Inlämning 4

Johan Alenlöv

April 22, 2024

Inlämning

Utgå från laborationsmallen, som går att ladda ned här, när du gör inlämningsuppgifterna. Spara denna som labb[no]_[liuID].R , t.ex. labb1_josad732.R om det är laboration 1. 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: `labb1_josad732.R` blir då `labb1_josad732.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="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!

Chapter 1

Inlämningsuppgifter

Ladda nu ner laborationsmallen, som finns **här** och spara namn och liuID i respektive variabel. Spara filen som `inl2_[ditt liuID].R` Här är ett exempel `inl2_joswi123.R` .

För att använda `markmyassignment` i denna laboration ange:

```
library(markmyassignment)
# eller
library(markmyassignment,lib="en mapp i din hemkatalog")
lab_path <-
"https://raw.githubusercontent.com/STIMALiU/KursRprgm2/main/Labs/Tests/inl4.yml"
set_assignment(lab_path)

Assignment set:
Inl4: Statistisk programmering med R: Inlämning 4
The assignment contain the following task:
- wordcount
```

1.1 Lösa inlämningsuppgifter

Här kommer lite tips till när ni ska lösa inlämningsuppgifter.

- Inlämningsuppgifter utgår ifrån att ni har gjort **övningsuppgifterna** ovan först. Tänk att ni ska göra minst 70 % av övningsuppgifterna innan ni börjar med inlämningsuppgifterna.
- Se först till att ni förstår ”problemet” i uppgiften. Vad ska funktionen göra? Beskriv problemet för er själva. Vissa problem är det bra att bryta ner i mindre delproblem.
- Börja med att lösa problemet (eller det första delproblemet) med vanlig kod (dvs inte i en funktion) i ett R-script.
  - Lös ett delproblem i taget vid behov. Sätt sedan samman lösningarna på delproblemen. Under sök om koden fungerar som den ska.
- Sätt sedan in er kod i en funktion. Testa om funktionen kan återskapa de exempel som visas under beskrivningarna (testfallen). Se till att funktionen har rätt namn och rätt namn på argumenten.
- Ta er funktion och lägg in den i kodmallen, dvs i ett nytt R-script. Skapa variablerna `Namn` och `LiuId` med rätt innehåll. Spara filen med rätt namn.
- Kommentera bort all kod i er fil som inte är de aktuella funktionerna, namn- och ID-variabler och ev. kommentarer.
  - Detta minskar risken för fel.
- Testa nu att använda markmyassignment för att rätta er funktion.

  - Använd markmyassignment bara när er funktionen är ”hyfsat klar”, annars kommer ni att få många fel i markmyassignment, och det är svårt att veta var man ska börja kolla.
  - Det är vikitgt att funktionen rätt namn och rätt namn på argumenten, annars kommer ni att få många fel i markmyassignment.
- Om ni inte klarar alla tester i markmyassignment: undersök vilka felmeddelanden som ni får. Ta hjälp av lärare/labbassitenter vid behov om det är svårt att tolka.
- Innan ni lämnar in:
  - Se till att ni har löst uppgiften på det sätt som den är beskriven i uppgiftstexten. Ibland måste en viss metod användas för att lösa uppgiften. Ibland är vissa funktioner inte tillåtna i en viss uppgift. Så läs noga i varje uppgift vad som gäller.
  - Se till att ni klarar alla tester i markmyassignment

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 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 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.

1.2 `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 ...
```

Grattis! Nu är du klar!