plot_by_continent_card()),
    return(plot_by_continent_card_forecast()))
})
plot_by_continent_card <- function(){
  ifelse(input$moving_av_country3 ==T & !is.null(input$moving_av_days_country3),
    dat_ma <- clean_data_continent_card()%>%
      group_by(location)%>%
      mutate(ma2=rollapply(metric,ma_days4(),mean,align='right',fill=NA)),
    dat_ma <- clean_data_continent_card()
  )

  plt <- plot_ly (data = dat_ma, x=~date, color=~location, text = ~location)
  plt <- plt %>% add_trace(y=~metric, type='scatter', mode='lines',
    line = list(color = 'black'),
    hovertemplate = paste(paste0(
      '<extra></extra>Continent Name: %{text}\n', 'Metric: ',
      name_fix(input$metric_continent_card),
      ': %{y}\nDate: %{x} ' )
    ))
  if(input$moving_av_country3==T & !is.null(input$moving_av_days_country3)){
    plt <- plt %>% add_trace(y=~ma2, type='scatter', mode='lines',
      line=list(dash="dot", color='#7DE2D1'), showlegend=F,
      hovertemplate = paste(paste0(
        "<extra>Moving Average</extra>Continent Name: %{text}\n",
        'Metric: ', name_fix(input$metric_continent_card),
        ': %{y}\nDate: %{x}')) )
  }
  highlight(plt)
}

```



Slika 20: Graf primerjave celin

5.2.6. Prikaz na zemljevidu

Na naslednji strani prikažemo podatke še na zemljevidu sveta. Na vrhu strani imamo okno, v katerem lahko izberemo za kateri kriterij želimo prikaz podatkov in izbiri enega datuma, zraven pa tudi gumb za možnost izbire dveh datumov, med katerimi nato izračunamo povprečje. Na strani nato prikažemo zemljevid sveta, na katerem so države obarvane glede na izbrani kriterij in legendo, prikazano desno spodaj.

Najprej preverimo, ali je uporabnik izbral le en datum za prikaz podatkov ali pa časovni okvir in glede na to pokličemo različni funkciji, ki vrnete zemljevid. V funkciji *worldmap* najprej k začetnim podatkom, ki smo jih uporabili za ostale grafe v aplikaciji dodamo še 3-mestne kode za države, ki jih potrebujemo za izdelavo zemljevida. Nato v spremenljivko *world_spdf* shranimo datoteko, ki vsebuje obris zemljevida vseh držav sveta. Le ti dve spremenljivki nato združimo v eno z uporabo funkcije *merge*. Nato z uporabo paketa *Leaflet* in končnih podatkov ustvarimo zemljevid sveta.

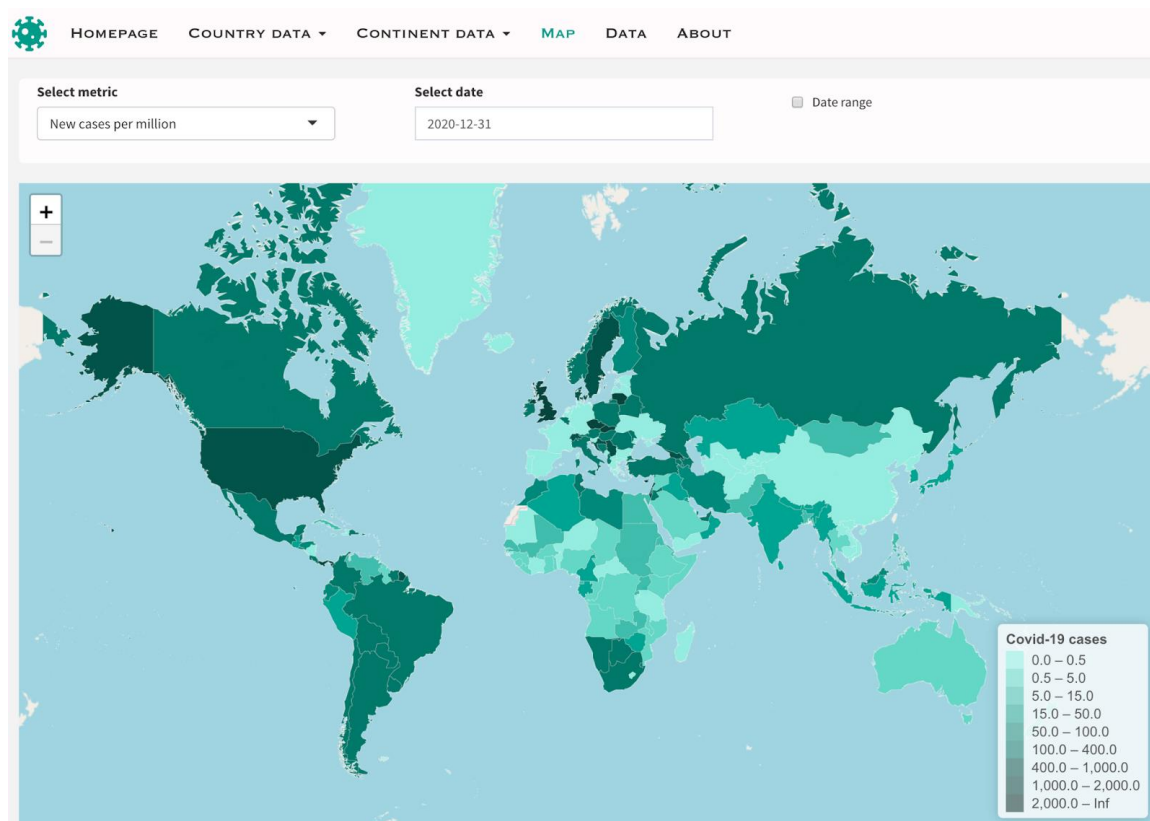
```

output$world_map <- renderLeaflet({
  req(input$map_date_i)
  if (!input$checkbox_map){
    worldmap()
  }else{
    req(input$map_dates_i)
    worldmap_range()
  }
})
worldmap <- function(){
  dataf = merge(x = clean_data_map(), y = dat, by = "location")
  world_spdf = readOGR(dsn=getwd(), layer="TM_WORLD_BORDERS_SIMPL-0.3")
  wrldx <- merge(world_spdf, dataf, duplicateGeoms = T)
  wrldx$metric = as.numeric(as.character(wrldx$metric))
  bins = c(0, 0.5, 5, 15, 50, 100, 400, 1000, 2000, Inf)
  pallmy = c("#a0ebdf", "#75d1c3", "#55b5a6", "#339989", "#237d6f", "#16695c", "#0b423a",
    "#06332c", "#011a16" )
  pal = colorBin(palette = pallmy, domain=wrldx$metric, na.color = "transparent",
    bins=bins)

  customLabel = paste("Country: ", wrldx$NAME, "<br/>", input$map_metric_i,
    ": ", wrldx$metric, sep = "") %>%lapply(htmltools::HTML)

  wrldmap <- leaflet(wrldx)%>%
    addProviderTiles(providers$OpenStreetMap,
      options = tileOptions(minZoom=2, maxZoom=8))%>%
    addPolygons(fillColor = ~pal(metric),
      fillOpacity = 0.9,
      stroke = TRUE,
      color="white",
      highlight=highlightOptions(
        weight = 5,
        fillOpacity = 0.3),
      label = customLabel,
      weight = 0.3,
      smoothFactor = 0.2)%>%
    addLegend(
      pal=pal,
      values = ~metric,
      position = "bottomright",
      title = "Covid-19 cases"
    )
  return(wrldmap)
}

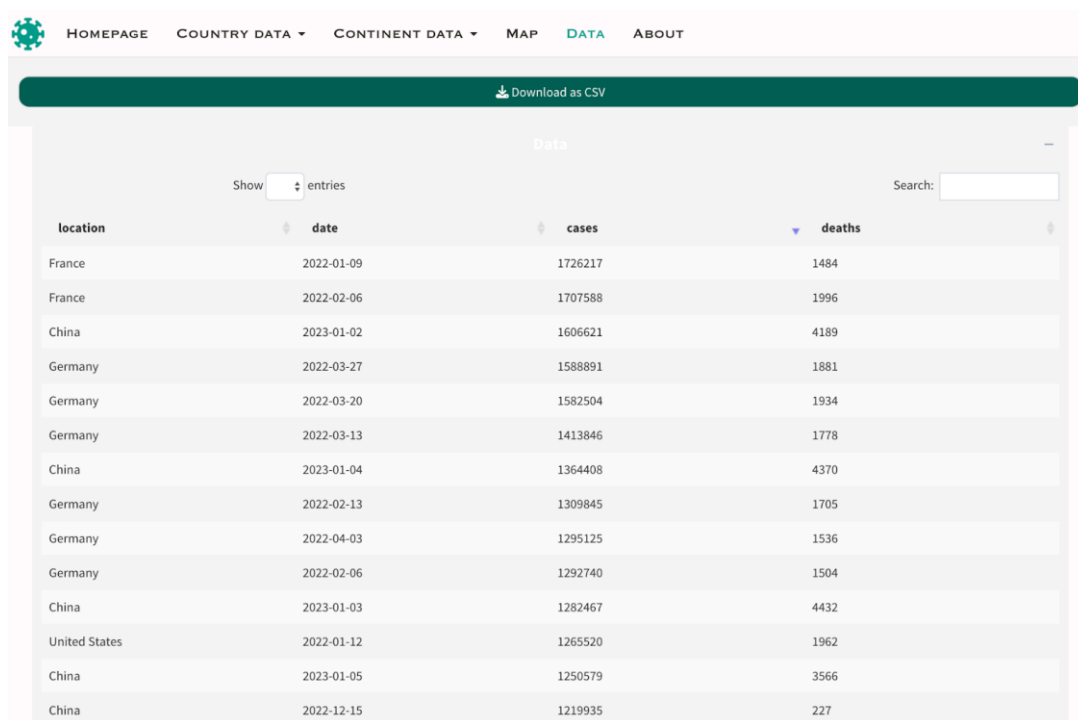
```



Slika 21: Zemljevid

5.2.7. Podatki

Na naslednji strani prikažemo vse podatke celotne pandemije v obliki razpredelnice, v kateri lahko razvrščamo po kriterijih, datumu ali državah. Na vrhu strani dodamo še gumb, s katerim si lahko uporabnik prenese podatke, ki jih uporablja aplikacija.

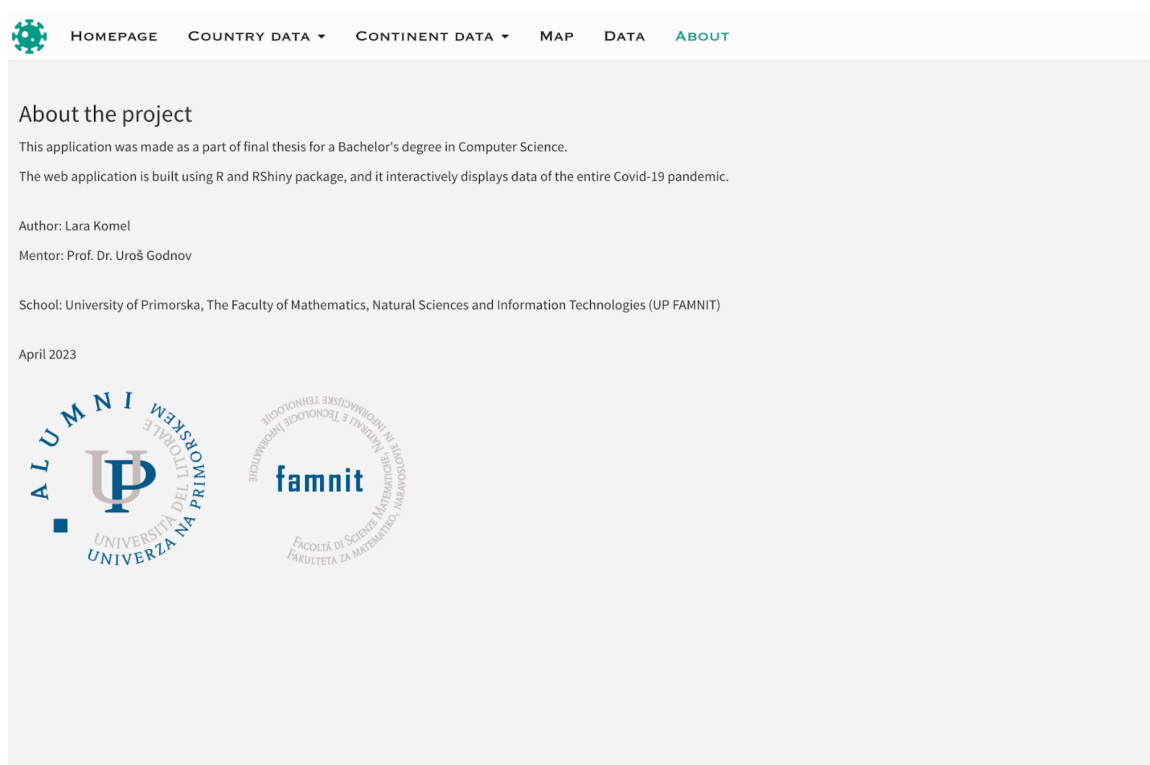


location	date	cases	deaths
France	2022-01-09	1726217	1484
France	2022-02-06	1707588	1996
China	2023-01-02	1606621	4189
Germany	2022-03-27	1588891	1881
Germany	2022-03-20	1582504	1934
Germany	2022-03-13	1413846	1778
China	2023-01-04	1364408	4370
Germany	2022-02-13	1309845	1705
Germany	2022-04-03	1295125	1536
Germany	2022-02-06	1292740	1504
China	2023-01-03	1282467	4432
United States	2022-01-12	1265520	1962
China	2023-01-05	1250579	3566
China	2022-12-15	1219935	227

Slika 22: Prikaz podatkov

5.2.8. O aplikaciji

Na zadnji strani je še kratek opis namena izdelave celotne aplikacije.



Slika 23: Stran o aplikaciji

6 ZAKLJUČEK

S koncem izdelave diplomske naloge imamo torej funkcionalno spletno aplikacijo, ki omogoča pregled podatkov o pandemiji Covid-19 po katerikoli državi ali celini sveta, primerjavo več držav ali celin po željenih kriterijih, prikaz podatkov na zemljevidu sveta in pregled podatkov v obliki razpredelnice. V aplikaciji je zagotovo še veliko prostora za nadgradnje, kot so dodajanje pogleda, v katerem bi lahko primerjali potrjene primere ali smrti z različnimi dejavniki v državi, na primer povprečni prihodek, število kobilcev, razvitost držav in podobni. Si pa tudi v trenutni verziji lahko natančno pogledamo, kako se je pandemija Covid-19 odvijala. Celotna koda aplikacije je dostopna na naslovu <https://github.com/komellara99/covidapp/tree/main> in aplikacija na naslovu <https://f7o58x-komellara99.shinyapps.io/covid19app/>.

7 LITERATURA IN VIRI

- [1] H. WICKHAM, „Mastering Shiny,“ [Elektronski]. Available: <https://mastering-shiny.org/>.
- [2] R. S. Inc., „How to start Shiny,“ [Elektronski]. Available: <https://shiny.rstudio.com/tutorial/>.
- [3] D. Granjon, „Outstanding User Interfaces with Shiny,“ [Elektronski]. Available: <https://unleash-shiny-interface.com>.
- [4] G. H. Wickham, R for Data Science, 2016.
- [5] WebMD, „Coronavirus and COVID-19: What you should know,“ [Elektronski]. Available: <https://www.hopkinsmedicine.org/health/conditions-and-diseases/coronavirus>. [Poskus dostopa februar 2023].
- [6] Arnes.si, „Obdelava in prikazovanje podatkov,“ [Elektronski]. Available: <http://daljavaosmesto.splet.arnes.si/files/2020/03/Obdelava-in-prikazovanje-podatkov.pdf>. [Poskus dostopa februar 2023].
- [7] D. Smith, R Tops Data Mining Poll, 2012.
- [8] „Wikipedija, R (programming language),“ [Elektronski]. Available: [https://en.wikipedia.org/wiki/R_\(programming_language\)#cite_note-7](https://en.wikipedia.org/wiki/R_(programming_language)#cite_note-7). [Poskus dostopa februar 2023].
- [9] IUS_INFO, „Kdaj epidemija postane pandemija,“ 2020. [Elektronski]. Available: <https://www.iusinfo.si/medijsko-sredisce/v-srediscu/259119>. [Poskus dostopa februar 2023].
- [10] G. G. H. Wickham, „R for Data Science,“ 2023. [Elektronski]. Available: <https://r4ds.had.co.nz/>. [Poskus dostopa februar 2023].
- [11] D. Y. S. J. Weston, „Why You Should Become a User: A Brief Introduction to R,“ 2017. [Elektronski]. Available: <https://www.psychologicalscience.org/observer/why-you-should-become-a-user-a-brief-introduction-to-r>. [Poskus dostopa februar 2023].
- [12] University Kent State, „Statistical & Qualitative Data Analysis Software: About R And RStudio,“ [Elektronski]. Available: <https://libguides.library.kent.edu/statconsulting/r>. [Poskus dostopa februar 2023].
- [13] „Wikipedija, RStudio,“ [Elektronski]. Available: <https://en.wikipedia.org/wiki/RStudio>. [Poskus dostopa februar 2023].
- [14] Domino, „Shiny (in R),“ [Elektronski]. Available: <https://www.dominodatalab.com/data-science-dictionary/shiny-in-r>.
- [15] Hadrien, „How to Build a Shiny Application from Scratch,“ [Elektronski]. Available: https://bookdown.org/hadrien/how_to_build_a_shiny_app_from_scratch/#components-of-a-shiny-application. [Poskus dostopa februar 2023].

-
- [16] Github, „Covid-19 Data,“ [Elektronski]. Available: <https://github.com/owid/covid-19-data/tree/master/public/data>. [Poskus dostopa februar 2023].
- [17] „HTML table basics,“ [Elektronski]. Available: <https://developer.mozilla.org/en-US/docs/Learn/HTML/Tables/Basics>. [Poskus dostopa februar 2023].
- [18] Medium, „What is Histogram?,“ [Elektronski]. Available: <https://medium.com/wicds/what-is-histogram-d9e1fe230ae7>. [Poskus dostopa februar 2023].
- [19] „SCSS CHLS,“ [Elektronski]. Available: <https://www.apti4all.com/ssc/previous-year-papers/chsl/ssc-chsl-10th-june-shift-3-qa>. [Poskus dostopa februar 2023].
- [20] „Country Profiles,“ [Elektronski]. Available: https://www3.paho.org/english/dd/ais/be_v25n2-perfil-guatemala.htm. [Poskus dostopa februar 2023].
- [21] „VizEdit Line Chart,“ [Elektronski]. Available: <https://pbivizedit.com/gallery/smooth-lines-in-line-chart>. [Poskus dostopa februar 2023].
- [22] Matplotlib, „Horizontal bar chart,“ [Elektronski]. Available: https://matplotlib.org/3.2.2/gallery/lines_bars_and_markers/barh.html. [Poskus dostopa februar 2023].

