O S T R A V S K Á U N I V E R Z I T A
P ř í r o d o v ě d e c k á f a k u l t a

Cvičení z biostatistiky

Základy práce se softwarem R

Pavel Drozd

OSTRAVA 2007

ISBN 978-80-7368-433-4

Odborná recenze:
RNDr. PaedDr. Hashim Habiballa, PhD. (Katedra informatiky a počítačů PřF OU)
Prof. MVDr. Emil Tkadlec, CSc. (Katedra ekologie a životního prostředí PřF UP)

OBSAH

ÚVOD

Biolog tráví více času u počítače než v terénu nebo v laboratoři. Ačkoliv se laikům zdá tento výrok dost přitažený za vlasy, mí kolegové potvrdí, že mám pravdu. Zpracování dat a ověřování hypotéz je zcela nezbytnou součástí práce biologa. Naučit se všechny postupy a metody je záležitost dost náročná a vyžaduje:

- pochopení teorie testování a přípravy experimentu;
- znalost jednotlivých testů a pochopení principu výpočtů;
- dovednost práce se statistickým softwarem (kalkulačky už dávno statistici odepsali);
- schopnost správně interpretovat jednotlivé výsledky.

Cílem cvičení z biostatistiky není vysvětlit teoretické principy testování a zpracování dat, ale navázat na teorii a obecné znalosti testů a propojit je s praktickými postupy při provádění analýzy dat. Tento díl skripta se nebude přímo zabývat postupy při analýze biologických dat. Je zaměřen na základy práce se statistickým (vlastně více než statistickým) softwarem R. Proč rovnou neanalyzovat? R je komplexní systém, který je pro začátečníka poměrně náročný, ale umožňuje špičkové zpracování dat včetně grafických výstupů. Zvládnutí základních pravidel a manipulace s daty v R proto zabere dost času (předchozí dvě věty si v různých obměnách přečtete ještě několikrát dále v textu). Doporučuji čtenářům, aby se pochopení postupů poctivě věnovali. Není účelem zapamatovat si všechny příkazy a jejich posloupnosti při analýze. Důležité je porozumět funkcím a zadávání jejích argumentů a také pochopit, proč jsme volili právě tento postup.

Skriptum je určeno jako doplňující materiál pro studenty denního studia a jako hlavní studijní materiál pro studenty kombinovaného studia (nebo pro studenty denního studia, kteří se mylně domnívají, že cvičení jsou zbytečná). Vlastní texty a návody jsou proloženy komentáři a soubory otázek a řešených úkolů. Zadané úkoly je nutno řešit až po pochopení uvedených funkcí a postupů. Otázky jsou vždy odstupňovány od nejlehčích po nejsložitější. Nevěřím klasickému lidovému výroku "na to musí mít člověk buňky". Řešení úkolů vyžaduje dvě vlastnosti, které by se měl vysokoškolský student stejně neustále posilovat – trpělivost a kreativitu. Pokud vás při řešení napadne jiný postup, než je uveden v řešení na konci kapitoly, pak jdete tou nejlepší cestou.

Vím, že dokonalé zvládnutí všech informací, které jsou obsaženy ve skriptu je pro začátečníka naprosto nereálné. Nejde však o to, všechny funkce znát nazpaměť (některé uvádím spíše pro úplnost). Důležité je porozumět principům a naučit se pracovat s nápovědou (včetně tohoto skripta). Já sám si spoustu uvedených funkcí nepamatuji, ale vím, že je najdu v nápovědě s podrobným popisem a příklady.

Ještě poslední poznámka. Toho, kdo se bude divit, že se některé myšlenky z úvodu objevuji ještě na dalších místech skripta, chci uklidnit, že se jedná o záměr. Předpokládám, že většina čtenářů úvody nečte.

Příjemné studium.

ORIENTACE V TEXTU

Tento studijní text je určen pro distanční studium a je tedy koncipován tak, aby byl čtenář při studiu částečně veden a komunikoval jako při prezenční formě výuky. Najdete v něm proto určité zvláštnosti a značky, které by měly samostudium usnadnit:

Cíle kapitoly – jsou uvedeny na začátku každé kapitoly a jejich smyslem je ujasnit čtenáři, co by měl po nastudování kapitoly znát.

Klíčová slova – stejně jako ve vědeckých článcích obsahuje tento odstavec základní pojmy, které charakterizují lekci.

Průvodce – může obsahovat rady, jak danou kapitolu studovat, vlastní zkušenosti apod. Je oddělen od textu rámečkem a šedým stínováním textu.

Příklady řešení problémů, detailní informace – jejich cílem je konkretizovat a demonstrovat uvedené postupy. Jsou odděleny rámečkem a menším písmem. Vzhledem k velkému množství příkladů v textu nemusí být tento symbol striktně uveden u všech odstavců.

Kontrolní úkoly – ověřují, zda jste text prostudovali dostatečně důkladně. Úkoly se nemusí nutně týkat všech funkcí. Princip učení se postupům by měl být následující. Prostudujete si kapitolu. Projdete všechny příklady a budete se snažit jim porozumět. Poté se podíváte na kontrolní úkoly a pokusíte se je vyřešit. Až tehdy, když jste vynaložili značné úsilí a stále si nevíte rady, podívejte se na Výsledky.

Shrnutí – podává stručně obsah kapitoly. Pozor! Nestačí pouze znalost souhrnu k tomu, abyste porozuměli dané kapitole.

Metody k zapamatování – nejdůležitější metody, funkce a postupy z dané kapitoly, které by si měl čtenář (spolu s jejich významem) zapamatovat.

Výsledky – řešení kontrolních otázek. – podává stručně obsah kapitoly. Na výsledky se nedívejte po pěti minutách, kdy vám dojde trpělivost, ale až v okamžiku, kdy opravdu nevíte, jak úkol řešit. Jinak jejich význam ztrácí smysl.

Korespondenční úkol – tento typ úkolů zašlete podle instrukcí elektronickou poštou svému tutorovi.

ZÁKLADNÍ INFORMACE O R

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- budete znát základní údaje o softwaru R;
- dozvíte se o tom, proč budeme využívat právě tento software;
- seznámíte se se základní prací s konzolou.

Klíčová slova: R-CRAN project, free software, R.

Průvodce
V úvodní kapitole se vám pokusím vysvětlit, proč pracovat v R, jak program vypadá a jak.se v něm jednoduše orientovat. Nedoporučuji kapitolu vynechat, protože obsahuje základní údaje a pojmy, metodu instalace programu atd. Na své si přijdou také fandové PC, pro které bude určitě motivační. R je totiž výzva. Kdo se trochu vyzná v počítačích, jistě slyšel pojmy jako Linux, TeX nebo OpenOffice. Jedná se o tzv. free software, tedy volně stažitelné programy, které jsou většinou alternativou ke klasickým komerčním produktům. Linux je operační systém, TeX program pro sazbu dokumentů, OpenOffice je volně stažitelná obdoba Microsoft Office. R je moderní alternativou komerčních statistických programů. Krátkozrací uživatelé většinou tyto programy kritizují pro jejich složitost, avšak free software má řadu výhod, které počáteční dojem složitosti značně převyšují. Složité ovládání je opravdu jen prvotní dojem uživatelů "zmlsaných" drahým softwarem (často pirátsky staženým z Internetu). Aby člověk vyjel ze zaběhnutých kolejí stačí trocha trpělivosti. Buďte prosím i vy trpěliví. Uvidíte, že se to vyplatí.

Co je R?

Extenzivní vývoj statistických metod v biologii vyvíjí tlak také na software, který umožňuje tyto metody provádět. Biologové se díky tomu často dostávají před zcela zásadní problém. Mohou začít využívat komerční statistický software, který bývá finančně nákladný a je nutné průběžně provádět update (samozřejmě zpoplatněný) s rizikem, že nově vyvinuté speciální analýzy nebudou do těchto programů implementovány, nebo používat volně dostupný specializovaný software, který vyvíjí kolegové zdatnější v programování. To ale znamená, že často používají velké množství nekompatibilních formátů dat a velkou část úprav jsou nuceni provádět manuálně. Velmi dobrým řešením pro základní i pokročilejší techniky analýzy biologických dat je software R.

Jedná se o programovací jazyk a prostředí pro statistické analýzy a grafiku, které původně vyvinuli Robert Gentleman a Ross Ihaka z Aucklandské Univerzity. Jazyk R podobný jazyku S známému zejména díky populárnímu komerčnímu programu S-plus, který byl vyvinu právě na bázi jazyka S. V současné době má projekt R 17 členů „core team members“ (základní tým vývojářů), 55 „contributors“ (stálí přispívatelé; zjistíme přímo v R po zadání příkazu `contributors()`) a je podporován 21 významnými institucemi jako například Katedra biostatistiky Kalifornské univerzity atd. Hlavními výhodami programu je:

- **Dostupnost**. R patří mezi tzv. „open source“ programy, tedy volně dostupné, šiřitelné a modifikovatelné programy. Je zařazen v rámci projektu GNU nadace „Free Software Foundation“. Tato nadace používá místo označení "copyright" slovní hříčku "copyleft".
- **Kompatibilita**. R je vyvinuto pro operační systém Windows, Unix/Linux i Macintosh. Data je možno importovat z různých formátů včetně schránky (clipboard), csf formátů, některých aplikací pro GIS atd.
- **Množství analytických nástrojů**. Kromě základních funkcí obsahuje R ohromné množství doplňujících balíčků s knihovnami funkcí pro různé typy analýz (včetně aplikací v biologických a příbuzných vědních oborech – např. bioinformatika, taxonomie, genetika, geografie atd.). V současné době je k dispozici okolo 750 balíčků (packages). Během instalace další knihovny se samozřejmě o nové funkce doplní také nápověda.
- **Aktuálnost**. Přispívatelé velice rychle (často bezprostředně) reagují na vývoj nových metod ve statistice, takže se v R objevují metody, které často ještě nejsou implementovány do klasického komerčního software.
- **Rozsáhlé možnosti grafických výstupů**. Grafika v R obsahuje řadu standardních i moderních grafických výstupů. Narozdíl od běžných programů je možno pracovat s grafikou na několika uživatelských úrovních, od nejjednoduššího nastavení až po uživatelem definované měřítko, osy, formáty bodů, kombinované grafy apod..
- **Velké množství studijních materiálů**. Na stránkách projektu R jsou volně dostupné manuály pro práci s programem (většinou ve formě PDF). Kromě toho byla o programu R publikována cela řada učebnic, manuálů a odborných prací.
- **Programování**. Pro pokročilejší PC fandy dodávám, že R je velice efektivní objektově orientovaný programovací jazyk. Po seznámení se základní syntaxí může i naprostý začátečník naprogramovat jednoduché funkce urychlující např. zpracování dat nebo grafické výstupy. Pokročilejšímu uživateli umožní velice snadno propojit i sérii dosti komplikovaných statistických analýz a funkcí, které již jsou vestavěnou součástí R (např. mnohorozměrná analýza dat apod.). Zdrojový kód většiny vestavěných funkcí je navíc možné zobrazit, krokovat a následně upravovat.

Základní informace o instalaci a prostředí

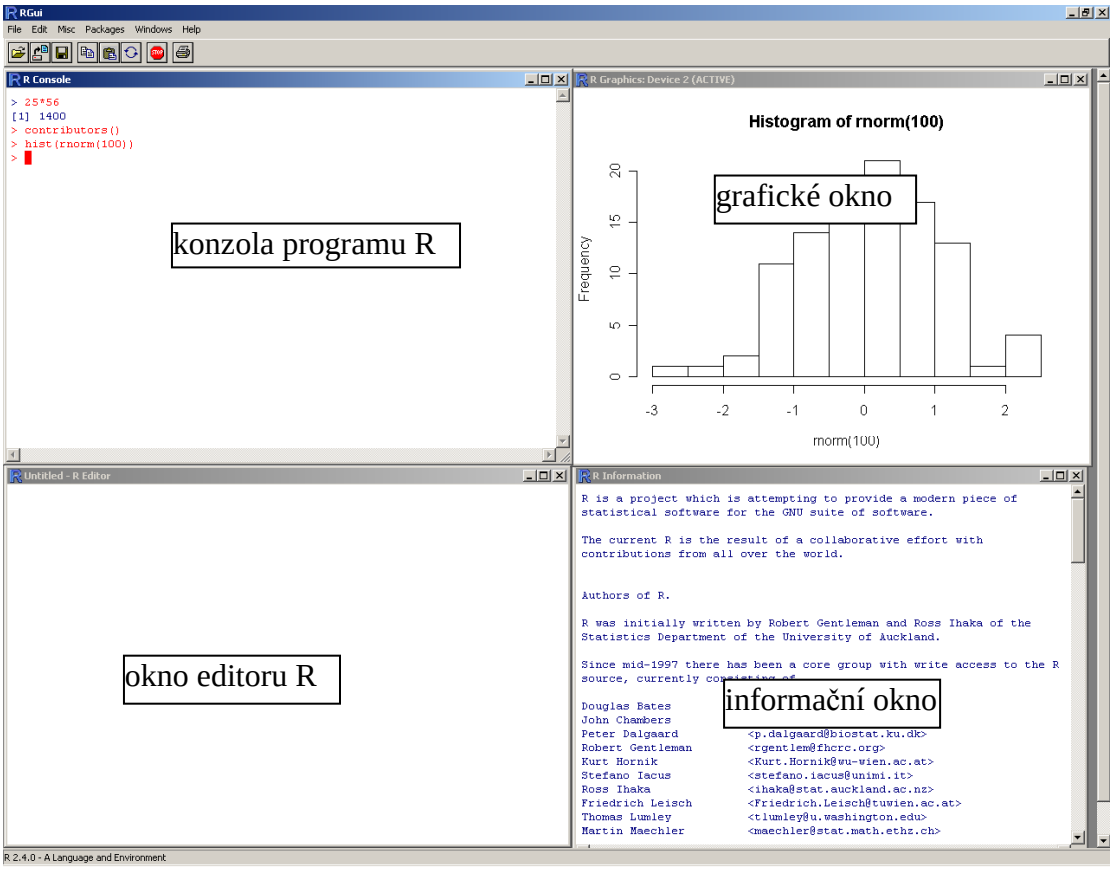
Program lze získat z webových stránek projektu R (http://www.R-project.org), respektive z archivu CRAN (např. http://cran.at.r-project.org). K vlastnímu instalačnímu programu se dostanete v levém menu v oddílu *Software* volbou *R Binaries*. Dále se rozhodnete podle vašeho operačního systému (např. Windows) a následně zvolíte *base*, kde je stažitelný základní instalační program, např. R-2.5.1-win32.exe (27 MB). Vlastní instalace je poměrně jednoduchá a nevyžaduje žádné speciální znalosti.

Balíček *base* obsahuje pouze základní knihovny (pro začátečníka jsou naprosto dostačující) – base (základní operace), datasets (knihovna příkladů dat), graphics, grDevices, grid (knihovny pro grafiku), methods (formálně definované metody a třídy objektů a programovací nástroje), stats, stats4, splines (statistické metody), tcltk, tools, utils (nástroje pro programování, vývoj a administraci). Kromě těchto základních knihoven však R obsahuje ohromné množství dalších nástrojů, které lze stáhnout z archivu CRAN (*Packages*) nebo přímo v menu programu *Packages – Install packages*.

Po nainstalování a spuštění se objeví okno programu s konzolou (**R console**). V tomto okně lze spouštět funkce, vkládat objekty a objevují se zde i základní výstupy. Za specifických podmínek se mohou zobrazit ještě následující okna:

- Okno nápovědy – v případě dotazu na určitou funkci (help, ?). Toto okno už v nových verzích můžeme nastavit jako samostatné a není přímo součástí okna R.
- Grafické okno (R Graphics) – jestliže spustíme grafickou funkci (např. plot, hist, boxplot)
- Editor programu (R Editor) – jestliže v menu File zadáme New Script, nebo otevřeme New Script. V tomto okně si můžeme psát celé programy, poznámky k funkcím atd.
- Informační okno (R Information) – objevují se v něm různé informace volané funkcemi (např. contributors)

Aktuální citaci R získáme také pomocí příkazu `citation()`, citaci jednotlivých knihoven pak jako `citation("název knihovny")`:
R Development Core Team (2006). R: A language and environment for statistical computing. (software) R Foundation for Statistical Computing, Vienna (Austria). ISBN 3-900051-07-0, dostupné z: http://www.R-project.org.



Obr 1: Základní okno programu R se čtyřmi typy oken.

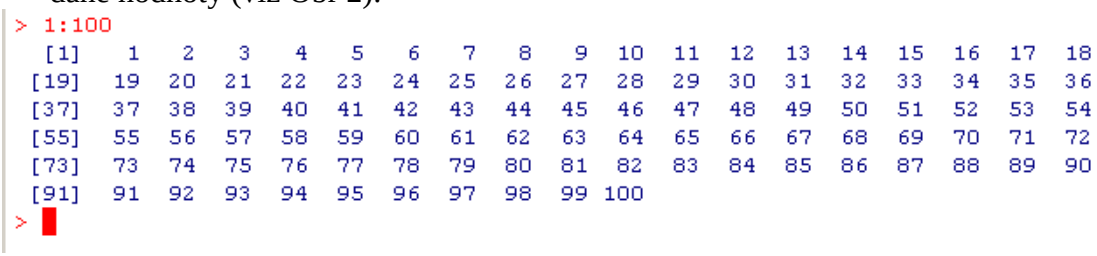
Kontrolní úkoly

1. Prohlédněte si stránky projektu (www.r-project.org). Poté přejděte do archívu a stáhněte si instalační program.
2. Nainstalujte si program R a spusťte jej..

Práce s konzolou

Okno konzoly slouží jako základní vstupní a výstupní rozhraní programu. Při práci s konzolou je nutné znát následující pravidla.

- Vstupní řádky, do nichž vepisujeme příkazy, jsou odlišeny červeným písmem a začínají znakem **>** (vypisuje se automaticky), pokud nezměníme základní nastavení.
- Výstupní řádek je psán modrým písmem a v případě hodnot začíná pořadovým číslem dané hodnoty (viz Obr 2).



Obr 2: Příkaz pro vypsání čísel od 1 do 100. Jestliže se čísla nevejdou na řádek, pak pokračují na následujícím s označením pořadí (číslo 19 je devatenácté v pořadí).

- Napíšeme-li určitý příkaz do konzoly a spustíme jej pomocí klávesy Enter nelze jej již opravit přímo, ale v již napsaných příkazech můžeme listovat pomocí kláves ↑↓. Např. k příkazu 1:100 na obrázku se vrátíme pomocí stisku klávesy ↑. Historii příkazů lze uložit přes menu *File – Save History* (lze ji otevřít např. v Poznámkovém bloku).
- Obsah konzoly lze celý uložit pomocí *File – Save to File*.
- Konzolu vymažeme klávesovou zkratkou Ctrl L nebo v menu *Edit – Clear console*.
- Příkazy můžeme kopírovat a vkládat pomocí klasických zkratek Ctrl C, Ctrl V nebo vkládáme přes menu *Edit – Paste commands only*, která z vybraného textu vynechá vše kromě příkazů, tzn. zejména symbol >.
- konzola R je "case sensitive" – citlivá na velká a malá písmena, proto například funkce print není totéž co Print.

Obr 3: Příklad citlivosti na velká a malá písmena. R zná příkaz print (tiskni), ale nezná příkaz Print.

- V případě, že by byl příkaz velmi dlouhý a chtěli bychom jej rozdělit do více řádků, pak můžeme skočit na další řádek pomocí klávesy enter s tím, že se automaticky přidá na začátek nového řádku symbol +. To znamená, že program čeká na ukončení příkazu na tomto řádku (popř. dalších). Jestliže se symbol objeví omylem, pak můžeme psaní příkazu zrušit klávesou ESC nebo tlačítkem *Stop current computation*.

Obr 4: Nedopíšeme-li příkaz na jednom řádku, pokračuje R na dalším a navazující řádek automaticky označí symbolem +.

- Pro ukončení programu R lze použít standardní metody programů ve Windows nebo příkaz `q()`.

Základní syntaxe

Následující výpisy příkazů používaných v R budou uváděny tak, aby byly kopírovatelné do programu. Proto chcete-li vyzkoušet určitý příkaz, stačí jej pouze překopírovat (vstupní příkazy jsou modré, černé výstupy nekopírujte) z tohoto dokumentu do R.

R je jako každý programovací jazyk velice striktní ve stylu zápisu. Při nedodržení přesného zápisu program buď hlásí chybu (syntax error) nebo může dojít k provedení odlišné operace. V tomto případě se jedná o sémantickou chybu, kterou počítač neodhalí, protože neví, že se po něm požaduje něco jiného. Základní principy zápisu jsou uvedeny v následujících bodech.

- Všechny operace, včetně práce s výpisem a zadáváním objektů (dat) je nutné provádět pomocí vepsaných příkazů a funkcí.
- `#` – symbol slouží pro označení poznámky. Od místa označeného tímto symbolem do konce řádku se nic nevypisuje.

```
25 #vypíše pouze číslo 25 a ne tuto poznámku
[1] 25
```

- `()` Při zadávání většiny funkcí a příkazů je nutné do kulatých závorek zadat tzv. argumenty (parametry), které upřesňují použití funkce. Některé z nich mohou být automaticky nastaveny, některé vyžadují nastavení přímo. Informace o argumentech a jejich nastavení získáte buď v nápovědě nebo pomocí funkce `args(název funkce)`. Základní tvar zápisu funkce je tedy: **příkaz (soubor argumentů oddělených čárkou)**

```
round(x, digits = 0)
```
Zaokrouhlování čísla (nebo více čísel) x na určitý počet desetinných míst (digits), digits je standardně nastaveno na 0 desetinných míst, tzn. např. round(4.569) vrátí výsledek 4. Argumenty je často možné zkracovat (jestliže zkratka nemůže znamenat jiný argument), např. digits=2 stačí psát v podobě d=2.

```
args(print.default) #vypíše argumenty funkce print.default
function (x, digits = NULL, quote = TRUE, na.print = NULL,
    print.gap = NULL,right = FALSE, max = NULL, ...)
```
Jestliže se při výpisu funkce objeví symbol tří teček (...) znamená to, že existují ještě další obecné argumenty přiřazené k tomuto typu funkce, ale specifikované u některé jiné funkce.

- `;` Více příkazů na jeden řádek lze zapsat tak, že je oddělíme středníkem, tzn. `funkce(argumenty); funkce(argumenty)`

```
print(1);print(3:8) #vypíše číslo 1 a pak vypíše čísla 3-8
[1] 1
[1] 3 4 5 6 7 8
```

- `{}` Výraz ve složených závorkách může být použit pro soubor příkazů, které se mají spustit bezprostředně po sobě (slouží např. pro psaní programů nebo delších funkcí). Symbol + na začátku řádku opět znamená, že výraz pokračuje z předchozího řádku. Po stisknutí ENTER tak můžeme psát na další řádek dokud neukončíme pravou složenou závorkou.

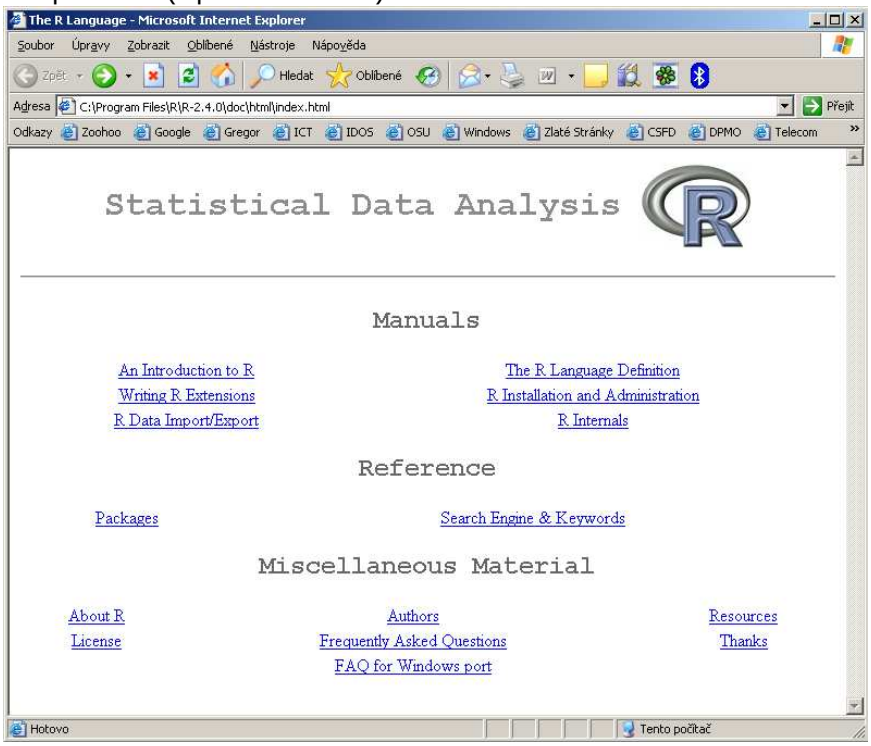
```
> {print(1:10)
+ print(1:5)
+ print(log(100))} #vytiskne postupně čísla 1-10, 1-5 a logaritmus 100
 [1]  1  2  3  4  5  6  7  8  9 10
[1] 1 2 3 4 5
[1] 4.60517
>
```

Obr 5: Ukázka spojení příkazů složenými závorkami.

Práce s nápovědou

Nápověda je pro používání programu R zcela zásadní. Nelze si totiž zapamatovat všechny funkce a argumenty, proto ji doporučuji maximálně využívat. R obsahuje tři základní typy nápovědy, přičemž všechny můžeme používat současně:

- **PDF manuály**. Obsahují základní manuály a základní reference o funkcích. Spouštíme je přes menu *Help – Manuals (in PDF)*.
- **HTML nápověda**. Vestavěná nápověda ve formě html stránky, která obsahuje manuály, seznamy funkcí v jednotlivých knihovnách (packages) a další materiály. Spouštíme ji přes menu *Help – Html Help* nebo příkazem `help.start(update=FALSE)`.

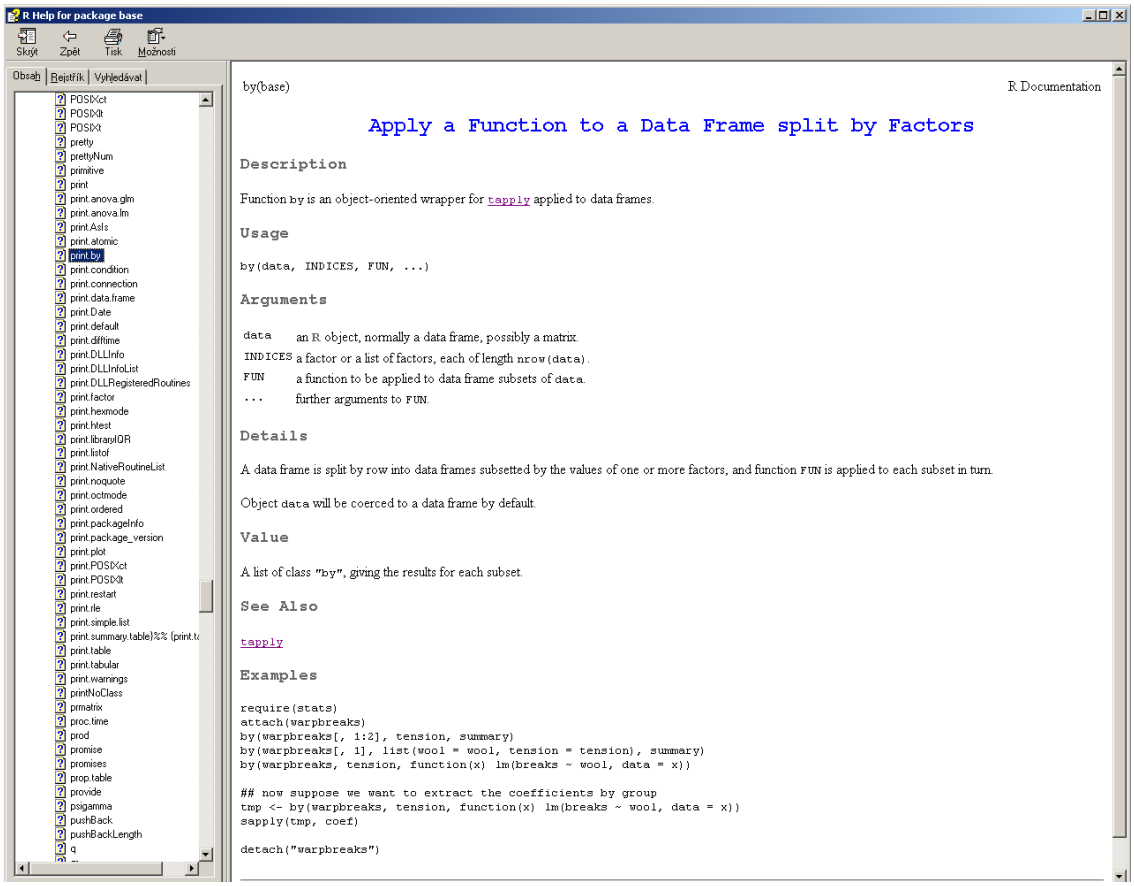


Obr 6: Html nápověda.

- **Reference.** Jedná se o nápovědu ke konkrétním objektům a funkcím v aktuálně zavedených knihovnách. Spouštíme ji buď příkazem `help(téma)` nebo `?téma` nebo v menu *Help – R-functions (text...)*.
  Reference mají obvykle následující položky:
  - Description – stručný popis daného tématu (funkce, objektu atd.)
  - Usage – způsob použití (zápis).
  - Arguments – seznam argumentů a jejich vysvětlení.
  - Details – upřesnění týkající se popisu tématu nebo určitého argumentu.
  - Value – výstupní hodnota dané funkce (metody) – typ a struktura.

- Author – autor dané funkce (objektu).
- References – odkaz na literaturu vztahující se k tématu.
- See also – podobná nebo navazující témata.
- Examples – příklady.

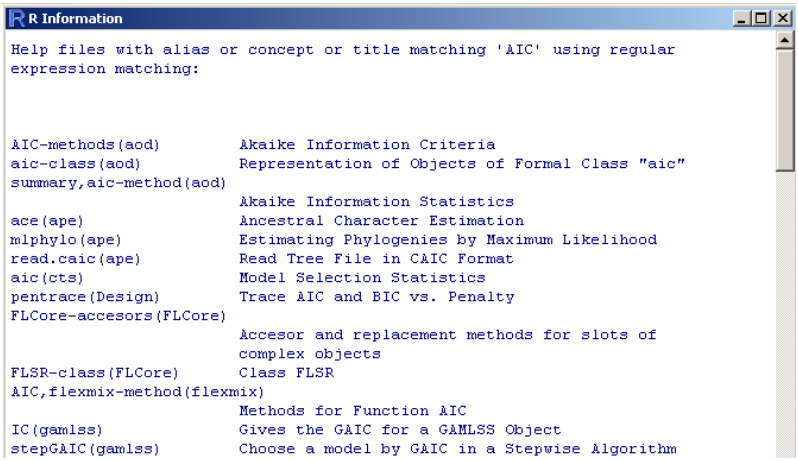
```
help (ls) # alternativní k ? ls
help (glm) # alternativní k ? glm
help (summary) # alternativní k ? summary
?glm
```



Obr 7: Reference k funkci *by*.

- **Vyhledávání témat v kontextu.** Předchozí příkaz vyhledává pouze v názvech a zkrácených popisech funkcí a objektů, zatímco tato možnost vyhledá dostupné položky nápovědy, v jejichž názvu, popisu (description) nebo klíčových slovech se vyskytuje daný text (téma). Kontextové vyhledávání spouštíme funkcí `help.search ("téma")` nebo menu *Help – Search Help*.

```
help.search("Wilcoxon")
```



Obr 8: Informační okno po příkazu `help.search("AIC")`. Vypíšou se dostupné položky, v závorkách knihovny, ve kterých se položky nacházejí, a popis položek nápovědy.

Kontrolní úkoly

3. Zjistěte jaké argumenty má funkce help a funkce rep.
4. Vepište do R uváděné příklady pro práci s konzolou a nápovědou a spusťte je.
5. Z nápovědy se pokuste zjistit, k čemu slouží funkce rep.

Shrnutí:

R je prostředí a programovací jazyk vhodný pro statistickou analýzu dat a jejich grafickou prezentaci. Jedná se obdobu jazyka S známého díky komerčnímu produktu S-plus. Narozdíl od S-plus patří R mezi tzv. free software, tedy software volně šířitelný. U každého programovacího jazyka je velice důležité znát základní syntaxi příkazů a systém práce s konzolou (oknem, ve kterém zadáváme příkazy). R obsahuje poměrně rozsáhlou nápovědu, která usnadňuje orientaci v příkazech.

Metody k zapamatování:

Adresa, ze které lze stáhnout R a doplňující materiály:
http://www.R-project.org, resp. archiv CRAN (např. http://cran.at.r-project.org**).**
poznámka v příkazovém řádku
() slouží ke vkládání argumentů do funkce
`args``(název funkce)` – výpis argumentů funkce
`;` odděluje více příkazů na jednom řádku
`+` označuje spojení více řádků do jednoho příkazu
`{}` umožňuje provést sled příkazů umístěný v závorkách na více řádcích
`help``(název funkce)` nápověda pro danou funkci (jinak také ?název)
`help.search``("kontext")` hledání funkcí podle kontextu

Výsledky

ad 3)

```
> args(help)
function (topic,offline = FALSE, package = NULL,lib.loc = NULL,verbose = getOption
   ("verbose"), try.all.packages = getOption("help.try.all.packages"),
   chmhelp = getOption("chmhelp"), htmlhelp = getOption("htmlhelp"),
   pager = getOption("pager"))
```

ad 5)

```
> help(rep)
```

slouží ke zopakování objektu uvedeného v argumentu x (viz další kapitola)

OBJEKTY

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- naučíte se přiřazovat jména k objektům;
- budete znát základní typy hodnot, které lze přiřadit objektům;
- zvládnete vytváření a manipulaci zejména s vektory, maticemi, tabulkami dat, poli a seznamy;
- budete umět studovat strukturu objektů a převádět objekty na jiné typy.

Klíčová slova: objekty, typy objektů, struktura objektů.

Průvodce
Následující kapitola je zcela zásadní. Abychom mohli do R zadávat data, je nutné pochopit, jaké typy dat lze vytvořit a použít, jak je možné zkoumat jejich strukturu a manipulovat s ní. Vřele vám doporučuji postupovat pečlivě a pomalu. A nezapomeňte na kontrolní úkoly.

Názvy a základní typy

Vše s čím v programu R pracujeme jsou objekty. Objektem může být číslo, množina čísel, text, matice, funkce atd.. Objekty většinou při vytváření pojmenujeme, tzn. objektu „přiřadíme název a pak nahrazujeme vypisování objektu pouze jeho názvem“. Přiřazení většinou provádíme funkcí `assign` nebo pomocí „šipky“ `<-`. Název nám potom zastupuje daný objekt. Podle toho, zda objekty dále měníme rozlišujeme konstanty a proměnné. Názvy objektu vytváříme podle těchto pravidel:

- Název by se neměl by se shodovat s názvem jiného objektu (např. existuje funkce *log*, proto bychom neměli dávat jinému objektu název *log*, protože bychom přepsali původní). Je také nevhodné používat písmena c, q, t, C, D, F, I, T. (Poznámka: Jestliže chceme obecně zjistit, které názvy jsou již použity stačí je vyzkoušet napsat.)

```
a # momentálně není definován objekt s tímto názvem
Error: object "a" not found

c # název je použit pro funkci
.Primitive("c")

F # objekt s tímto názvem existuje
[1] FALSE

plus # momentálně není definován objekt s tímto názvem
Error: object "plus" not found

lm # název lm je použit pro funkci
function (formula, data, subset, weights, na.action, method = "qr",
    model = TRUE, x = FALSE, y = FALSE, qr = TRUE, singular.ok = TRUE,
    contrasts = NULL, offset, ...)
```

- Pojmenování by mělo mít logiku (pro objekt obsahující množinu naměřených nadmořských výšek dáme např. název *altitude*, *vyska* nebo jejich zkratky, abychom nemuseli dlouho přemýšlet, jak jsme objekt nazvali).

- Názvy by neměly být zbytečně dlouhé (např. objekt s naměřenými hodnotami *intoxikace* nazveme raději *intox* nebo *altitude* nazveme *altit* abychom nemuseli ve funkcích objekt dlouze vypisovat).
- Rozdělení názvu na dvě části se provádí pomocí tečky (např. průměrnou nadmořskou výšku můžeme nazvat *altit.prumer*, odchylku pak *altit.odchyl*). Názvy objektů nesmí obsahovat mezery (např. nelze vytvořit název *altit prumer*)
- Musíme respektovat přesné psaní velkých a malých písmen (např. *Altit*, *altid* a *ALTID* jsou tři různé názvy pro objekty).
- **`<-, assign`**`(x, value)` – přiřadí název objektu (také =, ale nedoporučuji); *x* zastupuje název, *value* je hodnota, popř. jiný objekt, kterému přiřazujeme název (viz příklad), alternativou je symbol "=" (autoři většiny učebic ale používají ->).
- Napsáním názvu objektu pak tento objekt vypisujeme na konzolu.

```
25 # příklad objektu - číslo 25
[1] 25
"test"   # příklad objektu - text "test"
[1] "test"
xd2 <- 25 # objektu přiřazuji název xd2, je shodné s assign(xd2,25)
xd2 # výpis objektu xd2
[1] 25
Xd2 <- 26 # objektu přiřazuji název Xd2
Xd2
[1] 26
xd2<-Xd2 # hodnotu z objektu Xd2 přiřazuji objektu xd2
xd2
[1] 26
assign(x="my.l",value=25) #jiný způsob přiřazení názvu objektu
my.l
[1] 25
```

Každý objekt přísluší do určité třídy (class) objektů. Podle toho, do které třídy patří je možné s daným objektem manipulovat. R rozlišuje tzv. třídy (class), mody (mode) a typy (typeof) objektů. Obecně ale můžeme prohlásit, že základní dělení je následující:

- **čísla** (numeric)
  - celá (integer) – `2, 3, 4`
  - „reálná“ (double) – `.5, 3.8, 4.3`
  - komplexní (complex) – `4+2i, 25-3i, i`
- **textové řetězce** (character) – `"test", "Ahoj";`
- **datum** (date) – zadání je složitější, ale objekt vypadá následovně `"2006-03-15";`
- **logické hodnoty** (logical) – `TRUE, FALSE`;
- **vektory** (vector) – řady čísel, řetězců nebo jiných dat;
- **faktor** (factor) – speciální typ vektoru s nominálními nebo ordinálními daty (viz dále);
- **matice** (matrix) – klasické matematické matice nebo matice řetězců a jiných dat;
- **pole** (array) – ve své podstatě vícerozměrné matice;
- **seznamy** (list) – specifické objekty sdružující různé jiné objekty různých typů;
- **tabulky dat** (data.frame) – klasické databázové tabulky (viz dále);
- **funkce** (function) – příkazy, kterými spouštíme určitou operaci s objekty;
- **výrazy** (expression), **rovnice** (formula), speciální objekty (výsledky testů, analýz).

Striktní klasifikace objektů má své důvody. Hlavní výhoda spočívá v tom, že řada funkcí (tzv. generic funkce) funguje pro různé objekty různě. Například funkce `summary` nám v případě vektoru (řady) čísel vypočítá základní charakteristiky polohy (minimum,

maximum, medián, průměr a kvartily), zatímco pro vektor faktorů (viz dále) vypíše seznam jednotlivých položek a jejich četnost. Pro začátečníka je tento přístup zpočátku dost matoucí, nicméně použití různých příkazů pro různé typy objektů by bylo zbytečně složité (viz `methods`).

```
summary(c(1,3,4,8)) #shrnutí pro vektor čísel
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.5     3.5     4.0     5.0     8.0

summary(factor(c("a","aa","a","b"))) #shrnutí pro vektor tzv. faktorů
 a aa  b
 2  1  1
```

Naproti tomu, některé funkce striktně vyžadují objekty určité třídy. Jestliže si nejsme jisti typem objektu a nerozpoznáme jej při výpisu na obrazovku (např. při importování dat z jiných programů nebo při použití výsledku určité funkce), můžeme využít funkce, které jsou schopny typ objektu určit:

- **`class`**`(x)` – výpis třídy objektu (vrátí např. „numeric“, „function“, „list“)
- **`mode`**`(x)` – podobné jako class, nerozlišuje však složitější objekty a pouze vypíše jakého typu jsou hodnoty v těchto objektech, např. numeric, complex, logical, character, raw, list
- **`typeof`**`(x)` – podobné jako mode, rozlišuje však některé **podtypy** (např. čísla rozlišuje na celá a reálná)

```
my.x<-2.8 # objekt s názvem my.x obsahuje číslo 2,8 (desetinná tečka)
class(my.x) # třída objektu my.x
[1] "numeric"
mode(my.x) # mode objektu my.x je shodný s třídou
[1] "numeric"
typeof(my.x) # typ objektu my.x ukazuje, že se jedná o reálné číslo
[1] "double"

class(array(1,2,3)) # testuje vytvořené pole
[1] "array"
mode(array(1,2,3)) # určuje že pole je číselné
[1] "numeric"
typeof(array(1,2,3)) # určuje že je vyhrazeno místo pro reálná čísla
[1] "double"

class(class) # určuje třídu funkce class, tedy že class je funkce
[1] "function"
mode(class) # v tomto případě shodný výsledek s třídou
[1] "function"
typeof(class) # zjišťuje, že class je vestavěná (builtin) funkce
[1] "builtin"
```

- **`methods`**`(x)` – vypíše všechny metody přiřazené k již zmiňované generic funkci. Uživatel tak může přímo použít metodu pro určitý objekt nebo ji vyhledat v nápovědě.

```
methods(summary)
 [1] summary.aov            summary.aovlist        summary.connection
 [4] summary.data.frame     summary.Date           summary.default
 [7] summary.ecdf*          summary.factor         summary.glm
[10] summary.infl           summary.lm             summary.loess*
[13] summary.manova         summary.matrix         summary.mlm
[16] summary.nls*           summary.packageStatus* summary.POSIXct
[19] summary.POSIXlt        summary.ppr*           summary.prcomp*
[22] summary.princomp*      summary.stepfun        summary.stl*
[25] summary.table          summary.tukeysmooth*

   Non-visible functions are asterisked
```

Základní typy hodnot

- **Číslo** (numeric). Vypisujeme klasickým způsobem s tím rozdílem, že k oddělení desetinných míst používáme tečku, tzn. „1,25“ zapisujeme „1.25“ (velice důležité, protože v případě chybného psaní vypisuje R syntaktickou chybu!!!!). Pro složitější matematické výpočty je velkou výhodou to, že R počítá také s komplexními čísly. Zápis je podobný jako u reálných čísel, k imaginárnímu členu přidáváme písmeno i (např. 2+3i).

```
1.25 #číslo 1,25
[1] 1.25
120000000 #12 x 10^8
[1] 1.2e+08
0.000000001 #1 x 10^-9
[1] 1e-09
17+2i #komplexní číslo
[1] 17+2i
2e-2 # 2 x 10^-2
[1] 0.02
5.4e6 # 5,4 x 10^6
[1] 5400000
1,5 # chyba syntaxe protože jsme pro číslo použili desetinnou čárku
Error: syntax error in "1,"
```

- **Znak, řetězec** (character, string). Pro výpisy výsledků, označení názvů sloupců u tabulek, popř. psaní atributů musíme používat znaky nebo textové řetězce. Stejně jako v jiných programech a jazycích i zde musí být řetězec v uvozovkách (jednoduchých nebo klasických). Znak a řetězec bývá také často použit pro zvláštní objekt, který se nazývá **factor**. Jedná se o hodnoty dat na nominální škále (tzv. nominální proměnná). Přesný popis tohoto typu objektu bude v následující kapitole.

```
"Petr" # řetězec Petr je v uvozovkách, jinak je považována za název objektu
[1] "Petr"
Petr # není v uvozovkách, proto R považuje slovo Petr za název objektu
Error: object "Petr" not found
'Ahoj' # mohou být použity také jednoduché uvozovky
[1] "Ahoj"
```

- **Logická hodnota** (logical). Logické hodnoty se používají pro některé atributy funkcí, jsou výsledkem testování výrazů (např. zda 5 > 1) nebo se používají pro indexování vektorů (vše viz výše). Existují dvě základní logické hodnoty pravda (*TRUE*, *T*) a nepravda (*FALSE*, *F*). Hodnoty jsou opět „case sensitive“ tedy citlivé na velká a malá písmena, proto je nelze psát jinak než uvedeným způsobem.

```
F # zkrácený způsob zápisu FALSE (nepravda)
[1] FALSE
T # zkrácený způsob zápisu TRUE (nepravda)
[1] TRUE

FALSE # obecný způsob zápisu FALSE (nepravda)
[1] FALSE
False # chybný způsob zápisu FALSE (nepravda), jsou použita malá písmena
Error: object "False" not found
```

- **Datum** (date). Zápis pro datum je trochu složitější a vyžaduje další znalosti syntaxe R. Vzhledem k tomu, že se datum pro analýzy dat většinou nepoužívá (spíše např. den v roce atd.), nebudeme zápis podrobněji rozebírat.

```
as.Date("2007-10-16") #základní styl zápisu data (v mezinárodním systému)
[1] "2007-10-16"
as.Date("16.10.2007",format="%d.%m.%Y")# styl zápisu data ve zvoleném systému
[1] "2007-10-16"
```

- **Vzorec (**formula**) –** označuje různé druhy vzorců pro modely, grafy a některé výpočty. Základní zápis je y~x, což znamená „y závislé na x“. Podrobnosti je nutné znát až při tvorbě grafů a analýze závislosti (viz podkapitola další důležité grafy).

```
y~x
y ~ x
class(y~x)
[1] "formula"
```

- **Chybějící a další speciální hodnoty. `NA`**: chybějící hodnota (např. při načítání dat), **`Inf`**: nekonečno (např. při 1/0), **`NaN`**: not a number, nečíselná hodnota (např. při dělení 0/0), **`NULL`**: reprezentuje nulový objekt.

```
1/0
[1] Inf
-5/0
[1] -Inf
x<-NA
x
[1] NA
0/0
[1] NaN
```

Průvodce
Jak jste si jistě všimli v textu, z uvedených typů hodnot budou pro nás prozatím podstatné pouze čísla, řetězce a logické hodnoty. Jejich styl zápisu musíme znát nejen při zadávání dat, ale také pro specifikaci argumentů jednotlivých funkcí.

Tvorba vektorů a dalších jednoduchých objektů

- **Vektor** (variační řada, vector). Vektory představují množinu hodnot (čísel, znaků, řetězců, datum, logické hodnoty, faktor atd.), kde každý prvek má své pořadí. Jsou to asi nejpoužívanější objekty v R, proto je velice důležité se s nimi naučit dobře pracovat. Obecně pracujeme s vektory jako s variační řadou ve statistice. Při vytváření vektorů si musíme být vědomi toho, že nelze vytvořit jeden vektor obsahující současně různé typy hodnot (např. čísla a znaky nebo logické hodnoty, viz příklad). Základní zadávání vektorů je následující:
  - `c(..., recursive=F)`. Sloučí hodnoty v závorce do vektoru (řady). Typ výsledného vektoru závisí na zadaných hodnotách.
    - `recursive` používá se pro tzv. seznamy (viz *`help (c)`*)

```
c() #nulový objekt
NULL
c(1,2,4,8) # číselný vektor
[1] 1 2 4 8
c("ab","ad","dd","lc") # vektor řetězců
[1] "ab" "ad" "dd" "lc"
c(T,T,T,F,T,F) # vektor logických znaků
[1]  TRUE  TRUE  TRUE FALSE  TRUE FALSE

x<-c(4,6,9,3) #vektoru přiřadím název x
y<-c(5,9,7,5) #vektoru přiřadím název y
c(x,y) # z vektoru x a y vytvoříme jeden vektor
[1] 4 6 9 3 5 9 7 5

c(c(1,2,3),c(4,5,6))#dva vektory sloučené v jeden
[1] 1 2 3 4 5 6

c(1,2,T,F) # všechny hodnoty jsou převedeny na čísla
[1] 1 2 1 0
```

```
c(1,2,"a",F) # jedna z hodnot je znak takže vektor bude znakový
[1] "1"     "2"     "a"     "FALSE"
```

- **vector**(mode="logical",length=0). Vytvoří vektor daného typu (*mode* – viz funkce mode), dané délky *length* (tzn. počet členů/prvků řady) s nulovými hodnotami (*FALSE* pro logical, prázdný řetězec "" pro character, *0+0i* pro complex, atd.)

```
vector() #přednastavené hodnoty – logický vektor, počet prvků = 0)
logical(0)
vector(length=3) #vektor logických hodnot o 3 prvcích
[1] FALSE FALSE FALSE
vector(mode="integer",length=3) #vektor celých čísel o 3 prvcích
[1] 0 0 0
vector(mode="complex",length=5) #vektor komplexních čísel
[1] 0+0i 0+0i 0+0i 0+0i 0+0i
vector(mode="character",length=3) #vektor znaků o 3 prvcích
[1] "" "" ""
```

- **sequence**(nvec). Vytvoří řadu celých čísel od 1 do čísla (čísel) uvedeného v *nvec* (pro každé uvedené číslo).

```
sequence(5) # vektor celých čísel od 1 do 5
[1] 1 2 3 4 5
sequence(-5) # vektor od 1 do -5
[1]  1  0 -1 -2 -3 -4 -5
sequence(c(10,4)) #vektor od 1 do 10 a od 1 do 4
 [1]  1  2  3  4  5  6  7  8  9 10  1  2  3  4
```

- **a:b**. Vytvoří aritmetickou posloupnost celých čísel od čísla *a* do čísla *b*.

```
5:10 # vektor celých čísel od 5 do 10
[1]  5  6  7  8  9 10
5:1 # od 5 do 1
[1] 5 4 3 2 1
-20:-11 # od -20 do -11
 [1] -20 -19 -18 -17 -16 -15 -14 -13 -12 -11
 c(1:4) # shodne s 1:4
[1] 1 2 3 4
```

- **seq**(from=1,to=1,by=((to-from)/(length.out-1)),length.out= NULL,along.with=NULL,…). Vytvoří sekvenci od *from* do *to*, po dílcích (*by*), o výsledné délce (*length.out*) nebo o délce shodné s délkou vektoru v *along.with*. (podobné funkce viz seq.int, seq_along, seq_len).

```
seq(5) #klasická sekvence od 1 do 5 (jako sequence)
[1] 1 2 3 4 5
seq(5,9) # sekvence od 5 do 9 (jako 5:9)
[1] 5 6 7 8 9
seq(5,9,by=0.5) # sekvence od 5 do 9 po 0.5 dílcích
[1] 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5 9.0
seq(from=5,by=0.2,length.out=18) # 18 čísel od 5 po 0.2 dílcích
[1] 5.0 5.2 5.4 5.6 5.8 6.0 6.2 6.4 6.6 6.8 7.0 7.2 7.4 7.6 7.8 8.0 8.2
8.4
seq(from=5,by=2,length.out=10) # 10 čísel od 5 po 2 dílcích
 [1]  5  7  9 11 13 15 17 19 21 23
seq(from=5,by=0.2,along.with=5:10)# od 5 po 0.2 o délce 6 hodnot
[1] 5.0 5.2 5.4 5.6 5.8 6.0
```

- **rep**(x,times=1,length.out=NA,each=1). Vytvoří vektor opakování objektu *x* (hodnota nebo vektor). Argument *times* udává počet opakování *x*,
  - *length.out* – délka výsledného vektoru,
  - *each* – počet opakování každého prvku (v případě, že x je vektor, viz příklady)

```
rep(12,times=6) #6x opakování čísla 12
[1] 12 12 12 12 12 12
```

```
rep("a",times=6) #6x opakování písmene "a"
[1] "a" "a" "a" "a" "a" "a"
rep(1:3,times=6) #6x opakování vektoru s prvky 1,2,3
 [1] 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3
rep(1:3,each=2) #2x opakování každého prvku
[1] 1 1 2 2 3 3
rep(1:3,times=3, each=2) #2x opakování každého prvku a 3x celého vektoru
 [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3
rep(1:3,times=3, each=2,length.out=15)#totéž, ale délka pouze 15 prvků
 [1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2
rep(c(1,4,10),times=3) #3x opakování vektoru s uvedenými prvky
[1]  1  4 10  1  4 10  1  4 10
rep(c(1,4,10),times=c(1,3,2)) #1x první prvek, 3x druhý prvek, 2x třetí
prvek
[1]  1  4  4  4 10 10
rep(c(1,4,10),times=c(2,3)) #chybné zadání vektoru opakování (2 prvky)
Error in rep(c(1, 4, 10), times = c(2, 3)) :
        invalid 'times' argument
```

Kontrolní úkoly

1. Vytvořte vektor s názvem *hodnoty*, který bude obsahovat tato data: 1, 1.5, 3.4, 15, 12, 6.4
2. K vektoru *hodnoty* přidejte čísla 5 a 1 a uložte do vektoru *hodnoty*
3. Vytvořte objekt *my.stat*, ve kterém bude slovo: statistika
4. Vytvořte vektor s názvem *zeme* a s hodnotami: Asie, Evropa, Amerika, Evropa, Amerika, Amerika, Evropa
5. Vytvořte vektor čísel od 15 do -15.
6. Vytvořte vektor čísel od 0 do 1 s přírůstkem 0.05.
7. Vytvořte vektor čísel od -5 do -2 s přírůstkem -0.5.
8. Vytvořte vektor čísel od 10 do 2 snižující se o 0.5.
9. Vytvořte vektor, ve kterém se budou opakovat hodnoty „a“ a „b“ 20×.
10. Vytvořte vektor, ve kterém se bude opakovat 20× znak „a“ a pak 20× znak „b“
11. Vytvořte vektor, ve kterém se bude opakovat 10× znak „a“ a pak 5× znak „b“
12. Vytvořte vektor, ve kterém se bude opakovat 3× číslo 5, 4× číslo 6 a 5× číslo 7.
13. Vytvořte objekt *my.l*, do kterého uložíte vektor lichých čísel od 27 do 59.
14. Zdvojte hodnoty uložené v objektu my.l tak aby šly 27, 27, ... 59,59. Následně zdvojte znovu hodnoty tak aby šly 27...59, 27...59.

- **`matrix`** `(data=NA,nrow=1,ncol=1,byrow=F,dimnames=NULL)` – vytvoří matici hodnot (*`data`*), o *`nrow`* počtu řádků, *`ncol`* počtu sloupců, názvy dimenzí (*`dimnames`*) jsou vytvořeny jako seznam. Jestliže je hodnot v data méně než má být prvků v matici, pak se budou po použití posledního prvku vektoru v data znovu opakovat od první položky (viz příklad).
  - *`byrow`* – je logický argument, zda hodnoty uvedené v data mají být nejprve vypisovány do řádku (popř. do sloupce).

```
matrix(5,nrow=4,ncol=6)# matice o 4 řádcích a 6 sloupcích
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    5    5    5    5    5    5
[2,]    5    5    5    5    5    5
[3,]    5    5    5    5    5    5
[4,]    5    5    5    5    5    5
matrix(1:6,2,3) # vytvoří matici 2×3, hodnoty jsou řazeny po sloupcích
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

# v případě, že je hodnot málo, jsou opětovně doplňovány z vektoru
matrix(1:4,2,3,byrow=TRUE)#hodnoty jsou po řádcích a opakují se znovu
     [,1] [,2] [,3]
```

```
[1,]    1    2    3
[2,]    4    1    2
Warning message:
data length [4] is not a sub-multiple or multiple of the number of
columns [3] in matrix

matrix(c("a","b","a","d","a","c"),2,3,T)#podobné, ale hodnoty jsou řetězce
     [,1] [,2] [,3]
[1,] "a"  "b"  "a"
[2,] "d"  "a"  "c"

#tvorba názvů dimenzí v matici pomocí seznamu (viz dále), hodnoty jsou logické
matrix(c(T,T,T,F,F,T),2,3,T,dimnames=list(sex=c("muž","žena"),otazka=1:3))
      otazka
sex        1     2    3
  muž   TRUE  TRUE TRUE
  žena FALSE FALSE TRUE
```

- **array** (data=NA, dim=length(data),dimnames=NULL). Vytvoří pole, tedy jedno-vícerozměrnou matici dat s hodnotami v *data* a rozsahem jednotlivých dimenzí podle vektoru v *dim* a názvech dimenzí v *dimnames* (zapsáno jako seznam – list, viz dále) (pro pochopení pojmu pole je nejvhodnější uvést konkrétní příklad, viz příklady).

```
# tvorba dvojrozměrného pole (1. dimeze=4, 2. dimenze = 3
# s hodnotami 1-12
array (1:12, c(4, 3))
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    2    6   10
[3,]    3    7   11
[4,]    4    8   12

# dvojrozměrné pole o 4 řádcích a 2 sloupcích, s názvy dimenzí vz (vzorek 1-4)
# a sex (pohlaví m=muž, z=žena)
array (c(1,5,3,7,8,9),c(4, 2),dimnames=list(vz=1:4,sex=c("m","z")))
   sex
vz  m z
  1 1 8
  2 5 9
  3 3 1
  4 7 5

#Studujeme barevnost skvrn (light – světlá, medium – střední, dark – tmavá
#na krovkách u samců (male) a samic (female) určitých brouků na 4 různých
#lokalitách (L1,L2,L3,L4). V tabulce chceme zpřehlednit počty jedinců daného
#pohlaví s určitými křídly na každé lokalitě. Počty jedinců pro každou #kombinaci
můžeme zobrazit pomocí trojrozměrného pole.

x1<-array(c(12,3,5,4,11,5,4,1,7,3,6,6,10,4,8,4,12,7,9,9,14,10,2,3), dim =
c(3,4,2), dimnames=list(color=c("light","medium","dark"), locality =
c("L1","L2","L3","L4"), sex=c("male","female")))
x1
, , sex = male

        locality
color    L1 L2 L3 L4
  light  12  4  4  3
  medium  3 11  1  6
  dark    5  5  7  6

, , sex = female

        locality
color    L1 L2 L3 L4
  light  10  4  9 10
  medium  4 12  9  2
  dark    8  7 14  3
```

```
array (3, c(1, 4)) #jednorozměrné pole o 1 řádku a 4 sloupcích
     [,1] [,2] [,3] [,4]
[1,]    3    3    3    3

array (3, c(4, 1)) #jednorozměrné pole o 1 sloupci a 4 řádcích
     [,1]
[1,]    3
[2,]    3
[3,]    3
[4,]    3
```

- **`list`** `(tag=value, ...)`. Objekt seznam je zvláštní typ objektu, se kterým se setkáváme buď při zadávání některých argumentů funkcí (např. dimnames u array) nebo spíše jako s výstupy některých funkcí. Do tohoto objektu můžeme uložit čísla, vektory, pole, matice i řetězce najednou. Seznamy mohou mít názvy jednotlivých položek (*tag* – nepovinné) a v rámci nich jsou uloženy hodnoty (v každé položce pouze jeden typ). V případě, že chceme nechat položku prázdnou (pouze tag), pak používáme funkci **`alist`** hodnota v dané položce je pak NULL

```
#tvorba seznamu tří vektorů různé délky (bez názvů položek)
list(c(3,2),c(4,5,7),c(5,4,2,7))
[[1]]
[1] 3 2

[[2]]
[1] 4 5 7

[[3]]
[1] 5 4 2 7

#tvorba seznamu matice a dvou vektorů různé délky a typu s názvy položek
list(matice=matrix(1:4,2,2),typ=c("součet","rozdíl","součin"),
     vysledek=c(4,-2,3,6,-2,8))
$matice
     [,1] [,2]
[1,]    1    3
[2,]    2    4

$typ
[1] "součet" "rozdíl" "součin"

$vysledek
[1]  4 -2  3  6 -2  8

list(t1=,t2=1:3)#vytvoří pouze t2, protože je neprázdná
Warning: an element is empty and has been omitted
the part of the args list of 'list' being evaluated was:
   (t1 = , t2 = 1:3)
$t2
[1] 1 2 3

alist(t1=,t2=2)#vytvoří také prázdnou položku t1
$t1

$t2
[1] 2

#vytvoří "seznam v seznamu" (v položce hodnoty je seznam vektorů a,b)
list(hodnoty=list(a=1:4,b=5:15),lokality=c("Beskydy","Jeseníky"))
$hodnoty
$hodnoty$a
[1] 1 2 3 4

$hodnoty$b
 [1]  5  6  7  8  9 10 11 12 13 14 15
```

```
$lokality
[1] "Beskydy"  "Jeseníky"
```

- **data.frame** (tag=values, ..., row.names, check.rows=F, check.names=T, stringsAsFactors=T). Data frame je klasická dvojrozměrná tabulka dat, kde narozdíl od pole a matice mohou jednotlivé sloupce (pole) obsahovat různé typy dat (ale v jednom sloupci nanejvýš jeden typ). Zápis tabulky dat je podobný jako u seznamu s tím rozdílem, že počet hodnot ve sloupcích musí být shodný pro všechny sloupce. Obecně většinou data nejsou zadávána tímto způsobem, ale do formátu data.frame jsou importována např. z MS Excel nebo vytvářena ve speciálním vestavěném editoru.
  - *row.names* může obsahovat vektor s názvy řádků.
  - *check.rows* a *check.names* jsou logické argumenty pro kontrolu správnosti řádků a názvů sloupců (např. se názvy sloupců nesmí dublovat).
  - *tag* – název sloupce je nepovinný, stačí zadat pouze hodnoty (*values*).
  - *stringsAsFactors* – argument standardně nastaven na TRUE, tzn. v případě, že zadáme sloupec znaků, pak je převeden na typ faktor (viz následující objekt typu factor), což většinou není na závadu, protože předpokládáme, že ve sloupci je určitý statistický znak). Jestliže přesto chceme převodu zabránit, stačí nastavit *stringsAsFactors=F*.

```
#tabulka se sloupci (poli) nr (číslo), typ (faktor) a polut (logická hodnota)
#vstupní data sloupce typ jsou řetězce, ale jsou převedeny na faktory
data.frame(nr=5:9,typ=c("a","b","c","d","e"),polut=c(T,F,T,T,F))
  nr typ polut
1  5   a  TRUE
2  6   b FALSE
3  7   c  TRUE
4  8   d  TRUE
5  9   e FALSE

#zadání v případě opakování některých údajů ve sloupcích, povšimněte si, že
#u funkce seq byl použit argument length.out jen zkráceně jako len
data.frame(exp=rep(1:3,2),latka=rep(c("a","b"),3),hodn=seq(3,by=0.5,len=6))
  exp latka hodn
1   1     a  3.0
2   2     b  3.5
3   3     a  4.0
4   1     b  4.5
5   2     a  5.0
6   3     b  5.5

#dva sloupce se stejným názvem
data.frame(exp=rep(1:4),exp=rep(1:2,2),hodn=seq(3,by=0.5,len=4),check.names=F)
  exp exp hodn
1   1   1  3.0
2   2   2  3.5
3   3   1  4.0
4   4   2  4.5

#check.names detekovalo stejné názvy a vytvořilo jiný název pro druhý sloupec
data.frame(exp=rep(1:4),exp=rep(1:2,2),hodn=seq(3,by=0.5,len=4),check.names=T)
  exp exp.1 hodn
1   1     1  3.0
2   2     2  3.5
3   3     1  4.0
4   4     2  4.5
```

- **factor** (x=character(),levels=sort(unique.default(x),na.last=TRUE),labels=leve

`ls,exclude=NA,ordered=is.ordered(x))`. Ve své podstatě je to vektor znaků (statistických), které jsou nominální (například pohlaví – samec, samice; typ použitého hnojiva – fosfor, vápník, dusík) nebo ordinální (velikost – menší, střední, větší). Pochopení objektu factor je důležité a bude se ještě dále upřesňovat.

- *levels* – nastavení možných hodnot faktoru (úrovní). Jedná se o hodnoty, kterých vektor může nabývat, ostatní jsou změněny na NA hodnotu. V případě, že položku nepoužijeme, nastaví se levels automaticky tak, že jsou vybrány všechny možné hodnoty obsažené ve vektoru a ty jsou seřazeny.
- *labels* – mění názvy hodnot v pořadí, v jakém jsou uvedeny v levels.
- *exclude* – při tvorbě objektu je možno předem určité hodnoty ignorovat, ty jsou pak , např. vektor hodnot v exclude vynechává automaticky NA. Tomu lze zabránit nastavením exclude na NULL.
- *ordered* – slouží k označení, zda je faktor čistě nominální proměnná nebo ordinální.

```
factor(c(1,1,2,1,3,4,4,2,1))#tvorba jednoduchého objektu typu faktor
[1] 1 1 2 1 3 4 4 2 1
Levels: 1 2 3 4
factor(c(1,1,2,1,3,4,4,2,1),levels=4:1) #jiné řazení hodnot
[1] 1 1 2 1 3 4 4 2 1
Levels: 4 3 2 1
factor(c(1,1,2,1,3,4,4,2,1),levels=4:1,labels=c(40,30,20,10)) #jiné názvy
[1] 10 10 20 10 30 40 40 20 10
Levels: 40 30 20 10
factor(c(1,1,2,1,3,4,4,2,1),levels=4:1,labels=c("a","b","c","d"))#jiné názvy
[1] d d c d b a a c d
Levels: a b c d
factor(c(1,1,2,1,3,4,4,2,1,NA,NA)) #NA v levels nezařazeno (automat. v exclude)
 [1] 1    1    2    1    3    4    4    2    1    <NA> <NA>
Levels: 1 2 3 4
factor(c(1,1,2,1,3,4,4,2,1,NA,NA),exclude=NULL)#do levels je zařazeno také NA
 [1] 1    1    2    1    3    4    4    2    1    <NA> <NA>
Levels: 1 2 3 4 <NA>
factor(c(1,1,2,1,3,4,4,2,1),exclude=1)#nezahrnout 1, převedena na NA
[1] <NA> <NA> 2    <NA> 3    4    4    2    <NA>
Levels: 2 3 4
factor(c(1,1,2,1,3,4,4,2,1),ordered=T)#tvorba ordinálního faktoru
[1] 1 1 2 1 3 4 4 2 1
Levels: 1 < 2 < 3 < 4

# i když faktor vypadá jako soubor čísel, po vynásobení vrátí chybu, protože se
# o čísla nejedná
factor(c(1,1,2,1,3,4,4,2,1))*3
[1] NA NA NA NA NA NA NA NA NA
Warning message:
* not meaningful for factors in: Ops.factor(factor(c(1, 1, 2, 1, 3, 4, 4, 2, 1)),
3)
```

Kontrolní úkoly

15. Vytvořte matici 3 řádky a 4 sloupce ve které budou pouze čísla 1.
16. Vytvořte stejnou matici, ve které se budou opakovat čísla 10, 20, 30 za sebou ve sloupcích. Následně vytvořte matici, kde e bude opakovat v řádcích 10, 20, 30, 40.
17. Vytvořte matici, ve které budou názvy řádků země ČR a SR (proměnná země) a ve sloupcích proměnná plocha s hodnotami pevnina a voda. Vlastní data potom budou údaje o rozloze 77 276, 49 036, 1 590, NA.
18. Vytvořte pole 9x3 s hodnotami 1:27.
19. Vytvořte trojrozměrné pole 3x4x2 s hodnotami 12:36

20. Vytvořte pole s dimenzemi 2x2x3, kde 1. dimenze je označení nemoc – nakažen, nenakažen, 2. dimenze je pohlaví – muž, žena, 3. dimenze je kontinent – Afrika, Amerika, Evropa, hodnoty jsou 55,12,38,12,16,14,28,31,26,5,42,36
21. Vytvořte seznam, kde v první položce nazvané *popis* bude textový řetězec "výsledek testu", v druhé položce nazvané *hodnota* bude vektor čísel 21,34,25
22. Vytvořte seznam, ve kterém budou uloženy výsledky předchozích úkolů 15:20.
23. Vytvořte tabulku dat *obyv* ve které budou sloupce *country*, *population* a *percent* s hodnotami pro 1. sloupec Čína, Indie, EU, USA, Indonésie, pro 2. sloupec 1320955000, 1169016000, 492964961, 303004000, 231627000, pro 3. sloupec 19.8, 17.52, 7.39, 4.54, 3.47.
24. Vytvořte vektor hodnot faktoru 1,2,1,1,2,1,2,1,1
25. Změňte zadání vektoru tak, aby byla hodnota 2 považována za první v pořadí
26. Upravte předchozí zadání tak, že místo hodnoty 1 se bude vypisovat "žena" a místo 2 "muž".

Převod objektů

V případě, že je nutné určitý objekt převést na jiný typ, je k dispozici velké množství transformačních funkcí a funkcí, které testují daný objekt podobně jako funkce class, ale dotazují se na konkrétní typ objektu a vrací logickou hodnotu.

- **`is.`**typ objektu – testuje, zda je daný objekt dané třídy (typu) (viz `methods(is)`)
- **`as.`** – typ objektu převede objekt na objekt dané třídy (viz `methods(as)`)

```
x<-c(1:5)
is.vector(x) # je daný objekt typu vektor? ano
[1] TRUE
is.integer(x) # obsahuje pouze celá čísla? ano
[1] TRUE
is.list(x) # je daný objekt typu seznam? ne
[1] FALSE
as.list(x) # převeď objekt x na seznam
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

[[4]]
[1] 4

[[5]]
[1] 5

x<-as.list(x) # převeď objekt x na seznam
is.list(x)
[1] TRUE
```

Vyhledávání objektů a čtení jejich struktury

Všechny objekty, které v průběhu práce vytváříme, se ukládají do paměti a můžeme s nimi kdykoliv manipulovat. Vytvořené objekty jsou součástí tzv. workspace (pracovního prostoru), který celý ukládáme na harddisk (menu *File – Save Workspace*). Vytvoříme tak soubor s koncovkou .R. Tento soubor lze kdykoliv otevřít a objekty v něm uložené dále

používat. Jestliže zavřeme R bez uložení, pak při dalším spuštění již námi vytvořené objekty nejsou k dispozici.

Pro práci s programem R a objekty je důležité kromě typu objektu znát několik dalších operací, které nám práci usnadní. Jsou to zejména:

- **ls**(name, pattern, all.names=F) – je shodné s **objects** – vypíše objekty v daném prostředí (workspace, popř. speciální dle čísla nebo názvu – *name*). Pro naše potřeby stačí ls() – výpis objektu, které jsou aktuálně v daném prostředí (workspace)
  - *pattern* – vyhledávací výraz (co hledat, např. objekty začínající na „a“,
  - *all.names* – vyhledává také objekty začínající tečkou

```
#nejprve uložíme do workspace několik objektů
x1<-25
x3<-26
cmy<-"ahoj"
x<-29
ls() #vypíše všechny aktuálně existující objekty
[1] "cmy" "x"   "x1"  "x3"
objects() # shodné s ls()
[1] "cmy" "x"   "x1"  "x3"
#příklad vyhledávání objektů
a<-1; b<-4; aa<-2; b.a<-4; .a<-"Petr"
ls()
[1] "a"   "aa"  "b"   "b.a"
ls(pat="a",all.names=T) #vypisuje také objekty začínající tečkou
[1] ".a"  "a"   "aa"  "b.a"
```

- **rm**(..., list , pos = -1, envir, inherits = F) – maže vybrané objekty zadané buď přímo nebo jako vektor názvů (*list*) v daném prostředí (envir), inherits – prohledá také všechny vazby (viz ?rm)

```
x1<-15;x2<-15;x3<-15;x4<-15; ls()
[1] "x1" "x2" "x3" "x4"
rm(x1,x2);ls() #dva objekty jsou vymazány
ls() #seznam aktuálních objektů
[1] "x"  "x3" "x4" "y"  "z"

rm(list=ls());ls() #návod, jak vymazat všechny objekty
character(0)
```

- **str**(x): přehledně vypíše podrobnou vnitřní strukturu objektu *x*

```
# tvorba tabulky dat (data.frame) se třemi sloupci
x1<-data.frame(exp=rep(1:3,2),latka=rep(c("a","b"),3),hodn=seq(3,by=0.5,len=6))
str(x1) # výpis struktury sloupců (6 řádků, 3 sloupce atd.)
'data.frame':   6 obs. of  3 variables:
 $ exp  : int  1 2 3 1 2 3
 $ latka: Factor w/ 2 levels "a","b": 1 2 1 2 1 2
 $ hodn : num  3 3.5 4 4.5 5 5.5
```

- **comment**(x) – nastaví nebo vypíše komentář k danému objektu *x*

```
comment(x1)<-c("Experiment testování kalcia","2002"); comment(x1)
[1] "Experiment testování kalcia" "2002"
```

- **attributes**(x) – objekty mají různé vlastnosti, které lze dále zkoumat a nastavovat. Příslušná funkce vypisuje všechny atributy objektu *x*

```
attributes(x1)
$names
[1] "exp"   "latka" "hodn"

$row.names
[1] 1 2 3 4 5 6

$class
[1] "data.frame"
```

```
$comment
[1] "Experiment testování kalcia" "2002"
```

- **attr**(x, which) – nastaví nebo přečte danou vlastnost (atribut *which*) objektu *x*

```
attr(x1,"class")
[1] "data.frame"
```

- **demo**(topic) – spustí nebo zobrazí ukázkový program (*topic*)

```
demo() #vypíše informační okno o dostupných demo programech
demo(image) #spuštění ukázky
demo(graphics)
demo (persp)
demo (plotmath)
```

- **data**(..., list, package, lib.loc, verbose, envir) – vypíše ukázková data v dané knihovně (*package*).

```
data()
```

- **summary**(objekt, maxsum, digits, ...) – celkový přehled o určitém objektu (podle typu objektu), *maxsum* – kolik úrovní bude vypsaných, *digits* – počet desetinných míst

```
summary(1:25) #souhrn pro vektor čísel
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1       7      13      13      19      25

#souhrn pro již použitou tabulku x1 je souhrnem pro každou položku
#vidíme, že exp je číslo a ne faktor (i když by měl správně být)
summary(x1)
      exp       latka      hodn
 Min.   :1.00   a:3   Min.   :3.000
 1st Qu.:1.25   b:3   1st Qu.:3.625
 Median :2.00         Median :4.250
 Mean   :2.00         Mean   :4.250
 3rd Qu.:2.75         3rd Qu.:4.875
 Max.   :3.00         Max.   :5.500

summary(c("a","b","a","a","b")) #souhrn pro vektor znaků
   Length     Class      Mode
        5 character character

#tvorba seznamu a jeho souhrn
x<-list(hodnoty=list(a=1:4,b=5:15),lokality=c("Beskydy","Jeseníky"))
x
$hodnoty
$hodnoty$a
[1] 1 2 3 4

$hodnoty$b
 [1]  5  6  7  8  9 10 11 12 13 14 15


$lokality
[1] "Beskydy"  "Jeseníky"

summary(x) #souhrn pro seznam
         Length Class  Mode
hodnoty  2      -none- list
lokality 2      -none- character

summary(matrix(1:10,5,2)) #souhrn pro matici (hodnoceny sloupce)
       V1          V2
 Min.   :1   Min.   : 6
 1st Qu.:2   1st Qu.: 7
 Median :3   Median : 8
```

```
 Mean   :3   Mean   : 8
 3rd Qu.:4   3rd Qu.: 9
 Max.   :5   Max.   :10
```

Kontrolní úkoly

27. Zkontrolujte, které objekty máte aktuálně ve workspace. V případě, že žádné nejsou, vytvořte několik vektorů a matic z předchozích příkladů a nazvěte je podle svého uvážení.
28. Matici z příkladu 15 převeďte na pole. Změní zásadně svou strukturu?
29. Prostudujte strukturu jednotlivých objektů, které máte aktuálně vytvořeny ve workspace.
30. Podívejte se na summary pro objekt *obyv* z příkladu 26.
31. Vymažte objekt *obyv*.

Práce s knihovnami v R

I když by se z nápovědy zdálo, že základní nabídka funkcí v R je nepřeberná, řada složitějších nebo specializovaných analýz (např. funkce pro tvorbu taxonomických analýz, mnohorozměrná analýza dat atd.) je umístěna v dodatečných balíčcích (knihovnách). K zjišťování aktuálně načtených knihoven, popř. k jejich načtení slouží řada funkcí z nichž jsou nejdůležitější následující dvě.

- **`library`**`(package, help, pos=2, lib.loc=NULL, character.only=F, logical.return=F, warn.conflicts=T, keep.source, verbose, version)` – vypíše do informačního okna, které knihovny jsou k dispozici, pomocí této funkce lze také přiinstalovat nebo odinstalovat dané doplňkové balíky (package).

```
library(gam)#přiinstaluje knihovnu gam (vyžaduje také knihovnu splines)
Loading required package: splines
```

- **`search`**`()` – vyhledá přiinstalované knihovny, popř. objekty typu `environment` atd., umístění jednotlivých prvků vyhledá `searchpaths()`. Podobná funkce `(.packages())`.

```
search()
[1] ".GlobalEnv"        "package:methods"   "package:stats"
[4] "package:graphics"  "package:grDevices" "package:utils"
[7] "package:datasets"  "Autoloads"         "package:base"
searchpaths()
[1] ".GlobalEnv"
[2] "C:/PROGRA~1/R/R-24~1.0/library/methods"
[3] "C:/PROGRA~1/R/R-24~1.0/library/stats"
[4] "C:/PROGRA~1/R/R-24~1.0/library/graphics"
[5] "C:/PROGRA~1/R/R-24~1.0/library/grDevices"
[6] "C:/PROGRA~1/R/R-24~1.0/library/utils"
[7] "C:/PROGRA~1/R/R-24~1.0/library/datasets"
[8] "Autoloads"
[9] "C:/PROGRA~1/R/R-24~1.0/library/base"
```

- **`attach`** `(what,pos=2,name=deparse(substitute(what)),warn.conflicts=T))`, **`detach`**`(name,pos=2,version,unload=F)` – zjednodušuje práci zejména s objekty data.frame a list tím, že "připojí" (detach odpojí) objekt. Jednotlivé položky objektu pak můžeme používat bez specifikace objektu.

```
x<-list(pol1=2:5,pol2=3:8) #vytvoří seznam se dvěma položkami
pol1 #položku nelze vypsat bez použití x$pol1
Error: object "pol1" not found
attach(x) #připojení objektu
pol1 #položku lze vypsat bez specifikace objektu
[1] 2 3 4 5
```

Shrnutí:

Objektem v prostředí R mohou být různé struktury – od jednotlivých proměnných (číselných, textových atd.) až po funkce. Pro zjednodušení práce s objekty jim většinou přiřazujeme podle určitých pravidel názvy (jako názvy proměnné). Typ objektu poznáme buď přímo jeho výpisem, popř. identifikací pomocí funkcí. Při tvorbě objektů (v případě, že je neimportujeme, což je většinou mnohem jednodušší) používáme speciální funkce pro každý typ objektu. Následně je lze na některé jiné typy objektů převádět. Všechny vytvořené objekty jsou součástí tzv. pracovního prostoru. V rámci pracovního prostoru mohu získat seznam všech uložených objektů, objekty mazat atd. Při manipulaci s objekty jsou velice důležité funkce umožňující číst jejich strukturu a komentáře k jednotlivým objektům. V případě, že některé pokročilé funkce nejsou momentálně k dispozici (nejsou načteny patřičné knihovny), lze knihovny (balíčky, package) je jednoduchým způsobem stáhnout z WWW, přiinstalovat (viz předchozí kapitoly) a načíst do aktuálního prostředí.

Metody k zapamatování:

- identifikace objektu – class, mode, typeof
- tvorba vektoru – c, rep, seq, sequence, a:b
- tvorba dalších objektů – matrix, array, list, data.frame. factor
- převod a testování objektů – as.typ objektu, is.typ objektu
- výpis a mazání objektů – ls, rm
- struktura objektů, komentáře, vlastnosti – str, summary, comment, attributes, attr
- pomocné funkce – demo, data
- seznam aktuálních knihoven a načítání knihoven: library, search

Výsledky

```
1.  hodnoty <- c(1, 1.5, 3.4, 15, 12, 6.4)
2.  hodnoty <- c(hodnoty, 5, 1)
3.  my.stat <- "statistika"
4.  zeme <- c("Asie", "Evropa", "Amerika", "Evropa", "Amerika", "Amerika", "Evropa")
5.  15:-15
6.  seq(0,1,by=0.05)
7.  seq(-5,-2,by=0.5)
8.  seq(10,2,by=-0.5)
9.  rep(c("a","b"),times=20)
10. rep(c("a","b"),each=20)
11. rep(c("a","b"),times=c(10,5))
12. rep(5:7,times=3:5)
13. my.l <- seq(27,59,2)
14. rep(my.l,each=2); rep(my.l,times=2)
15. matrix(1,3,4)
16. matrix(c(10,20,30),3,4); matrix(c(10,20,30,40),3,4, byrow=T)
17. matrix(c(77276, 49036, 1590, NA),2,2,dimnames= list(zeme=c("ČR","SR"),
    plocha=c("pevnina","voda")))
18. array(1:27,c(9,3))
19. array(12:36,c(3,4,2))
20. array(c(55,12,38,12,16,14,28,31,26,5,42,36),c(2,2,3),dimnames=list(nemoc=c("nakažen","n
    enakažen"),pohlavi=c("muž","žena"),kontinent=c("Afrika","Amerika","Evropa")))
21. list(popis="výsledek testu",hodnota=c(21,34,25))
22. list(u15= matrix(1,3,4), atd. až po u20)
23. obyv<-data.frame(country=c("Čína","Indie","EU","USA","Indonésie"),population=
    c(1320955000,1169016000,492964961,303004000,231627000),
    percent=c(19.8,17.52,7.39,4.54,3.47))
24. factor(c(1,2,1,1,2,1,2,1,1))
25. factor(c(1,2,1,1,2,1,2,1,1), levels=c(2,1))
26. factor(c(1,2,1,1,2,1,2,1,1), levels=c(2,1), labels=c("muž","žena"))
27. ls()
28. as.array(matrix(1,3,4))
29. použijte str a attributes
30. summary(obyv)
31. rm(obyv)
```

KONSTANTY, OPERÁTORY A MATEMATICKÉ VÝPOČTY

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- naučíte se pracovat se základními logickými a množinovými operacemi;
- zvládnete práci s matematickými operátory;
- seznámíte se s nejčastěji používanými konstantami v R;
- pochopíte principy zaokrouhlování v R.

Klíčová slova: konstanta, operátory, zaokrouhlování.

Průvodce
Narozdíl od předchozího textu budou pro čtenáře následující probírané funkce a operátory asi známější. S podobnými příkazy se setkáte také v programech typu MS Excel atd.

Při práci s jednoduchými objekty můžeme snadno používat různé logické, množinové i matematické operátory a funkce. Ve většině případů (pokud není uvedeno jinak) jsou operace aplikovatelné také na vektory, matice, pole a části tabulek dat. Výstupy pak mohou být opět uloženy do proměnné daného typu (vektor, číslo, logická hodnota).

Logické a množinové operace

- **==, !=** rovno, není rovno
- **<, >, <=, >=** menší než, větší než, menší nebo rovno, větší nebo rovno
- **&, |, !** a zároveň, nebo, negace

```
6==5 #testuje, zda 6 je rovno 5 (nepravda)
[1] FALSE
6!=5 #testuje, zda 6 není rovno 5 (pravda), jinak !6==5
[1] TRUE

5>4|5>8 #5>4 nebo 5>8
[1] TRUE
5>4&5>8 #5>4 a zároveň 5>8
[1] FALSE
!5<3  #negace (není pravda, ze) 5<3
[1] TRUE

x<-c(1,5,4,7,9,12,4,15,7) # vektor x
x>8 #testuje každou hodnotu vektoru a vrací vektor logických hodnot
[1] FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE
x==4 #testuje každou hodnotu vektoru a vrací vektor logických hodnot
[1] FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE FALSE
```

- **all**(relace) – testuje, zda jsou všechny relace pravdivé (r1 & r2 & r3 ...),
- **any**(relace) – testuje, zda je alespoň jedna relace pravdivá (r1 | r2 | r3 ...)
- **which**(relace, arr.ind = FALSE) – testuje, které ze množiny relací jsou pravdivé. Vrací pořadí hodnot splňující podmínku. V případě testování polí (arrays) může vracet pro *arr.ind=T* výpis hodnot v podobě čísel řádků a sloupců, popř. dalších dimenzí.

```
all(c(1,2,3,4,5)<6) #jsou všechny hodnoty <6?
[1] TRUE
```

```
any(c(1,2,3,4,5)>4) # je alespoň 1 hodnota >4?
[1] TRUE
which(c(1,2,3,7,8)>3) #která z hodnot je >3? 4. a 5. hodnota (7,8)
[1] 4 5

my.ar<-array(5:12,c(2,4)) #tvorba pole (2 řádky, 4 sloupce)
my.ar  #výpis pole (pro názornost)
     [,1] [,2] [,3] [,4]
[1,]    5    7    9   11
[2,]    6    8   10   12
which(my.ar>9)#výpis prvků >9 (6.,7. a 8. hodnota - po sloupcích)
[1] 6 7 8
which(my.ar>9,arr.ind=T) #tři hodnoty (2.ř,3.sl; 1ř.,4sl.;2ř.;3.sl)
     row col
[1,]   2   3
[2,]   1   4
[3,]   2   4
```

- **`%in%`** – testuje, zda je první množina prvkem druhé množiny také **`is.element(x,y)`**
- **`intersect`**`(množiny)` – průnik množin (vektorů) uvedených v závorce
- **`union`**`(množiny)` – sjednocení množin (vektorů) uvedených v závorce
- **`setdiff`**`(a,b)` – vrátí vektor čísel neobsažených ve druhé množině,

```
2 %in% c(1,3,5,9) # je číslo 2 obsaženo v uvedeném vektoru?
[1] FALSE
5 %in% c(1,3,5,9) # je číslo 5 obsaženo v uvedeném vektoru?
[1] TRUE
1:3 %in% c(1,3,5,9) # jsou čísla 1,2,3 obsažena v uvedeném vektoru?
[1]  TRUE FALSE  TRUE
intersect(c(2,3,5,7),c(2,4,8,7,9)) #průnik vektorů (oba obsahují 2 a 7)
[1] 2 7
union(c(2,3,5,7),c(2,4,2,7,9)) #sjednocení vektorů
[1] 2 3 5 7 4 9
setdiff(c(2,3,5,7),c(2,4,2,7,9)) #ve druhém vektoru nejsou obsažena čísla 3 a 5
[1] 3 5
```

Kontrolní úkoly

1. Do objektu x1 vložte čísla 5, 4, 3, 8, 11 a 15 do objektu x2 vložte 3, 4, 7, 12 a 10.
2. Testujte, které prvky x1 jsou a) větší nebo rovny 5, b) nejsou rovny 5.
3. Testujte, zda jsou si prvky vektorů x1, x2 rovny.
4. Testujte, které prvky vektoru x2 jsou větší než 5 a zároveň menší než 10.
5. Testujte, zda jsou všechny prvky vektoru x1 a) větší než 1, b) větší než 5.
6. Testujte, zda jsou pravdivé výroky x1>1, x1<30, x1 není rovno 6.
7. Testujte, zda je alespoň 1 prvek vektoru x1 roven číslu 11.
8. Který prvek (jeho pořadí) vektoru x1 a) je větší než 3, b) je větší než 8.
9. Zjistěte, zda prvky 3:6 leží ve vektoru x1.
10. Proveďte průnik a sjednocení vektorů x1 a x2.

Základní aritmetické operátory

- + (sčítání), - (odčítání), * (násobení), / (dělení), ^ (umocňování)
- **`%/%`** – celočíselné dělení (výsledek je celé číslo)
- **`%%`** – zbytek po celočíselném dělení (dělení „modulo“)
- **`sum(..., na.rm = FALSE)`** – sčítání množiny (vektoru, matice, pole atd.) čísel.

- *na.rm* – v případě nastavení TRUE jsou hodnoty NA odstraněny z množiny, jestliže je na.rm nastaveno na FALSE a vektor obsahuje NA hodnoty, pak je výsledkem NA.

- **prod(..., na.rm = FALSE)**- násobení množiny čísel (*na.rm* – shodné jako u sum)

```
3+1569 #klasická operace sčítání
1572
x<-c(1,5,4,7,9,12,4,15,7) # vektor x
3*x #každá položka vektoru je vynásobena 3
[1]  3 15 12 21 27 36 12 45 21
x+x #sečtení položek stejného pořadí (1+1,5+5, atd.)
[1]  2 10  8 14 18 24  8 30 14
c(3,1)*x #jako přechozí, ale 3. hodnota znova *3, 4.hodnota *1, atd.
[1]  3  5 12  7 27 12 12 15 21
Warning message:
longer object length
        is not a multiple of shorter object length in: c(3, 1) * x
#Warning message: pouze varování, že vektory nejsou stejně dlouhé


c(2+6, 3*8, 9-4, 25*(5-3), 2^3, 64/8, 9 %% 4, 9 %/% 4)#různé operace
[1]  8 24  5 50  8  8  1  2
sum(2,3,4,7,9) #součet hodnot v závorce
[1] 25
sum(c(2,3,4,7,9))#totéž
[1] 25
prod(2,3,4)# součin hodnot
[1] 24
```

Konstanty

- **letters, LETTERS, month.abb, month.name, pi, exp(x)** – e^x (Eulerovo číslo)

```
letters #malá písmena abecedy
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l" "m" "n" "o" "p" "q"
[18] "r" "s" "t" "u" "v" "w" "x" "y" "z"letters #malá písmena abecedy
LETTERS #velká písmena abecedy
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "N" "O" "P" "Q"
[18] "R" "S" "T" "U" "V" "W" "X" "Y" "Z"
month.name #názvy měsíců
 [1] "January"   "February"  "March"     "April"     "May"
 [6] "June"      "July"      "August"    "September" "October"
[11] "November"  "December"
month.abb #zkratky názvů měsíců
 [1] "Jan" "Feb" "Mar" "Apr" "May" "Jun" "Jul" "Aug" "Sep" "Oct" "Nov" "Dec"
pi # π Ludolfovo číslo
[1] 3.141593
exp(1) # Eulerovo číslo základ přirozeného logaritmu
[1] 2.718282
```

Matematické funkce

- **abs**(…) – absolutní hodnota
- **exp**(…) – mocnina základu přirozeného logaritmu (e) neboli Eulerova čísla, e = `exp(1)`
- **logb**(…, `base`) – logaritmus čísla (vektoru) při základu *base*.
- **log**(…) – přirozený logaritmus (ekvivalent s `logb(…, base = exp(1))`).
- **log10**(…) – dekadický logaritmus,

- **sqrt**(…) – druhá odmocnina (jiné odmocniny pomocí mocnin např. $\sqrt[3]{x}$ = x^(1/3)).
- **sign**(…) – hodnoty -1, 0, 1 podle toho, jestli je hodnota záporná, nulová nebo kladná),
- **sin**(…), **cos**(…), **tan**(…), **sinh**(…), **cosh** (…), **tanh**(…) – trigonometrické a hyperbolické funkce
- **asin**(…), **asinh**(…), **acos**(…), **acosh**(…), **atan**(…), **atanh**(…) – inverzní trigonometrické a hyperbolické funkce
- **factorial**(…) – faktoriál, 5! = 5.4.3.2.1, factorial (x) shodné s prod(1:x)
- **choose**(n, k) – počet kombinací bez opakování (*n* nad *k*), shodné s viz příklad

```
x<-10; y<-3; factorial(x)/(factorial(y)*factorial(x-y))
[1] 120
choose(10,3)
[1] 120
```

- **sample(x, size, replace=FALSE, prob=NULL)** – vytvoří náhodnou kombinaci (permutaci) čísel z vektoru *x* o velikosti *size* s opakováním (*replace*=T) nebo bez opakování (*replace*=F). Argument *prob* umožňuje nastavení pravděpodobnosti výběru jednotlivých prvků.

```
sample(1:10,size=5,replace=T) #náhod. komb. 5 hodnot z čísel 1 – 10 s opak.
[1] 3 7 1 7 1
sample(1:10,size=10,replace=T) #náhod. komb. 10 hodnot z čísel 1 – 10 s opak.
 [1]  9  6  9  4  1  1  5  4 10  8
sample(1:10,size=10,replace=F) #náhodná permutace 10 čísel (bez opak)
 [1]  1  9 10  5  2  8  3  4  6  7

#náhodná kombinace čísel 1-3 s pravděpodob. 0.5 pro 1, 0.3 pro 2 a 0.2 pro 3
sample(1:3,size=20,rep=T,prob=c(0.5,0.3,0.2))
 [1] 1 1 1 1 1 1 3 1 1 2 2 1 1 2 3 1 3 2 2 1
```

Kontrolní úkoly

11. Zkontrolujte, zda máte načteny vektory z úlohy č. 1 této kapitoly. K vektoru x1 přičtěte číslo 5, vektor x2 umocněte na čtvrtou.
12. K vektoru x1 přičtěte vektor c(3,4). Vysvětlete, co je výsledkem
13. Vypočítejte přirozený logaritmus x2 a následně logaritmus se základem 5
14. Vypočítejte odmocninu z čísla 121 a následně třetí odmocninu z 27
15. Proveďte losování tahu sportky, tzn. vyberte 7 čísel od 1 do 49 tak, aby se neopakovala
16. Simulujte 100 hodů mincí, kdy vám může padnout hodnota "panna"nebo "orel".

Zaokrouhlování

- **round**(x, digits=0) – klasické zaokrouhlení objektu x na určitý počet desetinných míst (*digits*) (2.5 na 2, 3.5 na 4), typ zaokrouhlení (jak zaokrouhlit 5) závisí na operačním systému. U jednodušších čísel zachovává systém „k nejbližšímu sudému“, pokud za číslicí 5 následuje nula (ne systém od 5 nahoru, který je obecně rozšířený a učí se ve školách).
- **trunc**(x) – vrací celou část daného čísla
- **floor**(x), **ceiling(x)** – zaokrouhlení na nižší (vyšší) celé číslo
- **signif**(x, digits) – zaokrouhlí číslo na určitý počet platných cifer (*digits*), zbytek doplní jako 0.

```
round(c(0.5,1.5,2.5,5.5,6.5,7.5,8.5), digits=0)#zaokrouhlí na 0 deset. míst
```

```
 [1] 0 2 2 6 6 8 8
round(c(12.5545, 2.5545), digits=3) #zaokrouhlí na 3 des. místa
[1] 12.555  2.554
trunc(c(0.5,1.5,2.5,5.5,6.5,7.5,8.5)) #uřízne celou část čísla
[1] 0 1 2 5 6 7 8
ceiling(c(2.7,-2.3, -1.8))#zaokrouhlí nahoru
[1]  3 -2 -1
floor(c(2.7,-2.3, -1.8)) #zaokrouhlí dolů
[1]  2 -3 -2
signif(c(12.5545, 2.5545), digits=3) # tři cifry od začátku pro všechna čísla
[1] 12.60  2.55
signif(c(1255, 255,12.3), digits=3)
[1] 1260.0  255.0   12.3
```

Generování pseudonáhodných čísel v R

Procvičení mnohých funkcí je nutné provést na více příkladech. Abychom se nemuseli omezovat pouze na příklady uvedené ve skriptu, vysvětlíme si alespoň ve zkratce tvorbu pseudonáhodných čísel ještě před vlastním vysvětlením práce s rozděleními náhodných veličin (viz další části skripta). Náhodná čísla generovaná v R (a obecně ve všech softwarech) nejsou ve skutečnosti zcela náhodná, ale jsou generována na základě specifických algoritmů tak, aby se náhodným číslům podobala (tzv. pseudonáhodná čísla) a mohli jsme s nimi pracovat jako s náhodnými čísly. Pro naše účely je takový typ náhody dostačující. Podle pravděpodobnosti výskytu jednotlivých hodnot můžeme generovat čísla z různých typů rozdělení. Pro základní účely postačí následující tři:

- **runif**(n,min=0,max=1) – vypíše *n* reálných náhodných čísel v intervalu *(min,max)*. Interval je otevřený, tzn. krajních hodnot intervalu není nikdy dosaženo. Jedná se o uniformní distribuci, tzn. každé číslo má stejnou pravděpodobnost výskytu.

```
runif(10) #generování 10 náhodných čísel od 0 do 1
 [1] 0.81726314 0.55485435 0.88582695 0.32683955 0.02863844 0.06011710
 [7] 0.44680456 0.32662559 0.94091232 0.72545513
runif(10,min=5,max=8) #generování 10 čísel od 5 do 8
 [1] 5.132661 7.830364 5.113809 6.908226 5.802121 5.370527 6.709676 7.141092
 [9] 6.540720 5.114725
trunc(runif(10,min=5,max=8))#generování 10 celých čísel od 5 do 7
 [1] 5 7 7 6 6 5 6 5 7 5
```

- **rpois**(n,lambda) – vypíše n celých náhodných čísel v intervalu s průměrem *lambda*. Jedná se o tzv. Poissonovu distribuci, tzn. čísla kolem průměru se budou objevovat častěji než dál od průměru.

```
rpois(30,lambda=3) #generování 30 celých čísel s průměrem kolem 3
 [1] 6 3 3 2 3 2 1 2 2 1 2 6 2 2 3 4 1 1 4 5 3 6 4 4 0 2 4 2 2 4searchpaths() #
```

- **rnorm**(n,mean=0,sd=1) – vypíše n reálných náhodných čísel v intervalu s průměrem *mean*. a směrodatnou odchylkou *sd*. Jedná se o tzv. normální rozdělení, kdy nejčastější výskyt budou vykazovat čísla kolem průměru.

```
rnorm(20) #generování 20 reálných čísel s průměrem kolem 0 a směr. odch. 1
 [1]  2.344463021 -0.138127698 -1.009893425  1.187891442  0.904608210
 [6] -0.173483318 -0.281908102 -0.958777797 -0.372863274 -0.723849980
[11] -1.175724988  1.227925261 -1.780056890  0.123418655 -0.197777911
[16]  0.040842241  0.167808499  0.075059917  1.018527431  0.004379758
rnorm(20,mean=10,sd=2) #totéž s průměrem kolem 10 a směr. odch. 2
 [1] 12.338503 13.893136  7.162175 12.410630 10.599693  9.627598 10.269063
 [8] 14.114875 11.453652  9.774820  9.110875 10.712443 11.750055  9.198243
[15] 10.215472  9.170082  8.465258 11.461669 11.039822 13.569389
```

Kontrolní úkoly

17. Vytvořte vektor m.norm sestavený z 20 čísel z normálního rozdělení s průměrem 5 a směrodatnou odchylkou 2.
18. Vytvořte vektor m.zaok tak, že m.norm zaokrouhlíte na celá čísla.
19. Vytvořte vektor m.cel tak, že odstraníte z m.norm čísla za desetinnou čárkou.
20. Porovnejte pomocí logických operátorů, které hodnoty m.zaok a m.cel se sobě rovnají (pořadí čísel). Výsledek zkontrolujte tak, že si vektory vypište pod sebe.
21. Zaokrouhlete m.norm na na 2 desetinná čísla.
22. Generujte 25 čísel z Poissonova rozdělení s průměrem 4.
23. Generujte čísla od 0 do 10 z rovnoměrného rozdělení do objektu vek1, pak je zaokrouhlete na celá čísla. Jaký bude rozdíl v rozsahu čísel, když použijete funkci trunc.

Shrnutí:

Základní aritmetické operace lze v R provádět nejen s čísly, ale také s jednoduchými objekty typu vektor, matice atd. Kromě aritmetických operací je zde také řada vestavěných logických a množinových operátorů a komplikovanějších matematických funkcí. Při procvičování práce s některými funkcemi je užitečné vědět, jak vytvářet vektory náhodných čísel. R nabízí mnoho různých typů náhodných veličin, z nichž nejčastěji budeme využívat náhodná čísla z pravidelného, Poissonova a normálního rozdělení.

Metody k zapamatování:

- logické operátory: ==, != , <, >, <=, >= , &, |, !, all, any, which
- množinové operátory: %in%, intersect, union, setdiff
- aritmetické operátory: +, -, *, /, ^, %/%, %%, sum, prod
- konstanty: letters, LETTERS, month.abb, month.name, pi, exp(x)
- matematické funkce: abs, exp, logb, log, log10, sqrt, sign,, sin, cos, tan, sinh, cosh, tanh, asin, asinh, acos, acosh, atan, atanh, factorial, choose, sample
- zaokrouhlování: round, trunc, floor, ceiling, signif
- generování pseudonáhodných čísel: runif, rpois, rnorm

Výsledky

```
1.  x1<- c(5,4,3,8,11,15);x2<- c(3,4,7,12,10)
2.  x1>=5; x1!=5
3.  x1==x2 #vrátí TRUE jen u 4, protože je u obou vektorů na 2. místě
4.  x2>5 & x2<10
5.  all(x1>1); all(x1>1)
6.  all(x1>1,x1<30,x1!=6)
7.  any(x1==11)
8.  which(x1>3); which(x1>8)
9.  3:6 %in% x1
10. intersect(x1,x2); union(x1,x2)
11. x1+5; x2^4
12. x1+c(3,4) #střídavě se bude přičítat 3 a 4
13. log(x2); logb(x2,5)
14. sqrt(121); 27^(1/3)
15. sample(1:49,7,rep=F)
16. sample(c("panna","orel"),100,rep=T)
17. m.norm<-rnorm(20,5,2)
18. m.zaok<-round(m.norm)
19. m.cel<-trunc(m.norm)
20. which(m.zaok==m.cel)
21. round(m.norm,2)
22. rpois(25,4)
23. vek1<-runif(30,0,10); round(vek1) #rozsah 0-10, při použití fce trunc bude rozsah 0-9
```

MANIPULACE S OBJEKTY

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- naučíte se načíst data z jiných aplikací;
- jednoduše provádět úpravy v jednotlivých objektech;
- seznámíte se s funkcemi umožňujícími složitější manipulaci s objekty;
- pochopíte princip funkcí pro hromadné operace na složitých objektech.

Klíčová slova: import objektů, editace objektů, hromadné operace na objektech.

Průvodce
Všimněte si, že jsme až dosud nepoužili pro naši práci žádná složitější data. Je to tím, že data pro účely statistického zpracování jsou často dost rozsáhlá a zadávání uvedeným způsobem by bylo ohromně zdlouhavé. To, jak data "dostat do R" a jak s nimi pak dále pracovat se dozvíte v následující kapitole. Ještě drobnou poznámku k manipulaci s daty. Z vlastní zkušenosti doporučuji data vždy ukládat do databáze (např. v MS Excel), následně odfiltrovat chyby a až finálně upravená data načíst do R. Pokud totiž neprovádíme specifické automatické hromadné operace jsou pro úpravu dat mnohem vhodnější "user friendly" programy.

Načítání objektů a jejich editace

Pokud nechceme používat cvičná data zabudovaná v R, je nutné datové objekty vytvořit nebo načíst. Základním problémem je velice často načtení objektu. Načítání probíhá několika způsoby a to buď ze souboru nebo ze schránky. Při načítání ze souboru musí být uvedena celá cesta k souboru. Stávající nastavení složky, ze které jsou načítány soubory získáme pomocí funkce `getwd()`nebo pomocí položky ve *File – Change dir*. Při načítání ze schránky musí být data zkopírována z jakéhokoliv programu (např. MS Word, MS Excel, ...) pomocí CTRL C (nebo v menu *Úpravy – Kopírovat*) do schránky a následně pomocí příkazu `scan` nebo `read.table` načtena.

- **`scan`**`(file, what, nmax, n, sep, quote, dec, skip, nlines, na.strings, flush = F, fill = F, strip.white = F, quiet = F, blank.lines.skip = T, multi.line = T, comment.char)` – načítá vektor ze souboru `file` (musí být uvedena plná cesta s názvem souboru ve formě character nebo načítá ze stávajícího adresáře viz úvod této kapitoly), popř. ze schránky `file = "clipboard"`. Do file je možno také zadat typ objektu connection.
  - `what` – typ načítané hodnoty (*`logical()`*, *`integer()`*, *`numeric()`*, *`complex()`*, *`character()`*, *`raw()`*)
  - `nmax` popř. `n` – maximální počet načtených položek.
  - `sep` – specifikuje znak, kterým jsou odděleny jednotlivé položky, např. `sep=","`, `sep=";"`, `sep="\t"` je tabelátor, `sep="\n"` je konec řádku.
  - `quote` – znak pro uvozovky
  - `dec` – označuje znak pro desetinnou čárku
  - `skip` – počet řádků, které se mají v souboru vynechat než se začnou načítat data
  - `nlines` – maximální počet řádků, který má být načten

- `na.strings` – vektor hodnot, které mají být chápány jako NA hodnoty
- `quiet` – logická hodnota, nebude se vypisovat řádek s počtem načtených položek
- `allowEscape` – při načítání budou brány v úvahu specifické znaky (\n, \t)
- další viz help(scan)

```
# Princip načtení ze schránky do objektu s názvem x
# 1. Zkopírujte následující řádek do konzoly (nespouštět Enter)
     x<-scan("clipboard")
# 2. Zkopírujte řádek s čísly pomocí CTRL C (provádějte i v násl. příkladech)
#    (data se uloží do tzv. schránky "clipboard")
     5 4 6 9 1 24 36 4 7 9 19 4
# 3. Spusťte připravený řádek x<-scan ...
Read 12 items
x #vypíše načtený vektor
[1]  5  4  6  9  1 24 36  4  7  9 19  4

# Načtení tohoto řetězce ze schránky do R (neuloženo do objektu)
scan("clipboard",what=character())
Read 11 items
 [1] "#"          "Načtení"    "tohoto"     "řetězce"    "ze"
 [6] "schránky"   "do"         "R"          "(neuloženo" "do"
[11] "objektu)"

# Načtení dat oddělených čárkou (sep) bez vypsání počtu položek
5, 4, 6, 9, 1, 24, 36, 4, 7, 9, 19, 4
scan("clipboard",sep=",",quiet=T)
 [1]  5  4  6  9  1 24 36  4  7  9 19  4

# Načtení následujících řetězců ze schránky do R (neuloženo do objektu)
"ahoj"; "čau";"ahoj";"čau";"čau";"zdravím";"zdravím";"zdravím";"ahoj"
scan("clipboard",what=character(),sep=";")
Read 9 items
[1] "ahoj"    " čau"    "ahoj"    "čau"     "čau"     "zdravím" "zdravím"
[8] "zdravím" "ahoj"

# Načtení čísel s desetinnou tečkou
1.2 1.8 1.5 1.4 2.8 3.9
scan("clipboard")
Read 6 items
[1] 1.2 1.8 1.5 1.4 2.8 3.9

# Načtení čísel s desetinnou čárkou
1,2 1,8 1,5 1,4 2,8 3,9
scan("clipboard", dec=",")
Read 6 items
[1] 1.2 1.8 1.5 1.4 2.8 3.9
```

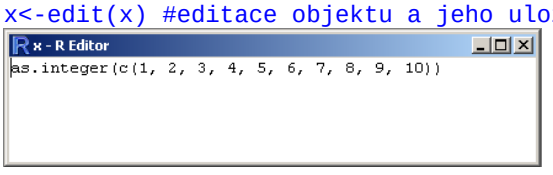
- **`read.table`**`(file, header = FALSE, sep = "", quote = "\"'", dec = ".", row.names, col.names, as.is = !stringsAsFactors, na.strings = "NA", colClasses = NA, nrows = -1, skip = 0, check.names = TRUE, fill = !blank.lines.skip, strip.white = FALSE, blank.lines.skip = TRUE, comment.char = "#", allowEscapes = FALSE, flush = FALSE, stringsAsFactors = default.stringsAsFactors())` – podobné funkci scan, ale načítá soubor do tabulky dat – data.frame(např. z MS Excel, ale také z MS Word). Ideální pro import vlastních tabulek. Načítání ze schránky opět pomocí `file = "clipboard"`.
  - `header` – logická hodnota, zda je v datech obsažena hlavička sloupců.
  - při načítání v MS Excel i jiných programech mohou zejména vzniknout chyby a) jestliže používáme v hlavičce mezery nebo nestandardní znaky (včetně háčků a čárek), b) jestliže importujeme v Excelu prázdné buňky a nezadáme jako sep="\t")

```
#tabulka dat, která bude načtena
2 3 42 7
8 9 7 51
```

```
1 21 5 4

read.table("clipboard") #načtení jednoduchých dat bez hlavičky
  V1 V2 V3 V4
1  2  3 42  7
2  8  9  7 51
3  1 21  5  4

#tabulka dat, která bude načtena, data oddělena čárkou
2,3,4,7
8,9,7,5
1,2,5,4

read.table("clipboard") #chybné načtení (data načtena jako character)
       V1
1 2,3,4,7
2 8,9,7,5
3 1,2,5,4

read.table("clipboard",sep=",")#správné načtení
  V1 V2 V3 V4
1  2  3  4  7
2  8  9  7  5
3  1  2  5  4

#tabulka dat s hlavičkou
m1,m2,m3,m4
8,9,7,5
1,2,5,4

read.table("clipboard",header=T,sep=",")
  m1 m2 m3 m4
1  8  9  7  5
2  1  2  5  4
#tabulka dat s hlavičkou (ale ne v prvním sloupci, takže R automaticky odliší)
   Strava  pH
1     mas 7.5
2     mas 7.1
3     mas 8.2
4     mas 8.0
5     mas 7.9
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2
mytab<-read.table("clipboard")
mytab #vypíše načtenou tabulku (totožné se vstupním formátem)
```

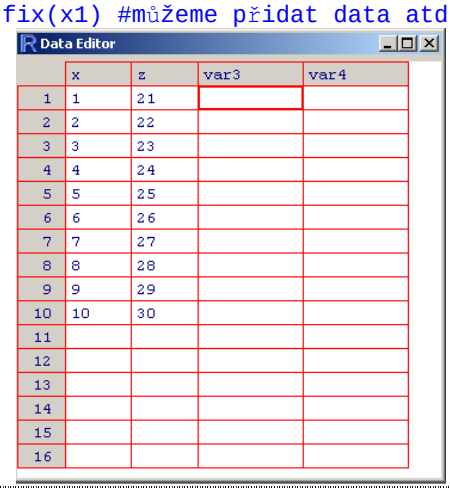
- **edit**(name = NULL, file = "", title = NULL,editor = getOption ("editor"), …)) – otevře objekt *name* v editoru dat. V editoru lze s daty dále pracovat a následně jsou uložena do file nebo vytištěna na obrazovku. Abychom úpravy dat uložili do objektu, pak je nutné přiřadit editovaným datům název (např. x<-edit(x)).
- **fix**(name = NULL, file = "", title = NULL,editor = getOption("editor"), …)) – otevře editor dat, ve kterém lze data upravovat

```
x<-1:10
x<-edit(x) #editace objektu a jeho uložení do vektoru x, alternativa fix(x)
```




```
x1<-data.frame(x=1:10,z=21:30)
```

```
fix(x1) #můžeme přidat data atd., alternativa je x1<-edit(x1)
```



	x	z	var3	var4
1	1	21		
2	2	22		
3	3	23		
4	4	24		
5	5	25		
6	6	26		
7	7	27		
8	8	28		
9	9	29		
10	10	30		
11				
12				
13				
14				
15				
16				

Kontrolní úkoly

1. Načtěte následující vektory jako číselné vektory.
   2 8 4 25 36 78 1 4 89
   15, 25, 14, 17, 2, -1, 7, 24, 36, 47, 55
   2.5, 3, 4, 1.8, 2.7, -0.7
   1,27; 2,36; 4,25; 8,26
   1, , 2,5,,,7,10
2. Načtěte následující vektor jako logický (TRUE, FALSE): T,F,T,F,F,F,T,T,T,F
3. Načtěte znaky ve sloupci do vektoru m.vec:
   c
   c
   d
   e
   f
4. Přidejte pomocí funkce fix další znaky do vektoru m.vec.
5. Načtěte tabulku dat do objektu m.tab:

```
      druh ks
1  Carabus 10
2  Carabus  8
3     Abax 22
4  Carabus 11
5  Carabus 12
6  Carabus  5
7  Carabus 17
8  Carabus 19
9     Abax 13
10    Abax 17
```

6. Pomocí funkce fix upravte objekt m.tab – přidejte jeden sloupec s libovolnými hodnotami a další s logickými hodnotami. Zkontrolujte, zda jsou hodnoty ve sloupcích správného typu.
7. Vytvořte tabulky v Excelu s hlavičkou i bez hlavičky s různými typy dat. Pokuste se také načíst data, ve kterých chybí údaje.

Základní práce s jednotlivými prvky objektů

- **`[`**`x, drop=T`**`]`**. Hranaté závorky za vektorem s číslem nebo logickou hodnotou (x) nám dovoluje filtrovat z vektoru určité prvky, popř. získávat jen části polí, matic nebo tabulek. získáme část z daného vektoru (tabulky). Argument ***drop*** – zjednodušuje objekt (např. u jednořádkové matice vytvoří vektor, u pole může vytvořit pole s nižší dimenzí).

```
##################### indexování vektorů ##############################

kc<-c(5,2,2,2,1,6,1,7,4,5,0,1,6,2,4,4,2,7,4,2,2,3,3,3,1,5,3)
kc[3] # výpis 3. prvku vektoru
[1] 2
kc[3:7] # výpis 3.-7. prvku vektoru
[1] 2 2 1 6 1
kc[c(1,5,7,2,1,1)] # výpis prvků vektoru(1. prvek se opakuje 3×)
[1] 5 1 1 2 5 5
kc[-(1:10)] # vynechán 1.-10. prvek
[1] 0 1 6 2 4 4 2 7 4 2 2 3 3 3 1 5 3
kc[kc==2] #vypíše prvky rovnající se 2
[1] 2 2 2 2 2 2 2
kc[kc>5] #vypíše prvky větší než 5
[1] 6 7 6 7

#výpis těch prvků, které jsou nastaveny na TRUE
kc[c(T,F,T,F,T,F,T,F,T,F,T,F,T,T,T,T,T,T,T,T,F,F,F,T,F,F)]
 [1] 5 2 1 1 4 0 6 2 4 4 2 7 4 2 2 1
# počet logických prvků neodpovídá počtu prvků vektoru, schéma se opakuje
# tzn. vynechána každá 3. hodnota
kc[c(T,T,F)]
 [1] 5 2 2 1 1 7 5 0 6 2 4 2 4 2 3 3 1 5

obdobi<-c("jaro","leto","podzim","zima") #vytvořen vektor prvků období
obdobi[1] # první položka vektoru období
[1] "jaro"
obdobi[2:4] #2.-4. položka
[1] "leto"   "podzim" "zima"
obdobi[-3]
[1] "jaro" "leto" "zima"
```

```
##################### indexování matic ##############################
mymat<-matrix(1:6,2,3,TRUE); mymat #vytvoří matici a vypíše ji
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
mymat[2,1] #prvek matice mymat ve 2. řádku a 1. sloupci
[1] 4
mymat[2,3] #prvek ve 4.řádku a 3. sloupci
[1] 6
mymat[2,] # prvky druhého řádku
[1] 4 5 6
mymat[1,,drop=F] # totéž, ale výstup je matice
     [,1] [,2] [,3]
[1,]    1    2    3
mymat[c(1,2),c(2,3)] #vypíše hodnoty [1,2],[1,3],[2,2],[2,3]
     [,1] [,2]
[1,]    2    3
[2,]    5    6
mymat[,c(T,F,T)]#výpis sloupce 1 a 3 (označný jako TRUE)
     [,1] [,2]
[1,]    1    3
[2,]    4    6
```

```
##################### indexování seznamů ##############################
x<-list(let=letters[1:10],num=1:5) #seznam
x #výpis seznamu
```

```
$let
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"

$num
[1] 1 2 3 4 5

x[2] #výpis druhého objektu seznamu (včetně názvu položky)
$num
[1] 1 2 3 4 5
x[c(2,1)] #výpis druhého objektu a následně prvního objektu seznamu
$num
[1] 1 2 3 4 5

$let
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
x[[1]] #výpis prvního objektu seznamu (bez názvu položky)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
x[[1]][3] #výpis prvního objektu třetího prvku
[1] "c"
x[[2]][3:6] #výpis 2. objektu 3.-6. prvku (6. neexistuje, takže NA)
[1]  3  4  5 NA
x[[c(1,3)]] #výpis 1. objektu 3. prvku
[1] "c"

x["let"]#výpis prvního objektu seznamu pomocí jména (s názvem tag, tzn. list)
$let
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
x[["let"]]#výpis prvního objektu seznamu pomocí jména (vektor)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"

################################################################################
#indexování tabulek dat

mytab<-
data.frame(Strava=rep(c("mas","mix"),each=5),pH=c(7.5,7.1,8.2,8.0,7.9,7.6,7.5,7.2
,7.9,8.2))
   Strava  pH
1     mas 7.5
2     mas 7.1
3     mas 8.2
4     mas 8.0
5     mas 7.9
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2

mytab[,"pH"]#výpis sloupce Strava také mytab[,2]
 [1] 7.5 7.1 8.2 8.0 7.9 7.6 7.5 7.2 7.9 8.2

mytab[,"Strava"]#výpis sloupce Strava také mytab[,1]
 [1] mas mas mas mas mas mix mix mix mix mix
Levels: mas mix

mytab[c(2,5,7),"Strava"]#výpis řádků 2,5,7 ve sloupci Strava
[1] mas mas mix
Levels: mas mix

mytab[2:4,"Strava"]#řádků 2,3,4, ve sloupci Strava
[1] mas mas mas
Levels: mas mix

mytab[2:4,] #řádků 2,3,4
  Strava  pH
2    mas 7.1
3    mas 8.2
4    mas 8.0
```

```
mytab[2,1] #výpis 2. položky sloupce 1
[1] mas
Levels: mas mix

mytab[c(T,F),2]#výpis lichých položek sloupce 2
[1] 7.5 8.2 7.9 7.5 7.9

mytab[mytab$Strava=="mix",]#výpis tabulky pro hodnoty mix
   Strava  pH
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2
mytab[mytab$pH<8,1]#výpis polože sloupce Strava pro hodnoty pH<8
[1] mas mas mas mix mix mix mix
Levels: mas mix
```

- **$** – název$tag – výpis sloupců tabulky nebo položek seznamu

```
mytab$Strava #výpis sloupce Strava předchozí tabulky
[1] mas mas mas mas mas mix mix mix mix mix
mytab$pH[3:4] #výpis řádků 3 a 4 ve sloupci Strava předchozí tabulky
[1] 8.2 8.0
x$let #výpis vektoru let z předchozího seznamu
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
```

- **append**(x, values, after=length(x)) – přidá hodnoty *values* k vektoru (tabulce) *x* za *after* hodnotu

```
x<-c(1,1,2,4,5,7);x
[1] 1 1 2 4 5 7
append(x,c(2,3,5,5)) #připojí k x další vektor
[1] 1 1 2 4 5 7 2 3 5 5
```

- **cbind**(x1, x2, ...), **rbind**(x1, x2, ...) – sloučí vektory *x1, x2* ... (popř. tabulky, matice) po řádcích nebo po sloupcích

```
cbind(c(1,1,2,4),c(5,6,7,8))#sloučí dva vektory (jako sloupce) do matice
     [,1] [,2]
[1,]    1    5
[2,]    1    6
[3,]    2    7
[4,]    4    8
cbind(x1=c(1,1,2,4),x2=c(5,6,7,8)) #totéž, pouze navíc názvy sloupců
     x1 x2
[1,]  1  5
[2,]  1  6
[3,]  2  7
[4,]  4  8

cbind(1:3) #z vektoru vytvoří matici
     [,1]
[1,]    1
[2,]    2
[3,]    3

rbind(c(1,1,2,4),c(5,6,7,8))#sloučí dva vektory do dvou řádků
     [,1] [,2] [,3] [,4]
[1,]    1    1    2    4
[2,]    5    6    7    8

rbind(m=1:4, c=2, "a.1" = 10, c(1,2)) #sloučí dva vektory, přidá názvy řádků
    [,1] [,2] [,3] [,4]
m      1    2    3    4
c      2    2    2    2
a.1   10   10   10   10
       1    2    1    2
```

- **`length`**`(x)` – velikost objektu `x` (délka vektoru, počet položek seznamu, sloupců tabulky dat, prvků v matici)

```
length(1:121)
[1] 121
length(5:121)
[1] 117
length(letters)
[1] 26
length(c(1,2,8))
[1] 3
length(list(a=4:7,b=1:10,d=1:13))
[1] 3
x.1<-c(2,2,4,5,6,8,1,2,3,4)
length(x.1[x.1==2])
[1] 3
```

- **`rank`**`(x, na.last=TRUE, ties.method= c("average", "first", "random", "max", "min"))` – vypíše pořadí jednotlivých hodnot v dané řadě (vektoru) `x`.
  - *`na.last`* – logical, hodnoty NA umístěny na konec
  - *`ties. method`* – jaké přiřadit pořadové číslo shodných hodnot

```
# 5 způsobů řazení vektoru (rozdíl u stejných hodnot)
rank(c(5,5,5,4,3,7,9,9,1),ties.method="average")#průměrná hodnota
[1] 5.0 5.0 5.0 3.0 2.0 7.0 8.5 8.5 1.0
rank(c(5,5,5,4,3,7,9,9,1),ties.method="min") #minimum z pořadí
[1] 4 4 4 3 2 7 8 8 1
rank(c(5,5,5,4,3,7,9,9,1),ties.method="max") #maximum z pořadí
[1] 6 6 6 3 2 7 9 9 1
rank(c(5,5,5,4,3,7,9,9,1),ties.method="first") #podle pozice v řadě
[1] 4 5 6 3 2 7 8 9 1
rank(c(5,5,5,4,3,7,9,9,1),ties.method="random") #náhodně
[1] 5 4 6 3 2 7 9 8 1
```

- **`rev`**`(x)` – převrátí pořadí hodnot vektoru

```
rev(1:10)#vektor se vypíše opačně
 [1] 10  9  8  7  6  5  4  3  2  1
rev(c(2,4,3)) #vektor se vypíše opačně
[1] 3 4 2
```

- **`sort`**`(x, partial=NULL, na.last=NA,decreasing = F, method=c("shell", "quick"), index.return=F)` – seřadí hodnoty vzestupně (v případě *`decreasing=T`* sestupně),
  - *`index.return`* – logical, vypsat také původní pořadí hodnot?
  - *`partial`* – vektor hodnot, které mají být umístěna přesně, nižší a vyšší hodnoty jsou umístěny libovolně
  - *`na.last`* – logical, jak řadit hodnoty NA, pro `NA` jsou vynechány, `T` poslední místo
  - *`decreasig`* – logická hodnota, řadit sestupně?

```
sort(c(100,102,95,21,300))#seřadí vektor
[1]  21  95 100 102 300
sort(c(5,5,5,4,3,7,9,9,1),index.return=T) #seřadí + ix původní pořadí čísel
$x
[1] 1 3 4 5 5 5 7 9 9

$ix
[1] 9 5 4 1 2 3 6 7 8

sort(c(5,5,5,4,3,7,9,9,1,8),partial=c(4))#přesně umístí jen 4, zbytek libovolně
 [1] 1 3 4 5 5 7 9 9 5 8
sort(c(5,5,5,4,3,7,9,9,1,8),partial=c(7,9)) #přesně umístí jen 7 a 9
 [1] 1 5 3 4 5 5 7 8 9 9
```

- **order**(x, na.last=T, decreasing=F) – vypíše indexy seřazených hodnot, v případě více vektorů spojených seřadí nejprve podle prvního a v případě opakovaných hodnot použije druhý vektor atd.
  - *na.last* – logical, NA řadit na poslední místo?
  - *decreasig* – logical, řadit sestupně?

```
order(c(9:2)) # vypíše pořadí hodnot
[1] 8 7 6 5 4 3 2 1

mytab<-
data.frame(Strava=rep(c("mas","mix"),each=5),pH=c(7.5,7.1,8.2,8.0,7.9,7.6,7.5,7.2
,7.9,8.2))
mytab #tabulka dat, která má být seřazena
   Strava  pH
1     mas 7.5
2     mas 7.1
3     mas 8.2
4     mas 8.0
5     mas 7.9
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2
my.sort<-order(mytab[,2],mytab[,1])#řazení podle sloupce 2 a pak podle 1
my.sort #vypíše původní indexy hodnot v pořadí, v jakém jsou seřazeny
 [1]  2  8  1  7  6  5  9  4  3 10
mytab[my.sort,]#výpis nově seřazené tabulky
   Strava  pH
2     mas 7.1
8     mix 7.2
1     mas 7.5
7     mix 7.5
6     mix 7.6
5     mas 7.9
9     mix 7.9
4     mas 8.0
3     mas 8.2
10    mix 8.2
```

- **replace**(x, list, values): nahradí hodnoty na určitém místě (list) ve vektoru x hodnotami values

```
replace(c(1,3,4,2,1,7),list=2,values=12)#nahradí 2. položku číslem 12
[1]  1 12  4  2  1  7
replace(c(1,3,4,2,1,7), list=c(2,4), values=10) # 2. a 4. položka bude 10
[1]  1 10  4 10  1  7
replace(c(1,3,4,2,1,7),c(2,4),c(10,11))#2. položka 10 a 4. položka 11
[1]  1 10  4 11  1  7
```

- **duplicated**(x, incomparables=F,…) – vyhodnotí, zda se hodnota příslušné položky opakuje nebo je poprvé, incomparables – hodnoty které nebudou porovnány, pro pole je dále možné přidat argument MARGIN.

```
duplicated(c(1,1,2,2))#jestliže je číslo poprvé, pak F, při opakování je T
[1] FALSE  TRUE FALSE  TRUE
duplicated(c(10,2,1,1,2,2,6,7))
[1] FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE

x<-data.frame(sl1=c(1,1,1,1,2),sl2=c(3,3,2,1,4))#tabulka dat
x
  sl1 sl2
1   1   3
2   1   3
3   1   2
4   1   1
5   2   4
```

```
duplicated(x) #označuje duplikáty řádků
[1] FALSE  TRUE FALSE FALSE FALSE
```

- **unique**(x, incomparables=F,…) – vypíše objekt x (vektor, tabulka dat) s vynechanými duplikáty

```
unique(c(1,1,3,3,4))#zobrazí všechny hodnoty bez duplikátů
[1] 1 3 4
kc<-c(5,2,2,2,1,6,1,7,4,5,0,1,6,2,4,4,2,7,4,2,2,3,3,3,1,5,3)
unique(kc)
[1] 5 2 1 6 7 4 0 3
x<-data.frame(sl1=c(1,1,1,1,2),sl2=c(3,3,2,1,4))# tabulka jako u duplicated
unique(x) #zobrazí všechny řádky bez duplikátů
  sl1 sl2
1   1   3
3   1   2
4   1   1
5   2   4
```

- **ncol**(x), **nrow**(x) – počet sloupců, řádků tabulky dat, matice nebo pole.

```
x<-data.frame(sl1=c(1,1,1,1,2),sl2=c(3,3,2,1,4))# tabulka jako u duplicated
ncol(x) #počet sloupců tabulky dat
[1] 2
nrow(x) #počet řádků tabulky dat
[1] 5
nrow(2:12) #nevyhodnotí pro vektor
NULL
NROW(2:12) #počítá také pro pro vektor
[1] 11
NCOL(2:12)
[1] 1
```

- **dim**(x) – vypíše počet dimenzí jednotlivých objektů (polí, matic, tabulek, vektorů).

```
x<-matrix(1:10,2,5); x #matice 2 řádky, 5 sloupců
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    3    5    7    9
[2,]    2    4    6    8   10
dim(x) #výpis dimenzi objektu
[1] 2 5

mv<-1:10
dim(mv) #objekt mv je vektor a nemá dimenzionální strukturu
NULL

x<-data.frame(sl1=c(1,1,1,1,2),sl2=c(3,3,2,1,4))# tabulka jako u duplicated
dim(x)
[1] 5 2
```

- **dimnames**(x) – vypisuje nebo nastavuje názvy jednotlivých dimenzí objektu *x*.

```
x<-data.frame(sl1=c(1,1,1,1,2),sl2=c(3,3,2,1,4))# tabulka jako u duplicated
dimnames(x)
[[1]]
[1] "1" "2" "3" "4" "5"

[[2]]
[1] "sl1" "sl2"
```

- **subset**(x, subset, select, drop=F, …) – vybere z vektoru, matice, pole nebo tabulky dat x pouze určité položky (řádky nebo prvky)
  - *subset* – podmínka, kterou musí splňovat vybrané řádky (u vektoru prvky)
  - *select* – čísla sloupců nebo podmínka, kterou musí splňovat sloupce
  - *drop* – v případě, že lze objekt zjednodušit, pak provést zjednodušení?

```
x<-1:20;x  #vektor čísel
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
subset(x,x<12) # vybere čísla větší než 12, alternativa x[x<12]
 [1]  1  2  3  4  5  6  7  8  9 10 11

x<-matrix(1:15,5,5);x # tvorba a výpis matice
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    6   11    1    6
[2,]    2    7   12    2    7
[3,]    3    8   13    3    8
[4,]    4    9   14    4    9
[5,]    5   10   15    5   10

subset(x,subset=x[,1]==3) # vybere řádky, kde je v prvním sloupci číslo 3
# alternativa x[x[,1]==3,,drop=F]
     [,1] [,2] [,3] [,4] [,5]
[1,]    3    8   13    3    8

subset(x,subset=x[,1]==3,drop=T) #totéž, ale výsledek zjednoduší na vektor
# alternativa x[x[,1]==3,]
 [1]  3  8 13  3  8

subset(x,select=x[1,]==1) # vybere sloupce, kde je v 1. řádku číslo 1
# alternativa x[,x[1,]==1]
     [,1] [,2]
[1,]    1    1
[2,]    2    2
[3,]    3    3
[4,]    4    4
[5,]    5    5

mytab<-
data.frame(Strava=rep(c("mas","mix"),each=5),pH=c(7.5,7.1,8.2,8.0,7.9,7.6,7.5,7.2
,7.9,8.2)) # tabulka dat
subset(mytab,Strava=="mix")# vybere řádky, kde Strava = mix
# alternativa x[x$Strava==“mix“,]
   Strava  pH
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2

subset(mytab,Strava=="mix",select=2) # vybere řádky, kde Strava=mix, 2. sloupec
# alternativa x[x$Strava==“mix“,2]
    pH
6  7.6
7  7.5
8  7.2
9  7.9
10 8.2
```

Kontrolní úkoly

8. Vytvořte vektor s názvem vec1 s 40 prvky tvořenými náhodnými čísly z Poissonova rozdělení s průměrem kolem 3.
9. Vypište 10. až 20. prvek vektoru vec1
10. Vypište vektor tvořený 1.,3., 5. a 17. prvkem.
11. Vypište prvky na sudé pozici, na liché pozici a každý pátý prvek. Pokuste se použít také způsob pomocí logických hodnot (viz výsledky) a vysvětlete jej.
12. Vypište všechny prvky, které jsou menší než 2, větší než 2 a zároveň menší než 6
13. Vypište všechny prvky, které nejsou rovny 3, pak všechny kromě 10. prvku
14. Vypište všechny prvky kromě 10. až 20., pak všechny kromě 5., 8. a 25. prvku
15. Vektor vec1 použijte pro tvorbu matice o 10 řádcích a 4 sloupcích nazvanou mat1.
16. Přečtěte prvek na 6. řádku ve 3. sloupci matice mat1. Přečtěte první tři řádky matice. Přečtěte sloupec 2 a 4 matice.

17. Přečtěte oblast tvořící řádky 1, 2, 3 a sloupce 2, 3, 4. Přečtěte buňky tvořící průnik těchto řádků a sloupců tzn. body matice se souřadnicemi [1,2], [2,3], [3,4].
18. Vypište ty řádky matice, které mají v prvním sloupci číslo 3.
19. Z vektoru vec1 vytvořte pole ar1 s rozměry 4, 5, 2. Přečtěte prvek se souřadnicemi [2,4,2]. Vypište část pole tvořenou pouze prvními řádky, pak ještě jednou tak, aby byla zachována struktura trojrozměrného pole.
20. Načtěte následující tabulku dat a nazvěte ji myt. Tabulka obsahuje výsledky experimentů na třech polích (field) s různými prvky dodanými experimentálně (fertil). Pro každou kombinaci byla vždy provedena dvě měření výšky dospělé rostliny.

```
   field fertil height
1      a     Ca   12.1
2      b      P    9.8
3      c      P    8.9
4      a     Ca   11.3
5      b     Ca   11.0
6      c     Ca   11.9
7      a      P   10.6
8      b      P    9.7
9      c      P    9.2
10     a      P   10.3
11     b     Ca   11.2
12     c     Ca   10.4
```

21. Vypište řádky 2-6 tabulky myt. Vypište vektor výšek rostlin. Vypište první dva sloupce tabulky myt (pole field a fertil).
22. Zkontrolujte strukturu tabulky. Jakého typu jsou první dva sloupce? Převeďte 1. sloupec na vektor typu character a následně zpět na faktor.
23. Vypište pouze ty řádky, kde přidaná látka je Ca. Vypište pouze hodnoty výšky rostliny (height) z pole b.
24. Převeďte tabulku myt na seznam (list) nazvaný myls. Vypište první položku seznamu (field) celou a následně první prvek této položky.
25. K vektoru vec1 (příklad 8) přidejte vektor s hodnotami 1-10 od 10. pozice. Proveďte stejnou operaci, ale přidejte vektor na konec.
26. Zjistěte délku vektoru vec1. Zjistěte pomocí stejné funkce kolikrát je ve vektoru číslo 3 a kolik je čísel větších nebo rovno 4.
27. Zjistěte délku matice mat1 (úkol 15), pole ar1 (úkol19), tabulky myt (úkol 20), seznamu myls (úkol 24). Pokuste se vysvětlit co znamenají hodnoty.
28. Slučte vektory čísel 5-10 a 10-15 do sloupců a pak do řádků.
29. Vytvořte dva sloupce tahů sportky (tah1, tah2) s náhodně generovanými čísly (viz předchozí kapitoly) a uložte do objektu sportka.
30. Slučte vektor prvních 5 písmen abecedy a vektor 1:5 do sloupců. Jakého typu je výsledný objekt?
31. Vypište pořadí čísel, která padla v prvním tahu sportky (viz úkol 29).
32. Vypište pořadí čísel vektoru c(5,1,2,2,3,4,5,1,1,1). Zkuste různé ties metody a předem odhadujte, jaký bude výsledek. Vysvětlete rozdílnost výsledků.
33. Seřaďte předchozí vektor vzestupně. Seřaďte vektor vec1 (příklad 8) sestupně.
34. Setřiďte řádky tabulky myt (úkol 20) podle pole a následně podle typu přidaného prvku.
35. Zjistěte obor všech hodnot vektoru vec1 (kterých hodnot nabývá).
36. Zjistěte počet řádků a sloupců tabulky myt. Zjistěte dimenze pole ar1 (úkol19).

Funkce pro práci s maticí a polem

- **t**(x) – transponuje matici *x* (zamění řádky za sloupce)

```
x<-matrix(c(1,2,5,1,4,3,3,7,1,4,2,3),3,4)
     [,1] [,2] [,3] [,4]
[1,]    1    1    3    4
[2,]    2    4    7    2
[3,]    5    3    1    3

t(x)
     [,1] [,2] [,3]
[1,]    1    2    5
[2,]    1    4    3
[3,]    3    7    1
[4,]    4    2    3
```

- **det**(x) – vypočítá determinant čtvercové matice *x*

```
my.mat<-matrix(1:6,3,3)
my.mat
     [,1] [,2] [,3]
[1,]    1    4    1
[2,]    2    5    2
[3,]    3    6    3
det(my.mat)
[1] 4.996004e-16
```

- **diag**(x) – vypíše nebo vytvoří diagonálu matice *x*

```
diag(x)
[1] 1 4 1
diag(my.mat)
[1] 1 5 3

diag(c(1,2,1,4))
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    1    0
[4,]    0    0    0    4
```

- **%*%** – násobení matic (sčítání matic je stejné, jako normální sčítání)

```
x %*% t(x)
     [,1] [,2] [,3]
[1,]   27   35   23
[2,]   35   73   35
[3,]   23   35   44

my.mat %*% (x)
     [,1] [,2] [,3] [,4]
[1,]   14   20   32   15
[2,]   22   28   43   24
[3,]   30   36   54   33
```

- **rownames**(x, do.NULL=T, prefix="row"), **colnames**(x, do.NULL=T, prefix="col") – vypíše nebo nastaví jména řádků (sloupců)
  - *do.NULL* – logical, F vrací názvy na základě prefix a čísla řádku (sloupce), i když nejsou vytvořeny
  - *prefix* – character, jaká má být použita předpona pro vytvořené názvy

```
x<-matrix(1:15,3,5) #vytvoří matici
colnames(x) #názvy sloupců neexistují
NULL
colnames(x,do.NULL=F) #vektor potenciálních názvů sloupců
[1] "col1" "col2" "col3" "col4" "col5"
colnames(x,do.NULL=F,prefix="sloupec")#názvy řádků malými písmeny abecedy
[1] "sloupec1" "sloupec2" "sloupec3" "sloupec4" "sloupec5"
```

```
rownames(x)<-letters[1:3] #názvy řádků malými písmeny abecedy
x
  [,1] [,2] [,3] [,4] [,5]
a    1    4    7   10   13
b    2    5    8   11   14
c    3    6    9   12   15
rownames(x) # přečte názvy řádků
[1] "a" "b" "c"
```

Kontrolní úkoly

37. Načtěte matici mymat (5 řádků, 5 sloupců) s náhodnými celými čísly od 0 do 25 (čísla se mohou opakovat). Následně vypište čísla na diagonále.
38. Vypočítejte determinant matice, pak matici transponujte a opět vypočítejte determinant.
39. Vynásobte matici mymat transponovanou maticí mymat.
40. Nazvěte sloupce matice malými písmeny.

Funkce pro práci s tabulkou dat, seznamem nebo faktorem

- **names**(x) – v případě, že objekt má položky typu tags (seznam – list nebo tabulka dat – data.frame), pak vypíše názvy položek. Pomocí names lze tyto názvy také měnit.

```
# tvorba tabulky dat (data.frame) se třemi sloupci a přejmenování sloupců
x1<-data.frame(exp=rep(1:3,2),latka=rep(c("a","b"),3),hodn=seq(3,by=0.5,len=6))
x1
  exp latka hodn
1   1     a  3.0
2   2     b  3.5
3   3     a  4.0
4   1     b  4.5
5   2     a  5.0
6   3     b  5.5
names(x1) #výpis názvů jednotlivých polí (položek tags)
[1] "exp"   "latka" "hodn"
names(x1)<-c("a1","b1","c1") #změna názvů všech polí
x1
  a1 b1  c1
1  1  a 3.0
2  2  b 3.5
3  3  a 4.0
4  1  b 4.5
5  2  a 5.0
6  3  b 5.5

# tvorba seznamu (list) a přejmenování položek tags
x<-list(x1=1:3,y1=1:10)
x
$x1
[1] 1 2 3

$y1
 [1]  1  2  3  4  5  6  7  8  9 10
names(x) #výpis názvů položek (tags)
[1] "x1" "y1"
names(x)<-c("x2","y2") #změna názvů položek (tags) na x2 a y2
names(x) #znovu výpis názvů položek (tags) – jsou změněny
[1] "x2" "y2"
x #výpis seznamu
$x2
[1] 1 2 3

$y2
```

```
 [1]  1  2  3  4  5  6  7  8  9 10
```

- **factor**(x=character(), levels=sort(unique.default(x), na.last=T), labels=levels, exclude=NA, ordered=is.ordered(x))) – vytváří z vektoru objekt typu faktor (nominální proměnná), viz kapitola vytváření objektů.

```
kc<-c(5,2,2,2,1,6,1,7,4,5,0,1,6,2,4,4,2,7,4,2,2,3,3,3,1,5,3)
factor(kc) #vektor čísel transformován na faktor
 [1] 5 2 2 2 1 6 1 7 4 5 0 1 6 2 4 4 2 7 4 2 2 3 3 3 1 5 3
Levels: 0 1 2 3 4 5 6 7
factor(kc,levels=7:0) #vypsán vektor s nastavenými úrovněmi (levels, hodnoty)
 [1] 5 2 2 2 1 6 1 7 4 5 0 1 6 2 4 4 2 7 4 2 2 3 3 3 1 5 3
Levels: 7 6 5 4 3 2 1 0
factor(kc,ordered=T) #faktor ordinální (u hodnot je známo pořadí)
 [1] 5 2 2 2 1 6 1 7 4 5 0 1 6 2 4 4 2 7 4 2 2 3 3 3 1 5 3
Levels: 0 < 1 < 2 < 3 < 4 < 5 < 6 < 7
factor(kc,levels=6:0)
 [1] 5    2    2    2    1    6    1    <NA> 4    5    0    1    6    2    4
[16] 4    2    <NA> 4    2    2    3    3    3    1    5    3
Levels: 6 5 4 3 2 1 0
> factor(kc,ordered=T,label="druh")
 [1] druh6 druh3 druh3 druh3 druh2 druh7 druh2 druh8 druh5 druh6 druh1 druh2
[13] druh7 druh3 druh5 druh5 druh3 druh8 druh5 druh3 druh3 druh4 druh4 druh4
[25] druh2 druh6 druh4
Levels: druh1 < druh2 < druh3 < druh4 < druh5 < druh6 < druh7 < druh8

x<-factor(kc,levels=7:0)
sort(x) #jestliže jsou levels nastaveny, pak sort řadí podle levels
 [1] 7 7 6 6 5 5 5 4 4 4 4 3 3 3 3 2 2 2 2 2 2 2 1 1 1 1 0
Levels: 7 6 5 4 3 2 1 0
```

- **levels**(x) – vypíše nebo nastaví vektor všech hodnot daného objektu typu faktor

```
kc<-factor(c(5,2,2,2,1,6,1,7,4,5,0,1,6,2,4,4,2,7,4,2,2,3,3,3,1,5,3))
levels(kc) #výpis hodnot faktoru
[1] "0" "1" "2" "3" "4" "5" "6" "7"
levels(kc)[8]<-NA #vypuštěna hodnota 7 pro daný objekt
kc
 [1] 5    2    2    2    1    6    1    <NA> 4    5    0    1    6    2    4
[16] 4    2    <NA> 4    2    2    3    3    3    1    5    3
Levels: 0 1 2 3 4 5 6

kc<-factor(c(5,2,2,2,1,6,1,7,4,5,0,1,6,2,4,4,2,7,4,2,2,3,3,3,1,5,3))
levels(kc)<-letters[1:8] #hodnoty přenastaveny
kc
 [1] f c c c b g b h e f a b g c e e c h e c c d d d b f d
Levels: a b c d e f g h
```

Kontrolní úkoly

41. Načtěte tabulku z úkolu 20 (uložte pod názvem tab2). Vypište názvy sloupců této tabulky.
42. Změňte název druhého sloupce z "fertil" na "hnoj".
43. Ověřte, zda je ve sloupci 2 proměnná typu faktor. Jestliže ano, zjistěte jakých může nabývat hodnot?
44. Změňte názvy hodnot z Ca a P na vápník a fosfor.
45. Vytvořte vektor 20 náhodných čísel 0 nebo 1 s názvem "test1". Převeďte vektor test1 na vektor s proměnnou třídy faktor tak, že 1 bude v pořadí 1. hodnota a 0 bude v pořadí 2. hodnota. Změňte hodnoty faktoru tak, že 0 bude "ne" a 1 bude "ano".

Funkce pro práci s řetězci a znaky

- **paste**(…, sep=" ", collapse=NULL) – sloučí vektory … do řetězců a oddělí je značkou separátoru sep
  - *collapse* – určí jakým znakem mají být výsledné vektory sloučeny dohromady

```
paste(c("a","b","c","d","e"),1:5)
[1] "a 1" "b 2" "c 3" "d 4" "e 5"
paste(c("a","b","c","d","e"),1)
[1] "a 1" "b 1" "c 1" "d 1" "e 1"
paste("loc",1:5,sep="_")
[1] "loc_1" "loc_2" "loc_3" "loc_4" "loc_5"
paste(c("a","b","c","d","e"),1,sep="")
[1] "a1" "b1" "c1" "d1" "e1"
paste(c("a","b","c","d","e"),1:2,sep="-",collapse=",")
[1] "a-1,b-2,c-1,d-2,e-1"
paste(c("a","b","c","d","e"),collapse="")
[1] "abcde"
```

- **tolower**(x), **toupper**(x), **casefold**(x, upper=F) – převede řetězec na malá (velká) písmena, nebo podle vektoru v upper

```
a<-"Řetězec"
toupper(a) #vše velkými písmeny
[1] "ŘETĚZEC"
tolower(a) #vše malými písmeny
[1] "řetězec"
casefold("AxbC",upper=F) #vše malými písmeny
[1] "axbc"
```

- **chartr**(old, new, x) – v řetězci *x* nahradí staré znaky *old* novými *new*

```
a<-"částečně"
chartr("áě","ae",a) #v řetězci "a" změní č za c, á za a, ě za e
[1] "castecne"
```

- **noquote**(x,) – vynechá mezery při vypisování řetězce

```
noquote(a)
[1] částečně
```

- **substr**(x, start, stop) – vyřízne z textu *x* nebo umístí do textu část od–do

```
substr("abcdef",start=2,stop=4) #vyřízne z textu od 2 do 4 znaku
[1] "bcd"
substr("částečně",start=2,stop=15) #vyřízne z textu od 2 znaku do konce
[1] "ástečně"
a<-"částečně"
substr(a,start=2,stop=4)<-"ham" #nahradí řetězcem ham 2.-4. znak v řetězci a
a
[1] "čhamečně"
```

- **nchar**(x, type = "bytes") – vrátí počet znaků řetězce *x* (znaky brány podle type viz help(nchar))

```
nchar("abcede") #počet znaků v řetězci
[1] 6
nchar(c("aaa","aaaaa"))#počet znaků ve dvou řetězcích
[1] 3 5
```

- **strsplit**(x, split, extended=T, fixed=F, perl=F) – rozdělí text *x* podle zadaného řetězce. Vrátí seznam (zrušení seznamu funkcí unlist) možnosti spec. vyhledávání help(regex) v help
  - *split* – znak pro rozdělení řetězce
  - *extended, perl* – při vyhledávání znaku rozdělení použity speciální znaky
  - *fixed* – žádné speciální znaky pro vyhledávání nebudou použity (T)

```
strsplit("ac-bd-cd", split="-")#rozdělení podle pomlčky
```

```
[[1]]
[1] "ac" "bd" "cd"

strsplit("ac.bd.cd", split=".", fixed=T) #použito fixed "." je zástupný znak)
[[1]]
[1] "ac" "bd" "cd"

strsplit("ac.bd.cd", split="") #každý znak zvlášť
[[1]]
[1] "a" "c" "." "b" "d" "." "c" "d"

unlist(strsplit("Pavel",split="v"))#rozdělení a vytvoření vektoru (ne seznam)
[1] "Pa" "el"

strsplit(c("ac-bd-cd","a-b-c"), split="-")#rozdělení více řetězců podle pomlčky
[[1]]
[1] "ac" "bd" "cd"

[[2]]
[1] "a" "b" "c"

strsplit(c("ac-bd-cd","afbfc"), split=c("-","f"))#rozdělení pomocí více znaků
[[1]]
[1] "ac" "bd" "cd"

[[2]]
[1] "a" "b" "c"
```

Práce s funkcemi a výrazy

Práce s funkcemi tvoří převážnou část analýz v R. Většinu funkcí můžeme přímo prohlížet pomocí výpisu názvu funkce bez závorek. Tento výpis funkce není umožněn u části funkcí (zejména v základních knihovnách jsou to funkce typu internal, primitive, external), protože jsou psány v interním kódu.

Pro základní operace je výčet funkcí v R dostačující. Je však možné, že při častém používání sledu určitých budete chtít celý proces urychlit a zjednodušit. Pro tento případ je možné nadefinovat vlastní funkci (viz function).

- **`function`**`(arg1=x, …)` – definuje funkci s argumenty *`arg1`* (nastaven na *x*) …

```
# funkce, která vypočítá přeponu rovnostranného trojúhelníka
prepona<-function(strA,strB) sqrt(strA^2+strB^2)
prepona #výpis funkce
function(strA,strB) sqrt(strA^2+strB^2)
prepona(3,4) #výpočet pro strA=3 a strB=4
[1] 5

# funkce, která vypočítá body na přímce y=a*x+b pro zadané x a zobrazí rovnici
primka<-function(a=1,b=0,x=1:10){   #název, argumenty a přednastaveny hodnoty
myx<-a*x+b #vypočítány hodnoty y
myrov<-paste(a,"x +",b,collapse="") #tvar rovnice
list(rovnice=myrov,hodnoty=myx) #tisk seznamu s rovnicí a hodnotami
}
primka() #výpočet s přednastavenými hodnotami
$rovnice
[1] "1 x + 0"

$hodnoty
 [1]  1  2  3  4  5  6  7  8  9 10


primka(a=2,b=3,x=seq(from=1,to=2,by=0.1)) #nastaveny jiné argumenty
$rovnice
[1] "2 x + 3"
```

```
$hodnoty
 [1] 5.0 5.2 5.4 5.6 5.8 6.0 6.2 6.4 6.6 6.8 7.0

(function(x) x^2)(5) #anonymní funkce mocniny čísla x, dosazeno číslo 5
[1] 25
```

- **args**(f), **formals**(f) – vypíše argumenty funkce, s možností je přednastavit (u formals)

```
args(strsplit) #výpis argumentů funkce strsplit
function (x, split, extended = TRUE, fixed = FALSE, perl = FALSE)
NULL

formals(strsplit) #výpis argumentů funkce strsplit s možností nastavit je
$x

$split

$extended
[1] TRUE

$fixed
[1] FALSE

$perl
[1] FALSE

args(primka) #výpis argumentů funkce primka z předchozí ukázky
function (a = 1, b = 0, x = 1:10)
formals(primka)$b<-2 #změna přednastavené hodnoty u b z 0 na 2
args(primka) #opětovný výpis argumentů
function (a = 1, b = 2, x = 1:10)
NULL
```

Kontrolní úkoly

46. Vytvořte vektor obsahující název lokalita 1, lokalita 2 ... lokalita 100. Poté vytvořte vektor vektor L1 – L100 (bez mezery). Zopakujte úkol tak, aby se hodnoty vypisovaly bez uvozovek.
47. Spojte předchozí vektor tak, že bude napsáno L1, L2, L3, ..., L100. Vypočítejte, kolik má řetězec znaků.
48. Tuto větu použijte jako řetězec nazvaný ret1. Transformujte ret1 tak aby byl psán velkými písmeny.
49. Zaměňte v řetězci znak "t" za "r". Vyberte z řetězce podřetězec od 14 do 18 znaku.
50. Rozdělte řetězec do vektoru na jednotlivá slova.
51. Zjistěte argumenty funkce chartr.
52. Vytvořte funkci msum, která vám sečte čísla od x do y. Proměnné x a y nastavte tak, aby se při nezadání argumentů sečetla čísla od 1 do 1, tzn.1. Otestujte na číslech 2 a 10.

Hromadné provádění výpočtů na objektech

- **rowsum**(x, group, reorder=T, na.rm=F …) – sečte u objektů *x* typu matice řádky
  - *group* – vektor, podle kterého jsou řádky seskupeny
  - *reorder* – výsledky seřazeny podle pořadí hodnot v group
  - *na.rm* – vynechány hodnoty NA

```
x<-matrix(1:6,3,4)
x
     [,1] [,2] [,3] [,4]
[1,]    1    4    1    4
```

```
[2,]    2    5    2    5
[3,]    3    6    3    6
rowsum(x,group=c(1,1,1))#všechny řádky patří do jedné skupiny, součet všech
  [,1] [,2] [,3] [,4]
1    6   15    6   15
rowsum(x,group=c(1,2,2))#první samostatně, další dva sečteny
  [,1] [,2] [,3] [,4]
1    1    4    1    4
2    5   11    5   11
rowsum(x,group=c(3,1,2)) #první samostatně, další dva sečteny
  [,1] [,2] [,3] [,4]
1    2    5    2    5
2    3    6    3    6
3    1    4    1    4
rowsum(x,group=c(3,1,2),reorder=F)
  [,1] [,2] [,3] [,4]
3    1    4    1    4
1    2    5    2    5
2    3    6    3    6
```

- **rowSums colSums rowMeans colMeans** (x, na.rm=F, dims=1) – vypočítá sumu (průměr) v řádcích (sloupcích). Nastavení *dims* mění rozsah součtu u vícerozměrných polí.

```
x<-matrix(1:6,3,4)
x
     [,1] [,2] [,3] [,4]
[1,]    1    4    1    4
[2,]    2    5    2    5
[3,]    3    6    3    6
rowSums(x) # suma v každém řádku
[1] 10 14 18
colSums(x) # suma v každém řádku
[1]  6 15  6 15

x<-array(1:24,dim=c(3,4,2)) #trojrozměrné pole
x
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
rowSums(x,dim=1)#suma pro každý řádek
[1] 44 52 60
> rowSums(x,dim=2) #suma pro každý řádek a sloupec
     [,1] [,2] [,3] [,4]
[1,]    2    8   14   20
[2,]    4   10   16   22
[3,]    6   12   18   24
```

- **tabulate**(bin, nbins=max(1, bin)) – spočítá četnosti jednotlivých celých kladných hodnot vektoru *x* (od 1 až do *nbins*), jiná čísla ignoruje

```
tabulate(c(5,5,4,4,4,2))#počet čísel 1,2,3,4,5
[1] 0 1 0 3 2
tabulate(c(1,1,4,1,2,3,1,4,4,7)) #obdobné jako předchozí
[1] 4 1 1 3 0 0 1
```

- **`table`**`(…, exclude=c(NA, NaN), dnn=list.names(…), deparse.level=1)` – spočítá četnosti faktoru nebo kombinace faktorů … (popř. hladin faktoru, viz factor). Využívá se při tvorbě kontingenčních tabulek. Podobné funkce také `xtabs` a `ftable`.
  - *`exclude`* – hodnoty, které mají být vynechány
  - *`dnn`* – jména výsledných dimenzí
  - *`deparse.level`* – určí typ tvorby názvu jmen dimenzí

```
table(c(0,0,1,1,4,1,2,3,1,4,4,7)) #četnosti jednotlivých hodnot
0 1 2 3 4 7
2 4 1 1 3 1
table(factor(c(0,0,1,1,4,1,2,3,1,4,4,7),levels=0:7))#faktor s 8 úrovněmi
0 1 2 3 4 5 6 7
2 4 1 1 3 0 0 1

x1<-c(0,0,1,1,4,1,2,3,1,4,4,7)
x2<-rep(1:2,6)#vektor rozdělí x1 do dvou skupin
x1
 [1] 0 0 1 1 4 1 2 3 1 4 4 7
x2
 [1] 1 2 1 2 1 2 1 2 1 2 1 2
table(x2,x1) #počty hodnot x1 v jednotlivých skupinách x2
   x1
x2  0 1 2 3 4 7
  1 1 2 1 0 2 0
  2 1 2 0 1 1 1
table(x2,x1,dnn=c("a","b"))#totéž, ale s názvy dimenzí
   b
a   0 1 2 3 4 7
  1 1 2 1 0 2 0
  2 1 2 0 1 1 1
```

- **`apply`**`(X, MARGIN, FUN, …)` – vypočítá funkci pro každý řádek (nebo další dimenze) objektu *`X`*
  - *`MARGIN`* – vektor dimenzí, pro které má být výpočet proveden (sloupce=2, řádky=1)
  - *`FUN`* – funkce, která má být provedena (může být i vlastní)
  - … – argumenty příslušné funkce

```
x1<-matrix(1:6,2,3) #matice 2 řádky, 3 sloupce
x1
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
apply(x1,1,sum) #sečti přes řádky
[1]  9 12
apply(x1,2,prod) #součin přes sloupce
[1]  2 12 30
apply(x1,2,function(x) sum(x*2-3)) #vlastní funkce součet všech položek*2-3
[1]  0  8 16


x<-array(1:24,dim=c(3,4,2)) #trojrozměrné pole
x
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
```

```
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24

apply(x,3,sum) #suma přes třetí dimenzi
[1]  78 222
apply(x,1,sum) #suma přes první dimenzi
[1]  92 100 108
apply(x,c(1,3),sum) #suma přes první a třetí dimenzi
     [,1] [,2]
[1,]   22   70
[2,]   26   74
[3,]   30   78
```

- **lapply**(X, FUN, …), **sapply**(X, FUN, ..., simplify=T, USE.NAMES=T) – provádí výpočet určité funkce v jednotlivých položkách seznamu *X*
  - *FUN* – funkce, která má být provedena (může být i vlastní)
  - *simplify* – má být výsledek zjednodušen?
  - *USE.NAMES* – použít názvy položek seznamu pro výsledek?

```
x<-list(x=1:5,y=5:14,z=2:6) #vytvořen seznam
x
$x
[1] 1 2 3 4 5

$y
 [1]  5  6  7  8  9 10 11 12 13 14

$z
[1] 2 3 4 5 6

lapply(x,sum) #suma přes jednotlivé položky
$x
[1] 15

$y
[1] 95

$z
[1] 20

sapply(x,sum) #totéž, jen v jiném formátu
 x  y  z
15 95 20
```

- **mapply**(FUN, …, MoreArgs=NULL, SIMPLIFY=T, USE.NAMES=T) – umožňuje provést funkce s různým nastavením pro každou položku (*MoreArgs*)

```
mapply(seq, from=c(1,3,7),to=c(4,12,20))#vytvoří postupně sekvence
[[1]]
[1] 1 2 3 4

[[2]]
 [1]  3  4  5  6  7  8  9 10 11 12

[[3]]
 [1]  7  8  9 10 11 12 13 14 15 16 17 18 19 20

mapply(rep, 2:4, times=c(3,5,2)) #replikuje 3x,5x,2x čísla 2,3,4
[[1]]
[1] 2 2 2

[[2]]
[1] 3 3 3 3 3

[[3]]
[1] 4 4
```

- **tapply**(X, INDEX, FUN = NULL, ..., simplify = TRUE) – používá se pro aplikaci funkce na roztříděná data v tabulce dat
  - *FUN* – funkce, která má být provedena (může být i vlastní)
  - *simplify* – má být výsledek zjednodušen?
  - *index* – seznam položek pro roztřídění

```
# tvorba tabulky dat (data.frame) se třemi sloupci
x1<-data.frame(exp=rep(1:3,2),latka=rep(c("a","b"),3),hodn=seq(3,by=0.5,len=6))

x1
  exp latka hodn
1   1     a  3.0
2   2     b  3.5
3   3     a  4.0
4   1     b  4.5
5   2     a  5.0
6   3     b  5.5

tapply(x1$hodn,INDEX=x1$latka, mean) # průměr pro jednotlivé typy látky
  a   b
4.0 4.5

#počty měření pro kombinaci faktorů látka a expozice
tapply(x1$hodn,INDEX=list(lat=x1$latka,exp=x1$exp), length)
   exp
lat 1 2 3
  a 1 1 1
  b 1 1 1

x<-cbind(data=c(1,1,2,4,2,8,4,6,1),skup=rep(1:3,each=3));x
      data skup
 [1,]    1    1
 [2,]    1    1
 [3,]    2    1
 [4,]    4    2
 [5,]    2    2
 [6,]    8    2
 [7,]    4    3
 [8,]    6    3
 [9,]    1    3

tapply(x[,1],x[,2],FUN=max) # maximum pro jednotlivé skupiny
1 2 3
2 8 6
tapply(x[,1],x[,2],FUN=mean) # průměr pro jednotlivé skupiny
       1        2        3
1.333333 4.666667 3.666667
tapply(x[,1],x[,2],FUN=length) # počet hodnot pro jednotlivé skupiny
1 2 3
3 3 3
```

- **by**(data, indices, FUN) – podobné jako tapply, rozdělí *data* podle vektoru (nebo seznamu), *indices* (podle faktorů nebo speciálních vlastností) a pro každou skupinu vypočítá funkci *FUN*

```
mytab<-
data.frame(Strava=rep(c("mas","mix"),each=5),pH=c(7.5,7.1,8.2,8.0,7.9,7.6,7.5,7.2
,7.9,8.2))#tvorba tabulky
mytab
   Strava  pH
1     mas 7.5
2     mas 7.1
3     mas 8.2
4     mas 8.0
5     mas 7.9
6     mix 7.6
7     mix 7.5
```

```
8     mix 7.2
9     mix 7.9
10    mix 8.2

by(mytab$pH,mytab$Strava,FUN=mean)# průměr pH pro jednotlivé typy stravy
INDICES: mas
[1] 7.74
------------------------------------------------------------
INDICES: mix
[1] 7.68

by(mytab$pH,INDICES=mytab$pH<8,FUN=length) # počty hodnot pro skupinu<8 a >=8
INDICES: FALSE
[1] 3
------------------------------------------------------------
INDICES: TRUE
[1] 7
```

- **aggregate**(x, by, FUN, ...) – sloučí položky v tabulce dat x, položky jsou slučovány podle seznamu v argumentu *by* a sloučení se provádí pomocí funkce FUN.

```
#vypočítá četnosti pro jednotlivé hodnoty, výsledek je opět tabulka dat
aggregate(c(1,1,1,2,2,3,3,3,3),by=list(samp=c(1,1,1,2,2,3,3,3,3)),length)
  samp x
1    1 3
2    2 2
3    3 4


   Strava  pH
1     mas 7.5
2     mas 7.1
3     mas 8.2
4     mas 8.0
5     mas 7.9
6     mix 7.6
7     mix 7.5
8     mix 7.2
9     mix 7.9
10    mix 8.2

#vrací v řádcích průměry pro typy stravy, výsledek je opět tabulka dat
aggregate(mytab$pH,by=list(diet=mytab$Strava),mean)
  diet    x
1  mas 7.74
2  mix 7.68
```

Kontrolní úkoly

53. Vytvořte matici mat2 (10 řádků × 5 sloupců) s náhodnými čísly z normálního rozdělení. Proveďte součty po řádcích a po sloupcích.
54. Vytvořte matici, která bude obsahovat součty prvních pěti a posledních pěti řádků.
55. Vygenerujte do objektu vec2 100 náhodných čísel z Poissonova rozdělení (s průměrem 6). Spočítejte četnosti jednotlivých hodnot.
56. Proveďte operaci ještě jednou tak, aby se v přehledu četností vyskytovaly všechny hodnoty od 0 do maxima.
57. Vytvořte vektor vec3 obsahující 40 písmen od "a" do "e" v náhodném pořadí. Spočítejte četnosti jednotlivých písmen.
58. Načtěte následující tabulku a spočítejte součet naměřených hodnot výšky na jednotlivých polích a, b, c.

```
   field fertil height
1      a     Ca   12.1
2      b      P    9.8
```

```
3      c      P    8.9
4      a     Ca   11.3
5      b     Ca   11.0
6      c     Ca   11.9
7      a      P   10.6
8      b      P    9.7
9      c      P    9.2
10     a      P   10.3
11     b     Ca   11.2
12     c     Ca   10.4
```

59. Proveďte výpočet průměru (funkce mean) pro jednotlivá pole (field) i typy obohacení půdy (fertil) .
60. Pro matici mat2 proveďte výpočet průměru přes řádky i sloupce zaokrouhlený na 2 desetinná místa.

Shrnutí:

V R můžeme provádět načítání dat ze schránky nebo ze souboru. Vektory a tabulky dat se načítají dvěma rozdílnými funkcemi, nicméně argumenty funkcí jsou dost podobné. Nejjednodušší editace objektů je pomocí R editoru (funkce edit a fix). Opravy lze také provádět přímo přepisem jednotlivých (filtrovaných) prvků objektů. Následně lze objekty spojovat řadit, určovat pořadí jednotlivých prvků, vypisovat názvy podobjektů atd. Mimořádně užitečná je editace vytvořených funkcí. Můžeme si vytvářet vlastní funkce s argumenty, které provádí operace specifikované na naše data. Podobně jako např. MS Excel existují tzv. "kontingeční tabulky" pomocí nichž jsou prováděny hromadné operace na složitějších tabulkách, má také R zabudované funkce pro výpočty v rámci tabulek dat, seznamů, polí a matic.

Metody k zapamatování:

- načítání objektů: scan, read.table
- úprava objektů: edit, fix
- práce s prvky jednoduchých objektů (obecně): [], $, append, cbind, rbind, length, rank, sort, order, replace, duplicated, unique, ncol, nrow, dim, dimnames, subset
- práce s maticí a poli: t, det, diag, %*%, rownames, colnames
- práce s tabulkou dat, seznamem nebo faktorem: names, factor
- práce s řetězci a znaky: paste, tolower, toupper, casefold, chartr, substr, nchar, strsplit
- práce s funkcemi: function, args, formals
- hromadné provádění výpočtů: rowsum, rowSums, colSums, rowMeans, colMeans, tabulate, table, apply, sapply, lapply, mapply, tapply, by

Výsledky

```
1.  scan("clipboard")
    scan("clipboard",sep=",")
    scan("clipboard")
    scan("clipboard",sep=";",dec=",")
    scan("clipboard",sep=",")
2.  scan("clipboard",what=logical(),sep=",")¨
3.  m.vec<-scan("clipboard",what=character(),sep="\n")
4.  fix(m.vec) #do závorek přidáme znaky v příslušném tvau a editační okno, ve kterém jsme
    měnili hodnoty zavřeme (na dotaz, zda uloženit změny do objektu odpovíme ano)
5.  m.tab<-read.table("clipboard")
```

```
fix(m.tab) #přidáme sloupce, uložíme a zkontrolujeme pomocí str(m.tab)
je uvedeno v příkladech k funkci read.table
vec1<-rpois(40,3)
vec1[10:20]
vec1[c(1,3,5,17)]
vec1[seq(2,40,by=2)]; vec1[seq(1,39,by=2)]; vec1[seq(5,40,by=5)]
jiné řešení vec1[c(F,T)]; vec1[c(T,F)]; vec1[c(F,F,F,F,T)]
vec1[vec1<2]; vec1[vec1>3 & vec1<6]
vec1[vec1!=3]; vec1[-10]
vec1[-(10:20)]; vec1[-c(5,8,25)]
mat1<-matrix(vec1,10,4)
mat1[6,3]; mat1[1:3,]; mat1[,c(2,4)] #není pouze jedno řešení lze použít např. T a F
mat1[c(1,2,3),c(2,3,4)]; mat1[cbind(c(1,2,3),c(2,3,4))] # body je nutno zadat jako
dvourozměrné pole, kde každý řádek je souřadnice bodu.
mat1[mat1[,1]==3,]
ar1<-array(vec1,c(4,5,2)); ar1[2,4,2]; ar1[2,,], ar1[2,,,drop=F]
myt<-read.table("clipboard")
myt[2:6,]; myt[,3] nebo myt$height; myt[,1:2] nebo myt[,c("field","fertil")]
str(myt); myt[,1]<-as.character(myt[,1]); myt[,1]<-as.factor(myt[,1])
myt[myt[,2]=="Ca",]; myt[myt[,1]=="b",3]
myls<-as.list(myt); myls[[1]] nebo myls$field aj.; myls[[1]][1] nebo myls$field[1]
append(vec1,1:10,after=9); c(vec1,1:10)#lze ale shodně s předchozím append(vec1,1:10)
length(vec1[vec1==3]); length(vec1[vec1>=4])
použijte funkci length, u matice a pole se jedná o celkový počet prvků, u tabulky a
seznamu o počet položek (resp. polí neboli sloupců)
cbind(5:10,10:15); rbind(5:10,10:15)
sportka<-cbind(tah1=sample(1:49,7),tah2=sample(1:49,7))
cbind(pism=letters[1:3],por=1:3); class(cbind(pism=letters[1:3],por=1:3))
rank(sportka[,1])
rank(c(5,1,2,2,3,4,5,1,1,1), ties.method = můžete zadat "average", "first", "random",
"max", "min") vysvětlení v textu k funkci
sort(c(5,1,2,2,3,4,5,1,1,1);sort(vec1,decreasing=T)
myt[order(myt[,1],myt[,2]),]
unique(vec1)
ncol(myt); nrow(myt);dim(ar1)
mymat<-matrix(sample(0:25,25,rep=T),5,5); diag(mymat)
det(mymat); det(t(mymat))
mymat %*% t(mymat)
colnames(mymat)<-letters[1:5]
tab2<-read.table("clipboard"); names(tab2)
names(tab2)[2]<-"hnoj"
class(tab2$hnoj); levels(tab2$hnoj)
levels(tab2$hnoj)<-c("vápník","fosfor")
test1<-sample(0:1,20,rep=T); test1<-factor(test1,levels=c(1:0); levels(test1)<-
c("ano","ne")
paste("lokalita",1:100); paste("L",1:100,sep=""); noquote(paste("lokalita",1:100));
noquote(paste("L",1:100,sep=""))
paste("L",1:100,sep="",collapse=", "); nchar(paste("L",1:100,sep="",collapse=", "))
ret1<-"Tuto větu použijte jako řetězec nazvaný ret1" (větu přímo vkopírujte do R);
toupper(ret1)
chartr("t","r", ret1);substr(ret1,14,18)
strsplit(ret1,split=" ")
args(chartr) nebo formals(chartr)
msum<-function(x=1,y=1) sum(x:y); msum(2,10)
mat2<-matrix(rnorm(100),10,5); rowSums(mat2);colSums(mat2)
rowsum(mat2,c(1,1,1,1,1,2,2,2,2,2))
vec2<-rpois(100,6); table(vec2)
table(factor(vec2,levels=0:max(vec2)))
vec3<-sample(letters[1:5],40,rep=T); table(vec3)
mtab2<-read.table("clipboard"); tapply(mtab2$height,mtab2$field,sum)
tapply(mtab2$height,list(pole=mtab2$field,hnoj=mtab2$fertil),mean)#
round(apply(mat2,1,mean),2); round(apply(mat2,2,mean),2) nebo rowMeans a colMeans
```

ZÁKLADY GRAFIKY V R

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- budete znát základní prvky grafu v programu R;
- zvládnete jednoduchou tvorbu grafu v R;
- pochopíte argumenty funkce plot;
- naučíte se nastavovat jednotlivé parametry grafického okna;
- budete umět vytvářet hlavní typy grafů.

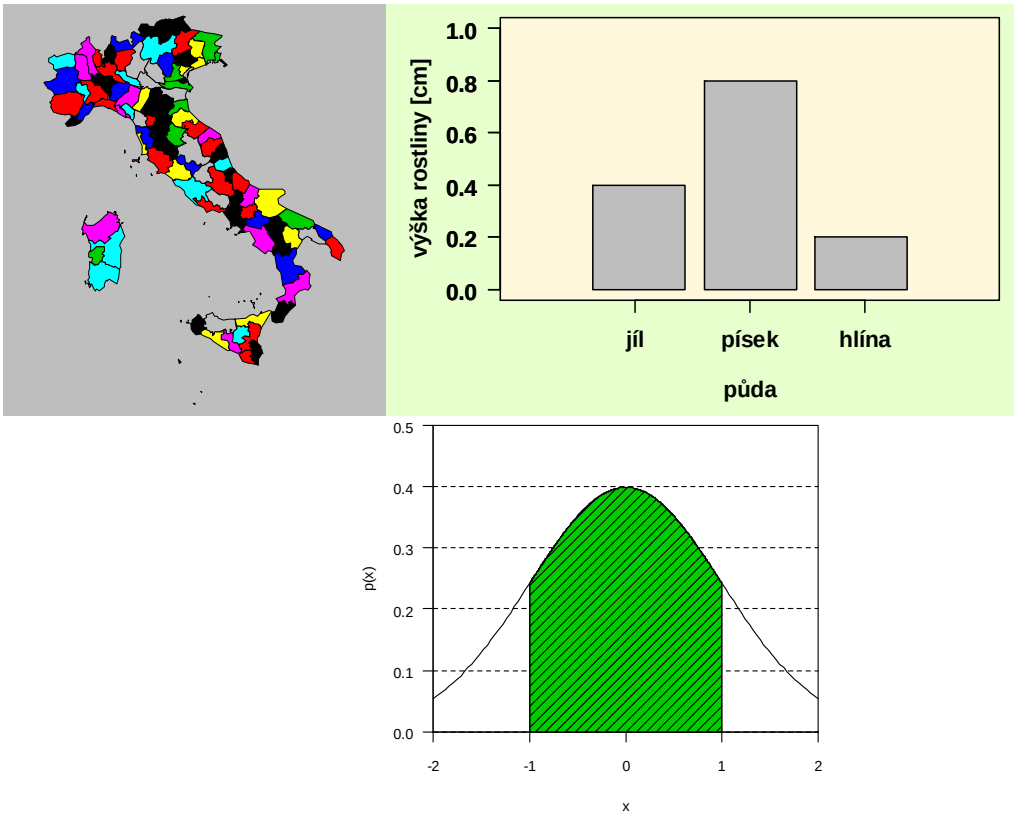
Klíčová slova: grafy, high-level, low-level grafika, parametry grafu.

Průvodce
Jestliže vám připadá, že to, co jsme se dosud učili se dá tak či onak vytvořit i v jiných počítačových programech, které již znáte, grafika vás určitě překvapí. Ten, kdo je zvyklý na běžnou práci s grafy v Excelu, bude asi v první fázi zoufalý. Měřítko grafu, jednotlivé body atd. lze také nastavovat, ale ne pokliknutím na objekt v grafu. Vše je nutno zadávat v argumentech grafických funkcí. Pokročilejší uživatelé zajásají, protože velice brzy zjistí, že tento na první pohled neobvyklý přístup přináší velkou řadu výhod. Projděte si celou kapitolu dost podrobně. V případě, že budete mít pocit, že je práce s grafy stále složitá, doporučuji trpělivě opakovat jednotlivé postupy a úkoly.

Ačkoliv je na první pohled grafika v R „méně dokonalá“ než u ostatních systémů, skutečnost je zcela jiná. Grafické výstupy jsou plně srovnatelné např. se softwarem S-plus a daleko převyšuje možnosti grafů v MS Excel a běžných statistických programech. R totiž umožňuje pracovat v několika různých úrovních grafiky:

- **High-level**. Tato úroveň je vhodná pro tvorbu jednoduchých grafů a je snadno použitelná i začátečníky v tvorbě grafů. Patří sem funkce jako plot, hist, dotplot atd.
- **Low-level**. Umožňuje mnohem více nastavení os a jednotlivých prvků grafu, nastavení přesných rozměrů grafu, vepisování a vkreslování různých objektů do grafu (kruhy, text, body, matematické křivky atd.)
- **Interaktivní grafika**. Dovoluje interaktivně přidávat data do grafu a zpětně informace extrahovat. Jedná se o specifické funkce které pracují s polohou kurzoru v grafu.

V případě, že jsou tyto grafy nedostačující, můžeme nalézt další knihovny obsahující množství jiných typů grafiky. Mezi nejpropracovanější patří balíčky `grid`, `lattice`, `iplots`, `misc3D`, `rgl`, `scatterplot3d`. Jedním z doplňujících balíčků je také knihovna maps obsahující nejrůznější mapy.



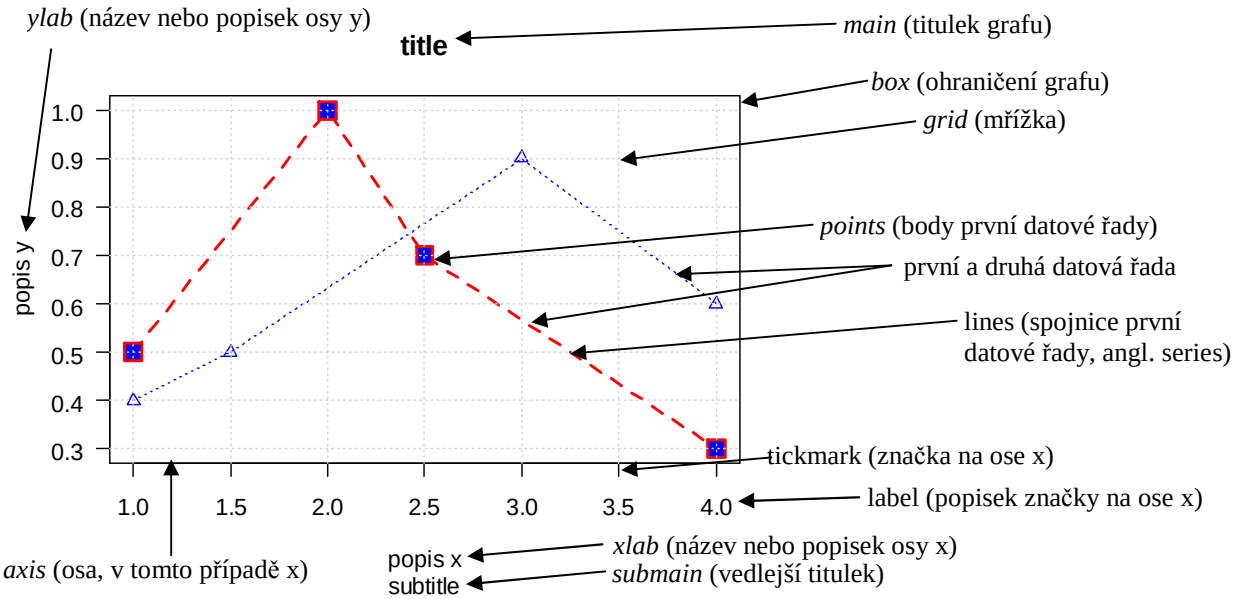
Obr 9: Ukázky grafiky v R. Mapa Itálie (library maps), sloupcový graf, vykreslení funkce Studentova rozdělení.

Přehled funkcí a vlastností používaných pro grafiku

Vzhledem k ohromnému množství různých grafických prvků, je dobré pro přehlednost uvést zjednodušený přehled základních grafických funkcí (high-level, tak také další typy.

- **Základní typy grafů.** *plot* (normální graf), *barplot* (sloupcový graf), *boxplot* (krabice s vousy), *pie* (koláčový graf), *histogram* (sloupcový graf četnosti), *matplot* (graf s více řadami dat), *persp* (prostorový graf – povrchy).
- **Speciální typy grafů.** *pairs* (skupiny XY grafů v jednom grafickém okně), *stem* (stonek s lístky), *stars* (hvězdový graf), *dotchart* (Clevelandův bodový graf), *stripchart* (pásový graf – 1D), *sunflowerplot* (slunečninový graf – body se shodnými souřadnicemi se vykreslí jako lístky vycházející z bodu), *spineplot* (speciální sloupcový graf s rozestupy a densitami), *mosaicplot* (mozaikový graf), *fourfoldplot* (čtyřlístkový graf), *filled.contour* (barevné kontury), *contour* (kontury), *coplot* (speciální matice XY grafů), *cdplot* (graf s výplní pod osou), *bxp* (jiný typ zadání boxplotu), *assocplot* (Cohen-Friendly graf), *image* (speciální typ grafu podobný filled.contour).
- **Prvky grafu vykreslitelné samostatně.** *axis* (osy), *grid* (mřížka), *legend* (legenda), *rug* (kartáč – vykresluje hustotu bodů), *title* (titulky a popisky), *text* (textová pole), *points* (body), *lines* (spojené čáry), *segments* (úsečky), *abline* (přímky), *mtext* (text na okraji grafu), *matpoints* (body ve více samostatných řadách), *matlines* (spojnice ve více samostatných řadách), *curve* (křivky, matematické funkce), *box* (obrys kolem grafu).
- **Grafické prvky.** *symbols* (kružnice, obdélníky, hvězdy atd.), *rect* (obdélník), *polygon* (mnohoúhelník), *arrows* (šipky).

- **Nastavení grafiky.** *windows* (otevření a nastavení grafického okna), *dev.set* (výběr okna pro výstup), *plot.window* (nastavení koordinát, druhá, popř. třetí osa), *par* (nastavení parametrů grafického výstupu), *split.screen* (rozdělí okno, další close.screen, erase.screen), *screen* (obdélník), *strwidth* (počítá velikost textu v grafickém okně), *locator* (čte pozici kurzoru v grafickém okně), *identify* (identifikuje nejbližší vykreslený bod od pozice kurzoru), *layout* (nastaví rozdělení okna, víc nastavení než split.screen), *frame* (podobné plot.new, vytvoří grafické okno), *xy.coords* (souřadnice x a y), *rgb* (namíchá barvu), *colors* (přednastavené barvy), *palette*, *rainbow*, *hcl*, *terrain.colors* (palety barev), *recordPlot* (uložení grafu jako proměnné), *plotmath* (vykreslení matem. značek), *windowsFons*, *Hershey* (typy fontů), další funkce (knihovna grDevices).



Obr 10: Jednotlivé části grafu (české názvy částí v závorce) a funkce popř. argumenty související s nimi.

```
#vykreslení předchozího grafu
plot(x=c(1,2,2.5,4),y=c(0.5,1,0.7,0.3),type="o",lty=2,lwd=2,col=2,pch=22,bg=4,mai
n="title",sub="subtitle",xlab="popis x",ylab="popis y",cex=2,las=1)
```

Tvorba základních grafů

Při tvorbě high-level grafů je nutné dbát na několik důležitých věcí. High-level grafy většinou mohou jako data používat několik různých objektů a podle toho se potom chovají. Například funkce `plot` nejprve rozliší typ vstupujícího objektu a následně volá jinou funkci dle příslušného objektu (např. `plot.default` – základní graf, `plot.lm` – graf pro lineární model atd.). Velkou část parametrů, které nelze nastavit přímo jako argumenty dané funkce lze pak nastavit jako tzv. parametry grafického výstupu pomocí funkce `par` (viz podkapitola Parametry grafických výstupů). Při přepnutí do okna pro grafiku se mění menu a kontextové menu tak, že umožňuje zkopírovat nebo uložit výslednou grafiku do schránky nebo do souboru v různých výstupních formátech. Údaje v palcích lze do cm převést funkcí `cm(x)`.

- **`plot`**`(x,y=NULL,type="p",xlim=NULL,ylim=NULL,log= "",main=NULL,sub=NULL, xlab=NULL,ylab=NULL,ann=par("ann"),axes=T,frame.plot=axes,panel.first= NULL, panel.last=NULL,asp=NA, …)` – vytvoří graf různého typu podle vstupních

dat (plot.default, plot.design, plot.data.frame může být funkce, tabulka, vektor atd.) – podrobnější výčet je uveden při výpisu `methods (plot)`. Výčet argumentů je pouze částečný. Lze použít také jiné argumenty, např. některé argumenty funkce `par` (viz dále). K vykreslení jednoduchého grafu stačí zadat vektor čísel, popř. faktor nebo křivku. V případě vektoru čísel je základní vykreslení nastaveno na body, u faktoru jsou to sloupce četností jednotlivých hodnot, u křivky se jedná o hladkou čáru. Názvy os jsou tvořeny automaticky, nebo je lze nastavit. Velkou část parametrů lze nastavovat jako vektory (např. barvy, symboly) tak, že může být část vykreslena jednou barvou a další část jinou).

type – `"p"` bodový, `"l"` liniový, `"h"` vertikální linie, `"o"` linie s body, `"n"` nevynášet, `"b"` přerušované linie s body, `"c"` přerušované linie bez bodů, `"s"` schodovité s první linií horizontální, `"S"` schodovité s první linií vertikální

xlim, *ylim* – vektor (2 prvky) rozmezí souřadnic na ose x (popř. y).

log – řetězec (`"x"`, `"y"` nebo `"xy"`) označující, která osa má být logaritmována.

main, *sub* – textový řetězec názvu grafu a podnázvu

xlab, *ylab* – textový řetězec názvů osy x a y

plot – logická hodnota, zda se má graf vykreslit nebo vypsat jeho vlastnosti

ann – logická hodnota, zda se mají uvádět v grafu názvy os a název a podnázev grafu a názvy os.

axes – logická hodnota, zda se mají vykreslovat osy.

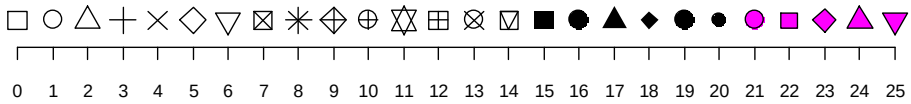
frame.plot – logická hodnota, zda má být graf ohraničen.

panel.first, *panel.last* – funkce, které se mají provést po nastavení souřadnic a před a po vykreslení grafu.

col – integer, string, vektor barev bodů, obrysů bodů nebo barvy spojnice

základní barvy (další colors, gray, rgb atd.)

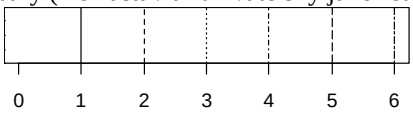
pch – integer, typ vykreslovaných bodů, od 21 lze body vybarvovat



bg – barva pozadí dvoubarevných bodů (pch je od 20 vyšší)

cex – relativní velikost znaků a bodů oproti standardním

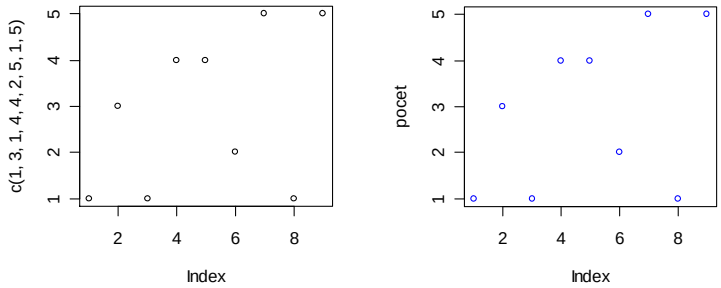
lty – typ čáry (lze nastavit i uživatelsky jako řetězec délek úseček např. "44")



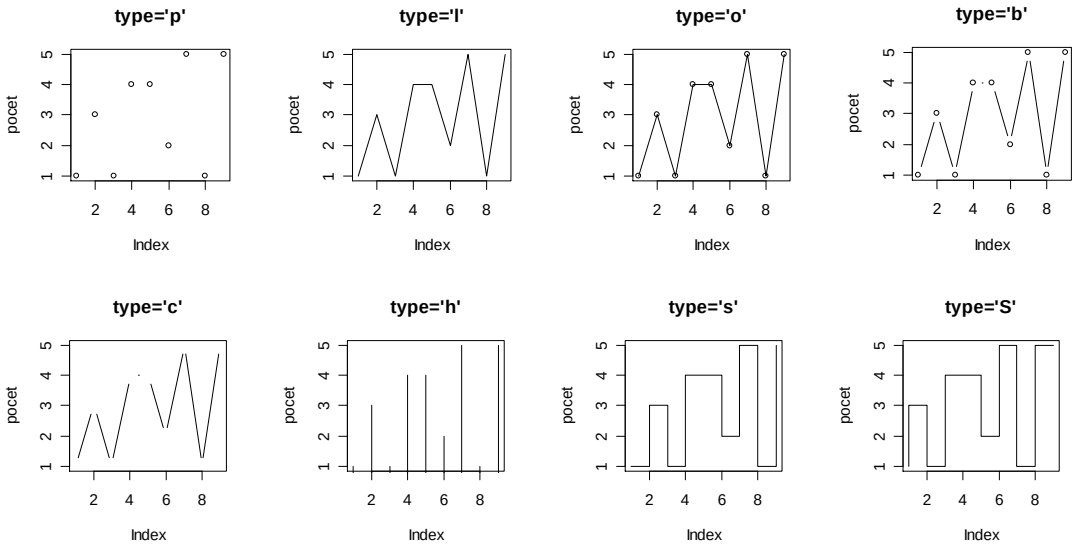
lwd – tloušťka čáry (základní = 1).

asp – poměr osy y/x. asp = 1 poměr osy x a y bude zachován.

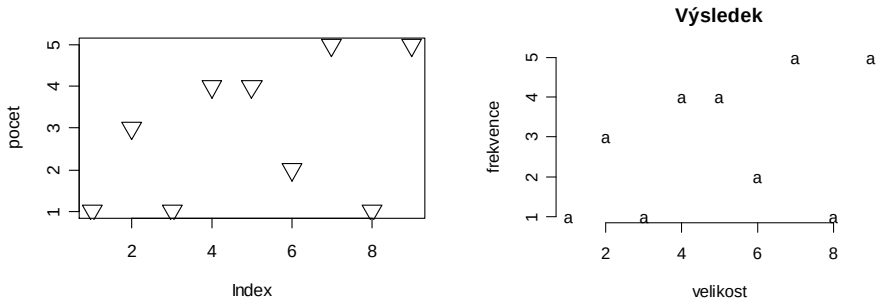
```
plot(c(1,3,1,4,4,2,5,1,5))# vložen pouze vektor, názvy os automatické
pocet<-c(1,3,1,4,4,2,5,1,5)# vektor pojmenován
plot(pocet,col=4) # osa y má název podle názvu vektoru, barva bodů modrá
```



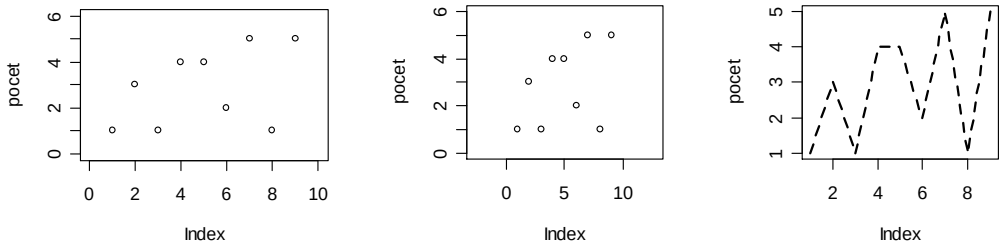

```
#vykreslení grafu plot(pocet) s argumentem type
```



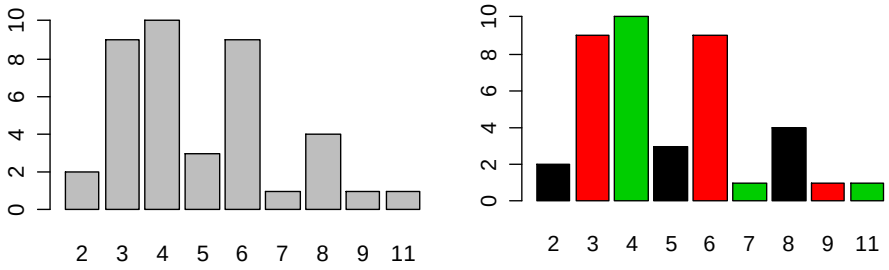

```
plot(pocet,pch=6,cex=2) #jiný symbol, větší velikost symbolu
plot(pocet,pch="a",frame.plot=F,xlab="velikost",ylab="frekvence",
     main="Výsledek") #popisky os, vypnut okraj, písmeno místo symbolu,nadpis
```




```
plot(pocet,xlim=c(0,10),ylim=c(0,6),asp=1)#nastaveno měřítko os a poměr osy y/x
plot(pocet,xlim=c(0,10),ylim=c(0,6),asp=2)#osa y 2× větší než x
plot(pocet,type="l",lty=2,lwd=2)#typ čáry 2 (přeruš.), tloušťka 2
```

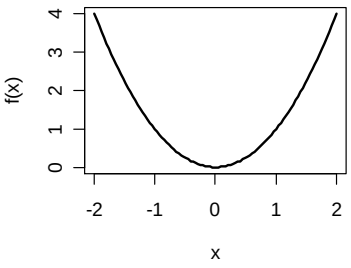


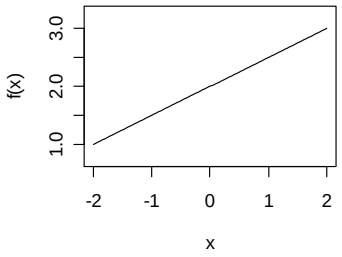

```
x<-factor(rpois(40,5))
plot(x)# vykreslení faktoru (plot.factor)
plot(x,ylim=c(0,10),col=1:3) #změněna barva
```

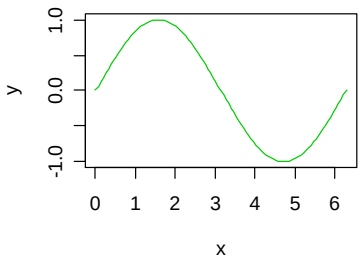


```
# v případě zadání objektu ve tvaru funkce (viz bude dále), je nutno zadat
```

```
# interval x (obor hodnot), ve kterém funkci zobrazujeme

plot(function(x) x^2,-2,2,ylab="f(x)",lwd=2) #kvadratická funkce, silná čára
plot(function(x) 0.5*x+2,-2,2,ylab="f(x)",asp=1)# přímka měřítko os 1:1
plot((function(x) sin(x)),0,2*pi, ylab="y",xlab="x",col=3)# sinusoida
```

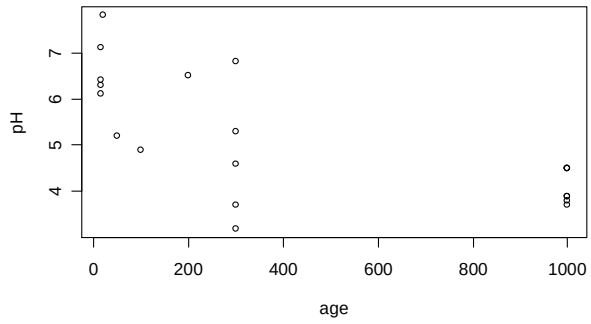


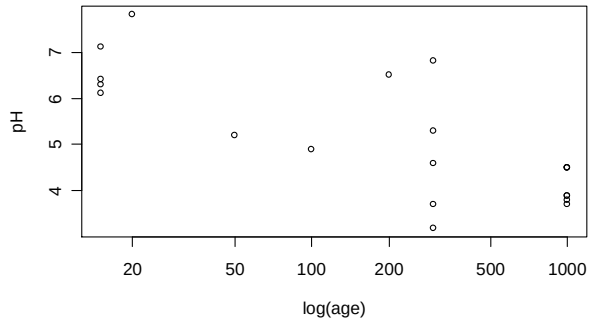




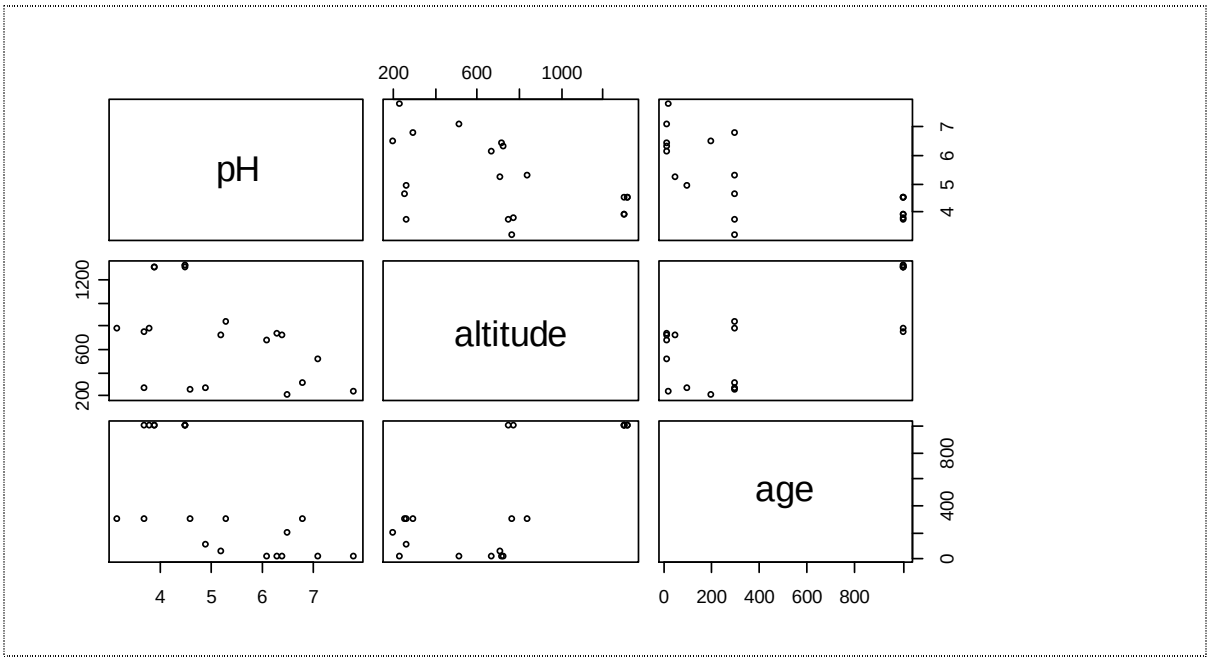

```
plot(rep(0.5,26),pch=0:25,axes=F,ann=F,frame.plot=T)#různé typy bodů
```

```
# cvičná data pro xy graf
    pH altitude  age
1  3.8      780 1000
2  3.7      750 1000
3  3.2      770  300
4  3.9     1300 1000
5  3.9     1300 1000
6  5.3      840  300
7  4.5     1315 1000
8  4.5     1300 1000
9  5.2      710   50
10 6.1      670   15
11 7.1      515   15
12 6.3      725   15
13 6.5      200  200
14 4.6      255  300
15 4.9      265  100
16 6.8      300  300
17 3.7      265  300
18 4.5     1320 1000
19 6.4      720   15
20 7.8      230   20
```






```
plot(x$altitude,x$pH,xlab="altitude",ylab="pH")#xy graf
plot(x$age,x$pH,xlab="log(age)",ylab="pH",log="x")#xy graf s logaritmovanou osou
x
plot(x)#vykreslí xy závislosti všech proměnných v data.frame
```



Kontrolní úkoly

1. Do vektoru x načtěte následující hodnoty: 1,1,2,3,4,5,4,4,1,1,2,2,5,4,6,5,3. Vytvořte jednoduchý graf hodnot.
2. Převeďte na další typy grafů – spojnicový, horizontální linie. Změňte typ na spojnicový s body, měřítko osy x na 0-20, osy y na 0-10 a barvu linií na zelenou.
3. Změňte čáru na silnější, tečkovanou a modrou.
4. Změňte typ grafu na bodový, body nastavte na typ kruh s výplní a barvu pozadí bodu změňte na červenou.
5. Převeďte data na faktor a vyneste do grafu. Vytvořte graf tak, aby byly první tři sloupce červené a další modré.
6. Použijte tutéž funkci ale zadejte argumenty tak, aby se nevykreslil graf, ale vypsaly jeho vlastnosti. Pokuste se odhadnout, co se vypsalo?
7. Vykreslete funkci log(x) v intervalu 0-10 na ose x a funkci abs(-2x+1) v intervalu -2 až 2 na ose x.

Parametry pro nastavení grafu

Funkce par slouží k nastavení základních parametrů grafu při otevření grafického okna, tedy ještě před vykreslením grafu. Umožňuje jednotnou podobu vykreslování grafů, specifická nastavení, která nelze zadat ve funkci plot, dělení obrazovky na více částí atp. Nastavení funguje do okamžiku zavření grafického okna, pak se opět načítají standardní hodnoty. Je velice důležité vědět, že většinu parametrů můžeme použít také jako argumenty ostatních grafických funkcí (např. `srt` pro rotaci textu můžeme použít ve funkci `text`, `las` ve funkci `plot` atd.).

- **`par`**`(parameter=value,…)` – nastaví parametry grafiky na hodnoty `value`

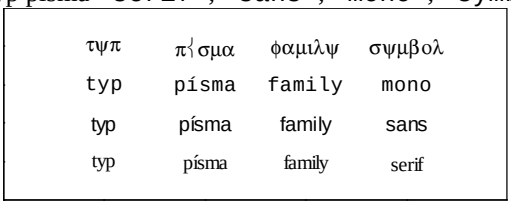
  textové

  `adj` – číslo, určuje, jak je zarovnaný textový řetězec (0 zleva, 0.5 uprostřed, 1 vpravo). Povoleno vše z intervalu [0, 1], pro funkci `text` povoleno také `adj = c(x, y)` pro různé zarovnání ve směru horizontálním a vertikálním.

  `ann` – logická, výpis anotací (název, popisky os) viz plot.

  `cin` – pouze čtení; velikost aktuálních znaků (šířka, výška) v palcích.

cex – relativní velikost znaků (1 – normální velikost, 2 – dvojnásobná).
cex.main, cex.sub – relativní velkost znaků pro název grafu, pro podtitulek.
col.main, col.sub – barva pro název grafu, pro podtitulek.
cra – pouze čtení; velikost aktuálních znaků (šířka, výška) v rastru (pixelech).
crt – ve stupních vyjádřená rotace jednotlivých znaků (doporučené jsou násobky 90).
csi – pouze čtení; velikost aktuálních znaků (šířka, výška) v palcích.
cxy – pouze čtení; velikost aktuálních znaků (šířka, výška) v uživatelských jednotkách `par("cin")/par("pin")`. Obvykle je mnohem přesnější `c(strwidth(ch), strwidth(ch))` pro řetězec `ch`.
family – typ písma `"serif"`, `"sans"`, `"mono"`, `"symbol"` a Hershey fonty.



font – typ písma pro text: `1` (normální), `2` (tučné), `3` (kurzíva), `4` (tučná kurzíva).
font.main, font.sub – typ písma pro název grafu, pro podtitulek.
lheight – výška řádku textu, standardně 1, nastavené hodnoty jsou násobky.
ps – velkost textu a symbolů (jednotky jsou pixely – body).
srt – rotace řetězců v úhlech.
tmag – poměrné zvětšení velikosti hlavního názvu grafu k ostatnímu textu.

graf
ask – uživatel je dotázán na vstup (nutnost kliknout před vykreslením).
bg – barva pozadí grafu, pro points a plot také barva výplně bodů `pch`>20.
gamma – specifikace barev na výstupním zařízení, tzv. gamma correction (viz `?par`).
bty – typ ohraničení kolem grafu `"o"` (okolo), `"l"` (levý a dolní), `"7"` (pravý a horní), `"c"` (horní, dolní, levý), `"u"` (levý, pravý, dolní), `"]"` (dolní, horní, pravý), `"n"` (potlačí ohraničení).
din – pouze čtení; rozměry výstupního zařízení (jednotky, okna) v palcích.
fig – numerický vektor `c(x1, x2, y1, y2)`, který nastavuje souřadnice grafické oblasti v daném zobrazení (lze měnit i po vykreslení grafu, ale je nepraktické).
fin – numerický vektor, udává rozměry grafické oblasti v palcích `c(x, y)`.
mai – numerický vektor nastavující okraje vykreslovací plochy v palcích. `c(dolní,levý,horní,pravý)`, standardně `mar=c(0.96,0.77,0.77,0.39)`.
mar – numerický vektor nastavující počet řádků na okrajích vykreslované plochy, jako mai standardně `mar=c(5.1, 4.1, 4.1, 2.1)` .
mex – faktor rozšíření nebo zúžení okrajů grafu, velikost textu ponechá stejný.
mfcol, *mfrow* – vektor rozdělující na určitý počet podobrázků `c(řádků, sloupců)` . Rozdíl je v postupném vypisování (po sloupcích nebo po řádcích (alternativy `layout, split.screen`).
mfg – numerický vektor, který vypisuje který z obrázků se bude vykreslovat (popř. momentálně vykresluje) .
usr – vektor, kterým můžeme číst nebo nastavit extrémy koordinát vykreslované oblasti `c(x1,x2,y1,y2)`.
oma – vektor `c(dolní,levý,horní,pravý)` udávající velikost vnějších "textových" okrajů (také okrajů v oknech s více grafy).
omd – vektor udávající vnější "textový" okraj `c(dolní,levý,horní,pravý)` v jednotkách NDC (frakce celkového grafického okna).
omi – vektor `c(dolní,levý,horní,pravý)` udávající velikost vnějších okrajů v palcích
pin – šířka a výška aktuálního grafu v palcích.
plt – vektor `c(x1, x2, y1, y2)` souřadnic grafické oblasti (plotting region) jako proporce k celkové oblasti vykreslování (figure region) .
pty – specifikuje typ grafické oblasti `"s"` čtvercová, `"m"` maximální (může být i obdélník)
xpd – logická hodnota nebo NA. F – vykreslování vztaženo na grafickou oblast, T – oblast vykreslování na oblast grafického zařízení, tzn. dovoluje kreslit i mimo plochu grafu.

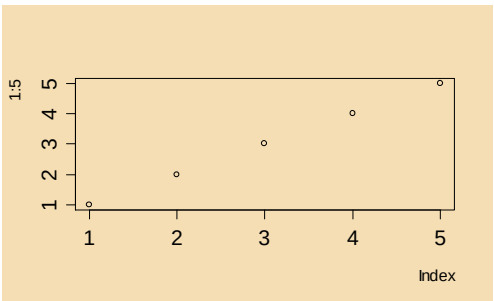
osy

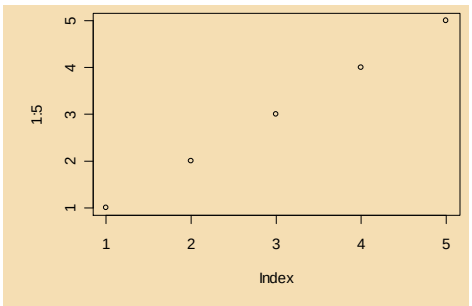
cex.axis, *cex.lab* – velikost písma pro popisky os, pro názvy osy.
col.axis, *col.lab* – barva pro popisky os, pro názvy osy.
fg – barva pro osy a orámování (ohraničení) .
font.axis, *font.lab* – *font* pro popisky os, pro názvy os.
las – numerická hodnota otočení popisků os: `0` – y svisle a x vodorovně, `1` – xy vodorovně, `2` – x svisle a y vodorovně, `3` – xy svisle.
lab – numerický vektor `c(x, y, len)` určující přibližné počty značek osy x a y a velikost popisků (len není implementováno), nastavuje se před vykreslením os.
mgp – vektor `c(název, popisky, osa)`, okraj (v měřítku mex) pro odsazení názvu osy, popisků osy a vlastní osy od ohraničení grafu.
tck – velikost značek na osách jako proporce šířky nebo výšky grafické oblasti (`tck=1` vykreslí mřížku) .
tcl – velikost (výška) značek na osách jako proporce velikosti řádku textu.
xaxp, *yaxp* – vektor `c(x1, x2, n)` – min a max pro značku na ose a počet intervalů mezi značkami. Pro logaritmickou osu má jiný význam, viz `?axTicks`.
xaxs, *yaxs* – styl osy (`"r"` = odsadí o 4 % data, `"i"` = přesné nasazení k ohraničení
xaxt, *yaxt* – typ osy: `"n"` = nastavena, ale nezobrazena, `"s"`, `"l"`, `"e"` = zobrazena.
xlog, *ylog* – logická hodnota, (`T` = logaritmické měřítko na ose, `F` = normální měřítko) .

řady

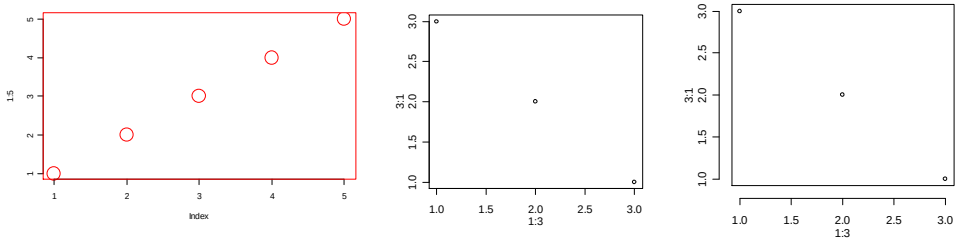
col – barva bodů, popř. ohraničení.
lend – ukončení čáry: `0` nebo `"round"`; `1` nebo `"butt"`; `2` nebo `"square"`.
ljoin – hrana spoje dvou čar `0` – `"round"`; `1` – `"mitre"`; `2` – `"bevel"`.
lmitre – limit zkosení při spojování čar od `1` standardně `10`.
lty – typ čáry, viz plot.
lwd – šířka čáry, standardně 1, viz plot.
pch – znak pro vykreslování bodů grafu, viz plot.

```
par("mai") #zjištění hodnoty mai (okraje)
[1] 0.95625 0.76875 0.76875 0.39375

#před vykreslením dotaz, pozadí, zvětšení os, zarovnání os doprava
par(ask=T,bg="wheat",cex.axis=1.3,adj=1)
plot(1:5)
Waiting to confirm page change...

par(mai=c(1,1,0.1,0.1), bg="wheat")#změna okrajů
plot(1:5)
```

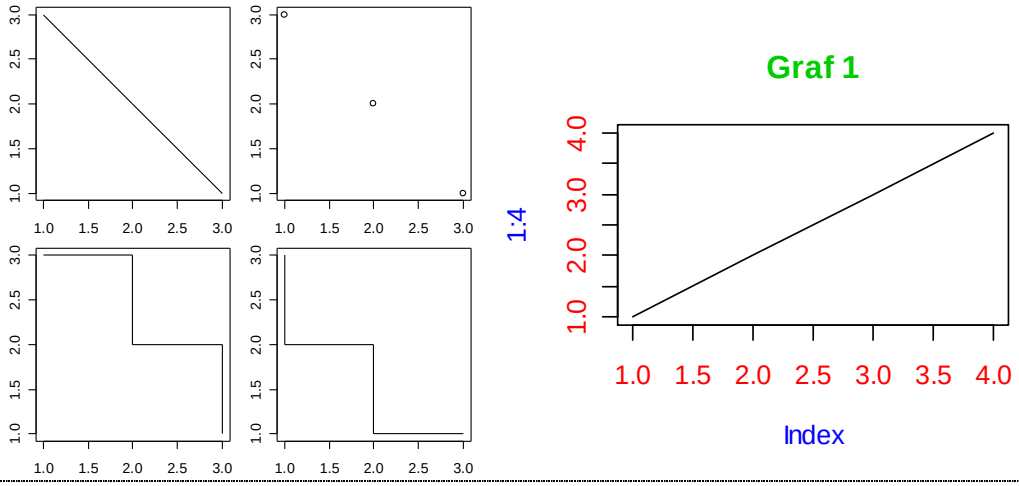





```
par(mai=c(1,1,0.1,0.1), cex=0.7,col=2)# okraje,velikost znaků, barva
plot(1:5,cex=4) #graf vlevo
par(mai=c(1,1,0.1,0.1),mgp=c(2,1,0))#okraje, vzdálenosti osy, popisků a názvu
plot(1:3,3:1) #graf uprostřed
par(mai=c(1,1,0.1,0.1),mgp=c(3,2,1)) #jako přechozí, jen posunuto
plot(1:3,3:1) #graf vpravdo
```



```
#vykreslí 4 grafy v jednom grafickém okně s nastavením mezer mezi nimi
par(mfrow=c(2,2), omi=c(0,0,0,0), mai=c(0.35,0.35,0.1,0.1))
plot(c(3,2,1),type="l");plot(c(3,2,1),type="p");plot(c(3,2,1),type="s");
plot(c(3,2,1),type="S")
#graf vpravo s různým nastavením barev (parametry jsou použity jako argumenty)
plot(1:4,type="l",main="Graf 1",col.main=3,col.axis=2,col.lab=4)
```



Kontrolní úkoly

8. Vygenerujte 30 náhodných čísel z Poissonova rozdělení (průměr je 5) do objektu x1. Vytvořte bodový graf.
9. Nastavte parametry nejprve přímo jako argumenty funkce plot a následně jako argumenty funkce par tak, aby osa y měla hodnoty otočené horizontálně, osa x i y je bez odsazení a popisky jsou jen 90% velikosti.
10. Aniž zavřete grafické okno z předchozího úkolu vytvořte nový graf s hodnotami 1,2,4,2,1,5,7 bez nastavení dalších argumentů. Pak grafické okno zavřete a znovu opakujte vykreslení grafu. Jaké jsou rozdíly a proč.
11. Nastavte okraje a vykreslete graf z předchozího úkolu tak, aby u dolního okraje byl odsazen 4 řádky, u levého okraje 4 řádky, u horního okraje 1 řádek a u pravého 1 řádek.
12. Vykreslete v jednom grafickém okně 4 grafy se stejnými libovolnými hodnotami. Jeden graf musí být bodový, jeden liniový, jeden svislé linie a jeden linie s body.
13. Vykreslete xy bodový graf, kde souřadnice bodů jsou následující:

	x	y
1	12	5
2	2	3
3	4	8
4	10	11
5	5	7
6	7	8

Další důležité grafy

Než se budeme naplno věnovat grafům, je ještě nutné vysvětlit práci se vzorci (formula, viz základní typy objektů). U velké části grafů máme dvě možnosti jak zadávat data. Buď jsou jednotlivé hodnoty pro různé typy měření rozděleny do sloupců nebo jeden sloupec charakterizuje typ položky a druhý obsahuje vlastní hodnoty. Jako příklad můžeme uvést měření hladiny určité látky u dvou typů pacientů. Dejme tomu, že máme stejný počet měření (5), pak můžeme mít hodnoty ve dvou sloupcích (pac1 a pac2). Druhá možnost je vytvořit sloupec např. "typ", do kterého uvádíme typ každého pacienta (pac1, pac2) a do dalšího sloupce naměřenou hladinu látky. V prvním případě máme 5 řádků, ve druhém případě 10 řádků. Mnohem lepší je druhý zápis, protože nemusíme mít stejný počet měření pro obě skupiny a chceme-li každého pacienta specifikovat ještě dále, pak stačí přidat sloupec (což v prvním případě je mnohem obtížnější).

styl 1

```
  pac1 pac2
1 10.7  8.6
2 11.2  9.9
3  9.7  7.6
4 10.8  9.1
5  9.6  8.1
```

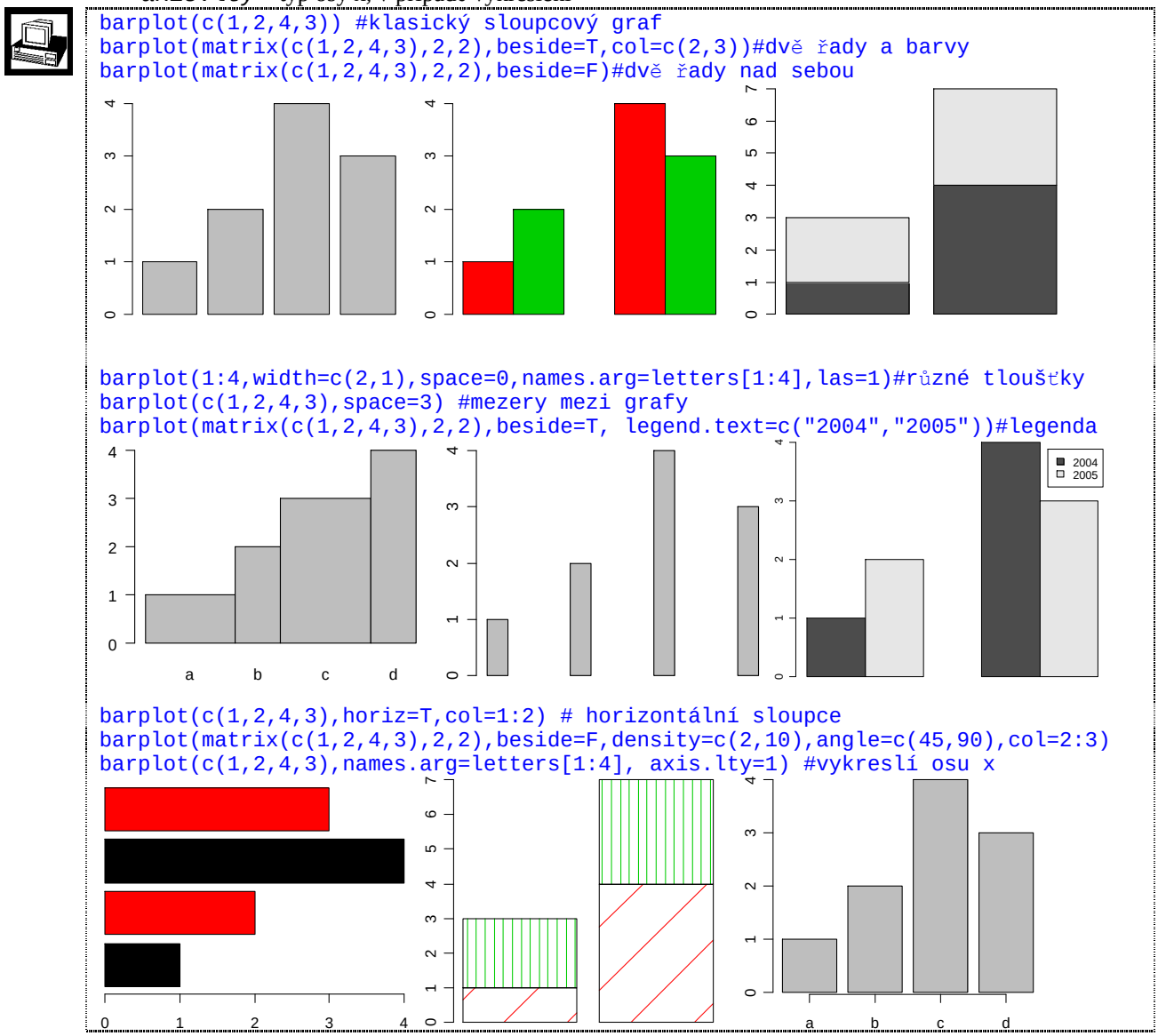
styl 2

```
    typ hodnota
1  pac1    10.7
2  pac1    11.2
3  pac1     9.7
4  pac1    10.8
5  pac1     9.6
6  pac2     8.6
7  pac2     9.9
8  pac2     7.6
9  pac2     9.1
10 pac2     8.1
```

U velké částí grafů, kde chceme jednotlivé skupiny odlišit (např. boxploty atd.) lze použít obě varianty. První typ zápisu vkládáme jako objekt (tabulku dat nebo matici), druhý typ je nutné zapsat vzorcem (formula) hodnota~typ (popř. další charakteristiky s připojují většinou pomocí +) , data = název tabulky dat.

- **barplot**(height,width,space,names.arg,legend.text,beside=F,horiz=F, density,angle,col,border,main,sub,xlab,ylab,xlim,ylim,xpd=T,axes=T, axisnames=T,cex.axis,cex.names,inside=T,plot=T,axis.lty,offset,…) – vytvoří klasický sloupcový graf.
  height– vektor nebo matice dat
  beside – vykreslení více řad vedle sebe (T), nad sebou (F) – data v height jsou zapsány formou matice
  width – hodnota (vektor) šířky sloupců (pro jednu hodnotu je nutné nastavit xlim)
  space – vektor nebo číslo: velikost mezery mezi sloupci nebo jednotlivých mezer, u maticových dat a beside =T specifikujeme dvě čísla – mezi skupinami a mezi jednotlivými sloupci
  names.arg – vektor názvů sloupců osy *x*
  legend.text – v případě více datových řad uvádí názvy v legendě.
  horiz – nastavuje horizontální nebo vertikální vykreslování sloupců.
  density – nastavuje šířku šrafování sloupců (vektor nebo hodnota).
  angle – sklon šrafování.
  col – barva šrafování nebo dalších komponent nebo barva sloupců.
  border – barva okraje sloupců.
  main, sub, xlab, ylab, xlim, ylim, cex.axis, cex.names – viz par a plot.
  xpd – je dovoleno sloupcům přesahovat oblast grafu?
  axes, *axisnames* – mají být vykreslovány osy a názvy os?

inside – čáry mezi sloupci
plot – vykreslovat nebo nevykreslovat graf
offset – posunutí grafu na ose y o určitý počet
axis.lty – typ osy x, v případě vykreslení



```
barplot(c(1,2,4,3)) #klasický sloupcový graf
barplot(matrix(c(1,2,4,3),2,2),beside=T,col=c(2,3))#dvě řady a barvy
barplot(matrix(c(1,2,4,3),2,2),beside=F)#dvě řady nad sebou
```

```
barplot(1:4,width=c(2,1),space=0,names.arg=letters[1:4],las=1)#různé tloušťky
barplot(c(1,2,4,3),space=3) #mezery mezi grafy
barplot(matrix(c(1,2,4,3),2,2),beside=T, legend.text=c("2004","2005"))#legenda
```



```
barplot(c(1,2,4,3),horiz=T,col=1:2) # horizontální sloupce
barplot(matrix(c(1,2,4,3),2,2),beside=F,density=c(2,10),angle=c(45,90),col=2:3)
barplot(c(1,2,4,3),names.arg=letters[1:4], axis.lty=1) #vykreslí osu x
```

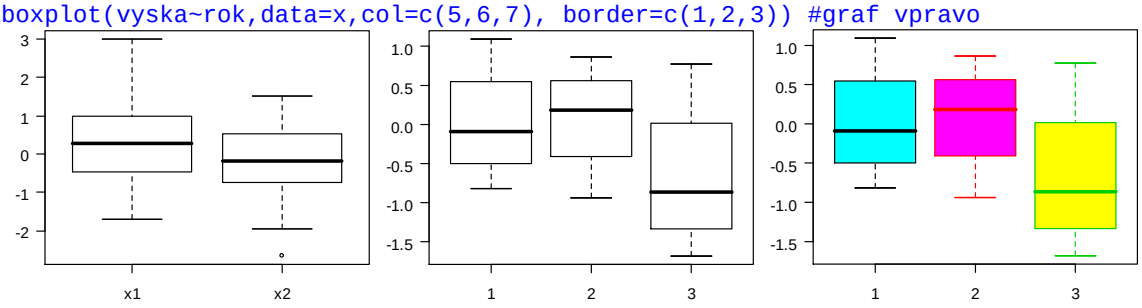

- **boxplot**(`formula,data,subset,…,range=1.5,width,varwidth=F,notch=F,outline=T,names,plot=T,border,col,log,pars,horizontal=F,add=F,at`) – vytvoří klasický krabicový graf podle vzorce formula z tabulky `data` a podmnožiny `subset`. Boxplot tvoří medián, horní a dolní kvartil (krabice), minimální a maximální proměnná ležící v dolní a horní "hradbě" (1.5 × mezikvartilové rozpětí), za hradbou jsou data zobrazovány jako odlehlé hodnoty (kolečka).
  formula – vzorec závislosti, tzn. závislá proměnná ~ nezávislá proměnná, místo zadání lze ale uvádět sloupce tabulky dat.
  range – násobeno IQR (mezikvartilovým) rozpětím udává rozmezí „vousů“, norm. 1.5 (0 udává rozpětí až po extrémy).
  width – relativní velikost „krabic“ ve vzájemném poměru.
  varwidth – proporční velikost krabic vzhledem k odmocnině počtu dat v každé ze skupin.
  notch – logical, zobrazuje výřezy počítané jako `±1.58*IQR/sqrt(n)`, kde n je počet pozorování, jestliže se výřezy dvou krabic nepřekrývají, lze dost průkazně odhadovat, že se mediány liší.
  outline – logical, mají se zobrazovat odlehlé hodnoty?

names – názvy jednotlivých „krabic“ na ose x.
border, *col*, *log*, *plot* – viz předchozí typy grafů.
horizontal – logical, vykreslovat krabice horizontálně?
add – logical, přidat graf do již nakresleného grafu.
at – numerický vektor popisující, kde vykreslit boxploty (zejména) v případě `add = T`)
na.action – jak nakládat s hodnotami NA
pars
- *boxwex*: změna velikosti "krabic".
- *staplewex*, *outwex*: změna velikosti čar "hradby" a "extrémů".
- *boxlty*, *boxlwd*, *boxcol*, *boxfill*: typ čáry, šířka, barva a výplň krabic.
- *medlty*, *medlwd*, *medpch*, *medcex*, *medcol*, *medbg*: typ čáry, šířka, typ bodu, velikost bodu, barva a výplň znaku pro medián nebo čáry mediánu (lty="blank", outpch=NA vypínají jednotlivé typy).
- *whisklty*, *whisklwd*, *whiskcol*: typ čáry, šířka a barva "vousů".
- *staplelty*, *staplelwd*, *staplecol*: typ čáry, šířka a barva čar hradby.
- *outlty*, *outlwd*, *outpch*, *outcex*, *outcol*, *outbg*: typ čáry, šířka, typ bodu, velikost bodu, barva a výplň znaku pro outliers (lty="blank", outpch=NA vypínají jednotlivé typy).

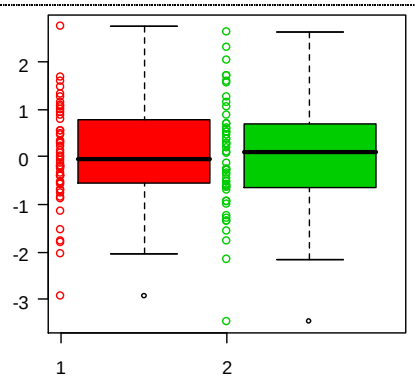
výstupy

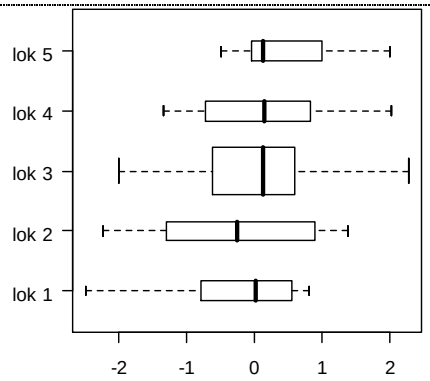
$stats – pro každou skupinu je uveden výpočet následujících statistik: dolní vous, dolní část krabice, medián, horní fous, horní část krabice.
$n – počet pozorování v každé skupině.
$conf – horní a dolní extrémy výřezů (notch, konfidenční intervaly)
$out – odlehlé hodnoty
$group – do které skupiny odlehlá hodnota patří
$names – názvy skupin

```
par(mar=c(2.1,2,0.1,0.1),las=1)#nastavení parametrů
boxplot(data.frame(x1=rnorm(40),x2=rnorm(40))) #graf vlevo, zobrazení náh. čísel
x<-cbind(rok=factor(rep(1:3,4)),vyska=rnorm(12))#tvorba tabulky pro 2 grafy
boxplot(vyska~rok,data=x) #graf uprostřed
boxplot(vyska~rok,data=x,col=c(5,6,7), border=c(1,2,3)) #graf vpravo
```

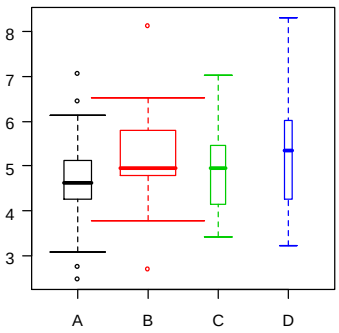


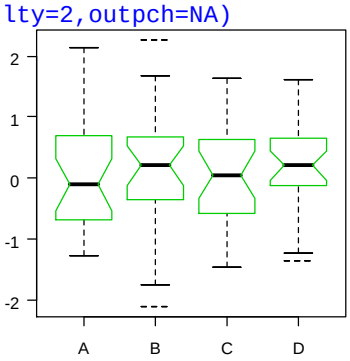

```
#boxplot vepsaný do normálního xy grafu (obr. vlevo)
x1<-data.frame(x=rep(c(1,2),each=50),y=rnorm(100))#vygenerována data
plot(x1$y~x1$x,xlim=c(1,3),xaxp=c(1,2,1),pch=21,col=rep(2:3,each=50))# x,y graf
boxplot(y~x,data=x1,at=c(1.5,2.5),axes=F,add=T,col=2:3)

#horizontální graf s proporcemi varwidth (na lok 3 60 dat, na ostatních 10)
par(mar=c(2.1,3.3,0.1,0.1),las=1,cex=0.8)
x1<-data.frame(x=rep(paste("lok",1:5),times=c(10,10,60,10,10)),y=rnorm(100))
boxplot(y~x,data=x1,varwidth=T,horizontal=T)
```






```
x1<-data.frame(x=rep(LETTERS[1:4],times=25),y=rnorm(100)+5)
boxplot(y~x,data=x1,width=c(1,2,0.5,0.3),border=1:5,staplewex=2)

x1<-data.frame(x=rep(LETTERS[1:4],times=25),y=rnorm(100))
boxplot(y~x,data=x1,notch=T,boxcol=3,outlty=2,outpch=NA)
```






```
boxplot(y~x,data=x1,plot=F)# výpis výstupů (viz výstupy)
$stats
           [,1]       [,2]        [,3]       [,4]
[1,] -1.2602766 -1.7488486 -1.46574430 -1.2298423
[2,] -0.6830814 -0.3536459 -0.58231448 -0.1139109
[3,] -0.1079411  0.2109140  0.05878168  0.2090689
[4,]  0.6929362  0.6650194  0.63176855  0.6464986
[5,]  2.1347571  1.6851194  1.62731758  1.6190020

$n
[1] 25 25 25 25

$conf
           [,1]       [,2]       [,3]        [,4]
[1,] -0.5427626 -0.1109842 -0.3248686 -0.03122051
[2,]  0.3268805  0.5328122  0.4424319  0.44935829

$out
[1]  2.257766 -2.102357 -1.360767

$group
[1] 2 2 4

$names
[1] "A" "B" "C" "D"
```

- **stem**(x,scale=1,width=80,atom=1e-08) – vytvoří diagram leaf-and-stem (stonek s listy) graficky znázorňující četnosti jednotlivých hodnot (podobně jako histogram).
  scale – stupeň škálování
  atom – tolerance rozlišování hodnot
  width – určuje šířku grafu

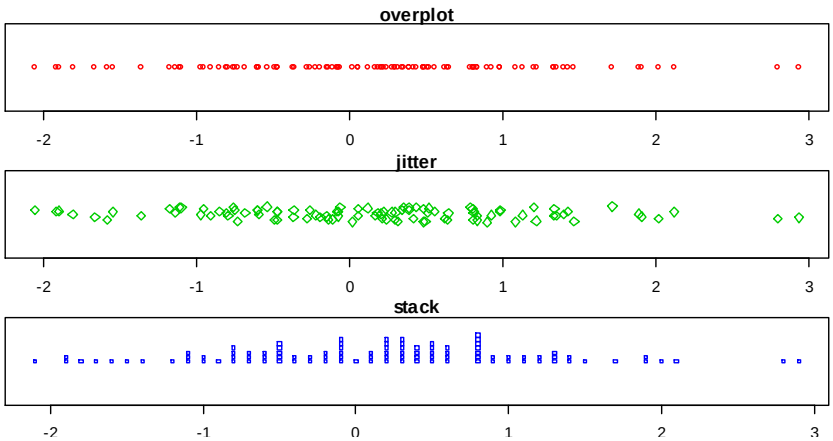
```
x<-c(1,1,1,1,2,2,2,2,2,2,2,3,3,3,3,3,3,4,4,4,4,5,5,5,6)
stem(x) #škála leaf and stem od 0-1,2-3,4-5,6-7, nula nebo jednička 4krát ...
  The decimal point is at the |
```

```
  0 | 0000
  2 | 0000000000000
  4 | 0000000
  6 | 0

stem(x,scale=2) # rozšíří diagram o další třídy

  The decimal point is at the |

  1 | 0000
  2 | 0000000
  3 | 000000
  4 | 0000
  5 | 000
  6 | 0

stem(x,scale=3) # rozšíří diagram o intervaly po 0.5

  The decimal point is at the |

  1 | 0000
  1 |
  2 | 0000000
  2 |
  3 | 000000
  3 |
  4 | 0000
  4 |
  5 | 000
  5 |
  6 | 0

x<-rnorm(40)
stem(x) #diagram pro čísla s desetinnými místy (-2.4,-1.5,…,1.2,2.7)

  The decimal point is at the |

  -2 | 4
  -1 | 5211000
  -0 | 99987666442211
   0 | 1122223344457799
   1 | 2
   2 | 7

stem(x,scale=2) #rozlišení po 0.5

  The decimal point is at the |

  -2 | 4
  -1 | 5
  -1 | 211000
  -0 | 99987666
  -0 | 442211
   0 | 11222233444
   0 | 57799
   1 | 2
   1 |
   2 |
   2 | 7
```

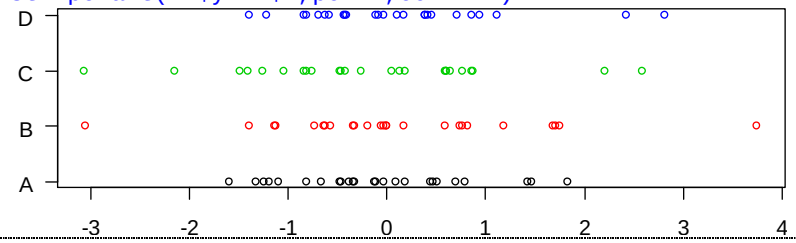
- **`stripchart`**`(x,method,jitter,offset,vertical=F,group.names,add=F,at,xlim, ylim,main,ylab,xlab,log,pch,col,cex)` – vykresluje diagram rozptýlení, rozmítnutý diagram rozptýlení a uspořádaný rozmítnutý diagram rozptýlení (viz obrázky). Používá se pro základní jednoduché zobrazení dat.
  `method` – `"overplot"` na ose, `"jitter"` náhodně kolem osy, `"stack"` naskládáno na sobě

jitter – rozsah rozložení kolem osy
offset – rozestup při metodě stack
vertical – vykresluje vertikálně
group.names – názvy skupin, které se mají vykreslovat po bocích grafů (v případě více grafů)
add – přidat graf do již vykresleného
at – ve které pozici (y) se má začít vykreslovat

```
x<-rnorm(100)
par(las=1,mfrow=c(3,1),omi=c(0,0,0,0),mar=c(2,0.1,1.5,0.1))
stripchart(x,method="overplot",pch=1,col=2,cex=1,main="overplot")
stripchart(x,method="jitter",pch=5,col=3,cex=1,main="jitter")
stripchart(round(x,1),method="stack",pch=0,col=4,cex=0.5,main="stack")
```

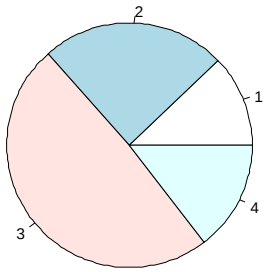


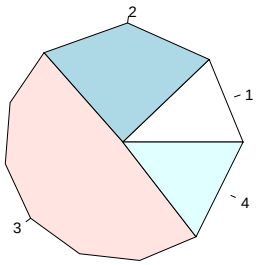

```
x1<-data.frame(x=rep(LETTERS[1:4],times=25),y=rnorm(100))
par(las=1,mar=c(2,2,0.1,0.1))
stripchart(x1$y~x1$x,pch=1,col=1:4)
```

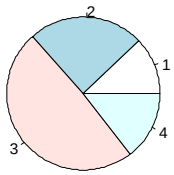


- **pie**(x,labels=names(x),edges=200,radius=0.8,clockwise=F,init.angle, density=NULL, angle=45, col=NULL, border=NULL, lty=NULL, main=NULL,…) – vytvoří klasický koláčový graf s různými možnostmi dalšího nastavení
  labels – označuje popisky u grafu
  edges – místo kružnice se vytvoří mnohoúhelník s udaným počtem stran
  radius – velikost grafu od -1 do 1 (záporná čísla – graf je převrácený)
  density – hustota šrafování jednotlivých dílů "koláče"
  angle – úhel šrafování jednotlivých dílů
  clockwise – vykreslování ve směru nebo proti směru hodinových ručiček
  init.angle – počáteční úhel (natočení grafu)
  col – barvy šrafování nebo výplně jednotlivých dílů
  border – vektor barev okraje jednotlivých dílů
  lty – vektor typu čáry tvořící okraje jednotlivých dílů

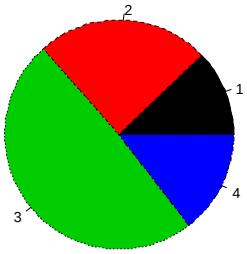
```
pie(c(5,10,20,6))
pie(c(5,10,20,6),edges=16)#mnohoúhelník
pie(c(5,10,20,6),radius=0.5)#graf v okně zmenšen na polovinu
```

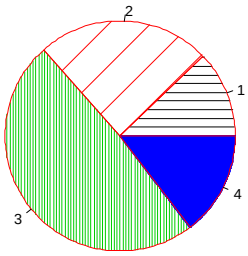


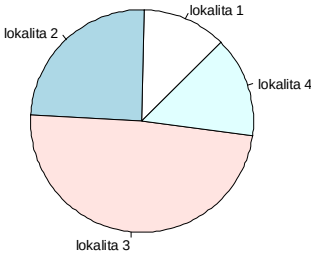





```
pie(c(5,10,20,6),col=1:4,lty=1:4)#změna barev a typů čar ohraničení
pie(c(5,10,20,6),density=c(10,3,30,1000),angle=c(0,45,90,0),col=1:4,border=2)
pie(c(5,10,20,6),init.angle=45,labels=paste("lokalita",1:4))#pootočeno, popisky
```







- **hist**(x,breaks,freq=T,probability=!freq,include.lowest=T,right=T, density,angle,col,border,main,xlim,ylim,xlab,ylab,axes=T,plot=T, labels=F,nclass,…) – vytváří histogram četnosti dané variační řady (číselné)

  breaks – nastavuje třídy četnosti několika způsoby:
  - *typ vytváření* – zadán algoritmus vytváření intervalů "Sturges", "Scott", "FD"
  - *počet tříd četnosti* – počet tříd, číslo, udávající přibližný počet tříd
  - *interval tříd četnosti* – vektor, udávající hranice intervalů
  - *funkce* – funkce pro výpočet počtu tříd

  freq (= opak *probability)* – T – absolutní, F – relativní četnost

  include.lowest – logická, zda do nejnižšího intervalu je zahrnuta také hodnota minima, pozor, automaticky nastaveno na T, tzn. 1. třída u celých čísel bude zahrnovat první dvě hodnoty

  right – pro right=T jsou intervaly uzavřené zprava a zleva, jinak naopak

  angle, col, border, main, … – shodné s předchozími grafy

  výstupy

  breaks – hranice tříd.

  counts – n hodnot s absolutními četnostmi pro intervaly.

  density – hustota pravděpodobnosti pro jednotlivé třídy, jestliže jsou intervaly = 1 pak relativní četnosti.

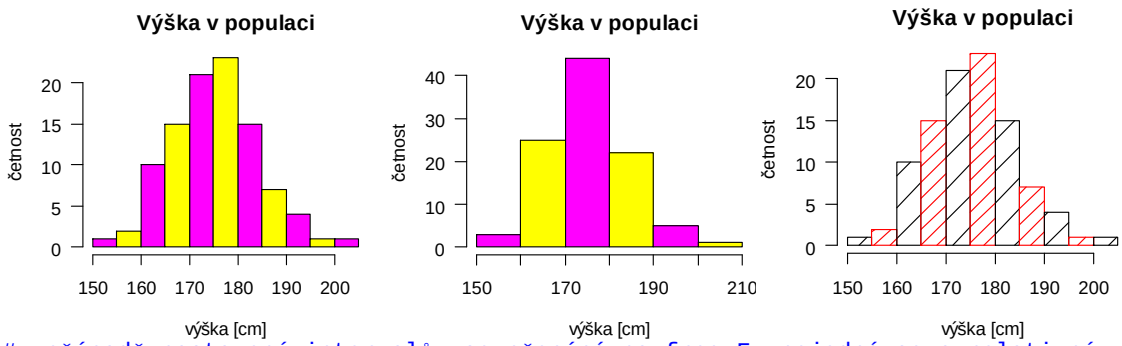
  intensities – shodné s densitou.

  mids – středy tříd.

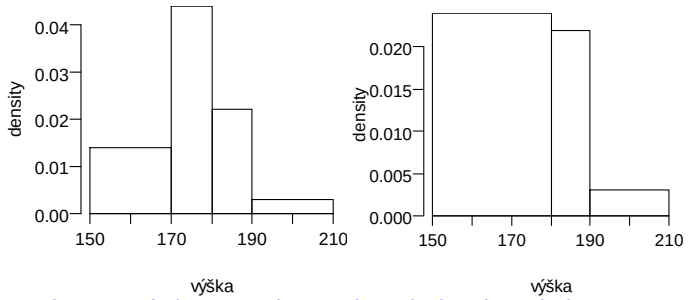
  xname – objekt v argumentu x.

  equidist – logická hodnota indikující, zda je shodná velikost tříd.

```
x<-rnorm(100,175,10) #vygenerováno 100 hodnot výšky, průměr 175, sm. odch. 10
par(mar=c(4.2,4,3,0.1),las=1)
hist(x,col=6:7,main="Výška v populaci",xlab="výška [cm]",ylab="četnost")
hist(x,col=6:7,breaks=5,main="Výška v populaci",xlab="výška [cm]",ylab="četnost")
hist(x,co=1:2,m="Výška v populaci",xla="výška
[cm]",yla="četnost",density=c(5,10))
```




```
#v případě nastavení intervalů, se přepíná na freq=F, nejedná se o relativní
#frekvenci, ale densitu, tzn. plocha sloupce je relativní četnost
par(mar=c(4.2,4,0.3,0.2),yaxs="i",las=1,mgp=c(3,0.6,0))
hist(x,breaks=c(150,170,180,190,210),m=NULL,xlab="výška",ylab="density")
hist(x,breaks=c(150,180,190,210),m=NULL,xlab="výška",ylab="density")
```



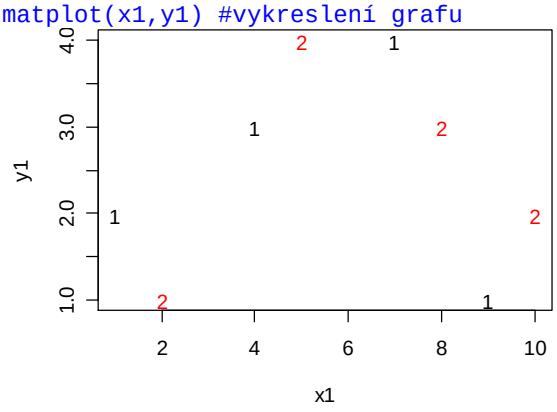

```
#práce s right a zahrnutím minim (maxim)
hist(c(1,1,2,2,2,3,3,3,3,3,4,4,5),m=NULL,xlab="hodnoty")#frekvence 1 a 2
dohromady
hist(c(1,1,2,2,2,3,3,3,3,3,4,4,5),br=0:5,m=NULL,xlab="hodnoty")
hist(c(1,1,2,2,2,3,3,3,3,3,4,4,5),breaks=1:6,m=NULL,xlab="hodnoty",right=F)
```


```
x<-rnorm(100,175,10)
hist(x,plot=F)
$breaks
 [1] 150 155 160 165 170 175 180 185 190 195

$counts
[1]  1  2 11 15 20 21 21  7  2

$intensities
[1] 0.002000000 0.004000000 0.022000000 0.030000000 0.040000000 0.042000000
[7] 0.042000000 0.014000000 0.004000000

$density
[1] 0.002000000 0.004000000 0.022000000 0.030000000 0.040000000 0.042000000
[7] 0.042000000 0.014000000 0.004000000

$mids
[1] 152.5 157.5 162.5 167.5 172.5 177.5 182.5 187.5 192.5

$xname
```

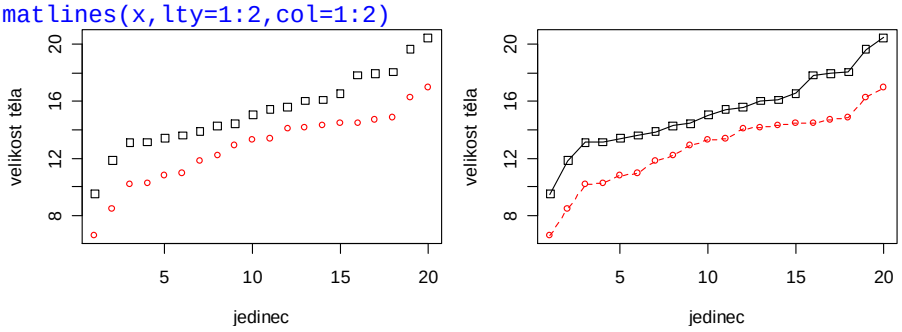
```
[1] "x"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
```

- **matplot**(x,y,type,lty,lwd,pch,col,cex,xlab,lab,xlim,ylim,…,add=F,verbose) – vynáší více řad (viz obr Obr 10) do jednoho grafu, aniž bychom museli použít pro přidání funkce points nebo lines. Souřadnice pro body jsou zadány ve sloupcích matic, zvlášť pro x a pro y. **matpoints** a **matline** jsou podobné jako matplot, ale body nebo přímky se přidávají do stávajícího grafu.

```
x1<-matrix(c(1,4,7,9,2,5,8,10),4,2);x1 # x-ové souřadnice pro 1. a 2. řadu
     [,1] [,2]
[1,]    1    2
[2,]    4    5
[3,]    7    8
[4,]    9   10
y1<-matrix(c(2,3,4,1,1,4,3,2),4,2);y1 # y-ové souřadnice pro 1. a 2. řadu

     [,1] [,2]
[1,]    2    1
[2,]    3    4
[3,]    4    3
[4,]    1    2
par(mar=c(4.2,4.2,0.1,0.1)) #nastavení okrajů grafu
matplot(x1,y1) #vykreslení grafu
```




```
# seřazené hodnoty náhodně vygenerovaných velikostí samic a samců určitého druhu
x<-data.frame(samci=sort(rnorm(20,15,2)),samice<-sort(rnorm(20,14,3)))
matplot(x,pch=0:1,xlab="jedinec",ylab="velikost  těla")
matlines(x,lty=1:2,col=1:2)
```



Shrnutí:
Metody vytváření grafů v R jsou trojího typu – tvorba high-level (vysokoúrovňové), low-level (nízkoúrovňové) a interaktivní grafiky. Základní tvorba klasických high-level grafů probíhá pomocí funkce plot, která pro různé typy objektů vytváří různé typy grafů.

Specifické vlastnosti grafu nastavujeme pomocí argumentů této funkce nebo pomocí nastavení parametrů grafického okna (funkce par). Grafy typu histogram, boxplot, sloupcový nebo koláčový graf vytvoříme pomocí dalších uvedených funkcí. U některých grafů mohou data vstupovat ve dvou typech formátu. Naměřené hodnoty pro každou hodnotu znaku v samostatných sloupcích nebo jeden sloupec obsahuje znak pro třídění a další naměřené hodnoty. Ve druhém případě musíme data do grafu zadávat v podobě vzorce (formula).

Metody k zapamatování:

- Základní graf: plot, argumenty – xlim, ylim, main, xlab, ylab, col, pch, bg, lty, lwd.
- Parametry graf. okna: par, argumenty – cex, mar, mfrow, las, xaxs, xaxt, tck.
- Další typy grafů: barplot, boxplot, hist, stripchart, stem, pie, matplot.

Kontrolní úkoly

14. U tří lokalit (Opava, Ostrava, Karviná) porovnáváme průměrný počet hnízd na hektar (12, 8.5, 13.6) Vytvořte sloupcový graf s názvy osy x a y a popisy jednotlivých sloupců (názvy lokalit), s měřítkem na ose y od 0 do 14.
15. Zjistěte, jaké jsou souřadnice středů sloupců v předchozím grafu.
16. Místo sloupcového grafu vytvořte pro uvedené lokality koláčový graf i s popisky jednotlivých dílků.
17. Vytvořte stejný graf jako u úkolu 14, ale pro dva druhy (modrý a červený), v rozmezí 0-25 na ose y a přidejte legendu. Vytvořte nejprve graf s jedním sloupcem pro každou lokalitu, kde druhy budou nad sebou, pak graf, kde bude mít každý druh samostatný sloupec.

```
  Opava Ostrava Karviná
1    12       8      13
2     5       4       3
```

18. Vygenerujte 50 náhodných čísel z Poissonova rozdělení o průměru 3. Vektor nazvěte vec1. Použijte leaf-and-stem diagram.
19. Načtěte data z tabulek ukazujících různé styly zápisu dat (jsou uvedeny na začátku této podkapitoly) do objektů s názvem styl1, styl2. Vytvořte boxploty pro oba styly.
20. Vytvořte tabulku dat mtab2 se sloupci "samec", "samice"kde bude vždy 25 naměřených velkostí těla samců a samic určitého druhu. Vytvoříme je jako 25 náhodných čísel z normálního rozdělení pro každý sloupec (průměr=15, sm. odch.=3 pro samce, průměr=10, sm. odch=1 pro samice). Vytvořte boxplot pro tabulku, kde náhodné hodnoty budou velikosti jedinců. Graf by měl obsahovat také názvy os.
21. Vytvořte matplot hodnot tabulky mtab2. Následně upravte body (pro samce kroužky a pro samice čtverce).
22. Vypište leaf-and-stem diagram pro samce i samice z tabulky mtab2. Co diagram obsahuje?
23. Vytvořte do jednoho grafického okna histogram pro výšku samců a druhý histogram pro výšku samic. Uveďte nadpis histogramů (samci, samice), název osy x je "velikost těla [cm]".

Výsledky

```
1. x<-c(1,1,2,3,4,5,4,4,1,1,2,2,5,4,6,5,3); plot(x)
2. plot(x,type="l"); plot(x,type="h"); plot(x,type="o",xlim=c(0,20),ylim=c(0,10),col=3)
3. plot(x,type="o",xlim=c(0,20),ylim=c(0,10),col=3,lty=3,lwd=3)
4. plot(x,type="p",pch=21,bg=2,main="Můj graf",sub="1. část")
5. plot(factor(x)); plot(factor(x),col=c(2,2,2,4,4,4))
```

6. plot(factor(x),plot=F); vypsaly se x-ové souřadnice středů sloupců
7. plot(function(x) log(x),0,10); plot(function(x) abs(-2*x+1),-2,2)
8. x1<-rpois(30,5);plot(x1)
9. plot(x1,xaxs="i",yaxs="i",cex.axis=0.9,las=1);
   par(xaxs="i",yaxs="i",cex.axis=0.9,las=1);plot(x1)
10. plot(c(1,2,4,2,1,5,7)); v prvním případě budeou parametry nastaveny jako v grafu z úkolu 9. Zavřením grafického okna se parametry ruší, takže po opakování procedury už nastavení nebude shodné s předchozím.
11. par(mar=c(4,4,1,1)); plot(c(1,2,4,2,1,5,7))
12. par(mfrow=c(2,2)); plot(c(1,2,4,2,1,5,7)); plot(c(1,2,4,2,1,5,7),type="l");
    plot(c(1,2,4,2,1,5,7),type="o"); plot(c(1,2,4,2,1,5,7),type="h")
13. mtab<-read.table("clipboard"); plot(mtab$x,mtab$y)
14. barplot(c(12,8.5, 13.6), names.arg=c("Opava","Ostrava","Karviná"),xlab="lokality",
    ylab="průměr na ha",ylim=c(0,14))
15. ke grafu přidáme argument plot=F
16. pie(c(12,8.5, 13.6),labels=c("Opava","Ostrava","Karviná"))
17. mtab<-read.table("clipboard"); barplot(as.matrix(mtab),legend.text=c("druh 1", "druh 2"),ylim=c(0,25), col=c(2,4)); pro druhou část přidáme pouze argument beside=T
18. vec1<-rpois(50,3);stem(vec1)
19. mtab<-read.table("clipboard"); boxplot(styl1);boxplot(hodnota~typ,data=styl2)
20. mtab2<-data.frame(samci=rnorm(25,15,3), samice=rnorm(25,10,1)),
    boxplot(mtab2,xlab="pohlaví",ylab="velikost těla [cm]")
21. matplot(mtab2); matplot(mtab2,pch=1:0)
22. stem(mtab2$samci); stem(mtab2$samice) #stem (kmen obsahuje desítky a listy jsou jednotky, tzn. hodnoty jsou zaokrouhleny na celá čísla)
23. par(mfrow=c(1,2)); hist(mtab2[,1],main="samci",xlab="velikost těla [cm]");
    hist(mtab2[,2],main="samice",xlab="velikost těla [cm]")

POKROČILEJŠÍ GRAFIKA

Cíle kapitoly:
Po prostudování kapitoly zvládnete toto:
- naučíte se používat různé grafické prvky s symboly;
- budete umět používat pokročilejší nastavení jednotlivých částí grafu;
- pochopíte metodu vkládání matematických symbolů do grafu;
- seznámíte se s některými speciálními typy grafů.

Klíčová slova: Speciální grafy, grafické symboly, prvky grafu, nastavení grafického okna.

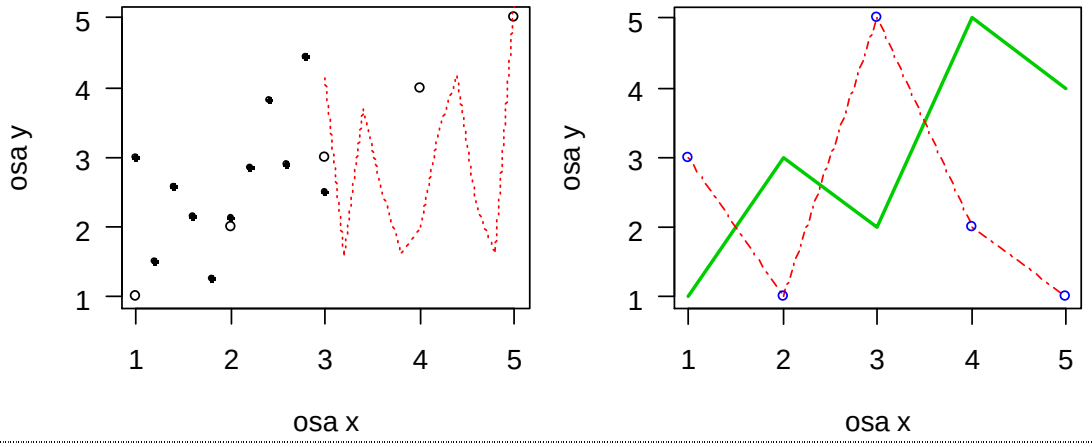
Průvodce
V poslední kapitole se zaměříme na náročnější způsoby práce s grafy. Je určena spíše pokročilejším uživatelům a těm, kteří jsou po prostudování předchozí kapitoly přesvědčeni, že program neumí kvalitně pracovat s grafické prvky. Začátečníkovi proto doporučuji projít si jednotlivé funkce pouze tak, aby si uvědomil, co lze ještě dále v grafech upravovat. Aby například věděl, že je možné upravovat zvlášť jednotlivé osy grafu, že lze do grafu vkládat další text atd.

Dále uváděné grafické prvky a funkce patří mezi tzv. low-level grafiku. Tvorba grafu s low-level graphics je sice o něco delší, umožňuje však mnohem lepší nastavení jednotlivých grafických prvků. Vytváříme totiž buď celý graf po samostatných částech s větším množstvím argumentů než u high-level grafiky nebo přidáváme jen některé prvky do již existujícího grafu. Je nutné upozornit, že v následujícím přehledu nejsou uvedeny všechny dostupné funkce. Vynechány jsou zejména pokročilejší příkazy pro práci s grafickými výstupy (příkazy z knihovny grDevices).

Grafické prvky

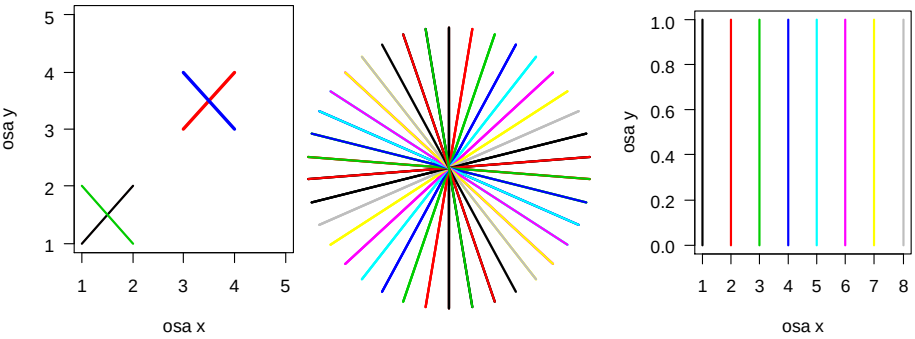
- **points**(x, y, type, pch, col, bg, cex, ...) – umožňuje vykreslení bodů se souřadnicemi x a y (další parametry viz plot) do již existujícího grafu. Využívá se velice často k přidání řady do grafu v případě, že nepoužíváme matplot.
- **lines**(x, y, type, col, lty, ...) – vytvoří souvislou čáru mezi body danými souřadnicemi x a y (další parametry viz plot) do již existujícího grafu.

```
#graf vlevo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(1:5,xlab="osa x",ylab="osa y")#vykreslí graf se základními body
points(seq(1,3,0.2),rnorm(11,3,1),pch=20)#vykreslí náhodně body
lines(seq(3,5,0.2),rnorm(11,3,1),lty=3,col=2)#vykreslí náhodně čáru
#graf vpravo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(1:5,type="n", xlab="osa x",ylab="osa y")#vykreslí graf bez bodů
lines(c(1,3,2,5,4),lwd=2,col=3)
lines(c(3,1,5,2,1),lwd=1,col=2,lty=4)
points(c(3,1,5,2,1),col=4)
```



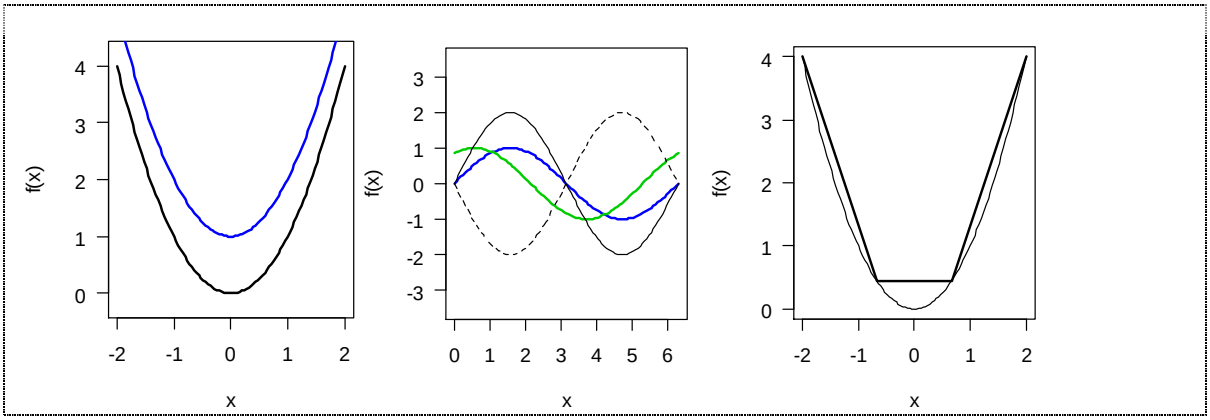
- **segments**(x0, y0, x1, y1, col, lty, lwd, ...) – vykreslí samostatné úsečky mezi počátečními (A) a koncovými body (B) se souřadnicemi A[x0,y0], B[x1,y1]; v tomto případě (narozdíl od lines) lze měnit barvy úseček.

```
#graf vlevo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(1:5,type="n", xlab="osa x",ylab="osa y")#vykreslí graf bez bodù
segments(c(1,3,1,3),c(1,3,2,4),c(2,4,2,4),c(2,4,1,3),lwd=1:2,col=2:3)
#graf uprostřed
par(mar=c(0,0,0,0))#nulové okraje
x<-seq(0,pi,along=1:20)[-20]#interval 0,1 rozdělený na 20 dílků
plot(c(-1,1), c(-1,1),type="n", asp = 1,axes=F,ann=F)#čistý graf
segments(x0=sin(x),x1=-sin(x),y0=cos(x),y1=-cos(x),col=1:8,lwd=2)#hvězda
#graf vpravo
plot(0,0,xlim=c(1,8),ylim=c(0,1),ty="n", xlab="osa x",ylab="osa y")
segments(1:8,rep(0,8),1:8,rep(1,8),lwd=2,col=1:8)
```



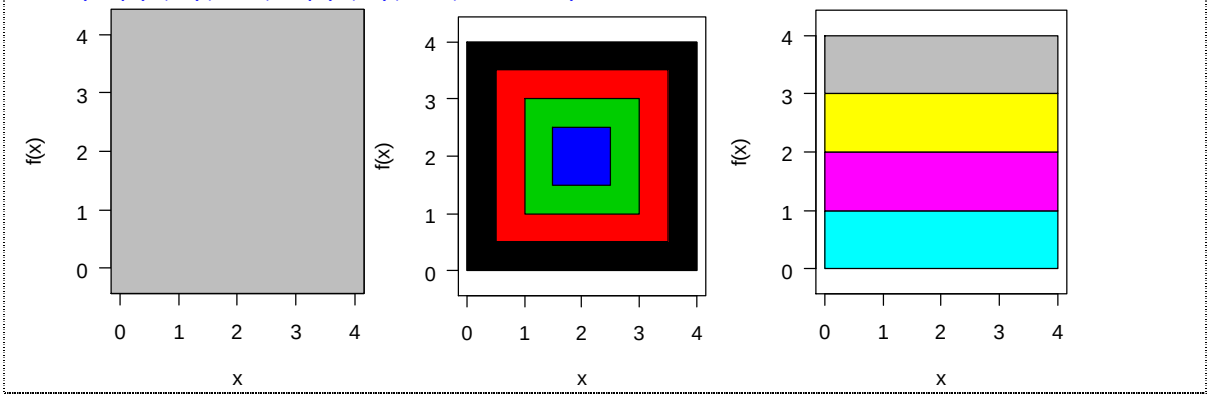
- **curve**(expr, from, to, n, add=F, type, ylab, log, xlim, ...) – vykreslí funkci podle výrazu expr.
  from , to – interval pro který se funkce vykreslí hodnot od do
  n – počet bodů spojených liniemi nebo body (podle type, viz např. plot),
  add – vykreslit nový graf nebo přidat do již vytvořeného grafu, ostatní výrazy jako u plot

```
#graf vlevo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(-2,2),ylim=c(0,4),asp=1)
curve(x^2,-2,2,lwd=2,add=T)#první parabola (černá), vynecháním add překreslíme
curve(x^2+1,-2,2,lwd=2,col=4,add=T) # druhá parabola (modrá)
#graf uprostřed
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(0,2*pi),ylim=c(-1,1),asp=1)
curve(sin(x),0,2*pi,lwd=2,col=4,add=T)#modrá křivka
curve(sin(x+1),0,2*pi,lwd=2,col=3,add=T) #zelená křivka
curve(2*sin(x),0,2*pi,lwd=1,col=1,add=T) #normální černá křivka
curve(-2*sin(x),0,2*pi,lty=2,col=1,add=T) #přerušovaná černá křivka
#graf vpravo
curve(x^2,-2,2,lwd=1,xlab="x",ylab="f(x)")#vykreslí nový graf
curve(x^2,-2,2,lwd=2,n=4,add=T)#shodná křivka, ale jen 4 body
```



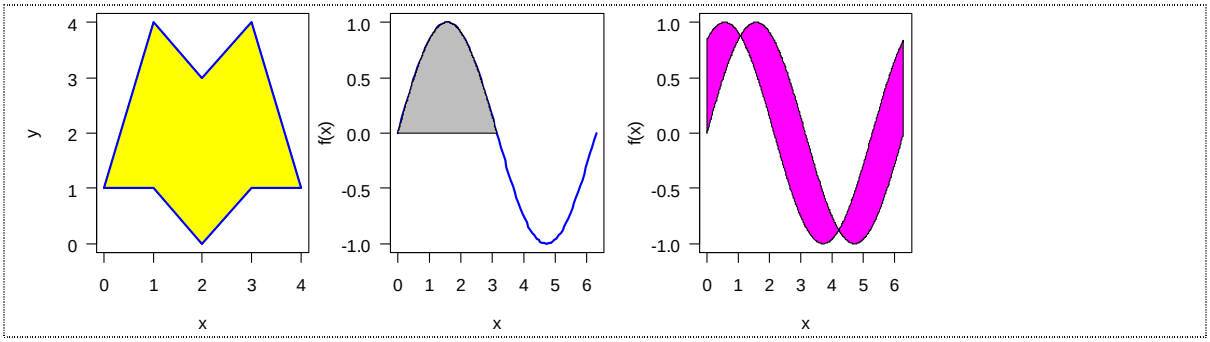
- **rect**(x1, y1, x2, y2, density, angle, col, border, lty, lwd, xpd, …) – vykreslí obdélníkovou oblast podle souřadnic (xvlevo, ydole, xvpravo, ynahoře). Další nastavení parametrů stejné jako u ost. graf. prvků.

```
#graf vlevo #vykreslení pozadí plochy grafu
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(0,4),ylim=c(0,4),asp=1)
rect(par("usr")[1], par("usr")[3], par("usr")[2], par("usr")[4],col=8)
#graf uprostřed
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(0,4),ylim=c(0,4),asp=1)
x<-seq(0,1.5,0.5);y<-seq(4,2.5,-0.5)
rect(x,x,y,y,col=1:8)
#graf vpravo
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(0,4),ylim=c(0,4),asp=1)
rect(rep(0,4),0:3,rep(4,4),1:4,col=5:8)
```



- **polygon**(x,y,density,angle,border,col,lty,xpd,…) – vytvoří mnohoúhelník(y) s vrcholy v bodech zadaných pomocí x a y (spojení mnohoúhelníku) je podle pořadí bodů).

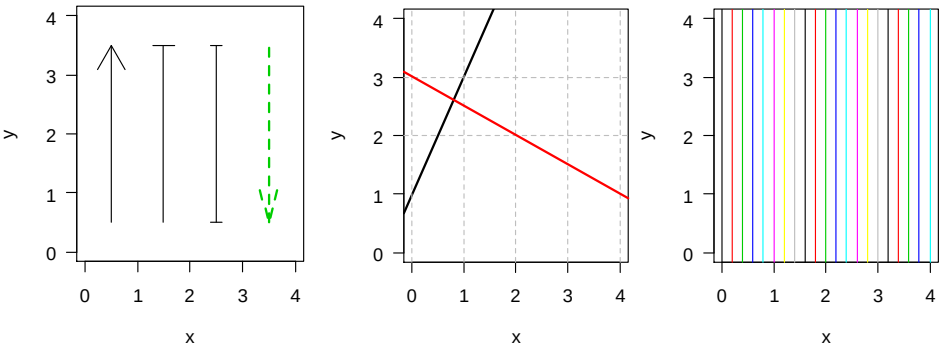
```
#graf vlevo #vykreslení mnohoúhelníku
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
polygon(c(0:4,3:1),c(1,1,0,1,1,4,3,4),col=7,border=4,lwd=2)
#graf uprostřed
curve(sin(x),0,2*pi,lwd=2,col=4,xlab="x",ylab="f(x)")
polygon(seq(0,pi,0.01),sin(seq(0,pi,0.01)),col=8)
#graf vpravo
plot(0,0,type="n",xlab="x",ylab="f(x)",xlim=c(0,2*pi),ylim=c(-1,1))
polygon(c(x,rev(x)),c(sin(x),rev(sin(x+1))),col=6)
```



- **`arrows`**`(x0, y0, x1, y1, length, angle, code, col, lty, lwd, xpd)` – vykreslí šipku podle zadaných souřadnic (popř. šipky, jestliže jsou souřadnice zadány jako vektory). Pomocí šipek se dají také vykreslit chybové úsečky v grafech (např. konfidenční intervaly nebo střední chyby průměru). Stačí pouze vytvořit šipku na obě strany s úhlem 90°.
  `length` – délka hlavy šipky
  `angle` – úhel hlavy šipky
  `code` – typ šipky (0 – bez šipky, 1 – šipka na začátku, 2 – šipka na konci, 3 – obě strany)

- **`abline`**`(a,b,untf=F,…)` nebo `(h,…)` nebo `(v,…)` nebo `(reg` regresní objekt`)` – vykreslí přímku (nebo přímky) typu *`bx+a`* (*`untf`* – v případě logaritmovaných os vykreslí netransformovanou přímku), popř. je možno zadat *`h`* a *`v`* jako vektory horizontálních a vertikálních přímek.

```
#graf vlevo #šipky
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
arrows(0.5,0.5,0.5,3.5)
arrows(1.5,0.5,1.5,3.5,angle=90,length=0.1)
arrows(2.5,0.5,2.5,3.5,angle=90,length=0.05,code=3)
arrows(3.5,0.5,3.5,3.5,angle=15,length=0.3,code=1,lwd=2,lty=2,col=3)
#graf uprostřed
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
abline(1,2,lwd=2) #vykreslení přímky 2x+1
abline(v=0:4,lty=2,col=8) #vykreslení jemné mřížky vertikálně
abline(h=2:3,lty=2,col=8) #vykreslení jemné mřížky horizontálně
abline(3,-0.5,lwd=2,col=2) #vykreslení přímky -0.5x+3
#graf vpravo
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
abline(v=seq(0,4,0.2),col=1:8)
```

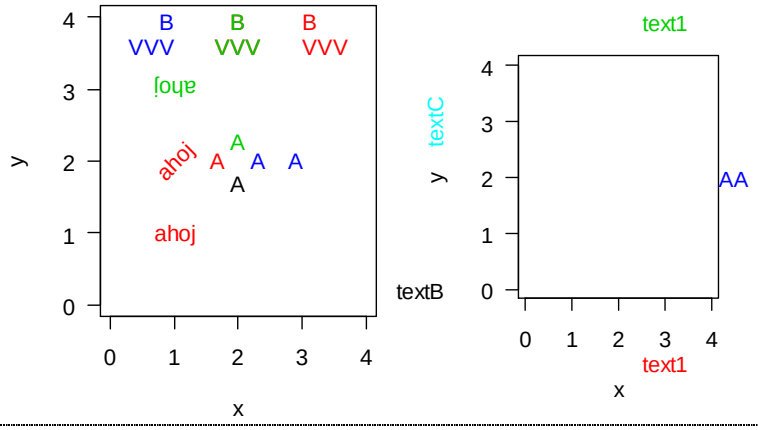


- **`text`** `(x, y, labels, adj, pos, offset, vfont, cex, col, font, xpd, ...)` – vypíše text(y) do grafu na souřadnicích `[x,y]`. Může se využít k popisování jednotlivých bodů grafu atd.
  `labels` – text, který chceme vypsat. Může být i vektor, ale pak musí být také xy souřadnice vektory.
  `adj` – zarovnání (0 – vlevo, 0.5 – doprostřed, 1 – vpravo)
  `pos` – umístění od pozice (1 dolů, 2 vlevo, 3 nahoru, 4 vpravo)
  `offset` – o kolik se má text zvýšit nebo snížit (relativně k velikosti znaku) vztaženo k hodnotě `pos`

vfont – specifikace vektorového fontu (lze použít Hershey vektorové fonty viz demo(Hershey)
font – typ normálního fontu

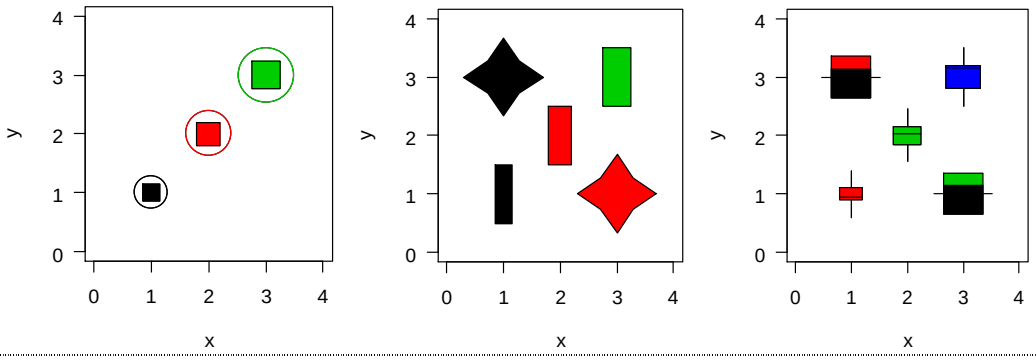
- **mtext**(text,side,line,outer=F,at,adj,padj,cex,col,font,vfont,las,…) – výpis textu na okraje grafu.
  side – umístění na okraj (1 dolů, 2 vlevo, 3 nahoru, 4 vpravo)
  line – který řádek v okraji
  outer – u více grafů v jednom okně vzniká ještě vnější okno, psát do tohoto okna?
  at – specifikace pozice
  padj – zarovnání svislých řetězců
  las – sklon na jednotlivých osách, viz plot

```
#graf vlevo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
text(1,1,"ahoj",col=2)
text(1,2,"ahoj",col=2,srt=45)
text(1,3,"ahoj",col=3,srt=180)
text(2,2,"A",pos=1,col=1) # pod bod
text(2,2,"A",pos=2,col=2) # vlevo od bodu
text(2,2,"A",pos=3,col=3) # nad bod
text(2,2,"A",pos=4,col=4) # vpravo od bodu
text(2,2,"A",pos=4,col=4,offset=2)
text(1,3.8,"B\nVVV",adj=1,col=4)#zalomení řádku a zarovnání vpravo
text(3,3.8,"B\nVVV",adj=0,col=2) #zalomení řádku a zarovnání doprostřed
text(2,3.8,"B\nVVV",adj=0.5,col=3) #zalomení řádku a zarovnání vlevo
#graf vpravo
par(mar=c(5,5,2,2),las=1)
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
mtext(side=1,at=3,text="text1",col=2,line=2)
mtext(side=3,at=3,text="text1",col=3,line=1)
mtext(side=4,at=2,text="AA",col=4,line=0)
mtext(side=2,at=0,text="textB",col=1,line=3)
mtext(side=2,at=3,text="textC",col=5,line=3,las=3)
```



- **symbols**(x,y,circles,squares,rectangles,stars,thermometers,boxplots, inches=T,add=F,fg,bg,xlab,ylab,main,xlim,ylim,…) – vykresluje symboly dle zadaných hodnot.
  x,y – zadání souřadnic středu daného symbolu
  circles – vektor poloměrů kružnic
  squares – vektor stran jednotlivých čtverců
  rectangles – 2 sloupcová matice (1. sloupec šířka, 2. sloupec výška)
  stars – matice s 3 a více sloupci označujícími vzdálenost jednotl. bodů hvězdy od středu
  thermometres – 3-4 sloupce označující šířku, výšku, proporci vybarvení (3. a 4. sloupec) symbolu teploměru (barvy pro vybarvení jednotlivých části jsou argumenty fg a bg)
  boxplots – matice s 5 sloupci (šířka a výška krabice, délky vousů a proporce mediánu)
  inches – měřítko pro symboly je v palcích (T) nebo v jednotkách grafu (F)
  add – přidat do stávajícího grafu?

```
#graf vlevo
par(mar=c(4.2,4,0.3,0.2),las=1)
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
symbols(1:3,1:3,circles=c(0.3,0.4,0.5),inches=F,add=T,fg=1:3)
symbols(1:3,1:3,squares=c(0.3,0.4,0.5),inches=F,add=T,bg=1:3)
#graf uprostřed
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
symbols(1:3,1:3,rectangles=matrix(c(0.3,0.4,0.5,1,1,1),3,2),inc=F,add=T,bg=1:3)
symbols(c(1,3),c(3,1),stars=matrix(rep(c(0.7,0.4),8),2,8,T),inc=F,ad=T,bg=1:3)
#graf vpravo
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
x<-matrix(c(0.4,0.5,0.6,0.2,0.3,0.4,rep(0.3,6),0.2,0.6,0.9),3,5)
x
     [,1] [,2] [,3] [,4] [,5]
[1,]  0.4  0.2  0.3  0.3  0.2
[2,]  0.5  0.3  0.3  0.3  0.6
[3,]  0.6  0.4  0.3  0.3  0.9
symbols(1:3,1:3,boxplot=x,inc=F,add=T,bg=2:4)
symbols(c(1,3),c(3,1),thermometers=matrix(rep(0.7,6),2,3),inc=F,ad=T,fg=2:3)
```



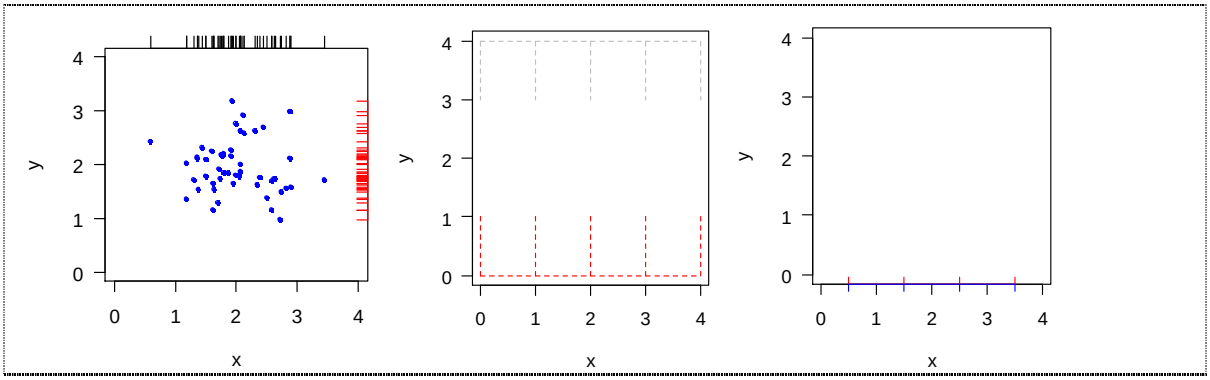
- **rug** (x,ticksize=0.03,side=1,lwd=0.5,col,quiet,…) vytvoří „rohož“ s rozestupy podle vektoru x).

  ticksize – velikost "třásní rohože" v proporci grafu (1 – přes celý graf, záporné hodnoty jsou vykreslovány ven z grafu

  side – strana na které se má "rohož" vykreslovat (1 dole, 2 vlevo, 3 vpravo, 4 nahoře)

  quiet – vypisovat hlášení o překrývajících se hodnotách?

```
#graf vlevo
par(mar=c(4.2,4,1,1),las=1)
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
x<-rnorm(50,2,0.5);y<-rnorm(50,2,0.5)
points(x,y,pch=20,col=4)
rug(x,ticksize=-0.05,side=3)
rug(y,ticksize=0.05,side=4,col=2)
#graf uprostřed
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
rug(0:4,ticksize=0.25,side=1,col=2,lty=2,pos=0)
rug(0:4,ticksize=0.25,side=3,col=8,lty=2,pos=4)
#graf vpravo
plot(0,0,type="n",xlab="x",ylab="y",xlim=c(0,4),ylim=c(0,4))
rug(seq(0.5,3.5,1),ticksize=0.03,side=1,col=2)#vykreslení červených ticks
rug(seq(0.5,3.5,1),ticksize=-0.03,side=1,col=4) #vykreslení modrých ticks
```



Kontrolní úkoly

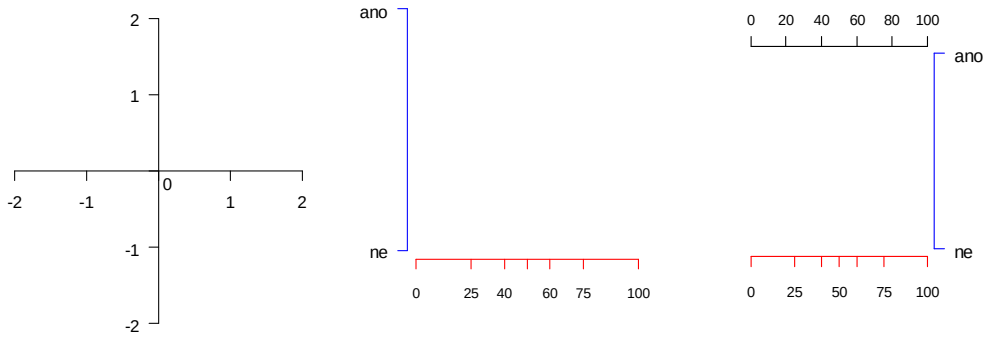
1. Vytvořte prázdný graf s měřítkem 0-10 na ose x a 0-10 na ose y, názvy osy x i y chybí, nastavení okrajů (v měřítku řádků) je 4,4,1,1), popisky osy x a y jsou vypsány v horizontální pozici. Toto nastavení používejte pro všechny úkoly (nemusíte graf překreslovat, jen dodávejte další objekty).
2. Vytvořte body (tvar obdélníku) na souřadnicích [1,3], [2,8], [3,1]. První bod bude červený, druhý zelený a třetí modrý, velikost bodu bude postupně 1, 2, 3.
3. Vykreslete přerušovanou spojnici bodů, tloušťka čáry je 2 a barva modrá.
4. Zakreslete do předchozího grafu funkci sin(x)+5 červeně, a funkci sin(x+1)+5 modře.
5. Vykreslete šipku směřující z bodu 0,0 do bodu 1,1.
6. Vytvořte vektor čísel 5,4,7,8,3,1,2 nazvaný vec3. Vykreslete body se souřadnicemi x postupně 4-10, souřadnice y jsou uloženy ve vektoru vec3. Vytvořte chybové úsečky bodů pohybující se v rozmezí ±1 okolo bodů (vertikálně).
7. Vykreslete přímku -3x+10, dále svislé žluté přímky protínající osu x v bodech 3 a 4 a 3 horizontální přímku (přerušovaná čára, šedá barva) protínající osu y v bodě 10.
8. Vedle bodů z úkolu 6 vepište postupně popisky a-g.
9. Vepište pod osu x text (osa x) tak, aby ležel na pozici pod souřadnicí 9 na 3. "řádku". Proč je 3. řádek line=2?

Části grafu

- **`axis`**`(side, at, labels=T, tick=T, line, pos, outer=F, font, vfont, lty, lwd, col, padj, ...)` – vykreslí osu, přičemž umožňuje nastavit značky osy a popisky značek naprosto libovolně (i s různými rozestupy).
  `side` – strana na které se má osa vykreslovat (1 dole, 2 vlevo, 3 vpravo, 4 nahoře)
  `at` – hodnoty, ve kterých se mají vykreslit značky
  `labels` – názvy pro jednotlivé značky
  `tick` – logical, mají se značky vykreslovat?
  `lines` – na kterém řádku v okraji se bude osa vykreslovat
  `pos` – na které pozici se osa vykreslí
  `outer` – pro více grafů v jednom, má se vykreslovat osa na vnějším okraji?

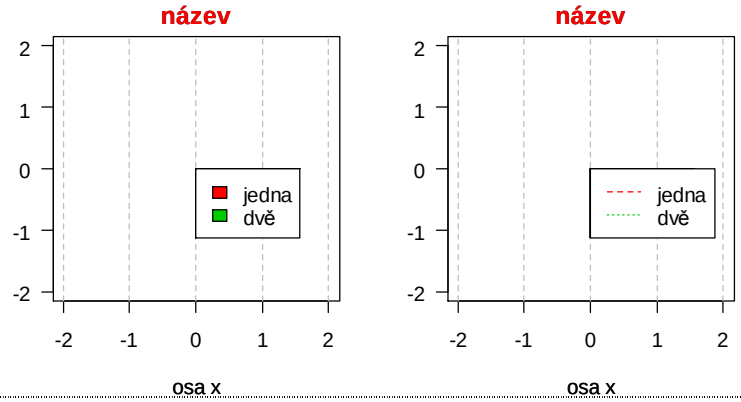
```
#graf vlevo
par(mar=c(0.1,0.1,0.1,0.1),las=1)
plot(0,0,type="n",axes=F,ann=F,xlim=c(-2,2),ylim=c(-2,2))
axis(1,at=-2:2,labels=c(-2,-1,"",1,2),pos=0)
axis(2,at=-2:2,labels=c(-2,-1,"",1,2),pos=0)
text(0,0,adj=c(-0.5,1.3),labels=0)
#graf uprostřed
par(mar=c(4,4,0.1,0.1),las=1)
plot(0,0,type="n",axes=F,ann=F,xlim=c(0,100),ylim=c(0,1))
axis(1,at=c(0,25,40,50,60,75,100),col=2,cex.axis=0.8)
axis(2,at=0:1,labels=c("ne","ano"),las=1,col=4)
```

```
#graf vpravo
par(mar=c(4,4,3,3),las=1)
plot(0,0,type="n",axes=F,ann=F,xlim=c(0,100),ylim=c(0,1))
axis(1,at=c(0,25,40,50,60,75,100),col=2,cex.axis=0.8)
axis(3,at=seq(0,100,20),col=1,cex.axis=0.8)
axis(4,at=0:1,labels=c("ne","ano"),las=1,col=4)
```



- **`title`**`(main,sub,xlab,ylab,line,outer=F,…)` – dodatečné dokreslení názvu, podnázvu nebo popisků os do grafu.
- **`grid`**`(nx,ny,col,lty,lwd,equilogs=T)` – vykreslí mřížku v grafu.
  `nx,ny` – udává počet vertikálních, popř. horizontálních čar, nenastaveno vykresluje čáry podle stávajících značek na ose
  `equilogs` – pro logaritmované osy (viz ?grid)
- **`legend`**`(x,y,legend,fill, col, lty, lwd, pch, angle, density, bty, bg, pt.bg, cex, pt.cex, pt.lwd, xjust, yjust, x.intersp, y.intersp, adj, text.width, text.col, merge, trace=F, plot=T, ncol, horiz=F)` – vykreslí legendu grafu na souřadnicích x a y (vysvětlení všech argumentů viz ?legend)
  `legend` – popisky v legendě
  `fill` – značky popisu legendy (barevné obdélníky)
  `lty` – typ čar v legendě

```
par(mar=c(4,4,2,0.1),las=1)
plot(0,0,type="n",ann=F,xlim=c(-2,2),ylim=c(-2,2))
title(main="název",xlab="osa x",col.main=2)
grid(ny=0,col=8,lty=2)
legend(0,0,legend=c("jedna","dvě"),fill=2:3,bg="white")
legend(0,0,legend=c("jedna","dvě"),lty=2:3,col=2:3,bg="white")
```



Grafický zápis matematických symbolů a vzorců

Speciální matematický zápis – R umožňuje pomocí speciálních symbolů zápis matematických výrazů v kvalitní grafické interpretaci. Zápis lze využít při psaní popisků os, nadpisů, popisků značek, textu, textu na okraji atd. Místo řetězce se vepisuje výraz `expression(…)`.

x+y	$x+y$	hat(x)	$\hat{x}$	paste(x,y,z)	xyz
x-y	$x-y$	tilde(x)	$\tilde{x}$	list(x,y,z)	x, y, z
x*y	xy	ring(x)	$\mathring{x}$	plain(Ahoj)	Ahoj
x/y	x/y	bar(x)	$\bar{x}$	italic(Ahoj)	*Ahoj*
-x	$-x$	widehat(xy)	$\widehat{xy}$	bold(Ahoj)	**Ahoj**
+x	$+x$	widetilde(xy)	$\widetilde{xy}$	bolditalic(Ahoj)	***Ahoj***
x[i]	x_i	x%<->%y	$x \leftrightarrow y$	underline(Ahoj)	<u>Ahoj</u>
x^i	x^i	x%<-%y	$x \leftarrow y$	list(x[1],...,x[n])	$x_1, \ldots, x_n$
x<y	$x<y$	x%subset%y	$x \subset y$	x[1]+...+x[n]	$x_1 + \cdots + x_n$
x>y	$x>y$	x%subseteq%y	$x \subseteq y$	list(x[1],cdots,x[n])	$x_1, \cdots, x_n$
x<=y	$x \leq y$	x%supset%y	$x \supset y$	x[1]+ldots+x[n]	$x_1 + \ldots + x_n$
x>=y	$x \geq y$	x%supseteq%y	$x \supseteq y$	displaystyle(x)	x
x==y	$x = y$	x%notsubset%y	$x \not\subset y$	textstyle(x)	x
x!=y	$x \neq y$	Alpha-Omega	$A - \Omega$	scriptstyle(x)	x
x%+-%y	$x \pm y$	alpha-omega	$\alpha - \omega$	scriptscriptstyle(x)	x
x%/%y	$x \div y$	phi1+sigma1	$\varphi + \varsigma$	x+phantom(0)+y	x+ +y
x%*%y	$x \times y$	Upsilon1	Υ	over(1,phantom(0))	$\frac{1}{}$
x%*%y	$x \cdot y$	infinity	∞	frac(x,y)	$\frac{x}{y}$
sqrt(x)	$\sqrt{x}$	32*degree	32°	over(x,y)	$\frac{x}{y}$
x%~~%y	$x \approx y$	60*minute	$60'$	atop(x,y)	$\begin{matrix} x \\ y \end{matrix}$
x %=~%y	$x \cong y$	30*second	$30''$	sum(x[i],i=1,n)	$\sum_{1}^{n} x_i$
x%==%y	$x \equiv y$	x~~y	$x \; y$	prod(plain(P)(X==x),x)	$\prod_{x} P(X = x)$
x%prop%y	$x \propto y$	inf(S)	$\inf S$	integral(f(x)*dx,a,b)	$\int_a^b f(x)dx$
x%in%y	$x \in y$	sup(S)	$\sup S$	union(A[i],i==1,n)	$\bigcup_{i=1}^{n} A_i$
x%notin%y	$x \notin y$	(x+y)*z	$(x+y)z$	intersect(A[i],i==1,n)	$\bigcap_{i=1}^{n} A_i$
x%->%y	$x \rightarrow y$	x^y+z	$x^y + z$	lim(f(x),x%->%0)	$\lim_{x \to 0} f(x)$
x%up%y	$x \uparrow y$	x^(y+z)	$x^{(y+z)}$	min(g(x),x>=0)	$\min_{x \geq 0}(x)$
x%down%y	$x \downarrow y$	x^{y+z}	x^{y+z}	group("(",list(a,b),"]")	$(a, b]$
x%<=>%y	$x \Leftrightarrow y$	x%dblup%y	$x \Uparrow y$	bgroup("(",atop(a,b),")")	$\binom{a}{b}$
x%=>%y	$x \Rightarrow y$	x%dbldown%y	$x \Downarrow y$	group(lceil,x,rceil)	$\lceil x \rceil$
x%<=%y	$x \Leftarrow y$			group(lfloor,x,rfloor)	$\lfloor x \rfloor$
				group("|",x,"|")	$\|x\|$

Kontrolní úkoly

10. Vytvořte prázdný graf s měřítkem -5, +5 na ose x a 0-8 na ose y, názvy se x i y chybí, nastavení okrajů (v měřítku řádků) je 4,4,1,1), osy se nevykreslí. Toto nastavení používejte pro všechny úkoly (nemusíte graf překreslovat, jen dodávejte další objekty).
11. Vykreslete svislou i vodorovnou mřížku (šedá barva)
12. Vykreslete osu y uprostřed (procházející bodem 0 na ose x).
13. V bodě [3,7] vykreslete legendu, kde bude modré označení a popisek "druh 1" a červené označení a popisek "druh 2".
14. Vykreslete osu x na které budou značky v bodech -5,-3, 3, 5, kde u 5 a -5 nebude žádný popisek a u -3 a 3 bude napsáno x_0-Δt a x_0+Δt.
15. Vypište název grafu "Graf pro druhy" na řádek 0 horního okraje grafu

Manipulace s grafickými okny

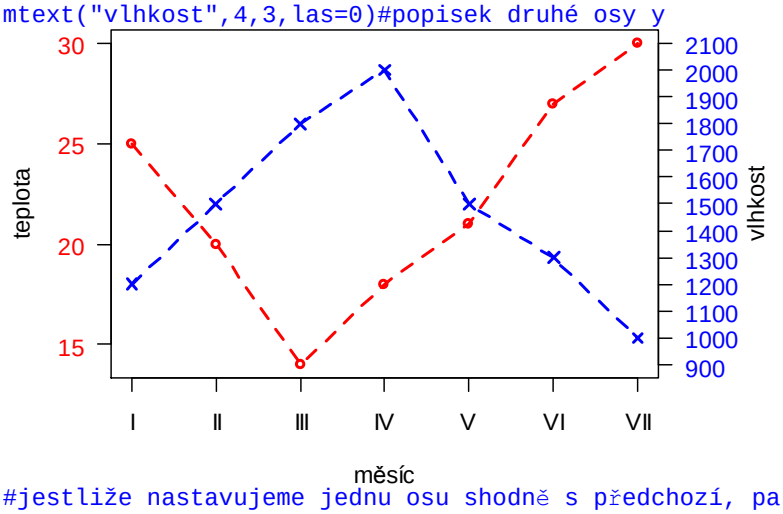
- **`recordPlot()`** – umožňuje nahrát aktuální vykreslený graf jako objekt (např. `myplot<-recordPlot()`), zpětně lze vykreslit pomocí `replayPlot()` nebo názvem objektu. Tuto funkci nelze vždy využít mezi jednotlivými verzemi programu.
- **`windows`**`(width=7,height=7,pointsize=12,record=getOption("graphics.record"),rescale=c("R", "fit", "fixed"), xpinch, ypinch,bg= "transparent", canvas="white",gamma=getOption("gamma"),xpos=NA,ypos=NA,buffered=getOption("windowsBuffered"),restoreConsole=F)` – vykreslí grafické okno o určité velikosti, využívá se pro přesné nastavení velikosti okna, písma atd., díky otevření více oken lze kreslit najednou do více oken
  `width`,*`height`* – šířka a výška okna v palcích (závisí na nastavení xpinch a ypinch)
  `pointsize` – velikost písma
  `rescale` – při změně velikosti okna se mění i graf (`"R"`), graf se mění, ale je zachována proporce (`"fit"`), graf zůstává stejně velký jako původní (`"fixed"`)
  `xpinch`,*`ypinch`* – počet bodů na palec (horizontálně a vertikálně)
  `bg`,*`canvas`* – barva pozadí a popředí grafického okna
  `xpos`,*`ypos`* – horizontální a vertikální pozice okna na obrazovce

```
windows(width=5,height=10,xpos=500,ypos=50)
plot.new()
windows(width=10,height=5,xpos=50,ypos=200)
plot.new()
```

- **`dev.cur()`** – číslo aktivního grafického okna
- **`dev.list()`** – seznam otevřených grafických oken
- **`dev.set`**`(which = dev.next())`, **`dev.next`**`(which = dev.cur())`, **`dev.prev`**`(which = dev.cur())`– přepnutí na určité grafické okno (*`which`*), následující okno nebo přechozí okno
- **`dev.off`**`(which = dev.cur())`, **`graphics.off`**`()`– vypíná určité grafické okno (*`which`*, standardně *`dev.cur()`*), popř. všechna grafická okna
- **`plot.new(),frame()`** – používá se ke konstrukci nového (prázdného) grafu, nebo k přeskakování grafů v případě více grafů v jednom grafickém okně.
- **`plot.window`**`(xlim,ylim,log="",asp=NA,…)` – nastavuje extrémy na ose x a y (limity – *`xlim`*, *`ylim`*, logaritmickou osu – *`log`*, *`asp`* – poměr stran), umožňuje vkreslit do stávající grafické oblasti graf se sekundární osou y (popř. také x) s jiným měřítkem.

```
teplota<-c(25,20,14,18,21,27,30)#hodnoty teploty
```

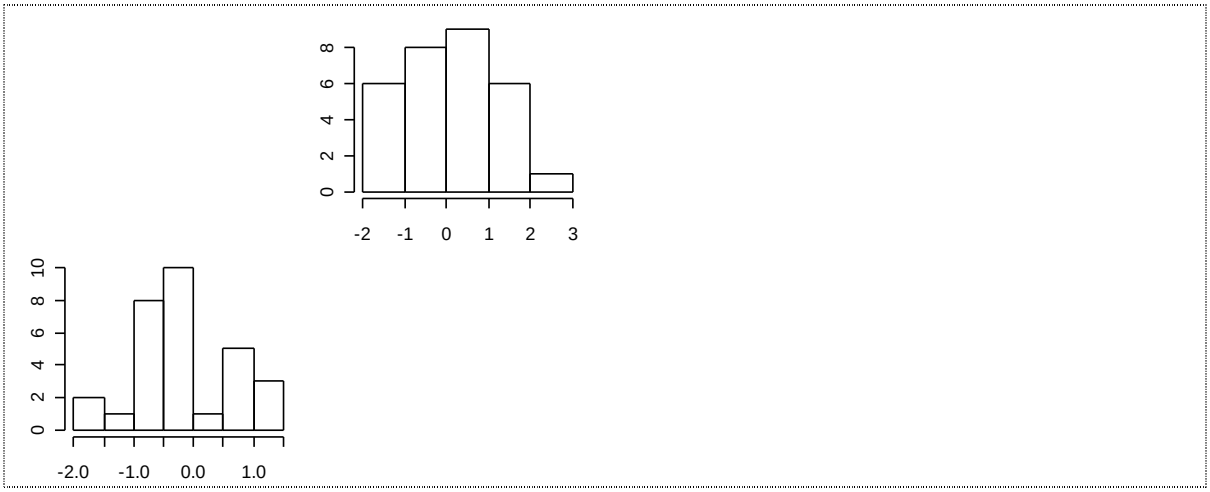
```
vlhkost<-c(1200,1500,1800,2000,1500,1300,1000)#hodnoty vlhkosti (jiný rozsah)
par(mar=c(4.1,4.1,0.1,4.1),las=1)#základní parametry
plot(teplota,xaxt="n",xlab="měsíc",type="o",lty=2,col=2,lwd=2,col.axis=2)#graf
axis(1,at=1:7,labels=c("I","II","III","IV","V","VI","VII"))#osa x
plot.window(xlim=c(1,7),ylim=c(900,2100))#změna měřítka grafu
points(vlhkost,type="o",pch=4,col=4,lty=2,lwd=2)#vykreslení hodnot vlhkosti
axis(4,at=seq(900,2100,100),col.axis=4) #druhá osa y
mtext("vlhkost",4,3,las=0)#popisek druhé osy y
```




```
#jestliže nastavujeme jednu osu shodně s předchozí, pak limit na této ose
#zjistíme pomocí parametru "usr" (viz par) nesmíme přitom zapomenout nastavit
#danou osu bez odsazení okrajů, jinak se nám měřítko posune, viz násl. příklad
#kde se osa x nemění a mění se jen osa y
plot(rnorm(40),rnorm(40))
plot.window(xlim=c(par("usr")[1],par("usr")[2]), ylim=c(5,9),xaxs="i")
abline(v=0) #čára by měla procházet přesně nulou (bez nastavení xaxs je posunuta)
```

- **split.screen**(figs,screen,erase=T) – rozdělí okno na řádky a sloupce, do kterých se pak vkreslují grafy (podobně mfrow, mf col u funkce par)
  figs – dvouprvkový vektor (počet řádků, počet sloupců),
  screen – které okno rozdělit
  erase – logical, vymazat před rozdělením?
- **screen**(n, new=T) – vybrat okno *n*, *new* – vymazat okno před vykreslováním?
- **erase.screen**(n) – vymaže okno *n*
- **close.screen**(n,all.screens=F) – zavřít okno *n*, *all.screens* – všechna okna

```
plot.new()
split.screen(c(2,2))
[1] 1 2 3 4
screen(2); hist(rnorm(30),main=NULL)
screen(3); hist(rnorm(30),main=NULL)
```



- **identify**(x,y,labels,pos=F,n,plot=T,offset,…) – interaktivní funkce, po kliknutí do grafického okna určuje, kterým ze zadaných souřadnic (*x*, *y*) byl kurzor nejblíže a vytiskne u nich popisky.
  n – nastavuje počet kliknutí, která se budou zaznamenávat
  labels – je-li zadáno vypisuje do grafu přímo názvy bodů (jinak jejich souřadnice)
  plot – zda mají být vykresleny popisky
  pos – kde mají být popisky bodů vykresleny

- **locator**(n =512,type="n",…) – určuje polohu grafického kurzoru při kliknutí do grafického okna
  n – nastavuje počet kliknutí, která se budou zaznamenávat
  type – po kliknutí je možné vykreslovat body, linie a další typy grafů (viz *plot*)

```
plot(c(1:10))
identify(c(1:10),n=3) #postupně klikněte na 3 vykreslené body

par(mar=c(4,4,2,0.1),las=1)
plot(0,0,type="n",ann=F,xlim=c(-2,2),ylim=c(-2,2))
locator(n=10,type="p",pch=21,bg=4) #klikněte 10-krát do okna
```

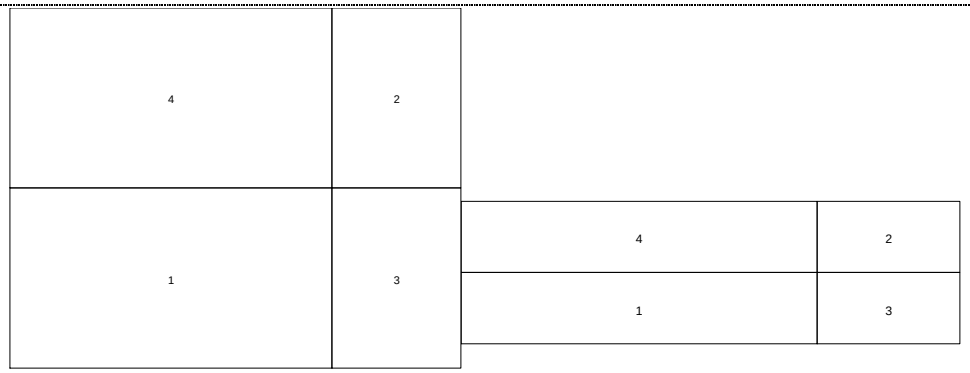
- **xinch**(x = 1, warn.log=T) , **yinch**(y = 1, warn.log=T), **xyinch**(xy = 1, warn.log=T) – převádí číslo (nebo dvojici čísel) v palcích na skutečné rozměry v grafu

- **cm**(x) – převádí palce na centimetry.

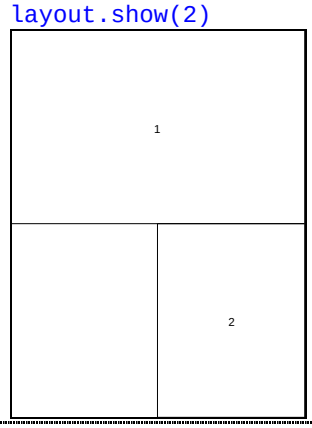
- **layout**(mat, widths, heights, respect=F) – vytvoří plán pro grafické okno, tzn. rozdělí aktuální grafické okno na podokna, do kterých je možné zakreslovat grafy postupně podle zadaného pořadí v matici *mat*, o šířkách sloupců a výškách řádků. Funkce je nekompatibilní s screen a mfrow, mfcol
  mat – matice udávající postupný
  labels – je-li zadáno vypisuje do grafu přímo názvy bodů (jinak jejich souřadnice)
  plot – zda mají být vykresleny popisky
  widths, *heights* – vztaženo relativně (poměrově šířka1/šířka2 nebo poměr délka–šířka) nebo absolutně (ve skutečných hodnotách pomocí lcm()).

- **layout.show**(n) – vykreslí n plánů

```
layout(matrix(c(4,2,1,3),2,2, byrow=T), width=c(5,2),height=c(1,1))
layout.show(4) #nastaví grafické okno v poměru 1:1 (výška), 5:3 (šířka)
#layout(matrix(c(4,2,1,3),2,2, byrow=T), width=5/2,height=1) totéž
layout(matrix(c(4,2,1,3),2,2, byrow=T), width=c(5,2),height=c(1,1),respect=T)
layout.show(4) #shodný poměr x a y (obr vpravo)
layout(matrix(c(4,2,1,3),2,2, byrow=T), width=lcm(c(5,2)),height=lcm(c(1,1)))
layout.show(4) #nastaví přesné rozměry v cm
```




```
layout(matrix(c(1,1,0,2),2,2, byrow=T))#vynechá třetí okno a 1. spojí
layout.show(2)
```



Práce s barvami

Při práci s barvami jsme dosud využívali pouze zadávání argumentu barvy číslem 1–8. R však využívá mnohem rozsáhlejší paletu barev. Argumenty typu `col` a `bg` lze zadávat také jako název barvy (v angličtině), v tzv. RGB kódu (popř. HSV) nebo pomocí čísel barev vybraných ze zvolené palety.

```
#příklady vysvětlené dále u funkcí
par(mar=c(2.2,3,0,0),las=1)
plot(0.3,0.3,xlim=c(0,1),ylim=c(0,1),pch=16,cex=4,col="brown")
points(0.5,0.5,pch=16,cex=4,col="lightblue")
points(0.1,0.5,pch=16,cex=4,col=colors()[9])
points(0.6,0.6,pch=16,cex=4,col=rgb(0.1,0.3,0.2))
points(0.5,0.1,pch=16,cex=4,col=gray(0.3))
points(0.5,0.1,pch=16,cex=4,col="#FFD39B")#rrggbb v 16-kové soustavě
```

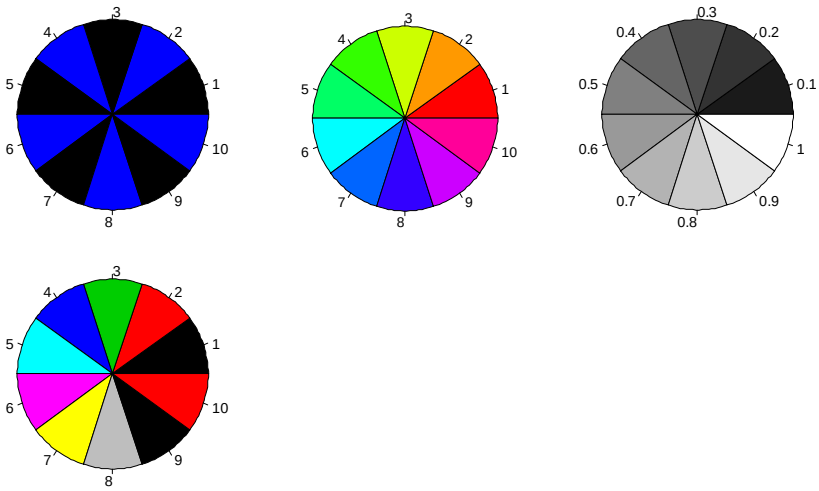
- **`colors()`** – vypisuje vektor názvů dostupných barev. Tyto názvy lze použít přímo pro nastavování barev.

```
colors()[1:15] #vypíše seznam nastavených názvů barev (0-657)
 [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
 [5] "antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
 [9] "aquamarine1"    "aquamarine2"    "aquamarine3"    "aquamarine4"
[13] "azure"          "azure1"         "azure2"
```

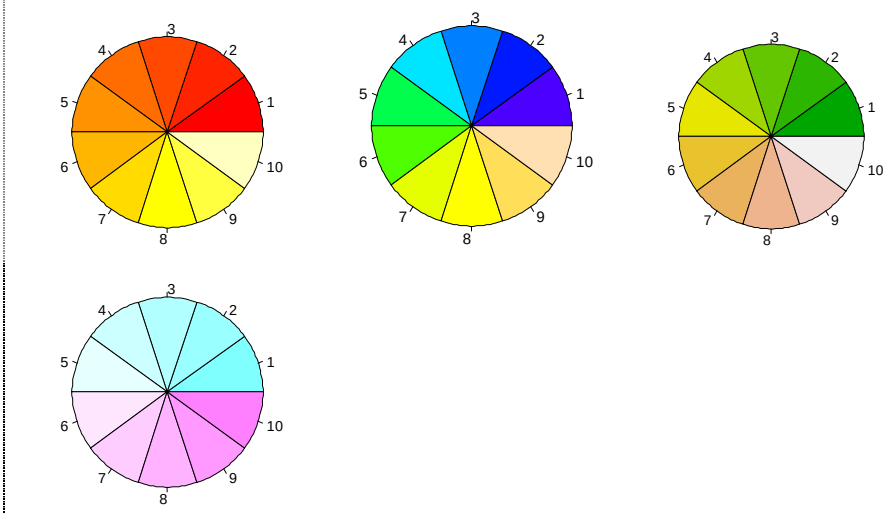
- **`palette`**`(value)` – vypisuje nebo nastavuje aktuální využívanou paletu barev (nastavení pomocí vektoru *`value`*
- **`gray`**`(value)` – vytváří paletu odstínu šedé v intervalu 0 – 1, podle zadaného vektoru
- **`rainbow`**`(n,s=1,v=1,start=0,end=max(1,n - 1)/n,gamma=1,alpha=1)` – vytváří paletu n barev (podle duhového spektra) v rozlišení *n*, nasycení (*s*), odstínu (*v*), s gamma korekcí (*gamma*) a průhledností *`alpha`* (viz níže u hsv). Podobné funkce:

```
heat.colors(n, alpha = 1)
terrain.colors(n, alpha = 1)
topo.colors(n, alpha = 1)
cm.colors(n, alpha = 1)
```

```
palette() #aktuální barevná paleta pro argument typu col a bg
[1] "black"   "red"     "green3"  "blue"    "cyan"    "magenta" "yellow"
[8] "gray"

par(mar=c(0.1,0.1,0.1,0.1))
#graf vlevo
palette(c("black","blue"))#nastavení palety na černou a modrou
pie(rep(1,10),col=1:10,lab=1:10)#graf opakuje pouze 2 nastavené barvy
#graf druhý zleva
palette(rainbow(10))
pie(rep(1,10),col=1:10,lab=1:10)
#graf třetí zleva
palette(gray(seq(0.1,1,0.1)))
pie(rep(1,10),col=1:10,lab=seq(0.1,1,0.1))
#graf vpravo
palette("default")#nastavení palety zpět na standardní
pie(rep(1,10),col=1:10,lab=1:10)#graf opakuje pouze 2 nastavené barvy
#alternativně lze první tři grafy vykreslit beze změny celé palety následovně
palette("default")
pie(rep(1,10),col=c(1,4),lab=1:10)#nebo col=c("black,"blue")
pie(rep(1,10),col=rainbow(10),lab=1:10)
pie(rep(1,10),col=gray(seq(0.1,1,0.1)),lab=seq(0.1,1,0.1))
```

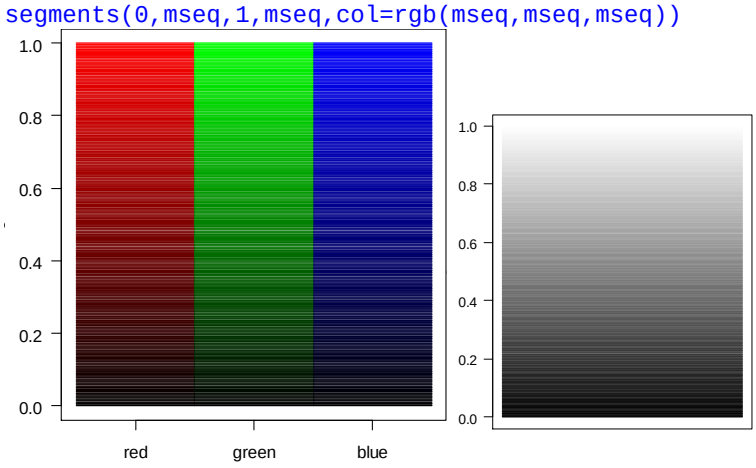



```
pie(rep(1,10),col=heat.colors(10),lab=1:10)
pie(rep(1,10),col=topo.colors(10),lab=1:10)
pie(rep(1,10),col=terrain.colors(10),lab=1:10)
pie(rep(1,10),col=cm.colors(10),lab=1:10)
```

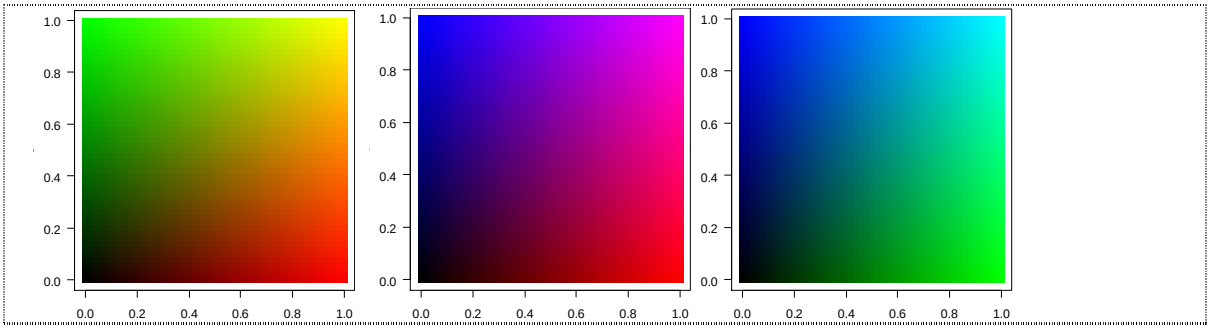


- **`rgb`**`(red,green,blue,alpha,names=NULL,maxColorValue=1)` – namíchá barvu jako kombinaci odstínů červené, zelené a modré (udávají se v rozsahu *maxColorValue*, standardně 0-1) – vrací řetězec – číslo v 16-kové soustavě
  `alpha` – průhlednost (0 = průhledná, 1 = neprůhledná)
  `names` – názvy barev
  `maxColorValue` – maximum, kterého nabývá základní barva (r-g-b)

```
rgb(0,1,1) #vrátí řetězec – číslo barvy v 16-kové soustavě
[1] "#00FFFF"
#míchání dvojic barev (r-g,r-b,g-b)
par(mar=c(2.2,3,0,0),las=1)
plot(0,0,xlim=c(0,0.6),ylim=c(0,1),type="n",xaxt="n")
mseq<-seq(0,1,0.002)
segments(0,mseq,0.2,mseq,col=rgb(mseq,0,0))
segments(0.2,mseq,0.4,mseq,col=rgb(0,mseq,0))
segments(0.4,mseq,0.6,mseq,col=rgb(0,0,mseq))
axis(1,at=c(0.1,0.3,0.5),labels=c("red","green","blue"))
plot(0,0,xlim=c(0,1),ylim=c(0,1),type="n",xaxt="n")
segments(0,mseq,1,mseq,col=rgb(mseq,mseq,mseq))
```




```
#míchání dvojic barev (r-g,r-b,g-b)
plot(0,0,xlim=c(0,1),ylim=c(0,1),type="n")
xcol<-rep(mseq,each=51)
ycol<-rep(mseq,51)
points(xcol,ycol,col=rgb(xcol,ycol,0),pch=15)
points(xcol,ycol,col=rgb(xcol,0,ycol),pch=15)
points(xcol,ycol,col=rgb(0,xcol,ycol),pch=15)
```



- **col2rgb**`()` – vrací rozbor barvy do jednotlivých základních barev RGB

```
mycol<-col2rgb(colors()[1:4])
colnames(mycol)<-colors()[1:4]
mycol
      white aliceblue antiquewhite antiquewhite1
red     255       240          250           255
green   255       248          235           239
blue    255       255          215           219
```

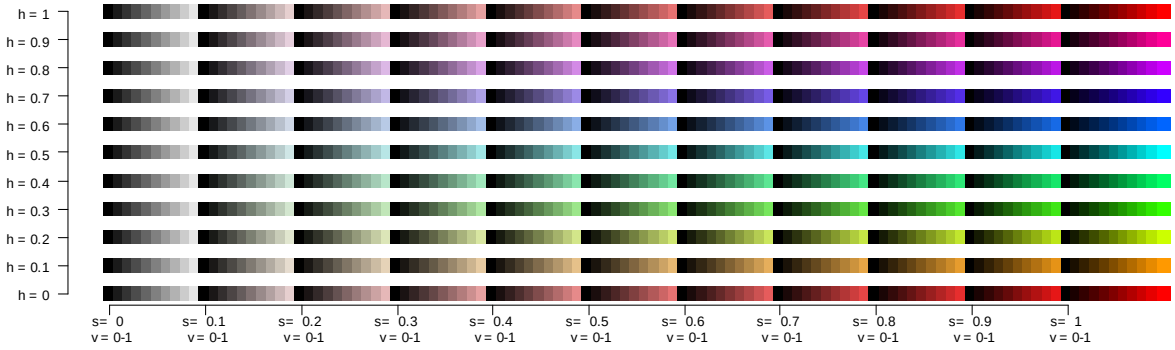
- **hsv** `(h=1,s=1,v=1,gamma=1,alpha)` – namíchá barvu podle barvy (hue, *h*), sytosti (saturation, *s*) a odstínu (value, *v*), dále lze barvy měnit pomocí gamma korekce (*gamma*) a průhlednosti (*alpha*), hodnoty se zadávají v rozsahu 0 – 1. Vrací řetězec – číslo v 16-kové soustavě

  alpha – průhlednost (0 = průhledná, 1 = neprůhledná)

  names – názvy barev

  maxColorValue – maximum, kterého nabývá základní barva (r-g-b)

```
#vykreslí graf znázorňující možnost nastavení barvy pomocí hsv
par(mar=c(3.1,3.7,0.1,0.1))
mtb<-data.frame(h=rep(0:10,each=121),s=rep(rep(0:10,each=11),11),v=rep(0:10,121))
plot(0,0,xlim=c(0,110),ylim=c(0,10),type="n",axes=F,ann=F)
points(x=mtb$s*10+mtb$v,y=mtb$h,pch=15,col=hsv(h=mtb$h/10,s=mtb$s/10,v=mtb$v/10),
cex=2)
axis(2,at=0:10,labels=paste("h =",0:10/10),las=1,cex.axis=0.8)
axis(1,at=seq(0,100,10),labels=paste("s= ",0:10/10,"\n","v = 0-1"),cex.axis=0.8)
```



- **rgb2hsv** `(r,g=NULL,b=NULL,gamma=1,maxColorValue=255)` – převede RGB barvu (zadanou jako řetězec nebo jako r,g,b) do systému HSV.

- **colorRamp**`(colors,bias=1,space=c("rgb","Lab"),interpolate=c("linear", "spline"))` – interpoluje barevnou paletu z vektoru zadaných barev, vhodné pro výpočet barevných přechodů, vrací RGB kód barvy

  colors – vektor barev mezi kterými se mají vytvořit přechody

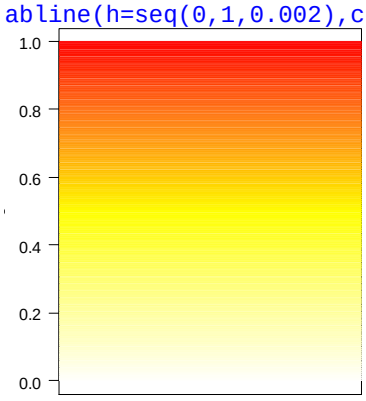
  bias – nahloučení barev směrem k "horní barvě" (0–1)

  space – v jakém systému se vytvářejí barvy

  interpolate – možnost lineární nebo spline interpolace

```
#funkce vytvoří 501 barev mezi bílou žlutou a červenou
par(mar=c(2.2,3,0,0),las=1)
```

```
mseq<-seq(0,1,0.02)
plot(0,0,xlim=c(0,1),ylim=c(0,1),type="n",xaxt="n")
abline(h=seq(0,1,0.002),col=colorRampPalette(c("white","yellow","red"))(501))
```

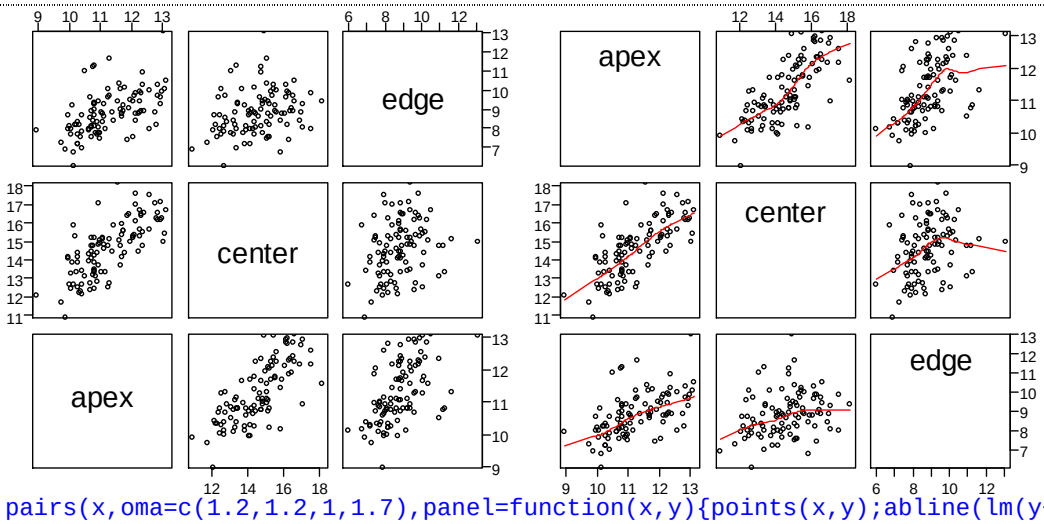



```
colorRampPalette(c("white","red"))(10)
[1] "#FFFFFF" "#FFE2E2" "#FFC6C6" "#FFAAAA" "#FF8D8D" "#FF7171" "#FF5555"
[8] "#FF3838" "#FF1C1C" "#FF0000"
```

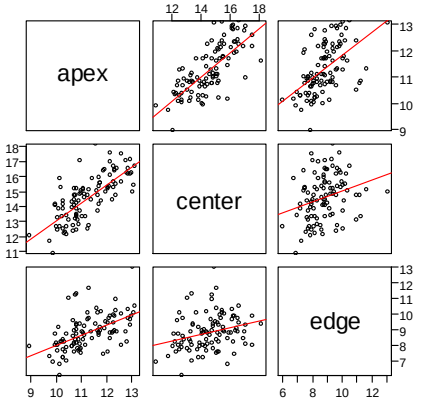
Speciální typy grafů

- **pairs**(formula,data,…,subset,na.action) nebo **pairs**(x,labels,panel,…, lower.panel,upper.panel,diag.panel,text.panel,label.pos,cex.labels, font.labels,row1attop=T,gap) – vytvoří matici xy grafů kde hodnoty x a y postupně tvoří všechny sloupce dané tabulky (table, array, matice,vektory). Data jsou zadávána buď pomocí vzorce (formula) nebo přímo jako tabulka dat (data frame). Graf je velice dobře využitelný při základní analýze závislosti několika proměnných (např. při mnohorozměrných metodách). Graf vykresluje také proměnnou typu factor, kde jednotlivé hodnoty jsou postupně zobrazovány jako přirozená čísla (1, 2, 3). Funkce má několik užitečných argumentů (panel, lower.panel, ...), které umožňují dokreslovat speciální křivky nebo další grafy přímo do jednotlivých panelů (včetně diagonálních).
  subset – z tabulky data je možné vybrat podmnožnu podle zadaného kritéria
  na.action – jak pracovat s NA hodnotami na.omit, na.fail, na.exclude, na.pass
  row1attop – mají grafy x~x začít první řádkem diagonály, nebo diagonála začíná dole.
  gap – velikost mezery mezi panely.
  panel, lower.*panel*,*upper.panel*,*diag.panel*,*text.panel* – funkce, které se provedou při vykreslování jednotlivých panelů
  label.pos – pozice textu v diagonálním panelu.
  cex.labels – velikost textu v diagonálním panelu.
  font.labels – font textu v diagonálním panelu (viz font pro par).

```
#studujeme závislost množství taninu v jednotlivých částech listů
# použitá data jsou vytvořena jako náhodná čísla z normálního rozdělení
apex<-10+rnorm(100,1,1);center<-apex*1.2+rnorm(100,1,1);edge<-0.8*apex+rnorm(y)
x<-data.frame(apex,center,edge)#tabulka z vygenerovaných dat
pairs(x,data=x,oma=c(1.2,1.2,1,1.7),row1attop=F)
pairs(x,oma=c(1.2,1.2,1,1.7),panel=panel.smooth,label.pos=0.8)#křivka smooth
```



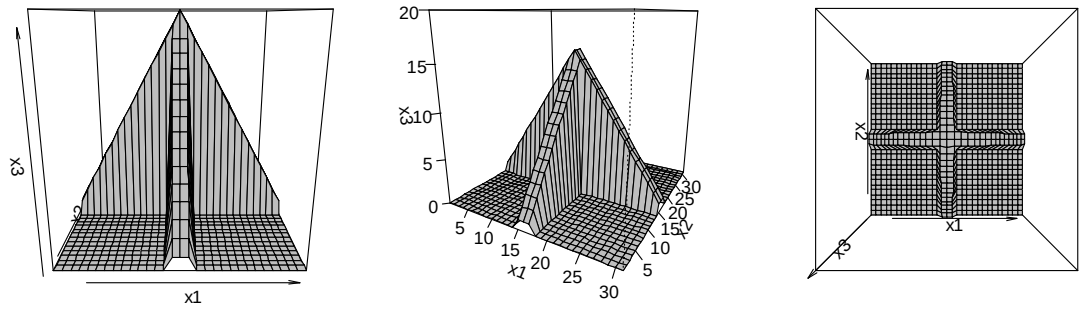

```
pairs(x,oma=c(1.2,1.2,1,1.7),panel=function(x,y){points(x,y);abline(lm(y~x),col=2
)}) #použití funkce pro vykreslení regresních přímek závislosti xy
```



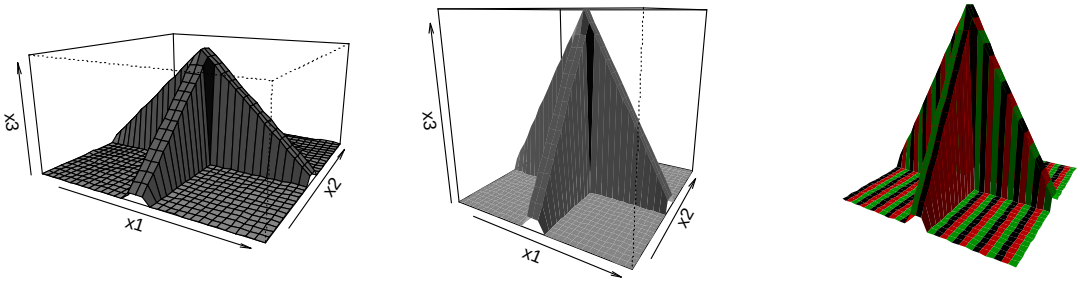
- **persp**(x,y,z,xlim,ylim,zlim,xlab,ylab,zlab,main,sub,theta,phi,r,d,scale =T,expand,col,border,ltheta,lphi,shade,box=T,axes=T,nticks,ticktype,…) – 3D povrchový graf, hodnoty x a y jsou řazeny vzestupně a jejich počet je shodný s počtem řádků a sloupců matice pro koordináty z. Další trojrozměrné grafy lze nalézt v doplňkových knihovnách (např. misc3d, lattice, scatterplot3d)

  xlim,ylim,zlim – interval vykreslování hodnot na ose x, y a z.
  main,sub,col,border,box,axes – podobné jako u ostatních grafů.
  theta, phi – horizontální a vertikální úhel pootočení grafu.
  r – vzdálenost oka od středu grafu, ovlivňuje úhly horní a dolní plochy)
  d – účinnost efektu perspektivy.
  scale – v případě rozdílného rozsahu os přepočítá nebo ponechá osy v původní proporci.
  expand – zvětší nebo zmenší poměr osy z k ostatním osám.
  shade, ltheta, lshade – stínování (0-1) a nastavení úhlu stínování grafu.
  box, axes – vypnutí celého ohraničení grafu a os.
  ticktype,nticks – typ os ("simple" a "detailed") a přibližní počet značek na osách.

```
x1<-1:31;x2<-1:31;x3<-matrix(0,31,31)#nastavení souřadnic x,y,z
x3[15:17,]<-rep(c(1:16,15:1),each=3)
x3[,15:17]<-c(1:16,15:1) #vytvoření prostorového objektu
par(mar=c(1,2,0.2,0.2))
persp(x1,x2,x3,col=8) #vykreslení objektu bez rotace verikální a horizontální
persp(x1,x2,x3,zlim=c(0,20),theta=30,col=8,ticktype="detailed")#podél vertikály
persp(x1,x2,x3,zlim=c(0,20),phi=90,col=8)#rotace podél horizontály
```

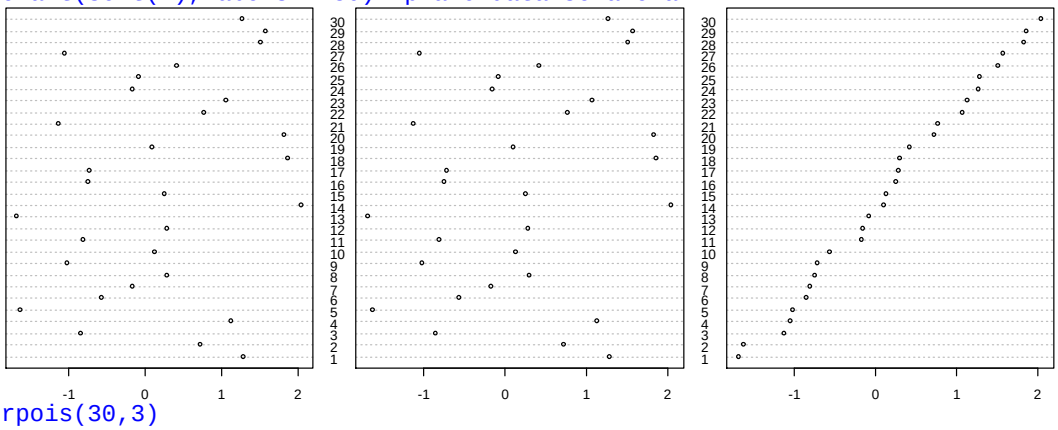



```
persp(x1,x2,x3,th=30,col=8,shade=0.5,expand=0.5)#stínování a expanze z
persp(x1,x2,x3,th=30,col=8,shade=0.5,border=NA) #vypnutí mřížky
persp(x1,x2,x3,th=30,col=1:3,shade=0.5,box=F,border=NA) #vypnutí os
```

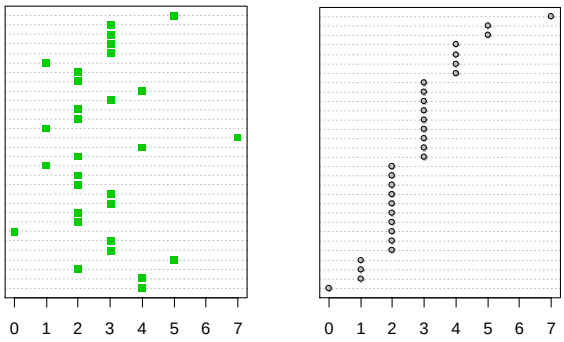


- **dotchart**(x,labels=NULL,groups=NULL,gdata=NULL,cex,pch=21,gpch=21,bg, color,gcolor,lcolor,xlim,main=NULL,xlab=NULL,ylab=NULL,…) – Clevelandův bodový graf, vynáší každou hodnoty na osu x a pořadí hodnoty na osu y. Využíván je zejména pro základní exploratorní analýzu dat.
  labels – označuje popisky na ose y
  groups – faktor označující skupiny podle kterých se vykreslí více grafů
  gdata – hodnota zobrazující se na speciální lince nad grafem (obvykle medián nebo průměr)
  lcolor – barva horiz. linek
  gcolor, gpch – barvy a typy bodů pro skupiny dat

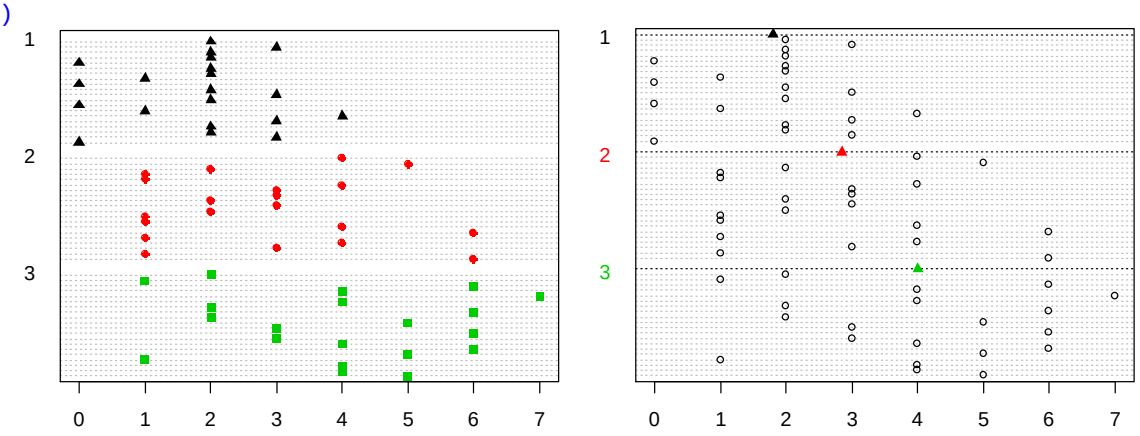
```
x<-rnorm(30)
round(x,2)
 [1]  1.29  0.72 -0.85  1.13 -1.63 -0.57 -0.17  0.30 -1.01  0.13 -0.81  0.29
[13] -1.69  2.05  0.26 -0.75 -0.72  1.87  0.10  1.84 -1.13  0.78  1.07 -0.16
[25] -0.08  0.42 -1.05  1.51  1.58  1.28
dotchart(x)#vlevo vynáší náhodná čísla z norm. rozdělení
dotchart(x,labels=1:30)#uprostřed přidána osa y (pořadí hodnoty x)
dotchart(sort(x),labels=1:30)#vpravo data seřazena
```



```
x<-rpois(30,3)
dotchart(x,pch=15,col=3)#vlevo
dotchart(sort(x),pch=21,bg=8)#vpravo
```

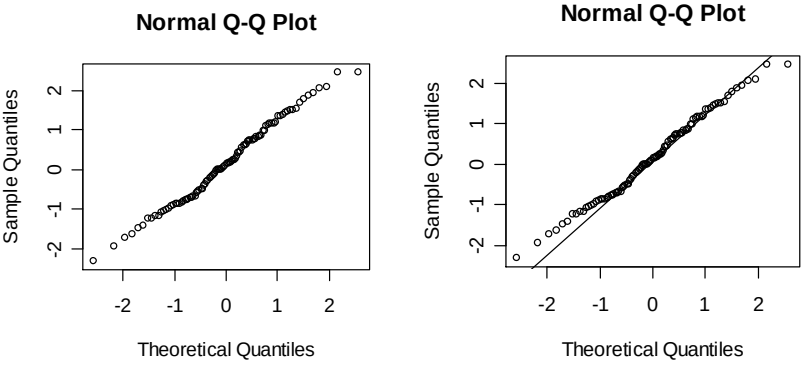


```
#data lze rozdělit do samostatných skupin (nejlépe podle faktoru) (graf vlevo)
x<-data.frame(gr=rep(1:3,each=20),hod=c(rpois(20,2),rpois(20,3),rpois(20,4)))
dotchart(x$hod,groups=factor(x$gr),col=x$gr,pch=14+x$gr)
#u skupin lze počítat určité charakteristiky a umístit je na spec. osu (vpravo)
dotchart(x$hod,groups=factor(x$gr),gdata=tapply(x$hod,x$gr,mean),gcol=1:3,gpch=17
)
```



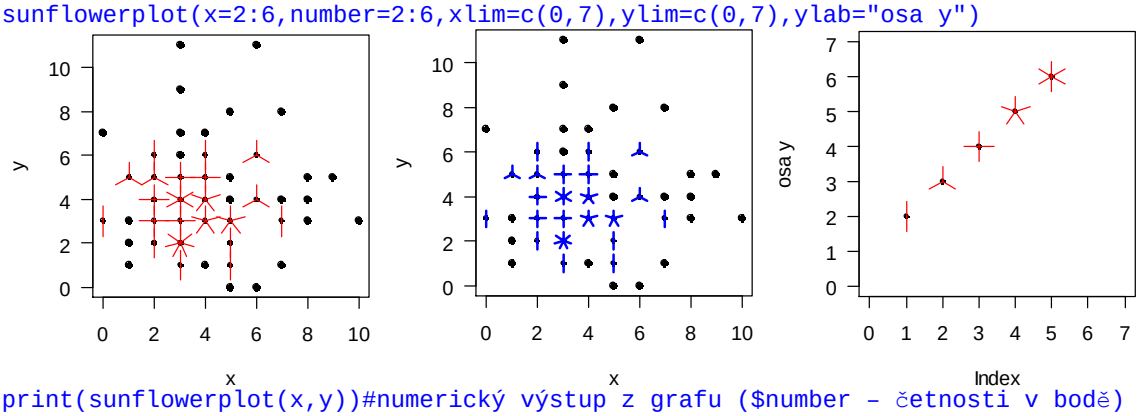
- **qqnorm** (y,ylim,main="Normal Q-Q Plot",xlab="Theoretical Quantiles", ylab="Sample Quantiles",plot.it=T,datax=F,...), **qqline** (y,...), **qqplot** (x,y,plot.it=T,xlab,ylab, ...) – graficky porovná kvantily variační řady y s teoretickými kvantily normálního rozdělelní (qqnorm), popř. přidá přímku očekávaného tvaru grafu v případě normality dat (qqline) nebo porovná kvantily dvou variačních řad. Grafy jsou využívány právě při ověřování normality dat.
  plot.it – logická, mají být výsledky vykresleny nebo vypsány v tabulce?
  plotx – prohození osy x a y (teoretické kvantily na ose x datax=TRUE)

```
x<-rnorm(100)#100 náhodných čísel z normovaného normálního rozdělení
qqnorm(x)#základní graf
qqline(x) #přímka označuje teoretické rozmístění dat v případě normality
```



- **`sunflowerplot`**`(x,y,number,log,digits,add=F,rotate=F,pch,cex,cex.fact, size,seg.col,seg.lwd,…)` – jedná se o xy-bodový graf (slunečnicový), ve kterém se překrývající body zobrazují s „okvětními“ lístky podle počtu hodnot v daném bodě
  `number` – počet opakování v daném bodě lze zadat také pomocí tohoto argumentu
  `digits` – jestliže jsou body blízko sebe, na kolik desetinných míst je možné považovat shodu
  `rotate` – logical, náhodné rotování okvětních lístků (aby se zabránilo překryvům)
  `size, seg.col, seg.lwd` – délka, barva a tloušťka "okvětních lístků"

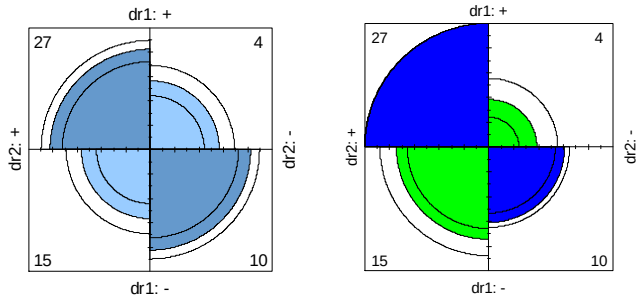
```
x<-rpois(100,4)
y<-rpois(100,4)
par(mar=c(4.1,4.1,0.1,0.1),las=1)
sunflowerplot(x,y)#graf, kde souřadnice jsou náhodná čísla z Poiss. rozdělení
sunflowerplot(x,y,size=0.07,seg.lwd=2,seg.col=4)#graf uprostřed
# nastavení opakování přímo pomocí number
sunflowerplot(x=2:6,number=2:6,xlim=c(0,7),ylim=c(0,7),ylab="osa y")
```




```
print(sunflowerplot(x,y))#numerický výstup z grafu ($number - četnosti v bodě)
$x
 [1] 0 1 1 1 1 2 2 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 4 5 5 5 5 5 5 5 6 6 6 7
[39] 7 7 7 7 7 8 8 8 8 9

$y
 [1] 3 2 4 5 6 1 2 3 4 5 6 7 8 9 2 3 4 5 6 7 0 2 3 4 5 6 8 0 1 2 3 4 5 6 3 5 6 1
[39] 2 3 4 5 6 2 3 5 6 5

$number
 [1] 1 3 1 1 1 1 2 2 2 1 3 2 2 1 4 4 6 2 3 2 2 3 3 2 6 1 2 1 1 1 2 5 2 5 3 3 1 1
[39] 1 2 1 1 2 1 1 1 1 1
```

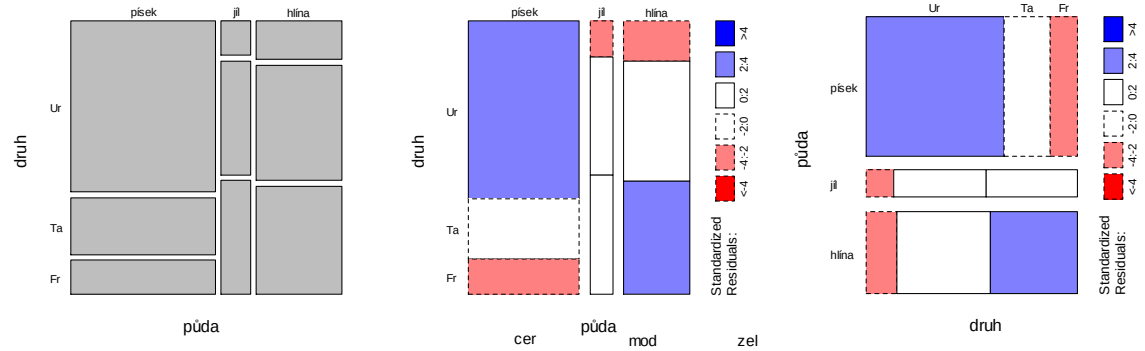
- **`fourfoldplot`**`(x,color,conf.level=0.95,std,margin=c(1,2),space=0.2,main= NULL,mfrow=NULL,mfcol=NULL)` – vytvoří „čtyřlístkový graf“. Používá se ke znázornění vztahů v čtyřpolní tabulce, přičemž konfidenční intervaly určují, zda jsou rozdíly významné.
  `x` – data ve formě matice nebo pole (popisky grafu vychází z popisu dimenzí pole nebo matice).
  `color` – barevné rozlišení vektorem c(b1,b2), standardně `c("#99CCFF", "#6699CC")`
  `conf.level` – vykreslí konfidenční interval pro danou kombinaci faktorů
  `std` – standardizace dat (`"margins", "ind.max", "all.max",` popř. první písmeno) – zda bude tabulka vyvářená po úhlopříčkách nebo zda každý z kvadrantů bude vyjadřovat četnosti
  `margins` – určuje, zda bude graf standardizován podle sloupců (2), řádků (1) nebo obojí (c(1,2)), má význam jen když je `std = "m"`
  `space` – velikost písma
  `mfrow,mfcol` – vektor rozdělení grafické plochy na rows x columns a vykreslení více grafů najednou

```
# na 56 lokalitách byla sledována přítomnost dvou druhů, podle toho byla
# vytvořena tabulka závislosti druhů na sobě
x<-array(c(27,15,4,10),dim=c(2,2),dimnames=list(dr1=c("+","-"),dr2=c("+","-")))
fourfoldplot(x)
fourfoldplot(x,std="i",color=c("green","blue"))
```

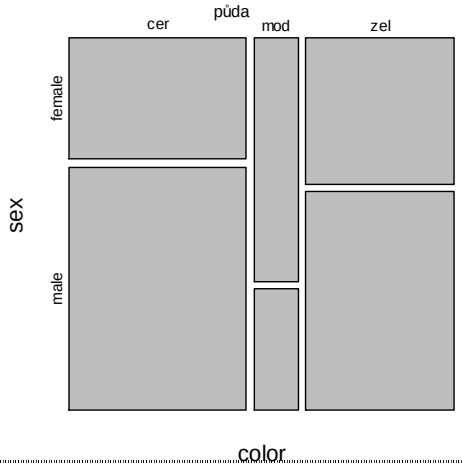


- **mosaicplot**(x,main,sub,xlab,ylab,sort,off,dir,color=F,shade=F,margin, cex.axis,las,type,…) nebo **mosaicplot**(formula,data,subset, na.action) – vytvoří graf s obdélníky o stranách podle velikosti vzorku v jednotlivých skupinách. Využívá se pro názorné zobrazení vztahů v kontingenčních tabulkách.
  off – vektor relativních velikostí mezer mezi obdélníky (dle počtu dimenzí)
  dir – vektor směru vynášení ("v" – vertikální, "h" – horizontální)
  color – vektor barev nebo logická hodnota, zda mají být použity barvy pro jednotlivé obdélníky
  shade – vektor barev intervalů pro odlišení barvami nebo logická hodnota, zda má být stínováno
  type – typ residuálů, které se mají zobrazit při barevném rozlišení
  margin – seznam vektorů s okrajovými součty pro loglineární model (viz loglin)
  sort – vektor permutací řazení jednotlivých proměnných (permutace 1:počet dimenzí)

```
myt<-
array(c(75,3,10,25,10,30,15,10,28),dim=c(3,3),dimnames=list("půda"=c("písek",
       "jíl","hlína"),druh=c("Ur","Ta","Fr")))#pole - kontingenční tabulka
myt #tabulka závislosti typu půdy na výskytu rostliny
       druh
půda    Ur Ta Fr
  písek 75 25 15
  jíl    3 10 10
  hlína 10 30 28

mosaicplot(myt,main=NULL,shade=F,las=1)
mosaicplot(myt,main=NULL,shade=T,las=1,off=c(10,0))#mezery a barvy
mosaicplot(myt,main=NULL,shade=T,las=1,off=c(10,0),dir=c("h","v"))
```



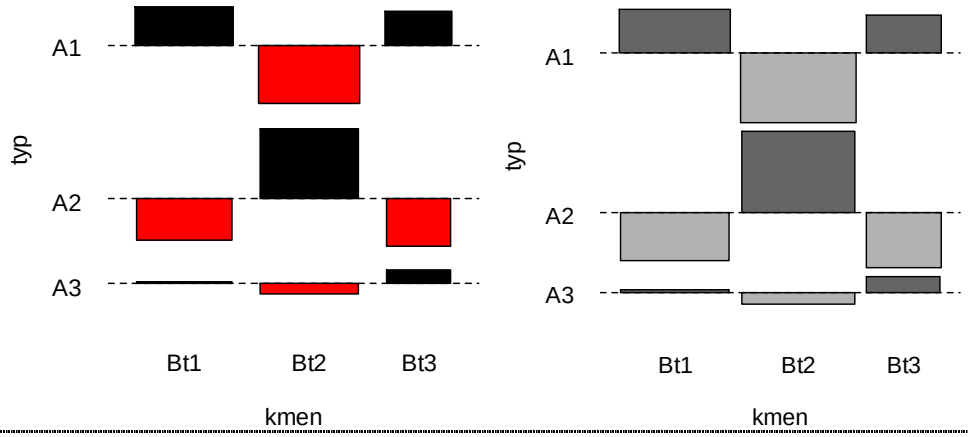

```
#jiný typ dat pro načtení
   color    sex
1    zel   male
2    cer   male
3    mod female
4    mod female
5    mod   male
6    cer female
7    cer   male
8    cer   male
9    cer   male
10   cer   male
11   cer   male
12   cer female
13   cer female
```



```
14   zel   male
15   zel   male
16   zel   male
17   zel   male
18   zel female
19   zel female
20   zel female
21   zel female
22   zel   male
23   cer   male
24   cer   male
25   cer female

mosaicplot(~color+sex,data=x,main=NULL)#výpis stylu formula
```

- **assocplot**(x, col, space, main, xlab, ylab) – graficky zobrazuje vztahy mezi očekávanou hodnotou a skutečnou hodnotou v kontingenčních tabulkách. V případě, že je hodnota vyšší než očekávaná, pak zakreslí obdélník černě, v opačném případě červeně, barvy lze nastavit jako dvouprvkový vektor v *col*,
  space – prostor mezi jednotlivými složkami

```
# ke třem kmenům bakterie E. coli (Bt1, Bt2, Bt3) byly přidávány tři druhy
# antibiotika (pro každou kombinaci 30 opakování) a určeno na kolika plotnách
# kmeny zanikly.
myt<-as.table(array(c(25,16,12,16,29,6,20,20,10),c(3,3),
     list(kmen=c("Bt1","Bt2","Bt3"),typ=c("A1","A2","A3"))))
myt #trida objektu tabulka, vytvořena z 2-dimenzionálniho pole (array)
     typ
kmen  A1 A2 A3
  Bt1 25 16 20
  Bt2 16 29 20
  Bt3 12  6 10
par(mar=c(4.1,4.1,0.1,0.1),las=1)
assocplot(myt)
assocplot(myt,space=0.1,col=gray(c(0.4,0.7)))
```

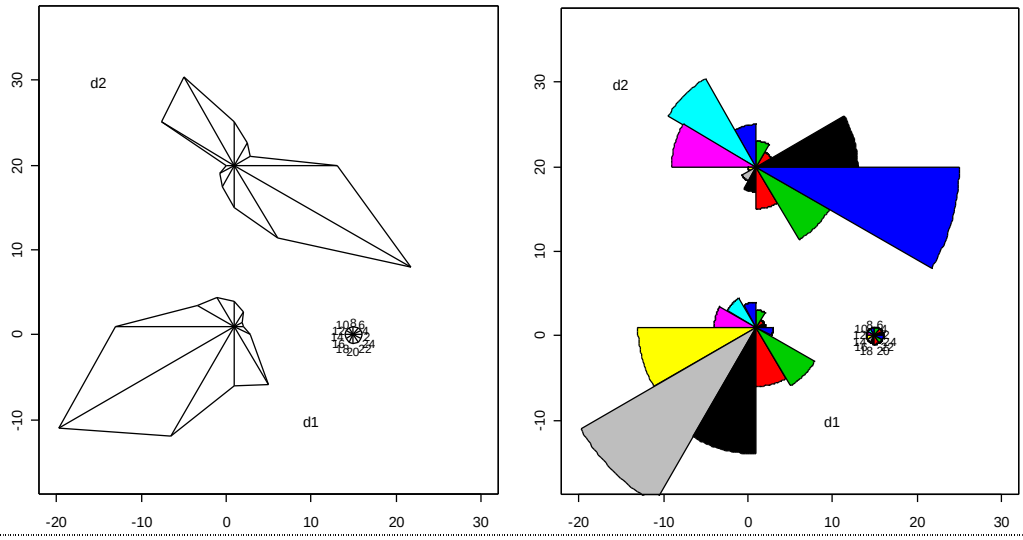


- **stars**(x,full=T,scale=T,radius=T,labels,locations,nrow,ncol,len,key.loc, key.labels,key.xpd=T,xlim,ylim,flip.labels,draw.segments=F,col.segments, col.stars,axes=F,frame.plot,main,sub,xlab,ylab,cex,lwd,lty,xpd,mar, add=F,plot=T,…) – vytvoří kruhový (hvězdicový diagram), kde každý řádek je jeden graf a každý sloupec jeden úsek kruhu
  full – jestliže je T, pak se jedná o kruhový graf, F – půlkruhový
  scale – T – všechny hodnoty jsou vztaženy relativně ve sloupci podle minimální=0 a maximální=1 hodnoty (u příkladu je proto vypnuto, protože jsou jen dva řádky, tedy jen 0 nebo 1), v případě nastavení F je relativní výpočet vztažen k řádku
  radius – jsou vykreslovány výseče grafu, jestliže ne, pak jen obrysy
  labels – popisy jednotlivých grafů (v příkladu je to d1 a d2)

locations – umístění grafů v prostoru (matice, kde v řádcích jsou souřadnice x a y)
nrow, *ncol* – počet řádků a počet sloupců grafů
len – škálovací faktor, poměrná (délka výseků) např. 2 – 2x zvětšeno
key.loc – lokace klíče s vysvětlením hodnot jednotlivých úseků
key.labels– označení názvů jednotlivých úseků (cípů)
key.xpd – stejné jako *xpd* v par (přichycení klíče)
xlim,*ylim* – extrémy oxy x a y
flip.labels – uchycení názvů grafu (není podrobně vysvětleno)
draw.segmets – vykreslení segmentů a ne hvězdic
col.segments – barvy jednotlivých segmentů
col.stars – barvy jednotlivých hvězdic

Výskyt dvou druhů během 24 hodin umístíme do objektu myt

```
     2   4   6   8   10    12    14    16    18    20    22    24
d1   1   1   2   3    4     5    14    24    15     7     8     2
d2  12   2   3   5   12    10     1     2     3     5    10    24
stars(myt,full=T,locations=matrix(c(1,1,1,20),2,2),scale=F,axes=T,xlim=c(-
20,30),ylim=c(-10,30),key.loc=c(15,0),labels=NULL)
text(c(-15,10),c(30,-10),c("d2","d1"))

stars(myt,full=T,locations=matrix(c(1,1,1,20),2,2),scale=F,draw.segments=T,axes=T
,xlim=c(-20,30),ylim=c(-10,30),key.loc=c(15,0),labels=NULL)
text(c(-15,10),c(30,-10),c("d2","d1"))
NULL
```



- **coplot**(formula, data, given.values, panel, rows, columns, show.given=T, col, pch, bar.bg, xlab, ylab, subscripts=F, axlabels, number, overlap, xlim, ylim, ...) – vytváří xy graf závislosti dvou proměnných rozdělených podle určité úrovně (vyjádřeno rovnicí)

formula – vzorec závislosti (např. „y ~ x | a“ závislost y na x rozděleno podle a, lze přidat a*b takže závislost na dvou proměnných), podle množství úrovní jednotlivých faktorů se pak vytvoří daný počet grafů závislostí
data – která tabulka dat byla použita
given.values – seznam nebo hodnota faktoru, podle kterého se budou data dělit
panel – funkce (x,y,pch,col),která má být použita v každém grafu např. points, lines, panel.smooth apod.
rows,*columns* – počet jednotlivých řádků a sloupců pro výsledné grafy
show.given – logická hodnota nebo vektor, zda se mají zobrazovat okrajové grafy
bar.bg – numerický vektor barev pro jednotlivé sloupce v okrajových grafech
xlab, ylab – názvy jednotlivých os grafů (hodnota nebo dvousložkový vektor)
subscripts – logická hodnota, T – do funkce panel se přidá argument subscripts zadávající podmínky pro data odpovídající panelu grafu
axlabels – funkce vytvářející značky na ose x nebo y, když se jedná o faktory

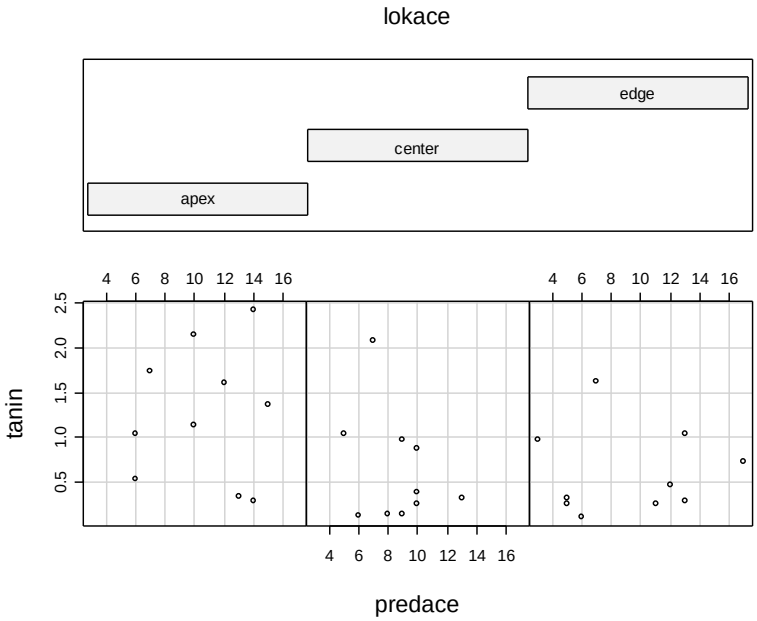
number – počet kategorií v případě, že jedna z proměnných podle kterých se data člení není faktor (v případě, že obě nejsou, pak dvouprvkový vektor)

overlap – překryv jednotlivých kategorií proměnných ($0 \leq overlap < 1$)

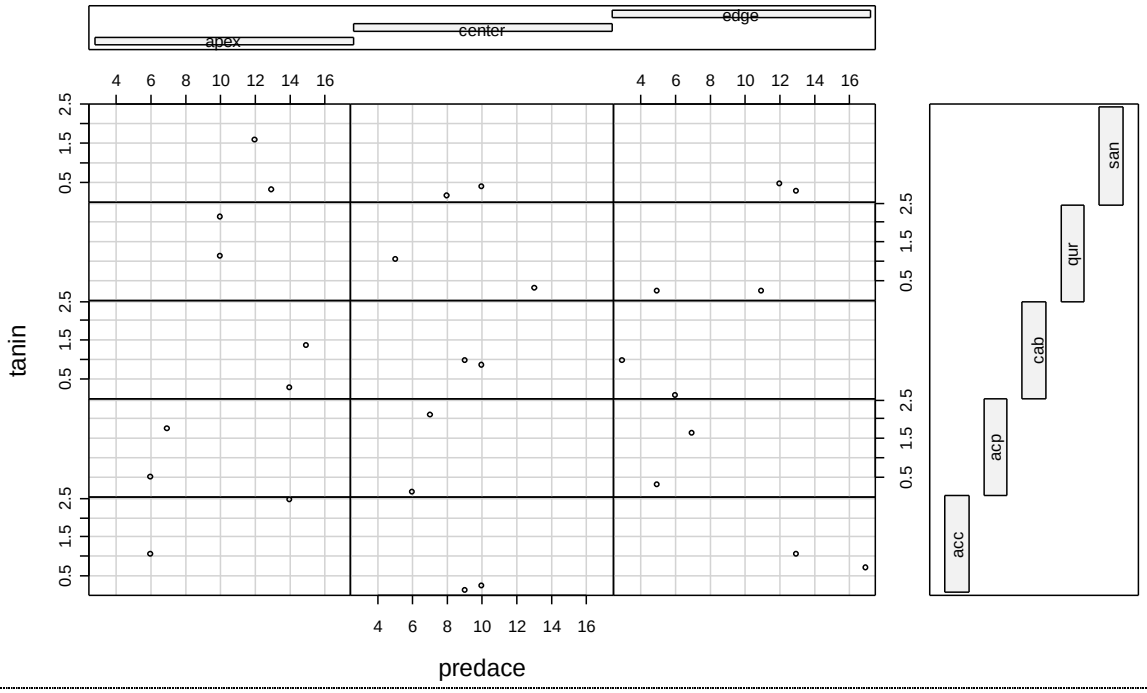
xlim,ylim – rozsah hodnot na ose x a y

Příklad: je dána tabulka s hodnotami měření množství taninů v listu a stupněm predace na listech (v tomto případě byly hodnoty generovány náhodně), výzkumy probíhaly vždy na třech lokacích na listu (center, edge, apex), zkoumáme, zda existují na jednotlivých lokacích závislosti.

```
myt<-data.frame(lokace=rep(c("center","edge","apex"),10),tanin=abs(rnorm(30)),
pred=rpois(30,10))
coplot(tanin ~ pred | lokace, data = myt,rows=1,xlab=c("predace","lokace"))
```

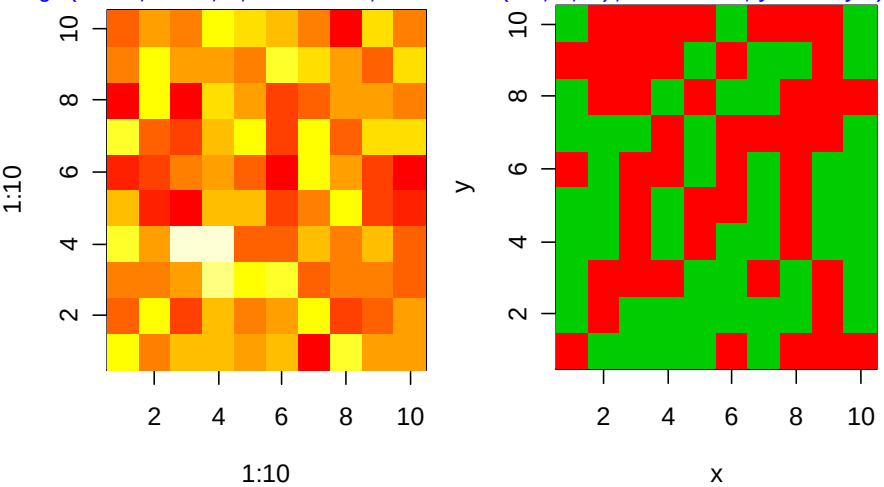



```
myt<-cbind(myt,druh=rep(c("acc","acp","cab","san","qur"),6)) #pridame druhy
coplot(tanin ~ pred|lokace*druh, xlab=c("predace",""),ylab=c("tanin",""),
data=myt)
```

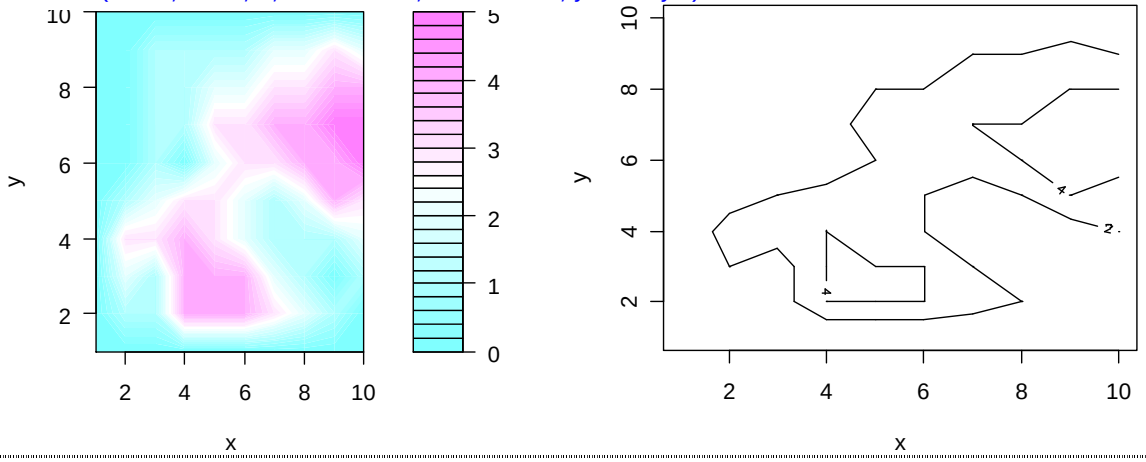


- **`image`** `(x,y,z)` – graf vytvářející speciální obdélníkové oblasti různé barvy podle matice *z*, *x* a *y* jsou vektory nastavující středy obdélníků na ose x a y. Podobně fungují grafy **`contour`** a **`filled.contour`**, které mají podobné zadání dat.

```
par(mar=c(4.1,4,0.1,0.1))
#zobrazí matici 100 náhodných čísel z normálního rozdělení, rozdělených do
#10 řádků a 10 sloupců
image(1:10,1:10,matrix(rnorm(100),10,10))
# stejné, ale 100 náhodných čísel z rovnoměrného rozdělení od -1 do +1
#kladná čísla zelená, záporná červená (viz col a breaks)
x<-matrix(runif(100,-1,1),10,10)
image(1:10,1:10,x, col=2:3,breaks=c(-1,0,1),xlab="x",ylab="y")
x<-matrix(runif(100,-1,1),10,10)
image(1:10,1:10,x, col=2:3,breaks=c(-1,0,1),xlab="x",ylab="y")
```




```
# načteme následující data do proměnné x1
0 0 0 0 0 0 0 0 0 0 0 1 1 4 4 4 3 2 1 0 0 2 1 4 4 4 2 1 0 1 0 3 3 4 3 2 1 1 1 2 0
1 2 3 3 2 1 2 4 3 0 0 1 0 2 3 3 4 4 5 0 0 1 1 3 3 4 4 5 5 0 0 1 1 2 2 3 3 4 4 0 0
1 1 1 1 2 2 3 2 0 0 0 0 0 0 0 0 0 0
x<-matrix(x1,10,10) # tvorba matice z načtených dat
x #matice jejíž řádky budou na ose x, sloupce na ose y
      col1 col2 col3 col4 col5 col6 col7 col8 col9 col10
 [1,]    0    0    0    0    0    0    0    0    0     0
 [2,]    0    1    2    3    1    0    0    0    0     0
 [3,]    0    1    1    3    2    1    1    1    1     0
 [4,]    0    4    4    4    3    0    1    1    1     0
 [5,]    0    4    4    3    3    2    3    2    1     0
 [6,]    0    4    4    2    2    3    3    2    1     0
 [7,]    0    3    2    1    1    3    4    3    2     0
 [8,]    0    2    1    1    2    4    4    3    2     0
 [9,]    0    1    0    1    4    4    5    4    3     0
[10,]    0    0    1    2    3    5    5    4    2     0
filled.contour(1:10,1:10,x, xlab="x",ylab="y")
contour(1:10,1:10,x,nlevels=3, xlab="x",ylab="y")
```



Kontrolní úkoly

16. Vzhledem k tomu, že se další části týkaly specializovaných problémů, je spíše na každém čtenáři, jak tyto záležitosti využije. Doporučuji vám projít si všechny funkce a sami si vyzkoušet jestli je dokážete nastavit a použít. Použijte příklady uváděné ke každé funkci a měňte argumenty tak, abyste pochopili jejich význam. To, že zde nejsou uvedeny konkrétní úkoly neznamená, že funkce nebudeme využívat v běžných statistických postupech. Z grafů se zaměřte zejména na pairs, persp, dotchart, sunflowerplot, fourfoldplot, mosaicplot a image.

Shrnutí:

Metody low-lewel grafiky spočívají v tom, že graf skládáme z jednotlivých prvků. Mezi nastavitelné části grafu patří vlastní grafické okno, jednotlivé řady zobrazených dat, obrys grafického okna (box), názvy grafu (text), osy grafu, značky na osách atd. Do grafu můžeme ještě dále doplňovat různé symboly (včetně matematických vzorců), text, křivky, čáry a grafické objekty (obdélníky, mnohoúhelníky atd.) Pro podrobnější exploratorní analýzu dat a konkrétní statistické testy (např. χ^2-test) jsou v R implementovány další speciální grafy jako párový, čtyřlístkový, slunečnicový, hvězdicový aj.

Metody k zapamatování:

- Grafické prvky: points, lines, segments, curve, rect, polygon, arrows, abline, text, mtext, symbols, rug.
- Části grafu. axis, title, grid, legend.
- Grafický zápis matematických vzorců: expression(...).
- Manipulace s grafickými okny: recordPlot, windows, dev.cur, dev.set, dev.off, plot.new, frame, plot.window, split.screen, screen, erase.screen, close screen, identify, locator, xinch, cm, layout, layout.show.
- Práce s barvami: colors, palette, gray, rainbow, hsv, rgb, col2rgb, rgb2hsv, colorRamp.
- Speciální typy grafů: pairs, persp, dotchart, qqnorm, qqplot, qqline, sunflowerplot, fourfoldplot, mosaicplot, assocplot, stairs, coplot, image, contour, filled.contour.

Výsledky

```
1.  par(mar=c(4,4,1,1),las=1); plot(0,0,type="n",xlab="",ylab="",xlim=c(0,10),ylim=c(0,10))
2.  points(x=1:3,y=c(3,8,1),cex=1:3,pch=0,col=2:4)
3.  lines(x=1:3,y=c(3,8,1),lty=2,col=2,lwd=2)
4.  curve(sin(x)+5,0,10,col=2,add=T); curve(sin(x+11)+5,0,10,col=4,add=T)
5.  arrows(0,0,1,1)
6.  vec3<-c(5,4,7,8,3,1,2); points(4:10,vec3); arrows(4:10,vec3+1,4:10,vec3-
    1,code=3,length=0.1,angle=90)
7.  abline(10,-3); abline(v=3:4,col=7); abline(h=10,col=8,lty=2)
8.  text(4:10,vec3,labels=letters[1:7],pos=2,offset=0.5)
9.  mtext(text="osa x", side=1,at=7,line=2,col=2); měňte argument line od 0 do 3 a
    zjistíte, že první řádek je označen jako 0.
10. par(mar=c(4,4,1,1),las=1); plot(0,0,axes=F,type="n",xlab="",ylab="",xlim=c(-
    5,5),ylim=c(0,8))
11. grid(col=8)
12. axis(side=2,pos=0)
13. axis(side=1,at=c(-5,-3,3,5),labels=c("",expression(x[0]-Delta*t),
    expression(x[0]+Delta*t), ""), pos=0)
14. title(main="Graf pro druhy", line=0)
```

REJSTŘÍK

LITERATURA

Crawley M.J., 2005: Statistics: An Introduction using R. John Wiley & Sons.

Dalgaard P., 2004: Introductory Statistics with R. Springer.

Everitt B.S.& Nothorn T., 2006: A Handbook of Statistical Analyses Using R. Chapman & Hall.

Maindonald J., 2003: Data Analysis and Graphics Using R. Cambridge University Press.

Murrell P., 2005: R Graphics (Computer Science and Data Analysis). Chapman & Hall.

R Development Core Team, 2007: R: A language and environment for statistical computing. (online) R Foundation for Statistical Computing, Vienna, Austria. Dostupné z http://www.R-project.org

Venables W.N. & Smith D.M., 2002: An Introduction to R. Network Theory Ltd.

Verzani J., 2004: Using R for Introductory Statistics.Chapman & Hall.

Venables W.N. & Ripley B.D., 2003: Modern Applied Statistics with S. Springer.