LINKÖPINGS UNIVERSITET
Institutionen för datavetenskap
Avdelningen för statistik
Josef Wilzén & Måns Magnusson

2014-03-28
Programmering i R, 7.5 hp
732G33

Tentamen i Programmering i R, 7.5 hp

Skrivtid: 14-18
Hjälpmedel: Inget tryckt material, dock finns "R reference card v.2" av Matt Baggot tillgängligt elektroniskt.

Jourhavande lärare: Josef Wilzén
Besöker tentan kl 15.15 och kl 17.00

Betygsgränser: Tentamen omfattar totalt 20 poäng. 12 poäng ger Godkänt. 16 poäng ger Väl godkänt.
Skriv dina lösningar i fullständig och läsbar kod.
Lösningen skrivs i en körbar textfil med namnet Main.R
Se filen DocStudent.pdf för hur tentan ska lämnas in.
Kommentera direkt i Main.R filen när något behöver förklaras eller diskuteras.
Eventuella grafer som skapas under tentans gång behöver INTE skickas in för rättning, det räcker med att skicka in den kod som producerar figurerna.

OBS: Glöm inte att spara din fil ofta! Om R krashar kan kod förloras.

1. Datastrukturer (4p)

   (a) Skapa följande objekt: **1.5p**

      i. `myText`, som innehåller textsträngen **`"Hello World!"`**
      ii. `X`: en numerisk vektor innehållande alla heltal mellan 1 och 15.
      iii. `myBoolean`: en logisk vektor innehållande `c(TRUE,TRUE,FALSE,TRUE,FALSE)` upprepad 3 gånger.

   (b) Skapa en lista, kallad `myList`, där objekten i (a) ska vara element och har ordningen som ges ovan. Namnen på de olika elementen ska vara "`myText`","`X`" och "`myBoolean`" **1p**

   (c) Kopiera `myList` till `newList`. Gör sedan följande med `newList`:
   Ändra `X` på följande sätt: de element i `X` som motsvaras av `TRUE` i vektorn `myBoolean` ska ändras genom beräkningen $X/-100$
   Lägg till matrisen `Y` nedan som ett fjärde element i `newList`, namnet på elementet ska vara `matrixElement` **1.5p**

```
Y <- cbind(1:10, 10:1)
```

2. Kontrollstrukturer (3p)

   (a) Skriv en funktion, kallas `testType`, som använder if-else för att göra följande: Givet ett argumentet `X`, testa om `X` är av typen numeric, returnera då strängen "X is numeric". Om `X` är av typen character returnera då strängen "X is character". Om `X` är varken av typen numeric eller character, så returnera då strängen "Other". **1.5p**

```
testType <- function(X) {
    # skriv din kod här
}
testType(X = 1:10)
testType(X = "hej")
testType(X = TRUE)
```

   (b) Skapa matrisen `X` nedan. Skriv sedan en for-loop som loopar över varje kolumn i `X` och gör följande: **1.5p**

   - Beräknar medianen för den aktuella kolumnen och avrundar värdet till 3 decimaler.
   - Skriver ut texten **"I kolumn i är medianen value"** till skärmen, där `i` är aktuellt loop-index och `value` är den aktuella medianen.

```
set.seed(1234)
x1 <- rchisq(n = 9, df = 1)
x2 <- rchisq(n = 9, df = 5)
x3 <- rchisq(n = 9, df = 10)
x4 <- rchisq(n = 9, df = 15)
X <- cbind(x1, x2, x3, x4)
```

3. Strängar och datum (4.5p)

   (a) Läs in paketen `lubridate` och `stringr` i R. **0.5p**

   (b) Skapa en vektor som innehåller datum för alla dagar under 2014. Döp vektorn till `year2014`. Nedan visas de fem första elementen i vektorn. **1p**

```
year2014[1:5]

[1] "2014-01-01 UTC" "2014-01-02 UTC" "2014-01-03 UTC" "2014-01-04 UTC"
[5] "2014-01-05 UTC"
```

   (c) Skapa en ny vektor `weekends`, som innehåller alla datum för lördagar och söndagar under 2014. `weekends` kommer att innehålla 104 element. **1p**

   (d) Läs i filen "`ageData.txt`" i R så att varje rad blir ett element i en vektor, döp vektorn till `ageData`. Din uppgift är nu att utifrån vektorn `ageData` skapa en data.frame som ska heta `personAge`. `personAge` ska ha två variabler: `ID` som ska innehålla person ID (person 1, person 2 osv) och `age` som ska innehålla åldern för varje person. För att lösa uppgiften ska ni använda olika funktioner i R som kan hantera strängar. Det är **inte** en godkänd

lösning att manuellt skriva av aktuella värden från `ageData`. Se nedan för hur `personAge` ska se ut: **2p**

```
        ID age
1 Person 1  35
2 Person 2   3
3 Person 3  78
4 Person 4  56
5 Person 5  43
6 Person 6  28
7 Person 7 102
```

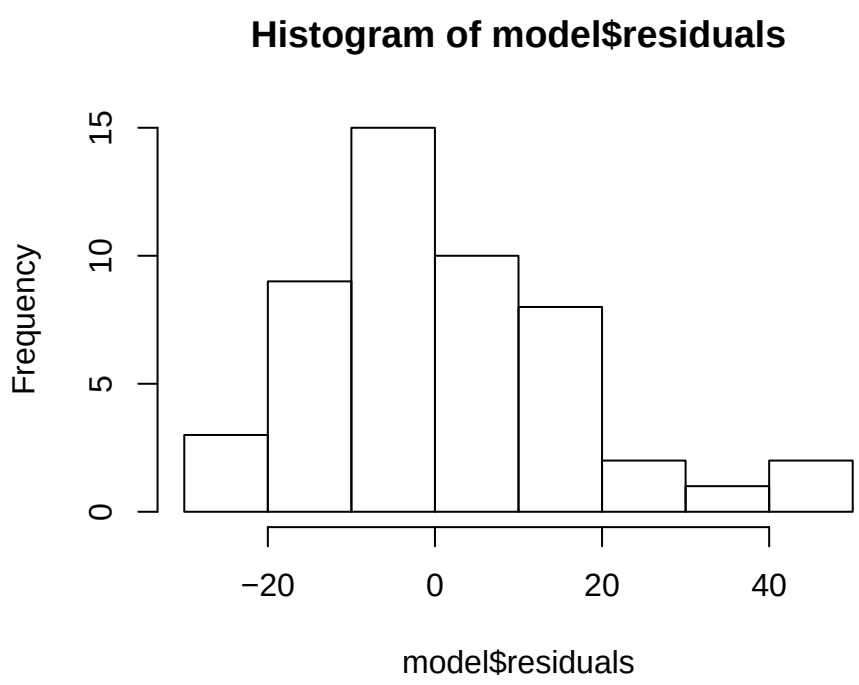
4. Funktioner: Statistik och grafik (4p)

   (a) Läs först in datamaterialet `cars` med funktionen `data()`. Skapa sedan funktionen `myRegression(x=,y=)`, som ska göra följande: **2p**

   i. Argument: `x` - numerisk vektor, `y` - numerisk vektor (båda av samma längd)
   ii. Gör en linjär regression med `y` som beroende variabel och `x` som förklarande variabel.
   iii. Plotta ett histogram över residualerna (felen i modellen).
   iv. Skapa en matris som har kolumner för skattning av regressionskoefficienterna (Estimate), standardfel (Std. Error) och p-värden. Raderna ska innehålla värden för β_0 och β_1. Kolla på matrisen `test` i exemplet nedan för att se hur matrisen ska se ut och vilka rad- och kolumnnamn som den ska ha. Kalla matrisen `resultatMat`.
   v. Returnera matrisen `resultatMat.`

   Testa din funktion med testexemplet nedan:

```
test <- myRegression(x = cars$speed, y = cars$dist)
```




```
test

             Estimate Std. Error    Pr(>|t|)
(Intercept) -17.57909   6.758440 1.23188e-02
x             3.93241   0.415513 1.48984e-12
```

(b) Skapa en scatterplot med `cars$speed` på x-axeln och `cars$dist` på y-axeln. Använd plot-funktioner från base package, dvs de valiga funktionerna för plottar. Axelnamnen ska vara "speed" för x-axeln och "cars" för y-axeln. Lägg sedan till skattad regressionslinje till plotten. **1p**

(c) Skapa en scatterplot med cars$speed på x-axeln och cars$dist på y-axeln, med tillhörande regressionslinje, men använd **endast** plot-funktioner från ggplot2-package. **1p**

5. Funktioner: Tärningar (4.5p)

I denna uppgiften betyder D6 en vanlig 6-sidig tärning (med sidorna 1,2,3,4,5,6), och där alla utfall har samma sannolikhet (1/6).

(a) Skapa en funktion kallad `myD6(N=)`, som ska simulera N stycken kast från en D6 och returnerar dessa som en numerisk vektor. Tex om `N=3` kan resulatet bli talen $(2, 4, 1)$. **1p**

(b) Skapa funktionen `sumOfDice(M=)`, som ska göra följande: simulera `M` kast från en D6 och summera värdena från kasten. Sedan ska funktionen returnera summan. Tex om `M=4`, kan vi få värdena $(1, 5, 3, 1)$ och då ska talet 10 returneras. **1p**

(c) Skapa funktionen `randomSum(K=,mySeed=)`, som ska göra följande: **2.5p**

i. Argument: `K` - antal simuleringar, `mySeed` - seed för att kontrollera slumptalsgeneringen, defaultvärde ska vara NULL.

ii. Börja med att sätta seed till `set.seed(mySeed)`.

iii. Sätt upp en tom matris med två kolumner och `K` antal rader. Döp första kolumnen till `noDice` och den andra till `values`. Kalla matrisen `myResult`.

iv. Gör en for-loop som ska loopa över talen $1, 2, 3, ..., K$. I varje iteration ska loppen: först simulera **ett** kast med en D6 och spara värdet som `x`. Sen ska `sumOfDice()` anropas med `M=x`, spara resultatet i `y`. Spara sedan `x` i första kolumnen och `y` i andra kolumnen i matrisen `myResult` på aktuell rad.

v. Ta fram beskrivande statistik över kolumnen `values` i `myResult` med funktionen `summary()`, spara i vektorn `stats`.

vi. Skapa en lista där första element ska vara vektorn `stats` och andra elementet ska vara matrisen `myResult`. Namnen ska vara "stats" för första element och "values" för andra element.

vii. Returnera listan från föregående steg.

Testa om din funktion klarar testfallen nedan:

```
# test 1:
randomSum(K = 6, mySeed = 431)
$stats
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   5.00    5.25    7.50    9.17   10.50   19.00

$values
     noDice values
[1,]      5     19
[2,]      3      9
[3,]      1      5
[4,]      1      6
[5,]      1      5
[6,]      5     11
# test 2:
randomSum(K = 12, mySeed = 871)
$stats
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    5.0     7.0    10.5    12.2    12.8    31.0

$values
      noDice values
 [1,]      5     11
 [2,]      3     12
 [3,]      3      7
 [4,]      6     31
 [5,]      1      5
 [6,]      3      8
 [7,]      4     10
 [8,]      2      7
```

```
 [9,]      3      5
[10,]      4     12
[11,]      6     24
[12,]      6     15
```

Lycka till!