Datorlaboration 8

Josef Wilzén och Måns Magnusson

13 mars 2023

Instruktioner

- Denna laboration ska göras i grupper om **två och två**. Det är viktigt för gruppindelningen att inte ändra grupper.
- En av ska vara **navigatör** och den andra **programmerar**. Navigatörens ansvar är att ha ett helhetsperspektiv över koden. Byt position var 30:e minut. **Båda** ska vara engagerade i koden.
- Det är tillåtet att diskutera med andra grupper, men att plagiera eller skriva kod åt varandra är **inte tillåtet**. Det är alltså **inte** tillåtet att titta på andra gruppers lösningar på inlämningsuppgifterna.
- Använd gärna Teams för att ställa frågor. Det finns olika kanaler:
  - **Questions**: Skriv era frågor här. Svar kommer att ges öppet direkt i kanalen. Publicera inte kod till inlämningsuppgifter här (andra kan då se det). Det går bra att skriva frågor om inlämningsuppgifter här så länge ni inte inkluderar kod med lösningar till dessa uppgifter. Det går bra att publicera kod till övningsuppgifter här.
  - **Raise_your_hand**: Skriv här om ni vill ha hjälp men inte ställa er fråga öppet. Skriv något i stil med "Jag vill ha hjälp". Då kommer en lärare att kontaka er när de har tid (i chatten på Teams). Vill flera ha hjälp så bildar de olika kommentarerna en kö, och hjälp kommer att ges i ordning efter kön. En "tumme upp" på kommentaren innebär att läraren har börjat hjälpa den aktuella studenten. Ett "hjärta" på kommentaren innebär att läraren har hjälpt klart studenten.
- Använd inte `å`, `ä` eller `ö` i variabel- eller funktionsnamn.
- Utgå från laborationsmallen, som går att ladda ned **här** (obs ny mall jämfört med tidigare veckor), när du gör inlämningsuppgifterna. Spara denna som `labb[no]_grupp[no].R` , t.ex. labb5_grupp01.R om det är laboration 5 och ni är grupp 01. Ta inte med hakparenteser i filnamnet. Denna fil ska **inte** innehålla något annat än de aktuella funktionerna, namn- och ID-variabler och ev. kommentarer. Alltså **inga** andra variabler, funktionsanrop för att testa inlämningsuppgifterna eller anrop till markmyassignment-funktioner.
- Precis innan inlämning på Lisam, döp om er R-fil till en **`.txt`** fil, detta görs för att kunna skicka in filen till Ouriginal för plagieringskontroll. Exempel: `labb5_grupp01.R` blir då `labb5_grupp01.txt` Ladda upp den filen (som slutar på .txt) på Lisam under rätt inlämning innan deadline.
- Laborationen består av två delar:
  - Datorlaborationen (= övningsuppgifter)
  - Inlämningsuppgifter
- I laborationen finns det extrauppgifter markerade med *. Dessa kan hoppas över.
- Deadline för laboration framgår på LISAM
- **Tips!** Använd "fusklapparna" som finns här. Dessa kommer ni också få ha med på tentan.

Innehåll

Del I

Datorlaboration

Kapitel 1

Texthantering och regular expression i R med `stringr`

Gå igenom följande delar i *Handling and Processing Strings in R* (av Gaston Sanchez) och testa koden i exemplen.

Kap 2 Hela

Kap 3 3.1, 3.3

Kap 4 4.2.1 - 4.2.3

Kap 5 5 - 5.2.2, 5.2.6, 5.3.1-5.3.2

Kap 6 6-6.1.3, 6.2.2, 6.4-6.4.10

Kap 7 7.1, 7.2

Boken finns fritt tillgänglig här.

Kapitel 2

Modern datahantering

I följande kapitel gås de verktyg som idag är state-of-the-art för att snabbt och effektivt bearbeta stora datamängder i R.

2.1 Piping med %>%

När vi arbetar med databearbetning av stora datamaterial kan innebär det ofta ett stort antal funktionsanrop. För att göra en databearbetningsprocess överskådlig och snabb finns så kallade pipes, eller ”rör” i R för att skicka datamaterial i ett flöde av olika modifikationer. Pipingoperatorn innbär att

```
z <- a %>% fun1(b) %>% fun2(c) %>% fun3()
```

är exakt samma sak som

```
x <- fun(a, b)
y <- fun(x, c)
z <- fun3(y)
```

Detta flöde kan ofta göra det tydligare hur data bearbetas i olika databearbetningssteg.

2.2 `tidyr`

Datamaterial kan många gånger komma i ett otal olika tabellstrukturer. Alla statistiska metoder, databearbetningsverktyg som `dplyr` och visualiseringspaket som `ggplot2` kräver att datamaterialet är i ett så kallat `tidy` format. För att konvertera olika tabeller och datamaterial till ett `tidy` format används paketet `tidyr` och funktionen `gather()`.

Gå igenom och reproducera koden i följande introduktionstext.

2.3 `dplyr`

R-paketet dplyr har under kort tid att bli det huvudsakliga verktyget för att arbeta med större datamängder i R. Det finns framförallt tre anledningar till dess popularitet.

1. Paketet har bara ett fåtal funktioner för att arbeta med data vilket gör det snabbt att lära sig.
2. `dplyr` är skrivet i kraftigt optimerad C++ kod vilket gör hanteringen av stora datamängder snabbare än något annat statistik- eller analysverktyg.
3. `dplyr` kan kopplas mot databaser för att direkt bearbeta större datamängder. dplyrs verb används också i `sparlyr`, vilket är ett paket för att hantera data som inte får plats på enskilda datorer. Att lära sig `dplyr` är således en approach som möjliggör att hantera i princip hur stora datamaterial som helst.

Gå igenom och reproducera koden i följande introduktionstext.

Del II

Inlämningsuppgifter

Inlämning

Utgå från laborationsmallen, som går att ladda ned här, när du gör inlämningsuppgifterna. Spara denna som `labb[no]_grupp[no].R` , t.ex. `labb5_grupp01.R` om det är laboration 5 och ni tillhör grupp 1. Ta inte med hakparenteser i filnamnet. Denna fil ska laddas upp på LISAM och ska **inte** innehålla något annat än de aktuella funktionerna, namn- och ID-variabler och ev. kommentarer. Alltså **inga** andra variabler, funktionsanrop för att testa inlämningsuppgifterna eller anrop till markmyassignment-funktioner.

Precis innan inlämning på Lisam, döp om er R-fil till en **`.txt`** fil, detta görs för att kunna skicka in filen till Ouriginal för plagieringskontroll. Exempel: `labb5_grupp01.R` blir då `labb5_grupp01.txt` Ladda upp den filen (som slutar på .txt) på Lisam under rätt inlämning innan deadline.

Tips!

Inlämningsuppgifterna innebär att konstruera funktioner. Ofta är det bra att bryta ned programmeringsuppgifter i färre små steg och testa att det fungerar i varje steg.

1. Lös uppgiften med vanlig kod direkt i R-Studio (precis som i datorlaborationen ovan) utan att skapa en funktion.
2. Testa att du får samma resultat som testexemplen.
3. Implementera koden du skrivit i 1. ovan som en funktion.
4. Testa att du får samma resultat som i testexemplen, nu med funktionen.

Automatisk återkoppling med `markmyassignment`

Som ett komplement för att snabbt kunna få återkoppling på de olika arbetsuppgifterna finns paketet `markmyassignment`. Med detta är det möjligt att direkt få återkoppling på uppgifterna i laborationen, oavsett dator. Dock krävs internetanslutning.

Information om hur du installerar och använder `markmyassignment` för att få direkt återkoppling på dina laborationer finns att tillgå **här**.

Samma information finns också i R och går att läsa genom att först installera `markmyassignment`.

```
install.packages("markmyassignment")
```

Om du ska installera ett paket i PC-pularna så behöver du ange följande:

```
install.packages("markmyassignment",lib="sökväg till en mapp i din hemkatalog")
```

Tänk på att i sökvägar till mappar/filer i R i Windowssystem så används `"\\"`, tex `"C:\\Users\\Josef"`.

Därefter går det att läsa information om hur du använder `markmyassignment` med följande kommando i R:

```
vignette("markmyassignment")
```

Det går även att komma åt vignetten **här**. Till sist går det att komma åt hjälpfilerna och dokumentationen i `markmyassignment` på följande sätt:

```
help(package="markmyassignment")
```

Lycka till!

Kapitel 3

Inlämningsuppgifter

För att använda `markmyassignment` i denna laboration ange:

```
library(markmyassignment)
lab_path <-
"https://raw.githubusercontent.com/STIMALiU/KursRprgm2/main/Labs/Tests/d8.yml"
suppressWarnings(set_assignment(lab_path))

Assignment set:
D8: Statistisk programmering med R: Lab 8
The assignment contain the following task:
- wordcount
```

Dokumentation och kodstil

Från och med denna laboration och de resterade laborationerna i kursen så ska ni förutom att lösa angivna uppgifter också **kommentera** era funktioner och ha en **god kodstil** för att bli godkända.

- Kodstil:
  - Det viktiga är att koden ska vara **tydlig** och **läsbar**.
  - Följ någon av kodstilarna i kapitlel "Kodstil" i datorlaboration 4. Ni måste inte följa dessa exakt, men koden ska se bra ut och var konsekventa i den stil som ni väljer att använda.
  - Tänk särskilt på:
    - Enhetlighet och struktur
    - Ha lämplig indentering och avstånd
    - Ha bra variabelnamn
- Kommentarer:
  - Funktionshuvud: Använd mallen för `roxygen2` som ges i kapitlel "Dokumentation av funktioner - `roxygen2`" i datorlaboration 4. Ni ska ha med:
    - `@title` Här skriver ni funktionens namn
    - `@description` Förklara kort vad funktionen gör
    - `@param` Skriv först arguments namn, sen mellanslag och sen förklara kort argumentet. Upprepa detta för alla argument i funktionen. Ex: `x Numerisk vektor, används vid beräkning av medelvärde.`
    - `@return` Förklara vad funktionen returnerar
  - Kommenter i funktionen: Era lösningar ska innehålla lämpliga kommenater, där ni förklarar de övergripande dragen och de viktiga stegen i er kod. Ni behöver inte föklara alla detaljer. Kommentarerna ska berätta sådant för programmeraren som inte står i koden. Använd luft och kommentarer för att gruppera och strukturera er kod.

- Tips: Tänk att det ska vara lätt att förstå er funktion långt senare, tex om ett år. Vilka kommentarer behövs då?

För att bli godkänd på inlämningsuppgifterna måste ni följa ovanstående instruktioner för kommentarer och kodstil.

3.1 `wordcount()`

Nu är uppgiften att skapa en funktion som ska kunna räkna hur många gånger olika ord förekommer i texten. Funktionen ska heta `wordcount()` och ha argumentet `text` som ska vara en `character`-vektor. Funktionen ska ta en text (i form av en text vektor) och returnera en `data.frame` med två variabler `word` (textvariabel) och `freq` (integervariabel).

I variabeln `word` ska respektive ord ingå, men med små bokstäver, och i variabeln `freq` ska frekvensen av orden framgå. Den `data.frame` som returneras ska vara sorterad efter variabeln `word`. Funktionen ska också skriva ut meningen "`The most common word is '[ord]' and it occurred [antal] times.`" med `message()`.

Tips! `table()`

Nedan är ett förslag på hur ni kan implementera funktionen.

1. Läs in paktet i `stringr` i den aktuella R-sessionen. OBS: ej installera paktet.
2. Börja med att sätta ihop de olika textelementen till en textsträng, men denna gång använd mellanslag som avskiljare istället för `\n`.
3. Ta bort punkter och kommatecken i textsträngen.
4. Gör om alla ord till endast gemener.
5. Dela upp teckensträngen med `str_split()` för att få ut respektive ord. [**Tips!** Tänk på att du får ut en lista med denna funktion, inte en vektor. `unlist()` kan då vara till hjälp.]
6. Räkna respektive ord och skapa en data.frame med respektive ord i kolumn 1 och antalet förekomster av detta ord i kolumn 2. Döp kolumn 1 till "`word`", och kolumn 2 till "`freq`".
7. Sortera datasetet efter `word`.
8. Använd `str_c()` och `message()` för att baserat på datsetet ovan skriva ut följande mening "`The most common word is '[ord]' and it occurred [antal] times.`".
9. Returnera din `data.frame`.

Funktionen `word()` är inte tillåten på denna uppgift. Kolla om testfallen nedan fungerar:

```
# Laddar ned testdata
library(downloader)
transtrommer_remote <-
  "https://raw.githubusercontent.com/STIMALiU/KursRprgm2/master/Labs/DataFiles/transtrom.txt"
transtrommer_local <- paste0(getwd(), "/transtrom.txt")
download(url = transtrommer_remote, destfile = transtrommer_local)

# Test
text<-readLines("transtrom.txt",encoding = "UTF-8")
worddata<-wordcount(text=text)

The most common word is 'the' and it occurred 8 times.

head(worddata)

        word freq
1          a    6
2        and    4
3 approached    1
```

```
4         as    1
5     before    1
6      black    1
```

```
head(worddata[order(worddata[,2], decreasing=TRUE),])
```

```
   word freq
59  the    8
1     a    6
2   and    4
24 have    3
40   of    3
62 they    3
```

```
head(wordcount(text=rep("a",10)))
```

The most common word is 'a' and it occurred 10 times.

```
  word freq
1    a   10
```

```
set.seed(39)
random_text<-sample(month.name,size = 60,replace = TRUE)
A<-wordcount(text=random_text)
```

The most common word is 'august' and it occurred 10 times.

```
head(A)
```

```
      word freq
1    april    1
2   august   10
3 december    5
4 february    7
5  january    3
6     july    2
```

```
str(A)
```

```
'data.frame': 12 obs. of  2 variables:
 $ word: chr  "april" "august" "december" "february" ...
 $ freq: int  1 10 5 7 3 2 3 8 7 4 ...
```

Nu är ni klara!