R-programmering VT2022

Föreläsning 8

Josef Wilzén

2022-03-13

Linköpings Universitet

Föreläsning 8

Innehåll föreläsning 8

- Tentainfo
- Texthantering
  - `stringr`
  - regex
- Modern databearbetning
  - `tidyr`
  - `dplyr`

Tentainfo

- Tentamen är **2023-03-30, 4 h**.
- Skrivplats är en SU-sal, inget internet tillgängligt
- Endast anmälda studenter får skriva tentan!
- Tentan kommer att likna de gamla tentorna som finns på kursrummet på Lisam. Vissa förändingar:
  - 4 uppgifter totalt istället för 5 som det var tidigare
  - 3 uppgifter kommer att handla om att skriva funktioner, 1 uppgift kommer att handla om annat
- Hjälpmedel kommer vara ett antal "cheatsheets", de kommer finnas som pdf, ni ska **INTE** ta med några papper eller böcker.

Texthantering i R

Arbeta med strängar

- En sträng är en samling bokstäver
- R har ett antal inbyggda funktioner för att hantera text
  - `paste()`
  - `substr()`
  - `nchar()`
- Använder hellre paketet `stringr`
  - Enklare
  - Enhetligt

Arbeta med strängar - II

- `readLines(con = , encoding = )` används för att läsa in en text.
  - `con` är "connection" t.ex. vart en fil ligger
  - `encoding` är vilken text-kodning som används
    - `"latin1"` och `"utf8"` är vanligast.

Paketet stringr

Paketet stringr

- Ett paket med funktioner för strängar
  - Optimerade och effektiva funktioner
  - Funktioner börjar med `str_`
- Två delar:
  - Standard funktioner
  - Mönstermatchande funktioner

Grundläggande strängfunktioner

stringr	base	Användning
`str_sub()`	`substr`	substring, välj ut en av en sträng (regex)
`str_c()`	`paste()` `paste0()`	Slår ihop strängar
`str_split()`	`strsplit()`	Dela upp en sträng i flera element (regex)
`str_length()`	`nchar()`	Antal tecken
`str_trim()`		Tar bort mellanslag
`str_pad()`		Lägger till mellanslag

Mönstermatchning

Regular expression (regex)

Från Wikipedia:

> *A regular expression (shortened as regex) is a sequence of characters that specifies a search pattern in text. Usually such patterns are used by string-searching algorithms for "find" or "find and replace" operations on strings, or for input validation. It is a technique developed in theoretical computer science and formal language theory.*

- Notation för att beskriva strängar
  - Hitta en specifik del som uppfyller ett villkor
  - Textmanipulation
- Byggs upp av
  - literals: Vanliga bokstäver och siffror
  - metacharacters: Speciella regler

Regular expression: Metacharachters

Tecken	Betydelse
`.`	samtliga tecken (exkl. det “tomma” tecknet “ ”)
`^`	det “tomma” tecknet i början av en text
`$`	det “tomma” tecknet i slutet text
`*`	föregående tecken 0 eller fler gånger
`+`	föregående tecken 1 eller fler gånger
`?`	föregående tecken är valfritt
`{n,m}`	föregående tecken `n` eller max `m` gånger
`[...]`	teckenlista (character list)
`|`	ELLER
`(...)`	Gruppering
`\`	Används för att “undvika” metatecken/specialtecken.

Obs! I R krävs: `\\`

Regular expression: teckenklass

- Med `[ ]` skapas en lista över tänkbara tecken.
- Används för att identifiera en mängd av tecken
- Inom `[ ]` har bara följande meta-tecken en särskild betydelse

Tecken	Betydelse	Exempel
-	tecken	A-Z a-z 0-9
^	ICKE	`\^0-9`
\	specialtecken	`\t\n`

Obs! I R krävs: `\\`

Regular expression: teckenklass - II

Vanliga fördefinerade klasser, (se `?regexp`)

- `[:digit:]` Nummer
- `[:lower:]` gemener
- `[:upper:]` VERSALER
- `[:punct:]` tecken, ej bokstäver och siffror
- `[:space:]` mellanslag, tab, radbrytning, m.m.
- i R behöver vi ange att det är en teckenklass `[[:space:]]`

Lära sig regular expression

- Testa dina expressions
  - **https://regexr.com**
  - **https://www.regexpal.com**
- Roliga lekar med regex
  - Regex Golf **https://alf.nu/RegexGolf**
  - Regex crossword **https://regexcrossword.com**

Mönstermatchning i R

- `pattern` är ett regular expression i R

`stringr`	`base`	Användning
`str_detect()`	`grepl()`	Identifiera `pattern`, returnera logisk vektor
`str_locate()`	`gregexpr()`	Identifiera `pattern`, returnera position i texten
`str_replace()`	`gsub()`	Identifiera `pattern`, ersätter med ny text
`str_extract_all()`		Plocka ut alla strängar som uppfyller `pattern`

Modern databearbetning

Varför databeartbetning

- Datamängder blir bara större och större
- Smart hantering minskar arbetsbördan
- Smart hantering för bearbetningen snabb
- Analysfunktioner kräver särskilt format
- Skriv kod för människor

piping

Piping görs med %>%

```
z <- a %>%
  fun1(b) %>%
  fun3()
```

är samma som

```
x <- fun1(a,b)
z <- fun3(x)
```

tidyr: Tidy data

- Data är ofta “messy”
- Tidy data:
  - Varje kolumn en variabel
  - Varje rad en observation
- `tidyr` är ett paket för att konvertera “messy” till “tidy”
- Effektivt både minnesmässigt och beräkningsmässigt
- kommer bespara er mycket tid

dplyr

- Paket i R för att hantera **stora** datamängder.
- En liten uppsättning funktioner (verb) för datahantering.
- Väldigt optimerad kod för snabb och minneseffektiv hantering.
- Går att koppla till databaser och Spark.
- Lägger på klassen `tbl_df` till `data.frame`

dplyr verb

verb	beskrivning
`select()`	välj kolumn
`filter()`	filtrera rader
`arrange()`	arrangera rader
`mutate()`	skapa nya kolumner
`summarise()`	aggregera rader över grupp
`group_by()`	gruppera för “split-apply-combine”/aggregera
`join`	kombindera olika dataset
`bind_rows`	kombindera dataset “på höjden”
`bind_cols`	kombindera dataset “på bredden”

dplyr joins

- Slå ihop data är oftast centralt
- Inom databser talar man om “joins”

funktion	beskrivning
`left_join()`	slå ihop efter variabel, behåll obs. i vänstra data.frame
`right_join()`	slå ihop efter variabel, behåll obs. i högra data.frame
`full_join()`	slå ihop efter variabel, behåll alla obs.
`anti_join()`	slå ihop efter variabel, behåll obs. som inte finns i båda