Data inlezen

Doelstelling

In deze sessie worden de methodes toegelicht om een gamma aan bestandstypen of databases in te lezen in R. Vooral bij het inlezen van tekstbestanden wordt wat dieper ingegaan op veel voorkomende problemen hierbij en hoe deze aan te pakken.

Inleiding

De basis R bevat enkel het platform om data in te lezen, het zijn de vele R pakketten (bibliotheken) die de ware kracht bevatten. Het nadeel hiervan is natuurlijk dat je verschillende bibliotheken nodig hebt om verschillende data formaten in te lezen. Hier kiezen we voor de wijze die het dichtst aansluit bij de "tidyverse" omgeving, welke een grote kracht heeft, maar wel niet altijd het eenvoudigst is als R beginner.

De meest voorkomende dataformaten om in te lezen zijn (vetgedrukt de R pakketten):

- tekstbestanden: txt, tsv, csv, fwf —> **readr**
- excel bestanden: xls, xlsx —> **readxl**
- access databases: mdb, accdb —> **RODBC (odbc)**
- eenvoudige databestanden: sqlite —> **odbc**
- databank server connecties: SQL server, MySql, postgres —> **RODBC / DBI / odbc**
- googlesheets —> **googlesheets**
- R binaire objecten —> standaard R

Om deze bibliotheken te kunnen gebruiken, moeten ze in eerste instantie geïnstalleerd zijn. Daarnaast moet je deze iedere R sessie opnieuw laden, daarom dat je het laden van de bibliotheken best in je script zelf opneemt.

```
#eenmalige installatie
install.packages("readxl")
#alternatief, ga in Rstudio naar het tabblad "Packages" en kies daar voor install.

#iedere keer je R opnieuw opent
library("readxl")
#alternatief, in het tabblad Packages vink readxl aan
```

Standaard wordt er vanuit gegaan dat de data aan volgende voorwaarden voldoet:

- rechthoekig
- kolomnamen op de eerste rij
- 1 observatieeenheid per rij
- slechts 1 datatype per kolom
- geen exotische karakters
- kleurcodes van cellen staan in een extra kolom als die belangrijk zijn, kleuren zelf zijn niet importeerbaar
- 1 dataset per tabblad
- tabblad of extern bestand met metadata die beschrijft wat de kolommen voorstellen en wat de mogelijke waarden zijn

Dit is niet vaak het geval, dus pas je ofwel de bronbestanden aan, ofwel gebruik je in R functies om hiermee om te gaan. Als je aanpassingen aanbrengt, zorg dat dit op een kopie van je origineel bestand gebeurt!

Verwijzen naar een bestand

De data wordt ingelezen via relatieve padnamen ten opzichte van je werkdirectory (die kan je terugvinden via het commando *getwd()*).

```r
getwd()

#data die in de werkdirectory staat
data <- read_csv("mijndata.csv")

#data die in een subfolder van je werkdirectory staat
data <- read_csv("subfolder/mijndata.csv")

#data die in 1 folder hoger
data <- read_csv("../mijndata.csv")

#data 2 folders hoger
data <- read_csv("../../mijndata.csv")

#data die in een andere folder staat op hetzelfde niveau als je werkfolder
data <- read_csv("../anderefolder/mijndata.csv")

#je kan ook absolute padnamen gebruiken
data <- read_csv("c:/ikke/projecten/data/mijndata.csv")

#in plaats van / kan je ook altijd \\ gebruiken in Windows systemen
#en je kan die zelfs mixen als je verwarrend wil overkomen
data <- read_csv("c:\\ikke/projecten\\data/mijndata.csv")
```

Inlezen van tekstbestanden

Tekstbestanden zijn alle bestanden die leesbaar zijn in een programma zoals kladblok. Dit zijn meestal .txt, .csv en .tsv bestanden. Deze bestanden kunnen op veel verschillende manieren gecodeerd zijn, zo gebruiken ze in de VS vooral "," als scheidingsteken en "." als decimaalteken, terwijl in Europa doorgaans eerder ";" als scheidingsteken gebruikt wordt en "," als decimaalteken.

Tekstbestanden zullen we hier via het **readr** pakket inlezen. De basisfunctie in dit pakket is *read_delim*, maar die heeft ook nog afgeleide functies met andere defaultwaarden zoals:

- *read_table* en *read_table2* : verwachten een of meerdere spaties als kolomscheidingsteken
- *read_csv* en *read_csv2*: Amerikaanse en Europese csv bestanden
- *read_tsv*: verwacht een tab als scheidingsteken (een tab wordt in R voorgesteld als "\t")
- *read_fwf*: verwacht dat iedere kolom hetzelfde aantal characters bevat
- *spec_delim*: leest enkel de kolominformatie in

Het is heel belangrijk om goed te checken of je data wel correct ingelezen is, hiervoor zijn de basisfuncties *View*, *summary*, *head*, *str* , *dim*, *table*, ... een goede hulp.

```r
library(readr)
bestandSP <- "data/20180222_species.csv"

#bewaar de filenaam in een R object (dit is niet nodig)
#Je kan die zelf laten aanvullen, bijvoorbeeld als je data/ intypt en dan op tab drukt krijg je een ove
```

Eenvoudig tekstbestand zonder teveel problemen

```r
#het inlezen
dfSpecies <- read_delim(bestandSP) #begrijpbare foutmelding

##Error in read_delim(bestandSP) : could not find function "read_delim"
```

```
dfSpecies <- read_delim(bestandSP, delim = "\t") #wtf?
```

```
## Parsed with column specification:
## cols(
##   `species_id,genus,species,taxa` = col_character()
## )
```

```
## Warning: 326 parsing failures.
## row                          col           expected actual                         file
##   1 species_id,genus,species,taxa delimiter or quote      , 'data/20180222_species.csv'
##   1 species_id,genus,species,taxa delimiter or quote      A 'data/20180222_species.csv'
##   1 species_id,genus,species,taxa delimiter or quote      , 'data/20180222_species.csv'
##   1 species_id,genus,species,taxa delimiter or quote      b 'data/20180222_species.csv'
##   1 species_id,genus,species,taxa delimiter or quote      , 'data/20180222_species.csv'
## ... ............................ .................. ...... ............................
## See problems(...) for more details.
```

```
dfSpecies <- read_delim(bestandSP, delim = ",") #dit lijkt er al op
```

```
## Parsed with column specification:
## cols(
##   species_id = col_character(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character()
## )
```

```
dfSpecies <- read_csv(bestandSP) #alternatief
```

```
## Parsed with column specification:
## cols(
##   species_id = col_character(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character()
## )
```

```
#Nakijken of de data wel juist ingelezen is

##Toon de eerste/laatste 5 rijen/de dimensies
head(dfSpecies)
```

```
## # A tibble: 6 x 4
##   species_id genus            species         taxa
##   <chr>      <chr>            <chr>           <chr>
## 1 AB         Amphispiza       bilineata       Bird
## 2 AH         Ammospermophilus harrisi         Rodent-not censused
## 3 AS         Ammodramus       savannarum      Bird
## 4 BA         Baiomys          taylori         Rodent
## 5 CB         Campylorhynchus  brunneicapillus Bird
## 6 CM         Calamospiza      melanocorys     Bird
```

```
tail(dfSpecies)
```

```
## # A tibble: 6 x 4
##   species_id genus       species    taxa
##   <chr>      <chr>       <chr>      <chr>
```

```
## 1 UP         Pipilo      sp.        Bird
## 2 UR         Rodent      sp.        Rodent
## 3 US         Sparrow     sp.        Bird
## 4 XX         <NA>        <NA>       Zero Trapping Success
## 5 ZL         Zonotrichia leucophrys Bird
## 6 ZM         Zenaida     macroura   Bird
```

```
dim(dfSpecies)
```

```
## [1] 55  4
```

```
##Toon een samenvatting van de kolommen
summary(dfSpecies)
```

```
##   species_id           genus             species
##  Length:55          Length:55          Length:55
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      taxa
##  Length:55
##  Class :character
##  Mode  :character
```

```
##Toon de data in een sorteerbaar en filterbare tabel
##is hetzelfde als op het object te klikken in het "Environment" tabblad
##Als je goed kijkt kan je hier iets vreemd zien in de eerste kolom
View(dfSpecies)

##Toon de inwendige structuur van de data
##is hetzelfde als het object uitvouwen in het "Environment" tabblad
str(dfSpecies)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 55 obs. of  4 variables:
##  $ species_id: chr  "AB" "AH" "AS" "BA" ...
##  $ genus     : chr  "Amphispiza" "Ammospermophilus" "Ammodramus" "Baiomys" ...
##  $ species   : chr  "bilineata" "harrisi" "savannarum" "taylori" ...
##  $ taxa      : chr  "Bird" "Rodent-not censused" "Bird" "Rodent" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   species_id = col_character(),
##   ..   genus = col_character(),
##   ..   species = col_character(),
##   ..   taxa = col_character()
##   .. )
```

```
##Stukje code die je gewoon kan repliceren, checkt gewoon het aantal niet-aanwezige waarden
apply(dfSpecies, 2, function(x) sum(is.na(x)))
```

```
## species_id      genus    species       taxa
##          1          1          1          0
```

```
# ---> Waar zit de fout?

?read_csv #kijk naar de parameter "na"
```

```
## starting httpd help server ...
```

```
##  done
```

```
dfSpecies <- read_csv(bestandSP, na = "") #ofwel na = character()
```

```
## Parsed with column specification:
## cols(
##   species_id = col_character(),
##   genus = col_character(),
##   species = col_character(),
##   taxa = col_character()
## )
```

Tekstbestand met iets meer problemen

Het bestand sdg_02_40.tsv is een tabgscheiden bestand die voor alle landen, alsook de Europese totalen geeft van het percentage organische landbouw.

```
bestandOF <- "data/sdg_02_40.tsv"

dfOrganic <- read_delim(bestandOF, delim = "\t") #\t betekent tab als scheidingsteken
```

```
## Parsed with column specification:
## cols(
##   `unit,crops,agprdmet,geo\time` = col_character(),
##   `2000 ` = col_character(),
##   `2001 ` = col_character(),
##   `2002 ` = col_character(),
##   `2003 ` = col_character(),
##   `2004 ` = col_character(),
##   `2005 ` = col_character(),
##   `2006 ` = col_character(),
##   `2007 ` = col_character(),
##   `2008 ` = col_character(),
##   `2009 ` = col_character(),
##   `2010 ` = col_character(),
##   `2011 ` = col_character(),
##   `2012 ` = col_character(),
##   `2013 ` = col_character(),
##   `2014 ` = col_character(),
##   `2015 ` = col_character(),
##   `2016 ` = col_character(),
##   `2017 ` = col_character()
## )
```

```
dfOrganic <- read_tsv(bestandOF) #is hetzelfde
```

```
## Parsed with column specification:
## cols(
##   `unit,crops,agprdmet,geo\time` = col_character(),
##   `2000` = col_character(),
##   `2001` = col_character(),
##   `2002` = col_character(),
##   `2003` = col_character(),
##   `2004` = col_character(),
##   `2005` = col_character(),
##   `2006` = col_character(),
##   `2007` = col_character(),
```

```
##   `2008` = col_character(),
##   `2009` = col_character(),
##   `2010` = col_character(),
##   `2011` = col_character(),
##   `2012` = col_character(),
##   `2013` = col_character(),
##   `2014` = col_character(),
##   `2015` = col_character(),
##   `2016` = col_character(),
##   `2017` = col_character()
## )
```

```
#Tevreden?
summary(dfOrganic)
```

```
##  unit,crops,agprdmet,geo\\time     2000               2001
##  Length:37                     Length:37          Length:37
##  Class :character              Class :character   Class :character
##  Mode  :character              Mode  :character   Mode  :character
##      2002               2003               2004
##  Length:37          Length:37          Length:37
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      2005               2006               2007
##  Length:37          Length:37          Length:37
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      2008               2009               2010
##  Length:37          Length:37          Length:37
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      2011               2012               2013
##  Length:37          Length:37          Length:37
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      2014               2015               2016
##  Length:37          Length:37          Length:37
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      2017
##  Length:37
##  Class :character
##  Mode  :character
```

```
View(dfOrganic)

#we verwachten numeriek, maar alles is character, dus de "na"" aanpassen
dfOrganic2 <- read_delim(bestandOF,
                         delim = "\t",
                         na = ": ")
```

```
## Parsed with column specification:
## cols(
##   `unit,crops,agprdmet,geo\time` = col_character(),
##   `2000 ` = col_character(),
##   `2001 ` = col_character(),
```

```
##   `2002 ` = col_character(),
##   `2003 ` = col_character(),
##   `2004 ` = col_character(),
##   `2005 ` = col_character(),
##   `2006 ` = col_character(),
##   `2007 ` = col_character(),
##   `2008 ` = col_character(),
##   `2009 ` = col_character(),
##   `2010 ` = col_character(),
##   `2011 ` = col_character(),
##   `2012 ` = col_character(),
##   `2013 ` = col_character(),
##   `2014 ` = col_character(),
##   `2015 ` = col_character(),
##   `2016 ` = col_character(),
##   `2017 ` = col_character()
## )
```

```
head(dfOrganic2)
```

```
## # A tibble: 6 x 19
##   `unit,crops,agp~ `2000 ` `2001 ` `2002 ` `2003 ` `2004 ` `2005 ` `2006 `
##   <chr>            <chr>   <chr>   <chr>   <chr>   <chr>   <chr>   <chr>
## 1 PC_UAA,UAAXK000~ "13.8 " "14.0 " "14.5 " "15.4 " "16.0 " "16.7 " "16.7 "
## 2 PC_UAA,UAAXK000~ "1.5 "  "1.6 "  "2.1 "  "1.7 "  "1.7 "  "1.7 "  "2.1 "
## 3 PC_UAA,UAAXK000~ <NA>    <NA>    <NA>    <NA>    <NA>    0.2 e   "0.1 "
## 4 PC_UAA,UAAXK000~ <NA>    <NA>    <NA>    <NA>    <NA>    "11 "   <NA>
## 5 PC_UAA,UAAXK000~ <NA>    <NA>    <NA>    <NA>    "0.6 "  "1 "    "1.2 "
## 6 PC_UAA,UAAXK000~ <NA>    <NA>    <NA>    "7 "    "7.2 "  "7.1 "  "7.2 "
## # ... with 11 more variables: `2007 ` <chr>, `2008 ` <chr>, `2009 ` <chr>,
## #   `2010 ` <chr>, `2011 ` <chr>, `2012 ` <chr>, `2013 ` <chr>, `2014
## #   ` <chr>, `2015 ` <chr>, `2016 ` <chr>, `2017 ` <chr>
```

```
View(dfOrganic2)

#cijfers krijgen soms commentaar als suffix (e, p, r)
#oplosbaar via parse_number
#onderstaande code loopt gewoon door kolom 2 tot 19 en
#verandert de waarden naar pure getallen
#het blijft belangrijk op te verifiëren dat je geen extra NA waarden hebt gecreëerd
for (i in 2:19){
  dfOrganic2[[i]] <- parse_number(dfOrganic2[[i]])
}

View(dfOrganic2)
summary(dfOrganic2)
```

```
##  unit,crops,agprdmet,geo\\time     2000             2001
##  Length:37                     Min.   : 0.600   Min.   : 0.700
##  Class :character              1st Qu.: 1.200   1st Qu.: 1.600
##  Mode  :character              Median : 1.800   Median : 2.300
##                                Mean   : 3.538   Mean   : 3.969
##                                3rd Qu.: 5.900   3rd Qu.: 6.375
##                                Max.   :13.800   Max.   :14.000
##                                NA's   :21       NA's   :21
```

```
##      2002             2003             2004             2005        
##  Min.   : 0.700   Min.   : 0.20   Min.   : 0.000   Min.   : 0.100  
##  1st Qu.: 2.100   1st Qu.: 2.15   1st Qu.: 1.725   1st Qu.: 1.975  
##  Median : 2.600   Median : 3.45   Median : 3.400   Median : 3.900  
##  Mean   : 4.212   Mean   : 4.38   Mean   : 4.112   Mean   : 4.447  
##  3rd Qu.: 6.500   3rd Qu.: 6.55   3rd Qu.: 6.250   3rd Qu.: 6.725  
##  Max.   :14.500   Max.   :15.40   Max.   :16.000   Max.   :16.700  
##  NA's   :20       NA's   :17      NA's   :11       NA's   :7       
##      2006             2007             2008             2009        
##  Min.   : 0.100   Min.   : 0.300   Min.   : 0.300   Min.   : 0.200  
##  1st Qu.: 2.100   1st Qu.: 1.900   1st Qu.: 2.100   1st Qu.: 2.600  
##  Median : 3.700   Median : 4.500   Median : 5.100   Median : 4.800  
##  Mean   : 4.634   Mean   : 4.821   Mean   : 5.155   Mean   : 5.562  
##  3rd Qu.: 7.200   3rd Qu.: 6.600   3rd Qu.: 7.300   3rd Qu.: 7.500  
##  Max.   :16.700   Max.   :17.000   Max.   :17.400   Max.   :18.500  
##  NA's   :8        NA's   :8        NA's   :8        NA's   :8       
##      2010             2011             2012             2013        
##  Min.   : 0.200   Min.   : 0.200   Min.   : 0.320   Min.   : 0.060  
##  1st Qu.: 2.800   1st Qu.: 2.875   1st Qu.: 3.260   1st Qu.: 2.770  
##  Median : 5.500   Median : 5.450   Median : 5.570   Median : 5.270  
##  Mean   : 6.089   Mean   : 6.282   Mean   : 6.599   Mean   : 6.132  
##  3rd Qu.: 8.450   3rd Qu.: 8.250   3rd Qu.: 8.830   3rd Qu.: 8.152  
##  Max.   :19.500   Max.   :19.600   Max.   :18.620   Max.   :18.400  
##  NA's   :9        NA's   :9        NA's   :6        NA's   :3       
##      2014             2015             2016             2017        
##  Min.   : 0.270   Min.   : 0.170   Min.   : 0.210   Min.   : 0.350  
##  1st Qu.: 2.757   1st Qu.: 2.550   1st Qu.: 3.055   1st Qu.: 3.005  
##  Median : 5.310   Median : 5.170   Median : 6.050   Median : 6.460  
##  Mean   : 6.323   Mean   : 6.527   Mean   : 7.027   Mean   : 7.427  
##  3rd Qu.: 9.105   3rd Qu.: 9.160   3rd Qu.: 9.435   3rd Qu.: 9.750  
##  Max.   :19.350   Max.   :20.300   Max.   :21.250   Max.   :23.370  
##  NA's   :3        NA's   :2        NA's   :2        NA's   :2
```

```
#het data formaat is nu OK, maar de kolomnamen en de inhoud van de eerste kolom kan nog beter
#technieken hiervoor komen volgende lessen aan bod
```

Automatische interpretatie van kolomtypes

Nu gaan we iets dieper in op enkele extra argumenten die van belang kunnen zijn: **col_names**, **col_types**, **skip**, **guess_max**

Onderstaande dataset is qua formaat geïnspireerd op een logger dataset, maar de waarden zelf zijn gewoon willekeurig ingevoerd.

De dataset bestaat uit 5 variabelen:

- datum die altijd aanwezig is
- meting aan het begin van het transect
- meting in het midden van het transect
- meting aan het einde van het transect die veel later maar meegenomen werd
- wolkenbedekking welke altijd aanwezig is en de categorieën (1,2,3,4,4+) heeft

Met de import van deze file willen we vooral het belang van data verificatie aantonen. Bij iedere poging wordt een andere naam voor de dataset gebruikt. De reden hiervoor is, dat als je een bestaande dataset in R opnieuw inleest, en er gebeuren fouten, dat de bestaande dataset ongewijzigd blijft

```r
bestandL <- "data/logger1684.txt"

#Poging om het bestand in te lezen

dfLogger1 <- read_csv(bestandL)
```

```
## Parsed with column specification:
## cols(
##   `#loggerID1684;;;;` = col_character()
## )
```

```
## Warning: 1426 parsing failures.
## row col  expected    actual                  file
##   4  -- 1 columns 3 columns 'data/logger1684.txt'
##   5  -- 1 columns 3 columns 'data/logger1684.txt'
##   6  -- 1 columns 3 columns 'data/logger1684.txt'
##   7  -- 1 columns 3 columns 'data/logger1684.txt'
##   8  -- 1 columns 3 columns 'data/logger1684.txt'
## ... ... ......... ......... ......................
## See problems(...) for more details.
```

```r
View(dfLogger1)  #wat is het probleem?

#read_csv2 gaat uit van ";" als scheidingsteken en "," als decimaalteken
dfLogger2 <- read_csv2(bestandL)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Warning: Missing column names filled in: 'X2' [2], 'X3' [3], 'X4' [4],
## 'X5' [5]
```

```
## Parsed with column specification:
## cols(
##   `#loggerID1684` = col_character(),
##   X2 = col_character(),
##   X3 = col_double(),
##   X4 = col_logical(),
##   X5 = col_double()
## )
```

```
## Warning: 461 parsing failures.
##  row col               expected actual                  file
## 1038  X5 no trailing characters   +    'data/logger1684.txt'
## 1047  X4 1/0/T/F/TRUE/FALSE       1,91 'data/logger1684.txt'
## 1048  X4 1/0/T/F/TRUE/FALSE       3,64 'data/logger1684.txt'
## 1049  X4 1/0/T/F/TRUE/FALSE       0,99 'data/logger1684.txt'
## 1050  X4 1/0/T/F/TRUE/FALSE       0,27 'data/logger1684.txt'
## .... ... ..................... ...... ......................
## See problems(...) for more details.
```

```r
head(dfLogger2)
```

```
## # A tibble: 6 x 5
##   `#loggerID1684`             X2       X3 X4       X5
##   <chr>                       <chr> <dbl> <lgl> <dbl>
## 1 #Siebens-175                <NA>  NA    NA       NA
## 2 #LastCalibration 2011-04-10 <NA>  NA    NA       NA
```

```
## 3 #begin/mid/eind/clouds      <NA>  NA    NA       NA
## 4 16/04/2011 10:00            7,33   1.01 NA        4
## 5 16/04/2011 11:00            7,7    3.05 NA        1
## 6 16/04/2011 12:00            4,5    6.39 NA        2
```

```
tail(dfLogger2)
```

```
## # A tibble: 6 x 5
##   `#loggerID1684`  X2       X3 X4       X5
##   <chr>            <chr> <dbl> <lgl> <dbl>
## 1 17/06/2011 14:00 3,48   4.09 NA        2
## 2 17/06/2011 15:00 4,65   4.23 NA        3
## 3 17/06/2011 16:00 0,53   1.44 NA        1
## 4 17/06/2011 17:00 8,43   1.09 NA        4
## 5 17/06/2011 18:00 9,24   1.19 NA        1
## 6 #EOF             <NA>  NA    NA       NA
```

```
View(dfLogger2)

#de eerste rijen horen niet thuis in de dataset
?read_csv2
dfLogger3 <- read_csv2(bestandL, skip = 4)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Warning: Missing column names filled in: 'X4' [4]
```

```
## Parsed with column specification:
## cols(
##   `16/04/2011 10:00` = col_character(),
##   `7,33` = col_character(),
##   `1,01` = col_double(),
##   X4 = col_logical(),
##   `4` = col_double()
## )
```

```
## Warning: 461 parsing failures.
##  row col               expected actual                  file
## 1034  4  no trailing characters   +    'data/logger1684.txt'
## 1043  X4 1/0/T/F/TRUE/FALSE       1,91 'data/logger1684.txt'
## 1044  X4 1/0/T/F/TRUE/FALSE       3,64 'data/logger1684.txt'
## 1045  X4 1/0/T/F/TRUE/FALSE       0,99 'data/logger1684.txt'
## 1046  X4 1/0/T/F/TRUE/FALSE       0,27 'data/logger1684.txt'
## .... ... ..................... ...... ......................
## See problems(...) for more details.
```

```
head(dfLogger3)
```

```
## # A tibble: 6 x 5
##   `16/04/2011 10:00` `7,33` `1,01` X4      `4`
##   <chr>              <chr>   <dbl> <lgl> <dbl>
## 1 16/04/2011 11:00   7,7      3.05 NA        1
## 2 16/04/2011 12:00   4,5      6.39 NA        2
## 3 16/04/2011 13:00   7,15     6.53 NA        4
## 4 16/04/2011 14:00   9,74     5.83 NA        1
## 5 16/04/2011 15:00   0,66     3.66 NA        3
## 6 16/04/2011 16:00   7,57     3    NA        2
```

```
#oei, de eerste rij wordt als kolomnamen gezien:
#reden: R verwacht standaard kolomnamen (de standaardwaarde voor col_names = TRUE)
#daarnaast wordt X4 als een logical (TRUE/FALSE) gezien --> te maken met guess_max

dfLogger4 <- read_csv2(bestandL, skip = 4, col_names = FALSE)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_double(),
##   X4 = col_logical(),
##   X5 = col_double()
## )
```

```
## Warning: 461 parsing failures.
##  row col               expected actual                  file
## 1035  X5 no trailing characters   +    'data/logger1684.txt'
## 1044  X4 1/0/T/F/TRUE/FALSE       1,91 'data/logger1684.txt'
## 1045  X4 1/0/T/F/TRUE/FALSE       3,64 'data/logger1684.txt'
## 1046  X4 1/0/T/F/TRUE/FALSE       0,99 'data/logger1684.txt'
## 1047  X4 1/0/T/F/TRUE/FALSE       0,27 'data/logger1684.txt'
## .... ... ..................... ...... .....................
## See problems(...) for more details.
```

```
View(dfLogger4) #bekijk de inhoud van de kolommen, en sorteer eens van klein naar groot en omgekeerd
dim(dfLogger4) #Je kan dit ook al direct zien in het "Environment tabblad"
```

```
## [1] 1498    5
```

```
head(dfLogger4)
```

```
## # A tibble: 6 x 5
##   X1               X2       X3 X4       X5
##   <chr>            <chr> <dbl> <lgl> <dbl>
## 1 16/04/2011 10:00 7,33   1.01 NA        4
## 2 16/04/2011 11:00 7,7    3.05 NA        1
## 3 16/04/2011 12:00 4,5    6.39 NA        2
## 4 16/04/2011 13:00 7,15   6.53 NA        4
## 5 16/04/2011 14:00 9,74   5.83 NA        1
## 6 16/04/2011 15:00 0,66   3.66 NA        3
```

```
#ziet er al beter uit, maar X2 wordt als character aanzien, X3 is OK,
#X4 als logical is niet gewenst en X5 als numerieke waarde klopt ook niet
#Ook de laatste regel mag weg, want dit bevat enkel een end of file indicatie
dfLogger5 <- read_csv2(bestandL, skip = 4,
                       col_names = c("tijdstip", "begin", "midden", "einde", "clouds"),
                       na = c("", "ERROR"),
                       n_max = 1497)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Parsed with column specification:
## cols(
##   tijdstip = col_character(),
##   begin = col_double(),
```

```
##   midden = col_double(),
##   einde = col_logical(),
##   clouds = col_double()
## )
```

```
## Warning: 407 parsing failures.
##  row    col               expected actual                  file
## 1035 clouds no trailing characters   +    'data/logger1684.txt'
## 1044 einde  1/0/T/F/TRUE/FALSE       1,91 'data/logger1684.txt'
## 1045 einde  1/0/T/F/TRUE/FALSE       3,64 'data/logger1684.txt'
## 1046 einde  1/0/T/F/TRUE/FALSE       0,99 'data/logger1684.txt'
## 1047 einde  1/0/T/F/TRUE/FALSE       0,27 'data/logger1684.txt'
## .... ...... ..................... ...... .....................
## See problems(...) for more details.
```

```
head(dfLogger5)
```

```
## # A tibble: 6 x 5
##   tijdstip         begin midden einde clouds
##   <chr>            <dbl>  <dbl> <lgl>  <dbl>
## 1 16/04/2011 10:00  7.33   1.01 NA         4
## 2 16/04/2011 11:00  7.7    3.05 NA         1
## 3 16/04/2011 12:00  4.5    6.39 NA         2
## 4 16/04/2011 13:00  7.15   6.53 NA         4
## 5 16/04/2011 14:00  9.74   5.83 NA         1
## 6 16/04/2011 15:00  0.66   3.66 NA         3
```

```
table(dfLogger5$clouds) #de 4+ is weggevallen uit clouds kolom en NA geworden
```

```
##
##   1   2   3   4
## 397 389 352 352
```

```
#Standaard wordt naar de eerste 1000 rijen gekeken om te raden welk datatype,
#maar de eerste 4+ komt bij clouds na de 1000ste rij voor
#dus het wordt als numeriek ingeschat, waardoor 4+ geen geldige waarde is en NA wordt
#Verder is er nog geen data de eerste 1000 rijen voor de kolom einde,
#dus de functie kan hier ook niet juist raden welk datatype het is
#en kiest bij alleen NA voor logical
dfLogger6 <- read_csv2(bestandL, skip = 4,
                       col_names = c("tijdstip", "begin", "midden", "einde", "clouds"),
                       na = c("", "ERROR"),
                       n_max = 1497,
                       guess_max = 5000)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Parsed with column specification:
## cols(
##   tijdstip = col_character(),
##   begin = col_double(),
##   midden = col_double(),
##   einde = col_double(),
##   clouds = col_character()
## )
```

```
summary(dfLogger6)
```

```
##    tijdstip             begin            midden          einde
##  Length:1497        Min.   : 0.000   Min.   :0.000   Min.   :0.020
##  Class :character   1st Qu.: 2.600   1st Qu.:1.670   1st Qu.:1.030
##  Mode  :character   Median : 5.060   Median :3.420   Median :2.390
##                     Mean   : 4.966   Mean   :3.451   Mean   :2.428
##                     3rd Qu.: 7.400   3rd Qu.:5.200   3rd Qu.:3.810
##                     Max.   :10.000   Max.   :7.000   Max.   :5.000
##                     NA's   :184      NA's   :67      NA's   :1096
##     clouds
##  Length:1497
##  Class :character
##  Mode  :character
##
##
##
##
```

```
#Alternatief kan je ook op voorhand de kolomtypes vastleggen
?read_csv #kijk onder col_types, daar kan je zien wat Tdddc betekent
dfLogger6b <- read_csv2(bestandL, skip = 4,
                        col_names = c("tijdstip", "begin", "midden", "einde", "clouds"),
                        na = c("", "ERROR"),
                        n_max = 1497,
                        col_types = "Tdddc")
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Warning: 1497 parsing failures.
## row      col   expected           actual                   file
##   1 tijdstip date like  16/04/2011 10:00 'data/logger1684.txt'
##   2 tijdstip date like  16/04/2011 11:00 'data/logger1684.txt'
##   3 tijdstip date like  16/04/2011 12:00 'data/logger1684.txt'
##   4 tijdstip date like  16/04/2011 13:00 'data/logger1684.txt'
##   5 tijdstip date like  16/04/2011 14:00 'data/logger1684.txt'
## ... ........ .......... ................. ......................
## See problems(...) for more details.
```

```
summary(dfLogger6b)
```

```
##     tijdstip        begin            midden          einde
##  Min.   :NA     Min.   : 0.000   Min.   :0.000   Min.   :0.020
##  1st Qu.:NA     1st Qu.: 2.600   1st Qu.:1.670   1st Qu.:1.030
##  Median :NA     Median : 5.060   Median :3.420   Median :2.390
##  Mean   :NA     Mean   : 4.966   Mean   :3.451   Mean   :2.428
##  3rd Qu.:NA     3rd Qu.: 7.400   3rd Qu.:5.200   3rd Qu.:3.810
##  Max.   :NA     Max.   :10.000   Max.   :7.000   Max.   :5.000
##  NA's   :1497   NA's   :184      NA's   :67      NA's   :1096
##     clouds
##  Length:1497
##  Class :character
##  Mode  :character
##
##
##
##
```

```
#We zijn er bijna. We hebben pech dat de datumtijd geen herkend formaat is,
#dus dit zullen we nog moeten aanpassen
?strptime

#De datum wordt nog niet correct ingelezen, dus nog een laatste poging nodig
dfLogger7 <- read_csv2(bestandL, skip = 4,
                       na = c("", "ERROR", "#EOF"),
                       col_names = c("tijdstip","begin","midden","einde","clouds"),
                       guess_max = 5000,
                       col_types = cols(tijdstip = col_datetime(format = "%d/%m/%Y %H:%M"),
                                        clouds = col_factor(levels = c(1,2,3,4,"4+"))),
                       n_max = 1497)
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
summary(dfLogger7)
```

```
##     tijdstip                       begin            midden
##  Min.   :2011-04-16 10:00:00   Min.   : 0.000   Min.   :0.000
##  1st Qu.:2011-05-02 00:00:00   1st Qu.: 2.600   1st Qu.:1.670
##  Median :2011-05-17 14:00:00   Median : 5.060   Median :3.420
##  Mean   :2011-05-17 14:00:00   Mean   : 4.966   Mean   :3.451
##  3rd Qu.:2011-06-02 04:00:00   3rd Qu.: 7.400   3rd Qu.:5.200
##  Max.   :2011-06-17 18:00:00   Max.   :10.000   Max.   :7.000
##                                NA's   :184      NA's   :67
##      einde       clouds
##  Min.   :0.020   1 :397
##  1st Qu.:1.030   2 :389
##  Median :2.390   3 :352
##  Mean   :2.428   4 :352
##  3rd Qu.:3.810   4+:  7
##  Max.   :5.000
##  NA's   :1096
```

```
View(dfLogger7)
```

Inlezen uit Excel

Vroeger werden Excel bestanden op het INBO via het pakket **RODBC** ingelezen, en er zijn ook nog verschillende alternatieve pakketten hiervoor, hier gaan we het pakket **readxl** gebruiken.

Dit pakket vermijdt het probleem dat vroeger vaak voorkwam, waar we de 32-bit versie van Excel hebben, terwijl R standaard 64-bit gebruikt en dus moest je de R versie in je Rstudio options veranderen. Daarnaast is de geïmporteerde dataset van hetzelfde formaat als je verkrijgt voor tekstbestanden via het **readr** pakket.

De voornaamste functie uit het **readxl** pakket is *read_excel* Veel argumenten van *read_excel* komen overeen met deze van read_delim uit het **readr** pakket, spijtig genoeg is de implementatie van de parameters wel niet altijd hetzelfde.

De verificatie is hier zeker even belangrijk als bij *read_csv*. In tegenstelling tot tekstbestanden kan je in veel gebruiksvriendelijkere Excel gemakkelijker je data op voorhand goed zetten.

Zorg vooral dat je in Excel de kolommen consistent 1 datatype geeft, zeker voor datumtijd, omdat er dan meer kans is dat R die onmiddellijk juist inleest.

```
library("readxl")
```

```r
bestandSV <- "data/20190124_survey_part1.xlsx"

#Naïef inlezen
dfSurvey1 <- read_excel(bestandSV, sheet = 1) #als sheet kan je ook de naam gebruiken "Blad1"
```

```
## New names:
## * `` -> `..2`
## * `` -> `..3`
## * `` -> `..4`
## * `` -> `..5`
## * `` -> `..6`
```

```r
summary(dfSurvey1) #geen kolomnamen, allemaal charactervariabelen, zelfs 1 logical
```

```
##    Plot: 2             ..2                ..3
##  Length:25          Length:25          Length:25
##  Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character
##      ..4              ..5              ..6
##  Length:25          Mode:logical   Length:25
##  Class :character   NA's:25        Class :character
##  Mode  :character                  Mode  :character
```

```r
head(dfSurvey1)
```

```
## # A tibble: 6 x 6
##   `Plot: 2`      ..2     ..3   ..4    ..5   ..6
##   <chr>          <chr>   <chr> <chr>  <lgl> <chr>
## 1 Date collected Species Sex   Weight NA    Var3
## 2 41647          NA      <NA>  <NA>   NA    1
## 3 41647          DM      M     44     NA    2
## 4 41647          DM      M     38     NA    3
## 5 41647          OL      <NA>  <NA>   NA    <NA>
## 6 41647          PE      M     22     NA    <NA>
```

```r
tail(dfSurvey1)
```

```
## # A tibble: 6 x 6
##   `Plot: 2` ..2                                    ..3   ..4   ..5   ..6
##   <chr>     <chr>                                  <chr> <chr> <lgl> <chr>
## 1 41688     DM                                     M     52    NA    <NA>
## 2 <NA>      <NA>                                   <NA>  <NA>  NA    <NA>
## 3 <NA>      <NA>                                   <NA>  <NA>  NA    <NA>
## 4 <NA>      <NA>                                   <NA>  <NA>  NA    <NA>
## 5 <NA>      <NA>                                   <NA>  <NA>  NA    <NA>
## 6 <NA>      gray cell means my measurement device ~ <NA>  <NA>  NA    <NA>
```

```r
#Dit kan gemakkelijk opgelost worden
dfSurvey2 <- read_excel(bestandSV, sheet = "Blad1", skip = 1, n_max = 19 )
```

```
## New names:
## * `` -> `..5`
```

```r
summary(dfSurvey2)
```

```
##  Date collected                  Species              Sex
##  Min.   :2014-01-08 00:00:00   Length:19          Length:19
```

```
##  1st Qu.:2014-01-08 00:00:00   Class :character   Class :character
##  Median :2014-01-08 00:00:00   Mode  :character   Mode  :character
##  Mean   :2014-01-14 11:22:06
##  3rd Qu.:2014-01-08 00:00:00
##  Max.   :2014-02-18 00:00:00
##
##      Weight        ..5              Var3
##  Min.   :  7.0   Mode:logical   Min.   :1.0
##  1st Qu.: 36.0   NA's:19        1st Qu.:1.5
##  Median : 43.0                  Median :2.0
##  Mean   : 55.6                  Mean   :2.0
##  3rd Qu.: 46.5                  3rd Qu.:2.5
##  Max.   :218.0                  Max.   :3.0
##  NA's   :4                      NA's   :16
```

```
#Je kan ook een range specifiëren en de na-waarden instellen
dfSurvey3 <- read_excel(bestandSV, sheet = "Blad1", skip = 0, range = "A2:D21",
                        na = c("", "#NAAM?"))
summary(dfSurvey3)
```

```
##  Date collected                  Species             Sex
##  Min.   :2014-01-08 00:00:00   Length:19          Length:19
##  1st Qu.:2014-01-08 00:00:00   Class :character   Class :character
##  Median :2014-01-08 00:00:00   Mode  :character   Mode  :character
##  Mean   :2014-01-14 11:22:06
##  3rd Qu.:2014-01-08 00:00:00
##  Max.   :2014-02-18 00:00:00
##
##      Weight
##  Min.   :  7.0
##  1st Qu.: 36.0
##  Median : 43.0
##  Mean   : 55.6
##  3rd Qu.: 46.5
##  Max.   :218.0
##  NA's   :4
```

```
#Je kan ook het blad in de range specifiëren en kolommen overslaan
dfSurvey4 <- read_excel(bestandSV,  range = "Blad1!A2:F21",
                        col_names = TRUE,
                        col_types = c("date", "text", "text", "numeric", "skip", "numeric"))
summary(dfSurvey4)
```

```
##  Date collected                  Species             Sex
##  Min.   :2014-01-08 00:00:00   Length:19          Length:19
##  1st Qu.:2014-01-08 00:00:00   Class :character   Class :character
##  Median :2014-01-08 00:00:00   Mode  :character   Mode  :character
##  Mean   :2014-01-14 11:22:06
##  3rd Qu.:2014-01-08 00:00:00
##  Max.   :2014-02-18 00:00:00
##
##      Weight           Var3
##  Min.   :  7.0   Min.   :1.0
##  1st Qu.: 36.0   1st Qu.:1.5
##  Median : 43.0   Median :2.0
##  Mean   : 55.6   Mean   :2.0
```

```
##  3rd Qu.: 46.5   3rd Qu.:2.5
##  Max.   :218.0   Max.   :3.0
##  NA's   :4       NA's   :16
```

```
dfSurvey5 <- read_excel(bestandSV,  range = "Blad1!A2:F21",
                        col_names = c("datum", "soort", "geslacht", "gewicht"),
                        col_types = c("date", "text", "text", "numeric", "skip", "skip"))
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i =
## sheet, : Expecting date in A2 / R2C1: got 'Date collected'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i =
## sheet, : Expecting numeric in D2 / R2C4: got 'Weight'
```

```
summary(dfSurvey5) #plotseling extra NA waarden
```

```
##      datum                         soort            geslacht
##  Min.   :2014-01-08 00:00:00   Length:20          Length:20
##  1st Qu.:2014-01-08 00:00:00   Class :character   Class :character
##  Median :2014-01-08 00:00:00   Mode  :character   Mode  :character
##  Mean   :2014-01-14 11:22:06
##  3rd Qu.:2014-01-08 00:00:00
##  Max.   :2014-02-18 00:00:00
##  NA's   :1
##     gewicht
##  Min.   :  7.0
##  1st Qu.: 36.0
##  Median : 43.0
##  Mean   : 55.6
##  3rd Qu.: 46.5
##  Max.   :218.0
##  NA's   :5
```

```
head(dfSurvey5)
```

```
## # A tibble: 6 x 4
##   datum               soort   geslacht gewicht
##   <dttm>              <chr>   <chr>      <dbl>
## 1 NA                  Species Sex           NA
## 2 2014-01-08 00:00:00 NA      <NA>          NA
## 3 2014-01-08 00:00:00 DM      M             44
## 4 2014-01-08 00:00:00 DM      M             38
## 5 2014-01-08 00:00:00 OL      <NA>          NA
## 6 2014-01-08 00:00:00 PE      M             22
```

```
#Als je expliciet de kolomnamen opgeeft, gaat de functie er vanuit dat
#de eerste rij een data-rij is in plaats van kolomheader

#Hier geven we de kolomnamen expliciet op, maar dit wil zeggen dat de data maar op rij 3 begint
dfSurvey6 <- read_excel(bestandSV,  range = "Blad1!A3:F21",
                        col_names = c("datum", "soort", "geslacht", "gewicht"),
                        col_types = c("date", "text", "text", "numeric", "skip", "skip"))
summary(dfSurvey6)
```

```
##      datum                         soort            geslacht
##  Min.   :2014-01-08 00:00:00   Length:19          Length:19
##  1st Qu.:2014-01-08 00:00:00   Class :character   Class :character
```

```
##  Median :2014-01-08 00:00:00   Mode  :character   Mode  :character
##  Mean   :2014-01-14 11:22:06
##  3rd Qu.:2014-01-08 00:00:00
##  Max.   :2014-02-18 00:00:00
##
##     gewicht
##  Min.   :  7.0
##  1st Qu.: 36.0
##  Median : 43.0
##  Mean   : 55.6
##  3rd Qu.: 46.5
##  Max.   :218.0
##  NA's   :4
```

```
View(dfSurvey6)
```

Connectie met Access

Het was de bedoeling om Access bestanden via de pakketten **DBI** en **odbc** in te lezen, maar het stuurprogramma hiervoor is niet meer op onze computers geïnstalleerd. Daarom wordt gekozen om te connecteren via het **RODBC** package. Dit geeft een pure data.frame als formaat voor de data in tegenstelling tot de "tibble" van bij readr en readxl. De output zal er dan ook lichtjes anders uit zien.

Om RODBC goed te laten werken, moet je architectuur van je R hetzelfde zijn als van de Access DB. Over het algemeen staat je R op 64-bit, maar op het INBO is Access nog 32-bit. Dus om dit goed te laten werken, moet je via Tools > Global options > General de R versie op een 32-bit versie zetten en Rstudio opnieuw opstarten.

Je maakt een ODBC connectie met de Access database via de functie *odbcConnectAccess* voor .mdb bestanden en *odbcConnectAccess2007* voor de .accdb bestanden. Eens de connectie gemaakt is, kan je enkele functies en SQL statements gebruiken om je data te bevragen. Als je in Access queries hebt kan je die ook gewoon als een tabel inlezen.

Let op. Hoewel vroeger Access gepromoot werd, en dit nog altijd een degelijk programma is om je data te bewaren, is het gevaarlijk Access te gebruiken in combinatie met Google Drive. Je databank kan immers gecorrupteerd geraken.

```
library(RODBC)

bestand <- "data/bosvitaliteit.accdb"

conn <- odbcConnectAccess(bestand)
```

```
## Warning in odbcDriverConnect(con, ...): [RODBC] ERROR: state HY000, code
## -1028, message [Microsoft][ODBC Microsoft Access-stuurprogramma] Cannot
## open database '(onbekend)'. It may not be a database that your application
## recognizes, or the file may be corrupt.
```

```
## Warning in odbcDriverConnect(con, ...): [RODBC] ERROR: state 01000, code
## 1, message [Microsoft][ODBC Microsoft Access-stuurprogramma]Algemene
## waarschuwing Kan de registersleutel Temporary (volatile) Jet DSN for
## process 0x2938 Thread 0xc9c DBC 0x7b99024 Jet niet openen.
```

```
## Warning in odbcDriverConnect(con, ...): [RODBC] ERROR: state 01000, code
## 1, message [Microsoft][ODBC Microsoft Access-stuurprogramma]Algemene
## waarschuwing Kan de registersleutel Temporary (volatile) Jet DSN for
## process 0x2938 Thread 0xc9c DBC 0x7b99024 Jet niet openen.
```

```
## Warning in odbcDriverConnect(con, ...): ODBC connection failed
```

```
conn #als dit -1 is, of een tekst van 2 regels dan is de connectie niet gelukt
```

```
## [1] -1
```

```
conn <- odbcConnectAccess2007(bestand)
conn
```

```
## RODBC Connection 2
## Details:
##   case=nochange
##   DBQ=data\bosvitaliteit.accdb
##   Driver={Microsoft Access Driver (*.mdb, *.accdb)}
##   DriverId=25
##   FIL=MS Access
##   MaxBufferSize=2048
##   PageTimeout=5
##   UID=admin
```

```
#Toon de aanwezige tabellen
sqlTables(conn)
```

```
##                    TABLE_CAT TABLE_SCHEM                 TABLE_NAME
## 1  data\\bosvitaliteit.accdb        <NA>          MSysAccessStorage
## 2  data\\bosvitaliteit.accdb        <NA>              MSysAccessXML
## 3  data\\bosvitaliteit.accdb        <NA>                   MSysACEs
## 4  data\\bosvitaliteit.accdb        <NA>         MSysComplexColumns
## 5  data\\bosvitaliteit.accdb        <NA>            MSysIMEXColumns
## 6  data\\bosvitaliteit.accdb        <NA>              MSysIMEXSpecs
## 7  data\\bosvitaliteit.accdb        <NA>                MSysNameMap
## 8  data\\bosvitaliteit.accdb        <NA> MSysNavPaneGroupCategories
## 9  data\\bosvitaliteit.accdb        <NA>          MSysNavPaneGroups
## 10 data\\bosvitaliteit.accdb        <NA>  MSysNavPaneGroupToObjects
## 11 data\\bosvitaliteit.accdb        <NA>       MSysNavPaneObjectIDs
## 12 data\\bosvitaliteit.accdb        <NA>                MSysObjects
## 13 data\\bosvitaliteit.accdb        <NA>                MSysQueries
## 14 data\\bosvitaliteit.accdb        <NA>          MSysRelationships
## 15 data\\bosvitaliteit.accdb        <NA>              MSysResources
## 16 data\\bosvitaliteit.accdb        <NA>              ~TMPCLP395461
## 17 data\\bosvitaliteit.accdb        <NA>                 cdeSoorten
## 18 data\\bosvitaliteit.accdb        <NA>                 tblOpnames
## 19 data\\bosvitaliteit.accdb        <NA>                qryMetingen
##       TABLE_TYPE REMARKS
## 1  SYSTEM TABLE    <NA>
## 2  SYSTEM TABLE    <NA>
## 3  SYSTEM TABLE    <NA>
## 4  SYSTEM TABLE    <NA>
## 5  SYSTEM TABLE    <NA>
## 6  SYSTEM TABLE    <NA>
## 7  SYSTEM TABLE    <NA>
## 8  SYSTEM TABLE    <NA>
## 9  SYSTEM TABLE    <NA>
## 10 SYSTEM TABLE    <NA>
## 11 SYSTEM TABLE    <NA>
## 12 SYSTEM TABLE    <NA>
```

```
## 13 SYSTEM TABLE    <NA>
## 14 SYSTEM TABLE    <NA>
## 15 SYSTEM TABLE    <NA>
## 16        TABLE    <NA>
## 17        TABLE    <NA>
## 18        TABLE    <NA>
## 19         VIEW    <NA>
```

```
sqlTables(conn, tableType =  c("TABLE", "VIEW"))
```

```
##                   TABLE_CAT TABLE_SCHEM    TABLE_NAME TABLE_TYPE REMARKS
## 1 data\\bosvitaliteit.accdb        <NA> ~TMPCLP395461      TABLE    <NA>
## 2 data\\bosvitaliteit.accdb        <NA>    cdeSoorten      TABLE    <NA>
## 3 data\\bosvitaliteit.accdb        <NA>    tblOpnames      TABLE    <NA>
## 4 data\\bosvitaliteit.accdb        <NA>   qryMetingen       VIEW    <NA>
```

```
#Je kan de queries ook gewoon als een tabel inlezen
dfTest <- sqlFetch(conn, sqtable = "qryMetingen")
summary(dfTest)
```

```
##       JAAR          PRVNR          BMNR           OMTREK
##  Min.   :1995   Min.   :101   Min.   : 1.00   Min.   : 27.0
##  1st Qu.:1996   1st Qu.:301   1st Qu.: 7.00   1st Qu.: 81.0
##  Median :1997   Median :505   Median :13.00   Median :107.0
##  Mean   :1997   Mean   :492   Mean   :13.72   Mean   :117.8
##  3rd Qu.:1998   3rd Qu.:703   3rd Qu.:20.00   3rd Qu.:147.0
##  Max.   :1999   Max.   :910   Max.   :40.00   Max.   :348.0
##
##       NNV                   soort
##  Min.   :  0.0   Zomereik       :2667
##  1st Qu.: 15.0   Grove den      :2188
##  Median : 20.0   Beuk           : 988
##  Mean   : 21.7   Amerikaanse eik: 716
##  3rd Qu.: 25.0   Corsicaanse den: 600
##  Max.   :100.0   populier       : 496
##                  (Other)        : 960
```

```
#Dit kan ieder mogelijke query zijn, zelfs zeer complex
query <- "select JAAR, PRVNR, BMNR, OMTREK from tblOpnames"

dfTest <- sqlQuery(conn, query)
summary(dfTest)
```

```
##       JAAR          PRVNR          BMNR           OMTREK
##  Min.   :1995   Min.   :101   Min.   : 1.00   Min.   : 27.0
##  1st Qu.:1996   1st Qu.:301   1st Qu.: 7.00   1st Qu.: 81.0
##  Median :1997   Median :505   Median :13.00   Median :107.0
##  Mean   :1997   Mean   :492   Mean   :13.72   Mean   :117.8
##  3rd Qu.:1998   3rd Qu.:703   3rd Qu.:20.00   3rd Qu.:147.0
##  Max.   :1999   Max.   :910   Max.   :40.00   Max.   :348.0
```

```
str(dfTest)
```

```
## 'data.frame':    8615 obs. of  4 variables:
##  $ JAAR  : num  1995 1995 1995 1995 1995 ...
##  $ PRVNR : num  101 101 101 101 101 101 101 101 101 101 ...
##  $ BMNR  : num  1 2 3 4 5 6 7 8 9 10 ...
```

```
##  $ OMTREK: num  102 80 104 148 129 127 129 149 131 138 ...
```

```
odbcClose(conn) #is altijd  properder je connecties weer af te sluiten

##Via DBI

#dit werkt bij mij niet omdat ik de stuurprogramma's voor MS Access niet heb
#De connectiecode zou er als volgt moeten uitzien

# dbq_string <- paste0("DBQ=", bestand)
# driver_string <- "Driver={Microsoft Access Driver (*.mdb, *.accdb)};"
# db_connect_string <- paste0(driver_string, dbq_string)
#
# myconn <- DBI::dbConnect(odbc::odbc(),
#                      .connection_string = db_connect_string)
#
# sql <- "select * from tblOpnames"
# Data <- dbGetQuery(myconn, sql)
```

Connectie met googlesheets

Om een googlesheet te kunnen inlezen heb je het *googlesheets* pakket nodig. De eerste keer, en om de zoveel tijd zal je je opnieuw moeten authentificeren via google via *gs_auth()*.

Daarna werkt het een beetje zoals **RODBC**, je moet eerst het googlesheet registreren, wat gebeurt via *gs_key* of *gs_url* of *gs_title*

Stel de URL:

https://docs.google.com/spreadsheets/d/1lJWxLIyynPHGiF6GaM1I3yO0l592tMwKOFPbFwpjvYc/edit#gid=0

en de titel van het spreadsheet "validatie" dan kan je via onderstaande code werken.

Google zelf gebruikt intern geen folderstructuur, dus er wordt altijd gezocht naar de key in de database van alle bestanden waartoe je toegang hebt.

Eens de sheet geregistreerd, dan kan je de data uitlezen via *gs_read*, de argumenten die je hier kan gebruiken zijn grotendeels hetzelfde als bij *read_delim* uit het **readr** package.

Je kan net zoals bij excel een sheet en een range specifiëren, maar het inlezen zal veel sneller gaan indien je geen range meegeeft, omdat de file dan op een andere snellere manier ingelezen wordt.

```
library(googlesheets)

gs_auth() #is enkel nodig als je authorisatie verlopen is

#De volgende 3 manieren hebben allemaal hetzelfde resultaat
#De key is een onderdeel van de URL, en is eigenlijk de basisidentificatie

ss_key <- gs_key("1lJWxLIyynPHGiF6GaM1I3yO0l592tMwKOFPbFwpjvYc")
```

```
## Sheet successfully identified: "validatie"
```

```
ss_url <-
  gs_url("https://docs.google.com/spreadsheets/d/1lJWxLIyynPHGiF6GaM1I3yO0l592tMwKOFPbFwpjvYc/edit#gid=0
```

```
## Sheet-identifying info appears to be a browser URL.
## googlesheets will attempt to extract sheet key from the URL.
```

```
## Putative key: 1lJWxLIyynPHGiF6GaM1I3yOOl592tMwKOFPbFwpjvYc
```

```
## Sheet successfully identified: "validatie"
```

```
ss_title <- gs_title("validatie")
```

```
## Sheet successfully identified: "validatie"
```

```
#Eens geregistreerd kan je de info inlezen

dfLab <- gs_read(ss_key, ws = 1)
```

```
## Accessing worksheet titled 'Sheet1'.
```

```
## Parsed with column specification:
## cols(
##   text_id = col_character(),
##   sample_name = col_character(),
##   analysis = col_character(),
##   ENTERED_ON = col_datetime(format = ""),
##   NO2 = col_character(),
##   NO3 = col_character(),
##   SO4 = col_character(),
##   PO4 = col_character()
## )
```

```
head(dfLab)
```

```
## # A tibble: 6 x 8
##   text_id  sample_name analysis ENTERED_ON          NO2   NO3   SO4   PO4
##   <chr>    <chr>       <chr>    <dttm>              <chr> <chr> <chr> <chr>
## 1 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:10:30 1.96~ 9.94~ 10.0~ 3.95~
## 2 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:13:43 2.01~ 10.3~ 10.5~ NULL
## 3 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:13:44 NULL  NULL  NULL  3.93~
## 4 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:16:41 1.96~ 10.2~ 10.3~ NULL
## 5 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:16:42 NULL  NULL  NULL  3.84~
## 6 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-11 11:11:04 2.08~ 10.0~ 10.0~ 4.06~
```

```
dfLab2 <- gs_read(ss_key, ws = "Sheet1", na = c("", "NULL"), range = "A1:H44")
```

```
## Accessing worksheet titled 'Sheet1'.
```

```
## Parsed with column specification:
## cols(
##   text_id = col_character(),
##   sample_name = col_character(),
##   analysis = col_character(),
##   ENTERED_ON = col_datetime(format = ""),
##   NO2 = col_double(),
##   NO3 = col_double(),
##   SO4 = col_double(),
##   PO4 = col_double()
## )
```

```
head(dfLab2)
```

```
## # A tibble: 6 x 8
##   text_id  sample_name analysis ENTERED_ON            NO2   NO3   SO4   PO4
##   <chr>    <chr>       <chr>    <dttm>              <dbl> <dbl> <dbl> <dbl>
```

```
## 1 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:10:30  1.97  9.94  10.0  3.95
## 2 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:13:43  2.02 10.4   10.5 NA
## 3 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:13:44 NA    NA     NA    3.94
## 4 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:16:41  1.96 10.2   10.4 NA
## 5 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-05 14:16:42 NA    NA     NA    3.84
## 6 RF_AN_0~ IC_ANION_H~ IC_ANIO~ 2017-01-11 11:11:04  2.08 10.1   10.1  4.06
```

```
#als je in google sheets consistent bent met je datumtijd notatie,
#dan is de kans reëel dat het onmiddellijk als datumtijd ingelezen wordt,
#indien niet zal je weer met col_types moeten werken zoals in het readr package
```

Connectie met INBO databanken en datawarehouses

Connectie via RODBC

Hiervoor is het het eenvoudigste dat je een gegevensbron op voorhand aanmaakt. Je kan dit door in het start menu te zoeken naar "gegevensbronnen (odbc)". Je kan aan IT de juiste coördinaten vragen voor je databank.

Onderstaand voorbeeld zou voor iedere wetenschapper moeten werken.

1. Je gaat naar start en typt gegevensbronnen en selecteert "Gegevensbronnen (ODBC)"
2. Dan kom je in het venster ODBC-gegevensbronbeheer
3. Kies toevoegen ...
4. Kies SQL server (of SQL server Native client 11 als dit niet werkt)
5. Druk op voltooien
6. Geef bij naam en beschrijving "W0003_00_Lims" in (al mag je dit vrij kiezen, maar dit is de naam waarmee je zal connecteren)
7. Als server kies je voor inbo-sql08-prd.inbo.be
8. Ga naar volgende
9. Laat op Windows Authentificatie staan en ga naar volgende
10. Vink "change default database" aan en kies "W0003_00_Lims" uit de lijst
11. Druk op volgende
12. Druk op voltooien
13. Test Data Source

```
#via RODBC
library(RODBC)

odbcDataSources() #toont de beschikbare datasources
```

```
##                                            dBASE Files
##    "Microsoft Access dBASE Driver (*.dbf, *.ndx, *.mdx)"
##                                           Excel Files
## "Microsoft Excel Driver (*.xls, *.xlsx, *.xlsm, *.xlsb)"
##                                      MS Access Database
##               "Microsoft Access Driver (*.mdb, *.accdb)"
##                                    D0004_00_Bosvitaliteit
##                                               "SQL Server"
##                                           W0003_00_Lims
##                           "SQL Server Native Client 11.0"
##                                                    test
##                           "SQL Server Native Client 11.0"
```

```
conn <- odbcConnect("W0003_00_Lims")
conn
```

```
## RODBC Connection 3
```

```
## Details:
##   case=nochange
##   DSN=W0003_00_Lims
##   Description=W0003_00_Lims
##   UID=
##   Trusted_Connection=Yes
##   WSID=LDEL001770
##   DATABASE=W0003_00_Lims
```

```
sqlTables(conn)[1:10,1:5] #er zijn er teveel om allemaal te tonen
```

```
##        TABLE_CAT TABLE_SCHEM    TABLE_NAME TABLE_TYPE REMARKS
## 1  W0003_00_Lims         dbo   DimAnalysis      TABLE    <NA>
## 2  W0003_00_Lims         dbo      DimBatch      TABLE    <NA>
## 3  W0003_00_Lims         dbo  DimComponent      TABLE    <NA>
## 4  W0003_00_Lims         dbo   DimCustomer      TABLE    <NA>
## 5  W0003_00_Lims         dbo       DimDate      TABLE    <NA>
## 6  W0003_00_Lims         dbo DimInstrument      TABLE    <NA>
## 7  W0003_00_Lims         dbo   DimLocation      TABLE    <NA>
## 8  W0003_00_Lims         dbo     DimMatrix      TABLE    <NA>
## 9  W0003_00_Lims         dbo    DimProject      TABLE    <NA>
## 10 W0003_00_Lims         dbo     DimSample      TABLE    <NA>
```

```
#Dit zijn de nuttige
sqlTables(conn, tableType = c("VIEW", "TABLE"), schema = "dbo")
```

```
##        TABLE_CAT TABLE_SCHEM     TABLE_NAME TABLE_TYPE REMARKS
## 1  W0003_00_Lims         dbo    DimAnalysis      TABLE    <NA>
## 2  W0003_00_Lims         dbo       DimBatch      TABLE    <NA>
## 3  W0003_00_Lims         dbo   DimComponent      TABLE    <NA>
## 4  W0003_00_Lims         dbo    DimCustomer      TABLE    <NA>
## 5  W0003_00_Lims         dbo        DimDate      TABLE    <NA>
## 6  W0003_00_Lims         dbo  DimInstrument      TABLE    <NA>
## 7  W0003_00_Lims         dbo    DimLocation      TABLE    <NA>
## 8  W0003_00_Lims         dbo      DimMatrix      TABLE    <NA>
## 9  W0003_00_Lims         dbo     DimProject      TABLE    <NA>
## 10 W0003_00_Lims         dbo      DimSample      TABLE    <NA>
## 11 W0003_00_Lims         dbo      DimStatus      TABLE    <NA>
## 12 W0003_00_Lims         dbo        DimUnit      TABLE    <NA>
## 13 W0003_00_Lims         dbo     FactResult      TABLE    <NA>
## 14 W0003_00_Lims         dbo vwDocumentatie       VIEW    <NA>
```

```
dfDocu <- sqlFetch(conn, "vwDocumentatie", max = 1000)

sql <- "select LabSampleID, FieldSampleID, SampleType, Project "
sql <- paste(sql, "from dimSample where Project = 'I-17W001-02'")
dfSamples <- sqlQuery(conn, query =  sql)

head(dfSamples)
```

```
##   LabSampleID FieldSampleID SampleType     Project
## 1   17-000269    AN_RPT_004       <NA> I-17W001-02
## 2   17-000270    AN_MUI_003       <NA> I-17W001-02
## 3   17-000271    AN_RPT_003       <NA> I-17W001-02
## 4   17-000272    AN_DES_004       <NA> I-17W001-02
## 5   17-000273    AN_MOL_003       <NA> I-17W001-02
```

```
## 6   17-000274    AN_MOL_004       <NA> I-17W001-02
```

```
odbcClose(conn)
```

Via DBI

In principe zou dit voor SQL server ook moeten werken via DBI. Als de connectie succesvol gelegd is, zou je ook de datastructuur moeten zien in het "connections" tabblad in Rstudio.

Deze methode zou de data veel sneller moeten inlezen, en de data wordt direct in het "tibble" formaat gezet.

```
library(DBI)
library(odbc)

con <- DBI::dbConnect(odbc::odbc(),
                      driver = "SQL Server",
                      server = "inbo-sql08-prd.inbo.be",
                      database = "W0003_00_Lims")

dbListTables(con, table_type = "TABLE", schema_name = "dbo")
```

```
##  [1] "DimAnalysis"   "DimBatch"      "DimComponent"  "DimCustomer"
##  [5] "DimDate"       "DimInstrument" "DimLocation"   "DimMatrix"
##  [9] "DimProject"    "DimSample"     "DimStatus"     "DimUnit"
## [13] "FactResult"
```

```
dfUnits <- dbReadTable(con, "dimUnit")
head(dfUnits)
```

```
##   UnitKey   LimsUnit       Unit            UnityDescription Offset Factor
## 1      -2 Ontbrekend Ontbrekend                  Ontbrekend     NA     NA
## 2      -1   Onbekend   Onbekend                    Onbekend     NA     NA
## 3       1      µEQ_L      µeq/l microequivalenten per liter      0      1
## 4       2       µG_L       µg/l         microgram per liter      0      1
## 5       3      µS_CM      µs/cm                      µq/cm      0      1
## 6       4      1000G     1000 g                     1000 g      0      1
##   IsUnit UnitType IsInferred ChangeReason InsertAuditKey UpdateAuditKey
## 1     NA       NA      FALSE            -             -2             -2
## 2     NA       NA      FALSE            -             -1             -1
## 3      0       NA      FALSE       Insert             68             68
## 4      0       NA      FALSE       Insert             68             68
## 5      0       NA      FALSE       Insert             68             68
## 6      0       NA      FALSE       Insert             68             68
##   LineageID
## 1        -2
## 2        -1
## 3         1
## 4         2
## 5         3
## 6         4
```

```
View(dfUnits)

sql = "select LabSampleID, FieldSampleID, SampleType, Project from dimSample where Project = 'I-17W001-0
dfSamples <-dbGetQuery(con, sql, n = 500)
head(dfSamples)
```

```
##   LabSampleID FieldSampleID SampleType     Project
## 1   17-000269    AN_RPT_004       <NA> I-17W001-02
## 2   17-000270    AN_MUI_003       <NA> I-17W001-02
## 3   17-000271    AN_RPT_003       <NA> I-17W001-02
## 4   17-000272    AN_DES_004       <NA> I-17W001-02
## 5   17-000273    AN_MOL_003       <NA> I-17W001-02
## 6   17-000274    AN_MOL_004       <NA> I-17W001-02
```

```r
View(dfSamples)

dbDisconnect(con)
```

binaire R objecten inlezen

Vaak heb je 1 grote data cleaning en eens je data op orde staat, is het niet meer nodig dat je telkens diezelfde stappen opnieuw uitvoert. Je kan de gecleande data opslaan als een tekstbestand, maar zeker voor grote data is een binair bestand veel sneller.

```r
#Bewaar enkele geïmporteerde datasets samen in de file mijngegevens.Rdata
save(dfSamples, dfLogger7, file = "mijngegevens.Rdata")

#Verwijder deze datasets
rm(dfSamples, dfLogger7)

#Lees deze opnieuw in via het binaire bestand
load(file = "mijngegevens.Rdata")
```