

732G33/83 – R-programmering

Föreläsning 1

Josef Wilzén

Linköpings Universitet

Föreläsning 1:

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

Lärare:

- Josef Wilzén (examinator), ansvarig för del 1
- Johan Alenlöv, ansvarig för del 2

Labbassistenter:

- Algot Larsson Eskilsson
- Hampus Beijer
- Duy Thai Pham
- Joseph Mumper

Information om kursen finns i kursplanen: [länk](#)

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. Tillämpningar i statistik.

Information om kursen finns i kursplanen: [länk](#)

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. Tillämpningar i statistik.

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. Antal svar 2024: $5/51 = 9.8 \%$

1. Kursens ämnesinnehåll har gett mig möjlighet att uppnå kursens lärandemål: 4,60
2. Kursens examinerande moment har varit relevanta i relation till kursens lärandemål: 4,25
3. Vilket helhetsbetyg ger du kursen? 4,40

Förändringar till detta år:

- Mindre förändringar i föreläsningar, seminarier och datorlaborationer

Kursen består av två delar:

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

Kursen består av två delar:

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

Varje vecka

- Föreläsning
 - Nytt material och teorier
- 2 x laborationer
 - Jobba med uppgifter och inlämningar
- Seminarie från kursvecka 2
 - Koddemo, lösningar, svara på frågor
- Inlämningar:
 - Dels via Lisam, varannan söndag från kursvecka 3
 - Muntligt för lärare/assistent under laborationerna

Del 1: Grundläggande programering

- Grunderna i R
 - Lära sig hantera RStudio
- Fyra föreläsningar
- 2 inlämningar
- Labbarna görs **en och en**

Del 2: Tillämpningar

- Statistisk analys med R
- Fyra föreläsningar
- 2 inlämningar + **miniprojekt**
- Labbarna och miniprojektet förs genom **parprogrammering** (grupper om två)

Del 2: Tillämpningar

- Statistisk analys med R
- Fyra föreläsningar
- 2 inlämningar + **miniprojekt**
- Labbarna och miniprojektet förs genom **parprogrammering** (grupper om två)

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 **RStudio**

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.

- Inlämningar, 4st
- Miniprojekt
- 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
- Övningsuppgifter + inlämningsuppgifter
- Inlämningsuppgifterna 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.
 - Den ska visa helt rätt innan ni presenterar muntligt och laddar upp på Lisam.
 - Ta hjälp av labassistenter för att se till att detta sker.
- Muntlig redovisning där ni ska kunna förklara hur ni löst uppgiften
- 100% rätt för att bli godkänd

- Arbetstakt:
 - Kursveckorna går måndag till söndag
 - Kursen går på halvfart ~20h/vecka. Ungefär 15h/vecka till labbar.
 - Mjuk deadline: söndag kväll varannan vecka från vecka 2 (föreslagen arbetstakt)
 - Hård deadline: söndag kväll varannan vecka från vecka 3
- Kompletteringar:
 - Komplettering i samband med tentan och omtentor.
 - Möjlighet att redovisa lösningar då.

- Generativ AI = program som kan generera text, bilder, ljud etc baserat på input/fråga
- Kan vara ett väldigt bra verktyg
- **Förbjudet att använda generativ AI för inlämningar!**
- Använd det gärna för:
 - Felmeddelanden
 - Förstå koncept
 - Hitta buggar

- Ni har ansvar för er egen inlärnin
- Detta kräver eget arbete kontinuerligt under kursen
- Programmering:
 - Teoretisk färdighet
 - Praktisk färdighet
- Skriv mycket kod!
- Räcker inte att bara jobba under laborationerna!
 - Kolla och jobba med materialet innan ni kommer.
- Förslag på upplägg för en vecka:
 - Föreläsning (2 h), seminarium (2 h)
 - Datorlaborationerna (4 h), läs kurslitteratur/se videor (1 h)
 - Eget arbete med datorlaborationerna (11 h)
 - = 20 h studietid

Varför lära sig programera?

- Lösa problem

Varför lära sig programera?

- Lösa problem
- Hantera **stora** datormängder

Varför lära sig programera?

- Lösa problem
- Hantera **stora** datormängder
- Replikerbarhet

Varför lära sig programera?

- Lösa problem
- Hantera **stora** datormängder
- Replikerbarhet
- Komplexa beräkningar

Varför lära sig programera?

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

- Programmering handlar om att beskriva för en dator vad den ska göra
- Kräver ett programmeringsspråk
 - Finns många olika språk med svagheter och styrkor
- Exempel:
 - Python
 - Javascript
 - C
 - Java
- För statistik/dataanalys
 - R
 - Python
 - Julia
 - Matlab

Vad är R?

- R är ett populärt programmeringsspråk för statistiker/dataanalytiker

Vad är R?

- R är ett populärt programmeringsspråk för statistiker/dataanalytiker
- Öppen källkod

Vad är R?

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

Vad är R?

- R är ett populärt programmeringsspråk för statistiker/dataanalytiker
- Öppen källkod
- Många utvecklare
- Interpretande högnivåspråk

Ett exempel på ett program i R

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

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
- RStudio är en IDE för R
- Båda är gratis och går att ladda ner och installera på er egna dator.
Se kurshemsidan för information.

Demo: RStudio

- Datorlaborationerna sker i SU-salarna i B-huset
- Linuxdatorer
- Om det är ledigt är det bara att använda datorerna för självstudier.
- Går också bra att använda PC1-5 i E-huset

Hur kommer man igång i datorsalarna

1. Logga in med Liu-ID och lösenord
2. Öppna en terminal
 - Tryck `ctrl+alt+T`
 - Eller högerklicka på skrivbordet och välj `open terminal here`
3. Skriv `module add courses/732G33` i terminalen och tryck enter
 - Gör så att ni får tillgång till all programvara som behövs i kursen
4. Skriv `rstudio` i terminalen och tryck enter

- Inbyggd hjälp i R
- Sök i Google / Generativ AI
- 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"
```


Olika typer av värden

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

Demo: Variabler

- En funktion utför något → har en specifik uppgift
- 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
```

Demo: Funktioner

“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

Demo: Funktioner II

- R-paket för att rätta uppgifter
- Används i kursen för en första koll om ni har gjort rätt
- Ska visa alla rätt innan ni reodivisar och lämnar in
 - En inlämning som inte ger alla rätt kommer inte godkännas!
 - Följ instruktionerna noggrant så blir det mycket lättare.

Demo: markmyassignment

- 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 vanliga 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]
```

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 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	$!$

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

Demo: Logik

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]
```

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
```