ДРУШТВО МАТЕМАТИЧАРА СРБИЈЕ

АКРЕДИТОВАНИ ПРОГРАМ:

345.
ДРЖАВНИ СЕМИНАР О НАСТАВИ МАТЕМАТИКЕ И РАЧУНАРСТВА ДРУШТВА МАТЕМАТИЧАРА СРБИЈЕ

Компетенција: К1
Приоритети: 3

ТЕМА 23:

ПРИМЕНА *R* ЈЕЗИКА У ОСНОВНОЈ СТАТИСТИЧКОЈ И ГРАФИЧКОЈ АНАЛИЗИ ПОДАТАКА

РЕАЛИЗАТОРИ СЕМИНАРА:

МИЉАН ЈЕРЕМИЋ,
МИЛАН ГОШИЋ

БЕОГРАД,
09. – 10. 02. 2019.

Примена R језика у основној статистичкој и графичкој анализи података

Циљеви предавања:

- Упознати се са инсталацијом и основним окружењем програмског језика R и његовом синтаксом.
- Упознати се са основним могућностима језика R у области статистичке и графичке анализе података.

У оквиру предавања врши се упознавање са синтаксом језика R и његова примена у основној статистичкој и графичкој анализи података, обрађују основе рада у окружењу програмског језика R као и његово коришћење у пракси. Предавање је намењено наставницима који се по први пут сусрећу како с окружењем програмског језика R, тако и са анализом података.

За похађање овог предавања потребно је познавање основа рада с рачунаром и оперативним системом MS Windows, као и познавање основа рада на интернету, а подразумева се и минимално искуство у програмирању.

Наука о подацима

Наука о подацима (енгл. *data science*) присутна је у рачунарству око 50 година, а у новије време изражене су потребе да се огромне количине података (енгл. *big data*) моделирају у пословне сврхе. Потреба за анализом информација, у циљу праћења пословних резултата предузећа и пословног одлучивања, остварује се интеграцијом података и приказом јединствених показатеља пословања. Основни алати пословне интелигенције (енгл. *Business Intelligence*) дају збирне извештаје, захтевајући знања и вештине из рачунарства, математике и статистике, али и висок степен креативности и вештина комуникације. С тога постоје основани захтеви за унапређењем наставе применом R језика.

R језик

R је програмски језик и окружење за статистичка израчунавања и визуелизацију. Представља слободан програмски језик (*GNU*) што значи да се може слободно користити и дистрибуирати, као и да је отвореног кода (енгл. *open-source*). Имплементиран је из језика S којег је развио John Chambers са колегама у Bell Laboratories, као и Robert Gentleman са факултета у Окланду.

Представљен је као производ активног покрета међу статистичарима који има за циљ стварање моћног, преносивог, отвореног програмског окружења примењивог при решавању већине сложених проблема, али и за проверу рутинских анализа. Пружа нам широк избор статистичких метода за линеарно и нелинеарно моделирање, класичне статистичке тестове, анализе временских серија или кластеризацију. Овај језик је лако проширив са великим избором графичких техника. Статистичари су развили стотине специјализованих статистичких процедура за широк спектар примена путем придодатих пакета (енгл. *contributed packages*) који су слободно доступни и интегрисани директно у R језик.

До скора је постојало мишљење да R језик користи углавном академска заједница, али се то данас променило. Тако на пример, Федерална агенција за лекове (*Federal Drug Administration - FDA*) идентификовала је делове R језика погодним за тумачење података из клиничких истраживања. Велики је број компанија (Google, Oracle, Microsoft), које сада прелазе на R језик за анализу и презентацију својих података.

Предности и недостаци R језика

Неке од предности R језика су:

- Производ је међународне сарадње врхунских статистичара и дизајнера програмских језика.
- R је доступан као слободан софтвер под условима *Free Software Foundation GNU.*
- Доступан је путем интернета.
- Ради на разним платформама: UNIX (укључујући FreeBSD и Linux), *Windows* и Mac OS.
- Омогућава статистичке анализе и визуелизацију података.
- Омогућава рад са великим и комплексним објектима.

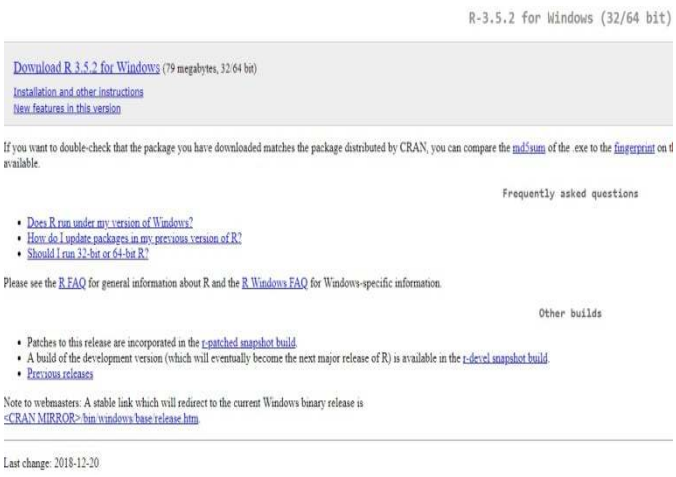
- Нуди могућност размене података са другим програмима као што су датотеке формата MS *Excel*, текста, SAS, датотеке са дефинисаним раздвајањима (CSV). Омогућава приступ базама података као што су Oracle и MS *Access* путем ODBC приступа базама коришћењем SQL упита.

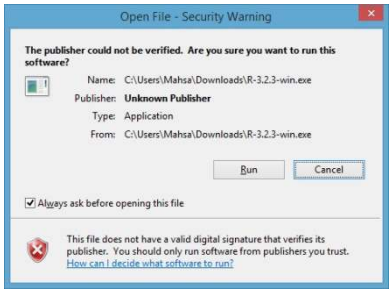
Неки од недостатака R језика су:

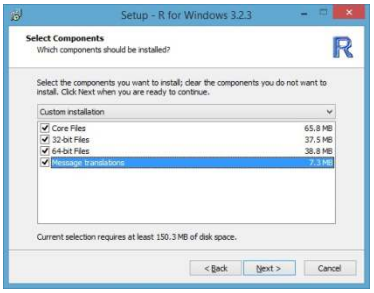
- Постоји слаба подршка анализама коришћењем графичког-корисничког интерфејса (*Graphical User Interface* – GUI).
- Захтева писање наредби како би се вршиле анализе и направила визуелизација резултата.

Инсталација R језика

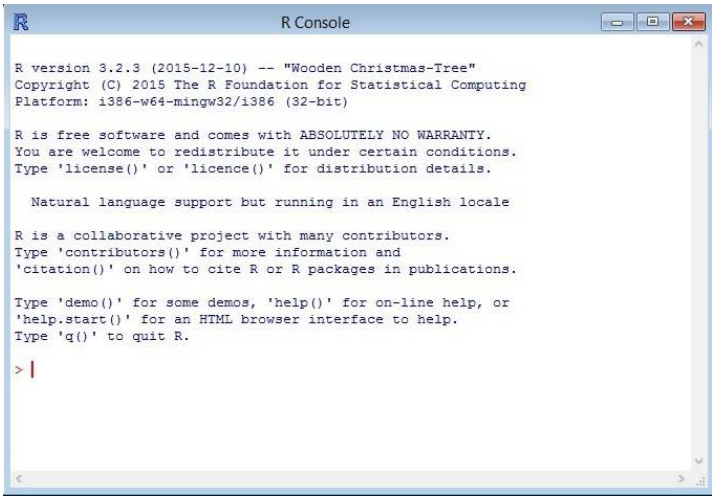
R је погодан за интерпретацију статистичких израчунавања и омогућава графички приказ података што има велики утицај на наше разумевање променљива и њихових односа.
Као први корак треба у интернет читач унети адресу https://cran.r-project.org/bin/windows/base/ и инсталирати одговарајућу верзију која одговара вашем оперативном систему.
Ако поседујете оперативни систем Windows, треба изабрати верзију R 3.5.2 за Windows.







Након инсталирања треба покренути R за унос наредби и креирање статистичких интерпретација и графика.



У овом материјалу су наредбе писане пропорционалним словима (на пример, наредба `install.packages()`).

Синтакса наредби писана је плавим пропорционалним словима са сивим коментарима за делове које програм не изводи:

```
> library(help = "base") #pomoc za paket base
```

Почетак рада

Постављање радног директоријума

Пре почетка рада потребно је дефинисати радни директоријум. Информација о тренутном радном директоријуму добија се на следећи начин:

```
> getwd()
```

Постављање радног директоријума се одређује коришћењем наредбе:

```
> setwd("putanja/moj_direktorijum")
```

Информације о функцији коју желимо користити добијају се уносом упитника испред жељене наредбе. На пример, за функцију `lm()`:

```
> ?lm
```

Основне врсте података

Језик R препознаје шест основних атомских (енгл. *atomic*) врста података:

- знак / слово / текст (енгл. *character*, *string*),
- реалан број (енгл. *numeric*),
- целобројни број (енгл. *integer*),
- комплексни број (енгл. *complex*),
- логички (енгл. *logical*),
- сирови (енгл. *raw*).

Дефинисани су и специјални бројеви бесконачности **Inf** (енгл. *infinity*) који се могу добити дељењем нулом, на пример 1/0, док је 1/Inf = 0.

```
> 1/0
> 1/Inf
```

Генерално дефинисане, а недостајуће вредности у R окружењу означене су ознаком **NA** (енгл. *not available*), док се недефинисане вредности, као што је случај са 0/0 означавају са **NaN** и представљају недефинисану вредност (енгл. *not a number*).
Пример 1:

```
> a < -c(10, 25)
> a[3]
> is.na(c(0/0, NA))
> is.nan(c(0/0, NA))
```

У овом контексту за испитивање својстава бројева користе се наредбе `is.finite()` и `is.infinite()`.

Операције у језику R

R знакови за операторе	Оператор
+, -, *, /, %, ^	аритметички
>, >=, <, <=, ==, !=	релациони
!, &, |, &&, ||, xor(x, y)	логички
~	модел формуле
<-, ->	придруживање
$	индексирање листе
:	креирање секвенце

Функције у језику R

Функција	Значење
abs(x)	апсолутна вредност за x
log(x)	логаритам од x за основу e (ln)
exp(x)	експонент броја x
log(x, n)	логаритам од x за основу n
log10(x)	логаритам од x за основу 10
sqrt(x)	квадратни корен од x
factorial(x)	факторијел број x, x!
choose(n,x)	биномни коефицијент n!/(x! (n-x)!)
gamma(x)	Γ(x) , за реално x (x-1)!, за целобројну вредност x
floor(x)	највећи цео број мањи од x
ceiling(x)	најмањи цео број већи од x
trunc(x)	цео број најближи вредности x између x и 0; trunc(1.5) = 1, trunc(-1.5) = -1
round(x, digits=0)	заокружена вредност x на цео број
signif(x, digits=6)	x на 6 децимала
runif(n)	n случајних бројева између 0 и 1 (униформна дистрибуција)
cos(x), sin(x), tan(x)	тригонометријске функције у радијанима
acos(x), asin(x), atan(x)	инверзне тригонометријске функције
acosh(x), asinh(x), atanh(x)	инверзне хиперболичке тригонометријске функције

Пример 2:

```
> 5 + 6 + 3 + 6 + 4
[1] 24

> 2 + 7; 5 * 6; 3 - 8
[1] 9
[1] 30
[1] -5
> log(10)
[1] 2.302585

> log(42/7.3)
[1] 1.749795
```

```
> exp(1)
[1] 2.718282

> log10(10)
[1] 1


> pi
[1] 3.141593
> sin(pi/2)
[1] 1
> cos(pi/2)
[1] 6.123032e-17
```

Основне структуре података

R је објектно оријентисани језик. Објектно оријентисани језици се темеље на концепту да објекат поседује атрибуте који га описују, као и њему придружене процедуре које се још називају методама. Тако и у основи R окружења постоје класе и методе. Класа је дефиниција објекта. Окружење класа говори нам шта поједини елементи у језику R у ствари представљају. Типично за класе је да унутар себе имају дефинисане исечке (енгл. *slot*) који се користе за чување специфичних информација за ту класу објеката. Методе су функције које делују једино на одређеној класи објеката. Генеричке функције у себи носе неку генеричку (базичну) функционалност - концепт, тако функција `print()` исписује објекте, функција `plot()` их црта, `predict()` ради пројекцију на новом скупу података, итд. Свака од тих функција понаша се нешто другачије у односу на класу објеката на који је примењена.

Табела у наставку даје преглед структура података у R окружењу на темељу њихове димензионалности и хомогености података (сви садржаји морају бити исте врсте) или хетерогени (садржај може бити различитих врста):

N- димензија	Хомогенни	Хетерогени
1	Вектори (*Atomic Vector*)	Листе (*List*)
2	Матрице (*Matrix*)	Скупови података (*Data Frame*)
N	Низови / поља (*Array*)	

Може се приметити једна особина окружења R, а то је да нема скалар, односно врсту података без димензије. Сви скалари (алфанумерички знакови) су у окружењу R дефинисани као вектори дужине 1.

Готово сви други објекти у окружењу R су изграђени на овим основама. Алати унутар објектно оријентисаног језика се такође граде на овим основама.

Ако желите да боље разумете структуру неког објекта у окружењу R, најважнија је функција `str()`, што је скраћеница енглеске речи *structure* која нам на конзоли даје опис објекта на једноставно читљив начин.

Вектори

Основна врста података у окружењу R је вектор. Вектори долазе у два облика: атомски (енгл. *atomic*) и листе (енгл. *list*). Ове две врсте разликују се по свом садржају. Док подаци унутар атомског вектора нужно морају бити исте врсте, садржај листе то не мора бити. Вектори у окружењу R имају три основна обележја:

1. Ко су? Које су врсте?

```
> typeof()
```

2. Дужина вектора

```
> length()
```

3. атрибути – додатни метаподаци о вектору

```
> attributes()
```

Операције са векторима у језику R

Функција	Значење функције
max(x)	Максимална вредност вектора x
min(x)	Минимална вредност вектора x
sum(x)	Сума вредности вектора x
mean(x)	Аритметичка средина вредности вектора x
median(x)	Вредност функције медијане вектора x
range(x)	Вектор од минималног x и максималног x
var(x)	Варијанса вектора x
cor(x,y)	Корелација између вектора x и y
sort(x)	Сортиране вредности вектора x
rank(x)	Вектор рангова вредности вектора x
order(x)	Целобројни вектор који садржи сортирани вектор x у растући редослед
quantile(x)	Вектор који садржи минимум, нижи квартил, медијан, горњи квартил и максимум вектора x
cumsum(x)	Вектор који садржи суму свих елемената до тог тренутка
cumprod(x)	Вектор који садржи производ свих елемената до тог тренутка
cummax(x)	Вектор неопадајућих бројева који су кумулативни максимуми вредности у x до те тачке
cummin(x)	Вектор непостојећих бројева који су кумулативни минимуми вредности у x до те тачке
pmax(x, y, z)	Вектор, дужине једнаке најдужем од x и y и z, који садржи максимум од x и y и z за и-ти положај у сваком од њих

Вектори се најчешће генеришу функцијом `c()` што представља прво слово енглеске речи *combine*. Неко наводи „c“ као прво слово речи *concatenate*, што значи повезивање у облику ланца.

Пример 3:

```
A < -1:20
B < -c(3, 5, 7)

x < - c(4, 3, 8, 7, 2, 7, 6)
Четврти елемент у вектору x   x[4]
[1] 7
```

```
y < -4
z < -y[-1]

length(z)
[1] 0
```

Матричне операције у језику R

Пример 4:

```
> X <- matrix(c(1, 2, 3, 4, 5, 6, 7,
8, 9), nrow = 3)
X
[, 1] [, 2] [, 3]
[1,] 1 2 3
[2,] 4 5 6
[3,] 7 8 9

> class(X)
[1] "matrix"

> attributes(X)
$dim
[1] 3 3
```

```
> array < -1:25
> is.matrix(array)
[1] FALSE
> dim(array)
NULL
> dim(array) < -c(5, 5)
[1] 5 5
> is.matrix(array)
[1] TRUE

> array
[, 1] [, 2] [, 3] [, 4] [, 5]
[1,] 1 6 11 16 21
[2,] 2 7 12 17 22
[3,] 3 8 13 18 23
[4,] 4 9 14 19 24
[5,] 5 10 15 20 25

> is.table(array)
[1] FALSE
```

Пример 5:

Израда текстуалног вектора и прикупљање у објекат који се зове *tekstualni_vektor*.

```
> tekstualni_vektor <- c("tekst1", "tekst2", "tekst3")
```

Израда нумеричког вектора и прикупљање у објекат који се зове *numericki_vektor*.

```
> numericki_vektor <- c(1, 3.5, 6, 4.4)
> numericki_vektor_2 <- c(1, 5, 28, 13)
> celobrojni_vektor <- c(2L, 4L, 6L) #eksplicitna izrada celobrojnog vektora
> celobrojni_vektor_2 <- 1:10 #sekvenca celobrojnih vrednosti
```

Израда логичког вектора и прикупљање у објекат под називом *logicki_vektor*:

```
> logicki_vektor <- c(TRUE, FALSE, T, F)
```

Вектор се може генерисати и функцијом `vector()`.

```
> logicki_vektor_2 <- vector("logical", length = 10)
```

Одређивање типа вектора ради се упитима `typeof()`, или специфичним тестовима као што су: `is.character()`, `is.double()`, `is.integer()`, `is.logical()`, или посебним као: `is.atomic()`, `is.na()`.

Индексирање и селекција подскупа вектора

Селекција елемената вектора (енгл. *subset*) изводи се операндом []. Унутар заграда индексом редног броја елемента у вектору добија се жељена вредност.

Селекција другог елемента објекта *vektor*_1:

```
> vektor_1[2]
```

Селекција првог и трећег елемента вектора *vektor*_3:

```
> vektor_3[c(1, 3)]
```

Листа

Листа (енгл. *list*) за разлику од атомских вектора, може садржати и друге врсте вектора, укључујући и саме листе. Због овог својства још их називамо и рекурзивним векторима. Могу се генерисати помоћу функције `list()`, а не више уз помоћ функције `c()`.

```
> lista_1 <- list(10, "z", TRUE, matrix(1:15, ncol = 5, nrow = 3))
> lista_1

[[1]]
[1] 10

[[2]]
[1] "z"

[[3]]
[1] TRUE

[[4]]
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15
```

Селекција другог реда, треће колоне, на четвртом елементу листе би се радила на овај начин:

```
> lista_1[[4]][2, 3]
> lista_2 <- list("Pera", 1, matrix(runif(10)),5.3)
> names(lista_2) <- c("ime", "broj", "moja_matrica", "starost")
```

Тражење другог елемента у листи:

```
> lista_1[[2]]
```

Тражење елемента одређеног назива у листи:

```
> lista_2[["grad"]]
```

Вектор можемо именовати на три начина:

- приликом израде вектора

```
> x <- c(var1 = 1, var2 = 2, var3 = 3)
```

- изменом постојећег вектора

```
> x <- 1:3;
> names(x) <- c("var1", "var2", "var3")
```

- креирањем произвољног вектора

```
> x <- setNames(1:3, c("var1", "var2", "var3"))
```

Матрице и поља

Поље (енгл. *array*) је релативно ретко коришћена структура података. Може бити вишедимензионално, а посебан случај дводимензионалног поља је матрица (енгл. *matrix*) и оне чине основу за велики део статистичке анализе. Садрже атрибут димензија који се тражи функцијом `dim()`.

Матрица се може генерисати са аргументима броја редова и колона, а притом треба обратити пажњу на начин попуњавања матрице вектором. Давањем димензија, матрица се може направити приликом генерисања матрице:

```
> matrica_1 <- matrix(1:20, ncol = 4, nrow = 5)
```

Функција `seq()` користи се за генерисање секвенци

```
> ?seq #upoznavanje sa funkcijom
> vektor_d <- seq(1, 100, by = 5)
> vektor_d

[1]   1     6 11 16 21 26 31 36 41 46 51 56 61 66 71 76 81 86 91 96

> vektor_d[c(T, F)]
```

Пример 6:

Најпре се присетите како изгледа matrica_1. Уколико је нема више у радном простору, учитајте је као снимљени објекат који се налази у радном директорију назива **matrica_1.RData**. Овај објекат снимљен је у формату R језика употребом функције `dput()`:

```
> dput(matrica_1, "matrica_1.RData")
> rm(matrica_1) #brisanje objekta iz radnog prostora
```

Исти ћемо учитати у радни простор функцијом `dget()`:

```
> matrica_1 <- dget("matrica_1.RData") #vracanje u radni prostor
```

Основно својство вектора `length()` и `names()` сада добијају вишедимензионалну надградњу код матрица и поља на начин:

- `length()` надограђује се функцијама `nrow()` и `ncol()` за матрице и поља, као и `dim()` за поља.
- Истовремено, функција `names()` се надограђује функцијама `rownames()` и `colnames()` за матрице и поља, као и `dimnames()` за поља.

Индексирање и селекција подскупа матрице

Селекција елемената матрице (енгл. *subset*) ради се оператором []. Унутар заграда, индексом реда и колоне жељеног елемента у матрици добија се његова вредност.

Селекција елемената у другом реду и трећој колони матрице:

```
> matrica_1[2,3]
```

или путем атрибута, други елемент у променљивој *var*_3:

```
> matrica_1[2, "var_3"]
```

или целости путем атрибута, име редова у променљивој *var*_3

```
> matrica_1["red_2", "var_3"]
```

Селекција над векторима, матрицама, пољима и скуповима података може бити у услову, односно из објекта ће бити препознати само они елементи који одговарају задатом услову:

```
> selekt <- vektor_d > 10     #daje logicki vektor oznacavajuci sve elemente
koji zadovoljavaju uslov
> vektor_d[selekt]
```

или коришћењем функције `which()` на начин:

```
> select_2 <- which(vektor_d > 10, arr.in = TRUE)
> vektor_d[select_2]
```

Скупови података

Скупови података (енгл. *data frames*) су најуобичајенији начин чувања података у окружењу R. Прецизније речено, скуп података је испис вектора исте дужине. Према овој дефиницији они имају две димензије и деле својства и матрица и исписа. То значи да скуп података има `names()`, `colnames()` и `rownames()` с обзиром да су у овом случају `names()`и `colnames()`исте ствари.

Пример 7:

```
> skup.podataka_1 <- data.frame(x = 1:4, y = c("A", "B", "C", "D"))
> str(skup.podataka_1)

'data.frame':   4 obs. of  2 variables:
 $ x: int  1 2 3 4
 $ y: Factor w/ 4 levels "A","B","C","D": 1 2 3 4

> skup.podataka_2 <- data.frame(x = 1:4, y=c("A", "B", "C", "D"),
stringsAsFactors = FALSE)
> str(skup.podataka_2)

'data.frame':   4 obs. of  2 variables:
 $ x: int  1 2 3 4
 $ y: chr  "A" "B" "C" "D"

#izrada skupa podataka i odredjivanje na koji se nacin ponasa prema znakovima:
smatramo ih character ili factor.
> df <- data.frame(brojevi = 10:19, slova = letters[1:10], stringsAsFactors =
FALSE)
> str(df)

'data.frame':   10 obs. of  2 variables:
 $ brojevi: int  10 11 12 13 14 15 16 17 18 19
 $ slova  : chr  "a" "b" "c" "d" ...

> df2 <- data.frame(brojevi = 1:10, slova = letters[1:10], stringsAsFactors =
TRUE)
> str(df2)

'data.frame':   10 obs. of  2 variables:
```

```
 $ brojevi: int  1 2 3 4 5 6 7 8 9 10
 $ slova  : Factor w/ 10 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9 10

#prebacujemo numericku iz skupa podataka df tipa character u tip factor.
> df$brojevi_f <- as.factor(df$brojevi)

#prebacujemo     tek napravljanu faktorsku promenljivu ponovno u promenljivu
tipa numeric.
> df$brojevi <- as.numeric(df$brojevi_f)
```

Исправно је пребацивање фактора у нумеричку промењиву:

```
> as.numeric(levels(x))[x]
> as.numeric(as.character(x))
```

Учитавање постојећих података у R

У окружењу R податке можемо да учитамо на више начина. Основни типови учитавања:

- табелираних података (`read.table()`унутар којих је могуће дефинисати велики број параметара,
- текста `readLines()`,
- кода окружења R `source()`,
- објеката окружења R `dget()`,
- бинарних објеката `unserialize()`,
- радног простора окружења R (енгл. *workspace*) `load()`.

Посебно су развијени пакета и функције за учитавање SAS-датотека (пакет *sas7bdat*), као и просторних података (пакети *rgdal*, *shapefiles*, *maptools, ...*).

Постоји низ развијених специфичних пакета за рад са датотекама програма *Excel* (пакети *xlsx* и *openxlsx*). Велики број формата записа могуће је учитати уз помоћ пакета *foreign*. SPSS, STATA и SAS датотека се могу учитати уз помоћ пакета *Hmisc*.

Функције у R језику

R је тзв. функционални програм који се фокусира на израду и управљање функцијама. Основно је правило да се унутар окружења R са функцијама може направити све што се може направити с векторима - могу се доделити некој промењивој, спремити за испис, бити аргументи неких других функција. Чак је могуће израдити безимену функцију, као и функцију унутар функције. Функције чине најмоћнији део окружења R и бришу границе између програмера, окружења и корисника.

Корисник може креирати своју функцију употребом наредбе `function()`:

```
> f <- function(<arguments>) {# funkcija nesto radi}
> args(f) #trazenje informacije o definisanim argumentima funkcije
```

У наставку су примери функција које рачунају елементарне статистике у окружењу R.

Функције дескриптивне статистике:

```
> min(x)
> max(x)
> sum(x)
> range(x)
> cumsum(x)
> cumprod(x)
> diff(x)
```

```
> sd(x)
> sd(X)
> var(X)
> cor(X)
> quantile(x, 0.75)
> quantile(x)
> rank(x)
> sort(x)
> order(x)
```

```
> summary(x)
> mean(x)
> median(x)
```

Основно о петљама у R језику

Унутар окружења R постоји неколико начина којим се дефинишу петље. Осим стандардних *for* и *while* петљи, окружење R има развијене и нестандардне петље које су израђене како би радиле понављајуће ствари у специфичним случајевима. То је тзв. *apply* петља, а осим функције садржи и `lapply()`, `sapply()` и `tapply()`. Овде се даје кратак осврт на споменуте петље.

Функција *apply()*

Ово је функција која спроводи другу задату функцију са задатим маргинама (ред или колона) матрице или низа(поља).

```
apply(X, MARGIN, FUN, ...)
```

Пример 8:

```
> apply(moji_podaci, 2, mean) #sredine po kolonama
> apply(moji_podaci, 1, mean) #sredine po redovima
```

Функција *lapply()*

Ово је функција која примењује задату функцију на сваки елемент исписа (листе). Резултат функције је испис.

```
> numericka_lista <- list(ime1 = 1:5, ime2 = rnorm(10))
> lapply(numericka_lista, max)
```

Функција sapply()

Ово је функција слична функцији `lapply()` с том разликом да резултат покушава структуирати у једноставнији облик ако је то могуће.

```
> sapply(numericka_lista, max)
```

Функција tapply()

Ово је функција која ради одређени процес на селекцији вектора са пољима промењљивих дужина. Груписање унутар петље `tapply` дефинише се факторском променљивом.

```
> set.seed(1) #reproduktivan primer
> podaci <-data.frame(osoba = 1:100, pritisak = rnorm(100, mean = 120, sd =
80), tretman = gl(2, 50, labels = c("Tretman", "Kontrola")))
> summary(podaci)


    osoba          pritisak          tretman
 Min.   :  1.00   Min.   :-57.18   Tretman :50
 1st Qu.: 25.75   1st Qu.: 80.46   Kontrola:50
 Median : 50.50   Median :129.11
 Mean   : 50.50   Mean   :128.71
 3rd Qu.: 75.25   3rd Qu.:175.32
 Max.   :100.00   Max.   :312.13

> names(podaci) #imena promenljivih u skupu podataka
```

```
[1] "osoba"    "pritisak" "tretman"

> tapply(podaci$pritisak, podaci$tretman, mean)

 Tretman Kontrola
128.0359 129.3861
```

Расподеле у R језику

Окружење R поседује богату фамилију функција дистрибуције вероватноће као што су, на пример, нормална, бета, биномна, гама, хипергеометријска, итд. Функција `?distributions` из пакета *stats* даће попис свих уграђених дистрибуција у окружењу R. За сваку од наведених функција расподеле вероватноће постоје четири функције које су:

Функција d (x, ...) рачуна густину расподеле на x-оси

```
> dnorm(0) #gustina na 0; standardna normalna distribucija

[1] 0.3989423
```

Функција p (q, ...) рачуна кумулативну вероватноћу до q P(x # q)

```
> pnorm(1.96) #kumulativna distribucija P(Z<1.96)

[1] 0.9750021
```

Функција q (p, ...) за дато p

```
> qnorm(0.89) #98-i kvantil

[1] 1.226528
```

Функција r (n, ...) генерише случајну променљиву задате величине

```
> set.seed(1)
> rnorm(10)

[1] -0.6264538  0.1836433 -0.8356286  1.5952808  0.3295078 -0.8204684
[7]  0.4874291  0.7383247  0.5757814 -0.3053884
```

Графичко окружење у R језику

Унутар окружења R постоји неколико готово потпуно одвојених окружења графичке репрезентације објеката. Овде ће бити више речи о основама рада у овим окружењима.

Графика у окружењу R језика

Графика основног окружења налази се аутоматски након инсталације софтвера. Функције за рад са графиком налазе се у основном пакету *graphics*, а укључују функције као што су `plot()` за цртање графика, `hist()` за израду хистограма, `boxplot()` за *Box-Whiskers plot* итд. Више детаља о пакету *graphics* могу се добити наредбом

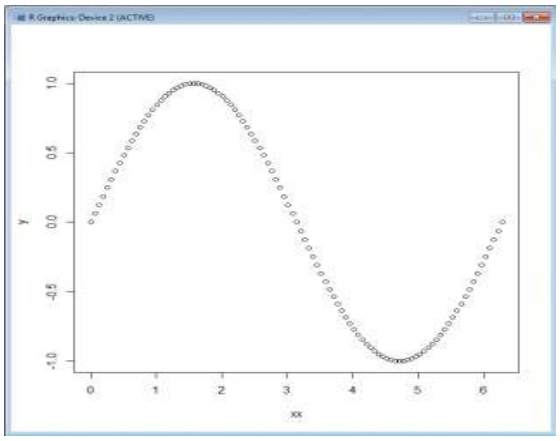
```
> help(package = graphics)
```

Пример 9:

```
> x = 0:100
> x
[1] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17
```

```
[19] 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35
[37] 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53
[55] 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71
[73] 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
[91] 90 91 92 93 94 95 96 97 98 99 100
```

```
> xx = x * 2 * pi/100
> y = sin(xx)
> plot(xx, y)
```



Ако позовемо функцију `plot()` у пакету base и немамо отворен графички прозор (енгл. *plotting device*), она га аутоматски отвара. Ако желимо да имамо већи број отворених графичких прозора можемо сами отворити нови графички прозор функцијом `dev.new()`. Ово није случај ако се ради са RStudiom. Ако се креира нека графика ради презентације у неком документу боље је отворити датотечки уређај (енгл. *file device*) него екран (енгл. *screen device*).

Боје у R језику

За визуелизацију различитих типова података потребне су и различите палете. За тражење већ раније припремљених палета за различите потребе веома је корисно упознати се с функцијама пакета *RColorBrewer*.

Пример 10:

```
> data(volcano)
> library(RColorBrewer)
> boje <- brewer.pal(5,"Blues") #biramo pet boja iz plave palete
> boje
```

```
[1] "#EFF3FF" "#BDD7E7" "#6BAED6" "#3182BD" "#08519C"
```

Ако желимо, на пример 20 боја, али на основу горе изабраних 5 писаћемо

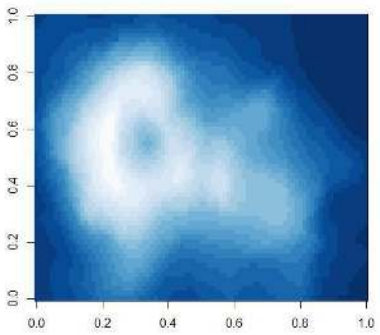
```
> boje20 <- colorRampPalette(boje)(20)
```

једнако као и

```
> boje20 <- colorRampPalette(brewer.pal(5,"Blues"))(20)
> str(volcano)
```

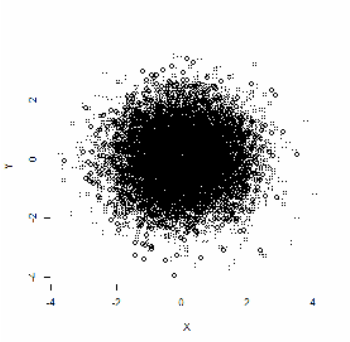
```
num [1:87, 1:61] 100 101 102 103 104 105 105 106 107 108 ...
```

```
> image(volcano, col = boje20)
```



Споменућемо још једну функцију која је врло корисна при визуелизацији великог броја тачака:

```
> X <- rnorm(10000)
> Y <- rnorm(10000)
> Z <- cbind(X, Y)
> plot(X, Y)
```



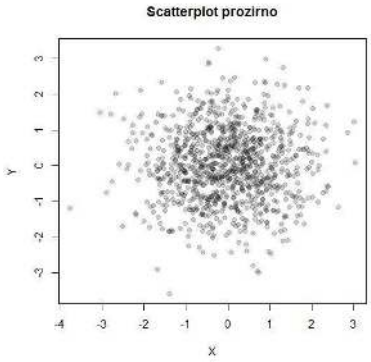
```
> smoothScatter(Z)
> graphics::plot(Z, col = densCols(Z), pch = 20, main = "Gustina tacaka")
> savePlot(filename = "Proba_smooth.jpg", type = "jpg")
> dev.copy(png, 'myplot.png')
```

Функција `savePlot()` креира графике тренутно приказане у активном графичком прозору, али не ради из *RStudio*-ја. Ово није оптималан начин да се прикаже график, требало би да се уради то овако:

```
> jpeg('rsmoothplot.jpg') #otvaramo zeljeni medij - jpg
> smoothScatter(Z) #crtamo u medij
> dev.off() #zatvaramo medij
```

Тачно одређена боја може се направити функцијом `rgb()` дефинисањем тачних спецификација црвене, зелене и плаве боје уз могућност дефинисања прозирности боје као последњег аргумента функције.

```
> plot(X, Y, pch = 19) #gotovo nista se ne vidi
> plot(X, Y, col = rgb(0, 0, 0, 0.2), pch = 19, main = "Scatterplot prozirno")
#prozirno
> savePlot(filename = "Proba_smooth.jpg", type = "jpg")
```



Линеарни модел података

Приликом истраживања међусобних веза двеју променљивих примењују се методе просте (линеарне и нелинеарне) регресионе и корелационе анализе. У случају више променљивих реч је о методама вишеструке (линеарне и нелинеарне) регресионе и корелационе анализе. Прост регресиони модел је математички модел који има само две променљиве: зависну и независну. Прост линеарни регресиони модел је регресиони модел којим се описује линеарна веза између зависне и независне рроменљиве.

Пример 11:

Претпоставимо да имамо експеримент да видимо колико људска бића одрастају (по висини) у складу са својим годинама. Бира се неколико особа у различитим годинама без обзира на њихов пол, расу и сл. Зато што је важно да се зна шта је понашање особа које расту само на основу њиховог узраста. Зато ћемо имати модел који указује на образац растуће људске висине. Зато што стварни свет није увек детерминистички и постоје изузеци. У овом примеру постоје и други фактори који могу да утичу на резултат. За случајеве пола, раса су фактори које не желимо да их доведемо на наше рачунање. Овде је "Height - висина" зависна променљива јер се мери на основу "доба", а " Age - старост" се назива независна променљива или предиктор или фиксни ефекат:

Age (cm)	**Height**
55	5
15	120
20	170
25	175

Формула за фиксни ефекат линеарног модела је:

```
> height  ~ age + error
```

error укључује све остале факторе и параметре који би могли утицати на резултат, али их нисмо свесни или их не узимамо у обзир намерно или недовољно.

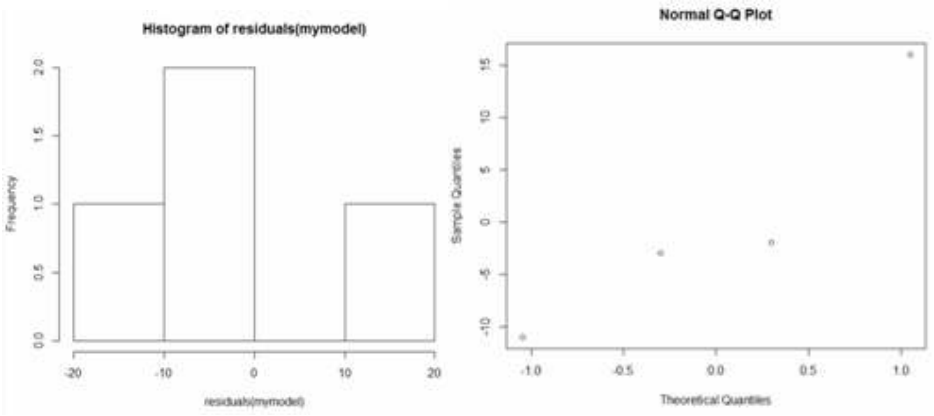
Линеарни модел нам показује релацију и корелацију између старости и висине, док се мора узети у обзир грешка, јер ако се стави раса и пол као параметар резултат је промењен. Постоје узорци који су краћи или виши због пола или расе.

У командној линији унети:

```
> height = c(55, 120, 170, 175)
> age = c(5, 15, 20, 25)
> mydf = data.frame(age, height)   # "data.frame" dodeljuje i kombinuje dve
promenljive jedna drugoj
> mymodel = lm(height~age, mydf) # "lm" je komanda za linearni model
> summary(mymodel) # "summary()" daje informacije o podacima
```

```
Call:
lm(formula = height ~ age, data = mydf)

Residuals:
  1   2   3   4
 -3  -2  16 -11

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  26.0000    16.8565   1.542   0.2629
age           6.4000     0.9442   6.779   0.0211 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 13.96 on 2 degrees of freedom
Multiple R-squared:  0.9583,    Adjusted R-squared:  0.9374
F-statistic: 45.95 on 1 and 2 DF, p-value: 0.02108

> hist(residuals(mymodel)) # pravi model histograma
> qqnorm(residuals(mymodel)) # dijagram reziduala kojim se uporedjuje nas skup
podataka sa normalnom distribucijom
```

Резидуал је разлика између стварне вредности висине и предвиђене висине. Команда `plot(age, height)` може илустровати разбацане тачке, док `plot(mymodel)` има више смисленог графикона о нашем моделу.



Литература

- Chambers (2008) Software for data analysis, Springer
- Chambers (1998) Programming with data, Springer
- Murrel (2005) R Graphics, Chapman & Hall / CRC Press
- Use R! Serija knjiga, Springer
- http://www.r-project.org/doc/bib/R-books.html
- The R Journal – http://journal.r-project.org/current.html.

Приручници су доступни и употребом функције `help.start()`.

```
> help.start()
```