Föreläsning 1

Programering i R

transition: rotate ## Föreläsning 1: Introduktion ### Josef Wilzén STIMA, Linköpings universitet

2021-01-21

Föreläsning 1: Innehåll

incremental: true - Upprop - Kursvärdering: 732G45 “Grundläggande statistik och dataanalys” - Kursens upplägg och praktisk information - R, R-Studio och vetenskaplig programmering - Introduktion till programering i R - miniräknare,variabler, vektorer, hjälp, funktioner, logik

Kursvärdering: 732G45

Gäller programstudenter på “Statistik och dataanalys” - 732G45
“Grundläggande statistik och dataanalys” - Svara på din dator/mobil
- Studentmail: sök på 732G45, kursvärdering - 5 min

Kursens upplägg och praktisk information

type: section

Lärare

Föreläsare:

- Josef Wilzén

Labbassar:

- Andrea Dahl Sturedahl, Faton Rekathati, Hugo Knape och Sidney Rydström

Kursens mål

incremental: true

Information om kursen finns i **kursplanen**. Denna kan sammanfattas till:

- Mål 0. Bli bekväma med datorer
- Mål 1. Tänka som en programmerare. Ställa upp problemet rätt. Struktur.
- Mål 2. Skriva programkod.
- Mål 3. Bli bekväma med R. Förstå hur du kan använda R i ditt framtida statistiska arbete.
- Mål 4. Ha kul!

Kursens delar

incremental: true ### Kursen består av två delar: - *Del 1:* Grundläggande programering - *Del 2:* Tillämpningar relaterade till statistik, grafik och datahantering

Varje vecka (block):

- En föreläsning
- En ~~obligatorisk~~ datorlaboration
- 4 timmar lärarledd laboration + **många timmar eget arbete**
- Inlämning:
  - Labbar: **varje onsdag kl. 18.00** via LISAM

Del 1: Grundläggande programering

incremental: false

- Grunderna för programmering i R
- Fyra föreläsningar
- Fyra labbar
- Labbarna görs **en och en**

Del 2: Tillämpningar

incremental: false

- Statistisk analys med R
- Fyra föreläsningar
- Fyra labbar
- Labbarna görs genom **parprogrammering**
- **Miniprojekt** i två delar (del av labb 5 och labb 7)

Kursens viktigaste moment:

~~Eget arbete med labbarna!~~

Praktisk information

Kurslogistik

Hemsidan innehåller bland annat schema, föreläsningar m.m. och finns [här]. Kursen har också en github-sida [här]
LISAM: används för inlämning av labbar och kompletteringar.
Teams: används för kommunikation
Kurskod: **732G33**, **732G83**

Programvara

För denna kurs krävs R och R-Studio.

Kurslitteratur I:

Kursboken

Kursboken *A first course in statistical programming with R* av Braun och Murdoch, första eller andra upplagan.

Artiklar

- *Dates and Times Made Easy with* `lubridate` [ladda ned]
- *Handling and processing strings in R* [ladda ned]
- *Best practices for scientific computing* [ladda ned]

Kurslitteratur II:

Videoföreläsningar

- Google Developers (GD) videomaterial [här]
- Roger Pengs föreläsningar (RP) [här]
- Finns även andra på kurshemsidan

Reference cards:

Ett par sidor med viktiga funktioner i R. [ladda ned]
Hjälp till RStudio [ladda ned]
Det kommer flera reference cards under kursens gång.

Examination

- Datorlabbar, 8 st
- Datortentamen i datorsal
  - *Hjälpmedel:* R reference card v.2 (digitalt) + några till reference cards (exakt vad ni får använda finns på hemsidan) Dessa erhålls digitalt på tentan.

Datorlaborationer

- Börja direkt!
- Är ~~obligatoriska~~.
- $\sim$ 15 h arbete/v
- Laborationsmall finns på hemsidan, kolla här
- Labbarna lämnas in via LISAM och autorättas
  - R-paketet `markmyassignment`
  - ~~labbarna måste bli godkända av markmyassignment~~
- Filer finns på kurshemsidan
- 100 % rätt för godkänt
- Vissa labbuppgifter är inspirerade av linjär algebra

Datorlaborationer

- Registrera er på kursen för att få tillgång till kursrummet på LISAM
- Arbetstakt:
  - kursveckorna går måndag till måndag
  - Kursen går på halvfart $\sim$ 20 h/vecka, ca 15 h/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 LISAM)
  - Komplettering i samband med tentan och omtentor
    - Detta är hårda deadlines!

markmyassignment

- Paket för direkt återkoppling på labbar
- Underlätta lärandet
- **Kom gärna med synpunkter!**

Datorn, programering och R

type: section

R och datorn

- En bra minnesregel är:
  - Hårddisk = I bokhyllan
  - Internminne (RAM) = På skrivbordet

R arbetar på i internminnet (skrivbordet)

Vad är programering?

incremental: true - Instruera en dator att utföra uppgifter - Mjukvaruutveckling / vetenskaplig programmering - Maskinkod / Lågnivåspråk / Högnivåspråk - Kompilerande / Intepreterande språk

Varför programera?

incremental: true - Hantera (stora och komplexa) data - Replikerbarhet i analyser - Utföra komplexa beräkningar - Utföra och automatisera (tråkiga) rutinuppgifter - Effektivt användande av tid

Vad är R?

- Populäraste programmeringsspråket för statistiker
  - se här och här
- Bygger på öppen källkod (gratis och transparent)
- Många utvecklare - stor funktionalitet
- Finns för Windows, Mac och Linux
- Högnivåspråk (enklare)
- Mer information i RP:s video “Overview and Background”
- Ett “klister” för statistiska analyser

Ett exempel på ett program

incremental: true Skapa ett program som utgår från talet 7 och skriver sedan de följande 3 udda talen på skärmen.
Programmet ska avslutas med meddelandet 'Klar!'.

I R ser det ut på följande sätt

```
talet <- 7
for (i in 1:3) {
  talet <- talet + 2
  print(talet)
}
print("Klar!")
```

Resultatet

Kör vi koden i R får vi följande resultat:

```
## [1] 9
## [1] 11
## [1] 13

## [1] "Klart!"
```

Introduktion till programering i R

type: section

Installation av R och R-Studio

R

1. Gå till https://cran.r-project.org/
2. Välj ditt operativsystem, ladda ned och installera R

R-Studio

1. Gå till http://www.rstudio.com/
2. Download > Download RStudio Desktop
3. Välj ditt operativsystem, ladda ned och installera R-Studio

Se kurshemsidan för mer info.

Demo: R och R-Studio

type: section

Variabler (objekt) och vektorer i R

- "spara" värden
- Vektorer är 1+ element av samma typ (ex. heltal). jmf. linjär algebra.
- Skapas (enklast) med `c()`
- Indexering/slicing med `[ ]` Exempel:

```
## [1] 20
```

```
## [1]  7 23
```

Vektoraritmetik

- Sker elementvis

```
## [1] 14.0 30.0 40.0 46.0  2.4
```

- Recycling

```
## Warning in y + x: longer object length is not a multiple
## length
```

```
## [1] 12  6  8
```

Att söka hjälp

incremental: true - Hjälpen i R - Google, Google, Google - Stack overflow - ~~Sök på engelska~~ - Sök / studera (slutet) på felmeddelandet från R

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

- Mer information i RP:s video “How to get Help”

Variabeltyper

type: section

Variabeltyper

- Olika typer av värden på variabler, ex:
  - Heltal, Numeriska värden, Textsträngar
- Kallas för atomära klasser
- För att undersöka vilken klass en variabel har används: `typeof()`
- För att konvertera mellan klasser används `as.` - funktioner (finns för alla variabeltyper). Ex:

```
## [1] "1" "2" "3" "4" "5"
```

Variabeltyper, översikt

Beskrivning	Synonymer	`typeof()`	Exempe
Heltal ($\mathbb{Z}$)	`int`	`integer`	`-1, 0,`
Reella tal ($\mathbb{R}$)	`real`, `float`	`double` (`numeric`)	`1.03,`
Komplexa tal ($\mathbb{C}$)	`cplx`	`complex`	`1 + 2i`
Logiska värden	`boolean`, `bool`, `logi`	`logical`	`TRUE F`
Textsträngar	`string`, `char`	`character`	`Go R!`

Demo: Variabler och vektorer

type: section

Introduktion till funktioner i R

type: section

Introduktion till funktioner

incremental: true

- "Allt som existerar är objekt, allt som sker är funktioner"
- Tar noll eller flera **argument**
- Flera funktioner samlas i R-paket
- Många och små funktioner är det bästa!
  - Gör en sak och gör det bra!
- Kod som vi vill använda flera gånger
- Svårare att felsöka än vanlig kod.
- Grunden för bra och säker kod. Modularisering. Testbarhet (läs markmyassignemnt).

Funktioners struktur

En funktion i R består (ofta) av: - ett funktionsnamn (ex. `f`) - en funktionsdefinition: `function()` - 0 eller flera argument (ex. `x` och `y`) - “curly bracers” `{}` - programkod / funktionen (ex. `res <- x + y`) - returnera värde (ex. `return(res)`)

Exempel på funktion

incremental: true

```
f <- function(x, y){
  res <- x + y
  return(res)
}
```

```
f(x = 3, y = 2)
```

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

```
f(x = 1, y = 11)
```

```
## [1] 12
```

Lokal miljö

- "Det som sker i funktionen, stannar i funktionen"

```
f <- function(x, y){
  z <- 1
  res <- z+x+y
  return(res)
}
```

Lokal miljö II

incremental: true

```
ls()
```

```
## [1] "f"         "i"         "talet"     "testScore" "x"
```

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

```
## [1] 4
```

```
ls()
```

```
## [1] "f"         "i"         "talet"     "testScore" "x"
```

Att tänka på med funktioner

- Funktionen måste "köras/läsas in" innan den fungerar
- `return()` avslutar funktionen
- ~~Skriv funktionen i flera delar~~
  - Börja med kod som fungerar (men med funktionens argument)
  - Lyft in koden i funktionen
  - Pröva funktionen

Demo: Introduktion till funktioner och markmyassignment

type: section

Logik

type: section

Kort om logik

incremental: true - Logik vanligt i programmering + Kallas ibland *booleans* + Används i if-else + I R: Användas för indexering (välja ut rader/kolumner) - I R: De logiska värdena: `TRUE`, `FALSE`, `NA` (`NA` betyder saknade värden och används i olika sammanhang) - I R: Skapas på två sätt: + Som vanliga vektorer + Med relationsoperatorer

Kort om logik II

```
## [1] 12  7
```

Logiska operatorer

- Boolsk algebra
- Operatorer:

Operator	Symbol	Operator i R
och (and)	$\wedge$	&
eller (or)	$\vee$	|
icke (not)	$\neg$	!

Logiska operatorer (forts.)

incremental: true

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

- Kan kombineras för mer komplicerade uttryck:

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

Relationsoperatorer

- Jämförelser
- I *R reference card v.2* under operatorer
- 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%`

Relationsoperatorer (Exempel)

incremental: true

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

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

Kort om logik III - Allt på en gång

incremental: true

```
## [1] 12  7
```

Demo: Logik

type: section

Demo: Datorlaborationen/markmyassignment

type: section