Jazyky pre Dátovú Analytiku (JDA)

Prednáška č.1

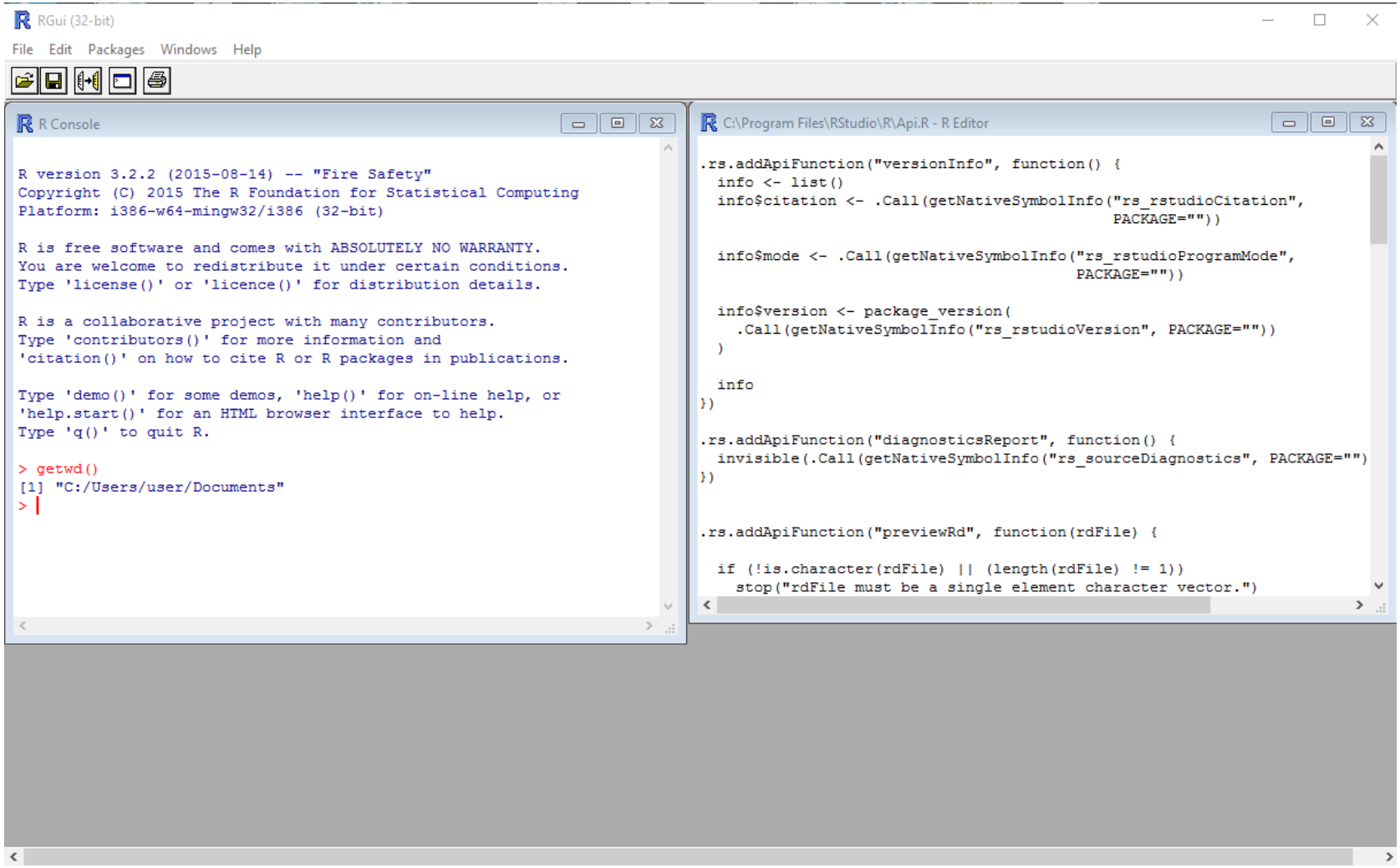
Predmet JDA

- Cieľ: Naučiť sa základy najpoužívanejších jazykov pre analýzu dát – R a Python – pre potreby ďalších predmetov počas štúdia / záverečných prác
- Rozsah:
  - cca 1h Pred. (v 2h blokoch, 3 x R + 2 x Python + 1 x ukážky použitia)
  - 2h Cvič.
- Podmienky absolvovania:
  - Absolvovanie cvičení - podľa pokynov cvičiacich
  - Klasifikovaný zápočet (100 bodov)
    - Celkové hodnotenie pozostáva z 2 písomiek zameraných na riešenie úloh v jazykoch R (50b) a Python (50b).
    - Študent prospeje a úspešne získa klasifikovaný zápočet, keď splní podmienku získať min. 25b z 50b v každej časti (R aj Python) a celkovo min. 51 bodov zo 100.
- Aktuálne dokumenty k predmetu, e-learning prostredie
  - MS Teams – tím JDA2023
  - Stránka k Pr/Cv - http://people.tuke.sk/viera.maslej.kresnakova/JDA/

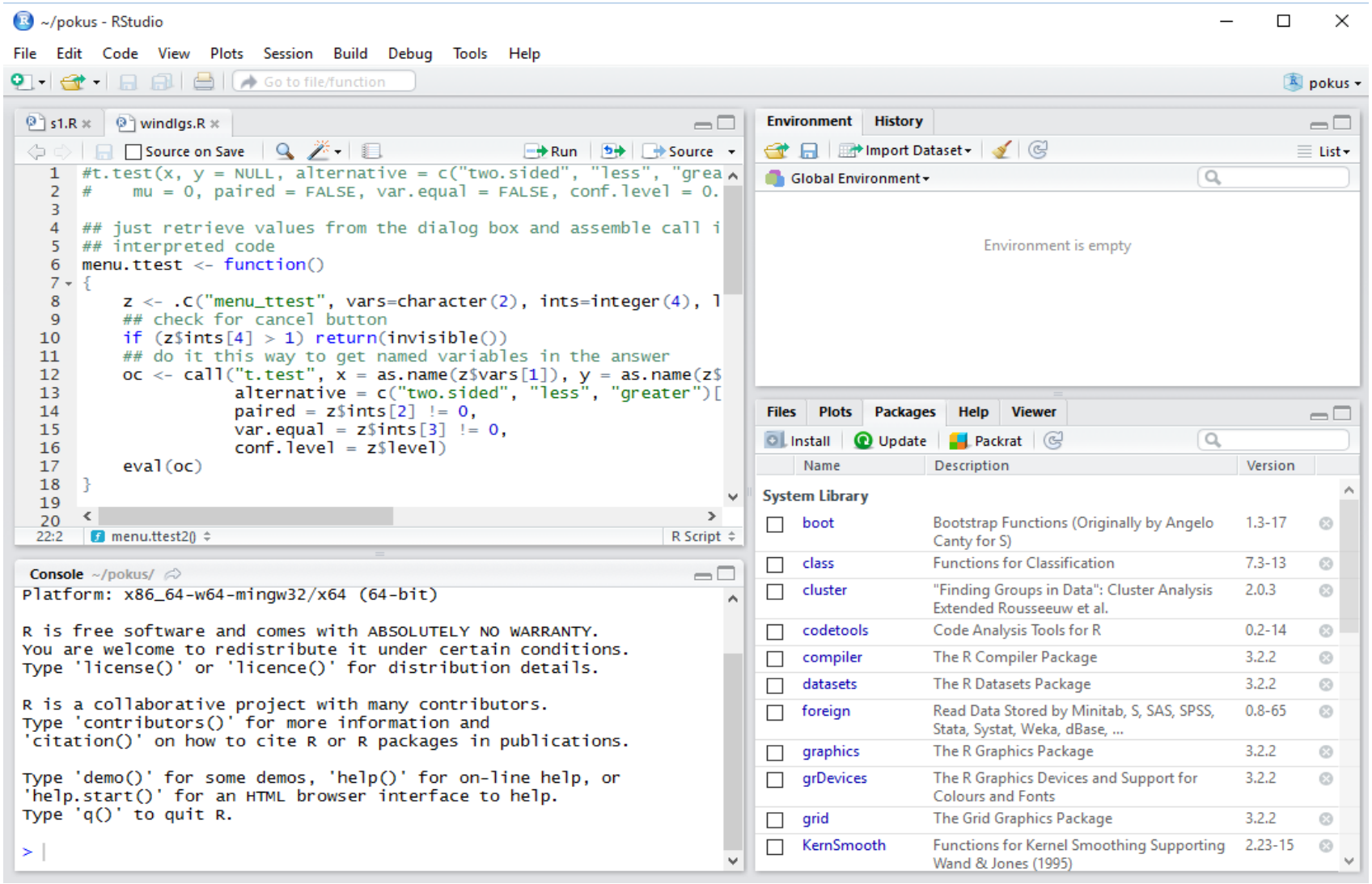
R

- R – štatistický softvér + jazyk
  https://www.r-project.org/
- Je voľne dostupný – open-source / free
- Má širokú / komplexnú paletu balíkov
- Poskytuje všetko pre dátovú analytiku
  - Prístup k dátam
  - Čistenie a transformácie dát
  - Analýzy
  - Reportovanie a vizualizácie
- Vývojové prostredie Rstudio - http://www.rstudio.com/
- Veľmi aktívna komunita vývojárov a používateľov
- Patrí medzi najpoužívanejšie nástroje pre prácu s dátami
  - Vrátane mnohých firiem majúcich vlastný softvér

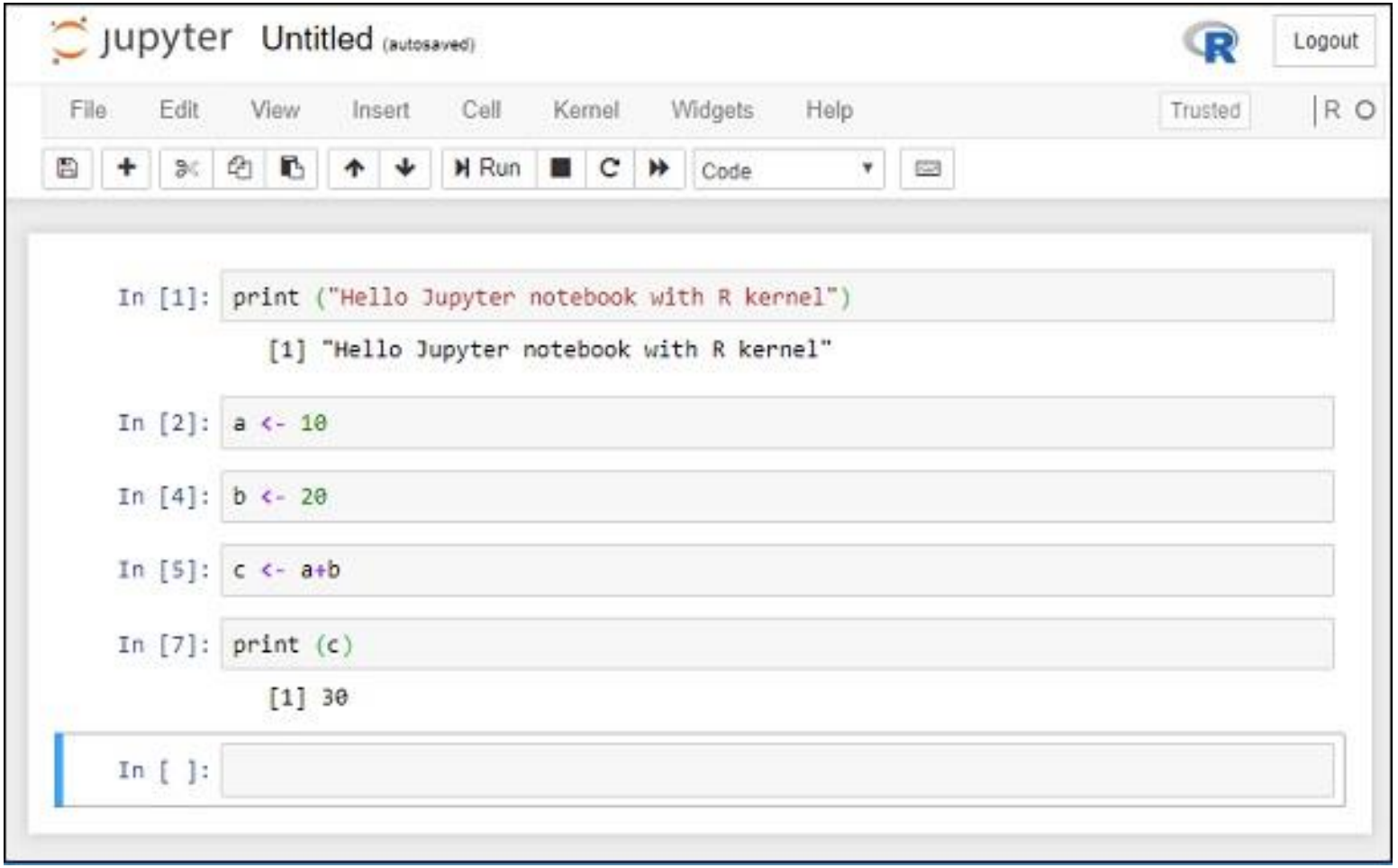
Klasické rozhranie R



R Studio


~/pokus - RStudio
File Edit Code View Plots Session Build Debug Tools Help
Go to file/function
pokus
s1.R
windlgs.R
Source on Save
Run
Source
#t.test(x, y = NULL, alternative = c("two.sided", "less", "grea
mu = 0, paired = FALSE, var.equal = FALSE, conf.level = 0.
just retrieve values from the dialog box and assemble call i
interpreted code
menu.ttest <- function()
{
z <- .C("menu_ttest", vars=character(2), ints=integer(4), l
check for cancel button
if (z$ints[4] > 1) return(invisible())
do it this way to get named variables in the answer
oc <- call("t.test", x = as.name(z$vars[1]), y = as.name(z$
alternative = c("two.sided", "less", "greater")[
paired = z$ints[2] != 0,
var.equal = z$ints[3] != 0,
conf.level = z$level)
eval(oc)
}
22:2
menu.ttest2()
R Script
Console ~/pokus/
Platform: x86_64-w64-mingw32/x64 (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment History
Import Dataset
List
Global Environment
Environment is empty
Files Plots Packages Help Viewer
Install Update Packrat
Name Description Version
System Library
boot Bootstrap Functions (Originally by Angelo Canty for S) 1.3-17
class Functions for Classification 7.3-13
cluster "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. 2.0.3
codetools Code Analysis Tools for R 0.2-14
compiler The R Compiler Package 3.2.2
datasets The R Datasets Package 3.2.2
foreign Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ... 0.8-65
graphics The R Graphics Package 3.2.2
grDevices The R Graphics Devices and Support for Colours and Fonts 3.2.2
grid The Grid Graphics Package 3.2.2
KernSmooth Functions for Kernel Smoothing Supporting Wand & Jones (1995) 2.23-15

R v Jupyter notebook-u



R console – evaluácia a výpis

- Konzola (console)
  - Po zápise výrazu (expression) + ENTER sa tento vyhodnotí a vráti sa výsledok (ak existuje návratová hodnota – napr. tlač hodnôt premennej) => EVALUÁCIA
- Niekoľko základných prvkov a vlastností
  - Priradzovací operátor: <- alebo = napr. x = 3
  - Explicitný výpis: print(x) ..... vypíše [1] 8
  - Implicitný výpis: x ..... aj toto vypíše [1] 8
  - Komentár: # toto je komentár
  - Reťazec: vždy v „“ ..... msg = "hello"
  - **R rozlišuje veľké a malé písmená (je case-sensitive) !**

Atomické dátové typy (triedy) v R

- R má 5 základných (atomických) tried objektov:
  - Reťazec (**character**) ..... "hello"
  - Reálna/numerická hodnota (**numeric**) 7.58
  - Celočíselná hodnota (**integer**) 3
  - Logická hodnota (**logical**) ..... TRUE alebo FALSE
  - Komplexné číslo (**complex**) ..... 1 + 2i

Vektory

- Najzákladnejší objekt v R je vektor (**vector**)
  - Vektor môže obsahovať iba objekty rovnakej triedy
  - Vytvorenie vektora často cez funkciu c() ... napr. v = c(2.5,1.2,0.8) vytvorí vektor **v** dĺžky 3 typu numeric, t.j. implicitný výpis **v** dá ….

    [1] 2.5 1.2 0.8
  - Prístup k i-tému prvku vektora x je pomocou x[i]
    - napr. v[3] vráti [1] 0.8
    - Možeme vybrať aj viac prvkov vektora a vrátiť tak jeho podčasť, napr. v[c(1,3)] vráti prvý a tretí prvok: [1] 2.5 0.8
  - Špeciálny operátor na vytvorenie vektora celých čísel od **m** po **n**

    m:n …. napr. ak chcem vektor celých čísel od 7 do 9, potom 7:9 vytvorí vektor

    [1] 7 8 9

Vektory (2)

- Na vytváranie vektorov sa najčastejšie používa c() funkcia – výsledný vektor je však vždy určitej atomickej triedy, zistiť triedu môžeme pomocou funkcie class()

```
> v1 = c(1.2, 0.7) # numerický vektor
> v2 = c(TRUE, FALSE) # vektor logických hodnôt
> v3 = c(T, F, T) # vektor logických hodnôt
> v4 = c("y", "w", "z") # vektor znakov
> v5 = c(2,5,3) # celočíselný vektor
> v6 = c(1+3i, 2+2i) # komplexný vektor
```

- Pri pomiešaní typov dôjde k implicitnej korekcii na jeden z typov

```
> y1 <- c(2.5, "ahd") # výsledok – vektor znakov
> y2 <- c(TRUE, 7) # výsledok – numerický vektor
> y3 <- c("a", TRUE) # výsledok – vektor znakov
```

Zistenie triedy objektu v R cez class:

```
> class(v1)
[1] "numeric"
> class(v6)
[1] "complex"
```

Explicitná korekcia (úprava typu) vektora

- Objekty môžeme aj explicitne „ opraviť “ (v podstate pretypovať) z jednej triedy na inú pomocou *as*.* funkcií (ak sú k dispozícii)

```
> x <- 0:4
> class(x)
[1] "integer"
> as.numeric(x)
[1] 0 1 2 3 4
> as.logical(x)
[1] FALSE TRUE TRUE TRUE TRUE
> as.character(x)
[1] "0" "1" "2" "3" "4"
```

- Existujú však prípady nelogických (nezmyselných resp. neurčených) výsledkov => výsledkom môžu byť NA hodnoty (Not Available)

```
> x <- c("a", "b", "c")
> as.numeric(x)
[1] NA NA NA
Warning message:
NAs introduced by coercion
```

Zoznam ako objekt v R

- Zoznam (**list**) je špeciálny prípad vektoru, ktorý môže obsahovať objekty rôznych tried, a to napríklad:
  - Prvky atomických tried
  - Vektory
  - Ďalší zoznam (čiže v podstate podzoznam – to umožňuje vytvárať štruktúrovanú hierarchiu prvkov)
  - Dátovú tabuľku (viď. neskôr dataframe)
- Príklad – zoznam obsahujúci rôzne atomické prvky, vektor, ďalší zoznam (podzoznam)

```
> mylist = list("Red", TRUE, 11.54,
c(1,5,3), list(2,"Kosice"))
> mylist
[[1]]
[1] "Red"

[[2]]
[1] TRUE

[[3]]
[1] 11.54

[[4]]
[1] 1 5 3

[[5]]
[[5]][[1]]
[1] 2

[[5]][[2]]
[1] "Kosice"
```

Prístup k prvkom zoznamu (len príklady):

```
> mylist[[4]]
[1] 1 5 3
> mylist[[5]][[2]]
[1] "Kosice"
> mylist[[c(5,2)]]
[1] "Kosice"
> mylist[4]
# vráti zoznam !!!
[[1]]
[1] 1 5 3
```

Atribúty dátových objektov

- R objekty môžu mať atribúty
  - Názvy elementov, dimenzií, riadkov, stĺpcov v rámci objektu (names, dimnames, row.names, ...)
    - Príklad: `> z = list(a = 1, b = "c", c = 1:3)`
      `> names(z)`
      `[1] "a" "b" "c"`
  - Dimenzie (dimensions ..... dôležité napr. pre matice)
  - Trieda (class)
  - Dĺžka/veľkosť (length)
  - ... [ďalšie atribúty a metadáta objektov]
- Všetky atribúty objektu môžu byť vypísané aj pomocou funkcie *attributes()*

Matice

- Matice v R sú objekty *vector* s definovaným *dimension* atribútom – integer vektor dĺžky 2 (s hodnotami *nrow* a *ncol*)
- Matice je možné vytvoriť rôznymi spôsobmi, napríklad:
  - Cez *matrix()* funkciu vložením vektora hodnôt a pridaním dimenzií matice
  - Pridaním dimenzií priamo k existujúcemu vektoru cez *dim()*
  - Spojením vektorov rovnakej dĺžky do matice ako jej stĺpcov alebo riadkov (stĺpcové alebo riadkové priradenie, tzv. binding) – funkcie cbind() a rbind(), pre spojenie vektorov po stĺpcoch resp. riadkoch (column-binding resp. row-binding)
- Matica sa hodnotami vektora štandardne vypĺňa po stĺpcoch
  - ak chceme vyplnenie po riadkoch, môžem použiť argument funkcie matrix *byrow=TRUE*

```
> m1 = matrix(c(7,2,5,8,3,6), nrow = 2, ncol = 3)
> m1
     [,1] [,2] [,3]
[1,]  7  5  3
[2,]  2  8  6
```

```
> m2 = matrix(c(7,2,5,8,3,6), nrow = 2, ncol = 3, byrow=TRUE)
> m2
     [,1] [,2] [,3]
[1,]    7    2    5
[2,]    8    3    6
```

```
> v = c(2,8,4,5)
> v
[1] 2 8 4 5
> dim(v) = c(2,2)
> v
     [,1] [,2]
[1,]    2    4
[2,]    8    5
```

```
> x = c(3,4,5)
> y =c(13,14,15)
> cbind(x, y)
      x y
[1,] 3 13
[2,] 4 14
[3,] 5 15
```

```
> x = c(3,4,5)
> y =c(13,14,15)
> rbind(x, y)
  [,1] [,2] [,3]
x   3    4    5
y  13   14   15
```

Matice (2)

- Prístup k hodnotám matice
  - Konkrétny prvok matice M, riadok i stĺpec j ... M[i,j]
  - Získanie celého i-tého riadku ... M[i,]
  - Získanie celého j-tého stĺpca ... M[,j]
- Je možné pomenovať riadky a stĺpce matice – pomenovania sa ukladajú do atribútu *dimnames* ako zoznam dvoch vektorov s názvami jednotlivých riadkov resp. stĺpcov
  - Praktické funkcie pre zistenie a nastavenie názvov stĺpcov alebo riadkov sú colnames() resp. rownames()

```
> M = c(2,8,4,5); dim(M) = c(2,2)
> M
     [,1] [,2]
[1,]    2    4
[2,]    8    5
> M[1,2]   # vyber konkretneho prvku M
[1] 4
> M[1,]  # vyber prveho riadku M
[1] 2 4
> M[,2]  # vyber druheho stlpca M
[1] 4 5
> colnames(M) = c("Prvy","Druhy")
> M
     Prvy Druhy
[1,]    2    4
[2,]    8    5
```

Vektorové (maticové) operácie

Mnoho operácií v R je riešených vektorovo (maticovo) => zlepšuje sa tak efektívnosť a čitateľnosť kódu

```
> x <- 1:4; y <- 6:9
> x + y
[1] 7 9 11 13
> x > 2
[1] FALSE FALSE TRUE TRUE
> x >= 2
[1] FALSE TRUE TRUE TRUE
> y == 8
[1] FALSE FALSE TRUE FALSE
> x * y
[1] 6 14 24 36
> x / y
[1] 0.1666667 0.2857143 0.3750000
    0.4444444
```

```
> M = matrix(1:4, 2, 2)
> N = matrix(c(10,5,5,1), 2, 2)
> M
     [,1] [,2]
[1,]    1    3
[2,]    2    4
> N
     [,1] [,2]
[1,]   10    5
[2,]    5    1
> M * N     # násobenie po elementoch
     [,1] [,2]
[1,] 10   15
[2,] 10    4
> M %*% N   # skutočné násobenie matíc
     [,1] [,2]
[1,] 25    8
[2,] 40   14
```

Faktory

- Faktor (*factor*) sa používa pre reprezentáciu kategoriálnych (diskrétnych) atribútov
  - Pre vytvorenie sa používa funkcia *factor*
  - Môže byť neusporiadaný (nominálny atribút) alebo usporiadaný (ordinálny atribút, nastavenie pomocou *ordered=TRUE* pri použití funkcie *factor*)
  - Je to obdoba integer vektora, kde každá hodnota môže mať popis (názov, reprezentujúci úroveň = level), vnútorná reprezentácia je práve integer vector
  - Použitie faktorov s popismi namiesto integer zlepšuje pochopiteľnosť dát
  - Poradie hodnotových úrovní (levels) môžeme zmeniť (hlavne pri ordered)

```
> x = factor(c("ford", "bmw", "ford", "bmw", "bmw"))
> x
[1] ford bmw ford bmw bmw
Levels: bmw ford
> table(x)
bmw ford
      3     2
> unclass(x)
    [1] 2 1 2 1 1
> attr(,"levels")
[1] "bmw" "ford"
> x = factor(c("ford", "bmw", "ford", "bmw", “bmw"),
    levels = c("ford", "bmw“))
> x
[1] ford bmw ford bmw bmw
Levels: ford bmw
```

Dátum a čas

- V R je vytvorená špeciálna reprezentácia dátumov a časov
  - Dátumy sú reprezentované ako trieda Date – interne ako počet dní od 1.1.1970 (1970-01-01), môžu byť prevádzané z reťazca cez as.Date()
  - Presný čas je reprezentovaný cez POSIXct (kalendárový čas ako počet sekúnd od 1.1.1970, integer, ct = calendar time) alebo POSIXlt triedy (lt = local time, uchováva rôzne užitočné informácie ako prvky vektora)

```
> x = Sys.Date()          > y = Sys.time()
[1] "2018-09-20"        "2018-09-20 21:38:08 CEST"
```

- Reťazce znakov môžu byť upravené na Date/Time triedy pomocou funkcie *strptime()* alebo cez funkcie *as.Date()*, *as.POSIXlt*, *as.POSIXct*
- Vďaka reprezentácii je možné s nimi robiť porovnania a jednoduché numerické a štatistické operácie
- Formát zobrazenia a mnoho ďalších aspektov je možné meniť, existujú viaceré balíky pre uľahčenie práce s dátumami a časom ako *lubridate*, *chron*, ...
- Viac info – web – tutoriály, dokumentácie, atď. – napr. pdf na webe: http://biostat.mc.vanderbilt.edu/wiki/pub/Main/ColeBeck/datestimes.pdf

Chýbajúce a nedefinované hodnoty

- Chýbajúce hodnoty sú označené NA (Not Available)
- NaN (Not A Number) sa používa pre nedefinované (matematické) výsledky operácií
  - Napr. 0 / 0 dáva NaN hodnotu
- Funkcia is.na() sa používa na testovanie či ide o NA hodnotu
- Funkcia is.nan() sa používa na testovanie či ide o NaN hodnotu
- Hodnoty NA takisto majú svoju triedu – existuje integer NA, character NA, atď.
- NaN hodnota je zároveň NA hodnota – opačne to neplatí

```
> x <- c(1, NaN, NA)
> is.na(x)
[1] FALSE TRUE TRUE
> is.nan(x)
[1] FALSE TRUE FALSE
```

Dátové rámce (Data Frames)

- Používajú sa na ukladanie tabuľkových dát, pričom
  - Sú reprezentované špeciálnym typom zoznamu, kde každý element má rovnakú dĺžku
  - Každý element zoznamu je vlastne stĺpec a dĺžka je počet riadkov, pričom každý stĺpec môže byť inej triedy
  - Majú špeciálny atribút row.names pre názvy riadkov, default hodnota je číslo riadku pri vytvorení
  - Z pohľadu dátovej analytiky, riadok reprezentuje objekt = inštanciu = príklad, stĺpec reprezentuje atribút
  - DataFrame môžeme vytvoriť pomocou funkcie *data.frame*, pričom stĺpce (atribúty) sú vložené priamo ako argumenty
  - S existujúcich súborov sa načítavajú dáta do dátových rámcov volaním *read* funkcií, napr. *read.table*, *read.csv*, ...
  - Počet riadkov resp. stĺpcov je možné získať napríklad pomocou funkcií *nrow* a *ncol*

```
> df = data.frame(a1= 1:4, a2 = c(0.2, 0.7, 1.5, 2.2))
> df
  a1 a2
1  1  0.2
2  2  0.7
3  3  1.5
4  4  2.2
> row.names(df)
[1] "1" "2" "3" "4“
> nrow(df)
[1] 4
> ncol(df)
[1] 2
> df[4,2]
[1] 2.2
> df[2, "a2" ]
[1] 0.7
```

Názvy objektov

- Objekty v R je možné pomenovať, čo umožňuje lepšie pochopenie dátových objektov a zvyšuje čitateľnosť kódu
- Pomenovať môžeme hodnoty vektorov, zoznamov alebo matíc

```
> x = 1:2
> names(x) = c("a1", "a2")
> x
 a1 a2
  1  2
```

```
> x = list(a = 1, b = 2, c = 3)
> x
$a
[1] 1
$b
[1] 2
$c
[1] 3
```

```
> m = matrix(1:4, nrow = 2, ncol = 2)
> dimnames(m) = list(c("a", "b"), c("c", "d"))
> m
  c d
a 1 3
b 2 4
```

Štruktúrovanie programu v R

- Beh programu v R je možné kontrolovať pomocou nasledujúcich štruktúr :
  - **if, else**: testovacia podmienka if(x > 3) {y = 10} else {y = 0}
  - **for**: cyklus fixného počtu opakovaní for(i in 1:10) {print(i)}
  - **while**: cyklus bežiaci kým je platná podmienka (ako while v C)
  - **repeat**: spúšťa nekonečný cyklus !!!! (while bez podmienky)
  - **break**: ukončuje beh cyklu (ako break v C)
  - **next**: preskočí interakciu v cykle (ako continue v C)
  - **return**: výstup funkcie
- Veľmi často sa však kvôli efektívnosti a vektorizácii namiesto kontrolných štruktúr používajú funkcie realizujúce rovnakú požadovanú činnosť, napríklad:
  - Funkcie pre spracovanie cyklov, tzv.*apply funkcie = cyklické funkcie
  - Rozhodovacie funkcie – príkladom je *ifelse* funkcia realizujúca if/else kontrolnú štruktúru, ktorá navyše ľahko umožní aj vektorové spracovanie
    - napr. ak chceme realizovať rôzne operácie pre prvky vektora podľa zvolenej podmienky – povedzme že pre prvky vektora *x* ktoré sú > 0 chceme realizovať násobenie *2 a pre ostatné *4

```
> x = c(1,-3,2,4,-1)
> ifelse(x > 0, 2*x, 4*x)
[1]  2 -12  4  8 -4
```

Funkcie

- Funkcie sa tvoria pomocou direktívy *function()* a sú takisto objektmi jazyka R (trieda *function*)
- Zavádzajú sa priamo vložením do konzoly alebo načítaním zo súboru (zvyčajne **.R** súboru) cez *source()*
- Pre funkcie v R platí:
  - Môžu byť odovzdávané ako argumenty ďalším funkciám
  - Funkcie môžu byť vnorené, môžete definovať funkciu vo vnútri inej funkcie
  - Návratová hodnota je posledný výraz v tele funkcie, ktorý je vyhodnotený
- Pomenované argumenty funkcie môžu mať definovanú „default“ hodnotu (viď. druhý argument v príklade) – tá sa použije ak neuvedieme argument pri volaní

```
nasobenie = function(a, b=2) {
      a*b
}
```

```
> nasobenie(4,5)
[1] 20
> nasobenie(5)
[1] 10
```

Spárovanie argumentov funkcie

- Argumenty volania funkcie môžu byť spárované s definíciou **pozične** alebo cez **pomenovanie**
  - Použitie pomenovania argumentu je užitočné najmä pri funkciách s veľkým počtom argumentov, kde je problém si pamätať pozíciu argumentu
  - Je možné ich miešať – t.j. použiť pozičné volanie alebo s pomenovaním, alebo dokonca kombinovať obidve v jednom volaní
  - Argumenty môžu byť aj čiastočne spárované, pričom poradie priorít je

    (1) exaktné spárovanie pomenovaním (1), čiastočné spárovanie pomenovaním (2) a pozičné (3)
- Príklad ekvivalentných volaní funkcie sd(), ktorá má definované argumenty (x, na.rm = FALSE), pričom vstupné dáta sú mydata

```
> sd(mydata)
> sd(x = mydata)
> sd(x = mydata, na.rm = FALSE)
> sd(na.rm = FALSE, x = mydata)
> sd(na.rm = FALSE, mydata)
```

„Lazy“ evaluácia

- Argumenty funkcie sú tzv. „lenivo“ vyhodnocované, t.j. až v momente keď je to potrebné
  - f <- function(a, b) { a^2} ..... Táto funkcia vlastne nepotrebuje *b* argument, čiže pri volaní f(2) nevyprodukuje chybu, nakoľko 2 je pozične spárované s argumentom *a* a ten na beh funkcie postačuje
  - Naopak g <- function(a, b) { a*b} pri volaní g(2) vyprodukuje chybu, lebo potrebuje b a nemá ho pri volaní k dispozícii a neexistuje ani default hodnota v definícii
  - Ak však do definície pridáme default hodnotu do definície, bude volanie úspešné, t.j. g <- function(a, b = 1) { a*b} prejde f(2) bez chyby s výsledkom 2 (argument *a* sa páruje pozične s hodnotou 2 vo volaní funkcie, pre argument *b* sa použije default hodnota 1, čiže výsledok bude 2*1=2)

Argument „...“

- Argument „...“ je potrebný v prípadoch keď počet argumentov predaných funkcii nemôžeme určiť dopredu

```
> args(paste)
function (..., sep = " ", collapse = NULL)
```

- Jedna podmienka použitia „...“ je že akýkoľvek argument uvedený za ním v definícii musí byť vždy menovaný explicitne

```
> paste("ahoj", "ja", "som", "tu", sep = ":")
[1] "ahoj:ja:som:tu"
> paste("ahoj", "ja", "som", "tu", ":")
[1] "ahoj ja som tu :"
```

Funkcie pre cyklické spracovanie (loops)

- Namiesto písania písanie cyklov je možné použiť pre možné využiť špeciálne funkcie pre cyklické spracovanie (často označované ako *apply funkcie), ktoré takéto situácie uľahčujú. Príkladmi sú:
  - apply: aplikuje funkciu cez ohraničenia poľa
  - lapply: cyklus cez zoznam a evaluácia funkcie nad každým elementom
  - sapply: ako lapply, ale zjednošuje výstup
  - vapply: sapply, avšak definujeme výstupný typ funkcie (bezpečnejšia verzia)
  - tapply: aplikuje funkciu na podmnožinu vektora/zoznamu (podmnožiny sú definované inou hodnotu vybraného faktorového atribútu)
  - mapply: multivarietná verzia sapply
- Výhody cyklických funckií
  - ľahké použitie pri spracovaní vektorov a matíc zápisom jednej funkcie
  - môžu byť efektívnejšie ako vlastné napísanie cyklov aj z hľadiska rýchlosti spracovania (ich použitie je už optimalizované)

apply

- *apply* sa používa na evaluáciu funkcie nad objektom (ako matica) vzhľadom k definovaným ohraničeniam výberu ich elementov
  - Používa sa pre zovšeobecnené polia, vstupom je *matrix* alebo *dataframe*
  - Často sa aplikuje funkcia voči riadkom alebo stĺpcom matice / tabuľky dát
  - apply (X, MARGIN, FUN, ...) ... X je vstupný objekt (matica alebo tabuľka dát), MARGIN je integer vector ohraničujúci vybrané prvky, FUN je aplikovaná funkcia, „...“ je miesto pre ďalšie parametre funkcie FUN

```
> M = matrix(1:6, 3, 2)
> apply(M, 2, sum)   # urobí sumu po stĺpcoch
 [1] 6 15
> apply(M, 1, mean) # priemer po riadkoch
 [1] 2.5 3.5 4.5
```

lapply

- *lapply* je cyklická funkcia pre spracovanie zoznamu, má 3 argumenty: (1) zoznam X (2) funkciu FUN (3) iné argumenty cez „...“
  - Ak X nie je zoznam, je upravený na zoznam cez *as.list*
  - Výstupom je vždy opäť zoznam, kde na každý prvok pôvodného zoznamu je aplikovaná funkcia FUN
  - Ak pre funkciu použijete argumenty, pridávajú sa za meno funkcie ako iné (ďalšie) argumenty

```
> X = list(a = 1:5, b = 6:10)
> lapply(X, mean)
$a [1] 3
$b [1] 8
```

```
> lapply(X, quantile, probs = seq(0, 1, 0.5))
$a 0% 50% 100%
   1   3   5
$b 0% 50% 100%
   6   8   10
```

*apply – použitie anonymnej funckie

- Všeobecne pre *apply funkcie sa používajú často tzv. anonymné funkcie na mieste argumentu FUN
- Príklad:
  - Máme zoznam matíc a chceme extrahovať prvý stĺpec každej z nich – jeden spôsob ako to urobiť je že definujeme na pozícii FUN argumentu anonymnú funkciu, ktorá to zabezpečí

```
> X = list(a = matrix(1:4, 2, 2), b = matrix(1:8, 4, 2))
> lapply(X, function(e) e[,1])
$a [1] 1 2
$b [1] 1 2 3 4
```

sapply

- *sapply* sa snaží zjednodušiť výsledok *lapply* (ak je to možné)
  - Ak je výsledok zoznam s elementmi dĺžky 1, vráti vektor
  - Ak je výsledok zoznam s elementmi, ktorými sú vektory rovnakej dĺžky (väčšej ako 1), vráti maticu
  - Inak vráti zoznam ako *lapply*

```
> X = list(1:5,6:10)
> sapply(X, mean)
[1] 3 8
```

mapply

- *mapply* aplikuje paralelne funkciu pre rôzne nastavenia jej parametrov a vráti list/vektor výsledkov jednotlivých aplikácií funkcie
- mapply (FUN, ..., MoreArgs = NULL, SIMPLIFY = TRUE, USE.NAMES = TRUE)
  - FUN je funkcia, „...“ argumenty cez ktoré sa aplikuje FUN, MoreArgs je zoznam ďalších argumentov funkcie FUN, SIMPLIFY indikuje zjednodušenie výstupu (podobne ako pri sapply)
- Príklad:
  - Povedzme že máme funkciu **scitaj** z dvomi argumentmi **x** a **y**

```
> scitaj = function(x,y) {x+y}
```

  - Následne chceme ju zavolať pre ich rôzne hodnoty (povedzme že 3 rôzne nastavenia x a y, napr. 1 a 2, 4 a 2, 8 a 6) a uložiť výsledky do jedného objektu (listu / vektora) – namiesto samostatného volania funkcie **scitaj** 3-krát (t.j. napríklad pre prvú dvojicu scitaj(1,2), atď.) a skladania výsledkov do vektora využijeme mapply pre celú sadu parametrov x a y

```
> mapply(scitaj, x=c(1,4,8), y=c(2,2,6))
[1]  3  6 14
```

tapply a split

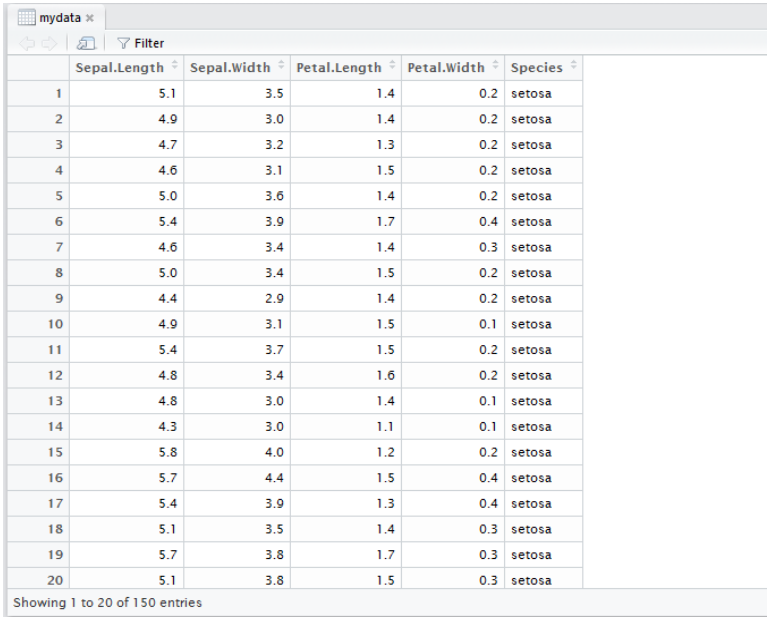
- *tapply* sa používa na aplikáciu funkcie na podmnožiny vektora ktoré sú určené zvoleným faktorovým atribútom
- tapply (X, f, FUN = NULL, ..., simplify = TRUE)

  X – vektor, f – faktor alebo zoznam faktorov (alebo niečo upraviteľné na faktor(y)), FUN – aplikovaná funkcia, ... – iné argumenty pre FUN, simplify – indikátor zjednodušenia výstupu

  Príklad: máme v dátovej tabuľke atribút **vyska** (numerický) a **pohlavie** (faktorový atribút s hodnotami muž alebo žena) – tapply nám umožní vypočítať funkciu FUN (napr. priemer) z vektora výšok (X je teda vyska) samostatne pre mužov a ženy (použitím atribútu pohlavie ako f)
- *split* funkcia je v podstate prvá časť tapply - zoberie vektor alebo iné objekty (zoznam, dátovú tabuľku) a rozdelí ich do skupín podľa faktoru alebo zoznamu faktorov
  - split (X, f, drop = FALSE, ...), kde X – vektor, zoznam alebo dátová tabuľka, f - faktor, drop – či vylúčiť prázdne úrovne faktorov
  - Použitie split a lapply za sebou dokopy dávajú to isté čo tapply

Získanie a príprava dát

- Pred analýzou dát potrebujeme dáta najprv získať
  - Zo súborov, webu, pomocou API, z databáz, ...
- Často však dáta neprichádzajú v ideálnej forme
  - Čistenie dát, transformácia dát => príprava dát => dobre pripravené uľahčujú pochopenie a urýchľujú analýzy
- Množina položiek = množina objektov / inštancií o ktoré sa zaujímame
  - Riadok = položka = objekt = inštancia = príklad = prípad (case)
  - Premenná = atribút = stĺpec = meranie alebo charakteristika objektu
    - Kvalitatívna – pohlavie, krajina pôvodu, ...
    - Kvantitatívna – výška, váha, ...

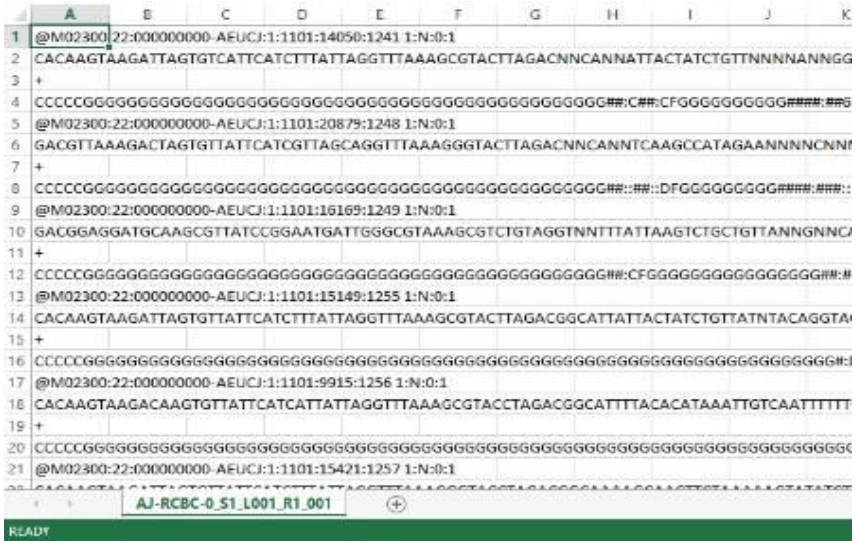


mydata

Filter

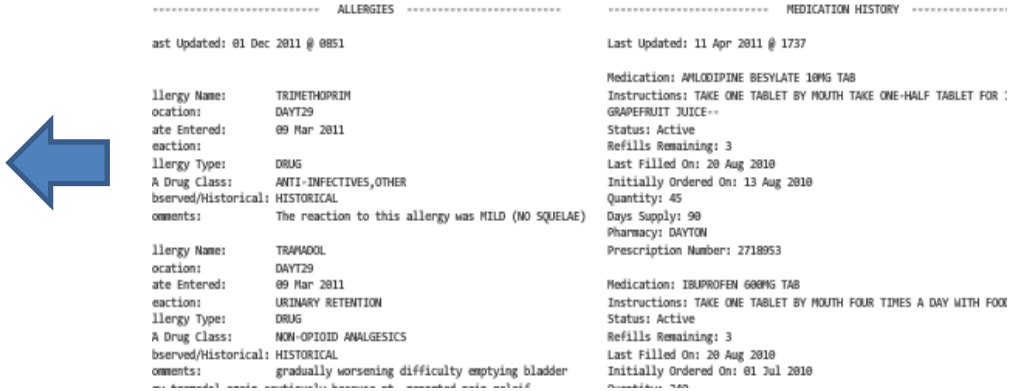
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa

Showing 1 to 20 of 150 entries



AJ-RCBC-0_S1_L001_R1_001

READY



```
---------- ALLERGIES ----------                    ---------- MEDICATION HISTORY ----------

ast Updated: 01 Dec 2011 @ 0851                   Last Updated: 11 Apr 2011 @ 1737

                                                  Medication: AMLODIPINE BESYLATE 10MG TAB
llergy Name:       TRIMETHOPRIM                   Instructions: TAKE ONE TABLET BY MOUTH TAKE ONE-HALF TABLET FOR :
ocation:           DAYT29                         GRAPEFRUIT JUICE--
ate Entered:       09 Mar 2011                    Status: Active
eaction:                                          Refills Remaining: 3
llergy Type:       DRUG                           Last Filled On: 20 Aug 2010
A Drug Class:      ANTI-INFECTIVES,OTHER          Initially Ordered On: 13 Aug 2010
bserved/Historical: HISTORICAL                    Quantity: 45
omments:           The reaction to this allergy was MILD (NO SQUELAE)  Days Supply: 90
                                                  Pharmacy: DAYTON
llergy Name:       TRAMADOL                       Prescription Number: 2718953
ocation:           DAYT29
ate Entered:       09 Mar 2011                    Medication: IBUPROFEN 600MG TAB
eaction:           URINARY RETENTION              Instructions: TAKE ONE TABLET BY MOUTH FOUR TIMES A DAY WITH FOOI
llergy Type:       DRUG                           Status: Active
A Drug Class:      NON-OPIOID ANALGESICS          Refills Remaining: 3
bserved/Historical: HISTORICAL                    Last Filled On: 20 Aug 2010
omments:           gradually worsening difficulty emptying bladder  Initially Ordered On: 01 Jul 2010
```

```
        },
        "possibly_sensitive": false,
        "retweet_count": 0,
        "retweeted": false,
        "source": "\u003ca href=\"http://twitter.com/download/iphone\"
rel=\"nofollow\"\u003eTwitter for iPhone\u003c/a\u003e",
        "text": "@xjillxdxe dikke crisis HAHAAHHA",
        "timestamp_ms": "1426616639678",
        "truncated": false,
        "user": {
                "contributors_enabled": false,
                "created_at": "Sun Jan 09 17:24:11 +0000 2011",
                "default_profile": false,
                "default_profile_image": false,
                "description": "I can't change the way I think, and I can't change the way I
am. But if I offended you.. Good. 'Cause I still don't give a fuck.",
                "favourites_count": 5023,
                "follow_request_sent": null,
                "followers_count": 1390,
                "following": null,
                "friends_count": 513,
                "geo_enabled": true,
                "id": 236040713,
                "id_str": "236040713",
                "is_translator": false,
                "lang": "en",
                "listed_count": 4,
                "location": "045",
                "name": "mirte",
                "notifications": null,
                "profile_background_color": "131516",
                "profile_background_image_url": "http://pbs.twimg.com/profile_background_images
/589241405/02s12g8gk0qv5mo2r6b6.jpeg",
```

Stiahnutie súboru z webu

- download.file()
- Hlavné parametre:
  - url – URL adresa súboru
  - destfile, method

```
> fileUrl <- "http://people.tuke.sk/peter.butka/res/irisdata.csv"
> download.file(fileUrl, destfile = "iris.csv")
trying URL 'http://people.tuke.sk/peter.butka/res/irisdata.csv'
Content type 'text/csv' length 4586 bytes
downloaded 4586 bytes
```

Načítanie textových súborov

- Textové súbory – read.table()
  - Hlavná funkcia pre načítanie dát do R, flexibilná, robustná, vyžaduje však viacero parametrov
  - Parametre: **file**, header (T/F), sep, dec, row.names, **nrows** (počet riadkov pre načítanie), **quote (**quote="" – často problém**)**, fill, **skip** (počet vynechaných riadkov na začiatku), **na.strings** (chýbajúce hodnoty), ...

```
> dfiris = read.table("iris.csv", sep = ";", header = TRUE)
> head(dfiris) # prvé riadky (n – druhý par.), pre posledné tail
```

- Verzie pre CSV – špeciálny read.table s default hodnotami pre separátor a desatinnú čiarku
  - read.csv() ... hlavička + sep="," a dec="."
  - read.csv2() ... hlavička + sep=";" a dec=","
- Rstudio – Import dataset – zo súboru, alebo cez linku – je možné tam nastaviť parametre cez GUI rozhranie

JSON súbory

- JSON - Javascript Object Notation
  - ukladanie a zdieľanie dát v štruktúrovanej forme, key:value páry
  - Dáta - čísla, reťazce, boolovské hodnoty, polia [], vnorené objekty {}

data.JSON

```
[
  {
    "meno": "Peter",
    "priezvisko": "Hromada",
    "adresa": {
      "ulica": "Hlavna",
      "cislo": 23,
      "mesto": "Kosice"
    },
    "vek": 27
  },
  {
    "meno": "Jan",
    "priezvisko": "Hruska",
    "adresa": {
      "ulica": "Bocna",
      "cislo": 10,
      "mesto": "Presov"
    },
    "vek": 23
```

Načítanie JSON súborov

- Napr. knižnica **jsonlite**

```
> mydata = fromJSON("data.JSON")
> names(mydata)
[1] "meno"  "priezvisko" "adresa"     "vek"
> mydata
meno priezvisko adresa.ulica adresa.cislo adresa.mesto vek
1 Peter   Hromada  Hlavna     23       Kosice  27
2   Jan    Hruska     Bocna      10       Presov  23
3   Jan    Kovac     Kosicka     10   Michalovce  18
> names(mydata$adresa)
[1] "ulica" "cislo" "mesto"
> mydata$adresa$mesto
[1] "Kosice"     "Presov"     "Michalovce"
```

Uloženie dát do JSON súboru

- toJSON()

  > myjson = toJSON(iris, pretty=TRUE)

  > cat(myjson)


```
,
{
  "Sepal.Length": 6.2,
  "Sepal.Width": 3.4,
  "Petal.Length": 5.4,
  "Petal.Width": 2.3,
  "Species": "virginica"
},
{
  "Sepal.Length": 5.9,
```


- Zapísanie do súboru

  > write(myjson, file="irisdata.JSON"))

Načítanie/práca s dátami iných zdrojov

- Excel – Balík **xlsx** (s funkciou read.xlsx()), balík XLConnect, prípadne uložiť csv formát a načítať cez read.csv
- XML súbory – balík XML
- Databázy – pre načítanie dát z databázy existujú rôzne balíky
  - Špecializované, napr.: pre MySQL je balík RMySQL, pre PostgreSQL balík RPostgreSQL, pre MongoDB balík Rmongo, ...
  - Všeobecnejšie, napr.: balík DBI – všeobecné rozhranie na databázy, pre prístup cez ODBC protokol – balík RODBC
- Načítanie dát z obsahu web stránok – tzv. web scrapping – napr. tiež možné cez XML balík
- Využitie špeciálnych API pre tvorbu aplikácia s prístupom k zdrojom ako Twitter, Facebook, GoogleMaps, ...
- Balík **foreign** – pre načítanie súborov z iných softvérov podobných R – väčšinou v tvare read.* funkcií, napr. read.dta(), read.octave(), read.spss(), ...
- Balíky pre načítanie obrázkov (balíky jpeg, png, ...), zvukových dát (balíky ako tuneR, seewave, ...), či pre načítanie GIS dát (Geografické IS, balíky ako raster, rgeos, ...)
- ....

Práca s dátami

- Tabuľka dát = dátový rámec (data frame)
  - Existuje viacero implementácii – štandardný **data.frame**, data.table, tidyverse (tibble), ... (väčšinu funkcionalít rôzne implementácie zdieľajú s hlavným data.frame)
  - Existuje aj viacero balíkov s funkciami pre predspracovanie dát a ich ďalšie používanie – napr. **dplyr** (plyr), tidyr, purrr, ...
- Na cvičeniach sa zameriame na vybrané základné funkcie **data.frame** použiteľné pre každú implementáciu + používanie **dplyr**

Práca s tabuľkou dát

- Klasické vyberanie podčastí - pri použití (nielen) štandardnej implementácie data frame (majme dataframe X)
  - X[,1]; X[,"a"]; X[1:5, "a"] ….
  - Logické operátory AND (&), OR(|), použitie which(), ktoré vrátia indexy výberu (ktoré riadky spĺňajú podmienky)
    - X[(which(X$a>8) & (X$b = "fiat"| X$b = "ford")),]
- Pridanie nového stĺpca priradením alebo úprava existujúceho priradením
  - Napr. X$d = rnorm(10); X$d = cos(X$d)
- Použitie funkcií any, all na overenie vlastností hodnôt v tabuľke/vektore
  - any() – TRUE ak aspoň jedna hodnota spĺňa podmienku, inak FALSE
  - all() – TRUE ak všetky hodnoty spĺňajú podmienku, inak FALSE
- Triedenie vektora
  - Funkcia sort() … default je decreasing=FALSE
    - sort(X$a, decreasing=TRUE, na.last =TRUE)
- Usporiadanie tabuľky
  - Funkcia order() ... celý data frame sa preusporiada podľa vybraných premenných (môžeme ich vybrať aj viac, budú aplikované podľa poradia v order) …default ascen., - pre desc.
    - X[order(X$a,-X$c),]

Náhľad a sumarizácie s tabuľkou dát

- Náhľad dátovej tabuľky
  - head(X,n) .... prvých n riadkov, default n=6
  - tail(X,n) ... posledných n riadkov, default n=6
- Sumárny pohľad na dáta
  - summary(X) .... sumár informácií o tabuľke cez atribúty – pre numerické (min, max, median, mean, 3rdQu.,1stQu.), pre diskrétne (početnosti), ...
- Ďalšie sumarizácie
  - str(X) ... detailný pohľad na atribúty (používa sa pri rozkliku dát v časti Data v Enviroment tab v RStudiu)
  - quantile(X$a) …. kvantily (default kvartily) pre numerický atribút
    - Pre default nastavenie detto ako v summary, je však možné nastaviť vlastné rozdelenie
  - table(X$f, …) – štatistika početností rozdielnych hodnôt atribútu resp. kombinácií hodnôt rôznych atribútov (ak ich zadáme 2, či viac)

Balík **dplyr**

- https://cran.r-project.org/web/packages/dplyr/index.html
- Balík (s vlastnou syntaxou) pre manipuláciu z data frames
  - Optimalizovaná verzia **plyr** balíka, napojiteľná na SQL rozhranie cez DBI, data.table pre veľké tabuľky, ...
  - Funkčnosť nie je nová, zjednodušuje ale operácie, navyše sú **rýchle** (C++), má vlastnú „gramatiku“ pre manipulácie
- Operácie (args: data frame, ďalšie argumenty operácie, výstupný data frame)
  - select : vráti podmnožinu stĺpcov
  - filter : extrahuje podmnožinu riadkov na základe logických podmienok
  - arrange : mení poradie riadkov
  - rename : premenúva premenné
  - mutate : pridáva nové premenné alebo transformuje existujúce
  - summarise / summarize : generuje sumárne štatistiky rôznych premenných + je možné použiť „group_by“
  - print : metóda na výpisy, zabraňuje výpisom veľkého množstva dát do konzoly
- Zápis je možné zjednodušiť použitím tzv. pipe operátora %>% pre jednoduché zreťazenie operácií (čiže urobím operáciu, s jej výsledkom urobím ďalšiu, atď.)

```
mydata  %>%
select(name, order, income_total) %>%
arrange(order, income_total) %>%
filter(income_total >= 16)
```