Grafy v R

Grafické znázornění dat je nejefektivnějším způsobem jejich prezentace, kvalitní graf podává ucelenou informaci o datech a není náhodou, že R obsahuje nástroje pro tvorbu mimořádně kvalitních grafů nepřeberné škály typů a kombinací.

Barvy

Dříve než si představíme nejpoužívanější typy grafů, dovolte mi ukázat neocenitelnou vlastnost tvorby v R pro větší množství grafů (typické právě pro diplomovou práci). Jedním ze základních požadavků bude u grafického výstupu jednotnost barev – ať už zvolíme černobílý tisk nebo barevný, vždy chceme udržet jistý soulad. Při tvorbě grafů např. v Calcu je, nechceme-li akceptovat výchozí nastavení, velmi náročné udržet takový řád, při změně konceptu (barevné grafy nahradit stupni šedé) pak téměř neřešitelný. Zde si můžeme nadefinovat barevnou škálu ať z konkrétních barev, nebo z přednastavených palet. Barevné podání všech grafů pak můžeme měnit na několika málo řádkách kódu.

```
#pevně definované barvy
barvyvek<-c("#FFF840", "#FFDD00", "#FFA840","#FF7400", "#FF5D40", "#FF0D00")
#vektor pěti barev z oranžovožluté palety
col=heat.colors(5)
#vektor o délce odpovídající počtu prvků budoucího grafu
col=heat.colors(length(zsData))
```

Doporučuji využívat předdefinovaných palet, dostupné jsou tyto:

- heat

Boxplot - krabicový graf

boxplot() nakreslí krabicový graf v případě numerického vektoru, u logických hodnot pracuje s jejich číslenou interpretací {0,1}, u faktoru ohlásí chybu.

Barplot/Histogram - sloupcový graf

hist() kreslí u numerického vektoru jeho histogram (tedy rozložení dat v rámci celého rozsahu), u faktoru taktéž zahlásí chybu

Plot - univerzál

Funkce plot() vytvoří graf v závislosti na povaze vstupních parametrů – voláme ji plot(dusledek∼ pricina) [12] V závislosti na povaze paramterů (zda jsou číslené, faktory nebo jejich kombinace) vykreslí smysluplný graf – zde asi nejvíce vysvětlí příklad všech tří možností xxx-čísločíslo, faktročíslo, faktorfktor

[12] Znak ∼ se jmenuje tilda – pokud ji na klávesnici nenajdete, vygooglete si ji pomocí jejího jména a vložte přes schránku :-).

Dodatečné čáry a texty v grafech

Častým požadavkem je vložení doplňující grafiky, která například ukazuje srovnání s předchozím výzkumem či odděluje určité kategorie; stejným mechanismem lze přidat i popisky. Uvádím zde pouze příklady volání těchto metod, podrobný popis jejich parametrů naleznete v nápovědě programu. Tyto funkce voláme těsně po vytvoření grafu – vždy se vkládají do naposledy vytvořeného výstupu.

points() lines() - kreslené body rovnou spojuje linkou abline() - absolutní člen + směrnice. h= nebo v= udělá horiz nebo vertikálu text()

```
abline(h =schmeil, col=barvaschmeil)
abline(v =2.5, untf = FALSE)
abline(v =4.9, untf = FALSE)
text(1, 90, paste("12 let"), pos = 4)
text(3.5, 90, paste("13 let"), pos = 4)
text(5.75, 90, paste("14 let"), pos = 4)
text(0, schmeil+6, paste("1899"), pos = 4, col=barvaschmeil)
```

Více grafů pohromadě

Přestože je možné složit dílčí grafy do mřížky přímo v textovém editoru, je taková práce zcela zbytečná. Již při jejich generování je můžeme seskupit a exportovat si tak již jen jeden obrázek obsahující velmi hezky sloučené grafy.

Umístění dvou grafů vedle sebe zajistí příkaz

```
layout(matrix(c(3,3,3,3,3,1,1,1,2,2), 2,5, byrow = TRUE), widths=c(1,1), )#heights=c(1,2))
```

Je možné definovat různou šířku dílčích grafů, ovšem v definici takových layoutů se lze vcelku snadno zamotat.

```
layout(matrix(c(3,3,3,3,3,1,1,1,2,2), 2,5, byrow = TRUE), widths=c(1,1), )#heights=c(1,2))
```

[iterace counter<-1:31 for(i in seq(along=counter)) otazka<-c(i) nakresli()]

Export grafů

Způsob jakým pracujete se skriptem, zda otevíráte uloženou workspace, kopírujete do konzole odjinud či otevíráte skript File → Open script… výrazně ovlivňuje způsob jakým exportovat grafy.

Vzorový graf můžeme vytvořit příkazem, otevře se v novém okně

```
> boxplot(trida8A.vek)
```

Změnou velikosti okna s grafem se dynamicky mění také jeho proporce, v okamžiku kdy jste spokojeni, tak pravým tlačítkem myši můžete uložit. Odhlédneme-li od nabídnutého formátu PostScript který pro řadu lidí představuje více komplikaci než výhodu, narazíme na další, závažnější problémy. Budeme-li chtít graf přegenerovat řekněme v jiném barevném schématu, budeme muset zajistit identický poměr stran a zopakovat proces uložení. Budeme-li navíc generovat několik grafů, budou se postupně objevovat v tomtéž okně a to je již velmi nešikovné.

Řešením je grafy již při jejich vzniku nevykreslovat na obrazovku, ale rovnou ukládat ve vhodném formátu i velikosti – to

je v R velmi snadné, nicméně konkrétní cesty jsou různé v závislosti na způsobu ukládání/zavádění skriptu, neboť na tomto základě nastavuje R pracovní adresář. Který to je, zjistíme voláním *getwd()*, změnu nastavíme voláním *setwd(path)*.

příkazy kopírujeme do konzole z textového souboru – pracovní adresář je nastaven na výchozí hodnotu (např. "C:/Users/Petr/Documents"). Na začátek textového souboru přidáme následující kód, který nastaví pracovní adresář, pokud neexistuje, tak vytvoří podadresář /output[13] a vytvoří objekt outputPath který bude cílovou destinací pro grafy.

[13] Pokud již adresář existuje, vypíše chybovou hlášku - ta je zde potlačena posledním parametrem

```
mainDir <- "c:/path/to/main/dir"
outputDir <- "output"
setwd(mainDir)
dir.create(file.path(mainDir, outputDir), showWarnings = FALSE)
outputPath <- file.path(mainDir, outputDir)
```

otevíráme skript z menu programu – tedy v porovnání s předchozí situací je jako pracovní již nastaven adresář, v němž se daný skript nachází. Můžeme tedy buď použít identický skript jako v předchozím případě nebo se ho ještě trochu více automatizovat.

```
mainDir <- getwd()
outputDir <- "output"
dir.create(file.path(mainDir, outputDir), showWarnings = FALSE)
outputPath <- file.path(mainDir, outputDir)
```

ukládáme a načítáme workspace – otevíráme-li workspace z menu programu, je pracovní adresář nastaven opět na výchozí hodnotu a použijeme první verzi; pokud ale otevřeme R poklikáním na soubor workspace (*.RData), nastává příležitost pro druhou verzi skriptu[14].

[14] Může se zdát zbytečné znovupřepisování objektu *outputPath*; nezapomeňte ale, že také provádí kontrolu zda podadresář /output vůbec existuje...

Po komplikaci s pracovním adresářem je již export grafů procházkou růžovým sadem, kdy identické příkazy generující graf prostě obalíme funkcí podle formátu výstupu[15]; zadáme jméno souboru, rozměry a rozlišení DPI, vykreslíme graf a zavřeme hotový soubor s grafem.

[15] Pro potřeby posterů je dobré vědět že R umí také již zmíněný PostScript či SVG, tedy vektory.

```
png(paste(outputPath, "/graf_trida8A.png", sep = ""),
  width= 190,
  height= 220,
  units= "mm",
  res = 300)
 boxplot(trida8A.vek)
dev.off()
```