Институт за математику и информатику
Природно-математички факултет
Универзитет у Крагујевцу

Софтверски алати 2

Увод у програмски језик R

Аутори

др Милош Ивановић
Татјана Бошковић

Крагујевац, децембар 2018.

Садржај

Глава 1

Увод

1.1 Дескриптивна статистика

Дескриптивна статистика садржи методе и процедуре за презентовање и сумирање података. Сврха дескриптивне статистике је да помоћу неколико бројева опише значење података који стоје иза њих. Подаци се добијају на основу опсервација на скупу различитих случајева који могу бити нпр. људи, животиње, градови, различити догађаји или нека комбинација свега наведеног. Дескриптивна статистика је обично први корак у анализи података, а служи за описивање прикупљених података.

Најчешће коришћене процедуре у дескриптивној статистици су графичко и табеларно приказивање података и израчунавање мера централне тенденције и варијабилитета. Већина аутора сврстава мере корелације и асоцијације променљивих у дескриптивну статистику, јер описују везу између две или више променљиве.

Једна од широко прихваћених платформи за статистичка израчунавања и графике је програмски језик и програмско окружење `R`. Употребљава се у припреми, визуелизацији и анализи података. Користи интерфејс командне линије, али и кроз више графичких корисничких окружења. `R` обезбеђује широк избор статистичких (линеарних и нелинеарних) модела, класичне статистичке тестове, анализу временских серија, класификацију, кластеризацију, итд. Пројектован је као прави програмски језик и омогућава корисницима додатну функционалност дефинисањем нових функција.

`R` може бити проширен, кроз пакете обезбеђене од стране корисника, за специфичне функције или специфичне области. Због свог `S` наслеђа, има бољу подршку за објектно-оријентисано програмирање него остали статистички програмски језици. Проширивост `R`-а је олакшана и његовим „лабавим” језичким опсегом. Следећа предност `R`-а су његове графичке могућности, које обезбеђују графике квалитета довољно доброг за публиковање које укључује математичке симболе. `R` поседује сопствени LaTeX-олики формат докумената, који се користи за представљање свеобухватне документације преко Интернета у бројним форматима или као штампана копија.

1.2 Шта је R?

Ова скрипта описује коришћење програмског пакета R у сврху савладавања предмета *Софтверски алати 2* на Природно-математичком факултету у Крагујевцу. Основна намера је омогућавање да се овај широко распрострањени софтвер искористи на курсевима нижег нивоа, где се често користе MINITAB, SPSS, Excel итд. Очекује се да читалац поседује знање добијено на било ком почетном академском курсу математике. Предности R за студента уводне статистике су:

- R је бесплатан. R је отвореног кода и ради на UNIX-у, *Windows*-у и *Mac*-у.
- R Има оуграђен систем за помоћ.
- R Има одличне графичке могућности.
- Језик R-а има моћну синтаксу, лаку за учење, са много уграђених статистичких функција.
- Језик је лако проширити функцијама које су писали његови корисници (CRAN библиотека).
- R је рачунарски програмски језик. Програмерима ће изгледати познато, а за нове кориснике рачунара следећи скок до програмирања неће бити тако велики.

Шта пак недостаје R-у у поређењу са другим софтверским решењима? Ево неколико ставки:

- Има ограничен графички интефејс. Ово значи да би учење могло да буде теже у почетку.
- Нема комерцијалне подршке (иако се може тврдити да је међународна мејлинг листа још боља у том смислу).
- Језик команди је програмски језик тако да студенти морају да науче да решавају проблеме заједно са синтаксом језика.
- R је статистичко окружење отвореног кода (GPL) израђено након S и S-а.

Језик S је развијен касних осамдесетих у AT&T лабораторијама. Пројекат R су започели *Robert Gentleman* и *Ross Ihaka* из Одсека за Статистику Универзитета у Оукланду 1995. године. Брзо је стекао широку публику. Тренутно га одржава R-развојни тим, огроман међународни тим програмера волонтера. Пројекат R хостован је на порталу `http://www.r-project.org` који је основни извор информација о R-у. На овом сајту налазе се упутства за прибављање софтвера, пратећих пакета и других извора документације.

1.3 Напомена о нотацији

У тексту се користи једноставна типографска конвенција која укључује различите фонтове за веб адресе, `R команде`, као и имена скупова података и различите поставке за дуге секвенце R наредби.

Глава 2

Рад са подацима

Статистика је наука која се бави проучавањем података. Након што смо успешно покренули R, прва ствар коју треба да урадимо је да научимо како да унесемо тражене податке и како да њима манипулишемо.

2.1 Покретање R

R се најлакше користи на тзв. интерактивни начин. Корисник поставља упит, а R моментално даје одговор. Упити се постављају и одговара се на командној линији. Да би се покренула командна линија R-а може се урадити следеће. У *Windows*-у се пронађе икона R и кликне, у *Unix*-у се са командне линије издаје команда R. Други оперативни системи могу имати другачије системе покретања. Такође се на свим значајнијим десктоп оперативним системима уместо стандардне конзоле може користити и бесплатни пакет RStudio, који је данас у великој мери и постао стандард за графичко окружење овог програмског језика. Једном када је R покренут, требало би да се појави нешто овако:

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```

```
[Previously saved workspace restored]

>
```

Знак „>” представља одзивни знак, дакле, не уписује га корисник. Ако је команда превише дуга да стане у један ред, користи се знак „+” за наставак.

2.2 Унос података помоћу команде `c()`

Најкориснија `R` команда за брзи унос релативно малих скупова података је функција `c()`. Ова функција комбинује или спаја термове. На пример, претпоставимо следећи скуп података који представља број грешака у куцању неког текста по странама неког уџбеника:

```
2 3 0 3 1 0 0 1
```

Ови подаци се у `R` сесију уносе следећим командама:

```
> typos = c(2,3,0,3,1,0,0,1)
> typos
[1] 2 3 0 3 1 0 0 1
```

Примећује се следеће:

- Додељују се вредности промењљивој названој `typos`.
- Оператор доделе је =. Овај оператор је валидан почевши од верзије 1.4.0 језика `R`. Претходно је то био (и још увек може бити) знак <-. Можемо користити било који од њих, мада је добро бити конзистентан и одлучити се за један од њих.
- Вредност промењљиве `typos` се не исписује аутоматски. Приказује се када се упише име промењљиве.
- Вредности промењљиве typos претходи [1]. Ово указује да је вредност `vector`. Више о томе касније.

2.3 Мање куцања

У многим имплементацијама језика `R` може се смањити непотребно куцање, ако се научи да стрелицама горе/доле можемо добити претходне команде. Посебно, свака команда се чува у историји и стрелица на горе ће пролазити уназад кроз историју, а стрелица на доле ће пролазити унапред. Лева и десна стрелица функционишу као што се очекује. Ово у комбинацији са мишем може знатно да олакша модификацију претходних команди.

2.4 Примена функције

R долази са многим уграђеним функцијама које могу да се примене на податке као што је typos. Једна од њих је функција mean() за налажење просечне вредности унетих података. Коришћење је једноставно:

```
> mean(typos)
[1] 1.25
```

Такође се може позвати median, или var да се пронађе медијана или варијанса. Синтакса је идентична – име функције коју прате заграде које садрже аргументе:

```
> median(typos)
[1] 1
> var(typos)
[1] 1.642857
```

2.5 Подаци су вектори

Подаци се у језику R чувају као vector. Ово једноставно значи да се прати редослед података оним редом којим су унети. Посебно постоји први елемент, други и све тако до последњег. Ово је добар приступ из више разлога:

- Овај једноставни вектор података typos има природан редослед – страна 1, страна 2 итд.
- Било би добро да подаци могу да се мењају део по део уместо да се уноси читав скуп података поново.
- Вектор је математички појам. Постоје природна проширења математичких операција, као што су сабирање и множење вектора, што олакшава рад са подацима када су дати као вектори.

Како се ово примењује на наш typos пример? Претпоставимо се да су ово грешке у кућању које важе за прву верзију откуцаног текста. Како пратити различите верзије како се број грешака мења (смањује)? Ово може да се постигне следећим командама:

```
> typos.draft1=c(2,3,0,3,1,0,0,1)
> typos.draft2=c(0,3,0,3,1,0,0,1)
```

Види се да су две грешке на првој страни исправљене. Примећују се два различита имена променљивих. За разлику од многих других језика, тачка се користи само за интерпункцију. Не може се користити _ (доња црта) да се означавају имена, као што се то може у неким другим програмским језицима, па је ово веома корисно [1].

[1] Доња црта се оригинално користила при додели, тако да би име као што је The_Data заправо доделило вредност променљиве Data променљивој The. Доња црта у новијим верзијама престаје да се користи и знак једнакости је замењује.

Неко би приметио да је много посла да се уносе подаци други пут. Зар не може R-у да се каже да само промени број грешака на првој страни? Одговор је наравно потврдан. Ево како:

```
> typos.draft1=c(2,3,0,3,1,0,0,1)
> typos.draft2=typos.draft1 # прави се копија
> typos.draft2[1] = 0 # првој страни се додељује 0 грешака
```

Примећује се неколико нових момената. Прво, карактер # се користи за коментаре. Све након овог знака се игнорише од стране R-а. Даље, додела прве вредности у вектор typos.draft2 се ради референцирањем првог елемента вектора. Ово се ради помоћу угластих заграда [], као и у неким другим програмским језицима. Важно је да се има на уму да су мале заграде () за позив функције, а угласте заграде [] за елементе вектора (и касније низове и листе). Конкретно, тренутно у typos.draft2 постоје следеће вредности:

```
> typos.draft2 # штампање вредности
[1] 0 3 0 3 1 0 0 1
> typos.draft2[2] # штампање вредности друге стране
[1] 3
> typos.draft2[4] # четврта страна
[1] 3
> typos.draft2[-4] # све осим четврте стране
[1] 0 3 0 1 0 0 1
> typos.draft2[c(1,2,3)] # штампање прве, друге и треће стране
[1] 0 3 0
```

Примећује се да негативни индекси дају све осим елемента тог индекса. Последњи пример је веома важан. Може се узети више од једне вредности користећи други вектор са нумерацијама индекса. Ово се још назива slicing (парчање).

Хајде сада да нађемо стране са стварно много грешака. Инспекцијом се може приметити да стране 2 и 4 представљају највећи проблем. Може ли се у R-у ово извести на систематичнији начин?

```
> typos.draft2 == 3
[1] FALSE  TRUE FALSE  TRUE FALSE FALSE FALSE FALSE
```

Примећује се употреба двоструког знака једнакости (==) у сврху поређења. Овим се тестирају све вредности промењљиве typos.draft2 на једнакост броју 3. Друга и четврта вредност добијају одговор TRUE док су остале негативне. Ово се може замислити као постављање питања да ли је вредност једнака 3? R одговара све одједном, и то другим вектором који садржи вредности TRUE и FALSE. Сада је питање како можемо да добијемо индексе (странице) које одговарају TRUE вредностима? Ако се питање преформулише, који индекси имају тачно три грешке? Команда which је решење:

```
> which(typos.draft2 == 3)
[1] 2 4
```

Друга идеја је да се креира нови вектор 1 2 3 ... који ће да бележи бројеве страна, и затим издвоји само оне за које важи `typos.draft2==3`:

```
> n = length(typos.draft2) # колико страна
> pages = 1:n # како се добија број страна
> pages # pages је једноставно од 1 до броја страна
[1] 1 2 3 4 5 6 7 8
> pages[typos.draft2 == 3] # logic издвајање. Врло корисно
[1] 2 4
```

За креирање вектора 1 2 3 ... коришћен је оператор „:”. Команда $a : b$ је просто $a, a + 1, a + 2, ..., b$ ако су a, b цели бројеви. Уопштенија `R` функција је `seq()` која захтева мало више куцања. Уколико се откуца `?seq` могу се видети њене опције. Да би се добило претходно наведено, требало би пробати `seq(a,b,1)`.

Употреба издвајања елемената вектора помоћу другог вектора исте дужине, који се састоји од `TRUE` и `FALSE` вредности назива се **екстракција логичким вектором**. Приметимо да је ово другачије од издвајања бројева страна парчањем, као што је рађено раније. Познавање употребе парчања и логичких вектора даје могућност да се једноставно приступи подацима. Наравно, могло је све ово да се уради и следећом командом:

```
>(1:length(typos.draft2))[typos.draft2 == max(typos.draft2)]
[1] 2 4
```

Међутим, овај начин је у доброј мери подложан грешкама, али одсликава како се изрази могу комбиновати у кратке моћне наредбе. Ово је важан закључак. Да би се ценила употреба језика `R`, требало би разумети како се комбинује излаз једне функције или операције са улазом друге. У математици се овај приступ назива композицијом.

Шта ако је потребно знати колико има грешака у куцању, или колико страна још увек има грешака које треба исправити, или пак која је разлика између верзија? Одговор на сва ова питања су математичке функције:

```
> sum(typos.draft2) # колико укупно има грешака
[1] 8
> sum(typos.draft2>0) # колико страна има са грешкама
[1] 4
> typos.draft1 - typos.draft2 # разлика између та два
[1] 2 0 0 0 0 0 0 0
```

2.5.1 Пример: Праћење вредности акција, проширивање података

Претпоставка је да су дневне завршне вредности акција на берзи за две недеље следеће: 45,43,46,48,51,46,50,47,46,45. Ове вредности могу да се бележе у језику `R` коришћењем вектора:

```
> x = c(45,43,46,48,51,46,50,47,46,45)
> mean(x) # средња вредност
```

```
[1] 46.7
> max(x) # највећа вредност
[1] 51
> min(x] # најмања вредност
[1] 43
```

Многе занимљиве функције могуће је израчунати на једноставан начин. Како се могу додати и неке друге вредности? Прво се додају вредности наредне две недеље у вектор x, нпр:

```
48,49,51,50,49,41,40,38,35,40
```

Њих додајемо вектору на следећи начин:

```
> x = c(x,48,49,51,50,49) # додаје вредности вектору x
> length(x) # колика је сад дужина вектора x (претходна дужина је 10)?
[1] 15
> x[16]=41 # додавање на одређени индекс
> x[17:20] = c(40,38,35,40) # додавање на више одређених
```

Примећује се да су извршене три различите операције да би се вектор проширио. Све су корисне, али их је потребно објаснити. Прво је коришћен `c` (од *combine*) оператор да се прошири претходна вредност x бројевима од следеће недеље. Затим је додељена вредност директно 16-том индексу. У време доделе, x је имао 15 чланова, па је аутоматски додат још један. Коначно, доделили смо вредности скупу индекса.

2.6 Графички интерфејси за унос података

Постоје и други начини за унос и уређивање података. Један од њих користи табелу као кориснички интерфејс. Ево примера са анотацијама:

```
> data.entry(x) # отвара табелу за унос података
> x=de(x) # исто, само што не чува податке
> x=edit(x) # користи едитор за измену x
> median(x) # средња вредност
```

Сви су једноставни за употребу, али забуна може наступити из потребе да променљива x претходно мора да буде дефинисана. На пример:

```
> data.entry(x) # резултује грешком, променљива није дефинисана
Error in de(..., Modes = Modes, Names = Names) : object 'x' not found
> data.entry(x=c(NA)) # сада ради, x је дефинисан у истој наредби
```

Сада можемо размотрити примену неких других уграђених функција на унете податке. Ево неколико примера. *Покретни просек* једноставно значи да се израчуна просек неког претходног броја дана. Претпоставимо да је потребан петодневни покретни просек. Наравно, могуће га је израчунати за дане од 5-ог до 20-ог, пошто за друге дане нема довољно података:

```
> day=5
> mean(x[day:(day+4)])
[1] 48
> day:(day+4) # трик је у томе што парчање издваја дане 5,6,7,8 и 9
[1] 5 6 7 8 9
```

Дакле, средња вредност узима само одређене вредности вектора x.

Следећи задатак који можемо поставити је израчунавање максималне вредности акција. Функција која се за то користи је `max(x)`. Уколико је потребно знати највећу вредност до неког датума, такође је једноставно ако занмо да `R` има уграђену функцију која се тиме бави. Та функција се назива `cummax` и враћа вредност кумулативног максимума. Ево резултата за податке од 4 недеље заједно са сличним `cummin`:

```
> cummax(x) # текући максимум
[1] 45 45 46 48 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51 51
> cummin(x) # текући минимум
[1] 45 43 43 43 43 43 43 43 43 43 43 43 43 43 43 41 40 38 35 35
```

2.6.1 Пример: Рад са математичким функцијама

`R` олакшава превођење математике на природнији начин, када податке већ имамо у меморији. На пример, узмимо да је годишњи број китова који су се насукали у Тексасу у периоду од 1990. до 1999. године следећи:

```
74 122 235 111 292 111 211 133 156 79
```

Која је средња вредност, варијанса и стандардна девијација? Поново, `R` овај посао чини тривијалним:

```
> whale = c(74, 122, 235, 111, 292, 111, 211, 133, 156, 79)
> mean(whale)
[1] 152.4
> var(whale)
[1] 5113.378
> std(whale)
[1] 71.50789
> sqrt(var(whale))
[1] 71.50789
> sqrt( sum( (whale - mean(whale))^2 /(length(whale)-1)))
[1] 71.50789
```

Прво, морају се запамтити имена функција. У овом случају, `mean` је лако погодити, `var` је прилично очигледно, мада мање, а за стандардну девијацију у примеру употребљен је квадратни корен варијансе. Коначно, последња линија илуструје да `R` може потпуно прецизно да опонаша математичку формулу за стандардну девијацију:

$$\sigma(X) = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2}$$

Приметимо да је сума $\sum$, $\overline{X}$ је `mean(whale)` и да се `length(x)` користи уместо n.

Једноставност дефинисања корисничких функција је врло привлачна карактеристика `R`-а која ће се касније често користити. У овом конкретном примеру за то ипак нема потребе, јер постоји и уграђена функција `sd()`. Она даје

```
> sd(whale)
[1] 71.50789
```

2.7 Приступ подацима

Има неколико начина да се извуку подаци из вектора. Ево прегледа коришћења парчања и екстракције помоћу логичког вектора. Претпоставимо да је x вектор података, нпр. $x = 1 : 10$.

колико елемената?	`length(x)`
i-ти елемент	$x[2] \quad (i = 2)$
сви осим i-тог елемента	$x[-2] \quad (i = 2)$
првих k елемената	$[1 : 5] \quad (k = 5)$
последњих k елемената	$x[(length(x) - 5) : length(x)] \quad (k = 5)$
одређени елементи	$x[c(1, 3, 5)]$ (Први, трећи и пети)
сви већи од неке вредности	$x[x > 3]$
мањи или већи од неке вредности	$x[x < -2 \vert x > 2]$
индекс највећег елемента	$which(x == max(x))$

Задаци

1. Претпоставимо да желимо да пратимо километражу сваки пут када сипамо гориво у аутомобил. На последњих 6 сипања километража је била: 65311 65624 65908 66219 66499 66821 67145 67447. Унесимо ове бројеве у `R` и употребимо функцију `diff` на подацима:

   ```
   > miles = c(65311, 65624, 65908, 66219, 66499, 66821, +
    67145, 67447)
   > x = diff(miles)
   ```

   Требало би да се добије број километара између пуњења. Користи се `max` да се пронађе максимални број километара између пуњења, функција `mean` налази просечан број километара, а помоћу `min` се добије минимални број километара између пуњења.

2. Претпоставимо да пратимо своју вожњу десет дана и налазимо следећа времена у минутима

   17 16 20 24 22 15 21 15 17 22

   Унесимо ово у `R`. Користимо функцију `max` да нађемо најдуже време вожње, функцију `mean` да нађемо просек и функцију `min` за минимум. Сада претпоставимо да је

24 била грешка. Требало је да буде 8. Како исправити? Урадимо то, и онда нађимо нови просек. Колико пута је ваша вожња трајала 20 минута или више? Рецимо:

```
> sum( commutes >= 20)
```

Шта се добија? Који проценат вожњи је трајао краће од 17 минута? Како можемо да одговоримо на ово помоћу језика `R`?

3. Рачун за мобилни телефон варира од месеца до месеца. Претпоставимо да је у години било рачуна са следећим износима

   46 33 39 37 46 30 48 32 49 35 30 48.

   Унети ове податке у променљиву `bill`. Користимо наредбу `sum` да пронађемо износ који сте потрошен на нивоу године. Који је најмањи износ који смо потрошили у једном месецу? Који је највећи? Колико месеци је рачун био већи од $40? Који је проценат ове појаве?

4. Желимо да купимо полован ауто и након одређеног истраживања добијамо следеће цене (претпоставимо да су сви аутомобили сличних карактеристика):

   9000 9500 9400 9400 10000 9500 10300 10200

   Наћи просечну вредност и упоредити је са проценом од $ 9500. Наћи минимум и максимум.

5. Који су резултати извршења следећих `R` команди? Претпоставимо да је

```
> x = c(1,3,5,7,9)
> y = c(2,3,5,7,11,13)
```

   (a) $x + 1$
   (b) $y * 2$
   (c) `length(x)` и `length(y)`
   (d) $x + y$
   (e) $sum(x > 5)$ и $sum(x[x > 5])$
   (f) $sum(x > 5|x < 3)$ # тумачити | као 'or', и & као 'and'
   (g) $y[3]$
   (h) $y[-3]$
   (i) $y[x]$ # Шта је NA?
   (j) $y[y >= 7]$

6. Нека подаци x буду задати са

```
> x = c(1, 8, 2, 6, 3, 8, 5, 5, 5, 5)
```

   Израчунати вредности следећих функција. Треба приметити да користимо $X1$ да означимо први елемент вектора x (који је 0) итд:

(a) $(X1 + X2 + \cdots + X10)/10$ (користимо `sum`)

(b) Пронађимо $log_{10}(Xi)$ за свако i. (Користимо `log` функцију која је предефинисана са основом e).

(c) Нађимо $(Xi-4.4)/2.875$ за свако i. (Треба урадити све одједном.)

(d) Нађимо разлику најмање и највеће вредности x. (Ради се о распону вектора. Може се користити `max` и `min` или уграђена наредба.)

Глава 3

Основне математичке структуре и операције

У овом поглављу ће бити дате смернице за употребу R-а за решавање задатака који се тичу линеарне алгебре, као и неколико примера из класичне математичке анализе. Ово дефинитивно није примарна намена овог програмског окружења, али студенту II године може помоћи у савладавању градива математичких предмета директном програмском демонстрацијом математичких концепата. Поред тога, у основи многих алгоритама Науке о подацима лежи баш линеарна алгебра.

3.1 Вектори и матрице

Вектор се може формитати на следећи начин:

```
> A=matrix(data = c(1:4))
> A
      [,1]
 [1,]    1
 [2,]    2
 [3,]    3
 [4,]    4
```

У R-у је подразумевано смештање података по колонама (као уз програмском језику Fortran, а супротно од програмског језика C). Сада ево илустрације формирања две идентичне дводимензионе матрице:

```
> B = matrix(data = c(1:16), ncol=4)
> C = matrix(data = c(1:16), nrow=4)
> C
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

Уколико желимо да матрицу попуњавамо по врстама, то се мора експлицитно укључити:

```
> D = matrix(data = c(1:16), nrow=4, byrow=T)
> D
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

Уколико постоји потреба, подаци могу и да се понављају:

```
> D = matrix(data = c(11:15), ncol=10, nrow=5)
> D
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]   11   11   11   11   11   11   11   11   11    11
[2,]   12   12   12   12   12   12   12   12   12    12
[3,]   13   13   13   13   13   13   13   13   13    13
[4,]   14   14   14   14   14   14   14   14   14    14
[5,]   15   15   15   15   15   15   15   15   15    15
```

Наредба `diag()` се може искористити на различите начине:

```
# Elementi glavne dijagonale matrice A
G = diag(A)
# Matrica cija dijagonala uzima vrednosti prosledjenog vektora
H = diag(c(1:5))
# Jedinicna matrica dimenzija 5x5
I = diag(5)
```

Транспоновање вектора (матрице) врши се наредбом `t()`:

```
A=matrix(c(1,3,2))
> A
     [,1]
[1,]    1
[2,]    3
[3,]    2
> t(A)
     [,1] [,2] [,3]
[1,]    1    3    2
```

Множење вектора скаларом, сабирање и одузимање се врше на интуитиван начин:

```
> a=matrix(c(1,3,2))
> b=2*matrix(c(1,3,2))
> a
```

```
     [,1]
[1,]    1
[2,]    3
[3,]    2
> b
     [,1]
[1,]    2
[2,]    6
[3,]    4
> a+b
     [,1]
[1,]    3
[2,]    9
[3,]    6
```

Транспоновање, множење скаларом, сабирање и одузимање се и за матрице постиже на исти начин као за векторе, па овде неће бити додатно објашњавано.

Да се подсетимо, **скаларни производ** два вектора врши се по следећем правилу:

$$\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

што се лако постиже наредбом `sum(a*b)`, док се **норма вектора** (интензитет вектора) рачуна по формули:

$$\|\mathbf{a}\| = \sqrt{\mathbf{a} \cdot \mathbf{a}} = \sqrt{\sum_{i=1}^{n} a_i^2},$$

што се у `R`-у постиже наредбом `sqrt(sum(a * a))`.

Множење матрице од r врста и c колона и колона вектора од c чланова изводи се по следећем правилу:

$$\mathbf{A} \cdot \mathbf{b} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1c} \\ a_{21} & a_{22} & \dots & a_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \dots & a_{rc} \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_c \end{bmatrix} = \begin{bmatrix} a_{11}b_1 + a_{12}b_2 + \cdots + a_{1c}b_c \\ a_{21}b_1 + a_{22}b_2 + \cdots + a_{2c}b_c \\ \vdots \\ a_{r1}b_1 + a_{r2}b_2 + \cdots + a_{rc}b_c \end{bmatrix}$$

Срећом, програмски се ова операција изводи једним јединим оператором:

```
> A=matrix(c(1,3,2,2,8,9), nrow=3)
> A
     [,1] [,2]
[1,]    1    2
[2,]    3    8
[3,]    2    9
> a=matrix(c(5,8))
> A%*%a
     [,1]
```

```
[1,]   21
[2,]   79
[3,]   82
```

Множење две матрице се обавља по сличном принципу, а истим оператором као у претходном случају:

$$\mathbf{A} \cdot \mathbf{B} = \begin{bmatrix} 1 & 2 \\ 3 & 8 \\ 2 & 9 \end{bmatrix} \cdot \begin{bmatrix} 5 & 4 \\ 8 & 2 \end{bmatrix} = \begin{bmatrix} 21 & 8 \\ 79 & 28 \\ 82 & 26 \end{bmatrix},$$

или програмски:

```
> A <- matrix(c(1, 3, 2, 2, 8, 9), ncol = 2)
> B <- matrix(c(5, 8, 4, 2), ncol = 2)
> A %*% B
     [,1] [,2]
[1,]   21    8
[2,]   79   28
[3,]   82   26
```

Детерминанта матрице рачуна се функцијом `det()`, док се **инверзна матрица** ($\mathbf{AB} = \mathbf{BA} = \mathbf{I}$) добија функцијом `solve()`. Треба водити рачуна да су и детерминанта матрице и инверзна матрица дефинисане само за квадратне матрице. Ево примера за рачунање инверзне матрице:

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}, \qquad \mathbf{B} = \mathbf{A}^{-1} = \begin{bmatrix} -2 & 1.5 \\ 1 & -0.5 \end{bmatrix}$$

или једноставно:

```
> A=matrix(c(1,2,3,4),ncol=2)
> solve(A)
     [,1] [,2]
[1,]   -2  1.5
[2,]    1 -0.5
```

Управо уведена наредба `solve()` заправо је првенствено намењена **решавању система линеарних једначина**, јер се системи линеарних једначина заправо могу представити у матричном облику. На пример:

$$\begin{aligned} x_1 + 3x_2 &= 7 \\ 2x_1 + 4x_2 &= 10 \end{aligned}$$

може да се напише као:

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 7 \\ 10 \end{bmatrix}, \text{ или } Ax = b,$$

Ако ову једначину са леве стране поможимо са $\mathbf{A}^{-1}$ ($\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$), онда:

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b} = \begin{bmatrix} -2 & 1.5 \\ 1 & -0.5 \end{bmatrix} \cdot \begin{bmatrix} 7 \\ 10 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Ово се директно транслира у програмски код:

```
> A = matrix(c(1, 2, 3, 4), ncol = 2)
> b = c(7, 10)
> x = solve(A) \%*\% b
> x
     [,1]
[1,]    1
[2,]    2
```

Заправо, операција инверзије матрице је прилично скупа са становишта времена и рачунарских ресурса, и пожељно ју је избећи када год је то могуће. Срећом, за сврху решавања система линеарних једначина постоје и друге, мање скупе, методе, као што је Гаусова елеминација, која је такође уграђена у команду `solve()`, само што се сада позива са додатним аргументом вектора десне стране:

```
> solve(A,b)
[1] 1 2
```

3.2 Нумеричко одређивање вредности извода и интеграла

При решавању различитих проблема науке и инжењерства, понекад је веома тешко доћи до тачне вредности извода или одређеног интеграла неке функције. Зато се примењују приближне нумеричке методе које је веома једноставно имплементирати на рачунару, а дају довољно тачна решења за практичну примену.

3.2.1 Извод функције

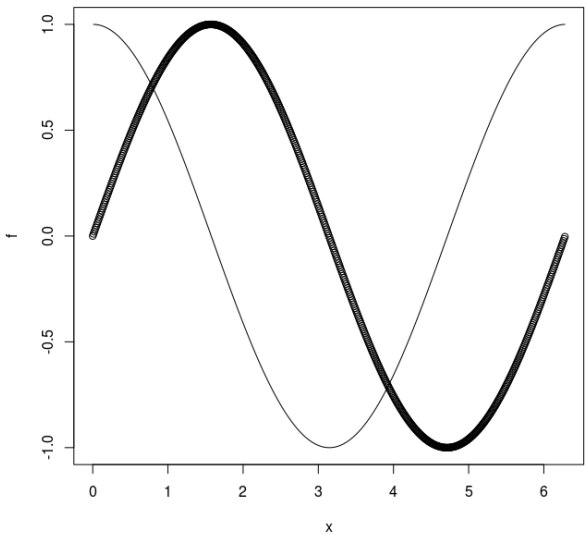
Формула за нумеричко рачунање извода следи директно из дефиниције извода функције:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \frac{df}{dx}$$

Ако диференцијале df и dx заменимо коначним прираштајима Δx, і Δf респективно, ево начина да се израчуна извод функције у `R`-у:

```
> x=seq(0,2*pi,0.01)
> f=sin(x)
> fprim=diff(f)/diff(x)
> plot(x,f)
> lines(x[-1],fprim)
```

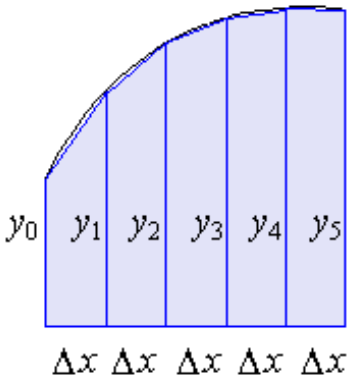
Дакле, направили смо довољно ситну поделу по x, и уз проглашавање $\Delta x = x_{i+1} - x_i$ и $\Delta f = f(x_{i+1}) - f(x_i)$, добили извод функције $sin(x)$, као што се види на Слици 3.1. Очигледно је да смо као извод добили функцију $cos(x)$. Поступак можемо поновити за било коју функцију која је диференцијабилна у датом интервалу, при чему треба водити рачуна да подела по x буде довољно ситна.



Слика 3.1: Извод функције $sin(x)$ добијен нумеричким путем по дефиницији

3.2.2 Рачунање одређеног интеграла методом трапеза

Како је вредност одређеног интеграла једнака површини између криве и апсцисе, приступићемо рачунању вредности интеграла на сличан начин као што смо то урадили у случају извода - поделом на релативно мале интервале (видети Слику 3.2). Вредност повр-



Слика 3.2: Трапезно правило за интеграцију

шине испод криве се може апроксимирати сумом површина правоуглих трапеза:

$$I = \int_a^b f(x) \approx \sum_{i=1}^{N} P_i, \qquad P_i = \Delta x \frac{f(x_i) + f(x_{i+1})}{2}.$$

Дакле, ако развијемо суму, добијамо:

$$I = \Delta x \cdot \left(\frac{f(x_0) + f(x_1)}{2} + \frac{f(x_1) + f(x_2)}{2} + \cdots + \frac{f(x_{N-1}) + f(x_N)}{2} \right)$$
$$I = \Delta x \cdot \left(\frac{f(x_0)}{2} + f(x_1) + \cdots + \frac{f(x_N)}{2} \right),$$

што се сада лако решава програмски, рецимо за функцију $sin(x)$ у границама од 0 до π:

```
> x=seq(0,pi,length=100)
> deltax=pi/(length(x)-1)
> f=sin(x)
> koef=rep(1,100)
> koef[1]=0.5
> koef[length(koef)]=0.5
> sum(koef*f*deltax)
1.9998
```

и што се прилично добро уклапа са стварном вредношћу интеграла која се може израчунати аналитички и износи 2.

Глава 4

Униваријантни подаци

Постоји разлика између типова података у статистици и `R` препознаје неке од ових разлика. Конкретно, подаци могу узети један од три основна типа: **категорички, дискретни нумерички и континуални нумерички**. Метод за преглед и сумирање података зависи од типа, тако да је потребно знати како се сваки од њих понаша и шта може да се уради са њим.

Категорички подаци су подаци који бележе категорије. Рецимо, анкета која бележи да ли је особа за или против неког предлога. Попис становништва укључује неколико питања чији је одговор категоричког типа. На пример, једно питање везано за расу на попису 2000-те године у Сједињеним Државама укључивало је 15 категорија са простором за дописивање још 3 вредности за ову ставку (вишерасни случајеви). Још један пример може бити медицински картон на коме се бележе подаци о пацијенту. Пол или историја одређене болести се могу третирати као категорије.

Међутим, старост особе и њена телесна тежина су нумеричке вредности. Старост је дискретна вредност (типично), као и телесна тежина. Ови бројеви се обично бележе као цели. Ако би неко морао да прецизира, теоријски би могао да узме и континуалне вредности. За неке статистичке тестове показује се да је важно знати да ли подаци могу да имају међусобне везе, када два или више података показују на исту вредност. Када дискретне вредности представљамо графички, то чинимо помоћу тачака, док континуалне случајне променљиве представљамо помоћу линија.

Једноставни, интуитивни начин да се одабере да ли представити променљиву као дискретну или континуалну је да се потражи средња вредност (просек). Ако он нема смисла, онда су подаци сигурно категорички (као што је нпр. просек пушача и непушача). Ако просек има смисла, али га треба заокружити (рецимо 18,5 за године у случају када бележимо само целе бројеве), онда су подаци дискретни нумерички. Ато није случај, подаци су вероватно континуални.

4.1 Категорички подаци

Често представљамо категоричке податке табеларно, али ову врсту података такође можемо посматрати и графички кроз стубне или кружне дијаграме.

4.1.1 Употреба табела

Команда `table` омогућава формирање табеле. Њена најједноставнија употреба је једноставно `table(x)` где је x категоричка променљива.

Пример: Анкета о пушењу

Анкета поставља испитаницима питање да ли пуше или не. Подаци су следећи:

```
Yes, No, No, Yes, Yes
```

Можемо да унесемо податке у `R` помоћу стандардне команде `c()`, и сумирамо их употребом команде `table`, као што следи:

```
> x=c("Yes","No","No","Yes","Yes")
> table(x)
x
No Yes
2 3
```

Команда `table` једноставно израчунава фреквенцију појављивања сваке јединствене вредности података, што ћемо касније представити и графички.

4.1.2 Фактори

Категорички тип се често користи да би се подаци класификовали на различите нивое или факторе. На пример, подаци о пушењу би могли да буду део шире анкете о студентским здравственим проблемима. `R` има специјалну класу за рад са факторима коју је повремено битно познавати, обзиром да се `R` аутоматски прилагођава када препозна да су подаци факторског типа. Једноставно је од података направити фактор командом `factor()` или `as.factor()`. Приметимо разлику у начину на који `R` третира факторе кроз овај пример:

```
> x=c("Yes","No","No","Yes","Yes")
> x               # штампа вредности из x
[1] "Yes" "No"  "No"  "Yes" "Yes"
> factor(x)       # штампа вредности из factor(x)
[1] Yes No  No  Yes Yes
Levels: No Yes    # приметимо да су нивои иштампани
```

4.1.3 Дијаграми

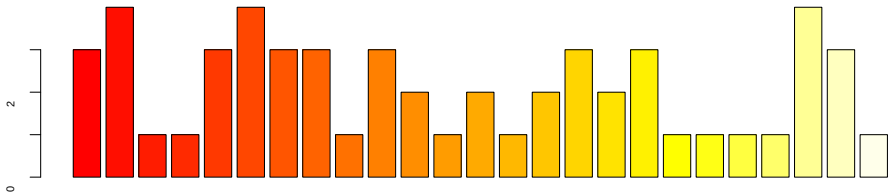
Стандардни стубни дијаграм (*barplot*) црта правоугаонике са висином пропорционалном вредности у табели. Висина може да буде задата фреквенцијом или пропорцијом. График ће изгледати исто, али размера може бити другачија.

Анкетирана је група од 25 људи у вези њихових преференција у испијању пива. Категорије су биле (1) домаће у лименци, (2) домаће флаширано, (3) домаће занатско пиво и (4) увозно. Сирови подаци су:

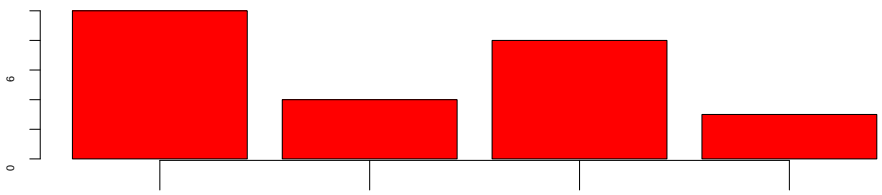
```
3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
```

Потребно је направити график фреквенција и пропорција. Прво се позива функција `scan()` да се учитају подаци, а затим се цртају (Слика 4.1):

```
> beer = scan()
1: 3 4 1 1 3 4 3 3 1 3 2 1 2 1 2 3 2 3 1 1 1 1 4 3 1
26:
Read 25 items
> barplot(beer)                # ово није тачно
> barplot(table(beer)) # ок, позив са сумираним подацима
barplot(table(beer)/length(beer))   # поделимо са n за пропорцију
```



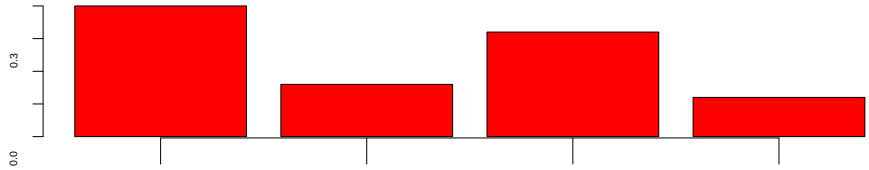
Слика 4.1: Пример за `barplot` - добили смо 25 категорија уместо 4



Слика 4.2: Пример за `barplot` - фреквенције

Дијаграми су дати на сликама 4.1, 4.2 и 4.3. Приметимо неколико ствари:

- Користили смо функцију `scan()` да бисмо учитали податке. Ова команда је врло корисна за учитавање података из фајла или са тастатуре. Команда `?scan` даје више информација, али основна употреба је прилично једноставна. Подаци се укуцавају редом, а ознака за крај је празан ред.



Слика 4.3: Пример за `barplot` - пропорције

- Подразумевана шема боја и није на највишем естетском нивоу.
- Направили смо 3 дијаграма. Први приказује да није упутно користити `barplot` на сировим подацима. Други приказује употребу `table` команде да би се креирали сумирани подаци, а резултат се прослеђује функцији `barplot`, креирајући дијаграм фреквенција.
- Коначно, команда

```
    > table(beer)/length(beer)
beer
   1    2    3    4
0.40 0.16 0.32 0.12
```

  даје пропорције. Поделили смо бројем података (25) или са `length(beer)`. Резултат је затим прослеђен функцији `barplot` да би се добио график. Приметимо да има идентичан облик као претходни, али ордината сада има опсег између 0 и 1, пошто мери пропорцију, а не фреквенцију.

4.1.4 Кружни дијаграми

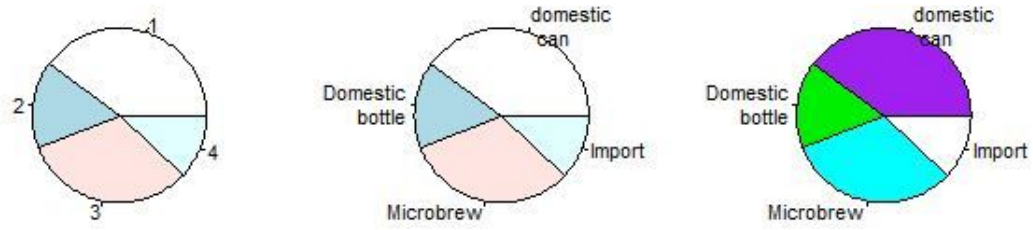
Исти подаци се могу проучавати и кроз кружне дијаграме (*piechart*) користећи `pie` функцију[1]. [2] Ево неких једноставних примера који илуструју употребу која је слична функцији `barplot()`, али са неким додатним карактеристикама:

```
> beer.counts = table(beer) # сачувамо резултат table
> pie(beer.counts) # први кружни дијаграм
> names(beer.counts) = c("domestic\n can","Domestic\n bottle", +
"Microbrew","Import")  # додавање имена
> pie(beer.counts) # штампање имена
```

[1] Пре верзије 1.5.0 ова функција се звала `piechart`

[2] Професионални статистичари све мање користе кружне дијаграме. Ипак се и даље често појављују у медијима. Интересантан уреднички коментар је направљен на страници за помоћ за кружни дијаграм. Можемо пробати `?pie` да бисмо видели овај коментар.

```
> pie(beer.counts,col=c("purple","green2","cyan", +
"white")) # сада у јарким бојама
```



Слика 4.4: Пример кружних дијаграма

Први дијаграм на слици 4.4 је био неинформативан, па су додата имена. Ово је урађено функцијом `names` која омогућава да се доделе имена категоријама. Резултујући кружни дијаграм показује како су имена употребљена. Коначно, додате су и упадљиве боје. Ово је постигнуто постављањем атрибута col. Изједначава се са вектором имена боја који је исте дужине као полазни вектор података. Команда за помоћ (`?pie`) даје неке примере за аутоматско добијање различитих боја, нарочито коришћењем `rainbow` и `gray`. Уочава се коришшћење додатних аргумената функције `pie`. Синтакса за њих је `ime=vrednost`. Могућност да се проследе именоване вредности функцији у великој мери олакшава рад у односу на језике као што су *C* или *Java*. Додатни бенефит је што је тако могуће креирати мањи број функција, како свака понаособ може да има више функционалности.

4.2 Нумерички подаци

Постоји пуно могућности за преглед нумеричких података. Прво ћемо разматрати стандардне нумеричке појмове центра и расипања.

4.2.1 Нумеричке мере центра и расипања

Да би се описала нека дистрибуција потребно је знати где је њен центар и какво је расипање око центра. Они се обично мере средњом вредношћу и варијансом (или стандардном девијацијом) или медијаном и карактеристичном петорком узорка. `R` команде за поменуте мере су, респективно, `mean`, `var`, `sd`, `median`, `fivenum` и `summary`.

Пример: Директорске плате

Извршено је узорковање годишњих компензација директора и добијене се следеће цифре (у милионима евра): 12 0.4 5 2 50 8 3 1 4 0.25.

```
> sals=scan() # учитавамо податке помоћу функције scan
1: 12 .4 5 2 50 8 3 1 4 0.25
11:
Read 10 items
> mean(sals) # просек
[1] 8.565
> var(sals) # варијанса
[1] 225.5145
> sd(sals) # стандардна девијација
[1] 15.01714
> median(sals) # медијана
[1] 3.5
> fivenum(sals) # минимум, доњи квартил, медијана, горњи квартил, максимум
[1]  0.25  1.00  3.50  8.00 50.00
> summary(sals)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.250   1.250   3.500   8.565   7.250  50.000
```

Обратимо пажњу на команду `summary`. За нумеричку променљиву штампа скуп пет вредности и медијану. За друге врсте променљивих прилагођава се на интелигентан начин.

Разлика између карактеристичне петорке и квантила

Може се приметити софистицирана разлика између команди `fivenum` и `summary`. Конкретно, једна даје 1.00 за доњи квартил, а друга 1.250 за први квантил. У чему је разлика?

Медијана је тачка у подацима која их дели на половину. То значи да је половина података изнад ње, а половина је испод. На пример, ако су дати подаци у сортираном низу:

```
10, 17, 18, 25, 28
```

јасно је да је 18 број у средини и да су два броја већа од њега, а два мања. Ако подаци имају додатну тачку

```
10, 17, 18, 25, 28, 28
```

онда им је средина негде између 18 и 25, јер су 3 већа и 3 мања. Rачуна се просек ове две вредности што износи 21,5 за медијану. Дакле, тачка у којој су подаци подељени на половине зависи од броја тачака података. Ако је број тачака непаран, онда тачка података са индексом $(n+1)/2$ у сортираном низу одговара медијани. Ако је број података паран поново се користи тачка индекса $(n+1)/2$, али пошто је то децимални број, рачуна се средња вредност тачака лево и десно.

Идеја о **квантилу** генерализује појам медијане. **p-квантил** (такође познат као 100--процентни перцентил), је тачка у где је 100p% података (у процентима) мање, а 100(1-p)% (у процентима) веће. Ако постоји n тачака података, онда се -квантил налази на позицији $1+(n-1)p$ са пондерисаним просеком, ако је то између целих бројева. На пример, .25 квантил бројева 10,17,18,25,28,28 се јавља на позицији 1+(6-1)(.25) = 2.25. То је 1/4 пута између другог и трећег броја, што је у овом примеру 17.25.

Квантилима 0,25 и 0,75 означени су **квартили**. Први квартил се означава са $Q1$, а трећи квартил са $Q3$. Може да се помисли да ће се други квартил означавати са $Q2$, али користи се назив медијана. Ове вредности израчунавају се функцијом `R`-а `summary`. Уопштено, постоји функција квантила која ће израчунати било који квантил између 0 и 1. За рачунање наведених квантила можемо да урадимо следеће:

```
> data=c(10, 17, 18, 25, 28, 28)
> summary(data)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  10.00   17.25   21.50   21.00   27.25   28.00
> quantile(data,.25)
  25%
17.25
> quantile(data,c(.25,.75)) # две вредности р одједном
  25%   75%
17.25 27.25
```

Постоји историјски популаран скуп алтернатива квартилима, назван *hinge*, које је нешто лакше рачунати мануелно. Медијана је дефинисана на исти начин како је горе наведено. Доњи *hinge* је тада медијана свих података лево од средине. Горњи *hinge* се дефинише као медијана свих података десно од средине. На пример, ако су наши подаци поново 10, 17, 18, 25, 28, 28, онда је медијана 21,5, а доњи *hinge* је медијана од 10, 17, 18 (што је 17), а горњи *hinge* је медијана од 25,28,28 што је 28. Они су доступни у функцији *fivenum*, а касније се појављују и у функцији *boxplot*.

Ево илустрације са подацима `sals`, којих има $n = 10$. На пример, доњи *hinge* је медијана скупа података лево од медијане (0,25, 0,40, 1,00, 2,00, 3,00). Први квратил ($Q1$) је, са друге стране, на месту број 1+0,25*9=3,25, а то је четвртина пута између бројева 1 и 2, тј. 1,25.

```
> sort(sals)
 [1]  0.25  0.40  1.00  2.00  3.00  4.00  5.00  8.00 12.00 50.00
> fivenum(sals) # напомена да је 1 трећа вредност, а 8 је осма
[1]  0.25  1.00  3.50  8.00 50.00
> summary(sals) # вредност на 3.25 месту је 1/4 пута између 1 и 2
   Min. 1st Qu.  Median   Mean  3rd Qu.    Max.
  0.250   1.250   3.500   8.565   7.250  50.000
```

4.2.2 Отпорне мере центра и расипања

Најчешће коришћене мере центра и ширења су средња вредност и стандардна девијација услед њихове везе са нормалном расподелом. Међутим, проблем настаје када подаци имају дуге репове или пуно нетипичних вредности. Различите мере центра и расипања су развијене како би се овај проблем решио. **Медијана** је управо таква мера. Медијана је отпорна када је у питању неколико произвољно великих вредности. Рецимо, ако направимо грубу грешку и у мерењу добијемо 1000000 за највећу вредност уместо 10, медијана ће остати непромењена.

Доступне су и неке друге мере отпорне на нетипичне вредности. Једна од уобичајених за центар је функција `trim`. Она је корисна ако подаци имају пуно нетипичних вредности (као подаци директорских плата, иако је боље када су подаци симетрични). Одбацујемо одређени проценат података са врха и дна, а затим узимамо просек. Да бисмо то учинили у `R`-у, морамо да кажемо функцији `mean()` колико треба да *тримује*.

```
> mean(sals,trim=1/10) # сечемо 1/10 и са врха и са дна
[1] 4.425
> mean(sals,trim=2/10)
[1] 3.833333
```

Ако се све више и више сече вредност, средина се приближава медијани која се заправо и достиже када је $trim = 1/2$. Поново се примећује коришћење именованог аргумента у функцији `mean`.

Мере расипања, као што су варијанса и стандардна девијација су такође осетљиве на нетипичне вредности. Отпорне мере расипања укључују `IQR` и `mad`.

```
    > IQR(sals)
[1] 6
```

`IQR` представља разлику између трећег и првог квартила. Средње просечно одступање (`mad`) је такође корисна, отпорна мера расипања. Налази средњу вредност апсолутних разлика од медијане, а затим се множи константом. Ево формуле:

$$median|X_i - median(X)| \cdot 1.4826$$

Потребно је пронаћи медијану, а затим све разлике од медијане. Од тако начињеног вектора се израчуна апсолутна вредност и потом пронађе средина овог новог скупа података. Добијена вредност се множи константом. На срећу, у `R`-у постоји уграђена функција за ову сврху:

```
> mad(sals)
[1] 4.15128
```

Ево и провере мануелном методом:

```
> median(abs(sals - median(sals))) # без нормализације константе
[1] 2.8
>  median(abs(sals - median(sals)))*1.4826
[1] 4.15128
```

Избор константе 1,4826 чини добијену вредност упоредивом са стандардном девијацијом за нормалну расподелу.

4.2.3 Дијаграми стабла и листова

Ако је скуп података релативно мали, дијаграм стабла и листова је веома користан за посматрање облика дистрибуције и вредности појединачних података, али је потребно мало привикавања. Број на левој страни траке је стуб, а број на десној цифра. Спојили смо их

како бисмо обавили посматрање. Претпоставимо да имамо укупан резултат кошаркашке утакмице и следеће бројеве поена за играче у оба тима:

```
2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5
```

Креирање дијаграма стабљике и листа је једноставно

```
> scores=scan()
1: 2 3 16 23 14 12 4 13 2 0 0 0 6 28 31 14 4 8 2 5
21:
Read 20 items
> apropos("stem") # шта је тачно назив?
[1] "R_system_version" "stem"    "system"   "system.file"
[5] "system.time"    "system2"
> stem(scores)

  The decimal point is 1 digit(s) to the right of the |

  0 | 000222344568
  1 | 23446
  2 | 38
  3 | 1
```

Основе R-а: `help`, `?` и `apropos`

Примећује се да се `apropos` користи да се нађе конкретно име за функцију. Тако видимо да је прави назив `stem()`, а није `stemleaf()` или нешто слично. `apropos()` је погодан када претпостављамо назив функције, али нисмо сигурни. Команда `help` помаже да нађемо помоћ за дату функцију или скуп података када знамо назив. На пример, `help(stem)` или скраћено `?stem` приказаће документацију за функцију `stem`. Претпоставимо да смо желели да разврстамо категорије у групе од по 5. То можемо урадити постављањем тзв. скале:

```
stem(scores, scale=2)

  The decimal point is 1 digit(s) to the right of the |

  0 | 000222344
  0 | 568
  1 | 2344
  1 | 6
  2 | 3
  2 | 8
  3 | 1
```

Пример: Претварање нумеричких података у категоричке

Категорички подаци могу настати од нумеричких груписањем интервала вредности. На пример, директорске плате из једног од претходних скупова података се могу сврстати у категорије од 0-1 милиона, од 1-5 милиона и преко 5 милиона евра. Да би се то урадило у `R`-у, користе се функције `cut()` и `table()`. Претпоставимо да су плате поново:

```
12 0.4 5 2 50 8 3 1 4 0.25,
```

и да треба да се сврстају у интервале $[0,1],(1,5],(5,50]$. Да би се користила функција `cut()`, неопходно је дефинисати карактеристичне тачке које деле интервале. У овом случају то су 0, 1, 5 и 50=(`max(sals)`). Синтакса је следећа:

```
> sals=c(12 .4 5 2 50 8 3 1 4 .25) # уношење података
> cats = cut(sals,breaks=c(0,1,5,max(sals))) # спецификација интервала
> cats # преглед интервала
 [1] (5,50] (0,1]  (1,5]  (1,5]  (5,50] (5,50] (1,5]  (0,1]  (1,5]  (0,1]
Levels: (0,1] (1,5] (5,50]
> table(cats) # организовање
cats
 (0,1]  (1,5] (5,50]
     3      4      3
> levels(cats)=c("poor","rich","rolling in it") # мењамо ознаке
> table(cats)
cats
         poor          rich rolling in it
            3             4             3
```

Напомена: функција `cut` одговара на питање *у ком је интервалу дата вредност?* Излаз је интервал дефинисан као `factor`. Због тога се команда `table` користи за резимирање резултата функције `cut`. Поред тога, имена интервала су промењена као илустрација како њима можемо једноставно манипулисати.

4.2.4 Хистограм

Ако имамо превелики број података или циљни аудиторијум не познаје како се чита дијаграм стабла и листова, може се покушати другим типом дијаграма. Најчешћи је на први поглед сличан стубном дијаграму и назива се **хистограм**. Хистограм дефинише секвенцу интервала, а затим пребројава податке који се уклапају у задате интервале. Ово је слично употреби функције `cut()`. Слично је као и на стубном дијаграму, али се стубићи додирују. Вредност висине може бити фреквенција или пропорција. У другом случају површине се сумирају на 1 - особина која звучи познато када се проучавају расподеле вероватноће. У сваком случају, површина је пропорционална вероватноћи.

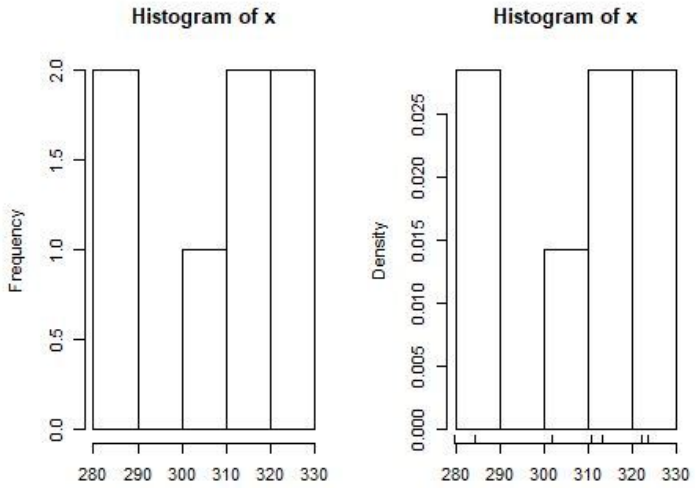
Почнимо са једноставним примером. Нека су 25 најбоље рангираних филмова направили следеће приходе за једну седмицу[3]:

[3]Подаци су доступни на movieweb.com (http://movieweb.com/movie/top25.html)

```
29.6 28.2 19.6 13.7 13.0 7.8 3.4 2.0 1.9 1.0 0.7 0.4 0.4 0.3
0.3 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1
```

Подаци се прво скенирају, а затим се црта хистограм:

```
> scan()
1: 29.6 28.2 19.6 13.7 13.0 7.8 3.4 2.0 1.9 1.0 0.7 0.4 0.4 0.3
15: 0.3 0.3 0.3 0.3 0.2 0.2 0.2 0.1 0.1 0.1 0.1 0.1
27:
Read 26 items
 [1] 29.6 28.2 19.6 13.7 13.0  7.8  3.4  2.0  1.9  1.0  0.7
 0.4  0.4  0.3  0.3  0.3  0.3  0.3  0.2  0.2  0.2  0.1  0.1
 0.1  0.1  0.1
> hist(x) # фреквенције
> hist(x,probability=TRUE) # пропорције (или вероватноће)
> rug(jitter(x)) # додају се подебљане ознаке
```

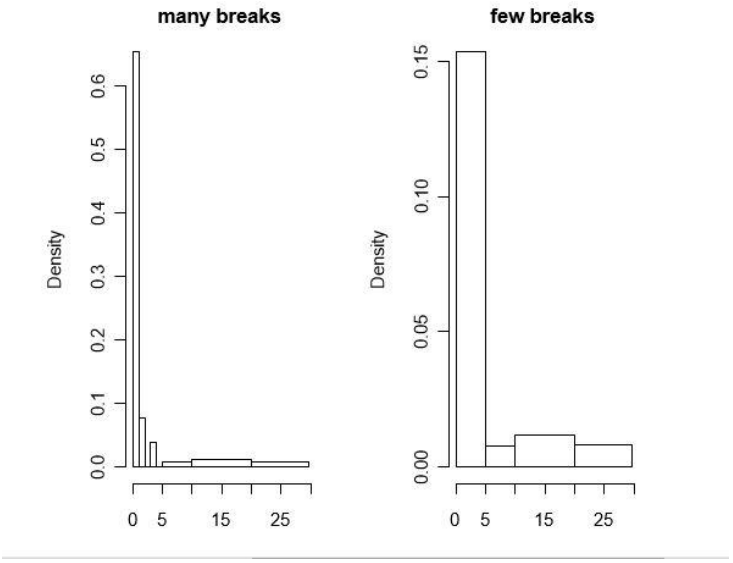


Слика 4.5: Пример за хистограм - фреквенције (лево) или пропорције (десно)

На слици 4.5 приказана су два графика (али не за скуп података о филмовима). Први је подразумевани график који приказује фреквенције. Други приказује хистограм пропорција чија је укупна површина јединична. Ово је пожељније, јер очигледније корелише са појмом густине расподеле. Једина разлика је на *y* оси. Леп додатак хистограму је исцртавање тачака командом `rug`. Коришћен је да би се добиле подебљане ознаке одмах изнад x осе. Ако су подаци дискретни и повезани, онда ће команда `rug(jitter(x))` додати уочљивост појединачним вредностима x-а.

Примећује се да ове команде отварају посебан прозор графика, ако се користи стандардно окружење (*RStudio* има посебан одељак за графике). График у `R`-у има неколико опција које су доступне употребом миша, али многи користе опције командне линије. `ggplot2` има много више опција, али захтева додатну инсталацију пакета. Основни хистограм има унапред дефинисан скуп тачака прекида (интервале). Ако је потребно, можемо одредити број прекида или чак задати сопствене тачке прекида (слика 4.6).

```
> hist(x,breaks=10) # 10 прекида или само hist(x,10)
> hist(x,breaks=c(0,1,2,3,4,5,10,20,max(x))) # специфичне тачке прекида
```



Слика 4.6: Хистограми са специфицираним тачкама прекида

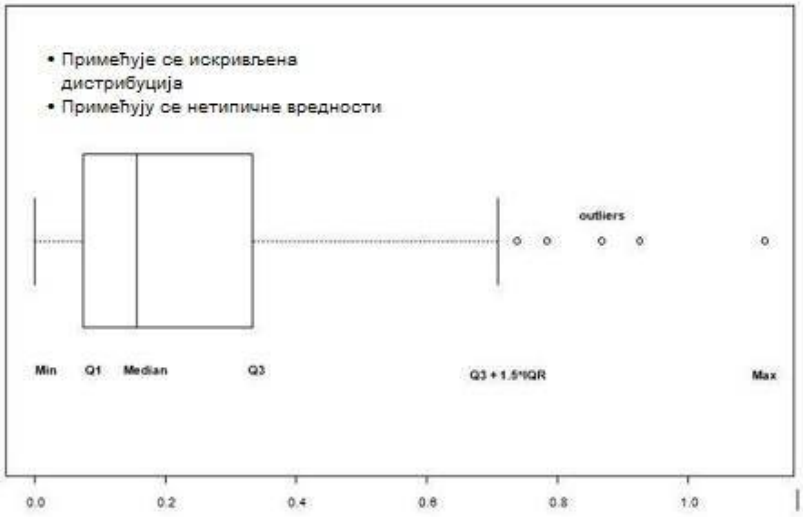
Из хистограма могу лако да се погоде средња вредност, медијана и IQR. Да би се то учинило, потребно је знати да медијана дели хистограм на два једнака дела површине, средња вредност би била тачка у којој би хистограм балансирао, а IQR бележи тачно средњу половину података.

4.2.5 Боксплот

Боксплот (слика 4.7) служи за упрошћавање сложеног скупа података, брзим приказом да ли су подаци симетични, или ако се сумња на постојање нетипичних вредности. Заснован је на карактеристичној петорци узорка, која је објашњена у претходном тексту. У његовој најједноставнијој употреби, боксплот има кутију са линијама на доњем $hinge$-у (у основи $Q1$), медијани, горњем $hinge$-у (у основи $Q3$) и брковима који се продужавају до минимума и максимума. Да би се показале могући нетипичне вредности, усвојена је конвенција како би се бркови скратили до дужине од 1,5 пута дужине кутије. Подаци који се нађу изван тог опсега су исцртани појединачним тачкама. Ови подаци се могу означити и другачије ако се простиру и изван три (3) дужине кутије. Стога, боксплотови дозвољавају брзу проверу симетрије (облик изгледа неуравнотежено) и нетипичних вредности (пуно тачака података изван брковa). На слици 4.7 видимо искривљену дистрибуцију са дугим репом.

Пример: Приходи филмова

У овом примеру, посматрају се подаци о приходима за 25 највећих филмова у датој седмици. Успут, уводи се и начин „читања” уграђеног скупа података. Овде су постављени



Слика 4.7: Типичан боксплот

подаци из скупова података `UsingR` [4].

```
> install.packages("UsingR")
> library("UsingR") # прочитати у библиотеци напомене
> data(movies) # читати у скупу података за бруто
> names(movies)
[1] "title" "current" "previous" "gross"
> attach(movies) # да приступите горе наведеним називима
> boxplot(current,main="current receipts",horizontal=TRUE)
> boxplot(gross,main="gross receipts",horizontal=TRUE)
> detach(movies) # завршетак рада са скупом података
```

Цртамо и тренутне и бруто приходе у боксплоту (Слика 4.8).

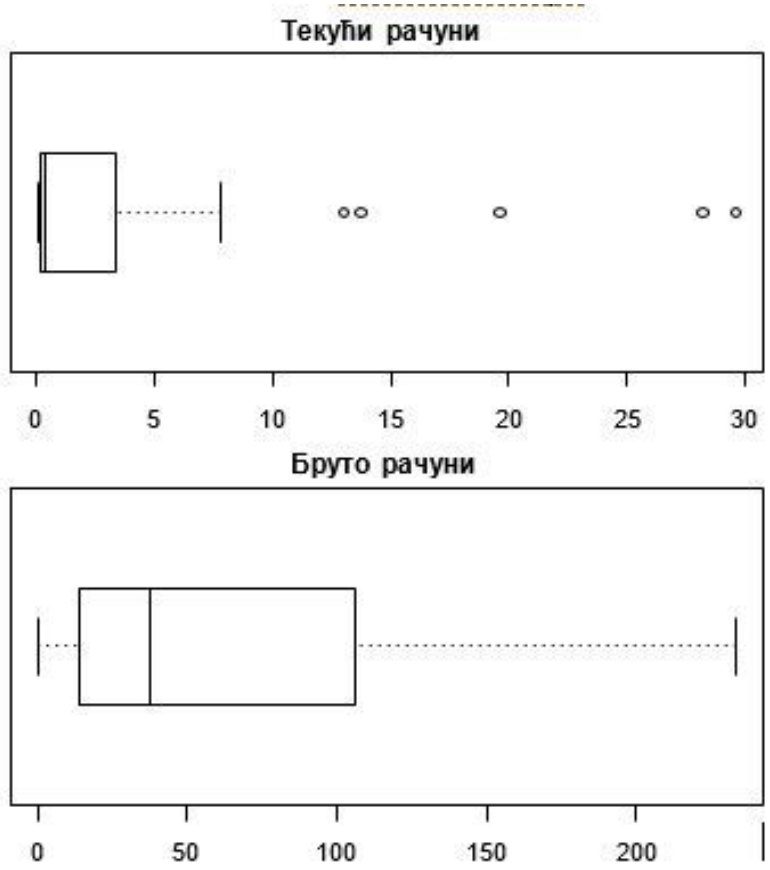
Напомена: Обе дистрибуције су искривљене, али бруто приходи значајно мање. Ово показује зашто је Холивуд толико заинтересован за „велики хит”. Прави велики хит може да генерише много више прихода него више просечних „хитова” укупно.

Основе R-а: Читање скупа података из пакета

У горенаведеном примеру чита се из уграђеног скупа података. Како се нпр. чита у скупу података из пакета `ts` (функције временских серија)? Прво се мора учитати пакет, а затим затражити да учита податке. Ево кода:

```
> library("ts") # учитати библиотеку
> data("lynx") # учитати податке
```

[4] Скупови података који прате скрипту доступни су на https://cran.r-project.org/web/packages/UsingR/UsingR.pdf и морају да буду инсталирани командом `install.packages('UsingR')`.



Слика 4.8: Текући и бруто приходи од филмова

```
> summary(lynx) # само оно што је lynx?
Min. 1st Qu. Median Mean 3rd Qu. Max.
39.0 348.3 771.0 1538.0 2567.0 6991.0
```

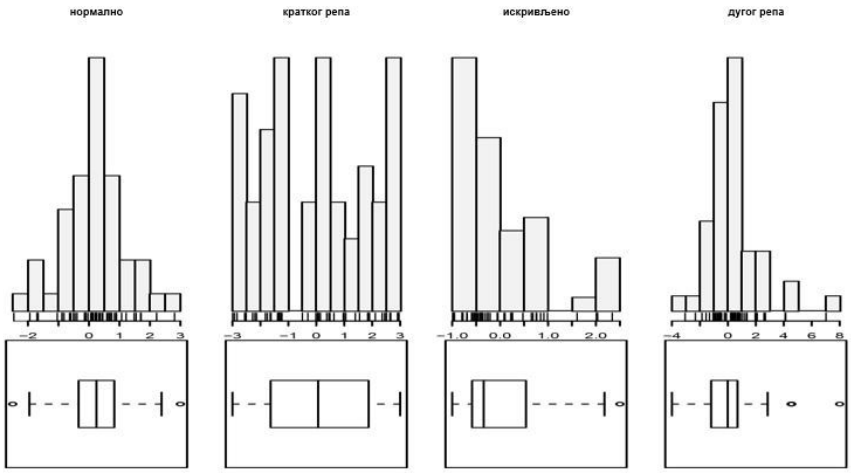
Команде `library` и `data` могу да се користе на неколико различитих начина. Да би се излистали сви доступни пакети користи се команда `library()`. За листање свих доступних скупова података користи се наредба `data()` без икаквих променљивих. За листање свих скупова података у датом пакету користи се `data(package='package name')`, на пример `data(package=ts)`. За читање података користи се `data('dataset name')`, као што је дато у горњем примеру `data(lynx)`. Најпре се учита пакет да би се добио приступ скуповима података, нпр. `library(ts)`.

Да би се пронашле информације о доступним скуповима података, користи се команда `help` да се види да ли постоји документација о датом скупу података. На пример `help(lynx)`, што је исто што и `?lynx`.

4.2.6 Истовремени приказ хистограма и боксплота

Функција `simple.hist.and.boxplot` исцртаће и хистограм и боксплот како би се приказао однос између два графика за исти скуп података. Слика 4.9 приказује неке при-

мере случајно генерисаних скупова података. Описи ових скупова података би могли бити (1) звоно (нормална расподела), (2) кратког репа, (3) искривљено и (4) дугог репа.



Слика 4.9: Случајно изабрана дистрибуција представљена и хистограмом и боксплотом

4.2.7 Полигони фреквенција

Понекад се информације представљене помоћу хистограма могу посматрати и на други начин. Уместо да се исцрта правоугаоник за сваки интервал, стави се тачка на врх правоугаоника, а затим се ове тачке повежу правим линијама. Ово се зове **фреквентни полигон**. Да би се генерисао, морају да се знају интервали и висине. Овде се показује начин како то урадити тако да `R` добије потребне вредности директно из наредбе `hist`. Претпоставимо да су просечни подаци успешности играча бејзбола за *New York Yankees*[5].

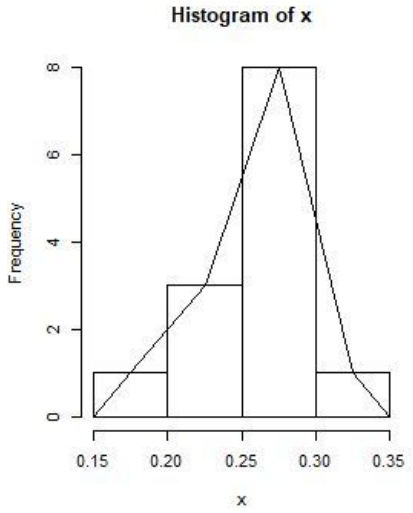
```
> x = c(.314,.289,.282,.279,.275,.267,.266,.265,.256,.250,.249,.211,.161)
> tmp=hist(x)  # чувају се резултати
> lines(c(min(tmp$breaks),tmp$mids,max(tmp$breaks)),
+ c(0,tmp$counts,0),type="l")
```

Као што се види, ово је релативно много куцања и зато постоји функција која то ради аутоматски, тј. `simple.freqpoly.R`. Такође, примећује се да тражени подаци могу да се добију и коришћењем функција `breaks` и `counts`.

4.2.8 Густина расподеле

Значај рада са полигоном фреквенције је да повеже хистограм са густином расподеле скупа података. За ову намену препоручљиво је користити уграђене функције. Уграђени скуп података `faithful` прати време између ерупција гејзира *Old Faithful*. Команда `density` може се користити за цртање софистицираних приказа са кривом, како то ради

[5] ови подаци су преузети са http://www.espn.com



Слика 4.10: Хистограм са полигоном фреквенција

полигон фреквенције. Функција `density()` имплицира аутоматску селекцију резолуције. Пун опис може се наћи на страницама за помоћ. Ако користимо подразумевани избор, једноставно је додати дијаграм густине расподеле хистограму. Позивамо функцију `lines` са резултатом густине расподеле, или `plot` ако је то први графикон. На пример:

```
> data(faithful)
> attach(faithful) # да се ерупције виде
> hist(eruptions,15,prob=T) # пропорције, а не фреквенције
> lines(density(eruptions)) # lines црта криву
> lines(density(eruptions,bw="SJ"),col='red')
# користи SJ кернел, у црвеној боји
```

Основна идеја је да се за сваку тачку узме нека врста просека за тачке у околини и да се на основу тога дају процене густине. Детаљи око тражења просека могу бити врло компликовани, али главна контрола је **ширина опсега** (*bandwidth*) и може се контролисати по жељи. За последњи график одабрана је ширина опсега `SJ`. Може се такође поставити и на фиксну вредност. На слици 4.11 су три примера са ширином пропусног опсега од 0.01, 1 и 0.1, респективно. Може се приметити да ако је ширина опсега премала, резултат је сувише оштар. Ако је пак превелика, дистрибуција се изгуби усредњавањем.

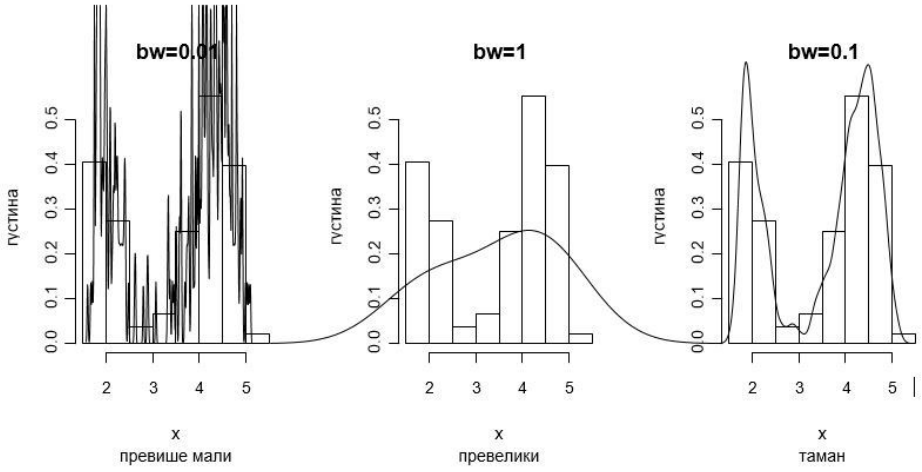
Задаци

1. Унети податке:

   60 85 72 59 37 75 93 7 98 63 41 90 5 17 97

   На основу њих креирати дијаграм стабла и листова.

2. Ишчитати график креиран у претходном задатку, а затим унети следеће податке и нацртати хистограм.



Слика 4.11: Хистограм и процене густине. Примећује се да је избор ширине опсега веома важан.

```
The decimal point is 1 digit(s) to the right of the |

8. | 028
9. | 115578
10. | 1669
11. | 01
```

3. Помоћу `R` команди могу се генерисати случајни подаци. На пример, изабере се 100 случајних бројева са нормалном расподелом. Треба направити два различита хистограма за два различита, случајно добијена скупа података. Да ли се добија исти хистограм?

4. Потребно је направити хистограм и боксплот од следећих скупова подаатака доступних у библиотеци `UsingR`: `south`, `crime` и `aid`. Који од ових скупова је искривљен? Који има нетипичне вредности, а који је симетричан?

5. Направити хистограм за `UsingR` скупове података `bumpers, firstchi, math`. Покушати предвиђање `mean`, медијане и стандардне девијације са слике. Проверити предвиђања прикладним `R` командама.

6. Број кварова О-прстена за прва 23 лета *Space Shuttle Atlantis* био је

   0 1 0 NA 0 0 0 0 0 1 1 1 0 0 3 0 0 0 0 0 2 0 1

   (NA значи недоступно - опрема је била изгубљена). Направити табелу могућих категорија. Покушати са `mean`. Потребно је изабрати `mean(x,na.rm=TRUE)` да би се избегла вредност `NA` или пробати са `x[!is.na(x)]`.

7. `UsingR` скуп података `pi2000` садржи првих 2000 цифара броја π. Направити хистограм. Није ли изненађујући? Затим пронаћи пропорцију за прву, другу и трећу цифру. Може ли се то урадити за свих 10 цифара, од 0-9?

8. Прилагодити процену густине на `UsingR` скуп података `pi2000`.

9. Пронаћи неки графикон на Интернету на сајту неке медијске куће. Помоћу `R`-а покушати добијање нечег сличног.

Глава 5

Биваријантни подаци

Предмет разматрања у статистици може бити евентуална веза између две величине. Рецимо, да ли су висина и ширина неког објекта повезани? Да ли су годиште неке особе и очитавање пулса некако повезани? Да ли су повезани приходи и плаћени порези? Да ли је нови лек бољи од старог? Да ли фудбалер прецизније шутира левом или десном ногом? Да ли временске прилике зависе од временских прилика претходних дана? Истраживање таквих повезаности је примарни циљ овог поглавља.

5.1 Категорички насупрот категоричких података

Команда `table` ће сумаризовати биваријантне податке, као што је то рађено са униваријантним подацима. Ево примера. Дата је студентска анкета са циљем да се испита да ли студенти пушачи заправо уче краће од студената непушача. Прикупљени подаци изгледају овако:

Особа	Пушач	Време учења
1	да	мање од 5 сати
2	не	5-10 сати
3	не	5-10 сати
4	да	више од 10 сати
5	не	више од 10 сати
6	да	мање од 5 сати
7	да	5-10 сати
8	да	мање од 5 сати
9	не	више од 5 сати
10	да	5-10 сати

У `R`-у се креирање овакве структуре постиже дефинисањем два вектора у које смештамо податке, и већ наученом командом `table`:

```
> smokes = c("Y","N","N","Y","N","Y","Y","Y","N","Y")
> amount = c(1,2,2,3,3,1,2,1,3,2)
> table(smokes,amount)
```

```
      amount
smokes 1 2 3
     N 0 2 2
     Y 3 2 1
```

Види се да можда постоји нека повезаност међу подацима. Сада је корисно извести и износе и пропорције, нпр. који проценат пушача учи 5 сати или мање. Познато је да је $\frac{3}{3+2+1} = \frac{1}{2}$, али како ово урадити у R-у? Команда `prop.table` ће ово решити аутоматски. Као аргументи се задају табела и број који показује да ли се желе редне пропорције $(a\,1)$ или пропорције по колонама $(a\,2)$. Ако се не зада други аргумент, пропорције се односе на све податке у табели.

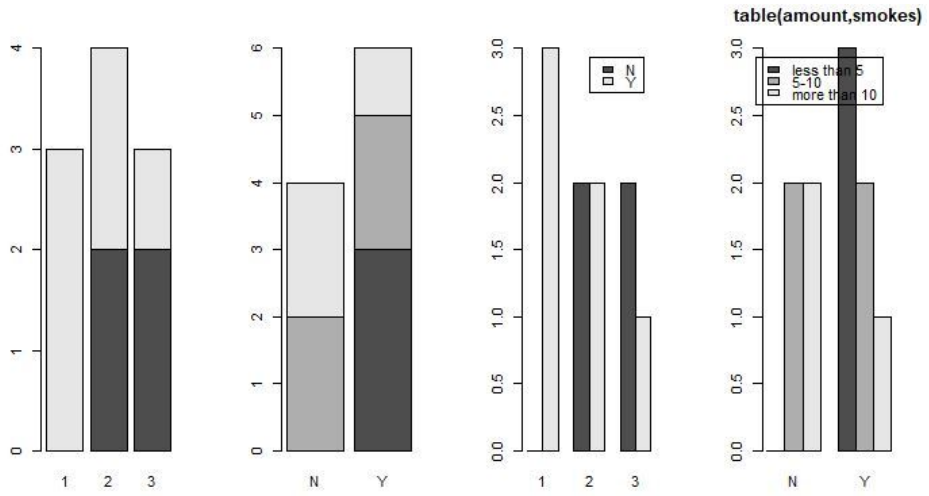
```
> tmp=table(smokes,amount) # сачувај табелу
> old.digits = options("digits") # сачувај број цифара за приказ
> options(digits=3) # штампај само 3 децимална места
> prop.table(tmp,1) # редови сада у суми дају 1
      amount
smokes     1     2     3
     N 0.000 0.500 0.500
     Y 0.500 0.333 0.167
> prop.table(tmp,2) # колоне у суми сада дају 1
      amount
smokes     1     2     3
     N 0.000 0.500 0.667
     Y 1.000 0.500 0.333
> prop.table(tmp) # сви бројеви у суми сада дају 1
      amount
smokes   1   2   3
     N 0.0 0.2 0.2
     Y 0.3 0.2 0.1
> option(digits=old.digits) # врати стари број цифара
```

5.1.1 Исцртавање табеларних података

Некада је потребно графички приказати податке сумиране у некој табели. За пример са пушењем, може се приказати вредност `amount` за свако `No` или `Yes`, или `No` и `Yes` за сваки ниво времена учења. У било ком случају, може се користити `barplot`, само је потребно позвати команду на одговарајући начин:

```
> barplot(table(smokes,amount))
> barplot(table(amount,smokes))
> smokes=factor(smokes) # за имена
> barplot(table(smokes,amount),
+ beside=TRUE,# ставити поред, ненаслагано
+ legend.text=T) # додати легенду
> barplot(table(amount,smokes),main="table(amount,smokes)",
```

```
+ beside=TRUE,
+ legend.text=c("less than 5","5-10","more than 10"))
```



Слика 5.1: Четири барплота за исте податке

На слици 5.1 се примећује важност редоследа при креирању табеле. Суштински, `barplot` исцртава сваки ред података. Може то урадити наслагано (подразумевана опција) или једно поред другог, постављањем `beside=TRUE`. Атрибут `legend.text` додаје легенду. Такође се могу изменити и називи променљивих, али најједноставнији начин је лабелирање редова из команде `table` и навођење `legend.text=T` у команди за цртање дијаграма.

5.1.2 Додатни увид: Условне пропорције

Уколико пожелимо да израчунамо однос броја пушача и непушача, потребно је поделити други и шести ред. Међутим, један или два реда је једноставно израчунати и ручно, али како се то може аутоматизовати? Наредба `apply` ће применити функцију на врсте или колоне у матрици. У овом случају, потребно је да дефинишемо функцију која израчунава пропорције вектора:

```
> prop = function(x) x/sum(x)
```

Применити ову функцију на матрицу x је сада једноставно. Прво се ураде колоне (индекс 2) са:

```
> apply(x,2,prop)
  amount
1 2 3
N 0 0.5 0.6666667
Y 1 0.5 0.3333333
```

Индекс 1 је за врсте, али мора се користити функција транспоновања `t()` да би резултат био валидан:

```
> t(apply(x,1,prop))
smokes 1 2 3
N 0.0 0.5000000 0.5000000
Y 0.5 0.3333333 0.1666667
```

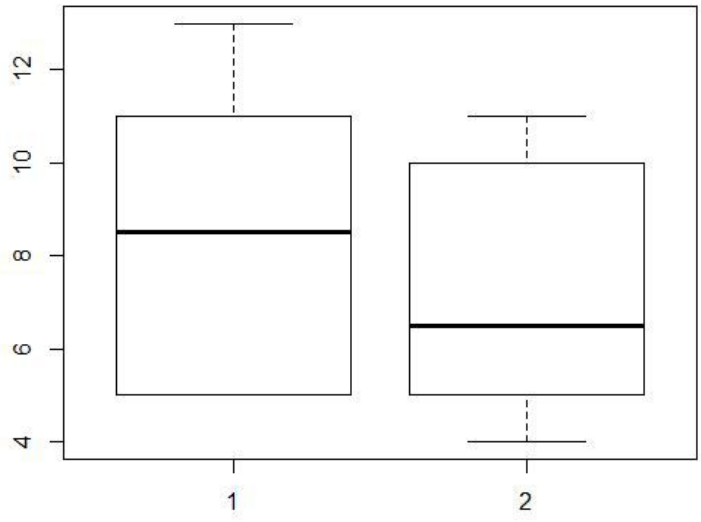
5.2 Категорички насупрот нумеричких података

Нека су дати нумерички подаци за неколико категорија. Једноставан пример може бити тест за неки лек, где су дати подаци (у одговарајућим јединицама) за експерименталну и контролну групу.

```
експериментална: 5 5 5 13 7 11 11 9 8 9
контролна: 11 8 4 5 9 5 10 5 4 10
```

Подаци се могу одвојено сумирати и поредити, али како се могу представити на истом графику? Боксплот представља солидан почетак. Једноставно га је генерисати:

```
> x = c(5, 5, 5, 13, 7, 11, 11, 9, 8, 9)
> y = c(11, 8, 4, 5, 9, 5, 10, 5, 4, 10)
> boxplot(x,y)
```



Слика 5.2: Боксплотови један поред другог

Из овог поређења се види (слика 5.2) да променљива y (контролна група означена са 2 на графику) изгледа нижа од променљиве x (експериментална група). Наравно, ови се подаци могу добити од експериментатора и у другачијем облику:

```
количина: 5 5 5 13 7 11 11 9 8 9 11 8 4 5 9 5 10 5 4 10
категорија: 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
```

Направити боксплотове један поред другог је и овде једноставно, ако се користи следећи синтаксни модел:

```
> amount=scan()
1: 5 5 5 13 7 11 11 9 8 9 11 8 4 5 9 5 10 5 4 10
21:
Read 20 items
> category=scan()
1: 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
21:
Read 20 items
> boxplot(amount~category) # примећује се ~
```

Израз $amount \sim category$ се посматра као разбијање вредности количине по категоријама и приказивање сваке од њих. Вербално, ово се може прочитати као *количина по категорији*. Више примера употребе ове специфичне синтаксе може се наћи у одељку о мултиваријантним подацима.

5.3 Нумерички насупрот нумеричких података

Упоређивање две нумеричке променљиве може се вршити на различите начине. Ако се сматра да су две варијабле независни узорци, могу се на неки начин упоређивати њихове расподеле. Међутим, ако се очекује веза између променљивих, боље је формирати парове тачака.

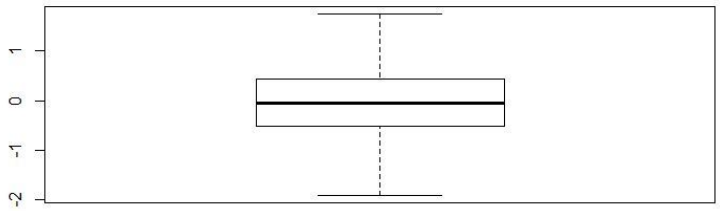
5.3.1 Упоређивање две расподеле помоћу плотова

Ако се упоређују две расподеле, то се може урадити боксплотовима типа *један до другог*. Међутим, ако се упоређују хистограми или неки други дијаграми да би се представило више података, постоји више различитих начина да се то уради.

Боксплотови са тепихом

Коришћењем команде `rug` при дну дијаграма постављамо линије за сваки податак појединачно. Ову опцију има смисла користити са малим скупом података, иначе морамо да позовемо и команду `jitter`, која уноси шум који умањује преклапање.

```
> library("UsingR");data(home) # учитавање скупа података
> attach(home)
> names(home)
[1] "old" "new"
> boxplot(scale(old),scale(new)) # исцртај боксплотове
> detach(home)
```



Слика 5.3: функција **scale**

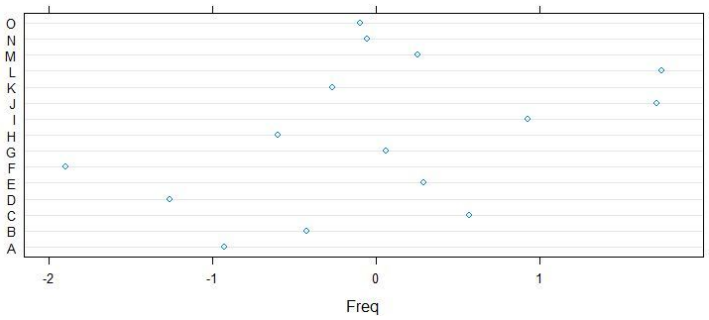
У овом примеру уводимо функцију `scale`. Она поставља два скупа података на исту скалу како би могли да се лакше упоређују.

Кад се направи овај боксплот (Слика 5.3), види се да две расподеле вероватно поседују међусобну корелацију. Проширен скуп података `homedata` ће ову процену само потврдити.

Коришћење тракастих или тачкастих графика

Тракасти (тачкасти) график ће исцртати све податке на начин који олакшава упоређивање расподела. За оквир података `hd` то се ради на следећи начин:

```
> stripchart(scale(old),scale(new))
```

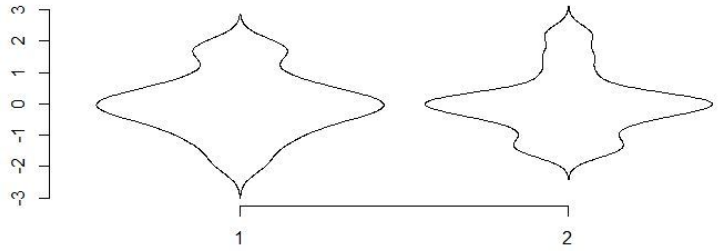


Слика 5.4: Тачкасти график за Homedata podatke

Упоређивање облика расподела

Коришћење функције густине `density` омогућава поређење облика расподела на истом дијаграму. Поређење расподела помоћу хистограма није баш згодно због преклапања. Међутим, функција `simple.violinplot` решава овај проблем коришћењем виолинских плотова. Они су слични боксплотовима, само што је уместо кутијом, густина расподеле представљена својом сликом у огледалу. Потребно је испробати ову команду да се види како то изгледа:

```
> simple.violinplot(scale(old),scale(new))
```



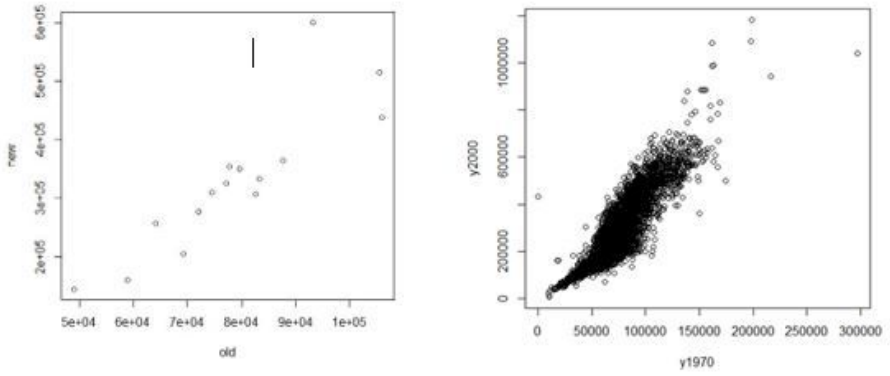
Слика 5.5: Виолински плот

Дијаграми расејања за упоређивање односа две варијабле

Често се намеће потреба да се истражи утицај једне променљиве на другу. На пример, висина оца у односу на висину његових синова или потрошња сладоледа у односу на потрошњу пива током године. Команда `plot` подразумевано приказује две променљиве на обичном дијаграму расејања.

Пример: Цене некретнина

Пример података о ценама некретнина из претходног одељка приказује стару процењену вредност (1970. година) у односу на нову процењену вредност (2000. година). Интуитивно, требало би да постоји нека веза између ове две колоне. Истраживање обављамо помоћу графика на слици 5.6.



Слика 5.6: Дијаграм расејања узорка (лево) и целог скупа података (десно) о ценама некретнина

```
> data(home);attach(home)
> plot(old,new)
> detach(home)
```

Други график је нацртан коришћењем целокупног скупа података. Он би требало да буде доступан као скуп података кроз команду `data()`. Олакшавамо себи посао користећи `attach`:

```
> data(homedata)
> attach(homedata)
> plot(old,new)
> detach(homedata)
```

Изгледа да графици илуструју **јак линеарни тренд**, који ће бити истражен нешто касније.

Основе R-а: Шта ради *attaching*?

Када се повезују `home` и `homedata`, може се приметити да се појављују иста имена променљивих: old и new. Шта у ствари ради *attaching*? Када тражимо од `R`-а да користи вредност променљиве или функције, `R` то чини кроз неколико тзв. *окружења*. Постављањем оквира података, имена променљивих се постављају у друго претраживано окружење (име оквира података је у првом окружењу). Оне су маскиране променљивама које већ имају исто име. Постоје последице на које треба обратити пажњу. Прво, може се појавити известан степен конфузије у вези са променљивом која се тренутно користи. Што је најважније, не могу се променити вредности променљивих у оквиру података без упућивања на тај оквир. На пример, креира се оквир података `df` са променљивама `x` и `y`:

```
> x=1:2;y=c(2,4);df=data.frame(x=x,y=y)
> ls() # листа свих познатих променљивих
[1] "df" "x" "y"
> rm(y) # брисање променљиве y
> attach(df) # повезујемо са оквиром
> ls()
[1] "df" "x" # y је видљиво али се не појављује
> ls(pos=2) # y је на позицији 2 од када је повезано
[1] "x" "y"
> y # y је видљиво јер је df повезано
[1] 2 4
> x # које x је нађено, x или df[['x']]
[1] 1 2
> x=c(1,3) # додељено x-y
> df # нема x-а y df
  x y
1 1 2
2 2 4
> detach(df)
> x # додељено правој променљивој x
[1] 1 3
> y
Error: Object "y" not found
```

Важно је запамтити да треба одвојити скуп података између употребе истоимених променљивих, јер се лако може заборавити на коју променљиву се заправо референцира.

У датим примерима су илустровани односи између различитих података. У оба случаја то је линеарна веза. Моделовање таквих односа је уобичајена статистичка пракса и омогућава нам да радимо предвиђања променљиве y на основу вредности променљиве x и обратно.

5.4 Линеарна регресија

Линеарна регресија је назив процедуре која одговара проналажењу најбоље праве која одговара унетим биваријантним подацима. Идеја је да вредност x експериментатор контролише, а вредност y експериментатор мери. Права добијена регресијом заправо може да предвиди вредност y за познату вредност x! Променљива x је независна променљива, а y зависна. Уколико напишемо једначину праве као:

$$\hat{y} = b_0 + b_1 x,$$

онда би за свако x_i предвиђена вредност била

$$\hat{y_i} = b_0 + b_1 x_i.$$

Међутим, измерена вредност је заправо y_i, а разлика се назива **остатком** и има вредност

$$e_i = y_i - \hat{y_i}.$$

Метод најмањих квадрата се користи да би се изабрале вредности b_0 и b_1 које минимизирају суму квадрата грешака остатака. Математички представљена вредност суме износи:

$$\sum_{i=1}^{n} (y_i - \hat{y_i})^2.$$

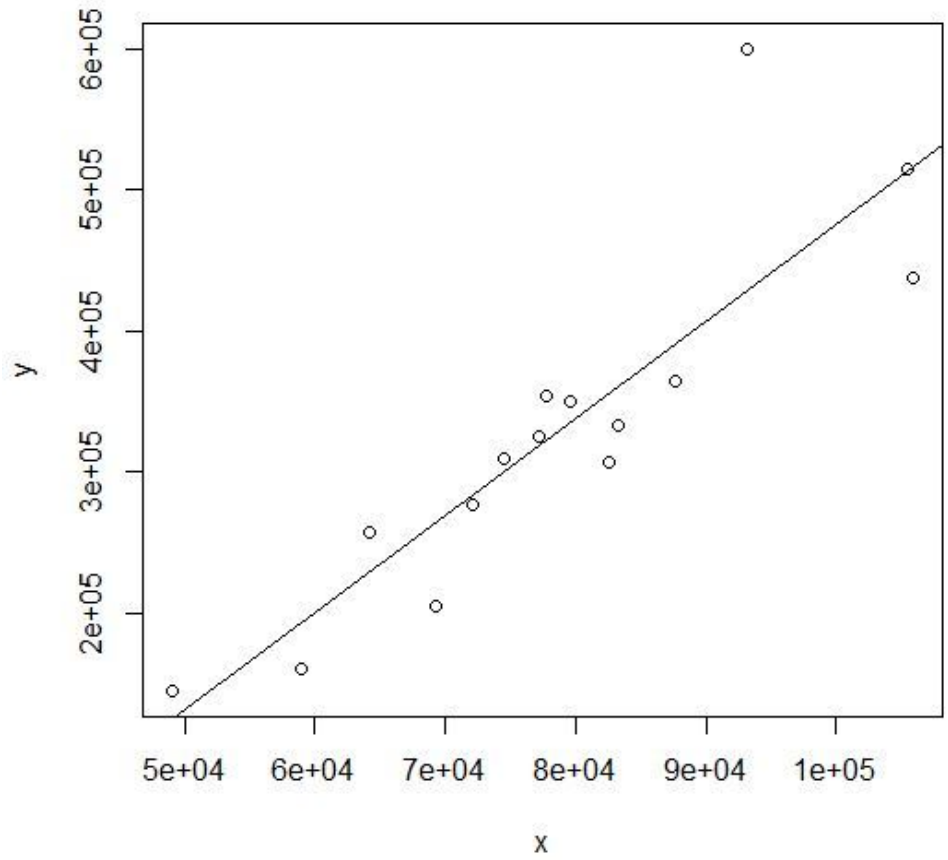
Ову суму треба минимизовати, а то се постиже изједначавањем извода горњег израза са нулом. Тиме се добија систем од две једначине са две непознате, b_0 и b_1. Решавањем система се добија:

$$b_1 = \frac{s_{xy}}{{s_x}^2} = \frac{\sum (x_i - \overline{x})(y_j - \overline{y})}{\sum (x_i - \overline{x})^2}, \qquad b_0 = \overline{y} - b_1 \overline{x}.$$

То је права са коефицијентом правца b_1, која пролази кроз тачку $(\overline{x}, \overline{y})$.

Помоћу R-а линеарна регресија се извршава у три корака: уцртати тачке, пронаћи вредности b_0, b_1, а затим додати линију дијаграму.

```
> data(home);attach(home)
> x=old # користе се генеричка имена променљивих
> y=new # само за илустрацију
> plot(x,y)
> abline(lm(y~x))
> detach(home)
```



Слика 5.7: Подаци о ценама некретнина са регресионом линијом

Функција `abline` штампа линију са задатим коефицијентима у тренутном прозору графика. Права коју штампа долази као резултат функције `lm`. Ово је функција за линеарни модел. Синтакса облика $y \sim x$ говори `R`-у да моделује променљиву y као линеарну функцију x. Ово је синтакса модела формуле `R`-а која понекад може бити збуњујућа, али је прилично јасна у овој ситуацији.

Као алтернатива горенаведеном, функција `simple.lm` из пакета `UsingR`, исцртаће исти график и вратити коефицијенте регресије:

```
> data(home);attach(home)
> x=old;y=new
> simple.lm(x,y)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
 -2.121e+05    6.879e+00

> detach(home)
```

Вредностима коефицијената се може приступити директно помоћу функције `coef`:

```
> lm.res=simple.lm(x,y) # сачувати одговоре у lm.res
> coef(lm.res)
  (Intercept)             x
-2.121158e+05  6.879161e+00
> coef(lm.res)
  (Intercept)             x
-2.121158e+05  6.879161e+00
```

5.4.1 График резидуала

Још један информативан график је график остатака (**резидуала**). Ово такође може бити урађено помоћу `simple.lm` функције. Наредба

```
simple.lm(x,y,show.residuals=TRUE)
```

производи график на Слици 5.8.

На слици 5.8 су приказана три нова графика. Нормални график ће бити објашњен касније. Горњи десни дијаграм је дијаграм резидуала насупрот одговарајућих вредности (y-оса). Ако подаци одговарају стандардном статистичком моделу, резидуали су расути око линије $y = 0$ са вредностима *нормалне* расподеле (у облику звона). За ове конкретне податке уочљиво је и једно искакање које заслужује додатну пажњу.

Како би се директно приступило резидуалима линеарне регресије, користи се команда `resid` на резултату команде `lm`:

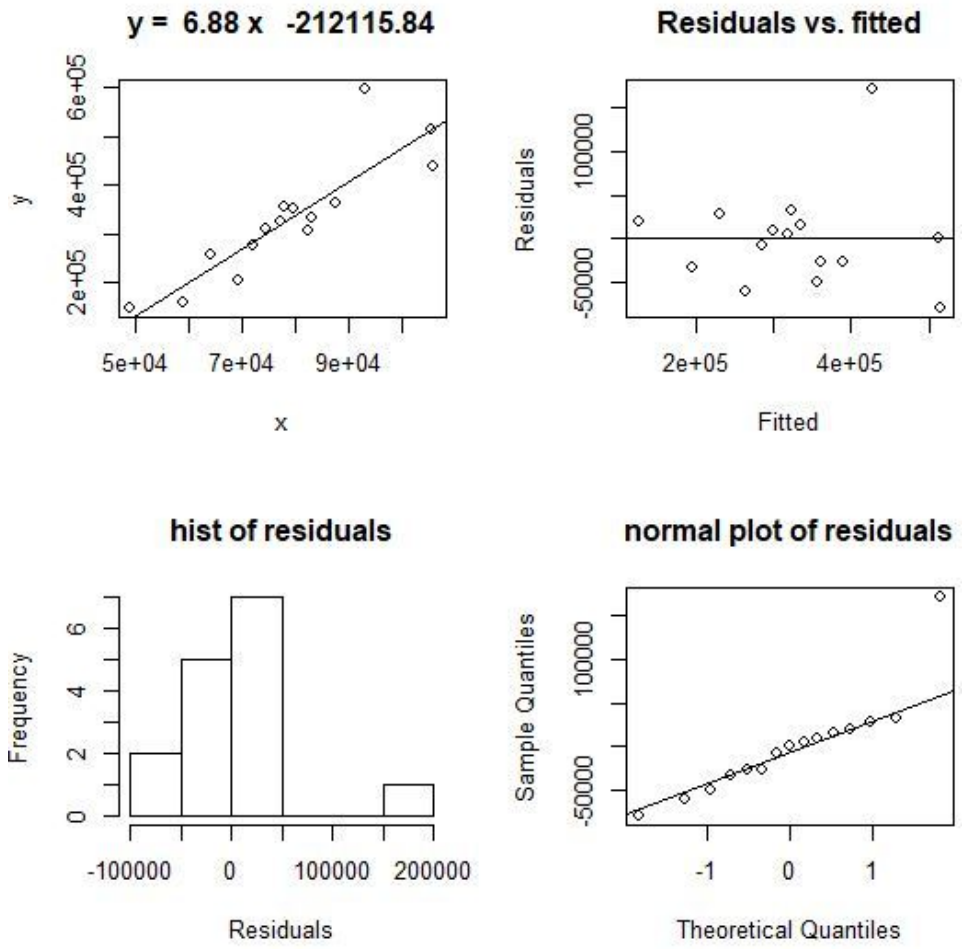
```
> lm.res=simple.lm(x,y)
> the.residuals=resid(lm.res) # како добити остатке
> plot(the.residuals)
```

5.4.2 Коефицијенти корелације

Нумеричка вредност јачине линеарне везе између две серије података је **Пирсонов коефицијент** R дефинисан као

$$R = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum (X_i - \overline{X})^2 \sum (Y_i - \overline{Y})^2}}$$

Коефицијент корелације је заправо скалирана верзија коваријансе између X и Y. То је мера како варирање једне променљиве утиче на другу. Вредност је скалирана да се уклопи у интервал $[-1, 1]$. Вредности R^2 блиске 1 указују на јаку линеарну везу, док вредности близу 0 на слабу, док вредноссти близу -1 такође указују на јаку линеарну везу, али обрнуте пропорционалости. Наравно, могуће је да још увек постоји веза, само нелинеарна. У `R`-у се коефицијент корелације налази позивом функције cor:



Слика 5.8: График резидуала за модел линеарне регресије

```
> cor(x,y) # да се добије R
[1] 0.8806976
> cor(x,y)^2 # да се добије R^2
[1] 0.7756283
```

Пирсонов коефицијент корелације се у `R`-у рачуна и када се извршава линеарна регресија, али се не штампа подразумевано. Може се добити командом `summary(lm (y~x))`. **Спирманова корелација** ранга се рачуна на исти начин као Пирсонова, само се примењује на рангове скупа података. Ранг скупа података је заправо вектор који даје релативни ранг вредности. Боље је објаснити примером:

```
> rank(c(2,3,5,7,11)) # већ су поређани
[1] 1 2 3 4 5
> rank(c(5,3,2,7,11)) # нпр. 5 је трећи одозго
[1] 3 2 1 4 5
> rank(c(5,5,2,7,5)) # када се подаци понављају
[1] 3 3 1 5 3
```

Да бисмо нашли Спирманов ранг корелације једноставно позивамо `cor()` на рангираним

подацима

```
> cor(rank(x),rank(y))
[1] 0.925
```

Овај број је близак 1 (или -1) ако постоји строго растући (опадајући) тренд у подацима. Тренд не мора бити линеаран. Као додатак, може се креирати и функција да изврши овај рачун, нпр:

```
> cor.sp <- function(x,y) cor(rank(x),rank(y))
> cor.sp(x,y)
[1] 0.925
```

Пример: Председнички избори у Флориди

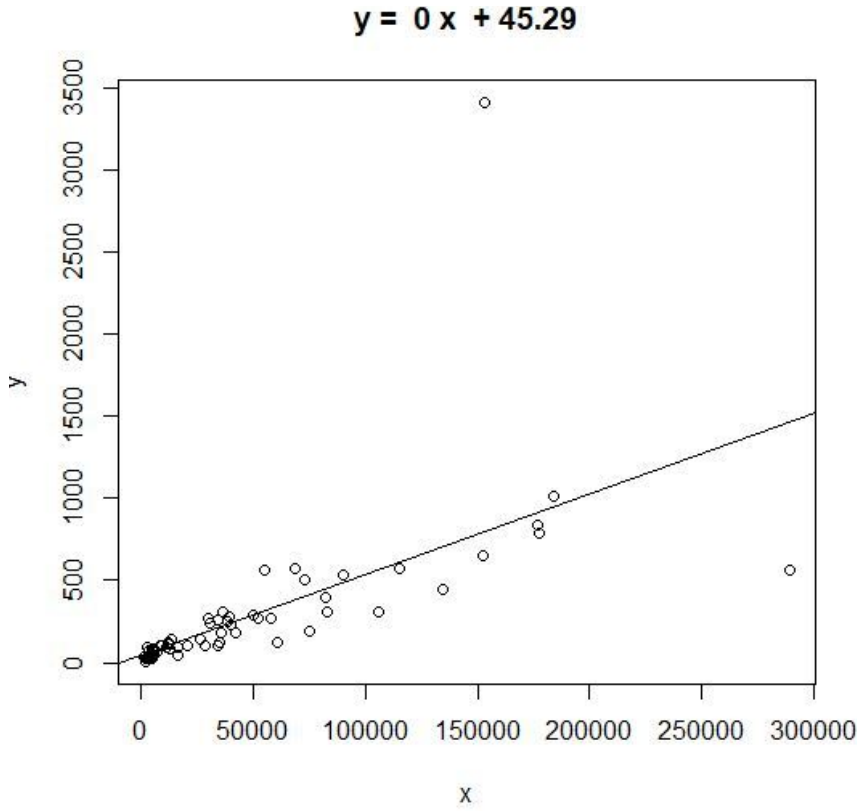
Разматра се скуп података са председничких избора 2000. године у држави Флорида[1]. Бележи број гласова за сваког кандидата које је добио у различитим окрузима државе Флориде. Истражује се однос између броја гласова за Буша насупрот броја гласова за Бјукенона.

```
> data(florida) # или read.table у florida.txt
> names(florida)
 [1] "County"     "GORE"       "BUSH"       "BUCHANAN"   "NADER"
 [6] "BROWN"      "HAGELIN"    "HARRIS"     "MCREYNOLDS" "MOOREHEAD"
[11] "PHILLIPS"   "Total"
> attach(florida) # да би се добила имена кандидата
> simple.lm(BUSH,BUCHANAN)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  45.289861     0.004917
> detach(florida) # рашчишћавање
```

На слици 5.9 се види јака линеарна веза, изузев две нетипичне тачке. Како се могу идентификовати ове тачке? Један од начина је да се претраже сами подаци. Ово функционише добро за мање скупове података, али за веће, `R` пружа неколико корисних функција: `identify` за проналажење индекса најближих (x, y) координата на клик миша и `locator`, како би се пронашле вредности (x, y) на које се кликнуло мишем.

Да би се идентификовале нетипичне вредности, потребни су њихови индекси које даје `identify`:

[1] Ови подаци су преузети из „John Maindonald, Using R for Data Analysis and Graphics”. Даља дискусија о овим подацима може се пронаћи на неколико веб страница.



Слика 5.9: График расејања Бјукененових гласова у односу на Бушове гласове

```
> identify(BUSH,BUCHANAN,n=2) # n=2 даје две тачке
[1] 13 50
```

Кликом на две нетипичне тачке налазе се одговарајући индекси, и то 13 и 50. Вредности се могу добити узимањем 13. и 50. вредности вектора:

```
> BUSH[50]
[1] 152846
> BUCHANAN[50]
[1] 3407
> florida[50,]
County V2 GORE BUSH BUCHANAN NADER BROWNE HAGELIN HARRIS MCREYNOLDS
50 50  39 268945 152846   3407 5564    743   143   45     302
MOOREHEAD PHILLIPS Total
50 103 188 432286
```

Последња линија показује цео ред за округ 50. Округ 50 је заправо округ Мајами-Дејд у коме је коришћени гласачки листић проузроковао велику конфузију међу гласачима. Тастер за Бјукенона на листићу је заправо био лоциран поред имена Ала Гора. Колико Горових гласова је ова конфузија додала Бјукенону? Један начин да се одговори је да се нађе регресиона линија за податке без ових тачака и онда употребе Бушови гласови да се предвиди реални број Бјукенонових гласова. Елиминисање једне тачке из вектора података може

се извести индексирањем, употребом минуса (BUSH[50] је 50. елемент, BUSH[-50] је све осим 50-ог елемента):

```
> simple.lm(BUSH[-50],BUCHANAN[-50])
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  65.573496     0.003482
```

Примећује се да је резултат значајно бољи. Нова линија регресије је

$$y = 65.57350 + 0.00348x$$

уместо $y = 45.28986 + 0.00492x$. Колико ово утиче на разлике? Регресиона права b предвиђа вредности за дато x. Ако је Буш добио 152846 гласова (`BUSH[50]`), онда се очекује да Бјукенон добије:

```
> 65.57350 + 0.00348 * BUSH[50]
[1] 597.4776
```

а не 3407 (`BUCHANAN[50]`) као што је стварно добио.

Додатни увид: употреба `simple.lm` за предвиђање

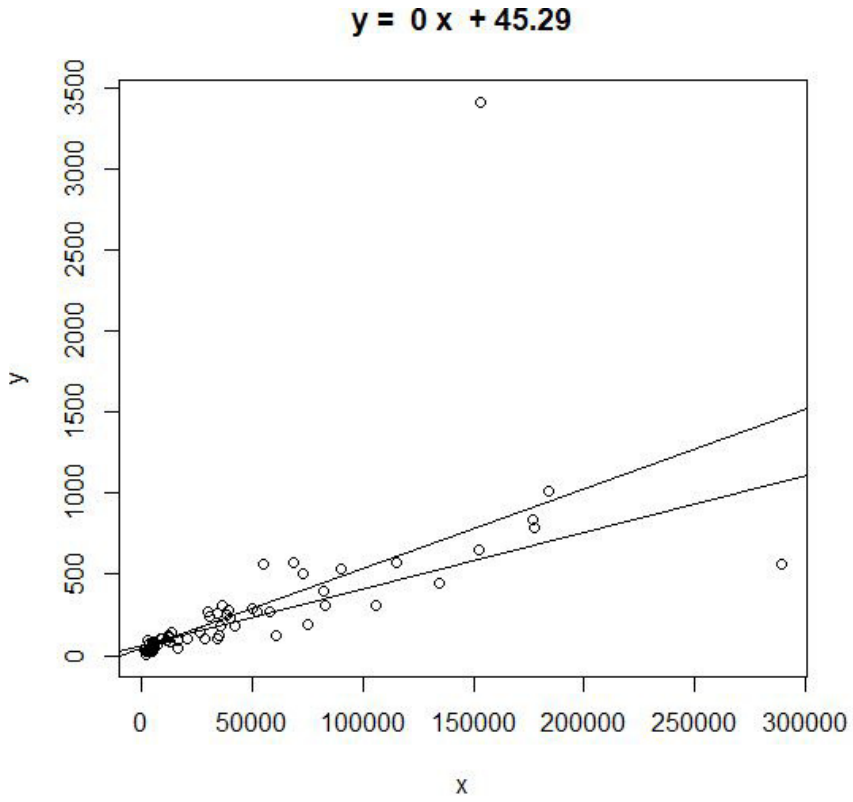
Овакво предвиђање се може извести и помоћу функције `simple.lm`. Ево како:

```
> simple.lm(BUSH[-50],BUCHANAN[-50],pred=BUSH[50])
       1
597.7677

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  65.573496     0.003482
```

5.4.3 Регресија отпорна на нетипичне вредности

Претходни пример илуструје још једну важну ставку. Као и средња вредност и стандардна девијација, линија регресије је веома осетљива на нетипичне вредности. Како линија регресије изгледа за податке са и без тих тачака? Пошто већ постоји једначина праве за цео скуп података, најједноставнији начин је да се прво нацрта линија за све податке, а затим дода линија без округа Мајами-Дејд. Ово се постиже функцијом `abline`. Слика 5.10 на добар начин илуструје осетљивост регресионе праве на *само једну* нетипичну вредност.

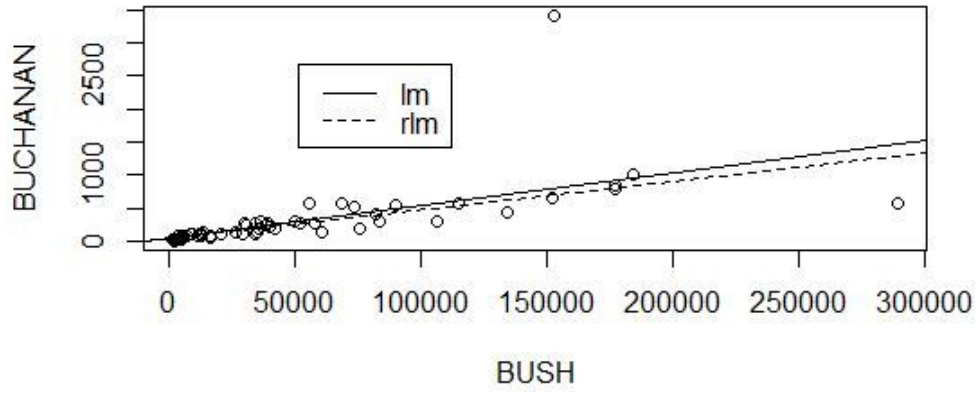
```
> simple.lm(BUSH,BUCHANAN)
Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
  45.289861     0.004917
> abline(65.57350,0.00348) # бројеви из претходног
```



Слика 5.10: Регресиона линија за податке са и без Мајами-Дејд нетипичне вредности

Отпорна регресија помоћу `rlm` и `lqs`

Отпорност у статистици значи да је поступак отпоран на неки проценат произвољно великих изузетака, а робусност значи да је поступак у великој мери прилагођен благим одступањима у претпоставкама. Постоје различити начини да се формира отпорна права регресије. У `R`-у постоје две функције у пакету `MASS` које се користе на сличан начин као и функција `lm` (али не и функција `simple.lm`). Функција `lqs` уместо минимизирања суме квадрата остатака за све резидуале, то чини само за одређени проценат резидуала. Функција *rlm* користи тзв. *M*-процењивач. Оба дају сличне резултате, али не и идентичне. У наредном тексту се користи `rlm`, али се би се могла користити и `lqs`, под условом да се прво учита библиотека (`library('lqs')`). `rlm` се примењује на податке о изборима на



Слика 5.11: График обичне и отпорне регресије

Флориди које смо већ користили. График обичне регресије и отпорне регресије дат је на слици 5.11.

```
> library(MASS) # учитати пакет
> attach(florida)
> plot(BUSH,BUCHANAN) # график расејања
> abline(lm(BUCHANAN ~ BUSH),lty=1) # lty поставља врсту линије
> abline(rlm(BUCHANAN ~ BUSH),lty=2)
> legend(locator(1),legend=c(lm,rlm),lty=1:2) # додавање легенде
> detach(florida) # поспремити
```

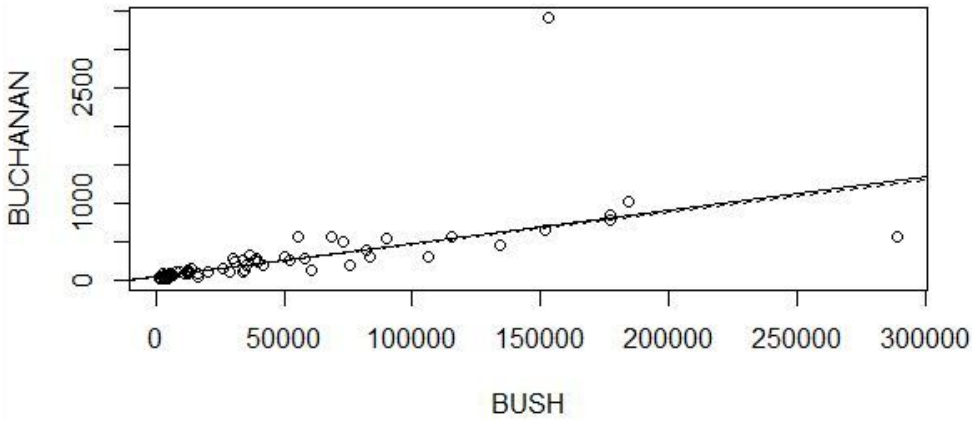
Може се приметити неколико ствари. Прво, коришћен је модел формулације $lm(y \sim x)$, што је исто што очекује и функција `rlm`. Такође се илуструје како се може променити врста линије (`lty`), као и како укључити легенду помоћу `legend`. Целокупан код дат је у наставку:

```
> plot(BUSH,BUCHANAN)
> abline(rlm(BUCHANAN ~ BUSH),lty=1)
> abline(rlm(BUCHANAN[-50] ~ BUSH[-50]),lty=2)
```

Овај график ће показати да уклањање једне тачке не чини разлику на правој која је отпорна (како се и очекује - Слика 5.12).

5.5 Основе R-а: Цртање графика

У овом одељку коришћена је команда `plot` за креирање графика расејања и наредба `abline` за додавање линије. Постоје и други начини да се манипулише плотовима које је корисно знати.



Слика 5.12: Отпорност графика на уклањање једне тачке

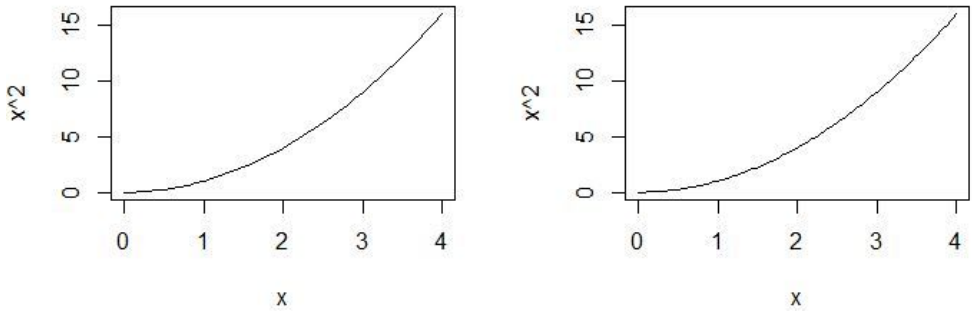
5.5.1 Цртање аналитичких функција помоћу `plot` и `curve`

Функција `plot` ће исцртати тачке као што је већ илустровано у више наврата. На пример, уколико се квадрирају равномерно распоређене тачке, добиће се парабола. Обратити пажњу да је обавезно креирати појединачне вредности на x оси дијаграма:

```
> x=seq(0,4,by=.1) # креирање вредности за x
> plot(x,x^2,type="l") # симбол "l" за линију
```

Међутим, функција `curve` је погоднија јер нема захтев да претходно зна тачке на апсциси (слика 5.13):

```
> curve(x^2,0,4)
```



Слика 5.13: Цртање графика помоћу *plot* и *curve*

Као што је већ речено, и `plot` и `curve` отварају нове графичке прозоре у подразумеваном окружењу. Ако користимо пакет *RStudio*, графици се цртају у наменском прозору и могуће их је увећати, снимити као слику, копирати на клипборд итд.

5.5.2 Додавање графику помоћу `points`, `abline`, `lines` и `curve`

Постојећем графику може се додати произвољан број функција. За додавање тачака користи се команда `points` која је слична команди `plot`. Раније је за додавање праве задарте коефицијентима показана функција `abline`. Функција `lines` се користи за додавање општих линија и функционише слично као додавање `type="l"` функцији `plot`. Коначно, функција `curve` ће активном графику додати линију ако је задато `add=TRUE`. Као илустрација, ако је дат скуп података:

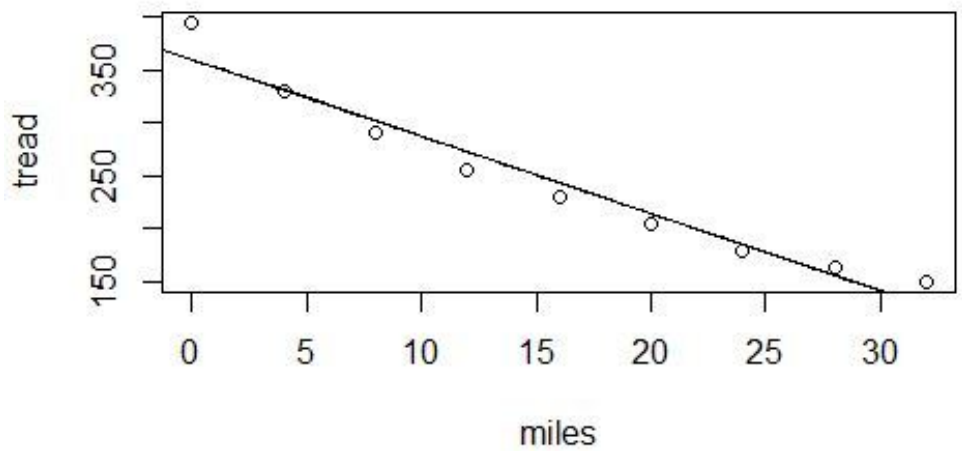
mileage	0	4	8	12	16	20	24	28	32
tread	394	329	291	255	229	204	179	163	150
wear									

Тада линија регресије има пресек са *y* осом 360 и коефицијент правца -7.3. Следе три начина за цртање података и регресионе линије (слика 5.14).

```
> miles=(0:8)*4 # 0 4 8 ... 32
> tread=scan()
1: 394 329 291 255 229 204 179 163 150
10:
Read 9 items
> plot(miles,tread) # график расејања
abline(lm(tread~miles))
## или ако се зна пресек са y осом и нагиб
> abline(360,-7.3)
## или ако се користи points
> points(miles,360-7.3*miles,type="l")
## или ако се користи lines
> lines(miles,360-7.3*miles)
## или ако се користи curve
> curve(360-7.3*x,add=T) # додаје се функција од x
```

Задаци

1. Студентска оцена наставника дата је на *Leichert*-овој скали оценама од 1 до 5. Одговори на прва три питања су дати у следећој табели:



Слика 5.14: Додавање графику помоћу *points, abline, lines* и *curve*

Студент	Питање 1	Питање 2	Питање 3
1	3	5	1
2	3	2	3
3	3	5	1
4	4	5	1
5	3	2	1
6	4	2	3
7	3	5	1
8	4	5	1
9	3	4	1
10	4	2	1

Подаци се учитавају за Питање 1 и Питање 2 коришћењем `c()`, `scan`, `read.table` или `data.entry`

- Формирати табелу од резултата за питање 1 и питање 2 одвојено.
- Формирати табелу случаја за питања 1 и 2.
- Формирати сложену табелу за питања 2 и 3.
- Нацртати упоредни график за сва три питања.

2. У библиотеци `MASS` је скуп података `UScereal` који садржи информације о популарним цереалијама. Потребно је закачити скуп података на следећи начин:

```
> library('MASS')
> data('UScereal')
> attach(UScereal)
> names(UScereal) # да се виде имена
```

Истражити односе и коментарисати виђено. Могу се користити табеле, плотови и графици расејања:

- однос између произвођача и полица,
- однос између масти и витамина,
- однос између масти и полица,
- однос између угљених хидрата и шећера,
- однос између влакана и произвођача,
- однос између натријума и шећера.

Постоје ли други односи које се могу предвидети и истражити?

3. Уграђени скуп података `mammals` садржи податке о телесној тежини у односу на тежину мозга сисара. Користити `cor` да би се пронашли Пирсонови и Спирманови коефицијенти корелације. Да ли су слични? Исцртати податке користећи команду `plot` и дискутовати евентуалну сличност. Овај дијаграм неће бити задовољавајући. Логаритмовати податке (`log`) и дискутовати да ли је дошло до побољшања у информативности дијаграма.

4. За скуп података који се односи на цене некретнина, `homedata`, истражити однос између старе и нове процењене вредности. Користити `old` као предикторску променљиву. Да ли подаци указују на линеарну везу? Постоје ли нетипичне вредности? Шта узрокује ове нетипичне вредности? Која је предвиђена нова процењена вредност за кућу од 75000 долара у 1970. години?

5. За скуп података `florida` Буша против Бјукенена, постоји још једна нетипична вредност која назначава да је Бјукенон добио мање гласова него што се очекивало. Ако уклонимо обе нетипичне вредности, која је предвиђена вредност за број гласова који би Бјукенен добио у округу Мајами-Дејд на основу броја гласова Буша?

6. За скуп података `emissions` исцртати БДП (бруто домаћи производ) по глави становника као предиктор за емисију CO_2. Идентификовати нетипичну вредност и пронаћи регресионе линије са овом тачком, као и без ове тачке.

7. Прикачити скуп података `babies`:

```
> library("Simple")
> data("babies")
> attach(babies)
```

Овај скуп података садржи пуно информација о бебама и њиховим мајкама за 1236 посматрања. Наћи коефицијент корелације (и Пирсонов и Спирманов) између узраста и тежине. Поновити исто за однос висине и тежине. Исцрати графике расејања сваког пара и проверити да ли процена има смисла.

8. Пронаћи скуп података који је кандидат за линеарну регресију. Потребне су нам две нумеричке варијабле, један предиктор и један одговор. Исцртати график расејања са регресионом линијом.

9. Уграђени скуп података `mtcars` садржи информације о аутомобилима из издања Мотор Тренд из 1974. године. Учитати скуп података (`data(mtcars)`) и одговорити на следеће:

   - Која су имена променљивих? Испробати команду `names`.
   - Који је максимални mpg?
   - Којих 5 аутомобила су први на листи?
   - Коју снагу (hp) има *Valiant*?
   - Које су све карактеристике Mercedesa 450 SLC (`Merc 450SLC`)?
   - Исцртати график расејања броја цилиндара (`cyl`) у односу на миље по галону (`mpg`, потрошња). Поставити регресиону линију. Да ли је ово добар кандидат за линеарну регресију?

10. Пронаћи неки график биваријантних података на Интернету. Искористити `R` и генерисати сличан дијаграм.

Глава 6

Мултиваријантни подаци

Да бисмо се упознали са приказивањем и манипулисањем подацима, пре свега је потребно организовати их. R користи тзв. оквире података, чиме помаже у организацији великих скупова података.

6.1 Мултиваријантни податаци у оквирима

Често су у статистици подаци представљени у табеларном формату. Колоне представљају различите атрибуте, а сваки ред представља по један узорак. На пример, скуп података home садржи две колоне, процењену вредност некретнине из 1970. године и процењену вредност из 2000. годдине. R користи оквире података да сачува ове вредности на окупу и поседује пуно олакшица за коришћење овако сачуваних података. Ако користимо R-ове уграђене скупове података, велике су шансе да су подаци већ доступни у облику оквира. Више о увожењу спољних података у R може се наћи у и документу *R Data Import/Export* који долази уз R.

Могуће је направити и сопствене оквире података и вероватно ће то и бити потребно. Заправо, довољно је да се подаци уклапају у правоугаони низ, као што је већ поменуто. Ако је тако, data.frame команда ће урадити посао. Као пример, чеетири особе су добиле по три питања: њихова тежина, висина и пол. Подаци су унети у R као посебне променљиве као што следи:

```
> weight = c(150, 135, 210, 140)
> height = c(65, 61, 70, 65)
> gender = c("Fe","Fe","M","Fe")
> study = data.frame(weight,height,gender) # направи оквир података
> study
    weight height gender
1    150     65     Fe
2    135     61     Fe
3    210     70      M
4    140     65     Fe
```

Колоне наслеђују имена променљивих. Могуће је дати и различита имена, по жељи:

```
> study = data.frame(w=weight,h=height,g=gender)
```

Називи се такође могу доделити и редовима. Ако су дати субјекти *Mary, Alice, Bob* и *Judy*, онда ће команда `row.names` излистати или поставити називе редова:

```
> row.names(study)<-c("Mary","Alice","Bob","Judy")
```

Команда `names` ће вратити имена колона, а може се користити и да их измени.

6.2 Приступање подацима у оквирима

Оквир `study` има три промењиве. Као и раније, могуће им је појединачно приступити након повезивања оквира података у `R` сесији користећи команду `attach`:

```
> weight = c(150, 135, 210, 140)
> height = c(65, 61, 70, 65)
> gender = c("Fe","Fe","M","Fe")
> study = data.frame(weight,height,gender)
> study
  weight height gender
1    150     65     Fe
2    135     61     Fe
3    210     70      M
4    140     65     Fe
> rm(weight) # почисти стару копију
> weight
Error: object 'weight' not found
> attach(study)
The following objects are masked _by_ .GlobalEnv:

    gender, height

> weight
[1] 150 135 210 140
```

Међутим, повезивање и одвајање оквира података може бити напоран посао ако се подацима приступа само једном. Поред тога, ако је оквир података повезан, није могуће истовремено мењати оригинални оквир података. Да би се приступило подацима корисно је знати да се оквири података посматрају као листе или низови и тако им се и приступа.

6.2.1 Приступ преко врсте и колоне

Матрица је начин за похрањивање података тако да им се може приступити преко индекса реда и колоне. Оквири података су заправо матрице, обзиром да имају колоне које су атрибути и редове који представљају појединачне узорке. Према томе, може се

приступити појединачним подацима специфицирајући ред и колону у угластим заградама (`[row,column]`). У општем случају, морају да постоје и ред и колона којима се може приступити. Ако је једно изостављено, добијемо цео ред или колону. Као пример, овако се добија променљива `weight`:

```
> study[,'weight'] # сви редови само weight колона
[1] 150 135 210 140
> study[,1] # сви редови само прва колона
[1] 150 135 210 140
```

Низовни приступ дозвољава и више флексибилности. Могу се добити `weight` и `height` навођењем прве и друге колоне одједном:

```
study[,1:2]
      weight height
Mary     150     65
Alice    135     61
Bob      210     70
Judy     140     65
```

Или, могу се добити све информације о *Mary* гледајући само њен ред или само њен атрибут `weight`, по жељи:

```
study['Mary',]
     weight height gender
Mary    150     65     Fe
study['Mary','weight']
[1] 150
```

6.2.2 Приступ као листи

Листа је општији концепт чувања података од оквира података. Листа је скуп објеката, при чему сваки може бити било ког типа. Оквир података је листа, где су објекти колоне које се чувају као вектори.

Да би се приступило листи, потребно је користити знак $ или дупле угласте заграде и број или име. Тако за *study* променљиву можемо приступити `weight` (прва колона) као листи на све наведене начине:

```
> study$weight # користећи $
[1] 150 135 210 140
> study[['weight']] # користећи име.
> study[['w']] # недвосмислене пречице су ок
> study[[1]] # према позицији
```

Ова два начина могу бити искомбинована као у овом примеру, рецимо да се извуку само информације о особама женског пола. Ово су редови где је пол `Fe`:

```
study[study$gender=='Fe',]
      weight height gender
Mary     150     65     Fe
Alice    135     61     Fe
Judy     140     65     Fe
```

6.3 Манипулисање оквирима података: `stack` и `unstack`

У појединим случајевима, могу бити корисна два начина за чување података. На пример, скуп података `PlantGrowth` изгледа овако:

```
> data (PlantGrowth)
> PlantGrowth
   weight group
1    4.17  ctrl
2    5.58  ctrl
3    5.18  ctrl
4    6.11  ctrl
5    4.50  ctrl
6    4.61  ctrl
7    5.17  ctrl
8    4.53  ctrl
9    5.33  ctrl
10   5.14  ctrl
11   4.81  trt1
12   4.17  trt1
13   4.41  trt1
14   3.59  trt1
15   5.87  trt1
16   3.83  trt1
17   6.03  trt1
18   4.89  trt1
19   4.32  trt1
20   4.69  trt1
21   6.31  trt2
22   5.12  trt2
23   5.54  trt2
24   5.50  trt2
25   5.37  trt2
26   5.29  trt2
27   4.92  trt2
28   6.15  trt2
29   5.80  trt2
30   5.26  trt2
```

Дакле, постоје три групе: контролна и два третмана. За сваку групу је посебно дата те-

жина. Заправо, подаци су генерисани бележењем тежине и групе за сваку индивидуалну биљку. Шта се дешава ако је потребно одштампати податке одвојено по групама? Методом грубе силе може се спровести раздвајање за сваку вредност `group`. Ово брзо постаје заморно за веће скупове података. Функција `unstack` ће то одрадити у једној линији. Ако су подаци структурирани исправно, креираће оквир података са променљивама које одговарају нивоима фактора:

```
> attach(PlantGrowth)
> weight.ctrl=weight[group=="ctrl"]
> unstack(PlantGrowth)
   ctrl trt1 trt2
1  4.17 4.81 6.31
2  5.58 4.17 5.12
3  5.18 4.41 5.54
4  6.11 3.59 5.50
5  4.50 5.87 5.37
6  4.61 3.83 5.29
7  5.17 6.03 4.92
8  4.53 4.89 6.15
9  5.33 4.32 5.80
10 5.14 4.69 5.26
```

Даље, да се би се креирао боксплот ове три групе могла би се користити команда у једној линији:

```
> boxplot(unstack(PlantGrowth))
```

6.4 Употреба R-ове модел формула нотације

Модел формула нотација којуR користи омогућава да се претходно раздвајање по групама уради на систематичан начин. На почетку је мало збуњујуће, али већина R-ових напредних функција користи ову флексибилну нотацију. Илуструјући пример изнад може се спровести и овако, ако је оквир података `PlantGrowth` већ закачен:

```
> boxplot(weight ~ group)
```

Шта ова команда ради? Разбија променљиву `weight` на вредности по фактору `group` и прослеђује резултат команди `boxplot`. Ред `weight ~ group` треба читати као: **врати `weight` по променљивој `group`**.

Када постоје две променљиве, запис је прилично интуитиван. Класификациона променљива је лево, док је предиктор на десној страни.

```
response ~ predictor # (за две променљиве)
```

Када има више од две *predictor* променљиве, ствари постају мало збуњујуће, нарочито зато што уобичајени математички оператори не раде оно што се мисли. Дато је неколико

различитих могућности које ће бити довољне за сада[1]. На пример, претпоставка је да су променљиве назване Y, X_1 и X_2:

Формула	**Значење**
$Y \sim X_1$	Y је моделовано са X_1
$Y \sim X_1 + X_2$	Y је моделовано са X_1 и X_2 као у вишеструкој регресији
$Y \sim X_1 * X_2$	Y је моделовано са X_1, X_2 и $X_1 * X_2$
$Y \sim (X_1 + X_2)^2$	Двојаке интеракције. Приметити обичан степен
$Y \sim X_1 + I(X_2^2)$	Y је моделовано са X_1 и X_2^2
$Y \sim X_1 \vert X_2$	Y је моделовано са X_1 и условно са X_2

Тачна интерпретација *моделовано по* варира у зависности од употребе. За `boxplot` команду другачије је него за команду `lm`. Такође треба приметити да су уобичајена математичка значења доступна, али морају бити укључена помоћу функције `I()`.

6.5 Начини приказивања мултиваријантних података

Када се научи како сачувати и приступити мултиваријантним подацима, време је да се прегледа велики број начина за визуелизацију оваквих скупова података.

6.5.1 *n*-тоструке табеле поређења

Двоструке табеле поређења се формирају командом `table`, као и оне вишег реда. Ако су w, x, y, z четири променљиве, тада команда `table(x,y)` креира двоструку табелу, `table(x,y,z)` креира двоструке табеле x наспрам y за сваку вредност z. Коначно, навођење x, y, z, w ће учинити исто за сваку комбинацију вредности z и w. Ако су променљиве сачуване у оквиру података, нпр. `df`, онда се команда `table(df)` понаша тако да свака променљива одговара колони у датом редоследу. Ради илустрације, размотиримо повезаности у скупу података `Cars93` из `MASS` библиотеке:

```
> library(MASS);data(Cars93);attach(Cars93)
## направити категоричке променљиве користећи cut
> price = cut(Price,c(0,12,20,max(Price)))
> levels(price)=c("cheap","okay","expensive"))
> mpg = cut(MPG.highway,c(0,20,30,max(MPG.highway)))
> levels(mpg) = c("gas guzzler","okay","miser"))
## погледати повезаности
> table(Type)
Type
Compact   Large Midsize   Small  Sporty     Van
     16      11      22      21      14       9
```

[1] Детаљно објашњење синтаксе и коришћења може се наћи у упутству *An Introduction to R*, и пратећем документу *Using R for Data Analysis and Graphics* од *Maindonald*-а. Такође, погледати команде `xtabs` и `ftable` уколико се укаже потреба за софистицираније употребом.

```
> table(price,Type)
           Type
price       Compact Large Midsize Small Sporty Van
  cheap           3     0       0    18      1   0
  okay            9     3       8     3      9   8
  expensive       4     8      14     0      4   1
> table(price,Type,mpg)
, , mpg = gas guzzler

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           0     0       0     0      0   0
  okay            0     0       0     0      0   2
  expensive       0     0       0     0      0   0

, , mpg = okay

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           1     0       0     4      0   0
  okay            5     3       6     0      6   6
  expensive       4     8      14     0      4   1

, , mpg = miser

           Type
price       Compact Large Midsize Small Sporty Van
  cheap           2     0       0    14      1   0
  okay            4     0       2     3      3   0
  expensive       0     0       0     0      0   0
```

6.5.2 Графици колоне

Као што је познато, графици колоне раде на сумарним подацима. Претходно је потребно пропустити податке кроз команду `table`. Команда `barplot` исцртава сваку колону као променљиву, баш као оквир података. Излаз команде `table` када је позвана са две променљиве користи прву променљиву за редове. Као и раније, графици колона су подразумевано наслагани. Можемо користити аргумент `beside=T` да би се графици исцртали један поред другог:

```
> barplot(table(price,Type),beside=T) # цена по различитим типовима
> barplot(table(Type,price),beside=T) # тип по различитим ценама
```

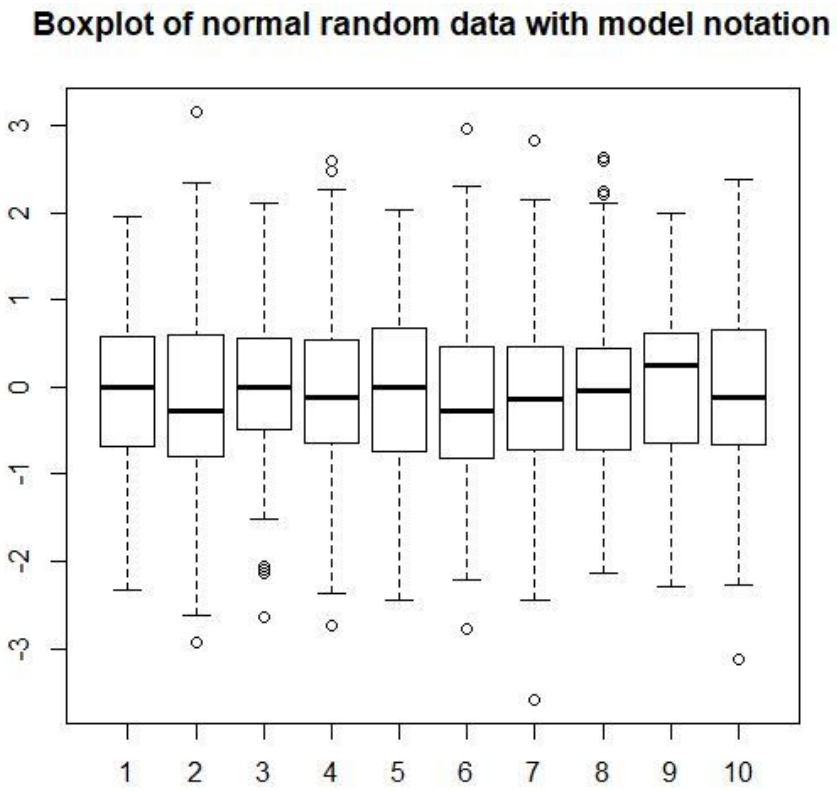
6.5.3 Боксплотови

Команда `boxplot` се једноставно користи за све типове чувања података. Наредба `boxplot(x,y,z)` ће произвести боксплотове један поред другог, као што је виђено раније. Према томе, једноставни примери `boxplot(df)` и `boxplot(y ~ x)` ће такође функционисати. Други облик користи модел формула нотацију.

Пример: Боксплот узорака насумичних података

Ево примера у ком се штампа 10 боксплотова података нормалне расподеле са средњом вредношћу 0 и стандардним одступањем 1. Овде се користи функција `rnorm` да произведе насумичне податке:

```
> y=rnorm(1000) # 1000 насумичних бројева
> f=factor(rep(1:10,100)) # бројеви 1,2,...10 100 пута
> boxplot(y ~ f,main="Boxplot of normal random data with model notation")
```



Слика 6.1: Бокс график направљен са `boxplot(y ~ f)`

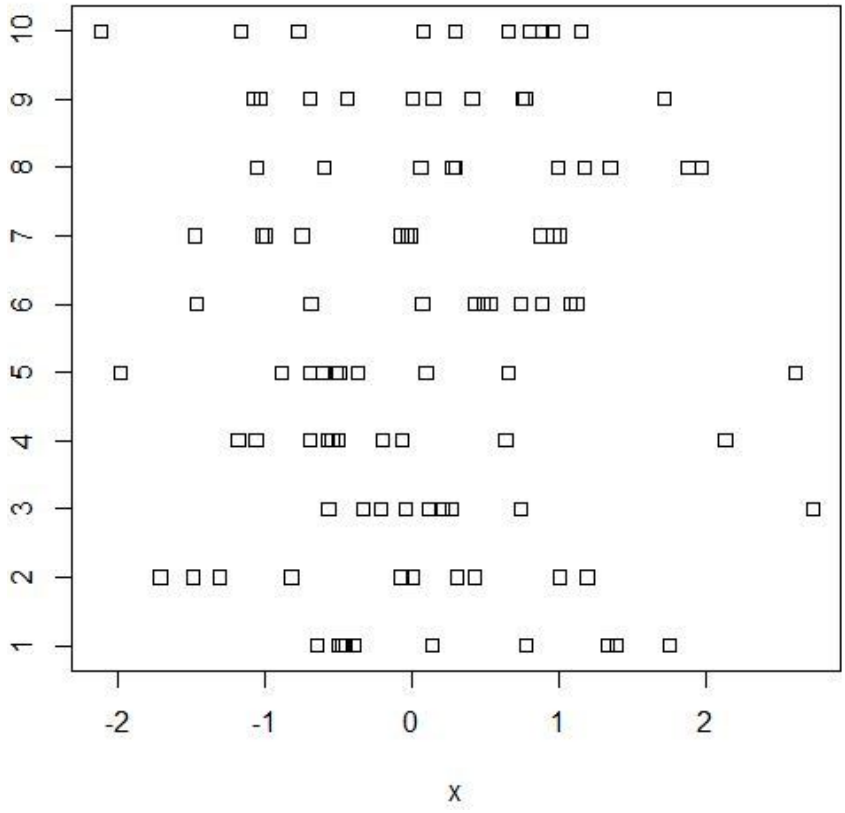
Приметити конструкцију f. Бројеви од 1 до 10 су поновљени по 100 пута да формирају `factor` вариајаблу исте дужине као y. Када се користи модел нотација, боксплот y података се ради за сваки ниво фактора f, тј. за сваку вредност y када је f једнако 1, затим 2, па до 10. Резултат је приказан на слици 6.1.

6.5.4 Тракасти дијаграми

Боксплотови исцртани један поред другог су корисни за поређење сличних расподела, нарочито ако има много података за сваку променљиву. Команда `stripchart` ради сличну ствар, а корисна је ако нема пуно података. Исцртава појединачне податке слично као команда rug која се користи са хистограмима. И `stripchart(df)` и `stripchart(x ~y)` ће функционисати, али не и `stripchart(x,y,z)`.

Као и у горњем примеру, потребно је генерисати 10 скупова насумичних бројева који подлежу нормалној расподели. Овог пута ће сваки скуп садржати само 10 насумичних бројева. Погледати слику 6.2:

```
> x = rnorm(100)
> y = factor(rep(1:10,10))
> stripchart(x ~ y)
```



Слика 6.2: Тракасти дијаграм

6.5.5 Виолински дијаграми и дијаграми густине

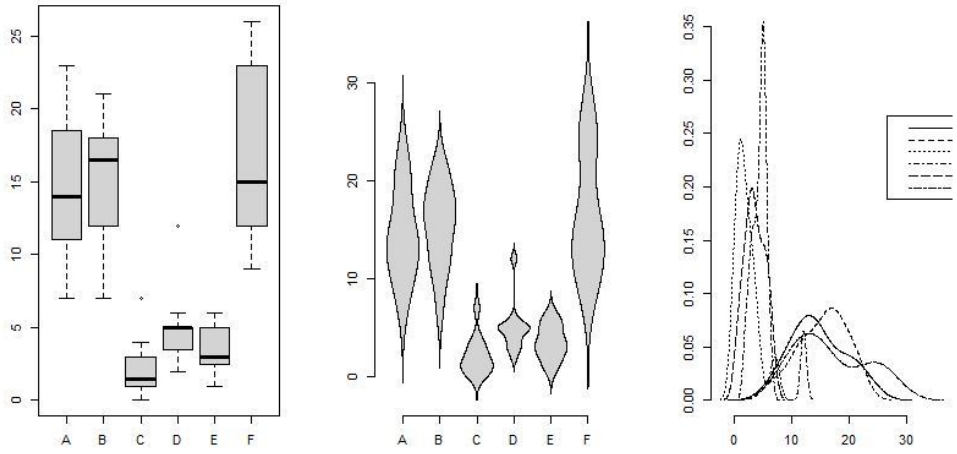
Функције `simple.violinplot` и `simple.densityplot` могу бити коришћене уместо боксплотова (један поред другог) да би се упоредиле различите расподеле. Обе користе емпиријску густину пронађену преко функције `density`, да би се илустровала расподела

случајне променљиве. Густина се може замислити као хистограм, само што има значајно мање „смећа” по графику, тако да се више сетова ефикасније може сместити на исти график.

График густине исцртава неколико густина на истој скали. Више хистограма изгледа лоше, али више густина изгледа подношљиво. Да подсетимо, **виолински дијаграм** је веома сличан боксплоту, само што је кутија замењена густином којој је додата слика у огледалу да би била јаснија.

Као илустрација је приказан исти скуп података на сва три графика на слици 6.3. График густине изгледа мало претрпано, али јасно се може видети да има два типа расподела. Треба приметити да се функције користе на сличан начин као код боксплота:

```
> par(mfrow=c(1,3)) # 3 графика по страни
> data(InsectSprays) # учитавање података
> boxplot(count~spray,data=InsectSprays,col="lightgray")
> simple.violinplot(count~spray,data=InsectSprays,col="lightgray")
> simple.densityplot(count~spray,data=InsectSprays)
```



Слика 6.3: Поређење боксплота, виолинског графика и графика густине за исте податке

6.5.6 График расејања

Нека су x предиктор (независна променљива), а y и z променљиве одзива. Ако је потребно исцртати их на истом графику, али са различитим ознакама, то се може учинити командом `pch`. Пример:

```
> plot(x,y) # обичан график расејања
> points(x,y,pch=2) # додајемо још једну променљиву
```

Треба приметити да друга команда није `plot` него `points`, која додаје на постојећи график, за разлику од `plot` која црта потпуно нови график.

Понекад су дати подаци x и y, који су такође разбијени по задатом фактору. Потребно је исцртати график расејања за податке x и y, али користећи различите ознаке графика за различите нивое фактора. Да би се то постигло, потребно је користити нивое фактора за ознаке. Нивои су интерно сачувани као бројеви и њих интерно користимо за вредности `pch`.

Пример: Раст зуба

`R`-ов уграђени скуп података `ToothGrowth` је потекао из студије која мери раст зуба као функцију количине витамина C. Извор витамина C су сок од поморанџе са једне стране и витаминski суплемент са друге стране. График расејања који приказује дозу наспрам дужине дат је на слици 6.4. Приметити различите симболе за два нивоа фактора који представља извор витамина C.

```
> data("ToothGrowth")
> attach(ToothGrowth)
> plot(len ~ dose,pch=as.numeric(supp)) ## кликнути мишем за додавање легенде
> tmp = levels(supp) # сачувај на тренутак
> legend(locator(1),legend=tmp,pch=1:length(tmp))
> detach(ToothGrowth)
```

Са графика делује да је за све вредности дозе, витаминска форма (VC) била мање ефикасна од сока (OJ).

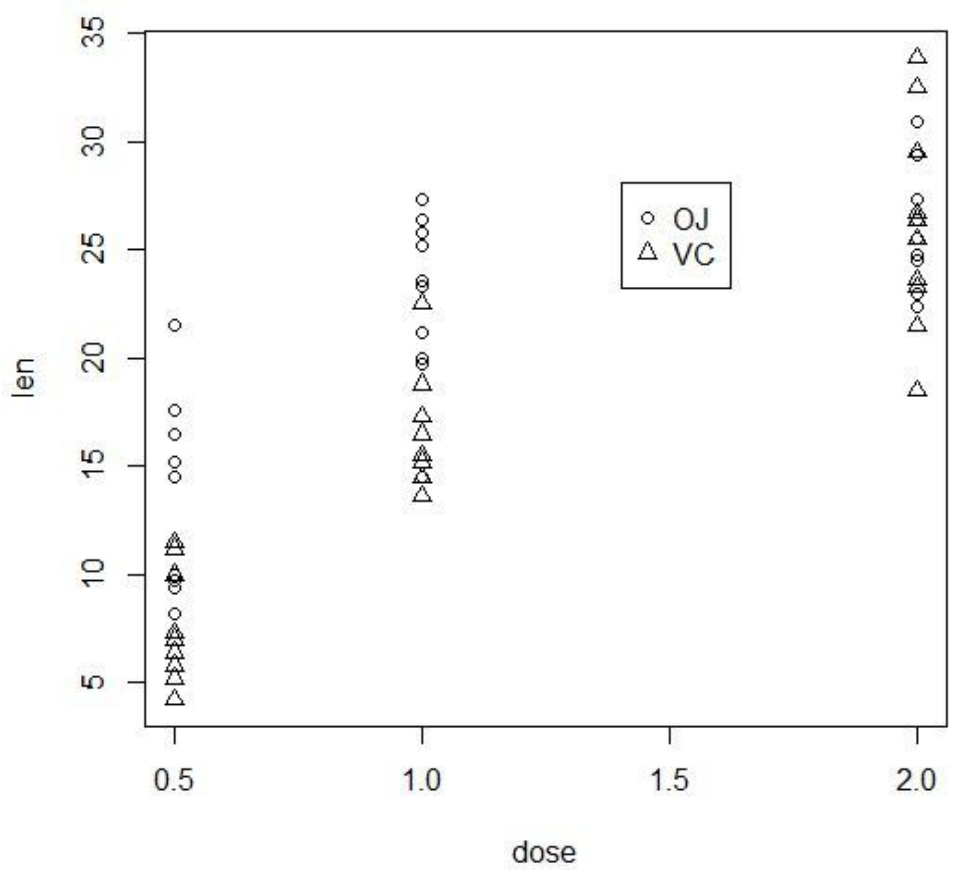
Понекад је потребно посебно размотрити расподеле x и y, али такође и њихову повезаност. Ово је лакше ако се може исцртати више графика одједном. Функција `simple.scatterplot` из пакета `UsingR` у томе може помоћи.

Пример: БДП наспрам емисије CO_2

Питање емисије CO_2 је тренутно веома актуелна тема због јасног утицаја на климатске промене. Скуп података `emissions` садржи податке за бруто домаћи производ (БДП), БДП по глави становника и CO_2 емисију у неколико европских земаља и САД за 1999. годину. График расејања на слици 6.5 је занимљив.

```
> data(emissions)
> attach(emissions)
> simple.scatterplot(perCapita,CO2)
> title("GDP/capita vs. CO2 emissions 1999")
> detach(emissions)
```

Приметити да са додатним информацијама на овом графику расејања можемо видети да је дистрибуција БДП прилично раширена за разлику од емисије CO_2 која очигледно има један усамљени податак.



Слика 6.4: Раст зуба као функција дозе витамина C

6.5.7 Упарени графици расејања

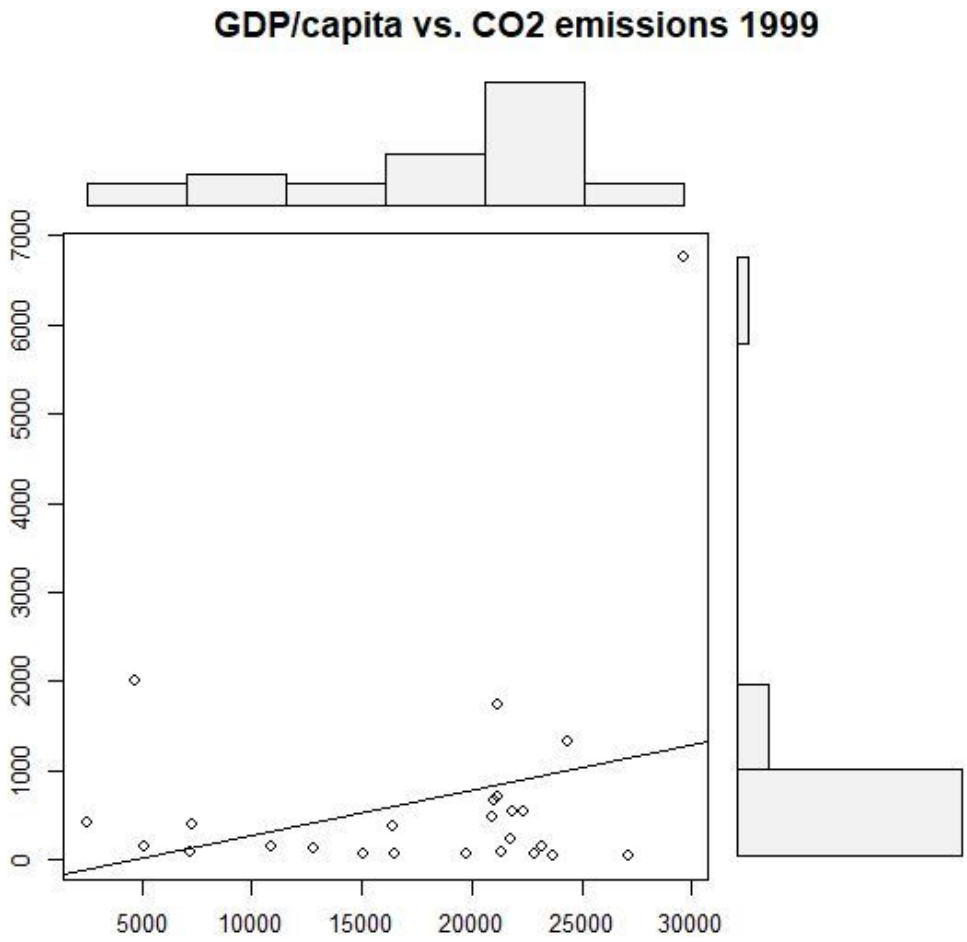
Ако три променљиве садрже нумеричке вредности, онда су графици расејања прикладан начин приказа. Команда `pairs` ће произвести график расејања за сваки могући пар варијабли. Може бити коришћена као `pairs(cbind(x,y,z))`, `pairs(df)` или у фактор облику `pairs(data.frame(split(times,week)))`. Најједноставније је када се подаци налазе у оквиру података. Ако нису у оквиру, потребно је позвати команду `cbind` која везује променљиве као колоне. Слика 6.6 даје пример коришћења `emissions` скупа података.

```
> pairs(emissions)
```

Команда `pairs` има велики број опција за уређивање графика. На страни за помоћ могу се наћи два поучна примера.

6.5.8 Пакет `lattice`

Додатни пакет `lattice` имплементира `Trellis` графичке концепте Била Кливленда. Он и пратећи мрежни пакет омогућавају различите начине посматрања мултиваријантних података поред горе описаних. Од верзије 1.5.0 ово су препоручени пакети, али нису и



Слика 6.5: БДП наспрам CO_2 емисије

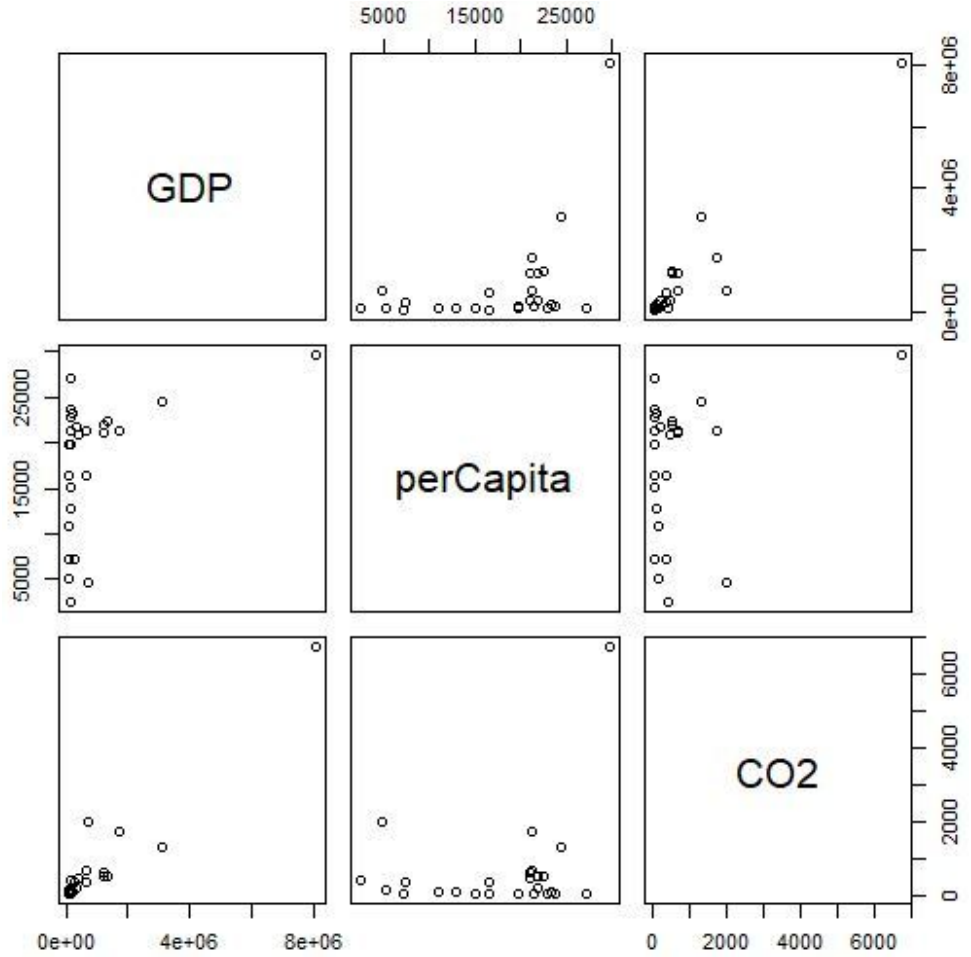
део базне верзије `R`-а. Неки корисни графици се једноставно креирају и приказани су у наставку.

За илустрацију се користи скуп података `Cars93`. Претпостављамо да је учитан, али није приложен како би илустровао употребу података `data=below`. Основна идеја је да се график састоји од више панела. Типично, ово одговара одређеној вредности условне променљиве. Функције се позивају са ознаком модела формуле. За универијантне графике као што су хистограми, променљива одзива, тј. лева страна је празна. За биваријантне графике је дата. Приметити да су имена функција природна, али још увек другачија од оних које користи базни `R`. На пример, користи се `histogram` уместо `hist`.

Хистограми

Хистограми су универијантни. Следећа команда приказује хистограм максималне цене условљене бројем цилиндара. Запажа се да на месту променљиве одзива остаје празно место:

```
> histogram( ~ Max.Price | Cylinders , data = Cars93)
```



Слика 6.6: Употреба `pairs` са `emissions` скупом података

Боксплотови

Боксплотови су такође униваријантни. Команда је `bwplot`:

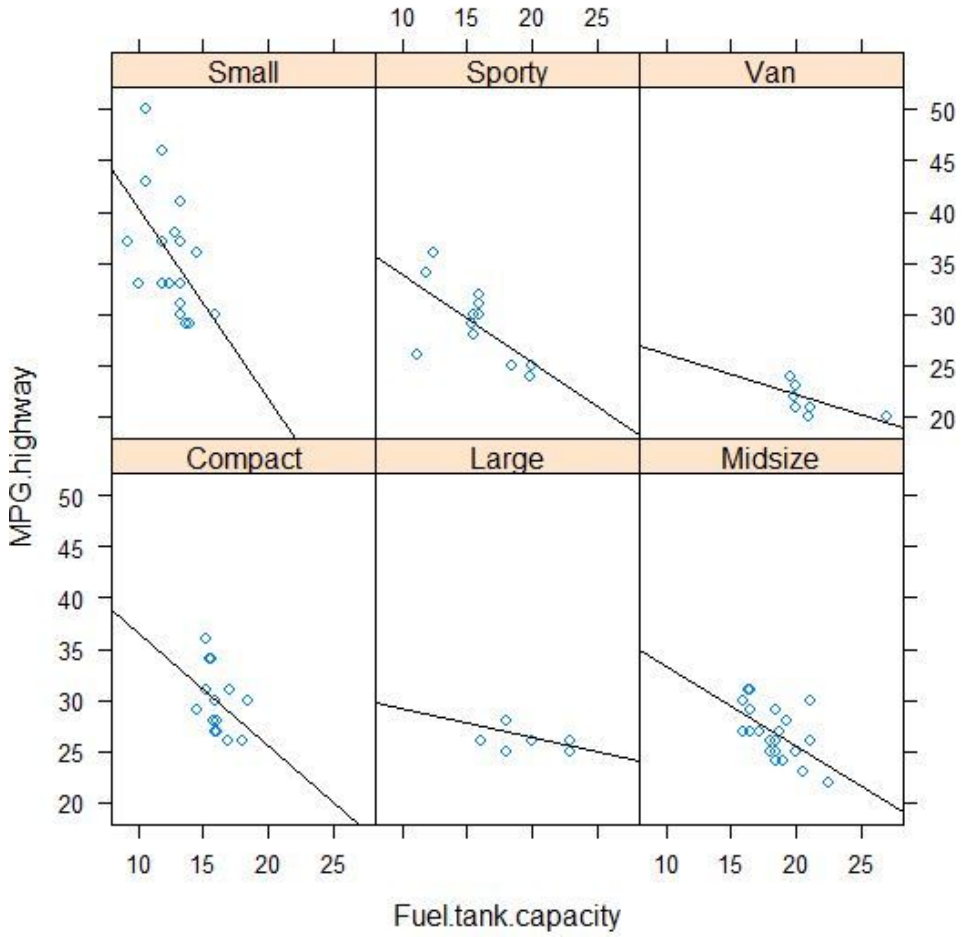
```
> bwplot( ~ Max.Price | Cylinders , data = Cars93)
```

Графици расејања

Графици расејања су такође доступни. Функција је `xyplot`, за разлику од базног `plot`. Како су подаци биваријантни, променљива одговора је неопходна. Следећи код црта однос између MPG (*Miles Per Gallon*) и величине резервоара. Очекујемо да аутомобили са бољом потрошњом могу имати мање резервоаре, слика 6.7. Ова врста графика нам омогућава да проверимо и да ли је однос исти за све типове аутомобила:

```
> attach(Cars93) # није нам потребан Cars93 у овом тренутку
> xyplot(MPG.highway ~ Fuel.tank.capacity | Type)
## плотирамо са регресионом линијом
## прво плотирамо регресиону линију цртајући функцију
```

```
> plot.regression = function(x,y) {
+ panel.xyplot(x,y)
+ panel.abline(lm(y~x))
+ }
> trellis.device(bg="white") # мењамо позадину у бело
> xyplot(MPG.highway ~ Fuel.tank.capacity | Type, panel = plot.regression)
%како да обезбедимо да текст не прелази маргине?
```



Слика 6.7: Пример `lattice` графика: Однос `MPG` и величине резервоара

Неки трендови се могу видети са саме слике. Изгледа да нагиб постаје мање стрм када се величина возила повећава. Обратити пажњу на команду `trellis.device` која подешава боју позадине на бело. Подразумеване боје су помало тамне. Цртеж укључује и регресиону линију. Ово је постигнуто функцијом за креирање панела. Подразумевано, `xyplot` ће користити функцију `panel.xyplot` да би формирао график расејања. Да би се добило више, дефинисали смо функцију од `x` и `y`, која је креирала график расејања (опет са `panel.xyplot`) и додала регресиону линију користећи функције `panel.abline` и `lm`.

Задаци

1. За скуп података `emissions` постоји нетипична вредност за емисију CO_2. Пронаћи ову вредност користећи `identify`, а затим поновити график без ове тачке.

2. Скуп података `chips` садржи податке о дебљини интегрисаних кола. Постоје подаци за два чипа, од којих је сваки мерен на 4 различита места. Направити *side by side* график за свако место мерења. На истом графику треба да буде 8 боксплотова. Да ли је средња вредност иста? Разлике?

3. *UsingR* скуп података `chicken` садржи телесне масе кокошака којима се даје једна од три врсте оброка. Направити боксплот за све три врсте оброка. Да ли изгледа да има разлика у средњој вредности?

4. *UsingR* скуп података `WeightData` садржи информације о телесној тежини деце узраста од 0 до 144 месеца. Направити *side by side* график распоређен по годинама. Које трендове уочавамо? Променљива `age` дата је у месецима. Претварање у године се може извршити коришћењем `cut` као што следи:

   ```
   > age.yr = cut(age,seq(0,144,by=12),labels=0:11)
   ```

   под претпоставком да је скуп података прикачен.

5. *UsingR* скуп података `Carbon` садржи концентрације угљен-моноксида на три различите индустријске локације. Подаци имају две променљиве: очитавање угљен-моноксида и факторску променљиву која прати локацију. Направити *side by side* график нивоа угљен-моноксида за сваку локацију. Да ли изгледа да постоји разлика?

6. За скуп података `babies` направити упарени график `(pairs(babies))` за истраживање односа између променљивих. Које променљиве изгледа да имају линеарни однос? За променљиве телесне тежине на рођењу и дужине трудноће креирати график који различитим карактерима (`pch`) приказује нивое фактора пушења.

Глава 7

Случајни подаци

Поред могућности да ради са реалним подацима који се учитавају из фајлова различитих формата или путем URL адреса, R поседује могућност генерисања случајних података.

```
sample(1:6,10,replace=T)
 [1] 6 4 1 1 3 1 4 2 4 6
```

или помоћу новокреиране функције

```
> RollDie = function(n) sample(1:6,n,replace=T)
> RollDie(5)
[1] 5 3 2 3 1
```

Заправо, R може да креира пуно различитих врста случајних бројева од познатих фамилија расподела до неких сасвим специјализованих расподела.

7.1 Генератори случајних бројева

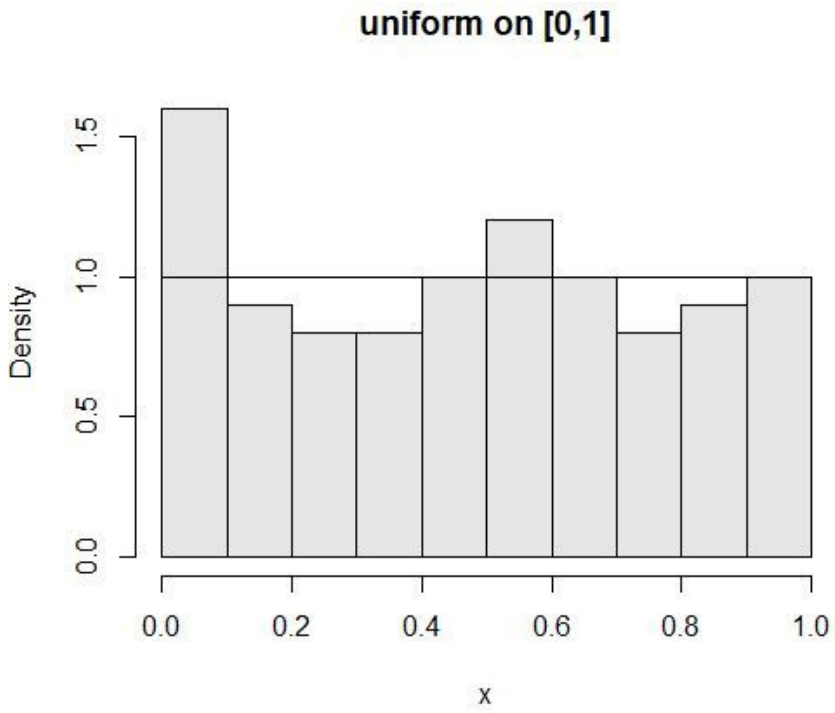
Као што је познато, случајни бројеви су описани расподелом. **Расподела** је функција која описује вероватноћу да се случајни број налази у неком интервалу, нпр. $P(a < X \leq b)$. Расподела се може задати густином вероватноће (у континуалном случају) или функцијом $P(X = k) = f(k)$ у дискретном случају. R има могућност да генерише случајне бројеве извучене из пуно различитих расподела. Да би се ова могућност користила, потребно је упознати се са параметрима који се дају функцијама као што су нпр. `mean` или `rate`. У наставку су примери најчешћих расподела. За сваки је дат хистограм за случајни узорак величине 100 и густина расподеле користећи функције са префиксом `d`.

7.1.1 Униформна расподела

Униформни случајни бројеви су они за које је *једнако вероватно* да се налазе у датом интервалу. Када се генеришу на рачунару, углавном се ради о интервалу [0,1], али у

пракси могу бити у произвољном интервалу [*a*,*b*], где *a* и *b* зависе од поставке проблема. На пример, време чекања на семафору може бити униформно на интервалу $[0, 2]$ (у минутима):

```
> runif(1,0,2) # време чекања на семафору
[1] 0.7850514 # такође runif(1,min=0,max=2)
> runif(5,0,2) # време чекања за 5 светала
[1] 0.4295075 0.7865266 0.6133130 1.6936797 0.8875498
> runif(5) # 5 случајних бројева из [0,1]
[1] 0.9488014 0.2816744 0.2567723 0.2196983 0.6296562
```



Слика 7.1: Стотину униформних случајних бројева на интервалу [0,1]

Општи облик је `runif(n,min=0,max=1)`, чиме је могуће конфигурисати колико случајних бројева је потребно (n) и интервал из којег се бирају ($[min, max]$). Да би се приказала расподела бројева за $min = 0$ и $max = 1$ (подразумевано, слика 7.1) потребно је:

```
> x=runif(100)
> hist(x,probability=TRUE,col=gray(.9),main="uniform on [0,1]")
> curve(dunif(x,0,1),add=T)
```

Једина компликованија ставка у претходном листингу била је исцртавање хистограма са позадинском бојом. Приметити како је функција `dunif` (густина расподеле) коришћена са функцијом `curve`.

7.1.2 Нормална расподела

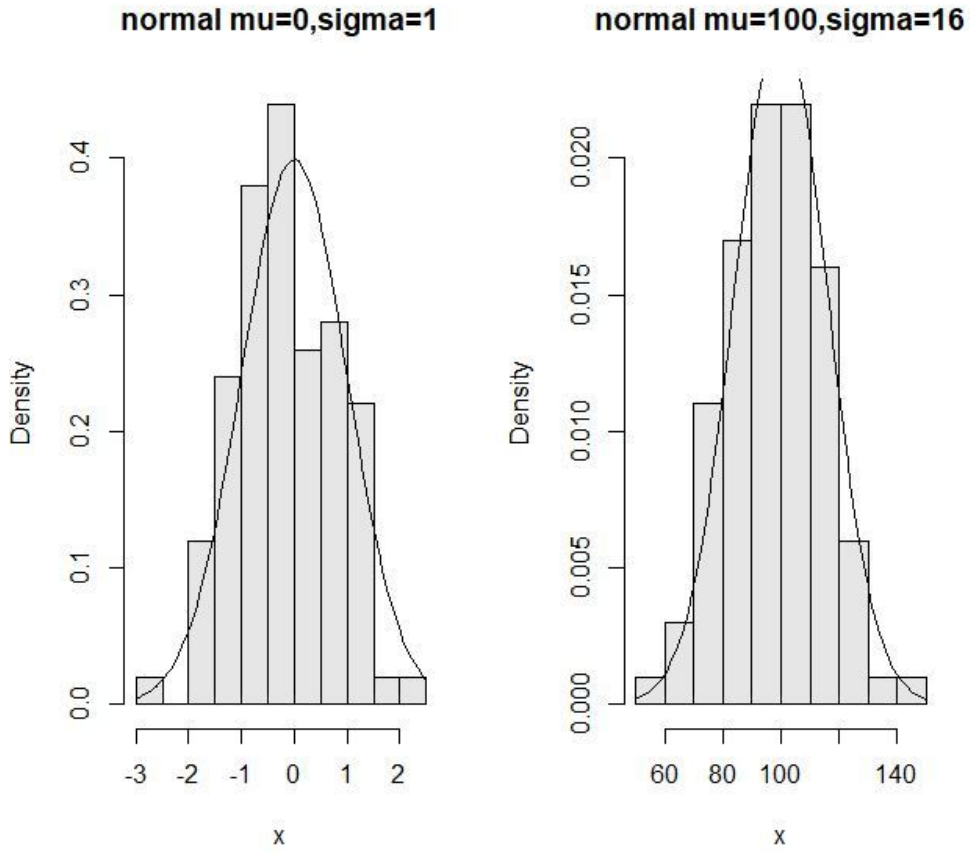
Нормални случајни бројеви су основа класичне статистичке теорије, због тзв. **Централне граничне теореме**. Нормална расподела има два параметра, средњу вредност μ и стандардно одступање σ. Ово су тзв. параметри локације и простирања.

На пример, IQ неке популацији је нормалне расподеле са средњом вредношћу 100 и стандардном девијацијом 16. Време развоја плода код људи може бити бити опуисано нормалном расподелом средње вредности 280 и стандардне девијације 10. Нормална расподела може бити стандардизована са средњом вредношћу 0 и варијансом 1. Ово се постиже *стандардизовањем* вредности помоћу формуле

$$Z = \frac{X - \mu}{\sigma}.$$

Ево неких примера:

```
> rnorm(1,100,16) # IQ
[1] 94.1719
> rnorm(1,mean=280,sd=10) # колико дана до порођаја (10 дана раније)
[1] 270.4325
```



Слика 7.2: Нормална расподела (0,1) и нормална расподела(100,16)

Функција `rnorm(n,mean=0,sd=1)` одређује и средњу вредност и стандардну девијацију. На слици 7.2 приказани су хистограми две нормалне расподеле.

```
> x=rnorm(100)
> hist(x,probability=TRUE,col=gray(.9),main="normal mu=0,sigma=1")
> curve(dnorm(x),add=T)
```

7.1.3 Биномна расподела

Биномни случајни бројеви су *дискретни* случајни бројеви. То је, у ствари, расподела броја успеха у n независних Бернулијевих екперимената. Један Бернулијев експеримент завршава успехом или неуспехом, а успех се дешава са вероватноћом p. Један Бернулијев оглед је дат са $n = 1$ у биномној расподели:

```
> n=1, p=0.5 # постави вероватноћу
> rbinom(1,n,p) # различита сваки пут
[1] 1
> rbinom(10,n,p) # 10 таквих различитих бројева
[1] 0 1 1 0 1 0 1 0 1 0
```

Број са биномном расподелом је једнак броју јединица у n таквих Бернулијевих случајних бројева. За последњи пример, то би била вредност 5. Дакле, расподела зависи од два параметра, и то n (број Бернулијевих огледа) и p (вероватноћа успеха). Да би се генерисали биномни бројеви, једноставно се промени вредност n од 1 до жељеног броја огледа. На пример, са $n = 10$ огледа бацања новчића где је вероватноћа добијања „главе" у сваком од њих $p = 0.5$:

```
> n = 10; p=0.5
> rbinom(1,n,p) # 6 успеха у 10 огледа
[1] 5
> rbinom(5,n,p) # 5 биномних случајних бројева
[1] 5 5 3 5 8
```

Број успеха је наравно дискретан, али када се n повећава, расподела почиње да личи на нормалну расподелу. Ово је последица Централне граничне теореме која у општем случају каже да:

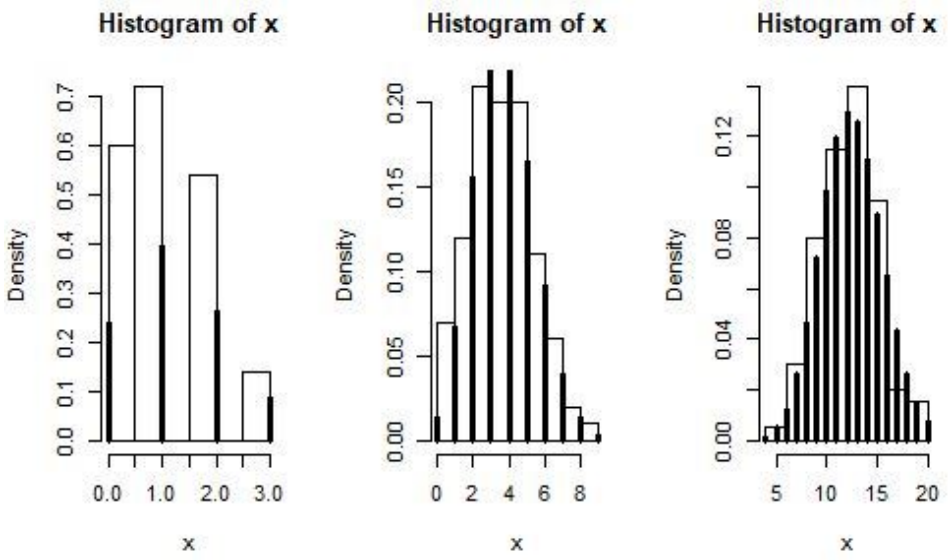
$$\frac{\overline{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

има нормалну расподелу. У нашем конкретном случају биномне расподеле

$$\frac{\hat{p} - p}{\frac{\sqrt{pq}}{n}},$$

где је $p = broj\_uspeha/n$, а $q = 1 - p$, треба да има нормалну расподелу.

Дијаграми на слици 7.3 показују 100 биномно распоређених случајних бројева за три вредности n (5,15,50) и за константно $p = 0.25$. Обратити пажњу да када се n повећава, облик постаје све више звоноаст. Ови дијаграми су креирани следећим командама:



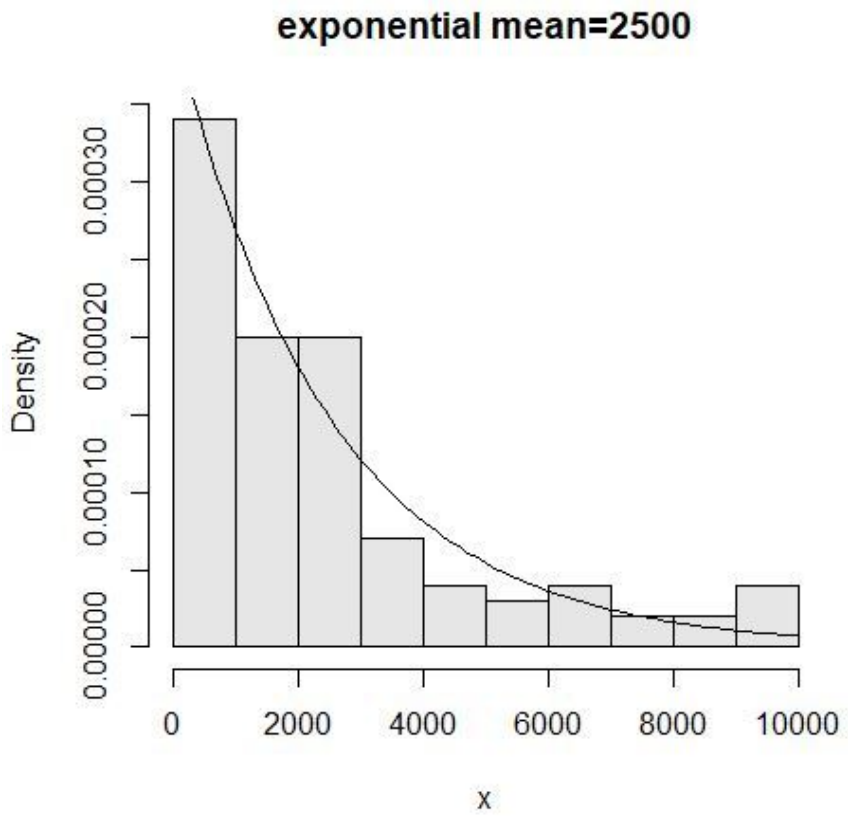
Слика 7.3: Случајни подаци са биномном расподелом

```
> par(mfrow=c(1,3))
> n=5;p=.25 # променити по потреби
> x=rbinom(100,n,p) # 100 случајно изабраних бројева
> hist(x,probability=TRUE,) # користити тачке, а не криву, дискретно
> xvals=0:n;points(xvals,dbinom(xvals,n,p),type="h",lwd=3)
> n=15;p=.25
> x=rbinom(100,n,p)
> hist(x,probability=TRUE,)
> xvals=0:n;points(xvals,dbinom(xvals,n,p),type="h",lwd=3)
> n=50;p=.25
> x=rbinom(100,n,p)
> hist(x,probability=TRUE,)
> xvals=0:n;points(xvals,dbinom(xvals,n,p),type="h",lwd=3)
```

7.1.4 Експоненцијална расподела

Експоненцијална расподела је важна у теоријском и практичном домену. На пример, користи се за опис животног века електричних компоненти. На пример, ако је просечни век сијалице 2500 сати, може се сматрати да је њен животни век случајан са експоненцијалном расподелом средње вредности 2500. Један од параметара је тзв. стопа, која се рачуна као $rate = 1/\mu$. Наредба за генерисање случајних бројева користи једноставну синтаксу `rexp(n,rate=1)`. Пример са стопом 1/2500 приказан је на слици 7.4.

```
> x=rexp(100,1/2500)
> hist(x,probability=TRUE,col=gray(.9),main="exponential mean=2500")
> curve(dexp(x,1/2500),add=T)
```



Слика 7.4: Хистограм података који одговарају експоненцијалној расподели

Постоје и друге расподеле од интереса у статистици. Уобичајене су Поасонова, Студентова t-расподела, F-расподела, β расподела и χ^2-расподела.

7.2 Узимање узорака са или без замене

R има могућност узимања узорка са и без замене. Наиме, узорковање се може вршити са заменом (попут бацања коцкице) или без замене (попут *Lotto*-а када број не може да се понови). **Подразумевано, `sample` генерише узорке без замене** и сваки објекат има једнаке шансе да буде изабран. Мора се поставити `replace=T` ако се жели замена узорка. Даље, могу се одредити одвојене вероватноће за сваку замену ако је потребно. Ево неколико примера:

```
## бацање коцкице
> sample(1:6,10,replace=TRUE)
 [1] 6 6 2 2 5 5 1 3 5 6 # нема четворки!
## бацање новчића
> sample(c("H","T"),10,replace=TRUE)
 [1] "T" "T" "T" "T" "T" "T" "H" "T" "T" "T"
## бирање 6 од 54 (лото)
> sample(1:54,6) # нема замене
```

```
[1] 43 45 14 13  6  8
## бирање карте (користи се \texttt{paste, rep})
> cards = paste(rep(c("A",2:10,"J","Q","K"),4),c("H","D","S","C"))
> sample(cards,5)
[1] "7 D" "8 S" "Q S" "8 H" "2 C"
## 2. бацање коцкице, модерније
> dice = as.vector(outer(1:6,1:6,paste))
> sample(dice,5,replace=TRUE) # укључена замена
[1] "5 1" "6 3" "2 6" "3 6" "2 5"
```

Последње две наредбе илуструју примере који се могу направити са релативно мало кућања и пуно размишљања, користећи команде `paste` за спајање стрингова, `rep` за понављање и `outer` за генерисање свих могућих комбинација.

7.3 *Bootstraping* узорка

Bootstraping је метод узорковања из скупа података, како би се извео статистички закључак. Интуитивна идеја је да се из неког мањег узорка може добити нека идеја о варијабилности података. Процес укључује више избора узорака и затим формира статистику. Ево једноставне илустрације о добијању узорка.

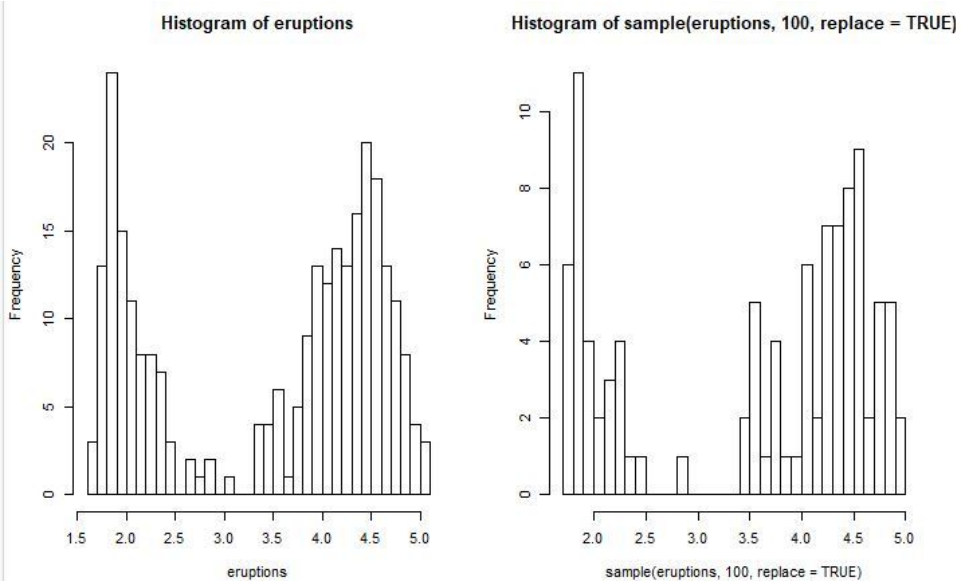
Уграђени скуп података `faithful` има променљиву `eruptions` која мери време између ерупција гејзира под називом `faithful`. Расподела је прилично необична. Узорковање `bootstrap` методом је заправо обично узорковање са заменом из датих вредности. Добија се на следећи начин:

```
> data(faithful) # део R-ове базе
> names(faithful) # нађи имена колона за faithful
[1] "eruptions" "waiting"
> eruptions = faithful[['eruptions']] # или закачи и откачи faithful
> sample(eruptions,10,replace=TRUE)
[1] 2.03 4.37 4.80 1.98 4.32 2.18 4.80 4.90 4.03 4.70
> hist(eruptions,breaks=25) # скуп података
## bootstrap узорак
> hist(sample(eruptions,100,replace=TRUE), breaks=25)
```

На слици 7.5 се примећује релативно мала разлика између расподеле на графику свих података (лево) и узорка од 100 података (десно).

7.4 Префикси `d`, `p` и `q`

Функције са префиксом **d**, као `dnorm` су већ коришћене за плотовање горенаведених теоретских густина расподеле. Као и код функција са префиксом `r`, потребно је навести параметре расподеле. `p` и `q` функције су за кумулативне функције вероватноће и квантиле. Као што је поменуто, расподела случајног броја је специфицирана вероватноћом да се



Слика 7.5: Пример Bootstraping методе

узорак нађе између a и b, што се обележава са $P(a < X \leq b)$. У ствари, довољна је вредност тзв. кумулативне функције вероватноће $F(x) = P(X \leq b)$, јер је:

$$P(a < X \leq b) = P(X \leq b) - P(X \leq a) = F(b) - F(a)$$

Дакле, `p` функције дају одговор колика је вероватноћа да је случајна променљива мања од x:

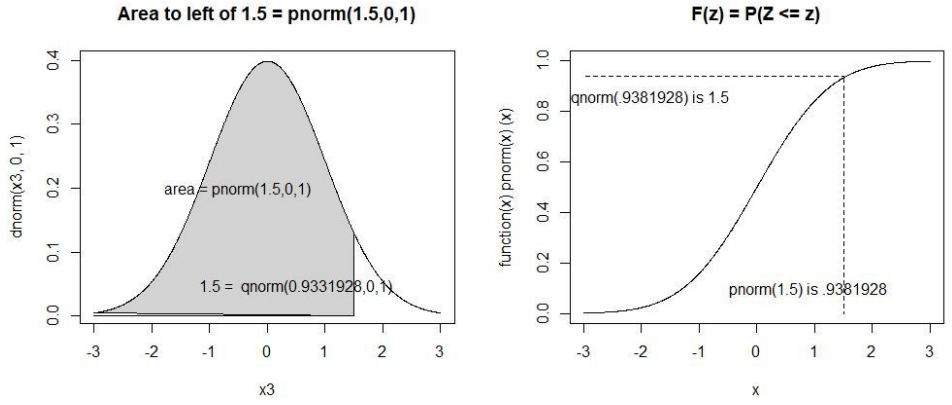
```
> pnorm(0.7)
[1] 0.7580363
> pnorm(0.7,1,1)
[1] 0.3820886
```

Горенаведено даје одговор на $P(X \leq 0,7) = F(0,7)$, где је X стандардна нормална или нормална (1,1) густина расподеле. Да би се се израчунала вероватноћа да $P(X > 0,7)$ довољно је $1-P(X \leq 0,7) = 1 - F(0,7)$ или пустити `R` да то уради сам специфицирањем `lower.tail=F`:

```
> 1-pnorm(0.7)
[1] 0.2419637
> pnorm(0.7,lower.tail=F)
[1] 0.2419637
```

```
> par(mfrow=c(1,2))
> x = seq(-3,3,length=1000)
> y = pnorm(x,mean=0,sd=1)
> plot(x,y,type="l",lwd=1,xlab="x",ylab="function(x) pnorm(x) (x)")
> title(main="F(x) = P(X <= x)")
```

```
> clip(-3,1.5, 0,0.9381928)
> abline(v=1.5,lty=2, h=0.9381928)
> clip(-5,5,-1,2)
> temp =locator(1) # кликнути на график на месту где треба да се појави текст
> text(temp,"pnorm(1.5) is 0.9381928")
> temp = locator(1) # поново кликнути на график
> text(temp,"qnorm(0.9381928) is 1.5")

> x = seq(-3,3,length=1000)
> y = pnorm(x,mean=0,sd=1)
> plot(x,y,type="l",lwd=1,xlab ="x",ylab = "function(x) pnorm(x) (x)")
> title(main ="F(x) = P(X <= x)")
> clip(-3,1.5, 0,0.9381928)
> abline(v=1.5,lty=2, h=0.9381928)
> clip(-5,5,-1,2)
> temp =locator(1) # кликнути на график
```



Слика 7.6: Илустрација за p и q функције

Префикс `q` је инверзна функција `p`. Даје одговор која вредност одговара датој вероватноћи. На пример, која вредност x одговара вероватноћи 0,75 подручја улево за стандардну нормалну расподелу (ово је $Q3$)?

```
> qnorm(0.75)
[1] 0.6744898
> pnorm(0.6744898)
[1] 0.75
```

Непознато x се налази речавањем једначине $0,75 = P(X \leq x) = F(0,75)$. Дакле, `qnorm` је у математичкој нотацији $F^{-1}(P)$.

7.5 Стандардизација, скала и *z* резултати

Да би се случајна променљива стандардизовала, одузме се средња вредност, а затим добијено подели са стандардном девијацијом:

$$Z = \frac{X - \mu}{\sigma}.$$

Овај рачун захтева познавање средње вредности и стандардне девијације. Такође се може стандардизовати и узорак. Постоји погодна функција која ће то урадити уместо нас и учинити да наш узорак има средњу вредност 0 и стандардну девијацију 1. Ова особина је корисна за упоређивање случајних променљивих чије се скале значајно разликују. Нормалне случајне променљиве су често стандардизоване пошто је расподела стандардизоване нормалне променљиве поново нормална са средњом вредношћу 0 и варијансом 1. Z-резултат нормалне вредности је вредност након стандардизације. Нека су дати нормални случајни подаци са средњом вредношћу 100 и стандардним одступањем 16. Онда ће z-резултати бити:

```
> x = rnorm(5,100,16)
> x
[1] 120.59601  85.31513 102.59779 119.39866  95.10614
> z = (x-100)/16
> z
[1]  1.2872506 -0.9178045  0.1623620  1.2124164 -0.3058664
```

Z-резултат може да се искористи да се очитају кумулативне вероватноће за дату случајну променљиву x. Међутим, са R-ом, ово није неопходно, јер се може директно користити функција `pnorm` навођењем средње вредноссти и стандардне девијације:

```
> pnorm(z)
[1] 0.9009965 0.1793606 0.5644896 0.8873235 0.3798532
> pnorm(x,100,16)
[1] 0.9009965 0.1793606 0.5644896 0.8873235 0.3798532
```

Задаци

1. Генерисати 10 случајних бројева из униформне расподеле на интервалу [0,10]. Искористити R за утврђивање максималних и минималних вредности X.

2. Генерисати 10 случајних нормалних бројева са средњом вредношћу 5 и стандардном девијацијом 5. Колико њих је мање од 0?

3. Генерисати 100 нормалних случајних бројева са средњом вредношћу 100 и стандардном девијацијом 10. Колико њих упада у интервал две стандардне девијације од средње вредности (мање од 80 или више од 120)?

4. Бацити новчић 50 пута (користећи R). Колико пута је пала „глава”?

5. Баците коцкицу 100 пута. Колико шестица сте видели?

6. Одаберите 6 бројева на Лотоу са 49 лоптица. Који је највећи број? Који је најмањи? Одговорити користећи `R`.

7. За нормалну расподелу (0,1) наћи број z решавајући P(Z ≤ z) = 0,05 (користити `qnorm`).

8. За нормалну расподелу (0,1), наћи број z решавајући $P(-z < Z \leq z) = 0,05$. Користити `qnorm` и симетрију.

9. Колико је велико подручје десно од 1,5 за нормалну расподелу (0,2)?

10. Направити хистограм од 100 експоненцијалних бројева са средњом вредношћу 10. Проценити медијану. Да ли је већа или мања од средње вредности?

11. Можете ли да утврдите шта ова наредба `R`-а ради?

```
> rnorm(5,mean=0,sd=1:5)
```

12. Користити `R` да бисмо изабрали 5 карата из шпила од 52 карте. Да ли добијамо пар или нешто боље? Поновити поступак док не добијемо пар. Колико итерације је било потребно?

Глава 8

Симулације

Симулација различитих типова података даје корисницима могућност да изводе експерименте и одговоре на питања брже него да изводе реалне експерименте. Ово је веома корисна вештина, али је није тривијално савладати.

Као што је приказано, R има већи број функција за генерисање случајних бројева. Њихова расподела може се приказати помоћу хистограма и других алата за визуелизацију. Сада нам је циљ генерисање нових типова случајних бројева и посматрање њихове расподеле.

8.1 Централна гранична теорема (ЦГТ)

За почетак је најбитније увести *Централну граничну теорему (ЦГТ)*. За произвољне X_i који се узимају из популације где су μ и σ познате константе, важи да је расподела

$$\frac{\overline{X} - \mu}{\sigma/\sqrt{n}}$$

асимптотски нормална са медијаном 0 и варијансом 1 (позната и као нормална расподела (0,1)). У преводу, за довољно велико n, расподела је приближна нормалној са медијаном μ и стандардном девијацијом $\sigma/\sqrt{n}$. Симулација је одличан начин да се ово потврди.

Прво ћемо моделом пробати да ли ЦГТ важи за биномну расподелу. По ЦГТ, ако S_n подлеже биномној расподели са параметрима n и p, онда

$$\frac{S_n - np}{\sqrt{np(1-p)}}$$

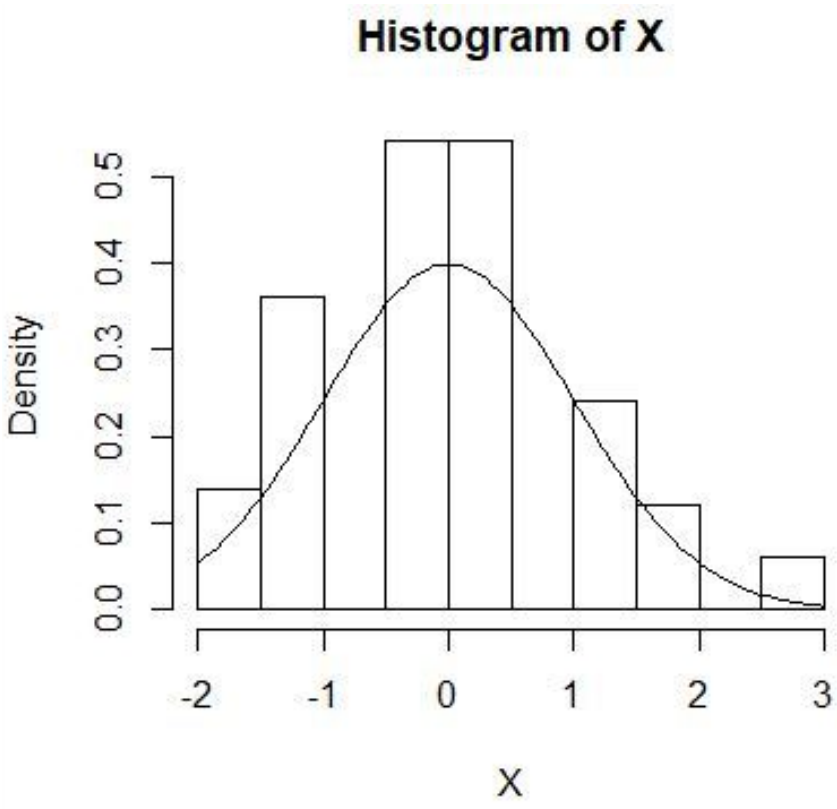
подлеже приближно нормалној расподели (0,1). Како се ово може употребити за генерисање неког од случајних бројева који подлежу претходном изразу?

```
> n=10;p=.25;S= rbinom(1,n,p)
> (S - n*p)/sqrt(n*p*(1-p))
[1] -0.3651484
```

Ово је само један од таквих случајних бројева. Потребно је пуно њих да би на хистограму била уочљива њихова расподела. На пример, 100 бројева се може једноставно генерисати - само се узме више узорака у функцији `rbinom`:

```
> n = 10;p = .25;S = rbinom(100,n,p)
> X = (S - n*p)/sqrt(n*p*(1-p)) # 100 случајних бројева
> hist(X,prob=T)
> x<-seq(-3,+3,by=0.01)
> curve(dnorm(x), add=TRUE)
```

Променљива X садржи резултате који се могу видети на хистограму на слици 8.1.



Слика 8.1: Скалирана биномна расподела је приближна нормалној (0,1)

8.2 `For` петље

Иначе, механизам за генерисање 100 случајних бројева можда неће увек бити тако једноставан, па треба узорковати бројеве један по један. Ово се може постићи `for` петљом, иако би неки корисници `R`-а радије користили команду `apply`. Команда `for` итерише кроз одређени скуп вредности као што су бројеви од 1 до 100. Ови резултати се чувају помоћу вектора додељивањем једне по једне вредности. Следи исти пример као претходни, али коришћењем `for` петље:

```
> results =numeric(0) # место за складиштење резултата
> for (i in 1:100) { # for петља
+ S = rbinom(1,n,p) # само једанпут
+ results[i]=(S- n*p)/sqrt(n*p*(1-p)) # сачувај одговор
+ }
```

Променљива `results` ће сачувати све одговоре. Затим се за свако i између 1 и 100, креира случајни број (сваки пут различит!) и чува у вектору `results` као i-ти унос. Резултати се могу видети помоћу хистограма навођењем `hist(results)`.

Синтакса за `for`

`For` петља има једноставну синтаксу:

```
for(променљива in вектор) {команда/е}
```

Велике заграде су опционе ако је у питању само једна команда, исто као и у програмском језику C. Променљива i се мења у свакој итерацији. Следе неки примери:

```
> primes=c(2,3,5,7,11);
## петља преко индекса
> for(i in 1:5) print(primes[i])
## или боље, петља директно
> for(i in primes) print(i)
```

8.3 ЦГТ са подацима из нормалне расподеле

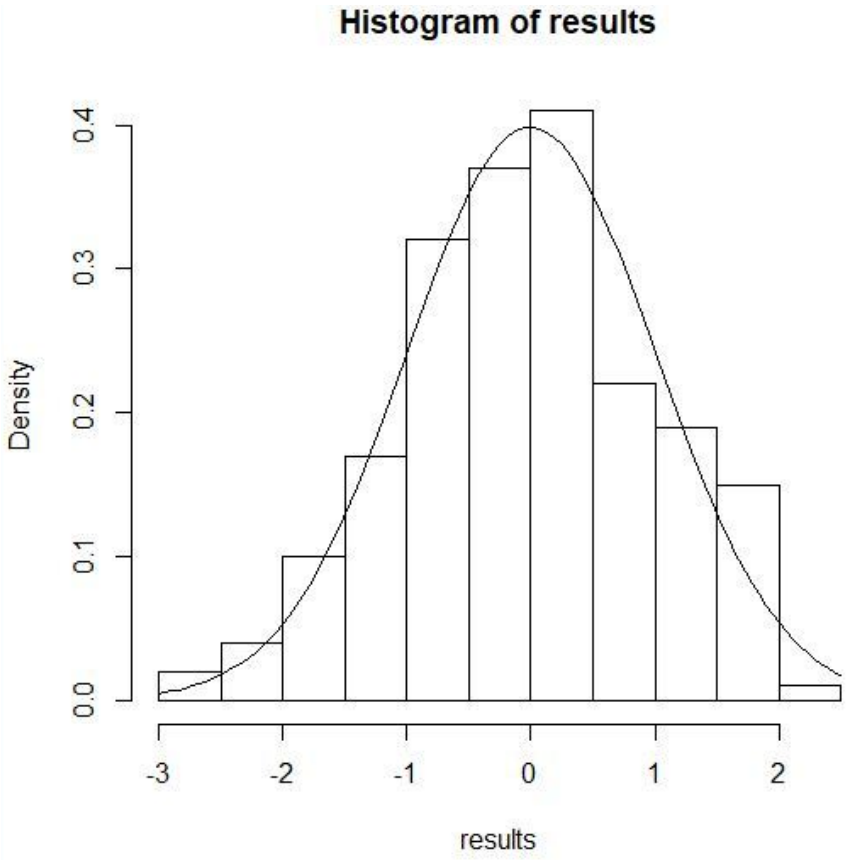
ЦГТ такође важи и за нормалну расподелу. Нека X_i подлеже нормалној расподели са медијаном $\mu = 5$ и стандардном девијацијом $\sigma = 5$. Потребна је функција за израчунавање вредности

$$\frac{(X_1 + X_2 + ... + X_n)/n - \mu}{\sigma/\sqrt{n}} = \frac{\overline{X} - \mu}{\sigma/\sqrt{n}} = (mean(X) - mu)/(sigma/sqrt(n))$$

Као і у претходном, може се искористити `for` петља

```
> results = c();
> mu = 0; sigma = 1
> for(i in 1:200) {
+ X = rnorm(100,mu,sigma) # генерисање случајних података
+ results[i] = (mean(X) - mu)/(sigma/sqrt(100))
+ }
> hist(results,prob=T)
```

Хистограм на слици 8.2 показује да је расподела приближно нормална.



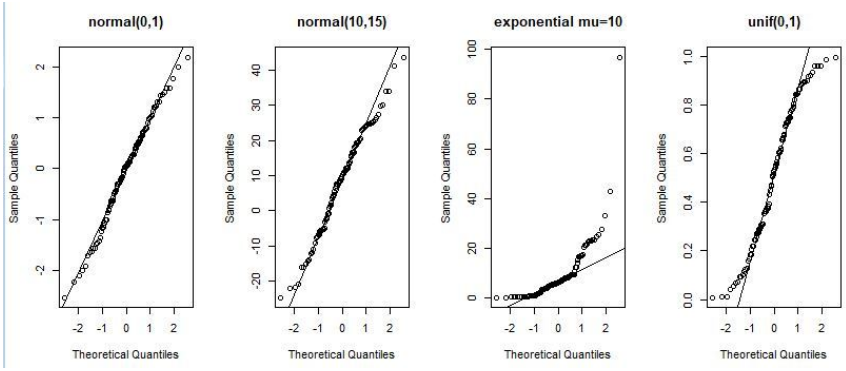
Слика 8.2: Симулација ЦГТ са нормализованим подацима. Примећује се облик звона.

8.4 Тестирање нормалности

Графичка представа напреднија од хистограма за утврђивање да ли је расподела приближно нормална је тзв. график **нормалности**. Основна идеја је да се графички представе квантили података наспрам теоријских квантила нормалне расподеле. Као што је већ објашњено, квантили скупа података имају исти смисао као медијана, Q_1 и Q_3, само што су уопштенији. Квантил q је тачка у којој је q 100% података из датог скупа мање. Тако је 0,25 квантил заправо $Q1$, 0,5 квантил је медијана, а 0,75 квантил је Q_3. Квантили за теоријску расподелу су слични, само је подручје са леве стране тачака података одређене величине. На пример, медијана дели подручје испод криве густине расподеле на половине. График нормалности је једноставан. У суштини, ако график изгледа као права линија, онда је расподела података приближно нормална. Било која кривина значи да расподела има кратке или дуге репове. Треба приметити да ово није права регресије. Ова права се црта кроз тачке формиране првим и трећим квантилом.

`R` олакшава све ово функцијама `qqnorm` (или општијом `qqplot`) и `qqline`, која црта референтну линију. Графици за неки узорак података дати су на слици 8.3. Прва два графика изгледају као нормалне расподеле, а друге две пак не.

```
> x=rnorm(100,0,1); qqnorm(x,main='normal(0,1)'); qqline(x)
> x=rnorm(100,10,15); qqnorm(x,main='normal(10,15)'); qqline(x)
```



Слика 8.3: Неки графици нормалности

```
> x=rexp(100,1/10); qqnorm(x,main='exponential mu=10'); qqline(x)
> x = runif(100,0,1); qqnorm(x,main='unif(0,1)'); qqline(x)
```

8.5 Употреба функције `simple.sim`

У овом одељку показан је начин за креирање функција које се користе у сарадњи са методом `simple.sim`. За потребе симулације, пожељно је да се `for` петља избегне уколико је то могуће. Функција `simple.sim` намењена је управо томе. Као аргумент `simple.sim` тражи управо функцију која генерише случајни број из произвољне расподеле.

На пример, у оквиру провере ЦГТ за биномну расподелу било је потребно да се генерише један случајни број са биномном расподелом. Функција која ово изводи је:

```
> f = function () {
+ S = rbinom(1,n,p)
+ (S- n*p)/sqrt(n*p*(1-p))
+ }
```

Са овом функцијом `simple.sim` се може искористити на следећи начин:

```
> x=simple.sim(100,f)
> hist(x)
```

Ово елиминише потребу за `for` петљом и чини симулације конзистентним. Када се креира функција за генерисање случајног броја, даље је једноставно. Добра пракса је дефинисање исправног начина за писање ових функција. У нашој функцији, омогућићемо модификацију броја покушаја n и вероватноћу успеха p, па је боље дефинисати f мало другачије:

```
> f = function(n=100,p=.5) {
+ S = rbinom(1,n,p)
+ (S- n*p)/sqrt(n*p*(1-p))
+ }
```

Формат за променљиве $n = 100$ говори да је n први параметар функције, подразумевано 100, p је други параметар и подразумевано је $p = 0.5$. Сада се `simple.sim` позива са

```
> simple.sim(1000,f,100,0.5)
```

Ево примера функције која враћа једну вредност:

```
> the.range = function (x) max(x) - min(x)
```

Ова функција враћа опсег вектора x. За `R` ово значи да је `the.range` функција, а њени аргументи се налазе у заградама, у овом случају (x).

Ако је функција сложенија и захтева више команди, користе се велике заграде (као за петље). Враћа се последња израчуната вредност. У примеру се израчунава IQR (интерквантилни распон) на основу доње и горње границе, а не квантила. Користимо резултате команде `fivenum` да бисмо добили границе:

```
> find.IQR=function(x){
+ five.num=fivenum(x)
+ five.num[4]-five.num[2]
+ }
```

Знак плус указује на нови ред и генерише га `R`, тј. не куца се. Добија се пет бројева: *минимум, доња границе, медијана, горња граница и максимум*, а дата функција одузима други од четвртог. Функција се позива по имену и са заградама. На пример:

```
> find.IQR=function(x){
+ five.num=fivenum(x)
+ five.num[4]-five.num[2]
+ }
> x=rnorm(100)
> find.IQR(x)
[1] 1.275791
```

8.6 Пример: Функција која сабира бројеве из нормалне расподеле

За налажење стандардизоване суме првих 100 бројева нормалне расподеле $(0, 1)$ може се користити:

```
> f = function(n=100,mu=0,sigma=1) {
+ nos = rnorm(n,mu,sigma)
+ (mean(nos)-mu)/(sigma/sqrt(n))
+ }
```

Затим се позива `simple.sim`:

```
> simulations = simple.sim(100,f,100,5,5)
> hist(simulations,breaks=10,prob=TRUE)
```

да би се добио хистограм нормалне расподеле (0,1).

8.7 ЦГТ са експоненцијалним подацима

Ако се почне са нагнутом расподелом, централна гранична теорема каже да ће ова расподела на крају личити на нормалну за довољно велико n. Шта значи *на крају*? Ово можемо да проверимо симулацијом.

Пример нагнуте дистрибуције је експоненцијална дистрибуција. Ако је медијана 10, онда је стандардна девијација такође 10, па је потребно само њу специфицирати. Следи функција за креирање једног стандардизованог просека (експоненцијална расподела има теоретску стандардну девијацију једнаку медијани):

```
> f = function(n=100,mu=10) (mean(rexp(n,1/mu))-mu)/(mu/sqrt(n))
```

Сада се симулација врши за различите вредности n. За сваку од ових, $m = 100$ (број генерисаних случајних бројева), али n варира од 1,5,15 и 50 (број случајних бројева у сваком од појединачних просека).

```
> xvals = seq(-3,3,.01) # за график густине расподеле
> hist(simple.sim(100,f,1,10),probability=TRUE,main="n=1",col=gray(.95))
> points(xvals,dnorm(xvals,0,1),type="l") # црта криву нормалне расподеле

... поновити за n=5,15,50
```

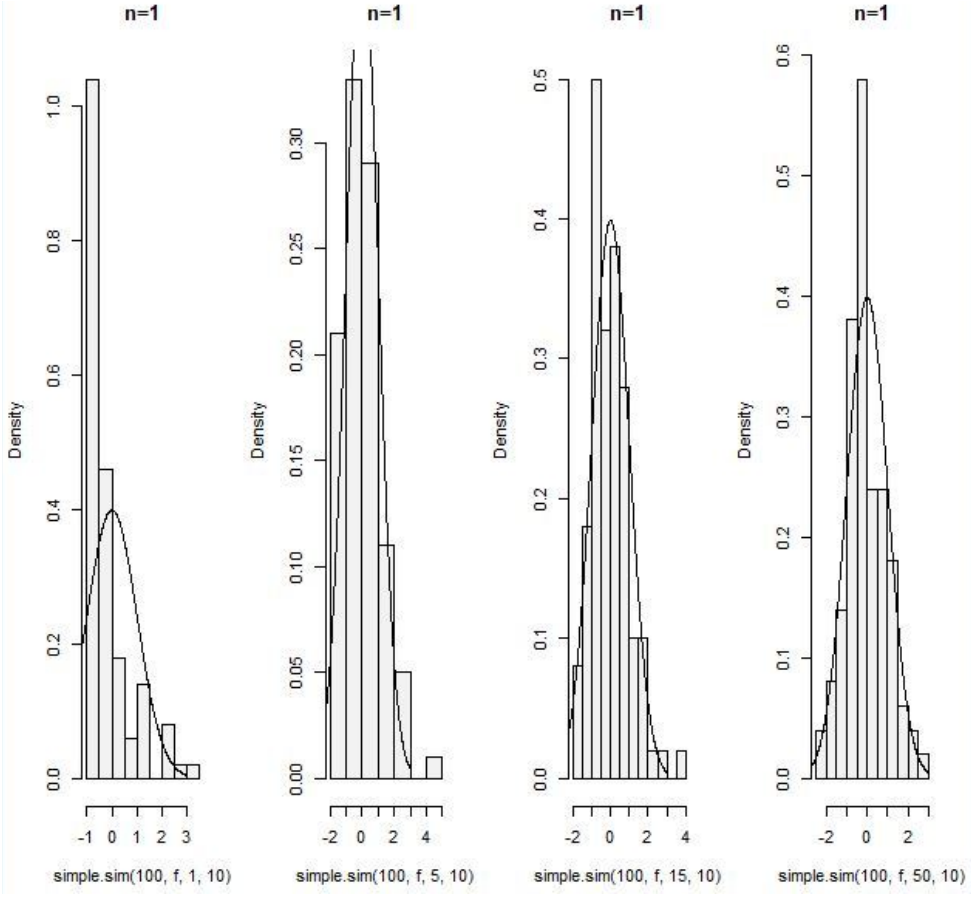
Хистограм на слици 8.4 добија звонасти облик између $n = 15$ и $n = 50$, мада на $n = 50$ изгледа и даље помало нагнуто.

Задаци

1. Урадити две симулације биномне расподеле са $n = 100$ и $p = 0.5$. Да ли добијамо исте резултате сваки пут? Шта је различито? Шта је слично?

2. Урадити симулацију нормалне расподеле два пута. Једну за $n = 10$, $\mu = 10$ и $\sigma = 10$, другу за $n = 10$, $\mu = 100$ и $\sigma = 100$. У чему се разликују? По чему су сличне? Да ли су обе приближно нормалне?

3. Биномна расподела је такође нагнута када p није 0.5. Урадити пример са $n = 100$ и $p = 0.25$, $p = 0.05$ и $p = 0.01$. Да ли су подаци приближно нормални у сваком од случајева? Искуствено правило гласи да ће расподела бити приближно нормална када је $np \geq 5$ и $n(1-p) \geq 5$. Да ли ово важи?



Слика 8.4: Симулација са експоненцијалним подацима. Нема савршен облик звона.

4. График нормалности је најбољи начин да се провери да ли је расподела приближно нормална. Мало примитивнији начин је да се провери да ли је 68% података унутар једне стандардне девијације, 95% унутар две стандардне девијације и 99.8% унутар три стандардне девијације. Креирати 100 случајних бројева X_i са нормалном расподелом медијане 0 и стандардне девијације 1. Који проценат података је у оквиру једне стандардне девијације у односу на медијану? А две стандардне девијације, три стандардне девијације? Да ли су подаци конзистентни са нормалном расподелом? Ево наговештаја:

```
> k = 1;sigma = 1
> n = length(x)
> sum( -k*sigma <x & x< k*sigma)/n
```

Ово се чита као „-1 мање од x и x мање од 1" и идентично је изразу $P(-1 < x \leq 1)$.

5. Интересантно је исцртавати расподелу стандардизованог просека како се n повећава. Урадити ово када X_i подлеже униформној расподели на интервалу $[0, 1]$. Размотрити хистограм када је n једнако 1, 5, 10 и 25. Да ли уочавамо облик нормалне расподеле? Правило је да ако X_i нису превише искошене, онда за $n \geq 25$ просек може бити приближно нормалан. Ево кода:

```
> f=function(n,a=0,b=1) { mu=(b+a)/2
sigma=(b-a)/sqrt(12)
(mean(runif(n,a,b))-mu)/(sigma/sqrt(n)) }
```

где су дате формуле за медијану и стандардну девијацију.

6. Постоји могућност да се сними видео ако се аутоматски приказује секвенца графова. Системска функција `System.sleep` може да направи паузу између кадрова. Код са следећег листинга ће приказати хистограм расподеле узорака за растуће n:

```
> for (n in 1:50) {
+ results = c()
+ mu = 10;sigma = mu
+ for(i in 1:200) {
+ X = rexp(200,1/mu)
+ results[i] = (mean(X)-mu)/(sigma/sqrt(n))
+ }

+ hist(results)
+ Sys.sleep(.1)
+ }
```

7. Направити нормалне графике за следеће случајне расподеле. Које од њих (ако их има) су приближно нормалне?

   (a) $rt(100, 4)$

   (b) $rt(100, 50)$

   (c) $rchisq(100, 4)$

   (d) $rchisq(100, 50)$

8. Техника `bootstrap` симулира на основу узимања узорака из података. На пример, следећа функција ће пронаћи медијану узорка `bootstrap`-а:

```
> bootstrap=function(data,n=length(data)) {
+ boot.sample=sample(data,n,replace=TRUE)
+ median(boot.sample)
+ }
```

   Нека подаци буду они из уграђеног скупа података `faithful`. Како изгледа `bootstrap` расподела за медијану? Да ли је нормална? Користити команду:

```
> simple.sim(100,bootstrap,faithful[['eruptions']])
```

   У зависности од типа података треба користити или аритметичку средину или медијану. Ево једног начина за упоређивање ова два броја када су подаци нормално дистрибуирани:

```
> res.median=c();res.mean=c() # иницијализација
> for(i in 1:200) { # креирање 200 случајних узорака
+ X = rnorm(200,0,1)
+ res.median[i] = median(X);res.mean[i] = mean(X)
+ }
> boxplot(res.mean,res.median) # упоређивање
```

Покренути овај код. Које су разлике? Покушати исти експеримент са дистрибуцијом са нетипичним вредностима као што је `X = rt(200,2)`. Постоји ли разлика? Објаснити.

9. У математичкој статистици постоји велики број могућих процена за центар скупа података. Да би се изабрао један од њих, често се узима најмања варијанса. Ова варијанса зависи од дистрибуције популације. Овде истражујемо коефицијент варијанси за аритметичку средину и медијану за различите расподеле. За нормалне (0,1) податке можемо проверити помоћу

```
> median.normal = function(n=100) median(rnorm(100,0,1))
> mean.normal = function(n=100) mean(rnorm(100,0,1))
> var(simple.sim(100,mean.normal)) /
+ var(simple.sim(100,median.normal))
[1] 0.8630587
```

Одговор је случајни број који ће обично бити мањи од јединице. Ово говори да је варијанса средине обично мања од варијансе медијане за нормалне податке. Поновити исто користећи експоненцијалну расподелу уместо нормалне. На пример:

```
> mean.exp = function(n=100) mean(rexp(n,1/10))
> median.exp = function(n=100) median(rexp(n,1/10))
```

и t-расподелу са 2 степена слободе:

```
> mean.t = function(n=100) mean(rt(n,2))
> median.t = function(n=100) median(rt(n,2))
```

Да ли је аритметичка средина увек боља од медијане? Такође, има смисла исцртати графике резултата команде `simple.sim` типа *један поред другог*.

Закључак

R је веома користан програмски језик за студенте на почетку студија, који још увек нису имали прилике да свеобухватно овладају неким програмским језиком, пре свега услед разумљиве синтаксе лагане за учење. Такође, од велике је користи за истраживаче који се баве различитим научним дисциплинама које користе статистику у истраживањима, а по вокацији нису софтверски инжењери. Његова примена је могућа у различитим областима где год је присутна статистика, као део биологије, медицине, економије, финансија и многих других наука.

У овом материјалу описан је рад са различитим врстама података у R-у, акценат је стављен на израду дијаграма који најсликовитије приказују податке и њихове међусобне односе. Следећи корак би требало да буде тестирање унетих података у складу са статистичким тестовима, као и анализа унетих података и коришћење екстерних пакета.

Литература

[1] John Verzani, Using R for Introductory Statistics, 2004.

[2] др сц. Андреја Радовић, Упознавање са синтаксом језика R и његова примјена у основној статистичкој и графичкој анализи података, 2015.

[3] M. Hutzenthaler, R course, 2011.

[4] http://www.e-statistika.rs/Article/Display/deskriptivna-statistika, Октобар 2018.

[5] http://www.ef.uns.ac.rs/Download/statistika/2016-11-15-Deskriptivna-statistika.pdf, Октобар 2018.