

SVEUČILIŠTE U SPLITU
PRIRODOSLOVNO-MATEMATIČKI FAKULTET U SPLITU

SEMINAR

Konvolucije i širenje zaraze

Matea Lukić

Mentor: *Prof. dr. sc. Ivan Slapničar*

Split, travanj 2021.

SADRŽAJ

1. Uvod	1
2. Motivacija	2
3. Konvolucije	3
3.1. Konvolucije općenito	3
3.2. Konvolucije i filtriranje	3
3.3. Konvolucije i Julia	5
4. Širenje zaraze	11
4.1. Obrada podataka i vizualizacija	11
4.1.1. Okvir podataka za COVID-19	11
4.1.2. Istraživačka analiza podataka (Exploratory data analysis)	13
4.1.3. Eksponencijalni rast broja zaraženih	13
4.2. Model širenja zaraze	15
4.3. Širenje zaraze putem binarnog stabla stabla	15
4.4. Od mikro do makro; Od diskretnog do kontinuiranog	17
4.4.1. Model oporavka	17
4.4.2. Deterministička dinamika za srednju vrijednost (mean): Intuitivno deriviranje	21
4.4.3. Deriviranje korištenjem srednje vrijednosti (mean-a) stohastičkih procesa	23
4.4.4. Neprekidno vrijeme	25
4.4.5. SIR-model	27
5. Zaključak	29
6. Literatura	30

1. Uvod

Teme ovog seminarskog rada su konvolucije i širenje zaraze. Kakvu direktnu vezu možemo uspotaviti između ta dva pojma?

Za početak moramo znati što je uopće konvolucija u matematici i računarstvu te za što ju koristimo. Matematičku definiciju operatora konvolucije ćemo ostaviti po strani te ćemo se baviti intuitivnim shvaćanjem. Ono za što će u ovoj temi konvolucije biti korisne je obrada slike. Recimo da imamo sliku te ju želimo obraditi kako bismo dobili filtriranu verziju te slike koja će nam biti pogodnija za analizu. Konvoluciju možemo gledati kao funkciju pomoću koje filtriramo.

Kada govorimo o širenju zaraze, ima smisla zapitati se o kojoj zarazi je riječ. Kako je 2021. godina, aktualna tema je virus COVID-19. Iz tog razloga će biti promatrani i analizirani podaci o širenju zaraze tzv. "Corona virusom". Dakle, podaci će biti obrađeni, vizualizirani te će biti provedene simulacije zaraze i oporavka. Širenje zaraze se može vrlo naivno predočiti stablastima grafovima i funkcijama koje opisuju zarazu. Naravno, za simulacije i oporavak je potreban neki bolji, stohastički model. Takav model će biti promatran u mikro i makro verziji te u diskretnom i u kontinuiranom vremenu.

2. Motivacija

Ostaje otvoreno pitanje : "Koja je poveznica između konvolucija i COVID-a 19 ???". Moj odgovor je da to mogu biti i dva potpuno disjunktna pojma, ali možemo govoriti o iskorištavanju konvolucija za obradu rengenskih snimaka pluća. Zamislimo da liječnici žele izoštiti, posvjetliti ili potamniti sliku kako bi ju mogli bolje analizirati. Još bolje iskorištavanje je ako hranimo konvolucijsku neuronsku mrežu snimcima pluća zaraženih, oporavljenih i onih koji nikad nisu bili u kontaktu s virusom. Tada bismo mogli istrenirati model na slikama i dobiti model koji kada mu damo novi rengenski snimak, zna je li vlasnik plućnog krila sa slike bio u kontaktu s virusom. Dakle, konvolucije se zaista primjenjuju u dijagnostici u vidu klasifikacije.

Navedena motivacija je ipak više područje umjetne inteligencije, a ono na čemu je naglasak, ali vrlo implicitno u ovom seminaru jest - upoznavanje s jezikom Julia za potrebe numeričke linearne algebre u okruženju Pluto reaktivnih bilježnica.

3. Konvolucije

3.1. Konvolucije općenito

U matematici konvolucija je matematička operacija dviju funkcija (f i g) koja daje treću funkciju ($f * g$) koja izražava kako je oblik jedne modificiran oblikom druge. Pojam konvolucija odnosi se i na funkciju rezultata i na proces izračunavanja. Definiran je kao integral umnoška dviju funkcija nakon što se jedna obrne i pomakne. Integral se procjenjuje za sve vrijednosti pomaka, stvarajući funkciju konvolucije.

3.2. Konvolucije i filtriranje

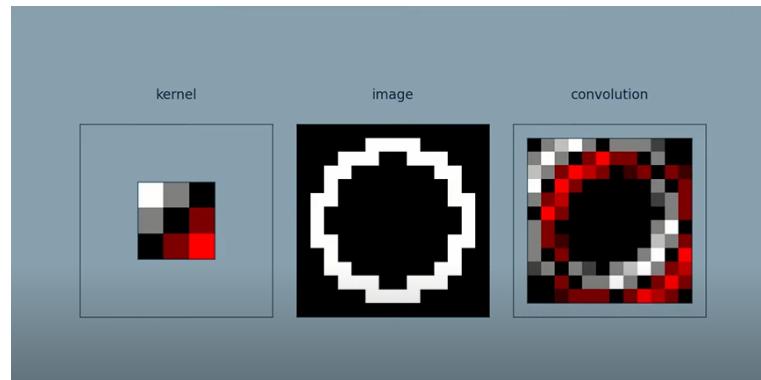


Slika 3.1

Imamo sliku Super Marija preko koje je navučena mreža i želimo ju zamagliti. Pomicemo se po slici 3×3 rešetkom koja u svakoj ćeliji ima "težinu" jednaku $1/9$. Kao rezultat dobivamo zamagljenu sliku kojoj je svaki kvadratični rezultat kretanja 3×3 rešetke po slici s lijeve strane. Rezultat se dobije tako što se uzme prosjek svih

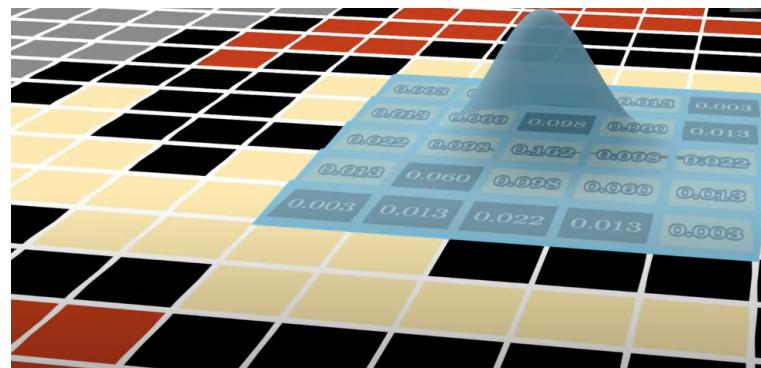
kvadratična originalne slike i doda u odgovarajući kvadratični rezultantne slike. Ako je 3×3 rešetka trenutno na dijelu originalne slike koji obuhvaća 7 sivih i 2 crne kvadratične, a težina svakog kvadratičnog je jednaka (iznosi $1/9$ jer zbroj mora biti jedan - radi se o distribuciji) to znači da će rezultat na odgovarajućem mjestu slike koja je produkt biti malo tamnije sivi kvadratični jer smo 7 sivih pomiješali s 2 crnimi. To je kao kada imamo akrilne boje i želimo naslikati kišni oblak nimbostratus među malim bijelim cirusima pa dodamo malo crne boje da bismo potamnili materijal za crtanje.

3×3 rešetka se u ovom kontekstu naziva kernel odnosno jezgra. Naravno, kernel može biti i drugih dimenzija, boja, rasporeda...

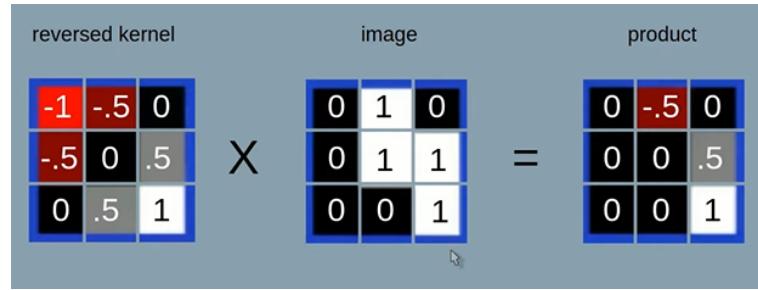


Slika 3.2: Kernel, slika, rezultat konvolucije

Ponekad nam ne odgovara da nam je svaki kvadratični kernel jendako važan. Najčešće su važniji kvadratični iz sredine od onih na krajevima. Zbog toga drugačije zadaemo težine u čelijama kernela. Ako nam je sredina najbitnija, uzimamo Gaussovnu krivulju i prema njoj distribuiramo vrijednosti čelija kernela. Tako nam najviše vrijednosti poprima centar.



Slika 3.3: Distribucija težina pomoću Gaussove krivulje



Slika 3.4: Kernel, slika, rezultat konvolucije

3.3. Konvolucije i Julia

Učitavanje demo slike u Pluto bilježnici. Julia sliku prepoznaće kao matricu RGB piksela.



```

• begin
•   maca=load("crnamaca.jpg")
• end

```

Slika 3.5

Funkcija konvolucije je implementirana u paketu ImageFiltering.jl i tamo je naravno optimizirano napisana. Nije loše promotriti ni ovaj kod metode konvolucije:

```

convolve (generic function with 2 methods)
- function convolve(M, kernel, M.index_func=clamp_at_boundary)
-   height = size(kernel, 1)
-   width = size(kernel, 2)
-   half_height = height ÷ 2
-   half_width = width ÷ 2
-   new_image = similar(M)
-
-   # (i, j) loop over the original image
-   @inbounds for i in 1:size(M, 1)
-     for j in 1:size(M, 2)
-       # (k, l) loop over the neighbouring pixels
-       new_image[i, j] = sum([
-         kernel[k, l] * M.index_func(M, i - k, j - l)
-         for k in -half_height:-half_height + height - 1
-         for l in -half_width:-half_width + width - 1
-       ])
-     end
-   end
-   return new_image
- end

```

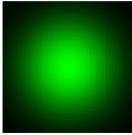
Slika 3.6: Kod metode konvolucije

Recimo da želimo zamutiti sliku. Tada trebamo kernel kojemu je težina skoncetriрана u sredini te se smanjuje prema krajevima. Dakle, uzimamo Gaussovou distribuciju za naše dvodomenzionalno polje - kernel i dobijemo jezgru kao na slici:

```

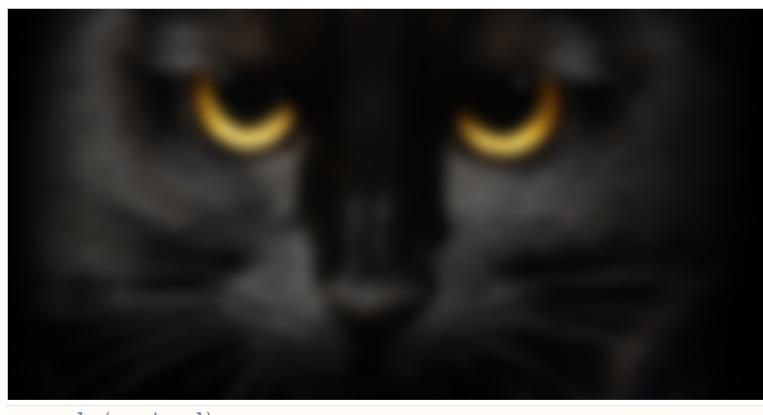
kernel =
41x41 OffsetArray(::Matrix{Float64}, -20:20, -20:20) with eltype Float64 with indices -20:
3.16486e-5 3.84629e-5 4.62792e-5 .. 4.62792e-5 3.84629e-5 3.16486e-5
3.84629e-5 4.67443e-5 5.62436e-5 5.62436e-5 4.67443e-5 3.84629e-5
4.62792e-5 5.62436e-5 6.76734e-5 6.76734e-5 5.62436e-5 4.62792e-5
5.51299e-5 6.7e-5 8.06157e-5 8.06157e-5 6.7e-5 5.51299e-5
6.50199e-5 7.90194e-5 9.50776e-5 9.50776e-5 7.90194e-5 6.50199e-5
7.5921e-5 9.22676e-5 0.000111018 .. 0.000111018 9.22676e-5 7.5921e-5
8.77677e-5 0.000106665 0.000128341 0.000128341 0.000106665 8.77677e-5
:
:
7.5921e-5 9.22676e-5 0.000111018 .. 0.000111018 9.22676e-5 7.5921e-5
6.50199e-5 7.90194e-5 9.50776e-5 9.50776e-5 7.90194e-5 6.50199e-5
5.51299e-5 6.7e-5 8.06157e-5 8.06157e-5 6.7e-5 5.51299e-5
4.62792e-5 5.62436e-5 6.76734e-5 6.76734e-5 5.62436e-5 4.62792e-5
3.84629e-5 4.67443e-5 5.62436e-5 5.62436e-5 4.67443e-5 3.84629e-5
3.16486e-5 3.84629e-5 4.62792e-5 .. 4.62792e-5 3.84629e-5 3.16486e-5
:
:
0.0017279534668674153
· kernel[0,0] # središnja vrijednost ! # ANIMACIJA SA SUPER MARIOM !!!
0.001660199442312856
· kernel[-2,-2] # ovo je prva vrijednost, dakle negativni indeksi ! !
:
:
show_colored_kernel(kernel)

```



Slika 3.7: Kernel - Gaussova distribucija težina

Kada kao argumente metode konvolucije ubacimo demo sliku i jezgru, dobivamo ovakav rezultat :



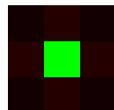
Slika 3.8: Zamućena slika

Ako bismo željeli izoštiti slike, trebamo definirati jezgru tako da iznad sporedne dijagonale budu negativne vrijednosti, a ispod pozitivne.

```

kernel2 =
3x3 OffsetArray(::Matrix{Float64}, -1:1, -1:1) with eltype Float64 with indices -1:1x-1:1:
-0.5 -1.0 -0.5
-1.0  7.0 -1.0
-0.5 -1.0 -0.5
· # idemo napraviti kernel tj. jezgru kojom ćemo izoštiti sliku
· kernel2 = centered([
·   -0.5 -1.0 -0.5
·   -1.0  7.0 -1.0
·   -0.5 -1.0 -0.5
· ])

```



```
· show_colored_kernel(kernel2)
```

```
cat =
```



Slika 3.9: Kernel za izoštravanje



```
· convolve2(cat,kernel2)
```

Slika 3.10: Izostrena slika

Slika krda zebri je uzeta kao demo u nastavku jer ima puno detalja pa se mogu uočiti razlike na grafu Fourierovog spektra za običnu sliku i zamućenog pandana te slike koji je produkt konvolucije. Imamo puno traka kako se mičemo slijeva nadesno. Ako pogledamo graf Fourierovog spektra vezanog za sliku zebri, možemo vidjeti da

kako se odmičemo od 0, frekvencija traka raste. Pogledajmo što se dešava kada uzmemo konvoluciju te slike

```

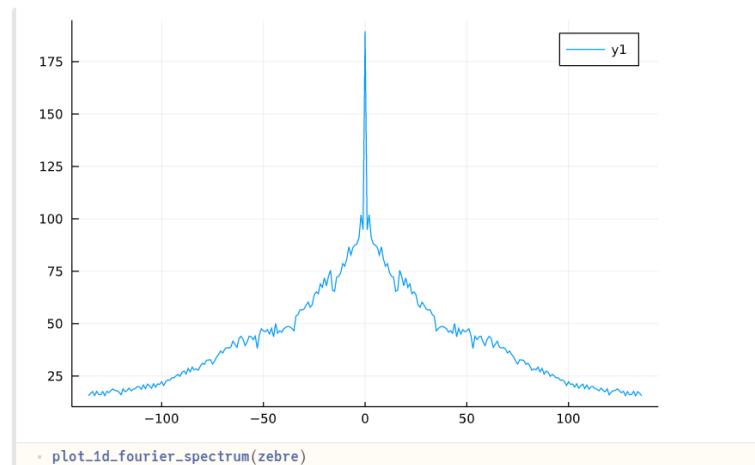
zebre =

• zebre=load("zebre.jpg")

krnl =
9x9 OffsetArray(::Matrix{Float64}, -4:4, -4:4) with eltype Float64 with indices -4:4x-4:4:
0.000763447 0.00183141 0.00342153 ... 0.00342153 0.00183141 0.000763447
0.00183141 0.00439334 0.00820783 ... 0.00820783 0.00439334 0.00183141
0.00342153 0.00820783 0.0153342 ... 0.0153342 0.00820783 0.00342153
0.0049783 0.0119423 0.0223112 ... 0.0223112 0.0119423 0.0049783
0.00564116 0.0135324 0.0252819 ... 0.0252819 0.0135324 0.00564116
0.0049783 0.0119423 0.0223112 ... 0.0223112 0.0119423 0.0049783
0.00342153 0.00820783 0.0153342 ... 0.0153342 0.00820783 0.00342153
0.00183141 0.00439334 0.00820783 ... 0.00820783 0.00439334 0.00183141
0.000763447 0.00183141 0.00342153 ... 0.00342153 0.00183141 0.000763447
• krnl = Kernel.gaussian((2,2))

```

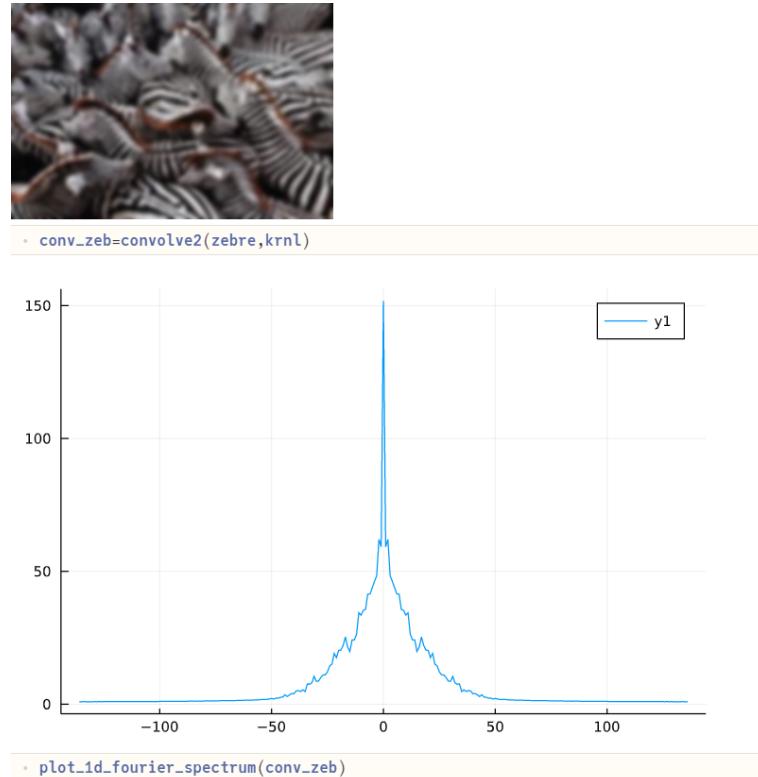
Slika 3.11: Zebre



Slika 3.12: Graf Fourierovog spektra slike krda zebri

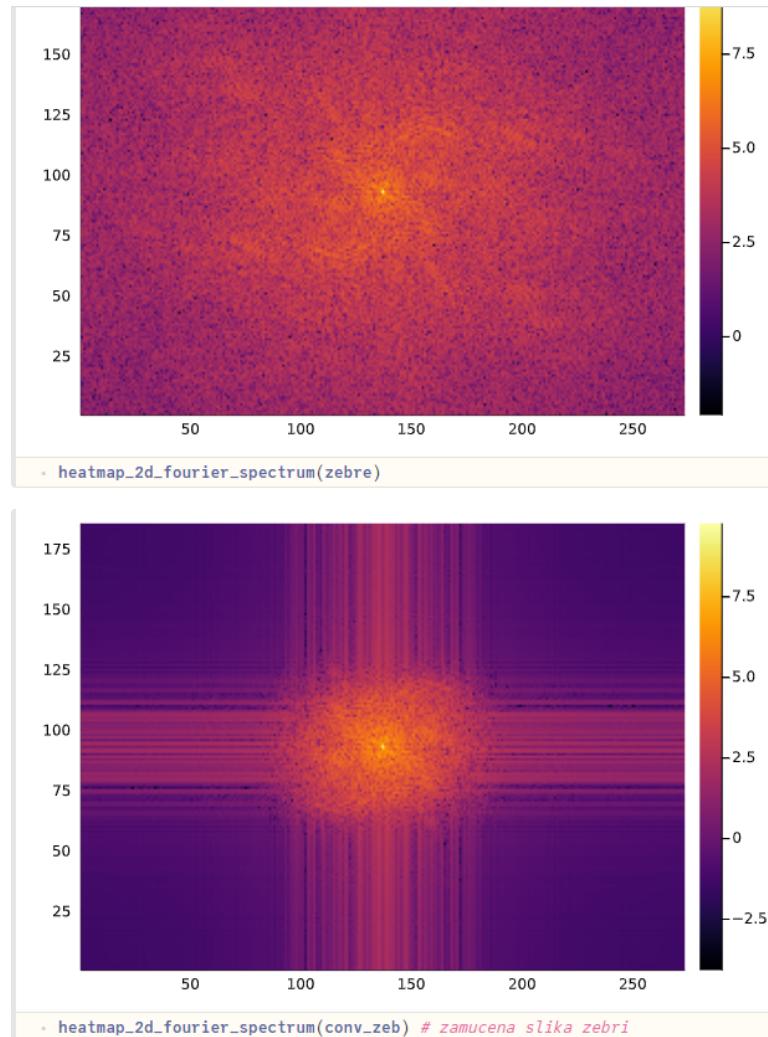
Kada pogledamo graf zamagljene slike koja je rezultat konvolucije, vidimo da dobijemo sličan oblik kao i originalne slike, ali su krajevi pogurani prema dolje i manje oštiri. To je zato što slika ima manje detalja. Ovdje se događa nešto specifično, a to je da je Fourierova transformacija Gaussove krivulje je opet Gaussova krivulja. Ako imamo usku krivulju, možemo je raširiti na ovaj način i obratno. Ako pogledamo Fourierove informacije tj. frekvenciju i pomnožimo ih sa zvonolikom krivuljom, kao rezultat dobijemo jedan ovako blaži graf sa spuštenim krajevima i to nam daje iste Fourierove informacije, ali u drugim intervalima. Kada promatramo to s računarske strane, možemo smanjiti kompleksnost na način da prevedemo stvari u termine frekvencije slike

gdje je kovolucija poput množenja (množimo dvije funkcije piksel po piksel) pa onda vratimo sve nazad u termine slike. Dakle, na taj način ovaj proces obavljamo brže !



Slika 3.13: Zamućena slika krda zebri i pripadni graf Fourierovog spektra

Pogledajmo dvodimenzionalni analogi grafički prikaz Fourierovog spektra u obliku toplinske karte. Dobivamo informacije koje govore koja je razina detaljanosti u vodoravnom i okomitom smjeru. Vidimo da su detalji konvoluntirane slike (zamućene slike zebri) koncentrirani u sredini. Dakle, kada običnu sliku pomnožimo s Gaussovom krvuljom u ovom Fourierovom prostoru, imamo manje detaljan spektar pa samim time i operacija konvolucije ide puno brže. U bibliotekama u koje je integrirana operacija konvolucije ili model konvolucijskih neuronskih mreža nemamo najobičnije jezgre koje se nalaze na početku ove bilježnice nego se koriste ove sofisticiranije.



Slika 3.14: Toplinska mapa originalne i zamućene slike

4. Širenje zaraze

4.1. Obrada podataka i vizualizacija

4.1.1. Okvir podataka za COVID-19

Za promatranje širenja zaraze, moramo prvo učitati prikupljene podatke. Podaci su preuzeti s interneta u .csv formati te u Juliji preoblikovati u okvir podataka.

	Province/State	Country/Region	Lat	Long	1/22/2020
1	missing	"Afghanistan"	33.9391	67.71	0
2	missing	"Albania"	41.1533	20.1683	0
3	missing	"Algeria"	28.0339	1.6596	0
4	missing	"Andorra"	42.5063	1.5218	0
5	missing	"Angola"	-11.2027	17.8739	0
6	missing	"Antigua and Barbuda"	17.0608	-61.7964	0
7	missing	"Argentina"	-38.4161	-63.6167	0
8	missing	"Armenia"	40.0691	45.0382	0
9	"Australian Capital Territory"	"Australia"	-35.4735	149.012	0
10	"New South Wales"	"Australia"	-33.8688	151.209	0

```
+ begin
+   csv_data = CSV.File("covid_data.csv");
+   data = DataFrame(csv_data)  # it is common to use 'df' as a variable name
+ end
```

Slika 4.1: Okvir podataka o širenju COVID-a 19

Izdvajamo zemlje iz okvira podataka.

```
all_countries =
► ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "A
◀ ┌─────────────────────────────────────────────────────────────────────────────────────────┐
  + all_countries = data[:, "country"]
```

Slika 4.2

```
countries =
► ["Afghanistan", "Albania", "Algeria", "Andorra", "Angola", "Antigua and Barbuda", "A
◀ ┌─────────────────────────────────────────────────────────────────────────────────┐
  + countries = unique(all_countries)
```

Slika 4.3

Recimo da želimo podatke za SAD. Nađemo indeks retka u kojemu su podaci za SAD i izdvojimo te podatke.

```
US_row = 250
· US_row = findfirst==(“US”), all_countries)

DataFrameRow (459 columns)

  province country latitude longitude 1/22/20 1/23/20 1/24/20 1/25/20 1/26/20
  String? String Float64? Float64? Int64 Int64 Int64 Int64 Int64
250 missing US 40.0 -100.0 1 1 2 2 5

· data[US_row, :]

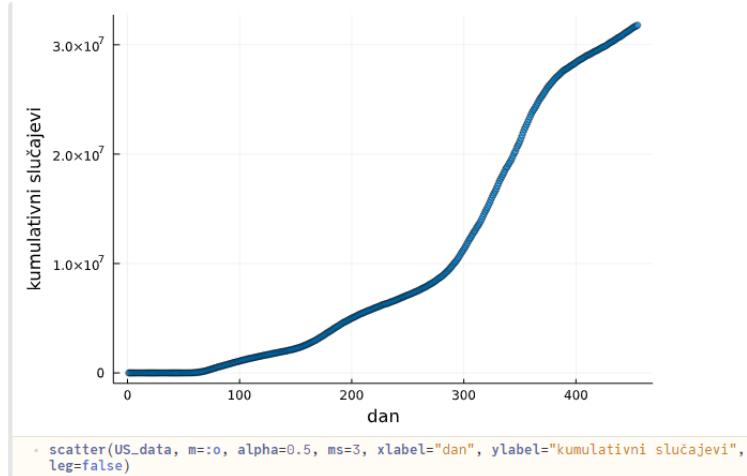
  province country latitude longitude 1/22/20 1/23/20 1/24/20 1/25/20 :: more
  1 missing "US" 40.0 -100.0 1 1 2 2

· data[US_row:US_row, :]

Now we can extract the data into a standard Julia Vector:
US_data =
▶ [1, 1, 2, 2, 5, 5, 5, 6, 6, 8, 8, 8, 11, 11, 11, 12, 12, 12, 12, 12, ... more]
◀ · US_data = Vector(data[US_row, 5:end])
```

Slika 4.4: Podaci za SAD

Podaci se lako prikazuju na grafu pomoću metode scatter() :

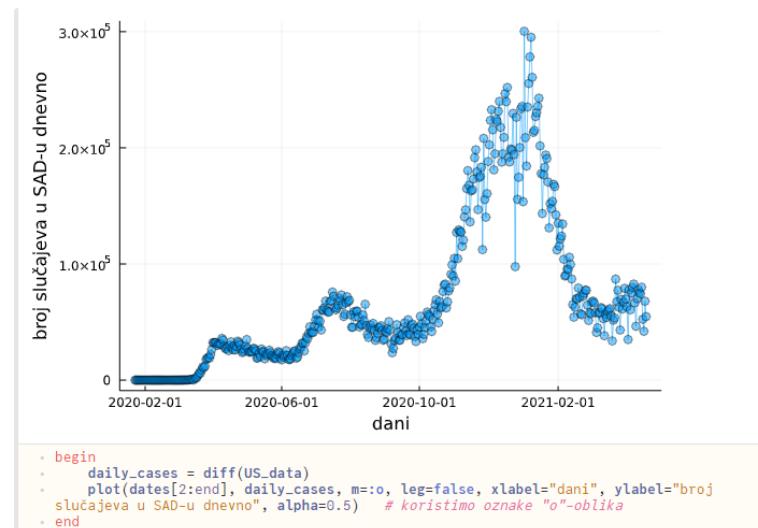


Slika 4.5: Podaci za SAD na grafu

Vidimo da je u SAD-u, npr. od prvog dana mjerena do, npr. 300-og dana, dakle negdje u rujnu, bilo oko 10 milijuna potvrđenih slučajeva.

4.1.2. Istraživačka analiza podataka (Exploratory data analysis)

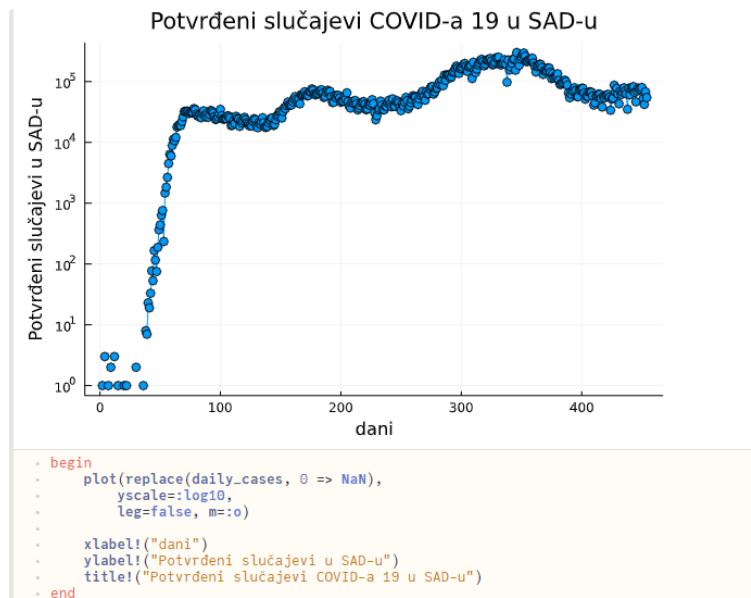
Rad s kumulativnim podacima često je manje intuitivan. Pogledajmo stvarni broj dnevnih slučajeva. Julia ima funkciju diff za izračunavanje razlike između uzastopnih unosa vektora:



Slika 4.6: Broj slučajeva zaraze u SAD-u dnevno

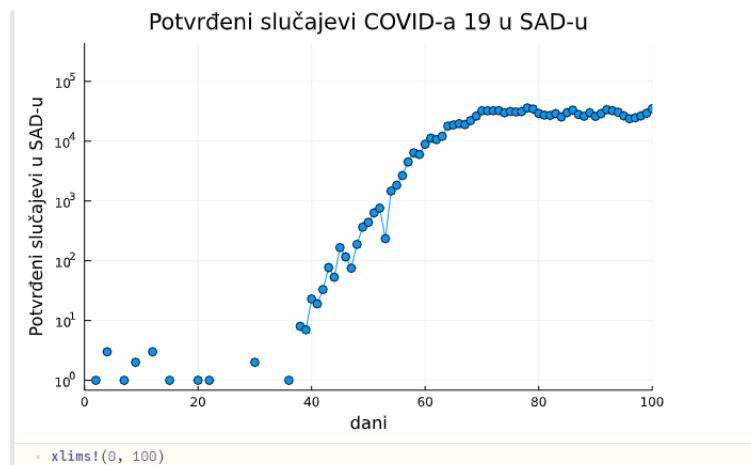
4.1.3. Eksponencijalni rast broja zaraženih

Jednostavni modeli širenja epidemije često predviđaju razdoblje s eksponencijalnim rastom. Potvrđuju li to podaci? Vizualna provjera za to sastoji se u prikupljanju podataka s logaritamskom skalom na osi (ali standardnom ljestvicom na osi).



Slika 4.7: Potvrđeni slučajevi COVID-a 19 u SAD-u

Zumirajmo regiju grafikona gdje rast izgleda linearno na ovom semi-log plot-u:

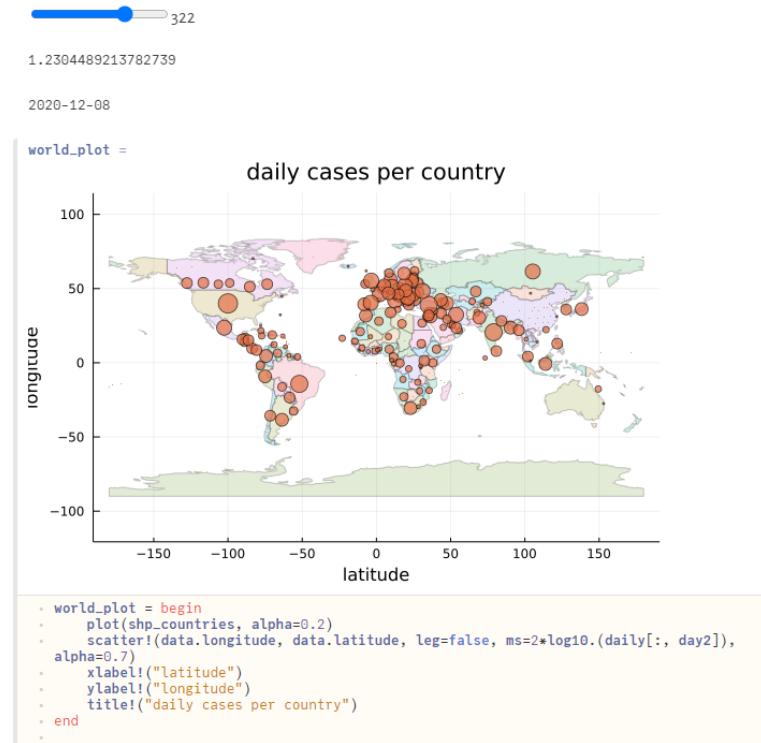


Slika 4.8: Potvrđeni slučajevi COVID-a 19 u SAD-u

Vidimo da postoji razdoblje koje traje oko dana 38 do otprilike 60 kad krivulja gleda ravno na semi-log plotu. To odgovara sljedećem datumskom rasponu [38,60] - Dakle, od kraja veljače do sredine ožujka 2020-te. Nametanje lockdown-a tijekom zadnjih 10 dana ožujka (različitim danima u različitim državama SAD-a) značajno je smanjilo prijenos virusa i to se vidi iz grafa.

Na koncu, učitani podaci iz početnog okvira podataka se mogu iskoristiti za stvaranje interaktivne mape koja u pozadini ima kartu svijeta, a svaka zemlja je označena kružićem koji se povećava/smanjuje u ovisnosti o ponašanju odgovarajuće funkcije

broja zaraženih za odgovarajuću zemlju. Na slici vidimo kako je svijet bio pogoden pandemijom nakon 322 dana od početka.



Slika 4.9: Potvrđeni slučajevi COVID-a 19 u SAD-u nakon 322 dana

4.2. Model širenja zaraze

4.3. Širenje zaraze putem binarnog stabla stabla

Uzmimo da "S" označava osjetljive osobe koje mogu biti zaražene, I označava zaražene osobe, a R oporavljene. U stablu, onaj tko je zaražen može zaraziti svoje susjede. Funkcija infekcije jednostavno po tim pravilima mijenja status osjetljivih u zaražene.

```

function infect(p::Person)
    if p.SIR == S
        p.SIR = I
        for neighbor in p.neighbors
            infect(neighbor)
        end
    else
        return
    end
end

```

Slika 4.10: Funkcija infekcije

Definiran je model u kojemu je čvor broj četiri oporavljen te se ne može zaraziti. Ostali su osjetljivi i djelujem funkcijom infekcije na prvog.

```

function sir_test()
    people = [Person(S,i) for i = 1:6]

    # setting the 4th person to be recovered
    people[4].SIR = R

    # this is where the helper function would allow us to specify only 1 dir!
    people[1].neighbors = [people[2], people[3]]
    people[2].neighbors = [people[3]]
    people[3].neighbors = [people[4], people[5]]
    people[5].neighbors = [people[6]]

    infect(people[1])

    for person in people
        println("infection status: ", person.SIR == I)
    end

    return people
end

```

Slika 4.11: SIR-model

Prvo je prenio zarazu na svoje susjede i tako su se redom svi zarazili do zadnjeg "lista" stabla osim, naravno, četvrtog čvora koji je oporavljen pa se više ne može zaraziti niti prenijeti zarazu.

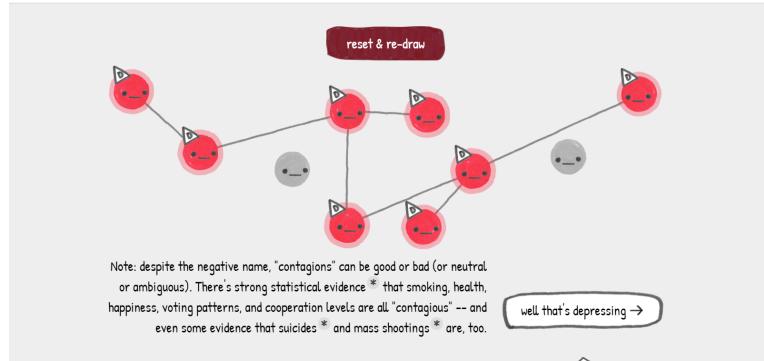
```

julia> include("SIR_test.jl")
sir_test (generic function with 1 method)

julia> tree = sir_test()
infection status: true
infection status: true
infection status: true
infection status: false
infection status: true
infection status: true

```

Slika 4.12: Rezultat zaraze SIR-modela



Slika 4.13: Grafički prikaz stabla i zaraze

4.4. Od mikro do makro; Od diskretnog do kontinuiranog

Proučavat ćemo kako se iz mikroskopsih stohastičkih modela (mala zajednica, mjesto) razvijaju makroskopski

U teoriji vjerojatnosti i srodnim poljima, stohastički ili slučajni proces je matematički objekt koji se obično definira kao obitelj slučajnih varijabli

Mikroskopski stohastički modeli, intuitivni su i korisni za uključivanje različitih učinaka. No, često nas zapravo zanima globalna (makro) slika o tome kako se epidemija razvija s vremenom: koliko je ukupno zaraznih jedinki u određenom trenutku, na primjer.

Moguće sažeti mikroskopsku dinamiku stohastičkog modela koristeći determinističke jednadžbe za makroskopski opis.

Makroskopske jednadžbe mogu biti u diskretnom vremenu (odnosi ponavljanja ili diferencijalne jednadžbe), ili možemo uzeti kontinuiranu granicu i pretvoriti ih u obične diferencijalne jednadžbe

4.4.1. Model oporavka

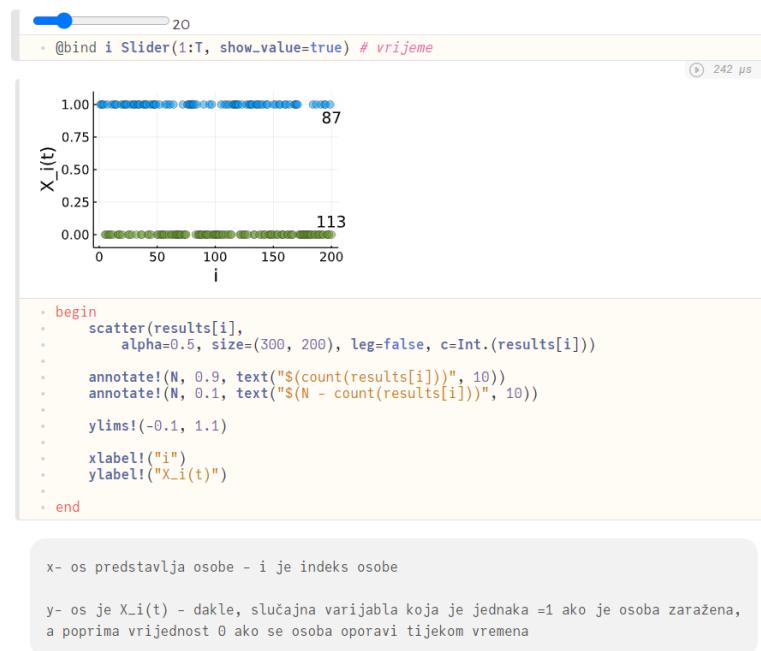
Vratimo se modelu oporavka od zaraze, s dopuštenim prijelazima : I->R. Imamo zaraženih ljudi u trenutku . Ako svaki ima vjerojatnost da se oporavi svaki dan, koliko ih je još zaraženih na dan broj ? Upotrijebimo pristup računalnog razmišljanja (computational thinking): krenimo kodiranjem simulacije i crtanjem rezultata.

Najprije generiramo funkciju step() koja predstavlja jedan korak i u tom koraku se zaraženi oporavljuju ovisno o odgovarajućoj slučajnoj Bernoullijevoj varijabli. Nadalje

petljom generiramo 100 logičkih vektora duljine N (u ovom slučaju N=200) pomoću funkcije step() koji diktiraju koji će se zaraženi oporaviti u pripadnom koraku.

Slika 4.14

Pogledajmo kakvo je stanje u 20-om koraku : Imamo 87 zaraženih i 113 oporavljениh. Naravno, svaki put dobijemo drugačije rezultate jer su nam logički vektori slučajni.



Slika 4.15

Sada možemo simulirati oporavak opet pomoću funkcije step() i to radimo 2 puta

tako da imamo dva pokretanja simulacije prikazana na grafu. Vidimo da broj infekcija opada s vremenom na sličan način u prvoj i drugoj simulaciji.

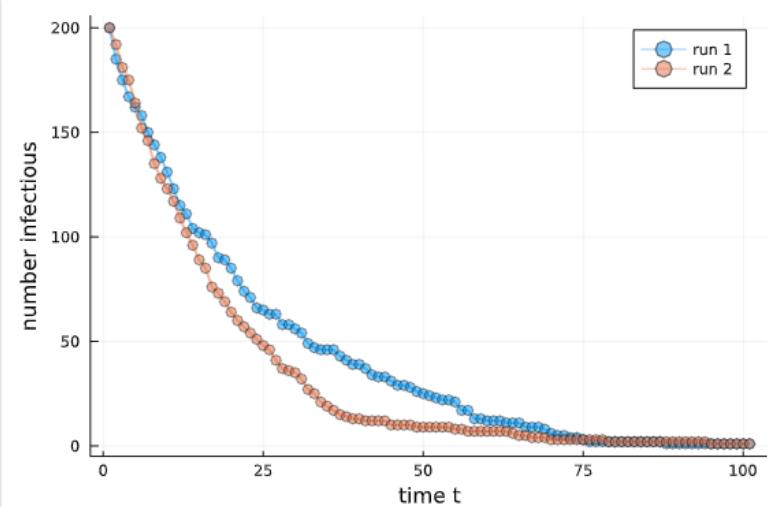
```

simulate_recovery (generic function with 1 method)
+   function simulate_recovery(p, T)
+     infectious = trues(N)
+     num_infectious = [N] # niz ima samo jedan element na početku -> el. N
+
+     for t in 1:T
+       step!(infectious, p)
+       push!(num_infectious, count(infectious))# push! dodaje elemente na kraj niza
+     end           # brojimo zaražene pomoću count()
+
+     return num_infectious
+ end

p = 0.1
+ p = 0.1

```

Na ovom grafu ispod su prikazane dvije simulacije - vidimo da su krivulje slične :



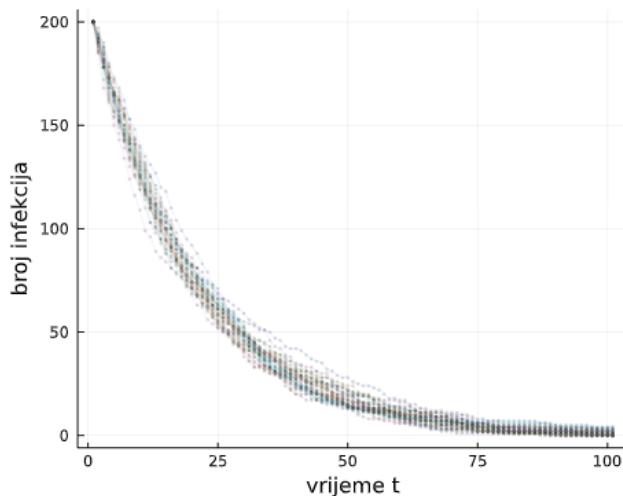
```

begin
  pp = 0.05
  plot(simulate_recovery(pp, T), label="run 1", alpha=0.5, lw=2, m=:o)
  plot!(simulate_recovery(pp, T), label="run 2", alpha=0.5, lw=2, m=:o)
  xlabel!("time t")
  ylabel!("number infectious")
end

```

Slika 4.16

Ako želimo nešto općenito zaključiti, trebamo puno simulacija - stohastički model izgleda kao dolje na grafu. Dakle, radi se veći broj pokretanja simulacija i svaka daje krivulju zaraženih ljudi po vremenu.



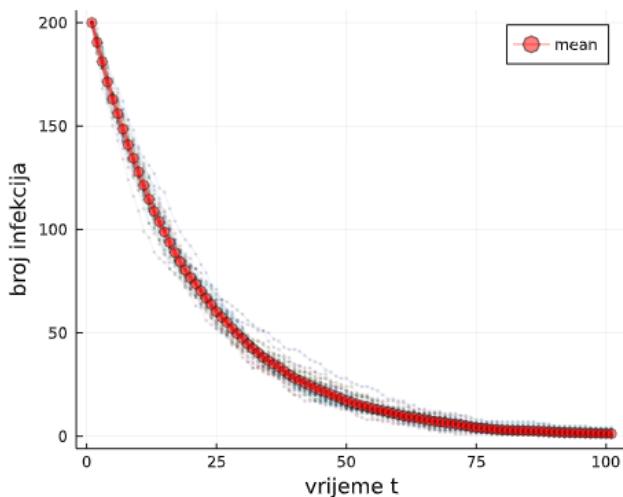
```

begin
plot(all_data, alpha=0.1, leg=false, m=:o, ms=1,
      size=(500, 400), label="")
 xlabel!("vrijeme t")
 ylabel!("broj infekcija")
end

```

Slika 4.17

Uzimamo mean svih krivulja - to je crvena krivulja. Mi trebamo pogoditi iz grafa o kojoj se funkciji radi. Rekli bismo da je eksponencijalna, ali moramo provjeriti!



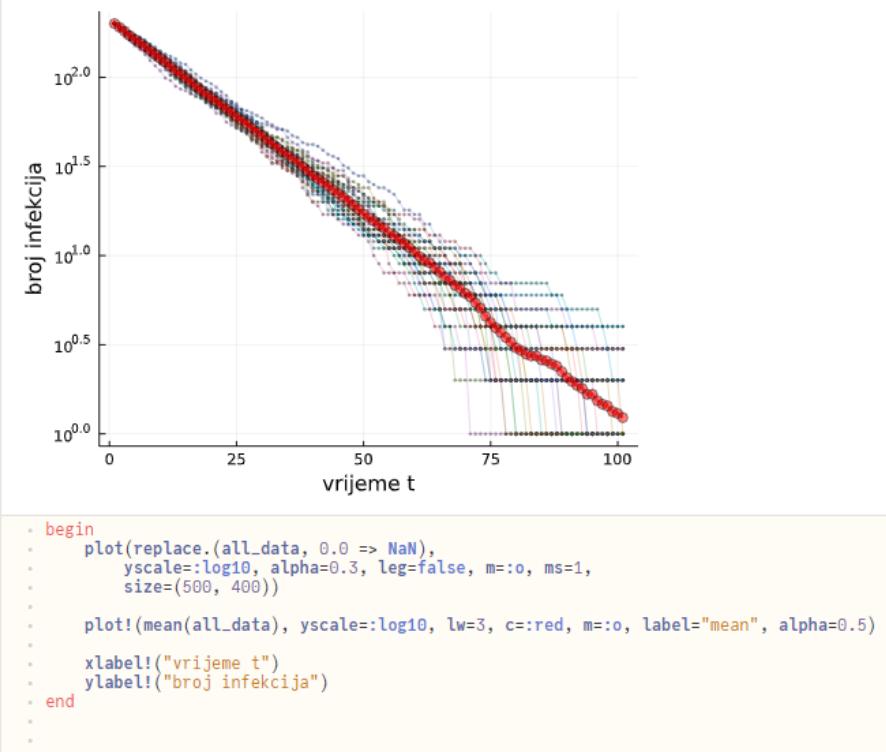
```

plot!(mean(all_data), leg=true, label="mean",
      lw=3, c=:red, m=:o, alpha=0.5,
      size=(500, 400))

```

Slika 4.18

Provjeru je li funkcija eksponencijalna provodimo tako što nacrtamo y os na logaritamskoj skali. Vidimo da jeste i da pada s određenom stopom.



Slika 4.19

To što znamo da je funkcija eksponencijalna, znači da bismo ju mogli derivirati!

4.4.2. Deterministička dinamika za srednju vrijednost (mean): Intuitivno deriviranje

Čini se da se srednja vrijednost (mean) s vremenom ponašala na prilično predvidljiv način. Možemo li to derivirati?

Čini se da se srednja vrijednost (mean) s vremenom ponašala na prilično predvidljiv način. Možemo li to derivirati?

Neka I_t bude broj zaraženih ljudi u vrijeme t . To se smanjuje jer se neki ljudi oporavljaju. Budući da se ljudi oporavljaju s vjerojatnosti p , broj ljudi koji se oporave u trenutku t je u prosjeku pI_t . [Imajte na umu da jedna vremenska jedinica odgovara jednom *zamahu* (sweep) simulacije.]

Imamo:

$$I_{t+1} = I_t - pI_t$$

ili

$$I_{t+1} = I_t(1-p).$$

U vrijeme t postoje I_t zarazne bolesti. Koliko prestaje (bivaju preboljene)? Svaka prestaje (preboljena je) s vjerojatnosti p , tako da se u prosjeku pI_t oporavi, pa se ukloni iz broja zaraznih, dajući promjenu $\Delta I_t = I_{t+1} - I_t = -pI_t$

Možemo preuređiti i riješiti relaciju :

$$I_{t+1} = (1-p)I_t.$$

pa

$$I_{t+1} = (1-p)(1-p)I_{t-1} = (1-p)^2 I_{t-1}$$

i time riješiti relaciju: (ovo dobivamo indukcijom)

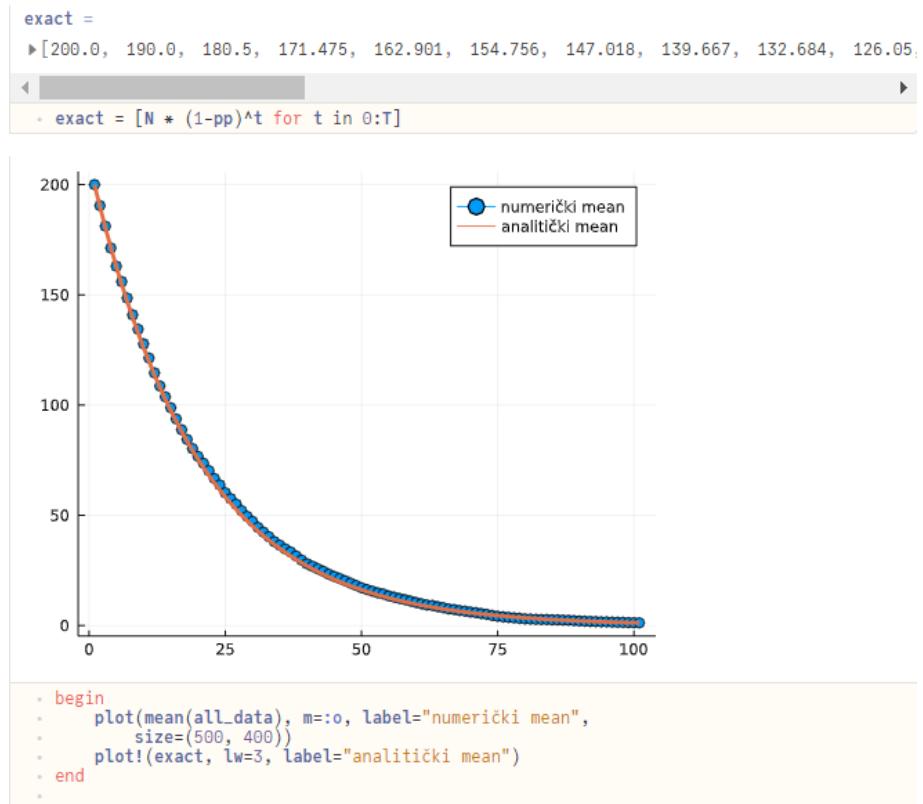
$$I_t = (1-p)^t I_0.$$

Slika 4.20

Dakle, $(1-p)$ je vjerojatnost da se zaražena osoba neće oporaviti. Tu vjerojatnost potenciramo s 't' (trenutak koji promatramo) i množimo s I_0 da bismo dobili broj zaraženih u trenutku 't'.

Usporedimo egzaktne i numeričke rezultate:

Egzaktna rješenja će trebati za analitički mean, a numerička rješenja (točke) već imamo iz grafova svih simulacija.



Slika 4.21

Dobro se slažu, kako bi i trebali. Očekuje se da će slaganje biti bolje (tj. fluktuacije manje) za veću populaciju.

Fluktacije - nepravilan porast i pad broja ili iznosa; varijacija

4.4.3. Deriviranje korištenjem srednje vrijednosti (mean-a) stohastičkih procesa

Umjesto da se u izvodu apeliramo na svoju intuiciju, možemo formulirati točan opis stohastičkog procesa u smislu slučajnih varijabli: X_t^i uzima vrijednost 1 ako je i -ta osoba zaražena u trenutku t , i 0 ako su se oporavili, kao u gornjem grafu (plotu).

Svakim okretanjem novčića dobiva se nezavisna Bernoullijeva slučajna varijabla B_t^i , koja ima vrijednost 1 (true) s vjerojatnošću p , i 0 (false) s vjerojatnošću $1 - p$.

Vrijednost u sljedećem vremenskom koraku, X_{t+1}^i , dana je s

$$X_{t+1}^i = \begin{cases} 0 & \text{ako } X_t^i = 0 \text{ ili } B_t^i = 1 \\ 1 & \text{inače} \end{cases}$$

Slika 4.22

(lijeva strana) i-ta osoba u sljedećem vremenskom trenutku :
 (desna strana) I) =0, ako se osoba u prethodnom trenutku već oporavila
 ili ako je Bernoullijeva slučajna varijabla=1 tj.
 "bacali smo novčić i palo je pismo - osoba se 100% oporavlja"
 II) =1 , ako se nije dogodilo navedeno pod I),
 situacija se ne mijenja za i-tu osobu, ona ostaje =1
 tj. zaražena (kad smo bacali novčić, pala je glava,
 a ni prethodno se nije bila oporavila i-ta osoba)

Slika 4.23

Možemo to zapisati kao :

$$X_{t+1}^i = X_t^i (1 - B_t^i).$$

Slika 4.24

Dakle, kada je B_t=0, onda je X_(t+1)=X_t (osoba ostaje zaražena). Kada je B_t=1 (oporavak garantiran), onda je X_(t+1)=0, zaraza je gotova. Tako modeliramo i prikazujemo dinamiku zaraze i oporavka pomoću slučajnih varijabli.

Označimo mean slučajne varijable X, označeno sa $\langle X \rangle$. U teoriji vjerojatnosti to se često naziva očekivanje (ili "očekivana vrijednost") i piše $E[X]$.

Ono što očekujemo treba biti jednakoj srednjoj vrijednosti (mean) svih simulacija. Ili, teoretski, imamo rezultate za neku populaciju pa uzimamo mean toga te očekujemo tu vrijednost.

Uzimajući očekivanja obje strane gornje jednadžbe i koristeći da je očekivanje umnoška neovisnih slučajnih varijabli umnožak njihovih očekivanja, dobivamo:

$$\langle X_{t+1}^i \rangle = \langle X_t^i \rangle (1 - \langle B_t^i \rangle).$$

Srednja vrijednost (MEAN) svake neovisne {bacanje novčića je neovisno}, ali identično distribuirane Bernoullijeve varijable B_t^i je p .

Da bismo pronašli srednji ukupan broj zaraženih pojedinaca, samo zbrajamo logičke (boolean) varijable X_t^i :

$$I_t = \left\langle \sum_{i=1}^N X_t^i \right\rangle.$$

Slika 4.25

Zbrajamo slučajne varijable koje su ili 1 ili 0 te je I_t njihova ukupna suma. I onda od te sume uzimamo MEAN da bismo dobili broj zaraženih ljudi.

Sastavljanje svega ovoga vraća nam naš prethodni rezultat,

$$I_{t+1} = (1 - p) I_t.$$

Slika 4.26

Tako smo dobili makroskopski opis u diskretnom vremenu. Dakle, uzimali smo, recimo vremenski korak od jedan dan. Ali, možda želimo biti precizniji i gledati št se događa na razini sati ili minuta. To onda moramo promatrati u neprekidnom vremenu.

4.4.4. Neprekidno vrijeme

Ako grafikon srednje vrijednosti (MEAN-a) promatramo u ovisnosti o vremenu, čini se da slijedi glatkou krivulju. Zapravo ima smisla pitati ne samo koliko je ljudi ozdravilo svaki dan, već težiti finijoj granulaciji.

Digresija za bolje razumijevanje, laički rečeno: "Neprekidno" odmah asocira na neprekidnu funkciju. Dakle, derivabilnu funkciju. Ako se prisjetimo definicije derivacije, znamo da se u izrazu pojavljuju delte. Te delte označavaju promijenu neke varijable (vrijednosti). Vrijednost koja se kod nas mijenja i koju gledamo na x-osi na grafu je vrijeme. Ima smisla prvo tražiti promijenu vremena i ovisno o njoj dalje formirati izraz. Sada nam 't' više nije indeks jer ne promatramo vrijeme kao korak (npr. dan) već promatramo što se događa u, npr. minutama. Stoga, nam je sada I funkcija ovisna o vremenu, a ne varijabla s indeksom koji označava vremenski korak.

Pretpostavimo da povećavamo vrijeme u koracima od δt ; gornja analiza bila je za $\delta t = 1$.

Tada ćemo morati prilagoditi vjerojatnost oporavka u svakom vremenskom koraku. Ispada da, da bismo imali smisla gledati u granici $\delta t \rightarrow 0$ (naravno, želimo što preciznije rezultate i zbog tog gledamo da nam je vremenska promjena što manja tj. da teži u 0), moramo odabratи vjerojatnost $p(\delta t)$ da se oporavi u vremenu t koja će biti proporcionalna δt :

$$p(\delta t) \simeq \lambda \delta t,$$

gdje je λ stopa oporavka. Imajmo na umu da je stopa vjerojatnost po jedinici vremena. [bilješka : a to da je stopa vjerojatnost znači da je iz intervala $[0,1]$]

Dobivamo :

$$I(t + \delta t) - I(t) \simeq -\lambda \delta t I(t)$$

Dijeljenjem s δt dobivamo :

$$\frac{I(t + \delta t) - I(t)}{\delta t} \simeq -\lambda I(t)$$

Lijevu stranu prepoznajemo kao definiciju **derivacije** kada je $\delta t \rightarrow 0$. Uzimanje te granice konačno daje

$$\frac{dI(t)}{dt} = -\lambda I(t)$$

Odnosno, dobivamo **običnu diferencijalnu jednadžbu** (ODJ) koja rješenje daje implicitno. Rješavanje ove jednadžbe s početnim uvjetom $I(0) = I_0$ daje

$$I(t) = I_0 \exp(-\lambda t).$$

Alternativno, to možemo izvesti prepoznavanjem eksponencijala u granici $\delta t \rightarrow 0$ sljedećeg izraza, koji je u osnovi izraz za složenu kamatu:

$$I_t = (1 - \lambda \delta t)^{(t/\delta t)} I_0$$

Slika 4.27

4.4.5. SIR-model

Sada proširimo postupak na puni SIR model, $S \rightarrow I \rightarrow R$. Budući da već znamo kako se nositi s oporavkom, uzmite u obzir samo SI model, gdje su osjetljivi agenti zaraženi kontaktom, s vjerojatnošću

Označimo sa S_t i nek je I_t broj osjetljivih (S) i zaraznih (I) ljudi u vrijeme t , odnosno za N ukupan broj ljudi.

U projektu, u svakom premetanju (sweep) svaki zarazni pojedinac ima priliku stupiti u interakciju s jednom drugom osobom. Ta je jedinka slučajno izabrana uniformno iz ukupne populacije veličine N . Ali nova infekcija događa se samo ako je odabrani pojedinac osjetljiv (S), što se događa s vjerojatnošću S_t/N , a zatim ako je zaraza uspješna, s vjerojatnošću b , recimo.

Stoga je promjena u broju zaraznih ljudi nakon tog koraka.

Smanjenje u S_t je također dano s ΔI_t .

$$\Delta I_t = I_{t+1} - I_t = b I_t \left(\frac{S_t}{N} \right)$$

Korisno je normalizirati s N , stoga definiramo:

$$s_t := \frac{S_t}{N}; \quad i_t := \frac{I_t}{N}; \quad r_t := \frac{R_t}{N}$$

Uključujući oporavak s vjerojatnosti c , dobivamo **SIR model s diskretnim vremenom**:

$$\begin{aligned} s_{t+1} &= s_t - b s_t i_t \\ i_{t+1} &= i_t + b s_t i_t - c i_t \\ r_{t+1} &= r_t + c i_t \end{aligned}$$

Slika 4.28

- i) BROJ OSJETLJIVIH U SLJEDEĆEM TRENUTKU = BROJ OSJETLJIVIH U OVOM TRENUTKU
 - (ZARAZNIH U OVOM TRENUTKU) [KOJI ĆE ZARASITI]
 * (OSJETLJIVE U OVOM TRENUTKU)
 * (S VJEROJATNOŠĆU USPJEŠNE ZARAZE)
- ii) BROJ ZARAZNIH U SLJEDEĆEM TRENUTKU = (BROJ ZARAZNIH TRENUTNO)
 + (BROJ ZARAZNIH TRENUTNO KOJI ZARAZE OSJETLJIVE S VJEROJATNOŠĆU USPJEŠNE ZARAZE)
 - (BROJ ZARAŽENIH TRENUTNO KOJI SU SE OPORAVILI S VJEROJATNOŠĆU 'c')
- iii) BROJ OPORALJENIH U SLJEDEĆEM TRENUTKU = (BROJ OPORAVLJENIH TRENUTNO)
 + (BROJ ZARAŽENIH TRENUTNO KOJI SU SE OPORAVILI S VJEROJATNOŠĆU 'c')

Slika 4.29

To opet možemo dobiti iz stohastičkog procesa uzimajući očekivanja . [Zanemarimo oporavak za početak i uzmite varijable Y_t^i koje su 0 ako je osoba osjetljiva i 1 ako je zaražena.]

I opet možemo dopustiti da se procesi odvijaju u koracima duljine δt i uzeti granicu $\delta t \rightarrow 0$. Uz stope β i γ dobivamo standardni (kontinuirani) **SIR model**:

$$\begin{aligned}\frac{ds(t)}{dt} &= -\beta s(t) i(t) \\ \frac{di(t)}{dt} &= +\beta s(t) i(t) - \gamma i(t) \\ \frac{dr(t)}{dt} &= +\gamma i(t)\end{aligned}$$

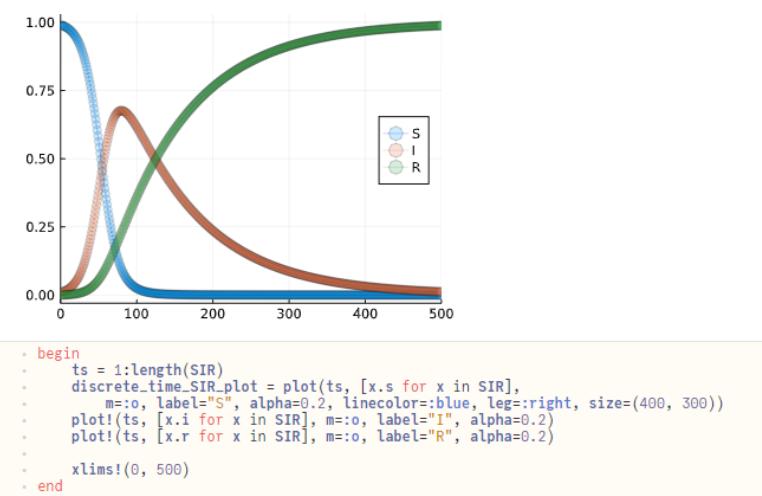
Slika 4.30

Ovdje derivaciju promatramo kao fizičari, promijena u vremenu. Dakle, od broja osjetljivih u sljedećem trenutku oduzimamo broj osjetljivih u ovom trenutku i tako dobivamo lijevu stranu. Na desnoj nam ostaje ono što je gore napisano, ali u notaciji gdje imamo varijable ovisne o vremenu i konstatne stope kojima ih množimo.

To možemo smatrati modelom kemijske reakcije s vrstama S, I i R. Pojam $s(t)i(t)$ poznat je kao **masovno djelovanje** oblik interakcije.

Imajmo na umu da niti jedno analitičko rješenje ovih (jednostavnih) nelinearnih ODJ-a nije poznato kao funkcija vremena! (Međutim, **parametarska rješenja su poznata**.)

Ispod je simulacija modela diskretnog vremena. Imajmo na umu da se najjednostavnija numerička metoda za rješavanje (približno) sustava ODJ-ova, **Eulerova metoda**, u osnovi svodi na rješavanje modela s diskretnim vremenom! Čitav paket naprednijih ODJ solvera nalazi se u **ekosistemu Julia DiffEq**.



Slika 4.31

5. Zaključak

U jeziku Julia i okruženju kao što su Pluto bilježnice zaista se mogu primjenjivati konvolucije na razne načine u korisne svrhe. Pokazano je kako funkcioniра kernel i kako se dolazi do filtriranih slika te značajnost konvolucija za optimiziranje i ubrzanje računanja. Julia je također moćan jezik za obradu i vizualizaciju podataka. Paketi za Juliju omogućavaju prikaz okvira podataka, ekplorativnu analizu i crtanje grafova. Iz grafova se dalo zaključiti da se zaraza koju smo pratili širila eksponentijalno, ali smo prikazali i kako izgleda širenje zaraze pomoću stablastog modela. To sve daje uvid u situaciju i zgodno je za dobivanje rezultata koji se mogu analizirati i na temelju kojih donosimo zaključke, ali prava korisnost je u vidu simulacija kojima možemo predvidjeti što će se dogoditi te na temelju tih simulacija donositi odluke i pomoći sprječavanju zaraze. U ovom seminarskom radu je obrađen stohastički model oporavka te je iz njega lako zaključiti koliki je vremenski period potreban da se određen broj ljudi oporavi.

6. Literatura

- [1] The julia programming language. [://computationalthinking.mit.edu/Fall20/Introduction to Computational Thinking](https://computationalthinking.mit.edu/Fall20/Introduction%20to%20Computational%20Thinking), 2021.