LINKÖPINGS UNIVERSITET
Institutionen för datavetenskap
STIMA
Josef Wilzén

2019-03-29
Programmering i R, 7.5 hp
732G33

Tentamen i Programmering i R, 7.5 hp

Skrivtid: 8.00-12.00
Hjälpmedel: Inget tryckt material, dock finns "R reference card v.2"
och några andra referenskort tillgängliga elektroniskt.
Betygsgränser: Tentamen omfattar totalt 20 poäng. 12 poäng ger Godkänt, 16 poäng ger Väl godkänt.
Tänk på följande:
Skriv dina lösningar i **fullständig och läsbar kod**. Kommentera din kod och använd en god kodstil.
Spara filen med namnet **tentaX.R** där **X** är ditt SC-nummer (klientnummer). Det numret kan du se i studentklienten. Exempel: om du har SC-nummer SC12345 så ska filen heta tentaSC12345.R
Filen lämnas sedan in via studentklienten. Notera att du ska lämna in **en** fil med alla dina lösningar.
När du har loggat ut från datorn är tentan avslutad. Frågor kan ställas till lärare via studentklienten.
Spara filer i mappen `/home/student_tilde/`
Kommentera direkt i din R-fil 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.

Programvaror

För att ladda in kursmodulen, kör följande kommando i en terminal

```
module load courses/732G33
```

så kommer du få tillgång till R, RStudio samt de R-paket som behövs för tentan. Öppna en terminal genom att trycka ctrl+alt+T

Kör sedan i terminalen

```
rstudio
```

för att öppna Rstudio.
Övriga program:

- `caja` används som filhanterare
- `pluma` används för att läsa textfiler, kan användas för att titta på datafiler innan inläsning.

Uppgifter

1. Datastrukturer och beräkningar (4p)

   (a) Läs in det inbyggda datamaterialet `ChickWeight` i R. Använd sedan funktioner i R för att göra följande: Välj ut de observationer där variablen `Time` har värdet 21, spara som `time_21`. Räkna ut medianen för variabeln `weight` grupperat på variablen `Diet` för `time_21`. Byt sedan namn på variablerna i `time_21` till "W", "T", "C" och "D" (i den givna ordningen) **1.5p**

   (b) Skapa en lista med fyra element med namnet `my_list`. Låt x vara en vektor med heltalen $1, 2, 3, \ldots, 9, 10$. Listelmentens namn ska vara "x1", "x2", "x3" och "sum". Första elementet ska innehålla $log_{10}(x)$[1]. Det andra ska innehålla x^2 och det trejde ska innehålla x^3. Det fjärde elementet ska innehålla summan av alla värden i de tre första elementen. **1.5p**

   (c) Återskapa matrisen nedan. Se till att ha rätt kolumnnamn. **1p**

```
      abc ABC zyx ZYX
 [1,] "a" "A" "z" "Z"
 [2,] "b" "B" "y" "Y"
 [3,] "c" "C" "x" "X"
 [4,] "d" "D" "w" "W"
 [5,] "e" "E" "v" "V"
 [6,] "f" "F" "u" "U"
 [7,] "g" "G" "t" "T"
 [8,] "h" "H" "s" "S"
 [9,] "i" "I" "r" "R"
[10,] "j" "J" "q" "Q"
[11,] "k" "K" "p" "P"
[12,] "l" "L" "o" "O"
```

2. Kontrollstrukturer (4p)

   (a) Utgå från vektorerna $x = \begin{pmatrix} 1 & 3 & 9 \end{pmatrix}$ och $y = \begin{pmatrix} 10 & 23 & 40 \end{pmatrix}$. Dessa beskriver en samling rektanglar. x anger längden på kortsidan och y anger längden på långsidan. Lös följande med en *nästlad for-loop*: räkna ut kvoten mellan arean och omkretsen för alla kombinationer av kortsidor och långsidor på rektanglarna: $kvot = area/omkrets$. Dessa kvoter ska avrundas till tre decimaler och sen skrivas ut till konsolen tillsamans med information om längderna på rektangels sidor. Om du gjort rätt ska du erhålla utskriften nedan. **2p**

```
[1] "kortsida = 1 långsida = 10 kvot = 0.455"
[1] "kortsida = 1 långsida = 23 kvot = 0.479"
```

[1] log_{10} är logaritmen med bas 10.

```
[1] "kortsida = 1 långsida = 40 kvot = 0.488"
[1] "kortsida = 3 långsida = 10 kvot = 1.154"
[1] "kortsida = 3 långsida = 23 kvot = 1.327"
[1] "kortsida = 3 långsida = 40 kvot = 1.395"
[1] "kortsida = 9 långsida = 10 kvot = 2.368"
[1] "kortsida = 9 långsida = 23 kvot = 3.234"
[1] "kortsida = 9 långsida = 40 kvot = 3.673"
```

(b) Utgå från det inbyggda datasetet `diamonds` som finns i paketet `ggplot2`. Börja med att bearbeta data enligt nedan. Du ska med hjälp av en while-loop gå igenom de olika variablerna (kolumnerna) i datasetet och skriva ut ett meddelande till konsolen för varje variabel. Notera att olika saker skrivs ut beroende på om variabeln är en factor eller inte. Om du gjort rätt ska du få utskriften enligt nedan. Numeriska värden avrundas till tre decimaler. **2p**

```
# bearbeta diamonds:
diamonds<-as.data.frame(diamonds)
diamonds<-diamonds[1:1000,]
head(diamonds,4)

  carat     cut color clarity depth table price    x    y    z
1  0.23   Ideal     E     SI2  61.5    55   326 3.95 3.98 2.43
2  0.21 Premium     E     SI1  59.8    61   326 3.89 3.84 2.31
3  0.23    Good     E     VS1  56.9    65   327 4.05 4.07 2.31
4  0.29 Premium     I     VS2  62.4    58   334 4.20 4.23 2.63

# utskrift från while-loopen:
```

```
[1] "name = carat, mean = 0.689, sd = 0.195"
[1] "name = cut, levels: Fair, Good, Very Good, Premium, Ideal"
[1] "name = color, levels: D, E, F, G, H, I, J"
[1] "name = clarity, levels: I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF"
[1] "name = depth, mean = 61.723, sd = 1.759"
[1] "name = table, mean = 57.735, sd = 2.468"
[1] "name = price, mean = 2476.54, sd = 839.576"
[1] "name = x, mean = 5.606, sd = 0.625"
[1] "name = y, mean = 5.599, sd = 0.612"
[1] "name = z, mean = 3.458, sd = 0.39"
```

3. Strängar och datum (4p)

   (a) Läs in paketen `lubridate` och `stringr` i R. Läs sedan in datamaterialet "word_data.txt" och spara som vektorn `my_word`. **0.5p**

   (b) Använd stringr för att: Välj ut de ord i `word_data` som har en vokal (a,e,i,o,u eller y) som andra tecken och sista tecken i ordet. Alla andra tecken kan vara antigen vokaler eller konsonanter. Exempel på ord som uppfyller kraven är: "hate", "necessary" och "lie". Spara som `my_word`. Räkna sen ut den genomsnittliga längden på orden i `my_word`, spara som `average_length`. **1.5p**

```
head(my_word)

[1] "line"      "terrible"  "debate"    "notice"    "positive"  "yesterday"
```

   (c) Nu ska du skapa funktionen `random_days(n,y,m)`. Funktionen ska generera ett antal (argumentet `n`) slumpmässiga dagar för ett givet år (argumentet `y`) och en given månad (argumentet `m`). Så tex `random_days(n=5,y="2012",m="04")` innebär att vi vill generara 5 slumpmässiga dagar från april 2012. Alla månadens dagar ska ha samma sannolikhet att bli dragna, men samma dag ska **inte** kunna bli dragen flera gånger. Om argementet `n` är större än det faktiska antalet dagar i den aktuella månaden så ska funktionen avbryta. Funktionen ska returnera en vektor med Date-klass. Se nedan hur funktion ska fungera. **2 p**

```
random_days(n = 40,y = "1999",m = "03")

Error in random_days(n = 40, y = "1999", m = "03"): n is too large!

# notera att pga slumpen så kan du få andra datum i exemplen
a1<-random_days(n = 4,y = "1999",m = "03")
a1

[1] "1999-03-15" "1999-03-10" "1999-03-04" "1999-03-24"

class(a1)

[1] "Date"

length(a1)

[1] 4

random_days(n = 1,y = "2012",m = "12")

[1] "2012-12-19"

a3<-random_days(n = 12,y = "2201",m = "09")
a3

 [1] "2201-09-30" "2201-09-12" "2201-09-05" "2201-09-21" "2201-09-19"
 [6] "2201-09-01" "2201-09-10" "2201-09-03" "2201-09-25" "2201-09-02"
[11] "2201-09-29" "2201-09-28"
```

4. Funktioner: **4p**

(a) Skapa nu funktionen `my_scale(x)`. Funktionen ska ta objektet `x` (som kan vara en matris eller en data.frame) och standarisera alla numeriska variabler. Standardiseringen ska ske enligt:

$$z_i = \frac{x_i - \mu_i}{\sigma_i}$$

Där x_i är den i:te kolumnen i `x`, μ_i är medelvärdet för x_i, och σ_i standardavvikelsen för x_i. Variabler som inte är numeriska ska inte transformeras på något sätt. Funktionen ska returnera ett objekt av samma klass och storlek som `x`, men där ev numeriska variabler är standardiserade. Det är inte tillåtet att använda några inbyggda funktioner som direkt standardiserar, tex så är funktionen `scale()` inte tillåten i denna uppgift. Se exemplen nedan hur funktionen ska fungera.

```
X1<-my_scale(x = iris)

head(X1,3)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1    -0.897674    1.015602     -1.33575    -1.31105  setosa
2    -1.139200   -0.131539     -1.33575    -1.31105  setosa
3    -1.380727    0.327318     -1.39240    -1.31105  setosa

round(mean(X1[,3]),10)

[1] 0

round(mean(X1[,1]),10)

[1] 0

sd(X1[,4])

[1] 1

dim(X1)

[1] 150   5

dim(iris)

[1] 150   5

class(X1)

[1] "data.frame"

X2<-my_scale(x = faithful)
dim(X2)

[1] 272   2

head(X2,5)
```

```
   eruptions   waiting
1  0.0983176  0.596025
2 -1.4787328 -1.242890
3 -0.1356115  0.228242
4 -1.0555576 -0.654437
5  0.9157554  1.037364

X3<-my_scale(matrix(TRUE))
X3

     [,1]
[1,] TRUE

class(X3)

[1] "matrix"

my_scale(matrix(1:5,5))

          [,1]
[1,] -1.264911
[2,] -0.632456
[3,]  0.000000
[4,]  0.632456
[5,]  1.264911
```

(b) Skapa nu funktionen `my_curve(x,a=1)` enligt formeln nedan. Funktionen ska avbryta om `x` inte är numerisk. Se exemplen för hur funktionen ska ska fungera.

$$y = f(x) = \begin{cases} sin(a \cdot 2\pi \cdot x) \cdot x & x \leq 0 \\ cos(0.2 \cdot \pi \cdot e^x) & x > 0 \end{cases}$$

```
my_curve(x = "hej")

Error in my_curve(x = "hej"): x is not numeric

x1<-c(-0.25,0,1)
my_curve(x = x1)

[1]  0.250000  0.000000 -0.136721

my_curve(x = x1,a = 1.2)

[1]  0.237764  0.000000 -0.136721

my_curve(x = x1,a = 3.14)

[1] -0.243979  0.000000 -0.136721

x2<-c(-30.2,-0.125,10,12)
my_curve(x = x2)
```

```
[1] 28.7219068  0.0883883 -0.6050354 -0.9914245
```

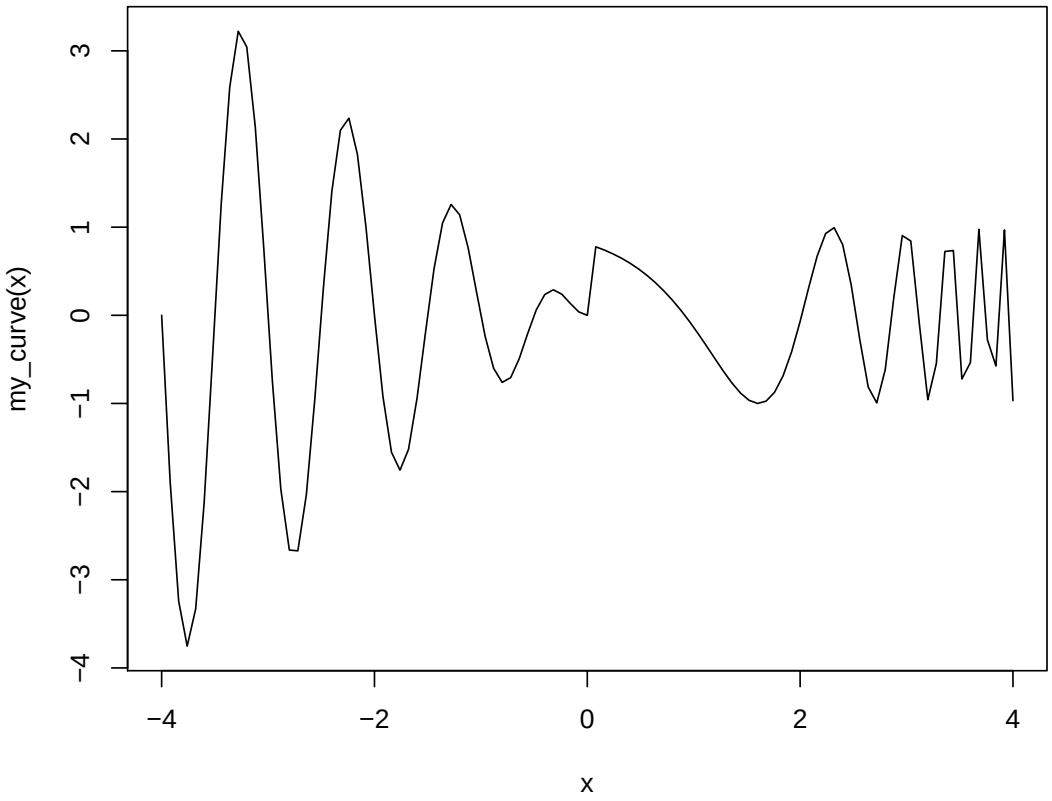
```
my_curve(x = x2,a = 2)
```

```
[1] 17.751115  0.125000 -0.605035 -0.991425
```

```
my_curve(x = x2,a = 3.1)
```

```
[1] -20.673323   0.081181  -0.605035  -0.991425
```

```
curve(expr = my_curve,from = -4,to = 4)
```



5. Linjär algebra, statistik, grafik (4p)

(a) Använd funktioner i R för att lösa ekavtionsssytemet nedan. Spara lösningen i variabeln `my_x`: **0.5p**

$$Ax = b \Leftrightarrow$$

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 2 & 3 & 4 \\ 3 & 3 & 3 & 4 \\ 4 & 4 & 4 & 4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 10 \\ 2 \\ -4 \\ 5 \end{pmatrix}$$

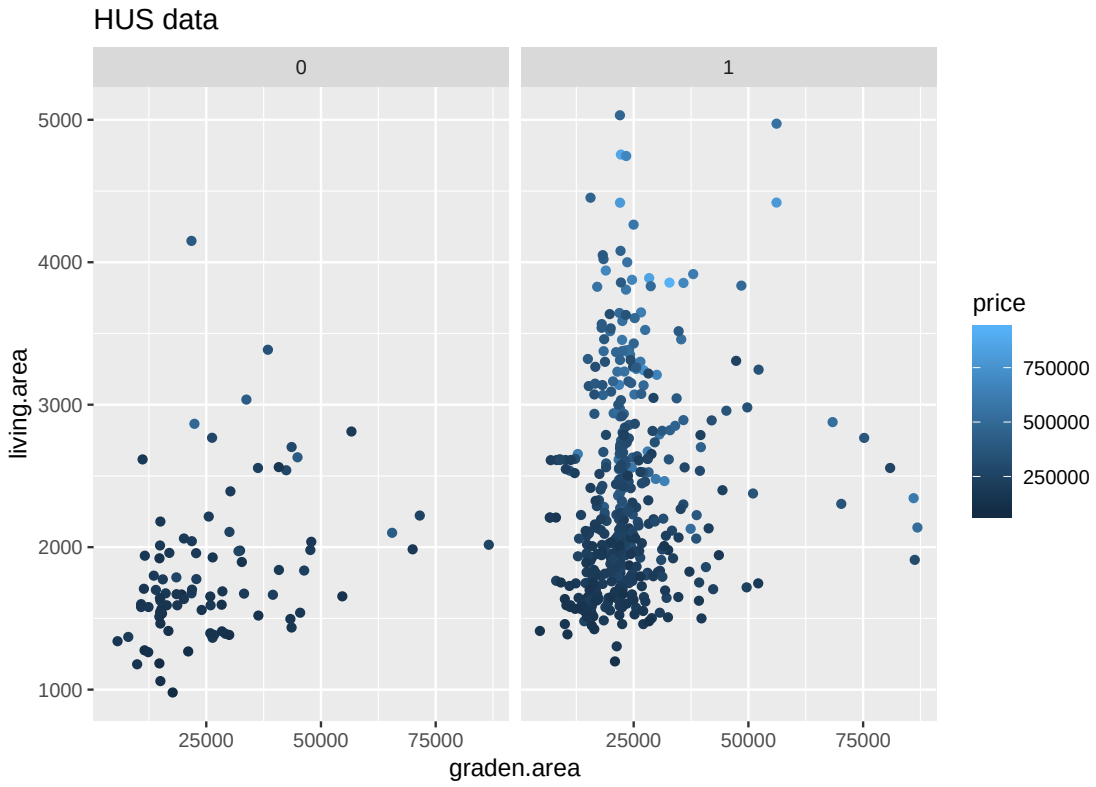
(b) Läs in datamaterialet "HUS_eng.csv" i R och spara som en data.frame med namnet `HUS`. **1.5p**

i. Skapa en ny data.frame, som du kallar `HUS2`, som endast innehåller de hus som har 2,3,4 eller 5 sovrum (`no.bedroom`). Utgå från `HUS2` i resten av uppgift (b).

ii. Skapa sedan en korstabell mellan antal sovrum och luftkonditionering (`air.conditioning`). Spara objektet som `my_tab`.

iii. Gör sedan ett chitvå-test på tabellen från (ii). Spara teststatistikan i variablen `test_stat` och spara p-värdet i variablen `p_value`.

(c) Återgå nu till din data.frame `HUS`, och använd ggplot2 för att göra följande: Återskapa figuren nedan. Figuren är en scatterplot mellan variablerna `graden.area` och `living.area`. Färgen på punkterna beror på variabeln `price`. Figuren är grupperad med avseende på variablen `air.conditioning` och har en titel. **2p**



Kom ihåg: Skriv alla dina lösningar i en körbar **R-fil**. Spara filen med namnet **tentaX.R** där **X** är ditt SC-nummer (klientnummer) Det numret kan du se i studentklienten. Exempel: om du har SC-nummer SC12345 så ska filen heta tentaSC12345.R Lämna sedan in din fil via studentklienten. När du har loggat ut från datorn är tentan avslutad.

Lycka till!