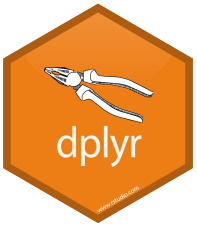
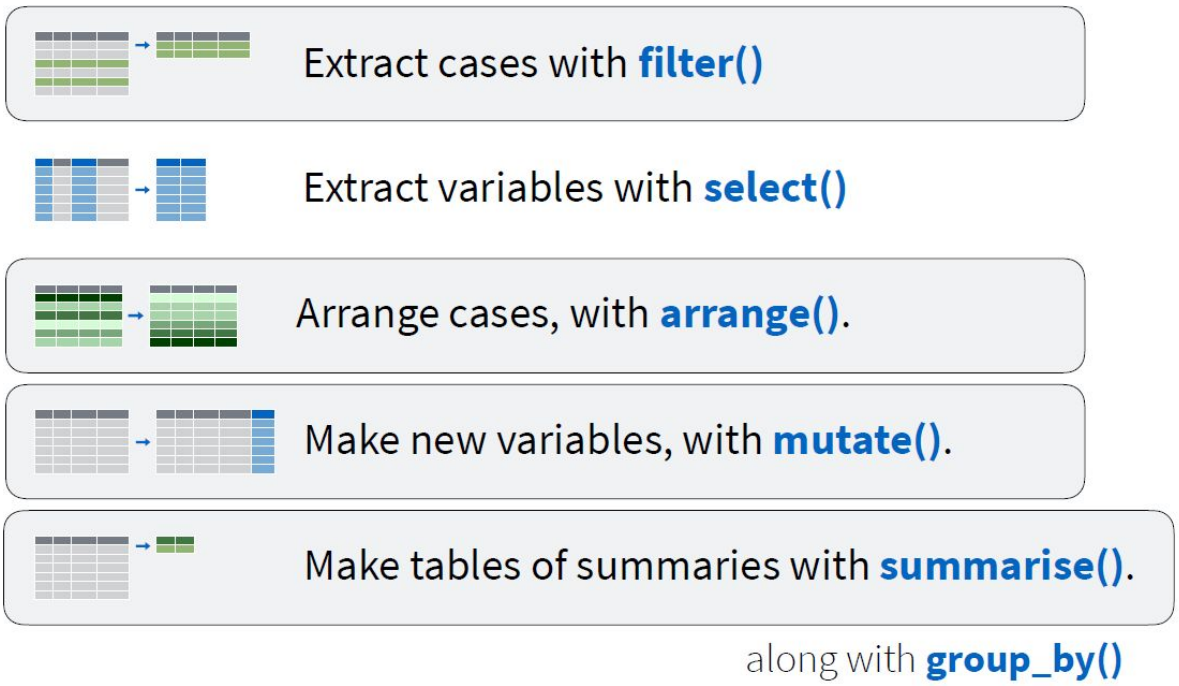
Introductie dplyr

Data manipulatie

Ivy Jansen, Pieter Verschelde



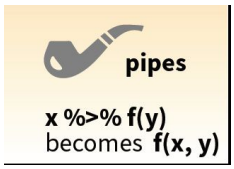
Wat kan je allemaal met dplyr?



- `dplyr` is een onderdeel van het `tidyverse` package
- Niet nodig om `dplyr` apart te installeren / laden

```
library(tidyverse)
```

Pipe %>%



Alle functies in het `dplyr` package werken op dezelfde manier.

- Als eerste argument verwachten ze de data waarop de bewerking uitgevoerd moet worden.
- Alle volgende argumenten zijn details over de uit te voeren bewerking.

Ideaal om gebruik te maken van `pipes`.

- Deze worden aangeduid met `%>%`
- Eerste argument in de functie is overbodig
  - Hetgeen voor de `pipe` staat, wordt als eerste argument (data) gebruikt in de functie na de `pipe`
- Een opeenvolging van bewerkingen mogelijk
  - Resultaat van de vorige bewerking is input voor volgende bewerking
- *Shortcut*: CTRL + SHIFT + M
- *Tip*: Eindig de regel met een `pipe` en zet de volgende bewerking op een nieuwe regel. Dit maakt de code beter leesbaar en vereenvoudigt het debuggen

```
dataset %>%
  functie1() %>%
  functie2() %>%
  ... %>%
  functieN()
```

`filter()`

- Rijen selecteren op basis van een of meerdere logische voorwaarden

```
filter(dataset, voorwaarden)
```

- Syntax met pipes

```
dataset %>%
  filter(voorwaarde)
```

Logische voorwaarden

x < y	Less than
x > y	Greater than
x == y	Equal to
x <= y	Less than or equal to
x >= y	Greater than or equal to
x != y	Not equal to
x %in% y	Group membership
is.na(x)	Is NA
!is.na(x)	Is not NA

Voorwaarden combineren

a & b	and
a | b	or
!a	not

De ampersand `&` mag ook vervangen worden door een komma om voorwaarden te combineren die beiden voldaan moeten zijn.

Voorbeelden

`iris` data

- Selecteer alle records voor de soort `virginica` en bewaar in `iris1`

```
iris1 <- filter(iris, Species == "virginica")
head(iris1)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1          6.3         3.3          6.0         2.5 virginica
## 2          5.8         2.7          5.1         1.9 virginica
## 3          7.1         3.0          5.9         2.1 virginica
## 4          6.3         2.9          5.6         1.8 virginica
## 5          6.5         3.0          5.8         2.2 virginica
## 6          7.6         3.0          6.6         2.1 virginica
```

- Selecteer alle records van `virginica` waarvoor `Sepal.Length` groter is dan of gelijk aan 7

```
filter(iris1, Sepal.Length >= 7)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 1           7.1         3.0          5.9         2.1 virginica
## 2           7.6         3.0          6.6         2.1 virginica
## 3           7.3         2.9          6.3         1.8 virginica
## 4           7.2         3.6          6.1         2.5 virginica
## 5           7.7         3.8          6.7         2.2 virginica
## 6           7.7         2.6          6.9         2.3 virginica
## 7           7.7         2.8          6.7         2.0 virginica
## 8           7.2         3.2          6.0         1.8 virginica
## 9           7.2         3.0          5.8         1.6 virginica
## 10          7.4         2.8          6.1         1.9 virginica
## 11          7.9         3.8          6.4         2.0 virginica
## 12          7.7         3.0          6.1         2.3 virginica
```

```
filter(iris, Species == "virginica" & Sepal.Length >= 7)
```

```
iris %>%
  filter(Species == "virginica") %>%
  filter(Sepal.Length >= 7)
```

`pilootstudie.csv` data

- Verwijder alle records waarvoor `Omtrek` of `Hoogte` ontbrekend zijn en bewaar het resultaat in `piloot2` (voor later gebruik)

```
piloot2 <- piloot %>%
  filter(!is.na(Omtrek) & !is.na(Hoogte))
```

- Selecteer in `piloot2` alle records van de ploegen 1, 5 en 7

```
piloot2 %>%
  filter(Ploeg %in% c(1, 5, 7))
```

```
## # A tibble: 383 x 8
##    Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##        <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
##  1         1     1     1      1   106        106.        1   22.8
##  2         1     1     1      2   105        106.        1   21.9
##  3         1     1     1      3   105        106.        1   20.7
##  4         1     1     5      1   108.       106.        2   20.6
##  5         1     2     1      1   109        109.        1   20.6
##  6         1     2     5      1   110        109.        2   21.4
```

```
##  7         1     3     1      1   103        101.       1   22.4
##  8         1     3     1      2   103        101.       1   21.7
##  9         1     3     1      3   103        101.       1   20.8
## 10         1     3     5      1   103        101.       2   19.6
## # ... with 373 more rows
```

Veel voorkomende fouten

- Gebruik van = in plaats van ==

```
filter(iris, Species = "virginica")
filter(iris, Species == "virginica")
```

- Vergeten van de aanhalingstekens rond tekst

```
filter(iris, Species == virginica)
filter(iris, Species == "virginica")
```

- Verschillende testen samengevoegd

```
filter(iris, 5 < Sepal.Length < 7)
filter(iris, 5 < Sepal.Length & Sepal.Length < 7)
```

`arrange()`

- Rijen ordenen van klein naar groot volgens een of meerdere kolommen (gescheiden door komma's)

```
arrange(dataset, variabele1, variabele2, ...)
dataset %>%
  arrange(variabele1, variabele2, ...)
```

- Ordenen van groot naar klein kan met `desc()`

```
arrange(dataset, desc(variabele))
```

- **Opgelet**: Volgorde van variabelen is belangrijk voor het resultaat van de rangschikking

Voorbeelden

`piloot2` data

- Vind de dunste bomen. Sorteer daarvoor volgens `Omtrek`

```
arrange(piloot2, Omtrek)
```

```
## # A tibble: 1,145 x 8
##    Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##        <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
##  1         5     7     4      1     30       30.9       1   8
##  2         5     7     4      1     30       30.9       2   7.8
##  3         5     7     1      1     31       30.9       1   7.74
##  4         5     7     1      2     31       30.9       1   8.08
##  5         5     7     1      3     31       30.9       1   7.96
##  6         5     7     2      1     31       30.9       1   7.95
##  7         5     7     2      2     31       30.9       1   7.76
##  8         5     7     2      3     31       30.9       1   7.63
##  9         5     7     3      1     31       30.9       1   8.22
## 10         5     7     5      1     31       30.9       2   7.6
## # ... with 1,135 more rows
```

- Vind de hoogste bomen. Sorteer daarvoor volgens `Hoogte`

```
piloot2 %>%
  arrange(desc(Hoogte))
```

```
## # A tibble: 1,145 x 8
##    Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##        <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
##  1         2     2     2      2   152        151.       1   30.6
##  2         3     2     4      1   135        132.       1   29.3
##  3         3     2     2      2   134        132.       1   29.2
##  4         3    11     5      2   122.       120.       2   29.2
##  5         3    11     5      3   121        120.       2   29.2
##  6         3     2     4      1   135        132.       2   28.8
##  7         3     8     1      2   142        141.       1   28.8
##  8         3     9     2      1   114        114.       1   28.8
##  9         2     2     1      1   153        151.       1   28.8
## 10         4    11     4      1   105        104        2   28.5
## # ... with 1,135 more rows
```

- Sorteer de bomen eerst volgens `Omtrek` en daarna volgens `Hoogte`

```
arrange(piloot2, Omtrek, Hoogte)
```

```
## # A tibble: 1,145 x 8
##    Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##        <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
##  1         5     7     4      1     30       30.9       2   7.8
##  2         5     7     4      1     30       30.9       1   8
##  3         5     7     5      1     31       30.9       2   7.6
##  4         5     7     2      3     31       30.9       1   7.63
##  5         5     7     1      1     31       30.9       1   7.74
##  6         5     7     2      2     31       30.9       1   7.76
##  7         5     7     2      1     31       30.9       1   7.95
##  8         5     7     1      3     31       30.9       1   7.96
##  9         5     7     1      2     31       30.9       1   8.08
## 10         5     7     3      1     31       30.9       1   8.22
## # ... with 1,135 more rows
```

- Sorteer de bomen eerst volgens `Hoogte` en daarna volgens `Omtrek`

```
piloot2 %>%
  arrange(Hoogte, Omtrek)
```

```
## # A tibble: 1,145 x 8
##    Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##        <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
##  1         7     8     3      1    110       108.       1   4.89
##  2         7     8     5      1    109       108.       2   5
##  3         7     8     4      3    110       108.       1   5.05
##  4         7     8     5      2    110       108.       2   5.1
##  5         7     8     5      3    110       108.       2   5.1
##  6         7     8     4      1    109       108.       1   5.16
##  7         7     8     4      2    109       108.       2   5.2
##  8         7     8     4      3    110       108.       2   5.2
##  9         7     8     4      1    109       108.       2   5.3
## 10         7     8     2      1    110       108.       1   5.3
## # ... with 1,135 more rows
```

`mutate()`

- Nieuwe variabele(n) aanmaken op basis van bestaande variabele(n) in de dataset

```
mutate(dataset, NieuweVariabele1, NieuweVariabele2, ...)
dataset %>%
  mutate(NieuweVariabele1, NieuweVariabele2, ...)
```

- Altijd van de vorm `NieuweVariabele = bewerking op bestaande variabele(n)`
- Mogelijk om meerdere variabelen tegelijk aan te maken
  - Gescheiden door komma's
  - Al mogelijk om nieuwe variabele onmiddellijk te gebruiken
- **Belangrijk**: resultaat moet een even lange vector zijn als de input

Voorbeelden

`iris` data

- Oppervlakte van de Sepal blaadjes

```
NieuweIris <- mutate(iris, Sepal.Opp = Sepal.Length * Sepal.Width)
head(NieuweIris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Sepal.Opp
## 1          5.1         3.5          1.4         0.2  setosa     17.85
## 2          4.9         3.0          1.4         0.2  setosa     14.70
## 3          4.7         3.2          1.3         0.2  setosa     15.04
## 4          4.6         3.1          1.5         0.2  setosa     14.26
## 5          5.0         3.6          1.4         0.2  setosa     18.00
## 6          5.4         3.9          1.7         0.4  setosa     21.06
```

- Verhouding van de oppervlakte van de kelk- en kroonblaadjes

```
iris %>%
  mutate(Sepal.Opp = Sepal.Length * Sepal.Width,
         Petal.Opp = Petal.Length * Petal.Width,
         Verhouding = Sepal.Opp / Petal.Opp) %>%
  head()
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species Sepal.Opp
## 1          5.1         3.5          1.4         0.2  setosa     17.85
## 2          4.9         3.0          1.4         0.2  setosa     14.70
## 3          4.7         3.2          1.3         0.2  setosa     15.04
## 4          4.6         3.1          1.5         0.2  setosa     14.26
## 5          5.0         3.6          1.4         0.2  setosa     18.00
## 6          5.4         3.9          1.7         0.4  setosa     21.06
##   Petal.Opp Verhouding
## 1      0.28   63.75000
## 2      0.28   52.50000
## 3      0.26   57.84615
## 4      0.30   47.53333
## 5      0.28   64.28571
## 6      0.68   30.97059
```

`select()`

- Een of meerdere kolommen selecteren, namen gescheiden door komma's

```
select(dataset, kolomnaam1, kolomnaam2, ...)
dataset %>%
  select(kolomnaam1, kolomnaam2, ...)
```

- Alternatief voor vierkante haken `[]` of namen expliciet meegeven

  - Past in de hele piping filosofie

- Interessante functies om gelijkaardige kolommen te selecteren
  - `starts_with("xxx")`: alle kolommen waarvan de naam begint met `xxx`
  - `ends_with("xxx")`: alle kolommen waarvan de naam eindigt met `xxx`
  - `contains("xxx")`: alle kolommen waarvan de naam `xxx` bevat

```
select(iris, starts_with("Petal"))
select(iris, -ends_with("Width"))
```

- Mogelijk om geselecteerde variabelen ineens van naam te veranderen
  - Nadeel dat alle niet-genoemde variabelen niet meegenomen worden

```
select(iris, Soort = Species)
```

`summarise()`

- Samenvattende waarde(n) berekenen
- Mogelijkheid om meerdere kenmerken (functies) te combineren
- Geeft slechts 1 waarde (per functie) terug

```
summarise(data, functie(variabele))
data %>%
  summarise(functie(variabele))
```

- Varianten om kenmerken te berekenen voor alle variabelen, of een selectie van variabelen (gebruiken een iets andere syntax, zie `help`)
  - `summarise_all()`
  - `summarise_at()`
  - `summarise_if()`

Voorbeelden

`piloot2` data

- Vind de dunste boom

```
summarise(piloot2, Dunste = min(Omtrek))
```

```
## # A tibble: 1 x 1
##   Dunste
##    <dbl>
## 1     30
```

- Vind de hoogste boom, het aantal proefvlakken, en de mediaan voor de referentiemetingen

```
piloot2 %>%
  summarise(HoogsteBoom = max(Hoogte),
            AantalProefVlakken = n_distinct(Proefvlak),
            MediaanRef = median(Referentie))
```

```
## # A tibble: 1 x 3
##   HoogsteBoom AantalProefVlakken MediaanRef
##         <dbl>              <int>      <dbl>
## 1        30.6                  8       111.
```

- Bereken voor alle variabelen het gemiddelde

```
piloot2 %>%
  summarise_all(mean)
```

```
## # A tibble: 1 x 8
##   Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##       <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
## 1      4.49  6.50  3.16   1.75   111.       110.    1.33   21.5
```

`iris` data

- Bereken voor alle numerieke variabelen het gemiddelde

```
iris %>%
  summarise_if(is.numeric, mean)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width
## 1     5.843333    3.057333        3.758    1.199333
```

- Bereken voor alle variabelen (behalve Species) het minimum en het maximum

```
iris %>%
  summarise_at(vars(-Species),
               list(minimum = min,
                    maximum = max))
```

```
##   Sepal.Length_minimum Sepal.Width_minimum Petal.Length_minimum
## 1                  4.3                   2                   1
##   Petal.Width_minimum Sepal.Length_maximum Sepal.Width_maximum
## 1                 0.1                 7.9                 4.4
##   Petal.Length_maximum Petal.Width_maximum
## 1                  6.9                 2.5
```

`group_by()`

- Gegevens groeperen volgens een of meerdere variabelen. Dit doet niks, behalve er een gegroepeerde `tibble` van maken.

```
group_by(data, variabele1, variabele2, ...)
data %>%
  group_by(variabele1, variabele2, ...)
```

- Groepering ongedaan maken met `ungroup()`
  - Meestal niet nodig, maar sommige functies kunnen niet om met gegroepeerde data
- Meestal gebruikt in combinatie met `summarise()` om per groep samenvattende kenmerken te kunnen berekenen

```
data %>%
  group_by(variabele_i) %>%
  summarise(functie(variabele_j))
```

Voorbeelden

`iris` data

- Bereken per soort het aantal waarnemingen en de gemiddelde `Sepal.Width`

```
iris %>%
  group_by(Species) %>%
  summarise(Aantal = n(),
            Gemiddelde = mean(Sepal.Width))
```

```
## # A tibble: 3 x 3
##   Species    Aantal Gemiddelde
##   <fct>       <int>      <dbl>
```

```
## 1 setosa         50       3.43
## 2 versicolor     50       2.77
## 3 virginica      50       2.97
```

piloot2 data

- Bereken per boom het minimum, gemiddelde en maximum van de omtrek en de hoogte (ook nodig om te groeperen per proefvlak omdat de nummering van de bomen in elk proefvlak opnieuw begint)

```
piloot2 %>%
  group_by(Proefvlak, Boom) %>%
  summarise(MinOmtrek = min(Omtrek),
            GemOmtrek = mean(Omtrek),
            MaxOmtrek = max(Omtrek),
            MinHoogte = min(Hoogte),
            GemHoogte = mean(Hoogte),
            MaxHoogte = max(Hoogte))
```

```
## # A tibble: 96 x 8
## # Groups:   Proefvlak [8]
##    Proefvlak  Boom MinOmtrek GemOmtrek MaxOmtrek MinHoogte GemHoogte
##        <dbl> <dbl>     <dbl>     <dbl>     <dbl>     <dbl>     <dbl>
##  1         1     1       105     106.        110      20.4      22.1
##  2         1     2        99     108.        111      19.3      21.4
##  3         1     3       101     102.        103      18.6      21.0
##  4         1     4       112     113.        114.     20.9      22.1
##  5         1     5       115     117.        118      20.8      23.4
##  6         1     6        98      99.7       109      19.5      20.9
##  7         1     7        83      84.4        86      19.5      21.0
##  8         1     8        99     101.        103      18.1      20.6
##  9         1     9        97      98.2        99      17        19.5
## 10         1    10       113     115.        116      19.7      21.7
## # ... with 86 more rows, and 1 more variable: MaxHoogte <dbl>
```

- Bereken per ploeg het aantal metingen per proefvlak

```
piloot2 %>%
  group_by(Proefvlak, Ploeg) %>%
  summarise(Aantal = n())
```

```
## # A tibble: 40 x 3
## # Groups:   Proefvlak [8]
##    Proefvlak Ploeg Aantal
##        <dbl> <dbl>  <int>
##  1         1     1     24
##  2         1     2     24
##  3         1     3     24
##  4         1     4     48
##  5         1     5     24
##  6         2     1     24
##  7         2     2     24
##  8         2     3     24
##  9         2     4     48
## 10         2     5     24
## # ... with 30 more rows
```

Andere interessante functies

`distinct()`

- Verwijder dubbele rijen
- Mogelijk om een of meerdere variabelen te specifiëren waarnaar gekeken moet worden

`top_n(data, n, variabele)`

- Sorteer rijen volgens een variabele en neem de bovenste n rijen

`slice()`

- Selecteer rijen op basis van de positie

`transmute()`

- Alternatief voor `mutate()`
- Behoudt enkel de nieuw aangemaakte variabelen

`rename(data, NieuweNaam = OudeNaam)`

- Variabelen hernoemen
- Alternatief voor hernoemen met `select()`
- Voordeel dat alle variabelen die niet hernoemd worden, identiek in de data blijven

`count(data, variabelen)`

- Tel het aantal rijen in de groepen gedefinieerd door de variabelen
- Verkorte vorm voor

```
data %>%
  group_by(variabelen) %>%
  summarise(n = n())
```

- Heeft nog de extra optie om te sorteren: `count(variabele, sort = TRUE)`

Alle varianten van `join`

Voeg kolommen van de ene tabel toe aan de andere tabel, door de rijen te koppelen volgens corresponderende waarden. Kan ook gebruikt worden om een tabel te filteren op de rijen van een andere tabel. Elke `join` resulteert in een andere combinatie van waarden uit beide tabellen.

- `left_join()`
- `right_join()`
- `inner_join()`
- `full_join()`
- `semi_join()`
- `anti_join()`

Tidy data

- Tidy data = ordelijke gegevens
- Volgens 4 principes
  - Elke observatie vormt een rij
  - Elke variabele vormt een kolom
  - Elke cel bevat een waarde
  - Elk type van observationele eenheid vormt een tabel

Untidy

Locatie	2008	2014	Oppervlakte	Provincie
Hasselt	71	76	10224	Limburg
Gent	237	251	15617	Oost-Vlaanderen
Brugge	117	117	13840	West-Vlaanderen

Tidy

Locatie	Jaar	Inwoners	OppWoonfunctie	PrijsBouwgrond
Hasselt	2008	71	1732	134
Gent	2008	237	2991	181
Brugge	2008	117	2067	205
Hasselt	2014	76	1837	190
Gent	2014	251	3132	276
Brugge	2014	117	2148	223

Locatie	Oppervlakte	Provincie	Longitude	Latitude
Hasselt	10224	Limburg	5.3325	50.9307
Gent	15617	Oost-Vlaanderen	3.7174	51.0543
Brugge	13840	West-Vlaanderen	3.2247	51.2093

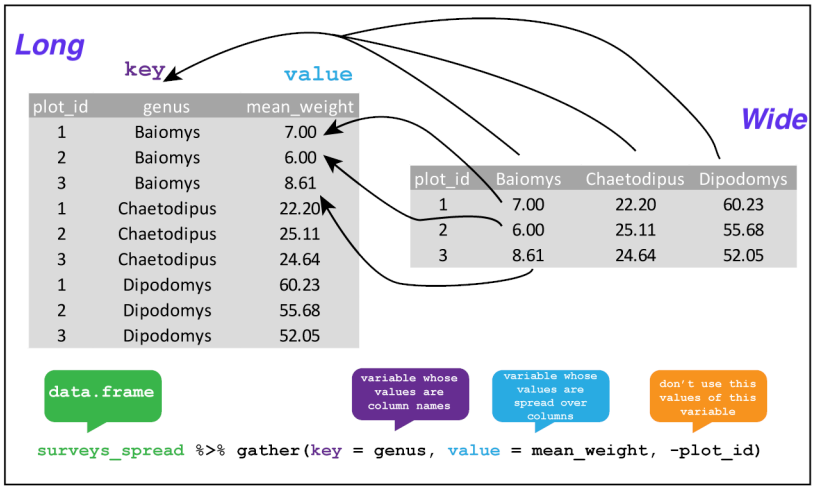
Gebruik functies uit het `tidyr` package (vervat in `tidyverse`) om de gegevens *tidy* te maken.

Van breed naar lang formaat met `gather()`

- Kolomnamen zijn geen variabelen, maar waarden van een variabele
- Informatie van deze kolommen *verzamelen* in nieuwe variabelen

```
gather(data, key = key, value = value, ...)
data %>%
  gather(key = key, value = value, ...)
```

- `key`: naam van de nieuwe variabele die de kolomnamen zal bevatten
- `value`: naam van de nieuwe variabele die de waarden zal bevatten
- `...`: kolomnamen die onder mekaar geplaatst moeten worden
  - Op dezelfde manier te selecteren als in de functie `select()`



Voorbeelden

iris data

We tonen deze resultaten voor een subset van de `iris` data (rijen 1, 51 en 101)

```
iris_mini <- iris %>%
  slice(c(1, 51, 101))
iris_mini
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 1          5.1         3.5          1.4         0.2     setosa
## 2          7.0         3.2          4.7         1.4 versicolor
## 3          6.3         3.3          6.0         2.5  virginica
```

- Maak de kolommen `Kenmerk` en `Waarde` aan, die alle info bevatten van de 4 `Sepal` en `Petal` variabelen

```
iris_mini %>%
  gather(key = Kenmerk, value = Waarde,
         Sepal.Length, Sepal.Width, Petal.Length, Petal.Width)
```

```
##        Species      Kenmerk Waarde
## 1       setosa Sepal.Length    5.1
## 2   versicolor Sepal.Length    7.0
## 3    virginica Sepal.Length    6.3
## 4       setosa  Sepal.Width    3.5
## 5   versicolor  Sepal.Width    3.2
## 6    virginica  Sepal.Width    3.3
## 7       setosa Petal.Length    1.4
## 8   versicolor Petal.Length    4.7
## 9    virginica Petal.Length    6.0
## 10      setosa  Petal.Width    0.2
## 11  versicolor  Petal.Width    1.4
## 12   virginica  Petal.Width    2.5
```

- Nieuwe kolommen bevatten info van alle kolommen, behalve van `Species` (idem resultaat)

```
iris_mini %>%
  gather(key = Kenmerk, value = Waarde, -Species)
```

pilootstudie.csv data

We tonen deze resultaten voor een subset van 4 rijen uit de `pilootstudie` data

```
piloot_mini <- piloot %>%
  sample_n(4)
piloot_mini
```

```
## # A tibble: 4 x 8
##   Proefvlak  Boom Ploeg Meting Omtrek Referentie Toestel Hoogte
##       <dbl> <dbl> <dbl>  <dbl>  <dbl>      <dbl>   <dbl>  <dbl>
## 1         1    11     7      3   94.5       94.7      NA   NA
## 2         5    12     4      3  116        115.        1   22.7
## 3         5    11     1      3   75         75.4       1   18.3
## 4         5    12     5      1  116.       115.        2   23.3
```

- Maak de kolommen `Kenmerk` en `Waarde` aan, die alle info bevatten van de variabelen `Omtrek` en `Hoogte`

```
piloot_mini %>%
  gather(key = Kenmerk, value = Waarde,
         Omtrek, Hoogte)
```

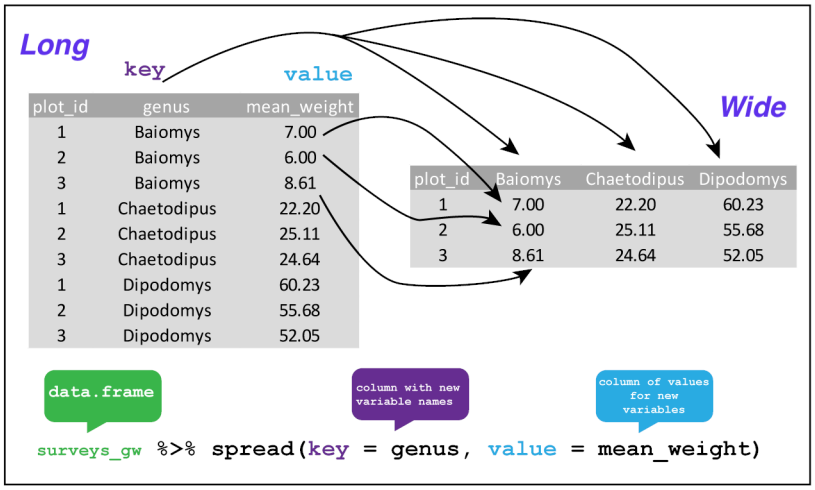
```
## # A tibble: 8 x 8
##   Proefvlak  Boom Ploeg Meting Referentie Toestel Kenmerk Waarde
##       <dbl> <dbl> <dbl>  <dbl>      <dbl>   <dbl> <chr>    <dbl>
## 1         1    11     7      3       94.7      NA Omtrek    94.5
## 2         5    12     4      3      115.        1 Omtrek   116
## 3         5    11     1      3       75.4       1 Omtrek    75
## 4         5    12     5      1      115.        2 Omtrek   116.
## 5         1    11     7      3       94.7      NA Hoogte    NA
## 6         5    12     4      3      115.        1 Hoogte    22.7
## 7         5    11     1      3       75.4       1 Hoogte    18.3
## 8         5    12     5      1      115.        2 Hoogte    23.3
```

Van lang naar breed formaat met `spread()`

- Omgekeerde van `gather()`
- Observaties zijn verspreid over meerdere rijen, en je wil ze *uitspreiden* over de kolommen

```
spread(data, key = key, value = value)
data %>%
  spread(key = key, value = value)
```

- `key`: de variabele die de nieuwe kolomnamen bevat
- `value`: de variabele die de waarden bevat



Voorbeelden

`iris` data

In de lange dataset (uit de voorbeelden van `gather()`) splitsen we de variabele `Kenmerk` nog op in `Blad` en `Afmeting` en bewaren het resultaat in `iris_lang`.

```
##       Species  Blad Afmeting Waarde
## 1      setosa Sepal   Length    5.1
## 2  versicolor Sepal   Length    7.0
## 3   virginica Sepal   Length    6.3
## 4      setosa Sepal    Width    3.5
## 5  versicolor Sepal    Width    3.2
## 6   virginica Sepal    Width    3.3
## 7      setosa Petal   Length    1.4
## 8  versicolor Petal   Length    4.7
## 9   virginica Petal   Length    6.0
## 10     setosa Petal    Width    0.2
## 11 versicolor Petal    Width    1.4
## 12  virginica Petal    Width    2.5
```

- Maak nieuwe kolommen die de lengte en breedte van de blaadjes bevatten

```
iris_lang %>%
  spread(key = Afmeting, value = Waarde)
```

```
##      Species  Blad Length Width
## 1     setosa Petal    1.4   0.2
## 2     setosa Sepal    5.1   3.5
## 3 versicolor Petal    4.7   1.4
## 4 versicolor Sepal    7.0   3.2
## 5  virginica Petal    6.0   2.5
## 6  virginica Sepal    6.3   3.3
```

`pilootstudie.csv` data

We behouden telkens de eerste meting en selecteren de variabelen `Proefvlak`, `Boom`, `Ploeg` en `Omtrek`.

```
## # A tibble: 672 x 4
##    Proefvlak  Boom Ploeg Omtrek
##        <dbl> <dbl> <dbl>  <dbl>
```

```
##  1         1     1     1   106
##  2         1     1     2   108
##  3         1     1     3   110
##  4         1     1     4   105
##  5         1     1     5   108.
##  6         1     1     6   107
##  7         1     1     7   107.
##  8         1     2     1   109
##  9         1     2     2   109
## 10         1     2     3   111
## # ... with 662 more rows
```

- Spreid de metingen van elke ploeg over verschillende kolommen

```
piloot_lang %>%
  spread(key = Ploeg, value = Omtrek)
```

```
## # A tibble: 96 x 9
##    Proefvlak  Boom   `1`   `2`   `3`   `4`   `5`   `6`   `7`
##        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1         1     1   106   108   110   105 108.    107 107.
##  2         1     2   109   109   111   109 110     110 109.
##  3         1     3   103   102   103   101 103     103 102.
##  4         1     4   112   114   114   113 114.    114 112.
##  5         1     5   115   117   118   116 117     117 117.
##  6         1     6    98   100    99    98 100.    100  94.4
##  7         1     7    85    84    86    83  85.5    85  83.8
##  8         1     8    99   103   103   100 102     100  99.8
##  9         1     9    97    99    99    97  99      98  97.1
## 10         1    10   113   115   115   114 114.    114 113.
## # ... with 86 more rows
```

More to learn

- R for data science (Hoofdstuk 5 en 12)
  - Boek van Hadley Wickham en Garrett Grolemund
  - Hardcopy beschikbaar op INBO
  - Digitale versie
- Datacamp
  - (gedeeltelijk) gratis lessen (video tutorials en oefeningen)
  - Account voor 72h voor volledige toegang, daarna betalende licentie (~ €25/maand)
  - Introduction to the Tidyverse
  - Data Wrangling
- Data Carpentry
  - Manipulating, analyzing and exporting data with tidyverse
- Stat 545
  - Introduction to dplyr
  - dplyr functions for a single dataset
- Coding Club Edinburgh
  - Easy and efficient data manipulation
- INBO Coding Club
  - Tidy data
- Cheat Sheets
  - In RStudio onder `Help` menu
  - Online

Referenties

- Transform data with dplyr
- R for data science