

# Grafički sustav u R-u 1

S740



Ovu inačicu priručnika izradio je autorski tim Srca u sastavu:

Autor: dr.sc. Andreja Radović

Recenzent: dr.sc. Vedran Franke

Urednik: Vlasta Pavičić

## TEČAJEVISrca

Sveučilište u Zagrebu

Sveučilišni računski centar

Josipa Marohnića 5, 10000 Zagreb

edu@srce.hr

Verzija priručnika: S740-ver4



Ovo djelo dano je na korištenje pod licencom Creative Commons Imenovanje-Nekomercijalno-Dijeli pod istim uvjetima 4.0 međunarodna. Licenca je dostupna na stranici: <http://creativecommons.org/licenses/by-nc-sa/4.0/>.

## Sadržaj

<b>1 Značaj statističke grafike</b>	<b>1</b>
1.1 Prostori boja i ljudski vid . . . . .	1
<b>2 Boje i palete u R-u</b>	<b>3</b>
2.1 Boje u R-u . . . . .	3
2.1.1 Neke funkcije iz paketa <i>grDevices</i> . . . . .	3
2.1.1.1 Funkcija <b>colors()</b> ili <b>colours()</b> { <i>grDevices</i> } . . . . .	3
2.1.2 Specifikacija boja Crveno-Zeleno-Plavo (engl. <i>Red-Green-Blue - RGB</i> ) . . . . .	4
2.1.2.1 Funkcija <b>rgb()</b> { <i>grDevices</i> } . . . . .	4
2.1.3 HSL/HSV specifikacija boja . . . . .	5
2.1.3.1 Funkcija <b>hsv()</b> { <i>grDevices</i> } . . . . .	7
2.1.4 HCL specifikacija boja . . . . .	8
2.1.4.1 Funkcija <b>hcl()</b> { <i>grDevices</i> } . . . . .	8
2.1.5 Munsell specifikacija boje . . . . .	9
2.1.6 Prilagodba za daltonizam . . . . .	9
2.1.7 Prijelaz između sustava boja . . . . .	9
2.1.7.1 Funkcija <b>col2rgb()</b> { <i>grDevices</i> } . . . . .	10
2.1.7.2 Funkcija <b>col2hex()</b> { <i>grDevices</i> } . . . . .	11
2.1.7.3 Funkcija <b>convertColor()</b> { <i>grDevices</i> } . . . . .	12
2.2 Palete (sheme boja) u R-u . . . . .	12
2.2.1 Funkcije za upravljanje paletama { <i>grDevices</i> } . . . . .	12
2.2.1.1 Funkcija <b>palette()</b> { <i>grDevices</i> } . . . . .	12
2.2.1.2 Funkcija <b>gray()</b> { <i>grDevices</i> } . . . . .	13
2.2.1.3 Funkcije <b>heat.colors(); rainbow(); topo.colors()</b> { <i>grDevices</i> } . . . . .	14
2.2.1.4 Funkcija <b>colorRampPalette()</b> { <i>grDevices</i> } . . . . .	16
2.2.2 Funkcije za upravljanje paletama { <i>RColorBrewer</i> } . . . . .	18
2.2.2.1 Funkcija <b>display.brewer.all()</b> { <i>RColorBrewer</i> } . . . . .	18
2.2.2.2 Funkcija <b>brewer.pal()</b> { <i>RColorBrewer</i> } . . . . .	18
<b>3 R grafika</b>	<b>22</b>
3.1 Izlazni grafički formati (grafički uređaji - engl. <i>graphical devices</i> ) . . . . .	24
<b>4 Glavni grafički sustavi u R-u</b>	<b>26</b>
4.1 Osnovna grafika (engl. <i>base</i> ) . . . . .	26
4.1.1 Paket <i>graphics</i> . . . . .	26
4.1.1.1 Funkcija <b>par()</b> { <i>graphics</i> } . . . . .	27
4.1.1.1.1 Grafički parametri koji kontroliraju veličinu teksta i simbola . . . . .	28
4.1.1.1.2 Grafički parametri koji kontroliraju simbole za točke na grafu ( <i>pch</i> ) . . . . .	30
4.1.1.1.3 Grafički parametri koji kontroliraju boju elemenata grafa ( <i>col</i> ) . . . . .	32
4.1.1.1.4 Grafički parametri koji kontroliraju vrste linija ( <i>lty</i> ) . . . . .	35
4.1.1.1.5 Grafički parametri koji kontroliraju širinu linije ( <i>lwd</i> ) . . . . .	38
4.1.1.1.6 Grafički parametri koji kontroliraju regiju crtanja ( <i>mfcol/mfrow; layout()</i> i <i>split.screen()</i> ) .	39
4.1.1.1.7 Grafički parametri koji kontroliraju marge grafa ( <i>mar/mai</i> ) i ( <i>oma/omi</i> ) . . . . .	44
4.1.1.1.8 Grafički parametri koji kontroliraju automatsko obilježavanje osi ( <i>ann</i> ) . . . . .	46
4.1.1.1.9 Grafički parametri koji kontroliraju pozadinu (podlogu) područja na kojoj se crta ( <i>usr/bg</i> ) .	48
4.1.1.1.10 Grafički parametri koji kontroliraju promjenu udaljenosti između osi te naslova i labela osi ( <i>mgp</i> ) . . . . .	49
4.1.2 Neke grafičke funkcije visoke razine (engl. <i>high-level</i> ) . . . . .	50
4.1.2.1 Funkcija <b>plot()</b> { <i>graphics</i> }( <i>high-level</i> ) . . . . .	50
4.1.3 Poboljšanje rezultata dobivenoga funkcijama visoke razine funkcijama niske razine . . . . .	52
4.1.3.1 Funkcija <b>lines()/curves()</b> { <i>graphics</i> }( <i>low-level</i> ) . . . . .	53
4.1.3.2 Funkcija <b>rect()/polygon()</b> { <i>graphics</i> }( <i>low-level</i> ) . . . . .	55
4.1.3.3 Funkcija <b>legend()</b> { <i>graphics</i> } ( <i>low-level</i> ) . . . . .	58

4.1.3.4 Funkcija <b>axis()</b> {graphics} ( <i>low-level</i> ) . . . . .	60
4.1.3.5 Funkcija <b>abline()</b> {graphics}( <i>low-level</i> ) . . . . .	62
4.1.3.6 Funkcija <b>text()</b> {graphics} ( <i>low-level</i> ) . . . . .	65
4.1.3.7 Funkcija <b>mtext()</b> {graphics} . . . . .	66
4.1.3.8 Funkcija <b>box()</b> {graphics}( <i>low-level</i> ) . . . . .	70
4.1.3.9 Funkcija <b>title()"/sub()</b> {graphics} ( <i>low-level</i> ) . . . . .	71
4.1.4 Neke grafičke funkcije visoke razine - nastavak . . . . .	73
4.1.4.1 Funkcija <b>pairs()</b> {graphics}( <i>high-level</i> ) . . . . .	73
4.1.4.2 Funkcija <b>boxplot()</b> {graphics}( <i>high-level</i> ) . . . . .	73
4.1.4.3 Funkcija <b>dotchart</b> (old: <b>dotplot()</b> ) {graphics}( <i>high-level</i> ) . . . . .	79
4.1.4.4 Funkcija <b>cotplot()</b> {graphics}( <i>high-level</i> ) . . . . .	81
4.1.4.5 Funkcija <b>pie()</b> {graphics}( <i>high-level</i> ) . . . . .	83
4.1.4.6 Funkcija <b>hist()</b> {graphics}( <i>high-level</i> ) . . . . .	87
4.1.4.7 Funkcija <b>barplot()</b> {graphics}( <i>high-level</i> ) . . . . .	89
4.1.4.8 Funkcija <b>mosaicplot()</b> {graphics}( <i>high-level</i> ) . . . . .	90
4.1.4.9 Funkcija <b>assocplot()</b> {graphics}( <i>high-level</i> ) . . . . .	93
4.1.4.10 Funkcija <b>persp()</b> {graphics}( <i>high-level</i> ) . . . . .	96
4.1.4.11 Funkcija <b>filled.contour()</b> {graphics}( <i>high-level</i> ) . . . . .	98
4.1.4.12 Funkcija <b>image()</b> {graphics} . . . . .	100
4.2 Grid grafika/Lattice grafika (temeljena na paketu <i>grid</i> ) . . . . .	101
4.2.1 Paket <i>grid</i> . . . . .	101
4.2.1.1 Funkcija <b>viewport()</b> {grid} . . . . .	103
4.2.1.2 Funkcija <b>pushViewport</b> {grid} . . . . .	103
4.2.1.3 Funkcija <b>current.viewport()</b> {grid} . . . . .	107
4.2.1.4 Funkcija <b>current.vpTree()</b> {grid} . . . . .	107
4.2.1.5 Funkcija <b>downViewport()</b> {grid} . . . . .	107
4.2.1.6 Funkcija <b>grid.layout</b> {grid} . . . . .	107
4.2.2 Paket(i) <i>lattice/latticeExtra</i> . . . . .	113
4.2.3 Viewports/layouts (predlošci) u <i>lattice</i> -u . . . . .	114
4.2.4 Lattice uređaji . . . . .	114
4.2.5 Lattice plot prikazi . . . . .	115
4.2.6 Opće karakteristike prikaza u <i>latticeu</i> . . . . .	116
4.2.7 Grafički parametri za <i>Trellis</i> prikaze . . . . .	117
4.2.8 Kontroliranje grafičkih parametara u grafici temeljenoj na <i>grid/lattice</i> . . . . .	117
4.2.8.1 Funkcija <b>show.settings()</b> {lattice} . . . . .	117
4.2.8.2 Functions <b>trellis.par.get()</b> / <b>trellis.par.set()</b> {lattice} . . . . .	119
4.2.8.3 Funkcija <b>trellis.par.get()</b> . . . . .	119
4.2.8.4 Funkcija <b>trellis.par.set()</b> . . . . .	120
4.2.8.5 Teme u <i>lattice</i> grafici {Lattice/LatticeExtra} . . . . .	124
4.2.9 Panel funkcije u <i>lattice</i> grafici {LatticeExtra} . . . . .	131
4.2.10 Multi-panel uvjet u paketu <i>lattice</i> . . . . .	135
4.2.11 Grafički parametri koji kontroliraju regiju crtanja . . . . .	140
4.2.12 Najznačajnije funkcije visoke razine iz paketa <i>lattice</i> . . . . .	143
4.2.12.1 Funkcija <b>xypot()</b> {lattice} . . . . .	143
4.2.13 Poboljšanje rezultata kreiranog <i>high-level</i> funkcijom funkcijama niske razine ( <i>low-level</i> ) . . . . .	147
4.2.13.1 Argument <b>scales</b> . . . . .	147
4.2.13.2 Argument <b>type</b> . . . . .	147
4.2.13.3 Argument <b>group</b> . . . . .	151
4.2.13.4 Argument <b>key/auto.key</b> . . . . .	155
4.2.13.5 Argument <b>subset</b> . . . . .	155
4.2.13.6 Argument <b>layout</b> . . . . .	156
4.2.14 Najznačajnije funkcije visoke razine ( <i>high-level</i> ) iz paketa <i>lattice</i> - nastavak . . . . .	160
4.2.14.1 Function <b>histogram()</b> ( <i>high-level</i> ){lattice} . . . . .	160
4.2.14.2 Funkcija <b>densityplot()</b> ( <i>high-level</i> ){lattice} . . . . .	162

4.2.14.3 Funkcija <b>dotplot()</b> ( <i>high-level</i> ) <b>{lattice}</b>	164
4.2.14.4 Funkcija <b>bwplot()</b> ( <i>high-level</i> ) <b>{lattice}</b>	168
4.2.14.5 Function <b>levelplot()</b> <b>{lattice}</b>	170
4.2.14.6 Funkcija <b>wireframe()</b> <b>{lattice}</b>	173
4.2.14.7 Function <b>cloud()</b> <b>{lattice}</b>	173
4.2.14.8 Function <b>qqmath()</b> <b>{lattice}</b>	177
4.2.14.9 Funkcija <b>barchart()</b> ( <i>high-level</i> ) <b>{lattice}</b>	178
<b>5 Literatura</b>	<b>180</b>

Ovaj dokument generiran je iz RMarkdown skripte (Rstudio), kombiniranjem mogućnosti jezika R, Latex-a i Pandoc-a.



## 1 Značaj statističke grafike

Dva su ključna cilja statističke grafike: 1) olakšavanje usporedbe između skupova grupa i 2) identificiranje trendova. Statistička grafika se često upotpunjuje uporabom informacija o bojama koje su sadržane u nekoj varijabli. U slučajevima kada statistička grafika uključuje sjenčanje područja (a ne samo iscrtavanje točaka ili linija), npr. kao što je slučaj u bar dijagramima, pita dijagramima, mozaičkim prikazima ili toplinskim mapama, veoma je važno da se boje temelje na percepciji i da ne stvaraju optičke iluzije ili sustavnu pristranost (engl. *systematic bias*). Autor filozofije grafičke gramatike (engl. „Grammar of Graphics“), Leland Wilkinson tako daje definiciju statističke grafike: "Statistička grafika je ili treba biti trans-disciplinarno područje koje u obzir uzima znanstvene, statističke, računalne, estetske, psihološke i druge čimbenike. Znanosti (prirodne, primijenjene ili društvene i, doista, također i različite literarne, lingvističke i povjesne discipline) generiraju podatke koji su relevantni za različite probleme, a koji se mogu prikazati kroz grafički prikaz. Statistika savjetuje, ili, bolje reći, upućuje što se treba crtati. Softver i hardver su potrebni kako bi se proizveli prikazi u praksi: kako je malo onih koji danas posežu za olovkom i grafičkim papirom kao što je to bilo uobičajeno prije nekoliko desetljeća. Jasno možemo vidjeti koliko je grafika privlačna ili ne, a posebice koliko grafika funkcioniра ili ne, u pogledu toga da ju korisnici adekvatno tumače. Također je naveo i da je: „... statistička grafika središnjica suvremenoga statističkog istraživanja“.

Općenito gledano, boja je sastavni dio grafičkih prikaza. Mnoge statističke grafike posebno sadrže boju kao sastavni dio svojih grafičkih prikaza. Bilo koji softverski paket može lako generirati grafike u boji, kao i slike u boji koje su, tako, doslovce sveprisutne u elektroničkim publikacijama kao što su tehnička izvješća, slajdovi prezentacija, ili elektroničke verzije članaka u časopisima te, sve više, i u tiskanim časopisima. Međutim, često odabir boja u takvim prikazima nije optimalan uslijed toga što odabir boja nije trivijalan zadatak i postoji relativno malo smjernica o tome kako bi se trebale odabrati adekvatne boje za određenu vrstu vizualizacije. Tri ključne prepreke koje netko mora premostiti kada odabire boje za statističku grafiku jesu:

Iako nije prijeko potrebno da boje u statističkom dijagramu odražavaju modne trendove, osnovna načela poput izbjegavanja velikih područja s potpuno zasićenim bojama (Tufte 1990) svakako što svakako treba slijediti. Nije nužno da korisnik ima diplomu iz grafičkog dizajna, već da softver korisnicima pruži intuitivan način odabira boja i kontroliranja njihovih osnovnih svojstava. Stoga, postoji potreba da se koristi model boje ili prostor boje koji opisuje boje u pogledu njihovih percepcijskih svojstava: nijansa, svjetlina i šarenilo.

Modeli boja su, općenito, trodimenzionalni. Model boje, kakav obično podržavaju softverski paketi, uključuje specifikaciju boja kao Crveno-Zeleno-Plavo (engl. *Red-Green-Blue - RGB*). Ipak, ovakva specifikacija odgovara generiranju boja na zaslonu računala, a ne ljudskoj percepciji boja. Doslovno je nemoguće da ljudi kontroliraju percepcijske karakteristike boja u ovakovom prostoru boja zato što nema jedne dimenzije koja odgovara, na primjer, nijansi ili svjetlini boje. Kao posljedica, postoji niz sugestija za različite prostore boja koji se temelje na percepciji. Svaka dimenzija prostora boje može se upariti sa percepcijskim svojstvom. Jedan, široko korišteni, pristup u softverskim paketima uključuje **Nijansa-Zasicenost-Vrijednost** (engl. *Hue-Saturation-Value - HSV*), jednostavnu transformaciju RGB trojki. Međutim, nesretna je okolnost da dimenzije u HSV prostoru slabo mapiraju percepcijska svojstva te korištenje HSV boja potiče korištenje visoko-zasićenih boja. Model boja koji se zasniva na percepciji, kojim se ovi problemi rješavaju, uključuje **Nijansa-Jačina (čistoća)-Osvijetljenost** (engl. *Hue-Chroma-Luminance - HCL*).

Kada se biraju boje, treba se voditi sljedećim temeljnim načelima:

- boje trebaju biti privlačne u održenoj mjeri
- boje trebaju "surađivati" jedna s drugom.

Tipična je svrha boje u statističkoj grafici da omogući razlikovanje područja ili simbola na grafičkom prikazu te da omogući razlikovanje između različitih skupina ili različitih razina varijable. Drugim riječima, u dijagramu će se koristiti nekoliko boja koje se nazivaju paleta boja. Iz svega navedenog veoma je važno znati i moći stvarati vlastite sheme boja za vizualizaciju različitih fenomena.

### 1.1 Prostori boja i ljudski vid

Kako bi se odabrale palete boja, bilo bi dobro imati osnovna poimanja o tome kako ljudi vide boje.

Prepostavlja se da ljudsko oko najbolje primjećuje:

1. percepciju kontrasta svjetla/tame

2. kontrasti žuto/plavo (obično se povezuje s našim poimanjem toplih/hladnih boja)
3. kontrasti zeleno/crveno

što je povezano s anatomskom građom oka i funkcionalnim stanicama.

Mrežnica sadrži dvije vrste fotoreceptorskih stanica: čunjiće i štapiće. Čunjići služe za gledanje uz normalnu i jaku rasvjetu, a štapići za gledanje uz vrlo slabo osvjetljenje, noću ili u tamnim prostorima. Čunjići stvaraju obojenu, a štapići samo sivo-crnu sliku. Dakle, pomoću čunjića u mrežnici raspoznajemo boje. Tri su različite skupine čunjića u oku:

- čunjići osjetljivi na "crvenu" svjetlost - R (red)
- čunjići osjetljivi na "zelenu" svjetlost - G (green)
- čunjići osjetljivi na "plavu" svjetlost - B (blue)

Ljudsko oko ima tri odvojena receptora osjetljiva na tri osnovne boje (crvenu, zelenu i plavu) i osjet boje nastaje preklapanjem triju osnovnih osjeta. Svaka se druga boja može stvoriti kombinacijom crvene, zelene i plave svjetlosti. Kad svjetlost padne na mrežnicu osjete je jedna ili više skupina čunjića, ovisno o boji svjetlosti. Podražaj čunjića pretvara se u električni impuls koji se kroz vidni živac prenosi u mozak.

## 2 Boje i palete u R-u

R je programski jezik otvorenoga kôda za statističku analizu i grafiku. R ima iscrpne i moćne grafičke mogućnosti koje su usko povezane s njegovim analitičkim mogućnostima. I jedne i druge se brzo razvijaju. Svakih nekoliko mjeseci pojave se nove karakteristike i mogućnosti. Inicijalnu verziju R-a razvili su Ross Ihaka i Robert Gentleman sa Sveučilišta u Aucklandu (engl. University of Auckland). Danas je **The Core Team** odgovoran za razvoj R-a u različitim institucijama širom svijeta. Poput Linux-a, R je sustav *otvorenoga kôda*. Izvorni kôd može se pregledati ili izmijeniti i prilagoditi drugim sustavima. Izlaganje kôda kritičkom pregledu visoko-stručnih korisnika pokazalo se iznimno učinkovitim načinom identificiranja grešaka u kôdu (engl. bugs) i ostalih nedostatnosti te je pobudilo ideje za unapređenjem.

### 2.1 Boje u R-u

Modeli boja općenito imaju tri dimenzije uslijed činjenice o tri različita receptora za boje u čovjekovu oku. Boja se u R-u može specificirati pomoću različitih sustava specifikacije:

- nazivom boje (za imenovane boje)
- RGB specifikaciji Crveno-Zeleno-Plavo
- HSV specifikaciji
- HCL specifikaciji
- Munsellovom načinu specifikacije.

#### 2.1.1 Neke funkcije iz paketa *grDevices*

Ovaj paket sadrži funkcije koje podržavaju i osnovnu *base* i *grid* grafiku (grid grafika se temelji na koordinatnoj mreži, o kojoj će biti riječi kasnije). Za potpunu listu funkcija, unesite sljedeće:

```
library(help = "grDevices")
```

Osnovna grafika se većinom oslanja na *grDevices* paket za odabir boja, gdje se može birati između nekoliko paleta. Paket također, pruža niz osnovnih operacija za prijelaz iz jednog u drugi prostor boja kao što su **adjustcolor()**, **col2rgb()**, **make.rgb()**, **rgb2hsv()**, **convertColor()** te za izrađivanje korisničkih paleta **rgb()**, **hsv()**, **hcl()**, **gray()**, **colorRamp()**, **colorRampPalette()**, **densCols()**, **gray.colors()**.

Neke boje u R-u imaju nazive. Njihove nazive, 675 njih, može se pronaći na popisu koji se dobije korištenjem funkcija **colors()** ili **colours()**.

#### 2.1.1.1 Funkcija **colors()** ili **colours()** {*grDevices*}

Kao što smo već spomenuli, 675 boja u sustavu ima i deklariranu boju. Kako bi se korisnici što lakše koristili imenovanim bojama u sustavu, Stower institut je napravio vizualizaciju istih koja je dostupna na mrežnim stranicama: <http://research.stowers-institute.org/efg/R/Color/Chart/>.

Rezultat funkcije **colors()** je vektor naziva boja kojim se upravlja na jednak način kao i bilo kojim drugim vektorom u sustavu. U ovom slučaju to je vektor tipa *character*, tekst, pa se na njemu mogu primjeniti i sve funkcije unutar sustava koje rade s ovim tipom vektora, kao što je funkcija **grep()**.

- Odabiremo prema relativnoj poziciji vektora nazive boja:

```
colors() [1:10]
```

```
[1] "white"          "aliceblue"       "antiquewhite"   "antiquewhite1"
[5] "antiquewhite2" "antiquewhite3" "antiquewhite4" "aquamarine"
[9] "aquamarine1"   "aquamarine2"
```

- Odabiremo nazine iz vektora naziva boja koji sadrže tekst (engl. *string*) "blue" u imenu a do čega se može doći korištenjem sljedeće naredbe:

```
colors() [grep("blue", colors())]
```

```
[1] "aliceblue"      "blue"          "blue1"
[4] "blue2"          "blue3"          "blue4"
[7] "blueviolet"     "cadetblue"     "cadetblue1"
[10] "cadetblue2"    "cadetblue3"   "cadetblue4"
[13] "cornflowerblue" "darkblue"      "darkslateblue"
[16] "deepskyblue"   "deepskyblue1" "deepskyblue2"
[19] "deepskyblue3"  "deepskyblue4" "dodgerblue"
[22] "dodgerblue1"   "dodgerblue2"  "dodgerblue3"
[25] "dodgerblue4"   "lightblue"     "lightblue1"
[28] "lightblue2"     "lightblue3"   "lightblue4"
[31] "lightskyblue"  "lightskyblue1" "lightskyblue2"
[34] "lightskyblue3"  "lightskyblue4" "lightslateblue"
[37] "lightsteelblue" "lightsteelblue1" "lightsteelblue2"
[40] "lightsteelblue3" "lightsteelblue4" "mediumblue"
[43] "mediumslateblue" "midnightblue" "navyblue"
[46] "powderblue"     "royalblue"     "royalblue1"
[49] "royalblue2"     "royalblue3"   "royalblue4"
[52] "skyblue"         "skyblue1"     "skyblue2"
[55] "skyblue3"        "skyblue4"     "slateblue"
[58] "slateblue1"      "slateblue2"   "slateblue3"
[61] "slateblue4"      "steelblue"    "steelblue1"
[64] "steelblue2"      "steelblue3"   "steelblue4"
```

**ZA SAMOSTALAN RAD:** na način kojim smo izdvjili nazine svih boja koje u svojem nazivu sadrže niz znakova *blue* odvojite u posebne objekte sve boje koje u svojem nazivu sadržavaju nizove znakova *red* i *yellow*. Koliko postoji takvih boja u vektoru imenovanih boja?

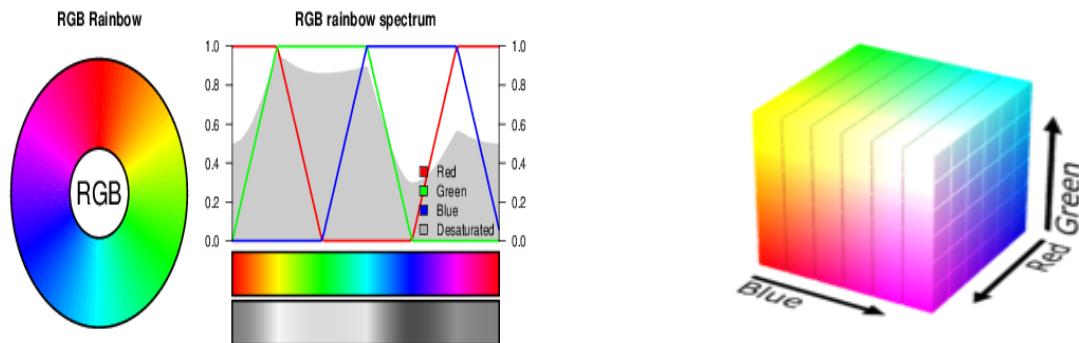
### 2.1.2 Specifikacija boja Crveno-Zeleno-Plavo (engl. Red-Green-Blue - RGB)

Ovo je najčešće korišten prostor boja (u dalnjem tekstu RGB). Svaka RGB boja se definira triom koji se sastoji od intenziteta za crvenu, zelenu i plavu valnu duljinu. Sustav se temelji na načinu na koji su konstruirani računalni zasloni i TV ekranii gdje je svaki piksel kombinacija tri svjetlosna izvora. Ako je samo jedna od tri boje na maksimalnom intenzitetu, a ostale su na nuli, tada je boja koja rezultira ili čisto crvena, čisto zelena ili čisto plava.

Ključni problem ove sheme boja je taj da se ne temelji na tome kako ljudsko oko vidi boje. Iz toga može nastati nekoliko problema kada se koriste palete boja koje se temelje na RGB-u. Jedno moguće rješenje je prebaciti se u drugi prostor boja gdje bi prvi izbori bilo korištenje HSV ili HSL specifikacije. Ove kratice znače Hue-Saturation-Value HSV i Hue-Saturation-Luminance HSL te se često koriste u izbornicima boja (engl. color pickers). Osnovni je problem taj da se, kako im i ime kaže (što se odnosi na dio naziva saturation (hrv. zasićenost)), temelje na zasićenosti. Stoga, ovakvi prostori boja ne rješavaju glavni problem skale boja RGB (iz <http://hclwizard.org/why-hcl/>).

Pored preciziranja boja u R-u pomoću imena boja, boje je moguće definirati i pomoću **RGB** specifikacije u obliku **#RRGGBB**, a koja se sastoji od dvije heksadecimalne znamenke koje daju vrijednost u rasponu od 00 (potpuno odsustvo) do FF (potpuna zasićenost). Neke funkcije u RGB sustavu trebaju drugačiji način preciziranja kombinacije crvene - zelene - plave boje, tako što se koristi raspon 0-255 za svaki segment, gdje se kreće od 0 za potpuno odsustvo specificirane RGB komponente i 255 za maksimalnu zasićenost.

#### 2.1.2.1 Funkcija `rgb(){grDevices}`



Slika 1: RGB model boja; Izvor:[https://en.wikipedia.org/wiki/RGB\\_color\\_model](https://en.wikipedia.org/wiki/RGB_color_model)

Ova funkcija izrađuje boje koje odgovaraju datim intenzitetima (između 0 i maksima) za crvenu, zelenu i plavu primarnu boju, nakon postavljanja na skalu u rasponu od 0-255. Detalje potražiti u R dokumentaciji funkcije. Kako bi se upoznali s funkcijom, potrebno je unijeti `?rgb` u R konzolu.

```
rgb((0:15)/15, green=0, blue=0, names=paste("red", 0:15, sep="_"))
```

```
red_0      red_1      red_2      red_3      red_4      red_5      red_6
"#000000" "#110000" "#220000" "#330000" "#440000" "#550000" "#660000"
red_7      red_8      red_9      red_10     red_11     red_12     red_13
"#770000" "#880000" "#990000" "#AA0000" "#BB0000" "#CC0000" "#DD0000"
red_14     red_15
"#EE0000" "#FF0000"

#postavljanje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

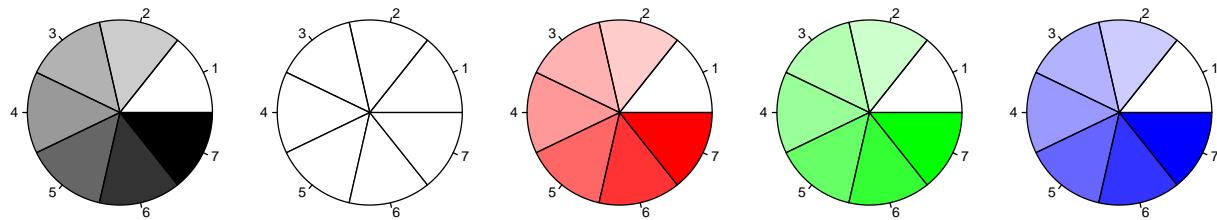
#izradivanje paleta
my_rgb_colors_min <- rgb(red=0, green=0, blue=0, alpha=c(0, 0.2, 0.3, 0.4, 0.6, 0.8, 1))
my_rgb_colors_max <- rgb(red=1, green=1, blue=1, alpha=c(0, 0.2, 0.3, 0.4, 0.6, 0.8, 1))
my_rgb_color_red <- rgb(red=1, green=0, blue=0, alpha=c(0, 0.2, 0.3, 0.4, 0.6, 0.8, 1))
my_rgb_color_green <- rgb(red=0, green=1, blue=0, alpha=c(0, 0.2, 0.3, 0.4, 0.6, 0.8, 1))
my_rgb_color_blue <- rgb(red=0, green=0, blue=1, alpha=c(0, 0.2, 0.3, 0.4, 0.6, 0.8, 1))

#vizualiziranje pripremljenih paleta
pie(rep(1,7), col= my_rgb_colors_min)
pie(rep(1,7), col= my_rgb_colors_max)
pie(rep(1,7), col= my_rgb_color_red)
pie(rep(1,7), col= my_rgb_color_green)
pie(rep(1,7), col= my_rgb_color_blue)
```

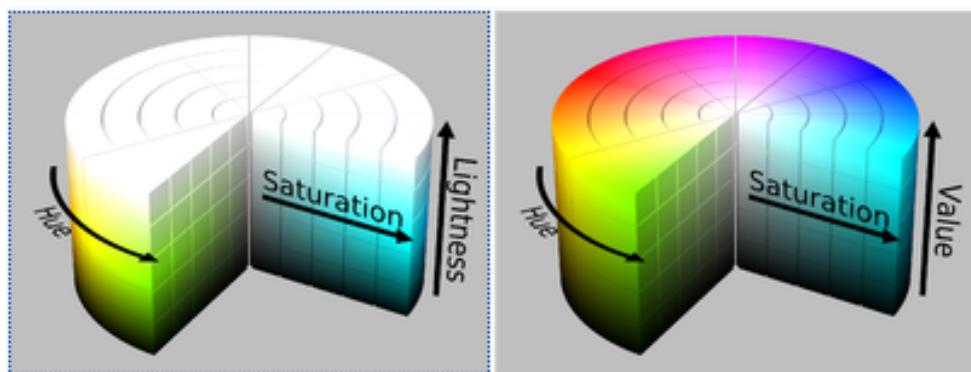
### 2.1.3 HSL/HSV specifikacija boja

Drugi model boja, koji tipično podržavaju softverski paketi, jest model koji uključuje specifikaciju boja kroz Zasićenost-Vrijednost-Osvjetljenje (engl. *Hue-Saturation-Value - HSV*), koje se dobivaju jednostavnom transformacijom RGB trija (u dalnjem tekstu *HSV*).

**HSL** i **HSV** su dvije naruobičajenije cilindrično-koordinatne prezentacije točaka u RGB modelu boja. Ove dvije prezentacije prelažu geometriju RGB-a uz pokušaj da ona bude intuitivnija i percepcijski relevantnija od dekartove (kubusne) reprezentacije. HSL i HSV su razvijeni tijekom 1970-ih godina za aplikacije računalne grafike. Danas se koriste u izbornicima boja (engl. *color pickers*)



Slika 2: Korisnički određena boja uz pomoć RGB sheme, prikazana kao pita dijagram s jednakim brojem podjela kao i boja u paleti - parametar alpha varira



Slika 3: HSL i HSV cilindri; Izvor: [https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV)

u softveru za uređivanje slika te, manje uobičajeno, u analizi slika i računalnoj vizualizaciji.

HSL znači nijansa, zasićenost i osvjetljenost te se često naziva i HLS. HSV predstavlja nijansu, zasićenost i vrijednost, a često se naziva i HSB (B za svjetlost, engl. *brightness*). Treći model koji je uobičajen u računalnim vizualnim primjenama jest HSI, što predstavlja nijansu, zasićenost i intenzitet. Iako su ove definicije obično konzistentne, one nisu standardizirane tako da se bilo koja od kratica može koristiti za bilo koji od navedena tri ili nekoliko drugih povezanih cilindričnih modela.

U svakom cilindru, kut oko središnje vertikalne osi odgovara „nijansi“, udaljenost od osi odgovara „zasićenosti“, a udaljenost duž osi odgovara „osvjetljenosti“, „vrijednosti“ ili „svjetlost“. Potrebno je uočiti da iako se „nijansa“ u HSL i HSV odnosi na isti atribut, njihove se definicije „zasićenosti“ značajno razlikuju.

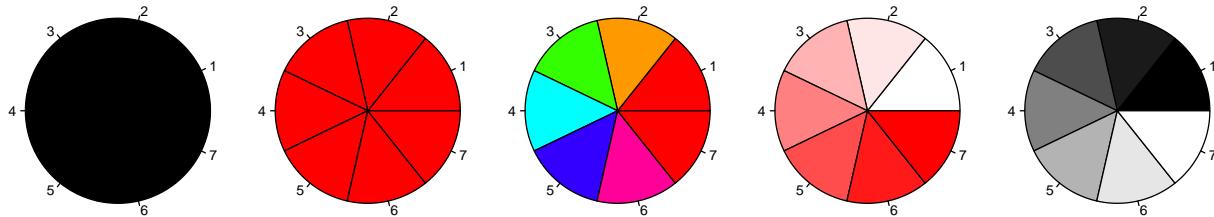
### 2.1.3.1 Funkcija **hsv()**{grDevices}

Funkcija **hsv()** koristi vrijednosti nijanse, zasićenosti i vrijednosti (u rasponu od 0 do 1) za specifikaciju boja. Funkcija prihvata ili jednu vrijednost za vektore vrijednosti ili vektore vrijednosti, i vraća heksadecimalne vrijednosti. Funkcija ima oblik **hsv(h=value, s=value, v=value, gamma=value, alpha=value)**. Funkciju **hcl()** može se gledati kao percepcijski temeljenu verziju **hsv()** funkcije.

```
#priprema paleta (sheme boja)
my_hsv_colors_000 <- hsv(h=0, s=0, v=0)
my_hsv_colors_111 <- hsv(h=1, s=1, v=1)
my_hsv_colors_h <- hsv(h=c(0,.1,.3, .5, .7,.9,1), s=1, v=1)
my_hsv_colors_s <- hsv(h=0, s=c(0,.1,.3, .5, .7,.9,1), v=1)
my_hsv_colors_v <- hsv(h=0, s=0, v=c(0,.1,.3, .5, .7,.9,1))

#postavljanje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,7), col= my_hsv_colors_000)
pie(rep(1,7), col= my_hsv_colors_111)
pie(rep(1,7), col= my_hsv_colors_h)
pie(rep(1,7), col= my_hsv_colors_s)
pie(rep(1,7), col= my_hsv_colors_v)
```

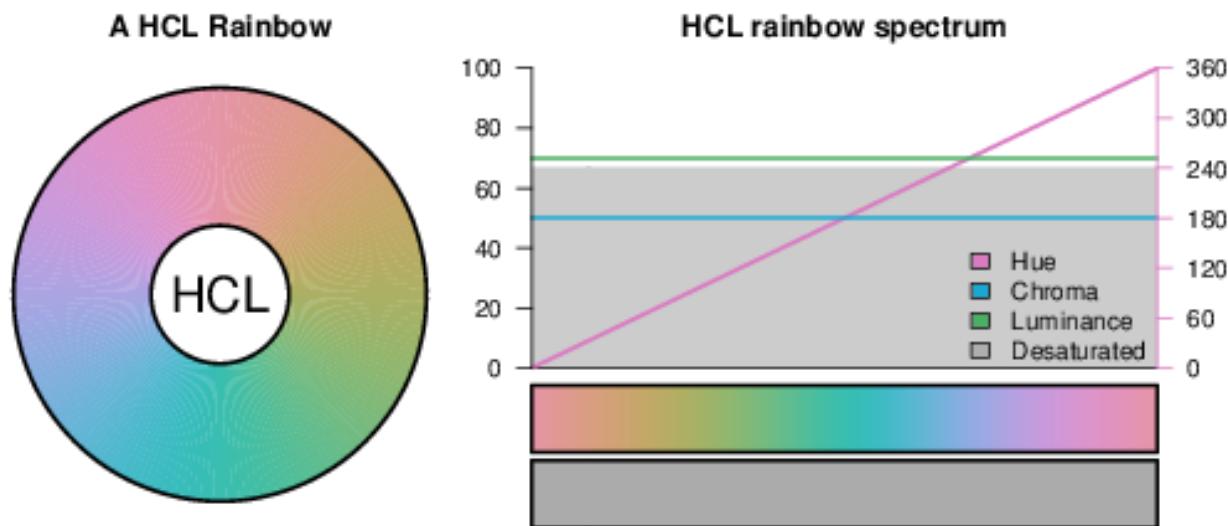


Slika 4: Korisničke boje određene HSV shemom; alfa =1 (default)

### 2.1.4 HCL specifikacija boja

*Hue-Chroma-Luminance* *HCL* prostor boja predstavlja alternativu prostorima boja poput RGB, HSV itd. Suprotno većini dostupnih prostora boja, HCL skala boja temelji se na tome kako ljudsko oko vidi boje. Svaka boja u HCL prostoru boja definira se trojkom vrijednosti. Kako je spomenuto, boje se tipično opisuju kao lokacije u trodimenzionalnim prostorima. Tri dimenzije koje ljudi tipično koriste za opisivanje boja su:

- nijansa: određuje boju
- zasićenost (šarolikost): definira obojenost
- svjetlost: definira svjetlinu (čistoću), količinu sive.



Slika 5: Boje definirane HCL sustavom; Izvor: <http://hclwizard.org/why-hcl>

Kao što smo već naveli, ljudsko oko ima dvije vrste stanica koje određuju naš vid od kojih jedne (čunjici) detektiraju različite dijelove elektromagnetskoga spektra te oko 20 puta više štapića, stanica mrežnice koje prihvataju intenzitet svjetla te su odgovorne za našu percepciju osvijetljenosti (kontrast tamno/svjetlo). HCL prostor boje temelji se na ovim osnovnim činjenicama o načinu funkciranja ljudskog oka i vida općenito. Unutar HCL sustava osoba/korisnik izravno kontrolira boju (nijansu), šarolikost (obojenost) i svjetlinu (čistoću, količinu sive). Ovo omogućava da se izbjegnu mnogi problemi vezani za RGB palete boja. Modeli boja koji nastoje obuhvatiti ove percepcijske osi nazivaju se i prostorima boja temeljeni na percepciji.

#### 2.1.4.1 Funkcija `hcl({grDevices}`

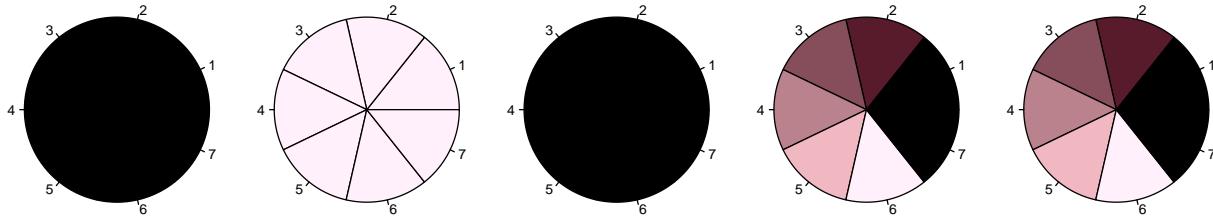
Ova funkcija je korisna za izrađivanje niza boja koje imaju približno jednake perceptualne promjene. Funkcija stvara vektor boja iz vektora koji preciziraju nijansu, obojenost i osvijetljenost.

Funkcija `hcl()` koristi vrijednosti nijanse, obojenosti i osvijetljenosti za jednoznačno određivanje boje. Funkcija prihvata bilo jedinstven skup vrijednosti ili vektore vrijednosti, te vraća vektor heksadecimalnih vrijednosti za raspon nijansi od 0 do 360. Nijansa boje koja je specificirana kao kut u rasponu [0,360]. Kut od 0 stupnjeva označava crvenu boju kut od 120 stupnjeva zelenu, a kut od 240 stupnjeva označava plavu boju. Raspon za obojenost ovisi o nijansi i osvijetljenosti, a raspon za osvijetljenost ovisi o nijansi i obojenosti. Za više detalja pogledati R dokumentaciju ali i primjer koji slijedi.

```
#priprema paleta
my_hcl_colors_0_35_0 <- hcl(h=0, c=35, l=0)
my_hcl_colors_360_35_100 <- hcl(h=360, c=35, l=100)
my_hcl_colors_h_35_0 <- hcl(h=c(0,70, 150, 220, 290, 360), c=35, l=0)
my_hcl_colors_0_35_1 <- hcl(h=0, c=35, l=c(0,20, 40, 60, 80, 100))
my_hcl_colors_360_35_1 <- hcl(h=360, c=35, l=c(0,20, 40, 60, 80, 100))

#definiranje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,7), col= my_hcl_colors_0_35_0)
pie(rep(1,7), col= my_hcl_colors_360_35_100)
pie(rep(1,7), col= my_hcl_colors_h_35_0)
pie(rep(1,7), col= my_hcl_colors_0_35_1)
pie(rep(1,7), col= my_hcl_colors_360_35_1)
```



Slika 6: Korisnička paleta određena HCL shemom s promjenjivim parametrima

Molim, komentirajte rezultate koje ste dobili.

### 2.1.5 Munsell specifikacija boje

U znanosti o bojama, Munsell sustav boja je prostor boja koji specificira boje temeljem tri dimenzije boja: nijansa, vrijednost (svjetlost) i obojenost (čistoća boje). Kreirao ga je profesor Albert H. Munsell početkom 20. stoljeća te ga je USDA prihvatile kao službeni sustav boja za istraživanje tla tijekom 1930-ih. Munsell sustav boja temelji se na nešto različitoj matematici ali ima određenu sličnost sa HCL sustavom.

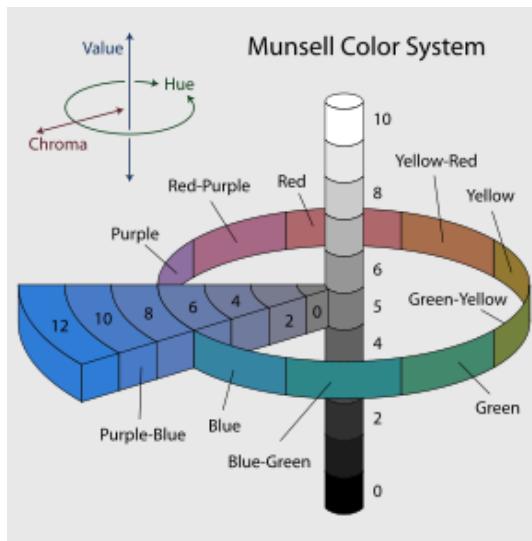
Paket **munsell** predstavlja okvir za rad s Munsell sustavom boja ([https://en.wikipedia.org/wiki/Munsell\\_color\\_system](https://en.wikipedia.org/wiki/Munsell_color_system)).

### 2.1.6 Prilagodba za daltonizam

U slučaju potrebe za prilagodbom za daltoniste, potrebno je upoznati se *dichroma* paketom, ali je to izvan djelokruga ovoga tečaja. Informacije o paketu i njegovim funkcijama prema potrebi pronađite na CRAN stranici paketa (<https://cran.r-project.org/web/packages/munsell/index.html>).

### 2.1.7 Prijelaz između sustava boja

Kako je već spomenuto, boje se u R-u mogu specificirati indeksom, nazivom, heksadecimalno, RGB-om i Munsellom. Na primjer, prva boja, bijela, može se predstaviti nazivom "white". RGB komponente bijele boje ekvivalentne su specifikaciji crvene, zelene,



Slika 7: Munsell model; Izvor: [https://en.wikipedia.org/wiki/Munsell\\_color\\_system](https://en.wikipedia.org/wiki/Munsell_color_system)

i plave redom: 255, 255, 255. Inter-operabilnost među opisanim sustavima je moguća i često je i neophodna <http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.pdf>.

#### 2.1.7.1 Funkcija `col2rgb(){grDevices}`.

Svaka od heksadecimalno definiranih boja koje su imenovane, mogu se prikazati u RGB formatu pomoću funkcije `col2rgb()`:

Koristite ovaj kôd da biste vidjeli RGB vrijednosti za sve imenovane boje (prvih 10 boja):

```
#u varijablu naziva nazivi_boja stavljamo imena imenovanih boja sustava
nazivi_boja <- colors()
#prebacujemo u RGB sustav
boje_rgb <- col2rgb(nazivi_boja)

#dodajemo atribut imena colona objektu klase matrix; preuzimamo iz vektora imena boja
colnames(boje_rgb) <- nazivi_boja

#transponiramo objekt radi lakše prezentacije - opcionalno
t(boje_rgb)[1:10,]
```

	red	green	blue
white	255	255	255
aliceblue	240	248	255
antiquewhite	250	235	215
antiquewhite1	255	239	219
antiquewhite2	238	223	204
antiquewhite3	205	192	176
antiquewhite4	139	131	120
aquamarine	127	255	212
aquamarine1	127	255	212
aquamarine2	118	238	198

U sljedećem primjeru, iz heksadecimalnog sustava na kojem boje specificira funkcija `terrain.colors()` prelazimo na RGB sustav boja na jednak način prikazan u sljedećem primjeru s 20 boja kreiranih iz `terrain.colors` paleta sustava R:

```
#prebacujemo u RGB sustav 20 boja paleta sustava terrain.colors
crgb <- col2rgb(cc <- terrain.colors(20))

#davanje atributa objektu
colnames(crgb) <- cc

#transponiranje radi lakše prezentacije - opcionalno
t(crgb)
```

	red	green	blue
#00A600FF	0	166	0
#13AD00FF	19	173	0
#28B400FF	40	180	0
#3EBB00FF	62	187	0
#56C200FF	86	194	0
#70C900FF	112	201	0
#8BD000FF	139	208	0
#A7D700FF	167	215	0
#C6DE00FF	198	222	0
#E6E600FF	230	230	0
#E7D217FF	231	210	23
#E8C32EFF	232	195	46
#E9B846FF	233	184	70
#EBB25EFF	235	178	94
#ECB176FF	236	177	118
#EDB48EFF	237	180	142
#EEBCA7FF	238	188	167
#FOC9C0FF	240	201	192
#F1DBD9FF	241	219	217
#F2F2F2FF	242	242	242

### 2.1.7.2 Funkcija col2hex(){grDevices}

Svaka imenovana boja može se prikazati u heksadecimalnom formatu korištenjem funkcije **col2hex()** na način prikazan u sljedećem primjeru za prvih deset imenovanih boja sustava:

Koristite sljedeće naredbekako bi se ispisale RGB vrijednosti za sve imenovane boje (prvih 10 boja):

```
boje_rgb <- col2rgb(imena_boja <- colors())
colnames(crgb) <- cc
t(crgb)[1:10,]
```

	red	green	blue
#00A600FF	0	166	0
#13AD00FF	19	173	0
#28B400FF	40	180	0
#3EBB00FF	62	187	0
#56C200FF	86	194	0
#70C900FF	112	201	0
#8BD000FF	139	208	0
#A7D700FF	167	215	0
#C6DE00FF	198	222	0
#E6E600FF	230	230	0

Isti se primjer može napraviti pomoću vektora različito specificiranih boja, kao što je slučaj u primjeru koji slijedi:

```
col2rgb(c("#4682B433", "#104E8b", "royalblue3"))
```

```
[,1] [,2] [,3]
red    70   16   58
green 130   78   95
blue   180  139  205
```

### 2.1.7.3 Funkcija convertColor(){grDevices}

Novija i fleksibilnija funkcija za prelazak između prostora boja je **convertColor(){grDevices}**. Funkcija omogućava konverziju boje između njihovih prikaza u standardnim prostorima boja. Kako biste se upoznali sa funkcijom unesite **?convertColor** u R konzolu. Postoje i cijeli kontribuirani paketi koji pružaju različite funkcije sa specifičnom sintaksom za inter-operabilnost među sustavima boja. Jedan od takvih paketa je i paket **colorspace**. Dodatne informacije o paketu, prema potrebi pronađite na službenim CRAN stranicama paketa (<https://cran.r-project.org/web/packages/colorspace/index.html>).

## ZA SAMOSTALNI RAD

- Napravite vektor duljine 8, definirajte nazive boja za daljnju uporabu. Dodijelite ovaj vektor objektu **moje\_boje**. Za pomoć, koristite sljedeći dokument: <http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>.
- Odaberite nazive svih imenovanih boja u R-u koje sadrže riječ *green* i dodijelite te vrijednosti karkatera objektu pod nazivom **green\_colors**. Koja je duljina objekta?
- Koji je naziv 53. boje po redu imenovanih boja u R-u?

## 2.2 Palete (sheme boja) u R-u

Općenito, razlikujemo tri tipa paleta:

- **Kvalitativne palete**

Kvalitativne palete su skupovi boja kojima se prikazuju različite kategorije tj. koriste se za kodiranje kategorijkih varijabli. Obično daju istu percepcijsku težinu svakoj kategoriji tako da se nijedna grupa ne percipira većom ili značajnjom od druge.

- **Sekvencijske palete**

Sekvencijske palete koriste se za kodiranje numeričkih informacija koje imaju raspon u određenom intervalu i gdje se niske vrijednosti smatraju nezanimljivim, a visoke zanimljivim.

- **Divergirajuće palete**

Divergirajuće palete se, također, koriste za kodiranje numeričkih informacija koje imaju raspon na određenom intervalu; međutim, ovaj interval sadrži i neutralnu vrijednost.

R nudi niz predloženih shema boja. S izuzetkom sivih, sve imaju istu sintaksu. Unesite se željeni broj nijansi koje želite sa sheme i funkcija ispiše vektor boja. Slijede neke korisne funkcije za rad s paletama u R-u iz *grDevices* i *RColorBrewer* paketa.

### 2.2.1 Funkcije za upravljanje paletama {grDevices}

#### 2.2.1.1 Funkcija palette(){grDevices}

Ovom funkcijom tražimo informacije o aktivnoj paleti u sustavu. Dodatno, funkcijom **palette()** možemo i upravljati paletom boja koja se koristi te možemo odrediti vlastitu, proizvoljnu paletu.

U primjeru ispod aktivna je paleta: black, red, green3, blue, cyan, magenta, yellow, gray što znači da u slučaju potrebe za samo jednom bojom na našem grafu, koristi se prva boja, a to je crna. Moguće je specificirati crnu kao željenu boju korištenjem grafičkoga parametra col=1 (kasnije će biti objašnjeno) (jer je ovdje crna boja prva boja u zadanoj paleti) ili col="black". Ako je potrebna druga boja, druga će se koristiti kao zadana, ovdje je to red; ako je potrebna treća boja, koristit će se *green3* itd.

### **palette()**

```
[1] "black"    "red"      "green3"   "blue"     "cyan"     "magenta" "yellow"
[8] "gray"
```

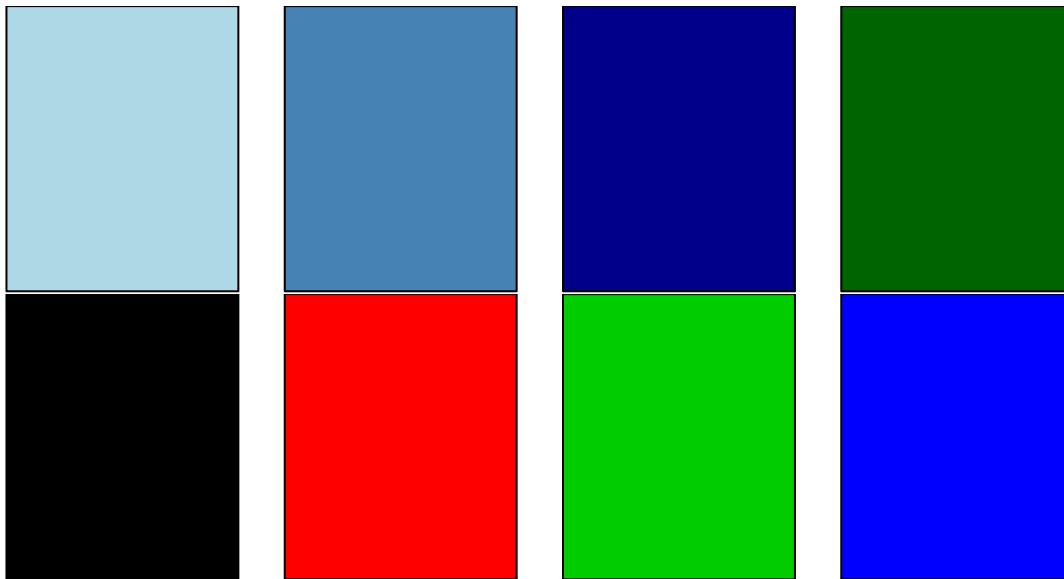
Povratak na prvobitnu paletu postiže se naredbom **palette("default")**.

Paleta boja može se promijeniti tako što ćemo zadati vektor boja:

```
par(mfrow=c(2,1), mar=c(0,0,0,0))
```

```
#postavljanje naše palete
palette(c("lightblue", "#4682B4", "#00008B", "darkgreen"))
barplot(c(1,1,1,1), axes=FALSE, col=c(1,2,3,4))

#povratak na zadanu (default) paletu
palette("default")
barplot(c(1,1,1,1), axes=FALSE, col=c(1,2,3,4))
```



Slika 8: Promjena zadane (default) palete sa željenom; četiri proizvoljne boje

#### 2.2.1.2 Funkcija **gray(){grDevices}**

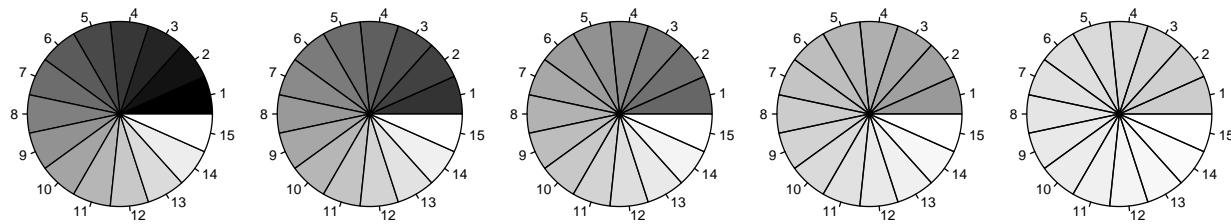
Funkcija kreira vektor boja iz vektora sivih razina. Funkcija **gray()** prima vrijednosti od 0 do 1, gdje 0 znači crno, a 1 znači bijelo. U kôdu koji slijedi, postavlja se vektor duljine numeričkih podataka kako biste dobili vektor sivih nijansi. Pita dijagrami su uobičajeni način vizualiziranja kreiranih shema boja te će se u tom kontekstu koristiti u svim daljnijim primjerima.

Ako želimo izraditi shemu boja (paletu) koja se sastoji od 15 sivih boja, a koje se protežu kroz cijeli spektar (0 - crno do 1 - bijelo) koristit ćemo sljedeći kôd:

```
#priprema paleta
my_gray_colors_1 <- gray((0:14/14), alpha=1)
my_gray_colors_08 <- gray((0:14/14), alpha=0.8)
my_gray_colors_06 <- gray((0:14/14), alpha=0.6)
my_gray_colors_04 <- gray((0:14/14), alpha=0.4)
my_gray_colors_02 <- gray((0:14/14), alpha=0.2)

#postavljanje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,15), col= my_gray_colors_1)
pie(rep(1,15), col= my_gray_colors_08)
pie(rep(1,15), col= my_gray_colors_06)
pie(rep(1,15), col= my_gray_colors_04)
pie(rep(1,15), col= my_gray_colors_02)
```



Slika 9: Korisnička paleta sivih boja, prikazana korištenjem pita dijagrama s jednakim brojem podjela; alpha argument varira

### 2.2.1.3 Funkcije `heat.colors()`; `rainbow()`; `topo.colors()`{grDevices}

Sustav R posjeduje nekoliko ugrađenih paleta boja. U biblioteci `grDevices` dostupno je pet funkcija za izradu vektora kontinuiranih boja iz ugrađenih paleta: `heat.colors()`, `cm.colors()`, `terrain.colors()`, `topo.colors()` i `rainbow()`. Ove funkcije se donekle razlikuju od funkcije `gray()`. Više detalja potražiti u R dokumentaciji. Otvorite dokumentacijsku karticu naredbom `?heat.colors` i upoznajmo se sa sintaksom.

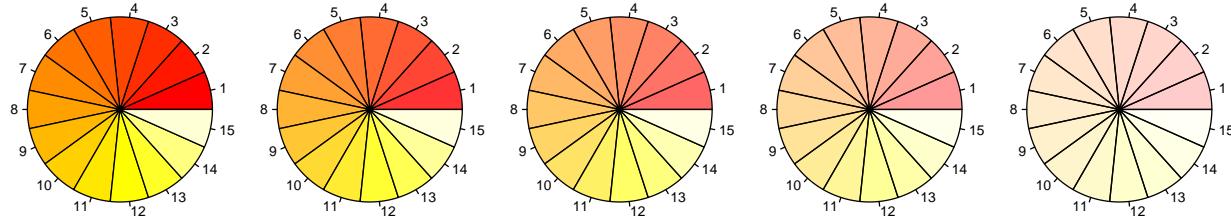
Primjeri izrade vektora kontinuiranih boja (`heat.colors()`, `rainbow()` i `topo.colors()`):

```
my_heat_colors_1 <- heat.colors(15, alpha=1)
my_heat_colors_08 <- heat.colors(15, alpha=0.8)
my_heat_colors_06 <- heat.colors(15, alpha=0.6)
my_heat_colors_04 <- heat.colors(15, alpha=0.4)
my_heat_colors_02 <- heat.colors(15, alpha=0.2)
```

```
#definiranje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
```

```
pie(rep(1,15), col= my_heat_colors_1)
pie(rep(1,15), col= my_heat_colors_08)
pie(rep(1,15), col= my_heat_colors_06)
pie(rep(1,15), col= my_heat_colors_04)
pie(rep(1,15), col= my_heat_colors_02)
```

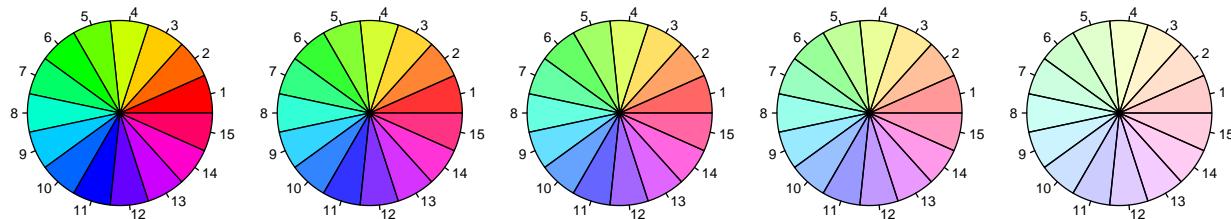


Slika 10: Izrađivanje vektora boja iz paleta engl. 'heat colors'; alpha argument varira

```
#radimo paletu boja
my_rainbow_colors_1 <- rainbow(15, alpha=1)
my_rainbow_colors_08 <- rainbow(15, alpha=0.8)
my_rainbow_colors_06 <- rainbow(15, alpha=0.6)
my_rainbow_colors_04 <- rainbow(15, alpha=0.4)
my_rainbow_colors_02 <- rainbow(15, alpha=0.2)

par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,15), col= my_rainbow_colors_1)
pie(rep(1,15), col= my_rainbow_colors_08)
pie(rep(1,15), col= my_rainbow_colors_06)
pie(rep(1,15), col= my_rainbow_colors_04)
pie(rep(1,15), col= my_rainbow_colors_02)
```



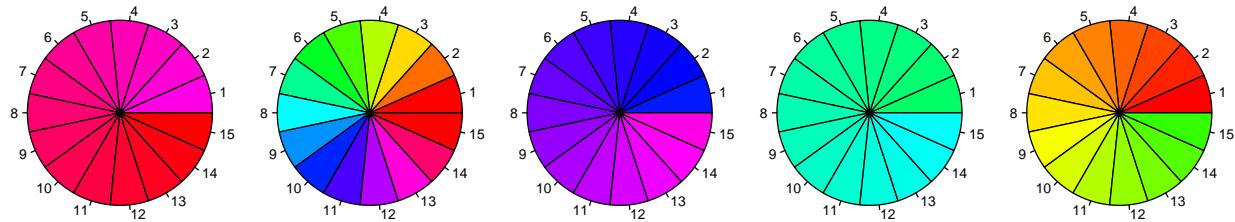
Slika 11: Izrađivanje vektora boja iz paleta engl. 'rainbow'; alpha argument varira

Funkcija **rainbow()** omogućava pod-raspone ukupnog spektra:

```
#stvaranje paleta boja
my_rainbow_colors_085_1 <- rainbow(15, start = 0.85, end = 1 )
my_rainbow_colors_065_1 <- rainbow(15, start = 0, end = 1)
my_rainbow_colors_065_085 <- rainbow(15, start = 0.65, end = 0.85)
my_rainbow_colors_04_05 <- rainbow(15, start = 0.4, end = 0.5)
my_rainbow_colors_0_03 <- rainbow(15, start = 0, end= 0.3)

par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,15), col= my_rainbow_colors_085_1)
pie(rep(1,15), col= my_rainbow_colors_065_1)
pie(rep(1,15), col= my_rainbow_colors_065_085)
pie(rep(1,15), col= my_rainbow_colors_04_05)
pie(rep(1,15), col= my_rainbow_colors_0_03)
```



Slika 12: Izrađivanje vektora boja iz palete engl. 'rainbow'; varira raspon

```
#radimo palete boja
my_topo_colors_1  <- topo.colors(15, alpha=1)
my_topo_colors_08 <- topo.colors(15, alpha=0.8)
my_topo_colors_06 <- topo.colors(15, alpha=0.6)
my_topo_colors_04 <- topo.colors(15, alpha=0.4)
my_topo_colors_02 <- topo.colors(15, alpha=0.2)
```

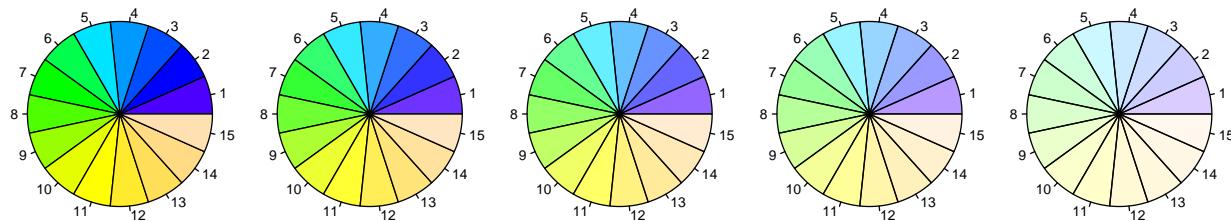
Ponovno ćemo vizualizirati palete pripremljene funkcijom **topo.colors()** željenim brojem boja i alpha parametrom.

```
#definiranje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#vizualiziranje pripremljenih paleta
pie(rep(1,15), col= my_topo_colors_1)
pie(rep(1,15), col= my_topo_colors_08)
pie(rep(1,15), col= my_topo_colors_06)
pie(rep(1,15), col= my_topo_colors_04)
pie(rep(1,15), col= my_topo_colors_02)
```

#### 2.2.1.4 Funkcija **colorRampPalette() {grDevices}**

Ova je funkcija slična funkciji **colorRamp{grDevices}** ali kao argumente broja boja koje se trebaju pripremiti ova funkcija uzima



Slika 13: Izrađivanje vektora boja iz palete engl. 'topo.colors'; alpha argument varira

cijeli broj kao vrijednost. Funkcija daje novu funkciju koja će generirati popis boja. Prvo ćemo se upoznati s funkcijom tako što ćemo unijeti **?colorRampPalette** u R konzolu.

```
#definiranje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0,0))

#stvaranje funkcije (funkcijom)
pal_1 <- colorRampPalette(colors=c("red", "blue"))

# odabir 5 boja iz palete
pal_1(15)

## [1] "#FF0000" "#EC0012" "#DA0024" "#C80036" "#B60048" "#A3005B" "#91006D"
## [8] "#7F007F" "#6D0091" "#5B00A3" "#4800B6" "#3600C8" "#2400DA" "#1200EC"
## [15] "#0000FF"

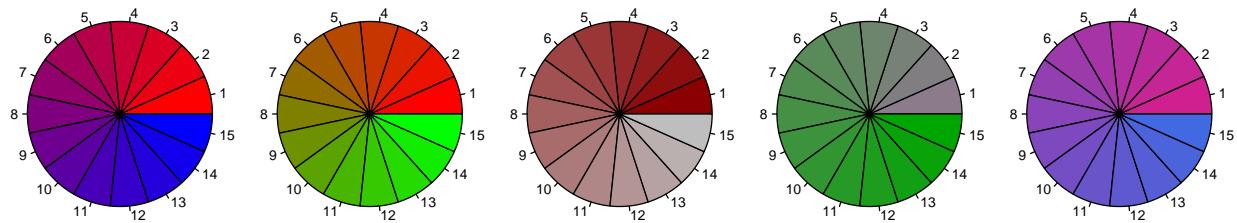
#interpolacija boja između određenih graničnika
my_colorRamp_1 <- pal_1(15)

#sve u jednom koraku
my_colorRamp_2 <- colorRampPalette(c("red", "green"))(15)
my_colorRamp_3 <- colorRampPalette(c("red4", "gray"))(15)
my_colorRamp_4 <- colorRampPalette(c("thistle4", "#00A600FF"))(15)
my_colorRamp_5 <- colorRampPalette(c("violetred", "royalblue"))(15)

#vizualizacija pripremljenih paleta
pie(rep(1,15), col= my_colorRamp_1)
pie(rep(1,15), col= my_colorRamp_2)
pie(rep(1,15), col= my_colorRamp_3)
pie(rep(1,15), col= my_colorRamp_4)
pie(rep(1,15), col= my_colorRamp_5)
```

Ponekad želimo izraditi vlastitu paletu kako bismo bojom označili faktorske varijable - razine faktora. U tom slučaju, uporabom funkcije **colorRampPalette()** potrebno je precizirati opcionalni argument **space** - što je mogućnost koja se koristi kada boje ne predstavljaju kvantitativnu skalu.

Za naprednije opcije u izradi shema boja koje definira korisnik paket *RColorBrewer* nudi još veći broj mogućnosti. Izvorni ColorBrewer osmisili su Cynthia Brewer i Mark Harrower te je bio namijenjen izradi mapa i kartiranju geografski referenciranih podataka, ali je pronašao svoj put i u grafikama tradicionalnih podataka. U biti, ovaj paket pruža prijedloge boja na osnovi kriterija kao što su broj željenih nijansi i tipu podataka - kategorički, sekvenčni ili divergentni. Interaktivna verzija može se pronaći na <http://colorbrewer2.org/>.



Slika 14: Izrađivanje funkcije kao rezultat druge funkcije **colorRampPalette()** sa zadanim brojem boja i graničnim bojama

## 2.2.2 Funkcije za upravljanje paletama {RColorBrewer}

Slijedi nekoliko korisnih funkcija iz paketa *RColorBrewer*.

### 2.2.2.1 Funkcija **display.brewer.all()**{RColorBrewer}

Paket *RColorBrewer* ima niz već izrađenih paleta (kvalitativne, sekvencijske i divergirajuće) koje možemo vidjeti pozivom funkcije **display.brewer.all()**.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,1), mai=c(1,1,1,1))

#prikaz svih paleta koje su raspoložive u paketu
display.brewer.all()
```

### 2.2.2.2 Funkcija **brewer.pal()**{RColorBrewer}

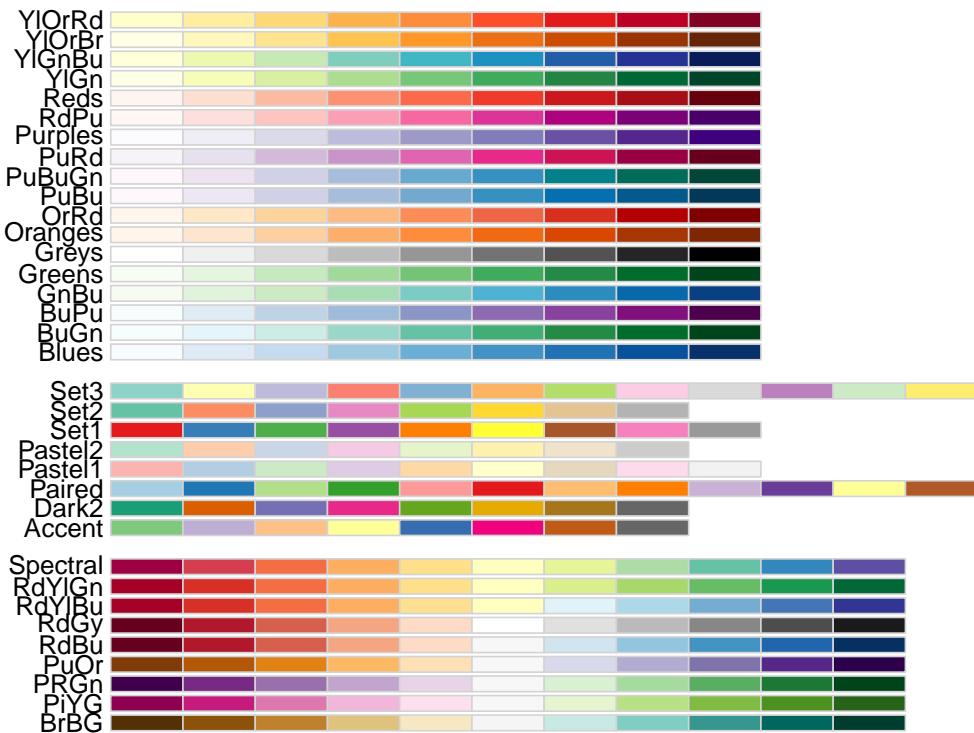
Funkcija omogućava raspoloživost paleta boja iz ColorBrewera u R-u. Kako biste naučili više o sintaksi i mogućnostima funkcije, unesite **?brewer.pal** u R konzolu.

Na primjer, ako želimo odabrati 15 boja iz palete **blues**, potrebno je napraviti sljedeće:

```
#postavljanje grafičkih parametara
par(mfrow=c(1,5), mar=c(0,0,0.0,0))

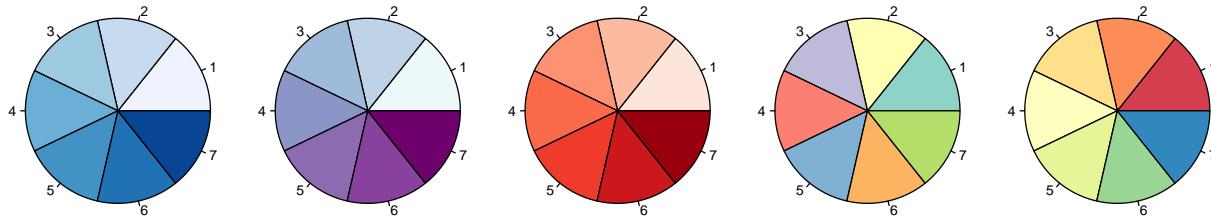
# odabir 7 boja iz pripremljenih paleta - RColroBrewer paket
my_brewer_1 <- brewer.pal(7,"Blues")
my_brewer_2 <- brewer.pal(7,"BuPu")
my_brewer_3 <- brewer.pal(7,"Reds")
my_brewer_4 <- brewer.pal(7,"Set3")
my_brewer_5 <- brewer.pal(7,"Spectral")

#vizualiziranje pripremljenih paleta
pie(rep(1,7), col= my_brewer_1)
pie(rep(1,7), col= my_brewer_2)
pie(rep(1,7), col= my_brewer_3)
```



Slika 15: Ranije pripremljene pelete iz *RColorBrewer* paketa, funkcija **display.brewer.all()**

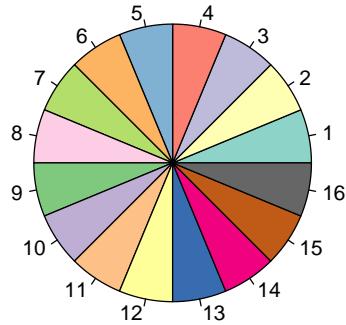
```
pie(rep(1,7), col= my_brewer_4)
pie(rep(1,7), col= my_brewer_5)
```



Slika 16: Izrađivanje paleta s *RColorBrewer* paketom **brewer.pal()**

Moguće je kombinirati palete prema svojim potrebama, što se vidi iz primjera koji slijedi:

```
par(mfrow=c(1,1), mar=c(0,0,0.0,0))
two_palettes <- c(brewer.pal(8, "Set3"), brewer.pal(8, "Accent"))
pie(rep(1,16), col= two_palettes)
```



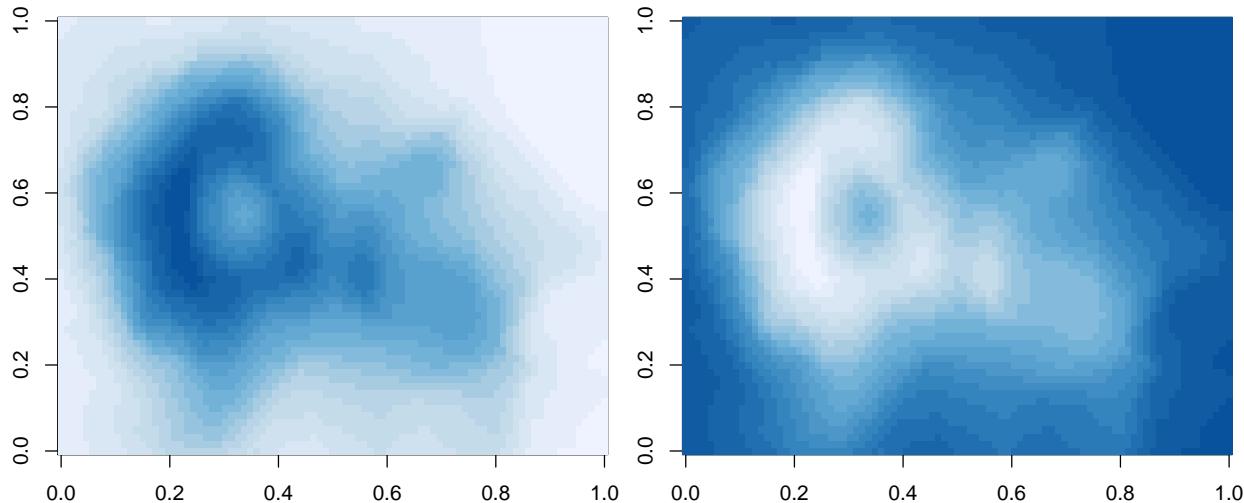
Slika 17: Izrađivanje palete kombinacijom dvaju paleta paketa *RColorBrewer*

Upravljenje bojama i paletama u sustavu R neograničeno je. Korisnik tako može kombinirati već postojeće palete unutar sustava ili čak unijeti standardne palete nekoga specifičnog paketa kao što su, primjerice, palete specijaliziranoga programa otvorenogog kôda za vizualizaciju i analitiku geografski referenciranih podataka SAGA GIS (<http://www.saga-gis.org/en/index.html>) putem RSAGA paketa koji ga veže na sustav R. U primjeru koji slijedi pripremljena paleta boja je kombinacija dvije palete paketa *RColorBrewer*.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,2), mar=c(2,2,0.5,1))

#radimo paletu od 20 plavih boja na osnovi pripremljene "BuPu" palete koja sadrži samo 9 plavih
my_colors <- colorRampPalette(brewer.pal(5,"Blues"))(20)

#vizualiziranje matrice sa željenim paletama
image(volcano, col= my_colors)
image(volcano, col= rev(my_colors))
```

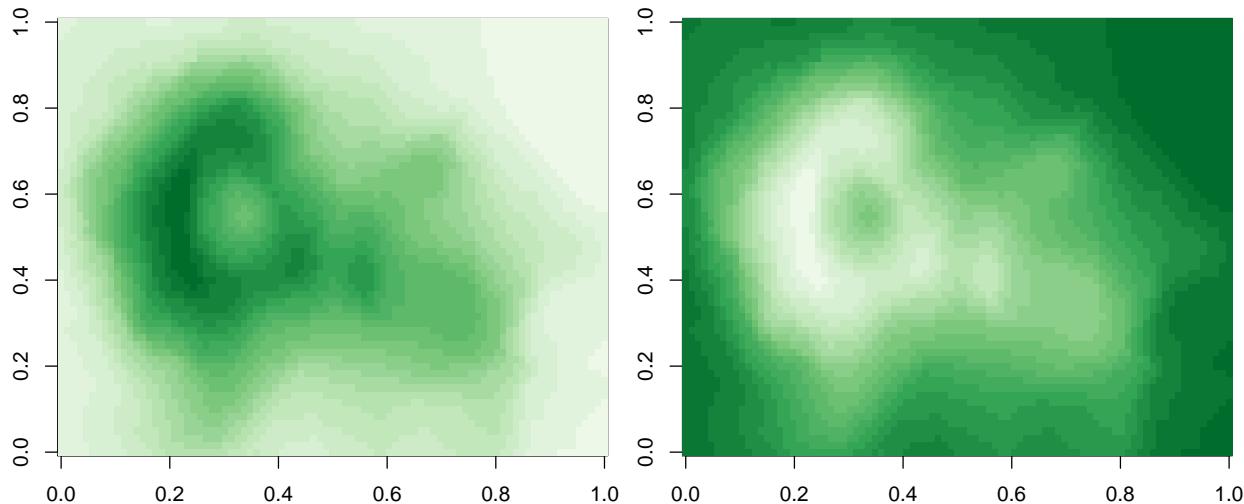


Slika 18: Izrađivanje korisničke palete kombinacijom funkcija paketa **graphics** i **RColorBrewer** primjenjeno na matrici *volcano*; Maunga Whau vulkan; a) paleta i b) reverzna shema boja - Blues

```
#postavljenje grafičkih parametara
par(mfrow=c(1,2), mar=c(2,2,0.5,1))

#radimo paletu od 20 plavih boja na osnovu pripremljene "Greens" palete
my_colors <- colorRampPalette(brewer.pal(5,"Greens"))(20)

#vizualiziranje matrice sa željenim paletama
image(volcano, col= my_colors)
image(volcano, col= rev(my_colors))
```



Slika 19: Izrađivanje korisničke palete kombinacijom funkcija paketa **graphics** i **RColorBrewer** primjenjeno na matrici *volcano*; Maunga Whau vulkan; a) paleta i b) reverzna shema boja - Greens

### 3 R grafika

Najznačajnija karakteristika postavke R grafike jest postojanje tri izdvojena sustava. Tako postoji tradicionalni sustav grafike (u daljem tekstu se na njega ponekad referira kao na osnovnu grafiku, *base* grafiku); *grid* grafički sustav ([http://www.e-reading.club/bookreader.php/137370/C486x\\_APPb.pdf](http://www.e-reading.club/bookreader.php/137370/C486x_APPb.pdf)) (u daljem tekstu se na njega ponekad referira kao na *lattice* grafiku) i *ggplot* sustav.

Grid grafika je jedinstvena za R i znatno je jača od tradicionalnog sustava. Većina funkcija koja producira potpune dijagrame korištenjem *grid* sustava dolazi iz *lattice* paketa Deepayan Sarkara, koji primjenjuje grafički sustav Billa Clevelanda, *trellis*. Tradicionalni grafički sustav primjenjuje mnoge tradicionalne grafičke mogućnosti S jezika ([https://en.wikipedia.org/wiki/S\\_%28programming\\_language%29](https://en.wikipedia.org/wiki/S_%28programming_language%29)). Za dodatno i sofisticiranije izrađivanje kompleksnih grafičkih prikaza, pogledati u knjizi **R graphics** (Murrell 2006) kao i u dokumentaciji novorazvijenih funkcija vizualizacije iz različitih kontribuiranih paketa.

Funkcije u sustavima grafike i grafičkim paketima mogu se podijeliti na tri glavna tipa:

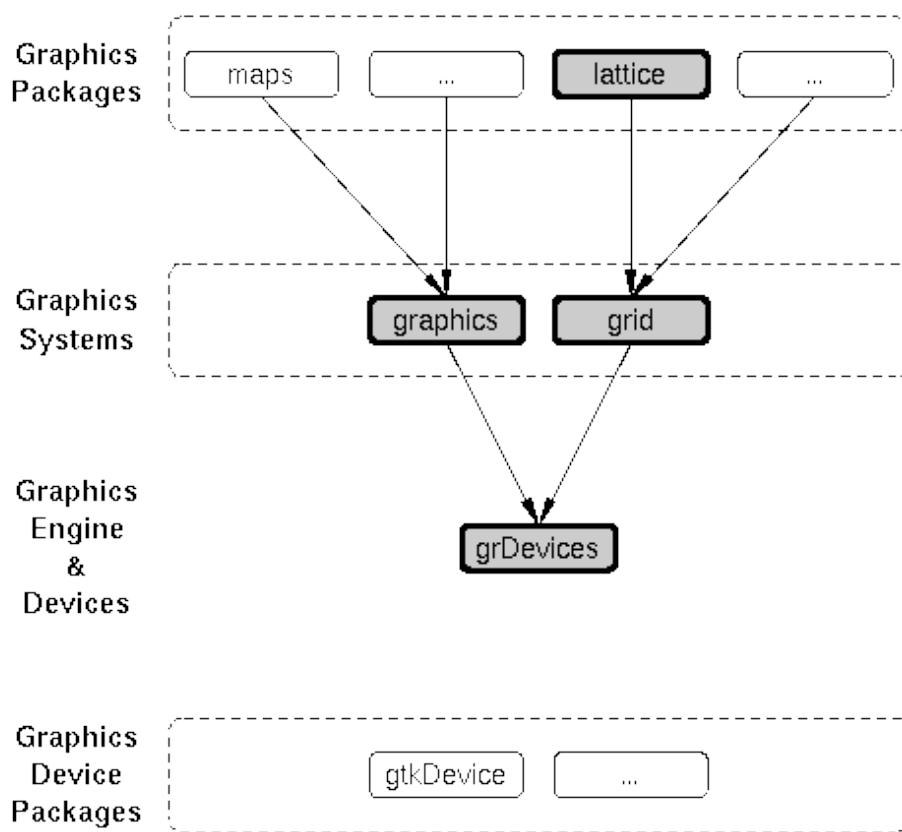
- 1) funkcije visoke razine (engl. *high-level*) koje rezultiraju (ili inicijaliziraju) potpune dijagrame; neki primjeri su: **plot**, **hist**, **boxplot**, ili **pairs**
- 2) funkcije niske razine (engl. *low-level*) koje daju dodatni ishod postojećim dijagramima koji su kreirani pomoću funkcija visoke razine; neki primjeri: **points**, **lines**, **text**, **axis**, **title**, **legend**, **abline**
- 3) funkcije za *interaktivni* rad s grafičkim uređajem.

Parovi funkcije visoke razine/*high-level* funkcije te funkcije niske razine/*low-level* funkcije bit će jednakopravno korištene u dalnjem tekstu. Slično vrijedi i za niz engleskih pojmova.

Jedna od najvećih prednosti jezika R je, svakako, mogućnost osnovne grafike. Tradicionalni sustav ili *base* sustav, temelji se na *graphics* paketu. Paketi temeljeni na paketu *graphics* pružaju spektar grafičkih funkcija visoke razine, čiji se rezultat može naknadno doradivati *low-level* funkcijama. Najznačajniji izuzetak je paket *lattice* koji pruža potpune dijagrame temeljem *grid* sustava. Postoje određene ograničene mogućnosti kombiniranja dva sustava korištenjem paketa **gridBase**, ali je to izvan djelokruga ovoga tečaja. Veoma je važno napomenuti da i nakon izrade novog dijagrama pomoću funkcije crtanja visoke razine (engl. *high-level*), možete ga dopuniti pozivanjem funkcija crtanja niske razine. Međutim, ovo se ne može napraviti nakon pozivanja Trellis funkcije paketa *lattice*.

R sustav grafike može se razložiti na četiri izdvojene cjeline: grafički paketi, grafički sustavi, pokretač grafike, uključujući standardne grafičke uređaje; i paketi grafičkih uređaja, što je prikazano na slici koja slijedi:

Ključna funkcionalnost R grafike, a koja je opisana u ovom priručniku, pruža se putem pokretača grafike i dva sustava grafike, tradicionalne grafike i grida. Grafički pokretač sastoji se od funkcija iz **grDevices** paketa i pruža temeljnu podršku u rukovanju stvarima poput boja i fontova a grafički uređaji proizvode izlazni rezultat u različitim formatima grafike. Tradicionalni grafički sustav (**base**) sastoji se od funkcija iz **graphics** paketa (<https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html>) a sustav grid grafike sastoji se od funkcija iz **grid** paketa (<https://stat.ethz.ch/R-manual/R-devel/library/grid/html/00Index.html>).



Slika 20: Organizacija R grafike. Izvor: [http://www.e-reading.club/bookreader.php/137370/C486x\\_APPb.pdf](http://www.e-reading.club/bookreader.php/137370/C486x_APPb.pdf)

### 3.1 Izlazni grafički formati (grafički uređaji - engl. *graphical devices*)

Uobičajen nači rada s grafičkim sustavima i vizualizacijom općenito je korištenje R sustava interaktivno - crtanje rezultata na zaslon. Vizualizacija podataka i međurezultata događa se u kontinuitetu. Uglavnom, tek konačni rezultat vizualizacije ili analize imamo potrebu izvesti kao grafiku visoke rezolucije na željeni medij (grafički uređaj, engl. *graphical devices*). Ponekad imamo potrebu izraditi dokument koji sadrži graf ili seriju statičkih grafova u željenim položajima. Također, veoma često smo limitirani pravilima publikacije u kojem formatu i rezoluciji primaju grafičke slike. Neki grafički formati (uređaji) više ili manje su standardni i prihvaćaju ih sve publikacije, kao što su PDF ili WMF, iz razloga što omogućavaju skalabilne slike. Formati poput bitmapa (BMP) trebaju se koristiti samo ako nema alternative.

Važno je napomenuti da svaki grafički uređaj ima svoj vlastiti skup grafičkih parametara koje možemo odrediti prema svojim potrebama. Ako nemamo aktivnih (otvorenih) grafičkih uređaja treba otvoriti isti prije postavljanja željenih grafičkih postavki. Koji je *default* grafički uređaj možemo pitati naredbom **options("device")**.

U drugom dijelu tečaja upoznat ćemo se s kontribuiranim R paketom *lattice* koji pruža funkcije *Trellis* grafičke. Kada se koristi *lattice* važno je navesti uređaj za koji postavljamo postavke korištenjem funkcije **trellis.device()** prije izdavanja grafičkih naredbi. Detaljnije o ovoj temi kasnije.

Unutar sustava R postoje funkcije koje omogućavaju rad s višestrukim grafičkim uređajima kao što su funkcija završetak postojećega i propagacija na novi uređaj ili graf **plot.new()** ili **frame()**. Neki integrirani razvojni sustavi za razvoj (engl. IDE), kao što je RStudio, ne dopuštaju postojanje više od jednog grafičkog prozora u istom trenutku. Iz tog razloga funkcije poput **windows()** (otvaranje novoga grafičkog prozora), **dev.set()** (definiranje aktivnoga grafičkog prozora) ili **graphics.off()** (zatvaranje svih otvorenih grafičkih uređaja) ne mogu se koristiti no ove funkcije je dobro imati na umu ako netko radi izravno u R-u ili nekom drugom IDE-u kao što je Tinn-R ili slični.

Kako bi se dobile informacije o zadanim grafičkim uređajima, kada se radi u RStudiju, u R konzolu unesite sljedeće:

```
options("device")
```

```
$device
function (width = 7, height = 7, ...)
{
  grDevices::pdf(NULL, width, height, ...)
}
<environment: namespace:knitr>
```

Što ste dobili kao rezultat prethodne funkcije **options("device")**? Kako se zove grafički uređaj RStudija?

Da bismo spremili rezultate vizualizacije napravljene u R-u, u neki od željenih grafičkih formata poput JPG, PNG, TIFF, PostScript, PDF ili nekog drugog s definiranim dimenzijama i rezolucijom, trebamo napraviti tri koraka:

- 1) otvaranje grafičkoga uređaja (npr. TIFF ili JPG)

```
jpeg('my_plot.jpg')
```

- 2) naš opcionalni graf na uređaju

```
plot()
```

- 3) zatvaranje uređaja.

```
dev.off()
```

Jedan jednostavan primjer izvoza visokorezolutne grafičke iz R-a u PNG datoteku s preciziranim dimenzijama i rezolucijom:

```
#otvorite uređaj s opcijama
png(filename = "iris_dataset.png", width=15, height=15, units="cm", res=300) #open the device

#postavljanje grafičkih parametara za otvoreni uređaj
par(mfrow=c(2,2))

#definiranje željene palete koja će se koristiti u grafu
```

```
palette(c("lightblue", "#4682B4", "#00008B", "darkgreen"))

#prvi graf (opcionalni graf)
hist(iris$Sepal.Length, col="lightblue", main="Histogram Sepal.Length")
rug(iris$Sepal.Length)

#drugi grafikon, dva histograma na jednom prikazu
hist(iris$Sepal.Width, col="lightblue", main="Histogram Sepal.Width /Sepal.Length")

hist(iris$Sepal.Length, col="lightgray", add=T)

#treći graf
plot(iris$Sepal.Width,iris$Sepal.Length, col=iris$Species)

#četvrti graf
hist(iris$Petal.Width, col="lightblue")
rug(iris$Petal.Width)

#zatvaranje uređaja
dev.off()
```

```
pdf  
2
```

Svaki uređaj ima svoj vlastiti skup grafičkih parametara. Ako je postojeći uređaj nulti uređaj, funkcija **par()** će otvoriti novi uređaj prije upita za ispitivanje/postavljanje parametara.

Korisnik može kontrolirati format zadanog uređaja korištenjem funkcije **options()**. Neki grafički uređaji, npr. PDF, dopuštaju višestruke stranice kao izlazni rezultat, dok drugi to ne omogućavaju (PNG, JPG itd.).

## 4 Glavni grafički sustavi u R-u

Grafika je uvijek bila jedna od najbolje razvijenih segmenata sustava R. Njegova grafika pruža neovisnost uređaja, funkcije crtanja visoke razine (engl. *high-level*) koje produciraju cijeli prikaz. Dodatno, sustav sadržava i funkcije niske razine koje dopunjavaju postojeći prikaz kao skup grafičkih parametara koji pružaju širok spektar kontrole nad detaljima crtanja.

### 4.1 Osnovna grafika (engl. base)

Osnovnu R grafiku karakterizira to što:

- slijedi proces ljudskoga razmišljanja
- daje odlične rezultate koji su potrebni kako bi se upravljalo svakim detaljem
- **par()** je korisna funkcija za ispitivanje, postavljanje i učenje o svim vrstama grafičkih parametara
- tipični tijek rada može se opisati kao: izrađivanje "praznog" ili "NULL" grafa koji samo postavlja koordinatni sustav. U sljedećim koracima korisnik postepeno dodaje željene grafičke elemene kao što su podaci, naslov, legenda, osi i mnoge druge.

Sustav osnovne R grafike temelji se na konceptu crtanja područja okruženih marginama. Na korisniku je da, ako želi, stvoriti višestruke grafove, svaki sa svojim marginama, a sve njih, kao cjelinu, definira vanjskom marginom. Kasnije ćemo vidjeti i upoznati se sa znatno drugačijim konceptom *viewport* u *lattice* grafici.

#### 4.1.1 Paket *graphics*

Ovaj paket sadrži funkcije *base* grafike. Kako smo već i naveli, *base* grafika je tradicionalna S grafika i najstariji je grafički sustav unutar R-a. U ovom ćemo se dijelu upoznati sa skupom najvažnijih/najčešćih funkcija paketa. Za sve ostale mogućnosti potrebno je posjetiti web stranicu paketa na CRAN-u (<https://stat.ethz.ch/R-manual/R-devel/library/graphics/html/00Index.html>).

Kao i obično, informacije o paketu mogu se dobiti korištenjem kratice za R dokumentaciju; funkcija **Question(){}{utils}** paketa. Unosom **?graphics** u R konzolu dobit ćemo informacije o odgovarajućem kôdu kojim se može dobiti potpuni popis funkcija paketa s njihovim opisima **library(help = "graphics")**.

Za citiranje paketa, koristite funkciju **citation(){}{utils}** paketa na sljedeći način:

```
citation("graphics")
```

The 'graphics' package is part of R. To cite R in publications use:

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

```
@Manual{,
  title = {R: A Language and Environment for Statistical Computing},
  author = {{R Core Team}},
  organization = {R Foundation for Statistical Computing},
  address = {Vienna, Austria},
  year = {2016},
  url = {https://www.R-project.org/},
}
```

We have invested a lot of time and effort in creating R, please cite it when using it for data analysis. See also 'citation("pkgname")' for citing R packages.

R osnovna grafika pruža uobičajeni spektar standardnih statističkih dijagrama, uključujući dijagrame raspršivanja, box dijagrame, histograme, bar dijagrame, pita dijagrame i osnovne 3D dijagrame. U R-u se ovi osnovni tipovi dijagrama mogu proizvesti jednim pozivanjem funkcije, ali se dijagrami koji tako nastanu mogu smatrati polaznom osnovom za složenije prikaze. Mogućnost dodavanja nekoliko grafičkih elemenata zajedno, kako bi se kreirao konačni rezultat, predstavlja temeljnu karakteristiku R grafike.

#### 4.1.1.1 Funkcija par(){graphics}

Ovo je jedna od najznačajnijih funkcija iz paketa **graphics**. S tom funkcijom moguće je postaviti ili propitati grafičke parametre koji se mogu promjeniti/definirati funkcijom **par()** ili **?par**. Prvih 10 parametara navedeni su ispod:

```
par() [1:10]
```

```
$xlog
[1] FALSE
```

```
$ylog
[1] FALSE
```

```
$adj
[1] 0.5
```

```
$ann
[1] TRUE
```

```
$ask
[1] FALSE
```

```
$bg
[1] "white"
```

```
$bty
[1] "o"
```

```
$cex
[1] 1
```

```
$cex.axis
[1] 1
```

```
$cex.lab
[1] 1
```

Da bi se imena svih parametara koji postoje vidjela poredana po abecedi, unesite:

```
sort(names(par()))
```

```
[1] "adj"      "ann"      "ask"      "bg"       "bty"
[6] "cex"      "cex.axis"  "cex.lab"   "cex.main"  "cex.sub"
[11] "cin"      "col"       "col.axis"  "col.lab"   "col.main"
[16] "col.sub"  "cra"       "crt"      "csi"      "cxy"
[21] "din"      "err"       "family"   "fg"       "fig"
```

```
[26] "fin"        "font"        "font.axis"   "font.lab"    "font.main"
[31] "font.sub"    "lab"         "las"        "lend"       "lheight"
[36] "lmitre"     "lty"         "lwd"        "ljoin"      "mai"
[41] "mar"         "mex"         "mfcol"      "mfg"        "mfrow"
[46] "mpg"         "mkh"         "new"        "oma"        "omd"
[51] "omi"         "page"        "pch"        "pin"        "plt"
[56] "ps"          "pty"         "smo"        "srt"        "tck"
[61] "tcl"         "usr"         "xaxp"       "xaxs"       "xaxt"
[66] "xlog"        "xpd"         "yaxp"       "yaxs"       "yaxt"
[71] "ylbias"     "ylog"
```

Kako je već spomenuto, svaki uređaj ima svoj skup grafičkih parametara. Ako je trenutačni uređaj null (engl. *null*), par će onda otvoriti novi uređaj prije propitivanja/postavljanja parametara (koji uređaj je kontroliran preko funkcije **options("device")**).

### Neki značajni grafički parametri

#### 4.1.1.1 Grafički parametri koji kontroliraju veličinu teksta i simbola

Neki od najznačajnijih grafičkih parametara za rad s tekstom (engl. *string*) su:

- **cex** - broj koji ukazuje na veličinu po kojoj će se tekst i simboli skalirati prilikom crtanja u odnosu na njihove zadane vrijednosti. 1= predodređena vrijednost (engl. *default*), 1.5 je 50% veća, 0.5 je 50% manja, itd. Na isti način može se kontrolirati i druga vrsta teksta koji se pojavljuje na dijagramu, poput povećavanja naslova/podnaslova, tekstova na osi i sl. Funkcije koje nam omogućavaju da kontroliramo određeni dio teksta imaju intuitivne nazive:
- **cex.axis** - povećavanje oznaka osi u odnosu na cex
- **cex.lab** - povećavanje x i y naziva u donosu na cex
- **cex.main** - povećavanje naslova u odnosu na cex
- **cex.sub** - povećavanje podnaslova u odnosu na cex.

Kako bi se dobile informacije o zadanoj vrijednosti specifičnog grafičkog parametra, u ovom slučaju **cex** parametra, u R konzolu se treba unjeti sljedeće:

```
par("cex")
```

```
[1] 1
```

U primjeru koji slijedi pokazat ćemo uporabu parametra **cex**.

Ako pogledamo sliku ispod, molim da odgovorite koje su vrijednosti **cex** parametra na grafovima koji slijede:

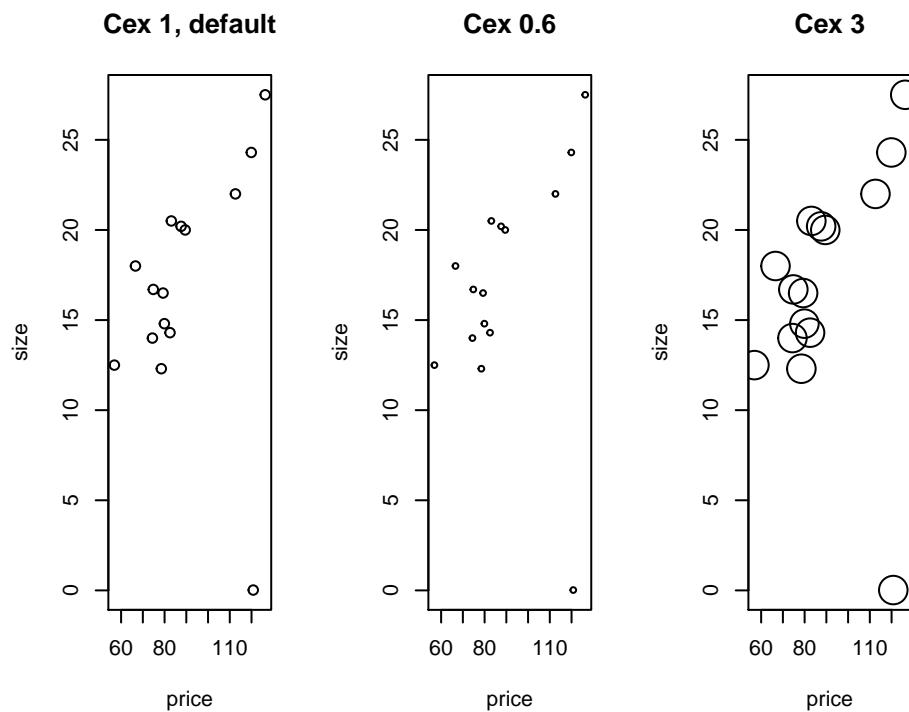
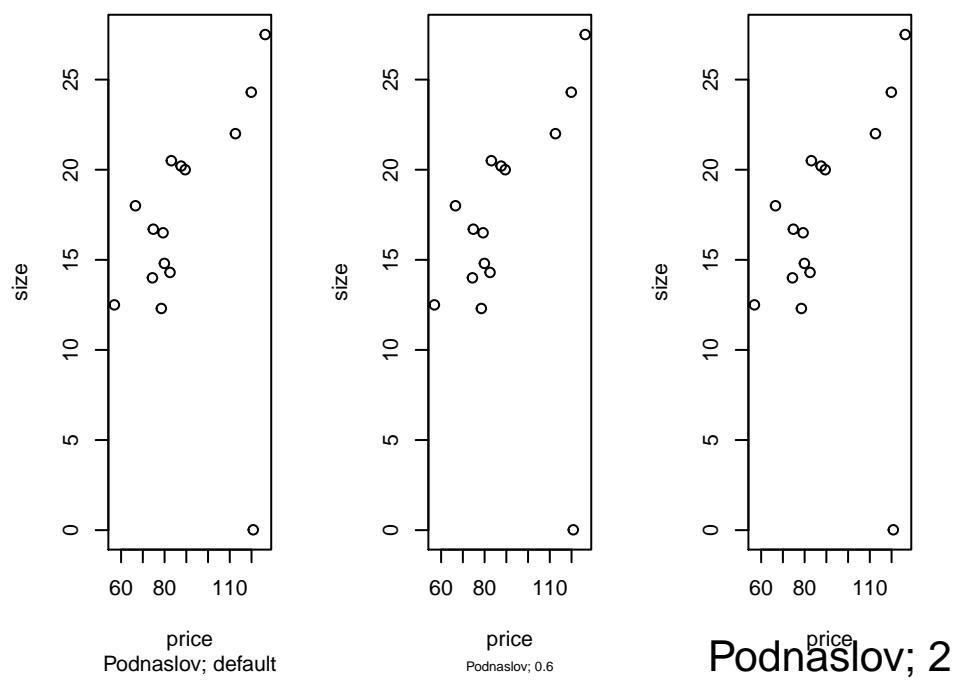
Slično tako, ako želimo manipulirati veličinom drugih elemenata grafa, poput veličine podnaslova, funkcija/parametar **cex.sub()** se treba promjeniti. Pogledajte primjer ispod i komentirajte rezultat.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#prvi graf
plot(prices, sub="Podnaslov; default")

#drugi graf
plot(prices, sub="Podnaslov; 0.6", cex.sub=.6)

#treći graf
plot(prices, sub="Podnaslov; 2", cex.sub=2)
```

Slika 21: Grafički parametar - primjer korištenja za veličinu simbola **cex** parametarSlika 22: Grafički parametar - veličina ispisa podnaslova **cex.sub**

#### 4.1.1.1.2 Grafički parametri koji kontroliraju simbole za točke na grafu (**pch**)

Svako se opažanje u skupu podataka može prikazati kao jedna točka u koordinatnom sustavu. Ako nacrtamo sva opažanja kao jedinstven skup točaka, koristit će se jedna vrsta simbola na grafu, ona koja je zadana.

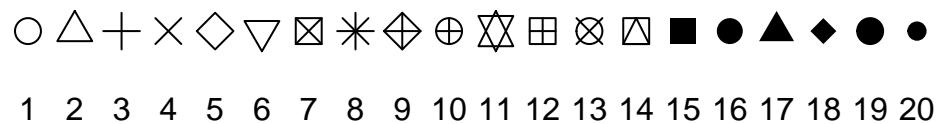
Kako bi se dobili podaci o zadanoj vrijednosti specifičnoga grafičkog parametra, u ovom slučaju **pch** parametra, u R konzolu treba unijeti sljedeće:

```
par("pch")
```

```
[1] 1
```

Ako si sami želimo prikazati koji su to najpopularniji simboli, možemo upisati sljedeće linije kôda:

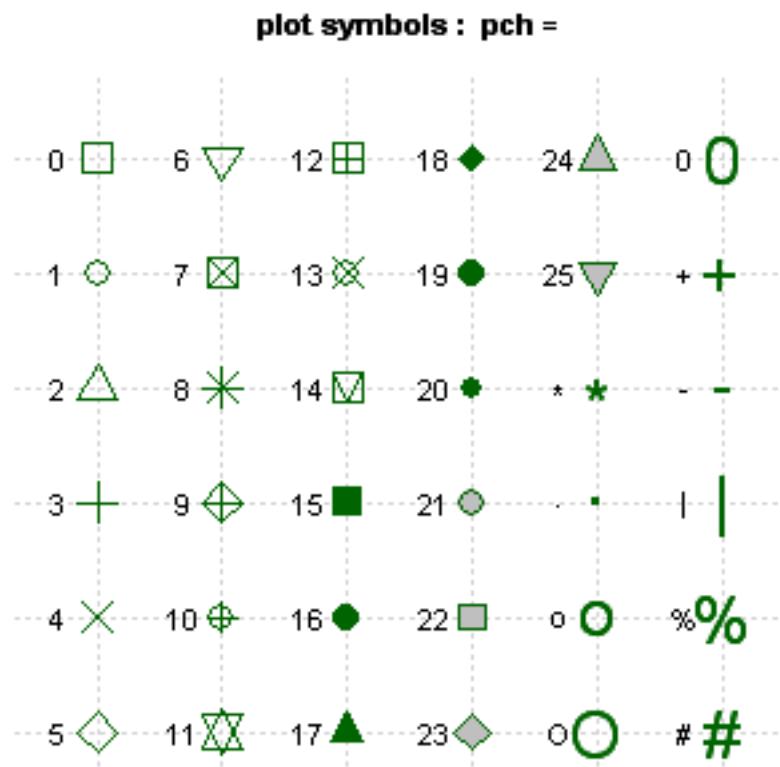
```
number_pch <- 20
x <- rep(1,number_pch)
```



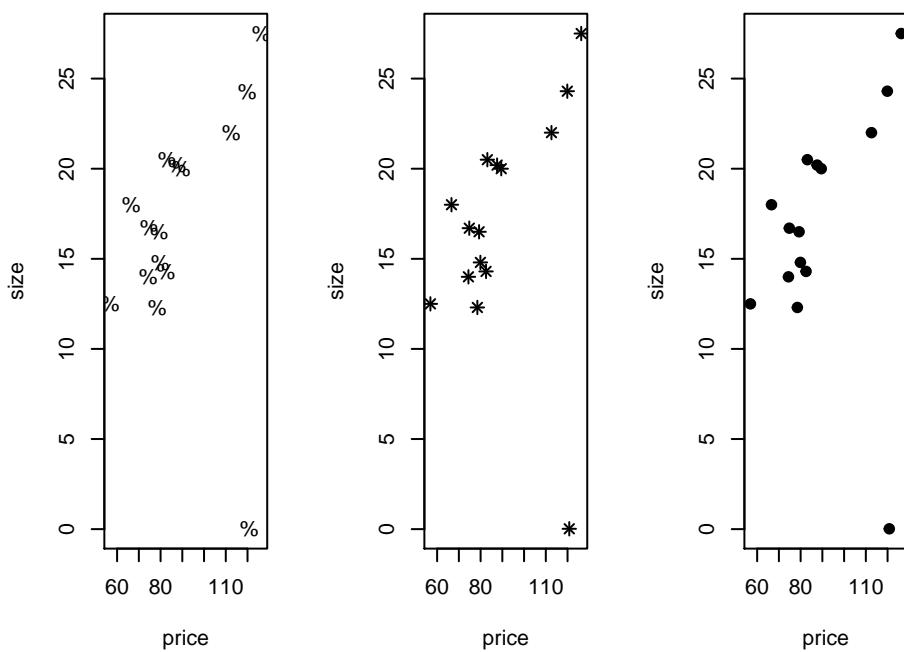
Slika 23: Grafički parametar - tip točke; prvih 20 **pch** parametar

Slika koja slijedi prikazuje skupinu najpopularnijih simbola koji se koriste za prikaz opažanja na grafovima, ali su omogućene i dodatne mogućnosti. Sa slike koja slijedi vidimo da parametar **pch** = 1 znači prazan kružić.

Odgovorite koje su vrijednosti **pch** parametra na grafovima koji slijede:



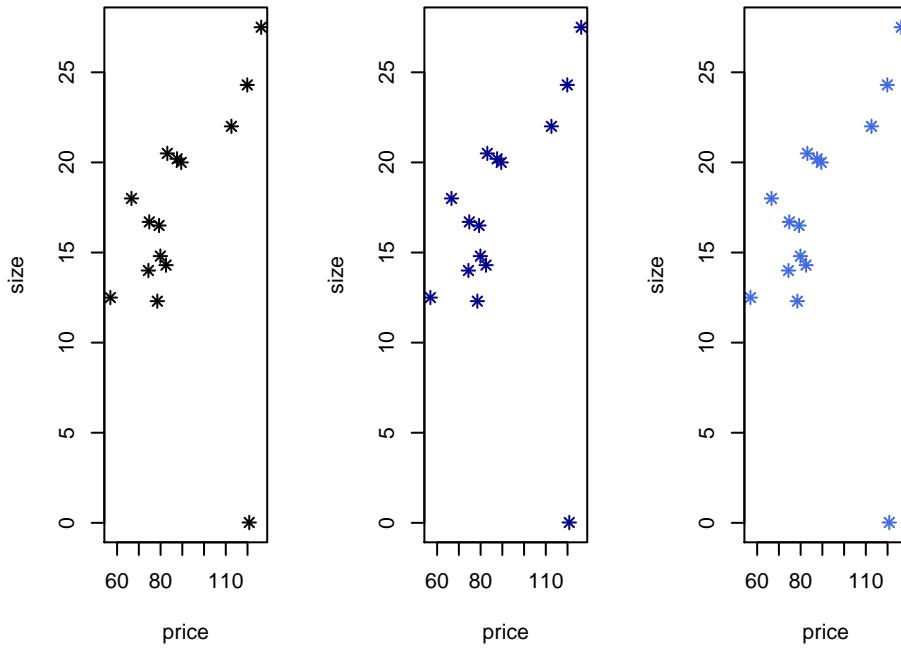
Slika 24: Grafički parametar - tip točke; **pch** parametar. Izvor: <http://www.statmethods.net/advgraphs/parameters.html>



Slika 25: Grafički parametar - tip točke; **pch** parametar

#### 4.1.1.1.3 Grafički parametri koji kontroliraju boju elemenata grafa (`col`)

U primjeru koji slijedi promijenit ćemo boju točaka na dijagramu dok će `pch` parametar ostati konstanta:



Slika 26: Grafički parametar - boje elementa grafa, `col` parametar

Pogledajmo prvih 6 elemenata u skupu podataka `iris` koji dolazi sa sustavom R te strukturu skupa podataka.

```
#prvih nekoliko podataka
head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

```
#struktura podataka
str(iris)
```

```
'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species      : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

Obratite pozornost na to da je jedna od varijabli u skupu podataka faktorska varijabla koja sadrži informacije o vrstama biljaka.

Ostale četiri varijable su kvantitativne varijable i opisuju određene morfološke karakteristike; mjere listića i latica biljke. Pored informacija o kvantitativnim varijablama s grafa, željeli bismo dati dodatne informacije o vrsti biljke (kategorija faktora) kojoj svaka točka na grafu pripada. Najjednostviji način da to napravimo je da svakom opažanju iz različite kategorije faktorske varijable dodijelimo drugačiji simbol točke ili boju. U ovom primjeru ćemo dodijeliti različitu boju za svaku razinu faktora varijable *Species*. Kvantitativne varijable od interesa su *Petal.Length* i *Petal.Width*. Želimo obojati nivoe faktora *Species* drugom bojom.

I dalje koristimo generičku funkciju `plot`, dajući dva kvantitativna argumenta, i tada je graf koji se dobije kao rezultat dijagram raspršivanja. Pored toga, zadat ćemo određeni vektor boja za svaki nivo faktorske varijable. U prvom slučaju, za nivoe faktora će biti će korištene predodređene boje `default` palette dok ćemo u drugom slučaju ćemo za svaki nivo faktora sami odrediti boju kojom će biti prikazani nivoi.

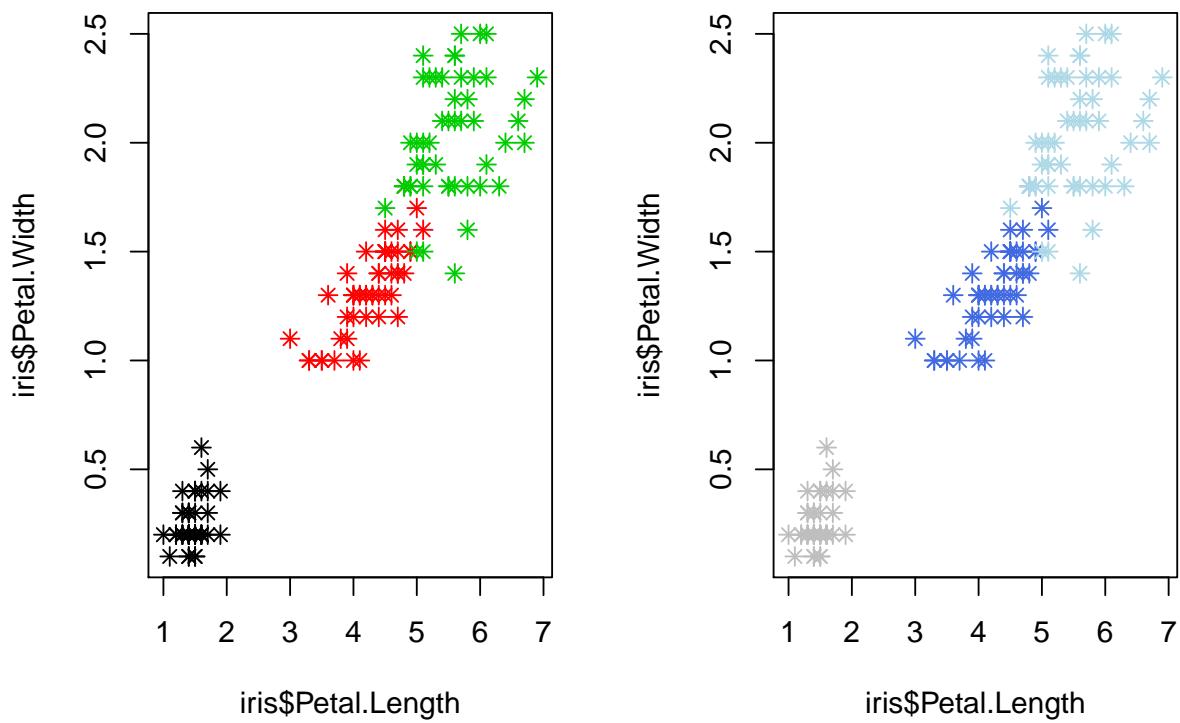
Dijagram raspršivanja je graf dvije kvantitativne varijable u jednom koordinatnom sustavu, ravnini. Ovo je svakako jedan od najznačajnijih grafičkih načina prikaza veze dviju kvantitativnih varijabli.

```
#postavljanje grafičkih parametara
palette("default")
par(mfrow=c(1,2))

#prvi graf
plot(iris$Petal.Length,
      iris$Petal.Width,
      pch=8,
      col=iris$Species)

#izradivanje boja
iris_colors <- c("gray", "royalblue", "lightblue")

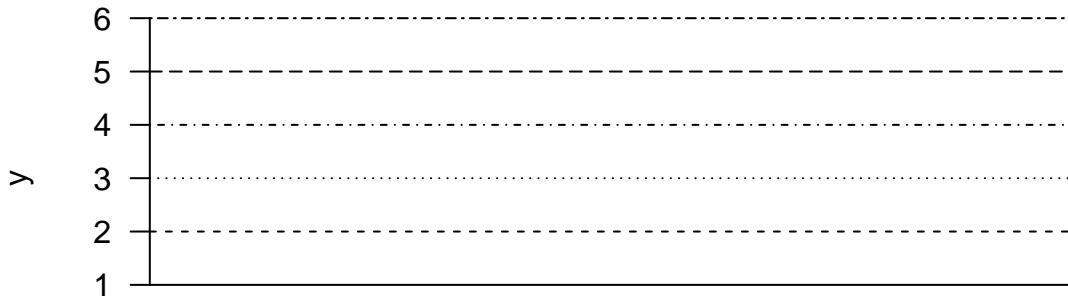
#drugi graf
plot(iris$Petal.Length,
      iris$Petal.Width,
      pch=8,
      col= iris_colors[iris$Species])
```



Slika 27: Bojanje prema nivou faktora jedne varijable u skupu podataka; default boje palete i b) zadane boje

#### 4.1.1.1.4 Grafički parametri koji kontroliraju vrste linija (*lty*)

Postoji šest unaprijed definiranih vrsta linija u sustavu R. Ipak, postoji i mogućnost da se definiraju i određene, prilagođene linije. Vrsta linije može se precizirati kao unaprijed definirani cijeli broj ili kao unaprijed definirano ime u obliku teksta (engl. string), ili kao heksadecimalni znakovi koji preciziraju neku vrstu prilagođene linije: primjeri vrsta linija mogu se vidjeti kao grafički parametri funkcije **abline** (funkcija **abline** će biti objašnjena kasnije: može se proizvoljno dodati prava linija postojećem dijagramu. Linije se mogu definirati kao precizirane horizontalne/vertikalne linije ili linije određene argumentima nagiba i konstante). U primjeru koji slijedi crtamo horizontalne linije čiji je položaj definiran argumentom *h*.



Slika 28: Unaprijed određeni tipovi linija u sustavu R

```
#postavljanje grafičkih parametara
par(mfrow=c(1,1), mar=c(1,4,0.5,1))

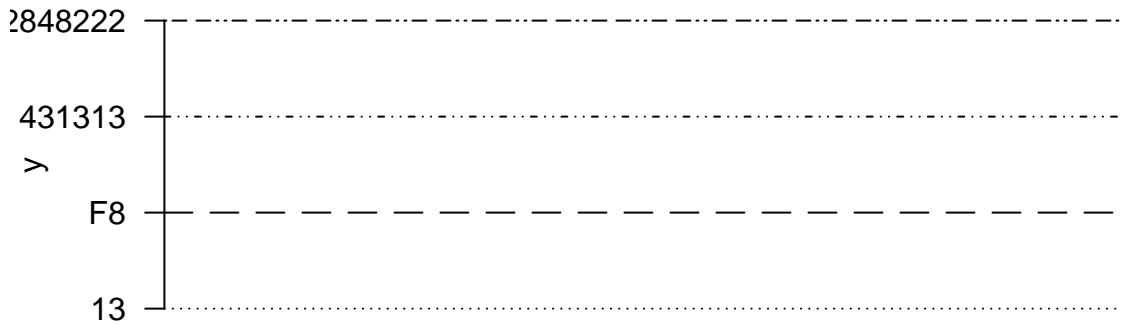
#priprema podataka za crtanje
x <- 1:7
y <- 1:7

#prvi graf
plot(x, y,
      xlim=c(1, 5),
      ylim=c(0,5),
      type="n",
      axes=F,
      xlab=NULL,
      ylab=NULL)

#dodavanje linija
abline(h=1, lty="13")
abline(h=2, lty="F8")
abline(h=3, lty="431313")
abline(h=4, lty="22848222")

#definiranje osi
```

```
axis(2,
  at=c(1:4),
  ylab=NULL,
  labels=c("13", "F8", "431313", "22848222"),
  cex.lab=0.4,
  tick = TRUE,
  las=1)
```



Slika 29: Korisnički tipovi linija

Pogledajte primjer i prokomentirajte rezultat:

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#priprema podataka
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#prvi graf
plot(c(0,3), c(0,1.3), type="n", xlab="x", ylab="y")

#dodavanje linija postojecem dijagramu
lines(x,
       y1,
       col="red4",
       lwd=2)

lines(x,
       y2,
       col="darkseagreen",
       lty=4,
       lwd=2)
```

```

#drugi graf
plot(c(0,3), c(0,1.3), type="n", xlab="x", ylab="y")

#dodavanje linija postojecem dijagramu
lines(x,
       y1,
       col="#00A600FF",
       lwd=2)

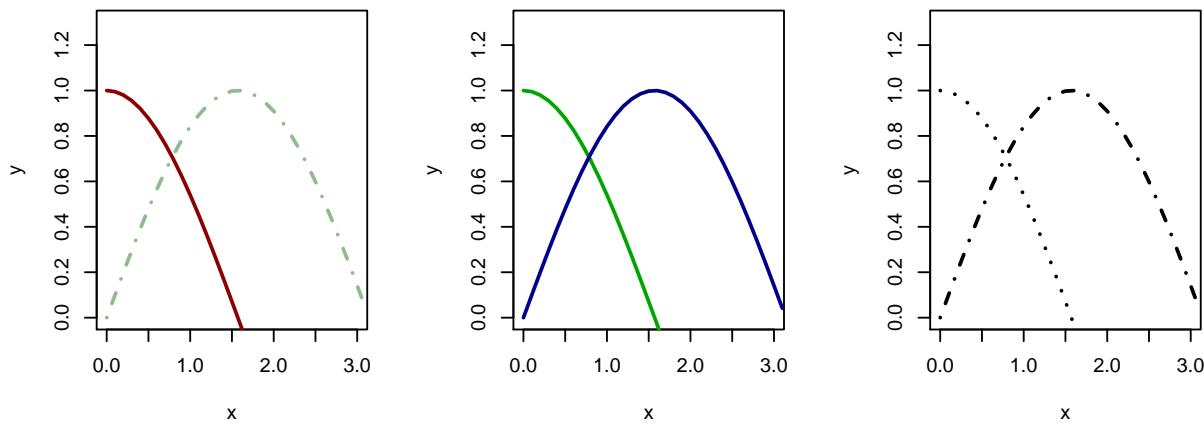
lines(x,
       y2,
       col="darkblue",
       lwd=2)

#treći graf
plot(c(0,3), c(0,1.3), type="n", xlab="x", ylab="y")

#dodavanje linija postojecem dijagramu
lines(x,
       y1,
       lty=3,
       lwd=2)

lines(x,
       y2,
       lty=4,
       lwd=2)

```



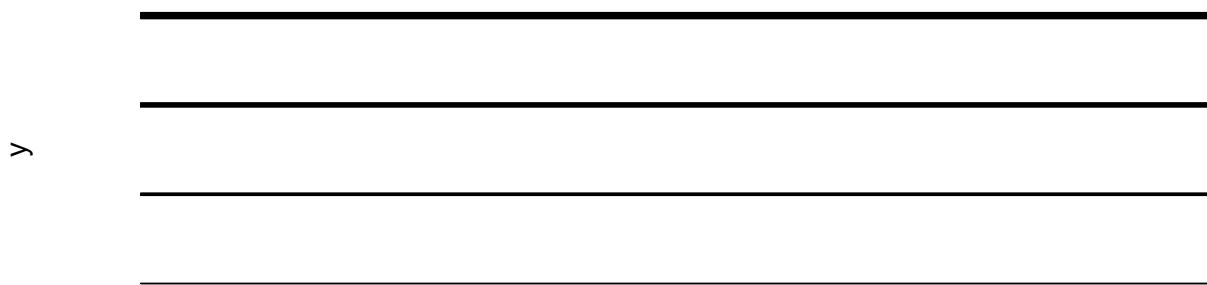
Slika 30: Dodavanje elemenata na postojeći graf - linije; parametar **lty** i **col** variraju

#### 4.1.1.5 Grafički parametri koji kontroliraju širinu linije (*lwd*)

Postoji jednostavan način kontroliranja širine (debljine) linije u sustavu. Komentirajte kôd koji slijedi:

```
par(mfrow=c(1,1), mar=c(1,4,0.5,1))
x <- 1:7
y <- 1:7
plot(x, y,
      xlim=c(1, 5),
      ylim=c(0,5),
      type="n",
      axes=F,
      xlab=NULL,
      ylab=c())

abline(h=1, lwd=1)
abline(h=2, lwd=2)
abline(h=3, lwd=3)
abline(h=4, lwd=4)
```



Slika 31: Grafički parametri - debljina linije **lwd** parametar

#### 4.1.1.1.6 Grafički parametri koji kontroliraju regiju crtanja (*mfcol/mfrow*); *layout()* i *split.screen()*

Vrlo korisna mogućnost R sustava je mogućnost definiranja vlastitoga područja crtanja. U primjeru koji slijedi podijelit ćemo prostor za crtanje na četiri područja, 1) dva retka i dva stupca i 2) jedan redak s četiri stupca.

```
# postavljanje grafičkih parametara
par(mfrow=c(2,2))

#definiranje boja za dijagram
palette(c("lightblue", "#4682B4", "#00008B", "darkgreen"))

#prvi graf (funkcija visoke razine)
hist(iris$Sepal.Length, col="lightblue", main="Histogram Sepal.Length")

#unaprijeđenje prvog dijagrama (niska razina)
#dodavanje vrijednosti x osi
rug(iris$Sepal.Length)

#drugi graf
hist(iris$Sepal.Width,
      col="lightblue", main="Histogram Sepal.Width")

rug(iris$Sepal.Width)

#treći graf
plot(iris$Sepal.Width,iris$Sepal.Length,
      col=iris$Species, main="Dijagram raspršenja (engl. *scatterplot*)")

#četvrti dijagram
hist(iris$Petal.Width, col="lightblue")
rug(iris$Petal.Width)

# postavljanje grafičkih parametara za otvoreni uređaj (jedan red s četiri stupca)
par(mfrow=c(1,4))

#definiranje boja za dijagram
palette(c("lightblue", "#4682B4", "#00008B", "darkgreen"))

#prvi graf (funkcija visoke razine)
hist(iris$Sepal.Length, col="lightblue", main="Histogram Sepal.Length") # opcionalni dijagram

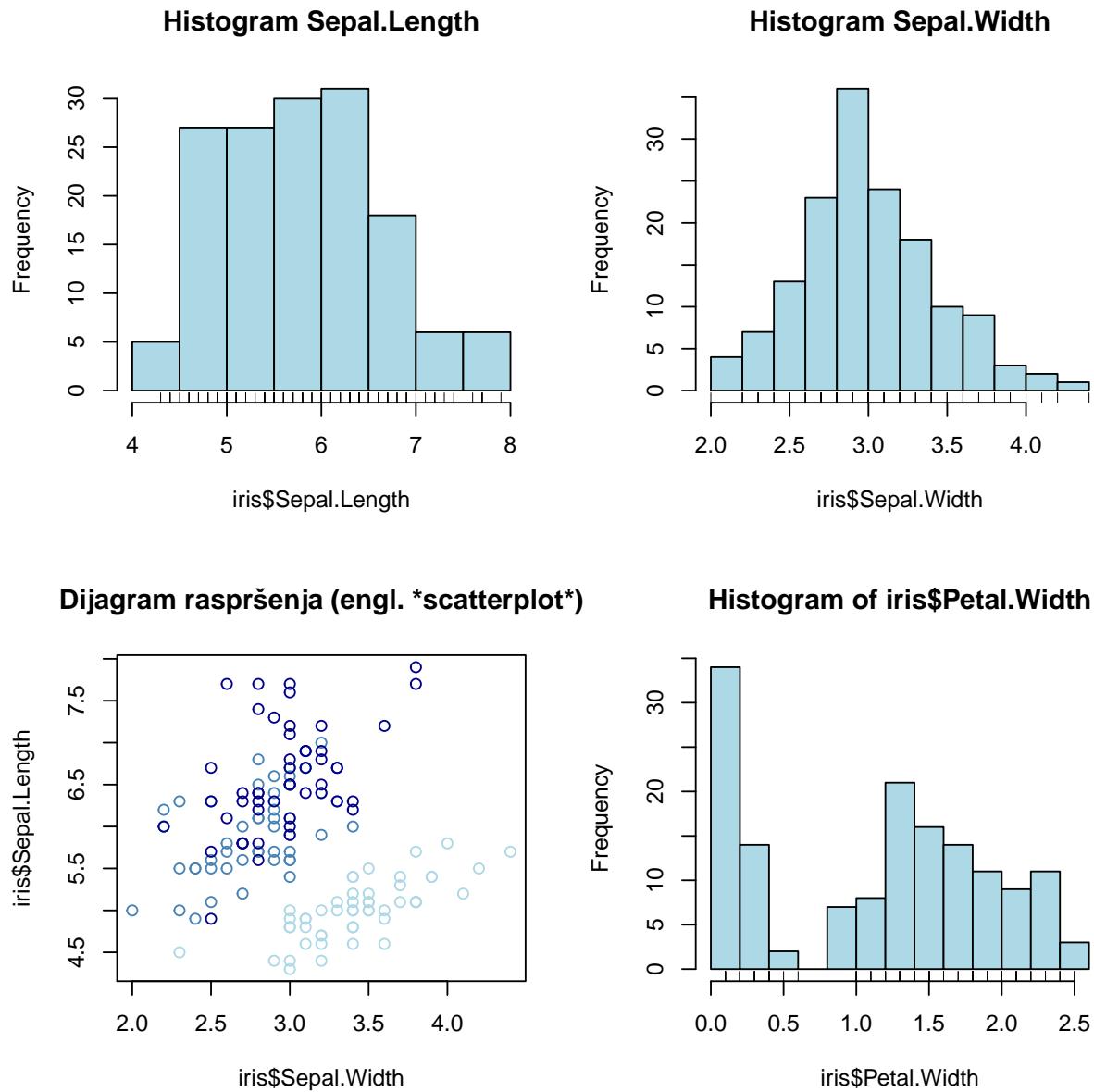
#unapređenja prvog dijagrama (engl. *low-level* funkcijom)
#dodavanje vrijednosti x osi
rug(iris$Sepal.Length)

#drugi graf
hist(iris$Sepal.Width, col="lightblue", main="Histogram Sepal.Width")

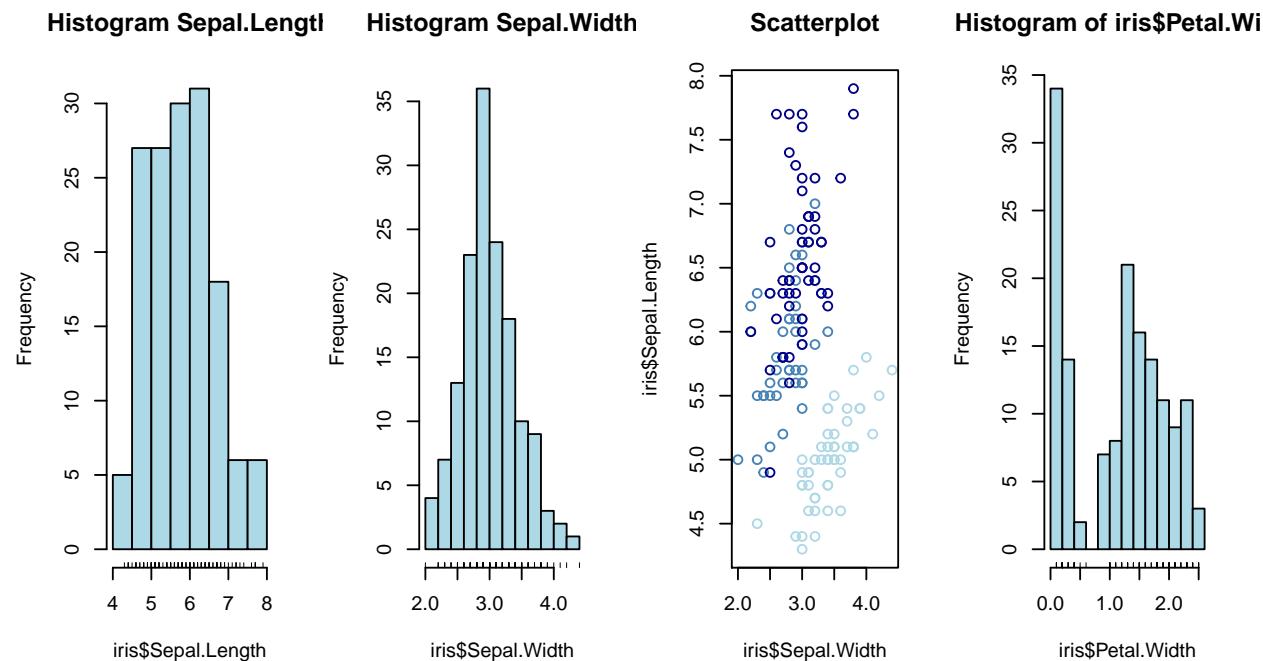
rug(iris$Sepal.Width)

#treći graf
plot(iris$Sepal.Width,iris$Sepal.Length, col=iris$Species, main="Scatterplot")

#četvrti dijagram
```

Slika 32: Podjela područja za crtanje; dva retka, dva stupca (`mfrow=c(2, 2)`)

```
hist(iris$Petal.Width, col="lightblue")
rug(iris$Petal.Width)
```



Slika 33: Podjela područja za crtanje; jedan redak s 4 grafa (mfrow=c(1, 4))

Molimo pogledajte grafove iznad i odgovorite na sljedeće:

- 1) Opišite razliku između dva prethodna grafa.
- 2) Kako su nastali naslovi grafova na slikama?
- 3) Sagledavši grafički prikaz i *iris* skup podataka, čemu služi funkcija **rug** ?

Postoje tri načina manevriranja predlošćima u osnovnoj grafici:

- 1) **par(mfrow)** - najjednostavnija metoda u osnovnoj grafici, koristi se za jednostavne grid predloške gdje je svaki panel iste veličine
- 2) **layout()** - funkcija omogućava kombiniranje panela
- 3) **split.screen()** - omogućava da se preciziraju koordinate na panelima koje više ne moraju biti jednostavni omjeri panela.

U većini primjera u ovom tečaju koristili smo najjednostavnije rješenje - **par(mfrow)** postavku. Ovdje predstavljamo korištenje **layout()** (**layout.show()**) funkcija. Ove funkcije uzimaju matricu i uvrštavaju je u predložak. Oblik matrice odgovara pojedinačnim dijelovima. Brojevi u matrici odgovaraju poretku grafa, a dijelovi s istim brojem predstavljaju jedan panel.

Kako je objašnjeno, prvo trebamo napraviti matricu koja će se konvertirati u predložak:

```
#rbind kombinira dva vektora kao uzastopne redove u matricu
mat_lay <- rbind(c(1, 1), c(2, 3))

#zatražite klasu objekta
class(mat_lay)

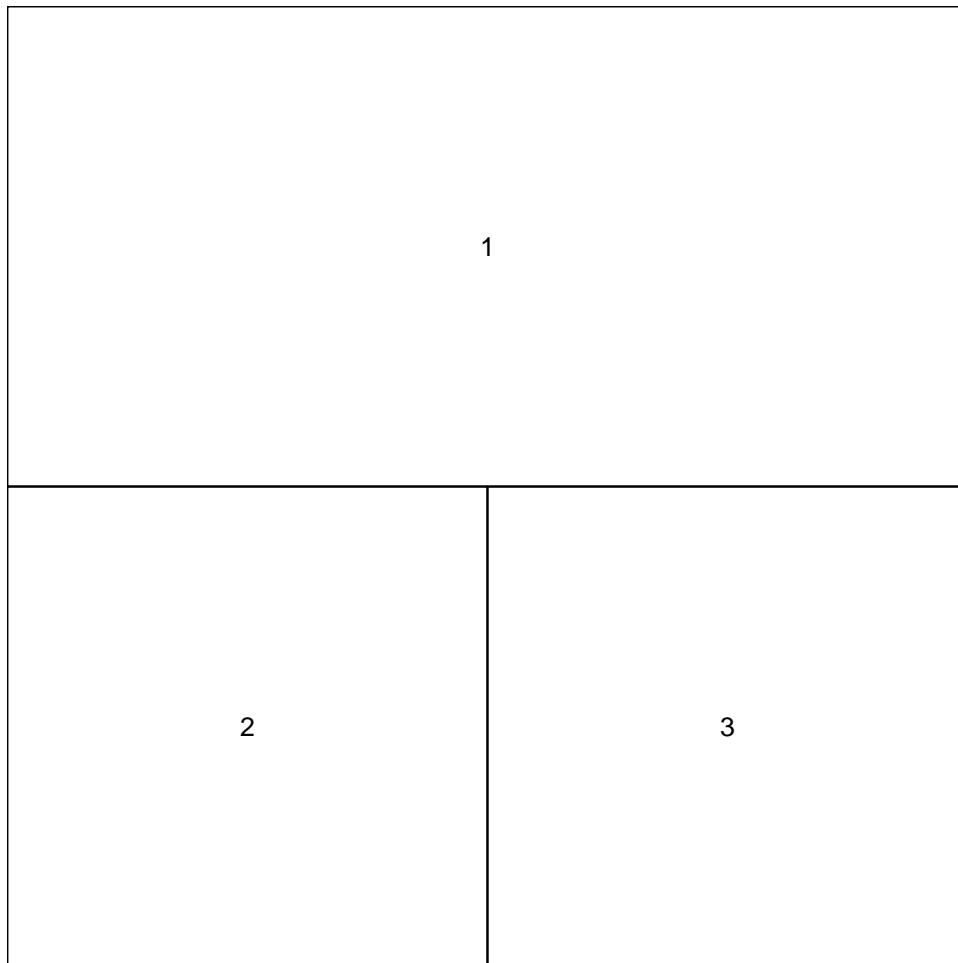
## [1] "matrix"
```

```
#ispis
mat_lay

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

#izradivanje predloška
layout(mat_lay)

#vizualiziranje predloška
layout.show(3)
```



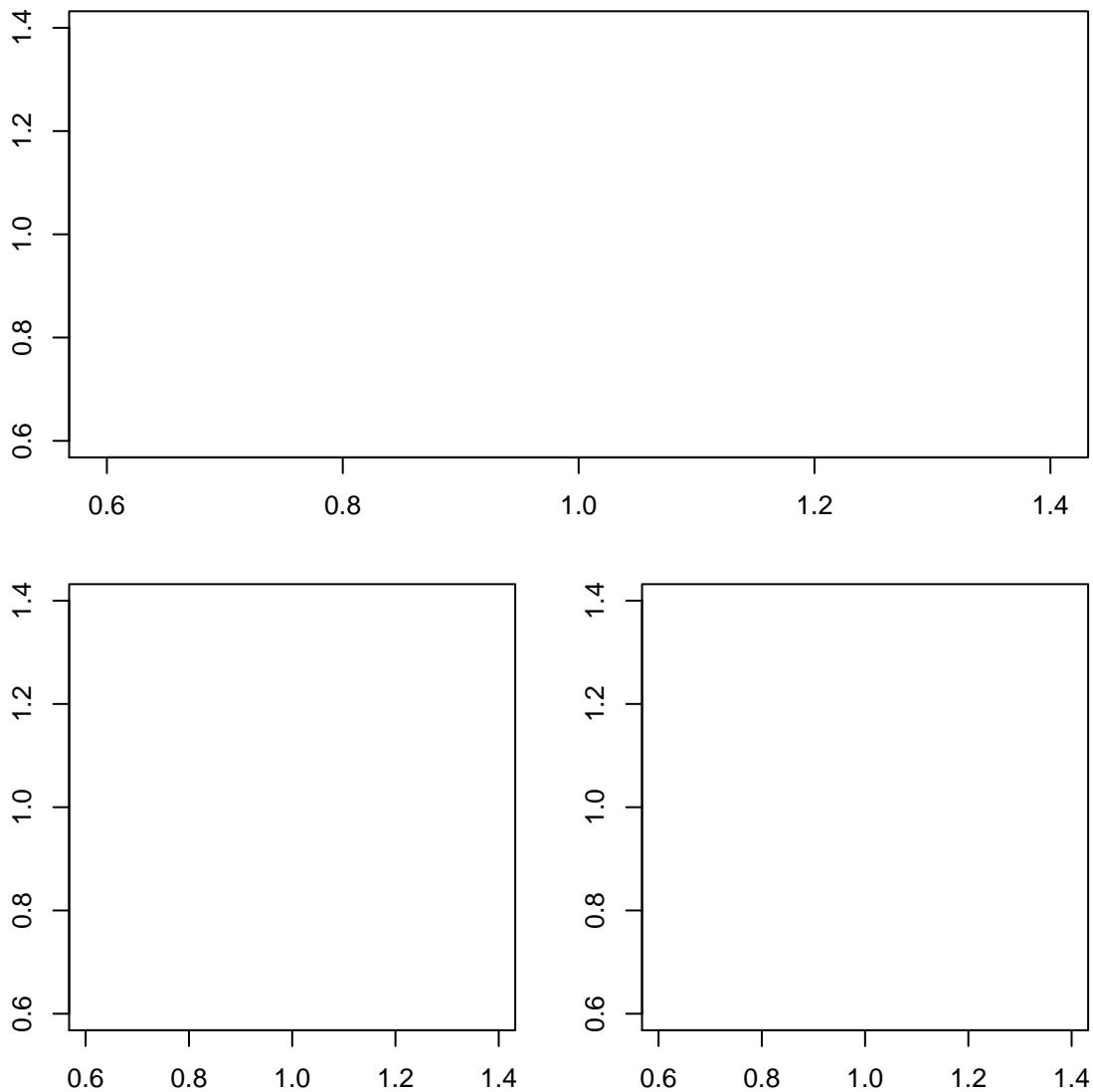
Slika 34: Podjela područja za crtanje; **show.layout()** funkcija

Sada kad imamo layout, u njega možemo crtati opcionalne grafičke prikaze.

```
#stvaranje *layouta*
layout(mat_lay)
par(mar=c(2,2,2,2))

for (i in 1:3) plot(1, 1, type = "n")
```

Posljednje rješenje **split.screen()** je izvan opsega ovoga tečaja i dalje se neće razmatrati.

Slika 35: Podjela područja za crtanje; **layout()** funkcija

#### 4.1.1.1.7 Grafički parametri koji kontroliraju margine grafa (*mar/mai*) i (*oma/omi*)

Moguće je postaviti margine za svaki panel parametrom *mar/mai* te vanjske margine parametrom/ima *oma/omi*.

Parametar *mar()*, numerički vektor oblika c(bottom, left, top, right) daje broj linija margine kako bi se precizirale četiri strane grafa. Zadano je c(5, 4, 4, 2) + 0.1. Opcionalno, margine se mogu postaviti uporabom parametra *mai*; numeričkog vektora oblika c(bottom, left, top, right) koji daje veličinu margina preciziranu u inčima. Drugi način je preciziranjem margina u inčima korištenjem argumenta *mai*, mjereno u centimetrima.

```
#postavljanje sveukupnih grafičkih parametara
par(mfrow=c(2,2))

#postavljanje parametara za prvi graf
par(mai=c(1,1,1,1))

#prvi graf (funkcija visoke razine)
hist(iris$Sepal.Length,
     col="lightblue",
     main="par(mai=c(1,1,1,1))")

#postavljanje grafičkih parametara za drugi graf
par(mai=c(0,0,0,0))

#drugi graf
hist(iris$Sepal.Width,
     col="lightblue",
     main="par(mai=c(0,0,0,0))")

#postavljanje grafičkih parametara za treći graf
par(mai=c(0.5,0.5,0.5,0.5))

#treći graf
hist(iris$Sepal.Width,
     col="lightblue",
     main="par(mai=c(0.5,0.5,0.5,0.5))")

#postavljanje grafičkih parametara za četvrti graf
par(mai=c(0.2,0.2,0.2,0.2))

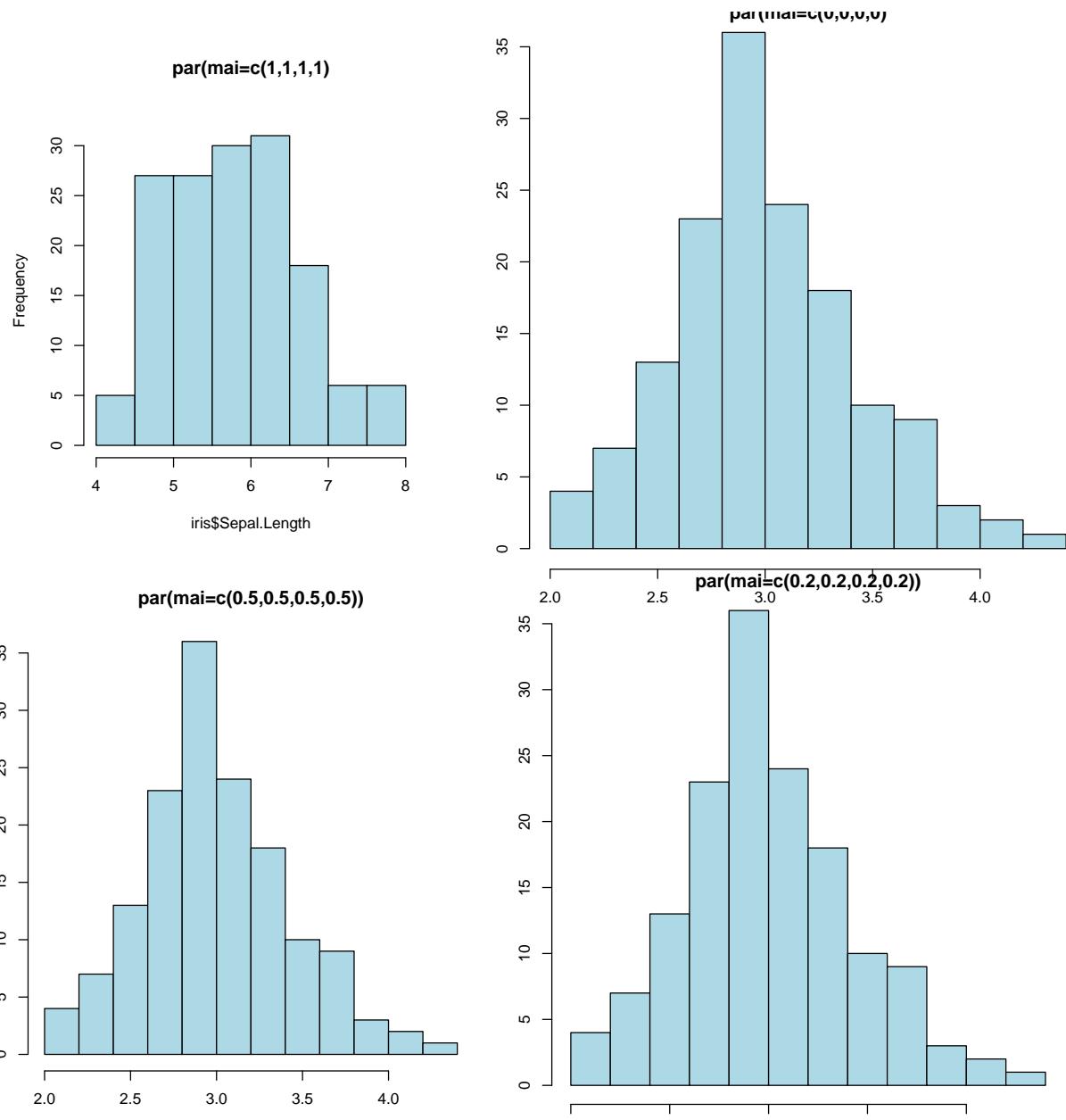
#četvrti graf
hist(iris$Sepal.Width,
     col="lightblue",
     main="par(mai=c(0.2,0.2,0.2,0.2))")
```

#### ZA SAMOSTALNI RAD

Napravite sljedeće: u ovoj kratkoj vježbi, koristite skup podataka *OrchardSprays* iz sustava R. Prvo se upoznajte sa skupom podataka tako što ćete pogledati strukturu skupa podataka i prvih nekoliko redova podataka:

```
#pozivamo podatke iz sustava
data(OrchardSprays)

#pogledamo nekoliko prvih redaka; default, 6 redaka
head(OrchardSprays)
```

Slika 36: Grafički parametar, određivanje margina u inch; `mai()` parametar

```

decrease rowpos colpos treatment
1      57      1      1      D
2      95      2      1      E
3       8      3      1      B
4      69      4      1      H
5      92      5      1      G
6      90      6      1      F

```

#pogledajmo strukturu podataka  
`str(OrchardSprays)`

```

'data.frame':   64 obs. of  4 variables:
 $ decrease : num  57 95 8 69 92 90 15 2 84 6 ...
 $ rowpos   : num  1 2 3 4 5 6 7 8 1 2 ...
 $ colpos   : num  1 1 1 1 1 1 1 1 2 2 ...
 $ treatment: Factor w/ 8 levels "A","B","C","D",...: 4 5 2 8 7 6 3 1 3 2 ...

```

Kreirajte png sliku (moja\_prva\_graf.png) 20 cm široku, 20 cm visoku, 300 dpi (engl. *dots per inch*) rezolucije na tvrdom disku vašega računala, tako što ćete slijediti ove upute, naravno, prvo se upoznavši sa sintaksom funkcije `png()`:

- 1) podijelite prostor za crtanje na tri segmenta, sva tri neka budu u jednom stupcu
- 2) U prvom segmentu dijagram koji crtate treba biti tipa histogram relativnih frekvencija varijable `decrease` sa zadanim brojem podjela, stupaca u dijagramu (engl. *bin*); histogram treba obojiti nekom bojom i treba se dati odgovarajući naslov
- 3) U drugom segmentu dijagram treba biti, također, histogram s relativnim frekvencijama varijable "decrease" sa 50 stupaca u dijagramu; histogram treba obojiti bilo kojom bojom i treba dati odgovarajući naslov; dodajte podatke na ovaj histogram (pomoć: koristite funkciju `rug()`)
- 4) U trećem dijelu ovog prikaza treba biti grafički prikaz adekvatan za kategoričke podatke (pomoć: bar dijagram) s relativnim frekvencijama varijable `treatment`; stavite odgovarajući naslov
- 5) Otvorite pripremljenu sliku.

#### 4.1.1.8 Grafički parametri koji kontroliraju automatsko obilježavanje osi (`ann`)

Ponekad je korisno imati mogućnost suzbijanja automatskoga označavanja osi kako bi se napravilo vlastito, u skladu s određenim potrebama. Ovo je moguće definiranjem parametra `ann`, stavljanjem logičke oznake treba li se napraviti automatsko obilježavanje (`ann=T`, default) ili ne (`ann=F`). Zadano obilježavanje stavlja nazive vektora (varijabli) koje se crtaju na graf. Pogledajte primjer ispod i komentirajte rezultat kôda.

```

#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#priprema podataka
x<- 1:10
y<-11:20

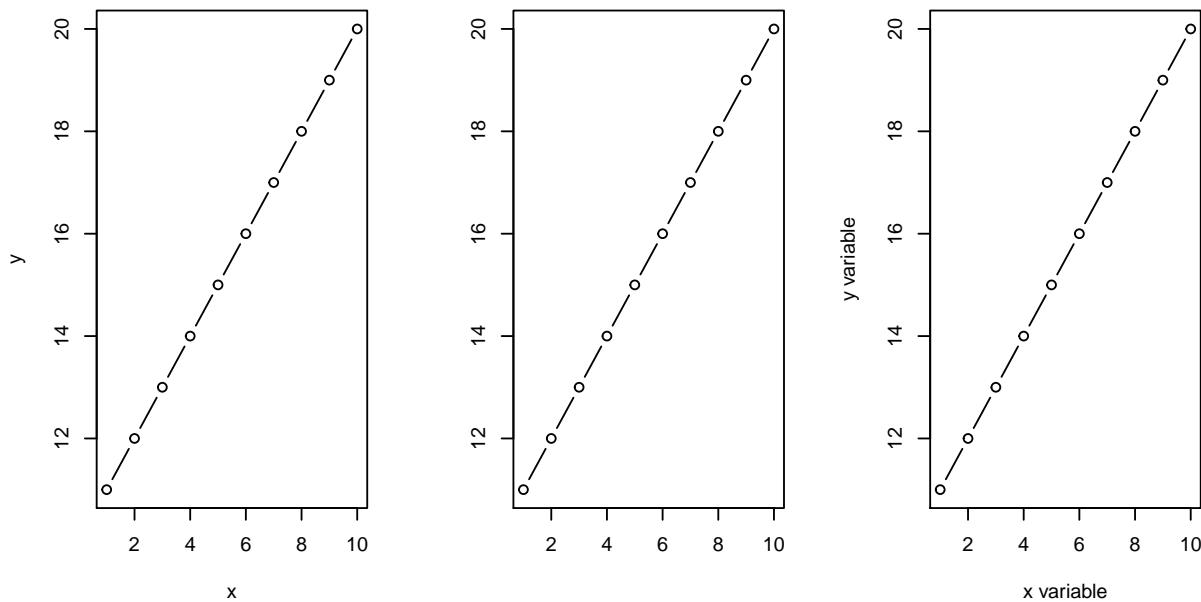
#prvi graf
plot(x,y,
      type="b",
      axes=TRUE,
      ann=TRUE)

#drugi graf
plot(x,y,
      type="b",
      axes=TRUE,
      ann=FALSE)

```

```
type="b",
axes=T,
ann=F)

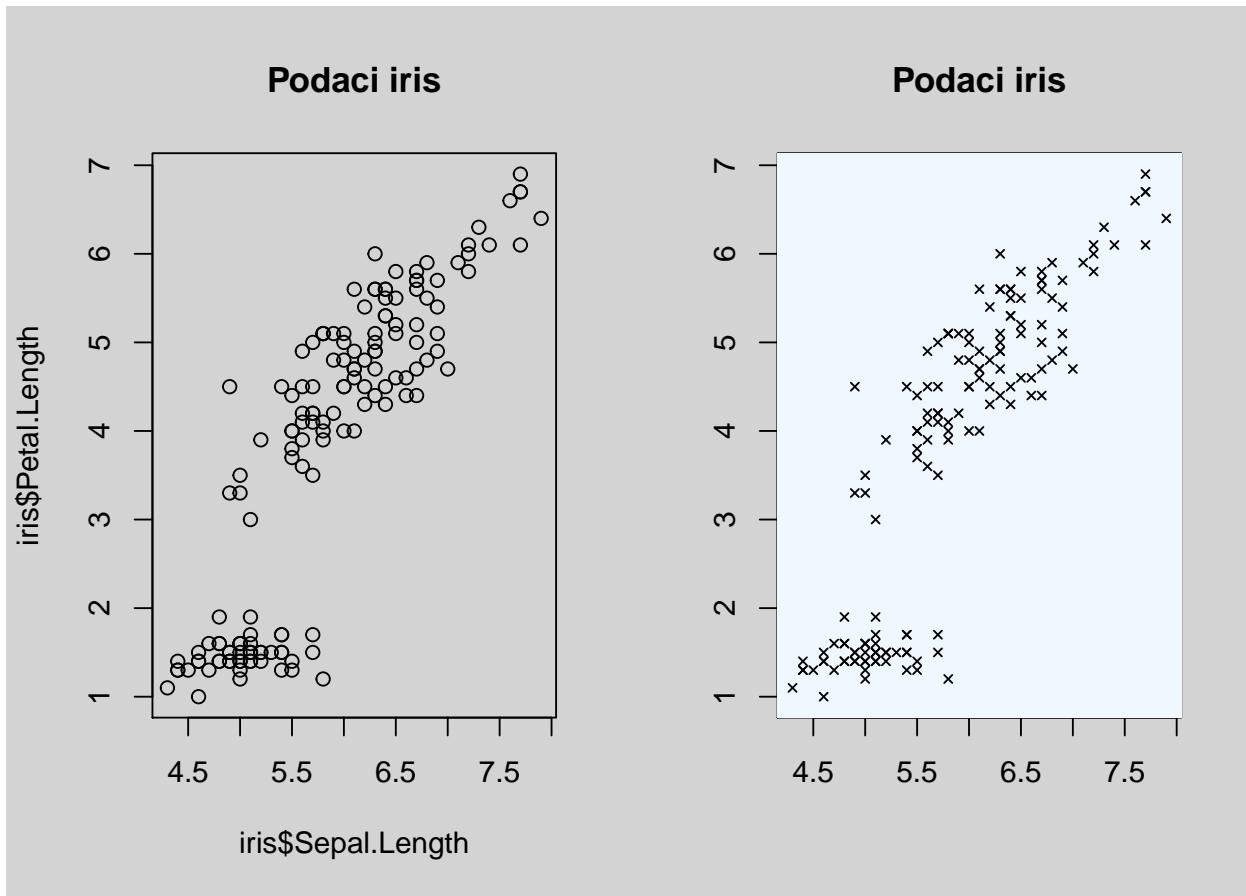
#treći graf
plot(x,y,
type="b",
xlab="x variable",
ylab="y variable")
```



Slika 37: Grafički parametar - automatska anotacija grafa; **ann** parametar

#### 4.1.1.1.9 Grafički parametri koji kontroliraju pozadinu (podlogu) područja na kojoj se crta (*usr/bg*)

Za promjenu boje podloge na grafu dovoljno je postaviti grafički parametar **bg**. Ali, ako je potrebno promijeniti bilo koji od grafičkih parametara koji su *read-only*, potreban je zaobilazni način koji je prikazan u primjeru koji slijedi.



Slika 38: Grafički parametar - pozadina grafa **usr/bg** parametar

#### 4.1.1.10 Grafički parametri koji kontroliraju promjenu udaljenosti između osi te naslova i labela osi (mgp)

Ovo je koristan parametar za kontroliranje udaljenosti između osi i njihovih naziva i labela. Da se upoznamo s parametrom i njegovim zadanim vrijednostima pogledajmo informacijsku karticu za funkciju **par()**. Linija margina (u mex jedinicama; mex je faktor raširenosti veličine karaktera koji se koristi kako bi se opisale koordinate na marginama grafa) za nazive osi, labele osi i linije osi. Uočite da `mgp[1]` utječe na naslov dok `mgp[2:3]` utječe na os. Zadane vrijednosti su vektor `c(3, 1, 0)`.

Primijetite da ćemo u sljedećem primjeru (treći primjer: `mgp = c(3, -2, 0)`), manipuliranjem parametrom labele osi ispisati unutar područja za prikaz podataka. Prvo pripremamo podatke koje ćemo kasnije koristiti u crtaju. Ovakave, jednostavne podatke stvaramo radi boljeg primjećivanja nastalih promjena u dimenzijama grafa:

```
#kreirajte podatke za crtanje
x <- 1:10
y <- 11:20
z <- paste("p", 1:10, sep="_")
p <- rep(1:2, 5)
df<- data.frame(x,y,z,p)

#kreirajte podatke za crtanje
x <- 1:10
y <- 11:20
z <- paste("p", 1:10, sep="_")
p <- rep(1:2, 5)
df<- data.frame(x,y,z,p)

#postavljanje grafičkih parametara
par(mfrow=c(1,3), mgp=c(3,1,0))

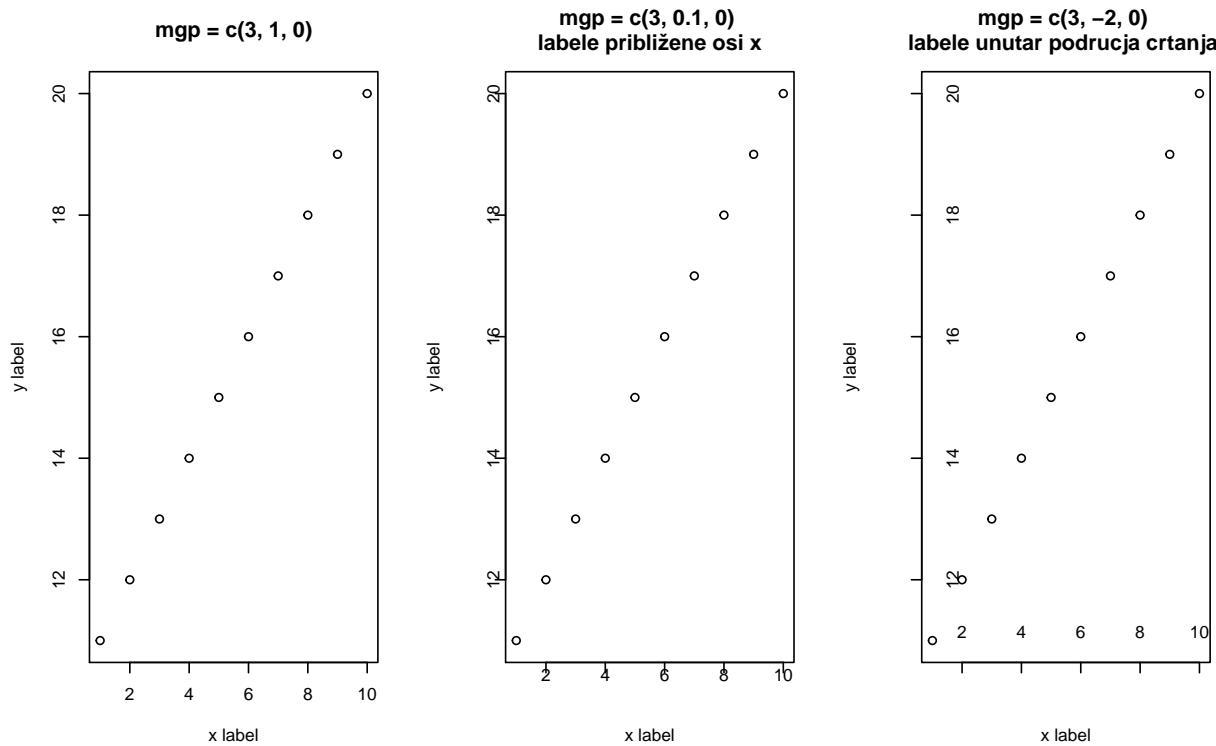
#prvi graf
plot(df$x, df$y, type="p",
      xlab="x label",
      ylab="y label",
      main="mgp = c(3, 1, 0)")

#promjena grafičkih parametara
par(mgp=c(3,.1,0))

#drugi graf
plot(df$x, df$y, type="p",
      xlab="x label",
      ylab="y label",
      main="mgp = c(3, 0.1, 0) \nlabele približene osi x")

#promjena grafičkih parametara
par(mgp=c(3,-2,0))

#treći graf
plot(df$x, df$y, type="p",
      xlab="x label",
      ylab="y label",
      main="mgp = c(3, -2, 0) \nlabele unutar područja crtanja ")
```



Slika 39: Promjena grafičkoga parametra `mgp`; udaljenost između osi i teksta

#### 4.1.2 Neke grafičke funkcije visoke razine (engl. high-level)

Kao što je već navedeno, "high-level" grafičke funkcije su one koje inicijaliziraju ili produciraju cijelokupan graf. Kasnije je moguće, korištenjem funkcija niske razine, "low-level" funkcijama poboljšati napravljeni graf prema našim potrebama i željama. U nastavku tečaja cijelo ćemo vrijeme koristiti mješavinu *high-level* i *low-level* funkcija kako bismo došli do željenog rezultata, a time i naučili kako cijeli proces *base* grafičke funkcije funkcioniра. Kada se pojedina *low-level* funkcija poziva unutar neke *high-level* funkcije tada ju nazivamo parametrom *high-level* funkcije.

##### 4.1.2.1 Funkcija `plot(){graphics}{high-level}`

Generička funkcija za crtanje objekata u R-u. Više detalja o argumentima grafičkog parametra može se naći pomoću funkcije `par()`.

U primjeru ćemo napraviti jednostavan graf, a tekst ćemo dodati nakon kreiranja grafa pomoću funkcije niske razine.

Kôd koji slijedi daje jednostavan primjer kako napraviti graf pomoću R-a. Za tu svrhu će se koristiti skup podataka `prices`. Skup podataka `prices` sadrži dvije numeričke varijable `price` i `size` zamišljenih kuća.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))
```

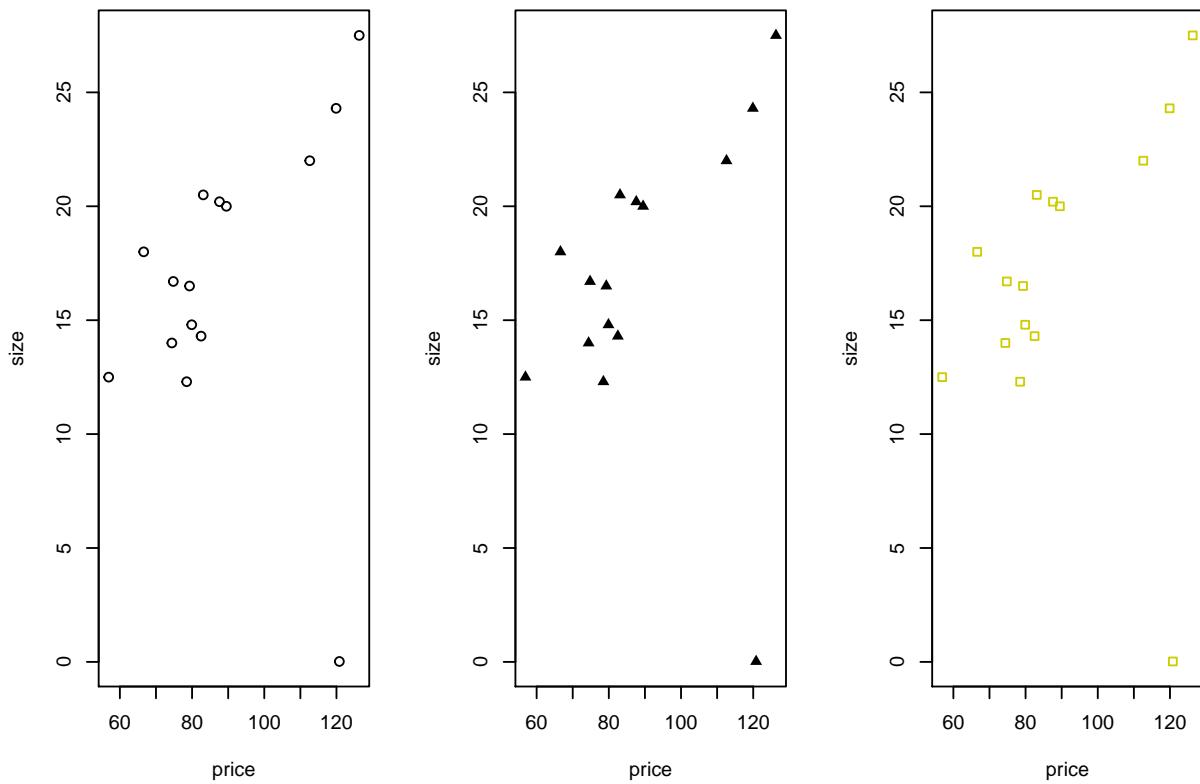
```
#struktura podataka
str(prices)
```

```
## 'data.frame':    15 obs. of  2 variables:
##   $ price: num  89.5 79.9 83.1 56.9 66.6 ...
```

```
## $ size : num 20 14.8 20.5 12.5 18 14.3 27.5 16.5 24.3 20.2 ...
#prvi graf
plot(prices)

#drugi graf
plot(prices,
      pch=17)

#treći graf
plot(prices,
      pch=22,
      col="yellow3")
```



Slika 40: Jednostavan graf; a) s predodređenim vrijednostima; b) **pch** parametar i c) boja mijenjana; skup podataka iz sustava R - prices

#### 4.1.3 Poboljšanje rezultata dobivenoga funkcijama visoke razine funkcijama niske razine

Mnoge funkcije crtanja visoke razine (plot, hist, boxplot,itd.) omogućavaju da uključite osi i opcije teksta (kao i druge grafičke parametre). Možemo mijenjati graf, pozvati dodatne funkcije niske razine, ili promjeniti grafičke parametre izravno u funkcijama visoke razine, sukladno našim potrebama/preferencijama. U primjeru:

- a) popunite prazni graf osima - zadane vrijednosti (engl. default)
- b) popunite taj prazni graf osima - precizirano od strane korisnika
- c) dodajte različite elemente grafa npr. naslov, podnaslov.

Postoji mnogo mogućnosti kako unaprijediti izgled grafa. U daljem tekstu uvodimo samo najznačajnije funkcije niske razine koje dodaju određene elemente na postojeći graf:

- proizvoljne linije / krivulje pomoću **lines()** / **curves()**
- poligoni s **polygon()**
- ravne linije s **ablines()**
- promjena izgleda osi / osi definirane od strane korisnika **axes()**
- legenda pomoću **legend()**
- tekst pomoću **text()**
- tekst na preciziranoj margini pomoću **mtext()**
- nacrtati okvir oko grafa **box()** {graphics}
- naslovi pomoću **title()** {graphics} funkcije
- i još mnogo toga!

#### 4.1.3.1 Funkcija `lines()`/`curves()` {graphics} (low-level)

Funkcija `lines()` dodaje povezane segmente linija na graf, a funkcija `curve()` crta krivulju koja odgovara funkciji tijekom nekog zadanoog intervala.

Ponekad želimo staviti određene vrste linija (krivulje) pored pravih linija na postojeći graf. U ovom primjeru stavit ćemo liniju koja je kreirana funkcijom `lowess()` {stats} koja radi izračun za *lowess smoother* koji koristi lokalno ponderiranu polinomsku regresiju na podacima.

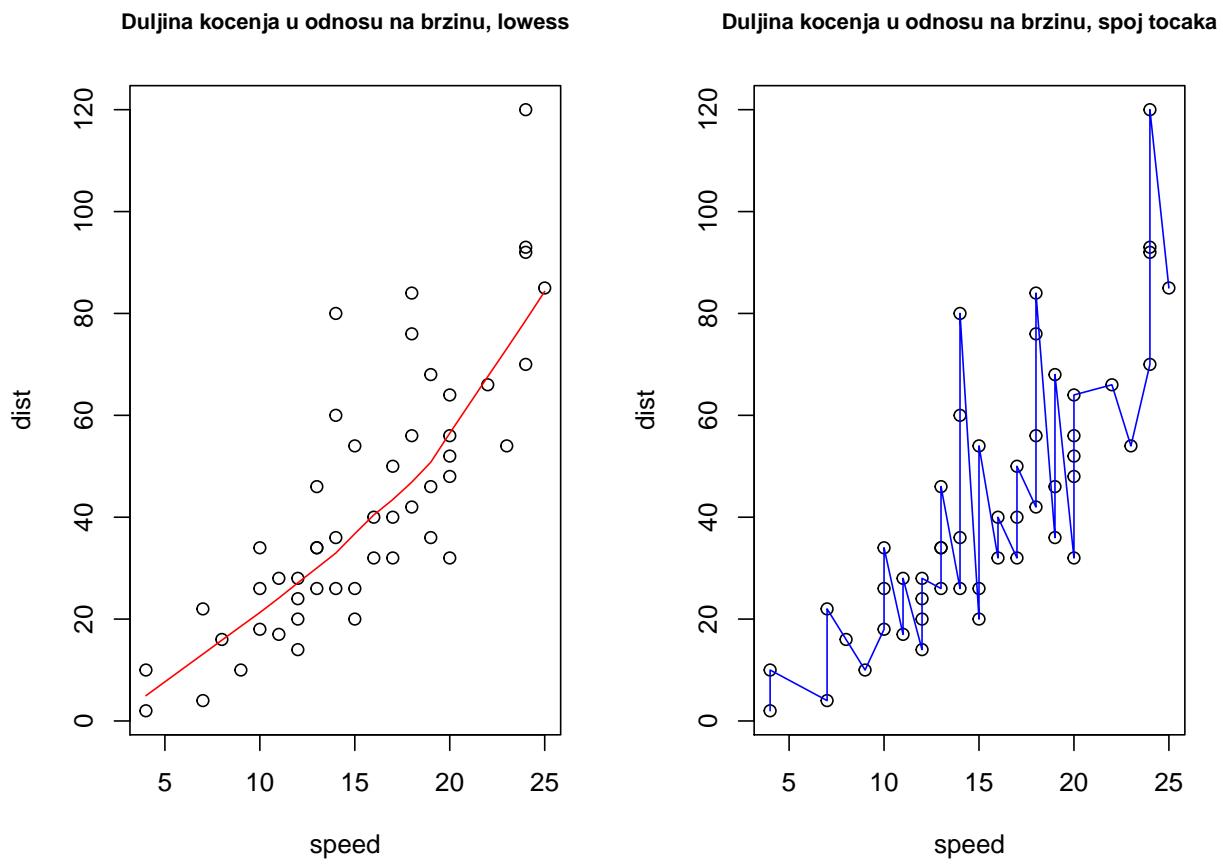
```
#postavljanje grafičkih parametara
par(mfrow=c(1,2))

#prvi graf
plot(cars,
      main = "Duljina kočenja u odnosu na brzinu, lowess",
      cex.main=0.8)

#eksplicitno traženje funkcije drugog paketa za izračun koordinata linije
lines(stats::lowess(cars), col="red")

#drugi graf
plot(cars,
      main = "Duljina kočenja u odnosu na brzinu, spoj točaka",
      cex.main=0.8)

# linije kroz točke
lines(cars, col="blue")
```



Slika 41: Dodavanje linija na postojeći graf a) funkcija izračuna `lowess` iz `stats` paketa (Kernel), b) jednostavno povezivanje točaka linijom

#### 4.1.3.2 Funkcija `rect()`/`polygon()`/`graphics`(*low-level*)

Funkcija `rect()` crta pravokutnik (ili sekvencu pravokutnika) sa zadanim koordinatama.

Funkcija `polygon` crta poligone čiji su čvorovi prikazani pomoću x i y. Vrlo korisna funkcija za sjenčanje područja ispod krivulja, značajno u vizualiziranju tijekom procesa testiranja hipoteza, ali i u mnogim drugim primjenama. Funkcija uzima x i y vektor, definirajući skup koordinata koji se uzima kako bi se moglo pratiti područje koje će se sjenčati.

Prije primjene funkcije(a), molimo pogledajte informacijske kartice funkcije:

```
#pripremite koordinate poligona
x <- c(0, 1, 2, 3, 5, 4, 3, 2, 1, 0)
y <- c(4, 2, 2, 1, 3, 0, 0, 1, 1, 0)

#pripremite koordinate tri pravokutnika
z<- c(0.5, 1, 1.5, 2)
p<- c(2, 2.5, 3, 3.5)
q<- c(3.5, 4, 4.5, 5)
g<- c(2.5, 3, 3.3, 4)

#postavljamo grafičke parametre
par(mfrow=c(1,2))

#prvi graf
plot(x,y, type="n", pch=22, main="Prvo poligon, pravokutnici kasnije", ylim=c(-1, 5))

#crtamo poligon sa skupom koordinata
polygon(x,y,col="lightblue")

#crtamo pravokutnike na postojećem grafu
rect(z, p, q, g,
      col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3"))

#imanovanje točaka
text (x,y ,labels = paste("t", 1:10, sep=""), pos=1)

#drugi graf
plot(x,y, type="n", pch=22, main="Prvo pravokutnici, poligon kasnije", ylim=c(-1, 5))

#crtamo pravokutnike na postojećem grafu
rect(z, p, q, g,
      col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3"))

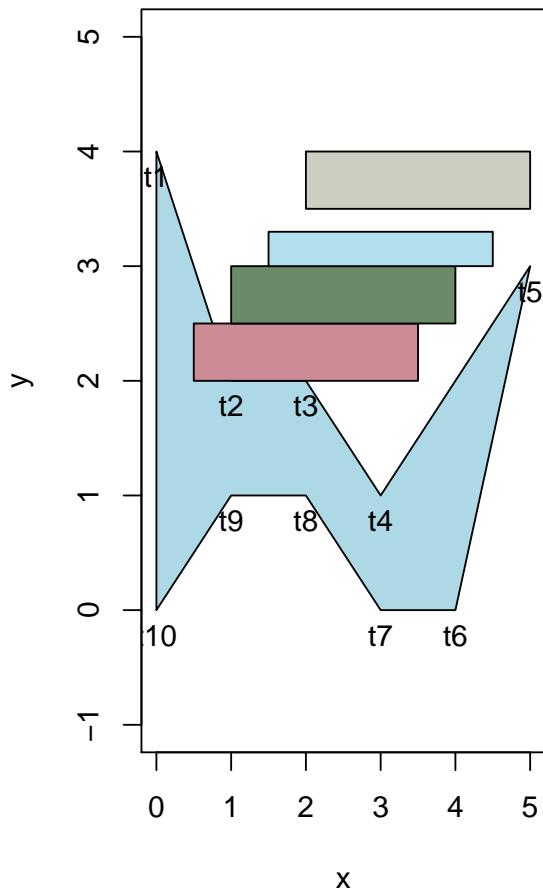
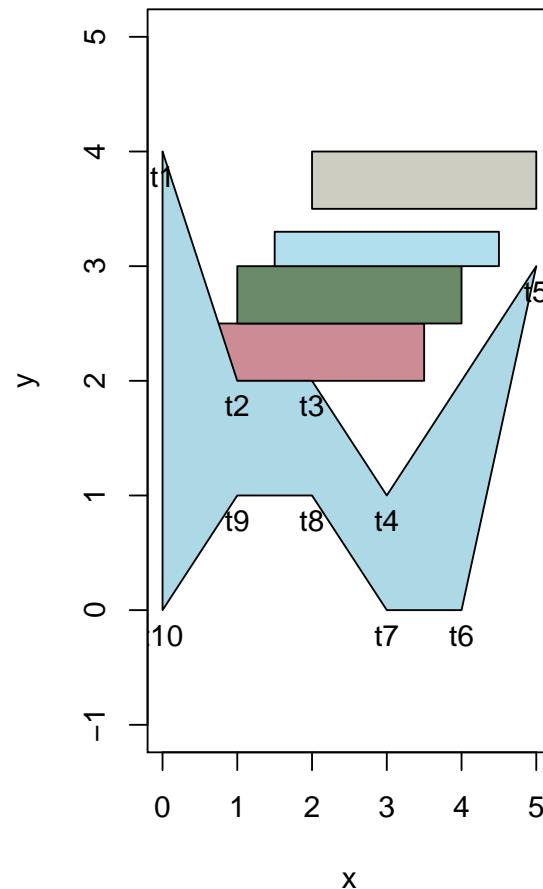
#dodajemo poligon na zadalu koordinatu
polygon(x,y,col="lightblue")

#stavljam labele
text (x,y ,labels = paste("t", 1:10, sep=""), pos=1)
```

U primjeru koji slijedi funkciju `polygon()` koristimo kako bismo naznačili regiju odbacivanja hipoteze:

```
#pretpostavimo da poznajemo pravi populacijske parametare - srednja vrijednost
populacijska_sredina <- 400
populacijska_sd <- 50

#statistika izračunata na reprezentativnom uzorku
srednja_vrijednost_uzorka <- 418
```

**Prvo poligon, pravokutnici kasnije****Prvo pravokutnici, poligon kasnije**Slika 42: Funkcije **polygon()**/**rect()**

```

#parametri procesa testiranja hipoteze o populacijskom parametru
alpha <- 0.05;

#veličina uzorka
n <- 50

#raspon za z
xmin <- -4; xmax <- 4

#sekvenca za koju crtamo
x <- seq(xmin, xmax, by=0.1)

#statistika izračunata na uzorku (z statistika, uzorak veličine n)
z <- (srednja_vrijednost_uzorka - populacijska_sredina)/ (populacijska_sd/sqrt(n))

#funkcija d_norm - gustoća
gustoca_x <- dnorm(x)

#crtamo
plot(x, gustoca_x, type="l", xlab="z vrijednosti",
      ylab="Gustoća",
      main="",
      axes=T)

#računamo kritične vrijednosti za regiju odbacivanja hipoteze
krit_lijeva <- -qnorm(1-alpha/2)
krit_desna <- qnorm(1-alpha/2)
i <- x >= krit_desna

#regija odbacivanja na desnoj strani
polygon(c(krit_desna, x[i], xmax),
         c(0, gustoca_x[i], 0),
         col="lightblue")

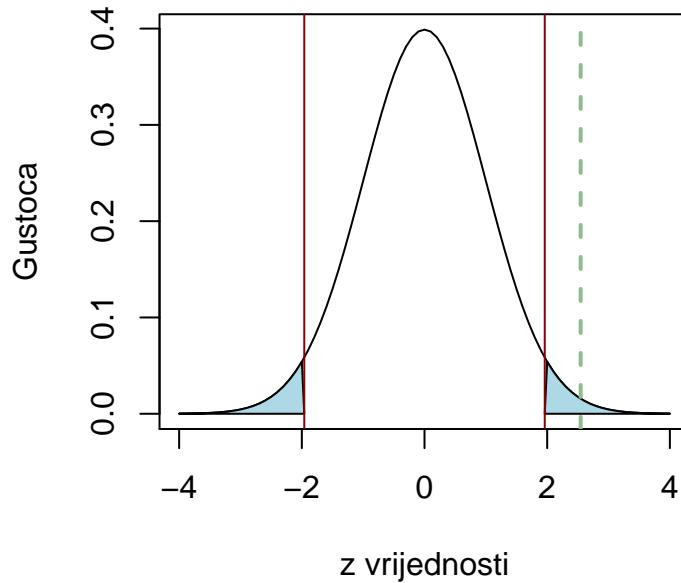
#regija odbacivanja na lijevoj strani
j <- x <= krit_lijeva
polygon(c(xmin, x[j], krit_lijeva),
         c(0, gustoca_x[j], 0),
         col="lightblue")

#crtamo z na uzorku
abline(v=z, lty=2, col="darkseagreen", lwd=2)

#dodatno označimo kritične vrijednosti

abline(v=krit_desna, lty=1, col="darkred", lwd=1)
abline(v=krit_lijeva, lty=1, col="darkred", lwd=1)

```



Slika 43: Funkcija **polygon()** u procesu testiranje hipoteze; dvosmjerni z test

#### 4.1.3.3 Funkcija **legend() {graphics} (low-level)**

Legendu postavljamo na zadatu koordinatu na grafu koristeći se funkcijom **legend()**. Kako biste otvorili informacijsku karticu funkcije i naučili više o njoj, unesite **?legend** u R konzolu.

```
#stvaramo podatke
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#funkcija visoke razine koje stvara (inicira) graf, tj. koordinatni sustav
plot(c(0,3),
      c(0,3),
      type="n", xlab="x", ylab="y")

#dodavanje linija na postojeći graf
lines(x, y1, col="red4", lwd=2)
lines(x, y2, col="darkblue", lwd=2)

#dodavanje legende na postojeći graf
legend("topright",
       inset=.05,
       cex = 1,
```

```

title="Legenda",
c("kosinus","sinus"),
horiz=TRUE,
lty=c(1,1),
lwd=c(2,2),
col=c("red4","darkblue"),
bg="lightgray")

#drugi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linija
lines(x, y1, col="red4", lwd=2)
lines(x, y2, col="darkblue", lwd=2)

#dodavanje legende
legend("topleft",
       inset=0.4,
       cex = 1,
       title="Legenda",
       c("Kosinus","Sinus"),
       horiz=TRUE,
       lty=c(1,1),
       lwd=c(2,2),
       col=c("red4","darkblue"),
       bg="white")

#treći graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linija
lines(x, y1, col="red4", lwd=2)
lines(x, y2, col="darkblue", lwd=2)

#dodavanje legende
legend("topright",
       inset=0,
       cex = 0.8,
       title="Drugačija legenda",
       c("Kosinus","Sinus"),
       horiz=TRUE,
       lty=c(1,1),
       lwd=c(2,2),
       col=c("red4","darkblue"),
       bg="white")

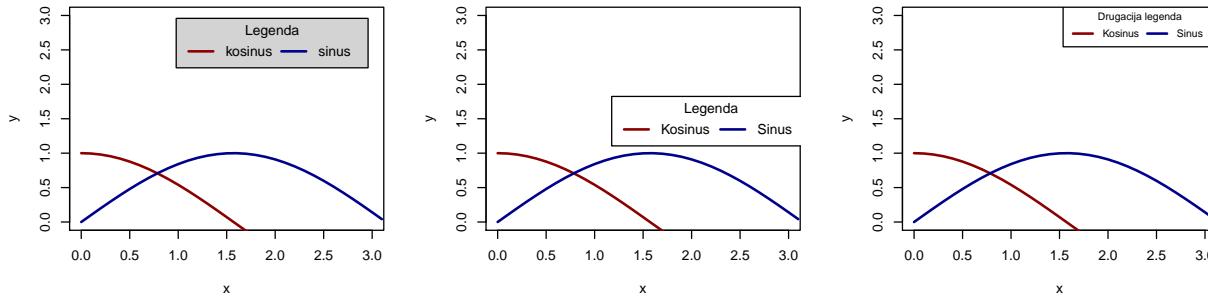
```

Za interaktivnu specifikaciju položaja legende, koristite funkciju **locator()**. Kako biste naučili o opcijama funkcije, unesite **?locator** u R konzolu.

## ZA SAMOSTALNI RAD

Korištenjem skupa podataka unutar sustava R **cars** grafički prikažite podatke na način da:

- 1) Podijelite grafički uređaj na jedan red s dva stupca;
- 2) **Prvi** graf treba izgledati ovako:



Slika 44: Dodavanje elemenata grafa - legenda

- inicirajte crtanje uporabom generičke funkcije **plot()**, potisnite automatske anotacije
- definirajte proizvoljno x i y osi, neka x os bude označena s "Brzina automobila"; neka y os bude označena "Udaljenost"
- dodajte točke dijagramu raspršena
- stavite legendu na graf u gornji lijevi kut (engl. *topleft*)
- stavite naslov ("Prva slika - skup podataka mtcars"), cex za naslov neka bude 0.;
- stavite naslov ("Moj proizvoljni naslov") u plavo boji; 90% **default** cex parametra.

3) **Drugi** graf treba izgledati ovako:

- nacrtajte podatke sa zadanim osima
- neka x os bude označena sa "Brzina"; neka y os bude označena sa "Udaljenost" kao 90% zadane veličine
- stavite legendu na graf na poziciju u gornjem desnom kutu (engl. *topright*)
- stavite naslov grafa ("Prva slika - skup podataka mtcars"), cex za naslov treba biti 0.9
- stavite naslov grafa ("Moj proizvoljni naslov") u sivoj boji; 120% zadanoga cex parametra.

#### 4.1.3.4 Funkcija **axis() {graphics}** (*low-level*)

Korisnik može napraviti prilagođene osi korištenjem **axis()** funkcije. Otvorite informacijsku karticu funkcije i naučite o njoj tako što ćete unijeti **?axis** u R konzolu.

Ima mnogo mogućnosti kako unaprijediti izgled osi na dijagramu, npr. promjenom boje x osi, promjenom boje y osi; orientacijom labela (oznaka) i još mnogo toga.

Pogledajte kôd koji slijedi i komentirajte!

```
#određivanje grafičkih parametara
par(mfrow=c(1,3))
```

```
#prije graf
plot(1:10,11:20,
      type="b",
      axes=F,
      ann=T)
```

```
#dodavanje osi
```

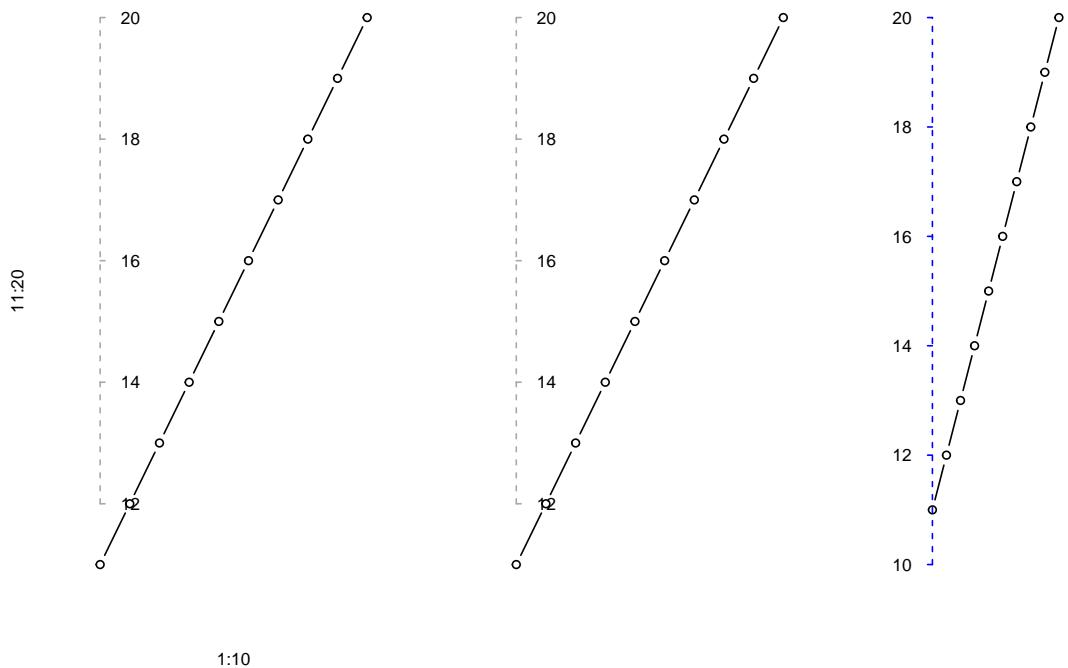
```
axis(side=4,
     pos=1,
     lty=2,
     col="darkgray",
     las=1)

#drugi graf
plot(1:10,11:20,
      type="b",
      axes=FALSE,
      ann=FALSE)

#dodavanje osi
axis(side=4,
     pos=1,
     lty=2,
     col="darkgray",
     las=1)

#treći graf
plot(1:10,11:20,
      type="b",
      axes=FALSE,
      ann=FALSE,
      xlim=c(1, 20),
      ylim = c(10,20))

#dodavanje osi
axis(side=1,
     labels=,
     pos=2,
     lty=2,
     col="red",
     las=1)
axis(side=2,
     labels=,
     pos=1,
     lty=2,
     col="blue",
     las=2)
```



1:10

Slika 45: Promjena izgleda grafičkih osi

#### 4.1.3.5 Funkcija abline() {graphics} (low-level)

Funkcija crta preciziranu liniju s mogućnošću definiranja vertikalne, horizontalne linije ili jednadžbe linije pravca korištenjem funkcije **abline()**. Informacijsku karticu moguće je otvoriti i naučiti više o funkciji unosom **?abline** u R konzolu.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#priprema podataka
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#prvi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodajte linije postojećem grafu
lines(x, y1, col="red4", lwd=2)
lines(x, y2, col="darkblue", lwd=2)

#dodavanje ravne linije ne postojeći graf
abline(a = NULL,
       b = NULL,
       h = 1.5,
       col="lightpink3",
```

```

lwd=3,
lty=2)

#drugi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodajte linije postojećem grafu
lines(x, y1, col="red4", lwd=2)
lines(x, y2, col="darkseagreen", lwd=2)

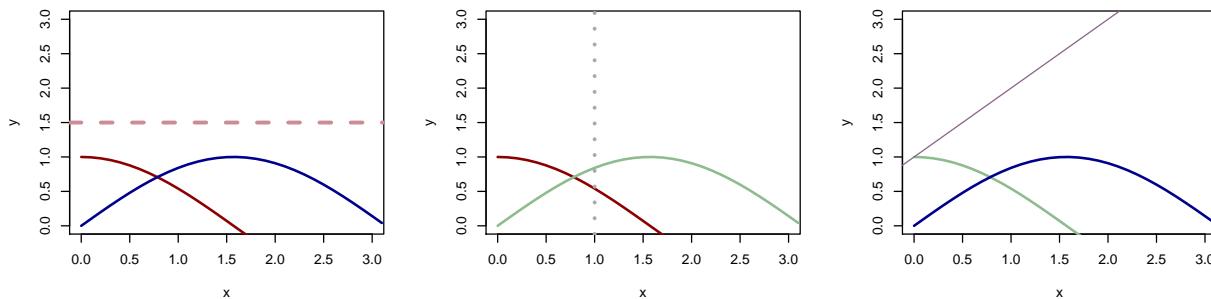
#dodavanje ravne linije postojećem grafu
abline(v = 1,
       col="darkgray",
       lty=3,
       lwd=3)

#treći grafu
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje ravne linije postojećem grafu
lines(x, y1, col="darkseagreen", lwd=2)
lines(x, y2, col="darkblue", lwd=2)

#dodavanje pravca postojećem grafu
abline(a = 1,
       b = 1,
       col="plum4",
       lty=1)

```



Slika 46: Dodavanje elemenata grafa - ravna linija

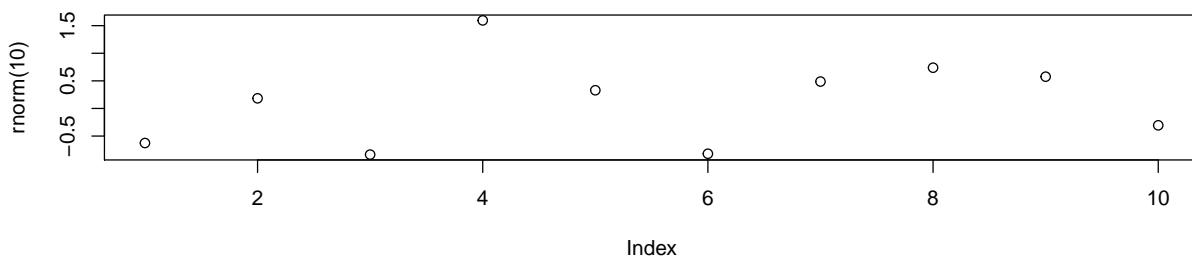
```

#postavljanje grafičkih parametara
par(mfrow=c(1,1))

#početak generatora slučajnih brojeva
set.seed(1)

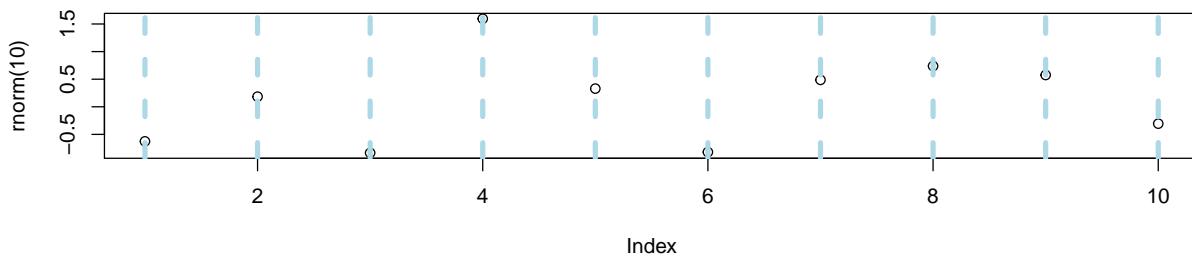
#crtanje nasumičnih točaka (10 točaka generiranih iz standardne normalne distribucije)
plot(rnorm(10)) # dijagram visoke razine

```



Slika 47: Dodavanje elemenata grafa kroz for petlju - ravna linija

```
##višestruke promjene niske razine u petlji (jedan R izraz)
for(i in 1:10) {
  abline(v = i, lty = 2, lwd=4, col="lightblue")
}
```



Slika 48: Dodavanje elemenata grafa kroz for petlju - ravna linija

#### 4.1.3.6 Funkcija `text(){graphics}` (*low-level*)

Funkcijom stavljamo proizvoljan tekst na preciziranu koordinatu u grafu. Kako biste otvorili informacijsku karticu funkcije i naučili o funkciji, unesite `?text` u R konzolu.

```
#priprema podataka
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#postavljanje grafičkih parametara
par(mfrow=c(1,3))
#prvi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linija postojecem grafu
lines(x, y1, col="red", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodati tekst na određenoj lokaciji postojecem grafu
text(1.5,2, "Funkcije sinus i kosinus")

#drugi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linija postojecem grafu
lines(x, y1, col="#00A600FF", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodati tekst na određenu lokaciju u postojecem grafu
text(1.5,2.5, "Funkcije sinus i kosinus", cex=0.6, col="orange")

#treći graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

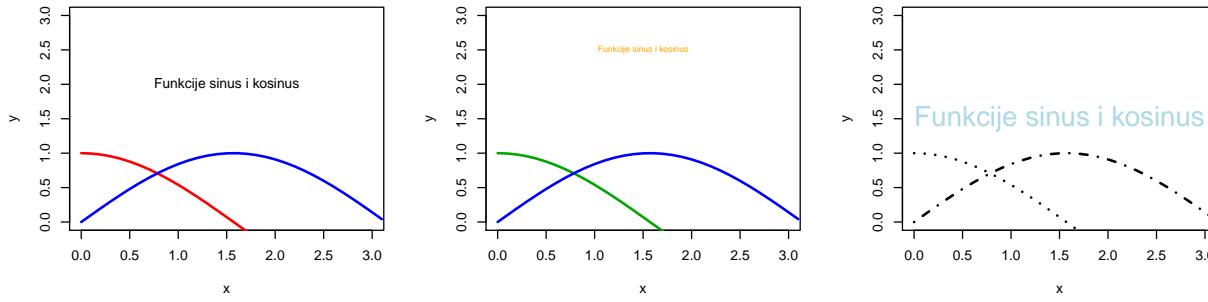
#dodavanje linija postojecem grafu
lines(x, y1, lwd=2, lty=3)
lines(x, y2, lwd=2, lty=4)

#dodati tekst na određenu lokaciju u postojecem grafu
text(1.5,1.5,
      "Funkcije sinus i kosinus",
      cex=2,
      col="lightblue")
```

U primjeru koji slijedi dodat ćemo određene oznake na svaku točku grafa. Oznake (labele) moraju biti tipa *character* ili u obliku izraza (ili se moraju moći prebaciti u takav tip).

```
#stvaramo podatke za crtanje
x <- 1:10
y <- 11:20

#radimo labele
p <- rep(1:2, 5)
```



Slika 49: Dodavanje elemenata grafa - tekst na zadatu koordinate

```

z <- paste("lab", 1:10, sep="_")

#stavljamo sve u data.frame
df<- data.frame(x,y,z,p)

#set graphical parameters
par(mfrow=c(1,3))

#prvi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - labele 1")

#dodavanje teksat na postojeći graf
text(x=x, y=y, labels=as.character(z), adj=1)

#drugi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - labele 2")

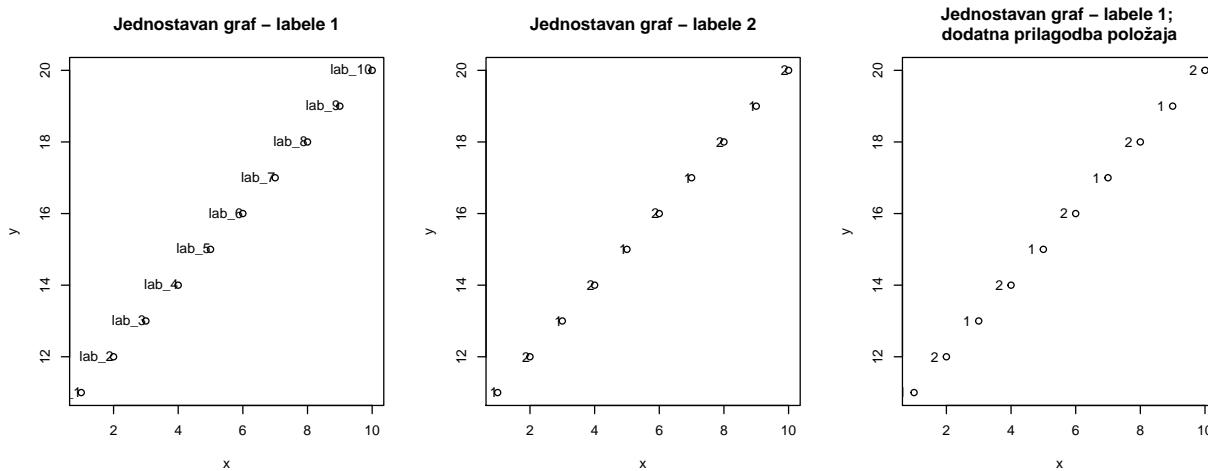
text(x=x, y=y, labels=as.character(p), adj=1)

#treći graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - labele 1; \ndodatna prilagodba položaja")
#dodavanje teksat na postojeći graf
text(x=x, y=y, labels=as.character(p), col.lab="blue", adj=2)

```

#### 4.1.3.7 Funkcija mtext() {graphics}

Ova funkcija smješta tekst na margine postojećega dijagrama; označava linije margina. Opcije funkcija se kao i za sve funkcije



Slika 50: Dodavanje elemenata grafa - oznaka točaka uz dodatnu prilagodbu položaja

mogu naći korištenjem funkcije **question**. Osi su numerirane na svakoj strani dijagrama (1=bottom, 2=left, 3=top, 4=right).

```
#postavljanje grafičkih parametara
par(mfrow=c(1,2))
```

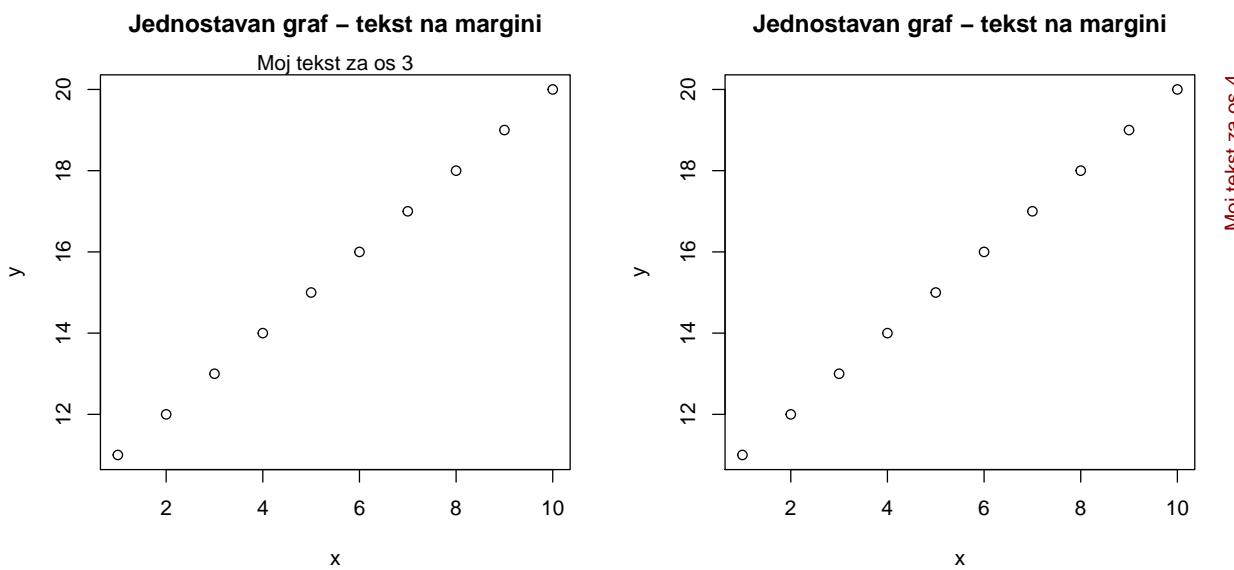
```
#prvi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf – tekst na margini")

#dodavanje teksta na osi 3
mtext("Moj tekst za os 3",
      side=3,
      line=0)

#drugi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf – tekst na margini")

#dodavanje teksta na osi 4
mtext("Moj tekst za os 4",
      side=4,
      line=1,
      adj=1,
      col="red4")
```

Preciziranje određenoga fonta specifično je za grafički uređaj. Font pripada određenoj obitelji fontova (npr. Helvetica ili Courier) i predstavlja određeno "lice" u okviru te obitelji (npr. **bold** ili *italic*). Moguće je specificirati i lice fonta koje mora biti cijeli broj, obično između 1 i 4. Specijalna vrijednost 5 indicira da se treba koristiti font simbola. Dodatne detalje moguće je pogledati na poveznici [http://www.e-reading.club/bookreader.php/137370/C486x\\_APPb.pdf](http://www.e-reading.club/bookreader.php/137370/C486x_APPb.pdf). Unutar sustava R postoje različiti paketi koji omogućavaju korisniku da uveze dodatne fontove korištenjem paketa **extrafont**.



Slika 51: Dodavanje elemenata grafa - tekst na željenoj osi

```
#stavljanje grafičkih parametara
par(mfrow=c(1,3))

#izrađivanje podataka
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#prvi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linije na postojeći graf
lines(x, y1, col="red", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodajte tekst na preciziranim koordinatama
text(x=1.5,
      y=2,
      "Funkcija sinus i kosinus",
      family="Helvetica")

#drugi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodajte linije postojećem grafu
lines(x, y1, col="#00A600FF", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodavanje teksta na postojeći graf; određene koordinate
text(1.5,2.5, "Funkcija sinusa i kosinusa",
```

```

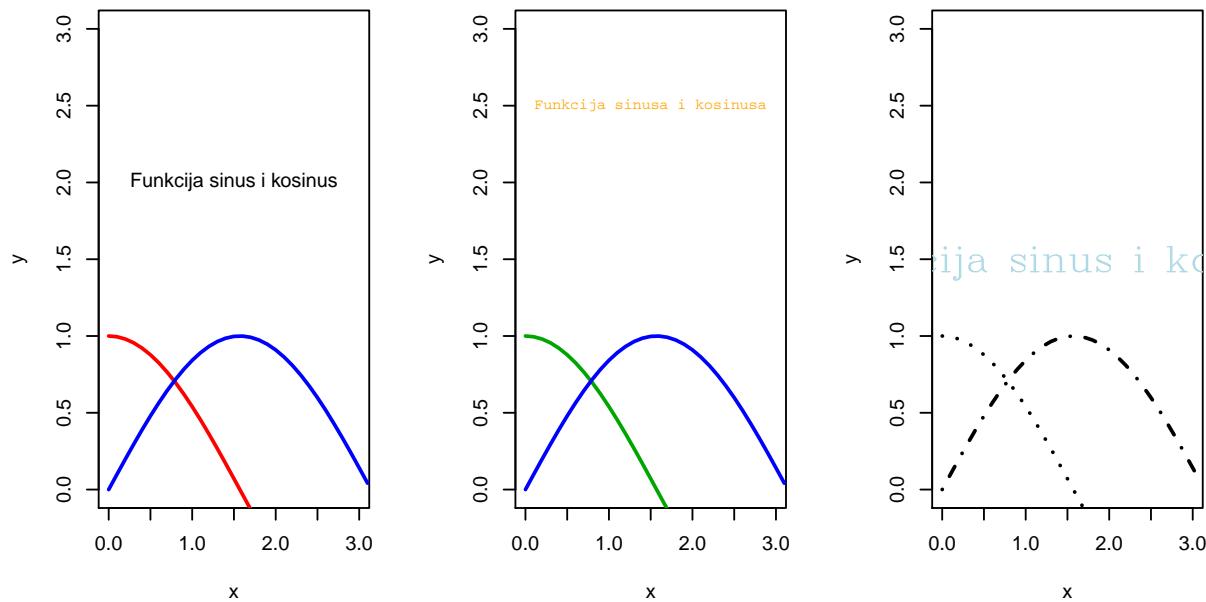
cex=0.8,
col="orange",
family="Courier")

#treći graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linije na postojeći graf
lines(x, y1, lwd=2, lty=3)
lines(x, y2, lwd=2, lty=4)

#dodavanje teksta na postojeći graf; željene koordinate i font
text(1.5,1.5,
      "Funkcija sinus i kosinus",
      cex=2,
      col="lightblue",
      family="HersheySerif",
      font=5)

```



Slika 52: Dodavanje elemenata grafa - tekst

#### 4.1.3.8 Funkcija `box()` {graphics}(low-level)

Ova funkcija crta okvir oko postojećeg grafa u zadanoj boji i vrsti linije. Pogledajte kôd i komentirajte razlike.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3), oma=c(1,1,1,1))
```

```
#prvi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - okvir",
      cex.main=.75)
```

```
#dodajte okvir oko grafa
box(lty=2,
     lwd=3,
     col="red")
```

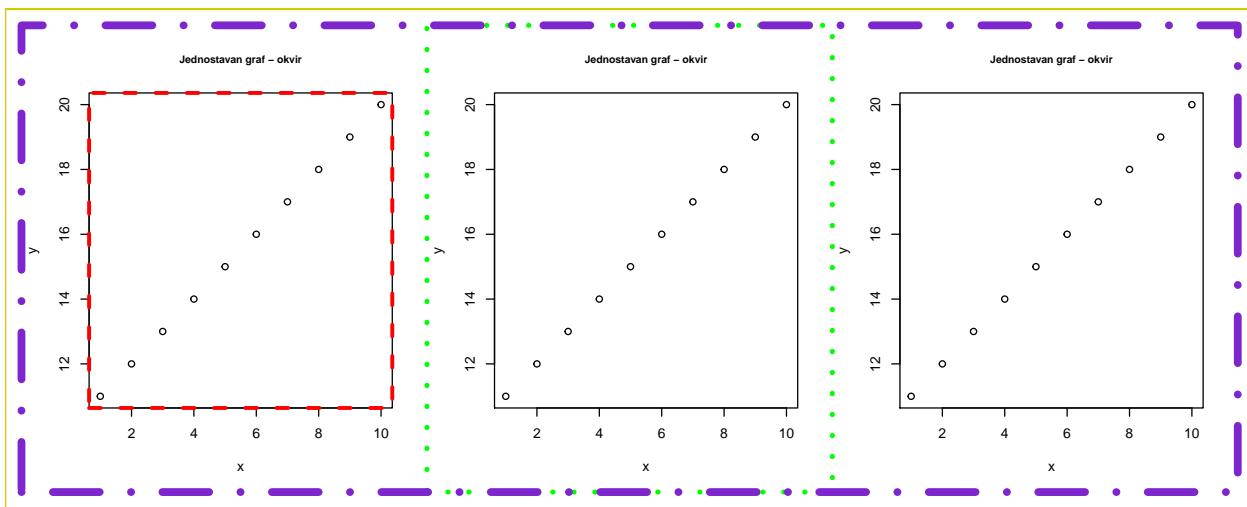
```
#drugi graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - okvir",
      cex.main=0.75)
```

```
#dodavanje okvira oko grafa
box(lty=3,
     lwd=4,
     which="figure",
     col="green")
```

```
#treći graf
plot(df$x, df$y, type="p",
      xlab="x",
      ylab="y",
      main="Jednostavan graf - okvir",
      cex.main=0.75)
```

```
#dodavanje okvira oko grafa unutarne / vanjske područja
box(lty = '1373',
     lwd=6,
     which="inner",
     col="purple3")
```

```
box(lty = 1,
     lwd=2,
     which="outer",
     col="yellow3")
```



Slika 53: Dodavanje elemenata grafa - okvir

#### 4.1.3.9 Funkcija title()/sub(){graphics} (low-level)

Ova se funkcija može koristiti kako bi se dodao glavni naslov na dijagram.

```
#postavite grafički parametar
par(mfrow=c(1,3))

#izrađivanje podataka
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)

#prvi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linije na postojeći graf
lines(x, y1, col="red", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodavanje naslova na postojeći graf
title(main = "Glavni naslov",
      sub = NULL,
      col.main="orange")

#drugi graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodavanje linije na postojeći graf
lines(x, y1, col="red", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodavanje linije na postojeći graf
title(main = "Vrlo dugačak glavni naslov \n za koji je potreban novi redak",
      lheight=2, #razmak između linija u naslovu
```

```

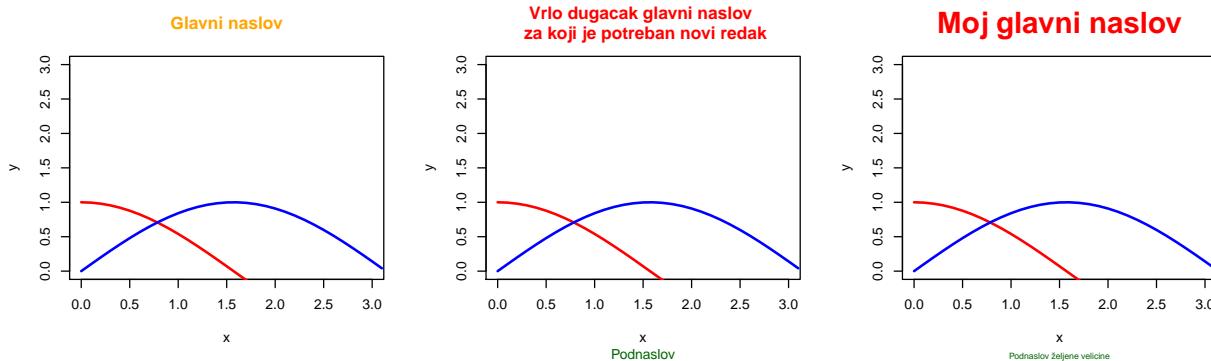
sub = "Podnaslov",
col.main="red",
col.sub="darkgreen")

#treći graf
plot(c(0,3), c(0,3), type="n", xlab="x", ylab="y")

#dodajte linije postojecem dijagramu
lines(x, y1, col="red", lwd=2)
lines(x, y2, col="blue", lwd=2)

#dodavanje linije na postojeci graf
title(main = "Moj glavni naslov",
      sub = "Podnaslov željene veličine",
      col.main="red",
      col.sub="darkgreen",
      cex.main=2.2,
      cex.sub=0.6)

```



Slika 54: Dodavanje elemenata grafa - glavni naslov / podnaslov

**ZA SAMOSTALNI RAD** Korištenjem skupa podataka iris, napravite dijagram raspršivanja (engl. scatterplot):

- koristite varijable Sepal.Length i Sepal.Width
- kao oznake točki koristite varijablu Species
- napravite labele (oznake) 90% od zadane veličine fonta;
- stavite ih na lijevu stranu točaka;
- napravite neki proizvoljni offset od točaka
- označite oznake tamno sivom ili bilo kojom bojom koja Vam se sviđa, osim crne
- zadajte font kao "mono" i napravite oznake podebljanim.

#### 4.1.4 Neke grafičke funkcije visoke razine - nastavak

U nastavku dajemo primjere za ranije navedene funkcije visoke razine unutar paketa *lattice*.

##### 4.1.4.1 Funkcija pairs(){graphics}(high-level)

Funkcija rezultira matricom raspršivanja odabranih varijabli iz skupa podataka. Na primjer koristit ćemo dobro poznati skup podataka iz sustava, skup iris (<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/iris.html>).

Prisjetimo se strukture podataka.

```
#prvih 6 redova
head(iris)

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5         1.4         0.2   setosa
2          4.9         3.0         1.4         0.2   setosa
3          4.7         3.2         1.3         0.2   setosa
4          4.6         3.1         1.5         0.2   setosa
5          5.0         3.6         1.4         0.2   setosa
6          5.4         3.9         1.7         0.4   setosa

#struktura podataka
str(iris)

'data.frame': 150 obs. of 5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

U ovom primjeru, **pairs()** funkcija primjenjuje se na cijeli skup podataka:

```
#skup grafičkih parametara
par(mfrow=c(1,1))

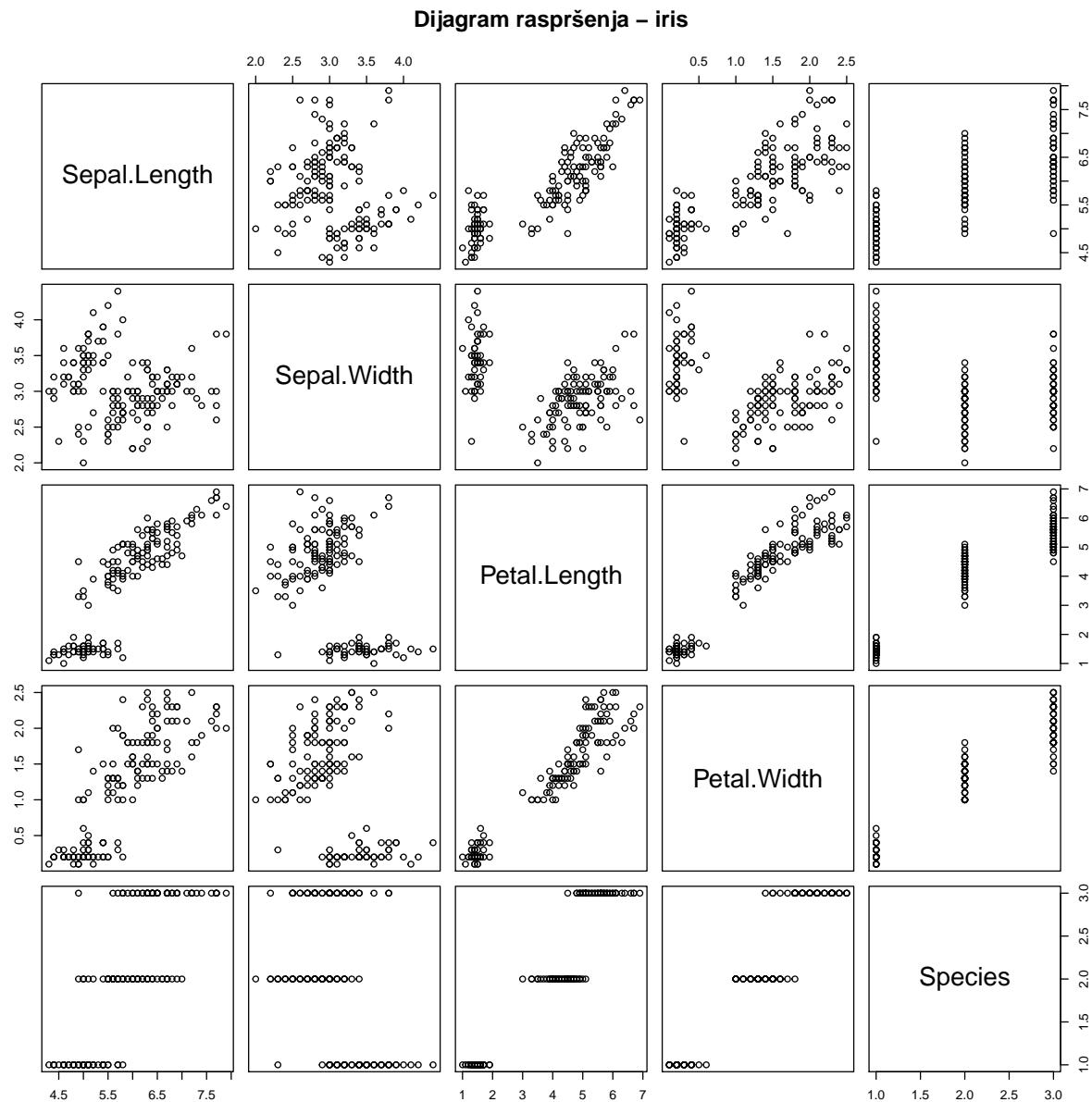
#graf
pairs(iris,
      main="Dijagram raspršenja - iris")
```

U slijedećem primjeru, ista funkcija je primjenjena samo na dio skupa podataka:

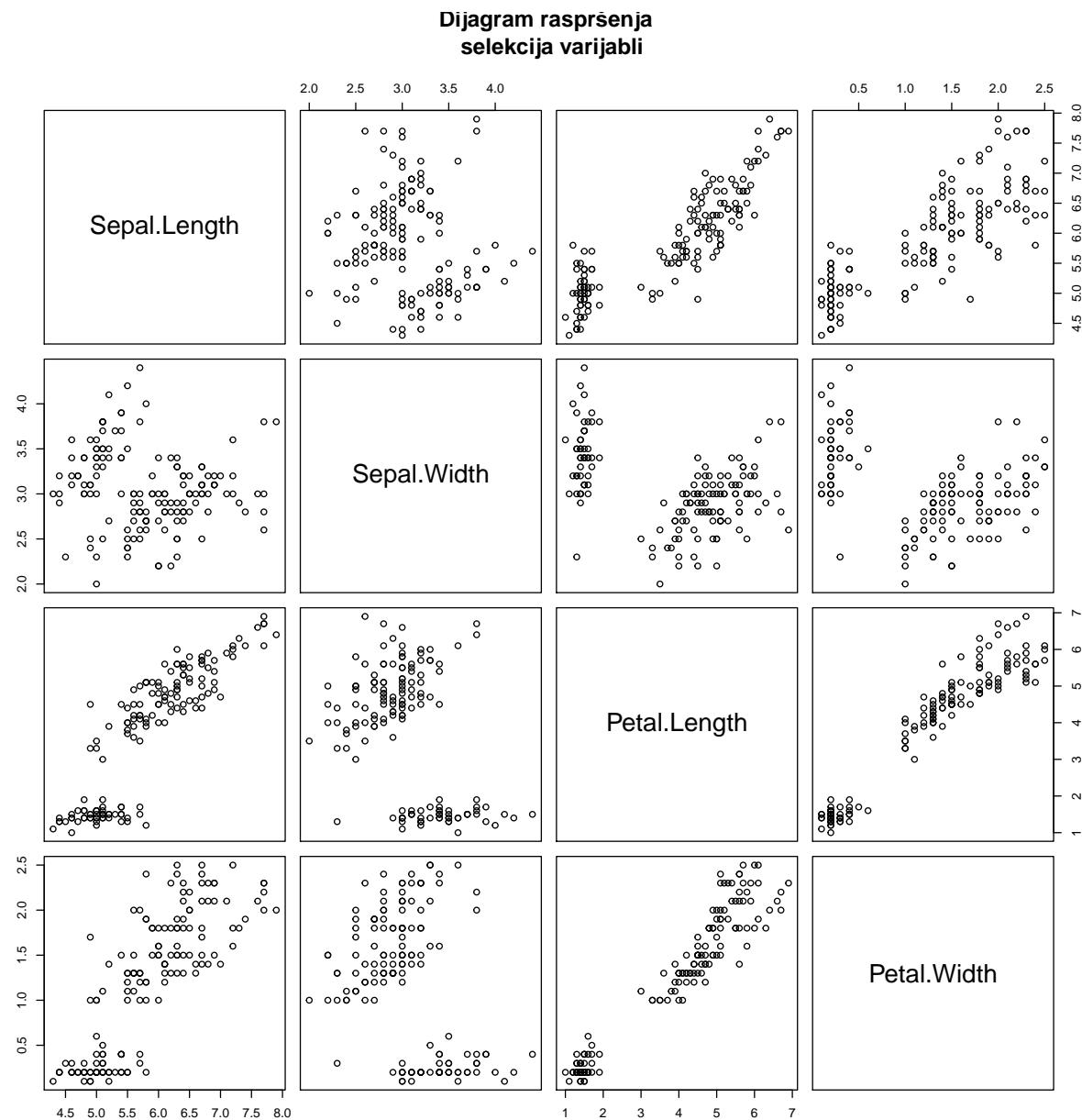
```
#postavljenje grafičkih parametara
par(mfrow=c(1,1))

#graf
pairs(~Sepal.Length+Sepal.Width+Petal.Length+Petal.Width,
      data=iris,
      main="Dijagram raspršenja \n selekcija varijabli")
```

##### 4.1.4.2 Funkcija boxplot(){graphics}(high-level)



Slika 55: Dijagram raspršenja - skup podataka iris



Slika 56: Dijagram raspršenja za sve selektirane varijable

Jedan od najpopularnijih i informativnijih grafičkih prikaza distribucije jedne kvantitativne varijable je Box-Whisker dijagram. Ponovno, koristit će se iris skup podataka iz sustava R. U ovom smo primjeru zainteresirani za varijablu duljine peteljki (Petal.Length).

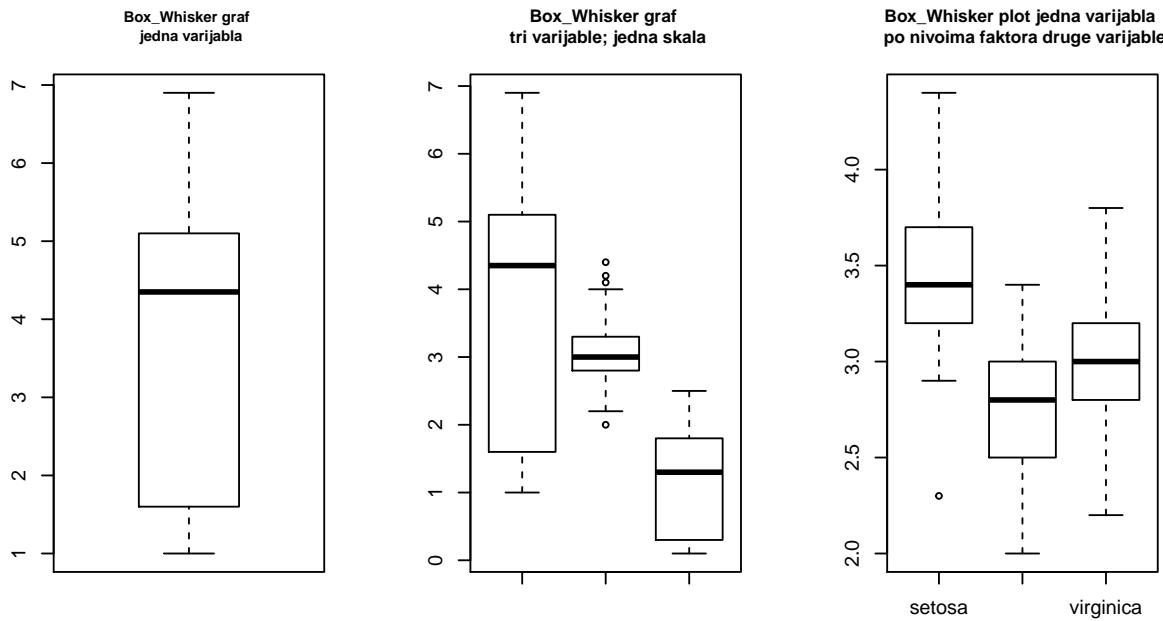
Pogledajte primjer ispod i komentirajte rezultat funkcije. Što se može uočiti u sljedeća tri grafička prikaza? Na koji način ove primjere mijenja argument `notch=T`?

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#prvi graf
boxplot(iris$Petal.Length,
        main="Box_Whisker graf \n jedna varijabla",
        cex.main=.8)

#drugi graf
boxplot(iris$Petal.Length,
        iris$Sepal.Width,
        iris$Petal.Width,
        main="Box_Whisker graf \n tri varijable; jedna skala",
        cex.main=.9)

#treći graf
boxplot(iris$Sepal.Width ~iris$Species,
        main="Box_Whisker plot jedna varijabla \n po nivoima faktora druge varijable",
        cex.main=.9)
```



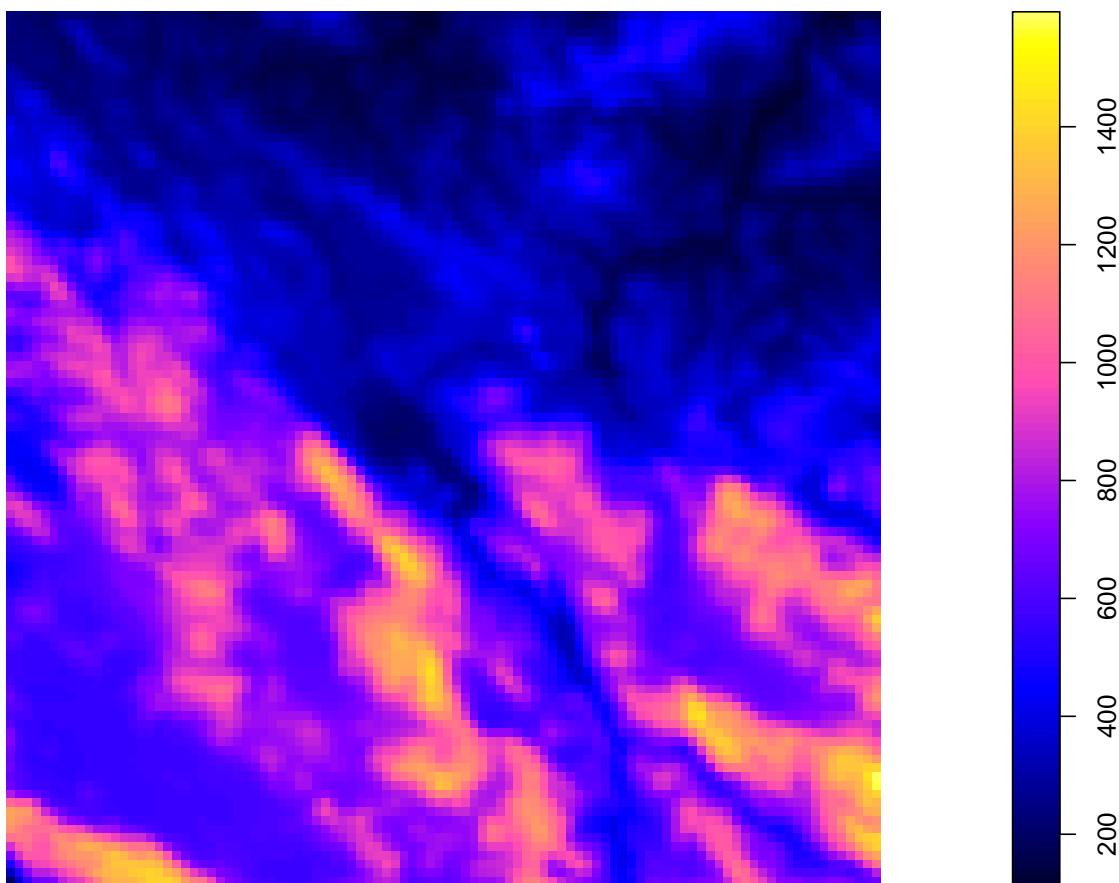
Slika 57: Box-Whisker graf; a) jedna varijabla, b) tri varijable i c) jedna varijabla po nivoima faktora druge varijable (iris)

U ovom tečaju neće biti govora o korištenju generičke funkcije `plot()` kao alata za vizualizaciju geografski referenciranih podataka ali svakako pogledajte primjer koji slijedi. Funkcijom `plot()` vizualiziramo podatke o nadmorskoj visini dijela Republike Hrvatske, učitanoga u sustav R putem `gdal` biblioteke za interoperabilnost geografskih podataka:

```
#učitavamo sgrd/sdat SAGA GIS format u *sp* klasu
dem <- readGDAL("dem_ETRS_1k.sdat")

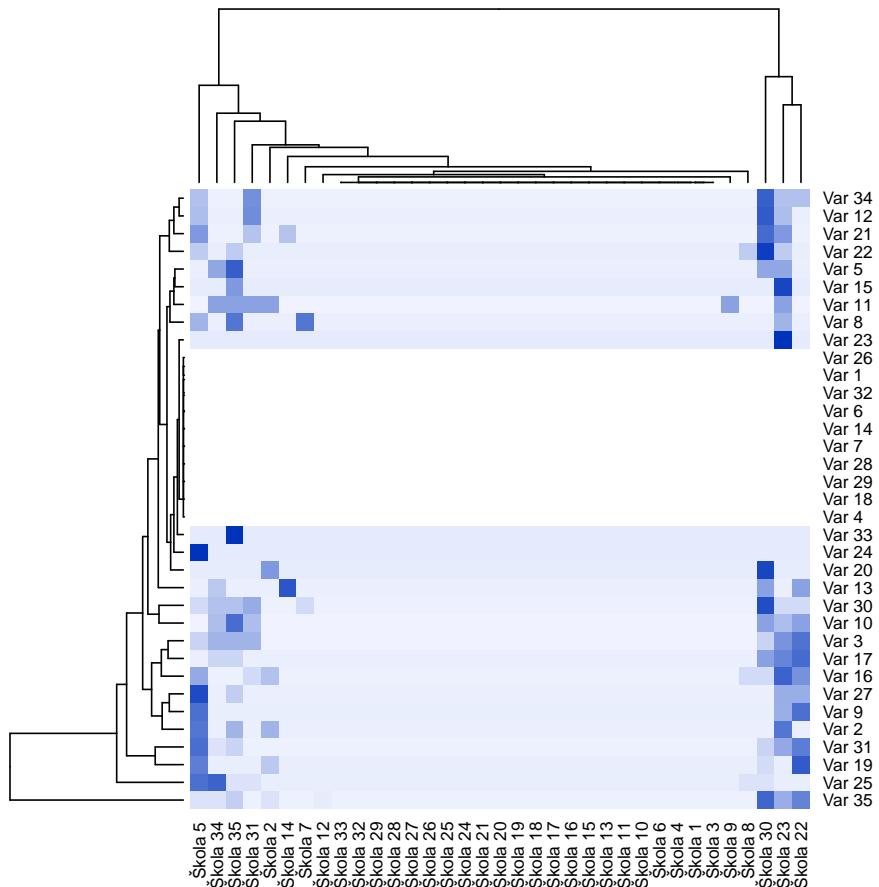
## dem_ETRS_1k.sdat has GDAL driver SAGA
## and has 100 rows and 100 columns

#vizualiziramo objekt klase SGDF, sp paket
plot(dem)
```



Slika 58: Funkcija **plot()**; generička funkcija primjenjena na geografski referenciranim podacima

```
load("podaci_heatmap.RData")
#određivanje boja
moje_boje <- colorRampPalette(c('#f0f3ff', '#0033BB'))(120)
#crtanje
heatmap(podaci_heatmap, Rowv = NULL, Colv =NULL, col=moje_boje)
```

Slika 59: Funkcija **heatmap()**

#### 4.1.4.3 Funkcija `dotchart` (`old:dotplot()`) {`graphics`}(*high-level*)

Funkcija crta Cleveland točkasti dijagram. Kako biste se upoznali prvo s funkcijom, unesite `?dotchart` u R konzolu i pročitajte informacijsku karticu o funkciji.

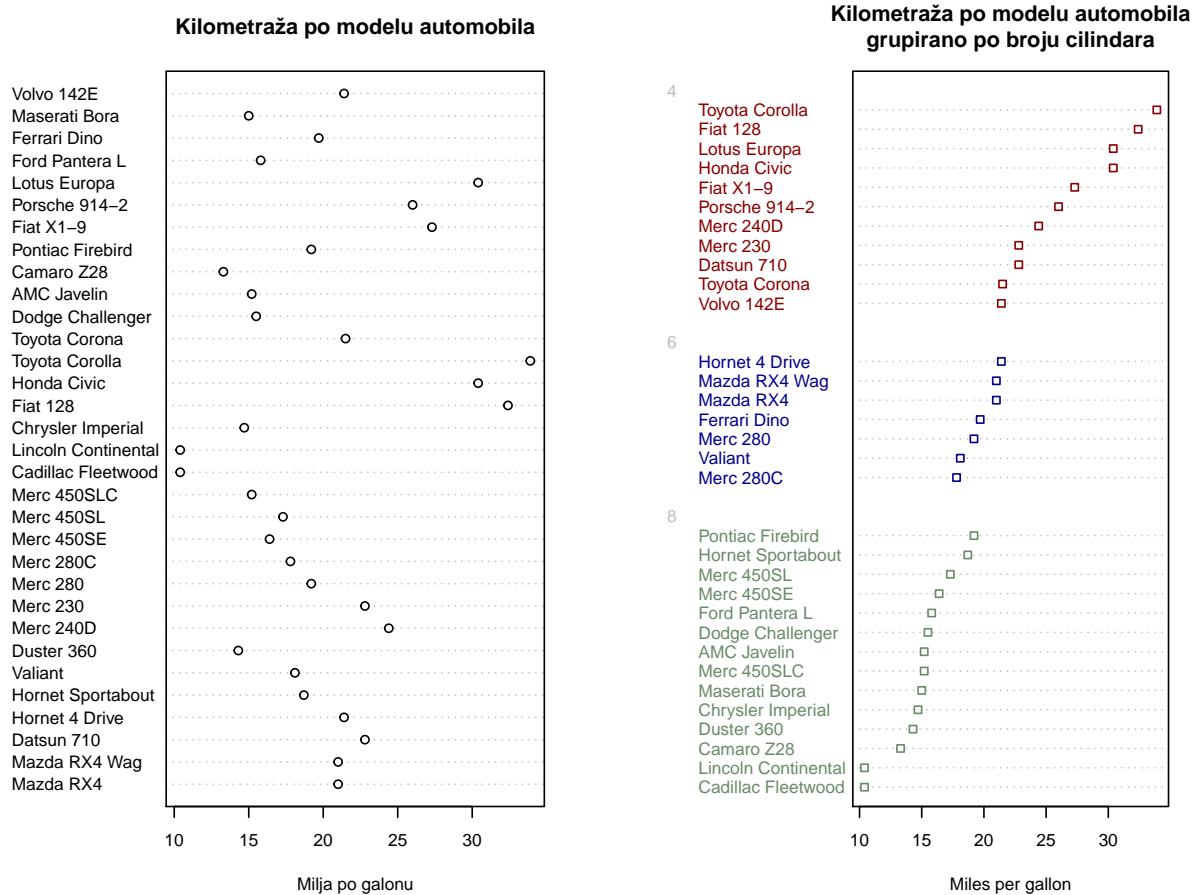
U slijedećem primjeru uočite korištenje parametra `gcolor`; za grupne oznake i vrijednosti koristit će se jedna boja.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,2))

#prvi graf
dotchart(mtcars$mpg,labels=row.names(mtcars),cex=.7,
         main="Kilometraža po modelu automobila",
         xlab="Milja po galonu")

#sortiranje po mpg, group i color po cylinder
#izrađivanje nove varijable sa informacijama o boji
x <- mtcars[order(mtcars$mpg),] # sortiranje po mpg
x$cyl <- factor(x$cyl) # mora biti faktor
x$color[x$cyl==4] <- "red4"
x$color[x$cyl==6] <- "darkblue"
x$color[x$cyl==8] <- "darkseagreen4"

#drugi graf
dotchart(x$mpg,
          labels=row.names(x),
          cex=.7,
          pch=22,
          groups= x$cyl,
          main="Kilometraža po modelu automobila\ngrupirano po broju cilindara",
          xlab="Miles per gallon", gcolor="gray", color=x$color)
```



Slika 60: Dotplot; a) jednostavan dotplot, b) grupirani, sortiran i obojen prema nivoima faktora (mtcars)

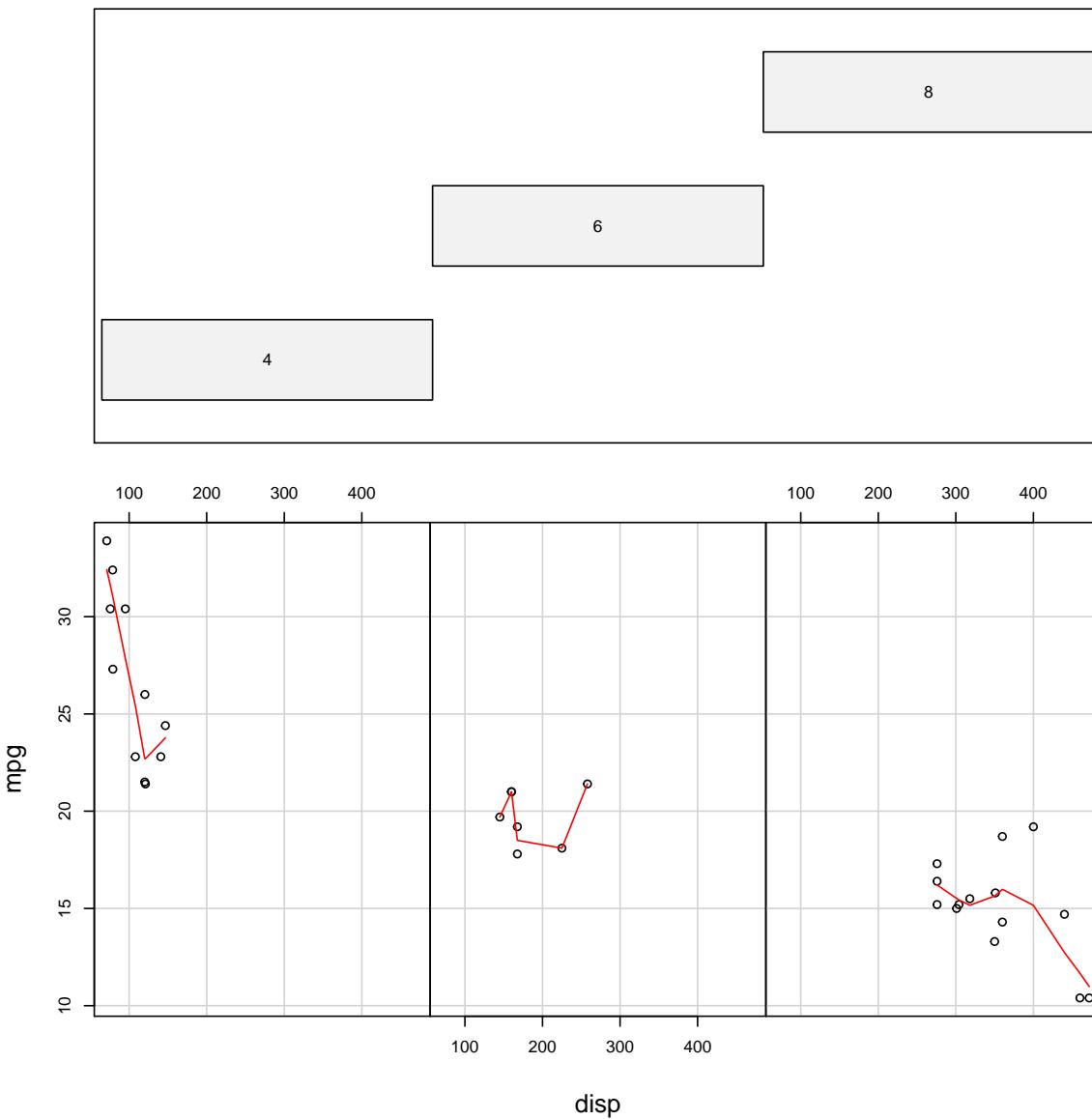
#### 4.1.4.4 Funkcija `cotplot()`{graphics}(high-level)

Ova funkcija proizvodi dvije varijante uvjetovanih prikaza. Prije korištenja bitno je razumjeti sintakstu funkcije, stoga otvorite informacijsku karticu.

Primjer s uvjetovanjem na jednoj varijabli:

```
cotplot(mpg ~ disp | as.factor(cyl), data = mtcars,
        panel = panel.smooth, rows = 1)
```

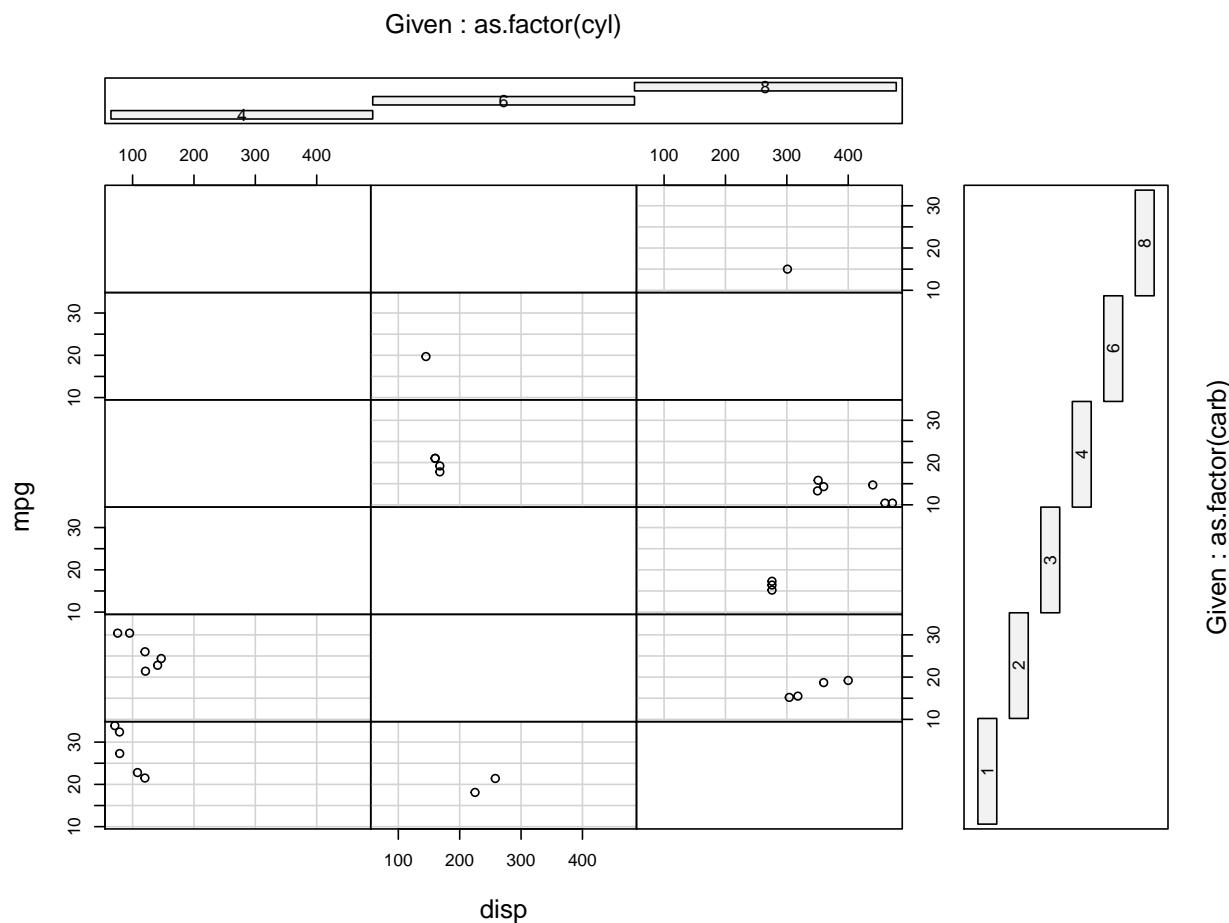
Given : `as.factor(cyl)`



Slika 61: Uvjetovani prikaz ovisnosti dvije numeričke varijable; uvjetovanje na jednoj varijabli (mtcars)

Primjer s uvjetovanjem na dvije varijable:

```
coplot(mpg ~ disp | as.factor(cyl)*as.factor(carb), data = mtcars, rows = 1)
```



Slika 62: Uvjetovani prikaz; uvjetovanje po dvije varijable; panel.smooth (mtcars)

### ZA SAMOSTALNI RAD

Napravite grafički prikaz skupa podataka iris na sljedeći način:

- 1) Kreirajte točkasti dijagram Cleavland za varijablu Sepal.Length, grupirano po vrstama.
- 2) Stavite odgovarajući naslov.

#### 4.1.4.5 Funkcija `pie()`{graphics}(high-level)

. Funkcija crta pita dijagram. Prisjetite se da smo već koristili pita dijagrame za vizualiziranje shema boja te ih preporučamo samo za takve potrebe.

U sljedećim primjerima ćemo izraditi pita dijagram s postotcima. Kako bismo to napravili, prvo moramo izračunati postotke iz podataka i izraditi adekvatne oznake. Molimo pogledajte sljedeći kôd i komentirajte.

```
#izrađivanje određenih podataka
pie_podaci <- c(20, 15, 3, 22, 11, 17)

#izračun postotaka
postotak <- round(pie_podaci/sum(pie_podaci)*100)

#izrađivanje labela (Oznaka)
lab <- c("US", "UK", "HR", "DE", "FR", "IT")

#dodavanje postotka labelama
lab_1 <- paste(lab, postotak)

#dodajemo oznaku % labelama
lab_2 <- paste(lab_1, "%", sep="")
```

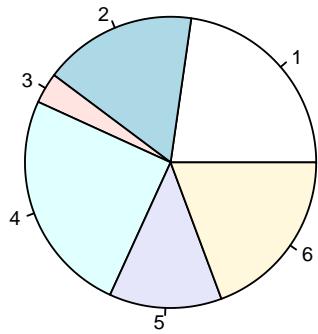
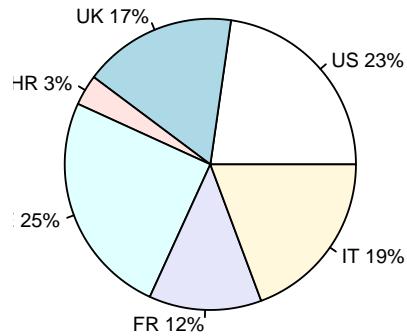
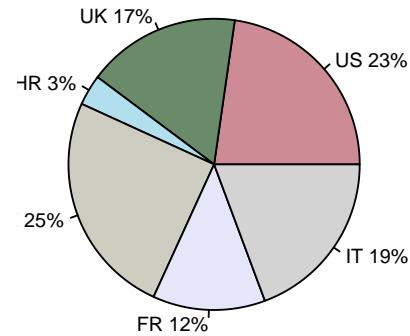
Sada, kada imamo sve komponente, možemo napraviti grafove:

```
#postavljanje grafičkih parametara set graphical parameters
par(mfrow=c(1,3), mar=c(0,0,3,1))

#prvi graf
pie(pie_podaci,
     main="Pita graf; default")

#drugi graf
pie(pie_podaci,
     labels = lab_2,
     main="Pita graf; labele")

#treći graf
pie(pie_podaci,
     labels = lab_2,
     col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3", "lavender", "lightgray"),
     main="Pita graf; labele, korisničke boje")
```

**Pita graf; default****Pita graf; labele****Pita graf; labele, korisnicke boje**

Slika 63: Pita dijagram a) jednostavnja, b) s kreiranim labelama i c) korisničke labele i boje

Moguće je postaviti dodatne prilagodbe vezano za grafove, kao što to pokazuje sljedeći primjer:

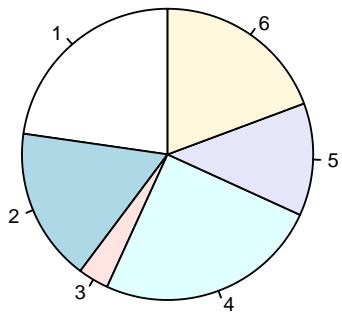
```
#postavljanje grafičkih parametara
par(mfrow=c(1,3), mar=c(0,0,3,1))

#prvi graf
pie(pie_podaci,
    init.angle=90,
    main="Pita dijagram; zadan početni kut")

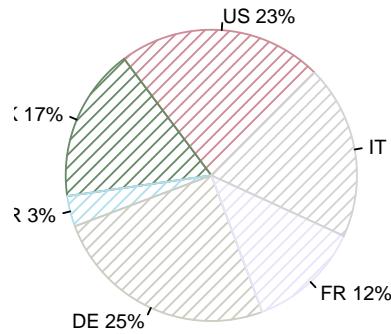
#drugi graf
pie(pie_podaci,
    init.angle=45,
    labels = lab_2,
    density= 20,
    col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3", "lavender", "lightgray"),
    main="Pita dijagram \n boje, labele, gustoća, kut")

#treći graf
pie(pie_podaci,
    labels = lab_2,
    col=rep("white", length(pie_podaci)),
    main="Pita dijagram; označen, bez boja \n prilagodba kuta i prošaranosti")
```

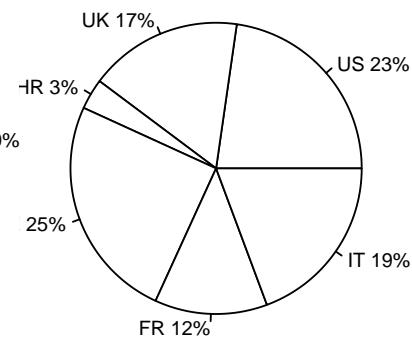
Pita dijagram; zadan pocetni kut



Pita dijagram boje, labele, gustoca, kut



Pita dijagram; ozначен, bez boja prilagodba kuta i prošaranosti



Slika 64: Pita graf; dodatne postavke; a) određivanje početnog kuta; b) određivanje visine, granica i prošaranosti i c) opoziv boja na grafu

Molimo, gledajući primjere odgovorite na slijedeće:

- 1) Koji argument funkcije `pie` kontrolira smjer crtanja pita dijagrama?
- 2) Kako su **default** kriške pita dijagrama organizirane? Što znači automatsko označavanje?
- 3) Opišite proces *suppressing* boja u pita dijagramima?

Pita dijagrami su popularni, ali ih mnogi analitičari/statističari ne smatraju pogodnim za prezentiranje podataka. Jedan od razloga je mogućnost manipuliranja učincima korištenjem boja, početnim kutom, sjenčanjem ili 3D vizualiziranjem pita dijagrama. Kao primjer, sljedeći je dijagram pripremljen pomoću `pie3D` funkcije iz `plotrix` paketa. U opisu `pie3D()` funkcije stoji sljedeće: “**Pita dijagrami su loš način prikaza informacija. Čovjekovo oko je u mogućnosti dobro procijeniti linearne ovisnosti ali je loše u prosudbi relativnih područja. Za prikaz podataka koje ljudi obično prikazuju pita dijagramom bolje je koristiti stupičasti dijagram (`barplot`) ili točkasti dijagram (`dotplot`).**”

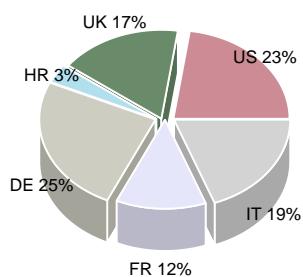
```
#postavljanje grafičkih parametara
par(mfrow=c(1,3), mar=c(0,0,1,1))

#prvi graf
pie3D(pie_podaci,
       radius=1.1,
       height=0.2,
       border="white",
       labels=lab_2,
       explode=0.1,
       theta=1.2,
       labelcex=0.6,
       col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3", "lavender", "lightgray"),
       main="3D pita dijagram; varijanta 1")
```

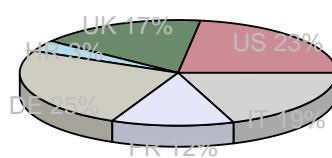
```
#drugi graf
pie3D(pie_podaci,
       radius=1.5,
       labels=lab_2,
       labelcex=0.8,
       labelcol="gray",
       col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3", "lavender", "lightgray"),
       main="3D pita dijagram; varijanta 2")

#treći graf
pie3D(pie_podaci,
       radius=1.2,
       labels=lab_2,
       explode=0.3,
       labelcex=0.7,
       border="white",
       col= c("lightpink3", "darkseagreen4", "lightblue2", "ivory3", "lavender", "lightgray"),
       main="3D pita dijagram; varijanta 3")
```

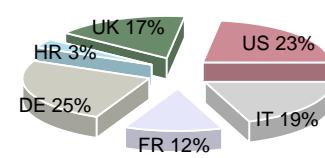
3D pita dijagram; varijanta 1



3D pita dijagram; varijanta 2



3D pita dijagram; varijanta 3

Slika 65: Pita dijagram; funkcija `pie3D` iz `plotrix` paketa, tri varijante

#### 4.1.4.6 Funkcija `hist(){graphics}(high-level)`

Generička funkcija `hist()` računa histogram danih vrijednosti podataka. Ako je argument postavljen na `plot = TRUE`, rezultirajući objekt klase "histogram" crta se pomoću `plot.histogram` metode.

U sljedećem primjeru napraviti ćemo sljedeće histograme: a) jednostavan histogram s predefiniranim (engl. `default`) vrijednostima za varijablu `Sepal.Length` iz skupa podataka iris i b) histogram iste varijable, ali ćemo promijeniti broj razdioba/binova na 35 te ćemo prikazati i vrijednosti iz skupa podataka korištenjem funkcije `rug()`; c) na trećem dijelu ćemo funkcijom `hist()` nacrtati, ne prave frekvencije, već vjerojatnosti pojave opservacije u pojedinom rasponu (engl. `bin`) te na isome prikazu superponirati i normalnu razdiobu. Prijе početka moramo se upoznati s funkcijom i pogledati njezinu informacijsku karticu.

Uvijek budite svjesni toga da je histogramski prikaz varijable loš za određivanje njezina oblika budući je veoma osjetljiv na broj intervala unutar kojih klasificiramo naše podatke, ali je veoma pogodan za vizualizaciju naših podataka (eksperimenata) u usporedbi s teorijskom distribucijom, kao što je pokazano na c dijelu slike.

```
#postavljanje grafičkih parametara
par(mfrow=c(1,3))

#prvi graf
hist(iris$Sepal.Length,
      col="lightblue",
      main="Histogram, default broj intervala",
      cex.main=1.2)

#drugi graf
hist(iris$Sepal.Length,
      col="lightblue",
      main="Histogram, 35 intervala",
      cex.main=1.2,
      breaks=35)
rug(iris$Sepal.Length) #rug

#treći graf
h<- hist(iris$Sepal.Length,
          col="lightblue",
          main="Histogram duljine listića; vjerojatnosti \n superponirana normalna distribucija",
          cex.main=1.2,
          breaks=35,
          freq=F)

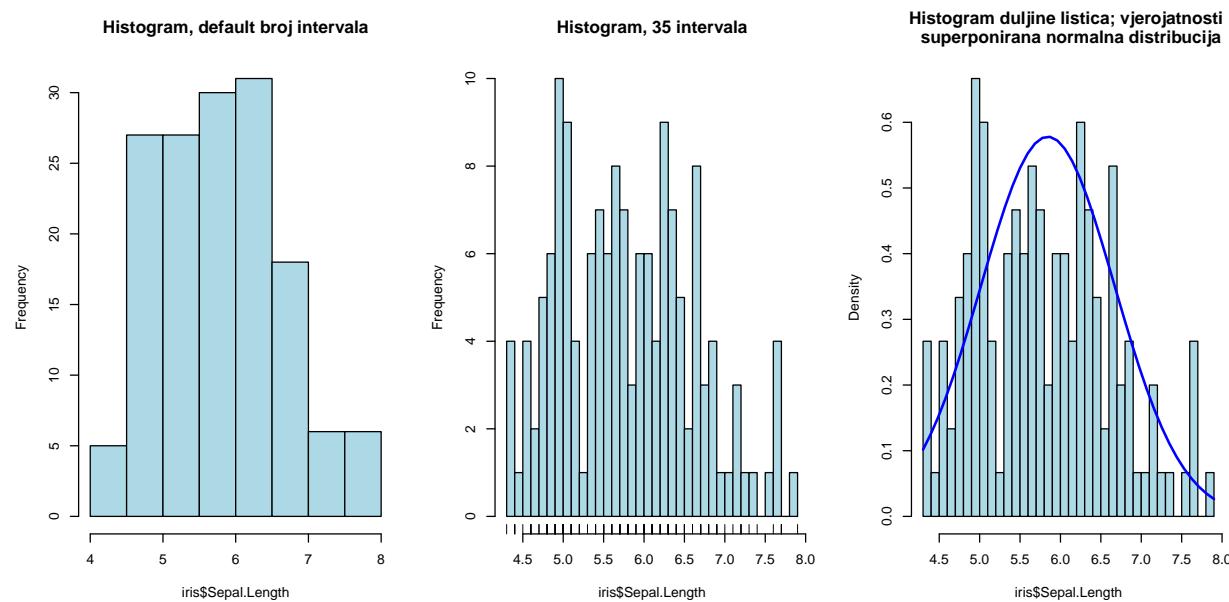
#računamo vrijednosti normalne distribucije na zadatom intervalu
xfit<-seq(min(iris$Sepal.Length),max(iris$Sepal.Length),length=40)
yfit<-dnorm(xfit,mean=mean(iris$Sepal.Length),sd=sd(iris$Sepal.Length))
yfit <- yfit*diff(h$mid[1:2])*length(x)

#dodavanje linija - fit; normalna distribucija
lines(xfit, yfit, col="blue", lwd=2)

#postavljanje grafičkih parametara
par(mfrow=c(1,1))

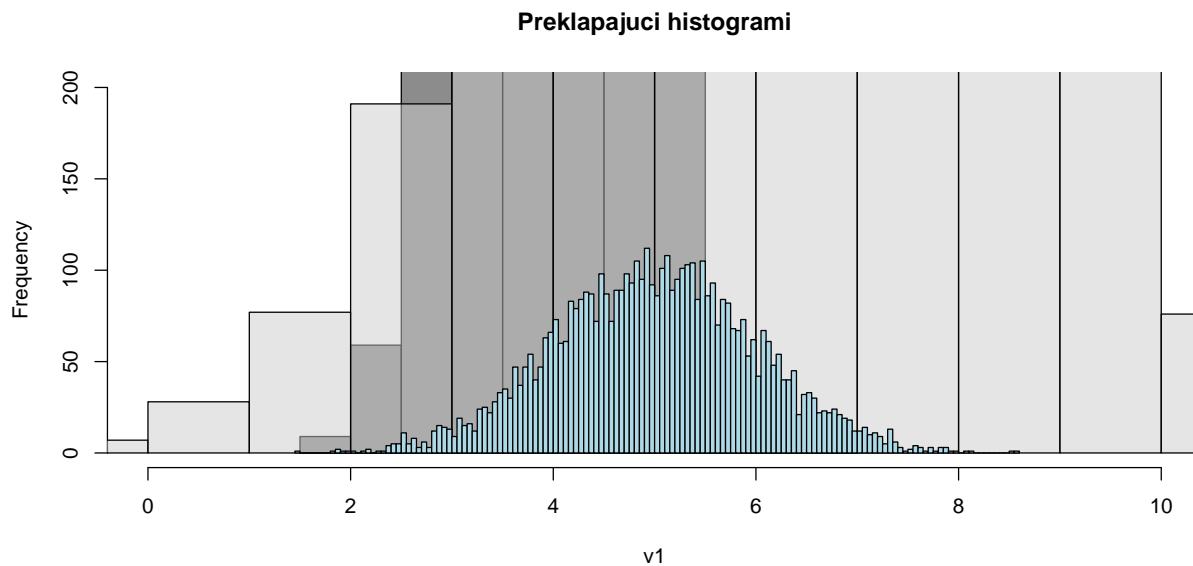
#početna vrijednost generatora slučajnih brojeva
set.seed(113)

#generiramo slučajne varijable iz različitih razdioba
```

Slika 66: Funkcija za crtanje histograma **hist**

```
v1 <- rnorm(n=5000, mean=4, sd=0.7)
v2 <- rnorm(n=5000, mean=6, sd=2)
v3 <- rnorm(n=5000, mean=5, sd=1)

#crtamo obje raspodjele na jednom grafu
#rgb parametar za transparentnost
hist(v1, col=rgb(0.1,0.1,0.1,0.5),
      xlim=c(0,10),
      ylim=c(0,200),
      main="Preklapajući histogrami")
hist(v2, col=rgb(0.8,0.8,0.8,0.5), add=T)
hist(v3,
      col=col2rgb("lightblue")[1],
      col=col2rgb("lightblue")[2],
      col=col2rgb("lightblue")[3],0.5, add=T)
```



Slika 67: Funkcija za crtanje histograma **hist**; prikaz tri raspodjele na jednom grafu

#### 4.1.4.7 Funkcija **barplot()**{graphics}(high-level)

Funkcijom crtamo stupičasti dijagram s vertikalnim i horizontalnim oznakama; adekvatan grafički prikaz za vizualiziranje jedne kategoriske varijable. U sljedećem primjeru vizualizirat ćemo kategorije (dobivene funkcijom **cut()**) raspona nadmorskih visina skupa podataka iz sustava *volcano*.

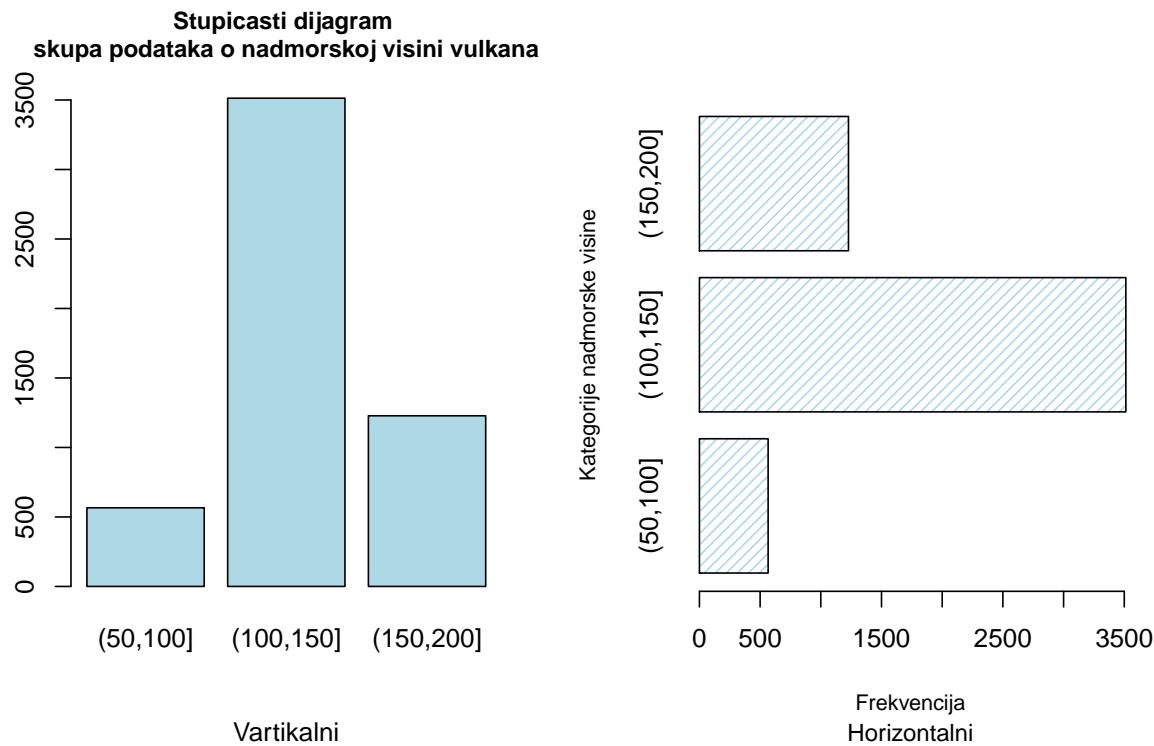
```
#postavljanje grafičkih parametara
par(mfrow=c(1,2))

#priprema podataka; kategorije
volcano_categories <- cut(volcano, c(50,100, 150, 200))

#dijagram 1
barplot(table(volcano_categories), col="lightblue",
        main="Stupičasti dijagram \nskupa podataka o nadmorskoj visini vulkana",
        cex.main=0.9, sub="Vartikalni")

#dijagram 2
barplot(table(volcano_categories), col="lightblue",
        main=NULL,
        cex.main=0.9,density=20,
        horiz=T, sub="Horizontalni",
        cex.sub=0.9,
        ylab= "Kategorije nadmorske visine", xlab="Frekvencija", cex.lab=.8)
```

Pogledajte prethodne slike i komentirajte sve razlike između dva dijagrama. Pronadite dijelove kôda koji kreiraju te razlike. Što će se dogoditi ako izostavimo dio iz drugog grafa **main=NULL**? Isprobajte to u R konzoli i komentirajte.



Slika 68: Funkcija **barplot**; vertikalni i horizontalni stupičasti dijagram

#### 4.1.4.8 Funkcija `mosaicplot()`{graphics}(high-level)

Mozaik dijagram je popularan graf za vizualiziranje dvije ili više kategorijskih varijabli.

Prvo ćemo pripremiti određene podatke s kategorijskim varijablama.

#kreirajte skup podataka s tri faktorske varijable

## #reproducibilna simulacija

```
#početni broj za generator slučajnih brojeva  
set.seed(1)
```

```
moji_podaci <- data.frame (faktor1 = sample (c("A", "B", "C", "D"), 200, replace = TRUE),  
faktor2 = sample (c("HL", "PS", "DS"), 200, replace = TRUE),  
faktor3 = sample (c("Male", "Female"), 200, replace = TRUE))
```

```
#postavke grafičkih parametara za dijagram  
par(mfrow=c(1,2))
```

#izradivanje grafa

```
#mosaicplot(~ fact1 + fact2 + fact3, data=moji_podaci, col=T,  
#main="Mosaicplot of three categorical variables")
```

#isto kao ovo

```
mosaicplot(table(moje_podaci)).
```

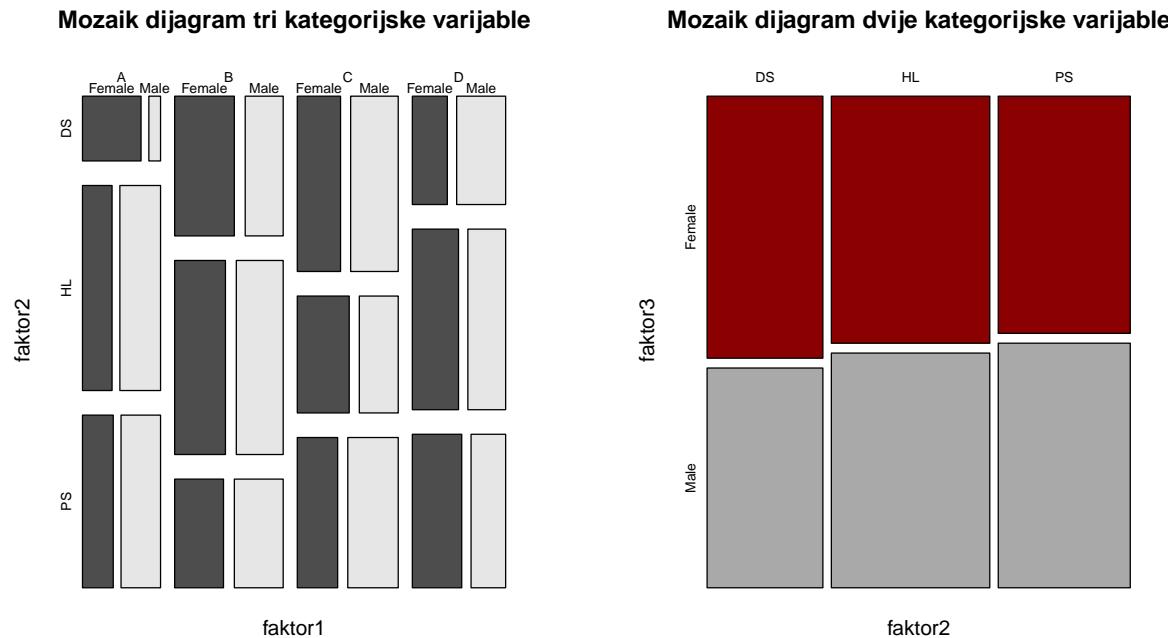
```

col=T,
main="Mozaik dijagram tri kategoriske varijable")

#drugi graf

mosaicplot(~ faktor2 + faktor3,
           data=moji_podaci,
           col=c("red4", "darkgray"),
           main="Mozaik dijagram dvije kategoriske varijable")

```



Slika 69: Vizualizacija interakcije za više od jedne kategoriske varijable - mozaik dijagram tri kategoriske varijable

Primjer koji slijedi nije iz sustava R, već podaci ankete djelatnika jedne kompanije. Pogledajmo prvo strukturu podataka koje ćemo crtati:

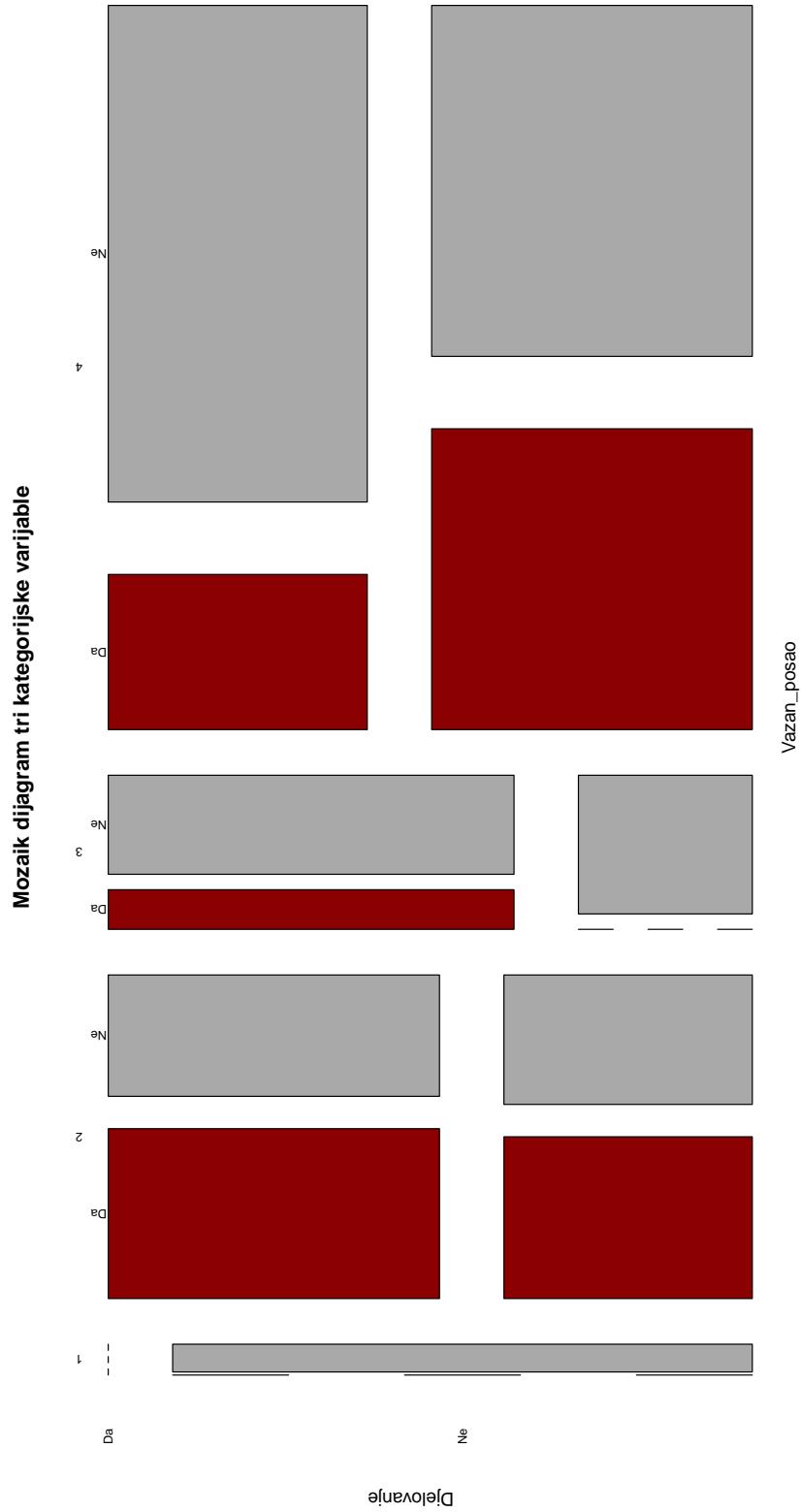
```
head(podaci_1)
```

```

##      Vazan_posao Zadovoljstvo Djelovanje
## 1            3       Ne      Da
## 3            4       Ne      Da
## 6            4       Da     Ne
## 8            2       Da     Ne
## 11           2       Da     Ne
## 12           4       Ne      Da

```

Gotovo uvijek postoji više mogućnosti kako nešto napraviti u R-u. Ima mnogo kontribuiranih paketa koji se bave različitim aspektima vizualiziranja podataka i/ili analize podataka. Na primjer paket **vcd** je paket specijaliziran za vizualiziranje kategoriskih podataka *citation(vcd)*.



Slika 70: Vizualizacija interakcije za više od jedne kategorističke varijable - mosaik dijagram tri kategorističke varijable, primjer 2

#### 4.1.4.9 Funkcija `assocplot() {graphics} (high-level)`

Asocijativni dijagram je popularan grafički prikaz za vizualiziranje dvije ili više kategorijskih varijabli. Prije vizualizacije uvijek je dobro potrebno ispitati strukturu podataka. Zapamtite, već smo kreirali skup podataka `moji_podaci`, klase `data.frame` koji sadrži kategoriske varijable.

```
#sjetite se strukture moji_podaci data frame-a
str(moji_podaci)

## 'data.frame':    200 obs. of  3 variables:
## $ faktor1: Factor w/ 4 levels "A","B","C","D": 2 2 3 4 1 4 4 3 3 1 ...
## $ faktor2: Factor w/ 3 levels "DS","HL","PS": 2 2 3 2 2 3 3 2 2 1 ...
## $ faktor3: Factor w/ 2 levels "Female","Male": 1 2 1 1 1 1 2 1 2 1 ...

#cijela tablica
moja_tablica <- table(moji_podaci)

#agregirani podaci
moja_marginalna_tablica <- margin.table (moja_tablica, c(1,3))
```

Kako je i navedeno, paket `vcd` je specijaliziran za vizualiziranje kategorijskih podataka. Funkcija `assoc()` je fleksibilnija i iscrpnija primjena dijagrama asocijacije u grid grafičkom sustavu.

```
#postavka grafičkih parametara za dijagram
par(mfrow=c(1,1))

#dijagram
assocplot(moja_marginalna_tablica,
          main="Asocijativni dijagram dvije kategoričke varijable",
          col=c("lightpink3", "lightgray"))
```

```
#struktura podataka
str(podaci_1)

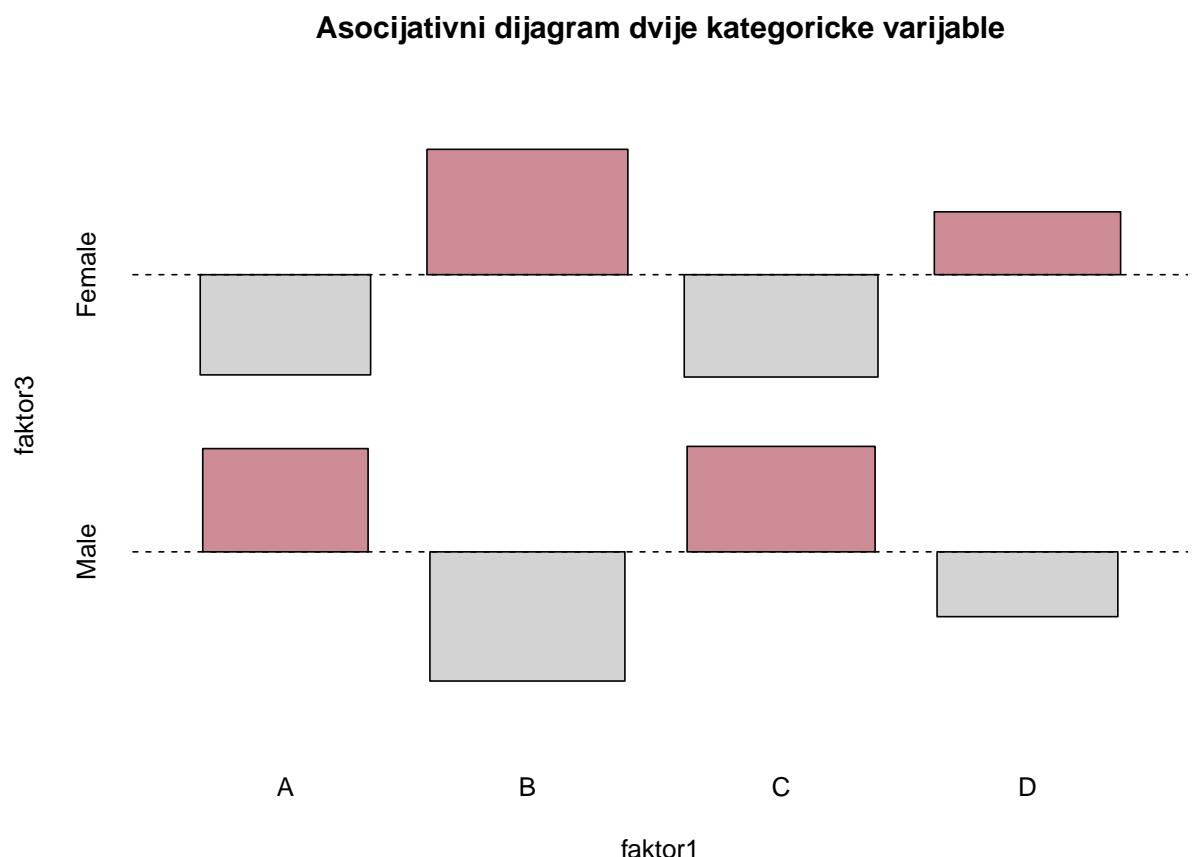
## 'data.frame':    80 obs. of  3 variables:
## $ Vazan_posao : Factor w/ 4 levels "1","2","3","4": 3 4 4 2 2 4 4 3 2 2 ...
## $ Zadovoljstvo: chr  "Ne" "Ne" "Da" "Da" ...
## $ Djelovanje  : chr  "Da" "Da" "Ne" "Ne" ...

#cijela tablica
moja_tablica_2 <- table(podaci_1)
```

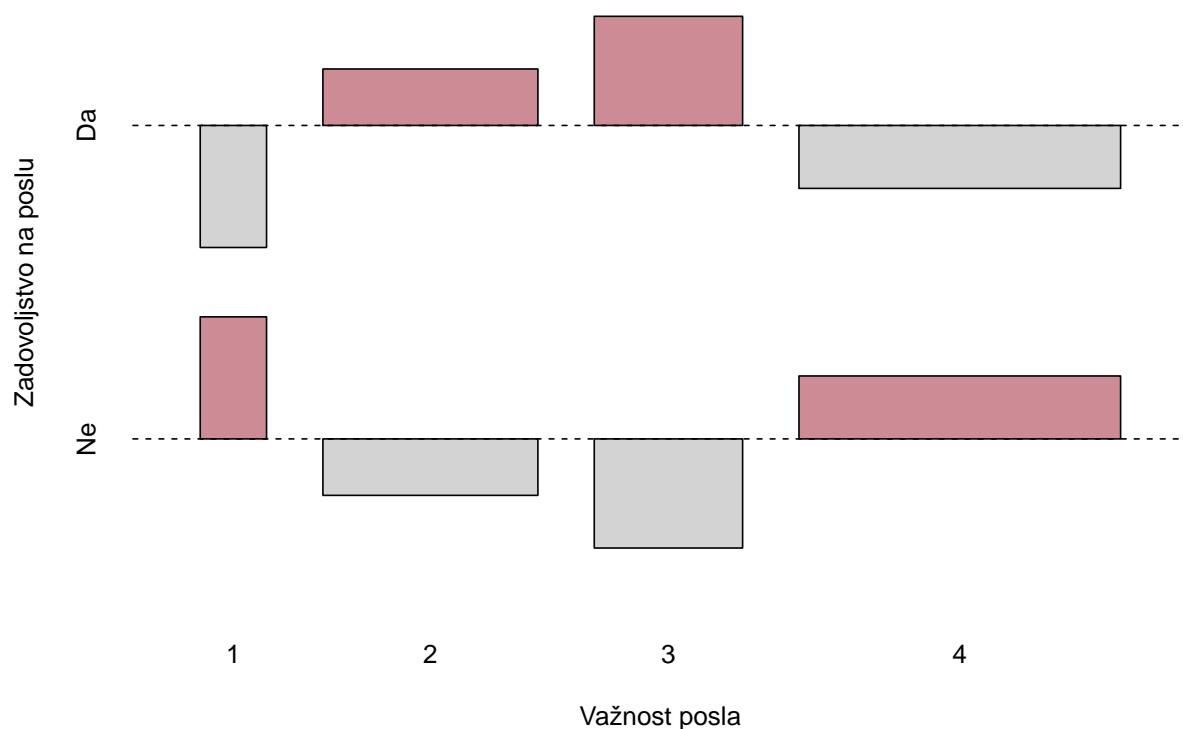
```
#agregirani podaci
moja_marginalna_tablica_2<- margin.table (moja_tablica_2, c(1,3))
```

```
#postavka grafičkih parametara za dijagram
par(mfrow=c(1,1))

#dijagram
assocplot(moja_marginalna_tablica_2,
          main="Asocijativni dijagram dvije kategoričke varijable",
          col=c("lightpink3", "lightgray"),
          xlab="Važnost posla",
          ylab="Zadovoljstvo na poslu")
```



Slika 71: Vizualizacija interakcije dviju kategorijskih varijabli - asocijativni dijagram

**Asocijativni dijagram dvije kategoricke varijable**

Slika 72: Vizualizacija interakcije dviju kategorijskih varijabli - asocijativni dijagram

#### 4.1.4.10 Funkcija `persp()`{graphics}(high-level)

Ova funkcija crta odgovarajuće grafove površine preko x - y ravnine. Funkcija `persp()` je generička funkcija. Ova vrsta grafa je, također, poznata kao površina okvira. Prvo se upoznajmo s funkcijom. Molim da otvorite informacijsku karticu funkcije tako što ćete u R konzolu unijeti `?persp`.

U sljedećem ćemo primjeru izraditi odgovarajuće dijagrame numeričkih matrica s topografskim podacima Maunga Whau vulkana iz Aucklanda, pomoću skupa podataka `volcano` iz R sustava:

```
#priprema podataka
dim(volcano)

## [1] 87 61
x <- 1:87

#isto kao
#x <- 1:dim(volcano)[1]

y<- 1:61
#y <- 1:dim(volcano)[2]

x

## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
## [70] 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87

y

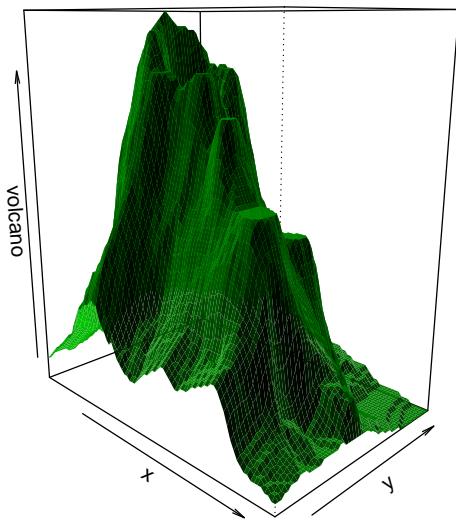
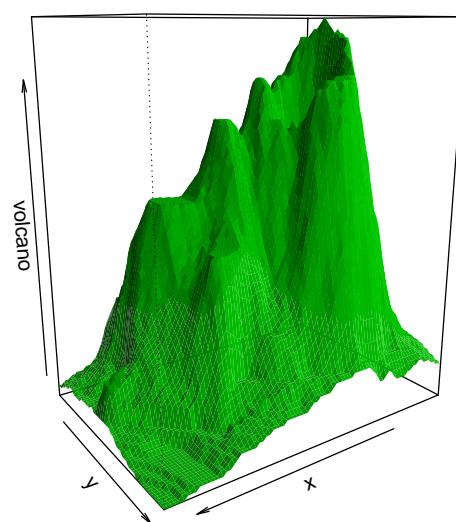
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61

par(mfrow=c(1,2))

#prvi graf
persp(x,y, volcano,
       theta = 45,
       phi = 15,
       col = "green3",
       scale = F,
       ltheta = -120,
       shade = 0.75,
       border = NA,
       box = T,
       main="Perspektiva vulkana 1")

#drugi graf
persp(x,y, volcano,
       theta = 145,
       phi = 15,
       col = "green3",
       scale = F,
       ltheta = -120,
       shade = 0.4,
```

```
border = NA,  
box = T,  
main="Perspektiva vulkana 2")
```

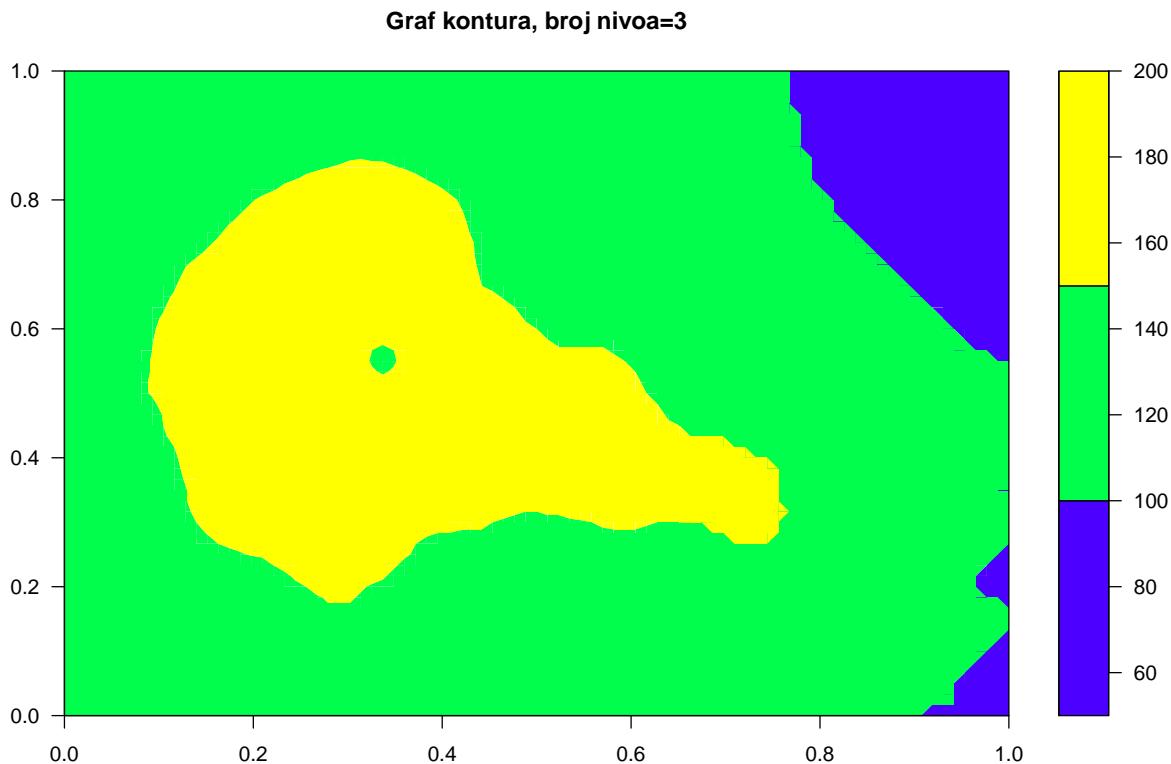
**Perspektiva vulkana 1****Perspektiva vulkana 2**

Slika 73: Graf površine; *perspective plot* na skupu podataka `volcano` - numerička matrica s vrijednostima topografskih visina za Maunga Whau vulkan.

#### 4.1.4.11 Funkcija `filled.contour()`{graphics}(high-level)

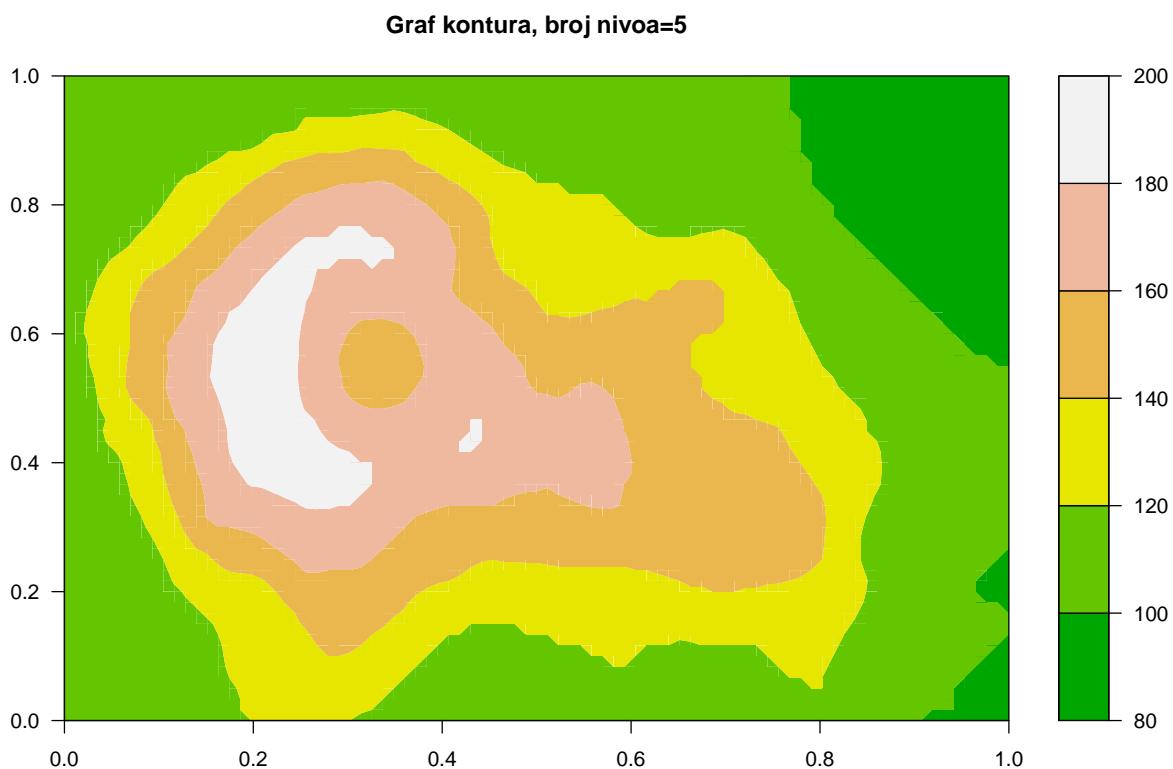
Ova funkcija proizvodi grafički prikaz kontura (grafiku razine) s područjima između kontura koje popunjava puna boja. U slijedećem primjeru, ponovno koristimo podatke iz skupa `volcano` koji se nalaze u sustavu:

```
filled.contour(volcano,
               color = topo.colors, nlevels=3,
               main="Graf kontura, broj nivoa=3")
```



Slika 74: Prikaz kontura, engl. *contour plot*; plot na skupu podataka `volcano` - numerička matrica s vrijednostima topografskih visina za Maunga Whau vulkan; tri nivoa; `nlevels= 3`

```
filled.contour(volcano,
               color = terrain.colors, nlevels=5,
               main="Graf kontura, broj nivoa=5")
```



Slika 75: Prikaz kontura, *contour plot*; plot na skupu podataka volcano - numerička matrica s vrijednostima topografskih visina za Maunga Whau vulkan; pet nivoa; nlevels= 5.

#### 4.1.4.12 Funkcija `image(){graphics}`

Funkcija prikazuje sliku boje. Ova funkcija zahtijeva listu x i y vrijednosti koje pokrivaju grid vertikalnih vrijednosti koje će se koristiti kreiranje površine. Ovdje su visine precizirane kao tablica vrijednosti, što je, u našem slučaju, sačuvano kao objekt z tijekom kalkulacija lokalnoga trenda površine. Uočite korištenje parametra `add` - logical; ako je postavljen na TRUE, dodaje sliku na postojeći graf. Ovaj argument se rijetko koristi, budući da funkcija "crtanje" preko postojeće grafike.

Pogledajte primjer na skupu podataka `volcano`:

```
#postavljanje grafičkih parametara
par(mfrow=c(1,2))

#prvi graf
image(x,y, volcano,
      col = heat.colors(12),
      main="Image funkcija, heat.colors skup podataka volcano",
      cex.main=.9)

#drugi graf
image(x,y, volcano,
      col = topo.colors(12),
      main="Image funkcija, topo.colors skup podataka volcano",
      cex.main=.9)
```

Image funkcija, `heat.colors` skup podataka `volcano`

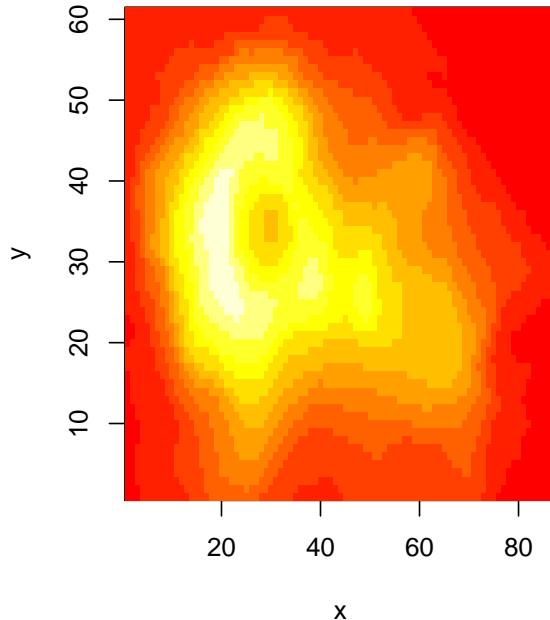
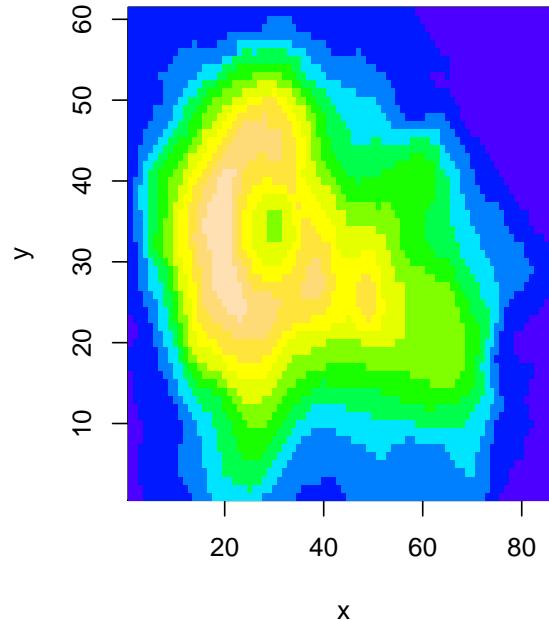


Image funkcija, `topo.colors` skup podataka `volcano`



Slika 76: Funkcija `image()` koja prikazuje sliku u boji; skup podataka `volcano`

## 4.2 Grid grafika/Lattice grafika (temeljena na paketu *grid*)

*Grid* grafika je alternativni sustav grafike koji je naknadno postao sastavni dio sustava R. Velika razlika između *grid* i izvornoga sustava grafike, *base*, jest to što *grid* omogućava izrađivanje višestrukih regija crtanja koje nazivamo pogledi (engl. *viewports*). Koncept pogleda/*viewports* bit će kasnije u tečaju detaljnije obrađen.

### 4.2.1 Paket *grid*

Paket *grid* je postao sastavnim dijelom instalacije R sustava i kao takav nije na raspolaganju kao samostalni paket na CRAN spremištu (<https://cran.r-project.org/>). Paket sadrži kolekciju alata niske razine za crtanje grafičkih prikaza. Paketi više razine kao što su *lattice* i *ggplot2* koriste funkcionalnosti paketa *grid* za svoje sofisticirane (engl. *high-level*) funkcije.

Komentirajte sljedeću liniju koda i ispis na konzoli. Na koji se način pravilno citira paketa *grid*?

```
citation("grid")
```

The 'grid' package is part of R. To cite R in publications use:

R Core Team (2016). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.

A BibTeX entry for LaTeX users is

```
@Manual{,
  title = {R: A Language and Environment for Statistical Computing},
  author = {{R Core Team}},
  organization = {R Foundation for Statistical Computing},
  address = {Vienna, Austria},
  year = {2016},
  url = {https://www.R-project.org/},
}
```

We have invested a lot of time and effort in creating R, please cite it when using it for data analysis. See also 'citation("pkgname")' for citing R packages.

Paket *grid* pruža nekoliko koncepata koji olakšavaju crtanje statističkih prikaza. *Grid* grafički prikaz sastoji se od:

- osnovnih grafičkih objekata (*grob*)
- jedinica, koordinata (*coordinate systems*) za lociranje i veličinu oblika
- grafičkih parametara koji kontroliraju izgled grafičkih objekata (*gpar*)
- pogleda (*viewports*) i rasporeda (*layouts*) za stvaranje lokalnoga konteksta
- ...

U ovom dijelu uvodimo nekoliko funkcija iz *grid* paketa koje su neophodne kako bi se razumio koncept *viewports* i *layouts* (predložaka) u sustavima koji se temelje na gridu kao što su *lattice* ili *ggplot*.

Upoznat ćemo se s najvažnijim funkcijama za određivanje **hijerarhije u grid scenama** i grafičkim objekti (*grob*) paketa *grid*

Temeljni graditeljski blokovi *grid* scene su grafički objekti (*grobs*), koji opisuju oblike koji će se crtati i ulazi/pogledi (engl. *viewports*), koji definiraju podpodručja na stranici kao cjeline za crtanje. Svaki grafički objekt sastoje se od definiranih lokacija i dimenzija u zadanom koordinatnom sustavu. Lokacije i dimenzije oblika su jedinice koje se sastoje od vrijednosti i informacija o koordinatnom

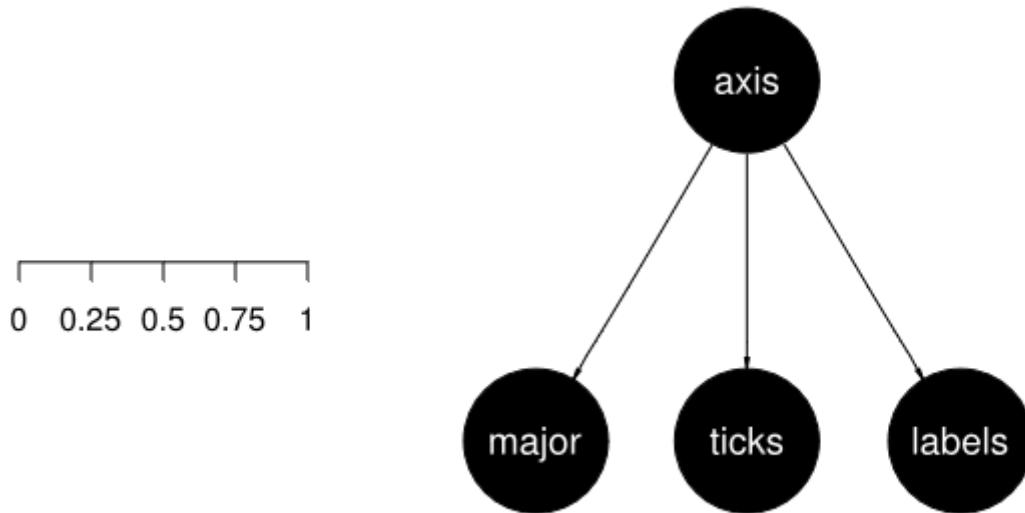
sustavu. Svaki osnovni oblik ima **gp** argument koji omogućava preciziranje grafičkih parametara. Glavni grafički parametri su: boja (**col**); unutarnja popunjenošć (**fill**); širina linije (**lwd**); vrsta linije (**lty**); multiplikator veličine teksta (**cex**).

Ovi graditeljski blokovi se mogu posložiti po hijerarhijama kako bi kreirali *stabla grafičkih objekata* (*gTrees*) and *trees of viewports* (*vpTrees*). Na primjer os na grafu može biti jedan, roditeljski *gTree* koji sadrži nekoliko "djece" *grobs* kako bi se predstavila glavna linija osi, debele oznaće i debele labele osi.

Bilo koji oblik koji je nacrtan pomoću paketa *grid* grafike može imati pridruženo ime. Ako je ime dano, moguće mu je pristupiti, ispitati i modificirati oblik kada je jednom nacrtan. Ove mogućnosti pružaju priliku vrlo detaljne prilagodbe grafa, ali i vrlo općih transformacija grafa koji su nacrtani pomoću paketa koji se temelje na paketu *grid*. Kada se "scena" (pričaz, engl. *scene*) nacrtava pomoću *grid* grafike u R-u, čuva se informacija o svakom obliku koji je korišten da se taj pričaz nacrtava. Ta se evidencija naziva *display list* i sastoji se od liste R objekata, jedne za svaki oblik u sceni.

U slučaju grafike gdje postoji eksplicitan naziv za svaki oblik (engl. *grob*) koji crtamo pomoću grida, omogućavamo pristup na niskoj razini svakom objektu unutar scene. Ovo nam omogućava da napravimo iscrpne prilagodbe scene, bez potrebe za dugim listama argumenata kao iz funkcija crtanja visoke razine te nam je omogućeno da propitujemo i transformiramo scenu kroz širok spektar načina.

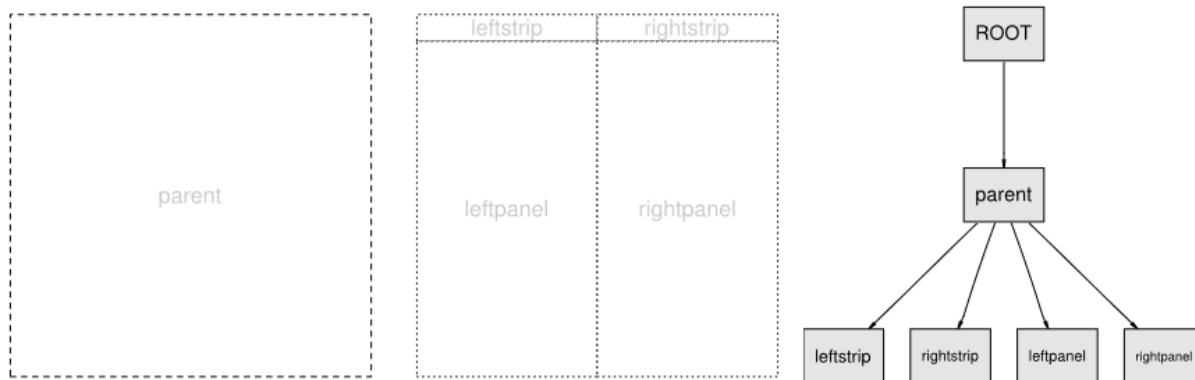
Važna razlika između grafike kako ju definira sustav kao što je R, nasuprot jednostavnih grafičkih paketa kao što su GIMP i/ili Inkscape je u tome što crtanje pomoću paketa *grid* uključuje definiranje **konteksta** za crtanje (engl. *viewports*) i samoga crtanja osnovnih grafičkih objekata (engl. *grobs*) u tim kontekstima (<https://www.stat.auckland.ac.nz/~paul/useR2011-grid/gridHandout.pdf>). Ovdje ćemo spomenuti samo najznačajnije funkcije koje je nužno razumjeti kako bi se što efikasnije iskoristile funkcionalnosti *lattice* i kasnije *ggplot* paketa. Funkcija **grid.ls()** se može koristiti kako bi se prikazala lista grafičkih elemenata *grobs* i *viewports* u sceni. Kod koji slijedi pokazuje što dobijemo nakon što nacrtamo scenu *A* *grob hierarchy*, koja sastoji se od samo roditeljske osi *grob* s ključnom linijom, debele oznaće, debele labele *grobs-ova* "djece".



Slika 77: Hiperhierarhija grafičkih objekata (engl. *grob*). Izvor: <https://www.stat.auckland.ac.nz/~paul/Reports/ggplotSlider/ggplotSlider.html>

*Grid* scena koja se sastoji od višestrukih *viewports*-a koji definiraju regiju za crtanje dijagrama koji sliči trellis-tipu. Jedan roditeljski *viewport* može sadržati kolekciju dijagrama koji unutar sebe imaju nekoliko *viewports* djece.

Postoji mogućnost korištenja nekoliko koordinatnih sustava za definiranje pozicija i veličina *lattice viewporta*. Tako je koordinate moguće zadati putem sljedećih koordinata:



Slika 78: Hjерархија погледа (engl. *viewports*). Izvor: <https://www.stat.auckland.ac.nz/~paul/Reports/ggplotSlider/ggplotSlider.html>

- Koordinate normaliziranog roditelja (engl. **Normalised parent coordinates, NPC**): ishodište *viewporta* je postavljewno na koordinatama (0,0) a cijeli viewport ima širinu i visinu 1
- Fizičke koordinate (engl. **physical coordinates**): absolutne, fizičke koordinate koje su definirane u inčima, centimetrima ili nekim drugim jedinicama
- Prirođene koordinate (engl. **native coordinates**): na osnovi raspona *viewporta* radi se skaliranje x i y osi. Ovo je stvarni koordinatni sustav koji se koristi za crtanje elemenata dijagrama
- Koordinate po osnovi karaktera, linije ili teksta (engl. **character, line or string based coordinates**): koordinatni sustavi su umnošci 1) nominalne veličine fonta, ili 2) vertikalne udaljenosti između baznih linija dvije linije teksta ili širine/visine dijela teksta. Ove koordinate su korisne za definiranje relativne pozicije jednoga elementa grafa u odnosu na druge elemente.

#### 4.2.1.1 Funkcija `viewport()` {grid}

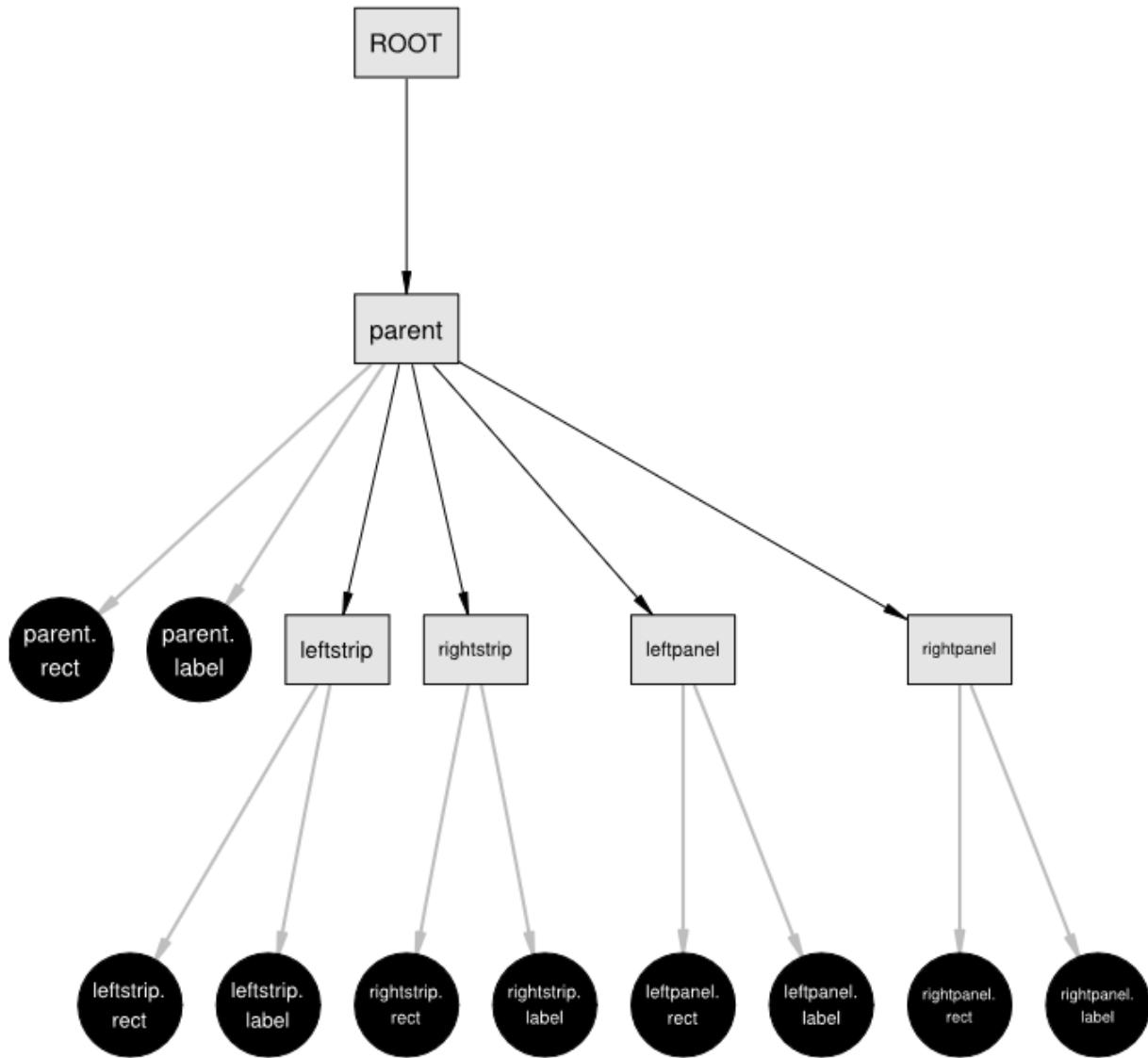
Funkcija `viewport()` stvara pravokutnu regiju na stranici. Pogledi su opisani s lokacijom i veličinom koje se mogu specificirati u bilo kojem opisanom koordinatnom sustavu.

Korištenjem funkcija napravljenih za rad s pogledima (engl. *viewports*) omogućen je prelazak između regija na uređaju (bez potrebe za ponovnim stvaranjem istih). Ovaj mehanizam korisniku pruža mogućnost pristupa, prema potrebi, svim regijama koje su napravljene tijekom crtanja.

Sam proces crtanja u konačnici nastaje pomoću `pushviewport()` funkcije:

#### 4.2.1.2 Funkcija `pushViewport` {grid}

```
#izradivanje viewport-a - kontekst za crtanje
plotvp <- viewport(x=unit(5, "lines"),
                     y=unit(5, "lines"),
                     width=unit(1, "npc") -
                     unit(8, "lines"),
                     height=unit(1, "npc") -
                     unit(8, "lines"),
                     just=c("left", "bottom"),
                     xscale=c(0, 6),
                     yscale=c(0, 6),
                     name="plotRegion")
```



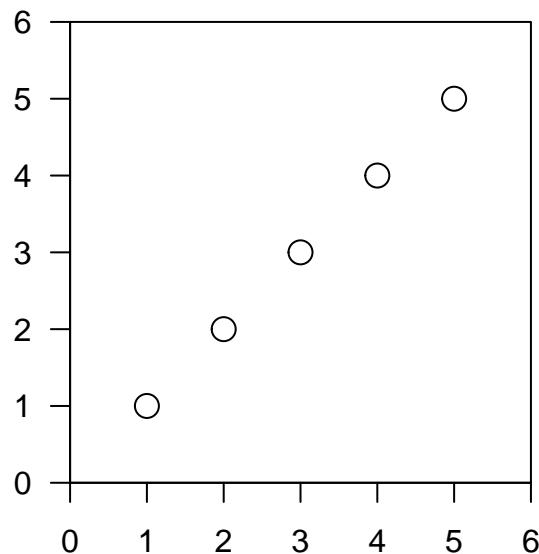
Slika 79: Dijagram kombinirane hijerarhije *vpTrees* i *gTrees* unutar *grid* scene koja se sastoji od roditeljskog *viewport*-a i četiri *viewport*-a "djece" (engl. *child*), s pravokutnikom i tekstualnim grafičkim objektom (engl. *grob*) koji je nacrtan unutar svih pogleda (engl. *viewport*). Izvor: <https://www.stat.auckland.ac.nz/~paul/Reports/ggplotSlider/ggplotSlider.html>

```
#izrađivanje konteksta za graf
pushViewport(plotvp)

#pravokutnik grafa
grid.rect(x=0.5, y=0.5, width=1, height=1)

#točke grafa
grid.points(1:5, 1:5, default.units="native")

#osи grafa
grid.xaxis(at=0:6)
grid.yaxis(at=0:6)
```



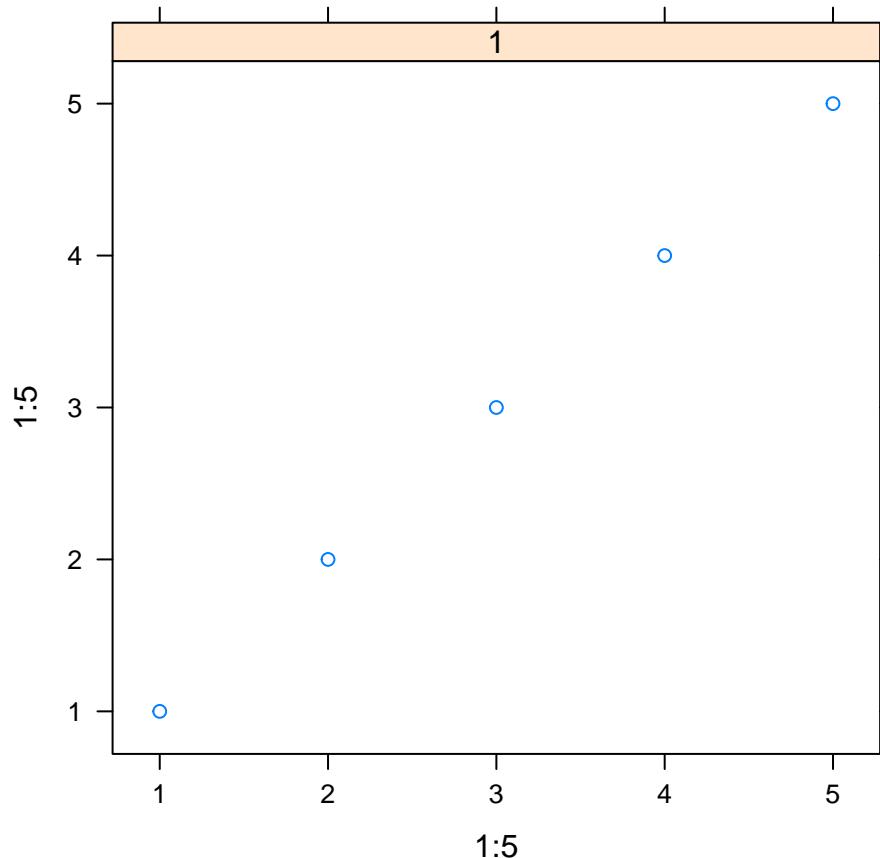
Slika 80: Izrađivanje konteksta (*viewport*) za crtanje objekata (*grobs*)

```
#lista viewports-a/grobs-a
grid.ls(fullNames=TRUE)

rect[GRID.rect.14]
points[GRID.points.15]
xaxis[GRID.xaxis.16]
  lines[major]
  segments[ticks]
  text[labels]
yaxis[GRID.yaxis.17]
  lines[major]
  segments[ticks]
  text[labels]
```

Kako je pokazano, veoma je važno shvatiti koncept pogleda i grafičkih objekata (*viewports* i *grob*). Na ove koncepte oslanjaju se sve *lattice* funkcije visoke razine rade; prvo stvore pogled/*viewports*, a onda crtaju grafičke objekte/*grob* u stvorenim pogledima. Usporedite sljedeći primjer koji je dobiven funkcijom **xyplot()**, funkcijom visoke razine iz paketa *lattice*.

```
#funkcija visoke razine iz *lattice* grafike  
xyplot(1:5 ~ 1:5 | 1)
```



Slika 81: Funkcija visoke razine temeljena na\* gridu\* iz paketa *lattice*

```
barchart(yield ~ variety | site, data = barley,
          groups = year, layout = c(1,6), stack = TRUE,
          auto.key = list(space = "right"),
          ylab = "Urod ječma",
          scales = list(x = list(rot = 45)))

grid.ls(viewports=T, grob=F, fullNames=F)
```

ROOT

Kako je ranije rečeno, upravljanje i uporaba ove funkcije izvan su djelokruga ovog tečaja, ali postoji potreba za razumijevanje koncepta *grid* grafike kako bi se mogla istražiti i koristiti većina grafičkih mogućnosti grafičkih paketa koji se temelje na *grid* grafici u budućnosti. Čak i ako još nismo upoznati sa *lattice* grafikom, pružamo primjer upravljanja pogledima/viewportima od strane korisnika. U primjeru koji slijedi nacrtat ćemo pravokutnik u veličini željenog pogleda te ga obojati prema željenoj boji. Najprije pogledajmo jedan grafički prikaz napravljen funkcijom visoke rezolucije paketa *lattice*.

```
#crtanje podataka o urodu ječma (engl. barley), funkcija viso
barchart(yield ~ variety | site, data = barley,
          groups = year, layout = c(1,6), stack = TRUE,
          auto.key = list(space = "right"),
          ylab = "Urod ječma",
          scales = list(x = list(rot = 45)))

#specificiramo dio grafa na koji želimo nešto dodati
downViewport("plot_01.panel.1.3.vp")

#downViewport("plot_01.panel.1.4.vp")

#bojamo željeni dio grafa
grid.rect(gp=gpar(col=NA, fill=rgb(0,.1,0,.4, 0.6)))
```

#### 4.2.1.3 Funkcija current.viewport() {grid}

. Funkcijom **current.viewport()** daje se informacija o trenutačnom pogledu.

#### 4.2.1.4 Funkcija current\_vpTree() {grid}

Poglede je moguće specificirati i odrediti im veličinu i korištenjem funkcije **layout()**, što je bolja opcija od davanja explicitnih koordinata veličine i položaja.

#### 4.2.1.5 Funkcija downViewport() {grid}

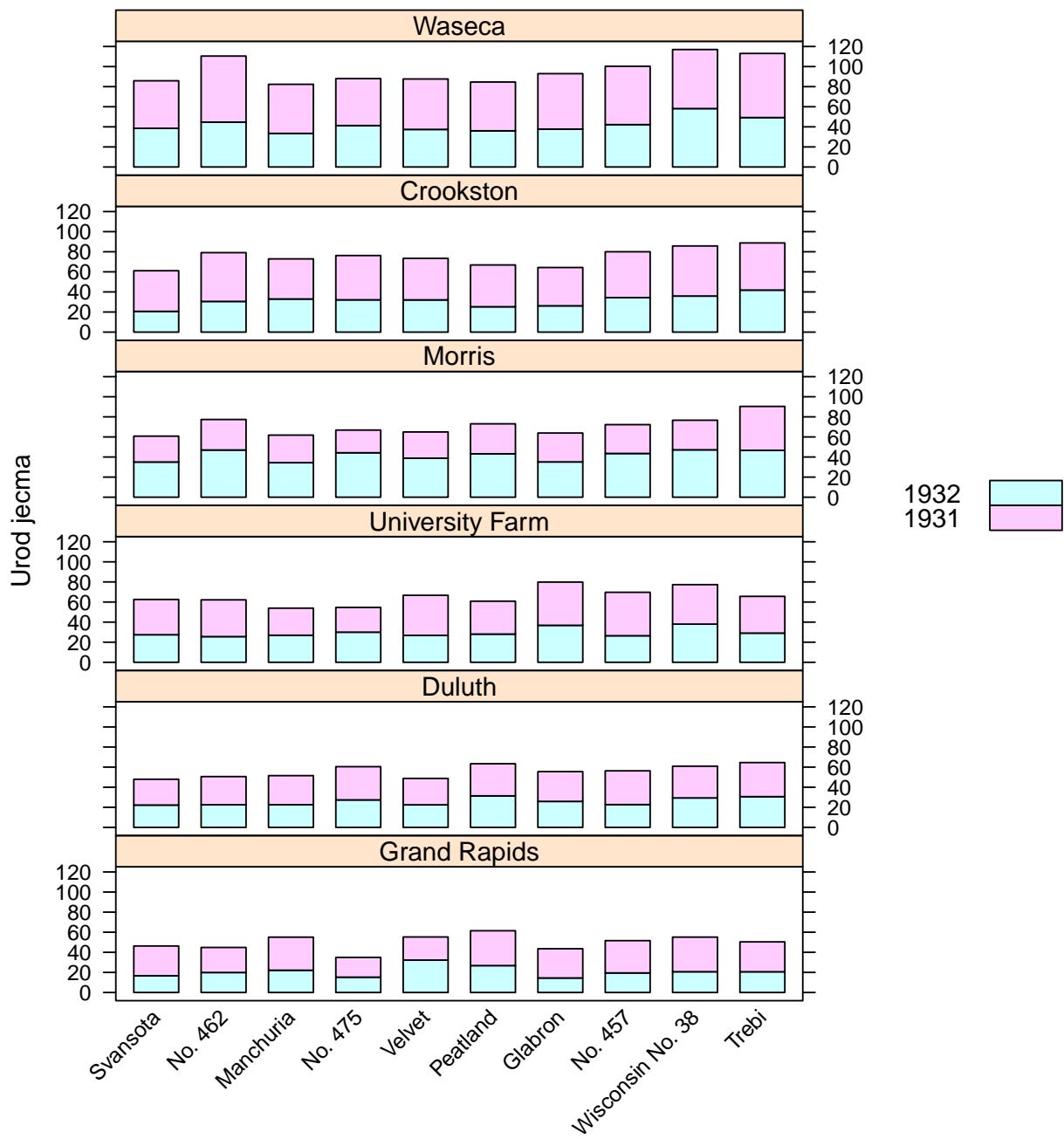
Funkcija se koristi za povratak na neki od postojećih pogleda na sceni.

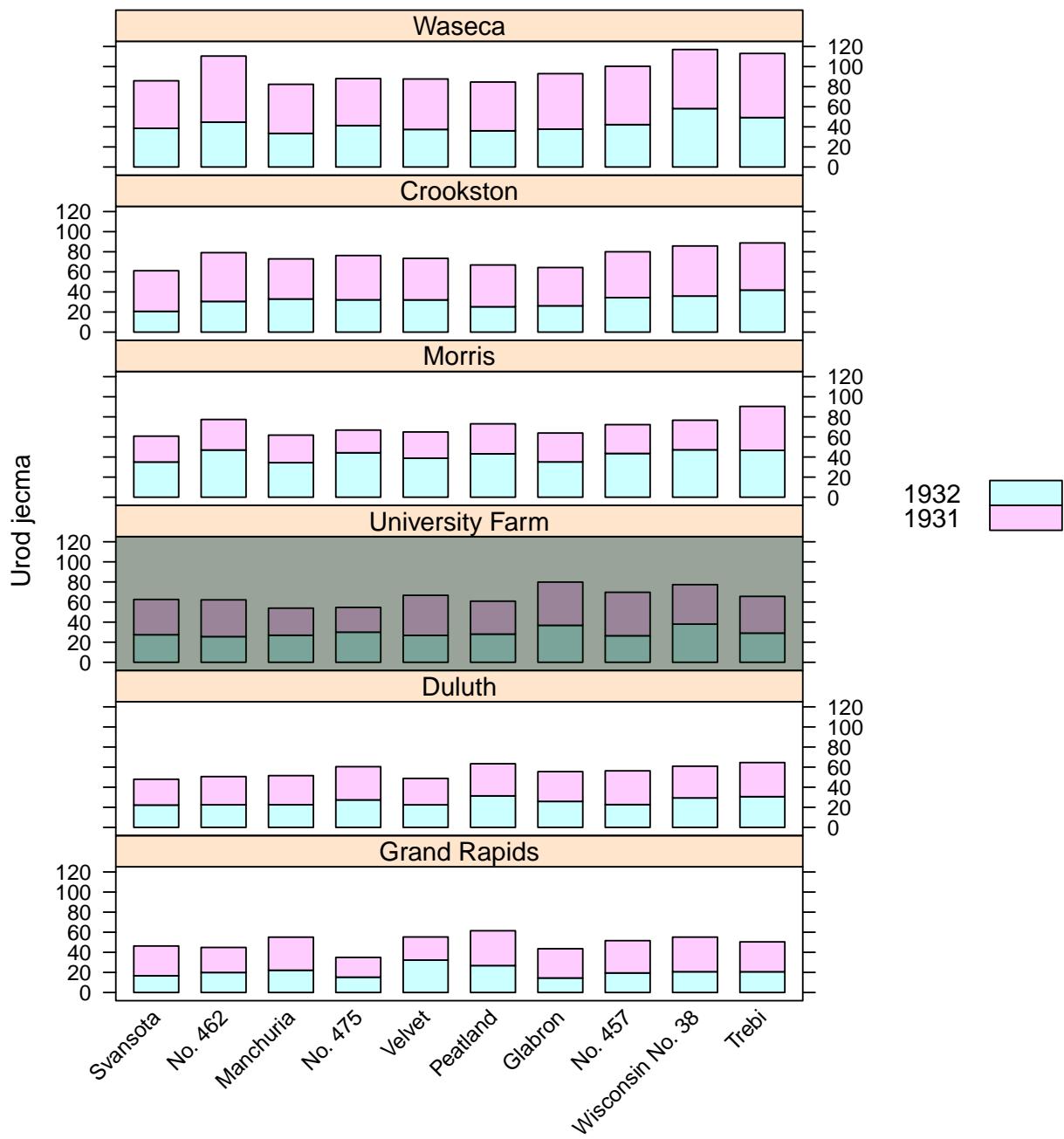
#### 4.2.1.6 Funkcija grid.layout {grid}

Funkcijom **layout()** dijelimo željeni pogled na retke i stupce. Visina i širina pojedinoga retka i/ili stupca može se zadati u bilo kojem opisanom koordinatnom sustavu uz dodatak specijalnog koordinatnog sustava, tzv. "null" koji postoji jedino u sustavima *layouta*.

Za sve koji žele znati nešto više ovoj temi mogu svoje znanje o upravljenjima pogledima u paketima temeljenim na *grid* paketu potražiti na <https://stat.ethz.ch/R-manual/R-devel/library/grid/doc/viewports.pdf> i službenoj dokumentaciji paketa.

I na kraju napomena, imenovati se može sve što crtamo pomoću neke funkcije temeljene na *grid* paketu. Način imenovanja elemenata scene bio bi primjerice: sve što se nacrtati unutar regije panela ima riječ "panel" u nazivu, zajedno sa sufiksom u obliku *i.j* kako bi se identificirao redak i stupac panela. Jednako vrijedi i za sve ostale klase grafičkih objekata koji se pojavljuju. Nemojte

Slika 82: Funkcija visoke razine temeljena na *gridu* iz paketa *lattice*

Slika 83: Primjer upravljenjima pogledima u funkciji visoke razine trellis u paketu *lattice*

se preplašiti dugačkih ispisa strukture pojedinog grafa. Pažljivim čitanjem vidjet ćete da su nazivi veoma intuitivni te ćete se već nakon nekoliko primjera jednostavno snalaziti s popisima elemenata. Postoje i alati koji pomažu u prikazu imena objekata na postojećoj sceni od kojih se veliki broj nalazi u paketu *gridDebug*.

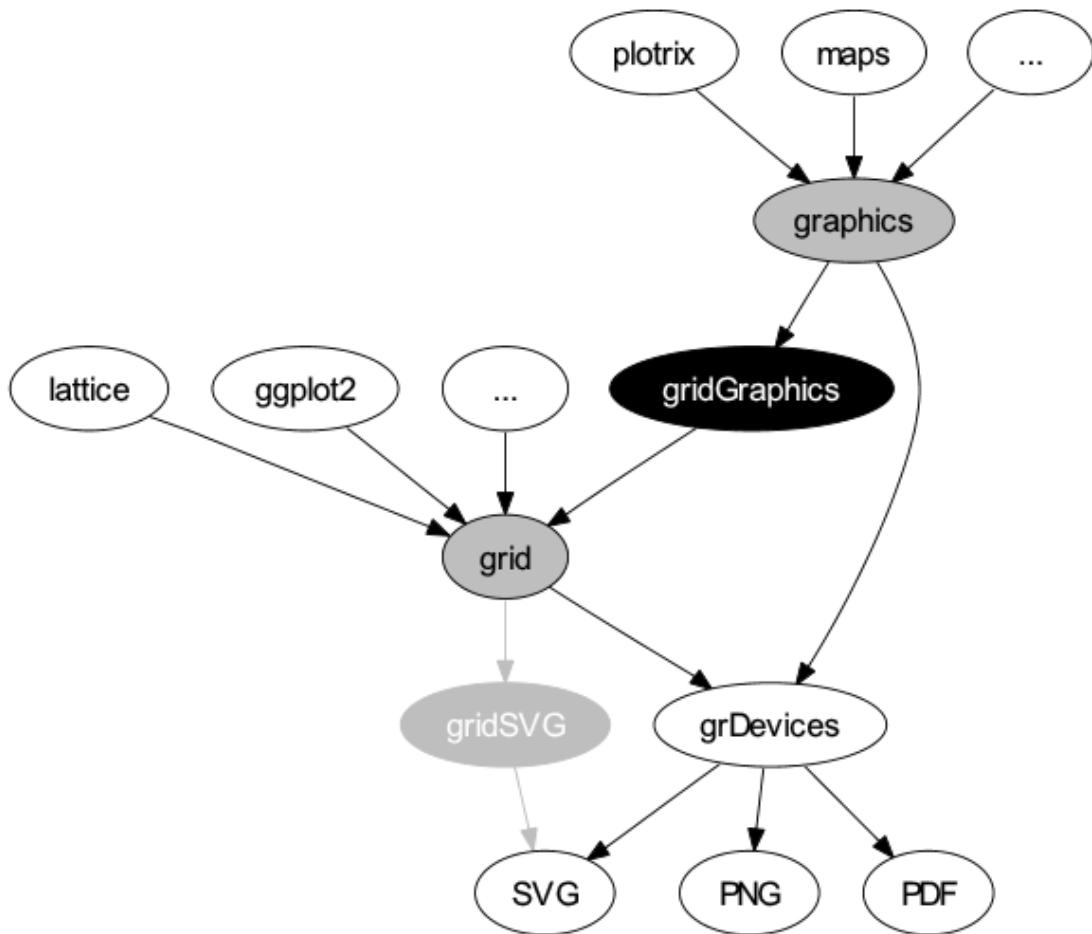
Drugi, veoma popularan paket s kojim ćemo se upoznati na tečaju S750 Grafički sustavi u R-u 2, je *ggplot2* paket. Shema imenovanja elemenata grafičkoga prikaza u paketu *ggplot2* je donekle drugačija: graf se stvara iz objekta klase *gTrees*, ne iz jednostavnih grafičkih objekata (*grobova*). Ova razlika proizlazi iz činjenice što *gTrees* već odražava strukturu cijelog grafa. Imena pojedinih elemenata mogu biti dana kako bi reflektirala njihovu *lokalnu* ulogu u pojedinoj komponenti grafa.

U uvodu smo napomenuli da miješanje funkcija triju grafičkih sustava u R-u (*base*, *lattice* i *ggplot*) nije jednostavno. Ipak, posljednjih se godina intenzivno razvijaju paketi koji omogućavaju interoperabilnost, kao između R grafičkih sustava tako i između R-a i grafičkih biblioteka pisanih u drugim programskim jezicima, primjerice Javi. S takvim ćemo se paketima upoznati u tečaju S750 Grafički sustavi u R-u 2. Ovdje samo spominjemo paket za kombiniranje funkcionalnosti paketa *grid* s *base* grafikom. Paket omogućava kombinaciju definiranja *viewport*-ova s klasičnim, *base* sustavom.

Poznavanje svega i razvoj paketa *gridGraphics* koji omogućava mješavinu dva sustava:

```
#učitavanje dodatne biblioteke
library(gridGraphics)

#izrađivanje funkcije
pf <- function() plot(mpg ~ disp, mtcars, pch=16)
pushViewport(viewport(x=0, y=0, width=2/3, height=2/3,
                      just=c("right", "bottom")))
#graf
grid.echo(pf, newpage=FALSE)
```



Slika 84: Novi pristupi: kombiniranje sustava osnovne i *grid* grafike. Izvor: <https://www.stat.auckland.ac.nz/~paul/Talks/useR2015/>

Takozvana *lattice* grafika je izgrađena na osnovi koncepta generičkoga područja crtanja koji nazivamo pogled (engl. *viewport*). Već smo se upoznali s konceptom pogleda prilikom upoznavanja s paketom *grid*. *Lattice viewport* je pravokutna regija koju određuje korisnik. Korisnik tako može odrediti lokaciju, margine/poravnjanje i veličinu pojedinačnog područja crtanja/pogleda. Tako jedan pogled/regija/*viewport* može biti cijeli graf, nekoliko grafova, samo jedna margina grafa ili čak pojedinačni simbol.

#### 4.2.2 Paket(i) *lattice/latticeExtra*

*Lattice* je dodatak na okruženje za statističko računanje u R-u a koje omogućava skup grafika na razini korisnika i koje je alternativa osnovnim funkcijama R-a. Radi se o implementiranju *Trellis plots* sustava S putem paketa *lattice* koji je pripremio Deepayan Sarkar. Grafika *lattice* je neovisna od osnovnog, *base* sustava grafike u R-u.

Paket *lattice* nazvan je tako zato što može generirati \*\*Trellis\* prikaze. *Trellis* prikaz označava grafove na više panela koji su pogodni za ilustriranje multivarijatnih podataka. Ova načela su očita i u broju novih vrsta dijagrama u *Trellisu* i kroz zadani odabir boja, oblika simbola, stilova linija koje omogućavaju *Trellis* dijagrami. Nadalje, *Trellis* dijagrami omogućuju karakteristiku poznatu kao uvjetovanje na višestrukim panelima (engl. *multipanel conditioning*), čime se kreiraju višestruki grafički prikazi podjelom podataka koji se crtaju sukladno razinama zadanih varijabli. Svaki panel grafički prikazuje podskup podataka. Svi paneli u *Trellis* prikazu sadrže istu vrstu grafa. Podskupovi podataka se biraju na uobičajen način, uvjetovanjem u pogledu uvjeta ili diskretnih varijabli u podacima te, stoga, predstavljaju koordiniranu seriju pregleda višedimenzionalnih podataka.

Paket je konstruiran povrh *grid* paketa (*grid* je tzv. *dependency* za *lattice*) koji je pokretački stroj koji funkcioniра u podlozi. Iz tog razloga je naslijedio i mnoga njegova svojstva. Budući da *lattice* grafika u R-u koristi *grid* grafiku u pozadini za stvaranje statističkih grafika visoke razine on je nezavisan od tradicionalnih *base* funkcija. Dva sustava su u većini slučajeva nekompatibilni ali kao što smo već rekli, noviji paketi (*gridgraphics*) brišu granicu između ova dva sustava.

Kao i kod drugih paketa, potrebno je odati priznanje pravilnim citiranjem osoba koje su uložile svoje znanje i vrijeme izradu funkcija paketa. U slučaju *lattice* i *latticeExtra* paketa to je:

```
citation("lattice")
```

To cite the *lattice* package in publications use:

Sarkar, Deepayan (2008) *Lattice: Multivariate Data Visualization with R*. Springer, New York. ISBN 978-0-387-75968-5

A BibTeX entry for LaTeX users is

```
@Book{,
  title = {Lattice: Multivariate Data Visualization with R},
  author = {Deepayan Sarkar},
  publisher = {Springer},
  address = {New York},
  year = {2008},
  note = {ISBN 978-0-387-75968-5},
  url = {http://lmdvr.r-forge.r-project.org},
}
citation("latticeExtra")
```

To cite package 'latticeExtra' in publications use:

Deepayan Sarkar and Felix Andrews (2016). *latticeExtra: Extra Graphical Utilities Based on Lattice*. R package version 0.6-28.  
<https://CRAN.R-project.org/package=latticeExtra>

A BibTeX entry for LaTeX users is

```
@Manual{,
  title = {latticeExtra: Extra Graphical Utilities Based on Lattice},
  author = {Deepayan Sarkar and Felix Andrews},
  year = {2016},
  note = {R package version 0.6-28},
  url = {https://CRAN.R-project.org/package=latticeExtra},
}
```

ATTENTION: This citation information has been auto-generated from the package DESCRIPTION file and may need manual editing, see 'help("citation")'.

Unaprjeđenjem korisničke kontrole nad grafikom, *Trellis* softver čini i da se grafičke funkcije ponašaju kao i bilo koje druge funkcije. Rezultat provođenja *Trellis* naredbe je objekt klase *Trellis*. Osim ako mu nije dodijeljeno ime ili se koristi u dalnjem izračunu, *Trellis* objekt se prikazuje (to je zadana opcija u R sustavu za sve klase objekata).

Kako je spomenuto, paket pod nazivom *lattice* pruža funkcije Trellis grafike. Kada se koristi *lattice* značajno je navesti uređaj koji koristimo **trellis.device()** prije davanja grafičkih naredbi.

**Dobre strane lattice** grafičkoga sustava su:

- manje potrebe za provođenjem detaljne kontrole
- izvrstan za izradu kohezivnih skupina dijagrama.

**Loše strane lattice**:

- mnogima se sustav čini pretežak i/ili nepoznat kako bi proveli detaljnu kontrolu.

Problem proizlazi iz dizajna sustava koji stvara sve što je potrebno za dijagram u jednoj naredbi korštenjem *grid* paket funkcionalnosti koji su za većinu korisnika neka vrsta "crne kutije". Ipak, glavni razlog zašto ljudi počnu koristiti *lattice/trellis* grafiku je spomenuta funkcionalnost kombiniranja većega broja panela na razuman način. Uvjetovanje na više panela je, doista, veliki napredak u odnosu na osnovnu R grafiku.

#### 4.2.3 Viewports/layouts (predlošci) u *lattice*-u

Već smo se upoznali s konceptom *viewports*, *layouts* i *grobs*. U *lattice* grafici moguće je definirati predloške (*layouts*) za specifične poglede (*viewports*) - podjelu roditeljskoga (engl. *parent*) *viewport*-a na retke po stupcima **različite** veličine. Neki podskup polja u predlošcima može se specifirati kao djeca. Dodatno, *viewport*-i se mogu smještati jedan unutar drugoga, te se tako kreira hijerarhijski dizajn. Ovaj sustav slaganja grafičkih objekata (strana - dijagram - područja dijagrama) po hijerarhijskom redu je od velike pomoći u statističkoj grafici, s obzirom na to da se statističari koji rade s podacima često trude vizualizirati podatke koji sadrže urođenu hijerarhiju. *Viewport*-i koje *lattice* kreira su dostupni korisniku preko **trellis.focus()** funkcije. Funkcije iz *grid* paketa se, također, mogu izravno koristiti.

Budući da je ovaj tečaj određena vrsta uvoda, pokušat ćemo pružiti pregled neophodnih funkcija, postavki i koncepta kako bi se počeo rad s paketom *lattice*, ali paket nudi znatno više mogućnosti za prilagodbu grafičkih prikaza. Svatko tko se želi više informirati može to učiniti putem velikoga broja knjiga, tutorijala, primjera i vježbi koje su dostupne putem interneta.

#### 4.2.4 Lattice uređaji

Paket *lattice* zadržava svoj vlastiti skup postavki grafičkih parametara koji kontrolira izgled grafa na svakom grafičkom uređaju. To su parametri poput boje linija, korištenih paleta, fonta teksta, veličine i još dosta toga.

Zadane postavke za uređaj ovise o vrsti uređaja koji se otvara (npr. PostScript, PDF, JPG, PNG ili slično). Kod jednostavnih uporabe to ne uzrokuje probleme zato što *lattice* automatski inicira ove postavke prvi put kada se producira izlazni rezultat iz

*lattice* sustava na uređaju. Ako je potrebno kontrolirati inicijalne vrijednosti ovih postavki, funkcija **trellis.device()** se koristi kako bi se uređaj otvorio s određenim postavkama *lattice* grafičkih parametara.

### **trellis.device()**

#### 4.2.5 Lattice plot prikazi

*Trellis* grafika iz *lattice* paketa ima brojne funkcije visoke razine za prikaz različitih vrsta podataka. Slijedi podjela vezano za broj varijabli koje se prikazuju na prikazu. Kako je načelo podskupova podataka ili uvjetovanoga prikaza više panela nešto što na isti način funkcionira u svim prikazima *lattice* pokazat ćemo koncept samo na odabranom prikazu. Zbog spomenutoga, za neke prikaze nećemo ni davati primjere.

Vrste prikaza dostupnih u paketu *lattice* su:

##### 1) Za jednu varijablu - univarijatni

- **barchart()** - bar dijagram
- **bwplot()** - komparativni box-i-whisker dijagrami
- **densityplot()** - kernel dijagram gustoće
- **dotplot()** - Cleveland točkasti dijagram
- **histogram()** - histogram
- **piechart()** - pita dijagram (statističari **NE** preporučaju uopće ovaj prikaz!)
- **qqmath()** - teoretski dijagram kvantila
- **stripplot()** - strip dijagramchart (usporedni 1-D dijagrami raspršenja)

##### 2) Dvije varijable, bivarijatni - *bivariate*

- **qq()** - kvantilni dijagram dva uzorka
- **timeplot()** - dijagram vremenskih nizova
- **xyplot()** - dijagram raspršenja

##### 3) Tri varijable, trivarijatni - *trivariate*

- **contourplot()** - konturni dijagram površina
- **levelplot()** - dijagrami razina
- **splom()** - dijagram raspršivanja matrice

##### 4) 3\_D prikazi

- **wireframe()** - trodimenzionalni dijagram perspektive površina
- **cloud()** - trodimenzionalni dijagram raspršenja
- **parallel()** - dijagram paralelnih koordinata.

Tijekom ovog tečaja upoznat ćemo se samo s nekim od spomenutih prikaza. Dodatne funkcije visoke razine (prikazi) pripremljene su i dostupne kroz paket *latticeExtra*. Za dodatne informacije pogledati: (<https://cran.r-project.org/web/packages/latticeExtra/latticeExtra.pdf>).

#### 4.2.6 Opće karakteristike prikaza u *latticeu*

Svaki spomenuti tip prikaza u *latticeu* povezan je s odgovarajućom funkcijom visoke razine (poput histograma). Prikaz se sastoji od različitih elemenata koji se koordiniraju sa zadanim parametrima kako bi pružili smislene rezultate, ipak, svaki element korisnik može kontrolirati neovisno od drugih. Ključni elementi prikaza su:

- **primarni prikaz** - panel
- **oznaka osi**
- **oznaka trake** (procesa uvjetovanja)
- **legende** (proces grupiranja).

Svaka vrsta prikaza povezuje se s odgovarajućom funkcijom visoke razine (histogram, dijagram gustoće itd). Dodatno, trellis prikazi su definirani ulogama varijabli:

- **primarne varijable**: one koje definiraju primarni prikaz
- **uvjetujuće varijable**: dijele podatke na podskupine, od kojih se svaka prikazuje u različitim panelima
- **grupirajuće varijable**: podskupine su suprotstavljene unutar panela tako što su postavljene na odgovarajućim prikazima.

Za kontrolu i prilagodbu stvarnoga prikaza u svakom panelu, stranica pomoći odgovarajuće zadane funkcije panela često će biti informativnija. Posebno, stranice pomoći opisuju brojne argumente koji se, općenito, koriste kada se poziva odgovarajuća funkcija visoke razine ali koje su specifične za te konkretnе panele.

U svakom slučaju, dodatni argumenti pozivanja funkcija visoke razine mogu se koristiti kako bi se aktivirali opći varijeteti, dozvoljena je potpuna fleksibilnost putem proizvoljnih funkcija koje definira korisnik. Ovo je posebno korisno za kontroliranje primarnoga prikaza putem panel funkcija (<http://lattice.r-forge.r-project.org/Vignettes/src/lattice-intro/lattice-intro.pdf>). Jedna od funkcija koje su dostupne u *latticeExtra*, a koja je od velike pomoći u statističkoj vizualizaciji je funkcija za izrađivanje *trellis* prikaza za funkciju empirijske kumulativne distribucije **ecdfplot()**.

*Lattice* prikaz se sastoji od različitih elemenata koji su koordinirani različitim parametrima kako bi dali smislene rezultate. Ključni elementi *lattice* prikaza su:

- **primarni prikaz** - panel
- **oznaka osi**
- **oznaka trake** (opisuje proces uvjetovanja)
- **legende** (tipično opisuju proces grupiranja).

Korisnik može kontrolirati svaki element kao je to potrebno. Postoji mogućnost pružanja dodatnih argumenata pozivanjima funkcija visoke razine kako bi se aktivirale zadane vrijednosti ili putem kreiranja funkcija koje korisnik proizvoljno definira.

Trellis prikazi su definirani vrstom grafike i ulogom koju različite varijabe u njoj igraju. Svaka vrsta prikaza povezana je s odgovarajućom funkcijom visoke razine (histogram, dijagram gustoće itd.). Moguće uloge ovise o vrsti prikaza ali, su tipične:

- **primarne varijable**: one koje definiraju primarni prikaz
- **uvjetujuće varijable**: dijele podatke na podskupine, svaka od njih predstavlja različite panele
- **grupirajuće varijable**: podskupine se porede unutar panela tako da se prikazuju u odgovarajućim prikazima.

Sjetimo se od ranije da su dva ključna cilja statističke grafike:

- 1) olakšati usporedbe
- 2) identificirati trendove.

Mnogi smatraju *lattice* grafiku boljom od tradicionalne u postizanju spomenutih ciljeva.

**graph\_type(formula, data=)**

gdje je graph\_type odabran iz prethodno navedenih. Formula precizira varijablu(e) koje će se prikazati i bilo koje uvjetujuće varijable. Na primjer:

- $\sim x|A$  - prikaz numeričke varijable x za svaku razinu faktora A
- $^{**}y\sim x | A*B^{**}$  - prikaz odnosa između numeričkih varijabli y i x odvojeno za svaku kombinaciju razine faktora A i B
- $\sim x$  znači prikaz same numeričke varijable x.

Dodatno ćemo objasniti ulogu uvjetujućih varijabli i dodatnih grafičkih prikaza u različitim funkcijama visoke razine. Način kreiranja višestrukih panela i ostale postavke panela su slični su u svim drugim prikazima u paketu.

Nakon preciziranja formule za lattice prikaz, moguće je dati dodatne argumente poput podataka, podskupa i mnogo drugoga. Prije korištenja bilo koje od funkcija za izrađivanje prikaza, upoznajte se s funkcijom kroz njenu informacijsku karticu.

Lattice, funkcije visoke razine, ne crtaju graf, one samo kreiraju objekt tipa trellis. Taj objekt sadrži sve elemente grafa. Kada pozovemo funkciju trellis implicitno dajemo naziv objekta R-u. Zapamtite da je zadano ponašanje R-a da pokuša nacrtati objekt. Alternativno, možemo precizno dodijeliti rezultat trellis funkcije određenom objektu poput g1 <- histogram(~x).

#### 4.2.7 Grafički parametri za *Trellis* prikaze

Postoji mnogo mogućnosti u *lattice* paketu za prilagodbu grafa kroz pripremljene funkcije postavki, ali i kroz mogućnost prilagodbe kroz funkcije koje definira korisnik.

Različiti grafički parametri (boja, vrsta linije, podloga itd.) koji kontroliraju izgled trellis prikaza, vrlo su podložni prilagodbi ali to nije uvijek lako za korisnika. Također, R može proizvesti grafove na nizu uređaja i očekuje se da različiti skupovi parametara bolje odgovaraju nekim različitim uređajima. Ovi se parametri pohranjuju interno u varijabli koja se naziva *lattice.theme*, koja predstavlja listu onih komponenti koje definiraju postavke za određene uređaje. Komponente se identificiraju nazivom uređaja koji predstavljaju. Ako nismo sigurni koji nam je zadani grafički uređaj možemo unijeti **.Device** u R konzolu da dobijemo odgovor.

Inicijalne postavke svakoga uređaja imaju zadane vrijednosti koje odgovaraju tom uređaju. Kada je uređaj jednom otvoren, nje-gove će se postavke mijenjati. Kada se drugom prilikom isti uređaj otvoriti kasnije uporabom *trellis.device*, postavke za taj uređaj će se resetirati na polazne, osim ako nije drugačije precizirano u pozivanju *trellis.device*. Postavke za različite uređaje se tretiraju odvojeno tako da otvaranje PDF uređaja sa zadanim postavkama neće promijeniti postavke postScript-a, koje će ostati kad god je uređaj aktivan.

Ako želimo sačuvati svoj dijagram *lattice/trellis* na određenom uređaju, na primjer PNG, proces je isti kao i ranije opisani za R *osnovnu* grafiku: 1) inicirate (otvorite) željeni grafički uređaj; 2) crtajte u uređaju i 3) zatvorite uređaj.

#### 4.2.8 Kontroliranje grafičkih parametara u grafici temeljenoj na *grid/lattice*

U ovom dijelu uvodimo najznačajnije parametre za kontroliranje izgleda grafike koja se priprema pomoću funkcija iz *lattice* paketa.

##### 4.2.8.1 Funkcija `show.settings()` {*lattice*}

Mnogi aspekti *lattice* grafike su određeni trenutačnom temom. Tema je velika lista grafičkih parametara koji pružaju detaljnu kontrolu grafike u *lattice*-u. Mnogi od naziva se sami objašnjavaju, posebno kada se vide uz izlazni rezultat **show.settings()**. Kako biste dobili vizualni pregled postavki, unesite:

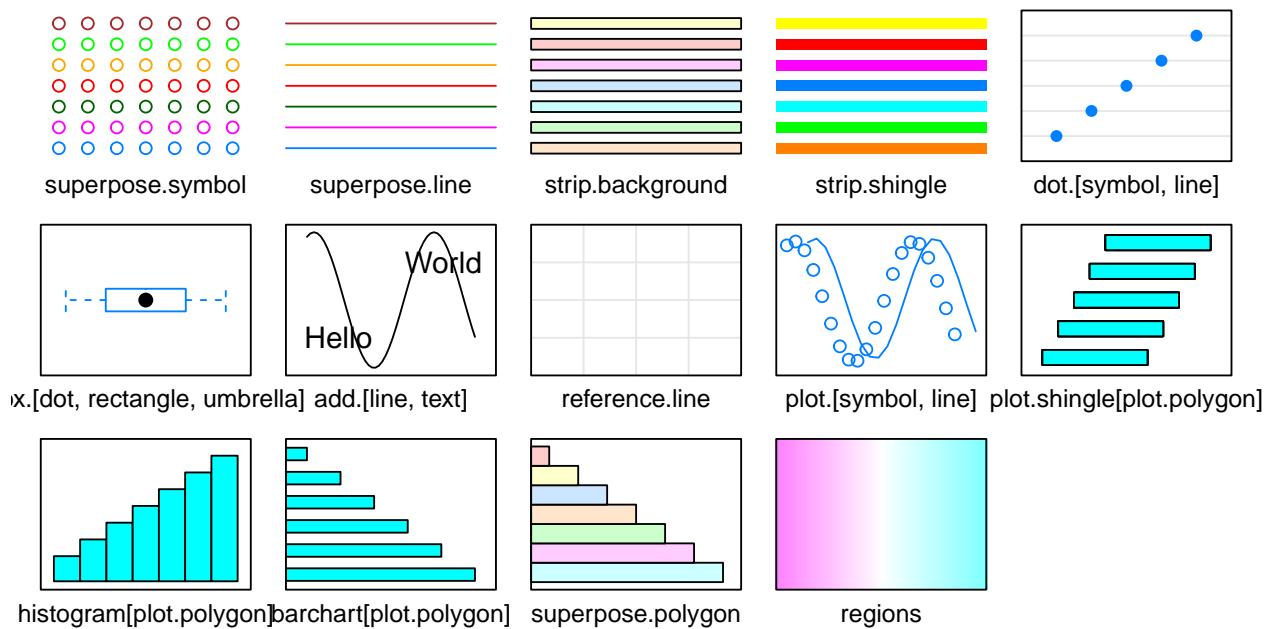
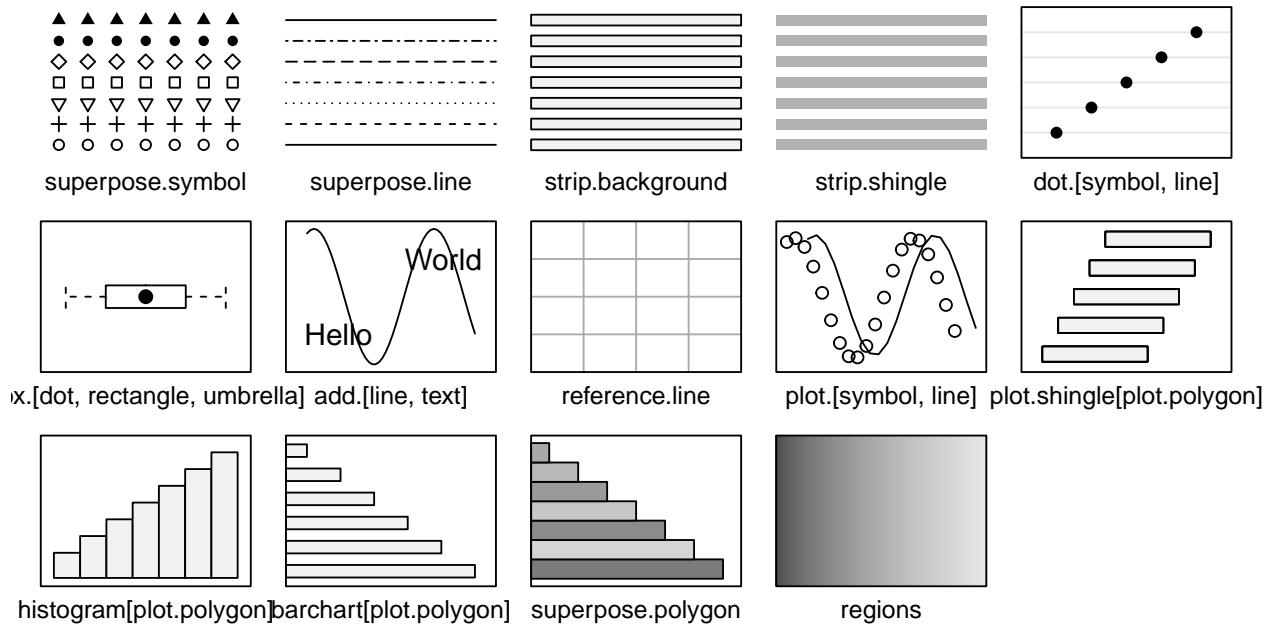
```
show.settings()
```

Ako želimo promijeniti grafičke postavke otvorenog uređaja, jedna od opcija koju imamo je sljedeća:

```
trellis.device('png', color=FALSE)
```

Vizualiziranje rezultata promjena:

```
show.settings()
```

Slika 85: Trenutačno važeće postavke za *Trellis* uređajSlika 86: Postavljanje postavki *Trellis* uređaja na crno-bijelu opciju

Komentirajte rezultat!

#### 4.2.8.2 Functions `trellis.par.get()` / `trellis.par.set(){lattice}`

Ove dvije funkcije omogućavaju postavku grafičkih parametara dijagrama *lattice* (*trellis*). Na neki način, `trellis.par.get` i `trellis.par.set` su zajedno zamjena za `par()` funkciju iz tradicionalne R grafike. Posebno, promjena par postavki ima malo (ako imalo) utjecaja na *lattice* ispis. Budući da se lattice dijagrami implementiraju pomoću *grid* grafike, sustav njegovih parametara nemaju utjecaja osim ako ih ne "pregazi" odgovarajuća postavka *lattice* parametra. Ovi parametri mogu se specificirati dijelom kao *lattice* tema (engl. *lattice theme*) unutar *grid.pars* komponente. Za detaljnije informacije pogledati `gpar()` listu validnih imena grafičkih parametara.

Sve elemente lattice grafa kontroliraju *trellis* postavke. Korisnicima rijetko treba da se izravno dotiču i mijenjaju *trellis* postavke, jer do većine parametara mogu doći i izravno i promjeniti ih putem opće funkcije prikaza (engl. *general display function*) ili panel funkcije. Određeni parametri (npr., "brkovi" u box dijagramu) mogu se promjeniti samo kroz *Trellis* postavke.

Za dodatne informacije o funkcijama unesite:

```
?trellis.par.get
```

#### 4.2.8.3 Funkcija `trellis.par.get()`

Kada se funkcija poziva bez ikakvih argumenata, ispis je potpuni popis postavki za aktivni uređaj. Kada je prisutan argument imena, povratno će ispisati samo tu komponentu. Kako biste vidjeli detalje Vaše trenutačne teme, koristite `trellis.par.get()` funkciju.

```
str(trellis.par.get(), max.level = 1)
```

```
List of 35
$ grid.pars      : list()
$ fontsize       :List of 2
$ background     :List of 2
$ panel.background :List of 1
$ clip           :List of 2
$ add.line        :List of 4
$ add.text        :List of 5
$ plot.polygon   :List of 5
$ box.dot         :List of 5
$ box.rectangle  :List of 5
$ box.umbrella   :List of 4
$ dot.line        :List of 4
$ dot.symbol      :List of 5
$ plot.line       :List of 4
$ plot.symbol     :List of 6
$ reference.line :List of 4
$ strip.background :List of 2
$ strip.shingle   :List of 2
$ strip.border    :List of 4
$ superpose.line  :List of 4
$ superpose.symbol :List of 6
$ superpose.polygon:List of 5
$ regions          :List of 2
$ shade.colors    :List of 2
$ axis.line        :List of 4
$ axis.text        :List of 5
$ axis.components  :List of 4
$ layout.heights  :List of 19
$ layout.widths   :List of 15
```

```
$ box.3d           :List of 4
$ par.xlab.text   :List of 5
$ par.ylab.text   :List of 5
$ par.zlab.text   :List of 5
$ par.main.text   :List of 5
$ par.sub.text    :List of 5
```

#### 4.2.8.4 Funkcija `trellis.par.set()`

Izmjene parametara mogu se provesti od strane korisnika pomoću `trellis.par.set()` funkcije što se, tipično, radi na sljedeći način (primjer za parametar `add.line`). Funkcija `trellis.par.set()` može se koristiti za modifikaciju `trellis` postavki unutar aktivne R sesije.

1) Postojeće postavke parametara ispitujemo:

```
str(trellis.par.get("plot.line"))
```

```
List of 4
$ alpha: num 1
$ col  : chr "#0080ff"
$ lty  : num 1
$ lwd  : num 1
```

Uočite da je linija definirana s 4 parametra: **alpha**, **col**, **lty** and **lwd**. Već smo upoznati sa svim ovim parametrima iz **base** grafike, grafičkim parametrima iz paketa **graphics**. Ako želimo promijeniti određeni element ili elemente s te liste, npr. širinu i boje za linije, trebamo napraviti sljedeće:

```
#izradivanje podataka
x <- rnorm(100, 20, 3)

#postavljanje grafičkih parametara za aktivnu R sesiju
trellis.par.set(plot.line = list(lwd=4, col = "darkred"))

#crtanje dijagrama
densityplot(x)
```

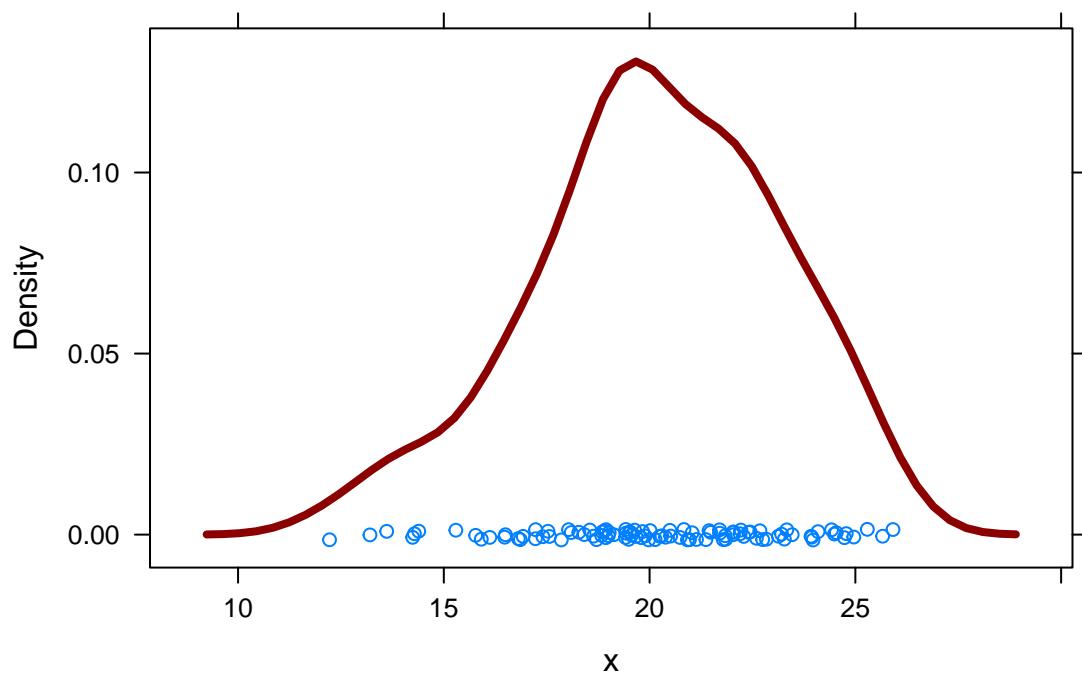
Ako želimo postavke `trellis` grafike vratiti na prvobitno stanje, ponovno koristimo funkciju `trellis.par.set()`.

```
#povratak na stare postavke
trellis.par.set(plot.line = list(lwd=1, col = "#0080ff"))
```

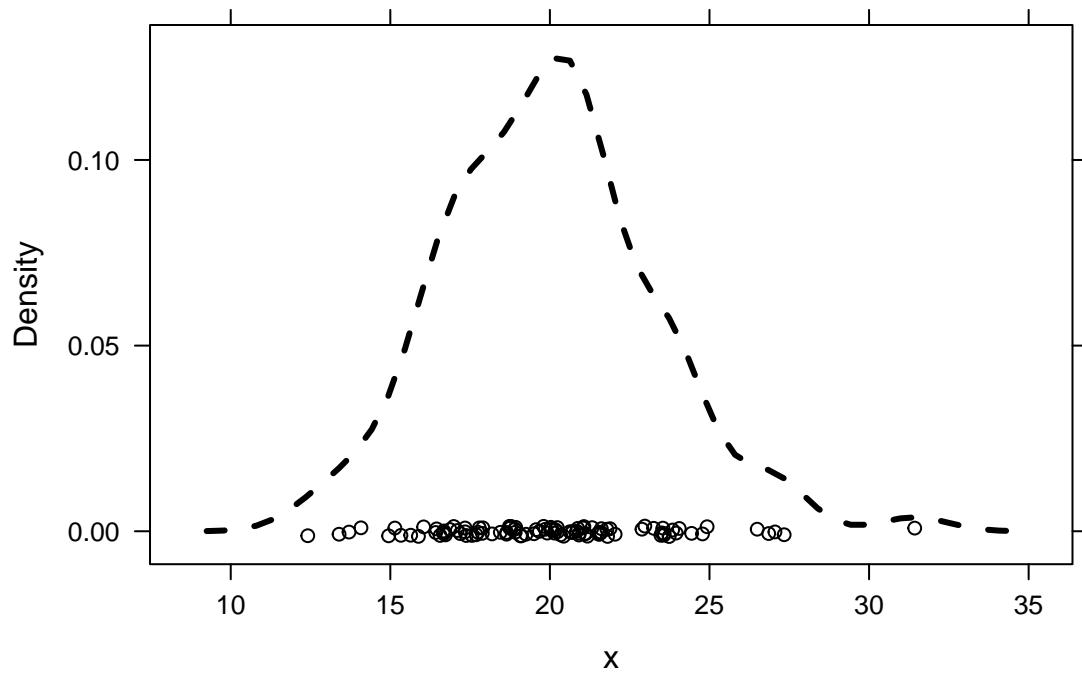
Budući da postoji mogućnost promjene grafičkoga izgleda na razini specifičnoga grafa visoke razine, najuobičajeniji (najlakši) način rada u *lattice* grafici je poput ovoga u sljedećem primjeru. Na taj smo način promijenili funkciju generalnoga izgleda (engl. *general display function*).

```
#postavljanje grafičkih parametara
x <- rnorm(100, 20, 3)

#zapamtiti da su neki parametri isti kao i u par() {graphics}
#plot
densityplot(x,
             lty=2,
             lwd=3,
             col="black",
             plot.symbol=22)
```



Slika 87: Promjena nekih postavki Trellis uređaja



Slika 88: Promjena nekih postavki Trellis uređaja

Primjeri za korištenje *par.settings*:

Zapamtiti: grafički parametri za elemente u *Trellis* dijagramima mogu se mijenjati (primijeniti) samo u jednom pozivanju naredbe, uporabom `*par.settings()` argumenta za bilo koje pozivanje visoke razine *latticea*. Drugi način je promjena globalnih postavki, što će se primijeniti na sve funkcije visoke razine u *latticeu* putem `trellis.par.set()**` funkcije kojom se daju argumenti željenih postavki za lattice dijagrame. Jedan primjer je kada ćemo definirati boje, vrste linija, boju pozadine za trake panela i staviti sve u objekt *my.settings*.

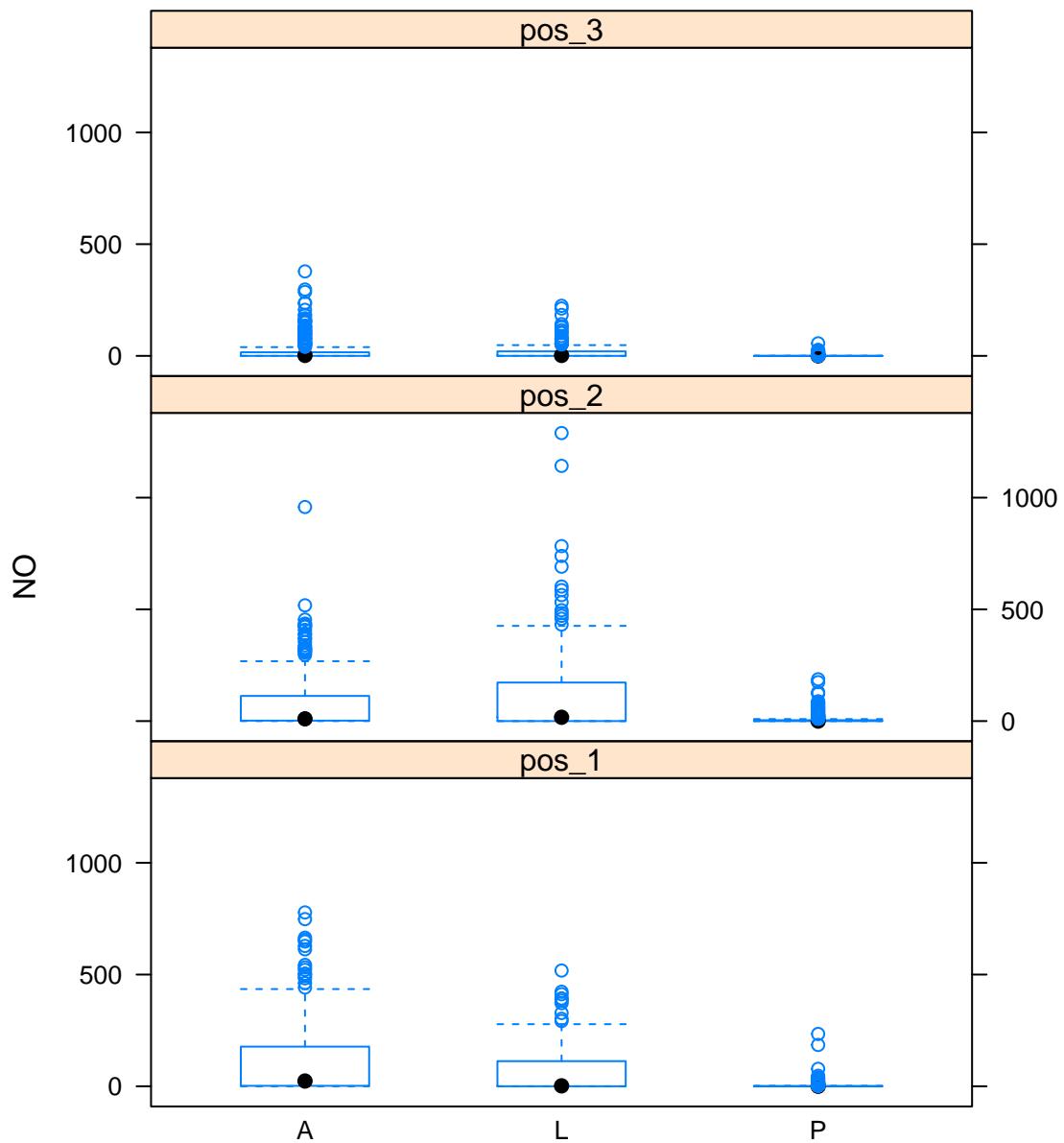
Prvo nacrtamo zadane grafičke parametre za *Trellis* uređaj:

```
trellis.par.set(old_theme)
```

```
#plot high-level Box-Whisker plot

bwplot(COUNT ~ A_L_P | POS_lab, data = bugs_data[bugs_data$OBJ=="2",] ,
       layout = c(1,3), stack = F, drop.unused.levels = F,
       auto.key = list(space = "right"),
       main="Zadane grafičke postavke",
       ylab = "NO")
```

### Zadane graficke postavke



Slika 89: Zadane postavke za trellis funkciju visoke razine; bw dijagram za podatke bugs\_data

Sada ćemo promijeniti neke grafičke parametre, a te će se promjene primijeniti samo na *lattice* funkciju unutra; u našem primjeru smo kreirali **temu** (bit će objašnjeno u narednom odjeljku) grafičkih parametara koji će se mijenjati. Molim komentirajte svaku liniju kôda koji slijedi:

```
#definira se lista grafičkih postavki koje će se promijeniti u prikazu
my.settings <- list(
  plot.symbol = list(col=c("black"), pch=c(20), cex=0.5),
  box.umbrella=list(col= c("dark gray"), lwd=1, lty=1),
  box.dot=list(col= c("lightpink3"), cex=0.8, pch=22),
  box.rectangle = list(col= c("black"), lwd=1),
  superpose.polygon=list(col="lightpink3", border="transparent"),
  strip.background=list(col="lightpink4"),
  strip.border=list(col="black")
) #zatvaramo listu postavki koje mijenjamo
```

Sada, primijenit ćemo definirane postavke na dijagram tipa Box-Whisker. Možemo primijeniti postavke svaki put kada koristimo Box-Whisker dijagrame bez potrebe za promjenom *general display function*.

```
bwplot(COUNT ~ A_L_P | POS_lab, data = bugs_data[bugs_data$OBJ=="2",],
       layout = c(1,3), stack = F, drop.unused.levels = F,
       auto.key = list(space = "right"),
       main="Postavke koje je korisnik definirao",
       ylab = "NO",
       par.settings = my.settings)
```

#### 4.2.8.5 Teme u *lattice* grafici {Lattice/LatticeExtra}

Prije nego isprobamo napraviti dodatne prilagodbe grafičkih prikaza u *latticeu*, ključno je razumjeti koncept **theme** u *lattice/trellis* postavkama. Kako smo već vidjeli, možemo doći do potpunog popisa grafičkih parametara za određeni uređaj putem **trellis.par.get()** funkcije. Moguće je promijeniti svaki element sa tog popisa. Korisnici obično mijenjaju te postavke putem mijenjanja *general display function* izravno kad pozivaju neku funkciju visoke razine, kao što se vidi iz prethodnog primjera. Ali, ponekad želimo promijeniti više elemenata i koristiti iste postavke u cijeloj R sesiji. Tada je korisno definirati koji elementi od onih opisanih na popisu grafičkih elemenata želimo promijeniti, zajedno sa definiranim promjenama u postavkama. Svi grafički parametri koji se mijenjaju u **otvorenom uređaju** mogu se sačuvati kao teme (engl. *theme* u *latticeu*). Dakle, **theme** je popis grafičkih parametara koji se trebaju promijeniti ili funkcija koja, kada je pozvana, proizvodi takav popis. Ako postoje grafički parametri koji nisu navedeni u *temi* (ili koje funkcija generira), ti se parametri neće promijeniti i koristi će se zadane vrijednosti. Izrađivanje korisničke funkcije za temu jako je korisno kada se slične postavke žele primijeniti u većem broju grafičkih prikaza.

Rad s temama: ovo je, uglavnom, opcija za korisnika, ali u primjeru koji slijedi pokazat ćemo jedan od najlakših načina. Sačuvat ćemo zadane postavke u objektu pod nazivom **old\_theme** da ih ponovno možemo pozvati kasnije. To napravimo unosom:

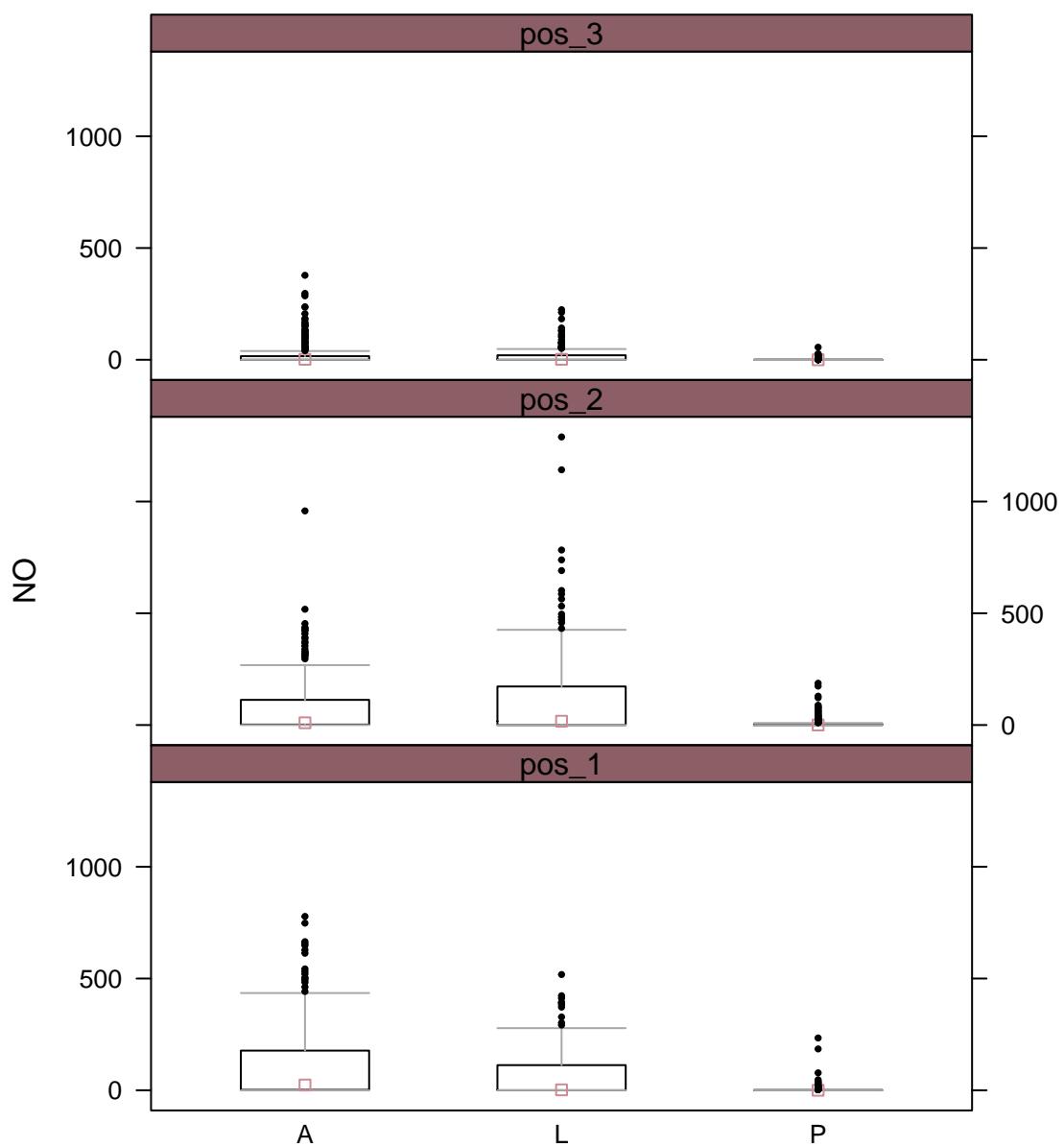
```
#u ovom slučaju trellis postavka za otvoreni uređaj - RStudioGD
old_theme <- trellis.par.get()
```

Sada imamo popis 35 grafičkih elemenata (duljina 35 za grafički uređaj RStudioG) u objektu **old\_theme**. Možemo napraviti novu kopiju liste parametara i pohraniti je u objekt pod nazivom **new\_theme** (ili bilo koje proizvoljno ime poput **my\_bw\_theme**, **theme\_for\_phd**, **theme\_my\_colors**) i promijeniti elemente u skladu sa svojim preferencijama.

```
#u ovom slučaju postavka za otvoreni trellis uređaj - RStudioGD
#izrađivanje objekta (liste) koji će se mijenjati
new_theme <- trellis.par.get()
```

Možemo koristiti bilo koji od tih popisa kada to želimo. Promijenit ćemo izgled simobola na grafu (**plot.symbol**), boju pozadine (engl. *background*) i širinu/vrste linija (**plot.line**). U narednim primjerima ćemo promijeniti izgled rezultirajućega grafičkog prikaza na nekoliko načina. Najznačajnije je razumjeti kôd i dobiti očekivani rezultat. Način na koji će neki mijenjati postavke ovisi o ukusu i preferencijama osobe.

### Postavke koje je korisnik definirao



Slika 90: Grafičke postavke definirane od strane korisnika unutar trellis funkcije visoke razine

```

#mijenja se pozadina
new_theme$background$col <- "lightgray"

#promjena širine linije na dijagramu
new_theme$plot.line$lwd <- 4

#promjena vrste linije na dijagramu
new_theme$plot.line$lty <- 2

#promjena boje linije
new_theme$plot.line$col <- "darkblue"

#promjena boje simbola na dijagramu
new_theme$plot.symbol$color<- "darkseagreen"

#promjena veličine simbola na dijagramu
new_theme$plot.symbol$cex<- 1.3

#promjena izgleda naslova
new_theme$par.main.text.col<-"darkred"

#promjena skupine fonta
new_theme$par.main.text.fontfamily <- "Arial"

```

Sada ćemo primijeniti nove postavke, prikazati postavke i nacrtati jednostavni dijagram raspršivanja koristeći stare i nove postavke:

```

#new settings
trellis.par.set(new_theme)
show.settings()

densityplot(x, main="Nove postavke s parametrima new_theme")

trellis.par.set(old_theme)

```

Sada ćemo se vratiti na izvorne, stare postavke, pogledajte:

```
densityplot(x, main="Povratak postavki na parametre old_theme")
```

Opisani način je jedan od mnogih kojima se može manevrirati popisom grafičkih parametara. U dijelu koji slijedi promijenit ćemo nekoliko elemenata u **new\_theme** i pokušati prelaziti s jednog na drugi od dva popisa grafičkih parametara.

Točnije, lakše bi bilo staviti sve željene promjene odjednom, zajedno na listu grafičkih parametara koji će se mijenjati, *theme*, kada se pozove. Moguće je definirati vlastite teme u *latticeu* i prelaziti između te vlastite teme i one koja je zadana "u letu" (engl. "on-the-fly"):

```

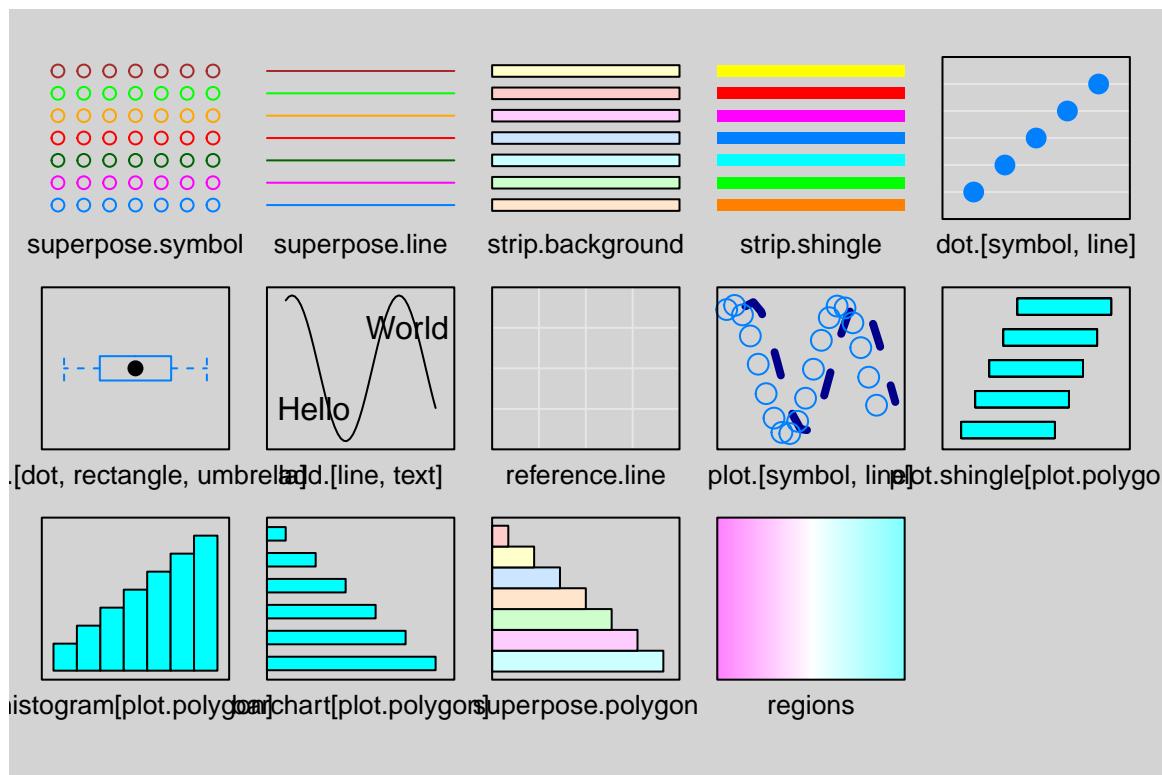
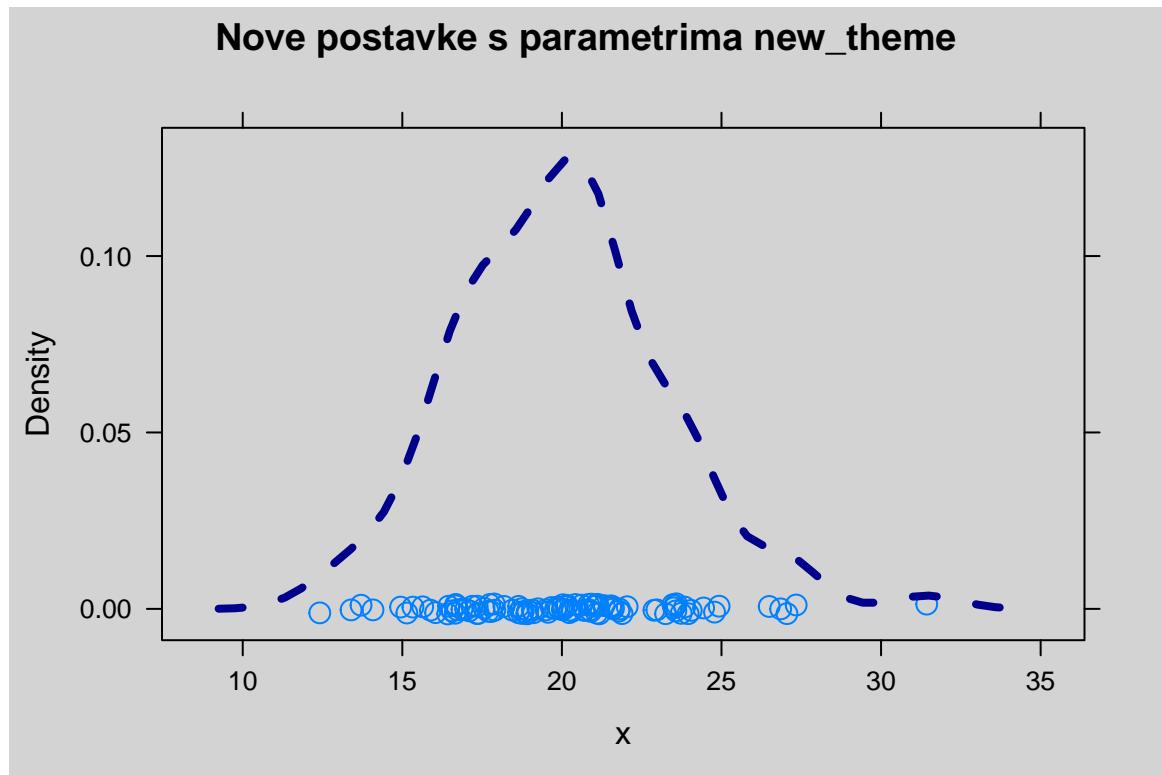
#zadana tema
trellis.device(new = FALSE, theme = NULL)

#funkcija moje definirane teme
#trellis.device(new = FALSE, theme = my_theme)

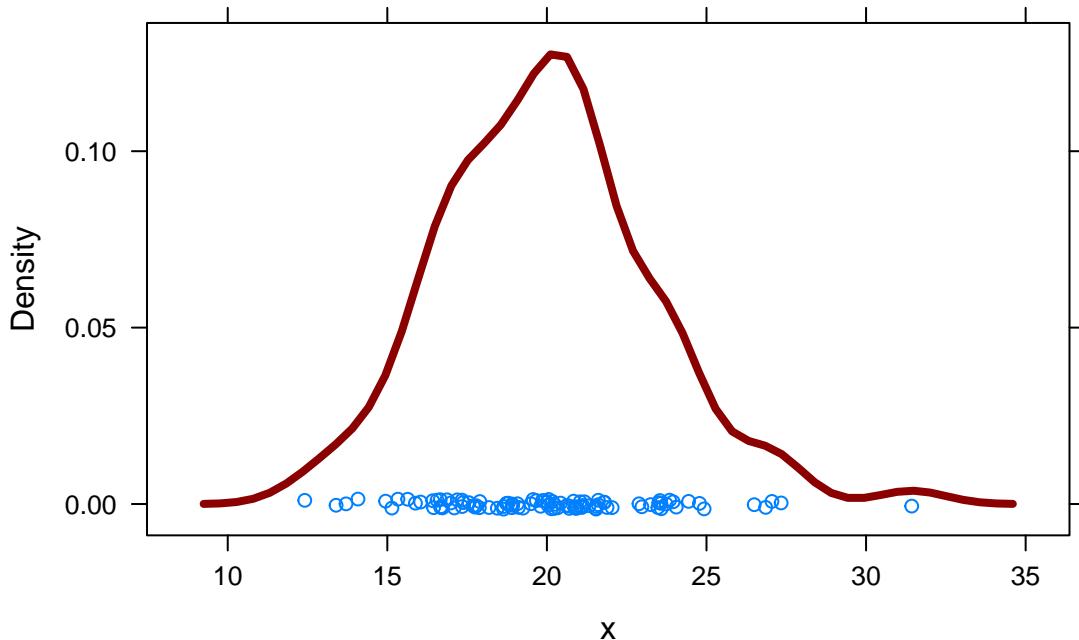
#povratak na zadanu kada se želi
trellis.device(new = FALSE, theme = NULL)

```

Postoje pripremljena sučelja za izrađivanje prilagođenih preferiranih tema, kao što je **simpletheme()** funkcija koja omogućava promjenu podskupa parametara, jednostavnija je u strukturi, s obzirom na to da traži vektorske vrijednosti umjesto popisa za preciziranje **theme**. Dodatna funkcija koja pomaže izraditi *lattice theme* je **custom.theme()** funkcija. Funkcija kreira **theme**

Slika 91: Nove grafičke postavke *trellis.par.set*Slika 92: Nove grafičke postavke; nekoliko elemenata uporabom popisa parametara koji će se mijenjati - *theme*

## Povratak postavki na parametre old\_theme



Slika 93: Nove postavke

dajući nekoliko boja. Dodatno, postoji mogućnost uporabe dodatnih, pripremljenih tema poput **col.whitebg()** ili **theme.mosaic()**. Kako smo spomenuli integraciju *base* i *lattice* grafičke, tako postoje i funkcije koje, primjerice, *latticeExtra* paketom oponašaju *ggplot* grafiku putem temom **ggplot2like()**.

U narednom primjeru dajemo primjer stvaranja korisničke teme s *BuPu* paletom *RColorBrewer* paketa. Tijekom ovog primjera dodatno ćemo se upoznati/podsjetiti na način *update*a postojećega *Trellis* objekta dodijeljenoga varijabli *p*, za što trebamo funkcionalnosti paket *LatticeExtra*:

```
#u ovom slučaju postavka za otvoreni trellis uređaj - RStudioGD
trellis.par.get(old_theme)
```

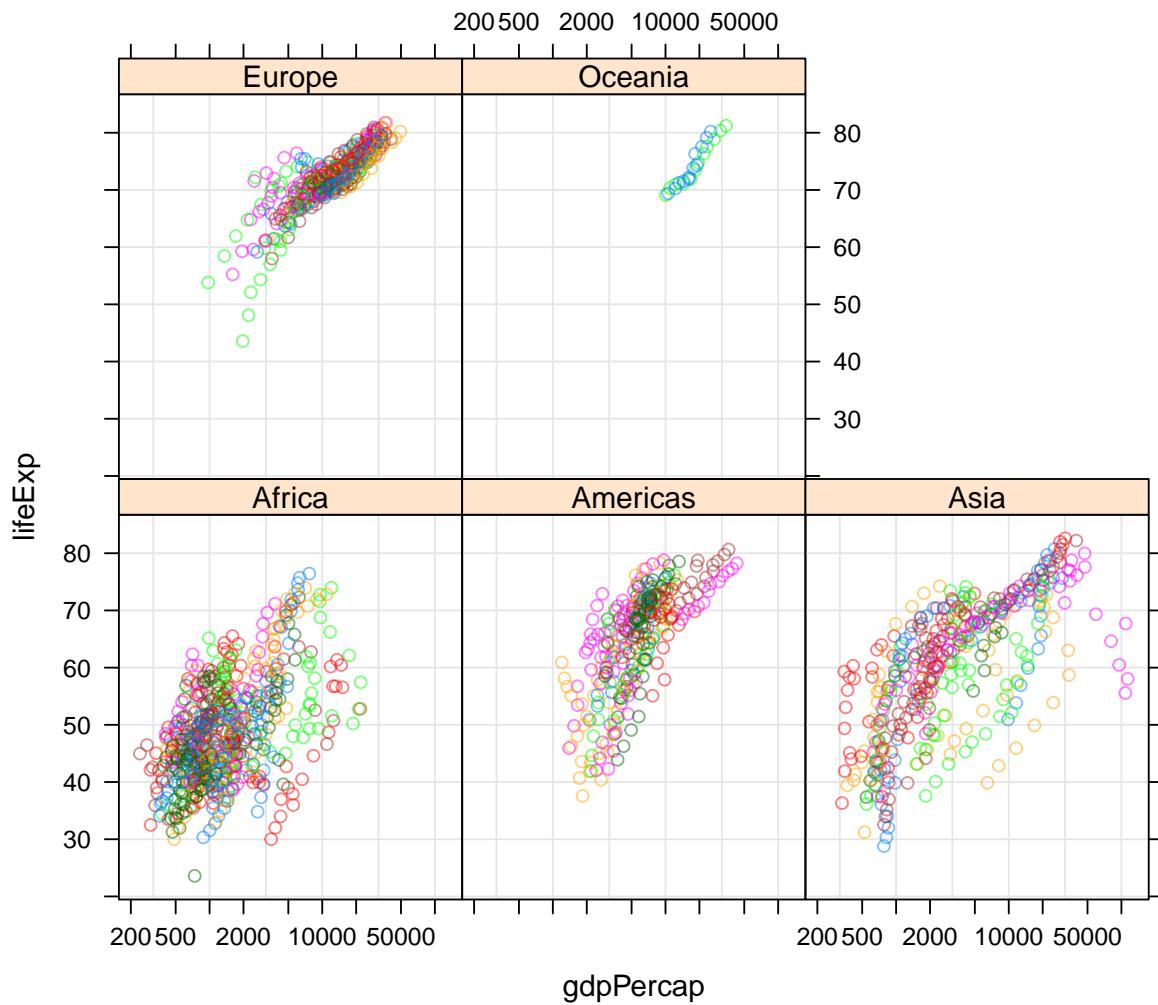
```
NULL

#cijeli *Trellis* objekt dodjeljujemo varijabli p
p <- xyplot(lifeExp ~ gdpPercap | continent,
            data=lattice_data,
            grid = TRUE,
            group= country,
            scales = list(x = list(log = 10, equispaced.log = FALSE)),
            type = c("p"), lwd = 4, alpha = 0.5)

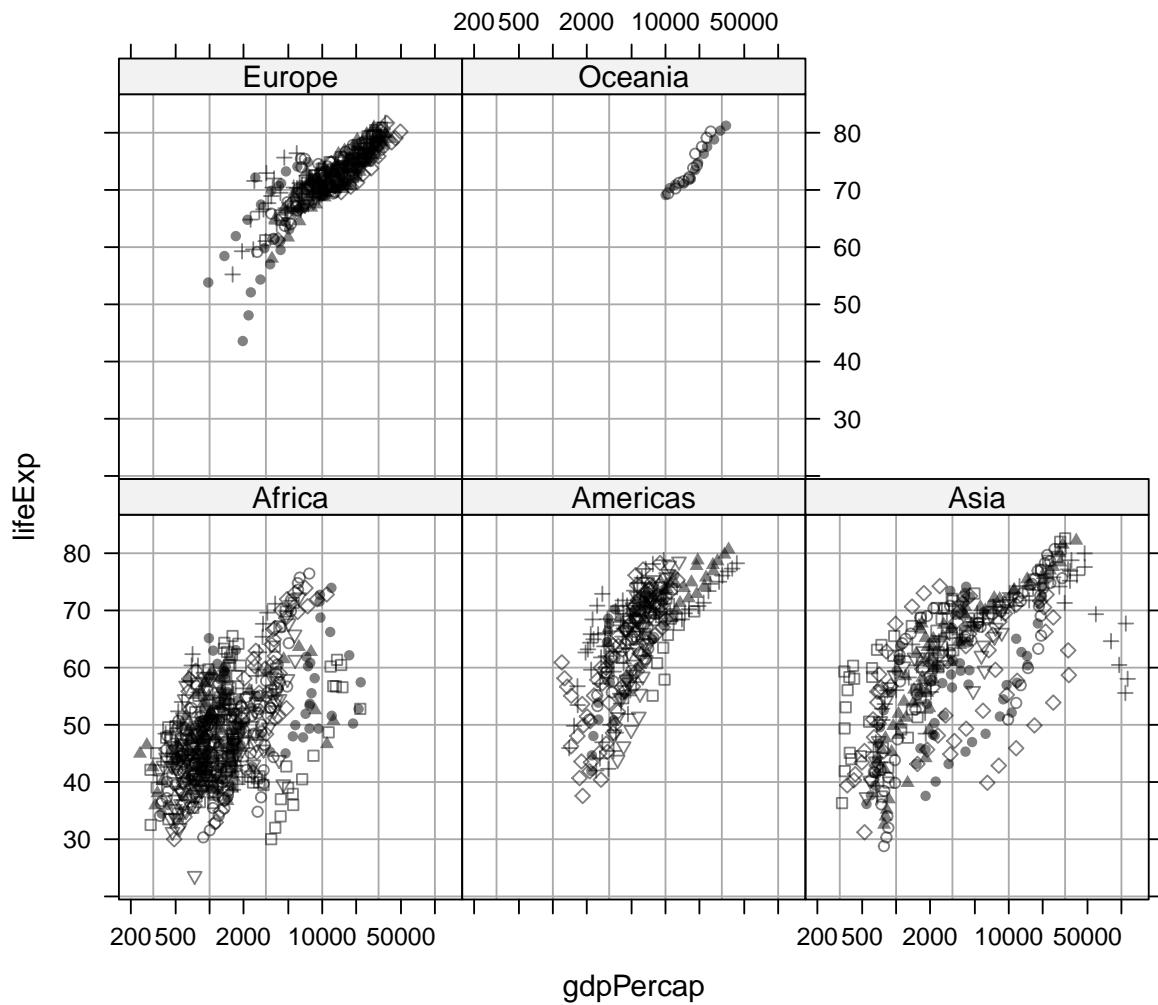
#crtamo varijablu p - Trellis objekt
p

update(p, par.settings = canonical.theme(color = FALSE))
```

Nakon nekoliko primjera vratit ćemo se na skup podataka *lattice\_data*, ponovno, i prezentaciji podataka dati konačni izgled. Prilagodit ćemo svaki element grafa odvojeno, ali prije toga važno je upoznati se sa definicijom i konceptom panela (engl. *panels*)



Slika 94: Izrađivanje *lattice* dijagrama za kasnije ažuriranje pomoću *custom.theme* bojanja

Slika 95: Osvježenje postojećeg trellis objekta s *custom.theme*

u *lattice* grafici.

#### 4.2.9 Panel funkcije u *lattice* grafici {*LatticeExtra*}

Stvaranje *Trellis* grafičkoga prikaza je proces od dva koraka:

U prvom koraku, *high-level* funkcija aktivira tzv. *funkciju generalnog prikaza* (engl. *general display function*) te kreira vanjske komponente prikaza kao što su okvir, osi, labele i drugi elementi grafa. U drugom koraku *general display function* poziva funkciju panela (engl. *panel function*) koja je odgovorna za sve što se crta u pojedinom panelu.

Svaki grafički prikaz u *lattice* grafici ima svoju predodređenu (engl. *default*) panel funkciju. Ime te funkcije je uvijek ime grafičkoga prikaza s prefiksom *panel*. Na taj način za funkciju **barchart()** postoji **panel.barchart()** panel funkcija; za **xyplot()** adekvatna **panel.xyplot()** i tako dalje redom za ostale prikaze. Panel funkcija prikaza poziva se uz pomoć *panel* argumenta općega prikaza (engl. *general display*). Pogledajmo sintaksu za **xyplot()**:

```
xyplot(y ~x, data=my_data,
       panel=panel.xyplot)
```

Svaki argument koji promijenimo u *general display function* utjecat će na sve na cjelokupnom grafičkom priazu; izravno se prenose na *panel* funkciju. U sljedećem primjeru mijenjamo *general display function*:

```
#pripremamo podatke za crtanje
x <- seq(0,pi,0.1)
y1 <- cos(x)
y2 <- sin(x)
my_data <- data.frame(x, y1, y2)

#graf
xyplot(y1~x, data=my_data,
       main="Parametri mijenjani kroz *general display function*",
       col="darkred", pch=22)
```

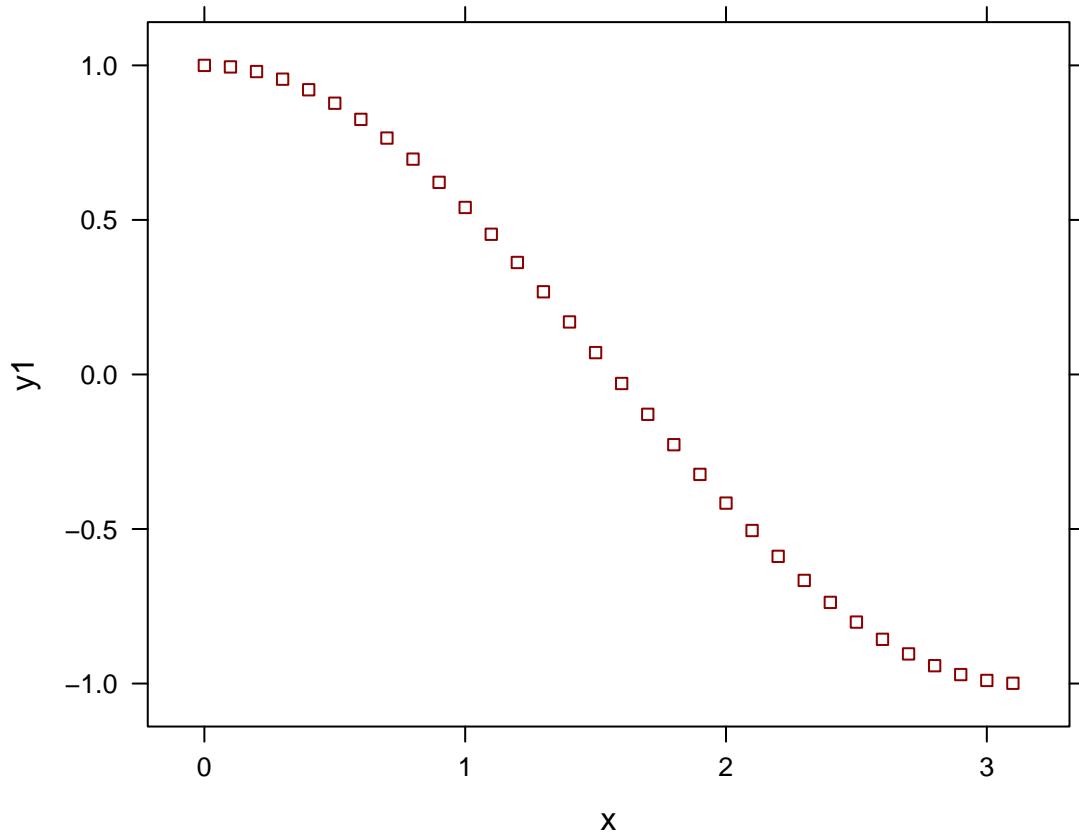
Argumenti poput *col*, *cex* i *pch* uvijek mijenjaju *general display function* (u našem primjeru histogram) i izravno su dani *panel.histogram* funkciji i kao takvi će bit primjenjeni na cjelokupan prikaz.

Drugi način promjene elemenata lattice prikaza je pozivom i modifikacijom *panel* funkcije izravno, pripremajući novu panel funkciju kojom ćemo zamjeniti staru.

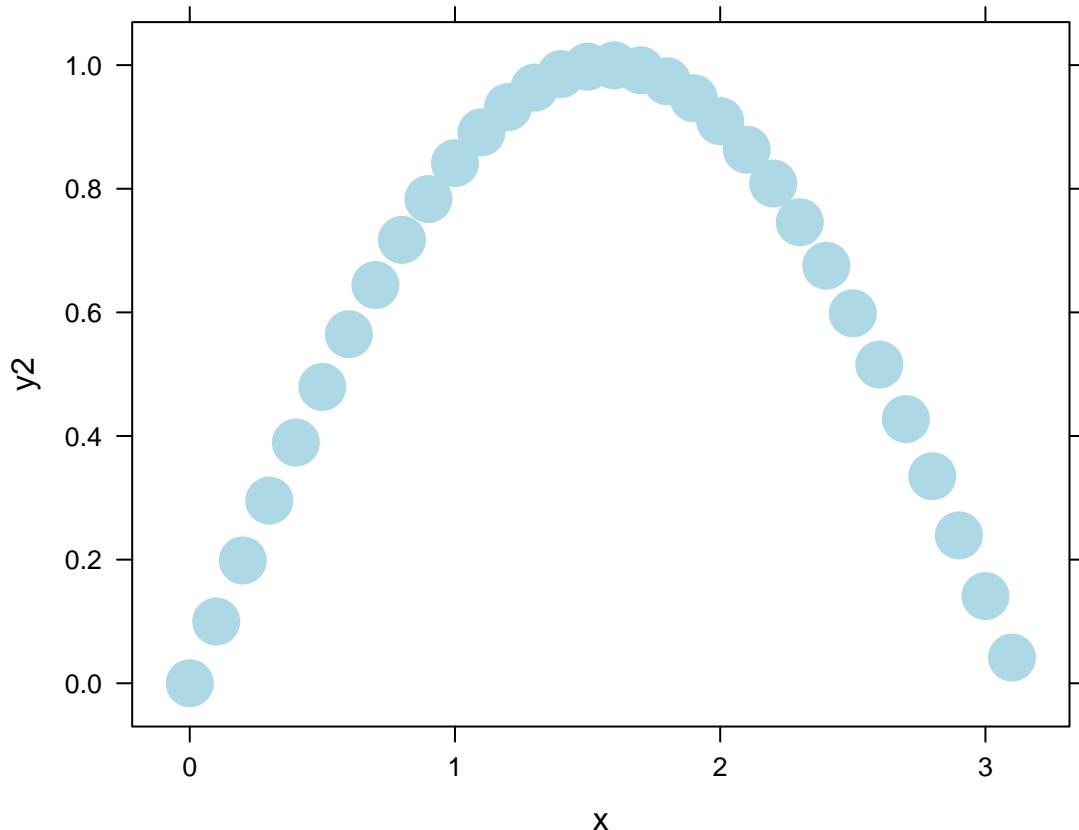
```
#graf
xyplot(y2~x, data=my_data,
       main="Promjena parametara kroz panel funkciju",
       panel = function (x, y) {
         panel.xyplot (x, y,
                       col="lightblue",
                       pch=16,
                       cex=3)
       }
)
```

Treći način promjene elemenata grafičkog prikaza je definiranje izgleda pojedinog parametra korištenjem **trellis.par.set()** funkcije. Na ovaj način postavke ostaju aktivne tijekom cijele R sesije. U sljedećem primjeru na ovaj ćemo način promijeniti neke grafičke parametre te ih zatim vratiti na *default* vrijednosti.

```
#postavljanje grafičkih parametara za R sesiju
trellis.par.set(plot.symbol = list(pch = 19,
                                      cex = 2,
                                      col = "darkseagreen"),
                plot.line = list(alpha = 0.8,
```

**Parametri mijenjani kroz \*general display function\***Slika 96: Grafički parametri promijenjeni putem *general display function*

### Promjena parametara kroz panel funkciju



Slika 97: Grafički parametri mijenjući *panel* funkciju izravno

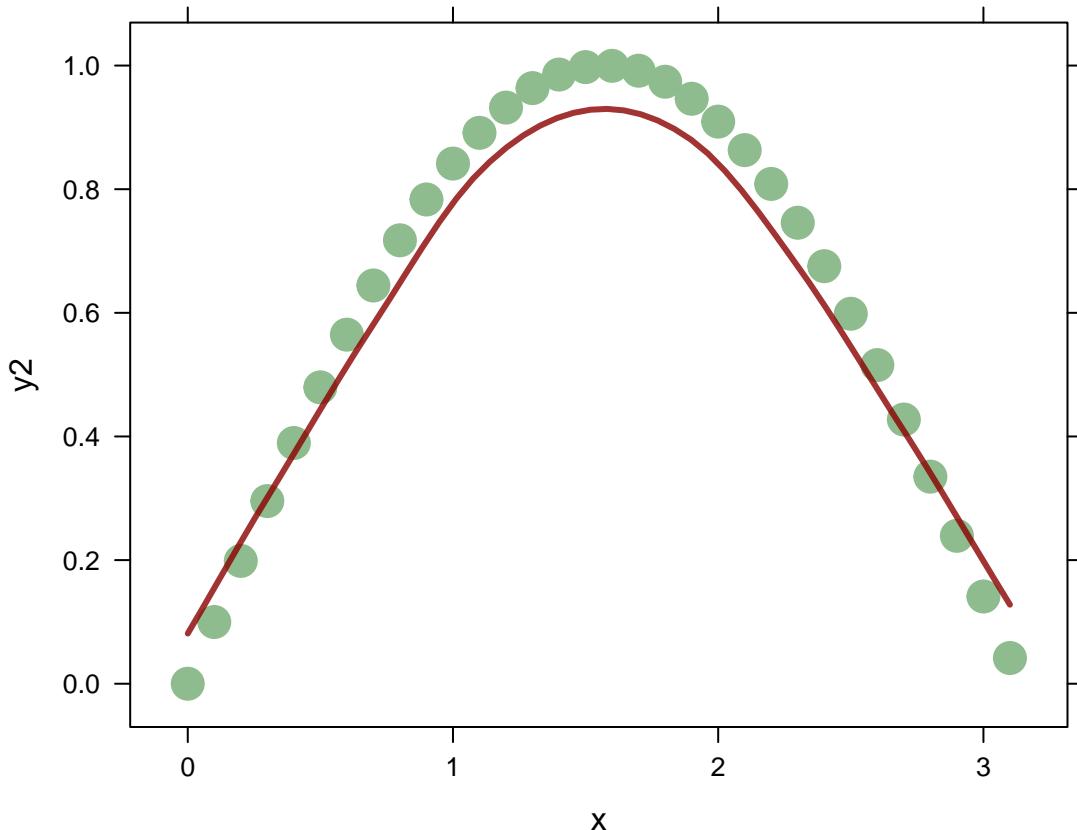
```

lwd = 3,
col  = "red4"),
strip.background = list(col="lightgray"))

#graf
xyplot(y2~x, data=my_data,
       type = c("p", "smooth"),
       main="Promjena grafičkih parametara koja vrijedi tijekom R sesije")

```

## Promjena grafičkih parametara koja vrijedi tijekom R sesije



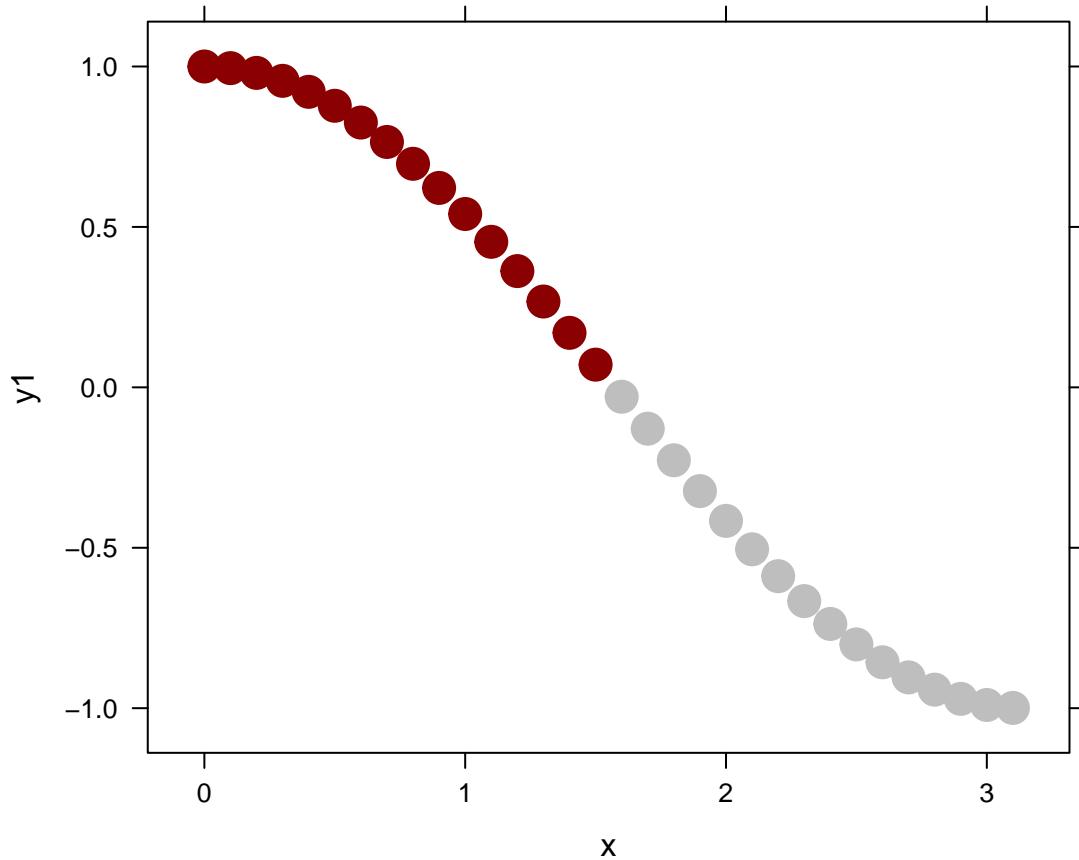
Slika 98: Grafički parametri za pojedinu R sesiju

Višestruke panel funkcije se često koriste simultano. Dodatno, panel funkcije mogu se koristiti i za dodavanje pojedinih elemenata grafičkoga prikaza:

- *panel.abline* - crta ravnu liniju na zadanom položaju
- *panel.lmline* - crta regresijsku liniju na dijagram raspršenja (engl. *scatterplot*)
- *panel.loess* - crta *loess fit* na dijagram raspršenja
- *panel.text* - dodaje tekst.

Jedna veoma korisna karakteristika panel funkcija je mogućnost selektivne modifikacije elemenata grafičkog prikaza prema nekom zadanom pravilu. U sljedećem primjeru definiramo različitu boju točaka prema zadanom pravilu:

```
#graf
xyplot(y1 ~ x, data = podaci_1,
       col = "black",
       panel = function (x, y) {
         panel.xyplot(x[x <= mean(x)], y[x <= mean(x)], col = "darkred")
         panel.xyplot(x[x > mean(x)], y[x > mean(x)], col = "gray")
       }
)
```



Slika 99: Grafički parametri za različit podskup podataka - zadovoljenje uvjeta za podskup podataka

Pažljivo pogledajte gornji primjer i odgovorite na koji je način definirana vrijednost parametra **cex**? Na koji način možete promijeniti vrijednost parametra na *default* vrijednost?

#### 4.2.10 Multi-panel uvjet u paketu *lattice*

Lattice grafika je veoma korisna ako želimo raditi zasebne panele prema nivoima faktora neke varijable.

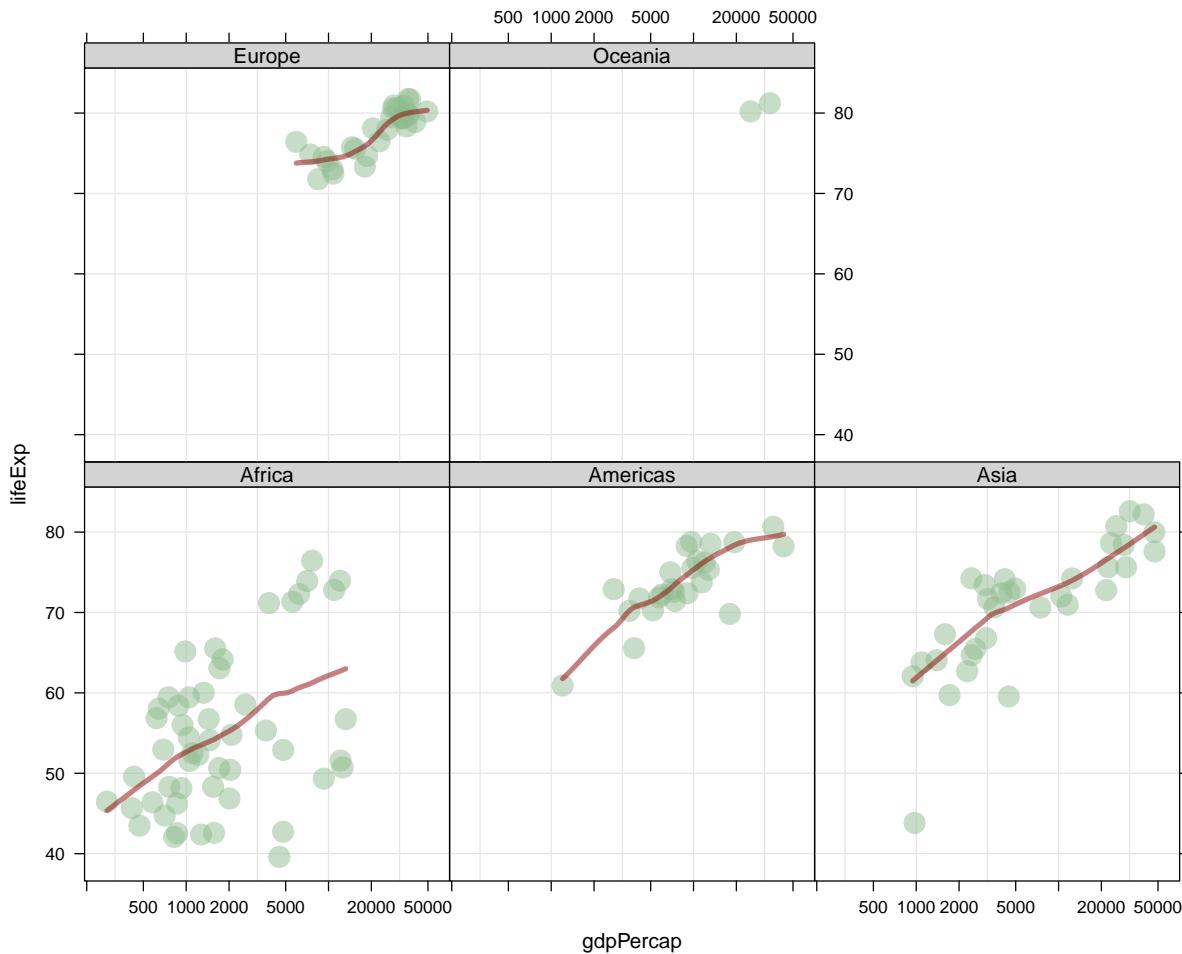
Postoji nekoliko funkcija u *lattice* paketu koje su stvorene za prikaz odnosa između nekoliko varijabli. Predstavljanje odnosa u multivarijatnim skupovima već je dugo vremena izazov. 3D dijagrami raspršenja (engl. *3D scatterplots*) su još uvijek prikladna vizualizacija za podatke do maksimalno tri varijable, ali ovaj pristup je nemoguć kada je broj dimenzije (varijabli) veći od tri. *Lattice* grafika koristi strategiju višestrukih panela. Koncept višestrukih panela temelji se na održavanju potencijalno važnih

varijabli konstantama. Za ilustraciju ovoga koncepta, pokušajte zamisliti skup podataka s tri varijable, var1, var2 i var3 na kojem želimo pokazati odnos (engl. *scatterplot*) var1 i var2 u nekom intervalu varijable var3. Lattice grafika će proizvesti takav podskup grafova za svaki interval varijable var3 raspoređenih u pravokutni uzorak (zbog toga je naziv *trellis/rešetke*). Na ovaj način izlaganja još moguće otkriti i prikazati složenu multivarijatnu strukturu podataka.

### Rješenja za preklapanja podataka na grafu

Ovisno o tome nastojimo li predočiti jednu ili više varijabli na istom grafu, statističari se često suočavaju s problemom preklapanja podataka kada više od jednog grafičkog objekta treba nacrtati na istim koordinatama, što skriva dio informacije. U univarijantnom okruženju, možemo takav problem riješiti uz pomoć *jitter* argumenta (na primjer u funkciji **stripplot()**). U dijagramima raspršenja rješenje za preklapanje može biti korištenje argumenta *alfa*; transparentnosti točaka ili putem argumenta odgovarajućega panela, primjerice, panel = **panel.smoothScatter()**. Usporedite rješenja u sljedeća dva primjera. Važno: ne podržavaju svi grafički uređaji *alfa* transparentnost.

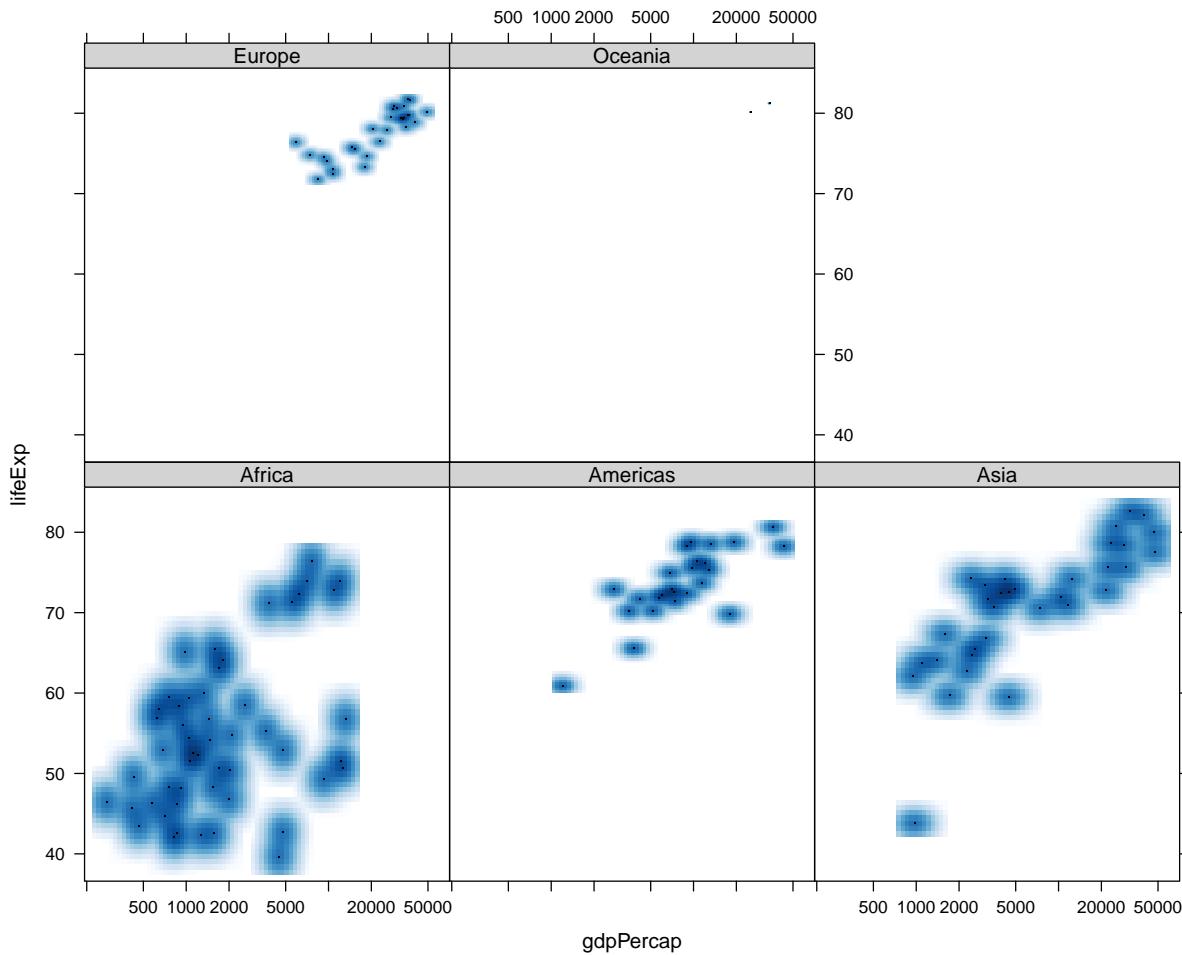
```
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data[lattice_data$year=="2007",],
       grid = TRUE,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p", "smooth"), lwd = 4, alpha = 0.5)
```



Slika 100: Razdvajanje podataka prema nivou faktorske varijable; podaci za jednu godinu

Pronađite u sljedećem primjeru parametar koji mijenja izgled prikaza:

```
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data[lattice_data$year=="2007",],
       grid = TRUE,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       panel = panel.smoothScatter)
```



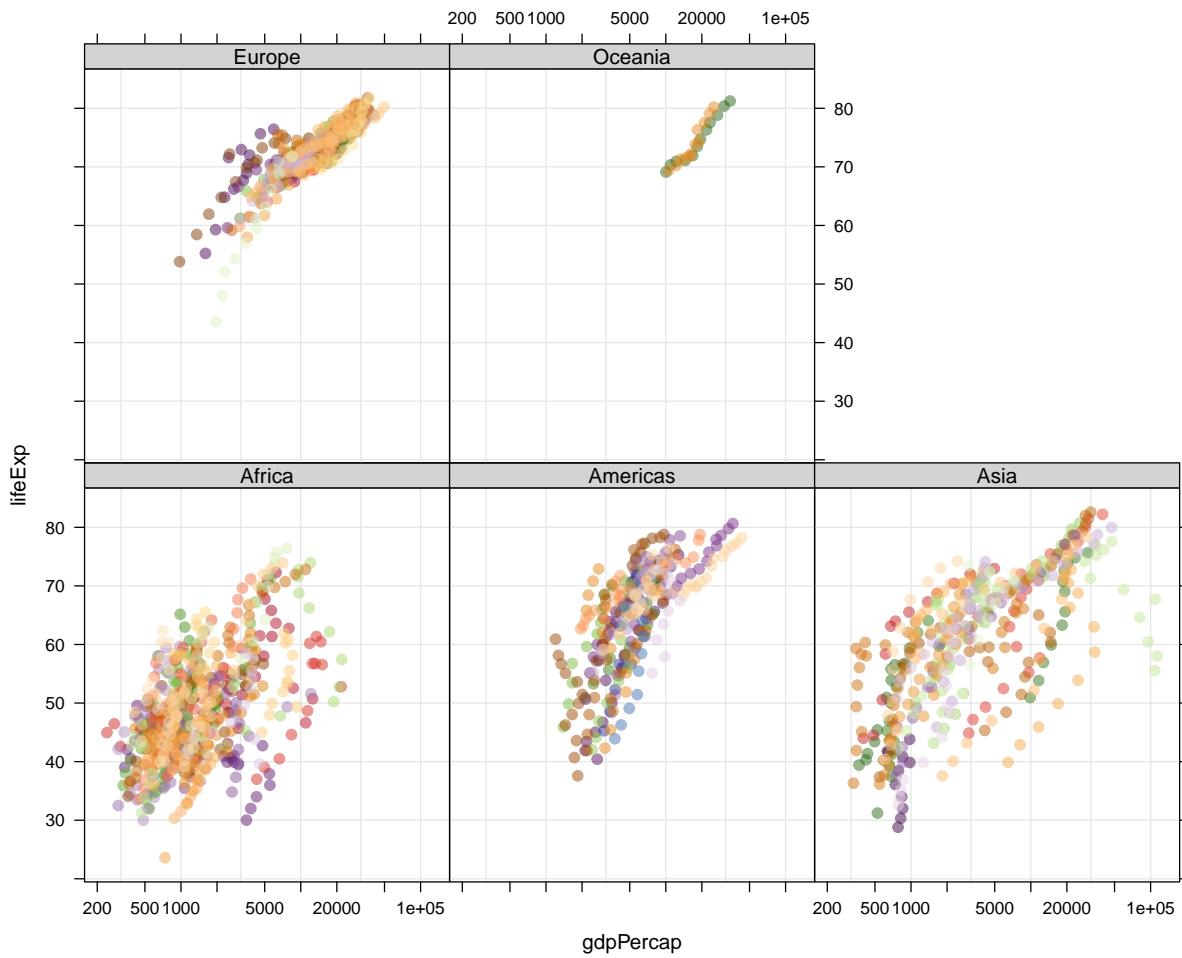
Slika 101: Rješenja za preklapajuće podatke kroz *panel*

Usporedite i komentirajte prethodni i sljedeći primjer. Kao što smo ranije onemogućili boju na otvorenom grafičkom uređaju, sada mijenjamo izgled pojedinoga grafičkog elementa. Želimo drugačiju boju simbola za svaki nivo faktora *country*. Ovo je uobičajen način definiranja boja za razine nekih faktora varijable u R-u; čuvaju se podatci o željenoj boji za svaki element zajedno s mjerjenjima (engl. *data*) u istom objektu, u našem slučaju **data.frame** klasa naziva *lattice\_data*.

```
trellis.par.set(superpose.symbol = list(pch = 19, cex = 1,
                                         col = lattice_data$country_color))
```

```
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data,
       grid = TRUE,
       group= country,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
```

```
type = c("p"), lwd = 4, alpha = 0.5)
```



Slika 102: Različiti podskupovi podataka za nivo faktora; promjena kroz *general display function*

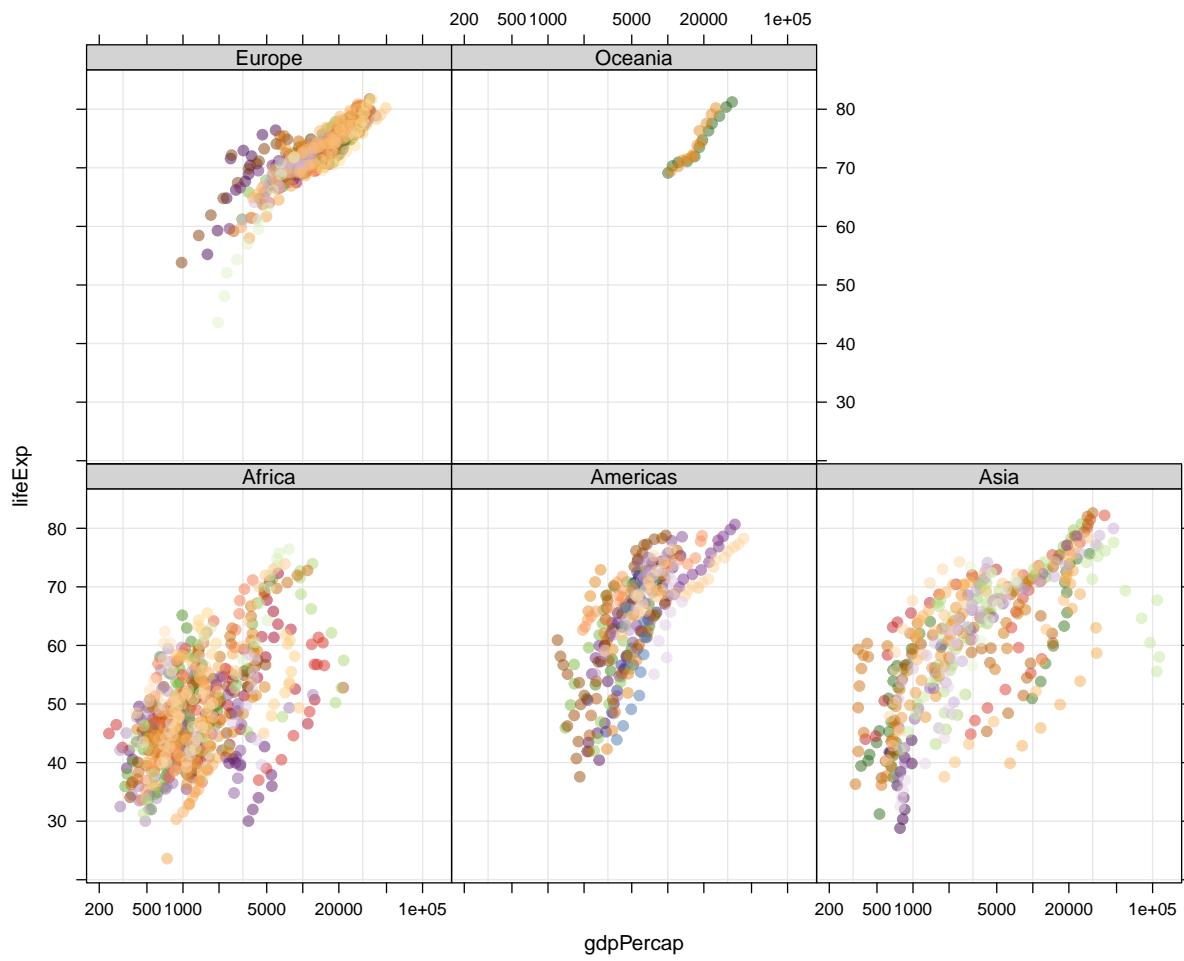
Ovaj rezultat je već blizu onoga što smo željeli. Jedino što još uvijek nije kako želimo jest parametar/element grafa *strip.background* i mi ćemo ga promjeniti na dva načina: 1) izravno u funkciji visoke razini (engl. *high-level*) **xyplot()** funkcije i 2) definiranjem **trellis.par.set()** stvaranjem **xy\_plt\_setting** popisa za daljnje korištenje.

```
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data,
       grid = TRUE,
       group= country,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p"), lwd = 4, alpha = 1/2,
       strip.background=list(col="lightpink4"))
```

I drugo rješenje, funkcija **trellis.par.get()** koji će se primjenjivati na sve *stripe.background* tijekom aktivne R sesije.

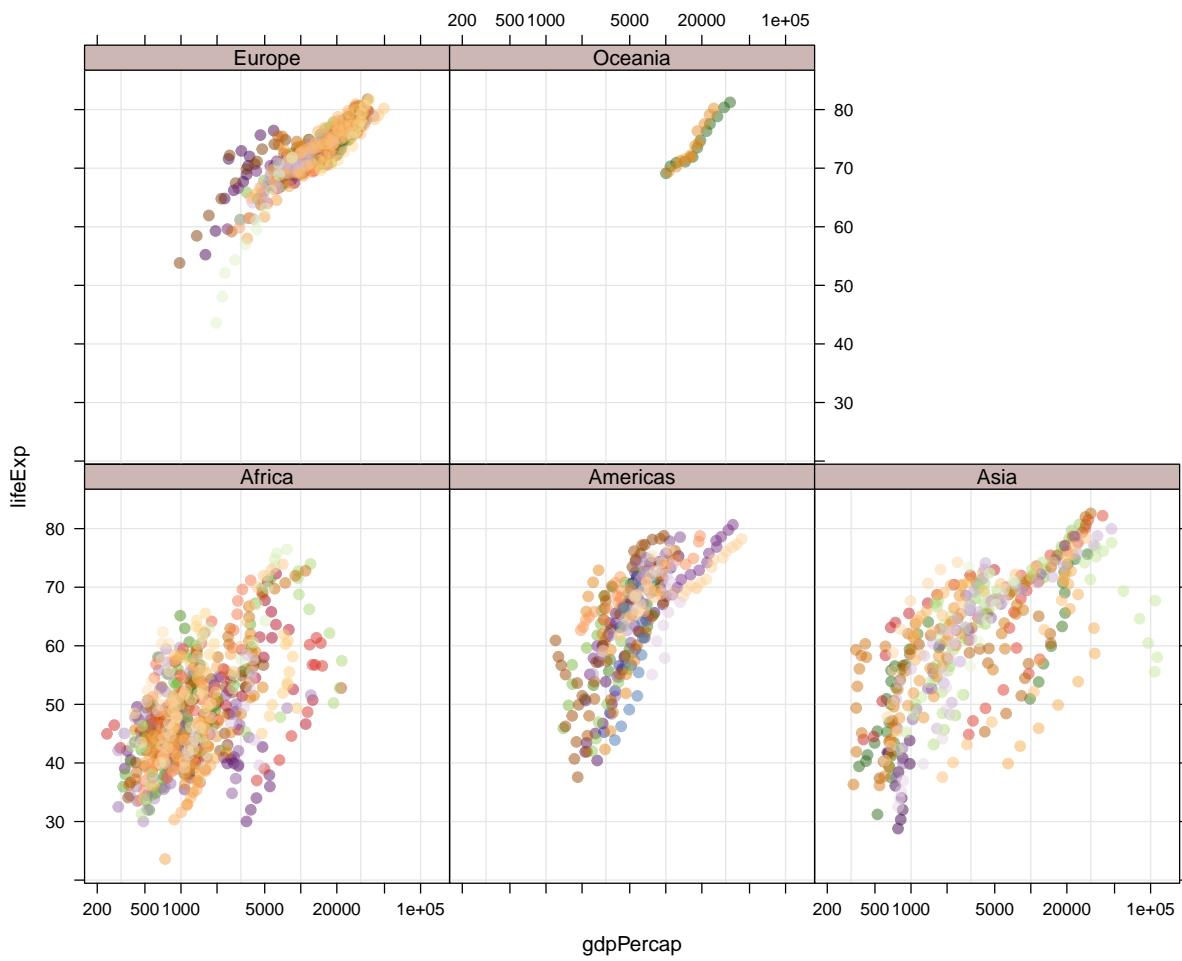
```
#grafičke postavke za aktivnu R sesiju
xy_plt_setting <- trellis.par.set(strip.background=list(col="mistyrose3"))

#plot the data
```



Slika 103: Različiti podskupovi podataka za nivo faktora; promjena kroz *general display function*

```
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data,
       grid = TRUE,
       group= country,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p"), lwd = 4, alpha = 1/2,
       par.settings = xy_plt_setting)
```



Slika 104: Različiti podskupovi podataka za nivo faktora; **trellis.par.set()**

#### 4.2.11 Grafički parametri koji kontroliraju regiju crtanja

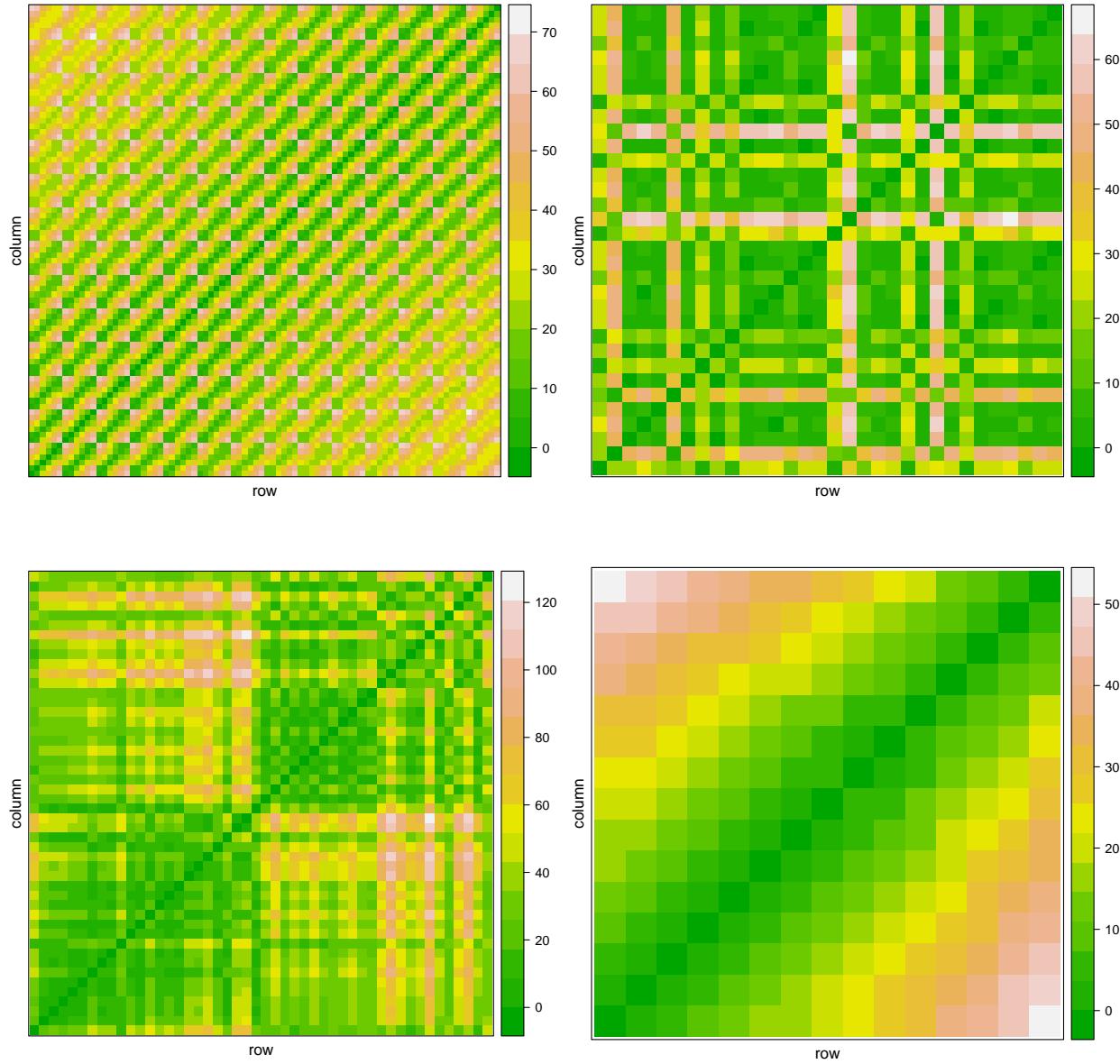
U ovom ćemo dijelu nastojati pokazati kako postići dodatnu kontrolu nad *lattice* grafičkim prikazima.

Primjer koji slijedi pokazuje kako, između ostalih načina, organizirati nekoliko dijagrama u jednoj "sceni" - rezultat je sličan kao kad se koristi `par(mfrow=c(2,2))` u *base* grafici. Zapamtite kako R radi - pokušava spojiti klasu objekata s adekvatnom metodom. U ovom primjeru **generic** funkcija ispisa primijenjena na *Trellis* objekt uparena je s `print.trellis()` metodom. Argument `split` ima četiri parametra. Zadnja dva odnose se na veličinu okvira, prisetite se grafičkoga parametra *base* grafike `mfrow()`, gdje prva dva parametra pozicioniraju Vaš dijagram u `nx` sa `ny` okvir.

```
#radimo podatke
w <- as.matrix(dist(Loblolly))
x <- as.matrix(dist(HairEyeColor))
y <- as.matrix(dist(rock))/100
z <- as.matrix(dist(women))

#grafovi dodijeljeni varijabli
#skale argumenta uklanjaju se osi
pw <- levelplot(w, col.regions=terrain.colors(25), scales = list(draw = FALSE))
px <- levelplot(x, col.regions=terrain.colors(25), scales = list(draw = FALSE))
py <- levelplot(y, col.regions=terrain.colors(25), scales = list(draw = FALSE))
pz <- levelplot(z, col.regions=terrain.colors(25), scales = list(draw = FALSE))

# dijagram se ispisuje
print(pw, split = c(1, 1, 2, 2), more = TRUE)
print(px, split = c(2, 1, 2, 2), more = TRUE)
print(py, split = c(1, 2, 2, 2), more = TRUE)
print(pz, split = c(2, 2, 2, 2), more = FALSE)
```

Slika 105: Organizacija crtanja grafova za *Trellis* uređaj

#### 4.2.12 Najznačajnije funkcije visoke razine iz paketa *lattice*

##### 4.2.12.1 Funkcija **xyplot()**{*lattice*}

Sve funkcije visoke razine u *lattice* paketu su generičke. Funkcija **xyplot()** je vjerojatno najpopularnija i najkorištenija funkcija iz paketa *lattice*; funkcija kreira opće bivarijatne *Trellis* dijagrame. Većina smještanja panela će se objasniti kroz korištenje ove funkcije ali je isto moguće i kod svih drugih ranije spomenutih funkcija.

Unutar ove funkcije moguće je primijeniti sve ranije spomenute postavke grafičkih parametara.

Kao i obično, prije primjene nove funkcije, potrebno se s njom upoznati čitanjem informacijske kartice.

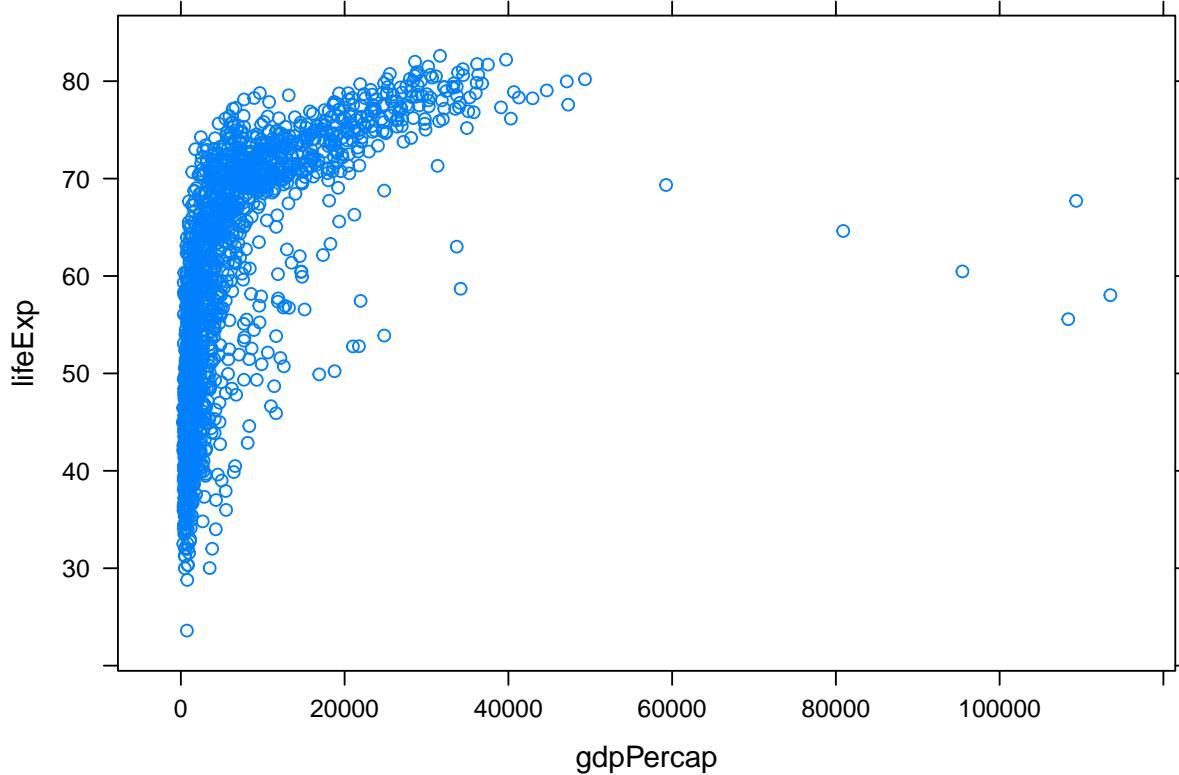
```
?lattice::xyplot
```

Kako je spomenuto, funkcije *Trellis* grafike mogu proizvesti višestruke panel prikaze, također su izvrsne za izradu osnovnih grafičkih jednoga panela. U ovom primjeru ćemo koristiti već opisane podatke *lattice\_data*. Ovaj grafički prikaz, dijagramima raspršenja, najpopularniji je prikaz dviju kontinuiranih varijabli. Dijagram raspršenja konceptualno je jednostavna, ali ipak moćna prezentacija podataka. Sljedi primjer:

```
trellis.par.set(old_theme)
```

```
xyplot(lifeExp ~ gdpPercap,
       data=lattice_data,
       main="Jednostavni dijagram raspršenja dvije numericke varijable")
```

### Jednostavni dijagram raspršenja dvije numericke varijable



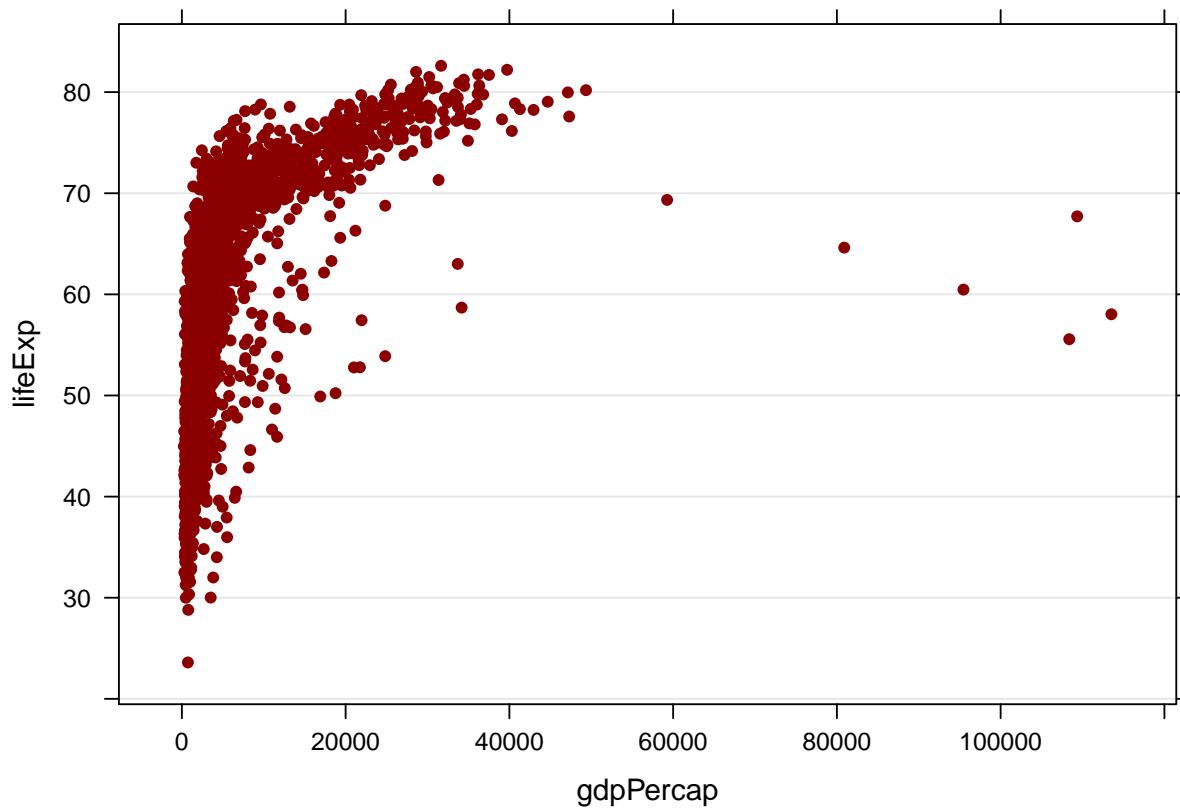
Slika 106: Funkcija **xyplot()**; jednostavan dijagram raspršenja

Sada ćemo izraditi isti, jednostavni dijagram raspršenja za varijable očekivane životne dobi i BDP-a korištenjem funkcije **xyplot()**,

uz dodatnu promjenu grafičkih parametara: uočite novi parametar *grid*.

```
p2 <- xyplot(lifeExp ~ gdpPercap,  
              lattice_data,  
              pch=16,  
              col="red4",  
              grid="h")
```

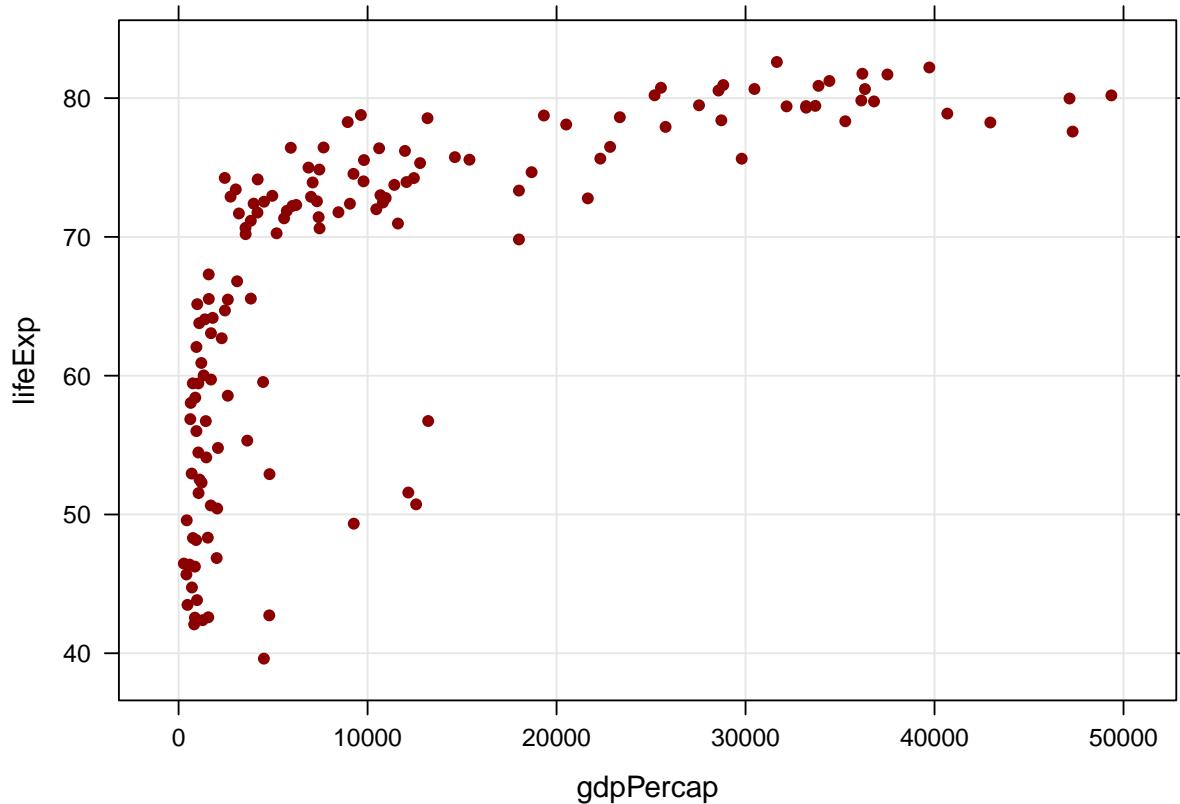
p2



Slika 107: Funkcija **xyplot()**; dijagram raspršenja dvije kvantitativne varijable, postavka dodatnih grafičkih parametara i postavljena horizontalna mreža pozadine

U sljedećim ćemo primjerima nacrtati samo podskup podataka, jednu godinu, te naglasiti labele osi grafa:

```
xypplot(lifeExp ~ gdpPercap,
        lattice_data[lattice_data$year=="2007",], #usporediti s argumentom *subset* - kasnije
#subset= year== "2007",
        pch=16,
        col="red4",
        grid=T)
```

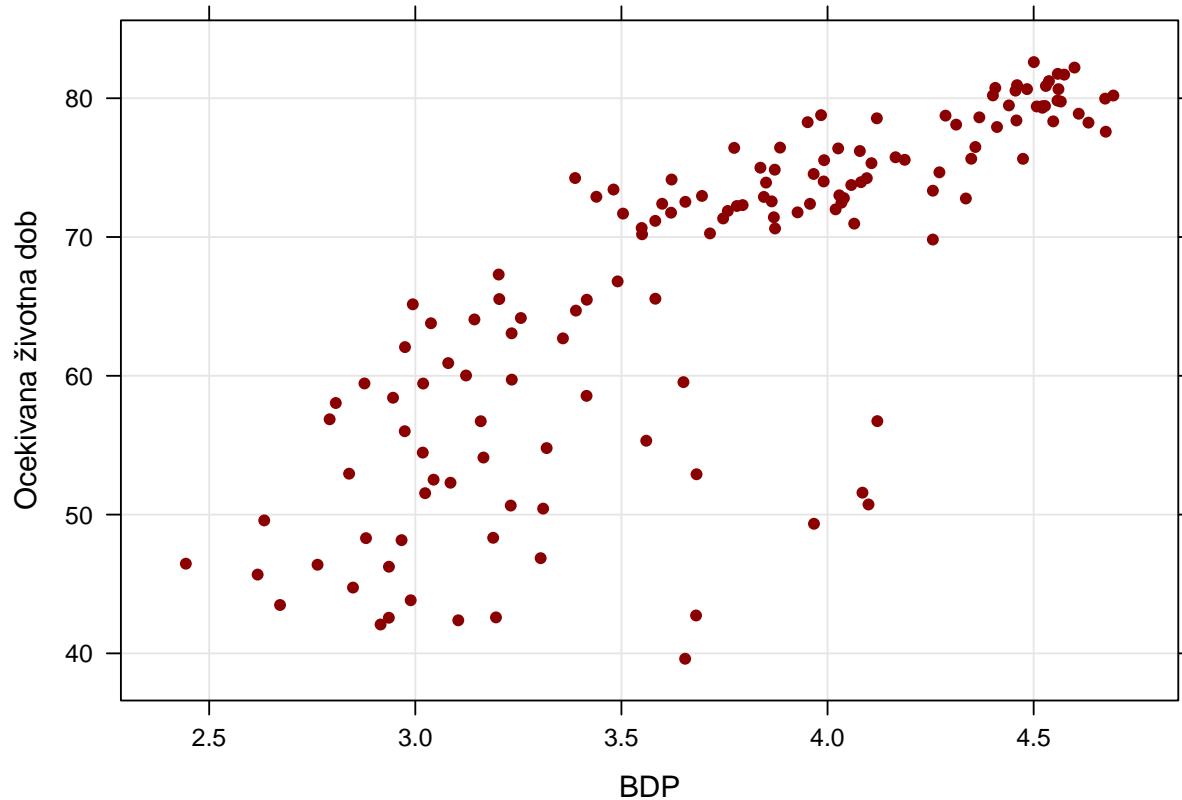


Slika 108: Funkcija **xypplot()**; dijagram raspršenja dviju kvantitativnih varijabli, a) odabir jedne godine

Molim komentirajte razliku u gridovima pozadine u zadnja dva prikaza.

Ako želimo koristiti vrijednosti jedne varijable na logaritamskoj skali, možemo prilagoditi os "ručno" ili primjenom **scales** argumenta unutar **xypplot()** funkcije. Za dodatne prilagodbe osi trebamo koristiti funkcije *latticeExtra* paketa što je izvan djelokruga našega tečaja.

```
xypplot(lifeExp ~ log10(gdpPercap),
        lattice_data[lattice_data$year=="2007",],
        pch=16,
        col="red4",
        grid=T,
        xlab = "BDP",
        ylab = "Očekivana životna dob")
```



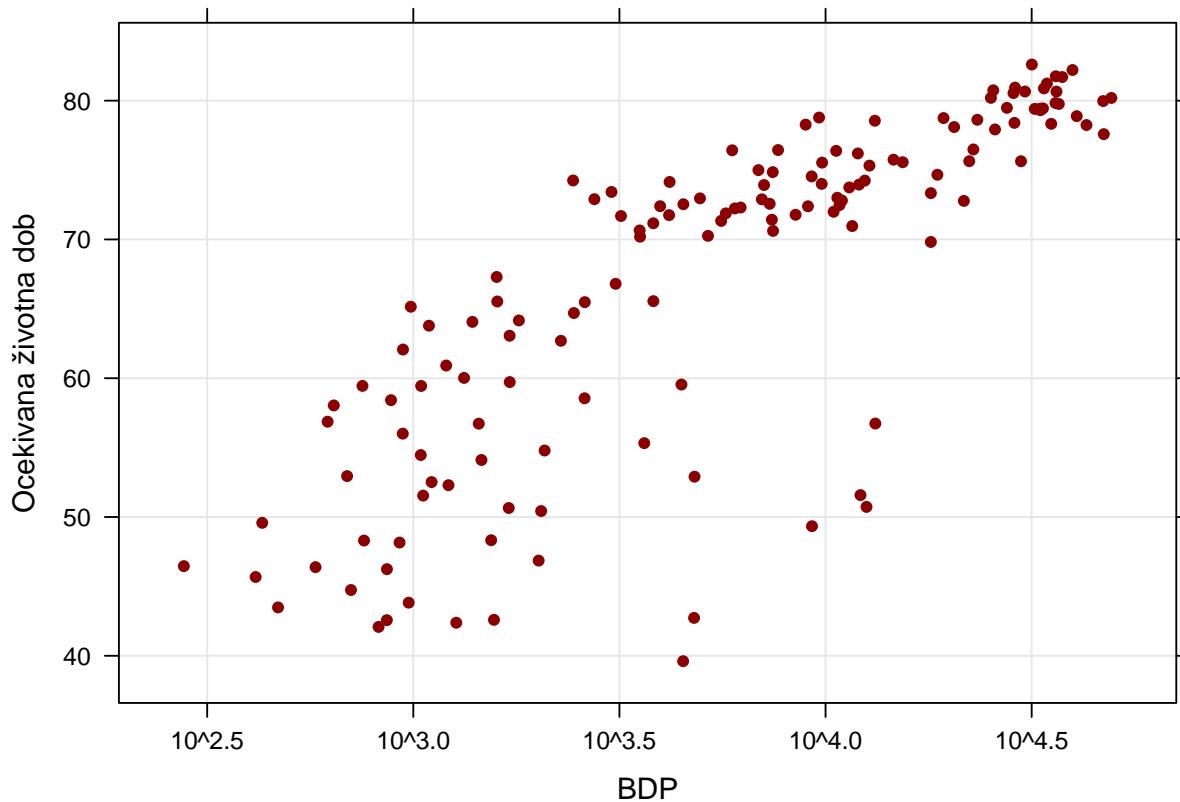
Slika 109: Funkcija **xyplot()**; dijagram raspršenja dviju kvantitativnih varijabli; ručno podešena log skala osi

#### 4.2.13 Poboljšanje rezultata kreiranog *high-level* funkcijom funkcijama niske razine (*low-level*)

##### 4.2.13.1 Argument scales

Korištenje argumenta `scales` radimo pravilno skaliranje vrijedosti osi kako je prikazano primjerom:

```
xyplot(lifeExp ~ gdpPerCap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       col="red4",
       grid=T,
       xlab = "BDP",
       ylab = "Očekivana životna dob",
       scales = list(x = list(log = 10)))
```



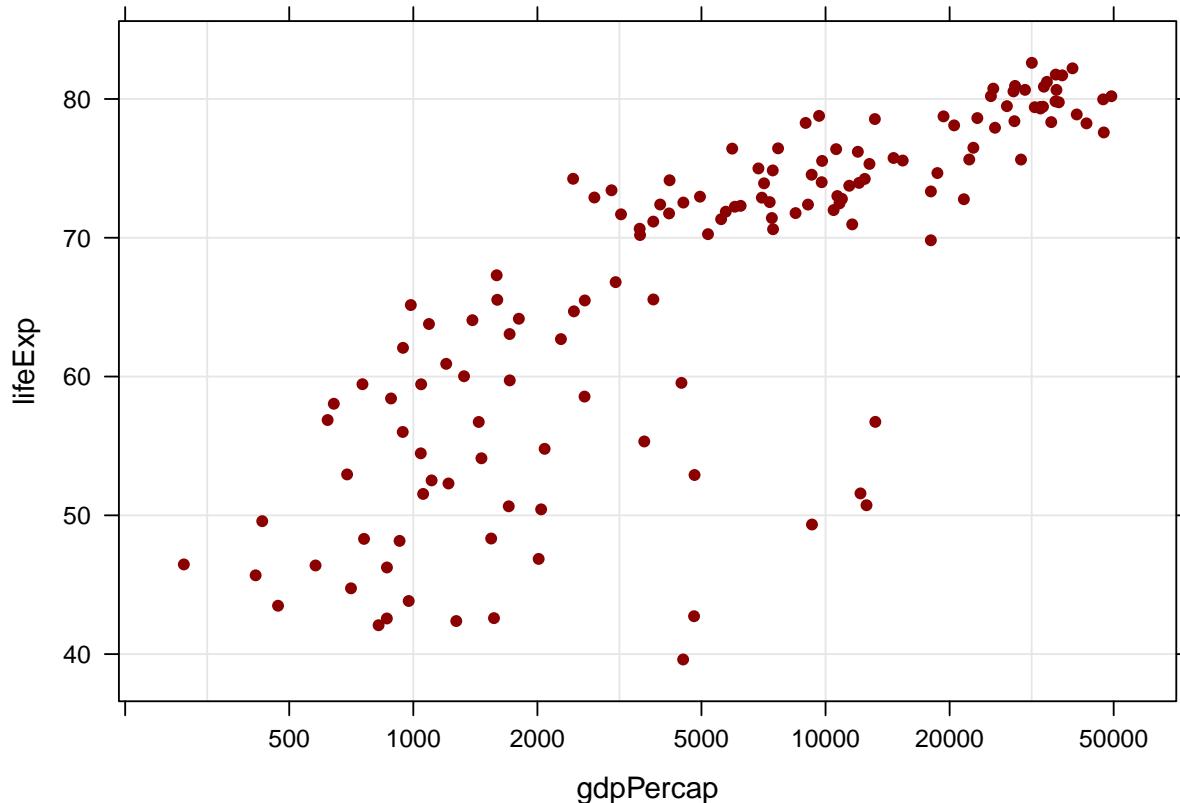
Slika 110: Funkcija `xyplot()`; dijagram raspršenja dvije kvantitativne varijable; `scale` argument

##### 4.2.13.2 Argument type

Argument `type` koristi se za promjene parametara reprezentacije podataka. Default vrijednost funkcije `type = "p"`, što označava da će na grafičkom prikazu podaci biti prikazani kao točke u koordinatnom sustavu. Dodatno je moguće raditi i kombinacije kao što je primjerice `type = c("p", "r")` koji kao rezultat daje i točke i regresijsku liniju.

U sljedećem primjeru slično kao i u \*base grafici graf radimo na način crtanja a) samo točaka i b) fitamo regresijsku liniju na točke; parametar `type`.

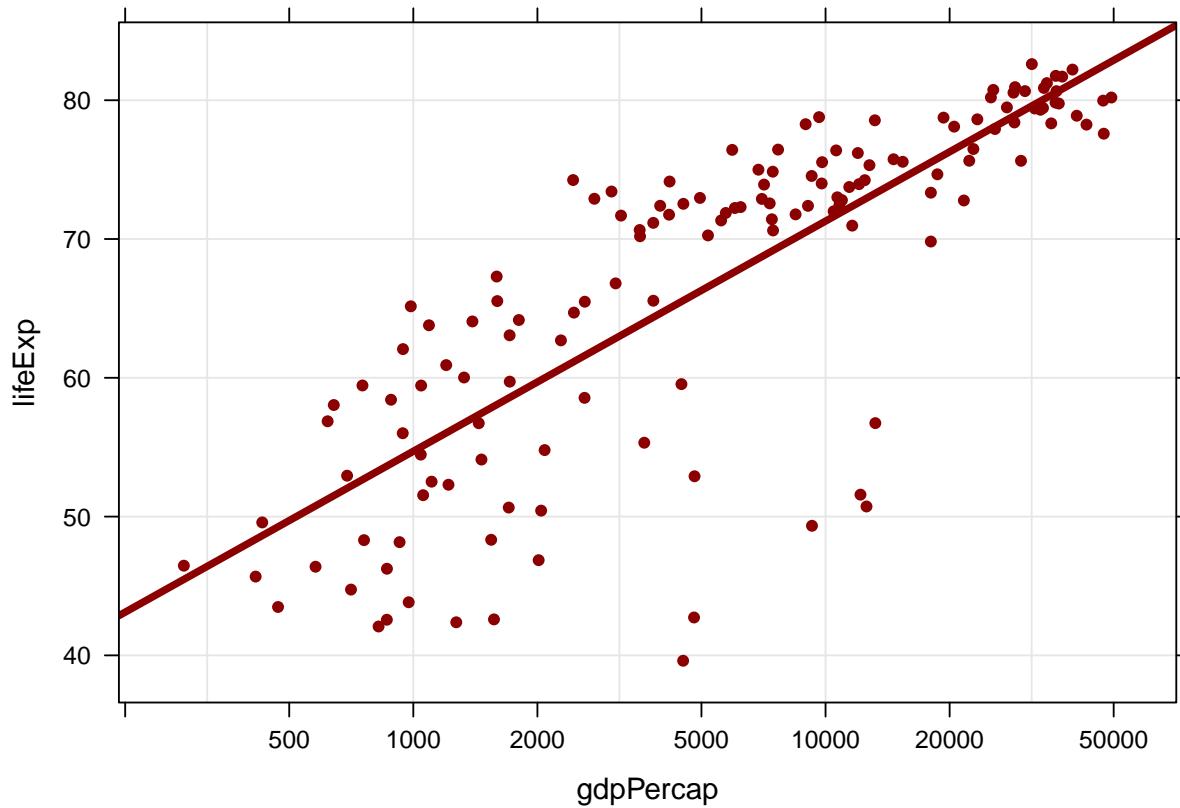
```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       pch=16,
       col="red4",
       grid=T,
       subset = year == "2007",
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = "p")
```



Slika 111: Funkcija **xyplot()**; dijagram raspršenja (engl. *scatterplot*) dviju kvantitativnih varijabli; **type** argument - *equispaced.log* za pravilno skaliranje osi

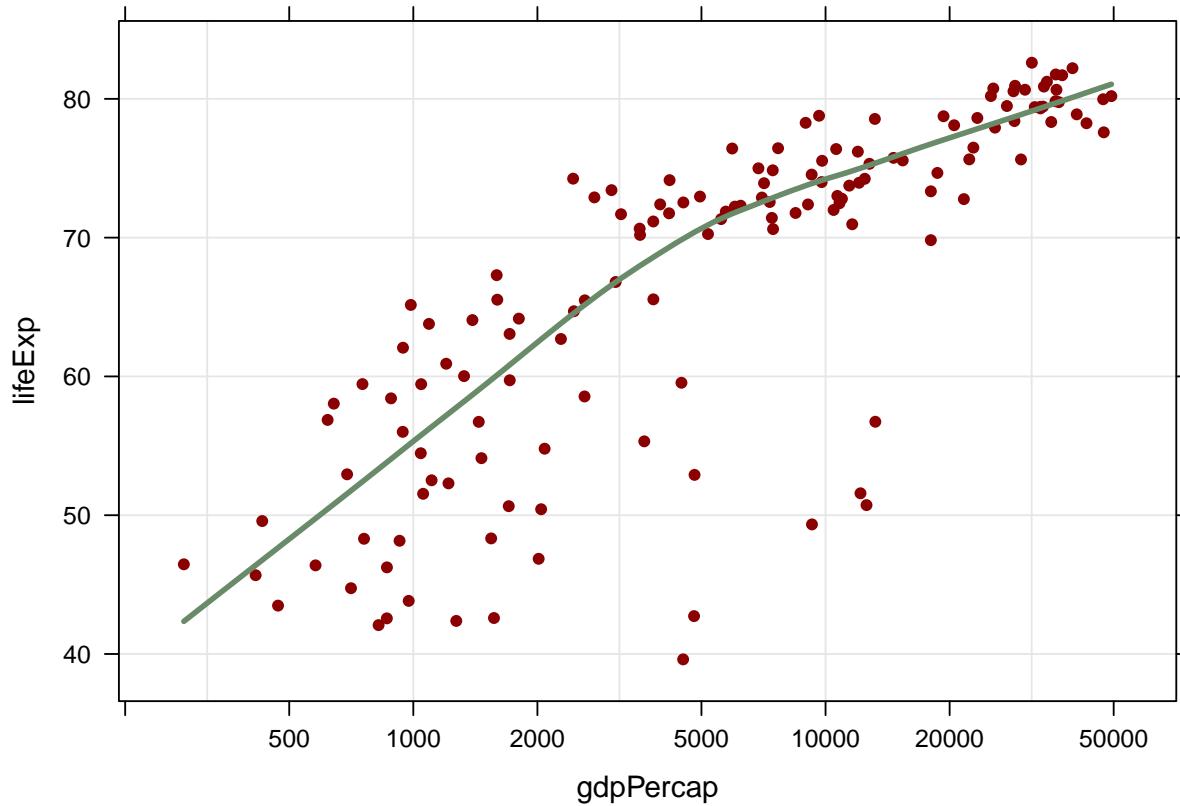
```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       col="red4",
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p", "r"))
```

Dodatno se može grafički prikaz poboljšati dodatnim grafičkim parametrima prema želji, u primjeru koji slijedi mijenjemo izgled regresijske linije.



Slika 112: Funkcija `xyplot()`; dijagram raspršenja (engl. *scatterplot*) dviju kvantitativnih varijabli; `type` argument - p r

```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       col="red4",
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p", "smooth"),
       col.line = "darkseagreen4",
       lwd = 3)
```



Slika 113: Funkcija **xyplot()**; dijagram raspršenja (engl. *scatterplot*) dvije kvantitativne varijable;; type argument - p, smooth; dodatni grafički parametri za kontrolu izgleda regresijske linije

Moguće opcije prilikom crtanjadijagrama raspršenja korištenjem funkcije **xyplot()** su:

- **p** - crtanje točaka
- **l** - spajanje točaka linijom
- **b** - crtanje i točaka i linije koja ih spaja
- **o** - crtanje točaka i linije koja ih preklapa
- **S, s** - crtanje step funkcije
- **h** - linije s ishodišta, slično histogram prikazu

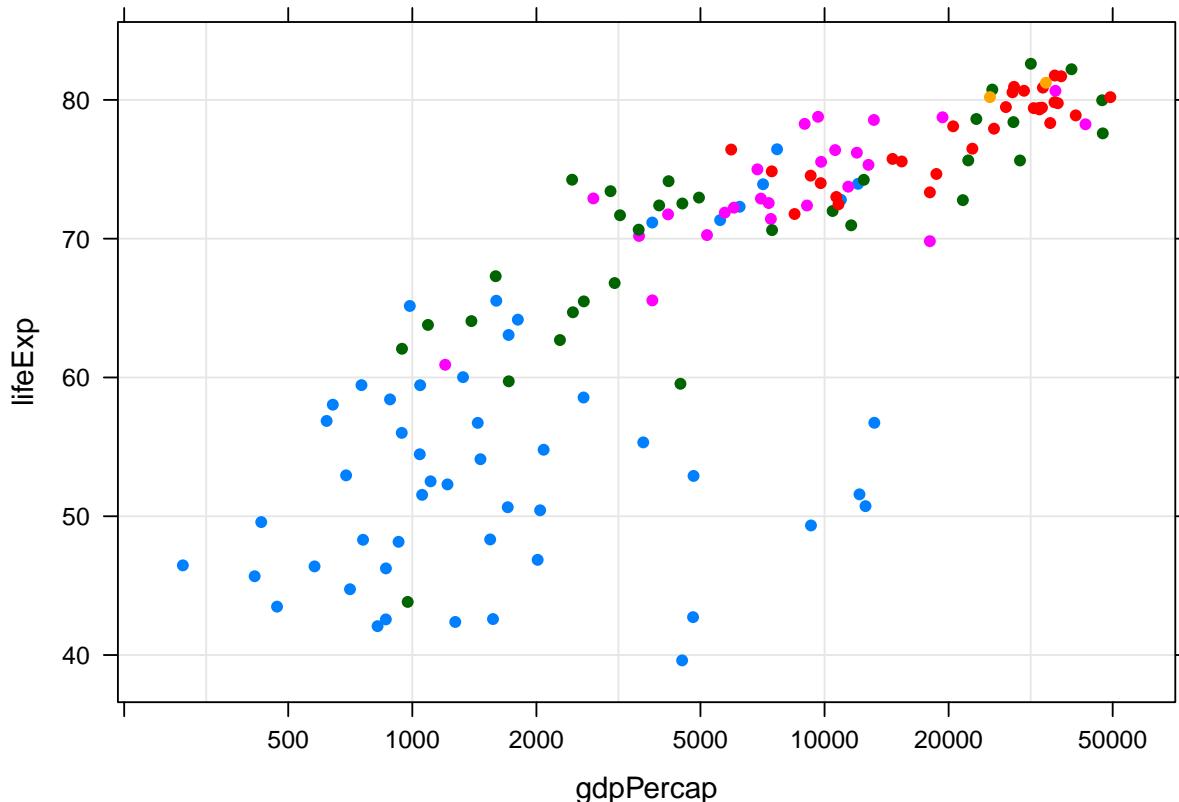
- **a** - spajanje točaka linijom nakon uprosječavanja (panel: **panel.average()**)
- **r** - crtanje regresijske linije (panel: **panel.lmline()**)
- **smooth** - crtanje polinomijalne lokalne kernel funkcije *LOESS smooth* (panel: **panel.loess()**)
- **g** crtanje referentne mreže.

**ZA SAMOSTALAN RAD:** poigrajte se promjenom argumenta **type** na prethodnim primjerima i komentirajte dobiveni rezultat.

#### 4.2.13.3 Argument **group**

U *lattice* grafici moguće je pojedine podskupove podataka predstaviti, grupirati, na način da na panele dodamo novu informaciju: svaku grupu obojimo različito ili prikažemo različitim simbolom. Proučite primjer na našim podacima *lattice\_data*. Ono što definiramo argumentom **group** obično stavimo i u legendu radi snalaženja na grafikonu (**key()** i/ili **auto.key()** argumenti).

```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       group = continent)
```

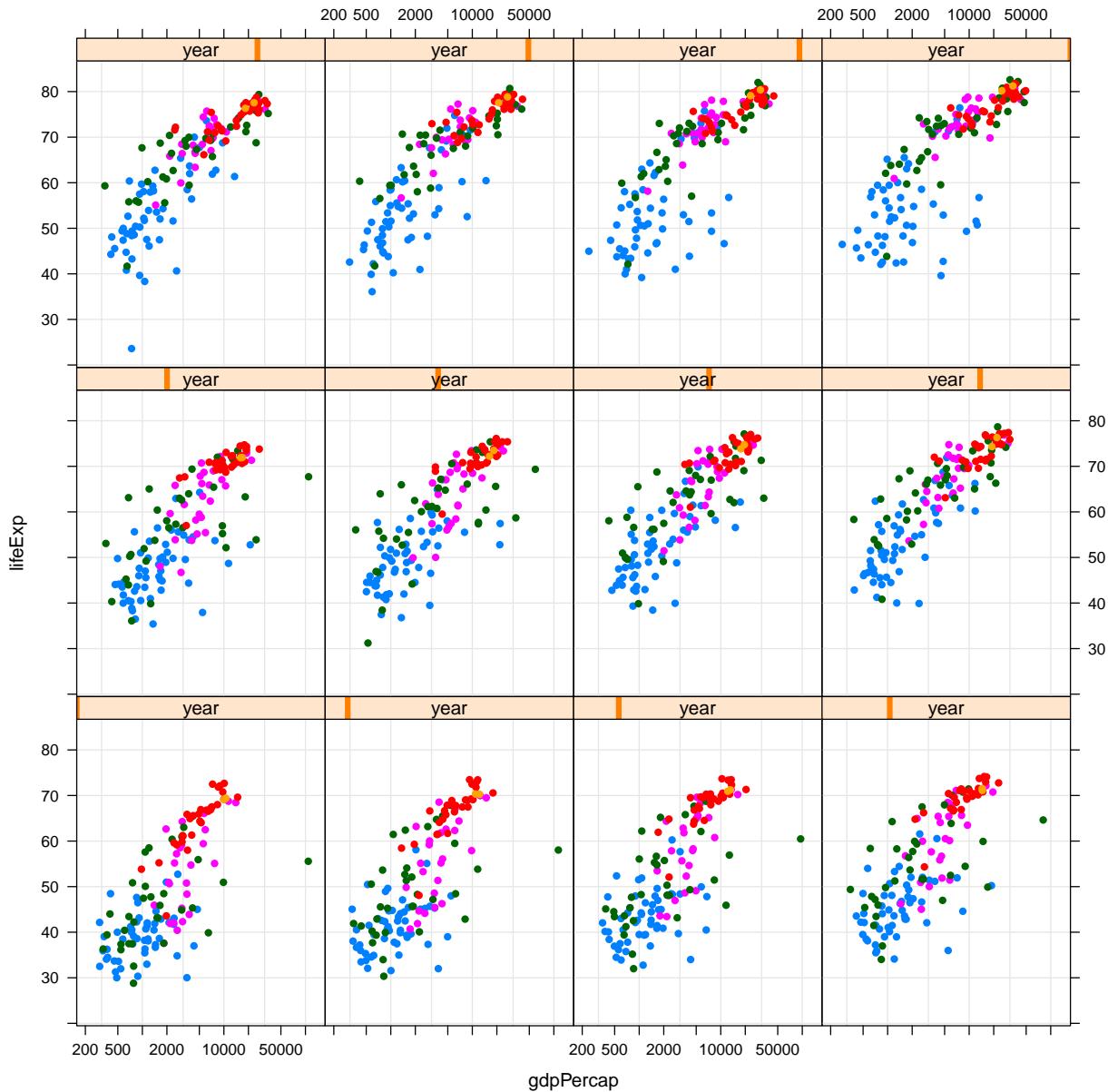


Slika 114: Funkcija **xyplot()**; dijagram raspršenja dvije kvantitativne varijable; argument **group**

U sljedećem primjeru crtamo svaki nivo faktora varijable posebnom bojom, a za vježbu koristimo podatke *lattice\_data* na način

da se podaci iscrtaju za svaku godinu u zasebnom panelu.

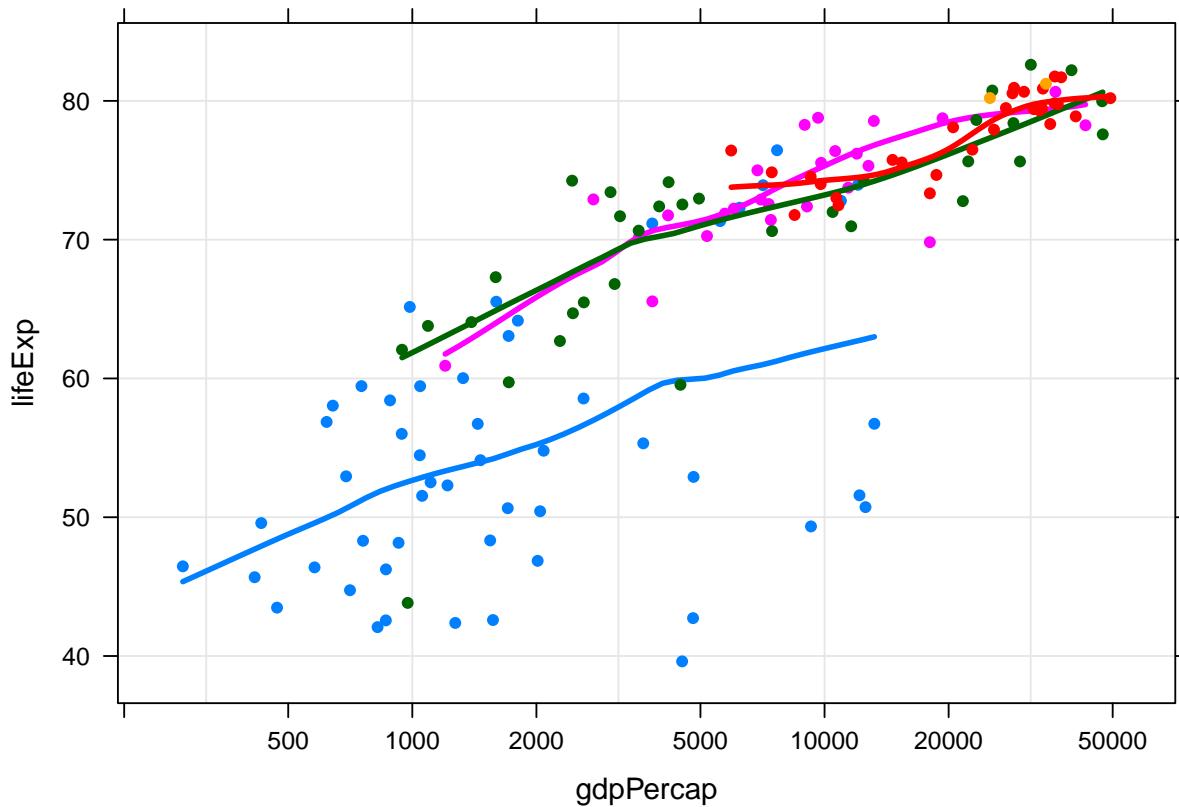
```
xyplot(lifeExp ~ gdpPerCap | year,
       lattice_data,
       #fill.color = lattice_data$continent_color,
       pch=16,
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       group = continent)
```



Slika 115: Funkcija **xyplot()**; dijagram raspršenja dvije kvantitativne varijable; **group** argument

Argumenti **group** i **subset** vrlo su moćno oruđe u vizualizaciji multivarijatnih podataka kada se koriste u kombinaciji. U primjeru koji slijedi fitamo regresijsku liniju za podskupove podataka za svaki kontinent, podaci za jednu godinu.

```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p", "smooth"),
       lwd=3,
       group = continent)
```

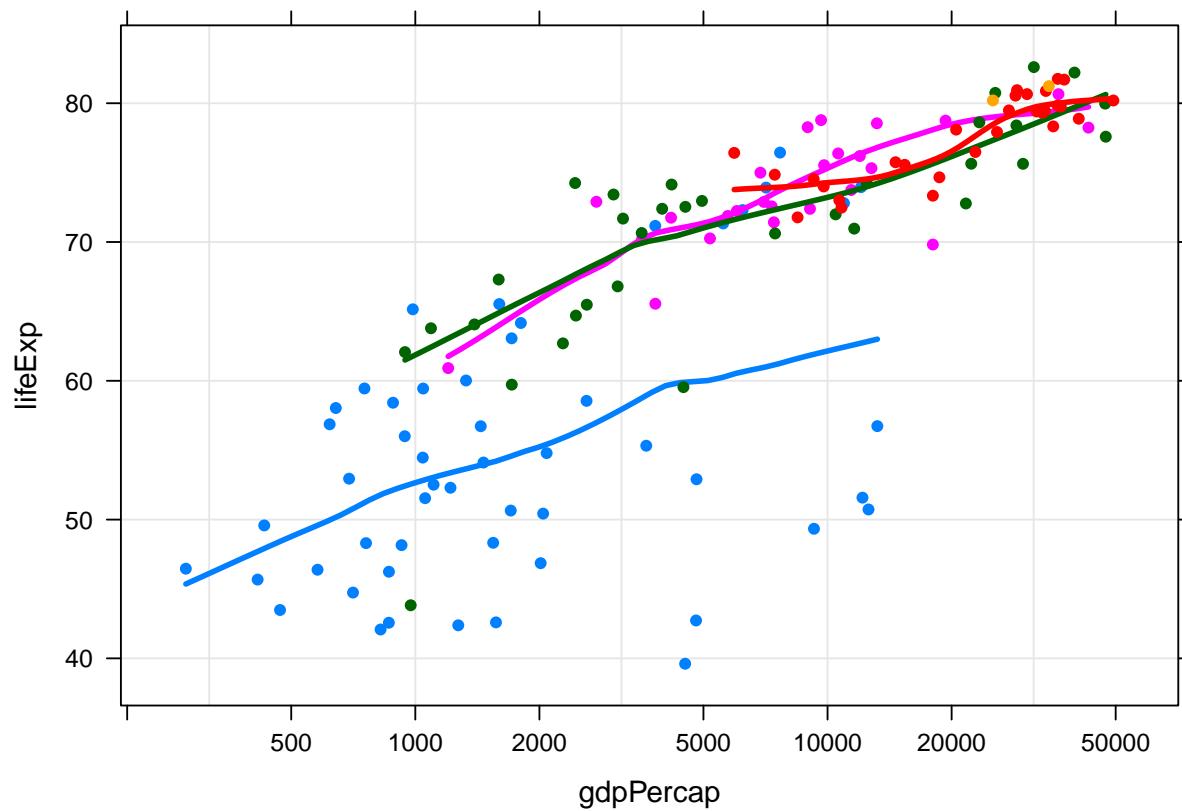


Slika 116: Function `xyplot()`; dijagram raspršenja dvije kvantitativne varijable; `type/group` argument; ; dodatni parametri za regresijsku liniju `smooth`

U primjeru koji slijedi bojamo podatke prema nivoima faktorske varijable za što ćemo ponovno koristiti, poznati nam skup podataka, *iris*. Kao dodatnu funkcionalnost, koja ponekad znatno doprinosi jasnoći grafičkogA prikaza koristimo argument `jitter` koji za zadanu vrijednost razmiče preklapajuće podatke.

```
xyplot(Sepal.Width ~ Sepal.Length,
       group=Species,
       data=iris,
       auto.key=list(space="right"),
       jitter.x=TRUE,
       jitter.y=TRUE)
```

**ZA SAMOSTALAN RAD:**



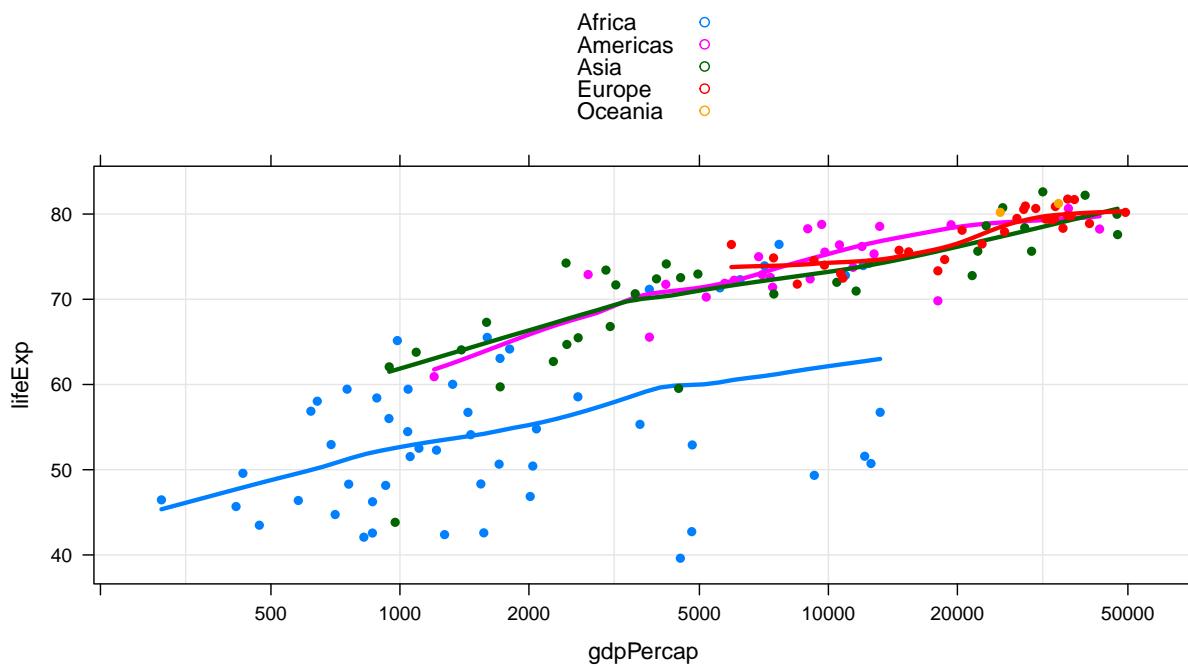
Slika 117: Funkcija `xyplot()` - primjer bojanja faktorske varijable prema nivoima faktora

Na skupu podataka iz sustava R *diamonds* paketa *ggplot2* napravite dijagram raspršenja na način da su podaci obojani prema nivou faktora varijable *cut* ili *clarity*.

#### 4.2.13.4 Argument key/auto.key

Ako grafički prikaz koji radimo u sebi sadržava podskupove objekata, obično definirani argumentom *subset*, argument *key* se može dodati kroz generalnu funkciju prikaza (*general display function*) na grafički prikaz kao pojašnjenje gledatelju. Argument *key* kontrolira izgled i položaj legende dodane grafičkom prikazu dok *auto.key* dodaje legendu automatski. Parametri koje možemo proslijediti argumentu *key* su u formatu liste koja sadrži elemente kao što su *text*, *points* ili *lines* u ovisnosti o informaciji na grafičkom prikazu. Također, proizvoljno je i određivanje položaja legende na prikazu te razmaka putem argumenta *space* i *border* argumenta.

```
xyplot(lifeExp ~ gdpPercap,
       lattice_data,
       subset = year == "2007",
       pch=16,
       grid=T,
       scales = list(x = list(log = 10, equispaced.log = FALSE)),
       type = c("p", "smooth"),
       auto.key=T,
       lwd=3,
       group = continent)
```



Slika 118: Funkcija **xyplot()** - argument **auto.key**

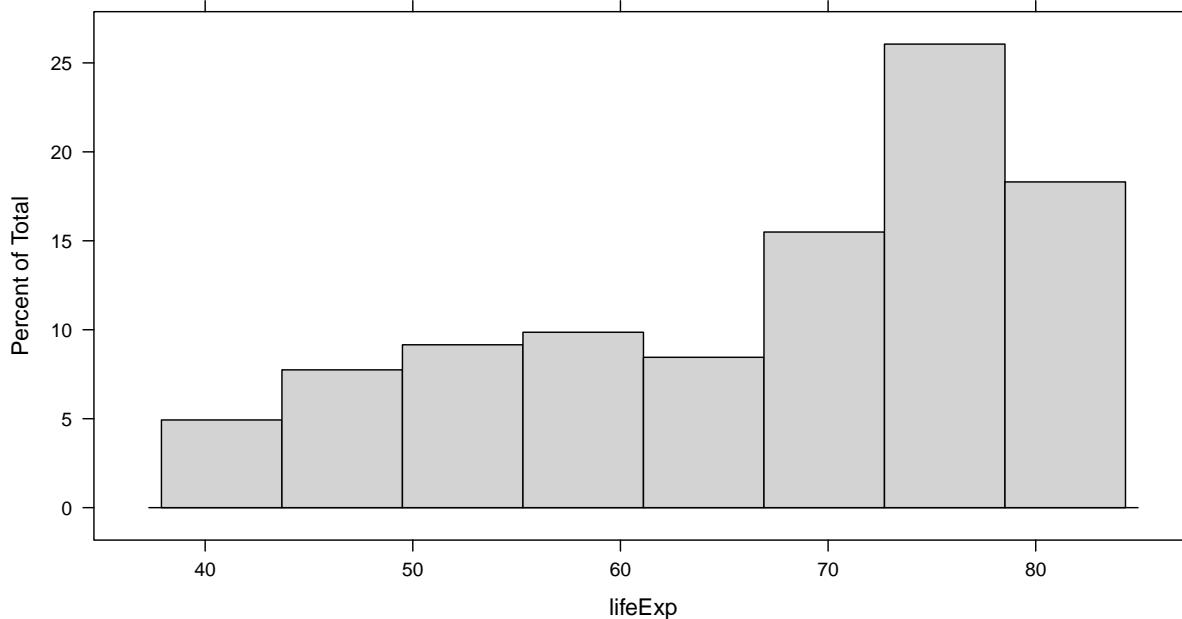
Primjer koji slijedi je primjer redefiniranja panel funkcije *on-the-fly*. Ovo je vrlo čest način kontroliranja grafičkih prikaza unutar *lattice* grafike. Dodatne primjere moguće je pronaći na sljedećoj poveznici: <https://www.r-project.org/conferences/useR-2007/program/presentations/sarkar.pdf>

Od ovoga trenutka grafičke postavke vraćamo na *default* vrijednost.

#### 4.2.13.5 Argument subset

Moguće je izraditi prikaze koji prikazuju samo podskup podataka - podatke koji zadovoljavaju neka logička pravila koja je korisnik definirao, poput primjera koji slijedi gdje je nacrtan samo podskup podataka za 2007. godinu:

```
histogram(~ lifeExp,
          lattice_data,
          subset = year == "2007",
          col="lightgray")
```



Slika 119: Argument **subset** - histogram

Primjetite daje jednak prikaz kao i sljedeće:

```
histogram(~ lifeExp,
          lattice_data[lattice_data$year=="2007",],
          col="lightgray")
```

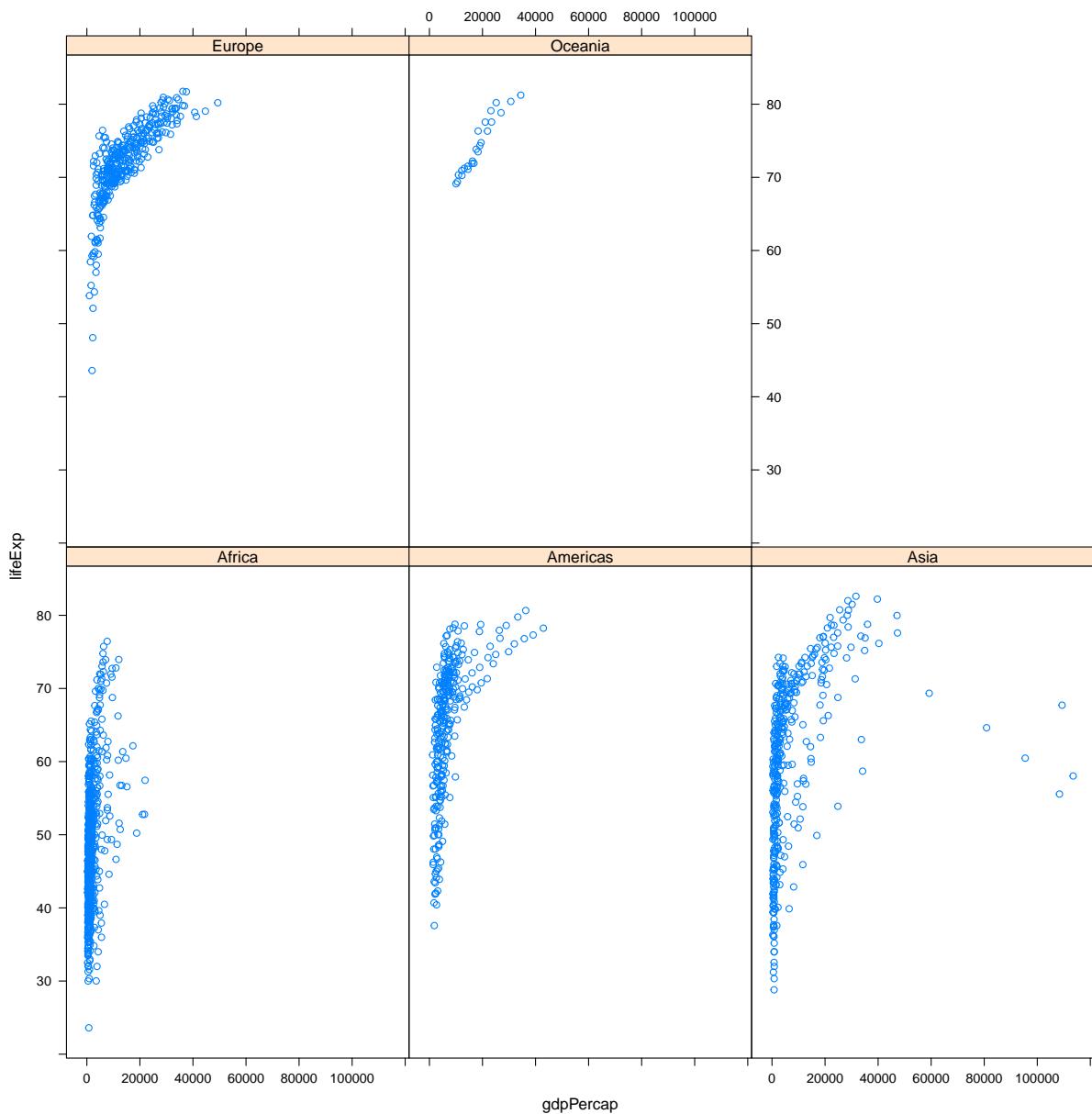
#### 4.2.13.6 Argument *layout*

Predložak panela u *Trellis* prikazu je posebno važan za točno shvaćanje grafičkih informacija. Argument *layout* precizira kako će paneli biti organizirani. Vrijednost ovog argumenta je vektor od dva elementa koji daje broj stupaca i redaka koji će se koristiti u prikazu. Opcionalno, može se dodati i treći element vektora, kojim će se zadati broj strana. Zadano (engl. *default*) je da *lattice* sustav izlaze panele polazeći od donjeg lijevog kuta, krećući se na desno i na gore. Logički argument, *as.table* se može koristiti kako bi se ovo promijenilo, tako da *lattice* počinje od gornjeg lijevog kuta i ide prema desno i prema dolje.

```
trellis.device('pdf', color=FALSE)

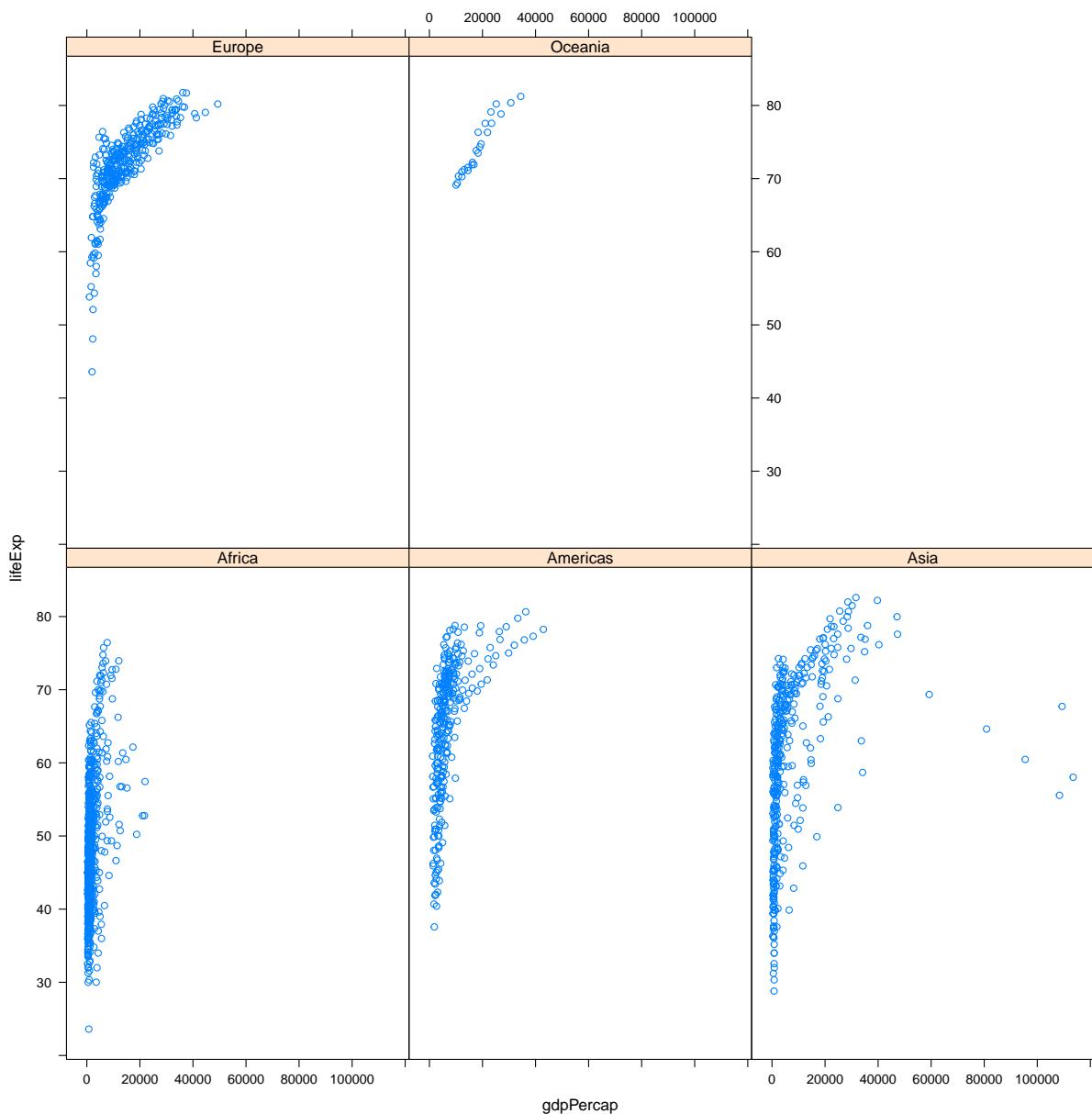
xyplot(lifeExp ~ gdpPercap | continent,
       data=lattice_data)
```

Usporedite s donjim primjerom.



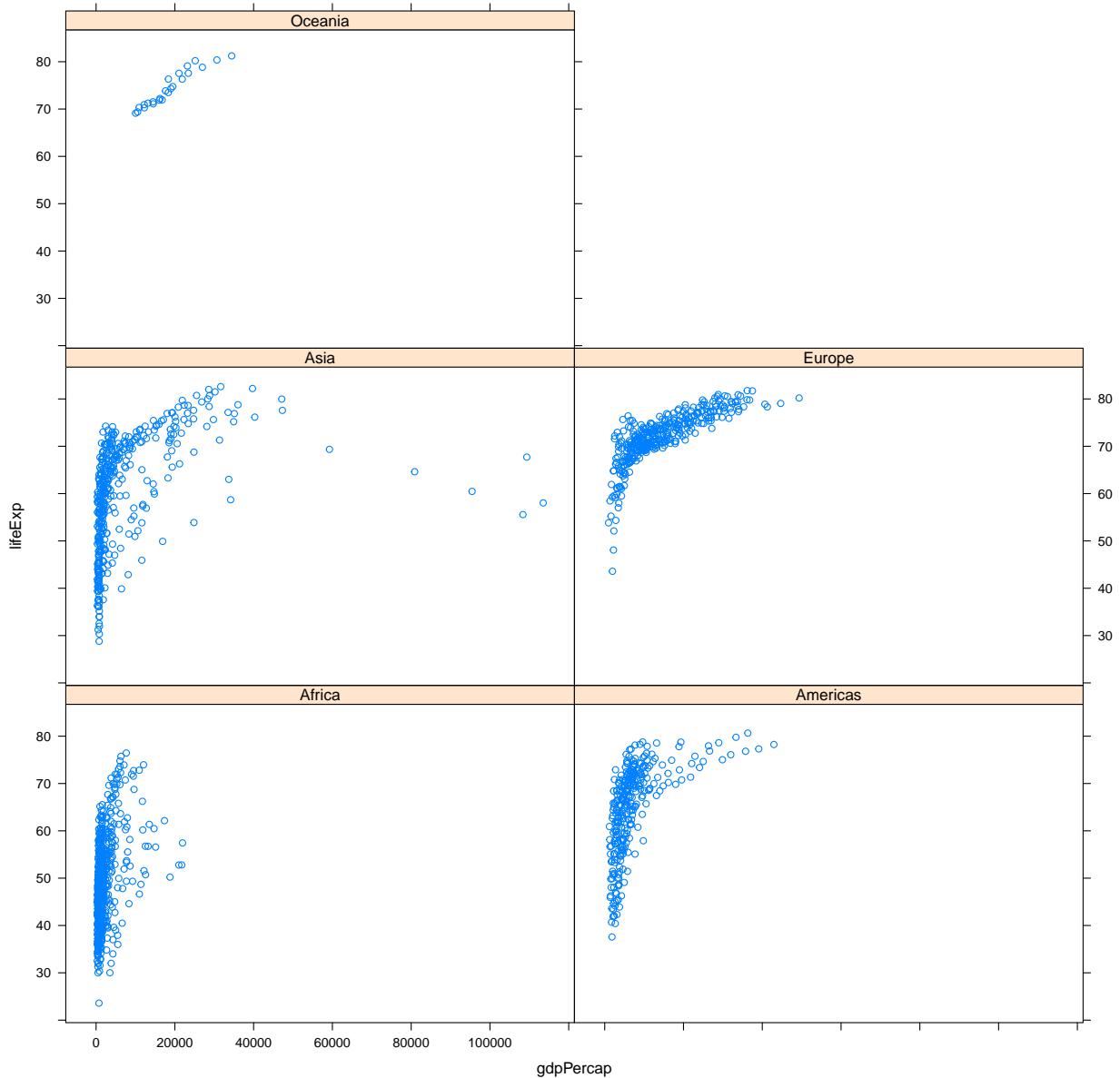
Slika 120: Funkcija **bwplot()** iz paketa *lattice* - Box-Whisker plot s uvjetovanjem; argument *layout default*

```
xyplot( lifeExp ~ gdpPercap | continent,  
       data=lattice_data, layout = c(3,2))
```



Slika 121: Funkcija **bwplot()** iz paketa *lattice* - Box-Whisker plot s uvjetovanjem; novi *layout* argument

```
xyplot( lifeExp ~ gdpPercap | continent,  
       data=lattice_data,  
       layout = c(2,3))
```



Slika 122: Funkcija **bwplot()** iz paketa *lattice* - Box-Whisker plot s uvjetovanjem; argument *layout*

#### 4.2.14 Najznačajnije funkcije visoke razine (*high-level*) iz paketa *lattice* - nastavak

##### 4.2.14.1 Function `histogram()`(*high-level*)`{lattice}`

Kao i obično, prije primjene nove funkcije, unesite `?histogram` (ili `?lattice::histogram`) u R konzolu kako bi se otvorila informacijska kartica o funkciji. Na drugom grafu je simulirani primjer, postavljeni su histogram i Kernel gustoće (engl. *kernel density*).

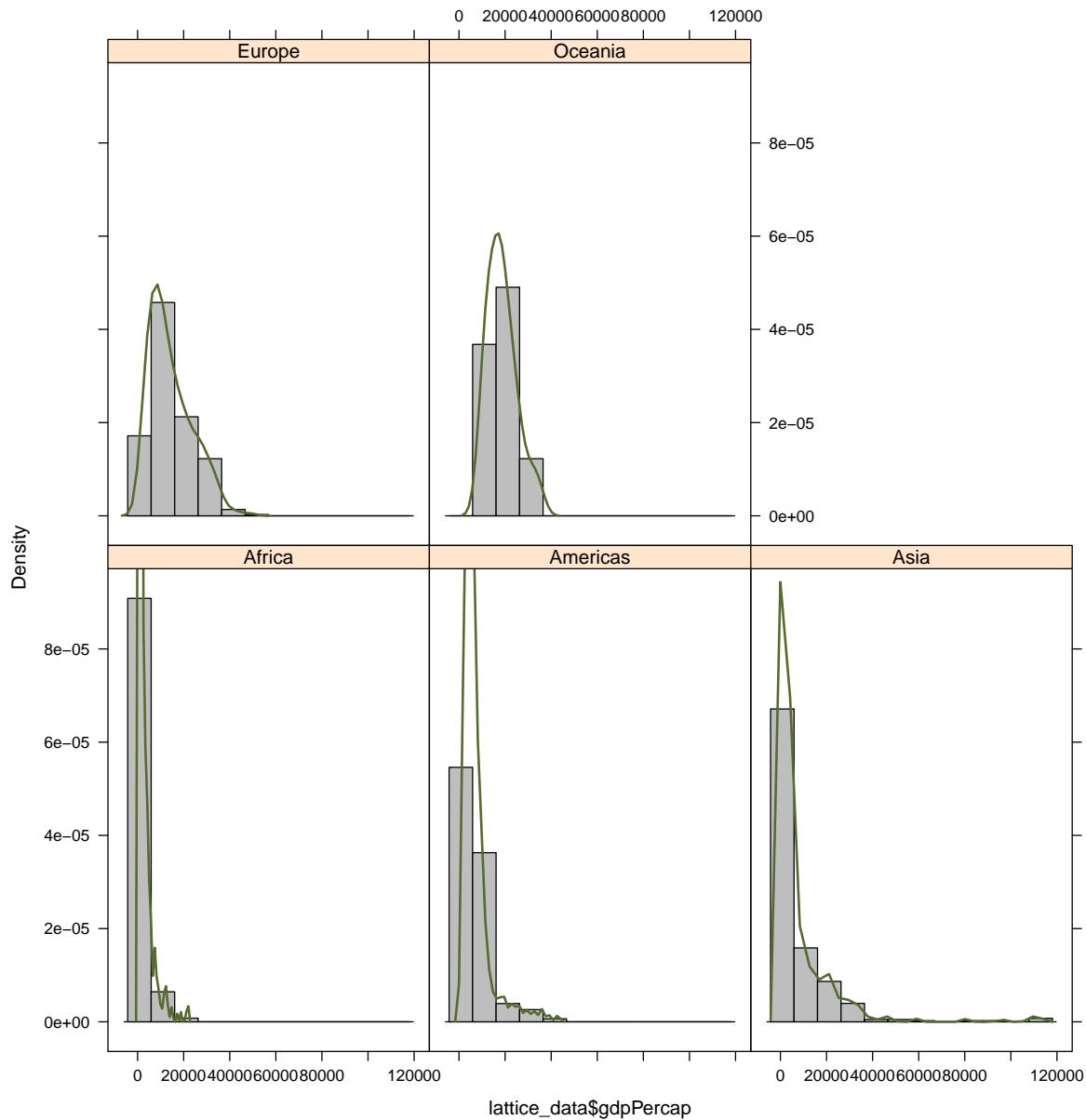
```
trellis.device('pdf', color=FALSE)
```

```
#trellis postavlja za otvoreni grafički uređaj - RStudioGD  
trellis.par.get(old_theme)
```

```
Warning in if (name %in% names(lattice.theme[[.Device]]))  
lattice.theme[[.Device]][[name]] else NULL: the condition has length > 1  
and only the first element will be used
```

```
NULL
```

```
#izradivanje dijagrama  
histogram(~lattice_data$gdpPercap | lattice_data$continent, data=lattice_data,  
          type = "density",  
          panel = function(x, ...) {  
            panel.histogram(x, col = "gray", ...)  
            panel.densityplot(x, col = "DarkOliveGreen", lwd=2, plot.points = FALSE)  
          })
```



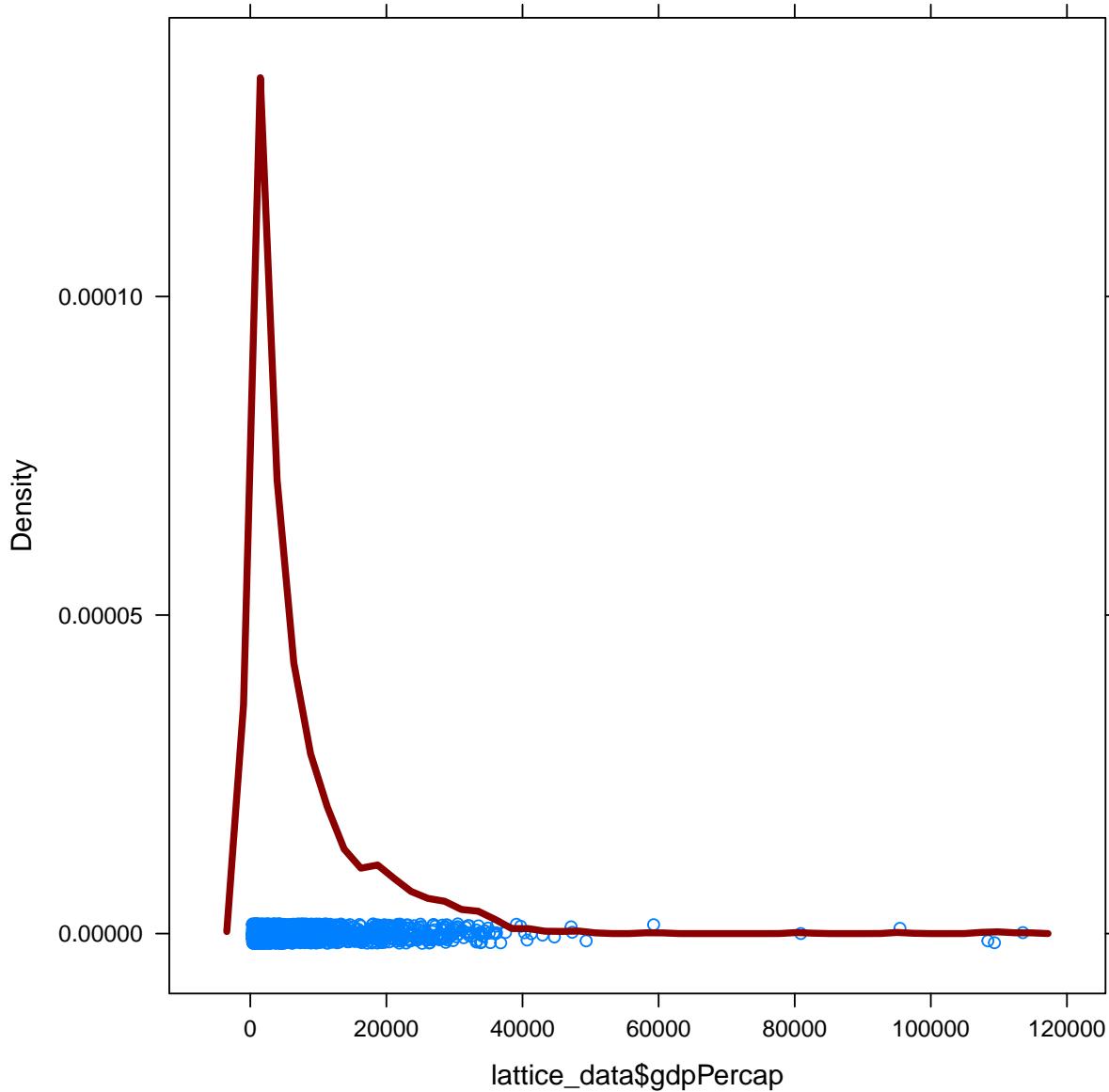
Slika 123: Histogram napravljen funkcijom paketa *lattice*; modifikacijom odgovarajuće panel funkcije; **panel.histogram()**

#### 4.2.14.2 Funkcija `densityplot()(high-level){lattice}`

Funkcija konstruira i prikazuje na grafu neparametarske procjene gustoće, mogućno uvjetovane faktorom. Primjer o gustoći potpune varijable `mpg` iz `mtcars` skupa podataka kao i pripadajuće razine faktora varijable `gear`.

Funkcija crta Kernel density prikaz. U jednostavnom primjeru koji slijedi, histogram i kernel gustoće su procijenjene i preklapljene/superponirane:

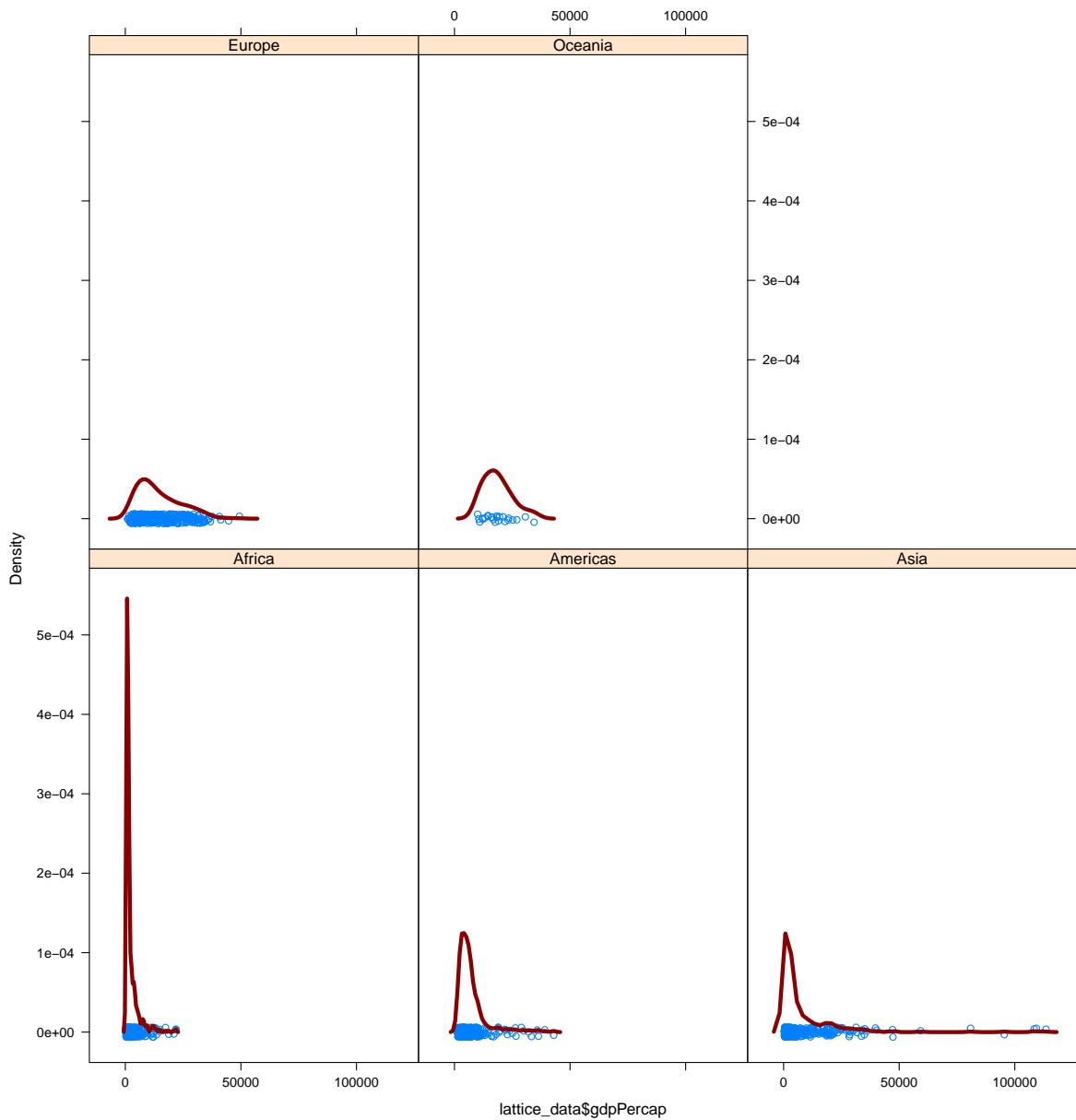
```
densityplot(~ lattice_data$gdpPerCap)
```



Slika 124: Kernal gustoće s paketom lattice, funkcija `densityplot()`

Kako će se kasnije objasnitи, lattice je izvrstan sustav za prikaz dijela podataka na odvojenim panelima, gdje je neka druga varijabla (u slučaju koji slijedi cyl je konstanta):

```
densityplot(~ lattice_data$gdpPercap | lattice_data$continent)
```



Slika 125: Kernal gustoće paketom *lattice*, funkcija **densityplot()**; uvjetovanje po nivoima varijable *continent*

**ZA SAMOSTALAN RAD:** Napravite graf gustoće jedne varijable *mpg* za svaki nivo faktora varijable *gear* za skup podataka *mtcars*.

#### 4.2.14.3 Funkcija `dotplot()`(*high-level*)`{lattice}`

Funkcija crtanja Cleveland točkastog dijagrama. U primjeru koji slijedi koristit će se podaci o prioritetima državnih politika iz 1992. godine.

```
#učitavanje podataka
policy <- read.table("policy.txt", header = T)

#upoznavanje s podacima
str(policy)

'data.frame': 50 obs. of 4 variables:
 $ state.abb: Factor w/ 50 levels "AK","AL","AR",...: 19 30 33 7 38 39 22 5 42 21 ...
 $ priority : num 0.497 0.501 0.504 0.51 0.512 ...
 $ region   : Factor w/ 4 levels "Midwest","Northeast",...: 2 2 2 2 2 2 1 4 3 2 ...
 $ state     : Factor w/ 50 levels "Alabama","Alaska",...: 21 29 32 7 38 39 22 5 42 19 ...

#kopiranje u novi objekt za sortiranje (redanje/poredak)
policy_o <- policy

#poredani skup podataka prema varijabli priority
policy_o$state <- reorder(policy_o$state, policy_o$priority)

#crtamo neporedane podatke
p1 <- dotplot(state ~ priority, data = policy,
               aspect = 1.5,
               xlab = "Neporedani podaci",
               scales = list(cex = .6),
               panel = function (x, y) {
                 panel.abline(h = as.numeric(y), col = "gray", lty = 1)
                 panel.xyplot(x, as.numeric(y), col = "black", pch = 21)})

#plot ordered data
p2 <- dotplot(state ~ priority, data = policy_o,
               aspect = 1.5,
               xlab = "Podaci poredani po prioritetu",
               scales = list(cex = .6),
               panel = function (x, y) {
                 panel.abline(h = as.numeric(y), col = "gray", lty = 3)
                 panel.xyplot(x, as.numeric(y), col = "black", pch = 16)})
```

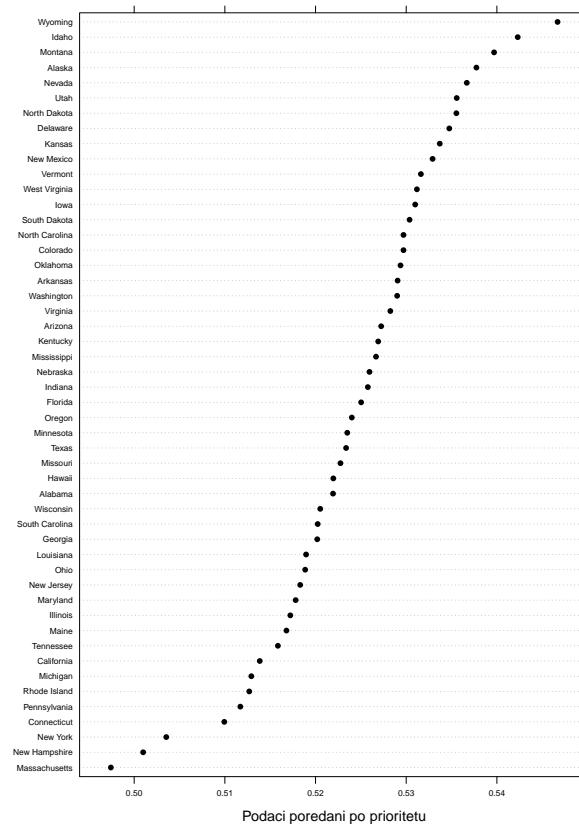
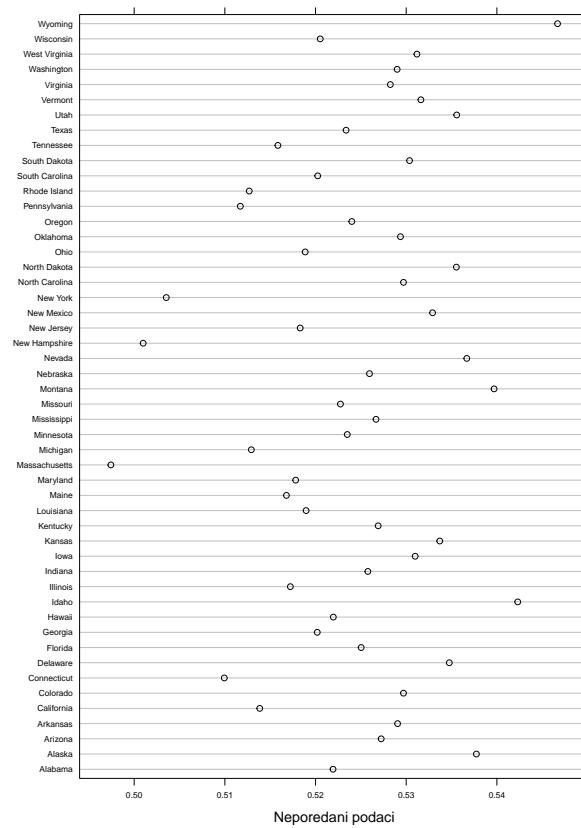
Nakon što smo pripremili poredane podatke, možemo ih nacrtati pomoću funkcije `dotplot()`:

Sljedeći primjer korištenja funkcije `dotplot()` napraviti ćemo na skupu podataka spremljenom na disku računala, “***spending.txt***” koji nosi informacije o potrošnji države za pojedinog studenta s nekoliko dodatnih varijabli kao što su država (`state`) i godina (`year`).

```
#učitavanje podataka
spending <- read.table("spending.txt", header = T)

#upoznavanje s podacima
str(spending)

'data.frame': 50 obs. of 4 variables:
 $ year       : int 2000 2000 2000 2000 2000 2000 2000 2000 2000 ...
 $ state.abb  : Factor w/ 50 levels "AK","AL","AR",...: 19 9 41 42 14 21 38 20 24 32 ...
```



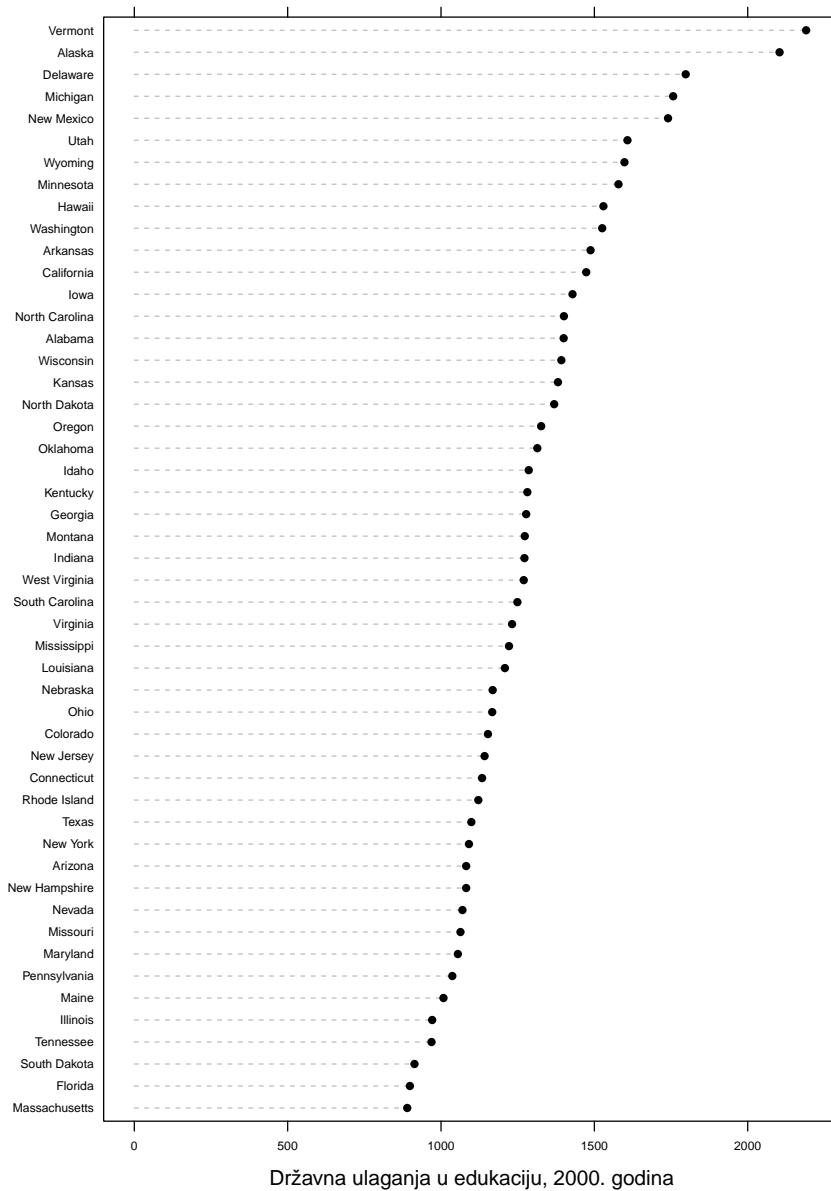
Slika 126: Točkasti dijagram za a) neporedane i b) poredane podatke uz dodatnu prilagodbu; lattice paket, podaci o politikama srce 165

```
$ educ.per.cap: num  890 899 913 969 971 ...
$ state      : Factor w/ 50 levels "Alabama","Alaska",...
#sortiramo (poredamo) podatke prema educ.per.cap
spending$state <- reorder(spending$state, spending$educ.per.cap)

dotplot(state ~ educ.per.cap, data = spending,
        aspect = 1.5,
        scales = list(cex = .6),
        xlim = c(-100, 2300),
        xlab = "Državna ulaganja u edukaciju, 2000. godina",
        panel = function (x, y) {
          panel.segments(rep(0, length(x)), as.numeric(y),
                         x, as.numeric(y), lty = 2, col = "gray")
          panel.xyplot(x, as.numeric(y), pch = 16, col = "black")})
```

#### ZA SAMOSTALNI RAD

- 1) Kreirajte Cleveland *dotplot* na objektu *lattice\_data* na način da je:
  - varijabla koja nas interesira *gdppercap*
  - uvjetujuća varijabla *continent* i
  - varijabla skupine je *year*.
- 2) Kreirajte Box-Whisker dijagram na objektu *bugs\_data* na način da je:
  - varijabla koja nas interesira *COUNT*
  - uvjetujuća varijabla je *A\_L\_P* i
  - varijabla skupine je *POS*
  - postavite automatiziranu legendu s desne strane.

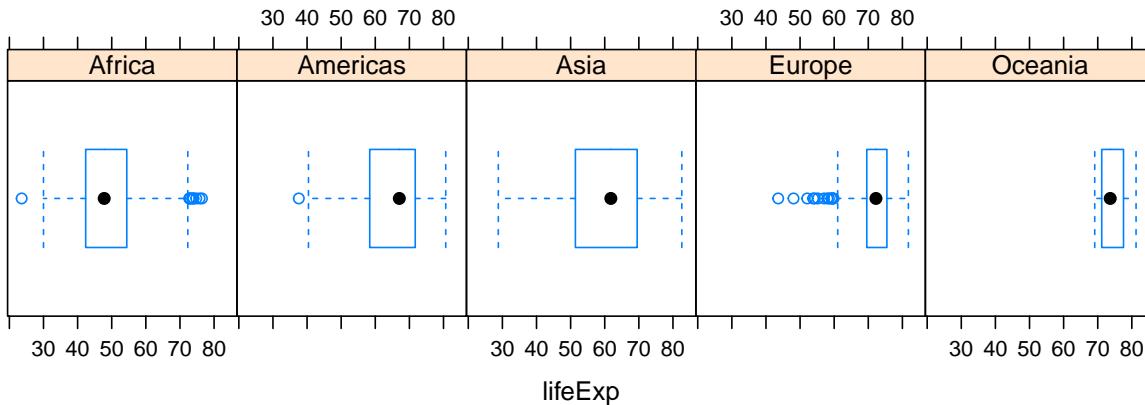
Slika 127: Cleveland dotplot 3; *lattice* paket

#### 4.2.14.4 Funkcija `bwplot()` (*high-level*) {*lattice*}

Lattice funkcija visoke razine za izradu Box-Whisker dijagrama.

```
trellis.device("pdf", color=F)
trellis.device(retain=T)

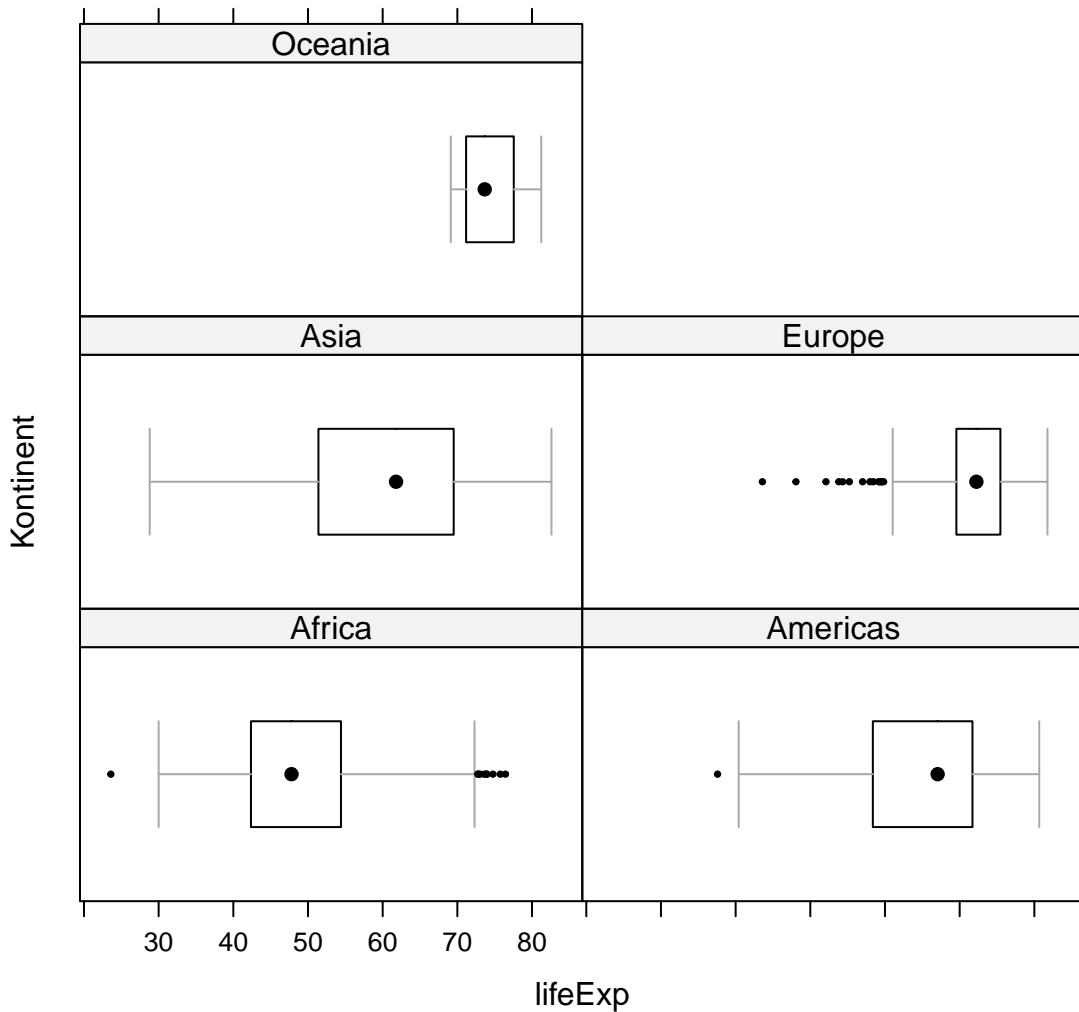
bwplot(~ lifeExp | continent, data=lattice_data)
```



Slika 128: Funkcija `bwplot()` *lattice* paket - primjer Box-Whisker prikaz s uvjetovanom varijablom

```
bwplot(~ lifeExp | continent, data = lattice_data,
       layout = c(2,3), stack = F, drop.unused.levels = F,
       auto.key = list(space = "right"),
       main="Očekivano trajanje života / godina / kontinent",
       ylab = "Kontinent",
       par.settings = list(
         plot.symbol = list(col=c("black"),pch=c(20), cex=0.5),
         box.umbrella=list(col= c("dark gray"),lwd=1, lty=1),
         box.dot=list(col= c("black"), cex=0.9, pch=16),
         box.rectangle = list(col= c("black")),
         box.rectangle=list(lwd=2)))
```

## Ocekivano trajanje života / godina / kontinent

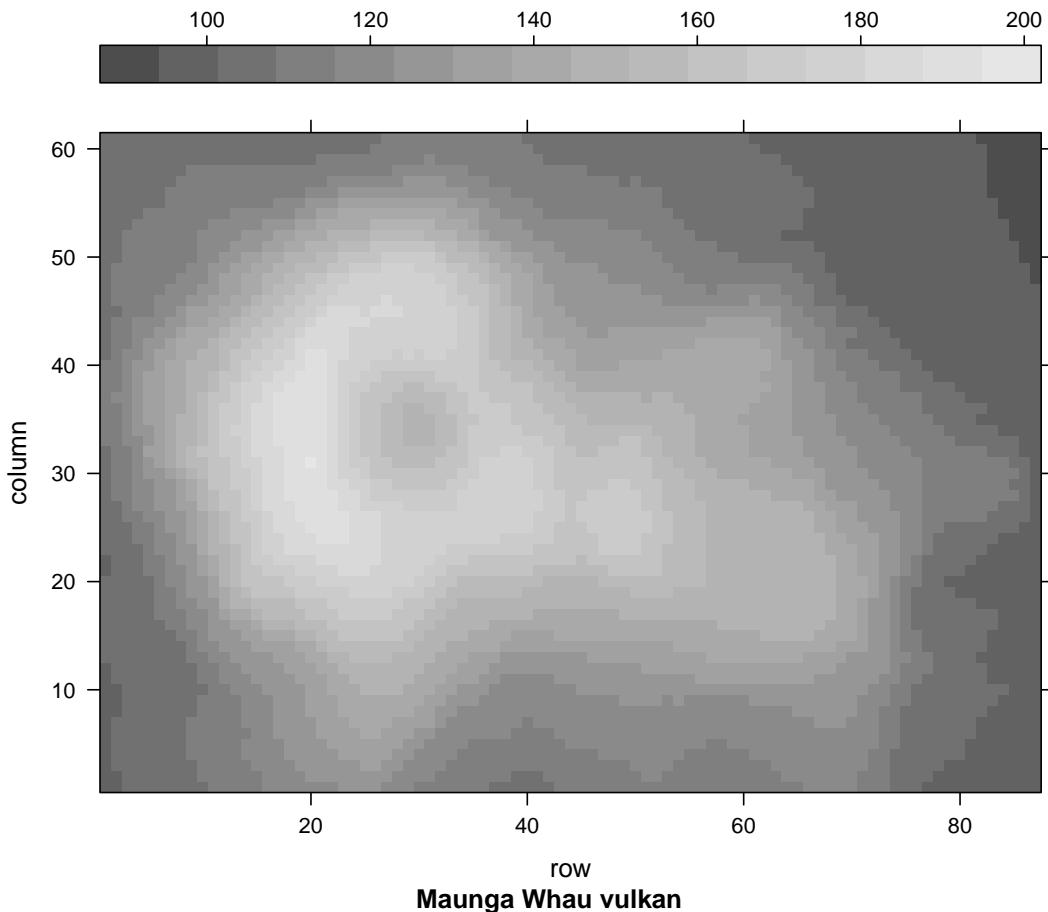


Slika 129: Box-Whisker dijagram; paket lattice; grafički parametri putem *general display function*

#### 4.2.14.5 Function `levelplot()`{lattice}

Lattice grafički paket sadrži funkciju `levelplot()` za ovu vrstu grafičkoga prikaza. Funkcija koristi gradijent boje kako bi se pokazale varijacije jedne varijable, po rasponima druge dvije. Raspon boja koje se koriste u lattice level plot-u može se precizirati kao vektor boja u `col.regions` argumentu funkcije. Koristimo funkciju `terrain.colors` kako bi kreirali ovaj vektor čiji je spektar od 100 boja manje upečatljiv od onih koje su prethodno korištene u osnovnoj grafici.

Već poznajemo podatke `volcano`. U primjerima koji slijede producirat ćemo `levelplot` za matricu `volcano`:



Slika 130: Levelplot matrice `volcano` iz sustava R; `lattice` paket

U primjeru koji slijedi koristimo podatke pohranjene u objektu `elevation` kako bismo nacrtali graf pomoću `lattice` grafike.

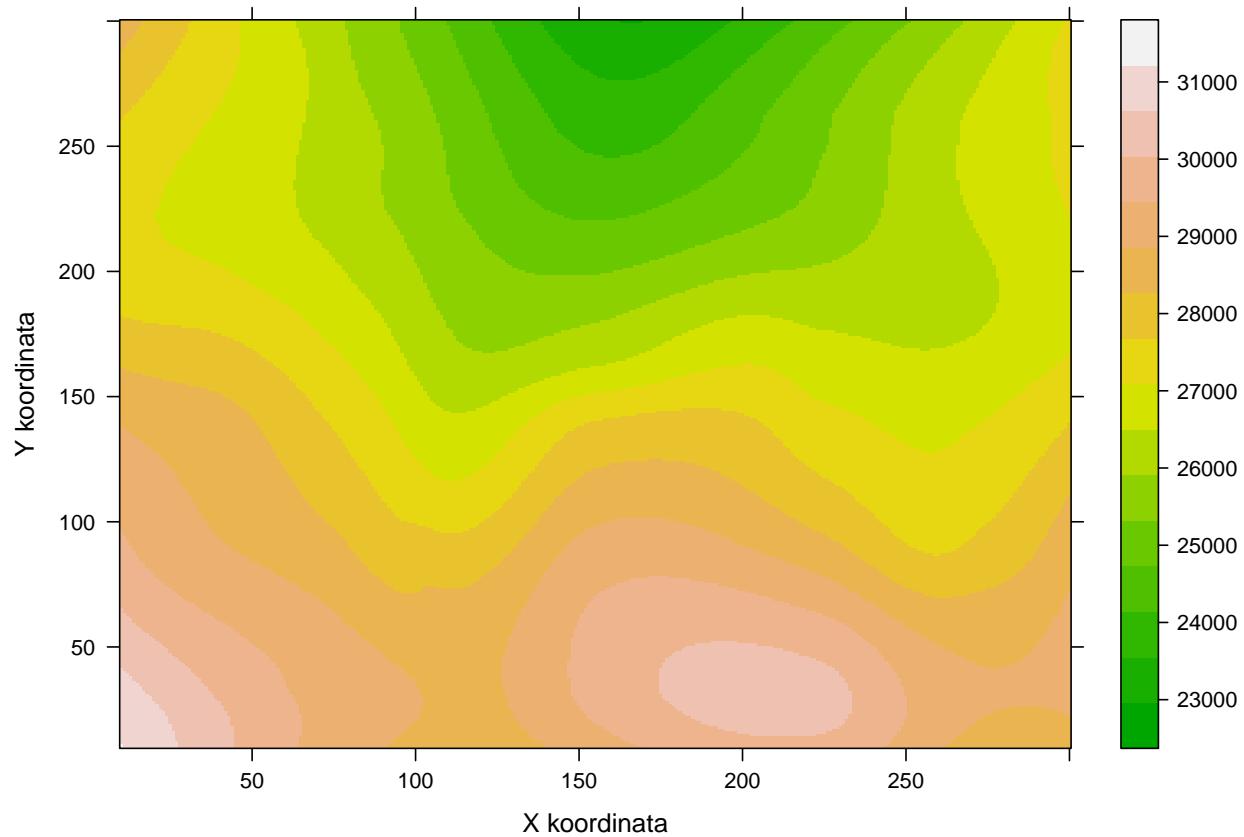
Pogledajmo strukturu podataka koje bismo željeli vizualizirati:

```
str(elevation.fit)
```

```
'data.frame': 84681 obs. of 3 variables:
 $ x      : num 10 11 12 13 14 15 16 17 18 19 ...
 $ y      : num 10 10 10 10 10 10 10 10 10 10 ...
 $ Height: num 31225 31191 31157 31123 31089 ...
 - attr(*, "out.attrs")=List of 2
   ..$ dim      : Named int 291 291
   ... ..- attr(*, "names")= chr "x" "y"
   ..$ dimnames:List of 2
   ... ..$ x: chr "x= 10" "x= 11" "x= 12" "x= 13" ...
```

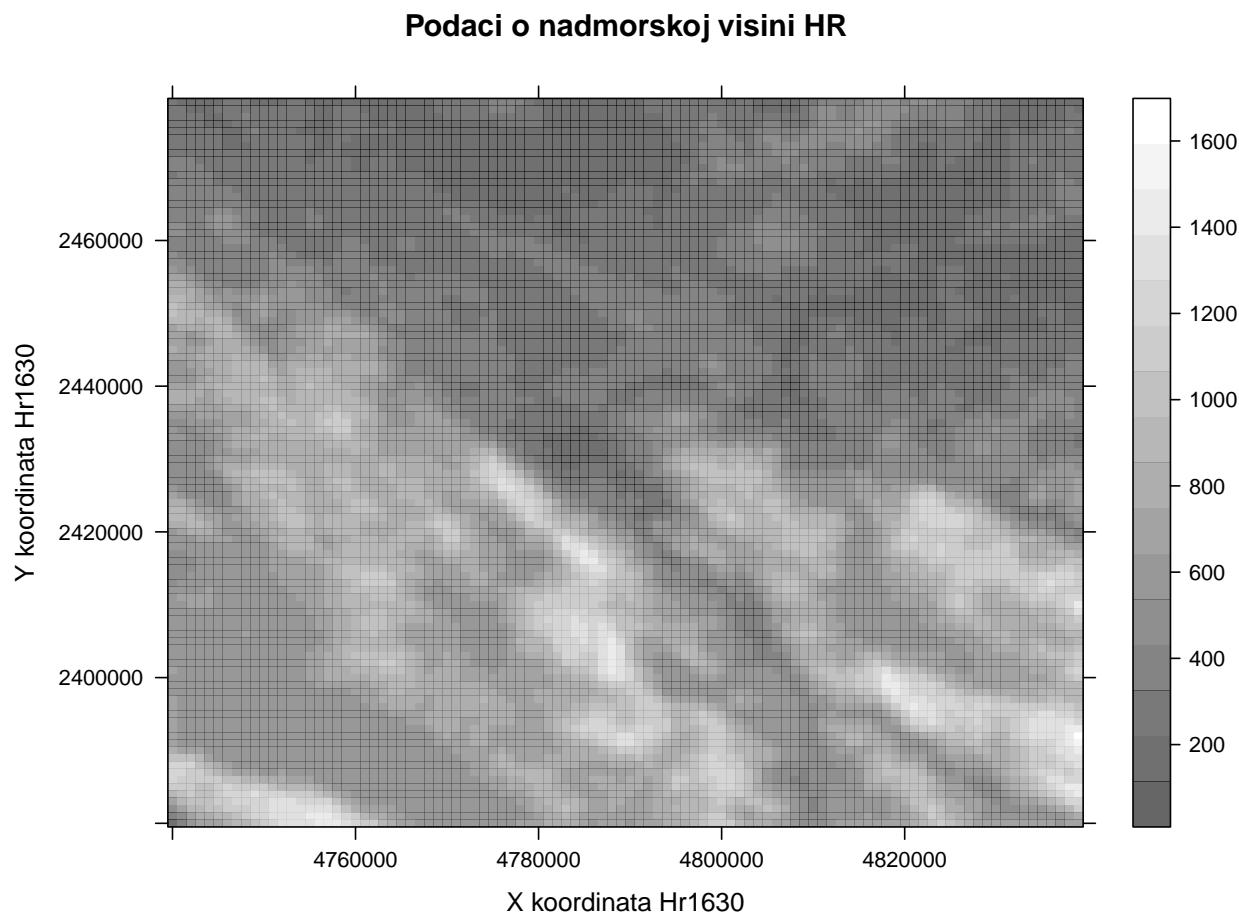
```
... .$. y: chr  "y= 10" "y= 11" "y= 12" "y= 13" ...
```

### Podaci o nadmorskoj visini



Slika 131: Levelplot podataka **elevation**; lattice paket

I na kraju jedan primjer vizualizacije nadmorske visine područja u Hrvatskoj, Lika i dio Velebita, siva paleta:



Slika 132: Levelplot nadmorske visine dijela Hrvatske, stara tzv. 2.5 projekcija

#### 4.2.14.6 Funkcija `wireframe(){lattice}`

Funkcija `wireframe()` crta graf trodimenzionalne perspektive; dijagram kontinuiranih varijabli u tri dimenzije.

Pošto su prikazi površine učinkoviti samo kada se podaci redovno prikupljaju na grid-u, ova funkcija pruža rezultat koji je sličan `persp()` funkciji iz osnovne grafike. Isti skup podataka ćemo vizualizirati na prikazu iz određene perspektive, u osnovnoj grafici, na skupu podataka `volcano`.

```
#koordinate skupa podataka volcano
dim(volcano)
```

```
## [1] 87 61
```

```
x <- 1:87
```

```
#x <- 1:dim(volcano)[1]
```

```
y<- 1:61
```

```
#y <- 1:dim(volcano)[2]
```

```
x
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69
## [70] 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87
```

```
y
```

```
## [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
```

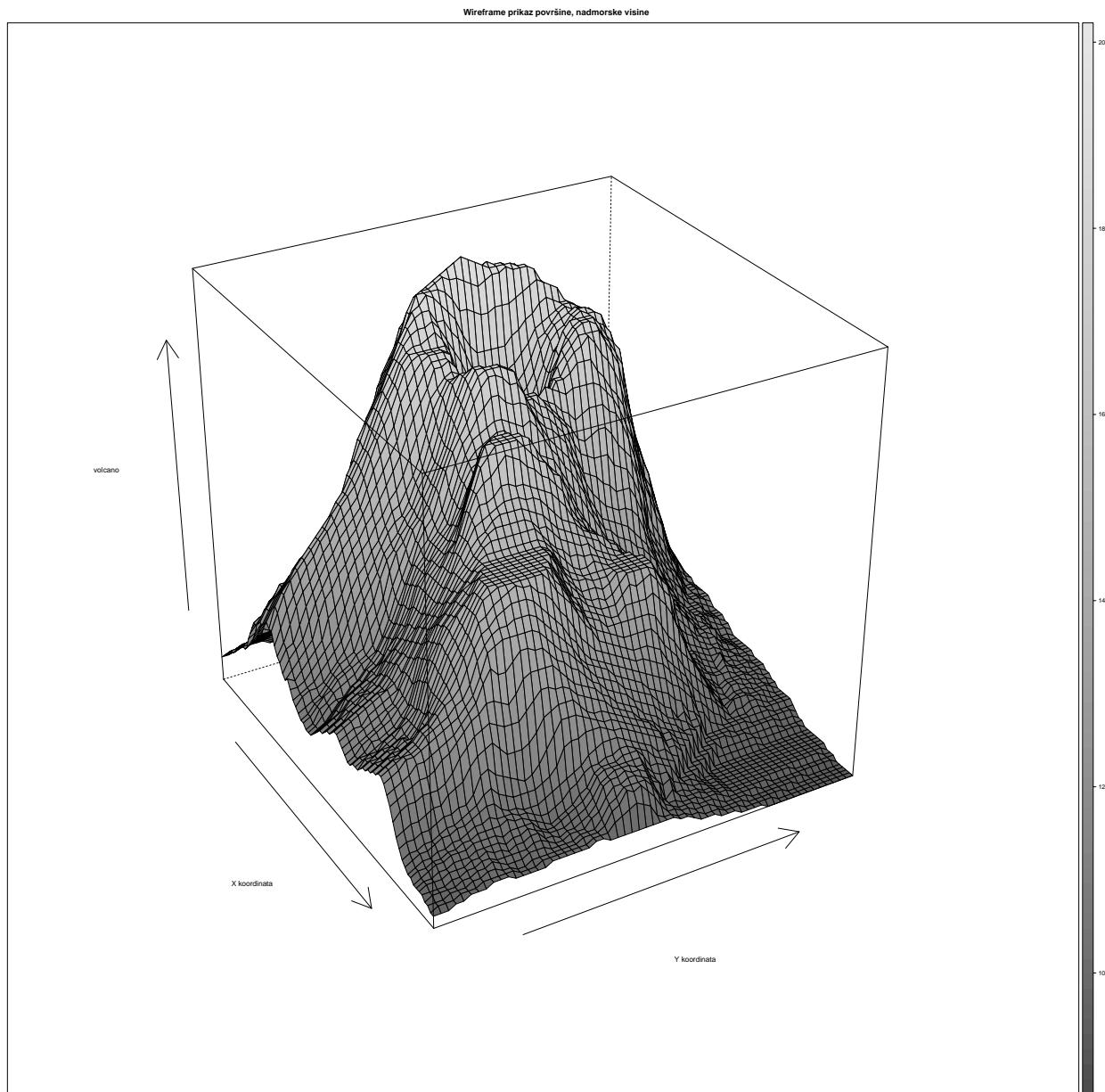
```
xy <- expand.grid(x,y)
```

```
names(xy) <- c("x", "y")
```

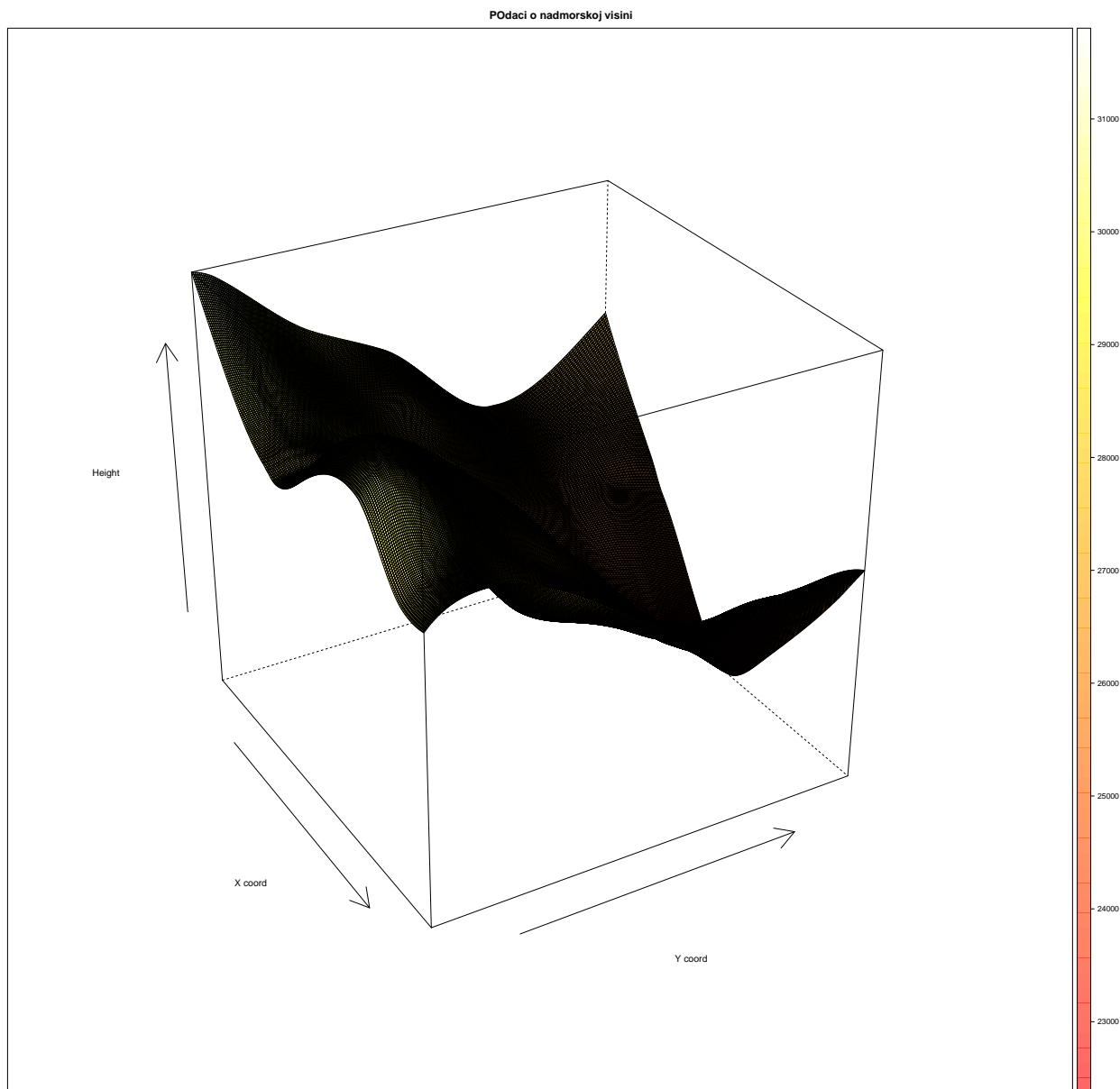
```
volcano_data <- data.frame(volcano, xy)
```

#### 4.2.14.7 Function `cloud()` {lattice}

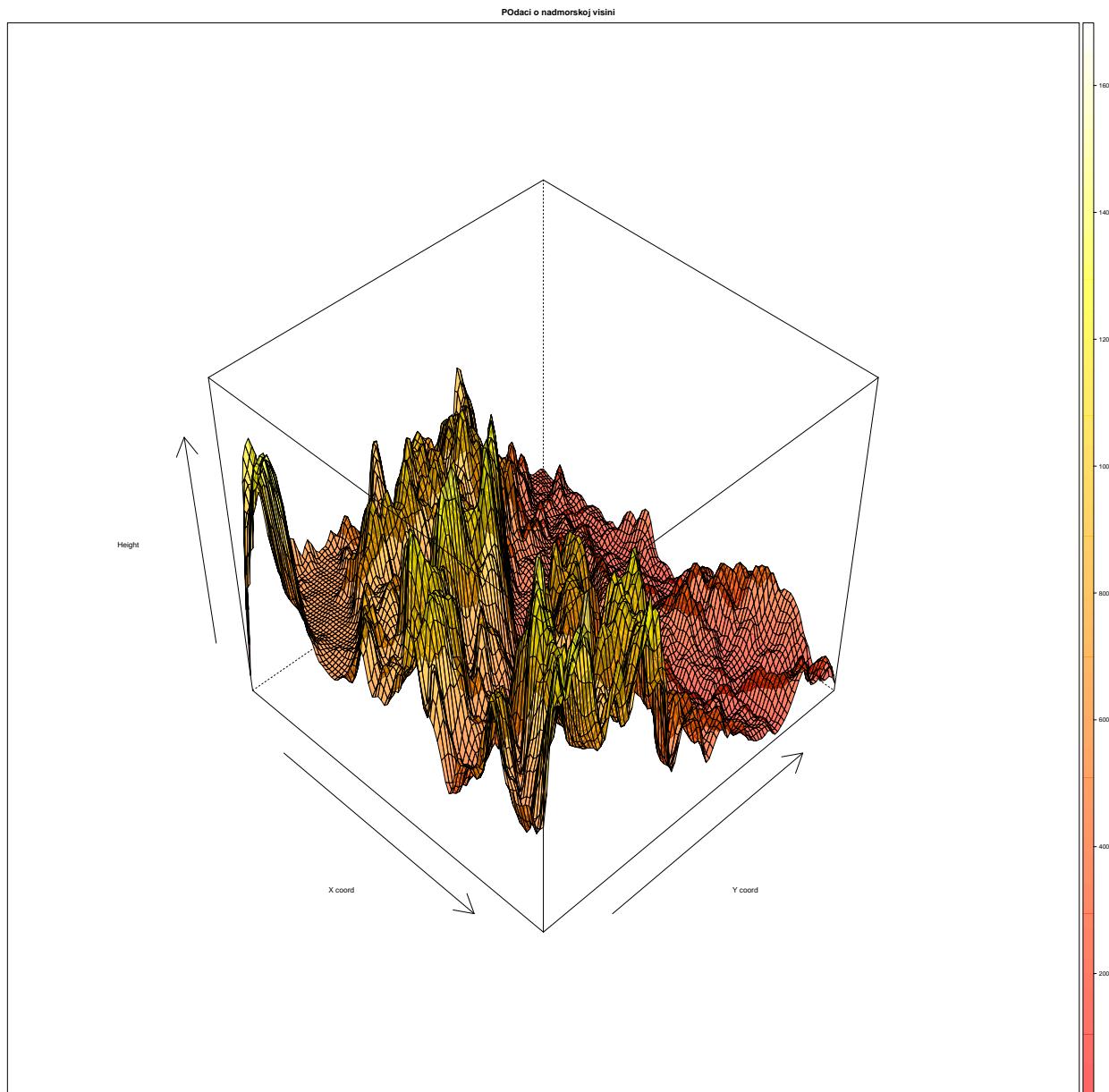
Funkcija `cloud()` proizvodi trodimenzionalne dijagrame raspršivanja. Mnogi od argumenata kao što su `pch` i `col`, `cloud()` funkcije identični su onima koji se koriste u `xypplot()` ali neki funkcioniraju drugačijie (npr. `scales` - ovdje se koriste da se uklone strelice koje su zadano (engl. `default`) nacrtane duž osi i da se zamijene debelim oznakama i skaliranjem vrijednosti za tri varijable). Pogledajte informacijsku karticu funkcije i pokušajte zaključiti kako koristiti `distance` i `screen`. Pokušajmo promjeniti te izraze u slijedećem kôdu i komentirajte.



Slika 133: Wireframe prikaz podataka **volcano**; *lattice* paket



Slika 134: Wireframe prikaz podataka nadmorske visine; lattice paket, podaci *elevation.fit*



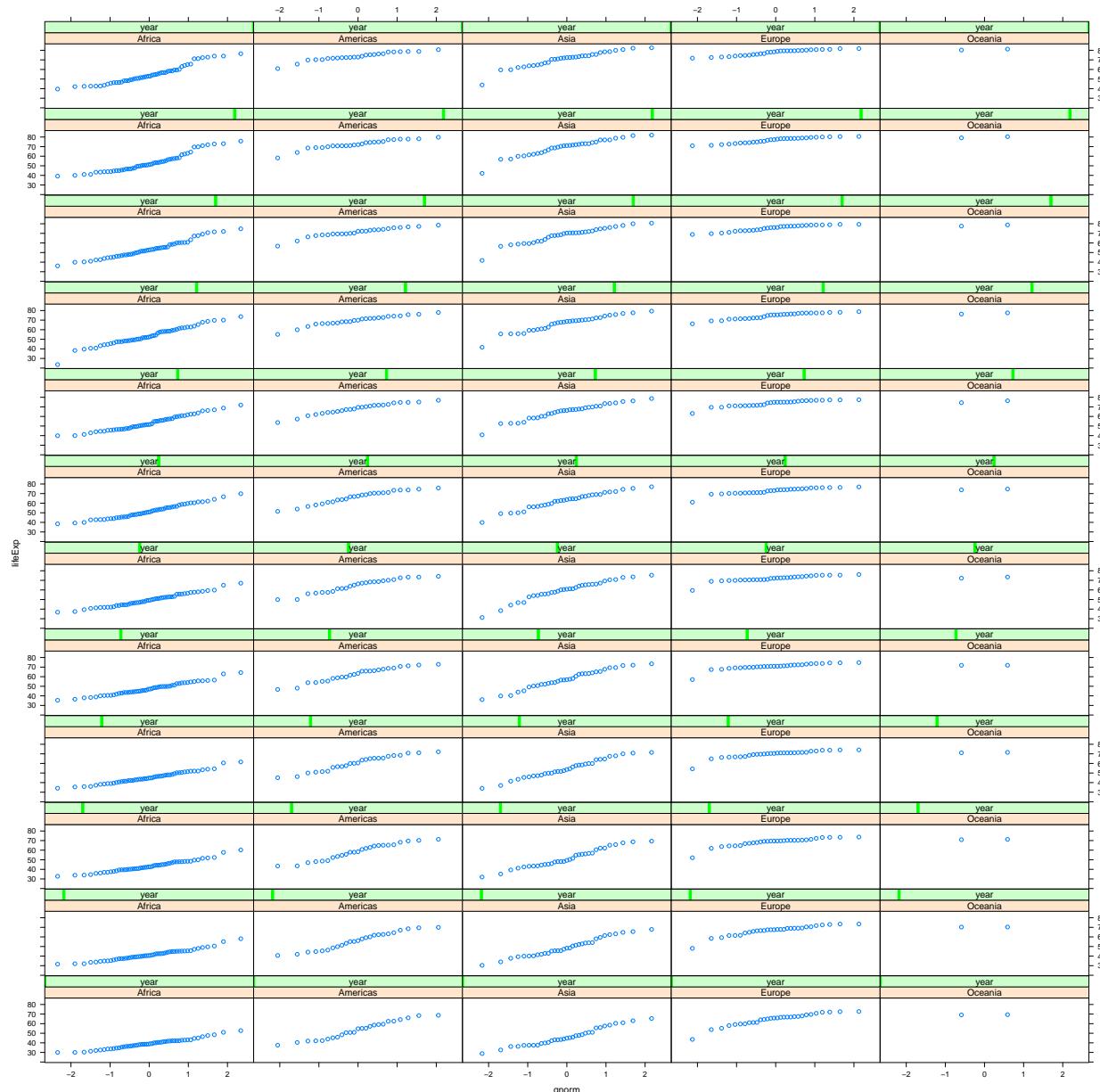
Slika 135: Wireframe prikaz podataka nadmorske visine; lattice paket, dio Hrvatske

#### 4.2.14.8 Function qqmath() {lattice}

Funkcija **qmath()** je dio skupine standardnih statističkih grafika kojima je namjera da se vizualizira distribucija kontinuirane slučajne varijable. Već smo vidjeli histograme i dijagrame gustoće, koji su procjene vjerojatnosti funkcije gustoće kontinuirane varijable. Funkcija **qqmath()** rezultira kvantil-kvantil (Q-Q) dijagrame uzorka u odnosu na teorijske distribucije, moguće uvjetovano drugim varijablama. Zadano ponašanje **qqmath()** slično je funkciji **qqnorm()**.

Kao što nam je već poznato, možemo izraditi i uvjetovane grafove, u sljedećem prikazu uvjetujemo (držimo konstantom) dvije varijable:

```
qqmath(~ lifeExp | continent * year, data = lattice_data)
```

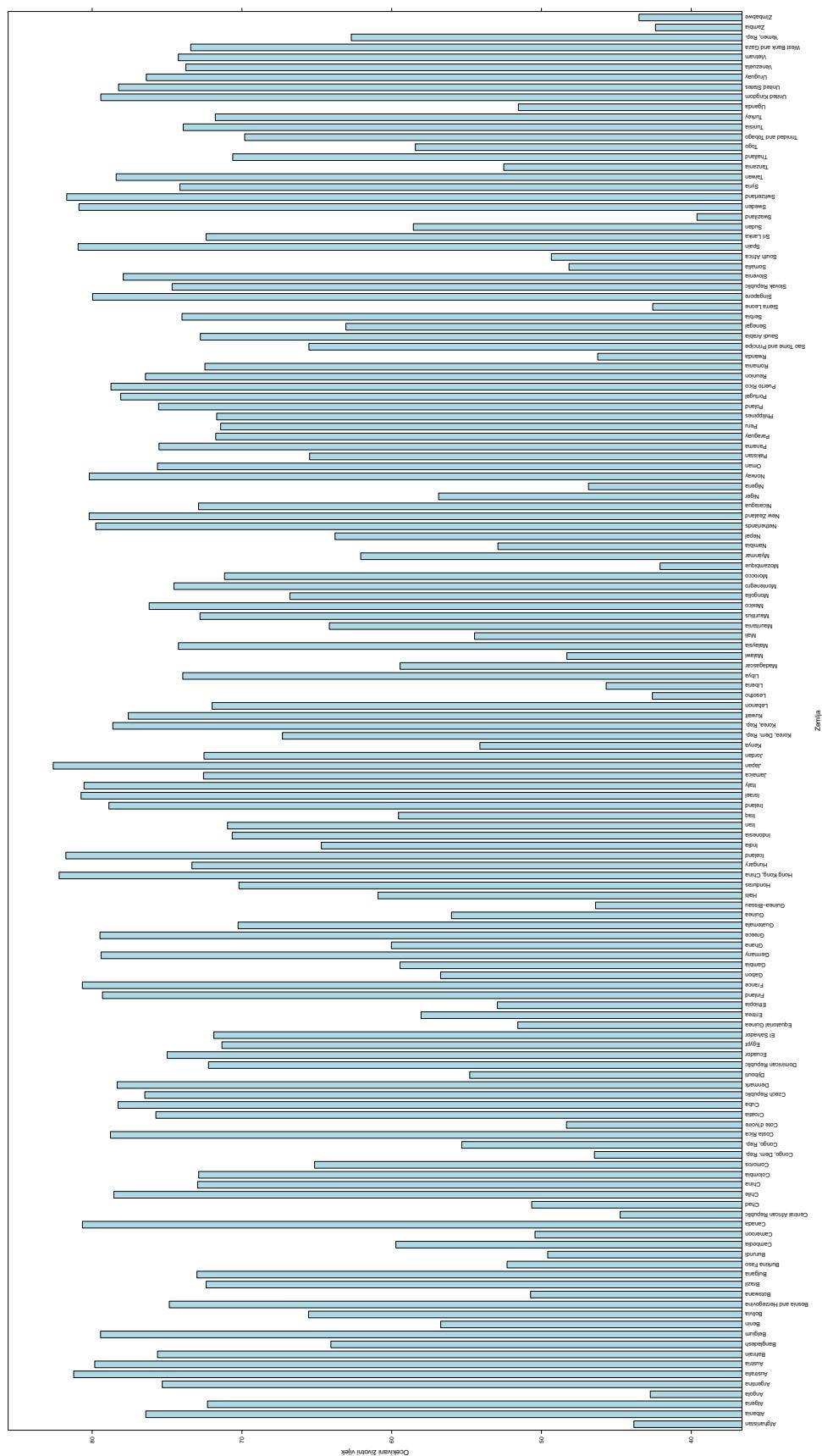


Slika 136: Funkcija **qqmath()** iz *lattice* paketa - primjer uvjetovanja po nivou faktorske varijable kontinent

#### 4.2.14.9 Funkcija `barchart()`(*high-level*)`{lattice}`

Funkcija koja kreira barplot prikaz, adekvatan grafički prikaz za kategoriske varijable.

```
barchart(lifeExp ~ country ,  
         lattice_data[lattice_data$year=="2007"],  
         horizontal=F,  
         col="lightblue",  
         xlab="Country",  
         ylab="Life expectancy",  
         scales = list(x = list(rot = 90)))
```



Slika 137: Funkcija `barchart()` iz lattice paketa

## 5 Literatura

- Becker, R. A., Chambers, J. M. and Wilks, A. R. 1988. The New S Language. Wadsworth & Brooks/Cole.
- Brewer C.A. 1999. Color Use Guidelines for Data Representation." In Proceedings of the Section on Statistical Graphics, American Statistical Association, pp. 55-60. Alexandria, VA. <http://www.personal.psu.edu/faculty/c/a/cab38/ColorSch/ASApaper.html>.
- Cleveland W. S. 1993 Visualizing Data. Summit, New Jersey: Hobart.
- Cleveland, W. S. 1985. The Elements of Graphing Data. Monterey, CA: Wadsworth.
- Cox N 2007. The Grammar of Graphics." Journal of Statistical Software, Book Reviews, 17(3), 1{7}. URL <http://www.jstatsoft.org/v17/b03/>.
- Deepayan Sarkar (2008) Lattice: Multivariate Data Visualization with R, Springer.
- ggplot2: Elegant Graphics for Data Analysis
- Harrower M.A., Brewer C.A. 2003. ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps." The Cartographic Journal, 40, 27-37. <http://ColorBrewer.org/>.
- <http://econ.uibk.ac.at/~zeileis/papers/Zeileis+Hornik+Murrell-2009.pdf>
- <http://lattice.r-forge.r-project.org/Vignettes/src/lattice-intro/lattice-intro.pdf>
- <http://lmdvr.r-forge.r-project.org/figures/figures.html>
- <http://polisci.msu.edu/jacoby/icpsr/graphics/lattice/Lattice,%20ICPSR%202016%20Outline,%20Ver%201.pdf>
- <http://www.ggplot2-exts.org/>
- <https://cran.r-project.org/web/packages/lattice/lattice.pdf>
- [https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV)
- <https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/iris.html>
- <https://stat.ethz.ch/R-manual/R-devel/library/grid/doc/viewports.pdf>
- <https://www.r-project.org/conferences/useR-2007/program/presentations/sarkar.pdf>
- <https://www.stat.auckland.ac.nz/~paul/RGraphics/rgraphics.html>
- <http://ms.mcmaster.ca/~bolker/misc/ggplot2-book.pdf>
- Ihaka R. 2003. Colour for Presentation Graphics." In K Hornik, F Leisch, A Zeileis (eds.), Proceedings of the 3rd International Workshop on Distributed Statistical Computing, Vienna, Austria. ISSN 1609-395X, (<http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/>).
- Jacoby, William G. 2006. "The Dot Plot: A Graphical Display for Labeled Quantitative Values." The Political Methodologist 14(1): 6-14.
- Meyer D., Zeileis A., and Hornik K. 2005 The strucplot framework: Visualizing multi-way contingency tables with vcd. Report 22, Department of Statistics and Mathematics, Wirtschaftsuniversität Wien, Research Report Series. [http://epub.wu.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01\\_8a1](http://epub.wu.ac.at/dyn/openURL?id=oai:epub.wu-wien.ac.at:epub-wu-01_8a1)
- Munsell A.H. 1905. A Color Notation. Munsell Color Company, Boston, Massachusetts.
- Murrell, P. 2005. R Graphics. Chapman & Hall/CRC Press.
- Smith A.R. 1978). Color Gamut Transform Pairs." Computer Graphics, 12(3), 12-19. ACM SIGGRAPH 78 Conference Proceedings, <http://www.alvyray.com/>.
- Yu D., Smith D.K., Zhu H., Guan Y., Lam T.T.Y\*. ggtrree: an R package for visualization and annotation of phylogenetic trees with their covariates and other associated data. Methods in Ecology and Evolution. doi:10.1111/2041-210X.12628.

- Zeileis,A., Hornik K., Murrell P. 2009. Escaping RGBland: Selecting Colors for Statistical Graphics. Computational Statistics & Data Analysis, 53(9), 3259-3270.