Inlämning 1

Josef Wilzén och Johan Alenlöv

2 april 2025

Inlämning

Utgå från laborationsmallen när du gör inlämningsuppgifterna. Det finns en mall för varje inlämningsuppgift, se nedan under Inlämningsuppgifter. Filen ni lämnar in ska **inte** innehålla något annat än de aktuella funktionerna, namn- och ID-variabler och ev. kommentarer.

För att lämna in ska ni gör följande:

- Lös uppgifterna enligt beskrivningarna i instruktionen.
- Se till att ni kan återskapa de testfall som visas i uppgiften.
- Se till att era funktioner klara alla tester i `markmyassignment`. Det rekommenderas att använda `markmyassignment` först när er funktion är hyfsat klar.
- På en av de schemalagda datorlaborationerna: kontakta en lärare/assistent och be om att få redovisa er inlämning.
- **Redovisning:**
  - Rensa den globala miljön (Environment i Rstudio) så att bara de variabler och funktioner som krävs för uppgiften finns kvar. Tips: Funktionen `rm()` tar bort objekt/variabler
  - Kör markmyassignment och visa att testerna går igenom.
  - Förklara kort hur ni har löst uppgifterna och svara på eventuella frågor från assistenten.
  - Om assistenten säger att ni är godkända, lämna då in er lösning på Inlämningar i kursrummet på Lisam.
- **Deadline** finns under Inlämningar i kursrummet på Lisam.

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 RStudio (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!

Kapitel 1

Inlämningsuppgifter

Ladda nu ner laborationsmallen, som finns **här** och spara namn och liuID i respektive variabel. Spara filen som `inl1_[ditt liuID].R` Här är ett exempel `inl1_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/d1.yml"
set_assignment(lab_path)

Assignment set:
D1: Statistisk programmering med R: Inlämning 1
The assignment contain the following (3) tasks:
- hi_name
- approx_e
- fast_stock_analysis
```

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
  - Redovisa era lösningar (se ovan).

1.2 `hi_name()`

Skapa en funktion `hi_name()` som tar argumentet namn. Funktionen ska skriva ut `"Hi [namn]! Keep coding!"` där [namn] ersätts med värdet på argumentet `namn`.

Obs! använd `cat()`, inte `return()` och tänk på att resultatet ska bli exakt detsamma som nedan för full poäng. [**Tips!** argumentet `sep` i `cat()`]

```
> hi_name(namn = "Johan")

Hi Johan! Keep coding!

> hi_name(namn = "Florence")

Hi Florence! Keep coding!
```

1.3 `approx_e()`

Talet e kan beskrivas som följande oändliga serie:

$$e = \sum_{n=0}^{\infty} \frac{1}{n!}$$

Där ! är fakultet. Denna serie gör att talet e kan approximeras godtyckligt bra på följande sätt, genom att välja ett värde på N:

$$e = \sum_{n=0}^{N} \frac{1}{n!}$$

Exempel, låt $N = 3$

$$e = \sum_{n=0}^{3} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!}$$

Skapa en funktion i R som du kallar `approx_e()` med argumentet `N` för att skapa en godtyckligt nogrann approximation av e. Pröva hur stort n behöver vara för att funktionen ska approximera e till och med fjärde decimalen[1].

[**Tips!** för att få ut e med ett antal decimaler i R, använd `exp(1)`]

[1] Detta ska inte redovisas i inlämningen. Det viktiga är att ni klarar testerna i markmyassignment.

```
> approx_e(N = 2)

[1] 2.5

> approx_e(N = 3)

[1] 2.6667

> approx_e(N = 4)

[1] 2.7083

> # jämför med exp(1)
```

1.4 `fast_stock_analysis()`

Vi ska nu skapa en funktion för att göra en snabb analys av ett dataset som finns sparat som `.csv`. Detta är ett exempel på användningsområde för funktioner i R. Mycket data kommer in löpande och då kan det vara så att vissa standardanalyser vill vi göra snabbt med en förprogrammerad funktion. Vi ska skapa en funktion `fast_stock_analysis()` med argumentet `file_path` och `period_length`. Syftet är att snabbt analysera de senaste dagarnas aktiekurs. Vi utgår att data har följande struktur:

- Variabler (i given ordning): `Date`, `Open`, `High`, `Low`, `Close`, `Volume`, `Adj.Close`
- Rader är dagar med observerade data. Raderna är sorterade så att de nyligaste dagarna är högst upp i filen.

Funktionen ska läsa in en `.csv` - fil som anges av argumentet `file_path` och returnera en lista med de namngivna listelementen `total_spridning`, `medel_slutpris`, `slutpris_upp` och `datum`.

Så här ska funktionen implementeras:

1. Skriv först en funktion som ska läsa in data
2. Läs in data (csv) med argumentet `file_path` (innehåller både filnamn och sökväg). Använd funktionen `read.csv()`. [**Tips!** tänk på att `stringsAsFactors` kontrollerar om datumvariablerna blir en faktorvariabel eller ej.].
   Obs! `file_path` ska bara innehålla sökvägen till filen, filen ska läsas in inne i funktionen.
3. Plocka ut de senaste `period_length` antal dagar från datasetet (anta att de senaste aktiekurserna är högst upp).
4. Räkna ut de värden som ska returneras av funktionen:
   (a) `total_spridning` (ett numeriskt element) är skillnaden mellan det högsta värdet på `High` och det lägsta värdet på `Low` under perioden.
   (b) `medel_slutpris` (ett numeriskt element) är det genomsnittliga slutpriset under perioden.
   (c) `slutpris_upp` (ett logiskt element) är ett logiskt värde som anger `TRUE` om slutpriset första dagen under perioden är lägre än slutpriset den sista dagen under perioden.
   (d) `datum` (vektor med två textelement) ska innehålla det första och det sista datumet för perioden. **OBS!** Detta ska vara en textvektor, inte en faktor.
5. Sätt ihop dessa värden till en namngiven lista med namnen ovan.

Ladda ner testdata `AppleTest.csv` och `google2.csv`: här. Klicka på filnamnet och sen på knappen ”Raw” och spara ned filen. Notera vi testar funktionen på dessa filer, men funktionen ska fungera på godtyckliga dataset som har samma struktur som dessa två filer. Här är testexempel på hur funktionen ska fungera (**Obs!** Nedan visas sökvägar till filer som ligger öppet tillgängliga på github. Funktionen ska fungera även för lokalt nedladdade filer om rätt sökväg är angiven.):

```
apple_path <-
"https://raw.githubusercontent.com/STIMALiU/KursRprgm2/main/Labs/DataFiles/AppleTest.csv"
google_path <-
"https://raw.githubusercontent.com/STIMALiU/KursRprgm2/main/Labs/DataFiles/google2.csv"
```

```
myList1 <- fast_stock_analysis(file_path = apple_path, period_length=5)
str(myList1)

List of 4
 $ total_spridning: num 11.8
 $ medel_slutpris : num 425
 $ slutpris_upp   : logi FALSE
 $ datum          : chr [1:2] "2012-01-18" "2012-01-24"

myList2 <- fast_stock_analysis(file_path = apple_path, period_length=10)
str(myList2)

List of 4
 $ total_spridning: num 12.7
 $ medel_slutpris : num 424
 $ slutpris_upp   : logi FALSE
 $ datum          : chr [1:2] "2012-01-10" "2012-01-24"

myList3 <- fast_stock_analysis(file_path = apple_path, period_length=3)
str(myList3)

List of 4
 $ total_spridning: num 8.9
 $ medel_slutpris : num 423
 $ slutpris_upp   : logi TRUE
 $ datum          : chr [1:2] "2012-01-20" "2012-01-24"

myList4 <- fast_stock_analysis(file_path = google_path, period_length=20)
str(myList4)

List of 4
 $ total_spridning: num 92.2
 $ medel_slutpris : num 631
 $ slutpris_upp   : logi FALSE
 $ datum          : chr [1:2] "2011-12-23" "2012-01-24"
```

Tänk på att filen du lämnar in endast ska innehålla de obligatoriska variablerna och funktionerna i inlämningsuppgifterna och **inget** annat. Var noga med att ge filen korrekt namn innan du lämnar in den.

Grattis! Nu är du klar!