

Föreläsning 5

Josef Wilzén

2023-02-20

Föreläsning 5: Innehåll

- ▶ Praktisk information om del II.
- ▶ Basgrafik
- ▶ Slumptal och simulering
- ▶ knitr och markdown
- ▶ Externa data och pxweb

R-programmering - del II

- ▶ Del 1: Grunderna i programmering
 - ▶ Variabler, tilldelning
 - ▶ Datastrukturer
 - ▶ Kontrollstrukturer
 - ▶ Funktioner
 - ▶ Debugging
- ▶ Del 2: Tillämpningar

R-programmering - del II

- ▶ Del 1: Grunderna i programmering
- ▶ Del 2: Tillämpningar:
 - ▶ Grafik
 - ▶ Statistik och analys
 - ▶ Externa data
 - ▶ Datum
 - ▶ Texthantering och regular expression
 - ▶ Linjär algebra
 - ▶ knitr, markdown, Literate programming

R-programmering - del II:

- ▶ Labbarna görs nu i par
- ▶ Parprogrammering:
 - ▶ **Turas om att skriva koden**
 - ▶ Den som inte kodar är engagerad i koden och problemet
 - ▶ Byter person var 20/30:e minut
 - ▶ Viktigt att kommentera koden: ROxygen, inline
 - ▶ **Båda** är delaktiga i programmeringen

Projekt del 1

Info här: *[länk](#)*

- ▶ Hitta data på webben → pxweb
 - ▶ Kommunal (tvärsnitt) data
 - ▶ Tidsseriedata
- ▶ Presentera med basgrafik, knitr och markdown
- ▶ Ska kunna vara reproducerbart
- ▶ Lämna in **PDF** och **.Rmd**
- ▶ Egen inlämning på Lisam
- ▶ Två veckors tid

Basgrafik

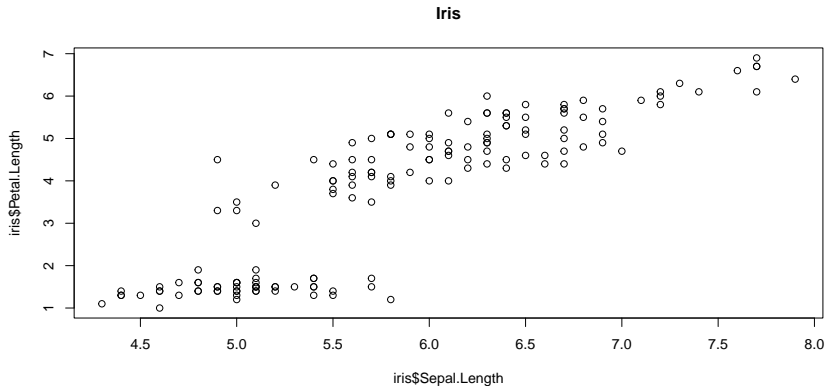
- ▶ Grafiken är en av styrkorna med R
- ▶ Olika paket: ggplot2, lattice m.m.
- ▶ Högnivå (funktioner, plot)
- ▶ Lågnivå (bygga upp en plot steg för steg)
- ▶ Används ofta för att snabbt visualisera data
- ▶ Använd grafik för att:
 - ▶ Sammanfatta tabeller/data visuellt
 - ▶ Jämföra olika dataset
 - ▶ Rita ut funktioner

Enkel grafik: `plot()`

- ▶ `plot()` - kan plotta många olika objekt:
 - ▶ `plot(x,y)`, ger en scatterplot, här är `x` och `y` vektorer som ger `x`- och `y`-koordinater
 - ▶ `plot(X)`, om `X` en `data.frame` så skapas en matrix-plot
 - ▶ Vanliga argument: `"type="`, `"main="`,
`"xlab="`, `"ylab="`, `"xlim="`, `"ylim="`, `"col="`

Exempel: plot()

```
data(iris)
plot(x = iris$Sepal.Length,
     y = iris$Petal.Length, main = "Iris")
```

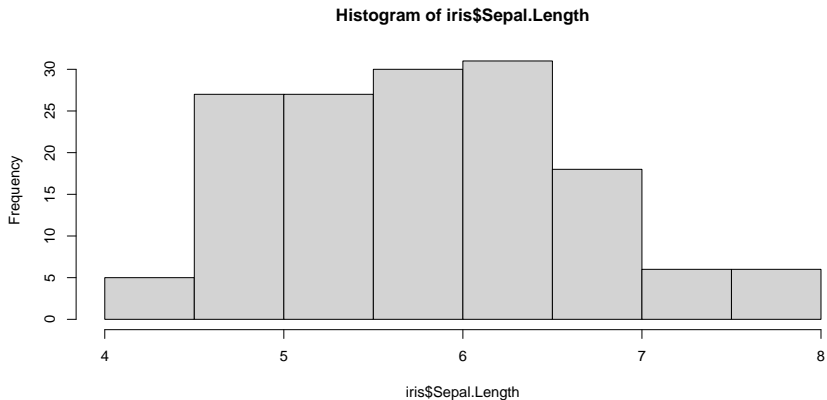


Olika diagramtyper

- ▶ `hist(x=,breaks=,freq=)` ger ett histogram,
 - ▶ "x"- numerisk vektor
 - ▶ "breaks=" - antal bins, default är att variationsområdet delas in i $\log_2(n) + 1$ intervall
- ▶ `boxplot()` ger boxplot
- ▶ `barplot()` ger stapeldiagram
- ▶ `pie()` ger en piechart
- ▶ Använd hjälpen för att se exempel och fler argument

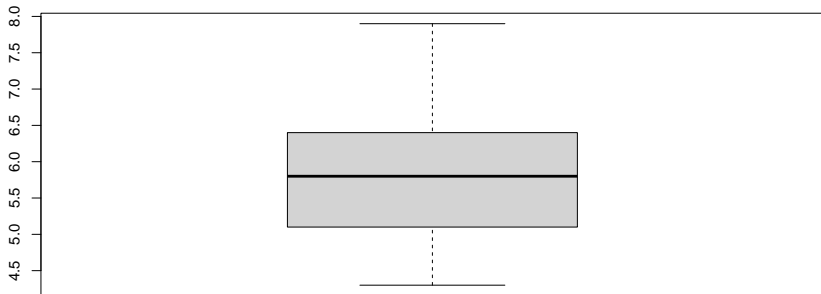
Exempel: hist()

```
hist(iris$Sepal.Length)
```



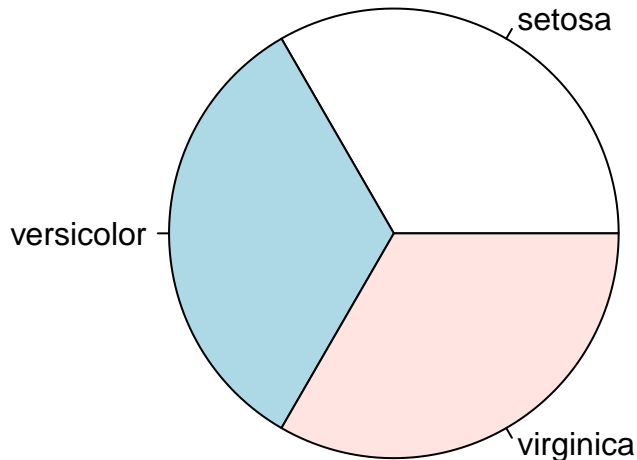
Exempel: boxplot()

```
boxplot(iris$Sepal.Length)
```



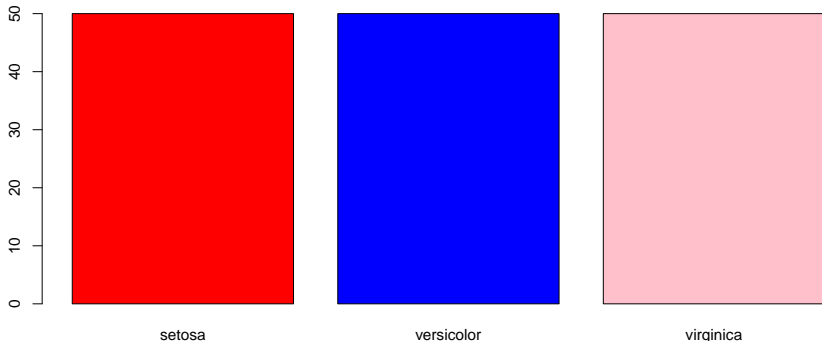
Exempel: pie()

```
tab <- table(iris$Species)  
pie(tab)
```



Exempel: barplot()

```
tab <- table(iris$Species)  # skapa frekvenstabell  
barplot(tab,col=c("red","blue","pink"))
```



Lågnivågrafik

- ▶ Kan användas för att bygga upp en graf från grunden
- ▶ Lägger till “lager på lager”:
 - ▶ `points(x, y, ...)` lägger till punkter
 - ▶ `lines(x, y)` lägger till linjer
 - ▶ `abline(a, b, h, v, ...)` lägger till räta linjer
 - ▶ `legend(x, y, legend, ...)` lägger till en förklaringsruta
 - ▶ `par()`: Fler grafiska alternativ

Lågnivågrafik

- ▶ Går att bygga upp många olika sorters plottar
- ▶ Dock är ggplot2 oftast **MYCKET** smidigare

Demo: Basgrafik

Slumtjal och simulering

- ▶ I R finns en uppsättning funktioner för fördelningar

prefix	Beskrivning	Exempel
r	simulera från fördelningen	<code>rnorm()</code>
d	täthetsfunktionen (pdf)	<code>dnorm()</code>
p	kumulativ fördelningsfunktion (cdf)	<code>pnorm()</code>
q	inversa kumulativa fördelningsfunktionen	<code>qnorm()</code>

- ▶ `?Distributions`
- ▶ Se CRAN task view: Distributions *här*

Slumtäl och simulering: Urval och slumpfrön

- ▶ Det finns ingen riktig slump i datorer
- ▶ Det finns slumtälsgeneratorer
- ▶ Kan styra “slumpen” med “slumpfrön” (`set.seed()`)
- ▶ För att dra ett (obundet) slumpmässigt urval använder vi `sample()`

Slumtjal och simulering: Exempel

```
rmnorm(n = 3, mean = 10, sd = 1)
```

```
## [1] 10.074627 9.504487 9.285568
```

```
set.seed(20180218)
```

```
rmnorm(n = 3, mean = 10, sd = 1)
```

```
## [1] 10.19380 10.74998 11.32583
```

```
set.seed(20180218)
```

```
rmnorm(n=3, mean=10, sd=1)
```

```
## [1] 10.19380 10.74998 11.32583
```

Slumtjal och simulering: Exempel sample()

```
text <- c("Linköpings", "Universitet")  
set.seed(20180218)  
sample(x=text, size=3, replace=TRUE)
```

```
## [1] "Universitet" "Universitet" "Universitet"
```

```
set.seed(654)  
sample(x=text, size=3, replace=TRUE)
```

```
## [1] "Universitet" "Linköpings"  "Universitet"
```

Demo: Slumptal

R-Markdown, Notebooks och knitr

- ▶ Kombinera text, kod och grafik i en fil.
- ▶ Förenkla för era laborationer i andra kurser
- ▶ Del av R-Studio
- ▶ Två delar:
 - ▶ Markdown (för text)
 - ▶ knitr (för R-kod)
- ▶ Kan skapa PDF, Word och/eller HTML (och presentationer)

Markdown

- ▶ Markupspråk (som HTML)
- ▶ Ersätter ofta HTML
- ▶ Enkelt
- ▶ Kan hantera matematik (med LaTeX)
- ▶ Integrerat med R-Studio
 - ▶ Kan behöva installera LaTeX för att skapa PDF

knitr

- ▶ Kör R-kod och ersätter med resultatet
- ▶ “Stickar ihop” text och kod
- ▶ “Dynamiska rapporter”
- ▶ För att skapa tabeller: `kable()`
- ▶ Kan hantera R och Python (och ev. SAS)

R-Studio Notebooks

- ▶ Är Rmd-filer
- ▶ Kan köras interaktivt direkt i R-Studio
- ▶ Inspiration från Jupyter Notebooks
- ▶ Skillnad mellan Markdown och Notebooks, läs *[här](#)*

Demo: Markdown och knitr

(Webbaserade) externa datakällor

Varför?

- ▶ Mer och mer data finns på webben
- ▶ Vill hantera detta programatiskt
- ▶ Kan vara lite klurigt i början
- ▶ Centralt för reproducerbarhet

Ladda ned och läsa in från webben

- ▶ Vill vi bara ladda ned: `download`
- ▶ Vill vi ladda in direkt i R: `repmis`

`repmis` hanterar: - `.Rdata` - `.csv` - `.txt`

Vanliga källor att använda

- ▶ Dropbox
- ▶ Google Docs
- ▶ Github
- ▶ etc. . .

Öppen data med pxweb api

- ▶ En “dörr” till pxweb apier
- ▶ pxweb är ett paket för att “gå in” genom dörren
- ▶ Fler och fler myndigheter använder pxweb api
- ▶ Kan användas för:
 - ▶ Navigera i datalager
 - ▶ Ladda ned förutbestämd data med kod (för produktion)


```
install.packages("pxweb")
```

```
library(pxweb)
```

```
mitt_data<-pxweb_interactive()
```

Se även denna *vignette*

Demo: pxweb