

# R-programmering VT2022

## Föreläsning 8

---

Johan Alenlöv

2022-03-14

Linköpings Universitet

# Föreläsning 8

---

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

- Tentamen är **31/3 kl 8:00 – 12:00**.
- Skrivplats är en SU-sal.
- Endast anmälda studenter får skriva tentan!
- Tentan kommer att likna de gamla tentorna som finns på kurshemsidan.
- Hjälpmedel kommer vara ett antal “cheatsheets”, de kommer finnas som pdf, ni ska **INTE** ta med några papper.
- Inlämning är en R-fil per uppgift.
- **testsession** 22/3 8-10 i SU-17/18.

# Texthantering i R

---

- 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

- `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

---



- 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
<code>str_sub()</code>	<code>substr</code>	substring, välj ut en av en sträng (regex)
<code>str_c()</code>	<code>paste()</code> <code>paste0()</code>	Slår ihop strängar
<code>str_split()</code>	<code>strsplit()</code>	Dela upp en sträng i flera element (regex)
<code>str_length()</code>	<code>nchar()</code>	Antal tecken
<code>str_trim()</code>		Tar bort mellanslag
<code>str_pad()</code>		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: Metacharacters

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: \\

Vanliga fördefinierade klasser, (se `?regex`)

- `[: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:]`

- 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

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

# Modern databearbetning

---

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

Piping görs med %>%

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

är samma som

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

- 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

- 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.
- Läger på klassen `tbl_df` till `data.frame`

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

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

---

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

---