R Shiny aplikacija za praćenje podataka o pandemiji i procjenu efektivnog reprodukcijskog broja

Šuvak, Katarina

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike i računarstva

Katarina Šuvak

R Shiny aplikacija za praćenje podataka o pandemiji i procjenu efektivnog reprodukcijskog broja

Završni rad

Osijek, 2021.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni preddiplomski studij matematike i računarstva

Katarina Šuvak

R Shiny aplikacija za praćenje podataka o pandemiji i procjenu efektivnog reprodukcijskog broja

Završni rad

Mentor: doc. dr. sc. Danijel Grahovac

Osijek, 2021.

Sažetak

U ovom radu objasnit će se Shiny paket jezika R koji služi za izradu interaktivnih web aplikacija, kako izgleda struktura takve aplikacije, što znači njena reaktivnost i kako se ona postiže. Bit će predstavljen primjer izrade Shiny aplikacije koja prati i prikazuje podatke vezane uz COVID-19 pandemiju korištenjem javno dosupnih izvora. Na primjer, pomoću nje može se pratit broj zaraženih, umrlih, oporavljenih osoba, napredak cijepljenja u raznim državama i u Hrvatskoj ili doznati ponešto o bolesti. Također, u radu će se detaljnije objasniti efektivni reprodukcijski broj kojeg aplikacija procjenjuje.

Ključne riječi

Shiny, reaktivnost, COVID-19, efektivni reprodukcijski broj

R shiny application to track pandemic data and estimate the effective reproduction number

Summary

This paper explains the Shiny package of R language that is used to create interactive web applications, how does the structure of Shiny application look like, what its reactivity means and how it is achieved. An example of a Shiny application that monitors and displays data related to the COVID-19 pandemic using publicly available sources will be presented. For example, it can be used to track the number of infected, deceased, recovered people, the progress of vaccination in various countries and in Croatia or to learn something about the disease. Also, the paper will explain in more detail the effective reproduction number that the application estimates.

Key words

Shiny, reactivity, COVID-19, effective reproduction number

Sadržaj

Uvod

Pandemija COVID-19 bolesti trenutno je razvijena u cijelom svijetu. Započela je u prosincu 2019. u Kini i od tada se širi. Utjecala je na brojne dijelove ljudskog života, od gospodarstva i politike, do obrazovanja i međuljudskih kontakta. Procjena smrtnosti u svijetu iznosi 0.24 % , a ona raste kako raste i dob zaraženih. Simptomi COVID-19 su različiti, ali često uključuju groznicu, kašalj, glavobolju, umor, poteškoće s disanjem i gubitak mirisa i okusa, no neki ljudi ih ni ne razviju. Detaljnije o bolesti može se vidjeti na [13] Mnogi ljudi su zabrinuti, osjećaju se uplašeno zbog situacije te stoga žele educirati se ili samo pratiti razvoj pandemije. Napravljena je aplikacija koja prati najnovije, javno dostupne podatke kroz vrijeme, grafički i tablično ih prikazuje, pomaže shvatiti posljedice bolesti, kako se ona može zaustaviti i koji čimbenici utječu na to. Služi i za procjenu efektivnog reprodukcijskog broja koji služi kao mjera zaraznosti bolesti. Aplikacija je izrađena korištenjem R jezika i njegova paketa Shiny.

R je jedan od najkorištenijih jezika u podatkovnoj znanosti, većinom se koristi pri statističkim analizama podataka, besplatan je i lako dostupan. Shiny je paket koji služi za izgradnju web aplikacija. Često se uspoređuje s Pythonovim paketom Dash. Kako bi aplikacije koje prikazuju podatke bile poželjne korisnicima, trebaju biti interaktivne i mijenjati se zajedno s podacima što Shiny upravo omogućuje. Za korištenje je jednostavan i intuitivan, nije potrebno dodatno predznanje HTMLa, CSSa ili JavaScripta, ali se ti alati mogu kombinirati s R kodom kako bi nastao ljepši ili traženi dizajn. Pomoću ovog paketa obrada velikog broja podataka o pandemiji je pojednostavljena i korisnicima su informacije i podaci ugodno i jednostavno prikazani.

1 R Shiny [9]

Shiny olakšava programerima da svoj rad nad bazama podataka ili analizu velikog broja informacija prikažu ljudima na što jednostavniji način putem izrade web aplikacije. Na taj način podaci su posloženi i uredno prikazani te su omogućeni velikom broju ljudi. Također, Shiny omogućuje reaktivnost podataka, tj. omogućuje korisnicima odabir ulaznih parametara pomoću izbornika kao što su klizači, polja za unos teksta, potvrdni okviri i sl. kako bi se podaci prikazali na određeni način, recimo stvaranjem tablica, grafova, histograma. Paket Shiny najprije je potrebno instalirati i učitati u R aplikaciju kako bi se mogao koristiti. Shiny aplikacija pokreće se ili prečacima na tipkovnici(Ctrl + Shift + Return) ili pritiskanjem RunApp gumba u RStudiu.

Primjer 1: Linija za učitavanje Shinyja

```
install.packages("shiny")
library(shiny)
```

1.1 Arhitektura Shiny aplikacije

Shiny aplikacije nalaze se uvijek u R skripti. Svaka Shiny aplikacija sastoji se od tri komponente: objekt korisničkog sučelja-UI(user interface), funkcija poslužitelj(server) i poziv funkciji ShinyApp koja stvara Shiny objekt. Tako je kod podijeljen na sučelje i podršku. Izgradnja Shiny aplikacije može se naći i na [12].

Primjer 2: Osnovna struktura aplikacije

```
library(shiny)
#Define UI
ui<-fluidPage(
)

#Define server logic
server<-function(input, output){
}

#Run the app
shinyApp(ui=ui,server=server)
```

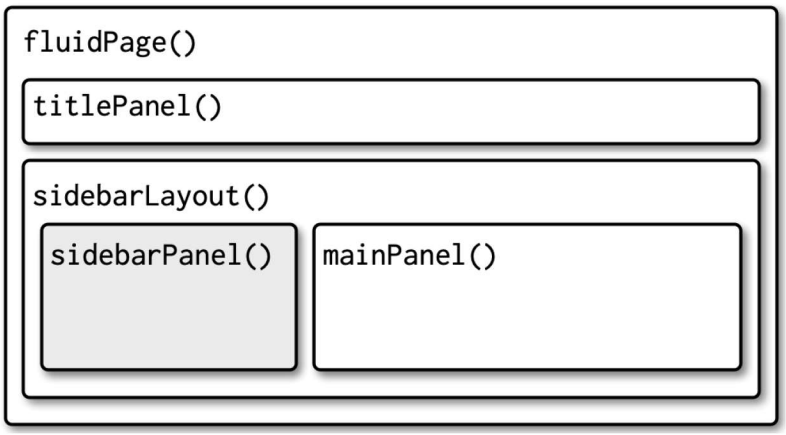
1.1.1 UI

Ui je funkcija zadužena za sučelje aplikacije. U njoj se pozivaju druge ugrađene Shiny funkcije ili funkcije definirane od strane korisnika te se tako slažu svi potrebni dijelovi aplikacije bez njihova definiranja i uređuje se njihov izgled. Prema zadanim postavkama Shiny koristi Bootstrap, najpopularniji HTML, CSS i JS okvir za razvoj responzivnih i mobilnih aplikacija na internetu. Shiny sam generira R kod u HTML i koristi ga za izgradnju web stranice. Može učitavati i Bootstrap teme koristeći bslib paket.

Raspored elemenata sučelja stvara se pozivima funkcijama među kojima postoji hijerarhijski raspored. Hijerarhija u Ru podudara se s hijerarhijom u generiranom HTML -u. To pomaže razumjeti kod sučelja. Najvažnija funkcija za izgled je fluidPage() koja stvara responzivnu stranicu. Ako želimo stranicu s navigacijskom trakom koristit ćemo navbarPage(). U nju slažemo druge funkcije, također onako kako su hijerarhijski poslagane, ako su jednake važnosti slažu se jedne unutar drugih kako se vidi u idućem primjeru.

Primjer 3: Primjer slaganja aplikacije

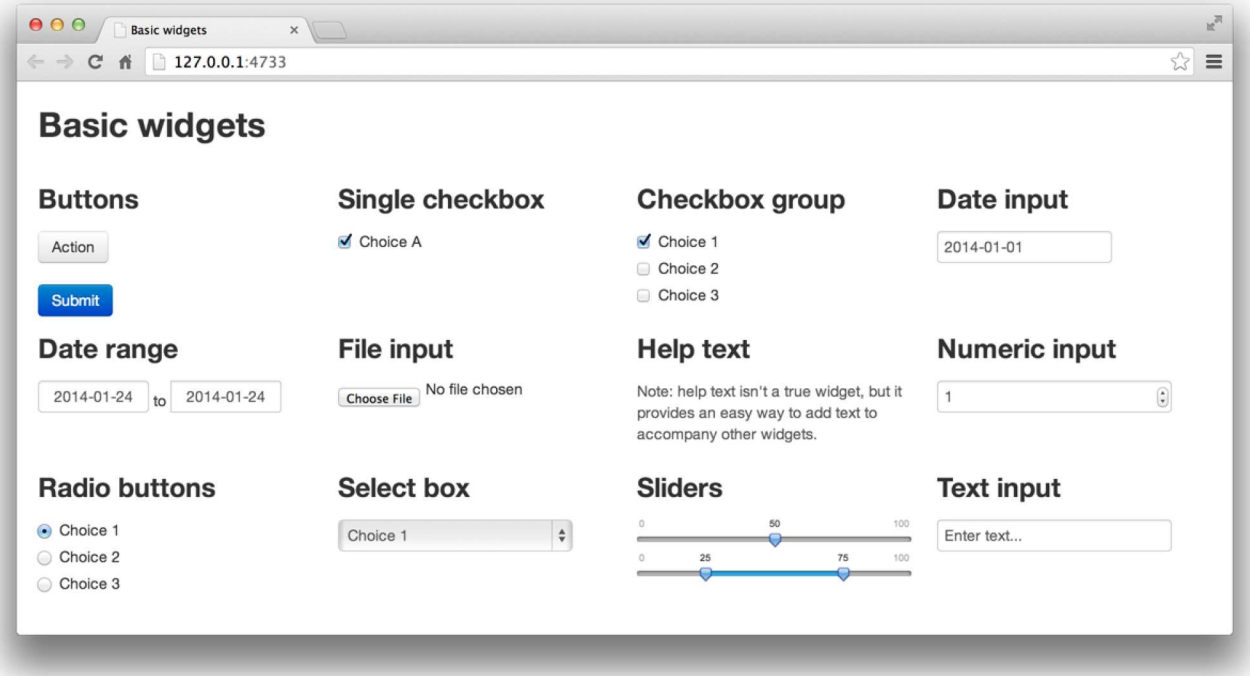
```
ui<-fluidPage(
  titlePanel(),
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```



Slika 1: Raspored funkcija dizajna

Dalje možemo pozivati fluidRow() i fluidColumn() za stvaranje retka, odnosno stupca. Isto tako, Shiny sadrži elemente koji se zovu HTML oznake. One su u principu manje cjeline poput teksutalnog odlomka, naslova, slike, hiperveze i dr. Nižu se jedna za drugom i zapravo su na dnu hijerarhije. Također, čisti HTML jezik može se pisati pozivanjem funkcije HTML().

Jedan od najbitnijih dijelova aplikacije su widgeti pomoću kojih korisnici upravljaju aplikacijom, služe za unos i upravljanje podacima, postavljanje njihove vrijednosti ili njihovo odabiranje i upravljaju njihovim prikazom. To bi, na primjer, bili gumbi, potvrdni okviri, klizači, prozori za unos datuma, teksta. Kao i HTML oznake, slažu se u određene funkcije za dizajn. Možemo reći da preko njih korisnik komunicira s aplikacijom.



Slika 2: Raspoloživi widgeti

Unutar funkcije UI potrebno je i odrediti gdje će se nalaziti njihovi izlazi. Oni se također grade ugrađenim R funkcijama čija imena započinju tipom izlaznog podatka, a završavaju s Output(). Izlazni se podaci definiraju u serveru.

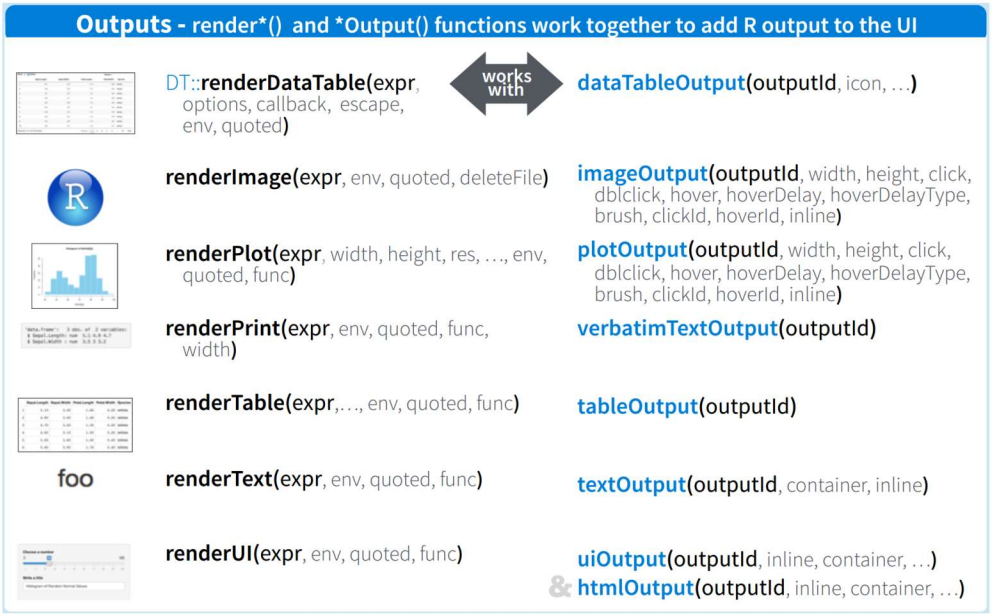
Odabirom već postojećih dijelova stranice i njihovim slaganjem dobiva se jednostavnost izgradnje aplikacije i njena preglednost.

Primjer 4: Primjer koda jednostavnog ui

```
library(shiny)
ui<-fluidPage(
  titlePanel("Hello Shiny!"),
  sidebarLayout(
    sidebarPanel(
      numericInput("numInput", "A numeric input:", value = 7, min = 1, max = 30)
    ),
    mainPanel(
      p(strong("bold font "), em("italic font")),
      a(href="http://www.google.com", "link to Google")),
      textOutput("text"),
  )
  fluidRow(
    column(6, plotlyOutput("plot1") ),
    column(6, plotlyOutput("plot2") )
  )

)
```

1.1.2 Server

Sada nakon što je struktura aplikacije izgrađena u UI, treba postaviti logiku aplikacije, odnosno treba odrediti izlazne podatke i njihov prikaz koji se mogu mijenjati kako korisnik mijenja postavke widgeta ako su o njima ovisni. Njihovo mjesto u aplikaciji je već određeno, jedino je potrebno osigurati da prikazuju podatke na ispravan način funkcijama zaslužnim za to. Te funkcije započinju s render i završavaju s tipom izlaznog podatka, slično kao u ui. Ulazne podatke u serveru pozivamo naredbom input$id gdje je id jedinstvena oznaka ulaza. Oni se mogu koristiti bilo gdje unutar servera prilikom stvaranja bilo kojeg izlaza. Na primjer, želimo postaviti funkciju za računanje površine kruga. U ui postavimo widget za postavljanje numeričkog ulaza u jednu od gradivnih jedinica stranice, npr. SidebaPanel() kako bismo mogli upisati radijus kruga i odredimo da će se tekstualni izlaz nalaziti u mainPanel(). U serveru definiramo kako će se računati površina kruga sa zadanim vrijednostima. Za kraj moramo još pozvati funkciju za stvaranje Shiny objekta.



Slika 3: Imena funkcija za server (ulaze) i ui (izlaze)

Primjer 5: Povezanost ui i servera

```
ui <- fluidPage(
  dataTableOutput("table")
)
server <- function(input, output) {
  output$table <- renderDataTable(mtcars, options = list(pageLength = 5))
}
```

1.2 Reaktivnost

Ideja reaktivnog programiranja je stvaranje ovisnosti među elementima aplikacije na način da se pri promjeni ulaza svi povezani izlazi automatski ažuriraju. U Shiny aplikacijama ulazi se mogu mijenjati tijekom vremena ili ih može mijenjati korisnik, npr. povlačenjem klizača, upisivanjem u tekstualne okvire, označavanjem i dr. što dovodi do pokretanja logike na poslužitelju (čitanje CSV-ova, učenje modela, mijenjanje podataka, ponovno provođenje upita nad bazama, ...) što u konačnici rezultira mijenjanjem izlaznih podataka-tablica, grafova i teksta koji ih prikazuju. Tako aplikacije postaju interaktivne. Ovo se dosta razlikuje od većine R koda, koji se obično bavi prilično statičkim podacima.
Da bi Shiny aplikacije bile optimalnije, potrebni su nam reaktivni izrazi i određeni izlazi se trebaju ažurirati samo ako se njihovi ulazi promijene. Želimo da rezultati budu usklađeni s ulazima, a da pritom osiguramo da nikada ne obavimo više posla nego što je potrebno.
U Shiny aplikaciji stvaramo reaktivnu vrijednost pomoću reactiveVal(). Vrijednost takve varijable mijenja se kako se mijenjanju i varijable o kojima je ovisna. Nju može mijenjati korisnik ili se može mijenjati u nekoj izvornoj bazi koju aplikacija učitava. U idućem primjeru vrijednost varijable temp_f mijenja se onda kada se promijeni vrijednost varijable temp_c().

Primjer 6: Primjer reaktivne varijable

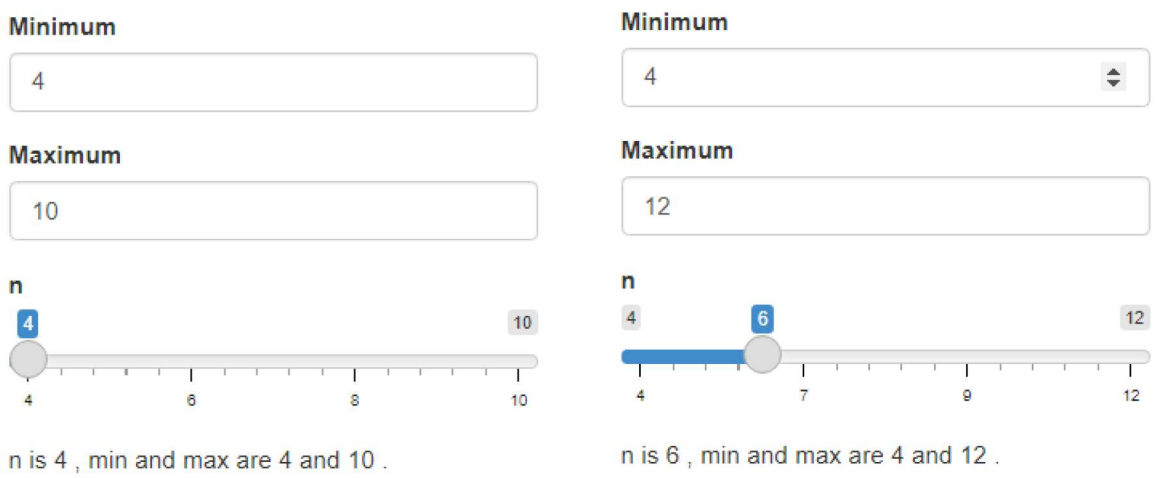
```
temp_c <- reactiveVal(10) # create
temp_c()                  # get
#> [1] 10
temp_c(20)                # set
temp_c()                  # get
#> [1] 20
temp_f <- reactive({
  message("Converting")
  (temp_c() * 9 / 5) + 32
})
temp_f()
#> Converting
#> [1] 68
temp_c(-3)
temp_c(-10)
temp_f()
#> Converting
#> [1] 14
```

Kod koji se pokreće svaki put kad se ažurira jedna od reaktivnih vrijednosti nalazi se unutar naredbe observe(). Ona je poput reaktivnog izraza po tome što može čitati reaktivne vrijednosti i pozivati reaktivne izraze te će se automatski ponovno izvršiti kad se te ovisnosti promijene. No, za razliku od reaktivnih izraza, ne daje rezultat i ne može se koristiti kao ulaz za druge reaktivne izraze. Ponekad se kod ne treba izvršiti čim se namjesti nova ulazna vrijednost, već treba čekati određenu korisnikovu radnju poput klikanja gumba. Reaktivna radnja koja se izvršava nakon takve radnje naziva se događaj, a nalazi se unutar funkcije

observeEvent(). Zajedno s observe() može se kombinirati i funkcija update() koja omogućava ažuriranje ulaza nakon što je prvobitno unesen.

Primjer 7: Primjer koda s reaktivnim varijablama

```
ui <- fluidPage(
  numericInput("min", "Minimum", 0),
  numericInput("max", "Maximum", 3),
  sliderInput("n", "n", min = 0, max = 3, value = 1),
  output$min<-renderText({
     input$n
  })
)
server <- function(input, output, session) {
  observeEvent(input$min, {
    updateSliderInput(inputId = "n", min = input$min)
  })
  observeEvent(input$max, {
    updateSliderInput(inputId = "n", max = input$max)
  })
  Output$min<-renderText({
      paste("n is", input$n, ", min and max are ", input$min, "and", input$max)
  })
}
```



Slika 4: Izlazni podaci prethodnog koda nakon promjena

2 Aplikacija za praćenje podataka o pandemiji

2.1 Općenito

Predstavljena je Shiny aplikacija o COVID-19 pandemiji. Ona prati podatke vezane uz bolest kao što su broj zaraženih, umrlih, izliječenih, cijepljenih osoba u svijetu i Hrvatskoj; prikazuje ih grafički i tablično, približava ljudima neke informacije o bolesti i računa procjenu efektivnog reprodukcijskog broja, odnosno prosječan broj ljudi u populaciji koje zaraženi pojedinac može zaraziti u razdoblju u kojem je zarazan. Svi korišteni podaci su javno dostupni, vremenski prate tok pandemije i redovno se učitavaju sa službenih stranica.
Aplikacija se može pronaći na sljedećoj poveznici:
`https://covid-app.shinyapps.io/Covid_app_shiny/`

2.2 Tehnologije

Projekt je napisan koristeći programski jezik R i sljedeće R pakete:

- shiny - izrada interaktivnih web aplikacija izravno iz R-a
- ggplot2 - vizualizacija podataka
- plotly - izrada interaktivnih grafova
- tidyr - sadrži alate za promjenu oblika i hijerarhije skupa podataka
- dplyr - pruža skup alata za učinkovito upravljanje skupovima podataka u R
- shinyWidgets - nudi prilagođene widgete i druge komponente
- EpiEstim - procjena efektivnog reprodukcijskog broja
- RColorBrewer - nudi palete boja za karte
- rgdal - rad s geoprostornim podacima
- leaflet - izrada interaktivnih karti
- jsonlite - JSON parser

2.3 Podaci

Kao što je već spomenuto, svi podaci su dostupni na stranicama koje se bave COVID-19 zarazom, poput [10]. Podaci se nalaze u nekoliko različitih baza koje se svakodnevno ažuriraju. Neki od podataka su zapisani u CSV, a neki u JSON formatu. Baze podataka učitavaju se pozivanjem naredbe u R-u i spremaju se njihove vrijednosti s kojima se dalje može raditi.
Prije korištenja podataka morali smo napraviti neke izmjene. Kod baza za podatke o broju zaraženih, umrlih i oporavljenih u svijetu i Hrvatskoj bilo je potrebno promijeniti nazive

stupaca i format datuma. Također, iz učitanih podataka načinili smo nove podatke poput novih dnevnih slučajeva na temelju stupca koji prikazuje ukupan broj slučajeva od početka pandemije do toga datuma. Na sličan način smo načinili podatke o broju dnevno umrlih i oporavljenih. Izvan aplikacije server su kreirane tablice i varijable potrebne za općeniti pregled podataka koje se mijenjaju samo ako se mijenja baza, dakle ne trebaju korisničku interakciju.

```
#stupci su pretvoreni u retke po datumima
data_confirmed_sub <- data_confirmed %>%
 pivot_longer(names_to = "date",values_to = "count", cols = 5:ncol(data_confirmed)) %>%
 group_by( `Country.Region`, date)%>%
 summarise("Total" = sum(count, na.rm = T))

#mijenjanje formata datuma
data_confirmed_sub$date <- data_confirmed_sub$date %>% sub("X", "", .) %>% as.Date("%m.%d.%y")

data_confirmed_sub<-data_confirmed_sub %>% arrange(Country.Region,date)

#novi stupac koji sadrži dnevni broj slučaja
data_confirmed_sub$diff <- ave(data_deceased_sub$Total, data_deceased_sub$`Country.Region`,
                               FUN=function(x) c(data_deceased_sub$Total[1], diff(x)))
```

Slika 5: Izlazni podaci prethodnog koda nakon promjena

	Country.Region	date	Total	diff
603	Afghanistan	2021-09-15	154283	103
604	Afghanistan	2021-09-16	154361	78
605	Afghanistan	2021-09-17	154487	126
606	Afghanistan	2021-09-18	154487	0
607	Afghanistan	2021-09-19	154487	0
608	Afghanistan	2021-09-20	154585	98
609	Afghanistan	2021-09-21	154712	127
610	Albania	2020-01-22	0	0
611	Albania	2020-01-23	0	0
612	Albania	2020-01-24	0	0
613	Albania	2020-01-25	0	0
614	Albania	2020-01-26	0	0
615	Albania	2020-01-27	0	0
616	Albania	2020-01-28	0	0

Slika 6: Prikaz dijela jedne od baza

2.4 UI

Unutar funkcije ui dizajniran je izgled aplikacije i mogu se naći HTML i CSS naredbe za ljepši prikaz podataka. Cijela aplikacija je u UI navedena kao navbarPage, odnosno stranica s navigacijskom trakom. Ona se sastoji od nekoliko kartica koje su kreirane kao TabPanel. Svaka kartica ima svoju temu i prikazuje različite podatke o bolesti. Kao i svaka Shiny aplikacija sastavljena je od kombinacija funkcija za izgled, HTML oznaka, widgeta i u njoj je određeno gdje će se nalaziti izlazi.

Slika 7: Početna stranica

2.5 Server

U funkciji server je svaki izlaz definiran i oblikovan. Iz već uređenih baza i spremljenih podataka nastaju svi grafički, tekstualni i tablični prikazi. Podaci su vizualizirani u obliku stupčastih dijagrama i vremenskih krivulja te posebno je umetnut graf funkcije za procjenu efektivnog reprodukcijskog broja i karta Hrvatske za praćenje broja zaraženih po županiji. Za crtanje svih grafova korištene su iduće funkcije:

- *ggplot()* koja inicijalizira ggplot objekt
- *ggplotly()* koja čini graf interaktivnim
- *layout()* koja služi za mijenjanje izgleda grafa, odnosno dodavanje legende i iskočnih prozorčića pri prelaskom miša preko grafa
- *labs()* koja dodaje oznake na grafu poput naslova, natpisa na osima,...
- *config()* koja modificira konfiguraciju grafa

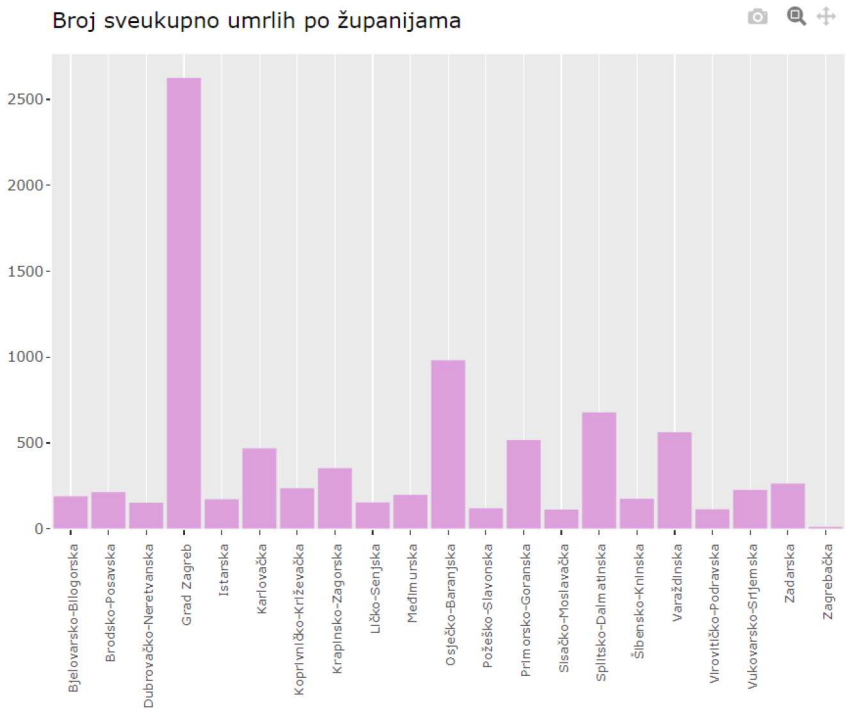
Svaka od ovih funkcija može se spojiti s dodatnim funkcijama za prikazivanje podataka na određen način. Tako su nastali stupčasti dijagrami funkcijama:

- *geom_ bar()* koji iscrtava stupce
- *coord_ flip()* koja zamjenjuje osi grafa
- *coord_ flip()* za vodoravan položaj dijagrama
- *theme()* koja modificira temu grafa, mijenja poziciju legende i naslova...

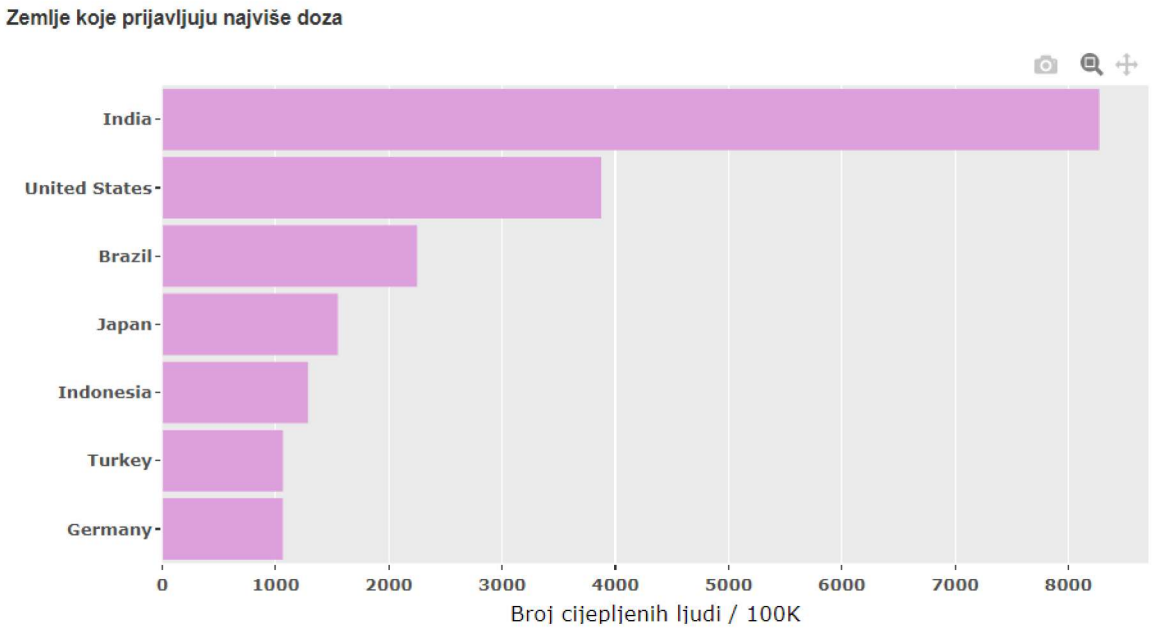
što možemo vidjeti u idućem primjeru:

```
output$County_deaths <- renderPlotly({
    zupanija<-tapply(data_hrv_zupanije_umrli$Total,data_hrv_zupanije_umrli$County,max)
    zupanije<-data.frame(zupanija)

    cjep1<-ggplot(zupanije,aes(x = rownames(zupanije), y=zupanije$zupanija)) +
        geom_bar(stat="identity", fill='#DDA0DD') +
        theme(
            panel.grid.minor.y = element_blank(),
            panel.grid.major.y = element_blank(),
            axis.text.x=element_text(size=8, angle=90,hjust=0.95,vjust=0.2),
            legend.position="none",
            plot.caption = element_text(hjust = 0, face= "italic"),
            plot.title.position = "plot"
        )+
        labs( y = "",x="",title = "Broj sveukupno umrlih po županijama")

    gr3<-ggplotly(cjep1,dynamicTicks=TRUE) %>%
      layout(hovermode="x unified")%>%
      config(displayModeBar = "static", displaylogo = FALSE,
             modeBarButtonsToRemove = list("sendDataToCloud", "autoScale2d", "resetScale2d",
                                           "hoverClosestCartesian", "hoverCompareCartesian",
                                           "select2d", "lasso2d", "zoomIn2d", "zoomOut2d",
                                           "toggleSpikelines"))
    gr3 %>%style(text = paste0(gr3$x$data[[1]]$x, "<br>",
                               "Preminuli: ", gr3$x$data[[1]]$y, "  "))

})
```

Slika 8: Primjer koda stupčastog dijagrama



Slika 9: Pripadni stupčasti dijagram



Slika 10: Stupčasti dijagram sa zamijenjenim osima

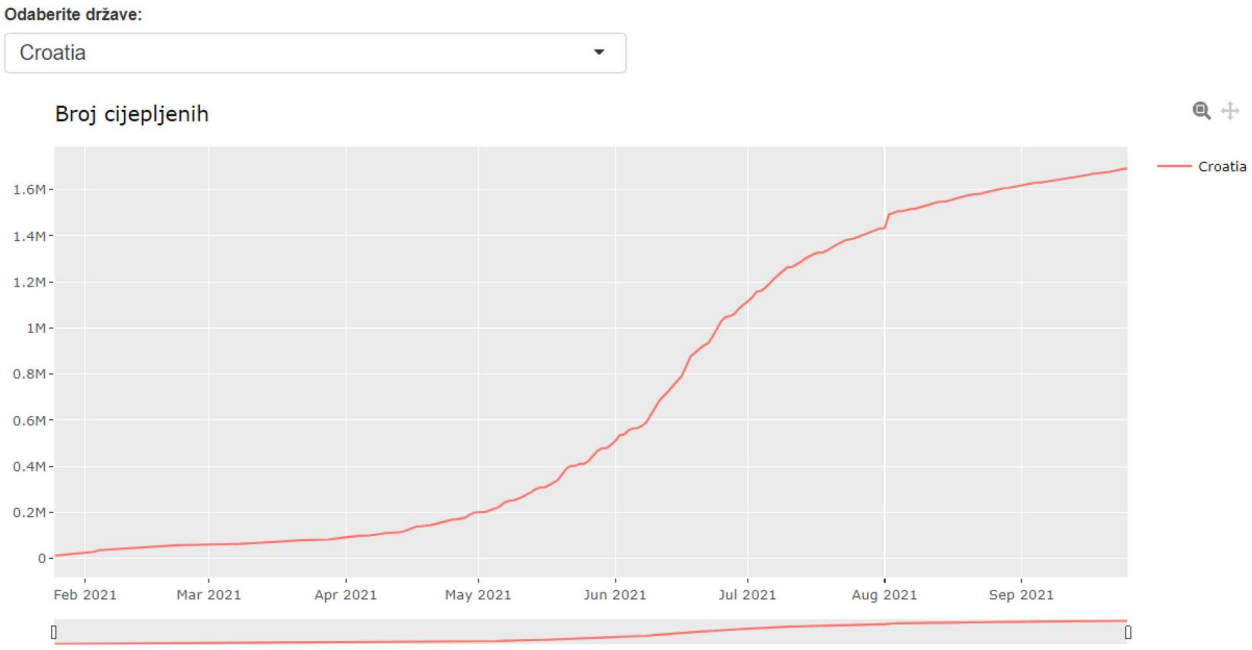
epidemijske krivulje:

- *geom_line()* koja iscrtava krivulju
- *scale_x_date()* koja omogućuje skaliranje x-osi te mijenjanje prikaza datuma
- *scale_y_continous()* koja skalira y-os, npr. log10 skaliranje

i to vidimo na idućem primjeru:

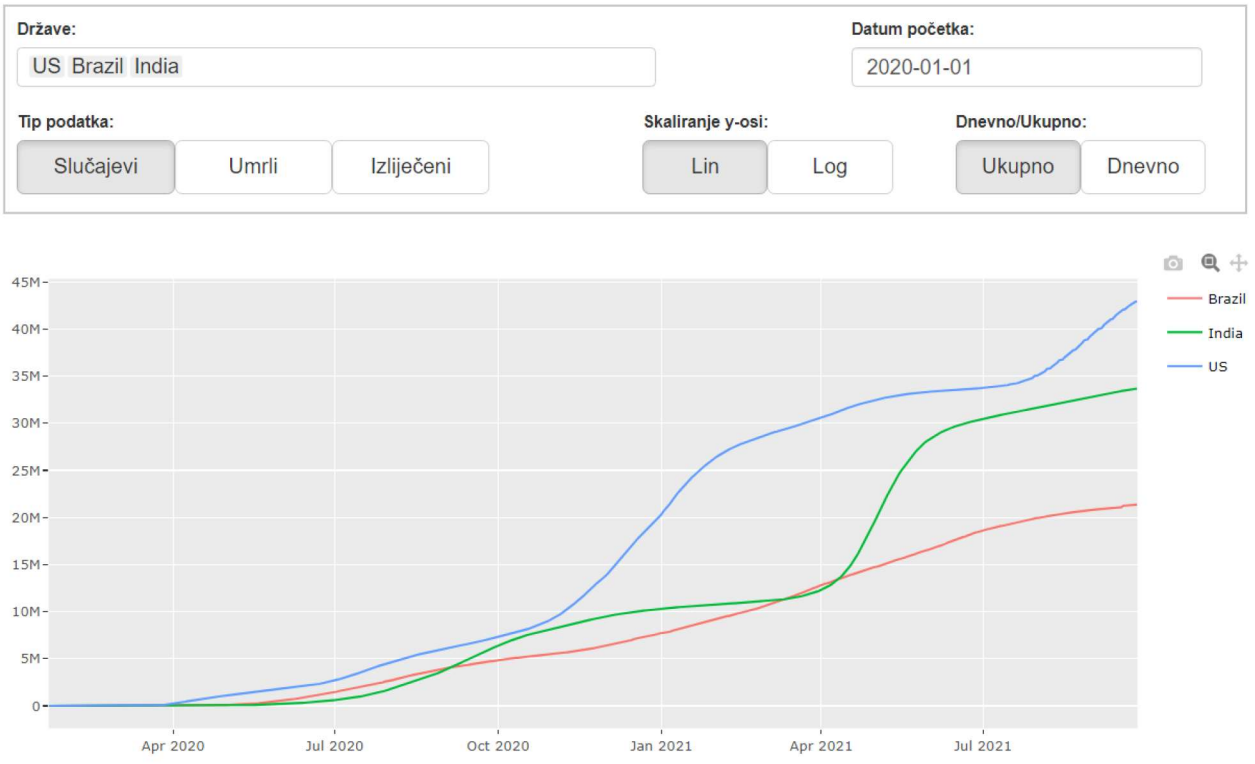
```
output$Vaccination_plot <- renderPlotly({
    cijepljenje <- Data[location==input$location & !is.na(people_fully_vaccinated),]

    cijep_graf <- ggplot(cijepljenje, aes(x=Datum, y=Cijepljeni,text = paste(location,'</br>'))) +
        geom_line(aes(group=location,color=location)) +
        labs(x="",y="",title="Broj cijepljenih ")+
        scale_y_continuous(labels = addUnits)+
        scale_color_discrete(guide = FALSE)+
        coord_cartesian(clip = 'off')+
        theme(plot.caption = element_text(hjust = 0, face= "italic"),
              plot.title.position = "plot")
    graf_cijepljenja<-ggplotly(cijep_graf,dynamicTicks=TRUE,tooltip = c("text","Datum","Cijepljeni"))%>%
      layout(hovermode="x unified")%>%
      rangeslider(thickness=0.05) %>%
      config(displayModeBar = "static", displaylogo = FALSE,
             modeBarButtonsToRemove = list("sendDataToCloud", "toImage", "autoScale2d", "resetScale2d",
                                           "hoverClosestCartesian", "hoverCompareCartesian", "select2d", "lasso2d",
                                           "zoomIn2d", "zoomOut2d", "toggleSpikelines"))
})
```

Slika 11: Primjer koda za reaktivnu epidemijsku krivulju koja prati broj cijepljenih



Slika 12: Pripadna krivulja

Za idući graf unutar servera je korisniku omogućeno biranje više država odjednom, podataka koji ga zanimaju, radi li se to o dnevnim podacima ili sveukupnim i odabire skaliranje y-osi. Tako se lakše može uspoređivati kako se zaraza širi u svijetu i kako to izgleda kroz vrijeme.



Slika 13: Graf koji prikazuje podatke o broju slučaja, umrlih i izliječenih

Za izradu karte Hrvatske bila je potrebna .geojson datoteka koja sadrži geoprostorne podatke poput geografske širine i geografske dužine. Za rad s kartama potrebno je učitati leaflet paket. Pri izradi karte koristimo funkciju leaflet() za stvaranje karte. Nadalje, funkcija addPolygons() iscrtava poligone i ispunjava ih određenim bojama. U ovom slučaju iscrtane su županije i njihove boje mogu se birati. Obojenost pojedine županije ovisi o broju zaraženih ljudi u danu. Dana je legenda za lakšu predodžbu. Funkcijom setView() centriramo pogled karte na temelju proslijeđenih vrijednosti geografske širine i dužine te također približavamo pogled. Osim toga, funkcija clearPolygons() briše poligone. Više o paketu može se pronaći na [6].
Najprije smo morali stvoriti kartu.

```
output$map <- renderLeaflet({
    leaflet(gas)%>%
        addTiles() %>%
        setView(lat=44.915399,lng=16.466568,zoom=7)
})
```

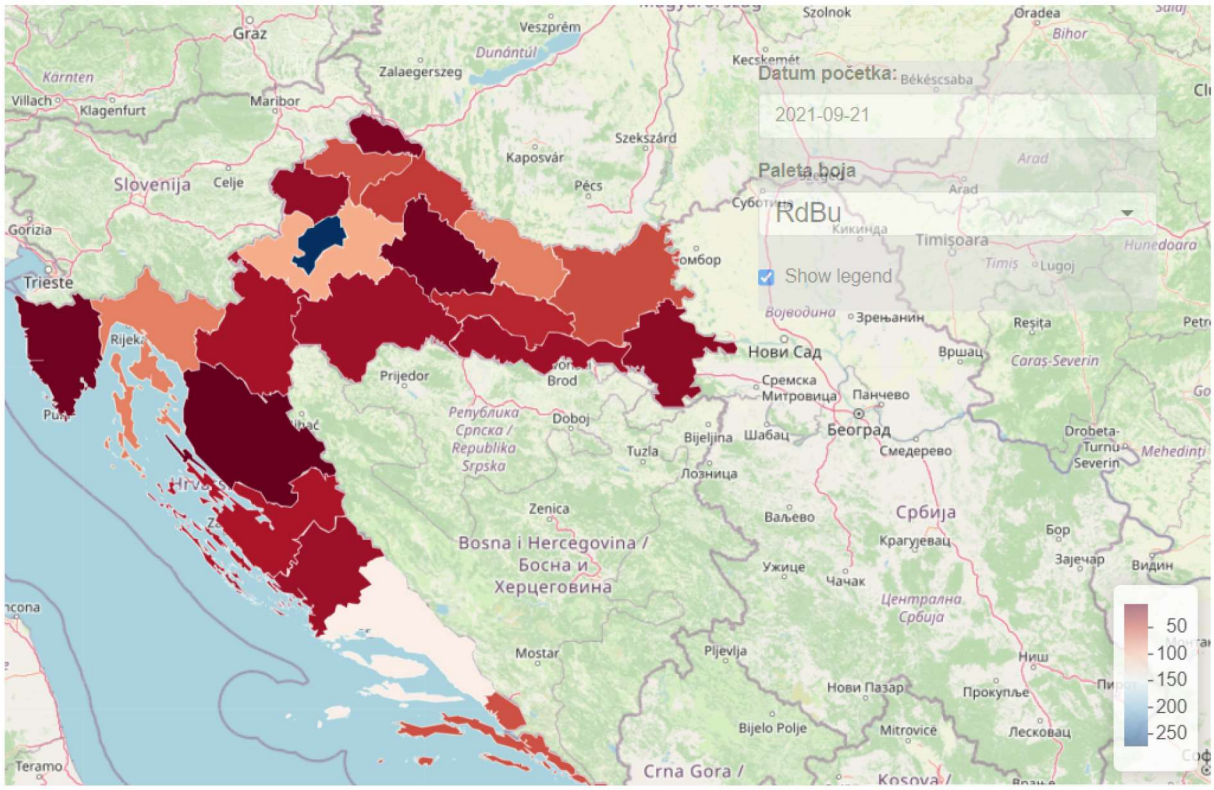
Slika 14: Stvaranje karte

Zatim su korištene su dvije funkcije observe(). Prva se pokreće svaki put kada se aplikacija učita, stvara obrise županija i boja ih, a druga kada se mijenja spektar boja i uklanja legendu ako korisnik tako odabere.

```
observe({
    req(input$tab== "Hrvatska")
    gas<-filteredData()
    pal <- colorpal()

    leafletProxy("map", data = gas) %>%
        clearShapes() %>%
        addPolygons(
            fillColor=~pal(Cases),
            stroke=TRUE,
            fillOpacity=1,
            color="white",
            weight=1,
            popup = ~paste0("SluÄŤajevi: ",Cases,"<br>",name))

})
observe({
    req(input$tab== "Hrvatska")
    proxy <- leafletProxy("map", data = gas)
    proxy %>% clearControls()
    if (input$legend) {
        gas<-filteredData()
        pal <- colorpal()
        proxy %>% addLegend(position = "bottomright",
                            pal = pal, values = gas$Cases
        )
    }

})
```

Slika 15: Karta Hrvatske sa županijama obojenih po broju zaraženih ljudi

Rezultat možemo vidjeti na sljedećoj slici:



Slika 16: Funkcije za bojanje županija na karti

2.6 Efektivni reprodukcijski broj

Zaraznost neke bolesti bitno je pratiti kako bi javno zdravstvo moglo predviđati raspon bolesti i uvesti mjere kao način zaustavljanja i sprječavanja zaraze. Jedna od mjera za njeno praćenje je efektivni reprodukcijski broj, R_e. Definiran je kao prosječni broj ljudi koje zarazi jedna zaražena osoba [7]. Praćenje R_e tijekom vremena daje informaciju o učinkovitosti mjera i trebaju li se one mijenjati. Njihova zadaća je smanjivati R_e, odnosno približavati ga nuli. Vrijednosti veće od jedan ukazuju na širenje bolesti. Mijenja se imunizacijom populacije, bilo individualnim imunitetom nakon infekcije ili cijepljenjem te smrću [2]. Kako bi mogli shvatiti kako se u aplikaciji procjenjuje R_e, moramo uvesti iduće pojmove.

Gama distribucija je generalizacija eksponencijalne distribucije. Neka je slučajni pokus ponavljanje događaja u vremenu sa zadanim konstantnim intezitetom (λ). Slučajna varijabla koja daje vrijeme potrebno da se dogadaj dogodi odredeni broj puta (α) ima gama distibuciju s parametrima α i λ. Vidi [1].

Serijski interval je vrijeme koje prođe između pojave simptoma kod primarnog slučaja zaraze i sekundarnog slučaja (osobe koju je osoba primarnog slučaja zarazila).

Pod distribucijom serijskog intervala mislimo na distribuciju iz koje dolaze vremena za sve prijenose infekcije između osoba, odnosno prate se kontakti prenositelja [3]. U aplikaciji je korištena Cori et al. metoda opisana u [4], pri čemu se pretpostavlja da serijski interval ima diskretiziranu gama distribuciju s očekivanjem 4.8, odnosno prosječni serijski interval iznosi 4.8 dana i standardnom devijacijom 2.3 prema [8]. Korištena je funkcija estimate_r iz paketa EpiEstim koji se temelji na Cori et al i Thompson et al metodi koja je opisana u

[11]. Više o funkciji i paketu može se pronaći na [5]. Vremenski interval je sedam dana. U ovom slučaju korisnik može mijenjati procjenu očekivanja i standardnu devijaciju serijskog intervala te datume početka i kraja praćenja za odabranu državu.
U sljedećem kodu objašnjeno je što se radi kako bi se iscrtao graf za procjenu.

```
#u df_selected spremamo vrijednosti koje se odnose na državu za koju nas zanima
#procjena efektivnog reprodukcijskog broja kroz vrijeme

df_selected <- data_confirmed_sub %>%filter(Country.Region==input$estimated_countries)

#izbacimo vrijednosti s nulama
new_tab<-df_selected %>%
  group_by(`Country.Region`) %>%
  filter(row_number() >= min(row_number()[diff != 0])) %>%
  ungroup()

#duljina stupca koji prikazuje dnevni broj zaraženih
T <- length(new_tab$diff)


#određivanje vremenskog intervala
t_start <- seq(2, T-7+1)
t_end <- t_start + 7-1

#ugrađena funkcija za procjenu efektivnog reprodukcijskog broja iz paketa EpiEstim
#ovdje korisnik odabire sam prosječnu vrijednost serijskog intervala i devijaciju
res_weekly <- estimate_R(new_tab$diff,
                         method="parametric_si",
                         config = make_config(list(
                           mean_si = input$NUM_mean,
                           std_si = input$NUM_dev,
                           t_start = t_start,
                           t_end = t_end
                         ))
)


#stvaramo tablicu koji u sebi sadrži datume, procjenu, 0.025 i 0.975-kvantil
#ona služi pri crtanju grafa
tablica <- data.frame("datum"=new_tab$Datum[-(1:7)],
                      "procjena"=res_weekly$R$`Mean(R)`,
                      "lb"=res_weekly$R$`Quantile.0.025(R)`,
                      "ub"=res_weekly$R$`Quantile.0.975(R)`)

#iscrtavanje
ggplot(data = tablica, aes(x = datum, y = procjena)) + geom_line(aes(color="firebrick")) +
  geom_ribbon(aes(x=datum,  ymin=lb, ymax=ub), fill="steelblue2", alpha=0.3) +
  geom_hline(yintercept=1, linetype = "dashed") +theme_minimal()+
  labs(x="Vrijeme", y="R", shape="")
```

Slika 17: Kod grafa koji prati R_e

Slika 18: Graf procjene po datumima

Literatura

Literatura

[1] Mirta Benšić i Nenad Šuvak. *Uvod u vjerojatnost i statistiku*. Sveučilište J.J. Strossmayera u Osijeku, Odjel za matematiku, 2014.

[2] Rachelle N Binny i dr. "Effect of Alert Level 4 on effective reproduction number: review of international COVID-19 cases". *medRxiv* (2020.). DOI: 10.1101/2020.04.30.20086934.

[3] Tim Churches. *Tim Churches Health Data Science Blog: Analysing COVID-19 (2019-nCoV) outbreak data with R - part 1*. 2020. URL: https://timchurches.github.io/blog/posts/2020-02-18-analysing-covid-19-2019-ncov-outbreak-data-with-r-part-1/.

[4] Anne Cori i dr. "A new framework and software to estimate time-varying reproduction numbers during epidemics". *American journal of epidemiology* 178.9 (2013.), str. 1505–1512.

[5] *Dokumentacija paketa Epiestim*. URL: https://cran.r-project.org/web/packages/EpiEstim/EpiEstim.pdf.

[6] *Dokumentacija paketa leaflet*. URL: https://rstudio.github.io/leaflet/shiny.html.

[7] *Izvješće za prethodnih 7 dana i dnevno izvješće za Republiku Hrvatsku na dan 22. ožujka 2021*. URL: https://www.koronavirus.hr/uploads/Tjedno_20izvje_C5_A1_C4_87e_20za_2022_3_pdf_2e1a98fce4.pdf.

[8] Hiroshi Nishiura, Natalie M Linton i Andrei R Akhmetzhanov. "Serial interval of novel coronavirus (COVID-19) infections". *International journal of infectious diseases* (2020.).

[9] *Shiny dokumentacija*. URL: https://shiny.rstudio.com/.

[10] *Službena stranica od WHO*. URL: https://www.who.int/.

[11] RN Thompson i dr. "Improved inference of time-varying reproduction numbers during infectious disease outbreaks". *Epidemics* 29 (2019.), str. 100356.

[12] Hadley Wickham. *Mastering shiny*. O'Reilly Media, 2020. URL: https://mastering-shiny.org/.

[13] *Wikipedia o COVID-19*. URL: https://en.wikipedia.org/wiki/COVID-19.