Bevingen dashboard bouwen in Shiny

Willy Tadema - Provincie Groningen

Versie 0.1 - 17 juni 2019

Inleiding

Dit is de handleiding voor de workshop *Bevingen dashboard bouwen in Shiny* tijdens de FOSS4GNL bij de Technische Universiteit Delft. De workshop is onderdeel van het Maptime track.

- Datum: **Donderdag 20 juni 2019**
- Tijd: **15:10 - 16:10 uur**
- Locatie: **Zaal R**

Workshop materiaal

De code en gegevensbestanden voor de workshop zijn te downloaden van GitHub. Hier vind je ook deze handleiding, als PDF en R Markdown bestand.

Het dashboard dat we in de workshop gaan bouwen kun je alvast bekijken op shinyapps.io.

Voorbereidingen

- Voor het volgen van de workshop heb je R en RStudio nodig. Je kunt deze software lokaal installeren of een RStudio Cloud account aanmaken.
- In beide gevallen heb je een internetverbinding nodig voor het volgen van de workshop.
- Download het zip-bestand met de code en gegevensbestanden voor de workshop of kloon de GitHub repository.
- We maken in de workshop gebruik van een aantal packages. Packages zijn bibliotheken met functies die de standaardfunctionaliteit van R uitbreiden. Het is handig om deze packages al van tevoren te installeren. Hoe dat moet, kun je lezen in paragraaf 1.

Inhoud

1 Packages installeren

Installeer de voor de workshop benodigde packages door onderstaande code uit te voeren.

```
install.packages(c("readr", "dplyr", "shiny", "shinydashboard",
    "DT", "plotly", "leaflet", "RColorBrewer", "jsonlite", "sf"))
```

2 Data inlezen en bekijken

In deze workshop gaan we een dashboard bouwen met gegevens over bevingen in Groningen als gevolg van de gaswinning. In de GitHub repository staat een CSV-bestand met gegevens van het KNMI over geïnduceerde bevingen. In deze oefening gaan we de gegevens inlezen, bekijken en presenteren in een dashboard.

Begin met het laden van de libraries die voor dit deel van de workshop nodig zijn.

```
library(readr)
library(dplyr)
library(shiny)
library(shinydashboard)
library(DT)
```

Lees vervolgens het CSV-bestand met gegevens over de bevingen in.

```
bevingen <- read_csv("./data/bevingen.csv")
```

```
## Parsed with column specification:
## cols(
##   datum = col_date(format = ""),
##   locatie = col_character(),
##   magnitude = col_double(),
##   diepte = col_double(),
##   latitude = col_double(),
##   longitude = col_double(),
##   jaar = col_double()
## )
```

Print de eerste zes rijen in de dataset.

```
head(bevingen)
```

```
## # A tibble: 6 x 7
##   datum      locatie   magnitude diepte latitude longitude  jaar
##   <date>     <chr>         <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 1986-12-26 Assen           2.8    1       53.0      6.55  1986
## 2 1987-12-14 Hooghalen       2.5    1.5     52.9      6.55  1987
## 3 1989-12-01 Kwadijk         2.7    1.2     52.5      4.97  1989
## 4 1991-02-15 Emmen           2.2    3       52.8      6.91  1991
## 5 1991-04-25 Eleveld         2.6    3       53.0      6.58  1991
## 6 1991-08-08 Assen           2.7    3       53.0      6.57  1991
```

Bepaal de datum van de eerste en de laatste beving in de dataset.

```
bevingen %>% summarise(eerste = min(datum), laatste = max(datum))
```

```
## # A tibble: 1 x 2
##   eerste     laatste
##   <date>     <date>
## 1 1986-12-26 2019-06-09
```

Bepaal de magnitude van de lichtste en de zwaarste beving.

```
bevingen %>% summarise(lichtste = min(magnitude), zwaarste = max(magnitude))
```

```
## # A tibble: 1 x 2
##   lichtste zwaarste
##      <dbl>    <dbl>
## 1     -0.8      3.6
```

Print de alleen de bevingen uit 1992.

```
bevingen %>% filter(jaar == 1992)
```

```
## # A tibble: 6 x 7
##   datum      locatie    magnitude diepte latitude longitude  jaar
##   <date>     <chr>          <dbl>  <dbl>    <dbl>     <dbl> <dbl>
## 1 1992-05-23 Geelbroek        2.6    3       53.0      6.57  1992
## 2 1992-05-24 Assen            1.6    3       53.0      6.56  1992
## 3 1992-06-11 Roswinkel        2.7    1.5     52.8      7.03  1992
## 4 1992-07-22 Assen            2.6    3       53.0      6.57  1992
## 5 1992-12-06 Loppersum        1.3    3       53.3      6.74  1992
## 6 1992-12-11 Froombosch       1.4    3       53.2      6.75  1992
```

Je hebt de data ingelezen en verkend. Je bent er klaar voor om je eerste Shiny app te maken!

Shiny is een krachtig framework voor het bouwen van interactieve web applicaties binnen R.

Een Shiny app is dus een web applicatie. Zo'n web applicatie is opgebouwd uit twee delen: een gebruikersinterface en serverfunctie.

De gebruikersinterface bestaat uit een header, sidebar en body.

De serverfunctie bevat logica voor het manipuleren van gegevens op basis van input van gebruikers en het renderen van grafische output.

Maak eerst een hele eenvoudige Shiny app. Laat de serverfunctie voorlopig nog leeg.

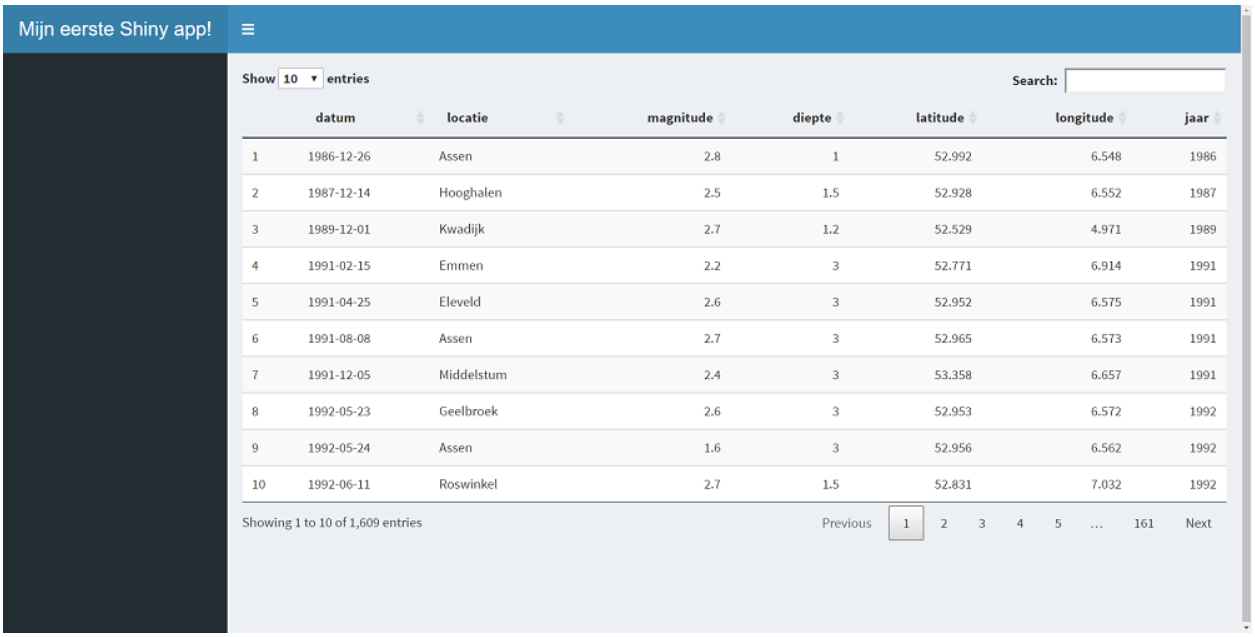
```
header <- dashboardHeader(title = "Mijn eerste Shiny app!")

sidebar <- dashboardSidebar("Dit is de sidebar")

body <- dashboardBody("Dit is de body")

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
}

shinyApp(ui, server)
```

Bij het afsluiten van de app krijg je een foutmelding, daar komen we aan het eind van deze paragraaf op terug. Voor nu kun je de foutmelding gewoon negeren.

Voeg nu een interactieve tabel toe aan het dasboard met daarin de gegevens van de bevingen. Gebruik hiervoor het DT package.

```
header <- dashboardHeader(title = "Mijn eerste Shiny app!")

sidebar <- dashboardSidebar()

body <- dashboardBody(DTOutput(outputId = "tabel"))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
    output$tabel <- renderDT(bevingen)
}

shinyApp(ui, server)
```

Merk op dat er aan de gebruikersinterface een *placeholder* is toegevoegd voor de outputvariabele `tabel`. De inhoud van deze variabele wordt gegenereerd binnen de serverfunctie.



De app ziet er al goed uit, alleen jammer dat de gebruikersinterface Engelstalig is. Pas het aan naar Nederlands.

De sidebar heeft op dit moment geen functie. Verberg dit onderdeel van de gebruikersinterface.

```
header <- dashboardHeader(title = "Mijn eerste Shiny app!")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(DTOutput(outputId = "tabel"))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
    dutch <- list(url = "http://cdn.datatables.net/plug-ins/1.10.19/i18n/Dutch.json")
    output$tabel <- renderDT(bevingen, options = list(language = dutch))
}

shinyApp(ui, server)
```

Voeg als laatste een knop toe voor het downloaden van de gegevens.

```
header <- dashboardHeader(title = "Mijn eerste Shiny app!")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(
  DTOutput(outputId = 'tabel'),
  downloadButton(outputId = 'downloadData', label = "Download gegevens")
)

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  dutch <- list(url = "http://cdn.datatables.net/plug-ins/1.10.19/i18n/Dutch.json")
  output$tabel <- renderDT(bevingen,
                           options = list(language = dutch))

  output$downloadData <- downloadHandler(
    filename = "bevingen.csv",
    content = function(file) {write_csv(bevingen, file)}
  )
}

shinyApp(ui, server)
```

De conventie is om de code van een Shiny app op te slaan in een bestand met de naam `app.R`. Je kunt er ook voor kiezen om de code voor de gebruikersinterface op te slaan in `ui.R` en voor de serverfunctie in `server.R`.

Ga naar de map `shiny_apps\1_data_inlezen_en_bekijken\app.R`. Het bestand bevat alle code die nodig is voor je eerste Shiny app!

Start de Shiny app. Merk op dat je nu géén foutmelding krijgt bij het afsluiten van de app.

3 Histogram maken

In deze paragraaf gaan we een interactief histogram maken. Het histogram geeft de frequentieverdeling van de magnitude van bevingen.

Begin met het laden van de libraries die we nodig hebben.

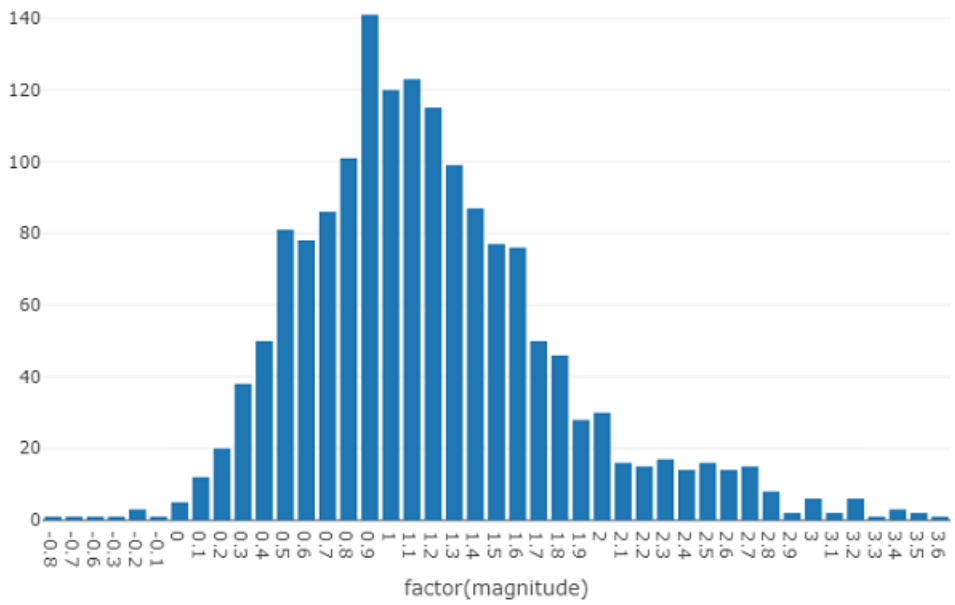
```
library(readr)
library(dplyr)
library(shiny)
library(shinydashboard)
library(plotly)
```

Lees vervolgens de gegevens in.

```
bevingen <- read_csv("./data/bevingen.csv")
```

Maak een histogram van de magnitude van de bevingen. Gebruik hiervoor Plotly, zodat je interactieve elementen kunt toevoegen.

```
plot_ly(data = bevingen, x = ~ factor(magnitude)) %>%
  add_histogram()
```



Pas de opmaak van de x- en y-as aan. Voeg titels toe en plaats minder labels langs de x-as.

```
plot_ly(data = bevingen, x = ~ factor(magnitude)) %>%
  add_histogram() %>%
  layout(
    xaxis = list(title = "Magnitude (Richter)", dtick = 10),
    yaxis = list(title = "Frequentie")
  )
```

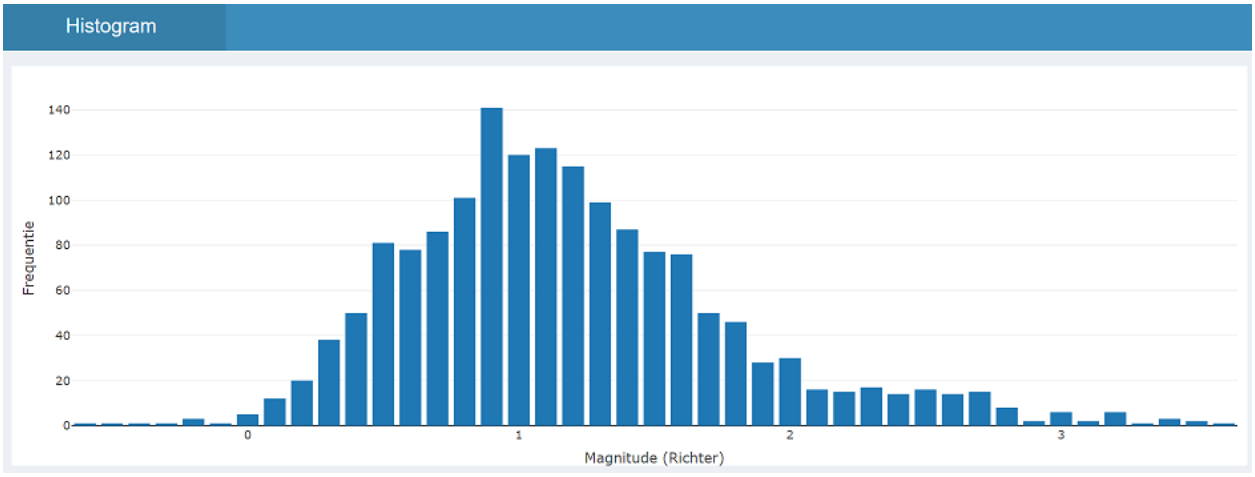
Pas de hovermodus aan, zodat je makkelijk de x- en y-waarde kunt aflezen. Verberg de menubalk van het histogram.

```
plot_ly(data = bevingen, x = ~ factor(magnitude)) %>%
  add_histogram() %>%
  layout(
    xaxis = list(title = "Magnitude (Richter)", dtick = 10),
    yaxis = list(title = "Frequentie"),
    hovermode = 'compare'
  ) %>%
  config(displayModeBar = F)
```

Maak een Shiny app van het histogram.

```
header <- dashboardHeader(title = "Histogram")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(plotlyOutput(outputId = "histogram"))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  output$histogram <- renderPlotly({
    plot_ly(data = bevingen, x = ~ factor(magnitude)) %>%
      add_histogram() %>%
      layout(
        xaxis = list(title = "Magnitude (Richter)", dtick = 10) ,
        yaxis = list(title = "Frequentie"),
        hovermode = 'compare'
      ) %>%
      config(displayModeBar = F)
  })
}

shinyApp(ui, server)
```

Merk op dat de gebruikersinterface een placeholder bevat voor de outputvariabele `histogram`. De inhoud van deze variabele wordt gerenderd door de serverfunctie.



Merk op dat het histogram de volledige breedte van het scherm in beslag neemt en er daardoor uitgerekt uitziet.

Shiny maakt gebruik van het Bootstrap framework om er voor te zorgen dat de apps *reponsive* zijn. Dat wil zeggen dat ze er goed uitzien op alle apparaten en mooi meeschalen als het scherm groter of kleiner wordt.

Door het histogram op te nemen in een `box()` krijg je meer controle over de afmetingen.

Bootstrap gebruikt een grid waarbij het scherm is opgedeeld in 12 kolommen. De afmetingen van een box druk je daarom uit in een (relatieve) schaal van 1 tot en met 12. Met `width = 6` specificeer je dat de box de helft van de breedte van het scherm in beslag mag nemen.

Voeg een box toe aan de gebruikersinterface en plaats het histogram daarbinnen.

Voer onderstaande code meerdere malen uit, iedere keer met een andere waarde voor `width` om een idee te krijgen van hoe Bootstrap werkt.

```
header <- dashboardHeader(title = "Histogram")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(box(width = 6, plotlyOutput(outputId = "histogram")))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  output$histogram <- renderPlotly({
    plot_ly(data = bevingen, x = ~ factor(magnitude)) %>%
      add_histogram() %>%
      layout(
        xaxis = list(title = "Magnitude (Richter)", dtick = 10) ,
        yaxis = list(title = "Frequentie"),
        hovermode = 'compare'
      ) %>%
      config(displayModeBar = F)
  })
}

shinyApp(ui, server)
```

Voeg aan de gebruikersinterface een slider toe voor het selecteren van een begin- en eindjaar. Op deze manier kan de gebruiker filteren op een bepaalde periode tussen 1985 en 2019.

```
header <- dashboardHeader(title = "Histogram")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(box(width = 6, plotlyOutput(outputId = "histogram")),
                      box(
                        width = 4,
                        sliderInput(inputId = "jaar", label = "Jaar",
                                    min = 1986, max = 2019, value = c(1986, 2019), sep =
                                    ↪  "")
                      ))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  # Filter data op basis van slider input
  gefilterdeBevingen <- reactive({
    bevingen %>%
      filter(jaar >= input$jaar[1] & jaar <= input$jaar[2])
  })

  # Render histogram
  output$histogram <- renderPlotly({
    # Databron is nu de reactive source!
    plot_ly(data = gefilterdeBevingen(), x = ~ factor(magnitude)) %>%
      add_histogram() %>%
      layout(
        xaxis = list(title = "Magnitude (Richter)", dtick = 10),
        yaxis = list(title = "Frequentie"),
        hovermode = 'compare'
      ) %>%
      config(displayModeBar = F)
  })
}

shinyApp(ui, server)
```

Via de slider kent de gebruiker een waarde toe aan de *inputvariabele* `jaar`. `input$jaar[1]` bevat het beginjaar en `input$jaar[2]` het eindjaar.

De serverfunctie maakt gebruik van de inputvariabele om de gegevens te filteren op periode. De filtering vindt plaats binnen de *reactive source* `gefilterdeBevingen`. Iedere keer als de gebruiker de slider aanpast en de inputvariabele `jaar` een nieuwe waarde krijgt, wordt de filtering automatisch aangepast en het histogram opnieuw gerenderd.

In het bestand `shiny_apps\2_histogram_maken\app.R` vind je alle code van de Shiny app uit deze paragraaf.

4 Kaart maken

Laad eerst de libraries die we in dit deel van de workshop nodig hebben.

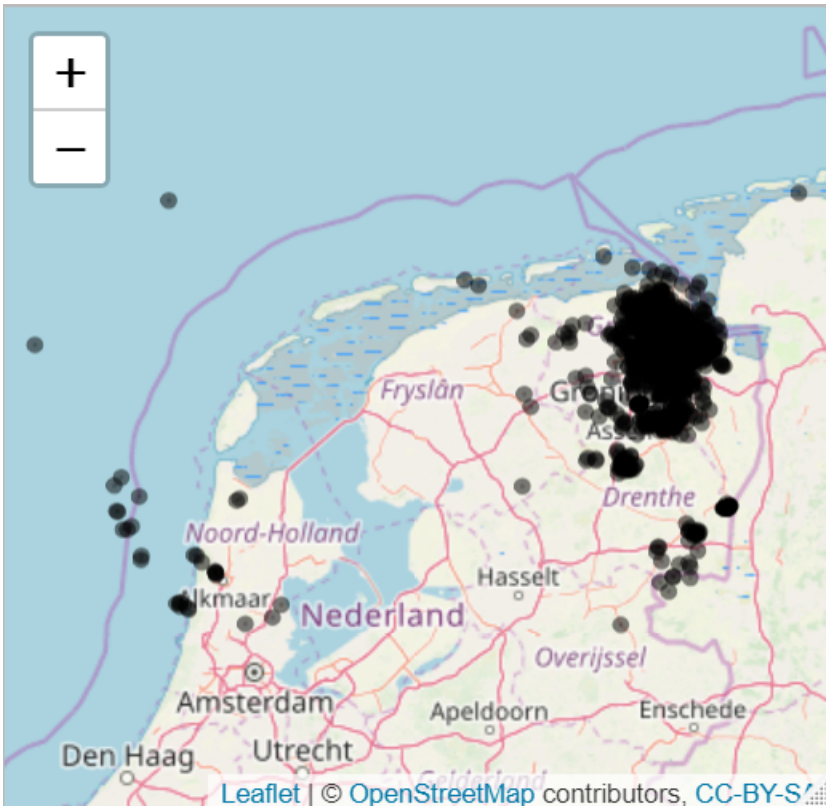
```
library(readr)
library(dplyr)
library(shiny)
library(shinydashboard)
library(leaflet)
library(RColorBrewer)
library(jsonlite)
```

Lees voor de zekerheid nogmaals de gegevens in.

```
bevingen <- read_csv("./data/bevingen.csv")
```

Plot de bevingen op een kaart. Maak hierbij gebruik van Leaflet. Leaflet is een open source JavaScript library voor interactieve kaarten.

```
leaflet() %>%
  addTiles() %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = "black", weight = 1)
```



Laat de grootte van de cirkels afhankelijk zijn van de magnitude van de bevingen. Maak de cirkels enigszins transparant.

Voor het berekenen van de radius van een cirkel hanteren we daarom de volgende (arbitraire!) formule: $\frac{10^{magnitude}}{10}$

```
leaflet() %>%
  addTiles() %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = "black", weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7)
```

Pas de kleur van de cirkels aan. Definieer een kleurenpalet met behulp van ColorBrewer. Maak de lichtste bevingen oranje en de zwaarste rood.

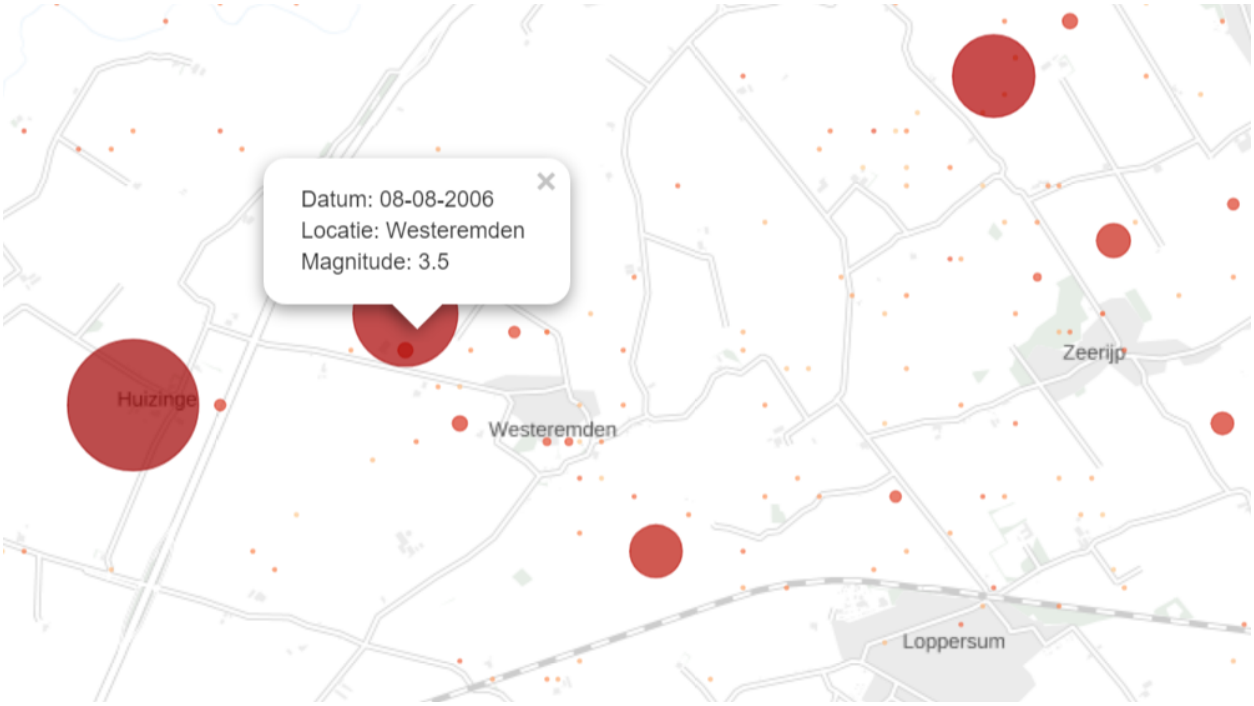
```
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

leaflet() %>%
  addTiles() %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = ~ pal(magnitude), weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7)
```

Voeg een popup toe.

```
leaflet() %>%
  addTiles() %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = ~ pal(magnitude), weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
             popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                             "Locatie:", locatie, "<br>",
                             "Magnitude:" , magnitude)
  )
```

Vervang de OpenStreetMap ondergrond door de BRT Achtergrondkaart.

```
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")

pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

leaflet() %>%
  addTiles(urlTemplate = brtAchtergrondkaart,
           attribution = "Kaartgegevens &copy; Kadaster",
           options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = ~ pal(magnitude), weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
             popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                             "Locatie:", locatie, "<br>",
                             "Magnitude:" , magnitude)
  )
```

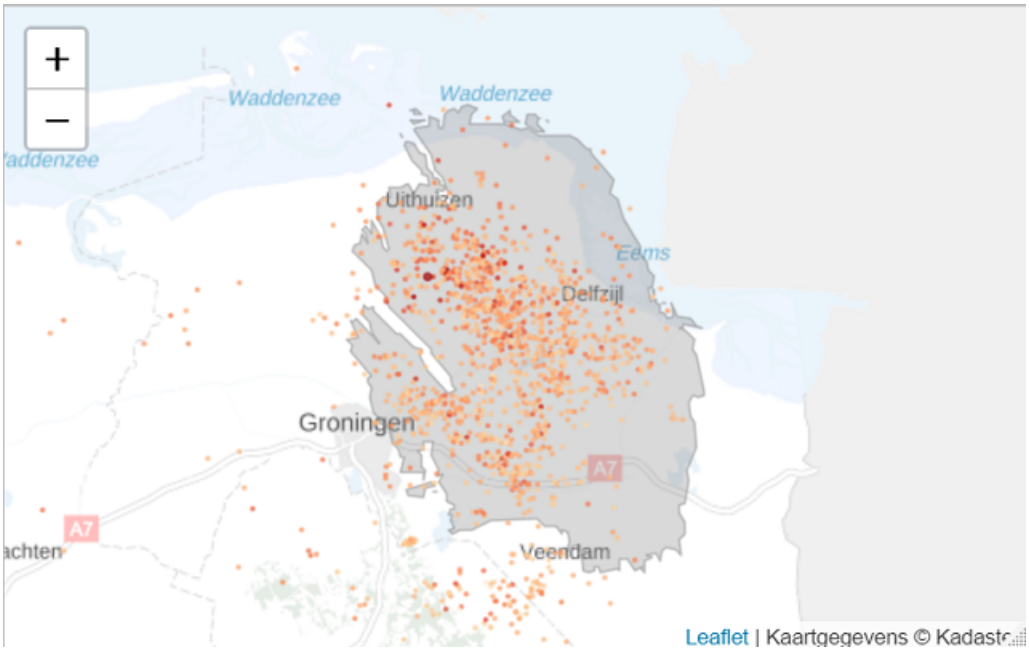


Zoom in op Groningen.

```
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")

pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

leaflet() %>%
  addTiles(urlTemplate = brtAchtergrondkaart,
           attribution = "Kaartgegevens &copy; Kadaster",
           options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = ~ pal(magnitude), weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
             popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                             "Locatie:", locatie, "<br>",
                             "Magnitude:" , magnitude))  %>%
  setView(6.8, 53.3, zoom = 10)
```

Voeg het Groningen veld toe als kaartlaag. De volgorde van de lagen is belangrijk: de laag met de bevingen moet bovenop de laag met het gasveld liggen, anders werkt de popup niet. Maak gebruik van de functie `addMapePane()` en de parameter `zIndex` om controle te houden over de volgorde van de lagen.

```
veld <- read_json("./data/groningenveld.json")

brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")

pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

leaflet() %>%
  addTiles(urlTemplate = brtAchtergrondkaart,
           attribution = "Kaartgegevens &copy; Kadaster",
           options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
  addMapPane("groningen veld", zIndex = 410) %>%
  addMapPane("bevingen", zIndex = 420) %>%
  addTopoJSON(topojson = veld,
              weight = 1, color = "grey", fillOpacity = 0.3,
              options = pathOptions(pane = "groningen veld")) %>%
  addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
             color = ~ pal(magnitude), weight = 1,
             radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
             popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                             "Locatie:", locatie, "<br>",
                             "Magnitude:" , magnitude),
             options = pathOptions(pane = "bevingen"))  %>%
  setView(6.8, 53.3, zoom = 10)
```



Maak een Shiny app van de kaart.

```
# Data
bevingen <- read_csv("./data/bevingen.csv")
veld <- read_json("./data/groningenveld.json")
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")

# Kleurenpalet
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

 # Gebruikersinterface
header <- dashboardHeader(title = "Kaart")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(
  box(width = 6, leafletOutput(outputId = "kaart", height = 400))
)

ui <- dashboardPage(header, sidebar, body)

# Server
server <- function(input, output) {
  output$kaart <- renderLeaflet({
    leaflet() %>%
      addTiles(urlTemplate = brtAchtergrondkaart,
               attribution = "Kaartgegevens &copy; Kadaster",
               options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
      addMapPane("groningen veld", zIndex = 410) %>%
      addMapPane("bevingen", zIndex = 420) %>%
      addTopoJSON(topojson = veld,
                  weight = 1, color = "grey", fillOpacity = 0.3,
                  options = pathOptions(pane = "groningen veld")) %>%
      addCircles(data = bevingen, lng = ~ longitude, lat = ~ latitude,
                 color = ~ pal(magnitude), weight = 1,
                 radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
                 popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                                 "Locatie:", locatie, "<br>",
                                 "Magnitude:" , magnitude),
                 options = pathOptions(pane = "bevingen"))  %>%
    setView(6.8, 53.3, zoom = 9)
  })
}

# Run de app
shinyApp(ui, server)
```

Pas de Shiny app aan, zodat je kunt filteren op bevingen met een magnitude groter dan of gelijk aan een bepaalde drempelwaarde.

Voeg een invoervak toe aan de gebruikersinterface voor het invoeren van de drempelwaarde. Zorg er voor dat er geen waarden kleiner dan -1 of groter dan 4 ingevoerd kunnen worden. Maak -1 de standaardwaarde.

```
bevingen <- read_csv("./data/bevingen.csv")
veld <- read_json("./data/groningenveld.json")
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

header <- dashboardHeader(title = "Kaart")
sidebar <- dashboardSidebar(disable = TRUE)
body <- dashboardBody(
  box(width = 6, leafletOutput(outputId = "kaart", height = 400)),
  box(width = 2, numericInput(inputId = "drempelwaarde", label = "Drempelwaarde",
                              min = -1, max = 4, step = 0.1, value = -1))
)
ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  # Filter data op basis van drempelwaarde
  gefilterdeBevingen <- reactive({
    bevingen %>%
      filter(magnitude >= input$drempelwaarde)
  })

  # Render de kaart
  output$kaart <- renderLeaflet({
    leaflet() %>%
      addTiles(urlTemplate = brtAchtergrondkaart,
               attribution = "Kaartgegevens &copy; Kadaster",
               options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
      addMapPane("groningen veld", zIndex = 410) %>%
      addMapPane("bevingen", zIndex = 420) %>%
      addTopoJSON(topojson = veld,
                  weight = 1, color = "grey", fillOpacity = 0.3,
                  options = pathOptions(pane = "groningen veld")) %>%
      addCircles(data = gefilterdeBevingen(), # Databron is nu de gefilterde dataset!
                 lng = ~ longitude, lat = ~ latitude,
                 color = ~ pal(magnitude), weight = 1,
                 radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
      popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                      "Locatie:", locatie, "<br>",
                      "Magnitude:" , magnitude),
      options = pathOptions(pane = "bevingen"))  %>%
    setView(6.8, 53.3, zoom = 9)
  })
}

shinyApp(ui, server)
```

De app werkt, maar het probleem is dat iedere keer wanneer de gebruiker een andere drempelwaarde opgeeft, de hele Leaflet kaart opnieuw wordt gerenderd. Dat is niet erg efficiënt! Dit kun je voorkomen door een *reactive observer* toe te voegen.

```
bevingen <- read_csv("./data/bevingen.csv")
veld <- read_json("./data/groningenveld.json")
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

header <- dashboardHeader(title = "Kaart")
sidebar <- dashboardSidebar(disable = TRUE)
body <- dashboardBody(
  box(width = 6, leafletOutput(outputId = "kaart", height = 400)),
  box(width = 2, numericInput(inputId = "drempelwaarde", label = "Drempelwaarde",
                              min = -1, max = 4, step = 0.1, value = -1))
)
ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  gefilterdeBevingen <- reactive({
    bevingen %>%
      filter(magnitude >= input$drempelwaarde)
  })

  output$kaart <- renderLeaflet({
    leaflet() %>%
      addTiles(urlTemplate = brtAchtergrondkaart,
               attribution = "Kaartgegevens &copy; Kadaster",
               options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
      addMapPane("groningen veld", zIndex = 410) %>%
      addMapPane("bevingen", zIndex = 420) %>%
      addTopoJSON(topojson = veld,
                  weight = 1, color = "grey", fillOpacity = 0.3,
                  options = pathOptions(pane = "groningen veld")) %>%
      setView(6.8, 53.3, zoom = 9)
  })

  observe({
    leafletProxy(mapId = "kaart") %>%
      clearShapes() %>%
      addCircles(data = gefilterdeBevingen(),
                 lng = ~ longitude, lat = ~ latitude,
                 color = ~ pal(magnitude), weight = 1,
                 radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
                 popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                                 "Locatie:", locatie, "<br>",
                                 "Magnitude:" , magnitude),
                 options = pathOptions(pane = "bevingen"))
  })
}

shinyApp(ui, server)
```

Verberg bij het opstarten de kaartlaag met het gasveld. Maak de laag pas zichtbaar, nadat er een vinkje in een checkbox is gezet.

```
bevingen <- read_csv("./data/bevingen.csv")
veld <- read_json("./data/groningenveld.json")
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))
header <- dashboardHeader(title = "Kaart")
sidebar <- dashboardSidebar(disable = TRUE)
body <- dashboardBody(
  box(width = 6,
      leafletOutput(outputId = "kaart", height = 400)),
  box(width = 2,
      numericInput(inputId = "drempelwaarde", label = "Drempelwaarde",
                   min = -1, max = 4, step = 0.1, value = -1),
      checkboxInput(inputId = "veld", label = "Groningen veld tonen", value = FALSE))
)
ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  gefilterdeBevingen <- reactive({
    bevingen %>%
      filter(magnitude >= input$drempelwaarde)
  })

  output$kaart <- renderLeaflet({
    leaflet() %>%
      addTiles(urlTemplate = brtAchtergrondkaart,
               attribution = "Kaartgegevens &copy; Kadaster",
               options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
      addMapPane("groningen veld", zIndex = 410) %>%
      addMapPane("bevingen", zIndex = 420) %>%
      addTopoJSON(topojson = veld,
                  weight = 1, color = "grey", fillOpacity = 0.3,
                  options = pathOptions(pane = "groningen veld"),
                  group = "groningen veld") %>%
      setView(6.8, 53.3, zoom = 9) %>%
      hideGroup("groningen veld")
  })

  observe({
    leafletProxy(mapId = "kaart") %>%
      clearShapes() %>%
      addCircles(data = gefilterdeBevingen(),
                 lng = ~ longitude, lat = ~ latitude,
                 color = ~ pal(magnitude), weight = 1,
                 radius = ~ 10 ^ magnitude / 10, fillOpacity = 0.7,
                 popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                                 "Locatie:", locatie, "<br>",
                                 "Magnitude:" , magnitude),
                 options = pathOptions(pane = "bevingen"))
  })
```

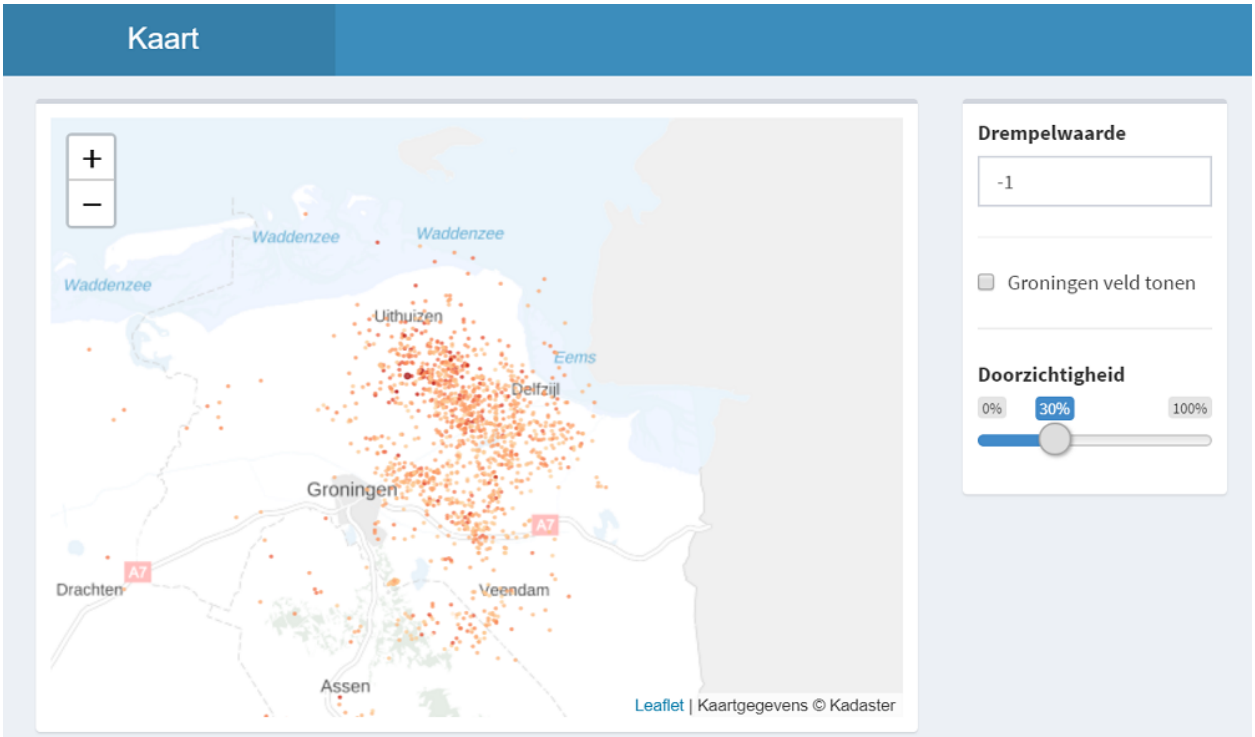
```
  # Zet de kaartlaag met het gasveld aan of uit afhankelijk van de waarde van de checkbox
  observe({
    if (input$veld == TRUE) {
      leafletProxy(mapId = "kaart") %>% showGroup("groningen veld")
    } else {
      leafletProxy(mapId = "kaart") %>% hideGroup("groningen veld")
    }
  })
}

shinyApp(ui, server)
```

Voeg tenslotte nog een slider toe waarmee de gebruiker de doorzichtigheid van de cirkels kan instellen.

```
bevingen <- read_csv("./data/bevingen.csv")
veld <- read_json("./data/groningenveld.json")
brtAchtergrondkaart <- paste0("http://geodata.nationaalgeoregister.nl/",
                              "tiles/service/wmts/brtachtergrondkaartgrijs/",
                              "EPSG:3857/{z}/{x}/{y}.png")
pal <- colorNumeric(palette = "OrRd", domain = c(-1:4))

header <- dashboardHeader(title = "Kaart")
sidebar <- dashboardSidebar(disable = TRUE)
body <- dashboardBody(
  box(width = 6,
      leafletOutput(outputId = "kaart", height = 400)),
  box(width = 2,
      numericInput(inputId = "drempelwaarde", label = "Drempelwaarde",
                   min = -1, max = 4, step = 0.1, value = -1),
      hr(),
      checkboxInput(inputId = "veld", label = "Groningen veld tonen", value = FALSE),
      hr(),
      sliderInput(inputId = "doorzichtigheid", label = "Doorzichtigheid",
                  min = 0, max = 100, step = 5, value = 30, ticks = FALSE, post = "%"))
)
ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  gefilterdeBevingen <- reactive({
    bevingen %>%
      filter(magnitude >= input$drempelwaarde)
  })

  output$kaart <- renderLeaflet({
    leaflet() %>%
      addTiles(urlTemplate = brtAchtergrondkaart,
               attribution = "Kaartgegevens &copy; Kadaster",
               options = tileOptions(minZoom = 6, maxZoom = 18)) %>%
      addMapPane("groningen veld", zIndex = 410) %>%
      addMapPane("bevingen", zIndex = 420) %>%
      addTopoJSON(topojson = veld,
                  weight = 1, color = "grey", fillOpacity = 0.3,
                  options = pathOptions(pane = "groningen veld"),
                  group = "groningen veld") %>%
      setView(6.8, 53.3, zoom = 9) %>%
      hideGroup("groningen veld")
  })
```

```
  observe({
    leafletProxy(mapId = "kaart") %>%
      clearShapes() %>%
      addCircles(data = gefilterdeBevingen(),
                 lng = ~ longitude, lat = ~ latitude,
                 color = ~ pal(magnitude), weight = 1,
                 radius = ~ 10 ^ magnitude / 10,
                 # fillOpacity is afhankelijk van slider input!
                 fillOpacity = 1 - (input$doorzichtigheid / 100),
                 popup = ~ paste("Datum: ", format(datum, "%d-%m-%Y"), "<br>",
                                 "Locatie:", locatie, "<br>",
                                 "Magnitude:" , magnitude),
                 options = pathOptions(pane = "bevingen"))
  })

  observe({
    if (input$veld == TRUE) {
      leafletProxy(mapId = "kaart") %>% showGroup("groningen veld")
    } else {
      leafletProxy(mapId = "kaart") %>% hideGroup("groningen veld")
    }
  })
}

shinyApp(ui, server)
```

In het bestand `shiny_apps\3_kaart_maken\app.R` vind je de code van de Shiny app die we in deze paragraaf hebben gebouwd.



5 Tijdreeks visualiseren

Begin met het laden van de libraries die we voor dit deel van de workshop nodig hebben.

```
library(readr)
library(dplyr)
library(shiny)
library(shinydashboard)
library(plotly)
```

Lees het CSV-bestand met het aantal bevingen en de gaswinning per jaar in.

```
jaargegevens <- read_csv("./data/gegevens geaggregeerd per jaar.csv")
```

```
## Parsed with column specification:
## cols(
##   jaar = col_double(),
##   aantal = col_double(),
##   gaswinning = col_double()
## )
```

Bekijk de eerste zes rijen in de dataset.

```
head(jaargegevens)
```

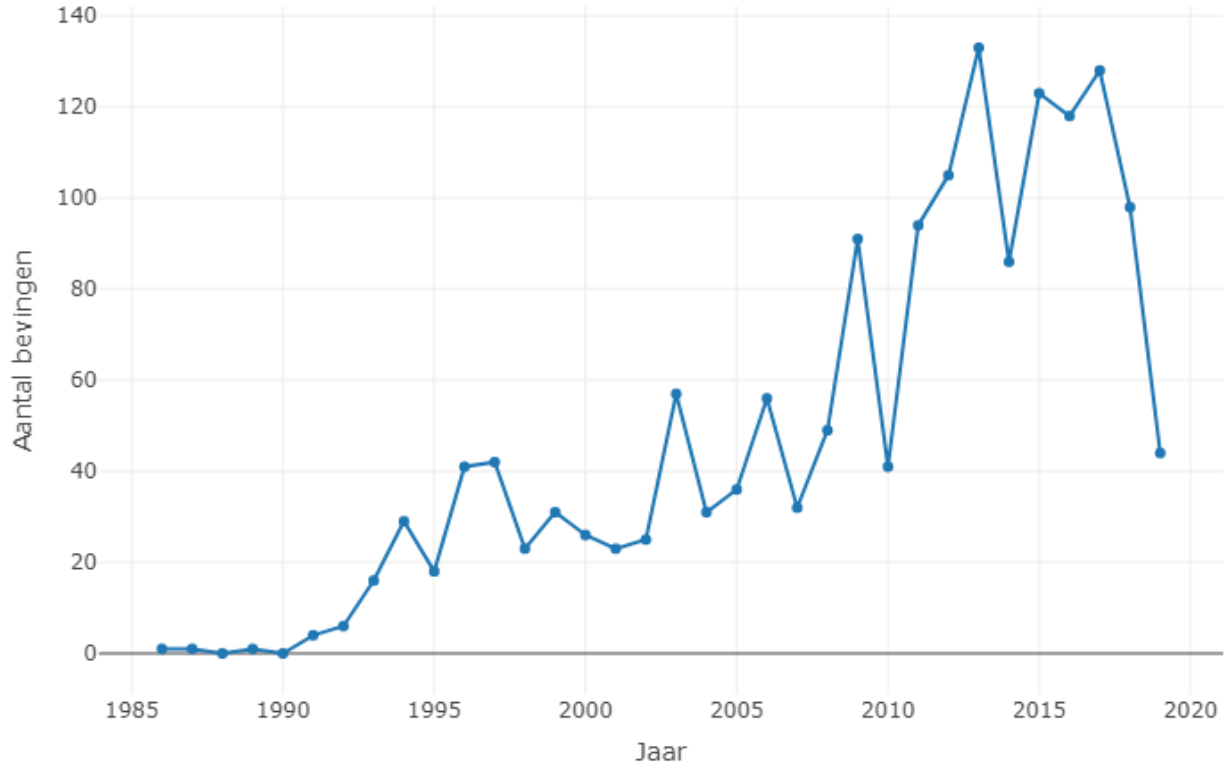
```
## # A tibble: 6 x 3
##    jaar aantal gaswinning
##   <dbl>  <dbl>      <dbl>
## 1  1986      1       41.6
## 2  1987      1       39.1
## 3  1988      0       30.3
## 4  1989      1       27.9
## 5  1990      0       29.1
## 6  1991      4       38.6
```

Maak een grafiek van het aantal bevingen per jaar.

```
plot_ly(data = jaargegevens, x = ~ jaar, y = ~ aantal,
        type = 'scatter', mode = 'markers+lines') %>%
  layout(xaxis = list(title = "Jaar"),
         yaxis = list(title = "Aantal bevingen"))
```

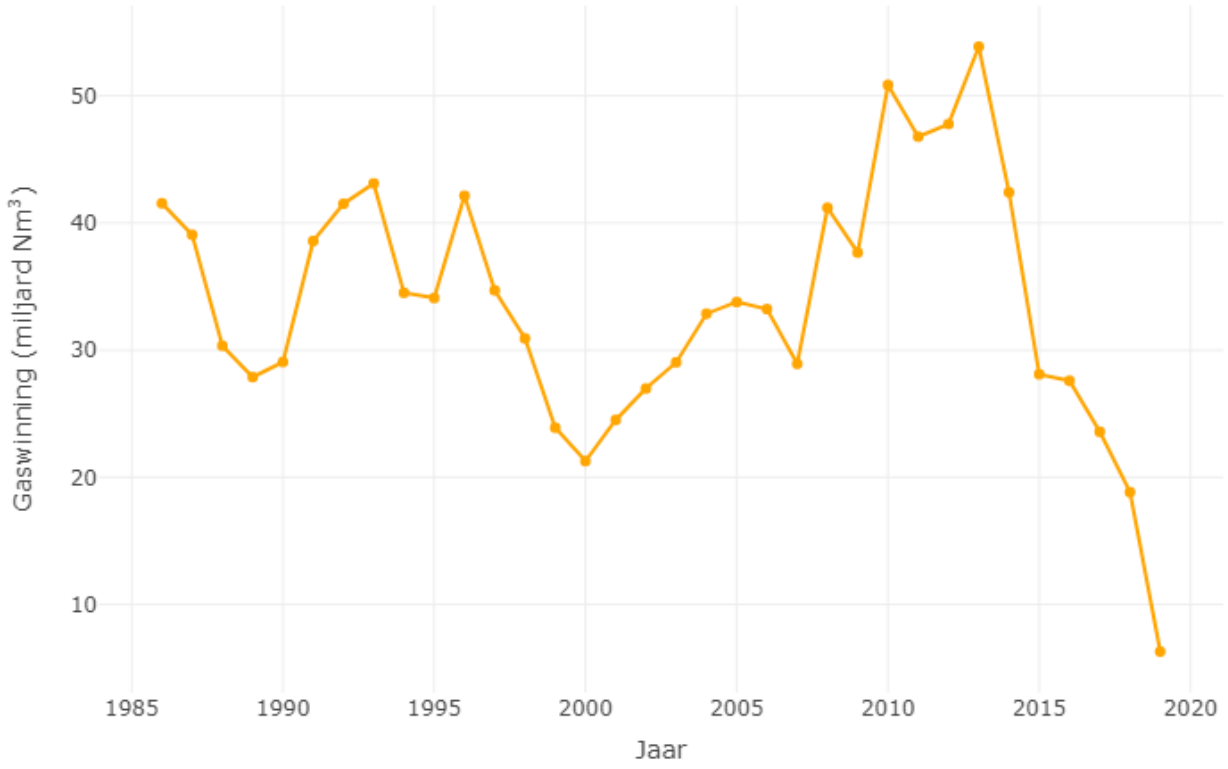
Pas de hovertekst aan en verberg de menubalk van de grafiek.

```
jaargegevens <- jaargegevens %>%
  mutate(hovertekst = paste("Jaar:", jaar, "<br>", "Aantal:", aantal))

plot_ly(data = jaargegevens, x = ~ jaar, y = ~ aantal,
        type = 'scatter', mode = 'markers+lines', hoverinfo = "text",
        text = ~ hovertekst) %>%
  layout(xaxis = list(title = "Jaar"),
         yaxis = list(title = "Aantal bevingen")) %>%
  config(displayModeBar = F)
```



Maak ook een grafiek van de gaswinning per jaar.

```
jaargegevens <- jaargegevens %>%
  mutate(hovertekst2 = paste("Jaar:", jaar, "<br>",
                             "Gaswinning:", gaswinning, "miljard Nm<sup>3</sup>"))

plot_ly(data = jaargegevens, x = ~ jaar, y = ~ gaswinning,
        type = 'scatter', mode = 'markers+lines', color = I('orange'),
        hoverinfo = "text", text = ~ hovertekst2) %>%
  layout(
    xaxis = list(title = "Jaar"),
    yaxis = list(title = "Gaswinning (miljard Nm<sup>3</sup>)")
  ) %>%
  config(displayModeBar = F)
```



Presenteer beide grafieken in een Shiny app.

```
# Data
jaargegevens <-
  read_csv("./data/gegevens geaggregeerd per jaar.csv") %>%
  mutate(
    hovertekst_aantal  = paste("Jaar:", jaar, "<br>", "Aantal:", aantal),
    hovertekst_winning = paste("Jaar:", jaar, "<br>",
                               "Gaswinning:", gaswinning, "miljard Nm<sup>3</sup>")
  )

# Gebruikersinterface
header <- dashboardHeader(title = "Bevingen in de tijd")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(box(width = 6, plotlyOutput(outputId = "grafiekAantal")),
                      box(width = 6, plotlyOutput(outputId = "grafiekGaswinning")))

ui <- dashboardPage(header, sidebar, body)

#Serverfunctie
server <- function(input, output) {
  output$grafiekAantal <- renderPlotly({
    plot_ly(data = jaargegevens, x = ~ jaar, y = ~ aantal,
      type = 'scatter', mode = 'markers+lines',
      hoverinfo = "text", text = ~ hovertekst_aantal) %>%
      layout(xaxis = list(title = "Jaar"),
             yaxis = list(title = "Aantal bevingen")) %>%
      config(displayModeBar = F)
  })

  output$grafiekGaswinning <- renderPlotly({
    plot_ly(data = jaargegevens, x = ~ jaar, y = ~ gaswinning,
            type = 'scatter', mode = 'markers+lines', color = I('orange'),
            hoverinfo = "text", text = ~ hovertekst_winning) %>%
      layout(
        xaxis = list(title = "Jaar"),
        yaxis = list(title = "Gaswinning (miljard Nm<sup>3</sup>)")
      ) %>%
      config(displayModeBar = F)
  })
}

shinyApp(ui, server)
```

Pas de Shiny app aan, zodat maar één grafiek tegelijkertijd wordt getoond en de gebruiker door het aanklikken van een radiobutton kan switchen tussen de grafieken.

```
header <- dashboardHeader(title = "Bevingen in de tijd")

sidebar <- dashboardSidebar(disable = TRUE)

body <- dashboardBody(box(width = 6,
                          plotlyOutput(outputId = "grafiek"),
                          radioButtons(inputId = "y", label = "y-variabele:",
                                       choices = c("Aantal bevingen", "Gaswinning"),
                                       inline = TRUE)
))

ui <- dashboardPage(header, sidebar, body)

server <- function(input, output) {
  y <- reactive(
    case_when(
      input$y == "Aantal bevingen" ~ jaargegevens$aantal,
      input$y == "Gaswinning"      ~ jaargegevens$gaswinning
    )
  )

  hovertekst <- reactive(
    case_when(
      input$y == "Aantal bevingen" ~ jaargegevens$hovertekst_aantal,
      input$y == "Gaswinning"      ~ jaargegevens$hovertekst_winning
    )
  )

  titel <- reactive(
    case_when(
      input$y == "Aantal bevingen" ~ "Aantal bevingen",
      input$y == "Gaswinning"      ~ "Gaswinning (miljard Nm<sup>3</sup>)"
    )
  )

  output$grafiek <- renderPlotly({
    plot_ly(data = jaargegevens, x = ~ jaar, y = y(),
            type = 'scatter', mode = 'markers+lines',
            hoverinfo = "text", text = hovertekst()) %>%
      layout(xaxis = list(title = "Jaar"),
             yaxis = list(title = titel())) %>%
      config(displayModeBar = F)
  })
}

shinyApp(ui, server)
```

Het bestand `shiny_apps\3_tijdreeks_visualiseren\app.R` bevat alle code voor het visualiseren van de tijdreeksen van bevingen en gaswinning in een Shiny dashboard.