Мастер академске студије
Информациони инжењеринг

Статистика у информационом инжењерингу

Напредни концепти језика R

(материјали за предавања)

Садржај

Атрибути

Атрибут

- обележје
- додатни опис објекта
  - објекат може имати више придодатих атрибута
- поједини атрибути уграђени у објекте одређеног типа

Атрибути

Уграђени атрибути

- атрибут **`names`**
  - користи за означавање елемената вектора, компоненти листе и колона у скупу података
- атрибут **`dim`**
  - код матрица и вишедимензионалних низова
    - вишедимензионални низ је вектор са придруженом димензијом чије су вредности поређане у редоследу попуњавања колона у вишедимензионалном низу
  - користи се за исказивање дужине по појединачним димензијама
- атрибут **`dimnames`**
  - код матрица и вишедимензионалних низова
  - користи се за означавање по појединачним димензијама

Атрибути

Уграђени атрибути

- атрибут **`levels`**
  - код фактора (неуређених) и код уређених фактора
  - користи се за означавање нивоа
- атрибут **`row.names`**
  - код скупова података
  - користи се за означавање редова у скупу података
- атрибут **`tsp`**
  - користи се за пратеће податке у временским серијама
    - почетак, крај, учесталост

Атрибути

Уграђени атрибути

- атрибут **`comment`**
  - користи се чување пратећег описа за објекат
- атрибут **`class`**
  - користи се за означавање класе објеката
    - потребно у оквиру система класа и уграђених механизама за објектно програмирање

Атрибути – Радње

- преглед доступних атрибута
  - помоћу функције **`attributes(...)`**
- очитавање и промена вредности доступних атрибута
  - помоћу функције **`naziv-atributa(objekat)`**
    - **`naziv-atributa`** означава атрибут чија вредност жели очитати или променити може користити скраћени назив атрибута
    - **`objekat`** означава објекат чији атрибут користи
  - помоћу функције **`attr(objekat, "naziv-atributa")`**
- додавање и уклањање атрибута
  - помоћу функције **`attr(objekat, "naziv-atributa")`**

Атрибути – Радње

```
> niz <- array(c(15,25,23,16,26,24), dim=c(3, 1, 2),
dimnames=list(c("jut", "pod", "vec"), c("temp"), c("SU", "NS")))
> attributes(niz)
$dim
[1] 3 1 2

$dimnames
$dimnames[[1]]
[1] "jut" "pod" "vec"

$dimnames[[2]]
[1] "temp"

$dimnames[[3]]
[1] "SU" "NS"


> dim(niz)
[1] 3 1 2
>
```

КОНЗОЛА

Атрибути – Радње

```
> m <- matrix(seq(1, 11, 2), ncol=2, nrow=3)
> m
     [,1] [,2]
[1,]    1    7
[2,]    3    9
[3,]    5   11
> dim(m) <- c(2, 3)
> m
     [,1] [,2] [,3]
[1,]    1    5    9
[2,]    3    7   11
> dim(m) <- NULL
> m
[1]  1  3  5  7  9 11
>
```

КОНЗОЛА

Атрибути – Радње

```
> p <- c(1, 2, 3)
> attr(p, "vreme") <- "00:30"
> attr(p, "vreme")
[1] "00:30"
> p
[1] 1 2 3
attr(,"vreme")
[1] "00:30"
> attr(p, "vreme") <- NULL
> p
[1] 1 2 3
> attributes(p)
NULL
>
```

КОНЗОЛА

Садржај

Уграђена подршка за објектно програмирање

- главни системи класа
  - S3 систем класа
    - првобитни систем класа
    - заснован на систему класа из верзије 3 језика *S* (*S3*)
    - подржани неки од основних концепата објектног програмирања
    - коришћен за развој многих класа које су уграђене у језик *R*
  - S4 систем класа
    - нови систем класа
    - заснован на систему класа из верзије 4 језика *S* (*S4*)
    - знатно шира подршка за објектно програмирање него код *S3* система
    - препоручени систем за развој нових класа
  - мутабилни систем класа
    - новији систем класа
    - заснован на S4 систему класа
    - допушта мутабилност инстанци и енкапсулацију метода у класама

Објектно програмирање

Уграђена подршка за објектно програмирање

- имплементација система класа
  - заснива на употреби атрибута
  - посебно битан атрибут **`class`**
    - вектор знаковних вредности
    - садржи називе класа које су наслеђене

S4 систем класа

- подржани концепти
  - класе
    - обичне
    - апстрактне
  - поља
  - методе
    - конструктори
  - полиморфизам
  - композиција класа
  - наслеђивање класа
    - једноструко
    - вишеструко

S4 систем класа – Класе

- дефинисање класе
  - помоћу функције **`setClass(...)`**
    - задавање назива класе
    - задавање поља класе
      - поље (слот, енгл. *slot*) поседује назив и тип
      - забрањени називи поља
        - резервисане речи
        - резервисани називи за поља (**`class`**)
    - задавање родитељских класа
    - ...

```
1 setClass("Testiranje",
2          representation(
3            naziv="character",
4            nivo="numeric"))
5
```

УЛАЗ

S4 систем класа – Инстанцирање

- инстанцирање класе
  - помоћу функције **`new(...)`**
    - задавање назива класе
    - задавање вредности поља
      - није обавезно задавање вредности за свако поље
      - интерно користи функција **`initialize(...)`** ако постоји као метода
        - поставља инстанцу у почетно стање на задати начин

S4 систем класа – Инстанцирање

инстанцирање класе

```
1 setClass("Testiranje",
2          representation(
3            naziv="character",
4            nivo="numeric"))
5 obj <- new("Testiranje",
6            naziv="Dankanov test",
7            nivo=0.01)
```

УЛАЗ

```
> obj
An object of class "Testiranje"
Slot "naziv":
[1] "Dankanov test"

Slot "nivo":
[1] 0.01

>
```

КОНЗОЛА

S4 систем класа – Поља

приступање пољима инстанце

преко потпуног назива поља

помоћу оператора **@** у облику **objekat@naziv-polja**

помоћу функције **slot(objekat, "naziv-polja")**

```
1 setClass("Testiranje",
2          representation(
3            naziv="character",
4            nivo="numeric"))
5 obj <- new("Testiranje",
6            naziv="Dankanov test",
7            nivo=0.01)
```

УЛАЗ

```
> obj@naziv
[1] "Dankanov test"
> obj@nivo <- 0.05
> slot(obj, "nivo")
[1] 0.05
>
```

КОНЗОЛА

S4 систем класа – Поља

преглед поља

називи поља помоћу функције **slotNames(objekat)**

класе поља помоћу функције **getSlots("naziv-klase")**

```
1 setClass("Testiranje",
2          representation(
3            naziv="character",
4            nivo="numeric"))
5 obj <- new("Testiranje",
6            naziv="Dankanov test",
7            nivo=0.01)
```

УЛАЗ

```
> slotNames(obj)
[1] "naziv" "nivo"
> getSlots("Testiranje")
      naziv        nivo
"character"   "numeric"
>
```

КОНЗОЛА

S4 систем класа – Конверзија инстанце

- конверзија инстанце из једне у другу класу
  - потребно направити конверзиону функцију
    - помоћу функције **`setAs(...)`**
      - задавање полазне класе
      - задавање циљне класе
      - задавање функције за конверзију
  - потребно позвати конверзиону функцију
    - помоћу функције **`as(...)`**
      - задавање полазног објекта за конверзију
      - задавање циљне класе

S4 систем класа – Конверзија инстанце

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3 setAs("Testiranje", "character",
4       function(from) {c(from@naziv, as.character(from@nivo))})
5 obj <- new("Testiranje", naziv="Dankanov test", nivo=0.05)
6
```

УЛАЗ

```
> class(obj)
[1] "Testiranje"
attr(,"package")
[1] ".GlobalEnv"
> typeof(obj)
[1] "S4"
> cha <- as(obj, "character")
> class(cha)
[1] "character"
> typeof(cha)
[1] "character"
> cha
[1] "Dankanov test" "0.05"
>
```

КОНЗОЛА

S4 систем класа – Исправност инстанце

- задавање функције за проверу исправности инстанце
  - приликом дефинисања класе
  - накнадно помоћу функције **`setValidity("naziv-klase", funkcija)`**
- провера исправности инстанце
  - аутоматски при инстанцирању класе
    - потребно претходно задати функцију за проверу исправности инстанце
    - инстанцирање не дозвољава уколико покушава формирати неисправна инстанца
  - помоћу функције **`validObject(instanca)`**

S4 систем класа – Исправност инстанце

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3 obj <- new("Testiranje", naziv="Dankanov test", nivo=0.01)
4
```

УЛАЗ

```
> setValidity("Testiranje", function(object) {object@nivo <= 1 &&
object@nivo >= 0})
Class "Testiranje" [in ".GlobalEnv"]

Slots:

Name:      naziv      nivo
Class: character   numeric
> validObject(obj)
[1] TRUE
> obj@nivo <- -1
> validObject(obj)
Error in validObject(obj) : invalid class “Testiranje” object:
FALSE
>
```

КОНЗОЛА

S4 систем класа – Припадност инстанце

- провера припадности инстанце класи
  - помоћу атрибута **`class`**
  - помоћу функције **`is(objekat, "naziv-klase")`**
    - подразумева и непосредну припадност класи
    - подразумева и посредну припадност класи преко наслеђивања

S4 систем класа – Припадност инстанце

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3 obj <- new("Testiranje", naziv="Dankanov test", nivo=0.01)
4
```

УЛАЗ

```
> class(obj)
[1] "Testiranje"
attr(,"package")
[1] ".GlobalEnv"
> is(obj, "Testiranje")
[1] TRUE
>
```

КОНЗОЛА

S4 систем класа – Методе

- механизам генеричких функција
  - генеричка функција
    - механизам да се исти назив функције може користити за различите конкретне функције које обрађују различите врсте аргумената
  - метода
    - конкретна функција која обрађује аргументе одређених врста
    - везује се за генеричку функцију
    - може позивати преко назива повезане генеричке функције
  - препоручена употреба
    - позивати генеричку функцију
    - не позивати непосредно методу

S4 систем класа – Дефинисање методе

дефинисање генеричке функције

- може искористити постојећа генеричка функција
- може трансформисати постојећа функција у генеричку помоћу **`setGeneric(...)`**

```
naziv <- function(object, ...) {telo}
setGeneric("naziv")
```

задавање методе за генеричку функцију

- додавање методе за генеричку функцију помоћу **`setMethod(...)`**

```
setMethod("naziv-generic-funkcije", signature="naziv-klase", definition=metoda)
```

  - **`metoda`** је функција која представља методу класе

S4 систем класа – Дефинисање методе

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3 rename <- function(object, name) {object@name <- name}
```

УЛАЗ

```
> setGeneric("rename")
[1] "rename"
> setMethod("rename", signature="Testiranje",
definition=function(object, name) {object@naziv <- name; object})
> obj <- new("Testiranje", naziv="Dankanov test", nivo=0.01)
> obj <- rename(obj, "Tukijev test")
> obj
An object of class "Testiranje"
Slot "naziv":
[1] "Tukijev test"

Slot "nivo":
[1] 0.01

>
```

КОНЗОЛА

S4 систем класа – Преглед метода

- преглед доступних метода по класи
  - помоћу команде **showMethods(classes="naziv-klase")**
- преглед доступних метода по генеричкој функцији
  - помоћу команде **showMethods("naziv-funkcije")**

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3
```

УЛАЗ

```
> showMethods(classes="Testiranje")
> library(stats4)
> showMethods("summary")
Function: summary (package base)
object="ANY"
object="mle"

>
```

КОНЗОЛА

S4 систем класа – Наслеђивање

- наслеђивање класа
  - наслеђују поља
  - наслеђују методе
  - дозвољено вишеструко наслеђивање

S4 систем класа – Наслеђивање

- механизам за наслеђивање класа
  - приликом дефинисање класе
    - помоћу функције **`setClass(...)`**
      - задавање родитељских класа преко аргумента **`contains`**
  - накнадно
    - помоћу функције **`setIs(...)`**
    - помоћу функције **`setAs(...)`**
      - уз задавање поступка конверзије објеката
- однос наслеђивања између две класе
  - провера постојања односа наслеђивања
    - помоћу функције **`extends("naziv-potomka", "naziv-pretka")`**

S4 систем класа – Наслеђивање

```
1 setClass("Testiranje",
2          representation(naziv="character", nivo="numeric"))
3 setClass("ZavrsenoTestiranje",
4          representation(odbacuje="logical"),
5          contains="Testiranje")
6 objA <- new("Testiranje", naziv="Dankanov test", nivo=0.01)
7 objB <- new("ZavrsenoTestiranje", odbacuje=T)
```

УЛАЗ

```
> objB
An object of class "ZavrsenoTestiranje"
Slot "odbacuje":
[1] TRUE

Slot "naziv":
character(0)

Slot "nivo":
numeric(0)

>
```

КОНЗОЛА

S4 систем класа – Наслеђивање

```
> setClass("A", representation(s="logical"))
> setClass("B", representation(sb="logical"), contains="A")
> setClass("C", representation(sc="logical"), contains="A")
> setClass("D", representation(sd="logical"), contains=c("B", "C"))
> (obj <- new("D"))
An object of class "D"
Slot "sd":
logical(0)

Slot "sb":
logical(0)

Slot "s":
logical(0)

Slot "sc":
logical(0)

> extends("D", "A")
[1] TRUE
>
```

КОНЗОЛА

S4 систем класа – Апстрактне класе

- механизам уније класа
  - креирање заједничке натклассе (апстрактне класе) за дате класе
  - формирање уније класе
    - помоћу функције **`setClassUnion(...)`**
  - додавање класе у унију класа
    - помоћу функције **`setIs(...)`**
  - провера да ли класа представља унију класа
    - помоћу функције **`isClassUnion(...)`**
- апстрактна класа
  - не садржи податке
  - не дозвољава инстанцирање
  - дозвољава задавање метода

S4 систем класа – Апстрактне класе

```
1 setClass("ParametarskoTestiranje",
2          representation(naziv="character", nivo="numeric"))
3 setClass("NeparametarskoTestiranje",
4          representation(naziv="character", nivo="numeric"))
5 setClassUnion("Testiranje",
6               c("ParametarskoTestiranje",
7                 "NeparametarskoTestiranje"))
```

УЛАЗ

```
> isClassUnion("Testiranje")
[1] TRUE
> extends("NeparametarskoTestiranje", "Testiranje")
[1] TRUE
> obj <- new("NeparametarskoTestiranje", naziv="Vilkoksonov test",
nivo=0.05)
> is(obj, "Testiranje")
[1] TRUE
> new("Testiranje")
Error in new("Testiranje") :
  trying to generate an object from a virtual class ("Testiranje")
>
```

КОНЗОЛА

S3 систем класа

- подржани концепти
  - класе
  - атрибути
  - методе
  - полиморфизам

S3 систем класа – Класе и инстанце

- дефинисање класе
  - постављање вредности атрибута **`class`** за одговарајући објекат
    - помоћу команде **`class(object) <- "naziv-klase"`**
  - помоћу функције **`setOldClass(...)`**
    - за потребе коришћења класе система *S3* у оквиру класа система *S4*
- инстанца класе
  - објекат који има придружен атрибут **`class`** са одговарајућом вредношћу

S3 систем класа – Класе и инстанце

```
> obj <- list(naziv="Tukijev test", nivo=0.05)
> class(obj) <- "Testiranje"
> obj
$naziv
[1] "Tukijev test"

$nivo
[1] 0.05

attr(,"class")
[1] "Testiranje"
>
```

КОНЗОЛА

S3 систем класа – Методе

- механизам генеричких функција
  - генеричка функција
    - прихвата објекат као први аргумент
    - класа објекта може варирати од случаја до случаја
    - класа аргумента одређује која ће метода даље бити позвана
  - метода
    - обрађује прихваћени објекат одређене класе
    - назив методе по шаблону
      - **`naziv-generičke-funkcije.naziv-klase`**
        - **`naziv-klase`** одговари класи објекта који метода обрађује
    - може постојати подразумевана метода
      - обрађује све објекте за које не постоји одговарајућа метода
  - препоручена употреба
    - позивати генеричку функцију
    - не позивати непосредно методу

S3 систем класа – Дефинисање методе

- дефинисање генеричке функције
  - дефинисање функције која садржи позив **`UseMethod(...)`**
    - **`naziv <- function(object, ...) UseMethod("naziv")`**
  - функција **`UseMethod(...)`** прослеђује аргументе одговарајућој методи
    - метода изабрана на основу класе првог аргумента
    - очекује да назив методе буде у очекиваном облику **`naziv.naziv-klase`**
- задавање методе за генеричку функцију
  - може користити и нека од уграђених генеричких функција
  - дефинисање функције са називом у очекиваном облику
    - **`naziv.naziv-klase <- function(object, ...) {telo}`**
      - очекује да класа аргумента **`object`** буде класа са називом **`naziv-klase`**

S3 систем класа – Дефинисање методе

```
> obj <- list(naziv="Tukijev test", nivo=0.05)
> class(obj) <- "Testiranje"
> obj
$naziv
[1] "Tukijev test"

$nivo
[1] 0.05

attr(,"class")
[1] "Testiranje"
> print.Testiranje <- function(object, ...) {cat(object[[1]],
object[[2]], "\n")}
> obj
Tukijev test 0.05
>
```

КОНЗОЛА

S3 систем класа – Преглед метода

- преглед доступних метода по класи
  - помоћу команде **`methods(class="naziv-klase")`**
- преглед доступних метода по генеричкој функцији
  - помоћу команде **`methods("naziv-funkcije")`**

```
> methods(class="dist")
[1] as.matrix format    labels    print
see '?methods' for accessing help and source code
> methods("labels")
[1] labels.default     labels.dendrogram* labels.dist*
[4] labels.lm*         labels.terms*
see '?methods' for accessing help and source code
>
```

КОНЗОЛА

Мутабилни систем класа

- подржани концепти
  - класе
  - поља
  - методе
  - композиција класа
  - наслеђивање класа
    - једноструко
    - вишеструко

Мутабилни систем класа – Класе

- дефинисање класе
  - помоћу функције **`setRefClass(...)`**
    - задавање назива класе
    - задавање пољâ класе
    - задавање метода класе
    - задавање родитељских класа

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4
                                                    УЛАЗ
```

Мутабилни систем класа – Инстанцирање

- инстанцирање класе
  - помоћу методе **`new(...)`**
    - у контексту генераторске функције
      - која добија приликом дефинисања класе
    - задавање вредности пољâ

Мутабилни систем класа – Инстанцирање

инстанцирање класе

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4 robj <- Testiranje$new(naziv="Fiserov test",
5                        nivo=0.001)
```

УЛАЗ

```
> robj
Reference class object of class "Testiranje"
Field "naziv":
[1] "Fiserov test"
Field "nivo":
[1] 0.001
>
```

КОНЗОЛА

Мутабилни систем класа – Поља

- приступање пољима класе
  - преко назива поља
    - помоћу оператора у облику **`objekat$naziv-polja`**

Мутабилни систем класа – Поља

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4 robj1 <- Testiranje$new(naziv="Fiserov test", nivo=0.001)
5 robj2 <- robj1
6
```

УЛАЗ

```
> robj1$naziv
[1] "Fiserov test"
> robj2$naziv
[1] "Fiserov test"
> robj2$naziv <- "Tukijev test"
> robj1$naziv
[1] "Tukijev test"
> robj2$naziv
[1] "Tukijev test"
>
```

КОНЗОЛА

Мутабилни систем класа – Поља

преглед поља класе

помоћу методе **`fields()`**

у контексту генераторске функције

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4
```

УЛАЗ

```
> Testiranje$fields()
      naziv        nivo
"character"   "numeric"
>
```

КОНЗОЛА

Мутабилни систем класа – Методе

- дефинисање методе
  - приликом дефинисања класе
  - накнадно после дефинисања класе
    - помоћу методе **`methods()`**
      - у контексту генераторске функције
- позивање методе
  - над објектом класе
    - помоћу оператора **`$`** у облику **`objekat$naziv-metode(argumenti)`**
- коришћење поља у оквиру методе
  - приступ пољима преко назива
  - измена вредности поља помоћу оператора нелокалне доделе **`<<-`**

Мутабилни систем класа – Методе

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4 robj <- Testiranje$new(naziv="Fiserov test", nivo=0.001)
5
```

УЛАЗ

```
> robj$preimenovanje("Tukijev test")
Error in envRefInferField(x, what, getClass(class(x)), selfEnv) :
  ‘preimenovanje’ is not a valid field or method name for reference
class “Testiranje”
> Testiranje$methods(preimenovanje=function(naz) {naziv <<- naz})
> robj$preimenovanje("Tukijev test")
> robj
Reference class object of class "Testiranje"
Field "naziv":
[1] "Tukijev test"
Field "nivo":
[1] 0.001
>
```

КОНЗОЛА

Мутабилни систем класа – Методе

- преглед метода класе
  - помоћу методе **`methods()`**
    - у контексту генераторске функције

Мутабилни систем класа – Методе

преглед метода класе

```
1 Testiranje <- setRefClass("Testiranje",
2                           fields = list(naziv="character",
3                                         nivo="numeric"))
4 Testiranje$methods(
5   preimenovanje=function(naz)
6     {naziv <<- naz})
```

УЛАЗ

```
> Testiranje$methods()
 [1] ".objectPackage" ".objectParent"  "callSuper"       "copy"

 [5] "export"         "field"          "getClass"
"getRefClass"
 [9] "import"         "initFields"     "preimenovanje"   "show"

[13] "trace"          "untrace"        "usingMethods"
>
```

КОНЗОЛА

Помоћне радње

- провера да ли објекат поседује класу
  - помоћу функције **`is.object(...)`**
- провера да ли је објекат инстанца неке класе система *S4*
  - помоћу функције **`isS4(...)`**
- приступ податку о класи
  - помоћу функције **`class(...)`**
  - помоћу функције **`oldClass(...)`**
- уклањање класе
  - помоћу функције **`unclass(...)`**

Објектно програмирање

Уграђене класе

`vector`	**`character`**	**`logical`**	**`complex`**
`integer`	**`double`**	**`single`**	**`numeric`**
`raw`	**`NULL`**	**`function`**	**`expression`**
`list`	**`data.frame`**	*(и тако даље)*	

Садржај

Руковање изузецима

Систем стања

- стање (енгл. *condition*)
  - посебна ситуација до које може доћи приликом извршавања
  - на појаву стања може се реаговати
- главне врсте стања
  - грешке
  - упозорења
  - поруке

Руковање изузецима

Пријава настанка проблема

главне функције

`stop(...)`

формира грешку

прекида евалуацију тренутног израза

приказује обавештење

`warning(...)`

формира упозорење

не прекида евалуацију тренутног израза

приказује обавештење

`message(...)`

формира поруку

приказује обавештење

Руковање изузецима

Пријава настанка проблема

```
 1 quadeq <- function(cs) {
 2   if(length(cs) != 3) {
 3     stop("three coefficients required")
 4   } else if(cs[1] == 0) {
 5     stop("not a quadratic equation")
 6   } else {
 7     root <- sqrt(as.complex(cs[2]^2 - 4 * cs[1] * cs[3]))
 8     denom <- 2 * cs[1]
 9     c((-cs[2] + root) / denom, (-cs[2] - root) / denom)
10   }
11 }
```

УЛАЗ

```
> quadeq(c(1, 3))
Error in quadeq(c(1, 3)) : three coefficients required
> quadeq(c(1, 3, -4)) # x^2 + 3*x - 4 = 0
[1]  1+0i -4+0i
> quadeq(c(1, 3, 4)) # x^2 + 3*x + 4 = 0
[1] -1.5+1.322876i -1.5-1.322876i
> quadeq(c(0, 3, 4))
Error in quadeq(c(0, 3, 4)) : not a quadratic equation
>
```

КОНЗОЛА

Руковање изузецима

Пријава настанка проблема

```
 1 quadeq <- function(cs) {
 2   if(length(cs) != 3) {stop("three coefficients required")
 3   } else if(cs[1] == 0) {stop("not a quadratic equation")
 4   } else {
 5     root <- sqrt(as.complex(cs[2]^2 - 4 * cs[1] * cs[3]))
 6     if(root == 0) {warning("one solution only")
 7     } else if(Im(root) != 0) {message("complex solutions")
 8     }
 9     denom <- 2 * cs[1]
10     c((-cs[2] + root) / denom, (-cs[2] - root) / denom)
11   }
12 }
```

УЛАЗ

```
> quadeq(c(1, 2, 1))
[1] -1+0i -1+0i
Warning message:
In quadeq(c(1, 2, 1)) : one solution only
> quadeq(c(1, 3, 4))
complex solutions
[1] -1.5+1.322876i -1.5-1.322876i
>
```

КОНЗОЛА

Руковање изузецима

Пријава настанка проблема

помоћне функције

`stopifnot(...)`

формира грешка ако неки од аргумената нема вредност TRUE
погодно за проверу аргумената прослеђених функцији

```
1 razdalj <- function(x, y) {
2   stopifnot(is.numeric(x), is.numeric(y))
3   sqrt(x^2 + y^2)
4 }
```

УЛАЗ

```
> razdalj(6, 8)
[1] 10
> razdalj("6", "8")
 Error in razdalj("6", "8") : is.numeric(x) is not TRUE
>
```

КОНЗОЛА

Руковање изузецима

Реаговање на настали проблем

главне функције

`try(...)`

аутоматски обрађује насталу грешку
омогућује наставак извршавања

```
1 f <- function(a, b) {
2   print(a + b)
3   print(a < b)
4 }
5 g <- function(a, b) {
6   try(print(a + b))
7   print(a < b)
8 }
```

УЛАЗ

```
> f(100, "a")
Error in a + b : non-numeric argument to binary operator
> g(100, "a")
Error in a + b : non-numeric argument to binary operator
[1] TRUE
>
```

КОНЗОЛА

Руковање изузецима

Реаговање на настали проблем

главне функције

`tryCatch(...)`

обрађује насталу грешку, упозорење или поруку

дозвољава задавање обрађивачâ преко аргумената **`error`**, **`warning`** и **`message`**

дозвољава обрађивач који увек активира а задаје преко аргумента **`finally`**

омогућује наставак извршавања

```
> mean("a")
[1] NA
Warning message:
In mean.default("a") : argument is not numeric or logical:
returning NA
> catchw <- function(x){print("upozorenje"); print(x)}
> tryCatch(mean("a"), warning=catchw)
[1] "upozorenje"
<simpleWarning in mean.default("a"): argument is not numeric or
logical: returning NA>
>
```

КОНЗОЛА

Руковање изузецима

Реаговање на настали проблем

помоћне функције

`suppressWarnings(...)`

занемарује упозорења која настају приликом евалуације аргумента

```
> mean("a")
[1] NA
Warning message:
In mean.default("a") : argument is not numeric or logical:
returning NA
> suppressWarnings(mean("a"))
[1] NA
>
```

КОНЗОЛА

Руковање изузецима

Реаговање на настали проблем

помоћне функције

`options(...)`

аргумент **`error`**
задавање израза на основу којег обрађују обичне грешке

аргумент **`warn`**
подешавање приказа обавештења за упозорења

`warn < 0` – занемаривање, **`warn == 0`** – приказ након првобитног позива

`warn == 1` – приказ по настанку, **`warn >= 2`** – упозорења постају грешке

аргумент **`warning.expression`**
задавање израза који треба бити евалуиран у случају појаве упозорења

аргумент **`warning.length`**
подешавање највеће дозвољене дужине обавештења за грешке и упозорења

```
> options(warn=-1)
> mean("a")
[1] NA
>
```

КОНЗОЛА

Садржај

Отклањање проблема

Откривање места настанка проблема

- интерактивно праћење извршавања
  - функција **`browser()`**
  - функције **`debug(...)`**, **`undebug(...)`** и **`debugonce(...)`**
- преглед остварених позива функције
  - функције **`trace(...)`** и **`untrace(...)`**
  - функција **`traceback()`**

Отклањање проблема

Интерактивно праћење извршавања

- режим интерактивног праћења извршавања
  - извршавање привремено зауставља
  - могућност испитивања активног окружења и затечених вредности
    - дозвољени редовни изрази
  - могућност контроле наставка извршавања
    - подржане посебне наредбе
  - могућност интерактивног праћења у оквиру интерактивног праћења

Отклањање проблема

Интерактивно праћење извршавања

контрола извршавања у режиму интерактивног праћења

- команда **`n`**
  - дозвола извршавања наредне наредбе
    - у случају позива функције, не изводи интерактивно праћење наредби функције
- команда **`s`**
  - дозвола извршавања наредне наредбе
    - у случају позива функције, изводи интерактивно праћење наредби функције
- команда **`f`**
  - дозвола завршетка извршавања функције или петље
- команде **`c`** и **`cont`**
  - излазак из режима интерактивног праћења и наставак извршавања

Отклањање проблема

Интерактивно праћење извршавања

- контрола извршавања у режиму интерактивног праћења
  - команда **`Q`**
    - излазак из режима интерактивног праћења и прекид извршавања
  - команда **`where`**
    - приказ низа непосредно претходних позива функција
  - команда **`help`**
    - преглед посебних наредби које су подржане у режиму интерактивног праћења

Отклањање проблема

Интерактивно праћење извршавања

функција **`browser()`**

активирање режима интерактивног праћења извршавања

```
1 f <- function(a, b) {
2   browser()
3   print(a + b)
4   print(a < b)
5   print(a > b)
6 }
```

УЛАЗ

```
> f(1, 2)
Called from: f(1, 2)
Browse[1]> n
debug at #3: print(a + b)
Browse[2]> n
[1] 3
debug at #4: print(a < b)
Browse[2]> c
[1] TRUE
[1] FALSE
>
```

КОНЗОЛА

Отклањање проблема

Интерактивно праћење извршавања

- функција **`debug(...)`**
  - укључивање аутоматске активације режима интерактивног праћења при сваком позиву одабране функције
- функција **`undebug(...)`**
  - искључивање аутоматске активације режима интерактивног праћења при сваком позиву одабране функције
- функција **`debugonce(...)`**
  - укључивање аутоматске активације режима интерактивног праћења при наредном позиву одабране функције

Отклањање проблема

Интерактивно праћење извршавања

```
1 f <- function(a) {
2   stopifnot(a >= 0)
3   factorial(a)
4 }
```

УЛАЗ

```
> debugonce(f)
> f(3)
debugging in: f(3)
debug at #1: {
    stopifnot(a >= 0)
    factorial(a)
}
Browse[2]> n
debug at #2: stopifnot(a >= 0)
Browse[2]> c
exiting from: f(3)
[1] 6
> f(3.5)
[1] 11.63173
>
```

КОНЗОЛА

Отклањање проблема

Преглед остварених позива функције

функција **`traceback()`**

приказ низа позива који претходио необрађеној грешци

у хронолошки обрнутом редоследу

```
1 f <- function(a, b) {
2   print(a + b)
3   print(a < b)
4 }
5 g <- function(a, b) {
6   f(a, b)
7   print(a > b)
8 }
```

УЛАЗ

```
> g(100, "a")
Error in a + b : non-numeric argument to binary operator
> traceback()
3: print(a + b) at #2
2: f(a, b) at #2
1: g(100, "a")
>
```

КОНЗОЛА

Отклањање проблема

Преглед остварених позива функције

- функција **`trace(...)`**
  - активирање праћења позива за одабрану функцију
    - позив одабране функције пропраћен исписом у конзоли
- функција **`untrace(...)`**
  - деактивирање праћења позива за одабрану функцију

```
> trace("+")
> 100 + "a"
trace: 100 + "a"
Error in 100 + "a" : non-numeric argument to binary operator
> 100 + 100
trace: 100 + 100
[1] 200
> untrace("+")
> 100 + 100
[1] 200
>
```

КОНЗОЛА

Садржај

Извори и литература

Основни извори и литература

- R Project. An introduction to R – Notes on R: A programming environment for data analysis and graphics version 4.5.1 (2025-06-13). Internet: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf
- R Project. R language definition – Version 4.5.1 (2025-06-13) DRAFT. Internet: https://cran.r-project.org/doc/manuals/r-release/R-lang.pdf
- R Project. R: A language and environment for statistical computing – Reference index – The R Core Team – Version 4.5.1 (2025-06-13). Internet: https://cran.r-project.org/doc/manuals/r-release/fullrefman.pdf
- Adler J. R in a nutshell: A desktop quick reference. 2nd edition. O'Reilly; 2012.

Мастер академске студије
Информациони инжењеринг

Статистика у информационом инжењерингу

Напредни концепти језика R

(материјали за предавања)