KlantArtikel Queries

Uitleg

We maken gebruik van dat die in een SQLite database zit, een zogenaamde 'portable' database. Deze database bevat allerlei verkoopdata. Iedere gele in de data staat voor een zogenaamde verkoopregel: op een besetelling kunnen meerdere producten besteld/gekocht zijn. 1 regel is dus 1 zo'n verkoopregel. Er kunnen meerdere verkoopregels bij een bestelling horen. ## Setup Voordat we aan de slag kunnen, moeten er altijd 2 zaken geregeld worden:

- De juiste libraries (voorgedefinieerde functies) inladen. als die er niet zijn, moeten we ze installeren mbv het `install.packages('naam library')` commando in de console
- De benodigde data inladen.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(RSQLite)
library(ggplot2)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(dplyr)
conn <- dbConnect(RSQLite::SQLite(), "CustomerSales.db")
```

Vervolgens laden we de data in via een SQL query en maken deze in R beschikbaar als data frame dmv de parameter `output.var` toe te voegen in de kop van de chunk.

```
Select * FROM factSales
```

En we kunnen nu disconnecten

```
dbDisconnect(conn = conn)
```

Stap 1: Data prepareren: Datums bruikbaar maken

De datumkolom in deze dataset is nog niet van een bruikbaar datumtype. Converteer deze naar een datumtype. Er blijkt iets mis te zijn in de datering, waardoor je overal 6 jaar bij op moet tellen. Maak daarvoor een kolom `realdate` aan.

```
##   bestelnr besteldatum klantnr artikelnr aantal regelomzet
## 1     1001  2018-01-06      15       408      9     146.25
## 2     1002  2018-01-07      16       407      6      77.70
## 3     1002  2018-01-07      16       410     15     315.00
## 4     1003  2018-01-08      13       401     18     387.00
## 5     1004  2018-01-14      19       406      7      52.50
## 6     1005  2018-01-19      17       406     19     142.50
```

Stap 2: Datum velden extraheren

Nu we een relevant en bruikbaar jaartal hebben, is het ivm de rapportages die we willen gaan maken handig om per regel aparte kolommen te maken voor - Jaar - Kwartaal - Dag van de week (maandag tm zondag)

```
##   bestelnr besteldatum klantnr artikelnr aantal regelomzet Jaar Kwartaal
## 1     1001  2018-01-06      15       408      9     146.25 2018        1
## 2     1002  2018-01-07      16       407      6      77.70 2018        1
## 3     1002  2018-01-07      16       410     15     315.00 2018        1
## 4     1003  2018-01-08      13       401     18     387.00 2018        1
## 5     1004  2018-01-14      19       406      7      52.50 2018        1
## 6     1005  2018-01-19      17       406     19     142.50 2018        1
##   DayOfWeek
## 1  Saturday
## 2    Sunday
## 3    Sunday
## 4    Monday
## 5    Sunday
## 6    Friday
```

Stap 3 : overige kolommen in juist formaat

Zoals je kan zien wordt klantnummer als een integere gezien. Dat klopt op zich, maar tegelijkertijd is het niet iets waar we mee gaan rekenen. We kunnen klantnummer daarom beter als een factor gaan omzetten, want een klantnr 12 is niet 'meer' dan klantnr 11 bijvoorbeeld. Idem voor bestelnr en artikelnr

```
Salesdata <- Salesdata %>%
  mutate(klantnr = as.factor(klantnr),
         bestelnr = as.factor(bestelnr),
         artikelnr = as.factor(artikelnr))
head(Salesdata)
```

```
##   bestelnr besteldatum klantnr artikelnr aantal regelomzet Jaar Kwartaal
## 1     1001  2018-01-06      15       408      9     146.25 2018        1
## 2     1002  2018-01-07      16       407      6      77.70 2018        1
## 3     1002  2018-01-07      16       410     15     315.00 2018        1
## 4     1003  2018-01-08      13       401     18     387.00 2018        1
## 5     1004  2018-01-14      19       406      7      52.50 2018        1
## 6     1005  2018-01-19      17       406     19     142.50 2018        1
##   DayOfWeek
## 1  Saturday
## 2    Sunday
## 3    Sunday
## 4    Monday
## 5    Sunday
## 6    Friday
```

Vraag 1 - Vergelijken kwartaalomzet

Nu wat basale bewerkingen hebben gedaan, s het tijd om het wat moeilijker te gaan maken. We moeten een grafiek laten zien waarin we de omzetontwikkeling in Q1 per jaar moeten laten zien.

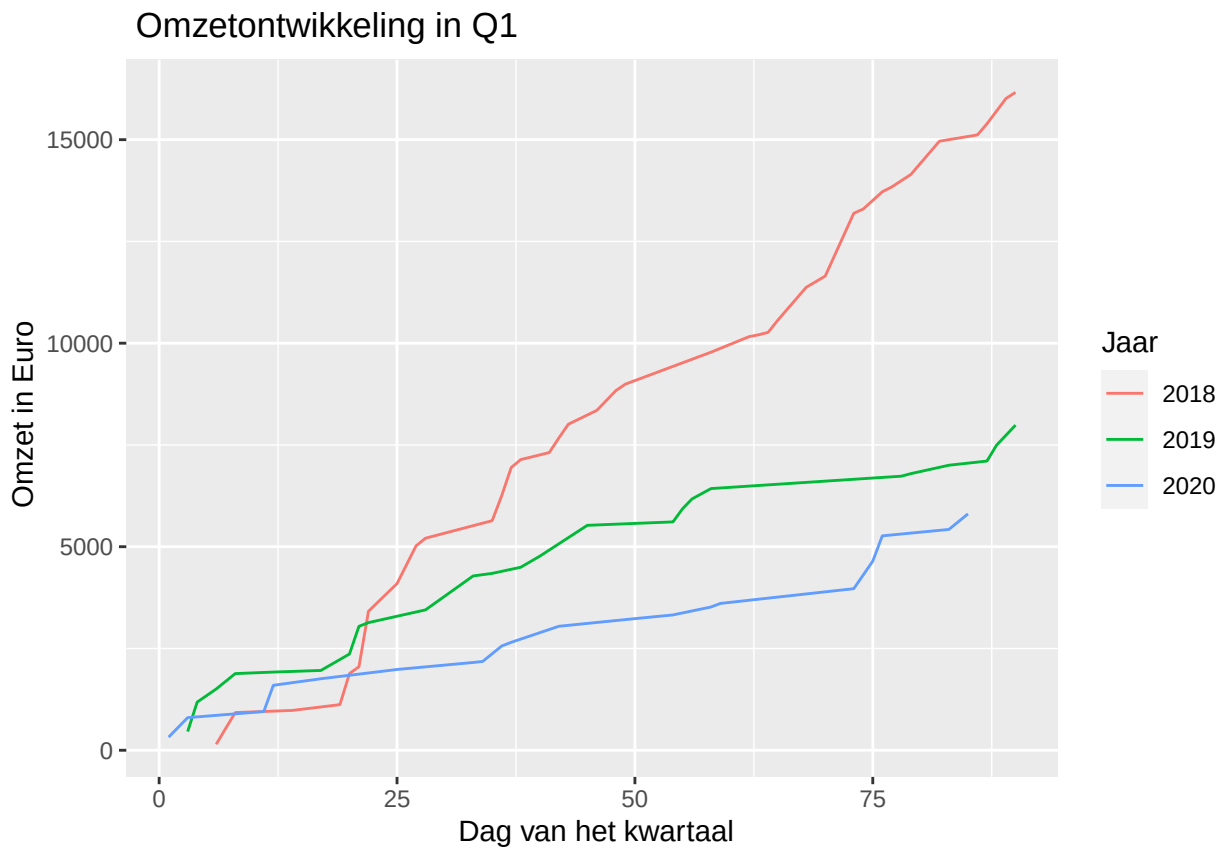
Aanpak

- Alleen data uit Q1 behouden
- Dag van het kwartaal toevoegen mbv `qday`
- Per dag (per jaar) de omzet optellen
- Visualiseren met jaar als kleur

```
## `summarise()` has grouped output by 'Jaar'. You can override using the `.groups` argument.
```

```
## # A tibble: 6 x 5
## # Groups:   Jaar [1]
##    Jaar besteldatum omzet runningomzet DayOfQuarter
##   <dbl> <date>      <dbl>        <dbl>        <dbl>
## 1  2018 2018-01-06  146.          146.            6
## 2  2018 2018-01-07  393.          539.            7
## 3  2018 2018-01-08  387           926.            8
## 4  2018 2018-01-14   52.5         978.           14
## 5  2018 2018-01-19  142.         1121.           19
## 6  2018 2018-01-20  763.         1884.           20
```

Nu kunnen we vrij eenvoudig de ontwikkeling visualiseren mbv een line chart



Vraag 2 : Vergelijken verkoopaantallen productcategorien 40x per seizoen

We definieren onze seizoenen als volgt: - Hoogseizoen: zomertijd dagen - Laagseizoen: wintertijd dagen

Aanpak

- Filteren van artikelnrs
- Bepalen wintertijd/zomertijd bij een verkoopregel
- Toevoegen seizoen obv zomer/wintertijd
- Visualiseren

Let op : het soort datum conversie dat je doet bepaalt wat je met de dataum kan. - as.Date kent geen tijdzones, as.POSIXct wel. we moeten de besteldatum dus even omzetten - om een TRUE/FALSE (boolean/logical) te converteren naar een character, gebruiken we een mutate icm een if_else: `mutate(seizoen = if_else(is_zomertijd, "zomertijd","wintertijd"))`

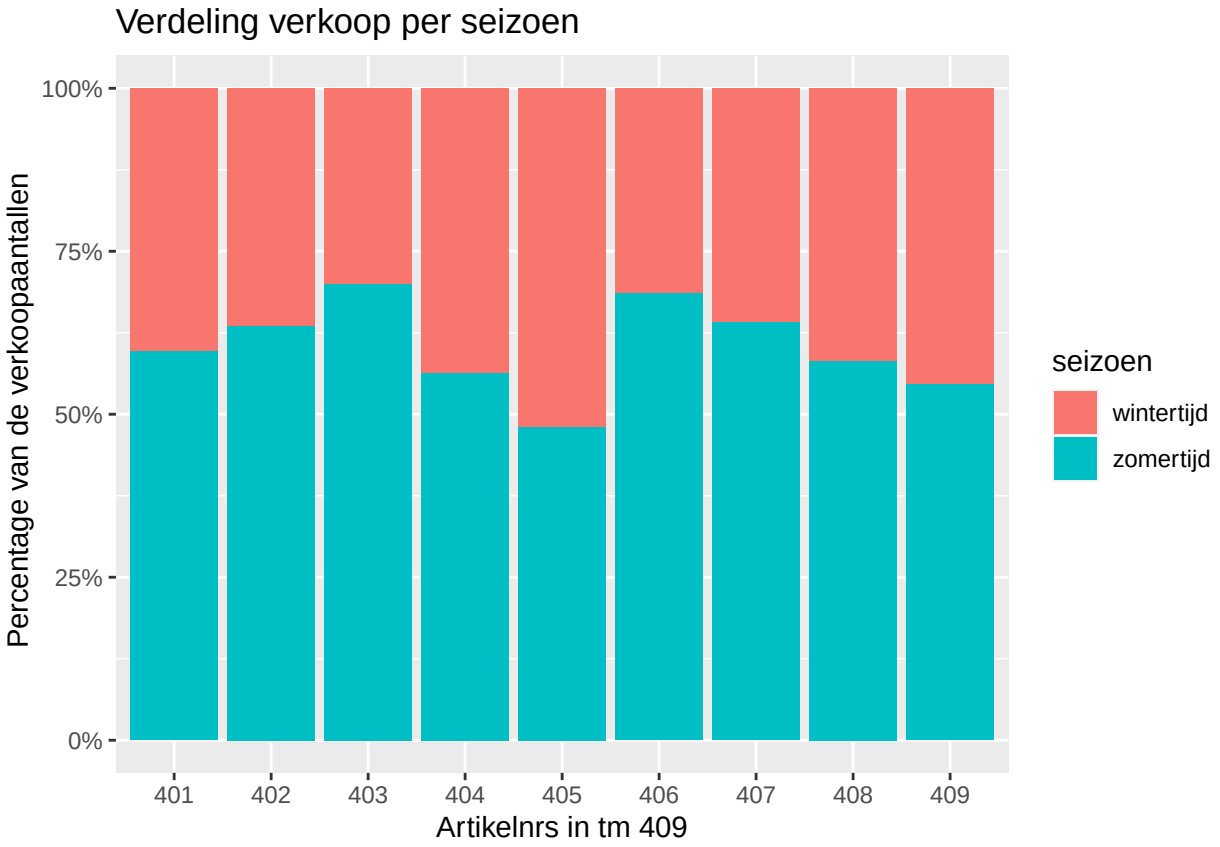
```
## `summarise()` has grouped output by 'artikelnr'. You can override using the `.groups` argument.
```

```
## # A tibble: 6 x 4
## # Groups:   artikelnr [5]
##   artikelnr is_zomertijd tot_aantal seizoen
##   <fct>     <lgl>             <int> <chr>
## 1 407       TRUE                489 zomertijd
## 2 401       TRUE                446 zomertijd
```

```
## 3 406       TRUE              443 zomertijd
## 4 405       FALSE             411 wintertijd
## 5 402       TRUE              395 zomertijd
## 6 405       TRUE              380 zomertijd
```

Per product kunnen we nu kijken of er meer van wordt verkocht in de zomer of winter. We zijn dus geinteresseerd in de verhoudingen.

let op: artikelnummer wordt als numerieke waarde gezien, terwijl het eigenlijk een categorie (factor) is. daar moet je dmv een `as.factor` rekening mee houden.



Vraag 3 : Tijd tussen bestellingen per klant (top 10)

Als laatste vraag voor deze oefening willen we eens onderzoek doen naar de tijd tussen bestellingen, bijvoorbeeld per klant, of gerelateerd aan aantal bestelingen. Als we de tijd tussen bestellingen weten, kunnen we daarmee beter en tijdiger deze kalnten benaderen met aanbiedingen of anticiperen met onze eigen inkoop.

Stap 1 - tijd tussen bestellingen

- We kijken naar bestelnr's en hebben dus gegevens per bestelling nodig. handig zijn: orderwaarde, aantal verschillende artikelen, dagen sinds vorige bestelling
- We moeten dus eerst groeperen en een aantal optellingen & berekeningen uitvoeren. Door te sorteren op besteldatum kunnen we dmv de `lag` functie naar de vorige rij kijken en de besteldatums met elkaar vergelijken.

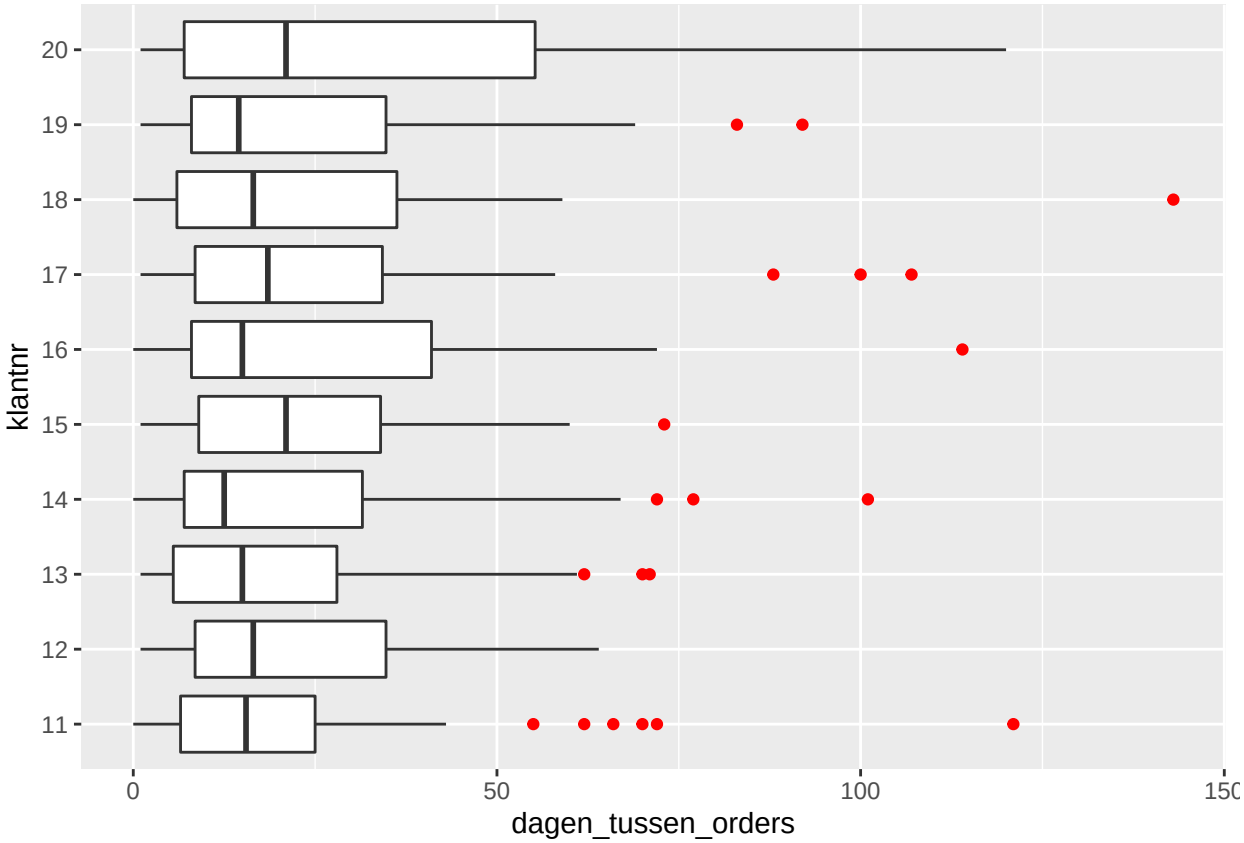
```
## `summarise()` has grouped output by 'bestelnr', 'klantnr'. You can override using the `.groups` argu
```

```
## # A tibble: 6 x 5
## # Groups:   klantnr [1]
##   bestelnr klantnr besteldatum aantal dagen_tussen_orders
##   <fct>    <fct>   <date>       <int>               <int>
## 1 1006     11      2018-01-20       1                  NA
## 2 1015     11      2018-01-28       1                   8
## 3 1018     11      2018-02-05       1                   8
## 4 1021     11      2018-02-06       1                   1
## 5 1024     11      2018-02-11       1                   5
## 6 1025     11      2018-02-12       1                   1
```

Stap 2 - Data verkennen

Nu we de tijd tussen bestellingen hebben berekend, kunnen we eens gaan kijken hoe die verschillen verdeeld zijn. Dat kan bv per klant dmv een boxplot op klant.

```
## Warning: Removed 10 rows containing non-finite values (stat_boxplot).
```



Conclusie: Klant 11 heeft best een aantal outliers (lange tijd tussen bestellingen, bij klant 20 is de spreiding het grootst). Klant 14 en 19 hebben de kortste besteldurен

Vraag 4 - Omzet per dag van de week

Als laatste vraag zijn we nog benieuwd naar de verdeling van de omzet naar dag van de week. zijn er weekdagen waarop er voor meer wordt besteld? Daarmee kunnen we de inkoop beter afstemmen.

Aanpak

- Dag van de week bepalen obv besteldatum
- Per bestelling de omzet berekenen
- Omzet per klant per dag van de week berekenen en visualiseren

```
## `summarise()` has grouped output by 'Dag'. You can override using the `.groups` argument.
```

```
## # A tibble: 6 x 3
## # Groups:   Dag [1]
##   Dag    klantnr order_omzet
##   <fct>  <fct>         <dbl>
## 1 Friday 11            1485.
## 2 Friday 12            2036.
## 3 Friday 13            2113.
## 4 Friday 14            2001.
## 5 Friday 15             900.
## 6 Friday 16            1240.
```

Dit kunnen we mooi visualiseren met een stacked bar