Introduktion till R (2022-09-14)

Table of Contents

Allmänt om seminarierna och upplägget

- Alla har – eller kommer efter idag(?) – att ha följande installerat på sin dator:
  - R, helst 4.x
  - RStudio
  - ett antal bibliotek (libraries) som gör livet enklare
- All kod kommer att finnas dels i
  - Teams, och dels på
  - https://leijonhufvud.org/R-seminarier2022/
- Tillsammans med koden finns även dokument med text hör till, som
  - RMarkdown: källan till alltihop,
  - PDF (från RMarkdown) och
  - alla bilder (som PDF, skapade med LaTeX Beamer)
  - på leijonhufvud.org finns även alla dokument som web-sidor
- Till alla avsnitt finns det övningsexempel där ni använder det vi just gått igenom. Till dessa finns det generellt ett facit, men som vi alla vet finns det mer än ett sätt att lösa ett problem, se mina lösningar som exempel: de av er som är mer drivna och skolade programmerare kan säkert förbättra mina lösningar!

Vad är R?

Ett programeringsspråk/miljö optimerad för att statistisk bearbetning och produktion av grafer. För närvarande är det uppe i version 4.2.1

I en normal installation av R finns det den exempeldata, som både är bra för undervisning och för att kunna kommunicera med andra utan att behöva visa upp data som kan vara interna för ens organisation eller data som är skyddade av GDPR. Du kan se en lista med kommandot `data()`. I den här serien kommer vi till stor del att arbeta med data därifrån

GUI

Det finns ett antal GUI för R, om man är så lagd. Alternativen är bl.a.

- R Commander
- BlueSky (Mayo Clinic gick från SAS till Blue Sky Statistics/R nyligen)

RStudio: vad är det och varför

RStudio är en utveckingsmiljö för R, där man kan se sin kod, köra den i R, installera nya bibliotek och se producerade grafer. Särskilt på Windows är det en bra miljö att arbeta i, om man arbetar i en Unix-miljö kan man ibland föredra att helt enkelt ha t.ex. tvåterminalfönster öppna, samt den nuvarande koden i en texteditor. Då kan man köra skripten i ett fönster, och öppna ev grafer i det andra. Det beror helt enkelt på hur man vill arbeta, vilken typ av arbetsflöden man är van vid och trivs med.

Måste man skriva sin kod i RStudio?

Nej! Ett R-skript är bara text, och om man är van vid en annan editor så går det lika bra. Till exempel VIM, Notepad++ eller Emacs hanterar R-skript väl, i alla fall om man behärskar dem någotsånär. Själv är jag inkörd på VIM sedan 90-talet, så jag använder i princip aldrig RStudios inbyggda editor.

Word är däremot ett mycket suboptimalt (eller t.o.m. pessimalt) val för all kodning, men det antar jag att alla här visste sedan tidigare…

Ett R-skripts anatomi och sådant (egentligen 12/10)

Detta är iofs en smaksak, och saker som indentering, variabelnamn, osv är antingen upp till var och en eller en gemensam policy onnom gruppen. Jag är personligen inte alltför rabiat, men brukar hålla mig till följande

1. En kort kommentar om vad skriptet gör och varför, kanske med en TODO om det är saker som jag delegerat till framtids-Pär.
2. Ladda paket
3. Sätt namn på saker som skall användas, t.ex. namnet på en exportfil med dagens datum/vecka/månad. Det senare främst om det är något som skall köras regelbundet: jag försöker i det längsta behöva rutin-uppdatera skript som körs regelbundet.
4. Läs in alla datafiler som skall läsas in.
5. Data cleaning? Jag brukar göra den rutinmässiga städningen här

- Datum? Vilket format är de?
- Är allt i rätt typ (numeric, integer, osv)?
- Skapa nya kolumer med t.ex. vecka, år…
- Skall man t.ex. rensa bort ifullständiga data?

6. Kommentera alla avsnitt så att (1) framtids-Pär ved vad jag sysslar med, och (2) jag lättare kan söka efter en sektion om det blir stort.
7. Skriv gärna en TODO i början för jag som borde fixas

Namn på data frames, variabler och funktioner.

Grundregeln är att alla namn måste börja med en bokstav (stor eller liten) men efter det kan det innehålla siffor, punkt och "underscore". Den får inte heller slute med ett skiljetecken.

Vad gäller namn finns det många olika traditioner. Själv brukar jag köra "_" eller "." för att separera snarare än CamelCase, mest för att jag försöker köra "STORA" för saker som gäller hela skriptet och "gemena" för saker som bara är tänkta att användas just här. Också för att enkelt hantera saker som DATUM.provtagning, DATUM.ankomssttid, osv.

Eftersom jag mest skriver kod som skall användas av mig, och återanvändas av mig, så tenderar jag att snarare kommentera ut saker än att städa bort saker som funkat men jag inte vill ha just nu. Inte blir det vackert, men ibland användbart.

Hjälp!

Ni som är vana vid Unix i någon av alla dess former har säkert vanan att skriva `man <kommando>` för att se vad det gör och vilka möjligheter det ger. I R finns det `?<kommando>`. Testa t.ex. `?mean` eller `?hist`. Om du tycker det är för kort att skriva "?" kan du istället skriva `help(median)`för att få samma hjälp.

Om du inte vet vilket kommando du letar efter finns "??" som söker efter hjälpsidor med den texten. Testa t.ex,`??barplot`. Den längre versionen är `help.search()`.

Tidyverse och andra Bra-Att-Ha bibliotek

Många föredrar att arbeta i R utan stora tillägg, men många andra anser att en miljö med lite fler och smidigare verktyg är att föredra. Eftersom jag inte är purist så använder jag relativt ofta tidyverse, vilket också gäller för den här seminarieserien. Tidy är inte helt okontroversiellt, då det gör vissa saker enklare, och andra saker svårare. Det har bl.a. hävdats att användningen av Tidy i grundläggande R-utbildning försvårar fördjupning. Personligen har jag ingen stark åsikt i den frågan.

Tidyverse: en samling R-paket anvsedda för "Data Science"

Grunderna är

- **ggplot2**: ett kraftfullaren verkyg för grafik
- **dplyr**: data manipulering
- **tidyr**: för att städa i data
- **readr**: för att enklare importera data i olika format:

  - **read_csv()** comma-separated values
    - read_csv2(): decimalkomma, med ; som avgränsare
  - **read_tsv()** tab-separated values
  - **read_delim()** andra "avgränsade format
  - **read_fwf()** fixed width filer
  - **read_table()** tabeller med "whitespace" som avgränsare
  - **read_log()** logfiler

- **purr**: för funktioner och vektorer
- **tibble**: en modernisering av data_frame
- **stringr**: för att arbeta med strängar
- **forecats**: arbeta med faktorer

Därtill finns i tidyverse t.ex.

lubridate

Lubridate förenklar arbete med datum och tider

Andra bra paket

ggthemes

Lite fler teman för grafer

gridExtra

Ställa samman flera grafer till en "multigraf"

ISOweek

För veckonummer enligt ISO 8601

Installera och ladda in biblotek och paket

```
# Installera
> install.packages("tidyverse")
# Ladda in
> library(tidyverse)
```

Lite grunder

På Rs kommandorad har du en prompt: **>**. När jag skriver en rad som börjar med den är meningen att man ger kommandot som följer prompten.

Vill du veta vad ett kommando gör kör du

```
> ?library
```

så får du en kort sammanfattning över vad kommandot gör och vilka argument det kan ta.

Lite beräkningar

```
> 1+1
[1] 2
> 42-2
[1] 40
> 3*12
[1] 36
> 2^3 # 2 upphöjt i 3
[1] 8
```

Många av de statistiska opereationerna finns givetvis inbygda. Om vi t.ex. arbetar med det inbygda data-setet "Nile" kan vi

```
> mean(Nile)
[1] 919.35
> median(Nile)
> sd(Nile)
[1] 169.2275
> min(Nile)
[1] 456
> max(Nile)
[1] 1370
> summary(Nile)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  456.0   798.5   893.5   919.4  1032.5  1370.0
```

Vad finns i ett objekt?

Vill vi veta lite om vad som finns i Nile kan vi enkelt få veta det:

```
> Nile
Time Series:
Start = 1871
End = 1970
Frequency = 1
  [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
1020  960
 [17] 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100  774  840
874  694
 [33]  940  833  701  916  692 1020 1050  969  831  726  456  824  702 1120
1100  832
 [49]  764  821  768  845  864  862  698  845  744  796 1040  759  781  865
845  944
 [65]  984  897  822 1010  771  676  649  846  812  742  801 1040  860  874
848  890
 [81]  744  749  838 1050  918  986  797  923  975  815 1020  906  901 1170
912  746
 [97]  919  718  714  740
```

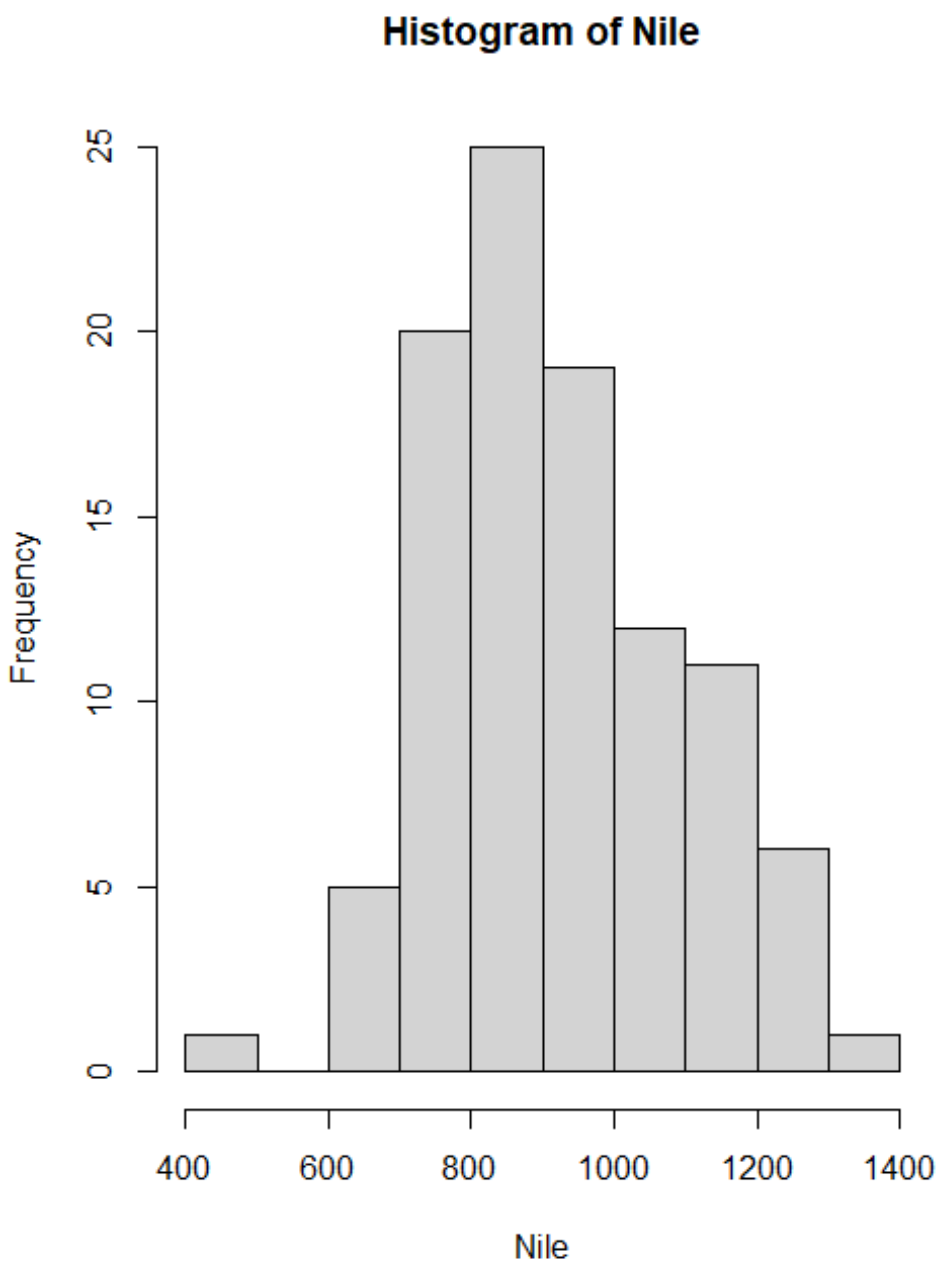
Notera att det finns ett radnummer, som anger vilket värde i ordningen som är först på den raden. Vi ser att Nile är en vektor.

Om det är många värden i ett objekt kan det vara mindre användbart att bara skiva ut alla värdena (R slutar dock efter 999).

Enklare grafer: ett enkelt histogram

För grafer finns det i "base-R" enklara versioner, vi kommer till mer "publikationsvänliga" grafer senare i seminarieserien. Men ett histogram ger en bra visuell översikt över hur fördelningen ser ut.

```
> hist(Nile)
```



Enskilda värden i ett objekt

Vi kan ta ut ett eller flera värden ur Nile:

```
> Nile[2]    # det andra värdet i Nile
[1] 1160
> Nile[c(1,3,5)]    # 1.a, 3:e och 5:e värdet
[1] 1120  963 1160
```

I det andra exemplet tar vi flera med c ("concatenate") som bygger en ny vektor, med bara dessa tre värden. Vill vi ha en serie kan vi t.ex. ange värden fr.o.m. 3 t.o.m. 5 med `3:5`.

```
> Nile[3:5]
[1]  963 1210 1160
```

Eftersom vi kan stapla funktioner på varandra i R kan vi nu t.ex. fr fram medelvärdet att de första 50 årens Nil-data samt avrunda till 1 decimal:

```
> mean(Nile[1:50])
[1] 984.32
> round(mean(Nile[1:50]), 1)
[1] 984.3
```

Notera att som med alla "smarta" sätt att skriva kod så kan det snabbt gå rejält överstyr, så om du börjar ha mer än 2-3 lager föreslår jag att du överväger att dela upp det på flera steg.

Om vi vill arbeta mer med just de första 50 åren är det enklare att spara dem som en egen vektor:

```
> nilen_150 <- Nile[1:50]
> nilen_150
 [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100  774
840  874  694  940  833  701  916
[37]  692 1020 1050  969  831  726  456  824  702 1120 1100  832  764  821
> mean(nilen_150)
[1] 984.32
```

Ta ut alla värden som matchar ett krav

Låt oss säga att vi vill ha alla värden i Nile som är större än 1200? Hur går vi vill väga med det, och kan vi enkelt få reda på hur många de är? Det kan vi:

```
> antal_över_1200 <- sum(Nile>1200)
> antal_över-1200
[1] 7
```

Hur gick det till?

Låt oss skapa en vektor x med tre värden

```
x <- <- c(5,12,13)
> x
[1]  5 12 13
```

Om vi skriver `x > 8` så återanvänder R "8" för varje post i vektorn x, och jämföär dem:

```
> x > 8
[1] FALSE  TRUE  TRUE
```

Som vi är vana vid från andra programmeringsmiljöer är "TRUE" 1 och "FALSE" 0, så vektorn FALSE,TRUE,TRUE är det samma som 0,1,1. Och med `sum` kan vi summera 0+1+1:

```
> sum(x > 8)
[1] 2
```

Kan vi få ut vilka värden som är större än 1200 i Nile?

```
> which(Nile > 1200)
[1]  4  8  9 22 24 25 26
```

Det är alltså värdena 4, 8, 9, 22 osv som är större än 1200. och detta ger oss ett till sätt att räkna antalen:

```
> nilen1200 <- which(Nile > 1200)
> nilen1200
[1]  4  8  9 22 24 25 26
> length(nilen1200)
[1] 7
```

Eller så kan vi ta en genväg, och ber R att visa oss alla värden i Nile som matchar kravet "> 1200":

```
> Nile[Nile > 1200]
[1] 1210 1230 1370 1210 1250 1260 1220
```

Tabeller: Data Frame

En vektor är en serie värden, men i vårt verkliga liv har vi oftare en tabell med ett antal olika kolumner. Vi kan jobba med riktiga data nu, alla aCovid-tester på Laboratoriemedicin under 2021 (verkliga data, men utan personkopplad information).

Vi tar det i tre steg:

1. Läs in tabellen

För att läsa in en CSV-tabell med semikolon, UTF8 skriver vi `read_csv2 <- "sökväg/till/tabellen.csv", fileEncoding="UTF-8-BOM")`

1. read.csv2 är en genväg till det som egentligen är en semikolon separerad fil.
2. fileEncoding = "UTF-8-BOM" säger att det är UTF samt har ett Byte Order Mark. Om du behöver BOM eller inte beror på ursprunget till din fil: för filer sparade som "CSV" i Excel verkar det behövas.

2. Vi får en överblick

Med `str()`får vi en överblick vilka kolumner som finns, och vilka typer av data som finns i vardera

3. Titta på de första 5 raderna

För att se de förstas fem raderana på vår tabell använder vi `head()`:

```
> acovid <- read.csv2("rjh-acovid.csv", fileEncoding = "UTF-8-BOM")
> str(acovid)
>'data.frame':  2044 obs. of  9 variables:
 $ Kön                : chr  "K" "K" "K" "M" ...
 $ Ålder..År.         : chr  "6,73" "3,32" "5,8" "47,48" ...
 $ Beställare         : chr  "ÖSJ   AKUT  BARN" "ÖSJ   AKUT  BARN" "ÖSJ
AKUT  BARN" "ÖSJ   AKUT  BARN" ...
 $ Analys             : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid   : chr  "2021-12-17 16:20" "2021-08-12 11:49" "2021-06-
03 19:28" "2021-03-26 17:54" ...
 $ Tekniskt.godkännande: chr  "2021-12-17 18:14" "2021-08-12 13:58" "2021-06-
03 21:55" "2021-03-26 20:14" ...
 $ Ankomsttid         : chr  "2021-12-17 17:27" "2021-08-12 12:00" "2021-06-
03 19:52" "2021-03-26 18:00" ...
 $ Ledtid             : chr  "00:47:00" "01:58:00" "02:03:00" "02:14:00" ...
 $ Mätvärde           : num  0.07 0.08 0.07 0.06 0.06 0.06 0.06 9.32 0.06
0.06 ...
> head(acovid)
  Kön Ålder..År.       Beställare           Analys Registreringstid
Tekniskt.godkännande       Ankomsttid   Ledtid Mätvärde
1   K       6,73 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-12-17 16:20
2021-12-17 18:14 2021-12-17 17:27 00:47:00     0.07
2   K       3,32 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-08-12 11:49
2021-08-12 13:58 2021-08-12 12:00 01:58:00     0.08
3   K        5,8 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-06-03 19:28
2021-06-03 21:55 2021-06-03 19:52 02:03:00     0.07
4   M      47,48 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-03-26 17:54
2021-03-26 20:14 2021-03-26 18:00 02:14:00     0.06
5   K      17,44 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-03-05 23:49
2021-03-06 01:18 2021-03-05 23:54 01:24:00     0.06
6   K      14,85 ÖSJ   AKUT  BARN S-anti-SARS-CoV-2 2021-02-24 06:48
2021-02-24 14:18 2021-02-24 08:36 05:42:00     0.06
```

(Jag valde det något långa namnet “acovid” för att det är enklare att komma ihåg vad det är för data, jag hade kunnat välja ett kortare namn, men då förlitat mig på mitt minne).

Arbeta med data i en kolumn

Vill vi bara t.ex. ta medianvärdet på mätvärdena kan vi skriva

```
> median(acovid$Mätvärde)
[1] NA
> median(acovid$Mätvärde, na.rm=TRUE)
[1] 0.1
```

Varför fick vi "NA" första gången? Eftersom det fanns en blankrad i början fick vektorn acov$mätvärden en punkt "NA" i början, och då anser R, kanske med rätta, att det inte går att ta medianvärdet av en blankrad. Med `na.rm=TRUE` säger vi till "mean" att ignorera de värdena som saknar värden. En notis om bra vanor: många skriver bara T eller F för TRUE eller FALSE, vilket sparar några tangenttryckningar. Det kommer att fungera bra tills du eller någon annan skrivit

```
> T <- 0
> F <- 1
```

Med potentiellt intressanta konsekvenser. TRUE och FALSE är däremot reserverade, och du bör därmed använda dem.

Gör ett utdrag ut acovid

Vilket värde finns i kolumn 3, rad 14?

```
> acovid[14,3]
[1] "ÖSJ   AKUT  BARN"
```

Precis som vi såg tidigare kan vi ta flera rader eller kolumner. Till exempel kolumn 1, 3 och 9, raderna 2-22:

```
> acovid[2:22,c(1,3,9)]
   Kön       Beställare Mätvärde
2    M ÖSJ   INTENSPIVA    19.20
3    M  ÖSJ   MEDSERLAB     0.08
4    K  ÖSJ   MEDSERLAB     0.08
5    K  ÖSJ   MEDSERLAB   317.00
6    K  ÖSJ   MEDSERLAB    67.00
7    M   ÖSJ   INF   30    23.90
8    M ÖSJ   INTENSPIVA    12.80
9    K  ÖSJ   MEDSERLAB     0.10
10   K ÖSJ   REMONTMOTT   138.00
11   M STRÖM PRIMV BACK     0.09
12   M ÖSTER PRIMV FRÖS     0.08
13   M ÖSJ   INTENSPIVA    19.10
14   K ÖSJ   AKUT  BARN     0.07
15   M HÄRJE PRIMV SVEG     0.08
16   K ÖSTER PRIMV BRUN     0.09
17   M   ÖSJ   INF   30     0.07
18   K STRÖM PRIMV NNST     0.08
19   K  ÖSJ   MEDSERLAB    38.00
20   M  ÖSJ   BARN  108    10.00
21   M ÖSTER PRIMV FRÖS     0.07
22   K STRÖM PRIMV NNST   154.00
```

Om vi vill ha alla värden i t.ex. kolumn 4 skriver vi hara `acovid[,4]`.

Kopiera ut alla rader som matchar ett visst kriterium

Som vi ser finns kön och mätvärde (analysresultatet) med, hur gör vi om vi vill ha separata tabeller för män och kvinnor, och kanske även för kvinnor med mätvärde över 20?

```
> acovid.kvinnor <- acovid[ acovid$Kön == "K", ]
```

Vad vi säger är

- Alla rader där kolumnen "Kön" har värdet "K"
- Alla kolumner

För kvinnorna som är positiva för Covid19-antikroppar (mätvärde > 20) kan vi använda tidyverse och kommandot *filter*:

```
> library(tidyverse)
> acovid.k.positiva <- filter(acovid, Kön == "K" & Mätvärde > 20 )
```

Vill vi hålla oss till "base R" går det också:

```
> acovid.k.positiva <- acovid[ (acovid$Kön == "K" & acovid$Mätvärde > 20), ]
```

Vi kan enkelt kolla att vi fick med det vi ville:

```
> nrow(acovid.k.positiva)
[1] 160
> unique(acovid.k.positiva$Kön)
[1] "K"
> summary(acovid.k.positiva$Mätvärde)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  20.43   41.27   74.00   88.70  129.25  317.00
```

Kopiera en tabell (eller delar av den)

Även om R inte arbetar med originalfilen (t.ex. den CSV-fil du läste in), så kan det ibland vara bra att att arbeta med en kopia av en fil, helt enkelt för att man vill göra ändringar som man *kanske* inte vill behålla.

```
> acovid.arb <- acovid
```

Om vi vill skapa en tom kopia av en data frame?

```
> acovid.tom <- acovid[0,]
> str(acovid.tom)
'data.frame':   0 obs. of  9 variables:
 $ Kön                 : chr
 $ Ålder..År.          : chr
 $ Beställare          : chr
 $ Analys              : chr
 $ Registreringstid    : chr
 $ Tekniskt.godkännande: chr
 $ Ankomsttid          : chr
```

```
 $ Ledtid              : chr
 $ Mätvärde            : num
```

Här ser vi tomma kolumner (vi tog ju bara med 0 rader…), men med samma namn och de har t.o.m. ärvt typen av den ursprungliga. Det här är praktiskt om vi t.ex. vill ha ett skript som plockar ut vissa rader och lägger till dem till en resultatfil (t.ex. hitta alla beställningar – LID – som har Analys1, men inte Analys2, och ge mig en lista på dem).

Lägga tll kolumner i en data frame

Om du tittar i acovid ser du ett antal datum-tid strängar (“2021-12-17 16:20”), om du t.ex. vill ha bara datumet (eller t.ex. år-vecka, men det tar vi nästa gång) för din statistik är det enkelt att skapa nya kolumner med de data. Och när R känner igen datum som datum kan det arbeta med dem, t.ex. i `summary()`.

```
> acovid.arb$Datum.A <- as.Date(acovid.arb$Ankomsttid)
> str(acovid.arb)
'data.frame':   2044 obs. of  10 variables:
 $ Kön                 : chr  "K" "K" "K" "M" ...
 $ Ålder..År.          : chr  "6,73" "3,32" "5,8" "47,48" ...
 $ Beställare          : chr  "ÖSJ   AKUT  BARN" "ÖSJ   AKUT  BARN" "ÖSJ
AKUT  BARN" "ÖSJ   AKUT  BARN" ...
 $ Analys              : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid    : chr  "2021-12-17 16:20" "2021-08-12 11:49" "2021-06-
03 19:28" "2021-03-26 17:54" ...
 $ Tekniskt.godkännande: chr  "2021-12-17 18:14" "2021-08-12 13:58" "2021-06-
03 21:55" "2021-03-26 20:14" ...
 $ Ankomsttid          : chr  "2021-12-17 17:27" "2021-08-12 12:00" "2021-06-
03 19:52" "2021-03-26 18:00" ...
 $ Ledtid              : chr  "00:47:00" "01:58:00" "02:03:00" "02:14:00" ...
 $ Mätvärde            : num  0.07 0.08 0.07 0.06 0.06 0.06 0.06 9.32 0.06
0.06 ...
 $ Datum.A             : Date, format: "2021-12-17" "2021-08-12" "2021-06-03"
"2021-03-26" ...
> summary(acovid.arb$Datum.A)
        Min.      1st Qu.       Median         Mean      3rd Qu.         Max.
NA's
"2020-12-31" "2021-01-20" "2021-02-10" "2021-02-23" "2021-03-02" "2021-12-30"
"142"
```

Arbetskatalogen

Vilken katalog du arbetar i påverkar t.ex. var R letar efter filer du öppnar, var sparade filer (grafer, exporterade filer, osv) hamnar och var din *.Rhistory* (se nedan) sparas. Du kan kolla vilken den är med

```
> getwd()
```

Ibland kan det vara bra att kunna byta arbetskatalog, och givetvis kan R det med

```
> setwd("C:/Min/nya/arbetakatalog")
```

Om du vill använda dig av en relativ sökväg går det också bra:

```
> setwd("../arbetakatalog2")
```

Notera att R inte använder sig av bakåtslaskar även på en Windows-dator: vill du använda dig av dem måste du "escapa" dem:

```
> setwd("C:\\Min/nya\\arbetakatalog")
```

Det här t.ex. användbart om du skall skapa ett större antal grafer och vill spara dem i en separat underkatalog (R kan både testa om en katalog finns och skapa den: prakiskt i skript, se `?dir.create` och `?dir.exists`)

Stäng ned R, men spara din historia

```
> q() # eller
> quit()
```

Allt det du skrivit i terminalen finns nu sparat i filen **.Rhistory**. Den är användbar för att när du startar R i en katalog läses .Rhistory in från den katalogen, och du kan återanvända dina kommandon (pil upp/ned), och för att du kan använda den som en genväg för att skapa ett skript från något du gjorde och vill kunna göra igen.

Vad är RMarkdown, förresten?

Markdown är svaret på frågan "Word, dess anfäder och dess kusiner är komplicerade och hamnar ibland ivägen för själva skrivandet. Vad händer om man går åt andra hållet och förenklar det så myvcket som möjligt?". Resultatet var ett enkelt system med *markup* i en textfil, som antingen kan förstås utan större ansträngning av en mer eller mindre naiv läsare, eller vid behov formaterat till ett "färdigt" dokument, vare sig det är en websida, en PDF, en presentation, eller ett Word-dokument.

RMarkdown är en dialekt av markdown som har tillägg för att hantera R-kod, antingen genom att visa koden eller genom att köra den och visa resultatet (e.g. en graf eller en siffra).

Resurser

Det finns ett antal externa källor till hjälp, som vanligt är oftast ett svar en sökning bort (skälet till att detta ligger som en websida…).

Webben, mailinglistor, osv

- The R Project for Statistical Computing
- Rstudio community
- Big Book of R
- The R Graph Gallery

- stackoverflow med alla *dess* för och nackdelar
- R-help mailinglista
- fasteR Ett paket för att lära sig R. Jag har stulit en del från den för de här seminarierna.
- Jag har en del på Pärs Backup Brain, som mest existerar för att förenkla när jag letar efter en lösning på ett problem jag redan löst. Men alltid hjälper det någon.

Böcker

- Kieran Hielys "Data Visualization" är suverän om man vill skapa bra grafer.
- Hadley Wickam har skrivit ett antal böcker, t.ex. "R for Data Science" som är mycket användbara. Den finns både som bok ifall man vill hålla papper i handen och på webben ifall man nöjer sig med det.
- Mark Gardening "Statistics for Ecologists Using R and Excel". Ganska bra into till att analysera data initialt i Excel och sedan fördjupande i R: ett arbetsflöde som nog är ganska vanligt.

Övningar

Facit

1. Ta fram listan på vilka R-packet som är installerade på din dator
2. Om tidyverse inte är installerat gör det.
3. Ladda in data-setet "ToothGrowth". Det är tandtillväxten hos marsvin som fått vitamin C eller apelsinjuice.
4. Titta på struktur:
   - Vilka kolumner finns det?
   - Vilka typer av data finns i var och en?
5. Hur många rader finns det i ToothGrowth?
6. Ta fram en sammanfattning av ToothGrowth: vilket är det lägsta respektive högsta värdet på "len"?
7. Ta ut den 7:e raden i ToothGrowth
8. Vilken var den genomsnittsliga tandtillväxten för marsvin som fick apelsinjuice respektive vitamin C?
9. Till nästa gång ladda in några data från det du jobbar med, eller valfri tabell från SCB, och se vad du har i dina data.