Refresher data prep & interactive data visualiatie

Trainee Sessie 01

Longhow Lam

Contents

1 The tidyverse

Het `tidyverse` package is een verzameling R packages met 'dezelfde' gedachtegoed om data te bewerken / manipuleren. Installeer tidyverse en attach de library zodat je in 1 keer een aantal libraries laadt:

- dplyr
- tibble
- ggplot2
- purrr
- forcats
- readr
- readxl
- stringr
- tidyr

Merk op: Het is soms niet zo handig om een hele lijst aan libraries aan je R sessie te koppelen. In bovenstaande lijst van packages hebben we nu meerdere packages met de functie `select`.

```
getAnywhere("select")
```

Een package kan je detatchen van je huidige sessie met het commando:

```
detach(package:raster)
detach(package:plotly)
```

2 Data importeren en exporteren

2.1 CSV files

2.2 RDS files

2.3 feather format

3 De dplyr library

Dit package / library is ontzettend handig om data te bewerken in R. De syntax is elegant en dplyr code kan je in sommige gevallen ook gebruiken voor data die niet in R zit. Bijvoorbeeld, data in Spark kan je met dplyr code bewerken. Er zijn veel 'sleutel' woorden, maar de belangrijkste zijn `select`, `filter`, `mutate`, `arrange`, `summarize`, `slice` en `rename`.

Het mooie is dat deze sleutel woorden ook achter elkaar gebruikt kunnen worden met de `%>%` chain operator. We gebruiken als voorbeeld data uit de datasets library. `mtcars` is een klein data set met wat auto gegevens en `iris` is de bekende data set met drie soorten bloemen.

3.1 kolommen selecteren met select

```
library(dplyr)

## enkele kolommen selecteren
```

```r
my.cars = select(mtcars, hp, mpg, drat)

## via de chain operator, dit is niet alleen handig maar zo kan je verschillende operaties achter elkaa
my.cars = mtcars %>%
  select(
    hp,
    mpg,
    drat
  )

## meerdere kolommen selecteren
test1 = iris %>%
  select(
    starts_with("Petal")
  )

## bepaalde kolommen selecteren
test2 = mtcars %>%
  select(
    contains("pg"),
    starts_with("c")
  )

## kolommen selecteren op basis van positie
my.cars = mtcars %>% .[2:6]
```

3.2 kolom maken of wijzigen met mutate

We zagen al dat je kolommen in een data frame kon maken of wijzigen met $

```r
my.cars$kolom2  = rnorm(32)
my.cars$kolom3  = my.cars$kolom2 + 9
```

Maar met `mutate` kan je dit veel eleganter doen

```r
mycars = mtcars %>%
  select(
    disp,
    mpg
  ) %>%
  mutate(
    displ_l = disp / 61.0237,
    tmp = mpg *1000
  )
```

Je kan nieuwe kolommen maken die afhankelijk zijn van kolommen die je net in mutate maakt.

```r
mycars = mtcars %>%
  mutate(
    displ_l = disp / 61.0237,
    tmp = mpg *1000,
    tmp2 = tmp + 1
  )
```

Je kan ook zelf gemaakte functies gebruiken in mutate.

```
normalize <- function(x){
    return((x- mean(x))/(max(x)-min(x)))
}

# test run van de functie
normalize(1:10)

# in mutate
mycars = mtcars %>%
  mutate(
    cyln = normalize(cyl)
  )
```

Twee functies die ontzettend handig kunnen zijn: `mutate_at` en `mutate_if`.

```
ABT = data.frame(
  matrix(round(100*runif(100)), ncol=10)
)

## maak eerst een kolom die de som is van de de 10 kolommen
ABT = ABT %>% mutate(sum = rowSums(.[1:10]))

## verander alleen kolom 4 t/m 8, ze worden door sum gedeeld
ABT2 = ABT %>%
  mutate_at(
    4:8, funs(. / sum)
  )

## deel alleen door de sum als de kolom som groter is dan 500
LARGE = function(x) {sum(x) > 500}
ABT3 = ABT %>%
  mutate_if(
    LARGE, funs(. / sum)
  )

ABT
ABT2
ABT3
```

3.3 distinct rows

Soms wil je data ontdubbelen, dit kan je in R doen met de functie `distinct`

```
distinct( select(mtcars, carb, gear))

distinct(select(mtcars, 10,11))

### pipe / chain operatie
myset = mtcars %>%
  select(10,11) %>%
  distinct
```

3.4 data sets filteren

Met `filter` en `slice` kan je rijen uit een data frame halen.

Met `slice` kan je op basis van row numbers data selecteren. Als voorbeeld, de eerste 8 records, of record 8 tot en met de laatste record.

```
slice(mtcars, 1:8)
slice(mtcars, 8:n())
```

3.5 data sets aggregeren

Aggregeren met de functie `group_by` en `summarise` deze gaan vaak hand in hand samen.

```
## apart aanroep van group_by en summarise... zo kan je zien wat group_by oplevert
by_cyl = group_by(mtcars, cyl)
class(by_cyl)

summarise(by_cyl, Naampje1 = mean(disp), Naampje2 = mean(hp))

## maar vaak doe je de twee in 1 run samen
out = mtcars %>%
  group_by(
    cyl
  ) %>%
  summarise(
    mdisp = mean(disp),
    mhp = mean(hp)
  )


out = filter(
  mtcars, mpg > 11
  ) %>%
  group_by(
    cyl,
    carb
  ) %>%
  summarise(
    N = n(),
    MeanDisp = mean(disp),
    SD_HP = sd(hp)
  )
```

3.6 joins

Met dplyr kun je makkelijk tabellen joinen. Er zijn verschillende joins die ondersteunt worden. We geven hier een aantal voorbeelden.

```
### maak twee data frames aan
df1 = data.frame(
  col1 = c(1,2,3,4,5), tt = rnorm(5)
)
```

```r
df2 = data.frame(
  col1 = c(3,4,5,6,7), xx = rnorm(5), zz = runif(5)
)

## selecteer rijen die zowel in df1 en df2 zitten
df3 = inner_join(
  df1,
  df2,
  by = c("col1" = "col1")
)

## alle rijen die in df1 zitten, geen match levert NA op
df4 = left_join(
  df1,
  df2,
  by = c("col1" = "col1")
)

## alle rijen die in df2 zitten
df5 = df1 %>% right_join(
  df2,
  by = c("col1" = "col1")
)

## alleen de rijen van df1 die niet in df 2 zitten
df6 = df1 %>% anti_join(
  df2,
  by = c("col1" = "col1")
)

## alle rijen van df1 en df2
full_join(df1, df2, by = c("col1" = "col1"))
```

3.7 data frames stapelen of plakken

Soms wil je twee tabellen op elkaar stapelen tot een nieuwe tabel, dat kan je doen met `bind_rows`. Als je twee tabellen naast elkaar wilt zetten tot een nieuwe tabel, gebruik dan `bind_rows`.

```r
## bind_rows
A = data.frame(x=1:5, y = rnorm(5))
B = data.frame(x=11:15, y = rnorm(5))
C = bind_rows(A,B)

## kolommen die niet in een van de data frames zitten worden aangevuld
E = data.frame(x=21:25, y = rnorm(5), z = rnorm(5))

bind_rows(A,E)
```

```r
A = data.frame(x=1:5, y = rnorm(5))
B = data.frame(q=11:15, q = rnorm(5))
bind_cols(A,B)

## als een van de data frames meer rijen heeft, dan wordt er NIET aangevuld
```

```r
A = data.frame(x=1:5, y = rnorm(5))
B = data.frame(q=11:17, q = rnorm(7))
bind_cols(A,B)
```

4 Tidy data en wide data

Tidy data is wanneer data in de volgende vorm zit:

- Elke variabele is in een kolom.
- Elke observatie is een rij.
- Elke waarde is een cel.

Stel we hebben de volgende data set

```r
library(tidyr)
stocks <- data.frame(
  time = as.Date('2009-01-01') + 0:9,
  X = rnorm(10, 0, 1),
  Y = rnorm(10, 0, 2),
  Z = rnorm(10, 0, 4)
)
stocks
```

Soms is het handig om data waarden in 1 kolom te hebben en een aparte variabele die de variabele weergeeft

```r
stocksm <- stocks %>% gather(stock, price, -time)
stocksm
```

en spread is het omgekeerde proces

```r
stock2 = stocksm %>% spread(stock, price)
stock2

## als er niet even veel observaties zijn?

test = data.frame(
  T = c(1,2,3,1,2),
  V = c("A","A","A","B","B"),
  price = c(4,5,6,7,8)
)

test2 = test %>% spread(V,price)
test2
```

Een handige functie in tidyr is `separate`, dit kan je gebruiken om een kolom uit elkaar te trekken. Standaard worden niet alfa numerieke tekens als separator gebruikt.

```r
df = tibble(x = c("A.B", "C.P", "P.Q"))
df %>% separate(x, c("Kol1", "Kol2"))

## Er kunnen tw weinig of teveel kolommen zijn.
df = tibble(x = c("A.B", "C.P", "P.Q", "pp", "P.2.4"))
df %>% separate(x, c("Kol1", "Kol2"))
```

```
df = tibble(x = c("A.B", "C.P", "P.Q", "pp", "P.2.4"))
df %>% separate(x, c("Kol1", "Kol2", "Kol3"))
```

5 Character data bewerken met het package `stringr`.

Character data in R kan je bewerken/manipuleren met het `stringr` package. Voordat we daar verder op in gaan is het handig om te weten wat reguliere expressies zijn.

5.1 Reguliere expressies

Dit is een soort 'mini' taaltje om character patronen te specificeren om er vervolgens bijvoorbeeld op te zoeken. Eerst wat dummy character data.

```
test = c(
  "Mijn nummer is 06-12345678",
  "dit is .. een 1628EP postcode test",
  "foutje?:  0234XD",
  "dit is er nog een 1628 EP",
  "en nog een foute 126EP",
  "nog een 1234    XX",
  "1234eZ en nog 4567PK",
  "12345 Aabcde", "&yfy."
)
test
```

Een paar voorbeelden van reguliere expressies.

```
library(stringr)

## een digit, en precies 1 digit
patroon = "\\d"
str_extract(test, patroon)

## 1 of meerdere digits
patroon = "\\d+"
str_extract(test, patroon)

## precies twee letters
patroon ="[[:alpha:]]{2}"
str_extract(test, patroon)

## Een postcode
patroon = "\\d{4}\\s?[[:alpha:]]{2}"
str_extract(test, patroon)

## Maar een postcode begint niet met een 0
patroon = "[1-9]\\d{3}\\s?[[:alpha:]]{2}"
str_extract(test, patroon)

## special punctuation characters !"#$%&'()*+,-./:;<=>?@[]^_`{|}~
```

```
patroon = "[[:punct:]]"
str_extract(test, patroon)
```

Sommige mensen zijn de wildcard notatie gewend om te zoeken in strings, deze wildcard notatie kan je vertalen naar reguliere expressies met `glob2rx`. Een paar voorbeelden zullen hieronder geven.

```
patroon = glob2rx("*23*")
str_extract(test, patroon)
```

Voor een cheatsheet over reguliere expressies zie regex cheatsheet

5.2 functies uit het package stringr

In het package `stringr` zijn diverse functies om met characters te werken, een aantal zullen we laten zien

```
testset = titanic::titanic_train %>% select(Name)
## maak extra kolom aan in titanic die namen zoekt
testset = testset %>%
  mutate(
    naamlengte = str_length(Name),
    Ownes = str_detect(Name, 'Owen'),
    Plek = str_locate(testset$Name, "Mr") %>% .[,1],
    Name2 = str_replace(Name, "Mr", "")
  )

str_locate(testset$Name, "Mr")
```

6 Factor maniuplatie met forcats

In R kun je factor variabelen bewerken met functies uit het `forcats` package. We geven een aantal voorbeelden op random gegenereerde data

```
library(forcats)

x = rnorm(1000)
y = sample(LETTERS[1:10], replace = TRUE, size = 1000, prob = (1:10)^2)

test = tibble(gewicht = x, type = y)
table(test$type)
```

Samenvoegen van levels: Op aantallen (Weinig voorkomende levels) of handmatig

```
test = test %>%
  mutate(
    z = fct_lump(type, n=5)
  )

table(test$z)

test = test %>%
  mutate(
    z2 = fct_other(type, keep = c("C","B")),
```

```
    z3 = fct_other(type, drop = c("A","B"))
  )

table(test$z2)
table(test$z3)
```

Hernoemen van factor levels

```
test = test %>%
  mutate(
    z2 = fct_recode(
      type,
      ZZ = "A",
      ZZ = "B",
      YY = "C",
      YY = "D"
    ),
    z3 = fct_collapse(
      type,
      missing = c("B", "C"),
      other = "D",
      rep = c("E", "F"),
      ind = c("G","H"),
      dem = c("I","J")
    )
  )
table(test$z2)
table(test$z3)
```

Herordenen van factor levels, factor levels kunnen een order bevatten, deze kan je maken of veranderen, kan soms nodig zijn bij plotten.

```
test = test %>%
  mutate(
    type2 = fct_reorder(type, gewicht, mean)
  )

table(test$type)
table(test$type2)

levels(test$type)
levels(test$type2)

## hier een voorbeeld van het verschil in plaatjes

library(ggplot2)

test2 = test %>%
  group_by(type) %>%
  summarise( gewicht = mean(gewicht))

ggplot(test2, aes(x=type)) + geom_bar(aes(weight=gewicht))

test2 = test %>%
  group_by(type2) %>%
```

```
  summarise( gewicht = mean(gewicht))

ggplot(test2, aes(x=type2)) + geom_bar(aes(weight=gewicht))
```

7 Dates and times in R

Voor datums (en tijden) in R zijn de packages `anytime` en `lubridate` heel handig. Laten we er van uitgaan dat we characters hebben met datums en daar mee willen rekenen. Dan kunnen we de functie `anydate` uit het anytime package gebruiken.

```
library(lubridate)
library(anytime)

y = c("1973-09-12", "1980-05-23", "1981-12-09")

testdata = tibble(DoB = y)

testdata = testdata %>%
  mutate(
    GeboorteDatum = anytime::anydate(DoB)
  )
```

De functie `anydate` parsed diverse character notatie naar een date kolom, de gaat meestal goed :-)

```
y = c("1973/09/12", "05/23/1980", "23/05/1980","1981-12-09")

testdata = tibble(DoB = y)

testdata = testdata %>%
  mutate(
    GeboorteDatum = anytime::anydate(DoB)
  )
```

Nu we een datum echt als date column in een data frame of tibble hebben kunnen we er mee rekenen.

```
testdata = testdata %>%
  mutate(
    leeftijd = today() - GeboorteDatum,
    leeftijd2 = as.numeric(today() - GeboorteDatum)/365,
    dag = wday(GeboorteDatum, label=TRUE)
  )
```

UNIX time stamps.... Wat zeg je?

Een Unix time stamps is het aantal seconden sinds 1-1-1970. Mocht je dit ooit tegen komen dan kan anytime er ook mee uit de voeten.

```
library(readr)
tmp = read_csv("data/unixtimestamps.csv")
tmp$datum = anydate(tmp$timestamp)
tmp
```

Tijdstippen kunnen in R ook worden weergegven met times

```
tmp = paste("2018-01-23", c("12:01:33", "12:01:38", "12:01:58"))
Waarde = c(2.4, 5.6, 7.8)

MeetData = tibble(tmp = tmp, Waarde= Waarde)

MeetData = MeetData %>%
  mutate(
    Meettijdstip = ymd_hms(tmp)
  )
```

7.1 durations en intervals

Als je in R twee datums/times van elkaar aftrekt krijg je in R een duration

```
MeetData = MeetData %>%
  mutate(
    duur = ymd_hms("2018-02-01 00:00:00") - ymd_hms(tmp)
  )
```

Ja kan durations ook zelf maken en optellen bij een datum

```
testduur = dweeks(3)
testduur = dweeks(3.1)
class(testduur)

MeetData = MeetData %>%
  mutate(
   t2 = Meettijdstip + dyears(2),
   t3 = Meettijdstip + dseconds(1800),
   t4 = Meettijdstip + dweeks(3)

  )
```

In R heb je ook zogenaamde `interval` objecten, dit is de tijdsperiode tussen een start en eind datum. Hiermee kan je de leeftijd in jaren ook precies uitrekenen.

```
y = c("1973-09-12", "1973-01-09")

testdata = tibble(DoB = y)

testdata = testdata %>%
  mutate(
    GeboorteDatum = anydate(DoB),
    interval1 = interval(GeboorteDatum, today()),
    interval2 = GeboorteDatum %--% today(),
    leeftijd = interval1 %/% years(1)
  )

testdata

# NB %/% is de integer division in R
13 / 6
13 %/% 6
```

8 dplyr met externe data

Je kan `dplyr` niet alleen gebruiken op data frames in R, ook kan je `dplyr` gebruiken op externe data in databases en, ook data in Spark. We geven hier eerst het Spark voorbeeld met het package `sparklyr`, en dan een MySQL voorbeeldje.

```
library(sparklyr)
library(dplyr)

## maak een connectie naar spark
sc <- spark_connect(master = "local")

## nu zie je ook in RStudio een Spark connection met spark data sets
## en een link naar de spark User Interface

## upload data naar spark, zie ook connections tab
iris_tbl <- copy_to(sc, iris)

## nu kan je dplyr gebruiken op spark
test = iris_tbl %>% filter(Sepal_Length >5)

class(test)
test %>% count()
```

Als je klaar bent: disconnect van Spark

```
spark_disconnect(sc)
```

Als je MySQL hebt en een odbc connectie hebt opgezet kan je een link leggen naar MySQL. Zie bijvoorbeeld ook deze link

```
library(odbc)
library(DBI)
library(dplyr)

## maak een connectie naar mySQL, zie ook connections tab
con_sql <-  dbConnect(odbc::odbc(), "my-connector")

## laat zien welek tabelllen er zijn
dbListTables(con_sql)

## copier lokale R data frames naar MySQL
dbWriteTable(conn = con_sql, name = 'mtcars_test', value = mtcars)

## je hebt een'handle' nodig naar een tabel, daar kan je dplyr op gebruiken
tbl_handle = tbl(con_sql, "mtcars")

tbl_handle %>% count()


zz = tbl_handle %>% filter(disp > 260)
class(zz)

lokaledata = zz %>% collect()
```

Als je klaar bent: disconnect van MySQL

```
dbDisconnect(con_sql)
```

9 Interactive grafieken

Naast de statistische grafieken die we hierboven hebben gemaakt kan je in R ook interactieve grafieken maken. We geven in deze sectie een aantal voorbeelden.

9.1 Het plotly package

Plotly is een zeer mooie library voor het maken van interactieve visualisaties. Zie voor meer info plotly. We zullen een aantal voorbeelden hieronder laten zien.

9.1.1 scatters

```
library(plotly)
mtcars = mtcars
mtcars$naampjes = row.names(mtcars)

plot_ly(data = mtcars, x=~mpg, y = ~wt)
plot_ly(data = mtcars, x=~mpg, y = ~wt, color = ~cyl)
plot_ly(data = mtcars, x=~mpg, y = ~wt, text = ~naampjes)
```

9.1.2 barcharts

We gebruiken de restaurants data set weer

```
Restaurants <- readr::read_csv("data/Restaurants.csv")
keuken = Restaurants %>% dplyr::group_by(keuken) %>% dplyr::summarise(n=n())

p = plot_ly(data = keuken,
  x = ~keuken,
  y = ~n,
  type = "bar"
)

p
```

Sorteren op volgorde van aantallen kan je doen door reorder factor

```
keuken = keuken %>%
  mutate(
  keuken = forcats::fct_reorder(keuken,n, .desc=TRUE)
)
p = plot_ly(data = keuken,
  x = ~keuken,
  y = ~n,
  type = "bar"
)
```

```
p
```

9.1.3 boxplots

```
p = plot_ly(data = Restaurants, y = ~aantalreviews, color = ~keuken,type = "box")
p
```

9.1.4 3D plots

If you have 3D data you can plot it in a 3D scatter plot

```
p = plot_ly(data = mtcars, x=~mpg, y=~cyl, z = ~disp)
p
```

A matrix with values can be plotted as a 3D surface plot

```
volcano
plot_ly(z = ~volcano) %>% add_surface()
```

9.1.5 ggplot integratie

Je kan een grafiek maken met ggplot en dan de functie ‘`ggplotly` gebruiken om de grafiek interactief te maken. Als de grafiek veel punten bevat kan het sloom over komen. Vandaar dat onderstaande voorbeeldje alleen 3000 punten sampled.

```
p = ggplot(
  data = diamonds %>% dplyr::sample_n(3000),
  aes(x = carat, y = price)
) +
  geom_point(
    alpha = 0.05,
    aes(text = paste("Clarity:", clarity))
  ) +
  geom_smooth(aes(colour = cut, fill = cut)) +
  facet_wrap(~ cut)
p

ggplotly(p)
```

9.2 Kaartjes met leaflet

Leaflets worden gebruikt om interactieve kaartjes te maken. By deault gebruiken open street maps als achtergrond.

```
library(leaflet)

m <- leaflet() %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng=174.768, lat=-36.852, popup = "The birthplace of R")
m
```

```
# kan geblokt worden door fire walls.
```

Plot de restaurants in Zwolle op een kaartje. Data van Iens gescraped.

```
Restaurants <- read.csv("data/Restaurants.csv")
Zwolle = Restaurants %>% filter(plaats == "Zwolle")

## tooltip teksten kunnen geplaatst worden
ptekst = paste(Zwolle$restNamen, '<BR> aantal reviews: ', Zwolle$aantalreviews)

m2 = leaflet(data = Zwolle)
m2 = m2 %>%
  addTiles() %>%  # Add default OpenStreetMap map tiles
  addMarkers(lng = ~LONGs, lat = ~LATs, popup = ptekst)
m2
```

Je kan kleurtje gebruiken, gebruik dan `addCirclemarkers`

```
reds = colorNumeric("Reds", domain = NULL)

Zwolle = Zwolle %>% filter(!is.na(aantalreviews), aantalreviews < 80)
ptekst = paste(Zwolle$restNamen, '<BR> aantal reviews: ', Zwolle$aantalreviews)

m2 = leaflet(data = Zwolle)
m2 = m2 %>%
  addTiles() %>%
  addCircleMarkers(
    lng = ~LONGs,
    lat = ~LATs,
    fillColor = ~reds(aantalreviews),
    fillOpacity = 1, popup = ptekst
  )
m2
```

9.2.1 polygonen op een leaflet

Je kan ook vormen/polygons op een leaflet kaart krijgen. Hieronder is een voorbeeld van provincies, eerst moet je de data van de polygonen hebben. Dit staat in zogenaamde shape files van de provincies, deze kan je downloaden van http://www.imergis.nl/asp/47.asp. Op deze site zie je provinciegrenzen_exact, download deze en pak deze uit. In mijn data directory is dat al gedaan.

Eerst is er een beetje voorbewerking nodig om de shape file te bewerken in R tot een zogenaamde SpatialPolygonsDataframe. Dan kan je dit met leaflet te kunnen plotten.

```
#### inlezen shape file
ProvNL <- readShapeSpatial("data/TopGrenzen-prov-actueel.shp")
class(ProvNL)
summary(ProvNL)

#### Zet coordinatensysteem
proj4string(ProvNL) <- CRS("+init=epsg:28992")

#### transformeer naar long /lat
ProvNL = spTransform(ProvNL, CRS("+proj=longlat +datum=WGS84"))
```

```
#### Het object ProvNL is een zgn spatialpolygons object, daar zit een data frame in
ProvNL@data

#### maak een hele simpele plot
plot(ProvNL)
```

Maar nu kun je het spatial polygons object ook op een leaflet plotten.

```
leaflet(ProvNL) %>%
  addTiles() %>%
  addPolygons(
    stroke = TRUE, weight = 1, fillOpacity = 0.15, smoothFactor = 0.15
  )
```

Je kan de kleuren van de provincies ook veranderen

```
ProvNL$Kleur = sample(c("red","blue"), 12, replace = TRUE)

ProvNL@data

leaflet(ProvNL) %>%
  addTiles() %>%
  addPolygons(
    stroke = TRUE, weight = 1, fillOpacity = 0.25, smoothFactor = 0.15, fillColor = ~Kleur
  )
```

** Kleuren Schemas **

De kleuren kan je ook data gedreven met een kleuren schema bepalen.

```
#### maak eerst wat data voor de provincies
ProvNL$Leeftijd = runif(12)

### definieer een 3 niveau schema
colpal <- colorQuantile(
  palette = green2red(3), n=3,
  domain = ProvNL$Leeftijd
)

### gebruik kleurtjes en een popup tekst
leaflet(ProvNL) %>%
  addTiles() %>%
  addPolygons(
    stroke = TRUE,
    weight = 1,
    fillOpacity = 0.25,
    smoothFactor = 0.15,
    fillColor = colpal(ProvNL$Leeftijd),
    popup = as.character(ProvNL$Leeftijd)
  )
```

Hieronder is een voorbeeld om op postcode 2 posities niveau een kaart te maken. We hebben eerst de data nodig die de contouren van een PC2 gebied definieert, en dan is er wat preparatie nodig om de data in polygoon objecten te krijgen. Deze gaat iets anders dan de provincies die in 1 shape file stonden.

```
# importeer data uit een CSV file
pc2_kaart <- read_csv("data/pc2_kaart.csv")
```

```
PC2_data = dplyr::select(
  pc2_kaart,
  Longitude,
  Latitude,
  PC2CODE ) %>%
  as.data.frame()

# Het moet een lijst van gebieden zijn.
Pc2_list = split(PC2_data, PC2_data$PC2CODE)
Pc2_list = lapply(Pc2_list, function(x) { x["PC2CODE"] <- NULL; x })


### Maak van elk lijst object een Polygon
plg = sapply(Pc2_list, Polygon)

plg[[1]]
```

Nu gaan we per PC2 gebied het aantal restaurants tellen en dit in een kaartje zetten.

```
PC2 = Restaurants %>%
  mutate(
    PC2 = stringr::str_sub(PCs,1,2)
  ) %>%
  group_by(PC2) %>%
  summarise(n=n()) %>%
  filter(PC2 >= "10")

## initieer de leaflet en voeg PC2 polygon 1 voor 1 toe

pc2_lf = leaflet() %>%  addTiles()

## definieer een kleuren schema
colpal <- colorQuantile(
  palette = green2red(7), n=7,
  domain = PC2$n
)

## elk polygoontje krijgt nu op basis van zijn restaurant aantal het 'juiste' kleurtje

### teken per polygon, dat is het makkelijkst
for (i in 1:length(plg)){
  ptxt = paste(
    "PC2: ", as.character(PC2$PC2[i]),
    "<br>",
    "Aantal Restaurants",
    as.character(PC2$n[i])
  )
  pc2_lf = pc2_lf %>%
    addPolygons(
      data = plg[[i]],
      weight = 2,
      smoothFactor = 1,
      fillOpacity = 0.55,
      fillColor= colpal(PC2$n[i]),
```

```
      popup = ptxt
    )
}

pc2_lf
```

popup texts in leaflets kunnen ook images en animated animated images bevatten. See as an example this leaflet of Amsterdam en this shiny app

9.3 Netwerk grafieken, visnetwork

Het package `visnetwork` kan worden gebruikt om interactieve netwerk grafieken te maken. Je hebt twee data frames nodig, een met de nodes en een met de edges. Laten we beginnen met een simpel voorbeeldje.

```
library(visNetwork)

nodes = data.frame(
  id = c(1,2,3,4)
)
edges = data.frame(
  from = c(1,3,4),
  to  = c(2,2,2)
)

visNetwork(nodes, edges)
```

Er zijn diverse opties die je kan meegeven. Een deel van de opties kan je via de data frames meegeven.

```
nodes <- data.frame(
  id = 1:10,
  label = paste("Node", 1:10),                                  # labels
  group = c("GrA", "GrB"),                                      # groups
  value = 1:10,                                                 # size
  shape = c("square", "triangle", "box", "circle", "dot", "star", "ellipse", "database", "text", "diamon
  title = paste0("<p><b>", 1:10,"</b><br>Node !</p>"),          # tooltip
  color = c("darkred", "grey", "orange", "darkblue", "purple"),  # color
  shadow = c(FALSE, TRUE, FALSE, TRUE, TRUE))                   # shadow

edges <- data.frame(from = sample(1:10,8), to = sample(1:10, 8),
  label = paste("Edge", 1:8),                                  # labels
  length = c(100,500),                                         # length
  arrows = c("to", "from", "middle", "middle;to"),             # arrows
  dashes = c(TRUE, FALSE),                                     # dashes
  title = paste("Edge", 1:8),                                  # tooltip
  smooth = c(FALSE, TRUE),                                     # smooth
  shadow = c(FALSE, TRUE, FALSE, TRUE))                        # shadow

visNetwork(nodes, edges)
```

Andere opties kan je via de functie `visnetwork` meegeven. Bijvoorbeeld highlight nearest

```
nodes <- data.frame(
  id = 1:15,
  label = paste("Label", 1:15),
  group = sample(LETTERS[1:3], 15, replace = TRUE)
```

```
)

edges <- data.frame(
  from = trunc(runif(15)*(15-1))+1,
  to = trunc(runif(15)*(15-1))+1
)

visNetwork(nodes, edges) %>%
  visOptions(
    highlightNearest = TRUE,
    nodesIdSelection = TRUE
  )
```

Nog een netwerk voorbeeldje van restaurant bezoekers in het mooie stadje Hoorn. Bij het scrapen van Iens, kan je ook zien dat een reviewer eerst een bepaald restaurant heeft gereviewed en dan een ander, dat is in een netwerk graph te zetten.

Focus nu alleen op het mooie stadje Hoorn.

```
Hoornnodes = readRDS("data/HoornNodes.RDs")
HoornEdges = readRDS("data/HoornEdges.RDs")

visNetwork(Hoornnodes, HoornEdges) %>%
  visLegend() %>%
  visOptions(
    highlightNearest = TRUE,
    nodesIdSelection = TRUE
  ) %>%
  visInteraction(
    navigationButtons = TRUE
  ) %>%
  visPhysics( maxVelocity = 25)
```

10 Zoomable circle packing plots, sunburst plots en chord diagrams

Circle packings zijn een leuke manier om hierarchische data te visualiseren. Installeer eerst het package van Github.

```
devtools::install_github("jeromefroe/circlepackeR")
```

Maak wat dummy data aan. Volgend voorbeeldje is een hierarchische lijst.

```
hierarchical_list <- list(
  name = "World",
  children = list(
    list(name = "North America",
         children = list(
           list(name = "United States", size = 308865000),
           list(name = "Mexico", size = 107550697),
           list(name = "Canada", size = 34033000))),
    list(name = "South America",
         children = list(
           list(name = "Brazil", size = 192612000),
```

```
          list(name = "Colombia", size = 45349000),
          list(name = "Argentina", size = 40134425))),
    list(name = "Europe",
         children = list(
          list(name = "Germany", size = 81757600),
          list(name = "France", size = 65447374),
          list(name = "United Kingdom", size = 62041708))),
    list(name = "Africa",
         children = list(
          list(name = "Nigeria", size = 154729000),
          list(name = "Ethiopia", size = 79221000),
          list(name = "Egypt", size = 77979000))),
    list(name = "Asia",
         children = list(
          list(name = "China", size = 1336335000),
          list(name = "India", size = 1178225000),
          list(name = "Indonesia", size = 231369500)))
  )
)
```

and plot it

```
circlepackeR::circlepackeR(hierarchical_list)
```

The above code is not convenient to plot data that are in data frames. For example click data or web path logs. Let's take the following data example from the traamap package.

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

```
library(circlepackeR)
library(data.tree)
library(treemap)

# Gross national income data
data(GNI2014)
GNI2014
```

```
##     iso3                        country              continent population
## 3    BMU                        Bermuda          North America      67837
## 4    NOR                         Norway                 Europe    4676305
## 5    QAT                          Qatar                   Asia     833285
## 6    CHE                    Switzerland                 Europe    7604467
## 7    MAC                Macao SAR, China                   Asia     559846
## 8    LUX                     Luxembourg                 Europe     491775
## 10   AUS                      Australia                Oceania   21262641
## 11   SWE                         Sweden                 Europe    9059651
```

```
## 12   DNK                    Denmark                 Europe    5500510
## 14   USA              United States           North America  313973000
## 15   SGP                  Singapore                   Asia    4657542
## 18   NLD                Netherlands                 Europe   16715999
## 19   CAN                     Canada           North America   33487208
## 20   AUT                    Austria                 Europe    8210281
## 21   KWT                     Kuwait                   Asia    2691158
## 22   FIN                    Finland                 Europe    5250275
## 23   DEU                    Germany                 Europe   82329758
## 24   ISL                    Iceland                 Europe     306694
## 25   BEL                    Belgium                 Europe   10414336
## 26   IRL                    Ireland                 Europe    4203200
## 27   ARE       United Arab Emirates                   Asia    4798491
## 28   GBR             United Kingdom                 Europe   62262000
## 29   FRA                     France                 Europe   64057792
## 30   JPN                      Japan                   Asia  127078679
## 31   ADO                    Andorra                 Europe      79218
## 32   NZL                New Zealand                Oceania    4213418
## 33   HKG       Hong Kong SAR, China                   Asia    7061200
## 34   BRN          Brunei Darussalam                   Asia     388190
## 35   ISR                     Israel                   Asia    7233701
## 36   ITA                      Italy                 Europe   58126212
## 37   ESP                      Spain                 Europe   40525002
## 38   KOR               Korea, Rep.                   Asia   48508972
## 39   CYP                     Cyprus                   Asia     531640
## 40   SAU               Saudi Arabia                   Asia   28686633
## 41   SVN                   Slovenia                 Europe    2005692
## 42   GRC                     Greece                 Europe   10737428
## 43   PRT                   Portugal                 Europe   10707924
## 44   BHR                    Bahrain                   Asia     727785
## 45   BHS             Bahamas, The           North America     309156
## 46   MLT                      Malta                 Europe     405165
## 47   TTO       Trinidad and Tobago           North America    1310000
## 48   PRI                Puerto Rico           North America    3971020
## 49   EST                    Estonia                 Europe    1299371
## 50   CZE             Czech Republic                 Europe   10211904
## 51   OMN                       Oman                   Asia    3418085
## 52   SVK           Slovak Republic                 Europe    5463046
## 53   URY                    Uruguay           South America    3494382
## 54   LTU                  Lithuania                 Europe    3555179
## 55   LVA                     Latvia                 Europe    2231503
## 56   BRB                   Barbados           North America     284589
## 57   KNA        St. Kitts and Nevis           North America      40131
## 58   CHL                      Chile           South America   16601707
## 59   SYC                 Seychelles Seven seas (open ocean)      87476
## 60   POL                     Poland                 Europe   38482919
## 61   ARG                  Argentina           South America   40913584
## 62   HUN                    Hungary                 Europe    9905596
## 63   ATG        Antigua and Barbuda           North America      85632
## 64   RUS         Russian Federation                 Europe  140041247
## 65   HRV                    Croatia                 Europe    4489409
## 66   VEN              Venezuela, RB           South America   26814843
## 67   KAZ                 Kazakhstan                   Asia   15399437
## 68   BRA                     Brazil           South America  198739269
```

```
## 69   PAN                          Panama           North America    3360474
## 70   MYS                        Malaysia                    Asia   25715819
## 71   PLW                           Palau                 Oceania      20796
## 72   TUR                          Turkey                    Asia   76805524
## 73   GNQ                   Equatorial Guinea                  Africa     650702
## 74   CRI                      Costa Rica           North America    4253877
## 75   LBN                         Lebanon                    Asia    4017095
## 76   SUR                        Suriname           South America     481267
## 77   MEX                          Mexico           North America  111211789
## 78   GAB                           Gabon                  Africa    1514993
## 79   MUS                       Mauritius Seven seas (open ocean)    1284264
## 80   ROU                         Romania                  Europe   22215421
## 81   TKM                      Turkmenistan                    Asia    4884887
## 82   COL                         Colombia           South America   45644023
## 83   GRD                          Grenada           North America      90739
## 84   LBY                            Libya                  Africa    6310434
## 85   BGR                         Bulgaria                  Europe    7204687
## 86   AZE                        Azerbaijan                    Asia    8238672
## 87   CHN                            China                    Asia 1338612970
## 88   BLR                          Belarus                  Europe    9648533
## 89   MNE                       Montenegro                  Europe     672180
## 90   LCA                        St. Lucia           North America     160267
## 91   BWA                         Botswana                  Africa    1990876
## 92   DMA                         Dominica           North America      72660
## 93   ZAF                     South Africa                  Africa   49052489
## 94   VCT St. Vincent and the Grenadines           North America     104574
## 95   IRQ                             Iraq                    Asia   31129225
## 96   MDV                         Maldives Seven seas (open ocean)     396334
## 97   PER                             Peru           South America   29546963
## 98   IRN                Iran, Islamic Rep.                    Asia   66429284
## 99   ECU                          Ecuador           South America   14573101
## 100  DOM               Dominican Republic           North America    9650054
## 101  SRB                           Serbia                  Europe    7379339
## 102  THA                         Thailand                    Asia   65905410
## 103  TUV                           Tuvalu                 Oceania      10640
## 104  NAM                          Namibia                  Africa    2108665
## 105  DZA                          Algeria                  Africa   34178188
## 106  JOR                           Jordan                    Asia    6342948
## 107  JAM                          Jamaica           North America    2825928
## 108  MKD                   Macedonia, FYR                  Europe    2066718
## 109  FJI                             Fiji                 Oceania     944720
## 110  BIH           Bosnia and Herzegovina                  Europe    4613414
## 111  BLZ                           Belize           North America     307899
## 112  ALB                          Albania                  Europe    3639453
## 113  PRY                         Paraguay           South America    6995655
## 114  MHL                Marshall Islands                 Oceania      64522
## 115  MNG                         Mongolia                    Asia    3041142
## 116  TON                            Tonga                 Oceania     120898
## 117  TUN                          Tunisia                  Africa   10486339
## 118  WSM                            Samoa                 Oceania     219998
## 119  ARM                          Armenia                    Asia    2967004
## 120  KSV                           Kosovo                  Europe    1859203
## 121  GUY                           Guyana           South America     772298
## 122  SLV                       El Salvador           North America    7185218
```

```
## 123  GEO                    Georgia                  Asia    4615807
## 124  IDN                  Indonesia                  Asia  240271522
## 125  UKR                    Ukraine                Europe   45700395
## 126  SWZ                  Swaziland                Africa    1123913
## 127  PHL                Philippines                  Asia   97976603
## 128  LKA                  Sri Lanka                  Asia   21324791
## 129  CPV                 Cabo Verde                Africa     429474
## 130  GTM                  Guatemala           North America   13276517
## 131  FSM       Micronesia, Fed. Sts.                Oceania     107434
## 132  VUT                    Vanuatu                Oceania     218519
## 133  MAR                    Morocco                Africa   34859364
## 134  WBG          West Bank and Gaza                  Asia    4550368
## 135  EGY          Egypt, Arab Rep.                  Africa   83082869
## 136  NGA                    Nigeria                Africa  149229090
## 137  KIR                   Kiribati                Oceania     112850
## 138  BOL                    Bolivia           South America    9775246
## 139  COG                Congo, Rep.                Africa    4012809
## 140  TMP                Timor-Leste                  Asia    1201542
## 141  MDA                    Moldova                Europe    4320748
## 142  BTN                     Bhutan                  Asia     691141
## 143  HND                   Honduras           North America    7792854
## 144  PNG          Papua New Guinea                Oceania    6057263
## 145  UZB                 Uzbekistan                  Asia   27606007
## 146  VNM                    Vietnam                  Asia   86967524
## 147  NIC                  Nicaragua           North America    5891199
## 148  SLB            Solomon Islands                Oceania     595613
## 149  SDN                      Sudan                Africa   25946220
## 150  ZMB                     Zambia                Africa   11862740
## 151  STP      Sao Tome and Principe                Africa     212679
## 152  LAO                   Lao PDR                  Asia    6834942
## 153  GHA                      Ghana                Africa   23832495
## 154  IND                      India                  Asia 1166079220
## 155  CIV              Cote d'Ivoire                Africa   20617068
## 156  PAK                   Pakistan                  Asia  176242949
## 157  YEM                Yemen, Rep.                  Asia   23822783
## 158  CMR                   Cameroon                Africa   18879301
## 159  LSO                    Lesotho                Africa    2130819
## 160  KEN                      Kenya                Africa   39002772
## 161  MRT                 Mauritania                Africa    3129486
## 162  MMR                    Myanmar                  Asia   48137741
## 163  KGZ            Kyrgyz Republic                  Asia    5431747
## 164  BGD                 Bangladesh                  Asia  156050883
## 165  TJK                 Tajikistan                  Asia    7349145
## 166  SEN                    Senegal                Africa   13711597
## 167  KHM                   Cambodia                  Asia   14494293
## 168  TCD                       Chad                Africa   10329208
## 169  SSD                South Sudan                Africa   10625176
## 170  TZA                   Tanzania                Africa   41048532
## 171  BEN                      Benin                Africa    8791832
## 172  ZWE                   Zimbabwe                Africa   12619600
## 173  HTI                      Haiti           North America    9035536
## 174  COM                    Comoros                Africa     752438
## 175  NPL                      Nepal                  Asia   28563377
## 176  BFA                Burkina Faso                Africa   15746232
```

```
## 177  RWA                       Rwanda                 Africa   10473282
## 178  SLE                 Sierra Leone                 Africa    6440053
## 179  AFG                  Afghanistan                   Asia   28400000
## 180  UGA                       Uganda                 Africa   32369558
## 181  MLI                         Mali                 Africa   12666987
## 182  MOZ                   Mozambique                 Africa   21669278
## 183  TGO                         Togo                 Africa    6019877
## 184  ETH                     Ethiopia                 Africa   85237338
## 185  GNB                Guinea-Bissau                 Africa    1533964
## 186  GIN                       Guinea                 Africa   10057975
## 187  GMB                  Gambia, The                 Africa    1782893
## 188  MDG                   Madagascar                 Africa   20653556
## 189  NER                        Niger                 Africa   15306252
## 190  COD            Congo, Dem. Rep.                  Africa   68692542
## 191  LBR                      Liberia                 Africa    3441790
## 192  CAF      Central African Republic                 Africa    4511488
## 193  BDI                      Burundi                 Africa    8988091
## 194  MWI                       Malawi                 Africa   14268711
##        GNI
## 3   106140
## 4   103630
## 5    92200
## 6    88120
## 7    76270
## 8    75990
## 10   64540
## 11   61610
## 12   61310
## 14   55200
## 15   55150
## 18   51890
## 19   51630
## 20   49670
## 21   49300
## 22   48420
## 23   47640
## 24   46350
## 25   47260
## 26   46550
## 27   44600
## 28   43430
## 29   42960
## 30   42000
## 31   43270
## 32   41070
## 33   40320
## 34   37320
## 35   35320
## 36   34270
## 37   29440
## 38   27090
## 39   26370
## 40   25140
## 41   23580
```

```
## 42   22680
## 43   21360
## 44   21060
## 45   20980
## 46   21000
## 47   20070
## 48   19310
## 49   19030
## 50   18370
## 51   16870
## 52   17750
## 53   16350
## 54   15430
## 55   15280
## 56   15310
## 57   14920
## 58   14910
## 59   14100
## 60   13690
## 61   13480
## 62   13340
## 63   13300
## 64   13220
## 65   12980
## 66   12500
## 67   11850
## 68   11530
## 69   11130
## 70   11120
## 71   11110
## 72   10830
## 73   10210
## 74   10120
## 75   10030
## 76    9950
## 77    9870
## 78    9720
## 79    9630
## 80    9520
## 81    8020
## 82    7970
## 83    7910
## 84    7820
## 85    7620
## 86    7590
## 87    7400
## 88    7340
## 89    7320
## 90    7260
## 91    7240
## 92    6930
## 93    6800
## 94    6610
## 95    6500
```

```
## 96    6410
## 97    6360
## 98    7120
## 99    6090
## 100   6040
## 101   5820
## 102   5780
## 103   5720
## 104   5630
## 105   5490
## 106   5160
## 107   5150
## 108   5150
## 109   4870
## 110   4760
## 111   4350
## 112   4450
## 113   4400
## 114   4390
## 115   4280
## 116   4260
## 117   4230
## 118   4060
## 119   4020
## 120   3990
## 121   3940
## 122   3920
## 123   3720
## 124   3630
## 125   3560
## 126   3550
## 127   3500
## 128   3460
## 129   3450
## 130   3430
## 131   3200
## 132   3160
## 133   3070
## 134   3060
## 135   3050
## 136   2970
## 137   2950
## 138   2870
## 139   2720
## 140   2680
## 141   2560
## 142   2370
## 143   2270
## 144   2240
## 145   2090
## 146   1890
## 147   1870
## 148   1830
## 149   1710
```

```
## 150   1680
## 151   1670
## 152   1660
## 153   1590
## 154   1570
## 155   1450
## 156   1400
## 157   1300
## 158   1350
## 159   1330
## 160   1290
## 161   1270
## 162   1270
## 163   1250
## 164   1080
## 165   1080
## 166   1050
## 167   1020
## 168    980
## 169    970
## 170    920
## 171    890
## 172    840
## 173    820
## 174    790
## 175    730
## 176    700
## 177    700
## 178    700
## 179    680
## 180    670
## 181    650
## 182    600
## 183    570
## 184    550
## 185    550
## 186    470
## 187    500
## 188    440
## 189    410
## 190    380
## 191    370
## 192    320
## 193    270
## 194    250
```

```r
# create one column with a path string
GNI2014 = GNI2014 %>%
  mutate(
    webpath = paste("world", continent, country, sep = "/")
  )

# transform this data to a hierarchical tree that is suitable for circlepackR
#
```

```
population <- as.Node(GNI2014, pathName = "webpath")
circlepackeR(population,  size = "population")
```

```
circlepackeR(population,  size = "GNI")
```

Interactieve sunburst plots kan je ook mooi gebruiken voor hierarchische data.

```
library(sunburstR)

seqData = read.csv(
  file = paste0(
    "https://gist.githubusercontent.com/mkajava/",
    "7515402/raw/9f80d28094dc9dfed7090f8fb3376ef1539f4fd2/",
    "comment-sequences.csv"
  )
  ,header = TRUE
  ,stringsAsFactors = FALSE
)

sunburst(
  seqData
)

sunburst(
  seqData
  ,count = TRUE
)

## aangepaste text in het midden
sunburst(
  seqData
  # apply sort order to the legends
  ,legendOrder = unique(unlist(strsplit(seqData[,1],":")))
  # just provide the name in the explanation in the center
  ,explanation = "function(d){return d.data.name}"
)
```

Chord diagrams zijn een leuke manier om "A naar B" data weer te geven, relaties tussen entiteiten.

```
#devtools::install_github("mattflor/chorddiag")
library(chorddiag)

# overgangsmatrix
m <- matrix(
  c(11975,  5871, 8916, 2868,
    1951, 10048, 2060, 6171,
    8010, 16145, 8090, 8045,
    1013,   990,  940, 6907
  ),
  byrow = TRUE,
  nrow = 4, ncol = 4
)

haircolors <- c("black", "blonde", "brown", "red")
dimnames(m) <- list(have = haircolors, prefer = haircolors)
```

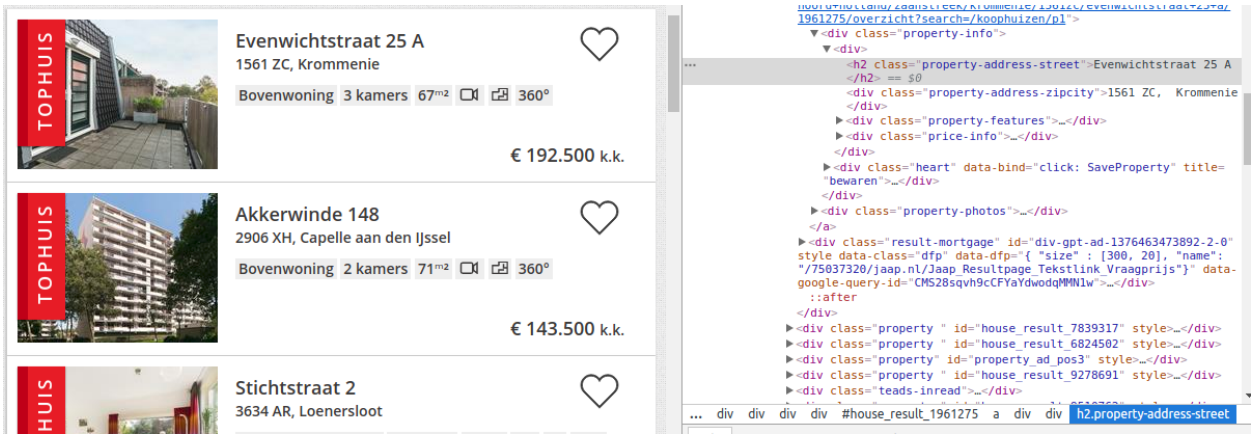

Figure 1:

```
m

groupColors <- c("#000000", "#FFDD89", "#957244", "#F26223")
chorddiag(m, groupColors = groupColors, groupnamePadding = 20)
```

11 Web Scraping met rvest

11.1 Basis voorbeelden

In R kan je web sites scrapen met het package `rvest`, dit is een heel handig package. Eerst de basics.

```
library(rvest)

## Geef een URL op
baseurl = "http://www.jaap.nl/koophuizen/p1"
out = read_html(baseurl)

## out is een zogenaamde xml_document object
class(out)
```

Met verschillende functies kan je nu een xml_document parsen, bestudeer de site met chrome om een idee te krijgen van de elementen en structuur. Met behulp van CSS of XPath (XML Path-taal of XML Path Language), een querytaal voor het adresseren van onderdelen van XML-documenten, kan je verschillende elementen uit een xml_document krijgen.

Een kort overzicht vind je hier . Gebruik dubbel slash `//` om alle elementen in een document te vinden, bijvoorbeeld `//h2` om alle h2 elementen in een docuemnt te pakken. Gebruik `@` om attributen te selecteren.

Neem als voorbeeld de straat van het huis, rechts klik op de straat en klik inspect element, het is een `h2` object met een bepaalde `class` property. Deze kan je dan met Xpath als volgt opgeven:

```
## pak als voorbeeld de straat
strout  = html_nodes(out,xpath='//h2[@class="property-address-street"]')

## Er zijn 30 van de straat objecten gevonden in de pagina
length(strout)
```

Figure 2:

```
## Het is een zogenaamde xml_nodeset
class(strout)
strout

## met de `html_text` functie kan je de tekst er uit halen
straat = strout %>% html_text()
straat
```

Nog een voorbeeldje, de prijs van een huis.

```
price =  html_nodes(out, xpath='//div[@class="property-price"]') %>% html_text()

## De prijs is zichtbaar maar het is niet handig om te gebruiken
price
```

Gebruik reguliere expressies om de prijs er uit te pulleken en als bruikbaar numeriek veld te hebben.

```
library(stringr)
prijs = str_extract(
  price,
  "[:digit:]+[\\.][:digit:]+[\\.]*[:digit:]*"
) %>%
str_replace_all("\\." , "") %>%
  as.numeric()
prijs
```

In de huizen data staan ook per huis een link, deze kan je er ook uit halen.

Het is een `a` tag met `class`=“property-inner” waarvan je de href moet hebben, dat kan je als volgt met xpath opgeven.

```
linkhuis =  html_nodes(out, xpath='//a[@class="property-inner"]/@href') %>% html_text()
linkhuis
```

In een `for` loop zou je al deze links weer kunnen scrapen.

11.2 Verder voorbeelden

```
tmp = html_form(read_html("https://hadley.wufoo.com/forms/libraryrequire-quiz/"))
class(tmp)
tmp[[1]]$fields$clickOrEnter$name
```

Zie ook de Environment browser waar je het object `tmp` kan browsen. Soms bevat een html pagina een tabel, deze kan je makkelijk vertalen naar een R data set met `html_table`.

```
## op deze pagina staan interessante geboorte statistieken
births <- read_html("https://www.ssa.gov/oact/babynames/numberUSbirths.html")

out = html_nodes(births, "table")

## er zijn twee tabellen op de pagina, de tweede bevat aantal M en F geboortes per jaar
out[[2]] %>% html_table()
```

11.3 Het jaap scrape script

Een compleet script om jaap.nl te scapen zie je hier. Switch naar jaap project.

11.4 NOS nieuws site scrapen

Nog een voorbeeld, nieuws archief berichten van de NOS site scrapen. De archieven zijn per dag georganiseerd. Zie bijvoorbeeld de links:

- https://nos.nl/nieuws/archief/2017-10-23/
- https://nos.nl/nieuws/archief/2017-10-22/
- https://nos.nl/nieuws/archief/2017-10-21/
- https://nos.nl/nieuws/archief/2016-10-23/
- etc.

Laten we als voorbeeld de laatste 10 dagen pakken

```
library(purrr)
library(dplyr)

OUT = data.frame()

## we kunnen handmatig 1 iteratie uitvoeren om te zien of het werkt
j=1

## en nu de rest :-)
for(j in 1:3){

  ### Maak de juiste URL aan
  datum = as.character(Sys.Date() - j)
  nieuwslinks = paste0("https://nos.nl/nieuws/archief/", datum)

  ### Lees de link in
  out = read_html(nieuwslinks)

  ### dit is een nodeset met artikelen, maar daar moeten we de links uit pulleken
  artikels = html_nodes(
    out,
    xpath = "//a[contains(@href, 'artikel')]"
  )

  ## maak een lijst van attributen waarbij je alleen de href genaamde list objecten pakt
  lijst = artikels %>%   html_attrs()

  ## gebruik uit purrr map_chr, waarbij je een lijst langs kan gaan en alleen bepaalde elementen kan pa
```

```
  artikels = lijst %>%  map_chr("href") %>%  unique()

  artikels = paste0("https://nos.nl", artikels)

  ### loop nu langs alle artiekelen links

 for(i in 1:length(artikels))
 {
    print(i)
    out2 = read_html(artikels[i])
    art = html_nodes(out2, xpath = '//div[@class="article_textwrap"]') %>% html_text %>% paste(collapse
    titel = html_nodes(out2, xpath = '//h1[@class="article__title"]') %>% html_text()
    datum = html_nodes(out2, xpath = '//time') %>% html_attrs() %>% .[[1]]
    categorie = html_nodes(out2, xpath='//a[@class="link-grey"]') %>% html_text
    tmp = data.frame(datum, titel, art, link = artikels[i], categorie)
    OUT = rbind(OUT,tmp)
  }
  OUT = OUT %>% distinct()
  print(j)
  print(datum)
  saveRDS(OUT, "nosnieuws.RDs")
  Sys.sleep(2*runif(1))
}
```

11.5 SelectorGadget in chrome

Met de selector gadget plugin (in Chrome) kan je elementen selecteren op een web pagina, je krijgt dan de CSS te zien en kan deze gebruiken in de functie `html_nodes`. Een aantal voorbeelden:

11.5.1 Hema voorbeeldje

```
### HEMA producten
clean_string = function(x){
  x %>%
    str_remove_all("\n") %>%
    str_remove_all("\r") %>%
    str_remove_all("\t") %>%
    str_trim()
}

hema = "https://www.hema.nl/winkel/baby/babyspeelgoed"
out = read_html(hema)
speelgoed = html_nodes(out, "h4") %>% html_text() %>% clean_string()
prijs = html_nodes(out, ".price") %>% html_text()
```

11.5.2 Mediamarkt voorbeeldje

```
### Media Markt laptops, scrape 1 pagina....
MM = "http://www.mediamarkt.nl/nl/category/_laptops-482723.html?searchParams=&sort=&view=PRODUCTLIST&pag
```

```
out = read_html(MM)

price = html_nodes(out, ".product-wrapper") %>%
  html_nodes(".price-box") %>%
  html_text() %>%
  clean_string() %>%
  str_extract("\\d+") %>%
    as.numeric()

merk = html_nodes(out, ".product-wrapper") %>%
  html_nodes("h2") %>%
  html_text() %>%
  clean_string()
```

EINDE SESSIE