

Praktiskā biometrija
Piemēri darbā ar programmu R

Didzis Elferts
©*Versija 0.4 2013. gada 26. aprīlī*

2013

Saturs

Ievads	7
1 Ievads darbā ar programmu R	8
1.1 Kas ir programma R un kā uzsākt darbu?	8
1.1.1 R vide	8
1.1.2 R instalācija un papildus paketes	8
1.1.3 Palīdzības iegūšana	9
1.2 Darbs ar R	9
1.2.1 Komandu veidošana	9
1.2.2 Kalkulators	9
1.2.3 Datu veidi	10
1.2.4 Datu struktūras	10
1.2.5 Datu importēšana	13
1.2.6 Datu eksportēšana	14
1.2.7 Darbs ar datiem	15
1.2.8 Darba vides sakārtošana	19
1.2.9 Darba sesijas saglabāšana	19
2 Grafiskās iespējas programmā R	20
2.1 Pamatgrafiki	20
2.2 Dati grafiku veidošanai	22
2.3 Grafiku parametru noteikšana	23
2.3.1 Grafiku izvietoējums lapā	23
2.3.2 plot() grafiku tipi	25
2.3.3 Līniju parametri	26
2.3.4 Simbolu veidi	28
2.3.5 Virsraksti un teksti pie asīm	29
2.3.6 Teksts grafika iekšienē	30
2.3.7 Tekstu parametri	31
2.3.8 Asis	32
2.3.9 Divas y asis	33
2.3.10 Krāsas	34
2.3.11 Lēģenda	34
2.3.12 Līnijas grafikos	35
2.4 Grafiku saglabāšana	36

3	Datu izzināšana un normalitātes testi	37
3.1	Datu izzināšana	37
3.2	Normalitātes testi	40
3.2.1	Grafiskā analīze	40
3.2.2	Analītiskā analīze	41
4	Statistiskie rādītāji un ticamības intervāli	43
4.1	Statistiskie rādītāji	43
4.2	Ticamības intervāli	46
4.2.1	Bootstrap ticamības intervāli	47
4.2.2	Ticamības intervālu grafiskais attēlojums	49
5	Paraugkopu salīdzināšana	51
5.1	Datu sagatavošana	51
5.2	Divu neatkarīgu paraugkopu salīdzināšana ar parametriskajām metodēm	52
5.3	Vairāku neatkarīgu paraugkopu salīdzināšana ar parametriskajām metodēm	53
5.4	Atkarīgu paraugkopu salīdzināšana ar parametriskām metodēm	54
5.5	Vienas paraugkopas t-tests	56
5.6	Divu neatkarīgu paraugkopu salīdzināšana ar neparametriskajām metodēm	56
5.7	Divu atkarīgu paraugkopu salīdzināšana ar neparametriskajām metodēm	57
5.8	χ^2 tests	57
5.9	Paraugkopu grafiskā salīdzināšana	58
5.10	Permutāciju jeb randomizēšanas tests paraugkopu salīdzināšanai	59
6	Dispersijas analīze	62
6.1	Dati	62
6.2	Dispersiju homogenitāte	63
6.3	Modeļu definēšana	63
6.4	Gradācijas klašu salīdzināšana	64
6.5	Rezultātu grafiskā attēlošana	65
7	Korelācijas analīze	67
7.1	Grafiskā pārbaude	67
7.2	Pīrsona korelācijas analīze	69
7.3	Rangu korelācijas analīze	71
7.4	Autokorelācija	71
8	Regresijas analīze	74
8.1	Dati	74
8.2	Pāru regresija	75
8.3	Regresijas analīzes pieņēmumu pārbaude	76
8.4	Vērtību prognozēšana	78
8.5	Daudzfaktoru regresija	79
9	Kovariācijas analīze (ANCOVA)	84
9.1	Dati	84
9.2	Dispersijas analīze	84
9.3	Kvantitatīva mainīgā iekļaušana modelī	85

10	Vispārinātie lineārie modeļi (GLM)	90
10.1	Dati	90
10.2	Puasona regresija	92
10.3	Binārā loģistiskā regresija	95

Tabulu saraksts

5.1	Koku skaits pa sugām divos parauglaukumos	58
-----	---	----

Attēlu saraksts

2.1	Izkliedes grafika piemērs	21
2.2	Box-plot grafika piemērs	21
2.3	Histogrammas piemērs	22
2.4	Slokšņu/stabiņu grafika piemērs	22
2.5	Grafiks ar diviem mainīgajiem	23
2.6	Vairāku grafiku izvietojums vienā lapā	24
2.7	Vairāku grafiku izvietojums vienā lapā	25
2.8	Funkcijas plot() grafiku veidi	26
2.9	Iespējamie līniju tipi grafikos	27
2.10	Līnijas izmaiņas, mainot dažādus parametrus	28
2.11	Simbolu veidi	29
2.12	Grafiks, kuram teksti definēti kā argumenti	30
2.13	Grafiks, kuram teksti definēti kā atsevišķas funkcijas	30
2.14	Tekstu novietojuma piemērs	31
2.15	Teksta parametru izmaiņas	31
2.16	Asu garuma izmaiņu piemērs	32
2.17	Asu definēšanas piemērs	33
2.18	Grafiks ar divām y asīm	34
2.19	Krāsu izmantošanas piemērs	34
2.20	Grafiks ar leģendu	35
2.21	Grafiks ar papildus līnijām	35
3.1	Viena mainīgā izkliedes grafiks	38
3.2	Divu mainīgo izkliedes grafiks	38
3.3	Box-plot grafiks vienam mainīgajam	38
3.4	Box-plot grafiks trīs paraugkopām	38
3.5	Punktu grafiks	39
3.6	Modificēts punktu grafiks	39
3.7	Pazīmes niedru lapu garums histogramma	40
3.8	QQ grafiks niedru lapu garumam	41
3.9	Modificēts QQ grafiks niedru lapu garumam	41
4.1	Bootstrap vidējo aritmētisko histogramma un QQ grafiks	48
4.2	Vidējais niedru lapu garums trīs parauglaukumos ar 99% ticamības intervāliem	49
5.1	Niedru lapu platuma box-plot grafiks trīs parauglaukumiem	59
5.2	Permutāciju testa rezultātā iegūto vidējo aritmētisko starpības histogramma	59
6.1	Gradācijas klašu box-plot attēls	65
6.2	Gradācijas klašu vidējo aritmētisko attēls	65

7.1	Izkliedes grafiki starp visām smiltāju datu tabulas kolonnām	68
7.2	QQ grafiki smiltāju tabulas mainīgajiem	69
7.3	Priežu gadskārtu platumu aukorelācijas attēls	72
8.1	Izkliedes grafiks starp pazīmēm gaisma un biomasa ar pievienotu regresijas līkni	76
8.2	Regresijas analīzes diagnosticējošie grafiki	77
8.3	Izkliedes grafiki starp regresijas modeļa atlikuma vērtībām un modelī neiekļautajiem mainīgajiem	78
8.4	Izkliedes grafiki starp visiem datu objekta augi2 mainīgajiem	79
8.5	Regresijas analīzes diagnosticējošie grafiki	83
9.1	Box-plot grafiks svara samazinājumam atkarībā no diētas veida	86
9.2	Izkliedes grafiks ar trenda līnijām starp sākotnējo svaru un svara samazinājumu atkarībā no diētas veida	88

Ievads

Šī grāmata ir mans mēģinājums sniegt praktiskus padomus statistisko analīžu veikšanā bioloģiem. Tā kā uzsvars ir likts uz vārdu "praktiski", tad lielāko grāmatas daļu sastāda piemēri tam, kā veikt katru no apskatītajiem statistiskajiem testiem.

Teorētiskā pamatojuma iegūšanai noderēs citu autoru darbi. Nenoliedzami nopietnākais darbs latviešu valodā biometrijas jomā ir jāmin I.Liepas (1974) grāmata, angļu valodā tas būtu kāds no Sokala un Rohlfa izdevumiem (piemēram, Sokal & Rohlf, 1995). Vieglākā formā nelielu teorētisko pamatojumu statistikas pamatmetodēm latviešu valodā var meklēt Arhipova & Bāliņa (2006), tikai tajā piemēri būs no ekonomikas jomas. Par modernākām metodēm informāciju var meklēt, piemēram, Zuur et al. (2007).

Programma R ir izvēlēta kā rīks, ar kuru veikt praktisko datu analīzi. Kāpēc tieši programma R? Pirmais arguments ir tas, ka šī programma ir bezmaksas, attiecīgi ir pieejama jebkuram lietotājam, kas izvēlēties izmantot šo grāmatu. Otrkārt, pamatdarbības programmā R notiek ar komandu rindām, kas lietotājam liek aizdomāties par darbībām, kuras viņš taisās veikt, nevis vienkārši kaut ko ņemt un klikšķināt. Trešais iemesls būtu tas, ka R šobrīd ir straujāk attīstošās statistikas programma, kas ļauj tajā veikt arī pašas jaunākās analīzes.

Dotais materiāls ir mans pirmais mēģinājums uzrakstīt grāmatu par R, tāpēc tajā ļoti daudz vēl pilnveidojamu lietu. Kā alternatīvus materiālus statistisko analīžu veikšanai programmā R var izmantot, piemēram, Verzani (2005), Everitt & Hothorn (2006) vai Maindonald & Braun (2010).

Nodaļa 1

Ievads darbā ar programmu R

1.1 Kas ir programma R un kā uzsākt darbu?

1.1.1 R vide

R ir vienlaicīgi datorprogramma un valoda (veidojusies no programmēšanas valodas 'S'), kas paredzēta datu apstrādei, aprēķiniem un grafiku veidošanai. Visi piemēri šajā grāmatā ir veidoti programmas R versijā 2.15.1 (R Core Team, 2012).

R priekšrocības ir:

- tā ir atvērta koda programma, kas nepārtraukti tiek papildināta un uzlabota, un šajā darbā ir iesaistīti tūkstošiem cilvēku visā pasaulē;
- R ir izmantojams uz dažādām datoru platformām: Unix, Linux, Windows, MacOS;
- lietotājam ir dota iespēja kontrolēt visus parametrus veicot dažādus aprēķinus;
- R ir ļoti labas grafiskās iespējas, kas ļauj veidot augstas kvalitātes grafikus.

1.1.2 R instalācija un papildus paketes

Lai instalētu programmu R, ir nepieciešams lejupielādēt Jūsu izmantotajai platformai atbilstošu R instalācijas failu no CRAN servera. Serveru adreses, kā arī citu informāciju var atrast mājas lapā <http://www.R-project.org>

Pirmkārt, ir jāuzinstalē pamatfails R-base. Programma R ir veidota tā, ka R-base satur tikai daļu no nepieciešamajām funkcijām, pārējās ir pieejamas papildus paketēs. Tās var uzinstalēt trīs veidos:

- izmantojot funkciju `install.packages()` un norādot instalējamās paketes nosaukumu.
- programmā R komandu rindā izvēlas *Packages/Install package(s)...* Tad izvēlas tuvāko serveri un paketes, kuras nepieciešams instalēt.
- ja paketes pirms tam tika lejupielādētas datorā, tad izvēlas opciju *Packages/Install package(s) from local zip files...*

Ja kāda no paketēm ir nepieciešama konkrētajā darba sesijā, tad pirms lietošanas tā ir „jāpievieno” ar funkciju `library()`, kur iekavās norādīts nepieciešamās paketes nosaukums, piemēram:

```
>library(grid)
```

1.1.3 Palīdzības iegūšana

Papildus informāciju un palīdzību darbam ar R var iegūt no dažādām pamācībām, kas brīvi pieejamas internetā. Par programmu R ir arī ļoti daudz grāmatu, no kurām nopietnākais darbs ir Crawley (2007). Ir pieejama arī R iekšējā palīdzībā. Ja nepieciešams iegūt palīdzību par kādu konkrētu funkciju, vieglākais veids ir izmantot `help()` funkciju, kurai iekavās norādīts otras funkcijas nosaukums, piemēram:

```
>help(plot)
```

Otrs variants ir rakstīt `? zīmi` pirms funkcijas nosaukuma. Jāņem vērā, ka šādā veidā ir iespējams meklēt palīdzību tikai par tām funkcijām, kuras atrodas konkrētajai darba sesijai pievienotajās paketēs. Ja nepieciešams meklēt funkciju visās instalētajās paketēs, tad jāizmanto divas `? zīmes`.

```
>??plot
```

1.2 Darbs ar R

1.2.1 Komandu veidošana

R vidē visas komandas tiek rakstītas pēc `>` zīmes (tā nav pašam jāraksta). Ja komandu rinda ir pārāk gara, tad pēc jebkura argumenta, vai arī pēc iekavām, komata, utt var spiest Enter taustiņu un turpināt pierakstu jaunā rindā. Šajā gadījumā automātiski parādīsies `+` zīme. Ja komanda tiek uzrakstīta pilnībā un nospiests Enter, šī komanda tiek izpildīta. Ja komanda neizpildās, bet parādās `+` zīme, tas nozīmē, ka komandas pierakstā ir kļūda un trūkst, piemēram, kādas iekavas.

Komandu rindas galā var pievienot arī komentāru, pirms komentāra ir jāraksta `#` zīme.

R vidē atstarpes starp dažādiem objektiem, iekavām tiek ignorētas, izņēmums ir kombinācija `<-`, ar kuru objektiem piešķir noteiktu vērtību.

Rakstot komandas ir būtiski ievērot lielo un mazo burtu izvietojumu, jo šajā gadījumā apzīmējums `AA` nebūs tas pats, kas `aa`.

1.2.2 Kalkulators

Programmu R savā veidā var izmantot kā kalkulatoru, ar kura palīdzību ir iespējams veikt dažādus aprēķinus.

```
> 4+7

[1] 11

> log(8,2)

[1] 3

> exp(2)

[1] 7.389056
```

1.2.3 Datu veidi

R vidē pārsvārā lieto trīs veidu datus: skaitliskos (numeric), rakstu zīmes (character) un loģiskos (logical).

Skaitliskie dati ir skaitļi, un tikai šāda veida objektus var izmantot, lai veiktu aprēķinus.

Rakstu zīmju datus parasti izmanto, lai norādītu mainīgajiem nosaukumus, kā arī, lai veiktu datu grupēšanu.

Loģiskajiem datiem var būt divas vērtības TRUE vai FALSE (jeb T un F).

1.2.4 Datu struktūras

Četras galvenās datu struktūras R vidē: vektors (vector), matrice (matrix), saraksts (list) un datu tabula (data frame).

Vektors

Vektors ir viendimensionāla datu struktūra, kas var sastāvēt tikai no viena tipa datiem – skaitļiem vai rakstu zīmēm. Vektora garums nav ierobežots – tas var būt tikai vienu zīmi liels, vai arī sastāv no ļoti daudz zīmēm vai skaitļiem.

Lai nodefinētu vektoru, kas sastāv tikai no vairāk kā viena mainīgā, jāraksta vēlamais objekta (vektora) nosaukums un aiz apzīmējuma “<-” jāraksta funkcija `c()`, kurai iekavās norāda mainīgos:

Rezultātā izveidotais objekts tiek saglabāts R atmiņā. Lai apskatītu izveidoto objektu, ir jāuzraksta tā nosaukums un jānospiež Enter.

```
> pirmais<-c(1,5,7,4,8,10)
> pirmais

[1] 1 5 7 4 8 10

> otrais<-c("A","B","C","D","E","F")
> otrais

[1] "A" "B" "C" "D" "E" "F"
```

Matrice

Matrice ir divdimensionāla datu struktūra, kas līdzīgi vektoram var saturēt tikai viena tipa datus. Matrici var izveidot izmantojot trīs funkcijas: `cbind()`, `rbind()` vai `matrix()`.

Funkcijā `matrix()` norāda visus matricē iekļaujamos elementus, kā arī norāda rindu skaitu (`nrow`) vai arī kolonnu skaitu (`ncol`); abus nevajag norādīt, jo, ja ir norādīts, piemēram, kolonnu skaits, tad rindu skaits tiks aprēķināts automātiski, ņemot vērā elementu skaitu matricē:

```
> tresais<-matrix(1:15,ncol=3)
> tresais
```

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

Funkciju `cbind()` izmanto, lai izveidotu matrici, tās elementus secīgi rakstot kolonnās (iekavās aiz funkcijas jānorāda elementi, kas atradīsies katrā no kolonnām), attiecīgi funkciju `rbind()` izmanto matricēs veidošanai pa rindiņām.

```
> ceturtais<-cbind(pirmais,otrais)
> ceturtais
```

```
      pirmais otrais
[1,] "1"      "A"
[2,] "5"      "B"
[3,] "7"      "C"
[4,] "4"      "D"
[5,] "8"      "E"
[6,] "10"     "F"
```

Saraksts

Saraksts ir datu struktūra, kas var saturēt jebkuru citu datu struktūru, tai skaitā arī apakšsarakstus. Saraksts ir ļoti ērts, lai vienā objektā apvienotu dimensionāli atšķirīgus elementus, piemēram, no vienas analīzes rezultātiem. Sarakstu veido ar funkciju `list()`.

```
> piektais<-list(vekt1=pirmais,matr1=tresais,matr2=ceturtais)
> piektais
```

```
$vekt1
[1] 1 5 7 4 8 10
```

```
$matr1
  [,1] [,2] [,3]
[1,]  1   6  11
[2,]  2   7  12
[3,]  3   8  13
[4,]  4   9  14
[5,]  5  10  15
```

```
$matr2
      pirmais otrais
[1,] "1"      "A"
[2,] "5"      "B"
[3,] "7"      "C"
[4,] "4"      "D"
[5,] "8"      "E"
[6,] "10"     "F"
```

Datu tabula

Datu tabula ir divdimensionāla datu struktūra, kas var saturēt dažāda tipa datus atsevišķās kolonnās (visām kolonnām vienā datu tabulā ir jābūt ar vienādu garumu). Datu tabulu izveido ar funkciju `data.frame()`. Funkcijas iekavās norāda kolonnu nosaukumus, kā arī datus, kas būs katrā no kolonnām. Dati var būt kā atsevišķi vektori, kas izveidoti jau iepriekš, vai arī datus var ierakstīt pašā funkcijā:

```
> sestais<-data.frame(pirmais,otrais)
> sestais
```

```
      pirmais otrais
1          1      A
2          5      B
3          7      C
4          4      D
5          8      E
6         10      F
```

```
> septitais<-data.frame(kol1=c(1,2,3,4),kol2=c(5,6,7,8))
> septitais
```

```
      kol1 kol2
1         1    5
2         2    6
```

```
3    3    7
4    4    8
```

Vēl datu tabulu var izveidot izmantojot datu tabulu veidotāju pašā programmā R. Šajā gadījumā no sākuma ar kādu Jums vēlamu nosaukumu ir jāizveido datu tabula ar funkciju `data.frame()`, kurai netiek norādīti nekādi papildus argumenti. Pēc tam jāizmanto funkcija `fix()`, kurai kā arguments jānorāda jaunās datu tabulas nosaukums. Rezultātā parādīsies datu tabulu veidošanas logs, kurā var ievadīt datus. Lai mainītu kolonnu nosaukumus, ir jāuzklikšķina uz tā – šajā brīdī ir iespējams arī norādīt vai dati būs skaitliski vai rakstu zīmes. Ir jāatceras, ka vienā kolonnā var atrasties tikai viena veida dati, kā arī, ka visu kolonnu garumiem ir jābūt vienādiem.

```
> jauna.tabula<-data.frame()
> fix(jauna.tabula)
```

Ar funkcijas `fix()` palīdzību ir iespējams labot arī citas jau esošās tabulas.

1.2.5 Datu importēšana

R vidē ir iespējams importēt dažāda formāta datu failus (datu tabulas), piemēram, no programmām Excell, SPSS, SAS. Vienkāršākais veids ir izmantot `.txt` vai `.csv` failus. Lai importētu `.txt` failus, tiek izmantota funkcija `read.table()`, attiecīgi `.csv` failus var importēt ar funkcijām `read.csv()` un `read.csv2()`. Funkcijā norāda faila nosaukumu un ceļu uz to (`file`). Ja pirmā tabulas rinda satur kolonnu nosaukumus, tad kā papildus arguments jānorāda `header=TRUE`. Vēl jānorāda skaitļu decimāldaļu atdalītājs (komats vai punkts), izmantojot papildus argumentu `dec="."` Tā kā `.txt` failiem mēdz būt dažādi kolonnu atdalīšanas veidi, tad jānorāda atbilstošais, izmantojot papildus argumentu `sep=" "`, iespējamās vērtības ir `"\t"` – tab delimited; `","` – kolonnu atdalītājs ir komats; `";"` – kolonnu atdalītājs ir semikols. Pilno ceļu uz importējamo failu var nenorādīt, ja fails atrodas darba direktoriā (Working directory) (var nomainīt ar `File/Change dir...`) (Windows vidē) vai funkciju `setwd()`.

```
> dati<-read.table(file="niedres.txt",header=TRUE,sep="\t",dec=".")
> dati2<-read.csv(file="niedres.txt",header=TRUE,sep="\t",dec=".")
> dati3<-read.csv2(file="niedres.txt",header=TRUE,sep="\t",dec=".")
```

```
> dati
      garums platums
1      31.6      2.5
2      23.2      2.3
3      39.2      2.1
```

```
.....
49  38.8    3.9
50  46.9    5.4
```

Pašā programmā R ir iekļautas daudzas datu tabulas, kas ir ļoti noderīgas, piemēram, lai izmēģinātu kādas funkcijas, vai arī sekotu līdzī piemēriem mācību materiāliem.

Lai redzētu, kuras datu tabulas ir pieejamas konkrētajā darba sesijā (atkarīgs no pievienoto pakešu daudzuma), jāizmanto funkcija `data()`, pēc kā parādīsies jauns logs, kurā uzskaitītas pieejamās datu tabulas un to apraksts. Lai redzētu visas pieejamās datu tabulas, jāraksta `data(package = .packages(all.available = TRUE))`.

```
> data()
> data(package = .packages(all.available = TRUE))
```

Lai pievienotu darba sesijai kādu no šīm datu tabulām, jāizmanto funkcija `data()`, kurai iekavās norādīts datu tabulas nosaukums. Ja pakete, kurā atrodas šī datu tabula jau ir pievienota darba sesijai, tad papildus argumenti nav vajadzīgi, bet ja pakete nav pievienota, tad funkcijai ir jāliek klāt papildus arguments `package= ```. Pēc tam datu tabula ir pievienota darba sesijai ar tādu nosaukumu, kāds ir pašai datu tabulai.

```
> data(cars)
> data(cars, package = "datasets")
> cars
```

```
      speed dist
1         4    2
2         4   10
3         7    4
4         7   22
5         8   16
6         9   10
7        10   18
...     ...   ...
```

1.2.6 Datu eksportēšana

Datu eksportēšanu veic ar funkciju `write.table()`, kurā jānorāda objekts, kuru vēlas eksportēt (`x=`), izveidojamā faila nosaukums un saite uz to (`file=" "`), kolonnu atdalītājs (`sep=" "`), decimālatdalītājs (`dec=" "`) un vai rindu nosaukumus iekļaut (`row.names=TRUE` vai `FALSE`).

```
> write.table(x=dati, file="eksports.txt", row.names=FALSE, sep="\t", dec=".")
```

1.2.7 Darbs ar datiem

Ar funkcijas `str()` palīdzību ir iespējams apskatīt jebkura datu objekta struktūru, tajā esošos datus un to veidu. Rezultāts ir atkarīgs no tā, kāda veida datu objekts tiek apskatīts.

```
> str(dati)

'data.frame':      50 obs. of  2 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num  2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...

> str(sestais)

'data.frame':      6 obs. of  2 variables:
 $ pirmais: num  1 5 7 4 8 10
 $ otrais : Factor w/ 6 levels "A","B","C","D",...: 1 2 3 4 5 6

> str(piektais)

List of 3
 $ vekt1: num [1:6] 1 5 7 4 8 10
 $ matr1: int [1:5, 1:3] 1 2 3 4 5 6 7 8 9 10 ...
 $ matr2: chr [1:6, 1:2] "1" "5" "7" "4" ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:2] "pirmais" "otrais"

> str(tresais)

int [1:5, 1:3] 1 2 3 4 5 6 7 8 9 10 ...

> str(otrais)

chr [1:6] "A" "B" "C" "D" "E" "F"
```

Datu tabulu un matricu apskatīšanai noderīgas ir funkcijas `head()` un `tail()`, kas attiecīgi parāda datu objekta pirmās sešas un pēdējās sešas rindīņas. Rindīņu skaitu ir iespējams mainīt, norādot papildus argumentu `n=`.

```
> head(dati)

  garums platums
1  31.6     2.5
2  23.2     2.3
3  39.2     2.1
4  37.4     5.8
5  21.1     2.2
6  37.0     4.1
```



```
> tail(dati)

      garums platums
45  46.1     5.3
46  29.1     3.5
47  33.7     4.6
48  43.1     3.7
49  38.8     3.9
50  46.9     5.4
```

Daļas no objekta atlasīšana

Ja ir nepieciešams atlasīt tikai daļu no objekta, veids kā to darīt ir atkarīgs no objekta veida. Tā kā vektors ir viendimensionāla datu struktūra, tad vien elementa atlasīšanai aiz objekta nosaukuma ir jāliek kvadrātiskās iekavas, kurās jānorāda atlasāma elementa kārtas numurs.

```
> otrais[1]

[1] "A"
```

Vairāku vektora elementu atlasīšanai kvadrātiskajās iekavās jāizmanto funkcija `c()`, kurā norāda atlasāmo elementu kārtas numurus.

```
> otrais[c(1,4)]

[1] "A" "D"
```

Ja ir nepieciešams atlasīt visus elementus, izņemot kādu konkrētu elementu, tad "nevēlamā" elementa kārtas numuru kvadrātiskajās iekavās norāda ar mīnus zīmi.

```
> otrais[-3]

[1] "A" "B" "D" "E" "F"
```

No matricas, kas ir divdimensionāla datu struktūra, konkrētu elementu var atlasīt kvadrātiskajās iekavās norādot divus skaitļus, kur pirmais nozīmē rindas numuru, bet otrais nozīmē kolonnas numuru.

```
> tresais[2,2]

[1] 7
```

Ja kvadrātiskajās iekavās norāda tikai vienu skaitli, bet otra vietu atstāj tukšu, tad attiecīgi tiek atlasīta visa rindiņa, vai arī visa kolonna.

```
> tresais[1,]
```

```
[1] 1 6 11
```

```
> tresais[,2]
```

```
[1] 6 7 8 9 10
```

Elementus no saraksta var atlasīt vairākos veidos, piemēram, rakstot objekta nosaukumu, tad \$ zīmi un saraksta elementa nosaukumu. Otrs variants ir izmantot dubultās kvadrātiskās iekavas un norādīt saraksta elementa kārtas numuru.

```
> piektais$vekt1
```

```
[1] 1 5 7 4 8 10
```

```
> piektais[[2]]
```

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

Ar iepriekšējām divām metodēm no saraksta ir iespējams atlasīt tikai visu elementu. Ja ir nepieciešams atlasīt kaut ko no paša saraksta elementa, tad jāizmanto otras kvadrātiskās iekavas.

```
> piektais[[2]][1,]
```

```
[1] 1 6 11
```

No datu tabulas visu kolonnu var atlasīt tās nosaukumu norādot ar \$ zīmes vai arī pēdējās ierakstot kvadrātiskajās iekavās.

```
> sestais$pirmais
```

```
[1] 1 5 7 4 8 10
```

```
> sestais["pirmais"]
```

```
  pirmais
1         1
2         5
3         7
4         4
5         8
6        10
```

Viena elementa atlasīšanai ir jāizmanto tāds pats pieraksts kā matricai – kvadrātiskajās iekavās jānorāda divi skaitļi, kas apzīmē rindiņas un kolonnas numuru.

```
> sestais[5,1]
```

```
[1] 8
```

Viena no programmas R specifikām ir tā, ka datu tabulām esošos kolonnu nosaukumus nav iespējams uzreiz izmantot kā mainīgos. Piemēram, ja uzrakstīsim mainīgo *garums*, kas atrodas pievienotajā datu tabulā *dati*, parādīsies paziņojums, ka tāda mainīgā nav. R ,veidojot datus tabulas un importējot failus ,katru no kolonnām neuztver kā atsevišķu objektu.

```
> garums
```

```
Error: object 'garums' not found
```

Lai datu tabulas katra kolonna tiktu uzskatīta par atsevišķu objektu, jāizmanto funkcija `attach()`. Tomēr ar šo funkciju jābūt ļoti uzmanīgiem, jo, ja datu tabulā kādai kolonnai būs tāds pats nosaukums kā esošam objektam, tas tiks aizstāts.

Pēc tabulas pievienošanas katru kolonnu var izmantot kā atsevišķu mainīgo.

```
> attach(dati)
```

```
> garums
```

```
[1] 31.6 23.2 39.2 37.4 21.1 37.0 24.7 31.3 37.4 39.7 38.0 24.9 30.8 26.7 ...
[22] 46.2 11.9 39.5 45.3 39.6 45.8 31.0 39.9 56.4 37.3 41.1 45.3 49.3 24.9 ...
[43] 26.9 35.7 46.1 29.1 33.7 43.1 38.8 46.9
```

Ja ir nepieciešams datu tabulu "atvienot" no darba sesijas, var izmantot funkciju `detach()`.

```
> detach(dati)
```

1.2.8 Darba vides sakārtošana

Ja darba sesijā rada dažādus objektus, pievieno daudz paketes, ar laiku var rasties apjukums par to, kādi objekti šobrīd ir aktīvi. Ar funkcijas `ls()` palīdzību, ir iespējams iegūt sarakstu ar visiem objektiem, kas šobrīd ir aktīvi:

```
> ls()

[1] "ceturtais" "dati"      "dati2"     "dati3"     "otrais"    "piektais"
[7] "pirmais"   "septitais" "sestais"   "tresais"
```

Lai kādu no šiem objektiem noņemtu no darba sesijas, jāizmanto funkcija `rm()`, kurai kā arguments jānorāda objekts, kuru vēlaties noņemt no darba sesijas. Šī metode gan attiecas tikai uz tiem objektiem, kurus esat radījuši paši, nevis uz datu tabulām, kas atrodas paketēs.

```
> rm(pirmais)
```

1.2.9 Darba sesijas saglabāšana

Beidzot darba sesiju R vidē, ir iespējams saglabāt dažādus sesijas elementus. Ar komandu *File/Save History...* ir iespējams saglabāt visas dotajā sesijā izmantotās funkcijas (komandu rindas) (fails `.Rhistory`). Nākamajā sesijā atverot `.Rhistory` failu (*File/Load History...*) būs pieejamas visas izmantotās komandu rindas (tās var izsaukt ar augšup vai lejup vērstajām bultiņām).

Lai saglabātu visus dotās sekcijas objektus, jāizmanto komanda *File/Save Workspace...* (faila paplašinājums `.RData`). Nākamajā sesijā atverot `.RData` failu (*File/Load Workspace...*), būs pieejami visi iepriekšējā sesijā definētie objekti.

Visas sesijā izmantotās funkcijas (komandas) un iegūtos rezultātus var saglabāt arī atsevišķā `.txt` failā (*File/Save to File...*), kas varētu būt noderīgi, lai izmantotu iegūtos rezultātus vai komandu rindas citās programmās (piemēram, veidojot protokolus).

Nodaļa 2

Grafiskās iespējas programmā R

Programmā R ir iespējams veidot ļoti augstas kvalitātes grafikus, kuriem ir iespējams definēt jebkuru no interesējošiem parametriem. Grafiku veidošanai ir pieejamas vairākas grafiskās sistēmas, no kurām populārās trīs ir:

- Tradicionālie grafiki (Murrell, 2006)
- Trellis jeb Lattice grafiki (Sarkar, 2008)
- ggplot2 grafiki (Wicham, 2009)

Šajā grāmatā apskatīti tikai tradicionālo grafiku veidošana.

2.1 Pamatgrafiki

Grafikus līdzīgi kā jebkuras citas darbības programmā R veic ar komandu rindām. Ir pieejamas funkcijas, kas izveido jau nodefinētu grafika veidu, vai arī var izveidot grafiku izmantojot atsevišķas funkcijas dažādu parametru definēšanai.

R specifika ir tā, ka, izveidojot grafiku, parādās grafiskais logs, kurā ir attēlots konkrētais grafiks. Ja tiek izsaukta jauna funkcija, kas arī veido grafiku, iepriekšējais grafiks tiek aizstāts. Ja ir nepieciešams, ka uz ekrāna vienlaicīgi ir redzami vairāki grafiki, tad pirms katra jauna grafika ir jāraksta funkcija `windows()` (Windows vidē) vai `quartz()` (MacOS vidē).

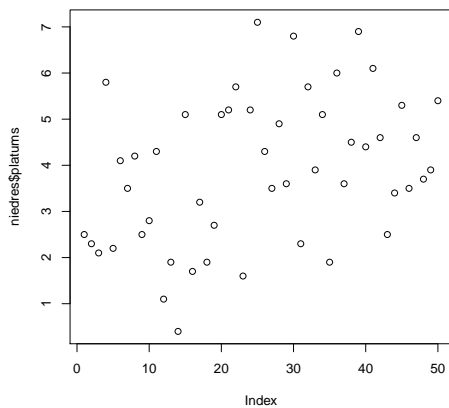
Lai veidotu grafiku, ir nepieciešami dati. Šajā nodaļā kā piemērs izmantots datu fails *niedres.txt*, kas satur informāciju par 50 niedru lapu garumu un platumu. Ar funkciju `read.table()` fails tiek importēts, pēc tam funkciju `str()` apskatīta tā struktūra.

```
> niedres<-read.table(file="niedres.txt",header=T,sep="\t",dec=".")
> str(niedres)

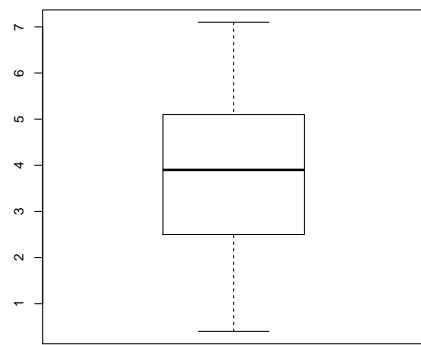
'data.frame':      50 obs. of  2 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num  2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
```

plot()

Visbiežāk pielietotā funkcija grafiku veidošanai ir `plot()`. Ja šajā funkcijā norāda vienu vai divas skaitļu rindas, tad tā veido izkliedes grafiku (scatterplot) (2.1 attēls). Šai funkcijai ir



Att. 2.1: Izkliedes grafika piemērs



Att. 2.2: Box-plot grafika piemērs

iespējams norādīt ļoti daudz papildus argumentus, kas maina izveidoto grafiku. Turklāt lietot šajā funkcijā atšķirīgus datus objektu, var iegūt ļoti atšķirīgu rezultātu.

```
> plot(niedres$platums)
```

boxplot()

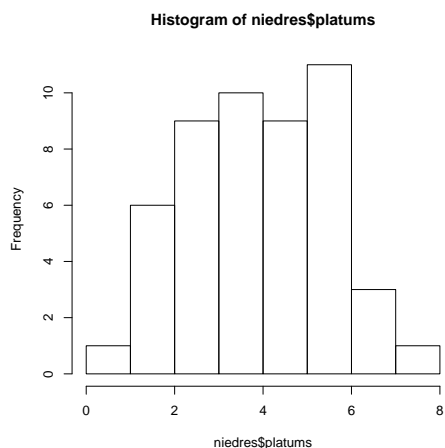
Funkcija `boxplot()` izveido Box-plot grafiku (2.2 attēls). Šai funkcijai var norādīt vienu datu rindu, kas satur skaitļus, vai arī divas datu rindas, kur vienā ir skaiļi, bet otrā ir apzīmējumi daļījuma līmeņiem. Otrajā gadījumā attsevišķs Box-plot grafiks tiks izveidots katram no daļījuma līmeņiem.

```
> boxplot(niedres$platums)
```

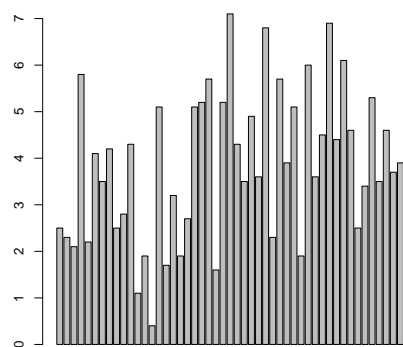
hist()

Ar funkcijas `hist()` palīdzību ir iespējams izveidot histogrammu datiem, kas vēl nav sagrupēti klasēs (izejas datiem) (2.3 attēls). Šajā funkcijā ir jānorādi tikai viena datu rinda, kuru ir nepieciešams attēlot. Ar papildus argumentiem ir iespējams mainīt to, cik daļās dati tiek dalīti.

```
> hist(niedres$platums)
```



Att. 2.3: Histogrammas piemērs



Att. 2.4: Sloksņu/stabiņu grafika piemērs

barplot()

Funkcija `barplot()` veido sloksņu jeb stabiņu grafiku (2.4 attēls). Šī funkcija ir īpaši noderīga gadījumā, ja dati jau ir apkopoti pa klasēm, jo funkcija attēlo datus tādā veidā kā tie norādīti mainīgajā.

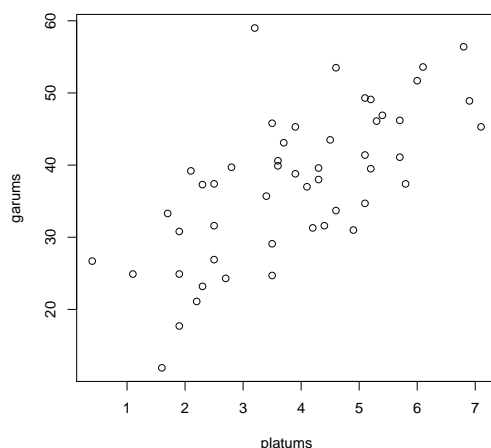
```
> barplot(niedres$platums)
```

2.2 Dati grafiku veidošanai

Lai veidotu grafikus, esošos datus no gatavām datu tabulām nav jā saglabā kā atsevišķus mainīgos – tos uzreiz var izmantot grafiku veidošanai. Ir vismaz trīs dažādi veidi kā norādīt attēlojamās mainīgos: (a) norādot pilnu datu objektu nosaukumu un tad kolonnas nosaukumu, kuru vēlas attēlot; (b) grafika veidošanas funkciju likt kā argumentu funkcijai `with()`; (c) izmantot funkciju `attach()`, lai pievienotu tabulu darba sesijai un turpmāk rakstīt tikai kolonnas nosaukumu (šis nav labākais variants). Visos šajos gadījumos izveidosies līdzīgs attēls (2.1 attēls), var atšķirties tikai paraksti zem asīm.

```
> plot(niedres$platums)
> with(niedres,plot(platums))
> attach(niedres)
> plot(platums)
```

Ja ir nepieciešams uz x un y ass attēlot divu mainīgo datus, tad ir divi iespējamie risinājumi: (a) kā pirmo argumentu raksta uz x ass attēlojamais mainīgo, tad komats un uz y ass attēlojamais mainīgais; (b) kā pirmo raksta uz y ass attēlojamais mainīgo, tad tildes zīme (`~`) un uz x ass attēlojamais mainīgais. Abi varianti izveido identisku grafiku (2.5 attēls)



Att. 2.5: Grafiks ar diviem mainīgajiem

```
> plot(platums,garums)
> plot(garums~platums)
```

2.3 Grafiku parametru noteikšana

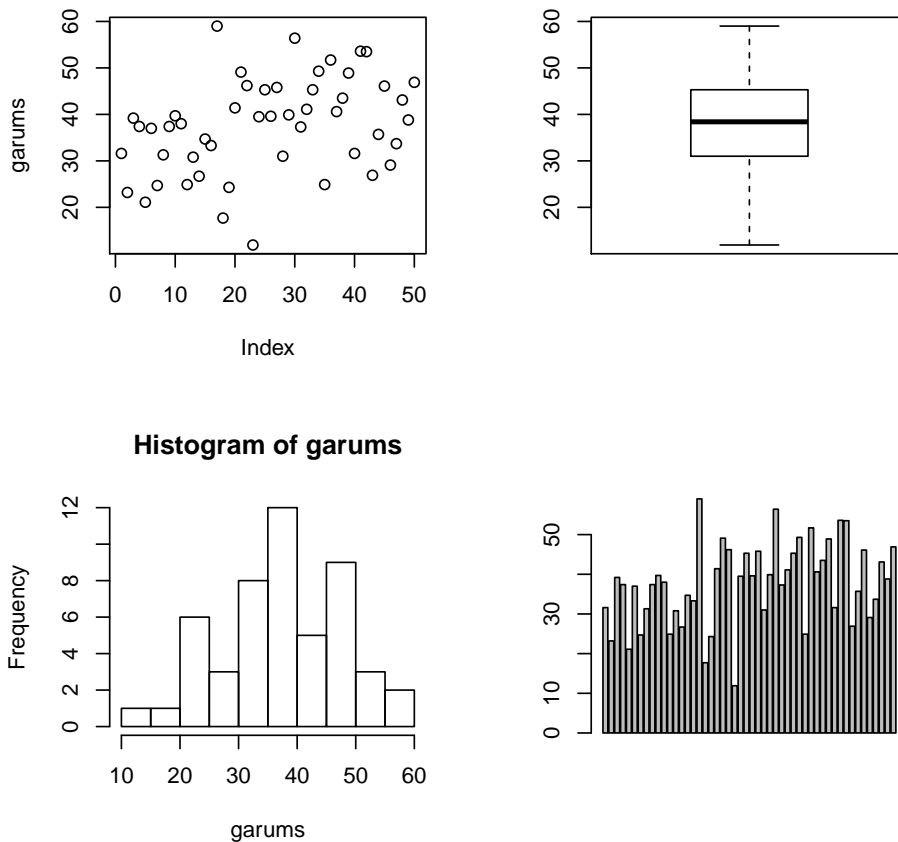
Programmā R grafikiem mainīt paramtrus var divos veidos – izmantojot speciālu funkciju `par()` vai arī rakstot argumentus iekavās pie citām grafiku veidošanas funkcijām. Funkciju `par()` lieto, lai noteiktu parametrus, kas ietekmē visus turpmākos grafikus, kā arī konkrētais grafiskais ekrāns vai vide ir atvērta. Šī funkcija ir jāizsauc kā pirmā pirms grafika veidošanas funkcijas izsaušanas. Daļu no parametriem var noteikt tikai kopā ar funkciju `par()`.

2.3.1 Grafiku izvietošana lapā

R piedāvā iespēju izvietot vienā lapā vairākus attēlus, kas tiek panākts izmantojot argumentus `mfrow`, `mfcoll` un funkciju `layout()`. Visi šie argumenti vai funkcija ir jādefinē pirms grafiku funkcijām, kā arī jāņem vērā tas, ka šīs funkcijas zaudē spēku, ja tiek izmantota funkcija `dev.off()` vai arī grafiku logs tiek aizvērts.

Argumenti `mfrow` un `mfcoll` tiek izmantoti kopā ar funkciju `par()`, pēc sekojoša principa `par(mfrow=c(rinduskaitis,kolonnuskaitis))`. `mfrow` un `mfcoll` atšķirība ir tā, ka pirmajā gadījumā grafiki tiek aizpildīti pēc rindu principa (pirmā rinda, tad otrā rinda), bet otrajā gadījumā aizpilda pēc kolonnu principa (pirmā kolonna, tad otrā kolonna).

Funkcija `layout()` strādā pēc līdzīga principa (2.6 attēls), tikai šajā gadījumā ir iespējams nedefinēt kādā secībā grafiki izvietosies – secīgi vai jauktā veidā, kā arī ar šo funkciju iespējams nedefinēt arī izmērus katram no atsevišķajiem grafikiem.

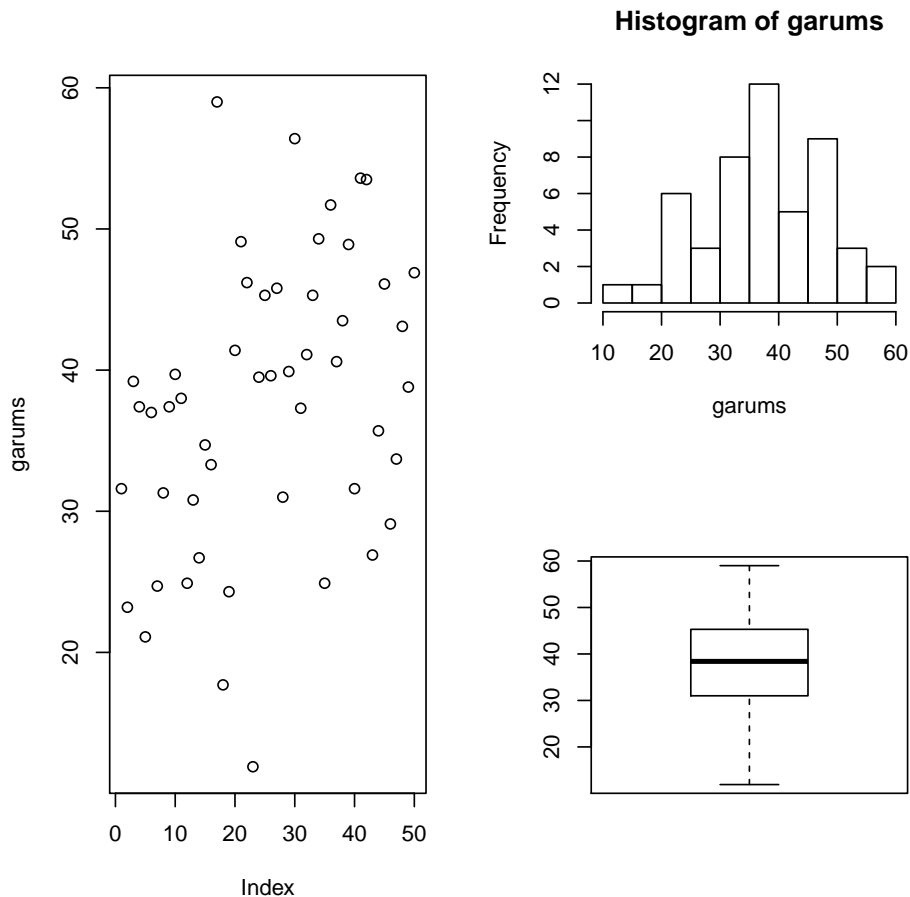


Att. 2.6: Vairāku grafiku izvietojums vienā lapā

```
> par(mfrow=c(2,2))
> par(mfcol=c(2,2))
> layout(matrix(c(1,2,3,4),ncol=2))
> plot(garums)
> hist(garums)
> boxplot(garums)
> barplot(garums)
```

Ja funkcijā `layout()` kāds no skaitļiem tiek atkārtots, tad grafiks tiks izvietots nevis vienā “rūtiņā”, bet gan divās, kāmēr pārējie grafiki izvietosies vienā rūtiņā (2.7 attēls).

```
> layout(matrix(c(1,1,2,3),ncol=2))
> plot(garums)
```



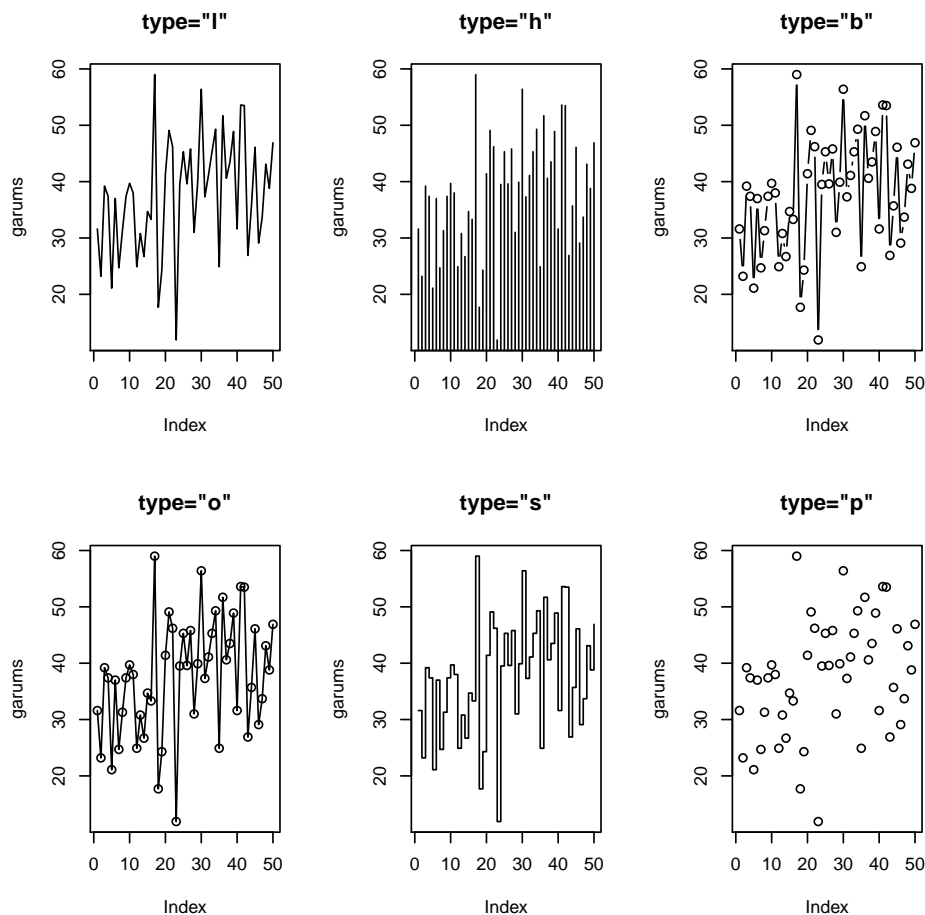
Att. 2.7: Vairāku grafiku izvietojums vienā lapā

```
> hist(garums)
> boxplot(garums)
```

2.3.2 plot() grafiku tipi

Pēc noklusējuma funkcija `plot()` veido izkliedes grafiku, bet to ir iespējams mainīt norādot papildus argumentu `type=` iekavās pie šīs funkcijas. Rezultātā var iegūt sešus dažādus grafiku veidus (2.8 attēls).

```
> par(mfrow=c(2,3))
> plot(garums,type="l",main="type=\"l\"")
> plot(garums,type="h",main="type=\"h\"")
> plot(garums,type="b",main="type=\"b\"")
> plot(garums,type="o",main="type=\"o\"")
```

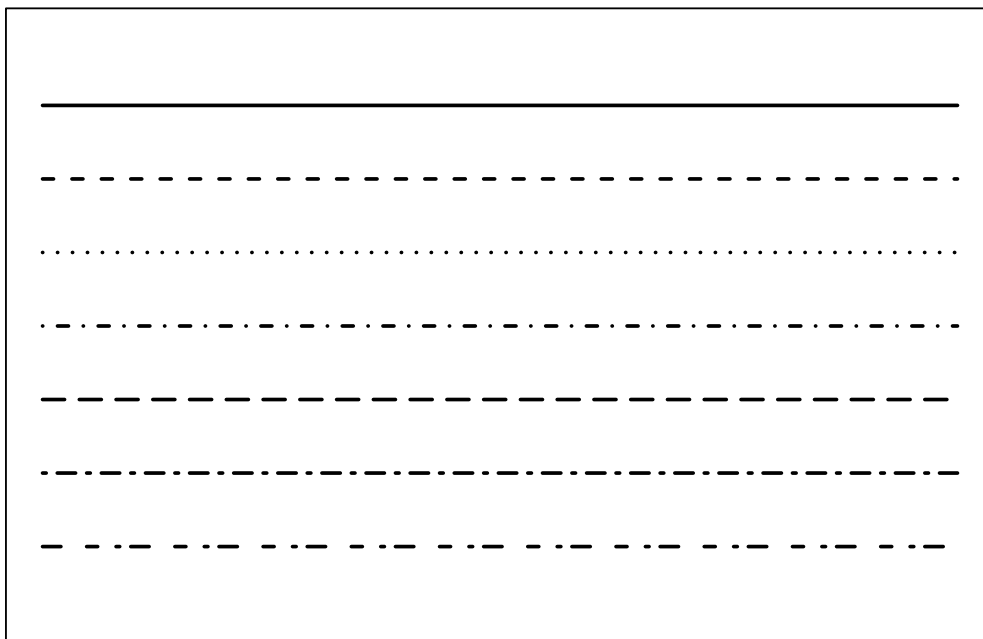


Att. 2.8: Funkcijas `plot()` grafiku veidi

```
> plot(garums,type="s",main="type=\"s\"")
> plot(garums,type="p",main="type=\"p\"")
> dev.off()
```

2.3.3 Līniju parametri

Līnijām, kas tiek attēlotas grafikos, ir iespējams nodēfinēt vairākus parametrus. Tas tiek panākts ar atsevišķiem argumentiem, kurus lieto kopā ar funkcijām grafika izveidošanai. Pirmais no parametriem, kuru var mainīt, ir līnijas veids. To dara ar argumentu `lty=`, norādot skaitli no 0 līdz 6 (bez pēdējām), vai arī lietojot līnijas veida angļu nosaukumu (attiecīgi "blank", "solid", "dashed", "dotted", "dotdash", "longdash", "twodash"). Ir iespējams nodēfinēt arī savu līnijas veidu – pēdējās aiz `lty=` jāraksta skaitļi, kur attiecīgi pirmais skaitlis apzīmē līnijas garumu un otrais skaitlis atstarpes garumu (2.9 attēls). Šajā piemērā funkcija `lines()` tiek izmantota, lai grafikam pievienotu papildus līnijas. Funkcija ir lietojama tikai tad, ja pamatgrafiks jau ir izveidots.



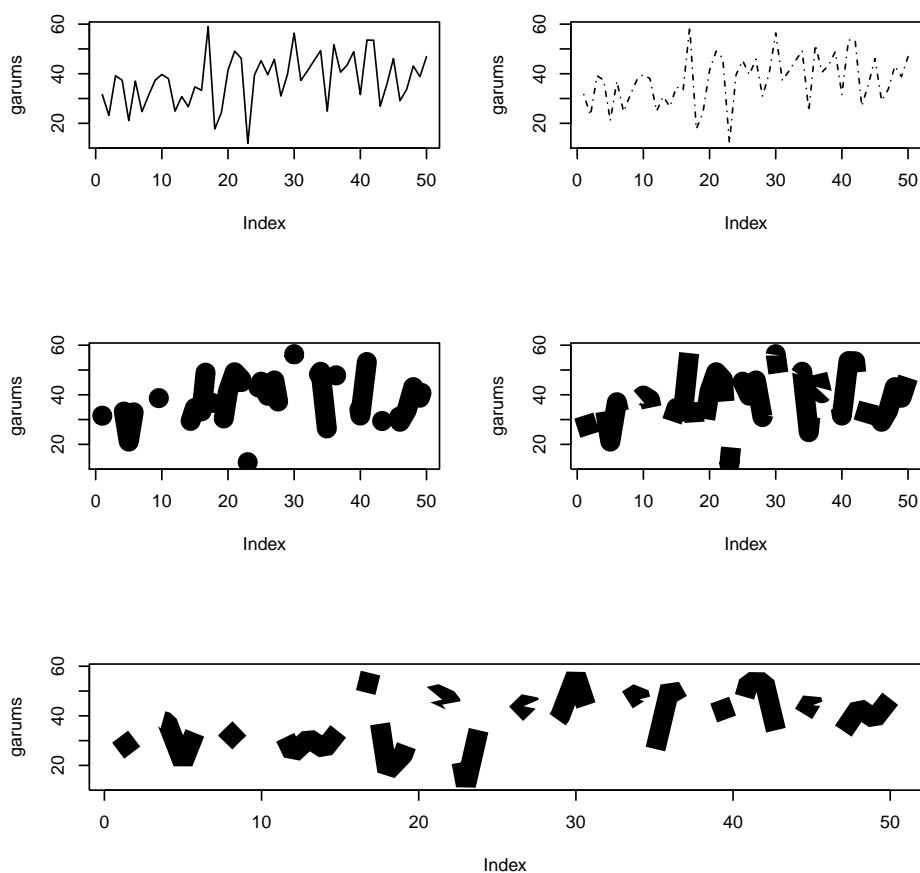
Att. 2.9: Iespējamie līniju tipi grafikos

```
> x<-rep(10,10)
> plot(x,type="l",lty=1,ylim=c(3,11),axes=F,ann=F,lwd=2)
> lines(x-1,lty=2,lwd=2)
> lines(x-2,lty=3,lwd=2)
> lines(x-3,lty=4,lwd=2)
> lines(x-4,lty=5,lwd=2)
> lines(x-5,lty=6,lwd=2)
> lines(x-6,lty="664422",lwd=2)
> box()
```

Lai noteiktu līniju platumu, lieto argumentu `lwd=`, norādot līnijas biezumu kā skaitli. Līnijas platumu būs atkarīgs no tā, kur grafiks tiks attēlots, tas ir, ja grafiku apskatīsies uz ekrāna vai eksportēsiet kā failu, līniju platumu var atšķirties.

Ar argumentu `lend=` nosaka kāds būs līnijas nobeigums: apaļš ("round") vai taisns ("square" vai "butt"). Arguments `ljoin=` nosaka kādā veidā līnijas savienojas savā starpā (ja ir lauza līnija). Savienojums var būt punktveida ("mitre"), apaļš ("round") un ass leņķis ("bevel"). Attēlā 2.10 var sekot tam, kā mainās līnijas izskats, mainot dažādus tās parametrus (platumu, veidu, savienojumu vietas).

```
> layout(matrix(c(1,3,5,2,4,5),ncol=2))
> plot(garums,type="l")
```



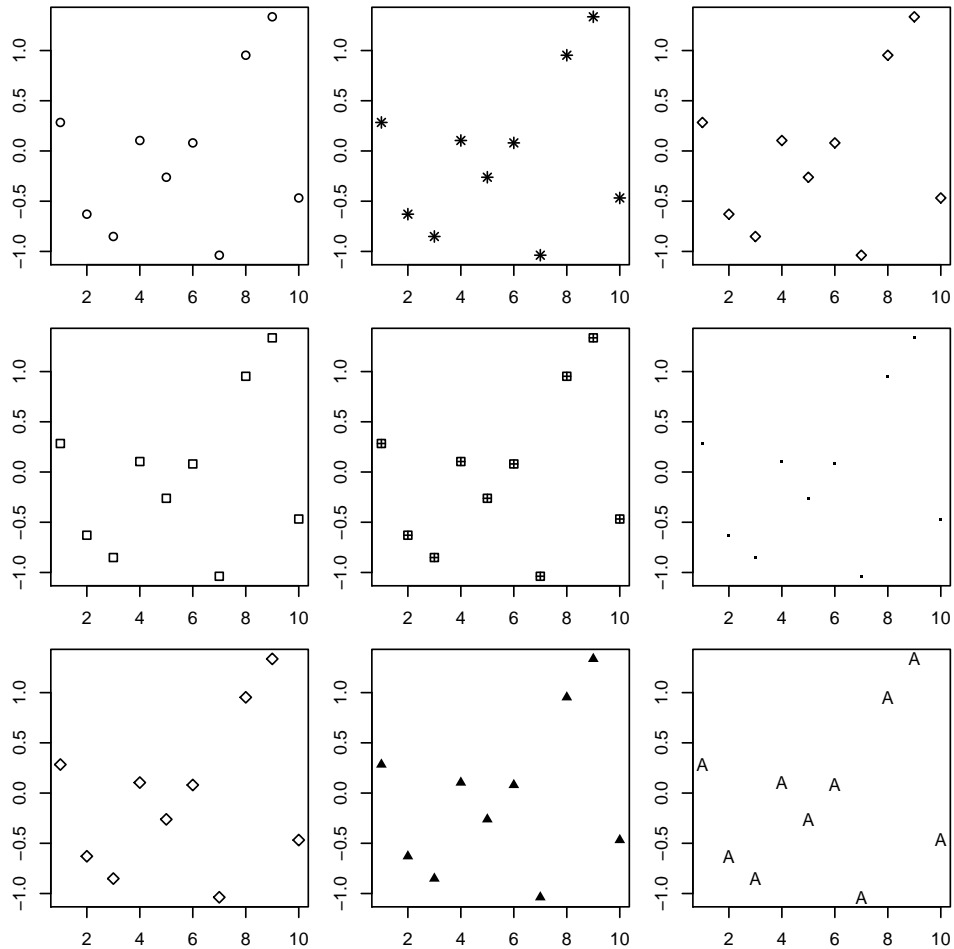
Att. 2.10: Līnijas izmaiņas, mainot dažādus parametrus

```
> plot(garums,type="l",lty=4)
> plot(garums,type="l",lty=4,lwd=12)
> plot(garums,type="l",lty=4,lwd=12,lend="butt")
> plot(garums,type="l",lty=4,lwd=12,lend="butt",ljoin="bevel")
```

2.3.4 Simbolu veidi

Tradicionālajiem grafikiem ir pieejami 26 iepriekš nedefinēti simbolu veidi datu attēlošanai, kurus var mainīt izmantojot argumentu `pch=` un norādot skaitli no 0 līdz 25 (2.11 attēls). Papildus ir iespējams norādīt arī burtus vai pat vārdus, tos liekot pēdējās aiz argumenta `pch=`.

```
> par(mar=c(2,2,1,1))
> layout(matrix(1:9,ncol=3))
> x<-rnorm(10)
```



Att. 2.11: Simbolu veidi

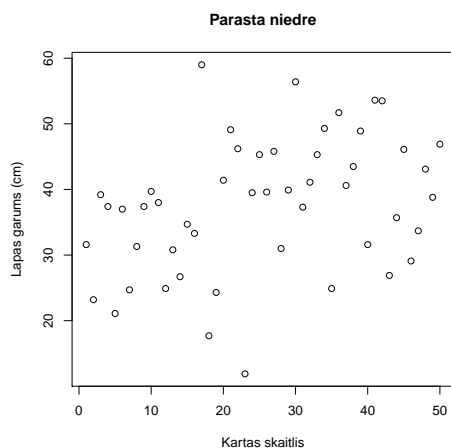
```

> plot(x, ann=F)
> plot(x, pch=0, ann=F)
> plot(x, pch=5, ann=F)
> plot(x, pch=8, ann=F)
> plot(x, pch=12, ann=F)
> plot(x, pch=17, ann=F, )
> plot(x, pch=23, ann=F)
> plot(x, pch=".", ann=F)
> plot(x, pch="A", ann=F)

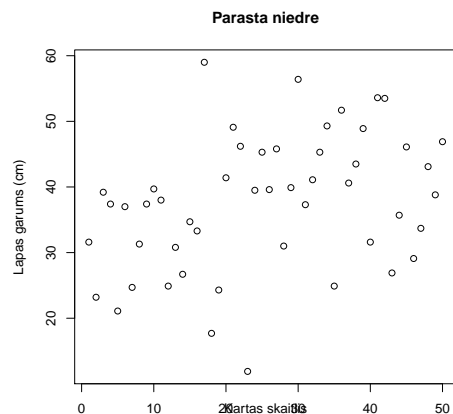
```

2.3.5 Virsraksti un teksti pie asīm

Virsrakstus un tekstus pie asīm var noteikt gan ar argumentiem, gan ar atsevišķām funkcijām. Kā argumenti (piemēram, funkcijai `plot()`) nāk `xlab=()` (izveido parakstu zem x ass), `ylab=` (izveido parakstu zem y ass) un `main=` (izveido virsrakstu), aiz vienādības zīmes pēdējās rakstot atbilstošo tekstu (2.12 attēls).



Att. 2.12: Grafiks, kuram teksti definēti kā argumenti



Att. 2.13: Grafiks, kuram teksti definēti kā atsevišķas funkcijas

```
> plot(garums,main="Parasta niedre",xlab="Kartas skaitlis",
+ ylab="Lapas garums (cm)")
```

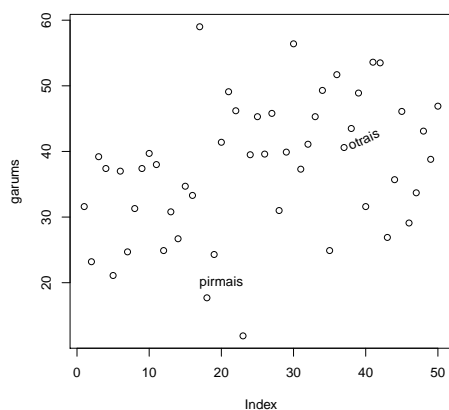
Lai varētu ietekmēt vairāk parametrus, tekstus var norādīt arī ar atsevišķām funkcijām, bet šajā gadījumā, piemēram, pie `plot()` funkcijas jānorāda arguments `ann=FALSE`, lai programma pati neveidotu parakstus zem asīm. Virsrakstu definē ar funkciju `title()`, kur pēdējās raksta nepieciešamo tekstu.

Vispārīgā gadījumā tekstus pie asīm nosaka ar funkciju `mtext('ko gribam redzēt',side=kurā.pusē,line=kurā.līnijā)`. Izvietojums (puse) atbilst asu izvietojumam, teksta līnijas rēķina attiecībā pret asīm, tas ir, ja norādīs `line=1`, tad teksts būs pie pašas ass un, jo lielāks skaitlis, jo tālāk no ass atradīsies teksts (2.13 attēls).

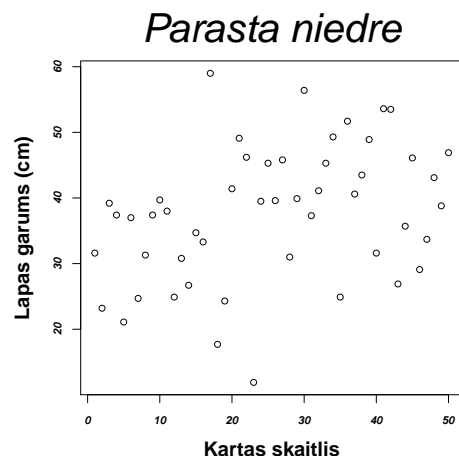
```
> plot(garums,ann=FALSE)
> title("Parasta niedre")
> mtext("Kartas skaitlis",side=1,line=1)
> mtext("Lapas garums (cm)",side=2,line=3)
```

2.3.6 Teksts grafika iekšienē

Ja nepieciešams izvietot tekstu grafika iekšienē, ir jāizmanto funkcija `text()`. Šai funkcijai kā argumentus jānorāda x un y koordinātes (atbilstoši asu skalām), kur jānovieto teksts un jānorāda pats teksts, kuru vēlas ievietot. Ir jāņem vērā, ka pēc noklusējuma teksts tiks izveidots tā, ka norādītās koordinātes ir teksts viduspunkts. Lai to mainītu, ir jānorāda papildus arguments `pos=` ar iespējamām vērtībām: 1 – teksts atrodas zem šī punkta; 2 – teksts atrodas pa kreisi no šī punkta; 3 – teksts atrodas virs šī punkta; 4 – teksts atrodas pa labi no šī punkta. Ja tekstem



Att. 2.14: Tekstu novietojuma piemērs



Att. 2.15: Teksta parametru izmaiņas

vajag mainīt orientāciju (lai tas nebūtu novietots horizontāli), jāizmanto papildus arguments `srt=`, kuram jānorāda teksta novietojums grādos.

Piemērā grafikā (2.14 attēls) ir izvietoti divi uzraksti: “pirmais” – bez papildus argumentiem; “otrais” – novietots virs koordināšu punkta un 25 grādu slīpumā.

```
> plot(garums)
> text(20,20,"pirmais")
> text(40,40,"otrais",pos=3,srt=25)
```

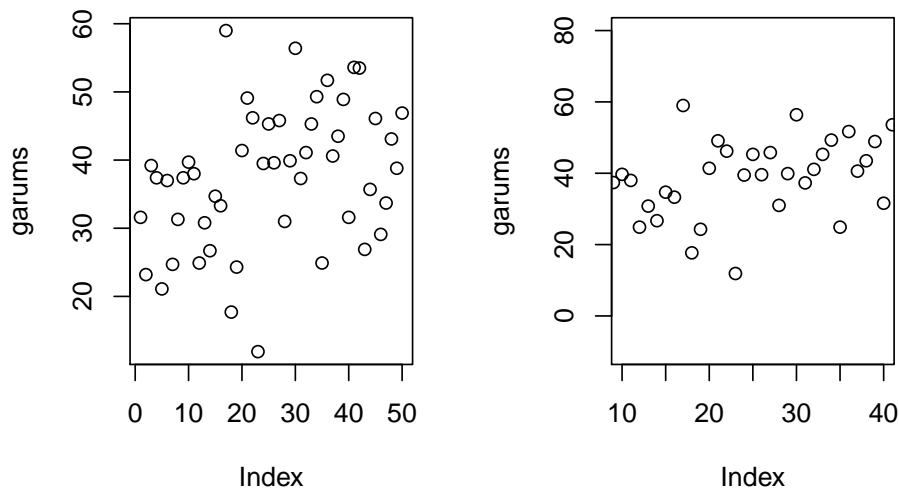
2.3.7 Tekstu parametri

Tekstam var definēt dažādus parametrus, no kuriem svarīgākie būtu fonts, fontu grupa un burtu izmērs. Nosakot fontu, jānosaka kādai fontu grupai tas piederēs ar argumentu `family=`. Fontu grupas ir, piemēram, `'mono'`, `'serif'`, `'sans'`. Fontu tipu nosaka ar argumentu `font=` ar iespējamo vērtību no 1 līdz 5, kur 1 apzīmē parastos burtus, **2 -- bold**, *3 -- italic*, **4 -- bold italic** un 5 -- simboli. Argumentu `family=` var lietot tikai kopā ar funkciju `par()`, bet `font=` var izmantot kopā arī ar citām funkcijām. Lai definētu fonta tipu asīm, asu parakstiem un virsrakstiem, attiecīgi var lietot argumentus `font.axis=`, `textfont.lab=` un `font.main=`.

Burtu izmēru var noteikt ar diviem argumentiem `ps=` un `cex=`. Arguments `ps=` nosaka absolūto burtu izmēru punktos. `cex=` nosaka relatīvo burtu izmēru, piemēram, `cex=1.5` nozīmēs, ka burtiem ir jābūt 1.5 reizes lielākiem nekā ir standarta izmēra burti. `ps=` lieto tikai kopā ar `par()`, bet `cex=` var izmantot kopā arī ar citām funkcijām. `cex.axis`, `cex.lab` un `cex.main` attiecīgi izmanto, lai apzīmētu asu simbolu, asu parakstu un virsrakstu relatīvo izmēru.

Piemērā (2.15 attēls) ir izmainīti izmēri virsrakstiem, parakstiem un skaitļiem pie asīm, kā arī katram no šiem parametriem ir izmantots cits fonta tips.

```
> plot(garums,main="Parasta niedre",xlab="Kartas skaitlis",
+ ylab="Lapas garums (cm)",cex.main=3,cex.lab=1.5,
```

Att. 2.16: Asu garuma izmaiņu piemērs

```
+ cex.axis=0.8,font.main=3,font.axis=4,font.lab=2)
```

2.3.8 Asis

Ja ir nepieciešams noteikt asu garumu (to ierobežojot kādā intervālā, vai arī tieši otrādi – padarot asi garāku, nekā tā tiek automātiski veidota) (2.16 attēls), var izmantot argumentus `xlim=` un `ylim=`, ko lieto, piemēram, ar funkciju `plot()`. Abiem argumentiem pieraksts ir vienāds

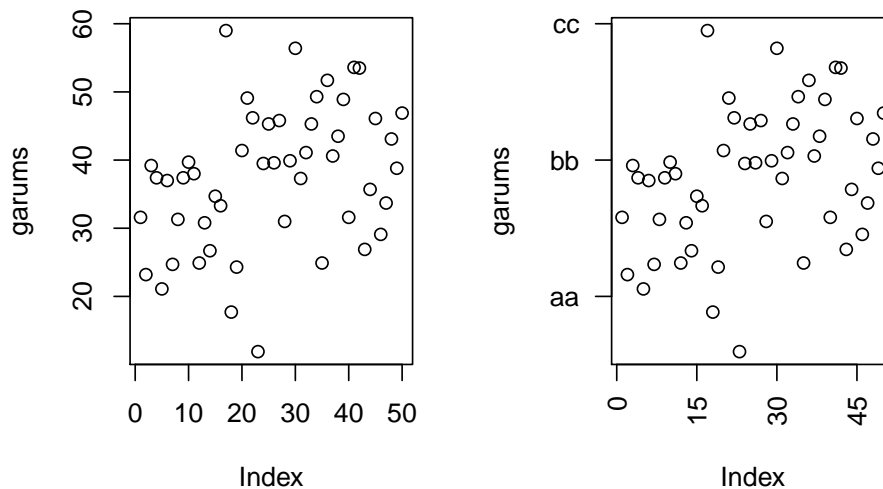
```
xlim=c(sākuma.vērtība,beigu.vērtība)
```

```
> par(mfcol=c(1,2))
> plot(garums)
> plot(garums,xlim=c(10,40),ylim=c(-10,80))
```

Ja grib ietekmēt vairākus ass parametrus, tad jāizmanto funkcija `axis()`, bet pirms tam, piemēram, kā arguments pie `plot()`, jāraksta `axes=FALSE`, lai programma automātiski pati neveidotu asis. Funkcijas pieraksts ir

```
axis(kura.ass,at=seq(sākuma.vērtība,beigu.vērtība,intervāls),labels=c(ko.attēlot)).
```

Asu numerācija ir sekojošā: apakšējā ir 1, kreisā ir 2, augšējā 3 un labā ir 4. Ja argumentu `labels=` nenorāda, tad tiek attēloti tie paši skaitļi, kas ir pie argumenta `at=`. Ar argumentu `las=` var mainīt skaitļu/tekstu novietojumu pie ass, kur 0 nozīmē paralēls asij, 1 – horizontāls, 2 – perpendikulārs asij, 3 – vertikāls.



Att. 2.17: Asu definēšanas piemērs

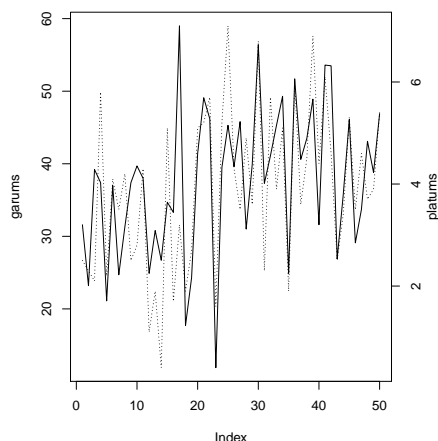
Piemērā (2.17 attēls) x asij nodefinēts, ka skaitļiem pie ass jābūt no 0 ik pēc 15, toties y asij pie skaitļiem 20, 40 un 60 ir jāparādās tekstam, nevis skaitļiem.

```
> par(mfcol=c(1,2))
> plot(garums)
> plot(garums,axes=FALSE)
> axis(1,at=seq(0,50,15),las=2)
> axis(2,at=c(20,40,60),labels=c("aa","bb","cc"),las=1)
> box()
```

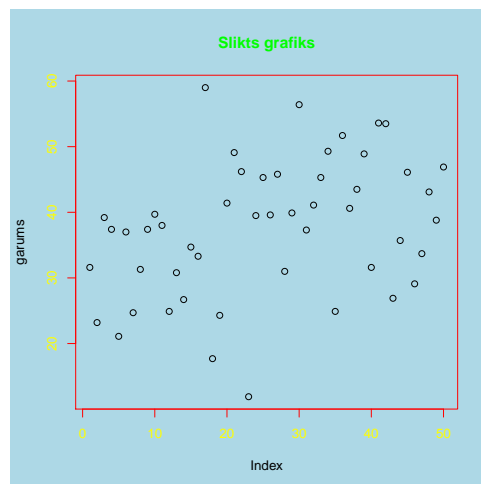
2.3.9 Divas y asis

Lai arī daļa speciālistu atzīst, ka grafiki ar y asīm nav labi, reizēm tādus ir nepieciešamas izveidot. Pirmkārt, ar funkciju `par()` un argumentu `mar=` jāizmaina malu izmēri, lai otrai y asij būtu vietas. Pēc tam jāuzzīmē pamatgrafiks, aiz kura nāk funkcija `par(new=TRUE)`, kas ļauj pa "virsu" zīmēt otru grafiku. Otrajam grafikam jālieto argumenti `axes=FALSE` un `ann=FALSE`, lai apzīmējumi nepārklātos. Tālāk jau ar funkcijām `axis()` un `mtext()` iegūst vēlamos apzīmējumus pie atbilstošās ass (2.18 attēls).

```
> par(mar=c(5,5,2,5))
> plot(garums,type="l")
> par(new=TRUE)
```



Att. 2.18: Grafiks ar divām y asīm



Att. 2.19: Krāsu izmantošanas piemērs

```
> plot(platums, type="l", lty=3, axes=F, ann=F)
> axis(4, at=seq(0, 8, 2))
> mtext(side=4, line=2, "platums")
```

2.3.10 Krāsas

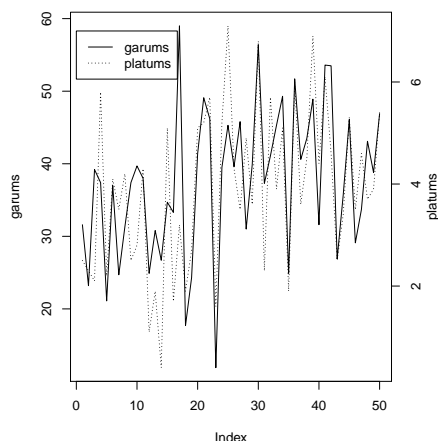
Ir iespējams mainīt krāsas visām grafika sastāvdaļām, izmantojot dažādus argumentus (2.19 attēls). Šos argumentus var rakstīt kā daļu no funkcijas `par()` un šajā gadījumā krāsas būs noteiktas visiem grafikiem, vai arī lietot kā daļu, piemēram, no funkcijas `plot()` un šajā gadījumā krāsas mainīsies tikai konkrētajam grafikam. Vieglākais veids kā apzīmēt krāsu, ir lietot tās angļu nosaukumu, piemēram, “red” vai “darkgreen”. Lai redzētu visu pieejamo krāsu nosaukumus, jāraksta komanda `colors()`.

Pamatā ir trīs argumenti krāsu maiņai: `col`, `bg` un `fg`. `col='krāsas.nosaukums'` ir visbiežāk lietotais krāsas parametrs, ar kuru pārsvārā tiek noteikts datu simbolu, līniju, tekstu un citu sastāvdaļu, kas atrodas grafikā krāsa. Lai noteiktu asu, asu parakstu, virsrakstu un apakšvirsrakstu krāsas, jālieto attiecīgi argumenti `col.axis`, `col.lab`, `col.main` un `col.sub`. `fg='krāsas.nosaukums'` pārsvārā izmanto, lai mainītu asu un grafika malu krāsu, un daļēji tas sakrīt ar `col.axis` un `col.main` argumentiem. `bg='krāsas.nosaukums'` nosaka fona krāsu, šo argumentu var lietot kopā tikai ar funkciju `par()`.

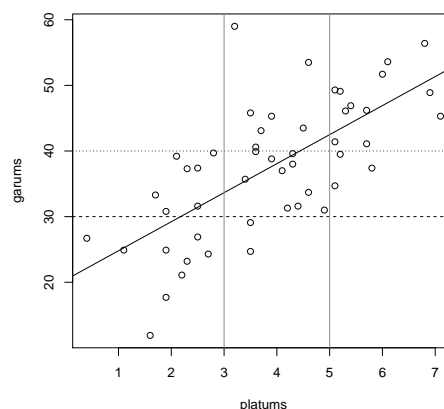
```
> par(bg="lightblue")
> plot(garums, fg="red", main="Slikts grafiks", col.main="green", col.axis="yellow")
```

2.3.11 Leģenda

Tradicionālajiem grafikiem leģendas programmā R veido ar atsevišķu funkciju `legend()`. Leģenda parasti ir novietota grafika iekšējā daļā (2.20 attēls). Dotajai funkcijai kā argumentus



Att. 2.20: Grafiks ar leģendu



Att. 2.21: Grafiks ar papildus līnijām

jānorāda: (a) leģendas novietojums – to nosaka ar x un y koordināti leģendas rāmīša augšējā kreisajam stūrim (koordinātes nosaka atbilstoši asu vērtībām, piemēram, ja uz x ass vērtības ir robežās no 5 līdz 10, bet uz y ass no 100 līdz 200, tad leģenas koordināte varētu būt 6 un 125); (b) nosaukumi, kuriem jāparādās leģendā; (c) datu simbolu/līniju veidu apzīmējumi (tiem ir jāsakrīt ar apzīmējumiem, kas izmantoti, lai definētu grafiku).

```
> par(mar=c(5,5,2,5))
> plot(garums,type="l")
> par(new=TRUE)
> plot(platums,type="l",lty=3,axes=F,ann=F)
> axis(4,at=seq(0,8,2))
> mtext(side=4,line=2,"platums")
> legend(0,7,lty=c(1,3),c("garums","platums"))
```

2.3.12 Līnijas grafikos

Ja grafikā ir nepieciešams izvietot taisnu līniju, var izmantot funkciju `abline()`. Ja ir nepieciešams uzzīmēt slīpu līniju, tad ir jānorāda divi skaitļi, kas attiecīgi norāda punktu, kurā līnija krusto x asi un līnijas slīpumu (intercept), vai arī jāizmanto papildus funkcija `lm()`, kas izveidos regresijas līkni grafikā attēlotajiem mainīgajiem (2.21 attēls). Ja nepieciešams izveidot horizontālas līnijas, tad jānorāda arguments `h=` un vērtības (viena vai vairākas), kurās vietās horizontālās līnijas krustos y asi. Attiecīgi vertikālām līnijām ir jānorāda arguments `v=`.

```
> plot(garums~platums)
> abline(lm(garums~platums))
> abline(h=c(30,40),lty=c(2,3))
> abline(v=c(3,5),col="grey55")
```

2.4 Grafiku saglabāšana

Programmā R iegūtos grafikus ir iespējams saglabāt dažādos formātos, izmantošanai citās programmās. Jau izveidotu grafiku, kas redzams uz ekrāna, ir iespējams saglabāt izmantojot opciju *File/Save as...* un izvēloties nepieciešamo faila veidu (Metafile, Postscript, PDF, Png, Bmp, Jpeg (kvalitāte 50%, 75% vai 100%)), kā arī grafiku ir iespējams ielikt atmiņā *File/Copy to clipboard...* Saglabājot jau izveidotu grafiku, ir jāņem vērā, ka uz ekrāna grafiks ne vienmēr tiek attēlots pareizākajās proporcijās, sevišķi, ja ekrāna izmērus esat pielāgojuši paši.

Grafikus ir iespējams uzreiz saglabāt nepieciešamajā formātā ar atbilstošu komandu rindu, tikai šajā gadījumā grafiks netiks parādīts uz ekrāna. Lai grafikus uzreiz saglabātu, pirms visām komandu rindām, ar kurām nosaka grafika parametrus, ir jāraksta atbilstoša komanda, kas nosaka saglabājamā faila formātu un faila nosaukumu:

```
postscript(file='nosaukums.ps')
png(file='nosaukums.png')
pdf(file='nosaukums.pdf')
jpeg(file='nosaukums.jpg')
bmp(file='nosaukums.bmp')
```

Pēc tam, kad ir uzrakstītas visas vajadzīgās grafika komandu rindas, ir “jāatslēdz” komanda, ar kuru noteica faila formātu un faila nosaukumu. Pretējā gadījumā visi nākamie grafiki arī tiks saglabāti konkrētajā failā, aizstājot iepriekšējo. Lai to izdarītu, raksta komandu `dev.off()`.

Nodaļa 3

Datu izzināšana un normalitātes testi

3.1 Datu izzināšana

Pirms uzsākt reālu datu analīzi, veicot dažādus statistiskos testus, vienmēr vajag sākotnējo datu izzināšanu, izmantojot dažādus datu grafiskos attēlojumus. Tādējādi ir iespēja jau pirms statistisko analīžu veikšanas novērtēt vai datus redzamas kādas tendences, savstarpējās saistības, grupēšanās, kā arī secināt vai datus nav kādas dīvainas, neiederīgas vērtības.

Kā piemērs izmantots datu fails *niedres2.txt*, kas satur informāciju par niedru lapu garumu un platumu trīs parauglaukumos.

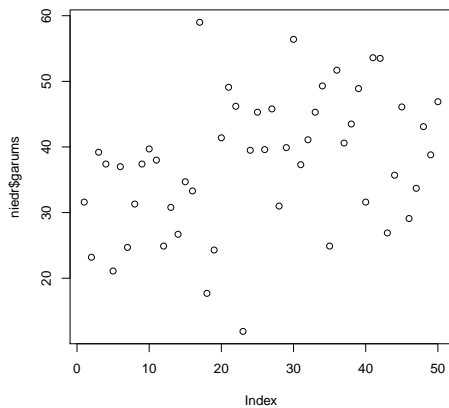
```
> niedr<-read.table(file="niedres2.txt",header=TRUE,sep="\t",dec=".")
> str(niedr)

'data.frame':      50 obs. of  3 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num  2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
 $ paraug : Factor w/ 3 levels "Austr","Riet",...: 1 1 1 1 1 1 1 1 1 1 ...
```

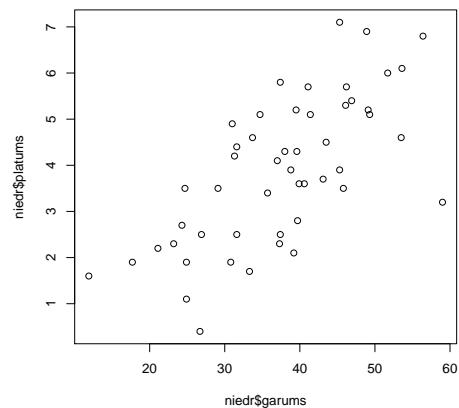
Vienkāršākais grafiks datu apskatīšanai ir izkliedes grafiks, ko programmā R var iegūt ar funkciju `plot()`. Šajā grafikā (3.1 attēls) uz x ass atlikts novērojuma kārtas numurs, uz y ass novērojumu vērtības. Pēc izkliedes grafika var gūt priekšstatu par vērtību izkliedi, gan arī novērtēt vai nav kādas ekstremāli mazas vai lielas vērtības.

```
> plot(niedr$garums)
```

Ja funkcijā `plot()` norāda divus mainīgos, tad vienlaicīgi var novērtēt gan šo mainīgo vērtību saistību (3.2 attēls), gan arī pamanīt kādus ekstrēmumus, ko nav iespējams novērtēt skatot katru mainīgo atsevišķi.



Att. 3.1: Viena mainīgā izkliedes grafiks

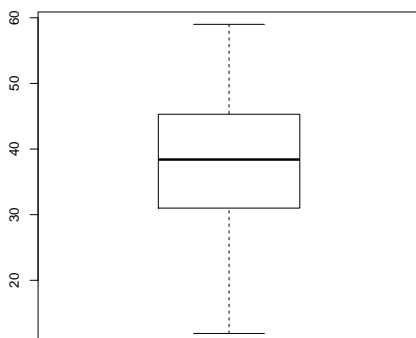


Att. 3.2: Divu mainīgo izkliedes grafiks

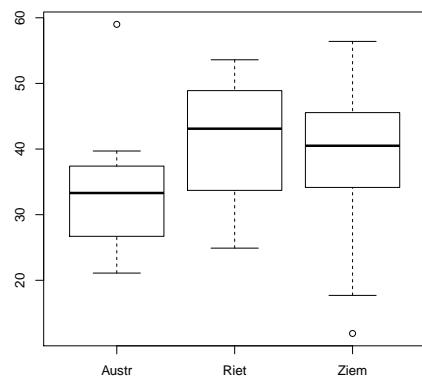
```
> plot(niedr$garums,niedr$platums)
```

Box-plot grafiki, ko iegūst ar funkciju `boxplot()`, ir piemēroti, lai novērtētu vērtību izkliedi datos (3.3 attēls). Box-plot grafikā līnija, kas atrodas taisnstūra vidū, atbilst mediānai, taisnstūra apakšējā un augšējā mala attiecīgi ir 1. un 3. kvartile. Apakšējā un augšējā līnija attiecīgi ir minimālā un maksimālā vērtība datos, ar piebildi, ka šīs līnijas neatrodas tālāk kā 1,5 reiz taisnstūra platumu (attālums starp 1. un 3. kvartili). Ja minimālā vai maksimālā vērtība ir tālāk nekā šīs 1,5 reizes, tad šos novērojumus apzīmē ar punktu.

```
> boxplot(niedr$garums)
```



Att. 3.3: Box-plot grafiks vienam mainīgajam



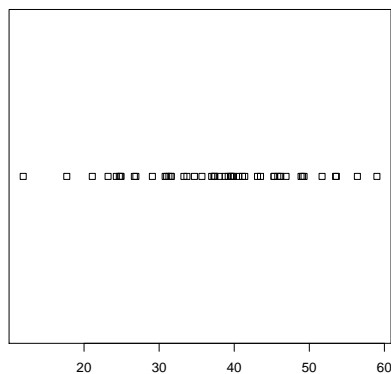
Att. 3.4: Box-plot grafiks trīs paraugkopām

Ja funkcijā `boxplot()` norāda mainīgo, kas satur dalījumu līmeņos, tad atsevišķs grafiks tiek izveidots katram no dalījuma līmeņiem (3.4 attēls). Pēc šī grafika var salīdzināt datu izkliedi uzreiz vairākiem līmeņiem.

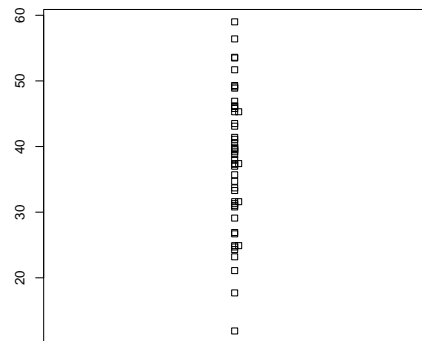
```
> boxplot(niedr$garums~niedr$paraug)
```

Gadījumos, kad novērojumu skaits ir salīdzinoši mazs, tad labāk datu izkliedi raksturo punktu grafiku (3.5 attēls), ko var iegūt ar funkciju `stripchart()`. Šajā grafikā katrs novērojums ir attēlots ar kvadrātiņiem un visi novērojumi izkārtoti vienā strīpā.

```
> stripchart(niedr$garums)
```



Att. 3.5: Punktu grafiks

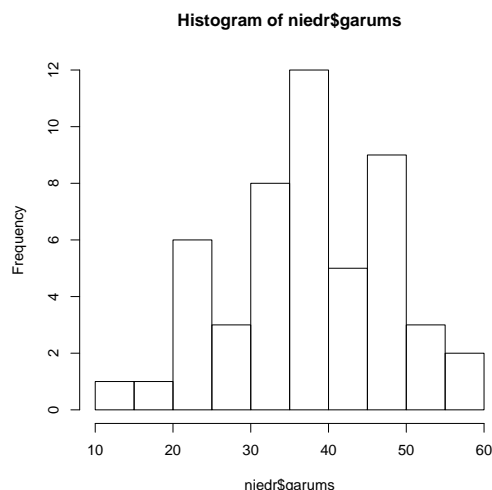


Att. 3.6: Modificēts punktu grafiks

Ja novērojumi pārklājas un nav iespējams precīzi attēlā novērtēt, cik daudz novērojumu atbilst katrai vērtībai, funkciju `stripchart()` var papildināt ar argumentu `method="stack"`, kas novietot kvadrātiņus blakus, ja tie pilnībā pārklājas (3.6 attēls). Arguments `vert=T` nodrošina, ka grafiks novietojas vertikāli, nevis horizontāli.

```
> stripchart(niedr$garums,method="stack",vert=T)
```

Vēl viens plaši izmantots datu attēlošanas veids ir histogramma, ko programmā R var izveidot ar funkciju `hist()`. Histogrammā dati tiek sadalīti klasēs un tādējādi var novērtēt vērtību sadalījuma veidu – vienmērīgi, zvanveidīgi, ar izteiktām minimālām vai maksimālām vērtībām (3.7 attēls). Funkcijai var mainīt argumentus gan nosakot cik klasēs dalīt datus, gan arī nosakot tieši dalījuma robežas.



Att. 3.7: Pazīmes niedru lapu garums histogramma

```
> hist(niedr$garums)
```

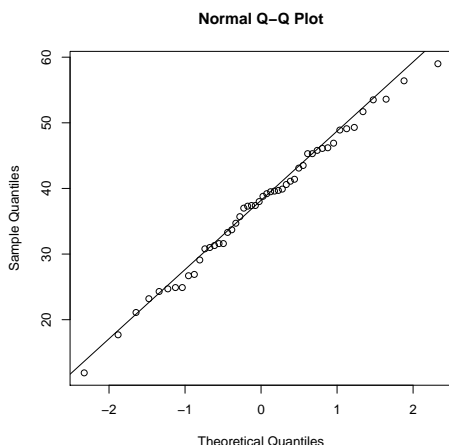
3.2 Normalitātes testi

3.2.1 Grafiskā analīze

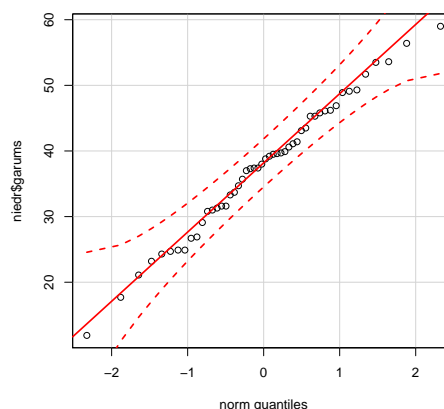
Daudzu statistisko metožu pieņēmums ir, ka analizējamie dati nāk no ģenerālkopas, kura atbilst normālajam sadalījumam. Lai par to pārlicinātos, ir jāveic noteikti statistiskie testi. Jaunākajā statistiskajā literatūrā arvien biežāk kā pamatmetode normalitātes novērtēšanai tiek izmantota grafiskā analīzē, nevis analītiskie testi.

Viens no grafiku veidiem, ko izmantot normalitātes novērtēšanai, var būt histogramma (3.7 attēls). Histogrammai ir jāveido apmēram zvanveidīgs izskats, lai pieņemtu, ka dati atbilst normālajam sadalījumam. Ar šo grafiku veidu problēma ir tā, ka histogrammas izskats lielā mērā būs atkarīgs no tā, cik klasēs dati ir sadalīti.

Uzticamāks grafika veids ir tā saukti QQ grafiki, kas attēlo attiecību starp reālo datu kvantilēm un teorētisko datu kvantilēm (teorētiskie dati veidoti balstoties uz reālo datu statistiskajiem rādītājiem tā, lai tie atbilstu normālajam sadalījumam). Ja reālie dati atbilst normālajam sadalījumam, tad grafikā visi punkti novietojas uz diagonāles. Programmā R QQ grafiku veido ar funkciju `qqnorm()`, kurai kā argumentu norāda reālos datus. Papildus var izmantot arī funkciju `qqline()`, kas novelk līniju, lai būtu vieglāk interpretēt rezultātus. Iegūtajā grafikā (3.8 attēls) ideālā gadījumā visiem punktiem būtu jāatrodas uz taisnes, bet nelielas novirzes arī ir akceptējamas. Lai iemācītos strādāt ar šādiem grafikiem, var ģenerēt mākslīgus datus no normālā sadalījuma un skatīties kā kātreiz izskatās QQ grafiks.



Att. 3.8: QQ grafiks niedru lapu garumam



Att. 3.9: Modificēts QQ grafiks niedru lapu garumam

```
> qqnorm(niedr$garums)
> qqline(niedr$garums)
```

Paketē `car` ir funkcija `qqPlot()`, kas arī veido QQ grafiku, tikai tas ir papildināts ar līnijām, kas parāda 95% ticamības intervālu (3.9 attēls), tādējādi atvieglojot interpretāciju. Ja visi punkti atrodas starp raustītajām līnijām, tad ar 95% pārliecību var apgalvot, ka dati nāk no ģenerālkopas, kas atbilst normālajam sadalījumam.

```
> library(car)
> qqPlot(niedr$garums)
```

3.2.2 Analītiskā analīze

Ja ir nepieciešams daudz formālāks veids kā novērtēt datu atbilstību normālajam sadalījumam, var izmantot kādu no analītiskajām metodēm. R bāzes versijā ir pieejams Šapiro-Vilka normalitātes tests, ko var iegūt ar funkciju `shapiro.test()`. Funkcijai kā arguments jānorāda tikai mainīgais, kuram veikt normalitātes testu. Nulles hipotēze par to, ka dati atbilst normālajam sadalījumam nebūs noraidīta, ja iegūtā p-vērtība būs lielāka par izvēlēto būtiskuma līmeni.

```
> shapiro.test(niedr$garums)

Shapiro-Wilk normality test

data:  niedr$garums
W = 0.991, p-value = 0.9668
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ pazīmes niedru lapu garums vērtības atbilst normālajam sadalījumam, jo p-vērtība ir lielāka par noteikto būtiskuma līmeni ($0,97 > 0,05$).

Funkciju `shapiro.test()` var izmantot arī kopā ar funkciju `tapply()`, tādējādi veicot šo testu vairākiem dalījuma līmeņiem uzreiz.

```
> tapply(niedr$garums, niedr$paraug, shapiro.test)
```

```
$Austr
```

```
Shapiro-Wilk normality test
```

```
data: X[[1L]]
```

```
W = 0.8863, p-value = 0.04028
```

```
$Riet
```

```
Shapiro-Wilk normality test
```

```
data: X[[2L]]
```

```
W = 0.9426, p-value = 0.3508
```

```
$Ziem
```

```
Shapiro-Wilk normality test
```

```
data: X[[3L]]
```

```
W = 0.9034, p-value = 0.09103
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ pazīmes niedru lapu garums vērtības atbilst normālajam sadalījumam Rietumu un Ziemeļu parauglaukumos, jo p-vērtības ir lielākas par noteikto būtiskuma līmeni ($0,35 > 0,05$ un $0,09 > 0,05$), toties Austrumu parauglaukuma dati neatbilst normālajam sadalījumam, jo p-vērtība ir mazāka par noteikto būtiskuma līmeni ($0,04 < 0,05$).

Programmā R ir pieejama pakete `nortest`, kurā ir apvienoti pieci dažādi normalitātes testi, piemēram, Andersona-Darlinga tests un Lilliefora (Kolmogorova-Smirnova) tests. Testi savā starpā atšķiras ar algoritmiem kādā veidā salīdzina reālos datus ar teorētiski sagaidāmajiem, attiecīgi arī iegūtās p-vērtības starp testiem mēdz atšķirties. Daļa no testiem ir striktāki, daļa mazāk strikti.

```
> library(nortest)
```

```
> ad.test(niedr$garums)
```

```
Anderson-Darling normality test
```

```
data: niedr$garums
```

```
A = 0.1737, p-value = 0.9223
```

Nodaļa 4

Statistiskie rādītāji un ticamības intervāli

4.1 Statistiskie rādītāji

Piemēram par statistisko rādītāju aprēķināšanu izmantot datu fails *niedres2.txt*, kas satur informāciju par niedru lapu garumu un platumu trīs parauglaukumos.

```
> niedr<-read.table(file="niedres2.txt",header=TRUE,sep="\t",dec=".")
> str(niedr)

'data.frame':      50 obs. of  3 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num  2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
 $ paraug  : Factor w/ 3 levels "Austr","Riet",...: 1 1 1 1 1 1 1 1 1 1 ...
```

Pamatstatistisko rādītāju aprēķināšanai ir jau definētas funkcijas: `mean()` – vidējais aritmētiskais, `sd()` – standartnovirze, `var` – dispersija, `median()` – mediāna. Visām šīm funkcijām kā mainīgais jānorāda viena kolonna/vektors un tiks aprēķināts vēlamais rādītājs.

```
> mean(niedr$garums)

[1] 37.594

> sd(niedr$garums)

[1] 10.31332

> var(niedr$garums)

[1] 106.3647

> median(niedr$garums)

[1] 38.4
```

Veicot aprēķinus rezultātam ciparu skaits aiz komata ir atkarīgs no tā, kādi ir sesijas uzstādījumi. Ja ir nepieciešams noapaļot skaitļus, var izmantot papildus funkciju `round()`, kurā norāda apaļojamo mainīgo vai visu funkciju, kā arī jānorāda vēlmais ciparu skaits aiz komata. Ir jāatceras, ka noapaļot drīkst tikai gala rezultātus – ja noapaļo skaitļus, kurus vēlāk izmanto citos aprēķinos, tas samazina iegūtā rezultāta precizitāti.

```
> round(mean(niedr$garums),2)
[1] 37.59
```

Viena no daudzu programma R funkciju īpatnībām ir tā, ka nav iespējams iegūt rezultātu, ja dati satur kādu iztrūkstošu vērtību. Piemēram, radam vektoru x , kas satur 20 skaitļus no normālā sadalījuma un vienu iztrūkstošu vērtību, ko R apzīmē ar NA. Ja šādam vektoram cenšas aprēķināt vidējo aritmētisko ar funkciju `mean()`, iegūst rezultātu NA, jo nezināmā vērtība var būt jebkas, attiecīgi arī vidējais var būt jebkas.

```
> x<-c(rnorm(20),NA)
> mean(x)
[1] NA
```

Šādā situācijā var norādīt, ka NA vērtības ir jāignorē. To panāk funkcijai kā papildus argumentu norādot `na.rm=TRUE`. Daļai funkciju šī argumenta pieraksts var būt atšķirīgs, to var precizēt funkcijas aprakstā.

```
> mean(x,na.rm=TRUE)
[1] 0.1944777
```

Minimālās un maksimālās vērtības aprēķināšanai var izmantot attiecīgi funkcijas `min()` un `max()`, vai arī funkciju `range()`, kas aprēķina uzreiz abus šos rādītājus.

```
> min(niedr$garums)
[1] 11.9
> max(niedr$garums)
[1] 59
> range(niedr$garums)
[1] 11.9 59.0
```

Kvartiļu aprēķināšanai izmanto funkciju `quantile()`, kas pārādīs uzreiz visas kvartiles tās izsakot kā procentiles.

```
> quantile(niedr$garums)

 0%   25%   50%   75%  100%
11.900 31.075 38.400 45.300 59.000
```

Ja ir nepieciešams aprēķināt procentiles, vai arī tikai kādu noteiktu kvartili, funkcijai `quantile()` kā papildus arguments jānorāda `probs=` un nepieciešamā procentile izteikta decimāldaļās.

```
> quantile(niedr$garums,probs=c(0.025,0.975))

 2.5%  97.5%
18.465 55.770
```

Ar funkciju `summary()` ir iespējams aprēķināt uzreiz vairākus statistiskos rādītājus, kas raksturo mainīgo. Šie rādītāji ir minimālā vērtība, 1. kvartile, mediāna, vidējais aritmētiskais, 3. kvartile un maksimālā vērtība. Šai funkcijai kā mainīgo var norādīt vienu kolonnu vai vektoru.

```
> summary(niedr$garums)

  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
11.90  31.08   38.40   37.59  45.30   59.00
```

Ja funkcijā `summary()` kā mainīgo norāda veselu datu tabulu, tad iepriekš minētie rādītāji tiek aprēķināti katrai kolonnai, kas satur skaitlisku informāciju. Kolonnai, kas satur faktoru jeb tekstuālu informāciju, aprēķina cik bieži atkārtojas katra unikālā vērtība.

```
> summary(niedr)

      garums      platums      paraug
Min.   :11.90  Min.   :0.400  Austr:17
1st Qu.:31.07  1st Qu.:2.500  Riet :17
Median :38.40  Median :3.900  Ziem :16
Mean   :37.59  Mean   :3.892
3rd Qu.:45.30  3rd Qu.:5.100
Max.   :59.00  Max.   :7.100
```

Funkcija `tapply()` ir ļoti noderīga gadījumos, ja nepieciešams aprēķināt kādu rādītāju viena vektora/kolonnas vērtībām tās sadalot pa līmeņiem, kurus nosaka otrs vektors/kolonna. Funkcijā jānorāda mainīgais, kas satur vērtības, mainīgais, kas satur dalījuma līmeņus, un funkcija, kuru izmantot aprēķinos. Ir iespējams izmantot ne tikai jau definētas funkcijas, bet arī definēt jaunas funkcijas, izmantojot `function()` funkciju. Piemērā aprēķināta standartnovirze kolonnai `garums`, ka sadalīta pa parauglaukumiem.

```
> tapply(niedr$garums,niedr$paraug,sd)
```

```
      Austr      Riet      Ziem
8.869876  9.376174 11.706401
```

4.2 Ticamības intervāli

Vidējā aritmētiskā ticamības intervālu var aprēķināt gan izmantojot gatavas funkcijas, gan arī izmantojot aprēķina formulas. Stjudenta kritērija teorētisko vērtību aprēķina ar funkciju `qt()`, kurai kā pirmais arguments jānorāda $(1 - \alpha/2)$ (šoreiz izvēlamies strādāt pie ticamības līmeņa 99%) un otrais arguments ir brīvības pakāpju skaits (49). Vidējā aritmētiskā reprezentācijas rādītāju aprēķina kā standartnovirze dalīts ar kvadrātsakni no novērojumu skaita. Pēc tam ticamības intervāla augšējo un apakšējo robežu iegūst pieskaitot vai atņemot robežas vērtību no vidējā aritmētiskā. Piemēram izmantots niedru lapu garums.

```
> robeza<-qt((1-0.01/2),49)*(sd(niedr$garums))/sqrt(50)
> augsa<-mean(niedr$garums)+robeza
> apaksa<-mean(niedr$garums)-robeza
> round(apaksa,2)
```

```
[1] 33.69
```

```
> round(augsa,2)
```

```
[1] 41.5
```

Secinājums: 99% ticamības intervāls niedru lapu garuma vidējam aritmētiskajam ir ni 33,69 līdz 41,50 cm.

Ticamības intervālu vidējām aritmētiskajam var aprēķināt arī ar funkciju `t.test()`, kurai kā argumentus norāda kolonnu/vektoru, kuram jāaprēķina ticamības intervāls, kā arī vēlamais ticamības līmenis (`conf.level=`). Funkcijas rezultātu ir vēlams saglabāt kā atsevišķu objektu. Ar funkciju `names()` apskatot objekta struktūru nosaukumus, var redzēt, ka ir atsevišķs objekts ar nosaukumu `conf.int`. Tam var atlasīt izmantojot `$` zīmi. Iegūtais rezultāts ir identisks tam, kuru ieguva izmantojot formulas.

```

> tests<-t.test(niedr$garums,conf.level=0.99)
> names(tests)

[1] "statistic"    "parameter"    "p.value"      "conf.int"     "estimate"
[6] "null.value"   "alternative"  "method"       "data.name"

> round(tests$conf.int,2)

[1] 33.69 41.50
attr(,"conf.level")
[1] 0.99

```

4.2.1 Bootstrap ticamības intervāli

Atsevišķos gadījumos ticamības intervālus nepieciešams aprēķināt rādītājiem, kuriem nav izstrādātā formula, vai arī zināmiem rādītājiem nevar izmantot parastos aprēķina veidus, jo tiek pārkāpti kādi pieņēmumi par datiem, piemēram, to homogenitāti. Šādos gadījumos var izmantot tā saukto bootstrap metodi. Metodes pamātā ir tas, ka no esošajiem datiem veido daudzas jaunas paraugkopas, kuru apjoms ir vienāds ar oriģinālajiem datiem. Jaunu paraugkopu veidošana notiek pēc paraugošanas ar aizvietošanu principa, tas ir, katra no oriģinālajām vērtībām var tik izraudzīta vairāk kā vienu reizi. Pēc tam katrai no jaunajām paraugkopām aprēķina interesējošo rādītāju un tad aprēķina ticamības intervālu, piemēram, balstoties uz procentilēm.

Lai izmantotu bootstrap metodi programmā R, ir izveidota tam īpaši paredzēta pakete *boot* (Canty & Ripley, 2012). Piemērā aprēķināts ticamības intervāls tiem pašiem niedru lapu garumiem. Sākumā ar funkciju `set.seed()` tiek panākts, lai rezultāts būtu identisks piemēram. Tālāk ir jānodēfinē funkcija, kas aprēķina vēlamo rādītāju. Šo funkciju nosaucam par `videjie` un tai būs divi mainīgie – *data* un *indices*, kuru radīs funkcija `boot()`. *indices* norādīs indeksus, lai atlasītu skaitļus no mainīgā *data*. Jaunie skaitļi tiks saglabāti mainīgajā *d* un šim mainīgajam aprēķinās vidējo aritmētisko. Tālāk izmanto funkciju `boot()`, kurai kā argumentus norādā oriģinālo skaitļu rindu/kolonnas, aprēķināmo rādītāju (funkcija `videjie`), kā arī veidojamo paraugu skaitu (šoreiz 1000).

```

> set.seed(1234)
> library(boot)
> videjie<-function(data,indices){
+   d<-data[indices]
+   mean(d)
+ }
> dat<-boot(data=niedr$garums,statistic=videjie,R=1000)

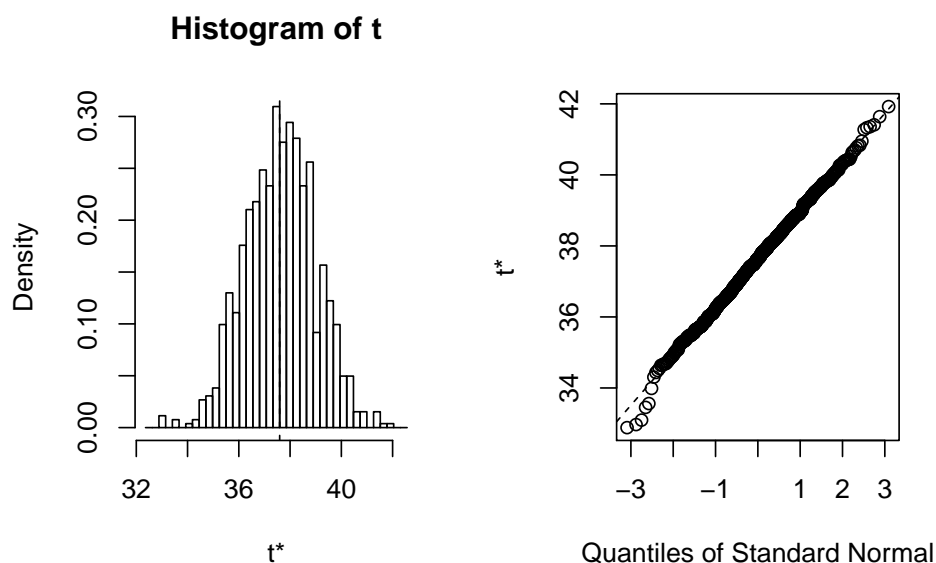
```

Ar funkciju `boot()` izveidoto objektu var apskatīt arī grafiski (4.1 attēls) – redzams, ka iegūtie vidējie aritmētiskie apmēram veido normālo sadalījumu

```

> plot(dat)

```

Att. 4.1: Bootstrap vidējo aritmētisko histogramma un QQ grafiks

Bootstrap ticamības intervālu aprēķināšanai izmanto funkciju `boot.ci()`, kurai kā argumentus norāda `boot()` izveidoto objektu, kā arī vēlamo ticamības līmeni (`conf=`). Funkcija aprēķina vairākus ticamības intervāla veidus (Manly, 2007), kas dod samērā līdzīgus rezultātus.

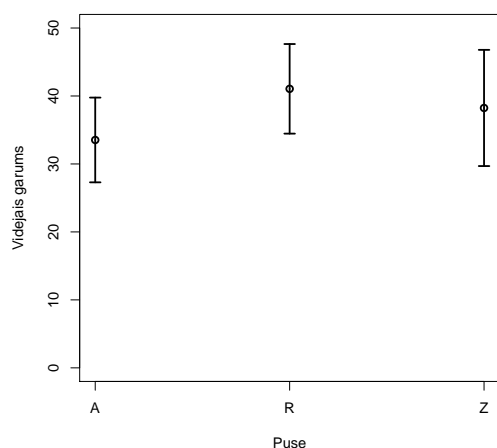
```
> boot.ci(dat,conf=0.99)

BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 1000 bootstrap replicates

CALL :
boot.ci(boot.out = dat, conf = 0.99)

Intervals :
Level      Normal              Basic
99%   (34.04, 41.09 )   (33.86, 41.62 )

Level      Percentile          BCa
99%   (33.56, 41.33 )   (33.39, 41.11 )
Calculations and Intervals on Original Scale
Some basic intervals may be unstable
Some percentile intervals may be unstable
Some BCa intervals may be unstable
```



Att. 4.2: Vidējais niedru lapu garums trīs parauglaukumos ar 99% ticamības intervāliem

4.2.2 Ticamības intervālu grafiskais attēlojums

Izmantojot funkcijas, kas pieejamas paketē *plotrix* (Lemon, 2006), ir iespējams izveidot grafiku ar ticamības intervāliem. Pirmkārt, ir nepieciešams objekts, kas satur vidējos aritmētiskos. Piemēram izmantots niedru lapu garums, kas aprēķināts katram no parauglaukumiem.

```
> videjie<-tapply(niedr$garums,niedr$paraug,mean)
> videjie
```

```
    Austr    Riet    Ziem
33.52941 41.05294 38.23750
```

Pēc tam jāizveido objekts, kas satur skaitļus Stjūdenta kritērija un vidējā aritmētiskā reprezentācijas rādītāja reizinājumus. Tam izmanto atkal funkciju `tapply`, tikai beigās norāda paša definētu funkciju.

```
> vid.rob<-tapply(niedr$garums,niedr$paraug,
+ function(x) (qt((1-0.01/2),(length(x)-1))*sd(x)/sqrt(length(x))))
> vid.rob
```

```
    Austr    Riet    Ziem
6.283364 6.642022 8.623851
```

Grafika ar ticamības intervāliem izveidošanai (4.2 attēls) jāizmanto funkcija `plotCI()`, kurai kā argumentus jānorāda `x` mainīgais (šoreiz skaitļi no 1 līdz 3, jo ir trīs parauglaukumi), pēc tam jānorāda vektors ar vidējiem aritmētiskajiem, vektors ar Stjūdenta kritērija un vidējā arit-

mētiskā reprezentācijas rādītāja reizinājumiem, kā arī var norādīt papildus argumentus grafika izskata uzlabošanai.

```
> library(plotrix)
> plotCI(1:3,as.vector(videjie),as.vector(vid.rob),axes=F,xlab="Puse",
+ ylab="Videjais garums",ylim=c(0,50),lwd=2)
> axis(1,at=1:3,labels=c("A","R","Z"))
> axis(2,at=seq(0,50,10))
> box()
```

Nodaļa 5

Paraugkopu salīdzināšana

5.1 Datu sagatavošana

Kā pirmais fails piemēriem izmantots *niedres2.txt*, kas satur informāciju par niedru lapu garumu un platumu trīs parauglaukumos – divos parauglaukumos ir pa 17 novērojumiem, bet vienā parauglaukumā ir 16 novērojumi.

```
> niedr<-read.table(file="niedres2.txt",header=T,sep="\t",dec=".")
> str(niedr)

'data.frame':      50 obs. of  3 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num   2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
 $ paraug : Factor w/ 3 levels "Austr","Riet",...: 1 1 1 1 1 1 1 1 1 1 ...

> summary(niedr)

      garums      platums      paraug
Min.   :11.90   Min.   :0.400   Austr:17
1st Qu.:31.07   1st Qu.:2.500   Riet :17
Median :38.40   Median :3.900   Ziem :16
Mean   :37.59   Mean   :3.892
3rd Qu.:45.30   3rd Qu.:5.100
Max.   :59.00   Max.   :7.100
```

No datu objekta *niedr* izveidots otrs objekts *niedr2*, kas satur novērojumus tikai no Austrumu un Rietumu parauglaukumiem. Tomēr apskatot ar funkciju `str()` jaunizveidotās datu tabulas struktūru redzams, ka joprojām tiek uzskatīts, ka kolonnā *paraug* ir trīs iespējamie līmeņi. Lai "nomestu" nevajadzīgo faktora līmeni, jāizmanto funkcija `droplevels()`. Pēc šīs darbības kolonnai *paraug* paliek tikai divi līmeņi.

```
> niedr2<-niedr[niedr$paraug=="Austr" | niedr$paraug=="Riet",]
> str(niedr2)
```

```
'data.frame':      34 obs. of  3 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num   2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
 $ paraug  : Factor w/ 3 levels "Austr","Riet",...: 1 1 1 1 1 1 1 1 1 1 ...

> niedr2<-droplevels(niedr2)
> str(niedr2)

'data.frame':      34 obs. of  3 variables:
 $ garums : num  31.6 23.2 39.2 37.4 21.1 37 24.7 31.3 37.4 39.7 ...
 $ platums: num   2.5 2.3 2.1 5.8 2.2 4.1 3.5 4.2 2.5 2.8 ...
 $ paraug  : Factor w/ 2 levels "Austr","Riet": 1 1 1 1 1 1 1 1 1 1 ...
```

5.2 Divu neatkarīgu paraugkopu salīdzināšana ar parametriskajām metodēm

Paraugkopu salīdzināšanas piemēram izmantota informācija par niedru lapu garumu divosa parauglaukumos. Pirmais solis paraugkoku salīdzināšanai ir to dispersiju salīdzināšana. Tam var izmantot funkciju `var.test()`, kas veic F testu divu paraugkopu dispersiju salīdzināšanai. Funkcijai kā argumenti jānorāda kolonna, kurā ir visi dati, un kolonna, kurā ir dalījuma līmeņi. Analīzes rezultātos parādās gan F vērtība (dispersiju dalījums), gan arī ticamības intervāls šim dalījumumam, brīvības pakāpju skaits un atbilstošā p-vērtība.

```
> var.test(niedr2$garums~niedr2$paraug)

      F test to compare two variances

data:  niedr2$garums by niedr2$paraug
F = 0.8949, num df = 16, denom df = 16, p-value = 0.827
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3240865 2.4711930
sample estimates:
ratio of variances
 0.8949191
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ nav statistiski būtiskas atšķirības starp abu paraugkopu dispersijām, jo p-vērtība ir lielāka par noteikto būtiskuma līmeni ($0,826 > 0,05$).

Vidējo aritmētisko salīdzināšanai izmanto funkciju `t.test()`, kurai arī kā argumenti jānorāda kolonna, kurā ir visi dati, un kolonna, kurā ir dalījuma līmeņi. Papildus jānorāda arī arguments `var.equal=TRUE`, lai norādītu, ka dispersijas ir homogēnas (par to pārlicinājās iepriekšējā testā.). Rezultātos norāda t-vērtību, brīvības pakāpju skaitu, atbilstošo p-vērtību. Tā pat tiek norādīts abu paraugkopu vidējais aritmētiskais, kā arī šo vidējo aritmētisko starpības ticamības intervāls.

```
> t.test(niedr2$garums~niedr2$paraug, var.equal=TRUE)

Two Sample t-test

data: niedr2$garums by niedr2$paraug
t = -2.4034, df = 32, p-value = 0.02221
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.899897  -1.147161
sample estimates:
mean in group Austr  mean in group Riet
      33.52941         41.05294
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ pastāv statistiski būtiska atšķirība starp abu paraugkopu vidējiem aritmētiskajiem, jo iegūtā p-vērtība ir zemāka par noteikto būtiskuma līmeni ($0,022 < 0,050$).

T-testu programmā R var arī veikt nepieņemot dispersiju vienādību (tas pat ir noklusētais uzstādījums). Šajā gadījumā t-tests tiek veikts ar Velča korekciju (mainās brīvības pakāpju skaits).

```
> t.test(niedr2$garums~niedr2$paraug)

Welch Two Sample t-test

data: niedr2$garums by niedr2$paraug
t = -2.4034, df = 31.902, p-value = 0.02223
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.900666  -1.146392
sample estimates:
mean in group Austr  mean in group Riet
      33.52941         41.05294
```

5.3 Vairāku neatkarīgu paraugkopu salīdzināšana ar parametriskajām metodēm

Vairāk kā divu paraugkopu salīdzināšanas piemēram izmantota oriģinālā *niedr* datu tabula, kurā bija visi trīs parauglaukumi. Vairāku dispersiju salīdzināšanai var izmantot funkciju `leveneTest()` no paketes `car`. Arī šai funkcijai kā mainīgie jānorāda kolonna ar datiem un kolonna ar dalījuma līmeņiem. Analīzes rezultāts atšķiras no tā, ko dot `var.test()`. Rezultātā ir norādīts brīvības pakāpju skaits, iegūtā F vērtība un atbilstošā p-vērtība (norādīta kolonnā Pr(>F)).

```

> library(car)
> leveneTest(niedr$platums~niedr$paraug)

Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  2  0.7235 0.4904
      47

```

Secinājums: visi trīs paraugkopu dispersijas ir homogēnas (atšķirības nav būtiskas) pie būtiskuma līmeņa $\alpha = 0,05$, jo p-vērtības ir lielāka par norādīto būtiskuma līmeņa ($0,49 > 0,05$).

Vidējo aritmētisko salīdzināšanai atkal jāizmanto t-tests, tikai šajā gadījumā jāizmanto funkcija `pairwise.t.test()`, kas veiks t-testus starp jebkuriem diviem līmeņiem (parauglaukumiem), kā arī vienlaicīgi pielāgos p-vērtības, lai samazinātu 1. tipa kļūdas rašanās iespēju veicot atkārtotos testus. Vēl viena šīs funkcijas īpatnība ir tā, ka mainīgo atdalīšanai izmanto komatu nevis tildes zīmi. Analīzes rezultātos norādītas p-vērtības katram no salīdzinājumiem, kā arī norādīta kāda p-vērtību pielāgošanas metode izmantota (to var mainīt).

```

> pairwise.t.test(niedr$platums,niedr$paraug)

Pairwise comparisons using t tests with pooled SD

data:  niedr$platums and niedr$paraug

      Austr Riet
Riet 0.013 -
Ziem 0.016 0.859

P value adjustment method: holm

```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ pastāv statistiski būtiska atšķirība starp Austrumu un Rietumu, kā arī Austrumu un Ziemeļu parauglaukumu lapu platumu vidējām vērtībām, jo atbilstošās p-vērtības ir mazākas par noteikto būtiskuma līmeni ($0,013 < 0,050$ un $0,016 < 0,050$), toties nav statistiski būtiskas atšķirības starp Rietumu un Ziemeļu parauglaukumu lapu platumiem ($0,859 > 0,050$).

5.4 Atkarīgu paraugkopu salīdzināšana ar parametriskām metodēm

Kā piemērs atkarīgu paraugkopu salīdzināšanai izmantots datu fails *rokas.txt*, kas satur informāciju par labās un kreisās rokas spēku 25 cilvēkiem. Tā kā ir pieejama informācija par vienu un tā paša cilvēkā labās un kreisās rokas spēku, tas šādus datus attiecīgi pieņemam par atkarīgiem.

Ja dati atbilst visiem pieņēmumiem, tad var izmantot T-testu modifikāciju atkarīgiem datiem. Programmā R tam izmanto atkal funkciju `t.test()`, kurai kā arguments jānorāda divas

datu kolonnas, kā arī `paired=TRUE`. Galvenais nosacījums, lai datu objekts būtu sagatavots pareizi – abās kolonnās objektu secībai jābūt vienādai.

```
> rokas<-read.table(file="rokas.txt",header=TRUE,sep="\t",dec=".")
> str(rokas)

'data.frame':      25 obs. of  2 variables:
 $ laba  : num  32.9 39.7 26.9 38.8 58 50.6 31.4 68.7 65.2 31.8 ...
 $ kreisa: num  21.9 39.2 29.9 34.8 48.7 51.4 34.5 56.4 57.5 27.8 ...

> t.test(rokas$laba,rokas$kreisa,paired=TRUE)

      Paired t-test

data:  rokas$laba and rokas$kreisa
t = 4.4743, df = 24, p-value = 0.0001581
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 2.413454 6.546546
sample estimates:
mean of the differences
          4.48
```

Secinājums: starpība starp labās un kreisās rokas vidējo spēku ir 4.48 un šī starpība ir statistiski būtiska pie būtiskuma līmeņa $\alpha = 0,05$, jo p-vērtība ir mazāka par šo būtiskuma līmeni ($0,0002 < 0,05$).

Ja šos pašus datus analizē nepieņemot to atkarību, iegūst pavisam citu rezultātu – šādā gadījumā secinājums ir, ka starpība starp vidējiem aritmētiskajiem nav statistiski būtiska.

```
> t.test(rokas$laba,rokas$kreisa)

      Welch Two Sample t-test

data:  rokas$laba and rokas$kreisa
t = 1.1261, df = 47.021, p-value = 0.2658
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
-3.523473 12.483473
sample estimates:
mean of x mean of y
 44.812    40.332
```


5.5 Vienas paraugkopas t-tests

Funkciju `t.test()` var izmantot arī vienas paraugkopas t-testam, piemēram, ja jānoskaidro vai paraugkopa var piederēt pie ģenerālkopas, kurai ir zināms tās vidējais aritmētiskais. Šādā gadījumā kā argumenti funkcijai jānorāda vektors/kolonna, kas satur paraugkopas datus, kā arī arguments `mu=` ar ģenerālkopas vidējo aritmētisko. Piemērā pārbaudīts vai niedru lapu platums varētu piederēt pie ģenerālkopas, kuras vidējā vērtība ir trīs.

```
> t.test(niedr$platums,mu=3.0)

      One Sample t-test

data:  niedr$platums
t = 3.9431, df = 49, p-value = 0.0002557
alternative hypothesis: true mean is not equal to 3
95 percent confidence interval:
 3.4374 4.3466
sample estimates:
mean of x
 3.892
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ niedru lapu platums konkrētajā paraugkopā nepieder pie ģenerālkopas, kuras vidējais aritmētiskais ir 3,0 cm, jo p-vērtības ir mazāka par noteikto būtiskuma līmeni ($0,0026 < 0,05$).

5.6 Divu neatkarīgu paraugkopu salīdzināšana ar neparametriskajām metodēm

Lai veiktu divu paraugkopu salīdzināšanu ar neparametriskām metodēm programmā R, ir jāizmanto funkciju `wilcox.test()`, kurai kā argumenti jānorāda kolonna ar datiem un kolonna ar dalījuma līmeņiem (tie drīkst būt divi). Piemēram izmantoti dati no datu objekta `niedr2` par niedru lapu garumu divos parauglaukumos. Atšķirībā no funkcijas `t.test()` rezultātiem, šajā gadījumā norāda tikai iegūtu p-vērtību un W rādītāju, ko izmanto atšķirības būtiskuma novērtēšanai.

```
> wilcox.test(niedr2$garums~niedr2$paraug)

      Wilcoxon rank sum test with continuity correction

data:  niedr2$garums by niedr2$paraug
W = 75, p-value = 0.01745
alternative hypothesis: true location shift is not equal to 0
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ pastāv statistiski būtiska atšķirība starp abu

paraugkopu niedru lapu garumiem, jo iegūtā p-vērtība ir zemāka par noteikto būtiskuma līmeni ($0,017 < 0,050$).

5.7 Divu atkarīgu paraugkopu salīdzināšana ar neparametriskajām metodēm

Atkarīgu paraugkopu salīdzināšanai ar neparametriskajām metodēm R arī izmanto funkciju `wilcox.test()`, kurai kā arguments jānorāda divas datu kolonnas, kā arī `paired=TRUE`. Galvenais nosacījums, lai datu objekts būtu sagatavots pareizi – abās kolonnās objektu secībai jābūt vienādai. Piemēram izmantoti dati par labās un kreisās rokas spēku.

```
> wilcox.test(rokas$laba,rokas$kreisa,paired=TRUE)

      Wilcoxon signed rank test with continuity correction

data:  rokas$laba and rokas$kreisa
V = 291.5, p-value = 0.0005445
alternative hypothesis: true location shift is not equal to 0
```

Secinājums: starpība starp labās un kreisās rokas spēku ir statistiski būtiska pie būtiskuma līmeņa $\alpha = 0,05$, jo p-vērtība ir mazāka par šo būtiskuma līmeni ($0,0006 < 0,05$).

5.8 χ^2 tests

χ^2 testa veikšanai programmā R izmanto funkciju `chisq.test()`, kurai norādāmie argumenti ir atkarīgi no tā, ko vēlas salīdzināt. Ja ir nepieciešams pārbaudīt vai empīriskais sadalījums atbilst iepriekš zināmajam teorētiskajam sadalījumam, tad jānorāda argumenti `x=`, kas satur empīriskos frekvenču vektoru, un arguments `p=`, kas satur teorētiskā sadalījuma relatīvās frekvences (iespējamības). Teorētiskā sadalījuma frekvenču vektora summai jābūt vienādai ar 1.

Kā piemērs pārbaudīta eksperimentā iegūto dažādu formu un krāsu zirņu skaita sadalījuma atbilstība sagaidāmajam sadalījumam atbilstoši Mendēļa likumiem. Sagaidāmās frekvences ir 9:3:3:1, un, lai panāktu, ka to summa būtu vienāda ar 1, tās tiek dalītas ar 16 (sagaidāmo frekvenču summa). Analīzes rezultātos parādās aprēķinātā χ^2 vērtība, brīvības pakāpju skaits un atbilstošā p-vērtība.

```
> zirni <- c(315, 108, 101, 32)
> teor.zirni <- c(9, 3, 3, 1)/16
> chisq.test(x = zirni, p = teor.zirni)

      Chi-squared test for given probabilities

data:  zirni
X-squared = 0.47, df = 3, p-value = 0.9254
```

Tabula 5.1: Koku skaits pa sugām divos parauglaukumos

Suga	Parauglaukums A	Parauglaukums B
Priede	12	27
Egle	34	33
Bērzs	56	47
Ozols	23	14
Kļava	8	11

Secinājums: eksperimentāli iegūto zirņu attiecība atbilst teorētiski sagaidāmajam sadalījumam (nav statistiski būtiskas atšķirības), jo iegūtā p-vērtība ir lielāka par būtiskuma līmeni ($0,925 > 0,05$).

Ja ir nepieciešams salīdzināt frekvenču sadalījumu starp divām paraugkopām, tad funkcijā `chisq.test()` kā arguments jānorāda matrice, kurā atbilstoši pa kolonnām izvietoti pirmās un otrās paraugkopas dati. Analīzes piemēram izmantoti dati par koku sadalījumu pa sugām divos parauglaukumos (5.1 tabula).

```
> koki<-matrix(c(12,34,56,23,8,27,33,47,14,11),ncol=2)
> chisq.test(koki)
```

Pearson's Chi-squared test

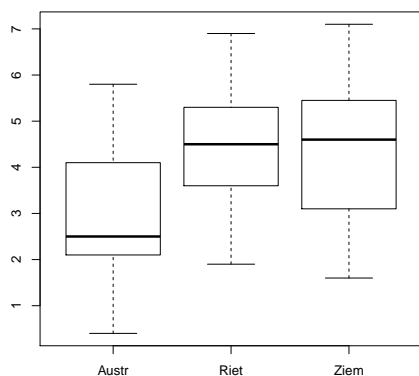
```
data: koki
X-squared = 9.2298, df = 4, p-value = 0.05561
```

Secinājums: pie būtiskuma līmeņa $\alpha = 0,05$ nav statistiski būtiskas atšķirības koku skaita sadalījumā pa sugām divos parauglaukumos, jo p-vērtība ir lielāka par noteikto būtiskuma līmeni ($0,056 > 0,05$).

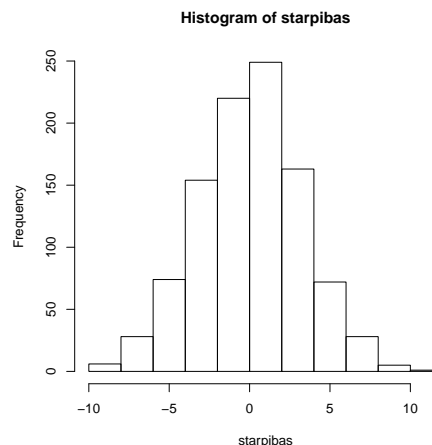
5.9 Paraugkopu grafiskā salīdzināšana

Viens no informatīvākajiem grafikiem vairāku paraugkopu grafiskai salīdzināšanai ir Box-plot, kuru iegūst ar funkciju `boxplot()`. Kā argumenti šai funkcijai jānorāda kolonna ar datiem un kolonna ar dalījuma līmeņiem. Pēc šāda grafika var spriest gan par datu izkliedi atsevišķiem dalījuma līmeņiem, gan arī par vērtību līdzību vai atšķirību (5.1 attēls). Tomēr jāatceras, ka pilnīgi drošu secinājumu var izdarīt tikai veicot kādu statistisko testu.

```
> boxplot(niedr$platums~niedr$paraug)
```



Att. 5.1: Niedru lapu platuma box-plot grafiks trīs parauglaukumiem



Att. 5.2: Permutāciju testa rezultātā iegūto vidējo aritmētisko starpības histogramma

5.10 Permutāciju jeb randomizēšanas tests paraugkopu salīdzināšanai

Gadījumos, kad nav iespējams veikt tradicionālos testus paraugkopu salīdzināšanai, vai arī ir nepieciešams salīdzināt rādītājus, kuru salīdzināšanai nav pieejams gatavs tests, ir iespējams izmantot tā sauktos permutāciju jeb randomizēšanas testus. Šo testu pamatdoma ir, ka divu esošo paraugkopu datus apvieno, un tad no šiem apvienotajiem datiem veido jaunas paraugkopas pēc nejaušības principa. Pēc tam tiek aprēķināts interesējošais rādītājs. Šis process tiek atkārtots daudz reizes (vismaz 200, labāk 1000 vai vēl vairāk reizes), un tad novērtē cik gadījumos iegūst tik pat ekstrēmu vai pat ekstrēmāku rādītāju kā oriģinālajos datos. Ja gadījumu skaita proporcija zemāka nekā izvēlētais būtiskuma līmenis, tad varam pieņemt, ka pastāv statistiski būtiska atšķirība.

Kā piemērs izmantots vienkāršākais gadījums – vidējo aritmētisko salīdzināšana, skatoties uz to starpību. Sākotnēji izveidotas divas paraugkopas *par1* un *par2*, kas satur informāciju par niedru lapu platumu Austrumu un Rietumu parauglaukumos. Katrā no paraugkopām ir 17 novērojumi. Veicot tradicionālo t-testu, noskaidrots, ka vidējie aritmētiskie būtiski atšķiras un vidējo aritmētisko starpība ir -7,52.

```
> par1<-niedr$garums[niedr$paraug=="Austr"]
> par2<-niedr$garums[niedr$paraug=="Riet"]
> t.test(par1,par2)
```

Welch Two Sample t-test

```
data: par1 and par2
t = -2.4034, df = 31.902, p-value = 0.02223
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -13.900666 -1.146392
sample estimates:
```

```

mean of x mean of y
 33.52941  41.05294

> vid.st<-mean(par1)-mean(par2)
> vid.st

[1] -7.523529

```

Permutāciju testa veikšanai, pirmkārt, izveido vektoru *kop*, kas satur abu paraugkopu datus. Otrkārt, izveido vektoru *starpibas*, kurā apvienos iegūtās vidējo aritmētisko starpības. Nākamais solis ir izmantot funkciju `for()`, lai veikti 999 permutācijas (jo viens atkārtojums jau oriģinālā starpība) – ar funkciju `sample()` izvēlas 17 nejaušus skaitļus no 1 līdz 34, kas apzīmēs indeksus. Pēc tam aprēķina vidējos aritmētiskos divām jaunām paraugkopām *vid1* un *vid2* no apvienotā skaitļu vektora *paraug*, kas attiecīgi satur izvēlētos 17 skaitļus un pārējos 17 skaitļus. Pēc tam aprēķina vidējo aritmētisko starpību un saglabā vektorā *starpibas*. Beigās kā 1000. novērojumu pievieno oriģinālo vidējo aritmētisko starpību.

```

> kopa<-c(par1,par2)
> starpibas<-NULL
> for(i in 1:999){
+   paraug<-sample(1:34,17)
+   vid1<-mean(kopa[paraug])
+   vid2<-mean(kopa[-paraug])
+   starpibas[i]<-vid1-vid2
+ }
> starpibas[1000]<-vid.st

```

Apskatot iegūto vidējo aritmētisko starpību histogrammu (5.2 attēls), varam redzēt, ka tā veido apmēram normālo sadalījumu, turklāt iespējamās vērtības mēdz būt lielākas nekā oriģinālojos datos.

```

> hist(starpibas)

```

Pēdējais solis ir novērtēt cik bieži ir iegūta tik pat ekstrēma vai ekstrēmāka starpība. Tam izmanto funkciju `mean()`, kurai kā arguments likts loģiskais vaicājums – vai vektora *starpibas* vērtības ir lielākas vai vienādas ar oriģinālo vidējo aritmētisko starpību. Loģiskā vaicājuma gadījumā iegūst vektoru, kas satur TRUE vai FALSE vērtības, un funkcija `mean()` aprēķina proporciju TRUE vērtībām. Papildus izmantota funkcija `abs()`, lai skatītu absolūtās starpības un veiktu divpusējo testu. Tā kā iegūtā proporcija ir 0,017, tad pie būtiskuma līmeņa $\alpha = 0,05$ varam secināt, ka pastāv statistiski būtiska atšķirība vidējos aritmētiskajos.

```
> mean(abs(starpibas) >= abs(vid.st))
```

```
[1] 0.017
```

Iegūtais rezultāts katrā gadījumā var atšķirties, jo paraugkopu veidošanā iesaistīts nejaušības princips.

Nodaļa 6

Dispersijas analīze

6.1 Dati

Dispersijas analīzes piemēram izmantosim datu failu *augi.txt*, kas satur informāciju par augu biomasu parauglaukumā eksperimentā, kurā pētīta barības vielu daudzuma (*barv*) un ūdens daudzuma (*udens*) ietekme. Barības vielu daudzumam ir četras iespējamās grupas: kontrole (*kontr*), papildus fosfors(*fosf*), papildus slāpekļis (*slap*) un šo abu elementu kombinācija (*slfo*). Faktoram ūdens ir divi līmeņi: kontrole (*norm*) un papildus ūdens (*papild*).

```
> augi<-read.table(file="augi.txt",header=TRUE,sep="\t",dec=".")
> str(augi)

'data.frame':      40 obs. of  3 variables:
 $ biomasa: num  60.8 61.3 49.6 53.8 54.1 47.6 63.7 50.3 50.9 55.7 ...
 $ barv   : Factor w/ 4 levels "fosf","kontr",...: 2 2 2 2 2 2 2 2 2 2 ...
 $ udens  : Factor w/ 2 levels "norm","papild": 1 1 1 1 1 2 2 2 2 2 ...

> summary(augi)

      biomasa      barv      udens
Min.   :47.60   fosf :10   norm  :20
1st Qu.:55.67   kontr:10  papild:20
Median :59.20   slap  :10
Mean   :59.66   slfo  :10
3rd Qu.:63.70
Max.   :74.90
```

Lai gūtu sākotnējo priekšstatu par augu biomasas atšķirībām starp katra no faktoriem gradācijas klasēm, var izmantot funkciju `tapply()`, kurai kā argumentus norādam kolonnu, kas satur datus, kolonnu, kas satur dalījuma līmeņus, kā arī rādītāju, kuru vēlamies aprēķināt. Šo funkciju pielietojam, lai apskatītu kāda ir vidējā aritmētiskā augu biomasu katrā no barības vielu un katrā no ūdens režīmiem.

```
> with(augi, (tapply(biomasa, barv, mean)))
```

```

fosf kontr  slap  slfo
61.04 54.78 57.44 65.37

> with(augi, (tapply(biomasa, udens, mean)))

norm papild
59.755 59.560

```

6.2 Dispersiju homogenitāte

Tā kā viens no dispersijas analīzes pieņēmumiem ir, ka dispersijas starp gradācijas līmeņiem ir homogēnas, tas ir, to atšķirība nav statistiski būtiska, ir jāveic dispersiju salīdzināšana. Viens no testiem, ko var izmantot, ir Bartletta tests, ko veic ar funkciju `bartlett.test()`. Šai funkcijai kā argumenti jānorāda kolonna, kas satur datus, kā arī kolonna, kas satur dalījuma līmeņus. Analīzes rezultātos parādās aprēķinātā K^2 vērtība, brīvības pakāpju skaits un p-vērtība.

```

> with(augi, bartlett.test(biomasa, barv))

      Bartlett test of homogeneity of variances

data:  biomasa and barv
Bartlett's K-squared = 4.4872, df = 3, p-value = 0.2134

Secinājums: tā kā iegūtā p-vērtība (0.2134) ir lielāka par būtiskuma līmeni ( $\alpha = 0.05$ ), tad atšķirība starp dispersijām nav statistiski būtiska jeb dispersijas ir homogēnas.

> with(augi, bartlett.test(biomasa, udens))

      Bartlett test of homogeneity of variances

data:  biomasa and udens
Bartlett's K-squared = 1.4017, df = 1, p-value = 0.2364

```

Secinājums: tā kā iegūtā p-vērtība (0.2364) ir lielāka par būtiskuma līmeni ($\alpha = 0.05$), tad atšķirība starp dispersijām nav statistiski būtiska jeb dispersijas ir homogēnas.

6.3 Modeļu definēšana

Pašas dispersijas analīzes veikšanai var izmantot vairākas funkcijas, no kurām viena ir funkcija `aov()`. Šai funkcijai kā argumenti jānorāda analizējamais mainīgais (šajā gadījumā *biomasa*), tad ir tildes zīme un viens vai vairāki ietekmējošie faktori. Ja jāpārbauda tikai faktoru atsevišķa ietekme, tad tos jāatdala ar plus zīmi, bet ja ir nepieciešams pārbaudīt arī faktoru kombinācijas ietekmi, tad faktorus var atdalīt ar zvaigznītes zīmi. Beigās jānorāda arī no kuras tabulas šos datus ņemt.

Dispersijas analizē ir svarīgi, lai programma faktoru uztvertu tiešām kā faktoru. Ja gradācijas līmeņi ir nosaukti ar rakstu zīmēm, tad problēmu nav. Ja gradācijas līmeņi ir apzīmēti ar skaitļiem, tad pirms analīzes veikšanas šo mainīgo vajag pārvērst par faktoru ar funkcijas `as.factor()` palīdzību.

Analīzes rezultātā parādās katra faktora un atlikuma vērtību brīvības pakāpju skaits, noviržu kvadrātu summa un vidējā noviržu kvadrātu summa, atbilstošā F vērtība un p-vērtība ($\Pr(>F)$).

```
> mod1<-aov(biomasa~barv*udens,data=augi)
> summary(mod1)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
barv	3	632.5	210.84	9.834	9.52e-05	***
udens	1	0.4	0.38	0.018	0.895	
barv:udens	3	43.7	14.56	0.679	0.571	
Residuals	32	686.1	21.44			

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Secinājums: faktora barības vielu daudzums ietekme uz augu biomasu ir statistiski būtiska, jo iegūtā p-vērtība ir mazāka par noteikto būtiskuma līmeni ($\alpha = 0.05$), bet faktora ūdens un abu faktoru kombinācijas ietekme nav statistiski būtiska, jo iegūtās p-vērtības ir lielākas par būtiskuma līmeni.

Tā kā faktora ūdens un faktoru kombinācijas ietekme nav būtiska, tad varam izveidot arī vienkāršāku modeli, kurā ir atstāts tikai faktors barības vielu daudzums. Arī šajā modelī barības vielu daudzuma ietekme ir būtiska.

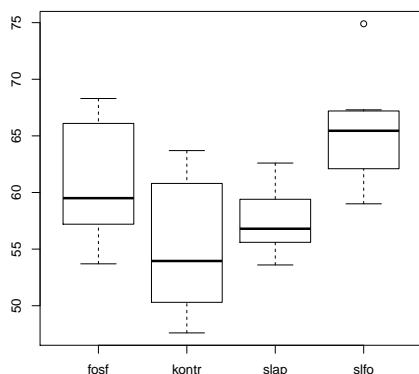
```
> mod2<-aov(biomasa~barv,data=augi)
> summary(mod2)
```

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
barv	3	632.5	210.84	10.39	4.54e-05	***
Residuals	36	730.1	20.28			

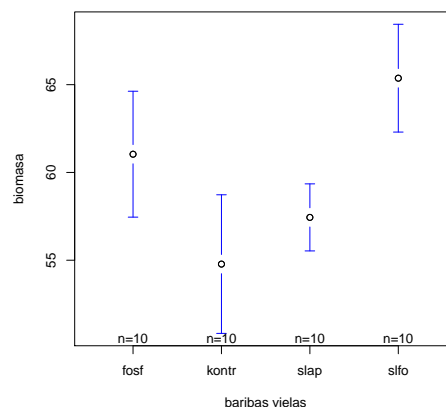
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

6.4 Gradācijas klašu salīdzināšana

Konstatējot kāda faktora būtisku ietekmi, var veikt tā saucamo post-hoc testu, tas ir, salīdzināt gradācijas klases savā starpā. Viena no funkcijām šādam testam ir **TukeyHSD**. Šai funkcijai kā argumenti jānorāda piemērotās dispersijas analīzes modelis, kā arī pēdējās jānorāda faktora nosaukums, kura gradācijas klases savā starpā vēlas salīdzināt. Analīzes rezultātos iegūst vidējo aritmētisko starpību starp katrām divām gradācijas klasēm, šo vidējo aritmētisko starpības ticamības intervālu un atbilstošo p-vērtību.



Att. 6.1: Gradācijas klašu box-plot attēls



Att. 6.2: Gradācijas klašu vidējo aritmētisko at-tēls

```
> TukeyHSD(mod2, "barv")

Tukey multiple comparisons of means
 95% family-wise confidence level

Fit: aov(formula = biomasa ~ barv, data = augi)

$barv
      diff       lwr       upr    p adj
kontr-fosf -6.26 -11.68427 -0.8357301 0.0183777
slap-fosf   -3.60  -9.02427  1.8242699 0.2958204
slfo-fosf    4.33  -1.09427  9.7542699 0.1570183
slap-kontr   2.66  -2.76427  8.0842699 0.5561038
slfo-kontr  10.59   5.16573 16.0142699 0.0000391
slfo-slap    7.93   2.50573 13.3542699 0.0019671
```

Secinājums: statistiski būtiska atšķirība pastāv starp gradācijas klasēm kontrole-fosfors, kontrole-fosfors/slāpekļis, kā arī slāpekļis-fosfors/slāpekļis, jo atbilstošās p-vērtības ir mazākas par būtiskuma līmeni. Pārējos gadījumos atšķirības starp gradācijas klašu vidējiem aritmētiskajiem nav statistiski būtiska, jo iegūtās p-vērtības ir lielākas par noteikto būtiskuma līmeni.

6.5 Rezultātu grafiskā attēlošana

Dispersijas analīzes rezultātu grafiskai attēlošanai var izmantot vairākus grafiku veidus, piemēram, box-plot grafiku apvienojumu, kur katrai gradācijas klasei ir savs mazais box-plot attēls (6.1 attēls).

```
> with(augi, boxplot(biomasa~barv))
```

Vēl viens variants ir izmantot funkciju `plotmeans()` no paketes `gplots`. Šī funkcija izveido grafiku, kurā attēlots katras gradācijas klases vidējais aritmētiskais un tā ticamības intervāls (6.2 attēls). Kā papildus arguments izmantots `connect=FALSE`, lai vidējie aritmētiskie netiktu savā starpā automātiski savienoti.

```
> library(gplots)
> with(augi, plotmeans(biomasa~barv, connect=FALSE, xlab="baribas vielas"))
```

Nodaļa 7

Korelācijas analīze

7.1 Grafiskā pārbaude

Pirms veikt analītiski korelācijas analīzi vienmēr ir ieteicams savus datus apskatīties grafiski, tādējādi ir iespējams pārliecināties vai datos nav kādas ekstrēmas vērtības, kā arī var gūt sākotnējo priekšstatu par to, kuras pazīmes ir savā starpā saistītas.

Piemēram izmantosim datu failu *smiltaji.txt*, kas satur informāciju par augsnes pH, atsegtas smilts un sūnu segumu parauglaukumos, kā arī šajos parauglaukumos konstatēto vaskulāro augu sugu skaitu.

```
> smiltaji<-read.table(file="smiltaji.txt",header=TRUE,sep="\t",dec=".")
```

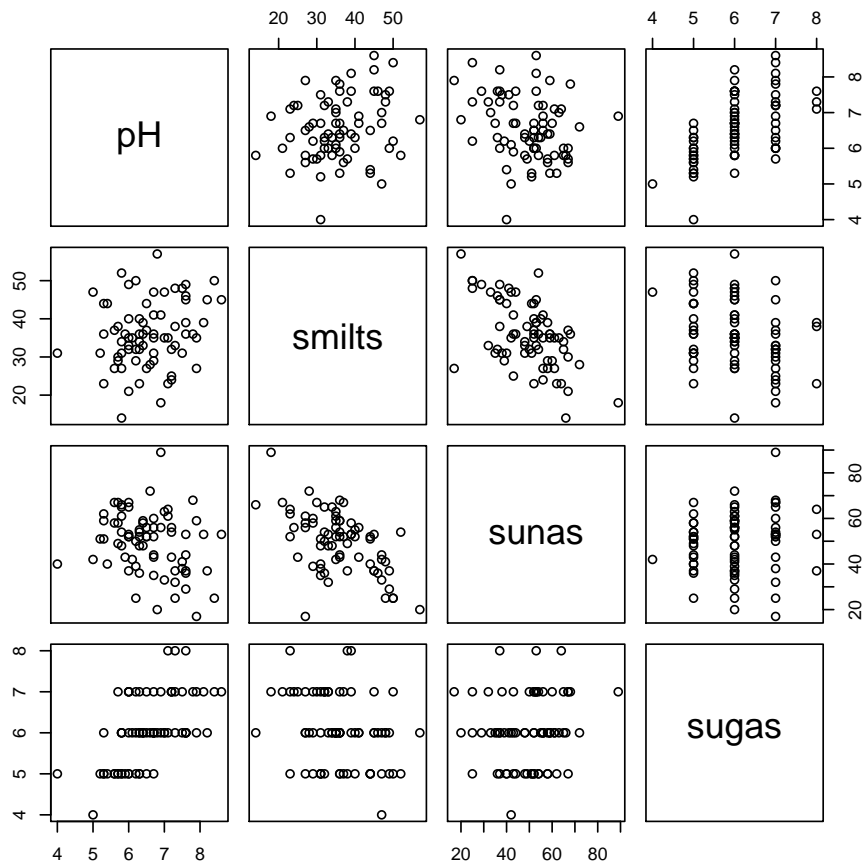
Tā kā šajā datu tabulā visās kolonnās ir skaitļi, tad izkliedes grafikus ir iespējams veidot visām kolonnām uzreiz. Tam noder funkcija `pairs()`, kas izveido izkliedes grafiku jebkurām divām no datu tabulas.

```
> pairs(smiltaji)
```

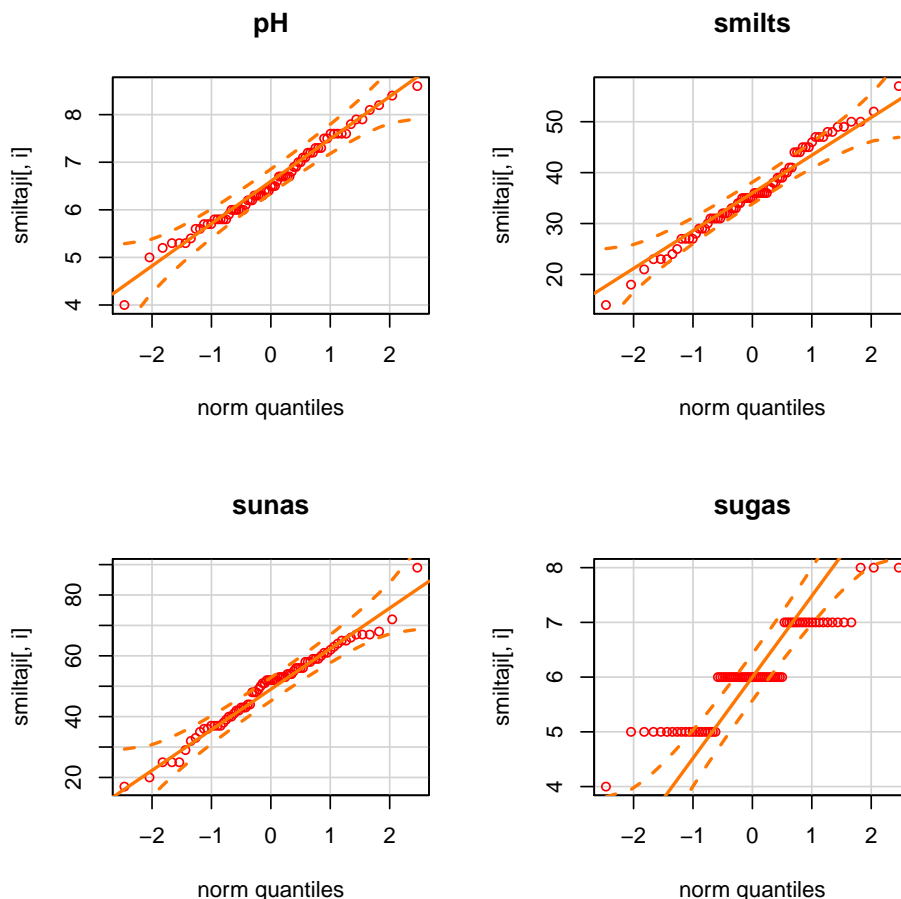
No attēla (7.1 attēls) redzams, ka ir saistība starp smilts un sūnu segumiem (negatīva), kā arī iespējams starp smilts segumu un augsnes pH. Attiecībā uz sugu skaitu veidojas divainas formas attēls, jo šim mainīgajam ir tikai piecas iespējamās vērtības.

Nākamais solis, ja tas jau nav izdarīts iepriekš, ir pārbaudīt vai dati atbilst normālajam sadalījumam. Šoreiz to pārbaudīsim grafiski katrai no kolonnām izveidojot QQ grafiku. Izmantojot funkciju `for()`, panākts, ka četras reizes nav jāraksta funkcija grafika zīmēšanai.

```
> par(mfrow=c(2,2))
> library(car)
> for(i in 1:4){
+ qqPlot(smiltaji[,i],main=names(smiltaji)[i])
+ }
```



Att. 7.1: Izkliedes grafiki starp visām smiltāju datu tabulas kolonnām



Att. 7.2: QQ grafiki smiltāju tabulas mainīgajiem

7.2 attēls skaidri parāda, ka mainīgais sugu skaits neatbilst normālajam sadalījumam, bet pārējiem trim mainīgajiem varam pieņemt atbilstību normālajam sadalījumam.

7.2 Pīrsona korelācijas analīze

Pamatfunkcijas korelācijas analīzes veikšanai programmā R ir `cor()` un `cor.test()`. Funkcijā `cor()` kā mainīgo var likt gan datu tabulu ar vairākām kolonnām, gan divas datu tabulas, ja nepieciešams noskaidrot korelāciju starp abu tabulu kolonnām. Rezultātā parādās matrice, kas satur korelācijas koeficientus starp jebkurām divām analizētajām kolonnām. Pēc noklusējuma funkcija aprēķina Pīrsona korelācijas koeficientu.

Piemērā ar tabulu `smiltaji` aprēķināti korelācijas koeficienti starp pirmajām trim kolonnām, jo ceturtās kolonnas dati neatbilda normālajam sadalījumam. Iegūtie korelācijas koeficienti ir robežās no -0.558 līdz 0.221 (korelācijas koeficients 1 mainīgajam pašam ar sevi). Diemžēl pēc funkcijas `cor()` rezultātiem nav iespējams secināt, kuri korelācijas koeficienti ir statistiski būtiski. Lai to izdarītu, ir jāizmanto specializētās tabulas vai arī papildus aprēķinus.

```
> cor(smiltaji[,1:3])
```

```
          pH      smilts      sunas
pH      1.0000000  0.2213315 -0.2517885
smilts  0.2213315  1.0000000 -0.5576408
sunas   -0.2517885 -0.5576408  1.0000000
```

Ja ir nepieciešams uzreiz aprēķināt gan korelācijas koeficientus, gan novērtēt to būtiskumu vairākam datu kolonnām, var izmantot funkciju `rcor.test` no paketes `ltm`. Rezultātā parādās matrica, kurā augšējā daļā ir korelācijas koeficienti, bet apakšējā daļā ir atbilstošās p-vērtības. Varam **secināt**, ka statistiski būtiska (pie $\alpha = 0.05$) korelācija pastāv starp pazīmēm pH un sūnu segums, kā arī smilts un sūnu segums, jo attiecīgi p vērtības ir 0,032 un <0,001. Korelācija starp pH un smilts segumu nav statistiski būtiska, kaut arī p-vērtība ir ļoti tuvu kritiskajai robežai (0,06).

```
> library(ltm)
> rcor.test(smiltaji[,1:3])
```

```
          pH      smilts sunas
pH      *****  0.221 -0.252
smilts  0.060 ***** -0.558
sunas   0.032 <0.001 *****
```

```
upper diagonal part contains correlation coefficient estimates
lower diagonal part contains corresponding p-values
```

Ar otru pamatfunkciju `cor.test()` var aprēķināt korelācijas koeficientu tikai starp divām kolonnām vai vektoriem (šajā gadījumā pH un sūnu segumu), toties analīzes rezultātos parādās gan pats korelācijas koeficients, gan arī tā ticamības intervāls, kā arī t-vērtības būtiskuma novērtēšanai un p-vērtība.

```
> with(smiltaji, cor.test(pH, sunas))
```

```
Pearson's product-moment correlation
```

```
data: pH and sunas
```

```
t = -2.1922, df = 71, p-value = 0.03164
```

```
alternative hypothesis: true correlation is not equal to 0
```

```
95 percent confidence interval:
```

```
-0.45547107 -0.02305682
```

```
sample estimates:
```

```
cor
```

```
-0.2517885
```

Secinājums: starp pazīmēm pH un sūnu segums ir statistiski būtiska korelācija pie būtiskuma līmeņa $\alpha = 0,05$.

7.3 Rangu korelācijas analīze

Spirmena un Kendela korelācijas analīzes veikšanai izmanto tās pašas funkcijas, kuras izmanto Pīrsona korelācijas analīzei: `cor()` un `cor.test()`. Vienīgās izmaiņas ir papildus arguments `method=` ar iespējamām vērtībām `'spearman'` un `'kendall'`. Ja izmanto funkciju `cor.test()`, tad analīzes rezultātos parādās korelācijas koeficients, rādītājs korelācijas koeficients būtiskuma novērtēšanai un iegūtā p-vērtība.

Piemēram izmantoti dati par sugu skaitu (dati neatbilda normālajam sadalījumam) un smilts segumu parauglaukumos.

```
> with(smiltaji, cor.test(sugas, smilts, method="spearman"))
```

```
      Spearman's rank correlation rho
```

```
data:  sugas and smilts
```

```
S = 83657.35, p-value = 0.01265
```

```
alternative hypothesis: true rho is not equal to 0
```

```
sample estimates:
```

```
      rho
```

```
-0.2905304
```

```
> with(smiltaji, cor.test(sugas, smilts, method="kendall"))
```

```
      Kendall's rank correlation tau
```

```
data:  sugas and smilts
```

```
z = -2.5457, p-value = 0.0109
```

```
alternative hypothesis: true tau is not equal to 0
```

```
sample estimates:
```

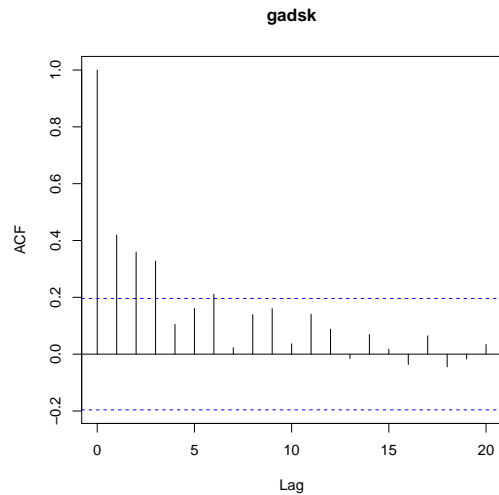
```
      tau
```

```
-0.2335632
```

Secinājums: gan ar Spirmena, gan ar Kendela korelācijas metodi starp pazīmēm sugu skaits un smilts segu pastāv statistiski būtiska negatīva korelācija (attiecīgi -0,29 un -0,23), jo iegūtās p-vērtības ir mazākas par noteikto būtiskuma līmeni (attiecīgi 0,013 un 0,011 <0,05).

7.4 Autokorelācija

Šai analīzei ir nepieciešama datu rinda, kas satur informāciju par laikā veiktiem novērojumiem (var izmantot arī telpā veiktos novērojumus, ja ievērots to izvietojums). Piemēram derēs fails *priedes.txt*, kas satur informāciju par priedes gadskārtu platumu simts gadu periodā.



Att. 7.3: Priežu gadskārtu platumu aukorelācijas attēls

```
> priede<-read.table(file="priede.txt",header=T,sep="\t",dec=".")
> str(priede)

'data.frame':      100 obs. of  1 variable:
 $ gadsk: num  0.89 0.92 0.87 1.39 1.11 0.92 1.16 1.08 1.13 1.71 ...
```

Autokorelācijas aprēķināšanai izmanto funkciju `acf()`, kurai kā arguments jānorāda kolonna/vektors, kas satur laika rindas datus. Rezultātā automātiski parādās attēls (7.3 attēls), kurā attēlotas autokorelācijas vērtības ar nobīdi līdz 20 vienībām (nobīdi var mainīt ar argumentu `lag.max=`). Korelācijas koeficients uzskatāms par būtiski, ja tā stabiņš pārsniedz raustīto līniju (uz augšu pozitīvajām vērtībām un uz leju negatīvajām vērtībām). Jāņem vērā, ka pirmais stabiņš attiecas uz nobīdi 0 un šis koeficients vienmēr būs 1.

```
> acf(priede)
```

Ja nepieciešams iegūt autokorelācijas skaitliskās vērtības, tad funkcijas `acf()` rezultāts jā-saglabā kā objekts, kuru pēc tam apskatot var iegūt konkrētās vērtības.

```
> akor<-acf(priede)
> akor

Autocorrelations of series 'priede', by lag
```

0	1	2	3	4	5	6	7	8	9	10
1.000	0.419	0.360	0.328	0.105	0.161	0.211	0.023	0.139	0.160	0.036
11	12	13	14	15	16	17	18	19	20	
0.141	0.088	-0.016	0.069	0.017	-0.036	0.064	-0.045	-0.017	0.034	

Nodaļa 8

Regresijas analīze

8.1 Dati

Regresijas analīzes piemēram izmantosim datu no datu faila *augi2.txt*. Šajā failā ir informācija no augu audzēšanas eksperimenta 45 parauglaukumos. No katra parauglaukuma ir informācija par nokrišņu daudzumu jūnijā, jūlijā un augustā, kā arī visā augšanas sezonā kopā, vidējais gaismas daudzums sezonā, augsnes pH un kopējā augu biomasa katrā no parauglaukumiem. Izmantojot funkciju `summary()` var apskatīt visu parametru vērtību izkriedi un raksturojumu.

```
> augi2<-read.table(file="augi2.txt",header=TRUE,sep="\t",dec=".")
> str(augi2)

'data.frame':      45 obs. of  7 variables:
 $ jun.nokr: num  78.2 79.7 76 65.7 82.4 92 64.2 40.1 87.5 66.2 ...
 $ jul.nokr: num  48.4 80.2 58.8 103.5 37 ...
 $ aug.nokr: num  100.3 34.8 79.6 64.4 70.1 ...
 $ sezona  : num  276 252 267 279 239 ...
 $ gaisma  : int  57 42 77 89 79 89 86 85 93 72 ...
 $ pH      : num  7.19 6.69 6.99 6.7 6.56 6.33 7.01 6.38 6.65 6.95 ...
 $ biomasa : num  37.2 30.6 36.3 59.9 32.9 44.7 43.7 51.6 53.8 44.3 ...

> summary(augi2)

      jun.nokr      jul.nokr      aug.nokr      sezona
Min.   : 40.10   Min.   : 26.5    Min.   : 34.80   Min.   :170.6
1st Qu.: 60.10   1st Qu.: 45.5    1st Qu.: 58.90   1st Qu.:221.4
Median : 73.40   Median : 51.6    Median : 66.90   Median :248.0
Mean   : 71.66   Mean   : 55.9    Mean   : 67.59   Mean   :244.4
3rd Qu.: 82.40   3rd Qu.: 66.9    3rd Qu.: 76.40   3rd Qu.:268.0
Max.   :106.20   Max.   :103.5    Max.   :100.30   Max.   :303.7

      gaisma      pH      biomasa
Min.   :35.00   Min.   :6.330   Min.   :14.20
1st Qu.:55.00   1st Qu.:6.630   1st Qu.:32.90
Median :72.00   Median :6.820   Median :39.50
Mean   :68.96   Mean   :6.796   Mean   :38.74
3rd Qu.:85.00   3rd Qu.:7.010   3rd Qu.:44.30
Max.   :96.00   Max.   :7.190   Max.   :59.90
```

8.2 Pāru regresija

Lineārās pāru regresijas veikšanai izmanto funkciju `lm()`, kurai kā pirmo argumentu norāda regresentu jeb atkarīgo mainīgo, tad nāk tildes zīme un viens vai vairāki regresori jeb atkarīgie mainīgie. Ja ir vairāk kā viens regresors, tad tos atdala ar plus zīmi. Kā pēdējais arguments jānorāda datu tabulu, no kuras jāņem dati. Regresijas analīzes rezultātu apskatīšanai jāizmanto funkcija `summary()`, kurai kā argumentu norāda objektu, kas satur regresijas analīzes modeli. Analīzes rezultātos ir vairākas sadaļas: (a) Residuals: atlikuma vērtību raksturojums; (b) Coefficients: regresijas vienādojuma koeficientu raksturojums – Estimate ir aprēķinātie regresijas vienādojuma koeficienti – (Intercept) - koeficients b_0 , vērtība pie regresora nosaukuma ir koeficients b_1 jeb slīpuma vērtība (slope); Std.Error – koeficientu standartklūdas, t value - atbilstošā T vērtība katram koeficientam un $\Pr(>|t|)$ ir atbilstošā p-vērtība novērtējot katram koeficienta būtiskumu atsevišķi; (c) apakšējā daļā ir visa modeļa kopējais novērtējums – atlikuma standartklūda un brīvības pakāpju skaits, determinācijas koeficients un pielāgotais determinācijas koeficients, F vērtība un p-vērtība visam modelim kopumā.

```
> mod1<-lm(biomasa~gaisma,data=augi2)
> summary(mod1)

Call:
lm(formula = biomasa ~ gaisma, data = augi2)

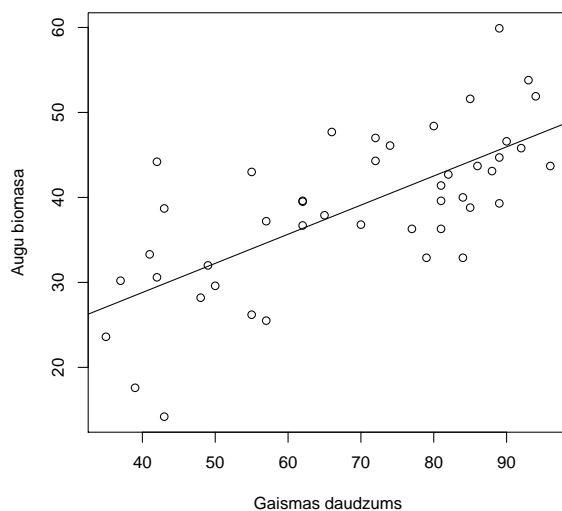
Residuals:
    Min       1Q   Median       3Q      Max
-15.6391  -3.8922  -0.5067   4.5209  14.7037

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 15.10044    3.86300   3.909 0.000324 ***
gaisma       0.34276    0.05411   6.334 1.19e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.709 on 43 degrees of freedom
Multiple R-squared:  0.4827,    Adjusted R-squared:  0.4707
F-statistic: 40.12 on 1 and 43 DF,  p-value: 1.193e-07
```

Secinājumi: iegūtais regresijas modelis ir $\text{biomasa} = 15,10044 + 0,34276 \cdot \text{gaisma}$, tas ir, gaismas daudzumam ir pozitīva ietekme uz biomasas pieaugumu. Gan viss regresijas modelis, gan arī atsevišķi katrs no regresijas koeficientiem šajā gadījumā ir statistiski būtiski, jo atbilstošās p-vērtības ir mazākas par būtiskuma līmeni. Dotais regresijas modelis izskaidro 47,07% no regresenta jeb biomasas vērtību variēšanas.

Regresijas analīzes rezultātu attēlošanai vai izveidot grafiku, kur reālie novērojumi ir attēloti kā punkti, bet papildus pievienota regresijas taisne (8.1 attēls). Šīs taisnes pievienošanai jāizmanto funkcija `abline()` kopā ar funkciju `lm()`.

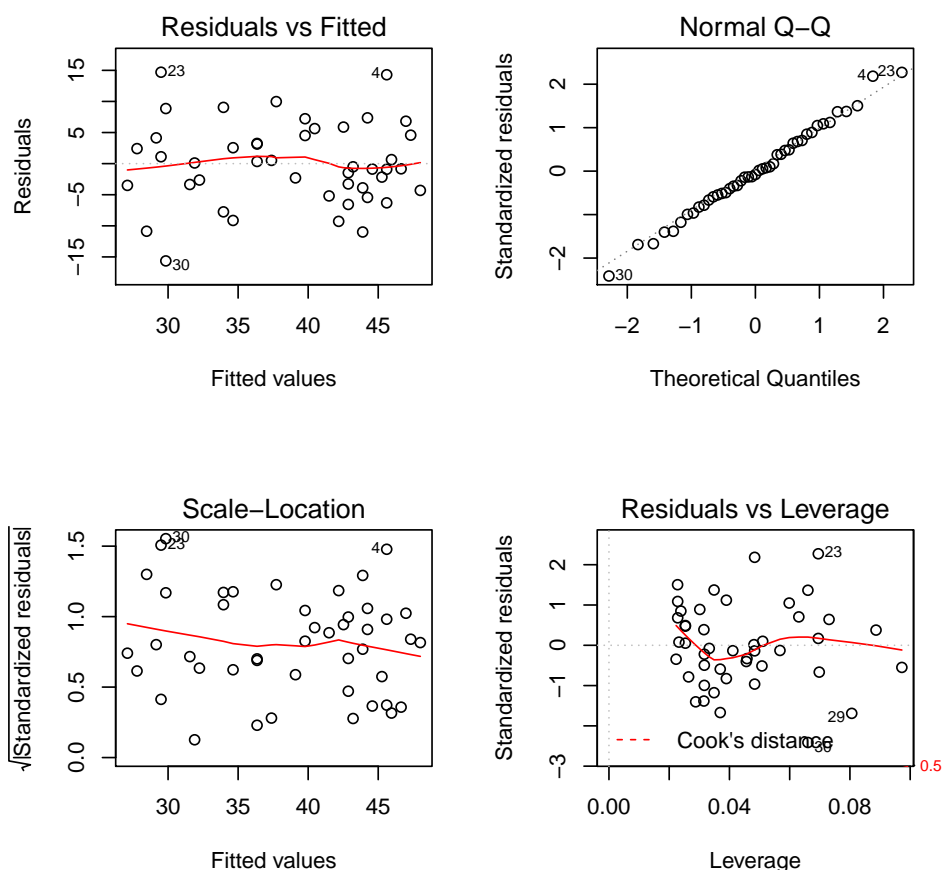


Att. 8.1: Izkliedes grafiks starp pazīmēm gaisma un biomasa ar pievienotu regresijas līkni

```
> plot(augi2$biomasa~augi2$gaisma,xlab="Gaismas daudzums",ylab="Augu biomasa")
> abline(lm(augi2$biomasa~augi2$gaisma))
```

8.3 Regresijas analīzes pieņēmumu pārbaude

Lai arī regresijas analīzē tiek konstatēts, ka modelis ir statistiski būtisks, tas nav analīzes noslēdzošais posms. Papildus ir jāpārbauda vai tiek ievēroti visi regresijas analīzes pieņēmumi. Viens no veidiem tā paveikšanai ir izmantot grafiskās iespējas. Ja funkcijā `plot()` kā argumentu norāda regresijas analīzes modeli, tad iegūst četrus diagnosticējošos grafikus. Visu četru grafiku vienlaicīgai apskatīšanai papildus jāizmanto funkcija `par()` ar argumentu `mfrow=`. Iegūtajos attēlos (8.2 attēls) pirmajā kolonnā ir redzamas atlikuma vērtības attiecībā pret prognozētajām vērtībām un standartizētās atlikuma vērtības attiecībā pret prognozētajām vērtībām. Šajos divos grafikos nedrīkst parādīties nekādas iezīmes, piemēram, trendi, kā arī punktiem ir jābūt nejauši izkārtotiem. Ja ir redzams trends, tas nozīmē, ka ir vēl kāds būtisks faktors, kas nav ņemts vērā. Ja ir vērojama vērtību izkliedes izmaiņas, vai punktu grupēšanās, tad ir problēmas ar dispersiju homogenitāti. Augšējā labējā stūrī ir grafiks, kas parāda standartizēto atlikuma vērtību QQ grafiku. Tas norāda vai atlikuma vērtības atbilst normālajam sadalījumam. Apakšējā labējā stūrī ir grafiks, pēc kura var spriest par katra novērojuma ietekmi uz kopējo modeli. Ja ir kāds punkts, kurš atrodas ārpus raustītajām Cook's distance līnijām (vērtības virs 0.5 vai virs 1), tad šim novērojumam ir liela ietekme uz kopējo regresijas modeli un tāpēc ir vērts pievērst uzmanību šim novērojumam.

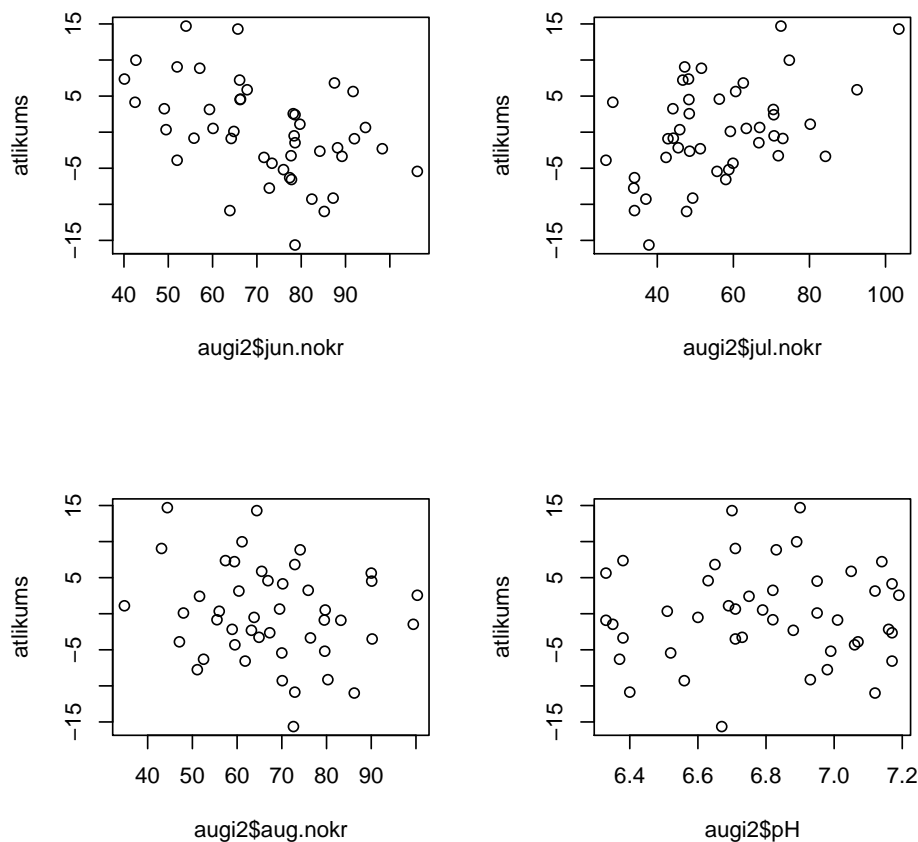


Att. 8.2: Regresijas analīzes diagnosticējošie grafiki

```
> par(mfrow=c(2,2))
> plot(mod1)
```

Vēl viens no variantiem, ko var pārbaudīt pēc regresijas analīzes, ir atlikuma vērtību izkliede atkarībā no citiem mainīgajiem, kas netika iekļauti modelī (protams, ja ir dati par šiem citiem mainīgajiem). Arī šajos attēlos nevajadzētu redzēt nekādus trendus, vai punktu grupēšanos. 8.3 attēlā ir samērā skaidri redzama saistība starp jūnija, jūlija un augusta nokrišņu daudzumu un atlikuma vērtībām, kas nozīmē, ka šo faktoru iekļaušana modelī to varētu uzlabot.

```
> atlikums<-resid(mod1)
> par(mfrow=c(2,2))
> plot(atlikums~augi2$jun.nokr)
> plot(atlikums~augi2$jul.nokr)
```



Att. 8.3: Izklīdes grafiki starp regresijas modeļa atlikuma vērtībām un modelī neiekļautajiem mainīgajiem

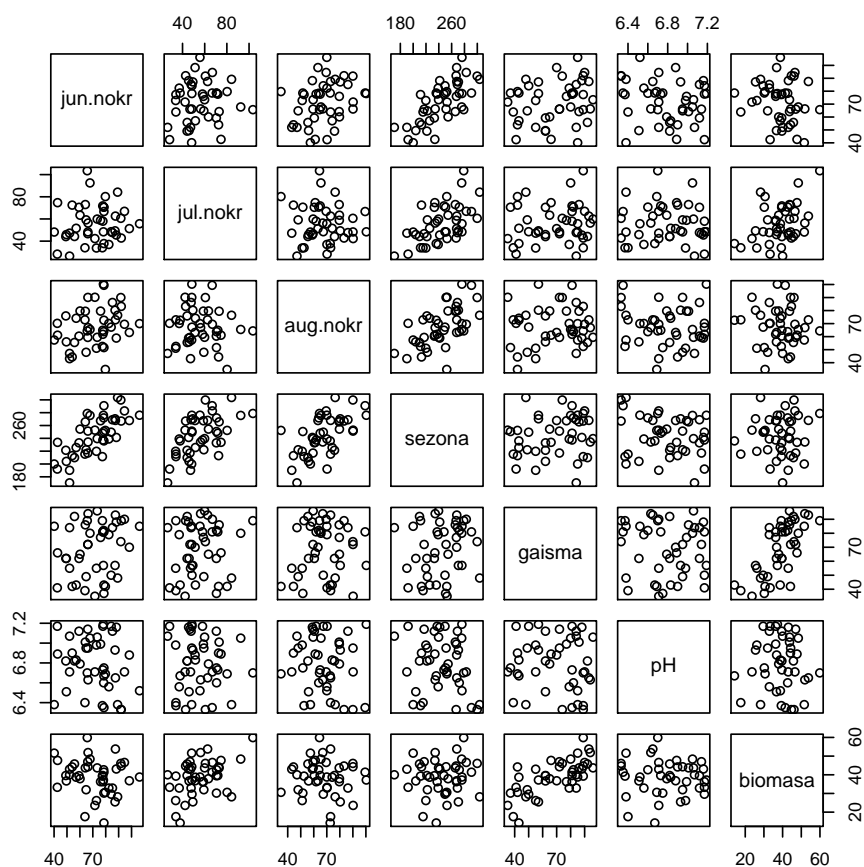
```
> plot(atlikums~augi2$aug.nokr)
> plot(atlikums~augi2$pH)
```

8.4 Vērtību prognozēšana

Viens no regresijas modeļu pielietojumiem ir jaunu vērtību prognozēšana. Lai to izdarītu, ir jāizmanto funkcija `predict()`, kurai kā argumentus jānorāda saglabātais regresijas modelis, kā arī datu tabula, kas satur jaunās vērtības. Papildus var iegūt arī ticamības intervālu šīm prognozētajām vērtībām ar argumentu `interval="prediction"`. Veidojot jauno datu tabulu ir jāatceras, ka tajā obligāti ir jābūt tādiem pašiem kolonnu nosaukumiem, kādi bija regresijas modelī izmantotie regresoru nosaukumi (tabulā drīkst būt arī citas kolonnas).

```
> progn<-data.frame(gaisma=c(10,50,100))
> predict(mod1,progn,interval="prediction")
```

```
      fit      lwr      upr
1 18.52802  3.411995 33.64405
2 32.23837 18.404298 46.07245
3 49.37631 35.284476 63.46814
```



Att. 8.4: Izkliedes grafiki starp visiem datu objekta augi2 mainīgajiem

8.5 Daudzfaktoru regresija

Pirmais solis daudzfaktoru regresijas analīzes veikšanai ir pārlicināšanās vai starp regresoriem nepastāv izteikta kolinearitāte (savstarpējā korelācija), jo tas būtu pārkāpums vienam no daudzfaktoru regresijas pieņēmumiem. Šim uzdevumam var izmantot gan grafisko analīzi, piemēram, ar funkciju `pairs()`, gan arī aprēķināt korelācijas koeficientus analītiski ar funkciju `cor()`. Gan no 8.4 attēla, gan arī no korelācijas analīzes rezultātiem ir redzams, ka ir būtis-

ka saistība starp veģetācijas sezonas kopējo nokrišņu daudzumu un atsevišķu mēnešu nokrišņu daudzumu, tāpēc nebūtu ieteicams vienlaicīgi visus šos mainīgos iekļaut regresijas modeli.

```
> pairs(augi2)
> round(cor(augi2),2)
```

	jun.nokr	jul.nokr	aug.nokr	sezona	gaisma	pH	biomasa
jun.nokr	1.00	0.11	0.30	0.73	0.22	-0.18	-0.18
jul.nokr	0.11	1.00	-0.05	0.57	0.11	-0.05	0.41
aug.nokr	0.30	-0.05	1.00	0.60	0.08	-0.11	-0.08
sezona	0.73	0.57	0.60	1.00	0.19	-0.19	0.07
gaisma	0.22	0.11	0.08	0.19	1.00	-0.08	0.69
pH	-0.18	-0.05	-0.11	-0.19	-0.08	1.00	-0.03
biomasa	-0.18	0.41	-0.08	0.07	0.69	-0.03	1.00

Daudzfaktoru regresijas analīzes veikšanai izmanto to pašu funkciju `lm()`, tikai papildus norāda nevis vienu, bet vairākus regresorus, kas savā starpā ir atdalīti ar plus zīmi. Piemērā apskatīts kā augu biomasu ietekmē pieci regresori.

```
> mod2<-lm(biomasa~gaisma+jun.nokr+jul.nokr+aug.nokr+pH,data=augi2)
> summary(mod2)
```

Call:

```
lm(formula = biomasa ~ gaisma + jun.nokr + jul.nokr + aug.nokr +
    pH, data = augi2)
```

Residuals:

Min	1Q	Median	3Q	Max
-10.054	-3.449	-1.333	3.439	8.766

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	22.955554	21.640357	1.061	0.295317
gaisma	0.361933	0.042487	8.519	1.95e-10 ***
jun.nokr	-0.221332	0.052954	-4.180	0.000159 ***
jul.nokr	0.200221	0.046717	4.286	0.000115 ***
aug.nokr	-0.001512	0.055002	-0.027	0.978213
pH	-0.648601	2.942489	-0.220	0.826689

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.115 on 39 degrees of freedom

Multiple R-squared: 0.7273, Adjusted R-squared: 0.6923

F-statistic: 20.8 on 5 and 39 DF, p-value: 4.495e-10

Secinājumi: kopējais regresijas modelis ir būtisks, jo p-vērtība ir mazāka par būtiskuma līmeni. Šis modelis spēj izskaidrot 69,23% no biomasas vērtību variēšanas. Tomēr apskatot katru no regresoriem atsevišķi, redzams, ka augsta nokrišņu daudzuma un augsnes pH koeficienti nav būtiski.

Šajā gadījumā ir iespējami vairāki risinājumi, piemēram, apstāties pie šiem rezultātiem, vai arī veidot jaunu modeli, kurā ir izņemti nebūtiskie faktori. Katrai no pieejām ir savi piekritēji un savi kritiķi. Jāatceras, ka katra koeficienta vērtība modelī ir atkarīga no pārējiem regresoriem, kas ir iekļauti šajā modelī. Šoreiz izmantosim pieeju, ka nepieciešams izveidot modeli, kurā visi koeficienti ir būtiski, tāpēc pakāpeniski ņemsim ārā nebūtiskos regresorus. Kā pirmo izslēgsim augsta nokrišņu summu, jo šim regresoram ir vislielākā p-vērtība (0,978).

```
> mod3<-lm(biomasa~gaisma+jun.nokr+jul.nokr+pH,data=augi2)
> summary(mod3)

Call:
lm(formula = biomasa ~ gaisma + jun.nokr + jul.nokr + pH, data = augi2)

Residuals:
    Min       1Q   Median       3Q      Max
-10.057  -3.453  -1.302   3.459   8.742

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  22.84355    20.98606   1.089   0.283
gaisma        0.36191     0.04194   8.629 1.13e-10 ***
jun.nokr     -0.22175     0.05010  -4.426 7.22e-05 ***
jul.nokr      0.20034     0.04592   4.362 8.79e-05 ***
pH           -0.64350     2.89971  -0.222  0.826
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 5.05 on 40 degrees of freedom
Multiple R-squared:  0.7273,    Adjusted R-squared:  0.7
F-statistic: 26.67 on 4 and 40 DF,  p-value: 8.057e-11
```

Secinājumi: Arī šis modelis kopumā ir statistiski būtisks, turklāt tam ir nedaudz lielāka izskaidrotā variācija (70,0%). Tomēr pH ietekme joprojām nav būtiska, tāpēc veidojam jaunu modeli.

```
> mod4<-lm(biomasa~gaisma+jun.nokr+jul.nokr,data=augi2)
> summary(mod4)

Call:
lm(formula = biomasa ~ gaisma + jun.nokr + jul.nokr, data = augi2)
```

```
Residuals:
```

```
   Min      1Q  Median      3Q      Max
-9.972 -3.318 -1.039   3.191   9.003
```

```
Coefficients:
```

```
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 18.29813     4.51651   4.051 0.000221 ***
gaisma       0.36229     0.04142   8.747 6.40e-11 ***
jun.nokr    -0.21996     0.04888  -4.500 5.50e-05 ***
jul.nokr     0.20066     0.04537   4.423 7.01e-05 ***
```

```
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 4.991 on 41 degrees of freedom
```

```
Multiple R-squared:  0.7269,    Adjusted R-squared:  0.707
```

```
F-statistic: 36.38 on 3 and 41 DF,  p-value: 1.242e-11
```

Secinājumi: pēdējā modelī visi atsevišķie koeficienti ir statistiski būtiski, kā arī būtisks ir viss modelis kopā. Izskaidrotā variācija ir 70,7%. Apskatot koeficientu vērtības, var secināt, ka gaismas daudzuma un jūlija nokrišņu daudzuma ietekme ir pozitīva, bet jūnija nokrišņu ietekme ir negatīva.

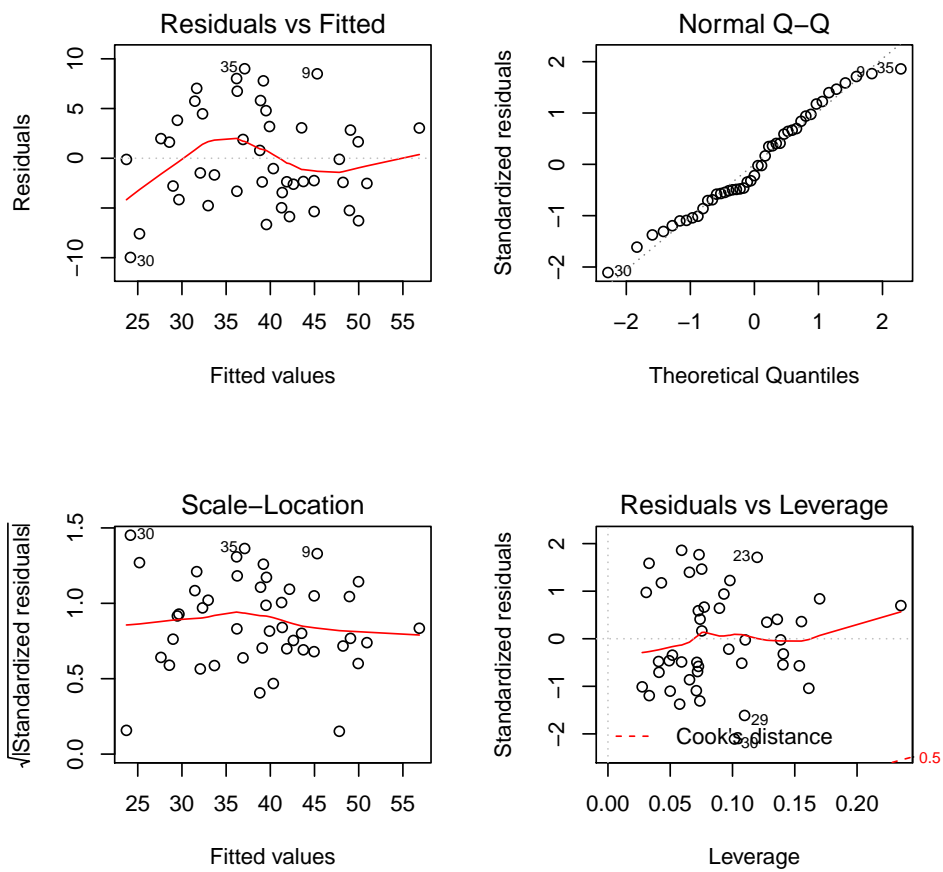
Lai salīdzinātu divus vai vairākus modeļus (ar kopīgu regresentu) savā starpā, var izmantot AIC rādītāju, ko aprēķina ar funkciju `AIC()`. Salīdzinot modeļus, labāks modelis ir tas, kura AIC vērtība ir mazāka. Šajā gadījumā mazākā AIC vērtība ir mod4, kurā bija iekļauti tikai trīs būtiskie regresori.

```
> AIC(mod2,mod4)
```

```
      df      AIC
mod2  7 282.1547
mod4  5 278.2109
```

Pārbaudot labāko modeli ar diagnosticējošiem grafikiem (8.5 attēls), varam izdarīt secinājumu, ka atlikuma vērtības atbilst normālajam sadalījumam, kā arī modelī nav ietekmīgu novērojumu.

```
> par(mfrow=c(2,2))
> plot(mod4)
```



Att. 8.5: Regresijas analīzes diagnosticējošie grafiki

Nodaļa 9

Kovariācijas analīze (ANCOVA)

9.1 Dati

Kovariācijas analīzes piemēram izmantosim datus no faila *dieta.txt* no pētījuma par to, vai svara samazinājums ir lielāks tad, ja ievēro diētu vai arī diētu kopā ar sportošanu (mainīgais *metode*). Par katru pētījumā iesaistīto cilvēku ir zināms tā sākotnējais svars un pētījumā laikā uzrādītais svara samazinājums.

```
> dieta<-read.table(file="dieta.txt",header=T,sep="\t",dec=".")
> str(dieta)

'data.frame':      42 obs. of  3 variables:
 $ svars          : num  60 62.5 65 67.5 70 72.5 75 77.5 80 82.5 ...
 $ metode         : Factor w/ 2 levels "esana","kopa": 2 2 2 2 2 2 2 2 2 ...
 $ samazinajums  : num  4.2 5.4 2.8 8.2 5 8.4 9.6 8.5 6.7 7.1 ...

> summary(dieta)

      svars      metode      samazinajums
Min.   : 60.00   esana:21   Min.    : 2.800
1st Qu.: 85.62   kopa :21   1st Qu.: 5.425
Median :100.00                Median  : 7.900
Mean   :100.00                Mean    : 7.502
3rd Qu.:114.38                3rd Qu.: 9.600
Max.   :140.00                Max.    :14.400
```

9.2 Dispersijas analīze

Pirmais variants kādā veidā varētu analizēt šos datus, ir izmantot dispersijas analīzi, kur kā atkarīgais mainīgais ir *samazinajums*, bet neatkarīgais mainīgais ir *metode*. Šoreiz dispersijas analīzes veikšanai izmantota cita pieeja – ar funkcijām `lm()` un `anova()`, kas dod vienādu rezultātu kā funkcija `aov()`.

```

> mod<-lm(samazinajums~metode,data=dieta)
> anova(mod)

Analysis of Variance Table

Response: samazinajums
      Df Sum Sq Mean Sq F value Pr(>F)
metode   1  19.339  19.3393   2.9155 0.09548 .
Residuals 40 265.330   6.6333
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Secinājums: dispersijas analīzes rezultāti parāda, ka faktora *metode* ietekme nav statistiski būtiska (p-vērtība ir lielāka par 0,05). Ja apskatām abu metožu vidējās vērtības, kā arī vērtību grafisko attēlojumu (9.1 attēls), var redzēt, ka starp abām grupām ir neliela starpība, bet vienlaicīgi arī vērojama izteikta vērtību pārklāšanās.

```

> with(dieta,tapply(samazinajums,metode,mean))

      esana      kopa
6.823810 8.180952

> with(dieta,boxplot(samazinajums~metode))

```

9.3 Kvantitatīva mainīgā iekļaušana modelī

Ja vienīgais statistiskais tests, kuru izvēlētos veikt šajā pētījumā būtu dispersijas analīze, tad paliktu pie secinājuma, ka diētas metodes savā starpā neatšķiras. Bet būtiski ir ņemt vērā arī to, ka ir pieejami dati arī par katra cilvēka sākotnējo svaru, turklāt šis sākotnējais svars katrā grupā ir atšķirīgs. Lai pārbaudītu vai šai papildus informācijai (cilvēka sākotnējais svars) ir būtiska ietekme uz to, kādu efektu dod diētas metode, var veikt kovariācijas analīzi. Šajā analīzē mēs pārbaudām viena vai vairāku faktoru ietekmi, vienlaicīgi ņemot vērā vēl cita kvantitatīvā mainīgā vērtības. Kovariācijas analīzi veic ar funkciju `lm()`, vienlaicīgi norādot abus mainīgo veidus. Rezultātu apskatīšanai izmanto funkcijas `anova()` un `summary()`.

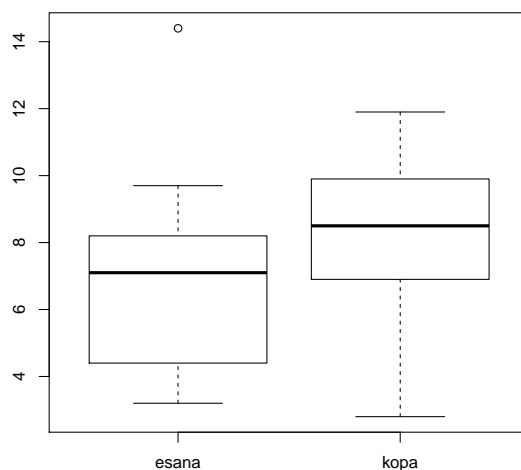
```

> mod1<-lm(samazinajums~metode+svars,data=dieta)
> anova(mod1)

```

Analysis of Variance Table

Response: samazinajums



Att. 9.1: Box-plot grafiks svara samazinājumam atkarībā no diētas veida

```

      Df Sum Sq Mean Sq F value Pr(>F)
metode  1  19.339   19.339   5.3628 0.02592 *
svars   1 124.688  124.688  34.5757 7.6e-07 ***
Residuals 39 140.643    3.606
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Secinājums: Iekļaujot modelī arī sākotnējo svaru, redzams, ka tagad pastāv statistiski būtiska atšķirība starp diētas veidiem (p -vērtība 0,026), kā arī sākotnējam svaram ir būtiska ietekme uz to, kāds ir svara samazinājums.

Analīzes rezultātu apskatot ar funkciju `summary()`, var iegūt atbilstošos lineāra modeļa koeficientus. Šeit svarīgi ņemt vērā to, ka kvalitatīvais mainīgais tiek pārkodēts pēc principa, ka alfabētiski pirmais mainīgā līmenis (šajā gadījumā `esana`) tiek izmantots kā references līmenis, bet visi pārējie līmeņi tiek salīdzināti ar šo līmeni.

```

> summary(mod1)

Call:
lm(formula = samazinajums ~ metode + svars, data = dieta)

Residuals:
    Min       1Q   Median       3Q      Max
-3.8419 -1.4185 -0.2487  1.2422  5.0153

Coefficients:

```

```

              Estimate Std. Error t value Pr(>|t|)
(Intercept) -6.26528    2.26424  -2.767   0.0086 **
metodekopa   4.77169    0.82502   5.784 1.03e-06 ***
svars        0.11382    0.01936   5.880 7.60e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.899 on 39 degrees of freedom
Multiple R-squared:  0.5059,    Adjusted R-squared:  0.4806
F-statistic: 19.97 on 2 and 39 DF,  p-value: 1.068e-06

```

Secinājums: sākotnējam svaram ir pozitīvs efekts uz svara samazinājumu, tas ir, jo lielāks ir sākotnējais svars, jo lielāks ir absolūtais svara samazinājums. Koeficients pie (Intercept) (b_0) atbilst situācijai, kad diētas metode bija *esana*, bet metodei *kopa* b_0 vērtība veidojas saskaitot koeficientus pie (Intercept) un pie metodekopa. Koeficients pie metodekopa ir būtisks, tas ir, pastāv statistiski būtiska atšķirība starp svara samazinājumu starp abām šīm diētas metodēm pie vienāda sākotnējā svara un šī starpība ir 4,77 kg.

Kovariācijas analīzē var pārbaudīt arī situāciju, kad ir ne tikai atšķirīga b_0 koeficienta vērtība starp gradācijas klasēm, bet arī to vai ir būtiska atšķirība starp b_1 koeficientiem (regresijas taisņu slīpumiem) starp gradācijas klasēm. Šī pētījuma kontekstā mēs varētu pārbaudīt vai pie dažādiem diētas veidiem ir vērojama atšķirīga saistība starp sākotnējo svaru un svara samazinājumu. Šādas analīzes veikšanai sākotnējā modeli ir jāiekļauj iedarbība (interaction) starp abiem mainīgajiem, ko panāk starp tiem norādot * zīmi.

```

> mod2<-lm(samazinajums~metode*svars,data=dieta)
> anova(mod2)

Analysis of Variance Table

Response: samazinajums
          Df Sum Sq Mean Sq F value    Pr(>F)
metode     1  19.339   19.339   5.3279   0.02652 *
svars      1 124.688  124.688  34.3511 8.832e-07 ***
metode:svars 1   2.710   2.710   0.7466   0.39299
Residuals 38  137.933   3.630
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

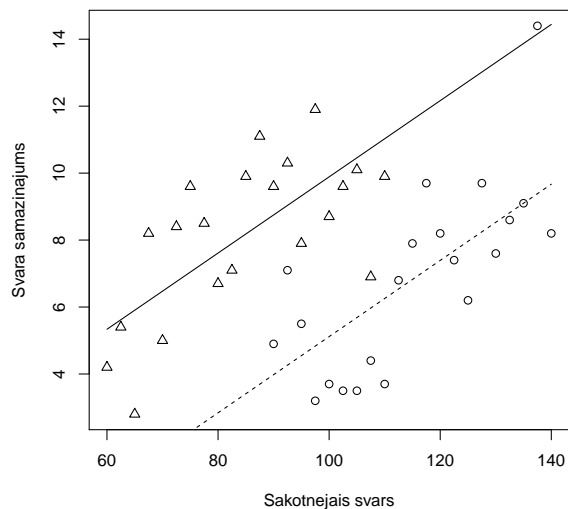
```

Secinājums: mainīgajam metode:svars nav statistiski būtiska ietekme uz svara samazinājumu, tas ir, nav atšķirības tajā kāda ir saistība starp sākotnējo svaru un svara samazinājumu atkarībā no diētas veida.

Ar funkcijas AIC() palīdzību var salīdzināt visus trīs modeļus kopā. Viszemākā AIC vērtība (attiecīgi labākais modelis) ir mod1, kurā bija iekļauti abi faktori bez savstarpējās iedarbības.


```
> AIC(mod,mod1,mod2)
```

	df	AIC
mod	3	202.6097
mod1	4	177.9501
mod2	5	179.1329



Att. 9.2: Izkliedes grafiks ar trenda līnijām starp sākotnējo svaru un svara samazinājumu atkarībā no diētas veida

Kovariācijas analīzes rezultātus vislabāk apskatīt grafiski (9.2 attēls). Kā pirmo soli radām mainīgo *jauns*, kas satur svara vērtības no 60 līdz 140 ik pa pieci – kopā 17 vērtības. Pēc tam ar funkciju `predict()` aprēķina prognozētās svara samazinājuma vērtības atsevišķi katrai no diētas metodēm. Ar funkciju `plot()` uzzīmē izkliedes grafiku starp oriģinālajām svara samazinājuma un sākotnējā svara vērtībām, turklāt simbola veids ir atkarīgs no diētas metodes. Pēc tam attēlam pa virsu uzliek trenda līnijas katrai no metodēm izmantojot funkciju `lines()`. Nepārtraukta līnija ir metodei kopa, bet raustīta līnija ir metodei esana.

```
> jauns<-seq(60,140,5)
> length(jauns)
```

```
[1] 17
```

```
> pred.esana<-predict(mod1,data.frame(svars=jauns,metode=rep("esana",17)))
> pred.kopa<-predict(mod1,data.frame(svars=jauns,metode=rep("kopa",17)))
> plot(dieta$samazinajums~dieta$svars,pch=as.numeric(dieta$metode),
```

```
+ xlab="Sakotnejais svars",ylab="Svara samazinajums")  
> lines(pred.kopa-jauns,lty=1)  
> lines(pred.esana-jauns,lty=2)
```

Nodaļa 10

Vispārinātie lineārie modeļi (GLM)

10.1 Dati

Divu GLM analīzes veidu (binārā loģistiskā regresija un puasona regresija) piemēram izmantoti četri datu faili. Pirmais fails `nezales.txt` satur informāciju par pētījumu, kurā analizēts nezāļu skaits parauglaukumos atkarībā no pielietotā augu aizsardzības līdzekļa - kontrole, līdzeklis 1 un līdzeklis 2 (kolonna grupa `grupa`), kā arī augsnes pH šajos parauglaukumos.

```
> nezales<-read.table(file="nezales.txt",header=T)
> str(nezales)

'data.frame':      300 obs. of  3 variables:
 $ grupa : Factor w/ 3 levels "kontrolē","lidzeklis1",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ pH    : num  6.26 6.8 6.79 6.52 6.08 6.21 6.47 6.74 6.71 6.11 ...
 $ skaits: int  2 10 5 6 5 4 2 2 5 3 ...

> head(nezales)

   grupa  pH skaits
1 kontrolē 6.26     2
2 kontrolē 6.80    10
3 kontrolē 6.79     5
4 kontrolē 6.52     6
5 kontrolē 6.08     5
6 kontrolē 6.21     4

> summary(nezales)

   grupa      pH      skaits
kontrolē :100  Min.   :5.310  Min.    : 0.00
lidzeklis1:100 1st Qu.:6.388  1st Qu.: 0.00
lidzeklis2:100 Median :6.700  Median : 1.00
              Mean  :6.699  Mean   : 2.14
              3rd Qu.:7.040  3rd Qu.: 3.25
              Max.  :7.920  Max.   :10.00
```

Failos `veids1.txt`, `veids2.txt` un `veids3.txt` ir viena un tā paša pētījuma dati, kas parādīti trīs dažādos veidos. Šajā pētījumā ir novērtēts vai augs ir bijis bojāts vai nē (slimības ietekmēts) atkarībā no tā šķirnes un audzēšanas veida (tunelis vai lauks). Failā `veids1` ir izejas dati, kur katram augam ir sava rindiņa un kolonnā `bojajums` ir atzīmē 0 vai 1 atkarībā no tā, vai augs nav bojāts, vai ir bojāts. Failā `veids2` dati jau ir apkopoti tādā veidā, ka katrai šķirnes un audzēšanas veida kombinācijai ir norādīts bojāto un nebojāto augu skaits. Failā `veids3` dati arī jau ir apkopoti, tikai šajā gadījumā ir kolonna ar procentuālo bojāto augu daudzumu (izteikts decimāldaļās) un kopējo augu skaitu katrā faktoru kombinācijā

```
> veids1<-read.table(file="veids1.txt", header=TRUE,sep="\t",dec=".")
> str(veids1)

'data.frame':      175 obs. of  3 variables:
 $ skirne  : Factor w/ 3 levels "A","B","C": 1 1 1 1 1 1 1 1 1 1 ...
 $ veids   : Factor w/ 2 levels "lauks","tunelis": 1 1 1 1 1 1 1 1 1 1 ...
 $ bojajums: int  1 1 1 1 0 1 0 1 0 1 ...

> head(veids1)

  skirne veids bojajums
1      A lauks         1
2      A lauks         1
3      A lauks         1
4      A lauks         1
5      A lauks         0
6      A lauks         1

> veids2<-read.table(file="veids2.txt", header=TRUE,sep="\t",dec=".")
> str(veids2)

'data.frame':      6 obs. of  4 variables:
 $ skirne  : Factor w/ 3 levels "A","B","C": 1 1 2 2 3 3
 $ veids   : Factor w/ 2 levels "lauks","tunelis": 1 2 1 2 1 2
 $ bojats  : int  17 10 14 16 21 2
 $ nebojats: int  3 17 11 24 13 27

> head(veids2)

  skirne  veids bojats nebojats
1      A  lauks    17         3
2      A tunelis  10        17
3      B  lauks    14        11
4      B tunelis  16        24
5      C  lauks    21        13
6      C tunelis   2         27

> veids3<-read.table(file="veids3.txt",header=TRUE,sep="\t",dec=".")
> str(veids3)
```

```
'data.frame':      6 obs. of  4 variables:
 $ skirne  : Factor w/ 3 levels "A","B","C": 1 1 2 2 3 3
 $ veids   : Factor w/ 2 levels "lauks","tunelis": 1 2 1 2 1 2
 $ procentis: num  0.85 0.37 0.56 0.4 0.618 ...
 $ skaits  : int   20 27 25 40 34 29

> head(veids3)

  skirne  veids  procentis skaits
1      A   lauks 0.85000000     20
2      A tunelis 0.37037037     27
3      B   lauks 0.56000000     25
4      B tunelis 0.40000000     40
5      C   lauks 0.61764706     34
6      C tunelis 0.06896552     29
```

10.2 Puasona regresija

Gadījumos, kad regresors (atkarīgais mainīgais) ir skaita dati, turklāt to vērtības ir mazas un sadalījums ir izteikti asimetrisks (daudz mazo vērtību, veidojas Puasona sadalījums), izmantot lineāro regresiju nebūtu pareizi, jo tiktu pārkāpti vairāki nosacījumi, kas izvirzīti šai analīzei. Analizējot skaita datus kā alternatīvu var izmantot vispārinātos lineāros modeļus (GLM) norādot, ka datiem ir Puasona atlikuma struktūra. Šo GLM veidu sauc arī par Puasona regresiju. GLM analīzi programmā R veic ar funkciju `glm()`. Funkcijā jānorāda trīs komponentes: formula, kas jāpārbauda (regresents atkarībā no viena vai vairākiem regresoriem un to kombinācijas), atlikumu struktūras veids, kas šajā gadījumā ir `family=poisson("log")`, kā arī datu objekts, kurā atrodas analizējamie mainīgie. Arguments `log` iekavās pie `poisson()` parāda kāda saistības funkcija ir izmantota, lai pārietu no oriģinālās saistības starp mainīgajiem uz lineāro saistību. Analīzes rezultātus var apskatīt ar funkcijām `summary()` un `anova()`. GLM modeļos var izmantot gan nepātraukti variējošus regresorus, gan arī kategorijas regresorus.

Piemērā apskatīts kā nezāļu skaitu ietekmē piederība pie kādas no pētījumu grupas un augsnes pH, kā arī šo divu faktoru kombinācija.

```
> mod<-glm(skaits~grupa*pH,family=poisson("log"), data=nezales)
> summary(mod)

Call:
glm(formula = skaits ~ grupa * pH, family = poisson("log"), data = nezales)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1527  -0.8192  -0.2107   0.6433   3.2340

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.1111111  0.1111111  9.95016  <2e-16 ***
grupa1         -0.1111111  0.1111111 -0.99502  0.31817
grupa2         -0.1111111  0.1111111 -0.99502  0.31817
pH             -0.1111111  0.1111111 -0.99502  0.31817
grupa1:pH      -0.1111111  0.1111111 -0.99502  0.31817
grupa2:pH      -0.1111111  0.1111111 -0.99502  0.31817
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Diagnostics:
(1) Cook's distance for each data point:
      [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
(2) Cook's distance for each parameter:
      [1] 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000 0.00000000
```

```

(Intercept)          1.51970    0.62685    2.424    0.0153 *
grupalidzeklis1     -2.21275    1.48261   -1.492    0.1356
grupalidzeklis2     -3.70175    2.41941   -1.530    0.1260
pH                   0.01156    0.09301    0.124    0.9011
grupalidzeklis1:pH  0.11340    0.21933    0.517    0.6051
grupalidzeklis2:pH  0.14455    0.36004    0.401    0.6881
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

Null deviance: 873.24  on 299  degrees of freedom
Residual deviance: 309.79  on 294  degrees of freedom
AIC: 879.63

Number of Fisher Scoring iterations: 5

> anova(mod, test="Chisq")

Analysis of Deviance Table

Model: poisson, link: log

Response: skaits

Terms added sequentially (first to last)

      Df Deviance Resid. Df Resid. Dev Pr(>Chi)
NULL                                299      873.24
grupa    2    562.83      297      310.41 <2e-16 ***
pH       1     0.23      296      310.18  0.6350
grupa:pH 2     0.39      294      309.79  0.8237
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Secinājumi: pēc analīzes kopsavilkuma tabulas var spriest, ka statistiski būtiska ir tikai Intercept vērtība (nezāļu skaits būtiski atšķiras no 0), tomēr pārējo koeficientu vērtības nav būtiskas. GLM analīzē, līdzīgi kā tas ir ANCOVA, ja ir iekļauts kvalitatīvs/kategorijas mainīgais (grupa), tad tas tiek pārkodēts un salīdzināšana tiek veikta references līmeni (grupu), kas šajā gadījumā ir kontrole. Koeficienti, kas parādās kopsavilkuma tabulā, ir koeficienti lineārājam vienādojumam. Lai atgrieztos pie sākotnējām skaita vērtībām, šis lineārais vienādojums ir jāizmanto kā pākāpe eksponentam - pretējā darbība naturālajam logaritmam. Ar funkciju `anova()` var apskatīt kā katrs faktors un kombinācija kopumā ietekmē analizējamo pazīmi. Pēc šīs tabulas varam secināt, ka būtiska ietekme ir grupai, bet nav pH un grupas:pH kombinācijai.

Tā kā pH ietekme nebija būtiska, šajā gadījumā varam izveidot otru modeli, kurā kā faktors ir tikai piederība pie grupas.

```

> mod2<-glm(skaitis~grupa,family=poisson("log"),data=nezales)
> summary(mod2)

Call:
glm(formula = skaitis ~ grupa, family = poisson("log"), data = nezales)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.1432  -0.8000  -0.1522   0.7063   3.1686

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)    1.59737    0.04499   35.50  <2e-16 ***
grupalidzeklis1 -1.44895    0.10317  -14.04  <2e-16 ***
grupalidzeklis2 -2.73680    0.18240  -15.01  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 873.24  on 299  degrees of freedom
Residual deviance: 310.41  on 297  degrees of freedom
AIC: 874.24

Number of Fisher Scoring iterations: 5

```

Secinājumi: Tā kā visi koeficienti ir būtiski, var secināt, ka gan grupā līdzeklis 1, gan līdzeklis 2 nezāļu skaits ir mazākas nekā kontroles grupā. Jāatceras, ka šie koeficienti nav interpretējami tiešā veidā kā atšķirības starp vidējiem nezāļu skaitiem, jo ir izmantota log saistības funkcija. Sagaidāmais nezāļu skaits kontroles grupā attiecīgi ir $e^{1.597}$, līdzeklis 1 grupā $e^{1.597-1.449}$ un līdzeklis 2 grupā $e^{1.597-2.737}$.

Tā kā tika ievēdota modelis, kurā bija divi mainīgie un to kombinācija, un modelis ar vienu mainīgo, tad šos modeļus var salīdzināt savā starpā, lai secinātu, vai modeļa vienkāršošana ir bijusi pamatota. Divu pakārtotu/saistītu modeļu salīdzināšanu var veikt ar funkcijā `AIC()`, vai `anova()`, kurai kā arguments norādīts `Chisq` tests.

```

> anova(mod,mod2,test="Chisq")

Analysis of Deviance Table

Model 1: skaitis ~ grupa * pH
Model 2: skaitis ~ grupa
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      294      309.80
2      297      310.41 -3  -0.61319  0.8934

```

```
> AIC(mod,mod2)

      df      AIC
mod    6 879.6302
mod2   3 874.2434
```

Secinājumi: izmantojot hī kvadrāta testu, konstatēts, ka atšķirība starp modeļiem nav būtiska (p vērtība ir 0,89), attiecīgi mēs varam izvēlēties vienkāršāko modeli, kurā ir tikai viens regresors. Arī salīdzinot pēc AIC vērtībām, otrajam modelim ir mazāka AIC vērtība un tas uzskatāms par piemērotāku.

10.3 Binārā loģistiskā regresija

Binārās loģistiskās regresijas veikšanai izmanto to pašu funkciju `glm()`, tikai šajā gadījumā kā `family=` jānorāda `binomial` un saistības funkcija ir `logit` (ko var arī mainīt pret `probit`). Formāts, kādā veidā norāda regresentu (atkarīgo mainīgo) ir atkarīgs no tā, kādā veidā ir pieejami dati. Binārajā loģistiskajā regresijā pārbauda vai iespējamības (ka iestāsies konkrētais notikums (būs bojājums)) būtiski atšķiras, atkarībā no ietekmējošiem faktoriem (gan kvantitatīviem, gan kvalitatīviem).

Ja ir izejas dati, kuros atkarīgais mainīgais sastāv no 0 un 1, tad funkcijā `glm()` šī kolonna ir jānorāda, kā atkarīgais mainīgais (šādā veidā dati ir pieejami failā `veids1`).

```
> mod1<-glm(bojajums~skirne+veids, family=binomial("logit"),data=veids1)
> summary(mod1)
```

Call:

```
glm(formula = bojajums ~ skirne + veids, family = binomial("logit"),
     data = veids1)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-1.7901	-0.9376	-0.6037	1.1195	1.8932

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	1.3773	0.3978	3.462	0.000536	***
skirneB	-0.4604	0.4203	-1.095	0.273354	
skirneC	-1.2396	0.4446	-2.788	0.005304	**
veidstunelis	-1.7476	0.3521	-4.963	6.95e-07	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance:	241.31	on 174	degrees of freedom
Residual deviance:	208.73	on 171	degrees of freedom

AIC: 216.73

Number of Fisher Scoring iterations: 4

Ja dati ir jau apkopoti un tajos ir kolonnas, kas parāda pozitīvo un negatīvo notikumu skaitu, kā failā `veids2` kolonnas `bojats` un `nebojats`, tad kā atkarīgais mainīgais formulā ir jānorāda abas kolonnas, kas apvienotas ar funkciju `cbind()`.

```
> mod2<-glm(cbind(bojats,nebojats)~skirne+veids,family=binomial("logit"),data=veids2)
> summary(mod2)
```

Call:

```
glm(formula = cbind(bojats, nebojats) ~ skirne + veids, family = binomial("logit"),
     data = veids2)
```

Deviance Residuals:

1	2	3	4	5	6
0.5944	-0.4049	-1.6402	1.2947	0.9796	-1.5666

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)	
(Intercept)	1.3773	0.3978	3.462	0.000536	***
skirneB	-0.4604	0.4203	-1.095	0.273354	
skirneC	-1.2396	0.4446	-2.788	0.005304	**
veidstunelis	-1.7476	0.3521	-4.963	6.95e-07	***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 40.8849 on 5 degrees of freedom

Residual deviance: 8.2977 on 2 degrees of freedom

AIC: 37.095

Number of Fisher Scoring iterations: 4

Trešā iespēja ir, ka dati apkopoti un tajos ir kolonna ar procentiem (decimāldaļās), kas parāda notiku iestāšanās biežumu (šajā gadījumā procentuālais bojāto augu daudzums). Lai analizētu šādus datus, kolonna ar procentiem jānorāda kā atkarīgais mainīgais, plus papildus jānorāda arguments `weights=` un kolonna, kurā ir oriģinālie kopējie skaita dati. To var nenorādīt gadījumos, ja skaita dati, no kuriem aprēķināti procenti ir vienādi starp visām faktoru kombinācijām (visās rindiņās).

```

> mod3<-glm(procents~skirne+veids,weights=skaits,family=binomial("logit"),data=veids3)
> summary(mod3)

Call:
glm(formula = procents ~ skirne + veids, family = binomial("logit"),
    data = veids3, weights = skaits)

Deviance Residuals:
    1      2      3      4      5      6 
0.5944 -0.4049 -1.6402  1.2947  0.9796 -1.5666

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)   1.3773     0.3978   3.462 0.000536 ***
skirneB      -0.4604     0.4203  -1.095 0.273354
skirneC      -1.2396     0.4446  -2.788 0.005304 **
veidstunelis -1.7476     0.3521  -4.963 6.95e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 40.8849  on 5  degrees of freedom
Residual deviance:  8.2977  on 2  degrees of freedom
AIC: 37.095

Number of Fisher Scoring iterations: 4

```

Secinājumi: neatkarīgi no datu sagatavošanas veida, visi iegūtie koeficienti un tie būtiskumi ir vienādi. Pastāv būtiska atšķirība starp skirnes A (izmantota kā bāzes līmenis) un skirnes C bojājumu iespējamību (šķirnei C tā ir mazāka nekā šķirnei A, ko parāda negatīvs koeficients pie šķirnes C), kā arī it būtiska atšķirība starp audzēšanas veidu - lauks un tunelis, turklāt tuneļa audzēšanas veidam bojājumu daudzums ir mazāks nekā laukam veidam. Jāatceras, ka arī šajā gadījumā šie ir lineārā vienādojuma koeficienti, kur y vērtības iegūtas izmantojot logit saistības funkciju no sākotnējiem bojājumu datiem.

Lai aprēķinātu prognozēto bojājumu iespējamību, var izmantot funkciju `predict()`, kurai kā arguments jānorāda modeļa objekts. Ja nenorāda papildus argumentus, tad aprēķinātās vērtības būs lineārajam vienādojumam. Prognozēto iespējamības vērtību iegūšanai jānorāda papildus arguments `type="response"`.

```

> predict(mod3)

    1      2      3      4      5      6 
1.3773246 -0.3702801  0.9168930 -0.8307117  0.1376954 -1.6099093

> predict(mod3,type="response")

```

1	2	3	4	5	6
0.7985610	0.4084733	0.7144086	0.3034946	0.5343696	0.1666012

Secinājumi: Izmantojot trešo modeli, iegūtas prognozētās bojājumu iespējamības vērtības katrai no sešām pārbaudītajām abu faktoru kombinācijām, piemēram, ja audzēšanas veids ir lauks un šķirne ir A (pirmā rindiņa failā veids3), tad prognozētais bojājumu iespējamība ir 0,798 jeb 79,8%.

Index

- χ^2 tests, 57
- ANCOVA, 84
- Atkarīgas paraugkopas, 54
- Bartletta tests, 63
- Binārā loģistiskā regresija, 95
- Bootstrap, 47
- Datu eksportēšana, 14
- Datu importēšana, 13
- Datu izzināšana, 37
- Datu tabula, 12
- Dispersijas analīze, 62
- Dispersiju salīdzināšana, 52
- F tests, 52
- Funkcijas
 - abline(), 35
 - acf(), 72
 - AIC(), 82
 - anova(), 84
 - aov(), 63
 - attach(), 18
 - axis(), 32
 - barplot(), 22
 - bartlett.test(), 63
 - boot(), 47
 - boot.ci(), 48
 - boxplot(), 21, 58
 - cbind(), 11
 - chisq.test(), 57
 - cor(), 69
 - cor.test(), 69
 - data(), 14
 - data.frame(), 12
 - detach(), 19
 - fix(), 13
 - glm(), 92
 - head(), 15
 - help(), 9
 - hist(), 21
 - layout(), 23
 - legend(), 34
 - leveneTest(), 53
 - library(), 8
 - lines(), 26
 - list(), 11
 - lm(), 75
 - ls(), 19
 - matrix(), 11
 - mtext(), 30
 - pairs(), 67
 - par(), 23
 - plot(), 20
 - plotCI(), 49
 - plotmeans(), 66
 - predict(), 78
 - qqnorm(), 40
 - qqPlot(), 41
 - qt(), 46
 - quartz(), 20
 - rbind(), 11
 - rcor.test(), 70
 - read.csv(), 13
 - read.table(), 13
 - shapiro.test(), 41
 - str(), 15
 - stripchart(), 39
 - t.test(), 52
 - tail(), 15
 - text(), 30
 - title(), 30
 - TukeyHSD(), 64
 - var.test(), 52
 - wilcox.test(), 56
 - windows(), 20
 - write.table(), 14
- GLM, 90
- Grafiki, 20
 - asis, 32
 - eksportēšana, 36
 - grafiku izvietojums, 23
 - krāsas, 34

- lēgenda, 34
- līnijas, 26
- simboli, 28
- teksti, 31
- virsraksti, 29

Instalācija, 8

Kolinearitāte, 79

Korelācijas analīze, 67

- Autokorelācija, 71
- Kendela, 71
- Pīrsona, 69
- Spirmena, 71

Kovariācijas analīze, 84

Matrice, 11

Normalitātes testi, 40

Paketes, 8

- boot, 47
- gplots, 66
- ltn, 70
- nortest, 42
- plotrix, 49

Paraugkopu salīdzināšana, 52

Permutāciju tests, 59

Puasona regresija, 92

Randomizēšanas tests, 59

Regresijas analīze

- Daudzfaktoru regresija, 79
- Pāru regresija, 75

Saraksts, 11

Statistiskie rādītāji

- Dispersija, 43
- Kvartiles, 45
- Mediāna, 43
- Procentiles, 45
- Standartnovirze, 43
- Vidējais aritmētiskais, 43

T tests, 52

Ticamības intervāls, 46

Vektors, 10

Vispārinātie lineārie modeļi, 90

Wilcoxon tests, 56

Literatūra

- Arhipova, I., & Bāliņa, S. (2006). *Statistika ekonomikā. Risinājumi ar SPSS un Microsoft Excel*. Datorzinību Centrs.
- Canty, A., & Ripley, B. (2012). *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-5.
- Crawley, M. (2007). *The R book*. Wiley.
- Everitt, B., & Hothorn, T. (2006). *A handbook of statistical analyses using R*. CRC Press.
- Lemon, J. (2006). Plotrix: a package in the red light district of r. *R-News*, 6(4), 8–12.
- Liepa, I. (1974). *Biometrija*. Zvaigzne.
- Maindonald, J., & Braun, W. (2010). *Data analysis and graphics using R. An example-based approach*. Cambridge.
- Manly, B. (2007). *Randomization, bootstrap and Monte Carlo methods in biology*. Chapman, Hall/CRC.
- Murrell, P. (2006). *R graphics*. CRC Press.
- R Core Team (2012). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
URL <http://www.R-project.org/>
- Sarkar, D. (2008). *Lattice. Multivariate data visualization with R*. Springer.
- Sokal, R., & Rohlf, F. (1995). *Biometry*. W.H. Freeman and company.
- Verzani, J. (2005). *Using R for introductory statistics*. Chapman and Hall/CRC.
- Wicham, H. (2009). *ggplot2. Elegant graphics for data analysis*. Springer.
- Zuur, A., Ieno, E., & Smith, G. (2007). *Analysing ecological data*. Springer.