Data cleaning med R (2022-09-28)

Table of Contents

Inledning

Data är oftast fulla av saker som inte borde vara där

- Datum: vilket format har de?
- Excels berömda datum-problem
- Gömmer det sig en felkod där du tror det bara är mätvärden?
- har du strängar, heltal eller decimaltal?
- När du importerar, vad är ett "NA": blankt, -, mer?
- Finns det dubbletter?

Importera data

Vi gick igenom detta förra gången, men tar en kort repetition

CSV:

```
> tabell1 <- read.csv2("./relativ/sökväg/till/filen.csv", fileEncoding="UTF-8-BOM")
```

Excel, med paketet xlsx: om jag börjar med library("xlsx") behöver jag inte ha med "xlsx::" före kommandot

```
> tabell2 <- xlsx::read.xlsx2("min-exelfil.xlsx", sheetIndex = 2, startRow = 2)
```

När vi använder read.xlsx2 måste vi ange vilken flik vi vill läsa in. I read.xlsx2 finns även en bra finess: många Excel-tabeller har några rader med rubriker överst: om du använder "startRow = 2" rensar du automatiskt bort dessa rader.

Sortera bort det som inte borde vara med

GDPR-städa

När vi har information som skulle kunna kopplas till person bör man rensa bort detta så fort man kan. Den personidentifierande informationen är t.ex.

- Personnummer
- Namn

- Räcker det med att ta bort namn och personnummer?
  - Hur många positiva aCovid tester på kvinnor i åldern 55-64 togs i Myrviken 2021-02-29?
  - Hur många blödarsjuka i Hallen?
  - Hur många män över 55 på Laboratoriemedicin svarar på personalenkäten?
- Minska risken att du röjer en person tills ingen möjlig sorteringsgrupp är så liten att det är enkelt att efterforska en individ.
- SCBs dataskyddspolicy

Skapa ett index som är unikt för varje rad

Om man inte vill ha kvar koppling till person är det enklaste att helt enkelt radera den kolumnen. Men ibland vill man ha möjlighet att komma åt den identifierande informationen: hur gör man då?

Mitt förslag är:

1. Sortera på ett sätt som inte gör det enkelt att koppla plats i ordningen till person
2. Skapa ett index för varje rad
3. Skapa en ny data.frame med identifierande information + index
4. Spara en CSV-fil med t.ex. pnr+index på en säker plats
5. Radera tabellen med nyckeln
6. Radera kolumnen med den identifierande informationen

För indexet brukar jag först sortera mina data för att göra det svårt att koppla index till person. I det här fallet sorterar jag på mätvärde, beställare, kön och ankomsttid.

```
> acov <- acov[order(acov$Mätvärde, acov$Beställare, acov$Kön,
acov$Ankomsttid), ]
> acov$INDEX <- as.character(1:nrow(acov))
> NYCKEL <-  data.frame(acov$INDEX, acov$LIDN)
> write.csv2(NYCKEL, file="NYCKEL_till_mina_data.csv")
> rm(NYCKEL)
> acov <- within(acov, rm(LIDN))
> str(acov)
'data.frame':   17937 obs. of  12 variables:
 $ X                  : int  17400 17401 1392 1337 1133 1419 831 1131 1128
1389 ...
 $ Kön                : chr  "K" "K" "K" "K" ...
 $ Ålder..År.         : chr  "32,03" "61,03" "49,66" "43,45" ...
 $ Beställare         : Factor w/ 3 levels "VC.GLESBYGD",..: 3 3 1 1 1 1 1 1
1 1 ...
 $ Analys             : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid   : chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 08:22" "2021-02-18 13:46" ...
 $ Tekniskt.godkännande: chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 18:00" "2021-02-19 18:05" ...
```

```
 $ Ankomsttid          : chr  "-" "-" "2021-02-17 15:45" "2021-02-19 15:40"
...
 $ Ledtid              : chr  "-" "-" "2:15:00" "2:25:00" ...
 $ Mätvärde            : num  0 0 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 ...
 $ Antal               : int  1 1 1 1 1 1 1 1 1 1 ...
 $ INDEX               : int  1 2 3 4 5 6 7 8 9 10 ...
```

Nu har vi blivit av med LIDN som eventuellt kan bedömmas som GDPR-känsligt, och istället har vi ett index-nummer, och vi har valt att spara en nyckel för att vid behov kunna återskapa kopplingen. Är man säker på att det inte kommer att behövas finns det givetvis inget skäl att spara en sådan nyckel.

Förresten: du sparar väl inte din nyckel på en delad share som andra kan komma åt lite hur som helst? Själv använder jag VeraCrypt, men det finns givetvis andra alternativ.

Gruppera för att skydda personkopplad information

Eftersom jag delar ut detta data-set har jag redan grupperat vårdcentraler och sjukhusets avdelningar. bakgrunden till detta är att det kan vara mycket få patienter som provtar sig för en viss sjukdom på en mindre vårdcentral en viss dag, särskilt om man lägger ett kön och en ålder. Det är därför bra att så långt som möjligt gruppera för att undvika små grupper.

Det finns två vägar att gå för att gruppera:

1. Gör det utanför R, t.ex. i en texteditor (t.ex. Notepad++) eller i Excel. Särskilt en bra text-editor är ett oerhört kraftfullt verktyg för att mönstersöka och ersätta, men även Excel klarar med lätthet av detta.
2. Gör det i R. I princip ersätter man flera olika enhetsnamn med ett enda, se nedan under “Gruppera” för hur man gör.

Typen av data: skillnaden mellan int, chr, num, date…

Olika typer av data: “42” är inte samma sak som 42

I R finns det ett antal olika typer som data kan ha, och oftast känner R av vad som finns och hanterar det korrekt. I motsats till Excel skriver R inte om värden den tolkar, så fel går att fixa. Värt att veta är dock att om det i en kolumn finns ett antal numeriska värden, och en enda text, då kommer hela kolumnen att klassas som text.

Numeric

Ett tal, med olika antal decimaler, den kanske vanligaste typen av data vi arbetar med. Till exempel hur långa en grupp människor är, analyssvaret på en laboratorieanalys, osv.

Integer

Ett heltal: 1, 2, 42, 987635 osv

Om det bara finns siffror och inga decimaler antar R att det är av typen integer. Exempel är antal patienter som fått en viss diagnos.

Character

Text, skrivs med citattecken: "Foo", "42,1" och "42" är alla text. Om R får en vektor med ett enda värde som är text kommer hela vektorn att klassas som text.

Factor

En speciell typ av text (character), men det finns bara en begränsad mängd möjliga värden; vårdcentraler, månader, osv är typiskt av klassen factor. Men kommandot `levels(faktorns.namn)` kan man se vilka värden den kan ha. Normalt sorteras de alfabetiskt, men man kan styra det manuellt:

```
> factor(Namn, levels = c("Kalle", "Ada"))
```

Nu kommer Kalle att komma före Ada.

Logical

Om det är sant eller falskt (TRUE, FALSE) är det logical. Som vi lärde oss förra gången är det ettor och nollor som lagras.

Byta mellan klasser

Man kan byta mellan klasser med ett kommando i formen `as.factor`.

```
> acov <- read.csv2("acov_GDPR.csv", fileEncoding = "UTF-8-BOM")
> str(acov)
'data.frame':   17937 obs. of  12 variables:
 $ X                  : int  17400 17401 1392 1337 1133 1419 831 1131 1128
1389 ...
 $ Kön                : chr  "K" "K" "K" "K" ...
 $ Ålder..År.         : chr  "32,03" "61,03" "49,66" "43,45" ...
 $ Beställare         : chr  "ÖSJ" "ÖSJ" "VC.GLESBYGD" "VC.GLESBYGD" ...
 $ Analys             : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid   : chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 08:22" "2021-02-18 13:46" ...
 $ Tekniskt.godkännande: chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 18:00" "2021-02-19 18:05" ...
 $ Ankomsttid         : chr  "-" "-" "2021-02-17 15:45" "2021-02-19 15:40"
...
 $ Ledtid             : chr  "-" "-" "2:15:00" "2:25:00" ...
 $ Mätvärde           : num  0 0 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 ...
 $ Antal              : int  1 1 1 1 1 1 1 1 1 1 ...
 $ INDEX              : int  1 2 3 4 5 6 7 8 9 10 ...
> acov$Beställare <- as.factor(acov$Beställare)
> str(acov$Beställare)
 Factor w/ 3 levels "VC.GLESBYGD",..: 3 3 1 1 1 1 1 1 1 1 ...
```

```
> levels(acov$Beställare)
[1] "VC.GLESBYGD"  "VC.Östersund" "ÖSJ"
```

På samma sätt finns: * as.numeric * as.character * as.integer * as.logical

Datum

För att jobba med datum använder jag så fort det blir komplicerat packetet `lubridate`, men för enklare saker behövs det inte:

```
acov$Datum <- as.Date(acov$Ankomsttid)
acov$År <- strftime(acov$Datum,  format = "%Y")
```

Med lubridate kan du få ut t.ex. veckodagar, tiden mellan två datum-tidsträngar, arbeta med tidszoner, osv. Om du vill ha veckor rekommenderar jag paketet ISOweek: viktigt att veta är att det finns två standarder för när en vecka börjar, och det finns därmed en risk för att datum hamnar i "fel" vecka (ja, jag har blivit biten av detta).

Det borde vara en siffra men är "42,42"

Ibland klarar inte read.csv2 av att tolka det vi ser som en siffra som en siffra. Till exempel så har vi åldern som *Ächaracter*. Hur löser vi det?

```
 str(acov)
'data.frame':   17937 obs. of  12 variables:
 $ X                  : int  17400 17401 1392 1337 1133 1419 831 1131 1128
1389 ...
 $ Kön                : chr  "K" "K" "K" "K" ...
 $ Ålder..År.         : chr  "32,03" "61,03" "49,66" "43,45" ...
 $ Beställare         : Factor w/ 3 levels "VC.GLESBYGD",..: 3 3 1 1 1 1 1 1
1 1 ...
 $ Analys             : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid   : chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 08:22" "2021-02-18 13:46" ...
 $ Tekniskt.godkännande: chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 18:00" "2021-02-19 18:05" ...
 $ Ankomsttid         : chr  "-" "-" "2021-02-17 15:45" "2021-02-19 15:40"
...
 $ Ledtid             : chr  "-" "-" "2:15:00" "2:25:00" ...
 $ Mätvärde           : num  0 0 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 ...
 $ Antal              : int  1 1 1 1 1 1 1 1 1 1 ...
 $ INDEX              : int  1 2 3 4 5 6 7 8 9 10 ...
```

Åldern är av type **chr**, och har kvar sitt decimalkomma. Hur fixar vi det? Första steget är att byta ut kommat mot en decimalpunkt. För det använder vi kommandot `gsub()`, som har den grundläggande syntaxen `gsub(leta_efter, ersätt_med, leta_i, ignore.case=FALSE)`. Vi kan ange en sträng att leta i, en vektor eller en kolumn i en data

frame. Vill vi kan vi t.o.m. använda Perls regex-system, vilket är praktiskt om man vill göra något elegant (alternativt förvirra sig själv).

```
## Byt ut  komma mot punkt
> acov$Ålder <- gsub(",",".", acov$Ålder..År.)
```

Som vi ser är det fortfarande en sträng med text, nu gör vi om den till nummer:

```
> str(acov$Ålder)
 chr [1:17937] NA "92.27" "76.17" "64.07" "81.89" "39.50" "75.12" "82.47"
"56.33" "53.16" "55.85" "60.75" "62.03" "77.28" "22.81" "24.90" "69.76"
"69.87" "69.39" "-" "-" ...
>
># Gör om Ålder till numeriska värden
> acov$Ålder <- as.numeric(acov$Ålder)
Warning message:
NAs introduced by coercion
> str(acov$Ålder)
 num [1:17937] NA 92.3 76.2 64.1 81.9 ...
```

Vi hade givetvis kunnat göra det i ett steg, till priset av att det blir lite svårare att följa

```
> acov$Ålder <- as.numeric(gsub(",",".", acov$Ålder..År.))
Warning message:
NAs introduced by coercion
> str(acov$Ålder)
 num [1:17937] NA 92.3 76.2 64.1 81.9 ...
```

När jag gör om en kolumn brukar jag skapa en ny med det nya värdet i, men man kan skriva tillbaka till samma kolumn: risken är att man gör fel och får börja om från början.

Ta ut det som är relevant

- Vill du bara ha män, positiva provsvar, svar på en fredag? Detta kan fixas både i "base-R" och i tidyverse

Alla kvinnor, i base-R

```
> nrow(acov[ acov$Kön=="K", ])
[1] 11391
> unique((acov[ acov$Kön=="K", ])$Kön)
[1] "K" NA

> (acov[ (acov$Kön=="K" & acov$Mätvärde > 20), ])
 # Många rader!
summary(acov[ (acov$Kön=="K" & acov$Mätvärde > 20), ])
       X              Kön             Ålder..År.         Beställare
Analys          Registreringstid   Tekniskt.godkännande  Ankomsttid
Ledtid
 Min.   :    5   Length:771         Length:771         Length:771
Length:771         Length:771         Length:771           Length:771
```

```
Length:771
 1st Qu.: 1924   Class :character   Class :character   Class :character
Class :character   Class :character   Class :character     Class :character
Class :character
 Median : 6554   Mode  :character   Mode  :character   Mode  :character
Mode  :character   Mode  :character   Mode  :character     Mode  :character
Mode  :character
 Mean   : 7245
 3rd Qu.:11892
 Max.   :17919
 NA's   :1
    Mätvärde          Antal       INDEX
 Min.   : 20.05   Min.   :1   Min.   :16674
 1st Qu.: 41.02   1st Qu.:1   1st Qu.:16991
 Median : 76.23   Median :1   Median :17320
 Mean   : 88.57   Mean   :1   Mean   :17308
 3rd Qu.:126.28   3rd Qu.:1   3rd Qu.:17627
 Max.   :756.00   Max.   :1   Max.   :17936
 NA's   :1        NA's   :1   NA's   :1
```

Med dplyr från tidyverse?

Givetvis kan man göra samma sak även med dplyr, den delen av tidyverse som handlar om data-städning

```
> library(dplyr)
> filter(acov, Kön == "M" & Mätvärde > 20)
> acov.män.över20 <- filter(acov, Kön == "M" & Mätvärde > 20)
```

Ta bort rader med NA i en viss kolumn?

Om vi vill ta bort alla rader utan mätvärden kan vi göra det på ett enkelt vis.

```
acov <- acov[ !is.na(acov$Mätvärde), ]
```

Här använder vi det faktum att utropstecken är en negation: `acov[ is.na(acov$Mätvärde), ]`hade gett oss de rader där Mätvärde är NA, men `acov[ !is.na(acov$Mätvärde), ]` ger oss alla rader där Mätvärde *inte* är NA.

Gruppera?

- Alla VC för sig eller grupperade? Glesbygd vs ÖSD? Åre?
- Åldersgrupper?
- Har alla stavat rätt? Människor är inte konsekventa!
  - "Järn" vs "järn" vs "Järn."
- Skriver vissa datum som "220921", andra som "2022-09-21" och andra som "28/9-22": hur fixa bäst? Var inte rätt för texteditorn!
- Döpa om kolumner

Vi börjar med att slå samman alla vårdcentraler:

```
> library(tidyverse)
> acov$Beställare <- str_replace_all(acov$Beställare, "VC.*", "VC")
> unique(acov$Beställare)
[1] "ÖSJ" "VC"
```

Vi kanske även vill gruppera åldrarna i grupper? Då skall vi först göra om dem till numeriska värden (se ovan), och sedan gruppera med med `cut()`, som gör om numeriska värden till en faktor: syntaxen är cut(data, brytpunkter, faktorer)

```
> acov$ålderpool <- cut(acov$Ålder, c(0,15,25,60,150), labels=c("0-15", "15-
25", "25-60", "60+"))
> str(acov)
'data.frame':   17936 obs. of  14 variables:
 $ X                  : int  17400 17401 1392 1337 1133 1419 831 1131 1128
1389 ...
 $ Kön                : chr  "K" "K" "K" "K" ...
 $ Ålder..År.         : chr  "32,03" "61,03" "49,66" "43,45" ...
 $ Beställare         : chr  "ÖSJ" "ÖSJ" "VC" "VC" ...
 $ Analys             : chr  "S-anti-SARS-CoV-2" "S-anti-SARS-CoV-2" "S-
anti-SARS-CoV-2" "S-anti-SARS-CoV-2" ...
 $ Registreringstid   : chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 08:22" "2021-02-18 13:46" ...
 $ Tekniskt.godkännande: chr  "2020-06-16 21:34" "2020-06-16 21:33" "2021-02-
17 18:00" "2021-02-19 18:05" ...
 $ Ankomsttid         : chr  "-" "-" "2021-02-17 15:45" "2021-02-19 15:40"
...
 $ Ledtid             : chr  "-" "-" "2:15:00" "2:25:00" ...
 $ Mätvärde           : num  0 0 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 ...
 $ Antal              : int  1 1 1 1 1 1 1 1 1 1 ...
 $ INDEX              : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Ålder              : num  32 61 49.7 43.5 76.8 ...
 $ ålderpool          : Factor w/ 4 levels "0-15","15-25",..: 3 4 3 3 4 3 3
4 4 3 ..
```

Ställ samman flera datafiler

Det händer ofta att man måste ställa samma flera datafiler till en. Då finns det lite olika sätt beroende på hur de ser ut, och hur man vill kombinera dem.

Två tabeller med samma kolumner till en längre tabell

Om vi har två tabeller som är lika i kolumnuppdelning kan vi kombinera med vertikalt med rbind

```
acov.65plus <- rbind(acov.m.65plus, acov.k.65plus)
```

Horisontellt

Om vi har två olika tabeller med en identifierande kolumn (PNR, LIDN, osv) kan vi slå ihop dem med merge(). Då kan vi välja:

- **inner join** Bara de rader som finns med i båda
- **full join** Alla rader
- **left join** Alla rader i den första, men bara de i den andra som också finns in den första
- **right join** Samma som left, men för den andra

Rensa bort outliers

- "5% av alla människor är outliers, vissa mer än andra"
- Skall de vara med eller inte?

Det här är med att rensa bort outliers är ett mer intrikat [statistiskt resonemang}(https://www.statology.org/remove-outliers/), som jag egentligen helst lämnar till de som är riktiga statistiker.

Avancerade städningar med regular expressions

Nyss berättade att man kan använda (https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/regex)[regular expressions] (regex) i gsub. Användbart t.ex. om man vill fixa datum som är skrivna utan århundrade eller i det horribla "28/9-22" formatet. Först skapar vi en vektor med några olika datum, sedan fixar vi stegvis

1. byter ut "." mot "-"
2. Alla tvåsiffriga år som börjar med 0,1 eller 2 får prefixet 20
3. Alla tvåsiffriga år som börjar med 3-9 får prefixet 19

```
> datum <- c("2021-09-21", "22-09-28", "87-03-24", "11.10.09", "28/9-22")
> datum <- gsub("\\.", "-", datum, perl=TRUE)
> datum
[1] "2021-09-21" "22-09-28"   "87-03-24"   "11-10-09"   "28/9-22"
> datum <- gsub("(^[012][0-9])-", "20\\1-", datum, perl=TRUE)
> datum
[1] "2021-09-21" "2022-09-28" "87-03-24"   "2011-10-09" "28/9-22"
> datum <-  gsub("(^[2-9][0-9])-", "19\\1-", datum, perl=TRUE)
> datum
[1] "2021-09-21" "2022-09-28" "1987-03-24" "2011-10-09" "28/9-22"
>
```

Några noteringar:

- Ett tips är att först ändra utan att spara till tabellen: då ser du på skärmen om rätt sak hänt

- Vad händer om du bara skriver “.” istället för “\.”? En punkt i Perls regex är “ett tecken, vilket som helst”: “\.” betyder just punkt.
- En strängs start markeras med “^”, och dess slut med “$”.
- “[123]” betyder 1, 2 eller 3
- “[3-9]” betyder 3 till 9 (funkar även med bokstäver!)
- “n{2,5}” betyder 2-5 n: nn, nnn, nnnn, nnnnn ({6,} betyder minst 6)
- Man kan gruppera med (och), för att sedan återanvända med “\1”
- Det hade gått att skriva en regex som gör allt i ett steg, men den hade varit *betydligt* svårare att tolka (och få rätt).
- Det hade självklart gått att hantera datum på 1920-talet korrekt

Pivot-tabeller

I tidyr finns det funktioner för pivottabeller, av olika typer.

```
> library(tidyverse)
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
>
> pivot_wider(mtcars, names_from = "cyl", values_from = "hp")
# A tibble: 32 × 12
     mpg  disp  drat    wt  qsec    vs    am  gear  carb   `6`   `4`   `8`
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  21    160   3.9   2.62  16.5     0     1     4     4   110    NA    NA
 2  21    160   3.9   2.88  17.0     0     1     4     4   110    NA    NA
 3  22.8  108   3.85  2.32  18.6     1     1     4     1    NA    93    NA
 4  21.4  258   3.08  3.22  19.4     1     0     3     1   110    NA    NA
 5  18.7  360   3.15  3.44  17.0     0     0     3     2    NA    NA   175
 6  18.1  225   2.76  3.46  20.2     1     0     3     1   105    NA    NA
 7  14.3  360   3.21  3.57  15.8     0     0     3     4    NA    NA   245
 8  24.4  147.  3.69  3.19  20       1     0     4     2    NA    62    NA
 9  22.8  141.  3.92  3.15  22.9     1     0     4     2    NA    95    NA
10  19.2  168.  3.92  3.44  18.3     1     0     4     4   123    NA    NA
```

En sak som jag ofta gör är att gruppera data i en sammanfattande tabell. I det enklaste fallet (t.ex. hur många i varje ålderpool, uppdelat på kön) är det enkelt:

```
> table(acov$ålderpool)

 0-15 15-25 25-60   60+
  229  1020  9377  7229
```

```
> table(acov$ålderpool, acov$Kön)

           K    M   NA
  0-15   107  122    0
  15-25  653  367    0
  25-60 6640 2737    0
  60+   3970 3259    0
```

Vill vi däremot t.ex. ha medel, median, kvartiler, osv för olika grupper är kommandot `summarisk` bra, i kombination med group_by (från dplyr, som ar många funktioner för att manipulera data). Vi använder "%>%" (från paketet magrittr) som är en pipe.

```
> acov %>%
       grop_by(ålderpool, Beställare) %>%
       summarize(
            Antal = length(Kön),
            Median = median(Mätvärde, na.rm=TRUE)
            )
`summarise()` has grouped output by 'ålderpool'. You can override using the
`.groups` argument.
# A tibble: 16 × 4
# Groups:   ålderpool [5]
   ålderpool Beställare   Antal Median
   <fct>     <chr>        <int>  <dbl>
 1 0-15      VC.GLESBYGD     33   0.1
 2 0-15      VC.Östersund    16   0.09
 3 0-15      ÖSJ            180   0.07
 4 15-25     VC.GLESBYGD    347   0.09
 5 15-25     VC.Östersund   233   0.07
 6 15-25     ÖSJ            440   0.08
 7 25-60     VC.GLESBYGD   2751   0.09
 8 25-60     VC.Östersund  2668   0.08
 9 25-60     ÖSJ           3958   0.08
10 60+       VC.GLESBYGD   1422   0.09
11 60+       VC.Östersund  1148   0.07
12 60+       ÖSJ           4659   0.08
13 NA        NA               1  NA
14 NA        VC.GLESBYGD     16   0.1
15 NA        VC.Östersund     1   0.09
16 NA        ÖSJ             64   0.09
```

Ett exempel

Här går jag igenom ett tänkt arbetsflöde med en fil.

Ladda in filen

Ladda ned tabell Utbildningsnivå 1990–2021 efter inrikes/utrikes född (fr.o.m. 2000) och kön, 25–64 år från SCBs hemsida.

```
> utbildning <- xlsx::read.xlsx2("tab10_2021.xlsx", sheetIndex = 2, startRow
= 3 )
> str(utbildning)
'data.frame':   95 obs. of  11 variables:
 $ År                                          : chr  "" "2000" "" "" ...
 $ Kön                                         : chr  "" "Totalt" "Kvinnor"
"Män" ...
 $ Befolkning..antal.                          : chr  "" "4023053" "1972971"
"2050082" ...
 $ Förgym.nasial.utb..kortare.än.9.år....      : chr  "" "8.944227182689366"
"7.628444614745984" "10.210518408531952" ...
 $ Förgym.nasial.utb..9.år....                 : chr  "" "10.88499206945571"
"9.796190618108426" "11.932839759580348" ...
 $ Gymnasial.utb..kortare.än.3.år....          : chr  "" "34.51160101544772"
"35.55379171817528" "33.508610875077196" ...
 $ Gymnasial.utb..3.år....                     : chr  "" "14.776588824457445"
"13.82311245324944" "15.694201500232674" ...
 $ Eftergym.nasial.utb..kortare.än.3.år....    : chr  "" "14.356410417660417"
"15.50043056892372" "13.255421002672088" ...
 $ Eftergym.nasial.utb..3.år.eller.längre....: chr  "" "15.505587423282766"
"17.096754083055455" "13.974270297480784" ...
 $ Forskar.utbildning....                      : chr  "" "0.7390407235500005"
"0.38758805882093555" "1.0772739822114432" ...
 $ Uppgift.om.utbildning.saknas....            : chr  "" "0.28155234345657393"
"0.21368788492076163" "0.3468641742135192" ...
```

Fyll i de tomma rutorna på År

Ofta skapas tabeller för att läsas av en människa, då ser vi enkelt att kolumnrubrikerna finns på rad 3, samt att de bara skrev ut året en gång i den första kolumnen. Genom att vi anger "startRow = 3" läser vi bara från och med rad 3, men de saknade värdena? I R är det enkelt

1	**Befolkningen 1990-2021[1], 25-64 år, fördelad efter utbildningsnivå och kön.**										
2											
3	År	Kön	Befolkning (antal)	Förgym-nasial utb. kortare än 9 år (%)	Förgym-nasial utb. 9 år (%)	Gymnasial utb. kortare än 3 år (%)	Gymnasial utb. 3 år (%)	Eftergym-nasial utb. kortare än 3 år (%)	Eftergym-nasial utb. 3 år eller längre (%)	Forskar-utbildning (%)	Uppgift om utbildning saknas (%)
31		Män	2 318 356	15	13	32	14	13	12	1,1	2
32											
33	1997	Totalt	4 598 712	12	12	34	13	14	12	0,7	1
34		Kvinnor	2 265 811	11	12	36	11	16	12	0,4	1
35		Män	2 332 901	13	13	32	14	13	12	1,1	1
36											
37	1998	Totalt	4 630 872	11	13	34	13	15	12	0,7	1

SCB tabell 10, flik 2

Först ersätter vi alla tomma rutor med NA, sedan fyller vi på med värden nedåt, gör vi om året och könet till en faktorer och slutligen tar vi bort alla "tomma" rader

```
> utbildning[ utbildning == ""] <- NA
>  utbildning <- utbildning %>% fill(År, .direction = "down")
```

```
>  utbildning$År <- as.factor(utbildning$År)
> utbildning$Kön <- as.factor(utbildning$Kön)
> utbildning <-  utbildning[ !is.na(utbildning$Befolkning..antal.),  ]
>  str(utbildning)
'data.frame':   66 obs. of  13 variables:
 $ År                                          : Factor w/ 24 levels "1
Kvalitetshöjning i statistiken över befolningens utbildning år 2000. Se
'Fördjupad information'",..: 2 2 2 3 3 3 4 4 4 5 ...
 $ Kön                                        : Factor w/ 3 levels
"Kvinnor","Män",..: 3 1 2 3 1 2 3 1 2 3 ...
 $ Befolkning..antal.                         : num  4023053 1972971 2050082
4033464 1977446 ...
 $ Förgym.nasial.utb..kortare.än.9.år....    : num  0.0894 0.0763 0.1021
0.0817 0.0689 ...
 $ Förgym.nasial.utb..9.år....               : chr  "10.88499206945571"
"9.796190618108426" "11.932839759580348" "10.723115416426179" ...
 $ Gymnasial.utb..kortare.än.3.år....        : chr  "34.51160101544772"
"35.55379171817528" "33.508610875077196" "33.82105307001624" ...
 $ Gymnasial.utb..3.år....                   : chr  "14.776588824457445"
"13.82311245324944" "15.694201500232674" "15.5540746117977" ...
 $ Eftergym.nasial.utb..kortare.än.3.år....  : chr  "14.356410417660417"
"15.50043056892372" "13.255421002672088" "14.44346596374729" ...
 $ Eftergym.nasial.utb..3.år.eller.längre....: chr  "15.505587423282766"
"17.096754083055455" "13.974270297480784" "16.230912188629922" ...
 $ Forskar.utbildning....                     : chr  "0.7390407235500005"
"0.38758805882093555" "1.0772739822114432" "0.7666859057128067" ...
 $ Uppgift.om.utbildning.saknas....          : chr  "0.28155234345657393"
"0.21368788492076163" "0.3468641742135192" "0.29123849871971086" ...
```

Räkna ut antalet i varje kategori

SCB valde att presentera proecentsatser, vilket är klart mer överskådligt. Men för oss är det enklare att arbeta med antalen. Eftersom vi har totalen i kolumn 3 kan vi räkna ut antalen. Vi skapar nya kolumner, med lite kortare namn.

1. Gör om till numeriska värden, och procent till decimaltal
2. Multiplicera

```
> utbildning$Befolkning..antal. <- as.numeric(utbildning$Befolkning..antal.)
> utbildning$Förgym.nasial.utb..kortare.än.9.år.... <-
as.numeric(utbildning$Förgym.nasial.utb..kortare.än.9.år..../100)
> utbildning$förgym.under9 <- utbildning$Befolkning..antal. *
utbildning$Förgym.nasial.utb..kortare.än.9.år....
```

Skapa en summerande tabell

```
> utbildning.kön <- utbildning %>%
+ group_by(Kön) %>%
+ summarise(Medelvärde = mean(förgym.under9,  na.rm=TRUE), Min =
min(förgym.under9), Max = max(förgym.under9), sd = sd(förgym.under9,
na.rm=TRUE))
> utbildning.kön
```

```
# A tibble: 3 × 5
  Kön     Medelvärde   Min    Max      sd
  <fct>        <dbl> <dbl>  <dbl>   <dbl>
1 Kvinnor     49644.  3126 150507  48412.
2 Män         76416.  5983 209324  68060.
3 Totalt     126060.  9109 359831 116414.
```

Övningar

Facit

1. Fortsätt att städa tabellen från SCBs hemsida som jag började med ovan∵
2. På grund av radbrytningar blir kolumnrubrikerna mer än lovligt jobiga i tabellen ovan. Använd `colnames`för att fixa dem till något tydligare.
3. Skapa en vektor med datum i olika format. Städa upp tills alla har samma format, förslagsvis “YYYY-MM-DD”