

Uporaba paketa **seqinr** za doseg in manipulacijo zaporedij

A. Blejec

6. januar 2016

Naslovi zbirk zaporedij

Aminokislinska zap

<http://www.uniprot.org/>
nukleotidna

<http://www.ncbi.nlm.nih.gov/nuccore>

Za začetek ...

- 1 S spletnim iskalcem poiščite informacijo o paketu **seqinr**, ki je shranjena v okviru `r-project` v arhivu CRAN.
- 2 Oglejte si dokumentacijo (v obliki PDF) in preberite razlago za funkcijo `c2s`.
- 3 Do spletnih strani R-core group in CRAN lahko pridete tudi s povezavami v izbiri *Help* v programu R .

Paket **seqinr**

Pripravimo paket za uporabo (load)

```
library("seqinr")
```

Informacije o paketu in pomoč za funkcije (npr. `s2c()`)

```
library(help = seqinr)  
`?`(s2c)  
help(s2c)
```

Zdaj pa vi ...

Oglejte si funkcije in primere s strani s pomočjo:

- `s2c()` in `c2s()`
- `translate()`
- `aaa()`
- funkcije z začetkom `get*`
- `read.fasta()`, `write.fasta()`

Funkcije za pretvorbo med malimi in velikimi črkami

```
tolower("ATCG")
```

```
[1] "atcg"
```

```
toupper("actg")
```

```
[1] "ACTG"
```

Funkcije za pretvorbo niza znakov v vektor znakov

```
niz <- "atgatatcgtaa"
```

```
(vektor <- s2c(niz))
```

```
[1] "a" "t" "g" "a" "t" "a" "t" "c" "g" "t" "a" "a"
```

```
c2s(vektor)
```

```
[1] "atgatatcgtaa"
```

Izbor iz baze podatkov

Izberemo dva proteina iz Swissprot. Accession Q9CD83 in A0PQ23.

```
library(seqinr)  
choosebank("swissprot")  
s1 <- query("s1", "AC=Q9CD83")  
s1seq <- getSequence(s1$req[[1]])  
s2 <- query("s2", "AC=A0PQ23")  
s2seq <- getSequence(s2$req[[1]])  
closebank()
```


Struktura objekta

str(s1)

List of 6

```
$ call      : language query(listname = "s1", query = "AC=Q9CD8
$ name      : chr "s1"
$ nelem     : int 1
$ typelist: chr "SQ"
$ req       :List of 1
  ..$ :Class 'SeqAcnucWeb'  atomic [1:1] PHBS_MYCLE
  .. .. ..- attr(*, "length")= num 210
  .. .. ..- attr(*, "frame")= num 0
  .. .. ..- attr(*, "ncbigc")= num 1
$ socket    :Classes 'sockconn', 'connection'  atomic [1:1] 5
  .. ..- attr(*, "conn_id")=<externalptr>
- attr(*, "class")= chr "qaw"
```

Posamezne komponente lahko izvlečem na različne načine ([])

Izbor komponent

Izberem po imenu

```
str(s1$req)
```

List of 1

```
$ :Class 'SeqAcnucWeb' atomic [1:1] PHBS_MYCLE  
.. ..- attr(*, "length")= num 210  
.. ..- attr(*, "frame")= num 0  
.. ..- attr(*, "ncbigc")= num 1
```

Izberem prvo komponento

```
s1$req[[1]]
```

name	length	frame	ncbigc
"PHBS_MYCLE"	"210"	"0"	"1"

Poglejmo zaporedje

Začetek zaporedja

```
head(slseq, 30)
```

```
[1] "M" "T" "N" "R" "T" "L" "S" "R" "E" "E" "I" "R"  
[13] "K" "L" "D" "R" "D" "L" "R" "I" "L" "V" "A" "T"  
[25] "N" "G" "T" "L" "T" "R"
```

in konec zaporedja

```
tail(slseq, 30)
```

```
[1] "E" "E" "L" "D" "R" "C" "Q" "Y" "S" "N" "D" "I"  
[13] "D" "T" "R" "S" "G" "D" "R" "F" "V" "L" "H" "G"  
[25] "R" "V" "F" "K" "N" "L"
```

Zdaj pa vi ...

1. Kako bi spremenili zaporedje s1seq v niz znakov?

```
[1] "MTNRTLSREEIRKLRDLRILVATNGTLTRVLNVVANEIIVVDIINQQLLDVAPKIP"
```

Združevanje zaporedij v strukturo list

```
(vse <- list(s1, s2))
```

```
[[1]]
```

```
1 SQ for AC=Q9CD83
```

```
[[2]]
```

```
1 SQ for AC=A0PQ23
```

```
names(vse) <- c("Q9CD83", "A0PQ23")
```

```
vse
```

```
$Q9CD83
```

```
1 SQ for AC=Q9CD83
```

```
$A0PQ23
```

```
1 SQ for AC=A0PQ23
```

Izbor elementov iz strukture list

```
vse$Q9CD83$req
```

```
[[1]]
```

name	length	frame	ncbicg
"PHBS_MYCLE"	"210"	"0"	"1"

```
names(vse$Q9CD83)
```

[1] "call"	"name"	"nelem"	"typelist"
[5] "req"	"socket"		

```
sapply(vse, function(x) x$req)
```

```
$Q9CD83
```

name	length	frame	ncbicg
"PHBS_MYCLE"	"210"	"0"	"1"

```
$A0PQ23
```

name	length	frame
"A0PQ23_MYCUA"	"212"	"0"
ncbicg		
"1"		

Kako identificiramo gen

Poiščite zapis gena v zaporedju

```
s <- "SA*ALMAGML*A"
```

Algoritem

[1] "SA*ALMAGML*A"

- 1 Poišči prvi STOP kodon (*)
- 2 Odstrani stop kodon in vse pred njim: ALMAGML*A
- 3 Poišči prvi START kodon (M)
- 4 Odstrani vse pred njim: MAGML*A
- 5 Poišči prvi STOP kodon (*)
- 6 Ohrani vse od START do STOP kodona: MAGML*

1. Poišči prvi STOP kodon (*)

```
s
```

```
[1] "SA*ALMAGML*A"
```

```
start <- regexpr("\\*.*M", s)
```

```
start
```

```
[1] 3
```

```
attr(, "match.length")
```

```
[1] 7
```

```
attr(, "useBytes")
```

```
[1] TRUE
```

2. Odstrani stop kodon in vse pred njim: ALMAGML*A

s

```
[1] "SA*ALMAGML*A"
```

s2c(s)

```
[1] "S" "A" "*" "A" "L" "M" "A" "G" "M" "L" "*" "A"
```

1:start

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

```
(s1 <- s2c(s)[- (1:start)])
```

```
[1] "A" "L" "M" "A" "G" "M" "L" "*" "A"
```

3. Poišči prvi START kodon (M) in
4. Ohrani vse od njega naprej: MAGML*A

```
s1
```

```
[1] "A" "L" "M" "A" "G" "M" "L" "*" "A"
```

```
s1 == "M"
```

```
[1] FALSE FALSE TRUE FALSE FALSE TRUE FALSE FALSE  
[9] FALSE
```

```
which(s1 == "M")
```

```
[1] 3 6
```

```
which(s1 == "M")[1]
```

```
[1] 3
```

```
which(s1 == "M")[1]:length(s1)
```

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

```
(s2 <- s1[which(s1 == "M")[1]:length(s1)])
```

```
[1] "M" "A" "G" "M" "L" "*" "A"
```

5. Poišči prvi STOP kodon (*) in

6. Ohrani vse od START do STOP kodona: MAGML*

```
s2
```

```
[1] "M" "A" "G" "M" "L" "*" "A"
```

```
s2 == "*"
```

```
[1] FALSE FALSE FALSE FALSE FALSE TRUE FALSE
```

```
which(s2 == "*")
```

```
[1] 6
```

```
which(s2 == "*")[1]
```

```
[1] 6
```

```
1:which(s2 == "*")
```

```
[1] 1 2 3 4 5 6
```

```
(s3 <- s2[1:which(s2 == "*")[1]])
```

```
[1] "M" "A" "G" "M" "L" "*"
```

Spremeni vektor v niz znakov

```
s4 <- c2s(s3)
```

```
s
```

```
[1] "SA*ALMAGML*A"
```

```
s4
```

```
[1] "MAGML*"
```

Funkcija najdiGen()

Pripravimo funkcijo z aktivnimi ukazi iz primera (lahko tudi vsemi)

```
najdiGen <- function(s = "SA*ALMAGML*A") {  
+   start <- regexpr("\\*.M", s)  
+   (s1 <- s2c(s)[-1:start])  
+   (s2 <- s1[which(s1 == "M")[1]:length(s1)])  
+   (s3 <- s2[1:which(s2 == "*")[1]])  
+   s4 <- c2s(s3)  
+   s  
+   return(s4)  
+ }
```

Uporaba funkcije

```
s = "SA*ALMAGML*A"  
najdiGen(s)
```

```
[1] "MAGML*"
```

Pa še kak drug

```
s = "SGAMG*ASHTMALASMHGTA*SHMR"  
najdiGen(s)
```

```
[1] "MALASMHGTA*"
```

Poizkusite s še kakšnim zaporedjem

Regularni vzorci

Učinkovito iskanje nizov znakov je možno s pomočjo regularnih vzorcev (*regular expression*)

s

```
[1] "SGAMG*ASHTMALASMHGTA*SHMR"
```

```
sub("^.*\\|*.*(M.*\\|*){1}.*", "\\|1", s)
```

```
[1] "MALASMHGTA*"
```

ali pa s funkcijo `strsplit()`

```
paste(c("", strsplit(strsplit(s, "\\|*")[[1]][2],  
+ "M")[[1]][-1]), collapse = "M")
```

```
[1] "MALASMHGTA"
```


Zdaj pa vi ...

Na spletu poiščite informacijo o

```
names <- c("Q9CD83", "A0PQ23")  
names
```

```
[1] "Q9CD83" "A0PQ23"
```

in obe hkrati prikažite na eni spletni strani.
Oglejte si, kako je sestavljen URL.

Sestavljanje nizov znakov: paste()

Zlepim imena s primerno
predpono (AC),
ločilom (=)
in povezovalcem (OR).

```
names <- c("Q9CD83", "A0PQ23")  
paste("AC", names, sep = " = ", collapse = " OR ")
```

```
[1] "AC = Q9CD83 OR AC = A0PQ23"
```

Priprava URL ogled na spletni strani

```
accNames <- paste("accession%3A", names, sep = "",  
+ collapse = "+or+")  
accNames  
  
[1] "accession%3AQ9CD83+or+accession%3AA0PQ23"  
  
url <- paste("http://www.uniprot.org/uniprot/?query=",  
+ accNames, "&sort=score", sep = "")  
if (interactive()) shell.exec(url)
```

Izpis s funkcijo `write.fasta()`

Za izpis potrebujemo zaporedje v obliki vektorja znakov in eno opis za v prvo vrstico.

```
s1seq[1:8]  
[1] "M" "T" "N" "R" "T" "L" "S" "R"  
  
write.fasta(s1seq, name = "Prvo zaporedje ...",  
+ file = "testFasta.txt")  
write.fasta(s2seq, name = "Drugo zaporedje ...",  
+ file = "testFasta.txt", open = "a")
```

Kaj je v datoteki

```
dir(pattern = "t.*txt")
```

```
[1] "testFasta.txt"
```

```
readLines("testFasta.txt")
```

```
[1] ">Prvo zaporedje ..."  
[2] "MTNRTLSREEIRKLDRDLRILVATNGTLTRVLNVVANEEIVVDIINQQLLDVAPKI  
[3] "NLKIGRILQRDILLKGQKSGILFVAAESLIVIDLLPTAITTYLTKTHHPIGEIMAA  
[4] "TYKEDAQVWIGDLPCWLADYGYWDLPKRAVGRRYRIIAGGQPVIITTEYFLRSVFQ  
[5] "EELDRCQYSNDIDTRSGDRFVLHGRVFKNL"  
[6] ">Drugo zaporedje ..."  
[7] "MLAVLPEKREMTECHLSDEEIRKLNRLRILIAATNGTLTRILNVLANDEIVVEIVK  
[8] "DAAPEMDGCDHSSIGRVLRRDIVLKGRRSVIPFVAAESFIAIDLLPPEIVASLLET  
[9] "GEVMAASCIETFKEEAKVWAGESPAWLELDRRRNLPKVVGRQYRVIAEGRPVI  
[10] "LRSVFEDNSREEP  
IRHQRSVGT  
SARSGRSICT"
```

Branje fasta formata

```
zaporedji <- read.fasta("testFasta.txt")
summary(zaporedji)
```

	Length	Class	Mode
Prvo	210	SeqFastadna	character
Drugo	212	SeqFastadna	character

```
str(zaporedji)
```

List of 2

```
$ Prvo :Class 'SeqFastadna' atomic [1:210] m t n r ...
.. ..- attr(*, "name")= chr "Prvo"
.. ..- attr(*, "Annot")= chr ">Prvo zaporedje ..."
$ Drugo:Class 'SeqFastadna' atomic [1:212] m l a v ...
.. ..- attr(*, "name")= chr "Drugo"
.. ..- attr(*, "Annot")= chr ">Drugo zaporedje ..."
```

PUBLIC DATA ACCESS PACKAGES

- GEOquery — Get data from NCBI Gene Expression Omnibus (GEO)
- Geometadb — A compilation of metadata from NCBI GEO