

R-programmering VT2022

Föreläsning 1

Johan Alenlöv

2022-01-24

Linköpings Universitet

Föreläsning 1:

- Introduktion till kursen
- R, R-studio
- Introduktion till R-programering
 - Miniräknare
 - Variabler
 - Vektorer
 - Hjälp
 - Funktioner
 - Logik

Föreläsare: - Johan Alenlöv

Labbassistent: - Rasmus Säfvenborg

Information om kursen finns i kursplanen.

Lärandemål

- skapa enkla program i programspråket R med hjälp av grundläggande programmeringstekniker som inläsning och utskrift av data, tilldelning och manipulation av datastrukturer, skriva egna funktioner, upprepningar och villkorsstyrda satser.

Vi sammanfattar detta till

- Bli bekväm med att använda R
- Hantera data med R
- Skriva program i R

Kollar man på tidigare kursutvärderingar har kursen fungerat bra.

1. Kursens ämnesinnehåll har gett mig möjlighet att uppnå kursens lärandemål. 4.09
2. Kursens examinerande moment har varit relevanta i relation till kursens lärandemål. 3.95
3. Vilket helhetsbetyg ger du kursen? 3.70

Ny föreläsare och examinator i år. Mindre förändringar kommer att ske i föreläsningar/laborationer. Inga planerade ändringar i innehåll eller upplägget av kursen.

Kursen består av två delar:

- Del 1: Grundläggande programmering
- Del 2: Tillämpningar relaterade till statistik, grafik och datahantering

Varje vecka

- En föreläsning
- En **obligatorisk** datorlaboration
- 4 timmar lärarledd laboration
- Inlämning:
 - Labbar: varje **onsdag kl. 18:00** via LISAM

Del 1: Grundläggande programering

- Grunderna i R
 - Lära sig hantera R-studio
- Fyra föreläsningar
- Fyra inlämningar (datorlaborationer)
- Labbarna görs **en och en**

Del 2: Tillämpningar

- Statistisk analys med R
- Fyra föreläsningar
- Fyra inlämningar
- Labbarna förs genom **parprogrammering** (grupper om två)
- **miniprojekt** i två delar (del av labb 5 och labb 7)

Jobba med materialet och skriv egen kod!

Kurslogistik

Hemsidan innehåller föreläsningar, labbar m.m.

LISAM används för inlämning av labbar och kompletteringar

Teams används för kommunikation

Programvara

I denna kurs använder vi **R** och **R-studio**

Kursboken

The Book Of R av Tilman M. Davies, 2016

Den finns som e-bok via biblioteket.

Artiklar

Dessa finns tillgängliga via kurshemsidan

- Dates and Times Made Easy with lubridate
- Handling and processing string in R
- Best practices for scientific computing

Videoföreläsningar

- Google Developers videomaterial
- Roger Pengs föreläsningar

Länkar finns på kurshemsidan

Reference cards:

Olika referenskort med funktionsnamn och hjälp finns på kurshemsidan.

- Datorlaborationer, 8st
- Datortentamen i datorsal
 - Hjälpmedel: R reference card (digitalt) + några fler. Information om vilka kommer komma på kurshemsidan. Dessa erhålls digitalt på tentamenstillfället, ni ska inte ta med er några papper.

- Börja direkt
- Är **obligatoriska**
- Ungefär 15 h arbete per vecka.
- Laborationsmall finns på hemsidan
- Laborationer lämnas in via LISAM
- Autorättning används på en del av uppgifterna, se till att följa instruktionerna.
- 100% rätt för att bli godkänd

- Arbetstakt:
 - Kursveckorna går måndag till måndag
 - Kursen går på halvfart ~20h/vecka. Ungefär 15h/vecka till labbar.
 - Mjuk deadline: söndag kväll
 - Hård deadline: onsdag kl 18:00 veckan efter
- Kompletteringar:
 - Labb 1-4: komplettering strax efter halva kursen. Se kurshemsidan för info.
 - Komplettering i samband med tentan och omtentor.
 - Hårda deadlines.

- Instruera en dator att utföra uppgifter
- Mjukvaruutveckling - vetenskaplig programmering
- Maskinkod - Lågnivåspråk - Högnivåspråk
- Kompilering - Interpreterande språk

Varför lära sig programera?

- Hantera **stora** datormängder
- Replikerbarhet
- Komplexa beräkningar
- Automatisera

Vad är R?

- R är ett populärt programmeringsspråk för statistiker
- Öppen källkod
- Många utvecklare
- Högnivåspråk

Ett exempel på ett program i R

Skapa ett program som skriver ut talen från 10 till 1 och sen skriver "kör!".

I R ser det ut på följande sätt

```
start <- 10
for (i in 1:10) {
  print(start)
  start <- start - 1
}
print("Kör!")
```

Kör vi koden i R får vi följande resultat

```
## [1] 10  
## [1] 9  
## [1] 8  
## [1] 7  
## [1] 6  
## [1] 5  
## [1] 4  
## [1] 3  
## [1] 2  
## [1] 1  
  
## [1] "Kör!"
```

- R är både ett program och ett programmeringsspråk
- R-Studio är en IDE för R
- Båda är gratis och går att ladda ner och installera på er egna dator.
Se kurshemsidan för information.

- Inbyggd hjälp i R
- Sök i Google
- Sök på **ENGELSKA**
- Kolla på felmeddelandet

```
## Error in eval(expr, envir, enclos) : object 'x' not found
```

Variabler och vektorer

- Variabler kan spara värden
 - Sätts med `<-` (eller `->`)
- Vektorer är en samling av likadana element
 - Skapas med `c()`
 - Välj element med `[]`

Exempel:

```
a <- 1
```

```
a
```

```
## [1] 1
```

```
testVektor <- c(2,3,5,7,11,13)
```

```
testVektor[c(1,3)]
```

```
## [1] 2 5
```

Räkna med vektorer

- Beräkningar sker elementvis

```
testVektor <- c(2,3,5,7,11,13)
testVektor+1
```

```
## [1] 3 4 6 8 12 14
```

- Beräkningar mellan vektorer sker cykliskt

```
testVektor <- c(2,3,5,7,11,13)
testVektor+c(1,2)
```

```
## [1] 3 5 6 9 12 15
```

Olika typer av värden

- Värden kan vara en av flera olika typer
 - t.ex. heltal, flyttal, textsträngar etc.
- Dessa typer kallas atomära klasser
- Kan kolla vilken typ det är med `typeof()`
- Kan konvertera med `as`.

```
as.character(4:8)
```

```
## [1] "4" "5" "6" "7" "8"
```

Beskrivning	Synonymer	<code>typeof()</code>	Exempel i R
Heltal (\mathbb{Z})	<code>int</code>	<code>integer</code>	-1, 0, 1
Reella tal (\mathbb{R})	<code>real</code> , <code>float</code>	<code>double</code>	1.03, -2.872
Komplexa tal (\mathbb{C})	<code>cplx</code>	<code>complex</code>	1 + 2i
Logiska värden	<code>boolean</code> , <code>bool</code>	<code>logical</code>	TRUE FALSE
Textsträngar	<code>string</code> , <code>char</code>	<code>character</code>	En textsträng

- En funktion utför något
- Tar noll eller flera [argument](#)
- Funktioner samlas i R-paket
- Många små funktioner, en funktion gör en sak.

En funktion i R är uppbyggd av

- ett funktionsnamn, t.ex. `area`
- en funktionsdefinition: `function()`
- 0 eller flera argument, t.ex. `hojd` och `bredd`
- “måsvingar” `{ }`
- kod, t.ex. `area <- hojd * bredd`
- returnera värde, t.ex. `return(area)`

Exempel på funktion i R

```
area <- function(hojd, bredd){  
  area <- hojd * bredd  
  return(area)  
}
```

```
area(hojd = 2, bredd = 3)
```

```
## [1] 6
```

```
area(hojd = 5, bredd = 11)
```

```
## [1] 55
```

“Det som sker i en funktion stannar i funktionen”

```
f <- function(x, y){  
  z <- 5  
  svar <- z*x + y  
  return(svar)  
}
```

z och svar kan **inte** användas utanför funktionen.

```
ls()
```

```
## [1] "a"          "area"        "f"           "i"           "star"
## [6] "testVektor"
```

```
f(1,2)
```

```
## [1] 7
```

```
ls()
```

```
## [1] "a"          "area"        "f"           "i"           "star"
## [6] "testVektor"
```

- Funktionen måste läsas in innan den fungerar.
- `return()` avslutar funktionen
- **Skriv funktionen i flera delar**
 - Skriv kod som gör det du vill
 - Lyft in koden i funktionen
 - Pröva funktionen

- Logik är vanligt i programmering
 - Används i `if`-satser
- I R finns de logiska värdena `TRUE`, `FALSE`, och `NA`
- Skapas på två olika sätt
 - Som vektorer
 - Genom relationsoperatorer
- Kan användas för att välja element i vektorer

Kan skapa en vektor med värdena TRUE och FALSE

```
testVektor <- c(2,3,5,7,11,13)
boolVektor <- c(TRUE, FALSE, FALSE, TRUE, FALSE, TRUE)
```

```
testVektor[boolVektor]
```

```
## [1] 2 7 13
```

Kan också skapa vektor genom en relation

```
testVektor > 5
```

```
## [1] FALSE FALSE FALSE TRUE TRUE TRUE
```


Relationsoperatorer

- Relationer används för att jämförelser
- Skapar logiska vektorer

Beskrivning	Operatorer i R
Lika med	==
Inte lika med	!=
Större än	>
Mindre än	<
Större än eller lika med	>=
Mindre än eller lika med	<=
Finns i	%in%

Logiska operatorer

- Boolsk algebra
- Operatorer:

Operator	Symbol	Operator i R
och	\wedge	$\&$
eller	\vee	$ $
inte	\neg	$!$

Symbol	A	B	$\neg A$	$A \wedge B$	$A \vee B$
i R	A	B	!A	A & B	A B
	TRUE	TRUE	FALSE	TRUE	TRUE
	TRUE	FALSE	FALSE	FALSE	TRUE
	FALSE	TRUE	TRUE	FALSE	TRUE
	FALSE	FALSE	TRUE	FALSE	FALSE

Logik exempel

```
testVektor <- c(2,3,5,7,11,13,17,19,23,29,31)
boolVektor <- testVektor < 6 | !(testVektor < 20)
```

Vad blir följande uttryck?

```
testVektor[boolVektor]
```

```
## [1]  2  3  5 23 29 31
```