Jazyky pre Dátovú Analytiku (JDA)

Prednáška č.3

Reportovanie a aplikácie v R

- Rozšírené funkčnosti systému na báze R pre reportovanie a tvorbu aplikácií
  - Proces reportovania analýz
    - Dokumentácia procesu analýzy, reprodukovateľnosť výsledkov
    - Prezentácia výsledkov vo vhodnej forme napr. pre podporu rozhodovania vo firme
  - Proces vývoja aplikácií
    - Verzovanie projektov, vývoj skriptov a ich zdieľanie, tvorba vlastných balíkov, tvorba dokumentácie
  - Web aplikácie
    - Tvorba interaktívnych web rozhraní (analytických nástrojov) pre podporu rozhodovania
- Podpora v R pre reportovania a vývoj aplikácií
  - R markdown – tvorba dokumentov (HTML, DOCX, PDF + prezentácie)
  - R sweave – vkladanie kódu do Latex dokumentov
  - R shiny – framework pre vývoj web aplikácií
  - Podpora verzovania – v RStudiu git, svn
  - R documentation file – Rd súbory – dokumentácia
  - ...

Markdown

- Zjednodušená verzia značkovacích jazykov (t.j. ako jazykov HTML, XML, ...)
- Umožňuje zamerať sa na písanie namiesto formátovania
- Jednoduché / Minimalistické a intuitívne formátovacie elementy
- Markdown dokument je jednoducho konvertovateľný na valídne HTML (alebo iné formáty) pomocou existujúcich nástrojov
- Markdown (oficiálna stránka)
  http://daringfireball.net/projects/markdown/
- R markdown je integráciou R kódu a značkovacích anotácií cez markdown

R Markdown

- Umožňuje vytvárať dokumenty obsahujúce „živý“ R kód, ktorý sa vyhodnocuje ako časť spracovania značiek
- Výsledky R kódu sa vkladajú do R markdown dokumentu
- R markdown dokument následne môže byť konvertovaný do štandardného markdown dokumentu pomocou **knitr** a **pandoc**
- Obsah markdown dokument môže byť konvertovaný do HTML pomocou balíka **markdown** v R
- Pre vytvorenie markdown dokumentu je možné použiť ľubovoľný textový editor
- Proces R markdown --> markdown --> HTML sa veľmi dobre manažuje v prostredí RStudio
- Výstup môže byť nielen HTML, ale aj DOCX (treba Word alebo OpenOffice “Word“), PDF (Latex!! – miktex vo Win)
- Je možné robiť aj prezentácie (ioslides, slidy,beamer)

Vytvorenie R Markdown súboru

- Výstupný typ súboru sa definuje pomocou YAML hlavičiek = množina párov kľúč-hodnota
  - Začínajú a končia - - -
  - Medzi nimi sú metadáta o vytváranom dokumente
  - Dôležité sú name (názov) a output (typ výstupu)
- Rstudio ponúka vytvorenie R markdown dokumentu rôznych typov
  - Cez File -> New File -> R Markdown
  - R studio nastaví hlavičku za vás
  - Output typy:
    - html_document – html file (web page)
    - pdf_document – pdf súbor
    - word_document – Microsof Word .docx
    - beamer_presentation – beamer prezentácia (pdf)
    - ioslides_presentation; slidy_presentation – ioslides alebo slidy prezentácia (html)
    - ... interaktívne verzie docs s použitím Rshiny ... html_document alebo ioslides_presentation s uvedeným runtime: shiny ako ďalšie metadáta

Základné prvky Markdown

- Zvýraznenie textu
  - Italic - *italic* alebo _italic_
  - Bold - **bold** alebo __bold__
  - Prečiarknuté písmo ~~slovo~~, horný index x^2^
- Nadpisy
  - # Header 1 ... ## Header 2 ... ### Header 3 ... atď
- Zoznamy
  - Nečíslované zoznamy

```
* Item 1
* Item 2
    + Item 2a
    + Item 2b
```

  - Číslovaný zoznam

```
1. Item 1
2. Item 2
    + Item 2a
    + Item 2b
```

Jednoduchý zoznam

```
- Item 1
- Item 2
- Item 3
```

Pre napísanie pomlčky: - -
Pre dlhšiu verziu: - - -
Pre „trojbodku“ : ...

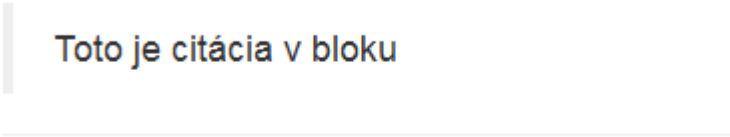
Základné prvky Markdown (2)

- Linky
  - Priamo uvedieme linku ... http://example.com
  - alebo s vl. textom ... [moj text](http://example.com)
- Obrázky

  ![moj text](http://example.com/logo.png)

  ![moj text](figures/img.png)
- Blok „citácie“

  > Toto je citácia v bloku

- Horizontálna čiara

  ****
- Tabuľky

  Názov 1 | Názov 2 | Názov 3

  ------- | ------- | -------

  Hodnota 1.1 | Hodnota 1.2 | Hodnota 1.3

  Hodnota 2.1 | Hodnota 2.2 | Hodnota 2.3

Názov 1	Názov 2	Názov 3
Hodnota 1.1	Hodnota 1.2	Hodnota 1.3
Hodnota 2.1	Hodnota 2.2	Hodnota 2.3
- Inline rovnice (Latex)

  $A = \pi*r^{2}$ $A = \pi * r^2$

Vkladanie R kódu - **knitr**

- Inline vloženie kódu

  Dva plus dva je `r 2 + 2`.

Dva plus dva je 4.

- Kúsky kódu v texte (code chunks)

  ```` {r}
  dim(iris)
  ```

```
dim(iris)
```

```
## [1] 150   5
```

- Nastavenia pre evaluáciu a výpis
  - eval ... TRUE/FALSE – vypíše/nevypíše „výpočet“ (def. TRUE)
  - echo ... TRUE/FALSE – vypíše/nevypíše „zdroják“ (def. TRUE)

  Príklad:

  ```` {r, eval=FALSE}
  dim(iris)
  ```

```
dim(iris)
```

- Ďalšie nastavenia viď. knitr balík resp. R markdown reference guide (napr. error, warning, message, cache, tidy, fig.* dev a out pre grafy, … )

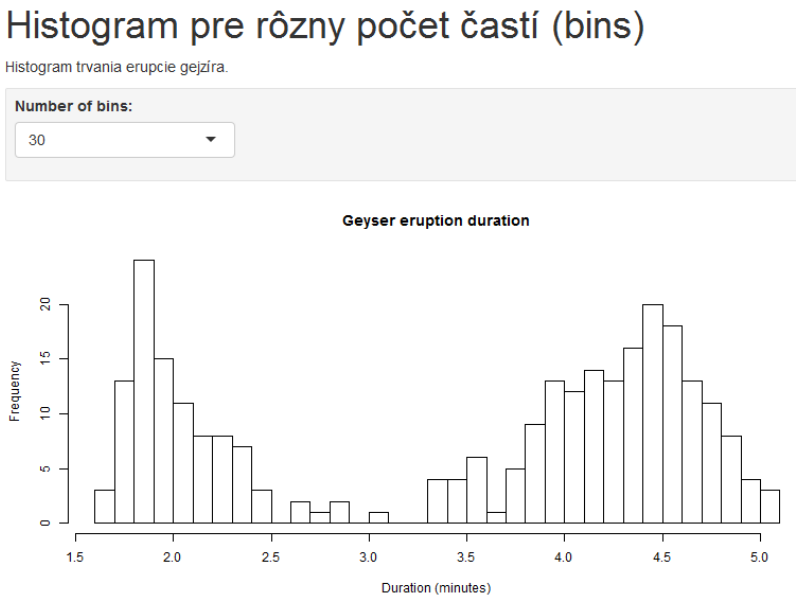
# Renderovanie výstupu R Markdown

- Vytvorenie (render) svojho výstupného reportu (súboru) je možné dosiahnuť
  - 1. Spustením rmarkdown::render("myfile.Rmd")
  - 2. Kliknutím na knit HTML button (alebo iného pre pdf/word podľa tam dostupného nastavenia) v rámci Rstudia pri otvorenom súbore v rozhraní
- Pri volaní render funkcie v R
  - Spustí (vyhodnotí) každý vložený kód a výsledok vloží do reportu
  - Vytvorí novú verziu reportu v danom výstupnom type súboru (output file type)
  - Otvorí náhľad výstupného súboru (Viewer tab) resp. podľa typu súboru v rámci daného programu (pdf prehliadač, word)
  - Uloží výstupný súbor v pracovnom adresári

# Vytvorenie interaktívneho HTML doc

- Interaktívne dokumenty – s použitím RShiny prvkov, ktoré vložíme do dokumentu
  - (viac detailov o Rshiny viď. neskôr)
- Postup:
  - Pridanie runtime: shiny do yaml hlavičky
  - Vloženie Shiny input a render funkcií
    - input funkcie pre načítanie požadovaných vstupov
    - render funkcie pre vizualizáciu výsledkov vzhľadom k vstupom
  - Spustenie cez render pre vytvorenie interaktívneho HTML dokumentu(2 spôsoby)
    - rmarkdown::run alebo „Run document“ button v Rstudiu
    - Výstup je buď html_document alebo ioslides_presentation

# Príklad – interaktívny dokument




```
---
title: "Histogram pre rôzny počet častí (bins)"
runtime: shiny
output: html_document
---

Histogram trvania erupcie gejzíra.

```{r, echo=FALSE}
inputPanel(
  selectInput("n_breaks", label = "Number of bins:",
              choices = c(10, 20, 30), selected = 30)
)

renderPlot({
  hist(faithful$eruptions, breaks = as.numeric(input$n_breaks),
       xlab = "Duration (minutes)", main = "Geyser eruption
duration")
})
```
```

# Príklad – prezentácia (cez ioslides)

```
---
title: "Moja prezentácia„
output: beamer_presentation
---

## Slajd s odrážkami

- Názov 1
- Názov 2
- Názov 3

## Slajd s R kódom a výstupom

```{r}
summary(cars)
```

## Slajd s grafom

```{r, echo=FALSE}
plot(cars)
```
```

Moja prezentácia

Slajd s odrážkami

- Názov 1
- Názov 2
- Názov 3

2/4

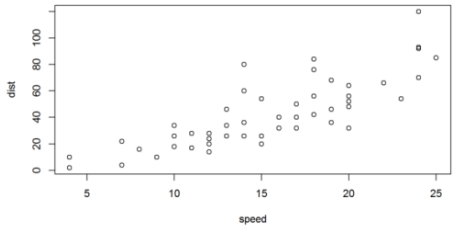
Slajd s R kódom a výstupom

```
summary(cars)
```

```
##      speed          dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

3/4

Slajd s grafom




4/4

# Ďalšie možnosti prípravy dokumentov

- Vytváranie Latex dokumentov priamo v RStudiu s možnosťou vkladania R kódu
  - R Sweave dokument
- Rd súbory (documentation file) – tvorba dokumentačného súboru pre funkciu v R
- Použitie šablón (template) pre R Markdown dokumenty
  - Tufte Handouts (knihy formátované podľa Edward Tufte štýlu)
  - Vignettes – dlhá dokumentácia balíkov v R
- Generovanie HTML(PDF,Word) priamo z obsahom R skriptu (tzv. „notebook“)
  - rmarkdown::render("nazovskriptu.R")
  - Použitím Compile notebook tlačítka v rozhraní editora

# R Shiny – web aplikácie

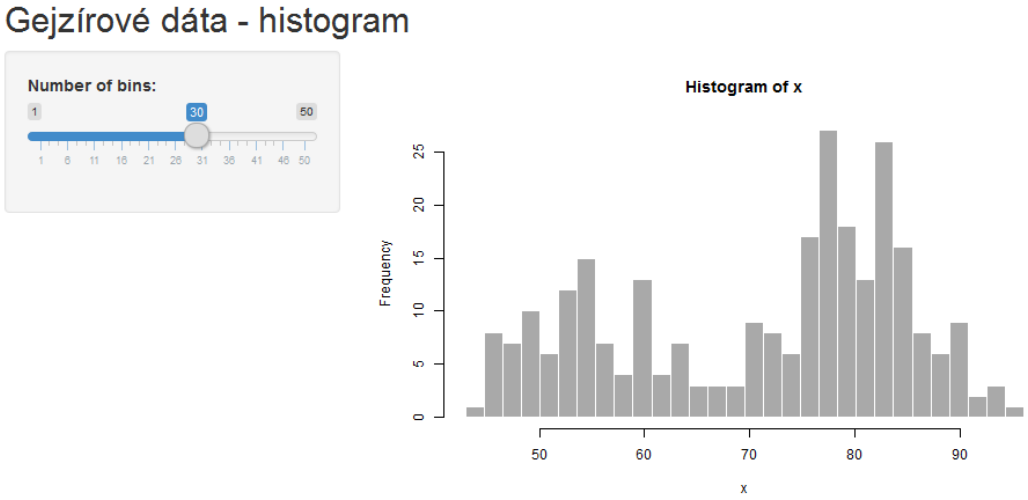
- Shiny je platforma pre vytváranie interaktívnych R programov vložených do web stránok => framework pre tvorbu web aplikácií v R => umožňuje premeniť analýzu na interaktívnu web aplikáciu
  - Príklad: Vytvoríme predikčný algoritmus a pomocou Shiny jednoducho vytvoríme web rozhranie so vstupným formulárom, ktorý podľa vložených vstupov spustí algoritmus a zobrazí výsledky analýzy.
- Pomocou Shiny je čas na vytvorenie webových interaktívnych produktov dátovej analýzy v R výrazne minimalizovaný
  - Nedosahuje však flexibilitu komplexných riešení s náročnejším softvérom (avšak je to často postačujúce pre potreby používateľov), komplexný softvér je často aj finančne náročný
- Shiny rozvíjajú tvorcovia R Studia, preto je integrácia s Rstudiom na na vysokej úrovni

# R Shiny – prerekvizity

- Aj keď Shiny nevyžaduje priamo znalosti HTML, CSS a JS (javascript), ako vždy pri web programovaní sa aspoň ich malá znalosť hodí
  - HTML – určuje štruktúru web stránky, rozdelenie na sekcie a značkovacie inštrukcie (markup)
  - CSS – poskytuje definície štýlov
  - JS – umožňuje interaktivitu
- Existuje veľké množstvo ľahkých tutoriálov / úvodov na webe pre tieto témy
- Shiny používa **bootstrap** framework pre štýl web stránok, ktoré sú na výstupe => ide o kombináciu HTML, CSS a JS vytvárajúcich peknú prezentáciu webu, ktorá je dobre zobrazovaná aj na mobilných platformách

# R Shiny projekt

- Shiny projekt je adresár (definuje zároveňnový pracovný adresár) obsahujúci súbory:
  - (v RStudiu … New Project -> New Directory -> Shiny Web Application vytvorí tieto základné súbory) – potom RunApp
  - ui.R – súbor pre kontrolu používateľského rozhrania klienta (vnorené R funkcie vytvárajúce UI rozhranie)
  - server.R – kontroluje napĺňanie obsahu rozhrania na serverovej časti aplikácie



# Príklad – server.R

```
library(shiny)
shinyServer(function(input, output) {
  output$distPlot <- renderPlot({
    # generuje bins (rozdelenia) na základe input$bins z ui.R
    x    <- faithful[, 2]
    bins <- seq(min(x), max(x), length.out = input$bins + 1)
    # kresli histogram pre specifikovany pocet binov
    hist(x, breaks = bins, col = 'darkgray', border = 'white')
  })
})
```

# Príklad – ui.R

```
library(shiny)
shinyUI(fluidPage(
        # Nazov aplikacie
        titlePanel("Gejzírové dáta - histogram"),
        # Layout – vzhľad – typu sidebar
        sidebarLayout(
                # sidebar panel – panel so slajderom pre vyber poctu binov
                sidebarPanel(
                        sliderInput("bins", "Number of bins:", min = 1,
                        max= 50,value = 30)
                ),
                # Hlavny panel so zobrazenim generovanej distribucie
                mainPanel(
                        plotOutput("distPlot")
                )
        )
))
```

# Alternatíva – jeden súbor app.R

```
library(shine)
ui <- fluidPage( ….
….. obsah z ui.R ……
….. )
server <- function(input,output){
  ... obsah z vnútra funkcie v server.R ....
}
shinyApp(ui=ui,server=server) # toto netreba pri ui.R a
    server.R verzii
```

Alternatívna definícia – jeden súbor app.R => „Šablóna“ Shiny aplikácie v tomto prípade:

```
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
```

- Ďalšie extra (voliteľné) súbory / adresáre aplikácie (app.R alebo rozdelenej na ui.R a server.R)
  - global.R – definuje globálne objekty spoločné pre ui aj server
  - DESCRIPTION, README ……. popisné súbory
  - <iné súbory> ….. dáta, skripty, atď.
  - www .... adresár pre súbory zdieľané na webe (obrázky, CSS, JS, ...)

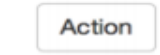
# Funkcie vstupu/výstupu/výpisu

- Základné funkcie definované v rozhraniach sú typov
  - *Input() funkcie – vstupné funkcie pre UI prvky
  - *Output() funkcie – výstupné funkcie pre UI prvky pre evaluáciu výstupných hodnôt
  - render*() funkcie – renderovacie funkcie pre uloženie výstupov (výpis do HTML)
- Postup realizácie:
  - Pridáme vstupy/výstupy do UI cez *Input/*Output
  - Serveru určíme ako má vypísať (renderovať) výsledok v server funkcii, pričom:
    - Výstupy sú referované cez output$<id>
    - Vstupy sú referované cez input$<id>
    - Kód serverovej funkcie obalíme do render* funkcie pred uložením výsledku

# Vstupy od používateľa – prehľad

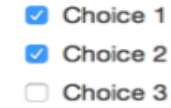
**Inputs - collect values from the user**

Access the current value of an input object with **input $<inputId>**. Input values are **reactive**.

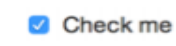



**actionButton**(inputId, label, icon, ...)


**actionLink**(inputId, label, icon, ...)




**checkboxGroupInput**(inputId, label, choices, selected, inline)




**checkboxInput**(inputId, label, value)

**dateInput**(inputId, label, value, min, max, format, startview, weekstart, language)

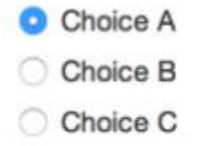
**dateRangeInput**(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)


**fileInput**(inputId, label, multiple, accept)

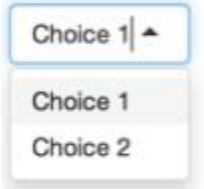

**numericInput**(inputId, label, value, min, max, step)



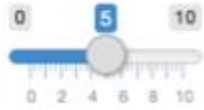

**passwordInput**(inputId, label, value)




**radioButtons**(inputId, label, choices, selected, inline)




**selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)




**sliderInput**(inputId, label, min, max, **value,** step, round, format, locale, ticks, animate, width, sep, pre, post)

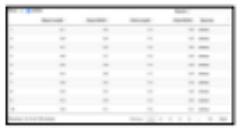

**submitButton**(text, icon)
(Prevents reactions across entire app)


**textInput**(inputId, label, value)

Užitočný zdroj: Shiny Cheat Sheet (pdf) - na shiny.rstudio.com

# Výstupné a renderovacie funkcie

**Outputs** - render*() and *Output() functions work together to add R output to the UI



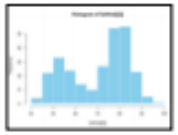
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

**dataTableOutput**(outputId, icon, …)

**renderImage**(expr, env, quoted, deleteFile)

**imageOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

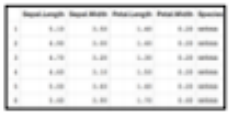


**renderPlot**(expr, width, height, res, …, env, quoted, func)

**plotOutput**(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

**renderPrint**(expr, env, quoted, func, width)

**verbatimTextOutput**(outputId)



**renderTable**(expr,…, env, quoted, func)

**tableOutput**(outputId)

foo

**renderText**(expr, env, quoted, func)

**textOutput**(outputId, container, inline)

**renderUI**(expr, env, quoted, func)

**uiOutput**(outputId, inline, container, …)

& **htmlOutput**(outputId, inline, container, …)

# Reaktívnosť prvkov v R Shiny

- Niekedy pre zrýchlenie aplikácie je užitočné reaktívne operácie (tie ktoré závisia na hodnotách vstupov vo vstupných elementoch) spúšťať mimo jednej render funkcie
  - Príklad: chceme využiť kód vo viacerých render* funkciách a neprepočítavať ju zakaždým - **reactive** funkcia slúži na tento účel

```
function(input, output) {
    x <- reactive({as.numeric(input$text1)+100})
    output$text1 <- renderText({x() })
    output$text2 <- renderText({x() + as.numeric(input$text2)})
}
```

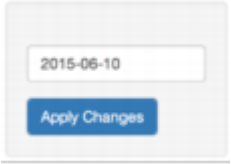
- Nereaktivita (?) reaktívnych prvkov
  - Niekedy nechceme aby Shiny reagoval okamžite a realizoval prepočet => občas chceme niečo ako submit tlačítko => na to slúži **isolate** funkcia
  - Príklad:

```
function(input, output) {
    output$text1 <- renderText({input$text1})
    output$text2 <- renderText({input$text2})
    output$text3 <- renderText({
      input$goButton
      isolate(paste(input$text1, input$text2))
    }
) }
```

# Kombinácia grafických elementov

- Je možné kombinovať viaceré elementy do panelov
  - Príklad:

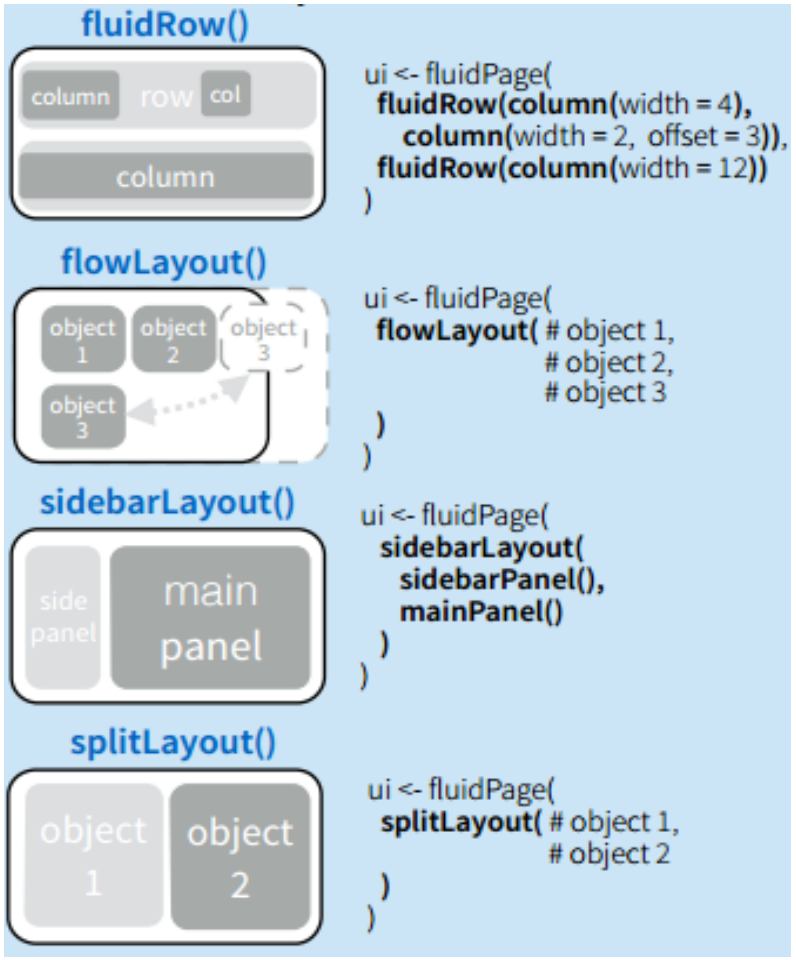
    wellPanel(dateInput("a", ""), submitButton())

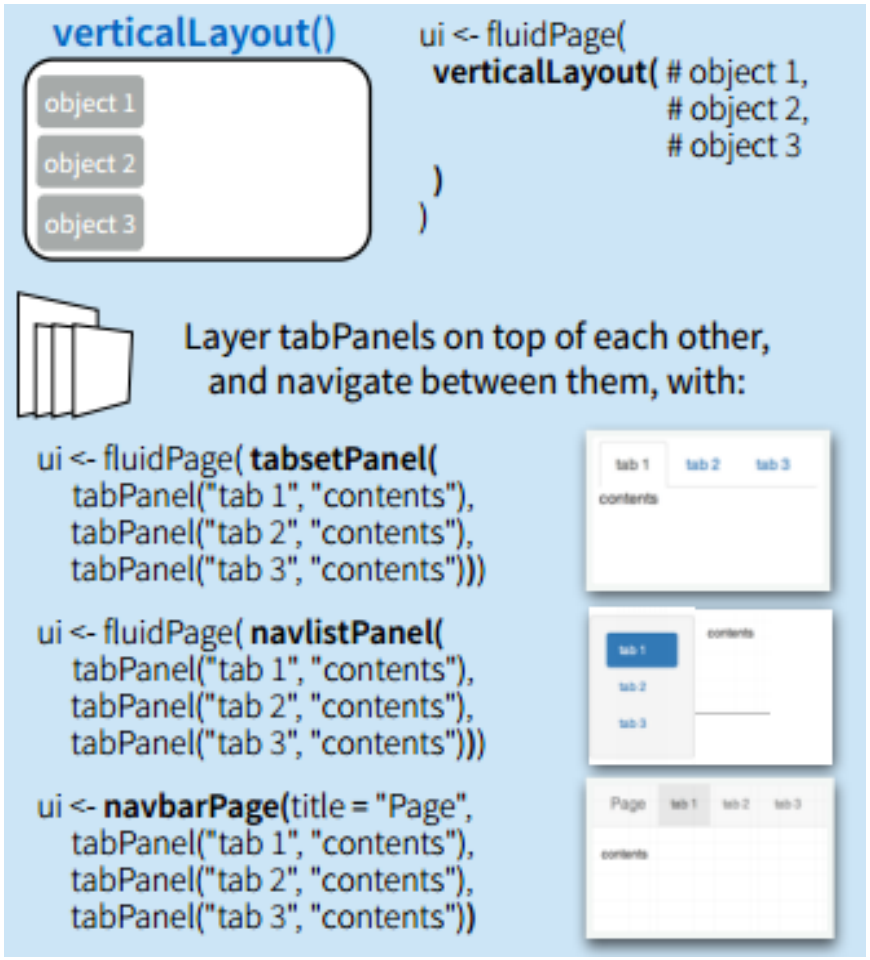



- Existujú  rôzne typy panelov (s prísluš. funkciami)
  - absolutePanel, fixedPanel ... panely s explicitným pozičným umiestnením
  - conditionalPanel ... viditeľný na základe splnenie podmienky
  - headerPanel, titlePanel ... názov aplikácie resp. panel pre názvy
  - inputPanel, wellPanel .... pre zgrupovanie vstupných elementov
  - mainPanel .... hlavná časť aplikácie pri vzhľade typu sidebar layout; doplňuje ho sidebarPanel pre panel na strane k hlavnému
  - navlistPanel ... navigačný panel pre navigačnú časť, obsahuje tabPanel elementy; podobne tabsetPanel obsahuje elementy tabPanel pre hlavnú časť aplikácie

# Celkový vzhľad stránky (layout)

- Layout = organizácia panelov a elementov do vzhľadu cez zvolenú (layout) funkciu (elementy/panely sú jej argumenty)
- Pre špecifický štýl -> HTML + CSS

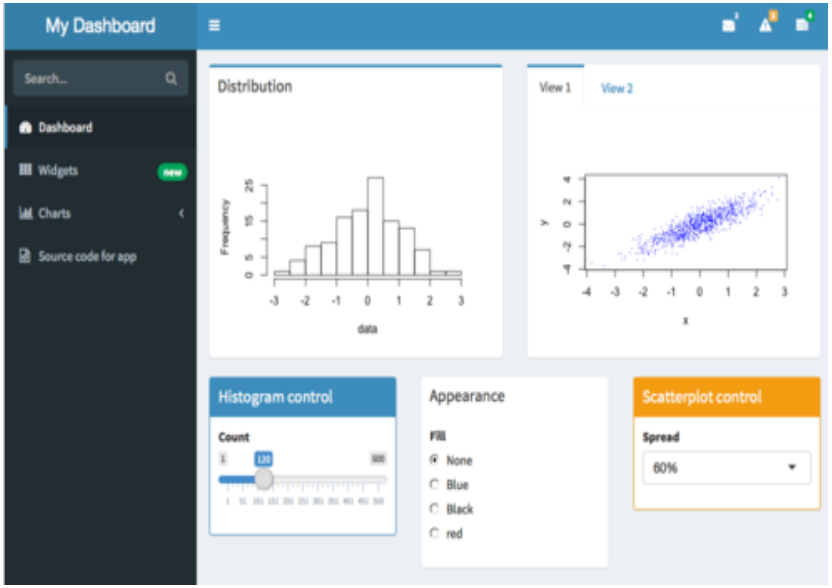


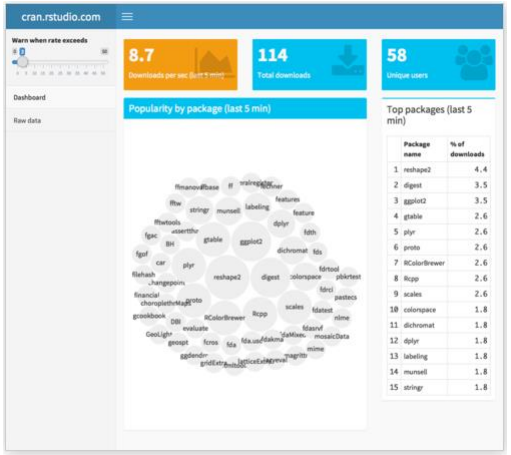



# R Shiny Dashboards

- Z hľadiska analytiky / BI (business intelligence) nástrojov je dnes dôležitá možnosť vytvárať **dashboard**-y
  - Zgrupujú hlavnú informáciu na úvodnej ploche, pre detaily je možné ísť na podčasti (tabs, widgets, ...)
- Je možné ich implementovať v RShiny
  - priamo cez RShiny pomocou kombinácie layoutov
  - Využiť balík **shinydashboard** (http://rstudio.github.io/shinydashboard/)
- Príklady aplikácií (nielen dashb.)

https://www.rstudio.com/products/shiny/shiny-user-showcase/




https://gallery.shinyapps.io/087-crandash

# Témy a HTML widgets

- V Shiny je možné využiť viacero tém pre vzhľad vytváranej stránky
  - Prednastavené alebo vlastné (resp. iné bootstrap témy) ... http://rstudio.github.io/shinythemes/
- Shinydashboard takisto umožňuje zmenu témy vzhľadu
  - Rôzne skin-y, typy panelov, farby, ikony, CSS štýly, http://rstudio.github.io/shinydashboard/appearance.html
- Je možné vytvárať / používať zobrazovacie prvky (elementy) „widgets“ na báze JS knižníc
  - Použijú sa JS vizualizačné knižnice v R konzole (ako plot), vložia sa do R markdown dokumentov, alebo do Shiny web aplikácií
  - Môžeme vytvoriť vlastný „widget“ pomocou framework-u prepájajúceho R a JavaScript
  - Knižnice JS ako: D3, Leaflet, dygraphs, MetricGraphics, networkD3, threejs, ...
  - Viac info: http://www.htmlwidgets.org/

# Projekty v RStudiu

- Vytvorenie projektu
  - RStudio projekt je asociovaný so zvoleným R pracovným adresárom, pričom môžeme vytvoriť projekt v:
    - Novom adresári (nový projekt)
    - Existujúcom adresári (obsahuje už napr. dáta a R kódy)
    - Napojením na repozitár projektu – adresár s projektom udržiavaný cez verzovanie (version control repository)
- Po vytvorení projektu sa:
  - Vytvorí projektový súbor *.Rproj
  - Skrytý adresár .Rproj.user pre projektovo špecifické dočasné súbory
  - Načíta sa projekt s novou R session
- Otvorenie projektu
  - Vytvorí sa nová R session, načíta sa projekt, špecifické súbory projektu (nastavenia, históriu, ...), nastaví sa working adresár na projektový, obnovia sa predtým otvorené súbory, ...
- Zatvorenie projektu (Close)
  - Uložia sa nastavenia, projektovo špecifické súbory, ukončí sa R session pre daný projekt

# Verzovanie projektu

- Kontrola a ukladanie verzií je veľmi dôležité tak
  - v prípade kolaboratívnej tímovej práce na projekte
  - ako aj pre podporu individuálnej práce na zložitejších analýzach / aplikáciách
- Rstudio má integrovanú podporu pre dva systémy verzovania
  - Git (http://git-scm.com/)
  - Subversion / SVN (http://subversion.apache.org/)
- Pre použitie verzovania v Rstudiu
  - Najprv potrebujete mať v operačnom systéme nainštalované softvéry Git a/alebo Subversion (viď. Inštalačky na stránkach softvérov)
  - Využiť Git/SVN pre verzovanie
    - Načítať existujúci projekt s verzovaním
    - Vytvoriť nový projekt klonovaním kódu z repozitára Git alebo checkout-om z repozitára SVN
    - Využiť ukladanie projektu do repozitára po zmenách

# R balíky a ich zdieľanie

- Čo je R balík?
  - Mechanizmus pre rozšírenie základnej funkcionality R
  - Kolekcia R funkcií alebo iných (dátových) objektov
  - Je systematicky organizovaný pre zabezpečenie konzistencie pri používaní
  - R balíky sú vytvárané používateľmi/vývojármi po celom svete
- Kde sú balíky k dispozícii
  - Primárne CRAN repozitár ... s inštaláciou cez install.packages()
  - Takisto sú často sprístupňované cez verzovacie systémy ako GitHub, Bitbucket, Gitorious, atď. ... s inštaláciou cez napr. install_github() z devtools balíka
  - Stiahnuteľné zo stránok napr. ako zip a nainštalovateľné lokálne
  - Nie je nutné vlastný balík vložiť do centrálneho repozitára, avšak je to užitočné pre ostatných ktorý si ho chcú nainštalovať (ak je to náš cieľ)

# Užitočnosť zdieľania vytvorených balíkov

- Prečo zdieľať kód balíkov ? => súčasťou nie je len zdieľanie kódov ... pri dodržaní určitých minimálnych štandardov to má viacero výhod
- Dokumentácia funkcií / vignettes (dlhá dokumentácia, často s vysvetlením základov)
- Centralizované zdroje ako napr. CRAN
- Aspoň základné dodržanie štandardov pre spoľahlivosť a robustnosť
- Údržba / rozšírenia balíka
- Definícia rozhraní / jasné API
- Používatelia vedia že prinajmenšom sa balík správne načíta do prostredia R

# Proces vytvorenia vlastného balíka

- Vytvoríme kód pre svoje výpočty (nový predikčný algoritmus, nový vizualizačný postup, ... ) v podobe skriptu(ov) (.R súbory)
- Chceme tento kód zdieľať s ďalšími používateľmi v štruktúrovanej forme => môžeme tak urobiť vytvorením nového balíka
- Preto zakomponujeme R skript(y) do štruktúry R balíka
- Napíšeme dokumentáciu pre používateľské funkcie balíka (používateľom volané funkcie)
- Priložíme ďalšie materiály (príklady, demá, dátové množiny, tutoriály, ...)
- Zabalíme všetko do podoby R balíka!

  (Dôležité: pre vytváranie balíkov treba nainštalovať Rtools + pre dokumentáciu treba Latex – napr. miktex v prípade Windows)
- Podpora vytvárania balíkov je súčasťou RStudia

# Základné prvky R balíka

- Vytvorenie R balíka sa začína vytvorením adresára (podobne ako pri Shiny projekte) s názvom balíka
- R balík musí obsahovať DESCRIPTION súbor zhrňujúci informácie o balíku
- Samozrejme – R kód(y) (v podadresári s názvom R)
- Dokumentácia (v podadresári man/ sub-directory)
- NAMESPACE (voliteľné, ale tradične sa vytvára) – deklarácia importovaných/exportovaných funkcií
- Všetky požiadavky resp. nastavenia  viď. dokumentácia vytvárania balíkov = „Writing R Extensions“

# DESCRIPTION súbor

- **Package**: Názov balíka pri volaní ( library(name) )
- **Title**: Úplný názov balíka
- **Description**: Dlhší popis balíka, zvyčajne v jednej vete (ale môže byť aj viac)
- **Version**: Číslo verzie (zvyčajne M.m-p formát)
- **Author**, **Authors@R**: Meno(á) autora(ov) balíka
- **Maintainer**: Meno + email osoby ktorá zabezpečuje údržbu balíka
- **License**: Licencia zdrojového kódu

Často použité voliteľné polia v súbore (predtým sú povinné)

- **Depends**: R balíky na ktorých závisí vytvorený balík
- **Suggests**: Voliteľné (odporúčané) balíky ktoré by mohol / chcel používateľ mať nainštalované
- **Date**: Dátum vydania verzie v YYYY-MM-DD formáte
- **URL**: Web stránka balíka

# Dokumentácia funkcií

- Dokumentačné súbory (.Rd) sa ukladajú do man podadresára
- Sú napísané v špecifickom značkovacom jazyku
- Vyžaduje sa dokumentovať každú exportovanú funkciu
  - Tiež dobrý dôvod limitovať počet exportovaných funkcií
- Môžeme dokumentovať ďalšie veci ako koncepty, prehľad balíka, referencie, atď. + vytvoriť dlhú dokumentáciu (vignette)
- Príklad:

```r
# Hello, world!
#
# This is an example function named 'hello'
# which prints 'Hello, world!'.
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'

hello <- function() {
  print("Hello, world!")
}
```

```
\name{hello}
\alias{hello}
\title{Hello, world!}
\usage{
hello()
}
\description{
Prints 'Hello, world!'.
}
\examples{
hello()
}
```

hello.Rd ▾ Find in Topic

hello {mypackage} R Documentation

## Hello, World!

### Description

Prints 'Hello, world!'.

### Usage

```
hello()
```

### Examples

```
hello()
```

# Dokončenie (vybudovanie) balíka

- Dokončenie (zabalenie) je možné buď v RStudiu (build/check), alebo cez programy príkazového riadku (volania cez *system* v R)
  - R CMD build – system("R CMD build mypackage") -vytvorí archívny tar-gz súbor
  - R CMD check – spustí overenie balíka (či má všetko čo treba) – system("R CMD check mypackage“)
- Spustiť ich môžeme aj priamo v príkazovom riadku operačného systému
- Overenie balíka – R CMD sputí súbor testov kde sa vyžaduje aby
  - Existovala dokumentácia (všetkých povinných typov + deklarované musia byť tiež ok)
  - Kód je možné načítať (load), nie sú žiadne problémy s kódom(chyby)
  - Príklady v dokumentácií sú spustiteľné
  - Všetky testy musia byť úspešné pre zaradenie do CRAN
  - Proces overenia viď.: http://r-pkgs.had.co.nz/check.html