R-programmering VT2024

Föreläsning 6

Johan Alenlöv

Linköpings Universitet

Innehåll föreläsning 6

- Information
- Programmeringsparadigm
  - Funktionell programmering
  - Objektorientering
- Datum och tid med `lubridate`
- Linjär algebra i R

Information

Information

- Kom ihåg att anmäla er till datortentan
  - Endast anmälda får skriva tentan
- Datortentan kommer att vara på plats
  - Tenta i SU-sal
  - Skriva R-kod (funktioner likt inlämningar)
    - Lämnas in som .R fil
  - Hjälpmedel finns på Lisam

Programmeringsparadigm

Programmeringsparadigm

- Finns många olika sätt att se på **hur** program ska skrivas.
- Olika programmeringsspråk följer olika paradigmer.
- Pratar ofta om två för R:
  - Funktionell programmering
  - Objektorienterad programmering

Funktionell programmering

- I funktionell programmering pratar man om två saker.
  - "Rena" funktioner.
    - Output beror bara på inargumenten.
    - Funktionen har inga sidoeffekter.
  - "First-class functions".
    - Funktioner ska bete sig som alla andra objekt
    - Kunna spara funktioner, returnera funktioner etc.
- Exempel:
  - Scala, Erland, Haskell

Objektorientering

- Objektorientering åtgår från klasser:
  - En klass är en mall som används för att skapa objekt
  - Kopplar samman objekt med funktioner
  - Objekt är en specifik realisering av en klass
  - Objekt skapas med en konstruktor
- Viktiga koncept:
  - **Inkapsling**: Gömma och ordna relaterad data.
  - **Polymorfism**: Generiska funktioner som hanterar olika objekt.
  - **Arv**: Underlättar specialisering av data och metoder via underklasser.

Objektorientering

- i R:
  - Alla objekt ha en klass
  - Kan undersöka ett objekts klass med `class( )`
  - Variabeltyper är "atomära" klasser
  - Klasser har en konstruktor
- Olika typer av klasser i R:
  - Basklasser
  - S3 (informellt): "lista med klassattribut"
  - S4 (formellt): element väljs ut med @
  - Reference classes

Generiska funktioner (S3)

- Objektorientering i R utgår från generiska funktioner
- Funktioner som gör olika saker beroende på objektets klass
- Ex:
  - `plot( )`
  - `mean( )`
  - `summary( )`
  - `names( )`
- Funktionerna anropar metoden för objektet när de används.

Metoder

- En funktion för en specifik klass
- Metoderna utgår från generisk funktion
- Alla metoder för en generisk funktion hittas med `methods( )`
- T.ex. funktionen `mean( )`
  - Defaultmetoden finns med `mean.default( )`
  - Specifik med `mean.difftime( )`
- Metoder kan också anropas direkt som vanliga funktioner.

Demo

Demo: Objektorientering

Datum och tid

Datum och tid

- Datum är klurigt att arbeta med, men används extremt mycket.
- Två typer av tid:
  - Relativ tid
  - Exakt tid
- Enklare funktioner för datum finns i `base`

Datum och tid - Exempel

```
my_date <- "2024-05-06"
class(my_date)
```

```
## [1] "character"
```

```
my_date_as_date <- as.Date(my_date)
my_date_as_date
```

```
## [1] "2024-05-06"
```

```
class(my_date_as_date)
```

```
## [1] "Date"
```

lubridate

- Paket för enkel och effektiv datumhantering
- Sammansatt av “lubricant” och “date”

Tre huvudsakliga delar:

1. Läsa in datum
2. Ändra inlästa datum
3. Göra beräkningar med datum

lubridate - Exempel

```
library(lubridate)
idag <- ymd("2024-05-06")
print(idag)
```

```
## [1] "2024-05-06"
```

```
week(idag)
```

```
## [1] 19
```

```
idag + weeks(2)
```

```
## [1] "2024-05-20"
```

lubridate

Elementordning	Funktion
år, månad, dag	`ymd()`
år, dag, månad	`ydm()`
månad, dag, år	`mdy()`
timme, minut	`hm()`
timme, minut, sekund	`hms()`
år, mån, dag, timme, min, sek	`ymd_hms()`

Källa: Grolemund and Wickham (2011, Table 4)

lubridate

För att "plocka ut" eller ändra delar av ett datum används följande funktioner

Datum	Funktion	Tidsdel	Funktion
år	`year()`	timme	`hour()`
månad	`month()`	minut	`minute()`
vecka	`week()`	sekund	`second()`
årsdag	`yday()`	tidszon	`tz()`
månadsdag	`mday()`		
veckodag	`wday()`		

Källa: Grolemund and Wickham (2011, Table 5)

lubridate - Exempel

```
idag <- ymd("2024-05-06")
week(idag)
```

```
## [1] 19
```

```
wday(idag, label = TRUE)
```

```
## [1] Mon
## Levels: Sun < Mon < Tue < Wed < Thu < Fri < Sat
```

```
year(idag) <- 2019
idag
```

```
## [1] "2019-05-06"
```

lubridate

- För att räkna med datum finns det fyra olika objekt i `lubridate`
  - `instant`
  - `interval`
  - `duration`
  - `period`

instant och interval

Instant

- Ett spcifikt tillfälle i tiden
- Viktiga funktioner:
  - `now( )`
  - `today( )`

Interval

- Tidsspannet mellan två `instant`
- `interval(start,end)`

instant och interval - Exempel

```
inst_1 <- ymd("2020-02-29")
inst_2 <- today()
my_interval <- interval(start = inst_1, end = inst_2)
my_interval
```

```
## [1] 2020-02-29 UTC--2024-05-05 UTC
```

duration

- Ett fixt tidsspann som mäts i sekunder
- Tänk kontinuerlig tid
- Absolut tid i sekunder
- Funktioner börjar med `d`
- Konvertera ett interval med `as.duration( )`
- Ex:
  - `duration( )`
  - `dseconds( )`
  - `dhours( )`

duration - Exempel

```
my_interval / ddays(30)
```

```
## [1] 50.9
```

```
as.duration(my_interval)
```

```
## [1] "131932800s (~4.18 years)"
```

Period

- Utgår från den aktuella enheten (dagar, månader, år)
- Tänk diskret tid
- Relativ tid
- Vad vi i dagligt tal menar med ex. två veckor
- Ex:
  - `period(num = , units = )`
  - `minutes( )`
  - `hours( )`
  - `weeks( )`

Period - Exempel

```
months(3) + weeks(2) + days(1)
```

```
## [1] "3m 15d 0H 0M 0S"
```

```
my_interval %/% months(4)
```

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

```
as.period(my_interval)
```

```
## [1] "4y 2m 6d 0H 0M 0S"
```

Att räknamed tid - Exempel

```
course_start <- ymd("2022-01-24")
course_start + weeks(5)
```

```
## [1] "2022-02-28"
```

```
course_start + months(1)
```

```
## [1] "2022-02-24"
```

```
course_start + ddays(30)
```

```
## [1] "2022-02-23"
```

Demo

Demo: Datum och Tid

Linjär algebra

Matriser

- Matriser är två-dimensionella vektorer
- Har jobbat med matriser förut.
- Skapa en matris med
  - `matrix( data = , nrow = , ncol = , byrow = )`
  - Kom ihåg cirkulering
  - `byrow = FALSE` per deafult
- Transponat fås med `t( )`
- Diagonalmatris med `diag( vektor )`
- Enhetsmatrisen med `diag( nummer )`

Linjär algebra

- Matrismultiplikation görs med `%*%`
- Invertera en matris med `solve( )`
  - Notera att `A %*% solve(A)` blir en enhetsmatris
- Lösa ekvationssystem $Ax = b$ med `solve(a=A,b=b)`
  - Notera att om A inverterbar är svaret $x = A^{-1}b$.
- Egenvärden och egenvektorer med `eigen( )`
  - Returnerar en lista med egenvärden och egenvektorer
- Summera rader eler kolumner med `rowSums( )` och `colSums( )`
- Kombinera matriser med `rbind( , )` eller `cbind( , )`

Linjär algebra - paketet Matrix

- Matrix är ett paket som implementerar effektivare/snabbare funktioner för linjär algebra.
- Grunden är funktionen `Matrix( )`, vilket är en konstruktor som skapar matriser.
- Matriserna i **Matrix** är S4 klasser.
- I **Matrix** är det skillnad på glesa och täta matriser.
- Kolla dokumentation för mer information.
- Rekomenderas för avancerade tillämpningar.

Demo

Demo: Linjär Algebra