UNIVERZA V LJUBLJANI
University of Ljubljana

Programsko orodje R v vodarstvu

Nejc Bezak

Ljubljana, 2024

Programsko orodje R v vodarstvu
Univerzitetni učbenik

Avtor: *Nejc Bezak*
Recenzenta: *Mojca Šraj, Lovrenc Pavlin*
Jezikovni pregled: *Mojca Vilfan*
Oblikovalec naslovnice: *Gašper Mrak*
Oblikovanje in prelom: *Nejc Bezak*
Bibliografsko-informacijska podpora: *Elizabeta Kralj, Matevž Rudolf*

Založnik: *Založba Univerze v Ljubljani*
Za založbo: *Gregor Majdič, rektor Univerze v Ljubljani*
Izdajatelj: *Univerza v Ljubljani, Fakulteta za gradbeništvo in geodezijo*
Za izdajatelja: *Violeta Bokan Bosiljkov, dekanja Fakultete za gradbeništvo in geodezijo*

Tisk učbenika je finančno podprla Unesco Katedra za zmanjševanje tveganj ob vodnih ujmah: https://www.unesco-floods.eu/si/.

Ljubljana, 2024
Prva izdaja, naklada: 50 izvodov
Publikacija je brezplačna.

Prva e-izdaja.
Publikacija je v digitalni obliki prosto dostopna na: https://ebooks.uni-lj.si
DOI: 10.15292/9789612974442

Kataložna zapisa o publikaciji (CIP) pripravili v
Narodni in univerzitetni knjižnici v Ljubljani

Tiskana knjiga
COBISS.SI-ID 213658115
ISBN 978-961-297-443-5

E-knjiga
COBISS.SI-ID 213709315
ISBN 978-961-297-444-2 (PDF)

Kazalo vsebine

Kazalo slik

1 Predgovor

Ta učbenik je namenjen vsem tistim, ki želite pridobiti poglobljeno znanje o analizah podatkov z uporabo programskega orodja R na področju vodarstva in okoljskega inženirstva ter se naučiti učinkovito uporabljati programsko orodje R kot orodje za obdelavo, analize in interpretacijo podatkov. Še posebej je učbenik namenjen študentom pri predmetu Programsko orodje R v vodarstvu in drugih predmetih, kjer se programsko orodje R pogosto uporablja, kot sta Hidrologija in Hidrološko modeliranje.

Programsko orodje R je odprtokodno in prostodostopno statistično orodje, ki nudi širok nabor funkcij za analizo podatkov, vizualizacijo rezultatov ter razvoj statističnih in drugih modelov. V sklopu učbenika bomo skozi praktične primere in grafične prikaze raziskovali, kako uporabiti programsko orodje R za reševanje različnih problemov na področju vodarstva in okoljskega inženirstva ter širše. Obogatili bomo znanje s konkretnimi primeri analiz vodarskih podatkov, hidroloških modelov, prostorskih analiz in drugih ključnih vidikov tega pomembnega področja. V učbeniku so prikazani številni praktični primeri uporabe programskega jezika R in številnih paketov, na voljo pa je še veliko drugih paketov, katerih uporabe tu ne prikazujemo posebej.

Namen tega učbenika ni le usvojiti tehnično znanje o programskem orodju R, temveč tudi spodbuditi razmišljanje o kompleksnih vprašanjih, povezanih z vodarstvom, ter razviti sposobnost kritičnega pristopa k analizi in interpretaciji podatkov in modelov. Vključene so tudi krajše praktične naloge, s katerimi lahko dodatno poglobite svoje znanje.

Verjamem, da bo ta učbenik koristno orodje za študente, raziskovalce, strokovnjake in vse tiste, ki se želijo poglobiti v področje vodarstva s pomočjo programskega orodja R. Želim vam uspešno delo in veliko zadovoljstva pri raziskovanju vodarskih izzivov!

Avtor

2 Programski jezik R in njegova zgodovina

2.1 Programski jezik S

R je različica programskega jezika S. S je programski jezik, ki so ga razvili John Chambers in sodelavci v podjetju Bell Telephone Laboratories. Programski jezik S se je začel uporabljati leta 1976 kot notranje okolje za statistične analize, prvotno je bil v uporabi v obliki knjižnic Fortran.

Leta 1988 je bil sistem na novo napisan v jeziku C in je začel spominjati na današnji sistem programskega jezika S. Različica 4 jezika S je bila pripravljena leta 1998 in jo uporabljamo tudi danes. Knjiga Programming with Data Johna Chambersa[1] opisuje to različico jezika.

Programski jezik S se je razvijal tudi od izdaje omenjene knjige, a njegove osnove se niso bistveno spremenile. Leta 1998 je programski jezik S prejel zelo prestižno nagrado na področju računalništva, nagrado združenja Association for Computing Machinery's Software System Award.

Filozofija programskega jezika S je nekoliko drugačna od filozofije običajnih programskih jezikov. Namen razvijalcev je bil ustvariti jezik, ki bi bil primeren tako za interaktivno analizo podatkov preko ukazne vrstice (angl. *command line*) kot tudi za pisanje daljših programov, kar je bolj podobno tradicionalnim programskim jezikom.

2.2 Programski jezik R

Ena od ključnih omejitev jezika S je bila, da je bil na voljo le v komercialnem paketu S-PLUS. Zato sta leta 1991 Ross Ihaka in Robert Gentleman na oddelku za statistiko na univerzi v Aucklandu ustvarila R, ki je bil leta 1993 tudi prvič javno objavljen. Izkušnje z razvojem programskega jezika R so opisane v članku v reviji Journal of Computational and Graphical Statistics iz leta 1996[2].

Leta 1995 je Martin Mächler prepričal prvotna avtorja, da sta uporabila javno licenco GNU (General Public Licence) in naredita R prostodostopen. To je omogočilo dostopnost izvorne kode celotnega sistema R vsakomur, ki se je želel ukvarjati z njo.

Leta 1997 je bila ustanovljena skupina R Core Group, v kateri je bilo večje število strokovnjakov, povezanih s S in S-PLUS. Trenutno ima ta skupina nadzor nad izvorno kodo programskega jezika R. Leta 2000 je bila objavljena različica R 1.0.0.

Danes R deluje na skoraj vseh standardnih računalniških platformah in operacijskih sistemih. Njegova odprtokodna narava omogoča, da lahko vsakdo prilagodi program na kateri koli platformi po lastni izbiri. R med drugim deluje tudi na sodobnih tabličnih računalnikih, telefonih, dlančnikih in igralnih konzolah.

Prednost programskega orodja R so pogoste posodobitve. Vsako leto, običajno oktobra, izdajo večjo posodobitev, v katero vključijo pomembne nove funkcije. Med letom se po

[1] https://link.springer.com/book/9780387985039.

[2] https://www.jstor.org/stable/1390807.

potrebi izdajajo manjši popravki za odpravljanje napak. Pogoste posodobitve in redno izdajanje kažejo na aktiven razvoj programske opreme in zagotavljajo, da so morebitne napake in druge težave pravočasno odpravljene. Osnovni razvijalci nadzorujejo primarno izvorno kodo, veliko ljudi po vsem svetu pa prispeva k razvoju v obliki novih funkcij, popravkov napak ali obojega.

Glavna prednost programskega orodja R pred številnimi drugimi statističnimi paketi je, da je brezplačen v smislu brezplačne programske opreme. Avtorske pravice za primarno izvorno kodo programa R ima R Foundation[3], objavljena pa je pod splošno javno licenco GNU različice 2.0[4]. V skladu z licenco lahko uporabnik: (i) prosto uporablja program za kateri koli namen; (ii) prilagodi program in dostopa do izvorne kode; (iii) izboljša program.

Še ena prednost, ki jo ima R pred mnogimi drugimi statističnimi programi, so njegove grafične zmogljivosti. Zmožnost R, da ustvari grafiko dobre kakovosti, obstaja že od samega začetka in je na splošno boljša od konkurenčnih programov. Danes, ko je na voljo veliko več vizualizacijskih programov kot v preteklosti, se ta trend nadaljuje. Osnovni grafični sistem programskega orodja R omogoča zelo natančen nadzor nad vsemi elementi grafa. Drugi novejši grafični sistemi in paketi, kot sta *lattice* in *ggplot2*, pa omogočajo kompleksne in grafično lepše vizualizacije podatkov.

Eden od pozitivnih vidikov pri uporabi programskega orodja R ni povezan s samim jezikom, temveč z aktivno skupnostjo uporabnikov. V mnogih pogledih je jezik uspešen, če ustvari platformo, s katero lahko veliko ljudi ustvarja nove elemente. R je takšna platforma in na tisoče ljudi po vsem svetu prispeva k razvoju R. R ima obsežno podporo na spletni strani Stack Overflow[5]. Omeniti velja tudi odlično dokumentacijo (npr. standardiziran opis pomoči funkcij) in pa centralni repozitorij razširitev.

Treba je poudariti, da ima R tudi pomanjkljivosti. Objekti v programu R morajo biti na primer shranjeni v fizičnem pomnilniku, kar je lahko težava pri analizah velike količine podatkov (npr. globalne analize, podnebni scenariji). Nadalje funkcionalnost R temelji na povpraševanju potrošnikov in prostovoljnih prispevkih uporabnikov. Če določena metoda še ni implementirana, jo mora uporabnik implementirati sam ali pa mora za to nekomu plačati. Tudi potrebni začetni vložek učenja je večji kot pri kakšnem drugem orodju za analizo podatkov (npr. pri Excelu).

2.3 Dodatna literatura

Na voljo je veliko prostodostopne literature s koristnimi informacijami, kako uporabljati R:

- Uvod v R na CRAN spletni strani[6].

---

[3] https://www.r-project.org/foundation/.

[4] https://www.gnu.org/licenses/old-licenses/gpl-2.0.html.

[5] https://stackoverflow.com/.

[6] https://cran.r-project.org/doc/manuals/r-release/R-intro.html.

- Gradivo na spletni strani R[7].
- Advanced R[8].
- Gradivo na spletni strani bookdown.org[9].
- Priročnik prof. Blejca[10].
- Knjiga osnovna statistična analiza prof. Žiberne[11].
- Spletna stran Rpubs in praktični primeri[12].

3 Kako začeti uporabljati R

3.1 Namestitev

Za začetek dela s programskim orodjem R morate orodje najprej namestiti na računalnik. R deluje na skoraj vseh operacijskih sistemih, ki so na voljo, vključno s široko dostopnimi sistemi Windows, Mac OS in Linux. Datoteke za namestitev so na voljo na spletni strani R-project[13].

Za R je na voljo tudi več grafičnih vmesnikov. Eden od njih je RStudio[14], ki ima lep urejevalnik z označevanjem posameznih sklopov kode (npr. komentarji, ukazi, objekti), pregledovalnik objektov in številne druge funkcije, ki olajšajo delo s programskim orodjem R. Tudi RStudio je prostodostopen in deluje na različnih operacijskih sistemih, kot so Windows, Mac OS in Linux.

---

[7] https://www.r-project.org/.

[8] https://www.routledge.com/Advanced-R-Second-Edition/Wickham/p/book/9780815384571.

[9] https://bookdown.org/.

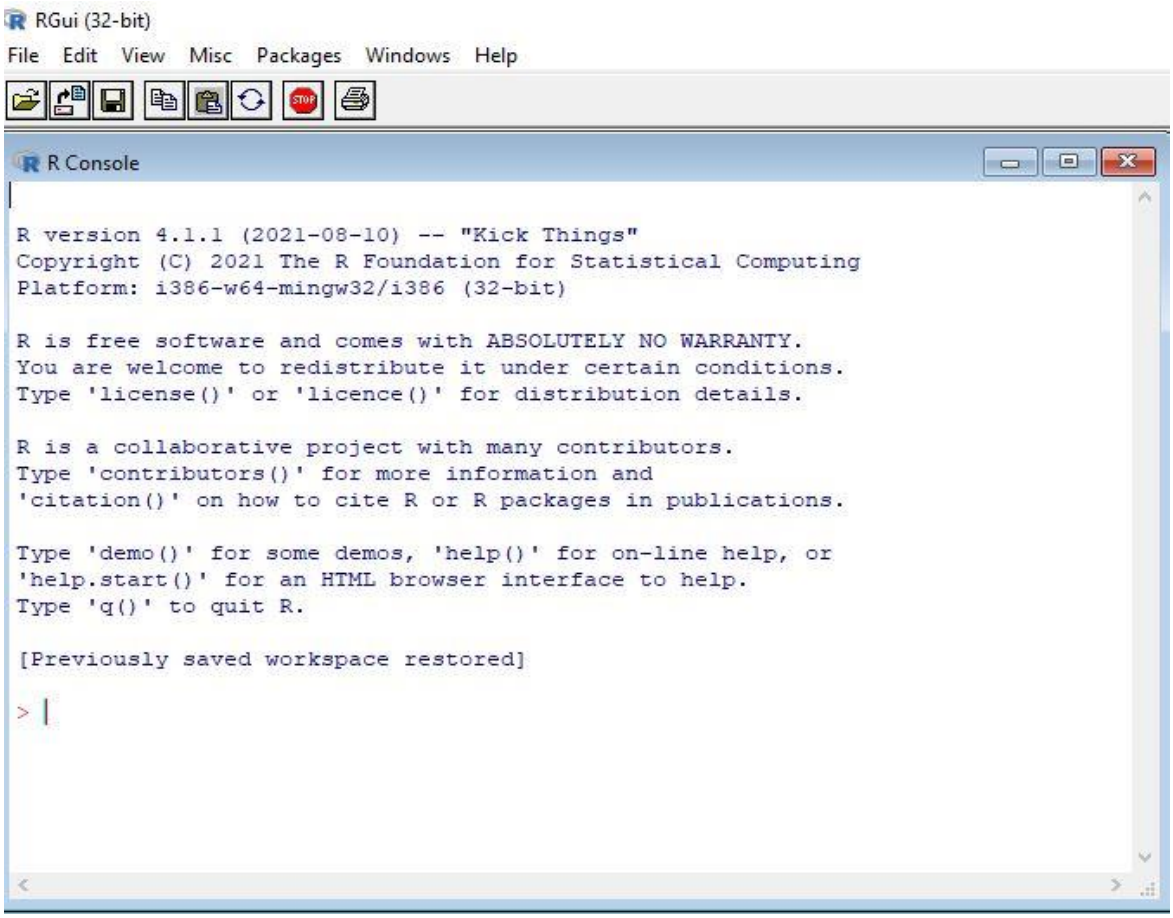
[10] https://ablejec.nib.si/R/#Manual.

[11] https://www.fdv.uni-lj.si/zalozba/domov/osnovna-statistična-analiza-v-r-ju.

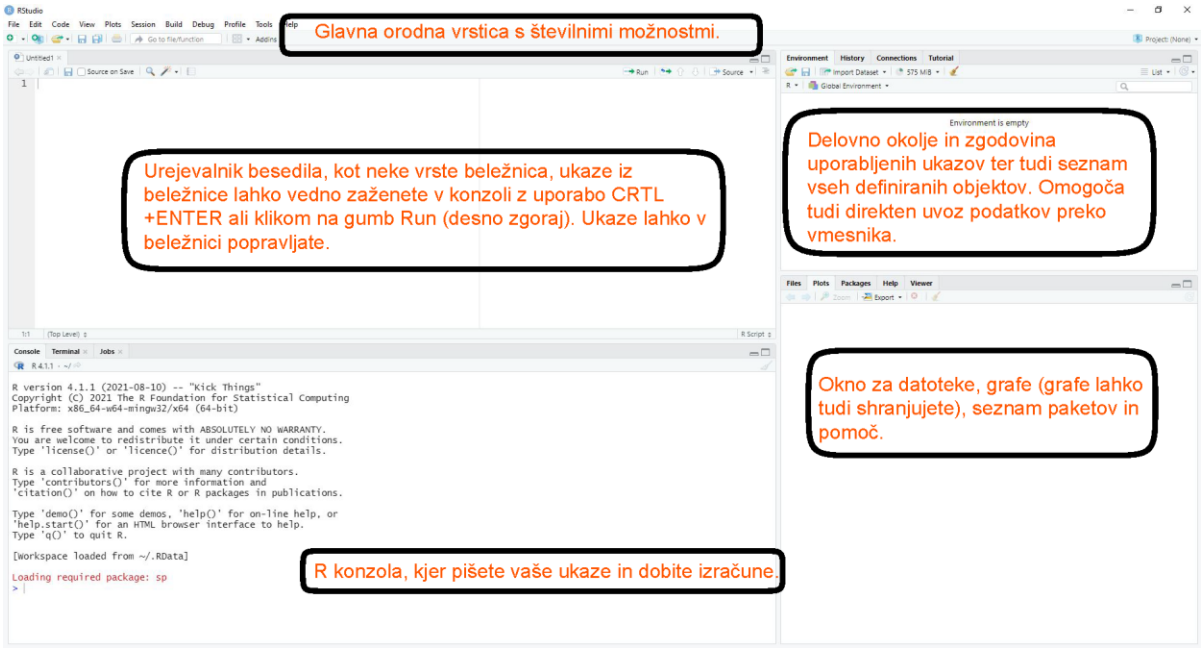
[12] https://rpubs.com/.

[13] https://cloud.r-project.org/.

[14] https://posit.co/.



Slika 1: Primer osnovnega programa R.



Slika 2: Primer grafičnega vmesnika RStudio.

3.2 Prvi koraki

Zdaj, ko ste namestili R in RStudio, se verjetno sprašujete: »Kako naj sedaj uporabljam R?« Najprej je treba poudariti, da je R za razliko od drugih statističnih orodij, kot sta npr. Excel ali SPSS, ki ponujajo grafične vmesnike »pokaži in klikni«, jezik, kjer morate v orodno vrstico (konzolo – Console) vnesti ukaze, napisane v kodi R. Z drugimi besedami, programirati morate v jeziku R. Čeprav za uporabo R ni treba biti izkušen programer, je še vedno treba razumeti niz osnovnih konceptov. R lahko uporabljate kot preprost kalkulator preko pisanja ukazov v konzolo (Console) in s pritiskom na gumb Enter:

```
3 + 4 # seštevanje

## [1] 7

3*4 # množenje

## [1] 12

8/2 # deljenje

## [1] 4
```

R omogoča tudi pisanje komentarjev (del kode, ki se ne izvrši) v kodi. Za ta namen se lahko uporabi znak #:

```
3 + 4 # to je komentar

## [1] 7
```

Možni so seveda tudi kompleksnejši izračuni:

```
2*(3+8)/4+5*1.5/(2/3)^3 # malce bolj kompleksen izračun

## [1] 30.8125
```

R prikaže sporočila, kadar pri izvedbi kode pride do nepravilnosti. Obstajajo tri različne vrste sporočil:

- Napake (angl. *Errors*): Če je rdeče besedilo napaka, je pred njim izpis »Error in…« (Napaka v…), ki poskuša pojasniti, kaj je šlo narobe. Običajno se izvedba kode ob napaki nemudoma ustavi.
- Opozorila (angl. *Warnings*): Kadar je izpisano besedilo opozorilo, je pred njim izpisano »Warning: «, R pa poskuša razložiti, zakaj je opozorilo zapisano. V večini primerov bo koda še vedno delovala, vendar morda z nekaterimi omejitvami, ali pa izračuni ne bodo pravilni oziroma v skladu s tem, kar ste od R zahtevali. Pri opozorilih torej svetujemo previdnost in kontrolo izračunov.
- Sporočila (angl. *Messages*): Če se izpisano besedilo ne začne z »Error« ali »Warning«, je to le sporočilo. Ta sporočila boste videli, ko boste nalagali pakete R ali ko boste z določenimi funkcijami uvažali podatke. To so koristna diagnostična sporočila in ne preprečujejo delovanja kode.

3.3 Uporaba funkcij

Ena izmed prednosti uporabe programskega orodja R je vključenost velikega števila funkcij v dodatne pakete. Uporaba funkcij je precej preprosta, treba pa se je zavedati, kaj so vhodni podatki posamezne funkcije oziroma kaj funkcija določi kot rezultat. Do dokumentacije oz. pomoči pri uporabi funkcije lahko dostopamo z enim od naslednjih ukazov:

```
help(mean) # pomoč za funkcijo mean
?min # pomoč za funkcijo min
```

Pomoč za skoraj vse funkcije (tudi tiste v paketih) je oblikovana na podoben način in vključuje naslednje točke:

- Description: opis namena funkcije.
- Usage: primer tipične uporabe funkcije.
- Arguments: seznam argumentov, ki jih funkcija sprejme, in pomen vseh argumentov.
- Details: podroben opis funkcije in argumentov.
- Value: informacija o vrsti rezultata določene funkcije.
- Author(s): avtor(-ji) posamezne funkcije.
- References: dodatna uporabna literatura, povezana z določeno funkcijo.
- Examples: tipični primeri uporabe posamezne funkcije.

Primer uporabe funkcije za izračun povprečja:

```
mean(c(3,5,3,5,3)) # izračunamo povprečje petih števil

## [1] 3.8
```

Zgoraj omenjena funkcija *mean* izračuna povprečje števil 3, 5, 3, 5, 3. Dodatno je uporabljena tudi funkcija *c*, ki omogoča združevanje posameznih števil v vektor (pomoč za funkcijo *?c* razloži primer uporabe). Še en dodaten primer je uporaba funkcije za zaokroževanje števil *round*:

```
round(x=3.54453445,digits=2) # zaokrožimo glede na vhodne podatke

## [1] 3.54
```

V primerjavi s primerom zgoraj vidite, da smo tukaj točno definirali tudi oba parametra oziroma argumenta, ki sta uporabljena v funkciji *round*, in sicer prvi parameter *x*, ki definira vhodni podatek v funkcijo, in parameter argument *digits*, ki definira število decimalnih mest rezultata. Če poznamo vrstni red parametrov oziroma argumentov, lahko imena argumentov izpustimo:

```
round(3.54453445,2) # zaokrožimo glede na vhodne podatke

## [1] 3.54
```

Če nismo prepričani o vrstnem redu argumentov oziroma parametrov, je bolje vsakega posebej definirati, saj lahko v nasprotnem primeru dobimo napačen rezultat:

```
round(2, 3.54453445) # obratna uporaba argumentov
```

```
## [1] 2
```

Definicija argumentov oziroma parametrov je v večini primerov smiselna, saj lahko pri uporabi napačnega vrstnega reda dobimo napačne rezultate:

```
round(digits=2, x=3.54453445) # zaokrožimo glede na vhodne podatke

## [1] 3.54
```

Za uporabo funkcij morate seveda poznati ime funkcije. Pri tem najbolj pomaga spletno iskanje najbolj smiselne funkcije za vaš primer oziroma problem, ki ga želite rešiti. Obstajajo pa tudi različne spletne strani, ki podajajo pregled najbolj pogosto uporabljenih funkcij[15].

V sklopu programskega jezika R skoraj ni statistično-matematičnega problema, za katerega v programskem okolju R ne bi obstajala funkcija v osnovnem paketu (*base*) R ali pa v katerem izmed več 10.000 razširitvenih paketov[16].

Spodaj so zapisane prve praktične naloge, ki jih lahko rešite in tako nadgradite svoje osnovno znanje o programu R.

Naloga 1: Poiščite funkcijo, s katero lahko izračunate standardno deviacijo v programskem orodju R, in funkcijo uporabite.

Naloga 2: Poiščite funkcijo, s katero izrišite graf v programskem orodju R (kakršen koli graf), in funkcijo uporabite.

Naloga 3: Poiščite funkcijo, s katero lahko generirate zaporedje števil, recimo 2, 4, 6, 8, 10 itd., in to funkcijo tudi uporabite za izračun zaporedja, ki naj ima 50 elementov, začetna vrednost je 2 in korak zaporedja prav tako 2.

Naloga 4: Poiščite funkcijo za zaokroževanje števil in zaokrožite število 2,464646 na 2 decimalni mesti.

3.4 Objekti

V programskem orodju R objekte definiramo z uporabo znaka <-, v grafičnem vmesniku RStudio do tega znaka pridemo tudi s sočasnim pritiskom tipk *ALT* in -.

```
x <- 1 # definiramo objekt x z enim elementom
```

Ob definiciji objekta R tega ne izpiše. Za izpis uporabimo funkcijo print ali pa zgolj ime objekta:

```
print(x) # izpišemo vsebino objekta z imenom x
```

---

[15] https://www.statmethods.net/management/functions.html.

[16] https://cran.r-project.org/web/packages/.

```
## [1] 1

x # enako kot zgoraj, a brez uporabe funkcije

## [1] 1
```

V programskem jeziku R je pogosto več načinov za rešitev enega problema oziroma za neko operacijo. Za definiranje objektov lahko na primer uporabimo tudi funkcijo *assign*:

```
assign("x", c(10,4,5)) # definiramo objekt z uporabo funkcije assign
c(4,5,3) -> x # objekte pa lahko definiramo tudi na ta način
```

V večini primerov lahko uporabimo tudi enačaj =.

Ko izpišete vektor, boste opazili, da je indeks vektorja natisnjen v oglatih oklepajih *[]* ob strani. Poglejmo si na primer številsko zaporedje dolžine 20. Številke v oglatih oklepajih niso del samega vektorja, temveč le del izpisa:

```
y <- 41:60 # generiramo cela števila med 41 in 60
y # preverimo vsebino objekta y

##  [1] 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60
```

R pozna pet osnovnih vrst objektov:

- character: elementi, sestavljeni iz nizov črk ali besed, npr. sosed, Marko.
- numeric: realna števila.
- integer: cela števila.
- complex: imaginarna ali kompleksna števila.
- logical: podatki z logičnimi vrednostmi, kot sta FALSE in TRUE.

Na prvi pogled med *integer* in *numeric* ni bistvene razlike. Številke v R se na splošno obravnavajo kot numerični objekti (tj. realna števila z dvojno natančnostjo). To pomeni, da tudi če v R vidite število, kot je 5 ali 10, ki bi ga lahko imeli za celo število, je v ozadju to verjetno predstavljeno kot numerični objekt (torej nekaj takega kot 5,00 ali 10,00). Če želite definirati celo število, morate za številom dodati črko *L*. Tako boste z vnosom *10* dobili numerični objekt, z vnosom *10L* pa objekt tipa *integer*.

```
z1 <- 10 # definiramo objekt z1
z2 <- 10L # definiramo objekt z2
str(z1) # preverimo strukturo objekta z1

##  num 10

str(z2) # preverimo strukturo objekta z2

##  int 10

is(z2) # malce drugačen način kontrole strukture objekta z2

## [1] "integer"             "double"              "numeric"
## [4] "vector"              "data.frameRowLabels"
```

Za običajnega uporabnika med obema vrstama objektov ni bistvene razlike, je pa razlika pomembna pri zelo velikih objektih, kjer objekti tipa *integer* zavzamejo veliko manj prostora kot objekti tipa *numeric*:

```
# preverimo velikost vektorja, tip integer
object.size(as.integer(seq(from=1,by=2,length.out=100000)))

## 400048 bytes

# preverimo velikost vektorja, tip numeric
object.size(seq(from=1,by=2,length.out=100000))

## 800048 bytes
```

Obstaja tudi posebno število *Inf*, ki predstavlja neskončnost. To nam omogoča, da določimo izraze, kot je 1/0. Tako lahko *Inf* uporabimo pri običajnih izračunih; npr. *1/Inf* je *0*. Vrednost *NaN* predstavlja »not a number« (ni število); npr. *0/0*. V programskem jeziku R se pogosto uporablja tudi izraz *NA*, ki označuje manjkajočo vrednost (angl. *Not Available*). Primer uporabe je prikazan v nadaljevanju:

```
z3 <- c(4,5,Inf,1/0,NA,10) # definiramo vektor
# vse elemente vektorja delimo z vrednostjo 10
z3/10

## [1] 0.4 0.5 Inf Inf  NA 1.0
```

R pozna kompleksna števila, kot je razvidno iz primera zgoraj. Bolj kompleksne definicije kompleksnih števil (npr. v polarni obliki) dobimo z ukazom *complex*. Omeniti velja še nekatere druge potencialno uporabne funkcije, povezane s kompleksnimi števili: *Re*, *Im*, *Mod*, *Arg*, *Conj*. Ker se kompleksna števila na področju vodarstva večinoma ne uporabljajo pogosto, se na tem mestu ne bomo spuščali v podrobnosti.

Najosnovnejša vrsta objekta v programskem okolju R je vektor. Prazni vektor lahko ustvarite s funkcijo *vector*. Za vektorje v R velja pravilo, da lahko vektor vsebuje samo elemente istega tipa, izjema pa je tip objekta list, ki lahko združuje različne vrste.

V programu R lahko definiramo tudi različne tipe vektorjev:

```
x <- c(0.4, 0.7) # numerični
# logični, namesto TRUE bi lahko uporabili T ali namesto FALSE F
x <- c(TRUE, FALSE)
x <- c("Marko","Jana","Vid") # besedilni
x <- 2:10 # celo število
x <- c(2+0i, 4+2i) # kompleksno število
```

Občasno se zgodi, da se različni tipi objektov v programskem okolju R pomešajo. Včasih se to zgodi po naključju, lahko pa tudi namerno. Primer:

```
c(3.4, "Zvone") # besedilo

## [1] "3.4"   "Zvone"

c(FALSE,3) # numerična vrednost
```

```
## [1] 0 3

c("Stanko", TRUE) # besedilo

## [1] "Stanko" "TRUE"
```

V vsakem od zgornjih primerov v vektorju mešamo predmete dveh različnih tipov. Vendar ne pozabimo, da edino pravilo o vektorjih pravi, da to ni dovoljeno. Ko v vektorju mešamo tipe podatkov, pride do spremembe (nižji tipi se pretvorijo v višje tipe), tako da je vsak element v vektorju istega razreda. V zgornjem primeru vidimo učinek posamezne spremembe. R poskuša najti način, kako bi razumno predstavil vse predmete v skupnem vektorju. Včasih naredi točno to, kar želite, včasih pa ne. Če na primer združite številsko vrednost z besedilnim tipom, bo nastal besedilni vektor, saj je številke običajno mogoče predstaviti tudi v tej obliki. V vsakem primeru pa se je smiselno tem kombinacijam izogniti. Tip objekta lahko tudi spremenimo z uporabo funkcij, kot so *as.numeric*, *as.integer*, *as.logical* ali *as.character*:

```
z2 <- 1:10 # generiramo vektor celih števil med 1 in 10
z3 <- as.character(z2) # spremenimo tip objekta v besedilnega
str(z3) # struktura objekta z3

##  chr [1:10] "1" "2" "3" "4" "5" "6" "7" "8" "9" "10"

z3 # vsebina objekta z3

##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10"
```

Do posameznih elementov objekta pridemo z uporabo oglatih oklepajev *[]* in vrstnega reda elementa v vektorju, kjer se štetje vedno začne z 1:

```
z6 <- 1:10 # generiramo vektor celih števil med 1 in 10
z6[3:4] # zanimata nas samo 3. in 4. element

## [1] 3 4

z6[c(2,8,10)] # zanimajo nas 2., 8. in 10. element

## [1]  2  8 10

z6[c(1, 3:5, 9)] # zanimajo nas določeni elementi

## [1] 1 3 4 5 9

z6[-2] # zanimajo nas vsi elementi z izjemo drugega

## [1]  1  3  4  5  6  7  8  9 10

z6[5] <- 1000 # izbrane elemente lahko tudi zamenjamo
z6[c(3,8)] <- c(100,200) # zamenjamo lahko tudi več elementov hkrati
```

Pri indeksiranju oziroma izbiri določenih elementov so uporabne tudi naslednje funkcije:

```
1 %in% z6 # preverimo, ali je 1 vsebovana v objektu z6
```

```
## [1] TRUE
```

```
2 == 2 # kontrola, ali sta dva elementa enaka
```

```
## [1] TRUE
```

```
2 == 4 # kontrola, ali sta dva elementa enaka
```

```
## [1] FALSE
```

```
3 != 4 # kontrola, ali dva elementa nista enaka
```

```
## [1] TRUE
```

```
# možne so tudi bolj kompleksne kombinacije, & označuje pogoj "in"
10 <= 20 & 22 >= 22
```

```
## [1] TRUE
```

```
10 <= 20 | 22 >= 22 # | označuje pogoj "ali"
```

```
## [1] TRUE
```

```
v7 <- 15:21 # generiramo cela števila med 15 in 21
v7 > 18 # preverimo, kateri elementi so večji od 18
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE  TRUE  TRUE
```

```
v7 == 17 # preverimo, kateri element je enak 17
```

```
## [1] FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
```

```
v7 != 20 # preverimo, kateri elementi so različni od 20
```

```
## [1]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
```

```
v7[v7 > 18] # elementi, ki ustrezajo izbranemu pogoju
```

```
## [1] 19 20 21
```

```
v7[v7 > 18 & v7 < 22] # uporaba dveh pogojev hkrati
```

```
## [1] 19 20 21
```

```
v7[v7 > 18 | v7 < 22] # malce drugače
```

```
## [1] 15 16 17 18 19 20 21
```

```
which(v7 == 17) # vrstni red elementa
```

```
## [1] 3
```

V primeru velikih objektov sta lahko uporabni tudi funkciji *head* in *tail*, ki prikažeta *n* prvih oziroma zadnjih elementov objekta:

```
v8 <- 1:10000 # generiramo velik objekt celih števil
head(x=v8, n=3) # preverimo prve tri elemente objekta v8

## [1] 1 2 3

tail(x=v8, n=4) # poglejmo zadnje štiri elemente objekta v8

## [1]  9997  9998  9999 10000
```

Pogosto pa nam prav pridejo tudi funkcije, kot so *subset*, *is.na*, *is.nan*, *summary* ter *na.rm*:

```
v9 <- c(4,NA,2,3,NA,10) # definiramo vektor
is.na(v9) # preverimo, kateri elementi niso definirani

## [1] FALSE  TRUE FALSE FALSE  TRUE FALSE

summary(v9) # izračunamo osnovne statistike in preverimo

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    2.00    2.75    3.50    4.75    5.50   10.00       2

# kateri elementi imajo vrednost NA
max(v9) # zaradi vrednosti NA funkcija ne deluje

## [1] NA

max(v9, na.rm=TRUE) # elementov NA ne upoštevamo pri izračunu

## [1] 10

v10 <- subset(v9, is.na(v9)!=TRUE) # shranimo elemente, ki so različni od NA
v11 <- subset(v10, v10 > 2 & v10 < 10) # izberemo samo določene elemente
v11

## [1] 4 3
```

Seznam definiranih objektov lahko vidite v desnem zgornjem oknu programa RStudio (zavihek Environment). Za pregled vseh definiranih objektov lahko uporabite tudi funkcijo *ls()*. Če želite določen objekt odstraniti, lahko to naredite z uporabo funkcij *rm* ali *remove*:

```
z6 <- 1:10 # generiramo vektor celih števil med 1 in 10
rm(z6) # objekt z6 odstranimo
```

Če želite odstraniti vse objekte, pa lahko uporabite naslednjo kombinacijo funkcij:

```
rm(list=ls(all=TRUE))
```

Objekti v programskem jeziku R imajo lahko atribute, ki so kot metapodatki objekta. Ti metapodatki so lahko zelo uporabni, saj pomagajo opisati objekt. Na primer, imena stolpcev v podatkovnem okviru (več informacij je v poglavju 3.7) nam pomagajo razložiti, kateri podatki so vsebovani v vsakem od stolpcev. Nekateri primeri atributov objekta R so: imena, imena stolpcev, dimenzije, tip objekta, dolžina. Določeni tipi objektov vsebujejo atribute, do katerih se lahko dostopa z uporabo funkcije *attributes* (v omenjenem primeru gre za uporabo

seta podatkov, ki je povezan s programskim orodjem R in do katerega lahko dostopamo z uporabo funkcije *data*):

```
data("airquality") # uvozimo podatke z imenom airquality
attributes(airquality) # preverimo atribute tega objekta

## $names
## [1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
##
## $class
## [1] "data.frame"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
18
##  [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35
36
##  [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53
54
##  [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71
72
##  [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
90
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107
108
## [109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126
## [127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143
144
## [145] 145 146 147 148 149 150 151 152 153
```

Naloga 5: Recimo, da izvajamo meritve padavin vsak dan v tednu (od ponedeljka do nedelje) na dveh padavinskih postajah. Na prvi postaji smo izmerili 10, 0, 0, 20, 15, 10, 5 mm padavin, na drugi postaji pa 5, 0, 0, 25, 10, 5, 10 mm padavin. Podatke shranite v dva ločena objekta. Nato z uporabo objektov izračunajte vrednosti padavin v vseh sedmih dneh: izračunajte povprečne padavine z upoštevanjem obeh postaj (postaja1 in postaja2), izračunajte povprečne padavine v celem tednu (za vse dni skupaj), izračunajte največje dnevne in najmanjše dnevne padavine glede na meritve z obeh postaj, izračunajte razpon izmerjenih vrednosti (range), mediano in standardno deviacijo za vsako padavinsko postajo.

Naloga 6: Poiščite funkcijo za generiranje naključnih števil (na podlagi enakomerne porazdelitve) in generirajte 10 naključnih števil v razponu med 1 in 35. Rezultate shranite v nov objekt in zaokrožite na eno decimalno mesto. Izračunajte vsoto in zmnožek vseh generiranih števil in razvrstite generirane vrednosti od najmanjše do največje.

Naloga 7: Izračunajte seštevek vseh celih števil med 1 in 10.000.

Naloga 8: Na primeru s padavinami iz naloge 5 preverite, v katerih dneh na postaji 1 je povprečno padlo več kot 10 mm padavin in manj kot 20 mm padavin.

Naloga 9: Naknadno je bilo ugotovljeno, da je bila izmerjena količina padavin za postajo 1 v četrtek napačna in bi morala znašati 30 mm. Preverite, ali ta sprememba vpliva na rezultat naloge 8.

3.5 Matrike

Matrike so vektorji z atributom dimenzije. Atribut dimenzije je celoštevilski vektor dolžine 2 (število vrstic, število stolpcev).

```
m1 <- matrix(nrow = 2, ncol = 3) # definiranje matrike, z vrednostmi NA
dim(m1) # dimenzije matrike

## [1] 2 3

m1 # vsebina

##      [,1] [,2] [,3]
## [1,]   NA   NA   NA
## [2,]   NA   NA   NA

attributes(m1) # atributi

## $dim
## [1] 2 3
```

Matrike so definirane po stolpcih, zato si lahko vnose predstavljamo tako, da se začnejo v »zgornjem levem« kotu (element 1,1) in se polnijo po stolpcih navzdol.

```
# definiramo matriko, ki vsebuje cela števila med 1 in 6
m2 <- matrix(1:6, nrow = 3, ncol = 2)
m2 # vsebina

##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6

dim(m2) <- c(2,3) # spremenimo dimenzije
m2 # vsebina preoblikovane matrike

##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

Matrike lahko definiramo tudi z združevanjem stolpcev ali vrstic s funkcijama *cbind* in *rbind*. V nekaterih primerih lahko uporabimo tudi funkcijo *array*:

```
m4 <- cbind (4:6,1:3) # združimo dva stolpca vektorjev s tremi elementi
m4 # vsebina
```

```
##      [,1] [,2]
## [1,]    4    1
## [2,]    5    2
## [3,]    6    3
```

```
m5 <- rbind(10:11, 20:21) # združimo dve vrstici v matriko dimenzij 2 × 2
m5 # vsebina
```

```
##      [,1] [,2]
## [1,]   10   11
## [2,]   20   21
```

```
m6 <- array(1:12,dim=c(3,4)) # definicija matrike z uporabo funkcije array
```

Za uporabo določenih elementov matrike lahko uporabimo matrično sklicevanje:

```
m4 <- cbind (4:6,1:3) # združimo dva stolpca vektorjev s tremi elementi
m4[,2] # drugi stolpec
```

```
## [1] 1 2 3
```

```
m4[1,] # prva vrstica
```

```
## [1] 4 1
```

```
m4[2,2] # element v drugem stolpcu in drugi vrstici
```

```
## [1] 2
```

```
m4[c(1,3),1] # elementa v prvi in tretji vrstici prvega stolpca
```

```
## [1] 4 6
```

```
m4[5] # peti element matrike, pri čemer so elementi urejeni po stolpcih
```

```
## [1] 2
```

```
c(m4) # matriko lahko preoblikujemo tudi v vektor, vrstni red po stolpcih
```

```
## [1] 4 5 6 1 2 3
```

```
as.vector(m4) # enako kot prej, a z uporabo funkcije as.vector
```

```
## [1] 4 5 6 1 2 3
```

Z matrikami lahko tudi računamo:

```
m4 <- cbind (4:6,1:3) # združimo dva stolpca vektorjev s tremi elementi
m5 <- m4*10 # vse elemente matrike m4 pomnožimo z 10
m5 <- log(m5) # za vse elemente izračunamo logaritem
m6 <- m4*m5 # matrike lahko množimo
print(m6) # rezultat množenja
```

```
##          [,1]     [,2]
## [1,] 14.75552  2.302585
```

```
## [2,] 19.56012  5.991465
## [3,] 24.56607 10.203592

crossprod(m4,m5) # matrično množenje

##          [,1]     [,2]
## [1,] 58.88170 44.59619
## [2,] 23.79596 18.49764

m4 %o% m5 # tenzorski produkt, alternativa je funkcija outer

## , , 1, 1
##
##          [,1]      [,2]
## [1,] 14.75552  3.688879
## [2,] 18.44440  7.377759
## [3,] 22.13328 11.066638
##
## , , 2, 1
##
##          [,1]      [,2]
## [1,] 15.64809  3.912023
## [2,] 19.56012  7.824046
## [3,] 23.47214 11.736069
##
## , , 3, 1
##
##          [,1]      [,2]
## [1,] 16.37738  4.094345
## [2,] 20.47172  8.188689
## [3,] 24.56607 12.283034
##
## , , 1, 2
##
##          [,1]     [,2]
## [1,]  9.21034 2.302585
## [2,] 11.51293 4.605170
## [3,] 13.81551 6.907755
##
## , , 2, 2
##
##          [,1]     [,2]
## [1,] 11.98293 2.995732
## [2,] 14.97866 5.991465
## [3,] 17.97439 8.987197
##
## , , 3, 2
##
##          [,1]      [,2]
## [1,] 13.60479  3.401197
```

```
## [2,] 17.00599  6.802395
## [3,] 20.40718 10.203592

t(m6) # transponiranje matrike

##           [,1]      [,2]     [,3]
## [1,] 14.755518 19.560115 24.56607
## [2,]  2.302585  5.991465 10.20359

m7 <- cbind(1:2,4:5) # definiramo kvadratno matriko
det(m7) # izračunamo determinanto

## [1] -3

# lastne vrednosti in vektorji v primeru, da matriko lahko diagonaliziramo
eigen(m7)

## eigen() decomposition
## $values
## [1]  6.4641016 -0.4641016
##
## $vectors
##            [,1]       [,2]
## [1,] -0.5906905 -0.9390708
## [2,] -0.8068982  0.3437238

dim(m6) # dimenzije matrike

## [1] 3 2
```

Za izračune v primeru večdimenzijskih objektov so zelo uporabne tudi funkcije *apply*, *sapply*, *mapply*, itd. Poglejmo primer:

```
m8 <- cbind (10:20,20:30) # združimo dva stolpca vektorjev
# izračunamo povprečje po stolpcih, MARGIN argument definira
# ali uporabimo stolpce ali vrstice
apply(X = m8, MARGIN = 2,FUN = mean)

## [1] 15 25

apply(X = m8, MARGIN = 1,FUN = mean) # izračunamo povprečje po vrsticah

##  [1] 15 16 17 18 19 20 21 22 23 24 25

apply(m8, 2, summary) # izračunamo glavne opisne statistike po stolpcih

##         [,1] [,2]
## Min.    10.0 20.0
## 1st Qu. 12.5 22.5
## Median  15.0 25.0
## Mean    15.0 25.0
## 3rd Qu. 17.5 27.5
## Max.    20.0 30.0
```

Naloga 10: Podatke o padavinah z dveh postaj, ki so bili podani v sklopu naloge 5, združite v matriko. Dodajte tretji stolpec, kjer so prikazane povprečne vrednosti padavin na obeh postajah (po dnevih). Dodajte oznake vrstic (imena dni v tednu) in stolpcev (postaja 1, postaja 2, povprečje) z uporabo funkcij colnames in rownames.

Naloga 11: Poskusite uporabiti funkcijo apply na prej definirani matriki (naloga 10) za izračun povprečnih vrednosti na postaji 1, postaji 2 in povprečnih vrednosti.

3.6 Faktorji

Faktorji se uporabljajo za predstavitev kategoričnih podatkov in so lahko neurejeni ali urejeni. Faktor si lahko predstavljamo kot celoštevilski vektor, kjer ima vsako celo število oznako. Faktorji so pomembni pri statističnih analizah in jih posebej obravnavajo tudi določene funkcije, kot sta *lm* in *glm*. Uporaba faktorjev z oznakami je boljša kot uporaba celih števil, ker se faktorji sami opisujejo. Spremenljivka, ki ima vrednosti »moški« in »ženska«, je v nekaterih primerih boljša od spremenljivke, ki ima vrednosti 1 in 2. Objekte faktorjev lahko definiramo s funkcijo *factor*. Pogosto se faktorji samodejno določijo, ko preberete niz podatkov s funkcijo, kot je *read.table*. Te funkcije pogosto privzeto ustvarijo faktorje, ko zaznajo podatke, ki so videti kot znaki ali nizi. Vrstni red stopenj faktorja lahko določite z argumentom *levels* v funkciji *factor*:

```
opa <- c("M", "F", "F", "M", "M") # definiramo vektor opazovanj
opafak <- factor(opa) # preoblikujemo objekt v faktor
levels(opafak) # poglejmo, kateri tipi so vključeni v naš objekt

## [1] "F" "M"

table(opafak) # poglejmo, koliko elementov določene vrste imamo

## opafak
## F M
## 2 3

summary(opafak) # podobno kot zgoraj

## F M
## 2 3

unclass(opafak) # preoblikujemo v numerične vrednosti

## [1] 2 1 1 2 2
## attr(,"levels")
## [1] "F" "M"

# definiramo vrstni red stopenj, levels
f1 <- factor(c("yes", "yes", "no", "yes", "no"), levels = c("yes", "no"))
table(f1) # struktura
```

```
## f1
## yes  no
##   3   2
```

3.7 Podatkovni okvirji

Podatkovni okvirji se uporabljajo za shranjevanje tabelaričnih podatkov v R. So pomembna vrsta objektov v programskem orodju R in se uporabljajo za različne namene. Na primer, paket *dplyr* ima optimiziran nabor funkcij in je zasnovan za učinkovito delo s podatkovnimi okvirji. Podatkovni okvirji so posebna vrsta seznama (Poglavje 3.8) ali matrike, kjer mora biti vsak element seznama enako dolg. Vsak element seznama si lahko predstavljamo kot stolpec, dolžina vsakega elementa seznama pa je število vrstic. Za razliko od matrik lahko podatkovni okvirji v vsakem stolpcu shranjujejo različne razrede predmetov. Pri matrikah mora biti vsak element istega razreda (npr. numerične vrednosti). Poleg imen stolpcev, ki označujejo imena spremenljivk, imajo podatkovni okvirji poseben atribut, imenovan *row.names*, ki označuje informacije o vsaki vrstici podatkovnega okvirja. Podatkovni okvirji se običajno definirajo z branjem podatkov z uporabo funkcije *read.table* ali *read.csv*. Vendar lahko podatkovne okvirje ustvarimo tudi neposredno z uporabo funkcije *data.frame* ali jih preoblikujemo iz drugih obstoječih objektov, kot so matrike. Podatkovne okvirje je mogoče pretvoriti v matriko s funkcijo *data.matrix.* Čeprav se včasih zdi, da je treba za pretvorbo podatkovnega okvirja v matriko uporabiti funkcijo *as.matrix*, je večinoma funkcija *data.matrix* boljša izbira.

```
# struktura podatkovnega okvirja, ki je vključen v programsko okolje R
str(mtcars)

## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...

nrow(mtcars) # število vrstic

## [1] 32

ncol(mtcars) # število stolpcev

## [1] 11

summary(mtcars) # osnovna statistika

##       mpg             cyl             disp             hp
##  Min.   :10.40   Min.   :4.000   Min.   : 71.1   Min.   : 52.0
##  1st Qu.:15.43   1st Qu.:4.000   1st Qu.:120.8   1st Qu.: 96.5
```

```
##  Median :19.20   Median :6.000   Median :196.3   Median :123.0
##  Mean   :20.09   Mean   :6.188   Mean   :230.7   Mean   :146.7
##  3rd Qu.:22.80   3rd Qu.:8.000   3rd Qu.:326.0   3rd Qu.:180.0
##  Max.   :33.90   Max.   :8.000   Max.   :472.0   Max.   :335.0
##       drat             wt             qsec             vs
##  Min.   :2.760   Min.   :1.513   Min.   :14.50   Min.   :0.0000
##  1st Qu.:3.080   1st Qu.:2.581   1st Qu.:16.89   1st Qu.:0.0000
##  Median :3.695   Median :3.325   Median :17.71   Median :0.0000
##  Mean   :3.597   Mean   :3.217   Mean   :17.85   Mean   :0.4375
##  3rd Qu.:3.920   3rd Qu.:3.610   3rd Qu.:18.90   3rd Qu.:1.0000
##  Max.   :4.930   Max.   :5.424   Max.   :22.90   Max.   :1.0000
##        am              gear            carb
##  Min.   :0.0000   Min.   :3.000   Min.   :1.000
##  1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:2.000
##  Median :0.0000   Median :4.000   Median :2.000
##  Mean   :0.4062   Mean   :3.688   Mean   :2.812
##  3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:4.000
##  Max.   :1.0000   Max.   :5.000   Max.   :8.000

colnames(mtcars) # imena stolpcev

##  [1] "mpg"  "cyl"  "disp" "hp"   "drat" "wt"   "qsec" "vs"   "am"   "gear"
## [11] "carb"

mtcars$hp # poglejmo vsebino stolpca z imenom hp

##  [1] 110 110  93 110 175 105 245  62  95 123 123 180 180 180 205 215 230
66  52
## [20]  65  97 150 150 245 175  66  91 113 264 175 335 109

mtcars$hp[2:5] # nekaj elementov tega stolpca

## [1] 110  93 110 175

mtcars[3:5,5] # uporabimo pa lahko tudi matrično sklicevanje

## [1] 3.85 3.08 3.15

mtcars[1:4, "hp"] # malce drugače način

## [1] 110 110  93 110
```

Definiramo pa lahko tudi svoje podatkovne okvirje:

```
# definiramo podatkovni okvir
primer <- data.frame(temp=c("visoka","nizka","visoka"),
pretok=c("velik","majhen","velik"),
motnost=c("velika","srednja","velika"),
vodostaj=c(30,10,28))
print(primer) # pogledamo strukturo

##     temp pretok motnost vodostaj
## 1 visoka  velik  velika       30
```

```
## 2  nizka majhen srednja       10
## 3 visoka  velik  velika       28
```

```
primer$pretok # poglejmo samo stolpec z imenom pretok
```

```
## [1] "velik"  "majhen" "velik"
```

Naloga 12: Na podlagi objekta airquality (za uporabo podatkov potrebujete funkcijo data(airquality)) preverite, katere dni je bila hitrost vetra večja od 10 mph (enota, v kateri je podana hitrost vetra), preverite, katere dni je bila temperatura zraka med 60 in 70 F (enota, v kateri je podana temperatura zraka) in kateri dan je bila izmerjena največja ter najmanjša koncentracija ozona.

Naloga 13: Razvrstite podatke airquality glede na izmerjeno temperaturo zraka. Preverite delovanje funkcij order in sort.

Naloga 14: Z uporabo funkcije apply izračunajte povprečje vseh stolpcev v objektu airquality.

3.8 Seznami

Seznami (*list*) so posebna vrsta vektorjev, ki lahko vsebujejo elemente različnih razredov. Seznami so zelo pomembna podatkovna vrsta v programskem okolju R, saj lahko v njih shranite veliko različnih podatkov. Seznami v kombinaciji z različnimi funkcijami, kot so *apply*, *sapply* ali *lapply*, omogočajo hitre in enostavne izračune na podlagi velike količine podatkov. Sezname lahko definiramo s funkcijo *list*, ki sprejme poljubno število argumentov:

```
# definiramo poljuben seznam
clovek <- list(ime="Janez", starost=35, zakonec="Marija",
starost_otrok=c(15, 13, 2))
clovek$starost_otrok # poglejmo vsebino elementa z imenom starost_otrok
```

```
## [1] 15 13  2
```

```
clovek[["ime"]] # uporabimo lahko tudi takšen način
```

```
## [1] "Janez"
```

```
clovek[c("ime","zakonec")] # ali pa več elementov
```

```
## $ime
## [1] "Janez"
##
## $zakonec
## [1] "Marija"
```

```
names(clovek) # imena lahko preverimo z uporabo funkcije names
```

```
## [1] "ime"          "starost"      "zakonec"      "starost_otrok"
```

Naloga 15: Definirajte nov objekt v obliki seznama (list), kjer združite dva poljubna stolpca objekta airquality in povprečne padavine, ki ste jih obravnavali v okviru naloge 5.

3.9 Uvoz in shranjevanje podatkov

Obstaja nekaj glavnih funkcij za uvoz podatkov v R:

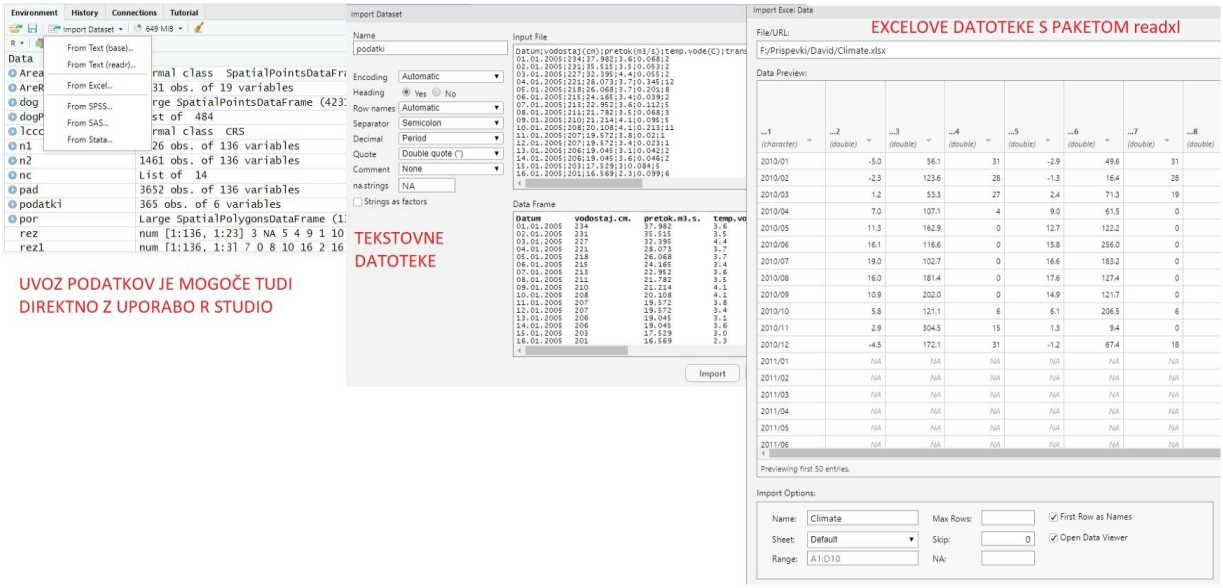
- *read.table*, *read.csv* za uvoz tabelaričnih podatkov;
- *readLines* za branje vrstic v besedilni datoteki;
- *readChar* za branje določenega števila znakov oblike *character*;
- *read_excel* za branje Excelovih datotek;
- *source* za branje datotek s kodo R (obratno od *dump*);
- *dget* za branje kodnih datotek R (obratno od *dput*);
- *load* za branje shranjenih delovnih prostorov;
- *scan* za branje datotek, rezultat objekta je vektor.

Obstaja veliko paketov R, ki so bili razviti za uvoz različnih vrst podatkov, npr. *readxl* za uvoz Excelovih preglednic ali *read_sav* (paket *haven*) za branje podatkovnih baz SPSS.

Za shranjevanje podatkov v datoteke zunaj programskega okolja R obstajajo analogne funkcije, kot so:

- *write.table*, *write.csv* za shranjevanje tabelaričnih podatkov v besedilne datoteke (npr. csv);
- *writeLines* za shranjevanje podatkov po vrsticah v datoteko;
- *dump* za shranjevanje tekstualnih podatkov v povezavi z različnimi objekti R;
- *dput* za shranjevanje besedilne predstavitve objekta R;
- *save* za shranjevanje poljubnega števila objektov R (v binarni, stisnjeni obliki) v datoteko.

Funkcija *read.table* je ena najpogosteje uporabljenih funkcij za uvoz podatkov v programsko okolje R. Pomoč za funkcijo *read.table* je vredno prebrati v celoti, saj se ta funkcija pogosto uporablja in omogoča številne nastavitve, s katerimi zagotovite, da se bodo vaši podatki prebrali v pravilni obliki (npr. ustrezna izbira decimalnega ločila (argument *dec*), izbira znaka, ki ločuje stolpce (argument *sep*)). Uvoz podatkov lahko poteka tudi neposredno preko grafičnega vmesnika RStudio (z uporabo funkcij v zavihku *Environment*):



Slika 3: Primer uvoza podatkov preko grafičnega vmesnika RStudio.

Kot primer bomo pokazali postopek uvoza podatkov z vodomerne postaje Veliko Širje na reki Savinji, ki so bili izmerjeni v letu 2005. Podatki so bili pridobljeni s spletne strani Agencije RS za okolje (ARSO)[17]. Podatki so odloženi na povezavi OneDrive[18]. Definiramo argumente funkcije *read.table* za uvoz podatkov, definirali bomo lokacijo datoteke (to v svojem primeru spremenite glede na mapo, kamor boste shranili podatke), ločilo stolpcev, uporabljeni decimalni simbol in to, da se prva vrstica prebere kot glava datoteke (*header*).

```
podatki <- read.table(file="C:/Users/nbezak/OneDrive - Univerza v Ljubljani/U
cbenik/Savinja-Veliko SirjeI-2005.txt",header=TRUE,sep=";",dec=".")
head(podatki) # preverimo prvih nekaj vrstic
```

```
##        Datum vodostaj.cm. pretok.m3.s. temp.vode.C.
## 1 01.01.2005          234       37.982          3.6
## 2 02.01.2005          231       35.515          3.5
## 3 03.01.2005          227       32.395          4.4
## 4 04.01.2005          221       28.073          3.7
## 5 05.01.2005          218       26.068          3.7
## 6 06.01.2005          215       24.165          3.4
##   transport_suspendiranega_materiala.kg.s.
## 1                                    0.068
## 2                                    0.053
## 3                                    0.055
## 4                                    0.345
```

[17] https://vode.arso.gov.si/hidarhiv/pov_arhiv_tab.php.

[18] https://unilj-my.sharepoint.com/:f:/g/personal/nbezak_fgg_uni-lj_si/EgiFWdB01CtEldvCU3XX844BrrqoodVBoTWiV-76eufZaA?e=YGyKsk.

```
## 5                                      0.201
## 6                                      0.039
##   vsebnost_suspendiranega_materiala.g.m3.
## 1                                       2
## 2                                       2
## 3                                       2
## 4                                      12
## 5                                       8
## 6                                       2
```

```
str(podatki) # preverimo strukturo prebranih podatkov
```

```
## 'data.frame':    365 obs. of  6 variables:
##  $ Datum                                  : chr  "01.01.2005" "02.01.2005
" "03.01.2005" "04.01.2005" ...
##  $ vodostaj.cm.                           : int  234 231 227 221 218 215
213 211 210 208 ...
##  $ pretok.m3.s.                           : num  38 35.5 32.4 28.1 26.1 .
..
##  $ temp.vode.C.                           : num  3.6 3.5 4.4 3.7 3.7 3.4
3.6 3.5 4.1 4.1 ...
##  $ transport_suspendiranega_materiala.kg.s.: num  0.068 0.053 0.055 0.345
0.201 0.039 0.112 0.068 0.095 0.213 ...
##  $ vsebnost_suspendiranega_materiala.g.m3. : int  2 2 2 12 8 2 5 3 5 11 ..
.
```

```
# vidimo lahko, da so bili datumi prebrani kot znaki - character
names(podatki) # preverimo imena stolpcev
```

```
## [1] "Datum"
## [2] "vodostaj.cm."
## [3] "pretok.m3.s."
## [4] "temp.vode.C."
## [5] "transport_suspendiranega_materiala.kg.s."
## [6] "vsebnost_suspendiranega_materiala.g.m3."
```

```
names(podatki) <- c("Datum", "Vodostaj", "Pretok", "Temperatura", "Transport"
, "Vsebnost") # spremenimo imena stolpcev
names(podatki) # še enkrat preverimo spremenjena imena
```

```
## [1] "Datum"       "Vodostaj"    "Pretok"      "Temperatura" "Transport"
## [6] "Vsebnost"
```

Naloga 16: Izračunajte povprečne vrednosti vseh petih spremenljivk, ki so v datoteki s podatki vodomerne postaje Veliko Širje na reki Savinji (za leto 2005).

Naloga 17: V programsko okolje R uvozite poljubne podatke, ki ste jih že kdaj uporabili.

Naloga 18: Shranite poljubni objekt v format .Rdata in ga nato naložite nazaj v R z uporabo funkcije load.

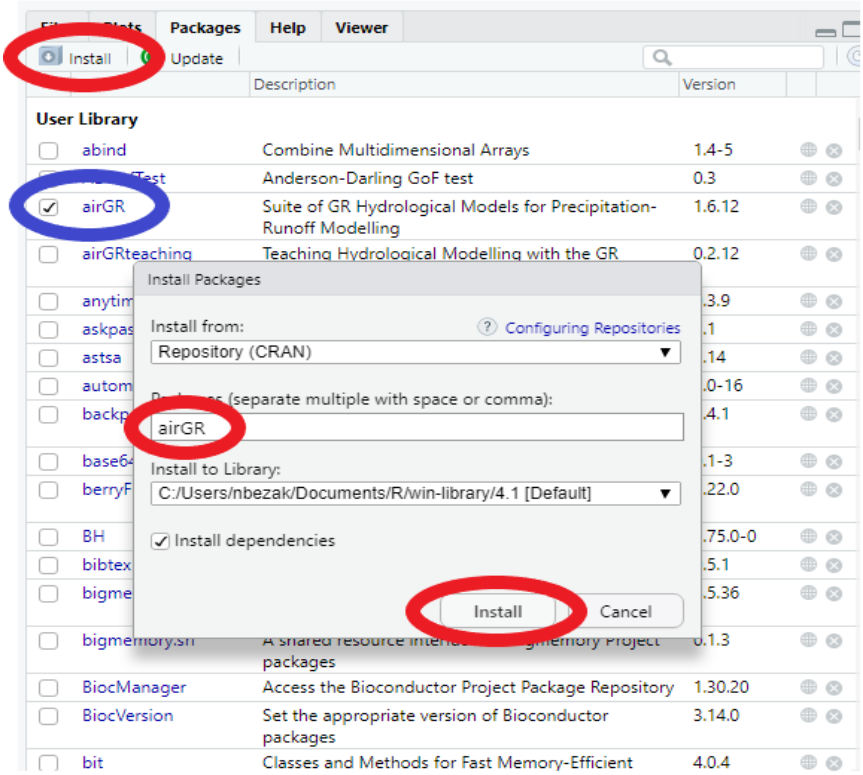
3.10 Paketi

Paketi v programskem orodju R razširjajo funkcionalnost programa z zagotavljanjem dodatnih funkcij, podatkov in dokumentacije. Pakete pripravlja svetovna skupnost uporabnikov R in jih je mogoče brezplačno prenesti z interneta. Paketi R so neke vrste aplikacije na mobilnem telefonu. Za uporabo funkcij, ki so na voljo v posameznem paketu, je treba izvesti naslednja dva koraka:

- Namestitev paketa: Ta je podobna namestitvi aplikacije na telefonu. Večina paketov ni privzeto nameščenih, ko namestite R in RStudio. Če torej želite paket uporabiti prvič, ga morate najprej namestiti. Ko paket enkrat namestite, ga verjetno ne boste več namestili, razen če ga želite posodobiti na novejšo različico (ali pa če ga boste vmes pomotoma odstranili).

- Aktivacija paketa: Aktivacija paketa je podobna kot odpiranje aplikacije na telefonu. Paketi niso privzeto aktivirani ob zagonu programa R ali RStudio. Vsak paket, ki ga želite uporabljati, morate aktivirati ob vsakem zagonu programa RStudio.

V programskem okolju R lahko paket namestite na dva načina: ali preko uporabe grafičnega vmesnika RStudio ali pa neposredno preko uporabe funkcije *install.packages*. Za namestitev preko grafičnega vmesnika lahko uporabimo naslednje okno (če je določen paket aktiviran, je to vidno s kljukico v belem kvadratu pred imenom paketa, s klikom na paket pa pridete do strani za pomoč pri uporabi posameznega paketa, kar je na voljo po namestitvi paketa):



Slika 4: Primer namestitve paketa preko grafičnega vmesnika RStudio. Z rdečo so prikazani posamezni koraki namestitve, z modro pa, ali je določen paket aktiviran ali ne.

Alternativni postopek namestitve (in aktivacije) paketa:

```
install.packages("airGR") # namestitev paketa z imenom "airGR"
library(airGR, quietly=TRUE) # aktivacija paketa
```

Paketi pogosto vsebujejo tudi določene podatke, ki služijo kot testni primer za boljše razumevanje delovanja paketa. Recimo paket *airGR* vsebuje tudi hidrološke podatke, ki se jih lahko uporabi kot primer za umerjanje, validacijo in zagon hidrološkega modela padavine–odtok, ki je vključen v ta paket in s katerim se bomo spoznali v nadaljevanju (Poglavje 4). Opis teh podatkov si lahko ogledate v pomoči za funkcijo *BasinObs* (uporabite *?BasinObs*).

```
library(airGR, quietly=TRUE) # aktivacija paketa
data(L0123001) # nalaganje podatkov z imenom L0123001
str(BasinObs) # pregled osnovnih značilnosti

## 'data.frame':    10593 obs. of  6 variables:
##  $ DatesR: POSIXct, format: "1984-01-01" "1984-01-02" ...
##  $ P     : num  4.1 15.9 0.8 0 0 0 0 0 2.9 0 ...
##  $ T     : num  0.5 0.2 0.9 0.5 -1.6 0.9 3.5 4.4 7 6.4 ...
##  $ E     : num  0.2 0.2 0.3 0.3 0.1 0.3 0.4 0.4 0.5 0.5 ...
##  $ Qls   : int  2640 3440 12200 7600 6250 5650 5300 4700 3940 5300 ...
##  $ Qmm   : num  0.634 0.826 2.928 1.824 1.5 ...

View(BasinObs) # ogled podatkov v ločenem oknu
```

Avtorji paketov izdajajo nove različice paketov s popravki napak in novimi funkcijami, zato je običajno dobro, da jih posodabljate. Imejte v mislih, da bodo nove različice paketov občasno vsebovale napake ali delovale nekoliko spremenjeno (npr. drugačno delovanje funkcij), kar morda pomeni, da vaši ukazi ne bodo več delovali tako, kot so pred posodobitvijo. Za posodobitve paketov lahko uporabite funkcijo *update.packages()* ali pa posodobitev izvedete preko grafičnega vmesnika RStudio.

Programsko orodje R ima centralizirano skladišče paketov, imenovano CRAN[19] (The Comprehensive R Archive Network). Vsi tamkajšnji paketi imajo visoke zahteve glede kakovosti. Ti paketi morajo biti redno posodobljeni in dokumentirani. Namestite lahko kateri koli paket iz skladišča CRAN neposredno iz konzole R oziroma preko grafičnega vmesnika RStudio, kot je bilo to prikazano zgoraj (Slika 4). Dodatne pakete, ki niso vključeni v CRAN, lahko najdete na:

- GitHub[20];
- Bioconductor[21];

[19] https://cran.r-project.org/.

[20] https://github.com/.

[21] http://bioconductor.org/.

- R-Forge[22].

Sezname paketov, povezanih s hidrologijo, lahko najdete tukaj:

- AboutHydrology[23].
- Pregled paketov na CRAN: Hidrološki podatki in modeliranje[24].

Kot zanimivost pokažimo še primer, kako lahko z uporabo programskega orodja R prenašamo datoteke neposredno s povezavo https in kako jih lahko uvozimo v programsko okolje R z uporabo paketa *readr*, prenos datoteke s spletne povezave in shranjevanje datoteke v delovni direktorij (angl. *working directory*), ki si ga lahko ogledate z uporabo funkcije *getwd()*:

```
download.file( "https://monashdatafluency.github.io/r-intro-2/r-intro-2-files
.zip",  destfile="r-intro-2-files.zip")
# datoteka je v formatu .zip, razširitev te datoteke
unzip("r-intro-2-files.zip")
# install.packages("readr") # namestitev paketa readr
library(readr, quietly=TRUE) # aktivacija paketa

## Warning: package 'readr' was built under R version 4.1.3

# branje datoteke z imenom geo.csv
geo <- read_csv("r-intro-2-files/geo.csv")

## Rows: 196 Columns: 7
## -- Column specification --------------------------------------------------
------
## Delimiter: ","
## chr (3): name, region, income2017
## dbl (2): lat, long
## lgl (2): oecd, g77
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this m
essage.

head(geo) # poglejmo prvih nekaj vrstic teh podatkov

## # A tibble: 6 x 7
##   name                region   oecd  g77     lat   long income2017
##   <chr>               <chr>    <lgl> <lgl> <dbl>  <dbl> <chr>
## 1 Afghanistan         asia     FALSE TRUE   33    66    low
## 2 Albania             europe   FALSE FALSE  41    20    upper_mid
## 3 Algeria             africa   FALSE TRUE   28     3    upper_mid
```

[22] https://r-forge.r-project.org/.

[23] http://abouthydrology.blogspot.com/2012/08/r-resources-for-hydrologists.html.

[24] https://cran.r-project.org/view=Hydrology.

```
## 4 Andorra             europe   FALSE FALSE  42.5   1.52 high
## 5 Angola              africa   FALSE TRUE  -12.5  18.5  lower_mid
## 6 Antigua and Barbuda americas FALSE TRUE   17.0 -61.8  high
```

R za shranjevanje datotek (v kolikor ni definirana točna (druga) lokacija) uporablja delovni direktorij, ki si ga lahko ogledate z uporabo funkcije *getwd()*, lahko pa ga tudi spremenite z uporabo funkcije *setwd()*. V povezavi s tema dvema funkcijama velja omeniti tudi funkcijo *list.files()*, ki vrne seznam vseh datotek v delovnem direktoriju. Ukaz je zelo uporaben pri recimo avtomatskem branju določenih datotek.

Naloga 19: Najdite paket za uporabo in izračun momentov L, paket namestite in ga uporabite, izračunajte L-momente podatkov o pretoku, ki ste jih obravnavali v okviru Naloge 16 (L-momenti so nekaj podobnega kot navadni statistični momenti, povprečje, varianca, asimetrija (angl. skewness), sploščenost (angl. kurtosis)).

Naloga 20: Za podatke, vključene v paket airGR, z uporabo funkcije apply izračunajte osnovno statistiko, z uporabo funkcije summary pa osnovne statistike za stolpce 2, 3 in 4 ter ugotovite, za katere hidrološke spremenljivke gre ter kakšne so njihove enote.

3.11 Datum in čas v programskem okolju R

V okviru hidroloških analiz se pogosto srečujemo s časovnimi podatki, ki predstavljajo hidrološke meritve ali kakšna druga opazovanja. Programsko orodje R uporablja naslednjo predstavitev datumov in časov, datumi so predstavljeni z razredom *Date*, časi pa z razredoma *POSIXct* in *POSIXlt*. Datumi so interno shranjeni kot število dni od 1970-01-01, časi pa kot število sekund od 1970-01-01. Iz znakovnega niza lahko definiramo datum s funkcijo *as.Date()*:

```
x <- as.Date("1970-01-01") # definiramo datum
x # poglejmo vsebino

## [1] "1970-01-01"

# definiramo argument format, ki določa format podatkov
x1 <- as.Date("1.1.1970", format="%d.%m.%Y")
unclass(x1) # pretvorba nazaj v obliko character

## [1] 0

unclass(as.Date("1971-01-01")) # še en primer z drugim datumom

## [1] 365
```

Pri definiranju oblike datumom lahko izbiramo med različnimi formati[25].

Okrajšava	Opis
%a	okrajšano ime dneva v tednu (glede na jezik operacijskega sistema)
%A	celo ime dneva v tednu (glede na jezik operacijskega sistema)
%b	okrajšano ime meseca (glede na jezik operacijskega sistema)
%B	celo ime meseca (glede na jezik operacijskega sistema)
%c	datum in čas: pri izvozu odvisno od sistema, pri uvozu pa v obliki %a %b %e %H:%M:%S %Y
%C	stoletje (00-99)
%d	**dan meseca (01-31)**
%D	datum v obliki %m/%d/%y
%e	dan meseca (1-31)
%F	ekvivalent %Y-%m-%d
%H	**ura (00-23)**
%I	ura (01-12)
%j	dan leta (001-366)
%m	**mesec (01-12)**
%M	**minuta (00-59)**
%p	AM/PM indikator, uporaben skupaj z %I (ne z %H)
%R	ekvivalent %H:%M
%S	**sekunda (00-61)**
%T	ekvivalent %H:%M:%S
%u	dan v tednu kot število (0-6, nedelja je 0)
%y	**leto brez stoletja (00-99)**
%Y	**leto s stoletjem**

Slika 5: Primer nekaterih uporabnih okrajšav datumov in časa, ki jih lahko uporabimo pri določitvi argumentov.

Podatek o času je v programskem okolju R predstavljen z razredom *POSIXct* ali *POSIXlt. POSIXct* je uporaben, kadar želite shraniti čase v obliki, podobni podatkovnim okvirjem. *POSIXlt* vsebuje še dodatne uporabne informacije, kot so dan v tednu, dan v letu, mesec, dan v mesecu. To je lahko v nekaterih primerih uporabno. Omeniti velja še nekatere druge uporabne funkcije, kot so *weekdays*, *months*, *quarters*.

```
x <- Sys.time() # definiramo objekt x glede na trenutni čas
x # vsebina

## [1] "2024-10-08 11:40:34 CEST"

class(x) # poglejmo, v kateri obliki je čas zapisan

## [1] "POSIXct" "POSIXt"

p <- as.POSIXlt(x) # preoblikujemo čas v format POSIXlt
names(unclass(p)) # poglejmo dodatne podatke, ki jih ta oblika vsebuje
```

25 http://www.ung.si/media/storage/cms/attachments/2016/09/21/14/57/57/skripta_R_2016.pdf.

```
##  [1] "sec"    "min"    "hour"   "mday"   "mon"    "year"   "wday"   "yday"
##  [9] "isdst"  "zone"   "gmtoff"

p$yday # preverimo julijanski dan

## [1] 281
```

Če so vaši datumi zapisani v drugačni obliki, je tu še funkcija *strptime* (inverz te funkcije predstavlja *strftime* funkcija). Omenjena funkcija uporabi znakovni vektor z datumi in časi ter jih pretvori v objekt *POSIXlt*. Izberemo si dva poljubna datuma, oblika zapisa meseca je odvisna od lokalnih jezikovnih nastavitev:

```
datum <- c("Januar 23, 2022 11:42", "Junij 10, 2018 12:10")
x2 <- strptime(datum, "%B %d, %Y %H:%M") # preoblikujemo v format POSIXlt
x2 # vsebina

## [1] "2022-01-23 11:42:00 CET"  "2018-06-10 12:10:00 CEST"

class(x2) # format tega objekta

## [1] "POSIXlt" "POSIXt"
```

Z datumi in časi se lahko izvaja tudi nekatere matematične operacije (npr. seštevanje in odštevanje). Izvajate lahko tudi primerjave (npr. *==*, *<=*):

```
x3 <- as.Date("2017-01-01") # definiramo poljuben datum
# definiramo še drugi objekt
y3 <- strptime("21 Marec 2015 13:44:51", "%d %B %Y %H:%M:%S")
# x3-y3 # ne deluje, ker format obeh objektov ni enak
x3 <- as.POSIXlt(x3) # preoblikujemo prvi objekt
x3-y3 # poskusimo še enkrat

## Time difference of 651.4689 days
```

Dobra lastnost pri teh tipih objektov v programskem okolju R je, da upoštevajo vse potencialno moteče stvari v povezavi z datumi in časi, kot so prestopna leta, poletni čas in časovni pasovi. Oglejmo si še primer z dvema različnima časovnima pasovoma, kjer je osnovni časovni pas določen glede na nastavitve računalnika:

```
x4 <- as.POSIXct("2012-10-25 01:00:00")
y4 <- as.POSIXct("2012-10-25 01:00:00", tz = "GMT") # poljuben tz
z4 <- as.POSIXct("2012-10-25 01:00:00", tz = "America/Chicago")
y4-x4 # poglejmo razliko

## Time difference of 2 hours

z4-x4 # razlika

## Time difference of 7 hours

as.double(z4-x4, units = "days") # pretvorba v število dni

## [1] 0.2916667
```

Za delo s časovnimi vrstami, kar hidrološki podatki pogosto so, so v R na voljo namenski paketi, kot je *zoo*, kjer so vključene funkcije, s katerimi lahko združujemo podatke o datumu in času ter različnih spremenljivkah v en objekt. Takšen pristop omogoča relativno enostavno pripravo določenih delov podatkov, kot je recimo določitev največjih letnih pretokov ali izračun mesečnih količin padavin. Pa poglejmo nekaj primerov na podlagi podatkov z vodomerne postaje Veliko Širje na reki Savinji:

```
#install.packages("zoo") # namestimo paket
library(zoo, quietly=TRUE)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

podatki <- read.table(file="C:/Users/nbezak/OneDrive - Univerza v Ljubljani/U
cbenik/Savinja-Veliko SirjeI-2005.txt",header=TRUE,sep=";",dec=".")
# spremenimo imena stolpcev
names(podatki) <- c("Datum", "Vodostaj", "Pretok",
"Temperatura", "Transport", "Vsebnost")
# datume preoblikujemo v format as.POSIXct
podatki[,1] <- as.POSIXct(strptime(podatki[,1],format="%d.%m.%Y"))
Qzoo <- zoo(podatki[,3],podatki[,1]) # definiramo objekt zoo
head(Qzoo) # poglejmo strukturo tega objekta

## 2005-01-01 2005-01-02 2005-01-03 2005-01-04 2005-01-05 2005-01-06
##     37.982     35.515     32.395     28.073     26.068     24.165

mes1 <- as.yearmon(time(Qzoo)+3600) # določimo mesečne vrednosti
head(mes1) # poglejmo strukturo podatkov

## [1] "jan. 2005" "jan. 2005" "jan. 2005" "jan. 2005" "jan. 2005" "jan. 2005
"

leto1 <- as.numeric(floor(as.yearmon(time(Qzoo)+3600))) # letne vrednosti
head(leto1) # struktura podatkov

## [1] 2005 2005 2005 2005 2005 2005

# s funkcijo aggregate lahko izračunamo letne maksimume pretokov
letneVsote <- aggregate(Qzoo, leto1, max)
letneVsote # rezultati

##  2005
## 382.1

mesVsote <- aggregate(Qzoo,mes1,sum) # mesečne vsote pretokov
head(mesVsote) # vsebina

## jan. 2005 feb. 2005 mar. 2005 apr. 2005  maj 2005 jun. 2005
##   578.223   360.518   973.600  1588.382  1124.756   455.644
```

```
# ter še en malce bolj kompleksen primer
mesVsotePovp <- aggregate(mesVsote,months(time(mesVsote)),sum)
```

Kot zanimivost omenimo še funkcijo *merge*, s katero lahko združujemo zvezne podatke in podatke, ki imajo manjkajoče vrednosti, kar je še posebej uporabno pri podatkih z urnim (ali bolj natančnim) časovnim korakom:

```
# prvi stolpec preoblikujemo v format Date
podatki[,1] <- as.Date(podatki[,1],format="%d.%m.%Y")
# predpostavimo, da manjkajo podatki za mesec februar
podatki <- podatki[-(32:59),]
# generiramo zvezen vektor datumov
time.seq <- seq(as.Date(podatki[,1],format="%d.%m.%Y")[1], as.Date(podatki[,1
],format="%d.%m.%Y")[length(podatki[,1])], by = "1 day")
# združevanje podatkov
zdr <- merge(x=podatki, y=as.data.frame(time.seq), by.x = "Datum", by.y = "ti
me.seq", all.y = T)
zdr[25:35,1:3] # vidimo, da so manjkajoči podatki označeni z NA
```

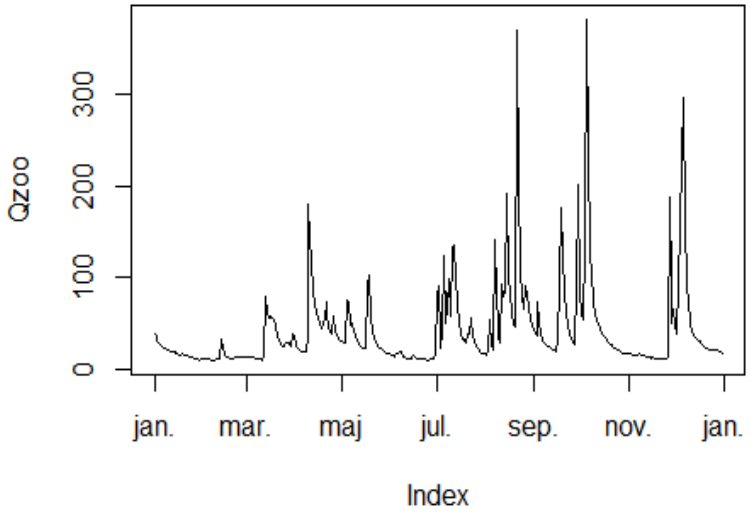
```
##         Datum Vodostaj Pretok
## 25 2005-01-24      193 13.126
## 26 2005-01-25      190 11.994
## 27 2005-01-26      190 11.994
## 28 2005-01-27      189 11.654
## 29 2005-01-28      189 11.654
## 30 2005-01-29      183  9.836
## 31 2005-01-30      187 11.006
## 32 2005-01-31       NA     NA
## 33 2005-02-01       NA     NA
## 34 2005-02-02       NA     NA
## 35 2005-02-03       NA     NA
```

Naloga 21: Izračunajte največje, najmanjše in povprečne vrednosti pretokov v posameznih obdobjih (januar–marec, april–junij itd., torej za četrtine leta). Namig: iščete podobno funkcijo kot je as.yearmon.

3.12 Izris osnovnih grafov

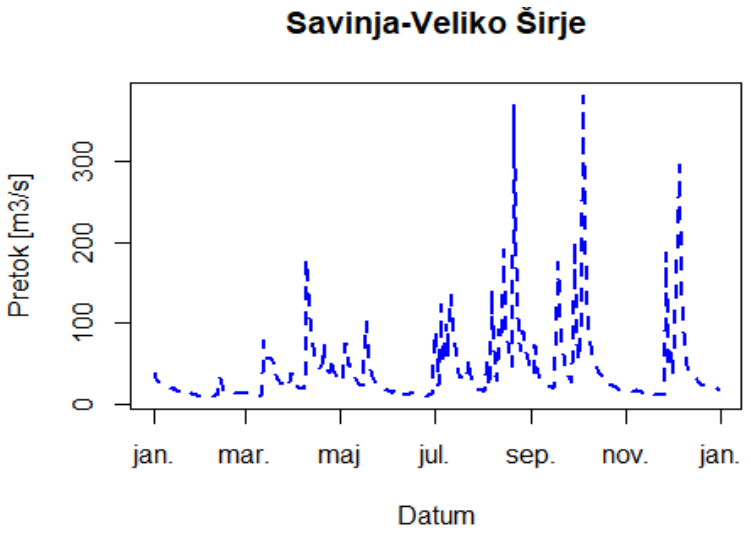
V tem poglavju bomo prikazali uporabo najpogostejših grafičnih prikazov podatkov. R ponuja veliko načinov predstavitve podatkov. Osnovna funkcija v programu R za izris grafov je *plot*, ki ima številne argumente, ki jih lahko spreminjamo in s katerimi določamo, kaj bo izrisano na določenem grafu. Poglejmo si en zelo poenostavljen primer, kjer bomo za izris uporabili objekt zoo, ki smo ga definirali v prejšnjih korakih, in izrisali graf. V tem primeru je struktura objekta takšna, da R prepozna časovne podatke in pripadajoče vrednosti ostalih podatkov in jih ustrezno izriše na x- in y-osi, kar je ena izmed prednosti uporabe paketov, kot je *zoo*, in tudi objektov v teh paketih:

```
plot(Qzoo)
```



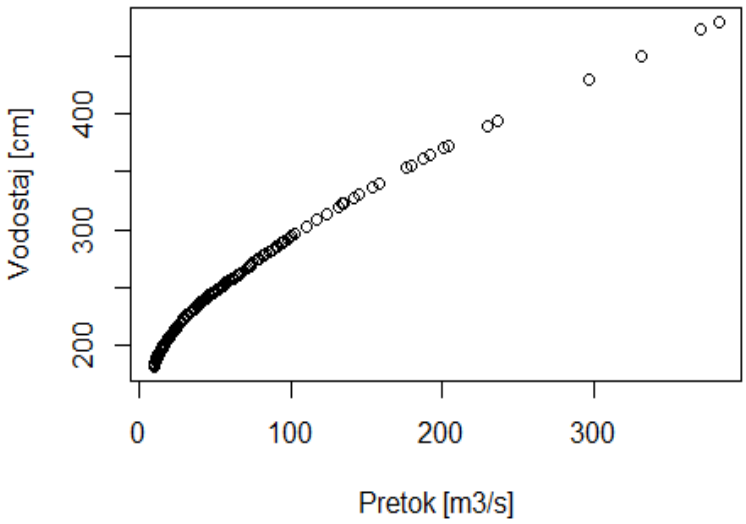
Slika 6: Izris linijskega grafa pretokov.

```
# spremenjene nastavitve, omeniti velja še argumenta xlim in ylim
# ki definirata razpon na x- in y-osi
plot(Qzoo,xlab="Datum",ylab="Pretok [m3/s]",main="Savinja-Veliko Širje",
col="blue",lty=2,lwd=2)
```



Slika 7: Izris linijskega grafa pretokov s spremenjenimi nastavitvami.

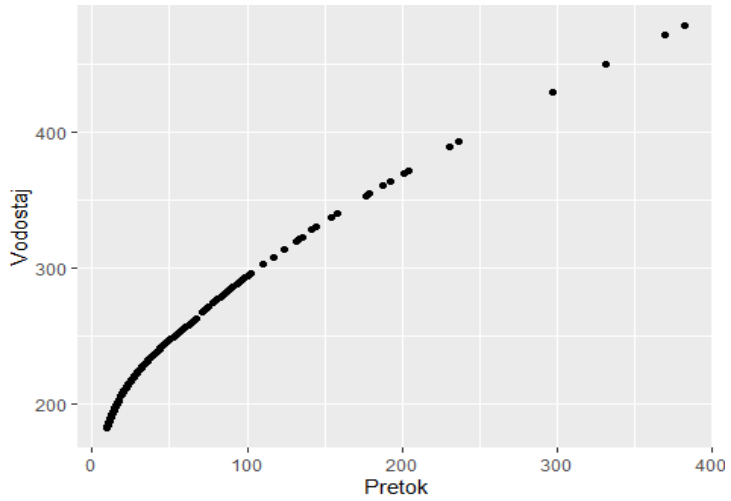
```
# izris preprostega raztresenega (angl. scatter) grafa
plot(x=podatki[,3],y=podatki[,2],xlab="Pretok [m3/s]",
ylab="Vodostaj [cm]", main="Q-H krivulja")
```



Slika 8: Q-H krivulja.

Argumenti, povezani z nastavitvami *par*, omogočajo spreminjanje številnih nastavitev[26]. Za izris vizualno nekoliko lepših grafov pa obstaja veliko paketov. Pogosto se recimo uporablja *ggplot2*:

```
#install.packages("ggplot2") # paket najprej namestimo
library(ggplot2, quietly=TRUE) # aktiviramo

## Warning: package 'ggplot2' was built under R version 4.1.3

ggplot(podatki, aes(x = Pretok, y = Vodostaj)) + # izris grafa
  geom_point() # prikaz točk
```

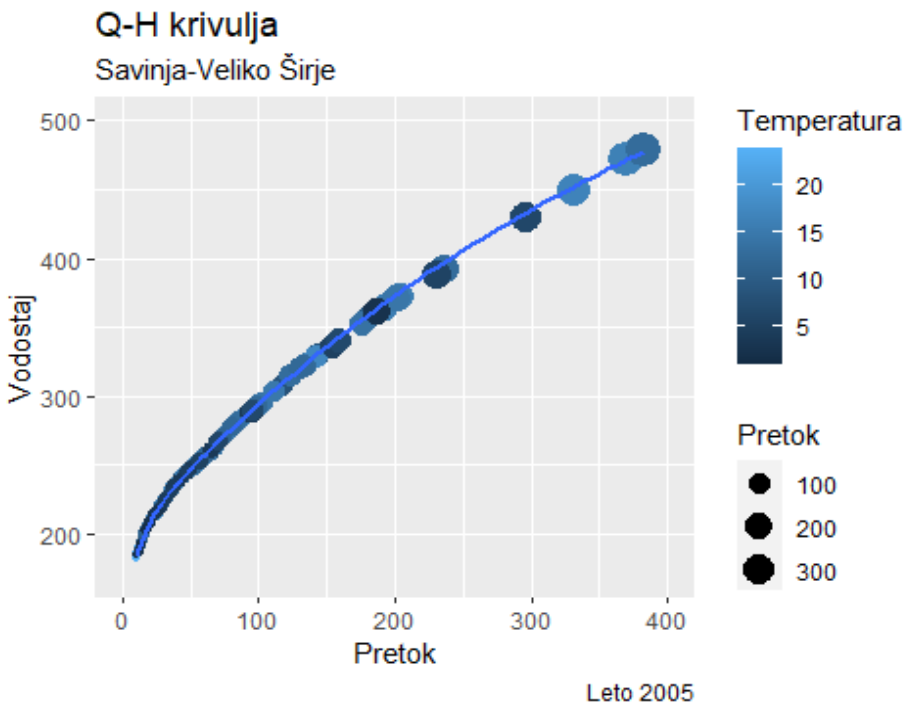


Slika 9: Q-H krivulja s paketom ggplot2.

[26] https://www.statmethods.net/advgraphs/parameters.html.

Običajni način uporabe funkcije *ggplot* je, da ji določite podatkovni okvir (data.frame) in ji nato poveste, katere stolpce naj uporabi za vrednosti x in y, kot je to prikazano na primeru zgoraj (slika 9). Paket *ggplot2* je zasnovan za delo s podatkovnimi okvirji kot virom podatkov in ne s posameznimi vektorji, zato boste z vektorji lahko uporabili le omejen del njegovih zmogljivosti. Knjižnica *ggplot* ali *ggplot2* omogoča sestavljanje grafa po načelu dodajanja posameznih grafičnih elementov ali slojev z uporabo operatorja *+*, zato so ukazi pogosto razporejeni v več vrsticah in vidite kodo v naslednji obliki:

```
# definiramo, katere podatke želimo izrisati
# temperatura se uporabi za barvno skalo, pretok za velikost
ggplot(podatki, aes(x=Pretok, y=Vodostaj)) +
  geom_point(aes(col=Temperatura, size=Pretok)) +
  geom_smooth(method="loess", se=F) + # dodamo funkcijo, ki opisuje podatke
  xlim(c(0, 400)) + # definiramo območje prikaza na x-osi
  ylim(c(170, 500)) + # definiramo območje prikaza na y-osi
  labs(subtitle="Savinja-Veliko Širje", # definiramo naslove in ostalo
    y="Vodostaj", x="Pretok", title="Q-H krivulja",
    caption = "Leto 2005")

## `geom_smooth()` using formula = 'y ~ x'
```

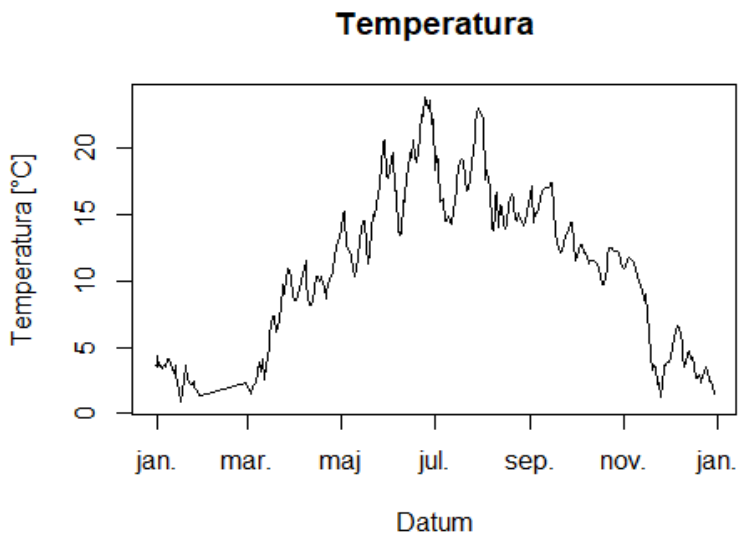


Slika 10: Q-H krivulja z uporabo funkcije ggplot z barvno skalo.

R vključuje veliko funkcij za izris cele vrste različnih grafov. Lep pregled je narejen na spletni strani R Graph Gallery[27]. Zanimiva je tudi spletna stran datacamp[28], ki ponuja tečaje uporabe programskega orodja R.

Pa poglejmo še nekaj drugih primerov. Če želimo s funkcijo *plot* izdelati linijski graf, kot vhodni podatek uporabimo vektorja x in y ter uporabimo argument *type* je *"l"*.

```
# izris linijskega grafa na podlagi podatkov o T vode
plot(x=podatki[,1],y=podatki[,4],type="l",xlab="Datum",ylab="Temperatura [°C]
", main="Temperatura")
```



Slika 11: Linijski graf za temperaturo vode.

Na obstoječi graf lahko dodajamo tudi točke (*points*), črte (*lines*), oznake besedila (*text*) in tudi legendo (*legend*). Pri točkah in linijah je treba definirati koordinato x in y ter paziti, da je ta enaka kot v primeru osnovnega grafa, na katerega se izrisujejo podatki. V nasprotnem primeru lahko pride do zamika pri izrisu točk in linij. Funkcija *text* zahteva koordinato x in y, kjer se besedilo izpiše (središče besedila), podobno velja za funkcijo *legend*. Alternativa koordinatam so oznake *top*, *bottomleft*, *right* itd. Funkcija *legend* ima številne argumente, s katerimi lahko določimo opis legende[29].

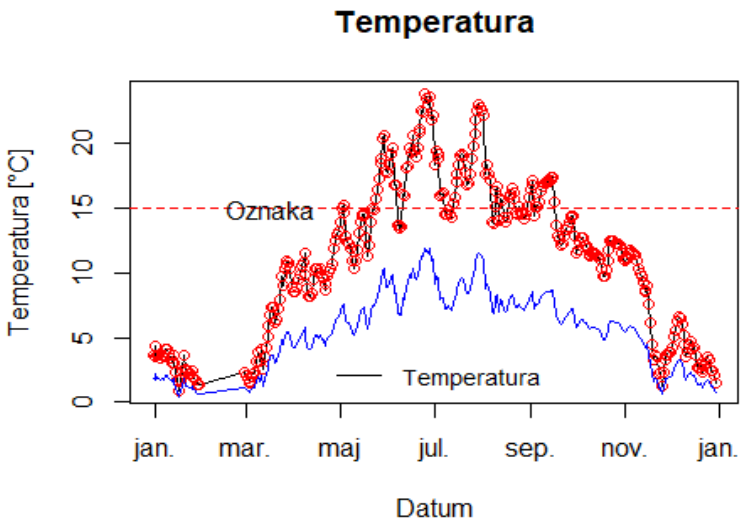
```
plot(x=podatki[,1],y=podatki[,4],type="l",xlab="Datum",ylab="Temperatura [°C]
", main="Temperatura")
points(x=podatki[,1],y=podatki[,4],col="red") # na graf dodamo še točke
# še dodatno linijo (1/2 izmerjene vrednosti temperature)
lines(x=podatki[,1],y=podatki[,4]/2,col="blue")
# dodamo horizontalno črto pri vrednosti 15
abline(h=15,col="red",lty=2)
```

[27] https://r-graph-gallery.com.

[28] https://www.datacamp.com/courses/data-visualization-in-r.

[29] https://r-coder.com/add-legend-r/?utm_content=cmp-true.

```
# poljubno besedilo
text(x=podatki[50,1],y=15, labels="Oznaka")
# legenda
legend("bottom",legend="Temperatura",col="black",lty=1,bty="n",cex=0.9)
```

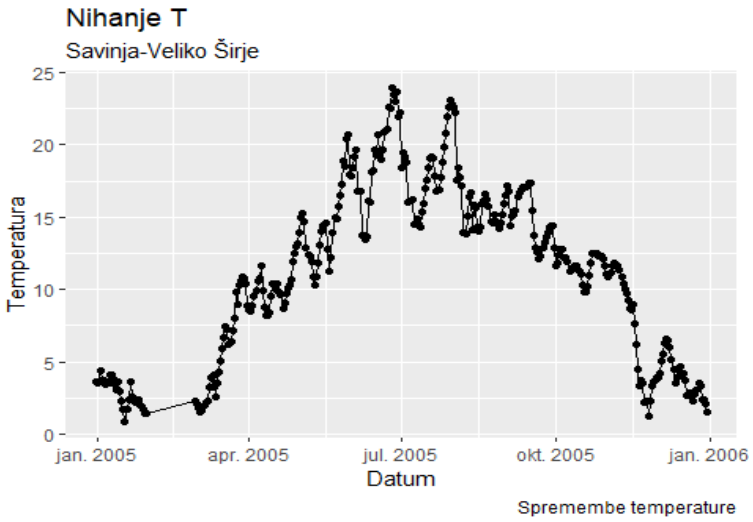


Slika 12: Linijski graf temperature z dodatnimi podatki.

V paketu *ggplot2* lahko podoben rezultat dobimo z uporabo funkcije *geom_line* za dodajanje linijskih grafov ter *geom_points* za dodajanje točk. Paket *ggplot2* omogoča številne nastavitve, nekatere izmed njih so opisane tudi tukaj[30].

```
ggplot(podatki[,], aes(x = Datum, y = Temperatura)) +
  geom_line() + # dodamo linijo
  geom_point() + # ter še točke
  labs(subtitle="Savinja-Veliko Širje", # definiramo naslove in ostalo
       y="Temperatura",
       x="Datum",
       title="Nihanje T",
       caption = "Spremembe temperature")
```

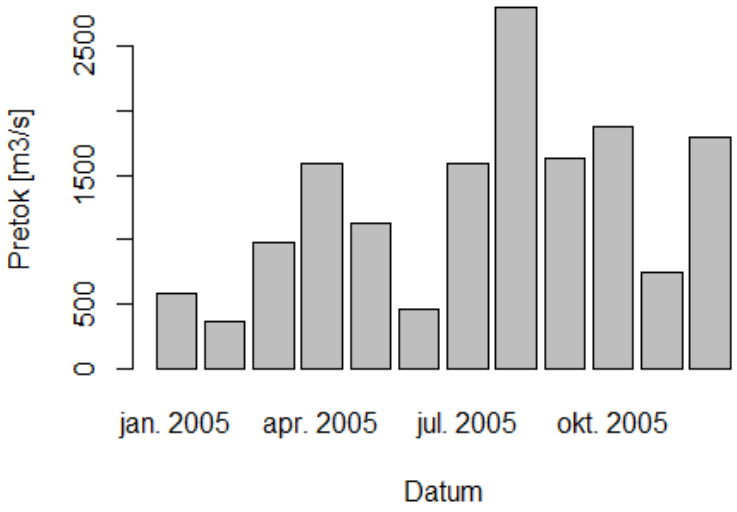
[30] https://datacarpentry.org/R-ecology-lesson/04-visualization-ggplot2.html.



Slika 13: Linijski graf temperature, prikazan s paketom ggplot2.

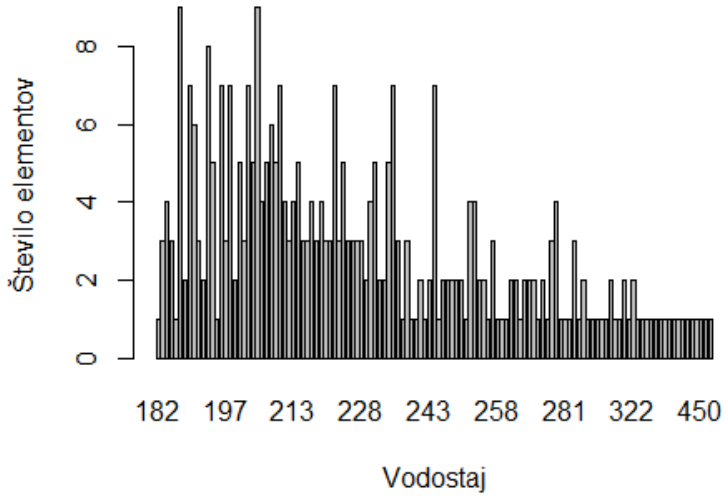
Za izris stolpičnega grafa lahko uporabite funkcijo *barplot*, kjer lahko kot vhodni podatek uporabite vektor vrednosti za višino vsake vrstice in (po želji) vektor oznak za vsako vrstico. Če so v vektorju imena elementov, se ta imena samodejno uporabijo kot oznake. Včasih se stolpični graf nanaša na graf, kjer stolpci predstavljajo število elementov v vsaki kategoriji. Podoben je histogramu, vendar ima diskretno in ne zvezno x-os. Če želite ustvariti število vsake edinstvene vrednosti v določenem vektorju, lahko uporabite funkcijo *table*. Tudi paket *ggplot2* omogoča izris podobnih grafov ali z uporabo funkcije *geom_col* oziroma *geom_bar*, odvisno od tipa podatkov, ki jih želimo prikazati. Pogosto pa želimo izrisati tudi histogram z uporabo funkcije *hist*, ki izriše število podatkov v definiranih razredih. V paketu *ggplot2* je analogna funkcija *geom_histogram*. Za izris stolpičnega grafa bomo uporabili mesečne vsote pretokov, ki smo jih izračunali zgoraj; round je funkcija za zaokroževanje podatkov.

```
round(mesVsote,0)

## jan. 2005 feb. 2005 mar. 2005 apr. 2005  maj 2005 jun. 2005 jul. 2005 avg.
2005
##       578       361       974      1588      1125       456      1591
2800
## sep. 2005 okt. 2005 nov. 2005 dec. 2005
##      1625      1881       742      1795

barplot(mesVsote,xlab="Datum", ylab="Pretok [m3/s]") # stolpični graf
```



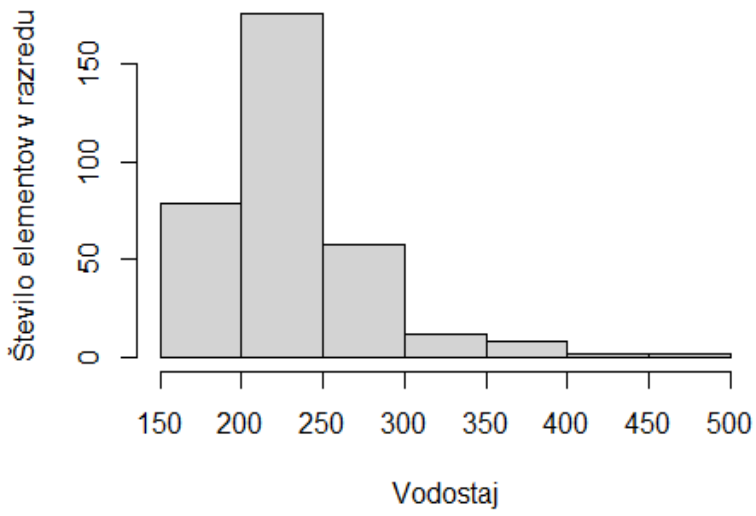
Slika 14: Stolpični graf pretokov.

```
head(table(podatki$Vodostaj)) # poglejmo rezultat takšnega grafa

##
## 182 183 184 185 186 187
##   1   3   4   3   1   9

table(mtcars$cyl) # za boljše razumevanje poglejmo še rezultat funkcije table
na podatkih mtcars in stolpcu, ki prikazuje število cilindrov, funkcija table
torej določi, koliko elementov ima 4, 6 in 8 cilindrov

##  4  6  8
## 11  7 14

# izris stolpičnega grafa
barplot(table(podatki$Vodostaj),xlab="Vodostaj",ylab="Število elementov")
```



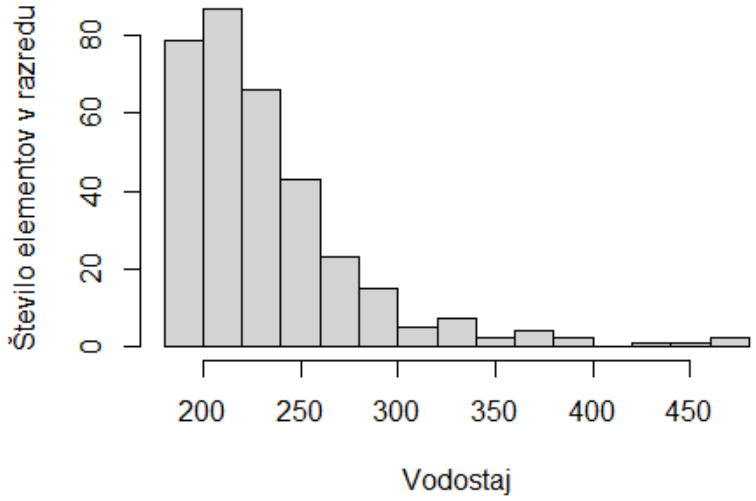
Slika 15: Stolpični graf števila elementov posameznih vrednosti vodostajev.

```
 # izris histograma
hist(podatki$Vodostaj,xlab="Vodostaj",ylab="Število elementov v razredu",main
="Histogram")
```



Slika 16: Histogram vodostajev.

```
# spremenimo število razredov na x-osi
hist(podatki$Vodostaj,breaks=12,xlab="Vodostaj",ylab="Število elementov v raz
redu",main="Histogram")
```

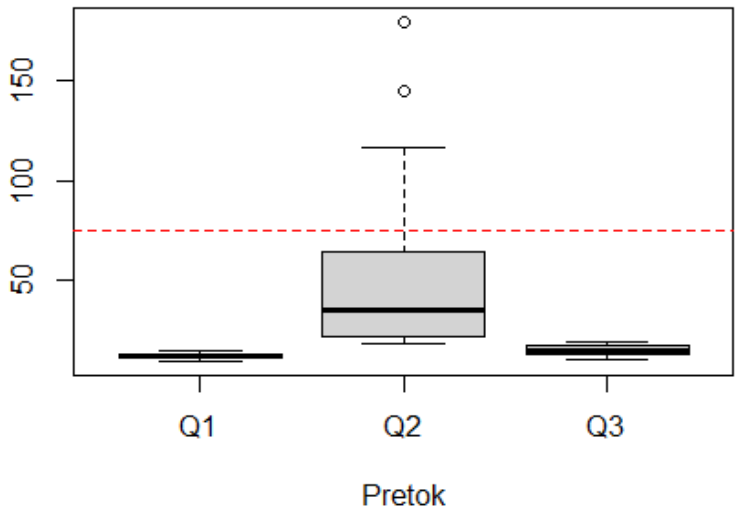


Slika 17: Histogram vodostajev z drugačnim številom razredov.

Eden izmed zadnjih grafov, ki ga bomo posebej omenili, je okvir z ročaji, ki ga lahko v primeru, da imamo opravka s faktorji, izrišemo kar z uporabo funkcije *plot*. V primeru numeričnih podatkov lahko uporabimo funkcijo *boxplot* oziroma *geom_boxplot* v primeru paketa *ggplot2*. V povezavi z okvirjem z ročaji je smiselno omeniti tudi t. i. violinske grafe, ki omogočajo malce

drugačen prikaz podatkov, kot pa okvirji z ročaji[31]. *geom_violin* je funkcija, ki jo lahko uporabite za izris takšnega tipa grafa z uporabo paketa *ggplot2*.

```
# prikažemo pretoke za tri različna obdobja
boxplot(podatki$Pretok[20:40],podatki$Pretok[60:80],podatki$Pretok[120:140],
        names=c("Q1","Q2","Q3"),xlab="Pretok")
abline(h=75,col="red",lty=2) # dodamo horizontalno črto pri vrednosti 75
```

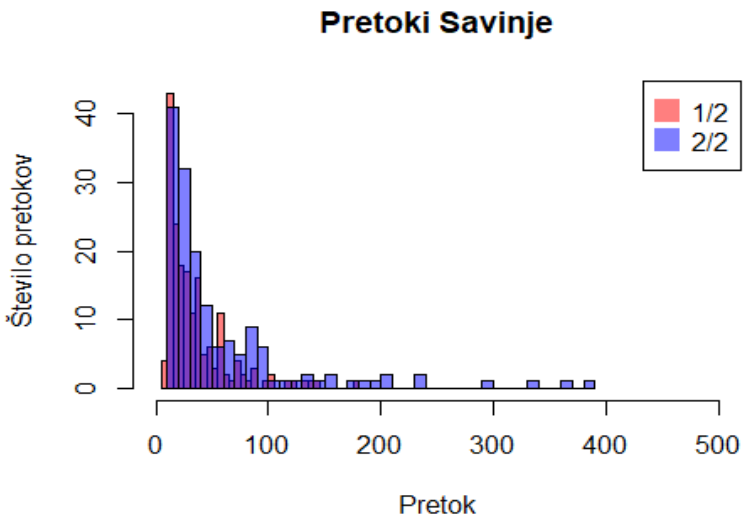


Slika 18: Okvir z ročaji za podatke o pretokih.

Dodatno je v nekaterih primerih lahko uporabna tudi funkcija *curve* (za izris krivulj) ter tudi funkciji *mtext* (za dodajanje besedila na robove grafov) in pa *segments* (za izris delov linij na graf). Poglejmo še en primer, kako lahko grafe izrišemo enega prek drugega. Alternativa za izris takšnega grafa je tudi paket *ggplot2*. Najprej bomo izrisali prvi histogram, kjer bodo prikazani podatki za polovico leta, argument *breaks* definira število razredov histograma. Ta argument omogoča tudi druge nastavitve (za več informacij poglejte pomoč za funkcijo *?hist*). Pomembno je, da so enote prikaza na x-osi pri prvem in drugem grafu identične, da ne pride do zamika podatkov. Na obstoječi histogram bomo nato dodali še drugega (to je definirano z uporabo argumenta *add=T* oziroma *add=TRUE*), kjer prikazujemo podatke za pol leta, barva in prosojnost sta definirani z uporabo funkcije *rgb*. Poleg tega bomo dodali še legendo.

```
hist(podatki[1:180,3], breaks=30, xlim=c(0,500), col=rgb(1,0,0,0.5),
xlab="Pretok", ylab="Število pretokov", main="Pretoki Savinje" )
hist(podatki[181:365,3], breaks=30, xlim=c(0,500),
col=rgb(0,0,1,0.5), add=T)
legend("topright", legend=c("1/2","2/2"), col=c(rgb(1,0,0,0.5),
  rgb(0,0,1,0.5)), pt.cex=2, pch=15 )
```

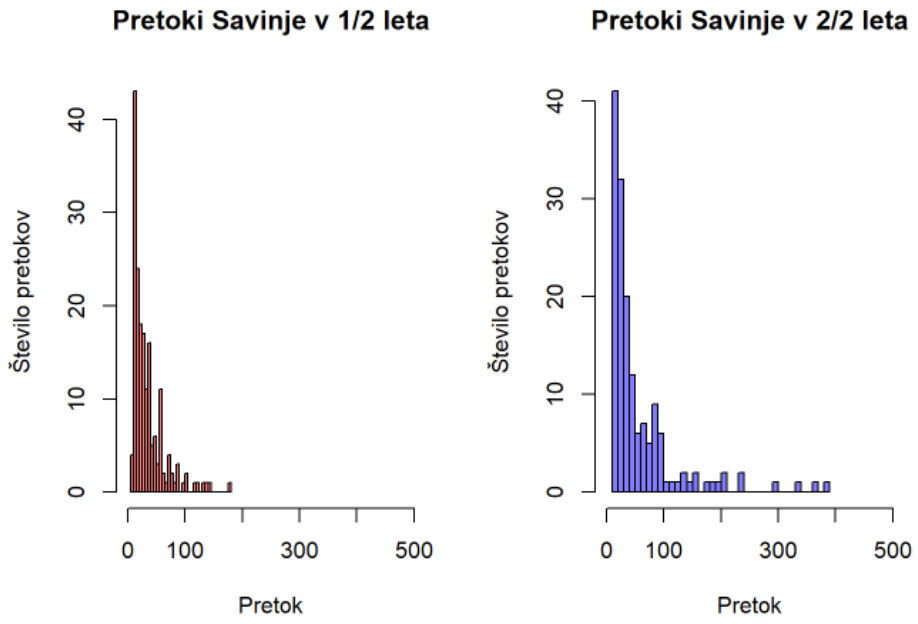
[31] https://r-graph-gallery.com/violin.html.



Slika 19: Izris dveh histogramov pretokov na isti graf.

Dodatno lahko risalno območje s funkcijo *par* razdelimo na več delov, kjer nato v vsakem delu risalnega območja izrišemo svoj graf.

```
par(mfrow=c(1,2)) # razdelitev risalnega območja na dva dela
# prvi histogram, nastavitve grafa so podobne kot zgoraj
hist(podatki[1:180,3], breaks=30, xlim=c(0,500), col=rgb(1,0,0,0.5),
xlab="Pretok",ylab="Število pretokov", main="Pretoki Savinje v 1/2 leta" )
# drugi histogram, nastavitve grafa so podobne kot zgoraj
hist(podatki[181:365,3], breaks=30, xlim=c(0,500), col=rgb(0,0,1,0.5),
xlab="Pretok", ylab="Število pretokov", main="Pretoki Savinje v 2/2 leta" )
```



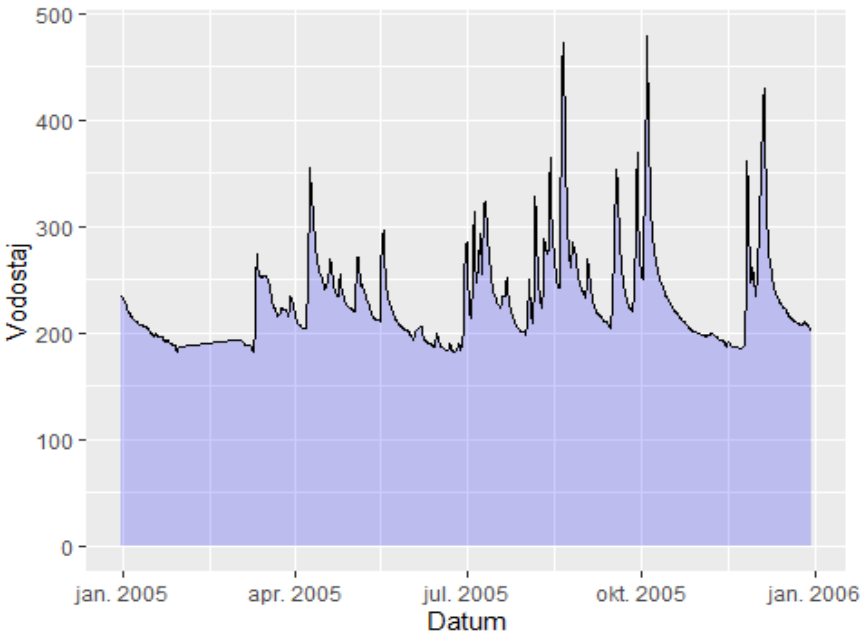
Slika 20: Dva histograma eden zraven drugega.

Shranjevanje grafov lahko poteka neposredno preko grafičnega vmesna Rstudio, kjer imate v zavihku *Plots* možnost, da izrisani graf shranite kot sliko v različnih formatih (npr. png, jpeg, tiff, eps) ali kot pdf-dokument ali pa graf direktno kopirate in prilepite z uporabo funkcije *Copy to Clipboard*. Alternativni postopek je uporaba funkcij, kot so *bmp*, *jpeg*, *png*, *tiff* ali *pdf*. Z uporabo teh funkcij lahko namreč shranite grafe tudi med izračuni z uporabo zank ali pri podobnih kompleksnih izračunih, kjer bi bilo ročno shranjevanje grafov precej zamudno. Postopek uporabe teh funkcij je prikazan za primer shranjevanja grafa preko uporabe funkcije *png*:

```
png(file = "mojgraf.png", bg = "transparent")
plot(1:10)
rect(1, 5, 3, 7, col = "white")
dev.off()
```

Paket *ggplot2* omogoča tudi izris barvnih grafov oziroma območij (funkcija *geam_area*).

```
ggplot(podatki, aes(x = Datum, y = Vodostaj)) +
  geom_area(colour = "black", fill = "blue", alpha = .2)
```



Slika 21: Primer linijskega grafa z obarvanim območjem pod linijo.

Obseg paketov in možnosti, kako se lotiti priprave grafov v programskem okolju R, je precej obsežen. Ustrezno rešitev določenega problema pogosto najdemo kar s spletnim iskanjem.

Kot primer naj omenimo paket *openair*, ki je namenjen prikazu podatkov o kakovosti zraka[32]. Zelo dober pregled grafičnih nastavitev pa podaja tudi učbenik R Graphics Cookbook[33].

[32] https://rpubs.com/NateByers/Openair.

[33] https://r-graphics.org/.

Naloga 22: Za podatke z vodotoka Savinja (vodostaj, pretok in transport suspendiranih snovi) izrišite okvir z ročaji (boxplot) in graf ustrezno opremite z naslovom, oznakami osi itd. Y-os na grafu naj bo izrisana v log merilu.

Naloga 23: Iz nabora bolj naprednih grafov (dober pregled podaja spletna stran R graph gallery[34]) izberite en tip grafa, ki ga tudi izrišite in ustrezno opremite.

3.13 Pisanje lastnih funkcij in uporaba zank

Uporabniki programskega jezika R imajo možnost pisati tudi lastne funkcije. Pisanje funkcij predstavlja pomemben korak pri prehodu od navadnega uporabnika k razvijalcu, ki ustvarja nove funkcionalnosti programskega jezika R. Funkcije se pogosto uporabljajo za operacije, ki jih je treba izvesti večkrat, morda pod nekoliko drugačnimi pogoji.

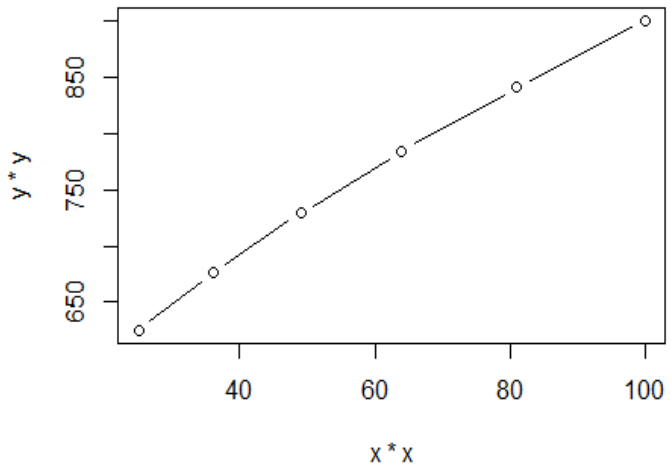
Pisanje funkcij omogoča razvijalcem, da ustvarijo vmesnik za kodo, ki je določen z nizom parametrov. Zapis kode v obliki funkcij poenostavi uporabo drugim uporabnikom, saj jim ni treba poznati vseh podrobnosti delovanja kode in jo lahko uporabljajo na enak način kot vse druge funkcije. Dodatno lahko funkcije uporabite tudi kot argument pri drugih funkcijah, kar je še posebej uporabno pri operacijah, kot so *apply*, *lapply* ali *sapply*. Funkcije so definirane z ukazom *function* in so shranjene kot objekti R. Poglejmo preprost primer funkcije za izračun kvadrata v programskem jeziku R:

```
# kvad bo ime funkcije, x je edini argument, ki ga funkcija uporablja
kvad <- function(x){
rez <- x*x # izračunamo kvadrat in ga shranimo v objekt rez
return(rez) # izpišemo rezultat
}
kvad(10) # preverimo delovanje funkcije

## [1] 100

kvad(3:6) # funkcijo lahko uporabimo tudi na celotnem vektorju

## [1]  9 16 25 36

# uporabimo kot enega izmed argumentov pri funkciji apply
apply(X = podatki[1:5,2:3],MARGIN = 2,FUN = kvad)

##   Vodostaj    Pretok
## 1    54756 1442.6323
## 2    53361 1261.3152
## 3    51529 1049.4360
## 4    48841  788.0933
## 5    47524  679.5406
```

[34] https://r-graph-gallery.com/.

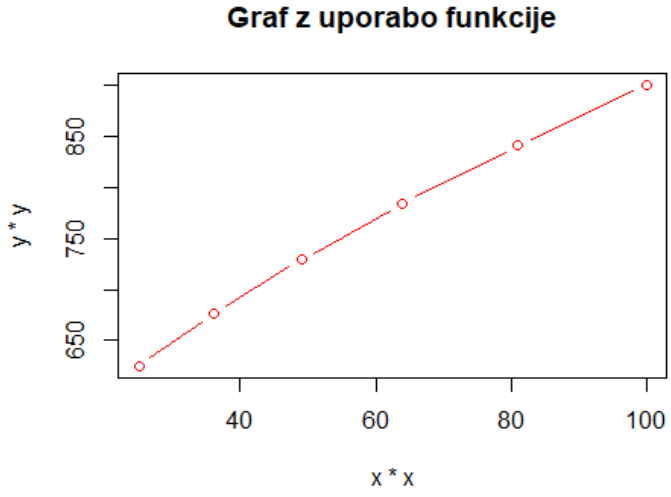
Funkcije imajo lahko tudi več argumentov, ki imajo lahko vnaprej izbrane vrednosti. V nekaterih primerih pa je smiselno pustiti odprt nabor argumentov, ki jih lahko pri določeni funkciji uporabimo. Recimo, da želimo pripraviti funkcijo, ki nam v osnovi izriše kvadratne vrednosti na x- in y-osi z uporabo funkcije *plot*, in pri tem želimo imeti linijski graf, kjer so obenem izrisane tudi točke. Graf je ime funkcije, ki ima tri argumente, *x* in *y* nista določena, *type* že določa tip grafa, oznaka ... pa označuje spremenljivo število argumentov, ki se običajno posredujejo drugim funkcijam. Argument ... se pogosto uporablja pri razširitvi na druge funkcije, ko ne želite kopirati celotnega seznama argumentov prvotne funkcije.

```
graf <- function(x, y, type="b", ...){
  plot(x*x,y*y, type=type, ...) # osnova funkcije
}
graf(x=5:10,y=25:30) # primer uporabe
```



Slika 22: Prvi primer uporabe lastne funkcije.

```
# uporaba dodatnih argumentov
graf(x=5:10,y=25:30,main="Graf z uporabo funkcije",col="red")
```



Slika 23: Uporaba dodatnih argumentov pri uporabi lastne funkcije.

Kontrolne strukture oziroma zanke v R omogočajo nadzor nad potekom izvajanja niza izračunov v programskem okolju R. Nadzorne strukture nam omogočajo, da v kodo R vnesemo nekaj »logike«, namesto da bi vedno izvajali isto kodo. Kontrolne strukture lahko uporabimo, da se odzovemo na vhodne podatke ali značilnosti podatkov in v skladu s tem izvedete različne operacije v programskem okolju R. Nekatere pogosto uporabljene kontrolne strukture so:

- funkciji *if* in *else* preverita določen pogoj in ukrepata na podlagi pogoja;
- *for* izvede zanko večkrat;
- *while* izvede zanko, dokler je pogoj izpolnjen;
- *repeat* izvede neskončne zanke (za ustavitev je treba zanko prekiniti);
- *break* prekine izvajanje zanke;
- *next* preskoči (določen) del zanke.

Večina kontrolnih struktur se uporablja pri pisanju lastnih funkcij ali daljših kombinacij izrazov. Poglejmo en tipičen primer *if* zanke[35]:

```
vrednost <-  25 # definiramo poljuben objekt
if (vrednost > 20) { # definiramo pogoj
    print('To pa je velika številka') # če je pogoj izpolnjen
} else {
    print('Številka ni tako velika') # če pogoj ni izpolnjen
}

## [1] "To pa je velika številka"

# poglejmo še primer, ko je vrednost manjša
vrednost <-  5 # definiramo poljuben objekt
if (vrednost > 20) { # definiramo pogoj
    print('To pa je velika številka') # če je pogoj izpolnjen
} else {
    print('Številka ni tako velika') # če pogoj ni izpolnjen
}

## [1] "Številka ni tako velika"
```

Poglejmo še en primer zanke *for*, ki v vsakem koraku izpiše vrednost indeksa *k*:

```
for(k in 1:5) { # definiramo zanko
print(k) # kaj se v zanki zgodi
} # konec zanke

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
```

[35] https://www.guru99.com/r-if-else-elif-statement.html.

Še primer dvojne zanke *for*, kjer v obeh zankah izpisujemo besedilo glede na vhodni vektor x in posamezne elemente tega vektorja:

```
x <- cbind(c("voda","zrak"),10:11) # definiramo poljuben objekt
for(i in 1:dim(x)[1]) { # definiramo prvo zanko
        for(j in 1:dim(x)[2]) { # definiramo drugo zanko
                print(x[i,j]) # definiramo, kaj se v zankah izvrši
        }   # zaključimo drugo zanko
} # zaključimo prvo zanko
```

```
## [1] "voda"
## [1] "10"
## [1] "zrak"
## [1] "11"
```

Treba je poudariti, da se marsikatera računska operacija v programskem okolju R lahko izvrši tudi z uporabo funkcij *apply*, *lapply*, *sapply* (namesto z uporabo zank). Poglejmo primer uporabe funkcije *sapply*, ki omogoča izračune po posameznih komponentah seznama. V naslednjem primeru bomo uporabili funkcijo *quantile*:

```
x <- list(a=1:10,b=4:9,log=c(TRUE,FALSE,FALSE,TRUE))
sapply(X=x, FUN=quantile)
```

```
##          a    b log
## 0%    1.00 4.00 0.0
## 25%   3.25 5.25 0.0
## 50%   5.50 6.50 0.5
## 75%   7.75 7.75 1.0
## 100% 10.00 9.00 1.0
```

V primerih, ko želimo kodo ponavljati, dokler ni izpolnjen pogoj, pa nam prav pride funkcija *while*. Zanke z uporabo funkcije *while* se začnejo s testiranjem pogoja. Če je pogoj izpolnjen, se izvede glavni del zanke. Ko so ukazi izvedeni, se pogoj ponovno preveri in tako naprej, dokler pogoj ni več izpolnjen, nato se zanka zaključi[36].

```
stejemo <- 0 # definiramo objekt
while(stejemo < 5) { # definiramo pogoj zanke while
    print(stejemo) # ukazi v zanki
     stejemo <- stejemo + 1
 } # konec
```

```
## [1] 0
## [1] 1
## [1] 2
## [1] 3
## [1] 4
```

[36] https://www.guru99.com/r-while-loop.html.

Kontrolne strukture, kot so *if*, *while* in *for*, omogočajo nadzor nad potekom izračunov v programu R. Uporaba zank naj bo čim bolj enostavna.

Naloga 24: Zapišite funkcijo, ki vam omogoča normalizacijo podatkov.

Naloga 25: Definirajte funkcijo, ki ima dva argumenta (vektor x in y) in ki vam izriše razsevni (scatter) graf teh dveh vektorjev, kjer sta x in y os v log merilu.

Naloga 26: Definirajte in uporabite funkcijo, ki vam izračuna povprečje in varianco vašega vzorca.

Naloga 27: Z uporabo zanke if preverite, ali so na voljo vsi dnevni podatki o pretokih z vodotoka Savinja za leto 2005. Rezultat zanke naj bo opisne oblike (character) in naj pove, ali je podatkov dovolj ali kakšen manjka.

Naloga 28: Z uporabo zanke for 5-krat naključno generirajte vektor x in vektor y (oba vektorja naj vsebujeta 10 elementov, glede na normalno porazdelitev, povprečje 0 in standardno deviacijo 1) ter oba vektorja prikažite na razsevnem diagramu z uporabo funkcije plot.

3.14 Statistični testi in porazdelitve

Vsaka porazdelitev, ki je vključena v programsko orodje R, ima štiri funkcije. Obstaja korensko ime funkcije, na primer korensko ime za normalno porazdelitev je *norm*. Pred tem korenskim imenom je ena od črk:

- *p* za kumulativno porazdelitveno funkcijo P(X manjše ali enako x);
- *q* za kvantilno funkcijo oziroma inverzno porazdelitveno funkcijo;
- *d* za točkovno verjetnost P(X = x) diskretnih oz. gostoto pX(x) zveznih porazdelitev;
- *r* za naključno vrednost, ki sledi določeni porazdelitvi.

Za normalno porazdelitev so te funkcije *pnorm*, *qnorm*, *dnorm* in *rnorm.* Za binomsko porazdelitev so te funkcije *pbinom*, *qbinom*, *dbinom* in *rbinom*. Podobno tudi za ostale porazdelitvene funkcije, ki so na voljo v programskem orodju R:

- beta porazdelitev, *beta*;
- eksponentna porazdelitev, *exp*;
- gama porazdelitev, *gamma*;
- logaritemsko normalna porazdelitev, *lnorm*;
- logistična porazdelitev, *logis*;
- Poissonova porazdelitev, *pois*;
- enakomerna porazdelitev, *unif*;
- Studentova t porazdelitev, *t*;
- Weibullova porazdelitev, *weibull*;
- f porazdelitev, *f*;
- itd.

Veliko porazdelitev, ki se uporabljajo v hidrologiji, je na voljo tudi v številnih dodatnih paketih. Paketa *lmom* in *lmomco* sta na primer namenjena oceni parametrov po metodi momentov L in vsebujeta številne dodatne porazdelitvene funkcije. Nekateri parametri porazdelitev imajo predpisane vrednosti (npr. povprečje in standardna deviacija pri normalni porazdelitvi), vendar jih lahko spremenimo, po drugi strani pa so nekateri parametri odvisni od drugih (npr. parameter scale pri porazdelitvi gama; 1/rate).

Za zvezno porazdelitev, kot je normalna, sta najbolj uporabni funkciji za reševanje problemov, ki vključujejo izračun verjetnosti, funkciji *prnorm* in *qnorm*, saj lahko gostoto verjetnosti, izračunano s funkcijo *dnorm*, uporabimo za izračun verjetnosti le s pomočjo integralov. Za diskretno porazdelitev, kot je binomska porazdelitev, funkcija *dbinom* izračuna gostoto verjetnosti, ki je v tem primeru verjetnost f(x) = P(X = x) in je zato pogosto uporabna pri izračunih. Poglejmo naslednji primer, kjer nas zanima P(X=3), kjer je slučajna spremenljivka X porazdeljena binomsko Bi(10, 0.4):

```
dbinom(3, size=10, prob=0.4)
```

```
## [1] 0.2149908
```

Predpostavimo, da obravnavamo temperaturo zraka, ki jo lahko ustrezno opišemo z normalno porazdelitvijo, srednja vrednost znaša 5 stopinj in standardna deviacija 8 stopinj. Kolikšen je 90. percentil podatkov o temperaturi zraka? Generiramo lahko tudi pet podatkov, ki sledijo tej porazdelitvi.

```
qnorm(0.9, mean=5, sd=8) # izvedemo izračun
```

```
## [1] 15.25241
```

```
rnorm(5, mean=5, sd=8) # generiramo pet naključnih števil
```

```
## [1]  2.707183  8.037603  7.369455 22.247355  4.864252
```

```
rnorm(5, mean=5, sd=8) # ponovimo še enkrat
```

```
## [1]  1.145086 -7.040315 10.442254  2.344917  1.650050
```

```
set.seed(30) # če želimo vsakič generirati ista števila
rnorm(2, mean=5, sd=8) # ponovimo še enkrat
```

```
## [1] -5.308146  2.218485
```

Vzorčenju je namenjena funkcija *sample*, ki vrne slučajno permutacijo vektorja x:

```
sample(x=1:5, size=2) # vzorec brez ponavljanja ustrezne velikosti iz x
```

```
## [1] 5 4
```

```
sample(x=1:5, size=4, replace=TRUE) # vzorec iz x s ponavljanjem
```

```
## [1] 2 3 2 5
```

Pogosto se pri analizah podatkov srečamo tudi z uporabo različnih statističnih testov. Seznam testov, vključenih v osnovni program R, je relativno obsežen, omenimo zgolj nekaj pogosto uporabljenih testov:

- *cor.test*;
- *chisq.test*;
- *kruskal.test*;
- *ks.test*;
- *poisson.test*;
- *t.test*;
- *wilcox.test*;
- *var.test*.

Poglejmo na primer pogosto uporabljeni *t.test*. V tem primeru testiramo hipotezo, da imata spremenljivki, definirani na različnih populacijah, enako povprečje (ničelna hipoteza trdi, da je mu1 = mu2, kjer sta mu1 in mu2 povprečji), pri čemer privzamemo, da sta vsaj približno normalno porazdeljeni, *alternative = two.sided* pomeni, da alternativna hipoteza pravi, da je mu1 različen od mu2; izbira *less* pomeni, da alternativna hipoteza trdi, da je mu1 manjši od mu2, oziroma izbira *greater* pomeni, da alternativna hipoteza določa, da je mu1 večji od mu2.

```
t.test(rnorm(15,10,2), rnorm(15,100,20), alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  rnorm(15, 10, 2) and rnorm(15, 100, 20)
## t = -17.659, df = 14.206, p-value = 4.576e-11
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -98.79112 -77.41894
## sample estimates:
## mean of x mean of y
##  8.831213 96.936245
```

Vidimo, da je izračunana p-vrednost manjša od 0,05, kar pomeni, da je smiselno zavrniti ničelno hipotezo (H0: povprečje je enako uporabljenemu vzorcu). Če p-vrednost ne bi bila manjša od 0,05, bi lahko sklepali, da ni smiselno zavrniti ničelne hipoteze, kot v naslednjem primeru:

```
t.test(rnorm(15,10,2), rnorm(15,16,4), alternative = "two.sided")
```

```
##
##  Welch Two Sample t-test
##
## data:  rnorm(15, 10, 2) and rnorm(15, 16, 4)
## t = -4.9492, df = 23.55, p-value = 4.97e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -9.426010 -3.874035
## sample estimates:
```

```
## mean of x mean of y
##  9.711689 16.361711
```

Omenimo lahko tudi neparametrični test Mann-Whitney, kjer ničelna hipoteza pravi, da spremenljivki pripadata isti porazdelitvi, in alternativna hipoteza, da ima spremenljivka ene skupine večje vrednosti od spremenljivke druge skupine[37].

```
# primer testa Mann-Whitney
wilcox.test(x=rnorm(100,10,2), y=runif(200,20,30), paired = FALSE)

##
##  Wilcoxon rank sum test with continuity correction
##
## data:  rnorm(100, 10, 2) and runif(200, 20, 30)
## W = 0, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Ali pa testiranje odvisnosti z uporabo funkcije *cor.test*, kjer argument *alternative* pove, kaj je alternativna hipoteza. Izbira *two.sided* pomeni, da alternativna hipoteza določa, da sta spremenljivki odvisni, izbira *less* pomeni, da alternativna hipoteza pravi, da sta spremenljivki pozitivno povezani, izbira *greater* pa pomeni, da alternativna hipoteza določa, da sta spremenljivki negativno povezani:

```
cor.test(x=runif(15,2,10), y=runif(15,20,30), alternative =  "two.sided")

##
##  Pearson's product-moment correlation
##
## data:  runif(15, 2, 10) and runif(15, 20, 30)
## t = 0.6245, df = 13, p-value = 0.5431
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.3743235  0.6280225
## sample estimates:
##       cor
## 0.1706647
```

V osnovi funkcija *cor.test* izračuna Pearsonov koeficient korelacije. Z uporabo argumenta *method* pa lahko izračunamo tudi Spearmanov ali Kendallov korelacijski koeficient. Dodatno lahko z uporabo argumenta *conf.level* izračunamo tudi interval zaupanja za korelacijski koeficient pri izbrani stopnji zaupanja. Primer v nadaljevanju pokaže visoko odvisnost med pretoki in vodostaji od vrednosti vodostajev, kar je pričakovano, saj so vrednosti pretokov izračunane na podlagi pretočne krivulje vodomerne postaje. Prikazan je tudi primer za Kendallov korelacijski koeficient. Izračun p-vrednosti za Kendallov korelacijski koeficient je v nekaterih primerih relativno zahteven in funkcija *cor.test* tako določi, v katerih primerih izračuna natančno vrednost in kdaj približek. Smiselno je biti pozoren na opozorilo (*Warning*

[37] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=107957&lang=eng.

message). Postopek izračuna lahko spremenimo z uporabo argumenta *exact,* kot je to prikazano v zadnjem primeru.

```
cor.test(x=podatki[1:20,2], y=podatki[1:20,3], method="pearson", conf.level =
0.05)

##
##  Pearson's product-moment correlation
##
## data:  podatki[1:20, 2] and podatki[1:20, 3]
## t = 41.216, df = 18, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 5 percent confidence interval:
##  0.9945819 0.9949009
## sample estimates:
##       cor
## 0.9947438

cor.test(x=podatki[1:20,2], y=podatki[1:20,3], method="kendall")

## Warning in cor.test.default(x = podatki[1:20, 2], y = podatki[1:20, 3], :
Cannot
## compute exact p-value with ties

##
##  Kendall's rank correlation tau
##
## data:  podatki[1:20, 2] and podatki[1:20, 3]
## z = 6.0862, p-value = 1.156e-09
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
## tau
##   1

cor.test(x=podatki[1:20,2], y=podatki[1:20,3], method="kendall", exact=FALSE)

##
##  Kendall's rank correlation tau
##
## data:  podatki[1:20, 2] and podatki[1:20, 3]
## z = 6.0862, p-value = 1.156e-09
## alternative hypothesis: true tau is not equal to 0
## sample estimates:
## tau
##   1
```

Veliko statističnih testov je na voljo tudi v drugih paketih (npr. *Hmisc, e1071, rcompanion*).

Naloga 29: Izrišite graf dveh naključno generiranih vzorcev. Oba vzorca generirajte z uporabo normalne porazdelitve: enkrat povprečje 0 in standardna deviacija 2, drugič

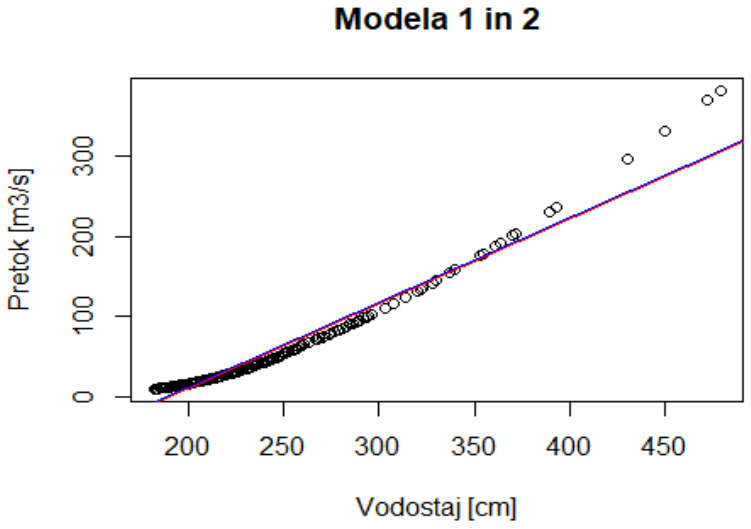
povprečje 3 in standardna deviacija 5. Na grafu naj bo 9 točk. Graf ustrezno opremite z oznakami, osmi, legendo itd.

Naloga 30: Poiščite ustrezen test, ki ga lahko uporabite za izračun, ali je trend v vzorcu statistično značilen ali ne. Naložite ustrezen paket in uporabite test z enim naključno generiranim vzorcem na podlagi normalne porazdelitve s srednjo vrednostjo 10 in standardno deviacijo 5 (generirajte 20 števil). Postopek ponovite za zaporedje števil od 1:20.

3.15 Preprosti modeli

Pogosto se v vodarstvu srečujemo z linearnimi in nelinearnimi modeli, na primer podatke o pretokih se večinoma izračuna na podlagi podatkov o vodostaju z uporabo pretočne krivulje.

```
# linearni model med pretokom in vodostajem
mod1 <- lm(Pretok ~ Vodostaj, podatki)
# vključimo še temperaturo vode
mod2 <- lm(Pretok ~ Vodostaj + Temperatura, podatki)
plot(podatki$Vodostaj,podatki$Pretok, xlab="Vodostaj [cm]", ylab="Pretok [m3/
s]", main="Modela 1 in 2")
# dodamo povezavo na podlagi prvega linearnega modela
abline(mod1, col="red")
# preverimo, ali podatek o temperaturi vode pomaga
abline(mod2, col="blue")

## Warning in abline(mod2, col = "blue"): only using the first two of 3 regre
ssion
## coefficients
```



Slika 24: Prikaz linearnega modela s Q-H krivuljo.

Linearni model, ki smo ga definirali, je ločena vrsta objekta v programskem okolju R. V povezavi s temi objekti lahko neposredno uporabimo različne praktične funkcije, kot so *anova*, *coef*, *deviance*, *formula*, *predict*, *plot*, *print*, *residuals*, *proj*, *summary*. Poglejmo nekaj primerov:

```
print(mod1) # koeficienti modela
```

```
##
## Call:
## lm(formula = Pretok ~ Vodostaj, data = podatki)
##
## Coefficients:
## (Intercept)     Vodostaj
##    -201.515        1.058
```

```
summary(mod1) # glavne značilnosti modela
```

```
##
## Call:
## lm(formula = Pretok ~ Vodostaj, data = podatki)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.292  -8.891  -2.693   5.978  76.735
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -201.51498    3.06039  -65.85   <2e-16 ***
## Vodostaj       1.05821    0.01288   82.14   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.04 on 335 degrees of freedom
## Multiple R-squared:  0.9527, Adjusted R-squared:  0.9526
## F-statistic:  6747 on 1 and 335 DF,  p-value: < 2.2e-16
```

```
formula(mod1) # uporabljena enačba
```

```
## Pretok ~ Vodostaj
```

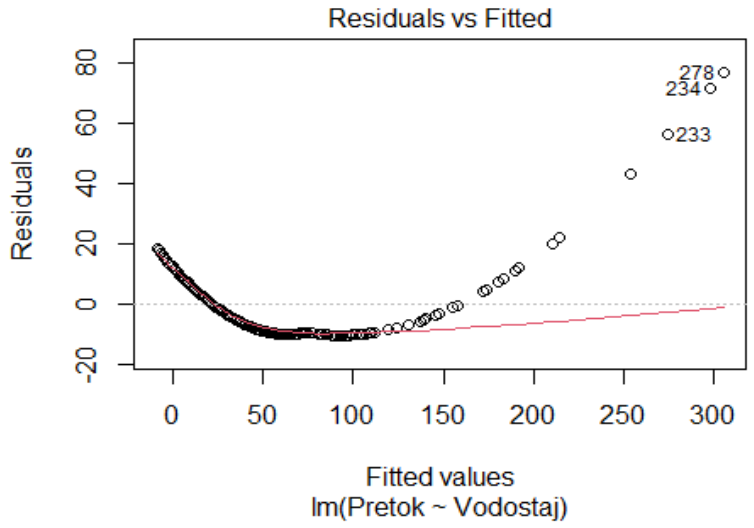
```
head(residuals(mod1)) # reziduali uporabljenega modela
```

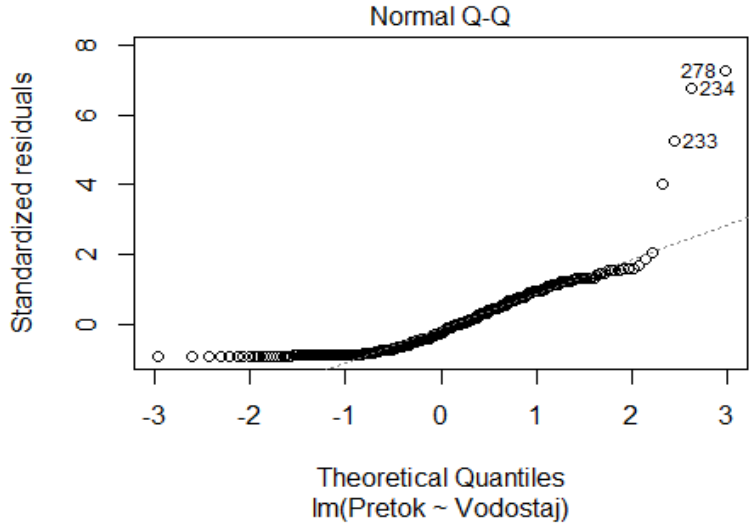
```
##         1         2         3         4         5         6
## -8.122995 -7.415380 -6.302560 -4.275330 -3.105715 -1.834100
```

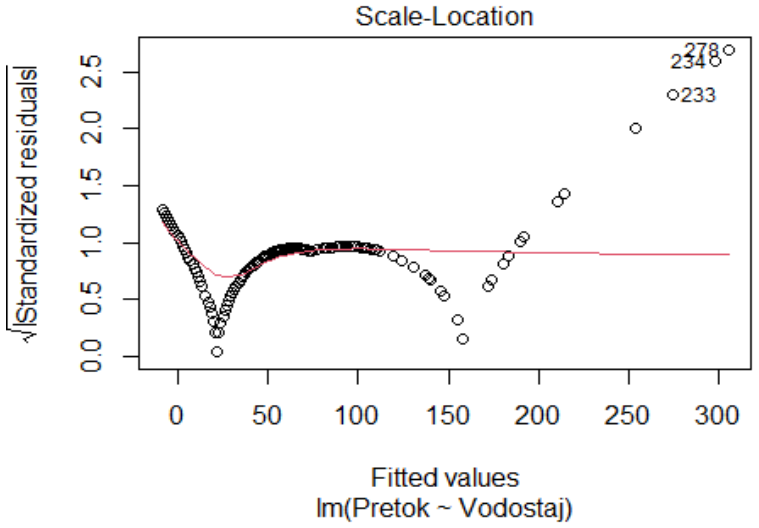
```
head(predict(mod1)) # napovedi uporabljenega modela 1
```

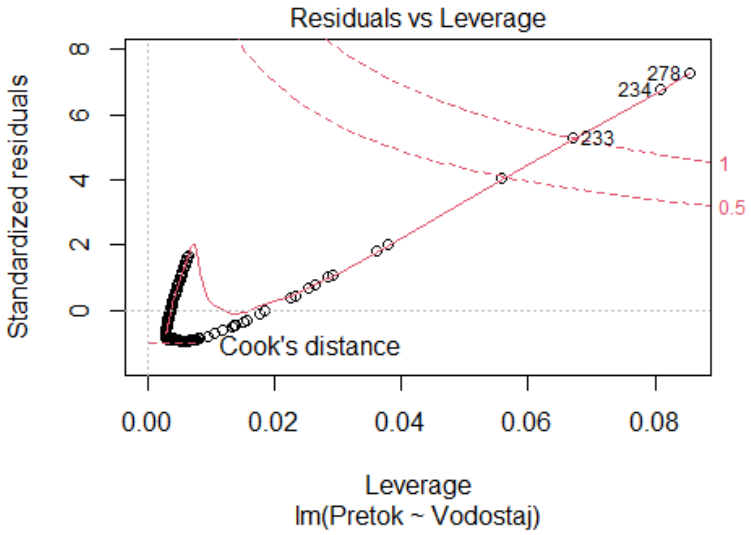
```
##        1        2        3        4        5        6
## 46.10499 42.93038 38.69756 32.34833 29.17371 25.99910
```

```
plot(mod1) # nekaj avtomatsko določenih grafov
```


Residuals vs Fitted
Residuals
Fitted values
lm(Pretok ~ Vodostaj)
278
234
233


Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Pretok ~ Vodostaj)
278
234
233

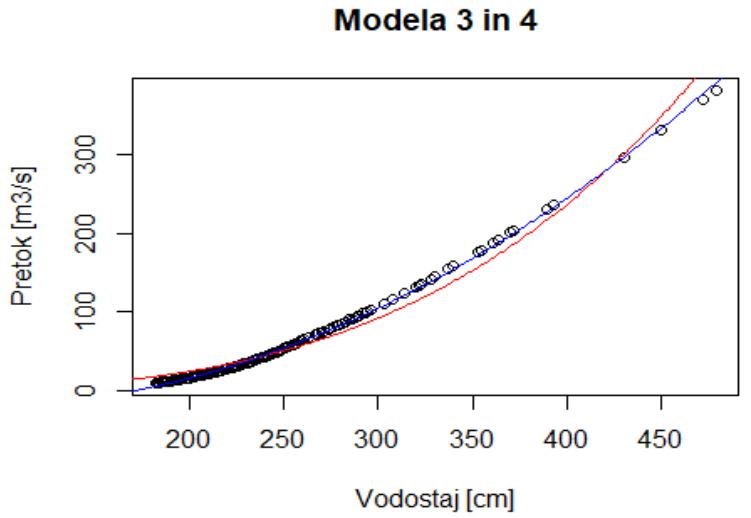

Scale-Location
√|Standardized residuals|
Fitted values
lm(Pretok ~ Vodostaj)
278
234
233



Slika 25: Različni grafi, ki prikazujejo ujemanje linearnega modela.

Na enak način lahko definiramo tudi bolj kompleksne (nelinearne) modele. Pri ocenjevanju parametrov so zelo pomembne začetne vrednosti parametrov, ki morajo biti ustrezno ocenjene:

```
mod3 <- nls(Pretok ~ a*(Vodostaj)^b, data = podatki,
    start = list(a=0.0000005,b=3)) # model 3
mod4 <- nls(Pretok ~ a*(Vodostaj)^2+b*(Vodostaj)+c, data = podatki,
    start = list(a=0.005,b=3,c=2)) # model 4
# izrišemo preprost graf
plot(podatki$Vodostaj,podatki$Pretok, xlab="Vodostaj [cm]",
  ylab="Pretok [m3/s]", main="Modela 3 in 4")
lines(150:500,coef(mod3)[1]*(150:500)^(coef(mod3)[2]),col="red")
# grafično ujemanje modela 3
lines(150:500,coef(mod4)[1]*(150:500)^2+coef(mod4)[2]*(150:500)+coef(mod4)[3]
,col="blue")
```



Slika 26: Testiranje dodatnih nelinearnih modelov za opis Q-H krivulje.

```
# grafično ujemanje modela 4
# izračunajmo še nekaj statistik, R2
RSS.p <- sum(residuals(mod3)^2)
TSS <- sum((podatki$Pretok - mean(podatki$Pretok))^2)
r2 <- 1 - (RSS.p/TSS)
r2 # r2 za model 3

## [1] 0.9695111

RSS.p <- sum(residuals(mod4)^2)
TSS <- sum((podatki$Pretok - mean(podatki$Pretok))^2)
r2 <- 1 - (RSS.p/TSS)
r2 # r2 za model 4

## [1] 0.998363
```

Vidimo, da ta zadnji model zelo dobro opisuje dejanske podatke, in verjetno je izbrana funkcija zelo podobna pretočni krivulji, ki je bila uporabljana za določitev pretokov na podlagi merjenih vodostajev.

Obstajajo tudi kompleksnejše funkcije, kot je npr. *gls* (paket *nlme*), ki jih lahko uporabimo za definiranje različnih linearnih in nelinearnih modelov. Nekaj zanimivih primerov je na voljo tudi na spletu[38].

Naloga 31: Najdite ustrezno funkcijo oziroma model, ki opiše povezavo med pretokom in vsebnostjo suspendiranih snovi tako, da je r2 večji od 0,3. Uporabite podatke z vodomerne postaje Veliko Širje na Savinji.

---

[38] https://www.guru99.com/r-simple-multiple-linear-regression.html.

4 Hidrološke analize in modeliranje

V naslednjih poglavjih so prikazane nekatere pogosto uporabljene metode in analize na področju vodarstva. Prikazane bodo metode za analize nizkih in visokih pretokov, multivariatne analize in analize trendov in sezonskosti. Dodatno bo poudarek tudi na analizah padavinskih podatkov, stohastičnem padavinskem modelu in erozivnosti padavin ter modeliranju površinskega odtoka z uporabo različnih metod. Dodatno bo prikazan postopek uporabe prostorskih podatkov in analize podnebnih sprememb.

4.1 Analize nizkih pretokov

Analize nizkih pretokov v hidrologiji so eden izmed ključnih vidikov upravljanja vodnih virov[39], kjer se osredotočamo na razumevanje in količinsko opredelitev značilnosti vodnatosti rek in potokov. To je še posebej pomembno v regijah, ki se soočajo s pomanjkanjem vode in hidrološko sušo, saj analize nizkih pretokov lahko pomagajo pri odločitvah, povezanih z dovoljenji za rabo in povratni in nepovratni odvzem vode, načrtovanjem infrastrukture in varstvom okolja. Nizki pretoki se običajno pojavijo v sušnih obdobjih ali v regijah z omejeno razpoložljivostjo vode. Analiza nizkih pretokov je pomembna iz več razlogov, kot so: (i) upravljanje vodnih virov (ocena razpoložljive vode), (ii) načrtovanje hidrotehnične infrastrukture (pregrade, zadrževalniki), (iii) varstvo okolja (vpliv nizkih pretokov na vodne ekosisteme), (iv) ocena tveganja (načrtovanje ukrepov v primeru nizkih razmer). Za razumevanje in izvedbo analiz hidrološke suše je na voljo veliko literature, pri čemer velja posebej izpostaviti monografijo Hydrological Drought[40]. V tem poglavju se bomo posebej osredotočili na nekatere metode, ki so vključene v paket *lfstat*[41]. Paket *lfstat* temelji na priročniku Svetovne meteorološke organizacije (WMO)[42].

Za analize bomo uporabili podatke z vodomerne postaje Veliko Širje na reki Savinji, ki smo jih uporabili že v prejšnjih poglavjih. Pred uporabo paketa *lfstat* ne pozabite na namestitev paketa z uporabo funkcije *install.packages* ali preko grafičnega vmesnika Rstudio ter na njegovo aktivacijo (npr. uporaba funkcije *library*).

```
library(lfstat, quietly=TRUE); library(zoo, quietly=TRUE)

## Warning: package 'lfstat' was built under R version 4.1.3

podatki <- read.table(file="C:/Users/nbezak/OneDrive - Univerza v Ljubljani/U
cbenik/Savinja-Veliko SirjeI-2005.txt",header=TRUE,sep=";",dec=".")
# uvozimo podatke in spremenimo imena stolpcev
names(podatki) <- c("Datum", "Vodostaj", "Pretok", "Temperatura", "Transport"
, "Vsebnost")
```

[39] https://actahydrotechnica.fgg.uni-lj.si/si/paper/a46mp.

[40] https://shop.elsevier.com/books/hydrological-drought/tallaksen/978-0-12-819082-1.

[41] https://cran.r-project.org/web/packages/lfstat/index.html.

[42] https://library.wmo.int/idurl/4/32176.

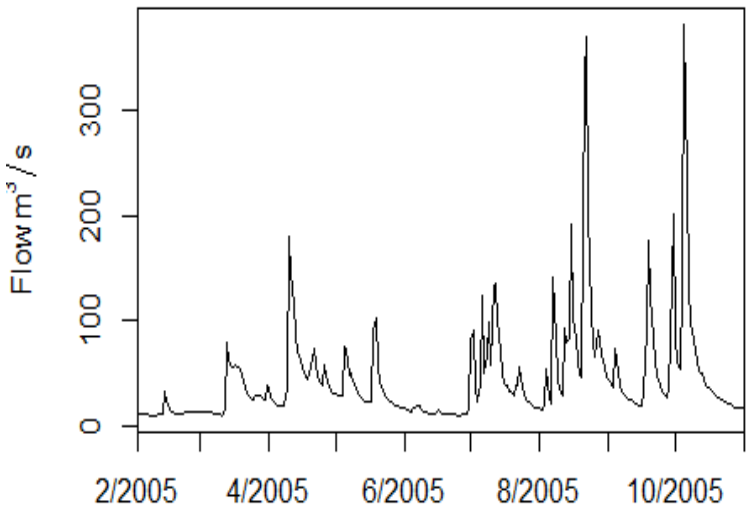
```
# datume preoblikujemo v format as.POSIXct
podatki[,1] <- as.POSIXct(strptime(podatki[,1],format="%d.%m.%Y"))
# definiramo objekt zoo
Qzoo <- zoo(podatki[,3],podatki[,1])
# objekt lfobj, argument hyeastart določa začetek hidrološkega leta
nizkiQ <- createlfobj(ts(Qzoo), startdate = "01/01/2005", hyearstart = 1)
setlfunit("m^3/s") # definiramo enote svojih podatkov
head(nizkiQ) # poglejmo prvih nekaj vrstic objekta
```

```
##   day month year   flow hyear baseflow
## 1   1     1 2005 37.982  2005       NA
## 2   2     1 2005 35.515  2005       NA
## 3   3     1 2005 32.395  2005       NA
## 4   4     1 2005 28.073  2005       NA
## 5   5     1 2005 26.068  2005       NA
## 6   6     1 2005 24.165  2005       NA
```

```
summary(nizkiQ) # poglejmo osnovne statistike
```

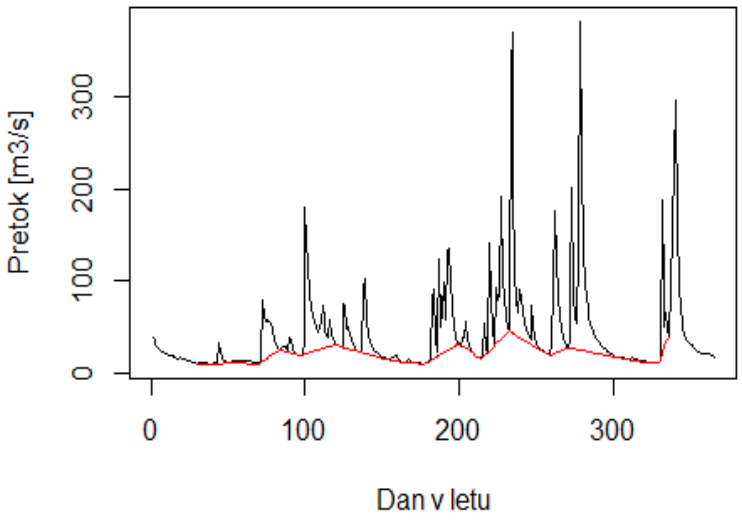
```
## Startdate:  2005-01-01     (calendar year)
## Enddate:    2005-12-31     (calendar year)
##
## Time series covers 1 years.
## The hydrological year is set to start on January 1st.
##
## Meanflow     MAM7      Q95      BFI
##  42.5100   9.8630  10.4600   0.4687
```

```
# dobimo nekatere značilnosti, kot je indeks BFI ali MAM7 in pretok Q95
# paket vsebuje funkcijo hydrograph, s katero lahko izrišemo hidrogram
hydrograph(nizkiQ,startdate = "01/02/2005", enddate = "31/10/2005")
```



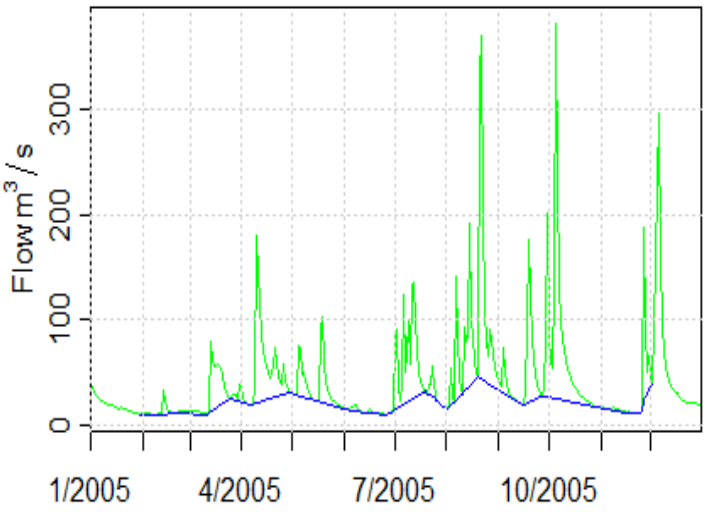
Slika 27: Hidrogram Savinja – Veliko Širje.

```
# prikažemo lahko tudi rezultate izločanja baznega odtoka (rdeča črta)
plot(nizkiQ$flow,type="l",xlab="Dan v letu",ylab="Pretok [m3/s]")
lines(nizkiQ$baseflow,col="red")
```



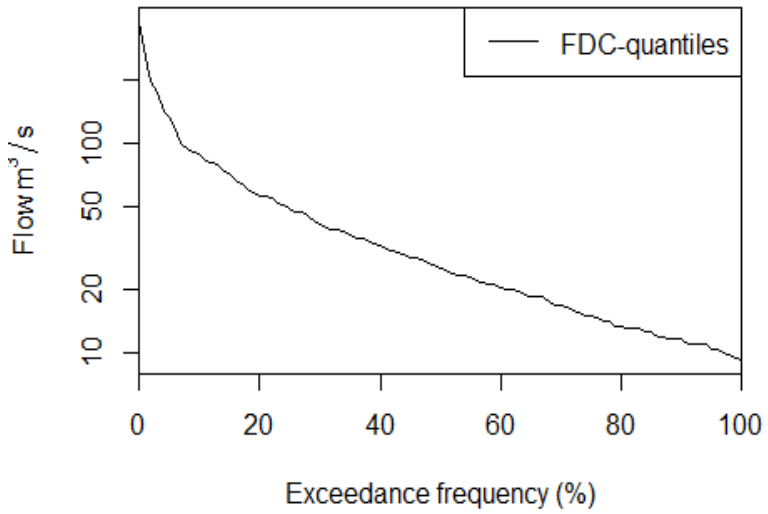
Slika 28: Hidrogram in izločanje baznega odtoka.

```
# oziroma direktno z uporabo funkcije v paketu lfstat
bfplot(nizkiQ)
```



Slika 29: Prikaz izločanja baznega odtoka na alternativen način.

```
## NULL

# izrišemo krivuljo trajanja naših pretokov
# funkcija fdc nam izpiše vse percentile pretokov
fdc(nizkiQ)
```



Slika 30: Krivulja trajanja za postajo Veliko Širje na Savinji.

```
##                  100%       99%       98%       97%      96%      95%       94%
93%
## FDC-quantiles 382.1 258.0889 198.5345 176.4337 143.076 133.593 117.7452 10
1.205
##                     92%      91%      90%     89%      88%       87%      86%
85%
## FDC-quantiles 95.33348 91.5466 89.0422 82.374 81.6566 78.89384 74.1346 72.
4836
##                   84%       83%     82%  81%      80%       79%      78%       77
%
## FDC-quantiles 67.155 64.38472 60.765 58.5 56.5106 55.81448 55.1224 52.3310
4
##                     76%     75%     74%     73%     72%      71%    70%     69%
68%
## FDC-quantiles 50.64424 48.992 47.027 47.027 45.191 42.7454 41.27 39.689 38
.83
##                     67%     66%      65%     64%       63%       62%     61%     6
0%
## FDC-quantiles 38.72824 37.982 36.8182 35.515 34.97236 34.46548 33.157 32.3
95
##                    59%     58%     57%     56%     55%     54%       53%       52%
## FDC-quantiles 31.646 30.908 30.182 29.467 28.764 28.764 28.01868 26.91232
##                     51%     50%      49%     48%       47%     46%     45%     44%
## FDC-quantiles 26.48848 25.423 25.0166 24.165 23.60196 23.553 22.952 22.362
##                    43%     42%     41%     40%     39%     38%     37%     36%    35
%
## FDC-quantiles 21.782 21.214 21.214 20.656 20.108 20.108 19.572 19.045 18.5
3
##                  34%    33%     32%     31%     30%     29%     28%       27%     2
6%
## FDC-quantiles 18.53 18.53 17.529 17.044 17.044 16.569 16.104 15.77712 15.2
05
```

```
##                   25%   24%    23%    22%    21%    20%      19%    18%
17%
## FDC-quantiles 15.205 14.77 14.344 14.344 13.523 13.523 13.18952 13.126 13.
126
##                   16%     15%    14%    13%    12%    11%    10%     9%
8%
## FDC-quantiles 12.739 12.5882 11.994 11.994 11.654 11.654 11.654 11.006 11.
006
##                    7%     6%      5%   4%     3%      2%      1%    0%
## FDC-quantiles 11.006 11.006 10.4596 10.4 10.113 9.91356 9.47476 9.309

# izračunamo določene vrednosti kvantilov
# za letne vrednosti (yearly) in za mesečne vrednosti argument monthly
Qxx(nizkiQ, c(50,90,95,99),monthly=TRUE)

##    month flow.50% flow.10%  flow.5%  flow.1%
## 1      1 16.56900 11.65400 11.33000 10.18700
## 2      2 11.65400  9.30900  9.30900  9.30900
## 3      3 27.39400 11.65400 11.65400 10.38140
## 4      4 43.26200 19.46780 18.53000 18.53000
## 5      5 28.76400 19.04500 18.02950 17.52900
## 6      6 11.99400 10.11300  9.96065  9.64572
## 7      7 38.83000 20.10800 17.78700 17.04400
## 8      8 67.15500 20.65600 18.85000 15.75670
## 9      9 35.93200 21.21400 20.31090 18.83218
## 10    10 37.14700 17.52900 16.80650 16.56900
## 11    11 13.93350 11.00600 10.67270 10.40000
## 12    12 30.18200 20.10800 19.05100 17.48980
```

Paket *lfstat* vsebuje tudi številne druge funkcije za analize nizkih pretokov. Zelo uporabne so lahko funkcije:

- *BFI* za izračun indeksa BFI (razmerje med baznim odtokom in celotnim odtokom);
- *MAM* za izračun najnižjih letnih pretokov;
- *meanflow* za izračun povprečnega letnega pretoka;
- *Q95*, *Q90*, *Q70*, *Qxx* za izračun določenih kvantilnih pretokov;
- *recession* za analizo recesijskih konstant (padajočih delov hidrogramov);
- *seasindex* za izračun indeksa sezonskosti;
- *seasratio* za izračun sezonskega razmerja (razmerje med Q95 v dveh obdobjih, lahko poletje in zima).

Izvedemo pa lahko tudi nekoliko naprednejše izračune ali analize[43], na primer verjetnostno analizo nizkih pretokov. Uporabimo podatke v paketu *lfstat*. Ker potrebujemo več kot 1 leto podatkov, najprej določimo najnižje pretoke za vsa leta podatkov, nato glede na podatke definiramo Weibullovo porazdelitev na podlagi momentov L, možno je izbrati tudi druge

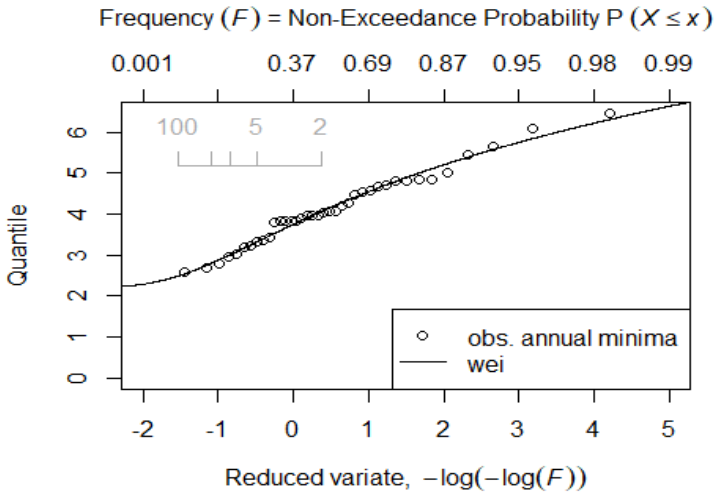
[43] https://doi.org/10.15292/acta.hydro.2019.01.

porazdelitvene funkcije, ki so vključene v paket *lmom*. Parameter *event* določa izbrano povratno dobo in nato lahko ocenimo nizki pretok s 100-letno povratno dobo.

```
data(ngarurоro)
nletni <- MAM(ngarurоro,n=1,yearly=TRUE)
head(nletni)
```

```
##   hyear   MAn
## 1  1964 3.344
## 2  1965 4.805
## 3  1966 5.019
## 4  1967 4.819
## 5  1968 3.210
## 6  1969 3.832
```

```
weibullpor <- tyears(ngarurоro, dist = "wei", event = 100, plot = TRUE)
```



Slika 31: Rezultati verjetnostnih analiz nizkih pretokov.

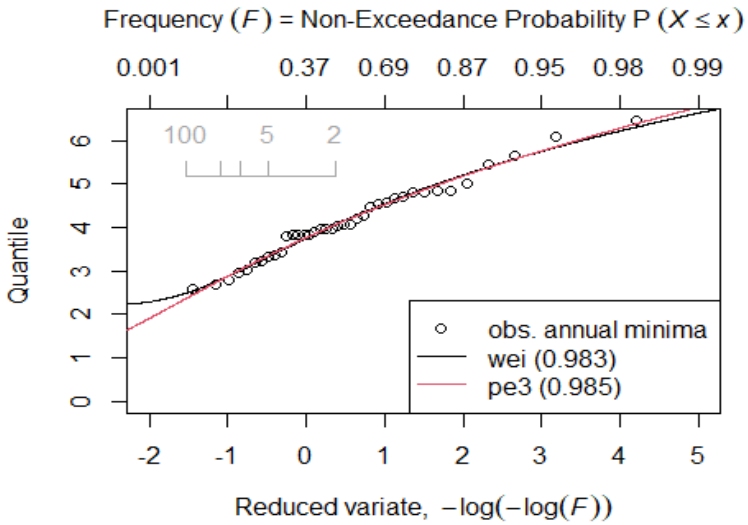
```
weibullpor
```

```
##              distribution
## return period      wei
##           100 2.494827
```

```
# ocenjene vrednosti parametrov Weibullove porazdelitve in L-momenti vzorca
summary(weibullpor)
```

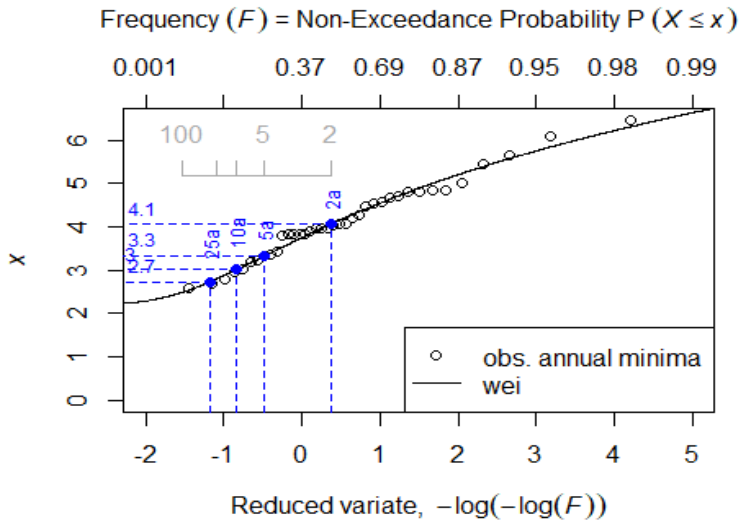
```
## Values: num [1:38(1d)] 3.34 4.8 5.02 4.82 3.21 ...
##
##
## L-Moments:
##        l_1        l_2        t_3        t_4
## 4.13686842 0.50741892 0.08370583 0.16018348
##
```

```
## Fitted Parameters of the Distribution:
## wei    zeta:   2.20731,  beta:   2.17832,  delta:  2.27166

# preverimo lahko več porazdelitev
vecpor <- tyears(ngaruroro, dist = c("wei", "pe3"), event = 100, plot = TRUE)

## Warning: For fitting minima, a Weibull distribution with parameter 'zeta =
0'
## may be best.
```



Slika 32: Uporaba dveh različnih porazdelitev za verjetnostne analize nizkih pretokov.

```
# naredimo nekaj izračunov za določene vrednosti povratnih dob
povratna <- c(2, 5, 10, 25)
# izračunamo vrednosti kvantilov glede na povratne dobe
evquantile(weibullpor, return.period = povratna)

##              distribution
## return period      wei
##            2  4.061063
##            5  3.332857
##            10 3.016214
##            25 2.740186

plot(weibullpor) # še enkrat izrišemo graf
# dodamo oznake za izbrane vrednosti povratnih dob na graf
rpline(weibullpor, return.period = povratna)
```



Slika 33: Oznaka izbranih vrednosti povratnih dob.

S paketom *lfstat* lahko analiziramo tudi različne vidike hidrološke suše, na primer iščemo sušna obdobja glede na izbrani kriterij. V spodnjem primeru je to pretok *Q90* (argument *probs*) ali *Q95*.

```
susa <- find_droughts(nizkiQ, threshold = quantile, probs = 0.1, na.rm = TRUE
)
summary(susa) # glede na kriterij določimo štiri večja sušna obdobja
```

```
## Summary of droughts
## River:  at
## Units: volumes in m3, duration in days
##
## Filtered 5 minor events of 9 total.
## Only droughts with volume >= 7320.672 m3 and duration >= 5 days are report
ed.
##
##   event.no      start       time        end    volume duration dbt       v
bt
## 1        1 2005-01-28 2005-02-12 2005-02-12 1464134.4       16  16 1464134
.4
## 3        3 2005-03-07 2005-03-11 2005-03-11  157075.2        5   5  157075
.2
## 5        5 2005-06-18 2005-06-22 2005-06-22  459129.6        5   5  459129
.6
## 9        9 2005-11-18 2005-11-25 2005-11-25  496627.2        8   8  496627
.2
##     qmin      tqmin
## 1  9.309 2005-02-06
## 3  9.836 2005-03-11
## 5 10.113 2005-06-21
## 9 10.400 2005-11-23
```

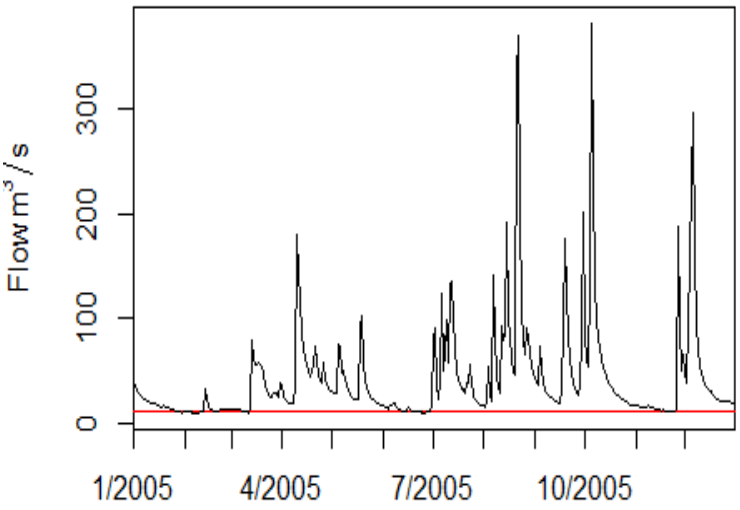
```
# uporabimo še pretok Q95 (argument probs)
susa1 <- find_droughts(nizkiQ, threshold = quantile, probs = 0.05, na.rm = TR
UE)
summary(susa1, drop_minor=0) # prikažemo vse dogodke
```

```
## Summary of droughts
## River:  at
## Units: volumes in m3, duration in days
##
##   event.no      start       time        end    volume duration dbt       v
bt
## 1        1 2005-01-30 2005-01-30 2005-01-30  53879.04        1   1  53879.
04
## 2        2 2005-02-05 2005-02-09 2005-02-09 402796.80        5   5 402796.
80
## 3        3 2005-02-11 2005-02-11 2005-02-11   5149.44        1   1   5149.
44
## 4        4 2005-02-19 2005-02-20 2005-02-20  10298.88        2   2  10298.
88
## 5        5 2005-03-11 2005-03-11 2005-03-11  53879.04        1   1  53879.
04
## 6        6 2005-06-20 2005-06-22 2005-06-22  65041.92        3   3  65041.
92
## 7        7 2005-06-24 2005-06-26 2005-06-26 160859.52        3   3 160859.
52
## 8        8 2005-06-29 2005-06-29 2005-06-29  29946.24        1   1  29946.
24
## 9        9 2005-11-23 2005-11-24 2005-11-24  10298.88        2   2  10298.
88
##     qmin      tqmin
## 1  9.836 2005-01-30
## 2  9.309 2005-02-06
## 3 10.400 2005-02-11
## 4 10.400 2005-02-19
## 5  9.836 2005-03-11
## 6 10.113 2005-06-21
## 7  9.568 2005-06-26
## 8 10.113 2005-06-29
## 9 10.400 2005-11-23
```

```
# izrišemo lahko interaktivni graf teh rezultatov z uporabo plot(susa1)
# uporabimo lahko spreminjajočo se mejno vrednost glede na obdobje
# recimo mesečne, tedenske, dnevne vrednosti
susa2 <- find_droughts(nizkiQ, threshold = vary_threshold(nizkiQ,
varying = "monthly", fun = function(x) quantile(x, probs = 0.05, na.rm = T)))
# preverimo glavne značilnosti suš, določenih glede na zgornji postopek
summary(susa2)
```

```
## Summary of droughts
## River:  at
## Units: volumes in m3, duration in days
```

```
##
## Filtered 10 minor events of 11 total.
## Only droughts with volume >= 2996.784 m3 and duration >= 5 days are report
ed.
##
##   event.no      start       time        end   volume duration dbt      vbt
## 3        3 2005-03-07 2005-03-11 2005-03-11 157075.2        5   5 157075.2
##    qmin      tqmin
## 3 9.836 2005-03-11

# analiziramo odvisnost med sušnimi dogodki, ki smo jih določili
summary(pool_sp(susa))

## Summary of pooled droughts
## River:  at
## Pooling: Sequent Peak, 2 were pooled
## Units: volumes in m3, duration in days
##
## Filtered 4 minor events of 7 total.
## Only droughts with volume >= 7320.672 m3 and duration >= 5 days are report
ed.
##
##   event.no      start       time        end    volume duration dbt       v
bt
## 1        1 2005-01-30 2005-02-12 2005-02-12 1464134.4       14  14 1464134
.4
## 5        5 2005-06-18 2005-06-29 2005-06-30 1086566.4       12  10 1145318
.4
## 7        7 2005-11-19 2005-11-25 2005-11-26  496627.2        7   7  496627
.2
##     qmin      tqmin
## 1  9.309 2005-02-06
## 5  9.568 2005-06-26
## 7 10.400 2005-11-23

# graf, iz katerega se vidijo obdobja, ko je pretok pod mejno vrednostjo
streamdefplot(nizkiQ,year=2005,threslevel = 95)
```

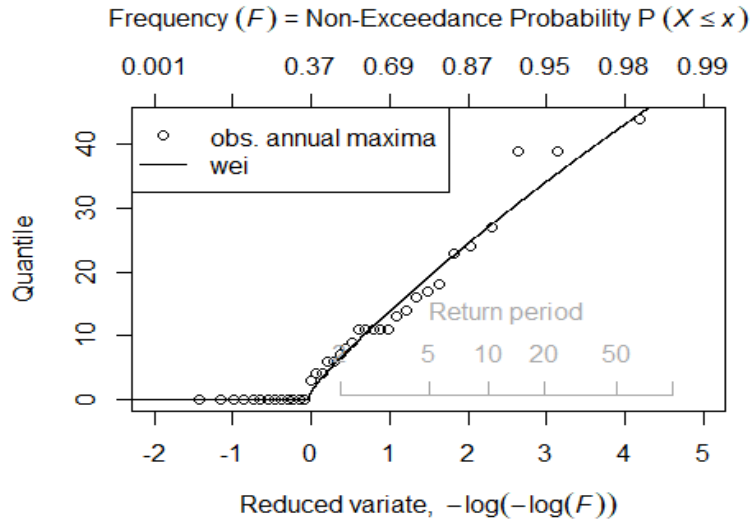


Slika 34: Prikaz obdobij, ko je pretok pod mejno vrednostjo.

```
## NULL
```

Podobno kot v prejšnjem primeru bi lahko naredili tudi verjetnostne analize ostalih vidikov suš, kot so število dni s sušnimi razmerami, maksimalni deficit ali maksimalni deficit volumna. Poglejmo primer verjetnostne analize za trajanje hidroloških suš. Uporabimo podatke v paketu *lfstat*, odločimo se za Weibullovo porazdelitev, zanima nas trajanje (*variable ="d"*, alternativa je *volumen "v"*), iščemo maksimalne vrednosti (*aggr="max"*), izračune delamo za izbrane povratne dobe ter upoštevamo tudi odvisne dogodke glede na parameter *pooling="IC"*).

```
maksTrajanje <- tyearsS(ngarururoro, dist = "wei", variable = "d",
aggr = "max", event = povratna, hyearstart = 1, pooling = "IC")
```



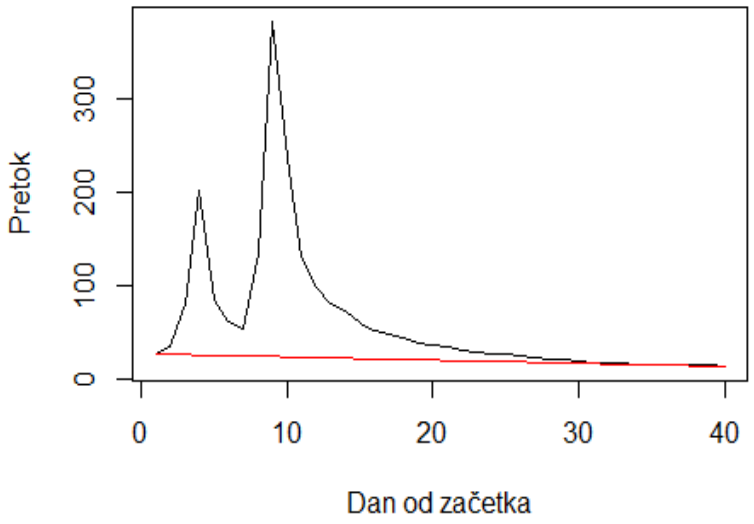
Slika 35: Verjetnostna analiza trajanja največjih suš.

Določanje baznega odtoka je pomembno z vidika poznavanja dinamike odziva porečij ter tudi za določitev volumnov visokovodnih valov. Obstajajo tudi drugi paketi in funkcije, s katerimi

lahko ocenjujemo bazni odtok (poleg paketa *lfstat*). Ena izmed funkcij je tudi *BaseflowSeparation*, ki je vključena v paket *EcoHydRology*. Ta sicer ni več na voljo v repozitoriju CRAN, so pa starejše verzije paketov še vedno na voljo na spletu[44] in se jih da namestiti z uporabo datotek *tar.gz* preko možnosti *Install* v grafičnem vmesniku Rstudio (namesto repozitorija CRAN se izbire možnost namestitve z uporabo arhivskih datotek paketa). Omenjena funkcija sicer uporablja enega izmed načinov avtomatskega ocenjevanja baznega odtoka[45].

Izločanje baznega odtoka je pogosto potrebno tudi pri analizi volumnov hidrogramov ter trajanj visokovodnih hidrogramov[46],[47]. Poglejmo preprost primer, kjer najprej poiščemo največjo vrednost pretoka v letu, nato določimo vse dneve v letu, ko je bazni odtok enak površinskemu (začetki in konci hidrogramov), nato pa izračunamo volumen visokovodnega vala.

```
nizkiQ <- createlfobj(ts(Qzoo), startdate = "01/01/2005", hyearstart = 1)
letnM <- which.max(nizkiQ$flow)
mejneT <- which(nizkiQ$flow==nizkiQ$baseflow)
# maksimalna vrednost konice pretoka je nastopila v 278. dnevu
# iz objekta mejneT vidimo, da se je val začel na 270. in končal na 309. dan
plot(nizkiQ$flow[270:309],type="l",xlab="Dan od začetka",ylab="Pretok")
lines(nizkiQ$baseflow[270:309],col="red")
```



Slika 36: Izločanje baznega odtoka visokovodnega hidrograma.

[44] https://cran.r-project.org/web/packages/EcoHydRology/index.html.

[45] https://doi.org/10.1029/WR026i007p01465.

[46] https://actahydrotechnica.fgg.uni-lj.si/si/paper/a42ms.

[47] https://sciendo.com/abstract/journals/johh/63/2/article-p134.xml.

```r
# vsota vseh pretokov odtoka
vsota <- sum(nizkiQ$flow[270:309]-nizkiQ$baseflow[270:309])
vsota*24*3600 # izračunamo lahko tudi volumen visokovodnega vala v m3

## [1] 134212464

# oziroma trajanje visokovodnega vala v dneh (ker imamo dnevne podatke)
length(nizkiQ$flow[270:309])

## [1] 40
```

Naloga 32: S spletne strani Agencija RS za okolje prenesite dnevne podatke o pretokih z vodomerne postaje Sava Šentjakob za obdobje 1990–2021. Z uporabo paketa lfstat analizirajte največje sušne dogodke (najnižji letni pretoki) v posameznem letu. Izvedite tako verjetnostne analize nizkih pretokov (Pearsonova porazdelitev tipa 3) kot analize trajanja teh sušnih dogodkov.

4.2 Verjetnostne analize visokovodnih konic

Verjetnostne analize konic pretokov so eno izmed pomembnih področij hidrološke znanosti. Njihov cilj je oceniti verjetnost pojava poplavnih dogodkov (visokovodnih konic) na določeni lokaciji. Rezultati teh analiz se imenujejo projektni pretoki in se uporabljajo pri načrtovanju hidrotehnične infrastrukture, upravljanju vodnih virov in razvoju strategij za zmanjševanje poplavne ogroženosti. Podatki o projektnih pretokih so eden izmed pomembnih rezultatov hidroloških študij in so na primer tudi eden izmed vhodnih podatkov za hidravlične analize in analize poplavne ogroženosti. Za izvedbo verjetnostnih analiz je na voljo veliko različnih metod, od metod vrednosti konic nad izbrano mejno vrednostjo – pragom (POT) do metode letnih maksimumov (AM)[48]. V sklopu tega učbenika se bomo posvetili uporabi metode AM, ki se v hidrološki praksi zelo pogosto uporablja[49].

Osnovni koraki izvedbe (univariatnih) verjetnostnih analiz visokovodnih konic so[50]:

- Izbira vzorca za izvedbo verjetnostnih analiz (npr. metoda največjih letnih visokovodnih konic (AM), če imamo na voljo dovolj dolg niz merjenih pretokov, kar pomeni, da iz dolgoletnih merjenih nizov podatkov o pretokih v vzorec uvrstimo največjo visokovodno konico v posameznem letu. V primeru krajših nizov merjenih podatkov o pretokih je priporočljiveje vzorec oblikovati po metodi vrednosti nad izbranim pragom (POT)).

---

[48] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=32640&lang=slv.

[49] https://actahydrotechnica.fgg.uni-lj.si/si/paper/a42ms.

[50] https://www.fgg.uni-lj.si/wp-content/uploads/2023/05/Koncno_porocilo_CRP_V2_2137.pdf.

- Ocena parametrov več izbranih porazdelitvenih funkcij (npr. log-normalna, Pearsonova III, log-Pearsonova III, Gumbelova, generalizirana porazdelitev ekstremnih vrednosti (GEV)) na podlagi metode momentov ali metode momentov L ali katere drugi metode).
- Ocena ustreznosti posamezne porazdelitve glede na ujemanje z merjenimi podatki o pretokih ali z uporabo grafičnih prikazov ali z uporabo statističnih testov. Pri oceni ustreznosti porazdelitve in interpretaciji rezultatov (predvsem v primeru daljših povratnih dob) je vsekakor priporočljivo uporabiti vse dodatne razpoložljive kvalitativne in/ali kvantitativne informacije ter se o ustreznosti porazdelitve in interpretaciji rezultatov odločiti na podlagi strokovnih izkušenj.

Dodatno se lahko postopek izvedbe verjetnostnih analiz dopolni z določitvijo celotnega projektnega hidrograma glede na metodo tipičnega visokovodnega hidrograma. V tem primeru se zahteva še izvedba naslednjih dveh korakov:

- Izbira ustreznih reprezentativnih hidrogramov v odvisnosti od namena uporabe rezultatov (npr. krajši dogodki, daljši dogodki, več različnih kombinacij).
- Upoštevanje rezultatov verjetnostnih analiz (skupaj s pripadajočimi intervali zaupanja, ki zajemajo vpliv negotovosti) in izbranih reprezentativnih hidrogramov za določitev projektnih hidrogramov za različne vrednosti povratnih dob in z upoštevanjem negotovosti.

Ocenjuje se, da je za izvedbo verjetnostnih analiz smiselno upoštevati vsaj 15–20 let zveznih meritev pretokov zadnjega obdobja (zaradi sodobnejšega načina merjenja in podnebne spremenljivosti), kar nam omogoča ustrezno izbiro vzorca (AM ali POT). Podatki o pretokih morajo biti seveda ustrezne kakovosti, kar se nanaša na stabilen merski profil, ustrezno pretočno krivuljo ipd. Vsekakor pa se moramo zavedati, da v primeru tako kratkih nizov podatkov negotovost ocenjenih projektnih pretokov z večanjem povratne dobe strmo narašča. Določitev in upoštevanje intervalov zaupanja ter previdnost pri določanju povratnih dob, ki so daljše od dvakratnika dolžine niza merjenih podatkov, je zato v takih primerih nujna[51].

Na območju Slovenije meritve vodostajev in pretokov izvaja Agencija RS za okolje (ARSO). Na svoji spletni strani objavlja tudi podatke o visokovodnih konicah[52]. Podatki o visokovodnih konicah so na voljo za večino vodomernih postaj in jih ARSO označuje s *Qvk*, največji pretok v posameznem letu pa je označen z *vQvk*.

V našem primeru bomo za izvedbo verjetnostnih analiz uporabili podatke z vodomerne postaje Litija na reki Savi. Za izračun intervalov zaupanja bo uporabljen postopek, ki so ga

[51] https://www.ceh.ac.uk/sites/default/files/2021-11/Flood-Estimation-Handbook-3-Statistical-Procedures-For-Flood-Frequency-Estimation_Alice-Robson_Duncan-Reed.pdf.

[52] https://www.arso.gov.si/vode/podatki/arhiv/hidroloski_arhiv.html.

predlagali Meylan in sodelavci[53]. Podatki o največjih letnih konicah za obdobje 1895–2021 so na voljo na naslednji povezavi[54]. Poudariti je treba, da se je način izvajanja meritev vodostajev na tej vodomerni postaji v obdobju 1895–2021 nekajkrat spremenil. Tako so od 1895 do 1952 opazovanja potekala enkrat dnevno, od 1953 do 2014 so se meritve izvajale z uporabo limnigrafa, od 2015 naprej pa se meritve izvajajo s tlačno sondo in radarskim merilnikom. Zaradi tega so tudi lastnosti podatkov nekoliko drugačne, saj v večini primerov podatki na podlagi dnevnih opazovanj podcenjujejo konice pretokov. Za oceno parametrov porazdelitvenih funkcij bomo najprej uporabili paketa *lmom* in *lmomco*, ki vsebujeta funkcije za oceno parametrov na podlagi metode momentov L[55].

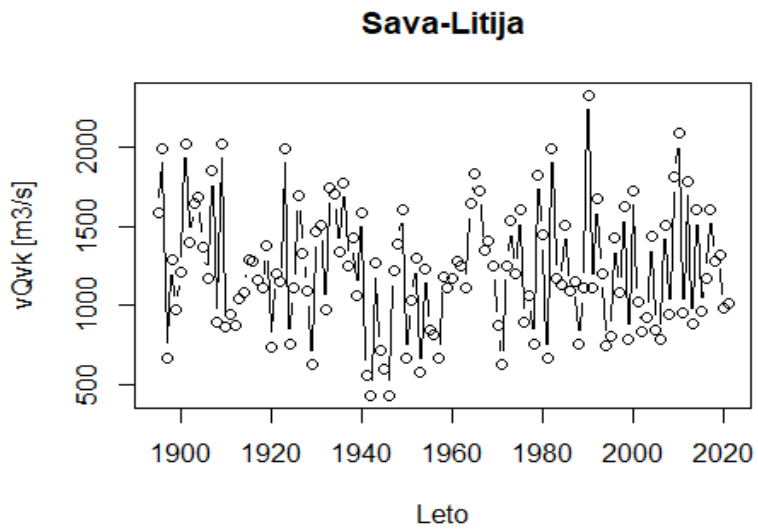
```
# uvozimo podatke o največjih letnih konicah v R
vQvk <- read.table("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/Litija-vQvk.txt",header=TRUE)
head(vQvk) # preverimo izgled podatkov

##   Leto   vQvk
## 1 1895 1586.6
## 2 1896 1993.2
## 3 1897  673.0
## 4 1898 1289.0
## 5 1899  973.8
## 6 1900 1213.0

str(vQvk) # preverimo strukturo podatkov

## 'data.frame':    127 obs. of  2 variables:
##  $ Leto: int  1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 ...
##  $ vQvk: num  1587 1993 673 1289 974 ...

summary(vQvk) # izračunamo osnovno statistiko naših podatkov

##       Leto           vQvk
##  Min.   :1895   Min.   : 434.0
##  1st Qu.:1926   1st Qu.: 945.5
##  Median :1958   Median :1199.0
##  Mean   :1958   Mean   :1228.7
##  3rd Qu.:1990   3rd Qu.:1507.0
##  Max.   :2021   Max.   :2326.0
```

[53] https://www.routledge.com/Predictive-Hydrology-A-Frequency-Analysis-Approach/Meylan-Favre-Musy/p/book/9781578087471.

[54] https://unilj-my.sharepoint.com/:f:/g/personal/nbezak_fgg_uni-lj_si/EgiFWdB01CtEldvCU3XX844BrrqoodVBoTWiV-76eufZaA?e=yIUr6o.

[55] https://www.cambridge.org/core/books/regional-frequency-analysis/8C59835F9361705DAAE1ADFDEA7ECD30.

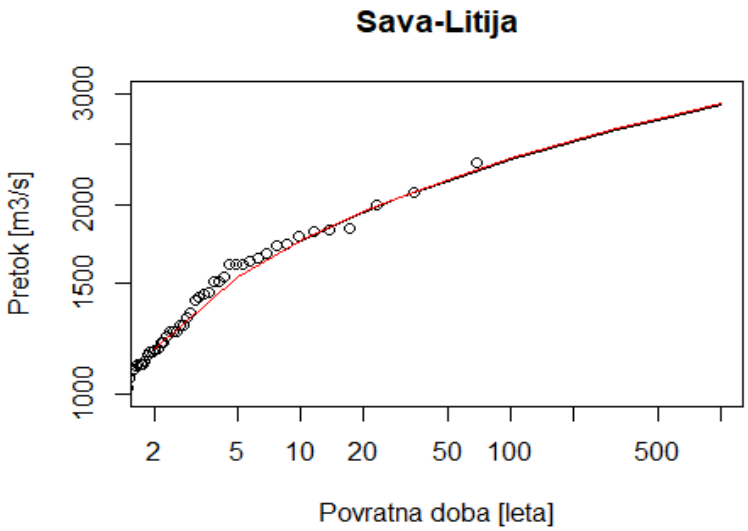
```
# izrišemo linijski graf
plot(vQvk,xlab="Leto",ylab="vQvk [m3/s]",main="Sava-Litija",type="b")
```



Slika 37: Največje letne visokovodne konice po metodi AM.

```
library(lmom, quietly=TRUE)
library(lmomco, quietly=TRUE) # aktiviramo oba paketa

##
## Attaching package: 'lmomco'

## The following objects are masked from 'package:lmom':
##
##     cdfexp, cdfgam, cdfgev, cdfglo, cdfgno, cdfgpa, cdfgum, cdfkap,
##     cdfln3, cdfnor, cdfpe3, cdfwak, cdfwei, quaexp, quagam, quagev,
##     quaglo, quagno, quagpa, quagum, quakap, qualn3, quanor, quape3,
##     quawak, quawei

# na podlagi vzorca za obdobje 1953–2021 izračunamo momente L
lmomenti <- lmoms(vQvk[60:127,2])
lmomenti$lambdas # momenti L

## [1] 1233.514412  214.272722   27.838481   19.071841    5.891533

# ocenimo parametre izbrane porazdelitvene funkcije
pe3par <- lmom2par(lmomenti,type="pe3")
pe3par # preverimo izračune

## $type
## [1] "pe3"
##
## $para
##           mu        sigma        gamma
## 1233.5144118  387.2970013    0.7919035
##
```

```
## $source
## [1] "parpe3"

# še porazdelitev GEV (generalizirana porazdelitev ekstremnih vrednosti)
gevpar <- lmom2par(lmomenti,type="gev")
# izračun projektnih pretokov za Fx z uporabo Pearsonove 3 porazdelitve
quape3(c(0.9,0.99),pe3par)

## [1] 1751.005 2351.094

# z uporabo porazdelitve GEV
quagev(c(0.9,0.99),gevpar)

## [1] 1749.472 2368.118

# naredimo izračune za različne vrednosti povratnih dob
Pdobe <- c(2,5,10,20,30,50,100,300,500,1000)
verj <- 1-1/Pdobe # vrednosti porazdelitve funkcije Fx
rezpe3 <- quape3(verj,pe3par) # Pearsonova porazdelitev tipa 3
rezgev <- quagev(verj,gevpar) # porazdelitev GEV
# preverimo rezultate za povratne dobe 100, 300, 500 in 1000 let
rezpe3[7:10]

## [1] 2351.094 2606.243 2720.772 2872.840

rezgev[7:10] # še z uporabo porazdelitve GEV

## [1] 2368.118 2628.055 2742.690 2892.341

# prikazali bomo tudi merjene podatke z uporabo Weibullove porazdelitve
# funkcija pp ima tudi parameter a, ki omogoča uporabo različnih enačb
weibull <- pp(vQvk[60:127,2],a=0)
Pdobe1 <- 1/(1-weibull) # izračunamo pripadajoče vrednosti povratnih dob
# izrišemo rezultate z uporabo Pearsonove porazdelitve tipa 3
plot(Pdobe,rezpe3,type="l",log="xy",xlab="Povratna doba [leta]",
ylab="Pretok [m3/s]",main="Sava-Litija",ylim=c(1000,3000))
# na graf dodamo tudi merjene podatke o pretokih
points(Pdobe1,sort(vQvk[60:127,2]))
# na graf dodamo tudi rezultate glede na porazdelitev GEV
lines(Pdobe,rezgev,col="red")
```



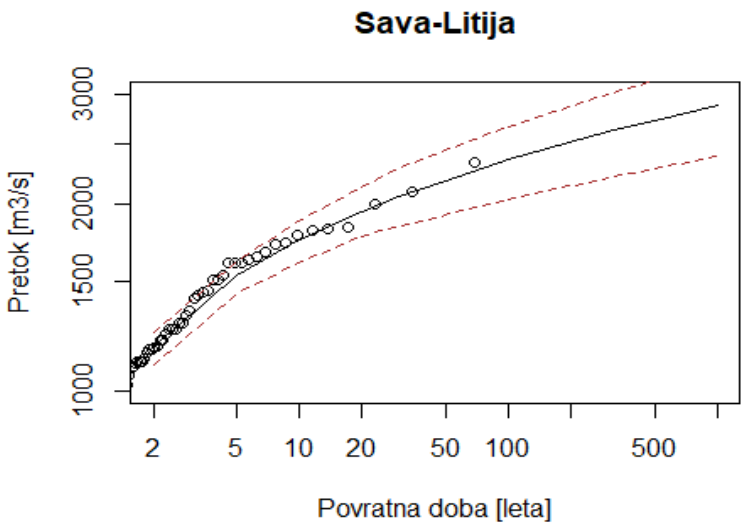
Slika 38: Rezultati verjetnostne analize visokovodnih konic z uporabo porazdelitve GEV.

Pri izvedbi verjetnostnih analiz je smiselno izvrednotiti tudi pripadajoče intervale zaupanja. Intervale zaupanja bomo generirali glede na postopek, ki so ga predlagali Meylan in sodelavci[56], pri čemer bomo v vsaki ponovitvi na podlagi ocenjenih parametrov izbrane porazdelitvene funkcije (v tem primeru Pearson tipa 3) generirali nov vzorec največjih letnih pretokov. Na podlagi vzorca bomo še enkrat ocenili parametre in nato na podlagi parametrov izračunali vrednosti projektnih pretokov za izbrane povratne dobe. Izbrali bomo 10 % empirične intervale zaupanja ter določili spodnje in zgornje meje za posamezne vrednosti povratnih dob. Pri generiranju podatkov je smiselno uporabiti veliko število ponovitev, npr. 10.000.

```
Nrep <- 100 # število ponovitev naključnega generiranja parametrov
# prazna matrika za shranjevanje rezultatov
rezultati <- matrix(numeric(0), Nrep, length(verj))
for(i in 1:Nrep){
sample_dummy <- rlmomco(length(vQvk[60:127,2]),
lmom2par(lmoms(vQvk[60:127,2]),type="pe3"))
parPE3_dummy <- lmom2par(lmoms(sample_dummy),type="pe3")
rezultati[i,] <- quape3(verj, parPE3_dummy)
}
alpha <- 0.1
# vrednosti intervalov zaupanja za spodnjo mejo za 2-letno povratno dobo
rez2l <- sort(rezultati[,1])[alpha*100/2]
# podobno še za zgornjo mejo
rez2u <- sort(rezultati[,1])[Nrep-alpha*100/2]
# in še za ostale vrednosti povratnih dob
rez5l <- sort(rezultati[,2])[alpha*100/2]
rez5u <- sort(rezultati[,2])[Nrep-alpha*100/2]
```

[56] https://www.routledge.com/Predictive-Hydrology-A-Frequency-Analysis-Approach/Meylan-Favre-Musy/p/book/9781578087471.

```r
rez10l <- sort(rezultati[,3])[alpha*100/2]
rez10u <- sort(rezultati[,3])[Nrep-alpha*100/2]
rez20l <- sort(rezultati[,4])[alpha*100/2]
rez20u <- sort(rezultati[,4])[Nrep-alpha*100/2]
rez30l <- sort(rezultati[,5])[alpha*100/2]
rez30u <- sort(rezultati[,5])[Nrep-alpha*100/2]
rez50l <- sort(rezultati[,6])[alpha*100/2]
rez50u <- sort(rezultati[,6])[Nrep-alpha*100/2]
rez100l <- sort(rezultati[,7])[alpha*100/2]
rez100u <- sort(rezultati[,7])[Nrep-alpha*100/2]
rez300l <- sort(rezultati[,8])[alpha*100/2]
rez300u <- sort(rezultati[,8])[Nrep-alpha*100/2]
rez500l <- sort(rezultati[,9])[alpha*100/2]
rez500u <- sort(rezultati[,9])[Nrep-alpha*100/2]
rez1000l <- sort(rezultati[,10])[alpha*100/2]
rez1000u <- sort(rezultati[,10])[Nrep-alpha*100/2]
# podatke združimo v podatkovni okvir
genrez <- data.frame(Povratne=Pdobe,spodnja=c(rez2l,rez5l,rez10l,rez20l,
  rez30l,rez50l,rez100l,rez300l,rez500l,rez1000l),
zgornja=c(rez2u,rez5u,rez10u,rez20u,rez30u,rez50u,rez100u,rez300u,
  rez500u,rez1000u))
# preverimo projektni pretok s povratno dobo 100 let
rezpe3[7]

## [1] 2351.094

rez100l # pogledamo še vrednost intervala zaupanja

## [1] 2032.378

rez100u

## [1] 2660.641

# izrišemo graf
plot(Pdobe,rezpe3,type="l",log="xy",xlab="Povratna doba [leta]",
ylab="Pretok [m3/s]",main="Sava-Litija",ylim=c(1000,3000))
# na graf dodamo tudi merjene podatke o pretokih
points(Pdobe1,sort(vQvk[60:127,2]))
# na grafu prikažemo tudi empirične intervale zaupanja
lines(genrez[,1],genrez[,2],col="brown",lty=2)
lines(genrez[,1],genrez[,3],col="brown",lty=2)
```



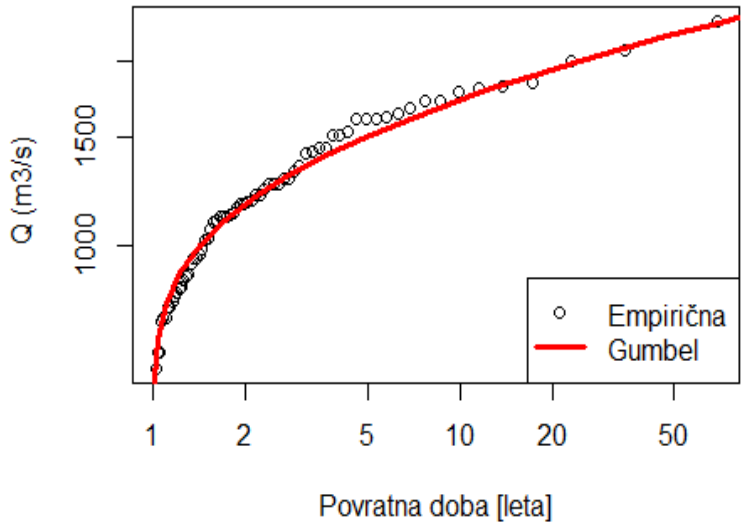
Slika 39: Rezultati verjetnostne analize z uporabo Pearsonove III porazdelitve skupaj z intervali zaupanja.

Vidimo lahko, da je v ocenjenih projektnih pretokih kljub relativno dolgemu nizu podatkov še vedno relativno velika negotovost samo zaradi postopka izvedbe verjetnostne analize (npr. ni upoštevana negotovost pri meritvah pretokov). Poleg metode momentov L lahko za ocenjevanje parametrov uporabimo tudi druge metode, na primer metodo momentov ali metodo največjega verjetja[57]. Spodaj je najprej prikazan postopek z uporabo metode momentov (in Gumbelove porazdelitvene funkcije) ter nato še z uporabo metode največjega verjetja (uporabljen bo paket *extRemes*). Ta paket omogoča tudi izračun nekaterih kriterijev ujemanja, kot sta Akaike information criterion (AIC) in Bayesian information criterion (BIC), ki ju lahko uporabimo za izbiro najbolj ustrezne porazdelitvene funkcije. V sklopu paketa je možno izbrati tudi druge porazdelitve, kot so GEV, generalizirana Pareto porazdelitev (type="GP") ali eksponentna. Paket omogoča tudi izris številnih grafov, iz katerih lahko razberemo ustreznost posamezne porazdelitvene funkcije, če testiramo več različnih porazdelitvenih funkcij.

```
# najprej je prikazan primer uporabe metode momentov
povp <- mean(vQvk[60:127,2]) # izračunamo povprečje našega vzorca
standev <- sd(vQvk[60:127,2]) # še standardno deviacijo
sigma <- (sqrt(6)/pi)*standev # scale parameter Gumbelove porazdelitve
mu <- povp - 0.5772*sigma # location parameter Gumbelove porazdelitve
# funkcija, s katero lahko za vrednosti pretokov izračunamo porazdelitveno f.
pGumbel <- function (pret, mu, sigma) {
  y <- (pret - mu)/sigma
  p <- exp(-exp(-y))
  return(p)
}
# izračun Fx za izbrane vrednosti pretokov, od 500 do 2500 m3/s (korak 100)
Fxgum <- pGumbel(seq(500,2500,100),mu,sigma)
```

[57] https://doi.org/10.1080/02626667.2013.831174.

```
plot(Pdobe1, sort(vQvk[60:127,2]), log="xy", xlab="Povratna doba [leta]",
     ylab=" Q (m3/s)")
lines(1/(1-Fxgum),seq(500,2500,100),col="red",lwd=3)
legend("bottomright",legend=c("Empirična","Gumbel"), lty=c(NA,1),lwd=c(1,3),
col=c("black","red"),pch=c(21,NA))
```



Slika 40: Verjetnostna analiza z Gumbelovo porazdelitvijo (metoda momentov).

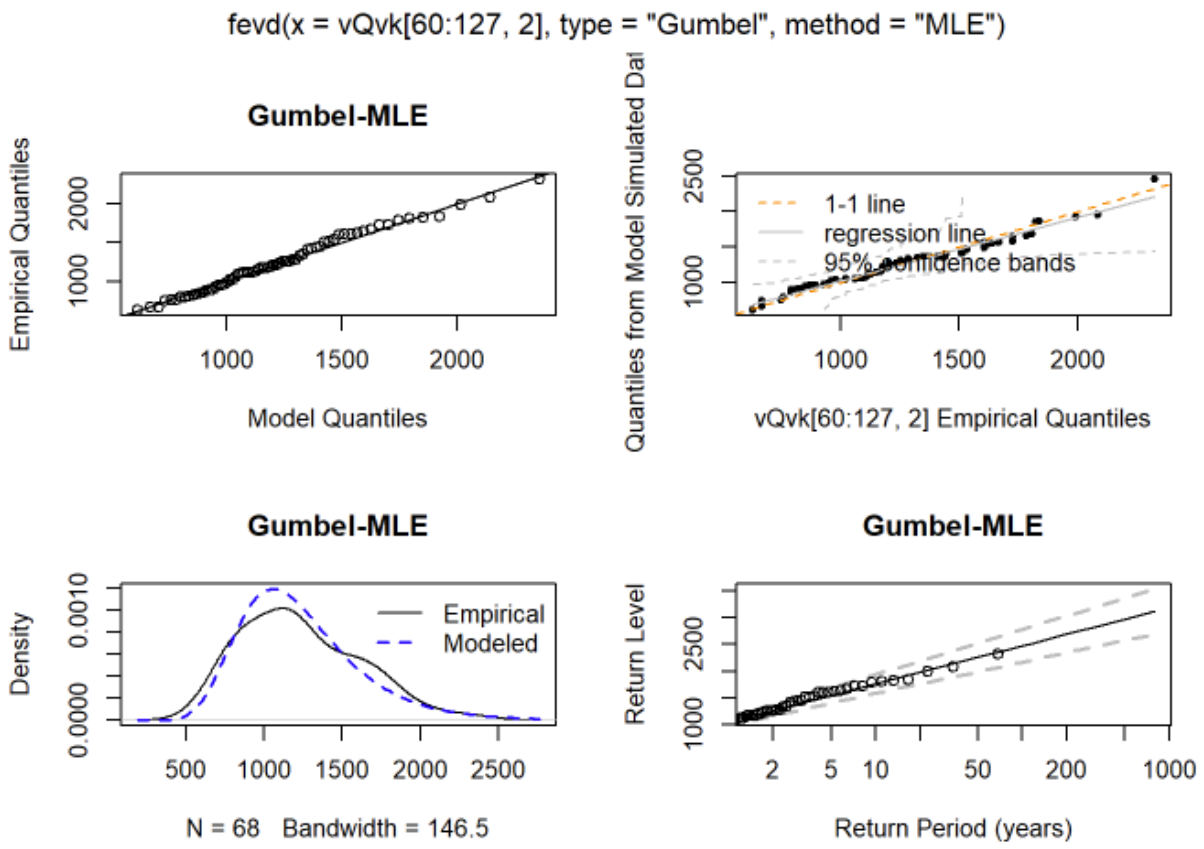
```
# izračunamo še projektni pretok s 100-letno povratno dobo
mu-sigma*log(-log(1-1/100))

## [1] 2420.529

# primerjamo oceno z izračunom na podlagi metode momentov L
quagum(1-1/100,lmom2par(lmomenti,type="gum"))

## [1] 2477.125

library(extRemes, quietly=TRUE)

##
## Attaching package: 'extRemes'

## The following objects are masked from 'package:stats':
##
##     qqnorm, qqplot

# ocenimo parametre Gumbelove porazdelitve po metodi največjega verjetja
Gumbelmle <- fevd(vQvk[60:127,2],type="Gumbel", method="MLE")
summary(Gumbelmle) # AIC in BIC

##
## fevd(x = vQvk[60:127, 2], type = "Gumbel", method = "MLE")
##
## [1] "Estimation Method used: MLE"
##
```

```
##
##  Negative Log-Likelihood Value:  496.4416
##
##
##  Estimated parameters:
##  location     scale
## 1057.7568  308.0549
##
##  Standard Error Estimates:
## location    scale
## 39.38974 29.75105
##
##  Estimated parameter covariance matrix.
##           location    scale
## location 1551.5516 371.2417
## scale     371.2417 885.1248
##
##  AIC = 996.8832
##
##  BIC = 1001.322
```

```
Gumbelmle$results$par # parametri Gumbelove porazdelitvene funkcije
```

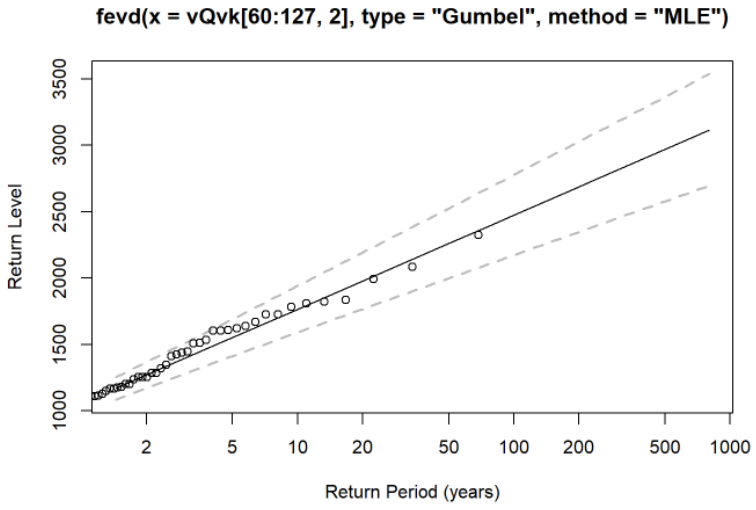
```
##  location     scale
## 1057.7568  308.0549
```

```
# diagnostični grafi
plot(Gumbelmle, main="Gumbel-MLE")
```



Slika 41: Diagnostični grafi v paketu extRemes.

```
# posamezni grafi z uporabo dodatnega argumenta type,
# izbiramo lahko med probprob, qq, density itd.
plot(Gumbelmle, type="rl")
```



Slika 42: Eden izmed diagnostičnih grafov v paketu extRemes.

```
# izračun projektnega pretoka
Gumbelmle$results$par[1]-Gumbelmle$results$par[2]*log(-log(1-1/100))

## location
## 2474.855
```

Za izbiro najustreznejše porazdelitvene funkcije je treba izvesti različne statistične teste ali izračunati kriterije ujemanja. V različnih paketih je na voljo veliko število testov, s katerimi lahko preverimo ustreznost posamezne porazdelitvene funkcije. Zgoraj smo že videli kriterija AIC in BIC. Eden izmed takšnih je tudi Kolmogorov-Smirnov test, kjer preverimo, ali se ocenjeni pretoki po izbrani porazdelitvi dovolj dobro ujemajo z merjenimi podatki. Če je p-vrednost večja od 0,05, to pomeni, da ničelne hipoteze (porazdelitev je enaka) ne moremo zavrniti z izbrano stopnjo značilnosti. Omeniti velja tudi paket *hydroGOF*, ki vključuje veliko funkcij za izračun kriterijev ujemanja, kot so RMSE, MAE.

```
# likelihood test (test razmerja verjetnosti) za dve izbrani porazdelitvi
lr.test(fevd(vQvk[60:127,2],type="Gumbel", method="MLE"),
    fevd(vQvk[60:127,2],type="GEV", method="MLE"))

##
##  Likelihood-ratio Test
##
## data:  vQvk[60:127, 2]vQvk[60:127, 2]
## Likelihood-ratio = 0.14202, chi-square critical value = 3.8415, alpha =
## 0.0500, Degrees of Freedom = 1.0000, p-value = 0.7063
## alternative hypothesis: greater

# Kolmogorov-Smirnov test
ks.test(sort(vQvk[60:127,2]),quape3(weibull,pe3par))

## Warning in ks.test(sort(vQvk[60:127, 2]), quape3(weibull, pe3par)): cannot
## compute exact p-value with ties

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sort(vQvk[60:127, 2]) and quape3(weibull, pe3par)
## D = 0.058824, p-value = 0.9998
## alternative hypothesis: two-sided

ks.test(sort(vQvk[60:127,2]),quagev(weibull,gevpar))

## Warning in ks.test(sort(vQvk[60:127, 2]), quagev(weibull, gevpar)): cannot
## compute exact p-value with ties

##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  sort(vQvk[60:127, 2]) and quagev(weibull, gevpar)
## D = 0.058824, p-value = 0.9998
## alternative hypothesis: two-sided
```

```
library(hydroGOF, quietly=TRUE)

## Warning: package 'hydroGOF' was built under R version 4.1.3

# RMSE za različne porazdelitvene funkcije
rmse(quagev(weibull,gevpar),sort(vQvk[60:127,2]))

## [1] 34.38188

rmse(quape3(weibull,pe3par),sort(vQvk[60:127,2]))

## [1] 32.59287

# ME za različne porazdelitvene funkcije
me(quagev(weibull,gevpar),sort(vQvk[60:127,2]))

## [1] -5.015486

me(quape3(weibull,pe3par),sort(vQvk[60:127,2]))

## [1] -5.177761

# MAE za različne porazdelitvene funkcije
mae(quagev(weibull,gevpar),sort(vQvk[60:127,2]))

## [1] 27.97724

mae(quape3(weibull,pe3par),sort(vQvk[60:127,2]))

## [1] 26.5385
```

Naloga 33: S spletne strani Agencije RS za okolje prenesite podatke o največjih letnih konicah z vodomerne postaje Sava Šentjakob za obdobje 1990–2021 (lahko uporabite tudi podatke iz naloge 32). Z uporabo paketov lmom in lmomco analizirajte največje konice pretokov. Izvedite verjetnostne analize konic pretokov (GEV in Gumbelova porazdelitev).

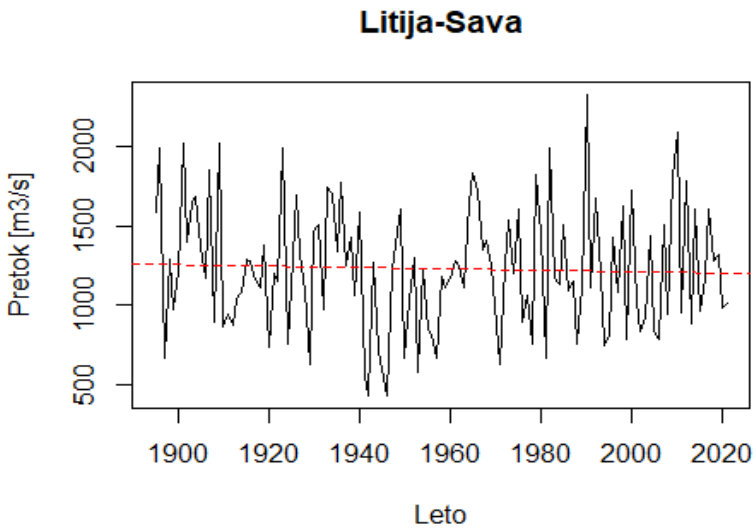
4.3 Analize trendov

Podnebne spremembe vplivajo na hidrološke procese, saj povzročajo spremembe v vodnem krogu[58]. Podnebne spremembe (ali spremenljivost) se lahko kažejo v spremenjenih vzorcih padavin, temperaturnih spremembah, spremembah pogostosti in intenzivnosti ekstremnih hidroloških dogodkov ter ostalih spremembah hidroloških procesov. Posledično se na področju vodarstva in v hidrologiji pogosto srečamo z analizami trendov in zaznavanjem

[58] https://www.ipcc.ch/report/ar6/wg2/chapter/chapter-4/.

sprememb v podatkih[59]. Analiza trendov v hidrologiji vključuje analize podatkov za ugotavljanje sprememb v podatkih, kot so npr. padavine, pretok, temperatura zraka ali gladina podzemne vode. Takšne analize so ključne za razumevanje hidrološkega kroga in zaznavanje sprememb, saj so to podatki, ki jih na primer potrebujemo za upravljanje vodnih virov ali zagotavljanje poplavne varnosti. Na voljo je veliko različnih metod, s katerimi lahko ugotavljamo, ali je v naših podatkih prisoten trend, ali je ta pozitiven ali negativen in ali je ta statistično značilen. Številne izmed pogosto uporabljenih metod so vključene v paket *trend*[60]. Pogosto se za analize uporablja Mann-Kendallov statistični test, ki je neparametrični test, pri čemer ničelna hipoteza pravi, da v podatkih ni prisotnega trenda, alternativna pa, da je v podatkih prisoten trend[61]. Alternativa Mann-Kendallovemu testu je na primer test Senovega naklona [62]. V nadaljevanju bodo prikazani nekateri primeri uporabe teh statističnih testov.

```
# izrišemo linijski graf
plot(vQvk[,1],vQvk[,2],type="l",xlab="Leto", ylab="Pretok [m3/s]",
    main="Litija-Sava")
# dodamo še linearno trendno črto
abline(lm(vQvk[,2] ~ vQvk[,1]),lty=2,col="red")
```



Slika 43: Linijski graf pretokov s prikazano trendno črto.

```
# preverimo smerni koeficient linearne funkcije, vidimo, da je ta negativen
# vQvk naj bi se glede na linearno f. znižal za približno 40 m³/s na 100 let
coef(lm(vQvk[,2] ~ vQvk[,1]))
```

---

[59] https://doi.org/10.1007/978-94-007-1861-6.

[60] https://CRAN.R-project.org/package=trend.

[61] https://shop.elsevier.com/books/time-series-modelling-of-water-resources-and-environmental-systems/hipel/978-0-444-89270-6.

[62] https://doi.org/10.1080/01621459.1968.10480934.

```
##  (Intercept)    vQvk[, 1]
## 2031.2582216   -0.4099017

# za analize trendov bomo uporabili paket trend
library(trend, quietly=TRUE)
# izvedemo analizo trendov z uporabo Mann-Kendallovega statističnega testa
# vidimo, da je v podatkih prisoten negativen trend (vrednost z)
# vendar ta ni statistično značilen (p-value je višja od meje, npr. 0,05)
mk.test(vQvk[,2])

##
##  Mann-Kendall trend test
##
## data:  vQvk[, 2]
## z = -0.53352, n = 127, p-value = 0.5937
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##             S          varS           tau
## -2.570000e+02  2.302423e+05 -3.213706e-02

# če nas zanima trend le za en del podatkov
mk.test(vQvk[1:65,2])

##
##  Mann-Kendall trend test
##
## data:  vQvk[1:65, 2]
## z = -2.2873, n = 65, p-value = 0.02218
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##             S          varS           tau
##  -405.0000000 31197.0000000    -0.1948521

# uporabimo tudi test Senovega naklona
sens.slope(vQvk[,2])

##
##  Sen's slope
##
## data:  vQvk[, 2]
## z = -0.53352, n = 127, p-value = 0.5937
## alternative hypothesis: true z is not equal to 0
## 95 percent confidence interval:
##  -2.5250  1.5625
## sample estimates:
## Sen's slope
##  -0.6064634

# samo za del podatkov
sens.slope(vQvk[1:65,2])
```

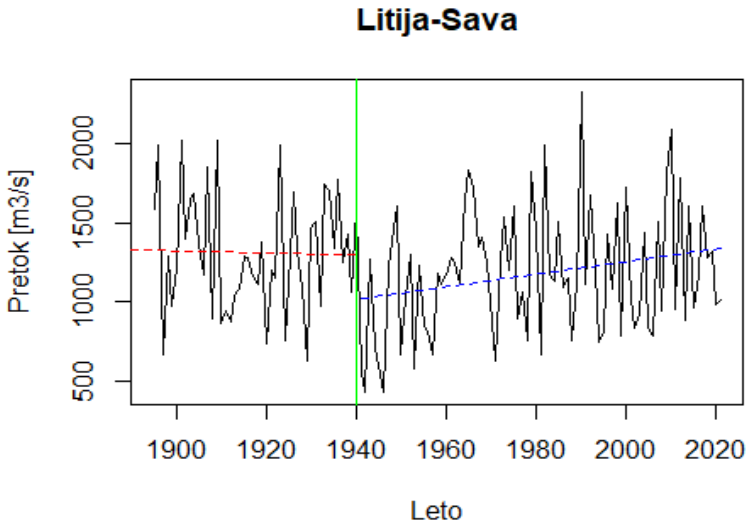
```
##
##  Sen's slope
##
## data:  vQvk[1:65, 2]
## z = -2.2873, n = 65, p-value = 0.02218
## alternative hypothesis: true z is not equal to 0
## 95 percent confidence interval:
##  -12.414286  -1.208333
## sample estimates:
## Sen's slope
##   -6.717045
```

Paket *trend* vključuje tudi številne različice Mann-Kendallovega testa, kot je sezonski test ali multivariatni test (tudi za test Senovega naklona). Paket dodatno vključuje tudi številne teste za analize nenadnih sprememb v podatkih, kot je na primer Pettittov test, ki je bil med drugim uporabljen tudi za zaznavanje sprememb v hidroloških podatkih kraških izvirov[63]. Najprej bo prikazana analiza točke nenadne spremembe v podatkih z uporabo Pettittovega testa na dejanskih podatkih in na generiranih podatkih. Nato bodo prikazane še nekatere druge funkcije iz paketa *trend*.

```
pettitt.test(vQvk[,2])

##
##  Pettitt's test for single change-point detection
##
## data:  vQvk[, 2]
## U* = 718, p-value = 0.447
## alternative hypothesis: two.sided
## sample estimates:
## probable change point at time K
##                              46

# podatek št. 46, rezultat pa je statično neznačilen (st. značilnosti 0,05)
# če generiramo naključne podatke na podlagi normalne porazdelitve
# (10 vrednosti s povprečjem 0 in 10 vrednosti s povprečjem 20)
pettitt.test(c(rnorm(n=10,mean=0),rnorm(n=10,mean=20)))

##
##  Pettitt's test for single change-point detection
##
## data:  c(rnorm(n = 10, mean = 0), rnorm(n = 10, mean = 20))
## U* = 100, p-value = 0.001581
## alternative hypothesis: two.sided
## sample estimates:
## probable change point at time K
##                              10
```

[63] https://doi.org/10.15292/acta.hydro.2020.01.

```
# test zazna statistično značilno točko preloma med dvema nizoma podatkov
# podoben rezultat kot zgoraj
# podobne analize lahko naredimo tudi z uporabo br.test in bu.test funkcij
lanzante.test(vQvk[,2])

##
##  Lanzante's procedure for single change-point detection
##  with Wilcoxon-Mann-Whitney Test
##
## data:  vQvk[, 2]
## W = 2222, p-value = 0.07213
## alternative hypothesis: two.sided
## sample estimates:
## probable change point at time K
##                              46

plot(vQvk[,1],vQvk[,2],type="l",xlab="Leto", ylab="Pretok [m3/s]",
    main="Litija-Sava")
# če želimo izrisati trendno črto samo na enem delu podatkov
clip(1800,1940,200,3000)
abline(lm(vQvk[1:46,2] ~ vQvk[1:46,1]),lty=2,col="red") # prvi del podatkov
clip(1940,2022,200,3000) # še za drugi del podatkov
abline(lm(vQvk[47:127,2] ~ vQvk[47:127,1]),lty=2,col="blue") # drugi del
do.call("clip", as.list(par("usr"))) # poenostavimo območje risanja
# dodamo na graf še točko preloma
abline(v=vQvk[pettitt.test(vQvk[,2])$estimate,1],col="green",lty=1)
```



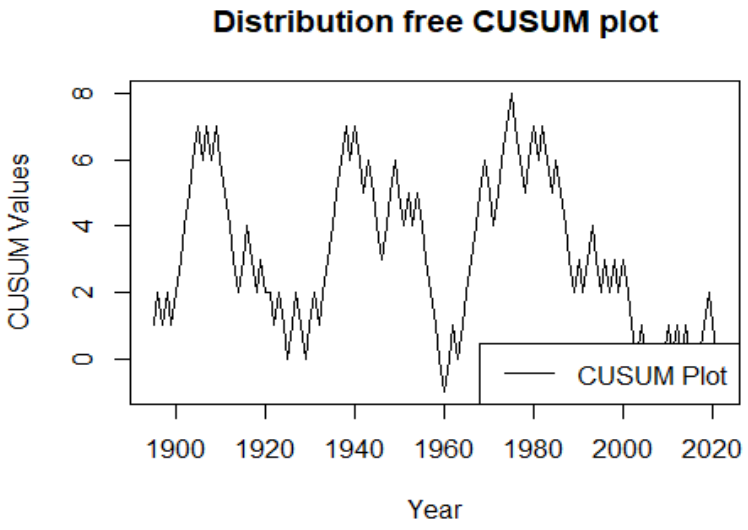
Slika 44: Linijski graf pretokov in točka preloma v podatkih.

Obstajajo pa tudi nekoliko naprednejše metode zaznavanja trendov, kot so tiste, ki so vključene v paket *trendchange*[64]. Dodatno je mogoče izvesti tudi regionalni Mann-Kendallov

[64] https://doi.org/10.1061/(ASCE)HE.1943-5584.0000556.

test, kot je bilo to prikazano v primeru podatkov s padavinskih postaj v Sloveniji[65]. Več informacij o regionalnem Mann-Kendallovem testu je na voljo v znanstvenem članku[66] in priročniku paketa *rkt*[67].

```
library(trendchange, quietly=TRUE) # aktiviramo paket trendchange

## Warning: package 'trendchange' was built under R version 4.1.3

##
## Attaching package: 'trendchange'

## The following object is masked _by_ '.GlobalEnv':
##
##     x

# grafični prikaz zaznavanja prelomnih točk v podatkih
dfcusum(vQvk[,2],startyear = 1895)
```



Slika 45: Zaznavanje trenda z uporabo funkcije CUMSUM iz paketa trendchange.

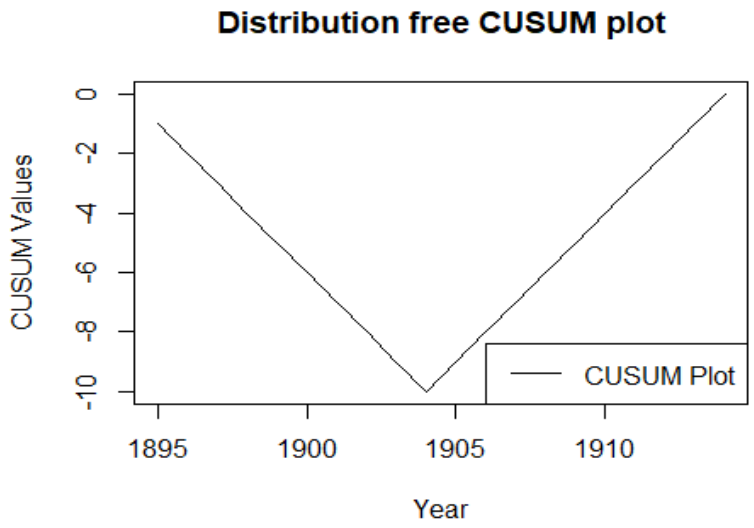
```
## $`CUMSUM Values`
##   [1]  1  2  1  2  1  2  3  4  5  6  7  6  7  6  7  6  5  4  3  2  3  4  3
2  3
##  [26]  2  2  1  2  1  0  1  2  1  0  1  2  1  2  3  4  5  6  7  6  7  6  5
6  5
##  [51]  4  3  4  5  6  5  4  5  4  5  4  3  2  1  0 -1  0  1  0  1  2  3  4
5  6
##  [76]  5  4  5  6  7  8  7  6  5  6  7  6  7  6  5  6  5  4  3  2  3  2  3
```

[65] https://doi.org/10.3390/w11102167.

[66] https://doi.org/10.1021/es051650b.

[67] https://cran.r-project.org/package=rkt.

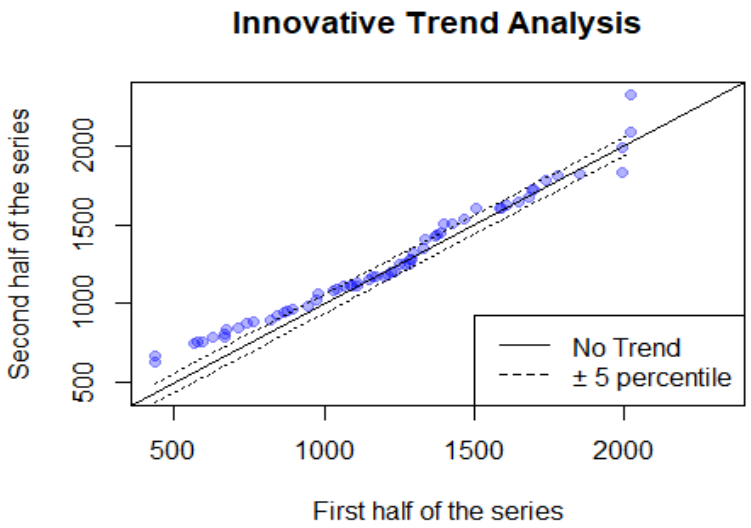
```
4  3
## [101]  2  3  2  3  2  3  2  1  0  1  0 -1  0 -1  0  1  0  1  0  1  0 -1  0
1  2
## [126]  1  0
##
## $`Maximum CUMSUM value`
## [1] 8
##
## $`Critical value at 90% CI`
## [1] 13.7487
##
## $`Critical value at 95% CI`
## [1] 15.32642
##
## $`Critical value at 99% CI`
## [1] 18.36917

# prelomne točke so prikazane kot minimumi na grafu
# to je lepo razvidno v primeru, kjer uporabimo dva podniza podatkov
dfcusum(c(rnorm(n=10,mean=0),rnorm(n=10,mean=20)),startyear = 1895)
```



Slika 46: Zaznavanje trenda z uporabo funkcije CUMSUM na podlagi generiranih podatkov.

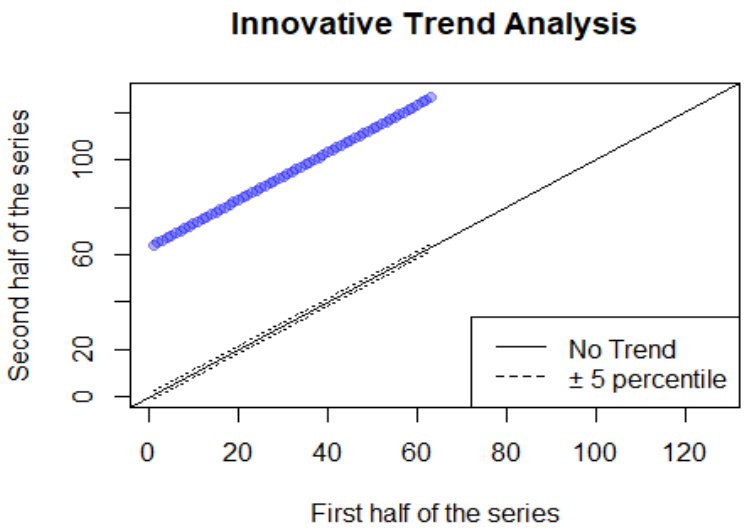
```
## $`CUMSUM Values`
##  [1]  -1  -2  -3  -4  -5  -6  -7  -8  -9 -10  -9  -8  -7  -6  -5  -4  -3
-2  -1
## [20]   0
##
## $`Maximum CUMSUM value`
## [1] 10
##
## $`Critical value at 90% CI`
## [1] 5.456006
##
## $`Critical value at 95% CI`
```

```
## [1] 6.082105
##
## $`Critical value at 99% CI`
## [1] 7.289582

# ter še grafični prikaz zaznavanja trendov
innovtrend(vQvk[,2],ci=95)
```



Slika 47: Inovativen način zaznavanja trendov na podatkih o pretokih.

```
##                         Trend Slope                       Trend Indicator
##                          0.89026622                            0.47028305
##           Slope Standard deviation                           Correlation
##                          0.09467647                            0.98515248
## Lower Cofidence Limit at 90percent Upper Cofidence Limit at 90percent
##                         -0.15574280                            0.15574280
## Lower Cofidence Limit at 95percent Upper Cofidence Limit at 95percent
##                         -0.18556589                            0.18556589
## Lower Cofidence Limit at 99percent Upper Cofidence Limit at 99percent
##                         -0.24388660                            0.24388660

# če bi imeli v podatkih izrazit trend, bi točke ležale
# ali nad intervali zaupanja (pozitiven) ali pod intervali (negativen)
# generiramo zaporedje števil od 1:127, kar predstavlja pozitiven trend
# na enak način bi preverili rezultat v primeru padajočega niza (127:1)
innovtrend(1:127,ci=95)
```



Slika 48: Inovativen način zaznavanja trendov na podlagi generiranih podatkov.

```
##                          Trend Slope                       Trend Indicator
##                             0.992126                             19.687500
##           Slope Standard deviation                           Correlation
##                             0.000000                              1.000000
## Lower Cofidence Limit at 90percent Upper Cofidence Limit at 90percent
##                             0.000000                              0.000000
## Lower Cofidence Limit at 95percent Upper Cofidence Limit at 95percent
##                             0.000000                              0.000000
## Lower Cofidence Limit at 99percent Upper Cofidence Limit at 99percent
##                             0.000000                              0.000000
```

Prikažimo še primer uporabe regionalnega Mann-Kendallovega testa. Prikazana je izvedba regionalnega Mann-Kendallovega testa v primeru, da bi poleg postaje Litija imeli na voljo še eno dolvodno postajo, kjer bi bili podatki o *vQvk* za 200 m^3/s večji od tistih na postaji Litija (definiramo z argumentom y). V tem primeru imamo potem dva niza podatkov o letih (argument *date*), argument *block* pa definira, kateri podatki pripadajo kateri postaji. Vidimo lahko, da v tem primeru regionalni Mann-Kendallov test ni statistično značilen (z izbrano stopnjo zaupanja 0,05), torej v podatkih ni prisotnega trenda. Dodatno bo prikazan še primer generiranih podatkov s prisotnim trendom (1:127 in 1001:1127), kjer pa so rezultati drugačni. V funkcijo *rkt* lahko seveda vključimo večje število postaj.

```
library(rkt, quietly=TRUE)
rkt(date=rep(1895:2021,2),y=c(vQvk[,2],vQvk[,2]+200),
block=c(rep(1,length(vQvk[,2])),rep(2,length(vQvk[,2]))))

##
## Standard model
## Tau = -0.03212098
## Score =  -514
## var(Score) =  460484.7
## 2-sided p-value =  0.4496617
## Theil-Sen's (MK) or seasonal/regional Kendall (SKT/RKT) slope=  -0.6064634
```

```
rkt(date=rep(1895:2021,2),y=c(1:127,1001:1127),
block=c(rep(1,length(vQvk[,2])),rep(2,length(vQvk[,2]))))

##
## Standard model
## Tau = 1
## Score =  16002
## var(Score) =  460502
## 2-sided p-value =  0
## Theil-Sen's (MK) or seasonal/regional Kendall (SKT/RKT) slope=  1
```

Naloga 34: S spletne strani Agencije RS za okolje prenesite podatke o največjih letnih konicah z vodomernih postaj na reki Savi: Radovljica I, Šentjakob, Litija (in Litija I), Hrastnik, Čatež I za obdobje 2000–2021. Izvedite analize trendov za posamezne vodomerne postaje (izberite poljuben test) in tudi regionalni Mann-Kendallov test.

4.4 Nestacionarne verjetnostne analize

Nestacionarne verjetnostne analize lahko izvedemo z uporabo paketa *extRemes*, ki smo ga že uporabili v primeru (običajnih) verjetnostnih analiz konic pretokov. V tem primeru v parametre izbranih teoretičnih porazdelitvenih funkcij vgradimo časovno ali kakšno drugo odvisnost izbranih spremenljivk (npr. visokovodnih konic, padavin)[68]. Definiramo lahko različne oblike nestacionarnih modelov, v odvisnosti od časa ali od drugih odvisnih spremenljivk, kot so npr. padavine[69]. V prejšnjih primerih smo ugotovili, da ima del naših podatkov statistično značilen negativen trend, uporabili bomo samo ta del podatkov. Na podlagi podatkov bomo ocenili parametre nestacionarnega modela z upoštevanjem metode največjega verjetja in uporabo generalizirane porazdelitve ekstremnih vrednosti (GEV), kjer je lokacijski parameter porazdelitve GEV funkcija časa. Dodatno bomo preverili statistiko testiranega modela, ki bi jo lahko primerjali tudi z recimo stacionarnim modelom (npr. na podlagi kriterijev BIC in AIC), funkcija *plot(model1)* pa omogoča tudi izris grafičnega ujemanja izbranega modela s podatki o največjih letnih konicah.

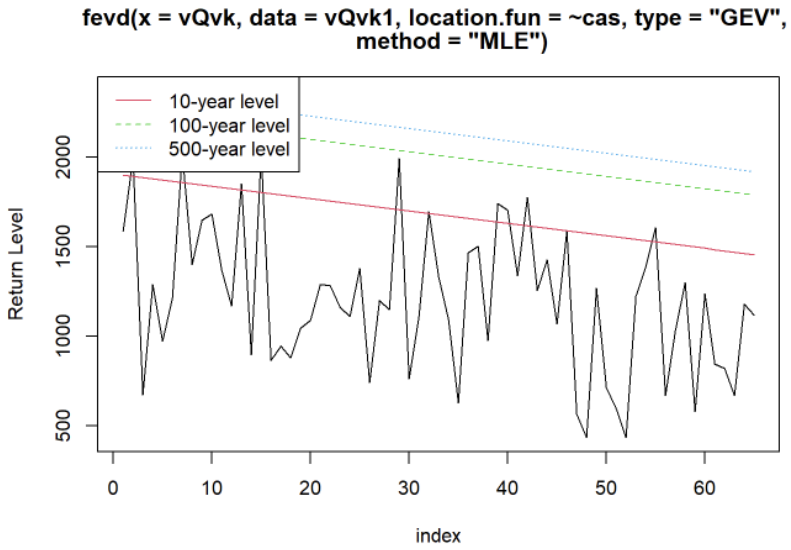
```
library(extRemes, quietly=TRUE)
vQvk1 <- vQvk[1:65,] # uporabimo samo del podatkov
cas <- 1:length(vQvk1[,2]) # definiramo vektor časa
# definiramo model
model1 <- fevd(x=vQvk,data=vQvk1,method="MLE",type="GEV",location.fun=~cas)
summary(model1) # osnovne značilnosti

##
## fevd(x = vQvk, data = vQvk1, location.fun = ~cas, type = "GEV",
```

---

[68] https://doi.org/10.1515/johh-2016-0032.

[69] https://doi.org/10.1016/j.jhydrol.2020.125374.

```
##     method = "MLE")
##
## [1] "Estimation Method used: MLE"
##
##
##  Negative Log-Likelihood Value:  477.1598
##
##
##  Estimated parameters:
##          mu0          mu1        scale        shape
## 1292.9295226   -6.9322010  370.3522898   -0.2875139
##
##  Standard Error Estimates:
##         mu0         mu1       scale       shape
## 102.1463941   2.5010305  39.8920436   0.1172437
##
##  Estimated parameter covariance matrix.
##                mu0           mu1       scale       shape
## mu0   10433.885836 -218.18402981  623.374966 -5.05417391
## mu1    -218.184030    6.25515372  -10.331939  0.06341077
## scale   623.374966  -10.33193937 1591.375143 -3.06325528
## shape    -5.054174    0.06341077   -3.063255  0.01374609
##
##  AIC = 962.3196
##
##  BIC = 971.0172

# izrišemo rezultate za različne vrednosti povratnih dob
plot(model1, type="rl",rperiods=c(10,100,500))
```



Slika 49: Rezultati nestacionarne verjetnostne analize.

```
# vrednosti projektnih pretokov se zmanjšujejo v odvisnosti od časa
return.level(model1,return.period=100) # izbrana vrednost povratne dobe
```

```
## fevd(x = vQvk, data = vQvk1, location.fun = ~cas, type = "GEV",
##     method = "MLE")
## get(paste("return.level.fevd.", newcl, sep = ""))(x = x, return.period = r
eturn.period)
##
##  GEV model fitted to  vQvk vQvk1
## Data are assumed to be  non-stationary
## [1] "Covariate data = vQvk1"
## [1] "Return Levels for period units in years"
##       100-year level
##  [1,]       2230.911
##  [2,]       2223.978
##  [3,]       2217.046
##  [4,]       2210.114
##  [5,]       2203.182
##  [6,]       2196.250
##  [7,]       2189.317
##  [8,]       2182.385
##  [9,]       2175.453
## [10,]       2168.521
## [11,]       2161.589
## [12,]       2154.656
## [13,]       2147.724
## [14,]       2140.792
## [15,]       2133.860
## [16,]       2126.928
## [17,]       2119.995
## [18,]       2113.063
## [19,]       2106.131
## [20,]       2099.199
## [21,]       2092.267
## [22,]       2085.334
## [23,]       2078.402
## [24,]       2071.470
## [25,]       2064.538
## [26,]       2057.606
## [27,]       2050.673
## [28,]       2043.741
## [29,]       2036.809
## [30,]       2029.877
## [31,]       2022.945
## [32,]       2016.012
## [33,]       2009.080
## [34,]       2002.148
## [35,]       1995.216
## [36,]       1988.284
## [37,]       1981.351
```

```
## [38,]       1974.419
## [39,]       1967.487
## [40,]       1960.555
## [41,]       1953.622
## [42,]       1946.690
## [43,]       1939.758
## [44,]       1932.826
## [45,]       1925.894
## [46,]       1918.961
## [47,]       1912.029
## [48,]       1905.097
## [49,]       1898.165
## [50,]       1891.233
## [51,]       1884.300
## [52,]       1877.368
## [53,]       1870.436
## [54,]       1863.504
## [55,]       1856.572
## [56,]       1849.639
## [57,]       1842.707
## [58,]       1835.775
## [59,]       1828.843
## [60,]       1821.911
## [61,]       1814.978
## [62,]       1808.046
## [63,]       1801.114
## [64,]       1794.182
## [65,]       1787.250
```

```
return.level(model1,return.period=100,qcov=make.qcov(model1,
  vals = list(mu1 = (65+50))),do.ci=TRUE)
```

```
## fevd(x = vQvk, data = vQvk1, location.fun = ~cas, type = "GEV",
##     method = "MLE")
##
## [1] "Normal Approx."
##
##      95% lower CI Estimate 95% upper CI Standard Error
## [1,]     916.5903  1440.64     1964.689        267.377
```

Izračunamo lahko tudi oceno projektnega pretoka s 100-letno povratno dobo (argument *return.period*) z upoštevanjem intervalov zaupanja (*do.ci* argument) za naš model (model1) 50 let po koncu našega niza merjenih podatkov (argument *qcov*, kjer za parameter, ki je odvisen od časa (*location*), določimo, da nas zanima stanje 50 let po koncu naših podatkov). Vidimo lahko, da so rezultati precej drugačni od dejanskega stanja, ki ga lahko vidimo v drugem delu naših podatkov (torej tisti del, ki ga nismo uporabili za določitev nestacionarnega modela). Očitno je naša predpostavka, da bodo podatki v prihodnosti (drugi del podatkov, torej 50 let, ki jih nismo uporabili za določitev nestacionarnega modela) sledili enakemu trendu kot v preteklosti (prvih 65 let podatkov), precej tvegana. Pri delu z nestacionarnimi modeli je torej potrebna pozornost pri interpretaciji podatkov, rezultatov in

seveda tudi pri dolžini napovedi za prihodnost. Zato je verjetno namesto časovne odvisnosti včasih boljša uporaba odvisnosti od nekaterih drugih spremenljivk, npr. padavin. V tem primeru bi v lokacijski parameter vgradili odvisnost od padavin, za kar potrebujemo še podatke o padavinah (za vsako leto), ki bi jih uporabili za definiranje argumenta *location.fun*. V tem primeru bi nato lahko ocenili, kako povečanje ali zmanjšanje letne količine padavin (mesečne ali količine padavin poljubnega drugega trajanja) vpliva na projektne pretoke z določeno povratno dobo. Posamezne modele lahko potem med seboj primerjamo tudi z uporabo funkcije *lr.test*, ki je vključena v paket *extRemes*.

4.5 Multivariatne verjetnostne analize

Kot alternativa pogosto uporabljenim univariatnim verjetnostnim analizam visokovodnih konic se lahko uporabijo tudi multivariatne verjetnostne analize, kjer v analize vključimo tudi dodatne spremenljivke (npr. volumen in trajanje visokovodnega vala). Funkcije kopula[70] so ena izmed metod, ki jih lahko uporabimo za izvedbo multivariatnih verjetnostnih analiz[71]. Ker so hidrološki procesi pogosto večdimenzionalni oziroma definirani z več spremenljivkami, je v analizah smiselno upoštevati več kot eno spremenljivko. Kopule omogočajo izgradnjo multivariatnega modela, kjer hkrati upoštevamo dve ali več v naravi odvisnih spremenljivk. Multivariatno d-dimenzionalno porazdelitev lahko zapišemo kot kombinacijo funkcije kopula C in robnih porazdelitvenih funkcij. Glavna prednost funkcij kopula pred običajnimi multivariatnimi porazdelitvenimi funkcijami (npr. multivariatna normalna porazdelitev) je prav ločenost robnih porazdelitev in funkcij kopula. Kot robne porazdelitve posameznih spremenljivk lahko izberemo različne porazdelitvene funkcije. Tako lahko pri verjetnostnih analizah poleg konic pretokov hkrati upoštevamo tudi volumne in čase trajanja visokovodnih valov ali druge spremenljivke. Dodatne spremenljivke lahko opišemo z različnimi parametričnimi in neparametričnimi funkcijami. Posamezne spremenljivke so nato združene z uporabo funkcije kopula. Ta koncept lahko uporabimo pri vseh multivariatnih problemih, kjer nastopa določeno število (več kot 1) medsebojno bolj ali manj odvisnih spremenljivk[72]. Tako so bile kopule uporabljene za izvedbo multivariatnih verjetnostnih analiz visokovodnih valov, verjetnostne analize padavin, analize suše, geostatistične interpolacije kot alternativa običajnemu krigingu, preverjanje ustreznosti prelivnega objekta na jezu ter še pri mnogih drugih hidroloških problemih[73].

V naslednjem primeru bomo prikazali postopek izvedbe multivariatnih verjetnostnih analiz visokovodnih valov. Kot izhodišče bomo uporabili podatke, ki so vključeni v paket *lfstat*. Gre za podatke s porečja reke Ngaruroro (Nova Zelandija), katerega prispevno območje vodotoka je 164 km^2. Prvi korak takšnih analiz je določitev vzorca. V našem primeru bomo uporabili podatke o konicah pretokov (največjih letnih), pripadajočih volumnih in trajanju

---

[70] https://doi.org/10.1007/1-4020-4415-1.

[71] https://doi.org/10.1007/978-981-13-0574-0.

[72] https://www.researchgate.net/publication/261031442_Uporaba_kopul_v_hidrologiji

[73] https://doi.org/10.1002/wat2.1579.

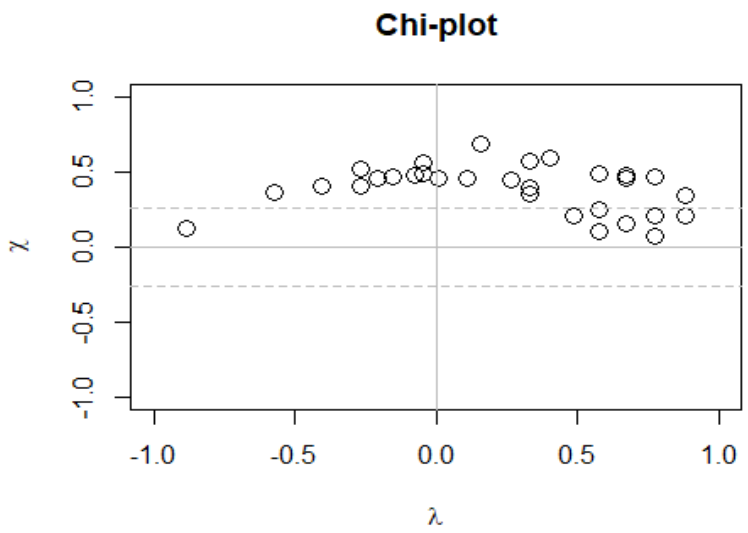
visokovodnih valov. Za izračun volumnov in trajanja valov bomo uporabili tudi podatke o oceni baznega odtoka (na podlagi funkcij v paketu *lfstat*), in sicer bomo predpostavili, da sta volumen in trajanje definirana s točko, ko je površinski del pretoka enak baznemu odtoku. Koda prikazuje, kako lahko določimo konice, volumne in trajanja visokovodnih valov. V prvem koraku bomo najprej določili naš vzorec, torej podatke o konici pretoka, volumnu ter trajanju visokovodnega vala. Za izračun volumna površinskega dela visokovodnega vala bomo najprej izločili bazni odtok, nato pa glede na točki (eno pred konico in eno za konico), kjer sta površinski in bazni odtok enaka, določili začetek in trajanja posameznega visokovodnega vala.

```
library(lfstat, quietly=TRUE)
data(ngaruroro) # uporabimo podatke o pretokih vključenih v paket lfstat
# podatke o pretokih preoblikujemo v format zoo
pret <- zoo(ngaruroro$flow,as.Date(paste0(ngaruroro$year,"-",ngaruroro$month,
"-",ngaruroro$day)))
# ne uporabimo prvega leta podatkov, saj podatki niso popolni
pret <- pret[104:length(pret)]
# podobno tudi v tem primeru izbrišemo prvi del podatkov
ngaruroro <- ngaruroro[-(1:103),]
# na podlagi podatkov določimo podatke o letu
leto1 <- as.numeric(floor(as.yearmon(time(pret))))
# preverimo, kateri dan v letu se pojavi naša konica pretoka (za vsa leta)
letQkat <- aggregate(pret,leto1,which.max)
# konica pretoka, nekatera leta imajo manjkajoče vrednosti
# zato je treba uporabiti argument na.rm=TRUE
letQkon <- aggregate(pret,leto1,max,na.rm=TRUE)
# uporabimo za izračun dejanskega vrstnega reda konic pretokov
letoQkat1 <- (cumsum(aggregate(pret,leto1,length))-365)+letQkat
# pripravimo prazen vektor za shranjevanje podatkov
volumni <- rep(NA,length(letoQkat1))
# enako za trajanja visokovodnih valov
trajanje <- rep(NA,length(letoQkat1))
# za vsakega od največjih letnih dogodkov izračunamo volumen in trajanje vala
for(i in 1:length(letoQkat1)){
# poiščemo začetek visokovodnega vala, to je točka
# ko je volumen površinskega odtoka enak volumnu baznega odtoka
# iskanje v tem primeru poteka za 40 dni pred nastopom konice pretoka
# v primeru večjih vodotokov je smiselno ta čas podaljšati
zac <- last(which((ngaruroro$flow-ngaruroro$baseflow)[(letoQkat1[i]-40):(leto
Qkat1[i])]==0))
# poiščemo lokacijo začetne točke glede na vse podatke o pretokih
zac <- letoQkat1[i]-41+zac
# na podoben način poiščemo tudi konec visokovodnega vala
# iščemo v obdobju 60 dni
kon <- first(which((ngaruroro$flow-ngaruroro$baseflow)[(letoQkat1[i]):(letoQk
at1[i]+60)]==0))
kon <- letoQkat1[i]+kon-1
# izrišemo lahko tudi vse visokovodne valove
# plot(ngaruroro$flow[(zac):(kon)],type="l",main=paste("Dogodek",i))
# lines(ngaruroro$baseflow[(zac):(kon)],col="red")
```

```
# izračunamo volumen (površinski, brez baznega odtoka) visokovodnega vala
# podatki o pretokih so v m3/s, kot rezultat dobimo volumen v m3
volumni[i] <- sum((ngaruroro$flow-ngaruroro$baseflow)[zac:kon])*24*3600
# izračunamo trajanje visokovodnega vala
trajanje[i] <- length((ngaruroro$flow-ngaruroro$baseflow)[zac:kon])
}
# podatke združimo
vzorec <- cbind(letQkon,volumni/10^6,trajanje)
vzorec <- vzorec[-c(15,20,25),]
# iz grafičnega pregleda vidimo, da imamo manjkajoče podatke
# te podatke izbrišemo (dogodki označeni z NA)
```

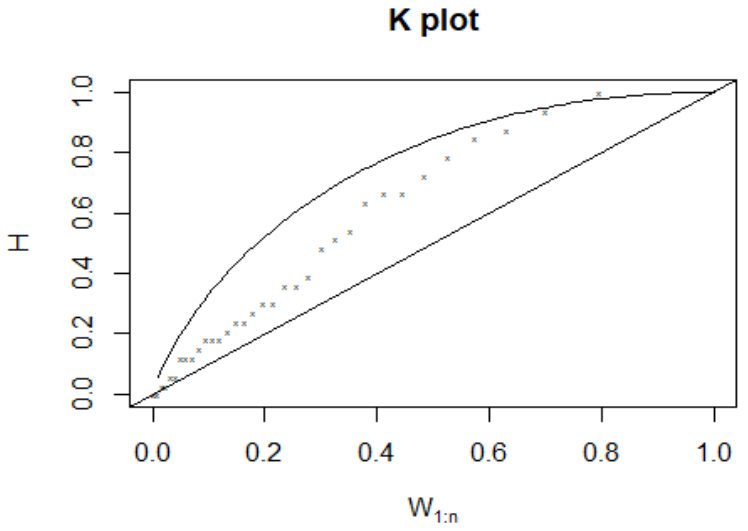
Ko imamo na voljo vzorec, se lahko lotimo analiz. V repozitoriju CRAN je na voljo precej paketov, ki vključujejo funkcije za delo s funkcijami kopula, kot so: *svines*, *igcop*, *copBasic*, *CondCopulas*, *riskSimul*, *acopula*, *VineCopula*, *lcopola*, *copula*. Zelo uporaben je predvsem paket *copula*, ki vključuje številne funkcije za izvedbo analiz. V spodnjem primeru bo prikazana uporaba Chi- in K-grafa. Chi-graf se lahko uporabi za oceno odvisnosti med pari spremenljivk (npr. za konice in volumne), podobno lahko naredimo tudi za ostala dva para spremenljivk (konice–trajanje in volumni–trajanje). Na grafu lambda prikazuje oddaljenost med pari podatkov (Xi,Yi) in središčem raztresenega diagrama. Če točke ležijo nad vrednostjo Chi=0, gre za pozitivno odvisnost, v primeru da so točke pod to premico, pa za negativno odvisnost. Intervala zaupanja označujeta mejo med značilno in neznačilno odvisnostjo. Premica Chi=0 označuje neodvisnost. K-graf se prav tako lahko uporabi za oceno odvisnosti. V spodnjem primeru bomo izrisali odvisnost za volumne in trajanja visokovodnih valov. V primeru K-grafa premica x=y označuje neodvisnost, krivulja pa popolno pozitivno odvisnost med podatki. Negativna odvisnost bi se prikazala pod premico x=y. Večja kot je oddaljenost posamezne točke od premice x=y, večja je odvisnost. Če gre za popolno pozitivno odvisnost, se točka nahaja na krivulji, če pa gre za popolno negativno odvisnost, pa na x-osi.

```
library(copula, quietly=TRUE)

## Warning: package 'copula' was built under R version 4.1.3

library(VineCopula, quietly=TRUE)

##
## Attaching package: 'VineCopula'

## The following object is masked from 'package:copula':
##
##     pobs

vzorec <- as.data.frame(vzorec) # preoblikujemo podatke v podatkovni okvir
colnames(vzorec) <- c("Q","V","T") # spremenimo imena stolpcev
# izračunamo psevdovrednosti, transformiramo podatke v [0,1]
u <- pobs(vzorec)
BiCopChiPlot(u[,1],u[,2],xlim=c(-1,1),ylim=c(-1,1),main="Chi-plot",
  col="black",cex=1.5)
```



Slika 50: Chi-graf za volumne in konice visokovodnih valov.

```
BiCopKPlot(u[,2],u[,3], PLOT=TRUE,main="K plot",col="black")
```



Slika 51: K-graf za volumne in konice visokovodnih valov.

Ko smo ocenili odvisnost med pari spremenljivk, se lahko lotimo ocenjevanja parametrov izbrane kopule. V spodnjem primeru bomo parametre ocenili na podlagi Kendallovega koeficienta korelacije. V paketu *copula* lahko izbiramo med različnimi funkcijami kopula, kot so Joe, Frank, Clayton, Gumbel. Podobno lahko izbiramo med različnimi metodami ocenjevanja parametrov, kot so psevdometoda največjega verjetja (mpl) ali pa metoda na podlagi Kendallovega koeficienta korelacije (itau), ki bo uporabljena v spodnjem primeru. Postopek je podoben tako v bivariatnem kot trivariatnem primeru.

```
# vrednosti Kendallovih koeficientov korelacije med pari spremenljivk
cor(u,method="kendall")

##           Q         V         T
## Q 1.0000000 0.4973262 0.1303322
```

```
## V 0.4973262 1.0000000 0.4634033
## T 0.1303322 0.4634033 1.0000000

# ocenimo parameter oziroma parametre izbrane funkcije kopula
param <- fitCopula(copula=joeCopula(),data=u[,c(1:2)],method="mpl")
# definiramo kopulo
kopul2 <- joeCopula(param@estimate, dim=2)
# na podoben način ocenimo tudi parameter trivariatne kopule
param1 <- fitCopula(copula=frankCopula(dim=3),data=u,method="mpl")
# definiramo trivariatno funkcijo kopula
kopul3 <- frankCopula(param1@estimate, dim=3)
```

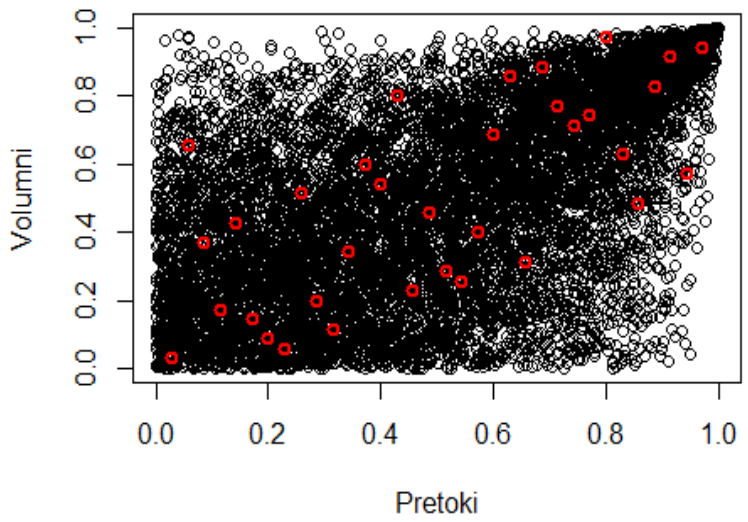
Ker lahko izbiramo med različnimi funkcijami, je večinoma treba poiskati, katere kopule ustrezno opišejo uporabljene vhodne podatke. Paket *copula* vsebuje statične teste, s katerimi lahko ovrednotimo ustreznost izbranih funkcij kopula. V primeru, ki je prikazan spodaj, smo uporabili test Cramér-von Mises. Več podrobnosti o izbranem testu je na voljo pri opisu funkcije *gofCopula.* Vidimo lahko, da je izračunana p-vrednost večja od 0,05, kar pomeni, da je izbrana kopula Joe ustrezna za izvedbo analiz. Podobno lahko preverimo še Gumbelovo kopulo, kjer je smiselno povečati število *N*, kar omogoča bolj natančen izračun testne statistike, je pa zato izračun nekoliko bolj počasen. Podobno lahko naredimo še za trivariatno funkcijo. Tudi v tem primeru testna statistika pokaže, da je izbrana funkcija ustrezna glede na izračunano p-vrednost.

```
# statistični testi ujemanja za različne kopule
gofCopula(copula=joeCopula(),u[,c(1:2)],optim.method = "L-BFGS-B",
    N=500,estim.method="mpl",simulation="mult",method="Sn")

##
##  Multiplier bootstrap-based goodness-of-fit test of Joe copula, dim. d
##  = 2, with 'method'="Sn", 'estim.method'="mpl":
##
## data:  x
## statistic = 0.045758, parameter = 2.3213, p-value = 0.07685

gofCopula(copula=gumbelCopula(),u[,c(1:2)],optim.method = "L-BFGS-B",
    N=500,estim.method="mpl",simulation="mult",method="Sn")

##
##  Multiplier bootstrap-based goodness-of-fit test of Gumbel copula, dim.
##  d = 2, with 'method'="Sn", 'estim.method'="mpl":
##
## data:  x
## statistic = 0.025186, parameter = 1.969, p-value = 0.2884

# trivariatni primer
gofCopula(copula=joeCopula(dim=3),u,optim.method = "L-BFGS-B",
    N=500,estim.method="mpl",simulation="mult",method="Sn")

##
##  Multiplier bootstrap-based goodness-of-fit test of Joe copula, dim. d
##  = 3, with 'method'="Sn", 'estim.method'="mpl":
```

```
##
## data:  x
## statistic = 0.060341, parameter = 1.8385, p-value = 0.1347
```

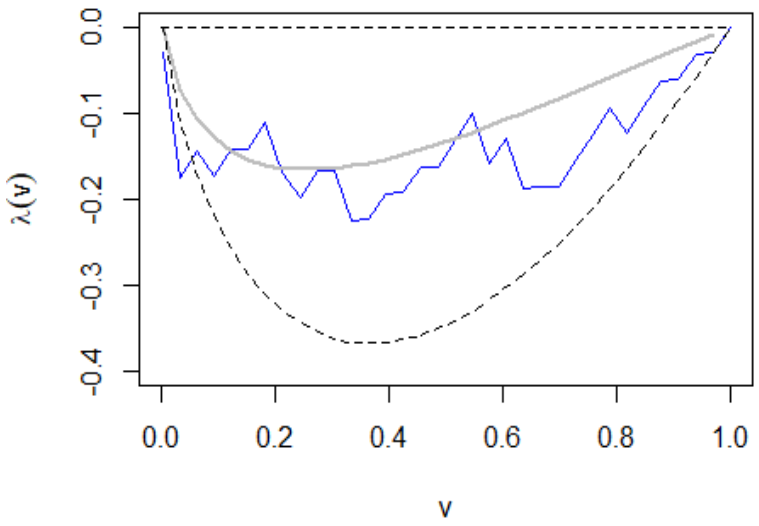
Preverimo pa lahko tudi grafično ujemanje z merjenimi podatki. Na podlagi bivariatne kopule Joe lahko generiramo naključen vzorec psevdovrednosti, ki jih lahko primerjamo z dejanskimi, in na ta način grafično ocenimo ustreznost izbrane funkcije kopula. Ker smo na graf dodali še dejanske podatke, vidimo, da izbrana kopula da večji poudarek na zgornjem repu (angl. *tail*), kar pa ni lastnost naših podatkov; posledično kopula Joe morda ni najbolj primerna za izvedbo bivariatnih analiz in bi veljalo preveriti ustreznost kakšne druge funkcije. To lahko potrdi tudi izračun indeksa *lower in upper tail dependence* (primer spodaj). Obstajajo pa tudi dodatni grafični prikazi, s katerimi lahko izberemo najbolj ustrezno funkcijo kopula, kot je na primer diagram lambda, ki je prikazan spodaj, kjer modra črta prikazuje podatke, siva pa oceno na podlagi kopule Joe.

```
gen2 <- rCopula(10000,kopul2) # generiramo podatke
plot(gen2,xlab="Pretoki",ylab="Volumni") # izrišemo graf
points(u[,c(1,2)],col="red",lty=2,lwd=2) # dodamo točke
```



Slika 52: Grafično ujemanje merjenih in generiranih podatkov v primeru multivariatnih analiz.

```
lambda(joeCopula(param@estimate)) # izračun indeksov upper/lower tail

##    lower    upper
## 0.000000 0.652018

# diagram lambda
BiCopLambda(u1=u[,1],u2=u[,2],family=6,par=param1@estimate,col="blue")
```



Slika 53: Diagram lambda za oceno ustreznosti izbrane kopule.

Simetrične kopule lahko uporabimo samo v primeru, če so spremenljivke »zamenljive«. Posledično je v paketu *copula* na voljo tudi test, s katerim lahko preverimo, ali sta spremenljivki zamenljivi. V spodnjem primeru vidimo, da je izračunana p-vrednost večja od 0, kar pomeni, da lahko uporabimo simetrične funkcije kopula, kot so Joe, Frank, Clayton, Gumbel-Hougaard. Podobno lahko naredimo tudi za ostale pare spremenljivk. Paket *copula* vključuje tudi funkcijo *xvCopula*, ki izvede navzkrižno validacijo in s katero lahko primerjamo ustreznost različnih funkcij kopula (višja kot je vrednost testne statistike, bolj ustrezna naj bi bila posamezna funkcija). Obstaja pa velika množica različnih funkcij kopula. Za multivariatne primere velja omeniti tudi asimetrične kopule Khoudraji-Liebscher, ki se pogosto uporabljajo v hidroloških analizah.

```
exchTest(u[,c(1,2)], N =100) # test zamenljivosti

##
##  Test of exchangeability for bivariate copulas with argument 'm' set to
##  0
##
## data:  u[, c(1, 2)]
## statistic = 0.019031, p-value = 0.4604

xvCopula(copula=joeCopula(),x=u[,c(1,2)]) # xvCopula test

## [1] 6.159292

xvCopula(copula=frankCopula(),x=u[,c(1,2)])

## [1] 9.039083

# vidimo, da je kopula Frank bolj ustrezna kot kopula Joe
xvCopula(copula=gumbelCopula(),x=u[,c(1,2)])

## [1] 8.454546
```

```
# Gumbel-Hougaard je nekoliko manj ustrezna kot Frank
fitCopula(khoudrajiCopula(copula1 = gumbelCopula(dim=3),
copula2=joeCopula(dim=3)),start = c(1.7,3.1, 0.5,0.6,0.05),
u,optim.method = "Nelder-Mead")

## Call: fitCopula(khoudrajiCopula(copula1 = gumbelCopula(dim = 3), copula2 =
joeCopula(dim = 3)),
##     data = u, ... = pairlist(start = c(1.7, 3.1, 0.5, 0.6, 0.05), optim.me
thod = "Nelder-Mead"))
## Fit based on "maximum pseudo-likelihood" and 34 3-dimensional observations
.
## Copula: khoudrajiExplicitCopula
## c1.alpha c2.alpha   shape1   shape2   shape3
##   2.1028   3.6646   0.6765   0.2631   0.0183
## The maximized loglikelihood is 21.07
## Optimization converged

# asimetrična kopula Khoudraji-Liebscher
```

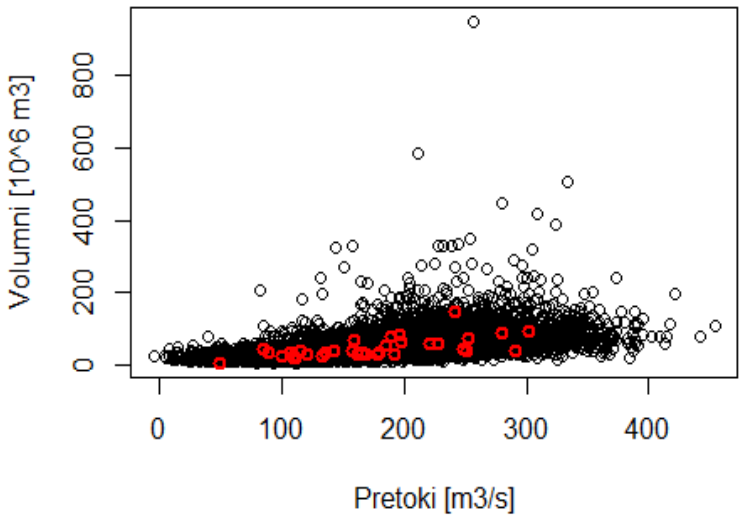
Potem ko smo na podlagi ustreznih grafičnih in statističnih testov izbrali ustrezno funkcijo kopula ter ko imamo definirane tudi robne porazdelitve posameznih spremenljivk, lahko določimo tudi različne povezave med povratnimi dobami (npr. OR ali AND povratna doba)[74] ter vrednostmi spremenljivk. V primeru multivariatnih analiz obstajajo tudi številne druge definicije povratnih dob, kot so pogojne ali pa sekundarne[75]. V primeru spodaj bomo uporabili kopulo Frank in porazdelitev GEV kot robno porazdelitev za volumne in konice pretokov. Izračunali bomo vrednosti povratnih dob OR in AND za primer največje konice pretoka v vzorcu in za največji volumen visokovodnega vala. Pri tem je treba poudariti, da se dejansko največja konica in volumen nista zgodila v sklopu istega dogodka. Povratna doba OR v primeru, da se zgodi ali največja konica pretoka ali največji volumen vala, znaša približno 21 let (primer spodaj). Povratna doba AND v primeru, da se sočasno zgodita tako največja konica pretoka kot največji volumen vala, je v tem primeru (prikazano spodaj) veliko večja, saj je verjetnost, da se sočasno zgodita tako največja konica kot volumen, seveda precej manjša.

```
# parametri kopule
paramf <- fitCopula(copula=frankCopula(),u[,c(1,2)],method="mpl")
# definiramo našo kopulo
kopul2 <- frankCopula(paramf@estimate, dim=2)
# definiramo robne porazdelitve
parGEVQ <- lmom2par(lmoms(vzorec[,1]),type="gev")
parGEVV <- lmom2par(lmoms(vzorec[,2]),type="gev")
# generiramo vzorec
gen <- rCopula(10000,kopul2)
# grafična primerjava
```

[74] https://doi.org/10.1002/hyp.10145.

[75] https://doi.org/10.5194/hess-17-1281-2013.

```
plot(quagev(gen[,1],parGEVQ),quagev(gen[,2],parGEVV),xlab="Pretoki [m3/s]",
     ylab="Volumni [10^6 m3]")
points(vzorec[,c(1,2)],col="red",lty=2,lwd=2)
```



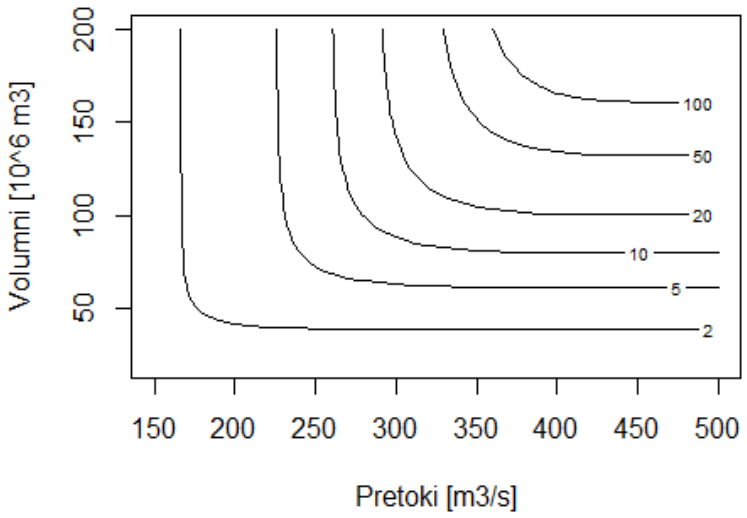
Slika 54: Primer grafičnega ujemanja merjenih in generiranih podatkov za kopulo Frank.

```
# izračun povratnih dob OR in AND
dl1 <- cdfgev(max(vzorec[,1]),parGEVQ)
dl2 <- cdfgev(max(vzorec[,2]),parGEVV)
vmes <- pCopula(c(dl1,dl2),kopul2)
# izračunamo povratno dobo OR
povratnaOR <- 1/(1-vmes)
# izračunamo povratno dobo AND
povratnaAND <- 1/(1-dl1-dl2+vmes)
povratnaOR

## [1] 21.34922

povratnaAND

## [1] 423.8661
```

Izračunane povratne dobe OR pa lahko naredimo za različne kombinacije pretokov in volumnov. V spodnjem primeru definiramo prazno matriko, kamor bomo shranili rezultate. Izračune bomo naredili v dveh zankah, najprej za pretoke in nato za volumne. V prvem koraku bomo izračunali rezultate za robne porazdelitve ter nato še za povratne dobe OR in AND. Izrisali bomo tudi graf izolinij za različne vrednosti povratnih dob.

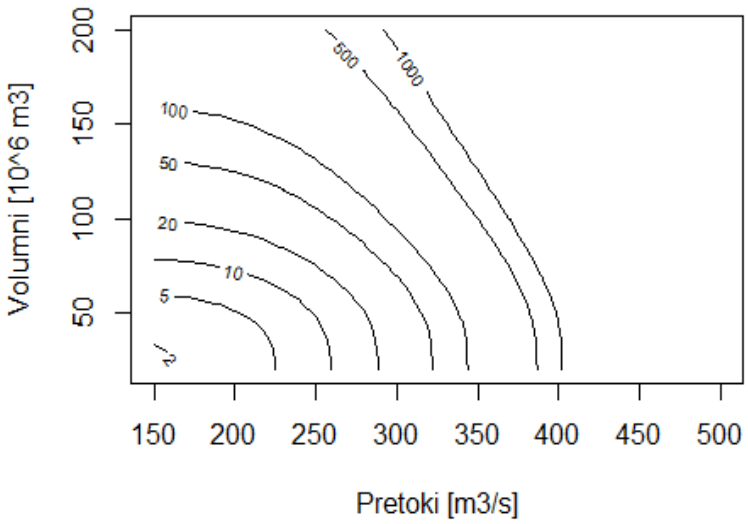
```
# povratna doba OR
matrikaOR <- matrix(NA,nrow=length(seq(from=150,to=500,by=10)),
  ncol=length(seq(from=20,to=200,by=5)))
for(i in 1:length(seq(from=150,to=500,by=10))){
  for(j in 1:length(seq(from=20,to=200,by=5))){
    dl1 <- cdfgev(seq(from=150,to=500,by=10)[i],parGEVQ)
    dl2 <- cdfgev(seq(from=20,to=200,by=5)[j],parGEVV)
    vmes <- pCopula(c(dl1,dl2),kopul2)
```

```
    matrikaOR[i,j] <- 1/(1-vmes) # povratna doba OR
  }
}
# graf kontur za različne vrednosti povratnih dob
contour(seq(from=150,to=500,by=10),seq(from=20,to=200,by=5),
matrikaOR,levels=c(2,5,10,20,50,100),xlab="Pretoki [m3/s]",
ylab="Volumni [10^6 m3]")
```



Slika 55: Graf izolinij za povratno dobo OR.

```
# povratna doba AND
matrikaAND <- matrix(NA,nrow=length(seq(from=150,to=500,by=10)),
  ncol=length(seq(from=20,to=200,by=5)))
for(i in 1:length(seq(from=150,to=500,by=10))){
  for(j in 1:length(seq(from=20,to=200,by=5))){
    dl1 <- cdfgev(seq(from=150,to=500,by=10)[i],parGEVQ)
    dl2 <- cdfgev(seq(from=20,to=200,by=5)[j],parGEVV)
    vmes <- pCopula(c(dl1,dl2),kopul2)
    matrikaAND[i,j] <- 1/(1-dl1-dl2+vmes)
  }
}
contour(seq(from=150,to=500,by=10),seq(from=20,to=200,by=5),
matrikaAND,levels=c(2,5,10,20,50,100,500,1000),xlab="Pretoki [m3/s]",
ylab="Volumni [10^6 m3]")
```



Slika 56: Graf izolinij za povratno dobo AND.

Na podoben način lahko izvedemo bivariatne analize tudi za druge spremenljivke[76], naredimo multivariatne verjetnostne analize[77] ali pa funkcije kopula uporabimo za ocenjevanje neznanih vrednosti[78] oziroma za določitev intervalov zaupanja[79].

Naloga 35: Na podlagi kopule Joe z vrednostjo parametra 2 generirajte bivariatni vzorec s 40 elementi. Na podlagi generiranega vzorca ocenite parametre kopule Frank in preverite njeno ustreznost za opis tega vzorca. Dodatno grafično primerjajte generirani vzorec na podlagi kopule Frank (N = 500) in kopule Joe.

4.6 Analize sezonskosti

Podatki o sezonskih značilnostih konic (ali nizkih) pretokov so lahko eden izmed indikatorjev podnebnih sprememb[80]. Hidrološke razmere v vodotokih so namreč rezultat vpliva padavin, vlažnosti tal in snežnih razmer v porečju. Posledično je pogosto smiselno analizirati tudi sezonske značilnosti teh spremenljivk in preučiti, ali se sezonskost spreminja ali ne. Sezonskost poplav je mogoče grafično prikazati v obliki krožne statistike z enačbami, ki jih

---

[76] https://doi.org/10.3390/w10080995.

[77] https://doi.org/10.1007/s11269-014-0606-2.

[78] https://doi.org/10.3390/w9080628.

[79] https://doi.org/10.1007/s10346-019-01169-9.

[80] https://doi.org/10.1126/science.aan2506.

podaja Burn (1997)[81], in obravnavajo nekatera druga dela[82]. Tudi na področju krožne statistike je na voljo veliko različnih testov, s katerimi lahko preverimo različne hipoteze. V tem kontekstu velja omeniti paket *CircStats*[83], ki vsebuje veliko takšnih funkcij. V naslednjem primeru bomo prikazali analizo sezonskosti največjih letnih konic za podatke, ki so vključeni v paketu *lfstat* in ki smo jih uporabili tudi v prejšnjem poglavju. Najprej bomo izračunali kote časovnih pojavov teh dogodkov v radianih in nato povprečni čas nastopa vseh konic. Nato bo prikazan izračun koeficienta sezonskosti, ki lahko zavzame vrednost med 0 in 1. Vrednost blizu 1 pomeni, da je sezonskost zelo izrazita in praktično vsi dogodki nastopijo v podobnem delu leta, vrednost blizu 0 pa pomeni, da ni izrazite sezonskosti. Uporabili bomo tudi sezonski prikaz največjih letnih konic (*vQvk*), kjer en krog predstavlja eno letno konico, velikost kroga je magnituda dogodka.

```
dan <- letQkat
# izračun kote nastopa v radianih
kot <- (dan-0.5)*(2*pi/365)
# izračunamo povprečni čas
xpov <- mean(cos(kot));x <- cos(kot)
ypov <- mean(sin(kot));y <- sin(kot)
r <- sqrt(xpov*xpov+ypov*ypov)
# izračunamo povprečni dan glede na izračunan r
if(xpov>=0 && ypov>=0) dan.rez <- (atan(ypov/xpov))*(365/(2*pi)) else
  if(xpov<0) dan.rez <- (atan(ypov/xpov) + pi)*(365/(2*pi)) else
    if (ypov<0 && xpov >= 0) dan.rez <- (atan(ypov/xpov) + 2*pi)*(365/(2*pi))
else
      end
# izračunan dan
dan.rez
```

```
## [1] 186.2764
```

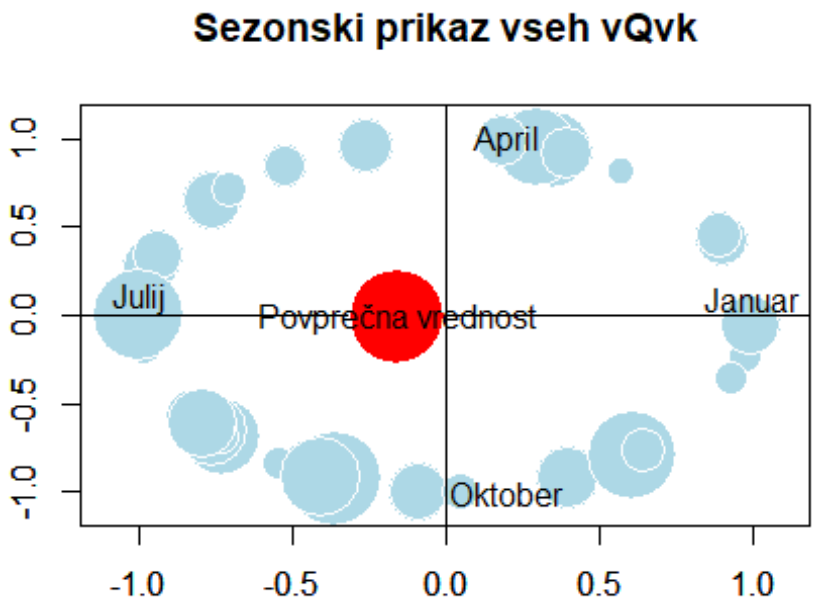
```
symbols(x,y,letQkon,inches=0.25,bg="lightblue",xlab="",ylab="",
xlim=c(-1.1,1.1),ylim=c(-1.1,1.1),fg="white",
main="Sezonski prikaz vseh vQvk");abline(0,0);abline(v=0)
par(new=TRUE)
symbols(xpov,ypov,mean(letQkon),inches=0.25,bg="red",
xlab="",ylab="",xlim=c(-1.1,1.1),ylim=c(-1.1,1.1),fg="white")
# tekstovne oznake na grafu
text(xpov,ypov,"Povprečna vrednost")
text(1,0.1,labels="Januar")
text(0.2,1,labels="April")
text(-1,0.1,labels="Julij")
text(0.2,-1,labels="Oktober")
```

---

[81] https://doi.org/10.1016/S0022-1694(97)00068-1.

[82] https://doi.org/10.1111/jfr3.12118.

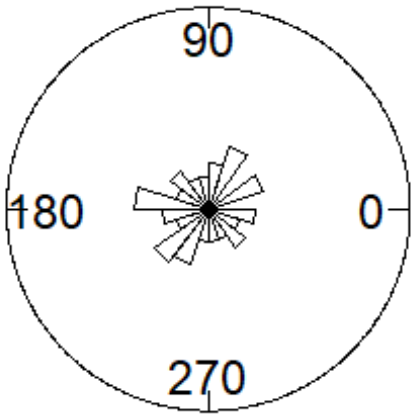
[83] https://cran.r-project.org/web/packages/CircStats/CircStats.pdf.



Slika 57: Sezonski prikaz največjih letnih konic.

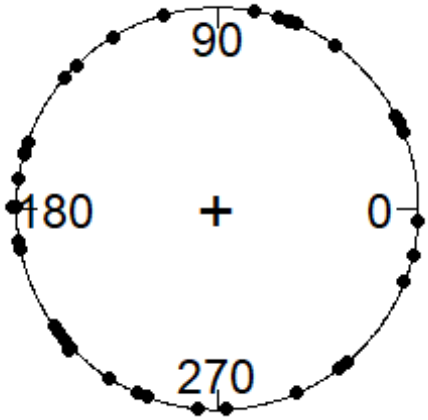
Podoben prikaz, kot smo ga zgoraj naredili ročno, lahko naredimo tudi z vgrajenimi funkcijami v paket *CircStats*. Diagram *Rose* se uporablja pri izrisu rože vetrov. Prikazali bomo vse dogodke na krožnici z uporabo krožnega grafa in izračunali osnovno statistiko, kjer je rezultat *rho* enak kot indeks sezonskosti *r*. Prikazan pa je tudi test, s katerim lahko preverimo, ali so v vzorcu prisotne spremembe v smeri in magnitudi trenda. Več informacij o funkciji in rezultatih lahko najdete pri opisu funkcije *change.pt*.

```
library(CircStats, quietly=TRUE)

## Warning: package 'CircStats' was built under R version 4.1.3

##
## Attaching package: 'boot'

## The following object is masked from 'package:lattice':
##
##     melanoma

rose.diag(kot,bins=20)
```



Slika 58: Rose diagram za podatke o sezonskosti največjih letnih pretokov.

```
circ.plot(kot,stack=TRUE,bins=360)
```



Slika 59: Krožni diagram vseh največjih letnih pretokov.

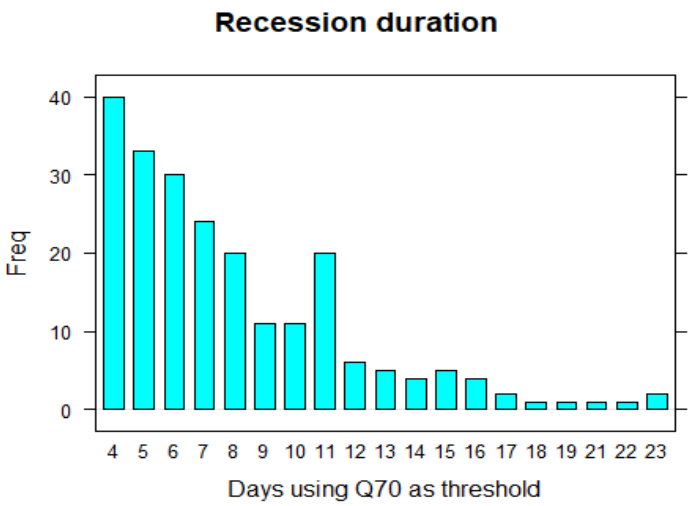
```
circ.summary(kot) #  rho

##    n  mean.dir       rho
## 1 37 -3.076585 0.1566884

change.pt(kot) # test sprememb

##    n       rho     rmax k.r     rave      tmax k.t       tave
## 1 37 0.1566884 4.547054  26 1.152155 0.1167458  35 0.04501709
```

Naloga 36: Za podatke z ene izmed postaj ARSO izvedite analize sezonskosti: (i) izračunajte koeficient sezonskosti in ii) pripravite krožni graf, kjer je prikazan čas nastopa izbranih dogodkov. Analize ponovite za vzorec največjih letnih konic (AM) in za vzorec POT, kjer povprečno izberite tri dogodke nad izbranim pragom na leto (POT3).

4.7 Analize padajočih delov hidrogramov

Postopno izcejanje vode, shranjene v porečju v obdobjih z malo ali nič padavinami, se odraža v obliki recesijskega dela hidrograma (tj. padajoči del hidrograma od konice pretoka do konca površinskega dela odtoka). Recesijska krivulja celostno opisuje, kako različni, z vodo bogati deli porečja in procesi v porečju nadzorujejo dotok vode v vodotok. Reke s počasno hitrostjo upada recesijskega dela hidrograma so večinoma tesno povezane s podtalnico ali jezeri, medtem ko je hitro znižanje pretoka (in tudi naraščanje) značilno za hudournike. Analiza recesijskega dela hidrograma se je izkazala za uporabno pri številnih vidikih upravljanja vodnih virov, vključno z napovedovanjem nizkih pretokov in oceno spremenljivk nizkih pretokov na merilnih mestih. Pogosto so takšni podatki uporabni tudi v primeru regionalnih verjetnostnih analiz ter modeliranju padavine–odtok[84]. Pregled različnih metod in osnovnih enačb je na voljo v članku, ki ga je pripravila Tallaksen (1995)[85]. Nekatere funkcije za analizo recesijskega dela hidrograma so vključene tudi v paket *lfstat*. Dodatno je zanimiv tudi prispevek[86], ki prikazuje vpliv izbire različnih kriterijev na vrednost recesijskih konstant nizkih pretokov, in sicer vpliv izbire metode izračuna, vpliv izbire dolžine segmenta recesijske krivulje in vpliv izbire obdobja, na podlagi katerega se določi začetni pretok, ki določa mejo za vključitev padajočih delov hidrograma v recesijsko analizo.

```
library(lfstat, quietly=TRUE)
# trajanje padajočega dela hidrograma glede na mejno vrednost
seglenplot(ngaruroro,threslevel = 70)
```



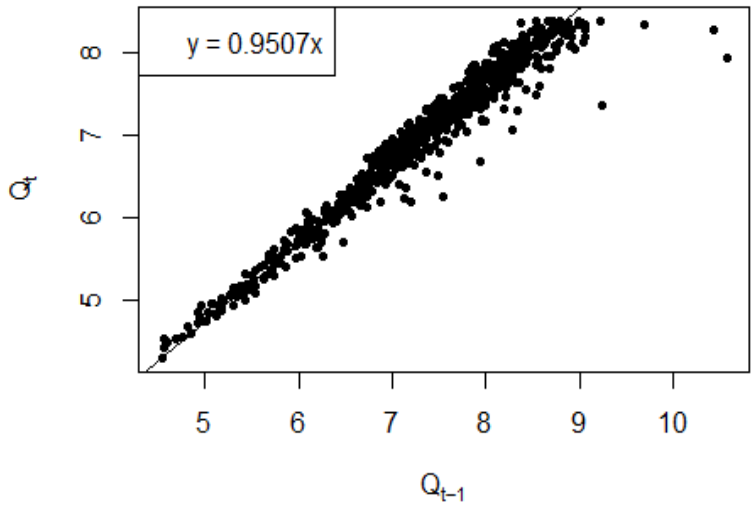
Slika 60: Analiza padajočega dela hidrogramov.

[84] https://library.wmo.int/idurl/4/32176.

[85] https://doi.org/10.1016/0022-1694(94)02540-R.

[86] https://doi.org/10.15292/acta.hydro.2019.01.

```
# primer glavne recesijske krivulje (MRC) z dolžino segmenta 5 dni
# in vrednostjo Q70 kot mejnim pragom
recession(ngaruroro,method = "MRC",seglen = 7,threshold = 70)
```



Slika 61: Primer glavne recesijske krivulje.

```
## [1] 19.77262

# funkcija za analizo sezonskosti nizkih pretokov
seasindex(ngaruroro)

## $theta
## [1] 1.088009
##
## $D
## [1] 63.20411
##
## $r
## [1] 0.8554912
```

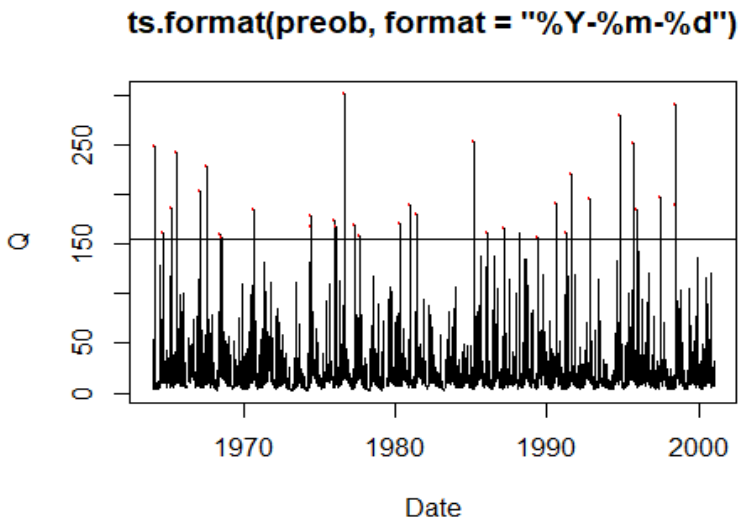
4.8 Določitev vzorca glede na metodo nad izbranim pragom (POT)

Za izvedbo verjetnostnih analiz visokovodnih konic se običajno uporabljata dva različna pristopa[87]. Pri metodi letnih maksimumov (AM) vzorec sestavljajo največji pretoki v vsakem posameznem letu. Vzorec torej vsebuje toliko elementov, kolikor let podatkov uporabimo za analizo. Pri tako oblikovanem vzorcu se pogosto zgodi, da iz analize izpustimo dogodke, ki sicer niso največji v posameznem letu, vendar so večji od izbranega dogodka, ki se je zgodil v sušnem letu, ko pretoki niso bili tako veliki. Temu se izognemo z možnostjo oblikovanja vzorca vseh dogodkov, ki ležijo nad določeno mejno vrednostjo oziroma pragom (metoda POT). Pri metodi POT (angl. *peaks over threshold*) ne upoštevamo le letnih maksimumov,

[87] https://doi.org/10.1080/02626667.2013.831174.

ampak se glede na izbrano vrednost praga oblikuje obsežnejši vzorec[88]. V programskem okolju R je na voljo več različnih paketov, ki omogočajo določitev vzorca po metodi POT, eden izmed teh je tudi paket *hydrostats* (na voljo tudi na spletni strani GitHub), s katerim lahko oblikujemo vzorec POT. Spodnji primer prikazuje oblikovanja vzorca vseh dogodkov nad izbranim pragom. Uporabili bomo funkcijo *partial.series*, kjer argument *ari* definira število dogodkov nad izbranim pragom (oziroma povprečno povratno dobo), argument *ind.days* pa določa kriterij neodvisnosti med dvema zaporednima dogodkoma. Če želimo določiti v povprečju pet dogodkov nad izbranim pragom (POT5), je povprečna povratna doba enaka 1/5=0,2. Hkrati smo določili, da mora biti med dvema dogodkoma vsaj sedem dni (*ind.days*), kot je to prikazano spodaj. Paket *hydrostats* pa vsebuje tudi druge uporabne funkcije, s katerimi lahko analiziramo konice pretokov (tudi nizke pretoke), kot je na primer funkcija *high.spells*, s katero lahko analiziramo čas nastopa, pogostost ter trajanje različnih dogodkov. Dodatno lahko določimo nekatere sezonske značilnosti, kot so povprečni mesečni pretoki ali procent letnega odtoka v šestih najbolj sušnih mesecih. Več informacij je na voljo v opisu funkcije *seasonality*.

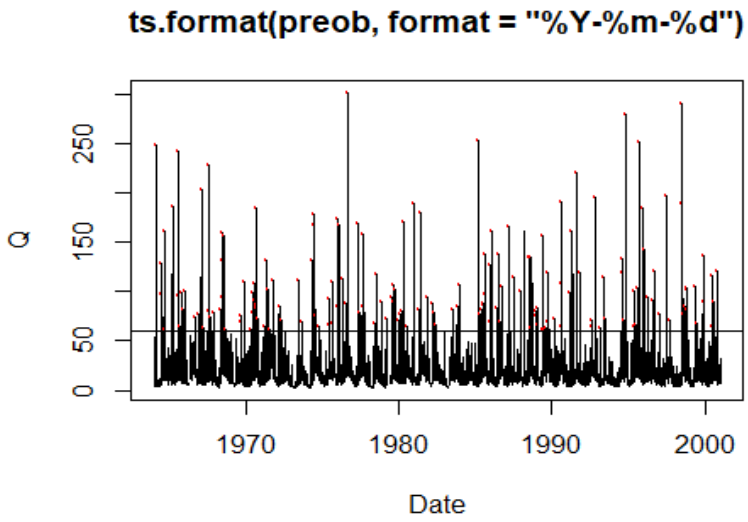
```
library(hydrostats, quietly=TRUE)

## Warning: package 'hydrostats' was built under R version 4.1.3

# podatkovni okvir, takšen format zahteva paket hydrostats
preob <- data.frame(as.character(time(pret)),as.numeric(coredata(pret)))
colnames(preob) <- c("Date","Q")
# letni maksimumi
partial.series(ts.format(preob,format="%Y-%m-%d"), ari=1, plot=TRUE, ind.days
=1)
```



Slika 62: Hidrogram in letni maksimumi (rdeče točke).

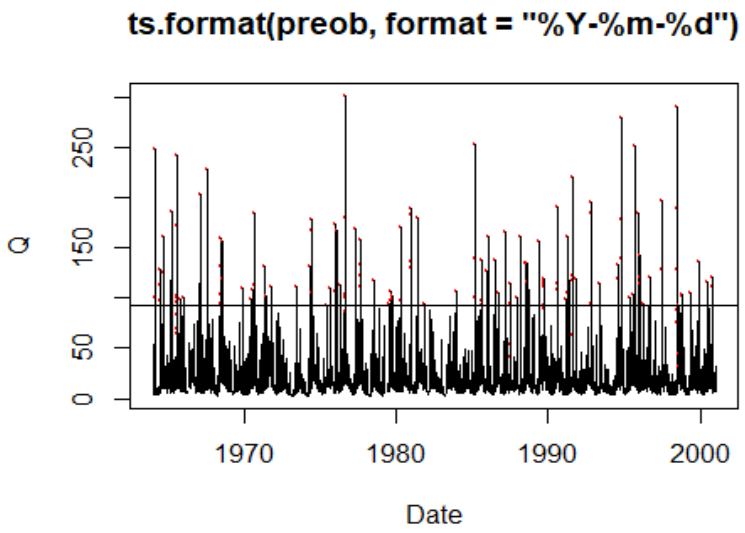
[88] https://repozitorij.uni-lj.si/Dokument.php?id=98022&lang=slv.

```
##   ari n.years n.events flow.threshold avg.duration max.duration
## 1   1      33       33        156.517     1.176471            3
##   med.spell.volume
## 1          28.2795

# povprečno pet dogodkov nad pragom in sedem dni med dogodki
partial.series(ts.format(preob,format="%Y-%m-%d"), ari=0.2, plot=TRUE, ind.da
ys=7)
```



Slika 63: Povprečno 5 dogodkov nad pragom (POT5).

```
##   ari n.years n.events flow.threshold avg.duration max.duration
## 1 0.2      33      165         61.647     1.875648           10
##   med.spell.volume
## 1           30.362

# izpis vseh dogodkov, oziroma samo prvih šest od vseh
head(partial.series(ts.format(preob,format="%Y-%m-%d"), ari=0.2, plot=FALSE,
  ind.days=7,series = TRUE)$p.series)

##             Date       Q hydro.year event.rank
## 4636  1976-09-09 301.535       1976          1
## 12602 1998-07-02 290.657       1998          2
## 11270 1994-11-08 280.164       1994          3
## 7745  1985-03-15 252.869       1985          4
## 11573 1995-09-07 251.044       1995          5
## 71    1964-03-11 248.107       1964          6

# primer uporabe funkcije high.spells
high.spells(ts.format(preob,format="%Y-%m-%d"),quant=0.99)
```



Slika 64: Uporaba funkcije high.spells s paketa hydrostats za določitev vzorca.

```
##   high.spell.threshold n.events spell.freq       ari min.high.spell.durati
on
## 1               92.009       85   2.575758 0.3882353
1
##   avg.high.spell.duration med.high.spell.duration max.high.spell.duration
## 1                1.729412                       1                       9
##   avg.spell.volume avg.spell.peak sd.spell.peak avg.rise avg.fall avg.max.
ann
## 1         66.13016       30.36957        34.344 84.83268 48.86572     172.
669
##   cv.max.ann flood.skewness ann.max.timing ann.max.timing.sd ann.max.min.d
ur
## 1     37.263       10.02056            200               103
1
##   ann.max.avg.dur ann.max.max.dur ann.max.cv.dur
## 1         3.30303               9       59.82942

high.spell.lengths(ts.format(preob,format="%Y-%m-%d"), threshold=200)

##    start.date spell.length
## 2  1964-03-11            1
## 4  1965-08-15            1
## 6  1967-02-03            1
## 8  1967-08-12            1
## 10 1976-09-09            1
## 12 1985-03-15            1
## 14 1991-08-09            1
## 16 1994-11-08            1
## 18 1995-09-07            1
## 20 1998-07-02            1

# funkcija seasonality
seasonality(ts.format(preob,format="%Y-%m-%d"), monthly.range=TRUE)
```

```
## $seasonality
## [1] 40.65079
##
## [[2]]
##       01       02       03       04       05       06       07       08
## 386.1272 289.4735 356.9346 384.5619 494.7906 674.8001 811.1845 840.7488
##       09       10       11       12
## 735.8962 582.4253 447.5639 413.2396
##
## $avg.ann.month.range
## [1] 1007.311
##
## $max.min.time.dif
## [1] 5
```

4.9 Padavinski indeksi

Analiza padavin v vodarstvu je ključnega pomena za razumevanje in upravljanje vodnih virov. Padavine imajo kot glavni vhodni element hidrološkega kroga temeljno vlogo pri oblikovanju razpoložljivosti in gibanju vode. Tako nas pogosto zanima prostorska in časovna spremenljivost padavin in njihove lastnosti[89]. Analiziranje prostorske porazdelitve padavin vključuje razumevanje, kako se padavine spreminjajo na različnih lokacijah. Analiza časovne porazdelitve vključuje preučevanje spreminjanja padavin v času, kot so sezonski vzorci, mesečni trendi in dnevni cikli. V inženirski praksi se pogosto srečujemo z uporabo krivulj intenziteta–trajanje–povratna doba (ITP)[90]. Te krivulje so v pomoč pri načrtovanju grajene infrastrukture, kot so sistemi za odvodnjavanje padavinske vode (krivulje ITP so vhodni podatek za racionalno enačbo), in so bistvene za oceno poplavne ogroženosti. Analize padavin pa so lahko tudi posredni indikator sušnih razmer preko analize različnih padavinskih indeksov, kot je standardizirani padavinski indeks (SPI)[91]. Prikazali bomo nekaj primerov analiz, ki so lahko uporabne na področju vodarstva. Prvi primer bo izračun indeksa SPI[92]. Najprej bo prikazan uvoz podatkov in priprava podatkov s časovnim korakom 30 minut. Podatki so strukturirani v treh stolpcih: v prvem je podatek o datumu, v drugem podatek o uri in v zadnjem količina padavin v 30 minutah (v mm).

```
# 30-minutni podatki o padavinah s postaje Lj.-Bežigrad
data <- read.table("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/Ljubljana-padavine.txt",header=FALSE, sep=" ")
head(data)
```

[89] https://www.mvd20.com/LETO2019/R11.pdf.

[90] https://www.dlib.si/details/URN:NBN:SI:doc-R43P4I4S.

[91] https://www.mvd20.com/LETO2012/R2.pdf.

[92] https://dlib.si/stream/URN:NBN:SI:DOC-S1CQI4ZZ/2a8c2c5e-cb31-4ff4-ab34-c845a264e66b/PDF.

```
##           V1       V2 V3
## 1 2010-01-01 00:00:00  0
## 2 2010-01-01 00:30:00  0
## 3 2010-01-01 01:00:00  0
## 4 2010-01-01 01:30:00  0
## 5 2010-01-01 02:00:00  0
## 6 2010-01-01 02:30:00  0
```

```
# podatke združimo v podatkovni okvir
dataP <- data.frame(Datum=as.POSIXct(paste0(data[,1]," ",data[,2]),
  format="%Y-%m-%d %H:%M:%S"),P=((data[,3])))
# preverimo, ali imamo v podatkih morda kakšne podvojene zapise
head(dataP[duplicated(dataP[,1]),])
```

```
##                     Datum P
## 14549 2010-10-31 02:00:00 0
## 14550 2010-10-31 02:30:00 0
## 32021 2011-10-30 02:00:00 0
## 32022 2011-10-30 02:30:00 0
## 49493 2012-10-28 02:00:00 0
## 49494 2012-10-28 02:30:00 0
```

```
# gre za datume, ki so povezani s prestavljanjem ure jeseni
# v primeru, da bi imeli del manjkajočih podatkov,
# bi lahko generirali zvezni časovni vektor
time.seq <- seq(as.POSIXct("1.1.2010 00:00:00",format="%d.%m.%Y %H:%M:%S"),
as.POSIXct("31.12.2020 23:00:00",format="%d.%m.%Y %H:%M:%S"), by = "30 mins")
length(time.seq)==dim(dataP)[1]
```

```
## [1] TRUE
```

```
# podatki dataP vsebujejo vse podatke, ki jih potrebujemo
```

Če bi imeli podvojene podatke in bi jih želeli izbrisati, bi lahko uporabili postopek, ki je prikazan spodaj. Tako bi lahko generirani časovni vektor primerjali z dejanskimi podatki in podatke združili glede na podatek o času.

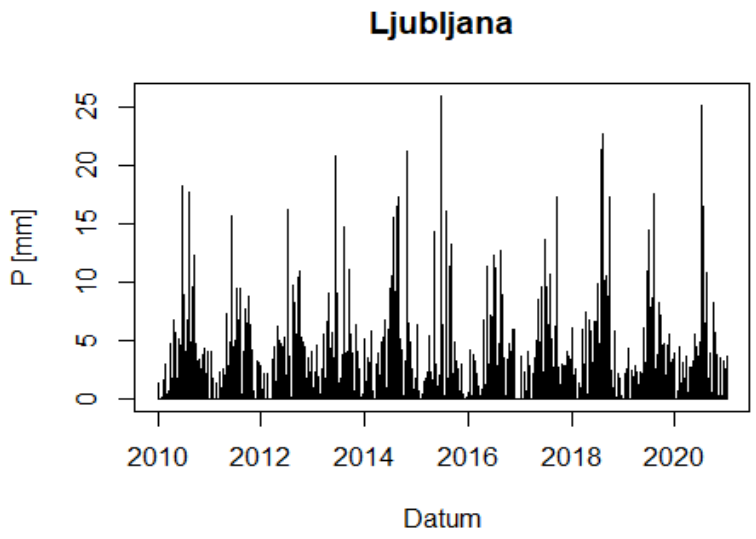
```
dataP <- dataP[-which(duplicated(dataP[,1])==TRUE),]
dataPsk <- merge(dataP,as.data.frame(time.seq),by.x="Datum",by.y="time.seq")
```

Podatke o padavinah bomo preoblikovali v objekt *zoo*.

```
# osnovne lastnosti 30-minutnih podatkov o padavinah
summary(dataP[,2])
```

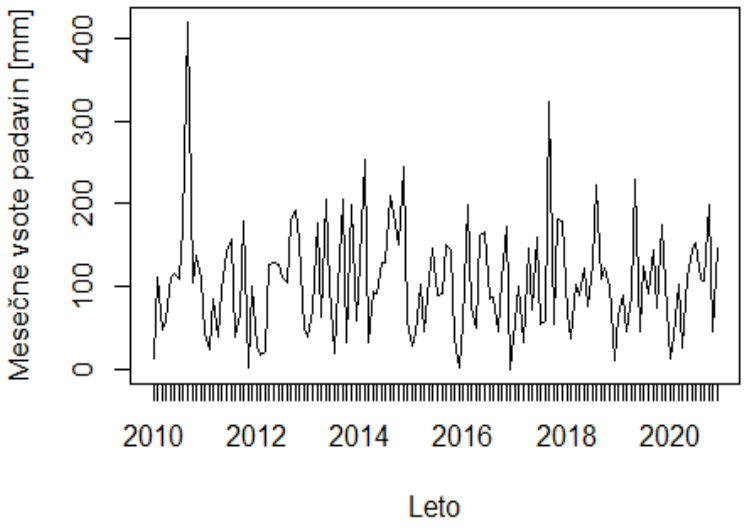
```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  0.00000  0.00000  0.00000  0.07252  0.00000 26.00000
```

```
# izrišemo osnovni graf uporabljenih podatkov
plot(dataP,type="h",ylab="P [mm]",main="Ljubljana")
```



Slika 65: 30-min podatki o padavinah.

```
# podatke preoblikujemo v objekt zoo
Pzoo <- zoo(dataP[,2],dataP[,1])

## Warning in zoo(dataP[, 2], dataP[, 1]): some methods for "zoo" objects do
not
## work if the index entries in 'order.by' are not unique

# izvrednotimo mesečne vrednosti
mes1 <- as.yearmon(time(Pzoo)+3600)
head(mes1) # poglejmo strukturo podatkov

## [1] "jan. 2010" "jan. 2010" "jan. 2010" "jan. 2010" "jan. 2010" "jan. 2010
"

tail(mes1) # zadnji del podatkov

## [1] "dec. 2020" "dec. 2020" "dec. 2020" "dec. 2020" "dec. 2020" "dec. 2020
"

mesVsote <- aggregate(Pzoo, mes1, sum)
plot(mesVsote, xlab="Leto",ylab="Mesečne vsote padavin [mm]")
```



Slika 66: Mesečne vsote padavin.

```
# poglejmo osnovno statistiko
summary(mesVsote)[,2]

##
## "Min.   :  0.0  " "1st Qu.: 53.9  " "Median :101.8  " "Mean   :106.0  "
##
## "3rd Qu.:145.2  " "Max.   :419.9  "
```

Za izračun indeksa SPI lahko uporabimo paket *SPEI*. Izračun indeksa SPI s 3-mesečnim korakom (SPI-3) je prikazan spodaj, argument *scale* določa obdobje akumulacije, izbiramo pa lahko tudi med različnimi porazdelitvenimi funkcijami (argument *distribution*). Vrednosti indeksa SPI manjše od -2 označujejo ekstremno sušo, vrednosti med -1,5 in -2 hudo sušo, med -1 in -1,5 pa zmerno sušo, vrednosti med 1 in -1 označujemo kot normalno stanje; na podoben način so definirana tudi mokra obdobja (zelo namočeno, precej namočeno ter zmerno namočeno). Indeks SPI, izračunan za krajša obdobja, je lahko indikator sprememb v krajših časovnih intervalih (npr. manjša vlažnost tal, zmanjšanje vode v manjših vodotokih), SPI, izračunan za daljša trajanja, pa je lahko indikator zmanjšanja napajanja podtalnice. Poudariti je treba, da je smiselno v takšne analize vključiti daljša obdobja podatkov (npr. vsaj 30 let podatkov), v našem primeru pa bo indeks SPI izračunan zgolj z upoštevanjem 11 let. Paket *SPEI* vsebuje tudi funkcijo za izračun indeksa SPEI, kjer se upošteva bilanca (padavine–evapotranspiracija), paket pa vsebuje tudi funkcije za izračun potencialne in referenčne evapotranspiracije na podlagi različnih metod.

```
library(SPEI, quietly=TRUE)

## Warning: package 'SPEI' was built under R version 4.1.3

## # Package SPEI (1.8.1) loaded [try SPEINews()].

spiLJ3 <- spi(data=as.numeric(mesVsote), scale=3, distribution = "Gamma")

## [1] "Calculating the Standardized Precipitation Evapotranspiration Index (
SPEI) at a time scale of 3. Using kernel type 'rectangular', with 0 shift. Fi
```

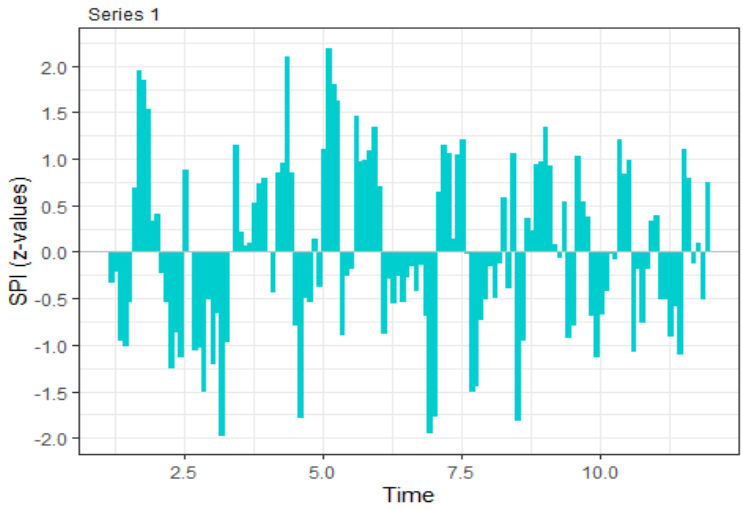
```
tting the data to a Gamma distribution. Using the ub-pwm parameter fitting me
thod. Checking for missing values (`NA`): all the data must be complete. Usin
g the whole time series as reference period. Input type is vector. No time in
formation provided, assuming a monthly time series."
```

```
# preverimo vrednosti indeksa SPI-3
spiLJ3
```

```
##   [1]           NA           NA -0.320274607 -0.194664298 -0.937433956
##   [6] -0.997291452 -0.532102209  0.694251435  1.956356663  1.848429557
##  [11]  1.534704427  0.330837079  0.400682870 -0.214182042 -0.527328861
##  [16] -1.244062574 -0.846891962 -1.118370879  0.887495674  0.004630253
##  [21] -1.043273171 -1.018107148 -1.484780898 -0.503925501 -1.198912813
##  [26] -0.649866162 -1.965812005 -0.964039477  0.013230781  1.153018972
##  [31]  0.221749097  0.061847109  0.093577654  0.530403106  0.740821193
##  [36]  0.796258538  0.009794913 -0.430663652  0.848850167  0.952027128
##  [41]  2.099860410  0.859576635 -0.783250374 -1.775407088 -0.484887868
##  [46] -0.534458983  0.141015585 -0.369850775  1.099274994  2.195683553
##  [51]  1.802138774  1.629224496 -0.882588862 -0.241162386 -0.171914258
##  [56]  1.458401392  0.967692769  0.993198025  1.087638631  1.347840830
##  [61]  0.708802357 -0.873485850 -0.277725207 -0.544605200 -0.241767869
##  [66] -0.533794532 -0.259785861 -0.134785235 -0.409735107 -0.128388531
##  [71] -0.675362384 -1.929543286 -1.753847072  0.645290230  1.155997095
##  [76]  1.054679110  0.137503564  1.046102783  1.210122917 -0.013273574
##  [81] -1.498376646 -1.432291985 -0.717582302 -0.498851772 -0.134448961
##  [86] -0.486447345 -0.111003443  0.582909529 -0.382092446  1.065977865
##  [91] -1.803891703 -0.938563641  0.368208342  0.236068571  0.936565757
##  [96]  0.976126447  1.345421186  0.922216408  0.077795125 -0.048752377
## [101]  0.535018388 -0.913392064 -0.783250374  1.030650745  0.548587794
## [106]  0.384780677 -0.668509843 -1.117839390 -0.668058412 -0.403802362
## [111] -0.013587585 -0.071196777  1.216234170  0.833255835  0.980794035
## [116] -1.060219411 -0.166013931 -0.750160220 -0.166819485  0.334108691
## [121]  0.392053300 -0.493317893 -0.498100846 -0.902303156 -0.569267491
## [126] -1.089178576  1.101313441  0.792667623 -0.115795328  0.094460418
## [131] -0.504844354  0.750705868
```

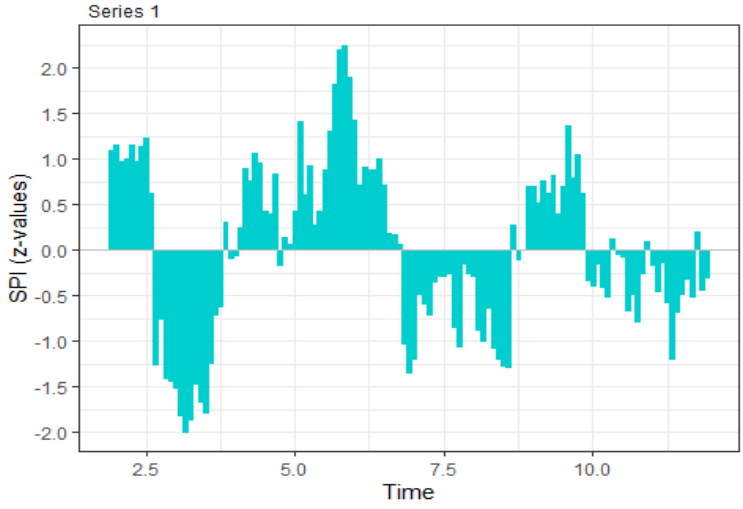
```
plot(spiLJ3)
```

```
## Warning: Removed 130 rows containing missing values (`geom_point()`).
```



Slika 67: Indeks SPI-3.

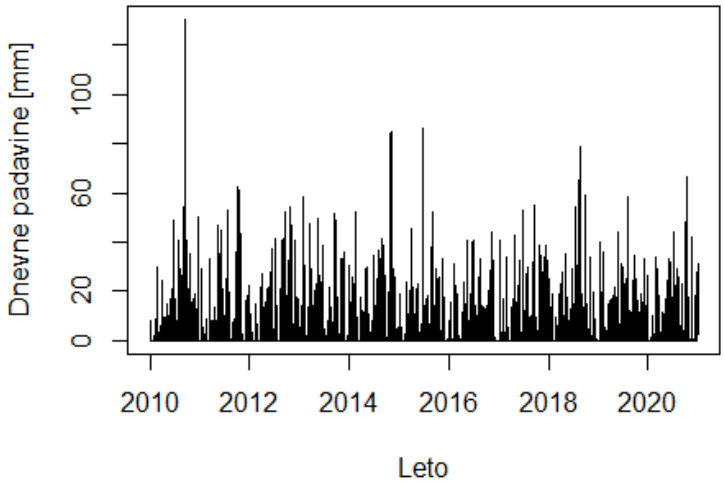
```
spiLJ12 <- spi(data=as.numeric(mesVsote), scale=12, distribution = "Gamma")

## [1] "Calculating the Standardized Precipitation Evapotranspiration Index (
SPEI) at a time scale of 12. Using kernel type 'rectangular', with 0 shift. F
itting the data to a Gamma distribution. Using the ub-pwm parameter fitting m
ethod. Checking for missing values (`NA`): all the data must be complete. Usi
ng the whole time series as reference period. Input type is vector. No time i
nformation provided, assuming a monthly time series."

plot(spiLJ12)

## Warning: Removed 121 rows containing missing values (`geom_point()`).
```



Slika 68: Indeks SPI-12.

Poleg indeksa SPI obstaja vrsta drugih indeksov[93]. Večina indeksov kot vhodni podatek uporablja mesečne padavine (ali vodno bilanco), le nekaj indeksov se lahko izračuna na podlagi dnevnih podatkov. Eden izmed takšnih je tudi efektivni indeks suše (angl. *Effective Drought Index*) (EDI), ki ga lahko izračunamo na podlagi dnevnih podatkov o padavinah[94]. Posledično je takšen indeks zelo uporaben za operativno spremljanje meteorološke in kmetijske suše[95]. EDI je bil uporabljen tudi za analize sočasnosti nastopa suše in ekstremne vročine v Evropi za obdobje 1961–2018[96].

```
# iz 30-min podatkov v dnevne vrednosti
dan1 <- as.Date(time(Pzoo)+3600,format="%Y-%m-%d")
# izračunamo dnevne vsote padavin
dnevneVsote <- aggregate(Pzoo, dan1, sum)
# preverimo osnovno statistiko dnevnih podatkov
summary(dnevneVsote)[,2]

##
## "Min.   :  0.000  " "1st Qu.:  0.000  " "Median :  0.000  " "Mean   :  3.4
81  "
##
## "3rd Qu.:  1.600  " "Max.   :130.500  "

plot(dnevneVsote, xlab="Leto",ylab="Dnevne padavine [mm]")
```



Slika 69: Dnevne vrednosti padavin.

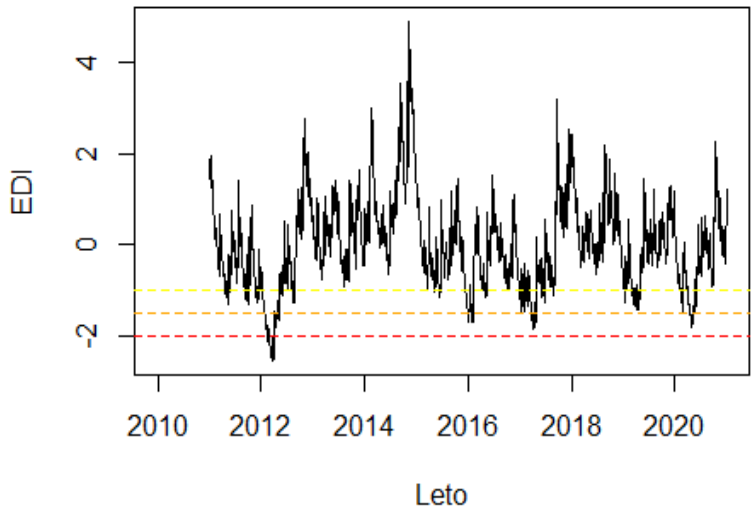
[93] https://www.preventionweb.net/files/1869_VL102136.pdf.

[94] https://www.droughtmanagement.info/effective-drought-index-edi/.

[95] https://doi.org/10.1175/2011JAMC2664.1.

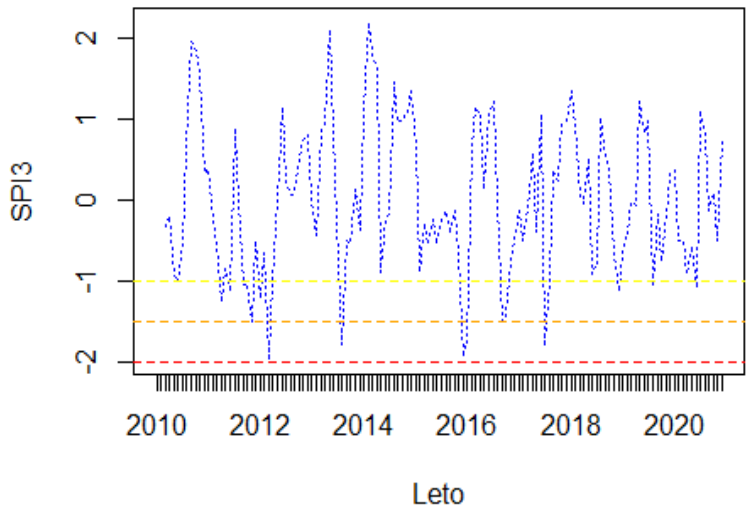
[96] https://doi.org/10.3390/w12123543.

```
# pred začetkom izračuna definiramo obdobje izračuna,
# torej EDI bomo izračunali na podlagi predhodnih 365 dni
# v objekt EP bomo shranjevali rezultate
obd <- 365;EP <- NA
vekt <- as.numeric(dnevneVsote)
# naredimo izračune vrednosti EP
for(i in (obd+1):length(as.numeric(dnevneVsote))){
dum <- 0
  for(j in 1:obd){
    dum <- dum + sum(vekt[((i-j+1):i)])/j
  }
  EP[i] <- dum
}
MEP <- mean(EP,na.rm=TRUE)
DEP <- EP-MEP
# izračun indeksa EDI
EDI <- DEP/sd(DEP,na.rm=TRUE)
susa <- which(EDI<(-1.5))
plot(time(dnevneVsote),EDI,type="l",xlab="Leto")
abline(h=-2,col="red",lty=2) # ekstremna suša
abline(h=-1.5,col="orange",lty=2) # huda suša
abline(h=-1,col="yellow",lty=2) # zmerna suša
```



Slika 70: EDI indeks.

```
plot(time(mesVsote),spiLJ3$fitted,type="l",col="blue",lty=3,xlab="Leto",
     ylab="SPI3")
abline(h=-2,col="red",lty=2); abline(h=-1.5,col="orange",lty=2)
abline(h=-1,col="yellow",lty=2)
```



Slika 71: SPI-3 indeks.

Iz primerjave rezultatov indeksa EDI z indeksom SPI-3 lahko vidimo ujemanje v določenih sušnih dogodkih (npr. leto 2012) ter slabše ujemanje za nekatere druge dogodke.

Naloga 37: S spletne strani ARSO (meteo.si, arhiv meritev) pridobite mesečne podatke o padavinah za obdobje 1961–2020 za postajo Murska Sobota ter izračunajte standardizirane padavinske indekse (SPI-1, SPI-3, SPI-6 in SPI-12) in jih prikažite tudi grafično.

4.10 Krivulje ITP

Na podlagi podatkov o padavinah je mogoče določiti tudi krivulje ITP, ki se uporabljajo kot eden od vhodnih podatkov pri določitvi projektnih padavinskih dogodkov[97]. V nadaljevanju je prikazan postopek določitve krivulj ITP na podlagi uporabljenih padavinskih podatkov. Treba je poudariti, da tudi pri določitvi krivulj ITP za izračun zanesljivih rezultatov potrebujemo čim daljše nize podatkov. Prikazani postopek sledi metodologiji, prikazani v paketu *IDF*[98], ki uporablja metodologijo, ki so jo razvili Koutsoyiannis in sodelavci[99]. Za določitev krivulj ITP se pogosto uporabljajo tudi druge porazdelitvene funkcije, kot je Gumelova porazdelitev[100]. V tem primeru je postopek izdelave krivulj ITP zelo podoben tistemu, ki smo ga prikazali pri izdelavi verjetnostnih analiz visokovodnih konic. V

---

[97] https://doi.org/10.1088/1748-9326/ab370a.

[98] https://cran.r-project.org/web/packages/IDF/index.html.

[99] https://doi.org/10.1016/S0022-1694(98)00097-3.

[100] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=27934&lang=slv.

nadaljevanju bomo za izbrana trajanja (v urah; argument *ds*) izračunali vzorec letnih maksimumov (v mm/h). Možna je tudi uporaba argumenta *which.mon*, s katerim lahko izberemo samo določene mesece, v katerih iščemo letne maksimume. Letne maksimume bi lahko določili tudi z uporabo paketa *zoo* in funkcije *aggregate*.

```
library(IDF, quietly=TRUE)
```

```
## Warning: package 'IDF' was built under R version 4.1.3
```

```
# preimenujemo vzorec glede na navodila paketa IDF
colnames(dataP) <- c("date","RR")
# definiramo vzorec
vzorec <- IDF.agg(data=list(dataP),ds=c(1,6,12,24),na.accept = 1)
# preverimo strukturo podatkov
head(vzorec)
```

```
##      xdat ds year  mon station
## 2010 23.9  1 2010 0:11       1
## 2011 24.2  1 2011 0:11       1
## 2012 27.4  1 2012 0:11       1
## 2013 23.2  1 2013 0:11       1
## 2014 40.3  1 2014 0:11       1
## 2015 27.5  1 2015 0:11       1
```

```
# xdat so letni maksimumi (mm/h) za različna trajanja ds (h)
# station označuje postajo
aggregate(zoo(rollapply(dataP[,2],12,sum),dataP[,1]),
floor(as.yearmon(time(zoo(rollapply(dataP[,2],2,sum),
dataP[,1]+3600)))), max)
```

```
## Warning in zoo(rollapply(dataP[, 2], 12, sum), dataP[, 1]): some methods f
or
## "zoo" objects do not work if the index entries in 'order.by' are not uniqu
e
```

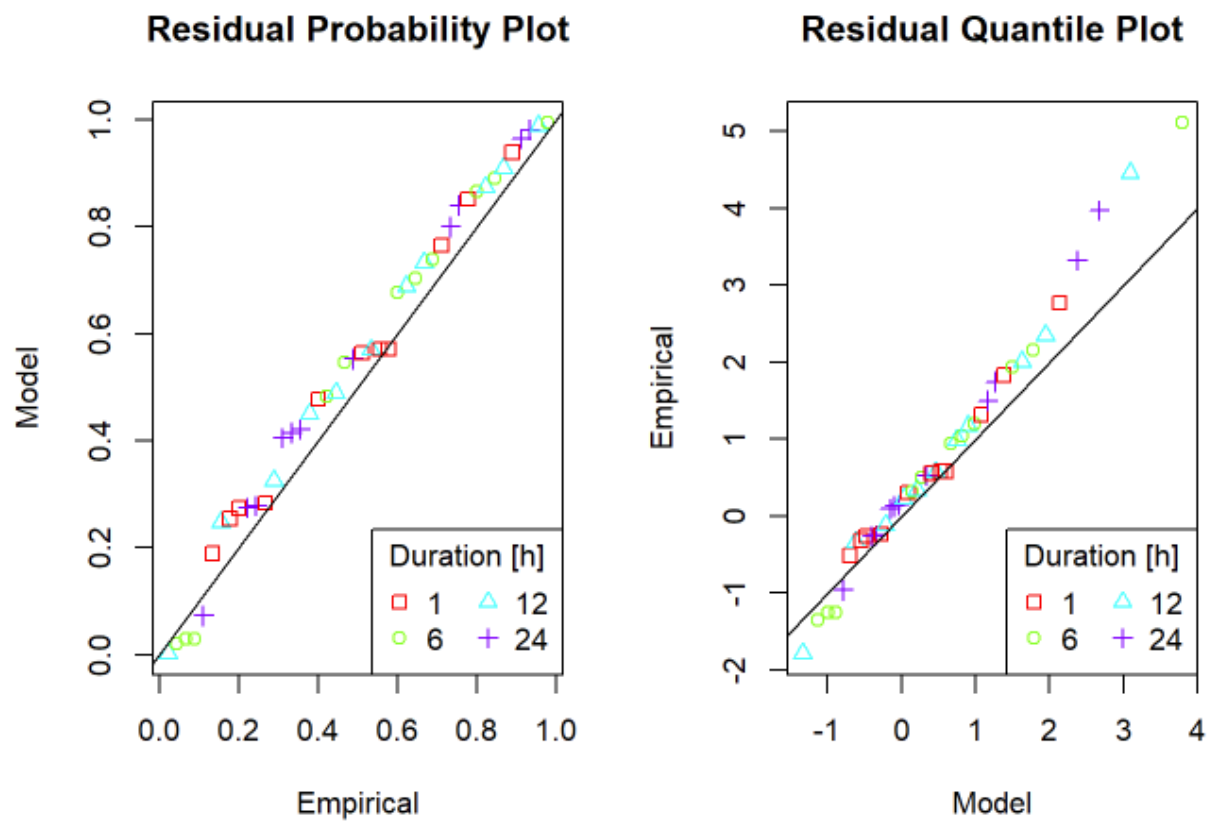
```
## Warning in zoo(rollapply(dataP[, 2], 2, sum), dataP[, 1] + 3600): some met
hods
## for "zoo" objects do not work if the index entries in 'order.by' are not u
nique
```

```
## jan. 2010 jan. 2011 jan. 2012 jan. 2013 jan. 2014 jan. 2015 jan. 2016 jan.
2017
##      61.5      54.1      52.7      37.0     112.9      46.8      36.5
48.2
## jan. 2018 jan. 2019 jan. 2020
##      63.9      37.0      51.8
```

```
# v tem primeru se izračuna 6-h vrednosti,
# drugi argument v funkciji rollapply definira, koliko podatkov seštevamo
# maksimalne vrednosti v mm (ne v mm/h) dobimo kot rezultat
# prilagodimo porazdelitev GEV z uporabo metode največjega verjetja (MLE)
test <- gev.d.fit(xdat=vzorec$xdat,ds=vzorec$ds)
```

```
## Warning in sqrt(diag(z$cov)): NaNs produced

## $conv
## [1] 0
##
## $nllh
## [1] 90.75036
##
## $mle
## [1] 6.43018911064108 3.89991482372726 0.23503591461929 0.00000002183837
## [5] 0.67972665428536
##
## $se
## [1] 0.17277214        NaN 0.10158694        NaN 0.02330413
```

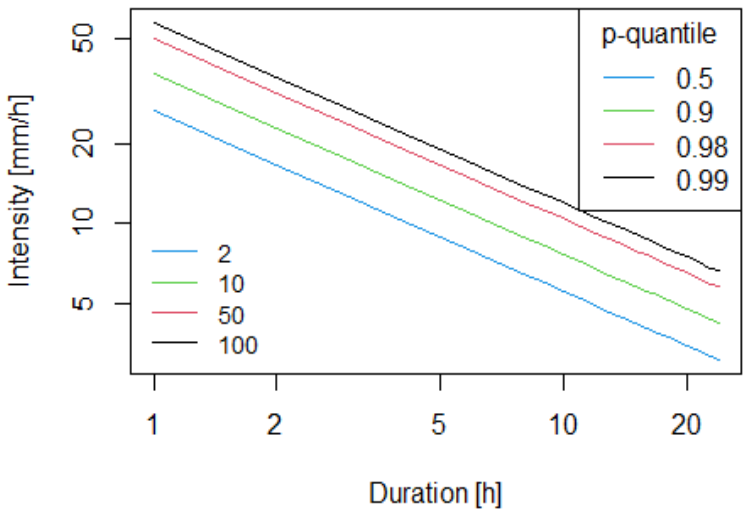
```
# preverimo grafično ujemanje med vzorcem in ocenjenimi vrednostmi
gev.d.diag(test)
```



Slika 72: Ujemanje GEV porazdelitve in vzorca vhodnih podatkov.

```
# izračunamo parametre porazdelitve GEV
parametri <- gev.d.params(test, ydat = NULL)
# določimo, za katere vrednosti povratnih dob bomo določili krivulje IDF
povratne <- c(2,10,50,100)
# izrišemo krivulje IDF
```

```
IDF.plot(durations=1:24,fitparams = parametri,probs = 1-1/povratne,cols=4:1)
# dodamo še legendo s prikazom povratne dobe v letih
legend("bottomleft",legend=povratne,col=4:1,lty=1,bty="n",cex=0.9)
```



Slika 73: ITP krivulje.

```
qgev.d(p=1-1/povratne,mut=parametri[1],sigma0=parametri[2],xi=parametri[3],
theta=parametri[4],eta=parametri[5],eta2=parametri[6],tau=parametri[7],d=24)

## [1] 3.063544 4.225090 5.765042 6.618706
```

V zadnjem koraku smo izračunali tudi intenzitete padavin (v mm/h) za izbrana trajanja, ki so podana z argumentom *d*. Argument *p* podaja izbrane vrednosti povratnih dob, za katere nas zanimajo rezultati, ostali argumenti pa definirajo parametre funkcije *qgev.d*.

4.11 Erozivnost padavin

Erozivnost padavin je med glavnimi dejavniki erozije tal[101]. Za erozivnost padavin je značilna velika prostorska in časovna spremenljivost[102]. Za pridobitev zanesljivih ocen o erozivnosti padavin so potrebni podatki o padavinah z visoko časovno ločljivostjo, na primer podatki s časovnim korakom 5 minut. Razpoložljivost postaj s podatki z visoko časovno frekvenco, ki se lahko uporabijo za izračun erozivnosti, je v mnogih delih sveta razmeroma majhna. Erozivnost padavin R je eden izmed vhodnih podatkov v enačbo RUSLE, ki se uporablja za oceno potencialnega sproščanja (erozije) tal zaradi delovanja vode[103]. Metodologija RUSLE tako določa postopek izračuna erozivnosti padavin R in kriterije za določitev erozivnih

[101] https://doi.org/10.1016/j.catena.2021.105577.

[102] https://doi.org/10.1016/j.scitotenv.2015.01.008.

[103] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=1512&lang=eng.

padavinskih dogodkov[104]. Glede na metodologijo sta dva neodvisna padavinska dogodka ločena v primeru manj kot 1,27 mm dežja v 6 urah (med dvema dogodkoma). V izračunih erozivnosti padavin se upoštevajo le erozijski padavinski dogodki z več kot 12,7 mm dežja v celotnem dogodku ali v primeru, da več kot 6,35 mm dežja pade v 15 minutah. Erozivnost padavin R se izračuna kot zmnožek maksimalne 30-minutne intenzitete padavin in kinetične energije padavinskega dogodka[105]. Letna erozivnost je seštevek erozivnosti za vse erozivne dogodke v določenem letu. Za izračun specifične kinetične energije padavin so bile razvite številne enačbe na podlagi meritev o lastnostih dežnih kapljic (hitrost in velikost)[106]. Spodaj prikazana metodologija (in R-koda) za izračun erozivnosti padavin je povzeta po prispevku, ki so ga pripravili Pidoto in sodelavci (2022)[107]. Funkcija za izračun erozivnosti omogoča tudi spremembo kriterijev za določitev neodvisnih padavinskih dogodkov, saj se v nekaterih primerih kot kriterij upošteva samo obdobje 6 h (ali 4 h) brez padavin. Funkcija za izračun erozivnosti padavin za neodvisne dogodke je zapisana v nadaljevanju. Argument *Pcp* so zvezni vhodni podatki o padavinah (v obliki vektorja), argument *Factor* določa, ali so enote podatkov mm (ali kakšne druge enote; vrednost 1 pomeni, da so enote v mm), argument *StartDate* določa začetni datum podatkov, argument *Timestep* določa časovni korak podatkov (v minutah), argument *SeparationMin* določa kriterij, kdaj sta dva dogodka združena (v mm; se lahko uporabi recimo vrednost 0), argument *PcpMin* določa kriterij, kateri dogodek se še upošteva kot erozivni (v mm), prav tako tudi argument *Pcp15Min* (v mm), le da gre v tem primeru za 15-minutno količino padavin.

```
GetErosiveEvents <- function(Pcp, Factor=1, StartDate=as.POSIXct("1900-01-01"
,
tz="UTC"),Timestep=5, SeparationMin=1.27, PcpMin=12.7, Pcp15Min=6.35 ) {
  print(paste("Factor:",Factor)) # izpis osnovnih značilnosti funkcije
  print(paste("Začetni datum:",StartDate))
  print(paste("Vsota padavin:",round(sum(Pcp/Factor, na.rm = TRUE)),"[mm]"))
  # morebitne negativne vrednosti se zamenjajo z 0
  Pcp[Pcp<0] <- as.integer(0)
  Pcp[is.nan(Pcp)] <- as.integer(0) # enako tudi NaN
  Pcp[is.na(Pcp)] <- as.integer(0) # in tudi manjkajoče vrednosti
  IsWet <- (Pcp>0) # ali so bile v določenem koraku padavine ali niso bile
  # funkcija rolling sum se uporabi za ločitev dveh neodvisnih dogodkov
  # k po 6 h = 1 + 6*60/Timestep
  PcpRollSum <- rollsum(Pcp, k = 1 + 6*60/Timestep, align = "left", fill = 0)
- Pcp
  if(Timestep<30) { # funkcija je bila v osnovi pripravljena za podatke
  # s časovnim korakom 5 min in kasneje prilagojena še za trajanje 30 min
  # v tem primeru je nemogoče izračunati 15-min intenziteto
```

[104] https://www.ars.usda.gov/arsuserfiles/64080530/rusle/ah_703.pdf.

[105] https://doi.org/10.1016/j.jhydrol.2022.128478.

[106] https://actahydrotechnica.fgg.uni-lj.si/si/paper/a29ac.

[107] https://doi.org/10.5194/esurf-10-851-2022.

```
  PcpRollI15 <- rollsum(Pcp, k = 15/Timestep, align = "center", fill = 0 )
  } else if (Timestep==30) {
  PcpRollI15 <- rollsum(Pcp, k = 30/Timestep, align = "center", fill = 0 )
  } else {
  return(print("Trajanje mora biti krajše ali enako 30 min"))
    }
  # podaljšanje mokrih obdobij med dogodki, če merila niso izpolnjena
  IsWet <- ( IsWet | (PcpRollSum>=SeparationMin*Factor)  )
  # razčlenitev dogodkov in kontrola, kateri dogodki ustrezajo kriterijem
  calcEventIMax <- function(a,b,Pcp,Dur=15) return( max(rollsum(x = Pcp[a:b],
k = min(b-a+1,Dur/Timestep) )) )
  # izračun največje intenzitete [mm/uro] za določeno trajanje [min]
  calcEventI15Max <- function(a,b,PcpMax) return( max(PcpMax[a:b]) )
  # izračun maksimalne intenzitete za trajanje 15 min
  calcEventVol <- function(a,b,Pcp) return(sum(Pcp[a:b]))
  # izračun količine padavin za dogodek
  # izračun dolžine in vrednosti serij enakih vrednosti v vektorju
  Events <- rle(IsWet)
  # pretvorba v podatkovni okvir
  Events <- data.frame(Length=Events[[1]],IsWet=Events[[2]])
  Events$endIndex <- cumsum(Events$Length) # konec dogodka
  # začetek dogodka
  Events$startIndex <- as.integer(Events$endIndex - Events$Length + 1)
  Events <- Events[Events$IsWet,] # izbira samo erozivnih dogodkov
  Events$Vol <- unlist(mapply(calcEventVol, Events$startIndex, Events$endInde
x, MoreArgs=list(Pcp=Pcp)))
  Events$I15 <- unlist(mapply(calcEventI15Max, Events$startIndex, Events$endI
ndex, MoreArgs=list(PcpMax=PcpRollI15)))
  # odstranitev dogodkov, ki ne ustrezajo kriterijem
  Events <- Events[ (Events$Vol>PcpMin*Factor | Events$I15>Pcp15Min*Factor )
, ]
  # izračun max 30-minutne intenzitete
  Events$I30 <- unlist(mapply(calcEventIMax, Events$startIndex, Events$endInd
ex, MoreArgs=list(Pcp=Pcp,Dur=30)))
  Events$I30 <- (Events$I30/Factor) / (30/60) # izračun intenzitete [mm/h]
  # izračun dejanskega trajanja padavinskih dogodkov
  CalcEventDur <- function(a,b,Pcp) return( max(which(Pcp[a:b]!=0)) - min(whi
ch(Pcp[a:b]!=0)) + 1 )
  CalcEventDate <- function(a,b,Pcp) return( min(which(Pcp[a:b]!=0))+a-2  )
  # izračun specifične kinetične energije glede na enačbo
  # Brown and Foster (1987). i: intenziteta padavin [mm/hr]
  CalcBrownFoster <- function(i) return( 0.29*(1-0.72*exp(-0.05*i)) )
  # izračun kinetične energije za izbrane dogodke
  CalcEventE <- function(a,b,Pcp) return( sum(CalcBrownFoster((Pcp[a:b]/Facto
r)/(Timestep/60)) * (Pcp[a:b]/Factor) ) )
  Events$E <- unlist(mapply(CalcEventE, Events$startIndex, Events$endIndex, M
oreArgs=list(Pcp=Pcp)))
  Events$R <- Events$E * Events$I30
  Events$Date <- StartDate + (unlist(mapply(CalcEventDate, Events$startIndex,
Events$endIndex, MoreArgs=list(Pcp=Pcp)))*Timestep*60)
```

```
  Events$D_h <- unlist(mapply(CalcEventDur, Events$startIndex, Events$endInde
x, MoreArgs=list(Pcp=Pcp))) * Timestep/60
  # intenziteta dogodka [mm/hr]
  Events$I_mmh <- (Events$Vol/Factor) / Events$D_h
  Events$P_mm <- (Events$Vol/Factor) # preimenujemo stolpec
  # odstranimo podatke, ki jih ne potrebujemo
  Events[,c("IsWet","Length","Vol","I15","startIndex")] <- NULL
  # spremenimo vrstni red
  Events <- Events[,c("Date","E","I30","R","P_mm","D_h","I_mmh","endIndex")]
  return(Events) # izpišemo rezultate
}
```

Uporabimo zgoraj definirano funkcijo in določimo erozivne dogodke glede na izbrane kriterije in z uporabo funkcije za izračun energije, ki sta jo predlaga Brown in Foster (1987), izračunamo tudi erozivnost padavin. V nadaljevanju bomo izrisali tudi graf vseh erozivnih dogodkov. Ker so podatki o erozivnosti precej asimetrični (nekaj velikih dogodkov, večina dogodkov ima manjšo erozivnost), lahko izračunano tudi različne koeficiente neenakomernosti, kot je to prikazano v članku Bezak in sodelavci[108]. Izrisali bomo Lorenzevo krivuljo skupaj izračunanih Ginijevih koeficientov (G; G* je normalizirani Ginijev koeficient). Razpon Ginijevega koeficienta je med 0 in 1, kjer vrednost blizu 0 pomeni, da imajo vsi erozivni dogodki približno enako erozivnost, vrednost blizu 1 pa, da so med erozivnimi dogodki velike razlike. Dodatno bomo izračunali tudi delež dogodkov, ki prispevajo 50 % vse erozivnosti.

```
dogodki <- GetErosiveEvents(dataP[,2],Factor=1,StartDate=as.POSIXct("2010-01-
01", tz="CET"),Timestep = 30, SeparationMin=1.27, PcpMin=12.7, Pcp15Min=6.35)
```

```
## [1] "Factor: 1"
## [1] "Začetni datum: 2010-01-01"
## [1] "Vsota padavin: 13987 [mm]"
```

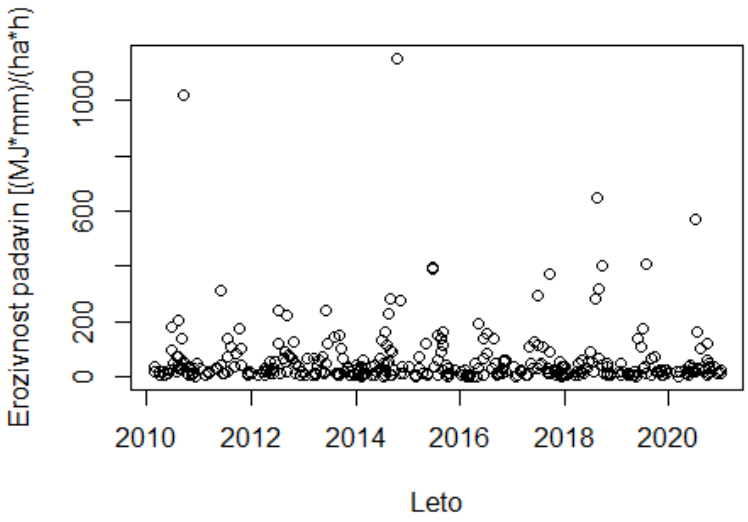
```
head(dogodki) # rezultati
```

```
##                   Date        E  I30         R P_mm  D_h     I_mmh endInd
ex
## 18  2010-02-19 11:30:00 5.900608  6.0 35.403649 55.0 28.5 1.9298246     24
32
## 28  2010-02-26 03:00:00 3.304734  5.8 19.167460 29.5 19.0 1.5526316     27
32
## 58  2010-03-31 07:30:00 2.743929  6.0 16.463575 24.1 15.5 1.5548387     43
16
## 62  2010-04-01 15:30:00 1.547226  4.0  6.188902 14.8 14.5 1.0206897     43
78
## 116 2010-04-26 21:00:00 1.529229 13.4 20.491667  9.2  2.0 4.6000000     55
64
```

[108] https://doi.org/10.1016/j.catena.2021.105577.

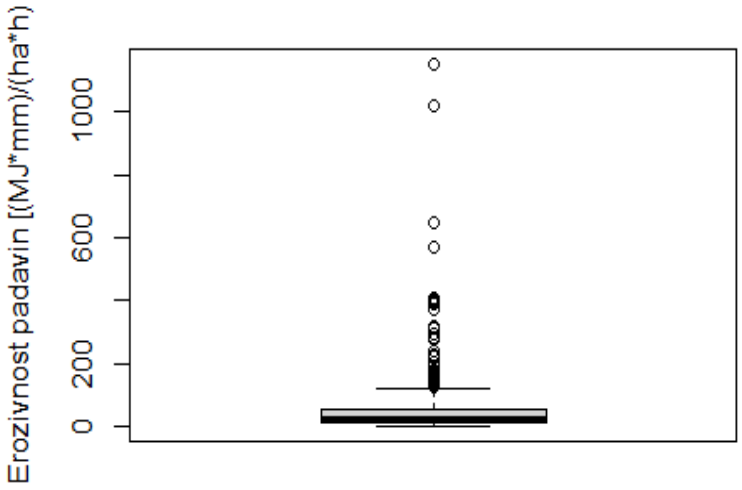
```
## 118 2010-05-03 01:30:00 1.405858  3.2  4.498746 14.8 18.0 0.8222222     58
93
```

```
plot(dogodki[,1],dogodki[,4],xlab="Leto",
  ylab="Erozivnost padavin [(MJ*mm)/(ha*h)")
```



Slika 74: Vsi erozivni padavinski dogodki.

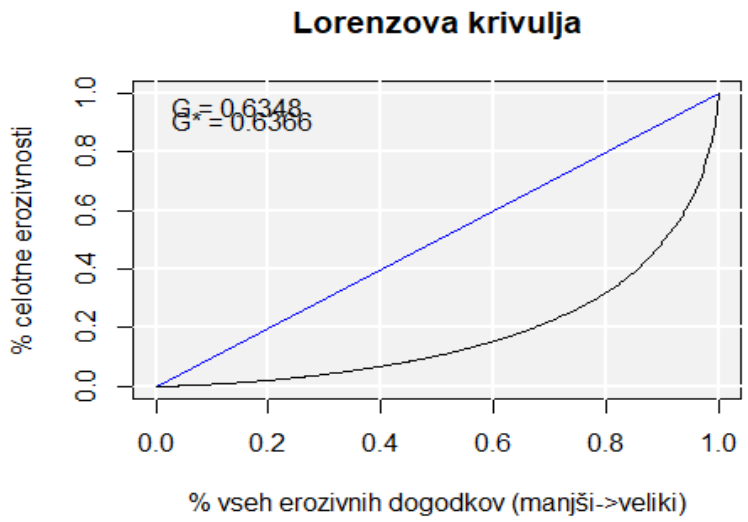
```
boxplot(dogodki[,4], ylab="Erozivnost padavin [(MJ*mm)/(ha*h)")
```



Slika 75: Okvir z ročaji vseh erozivnih padavinskih dogodkov.

```
library(ineq, quietly=TRUE)
library(REAT, quietly=TRUE)

##
## Attaching package: 'REAT'
```

```
## The following object is masked from 'package:ineq':
##
##     conc

## The following object is masked from 'package:readr':
##
##     spec

# Lorenzeva krivulja
lorenz(dogodki$R, lcg = TRUE, lcgn = TRUE,
lcx = "% vseh erozivnih dogodkov (manjši->veliki)",
lcy = "% celotne erozivnosti", lctitle = "Lorenzova krivulja")
```



Slika 76: Lorenzeva krivulja na podlagi erozivnih dogodkov.

```
# Ginijev koeficient
gini(dogodki$R)

## [1] 0.634825

# izračun vsote erozivnosti
vsota <- sum(sort(dogodki$R))
d50 <- which.min(abs(cumsum(sort(dogodki$R)) - vsota*0.5))/length(dogodki$R)
d50 # vidimo, da približno 10 % erozivnih dogodkov (tistih največjih)

## [1] 0.9030471

# prispeva 50 % celotne erozivnosti
vsota/11 # izračunamo pa lahko tudi povprečno letno erozivnost padavin R

## [1] 1931.834

# ki je eden izmed vhodnih podatkov v enačbi RUSLE
# (11 je število let podatkov, ki jih obravnavamo, obdobje 2010–2020)
```

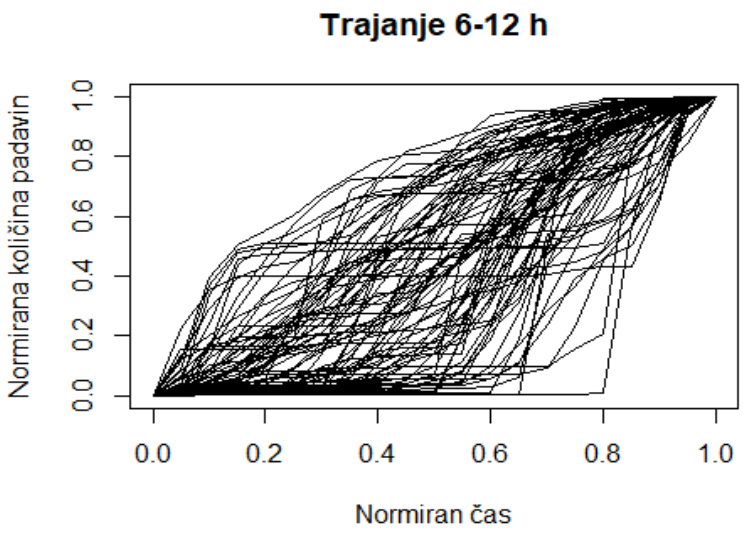
4.12 Huffove krivulje

Podatke o (neodvisnih) padavinskih dogodkih lahko uporabimo tudi za določitev Huffovih krivulj. Huffove krivulje so brezdimenzijske normirane krivulje padavin, ki opisujejo lastnosti padavinskega dogodka (v katerem delu padavinskega dogodka lahko pričakujemo večjo količino padavin)[109]. Huffove krivulje lahko uporabimo kot enega izmed vhodnih podatkov pri določitvi sintetičnih padavinskih dogodkov[110]. Huffove krivulje bomo v naslednjem primeru izdelali na podlagi dogodkov, ki imajo trajanja krajša od 12 h in daljša od 6 h.

```
# izberemo dogodke
izbira <- which(dogodki$D_h<12 & dogodki$D_h>6)
# generiramo zaporedje normiranih vrednosti
zaporedje <- seq(from=0,to=1,by=0.05)
# pripravimo prazno matriko, kamor bomo shranjevali rezultate
# za vsak dogodek bomo določili vrednosti Huffovih krivulj
huffvmes <- matrix(NA,length(izbira),length(zaporedje))
# izračune bomo naredili v obliki zanke za vse dogodke
for(i in 1:length(izbira)){
# padavine glede na začetek in konec dogodka
padizb <- dataP[(dogodki$endIndex[izbira[i]]-2*dogodki$D_h[izbira[i]]):dogodk
i$endIndex[izbira[i]],2]
# vsota padavin med dogodkom
vsotapad <- sum(padizb)
for(k in 0:(length(zaporedje))){
# linearna interpolacija za izbrane časovne korake
# glede na normirano količino padavin (med dvema točkama,
# za izbrano točko, ki je definirana z argumentom xout)
huffvmes[i,k] <- approx(seq(0,1,length=length(cumsum(padizb))),
  cumsum(padizb)/vsotapad,xout=zaporedje[k])$y
}
}
huffvmes[,1] <- 0 # vse začetne vrednosti morajo biti enake 0
huffvmes[,21] <- 1 # vse končne vrednosti pa enake 1
# izrišemo graf vseh normiranih krivulj za vse dogodke
plot(zaporedje,huffvmes[1,],type="l",xlab="Normiran čas",
  ylab="Normirana količina padavin",main="Trajanje 6-12 h")
# na graf dodamo še vse Huffove krivulje za dogodke
for(g in 2:dim(huffvmes)[1]){
lines(zaporedje,huffvmes[g,])
}
```

[109] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=104644&lang=eng.

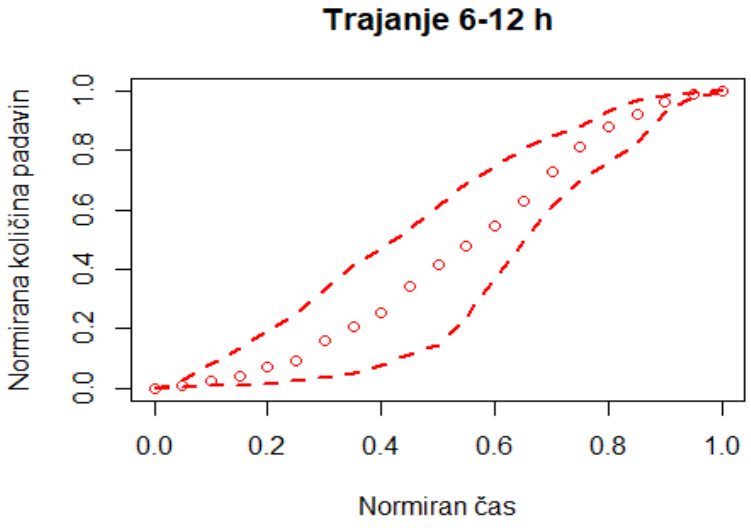
[110] https://www.dlib.si/details/URN:NBN:SI:doc-7ETK7J2S.



Slika 77: Huffove krivulje za vse izbrane padavinske dogodke.

Na podlagi Huffovih krivulj lahko določimo tudi intervale zaupanja, npr. določimo lahko 50-odstotne empirične intervale zaupanja. Mediana, ki jo lahko obravnavano kot najbolj verjetno porazdelitev padavin za izbrane padavinske dogodke (v trajanju med 6 in 12 h), pa predstavlja rezultat analize porazdelitev padavin med dogodki.

```
spmeja <- apply(huffvmes,2,quantile,probs=0.25) # spodnja meja
med <- apply(huffvmes,2,quantile,probs=0.5) # mediana
zgmeja <- apply(huffvmes,2,quantile,probs=0.75) # zgornja meja
plot(zaporedje,med,col="red",lty=1,lwd=1,xlab="Normiran čas",
  ylab="Normirana količina padavin",main="Trajanje 6-12 h")
lines(zaporedje,spmeja,col="red",lty=2,lwd=2) # spodnja meja
lines(zaporedje,zgmeja,col="red",lty=2,lwd=2) # zgornja meja
```



Slika 78: Huffova krivulja (mediana) in 50 % intervali zaupanja.

Naloga 38: Izdelajte Huffove krivulje za padavinske dogodke s trajanjem med 12 in 24 h ter določite tako mediano Huffovih krivulj kot intervale zaupanja (10 % in 90 % percentilni krivulji).

4.13 Stohastični simulator padavin (in temperature zraka)

Stohastični generatorji vremenskih spremenljivk, kot so padavine in temperatura zraka, imenovani generatorji vremena, so bili v zadnjih desetletjih razviti za različne namene, na primer hidrološke in agronomske aplikacije[111]. Takšni generatorji vremena so bili uporabljeni v številnih praktičnih primerih, povezanih s fenologijo rastlin in kmetijstvom[112] ter hidro-klimatskimi študijami[113]. Z generatorji vremenskih spremenljivk lahko reproduciramo oziroma generiramo časovne vrste izbranih spremenljivk, kot so padavine ali temperatura zraka. V nekaterih primerih je treba ohraniti tudi prostorske značilnosti padavin (in temperature zraka). V tem primeru je treba zagotoviti skladnost vremenskega generatorja za različne lokacije v prostoru[114]. V sklopu programskega okolja R so na voljo številni paketi za simulacije padavin in temperature zraka. Pokažimo postopek uporabe paketa *GWEX*[115]. Najprej bomo uvozili podatke s treh padavinskih postaj za obdobje od leta 2015 do 2022 na območju Slovenije. Podatke bomo preoblikovali v matriko, kot to zahteva paket *GWEX*, in ocenili parametre modela. Pri tem je smiselno argument *nChainFit* povečati (glede na primer spodaj), s čimer se sicer podaljša čas izračuna. Vrednost argumenta *th* določa količino padavin, ki se uporabi za določitev t. i. mokrih dni. S funkcijo *simGwexModel* lahko simuliramo padavine za določeno število dni v prihodnosti; v našem primeru smo za vsako lokacijo simulirali dve realizaciji podatkov (argument *nb.rep*). Več informacij o dodatnih parametrih je na voljo v opisu funkcije *fitGwexModel*.

```
postaje3 <- read.table("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/Padavine.txt",header=TRUE)
library(GWEX, quietly=TRUE)
skupaj <- as.matrix(postaje3[,2:4]) # preoblikujemo podatke
# definiramo vhodne podatke glede na zahteve paketa GWEX
myObsPrec <- GwexObs(variable='Prec',date=as.Date(postaje3[,1],
  format="%d.%m.%Y"),obs=skupaj)
# ocenimo parametre
myparPrec <- fitGwexModel(myObsPrec, listOption=list(th=0.5,nChainFit=1000))
```

[111] https://doi.org/10.1016/j.jhydrol.2015.12.036.

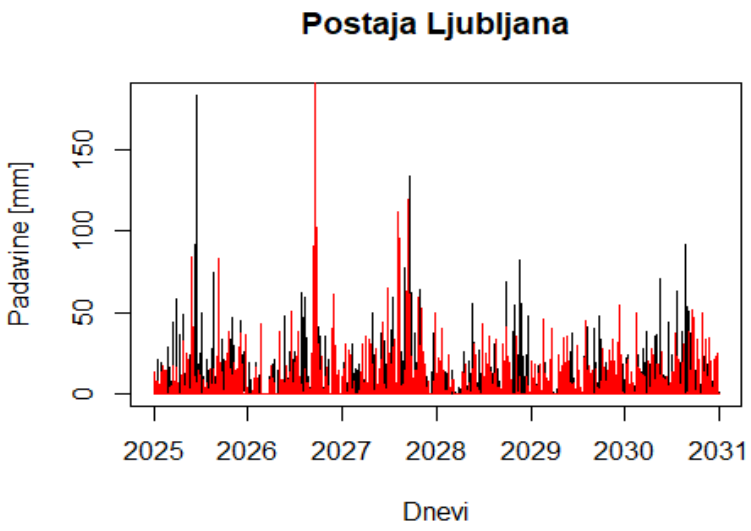
[112] https://doi.org/10.1371/journal.pone.0161620.

[113] https://doi.org/10.1016/j.jhydrol.2019.124443.

[114] https://doi.org/10.5194/hess-13-2299-2009.

[115] https://cran.r-project.org/web/packages/GWEX/index.html.

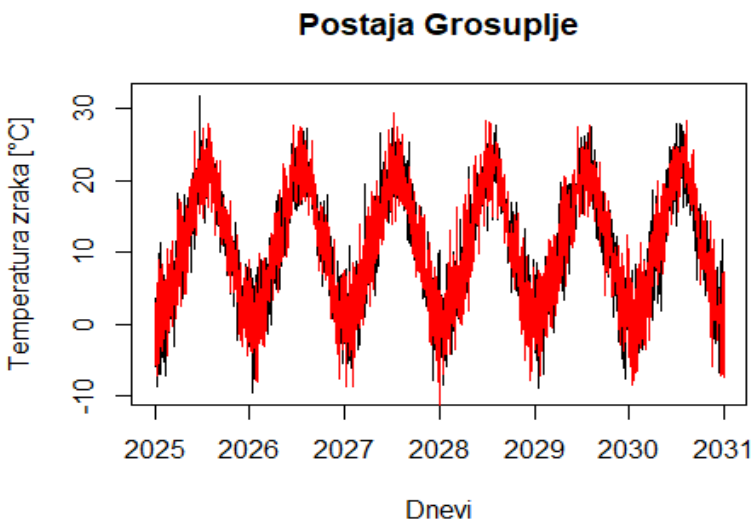
```
## [1] "Fit generator"

# simuliramo padavine
rez <- simGwexModel(myparPrec,nb.rep=2,d.start = as.Date("01012025","%d%m%Y")
,
    d.end = as.Date("31122030","%d%m%Y"))

## [1] "Generate scenarios"

str(rez@sim) # pogledamo strukturo simuliranih podatkov

##  num [1:2191, 1:3, 1:2] 0 3.42 13.51 0 0 ...

# preverimo osnovne statistike simuliranih podatkov
# v tem primeru realizacija številka 1 za postajo 1 (Topol)
summary(rez@sim[,1,1])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   4.722   2.817 199.176

# realizacija številka 2 za postajo 1 (Topol)
summary(rez@sim[,1,2])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   4.913   2.565 251.864

# realizacija številka 2 za postajo 3 (Grosuplje)
summary(rez@sim[,3,2])

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.000   0.000   3.635   1.786 140.426

# izrišemo realizacijo številka 2 za postajo Ljubljana
plot(rez@date,rez@sim[,2,][,2],type="l",xlab="Dnevi",ylab="Padavine [mm]",
    main="Postaja Ljubljana")
# na isti graf dodamo še realizacijo številko 1 za isto postajo
lines(rez@date,rez@sim[,2,][,1],col="red")
```



Slika 79: Simulirane padavine za postajo Ljubljana.

Na enak način lahko definiramo tudi model za podatke o temperaturi zraka, kjer najprej preberemo podatke (iste postaje kot v primeru padavin), jih preoblikujemo, ocenimo parametre in nato uporabimo stohastični model za generiranje podatkov.

```
postaje3T <- read.table("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbe
nik/Temperatura.txt",header=TRUE)
skupajT <- as.matrix(postaje3T[,2:4])
myObsTemp <- GwexObs(variable='Temp',date=as.Date(postaje3T[,1],
  format="%d.%m.%Y"),obs=skupajT)
myparTemp <- fitGwexModel(myObsTemp, listOption=list(hasTrend=TRUE,
  typeMargin='Gaussian',depStation='Gaussian',nChainFit=1000))

## [1] "Fit generator"
## ================================================================================
======

# generiramo še temperaturo zraka
rezT <- simGwexModel(myparTemp,nb.rep=2,d.start = as.Date("01012025",
  "%d%m%Y"),d.end = as.Date("31122030","%d%m%Y"))

## [1] "Generate scenarios"

# izrišemo realizacijo številka 2 za postajo Grosuplje
plot(rezT@date,rezT@sim[,3,][,2],type="l",xlab="Dnevi",
  ylab="Temperatura zraka [°C]",main="Postaja Grosuplje")
# na isti graf dodamo še realizacijo številko 1 za isto postajo
lines(rezT@date,rezT@sim[,3,][,1],col="red")
```



Slika 80: Simulirana temperatura zraka za postajo Grosuplje.

```
# delež mokrih dni
wet.day.frequency(skupaj,0.5)

##     Topol Ljubljana Grosuplje
## 0.3186174 0.2997947 0.3042437
```

Vidimo, da je v primerjavi s padavinami, ki so stohastično generirane, v primeru temperature zraka očiten sezonski vzorec spreminjanja te spremenljivke in da je tudi med dvema realizacijama manjša razlika med generiranimi podatki o temperaturi zraka kot v primeru padavin. Paket *GWEX* vključuje tudi številne druge uporabne funkcije, kot je recimo funkcija za izračun deleža mokrih dni glede na izbrano mejno vrednost (*wet.day.frequency*).

V sklopu programskega orodja R so na voljo tudi drugi paketi, ki omogočajo stohastične simulacije padavin (in tudi drugih spremenljivk). Omeniti velja predvsem paket *RMAWGEN*[116]. Tako paket *GWEX* kot tudi paket *RMAWGEN* se osredotočata na dnevne podatke o padavinah. Če želimo izvesti simulacije z urnim ali 30-minutnim časovnim korakom, lahko uporabimo paket *HyetosMinute*[117], ki sicer ni vključen v repozitorij CRAN, a so datoteke za namestitev paketa prostodostopne[118].

Naloga 39: S spletne strani meteo.si prenesite podatke o dnevnih padavinah za pet postaj na območju vzhodne Slovenije (za obdobje 2020–2023), definirajte stohastični

---

[116] https://cran.r-project.org/web/packages/RGENERATEPREC/vignettes/precipitation_stochastic_generation_v8.html.

[117] https://uwmh.eu/products/81-hyetosminute.html.

[118] https://www.itia.ntua.gr/el/softinfo/3/.

padavinski model in generirajte 20 realizacij padavinskih podatkov v trajanju treh let za vsako postajo.

4.14 Regresijska drevesa in gručenje

Programsko okolje R vsebuje tudi veliko paketov s področja rudarjenja po podatkih[119]. Velja omeniti pakete, kot so *data.table*, *dplyr*, *caret*, *e1071*, *xgboost*, *randomForest*[120]. Poglejmo najprej primer uporabe metode hierarhičnega gručenja in algoritma K-means. Metoda hierarhičnega gručenja omogoča določitev in vizualizacijo hierarhij v podatkih preko dendrograma[121]. Podobno velja tudi za algoritem K-means[122]. Za uporabo metod hierarhičnega gručenja bomo uporabili paket *dplyr*. Podatke bomo najprej normalizirali z uporabo funkcije *normalize* v paketu *BBmisc*. Sledil bo izračun razdalj med podatki; v tem primeru bomo analize naredili kar na podatkih *mtcars*, ki so že vgrajeni v osnovno programsko okolje R. Pri izračunu razdalj lahko izbiramo med različnimi metodami. Več informacij je v opisu funkcije *dist*, ki jo bomo uporabili. Izračun hierarhičnega gručenja bo izdelan s funkcijo *hclust*. Programsko okolje R pa vsebuje tudi funkcije za določitev gručenja na podlagi algoritma K-means (*kmeans*). Za vizualizacijo rezultatov algoritma K-means bomo uporabili paket *factoextra*.

```
library(dplyr, quietly=TRUE)

## Warning: package 'dplyr' was built under R version 4.1.3

##
## Attaching package: 'dplyr'

## The following object is masked from 'package:hydrostats':
##
##     recode

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:xts':
##
##     first, last
```

---

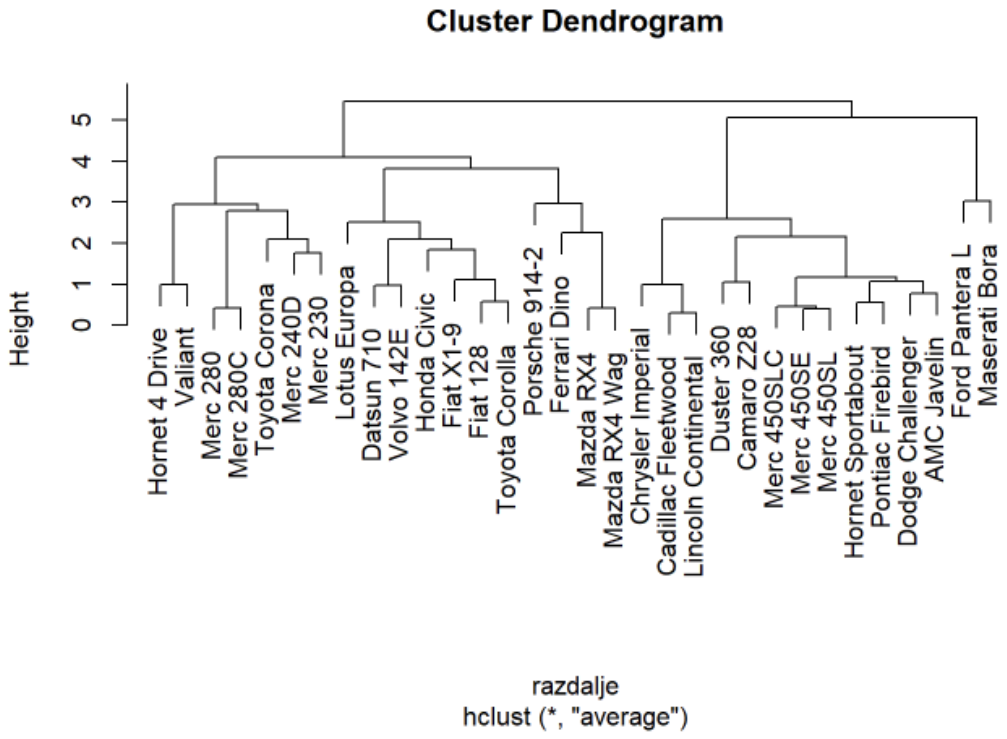
[119] https://www.geeksforgeeks.org/introduction-to-machine-learning-in-r/.

[120] https://www.geeksforgeeks.org/7-best-r-packages-for-machine-learning/.

[121] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=124149.

[122] https://dk.um.si/IzpisGradiva.php?id=35730&lang=eng&prip=rup:1491940:d4.

```
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(BBmisc, quietly=TRUE)

## Warning: package 'BBmisc' was built under R version 4.1.3

##
## Attaching package: 'BBmisc'

## The following objects are masked from 'package:dplyr':
##
##     coalesce, collapse, symdiff

## The following object is masked from 'package:base':
##
##     isFALSE

mtcars1 <- normalize(mtcars)
razdalje <- dist(mtcars1, method = 'euclidean')
gručenje <- hclust(razdalje, method = "average")
plot(gručenje) # izris rezultatov
```



Slika 81: Hierarhično gručenje podatkov iz podatkovne baze mtcars.

```
# če želimo podatke združiti v tri (k=3) skupine,
# lahko vidimo, kateri elementi so združeni v iste skupine
cutree(gručenje, k = 3)
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4
Drive
##                   1                   1                   1
1
##   Hornet Sportabout             Valiant          Duster 360           Merc
240D
##                   2                   1                   2
1
##            Merc 230            Merc 280           Merc 280C           Merc
450SE
##                   1                   1                   1
2
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Contin
ental
##                   2                   2                   2
2
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Co
rolla
##                   2                   1                   1
1
##       Toyota Corona    Dodge Challenger         AMC Javelin           Camar
o Z28
##                   1                   2                   2
2
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus E
uropa
##                   2                   1                   1
1
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo
142E
##                   3                   1                   3
1
```

```
kmeansrez <- kmeans(mtcars1,centers=3) # k-means
# rezultati gručenja so drugačni kot v primeru hierarhičnega gručenja
kmeansrez$cluster
```

```
##           Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4
Drive
##                   1                   1                   1
3
##   Hornet Sportabout             Valiant          Duster 360           Merc
240D
##                   2                   3                   2
3
##            Merc 230            Merc 280           Merc 280C           Merc
450SE
```

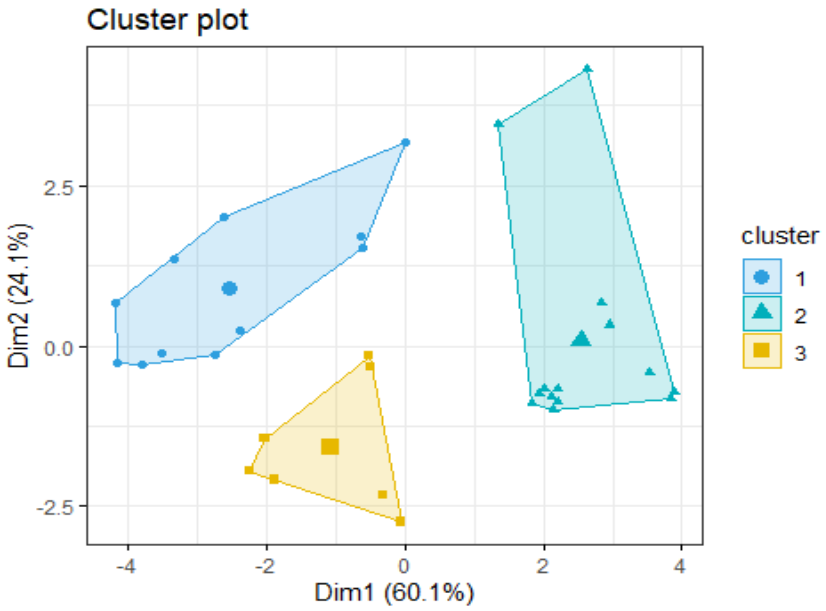
```
##                   3                   3                   3
2
##          Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Contin
ental
##                   2                   2                   2
2
##   Chrysler Imperial            Fiat 128         Honda Civic      Toyota Co
rolla
##                   2                   1                   1
1
##       Toyota Corona    Dodge Challenger         AMC Javelin          Camar
o Z28
##                   3                   2                   2
2
##    Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus E
uropa
##                   2                   1                   1
1
##      Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo
142E
##                   2                   1                   2
1
```

```
library(factoextra, quietly=TRUE)
```

```
## Warning: package 'factoextra' was built under R version 4.1.3
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://g
oo.gl/ve3WBa
```

```
# izrišemo tri skupine, ki smo jih določili na podlagi algoritma K-means
fviz_cluster(kmeansrez, data = mtcars,
  palette = c("#2E9FDF", "#00AFBB", "#E7B800"),
  geom = "point",ellipse.type = "convex",
  ggtheme = theme_bw())
```



Slika 82: Tri skupine, določene na podlagi K-means algoritma.

Omeniti velja tudi regresijska drevesa (angl. *regression trees*) in ojačana regresijska drevesa (angl. *generalized boosted regression trees*), ki so bila uporabljena tudi v primeru analize vpliva meteoroloških in vegetacijskih parametrov na prestrezanje padavin[123]. Regresijska drevesa so ena izmed metod strojnega učenja, ki omogoča določitev modela za napoved izbrane spremenljivke glede na vhodne podatke. Modeli regresijskih dreves se določijo s ponavljajočimi se delitvami vhodnih podatkov z določenim prilagajanjem modela za napoved glede na ciljno spremenljivko. Rezultat postopka določitve regresijskega drevesa lahko prikažemo grafično z odločitvenim oziroma regresijskim drevesom[124]. V sklopu programskega okolja R izgradnjo odločitvenega drevesa omogoča paket *rpart*. Po drugi strani ojačana regresijska drevesa izboljšujejo rezultate običajnih regresijskih dreves preko testiranja velikega števila modelov in analize rezultatov glede na uspešnost teh modelov[125]. Ojačana regresijska drevesa je v sklopu programskega okolja R mogoče določiti s paketom *gbm*. Za prikaz uporabe odločitvenih dreves bomo uporabili podatke o kakovosti zraka *airquality*. Uporabili bomo tiste podatke, kjer imamo na voljo podatek o vrednosti ozona ter samo stolpce od 1 do 4. Podatke bomo preoblikovali glede na mejne vrednosti ozona, ki označujejo različne vrednosti glede na izmerjene vrednosti. Podatek o ozonu je spremenljivka, ki jo napovedujemo glede na ostale spremenljivke. Več informacij o dodatnih možnostih funkcije *rpart* je na voljo v opisu funkcije.

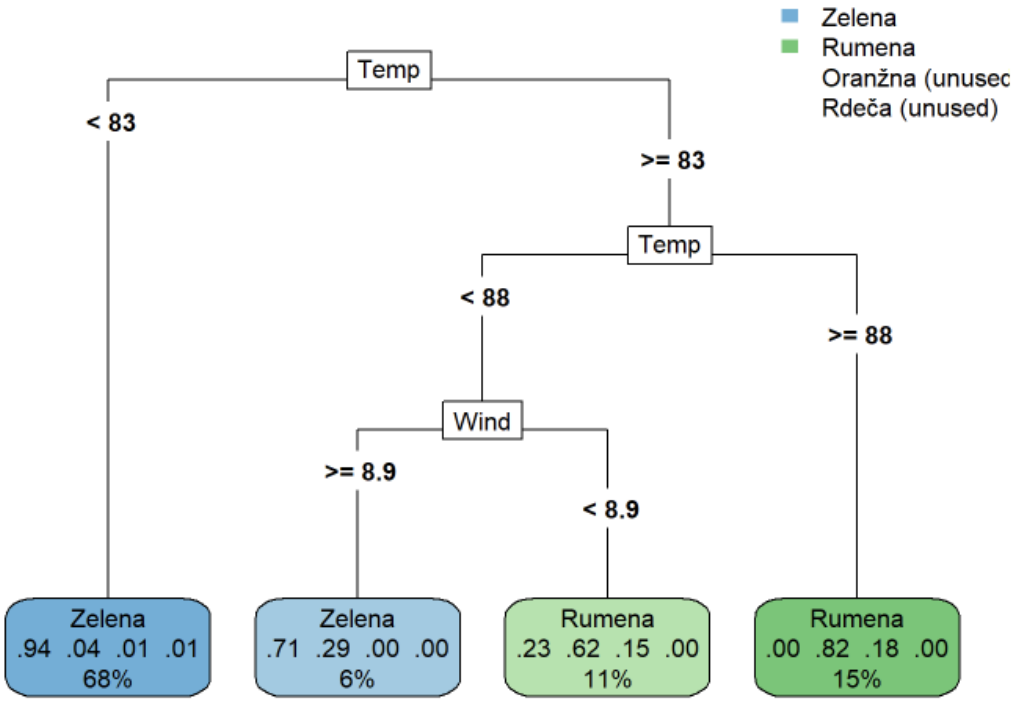
```
data("airquality")
vzorec <- airquality
```

[123] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=101711&lang=slv.

[124] https://repozitorij.uni-lj.si/IzpisGradiva.php?id=97239.

[125] https://doi.org/10.1111/j.1365-2656.2008.01390.x.

```
# samo podatke, kjer so na voljo meritve ozona
vzorec <- vzorec[-which(is.na(vzorec$Ozone)),1:4]
# razdelitev v kategorije
vzorec$Ozone <- cut(vzorec$Ozone,breaks=c(0,50,100,150,200),
  labels=c("Zelena","Rumena","Oranžna","Rdeča")) #
library(rpart, quietly=TRUE)
# paket rpart.plot omogoča več možnosti izrisa regresijskih dreves
library(rpart.plot, quietly=TRUE)

## Warning: package 'rpart.plot' was built under R version 4.1.3

# definiramo model
drevo <- rpart(Ozone ~ Solar.R + Wind + Temp,data=vzorec)
rpart.plot(drevo,type=5)
```



Slika 83: Odločitveno drevo za podatke o kakovosti zraka.

Vidimo, da je prvi element odločitve podatek v temperaturi zraka. Če je temperatura nižja od 83 F (približno 28 stopinj Celzija), so bile vrednosti ozona nizke (zelena barva), v primeru vrednosti nad 83 F pa imamo lahko vrednosti ozona tudi v rumenem razponu, odvisno od vrednosti hitrosti vetra (visoka hitrost -> nižje vrednosti ozona). Vidimo, da je bilo v našem vzorcu premalo oranžnih in rdečih vrednosti, in izbrani model regresijskega drevesa ni uspel napovedati teh vrednosti. Razlog je neuravnoteženost podatkov, kjer imamo premalo podatkov v kategorijah oranžna in rdeča. Posledično bi veljalo uporabili daljše nize podatkov oziroma uporabiti katero izmed drugih možnosti za pripravo ustreznega vzorca. Sedaj pa bomo preverili še ustreznost modela (kljub določenim pomanjkljivostim). Podatke bomo razdelili na dva dela, približno 70 % za umerjanje (kalibracijo) in preostanek za preverjanje (validacijo). Model bomo definirali samo na kalibracijskem delu podatkov. Ustreznost

modela bomo ocenili glede na enačbo (TP + TN)/ (TP + TN + FP + FN), kjer so TP (true positive) pravilne pozitivne, TN (true negative) pravilne negativne, FP (false positive) napačne pozitivne in FN (false negative) napačne negativne vrednosti. Torej TP in TN so pravilno napovedane vrednosti, FP in FN so nepravilno napovedane vrednosti, seštevek TP + TN + FP + FN je seštevek vseh napovedi.

```
# da zagotovimo, da bodo naključno generirana števila vedno enaka
set.seed(1)
# izmed 116 podatkov uporabimo 80 podatkov za umerjanje izbranega modela
izbira <- sample.int(116,80)
# izberemo del podatkov za umerjanje, približno 70 %
kalib <- vzorec[izbira,]
# preostanek za validacijo
valid <- vzorec[-izbira,]
# definiramo model
drevokalib <- rpart(Ozone ~ Solar.R + Wind + Temp,data=kalib)
# za validacijski del uporabimo prej definirani model in napovemo vrednosti
napoved <- predict(drevokalib, valid, type = 'class')
# podatke združimo
tabela <- table(valid$Ozone,napoved)
# preverimo rezultate, diagonalne vrednosti so pravilno napovedane vrednosti
tabela
```

```
##          napoved
##           Zelena Rumena Oranžna Rdeča
##   Zelena      26      1       0     0
##   Rumena       2      6       0     0
##   Oranžna      0      1       0     0
##   Rdeča        0      0       0     0
```

```
# izračun statistike
accuracy_Test <- sum(diag(tabela)) / sum(tabela)
accuracy_Test
```

```
## [1] 0.8888889
```

Vidimo, da je naš model regresijskih dreves uspel pravilno napovedati približno 89 % vseh dni, ki smo jih uporabili za validacijo našega modela, izboljšanje rezultatov bi bilo morda mogoče doseči z uporabo drugačnih nastavitev regresijskih dreves (*rpart*). Prikažimo pa še primer uporabe ojačenih regresijskih dreves z uporabo paketa *gbm*.

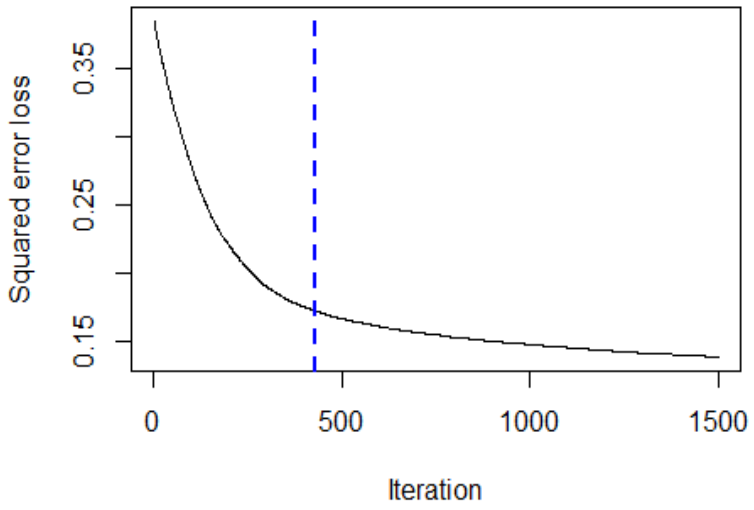
```
library(gbm, quietly=TRUE)
```

```
## Warning: package 'gbm' was built under R version 4.1.3
```

```
## Loaded gbm 2.1.8.1
```

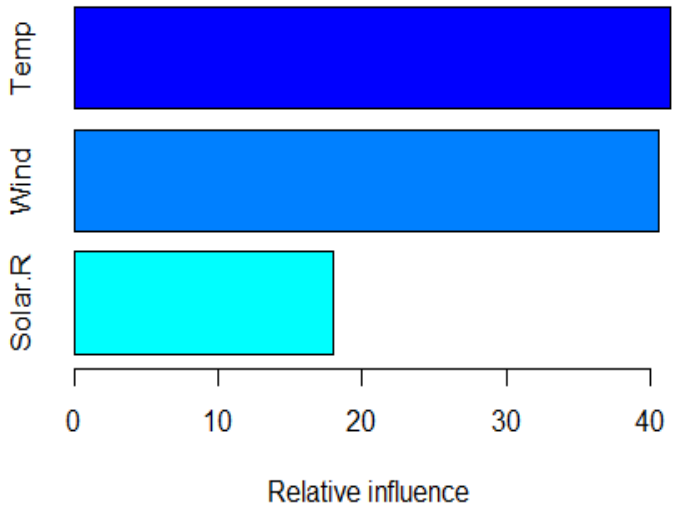
```
# definiramo model ojačenih regresijskih dreves
gbm1 <- gbm(formula = Ozone ~ Solar.R + Wind + Temp,
  distribution = "gaussian", data = vzorec, n.trees = 1500,
  shrinkage = .005, cv.folds=5)
```

```
# pogledamo najboljšo iteracijo modela
best.iter <- gbm.perf(gbm1, method="OOB")

## OOB generally underestimates the optimal number of iterations although pre
dictive performance is reasonably competitive. Using cv_folds>1 when calling
gbm usually results in improved predictive performance.
```



Slika 84: Najboljša iteracija modela ojačenih regresijskih dreves.

```
# povzetek rezultatov
summary(gbm1)
```



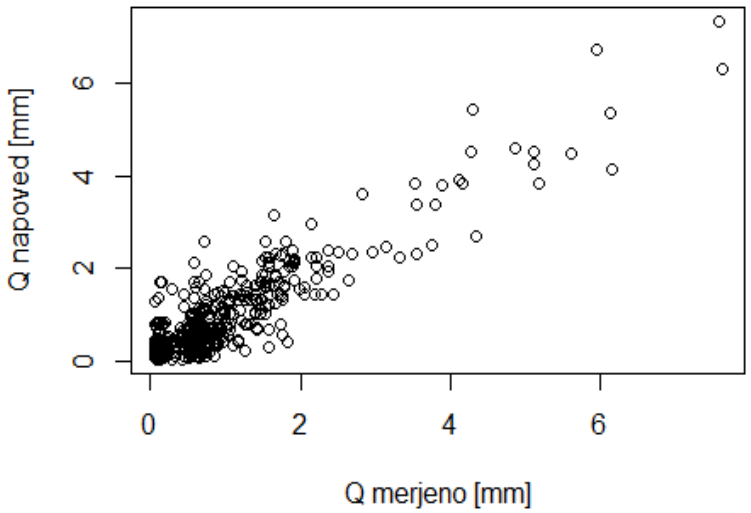
Slika 85: Relativni vpliv posameznih spremenljivk na vrednosti ozona.

```
##             var  rel.inf
## Temp       Temp 41.39562
## Wind       Wind 40.57956
## Solar.R Solar.R 18.02482
```

Vidimo lahko, da je vpliv temperature zraka in hitrosti vetra na vrednosti ozona precej večji kot vpliv sončne radiacije. Tudi v primeru ojačenih regresijskih dreves je mogoče vzorec razdeliti na dva dela in nato narediti napoved podatkov za neodvisni del podatkov z uporabo funkcije predict, ki je vključena v paket *gbm*.

Kot zanimivost pokažimo še primer uporabe algoritma KNN (angl. *k-nearest neighbour*)[126] na primeru numeričnih podatkov o pretokih, ki so vključeni v paket *airGR*. Algoritem KNN je vključen v paket *class*. Model KNN bomo umerili na delu podatkov, z uporabo algoritma KNN pa bomo nato napovedali vrednosti podatkov za drugi del vzorca in ocenjene vrednosti primerjali z izmerjenimi. Primerjava različnih metod strojnega učenja s klasičnimi hidrološkimi modeli je bila narejena tudi v okviru prispevka, ki so ga pripravili Sezen in sodelavci (2019)[127].

```
library(airGR, quietly=TRUE)
data(L0123001) # uporaba podatkov v paketu airGR
# izberemo del podatkov o pretokih za umerjanje modela
kalib2 <- BasinObs[5000:6000,c(2,3,4,6)]
# del pa za kontrolo
valid2 <- BasinObs[6001:6500,c(2,3,4,6)]
library(class, quietly=TRUE)
# uporaba algoritma KNN za napoved pretokov
# uporabimo tri najbližje vrednosti (k=3)
knnnapoved <- knn(train=kalib2,test=valid2,cl=kalib2$Qmm,k=3)
# primerjava merjenih in napovedanih vrednosti
plot(valid2$Qmm,as.numeric(as.character(knnnapoved)),
    xlab="Q merjeno [mm]",ylab="Q napoved [mm]")
```



Slika 86: Primerjava rezultatov algoritma KNN z meritvami pretokov.

[126] https://doi.org/10.1017/CBO9780511812651.

[127] https://doi.org/10.1016/j.jhydrol.2019.06.036.

```
# ter izračun vrednosti koeficienta determinacije R^2
cor(valid2$Qmm,as.numeric(as.character(knnnapoved)))^2

## [1] 0.8210452
```

Na podoben način bi lahko uporabili tudi nekatere druge algoritme strojnega učenja, kot je na primer algoritem random forest (paket *randomForest*).

Naloga 40: Na podatkih z vodomerne postaje Veliko Širje (Savinja, leto 2005) ocenite uspešnost algoritma regresijskih dreves za napovedovanje koncentracije suspendiranih snovi (na podlagi podatkov o pretokih in temperaturi vode), pri tem del podatkov (okoli 80 %) uporabite za umerjanje modela, preostanek pa za preverjanje.

4.15 Prostorski podatki

Programsko okolje R vključuje veliko število paketov, s katerimi lahko izvajamo različne analize prostorskih podatkov. Obstaja tudi veliko uporabnega gradiva, ki pokriva to področje, kot je na primer knjiga Applied Spatial Data Analysis with R[128] ali spletna stran rspatial.org[129], kjer so prikazani tudi številni praktični primeri. Za poglobljeno teoretično razumevanje tega področja velja omeniti tudi knjigo Introduction to Geostatistics[130] ali učbenik prof. Gorana Turka Prostorska statistika[131]. Omeniti pa velja, da so podatki, ki jih uvozimo v R, shranjeni v spominu, kar posledično omejuje zelo obsežne izračune (ali prostorsko ali časovno), ter da R prvotno ni bil razvit za analize in vizualizacijo kart (kot na primer programi GIS), kljub temu pa nekateri paketi vključujejo funkcije, ki to (delno) omogočajo. Naštejmo še nekaj paketov, ki vključujejo uporabne funkcije za analize podatkov in geostatistične operacije, kot so *sp*, *sf*, *rgdal*, *raster*, *terra*, *netcdf4*, *rgeos*, *maptools*, *gstat*, *geoR*, *Fields*, *spatialExtremes*, *spacetime*, *spBayes*. Paket *raster* ni več v razvoju in ga je nasledil paket *terra*, tudi paketi *rgdal*, *rgeos* in *maptools* niso več na voljo na repozitoriju CRAN. Nekatere prikaze uporabnih funkcij bomo naredili z uporabo podatkov za porečje reke Sore do vodomerne postaje Suha[132]. Najprej bodo prikazane nekatere osnovne funkcije za uvoz in shranjevanje podatkov ter prikaz prostorskih podatkov. Večina spodaj prikazanih funkcij izhaja iz paketa *terra*.

```
library(terra, quietly=TRUE)
library(raster, quietly=TRUE)
```

---

[128] https://doi.org/10.1007/978-1-4614-7618-4.

[129] https://rspatial.org/rs/index.html.

[130] https://osf.io/6jzpn/download.

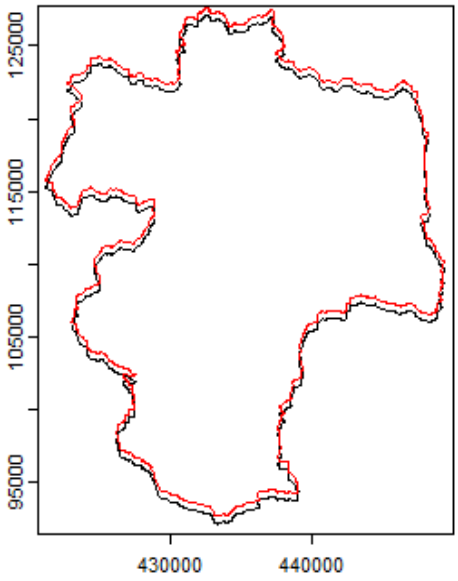
[131] https://doi.org/10.15292/9789612971991.

[132] https://unilj-my.sharepoint.com/:f:/g/personal/nbezak_fgg_uni-lj_si/Eo74IB589AJOgkL3baraPqEBHyoHbW6UXJ15qsh0p_YXjw?e=Q2eOcy.

```
# uvozimo podatke o porečju, razvodnico
porecje <- vect("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/
4200_Sora_Suha.shp")
# poglejmo osnovne lastnosti o porečju
porecje
```

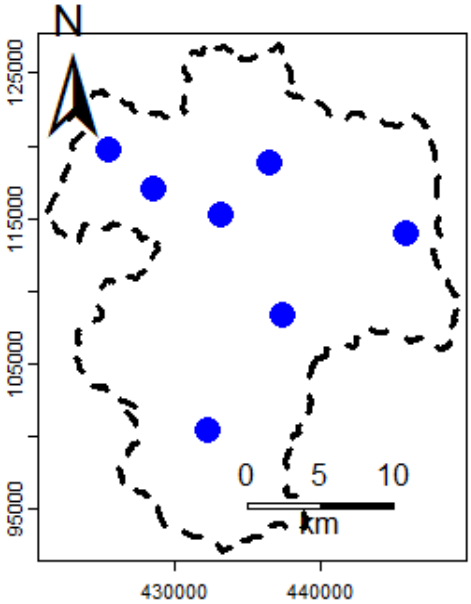
```
##  class       : SpatVector
##  geometry    : polygons
##  dimensions  : 1, 2  (geometries, attributes)
##  extent      : 421172, 449307.4, 92142.4, 127056.8  (xmin, xmax, ymin, yma
x)
##  source      : 4200_Sora_Suha.shp
##  coord. ref. : gauss_krueger_SLO
##  names       : opomba      AREA
##  type        :  <chr>     <num>
##  values      :     NA 5.689e+08
```

```
# za shranjevanje vektorskih podatkov v sklopu paketa terra
# lahko uporabimo funkcijo writeVector
# preberemo še podatke o padavinskih postajah
postaje <- vect("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/
Stations2.shp")
# preverimo koordinatni sistem padavinskih postaj
crspost <- crs(postaje)
# spremenimo koordinatni sistem izbranega porečja
porecje2 <- project(porecje,crspost)
# lahko tudi z uporabo epsg kode: https://epsg.io/3794,
# 3787 je stari koordinatni sistem v Sloveniji
porecje3 <- project(porecje, "+init=epsg:3794")
# opazimo lahko spremembo med obema koordinatnima sistemoma
plot(porecje); lines(porecje3,col="red")
```



Slika 87: Razvodnica porečja Sore (Suha). Prikazana je razvodnica v dveh različnih koordinatnih sistemih.

```
plot(porecje,lwd=3, lty=2,) # izrišemo razvodnico
# na graf dodamo še lokacije izbranih postaj
points(postaje, col="blue", pch=16, cex=2)
# alternativna možnost je tudi plot(postaje, add=TRUE)
# na graf dodamo še puščico za sever
# funkcija je na voljo v novejših različicah paketa terra
north(type=2,cex=1.5,"topleft")
# dodamo še legendo
sbar(10000, c(435000, 95000), type="bar", divs=2, below="km",label=c(0,5,10))
```



Slika 88: Porečje Sore in izbrane padavinske postaje.

Pokažimo še, kako lahko preverimo atribute padavinskih postaj. Do posameznih elementov atributne tabele lahko dostopamo podobno kot v primeru podatkovnih okvirjev oziroma deluje tudi matrično sklicevanje. Podatkom, ki smo jih uvozili, bomo dodali nov stolpec, ki predstavlja povprečno letno količino padavin za izbrane padavinske postaje za obdobje 2016–2021 glede na podatke, ki so na voljo v arhivu ARSO.

```
as.data.frame(postaje) # atributna tabela
```

```
##          Station    GKX    GKY
## 1          Topol 451758 105735
## 2    _kofja Loka 445736 114445
## 3      _elezniki 436375 119269
## 4   Martinji Vrh 433179 115743
## 5        Poljane 437385 108787
## 6           _iri 432255 100816
## 7 Zgornja Sorica 425430 120227
## 8          Dav_a 428474 117475
```

```
postaje$Station # posamezen atribut
```

```
## [1] "Topol"          "_kofja Loka"    "_elezniki"      "Martinji Vrh"
## [5] "Poljane"        "_iri"           "Zgornja Sorica" "Dav_a"
```

```
head(geom(porecje)) # geometrijske lastnosti
```

```
##      geom part        x        y hole
## [1,]    1    1 439632.6 105203.2    0
## [2,]    1    1 439616.5 105162.4    0
## [3,]    1    1 439610.6 105115.3    0
## [4,]    1    1 439594.4 105077.0    0
## [5,]    1    1 439571.0 105002.4    0
## [6,]    1    1 439538.2 104969.0    0
```

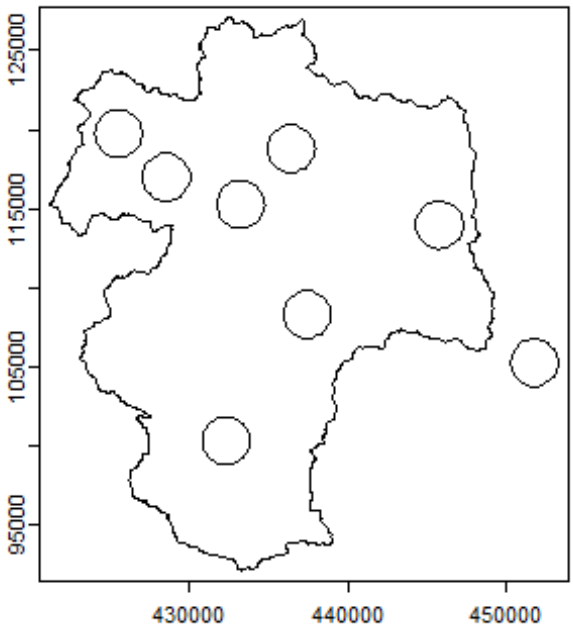
```
# dodamo nov stolpec v atributno tabelo
postaje$padavine = c(1684, 1575, 1803, 2347, 1620, 1917, 2224, 1950)
# postaje$GKX = NULL # če bi želeli izbrisati določen stolpec
expanse(porecje, unit="km") # površina območja v izbranih enotah
```

```
## [1] 569.0504
```

```
perim(porecje) # še obseg porečja
```

```
## [1] 146261.2
```

```
# aggregate(porecje, dissolve=T) # če bi imeli več porečij,
# bi jih lahko združili, uporabna je tudi funkcija expanse
postbuf <- buffer(postaje, 1500) # območja v okolici postaj (obseg 1500 m)
# združimo različne podatke v en sloj
# omeniti še funkciji intersect in symdif
plot(union(porecje,postbuf))
```



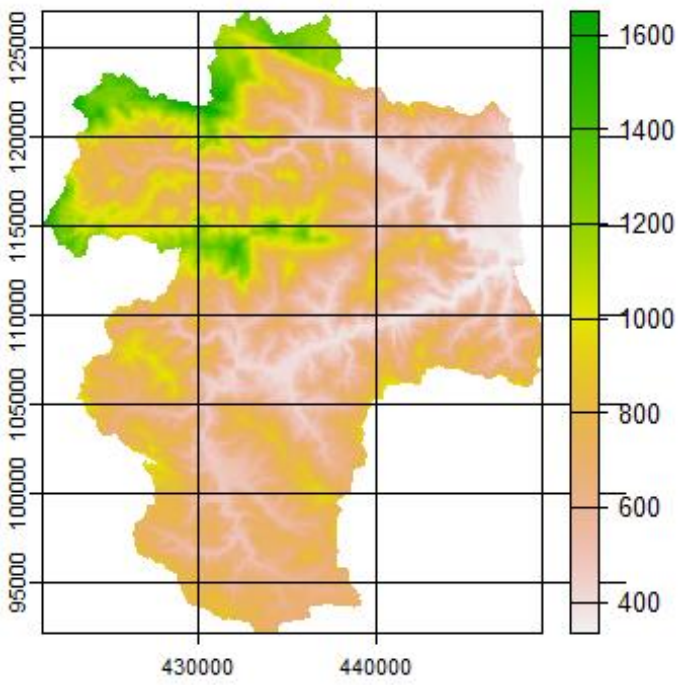
Slika 89: Porečje Sore in padavinske postaje z obsegom 1500 m okrog postaje.

Na podoben način lahko v programsko okolje R uvozimo tudi rastrske podatke. Uvozili bomo podatke o digitalnem modelu višin za isto porečje. Najprej bodo prikazane nekatere funkcije za izris rastrskih podatkov.

```
# uvoz podatkov
teren <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/So
raSuha.sdat")
# za shranjevanje rastrskih podatkov v paketu terra
# lahko uporabimo funkcijo writeRaster
print(teren) # osnovne lastnosti uporabljenega rastrskega sloja
```

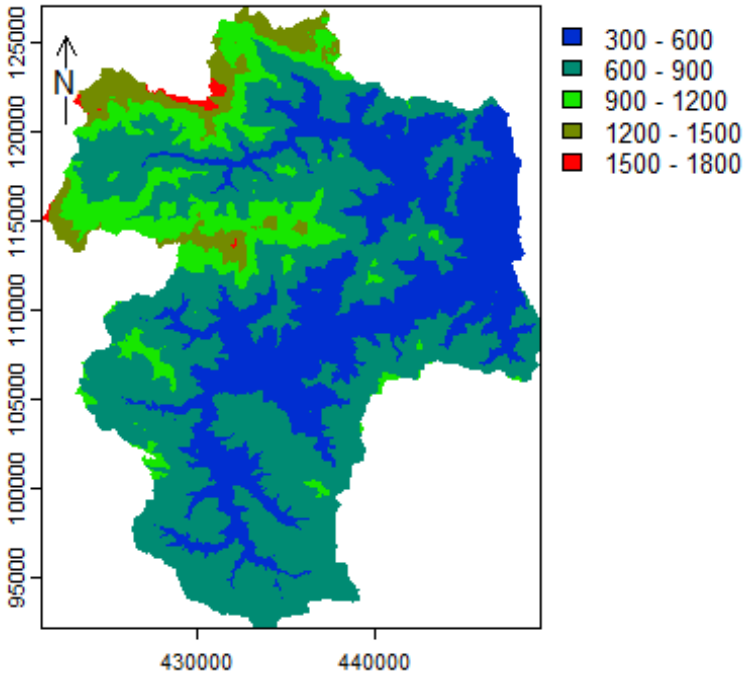
```
## class       : SpatRaster
## dimensions  : 349, 280, 1  (nrow, ncol, nlyr)
## resolution  : 100, 100  (x, y)
## extent      : 421212, 449212, 92172, 127072  (xmin, xmax, ymin, ymax)
## coord. ref. : gauss_krueger_SLO
## source      : SoraSuha.sdat
## name        :  SoraSuha
## min value   :  333.4742
## max value   : 1650.1703
```

```
# izrišemo digitalni model višin, funkcija plot omogoča
# tudi številne druge nastavitve
plot(teren, axes=T, grid=T)
```



Slika 90: Digitalni model višin.

```
# nekoliko bolj napreden graf
plot(teren, axes=T, breaks=c(300,600,900,1200,1500,1800),
   col = colorRampPalette(c("blue", "green", "red"))(12))
north("topleft") # dodamo še oznako za sever
```

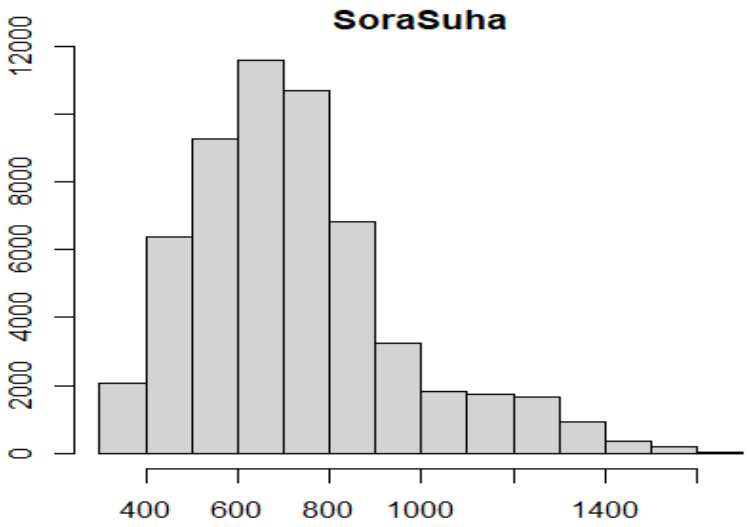


Slika 91: Digitalni model višin po razredih.

```
# podatki o nadmorski višini posameznih celic
head(as.data.frame(teren))
```

```
##     SoraSuha
## 113 1328.688
## 114 1322.766
## 115 1328.819
## 392 1327.994
## 393 1325.541
## 394 1312.359
```

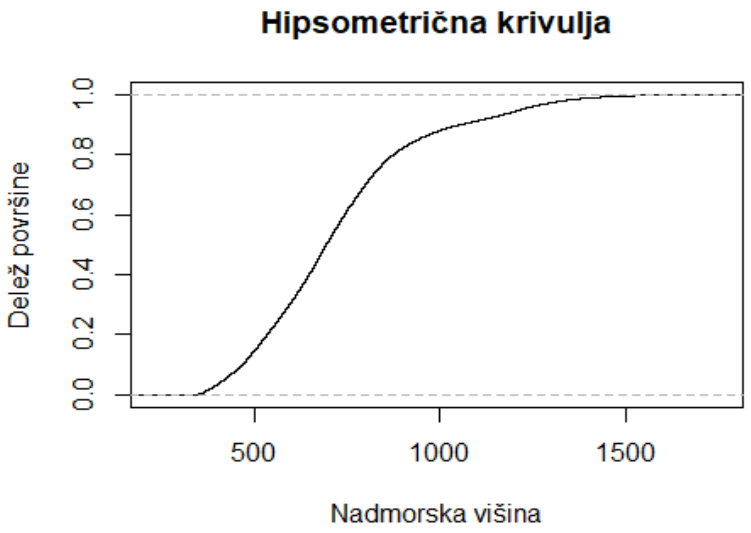
```
# uporabimo lahko tudi funkcijo values(teren)
hist(teren) # izrišemo lahko tudi histogram podatkov
```



Slika 92: Histogram podatkov o nadmorski višini.

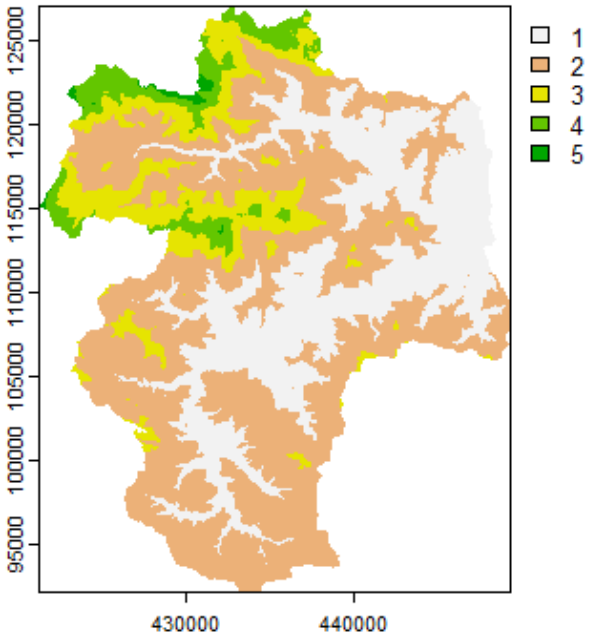
Izrišemo lahko tudi hipsometrično krivuljo porečja ter izvajamo številne uporabne operacije, kot so *aggregate* ali *disagg*. Omeniti velja še funkcije *mosaic* (združevanje večjega števila rastrskih podatkov v en sloj), *focal*, *crop* in *merge*. V spodnjem primeru bo prikazana tudi klasifikacija podatkov v razrede. Spremenili bomo rastrske vrednosti celic glede na meje (spodnja meja in zgornja meja) ter glede na nove vrednosti (3. številka za spodnjo in zgornjo mejo). V našem primeru bomo imeli pet razredov.

```
# hipsometrična krivulja
plot(ecdf(as.numeric(values(teren))),xlab="Nadmorska višina",
    ylab="Delež površine",main="Hipsometrična krivulja")
```



Slika 93: Hipsometrična krivulja porečja Sore.

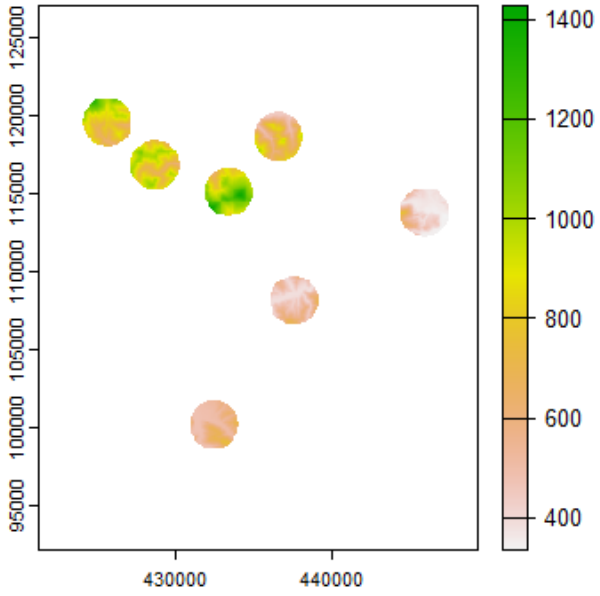
```
# izračunamo še posamezne vrednosti kvartilov
quantile(as.numeric(values(teren)),probs=c(0.25,0.5,0.75),na.rm=TRUE)

##      25%      50%      75%
## 563.6654 692.8478 830.6627

# preverimo, katera celica je locirana na določenih koordinatah
cellFromXY(teren, data.frame(x=436000,y=110000))

## [1] 47748

# obratno kot zgoraj
xyFromCell(teren,47748)

##           x      y
## [1,] 435962 110022

# spremenimo resolucijo podatkov (resampling)
teren2 <- aggregate(teren, fact=4, fun=mean)
# obratno kot zgoraj, disagregacija podatkov
teren3 <- disagg(teren2, fact=4, method="near")
# klasifikacija podatkov
terenklas <- classify(teren, rbind(c(300,600,1),c(600,900,2),c(900,1200,3),
  c(1200,1500,4),c(1500,1800,5)))
plot(terenklas) # izrišemo še spremenjene podatke
```



Slika 94: Rastrski podatki, razdeljeni v pet razredov.

Pogosto so zanimive tudi interakcije in operacije med rastrskimi in vektorskimi podatki. Transformacijo poligona v rastrsko obliko lahko izvedemo s funkcijo *rasterize*, obratna funkcija je *as.polygons*, ki omogoča transformacijo iz rastrskih v vektorske podatke. Uporabne so tudi funkcije *mask*, *crop*, *extract* itd.

```
# iz vektorskih v rastrske podatke
porecje_rast <- rasterize(porecje, project(teren, crs(porecje)))
# funkcija mask, koordinatni sistem mora biti enak
plot(mask(teren,postbuf))
```



Slika 95: Nadmorska višina v okolici padavinskih postaj.

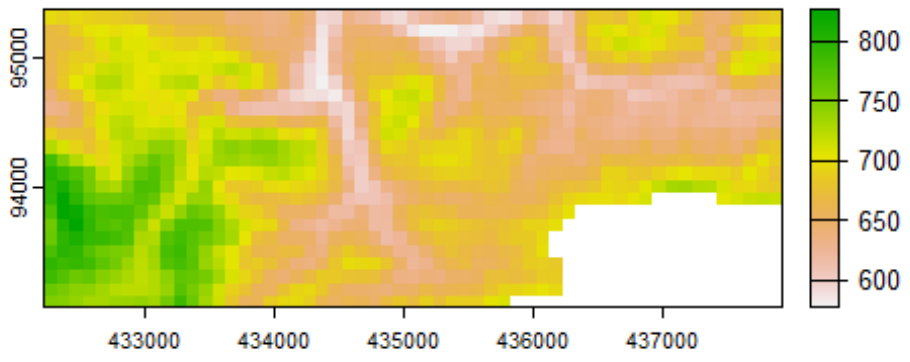
```
# izvrednotimo vrednosti za posamezne lokacije
# prva postaja ni locirana na porečju reke Sore
extract(teren,postaje)
```

```
##   ID  SoraSuha
## 1  1        NA
## 2  2  379.1530
## 3  3  784.2018
## 4  4 1051.6125
## 5  5  420.5624
## 6  6  529.8202
## 7  7  806.4953
## 8  8  842.2570
```

```
# enako še za večja območja v okolici padavinskih postaj,
# kjer se nato na podlagi vseh celic izračuna povprečje
extract(teren,postbuf,mean)
```

```
##   ID  SoraSuha
## 1  1       NaN
## 2  2  419.3978
## 3  3  676.0768
## 4  4 1011.1084
## 5  5  478.5840
## 6  6  554.3724
## 7  7  841.8691
## 8  8  858.5206
```

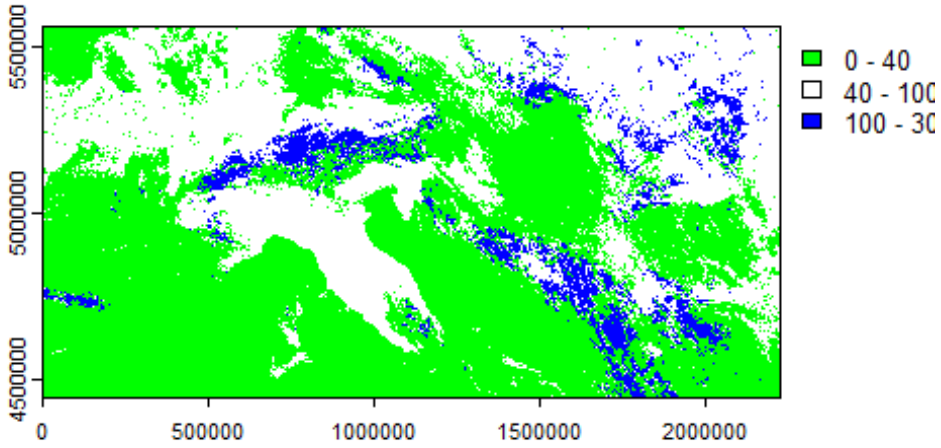
```
# uvozimo še linije vodotokov
vodotoki <- vect("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS
/Vodotoki.shp")
# uporaba funkcije crop za določene odseke vodotokov (2:5)
plot(crop(teren,vodotoki[2:5,]))
```



Slika 96: Uporaba funkcije crop.

V nekaterih primerih moramo analize narediti tudi na podlagi satelitskih podatkov, kot na primeru, ki so ga predstavili Parajka in sodelavci[133]. Podatke MODIS v povezavi s snežno odejo lahko sicer pridobite na spletni strani NASA[134]. V naslednjem primeru bo prikazana uporaba podatkov MODIS o snežni odeji. Ti podatki so v naslednji obliki: celice z vrednostjo 0–40 so območja brez snega, celice z vrednostjo 41–100 so območja s snegom, celice z vrednostjo 101–300 so območja, pokrita z oblaki. Naredili bomo presek s porečjem Sore in izračunali število celic z oblaki, s snegom in brez snega.

```
MODISsneg <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/MOD10A1.A2000057.Snow_Cover_Daily_Tile.tif")
# struktura podatkov MODIS
plot(MODISsneg,breaks=c(0,40,100,300),col=c("green","white","blue"))
```

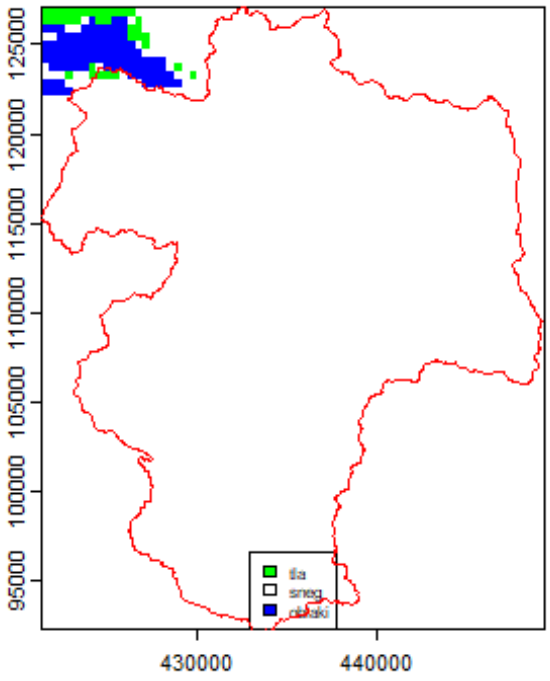


Slika 97: MODIS podatki o snežni odeji.

```
# spremenimo koordinatni sistem podatkov MODIS
MODISsneg2 <- project(MODISsneg,crspost)
# naredimo presek z mejo porečja
presek <- crop(MODISsneg2,porecje)
# izrišemo še presek
plot(presek,breaks=c(0,40,100,300),col=c("green","white","blue"),legend=F)
# dodamo še legendo
legend("bottom",legend = c("tla", "sneg", "oblaki"),
  fill = c("green","white","blue"),cex=0.5)
# izrišemo mejo porečja
lines(porecje,col="red")
```

[133] https://doi.org/10.2478/johh-2018-0011.

[134] https://n5eil01u.ecs.nsidc.org/MOST/MOD10A1.061/.



Slika 98: Podatki MODIS za porečje Sore.

```
# vidimo, da je bilo ta dan skoraj celotno porečje pokrito s snegom
# izračunamo število celic z oblaki
noblak <- freq(classify(presek,c(101,500,1),others=NA),value=1)[3]
# izračunamo število celic s snegom
nsneg <- freq(classify(presek,c(40,100,1),others=NA),value=1)[3]
# izračunamo število celic brez snega
ntla <- freq(classify(presek,c(0,40,1),others=NA),value=1)[3]
# izračunamo pokritost s snegom
nsneg*100/(nsneg+ntla)
```

```
##      count
## 1 99.33425
```

Obstajajo pa tudi številni drugi produkti MODIS. Nekatere izmed njih lahko uvozimo v programsko okolje R neposredno preko uporabe paketov, kot je *MODISTools*. V naslednjem primeru bo prikazana uporaba paketa *MODISTools* za uvoz enega izmed produktov (temperatura površja) za izbrano lokacijo glede na koordinate. Na podlagi podatkov bomo nato izračunali povprečno temperaturo površja za porečje Sore do vodomerne postaje Suha.

```
library(MODISTools, quietly=TRUE)
# seznam vseh produktov, ki jih lahko prenesemo
head(mt_products())
```

```
##        product
## 1       Daymet
## 2 ECO4ESIPTJPL
## 3      ECO4WUE
## 4       GEDI03
```

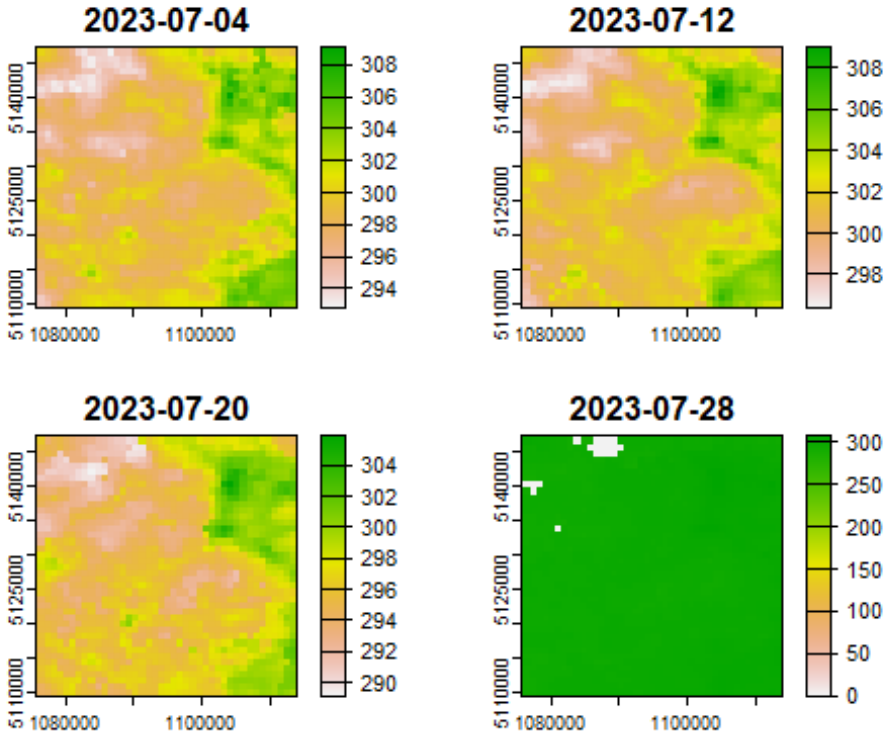
```
## 5     GEDI04_B
## 6      MCD12Q1
##                                                                    d
escription
## 1 Daily Surface Weather Data (Daymet) on a 1-km Grid for North America, Ve
rsion 4 R1
## 2               ECOSTRESS Evaporative Stress Index PT-JPL (ESI) Daily L4 G
lobal 70 m
## 3                          ECOSTRESS Water Use Efficiency (WUE) Daily L4 G
lobal 70 m
## 4                GEDI Gridded Land Surface Metrics (LSM) L3 1km EASE-Grid,
Version 2
## 5     GEDI Gridded Aboveground Biomass Density (AGBD) L4B 1km EASE-Grid, V
ersion 2.1
## 6              MODIS/Terra+Aqua Land Cover Type (LC) Yearly L3 Global 500
m SIN Grid
##   frequency resolution_meters
## 1     1 day              1000
## 2    Varies                70
## 3    Varies                70
## 4  One time              1000
## 5  One time              1000
## 6    1 year               500
```

```
# preverimo, kateri podatki so na voljo v določenem produktu
head(mt_bands("MYD11A2"))
```

```
##               band                          description valid_range fill_v
alue
## 1   Clear_sky_days               Day clear-sky coverage    1 to 255
0
## 2 Clear_sky_nights             Night clear-sky coverage    1 to 255
0
## 3    Day_view_angl View zenith angle of day observation    0 to 130
255
## 4    Day_view_time        Local time of day observation    0 to 240
255
## 5          Emis_31                    Band 31 emissivity    1 to 255
0
## 6          Emis_32                    Band 32 emissivity    1 to 255
0
##    units scale_factor add_offset
## 1   <NA>         <NA>       <NA>
## 2   <NA>         <NA>       <NA>
## 3 degree            1        -65
## 4    hrs          0.1          0
## 5   <NA>        0.002       0.49
## 6   <NA>        0.002       0.49
```

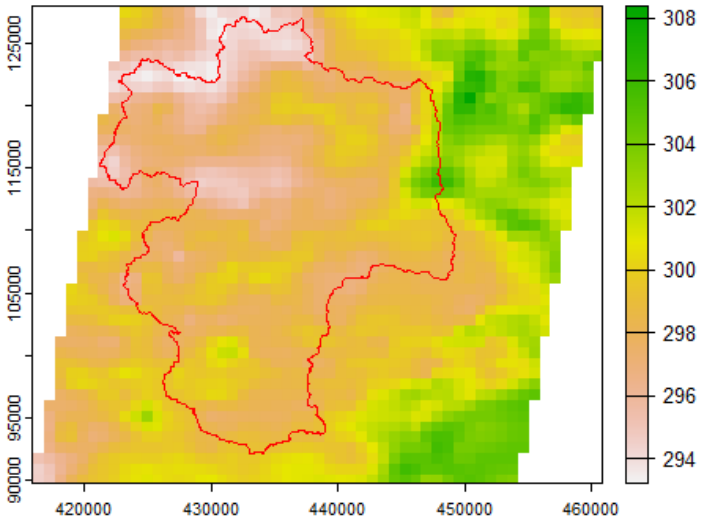
```
# izbrani produkt in določen podatek lahko sedaj tudi uvozimo
# glede na izbrano koordinato (lat, long) ter glede na +- razdaljo
```

```
# od te koordinate (argumenta km_lr in km_ab) ter za izbrane datume
LC <- mt_subset(product = "MYD11A2",
 lat = 46.12,
 lon = 14.21,
 band = "LST_Day_1km",
 start = "2023-07-01",
 end = "2023-07-31",
 km_lr = 20,
 km_ab = 20,
 site_name = "Sora",
 internal = TRUE,
 progress = FALSE)
# podatke preoblikujemo v format paketa terra
LC_r <- mt_to_terra(df = LC)
# gre za podatke o temperaturi površja v Kelvinih
plot(LC_r)
```



Slika 99: MODIS podatki o temperaturi površja.

```
# izberemo samo prvi podatek, 4. 7. 2023, spremenimo koordinatni sistem
izbran <- project(LC_r[[1]],crspost)
# izrišemo graf, temperatura površja v Kelvinih
plot(izbran)
lines(porecje,col="red") # dodamo mejo porečja
```



Slika 100: MODIS podatki o temperaturi površja na določen dan.

```
# izračunamo povprečno temperaturo površja (v K)
extract(izbran,porecje,mean)

##   ID 2023-07-04
## 1  1   298.0038

# vidimo, da je povprečna temperatura približno 25 stopinj Celzija
```

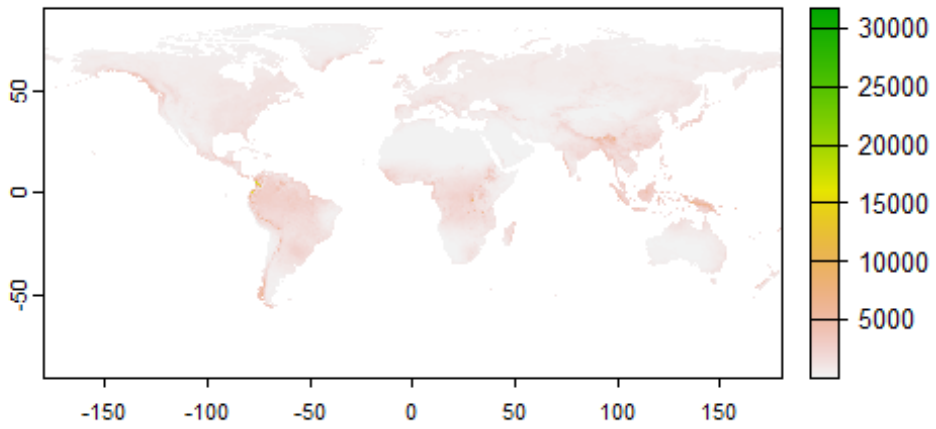
Na podoben način lahko v sklopu programskega okolja R pridobimo tudi nekatere druge prostorske podatke. Pridobitev podatkov o padavinah (pretekle, izmerjene padavine) omogoča tudi paket *pRecipe*. Večje število različnih produktov pa je na voljo tudi na spletni strani Climate Copernicus[135]. V naslednjem primeru bomo prenesli podatke produkta era5, in sicer letne vrednosti, in to samo vrednosti za kopno (argument *domain*). Datoteke bomo shranili v začasni delovni direktorij, katerega vsebino lahko preverimo s funkcijo *list.files(tempdir())*. Uporabili bomo podatke v formatu .nc, ki združuje večje število rastrskih podatkov, in sicer letne količine padavin za obdobje od leta 1959 do 2021.

```
library(pRecipe, quietly=TRUE)
# prenos podatkov
# download_data("era5", tempdir(), timestep = "yearly",domain="land")
# uvoz podatkov
padera5 <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/
era5_tp_mm_land_195901_202112_025_yearly.nc")
# poglejmo osnovne lastnosti prebranih podatkov
padera5

## class       : SpatRaster
## dimensions  : 720, 1440, 63  (nrow, ncol, nlyr)
## resolution  : 0.25, 0.25  (x, y)
```

[135] https://cds.climate.copernicus.eu/cdsapp#!/search?type=dataset.

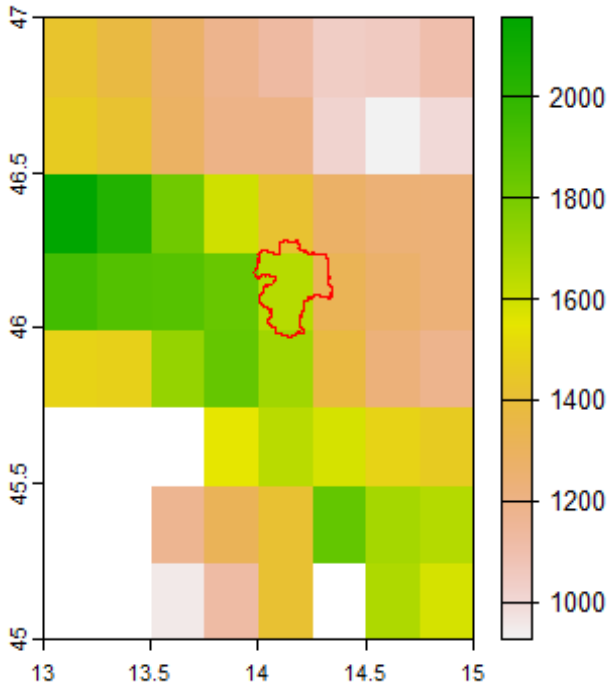
```
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84
## source      : era5_tp_mm_land_195901_202112_025_yearly.nc
## varname     : tp (Total monthly precipitation)
## names       : tp_1, tp_2, tp_3, tp_4, tp_5, tp_6, ...
## unit        :   mm,   mm,   mm,   mm,   mm,   mm, ...
## time (days) : 1959-01-01 to 2021-01-01
```

```
# izrišemo graf za leto 1959 (prvo leto podatkov)
plot(padera5[[1]])
```



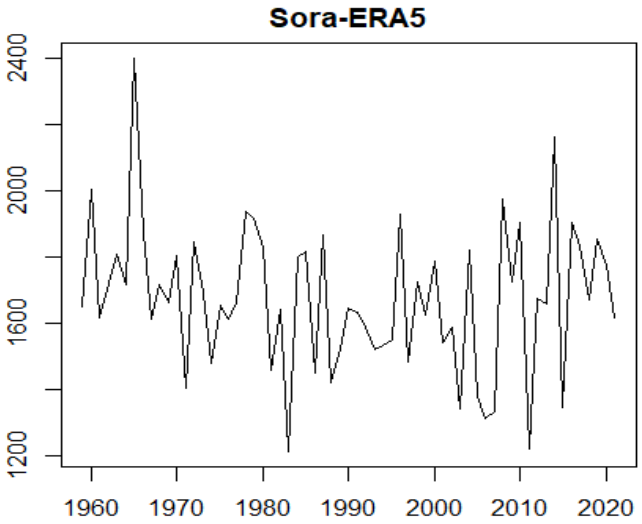
Slika 101: Letna količina padavin 1959, ERA5.

```
# za našo razvodnico spremenimo koordinatni sistem,
# da bo ta enak temu, ki ga imajo podatki ERA5
# uporabimo geografski koordinatni sistem
porecje4 <- project(porecje,crs(padera5))
# izrišemo samo manjše območje naših podatkov
plot(padera5[["tp_1"]],ext=c(13,15,45,47))
# na graf dodamo še mejo porečja
lines(porecje4,col="red")
```



Slika 102: Letne količine padavin, porečje Sore, ERA5.

```
# vidimo, da je resolucija podatkov zelo groba in
# da imamo na območju praktično samo eno celico
# preverimo, katere celice imamo na našem porečju
rez <- extract(padera5[[1]],porecje4,xy=TRUE,touches=TRUE,weights=TRUE)
rez

##   ID     tp_1      x      y      weight
## 1  1 1415.586 14.125 46.375 0.043502964
## 2  1 1839.068 13.875 46.125 0.007647287
## 3  1 1649.994 14.125 46.125 0.799024283
## 4  1 1319.189 14.375 46.125 0.170440718
## 5  1 1696.560 14.125 45.875 0.039453870

# izračunamo količino padavin z upoštevanjem uteži
sum(rez$tp_1*rez$weight)

## [1] 1685.811

# prazen objekt, kamor bomo shranjevali podatke
dummy <- rep(NA,dim(padera5)[3])
# preverimo, kakšne so povprečne padavine na območju
# upoštevamo celico, ki ima največji delež na porečju Sore
for(i in 1:dim(padera5)[3]){
dummy[i] <- as.numeric(extract(padera5[[i]],porecje4)[2])
}
plot(as.Date(time(padera5)),dummy,type="l",xlab="Leto",
  ylab="Letna količina padavin",main="Sora-ERA5")
```



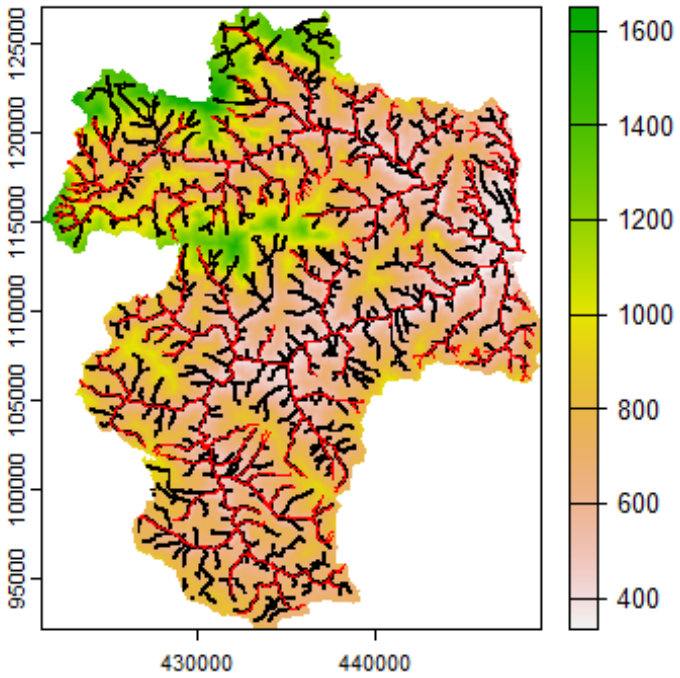
Slika 103: Letne količine padavin na porečju Sore za daljše časovno obdobje.

Pri izrisu grafa smo za pridobitev podatkov o času uporabili funkcijo *time*, uporabne so tudi funkcije *names*, *ext*, *crs*, *nlyr*, *values* itd.

Kot zanimivost pokažimo še primer določitve prispevnega območja na podlagi podatkov o digitalnem modelu višin, kjer bomo uporabili nekatere funkcije, ki so vključene v paket *whitebox*. Določitev prispevnega območja bo vključevala naslednje korake: zapolnitev lokalnih depresij, seštevanje pretoka v računski mreži D8, izračun smeri toka glede na algoritem D8, generiranje rečne mreže ter generiranje prispevne površine glede na izbrano koordinato, kjer bodo uporabljene koordinate vodomerne postaje Železniki na Selški Sori. Paket *whitebox* deluje tako, da ne nalaga podatkov v programsko okolje R, ampak bere in shranjuje datoteke neposredno na disku, torej v izbranem delovnem direktoriju.

```
library(whitebox, quietly=TRUE)
# namestimo tudi program whitebox, brez uporabe znaka #
# whitebox::install_whitebox()
# nastavimo delovno datoteko
setwd("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS")
# zapolnimo lokalne depresije
wbt_breach_depressions_least_cost(dem = "SoraSuha2.tif",
  output = "dem_breached.tif",dist = 500,fill = TRUE)
# zapolnimo depresije v terenu glede na izbrani algoritem
wbt_fill_depressions_wang_and_liu(dem = "dem_breached.tif",
  output = "terenzap.tif")
# izračunamo flow accumulation (seštevanje pretoka)
# glede na algoritem D8 za vse celice
wbt_d8_flow_accumulation(input = "terenzap.tif",output = "terenD8FA.tif")
# izračunamo smer toka glede na algoritem D8
wbt_d8_pointer(dem = "terenzap.tif", output = "terenD8smer.tif")
# na podlagi terena (in algoritma D8 ter flow accumulation)
# generiramo mrežo vodotokov
wbt_extract_streams(flow_accum = "terenD8FA.tif",
  output = "teren_vodotoki.tif",threshold = 20)
```

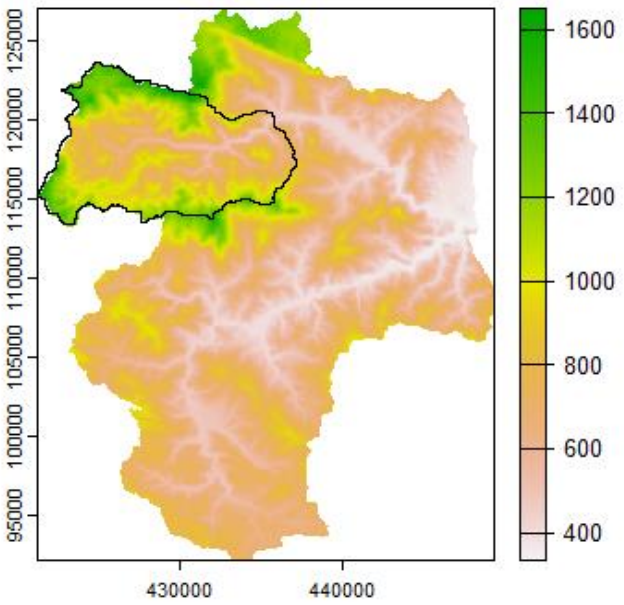
```
# preberemo podatke v okolje R z uporabo paketa terra
teren_vodotoki <- rast("teren_vodotoki.tif")
# preoblikujemo v poligon
teren_vodotoki_pol <- as.polygons(teren_vodotoki)
# primerjamo generirano in dejansko mrežo vodotokov
plot(teren)
lines(teren_vodotoki_pol) # generirana mreža
lines(vodotoki,col="red")  # dejanska mreža
```



Slika 104: Primerjava generirane in dejanske rečne mreže.

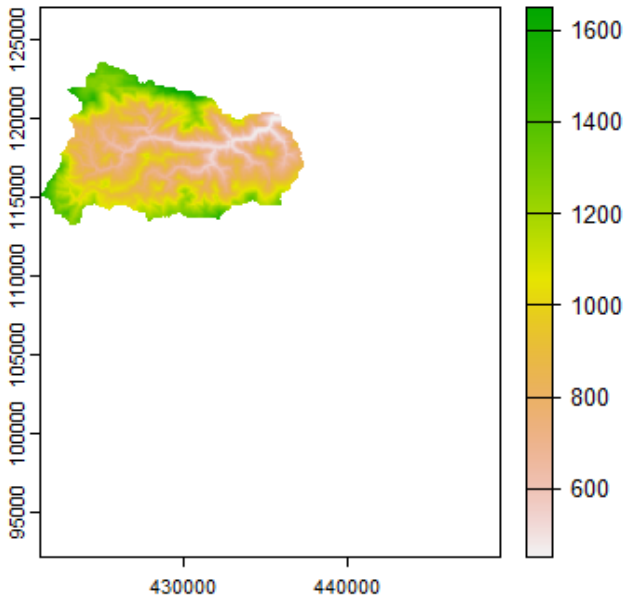
```
# koordinate postaje Železniki
koordinata <- data.frame(x=435710,y=120100)
# preoblikujemo v vektorski sloj
tocka <- SpatialPoints(coords = koordinata, proj4string = CRS(crspost))

## Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_
proj =
## prefer_proj): Discarded datum hermannskogel in Proj4 definition

# shranimo koordinato v mapo v delovnem področju
shapefile(tocka,"tocka.shp",overwrite=TRUE)
# glede na koordinato postaje poiščemo najbližjo točko
# na generirani rečni mreži, snap_dist je razdalja
# v enotah karte (v našem primeru so to metri)
wbt_jenson_snap_pour_points(pour_pts = "tocka.shp",
  streams = "teren_vodotoki.tif",output = "snaprez.shp", snap_dist = 100)
# na podlagi prej določene točke določimo prispevno površino
wbt_watershed(d8_pntr = "terenD8smer.tif",pour_pts = "snaprez.shp",
  output = "rezultat-por.tif")
# uvozimo podatke v okolje R
```

```
zelezniki <- rast("rezultat-por.tif")
# preoblikujemo v poligon
zelezniki_pol <- as.polygons(zelezniki)
plot(teren) # izrišemo graf terena
# dodamo na graf še prispevno območje do postaje Železniki
plot(zelezniki_pol,add=TRUE)
```



Slika 105: Prispevno območje do vodomerne postaje Železniki.

```
# izrišemo porečje Sore do postaje Železniki
plot(mask(teren,zelezniki_pol))
```



Slika 106: Digitalni model gorvodno od postaje Železniki.

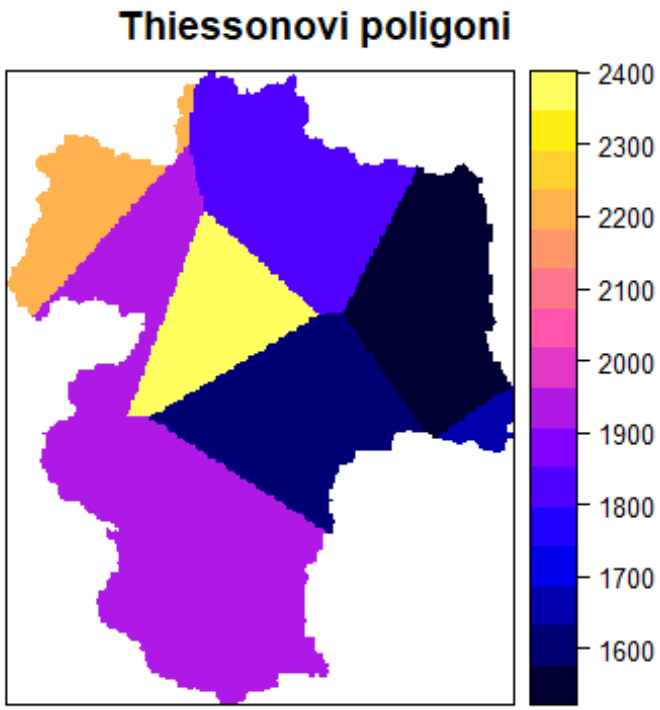
Paket *whitebox* vsebuje tudi številne druge uporabne GIS-funkcije, kot so funkcije za izračun različnih indeksov (npr. *shape indeks* ali pa *sediment transport index*) ter drugih karakteristik. Seznam funkcij je na voljo v opisu paketa *whitebox*. Omeniti velja tudi paket *RSAGA*, ki omogoča uporabo nekaterih funkcij, ki so vključene v program SAGA GIS[136]. Prikažemo lahko še različne metode prostorske interpolacije podatkov. Več teoretičnih informacij je na voljo na primer v knjigi z naslovom An Introduction to Applied Geostatistics[137]. Nekatere metode interpolacije bomo prikazali z uporabo funkcij, ki so vključene v paketa *sp* in *raster*. Tudi paket *terra* vključuje nekatere funkcije za interpolacijo, ki pa se nekoliko razlikujejo od funkcij, prikazanih v nadaljevanju. Prikazana bo interpolacija z uporabo različnih metod (npr. Kriging, Thiessonovi poligoni).

```
library(sp, quietly=TRUE)
library(gstat, quietly=TRUE)
library(raster, quietly=TRUE)
# uvozimo še enkrat podatke o terenu z uporabo paketa raster
teren1 <- raster("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS
/SoraSuha.sdat")

## Warning in showSRID(SRS_string, format = "PROJ", multiline = "NO", prefer_
proj =
## prefer_proj): Discarded datum Militar-Geographische Institut in Proj4 defi
nition

# k atributnim podatkom dodamo še podatek o nadmorski višini
postaje$teren1 <- teren[cellFromXY(teren, data.frame(postaje[,2:3]))][,1]
# preoblikujemo teren v obliko, ki jo bomo uporabili za interpolacijo
grid <- as(teren1, 'SpatialPixelsDataFrame')
# naredimo podatkovni okvir
postajexy <- data.frame(pad=postaje$padavine,SoraSuha=postaje$teren1,x=geom(p
ostaje)[,3],y=geom(postaje)[,4])
# preoblikujemo v format SpatialPointsDataFrame
coordinates(postajexy) <- ~x+y
# definiramo koordinatni sistem
crs(postajexy) <- crs(teren1)
# uporaba metode Thiessenovih poligonov (idp=1 in nmax=1),
# uporaba drugih argumentov idp in nmax omogoča tudi
# interpolacijo z uporabo metode najbližjega soseda
prec_nn <- idw(pad ~ 1, postajexy, grid, idp=1, nmax=1)

## [inverse distance weighted interpolation]

# preoblikujemo v obliko rastrskih podatkov
prec_nn <- as(prec_nn, 'RasterLayer')
```

[136] https://saga-gis.sourceforge.io/.

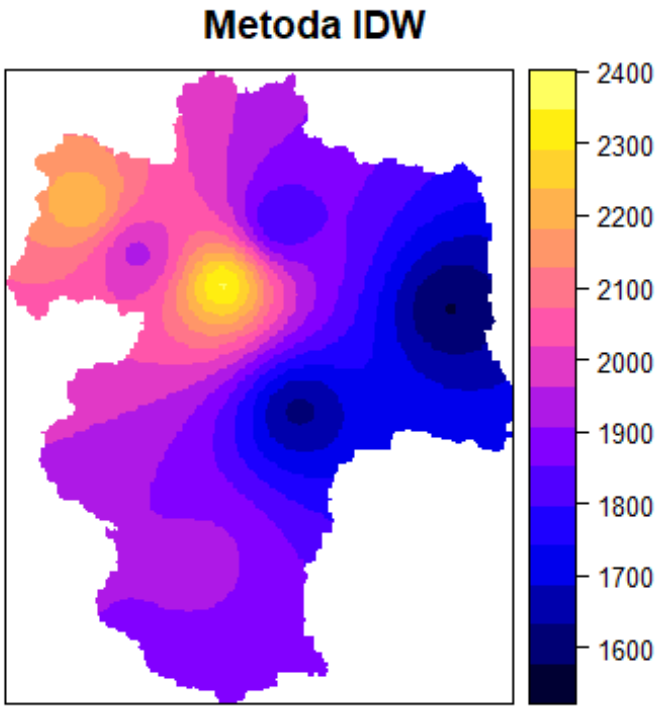
[137] https://www.amazon.com/Introduction-Applied-Geostatistics-Edward-Isaaks/dp/0195050134.

```
# izrišemo graf
spplot(prec_nn, main = "Thiessonovi poligoni")
```



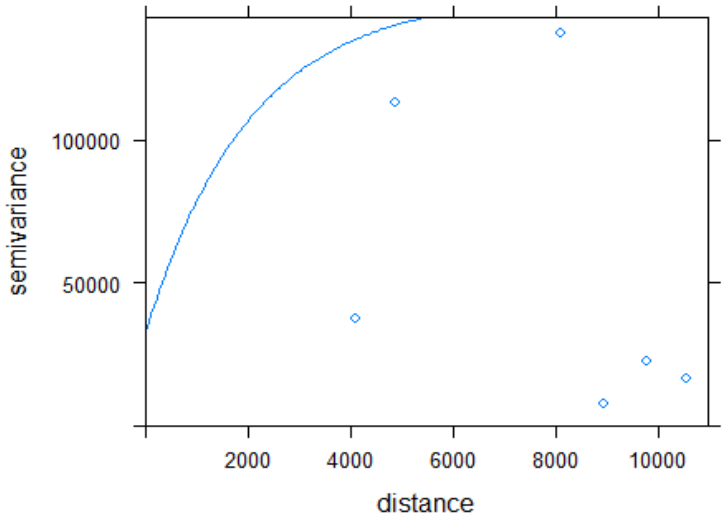
Slika 107: Thiessonovi poligoni.

```
# metoda IDW, argument idp je utež
prec_idw <- idw(pad ~ 1, postajexy, grid, idp = 2)

## [inverse distance weighted interpolation]

# preoblikujemo v raster
prec_idw <- as(prec_idw, 'RasterLayer')
# izrišemo graf
spplot(prec_idw, main = "Metoda IDW")
```



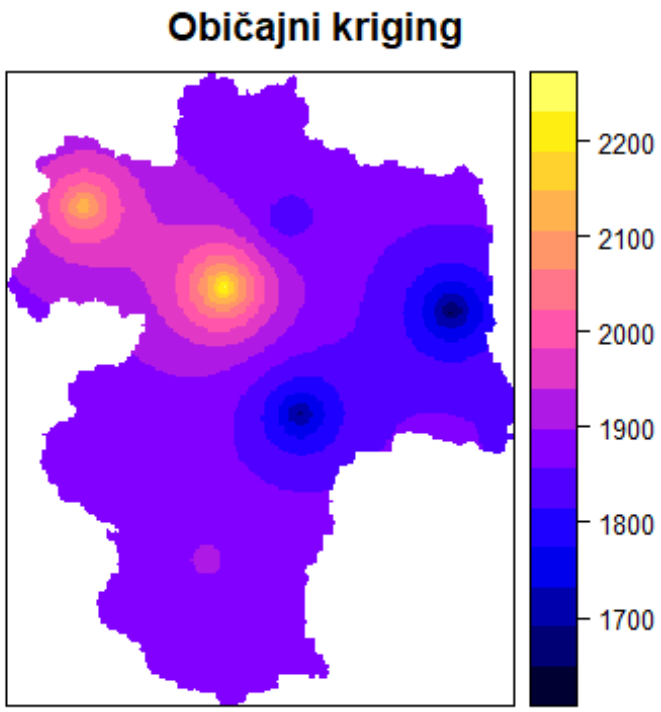
Slika 108: Metoda IDW.

```
# prikažemo še uporabo kriginga
# definiramo empirični variogram
emperical.var <- variogram(pad~1,postajexy)
# ocenimo teoretični variogram, kjer
# preverimo eksponentno in sferično funkcijo
theoretical.var <- fit.variogram(emperical.var, model=vgm("Exp", "Sph"),
  fit.sills = T, fit.ranges = T)

## Warning in fit.variogram(emperical.var, model = vgm("Exp", "Sph"), fit.sil
ls =
## T, : No convergence after 200 iterations: try different initial values?

# izrišemo oba variograma
plot(emperical.var,theoretical.var)
```



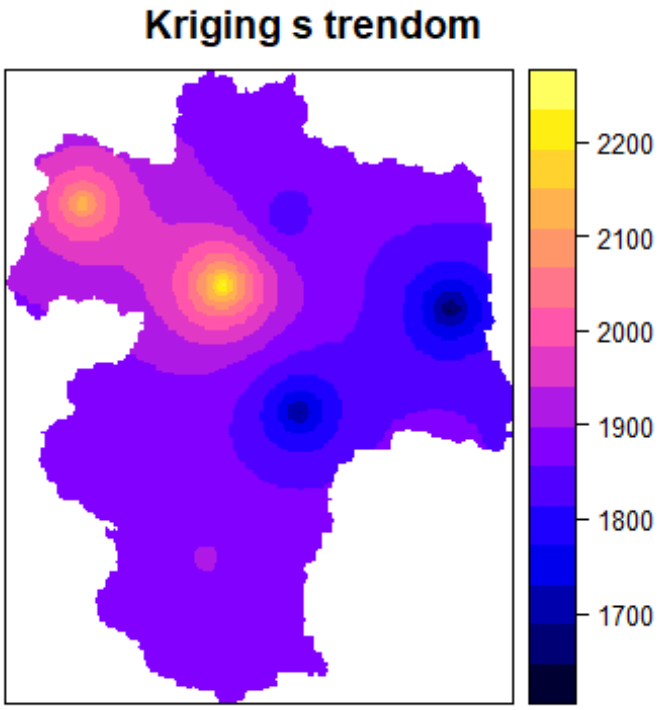
Slika 109: Variogram.

```
# vidimo, da rezultati niso najboljši, kar je rezultat
# relativno velike prostorske spremenljivosti padavin
# kljub temu prikažemo primer uporabe običajnega kriginga
prec_ok <- krige(pad ~ 1, postajexy, grid, model = theoretical.var,
  nmax = 40, nmin = 2, maxdist = 100e3)

## [using ordinary kriging]

# preoblikujemo v rastrski format
prec_ok <- as(prec_ok, 'RasterLayer')
# izrišemo graf
spplot(prec_ok, main = "Običajni kriging")
```



Slika 110: Običajni kriging.

```
# Kriging s trendom (nadmorska višina)
prec_edk <- krige(pad ~ SoraSuha, postajexy, grid, model = theoretical.var,
  nmax = 40, nmin = 2, maxdist = 100e3)

## [using universal kriging]

# preoblikujemo v raster
prec_edk <- as(prec_edk, 'RasterLayer')
 # izrišemo graf
spplot(prec_edk, main = "Kriging s trendom")
```



Slika 111: Kriging s trendom.

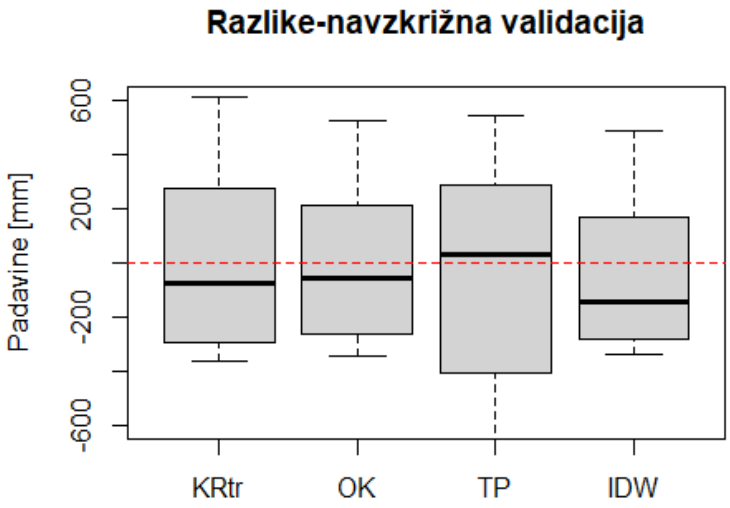
Opazimo lahko, da rezultati interpolacije verjetno ne bodo najboljši, kar je posledica relativno velike prostorske spremenljivosti na majhnem območju oziroma (pre)majhnega števila padavinskih postaj. Prikažemo še primer navzkrižne validacije (vsakič izpustimo eno postajo, ponovimo postopek prostorske interpolacije in izračunamo razliko med dejansko in ocenjeno količino padavin) za prej prikazane metode prostorske interpolacije.

```
# argument nfold je število postaj, najprej za običajni kriging
locv_ok <- krige.cv(pad~1, postajexy, theoretical.var, nmax = 40,
  nmin = 2, maxdist = 100e3, nfold = 8)
# rezultati
head(locv_ok)
```

```
##   var1.pred var1.var observed   residual     zscore fold
## 1  1906.335 174230.7     1684 -222.33513 -0.5326547    1
## 2  1921.339 173688.0     1575 -346.33923 -0.8310302    2
## 3  1925.517 169022.5     1803 -122.51719 -0.2980058    3
## 4  1823.621 165595.9     2347  523.37863  1.2861489    4
## 5  1920.963 172823.3     1620 -300.96295 -0.7239556    5
## 6  1870.298 174062.8     1917   46.70233  0.1119401    6
```

```
# vidimo lahko razlike med dejanskimi ocenjenimi padavinami
# Kriging s trendom
locv_edk <- krige.cv(pad~SoraSuha, postajexy, theoretical.var, nmax = 40,
  nmin = 2, maxdist = 100e3, nfold = 8)
# Thiessonovi poligoni
locv_nn <- krige.cv(pad~1, postajexy,  model = theoretical.var, nmin = 1,
  nmax = 1, maxdist = 100e3, nfold = 8)
```

```
# metoda IDW
locv_idw <- krige.cv(pad~1, postajexy, nfold = 8)
# združimo podatke
razlike <- cbind(KRtr = locv_edk$residual, OK = locv_ok$residual,
  TP = locv_nn$residual, IDW = locv_idw$residual)
# izrišemo okvir z ročaji
boxplot(razlike , main = "Razlike-navzkrižna validacija",
  ylim = c(-600, 600),ylab="Padavine [mm]")
# dodamo še horizontalno črto
abline(h = 0, col = "red", lty = 2)
```



Slika 112: Rezultati navzkrižne validacije.

Kot omenjeno, rezultati interpolacije niso bili najboljši.

Naloga 41: Na podlagi podatkov o digitalnem modelu višin določite karto naklona z uporabo funkcije terrain v paketu terra. Izračunajte osnovno statistiko naklonov.

4.16 Modeliranje površinskega odtoka

Modeliranje površinskega odtoka je eden izmed bolj pogosto uporabljenih praktičnih izzivov na področju vodarstva oziroma hidrologije. Poznamo različne modele (npr. konceptualne, empirične, fizikalno osnovane), s katerimi lahko modeliramo transformacijo padavin v površinski odtok. Eden izmed takšnih modelov je tudi model GR4J (modèle du Génie Rural à 4 paramètres Journalier)[138] in njegove izboljšave. Opis zgodovine in razvoja modela (z

[138] https://webgr.inrae.fr/models/daily-hydrological-model-gr4j/description-of-the-gr4j-model/.

ustreznimi referencami) je na voljo na spletni strani raziskovalnega inštituta INRAE[139]. Model GR4J je enovit konceptualni hidrološki model, ki na podlagi podatkov o padavinah in potencialni evapotranspiraciji omogoča simulacije pretokov. Na voljo je tudi različica v programu Excel (z omejenimi zmožnostmi). Model GR4J uporablja štiri parametre (X1, X2, X3 in X4), vključen je v paketa *airGR* in *airgrteaching*. V omenjenih paketih so na voljo tudi funkcije za nadgrajene različice hidroloških modelov (npr. GR5J in GR6J), snežni modul CemaNeige, funkcije za umerjanje modela itd.[140]. Na voljo so tudi paketi *airGRdatasim* za podatkovno asimilacijo in pa *airGRiwrm* za celostno upravljanje vodnih virov[141]. V nadaljevanju bo prikazan postopek uporabe programa GR4J in nekaterih izboljšanih različic na podlagi podatkov, ki so bili uporabljeni v sklopu uporabe modela za napoved proženja plitvih plazov na porečju Sore[142]. Podatki, ki jih bomo uporabili, bodo vključevali podatke o pretokih (m^3/s in mm postaja Železniki), padavinah (postaja Davča) in temperaturi zraka (Bohinjska Češnjica). Na podlagi podatkov o temperaturi zraka bomo izračunali tudi vrednosti potencialne evapotranspiracije. Spodaj bo najprej prikazan postopek umerjanja modela GR4J za obdobje 2005–2010, prvo leto podatkov bo uporabljeno za ogrevanje modela.

```
library(airGR, quietly=TRUE)
data <- read.table("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/airGR/Data-Example.txt",header=T)
head(data) # pogledamo prvih nekaj vrstic podatkov

##       Datum Q_m3s Q_mm P_mm  T_C
## 1 1.01.2005 5.711 4.74    0 -2.6
## 2 2.01.2005 5.452 4.53    0 -1.0
## 3 3.01.2005 4.696 3.90    0 -2.7
## 4 4.01.2005 4.451 3.69    0 -2.8
## 5 5.01.2005 4.209 3.49    0 -2.1
## 6 6.01.2005 4.209 3.49    0 -2.0

summary(data) # osnovna statistika podatkov

##     Datum               Q_m3s             Q_mm             P_mm
##  Length:4383        Min.   : 0.402   Min.   : 0.330   Min.   :  0.000
##  Class :character   1st Qu.: 1.405   1st Qu.: 1.180   1st Qu.:  0.000
##  Mode  :character   Median : 2.345   Median : 2.010   Median :  0.000
##                     Mean   : 4.028   Mean   : 3.417   Mean   :  5.168
##                     3rd Qu.: 4.447   3rd Qu.: 3.840   3rd Qu.:  3.900
##                     Max.   :76.227   Max.   :63.270   Max.   :227.900
##       T_C
```

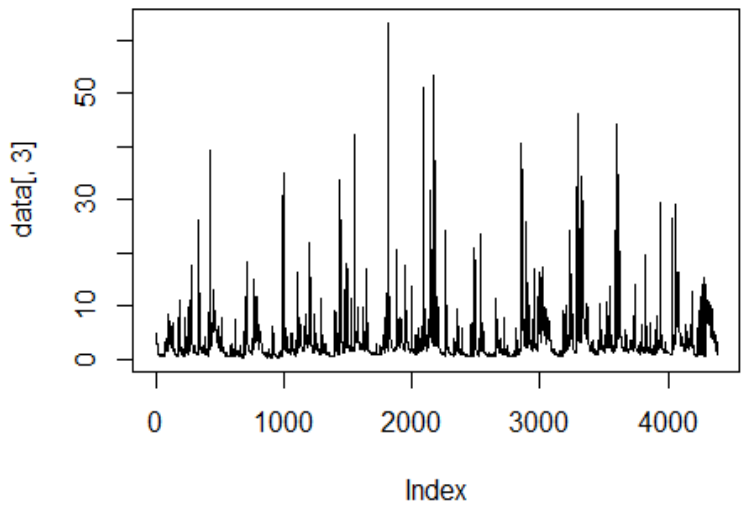
[139] https://webgr.inrae.fr/models/a-brief-history/.

[140] https://hydrogr.github.io/airGR/index.html.

[141] https://airgriwrm.g-eau.fr/index.html.

[142] https://link.springer.com/article/10.1007/s10346-019-01169-9.

```
##  Min.   :-13.400
##  1st Qu.:  2.200
##  Median :  9.500
##  Mean   :  8.884
##  3rd Qu.: 15.300
##  Max.   : 27.300

plot(data[,3],type="l") # izrišemo graf podatkov o pretokih
```



Slika 113: Merjeni podatki o pretokih.

```
# izračunamo potencialno evapotranspiracjo z uporabo enačbe Oudin
# na podlagi podatkov o temperaturi zraka in lokaciji postaje
potET <- PE_Oudin(JD = as.POSIXlt(strptime(data[,1], "%d.%m.%Y"))$yday,
  Temp = data[,5], Lat = 46.2, LatUnit = "deg")
summary(potET) # pogledamo izračune

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.433   1.500   1.805   3.069   5.175

# definiramo vhodne podatke (padavine, potencialna evapotranspiracija)
# ter obdobje modeliranja in model, ki ga bomo uporabili (GR4J)
InputsModel <- CreateInputsModel(FUN_MOD = RunModel_GR4J, DatesR = as.POSIXct
(strptime(data[,1], "%d.%m.%Y")), Precip = data[,4], PotEvap = potET)
# poiščemo točko v podatkih, ko se leto 2010 zaključi
konec <- which(format(data[,1], format = "%d.%m.%Y")=="31.12.2010")
# definiramo zaporedje, prvo leto (2005) bo uporabljeno
# za ogrevanje modela
Ind_Run <- seq(366, konec)
# definiramo nastavitve modela, obdobje ogrevanja in umerjanja
RunOptions <- CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = InputsM
odel, IndPeriod_Run = Ind_Run, IndPeriod_WarmUp = 1:365)
# izberemo kriterij za kontrolo ustreznosti modela
# (kriterij Nash-Sutcliffe) ter izberemo podatke o pretokih
InputsCrit <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = Inputs
```

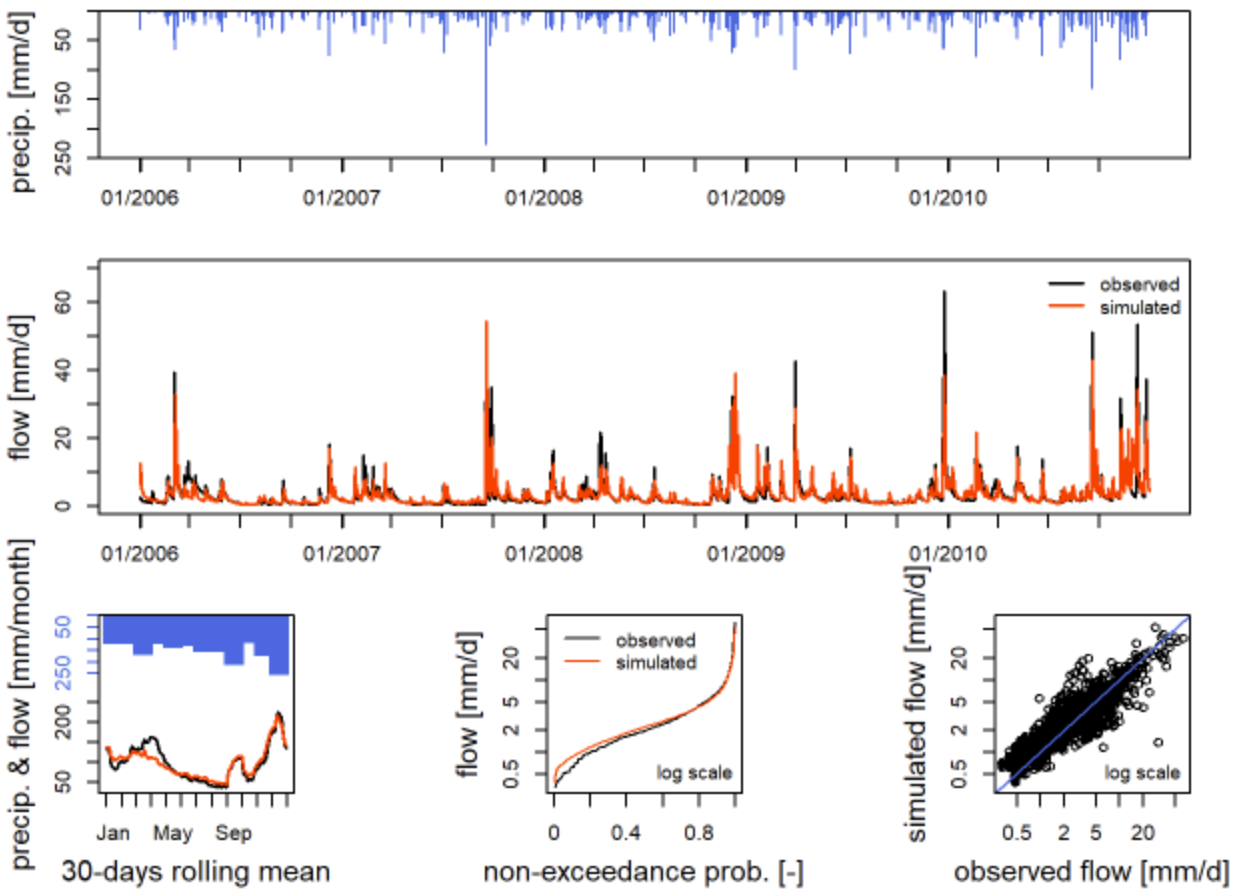
```
Model, RunOptions = RunOptions, Obs = data[366:konec,3])
# izberemo algoritem za umerjanje modela (algoritem Michel)
CalibOptions <- CreateCalibOptions(FUN_MOD = RunModel_GR4J, FUN_CALIB = Calib
ration_Michel)
# umerjanje modela glede na zgoraj definirane nastavitve
OutputsCalib <- Calibration_Michel(InputsModel = InputsModel, RunOptions = Ru
nOptions, InputsCrit = InputsCrit, CalibOptions = CalibOptions, FUN_MOD = Run
Model_GR4J)
```

```
## Grid-Screening in progress (0% 20% 40% 60% 80% 100%)
##   Screening completed (81 runs)
##       Param =  432.681,   -2.376,   83.096,    1.417
##       Crit. NSE[Q]       = 0.4598
## Steepest-descent local search in progress
##   Calibration completed (27 iterations, 261 runs)
##       Param =  677.881,   -0.621,  118.217,    0.500
##       Crit. NSE[Q]       = 0.7363
```

```
# shranimo parametre, ki smo jih določili v postopku umerjanja
ParamGR4J <- OutputsCalib$ParamFinalR
ParamGR4J # poglejmo umerjene vrednosti parametrov
```

```
## [1] 677.8810135  -0.6208393 118.2173963   0.5000000
```

```
# zaženemo model z umerjenimi parametri (za obdobje 2005-2010)
OutputsModel <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOpti
ons, Param = ParamGR4J)
# izrišemo primerjavo med merjenimi in modeliranimi podatki
plot(OutputsModel, data[366:konec,3])
```



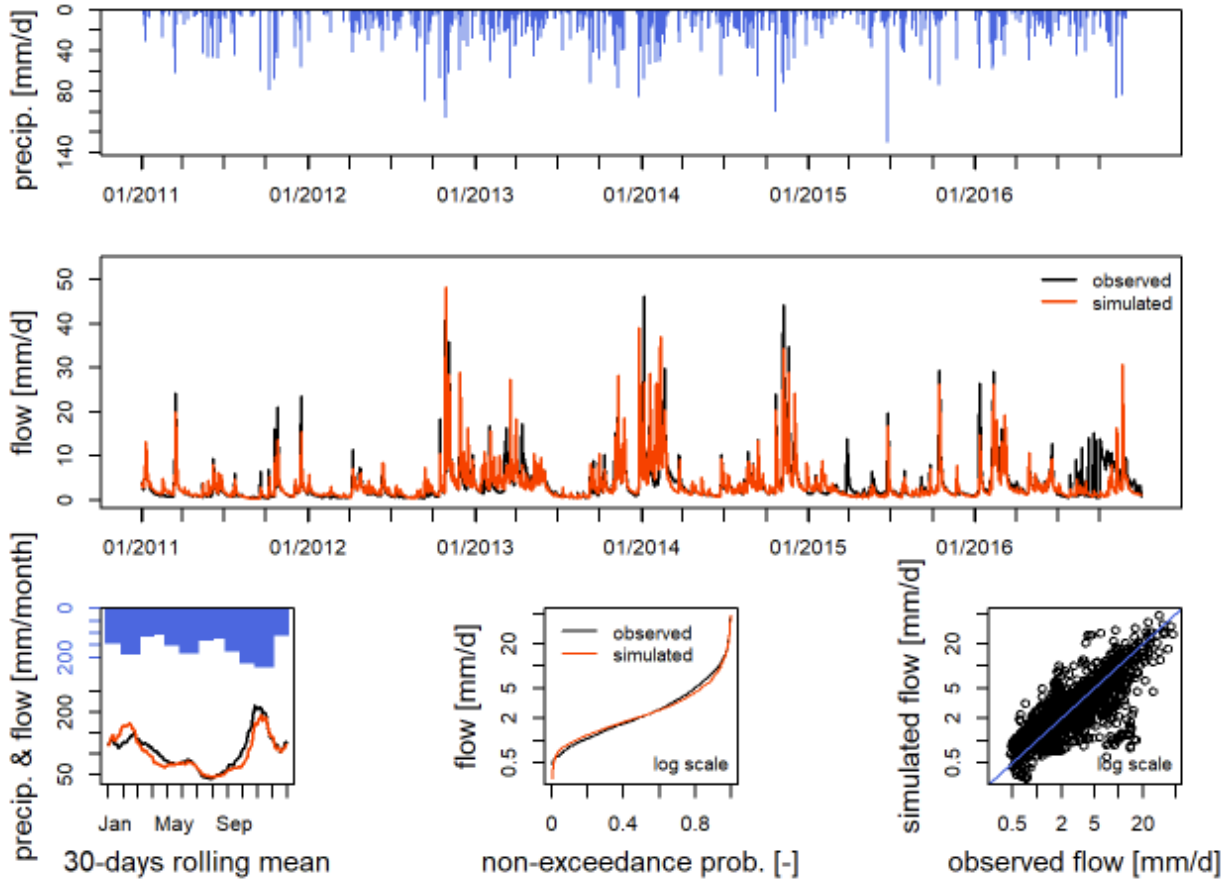
Slika 114: Rezultati umerjanja modela GR4J.

Sedaj se lotimo še validacije hidrološkega modela, in sicer za obdobje 2011–2016. Definirali bomo zaporedje podatkov, ki jih bomo uporabili za validacijo, zagnali model z umerjenimi parametri za obdobje validacije in preverili ujemanje v obdobju validacije. Primerjava bo narejena tako grafično kot na podlagi numeričnega kriterija ustreznosti testiranega hidrološkega modela padavine–odtok (NSE)[143]. Optimalna vrednost kriterija NSE bi bila 1, vrednost 0,61 pa predstavlja relativno dobro ujemanje med merjenimi in modeliranimi podatki.

```
Ind_Run1 <- seq((konec+1),dim(data)[1])
RunOptions1 <- CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = Inputs
Model, IndPeriod_Run = Ind_Run1)

## Warning in CreateRunOptions(FUN_MOD = RunModel_GR4J, InputsModel = InputsM
odel, : model warm up period not defined: default configuration used
##   the year preceding the run period is used
```

[143] https://doi.org/10.15292/acta.hydro.2018.06.

```
OutputsModel1 <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOpt
ions1, Param = ParamGR4J)
plot(OutputsModel1, Qobs = data[Ind_Run1,3])
```



Slika 115: Rezultati validacije modela GR4J.

```
InputsCrit1  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = Inpu
tsModel,RunOptions = RunOptions1, Obs = data[Ind_Run1,3])
OutputsCrit1 <- ErrorCrit_NSE(InputsCrit = InputsCrit1, OutputsModel = Output
sModel1)

## Crit. NSE[Q] = 0.6102

OutputsCrit1

## $CritValue
## [1] 0.610208
##
## $CritName
## [1] "NSE[Q]"
##
## $CritBestValue
## [1] 1
##
```

```
## $Multiplier
## [1] -1
##
## $Ind_notcomputed
## NULL
##
## attr(,"class")
## [1] "NSE"       "ErrorCrit"
```

Paket *airGRteaching* omogoča tudi interaktivno uporabo hidrološkega modela GR4J (in ostalih modelov, ki so vključeni v paket *airGR*). Paket *airGRteaching* zahteva nekoliko drugačno strukturo podatkov.

```
library(airGRteaching, quietly=TRUE)
preob <- data.frame(DatesR=as.POSIXct(strptime(data[,1], "%d.%m.%Y"),tz="UTC"
),P=data[,4], E=potET,Qmm=data[,3])
# zaženemo aplikacijo Shiny, ki se odpre v brskalniku
ShinyGR(ObsDF = preob, SimPer = c("2005-01-01", "2010-12-31"))
```

Kot smo lahko videli, model GR4J ne uspe ustrezno simulirati pretokov v zimskem času in ima določene težave z ocenjevanjem nizkih pretokov. Model GR6J je bil razvit s ciljem boljšega simuliranja nizkih pretokov[144]. Za primer porečja Selške Sore do vodomerne postaje Železniki pokažimo še primer uporabe modela GR6J z vključenim snežnim modulom CemaNeige. Snežni modul kot vhodni podatek potrebuje še podatke o temperaturi zraka ter hipsometrični krivulji porečja. Postopek uporabe modela CemaNeige GR6J je podoben kot v primeru modela GR4J. Dodatno je treba definirati še argument *MeanAnSolidPrecip*, ki vključuje delež snežnih padavin (za pet višinskih con) in je bil ocenjen glede na enačbo, podano v prispevku Alexopoulos in sodelavci (2023)[145]. Pri oceni tega parametra se svetuje previdnost.

```
library(terra, quietly=TRUE)
# aktiviramo podatke o porečju L0123001, ki so vključeni v paket airGR
data(L0123001)
# oblika podatkov o hipsometrični krivulji
# dodatno je format zapisan tudi v BasinInfo
BasinInfo
```

```
## $BasinCode
## [1] "L0123001"
##
## $BasinName
## [1] "Blue River at Nourlangie Rock"
##
## $BasinArea
## [1] 360
```

[144] https://doi.org/10.3390/w12010128.

[145] https://doi.org/10.5194/hess-27-2559-2023.

```
##
## $HypsoData
##   [1]  286  309  320  327  333  338  342  347  351  356  360  365  369  37
3  378
##  [16]  382  387  393  399  405  411  417  423  428  434  439  443  448  45
3  458
##  [31]  463  469  474  480  485  491  496  501  507  513  519  524  530  53
6  542
##  [46]  548  554  560  566  571  577  583  590  596  603  609  615  622  62
9  636
##  [61]  642  649  656  663  669  677  684  691  698  706  714  722  730  73
8  746
##  [76]  754  762  770  777  786  797  808  819  829  841  852  863  875  88
7  901
##  [91]  916  934  952  972  994 1012 1029 1054 1080 1125 1278
```

```
teren <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/airGR/
terrain-Sora.sdat")
# izračunamo hipsometrično krivuljo
hypso <- c(min(as.numeric(values(teren)),na.rm=T),as.numeric(quantile(as.nume
ric(values(teren)),probs=seq(from=0.01,by=0.01,to=0.99),na.rm=T)),max(as.nume
ric(values(teren)),na.rm=T))
# definiramo nastavitve modela, število višinskih
# con za snežni modul, hipsometrično krivljo
InputsModel2 <- CreateInputsModel(FUN_MOD = RunModel_CemaNeigeGR6J, as.POSIXc
t(strptime(data[,1], "%d.%m.%Y")), Precip = data[,4], PotEvap = potET, TempMe
an = data[,5],  ZInputs = median(hypso), HypsoData = hypso, NLayers = 5)
```

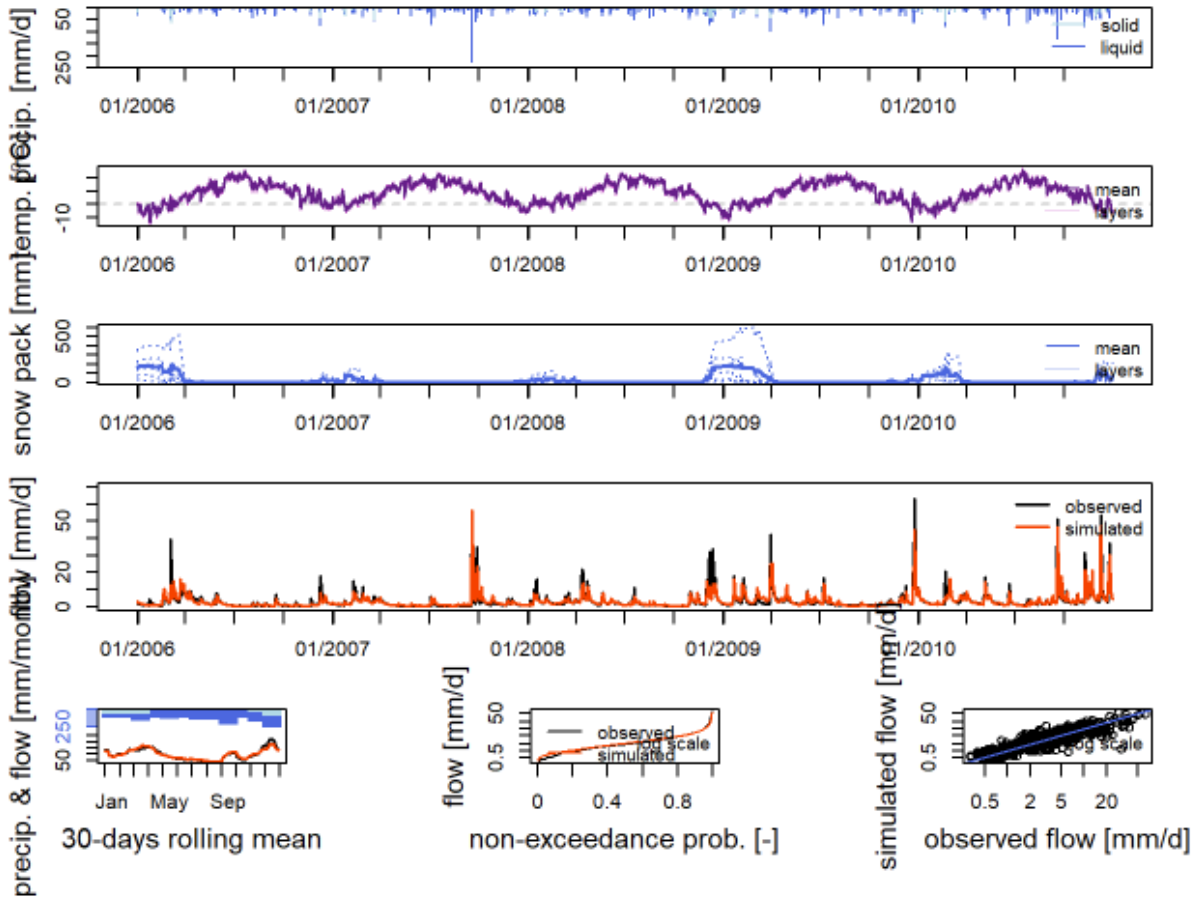
```
## input series were successfully created on 5 elevation layers for use by Ce
maNeige
```

```
RunOptions2 <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR6J, InputsModel
= InputsModel2, IndPeriod_Run = Ind_Run, IndPeriod_WarmUp = 1:365, MeanAnSoli
dPrecip=c(70,130,180,250,350))
# definiramo kriterij ustreznosti in podatke o pretoku
InputsCrit2 <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = Input
sModel2, RunOptions = RunOptions2, Obs = data[366:konec,3])
# način umerjanja modela
CalibOptions2 <- CreateCalibOptions(FUN_MOD = RunModel_CemaNeigeGR6J, FUN_CAL
IB = Calibration_Michel)
# zaženemo umerjanje modela, postopek traja dlje kot v primeru GR4J
OutputsCalib2 <- Calibration_Michel(InputsModel = InputsModel2, RunOptions =
RunOptions2, InputsCrit = InputsCrit2, CalibOptions = CalibOptions2, FUN_MOD
= RunModel_CemaNeigeGR6J)
```

```
## Grid-Screening in progress (
```

```
## 0% 20% 40% 60% 80% 100%)
##   Screening completed (6561 runs)
##       Param =   90.017,   -0.521,   27.113,    1.369,    0.220,   54.598,
0.002,    6.764
```

```
##       Crit. NSE[Q]       = 0.5421
## Steepest-descent local search in progress
##   Calibration completed (45 iterations, 7241 runs)
##       Param =  395.412,   -0.174,   61.234,    0.755,    0.113,   15.810,
0.026,   17.247
##       Crit. NSE[Q]       = 0.7401

# shranimo umerjene vrednosti parametrov
ParamGR6JCemaNeige <- OutputsCalib2$ParamFinalR
# zaženemo model z umerjenimi vrednostmi parametrov
OutputsModel2 <- RunModel_CemaNeigeGR6J(InputsModel = InputsModel2, RunOption
s = RunOptions2, Param = ParamGR6JCemaNeige)
# izrišemo graf
plot(OutputsModel2, data[366:konec,3])
```



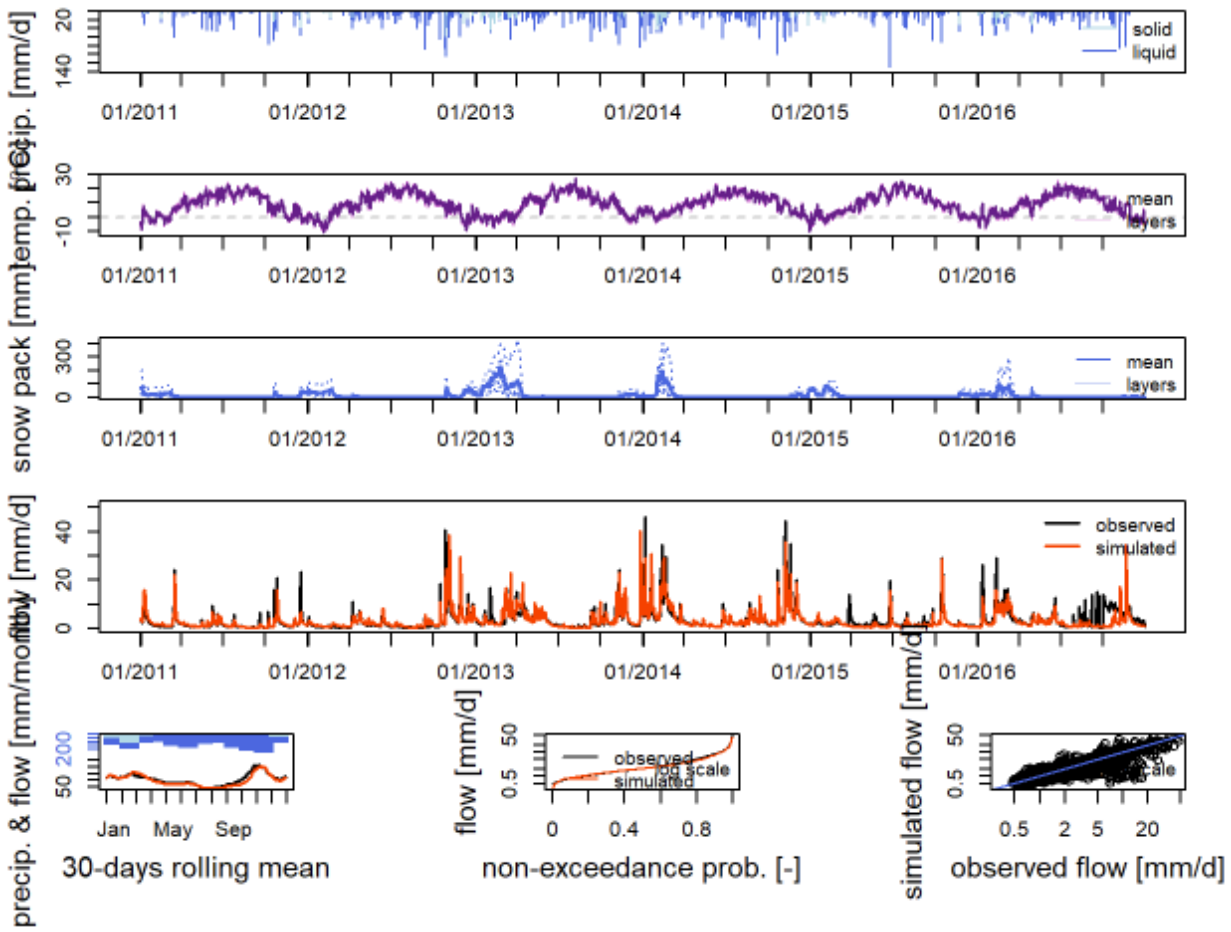
Slika 116: Rezultati umerjanja modela GR6J s snežnim modulom.

```
# izračunamo kriterij NSE
ErrorCrit_NSE(InputsCrit = InputsCrit2, OutputsModel = OutputsModel2)

## Crit. NSE[Q] = 0.7401
```

```
## $CritValue
## [1] 0.7400699
##
## $CritName
## [1] "NSE[Q]"
##
## $CritBestValue
## [1] 1
##
## $Multiplier
## [1] -1
##
## $Ind_notcomputed
## NULL
##
## attr(,"class")
## [1] "NSE"       "ErrorCrit"
```

Izračunajmo simulirane vrednosti pretokov še za obdobje validacije. Vidimo, da smo z uporabo modela GR6J s snežnim modulom uspeli dobiti boljše ujemanje med modeliranimi in merjenimi pretoki kot zgolj z uporabo modela GR4J brez snežnega modula.

```
RunOptions3 <- CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR6J, InputsModel
= InputsModel2, IndPeriod_Run = Ind_Run1)

## Warning in CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR6J, InputsModel
= InputsModel2, : model warm up period not defined: default configuration use
d
##   the year preceding the run period is used

## Warning in CreateRunOptions(FUN_MOD = RunModel_CemaNeigeGR6J, InputsModel
= InputsModel2, : 'MeanAnSolidPrecip' not defined: it was automatically set t
o c(328,328,328,328,328)  from the 'InputsModel' given to the function. Be ca
reful in case your application is (short-term) forecasting.

OutputsModel3 <- RunModel_CemaNeigeGR6J(InputsModel = InputsModel2, RunOption
s = RunOptions3, Param = ParamGR6JCemaNeige)
plot(OutputsModel3, Qobs = data[Ind_Run1,3])
```



Slika 117: Rezultati validacije modela GR6J s snežnim modulom.

```
InputsCrit3  <- CreateInputsCrit(FUN_CRIT = ErrorCrit_NSE, InputsModel = Inpu
tsModel2,RunOptions = RunOptions3, Obs = data[Ind_Run1,3])
OutputsCrit3 <- ErrorCrit_NSE(InputsCrit = InputsCrit3, OutputsModel = Output
sModel3)

## Crit. NSE[Q] = 0.6618

OutputsCrit3

## $CritValue
## [1] 0.6618097
##
## $CritName
## [1] "NSE[Q]"
##
## $CritBestValue
## [1] 1
##
## $Multiplier
## [1] -1
##
```

```
## $Ind_notcomputed
## NULL
##
## attr(,"class")
## [1] "NSE"       "ErrorCrit"
```

Na enak način kot v primeru modela GR4J lahko zaženemo tudi aplikacijo Shiny.

```
preob1 <- data.frame(DatesR=as.POSIXct(strptime(data[,1], "%d.%m.%Y"),
  tz="UTC"),P=data[,4], E=potET,Qmm=data[,3],T=data[,5])
ShinyGR(ObsDF = preob1, SimPer = c("2005-01-01", "2010-12-31"),
  ZInputs = median(hypso),HypsoData = hypso, NLayers = 5)
```

Paket *airGR* pa vključuje tudi model GR4H, ki omogoča simulacije pretokov z urnim časovnim korakom. V povezavi s tem modelom velja omeniti tudi to, da recimo funkcija *PE_Oudin* omogoča disagregacijo podatkov o potencialni evapotranspiraciji (v urni časovni korak). V spodnjem primeru bomo izračunali potencialno evapotranspiracijo z urnim časovnim korakom in preverili osnovno statistiko urnih podatkov.

```
potETh <- PE_Oudin(JD = as.POSIXlt(strptime(data[,1], "%d.%m.%Y"))$yday,
  Temp = data[,5], Lat = 46.2, LatUnit = "deg",TimeStepIn = "daily",TimeStepO
ut = "hourly")
summary(potETh)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.07519 0.10802 0.60550
```

Na voljo so tudi drugi paketi, ki omogočajo izvedbo podobnih hidroloških simulacij. Pregled nekaterih so v znanstvenem članku pripravili Astagneau in sodelavci[146]. Eden izmed teh paketov je tudi *TUWmodel*, ki omogoča uporabo modela TUWmodel[147]. Gre za model, ki je po strukturi relativno podoben hidrološkemu modelu HBV in omogoča simulacije pretokov[148]. V naslednjem primeru bo prikazana uporaba modela TUWmodel, ki bo sledil gradivu prof. Parajke[149]. Najprej bomo model zagnali z naključnimi vrednostmi parametrov, nato pa bomo izvedli še umerjanje modela z uporabo paketa *DEoptim*. Več informacij o parametrih je na voljo v pomoči funkcije *TUWmodel*. Aktivirali bomo tudi paket *hydroGOF*, ki vključuje nekatere funkcije za oceno ustreznosti modela.

```
library(TUWmodel, quietly=TRUE)
preob1 <- data.frame(DatesR=as.POSIXct(strptime(data[,1], "%d.%m.%Y"),
  tz="UTC"),P=data[,4], E=potET,Qmm=data[,3],T=data[,5])
# zaženemo model z naključno izbranimi vrednostmi parametrov
sim1 <- TUWmodel(prec=preob1[,2], airt=preob1[,5], ep=preob1[,3], area=1.,
```

---

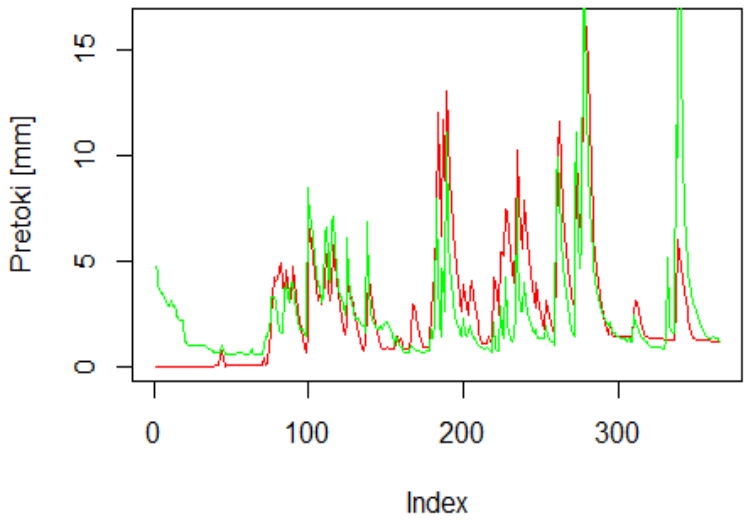
[146] https://doi.org/10.5194/hess-25-3937-2021.

[147] https://doi.org/10.5194/hess-19-2101-2015.

[148] https://doi.org/10.5194/hess-20-2085-2016.

[149] https://info.chmi.cz/konference/danube2021/workshop.php.

```
   param=c(1.3, 2.0, -1.0, 1.0, 0.0, 0.8, 360.0, 0.2, 0.3, 7.0, 150.0, 50.0,
2.0, 10.0, 25.0),
   incon=c(50,0,2.5,2.5))
# preverimo strukturo rezultatov
str(sim1)

## List of 22
##  $ itsteps: int 4383
##  $ nzones : int 1
##  $ area   : num 1
##  $ param  : num [1, 1:15] 1.3 2 -1 1 0 0.8 360 0.2 0.3 7 ...
##   ..- attr(*, "names")= chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
##  $ incon  : num [1, 1:4] 50 0 2.5 2.5
##   ..- attr(*, "names")= chr [1:4] "SSM0" "SWE0" "SUZ0" "SLZ0"
##  $ prec   : num [1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ airt   : num [1:4383] -2.6 -1 -2.7 -2.8 -2.1 -2 -1.6 -2.1 -3.2 -1.2 ...
##  $ ep     : num [1:4383] 0.0991 0.166 0.0959 0.0923 0.1223 ...
##  $ output : num [1, 1:20, 1:4383] 0.00229 0 50 0 0 ...
##  $ qzones : num [1, 1:4383] 0.00229 0.00767 0.01466 0.02324 0.02883 ...
##  $ q      : num [1, 1:4383] 0.00229 0.00767 0.01466 0.02324 0.02883 ...
##  $ swe    : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ melt   : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ q0     : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ q1     : num [1, 1:4383] 0.0434 0 0 0 0 ...
##  $ q2     : num [1, 1:4383] 0.0298 0.0326 0.0324 0.0322 0.032 ...
##  $ moist  : num [1, 1:4383] 50 50 50 50 50 50 50 50 50 50 ...
##  $ rain   : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ snow   : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ eta    : num [1, 1:4383] 0 0 0 0 0 0 0 0 0 0 ...
##  $ suz    : num [1, 1:4383] 0.457 0 0 0 0 ...
##  $ slz    : num [1, 1:4383] 4.47 4.89 4.86 4.83 4.8 ...

# izrišemo simulirane vrednosti (za prvo leto)
plot(as.numeric(sim1$q)[1:365],type="l", col="red",ylab="Pretoki [mm]")
# dodamo še merjene podatke o pretokih
lines(preob1[1:365,4], col="green")
```



Slika 118: Izračuni modela TUWmodel z naključnimi vrednostmi parametrov.

```
# vidimo, da so razlike relativne velike,
# kar je pričakovano, saj parametrov modela nismo umerili
library(hydroGOF, quietly=TRUE)
# preverimo še določene osnovne statistike ujemanja
gof(as.numeric(sim1$q)[1:365], preob1[1:365,4])
```

```
##           [,1]
## ME        0.09
## MAE       1.47
## MSE       5.51
## RMSE      2.35
## NRMSE % 84.50
## PBIAS %  3.30
## RSR       0.84
## rSD       1.03
## NSE       0.28
## mNSE      0.17
## rNSE      0.23
## d         0.80
## md        0.62
## rd        0.78
## cp       -0.57
## r         0.65
## R2        0.42
## bR2       0.36
## KGE       0.65
## VE        0.45
```

```
# definiramo funkcijo, ki jo bomo uporabili v procesu optimizacije
msespr <- function (param, precip, temp, potevap, runoff, area) {
# zaženemo model
simu <- TUWmodel(param,prec=as.numeric(precip), airt=as.numeric(temp),
  ep=as.numeric(potevap), area=area)$q
```

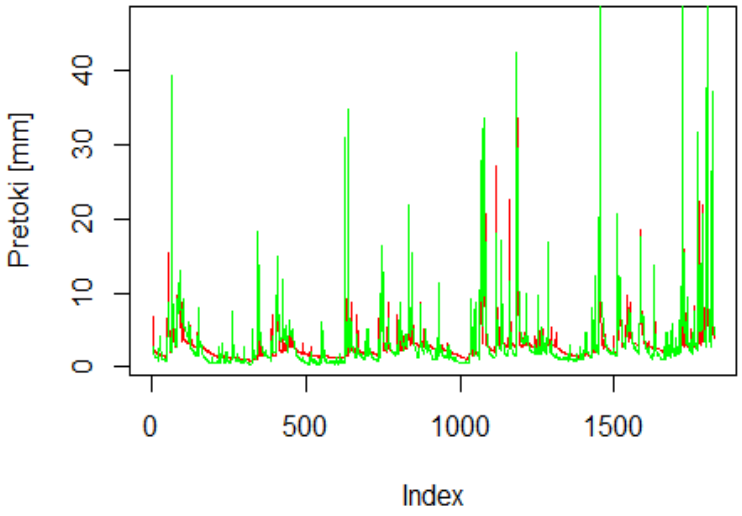
```
# prvo leto podatkov uporabimo za ogrevanje
simu <- simu[-c(1:365)]
obse <- runoff[-c(1:365)]
# izračunamo kriterij MSE, alternativa bi bil
# ročni izračun brez funkcije mean((obse - simu)^2)
mse(simu,obse)
}
```

Sedaj pa se lotimo tudi umerjanja modela z enim višinskim območjem. Definirali bomo nastavitve optimizacijskega algoritma (spremenimo – povečamo lahko tudi argument *itermax*, ki definira maksimalno število optimizacijskih korakov, s čimer sicer podaljšamo čas izračuna). Nato bomo zagnali model z najboljšo kombinacijo parametrov do konca leta 2010 (brez prvega leta, ki smo ga uporabili za ogrevanje) ter izračunali kriterij učinkovitosti modela NSE. Videli bomo, da je model načeloma bolj uspešen kot model GR4J za isto obdobje ter približno enako uspešen kot model GR6J CemaNeige, ki smo ga prav tako uporabili.

```
library(DEoptim, quietly=TRUE)

## Warning: package 'DEoptim' was built under R version 4.1.3

##
## DEoptim package
## Differential Evolution algorithm in R
## Authors: D. Ardia, K. Mullen, B. Peterson and J. Ulrich

area=1 # uporabimo eno višinsko cono
# umerjanje modela
calibrate_period1 <- DEoptim(fn=msespr, lower=c(0.9, 0.0, 1.0, -3.0, -2.0, 0.
0, 0.0, 0.0, 0.0, 2.0, 30.0, 1.0, 0.0, 0.0, 0.0),
upper=c(1.5, 5.0, 3.0, 1.0, 2.0, 1.0, 600.0, 20.0, 2.0, 30.0, 250.0, 100.0, 8
.0, 30.0, 50.0),control=DEoptim.control(NP=NA, itermax=60, reltol=1e-4, stept
ol=50, trace=10, parallelType=0), precip=preob1[1:konec,2], temp=preob1[1:kon
ec,5], potevap=preob1[1:konec,3], runoff=preob1[1:konec,4], area=area)

## Iteration: 10 bestvalit: 5.611297 bestmemit:    0.911294    2.485503    1.
399058   -1.700807   -1.092843    0.443469  522.158192    4.386748    1.50831
7   27.313302  102.445324    8.068820    5.506733    9.553699   36.350001
## Iteration: 20 bestvalit: 5.227388 bestmemit:    1.156470    1.321408    1.
117402   -2.484593    1.726505    0.175899  554.152875    4.267478    1.31950
8    2.543315   63.860740   20.742001    6.636973   13.645317   39.906745
## Iteration: 30 bestvalit: 4.572308 bestmemit:    1.091526    1.321408    1.
117402   -1.969616   -1.374502    0.595091  554.152875    4.267478    1.31950
8    2.543315   63.860740   24.942543    6.636973   13.645317   39.906745
## Iteration: 40 bestvalit: 4.543579 bestmemit:    1.091526    1.321408    1.
117402   -1.969616   -1.374502    0.595091  554.152875    4.267478    1.31950
8    2.543315   63.860740   24.942543    5.907190   13.557868   39.906745
## Iteration: 50 bestvalit: 4.390717 bestmemit:    1.065064    1.321408    1.
117402   -2.337340   -1.516016    0.377852  554.152875    4.267478    1.31950
8    2.543315   63.860740   50.156800    3.925980   13.557868   39.906745
## Iteration: 60 bestvalit: 4.356931 bestmemit:    1.160399    1.579897    1.
```

```
289074   -2.702604   -1.016168    0.517566  468.275596    3.256204    1.10386
4    3.247163   63.093022   36.892444    5.911351   13.731370   37.320947
```

```
sim2 <- TUWmodel(prec=preob1[1:konec,2], airt=preob1[1:konec,5],
  ep=preob1[1:konec,3], area=1, param=calibrate_period1$optim$bestmem)
# izrišemo simulirane vrednosti (do konca leta 2010,
# brez prvega leta, ki smo ga uporabili za ogrevanje)
plot(as.numeric(sim2$q)[366:konec],type="l", col="red",ylab="Pretoki [mm]")
# dodamo še merjene podatke o pretokih
lines(preob1[366:konec,4], col="green")
```



Slika 119: Rezultati za obdobje umerjanja z modelom TUWmodel.

```
# izračunamo še kriterij NSE
NSE(as.numeric(sim2$q)[366:konec],preob1[366:konec,4])
```

```
## [1] 0.8030478
```

Naloga 42: Izvedite umerjanje modela GR5J za obdobje 2005–2011 (z uporabo podatkov za porečje Selške Sore) in validacijo modela z uporabo umerjenih parametrov za leti 2012 in 2013. Prikažite rezultate validacije modela.

Naloga 43: Zaženite model GR4J za obdobje 2006–2007 s podatki s porečja Selške Sore in z naključno generiranimi parametri modela (neumerjenimi), pri čemer morate pred naključnim generiranjem štirih številk uporabiti funkcijo set.seed(15). Razponi parametrov so: x1: 0–2500 mm, X2: -5->5 mm; x3: 0–1000 mm; x4: 0–10 dni. Uporabite enakomerno porazdelitev. Model zaženite in primerjajte rezultate z meritvami.

Naloga 44: Zaženite model TUWmodel še za obdobje validacije (2011–2016) ter rezultate prikažite grafično in izračunajte kriterij NSE.

4.17 Analiza občutljivosti in negotovosti

Analiza občutljivosti je pri hidrološkem modeliranju še posebej pomembna z vidika razumevanja in interpretacije delovanja modela. Takšne analize pomagajo razumeti, kako spremembe vhodnih parametrov vplivajo na rezultate modela. To razumevanje je ključno za razlago obnašanja modela, opredelitev ključnih dejavnikov, povezanih z analizami, in vpogled v simulirane hidrološke procese. Rezultati analiz lahko tako usmerjajo izbiro parametrov, ki jim je treba dati poseben poudarek v procesu umerjanja modela, tako da se določi tiste, ki najbolj vplivajo na rezultate modela. Analiza občutljivosti dodatno pomaga količinsko opredeliti negotovost, povezano z različnimi parametri, kar omogoča boljše razumevanje zanesljivosti modelskih napovedi. To je povezano tudi s pridobitvijo vhodnih podatkov, saj je za parametre, ki imajo velik vpliv na delovanje modela, smiselno pridobiti čim bolj kakovostne podatke, tako z vidika negotovosti kot časovne ločljivosti. Kot primer občutljivostnih analiz bomo prikazali primer, ki je uporabljen na študiji preprostega modela erozije tal[150]. Postopek bo prikazal primer uporabe paketa *sensitivity*, ki vključuje številne funkcije za izvedbo občutljivostnih analiz. Uporabili pa bomo globalno metodo analize občutljivosti, ki jo je razvil Sobol[151]. Obstajajo pa tudi različni indeksi, kot sta standardni regresijski koeficient ali pa normalizirani indeks, ki omogočajo hitro oceno občutljivosti modela[152]. Te metode so bile med drugim uporabljene tudi za analize občutljivosti v povezavi z izračunom erozivnosti padavin[153] ali uporabo modela WATEM/SEDEM[154]. V nadaljevanju bomo definirali funkcijo, s katero lahko izračunamo potencialno sproščanje tal na podlagi Gavrilovićeve enačbe oziroma modela Erosion Potential Model (EPM). Opis vhodnih parametrov je na voljo v članku Bezak in sodelavci (2023)[155].

```
EPM <- function(Temp,Pa,A,rho,X,Y,S){
Tkoef <- sqrt(Temp/10+0.1) # temperaturni koeficient
Zkoef <- X*Y*(rho+sqrt(S)) # Z koeficient
EPM <- Tkoef*Pa*3.14*A*Zkoef^(3/2) # izračun potencialnega sproščanja
return(EPM) # funkcija vrne rezultat
}
EPM(10,1462,18.8,0.3,0.2,1.1,30/100) # primer izračuna

## [1] 7290.307

library(sensitivity, quietly=TRUE)
```

[150] https://doi.org/10.1016/j.catena.2023.107596.

[151] https://doi.org/10.1016/S0378-4754(00)00270-6.

[152] https://doi.org/10.1016/j.envsoft.2013.09.031.

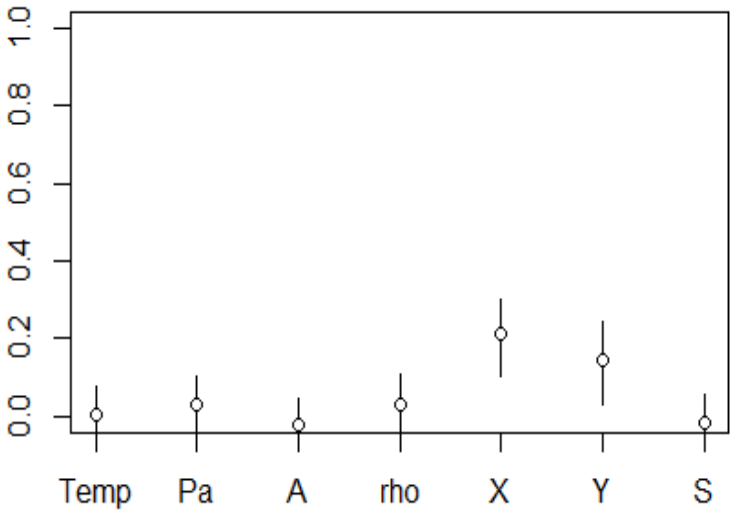
[153] https://doi.org/10.1016/j.envres.2018.08.020.

[154] https://doi.org/10.1007/s12665-015-4534-0.

[155] https://doi.org/10.1016/j.catena.2023.107596.

```
## Warning: package 'sensitivity' was built under R version 4.1.3

## Registered S3 method overwritten by 'sensitivity':
##   method    from
##   print.src dplyr

##
## Attaching package: 'sensitivity'

## The following object is masked from 'package:raster':
##
##     extract

## The following object is masked from 'package:terra':
##
##     extract

## The following object is masked from 'package:dplyr':
##
##     src

rangemin <- c(0,400,5000,0.1,0.05,0.25,1/100) # spodnje meje parametrov
rangemax <- c(35,4000,5000,1,1,2,50/100) # zgornje meje parametrov
# določimo število naključno generiranih parametrov
# v razponu med najmanjšo in največjo vrednostjo
n <- 1000
# generiramo parametre
nab1 <- cbind(runif(n,rangemin[1],rangemax[1]),runif(n,rangemin[2],rangemax[2
]),runif(n,rangemin[3],rangemax[3]),
  runif(n,rangemin[4],rangemax[4]),runif(n,rangemin[5],rangemax[5]),runif(n,r
angemin[6],rangemax[6]),
  runif(n,rangemin[7],rangemax[7]))
# spremenimo imena stolpcev
colnames(nab1) <- c("Temp","Pa","A","rho","X","Y","S")
# naredimo še en nabor parametrov
nab2 <- cbind(runif(n,rangemin[1],rangemax[1]),runif(n,rangemin[2],rangemax[2
]),runif(n,rangemin[3],rangemax[3]),
   runif(n,rangemin[4],rangemax[4]),runif(n,rangemin[5],rangemax[5]),runif(n,
rangemin[6],rangemax[6]),
   runif(n,rangemin[7],rangemax[7]))
# preoblikujemo stolpce
colnames(nab2) <- c("Temp","Pa","A","rho","X","Y","S")
# definiramo še eno preoblikovano funkcijo za model EPM
EPM1 <- function(X){
  Tkoef <- sqrt(X[,1]/10+0.1)
  Zkoef <- X[,5]*X[,6]*(X[,4]+sqrt(X[,7]))
  EPM <- Tkoef*X[,2]*3.14*X[,3]*Zkoef^(3/2)
  return(EPM)
}
# izvedemo analize občutljivosti za model EPM
rez <- sobol(model=EPM1,X1=nab1,X2=nab2,order=1,nboot=1000)
```

```
# izrišemo rezultate
plot(rez)
```



Slika 120: Analiza občutljivosti z uporabo metode Sobol in upoštevanjem modela EPM.

Vidimo, da sta parametra X in Y tista, ki imata največji vpliv na rezultate modela, torej rezultati modela EPM so najbolj občutljivi na parametra X in Y. Pogosto je pomembno, da se v sklopu hidroloških analiz izvede tudi analize negotovosti. V sklopu programskega okolja R je na voljo nekaj paketov, ki so vezani na negotovost, kot je npr. paket *uncertainty*[156] ali pa paket *metRology*[157]. Funkcije, vključene v teh paketih, omogočajo na primer izvedbo simulacij Monte Carlo ali pa analizo negotovosti v primeru merjenih podatkov. Sicer pa so dodatne metode in postopki za izvedbo analize negotovosti opisane tudi v knjigi Uncertainty Quantification using R[158] ali pa v delu Uncertainty Analysis of Experimental Data with R[159]. Za analize negotovosti s poudarkom na hidrološkem modeliranju izpostavljamo še knjigo Rainfall-Runoff Modelling: The Prime[160] in znanstveni prispevek z naslovom Parameter estimation and uncertainty analysis in hydrological modeling[161]. V nadaljevanju je prikazan primer simulacije pretokov, če poznamo razpon vrednosti parametrov in predpostavimo, da enakomerna porazdelitev ustrezno opiše možni razpon vrednosti parametrov.

```
n <- 50 # število simulacij
# generiramo parametre
```

---

[156] https://cran.r-project.org/web/packages/uncertainty/index.html.

[157] https://cran.r-project.org/web/packages/metRology/index.html.

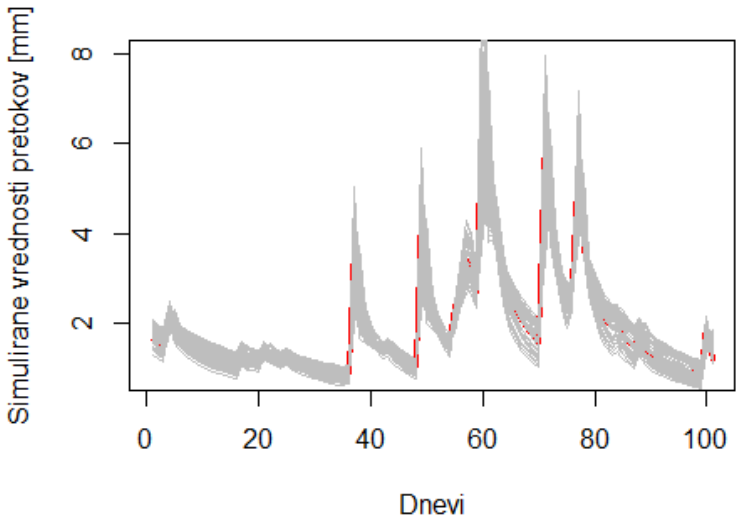
[158] https://doi.org/10.1007/978-3-031-17785-9.

[159] https://doi.org/10.1201/9781315366715.

[160] https://doi.org/10.1002/9781119951001.

[161] https://doi.org/10.1002/wat2.1569.

```
GR4Jnab1 <- cbind(runif(n,ParamGR4J[1]-50,ParamGR4J[1]+70),runif(n,ParamGR4J[
2]-0.9,ParamGR4J[2]+0.7),runif(n,ParamGR4J[3]-70,ParamGR4J[3]+110),round(runi
f(n,ParamGR4J[4],ParamGR4J[4]+0.7),1))
# spremenimo imena stolpcev
colnames(GR4Jnab1) <- c("X1","X2","X3","X4")
# zaženemo model z ocenjenimi vrednostmi parametrov
plot(RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOptions1,
  Param =ParamGR4J)$Qsim[100:200],type="l",col="red",
  ylab="Simulirane vrednosti pretokov [mm]",xlab="Dnevi",lwd=3)
# ter še za možen nabor parametrov
for(k in 1:n){
# naredimo izračune
simulacije <- RunModel_GR4J(InputsModel = InputsModel, RunOptions = RunOption
s1,
  Param = c(GR4Jnab1[k,1],GR4Jnab1[k,2],GR4Jnab1[k,3],GR4Jnab1[k,4]))
# izrišemo rezultate
lines(simulacije$Qsim[100:200],col="grey")
}
```



Slika 121: Simulirane vrednosti pretokov za različne kombinacije parametrov.

Naloga 45: Z modelom TUWien izračunajte vrednosti pretokov (za isto obdobje kot pri nalogi 44) za različne kombinacije parametrov, ki so bile uporabljene med optimizacijo oziroma umerjanjem modela. Rezultate prikažite grafično.

4.18 Podnebne spremembe

Analize vpliva podnebnih sprememb na procese v vodnem krogu se v zadnjih desetletjih pogosto izvajajo z različnimi cilji, osnova teh analiz so pogosto izračuni in napovedi

organizacije IPCC[162]. V sklopu programskega okolja R je na voljo veliko paketov, ki so namenjeni delu s podatki, povezanimi s podnebnimi modeli. Relativno veliko število paketov je na voljo v repozitoriju Github; takšni paketi so na primer *climate4R*[163], *downscaleR*[164] ali pa paket *climdex*[165]. Na voljo pa so tudi paketi v repozitoriju CRAN, kot je *musica*[166], ki omogoča analizo vpliva preračuna rezultatov globalnih modelov na lokalno raven (angl. *downscaling*), s poudarkom na hidroloških študijah[167], ali pa paket *spdownscale*[168]. Omeniti velja še paket *geodata*[169], ki med drugim omogoča analize podatkov CMIP6, ki so na voljo na spletni strani Worldclim[170]. V nadaljevanju bo prikazan primer uporabe paketa *geodata*. Najprej bomo prenesli podatke o padavinah (argument *var*) za izbrani podnebni model (argument *model*) ter scenarij izpusta toplogrednih plinov (*ssp*) za izbrano obdobje (*time*) in prostorsko resolucijo (*res*) z uporabo funkcije *cmip6_world*. Nato bomo določili povprečne mesečne padavine za porečje Sore za obdobje 2061–2080 za izbrani podnebni model. Paket *geodata* omogoča prenos tudi številnih drugih spremenljivk (tudi hitrosti vetra, temperature zraka, sončno sevanje).

```
library(geodata, quietly=TRUE)
# cmip6 <- cmip6_world(model="CNRM-CM6-1", ssp="245", time="2061-2080",
#      var="prec", res=10, path=tempdir()) # prenos podatkov
library(terra, quietly=TRUE)
cmip6 <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/wc
2.1_10m_prec_CNRM-CM6-1_ssp245_2061-2080.tif")
cmip6 # vidimo, da gre za mesečne vsote padavin (12 kart)
```

```
## class       : SpatRaster
## dimensions  : 1080, 2160, 12  (nrow, ncol, nlyr)
## resolution  : 0.1666667, 0.1666667  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (EPSG:4326)
## source      : wc2.1_10m_prec_CNRM-CM6-1_ssp245_2061-2080.tif
## names       : wc2.1~ec_01, wc2.1~ec_02, wc2.1~ec_03, wc2.1~ec_04, wc2.1~ec
_05, wc2.1~ec_06, ...
```

---

[162] https://www.ipcc.ch/.

[163] https://github.com/SantanderMetGroup/climate4R.

[164] https://github.com/SantanderMetGroup/downscaleR.

[165] https://github.com/pacificclimate/climdex.pcic.

[166] https://cran.r-project.org/web/packages/musica/index.html.

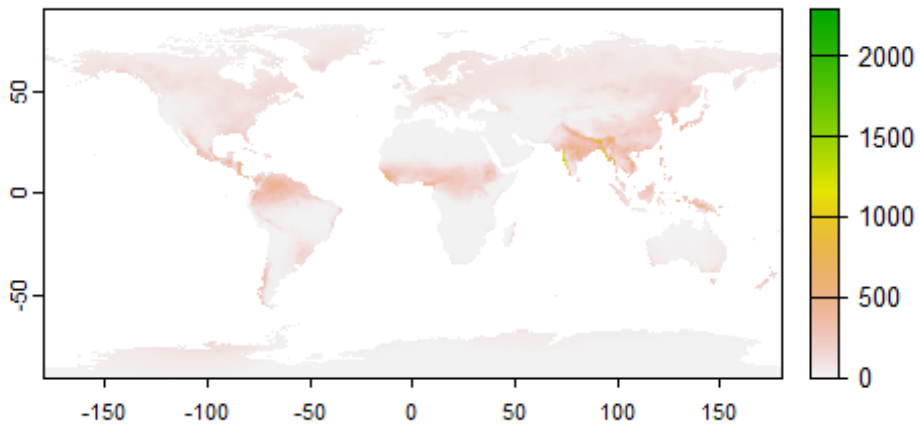
[167] https://doi.org/10.1016/j.envsoft.2017.03.036.

[168] https://cran.r-project.org/web/packages/spdownscale/.

[169] https://cran.r-project.org/web/packages/geodata/index.html.

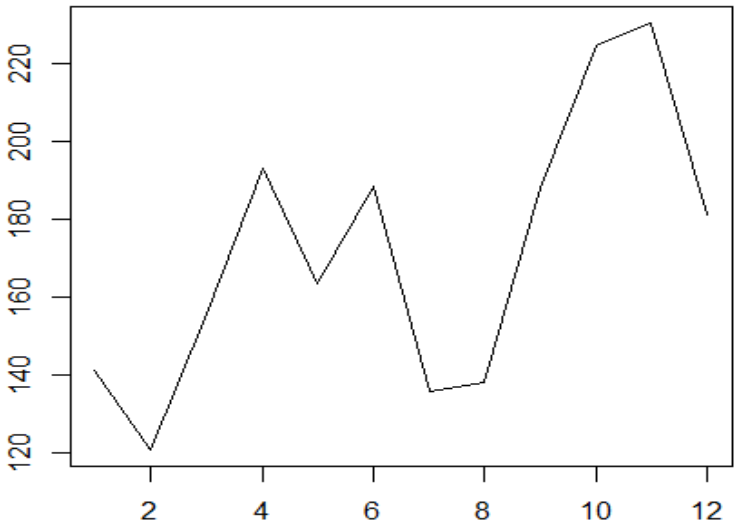
[170] https://www.worldclim.org/.

```
## min values  :         0.0,         0.0,           0,         0.0,
0.0,         0.0, ...
## max values  :      1044.4,       934.8,         688,       654.8,       88
2.1,      1703.8, ...

plot(cmip6[[7]]) # izrišemo npr. mesečne padavine za julij
```



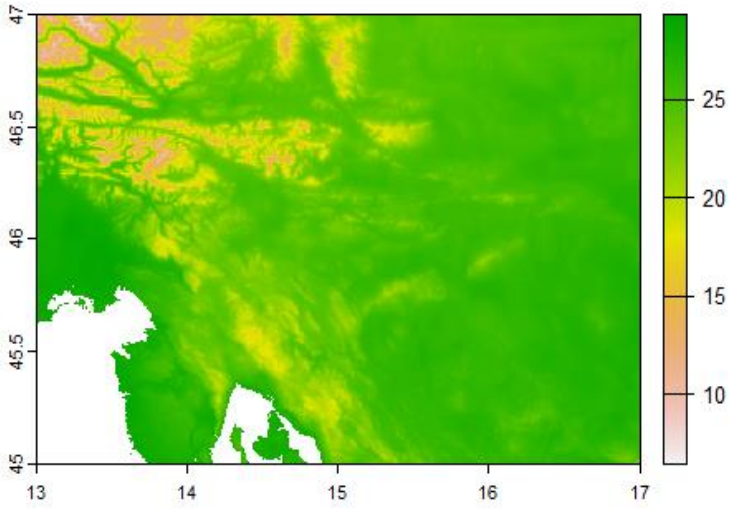
Slika 122: Mesečne padavine za julij za model CNRM-CM6-1 in scenarij RCP 2.45.

```
# deaktiviramo paket sensitivity, ki vsebuje funkcijo extract
detach("package:sensitivity", unload = TRUE)
# določimo povprečne mesečne padavine
extract(cmip6,porecje4)

##   ID wc2.1_2.5m_prec_01 wc2.1_2.5m_prec_02 wc2.1_2.5m_prec_03
## 1  1              141.1              120.9              155.9
##   wc2.1_2.5m_prec_04 wc2.1_2.5m_prec_05 wc2.1_2.5m_prec_06 wc2.1_2.5m_prec
_07
## 1              192.9              163.3              188.5              13
5.7
##   wc2.1_2.5m_prec_08 wc2.1_2.5m_prec_09 wc2.1_2.5m_prec_10 wc2.1_2.5m_prec
_11
## 1              138.1              188.1              224.4              23
0.1
##   wc2.1_2.5m_prec_12
## 1               181

plot(1:12, as.numeric(extract(cmip6,porecje4))[2:13],type="l",
  xlab="Mesec",ylab="Mesečne padavine [mm]")
```



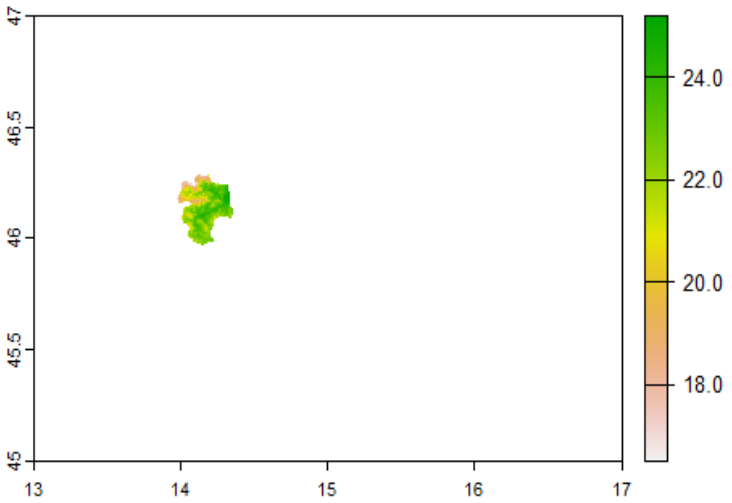
Slika 123: Mesečne padavine za porečje Sore (2061–2080).

```
# prenesemo podatke o maks. temperaturi zraka
#slovenija <- worldclim_country("Slovenia", var="tmax", path=tempdir())
slovenija <- rast("C:/Users/nbezak/OneDrive - Univerza v Ljubljani/Ucbenik/GIS/SVN_wc2.1_30s_tmax.tif")
# maksimalna temperatura zraka za mesec julij
plot(slovenija[[7]])
```



Slika 124: Maksimalna temperatura zraka za Slovenijo.

```
# samo za porečje Sore
plot(mask(slovenija[[7]],porecje4))
```



Slika 125: Maksimalna temperatura za porečje Sore.

Paket *geodata* vsebuje funkcije za prenos tudi drugih podatkov, kot so podatki o tleh (funkcija *soil_world*) ali podatki o digitalnem modelu višin (funkcija *elevation_global*). Podatki, ki smo jih uporabili v prejšnjem primeru, so bili že transformirani na lokalno raven. Poleg tega je bila popravljena sistematična napaka v podatkih (angl. *bias correction*). Programsko okolje R ima tudi številne pakete, ki omogočajo izvedbo teh operacij, če razpolagamo z osnovnimi podatki simulacij CMIP6. Tako so recimo Sezen in sodelavci[171] uporabili metodo quantile mapping za popravke padavin z uporabo paketa *qmap*[172] in pa metodo delta mapping za popravke temperature zraka z uporabo paketa *MBC*[173]. Omenjene metode so bile uporabljene tudi pri analizah podnebnih sprememb v Sloveniji[174]. Podatki projekta Ocena podnebnih sprememb v Sloveniji do konca 21. stoletja so sicer na voljo tudi na spletni strani Odprti podatki Slovenije (npr. za scenarij RCP4.5)[175], kjer se lahko podatke pridobi v formatu .nc, ki se ga da relativno enostavno odpreti in analizirati v programskem okolju R.

V programskem okolju R so na voljo tudi paketi, ki omogočajo dostop do različnih indeksov, kot je indeks North Atlantic Oscillation (NAO), ki je povezan z različnimi procesi v vodnem

---

171 https://doi.org/10.3390/app10041242.

172 https://cran.r-project.org/web/packages/qmap/index.html.

173 https://cran.r-project.org/web/packages/MBC/index.html.

174 https://meteo.arso.gov.si/uploads/probase/www/climate/text/sl/publications/OPS21_Porocilo.pdf.

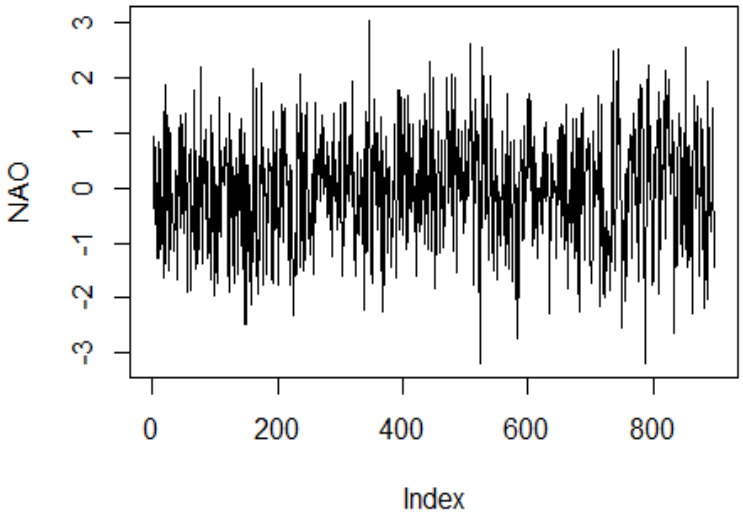
175 https://podatki.gov.si/dataset/arsopodnebne-spremembe-projekcije-visine-padavin-dnevni-podatki-scenarij-rcp4-5-locljivost-0-125.

krogu[176]. Omenimo lahko paket *rsoi*[177], ki omogoča prenos številnih indeksov. Izračunali bomo avtokorelacijo na izbranih podatkih indeksa NAO. Več informacij o funkciji je na voljo v opisu funkcije *acf*.

```
library(rsoi, quietly=TRUE)
nao <- download_nao() # prenesemo podatke o indeksu NAO
# preverimo strukturo podatkov
head(nao)

## # A tibble: 6 x 3
##    Year Month   NAO
##   <int> <ord> <dbl>
## 1  1950 jan.   0.92
## 2  1950 feb.   0.4
## 3  1950 mar.  -0.36
## 4  1950 apr.   0.73
## 5  1950 maj   -0.59
## 6  1950 jun.  -0.06

# vidimo, da gre za mesečne podatke od leta 1950 naprej
tail(nao) # pa vse do konca leta 2023

## # A tibble: 6 x 3
##    Year Month   NAO
##   <int> <ord> <dbl>
## 1  2024 jul.   1.46
## 2  2024 avg.   0.63
## 3  2024 sep.  -1.43
## 4  2024 okt.  NA
## 5  2024 nov.  NA
## 6  2024 dec.  NA

summary(nao$NAO) # osnovne statistike

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.      NA's
## -3.180000 -0.750000  0.040000 -0.005061  0.720000  3.040000         3

plot(nao$NAO,ylab="NAO",type="l") # izrišemo graf
```

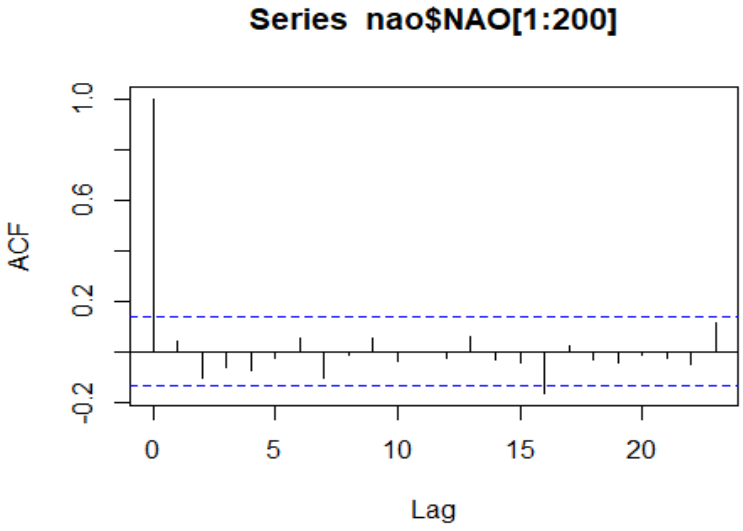
[176] https://www.jstor.org/stable/4315743.

[177] https://cran.r-project.org/web/packages/rsoi/index.html.



Slika 126: Nihanje indeksa NAO.

```
acf(nao$NAO[1:200])
```



Slika 127: Avtokorelacija v podatkih NAO.

Naloga 46: Na podlagi simulacij CMIP6 grafično prikažite mesečno dinamiko v padavinah v Sloveniji in okolici (lon=c(13,17),lat=c(45,47)) za obdobje 2061–2080 z uporabo modela CMCC-ESM2.

Naloga 47: S spletne strani ARSO prenesite podatke o mesečnih padavinah za postajo Murska Sobota, ki ste jih uporabili pri nalogi 37, in analizirajte, ali obstaja povezava med indeksom NAO in padavinami za izbrano postajo (za izbrano obdobje, glede na razpoložljivost podatkov ARSO).

---

5 Zaključek

Programsko okolje R je močno orodje za izvedbo različnih analiz na področju vodarstva oziroma hidrologije. Prilagodljivost programskega jezika R in širok nabor statističnih in grafičnih zmožnosti omogočata nadgrajeno razumevanje hidroloških procesov. Uporaba R lahko izboljša vizualizacijo in interpretacijo podatkov. S številnimi statističnimi zmožnostmi R prispeva k natančnosti in ponovljivosti hidroloških študij. Uporaba programa R na področju vodarstva lahko poenostavi delovne postopke, to pa poveča učinkovitost in prihranek časa. Z avtomatizacijo in razpoložljivostjo številnih paketov, prilagojenih za vodarske študije, je mogoče določene naloge, kot so obdelava podatkov, umerjanje modelov in analize velikih količin podatkov, izvesti hitreje, kar uporabnikom programskega okolja R omogoča, da se osredotočijo na interpretacijo rezultatov. Povezovalna narava programskega okolja R olajšuje izmenjavo znanj in sodelovanje med uporabniki programskega okolja. Čeprav uporaba programskega okolja R na področju vodarstva prinaša številne prednosti, se je treba zavedati izzivov, kot je relativno strma krivulja učenja. Zato upam, da bo ta učbenik v pomoč pri bolj učinkoviti uporabi programskega okolja R na področju vodarstva.