Datorlaboration 5

Josef Wilzén

11 februari 2014

Instruktioner

- Denna laboration ska göras i **grupper om två personer**. Det är viktigt att att följa gruppindelningen och inte ändra grupper. Om det är problem med grupperna så ska ni prata med Josef eller Måns.
- Det är tillåtet att diskutera med andra grupper, men att plagiera eller skriva kod åt varandra är **inte tillåtet**.
- Utgå från laborationsfilen som går att ladda ned **här**
- Laborationen består av två delar:
  - Datorlaborationen
  - Inlämningsuppgifter
- Innan du lämnar in laborationen:
  1. Starta om R-Studio eller rensa den globala miljön (Global environment) med `rm(list = ls())`.
  2. Ladda in funktionerna i R med `source`.
  3. Kontrollera att inget annat än funktionerna laddas in.
  4. Testa att funktionerna fungerar en sista gång.
- Deadline för labben framgår på **kurshemsidan**

Innehåll

Parprogrammering

Tanken är att ni ska öva på parprogrammering under labb4 till labb8.

- Detta innebär att två personer samarbetar och tillsammans löser programmeringsproblem vid en dator.
- Personerna turars om att ha rollerna:
  - **Föraren:** har kontroll över tangentbordet och skriver koden
  - **Navigatören:** Är delaktig i problemt genom att kommentera, diskutera och analysera koden som skrivs. Den här personen ska **inte** sitta och titta passivit.
- Det är viktigt att byta roller **ofta** och att båda personerna är involverade i lösningen av problemet. Vi rekomenderar att ni byter roller minst varje 30 min.
- Det är viktigt att kommentera sin kod så att ni båda kan förstå koden i efterhand. Tänk på skriva ner syftet med koden och inte exakt vad koden gör.
- Syftet med parprogrammering är:
  - Lära av varandra
  - Lära sig olika sätt att skriva kod (stilar) och olika sätt att lösa problem
  - Skriva mer effektiv kod med mindre buggar
  - Lära sig skriva kod som är lätt att förstå för andra

Del I
Datorlaboration

1 Grafik och beskrivande statistik

1. Kör koden nedan och notera hur plottarna ändras. Efter att ni kört sista raden, kolla in på `?par`, och leta er fram till rubrikerna "`lty`" och "`lwd`". Testa att ändra värdet på dessa och notera hur plotten ändras. Kolla på `?plot`, vilka olika val finns det för parametern "`type=`" ?

```
a <- seq(1, 3, length = 20)
b <- exp(a)
plot(a, b)
plot(x = a, y = b, col = "blue")
plot(a, b, col = 4, main = "Hej!")
plot(a, b, col = 4, type = "o")
plot(a, b, type = "l", col = "blue", lwd = 3, lty = "dashed")
```

2. Spara den sista plotten som pdf-fil med koden nedan. Kolla att plotten ligger i mappen som är ditt working directory.

```
dev.copy2pdf(file = "Min_plot")
```

3. Mer plottar! Vi har en numerisk vektor `x`, som är indelad i tre grupper enligt `minaGrupper`. Kör koden nedan och undersök plottarna. I sista raden, vad innebär `ifelse(minaGrupper=="b",1,2)` ? tips: `?ifelse()`

```
x <- c(1:12, 34:50, 42:36)
minaGrupper <- c(rep("a", 12), rep("b", 17), rep("c", 7))
plot(x)
barplot(x)
boxplot(x ~ minaGrupper)
barplot(x, col = c(1, 2))
barplot(x, col = ifelse(minaGrupper == "b", 1, 2))
```

4. För att kontrollera vilka värden som ska vara med på x- och y-axeln i en plot används argumenten `xlim=` och `ylim=`. Kör koden nedan. Ändra värdena på `xlim=` och `ylim=` och se vad som händer med plotten. Testa `?abline()`, vad händer om du ändrar värdet i "`v=`" och "`h=`" till numeriska

vektorer?

```
plot(x = a, y = b, col = "blue", xlim = c(-3, 5))
plot(x = a, y = b, col = "red", xlim = c(-3, 5), ylim = c(-5, 20))
abline(v = 0, h = 0)  # lägga till referenslinjer till plotten
```

5. Vi fortsätter med med `x` och `minaGrupper` från (3). Nu vi vi veta lite beskrivande statistik över `x`, grupperat på `minaGrupper`. Kör koden nedan:. Vad gör raderna med `by()`? Testa att byta ut `by()` mot `aggregate()`. Vad blir skillnaden på resultatet?

```
mean(x)
sd(x)
# Nu grupperat:
by(x, minaGrupper, mean)
minaSt <- by(x, minaGrupper, sd)
minaSt
class(minaSt)
str(minaSt)
```

6. Ladda in dataseten "AppleLabb5.csv" och "GoogleLabb5.csv" och spara dem som data.frames med namnen `Apple` och `Google` respektive. Dessa dataset kommer att användas under resten av labben.

7. Välj kolumen Close från båda dataseten och spara dem som `appleClose` och `googleClose`. Dessa representerar stängningspriset för Apple- och Googleaktien respektive. Skapa en ny variabel `time`, enligt nedan. time indikerar nu tiden för aktiepriset för de två företagen. (Notera att det senaste aktiepriset kommer först i data)

```
appleClose <- Apple[, "Close"]
googleClose <- Google[, "Close"]
time <- length(appleClose):1
```

8. Gör en scatterplot av time mot appleClose. Ha "Time" som rubrik på x-axeln och "Price" som rubrik på y-axeln. Huvudrubrik ska vara "Stock prices".

9. Gör om (8), men med en linjeplot. Sätt färgen till blå och ändra gränsen för y-axeln till: `ylim = c(0, max(appleClose, googleClose))`

10. Lägg till en linje för googleClose i plotten i uppgift 9. Låt den ha röd färg. Tips: `points()` eller `lines()`

11. Lägg till en bildtext (eng: legend) till figuren (kolla på ?legend()) enligt nedan. Testa att lägga "legend" på någon annan plats i plotten.

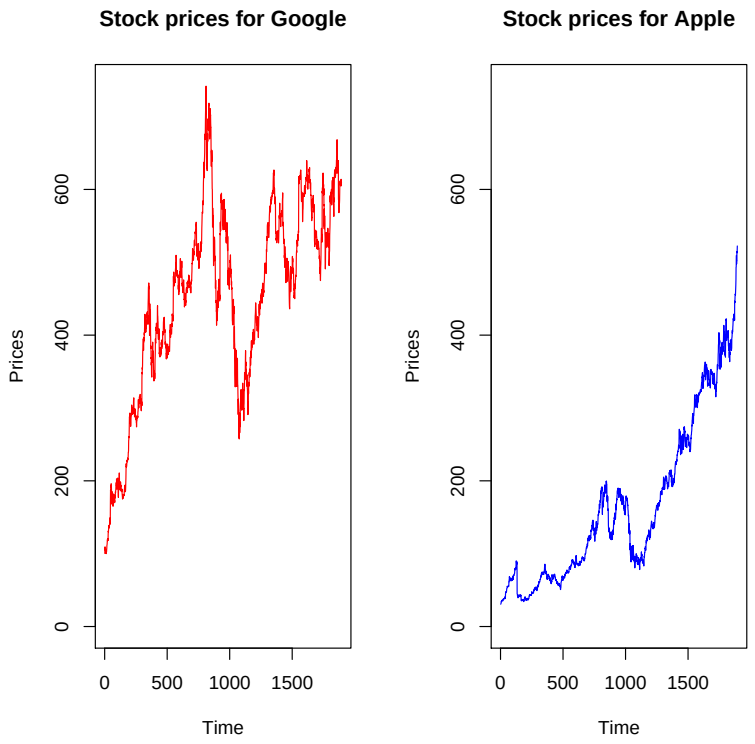
```
legend("topleft", lty = c("solid", "solid"), col = c("red", "blue"), legend = c("Google
    "Apple"))
```

12. Nu ska appleClose och googleClose plottas i en "1-by-2-subplot", Det innebär att två plottar kommer ligga brevid varandra i samma plot men som olika del plottar. Plotten ska se ut som figuren nedan. Observaera att y-axeln ska ha samma värden för både Apple och Google.

    - Spara uppgift 7 på följande sätt:

```
twoPlots <- function() {
    # Här placerar ni koden
}
# Testa genom att köra:
twoPlots()
```

    - Tips: Funktionen `par()` kan användas för att styra plottar på olika sätt. Vad händer om ni ändrar värdet på `mfrow=` ? Vad gör argumentet `mfg=`?



13. Testa att göra histogram över `appleClose` och `googleClose`. Tips `?hist()`

    (a) Ändra antalet breaks i histogrammen. Hur ser det ut om du har väldigt många eller vädigt några?

    (b) Ändra så att histogrammen visar relativa frekvenser.

    (c) Ändra nu rubriken och axeltexterna. Ändra färgen på histogrammet.

    (d) Spara histogrammet i (c) som png-bild. Använd koden nedan. Testa `?png()`, hur många olika bildformat hittar du? Hur gör du om du vill lägga bilden på något annat ställe än i ditt workning directory?

```
png(filename = "minBild1.png")
# kod för att plotta
dev.off()
```

14. Skatta en täthetskurva för `appleClose` och `googleClose`. Tips ?density().

    (a) Plotta täthetskurvorna i två olika plottar. Frivillig extra fråga: Hur ska ett sådant diagram tolkas?

(b) Plotta histogram med relativa frekenser för för `appleClose` och `googleClose`. Lägg sedan till täthetskurvorna (som linjer) i i histogammen och ge dessa färgen blå. Öka tjockleken något på linjerna.

(c) Plotta nu båda täthetskurvorna i samma plot. Se till att x-axeln är tillräckligt lång för båda kurvorna.

15. Nu har ni sett en del olika plottar i R. Titta nu på exemplen nedan för att få mer exempel på hur grafik kan se ut i R. Tryck på enter i konsolen för att hoppa till nästa plot.

```
example(plot)
example(points)
example(hist)
example(barplot)
example(pie)
example(image)
```

16. Nu ska beskrivande statistik tas fram för dataseten "`Google`" och "`Apple`". Gör följande beräkningar:

(a) Hitta det hösta och minsta värdet i dataseten (för alla variabler). Ta reda på vilken rad dessa värden finns på.

(b) Beräkna medelvärden för alla kolumnerna med `colMeans()`.

(c) Beräkna medelvärden för alla kolumnerna med `apply()`.

(d) Gör nu om uppgiften ovan för "`Apple`" men men med `lapply()` och `sapply()`. Vad blir skillnaden i de tre fallen? Vad är skillnaden mellan funktionerna generellt?

(e) Beräkna nu kvartiler för vektorn `appleClose`, se uppgift 7. Hur tolkas detta resultat? Hur gör du om du vill beräkna den 1 och 99 procentiga percentilen? Tips `?quantile()`

(f) Beräkna nu kvartiler för alla variabler i båda dataseten.

(g) Testa nu att använda funktionen `summary()` på `appleClose` först och sen på båda dataseten. Vad får du för resultat?

17. Skriv en funktion som hittar den variabel som har störst standardavvikelse i ett dataset. Funktionen ska ha en numerisk data.frame som argument. Funktion ska skriva ut variabelnammnet och den aktuella standardavvikelsen till konsolen som en textsträng. Testa din funktion med "Apple" och "Google". Tips `?sd()`, `?names()`, `?paste()`

18. Beräkna kovariansen och korrelation mellan variablerna `Open` och `Close` i "Apple".

19. Beräkna kovariansen mellan alla variabler i "Apple".

    (a) Använd först `apply()` tillsammans med `cov()`.

    (b) Testa nu att anropa `cov()` med hela "Apple" som argument. Om du gör rätt ska du få en kovariansmatris, vilket är en matris som innehåller de skattade kovarianserna för de olika variablerna. Hur ska du tolka kovariansmatrisen? Vad är det för värden på diagonalen?

    (c) Gör nu samma sak som i (b), men beräkna korrelationen och ta fram korrelationsmatrisen.

20. Skapa en frekvenstabell med med över vektorn `minaGrupper` från uppgift 3. Spara tabellen som `minaFrekvenser`. Tips `?table()`

21. Skapa en ny kategorisk variabel med minst tre olika kategorier och som är lika lång som `minaGrupper` (36 element), kalla den `nyaGrupper`. Gör en två-vägs frekvenstabell över `minaGrupper` och `nyaGrupper`. Spara tabellen som `twoWayFreq`. Tips: `?gl()`

2 Generiska funktioner, klasser och objekt

1. Testa att använda funktionerna nedan på några olika objekt från del 1 av labben. Vad får du för resultat?

    (a) `is.vector()`

    (b) `is.matrix()`

    (c) `is.list()`

    (d) `is.data.frame()`

2. Testa nu om du kan omvandla några av objekten från del 1 med funktionerna nedan:

    (a) `as.vector()`

    (b) `as.matrix()`

    (c) `as.list()`

    (d) `as.data.frame()`

    (e) `unlist()`

3. Skapa objektet `testObj` enligt koden nedan. Undersök `testObj` med funktionerna `class()` och `attributes()`, Hur tolkar du resultatet från `attributes()`?

```r
x <- seq(4, 19, length = 10)
y <- seq(12, -20, length = 10)
z <- seq(1, 95, length = 10)
testObj <- cbind(x, y, z)
colnames(testObj) <- c("x", "y", "z")
rownames(testObj) <- paste("Rad ", 1:10)
```

4. Återvänd till tabellen `twoWayFreq` från uppgift 21 i första sektionen. Testa att använda funktionerna `str()`, `class()`, `dim()` och `attributes()` på `twoWayFreq`. Testa även `colnames()` och `rownames()`. Vad får du för resultat?

5. `plot()` är en generisk funktion i R. Detta innebär att den gör olika saker med olika typer av objekt.

   (a) Testa att köra koden nedan. Vad får du för resultat i de olika fallen?

   ```r
   plot(x)
   plot(testObj)
   myData <- as.data.frame(testObj)
   plot(myData)
   ```

   (b) Testa att använda `class()` på `x`, `testObj` och `myData`. Testa att köra `methods(plot)`.

6. En annan generisk funktion är `summary()`. Den kan ge mycket olika resultat beroende på objekt. Testa koden nedan. Vad händer i de olika fallen? Testa att köra `class()` på de olika objekten.

```r
myGraph <- hist(appleClose)
summary(testObj)
summary(del1)
summary(twoWayFreq)
summary(myGraph)
# testa även:
str(myGraph)
```

7. I fallet `summary(twoWayFreq)` så används `summary()` på ett objekt av klassen table. Ta reda på vilka olika varianter av summary det finns med hjälp av methods(). Vad heter funktionen som används om ett objekt är av klassen table? Hitta hjälpavsnitt som ger förklaringen till resultatet av `summary(twoWayFreq)`. Tips: Det står under avsnittet "Value".

8. Testa att köra `as.data.frame()` på ett objekt av klassen table. Vad händer?

Del II
Inlämningsuppgifter

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.

Ett förslag på hur du kan angripa problemet är att:

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.

Varje uppgift kan ge flera poäng så även om du inte lyckas med alla delar i en uppgift kan du få poäng.

3 Beskrivande statistik (5 p)

Ni ska nu skapa en funktion som tar fram beskrivande statistik för ett dataset. Funktionen ska heta `myStatsFunc` och ta en data.frame som argument och returnera beskrivande statistik som sparas i en lista. Vilken statistik metod som används ska bero på det är en kontinuerlig eller kategorisk variabel som analyseras. Följ kodmallen för labben.

1. Skapa hjälpfunktionen nedan:

   (a) `findNumeric(myData)`: Funktionen ska hitta kolumnindex för de variabler som som är numeriska i data.frame `myData`, och sen returnera dem som en numerisk vektor.

   (b) `findCharacter(myData)`: Funktionen ska hitta kolumnindex för de variabler som som är character/factor i data.frame `myData`, och sen returnera dem som en numerisk vektor.

   Tips för a) och b): Kolla in `?sapply()`, vad gör argumentet "`simplify=`"? Tänk på att en data.frame är en sorts lista med variablerna som element i listan. Det går också att använda en for-loop. Kolla in `is.numeric()`/`is.character()` och `which()`

2. Skapa nu funktion `orderMyData()`, som ska ta en data.frame som argument. Funktionen ska skapa en lista med två element. I det första elementet

i listan ska den delmängd av variablerna som är numeriska sparas som en data.frame. I det andra elementet i listan ska den delmängd av variablerna som är character/factor sparas som en data.frame. Funktionen ska sedan returnera listan. Använd funktionerna från i 1 för att ta reda på vilka variabler som är av vilken typ. Tips: Använd drop=FALSE när ni väljer ut variabler, det underlättar senare i uppgiften. Nedan kommer ett testfall för `orderMyData()`.

```
x1 <- c(1, 3, 56, 3)
x2 <- c("a", "b", "a", "c")
x3 <- rep(TRUE, 4)
x4 <- 2:5
x5 <- rep(FALSE, 4)
x6 <- c("a", "a", "c", "b")
test1 <- data.frame(x1, x2, x3, x4, x5, x6)
# testa orderMyData():
orderMyData(test1)

[[1]]
  x1 x4
1  1  2
2  3  3
3 56  4
4  3  5

[[2]]
  x2 x6
1  a  a
2  b  a
3  a  c
4  c  b
```

3. Skapa nu funktionen `myStatsFunc()`, följande argument ska ingå:

   (a) `myDataSet`: en data.frame med data som ska analyseras

   (b) `numericFunc`: namnet på den funktion som ska anaysera numeriska variabler

   (c) `categoricFunc`: namnet på den funktion som ska analysera kategoriska variabler

4. Låt nu myStatsFunc() arbeta enligt nedan:

   (a) Anropa `orderMyData()` för att få ut en lista som är grupperad på variabeltyp.

(b) För numeriska variabler: använd funktionen `numericFunc` på varje variabel och spara resultatet för varje variabel i elementen i en lista (kalla den `numericList`). Döp de olika elementen i listan till samma namn som variablerna.

(c) För kategoriska variabler: använd funktionen `categoricFunc` på varje variabel och spara resultatet för varje variabel i elementen i en lista (kalla den `categoricList`). Döp de olika elementen i listan till samma namn som variablerna.

Tips för (b) och (c): Listan ska vara lika lång som antalet variabler. Observera att beroende på vilken sorts funktion `numericFunc`/`categoricFunc` är så kan olika typer av objekt sparas i listan. Tips: `lapply()` eller en for-loop. Exempel: Om vi har tre numeriska variabler i data och "`numericFunc=mean`" så ska `numericList` vara 3 element lång och i varje element ska medelvärdet för de olika variablerna sparas.

(d) Skapa en ny lista och kalla den `outputList`. Iaktag nu följande fall:

i. Om det finns **både** numeriska variabler och katgoriska variabler i `myDataSet`: Spara `numericList` i det första elementet och och `categoricList` i det andra elementet i outputList. Det sista som funktionen gör är att returnera `outputList`.

ii. Om det bara finns **numeriska** variabler i `myDataSet`: Spara `numericList` i det första elementet i `outputList`. Det sista som funktionen gör är att returnera `outputList`.

iii. Om det bara **katgoriska** variabler i `myDataSet`: Spara `categoricList` i det första elementet i `outputList`. Det sista som funktionen gör är att returnera `outputList`.

iv. Om det varken finns numeriska variabler och katgoriska variabler i `myDataSet`: Returnera strängen "`Inga variabler`" som en vektor (ingen `outPutList`).

Notera att i fallen (i)-(iii) så blir `outPutList` en nästlad lista, dvs en lista som där elementen består av listor (eftersom `numericList` och `categoricList` är listor).

Testa nu om funktion klarar av testfallen nedan:

```r
# testfall 1:
x1 <- c(seq(1, 40, length = 30))
x2 <- rep(c("a", "b", "a"), 10)
x3 <- rep(TRUE, 30)
x4 <- -30:-1
x5 <- rep(c("a", "a", "c", "b", "b"))
test1 <- data.frame(x1, x2, x3, x4, x5)
myStatsFunc(myDataSet = test1, numericFunc = var, categoricFunc = sort)
```

```
$numericList
$numericList$x1
[1] 140.2

$numericList$x4
[1] 77.5


$categoricList
$categoricList$x2
 [1] a a a a a a a a a a a a a a a a a a a a a a b b b b b b b b b b
Levels: a b

$categoricList$x5
 [1] a a a a a a a a a a a a a b b b b b b b b b b b b c c c c c c
Levels: a b c
```

```
# testfall 2:
myStatsFunc(myDataSet = data.frame(a = TRUE), numericFunc = mean, categoricFunc = table)
```

```
[1] "Inga variabler"
```

```
# testfall 3:
data(esoph)
names(esoph)
```

```
[1] "agegp"     "alcgp"     "tobgp"     "ncases"    "ncontrols"
```

```
myStatsFunc(myDataSet = esoph, numericFunc = median, categoricFunc = table)
```

```
$numericList
$numericList$ncases
[1] 1

$numericList$ncontrols
[1] 6


$categoricList
$categoricList$agegp

25-34 35-44 45-54 55-64 65-74   75+
   15    15    16    16    15    11

$categoricList$alcgp

0-39g/day     40-79    80-119      120+
```

```
       23        23        21        21

$categoricList$tobgp

0-9g/day    10-19    20-29      30+
      24       24       20       20

# testfall 4:
data(iris)
names(iris)

[1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"
[5] "Species"

myStatsFunc(myDataSet = iris, numericFunc = quantile, categoricFunc = table)

$numericList
$numericList$Sepal.Length
  0%  25%  50%  75% 100%
 4.3  5.1  5.8  6.4  7.9

$numericList$Sepal.Width
  0%  25%  50%  75% 100%
 2.0  2.8  3.0  3.3  4.4

$numericList$Petal.Length
  0%  25%  50%  75% 100%
1.00 1.60 4.35 5.10 6.90

$numericList$Petal.Width
  0%  25%  50%  75% 100%
 0.1  0.3  1.3  1.8  2.5


$categoricList
$categoricList$Species

    setosa versicolor  virginica
        50         50         50
```

4 Kontrollera en plot (5 p)

I denna uppgift ska ni skriva en funktion som kan kontrollera utseendet av av en plot som innehåller subplots. Se till att dataseten "Apple" och "Google" är inlästa i R. OBS: använd inte data från labb 2! Skapa `stockData` enligt nedan.

`stockData` ska användas för att testa vår funktion.

```
stockData <- cbind(Apple, Google)
appleNames <- paste("Apple", names(Apple), sep = "")
googleNames <- paste("Google", names(Google), sep = "")
names(stockData) <- c(appleNames, googleNames)
```

Funktionen ska heta `subPlotFunc()` och ska ha följande argument:

- `myData`: en data frame med numeriska variabler
- `rowLength`: ett tal, antal rader för subplots
- `colLength`: ett tal, antal kolumner för subplots
- `myOrder`: en vektor, ordning som variablerna ska plottas. Tex vektorn 1:4 plottar variabelerna i ordning 1,2,3,4. Siffrorna refererar till kolumnindex i `myData`.

`subPlotFunc()` ska utföra följande: givet värdena på `rowLength` och `colLength` så ska en lämplig ram för subplots sättas upp. Denna ram ska sedan fyllas radvis med histogram över några av variablerna i `myData`. Vilka variabler som ska användas och i vilken ordning ges av `myOrder`, som refererar till kolumn index. Om `myOrder=c(4,3,2)`, så ska variablerna 4, 3 och 2 användas i den givna ordningen. För att funktionen ska fungera så ska följande vara uppfyllt: `rowLength*colLength>=length(myOrder)`. Följ arbetsordning nedan:

1. Skapa stockData enligt koden ovan. Se till att stockData har 8 variabler och 1894 rader, och att variabelnamnen är korrekta. Ändra ej ordningen på variablerna.
2. Använd kodmallen för labben.
3. Testa först `rowLength*colLength>=length(myOrder)` att är uppfyllt. Om inte, så ska meddelandet **"`Argumenten funkar ej tillsammans`"** skrivas till konsolen och funktion avbryts. Om det är uppfyllt så ska funktion forstätta köra.
4. Resa bort eventuella plottar och sätt upp en lämplig subplot-ram. Tips `?frame(), ?par()`
5. Skapa en nested-for loop som kan loopa över både rader och kolumner i din ram för subplots. I det här fallet innebär det att du vill göra en for loop med en for loop inuti. Använd `par(mfg=)` för att styra vilken subplot som ska plottas. Koden nedan visar exempel på en nested-for loop. I exemplet så loopar den yttre index (i) över vektorn 1,2,3,4,5 och inre index (j) över vektorn 1,2,3. Detta gör det möjligt för oss att göra beräkningar med kombinationer av talen (1,2,3,4,5) och (1,2,3).

```
for (i in 1:5) {
    for (j in 1:3) {
        # kod som använder båda index: i och j
    }
}
```

6. För att välja ut vilken variabel som ska plottas behöves ett extra index skapas (förutom `i` och `j` i nested-for loop), låt oss kalla den `count`. Enklast är om `count` sätts till 0 utanför looparna, och räknas upp med ett steg i den innersta loopen. `count` kan nu användas för att välja ut element ur vektorn `myOrder`, som nu är det aktuella kolumnindex. Med hjälp av detta så väljs en kolumn ut ur `myData` som sen plottas med `hist()`.

7. Följande argument ska ändras i `hist()`: (OBS ändra inga andra än de nedanstående)

   (a) Ändra så att rubriktexten är på formen "Rad: 1 Kolumn: 2" om det är en subplot som ligger på första raden och andra kolumnen. Se testfallen nedan för mer exempel. Tips: `?paste()`

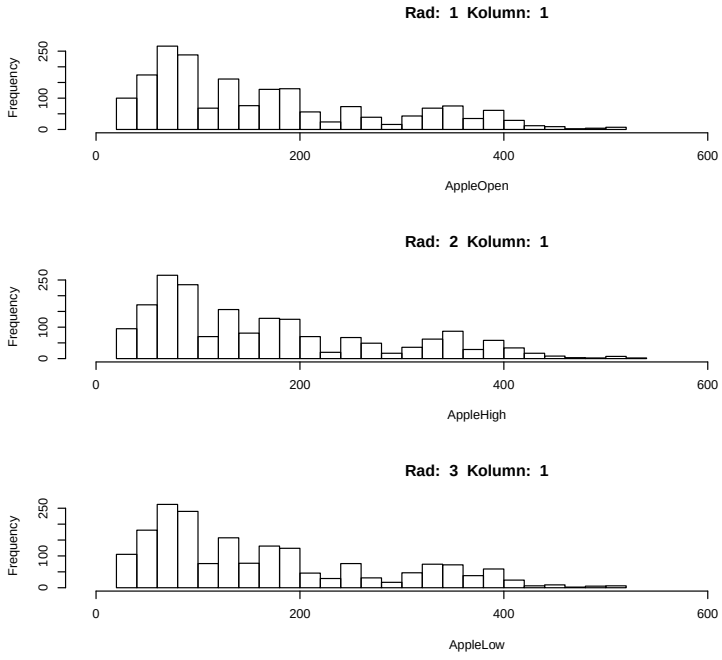
   (b) Ändra så att texten på x-axeln är namnet på den aktuella variabeln. Se testfallen nedan för exempel. Tips: `?names()`

   (c) `breaks=30`

   (d) Skalan på x-axeln ska gå från 0 till det största värdet i `myData`, så att alla plottar har samma skala.

Testa nu om din funktion fungerar med de fyra testfallen nedan. Lägg märke till i vilken ordning som variablerna plottas och rubrikerna.

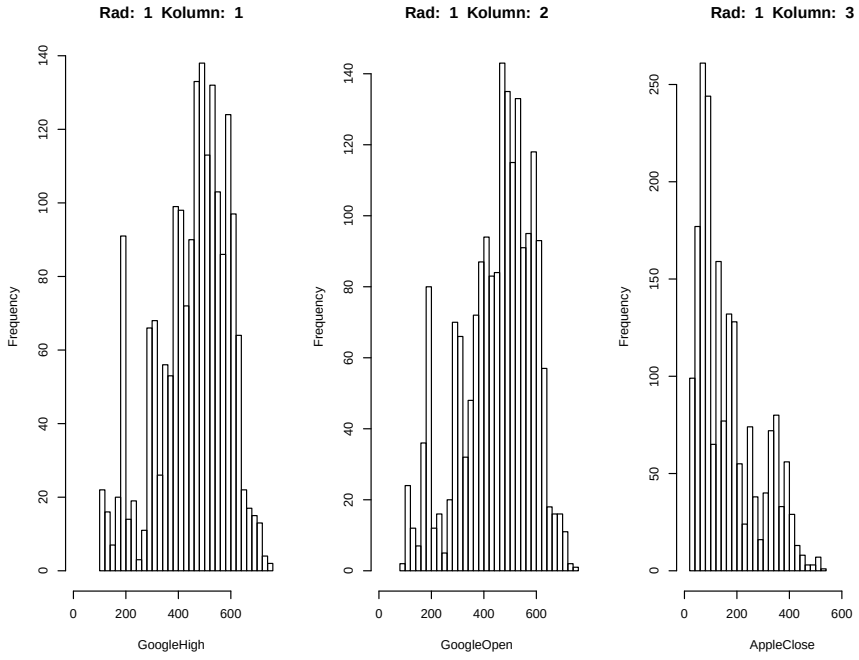
```
# testfall 1:
subPlotFunc(myData = stockData, rowLength = 3, colLength = 1, myOrder = 1:3)
```




```
# Förklaring: Vi vill plotta de tre första variablerna radvis
```

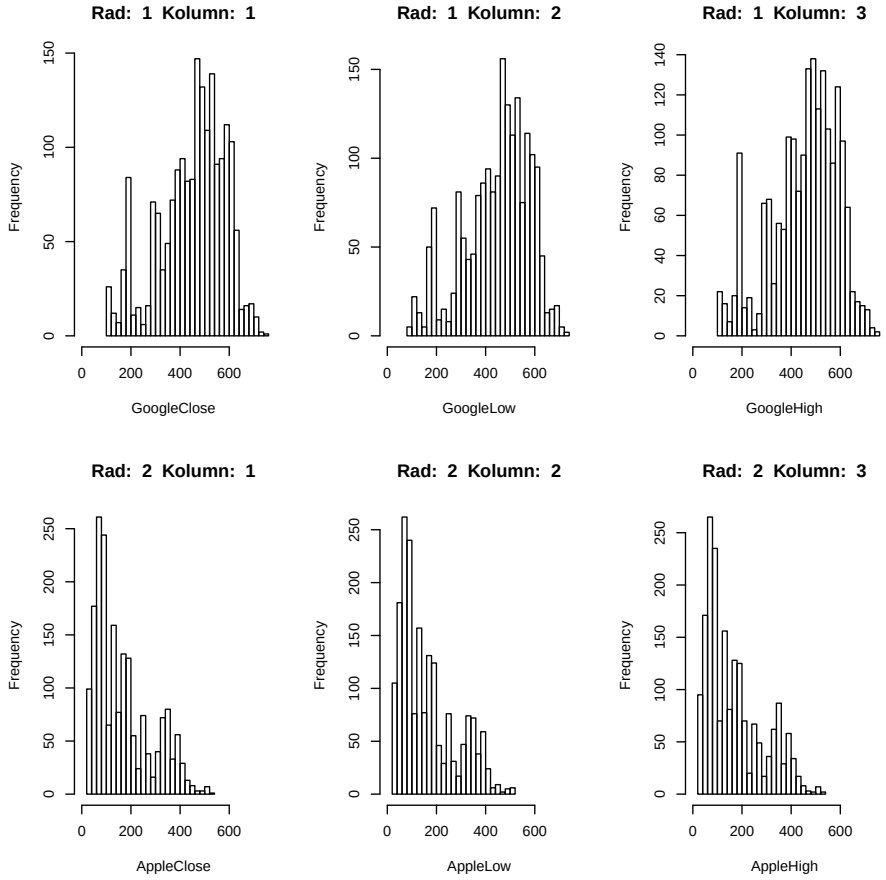
```
# testfall 2:
subPlotFunc(myData = stockData, rowLength = 3, colLength = 1, myOrder = 1:8)

[1] "Argumenten funkar ej tillsammans"

# Förklaring: Vi försöker välja fler variabler än som får plats i vår
# subplot-ram
```

```
# testfall 3:
subPlotFunc(myData = stockData, rowLength = 1, colLength = 3, myOrder = 6:4)
```




```
# Förklaring: Vi vill plotta variabel 6,5 och 4 kolumnvis (i den givna
# ordningen)
```

```
# testfall 4:
subPlotFunc(myData = stockData, rowLength = 2, colLength = 3, myOrder = c(8:6,
    4:2))
```




```
# Förklaring: Vi vill plotta variablerna Close, Low och High(i den givna
# ordningen), första raden för Google och andra raden för Apple
```

Nu är du klar!