Matematisk Statistik

Lunds Universitet

Introduktion till R

en guidad tur

VT 2018

Introduktion till R

I denna datorövning kommer steg för steg att lära oss vissa av de grundläggande funktionerna i R. Programmet kan laddas ner gratis från www.r-project.org. Det finns också ett praktiskt användargräns-snitt, RStudio, som kan laddas ner från www.rstudio.com. Notera att du måste installera R innan du installerar RStudio. Både R och RStudio finns installerat på datorerna i MH230.

Tonvikten i häftet kommer att ligga på de statistiska beräkningar och metoder som används under dina statistikstudier. Men innan vi börjar med uppgifterna behövs en kortfattad beskrivning av hur R fungerar. Den som tidigare har arbetat med kalkylprogram – t ex Excel – känner till viss del igen sig medan andra funktioner är helt annorlunda. Kalkylarket, databladet, där man matar in data ser likadant ut men det finns väsentliga skillnader.

> *I kalkylprogram kan man i samma kalkylark blanda data, resultat och diagram. I ett statistikprogram får endast individdata finnas på "databladet". Alla resultat, analyser och diagram, hamnar i separata fönster och kan inte sparas tillsammans med data utan måste sparas i separata filer eller flyttas över till ett ordbehandlingsprogram och sparas som ett dokument.*
>
> *I R kan man också spara vissa resultat i egna variabler, utanför databladet.*

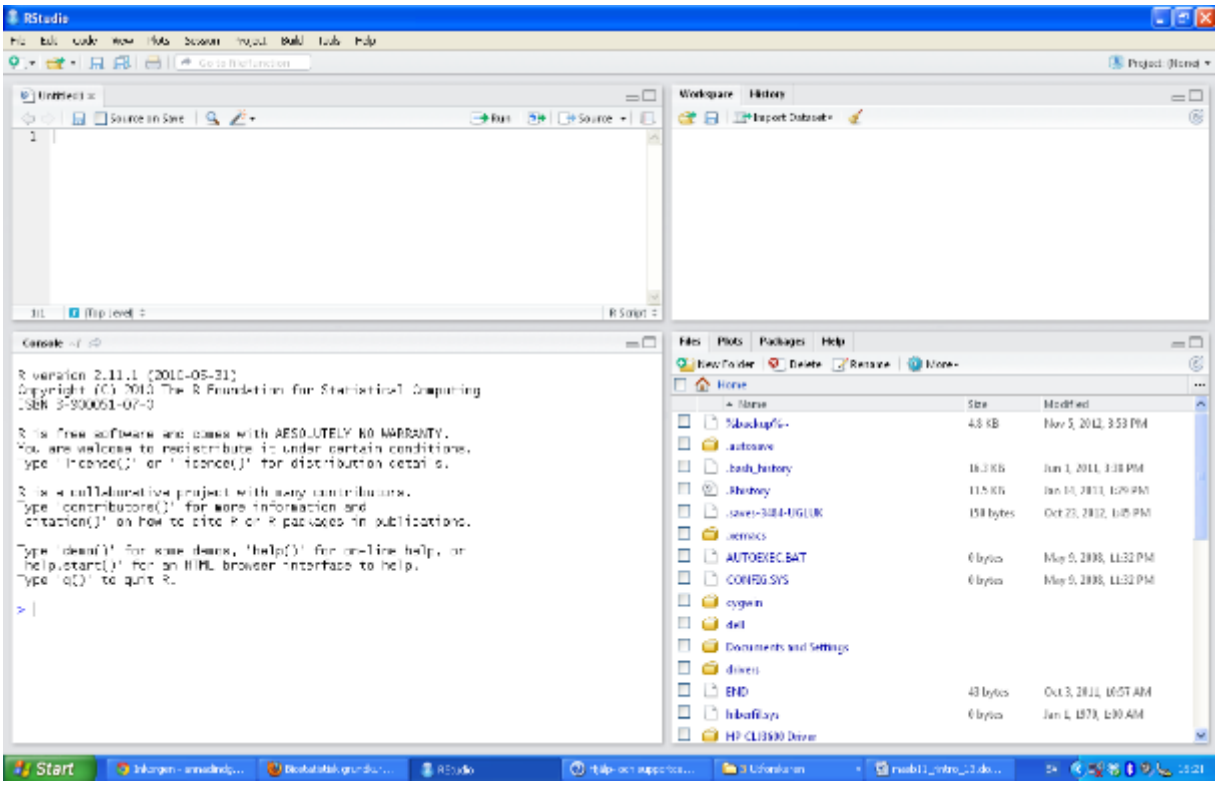
Programmet RStudio startas från Windows genom att man dubbelklickar på RStudio-ikonen och efter ett antal sekunder kommer skärmen att se ut ungefär som i figur 1.

Nere till vänster finns RStudios **console**, också kallat **command window**, där man kan exekvera kommandon direkt genom att skriva efter promptern > och trycka Enter.

I övre högra hörnet finns **environment/history window** (obs. workspace/history i äldre versioner). När environment fliken är aktiv kan du se vilka variabler som R har i minnet, om du klickar på dem kan du se vad de innehåller. När history fliken är aktiv kan du se vilka kommandon som har getts.

I nedre högra hörnet kan du se dina filer och plottar samt ladda ner så kallade packages, en samling av program. Här finner du också R's hjälpfunktion.

I övre vänstra hörnet finns R's **'editor window'** eller **'script window'**. Det är här vi skriver våra program, eller scripts. Om det inte syns kan du öppna det om du väljer fliken File/New File/ R Script.
Ett script i sin enklaste tappning är bara en sekvens av R-kommandon. Utför man samma beräkningar många gånger kan det vara bra att samla kommandona i ett R-script, så slipper man skriva om allting från gång till gång. Det är alltså en god idé att spara scriptet så man kan återskapa vad man gjorde.



I undre vänstra fönstret (Console) skrivs också resultaten av körningarna: korstabeller, medelvärden, med mera ut. Det är bara när beräkningarna skapat en ny variabel som de hamnar i Environmet. Man har t.ex. födelseår för individerna och vill räkna om det till ålder. Då skapar man en ny variabel och låter R göra tranformationen ålder = 2014-födelseår.

När man använder ett datablad ser man det inte om man inte särskilt ber om det. Data matas alltid in så att varje rad motsvarar en individ och varje kolumn en variabel, t.ex. ålder, kön, etc. Man kan mata in data för hand eller läsa in en datafil från något annat program.

> *En viktig skillnad gentemot kalkylprogrammen är också att de nya värdena räknas ut och läggs som data, och inte som en formel, i den nya kolumnen.*

Nog med prat, det bästa sättet att lära sig ett datorprogram är att själv slå sig ner och trycka på tangenterna och klicka med musen. Skulle det hänga upp sig någonstans finns alltid möjlighet att få hjälp. Botanisera under Help-fliken nere till höger eller under Help på menyraden.

I fortsättningen av kompendiet används konventionen att de kommandon man ska skriva är i Calibri, medan den vanliga texten är skriven med Times Roman/.

Innan vi börjar på riktigt ska vi ställa in vår arbetskatalog. Det är här R kommer att spara och läsa in datamaterial mm. Klicka i menyn på **Session** och välj **Set working directory** och sen

choose directory. Här väljer du t.ex. katalogen ”My Documents” . Man kan också skriva kommandot

```
setwd(”sökväg/My Documents”)
```

Vill man veta vilken katalog man arbetar mot skriver man kommandot

```
getwd()
```

Räkna med R

R är till för att räkna ut saker. Vill man veta vad 2 + 4 är så skriver man

```
2+4
```

och R svarar

```
> 2+4
[1] 6
```

[1] innebär att det första värdet i svaret är 6. (I detta fall finns det ju bara ett värde.) Vill man räkna ut en multiplikation skriver man

```
2*4
```

Självfallet finns i princip alla normala matematiska funktioner. För att räkna ut 4^2, $\sqrt{4}$, $\log_e 4$ och e^4 skriver man

```
4^2
sqrt(4)
log(4)
exp(4)
```

Notera att i R är decimaltecknet en punkt . och inte komma. Vill man spara resultatet av en uträkning får man tilldela det ett namn. För det används tilldelningskommandot ←. Om vi vill räkna ut vad 2 + 4 är och spara resultatet under namnet ”mitt.svar” skriver man

```
mitt.svar <- 2+4
```

Notera att R inte ger någon bekräftelse – du förutsetts veta vad du gör. Om du vill veta vad svaret är så får du snällt be R tala om det genom att skriva

```
mitt.svar
```

Det som händer rent tekniskt är att vi har skapat en variabel med namnet ”mitt.svar” som innehåller värdet 6. Variabeln kan vi sedan använda i en funktion. Vi kan t.ex. skriva

```
sqrt(mitt.svar)
```

så får vi kvadratroten ur 6. sqrt() är en *funktion*. Alla funktioner i R slutar med parenteser – även om det inte finns något argument. R består av en stor (enorm) mängd funktioner för att räkna ut olika saker.

Variabler

För att sätta ihop flera värden till en variabel (en vektor) använder man funktionen c() av engelska combine.

```
x <- c(3, 5, 7, 11, 13)
```

Man kan sedan utföra samma matematiska beräkningar som tidigare, fast nu på alla värdena samtidigt.

```
x + 3
sqrt(x)
```

Man kan också sätta ihop flera variabler till en lång variabel.

```
y <- c(17, 19, 23, 29, 31)
z <- c(x, y)
z
```

Ibland vill man skapa strukturerad data. Till exempel serier eller upprepade sekvenser. För detta finns det två kommandon: seq() och rep(). Dessutom kan man använda kolontecken. Pröva följande kommandon och försök förstå vad de gör.

```
seq(1, 100, 9)
seq(to=100, from=1, by=9)
seq(f=1, t=100, length.out=10)
1:3
3:1
rep(c(1,2,3), times=3)
rep(1:3, each=4)
rep(1:3, t=3, e=4)
rep(1:3, length.out=20)
```

Om man behöver hjälp med att förstå vad en viss funktion gör eller används kan använda hjälpfunktionen genom att skriva help(seq) eller ?seq. (Om man vill ha hjälp med någon annan funktion istället för seq skriver man såklart den funktionens namn t.ex. ?rep. Kolontecknet är ingen funktion utan en operator och därför får man skriva help(":").)

Ibland så vill vi bara ha några få värden i en variabel. Vi kan välja ut vissa värden med hjälp av [].

```
varden <- 21:30
varden
varden[1]
```

```
varden[c(1, 3, 5)]
varden[1:3]
```

Man kan också välja att utesluta vissa värden:

```
varden[-1]
varden[-c(1, 3, 5)]
varden[-(2:4)]
```

Några funktioner

Som nämndes ovan finns massor med funktioner i R. Här följer några exempel på grundläggande statistiska funktioner. Det bör vara ganska lätt räkna ut vad funktionerna gör. (Det första kommandot skapar 100 standardiserade normalfördelade slumptal.)

```
x <- rnorm(100)
x
mean(x)
var(x)
sd(x)
median(x)
boxplot(x)
boxplot(x, horizontal=TRUE)
hist(x)
```

De sista kommandona var ju exempel på diagram. Den grundläggande (och kanske viktigaste) funktionen för att rita diagram är dock plot() som ritar spridningsdiagram. Vi ska återkomma till den längre fram. Nu ger vi bara ett litet smakprov, genom att visa hur man kan rita upp täthetsfunktionen för en gammafördelning:

$$f(x) = 4x^2e^{-2x}$$

```
x <- seq(0, 10, .01)
f <- 4*x^2*exp(-2*x)
plot(x, f, type="l", main="En täthetsfunktion")
```

Objekt

Alla variabler är objekt. Vill man veta vilka objekt man har skapat så använder man kommando ls() (av engelska *list objects*). Vill man radera ett objekt så använder man kommandot remove().

```
skrap <- c(1, 19, 23.4)
ls()
remove(skrap)
ls()
```

Observera att R som vanligt inte ifrågasätter vad man ber om. Man bör alltså vara försiktig när man använder remove() eftersom det är svårt att ångra sig. Funktioner är för övrigt också objekt som man kan skapa och radera.

Att skriva script i RStudio

Ska man göra större beräkningar är det mer praktiskt att skriva ett program, eller script som det kallas i R. Vi skapar ett nytt fönster i editorn genom att trycka CTRL+Shift+Enter, eller genom att välja File/New File/R Script.
Vi skapar ett script som vi sparar som MittScript.R, där vi löser några uppgifter.

Exempel A: skapa funktionen

$$y = 5 + \frac{10.8}{x^2}$$

I vårt script skriver vi:

```
Minfunktion<-function(x){
 y=5+10.8/x^2
 y
}
```

Innan vi kan använda funktionen måste man trycka på ”source” i editor fönstret. Efter att vi gjort det kan vi använda den både i consolen och i vårt script. I consolen kan vi t.ex. skriva:

```
x<-1:5
Minfunktion(x)
[1] 15.800  7.700  6.200  5.675  5.432
```

Vill vi skriva det i vårt script måste vi använda oss av kommandot ”print” för att få resultatet utskrivet till consolen:

```
x<-1:5
print(x)
y<-Minfunktion(x)
print(y)
```

För att få denna del av programmet exekverat markerar vi det, och trycker därefter på ’run’, eller trycker CTRL+Enter.

Exempel B: plotta funktionen Minfunktion. I ett script skriver vi:

```
xvalue<-seq(0,5,0.2)
yvalue<-Minfunktion(xvalue)
plot(xvalue,yvalue, main="Plot av f(x)=5+10.8/x^2", xlab="x", ylab="f(x)", type="b")
```

Beskrivande Statistik - om data frames

I denna första uppgift ska vi mata in ett mindre datamaterial och göra några beräkningar. Till kursen MASB11 hade våren 2011 anmält sig bland annat följande 6 personer:

	Namn	Kön	Ålder	Poäng	Termin
1	Lasse	1	23	112.5	4
2	Bosse	1	20	45	2
3	Lisa	2	19	30	1
4	Anna	2	24	52.5	2
5	Britta	2	25	270	6
6	Olle	1	27	300	8

Variabeln Poäng innehåller information om hur många högskolepoäng (hp) individerna har vid kursens start. Variabeln Termin är antalet terminer av universitetsstudier. I R är det enklast, när vi har så lite data, att skriva in datamaterialet direkt i en så kallad Data Frame. Data Frame är R:s att samla ihop variablerna i ett datamaterial i ett objekt och motsvarar ett datablad. Skriv följande kommando i övre vänstra delfönstret (<- skrivs som ”mindre än” och ”bindestreck”)

```
masb11 <- data.frame(Namn = c("Lasse", "Bosse", "Lisa", "Anna", "Britta", "Olle"),
        Kön = c(1, 1, 2, 2, 2, 1),
        Ålder = c(23, 20, 19, 24, 25, 27),
        Poäng = c(112.5, 45, 30, 52.5, 270, 300),
        Termin = c(4, 2, 1, 2, 6, 8),stringsAsFactors=FALSE)
```

Själva kommandoraderna säger åt R att göra en Data Frame vid namn masb11 med variablerna Namn, Kön, Ålder, Poäng och Termin där variablerna ligger i varsin kolumn, c(...). Observera att R kräver decimalpunkt. Kommatecken betyder uppräkning. Citattecknen kring namnen anger att det är fråga om text, inte tal. Spara nu kommandofilen i din katalog. Ett lämpligt filnamn kan vara lab1.R. Kommandofiler i R slutar på .R. Det är en god idé att spara denna fil så fort man ändrat i den.

Markera sedan raderna och utför kommandot genom att klicka på ”Run”. Alternativt kan du ställa markören på den första raden och sedan köra raderna en efter en med Ctrl-R. Kommandot skrivs ut i Console nere till vänster allt eftersom. Ett plustecken betyder att kommandot inte är fullständigt och R väntar på nästa del. Om det blev fel kommer ett felmeddelande här också. Uppe till höger i enviroment kan du nu se att du fått ett objekt som heter masb11 med 6 observationer och 5 variabler.

För att se hur data ser ut kan du skriva och köra något av kommandona:

```
masb11

View(masb11)
```

Det första kommandot skriver ut innehållet i masb11 på konsolen. Det andra öppnar ett nytt fönster med innehållet. Du kan också dubbelklicka på masb11 i enviroment uppe till höger så öppnas data i ett fönster uppe till vänster. Du kan inte ändra i data här. Stäng datafönstret och skriv och kör kommandot:

```
masb11 <- edit(masb11)
```

Nu får du upp ett nytt fönster där du kan ändra data och lägga till fler personer och variabler. Allt du gör här sparas i masb11 så blir det fel får du börja om från början. Det är därför det är klokt att spara kommandona.

Lägg in dig själv som en sjunde person i datamaterialet.

Du kan också lägga till en 7:e person så här istället (som heter Pelle, är man, 23 år, har 45 hp på 4 terminer: byt ut mot dina värden):

```
masb11[7,”Namn”] <- ”Pelle” # Om kommandot inte fungerar använd edit(masb11) istället

masb11[7,2:5] <- c(1, 23, 45, 4)
```

Här betyder [7,”Namn”] att den sjunde radens Namn-kolumn i data-framen masb11 ska fyllas på med värdet ”Pelle” medan [7,2:5] betyder att rad 7:s kolumn 2-5 (dvs Kön-Termin) ska få värdena i c(…). Eftersom Namn är text och resten tal är det enklast att dela upp det så här.

Man behöver inte spara datamaterial separat i R. Man kan ha flera olika datamaterial och extra variabler samtidigt. Då kan man spara allt detta tillsammans som en ”Workspace”. Klicka på den lilla disketten i Environment-fönstret uppe till höger för att spara alla dina data till en fil i din katalog. Filen kommer att heta filnamn.RData som är R:s namn på workspace-filer.

Värdet 1 för män och 2 för kvinnor är naturligtvis praktiskt vid inmatning av data men vid utskrifter vill man hellre se det riktiga värdet, dvs Man eller Kvinna. Då anger man att värdena i variabeln Kön ska hanteras, inte som tal, utan som Faktorer. Det gör vi med kommandot

```
masb11$Kön <- factor(masb11$Kön, labels = c(”Man”, ”Kvinna”))
```

Här ändrar vi på variabeln Kön i data-framen masb11 (dollartecknet låter oss komma åt variablerna i en frame), och byter ut talen mot etiketter, labels, där det lägsta talet (1) ersätts med ”Man” och det andra (2) med ”Kvinna”. Se hur det blev med, t.ex. View(masb11).

Innan vi går vidare ska vi också se ett enkelt exempel på hur man kan skapa en ny variabel. Antag nu att vi vill bilda en ny variabel, Ptermin, som visar hur många poäng studenterna i genomsnitt har tagit per termin de studerat. Vi lägger till den nya variabeln till de andra i masb11 med hjälp av dollartecken:

```
masb11$Ptermin <- masb11$Poäng / masb11$Termin
```

Se hur det blev genom att skriva masb11.

Nu fortsätter vi med några statistiska beräkningar. Vi börjar med en frekvenstabell för variabeln Kön:

```
table(masb11$Kön)
```

Här får vi reda på absoluta frekvenserna, dvs, hur många av varje kön det är. Vill vi ha relativa frekvenser också kan vi göra så här:

```
antal <- table(masb11$Kön)

andel <- prop.table(antal)

andel

cbind(antal,andel)
```

Det första kommandot sparar tabellen och lägger den i variabeln antal. Det andra kommandot tar tabellen med antalen och beräknar andelar istället och sparar dem i variabeln andel. Nu har vi skapat två nya variabler, antal och andel, som INTE finns med i dataframen masb11 utan ligger separat som två tabeller i environment. Tredje kommandot skriver ut vad som finns i variabeln andel (det är inte nödvändigt men kan vara bra om man är nyfiken). Det sista kommandot, cbind, samlar sedan ihop de två tabellerna till en och presenterar resultatet. Det går bra eftersom de båda har två värden, ett för ”Man” och ett för ”Kvinna”.

Nästa steg blir att beräkna lite beskrivande mått för de numeriska variablerna i materialet. En snabbvariant är sammanställningskommandot

```
summary(masb11)
```

Här får du frekvenserna för text-variablen Namn och för faktorn Kön. Dessutom får du lite information om de numeriska variablerna, nämligen Min-Max-Median-Mean (=medelvärde)

och två kvartiler, dvs de värden som har 25% under (1st Qu.) respektive över (3rd Qu.) sig. Vi kan också göra en egen tabell över medelvärde (mean) och standardavvikelse (sd) för variablerna Ålder, Poäng, Termin och Ptermin. Detta går bra eftersom samtliga dessa variabler är av samma typ. Här väljer vi alla rader men bara vissa, namngivna, kolumner i masb11, sparar medelvärdena i en variabel och standardavvikelserna i en annan och presenterar sen resultatet:

```
medelv <- colMeans(masb11[,c("Ålder","Poäng","Termin","Ptermin")])

stdavv <- sqrt((colSums(masb11[,3:6]^2)-colSums(masb11[,3:6])^2/7)/6)

cbind(medelv,stdavv)
```

Pröva nu att kopiera in resultatet till Word: markera resultattabellen, gör Copy (Kopiera), öppna ett Word-dokument och gör Paste (Klistra in). Nu kan du snygga till resultatet i Word.

Gråsparvar

Nu ska vi studera ett lite större datamaterial. Det finns som ett sparat workspace, Grasparvar.RData, på kursens hemsida. Spara först ner filen i din katalog. Här följer en kort beskrivning av datamaterialet:

Datamaterial: Efter en svår februaristorm 1898 fångades 136 gråsparvar. Fåglarna var i dålig kondition och 72 hämtade sig medan 64 dog. I ett laboratorium mättes 8 morfologiska karaktärer samt vikten hos fåglarna:

Variabler:

Längd - Total längd (mm)

Vinglängd - Vinglängd (alar) (mm)

Vikt - Vikt (g)

HuvudNäbb - Längd av huvud och näbb (beak and head) (mm)

Överarmsben - Längd av överarmsben (humerus) (mm)

Lårben - Längd av lårben (femur) (mm)

Tars - Längd av tars (tibiotarsus) (mm)

Skallbredd - Skallbredd (skull) (mm)

Bröstbenskam - Längd av bröstbenskam (keel of sternum) (mm)

Ålder – Ålder (1=Adult, 2=Ung)

Kön – Kön (1=Hane, 2=Hona)

Överlevnad – Överlevnad (1=Död, 2=Överlevde)

Läs in datamaterialet i R så här: Under environment, klicka på ”load workspace” ikonen; Leta reda på filen och klicka på den. Nu ska du ha fått en ny Data Frame som heter Grasparvar.

Se hur datamaterialet ser ut med View(Grasparvar). I kolumnen Ålder står det ibland NA istället för 1 eller 2. NA står för Not Available och betyder att data saknas. I det här fallet är det honor som man inte kunnat avgöra om de är vuxna eller ungar; de ser likadana ut. För att se hur många det rör sig om kan vi skriva kommandot summary(Grasparvar).

Innan vi går vidare ska vi göra om variablerna Ålder, Kön och Överlevnad till faktorer med lämpliga etiketter. Gör det! Du behöver inte sätta en särskild label på Ålder=NA, det klarar R av själv.

Gör nu frekvenstabeller med både antal och andelar för de tre variablerna Ålder, Kön respektive Överlevnad. Vad hände med de saknade åldersvärdena?

Nu ska vi rita ett diagram över variabeln Ålder. I R finns lämpliga figurer fördefinierade till olika typer av data så vi kan helt enkelt säga:

```
plot(Grasparvar$Ålder)
```

Vad är detta för typ av diagram?

Om vi vill göra ett cirkeldiagram (pie) för Överlevnad istället måste vi använda frekvenstabellens data istället:

```
antal.overlevn <- table(Grasparvar$Överlevnad)

pie(antal.overlevn)
```

Du kan sätta andra färger på tårtbitarna med optionen col eller strecka bitarna med density och angle. Kommandot colors() ger en lista på alla färger som R känner till. Kommandot help(pie) ger mer förklaring och några exempel.

```
pie(antal.overlevn, col=c(”Blue”,”Yellow”))

# Streckade tårtbitar:

pie(antal.overlevn, density=c(20,10), angle=c(90,45))
```

För att får in figuren i ett Worddokument kan du göra så här: Plots: Export: Copy plot to clipboard och klistra sedan in den i ditt Worddokument.

Nu vill vi dela upp överlevnaden på Kön. I R kan man enkelt ange att en variabel (y) ska bero på en annan (x) så här: y ~ x. Tilde (~) anger är att det som står till vänster beror på det som står till höger. Vi kan då rita upp hur Överlevnaden beror på Kön:

```
plot(Grasparvar$Överlevnad ~ Grasparvar$Kön)
```

Tolka figuren. Vad är det de olika ytorna representerar?

För att få reda på hur stor andel av hanar respektive honor som dog gör vi först en korstabell och beräknar sedan relativa frekvenser, radvis:

```
antal <- table(Grasparvar$Kön, Grasparvar$Överlevnad)

antal

# margin=1 anger att det ska vara rad-andelar, margin=2 ger kolumnandelar, utan

# margin får man totalandelar.andel <- prop.table(antal, margin=1)

andel
```

(a) Hur stor andel av hanar respektive honor dog?
Hanar: _____%
Honor: _____%

Om vi vill rita ett cirkeldiagram av överlevnaden för hanar och ett för honor kan vi göra så här:

```
# main=”….” sätter en övergripande rubrik på figuren.

pie(antal[”Hane”,], main = ”Överlevnaden bland hanar”)

pie(antal[”Hona”,], main = ”Överlevnaden bland honor”)
```

Nu övergår vi till de numeriska variablerna. Rita ett histogram för variabeln HuvudNäbb:

```
hist(Grasparvar$HuvudNäbb)
```

Vi vill nu ändra klassindelningen till att omfatta klasserna 29.5-30.5, ..., 32.5-33.5. Vi kan lägga till klassindelningen till hist-kommandot så här:

```
# seq(a,b,c) betyder en sekvens från a till b i steg om c. Om man utelämnar c blir c=1.

hist(Grasparvar$HuvudNäbb, breaks = seq(29.5, 33.5, 1))
```

Rita ett spridningsdiagram som visar sambandet mellan Vikt och Längd med Längd på x-axlen och Vikt på y-axeln:

```
plot(Grasparvar$Längd, Grasparvar$Vikt)
```

Finns det något samband?

Beräkna beskrivande statistik för Längd, Vinglängd och Vikt. Uppställningen ska innehålla medelvärde (mean), standardavvikelse (sd), min (min), max (max) och antal observationer

som inte är NA. Eftersom Längd är angiven som heltal medan de två övriga som numeriska värden kan du inte beräkna medelvärde och standardavvikelse för alla tre variablerna på en gång som tidigare. Du kan använda colMeans eller hantera varje variabel för sig, t.ex. så här:

```
medelv<- c(mean(Grasparvar$Längd), mean(Grasparvar$Vinglängd), mean(Grasparvar$Vikt))

stdavv<- c(sd(Grasparvar$Längd), sd(Grasparvar$Vinglängd), sd(Grasparvar$Vikt))

mini <- c(min(Grasparvar$Längd), min(Grasparvar$Vinglängd), min(Grasparvar$Vikt))

maxi <- c(max(Grasparvar$Längd), max(Grasparvar$Vinglängd), max(Grasparvar$Vikt))

nobs<-c(sum(!is.na(Grasparvar$Längd)),  sum(!is.na(Grasparvar$Vinglängd)),

    sum(!is.na(Grasparvar$Vikt)))

# is.na(x) anger om x är NA, !is.na(x) är ”inte NA”, sum(!is.na(x)) räknar dem

# Sätt ihop dem till en tabell med:

cbind(medelv, stdavv, mini, maxi, nobs)
```

Antag att vi nu vill göra samma beräkningar som i föregående uppgift fast uppdelat på fåglarnas kön. I R kan man använda kommandot by(datamaterial, faktor, funktion) som beräknar funktionen för datamaterialet en gång per värde hos faktorn, t.ex. medelvärdena:

```
by(Grasparvar[,c(”Längd”)], Grasparvar$Kön, mean)
```

Gör samma sak med standardavvikelserna. Beräkna och minimum och maximum. Tänk på att då måste du göra varje variabel för sig. Dom andra variablerna kan du göra samma beräkningar på genom att byta ut ”längd” till en annan variabel i kommandot ovan.

Rita nu en boxplot på vinglängden där man delat upp på kön. Eftersom R anser att det lämpligaste sättet att rita en numerisk variabel uppdelat på en faktor är en boxplot kan vi helt enkelt skriva:

```
plot(Grasparvar$Vinglängd ~ Grasparvar$Kön)

# Detta funkar också fast då behöver vi själva ange namnen på axlarna.

boxplot(Grasparvar$Vinglängd ~ Grasparvar$Kön, xlab=”Kön”, ylab=”Vinglängd”)
```

Ibland vill man inte använda de numeriska värdena på en variabel utan man vill istället dela in individerna i t ex två grupper, under medel och över medel. Låt oss göra detta för variabeln Vinglängd. I tidigare uppgifter har vi räknat fram att medelvärdet på denna variabel är 245.04 för hela materialet. Vi vill nu dela in materialet i grupperna:

Grupp 1 ”Under medel” = 215-245.04
Grupp 2 ”Över medel” = 245.04-256

Om du har sparat medelvärdena för Längd, Vinglängd och Vikt i variabeln medelv kan du använda det som medelv[2], Värdena 215 och 256 är min och maxvärden från datamaterialet. Vi gör först en logisk variabel (Sant/Falskt = TRUE/FALSE) för om vinglängden är över medel. Sedan gör vi om den till en faktor där TRUE byts ut mot ”över medel” och FALSE mot ”under medel”:

```
Grasparvar$VLgrupp<-Grasparvar$Vinglängd>medelv[2]

Grasparvar$VLgrupp<-factor(Grasparvar$VLgrupp,

                levels=c(TRUE,FALSE),

                labels=c("över medel","under medel"))
```

Nu har vi en ny variabel (VLgrupp) i dataframen Grasparvar.

(b) Avsluta med att i en korstabell räkna ut hur stor andel av hanar respektive honor som ligger över medel.
Hanar: _____%
Honor: _____%

Svar på frågorna:

(a) Hanar 41.4% , Honor 57.1%

(b) Hanar 70.1% , Honor 20.4%

Sammanfattning R kommandon

datamaterial$variabel	en variabel i ett datamaterial
datamaterial[,”x”]	variabeln x i datamaterialet
datamaterial[1:3,c(”x”,”y”)]	rad 1-3 och variablerna x och y

c(1,2,5,3)	lista med talen 1, 2, 5 och 3
seq(a,b,c)	a till b i steg om c
summary(datamaterial)	sammanfattning
factor(variabel, labels=c(”en”,”två”))	gör faktor istället för tal
table(…)	Frekvenstabeller och korstabeller
prop.table(table(…), margin=…)	Relativa frekvenser
cbind(x,y)	sätt samman x och y till en tabell
mean(x)	medelvärdet av variabeln x
sd(…)	standardavvikelse
min(…)	minimum
max(…)	maximum
by(datamaterial, faktor, funktion)	funktion uppdelat på grupper
plot(…)	rita, R väljer diagramtyp
pie(table(…))	Cirkeldiagram
boxplot(…)	Box-plot
hist(…)	Histogram
plot(x,y)	Spridningsdiagram
plot(y ~ x)	Rita y som funktion av x
getwd(), setwd(…)	Om arbetskatalog

Appendix – Mer om R

Läsa in data från filer

För att spara sin data frame kan man antingen använda kommandot save() som sparar objektet i R:s eget format .Rdata eller write() som sparar det som en textfil vilken även kan läsas av andra format. (Glöm inte att först se till att du har rätt arbetskatalog t.ex. Mina dokument, använd vid behov **File > Change dir**.)

```
save(masb11, file="buller.Rdata")
```

```
write.table(masb11, file="buller.dat")
```

Självklart kan man också läsa en in data frame från en fil. Om filen är en textfil (.txt, .dat, .raw) använder man `read.table()`. Kommandot kan läsa filer som ser ganska olika ut så det är viktigt att ta veta hur just den filen är formaterad och anpassa argumenten till kommandot därefter. Skriv

```
args(read.table)
```

så listar R alla argument kommandot (funktionen) kan ta. Här går vi bara igenom de tre viktigaste. För att få information om alla argumenten är det bara att skriva `?read.table` eller `help(read.table)`.

`header` om variabelnamnen står överst ska `header=TRUE`

`sep` vilket tecken används för att skilja på olika värden? Vanliga alternativ är tabbar `sep="\t"`, komma `sep =","`, semikolon `sep=";"`eller bara ett blanksteg `sep=" "`

`dec` används punkt eller komma som decimaltecken? `dec="."` eller `dec=","`

För att pröva att läsa in en textfil ska du kopiera filen `Table 1_6.txt` från S:\R-filer till din arbetskatalog. Innan man börjar läsa in den kan det vara praktiskt att titta på hur den är formaterad. Välj **File > Display file(s)** och öppna filen. Vi noterar att filen innehåller variabelnamn, är tabbavgränsad och har komma som decimaltecken. Alltså använder vi följande kommando för att läsa in data mängden:

```
tabell1.6 <- read.table("Table 1_6.txt", header=TRUE, sep="\t",
    dec=",")
```

Använd summary eller edit för att controller att filen blev rätt inläst. Filen innehåller data om effekten av annonsering och kostnaden för densamma för ett antal företag (tabell 1.6 i Gujarati & Porter).

Funktionen lm()

Modellbeskrivningen använder vi även om vi vill göra en regressionsmodell. Vi kan spara modellen som ett eget objekt

```
min.modell <- IMPRESSION ~ ADEXP
```

Vi har nu skapat ett objekt av klassen *formula*.

```
class(min.modell)
```

Och sedan kan vi använda den för att skatta regressionen. Eftersom regressionsmodeller är en del av klassen av linjära modeller använder vi funktionen `lm()` (*linear model*).

```
lm(min.modell, data=tabell1.6)
```

Fast oftast är det mest praktiskt att även spara resultatet av analysen:

```
regr1 <- lm(min.modell, data=tabell1.6)
```

Då kan man nämligen fortsätta med flera kommandon och fördjupande analyser. `summary()` ger den viktigaste information och anova-tablån får man med `anova()`

```
summary(regr1)
anova(regr1)
```

Vi får fram residualerna med kommandot `resid()` och parameterskattningarna (koefficienterna) med `coef()`:

```
resid(regr1)
coef(regr1)
```

I båda fallen kan man givetvis spara dem som nya variabler. Om vi vill lägga till regressionslinjen på vår plot kan vi använda kommandot `abline()`:

```
abline(coef(regr1), col="blue")
```

ger en vackert blå linje. (Vill man hellre ha en annan färg så finns över 650 fördefinierade färgnamn att välja på. Skriv `colors()` så får du se och lägg märke till att både ”grey” och ”gray” accepteras.) Om man vill ändra linjetjockleken måste man ange argumentet `lwd` (*line width*):

```
abline(coef(regr1), col="forestgreen", lwd=5)
```

Oavsett linjetjockleken kan vi konstatera att regressionslinjen inte verkar passa så väl till data. Därför kan vi pröva att logaritmera våra variabler. Vi behöver därför en ny modell:

```
min.modell2 <- log(IMPRESSION) ~ log(ADEXP)
```

Notera att funktionen `log()` i R (precis som i nästan alla andra datorprogram) är den naturliga logaritmen $\log_e$. Om man av någon outgrundlig anledning behöver 10-logaritmen heter den `log10()`.

Med vår nya modell skattar vi en ny regressionslinje:

```
regr2 <- lm(min.modell2, data=tabell1.6)
```

Använd `summary()` och `anova()` för att se om den verkar vara bättre. Vi gör en ny plot också. För att öppna ett nytt fönster (*device*) att rita i använder vi kommandot `dev.new()`:

```
dev.new()
plot(min.modell2, data=tabell1.6, xlab="Advertising Expenditure",
   ylab="Impact of Advertising")
abline(coef(regr2), col="mediumvioletred", lwd=3)
```

Tjusigt, eller hur?

Som bekant från A-kursen bör man även undersöka om residualerna är normalfördelade. Det kan man göra på flera sätt. Vi gör det med ett normalfördelningsdiagram (*QQ plot*).

`qqnorm()` ger ett normalfördelningsdiagram men vill man även ha med referenslinjen får man dessutom använda kommandot `qqline()`.

```
dev.new()
qqnorm(resid(regr2))
qqline(resid(regr2), col="navyblue", lwd=1.5)
```

Ett alternativ är att rita upp den empiriska fördelningsfunktionen (*empirical disitribution function*) med kommandot `edf()`:

```
dev.new()
plot(ecdf(resid(regr2)))
xi <- seq(min(resid(regr2)), max(resid(regr2)), length.out=250)
points(xi, pnorm(xi, mean(resid(regr2)), sd(resid(regr2))),
    type="l", col="navyblue", lwd=1.5)
```

Läsa andra filformat

Om man har data i en Excel-fil är det enklast sättet att använda ”copy-paste”-metoden för att få in data. Kopiera filen `Table 1_6.xls` från S:\R-filer till din arbetskatalog. Öppna sedan filen i Excel. Markera data mängden inklusive variabelnamnen (men inte texten ovanför) och välj kopiera. Gå tillbaka till R och skriv

```
tabell1.6.2 <- read.table("clipboard", header=TRUE, dec=",")
```

så får du en ny kopia av samma datamängd.

R kan också läsa filer från en hel del andra statistikprogram som t.ex. SPSS, Minitab, Stata, SAS. Dessa funktioner finns i ett särskilt bibliotek (eller paket) som heter `foreign`. Detta bibliotek måste laddas separat med kommandot `library()`:

```
library(foreign)
```

eller välj **Packages > Load package** och markera sedan `foreign` och klicka OK.

Vilka funktioner har du nu fått tillgång till? Skriv

```
help(package="foreign")
```

så får du svaret. (Det räcker faktiskt att skriva `help(pa="foreign")` om man är lite slö.) Som synes heter kommandot för att läsa SPSS-filer inte helt oväntat read.spss(). Kopiera filen Table 1_6.sav från S:\R-filer till din arbetskatalog. Läs sedan in filen med kommandot

```
tabell1.6.3 <- read.spss("Table 1_6.sav", to.data.frame=TRUE)
```

Det är möjligt att du får ett varningsmeddelande men det kan man glatt ignorera. Om man inte anger argumentet `to.data.frame=TRUE` får man istället för en data frame en lista (*list*) som är en mer komplex datastruktur. Det finns tillfällen då detta kan vara bra, men för vårt vidkommande är data frame nästan alltid att föredra.

Bibliotek i R

Det finns väldigt många bibliotek eller paket till R. På hemsidan http://ftp.sunet.se/pub/lang/CRAN/ finns en lista över alla offentligt publicerade. Vissa innehåller ett stort antal funktioner medan andra bara har en eller två specialfunktioner. Det finns även paket som bara innehåller olika datamaterial eller färgskalor för att rita fina diagram. Har bibliotekets författare varit ambitiös har han/hon skrivit en lättillgänglig manual som kallas *vignette*. För att få reda på vilka paket (av dem som är installerade) som har vinjetter skriver man bara

```
vignette()
```

Vissa paket har också medföljande demos. För att få en liten demonstration av de grafiska möjligheterna i R skriver man

```
demo(graphics)
```

Vill man få reda på alla demos som finns (i de installerade paketen) skriver man bara

```
demo()
```