R-anvisningar till *Grundläggande statistik*

2023-09-20

Contents

Introduktion

Detta dokument är en kort introduktion till R för en kurs i grundläggande statistik.

Installation av R och RStudio

0.1 Inledning

För att köra R-kod på sin dator krävs en installation av programspråket R. För att effektivt arbeta i R används ofta en utvecklingsmiljö (ett tilläggsprogram som på flera sätt förenklar arbetet) och här ges anvisningar till RStudio - den vanligaste utvecklingsmiljön för R. För att komma ingång måste man alltså installera R och RStudio.

0.2 Installation av R

Programspråket R kan laddas ner från https://www.r-project.org/ med följande steg:

1. Klicka på *CRAN* längst upp till vänster.
2. Klicka på den översta länken under 0-Cloud.
3. Välj en nedladdning beroende på operativsystem.
4. För Windows, välj *base*. För macOS, välj den senaste tillgängliga versionen.
5. Installera R från den nedladdade filen. Installation sker som för andra nedladdade program.

0.3 Installation av RStudio

RStudio kan laddas ner från https://www.posit.co/ med följande steg:

1. Klicka på *Download RStudio* uppe till höger.
2. Scrolla nedåt och välj *Download* under *Install RStudio*.
3. Installera RStudio från den nedladdade filen. Installation sker som för andra nedladdade program.

0.4 Gränssnittet i RStudio

När man nu öppnar RStudio ser man att fönstret är uppdelat i fyra delar och att varje del består av en eller flera flikar. De viktigaste är i nuläget

- *Console* där kod körs och resultat skrivs ut,
- *Environment* där man ser skapade objekt,
- *History* där man ser tidigare körd kod,
- *Plots* där man ser skapade grafer, och
- *Help* där man ser hjälpsidor för funktioner.

Ofta skriver man inte sin kod direkt i konsollen, utan i ett separat *skript* - en vanlig textfil som innehåller den kod man vill köra. Genom att organisera sin kod i ett skript kan man lätt strukturera och dokumentera sitt arbete. I RStudio kan man öppna ett nytt skript genom att gå till *File > New File > R Script* eller genom att klicka *Ctrl + Shift + N*. Ett tomt skript öppnar sig då i det övre vänstra delfönstret. Om man skriver

```
a <- 5
```

i skriptet och trycker *Ctrl + Enter* bör man se att koden i skriptet körs i konsollen. Om man tittar i fliken *Environment* ska man också se att det nu skapats ett objekt *a*.

0.5 Paket i R

En av de stora styrkorna med R är att språket kan byggas ut av dess användare. De här tilläggen kan sedan samlas i paket (*packages*) och delas med andra. Rs officiella bibliotek för paket kallas för *CRAN* (*Comprehensive R Archive Network*) och består av drygt 20 000 uppladdade paket som innehåller allt från fritt tillgänglig data till avancerade statistiska modeller.

För att använda ett specifikt paket måste det först installeras. Om man vet namnet på paketet man vill installera kan man köra

```
install.packages("tidyverse")
```

I det här fallet installeras paketet `tidyverse`, vilket innehåller funktioner för hantering av data.

I RStudio kan man också installera paket från *Packages*-fliken.

Paket måste också laddas för varje ny session. Innan man kan använda innehållet i ett paket måste man därför köra

```
library(tidyverse)
```

Chapter 1

Datahantering och grafer

Datorövning 1 handlar om grunderna till R. Efter övningen ska vi kunna

- Starta RStudio och orientera oss i gränssnittet,
- Installera och ladda tilläggspaket (*Packages*),
- Definera objekt och tillämpa funktioner i R,
- Importera data från en online-källa,
- Transformera en tabell med data genom att välja kolumner, filtrera rader och summera per grupp,
- Skapa grafer med `ggplot2`.

1.1 Uppstart och orientering

För att arbeta i R måste vi installera språket R och ett gränssnitt för att arbeta i R, vanligen *RStudio*. Titta på kapitlet *Installation* om programmen inte är installerade på ditt system.

Starta RStudio, till exempel genom att gå till Startmenyn och söka på RStudio eller genom att dubbelklicka på en fil som öppnas i RStudio. Gränssnittet i RStudio är uppdelat i fyra delar och varje del består av en eller flera flikar. De viktigaste är i nuläget

- *Console* där kod körs och resultat skrivs ut,
- *Environment* där man ser skapade objekt,
- *History* där man ser tidigare körd kod,
- *Plots* där man ser skapade grafer, och
- *Help* där man ser hjälpsidor för funktioner.

Uppgift 1.1 (Help-fliken)**.** Hitta fliken *Help*, klicka på husikonen under fliken. Finns det en länk med *RStudio Cheat Sheets*? Följ den länken för att hitta guider till R som kan bli nyttiga längre fram. För nu, gå tillbaka till RStudio.

Ofta skriver man inte sin kod direkt i konsollen, utan i ett separat *skript* - en vanlig textfil som innehåller den kod man vill köra. Genom att organisera sin kod i ett skript kan man lätt strukturera och dokumentera sitt arbete. I RStudio kan man öppna ett nytt skript genom att gå till *File > New File > R Script* eller genom att klicka *Ctrl + Shift + N*. Ett tomt skript öppnar sig då i det övre vänstra delfönstret. Om du läser det här i RStudio, genom att ha laddat ner .R-filen, läser du just nu ett skript.

Uppgift 1.2 (Ett första skript)**.** Öppna ett nytt skript genom File-menyn eller genom *Ctrl + Shift + N*. Skriv

```
a <- 5
```

i skriptet och tryck *Ctrl + Enter*. Titta i flikarna *Console* och *Environment*. Har något hänt? Du bör se att koden i skriptet körts i konsollen och att ett nytt objekt `a` ligger i *Environment*.

1.2 *Packages* från CRAN

En av de stora styrkorna med R är att språket kan byggas ut av dess användare. De här tilläggen kan sedan samlas i paket (*packages*) och delas med andra. Rs officiella bibliotek för paket kallas för *CRAN* (*Comprehensive R Archive Network*).

För att använda ett specifikt paket måste det först installeras. Om man vet namnet på paketet man vill installera kan man köra

```
install.packages("tidyverse")
```

I det här fallet installeras paketet `tidyverse`, vilket innehåller funktioner för hantering av data.

I RStudio kan man också installera paket från *Packages*-fliken.

Uppgift 1.3 (Installera tidyverse-paketet)**.** Kör raden ovan för att installera `tidyverse`. Du kan antingen köra raden genom att skriva den i *Console* eller genom att skriva i ett skript och köra därifrån genom *Ctrl + Enter*.

Uppgift 1.4 (Installera gapminder-paketet)**.** Paketet `gapminder` innehåller lite intressant data vi kommer använda senare. Installera paketet `gapminder` genom att fylla i den saknade biten och köra raden nedan.

```
install.packages("___")
```

Paket måste också laddas för varje ny session. Innan man kan använda innehållet i ett paket måste man därför köra

```
library(tidyverse)
```

Uppgift 1.5 (Ladda gapminder-paketet)**.** Ladda paketet `gapminder` genom att fylla i och köra raden nedan.

```
library(___)
```

Uppgift 1.6 (Paket som inte finns)**.** Vad händer om man försöker installera ett paket som inte finns på *CRAN*? Testa till exempel

```
install.packages("ThisIsNotTheNameOfAnyPackage")
```

och

```
library(ThisIsNotTheNameOfAnyPackage)
```

1.3 Objekt och funktioner

Ett *objekt* i R är en namngiven informationsmängd. Objekt kan se ut på många olika sätt - under kursens gång används objekt som består av insamlad data (konstruerade som vektorer eller tabeller), objekt som är statistiska modeller, och flera andra former. I R skapar man objekt med *assign*-pilen `<-` (mindre än och bindestreck).

I ett tidigare exempel fanns raden

```
a <- 5
```

Här skapas ett objekt med namnet `a` som innehåller informationen `5`. *Assign*-pilen pekar alltså på det namn man vill ge objektet och pekar från objektets innehåll.

Ett lite mer komplicerat exempel på ett objekt ges av

```
b <- c(3, 1, 4, 1, 5, 9)
```

Här skapas ett objekt `b` som innehåller en *serie* numeriska värden (en *vektor*). Värdena i en vektor är ordnade och man kan plocka ut ett specifikt värde med hakparenteser.

```
b[3]              # Det tredje värdet i vektorn b
```

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

```
b[c(3,5)]         # Det tredje och femte värdet i b
```

```
## [1] 4 5
```

Uppgift 1.7 (Skapa en vektor)**.** Skapa ett objekt med namnet `new_vector` som innehåller värden 5, 7 och 10 genom att fylla i följande rad.

```
new_vector <- c(_, _, _)
```

Uppgift 1.8 (Ta ut andra värdet)**.** Använd hakparenteser för att plocka ut det andra värdet ur vektorn `new_vector`.

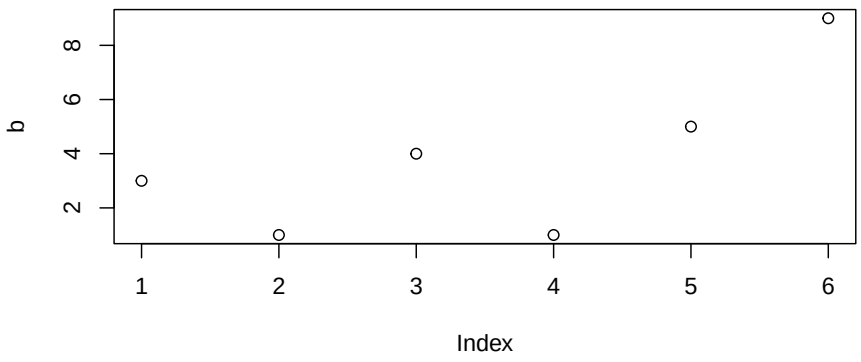
Objekt kan manipuleras genom att tillämpa *funktioner*. En funktion tar någon ingående data och ger något utgående resultat. Funktioner anges genom att skriva funktionens namn följt av ingående data inom parenteser, och resultatet kan antingen skrivas ut i konsollen eller sparas som ett nytt objekt. En grundinstallation av R innehåller en mängd färdiga funktioner, t.ex.

```
sum(b)
```

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

vilket ger summan av värdena i vektorn `b`,

```
plot(b)
```



som ger en simpel graf, och

```
sqrt(b)
```

```
## [1] 1.732051 1.000000 2.000000 1.000000 2.236068 3.000000
```

som beräknar kvadratroten för varje element i vektorn.

Uppgift 1.9 (Summera vektorn)**.** Fyll i och kör följande rad för att beräkna summan av vektorn `new_vector`

```
sum(___)
```

Vid konstruktionen av vektorn användes också en grundläggande funktion - funktionen `c()`, som tar en serie värden och skapar en sammanhängande vektor av

värden.

Alla R-funktioner har en tillhörande hjälpfil som kan plockas fram genom att skriva frågetecken följt av funktionsnamnet, t.ex. `?sum`. Från hjälpfilen får man att `sum()` tar numeriska vektorer som ingående värde och beräknar summan. Man kan styra funktionens beteende genom att sätta ett argument `na.rm` (vilket här styr hur funktionen hanterar saknade värden). Som illustration kan man titta på

```
b <- c(3, 1, 4, 1, 5, 9, NA)          # Lägger till ett saknat värde
sum(b)                                # na.rm = FALSE är grundinställning
```

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

```
sum(b, na.rm = TRUE)                  # na.rm sätts till TRUE
```

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

Det första försöket `sum(b)` ger utfallet `NA`, men om man sätter `na.rm = TRUE` beräknas summan efter att det saknade värdet plockats bort. Notera också att skript kan kommenteras med `#`.

1.4 Sekvenser av funktioner

Ofta vill man genomföra flera operationer på ett objekt. Man behöver då genomföra en sekvens av funktioner. Säg till exempel att man har värdena

$$(-4, -2, -1, 1, 2, 4)$$

och vill ta absolutvärde (vilket gör negativa tal till motsvarande positiva tal) och sedan summera. Den typen av sekvenser kan genomföras på ett par olika sätt. Ett första sätt är att spara resultatet i varje steg och sedan använda utfallet i nästa steg:

```
c <- c(-4, -2, -1, 1, 2, 4)           # Skapa en vektor av värden
c_absolute <- abs(c)                  # Ta absolutvärden. Spara som c_absolut
sum(c_absolute)                       # Summera värden i c_absolut
```

```
## [1] 14
```

Här skapas ett objekt `c` som innehåller en vektor där några tal är negativa. I nästa rad används `abs` för att skapa absolutvärden. Slutligen summeras absolutvärdena med `sum`. Notera att det är möjligt att skapa ett objekt med namnet `c` trots att det redan är namnet på en funktion - R förstår ur sammanhanget om objektet eller funktionen ska användas. Det kan dock bli lite oklart för en läsare, så försök som regel att undvika att skapa objekt med vanliga funktionsnamn som `sum` och `mean`.

Uppgift 1.10 (Kvadrat, summa och roten ur)**.** Fyll i och kör följande rader för att ta varje värde i `new_vector` i kvadrat, *sedan* summera, och sedan ta roten

ur.

```
new_vector_squared <- new_vector^2      # Ta kvadraten av varje värde
new_vector_squared_sum <- sum(___)      # Summera vektorn med kvadrater
sqrt(___)                               # Ta kvadratroten ur summan
```

Ett alternativ till att spara utfallet i varje steg är att skriva en senare funktion *runt* en tidigare funktion. Det fungerar för att R utvärderar funktioner inifrån-ut. Med samma exempel som tidigare får man

```
sum(abs(c(-4, -2, -1, 1, 2, 4)))        # Ta summan av absolutvärden av vektorn
```

medan beräkningen i övningen blir

```
sqrt(sum(new_vector^2))                 # Ta roten ur summan av vektorn i kvadrat
```

Den här typen av skrivning kan spara plats men blir snabbt svårläst.

Ett sista alternativ är att använda en så kallad *pipe* (namnet kommer från att en sekvens funktioner kallas en *pipeline*). En pipe skrivs `%>%` och kan i RStudio tas fram med kortkommandon *Ctrl + Shift + M*. Pipen tar utfallet av en funktion till vänster och sänder till en funktion till höger. Den kan utläsas i dagligt tal som *och sen*. Med samma exempel som tidigare kan vi skriva

```
library(tidyverse)                      # Ladda tidyverse, ej nödvändigt om redan gjort

c(-4, -2, -1, 1, 2, 4) %>%              # Skapa en datamängd och sen
  abs() %>%                             # ta absolutvärden, och sen
  sum()                                 # beräkna summan.
```

```
## [1] 14
```

Uppgift 1.11 (Kvadrat, summa och rot med pipe)**.** Fyll i de saknade funktionerna och kör följande rader för att ta varje värde i `new_vector` i kvadrat, *sedan* summera, och sedan ta roten ur, denna gång genom att länka funktionerna med en pipe `%>%`.

```
new_vector^2 %>%                        # Ta kvadraterna av new_vector, och sen
  ___() %>%                             # beräkna summan, och sen
  ____()                                # Ta kvadratroten med sqrt()
```

1.5 Datainskrivning och dataimport från web

1.5.1 Inskrivning av data

Det första praktiska steget i en statistisk analys är att importera data. I R kan det göras genom att direkt skriva in sin data och spara som ett nytt objekt, men ett bättre och vanligare sätt är att importera sin data från en extern fil eller databas.

I ett tidigare exempel användes funktionen `c()` för att skapa en vektor av data. Ofta ordnas flera vektorer i en tabell där varje kolumn är en vektor och varje rad en observation av någon enhet. En datatabell (en `data.frame` i R) skapas genom funktionen `data.frame()` följt av namngivna vektorer. Exempeldata kan skrivas in genom följande.

```
dat <- data.frame(Vecka = c(7,7,7,7,7,7,11,11,11,11,11,11),
                  Behandling = c("A","A","A","B","B","B","A","A","A","B","B","B"),
                  Vikt = c(232,161,148,368,218,257,1633,2213,972,2560,2430,855),
                  N = c(2.63,2.90,2.99,3.54,3.30,2.85,1.53,1.90,NA,2.58,NA,NA))
dat
```

```
##    Vecka Behandling Vikt    N
## 1      7          A  232 2.63
## 2      7          A  161 2.90
## 3      7          A  148 2.99
## 4      7          B  368 3.54
## 5      7          B  218 3.30
## 6      7          B  257 2.85
## 7     11          A 1633 1.53
## 8     11          A 2213 1.90
## 9     11          A  972   NA
## 10    11          B 2560 2.58
## 11    11          B 2430   NA
## 12    11          B  855   NA
```

Radbrytningar och blanksteg är oviktiga i R, och används bara för läsbarhet här. Saknade värden skrivs in som `NA` för *not available*. Notera att alla kolumner inte behöver vara av samma datatyp men att värden inom en kolumn måste vara det. Här är *Behandling* text medan övriga kolumner är numeriska.

Uppgift 1.12 (Alea iacta est)**.** Kasta din tärning tio gånger och skriv in resultatet i en datatabell i R med hjälp av grundkoden nedan. Om du saknar en tärning, fråga lämplig person om du kan få en. Behåll tärningen, den behövs till nästa datorövning (och närhelst man står inför ett avgörande livsbeslut).

```
dat_dice <- data.frame(Kast = c(1,2,3,4,5,6,7,8,9,10),
                       Utfall = c(_,_,_,_,_,_,_,_,_,_))
dat_dice
```

Objektet `dat` är av typen `data.frame` - en tabell med rader och kolumner. Man kan ange en specifik kolumn i en data.frame med dollartecken följt av kolumnens namn.

```
dat$Vikt
```

```
##  [1]  232  161  148  368  218  257 1633 2213  972 2560 2430  855
```

Man kan också plocka ut rader och kolumner med hakparenteser och ordningstal.

```
dat[2,3]          # Andra raden och tredje kolumnen
```

```
## [1] 161
```

```
dat[2, ]           # Tomt värde ger alla värden. Här alla värden i rad 2
```

```
##   Vecka Behandling Vikt   N
## 2     7           A  161 2.9
```

```
dat[ ,3]           # Alla värden i kolumn 3
```

```
##  [1]  232  161  148  368  218  257 1633 2213  972 2560 2430  855
```

Uppgift 1.13 (Plocka ut en specifik kolumn)**.** I den tidigare övningen skapade du ett objekt `dat_dice`. Använd dollartecken för att plocka ut kolumnen *Utfall* från det objektet.

```
dat_dice$___
```

Genom att plocka ut en kolumn från en data.frame kan man beräkna vanlig beskrivande statistik med funktioner som `mean()` (medelvärde) och `sd()` (standardavvikelsen).

```
mean(dat$Vikt)
```

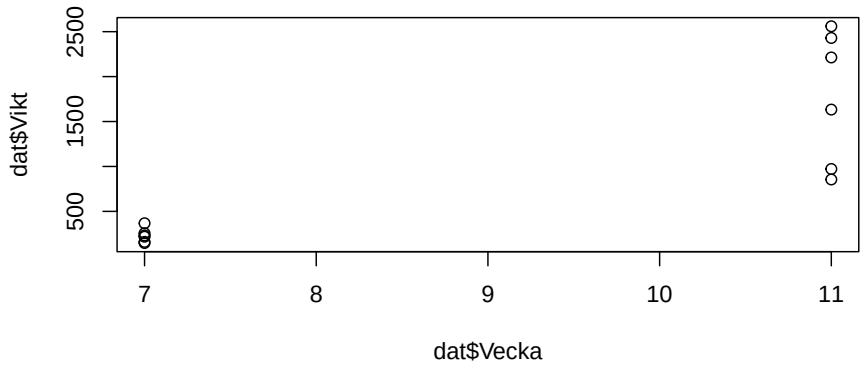
```
## [1] 1003.917
```

```
sd(dat$Vikt)
```

```
## [1] 951.3067
```

Funktionen `plot()` ger en enkel graf.

```
plot(dat$Vecka, dat$Vikt)
```



Uppgift 1.14 (Alea iacta est)**.** Använd datan i objektet `dat_dice` och skapa ett diagram med kolumnen kast på x-axeln och kolumnen Utfall på y-axeln.

```
plot(dat_dice$___, dat_dice$___)
```

1.5.2 Import från en extern fil

Inskrivning av värden är ofta tidskrävande och kan lätt leda till misstag. Det är därför mycket vanligare att data läses in från en extern fil. Det finns en mängd funktioner för dataimport och det exakta valet av funktion beror på vilken typ av fil datan är sparad i. Ett vanligt filformat är .csv (*comma separated values*). Här importerar vi en fil med data från Spotify. Filen ligger på Github, en populär sida för lagring av filer och kod.

```
dat <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2023/master/Data/Spotify_data.c
                                        # Läs in en csv-fil från Github
dat                                     # Skriv ut objektet dat
```

```
## # A tibble: 89,105 x 24
##    artist_name album_name  track_number track_name album_type album_release_date
##    <chr>       <chr>              <dbl> <chr>      <chr>      <chr>
##  1 100 gecs    1000 gecs ~            1 money mac~ album      2020-07-10
##  2 100 gecs    1000 gecs ~            2 ringtone ~ album      2020-07-10
##  3 100 gecs    1000 gecs ~            3 745 stick~ album      2020-07-10
##  4 100 gecs    1000 gecs ~            4 gec 2 Ü (~ album      2020-07-10
##  5 100 gecs    1000 gecs ~            5 hand crus~ album      2020-07-10
##  6 100 gecs    1000 gecs ~            6 800db clo~ album      2020-07-10
##  7 100 gecs    1000 gecs ~            7 stupid ho~ album      2020-07-10
##  8 100 gecs    1000 gecs ~            8 ringtone ~ album      2020-07-10
##  9 100 gecs    1000 gecs ~            9 xXXi_wud_~ album      2020-07-10
## 10 100 gecs    1000 gecs ~           10 745 stick~ album      2020-07-10
## # i 89,095 more rows
## # i 18 more variables: album_release_year <dbl>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, time_signature <dbl>, explicit <lgl>, type <chr>,
## #   key_name <chr>, mode_name <chr>, key_mode <chr>
```

Om importen fungerat kan man skriva ut tabellens översta rader genom att köra objektets namn.

För att snabbt se vilka artister datan täcker kan man köra

```
unique(dat$artist_name)                 # Skriv ut unika värden i kolumnen artist_name i dat
```

där `unique` är en funktion som tar bort alla dubletter och `dat$artist_name` används för att plocka ut kolumnen `artist_name` ur tabellen `dat`.

1.6 Urval ur en tabell med `select` och `filter`

En vanlig operation på en tabell är att göra ett urval - antingen ett urval av rader (t.ex. en viss artist), vilket kallas *filtrering* eller ett urval av variabler (t.ex. artist och albumnamn), vilket kallas *selektion*. Här tittar vi på hur det kan göras med funktionerna `filter()` och `select()` från paketet `tidyverse`.

För att filtrera på en given artist kan man använda pipe-funktionen från datan till en filter-funktion, t.ex.

```
dat %>%                                 # Ta spotify-datan och sen
  filter(artist_name == "Robyn")        # filtrera för en specifik artist
```

```
## # A tibble: 1,141 x 24
##    artist_name album_name track_number track_name  album_type album_release_date
##    <chr>       <chr>              <dbl> <chr>       <chr>      <chr>
##  1 Robyn       Honey                  1 Missing U   album      2018-10-26
##  2 Robyn       Honey                  2 Human Being album      2018-10-26
##  3 Robyn       Honey                  3 Because It~ album      2018-10-26
##  4 Robyn       Honey                  4 Baby Forgi~ album      2018-10-26
##  5 Robyn       Honey                  5 Send To Ro~ album      2018-10-26
##  6 Robyn       Honey                  6 Honey       album      2018-10-26
##  7 Robyn       Honey                  7 Between Th~ album      2018-10-26
##  8 Robyn       Honey                  8 Beach2k20   album      2018-10-26
##  9 Robyn       Honey                  9 Ever Again  album      2018-10-26
## 10 Robyn       Honey                  1 Missing U   album      2018-10-26
## # i 1,131 more rows
## # i 18 more variables: album_release_year <dbl>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, time_signature <dbl>, explicit <lgl>, type <chr>,
## #   key_name <chr>, mode_name <chr>, key_mode <chr>
```

Inom filterfunktionen anges ett logisk villkor `artist_name == "Robyn"` och utfallet är de rader där villkoret är sant. Notera de dubbla likhetstecknen - de måste användas för ett logisk villkor eftersom enkelt likhetstecken används för att skapa objekt och sätta funktionsargument.

Uppgift 1.15 (Filtrera för artist)**.** Vad måste ändras i koden för att istället plocka ut rader där artisten är Esperanza Spalding? Hur många rader har det urvalet i datan?

```
dat %>%                                 # Ta spotify-datan och sen
  filter(artist_name == "Robyn")        # filtrera för en specifik artist
```

Om man vill välja flera artister kan man använda funktionen `%in%` på ett liknande sätt.

```
dat %>%                                                    # Ta datan, och sen
  filter(artist_name %in% c("Robyn", "Esperanza Spalding"))  # filtrera för specifika ar
```

och om man vill ha mer än ett villkor kan man rada dem i filter-funktionen:

```
dat %>%                                                    # Ta datan, och sen
  filter(artist_name %in% c("Robyn", "Esperanza Spalding"),  # filtrera för specifika ar
         key_name == "D#")                                 # och för tonart
```

För att se fler eller färre rader kan man använda en pipe %>% till funktionen print(). Följande skriver ut fem rader

```
dat %>%                                                    # Ta datan, och sen
  filter(artist_name %in% c("Robyn", "Esperanza Spalding")) %>%  # filtrera för artister, och
  filter(key_name == "D#")  %>%                            # filtrera för tonart, och s
  print(n = 5)                                             # skriv ut de fem första rad
```

```
## # A tibble: 21 x 24
##   artist_name   album_name track_number track_name album_type album_release_date
##   <chr>         <chr>             <dbl> <chr>      <chr>      <chr>
## 1 Esperanza Sp~ SONGWRIGH~            7 Formwela 7 album      2021-09-24
## 2 Esperanza Sp~ 12 Little~            2 To Tide U~ album      2019-05-10
## 3 Esperanza Sp~ 12 Little~            3 'Til the ~ album      2019-05-10
## 4 Esperanza Sp~ Emily's D~            7 Ebony And~ album      2016-01-01
## 5 Esperanza Sp~ Emily's D~            7 Ebony And~ album      2016-01-01
## # i 16 more rows
## # i 18 more variables: album_release_year <dbl>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, time_signature <dbl>, explicit <lgl>, type <chr>,
## #   key_name <chr>, mode_name <chr>, key_mode <chr>
```

Om man istället vill göra ett urval av kolumner kan man använda select().
Som argument anges de kolumner man vill välja, t.ex.

```
dat %>%                                  # Ta datan, och sen
  select(artist_name, album_name)        # välj kolumnerna artist_name och album_name
```

```
## # A tibble: 89,105 x 2
##    artist_name album_name
##    <chr>       <chr>
##  1 100 gecs    1000 gecs and The Tree of Clues
##  2 100 gecs    1000 gecs and The Tree of Clues
##  3 100 gecs    1000 gecs and The Tree of Clues
##  4 100 gecs    1000 gecs and The Tree of Clues
##  5 100 gecs    1000 gecs and The Tree of Clues
##  6 100 gecs    1000 gecs and The Tree of Clues
##  7 100 gecs    1000 gecs and The Tree of Clues
```

```
##  8 100 gecs    1000 gecs and The Tree of Clues
##  9 100 gecs    1000 gecs and The Tree of Clues
## 10 100 gecs    1000 gecs and The Tree of Clues
## # i 89,095 more rows
```

Som avslutning ges ett lite mer komplicerat exempel på ett urval av artist, år och spår för spår med ett tempo över 180 bpm släppta under 2015.

```
dat %>%                                                    # Ta datan och sen
  filter(album_release_year == 2015,                       # filtrera för rader där år
         tempo > 180) %>%                                  # tempot över 160, och sen
  select(artist_name, album_release_year, track_name)      # selektera på artist, år o
```

```
## # A tibble: 129 x 3
##    artist_name            album_release_year track_name
##    <chr>                                <dbl> <chr>
##  1 A Sunny Day In Glasgow                2015 The Strange Presents of Idols (Mer~
##  2 A$AP Rocky                            2015 Lord Pretty Flacko Jodye 2 (LPFJ2)
##  3 A$AP Rocky                            2015 Lord Pretty Flacko Jodye 2 (LPFJ2)
##  4 Anderson .Paak                        2015 Off the Ground
##  5 Arcangel                              2015 50 Sombras De Austin
##  6 Arcangel                              2015 50 Sombras de Austin
##  7 Arcangel                              2015 50 Sombras de Austin
##  8 Arcangel                              2015 50 Sombras de Austin
##  9 Björk                                 2015 Notget - Live
## 10 Björk                                 2015 Notget
## # i 119 more rows
```

Uppgift 1.16 (Snabba spår)**.** Funktionen `arrange()` sorterar data efter en angiven kolumn. Följande stycke ger oss Björks snabbaste spår.

```
dat %>%                                                    # Ta datan, och sen
  filter(artist_name == "Björk") %>%                       # filtrera för rader där ar
  select(artist_name, album_name, track_name, tempo) %>%   # välj kolumner med artist,
  arrange(-tempo)                                          # ordna efter tempo (minus
```

Gör lämpliga ändringar för att hitta Kate Bushs snabbaste spår.

1.7 Grafer med ggplot2

Vi kan nu börja titta på grafer. Eftersom spotify-datan är ganska stor och grafer lätt blir oöversiktliga, börjar vi med att skapa en lite mindre datamängd.

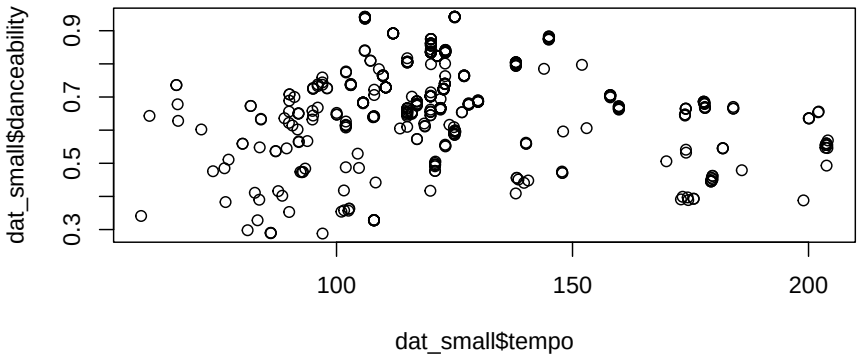
```
dat_small <- dat %>%                                       # Ta data, och sen
  filter(artist_name == "Robyn", album_type == "album")    # filtrera på artist och alb
```

Uppgift 1.17 (Valfritt artistbyte)**.** Om du vill titta på data för någon annan artist, gör lämplig ändring i stycket ovan. Kom ihåg att man kan skriva ut

artister i datan med `unique(dat$artist_name)`.

R har en mängd grundläggande funktioner för grafer. Vi såg tidigare ett exempel på funktionen `plot()`.

```
plot(dat_small$tempo, dat_small$danceability)    # Plotta en graf med tempo och dansbarhet på axl
```



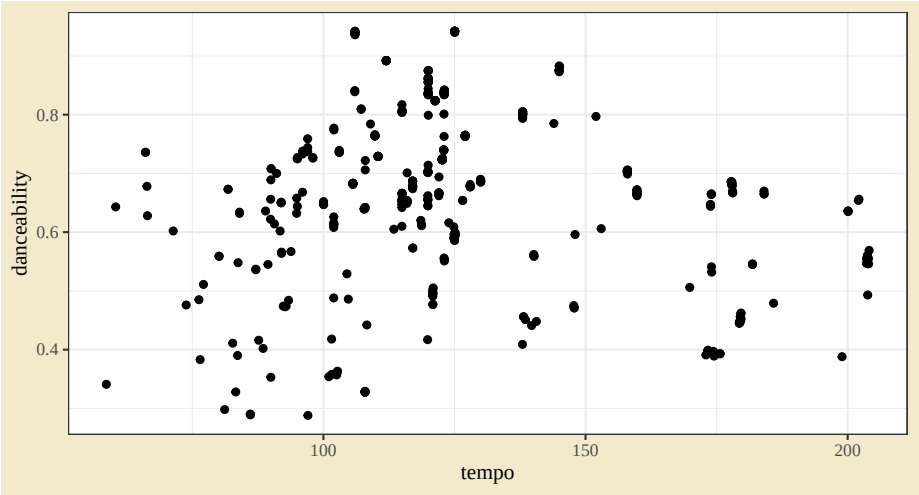
Tecknet `$` används här för att välja kolumnerna tempo och danceability ur objektet `dat_small`.

För mer avancerade grafer används dock ofta funktioner ur Rs paketbibliotek. Här illustreras det mest populära - `ggplot2`. I `ggplot2` byggs grafer upp med tre grundläggande byggstenar:

- *data*, informationen man vill visualisera,
- *aestethics*, en koppling mellan data och visuella element såsom grafens axlar, objekts storlek och färg,
- *geometries*, de geometriska former som visas i grafen.

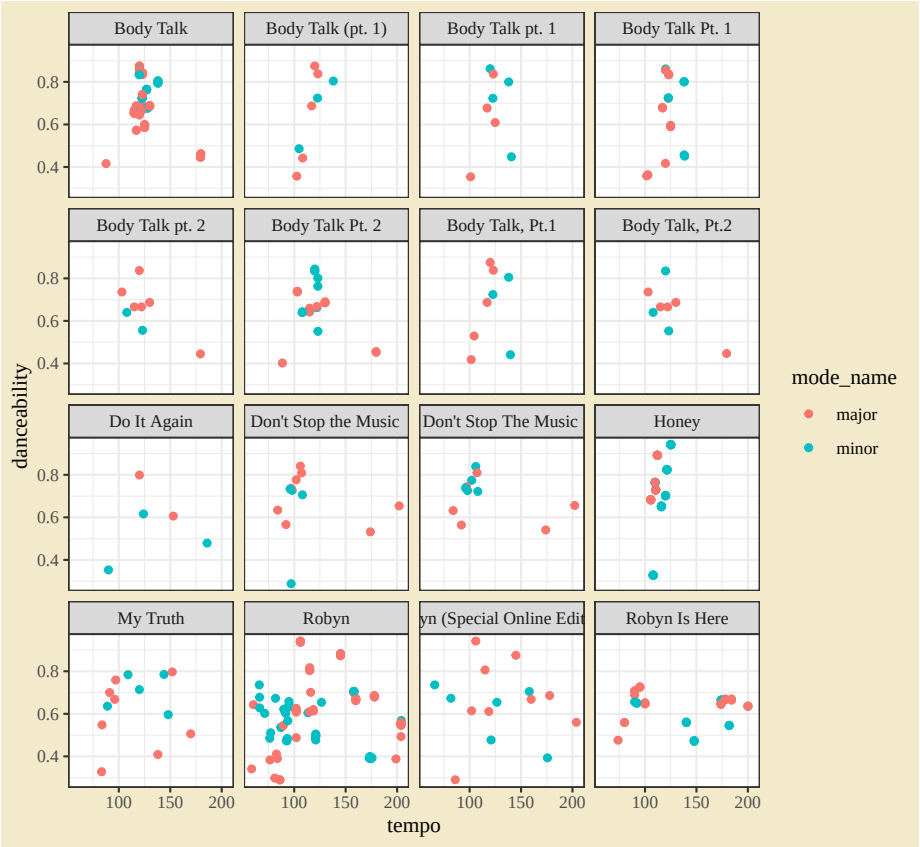
En graf skrivs med en startfunktion `ggplot` som anger namnet på datan och grafens *aestethics*, och därefter sätts geometriska element genom funktioner som börjar med `geom_`. Ett spridningsdiagram kan t.ex. skapas med `geom_point`.

```
ggplot(dat_small, aes(x = tempo, y = danceability)) +  # Ta datan, koppla tempo och dansbarhet t
  geom_point()                                         # och illustrera varje observation med pu
```



Grafen kan byggas ut genom att sätta *aestethics* för färg och storlek. Man kan också dela en graf i småfönster med `facet_wrap` och styra grafens utseende genom att sätta ett tema såsom `theme_bw`.

```
ggplot(dat_small, aes(x = tempo, y = danceability, color = mode_name)) +   # Ta datan,
  geom_point() +                                                           # Illustrer
  facet_wrap(~ album_name)                                                 # Skapa små
```



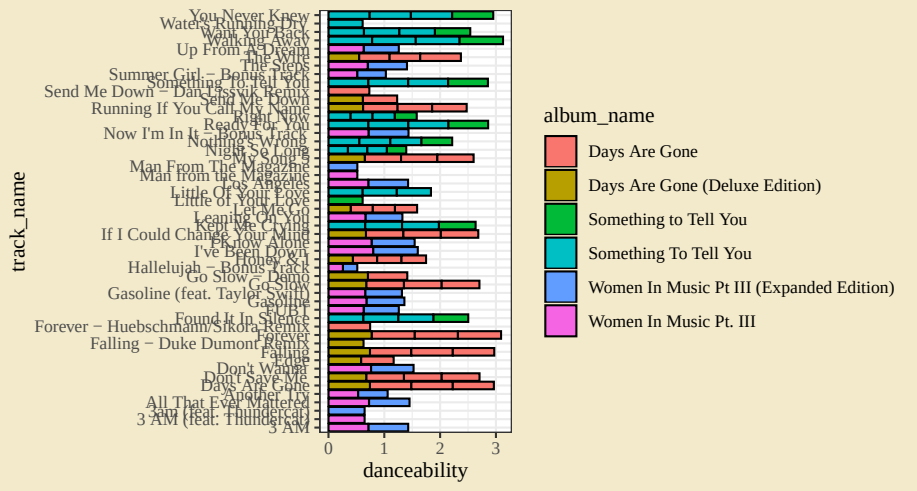
Uppgift 1.18 (Dur och moll)**.** Vad ska ändras i stycket nedan för att skapa en graf med dur/moll (`mode_name`) på x-axeln, valens (`valence`) på y-axeln och skilda småfönster för olika år (`album_release_year`)?

```
ggplot(dat_small, aes(x = ____, y = ____, color = album_name)) + # Ta datan och koppla variabler
  geom_point() +                                                 # Illustrera med punkter
  facet_wrap(~ album_release_year)                               # Skapa småfönster efter år
```

Har spår i dur (*major*) högre valens?

Andra graftyper kan skapas med andra `geom_`-funktioner. Stapeldiagram ges av `geom_col` (`col` för *column*). Man kan också använda `geom_bar` om man bara vill räkna antal rader per någon kategori. Följande väljer ut en artist och plottar spårens dansbarhet i ett (liggande) stapeldiagram.

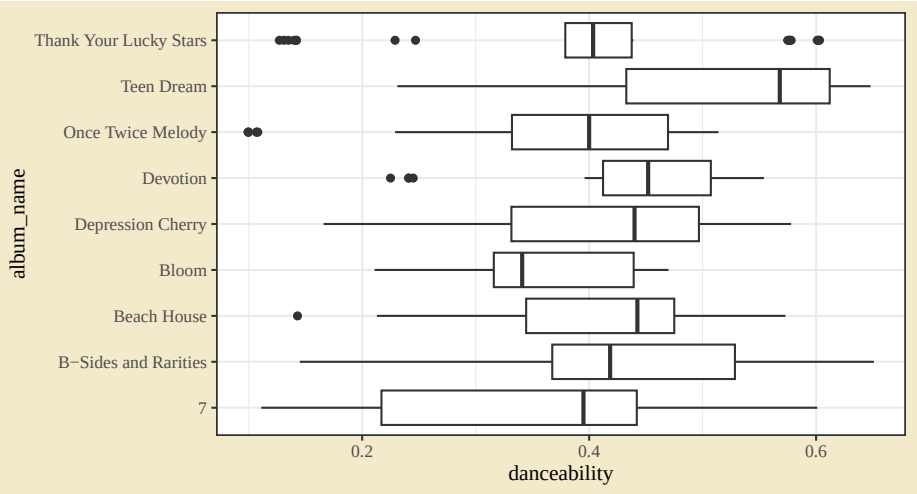
```
dat_small <- dat %>%
  filter(artist_name == "HAIM", album_type == "album")

ggplot(dat_small, aes(danceability, track_name, fill = album_name)) +
  geom_col(color = "black")
```



Argumentet `fill` styr färgen för ytor (här staplarnas ytor) medan `color` i `geom_col()` styr kanten runt varje stapel.

Man kan styra grafiken i en `ggplot` genom funktionen `theme()`. Det är ett ganska komplicerat ämne, men låt oss titta på några grunder. Vi börjar med att skapa en enkel graf: en boxplot över dansbarhet per album för bandet *Beach House*.

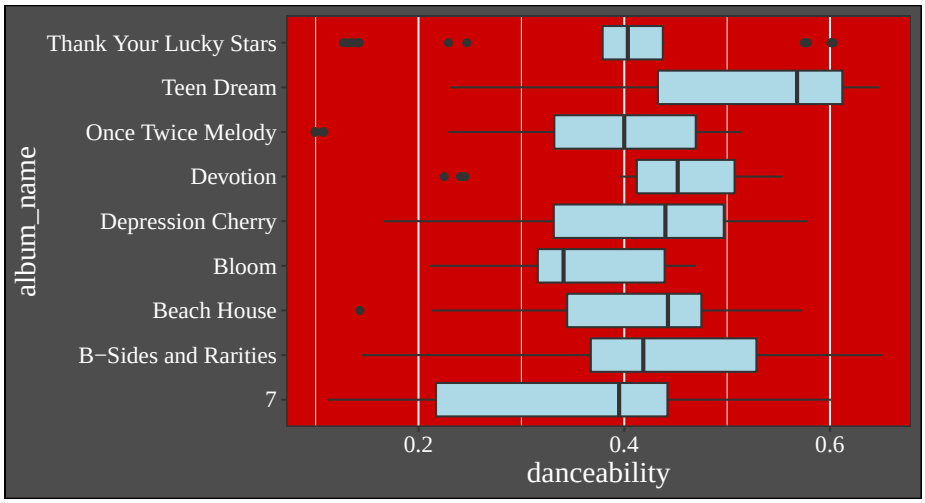
```
dat_small <- dat %>%                                    # Ta datan, och sen
  filter(artist_name == "Beach House", album_type == "album")    # filtrera på artist o

ggplot(dat_small, aes(x = danceability, y = album_name)) +      # Ta data och koppla d
  geom_boxplot()                                        # Illustrera med lådag
```



Vi kan ändra utseendet på grafen genom argument inom geometrier och med

funktionen `theme()`. I `theme()` sätter man de specifika egenskaper man vill ändra genom att tillskriva dem ett *element*. Valet av element beror på typen av grafiskt objekt - text sätts t.ex. med `element_text()` och ytor med `element_rect()` (för *rectangle*). Vi ger ett exempel med ändrad bakgrund, rutmönster, och teckenstorlek.

```
ggplot(dat_small, aes(danceability, album_name)) +
  geom_boxplot(fill = "lightblue") +                                    # Fyll lådagrammen m
  theme(panel.background = element_rect(fill = "red3"),                 # Sätt grafenfönstre
        text = element_text(size = 15, color = "white", family = "serif"),   # Sätt textens stor
        axis.text = element_text(color = "white"),                      # Sätt axel-textens
        plot.background = element_rect(fill = "grey30", color = "black"),    # Sätt grafens bakgr
        panel.grid.major.y = element_blank())                           # Sätt rutnätet till
```



Uppgift 1.19 (Temaval 1)**.** Ändra färgvalen i grafen ovan för att skapa snyggast möjliga graf. Funktionen `colors()` ger de färger som finns tillängliga i R. Man kan också använda hex-koden för färger, t.ex. `fill = "#ffdd00"`.

Uppgift 1.20 (Temaval 2)**.** Ändra färgvalen i grafen ovan för att skapa fulast möjliga graf. Visa de två graferna för någon annan och se om de kan säga vilken som är vilken.

1.8 Bonus. Interaktiva grafer med `plotly`

Låt oss ta en titt på `plotly`, ett av flera R-paket som gör det möjligt att skapa interaktiva grafer. Vi börjar med att installera och ladda paketet.

```
# install.packages("plotly")                    # Installera plotly (ej nödvändigt om redan inst
library(plotly)                                 # Ladda plotly
```

Paketet innehåller en smidig funktion `ggplotly()` för att göra en interaktiv

graf från en `ggplot`. Vi börjar med att filtrera datan för en specifik artist och albumtyp. I samma pipe skapar vi en ny kolumn decade, som beräknar årtiondet utifrån året. Den exakta beräkning är inte så viktig, men ta gärna en titt och se om du förstår vad som delarna gör.

```
dat_small <- dat %>%                                                    # Ta datan, och s
  filter(artist_name == "David Bowie", album_type == "album") %>%       # filtrera på art
  mutate(Decade = floor(album_release_year / 10) * 10)                  # skapa en variab
```

(Säg t.ex. att vi har året 1979. Att dela med 10 ger 197.9. Funktionen `floor` avrundar nedåt till 197. Multiplikationen med 10 ger 1970.)

Vi kan nu konstruera en graf med `ggplot()`. Låt oss ha dansbarhet på x-axeln och valens på y-axeln. Geomet `geom_point()` ger ett spridningdiagram och `facet_wrap(~ Decade)` delar i småfönster efter årtionde. Slutligen tar `theme(legend.position = "none")` bort *legenden* - guiden som anger vilken färg som är vilket album.

Notera att vi sparar grafen som ett objekt `g`. För att se grafen kör vi objektnamnet.

```
g <- ggplot(dat_small, aes(danceability, valence, color = album_name, text = track_name
  geom_point() +
  facet_wrap(~ Decade) +             # Skapa småfönster per årtionde
  theme(legend.position = "none")    # Ta bort legenden (kopplingen mellan färg och alb
g
```

När vi har en färdig ggplot kan `ggplotly()` ge en interaktiv version av samma graf.

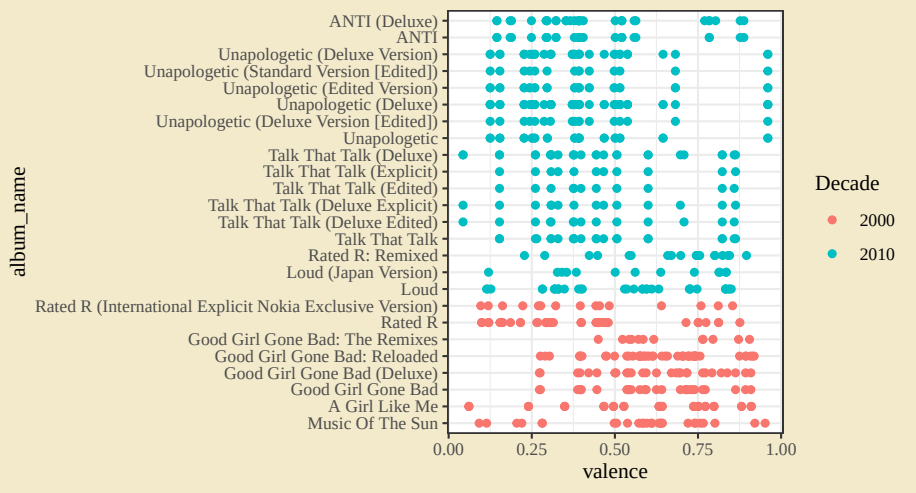
```
ggplotly(g)
```

Uppgift 1.21 (Interaktiv graf med annan artist)**.** Gör lämpliga ändringar i stycket nedan för att skapa en interaktiv graf med en annan artist och med tempo på x-axeln och dansbarhet på y-axeln. Kom ihåg att du kan se tillgängliga artister med raden `unique(dat$artist_name)`.

```
dat_small <- dat %>%
  filter(artist_name == "David Bowie", album_type == "album") %>%
  mutate(Decade = floor(album_release_year / 10) * 10)

g <- ggplot(dat_small, aes(danceability, valence, color = album_name, text = track_name
  geom_point() +                                  # Skapa punkter
  facet_wrap(~ Decade) +                          # Skapa småfönster per årtionde
  theme(legend.position = "none")                 # Ta bort legenden (kopplingen mellan
g

ggplotly(g)
```

Uppgift 1.22 (Interaktiv graf med andra axlar)**.** Vi fortsätter med ett nytt exempel, nu med ett spridningsdiagram med album på y-axeln och valens på x-axeln. Funktionen `reorder()` ordnar en kolumn efter en annan. Här ordnas album efter release-år.

```
dat_small <- dat %>%
  filter(artist_name == "Rihanna", album_type == "album") %>%
  mutate(Decade = floor(album_release_year / 10) * 10,
         Decade = as.character(Decade),
         album_name = reorder(album_name, album_release_year))

g <- ggplot(dat_small, aes(valence, album_name, color = Decade, text = track_name)) +
  geom_point()
g
```



Ändra gärna artist på lämpligt ställe. Vad måste läggas till för en interaktiv version av samma graf?

Hemsidan https://plotly.com/r/ innehåller fler exempel för den som är intresserad.

1.9 Bonus. Warming stripes

Warming stripes har sedan de först introducerades av Ed Hawking 2018 blivit en vanlig illustration av temperaturökning. I en warming stripe-graf anges varje år av en stapel och stapels färg ges av ett temperaturmått, vanligen årets medeltemperatur. I ggplot-terminologi har vi geometrier (staplar eller kolumner) med en x-position som ges av år och en ifylld färg som ges av temperatur.

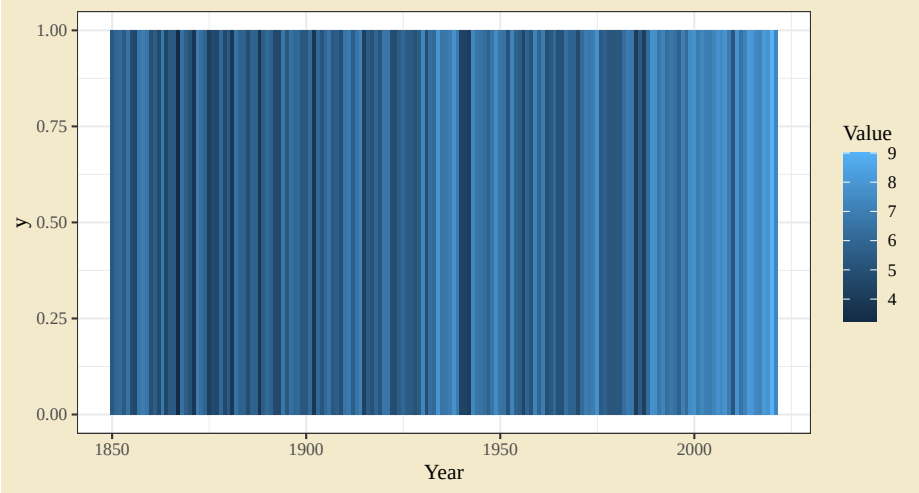
För att göra en graf behöver vi data över temperaturer. Följande rad hämtar temperaturdata från Stockholm. Källa: https://miljobarometern.stockholm.se

/klimat/klimat-och-vaderstatistik/medeltemperatur/

```
# Läs in data från fil
dat_temp <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2022/main/Data/
```

Vi skapar nu en graf, som alltså ska ha en x-axeln given av år (`Year`) och ifylld färg som ges av temperatur (`Value`). Höjden ska vara densamma för alla staplar, men det spelar ingen roll vad den är (så länge den inte är noll).

```
ggplot(dat_temp, aes(x = Year, y = 1, fill = Value)) +
  geom_col()
```



Uppgift 1.23 (Staplar utan mellanrum)**.** Ett första problem är att staplarna inte fyller ytan. Man kan styra staplars bredd med argumentet `width`, t.ex.

```
ggplot(dat_temp, aes(x = Year, y = 1, fill = Value)) +
  geom_col(width = 0.1)
```

Hitta ett värde för `width` som ger staplar utan mellanrum.

Uppgift 1.24 (Färgval)**.** Ett andra problem är att ggplots grundval för färger är från svart till blått. För klassiska warming stripes vill vi ha en skala från blått till rött. Färgerna i en skala ändras med särskilda `scale_()`-funktioner. En färgskala för ifylld färg kan sättas med `scale_fill_gradientn()`, till exempel

```
ggplot(dat_temp, aes(x = Year, y = 1, fill = Value)) +
  geom_col(width = 0.1) +
  scale_fill_gradientn(colours = c("darkgreen", "blue", "white", "yellow", "purple"))
```

Välj färger som ger en naturlig skala från blått till rött. Funktionen `colors()` ger valbara färger i R. Några möjliga val kan vara `darkblue`, `blue`, `white`, `red`, `salmon`, `darkred`, `steelblue` och `skyblue`.

Uppgift 1.25 (Enkel graf)**.** Slutligen brukar warming stripes presenteras med så lite kringinformation som möjligt. I ggplot kan grafelement tas bort med `theme()`. Här är som exempel en graf utan y-axel, tickmärken och legend.

```
ggplot(dat_temp, aes(x = Year, y = Value)) +
  geom_line() +
  theme(axis.title = element_blank(),            # Sätt titel till blank
        legend.position = "none",                # Ta bort legenden
        plot.background = element_blank(),       # Sätt bakgrunden till blank
        panel.background = element_blank(),      # Sätt graffältets bakgrund till blank
        axis.text.y = element_blank(),           # Sätt axeltext till blank
        axis.ticks = element_blank())            # Sätt axelticks till blank
```

Använd temat från exemplet för att skapa en enklare version av grafen från föregående uppgift. Ett liknande resultat kan fås med temat `theme_void()`.

Chapter 2

Beskrivande statistik

Datorövning 2 handlar om beräkning av beskrivande statistik i R. Efter övningen ska vi kunna

- Importera data från en excelfil,
- Beräkna lämpliga lägesmått för en variabel,
- Beräkna lämpliga spridningsmått för en variabel,
- Konstruera grafer som jämför två eller flera gruppers läge och spridning.

2.1 Repetition från datorövning 1

När man arbetar i R är det klokt att använda ett avancerat gränssnitt som RStudio och att skriva sin kod i ett separat skript. I RStudio kan man starta ett nytt skript genom *Ctrl + Shift + N*.

Mycket funktionalitet i R ligger i tilläggspaket (*Packages*). Paket måste installeras första gången de används och laddas varje session de används, t.ex.

```
# install.packages("tidyverse")        # Installera tidyverse (behövs ej om redan gjort)
library(tidyverse)                     # Ger ett felmeddelande om paketet inte installerats
```

Data läses in med import-funktioner där valet av funktion beror på typen av fil. Importerad data sparas som ett objekt i R genom *assign*-pilen `<-`.

```
dat <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2022/main/Data/Spotify_data.csv
dat                                                # Skriv ut objektet dat
```

```
## # A tibble: 89,105 x 24
##    artist_name album_name  track_number track_name album_type album_release_date
##    <chr>       <chr>              <dbl> <chr>      <chr>      <chr>
##  1 100 gecs    1000 gecs ~            1 money mac~ album      2020-07-10
```

```
##  2 100 gecs    1000 gecs ~            2 ringtone ~ album      2020-07-10
##  3 100 gecs    1000 gecs ~            3 745 stick~ album      2020-07-10
##  4 100 gecs    1000 gecs ~            4 gec 2 Ü (~ album      2020-07-10
##  5 100 gecs    1000 gecs ~            5 hand crus~ album      2020-07-10
##  6 100 gecs    1000 gecs ~            6 800db clo~ album      2020-07-10
##  7 100 gecs    1000 gecs ~            7 stupid ho~ album      2020-07-10
##  8 100 gecs    1000 gecs ~            8 ringtone ~ album      2020-07-10
##  9 100 gecs    1000 gecs ~            9 xXXi_wud_~ album      2020-07-10
## 10 100 gecs    1000 gecs ~           10 745 stick~ album      2020-07-10
## # i 89,095 more rows
## # i 18 more variables: album_release_year <dbl>, danceability <dbl>,
## #   energy <dbl>, key <dbl>, loudness <dbl>, mode <dbl>, speechiness <dbl>,
## #   acousticness <dbl>, instrumentalness <dbl>, liveness <dbl>, valence <dbl>,
## #   tempo <dbl>, time_signature <dbl>, explicit <lgl>, type <chr>,
## #   key_name <chr>, mode_name <chr>, key_mode <chr>
```

Funktioner agerar på objekt och ger något utfall. Här beräknas medeltempot med funktionen `mean()`. Dollartecknet används för att ange en specifik kolumn i dataobjektet. Funktioner styrs av möjliga argument - här används `na.rm` för att ange att saknade värden inte ska tas med i beräkningen

```
mean(dat$tempo, na.rm = T)                    # Beräkna medelvärdet av kolumnen tem
```

```
## [1] 119.0461
```

Funktionerna `filter()` och `select()` kan användas för att välja kolumner och rader. Funktioner kan länkas samman med en pipe `%>%` för att skapa sekvenser av funktioner. Man kan tänka på pipen som *och sen*.

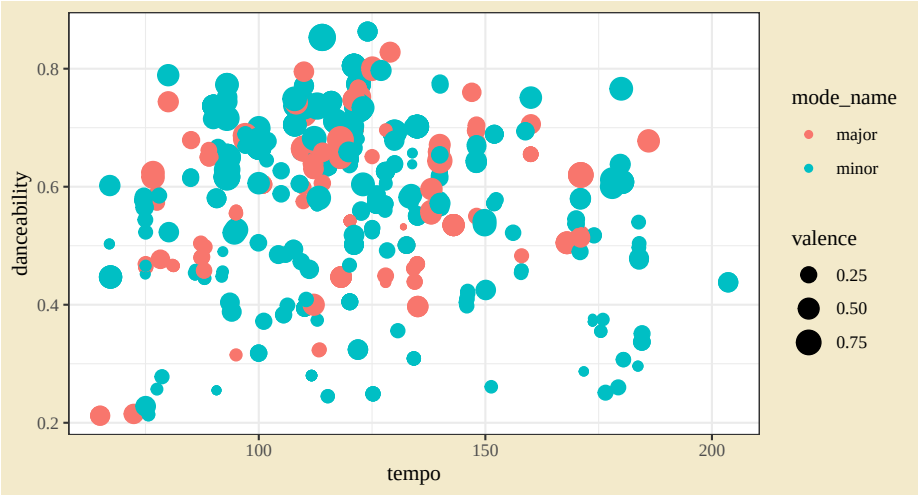
```
dat %>%                                                  # Ta datan, och sen
  filter(artist_name == "Tame Impala", tempo > 170) %>%  # ta ut rader där artisten
  select(artist_name, track_name, tempo)                 # ta ut kolumnerna artist_n
```

```
## # A tibble: 10 x 3
##    artist_name track_name                                        tempo
##    <chr>       <chr>                                             <dbl>
##  1 Tame Impala Lost In Yesterday                                  183.
##  2 Tame Impala Yes I'm Changing                                   180.
##  3 Tame Impala Led Zeppelin                                       184.
##  4 Tame Impala No Choice                                          180.
##  5 Tame Impala Borderline - Blood Orange Remix                    174.
##  6 Tame Impala Borderline - Blood Orange Remix                    174.
##  7 Tame Impala Guilty Conscience - Tame Impala Remix Extended     192.
##  8 Tame Impala Guilty Conscience - Tame Impala Remix              192.
##  9 Tame Impala Guilty Conscience - Tame Impala Remix Instrumental 192.
## 10 Tame Impala Guilty Conscience - Tame Impala Remix              192.
```

Slutligen tittade vi på grafer med ggplot2-paketet. En ggplot byggs upp med

tre grundelar: data, geometrier (grafens objekt och former), och *aesthetics* (utseende och placering av geometrierna). I ett enkelt spridningsdiagram är data två numeriska variabler, geometrierna är punkter, och punkternas placering ges av en x-koordinat och en y-koordinat. Ytterligare *aesthetics* kan vara punkternas färger (`color`) och storlek (`size`).

```
dat_small <- dat %>% filter(artist_name == "The Weeknd")                    # Skapa en m

ggplot(dat_small, aes(tempo, danceability, size = valence, color = mode_name)) +    # Koppla gra
  geom_point()                                                              # Illustrera
```



2.2 Import av data från en Excelfil

Inom vetenskapen är Excel det vanligaste filformatet för mindre datamängder. Till den här delen ska vi arbeta med data från *Gapminder*, en stiftelse som sprider information om socio-ekonomisk utveckling och global hälsa.

Uppgift 2.1 (Excelfil från Canvas)**.** Hitta excelfilen *Gapminder.xlsx* på Canvas och ladda ner den. Hitta mappen som filen laddats ned till.

I R kan man läsa in data från en Excel-fil med funktionen `read_excel()` från paketet `readxl`. Som argument till funktionen sätts filens sökväg - dess placering på hårddisken. Stycket nedan importerar från en excelfil som ligger på hårddisken *C:* i mappen *Downloads*, under *User_name*, under *Users*.

```
library(readxl)                                                 # Ladda readxl

gapminder <- read_excel("C:/Users/User_name/Downloads/Gapminder.xlsx")    # Läs in från en lokal
gapminder                                                       # Skriv ut objektet gap
```

Uppgift 2.2 (Importera från excelfil)**.** Var ligger den nedladdade filen *Gapmin-*

der.xlsx? Gör lämplig ändring i koden ovan för att läsa in data från den filen. Notera att R använder högerlutande snedstreck `/`, så om en kopierad sökväg har vänster-snedstreck måste de ändras. Kontrollera att datan blivit korrekt inläst genom att köra objektnamnet `gapminder`.

En R-session har alltid en grundmapp, ett *Working directory*. Man kan se vilken mapp det är genom att köra

```
getwd()                                    # Ange working directory
```

En filsökväg kan anges antingen som en fullständig sökväg, som ovan, eller relativt *working directory*. Om man till exempel har en fil *Gapminder.xlsx* som ligger i en mapp *Data* som i sin tur ligger i *working directory*, kan man importera data från filen med

```
gapminder <- read_excel("Data/Gapminder.xlsx")             # Läs in från en lokal exce
gapminder                                                  # Skriv ut objektet gapmind
```

```
## # A tibble: 1,704 x 6
##    country     continent  year lifeExp      pop gdpPercap
##    <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779.
##  2 Afghanistan Asia       1957    30.3  9240934      821.
##  3 Afghanistan Asia       1962    32.0 10267083      853.
##  4 Afghanistan Asia       1967    34.0 11537966      836.
##  5 Afghanistan Asia       1972    36.1 13079460      740.
##  6 Afghanistan Asia       1977    38.4 14880372      786.
##  7 Afghanistan Asia       1982    39.9 12881816      978.
##  8 Afghanistan Asia       1987    40.8 13867957      852.
##  9 Afghanistan Asia       1992    41.7 16317921      649.
## 10 Afghanistan Asia       1997    41.8 22227415      635.
## # i 1,694 more rows
```

Uppgift 2.3 (Working directory)**.** Identifiera *working directory* för din nuvarande Rs-session genom att köra `getwd()`.

RStudio har också en inbyggd funktionalitet för att importera data. Man kan hitta den genom att gå till *Environment*-fliken och sedan *Import Dataset*. Det kan vara en bra hjälp, i synnerhet om man vill sätta datatyp för någon specifik kolumn.

Om du inte har tillgång till Canvas kan Gapminder-datan alternativt hämtas från paketet Gapminder.

```
# install.packages("gapminder")          # Installera paketet gapminder (behövs ej om re
library(gapminder)                       # Ladda gapminder
gapminder                                # Skriv ut objektet gapminder
```

2.3 Ändra och skapa nya kolumner med `mutate`

Variabler kan omräknas och nya variabler kan skapas med `mutate`-funktionen. I gapminder-datan finns bnp per capita. Om man vill ha nationell BNP kan man skapa en ny kolumn och beräkna den som bnp per capita gånger populationen.

```
gapminder <- gapminder %>%              # Ta datan, och sen
  mutate(gdp = gdpPercap * pop)         # Beräkna en ny kolumn som bnp per capita (bnpPercap) gång
```

Den inledande delen med `gapminder <-` gör så att utfallet av beräkningen sparas i objektet `gapminder`. Vi kan skriva ut objektet och se resultatet av beräkningen:

```
gapminder %>% select(gdpPercap, pop, gdp)        # Ta datan och sen välj tre kolumner
```

```
## # A tibble: 1,704 x 3
##    gdpPercap      pop          gdp
##        <dbl>    <dbl>        <dbl>
##  1      779.  8425333  6567086330.
##  2      821.  9240934  7585448670.
##  3      853. 10267083  8758855797.
##  4      836. 11537966  9648014150.
##  5      740. 13079460  9678553274.
##  6      786. 14880372 11697659231.
##  7      978. 12881816 12598563401.
##  8      852. 13867957 11820990309.
##  9      649. 16317921 10595901589.
## 10      635. 22227415 14121995875.
## # i 1,694 more rows
```

Om man vill skapa en kolumn med mellanrum i namnet måste man skriva namnet inom *backticks* `\`` `för att ange att namnet ska tolkas som en enhet. Jag rekommenderar att undvika mellanrum i kolumnnamn och istället använda stora bokstäver eller understreck för ett nytt ord` (NationalGDP`eller`National_GDP‘).

```
gapminder %>%
  mutate(`National GDP` = gdpPercap * pop)
```

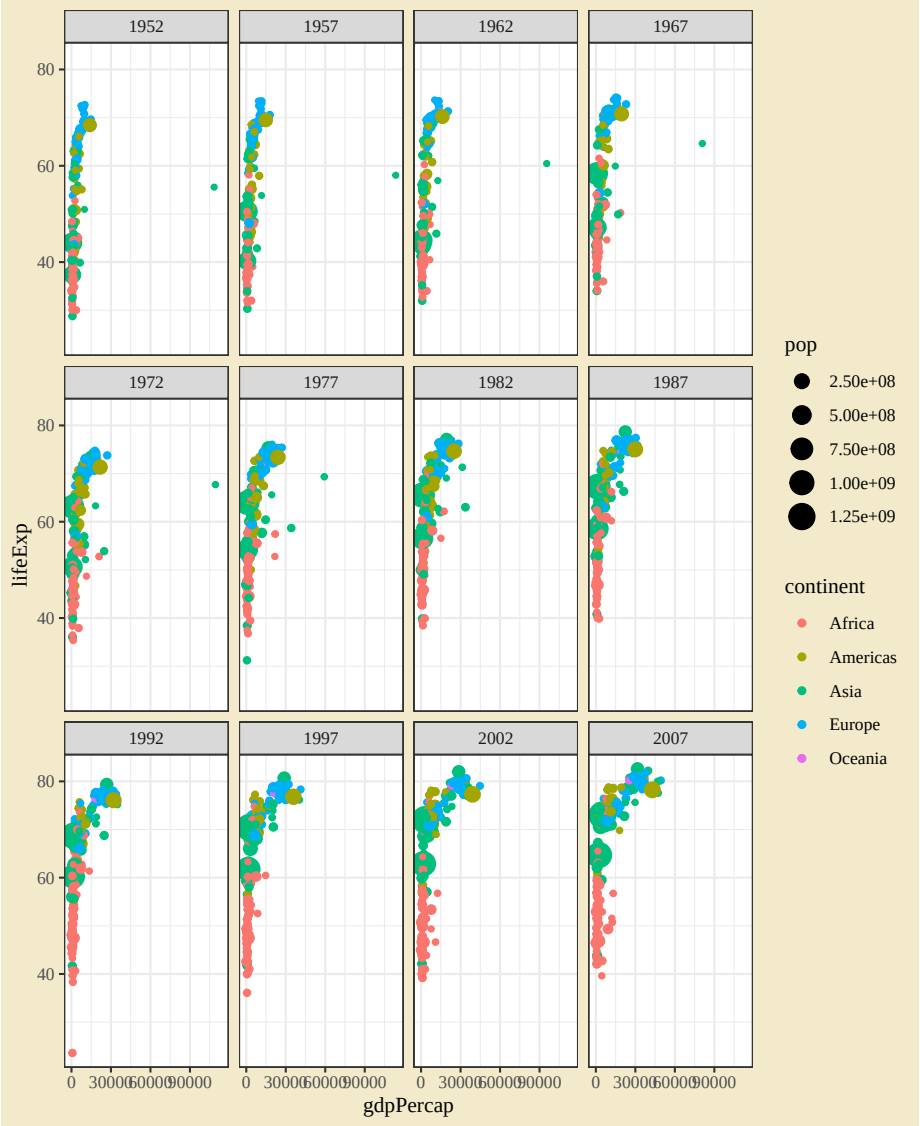
```
## # A tibble: 1,704 x 8
##    country     continent  year lifeExp      pop gdpPercap     gdp `National GDP`
##    <chr>       <chr>     <dbl>   <dbl>    <dbl>     <dbl>   <dbl>          <dbl>
##  1 Afghanistan Asia       1952    28.8  8425333      779. 6.57e 9    6567086330.
##  2 Afghanistan Asia       1957    30.3  9240934      821. 7.59e 9    7585448670.
##  3 Afghanistan Asia       1962    32.0 10267083      853. 8.76e 9    8758855797.
##  4 Afghanistan Asia       1967    34.0 11537966      836. 9.65e 9    9648014150.
##  5 Afghanistan Asia       1972    36.1 13079460      740. 9.68e 9    9678553274.
##  6 Afghanistan Asia       1977    38.4 14880372      786. 1.17e10   11697659231.
```

```
##  7 Afghanistan Asia       1982    39.9 12881816      978. 1.26e10   12598563401.
##  8 Afghanistan Asia       1987    40.8 13867957      852. 1.18e10   11820990309.
##  9 Afghanistan Asia       1992    41.7 16317921      649. 1.06e10   10595901589.
## 10 Afghanistan Asia       1997    41.8 22227415      635. 1.41e10   14121995875.
## # i 1,694 more rows
```

2.4 Sammanfattande lägesmått

Den importerade datan ger medellivslängd, populationsstorlek och bnp per capita per land och år. Vi kan börja med att producera en bubbelgraf över datan - en av de presentationer Gapminder ofta använder. En bubbelgraf är ett spridningsdiagram där punktens storlek beror på en tredje variabel.

```
ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, color = continent)) +    # Koppl
  geom_point() +                                                               # Illus
  facet_wrap(~ year)                                                           # Skapa
```



En interaktiv version kan vara bra om man vill identifiera någon specifik punkt.

```
g <- ggplot(gapminder, aes(gdpPercap, lifeExp, size = pop, color = continent, text = country)) +
  geom_point() +
  facet_wrap(~ year)

# install.packages("plotly")
library(plotly)                          # Ladda paketet plotly
ggplotly(g)                              # Ta fram en interaktiv version av grafen g
```

Under föreläsningen såg vi exempel på två lägesmått: medelvärdet (egentligen det *aritmetiska* medelvärdet) och medianen. De har bägge enkla funktioner i R: `mean()` respektive `median()`. Vi plockar ut en variabel ur datan och beräknar bägge.

```
gdpPercap <- gapminder$gdpPercap        # Skapa en vektor gdpPercap genom att ta ut kol
mean(gdpPercap)                         # Beräkna medelvärdet av gdpPercap
```

```
## [1] 7215.327
```

```
median(gdpPercap)                       # Beräkna medianen av gdpPercap
```

```
## [1] 3531.847
```

Samma sak kan göras med en pipe `%>%` och `summarise()`.

```
gapminder %>%                                   # Ta datan, och sen
  summarise(Mean = mean(gdpPercap),             # summera med medelvärdet av gdpPerca
            Median = median(gdpPercap))         # med medianen av gdpPercap
```

```
## # A tibble: 1 x 2
##    Mean Median
##   <dbl>  <dbl>
## 1 7215.  3532.
```

Uppgift 2.4 (Lägesmått av livslängd)**.** Gör lämpliga ändringar i exemplet ovan för att beräkna lägesmått för medellivslängd (`lifeExp`).

Den andra lösningen, med en pipe och `summarise()`, kan enkelt utvecklas med ett `group_by()`-steg för att beräkna medel och median per någon grupp, t.ex. per år.

```
gapminder %>%                                   # Ta datan, och sen
  group_by(year) %>%                            # gruppera efter år, och sen
  summarise(Mean = mean(gdpPercap),             # summera med medelvärdet av gdpPerca
            Median = median(gdpPercap))         # med medianen av gdpPercap
```

```
## # A tibble: 12 x 3
##     year   Mean Median
##    <dbl>  <dbl>  <dbl>
##  1  1952  3725.  1969.
##  2  1957  4299.  2173.
##  3  1962  4726.  2335.
##  4  1967  5484.  2678.
##  5  1972  6770.  3339.
##  6  1977  7313.  3799.
##  7  1982  7519.  4216.
##  8  1987  7901.  4280.
##  9  1992  8159.  4386.
## 10  1997  9090.  4782.
```

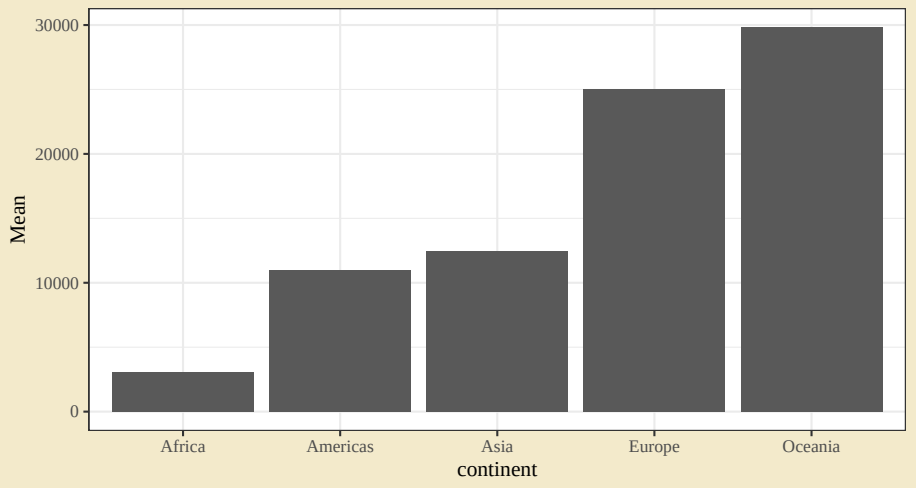
```
## 11  2002  9918.  5320.
## 12  2007 11680.  6124.
```

Uppgift 2.5 (Lägesmått per kontinent)**.** Gör lämpliga ändringar i exemplet ovan för att beräkna lägesmått per kontinent. Vad måste läggas till för att också beräkna maximum och minimum per kontinent (funktionerna `max()` och `min()`)?

Uppgift 2.6 (Upprepade mätningar)**.** Finns det några problem med att beräkna medelvärde per kontinent på den här datan? (Jag kan se minst två.)

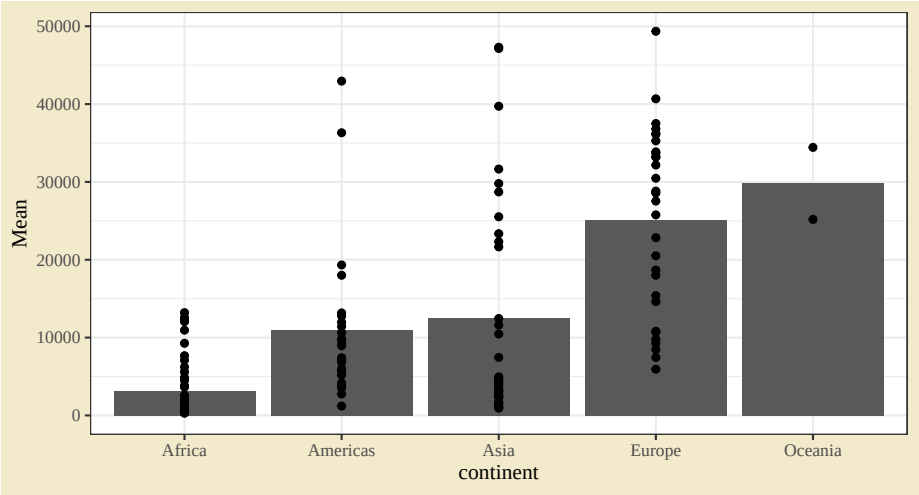
I vetenskapliga publikationer redovisas medelvärden ofta med ett stapeldiagram. Som exempel ges staplar för medelvärdet av bnp per kontinent för 2007.

```
dat_gdp_2007 <- gapminder %>%                    # Ta datan, och sen
  filter(year == 2007) %>%                       # filtrera för 2007, och sen
  group_by(continent) %>%                        # gruppera efter kontinent, och sen
  summarise(Mean = mean(gdpPercap))              # summera med medelvärdet av gdpPercap

ggplot(dat_gdp_2007, aes(continent, Mean)) +     # Skapa en ggplot med continent på x-axeln och M
  geom_col()                                     # Illustrera med kolumner (columns)
```



För att få lite mer information kan man lägga till de individuella punkterna.

```
ggplot(dat_gdp_2007, aes(continent, Mean)) +
  geom_col() +
  geom_point(aes(continent, gdpPercap), data = gapminder %>% filter(year == 2007))
```



Eftersom `geom_point()` här bygger på annan data än kolumnerna från `geom_col()` anger vi en ny `aes()`-funktionen och sätter argumentet `data`.

Uppgift 2.7 (Graf för livslängd)**.** Gör om stapeldiagrammet. Denna gång med livslängd (`lifeExp`) istället för bnp per capita (`gdpPercap`).

2.5 Sammanfattande spridningsmått

Under föreläsningarna såg vi några mått på spridning: varians, standardavvikelse och kvartilavstånd (IQR, *inter-quartile range*). De har alla motsvarande funktioner i R (`var()`, `sd()`, och `IQR()`) som kan användas på samma sätt som funktionerna för lägesmått.

```
gdpPercap <- gapminder$gdpPercap                # Skapa en vektor gdpPercap genom att

var(gdpPercap)                                  # Beräkna variansen av gdpPercap
```

```
## [1] 97169410
```

```
sd(gdpPercap)                                   # Beräkna standardavvikelsen av gdpPe
```

```
## [1] 9857.455
```

```
IQR(gdpPercap)                                  # Beräkna kvartilavståndet av gdpPerc
```

```
## [1] 8123.402
```

Alternativt med en pipe och `summarise()`.

```
gapminder %>%                                   # Ta datan, och sen
  summarise(Varians = var(gdpPercap),           # summera med varians,
            Standardavvikelse = sd(gdpPercap),  # standardavvikelse,
```

```
            Kvartilavstånd = IQR(gdpPercap))     # och kvartilavstånd
```

```
## # A tibble: 1 x 3
##     Varians Standardavvikelse Kvartilavstånd
##       <dbl>             <dbl>          <dbl>
## 1 97169410.             9857.          8123.
```

Lösningen med pipe och `summarise()` kan som tidigare utvecklas med `group_by()`.

```
gapminder %>%                                    # Ta datan, och sen
  group_by(year) %>%                             # gruppera efter år, och sen
  summarise(Varians = var(gdpPercap),            # summera med varians,
            Standardavvikelse = sd(gdpPercap),   # standardavvikelse,
            Kvartilavstånd = IQR(gdpPercap))     # och kvartilavstånd
```

```
## # A tibble: 12 x 4
##     year    Varians Standardavvikelse Kvartilavstånd
##    <dbl>      <dbl>             <dbl>          <dbl>
##  1  1952  86882249.              9321.          3049.
##  2  1957  97410232.              9870.          3946.
##  3  1962  75123173.              8667.          4650.
##  4  1967  65534132.              8095.          5925.
##  5  1972 112665135.             10614.          8252.
##  6  1977  69931225.              8362.          9847.
##  7  1982  59812359.              7734.         10985.
##  8  1987  68695607.              8288.         10667.
##  9  1992  81574244.              9032.          9414.
## 10  1997 103459275.             10171.         10656.
## 11  2002 124414278.             11154.         11950.
## 12  2007 165377988.             12860.         16384.
```

Uppgift 2.8 (Graf för livslängd)**.** Gör lämpliga ändringar i det sista exempel för att istället beräkna spridningsmått för livslängd.

Vi avslutar med tre vanliga illustrationer av vetenskaplig data - ett linjediagram med felstaplar, ett stapeldiagram med felstaplar, och ett lådagram. För linjediagrammet beräknar vi medelvärdet och spridningsmått för bnp över år och kontinent. Som spridningsmått använder vi *medelfelet*, vilket beräknas som standardavvikelse delat på roten ur antalet observationer.

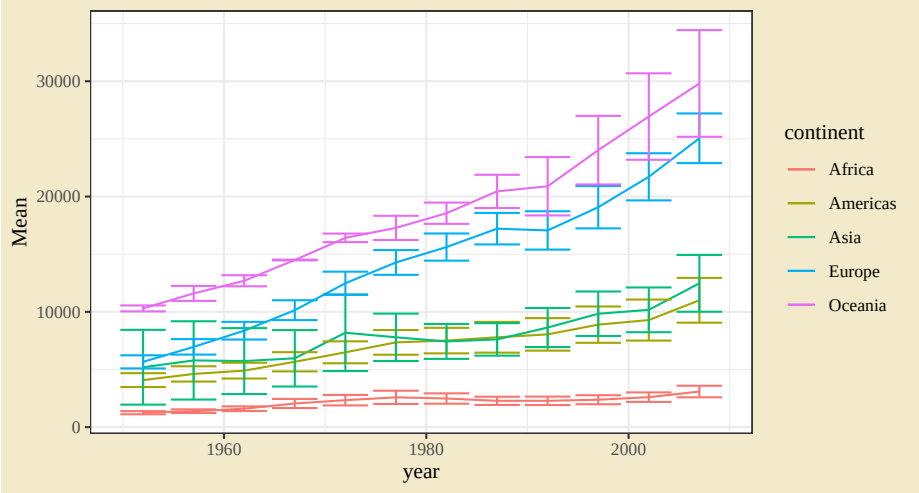
```
dat_sum <- gapminder %>%                         # Ta datan, och sen
  group_by(year, continent) %>%                  # grupper efter år och kontinent, och sen
  summarise(Mean = mean(gdpPercap),              # summera med medelvärde
            SE = sd(gdpPercap) / sqrt(n()))      # och medelfel (standardavvikelsen delat på rote
dat_sum
```

```
## # A tibble: 60 x 4
```

```
## # Groups:   year [12]
##     year continent   Mean    SE
##    <dbl> <chr>      <dbl> <dbl>
##  1  1952 Africa     1253.  136.
##  2  1952 Americas   4079.  600.
##  3  1952 Asia       5195. 3244.
##  4  1952 Europe     5661.  569.
##  5  1952 Oceania   10298.  258.
##  6  1957 Africa     1385.  157.
##  7  1957 Americas   4616.  662.
##  8  1957 Asia       5788. 3396.
##  9  1957 Europe     6963.  671.
## 10  1957 Oceania   11599.  649.
## # i 50 more rows
```

Med `ggplot2` kan vi bygga ett linjediagram med `geom_line()` och lägga till felstaplar med `geom_errorbar()`. Den senare behöver `aes()`-argument för `ymin` och `ymax` - nedre och övre del av felstapeln. Vi sätter dem till medelvärdet minus respektive plus ett medelfel.

```
ggplot(dat_sum, aes(year, Mean, color = continent)) +      # Skapa en ggplot från data
  geom_line() +                                            # Illustrera med linjer
  geom_errorbar(aes(ymin = Mean - SE, ymax = Mean + SE))   # Illustrera med felstaplar
```



Uppgift 2.9 (Bredd)**.** Felstaplarna från `geom_errorbar()` har väldigt breda ändar. Använd hjälpsidan för geomet `?geom_errorbar`, i synnerhet exemplen längst ned, och se om det går att ändra bredden.

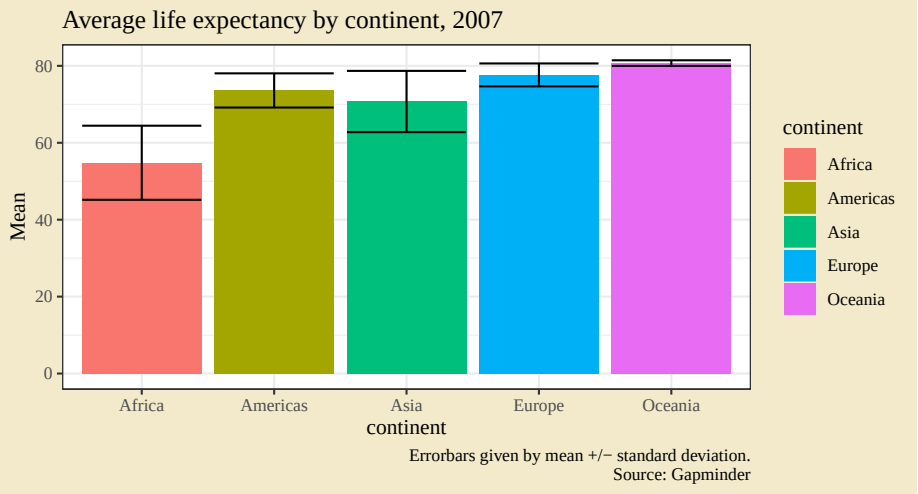
En graf med staplar och felstaplar kan konstrueras på ett likande sätt. Följande exempel visar staplar över livslängd per kontinent. Felstapeln ges av standardavvikelsen.

```
dat_sum <- gapminder %>%                          # Ta datan, och sen
  filter(year == 2007) %>%                        # filtrera på år, och sen
  group_by(continent) %>%                         # gruppera efter kontinent, och sen
  summarise(Mean = mean(lifeExp),                 # summera med medelvärde,
            SD = sd(lifeExp))                     # och standardavvikelse
dat_sum
```

```
## # A tibble: 5 x 3
##   continent  Mean    SD
##   <chr>     <dbl> <dbl>
## 1 Africa     54.8 9.63
## 2 Americas   73.6 4.44
## 3 Asia       70.7 7.96
## 4 Europe     77.6 2.98
## 5 Oceania    80.7 0.729
```

Vi bygger en ggplot med `geom_col()` och `geom_errorbar()`. Felstapels konstruktion kan anges i en notis med funktionen `labs()`.

```
ggplot(dat_sum, aes(continent, Mean, fill = continent)) +  # Skapa en ggplot med continen
  geom_col()+                                              # Illustrera med kolumner
  geom_errorbar(aes(ymin = Mean - SD, ymax = Mean + SD)) +  # Illustrera med felstaplar
  labs(title = "Average life expectancy by continent, 2007",  # Ange titel och förklarande t
       caption = "Errorbars given by mean +/- standard deviation.
       Source: Gapminder")
```

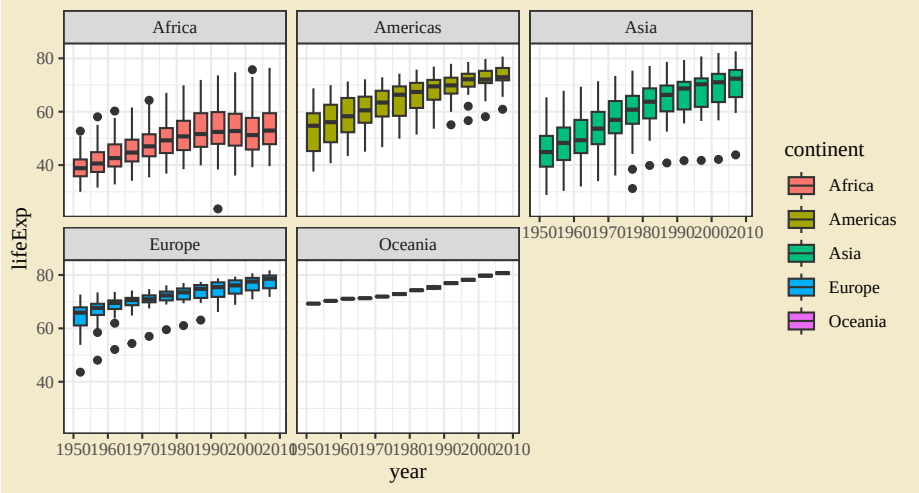


Uppgift 2.10 (Staplar för 1982)**.** Gör lämpliga ändringar i exempel ovan för att konstruera ett stapeldiagram med felstaplar för året 1982 och variabeln gdpPercap.

Ett lådagram anger fördelningen av en variabel genom att illustrera minimum,

maximum och kvartiler. Kvartiler är mått som delar en datamängd i fyra lika stora delar (så att en fjärdedel ligger under första kvartilen, en fjärdedel mellan första och andra kvartil, och så vidare). Med `ggplot2` kan vi bygga ett lådagram med `geom_boxplot()`. Exempel ger en låda per år och kontinent.

```
ggplot(gapminder, aes(year, lifeExp, fill = continent, group = year)) +
  geom_boxplot() +
  facet_wrap(~ continent)
```



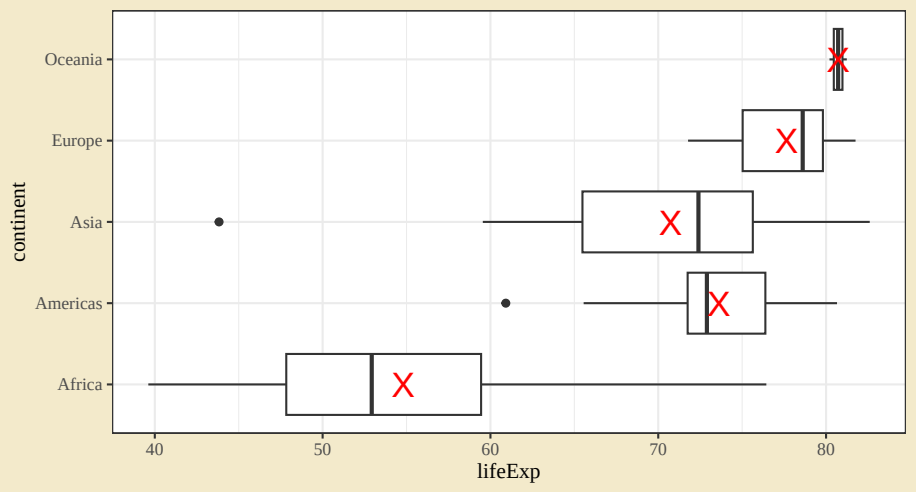
Uppgift 2.11 (Group-argumentet)**.** I lådagrammet används argumentet `group`. Vad gör det? Vad händer om man tar bort det?

Om man vill ha med både data på observationsnivån och summerad data i en graf kan man ange olika data till olika geom. Vi kan t.ex. lägga till en markör för medelvärdet i en boxplot.

```
gapminder_2007 <- gapminder %>% filter(year == 2007)
gapminder_agg <- gapminder_2007 %>%
  group_by(continent) %>%
  summarise(lifeExp = mean(lifeExp))
gapminder_agg
```

```
## # A tibble: 5 x 2
##   continent lifeExp
##   <chr>       <dbl>
## 1 Africa       54.8
## 2 Americas     73.6
## 3 Asia         70.7
## 4 Europe       77.6
## 5 Oceania      80.7
```

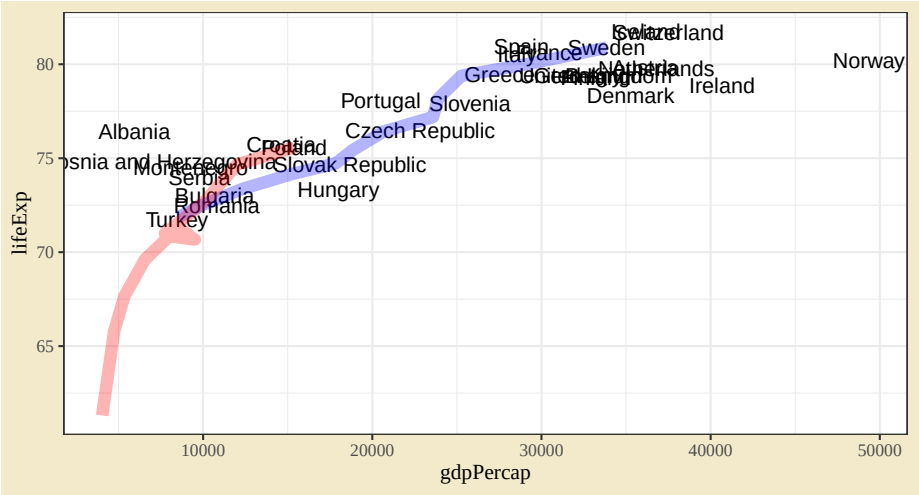
```
ggplot() +
  geom_boxplot(aes(lifeExp, continent), data = gapminder_2007) +
  geom_point(aes(lifeExp, continent), data = gapminder_agg, color = "red", shape = "X", size = 6)
```



Här byggs boxarna med datan gapminder och punkterna med den aggregerade datan.

Möjligheten att sätta data inom en `geom_()`-funktion kan blandas med `filter()` för att visa olika data i olika geom. Här ges ett spridningsdiagram över bnp och medellivslängd per land, och linjeserier (`geom_path()`) över tid för två specifika länder. Geom `geom_point()` är alltså på flera länder vid en tid, medan `geom_path()` är över tid för ett land.

```
gapminder_eu <- gapminder %>% filter(continent == "Europe")

ggplot(gapminder_eu, aes(gdpPercap, lifeExp, group = country, label = country)) +
  geom_text(data = gapminder_eu %>% filter(year == 2007)) +
  geom_path(color = "blue", size = 3, data = gapminder_eu %>% filter(country == "Sweden"), alpha
  geom_path(color = "red", size = 3, data = gapminder_eu %>% filter(country == "Poland"), alpha =
```



2.6 Ordna upp beskrivande statistik och exportera

Efter att ha beräknat någon beskrivande statistik kan det vara bra att titta på hur resultaten kan snyggas upp och exporteras i något lämpligt format. Ta den tabell med medelvärden vi producerade i ett tidigare exempel.

```
dat_sum <- gapminder %>%                          # Ta datan, och sen
  filter(year == 2007) %>%                        # filtrera på år, och sen
  group_by(continent) %>%                         # gruppera efter kontinent, och sen
  summarise(Mean = mean(lifeExp),                 # summera med medelvärde,
            SD = sd(lifeExp))                     # och standardavvikelse
dat_sum
```

```
## # A tibble: 5 x 3
##   continent  Mean    SD
##   <chr>     <dbl> <dbl>
## 1 Africa     54.8 9.63
## 2 Americas   73.6 4.44
## 3 Asia       70.7 7.96
## 4 Europe     77.6 2.98
## 5 Oceania    80.7 0.729
```

Ett vanligt skrivsätt för medelvärde och standardavvikelse är som $m \pm sd$. Vi kan använda funktionen `paste()` för att slå ihop kolumner till en sammanhängande text.

```
dat_sum %>%
  mutate(mean_plus_minus_sd = paste(Mean, "±", SD))        # Skapa en ny kolumn genom
```

```
## # A tibble: 5 x 4
##   continent  Mean    SD mean_plus_minus_sd
##   <chr>     <dbl> <dbl> <chr>
## 1 Africa     54.8 9.63  54.8060384615385 ± 9.63078067196179
## 2 Americas   73.6 4.44  73.60812 ± 4.44094763085538
## 3 Asia       70.7 7.96  70.7284848484849 ± 7.96372447069057
## 4 Europe     77.6 2.98  77.6486 ± 2.9798126601609
## 5 Oceania    80.7 0.729 80.7195 ± 0.729027091403335
```

Vi måste avrunda medelvärdet och standardavvikelsen först. Det kan man göra med `round()`, vars argument är den variabel man vill avrunda och antalet decimaler man vill avrunda till.

```
dat_sum <- dat_sum %>%
  mutate(mean_plus_minus_sd = paste(round(Mean, 1), "±", round(SD, 1)))        # Skapa en ny kolu
dat_sum
```

```
## # A tibble: 5 x 4
##   continent  Mean    SD mean_plus_minus_sd
##   <chr>     <dbl> <dbl> <chr>
## 1 Africa     54.8 9.63  54.8 ± 9.6
## 2 Americas   73.6 4.44  73.6 ± 4.4
## 3 Asia       70.7 7.96  70.7 ± 8
## 4 Europe     77.6 2.98  77.6 ± 3
## 5 Oceania    80.7 0.729 80.7 ± 0.7
```

Uppgift 2.12 (Tappade nollor)**.** Utfallet ovan är nära men inte heller riktigt vad som behövs. I de fall där funktionen avrundat till en nolla har decimal tappats. Hur kan man visa en avslutande nolla? Följande tråd på StackOverflow besvarar samma fråga.

https://stackoverflow.com/questions/42105336/how-to-round-a-number-and-make-it-show-zeros

Försök använda kod därifrån för att lägga till en avslutande nolla.

Objekt kan exporteras från R på liknande som det importeras - med särskilda exportfunktioner (*write*-funktioner) beroende på filtyp. För att exportera till en csv-fil man man anvnäda `write_csv()`. Ingående argument är det objekt man vill exportera och den fil man vill exportera till. R ger ingen varning om man skriver över en existerande fil, så var lite försiktiga här.

Precis som vid import använda R *working directory* om inget annat anges. Följande exporterar objektet `dat_sum` till en csv-fil *Exporterad data från R.csv* i *working directory*.

```
getwd()                                                    # Se nuvarande working directory
write_csv(dat_sum, "Exporterad data från R.csv")           # Skriv datan till en csv-fil.
```

Därifrån skulle man kunna öppna filen i något kalkylprogram, snygga till lay-

outen, och sedan klippa in i ett textdokument.

2.7 Kumulativt medelvärde

Om man har data som av någon anledning samlas in i sekvens kan det vara intressant att beräkna och illustrera den med ett *kumulativt medelvärde*. En serie med kumulativa medelvärden beräknas genom att för varje nytt värde ta medelvärden av de värden man hittills samlat in - vid tio värden tar man medelvärdet av de tio, vid elva värden medelvärdet av de elva, och så vidare.

Med de tärningsvärden vi hade innan kan vi beräkna ett kumulativt medelvärde genom att först beräkna summan med `cumsum()` och sedan dela på antalet kast. För att förenkla beräkningen av antalen tar vi fram sekvensen med antal kast i ett `mutate()`-steg.

```
dat_dice <- data.frame(Utfall = c(6,3,2,3,5)) %>%                # Skapa data med kolum
  mutate(Kast = 1:n())                                           # skapa en kolumn med
dat_dice
```

```
##   Utfall Kast
## 1      6    1
## 2      3    2
## 3      2    3
## 4      3    4
## 5      5    5
```

```
dat_dice <- dat_dice %>%                                         # Ta datan, och sen
  mutate(`Kumulativ summa` = cumsum(Utfall),                     # skapa en kolumn som
         `Kumulativt medelvärde` = `Kumulativ summa` / Kast)     # dela den kumulativa
dat_dice
```

```
##   Utfall Kast Kumulativ summa Kumulativt medelvärde
## 1      6    1               6              6.000000
## 2      3    2               9              4.500000
## 3      2    3              11              3.666667
## 4      3    4              14              3.500000
## 5      5    5              19              3.800000
```

Om beräkning inte är uppenbar här, ta någon minut för att förstå den.

Uppgift 2.13 (Kumulativt medelvärde)**.** Vad ska läggas till för att stycket nedan ska ge en linjegraf över medelvärdet?

```
ggplot(dat_dice, aes(x = Kast, y = ___)) +
  ___()
```

Uppgift 2.14 (Fler tärningskast)**.** Kasta din tärning ytterligare några gånger, gärna på en mjuk yta. Fyll i dina utfall och gör grafen från föregående uppgift. Kan man se en tendens för medelvärdet att minska i varians vid fler kast?

```
dat_dice <- data.frame(Utfall = c(___)) %>%
  mutate(Kast = 1:n(),                                    # Lägg till värden från 1 till n
         `Kumulativ summa` = cumsum(Utfall),              # Beräkna kumulativ summa av utfa
         `Kumulativt medelvärde` = `Kumulativ summa` / Kast)    # Dela summan på antal för relati
dat_dice
```

Uppgift 2.15 (Kumulativ frekvens)**.** Om man vill titta på andelen gånger ett visst utfall inträffat talar man om *kumulativ frekvens* snarare än *kumulativt medelvärde*. Använd stycket nedan för att titta på andelen gånger utfallet varit en etta (ett *positivt* utfall, i begreppets kliniska mening). Om ett inte är ett möjligt utfall på din tärning, ändra ettan till något mer lämpligt.

```
dat_dice <- data.frame(Utfall = c(___)) %>%
  mutate(Kast = 1:n(),
         `Positivt utfall` = Utfall == 1,                 # Skapa en variabel som anger om
         `Kumulativt antal` = cumsum(`Positivt utfall`),
         `Kumulativ frekvens` = `Kumulativt antal` / Kast)
dat_dice

ggplot(dat_dice, aes(x = Kast, y = `Kumulativ frekvens`)) +
  geom_line()
```

2.8 Darwin-exempel

På kursen Canvas-sida finns en excelfil *Uppgiftdata.xlsx* som innehåller data för de flesta räkneupgifter på kursen. I den här delen ska vi titta på datan i fliken *Darwin* som innehåller en jämförelse i planthöjd mellan kors- och självbefruktade plantor.

Uppgift 2.16 (Ladda ner uppgiftsdata)**.** Ladda ner filen med uppgiftsdata till din lokala hårddisk.

Vi såg tidigare hur en excelfil kan läsas in med `read_excel()`. Ett argument den funktionen kan ta är `sheet`, som styr vilken flik som ska läsas in. Som tidigare måste man ange var på datorn excel-filen ligger.

Uppgift 2.17 (Läs in Darwin-datan)**.** Gör lämpliga ändringar i koden nedan för att läsa in fliken *Darwin*.

```
dat_darwin <- read_excel("Data/Uppgiftsdata.xlsx", sheet = "___")      # Läs in data från flik
dat_darwin %>% print(n = 30)                                           # Skriv ut datan (högst
```

När data är inläst kan man sammanfatta den med medelvärde, standardavvikelse och medelfel (där medelfelet ges av standardavvikelsen delad på roten ur antalet observationer).

Uppgift 2.18 (Sammanfatta Darwin-datan)**.** Fyll i koden nedan för att beräkna

medelvärde, standardavvikelse, antal observationer och medelfel. Gör beräkningen per grupp (`Metod`)

```
dat_sum <- dat_darwin %>%                                          # Ta datan,
  group_by(___) %>%                                                # gruppera e
  summarise(Medelvärde = mean(___),                                # beräkna me
            Standardavvikelse = ___(Utfall),                       # beräkna st
            `Antal observationer` = n(),                           # beräkna an
            Medelfel = Standardavvikelse / sqrt(`Antal observationer`))   # beräkna me
dat_sum
```

Slutligen kan vi presentera de sammanfattande måtten med en lämplig graf. Ett vanligt val är ett stapeldiagram med felstaplar för medelfelen.

Uppgift 2.19 (Illustrera Darwin-datan)**.** Fyll i koden nedan för att skapa ett stapeldiagram med felstaplar av de sammanfattande måtten i objektet som skapades i uppgiften ovan. Felstaplarna styrs med argumenten `ymin` och `ymax`. Dess ska sättas till medelvärdet minus ett medelfel respektive medelvärdet plus ett medelfel. Välj lämpliga värden för staplarnas bredd. Välj en lämplig färg för staplarnas kant och inre del. Se `colors()` för en lista över färger, eller använd hex-koder som `"#ff00ff"`.

```
ggplot(dat_sum, aes(Metod, Medelvärde)) +
  geom_col(color = ___, fill = ___, width = ___) +
  geom_errorbar(aes(ymin = ___ - ___, ymax = ___ + ___),
                width = ___) +
  labs(caption = "Felstapel anger medelvärde ± ett medelfel")
```

Ett annat alternativ för en graf ett lådagram per grupp. Här används den ursprungliga datan, snarare än beräknad beskrivande statistik.

Uppgift 2.20 (Illustrera Darwin-datan)**.** Fyll i koden nedan för att skapa ett lådagram för de två metoderna. Låt x-axeln ange planthöjden (`Utfall`) och y-axeln metoden (`Metod`). Även här kan man styra färger och bredd med `color`, `fill` och `width`.

```
ggplot(dat_darwin, aes(x = ___, y = ___)) +
  geom_boxplot(color = "purple", fill = "hotpink", width = 0.2)
```

Funktionen `quantile()` kan användas för att beräkna kvartiler. Notera att man första måste dela per grupp.

```
dat_darwin_kors <- dat_darwin %>% filter(Metod == "Korsbefruktade")
quantile(dat_darwin_kors$Utfall)

dat_darwin_själv <- dat_darwin %>% filter(Metod == "Självbefruktade")
quantile(dat_darwin_själv$Utfall)
```

Vad är kopplingen mellan kvantilerna och lådagrammet?

2.9 Bonus. Tredimensionella grafer med `plotly`

Förra gången använde vi paketet `plotly` för att göra en interaktiv graf. Paketet har också funktioner för 3d-grafer. Börja med att ladda paketet.

```
library(plotly)
```

Vi börjar med ett enkelt exempel på en 3d-graf med lite skapad data.

```
dat_ex <- data.frame(Var1 = c(1,2,3), Var2 = c(3,1,2), Var3 = c(2,3,1), Type = c("A", "B", "C"))
dat_ex

plot_ly(dat_ex, x = ~Var1, y = ~Var2, z = ~Var3, color = ~Type) %>%
  add_markers()
```

Om grafen inte kommer upp direkt kan det fungera att trycka på den lilla ikonen med ett fönster och en pil i *Viewer*-fliken. Grafen ska då öppnas i en webbläsare.

Syntaxen till `plot_ly()` är inte helt olik `ggplot()`. Först anges datan, därefter argument för x- y-, och z-koordinater. Notera tilde-tecknet `~` före variabelnamnen. Eventuell färg sätts med `color`. Efter det lägger man till punkter (här *markers*) med en pipe in i `add_markers()`. Vi vill göra en liknande graf med gapminder-datan, men får börja med att filtrera på ett visst år.

Uppgift 2.21 (Filtrera för år)**.** Vad måste läggas till i funktionen nedan för att filtrera för data där året är 2007?

```
dat_2007 <- gapminder %>%
  ___(year == ___)
```

Vi kan nu konstruera en 3d-graf med datan.

Uppgift 2.22 (Gapminder i 3d)**.** Vad måste läggas till i funktionen nedan för en 3d-graf med befolkningsmängd (`pop`) på x-axeln, livslängd (`lifeExp`) på y-axeln, bnp per capita (`gdpPercap`) på z-axeln, och färg efter kontinent (`continent`)? För att kunna identifiera specifika länder kan man också sätta argumentet `text`.

```
plot_ly(___, x = ~___, y = ~___, z = ~___, color = ~___, text = ~country) %>%
  add_markers()
```

Uppgift 2.23 (Log-transformationer)**.** Inom statistiken är det vanligt att transformera variabler för att ta bort extremeffekter och visa på specifika dataegenskaper. En vanlig transform är att *logaritmera* ett värde, vilket innebär att man istället för att använda det ursprungliga värdet använder exponenten i någon bas (ofta basen tio). Ta till exempel värdet 10000, dess tio-logaritm är 4, eftersom 10 upphöjt i 4 är 10000. Logaritmer är vanliga vid data med extremvärden.

Grafen i uppgiften ovan präglas mycket av skillnader i bnp och befolkningsstorlek. Testa att tio-logaritmera variablerna och se om det blir en mer eller mindre överskådlig graf. Logaritmen kan göras genom att byta den ursprungliga variabeln mot en variabel transformerad med `log10()`. Fyll i stycket nedan.

```
plot_ly(___, x = ~log10(___), y = ~log10(___), z = ~___, color = ~___, text = ~country)
  add_markers()
```

Uppgift 2.24 (Följa ett land)**.** Likt en ggplot kan man lägga till graf-element. Här använder man dock en pipe för lägga till ett nytt element. Fyll i kodstycket nedan. Vad, om något, har lagts till i grafen?

```
plot_ly(___, x = ~log10(___), y = ~log10(___), z = ~___, color = ~___, text = ~country)
  add_markers() %>%
  add_lines(data = gapminder %>% filter(country == "Costa Rica"))
```

Uppgift 2.25 (Spotify 3d)**.** Som avslutning återvänder vi till spotify-datan från datorövning 1. Fyll i stycket nedan för att skapa en graf med tempo, dansbarhet och valens (`tempo`, `danceability`, `valence`) på axlarna, storlek efter energi (`energy`) och text efter spårnamn (`track_name`). Filtrera på valfri artist. Använd `unique(dat_spot$artist_name)` för att se tillgängliga artister.

```
dat_spot <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2022/main/Data/S
dat_small <- dat_spot %>% filter(artist_name == "Weyes Blood")

plot_ly(dat_small, x = ~___, y = ~___, z = ~___, size = ~___, text = ~___) %>%
  add_markers()
```

Chapter 3

Slumpvariabler

Datorövning 3 handlar om sannolikhetslära i R. Efter övningen ska vi kunna

- Identifiera en passande slumpfördelning för verkliga fenomen,
- Beräkna sannolikheter från en antagen fördelning,
- Simulera fenomen med en antagen fördelning,
- Använda slumptal för att utforska teoretiska resultat från sannolikhetsläran.

3.1 Repetition från datorövning 2

När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 2 tittade vi på hur insamlade variabler kan sammanfattas med lägesmått och spridningsmått. Ett enkelt sätt att ta fram dem är att använda `summarise()` och ange de mått och variabler man vill använda. Vi hade uppe ett exempel på data från Gapminder som vi importerade från en excel-fil. För nu kan vi dock hämta datan från paketet `gapminder`.

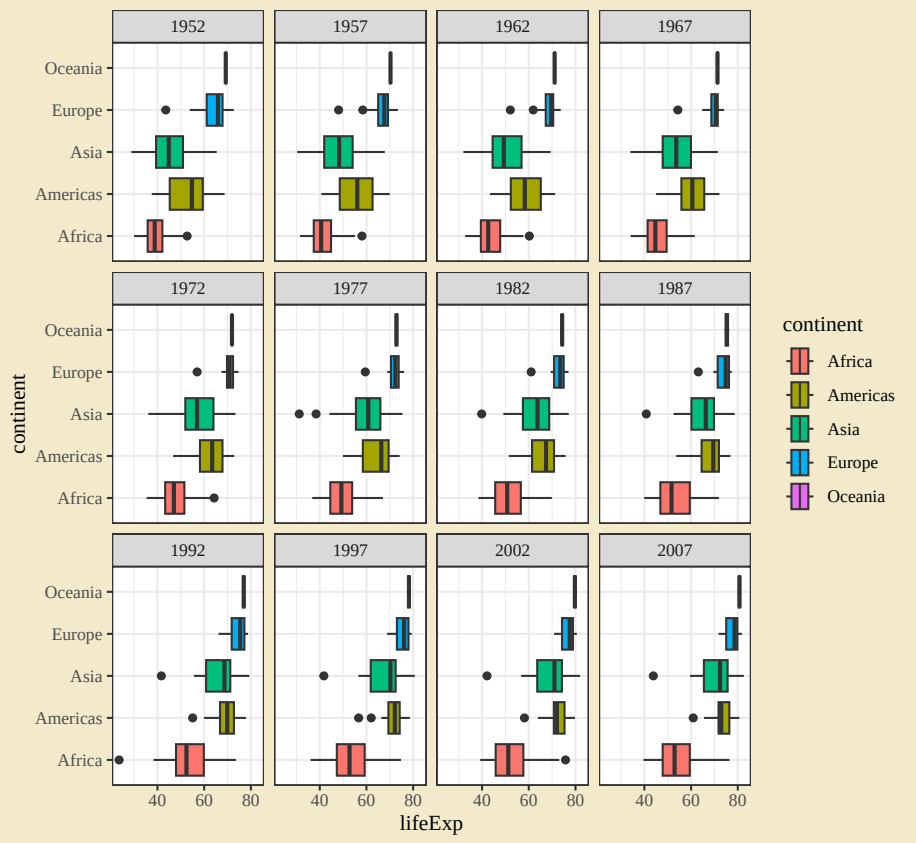
```
# install.packages("gapminder")
library(gapminder)

gapminder %>%
  filter(year == 2007) %>%
  group_by(continent) %>%
```

```
  summarise(`Livslängd, medel` = mean(lifeExp),
            `Befolkning, median` = median(pop),
            `Bnp per capita, standardavvikelse` = sd(gdpPercap))
```

```
## # A tibble: 5 x 4
##   continent `Livslängd, medel` `Befolkning, median` Bnp per capita, standardav~1
##   <chr>                  <dbl>                <dbl>                         <dbl>
## 1 Africa                  54.8             10093310.                         3618.
## 2 Americas                73.6              9319622                          9713.
## 3 Asia                    70.7             24821286                         14155.
## 4 Europe                  77.6              9493598                         11800.
## 5 Oceania                 80.7             12274974.                         6541.
## # i abbreviated name: 1: `Bnp per capita, standardavvikelse`
```

Beskrivande mått sammanfattas ofta i någon enkel vetenskaplig graf. Två vanliga val är lådagrammet, som illustrerar kvartiler och möjliga extremvärden, och stapeldiagrammet med felstaplar. Vi ger först ett exempel på ett lådagram över livslängd per kontinent uppdelat efter år.

```
ggplot(gapminder, aes(lifeExp, continent, fill = continent)) +
  geom_boxplot() +
  facet_wrap(~ year)
```



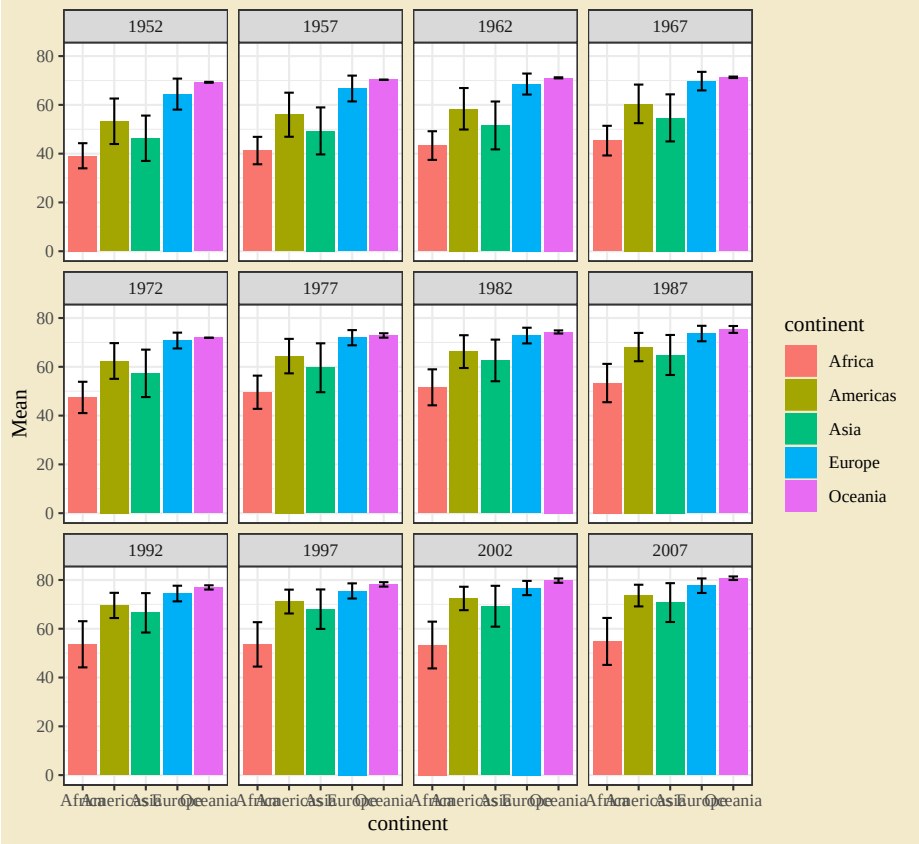
Därefter ett exempel på ett stapeldiagram med felstaplar för samma data. Felstapeln ges av standardavvikelsen.

```
dat_sum <- gapminder %>%
  group_by(continent, year) %>%
  summarise(Mean = mean(lifeExp),
            SD = sd(lifeExp))
dat_sum
```

```
## # A tibble: 60 x 4
## # Groups:   continent [5]
##    continent  year  Mean    SD
##    <chr>     <dbl> <dbl> <dbl>
##  1 Africa     1952  39.1  5.15
##  2 Africa     1957  41.3  5.62
##  3 Africa     1962  43.3  5.88
##  4 Africa     1967  45.3  6.08
##  5 Africa     1972  47.5  6.42
##  6 Africa     1977  49.6  6.81
```

```
##  7 Africa     1982  51.6  7.38
##  8 Africa     1987  53.3  7.86
##  9 Africa     1992  53.6  9.46
## 10 Africa     1997  53.6  9.10
## # i 50 more rows
```

```
ggplot(dat_sum, aes(continent, Mean, fill = continent)) +
  geom_col() +
  geom_errorbar(aes(ymin = Mean - SD, ymax = Mean + SD), width = 0.3) +
  facet_wrap(~ year)
```



3.2 Diskreta fördelningar i allmänhet

En diskret slumpvariabel kan bara anta specifika värden på tallinjen. Det absolut vanligaste fallet är när utfallen är heltal. Varje utfall är kopplat till *en* sannolikhet och summan av sannolikheterna är ett.

Ett kast med en 6-sidig tärning har sex möjliga utfall och samma sannolikhet för varje utfall. Eftersom summan av sannolikheter ska bli ett måste sannolikheten

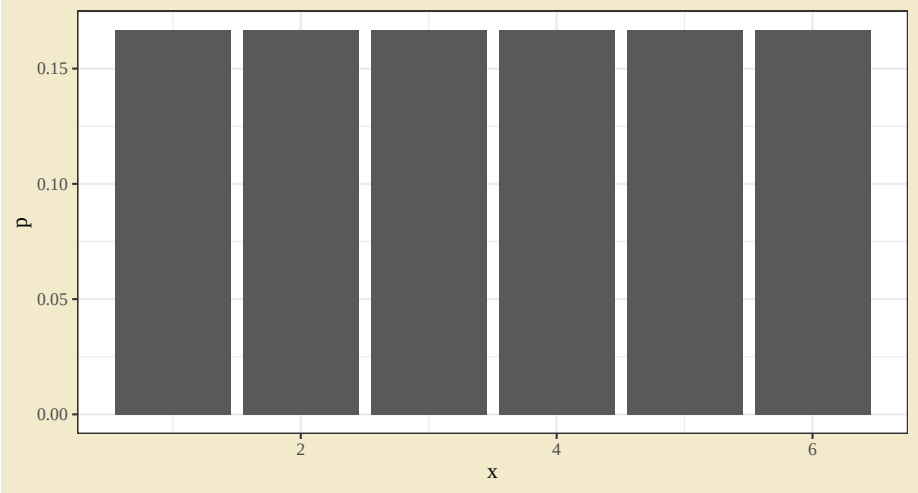
för ett specifik utfall vara en sjättedel. Vi kan kodifiera slumpvariabeln i en tabell över utfall och sannolikhet.

```
dat_dice6 <- data.frame(x = c(1,2,3,4,5,6),
                        p = 1/6)
dat_dice6
```

```
##   x         p
## 1 1 0.1666667
## 2 2 0.1666667
## 3 3 0.1666667
## 4 4 0.1666667
## 5 5 0.1666667
## 6 6 0.1666667
```

En diskret slumpvariabel illustreras ofta med ett stapeldiagram.

```
ggplot(dat_dice6, aes(x, p)) + geom_col()
```



Uppgift 3.1 (En annan tärning)**.** Ta fram en tärning som *inte* har sex sidor. Följ exemplet ovan för att kodifiera din tärning som en slumpvariabel. Ett bra namn på det nya objektet kan vara `dat_diceN` där N anger antalet sidor på tärningen, t.ex. `dat_dice20` för en 20-sidig tärning.

Under tidigare datorövningar såg vi hur man kan beräkna medelvärde och varians från ett stickprov. De måtten kan också beräknas på en teoretisk slumpvariabel. Den här kopplingen mellan beräkningar på ett stickprov och teoretiska egenskaper hos en slumpvariabel är väldigt central inom statistiken, eftersom vårt mål är att koppla verklig data till teoretiska modeller.

För en diskret slumpvariabel ges medelvärdet (som då också kallas *populationsmedelvärde*, eller vanligare *väntevärde*) av summan av utfallen gånger sanno-

likheterna. För vår tärning kan vi räkna ut det genom att multiplicera utfall och sannolikhet i ett `mutate()` steg och sedan summera.

```
dat_dice6 %>%
  mutate(x_times_p = x * p) %>%
  summarise(Expected_value = sum(x_times_p))
```

```
##   Expected_value
## 1            3.5
```

Notera att medelvärdet inte behöver vara ett möjligt utfall.

Uppgift 3.2 (Medelvärde för din tärning)**.** Upprepa beräkningen ovan, denna gång med den slumpvariabel du kodifierade i den tidigare uppgiften.

Beräkningen av en teoretisk varians (som vi kan kalla *populationsvarians*) är lite mer komplicerad och vi går inte in på några detaljer här. Koden nedan skapar en funktion som beräknar populationsvariansen.

```
pop_var <- function(x, p) sum(p * (x - sum(x * p))^2)
```

För en 6-sidig tärning kan vi beräkna variansen med följande.

```
dat_dice6 %>%
  summarise(varians = pop_var(x, p))
```

```
##    varians
## 1 2.916667
```

Uppgift 3.3 (Varians för din tärning)**.** Upprepa beräkningen ovan, denna gång med den slumpvariabel du kodifierade i den tidigare uppgiften.

Vår modell för tärningskastet säger att en serie tärningskast ska ha ett visst medelvärde och varians. Låt oss nu testa det genom att kasta tärning.

Uppgift 3.4 (Kasta tärningen)**.** Kasta din tärning 20 gånger. Gärna på en mjuk yta. Skriv in utfallen i koden nedan och beräkna medelvärde och varians.

```
utfall <- c(_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_,_)
mean(utfall)
var(utfall)
sd(utfall)
```

Ligger medelvärde och varians från stickprovet nära de teoretiska beräkningarna?

R kommer också med en rad funktioner för att simulera data. Funktionen `sample()` drar ett urval ur en serie möjliga utfall. Följande kod drar tiotusen slumputfall från en 6-sidig tärning, och beräknar medelvärde och varians.

```
slumputfall <- sample(x = dat_dice6$x, size = 10000, replace = T)

mean(slumputfall)
```

```
## [1] 3.5059
```

```
var(slumputfall)
```

```
## [1] 2.867652
```

Uppgift 3.5 (Simulera tärningen)**.** Använd `sample()` för att dra tiotusen observationer från din egen tärning. Beräkna medelvärde och varians från det stickprovet. Är utfallen nära de teoretiska beräkningarna av populationsmedelvärde och -varians?

3.3 Särskilda diskreta fördelningar: binomialfördelning

Under föreläsningen såg vi tre familjer av fördelningar som uppstår i vissa specifika situationer:

- binomialfördelningen (för antalet positiva utfall vid ett antal försök),
- poissonfördelningen (för antalet händelser vid en stor mängd försök), och
- normalfördelningen (för kontinuerliga variabler).

Binomialfördelningen ger sannolikheter för antalet positiva utfall vid ett visst antal försök. Varje försök har ett av två utfall (positivt och negativt) och försöken är oberoende av varandra. I en binomialfördelning ges antalet försök av en parameter n och sannolikheten för ett positivt utfall i ett försök av en parameter p.

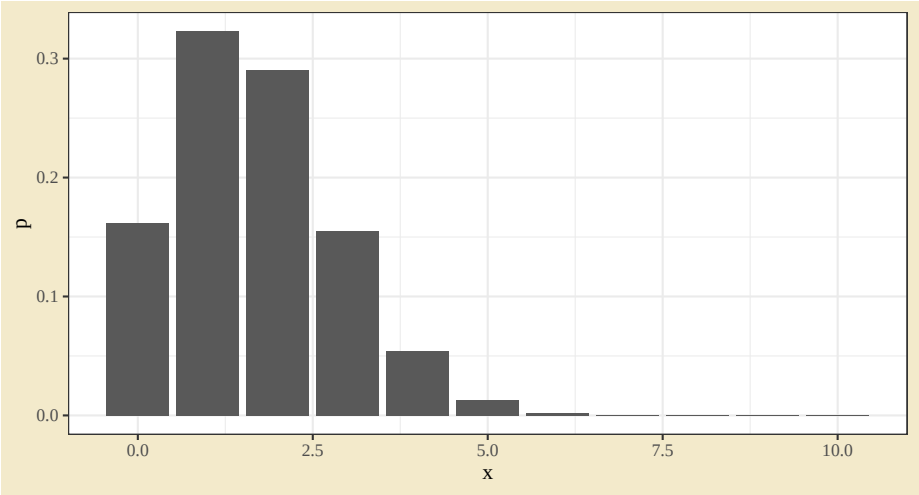
Som exempel kan vi ta ett tärningskast. Om man kastar en 6-sidig tärning tio gånger kommer antalet ettor (vårt positiva utfall) följa en binomialfördelning där n ges av tio och p av en sjättedel. I R kan sannolikheter från en binomialfördelning tas fram med `dbinom()`. Sannolikheten för tre ettor ges till exempel av

```
dbinom(3, size = 10, prob = 1/6)
```

```
## [1] 0.1550454
```

Om vi vill illustrera en binomalfördelning kan vi ta fram sannolikheterna för varje utfall och göra ett stapeldiagram.

```
dat_bin <- data.frame(x = 0:10) %>%
  mutate(p = dbinom(x, size = 10, prob = 1/6))

ggplot(dat_bin, aes(x, p)) + geom_col()
```



Uppgift 3.6 (Binomialfördelning för tärningen)**.** Ta tärningen från tidigare uppgift. Om man kastar tärningen tjugo gånger, vad är fördelningen för antalet gånger man får tärningens lägsta utfall? (Till exempel, vad är fördelningen för antalet ettor vid tjugo kast med en sexsidig tärning?) Fyll i stycket nedan för att beräkna sannolikheterna i den fördelningen och illustrera med ett stapeldiagram.

```
dat_bin <- data.frame(x = 0:20) %>%
  mutate(p = dbinom(x, size = 20, prob = ___))

ggplot(dat_bin, aes(x, p)) + geom_col()
```

Uppgift 3.7 (Sannolikheter i binomialen)**.** I den fördelning du beräknade i uppgiften ovan. Vad är sannolikheten att få exakt tre positiva utfall? (Ledning: för en sexsidig tärning skulle det ges av `dbinom(3, size = 20, prob = 1/6)`).)

Utöver sannolikhetsfunktionen (som ger sannolikheten för varje utfall) används ofta också fördelningsfunktionen (som för varje utfall ger sannolikheten att ligga exakt på eller under ett specifikt värde). Fördelningsfunktionen används vid sannolikhetsberäkningar, t.ex. är de tabeller för fördelningar man ofta ser i slutet av statistikböcker på fördelningsformen.

I R beräknas fördelningsfunktionen med `pbinom()`. För fallet med tio tärningskast kan man ta

```
dat_bin <- data.frame(x = 0:10) %>%
  mutate(p = dbinom(x, size = 10, prob = 1/6),
         P = pbinom(x, size = 10, prob = 1/6))
dat_bin
```

```
##    x           p         P
## 1  0 1.615056e-01 0.1615056
```

```
## 2   1 3.230112e-01 0.4845167
## 3   2 2.907100e-01 0.7752268
## 4   3 1.550454e-01 0.9302722
## 5   4 5.426588e-02 0.9845380
## 6   5 1.302381e-02 0.9975618
## 7   6 2.170635e-03 0.9997325
## 8   7 2.480726e-04 0.9999806
## 9   8 1.860544e-05 0.9999992
## 10  9 8.269086e-07 1.0000000
## 11 10 1.653817e-08 1.0000000
```

Fördelningsfunktionen ges av att summera sannolikhetsfunktionen uppifrån.

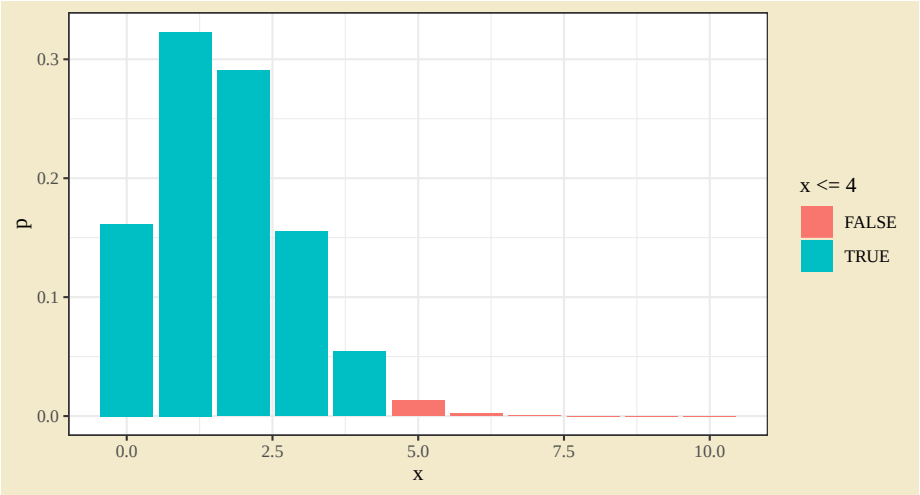
Uppgift 3.8 (Fördelningsfunktion i binomialen)**.** Använd binomialfördelningen från den tidigare uppgiften till att beräkna sannolikheten att få *tre eller färre* positiva utfall.

Vi kan använda ett logiskt uttryck för att illustrera sannolikheter i stapeldiagrammet. Om vi vill beräkna och illustrera sannolikheten för fyra eller färre positiva utfall i exemplet med en tärning kan vi använda följande.

```
pbinom(4, size = 10, prob = 1/6)
```

```
## [1] 0.984538
```

```
dat_bin <- data.frame(x = 0:10) %>%
  mutate(p = dbinom(x, size = 10, prob = 1/6),
         P = pbinom(x, size = 10, prob = 1/6))

ggplot(dat_bin, aes(x, p, fill = x <= 4)) +
  geom_col()
```



Den blå ytan motsvarar sannolikheten att få fyra eller färre ettor. Den sannolikheten kan beräknas till 0.985.

Uppgift 3.9 (Illustration av sannolikheten)**.** Gör lämpliga ändringar i exemplet ovan för att illustrera sannolikheten att få exakt tre positiva utfall i exemplet med tärningskastet.

Uppgift 3.10 (Teori och verklighet)**.** I en tidigare uppgift kastade du tärningen tjugo gånger. Hur många gånger fick du det lägsta möjliga utfallet på tärningen (t.ex. en etta på en vanlig sex-sidig tärning)?

I det första exemplet med tärningen kunde vi beräkna väntevärde och varians med de allmänna formlerna. För parametriserade fördelningar som binomialfördelningen finns ofta enklare formler som helt beror på parametrarna. I en binomial ges populationsmedelvärdet av antalet upprepningar gånger sannolikheten (n gånger p) och variansen av antalet upprepningar gånger sannolikheten gånger ett-minus-sannolikheten (n gånger p gånger (1 - p)). För en sexsidig tärning (n = 10 och p = 1/6) ges medelvärdet av 1.667 och variansen av 1.389.

Uppgift 3.11 (Medelvärde och varians i binomialen)**.** Beräkna medelvärde och varians för antalet ettor om man kastar en 16-sidig tärning tjugo gånger.

Uppgift 3.12 (Frösort)**.** För en viss frösort är sannolikheten att gro 60%. Om man sår 10 slumpmässigt valda frön, hur stor är då chansen att

a. högst 6 frön gror?
b. minst 6 frön gror?
c. Beräkna populationsmedelvärde och populationsvrians för antalet frön som gror.

(Denna uppgift finns också i uppgiftsdokumentet och kan lösas för hand.)

3.4 Särskilda diskreta fördelningar: poissonfördelning

En poissonfördelning är en vanlig sannolikhetsfördelning för antalsdata (alltså data som antar positiva heltal som 0, 1, 2, 3 och så vidare). Poissonfördelningen har en nära koppling till binomialfördelningen: om antalet försök n är stort och sannolikheten p liten liknar en binomial fördelning en poissonfördelning. Det innebär att en poissonfördelning är en lämplig fördelning för händelser som har många upprepningar men låg sannolikhet att inträffa i det enskilda försöket. Typexempel är olika typer av olyckor eller observationsantal för sällsynta växter och djur.

Poissonfördelningen styrs av en enda parameter, lambda. Till skillnad från en binomialfördelning, där det högsta möjliga utfall ges av n, har en poissonfördelning inget maxvärde. Man kan i teorin få vilket positivt heltalsutfall som helst.

För en poissonfördelning är populationsmedelvärdet och populationsvariansen lika med lambda.

I R kan man beräkna sannolikheter för en poissonfördelning med `dpois()` och `ppois()`.

Uppgift 3.13 (Sannolikhetsfunktionen för en poisson)**.** Följande ger sannolikheten att få utfall 2 i en poissonfördelning med lambda satt till 4.

```
dpois(2, lambda = 4)
```

```
## [1] 0.1465251
```

Gör lämpliga ändringar för att beräkna sannolikheten för exakt 5 i en fördelning med lambda satt till 3.

Uppgift 3.14 (Fördelningsfunktionen för en poisson)**.** Följande ger sannolikheten att få mindre än eller lika med 2 i en poissonfördelning med lambda satt till 4.

```
ppois(2, lambda = 4)
```

```
## [1] 0.2381033
```

Gör lämpliga ändringar för att beräkna sannolikheten för *mindre än eller lika med 5* i en fördelning med lambda satt till 3. Hur kan man beräkna sannolikheten att få mer än 5 i en fördelning med lambda satt till 3?

Uppgift 3.15 (Albinofödsel)**.** Sannolikheten att en viss individ i en viss population skall vara albino är 1/20 000. Hur stor är sannolikheten att på 40 000 födslar

a. ingen albino föds,
b. minst en albino föds.
c. Vilka antaganden skall vara uppfyllda för att sannolikheterna i a och b ska gälla?

(Denna uppgift finns också i uppgiftsdokumentet och kan lösas för hand.)

3.5 Kontinuerliga fördelningar i allmänhet

En kontinuerlig fördelning kan anta vilka värden som helst på hela tallinjen eller i något intervall på tallinjen. Ett enkelt exempel på en kontinuerlig slumpvariabel kan vara att stoppa ett stoppur slumpmässigt och titta på decimaldelen. Det kommer ge något värde mellan 0 och 1. (Stoppuret kommer naturligtvis avrunda värdet, så man får tänka sig ett magiskt stoppur med oändligt antal decimaler.) Till skillnad från en diskret slumpvariabel, som kan beskrivas med utfallen och dess sannolikheter, måste en kontinuerlig variabel förklaras med en matematisk funktion, en så kallad *täthetsfunktion*. Detta beror på att enskilda utfall alltid har sannolikhet noll för en kontinuerlig fördelning: om man

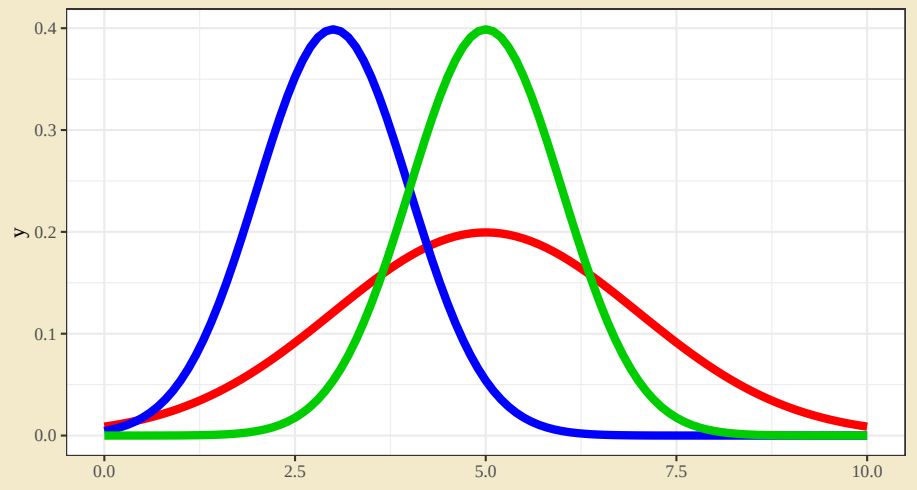
har oändligt antal decimaler är det som exempel sannolikhet noll att få exakt 0.345000… som decimaler på ett stoppur.

Den absolut vanligaste kontinuerliga fördelningen är en *normalfördelning*.

3.6 Särskilda kontinuerliga fördelningar: normalfördelningen

En normalfördelning är en kontinuerlig fördelning some kan anta värden över hela tallinjen och beror på två parametrar: mu som styr var fördelningens är centrerad, och sigma som styr hur utspridd fördelningen är. En normalfördelning har en karaktäristisk *klockform*. Vi illustrerar två normalfördelningar med hjälp av geomet `geom_function()` i en ggplot.

```
ggplot() +
  geom_function(fun = dnorm, args = list(mean = 5, sd = 2), color = "red", size = 2) +
  geom_function(fun = dnorm, args = list(mean = 3, sd = 1), color = "blue", size = 2) +
  geom_function(fun = dnorm, args = list(mean = 5, sd = 1), color = "green3", size = 2)
  xlim(0, 10)
```



Den gröna och röda kurvan har samma medelvärde mu men skilda standardavvikelser sigma. Den blå och gröna kurvan har skilda medelvärden mu men samma standardavvikelse sigma.

Uppgift 3.16 (Normalfördelningar)**.** Gör lämpliga ändringar i stycken ovan för att illustrera två normalfördelningar: en med medelvärde 0 och standardavvikelse 1 och en med medelvärde 1 och standardavvikelse 2. Kan du utifrån kurvorna säga vilken av de två fördelningarna som ger störst sannolikhet att få ett utfall under minus två?

Kurvorna illustrerar *täthetsfunktionen*. I en kontinuerlig fördelning har täthets-

funktionen ingen tolkning i termer av sannolikheter. För att kunna få fram sannolikheter från en normalfördelning behöver vi istället titta på *fördelningsfunktionen*. Kom ihåg från det diskreta fallet att fördelningsfunktionen anger sannolikheten för exakt lika eller under ett givet värde. Samma sak gäller för en kontinuerlig variabel. Fördelningsfunktionen värde ges av `pnorm()`. Om vi vill beräkna sannolikheten att ligga under 1 i en normalfördelning med medelvärde 2 och standardavvikelse 3 tar vi

```
pnorm(1, mean = 2, sd = 3)
```

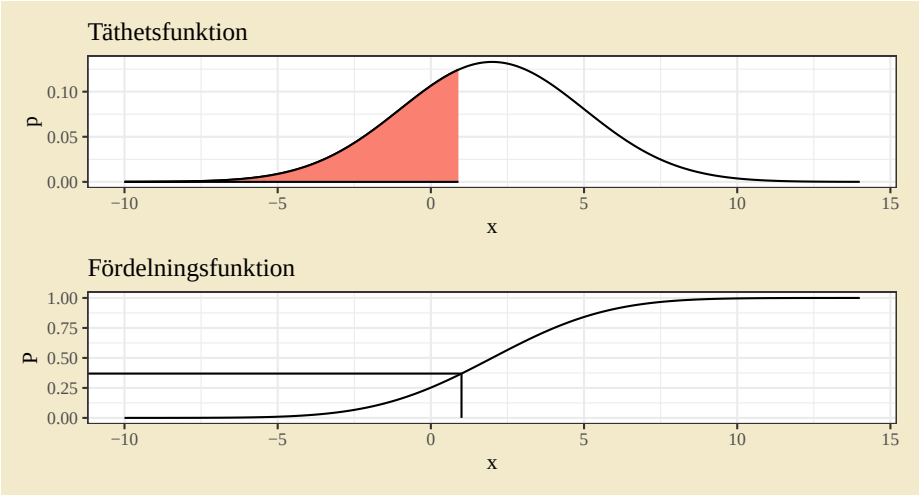
```
## [1] 0.3694413
```

Vi kan ta fram en illustration med följande kod. Detaljer spelar mindre roll här, men gör gärna några ändringar i `x_value`, `mu` och `sigma`för att se hur grafen ändras. Om paketet `patchwork` inte är installerat, kör raden som här är utkommenterad med `#`.

```
x_value <- 1
mu <- 2
sigma <- 3

P_value <- pnorm(x_value, mean = mu, sd = sigma)
P_value
```

```
## [1] 0.3694413
```

```
dat_norm <- data.frame(x = seq(from = mu - 4 * sigma, to = mu + 4 * sigma, 0.1)) %>%
  mutate(p = dnorm(x, mean = mu, sd = sigma),
         P = pnorm(x, mean = mu, sd = sigma))

g1 <- ggplot(dat_norm, aes(x, p)) +
  geom_line() +
  geom_ribbon(aes(ymin = 0, ymax = p), data = dat_norm %>% filter(x < x_value), fill = "salmon",
  labs(title = "Täthetsfunktion")

g2 <- ggplot(dat_norm, aes(x, P)) +
  geom_line() +
  annotate("segment", x = x_value, y = 0, xend = x_value, yend = P_value) +
  annotate("segment", x = x_value, y = P_value, xend = -Inf, yend = P_value) +
  labs(title = "Fördelningsfunktion")

# install.packages("patchwork")
library(patchwork)
g1 / g2
```



Sannolikheten att ligga under ett värde på x ges av kurvan för fördelningsfunktionen vid det x-värdet. Det motsvarar den fyllda ytan under täthetsfunktionen till vänster om x-värdet.

Uppgift 3.17 (Sannolikhet från normalfördelningen)**.** Fyll i kodstycket nedan för att beräkna sannolikheten att få ett värde under minus två i en normalfördelning med medelvärde 0 och standardavvikelse 1, och i en normalfördelning med medelvärde 1 och standardavvikelse 2.

```
pnorm(-2, mean = ___, sd = ___)
pnorm(-2, mean = ___, sd = ___)
```

Om man vill ta fram en sannolikhet att ligga i ett visst intervall kan man ta skillnaden mellan två värden från fördelningsfunktionen. Sannolikheten att ligga mellan 1 och minus 1 i en normalfördelning med medelvärde 0 och standardavvikelse 1 ges till exempel av

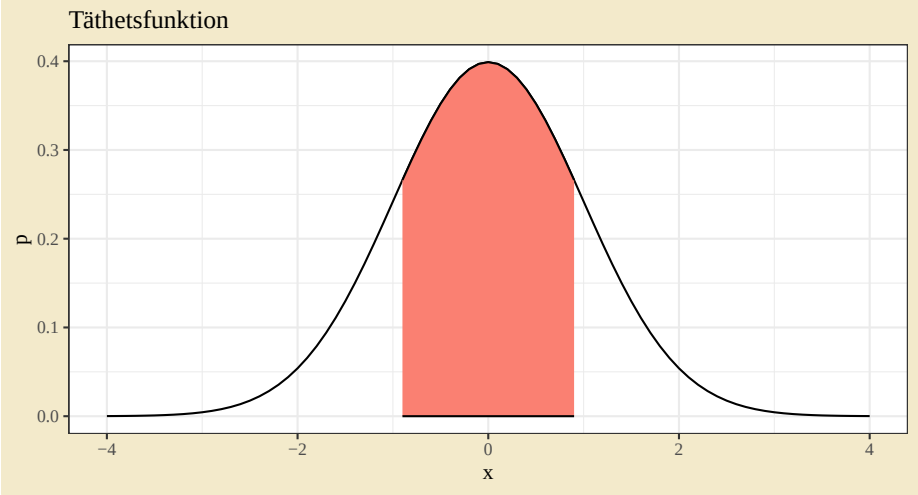
```
pnorm(1, mean = 0, sd = 1) - pnorm(-1, mean = 0, sd = 1)
```

```
## [1] 0.6826895
```

Den normalfördelningen (medelvärde 0 och standardavvikelse 1) kallas den *standardiserade* normalfördelning. Vi kan illustrerade med följande.

```
x_values <- c(-1,1)
mu <- 0
sigma <- 1

dat_norm <- data.frame(x = seq(from = mu - 4 * sigma, to = mu + 4 * sigma, 0.1)) %>%
  mutate(p = dnorm(x, mean = mu, sd = sigma))

ggplot(dat_norm, aes(x, p)) +
```

```
  geom_line() +
  geom_ribbon(aes(ymin = 0, ymax = p), data = dat_norm %>% filter(x < max(x_values) & x > min(x_v
  labs(title = "Täthetsfunktion")
```



Uppgift 3.18 (Sannolikhet mellan två värden)**.** Fyll i kodstycket nedan för att beräkna sannolikheten att få ett värde mellan *minus två* och tre i en normalfördelning med medelvärde 1 och standardavvikelse 2.

```
pnorm(___, mean = ___, sd = ___) - pnorm(___, mean = ___, sd = ___)
```

Fördelningsfunktionen ger sannolikheten att ligga under ett visst x-värde. Man kan enkelt beräkna sannolikheten att ligga över värdet genom att ta ett minus fördelningsfunktionen. Sannolikheten att ligga över 1.96 i en standardiserad normalfördelning ges till exempel av ungefär 2.5 procent.

```
1 - pnorm(1.96)
```

```
## [1] 0.0249979
```

Uppgift 3.19 (Sannolikhet över x)**.** Fyll i kodstycket nedan för att beräkna sannolikheten att få ett värde över sju i en normalfördelning med medelvärde 3 och standardavvikelse 5.

```
1 - pnorm(___, mean = ___, sd = ___)
```

En normalfördelning kan transformeras till en annan normal genom att addera och multiplicera med någon konstant. Mer specifik kan vilken normalfördelning som helst återföras till en standardiserad normalfördelning genom att dra ifrån medelvärdet och dela med standardavvikelsen. Det här utnyttjas när man beräknar sannolikheter för hand. Säg till exempel att vi har en normalfördelning med medelvärde 8 och standardavvikelse 4, och vi vill ta fram sannolikheten att ligga under 7. Det kan beräknas med

```
pnorm(7, mean = 8, sd = 4)
```

```
## [1] 0.4012937
```

Alternativt kan man standardiserade genom att att ta 7 minus 8, delat på 4, vilket ger (7 - 8) / 4 = -0.25, och sedan göra sannolikhetsberäkningen i den standardiserade normalen.

```
pnorm(-0.25, mean = 0, sd = 1)
```

```
## [1] 0.4012937
```

Uppgift 3.20 (Standardisering)**.** En slumpvariabel Y följer en normalfördelning med medelvärde 2 och varians 9. Vad är sannolikheten att få

- $P(Y > 2.75)$,
- $P(Y < 2.75)$,
- $P(2.30 < Y < 2.45)$?

Beräkna först sannolikheten för hand och sedan med `pnorm()`. Notera att denna fråga finns bland instuderingsuppgifterna.

3.7 Bonus. Summan av två slumpvariabler med slumptal

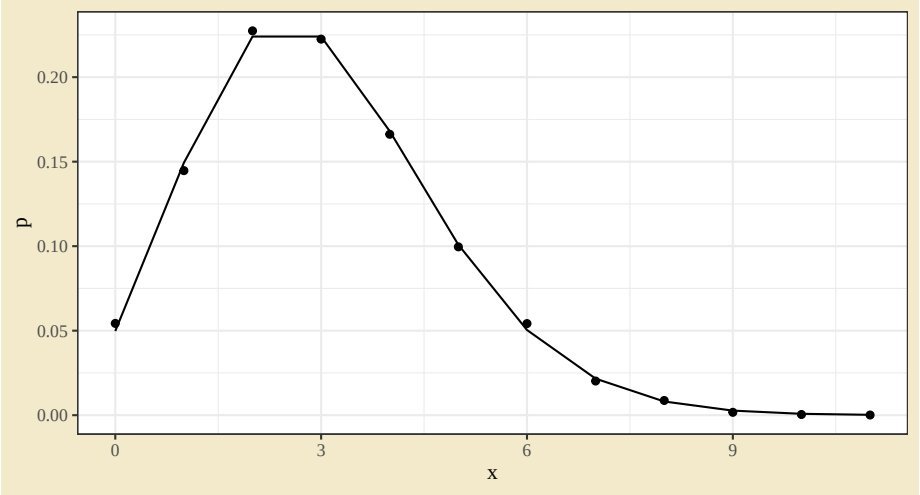
Utöver funktioner för sannolikhetsfunktion (eller täthetsfunktion) och fördelningsfunktion (som `dbinom()`, `dpois()`, `dnorm()`, respektive `pbinom()`, `ppois()`, `pnorm()`) har R funktioner för att ta fram slumptal (`rbinom()`, `rpois()`, `rnorm()`). Slumptal kan vara användbara för att undersöka egenskaper hos en slumpprocess.

Låt oss som ett första exempel undersöka egenskaper hos poissonfördelningen genom att titta på följande fråga: om vi har två poissonfördelningar, med lambda 3 respektive 2, vilken fördelning har summan av de två fördelningarna? Vi kan t.ex. tänka oss att antalet blåsippor i en försöksruta är poissonfördelat med medelvärde 3 och antalet vitsippor är poissonfördelat med medelvärde 2, och att vi vill veta fördelningen för antalet blåsippor plus antalet vitsippor.

Vi börjar med att dra slumptal för en poissonfördelning med lambda lika med 3 och jämför slumptalen med en teoretisk fördelning.

```
dat_pois <- data.frame(x = rpois(10000, lambda = 3)) %>%
  count(x) %>%
  mutate(p = n / sum(n),
         theoretical_p = dpois(x, lambda = 3))

ggplot(dat_pois, aes(x, p)) +
```

```
  geom_point() +
  geom_line(aes(y = theoretical_p))
```

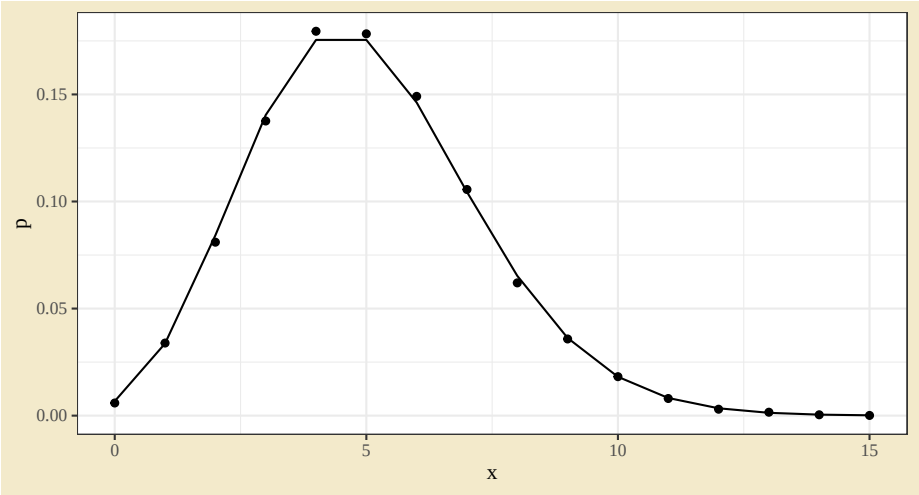


Linjen ger den teoretiska poissonfördelningen och punkterna ger fördelningen för slumptalen. Punkterna ligger nära linjen, vilket är helt efter förväntan. Notera dock att det kommer finnas vissa naturliga avvikelser när man arbetar med slumptal.

Uppgift 3.21 (Poisson från slumptal)**.** Gör lämpliga ändringar i kodstycket ovan för att göra motsvarande beräkning för en poissonfördelning med lambda lika med 2. Ligger slumptalen nära den teoretiska fördelningen?

Vi kan nu ta från två serier av slumptal, beräkna dess summa och jämföra med en teoretisk fördelning. Som teoretisk fördelning tar vi en poissonfördelning med lambda 5, eftersom 5 är summan av våra två lambdavärden.

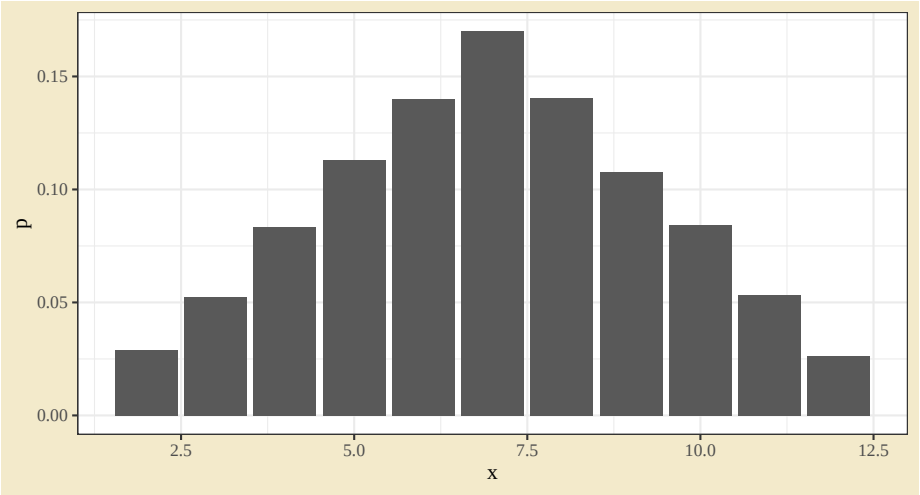
```
dat_pois <- data.frame(x1 = rpois(10000, lambda = 3),
                       x2 = rpois(10000, lambda = 2)) %>%
  mutate(x = x1 + x2) %>%
  count(x) %>%
  mutate(p = n / sum(n),
         theoretical_p = dpois(x, lambda = 5))

ggplot(dat_pois, aes(x, p)) +
  geom_point() +
  geom_line(aes(y = theoretical_p))
```



Summans värden ligger när den teoretiska fördelningen. Det stödjer tanken att summan av två poissonfördelningar är poissonfördelad.

Vi såg tidigare hur utfallet av en tärning kan ses som en slumpvariabel (n stycken utfall där samtliga är lika sannolika) och att funktionen `sample()` kan användas för att simulera tärningskast. Vi kan utnyttja det för att undersöka fördelningen för summan av två tärningskast. Exemplet nedan tittar på en sex-sidig tärning.

```
dat_dice_sum <- data.frame(x1 = sample(c(1,2,3,4,5,6), size = 10000, replace = T),
                           x2 = sample(c(1,2,3,4,5,6), size = 10000, replace = T)) %>%
  mutate(x = x1 + x2) %>%
  count(x) %>%
  mutate(p = n / sum(n))

ggplot(dat_dice_sum, aes(x, p)) +
  geom_col()
```



Summan av två tärningskast har en triangelformad fördelning där 7 är det vanligaste utfallet och 2 och 12 är de ovanligaste.

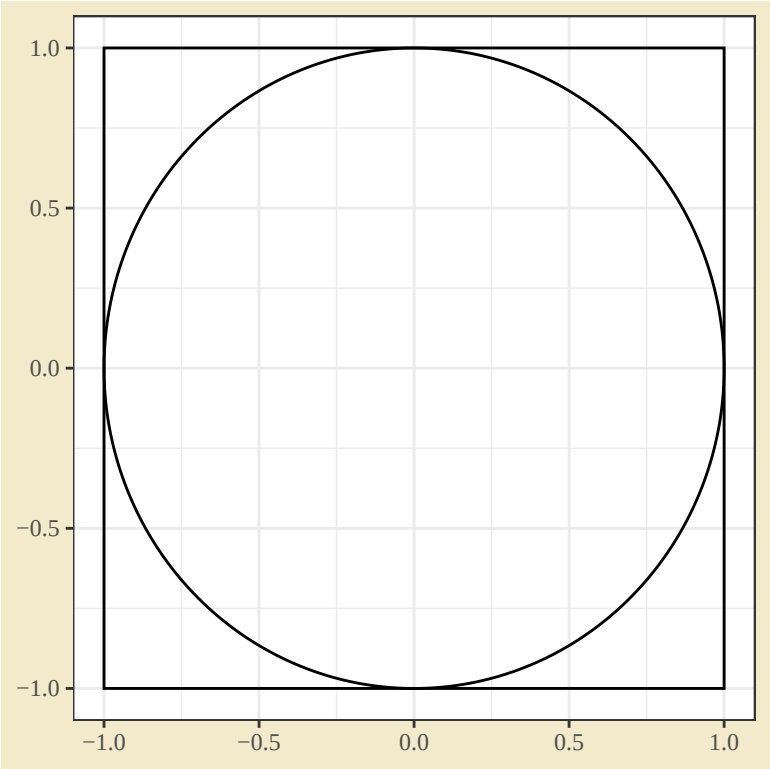
Uppgift 3.22 (Summan av tärningskast med annan tärning)**.** Gör lämpliga ändringar i stycket ovan för att beräkna summan av två utfall för en åtta-sidig tärning.

Uppgift 3.23 (Summan av tre tärningskast)**.** Gör lämpliga tillägg i stycket ovan för att beräkna summan av tre utfall för en sex-sidig tärning.

3.8 Bonus. Cirkelns area från slumptal

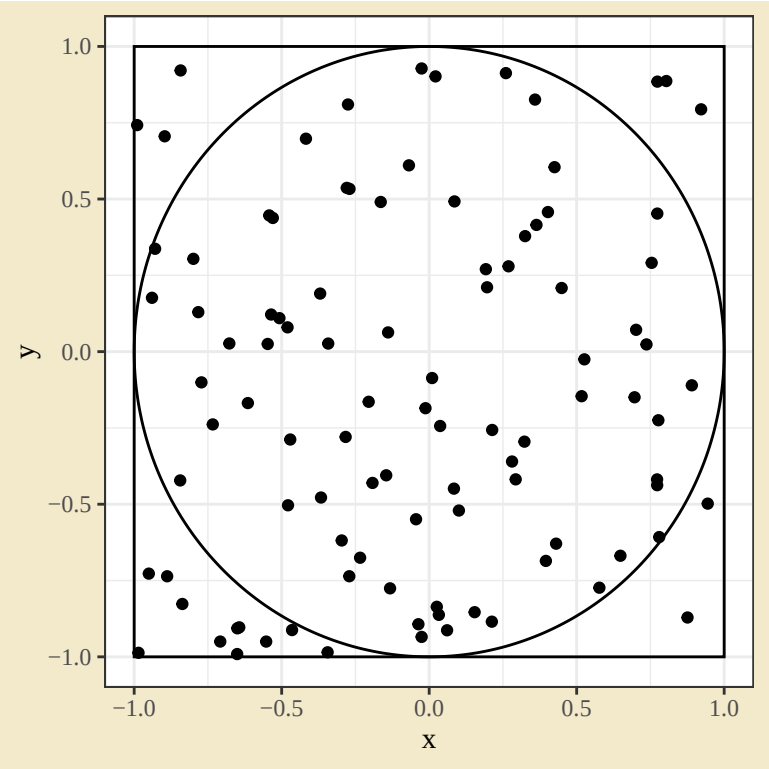
En intressant aspekt av slumptal är att de kan användas till att utforska icke-slumpmässiga problem. Som exempel tar vi cirkelns area. I figuren nedan är en cirkel inskriven i en kvadrat. Kvadraten har sidor med längden två, så dess totala yta är fyra (två i kvadrat). Om man vet hur stor andel av ytan upptas av cirkeln kan man beräkna cirkelns area genom att ta den andelen gånger fyra. Från rent matematiska resultat vet vi att arean ska vara lika med pi, alltså runt 3.14.

```
# install.packages("ggforce")
library(ggforce)
g <- ggplot() +
  geom_circle(aes(x0 = 0, y0 = 0, r = 1)) +
  geom_rect(aes(xmin = -1, ymin = -1, xmax = 1, ymax = 1), fill = NA, color = "black")
g
```



Funktionen `runif()` ger ett slumpmässigt värde mellan två gränsvärden (som sätts med argumenten `min` och `max`). Slumpmässiga punkter inom kvadraten kan dras genom att dra en x- och en y-koordinat. En fördelning där alla värden är lika sannolika kallas *uniform* (eller *likformig*).

```
dat_random <- data.frame(x = runif(100, min = -1, max = 1),
                         y = runif(100, min = -1, max = 1))

g + geom_point(aes(x, y), data = dat_random)
```



Pytagoras sats kan användas för att beräkna avståndet mellan en slumpmässig punkt och origo (nollpunkten). Punkter med ett avstånd under ett ligger inom cirkeln. Cirkelns area ska ges av andelen punkter i kvadraten som också ligger i cirkeln, gånger fyra.

```
dat_random <- data.frame(x = runif(100, min = -1, max = 1),
                         y = runif(100, min = -1, max = 1)) %>%
  mutate(inner = x^2 + y^2 < 1)

4 * mean(dat_random$inner)
```

```
## [1] 3.32
```

```
pi
```

```
## [1] 3.141593
```

Den beräknade arean ligger inte så långt från konstanten pi.

Uppgift 3.24 (Fler punkter ger bättre skattning)**.** Beräkningens precision ökar med antalet slumptal. Vad i kodstycket ovan ska ändras för att generera fler slumptal? Blir utfallet närmre det väntade värdet om antalet slumptal ökar?

3.9 Bonus. Slump och ordning - Sierpinski-triangeln

Slumpmässiga processer kan ge intressanta mönster. Ett exempel är följande procedur.

1. Ta tre *rampunkter* i planet så att de bildar en triangel.
2. Ta en *startpunkt* inom triangeln.
3. Välj slumpmässigt en av de tre rampunkterna och beräkna punkten mellan startpunkten och den valda rampunkten. Detta ger en ny punkt.
4. Välj slumpmässigt en av de tre rampunkterna och beräkna punkten mellan den nya punkten från föregående steg och rampunkten. Detta ger en ny punkt.
5. Upprepa steg 4 ett godtyckligt antal gånger.

I R kan proceduren programmeras med en `for`-loop. Här ges ett exempel där 100 punkter genereras. Koden är viss överkurs och innehåller funktioner vi inte behöver till den övriga kursen.

```
# Ta ut tre punkter
x_original <- c(0,1,2)
y_original <- c(0,2,0)

# Välj en punkt inom triangeln
x_new <- 0.4
y_new <- 0.2
dat_tri <- data.frame(x = x_new, y = y_new)

n <- 100

for(i in 1:n){
  new_point <- sample(c(1,2,3), 1)
  x_new <- (x_new + x_original[new_point]) / 2
  y_new <- (y_new + y_original[new_point]) / 2

  dat_tri <- bind_rows(dat_tri, data.frame(x = x_new, y = y_new))
}

ggplot(dat_tri, aes(x, y)) +
  geom_point()
```

Uppgift 3.25 (Sierpinski-triangeln)**.** Vad måste ändras i stycket ovan för att generera fler punkter? Tiotusen kan vara ett lämpligt antal för en tydligare illustration. Vad händer om man ändrar värdena i `x_original` och `y_original`?

Chapter 4

Ett stickprov av normalfördelad data

Datorövning 4 handlar om hypotestest och konfidensintervall för ett stickprov av normalfördelad data. Efter övningen ska vi kunna

- genomföra och tolka ett t-test för normalfördelad data,
- beräkna och tolka ett konfidensintervall för normalfördelad data,
- använda simulerad data för att förstå t-testets egenskaper.

4.1 Repetition av datorövning 3

När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 3 tittade vi på funktioner för att ta fram sannolikheter och slumptal från någon given fördelning.

Sannolikhetsfunktionen - som anger sannolikheten för ett specifikt utfall - kan i R tas fram med funktioner som börjar med `d`, till exempel `dbinom()` och `dpois()`. För en kontinuerlig slumpvariabel ger `d` istället täthetsfunktionens värde, till exempel `dnorm()` för normalfördelningen.

Binomialfördelningen är en fördelning för summan av positiva utfall vid n upprepningar av likadana binära slumphändelser. Fördelningens parametrar är n - antalet försök, och p - sannolikheten för varje enskilt försök. Ett exempel kan

vara att man kastar en tiosidig tärning fem gånger och tittar på antalet ettor. Följande beräkning ger sannolikheten att få tre ettor vid fem kast.

```
dbinom(3, 5, prob = 0.1)
```

```
## [1] 0.0081
```

Poissonfördelning är en fördelning för antalet händelser om händelsen potentiellt kan inträffa väldigt ofta men gör så med låg sannolikhet. Fördelningen har en parameter, lambda, och vanliga exempel är observationer av ovanliga arter och olika typer av sällsynta händelser. Säg som exempel att någon art vanligen observeras tio gånger per år i en viss region. Sannolikheten för exakt fem observationer ett visst år ges av

```
dpois(5, lambda = 10)
```

```
## [1] 0.03783327
```

Fördelningsfunktionen - som anger sannolikhet under eller lika med ett givet värde - tas fram med funktioner som börjar på `p`, till exempel `pbinom()`, `ppois()` och `pnorm()`. Fördelningsfunktionen är särskilt användbar för normalfördelningen. Om en plantas höjd tros vara normalfördelad med medelvärde 10 cm och standardavvikelse 2, kan man beräkna sannolikheten att en slumpmässigt vald planta är över 13 cm.

```
1 - pnorm(13, mean = 10, sd = 2)
```

```
## [1] 0.0668072
```

Eftersom `pnorm()` ger sannolikheten *under* 13, ger ett minus den sannolikheten värdet *över* 13.

Slumputfall kan beräknas med funktionen `sample()`, till exempel `sample(1:6, 10000000, replace = T)` för tio miljoner tärningskast. För specifika fördelningar finns särskilda funktioner som `rbinom()`, `rpois()` och `rnorm()`.

4.2 Test av medelvärde för normalfördelad data

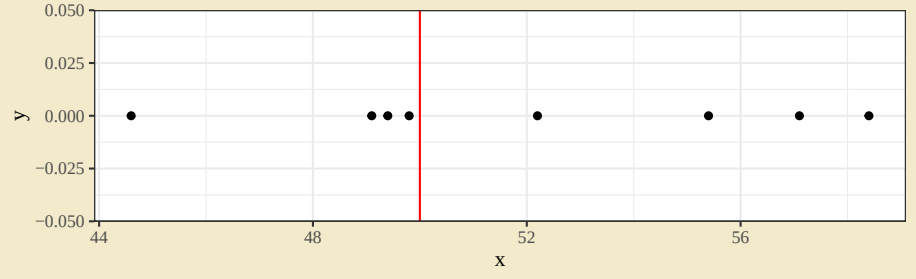
Om man har en normalfördelad variabel och vill testa om populationens medelvärde är skilt från något hypotetiskt värde μ_0 kan man använda ett *t-test för ett stickprov*. Ta som exempel följande data på 8 observationer av havreskörd. Av någon historisk anledning vill man testa om populationsmedelvärdet är skilt från 50.

```
library(tidyverse)

dat <- data.frame(x = c(49.8, 58.4, 49.4, 57.1, 52.2, 49.1, 44.6, 55.4))
dat
```

```
##      x
```

```
## 1 49.8
## 2 58.4
## 3 49.4
## 4 57.1
## 5 52.2
## 6 49.1
## 7 44.6
## 8 55.4
```

```
ggplot(dat, aes(x, 0)) +
  geom_point() +
  geom_vline(xintercept = 50, color = "red")
```



I grafen ser vi att värdena ligger jämt spridda kring 50, så 50 är nog ganska rimligt som medelvärde, men låt oss göra ett formellt test. I R kan ett t-test genomföras med `t.test()`.

```
t.test(dat$x, mu = 50)    # Ett t-test på variabeln x i dataobjektet dat
```

```
##
##  One Sample t-test
##
## data:  dat$x
## t = 1.2086, df = 7, p-value = 0.266
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  48.08713 55.91287
## sample estimates:
## mean of x
##        52
```

Utskriften ger ett p-värde från vilket vi kan dra en slutsats. I det här fallet är p-värdet högt (över fem procent) så vi kan inte förkasta nollhypotesen (vilken är att populationsmedelvärdet är lika med 50).

Vi tittar nu på stegen bakom t-testet. Ett t-test bygger, som alla hypotestest, på en serie steg:

1. sätt upp en *nollhypotes* och en *alternativhypotes*,

2. beräkna ett *testvärde* från en testfunktion,
3. identifiera en *testfördelning*,
4. beräkna ett *p-värde*, eller uppskatta ett genom att ställa testvärde mot ett kritiskt värde,
5. dra en klar *slutsats* om statistisk signifikans.

Vi vill testa om medelskörden är skild från 50, så hypoteser ges av

- H0: mu lika med 50
- H1: mu ej lika med 50

Alternativhypotesen är tvåsidig - vi tittar både på möjligheten att populationsmedelvärdet är större och på möjligheten att det är mindre.

Uppgift 4.1 (Ensidig mothypotes)**.** Hur hade hypoteserna sett ut om vi ville testa om medelvärdet är *större* än 50?

Vårt mål är att testa ett medelvärde och det är rimligt att anta normalfördelning för den undersökta variabeln. Det lämpliga testet är då ett t-test för ett stickprov och testvärdet kan beräknas av en testfunktion som ges av det observerade stickprovet minus nollhypotesens värde, delat på standardavvikelsen delat på roten ur antalet observationer. Låt oss beräkna detta i flera steg och jämföra med utskriften från `t.test()`.

```
mean(dat$x)
```

```
## [1] 52
```

```
sd(dat$x)
```

```
## [1] 4.680354
```

```
t_value <- (52 - 50) / (4.680354 / sqrt(8))
```

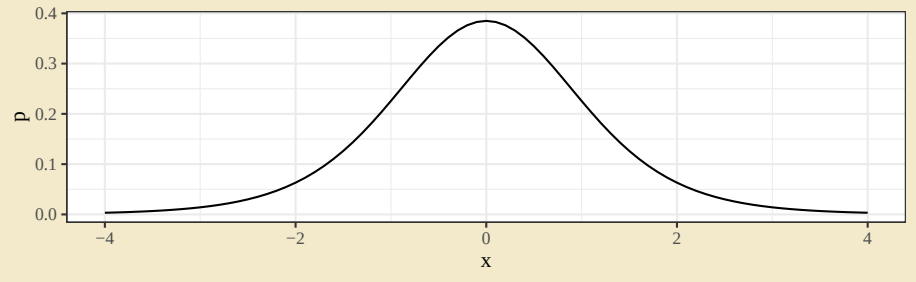
Uppgift 4.2 (Operationsordning)**.** Räkna ut samma sak på miniräknare eller telefon. Vad händer om man missar parenteser runt `4.680354 / sqrt(8)`?

Uppgift 4.3 (t-värdets delar)**.** Vad händer med t-värdet om något av följande händer, givet att övriga delar är desamma? Det observerade medelvärdet (här 52) ökar. Nollhypotesens värde (här 50) minskar. Standardavvikelsen (här 4.680354) minskar. Antalet observationer (här 8) ökar.

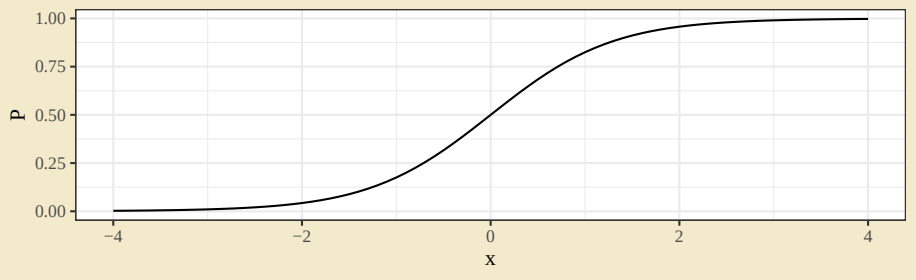
Testa genom att ändra värdena i kodstycket ovan och beräkna `t_value` på nytt.

Nästa steg är att identifiera testfördelning, det vill säga den slumpfördelning testvärdet följet om nollhypotesen är sann. Fördelningen ges i regel av statistisk teori. I det här fallet är testfördelning en t-fördelning med n - 1 frihetsgrader. Vi har åtta observationer, så antalet frihetsgrader blir 7. I R kan man ta fram täthetsfunktionen för en t-fördelning med `dt()` och fördelningsfunktionen med `pt()`.

```
dat_t <- data.frame(x = seq(-4, 4, 0.1)) %>%
  mutate(p = dt(x, df = 7),
         P = pt(x, df = 7))

ggplot(dat_t, aes(x, p)) +
  geom_line()
```

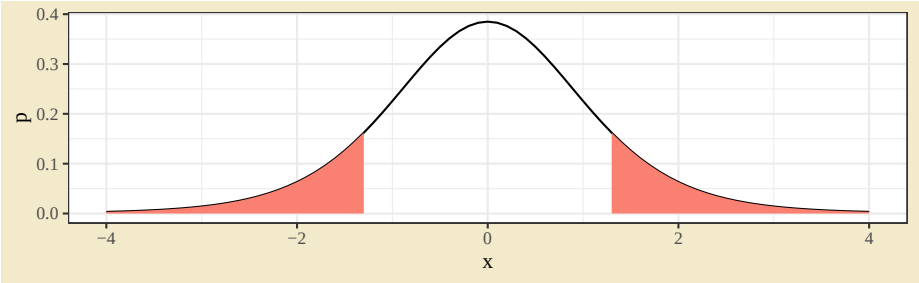



```
ggplot(dat_t, aes(x, P)) +
  geom_line()
```



Nästa steg i ett hypotestest är att ta fram ett p-värdet. P-värdet kan illustreras som ytan under t-fördelning *bortom* test-värdet. I ett tvåsidigt test tar vi med bägge svansarna.

```
ggplot(dat_t) +
  geom_line(aes(x, p)) +
  geom_ribbon(aes(x = x, ymin = 0, ymax = p), data = dat_t %>% filter(x > abs(t_value)), fill = '
  geom_ribbon(aes(x = x, ymin = 0, ymax = p), data = dat_t %>% filter(x < -abs(t_value)), fill =
```



De röda ytorna i *svansarna* motsvarar p-värdet. De anger sannolikheten att få ett större t-värde än det vi fått, under antagandet att nollhypotesen stämmer. Vi kan beräkna p-värdet med `pt()`. För ett tvåsidigt test multipliceras värdet med 2.

```
2 * pt(-abs(t_value), 7) # Tvåsidigt p-värde
```

```
## [1] 0.2660402
```

P-värdet ges av 0.266. En tolkning av det är om försöket upprepas ett stort antal gånger och nollhypotesen är sann, kommer vi 26.6 procent av gångerna få ett större testvärde än 1.2086. Det vi observerar är ganska sannolikt under nollhypotesen, vilket tyder på att nollhypotesen är rimligt.

Det avslutande steget är att dra en formell slutsats och ge ett tydligt svar. Det klassiska sättet är att jämföra p-värdet med en förbestämd signifikansnivå, oftast fem procent. Här är p-värdet över den nivån, så vi kan inte förkasta nollhyptesen. Slutsatsen är att det *inte* finns någon signifikant skillnad från 50 i havreskörd.

Uppgift 4.4 (Kritiskt värde)**.** Om man gör ett t-test för hand kan man inte enkelt ta fram ett p-värde, men kan se om p-värdet är större eller mindre än fem procent genom att ställa testvärdet mot ett kritiskt värde. Använd en tabell för t-fördelning för att hitta det kritiska värdet.

I R kan man ta fram kritiska värden med `qt()`. För fem procent i svansarna har man 0.025 i respektive svans och det kritiska värdet ges av

```
qt(0.975, 7)
```

```
## [1] 2.364624
```

Som visades i början har R en specifik funktion för t-testet, `t.test()`. Funktionens argument är datan man testar och ett nollhypotesvärde `mu`. Om man vill ha ett ensidigt test kan det sättas med argumentet `alternative`. För vår data ges testet av

```
t.test(dat$x, mu = 50) # Tvåsidigt test
```

```
##
##  One Sample t-test
```

```
##
## data:  dat$x
## t = 1.2086, df = 7, p-value = 0.266
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  48.08713 55.91287
## sample estimates:
## mean of x
##        52
```

Funktionen skriver ut det beräkna t-värdet, antal frihetsgrader och p-värdet.

Uppgift 4.5 (Ensigt test)**.** Använd `?t.test` för att ta fram funktionens hjälpsida. Försök att utifrån hjälpsidan beräkna ett *ensidigt* test för att se om medelskörden är *större* än 50.

Uppgift 4.6 (Ny nollhypotes)**.** Upprepa det tvåsidiga testet från exemplet ovan. Testa denna gång om medelskörden är skild från 48. Dra en tydlig slutsats.

Uppgift 4.7 (Importera smältpunkt-data)**.** På canvassidan finns en excelfil med data för kursens uppgifter *Uppgiftsdata.xlsx*. Fliken *Smältdata* innehåller data för en legerings smältpunkt.

Ladda ner filen till lämplig plats på datorn och importera datan genom att fylla i följande rad.

```
library(readxl)
dat_smält <- read_excel("___", sheet = "Smältpunkt")
```

Uppgift 4.8 (Plotta smältpunkt-data)**.** Illustrera smältpunktsdatan på samma sätt som exempeldatan genom att fylla i följande kod. Vårt mål är att testa om medelvärde är skilt från 1050, vilket här kan noteras med ett vertikalt streck vid 1050.

```
ggplot(___, aes(x = Smältpunkt, y = 0)) +
  ___() +
  geom_vline(xintercept = ___, color = "red")
```

Kan man utifrån grafen säga om 1050 är ett rimligt medelvärde för populationen, givet det stickprov vi observerar?

Uppgift 4.9 (Hypotestest för hand)**.** Genomför ett t-test för hand för att se om medelsmältpunkten är skild från 1050. Skriv ut tydliga hypoteser. Medelvärde och standardavvikelse ges av följande.

```
mean(dat_smält$Smältpunkt)
sd(dat_smält$Smältpunkt)
```

Ett kritiskt värde kan tas från en tabell över t-fördelningen eller beräknas i R med

```
qt(0.975, df = 9)
```

Uppgift 4.10 (Hypotestest i R)**.** Genomför ett t-test med funktionen `t.test()` för att se om medelsmältpunkten är skild från 1050.

4.3 Konfidensintervall för normalfördelad data

För exemplet på havredata tittade vi på två olika värden för nollhypotesen.

```
t.test(dat$x, mu = 50)
t.test(dat$x, mu = 48)
```

Från p-värdena kan man dra slutsatsen att förkasta vid nollhypotesen att mu är 48, men inte förkasta vid nollhypotesen att mu är 50. Värdet 50 är alltså i någon mening ett mer troligt värde på populationens medelvärde än vad 48 är. *Konfidensintervall* kan ses som en generalisering av den tanken: ett konfidensintervall ger ett spann av värden där man *inte* förkastar. Intervallet tolkas vanligen som att det täcker det sanna populationsmedelvärdet med en viss konfidens.

För ett stickprov och antagen normalfördelning ges konfidensintervallet av

medelvärde ± kvantil från t-fördelningen x standardavvikelse delat på roten ur antalet observationer

Kvantilen från t-fördelningen kan hämtas från en tabell (samma som det kritiska värdet i testet) eller genom R. Antalet frihetsgrader ges av antalet observationer minus ett. I det här fallet ges delarna av

```
mean(dat$x)
```

```
## [1] 52
```

```
sd(dat$x)
```

```
## [1] 4.680354
```

```
qt(0.975, 7)
```

```
## [1] 2.364624
```

och konfidensintervallet ges alltså av

52 ± 2.365 * 4.680 / sqrt(8)

Uppgift 4.11 (Konfidensintervall för hand)**.** Ta fram medelvärde, standardavvikelse och kritiskt värde för smältpunktsdata, med hjälp av R (eller en tabell för det kritiska värdet). Beräkna konfidensintervallet för smältpunktsdatan för hand.

Funktionen `t.test()` ger automatiskt ett konfidensintervall, direkt under utfallet av testet. Notera att konfidensintervallet inte beror på nollhypotesen.

Konfidensintervall kan beräknas med skilda konfidensnivåer, oftast 95 procent, vilket sätts med argumentet `conf.level`.

Uppgift 4.12 (Konfidensnivå)**.** Gör lämplig ändring i koden nedan för att beräkna ett 99-procentigt konfidensintervall, istället för ett 95-procentigt.

```
t.test(dat$x, conf.level = 0.95)
```

```
##
##  One Sample t-test
##
## data:  dat$x
## t = 31.425, df = 7, p-value = 8.538e-09
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  48.08713 55.91287
## sample estimates:
## mean of x
##        52
```

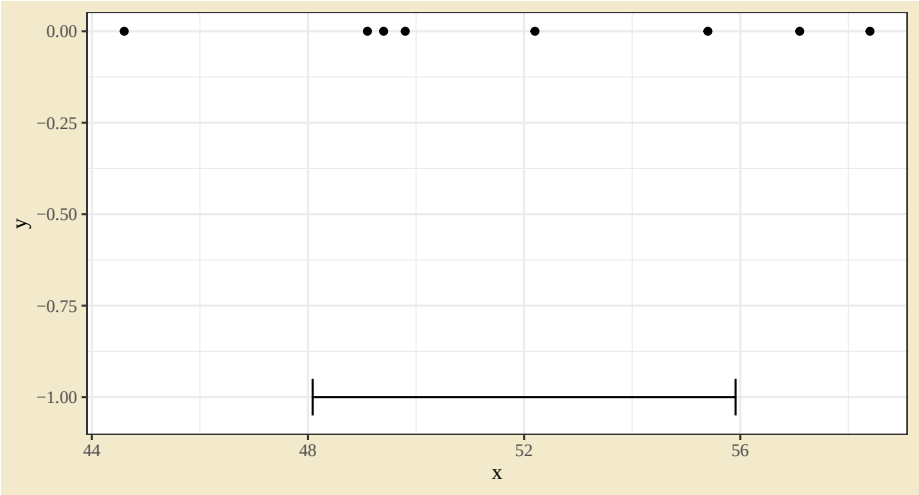
Är ett 99-procentigt konfidensintervall bredare eller smalare än ett 95-procentigt?

Uppgift 4.13 (Ensidiga konfidensintervall)**.** I en tidigare uppgift användes argumentet `alternative` för att göra ett ensidigt test med `t.test()`. Vad händer med konfidensintervallet om man anger ett ensidigt test?

Uppgift 4.14 (Konfidensintervall för smältdata)**.** Ta datan över smältpunkter och beräkna ett konfidensintervall med `t.test()`. Tolka intervallet.

Ett konfidensintervall illustreras ofta med en felstapel. Vi kan lägga till en till den punktgraf vi tidigare sett för observationerna.

```
interval <- t.test(dat$x)$conf.int
ggplot(dat, aes(x, 0)) +
  geom_point() +
  annotate("errorbar", xmin = interval[1], xmax = interval[2], y = -1, width = 0.1)
```

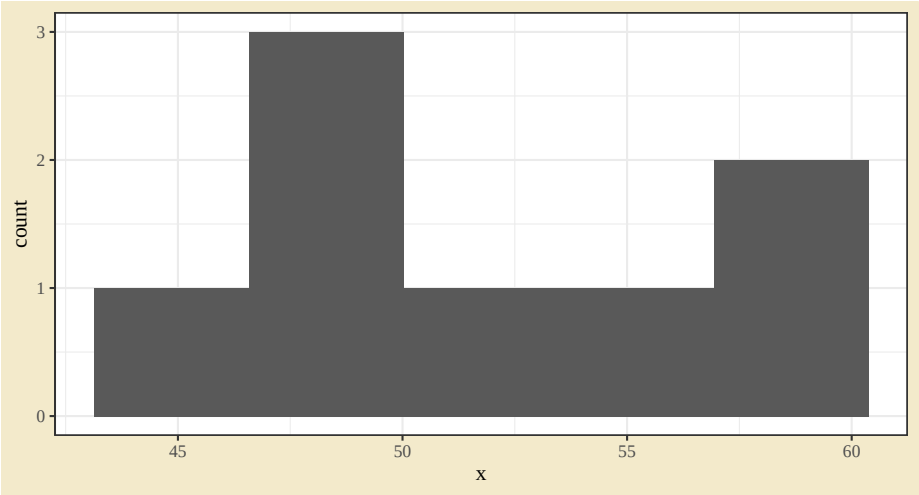


Uppgift 4.15 (Illustration av smältdata)**.** Använd exempelillustrationen för havredata till en liknande illustration av smältpunktsdatan.

4.4 Normalfördelad data och centrala gränsvärdesatsen

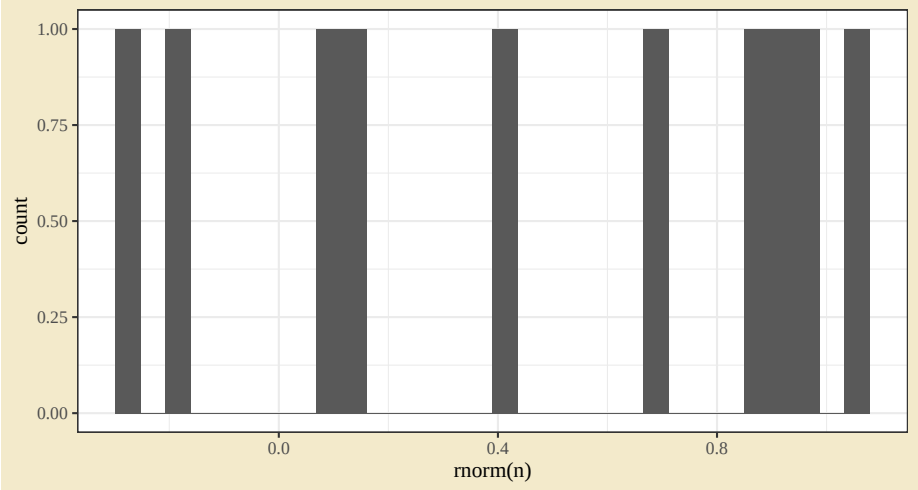
Eftersom t-testet bygger på att data är normalfördelad är det förstås bra att kunna undersöka om det antagandet stämmer. Ett sätt är att göra ett histogram över datan - om den underliggande variabeln är normalfördelad bör stickprovet ge den typiska klockformen. Det här kräver dock ganska mycket data. Ta ett histogram för havredatan som exempel

```
ggplot(dat, aes(x)) + geom_histogram(bins = 5)
```



Uppenbarligen helt meningslöst. Låt oss titta på histogram över genererad normalfördelad data.

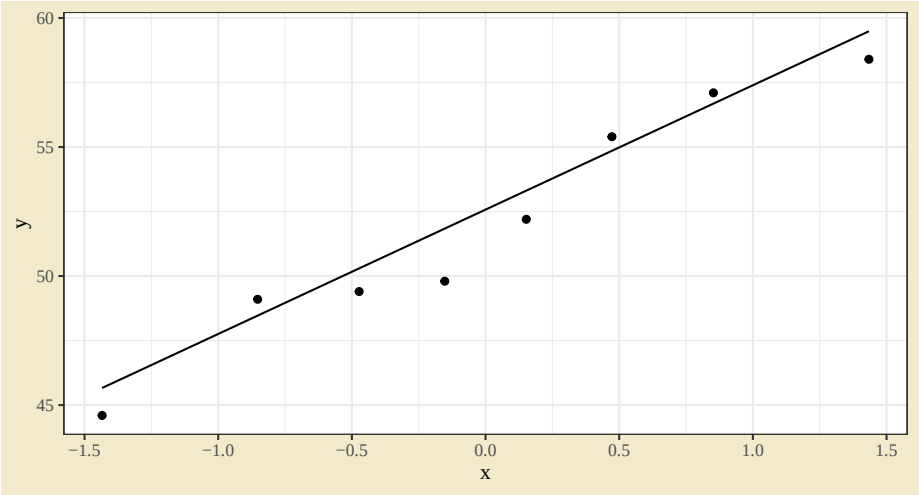
```
n <- 10
ggplot() + geom_histogram(aes(x = rnorm(n)), bins = 30)
```



Uppgift 4.16 (Histogram för normalfördelning)**.** Testa koden ovan för lite olika värden på `n`. Det kan vara nyttigt att sätta antalet staplar `bins` för att få ett bättre histogram. Hur stort måste n vara för att ge en karaktäristisk klockform för histogrammet?

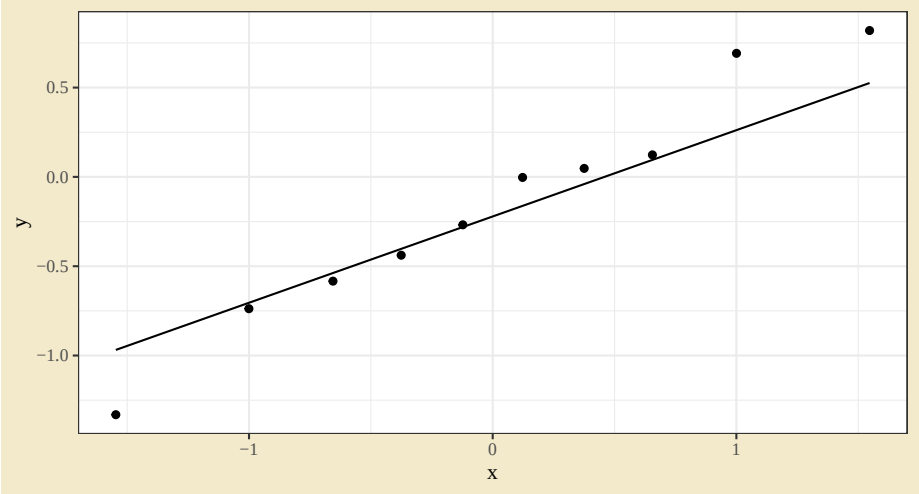
Ett annat vanligt alternativ för att grafisk undersöka om data följer en ungefärlig normalfördelning är en QQ-graf (*QQ-plot*). En qq-graf är ett spridningsdiagram med teoretiska kvantiler på en axel och datans kvantiler på den andra axeln. Om data perfekt följer en normalfördelning kommer grafen visa en rak diagonal linje. En QQ-graf kan tas fram med `qqnorm()` eller `geom_qq()` i en ggplot. En diagonal linje för jämförelse kan läggas till med `geom_qq_line()`.

```
ggplot(dat, aes(sample = x)) + geom_qq() + geom_qq_line()
```



Punkterna ligger nära linjen. Vi kan återigen demonstrera med lite genererad data.

```
n <- 10
dat_norm <- data.frame(x = rnorm(n))
ggplot(dat_norm, aes(sample = x)) + geom_qq() + geom_qq_line()
```

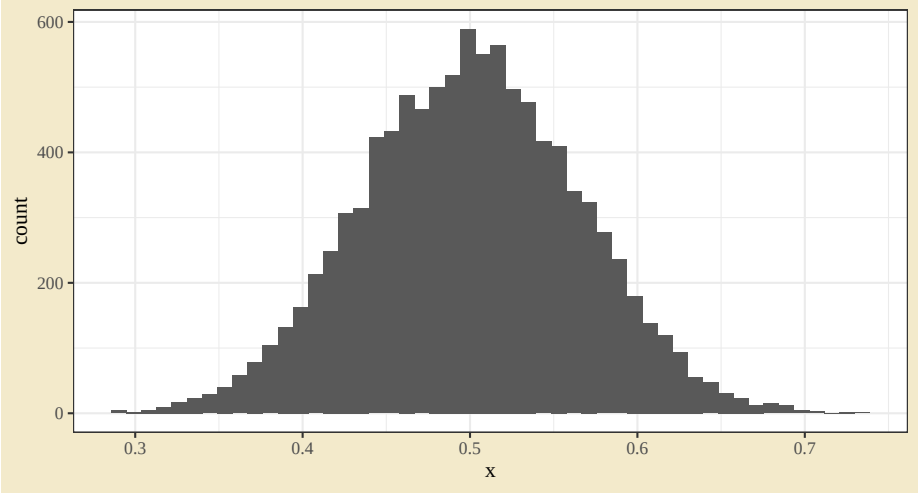


Uppgift 4.17 (Histogram för normalfördelning)**.** Funktionen `runif()` ger slumpmässiga värden mellan 0 och 1. Testa att ändra i kodstycket ovan så att slumptal genereras med `runif()` istället för `rnorm()`. Hur påverkar det QQ-grafen?

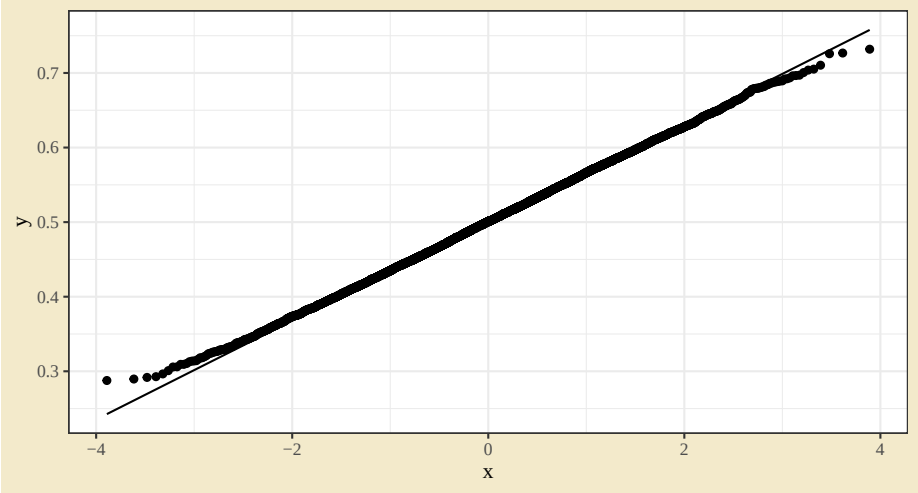
Även om data inte är normalfördelad kan t-testet vara ett lämpligt val av test. Detta beror på *centrala gränsvärdesatsen*, som säger att summor (och därmed även medelvärden) av lika slumpvariabler går mot en normalfördelning där an-

talet observationer ökar. Den tidigare uppgiften gav att `runif()` inte ger normalfördelad data. Vad händer om vi tar medelvärdet av flera observationer från `runif()`? Följande kod beräknar tiotusen medelvärden av två observationer.

```
n <- 20

dat_sim_unif <- expand_grid(Observation = 1:n, Upprepning = 1:10000) %>%
  mutate(x = runif(n())) %>%
  group_by(Upprepning) %>%
  summarise(x = mean(x))

ggplot(dat_sim_unif, aes(x)) + geom_histogram(bins = 50)
```




```
ggplot(dat_sim_unif, aes(sample = x)) + geom_qq() + geom_qq_line()
```

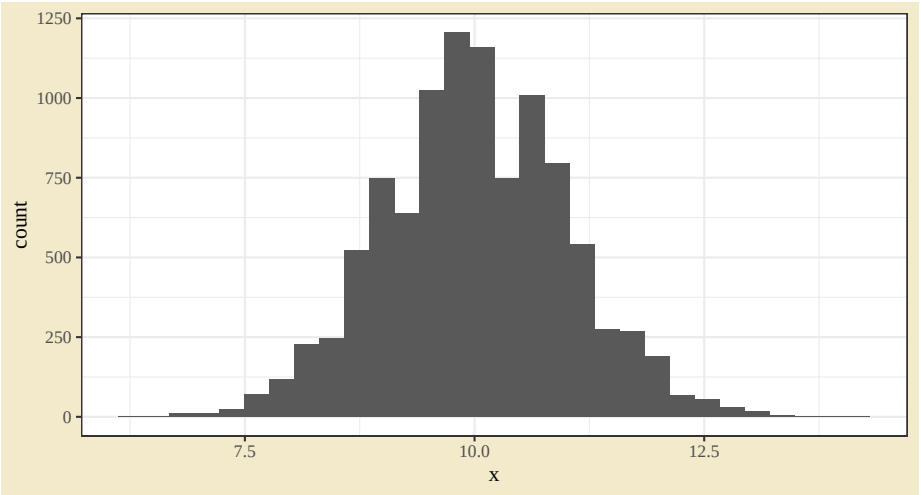


Fördelningen för summan är inte likformig, men inte heller särskilt normalfördelad. Vad händer om vi ökar antal termer i summan?

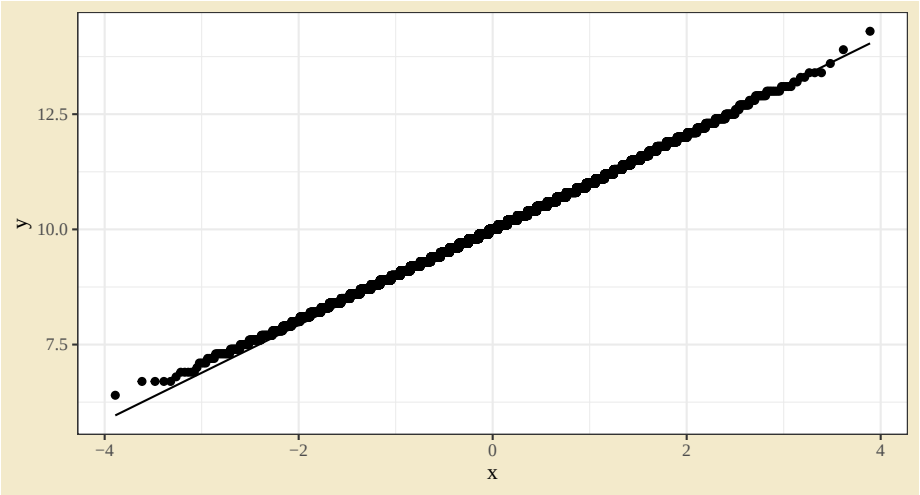
Uppgift 4.18 (Antalet observationer för normalfördelning)**.** Vad måste ändras i koden ovan för beräkna medelvärdet av tio observationer? Följer de medelvärdena en ungefärlig normalfördelning? Vad är det lägsta antalet observationer som ger ungefärligen normalfördelade medelvärden?

Under en tidigare datorövning såg vi exempel på diskreta fördelningar: binomial- och poissonfördelningarna. Vi ska senare titta på specifika test för variabler som följer en diskret fördelning, men centrala gränsvärdesatsen kan även då rättfärdiga ett t-test. Ta som exempel medelvärdet av tio observationer som följer en poissonfördelning med lambda = 10.

```
n <- 10
lambda <- 10
dat_sim_unif <- expand_grid(Observation = 1:n, Upprepning = 1:10000) %>%
  mutate(x = rpois(n(), lambda = lambda)) %>%
  group_by(Upprepning) %>%
  summarise(x = mean(x))

ggplot(dat_sim_unif, aes(x)) + geom_histogram(bins = 30)
```




```
ggplot(dat_sim_unif, aes(sample = x)) + geom_qq() + geom_qq_line()
```

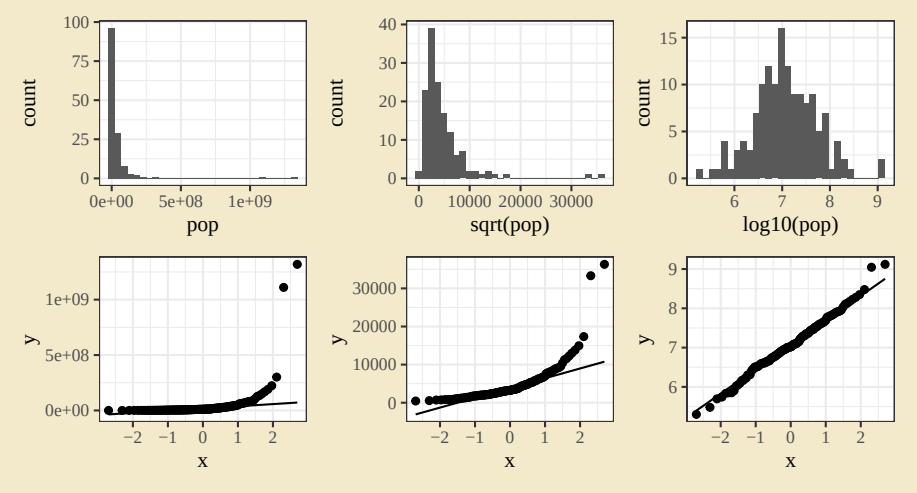


Histogrammet visar på en typisk klockform och punkterna följer linjen *ungefärligt*. QQ-grafens trappeffekt är typisk för diskret data. Det här tyder alltså på att t-testet är ett acceptabelt alternativ om man har en poissonfördelning med lambda runt tio och gör tio upprepningar.

Uppgift 4.19 (Svag normalapproximation)**.** Testa att minska värdena på n och lambda. Vad är de lägsta värdena som ger ett histogram med en symmetrisk fördelning och punkter nära linjen i QQ-grafen?

Det kan också finnas situationer där någon matematisk transformation kan göra icke-normal data till normalfördelad data. Vanliga transformationer är att ta en kvadratrot eller att *logaritmera* datan. Som exempel kan vi återvända till Gapminder-datan vi använde i en tidigare datorövning. Paketet `patchwork` kan användas för att placera flera grafer bredvid varandra. Den exakta koden är mindre viktig här.

```
library(gapminder)
gapminder_2007 <- gapminder %>% filter(year == 2007)

g1 <- ggplot(gapminder_2007, aes(pop)) + geom_histogram(bins = 30)
g2 <- ggplot(gapminder_2007, aes(sample = pop)) + geom_qq() + geom_qq_line()
g3 <- ggplot(gapminder_2007, aes(sqrt(pop))) + geom_histogram(bins = 30)
g4 <- ggplot(gapminder_2007, aes(sample = sqrt(pop))) + geom_qq() + geom_qq_line()
g5 <- ggplot(gapminder_2007, aes(log10(pop))) + geom_histogram(bins = 30)
g6 <- ggplot(gapminder_2007, aes(sample = log10(pop))) + geom_qq() + geom_qq_line()

library(patchwork)
g1 + g3 + g5 + g2 + g4 + g6
```



I grafen har vi den ursprungliga variabeln (befolkning per land 2007), den kvadratrot-transformerade variabeln (`sqrt()`) och den log-transformerade variabeln (`log10()`). De två första fallen påverkas kraftigt av extremvärden och är klart icke-normala medan den log-transformerade variabeln ger en ungefärlig normalkurva och följer diagonalen väl i QQ-grafen.

Uppgift 4.20 (Transformera medellivslängd)**.** Använd kodstycket ovan som mall och ta fram grafer för medellivslängd (`lifeExp`) istället för befolkningsstorlek (`pop`). Visar grafen samma mönster som för befolkningsdatan?

4.5 Bonus. Simuleringar för t-test och konfidensintervall

Följande kod simulerar ett dataset om tio observationer från en normalfördelning med medelvärde 7 och standardavvikelse 5, beräknar ett hypotestest med nollhypotesen att populationsmedelvärdet är 7, och beräknar ett konfidensintervall.

```
dat_sim <- data.frame(x = rnorm(10, mean = 7, sd = 5))
t.test(dat_sim$x, mu = 7)
```

```
##
##  One Sample t-test
##
## data:  dat_sim$x
## t = 0.027533, df = 9, p-value = 0.9786
## alternative hypothesis: true mean is not equal to 7
## 95 percent confidence interval:
##   3.211149 10.882216
## sample estimates:
```

```
## mean of x
##  7.046682
```

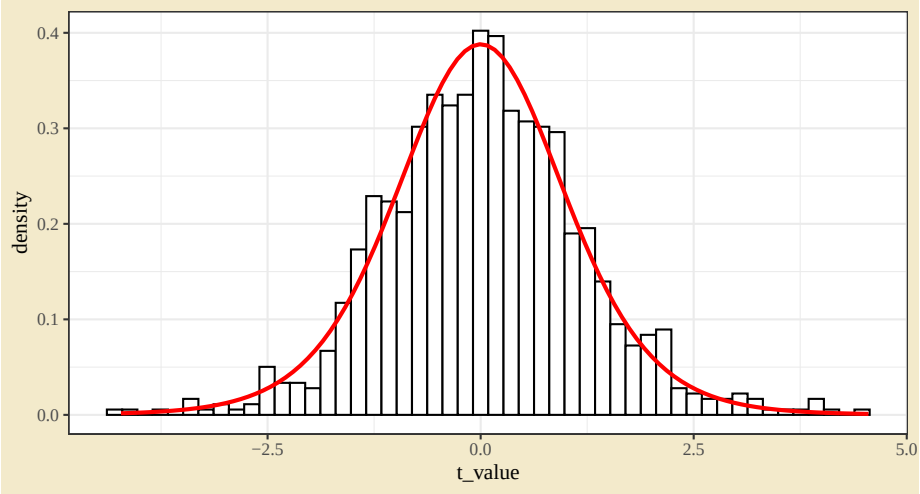
Uppgift 4.21 (Upprepad simulering)**.** Kör de två raderna i stycket ovan ett tiotal gånger. Du bör se att man ibland förkastar nollhypotesen trots att den ska stämma. Kan du få en känsla för hur stor andel av gångerna man felaktigt förkastar?

Låt oss upprepa simuleringen tusen gånger. Ett sätt är att upprepa ett steg flera gånger är genom en for-loop.

```
dat_sim <- data.frame()
for(i in 1:1000){
  new_data <- data.frame(x = rnorm(10, mean = 7, sd = 5))
  test <- t.test(new_data$x, mu = 7)
  new_results <- data.frame(t_value = test$statistic, p_value = test$p.value,
                ci_lower = test$conf.int[1], ci_upper = test$conf.int[2])
  dat_sim <- bind_rows(dat_sim, new_results)
}
```

Enligt statistisk teori ska t-värdet följa en t-fördelning med nio frihetsgrader. Vi kan undersöka det genom ett histogram med en överliggande t-fördelning.

```
ggplot(dat_sim) +
  geom_histogram(aes(t_value, y = ..density..), bins = 50, fill = "white", color = "black") +
  geom_function(fun = dt, args = list(df = 9), color = "red", size = 1)
```



Den teoretiska t-fördelning passar histogrammet nästan perfekt.

Vidare ska p-värdet vara under fem procent fem procent av gångerna.

```
mean(dat_sim$p_value < 0.05)
```

```
## [1] 0.056
```

Även det stämmer någorlunda väl. Det här innebär alltså att om man har en signifikansnivå på fem procent kommer man förkasta nollhypotesen fem procent av gångerna även om nollhypotesen stämmer. Det kallas ett *falskt positivt* utfall.

Uppgift 4.22 (Simulerade konfidensintervall)**.** Hur många av de simulerade konfidensintervallen täcker värdet 7?

Uppgift 4.23 (Signifikant skillnad)**.** Stycket nedan simulerar data när populationsmedelvärdet är 9 och t-test har nollhypotesen att populationsmedelvärdet är 7. Här vill vi alltså förkasta nollhypotesen.

```
dat_sim <- data.frame()
for(i in 1:1000){
  new_data <- data.frame(x = rnorm(10, mean = 9, sd = 5))
  test <- t.test(new_data$x, mu = 7)
  new_results <- data.frame(t_value = test$statistic, p_value = test$p.value,
                ci_lower = test$conf.int[1], ci_upper = test$conf.int[2])
  dat_sim <- bind_rows(dat_sim, new_results)
}
```

Använd kod från den första simuleringen för att undersöka hur väl histogrammet stämmer med den teoretiska fördelningen och för att se hur stor andel av gångerna man förkastar nollhypotesen på signifikansnivån 5 procent.

Uppgift 4.24 (50/50)**.** Stycket nedan simulerar data när populationsmedelvärdet är 9 och t-test har nollhypotesen att populationsmedelvärdet är 7. Ändra värdet för n och se hur det påverkar andelen gånger man förkastar nollhypotesen.

```
n <- 10

dat_sim <- data.frame()
for(i in 1:1000){
  new_data <- data.frame(x = rnorm(n, mean = 9, sd = 5))
  test <- t.test(new_data$x, mu = 7)
  new_results <- data.frame(t_value = test$statistic, p_value = test$p.value,
                ci_lower = test$conf.int[1], ci_upper = test$conf.int[2])
  dat_sim <- bind_rows(dat_sim, new_results)
}

mean(dat_sim$p_value < 0.05)
```

Ungefär hur många observationer behövs för att ha femtio procents sannolikhet att förkasta nollhypotesen?

Uppgift 4.25 (Konfidensintervallets bredd)**.** Ett konfidensintervall blir smalare och smalare ju större stickprovet är. Koden nedan ger medelvärdet för stickprovsbredden i simulerad data med standardavvikelsen 1.

```r
n <- 10

dat_sim <- data.frame()
for(i in 1:1000){
  new_data <- data.frame(x = rnorm(n, mean = 0, sd = 1))
  test <- t.test(new_data$x, mu = 7)
  new_results <- data.frame(t_value = test$statistic, p_value = test$p.value,
                ci_lower = test$conf.int[1], ci_upper = test$conf.int[2])
  dat_sim <- bind_rows(dat_sim, new_results)
}

mean(dat_sim$ci_upper - dat_sim$ci_lower)
```

Ungefär hur många observationer behövs för att konfidensintervallets bredd ska bli under 1, under 0.9, under 0.8, och så vidare ned till 0.1?

Chapter 5

Ett stickprov av icke-normalfördelad data

Datorövning 5 handlar om hypotestest och konfidensintervall för ett stickprov av icke-normalfördelad data. Efter övningen ska vi kunna

- genomföra och tolka ett z-test för proportioner,
- genomföra och tolka ett chi-två-test för nominal data
- beräkna och tolka ett konfidensintervall för proportioner,
- använda simulerad data för att förstå testens egenskaper.

Om det finns tid för en bonussektion kommer vi också titta på interaktiva kartor med `leaflet`.

5.1 Repetition av datorövning 4

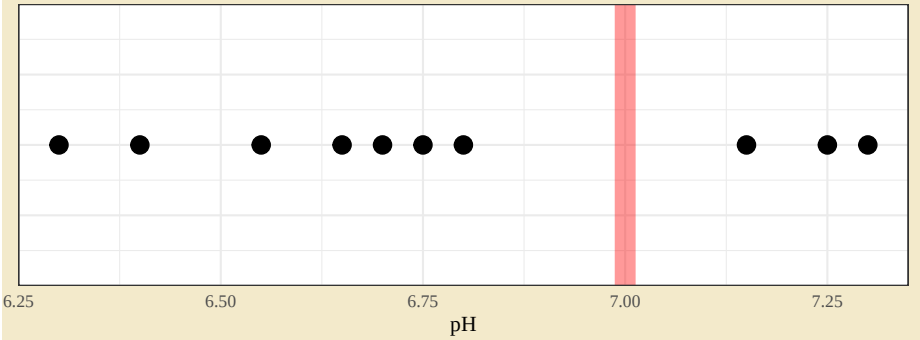
När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 4 tittade vi på analys av ett stickprov av normalfördelad data. Det underliggande upplägget är att vi vill säga något om en population genom att titta på ett stickprov. Ett *t-test* kan användas för att testa en given *nollhypotes*. Ett konfidensintervall ringar in populationens medelvärde med en viss konfidensgrad - oftast 95 procent. Ett normalfördelningsantagande kan undersökas med histogram eller QQ-grafer.

Ta som exempel följande data på jord-pH och låt oss anta att det är relevant att testa om populationens medelvärde är 7.

```
dat_pH <- data.frame(pH = c(6.3, 6.55, 6.75, 6.4, 7.25, 6.65, 6.8, 7.3, 7.15, 6.7))

ggplot(dat_pH, aes(pH, 0)) +
  geom_point(size = 4) +
  geom_vline(xintercept = 7, size = 5, color = "red", alpha = 0.4) +
  theme(axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks = element_blank())
```



Hypoteser ges av

- H0: mu lika med 7
- H1: mu ej lika med 7

Testet kan genomföras med funktionen `t.test()`.

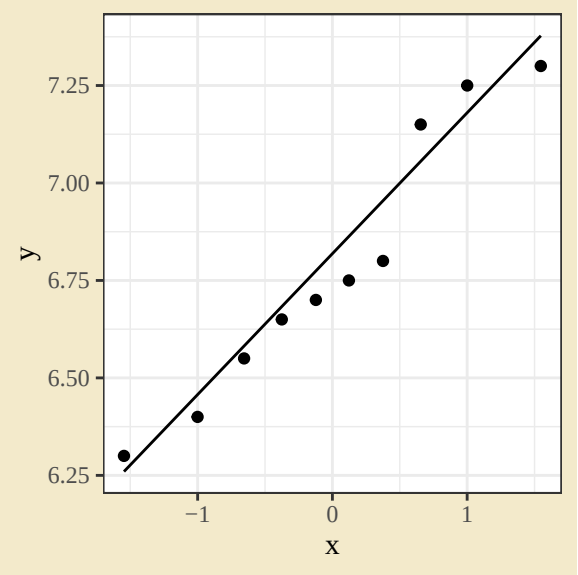
```
t.test(dat_pH$pH, mu = 7)
```

```
##
##  One Sample t-test
##
## data:  dat_pH$pH
## t = -1.9624, df = 9, p-value = 0.08132
## alternative hypothesis: true mean is not equal to 7
## 95 percent confidence interval:
##  6.537164 7.032836
## sample estimates:
## mean of x
##     6.785
```

Ett p-värde över 0.05 ger att vi inte förkastar nollhypotesen - den observerade skillnaden mot 7 är inte statistiskt signifikant. Konfidensintervallet ger att populationens medelvärde ligger mellan 6.54 och 7.03 med 95 procents konfidens.

En QQ-graf kan visa om det finns några avvikelser från normalantagandet. Med små datamängder är det oftast svårt att se några tydliga tecken på icke-normalitet.

```
ggplot(dat_pH, aes(sample = pH)) +
  geom_qq() +
  geom_qq_line()
```

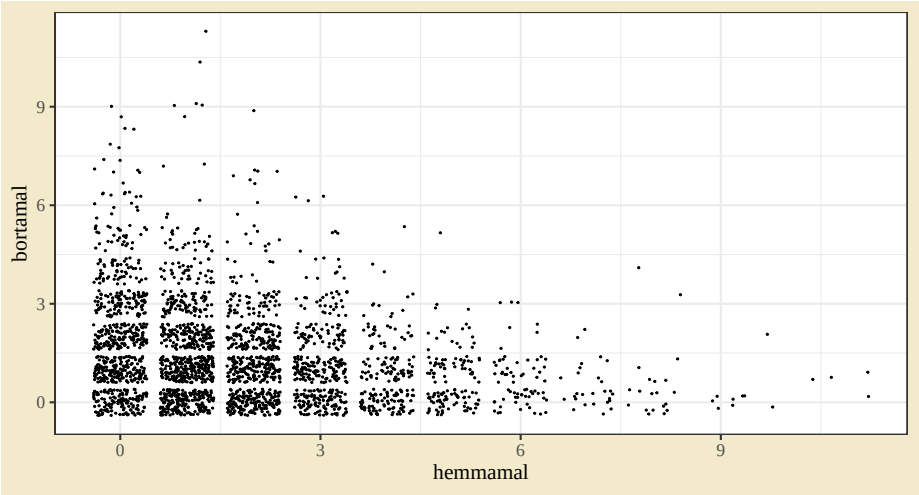


5.2 Proportioner från binär data

Binär data är data där en observation har ett av två utfall, vilka kan kodas som noll och ett. Man talar ibland om utfallet ett som ett *positivt* utfall. Binär data kan sammanfattas med en proportion - antalet positiva utfall delat på det totala antalet upprepningar. En proportion kan testas med ett z-test för proportioner (eller *relativ frekvens*). Testet följer stegen för hypotestest (Hypoteser - Testvärde - Testfördelning - P-värde (eller jämförelse med kritiskt värde) - Slutsats).

Låt oss importera lite exempeldata och beräkna ett exempel. Följande rad importerar matchresultat i fotbollsallsvenskan för damer 2000-2020.

```
library(tidyverse)
dat_alls <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2022/main/Data/Allsvenskan

ggplot(dat_alls, aes(hemmamal, bortamal)) +
  geom_jitter(size = 0.1)
```



Uppgift 5.1 (En interaktiv målgraf)**.** Kör stycket nedan för en interaktiv målgraf. Vilken match gav det högsta antalet gjorda bortamål?

```
# install.packages(plotly)
library(plotly)
g <- ggplot(dat_alls, aes(hemmamal, bortamal, text = paste(sasong, hemma, "-", borta))
  geom_jitter(size = 0.1)

ggplotly(g)
```

Gammal bollkunskap säger att var tredje match är en bortavinst. Vi kan testa det med ett z-test för proportioner. För att ta fram antalet bortasegrar och totalt antal matcher använder vi `count()` på kolumnen `resultat`. Man kan också använda funktionen `table()` för ett liknande resultat.

```
dat_alls %>% count(resultat)
```

```
## # A tibble: 3 x 2
##   resultat       n
##   <chr>      <int>
## 1 Bortaseger   947
## 2 Hemmaseger  1285
## 3 Oavgjort     518
```

```
table(dat_alls$resultat)
```

```
##
## Bortaseger Hemmaseger   Oavgjort
##        947       1285        518
```

Datan har 947 bortasegrar av totalt 947 + 1803 matcher. Vår skattade proportion `p` och totala antal `n` är alltså

```
n <- 947 + 1803
p_est <- 947 / n

p_est
```

```
## [1] 0.3443636
```

```
n
```

```
## [1] 2750
```

För att genomföra ett z-test sätter vi upp hypoteser om proportionen bortasegrar.

- Nollhypotes H0: p lika med 0.33
- Alternativhypotes H1: p ej lika med 0.33

Ett test kan köras i R med `prop.test()`.

```
prop.test(x = 947, n = 2750, p = 0.33, correct = F)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  947 out of 2750, null probability 0.33
## X-squared = 2.5661, df = 1, p-value = 0.1092
## alternative hypothesis: true p is not equal to 0.33
## 95 percent confidence interval:
##  0.3268327 0.3623288
## sample estimates:
##         p
## 0.3443636
```

P-värdet ställs mot en förbestämd signifikansnivå (vanligen 5 procent). I det här fallet leder det höga p-värdet till att nollhypotesen accepteras.

Om vi tar en närmre titt på testets steg börjar vi med att beräkna ett testvärde.

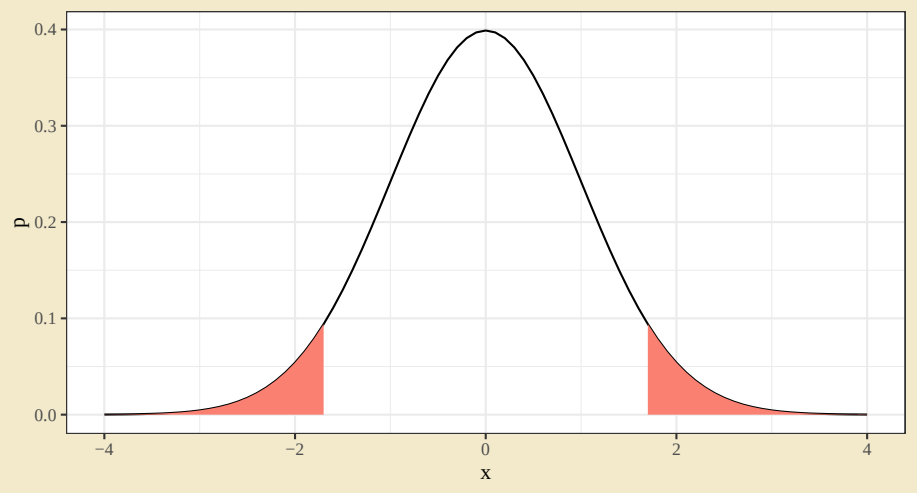
```
p0 <- 0.33
z_value <- (p_est - p0) / sqrt(p0 * (1 - p0) / n)
z_value
```

```
## [1] 1.601904
```

Därefter kan p-värdet räknas ut som arean under en standardiserad normalfördelning bortom z-värdet. Eftersom vi har en tvåsidig mothypotes adderas de två svansarna.

```
dat_norm <- data.frame(x = seq(-4, 4, 0.1)) %>%
  mutate(p = dnorm(x))
```

```
ggplot(dat_norm) +
  geom_line(aes(x, p)) +
  geom_ribbon(aes(x = x, ymin = 0, ymax = p), data = dat_norm %>% filter(x > abs(z_valu
  geom_ribbon(aes(x = x, ymin = 0, ymax = p), data = dat_norm %>% filter(x < -abs(z_val
```



Areans yta kan tas fram med normalfördelningens fördelningsfunktion `pnorm()`.

```
2 * pnorm(-z_value)
```

```
## [1] 0.1091769
```

Testets p-värde är ungefär 11 procent. Vår observation är alltså inte ett orimligt utfall om den faktiska sannolikheten för bortaseger är 0.33 och vi kan inte förkasta nollhypotesen på femprocentsnivån.

Om man löste uppgiften för hand skulle man istället för att beräkna p-värdet jämföra z-värdet med ett kritisk värde ur en tabell. Det kritiska värdet för femprocentig signifikans är 1.96. Vi kan också ta fram det genom `qnorm(0.975)`.

Vi kan jämföra den beräkning med den direkta R-funktionen.

```
prop.test(x = 947, n = 2750, p = 0.33, correct = F)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  947 out of 2750, null probability 0.33
## X-squared = 2.5661, df = 1, p-value = 0.1092
## alternative hypothesis: true p is not equal to 0.33
## 95 percent confidence interval:
##  0.3268327 0.3623288
## sample estimates:
```

```
##         p
## 0.3443636
```

Den stegvisa beräkningen gav samma utfall som funktionen (`p-value = 0.01092`). Funktionen ger inte z-värdet utan ett chi-två-värde (2.5661). Här är det värdet lika med z-värdet i kvadrat.

```
z_value^2
```

```
## [1] 2.566095
```

Uppgift 5.2 (Ensidigt test)**.** Titta på hjälpsidan med `?prop.test`. Hur genomför man ett ensidigt test? Gör lämpligt tillägg för att testa om andelen bortasegrar är större än 0.33.

Uppgift 5.3 (Test för proportionen oavgjorda)**.** Samma gamla bollkunskap säger att 20 procent av matcher blir oavgjorda. I datan är 518 av 2750 matcher oavgjorda. Ställ upp hypoteser och fyll i koden nedan för att testa om bollkunskapen stämmer.

```
prop.test(x = ___, n = ___, p = ___, correct = F)
```

Uppgift 5.4 (Test för proportionen hemmasegrar)**.** Slutligen är då resten av matcherna, 1285 av 2750, hemmasegrar. Gammal bollkunskap säger: *47 procent av alla matcher är hemmasegrar*. Genomför ett z-test för att testa det påstående.

```
prop.test(x = ___, n = ___, p = ___, correct = F)
```

Uppgift 5.5 (Population och stickprov)**.** Ett hypotestest bygger på en underliggande tanke med en population (med någon för oss okänd proportion positiva utfall) och ett stickprov (i vilket vi kan observera andelen positiva utfall). Det är inte alltid uppenbart vad som egentligen är populationen. I fallet med fotbollsdatan, vad kan ses som populationen? Hur långt skulle man kunna generalisera de slutsatser man kan dra från datan?

Uppgift 5.6 (Guldfiskgenetik)**.** (Fråga från Olsson, *Biometri*) En teori inom genetik förutsäger att tre fjärdedelar i en grupp guldfiskar ska ha genomskinliga fjäll. Observationer ger att nittio av hundra har genomskinliga fjäll. Genomför ett test med `prop.test()` för att se om den faktiska proportionen skiljer sig från 0.75. Lös gärna först uppgiften för hand eller med miniräknare.

Uppgift 5.7 (Mer guldfiskgenetik)**.** (Fråga från Olsson, *Biometri*) En konkurrerande teori inom genetik förutsäger att femton sextondelar (proportionen 0.9375) ska ha genomskinliga fjäll. Observationer ger att nittio av hundra har genomskinliga fjäll. Genomför ett test med `prop.test()` för att se om proportionen skiljer sig från 0.9375. Lös gärna först uppgiften för hand eller med miniräknare.

Hypotestestet för proportioner som används här, *z-testet*, bygger på en normalapproximation av en binomialfördelning. Approximation blir bättre när antalet observationer är stort och nollhypotesens värde p0 ligger nära 0.5. En

vanlig tumregel för när approximationen är giltig är att n gånger p0 gånger (1 - p0) ska vara större än 10. För fotbollsdatan över oavgjorda matcher ger det 2750 * 0.2 * 0.8 vilket är klart större än 10.

Uppgift 5.8 (Giltig approximation)**.** I det andra guldfiskexemplet är antalet observationer 100 och nollhypotesens värde 0.9375. Är normalapproximationen *giltig* i det fallet?

Uppgift 5.9 (Fågelproportioner)**.** I ett naturreservat tror man fördelningen av tre fåglar (tärnmås, fiskmås och fisktärna) är 50, 30 respektive 20 procent. En studie ger antalen 115, 54 respektive 31. Genomför tre z-test för att testa den antagna andelarna. För tärnmås får man till exempel `prop.test(115, 200, p = 0.5, correct = F)`.

Uppgift 5.10 (Förbestämd signifikans)**.** En intressant egenskap hos proportionstest är att man redan i förväg kan beräkna vilka utfall som ger signifikanta resulat. Säg att man har möjlighet att göra 100 replikat. Ändra i stycket nedan för att hitta det högsta värde på x som ger ett *icke*-signifikant utfall.

```
prop.test(x, n = 100, p = 0.5, correct = F)
```

Det är möjligt att göra liknande beräkningar för ett t-test för normalfördelad data, men då måste man göra antaganden om standardavvikelsens storlek.

5.3 Konfidensintervall för proportioner

Konstruktionen av ett konfidensintervall för en proportion är ganska lik konstruktionen för ett medelvärde. För en skattad proportion p och antal observationer n kan man beräkna p plus/minus ett z-värde från tabell gånger medelfelet, där medelfelet ges av roten ur p * (1 - p) / n. För exemplet med bortasegrar i allsvenskan är p = 0.344 och n = 2750. Tabellvärdet hämtas från en tabell över kvantiler. För ett 95-procentigt konfidensintervall tar vi kvantilen 0.975 (2.5 procent i respektive svans) vilket ger värdet 1.96. Konfidensintervallet ges av

```
n <- 947 + 1803
p <- 947 / n

p - 1.96 * sqrt(p * (1 - p) / n)
```

```
## [1] 0.3266042
```

```
p + 1.96 * sqrt(p * (1 - p) / n)
```

```
## [1] 0.3621231
```

Notera att 0.33, det värde som var nollhypotesen i det tidigare testet, *ingår* i intervallet. Om man tittar på utskriften från `prop.test()` kan man se ett konfidensintervall. Det intervallet är dock inte beräknat på samma sätt den

formel som förekommer på föreläsningarna. För att få matchande utskrift kan vi använda paketet `binom` och funktionen `binom.asymp()`.

```
#install.packages("binom")
library(binom)
binom.asymp(x = 947, n = 2750)
```

```
##       method   x    n      mean     lower     upper
## 1 asymptotic 947 2750 0.3443636 0.3266045 0.3621228
```

Uppgift 5.11 (99-procentigt konfidensintervall)**.** Gör lämplig ändring i koden nedan för att beräkna ett 99-procentigt konfidensintervall för andelen bortasegrar.

```
binom.asymp(x = 947, n = 2750, conf.level = 0.95)
```

Uppgift 5.12 (Perfekta utfall)**.** Vad händer om man försöker räkna ut ett konfidensintervall för ett perfekt utfall - t.ex. om man får 100 av 100 positiva utfall?

Uppgift 5.13 (Konfidensintervall för guldfiskar)**.** Använd funktionen `binom.asymp()` för att ta fram konfidensintervallet för andelen guldfiskar från den tidigare uppgiften. Hur förhåller sig resultatet till nollhypotesernas värden (0.75 respektive 0.9375)? Gör motsvarande beräkning med miniräknare.

5.4 Chi-två-test för goodness-of-fit

Ett proportionstest kan ses som ett test av en variabel med två möjliga klasser som utfall. Ett *goodness-of-fit*-test utvecklar det till valfritt antal klasser. Testet utförs som ett chi-två-test genom att beräkna ett observerat antal O och ett förväntat antal E för varje klass. Testvärdet ges av att man beräknar (O - E)^2 / E för varje klass och sedan summerar. Testfunktionen är en chi-två-fördelning där antalet frihetsgrader beror på antalet klasser.

Låt oss göra ett exempel baserat på fotbollsdatan. Där hade vi utfallet 947, 518 och 1285 för bortaseger, oavgjort och hemmaseger. Klassisk bollkunskap gav oss sannolikheterna 33, 20 och 47 procent. Testets hypoteser ges av

H0: sannolikheterna för de olika utfallet ges av 33, 20 respektive 47 procent

H1: minst något utfall har en annan sannolikhet än 33, 20 respektive 47 procent

För att få de förväntade värdena E multipliceras nollhypotesens sannolikheter med det totala antalet matcher.

```
O <- c(947, 518, 1285)
E <- c(0.33,0.20,0.47) * 2750
```

Uppgift 5.14 (Granska E)**.** Skriv ut objektet `E` och jämför med de observerade värdena. Notera att de förväntade värdena inte måste vara heltal, trots att de

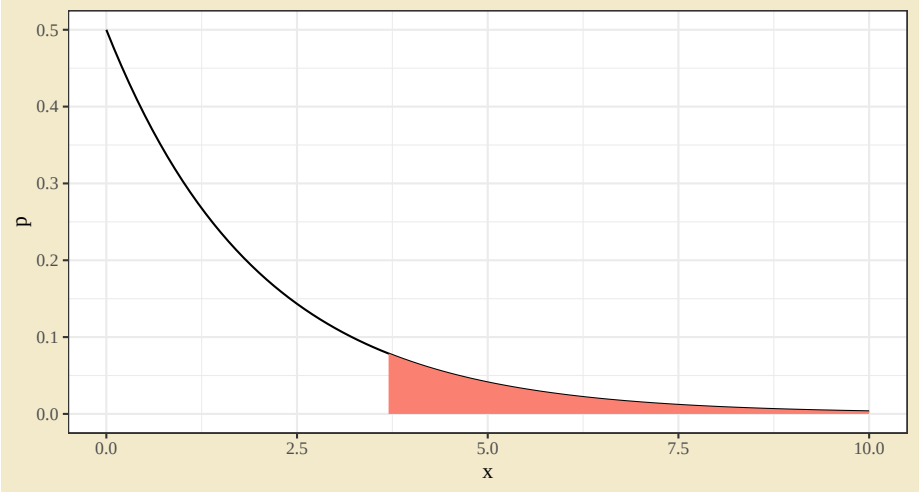
observerade värdena förstås alltid kommer vara det.

Testvärdet beräknas genom formeln för varje term följt av summan.

```
chisq_value <- sum((O - E)^2 / E)
```

P-värdet beräknas från en chi-två-fördelning. Antalet frihetsgrader ges av antalet klasser minus antalet skattade parametrar minus ett. I det här fallet har inga parametrar skattats från datan så antalet frihetsgrader blir två. Ett chi-två-test beräknas med kvadrater så vi är enbart intresserade av högra svansen.

```
dat_chisq <- data.frame(x = seq(0, 10, 0.1)) %>%
  mutate(p = dchisq(x, df = 2))

ggplot() +
  geom_line(aes(x, p), data = dat_chisq) +
  geom_ribbon(aes(x, ymin = 0, ymax = p), data = dat_chisq %>% filter(x > chisq_value)
```



Man kan också beräkna ytan i svansen med `pchisq()`. Ett minus det resultatet ger den övre delen.

```
1 - pchisq(chisq_value, df = 2)
```

```
## [1] 0.1632763
```

P-värdet är alltså 0.16, över den klassiska signifikansnivån på 5 procent, vilket ger att vi inte kan förkasta nollhypotesen. Om man gör ett chi-två-test för hand jämför man det observerade chi-två-värdet med ett tabellvärde över kvantiler. Tabellvärdet kan också hämtas med funktionen `qchisq()`, i det här fallet

```
qchisq(0.95, df = 2)
```

```
## [1] 5.991465
```

Notera att man tar 0.95 eftersom man alltid tittar på den yttre svansen i ett chi-två-test.

R har en inbyggd funktion för chi-två-test. Dess argument ges av observerade antal och sannolikheter.

```
chisq.test(O, p = c(0.33, 0.2, 0.47))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  O
## X-squared = 3.6246, df = 2, p-value = 0.1633
```

Testet ger samma chi-två-värde och p-värde som beräknats ovan.

Uppgift 5.15 (Chi-två med två klasser)**.** Situationen med flera klasser kan som sagt ses som en generalisering av fallet med två klasser. Det är alltså logiskt att chi-två-test kan användas även när man har två klasser. Följande exempel ger samma test som vi sett tidigare av andelen bortasegrar.

```
chisq.test(x = c(947, 1803), p = c(0.33, 0.67), correct = F)
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(947, 1803)
## X-squared = 2.5661, df = 1, p-value = 0.1092
```

Likt `prop.test()` sätter vi `correct` till `FALSE` för att inte göra en korrektion. Notera att `x` här anges som positiva och negativa utfall istället för positiva utfall och totalt antal utfall, vilket var fallet i `prop.test()`.

Använd stycket ovan som mall för att göra uppgiften om guldfiskar som ett chi-två-test. Testa nollhypotesen att andelen positiva utfall är 0.75.

Chi-två-testet bygger på en underliggande normal-liknande approximation. En vanlig tumregel är att alla förväntade värden ska vara större än 5. R ger en varning om så inte är fallet.

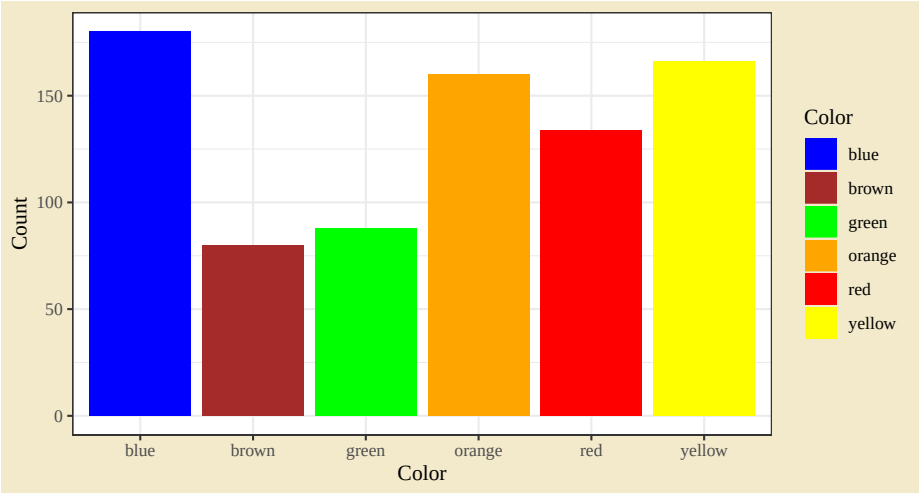
```
chisq.test(c(6,4), p = c(0.51, 0.49))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(6, 4)
## X-squared = 0.32413, df = 1, p-value = 0.5691
```

Uppgift 5.16 (Chi-två med lika sannolikheter)**.** En vanlig tillämpning av goodness-of-fit-testet är för att testa om alla klasser är lika sannolika. En jämn fördelning är grundinställning i `chisq.test()` så i det fallet behöver man bara

ange de observerade värdena. En datainsamling om M&M-godis gav följande antal.

```
dat_mnm <- data.frame(Color = c("blue", "brown", "green", "orange", "red", "yellow"),
                      Count = c(180, 80, 88, 160, 134, 166))

ggplot(dat_mnm, aes(Color, Count, fill = Color)) +
  geom_col() +
  scale_fill_manual(values = dat_mnm$Color)
```



Använd de observerade värdena i kolumnen `Count` för att testa om alla godisfärger är lika vanliga.

Om man har flera klasser kan det vara värdefullt att se vilken klass som bidrar mest till chi-två-värdet. Det kan ge en bild av vilka klasser som är mest speciella. Ett enkelt sätt att göra det på är att spara utfallet av `chisq.test()` som ett objekt och därifrån hämta `observed` och `expected`. Som fortsättning på fotbollsexemplet:

```
test <- chisq.test(c(947, 518, 1285), p = c(0.33, 0.2, 0.47))
test$expected
```

```
## [1]  907.5  550.0 1292.5
```

```
test$observed
```

```
## [1]  947  518 1285
```

```
(test$observed - test$expected)^2 / test$expected
```

```
## [1] 1.71928375 1.86181818 0.04352031
```

Antalet hemmavinster (det tredje värdet) ligger när den teoretiska sannolikheten medan de övriga två utfallen ligger längre från.

Uppgift 5.17 (Udda färger)**.** Spara testobjektet från testet på M&M-färger för att se vilka färger som avviker mest från det väntade utfallet.

Uppgift 5.18 (Fågelproportioner som chi-två)**.** I naturreservatet från en tidigare uppgift tror man fördelningen av tre fåglar (tärnmås, fiskmås och fisktärna) är 50, 30 respektive 20 procent. En studie ger antalen 115, 54 respektive 31. Genomför ett chi-två-test för att testa de antagna andelarna. Är resultatet i linje med de separata testen från den tidigare uppgiften?

Uppgift 5.19 (Udda fåglar)**.** Spara testobjektet från chi-två-testet för fåglarna för att se vilka fågelarter som avviker mest från det väntade utfallet.

5.5 Chi-två-test när någon parameter skattas från datan

(Det här stycket är överkurs och kan läsas översiktligt eller hoppas över.)

I de exempel vi sett hittills har nollhypotesen direkt givit de sannolikheter vi vill testa. Ett annat vanligt fall är att man testar om värdena följer en viss fördelning, men parametervärden i den fördelningen skattas från den insamlade datan. Ta som exempel frågan om antal mål per match följer en poissonfördelning. Hypoteserna ges av

H0: antal mål följer en poissonfördelning,

H1: antal mål följer ej en poissonfördelning.

Här måste vi skatta medelvärdet från datan för att beräkna sannolikheter från fördelningen.

```
mean_goals <- mean(dat_alls$hemmamal + dat_alls$bortamal)
mean_goals
```

```
## [1] 3.290182
```

```
dat_goals <- dat_alls %>%
  count(Mål = bortamal + hemmamal, name = "O") %>%
  mutate(p = dpois(Mål, lambda = mean_goals),
         E = p * 2750)
dat_goals
```

```
## # A tibble: 13 x 4
##      Mål     O        p       E
##    <dbl> <int>    <dbl>   <dbl>
##  1     0   138 0.0372   102.
##  2     1   383 0.123    337.
##  3     2   548 0.202    554.
##  4     3   542 0.221    608.
##  5     4   466 0.182    500.
```

```
##  6     5   306 0.120    329.
##  7     6   183 0.0656   180.
##  8     7    97 0.0308    84.8
##  9     8    42 0.0127    34.9
## 10     9    29 0.00464   12.8
## 11    10     6 0.00153    4.20
## 12    11     5 0.000456   1.26
## 13    12     5 0.000125   0.344
```

En graf kan illustrera faktiska antal (som punkter) och den skattade poissonfördelningen (som linje).

Uppgift 5.20 (Målgraf)**.** Fyll i de saknade delarna i koden nedan för en graf med faktiska antal `O` som punkter och förväntade antal `E` som en linje.

```
ggplot(dat_goals) +
  geom_point(aes(x = Mål, y = ___)) +
  geom_line(aes(x = Mål, y = ___))
```

De faktiska observationerna är inte så långt från poissonfördelningen.

Ett problem med de förväntade antalen är att några av dem är under 5 - den gräns vi satte för en acceptabel chi-två-approximation. Det vanligaste sättet att hantera det är att slå ihop klasser. Här slås klasser över 9 ihop till en grupp. Det är inte så väsentligt hur det görs här, även om `ifelse()` kan vara ett nyttigt trick.

```
dat_goals_merged <- dat_goals %>%
  mutate(Mål = ifelse(Mål > 9, 10, Mål)) %>%
  group_by(Mål) %>%
  summarise(O = sum(O),
            p = sum(p),
            E = sum(E))
dat_goals_merged
```

```
## # A tibble: 11 x 4
##      Mål     O       p      E
##    <dbl> <int>   <dbl>  <dbl>
##  1     0   138 0.0372  102.
##  2     1   383 0.123   337.
##  3     2   548 0.202   554.
##  4     3   542 0.221   608.
##  5     4   466 0.182   500.
##  6     5   306 0.120   329.
##  7     6   183 0.0656  180.
##  8     7    97 0.0308   84.8
##  9     8    42 0.0127   34.9
## 10     9    29 0.00464  12.8
```

```
## 11    10    16 0.00211   5.80
```

En sista svårighet är antalet frihetsgrader. I ett goodness-of-fit-test ges antalet frihetsgrader av antalet klasser minus antalet skattade parametrar minus 1. I det här fallet har vi nu 11 klasser och vi har skattat en parameter. Vi ska alltså ha 9 frihetsgrader i testet. Funktionen `chisq.test()` beräknar tyvärr antalet frihetsgrader internt som antalet klasser minus 1, så vi får beräkna chi-två-värde och p-värde på egen hand.

```
chisq_value <- sum((dat_goals_merged$O - dat_goals_merged$E)^2 / dat_goals_merged$E)
1 - pchisq(chisq_value, df = 9)
```

```
## [1] 6.984746e-12
```

Chi-två-värdet blir extremt stort och p-värdet väldigt lågt. Nollhypotesen att antalet mål följer en poissonfördelning förkastas.

5.6 Bonus. Interaktiva kartor med leaflet

Paketet `leaflet` (https://rstudio.github.io/leaflet/) kopplar R till leaflet - ett verktyg för interaktiva kartor som ofta använd för kartor online.

Uppgift 5.21 (Installera leaflet)**.** Installera och ladda 'leaflet genom att fylla i och köra raden nedan.

```
install.packages("leaflet")
library(leaflet)
```

Som en första kontroll kan vi köra den exempelkod som ges på hemsidan länkad till ovan.

```
m <- leaflet() %>%
  addTiles() %>%
  addMarkers(lng = 174.768, lat = -36.852, popup="The birthplace of R")
m
```

På canvassidan finns en excelfil med data tillgänglig på Artportalen - *Artportalen, SLU feromoninventering, 2011-2013.xlsx*. Datan kommer från från ett inventeringsprojekt vid SLU.

Uppgift 5.22 (Importera datan)**.** Ladda ner excelfilen och läs in datan med `read_excel()` från paketet `readxl`.

```
library(readxl)
read_excel("___")
```

Kartans utseende kan ändras genom att ange en källa och karttyp. Tillgängliga alternativ kan skrivas ut med `providers`.

```
dat_leaf %>%
  leaflet() %>%
  addTiles() %>%
  addProviderTiles(providers$Stamen.Toner)
```

Uppgift 5.23 (Baskarta)**.** Skriv ut tillgängliga baskartor med `providers`. Välj ett alternativ slumpmässigt och ändra koden ovan för att se hur det ser ut.

För att lägga till datapunkter kan man använda `addCircleMarkers()`.

```
dat_leaf %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(lng = dat_leaf$lng, lat = dat_leaf$lat, radius = 10)
```

Uppgift 5.24 (Cirkelstorlek)**.** Ändra storleken på cirklarna från `addCircleMarkers()` genom argumentet `radius`

Slutligen kan vi lägga till en etikett för år med argumentet `popup`. Texten kommer upp när man klickar på en punkt.

```
dat_leaf %>%
  leaflet() %>%
  addTiles() %>%
  addCircleMarkers(lng = dat_leaf$lng, lat = dat_leaf$lat, radius = 10, popup = dat_lea
```

Uppgift 5.25 (Artnamn)**.** Ändra i koden ovan för att ange artnamn som popup-text istället för rödlistestatus. Funktionen `paste()` kan också vara intressant för att ta med mer information i texten: `paste(dat_leaf$Rödlistade, dat_leaf$Antal)` skulle exempelvis ge både status och antal individer vid den specifika observationen.

Chapter 6

Test för två stickprov

Datorövning 6 handlar om hypotestest och konfidensintervall för jämförelse av två stickprov. Efter övningen ska vi kunna genomföra och tolka

- t-test för jämförelse av två medelvärden,
- z-test för jämförelse av två proportioner,
- chi-två-test för jämförelse fördelningar mellan två eller flera grupper,
- konfidensintervall för skillnaden mellan två medelvärden eller två proportioner.

6.1 Repetition av datorövning 5

När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 5 tittade vi på tester som inte bygger på normalfördelning: dels tester för proportioner, där varje observation är något av två möjliga utfall; dels tester för kategoridata, där varje observation har ett ufall i någon av flera kategorier.

Proportioner kan testas med ett z-test. Säg att man testar om en doft har en effekt på en insekt, man skickar 20 insekter i ett y-rör och 16 går mot doften. Hypoteserna ges av

- H0: proportionen i populationen är 0.5,
- H1: proportionen i populationen är inte 0.5.

I R genomförs ett z-test med `prop.test()`. Funktionen gör en korrektion som ger ett något bättre test än det man ofta beräknar för hand. Korrektionen kan sättas av med argumentet `correct`.

```
prop.test(x = 16, n = 20, p = 0.5, correct = F)
```

```
##
##  1-sample proportions test without continuity correction
##
## data:  16 out of 20, null probability 0.5
## X-squared = 7.2, df = 1, p-value = 0.00729
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.5839826 0.9193423
## sample estimates:
##   p
## 0.8
```

Ett lågt p-värde tyder på en signifikant skillnad från 0.5.

För att få samma konfidensintervall som det man beräknar för hand kan man använda `binom.asymp()` från paketet `binom`.

```
library(binom)
binom.asymp(16, 20)
```

```
##       method  x  n mean     lower     upper
## 1 asymptotic 16 20  0.8 0.6246955 0.9753045
```

Populationens proportion ligger med 95 procents konfidens mellan 0.62 och 0.98.

Om utfallen är mer än två kategorier kan en hypotes om datans fördelning testas med ett goodness-of-fit-test, vilket är ett slags chi-två-test. Testet bygger på att observerade antal (O) ställs mot förväntade antal (E). I R genomförs ett chi-två-test med `chisq.test()`. Säg som exempel att man studerar fågelpopulationer genom en observationsstudie. Observerade antal av fyra fågelarter är 102, 53, 75 och 12 och de förväntade andelarna av arterna är 50, 15, 25 och 10 procent. Testets hypoteser ges av

- H0: proportionerna följer den förväntade fördelningen,
- H1: proportionerna följer inte den förväntade fördelningen,

och testet ges av

```
chisq.test(c(102, 53, 75, 12), p = c(0.5, 0.15, 0.25, 0.1))
```

```
##
##  Chi-squared test for given probabilities
##
## data:  c(102, 53, 75, 12)
## X-squared = 20.292, df = 3, p-value = 0.0001477
```

Ett lågt p-värde tyder på att de antagna proportionerna inte stämmer. Antalet frihetsgrader ges i ett chi-två-test av antalet klasser minus antalet skattade parametrar minus ett. I det här fallet finns fyra klasser och ingen skattad parameter.

6.2 Två stickprov och normalfördelad data

Vid normalfördelad data från två stickprov eller grupper vill vi nästan alltid testa om populationerna har samma medelvärde. Det kan också ses som att vi testar om differensen mellan medelvärdena är noll. Vi skiljer mellan två fall: *matchade stickprov* - där varje observation i den ena gruppen är *kopplad* till en observation i den andra gruppen; och *oberoende stickprov* - där det inte finns någon sådan koppling mellan stickproven. Typiska exempel på matchade stickprov är när man mäter samma individ för och efter en behandling och syskonstudier där ett syskon får en behandling och den andra en annan behandling.

6.2.1 t-test för två matchade stickprov

Vid matchade stickprov kan varje observation i en behandlingsgrupp paras med en observation i den andra gruppen. Själva testet är ett t-test för *ett* stickprov på differensserien beräknat från varje par. I R kan man antingen beräkna den differensserien eller använda `t.test()` med två dataserier och argumentet för parvisa observationer satt till sant, `paired = T`. Som exempel ges följande data från en studie på äpple, där trädhöjd mätts före och efter en näringsbehandling.

```
dat_apple <- tibble(Tree = 1:4,
                    Before = c(48, 43, 30, 47),
                    After = c(51, 44, 42, 54))
dat_apple
```

```
## # A tibble: 4 x 3
##    Tree Before After
##   <int>  <dbl> <dbl>
## 1     1     48    51
## 2     2     43    44
## 3     3     30    42
## 4     4     47    54
```

Datan kan illustreras med ett punktdiagram där en linje binder samman paret. För att enkelt skapa grafen i `ggplot2` kan man först omstrukturera datan till lång form genom `pivot_longer`.

```
dat_long <- dat_apple %>% pivot_longer(-Tree, names_to = "Time", values_to = "Height")
dat_long
```

```
## # A tibble: 8 x 3
##    Tree Time   Height
```

```
##   <int> <chr>   <dbl>
## 1     1 Before     48
## 2     1 After      51
## 3     2 Before     43
## 4     2 After      44
## 5     3 Before     30
## 6     3 After      42
## 7     4 Before     47
## 8     4 After      54
```

Uppgift 6.1 (Äppelgraf)**.** Fyll i kodstycket nedan för en graf av äppeldatan. Axlarna ges av `Time` och `Height`. Två observationer kan kopplas genom att sätta `Tree` som grupp.

```
ggplot(dat_long, aes(___, ___, group = ___)) +
  geom_point() +
  geom_line()
```

För att testa för skillnad före och efter behandling sätter vi upp hypoteser

- H0: mu före behandling är lika med mu efter behandling
- H1: mu före behandling är skild från mu efter behandling

Testet kan antingen utföras som ett enkelt t-test på differensserien

```
t.test(dat_apple$Before - dat_apple$After)
```

```
##
##  One Sample t-test
##
## data:  dat_apple$Before - dat_apple$After
## t = -2.3681, df = 3, p-value = 0.09868
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -13.477405   1.977405
## sample estimates:
## mean of x
##     -5.75
```

eller som ett t-test för två stickprov där man särskilt anger att datan är parad

```
t.test(dat_apple$Before, dat_apple$After, paired = T)
```

```
##
##  Paired t-test
##
## data:  dat_apple$Before and dat_apple$After
## t = -2.3681, df = 3, p-value = 0.09868
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
```

```
##  -13.477405   1.977405
## sample estimates:
## mean difference
##           -5.75
```

För bägge alternativen måste datan vara ordnad så att de två vektorerna matchar varandra parvis. Ett p-värde på 0.0987 ger att man inte förkastar vid en signifikansnivå på fem procent. Vi drar därmed slutsatsen att det inte finns någon signifikant skillnad före och efter behandling.

Uppgift 6.2 (Ensidigt test)**.** Gör ett tillägg till ett av kodstyckena med `t.test()` för att beräkna ett ensidigt test med mothypotesen att träden ökar i höjd efter behandling. Hjälpsidan för `t.test()` kan tas fram genom att köra `?t.test()`.

Konfidensintervallet beräknas från differenserna på samma sätt som vid ett stickprov med normalfördelad data. Tolkningen liknar den för ett stickprov: med 95 procents konfidens ligger den sanna skillnaden i medelvärden i intervallet.

Uppgift 6.3 (Lökimport)**.** Åtta monoglukosidmätningar på lök samlas in från fyra konventionella och fyra ekologiska ordlare. Resultatet finns i fliken *Lökfärg* i excelfilen *Uppgiftsdata.xlsx* på canvassidan. Ladda ner filen och importera datan genom att fylla i raden nedan.

```
library(readxl)
dat_onion <- read_excel("____", sheet = "Lökfärg")
# dat_onion <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2023/main/Data/Uppgifts
```

Uppgift 6.4 (Lökgraf)**.** Fyll i stycket nedan för en graf av lökdatan från föregående uppgift.

```
dat_long <- dat_onion %>%
  pivot_longer(-Odlare, names_to = "Odlingstyp", values_to = "Utfall")
dat_long

ggplot(dat_long, aes(___, ___, group = Odlare)) +
  geom_point() +
  geom_line()
```

Tyder grafen på någon skillnad mellan odlingstyper?

Uppgift 6.5 (Löktest)**.** Använd lökdatan i föregående uppgift för att testa om det finns en signifikant skillnad mellan konventionell och ekologisk. Formulera hypoteser och genomför testet med `t.test()`. Lös gärna uppgiften med miniräknare först.

6.2.2 t-test för två oberoende stickprov

Ett t-test för två oberoende stickprov testar om två populationsmedelvärden är lika. Ta som exempel följande data på jordgubbsskörd vid två olika närings-

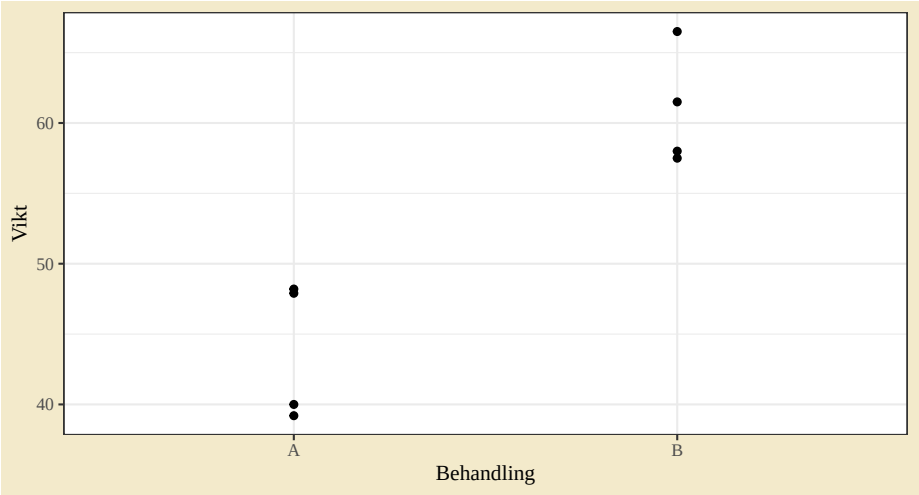
behandlingar (A och B). Här är stickproven inte matchade - det finns **ingen** direkt koppling mellan en observation i den ena behandlingsgruppen till någon observation i den andra.

```
dat_berry <- data.frame(Behandling = c("A", "A", "A", "A", "B", "B", "B", "B"),
                        Vikt = c(40, 48.2, 39.2, 47.9, 57.5, 61.5, 58, 66.5))
dat_berry
```

```
##   Behandling Vikt
## 1          A 40.0
## 2          A 48.2
## 3          A 39.2
## 4          A 47.9
## 5          B 57.5
## 6          B 61.5
## 7          B 58.0
## 8          B 66.5
```

Datan kan illustreras med ett enkelt punktdiagram.

```
ggplot(dat_berry, aes(Behandling, Vikt)) +
  geom_point()
```



Ett t-test för två oberoende stickprov har nollhypotesen att grupperna har samma populationsmedelvärde och alternativhypotesen att populationsmedelvärdena är skilda (för det tvåsidiga fallet):

- H0: mu_A är lika med mu_B
- H1: mu_A är ej lika med mu_B

Testet kan utföras i R genom funktionen `t.test()`. Data kan antingen anges som en formel med dess data `Vikt ~ Behandling, data = dat_berry` (vilket

man kan läsa som *vikt uppdelat efter behandling*) eller som två skilda vektorer. Det förra alternativet är oftast enklare om man har datan på lång form - med en kolumn som anger grupp (i exemplet *Behandling*) och en kolumn som anger utfallsvärdet (i exemplet *Vikt*).

För formen med formel ger det

```
# Formelskrivning
t.test(Vikt ~ Behandling, data = dat_berry, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  Vikt by Behandling
## t = -5.3157, df = 6, p-value = 0.001803
## alternative hypothesis: true difference in means between group A and group B is not equal to 0
## 95 percent confidence interval:
##  -24.898417  -9.201583
## sample estimates:
## mean in group A mean in group B
##          43.825          60.875
```

och för formen med vektorer

```
# Två separata vektorer
## Filtrera ut data där behandling är A
Vikt_A <- dat_berry$Vikt[dat_berry$Behandling == "A"]

## Filtrera ut data där behandling är B
Vikt_B <- dat_berry$Vikt[dat_berry$Behandling == "B"]

t.test(Vikt_A, Vikt_B, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  Vikt_A and Vikt_B
## t = -5.3157, df = 6, p-value = 0.001803
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -24.898417  -9.201583
## sample estimates:
## mean of x mean of y
##    43.825    60.875
```

Argumentet `var.equal = T` används för att beräkna testet där gruppernas varianser antas vara lika. Grundinställningen är testet där varianser inte antas vara lika, så `t.test(Vikt ~ Behandling, data = dat)` ger ett lite annat resultat.

Uppgift 6.6 (Ej lika varianser)**.** Vilka resultatvärden ändras i utskriften om man sätter `var.equal = F`?

Testet ger ett p-värde på 0.0018, vilket leder till att nollhypotesen förkastas på enprocentsnivån. Detta tyder på att det finns en viktskillnad mellan behandlingarna. Utskriften ger också ett 95-procentigt konfidensintervall på $(-24.898, -9.202)$. Tolkningen är att skillnaden mellan populationsmedelvärden ligger i intervallet med 95 procents konfidens. Notera att värdet noll inte ligger i intervallet.

Uppgift 6.7 (Ensidigt test)**.** Gör lämpliga tillägg till kodstycket nedan för att göra ett ensidigt test (om B ger högre vikt än A).

```
t.test(Vikt ~ Behandling, data = dat_berry, var.equal = T, alternative = "two.sided")
```

Om man har fler än två grupper kan man vilja göra parvisa t-test - alltså ett t-test för varje par av grupper. Ett exempel på funktionen `pairwise.t.test()` ges nedan. Funktionen bygger på att datan är i *lång* form, med en kolumn som anger det numeriska utfallet och en kolumn som anger behandlingen.

```
pairwise.t.test(dat_berry$Vikt, dat_berry$Behandling, p.adjust.method = "none", pool.s
```

```
##
##  Pairwise comparisons using t tests with non-pooled SD
##
## data:  dat_berry$Vikt and dat_berry$Behandling
##
##   A
## B 0.002
##
## P value adjustment method: none
```

Matchade observationer kan också kallas *parade* (eng. paired) så se upp med terminologin. Funktionen `pairwise.t.test()` för *parvisa jämförelse* mellan behandlingar, men testerna är t-test för oberoende stickprov.

Uppgift 6.8 (Ekorrdata)**.** I en undersökning av hur den europeiska ekorren (Sciurus vulgaris) förändras i vikt under övervintring mäts 7 slumpmässigt valda ekorrar före och 5 slumpmässigt valda ekorrar efter övervintring. Datan finns tillgänglig i excelfilen *Uppgiftsdata.xlsx* på canvassidan, i fliken *Ekorrar*. Ladda ner filen och fyll i stycket nedan för att importera datan.

```
dat_sq <- read_excel("___", sheet = "Ekorrar")
dat_sq

# dat_sq <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2023/main/Data/
```

Uppgift 6.9 (Ekorrgraf)**.** Fyll i följande stycke för en lämplig graf för att jämföra mätningarna före och mätningarna efter.

```
ggplot(dat_sq, aes(x = ___, y = ___)) +
  ___()
```

Finns det någon synlig viktskillnad?

Uppgift 6.10 (Ekorrtest)**.** Genomför ett t-test för två oberoende stickprov på ekorrdatan genom att fylla i kodstycket nedan. Formulera tydliga hypoteser och dra en klar slutsats.

```
t.test(___ ~ ___, data = dat_sq, var.equal = ___)
```

Uppgift 6.11 (Ekorrdesign)**.** Ett problem med att mäta skilda individer före och efter övervintring är att det kan finnas en stor skillnad i vikt mellan individuella ekorrar. Kan man lägga upp försöket på ett sätt som reducerar det problemet?

6.3 z-test och konfidensintervall för två proportioner

Om man vill jämföra två proportioner kan man använda z-testet för två stickprov. Säg till exempel att man har två sorter av någon planta och vill se hur stor proportion som är infekterad av bladmögel. I den ena gruppen (sort A) är 17 av 50 infektera och i den andra (sort B) är 26 av 60 infekterade. Testets hypoteser är i det tvåsidiga fallet

- H0: proportion A är lika med proportion B
- H1: proportion A är skild från proportion B

I R kan testet genomföras med `prop.test`-funktionen. Funktionens första argument är antalen infekterade, som en vektor med två värden, och dess andra argument är totalerna. Likt testet med ett stickprov finns en möjlighet att göra en kontinuitetskorrektion med `correct`-argumentet. För att få samma resultat som räkning för hand anger vi att korrektion inte ska göras med `correct = F`.

```
prop.test(c(17, 26), c(50, 60), correct = F)
```

```
##
##  2-sample test for equality of proportions without continuity correction
##
## data:  c(17, 26) out of c(50, 60)
## X-squared = 0.9978, df = 1, p-value = 0.3178
## alternative hypothesis: two.sided
## 95 percent confidence interval:
##  -0.27488771  0.08822105
## sample estimates:
##    prop 1    prop 2
## 0.3400000 0.4333333
```

Notera att funktionen inte ger ett z-värde utan ett χ^2-värde (utskrivet `X-squared`). Det beror på att funktionen beräknar z-testet som ett likvärdigt χ^2-test. Det z-värde man får om man genomför testet som ett z-test är detsamma som roten ur utskriftens χ^2-värde. Testet ger ett högt p-värde på 0.32 vilket innebär att nollhypotesen inte förkastas: det finns ingen signifikant skillnad i infektionsproportion.

Funktionen `prop.test()` ger också en utskrift av konfidensintervallet. Tolkning är att skillnaden i proportioner mellan populationerna ligger i intervallet med 95 procents konfidens. Notera att nollan ingår i intervallet.

Uppgift 6.12 (Lämplig approximation?)**.** Z-test bygger på en normalapproximation. Som tumregel för när approximationen är rimlig används ofta att n * p * (1 - p) ska vara större än 10 för bägge stickproven. Gör beräkningen för datan i exemplet (17 av 50 respektive 26 av 60).

Uppgift 6.13 (Burfågel)**.** Det finns en förvånansvärt stor mängd studier på kopplingen mellan innehav av burfågel och lungcancer. En sådan studie (Kohlmeier et al 1992) ger följande antal för burfågelägande och lungcancer.

```
dat_bird <- data.frame(Burfågel = c("Burfågel", "Ej_burfågel"),
                       Lungcancer = c(98, 141),
                       Ej_lungcancer = c(101, 328))
dat_bird
```

```
##      Burfågel Lungcancer Ej_lungcancer
## 1    Burfågel         98           101
## 2 Ej_burfågel        141           328
```

Datan tyder på att människor med burfågel har en förhöjd risk att drabbas av lungcancer. Genomför ett z-test för att se om andelen burfågelägare än densamma i de två patientgrupperna.

```
prop.test(x = c(___, ___), n = c(___, ___), correct = F)
```

Genomför ett z-test för att se om andelen cancerdrabbade är densamma i de två burfågelsgrupperna. Hur förhåller sig p-värdena i de bägge testerna till varandra?

```
prop.test(x = c(___, ___), n = c(___, ___), correct = F)
```

Finns det någon industri som kan ha ett intresse av att finansiera forskning som söker alternativa riskfaktorer för lungcancer?

6.4 Chi-två-test för korstabeller

Data med två kategoriska variabler kan presenteras med en korstabell. Ta som (ett något deppigt) exempel överlevnadsdata från Titanic. Datan finns

tillgänglig i R som `Titanic`. I detta fall ges överlevnad filtrerad på vuxna män, uppdelat efter klass.

```
dat_titanic <- Titanic %>% data.frame() %>% filter(Sex == "Male", Age == "Adult")
dat_titanic
```

```
##   Class  Sex   Age Survived Freq
## 1   1st Male Adult       No  118
## 2   2nd Male Adult       No  154
## 3   3rd Male Adult       No  387
## 4  Crew Male Adult       No  670
## 5   1st Male Adult      Yes   57
## 6   2nd Male Adult      Yes   14
## 7   3rd Male Adult      Yes   75
## 8  Crew Male Adult      Yes  192
```

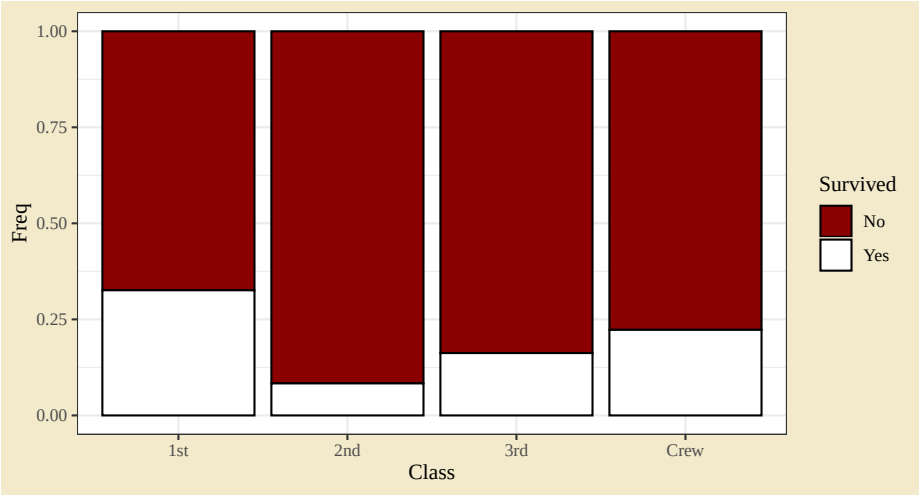
En korstabell kan konstrueras med `pivot_wider`.

```
dat_wide <- dat_titanic %>%
  pivot_wider(names_from = Survived, values_from = Freq)
dat_wide
```

```
## # A tibble: 4 x 5
##   Class Sex   Age      No   Yes
##   <fct> <fct> <fct> <dbl> <dbl>
## 1 1st   Male  Adult   118    57
## 2 2nd   Male  Adult   154    14
## 3 3rd   Male  Adult   387    75
## 4 Crew  Male  Adult   670   192
```

Datan tyder på att överlevnad är beroende av klass. Datan kan illustreras med uppdelade staplar

```
ggplot(dat_titanic, aes(Class, Freq, fill = Survived)) +
  geom_col(position = position_fill(), color = "black") +
  scale_fill_manual(values = c("red4", "white"))
```



Argumentet `position` i `geom_bar` används för att skapa proportionella staplar.

Ett chi-två-test på en korstabell har nollhypotesen att det inte finns något samband mellan variabeln för rader och variabeln för kolumner. Antal frihetsgrader ges av antal rader minus ett gånger antal kolumner minus ett. Testet kan enkelt göras med `chisq.test()`. Som ingångsvärde kan man plocka ut kolumnerna med numeriska värden genom hakparenteser.

```
dat_wide[, 4:5] # De två numeriska kolumnerna
```

```
## # A tibble: 4 x 2
##      No   Yes
##   <dbl> <dbl>
## 1   118    57
## 2   154    14
## 3   387    75
## 4   670   192
```

```
chisq.test(dat_wide[, 4:5])
```

```
##
##  Pearson's Chi-squared test
##
## data:  dat_wide[, 4:5]
## X-squared = 37.988, df = 3, p-value = 2.843e-08
```

Utskriften ger teststorheten, antal frihetsgrader, och p-värdet. I det här fallet är p-värdet mycket litet och slutsatsen blir att nollhypotesen förkastas - det finns ett samband mellan klass och överlevnad. Antalet frihetsgrader ges av antalet rader minus ett gånger antalet kolumner minus ett (här (4-1) * (2-1) = 3).

Chi-två-testet är ett asymptotiskt test - dess egenskaper är beroende av *stora*

stickprov. Som gräns för storleken används ofta att samtliga förväntade antal ska vara större än 5. Funktionen ger en varning om förväntade värden är små. En möjlig lösning i sådana fall är att slå ihop klasser.

```
test_result <- chisq.test(dat_wide[, 4:5])
test_result$expected # Samtliga förväntade värden över 5
```

```
##            No       Yes
## [1,] 139.5171  35.48290
## [2,] 133.9364  34.06359
## [3,] 368.3251  93.67487
## [4,] 687.2214 174.77864
```

Om detta krav inte är uppfyllt skriver funktionen ut en varning.

Uppgift 6.14 (Ogiltig approximation)**.** Ta följande lilla korstabell och kör `chisq.test()` för att få ett felmeddelande.

```
dat <- matrix(c(4,2,5,1), 2)
dat
```

```
##      [,1] [,2]
## [1,]    4    5
## [2,]    2    1
```

Uppgift 6.15 (Burfågeln återvänder)**.** En svensk studie på koppling mellan burfågel och lungcancer (Modigh et al, 1996) ger följande antal (för män).

```
dat_bird_swe <- data.frame(Burfågel = c("Burfågel", "Ej_burfågel"),
              Lungcancer = c(108, 144),
              Ej_lungcancer = c(171, 256))
dat_bird_swe
```

```
##      Burfågel Lungcancer Ej_lungcancer
## 1    Burfågel        108           171
## 2 Ej_burfågel        144           256
```

Genomför ett chi-två-test för att se om andelen cancerdrabbade än densamma i de två burfågelsgrupperna. Formulera tydliga hypoteser. För att få utfall som stämmer med en handräkning kan man sätta `correct = F`.

```
dat_bird_swe[, c(2,3)]
chisq.test(___, correct = F)
```

Chi-två-testet kan tillämpas på korstabeller med godtyckligt antal rader och kolumner.

Uppgift 6.16 (Po-ta-toes-import)**.** I en undersökning på potatis används fyra behandlingar (a1b1, a1b2, a2b1 och a2b2). 125 potatisar från varje behandling sorteras in i fyra olika färggrupper (A, B, C och D). Datan finns i fliken *Po-ta-toes* i excelfilen *Uppgiftsdata.xlsx* på canvassidan. Ladda ned filen och läs in

datan genom att fylla i stycket nedan.

```
dat_pot <- read_excel("___", sheet = "Po-ta-toes")
dat_pot

# dat_pot <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2023/main/Data/
```

Uppgift 6.17 (Po-ta-toes-graf)**.** För att göra en graf kan man pivotera datan till lång form.

```
dat_long <- dat_pot %>% pivot_longer(-Färg, values_to = "Antal", names_to = "Behandling
dat_long
```

Skapa ett stapeldiagram med uppdelade staplar genom att fylla i kodstycket nedan. Behandling ska vara på x-axeln och ifylld färg ska ges av `Färg`.

```
ggplot(dat_long, aes(x = ___, y = ___, fill = ___)) +
  geom_col(col = "black", width = 0.6) +
  scale_fill_brewer(palette = "Reds")
```

Finns det några synbara skillnader mellan behandlingar?

Uppgift 6.18 (Po-ta-toes-test)**.** Beräkna ett chi-två-test på potatisdatan för att se om det finns färgskillnader mellan behandlingarna. Formulera tydliga hypoteser och ge ett tydligt svar.

```
dat_pot[,-1]
chisq.test(___)
```

Uppgift 6.19 (Hemmasegrar över årtionden)**.** Vi vill undersöka om andelen hemmasegrar i herrallsvenskan förändrats över tid. Vi importerar data över matchresultat sedan 1920-talet.

```
dat_alls <- read_csv("https://raw.githubusercontent.com/adamflr/ST0060-2023/main/Data/
dat_alls
```

```
## # A tibble: 15,236 x 9
##    Hemmalag     Bortalag Hemmamål Bortamål Publik Domare Arena Datum      Säsong
##    <chr>        <chr>       <dbl>    <dbl>  <dbl> <chr>  <chr> <date>     <chr>
##  1 Örgryte      Hammarby        5        1   4000 Carl ~ <NA>  1924-08-03 1924_~
##  2 IFK Norrköp~ Landskr~        0        1   1500 Ivar ~ <NA>  1924-08-03 1924_~
##  3 IFK Malmö    IFK Göt~        1        1   3276 Johan~ <NA>  1924-08-03 1924_~
##  4 Helsingborg  Gais            1        2   3000 Carl ~ <NA>  1924-08-03 1924_~
##  5 Eskilstuna   Sleipner        1        3    700 Oscar~ <NA>  1924-08-03 1924_~
##  6 AIK          Västerå~        5        1   2000 Arne ~ <NA>  1924-08-03 1924_~
##  7 IFK Göteborg Gais            1        3   3600 Sigfr~ <NA>  1924-08-08 1924_~
##  8 Västerås IK  Eskilst~        0        1    600 Berti~ <NA>  1924-08-10 1924_~
##  9 Sleipner     Örgryte         0        1   2280 Ernfr~ <NA>  1924-08-10 1924_~
## 10 Landskrona   IFK Göt~        0        4   1000 Gusta~ <NA>  1924-08-10 1924_~
## # i 15,226 more rows
```

Följande kod skapar en variabel för årtionde, en variabel för hemmaseger, och räknar ut antalen hemmasegrar per årtionde. Detaljer är oviktiga här.

```
library(lubridate)
dat_hemma <- dat_alls %>%
  mutate(År = year(Datum),
         Årtionde = floor(År / 10) * 10,
         Hemmaseger = ifelse(Hemmamål > Bortamål, "Hemmaseger", "Ej_hemmaseger")) %>%
  count(Årtionde, Hemmaseger) %>%
  pivot_wider(values_from = n, names_from = Hemmaseger) %>%
  mutate(Total = Hemmaseger + Ej_hemmaseger,
         Proportion = Hemmaseger / (Hemmaseger + Ej_hemmaseger))
```

Fyll i koden nedan för att skapa en tidsserie (en linjegraf med tid på x-axeln) för andelen `Proportion`.

```
ggplot(dat_hemma, aes(x = ___, y = ___)) +
  ___()
```

Uppgift 6.20 (1920-talet mot 1960-talet)**.** Använd ett z-test för att se om proportionen hemmasegrar under 1920-talet (371 av 738) är skild från 1960-talet (590 av 1320).

```
prop.test(c(___, ___), n = c(___, ___), correct = F)
```

6.5 Bonus. Bilder i R

Det finns en stor mängd paket som kan hantera bilder. Låt oss ta en titt på ett av dem - `magick` - vilket bygger på en koppling till ImageMagick (https://imagemagick.org/).

```
# install.packages("magick")
library(magick)
```

Med funktionen `image_read()` kan man läsa in en bild, antingen från lokal hårddisk eller från en internetaddress. Här hämtar vi en bild av Nils Dardels *Den döende dandyn* (1918) från Wikipedia.

```
url <- "https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/Nils_Dardel_D%C3%B6ende_dandyn.
img <- image_read(url)
img
```

Uppgift 6.21 (Någon annan bild)**.** Hitta någon annan bild online, vad som helst. Gör lämplig ändring i stycken ovan för att läsa in bilden med `image_read()`.

Låt oss börja med att ändra storleken med `image_resize()`. Följande ger en bild där den kortaste av höjd och bredd är 500 pixlar.

```
img <- img %>%
  image_resize("500")
img
```

Uppgift 6.22 (Storlek)**.** Vad kan vara koden för att sätta en bild till halva storleken, alltså 50% av den ursprungliga bilden?

Man kan också manipulera egenskaper som kontrast, mättnad och färgton.

```
img %>%
  image_modulate(saturation = 50) %>%
  image_modulate(hue = 50)
```

För mer information av tillgängliga funktioner, titta på paketets hjälpsida med `?magick` och introduktionen på https://docs.ropensci.org/magick/articles/intro.html.

En enkel vetenskaplig tillämpning av bildanalys kan baseras på de relativa andelarna av olika färger. Det kan till exempel användas för att beräkna skadegrad på löv (som efter färgning kan ha specifika färger för skadade delar) eller storlek på trädkronor. Funktionen `image_quantize()` kan minska antalet färger i en bild till ett mer hanterbart antal.

```
img %>% image_quantize(max = 10)
```

Uppgift 6.23 (Antal färger)**.** Med 50 enskilda färger blir *Den döende dandyn* något mattare, men karaktärernas klädsel har klara färger. Hur få måste det totala antalet färger bli innan *du* ser en klar försämring av bilden?

Funktionen `image_data()` kan användas för att ta ut färgvärdet för varje pixel. Därefter kan man enkelt beräkna andelen för olika färger. Följande stycke förenklar bilden till tio färger, extraherar datan och beräknar antalet pixlar med respektive färg. Den exakta koden är inte så viktig här och kan läsas kursivt.

```
img <- img %>% image_quantize(max = 10)
info <- img %>% image_info()
pixel_values <- img %>% image_data() %>% as.vector()

dat_pix <- expand_grid(y = info$height:1, x = 1:info$width, color = c("R", "G", "B")) %
  mutate(value = pixel_values) %>%
  pivot_wider(values_from = value, names_from = color) %>%
  mutate(hex = paste0("#", R, G, B))

dat_pix
```

Den konstruerade datan innehåller koordinater med x och y samt färgvärden i tre färgband och en hexkod som anger färgen. Härifrån kan vi göra en grafversion av bilden med `geom_raster()`.

```
ggplot(dat_pix, aes(x, y)) +
  geom_raster(fill = dat_pix$hex)
```

Notera att vi sätter `fill` i geom-funktionen, eftersom målet är att sätta färgen till den som anges i kolumnen hex.

Uppgift 6.24 (Färg som aesthetic)**.** Vad händer om man sätter `fill = hex` inom `aes()`-funktionen istället?

```
ggplot(dat_pix, aes(x, y, fill = ___)) +
  geom_raster()
```

Funktionen `scale_fill_manual()` kan styra färgvalet i det fallet.

```
ggplot(dat_pix, aes(x, y, fill = ___)) +
  geom_raster() +
  scale_fill_manual(values = c('white', 'aliceblue',
                               'antiquewhite', 'antiquewhite1',
                               'antiquewhite2', 'antiquewhite3',
                               'antiquewhite4', 'aquamarine',
                               'aquamarine1', 'aquamarine2')) +
  theme_void()
```

Tillgängliga färger kan tas fram med `colors()`.

Slutligen kan vi nu göra en enkel bildanalys genom att räkna antal eller andel pixlar med en viss färg.

```
dat_pix_count <- dat_pix %>%
  count(hex) %>%
  mutate(hex = reorder(hex, n))

ggplot(dat_pix_count, (aes(n, hex))) +
  geom_col(fill = dat_pix_count$hex)
```

Uppgift 6.25 (Avslutande proportionstest)**.** Låt oss ta ett mindre stickprov från bilden. Funktionen `set.seed()` sätter ett startvärde för slumtalsgeneratorn, vilket är bra om man vill reproducera ett visst utfall.

```
set.seed(1573)
dat_sample <- dat_pix %>% slice_sample(n = 100)
dat_sample %>% count(hex)

ggplot(dat_sample, aes(x, y)) +
  geom_point(color = dat_sample$hex, size = 8)
```

I stickprovet är 62 av 100 pixlar en mörkblå färg. Genomför ett test med `prop.test()` för att se om andelen i populationen (som i detta fall är hela tavlan) är skild från 0.7. Jämför med proportionen i den större datamängden

`dat_pix`.

Chapter 7

Variansanalys

Datorövning 7 handlar om variansanalys. Efter övningen ska vi kunna

- beräkna en anova-modell i R,
- ta fram och tolka en anova-tabell,
- göra lämpliga tester av modellantaganden,
- göra parvisa jämförelser mellan behandlingar.

7.1 Repetition av datorövning 6

När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 6 tittade vi på tester för två stickprov. För normalfördelad data kan man då använda ett t-test för två stickprov (för två matchade stickprov eller för två oberoende stickprov beroende på situation) för data med utfall i två eller flera kategorier kan man använda ett z-test för två stickprov eller ett chi-två-test för en korstabell.

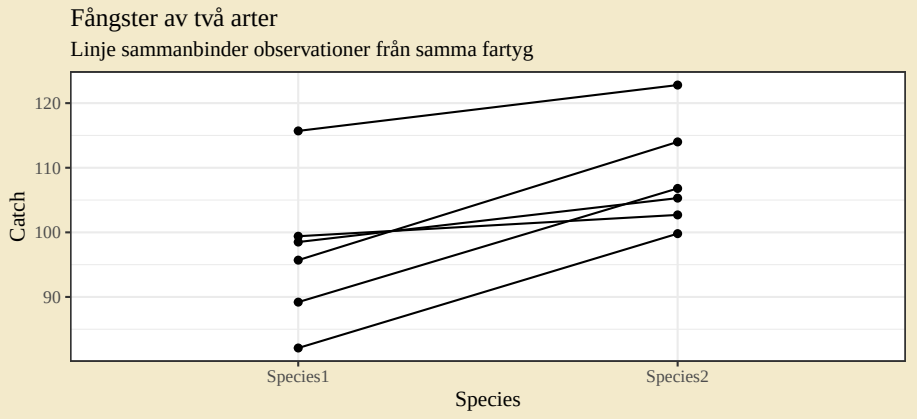
Ett t-test för matchade stickprov används när de två grupper man jämför är matchade så att en observation i den ena gruppen är kopplad till en observation i den andra gruppen. Ett t-test för oberoende stickprov används om man inte har matchade stickprov, det vill säga då det inte finns någon koppling mellan behandlinggrupperna.

Ta som exempel följande fiskefångster för sex båtar från två regioner och två fiskearter.

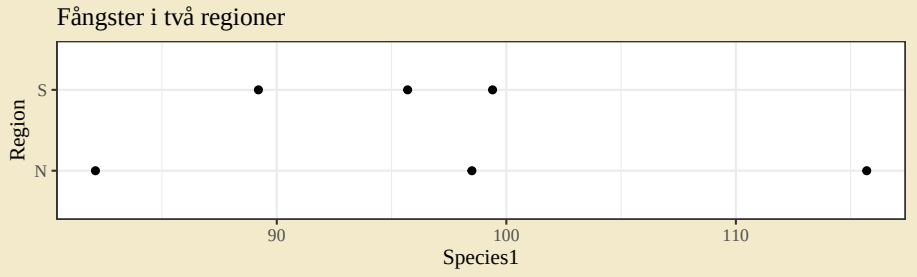
```
dat_fish <- data.frame(Vessel = c("A", "B", "C", "D", "E", "F"),
                       Region = c("N", "N", "N", "S", "S", "S"),
                       Species1 = c(115.7, 98.5, 82.1, 89.2, 95.7, 99.4),
                       Species2 = c(122.8, 105.3, 99.8, 106.8, 114, 102.7))
```

Vi vill här testa om det finns en skillnad mellan arter och om det finns skillnad mellan regioner.

```
dat_long <- dat_fish %>%
  pivot_longer(-c(Vessel, Region), names_to = "Species", values_to = "Catch")
ggplot(dat_long, aes(Species, Catch, group = Vessel)) +
  geom_point() +
  geom_line() +
  labs(title = "Fångster av två arter", subtitle = "Linje sammanbinder observationer fr
```




```
ggplot(dat_fish, aes(Species1, Region)) +
  geom_point() +
  labs(title = "Fångster i två regioner")
```



För arterna har vi matchad data - varje observation av den ena arten är kopplad till en observation från den andra arten eftersom den kommer från samma båt

- och vi kan testa om medelfångsterna av de två arterna är lika med ett t-test. Hypoteserna ges av

- H0: populationsmedelvärdet av fångster för art 1 är lika med det för art 2,
- H1: populationsmedelvärdena är ej lika.

I R kan ett test för matchad data genomföras med `t.test()` och argumentet `paired`, eller genom att beräkna differensen per båt och göra ett t-test för ett stickprov.

```
t.test(dat_fish$Species1, dat_fish$Species2, paired = T)
```

```
##
##  Paired t-test
##
## data:  dat_fish$Species1 and dat_fish$Species2
## t = -4.2613, df = 5, p-value = 0.008005
## alternative hypothesis: true mean difference is not equal to 0
## 95 percent confidence interval:
##  -18.918238  -4.681762
## sample estimates:
## mean difference
##           -11.8
```

```
# Alternativt
# t.test(dat_fish$Species1 - dat_fish$Species2)
```

Det beräknade p-värdet ställs mot en signifikansnivå, vanligen fem procent, och om p-värdet är under signifikansnivån förkastar vi nollhypotesen. I det här exemplet tyder på värdet på att nollhypotesen inte stämmer - en art är vanligare än den andra.

För att jämföra regioner kan vi göra ett t-test för två oberoende stickprov. Hypoteser ges av

- H0: populationsmedelvärdet av fångster är lika mellan regioner,
- H1: populationsmedelvärdet av fångster är ej lika mellan regioner.

Testet kan genomföras med `t.test()` och kan antingen göras med ett antagande om lika varianser (vilket motsvarar det som görs för hand under kursen) eller utan det antagandet. Variabler kan anges med en formel som `Species1 ~ Group`, vilket vi kan tänka på som värden för art 1 uppdelat efter grupp.

```
t.test(Species1 ~ Region, data = dat_fish, var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  Species1 by Region
```

```
## t = 0.39416, df = 4, p-value = 0.7136
## alternative hypothesis: true difference in means between group N and group S is not
## 95 percent confidence interval:
##  -24.17586  32.17586
## sample estimates:
## mean in group N mean in group S
##        98.76667        94.76667
```

Ett högt p-värde tyder på att det inte finns någon skillnad i fångst mellan regioner.

För data där utfallen är två eller flera kategorier kan ett chi-två-test testa om det finns något samband mellan två variabler. Följande data anger vilka partier ett urval väljare från tre kommuner planerar rösta på i nästa riksdagsval.

```
dat_parti <- data.frame(Kommun = c("Malmö", "Lund", "Kävlinge"),
                        S = c(54, 102, 40),
                        M = c(30, 98, 53),
                        MP = c(7, 50, 5))
dat_parti
```

```
##     Kommun   S  M MP
## 1    Malmö  54 30  7
## 2     Lund 102 98 50
## 3 Kävlinge  40 53  5
```

För att testa om det finns något samband mellan kommun och parti sätter vi upp hypoteserna

- H0: det finns inget samband mellan parti och kommun (ingen skillnad mellan kommuner),
- H1: det finns något samband mellan parti och kommun.

Detta kan testas med ett chi-två-test med funktionen `chisq.test()`. Som argument ges den numeriska delen av korstabellen - vi tar alltså bort den första kolumnen för kommun.

```
chisq.test(dat_parti[, -1])
```

```
##
##  Pearson's Chi-squared test
##
## data:  dat_parti[, -1]
## X-squared = 25.659, df = 4, p-value = 3.706e-05
```

Det låga p-värdet på 0.000037 ger att vi förkastar nollhypotesen och drar slutsatsen att det finns ett samband mellan kommun och parti.

7.2 Allmänt

Variansanalys (eller *anova-modellen*) är en statistisk modell där medelvärdet varierar beroende på en behandling och ett normalfördelat slumpfel. Från en anova-modell kan man beräkna ett F-test, som testar om det finns någon övergripande gruppskillnad, och post-hoc-test, som jämför specifika grupper med varandra.

Den specifika modellen beror på försöksupplägget. Här ges exempel på variansanalys med en faktor, en faktor med block, och två faktorer.

7.3 Variansanalys. En faktor

Vid variansanalys med en faktor har man observationer av en kontinuerlig utfallsvariabel från två eller flera behandlingsgrupper. Som exempel används en datamängd på ett odlingsförsök med tre behandlingar (varav en kontroll). Exemplet finns tillgängligt i R som `PlantGrowth`.

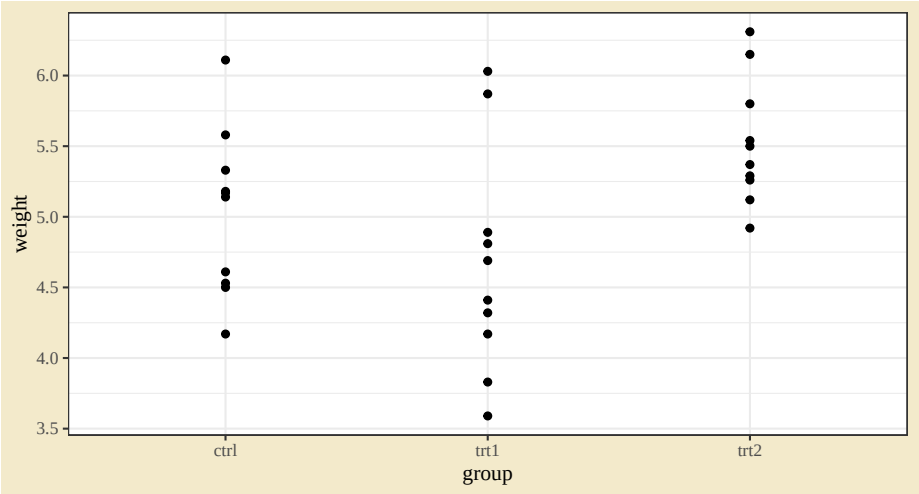
```
PlantGrowth
```

```
##    weight group
## 1    4.17  ctrl
## 2    5.58  ctrl
## 3    5.18  ctrl
## 4    6.11  ctrl
## 5    4.50  ctrl
## 6    4.61  ctrl
## 7    5.17  ctrl
## 8    4.53  ctrl
## 9    5.33  ctrl
## 10   5.14  ctrl
## 11   4.81  trt1
## 12   4.17  trt1
## 13   4.41  trt1
## 14   3.59  trt1
## 15   5.87  trt1
## 16   3.83  trt1
## 17   6.03  trt1
## 18   4.89  trt1
## 19   4.32  trt1
## 20   4.69  trt1
## 21   6.31  trt2
## 22   5.12  trt2
## 23   5.54  trt2
## 24   5.50  trt2
## 25   5.37  trt2
## 26   5.29  trt2
```

```
## 27   4.92  trt2
## 28   6.15  trt2
## 29   5.80  trt2
## 30   5.26  trt2
```

Datan har 30 observationer av vikt `weight` och varje observation tillhör någon specifik behandling `group`. Datan kan illustreras med ett spridningsdiagram.

```
ggplot(PlantGrowth, aes(group, weight)) +
  geom_point()
```



Behandling 1 verkar vara något lägre än kontrollen medan behandling 2 verkar vara något högre.

En anova-modell kan i R skattas med funktionen `lm()` (för *linjär modell*). Från modellobjektet kan man sedan plocka fram en anova-tabell (som bland annat anger utfallet av F-testet) och genomföra parvisa jämförelser genom `emmeans`.

```
mod <- lm(weight ~ group, data = PlantGrowth)
```

Modellen anges som en formel `weight ~ group`, vilket kan utläsas *vikt beroende på behandlingsgrupp*. Därefter anges data med argumentet `data`.

För anova-tabellen finns flera alternativ. Här används funktionen `Anova()` från paketet `car`.

```
library(car)
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: weight
##             Sum Sq Df F value  Pr(>F)
```

```
## group      3.7663  2  4.8461 0.01591 *
## Residuals 10.4921 27
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Anova-tabellen ger kvadratsummor (`Sum Sq`), frihetsgrader (`Df`) och utfallet av ett F-test. Testets hypoteser ges av

H0: alla behandlingsgrupper har samma medelvärde H1: alla behandlingsgrupper har inte samma medelvärde

Det låga p-värdet tyder på att nollhypotesen bör förkastas, vilket alltså pekar på att det finns någon eller några skillnader i medelvärde.

Uppgift 7.1 (Anova för hand)**.** Anovatabell från `Anova()` ger kvadratsummor och frihetsgrader. Använd den informationen för att, för hand, beräkna medelkvadratsummor och F-värdet.

Uppgift 7.2 (Tabellvärde för F-fördelningen)**.** Anova-tabellen ger ett p-värde från vilket vi kan dra en direkt slutsats. Om man istället löser uppgiften för hand ställer man det beräknade F-värdet mot ett kritiskt värde från en tabell över F-fördelningen. Se efter om man kan hitta ett lämpligt tabellvärde för det aktuella testet (med 2 och 27 frihetsgrader). Det är möjligt att det inte finns en rad för 27 i en vanlig F-fördelningstabell, använd isåfall värdet på närmast övre rad (t.ex. 26 eller 25). I R kan kvantiler för F-fördelningen tas fram med `qf()`, t.ex.

```
qf(0.95, 2, 27)
```

```
## [1] 3.354131
```

En naturlig följdfråga är vilka behandlingsgrupper som skiljer sig åt. För att besvara det krävs *parvisa jämförelser* där behandlingarna jämförs två och två. Parvisa jämförelse kan göras med paketet `emmeans` och funktionen med samma namn. Funktionen tar modellobjektet som första argument och en formel för jämförelsetyp som andra argument (här `pairwise ~ group`, en parvis jämförelse mellan nivåer i `group`).

```
# install.packages("emmeans")
library(emmeans)
emmeans(mod, pairwise ~ group)
```

```
## $emmeans
##  group emmean    SE df lower.CL upper.CL
##  ctrl    5.03 0.197 27     4.63     5.44
##  trt1    4.66 0.197 27     4.26     5.07
##  trt2    5.53 0.197 27     5.12     5.93
##
## Confidence level used: 0.95
##
```

```
## $contrasts
##  contrast    estimate    SE df t.ratio p.value
##  ctrl - trt1    0.371 0.279 27   1.331  0.3909
##  ctrl - trt2   -0.494 0.279 27  -1.772  0.1980
##  trt1 - trt2   -0.865 0.279 27  -3.103  0.0120
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

I den nedre tabellen med jämförelser ges alla parvisa jämförelser. Nollhypotesen är att de två grupper som jämförs har samma medelvärde - ett lågt p-värde tyder alltså på att de två grupperna är signifikant skilda. Notera också att p-värden justeras med tukey-metoden, även känt som Tukeys HSD.

Om man istället vill använda Fishers LSD kan man styra justeringen med argumentet `adjust`.

```
emmeans(mod, pairwise ~ group, adjust = "none")
```

```
## $emmeans
##  group emmean    SE df lower.CL upper.CL
##  ctrl    5.03 0.197 27     4.63     5.44
##  trt1    4.66 0.197 27     4.26     5.07
##  trt2    5.53 0.197 27     5.12     5.93
##
## Confidence level used: 0.95
##
## $contrasts
##  contrast    estimate    SE df t.ratio p.value
##  ctrl - trt1    0.371 0.279 27   1.331  0.1944
##  ctrl - trt2   -0.494 0.279 27  -1.772  0.0877
##  trt1 - trt2   -0.865 0.279 27  -3.103  0.0045
```

Parvisa jämförelser presenteras ofta med signifikansbokstäver (en. *compact letter display, cld*). Dessa kan plockas fram med `multcomp`-paketet och funktionen `cld()`.

```
em <- emmeans(mod, pairwise ~ group)

library(multcomp)
cld(em, Letters = letters)
```

```
##  group emmean    SE df lower.CL upper.CL .group
##  trt1    4.66 0.197 27     4.26     5.07  a
##  ctrl    5.03 0.197 27     4.63     5.44  ab
##  trt2    5.53 0.197 27     5.12     5.93   b
##
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 3 estimates
## significance level used: alpha = 0.05
```

```
## NOTE: If two or more means share the same grouping symbol,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

Tolkning av grupperingen till höger är att grupper som delar en bokstav inte är signifikant skilda. I det här fallet är den lägsta nivån skild från de två högsta. I övrigt finns inga signifikanta skillnader. Jämför gärna med p-värdena från tabellen med parvisa jämförelser. Man bör se att parvisa jämförelser med ett p-värde under fem procent motsvaras av att de behandlingarna inte delar någon bokstav i bokstavstabellen.

Uppgift 7.3 (Anova med två behandlingar)**.** Följande kod skapar en datamängd med två behandlingar.

```
dat_two <- PlantGrowth %>% filter(group %in% c("trt1", "trt2"))
```

Använd den datan för att göra ett t-test för två oberoende stickprov med lika varians, ett t-test för två oberoende stickprov utan antagande om lika varians, och ett F-test (ofullständig exempelkod nedan). Vad kan sägas om p-värdena från de tre testen?

```
t.test(___ ~ group, data = dat_two, var.equal = T)
t.test(weight ~ ___, data = dat_two, var.equal = F)

mod <- lm(weight ~ group, data = ___)
Anova(mod)
```

Uppgift 7.4 (Mass-signifikans)**.** Anledning till att vi justerar p-värden är att man vid varje test har en sannolikhet att förkasta. Om man gör ett stort antal tester är man nästan garanterad att få något (falskt) signifikant resultat. Justering höjer p-värdena för att minska den risken. Följande kod simulerar data med 5 grupper och producerar de parvisa jämförelserna.

```
n_groups <- 5
dat_sim <- expand_grid(obs = 1:10, group = letters[1:n_groups]) %>% mutate(y = rnorm(n()))
mod <- lm(y ~ group, dat_sim)
emmeans(mod, pairwise ~ group, adjust = "none")
```

```
## $emmeans
##  group  emmean    SE df lower.CL upper.CL
##  a     -0.1030 0.322 45   -0.752   0.5461
##  b     -0.0894 0.322 45   -0.738   0.5597
##  c     -0.0891 0.322 45   -0.738   0.5599
##  d     -0.6432 0.322 45   -1.292   0.0058
##  e     -0.0878 0.322 45   -0.737   0.5613
##
## Confidence level used: 0.95
##
## $contrasts
```

```
##  contrast  estimate    SE df t.ratio p.value
##  a - b    -0.013606 0.456 45  -0.030  0.9763
##  a - c    -0.013869 0.456 45  -0.030  0.9759
##  a - d     0.540269 0.456 45   1.186  0.2420
##  a - e    -0.015211 0.456 45  -0.033  0.9735
##  b - c    -0.000263 0.456 45  -0.001  0.9995
##  b - d     0.553875 0.456 45   1.215  0.2306
##  b - e    -0.001606 0.456 45  -0.004  0.9972
##  c - d     0.554137 0.456 45   1.216  0.2303
##  c - e    -0.001343 0.456 45  -0.003  0.9977
##  d - e    -0.555480 0.456 45  -1.219  0.2292
```

Kör koden tio gånger. Hur många gånger av de tio ger de parvisa jämförelserna *någon* signifikant skillnad (det vill säga något p-värde under 0.05)?

En passande xkcd-serie: https://xkcd.com/882/

Uppgift 7.5 (Äppelinfektionsimport)**.** En studie har givit ett mått på infektion hos äppelträd. Fyra sorter jämförs med tre replikat per sort. Data finns i fliken *Äppelangrepp* i excelfilen *Uppgiftsdata.xslx* på canvassidan. Fyll i kodstycket nedan för att importera datan.

Uppgift 7.6 (Äppelinfektionsgraf)**.** Fyll i kodstycket nedan för att skapa en graf av äppeldatan.

```
ggplot(___, aes(x = ___, y = ___)) +
  geom_point()
```

Uppgift 7.7 (Äppelinfektionsmodell)**.** Fyll i kodstycket nedan för att skatta en anovamodell och ta fram anovatabellen. Vad är F-testets noll- och alternativhypotes? Vilken slutsats kan man dra från testet?

```
mod <- lm(___ ~ ___, data = dat_apple)
Anova(mod)
```

7.4 Variansanalys. En faktor med block

I en blockdesign delas försöksobjekten (de enheter man ger en behandling och sedan mäter, t.ex. en försöksruta eller en planta) in i grupper av lika objekt (ett *block*). Sedan ger man enheterna inom blocket varsin behandling. Blockförsök är ofta balanserade, så att varje behandling förekommer en gång i varje block.

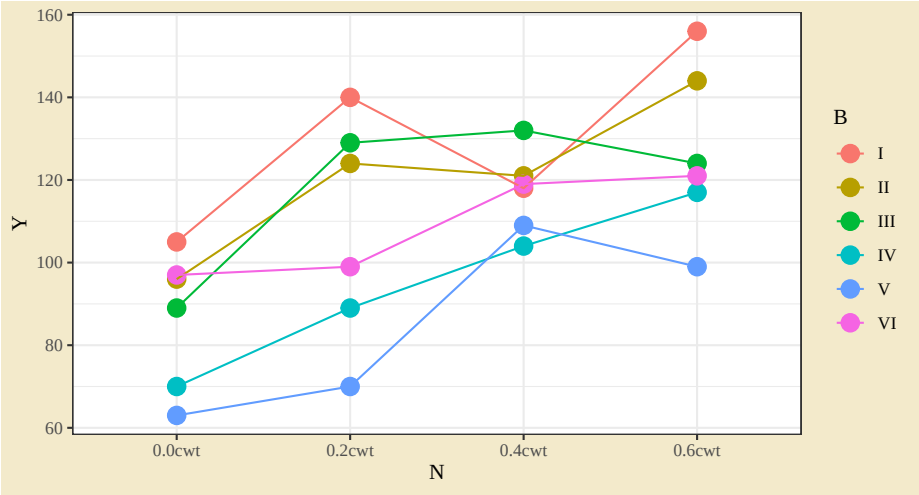
Som exempel på ett blockförsök kan vi titta på datan `oats` från paketet `MASS`. Datan kommer från ett agrikulturellt försök och blockdesignen sker genom att man delar in ett fält i flera delar (blocken) och sätter varje behandling i varje block. Datan har två faktorer (kväve `N` och sort `V`), men låt oss i den här första delen titta på en specifik sort.

```
library(MASS)
oats_marvel <- oats %>% filter(V == "Marvellous")
oats_marvel
```

```
##      B          V      N   Y
## 1    I Marvellous 0.0cwt 105
## 2    I Marvellous 0.2cwt 140
## 3    I Marvellous 0.4cwt 118
## 4    I Marvellous 0.6cwt 156
## 5   II Marvellous 0.0cwt  96
## 6   II Marvellous 0.2cwt 124
## 7   II Marvellous 0.4cwt 121
## 8   II Marvellous 0.6cwt 144
## 9  III Marvellous 0.0cwt  89
## 10 III Marvellous 0.2cwt 129
## 11 III Marvellous 0.4cwt 132
## 12 III Marvellous 0.6cwt 124
## 13  IV Marvellous 0.0cwt  70
## 14  IV Marvellous 0.2cwt  89
## 15  IV Marvellous 0.4cwt 104
## 16  IV Marvellous 0.6cwt 117
## 17   V Marvellous 0.0cwt  63
## 18   V Marvellous 0.2cwt  70
## 19   V Marvellous 0.4cwt 109
## 20   V Marvellous 0.6cwt  99
## 21  VI Marvellous 0.0cwt  97
## 22  VI Marvellous 0.2cwt  99
## 23  VI Marvellous 0.4cwt 119
## 24  VI Marvellous 0.6cwt 121
```

En vanlig illustration av ett blockförsök är ett punktdiagram kombinerat med ett linjediagram.

```
ggplot(oats_marvel, aes(N, Y, color = B, group = B)) +
  geom_point(size = 4) +
  geom_line()
```



Färg och linje sammanbinder observationer från samma block. Det finns tecken på en blockeffekt: block I är nästan alltid högst och block V är nästan alltid lägst. Det finns också en tydlig behandlingseffekt i att högre kväve ger högre skörd.

Blockeffekten kan enkelt föras in i modellen genom att lägga till variabeln `B` i `lm`-funktionen. Anova-tabellen och parvisa jämförelser kan göras på samma sätt som tidigare. Resultaten påverkas av att modellen har en blockfaktor; man behöver vanligen inte ange det explicit.

```
mod_bl <- lm(Y ~ N + B, data = oats_marvel)
Anova(mod_bl)
```

```
## Anova Table (Type II tests)
##
## Response: Y
##           Sum Sq Df F value    Pr(>F)
## N         5287.5  3 14.6241 0.0001004 ***
## B         5708.7  5  9.4735 0.0003106 ***
## Residuals 1807.8 15
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

P-värdet från F-testet på variabeln N är nu klart mindre än tidigare. Detta beror på att en stor del av variationen kan förklaras med blockeffekten, vilket är tydligt i att blockeffekten också har ett litet p-värde i F-testet.

Det kan vara intressant att jämföra med modellen utan block.

```
mod_wo_block <- lm(Y ~ N, data = oats_marvel)
Anova(mod_wo_block)
```

```
## Anova Table (Type II tests)
##
## Response: Y
##           Sum Sq Df F value  Pr(>F)
## N         5287.5  3  4.6896 0.01227 *
## Residuals 7516.5 20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Det som är residualens kvadratsumma i modellen utan block är i blockmodellen uppdelat i en blockeffekt och en residualterm. Eftersom F-testet bygger på en jämförelse mellan behandlingseffekten och residualtermen leder blockdesignen till starkare signifikans i blockmodellen. Å andra sidan kostar blockfaktorn frihetsgrader vilket ger oss ett svagare test. Effekten av att ta med ett block beror alltså på om det finns en verklig skillnad mellan blocken eller ej.

Vi kan gå vidare med att titta på parvisa jämförelser mellan kvävenivåer. Funktionen `emmeans()` och `cld()` fungerar som tidigare.

```
cld(emmeans(mod_bl, ~ N), Letters = letters)
```

```
##  N      emmean   SE df lower.CL upper.CL .group
##  0.0cwt   86.7 4.48 15     77.1     96.2  a
##  0.2cwt  108.5 4.48 15     98.9    118.1   b
##  0.4cwt  117.2 4.48 15    107.6    126.7   bc
##  0.6cwt  126.8 4.48 15    117.3    136.4    c
##
## Results are averaged over the levels of: B
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 4 estimates
## significance level used: alpha = 0.05
## NOTE: If two or more means share the same grouping symbol,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

Signifikansbokstäver anger att den lägsta nivån är skild från övriga och att den näst lägsta är skild från den högsta. Även här kan det vara intressant att jämföra med modellen utan block.

```
cld(emmeans(mod_wo_block, ~ N), Letters = letters)
```

```
##  N      emmean   SE df lower.CL upper.CL .group
##  0.0cwt   86.7 7.91 20     70.2      103  a
##  0.2cwt  108.5 7.91 20     92.0      125  ab
##  0.4cwt  117.2 7.91 20    100.7      134  ab
##  0.6cwt  126.8 7.91 20    110.3      143   b
##
## Confidence level used: 0.95
## P value adjustment: tukey method for comparing a family of 4 estimates
```

```
## significance level used: alpha = 0.05
## NOTE: If two or more means share the same grouping symbol,
##       then we cannot show them to be different.
##       But we also did not show them to be the same.
```

Modellen utan block ger samma medelvärden `emmean` men större medelfel `SE` och färre signifikanta skillnader.

Uppgift 7.8 (Block med två behandlingar. Graf)**.** Det minsta möjliga blocket är det med två behandlingar. Vi filtrerar havredatan för att den situationen.

```
dat_small_block <- oats %>% filter(V == "Marvellous", N %in% c("0.6cwt", "0.0cwt"))
dat_small_block
```

```
##      B          V      N   Y
## 1    I Marvellous 0.0cwt 105
## 2    I Marvellous 0.6cwt 156
## 3   II Marvellous 0.0cwt  96
## 4   II Marvellous 0.6cwt 144
## 5  III Marvellous 0.0cwt  89
## 6  III Marvellous 0.6cwt 124
## 7   IV Marvellous 0.0cwt  70
## 8   IV Marvellous 0.6cwt 117
## 9    V Marvellous 0.0cwt  63
## 10   V Marvellous 0.6cwt  99
## 11  VI Marvellous 0.0cwt  97
## 12  VI Marvellous 0.6cwt 121
```

Fyll i stycket nedan för att skapa en graf med `N` på x-axeln, `Y` på y-axeln och en gruppering som länkar observationer från samma block.

```
ggplot(dat_small_block, aes(x = ___, y = ___, group = ___)) +
  geom_point() +
  geom_line()
```

Uppgift 7.9 (Block med två behandlingar. Test)**.** Eftersom det är ett försök med en förklarande faktor och block kan man modellera det med den tidigare blockmodellen. Men eftersom man bara har två observationer per block kan man också se det som matchade stickprov, vilket kan lösas med ett t-test. Fyll i stycket nedan för att göra de två testen - utfallsvariabeln är skörd `Y` och den förklarande faktorn är kvävenivån `N`. Jämför resultaten.

```
mod <- lm(___ ~ ___ + B, data = dat_small_block)
Anova(mod)

t.test(___ ~ ___, data = dat_small_block, paired = ___)
```

Uppgift 7.10 (Majshybridimport)**.** I fliken *Majshybrider* i excelfilen *Uppgifts-data.xlsx* finns data på fyra majssorter, vardera sorterad på fem platser (som

agerar som block). Importera datan med funktionen `read_excel()` genom att fylla i kodstycket nedan.

```
dat_corn <- read_excel("", sheet = ___)
```

Uppgift 7.11 (Majshybridgraf)**.** Skapa en lämplig graf av datan på majshybrider. Grafen ska illustrera både jämförelsen mellan hybrider och jämförelsen mellan platser. Se exemplet ovan som guide.

Uppgift 7.12 (Majshybridmodell)**.** Fyll i koden nedan för att skatta en anovamodell med block för datan på majshybrider. Ta fram anovatabellen med `Anova()`. Vilka slutsatser kan man dra från anovatabellen?

```
mod <- lm(___ ~ ___ + Plats, data = dat_corn)
Anova(mod)
```

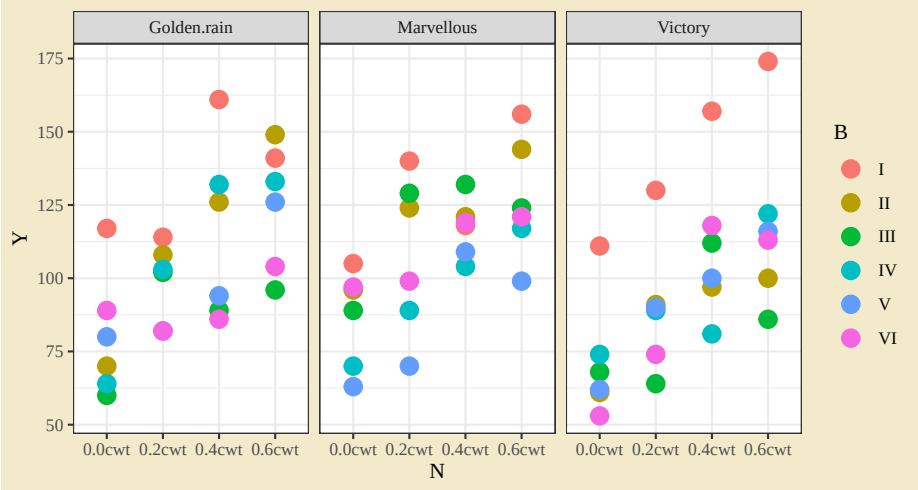
Uppgift 7.13 (Majshybridjämförelser)**.** Gör lämplig ändring i koden nedan för att jämföra hybrider, istället för platser.

```
emmeans(mod, pairwise ~ Plats)
```

7.5 Variansanalys. Två faktorer med block

Exempeldata på havre tar med två förklarande faktorer och ett block. Datan kan illustreras med ett punktdiagram där `facet_wrap` delar grafen efter sort.

```
ggplot(oats, aes(N, Y, color = B)) +
  geom_point(size = 4) +
  facet_wrap(~ V)
```



Grafen visar samma kvävesamband som tidigare. Det finns inga tydliga skillnader mellan sorter, möjligen har sorten Victory givit något lägre skörd än

övriga. Det finns fortfarande en tydlig blockeffekt, till exempel har block I höga värden och block V låga värden.

Modellen skattas genom att lägga till variabeln för sort (V för variety) i `lm`-formeln. En modell med två faktorer kan antingen vara med eller utan en *interaktion*. Interaktionstermen fångar påverkan mellan faktorerna. Ett exempel hade varit om någon sort svarat starkare på ökad kväve än någon annan. Standardmodellen är att ta med interaktionen, vilket vi anger genom att sätta `N * V` istället för `N + V`. Blocket tas fortfarande med som en adderad faktor

```
mod_two_fact <- lm(Y ~ N * V + B, data = oats)
```

Anovatabellen kan plockas fram på samma sätt som tidigare.

```
Anova(mod_two_fact)
```

```
## Anova Table (Type II tests)
##
## Response: Y
##            Sum Sq Df F value    Pr(>F)
## N         20020.5  3 26.2510 1.135e-10 ***
## V          1786.4  2  3.5134   0.03665 *
## B         15875.3  5 12.4894 4.093e-08 ***
## N:V         321.7  6  0.2109   0.97187
## Residuals 13982.1 55
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Raden `N:V` gäller interaktionseffekten mellan kväve och sort. I det här fallet är det ingen signifikant interaktion - vilket tyder på att sorterna svarar på kvävebehandling på liknande sätt. Samtliga huvudeffekter (raderna för N, V och B) är signifikanta. Kvadratsummorna och p-värdena tyder på att kväve förklarar mer av variationen än sort, vilket också är i linje med grafen ovan.

Vid flerfaktoriella försök kan man presentera parvisa jämförelser på flera olika sätt. Man kan ange huvudeffekter för en faktor utan att ange den andra faktorn, man kan ange medelvärden för samtliga kombinationer av två faktorer, och man kan ange medelvärden uppdelat efter nivåer i en annan faktor.

```
emmeans(mod_two_fact, ~ N)
```

```
##  N      emmean   SE df lower.CL upper.CL
##  0.0cwt   79.4 3.76 55     71.9     86.9
##  0.2cwt   98.9 3.76 55     91.4    106.4
##  0.4cwt  114.2 3.76 55    106.7    121.8
##  0.6cwt  123.4 3.76 55    115.9    130.9
##
## Results are averaged over the levels of: V, B
## Confidence level used: 0.95
```

```
emmeans(mod_two_fact, ~ N + V)
```

```
##  N      V           emmean   SE df lower.CL upper.CL
##  0.0cwt Golden.rain   80.0 6.51 55     67.0     93.0
##  0.2cwt Golden.rain   98.5 6.51 55     85.5    111.5
##  0.4cwt Golden.rain  114.7 6.51 55    101.6    127.7
##  0.6cwt Golden.rain  124.8 6.51 55    111.8    137.9
##  0.0cwt Marvellous    86.7 6.51 55     73.6     99.7
##  0.2cwt Marvellous   108.5 6.51 55     95.5    121.5
##  0.4cwt Marvellous   117.2 6.51 55    104.1    130.2
##  0.6cwt Marvellous   126.8 6.51 55    113.8    139.9
##  0.0cwt Victory       71.5 6.51 55     58.5     84.5
##  0.2cwt Victory       89.7 6.51 55     76.6    102.7
##  0.4cwt Victory      110.8 6.51 55     97.8    123.9
##  0.6cwt Victory      118.5 6.51 55    105.5    131.5
##
## Results are averaged over the levels of: B
## Confidence level used: 0.95
```

```
emmeans(mod_two_fact, ~ N | V)
```

```
## V = Golden.rain:
##  N      emmean   SE df lower.CL upper.CL
##  0.0cwt   80.0 6.51 55     67.0     93.0
##  0.2cwt   98.5 6.51 55     85.5    111.5
##  0.4cwt  114.7 6.51 55    101.6    127.7
##  0.6cwt  124.8 6.51 55    111.8    137.9
##
## V = Marvellous:
##  N      emmean   SE df lower.CL upper.CL
##  0.0cwt   86.7 6.51 55     73.6     99.7
##  0.2cwt  108.5 6.51 55     95.5    121.5
##  0.4cwt  117.2 6.51 55    104.1    130.2
##  0.6cwt  126.8 6.51 55    113.8    139.9
##
## V = Victory:
##  N      emmean   SE df lower.CL upper.CL
##  0.0cwt   71.5 6.51 55     58.5     84.5
##  0.2cwt   89.7 6.51 55     76.6    102.7
##  0.4cwt  110.8 6.51 55     97.8    123.9
##  0.6cwt  118.5 6.51 55    105.5    131.5
##
## Results are averaged over the levels of: B
## Confidence level used: 0.95
```

Även här kan man göra jämförelser mellan nivåer genom att sätta `pairwise ~ N + V` eller beräkna signifikansbokstäver med `cld`. Följande kod jämför

kvävenivåer *inom* sort.

Uppgift 7.14 (Sort uppdelat efter kvävenivå)**.** Gör lämplig ändring i koden ovan för att jämföra sorter *inom* kvävenivå. Finns det några signifikanta skillnader?

Uppgift 7.15 (Interaktion med ett block)**.** I modellen ovan är block en *additiv* faktor - den ingår inte i någon interaktionseffekt. Vad händer med testerna om man skattar modellen där samtliga interaktioner tas med? Varför?

```
mod_two_fact <- lm(Y ~ N * V * B, data = oats)
```

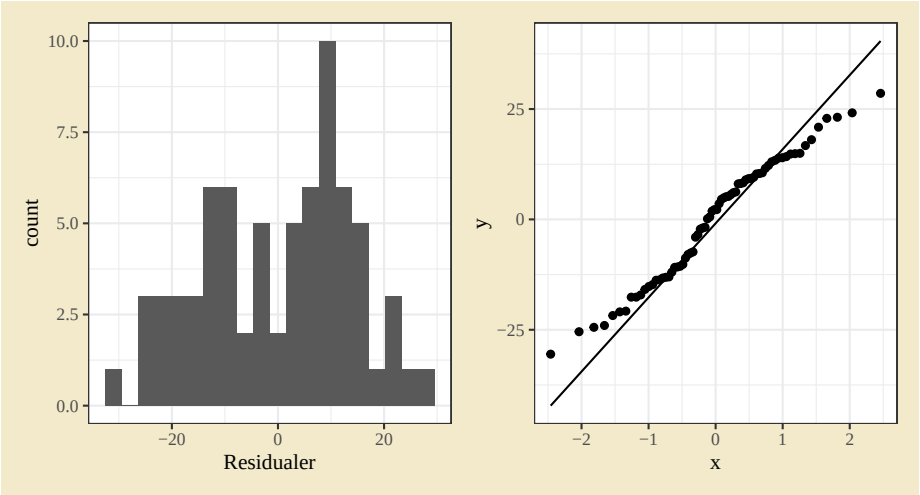
7.6 Modellantaganden och residualer

Samtliga anovamodeller har samma grundläggande antaganden: felltermerna (den kvarvarande slumpmässigheten) är normalfördelade, sinsemellan oberoende, och variansen är samma för samtliga behandlingsgrupper. Antagandena testas oftast genom att titta på modellens *residualer* - skillnaden mellan det faktiska värdet och det skattade värdet. För en skattad modell kan man ta upp residualerna med `residuals()` och de skattade värdena med `fitted()`. Vi kan lägga till residualer och skattningar till datan med ett `mutate()`-steg.

```
oats <- oats %>%
  mutate(Residualer = residuals(mod_two_fact),
         Skattade = fitted(mod_two_fact))
```

Normalfördelning kan undersökas grafiskt med ett histogram eller en QQ-graf.

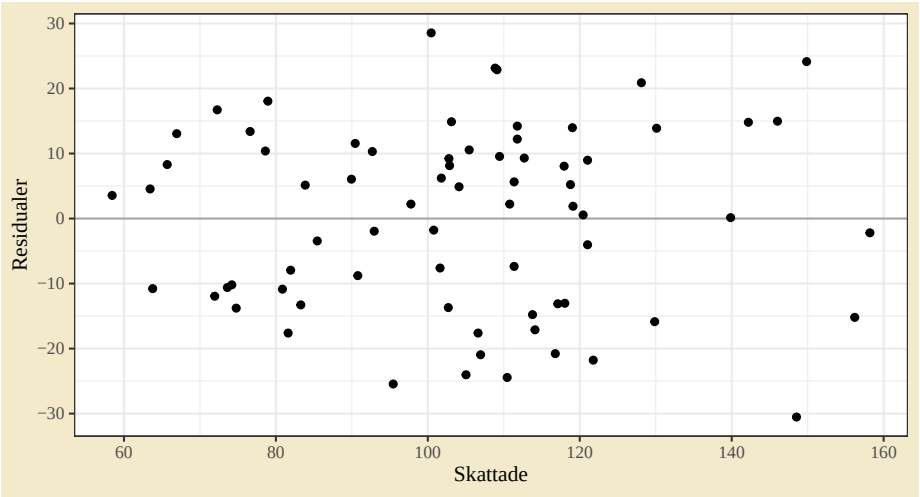
```
g_hist <- ggplot(oats, aes(Residualer)) + geom_histogram(bins = 20)
g_qq <- ggplot(oats, aes(sample = Residualer)) + geom_qq() + geom_qq_line()

library(patchwork)
g_hist + g_qq
```



Punkterna avviker något från normalfördelningen i svansarna, men det är förstås alltid en bedömningsfråga.

Lika varians undersöks ofta med ett spridningsdiagram med de skattade värdena på x-axeln och residualerna på y-axeln.

```
ggplot(oats, aes(x = Skattade, y = Residualer)) +
  geom_point() +
  geom_hline(yintercept = 0, alpha = 0.3)
```



Om datan är i linje med antaganden ska diagrammet se ut som slumpmässigt placerade punkter med ungefär lika stor spridning kring noll-linjen för samtliga nivåer på x-axeln. För det här exemplet ser det okej ut.

Uppgift 7.16 (Bakterieimport)**.** Fliken *Bakterier* i filen *Uppgiftsdata.xlsx* in-

nehåller data om tillväxt hos gräs efter inokulering av bakterier. Ladda ner filen och importera datan genom att fylla i koden nedan.

```
dat_bact <- read_excel("___", sheet = "Bakterier")
```

Uppgift 7.17 (Bakterieimport)**.** Illustrera datan med en lämplig graf, till exempel ett spridningsdiagram med `Inoculation` på x-axeln, `Dry weight` på y-axeln, småfönster efter `Cultivar` och färg efter `Block`.

```
ggplot(dat_bact, aes(x = ___, y = `___`, color = Block)) +
  geom_point(size = 6) +
  facet_wrap(~ ___)
```

Hur blev färgerna för blocket? Om de inte blev distinkta färger kan variabeln `Block` ha blivit inläst som numerisk. Transformera variabeln med `as.character()` och gör om grafen. Ändras färgerna?

```
dat_bact <- dat_bact %>%
  mutate(Block = as.character(Block))
```

Uppgift 7.18 (Bakteriemodell)**.** Bakteriedatan har två faktorer och en blockfaktor. Skatta en anova-modell med interaktion och block genom att fylla i stycket nedan. Ta fram anovatabell och dra en slutsats från F-testen. Ligger slutsatsen i linje med grafen?

```
mod <- lm(`___` ~ ___ * ___ + ___, data = dat_bact)
Anova(mod)
```

Uppgift 7.19 (Bakteriejämförelser)**.** Använd `emmeans()` för parvisa jämförelser mellan inokuleringsmetoder. Vilka par är signifikant åtskilda?

```
emmeans(mod, pairwise ~ ___)
```

Uppgift 7.20 (Bakterieresidualer)**.** Vi använder den skattade modellen för att ta fram skattade värden och residualer.

```
dat_bact <- dat_bact %>%
  mutate(Residualer = residuals(mod),
         Skattade = fitted(mod))
```

Använd exemplet på residualtester ovan för att undersöka antagandet om normalfördelade residualer.

7.7 Bonus. Statistik för ekologi

Här tittar vi på några statistiska metoder som är vanliga inom ekologin, men går bortom materialet på en statistisk grundkurs. Vi börjar med datastruktur och visualisering för populationsdata, för att sedan titta på diversitetsmått, principalkomponentanalys (PCA) och hierarkisk klustering. Det vanligaste paketet

för ekologi är `vegan`, så vi kan börja med att installera och ladda det. Vi kommer också använda `factoextra` för en graf.

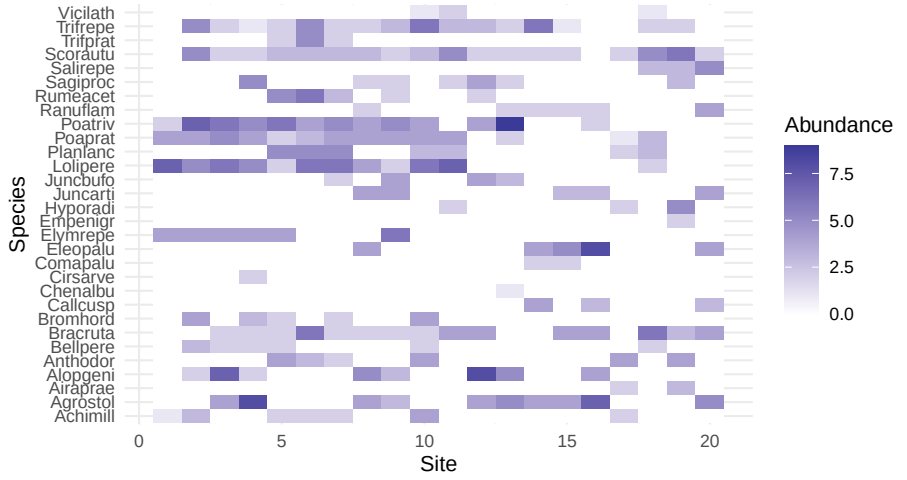
```
# install.packages("vegan")
library(vegan)

# install.packages("factoextra")
library(factoextra)
```

Data för ekologiska populationer för flera platser eller tillfällen ordnas oftast i en tabell med plats som rad och arter som kolumner. Värdena i tabellen anger antingen antalet observerade individer eller ett binärt utfall (1 för förekomst, 0 för ingen förekomst). Exempeldatan `dune`, som kan laddas med funktionen `data()`, ger ett exempel. För att illustrera en typisk jämförelsestudie skapar vi en kolumn för platstyp och lägger till ett plats-id.

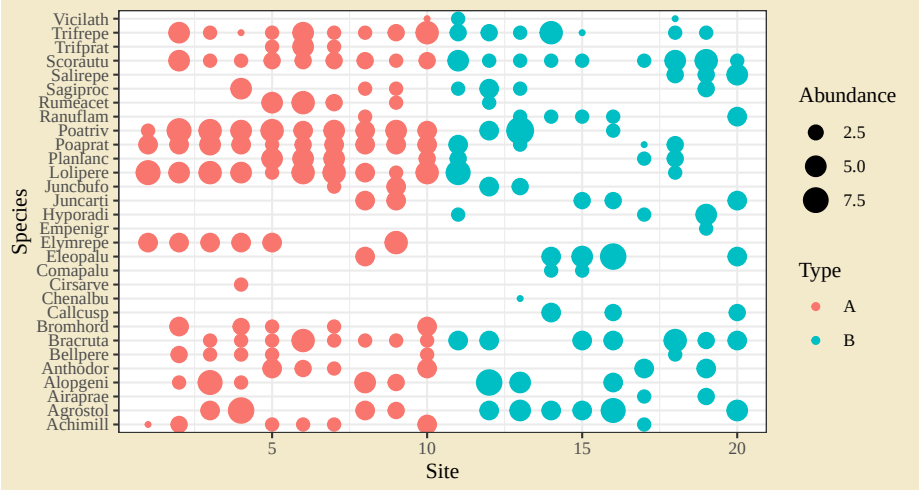
```
data(dune)
dune <- dune %>%
  mutate(Site = 1:n(),
         Type = rep(c("A", "B"), each = 10))
```

Vi kan illustrera data genom att pivotera till långt format och göra en graf med `ggplot()`. En heatmap eller ett spridningsdiagram med storlek för antal observationer kan vara lämpliga grafer. Tolkning kräver förstås kod artkännedom och beror på den vetenskapliga frågan.

```
dune_long <- dune %>%
  pivot_longer(-c(Site, Type), names_to = "Species", values_to = "Abundance")

ggplot(dune_long, aes(Site, Species, fill = Abundance)) +
  geom_tile() +
  scale_fill_gradient2() +
  theme_minimal()
```




```
ggplot(dune_long %>% filter(Abundance > 0), aes(Site, Species, size = Abundance, color
  geom_point()
```



Ytterligare alternativ kan vara upprepade lådagram eller staplar med småfönster per art.

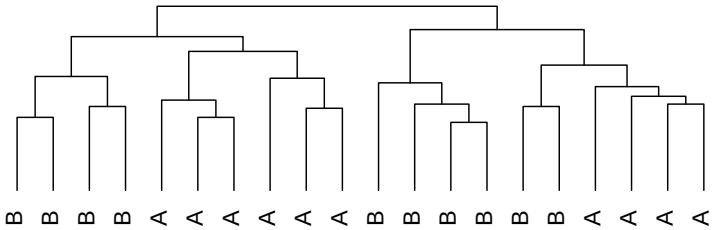
Uppgift 7.21 (Populationsgrafer)**.** Vad måste läggas till i stycket nedan för göra ett lådagram (med art på y-axeln och abundans på x-axeln) och ett stapeldiagram (med platstyp på x-axeln och abundans på y-axeln)?

```
dune_long
ggplot(dune_long, aes(x = ___, y = ___, fill = Type)) +
  geom_boxplot()

ggplot(dune_long, aes(x = ___, y = ___, fill = Type)) +
```

```
  geom_col() +
  facet_wrap(~ ___, nrow = 2)
```

Ekologiska populationer kan analyseras genom *heirarkisk klustring* - metoder där platser (rader) eller arter (kolumner) sorteras efter hur lika de är. Först beräknas ett avstånd mellan samtliga enheter (platser eller arter) och därefter sker klustringen genom att slå ihop enheter som ligger *nära* varandra. Resultatet illustreras med ett träddiagram.

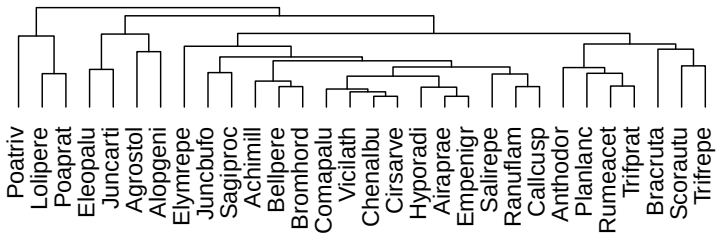
```
dune_data <- dune %>% select(-Site, -Type)
d <- dist(dune_data, method = "euclidean")
hc <- hclust(d)
plot(hc, hang = -1, labels = dune$Type,
     axes = F, xlab = "", ylab = "", ann = F)
```



Uppgift 7.22 (Avståndsmått)**.** Ta upp hjälpsidan till distansfunktionen med `?dist`. Under `method` finns flera möjliga avståndsmått. Vad måste ändras i kodstycket ovan för att ange ett Manhattan-avstånd? Har avståndet någon betydande effekt på träddiagrammet?

För att göra en klustring av arter kan man *transponera* data så att rader och kolumner byter plats med varandra. Här kommer artnamn automatisk med eftersom raderna i datan har namn. Det är inte alltid fallet, så det kan vara nödvändigt att sätta etiketter med argumentet `labels` i `plot()`.

```
dune_data <- t(dune_data)
d <- dist(dune_data, method = "euclidean")
hc <- hclust(d)
plot(hc, hang = -1,
     axes = F, xlab = "", ylab = "", ann = F)
```

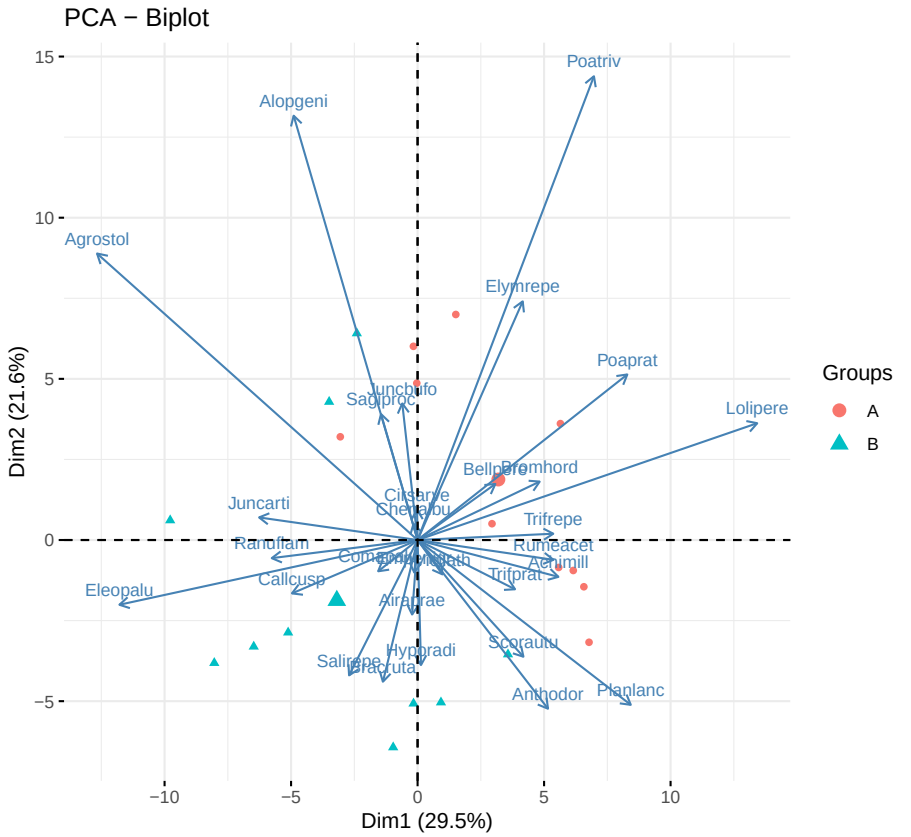


Träddiagrammet tolkas så att enheter vars koppling ligger lågt är mer lika varandra - arterna förekommer ofta på samma plats.

En annan vanlig metod för *multivariat* data, vilket populationsdata är ett exempel på, är *principalkomponentsanalys* (PCA, Principal Component Analysis). En PCA är ett försök att sammanfatta den ursprungliga datans 30 variabler (en per art) med ett mindre antal variabler. De nya variablerna - *komponenterna* - skapas genom att väga och addera de ursprungliga variablerna på ett sätt som förklarar så mycket som möjligt av variationen med minsta möjliga antal variabler. Resultatet illustreras vanligen med en *biplot* - ett spridningsdiagram som placerar ut både platser och arter.

I R kan en PCA göras med `prcomp()` och en biplot kan göras med `fviz_pca_biplot()` från `factoextra`.

```
dune_data <- dune %>% select(-Site, -Type)
pca <- prcomp(dune_data, scale. = F)
fviz_pca_biplot(pca, geom.ind = "point", habillage = dune$Type, labelsize = 3)
```



Platserna illustreras med punkter och arterna med pilar. Pilar i samma riktning motsvarar arter som är lika (de finns på samma platser), närliggande punkter motsvarar lika platser (de har samma arter), och punkter i samma riktning som en pil har höga värden för den arten.

Uppgift 7.23 (Skalning i en PCA)**.** En PCA kan göras med och utan att skala variablerna. Om variablerna skalas får en variabel som varierar mycket samma vikt som en variabel som varierar lite. Det kan vara bra om man har variabler som är mätta på olika sätt, till exempel om en variabel är i meter och en är i centimeter. Gör lämplig ändring i kodstycket ovan för att skala variablerna i `prcomp()`. Har det någon effekt på grafen?

Den sista ansatsen vi ska titta på är att sammanfatta en population i ett enskilt tal - ett diversitetsindex. Genom att beräkna ett index kan man reducera datan till en observation på plats. Man kan därifrån tillämpa de metoder vi sett i övriga delar av kursen (t-test och variansanalys). Det finns en stor mängd olika index. Det vanligaste än Shannon-Weaver indexet (eller entropi), vilket beräknas genom att ta andelen per art, multiplicera med logaritmen av andelen, summera över arter, och multiplicera med minus ett. Om man har tre arter

med andelarna 0.3, 0.5 och 0.2 ges Shannon-Weaver alltså av

```
-(0.3 * log(0.3) + 0.5 * log(0.5) + 0.2 * log(0.2))
```

```
## [1] 1.029653
```

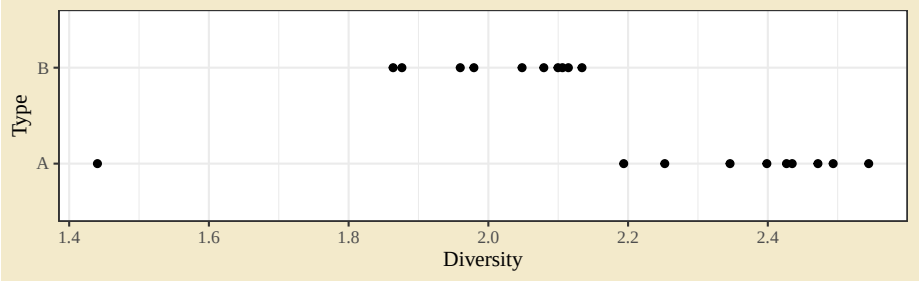
Indexet ökar om det finns många arter och om andelen per art är samma. En population med *en* dominant art kommer alltså ha ett lågt index.

För en tabell med data kan index beräknas med `diversity()`.

```
diver <- diversity(dune_data, index = "shannon")
dune <- dune %>% mutate(Diversity = diver)
```

Diversitetsindexen kan sedan illustreras och analyseras som vilken numerisk variabel som helst.

```
ggplot(dune, aes(Diversity, Type)) + geom_point()
```




```
mod <- lm(Diversity ~ Type, data = dune)
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: Diversity
##             Sum Sq Df F value  Pr(>F)
## Type       0.37586  1  6.6647 0.01881 *
## Residuals 1.01513 18
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
emmeans(mod, ~ Type)
```

```
##  Type emmean     SE df lower.CL upper.CL
##  A      2.30 0.0751 18     2.14     2.46
##  B      2.03 0.0751 18     1.87     2.18
##
## Confidence level used: 0.95
```

Här finns en signifikant skillnad med platstyper.

Uppgift 7.24 (Diversitetsindex)**.** Ta upp hjälpsidan till funktionen `diversity()`. Hur anger man att funktionen ska ge Simpsons index?

Uppgift 7.25 (Test på nytt index)**.** Gör om analysen på diversitet (anovamodellen och F-testet) med Simpsons index istället för Shannon-Weaver. Påverkar valet av diversitetsindex utfallet av testet?

Chapter 8

Regression och korrelation

Datorövning 8 handlar om regression och korrelation. Efter övningen ska vi kunna

- skatta en regressionsmodell i R,
- testa parametrar i modellen med F-test och t-test,
- göra lämpliga tester av modellantaganden,
- beräkna och tolka korrelationen mellan två variabler.

8.1 Repetition av datorövning 7

När man startar en ny R-session bör man ladda de paket man vet kommer behövas med `library()`. Om paket inte finns installerade måste man först köra `install.packages()`.

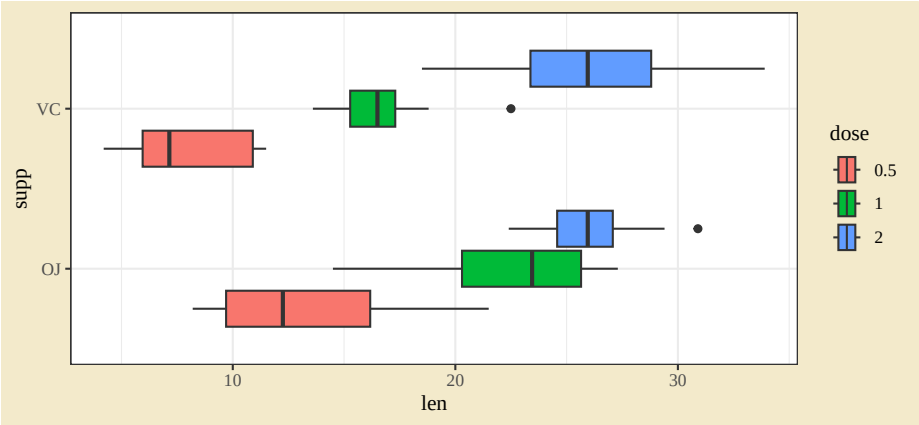
```
# install.packages("tidyverse")
library(tidyverse)
```

I datorövning 7 tittade vi på variansanalys - en metod som gör det möjligt att utvidga t-testet för två grupper till ett godtyckligt antal grupper eller kombinationer av faktorer. I variansanalys skattar man en modell som förklarar ett datautfall. Utifrån modellen sätter man upp en anova-tabell som delar upp den totala variansen i en förklarad del och en kvarvarande residualdel. Anova-tabell ger också ett F-test som testar om det finns några skillnader mellan grupper. Från en skattad modell kan man sedan göra parvisa jämförelser mellan specifika grupper och testa modellantaganden (främst antagande om normalfördelning och lika varians inom grupper).

Ta som exempel följande data på tandtillväxt (`len`) hos marsvin under C-vitaminbehandling i olika doser (`dose`) och två olika metoder (`supp`), tillgängligt

i R som objektet `ToothGrowth`.

```
ToothGrowth <- ToothGrowth %>% mutate(dose = as.character(dose))

ggplot(ToothGrowth, aes(len, supp, fill = dose)) +
  geom_boxplot()
```



Ett lådagram visar en klar skillnad mellan doser och en svagare skillnad mellan metoder. Det finns också tecken på att metoderna svarar olika på dos i att metoden *VC* ligger lägre än *OJ* vid de låga doserna men över (eller iallafall lika) vid den höga dosen.

En envägsanova-modell (en modell med en faktor) kan skattas med `lm()` och en anovatabell kan tas fram med `Anova()` från paketet `car`.

```
mod <- lm(len ~ dose, data = ToothGrowth)

library(car)
Anova(mod)
```

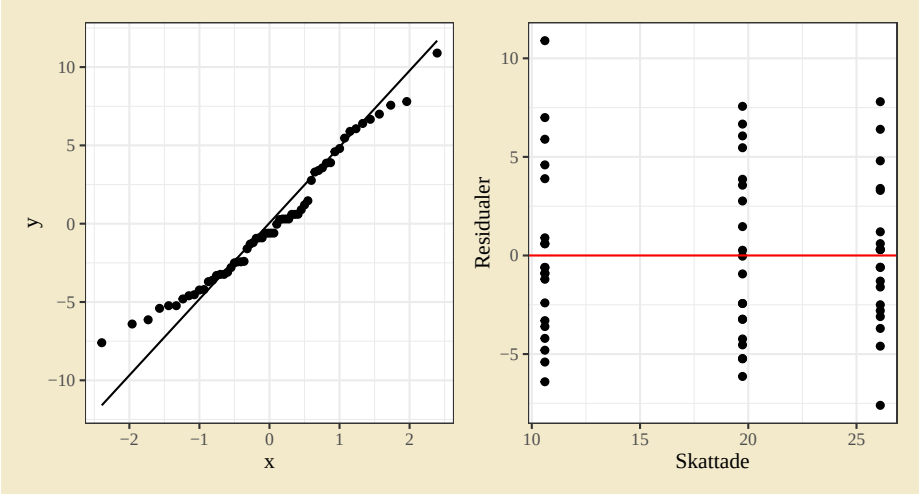
```
## Anova Table (Type II tests)
##
## Response: len
##           Sum Sq Df F value    Pr(>F)
## dose      2426.4  2  67.416 9.533e-16 ***
## Residuals 1025.8 57
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

F-testets nollhypotes är att alla grupper (här alla doser) har samma populationsmedelvärde. Det låga p-värdet pekar på en klar skillnad mellan doser.

En anovamodell bygger på antaganden om normalfördelning och lika varianser. Normalfördelningsantagandet kan undersökas med en QQ-graf över residualerna

och variansantagandet kan undersökas med en spridningsgraf över skattade värden och residualer.

```
library(patchwork)
ToothGrowth <- ToothGrowth %>%
  mutate(Skattade = fitted(mod),
         Residualer = residuals(mod))
g1 <- ggplot(ToothGrowth, aes(sample = Residualer)) + geom_qq() + geom_qq_line()
g2 <- ggplot(ToothGrowth, aes(Skattade, Residualer)) +
  geom_point() +
  geom_hline(yintercept = 0, color = "red")
g1 + g2
```



Punkterna ligger ungefär på linjen i QQ-grafen och punkterna har ungefär samma spridning för alla nivåer av det skattade värdet.

Anova-modeller kan lätt byggas ut genom att lägga till fler faktorer. Här är det till exempel naturligt att skatta en modell med både metod och dos, vilket kan göras genom att lägga till `supp` till formeln i `lm()`.

```
mod <- lm(len ~ dose * supp, ToothGrowth)
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: len
##             Sum Sq Df F value    Pr(>F)
## dose       2426.43  2  92.000 < 2.2e-16 ***
## supp        205.35  1  15.572 0.0002312 ***
## dose:supp   108.32  2   4.107 0.0218603 *
## Residuals   712.11 54
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Resultaten är i linje med grafen - dos har en stor effekt medan metod och interaktionen mellan dos och metod är något svagare, om än signifikanta.

En anovamodell kan användas för parvisa jämförelse, vilket ibland kallas post-hoc-test. Den vanligaste är Tukey-testet, men andra tester kan också förekomma. Testet kan utföras med `emmeans()` från paketet med samma namn. Följande ger en jämförelse mellan doser uppdelat efter metod.

```
library(emmeans)
emmeans(mod, pairwise ~ dose | supp)
```

```
## $emmeans
## supp = OJ:
##  dose emmean   SE df lower.CL upper.CL
##  0.5   13.23 1.15 54    10.93     15.5
##  1     22.70 1.15 54    20.40     25.0
##  2     26.06 1.15 54    23.76     28.4
##
## supp = VC:
##  dose emmean   SE df lower.CL upper.CL
##  0.5    7.98 1.15 54     5.68     10.3
##  1     16.77 1.15 54    14.47     19.1
##  2     26.14 1.15 54    23.84     28.4
##
## Confidence level used: 0.95
##
## $contrasts
## supp = OJ:
##  contrast        estimate   SE df t.ratio p.value
##  dose0.5 - dose1    -9.47 1.62 54  -5.831  <.0001
##  dose0.5 - dose2   -12.83 1.62 54  -7.900  <.0001
##  dose1 - dose2      -3.36 1.62 54  -2.069  0.1060
##
## supp = VC:
##  contrast        estimate   SE df t.ratio p.value
##  dose0.5 - dose1    -8.79 1.62 54  -5.413  <.0001
##  dose0.5 - dose2   -18.16 1.62 54 -11.182  <.0001
##  dose1 - dose2      -9.37 1.62 54  -5.770  <.0001
##
## P value adjustment: tukey method for comparing a family of 3 estimates
```

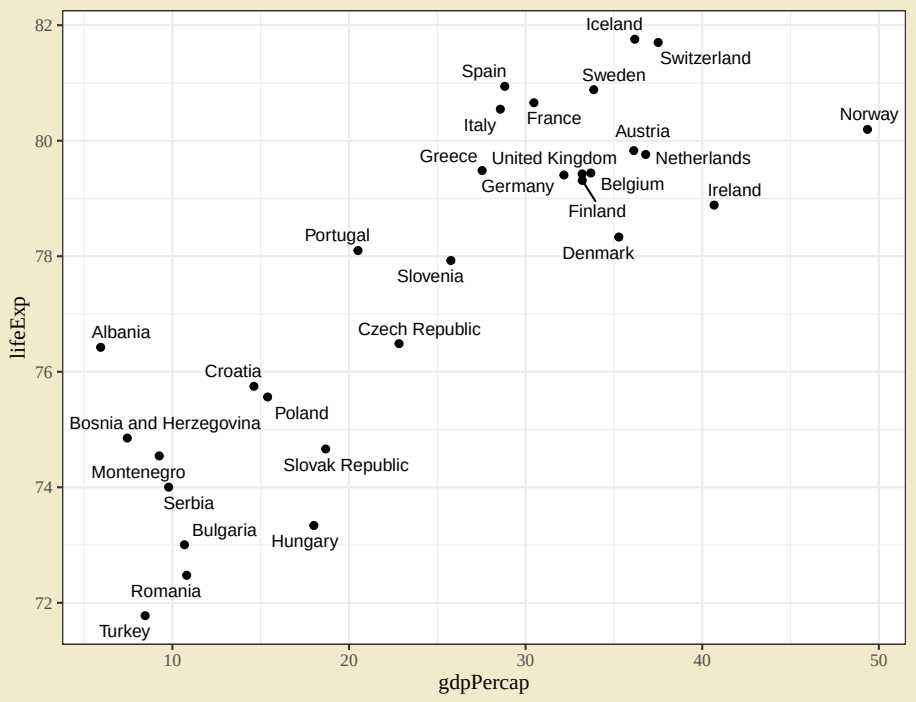
8.2 Regression

I en regression modelleras en numerisk variabel som en funktion av en annan numerisk variabel. Vid enkel linjär regression finns *en* sådan *förklarande variabel*

och förhållandet mellan variablerna antas vara linjärt.

Ta som exempel data på förväntad medellivslängd och bnp per capita. Datan hämtas från `gapminder`-paketet. Paketet `ggrepel` och funktionen `geom_text_repel()` kan användas för att sätta punktetiketter som inte överlappar. För enklare tolkning av modellen transformeras bnp per capita till att vara i tusen dollar, snarare än dollar.

```
library(gapminder)
dat_eu07 <- gapminder %>%
  filter(year == 2007, continent == "Europe") %>%
  mutate(gdpPercap = gdpPercap / 1000)

library(ggrepel)
ggplot(dat_eu07, aes(gdpPercap, lifeExp)) +
  geom_point() +
  geom_text_repel(aes(label = country), size = 3)
```



Datan visar ett positivt samband mellan variablerna - högre bnp per capita är kopplat till högre medellivslängd.

Uppgift 8.1 (Data för 1957)**.** Vad måste ändras i stycket nedan för att plocka ut data och göra en graf för Europa 1957?

```
dat_eu57 <- gapminder %>%
  filter(year == 2007, continent == "Europe") %>%
  mutate(gdpPercap = gdpPercap / 1000)

ggplot(dat_eu57, aes(gdpPercap, lifeExp)) +
  geom_point() +
  geom_text_repel(aes(label = country), size = 3)
```

En regressionmodell kan i R skattas med `lm`-funktionen. Syntaxen är väldigt lik den för anovamodellen, men istället för en faktor som förklarande variabel används nu en kontinuerlig variabel.

```
mod <- lm(lifeExp ~ gdpPercap, data = dat_eu07)
summary(mod)
```

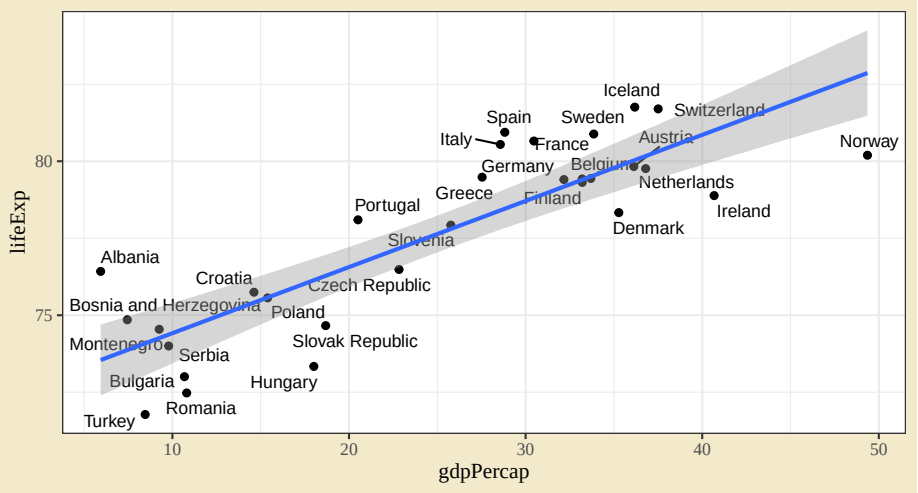
```
##
## Call:
## lm(formula = lifeExp ~ gdpPercap, data = dat_eu07)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.79839 -1.30472  0.00807  1.33443  2.87766
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 72.27106    0.69416 104.113  < 2e-16 ***
## gdpPercap    0.21463    0.02514   8.537  2.8e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.598 on 28 degrees of freedom
## Multiple R-squared:  0.7225, Adjusted R-squared:  0.7125
## F-statistic: 72.88 on 1 and 28 DF,  p-value: 2.795e-09
```

Funktionen `summary` ger en sammanfattning av modellen. Skattningen av modellens konstanta parameter ges som raden `(Intercept)` och dess tolkning är som förväntat värde i medellivslängd om bnp per capita är noll. Det är ofta lutningsparametern som är mer intressant. Skattningen av lutningsparametern ges på den rad som har samma namn som den förklarande variabeln, här `gdpPercap`. Den skattade parametern är 0.2146. Lutningsparametern har den generella tolkning som ökningen i y-variabeln när x-variabeln ökar med 1. I det här fallet ger 0.2146 att ett lands medellivslängd ökar med ungefär 0.2146 år (eller 78 dagar) när bnp per capita ökar med 1000 dollar.

Uppgift 8.2 (Modell för 1957)**.** Skatta samma modell som ovan, denna gång med data från 1957. Tolka lutningsparametern i ord. Är effekten av ökad bnp större 2007 än den var 1957?

Man kan enkelt rita ut regressionlinjen i en graf med `geom_smooth()` och argumentet `method` satt till `lm`.

```
ggplot(dat_eu07, aes(gdpPercap, lifeExp)) +
  geom_point() +
  geom_text_repel(aes(label = country), size = 3) +
  geom_smooth(method = lm)
```



Den blå linjen illustrerar regressionlinjen $72.27 + 0.2146x$. Det grå bandet kring linjen är ett konfidensintervall för skattningen av y-variabeln.

Uppgift 8.3 (Graf för 1957)**.** Använd `geom_smooth(method = lm)` för att lägga till en regressionslinje för data för 1957. Hur mycket påverkar de två avvikande länderna?

Utskriften från `summary` ger också tester av parametrarna (den högra kolumnen `Pr(>|t|)` ger p-värdet för ett test där nollhypotesen är att populationsparametern är noll). I det här fallet är både intercept och lutning skilda från noll. Motsvarande F-test för lutningen kan tas fram med en anova-tabell.

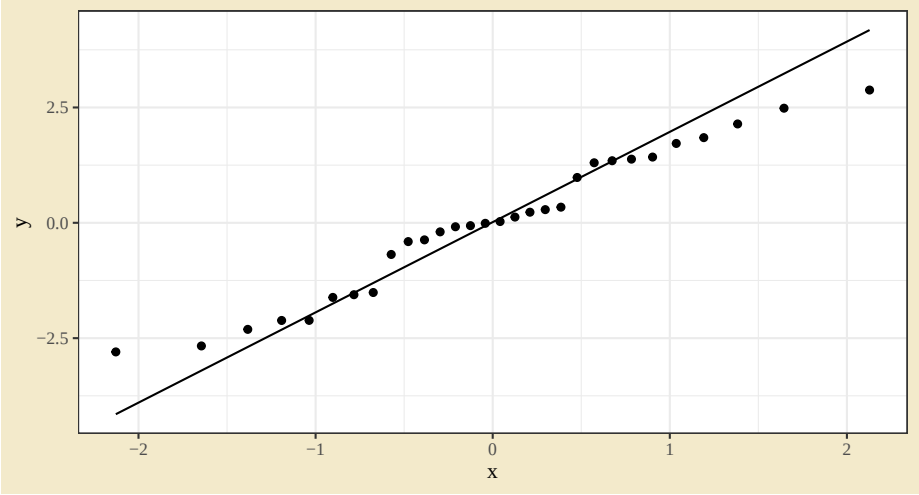
```
library(car)
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: lifeExp
##            Sum Sq Df F value    Pr(>F)
## gdpPercap 186.031  1  72.883 2.795e-09 ***
## Residuals  71.469 28
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

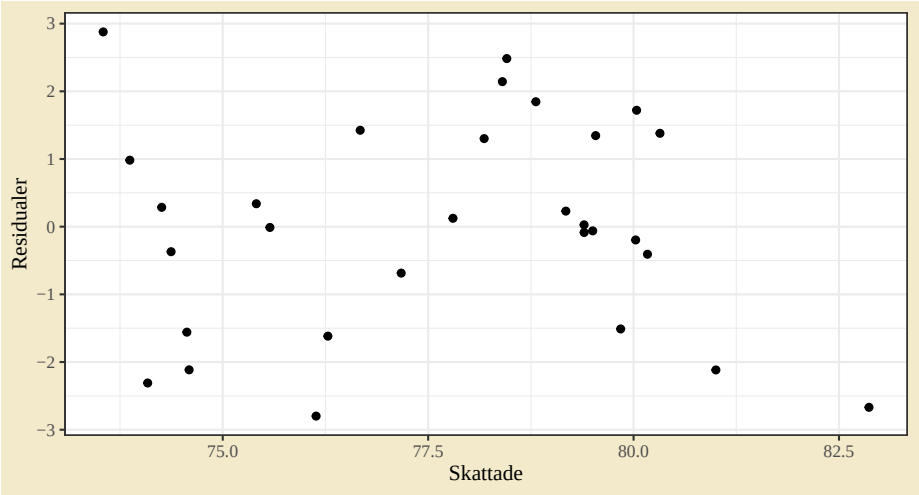
Testerna av en regressionsmodell bygger på ett normalfördelningsantagande och

ett antagande om *homoskedasticitet* (lika varians i y oavsett position på x-axeln). Antagandena kan undersökas genom att titta på skattningens *residualer* - skillnaden mellan det faktiska y-värdet och modellens värde. Residualerna kan undersökas med ett histogram eller en QQ-plot. En annan vanlig diagnosplot är ett spridningsdiagram med skattade värden på x-axeln och residualerna på y-axeln.

```
dat_eu07 <- dat_eu07 %>%
  mutate(Residualer = residuals(mod),
         Skattade = fitted(mod))

ggplot(dat_eu07, aes(sample = Residualer)) + geom_qq() + geom_qq_line()
```




```
ggplot(dat_eu07, aes(Skattade, Residualer)) + geom_point()
```



Om data följer en normalfördelning bör histogrammet visa en ungefärlig normalkurva, QQ-plotten bör visa punkter på den diagonala linjen och spridningsdiagrammet bör visa en slumpmässig spridning av punkter. Graferna pekar i det här fallet inte på några tydliga avvikelser från normalfördelningsantagandet, möjligen pekar QQ-plotten på mindre spridning i svansarna än en teoretisk normalfördelning.

Uppgift 8.4 (Diagnos för 1957)**.** Gör lämpliga ändringar i data ovan för diagnosgrafer för data från 1957. Finns det några tydliga avvikande värden?

Uppgift 8.5 (Icke-linjära samband)**.** Låt oss titta på hela gapminder-datan för 2007.

```
dat_2007 <- gapminder %>% filter(year == 2007)
ggplot(dat_2007, aes(gdpPercap, lifeExp)) + geom_point()
```

Hur ser sambandet mellan bnp och medellivslängd ut? Vad skulle vara problematiskt med simpel linjär regression i det här fallet? När vi tittade på normalfördelningen sa vi att man ofta kan logaritmera en variabeln och få *bättre* egenskaper. Vad ska ändras i koden ovan för att använda logaritmerad `gdpPercap` istället för den ursprungliga variabeln? Är det sambandet mer linjärt?

Uppgift 8.6 (Log-transformerad data)**.** Vad ska ändras i koden nedan för att använda logaritmerad `gdpPercap` istället för den ursprungliga variabeln? Är det sambandet mer linjärt?

```
dat_2007 <- gapminder %>% filter(year == 2007)
ggplot(dat_2007, aes(gdpPercap, lifeExp)) + geom_point()
```

Uppgift 8.7 (Blodtrycksdata)**.** Gör lämplig ändring i stycket nedan för att läsa in fliken *Blodtrycksdata* från filen *Uppgiftsdata.xlsx*.

```
library(readxl)
dat_blod <- read_excel("___", sheet = "Blodtryck")
```

Uppgift 8.8 (Blodtrycksgraf)**.** Gör ett spridningsdiagram med ålder på x-axeln och blodtryck på y-axeln. Lägg till en regressionslinje med `geom_smooth(method = lm)`.

```
ggplot(___, aes(x = ___, y = ___)) +
  ___() +
  ___()
```

Uppgift 8.9 (Blodtrycksmodell)**.** Skatta och tolka en regressionmodell med ålder som förklarande variabel och blodtryck som förklarad variabel.

```
mod <- lm(___ ~ ___, data = dat_blod)
```

Uppgift 8.10 (Blodtryckstest)**.** Använd `Anova()` för att testa om det finns ett signifikant samband mellan ålder och blodtryck. Vad är testets nollhypotes och alternativhypotes?

Uppgift 8.11 (Blodtrycksdiagnos)**.** Ta fram diagnosgrafer för blodtrycksmodell och avgör om det finns några tydliga avvikelser från normalfördelning eller några extrema värden.

```
dat_blod <- dat_blod %>%
  mutate(Residualer = residuals(mod),
         Skattade = fitted(mod))

ggplot(___, aes(sample = ___)) + geom_qq() + geom_qq_line()
ggplot(___, aes(Skattade, ___)) + geom_point()
```

8.3 Korrelation

Korrelation ger ett mått mellan -1 och 1 på hur väl två variabler samvarierar. En korrelation över noll tyder på ett positivt samband mellan variablerna - en observation med ett högt värde i den ena variabeln har också ett högt värde på den andra - medan en korrelation under noll tyder på ett negativt samband. I R kan korrelation beräknas med `cor()` och två variabler som första och andra argument. Funktionen `cor.test()` ger ett test där nollhypotesen är att korrelationen är noll.

```
cor(dat_eu07$lifeExp, dat_eu07$gdpPercap)
```

```
## [1] 0.8499711
```

```
cor.test(dat_eu07$lifeExp, dat_eu07$gdpPercap)
```

```
##
##  Pearson's product-moment correlation
##
## data:  dat_eu07$lifeExp and dat_eu07$gdpPercap
## t = 8.5372, df = 28, p-value = 2.795e-09
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.7058444 0.9265221
## sample estimates:
##       cor
## 0.8499711
```

Medellivslängd och bnp per capita har en stark positiv korrelation på 0.85 och den korrelation är signifikant skild från noll ($p < 0.001$). Notera att p-värdet är detsamma som för lutningsparametern i regressionen.

Uppgift 8.12 (Korrelationsmatris)**.** Om man har fler än två variabler sammanfattas korrelationer ofta med en korrelationsmatris.

```
dat_eu07[, 4:6]
```

```
## # A tibble: 30 x 3
##    lifeExp      pop gdpPercap
##      <dbl>    <dbl>     <dbl>
##  1    76.4  3600523      5.94
##  2    79.8  8199783     36.1
##  3    79.4 10392226     33.7
##  4    74.9  4552198      7.45
##  5    73.0  7322858     10.7
##  6    75.7  4493312     14.6
##  7    76.5 10228744     22.8
##  8    78.3  5468120     35.3
##  9    79.3  5238460     33.2
## 10    80.7 61083916     30.5
## # i 20 more rows
```

```
cor(dat_eu07[, 4:6])
```

```
##              lifeExp        pop gdpPercap
## lifeExp   1.00000000 0.06946716 0.8499711
## pop       0.06946716 1.00000000 0.0137427
## gdpPercap 0.84997107 0.01374270 1.0000000
```

Vad är korrelationen mellan befolkningsstorlek och bnp per capita?

Uppgift 8.13 (Anscombes data)**.** Den raka regressionslinjen eller det enkla korrelationsmåttet säger lite om hur data egentligen ser ut. En vanlig illustration av detta är *Anscombes kvartett*, fyra exempel konstruerade av den brittiske statistikern Francis Anscombe 1973. Datan finns tillgänglig i R som datasetet `anscombe`.

```
anscombe
```

Plotta de fyra graferna (`x1` paras med `y1` och så vidare) i spridningsdiagram och beräkna korrelation för varje par. Ett exempel ges för den första mängden nedan. Kommentera utfallet.

```
ggplot(anscombe, aes(x1, y1)) + geom_point()
cor(anscombe$x1, anscombe$y1)
```

Uppgift 8.14 (Datasaurus Dozen. Beskrivande mått)**.** Datasaurus-datan är en konstruerad datamängd som illustrerar hur skilda mönster i data kan ge samma punktskattningar (medelvärden, standardavvikelser och korrelationer). Datan finns tillgänglig som en del av TidyTuesday-projektet och kan hämtas med följande rad.

```
dat_saurus <- read_csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/data
```

Datan innehåller en gruppering (`dataset`) och x- och y-koordinater. Beräkna medelvärden, standardavvikelser och korrelation för varje grupp i `dataset` genom att fylla i stycket nedan.

```
dat_saurus %>%
  group_by(___) %>%
  summarise(mean(x), mean(y), sd(x), sd(y), cor(x, y))
```

Kommentera utfallet.

Uppgift 8.15 (Datasaurus Dozen. Grafer)**.** Illustrera datasaurus datan med spridningsdiagram. Använd `facet_wrap()` för småfönster per `dataset`.

```
ggplot(dat_saurus, aes(x, y)) +
  geom_point() +
  facet_wrap(~ ___)
```

Uppgift 8.16 (Galtons längdstudier. Installation av paket)**.** En modern förståelse av regression införs under slutet av 1800-talet av Francis Galton (1822 - 1911). I en studie från 1886 samlade Galton in data på längder hos föräldrar och barn. En av Galtons slutsatser från den datan var att barn till långa föräldrar ofta blev kortade än föräldrarna. Extremvärden hade en tendes att *återgå* mot mitten - härifrån kommer namnet *regression*.

Galtons längddata finns tillgänglig i paketet `HistData` som `Galton`. Installera paketet, ladda paketet, och skriv ut datan.

```
install.packages("___")
library(___)
Galton
```

Datan är i tum. Om man föredrar cm kan man multiplicera med 2.54.

```
Galton <- 2.54 * Galton
```

Uppgift 8.17 (Galtons längdstudier. Graf)**.** Gör en graf med föräldrars medellängd (`parent`) och barnets längd (`child`). Eftersom det finns överlappande punkter kan man använda `geom_count()` eller `geom_jitter()` istället för `geom_point()`.

```
ggplot(Galton, aes(parent, child)) + geom_count()
ggplot(Galton, aes(parent, child)) + geom_jitter()
```

Uppgift 8.18 (Galtons längdstudier. Modell)**.** Skatta en regressionmodell med barnets längd som förklarad variabel och förälderns längd som förklarande variabeln. Skriv ut resultaten och tolka lutningsparametern. Gör ett F-test med `Anova()`.

```
mod <- lm(___ ~ ___, Galton)
summary(___)
Anova(___)
```

Uppgift 8.19 (Galtons längdstudier. Konfidensintervall)**.** Paketet `emmeans()`, som vi tidigare använt för att ta fram effekter i anovamodeller, har också en

funktion för lutningsparametrar `emtrends()`. Vi kan använda den funktionen för att beräkna konfidensintervall för lutningen.

```
library(emmeans)
emtrends(mod, ~ 1, var = "parent")
```

Funktionen `emmeans()` kan också användas för ett konfidensintervall för barnets längd vid ett specifikt värde för föräldrarnas längd. Följande ger ett konfidensintervall för barnets längd om föräldrarnas medellängd är 170 cm.

```
emmeans(mod, ~ parent, at = list(parent = 68))
```

Vad ska ändras i stycket ovan för att beräkna ett konfidensintervall för barnets längd om föräldrarnas medellängd är 190 cm?

Uppgift 8.20 (Galtons längdstudier. Diagnosgrafer)**.** Galtondatan omfattar 928 mätningar. Ta ut residualerna med `residuals(mod)` och gör ett histogram med `hist()` eller `geom_histogram()`. Följer residualerna en ungefärlig normalfördelning?

8.4 Bonus. Skrapa data från webbsidor

Det är väldigt vanligt att hämta in data från externa källor för att bygga ut en statistisk analys, till exempel kan offentlig väderdata vara intressant för ett odlingsförsök. Den typen av data kan vara mer eller mindre lättillgänglig. Här tittar vi på några exempel på hur allmänt tillgänglig data kan hämtas och användas.

Kommunikation mellan datorer sker genom ett API (*Application Programming Interface*). Många organisationer som sprider data har ett öppet tillgängligt API som användare kan koppla upp sig till. Ofta finns R-paket som gör det enkelt att ange vilket data man är ute efter. Några exempel är

- `pxweb` - statistiska centralbyråns web-API, https://cran.r-project.org/web/packages/pxweb/vignettes/pxweb.html,
- `Eurostat` - europeiska statistikbyrån, https://ropengov.github.io/eurostat/articles/eurostat_tutorial.html,
- `Rspotify` - Spotifys API, https://github.com/tiagomendesdantas/Rspotify.

I följande exempel används paketet `osmdata` för att hämta data från OpenStreetMap, https://www.openstreetmap.org/.

```
#install.packages("osmdata")
library(osmdata)
dat_osm <- opq(bbox = 'Malmö') %>%
    add_osm_feature(key = 'admin_level', value = '10') %>%
    osmdata_sf()
```

```
dat_osm_pol <- dat_osm$osm_multipolygons

ggplot(dat_osm_pol, aes()) +
  geom_sf() +
  geom_sf_text(aes(label = name), size = 3)
```

Uppgift 8.21 (Malmös stadsdelar)**.** Vad kan ändras i exemplet ovan för att ta ut Lunds stadsdelar i stället för Malmös?

Ännu ett exempel. Denna gång Malmös restauranger efter typ.

```
dat_osm <- opq(bbox = 'Malmö') %>%
    add_osm_feature(key = 'amenity', value = 'restaurant') %>%
    osmdata_sf()

dat_osm_point <- dat_osm$osm_points %>%
  filter(cuisine %in% c("pizza", "sushi", "burger", "chinese", "indian", "vietnamese"))

ggplot() +
  geom_sf(data = dat_osm_pol) +
  geom_sf(data = dat_osm_point, aes(color = cuisine), size = 2)
```

Uppgift 8.22 (Offentlig konst)**.** Offentliga konstverk är ofta registrerade med `key = 'tourism'` och `value = 'artwork'`. Vad kan ändras i exemplet ovan för att ta ut offentliga konstverk i Malmö?

Det är inte alltid data finns tillgängligt genom en API. Mycket information finns publicerad som text eller tabeller på vanliga hemsidor. I såna fall kan man ofta ta hem data genom webbskrapning - att man med ett skript hämtar hem hemsidan, snarare än att själv läsa genom en webbläsare. I R kan det göras med paketet `rvest`. Ta som exempel den här tabellen över filmer i criterion-samlingen: https://www.criterion.com/shop/browse/list. För att läsa in den listan i R kan vi göra följande.

```
# install.packages("rvest")
library(rvest)

url <- "https://www.criterion.com/shop/browse/list"
html <- read_html(url)

dat_crit <- html %>%
  html_table()

dat_crit <- dat_crit[[1]] %>%
  select(-2) %>%
  filter(Director != "")
dat_crit
```

Uppgift 8.23 (Regissör)**.** Vilken regissör har flest filmer i criterion-samlingen? Använd datan från exemplet ovan och räkna antal filmer per regissör, t.ex. med `count()`.

Det finns flera paket som kan hämta data från Wikipedia, men det kan också göras med `rvest`. Här hämtas en tabell över mottagare av Nobelpriset i litteratur.

```
url <- "https://en.wikipedia.org/wiki/List_of_Nobel_laureates_in_Literature"
dat_nob <- url %>%
  read_html() %>%
  html_table()
dat_nob <- dat_nob[[1]]
```

Uppgift 8.24 (Skrivspråk)**.** Skapa ett stapeldiagram över antalet vinnare per språk (kolumnen `Language(s)`) genom att fylla i stycket nedan.

```
dat_agg <- dat_nob %>% count(`Language(s)`)

ggplot(dat_agg, aes(x = n, y = ___)) +
  geom_col()
```

Uppgift 8.25 (Valfri tabell)**.** Hitta en wikipedia-artikel med en tabell och försök hämta ner den till R genom att göra lämplig ändring i exemplet ovan.