

Introductie op Ruimtelijke Data en het Gebruik van R als een GIS

Nick Bearman, bewerking Harrie Jonkman en Mirte Boelens

Introductie

Dit is een bewerking van de tutorial die Nick Bearman eerder schreef (Introduction to Spatial Data & Using R as a GIS) en die vrij toegankelijk is (Hiet Git adres). De data die in deze tutorial worden gebruikt zijn eigen data ook om de techniek goed onder de knie te krijgen. Dit maakt deel uit van een project waar

Eindtermen: R Functies & Bibliotheken:

- R gebruiken om CSV data in te lezen `read.csv()` (p. 3)
- R gebruiken om ruimtelijke gegevens in te lezen `st_read()` (p. 4)
- Weten hoe ruimtelijke gegevens te plotten met behulp van R `qtm()` (p. 5) & `tm_shape()` (p. 10)
- Weten hoe je kleuren en classificaties moet aanpassen `style` (p. 10)
- Begrijpen hoe je loops moet gebruiken om meerdere kaarten te maken `for(){}` (p. 13)
- Weten hoe je ruimtelijke gegevens opnieuw geprojecteerd krijgt `st_transform()` (p. 15)
- In staat zijn om punten te gebruiken in veelhoekanalyse `poly.counts()` (p. 16)
- Weten hoe je shapefiles moet opslaan `st_write()` (p. 18)

Praktijk 1: Intro op R & GIS

R Basis

R begon als een statistisch programma en wordt nog steeds door veel gebruikers als een programma gebruikt. We gaan een programma gebruiken dat RStudio heet, dat bovenop R werkt en een goede gebruikersinterface biedt. Ik zal het in de presentatie even hebben over RStudio, en de belangrijkste gebieden van het venster zijn op de achterzijde gemaarkeerd.

- Open RStudio (klik op Start en typ RStudio in of dubbelklik op het icoontje op het bureaublad). R kan in eerste instantie als rekenmachine worden gebruikt - voer het volgende in de linkerkant van het venster in - het gedeelte met de titel Console:

6 + 8

```
## [1] 14
```

Maak je voorlopig geen zorgen over de [1] - let wel dat R 14 heeft afgedrukt, want dit is het antwoord op de som die je hebt ingetikt. In deze werkbladen laat ik soms de resultaten zien van wat je hebt ingetypt, zoals hieronder:

5 * 4

```
## [1] 20
```

Merk ook op dat * hier het symbool voor vermenigvuldiging is - in het laatste commando vroeg R om de berekening 5 maal 4 uit te voeren. Andere symbolen zijn - voor aftrekken en / voor delen:

12 - 14

```
## [1] -2
```

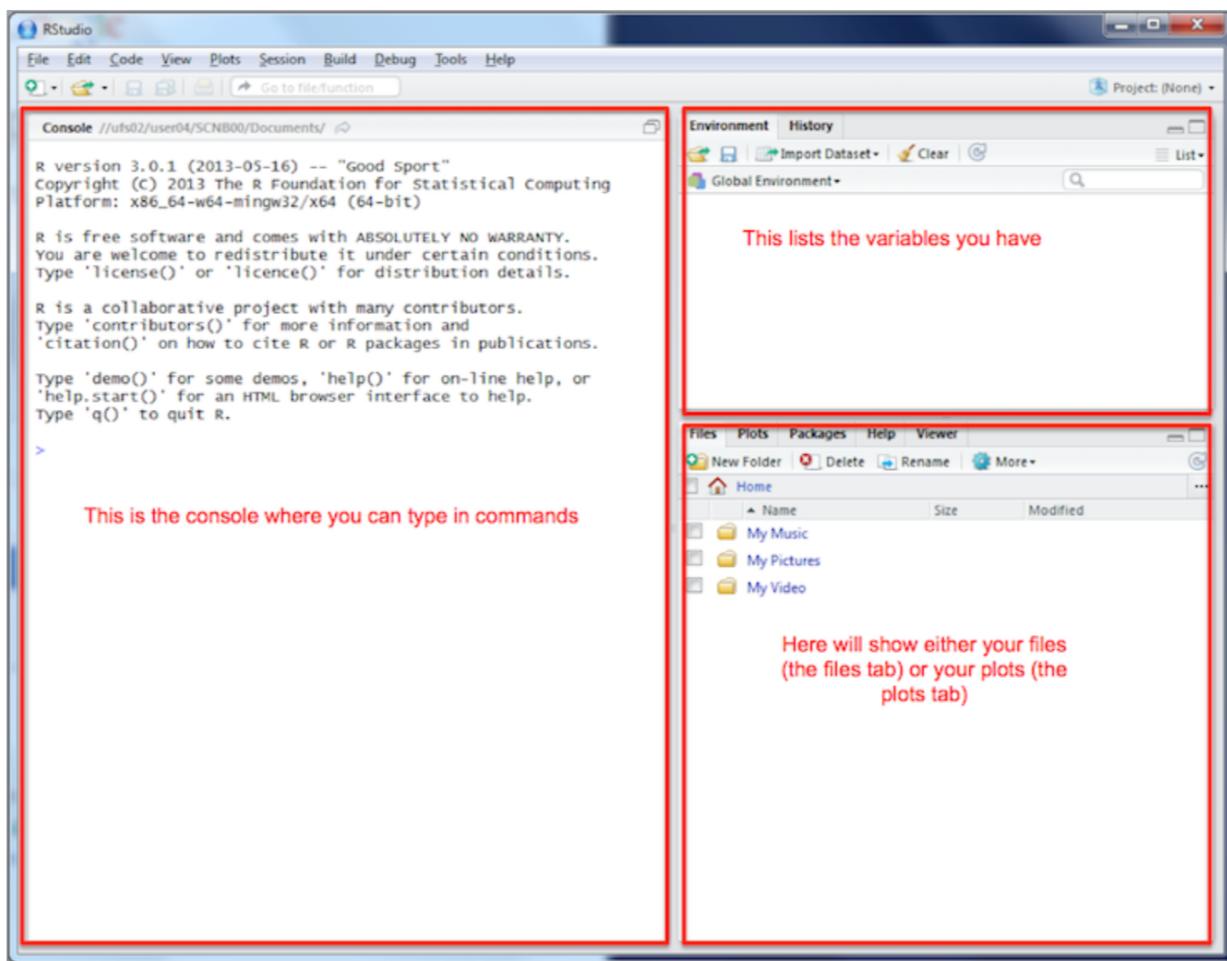


Figure 1: Screenshot of RStudio

6 / 17

```
## [1] 0.3529412
```

Je kunt de antwoorden van de berekeningen ook toewijzen aan variabelen en gebruiken in berekeningen.

```
price <- 300
```

Hier wordt de waarde 300 opgeslagen in de variabele prijs. Het `<-` symbool betekent dat de waarde rechts in de variabele links in de variabele wordt gezet, deze wordt getypt met een `<<` gevolgd door een `-`. De variabelen worden getoond in het venster met de naam Environment, rechtsboven in het venster. Variabelen kunnen gebruikt worden in volgende berekeningen. Om bijvoorbeeld een korting van 20% op deze prijs toe te passen, kunt je het volgende invoeren:

```
price - price * 0.2
```

```
## [1] 240
```

of gebruik tussenvariabelen:

```
discount<-price*0.2  
price-discount
```

```
## [1] 240
```

R kan ook werken met lijsten met nummers, maar ook met individuele nummers. Lijsten worden gespecificeerd met behulp van de c-functie. Stel dat je een lijst hebt met huizenprijzen in duizenden euro's. Je zou ze kunnen opslaan in een variabele die house.prices genoemd wordt, zoals hieronder:

```
house.prices<-c(120, 150, 212, 99, 199, 299, 159)  
house.prices
```

```
## [1] 120 150 212 99 199 299 159
```

Merk op dat er geen probleem is met punten in het midden van variabelenamen. U kunt dan functies toepassen op deze lijsten.

```
mean(house.prices)
```

```
## [1] 176.8571
```

Als de huizenprijzen in duizenden euro's zijn, dan zegt dit ons dat de gemiddelde huizenprijs 176.900 EURO bedraagt. Merk op dat het antwoord op jouw scherm meer cijfers kan weergegeven. Dus je kunt iets als 176.8571429 voor gemiddelde waarde hebben.

Het Dataframe

R heeft een manier om gegevens op te slaan in een object dat een dataframe wordt genoemd. Dit lijkt op een interne spreadsheet.

```
discount <- price * 0.2  
price - discount
```

```
## [1] 240
```

waar alle relevante gegevenselementen samen als een set kolommen worden opgeslagen.

We hebben een CSV-bestand van huizenprijzen en inbraakcijfers, dat we in R kunnen laden. We kunnen gebruik maken van een functie genaamd read.csv die, zoals je misschien wel kunt bedenken, CSV-bestanden leest. Voer de onderstaande coderegel uit, die het CSV-bestand in een variabele met de naam hp.data laadt.

```
hp.data<-read.csv("data/hpdata.csv")
```

Als we de gegevens inlezen, is het altijd een goed idee om te controleren of ze goed zijn binnengekomen. Om dit te doen, kunnen we een voorbeeld van de dataset bekijken. Het head-commando toont de eerste 6 rijen van de data.

HIER ONDER LEZEN WE DAN ONS DATABESTAND IN

```
head(hp.data)
```

```
##   ID Burglary Price  
## 1 21        0  200  
## 2 24        7  130  
## 3 31        0  200  
## 4 32        0  200  
## 5 78        6  180  
## 6 80       19  140
```

Je kunt ook op de variabele in het venster Environment klikken, die de gegevens in een nieuw tabblad zal tonen. Je kunt ook zelf invoeren en een tabblad openen met de gegevens:

```
View(hp.data)
```

Je kunt ook elke kolom in de dataset beschrijven met behulp van de **summary**-functie:

```
summary(hp.data)
```

```
##      ID      Burglary      Price
##  Min.   : 21.0   Min.   : 0.000   Min.   : 65.0
##  1st Qu.:615.5  1st Qu.: 0.000   1st Qu.:152.5
##  Median :846.5  Median : 0.000   Median :185.0
##  Mean   :654.0  Mean   : 5.644   Mean   :179.0
##  3rd Qu.:875.8  3rd Qu.: 7.000   3rd Qu.:210.0
##  Max.   :905.0  Max.   :37.000   Max.   :260.0
```

Voor elke kolom wordt een aantal waarden genoemd:

Item Beschrijving

Min. De kleinste waarde in de kolom 1st. Qu. Het eerste kwartiel (de waarde 1/4 van de variabele) Median De mediaan (de waarde 1/2 van de variabele) Mean Het gemiddelde van de kolom 3rd. Qu. Het derde kwartiel (de waarde 3/4 van de variabele) Max. De hoogste waarde in de kolom

Op basis van deze getallen kan een indruk worden verkregen van de spreiding van de waarden van elke variabele. Met name kan worden vastgesteld dat de mediaan van de huizenprijs in St. Helens per wijk varieert van 65.000 EURO tot 260.000 EURO en dat de helft van de prijzen tussen 152.500 EURO en 210.000 EURO ligt. Ook kan worden vastgesteld dat, aangezien de mediaan van het gemeten inbraakpercentage nul is, ten minste de helft van de gebieden geen inbraken had in de maand waarin de tellingen werden samengesteld..

We kunnen vierkante haken gebruiken om specifieke delen van het dataframe te bekijken, bijvoorbeeld hp.data[1,] of hp.data[,1]. We kunnen ook kolommen verwijderen en nieuwe kolommen aanmaken met behulp van de onderstaande code. Vergeet niet om het **head()** commando te gebruiken zoals we eerder deden om naar het dataframe te kijken.

```
#Creeer een nieuwe kolom in de hp.data dataframe en noem deze counciltax, en sla de waarde NA op
hp.data$counciltax <- NA
#Kijk wat er is gebeurd
head(hp.data)
```

```
##      ID      Burglary      Price counciltax
## 1 21          0     200        NA
## 2 24          7     130        NA
## 3 31          0     200        NA
## 4 32          0     200        NA
## 5 78          6     180        NA
## 6 80         19     140        NA

#Haal een kolom weg
colnames(hp.data)[3] <- "Price-thousands"
#Kijk wat er is gebeurd
head(hp.data)
```

```
##      ID      Burglary Price-thousands counciltax
## 1 21          0           200        NA
## 2 24          7           130        NA
## 3 31          0           200        NA
## 4 32          0           200        NA
```

```
## 5 78      6      180      NA
## 6 80      19     140      NA
```

Geografische Informatie

R heeft zich ontwikkeld tot een GIS waar gebruikers aan hebben bijgedragen met pakketten, of ‘libraries’, zoals R ze noemt. We zullen in de tutorial verschillende van dit soort ‘libraries’ gebruiken en zullen ze laden als dat nodig is.

Als u uw computer gebruikt, moet u de R-libraries installeren en ze ook laden. Om dit te doen, start u `install.packages ("library_name")`.

Om met ruimtelijke gegevens te kunnen werken, moeten we na dat installeren een aantal ‘libraries’ laden>

```
#Laden van libraries die we hier gebruiken
library(sf)
```

```
## Linking to GEOS 3.6.1, GDAL 2.1.3, PROJ 4.9.3
```

```
library(tmap)
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
```

Om met ruimtelijke gegevens te werken, moeten we enkele `libraries` laden. Daarmee is R echter alleen maar in staat om geografische data te verwerken. Het laadt nog geen specifieke data sets. Om dit te doen, moeten we enkele gegevens inlezen. Hiervoor gaan we **shapefiles** gebruiken - een bekend GIS-dataformat. We gaan LSOA(Lower layer Super Output Areas)-data gebruiken voor St. Helens in Merseyside.

R gebruikt werkmappen om informatie op te slaan die relevant is voor het huidige project waaraan u werkt. Ik stel voor dat je een map een bepaalde naam geeft die het R-werk ergens zinvol maakt. Dan moeten we R vertellen waar deze map staat, dus klik op **Session > Set Working Directory > Choose Directory**. . en selecteer de map die je hebt aangemaakt.

Zoals met de meeste programma’s, zijn er meerdere manieren om dingen te doen. Om bijvoorbeeld de werkmap in te stellen, kunnen we het volgende typen: `setwd("M:/R_werk")`. Jouw versie kan een langere titel hebben, afhankelijk van hoe je de map noemt. Merk ook op dat schuine streepjes worden aangegeven met een ‘/’ en niet ”.

Er is een set van shapefiles voor de St. Helens-wijken op dezelfde locatie als de dataset die je eerder hebt gelezen. Omdat er meerdere bestanden nodig zijn, heb ik deze in één zip-bestand gebundeld. Deze download je naar jouw lokale map en pakt deze vervolgens uit. Dit doe je met de volgende R-functies:

```
download.file("http://www.nickbearman.me.uk/data/r/sthelens.zip","sthelens.zip")
unzip("sthelens.zip")
```

De eerste functie downloadt het zip-bestand daadwerkelijk in uw werkmap. De tweede functie pakt het zip-bestand uit. Nu kunnen we het bestand in R lezen.

```
sthelens<-st_read("sthelens.shp")
```

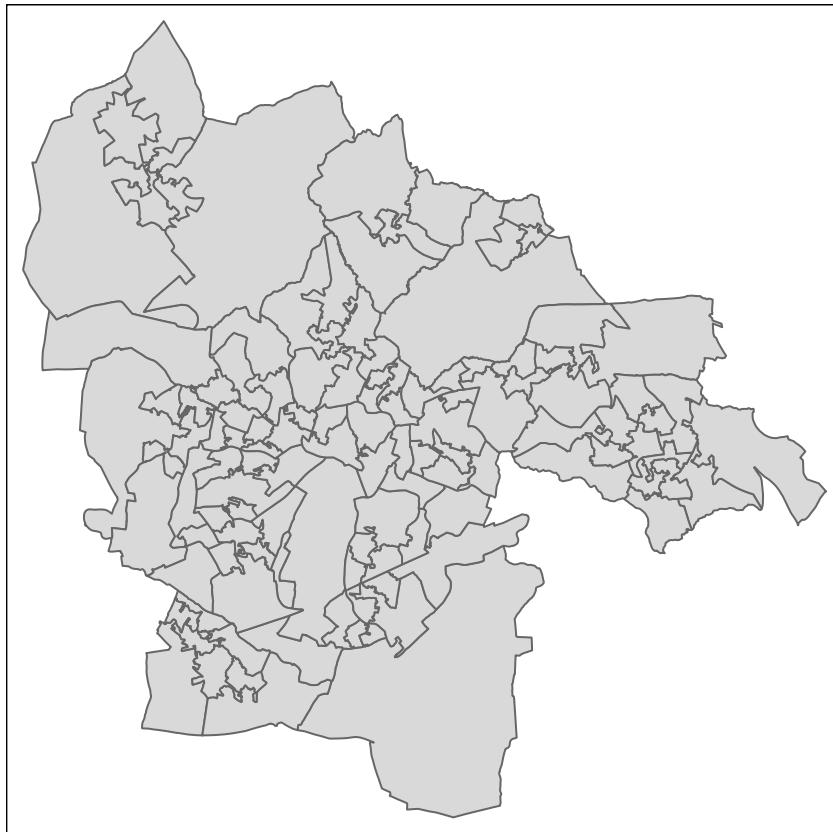
```

## Reading layer `sthelens` from data source `/Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/Geodaten/LSOAs/LSOA.shp'
## Simple feature collection with 118 features and 5 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:            xmin: 345350.2 ymin: 387836.6 xmax: 361798.8 ymax: 404145.6
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +datum=OSGB36

```

De `st_read` functie doet dit en slaat ze op als een Simple Features (of `sf`) object. Je kunt de `qtm`-functie gebruiken om de polygonen (d.w.z. de kaart van de LSOA) te tekenen.

```
qtm(sthelens)
```



We kunnen ook het `head()`-commando gebruiken om de eerste zes rijen te tonen, precies hetzelfde als bij een data frame.

```
head(sthelens)
```

```

## Simple feature collection with 6 features and 5 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:            xmin: 345350.2 ymin: 387836.6 xmax: 356013.9 ymax: 403054
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +datum=OSGB36
##   ID GID          NAME     LABEL ZONECODE
## 1 21 2049 St. Helens 009D 04BZE01006825 E01006825
## 2 24 2052 St. Helens 001B 04BZE01006883 E01006883
## 3 31 3564 St. Helens 022A 04BZE01006898 E01006898
## 4 32 3565 St. Helens 001D 04BZE01006885 E01006885

```

```

## 5 78 4389 St. Helens 023C 04BZE01006891 E01006891
## 6 80 4391 St. Helens 018C 04BZE01006828 E01006828
##           geometry
## 1 POLYGON ((348074.3 395549.9...
## 2 POLYGON ((347824.8 401169.7...
## 3 POLYGON ((351561.1 390368, ...
## 4 POLYGON ((351662.3 402905.2...
## 5 POLYGON ((348260 391439, 34...
## 6 POLYGON ((348188.1 392976.1...

```

Voor degene die met *GIS* werkt: Dit is hetzelfde als de attribuutentententabel in programma's als *ArcGIS*, *QGIS* of *MapInfo*. Als u het shapefile in *QGIS* of *ArcGIS* wilt openen om vast te stellen hoe het er zo'n beetje uit ziet, kunt u dat doen.

Je kunt zien dat er veel informatie beschikbaar is, inclusief de geometrie. Voor ons is het ID-veld belangrijk, en zien dat dit overeenkomt met het ID-veld in het *hp.data* bestand. We kunnen dit gebruiken om de twee datasets samen te voegen om de inbraakgegevens op de kaart te tonen.

Het idee is dat er in elke dataset een veld is dat we kunnen gebruiken om de twee samen te voegen; in dit geval hebben we het ID-veld in *sthèlens* en het ID-veld in *hp.data*.

```
sthèlens<-merge(sthèlens, hp.data)
```

Gebruik de head-functie om te controleren of de gegevens correct zijn samengevoegd.

```
head(sthèlens)
```

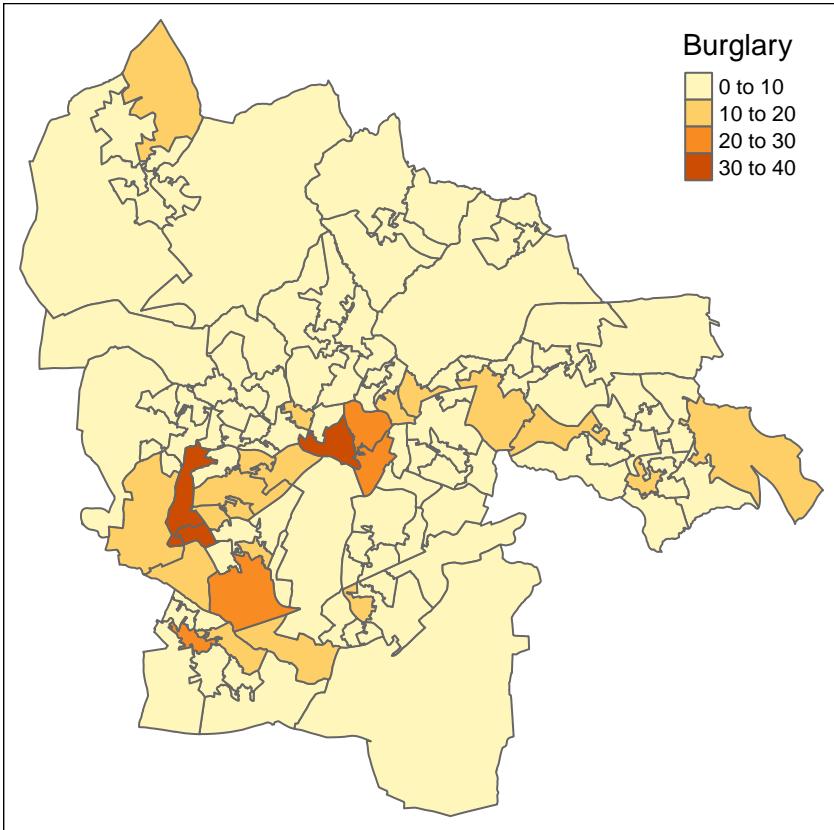
```

## Simple feature collection with 6 features and 8 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:            xmin: 346795.9 ymin: 389514.8 xmax: 351909.3 ymax: 404145.6
## epsg (SRID):   NA
## proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +datum=OSGB36
##   ID      GID      NAME      LABEL      ZONECODE      Burglary
## 1 100 4411 St. Helens 001C 04BZE01006884 E01006884      0
## 2 101 4412 St. Helens 001F 04BZE01006887 E01006887      0
## 3 102 4413 St. Helens 001E 04BZE01006886 E01006886     15
## 4 111 6849 St. Helens 023B 04BZE01006889 E01006889      6
## 5 112 6850 St. Helens 022E 04BZE01006910 E01006910     12
## 6 113 6851 St. Helens 019H 04BZE01006913 E01006913      8
##   Price-thousands      counciltax      geometry
## 1                 180      NA POLYGON ((348058.5 400721.8...
## 2                 200      NA POLYGON ((347413.4 401321, ...
## 3                 210      NA POLYGON ((348792.3 403164.7...
## 4                 170      NA POLYGON ((350108.1 390198.6...
## 5                 180      NA POLYGON ((351675.1 391422.1...
## 6                 160      NA POLYGON ((350807.5 393286.2...

```

Nu we de gegevens hebben samengevoegd, kunnen we een kaart maken van deze huizen-prijzen.

```
qtm(sthèlens, fill="Burglary")
```



Dit is een zeer snelle manier om een kaart met R te maken. Om de kaart te gebruiken, klikt u op de Export-knop en kiest u vervolgens voor Copy naar Clipboard. . . . Kies vervolgens Copy Plot. Als je ook Word hebt, kun je de kaart in je document plakken. Je kunt de kaart ook opslaan als Afbeelding of PDF.

Praktijk 2: Een Kaart maken Census Data

Werken met R vereist vaak meerdere coderegels code om een output te krijgen. In plaats van de code in de **Console** in te typen, kunnen we in plaats daarvan een script gebruiken. Daar kunnen we altijd naar teruggaan en de code zeer eenvoudig te bewerken, om fouten te corrigeren!

Maak een nieuw script aan (**File > New File > R-script**) en voer de code daar in. Vervolgens kunt je de regels die je wilt uitvoeren selecteren door ze te markeren en vervolgens op Ctrl+Enter te drukken, of door de **Run** knop bovenaan te gebruiken.

Nu gaan we hetzelfde principe gebruiken als voorheen om een kaart te maken van enkele gegevens uit de telling van 2011. We moeten de gegevens eerst downloaden. Hoewel er andere bronnen van deze gegevens zijn, zullen we in dit voorbeeld de website <https://www.nomisweb.co.uk/> gebruiken.

- Ga naar <https://www.nomisweb.co.uk/>.
- Onder **Census Statistics > 2011 Census** klik je op **Data catalogue**.
- Selecteer **Key Statistics (KS)**.
- Kies **KS102EW (Age Structure)**.
- Er zijn verschillende manieren om de data te downloaden - bekijk de verschillende opties maar eens. We gaan de tabel als een CSV file binnenhalen.

- Onder **Download (.csv)** kies je voor ‘super output areas - lower layer 2011’ van de drop down lijst.
- Klik op **Download**.
- Een file met de naam bulk.csv wordt binnengehaald - bewaar deze in jouw eigen workdirectory.

Open dit bestand in Excel en je kunt zien dat er een aantal verschillende kolommen zijn, die verschillende gegevens bevatten. Wij zijn geïnteresseerd in de leeftijdsgegevens - scroll over de verschillende waarden die ze hebben.

Voeg de onderstaande opdracht toe aan uw script, en voer het uit om in het CSV-bestand te lezen. De header = TRUE vertelt R om de eerste rij als variabele namen toe te wijzen.

```
pop2011<-read.csv("data/bulk.csv", header=TRUE)
```

Dan gebruik je de head-functie om te zien of de data goed zijn ingelezen. *R laat alle 23 variabelen zien, terwijl je in deze handout je alleen maar de eerste zes ziet.*

```
head(pop2011)
```

```
##   date      geography geography.code Rural.Urban
## 1 2011 Darlington 001B      E01012334      Total
## 2 2011 Darlington 001C      E01012335      Total
## 3 2011 Darlington 001D      E01012366      Total
## 4 2011 Darlington 001E      E01033481      Total
## 5 2011 Darlington 001F      E01033482      Total
## 6 2011 Darlington 002C      E01012323      Total
##   Age..All.usual.residents..measures..Value
## 1                               2466
## 2                               1383
## 3                               2008
## 4                               1364
## 5                               1621
## 6                               1563
##   Age..Age.0.to.4..measures..Value Age..Age.5.to.7..measures..Value
## 1                               161          134
## 2                                58          46
## 3                                72          61
## 4                               154          68
## 5                               153          82
## 6                               110          60
##   Age..Age.8.to.9..measures..Value Age..Age.10.to.14..measures..Value
## 1                               80          169
## 2                                19          81
## 3                               40          87
## 4                               33          61
## 5                               38          96
## 6                               31          110
##   Age..Age.15..measures..Value Age..Age.16.to.17..measures..Value
## 1                               38          54
## 2                               26          30
## 3                               30          60
## 4                               15          30
## 5                               21          41
## 6                               22          53
```

```

##   Age..Age.18.to.19..measures..Value Age..Age.20.to.24..measures..Value
## 1                      35                      75
## 2                      19                      47
## 3                      45                     95
## 4                      30                     77
## 5                      28                     71
## 6                      31                     83
##   Age..Age.25.to.29..measures..Value Age..Age.30.to.44..measures..Value
## 1                      69                     571
## 2                      35                     236
## 3                      74                     326
## 4                     165                    446
## 5                      87                     501
## 6                      92                     371
##   Age..Age.45.to.59..measures..Value Age..Age.60.to.64..measures..Value
## 1                     554                   190
## 2                     297                   130
## 3                     507                   183
## 4                     188                     31
## 5                     318                     58
## 6                     332                     72
##   Age..Age.65.to.74..measures..Value Age..Age.75.to.84..measures..Value
## 1                     194                   104
## 2                     215                   114
## 3                     264                   118
## 4                      53                     10
## 5                      92                     30
## 6                     115                   62
##   Age..Age.85.to.89..measures..Value Age..Age.90.and.over..measures..Value
## 1                      27                     11
## 2                      23                      7
## 3                      39                      7
## 4                      2                      1
## 5                      3                      2
## 6                     16                      3
##   Age..Mean.Age..measures..Value Age..Median.Age..measures..Value
## 1                     39.1                  41.5
## 2                     46.2                  49.0
## 3                     44.7                  48.0
## 4                     30.5                  31.0
## 5                     34.0                  35.0
## 6                     37.3                  38.0

```

Sommige variabelenamen worden niet duidelijk weergegeven. We kunnen de kolommen hernoemen, zodat wanneer we de `head`-commando uitvoeren, R de juiste namen weergeeft. Dit zal ons ook helpen om later naar de kolommen te verwijzen.

Voer onderstaande code uit, die een nieuwe variabele maakt die de namen ('newcolnames') bevat en deze vervolgens toepast op het pop2011-dataframe.

Het is goed om hier op te merken dat elke coderegel die begint met een # een commentaar is - d.w.z. dat R die regel zal negeren en naar de volgende regel zal gaan. Ik heb ze hier opgenomen zodat je kunt zien wat er aan de hand is, maar je hoeft ze niet in te typen.

Creeer een nieuwe variabele die de nieuwe namen van de variabelen bevat

```
newcolnames <- c("AllUsualResidents", "Age00to04", "Age05to07", "Age08to09", "Age10to14", "Age15", "Age16t")
```

```
#Pas deze toe op de pop2011 data frame  
colnames(pop2011)[5:23] <- newcolnames
```

Ga naar de volgende. Ik heb ze hier opgenomen zodat je kunt zien wat er aan de hand is, maar je hoeft ze niet in te typen.

De laatste coderegel (start colnames) werkt de namen van de variabelen bij. De vierkante haakjes worden gebruikt om te verwijzen naar specifieke elementen - in dit geval, kolommen 5 tot 23. *Bijvoorbeeld, pop2011[1,] toont de eerste rij en pop2011[,1] toont de eerste kolom.*

Nu hebben we de juiste kolomnamen voor het dataframe. Het zou ook goed zijn om te controleren of ze correct zijn toegepast op het pop2011 dataframe. **Welke code zou je gebruiken om dit te doen?*

```
date geography geography.code Rural.Urban AllUsualResidents 1 2011 Darlington 001B E01012334 Total  
2466 2 2011 Darlington 001C E01012335 Total 1383 3 2011 Darlington 001D E01012366 Total 2008 4 2011  
Darlington 001E E01033481 Total 1364 5 2011 Darlington 001F E01033482 Total 1621 6 2011 Darlington  
002C E01012323 Total 1563 Age00to04 Age05to07 Age08to09 Age10to14 1 161 134 80 169 2 58 46 19 81 3 72  
61 40 87 4 154 68 33 61 5 153 82 38 96 6 110 60 31 110
```

Nu we de attribuutgegevens (in dit geval het aantal personen per leeftijdsgroep in elke LSOA) hebben, moeten we deze attribuutgegevens toevoegen aan de ruimtelijke gegevens. Daarom moeten we eerst de ruimtelijke gegevens downloaden.

- Ga naar <http://census.edina.ac.uk/> en selecteer Boundary Data Selector.
- Stel vervolgens Country op England in, Geografie op Statistical Building Block, data op 2011 en later, en klik op Find.
- Selecteer English Lower Layer Super Output Areas, 2011 en klik op List Areas.
- Selecteer Liverpool uit de lijst en klik op Extract Boundary Data.
- Na 5 tot 20 seconden wachten klikt u op BoundaryData.zip om de bestanden te downloaden.

Pak de bestanden uit en verplaats alle bestanden die beginnen met de naam england_lsoa_2011 naar uw werkmap. Lees vervolgens de gegevens in:

```
#Lees de shapefile in  
LSOA <- st_read("england_lsoa_2011.shp")
```

```
## Reading layer `england_lsoa_2011' from data source `/Users/harriejonkman/Library/Mobile Documents/com  
## Simple feature collection with 298 features and 3 fields  
## geometry type: POLYGON  
## dimension: XY  
## bbox: xmin: 332390.2 ymin: 379748.5 xmax: 345636 ymax: 397980.1  
## epsg (SRID): NA  
## proj4string: +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +datum=OSGB36
```

Zoals eerder, kunnen we het `qtm()` commando gebruiken om een voorbeeld van de kaart te bekijken. We kunnen ook kijken naar de attribuut tabel met `head()`. Probeer deze nu allebei.

De volgende stap is om de attribuutgegevens aan de ruimtelijke gegevens toe te voegen, zoals we eerder in de oefening hebben gedaan. Kijk of je kunt zien hoe en waarom ik de code van eerder heb veranderd.

De volgende stap is om de attribuutgegevens aan de ruimtelijke gegevens toe te voegen, zoals we eerder in de oefening hebben gedaan. Kijk of je kunt zien hoe en waarom ik de code van eerder heb veranderd.

```
#Koppel attributedata aan LSOA  
LSOA <- merge(LSOA, pop2011, by.x="code", by.y="geography.code")
```

En we gebruiken het hoofd-commando om te controleren of de koppeling goed is verlopen. Jouw data moet de goed gelabelde ‘Age’-data in de 7de tot 26ste kolom bevatten.

```
head(LSOA)
```

```
## Simple feature collection with 6 features and 25 fields
## geometry type: POLYGON
## dimension: XY
## bbox: xmin: 334715 ymin: 385417 xmax: 339020.8 ymax: 390548
## epsg (SRID): NA
## proj4string: +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +datum=OSGB36
## code label name date geography
## 1 E01006512 E08000012E02001377E01006512 Liverpool 031A 2011 Liverpool 031A
## 2 E01006513 E08000012E02006932E01006513 Liverpool 060A 2011 Liverpool 060A
## 3 E01006514 E08000012E02001383E01006514 Liverpool 037A 2011 Liverpool 037A
## 4 E01006515 E08000012E02001383E01006515 Liverpool 037B 2011 Liverpool 037B
## 5 E01006518 E08000012E02001390E01006518 Liverpool 044A 2011 Liverpool 044A
## 6 E01006519 E08000012E02001402E01006519 Liverpool 056A 2011 Liverpool 056A
## Rural.Urban AllUsualResidents Age00to04 Age05to07 Age08to09 Age10to14
## 1 Total 1880 98 43 15 35
## 2 Total 2941 23 11 6 12
## 3 Total 2108 39 13 12 38
## 4 Total 1208 54 42 14 63
## 5 Total 1696 92 45 25 92
## 6 Total 1286 70 38 21 86
## Age15 Age16to17 Age18to19 Age20to24 Age25to29 Age30to44 Age45to59
## 1 6 34 124 615 332 355 158
## 2 1 14 624 1366 393 279 115
## 3 3 13 196 586 271 474 268
## 4 12 18 35 141 139 272 178
## 5 17 42 46 120 108 320 350
## 6 16 46 28 56 50 225 335
## Age60to64 Age65to74 Age75to84 Age85to89 Age90andOver MeanAge MedianAge
## 1 31 17 13 3 1 27.4 24
## 2 36 26 23 7 5 25.3 21
## 3 66 81 33 9 6 32.8 27
## 4 58 80 73 27 2 37.8 33
## 5 101 142 145 34 17 41.9 42
## 6 97 112 85 12 9 41.9 45
## geometry
## 1 POLYGON ((336203 390010, 33...
## 2 POLYGON ((335402.8 390317.5...
## 3 POLYGON ((335651.3 389926.8...
## 4 POLYGON ((335186 389604, 33...
## 5 POLYGON ((335537.2 389034.5...
## 6 POLYGON ((338014.6 386447.2...
```

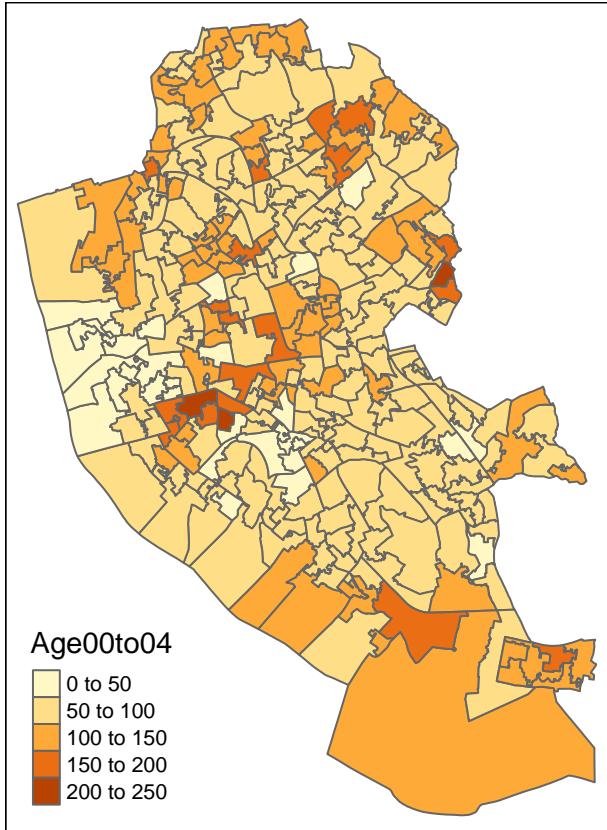
Kaarten maken

Nu we alle data hebben ingesteld, kunnen we de kaart daadwerkelijk maken. We kunnen de `qtm()` code gebruiken, net zoals we eerder deden.

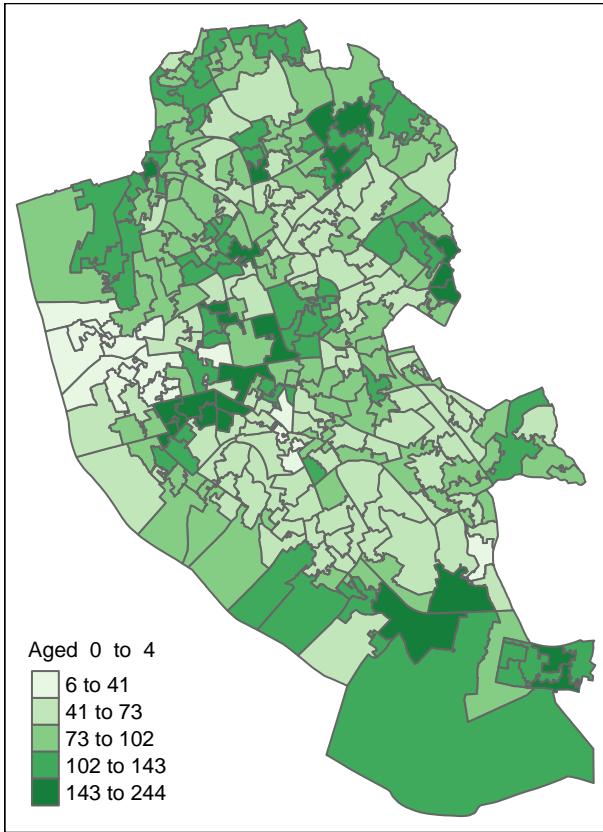
We kunnen de `fill`parameter gebruiken zoals we eerder deden met de inbraakgegevens. Probeer de code zelf uit te werken om de eerste set leeftijdsgegevens te tonen.

Dit werkt goed, en we kunnen kiezen welke variabele we willen tonen. Maar we krijgen hier niet veel opties mee. We kunnen een andere functie `tm_shape()` gebruiken, die ons meer opties geeft.

```
tm_shape(LSOA) +
  tm_polygons("Age00to04")
```



```
tm_shape(LSOA) +
  tm_polygons("Age00to04", title = "Aged 0 to 4", palette = "Greens", style = "jenks") + tm_
```

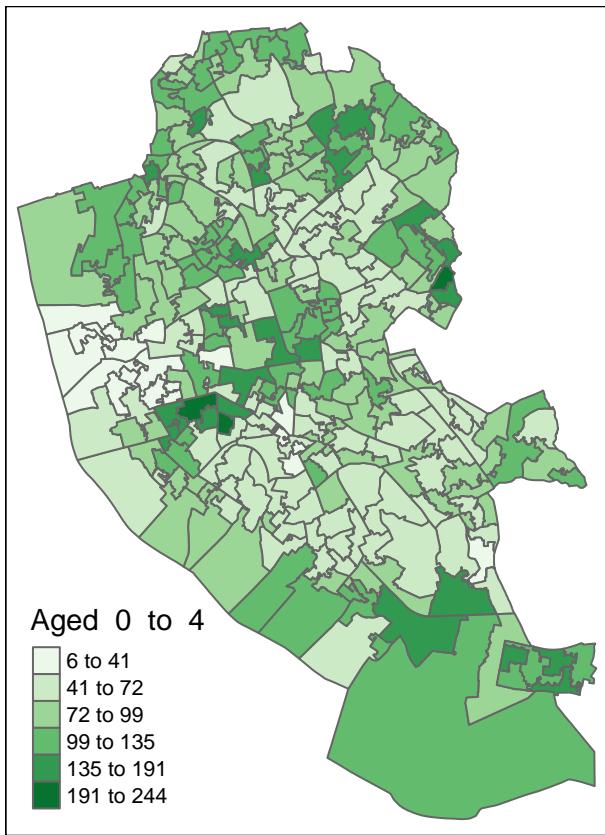


Dit stelt ons in staat om de titel, de kleuren en de grootte van de legende te veranderen. Probeer Blues te vervangen in Blues en de titel aan te passen.

Kleuren en categoriën

We kunnen kiezen uit veel verschillende kleuren van **ColorBrewer**, en ook verschillende indelingsmethoden. Om alle verschillende kleurenpaletten te tonen die we kunnen gebruiken, kunt u deze code gebruiken:

```
tm_shape(LSOA) +
  tm_polygons("Age00to04", title = "Aged 0 to 4", palette = "Greens", n = 6, style = ...)
```



We kunnen ook de stijl instellen die de indelingsmethode is. Standaard opties zijn:

Classificatie Naam Code Details of Voorbeeld

Gelijk Interval equal **Gelijke intervallen bv 0-5, 5-10, 10-15, 15-20**

Quantielen quantielDeel de data op in 5 equal categorieen, met hetzelfde aantal datapunten in elke categorie

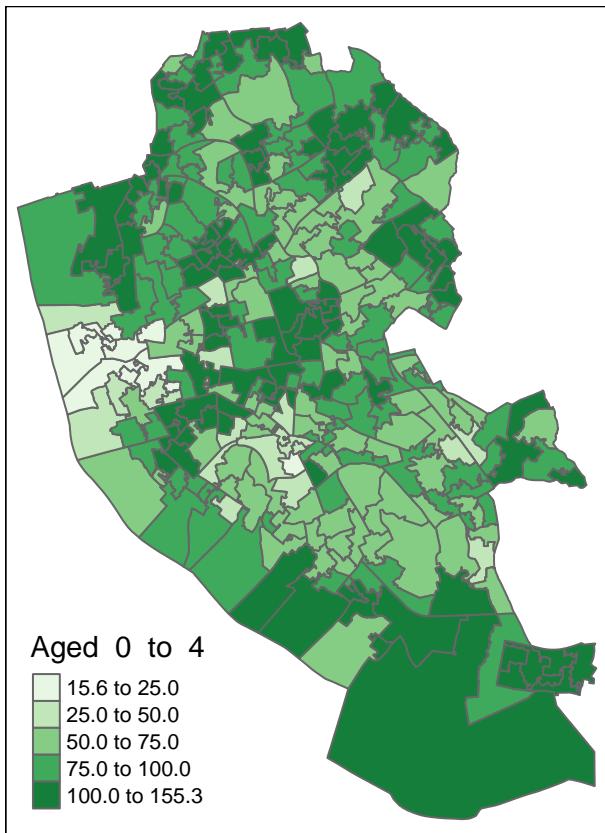
Natuurlijke Breuken jenks Algoritme om datagedreven categorien te maken

Standaard Deviatie sd Baseert klassen op data standaard deviatie bv -2SD naar -1SD, -1SD naar 0, 0 naar 1SD, 1SD naar 2SD

Vaste Breuken fixed Jij kiest de breuken - zie hieronder

Vast breuken voorbeeld:

```
tm_shape(LSOA) +
  tm_polygons("Age00to04", title = "Aged 0 to 4", palette = "Greens", n = 6, style = "fixed")
```

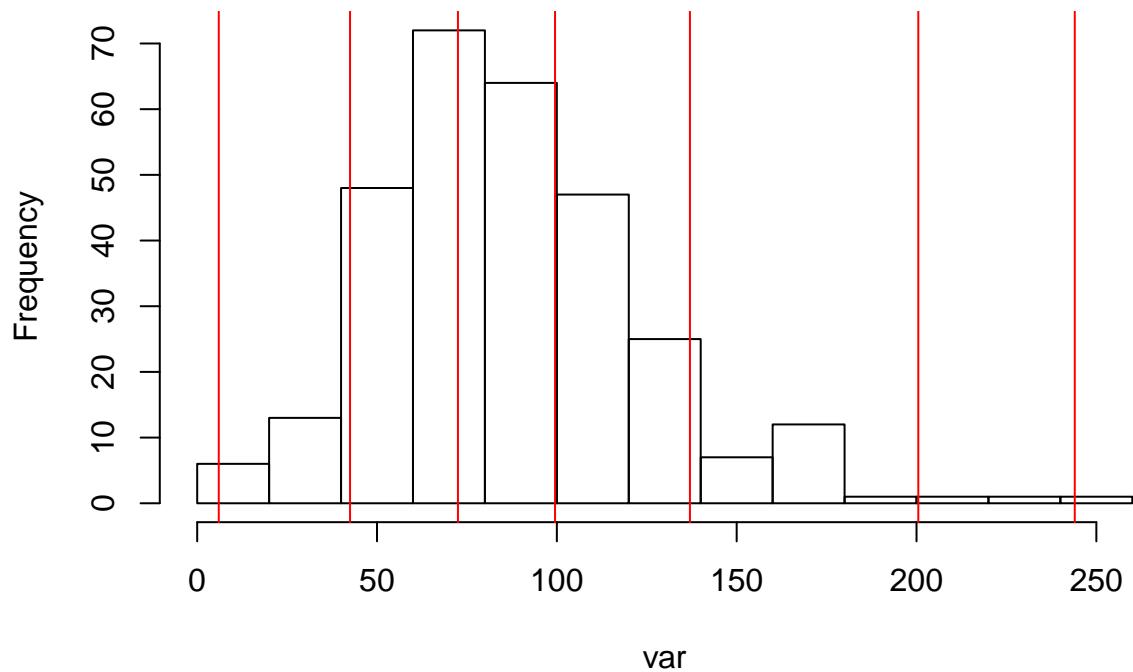


Classificatie op een Histogram (optionele oefening)

U kunt met deze code ook een histogram met de classificatiebreuken laten zien::

```
#Selecteer de variabele
var <- LSOA$Age00to04
#Calculeer de breuken
library(classInt)
breaks <- classIntervals(var, n = 6, style = "fisher")
#Teken een histogram
hist(var)
#Voeg de breuken toe aan de histogram
abline(v = breaks$brks, col = "red")
```

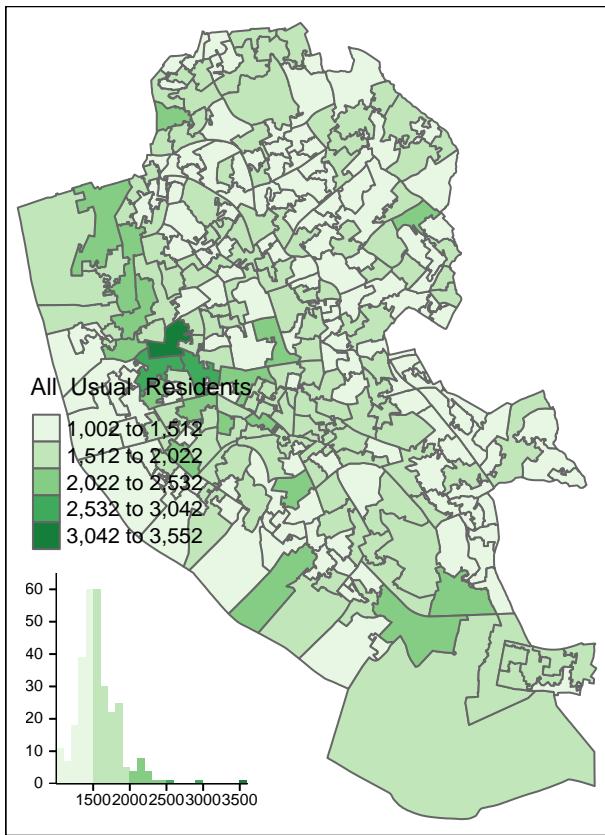
Histogram of var



Histogrammen

We kunnen ook een histogram van de gegevens aan de kaart toevoegen:

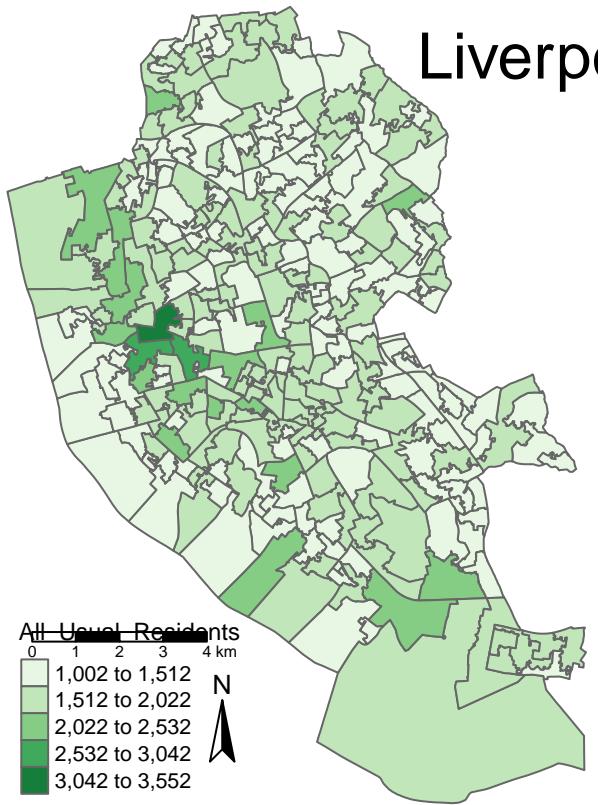
```
tm_shape(LSOA) +
  tm_polygons("AllUsualResidents", title = "All Usual Residents", palette = "Greens", style = "eq")
```



Schaalbalk en een pijl naar het Noorden (optionele oefening)

Het is ook een goede gewoonte om een schaalbalk en een pijl naar het noorden toe te voegen aan de kaarten die je maakt. Als je deze code runt worden die aan de kaart toegevoegd.

```
tm_shape(LSOA) +
#Stel kleuren en classificatiemethodes in
tm_polygons("AllUsualResidents", title = "All Usual Residents", palette = "Greens", style = "eq")
#Voeg schaalbalk toe
tm_scale_bar(width = 0.22, position = c(0.05, 0.18)) +
#Voeg kompas toe
tm_compass(position = c(0.3, 0.07)) +
#Stel layout details in
tm_layout(frame = F, title = "Liverpool", title.size = 2, title.position = c(0.7, "top"))
```



Mogelijk moet u de positie van de items op de kaart aanpassen. Probeer “tm_scale_bar positie” te googelen voor informatie over hoe dit te doen.

Vergeet niet dat elke regel die begint met een # een commentaar is, en zal worden genegeerd door R. Commentaar is erg nuttig voor ons om op te merken wat de code doet, vooral als je er 6 maanden later op terugkomt en je je niet kunt herinneren wat het moet doen!

Exporteren en het maken van meerdere kaarten

We kunnen de kaart automatisch opslaan als een bestand door het kaartobject aan te maken als een nieuwe variabele (m) en het vervolgens op te slaan met tmap_save(m).

```
#Creeer de kaart
m <- tm_shape(LSOA) +
  tm_polygons("AllUsualResidents", title = "All Usual Residents", palette = "Greens", style = "eq")
  tm_scale_bar(width = 0.22, position = c(0.05, 0.18)) + tm_compass(position = c(0.3, 0.07)) +
  tm_layout(frame = F, title = "Liverpool", title.size = 2, title.position = c(0.7, "top"))
#Sla de kaart op
tmap_save(m)

## Map saved to /Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/GithubHarrie/GISRotterdam/Map/LSOA/AllUsualResidents/AllUsualResidents.tmap
## Resolution: 1950 by 1350 pixels
## Size: 6.5 by 4.5 inches (300 dpi)
```

Door de kaart op te slaan met behulp van een code kunnen we heel eenvoudig meerdere kaarten maken. Een variabele (mapvariables) wordt gebruikt om een lijst te maken van de variabelen die in kaart moeten worden gebracht en dan begint de lijn die begint met een lus. Probeer de code uit te voeren en verander dan de variabelen die het in kaart brengen.

```

#Stel eerst vast welke variabelen in de kaart komen
mapvariables <- c("AllUsualResidents", "Age00to04", "Age05to07")
#Loop door elk van die kaarten
for (i in 1:length(mapvariables)) {
  #Definieer de kaart
  m <- tm_shape(LSOA) +
    #Stel de variabelen, kleuren en klassen vast
    tm_polygons(mapvariables[i], palette = "Greens", style = "equal") +
    #Stel de schaalbalk vast
    tm_scale_bar(width = 0.22, position = c(0.05, 0.18)) +
    #Stel het kompas in
    tm_compass(position = c(0.3, 0.07)) +
    #Stel de layout vast
    tm_layout(frame = F, title = "Liverpool", title.size = 2, title.position = c(0.7, "top"))
  #Sla de kaart op
  tmap_save(m, filename = paste0("map-", mapvariables[i], ".png"))
  #Einde van de loop
}

## Map saved to /Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/GithubHarrie/GISRotte...
## Resolution: 1950 by 1350 pixels
## Size: 6.5 by 4.5 inches (300 dpi)
## Map saved to /Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/GithubHarrie/GISRotte...
## Resolution: 1950 by 1350 pixels
## Size: 6.5 by 4.5 inches (300 dpi)
## Map saved to /Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/GithubHarrie/GISRotte...
## Resolution: 1950 by 1350 pixels
## Size: 6.5 by 4.5 inches (300 dpi)

Je kunt .png veranderen in .jpg of .pdf voor andere fileformaten.

```

Creëer een eenvoudige kaart titel (optionele oefening)

Zoals je misschien hebt gezien, zijn de kaart- en legendetitels niet geweldig. Alleen al het gebruik van de veldnaam geeft ons de informatie die we nodig hebben. Het garandeert nog niet dat de kaart er goed uitziet.

Hoe kunnen we een betere kaarttitel maken?

Doe een experiment en kijk of je het kunt uitwerken. Er is wat code beschikbaar (in script-examples/script-multiple-maps) maar probeer niet meteen te kijken!

Praktijk 3: Clustering van Criminaliteits Punten

In dit deel kijken we naar enkele puntdata en hoe we deze kunnen analyseren in R. Hiervoor laden we misdaaddata in. Omdat deze misdaaddata slechts een CSV-bestand is, moeten we een aantal bewerkingen uitvoeren om ze als ruimtelijke data in R te krijgen, inclusief het opnieuw projecteren van de data (het omzetten van het coördinatensysteem van WGS 1984 naar BNG).

```

#Lees de data in als een variable die we crimes noemen
crimes <- read.csv("http://nickbearman.me.uk/data/r/police-uk-2018-12-merseyside-street.csv")

```

Nu de CSV-data zijn ingelezen, bekijk ze snel met de `head()`-functie. U zult zien dat de data bestaan uit een aantal kolommen, elk met een koptekst. Twee daarvan heten Longitude (Lengtegraad) en Latitude (Breedtegraad) - dit zijn de kolomkoppen die de coördinaten van elk incident weergeven in de gegevens die je zojuist hebt gedownload. Een andere is getiteld Crime.Type en vertelt je welk type misdaad zich heeft voorgedaan.

Op dit moment bevinden de gegevens zich alleen in een datakader-object - geen enkel ruimtelijk object. Om het ruimtelijk object te maken, voert u het volgende in

```
#Creeer crimes data
crimes_sf <- st_as_sf(crimes, coords = c('Longitude', 'Latitude'), crs = "+init=epsg:4326")
```

We nemen de misdaadgegevens, vertellen R welke kolommen coördinaten hebben en zeggen dat ze in WGS 1984 staan (+init=epsg:4326).

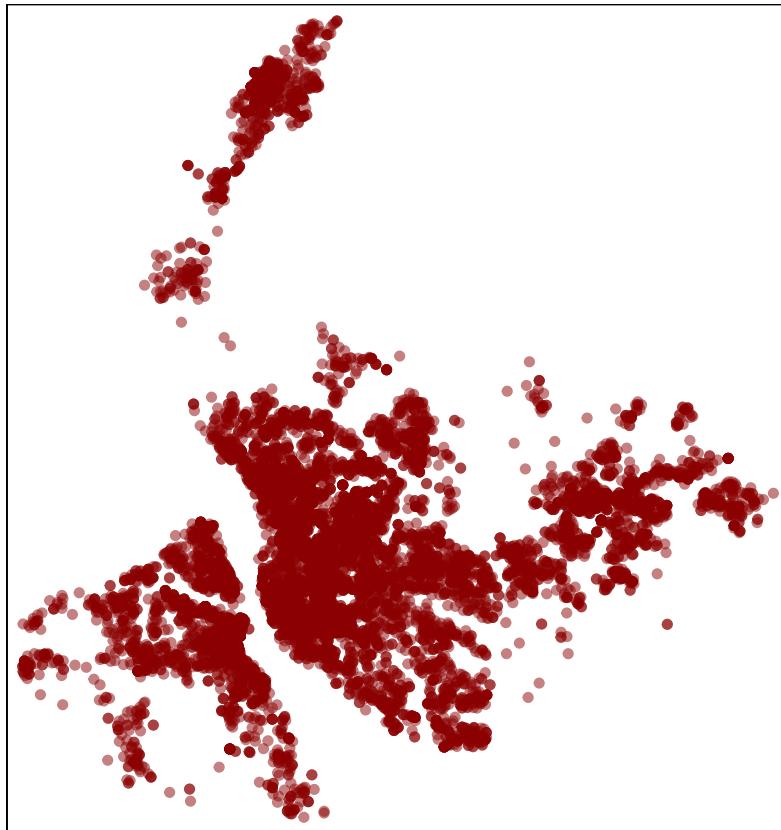
Als je een fout krijgt: ongeldig crs: +init=epsg:4326, reden: geen systeemlijst, fout: 2 of vergelijkbaar, dubbele controle op typefouten. Typfouten kunnen vaak epsg binnen sluipen!

Gebruik de `head()`-functie om te controleren of de gegevens correct zijn verwerkt. We kunnen ook `qtm()` gebruiken om een kaart te tekenen.

```
#Reprojecteer naar British National Grid, van Latitude/Longitude
crimes_sf_bng <- st_transform(crimes_sf, crs = "+init=epsg:27700")
```

Om het geografische patroon van deze misdaden te zien, voer je het volgende uit:

```
tm_shape(crimes_sf_bng) +
  tm_dots(size = 0.1, shape = 19, col = "darkred", alpha = 0.5)
```



We zien zo'n grofweg de kaders van de Merseyside Police area, net als het pad waar de rivier Mersey loopt. Je kunt de verschillende vormen van criminaliteit nu onderzoeken.

```



```

Punt in Polygon

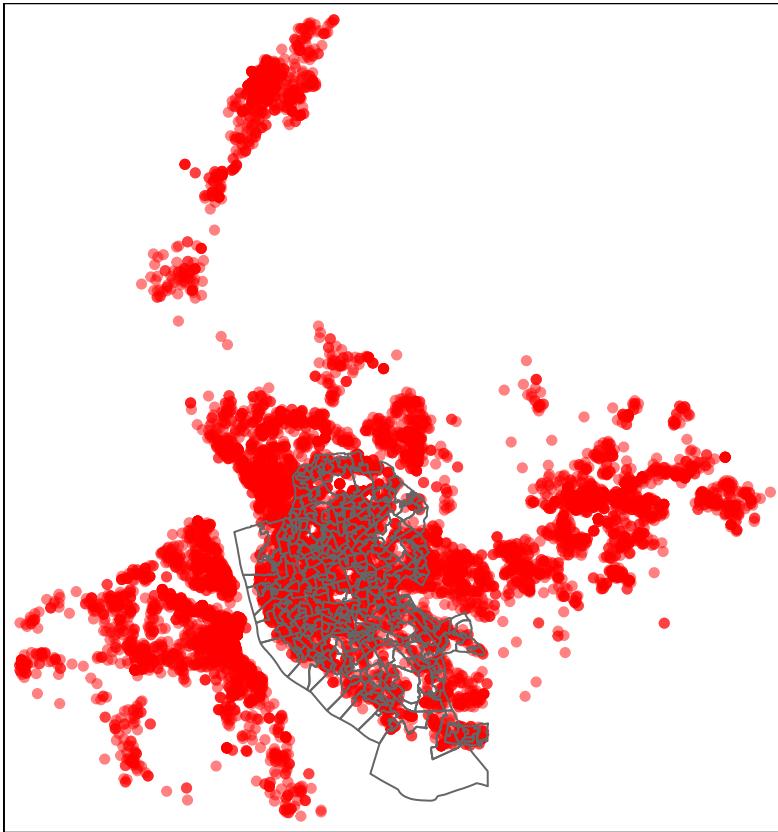
Het hebben van ruimtelijke lokaties van criminaliteit is natuurlijk fantastisch, maar het is goed dit te combineren met andere data om het enigszins te kunnen begrijpen.

Om te beginnen, leggen we LSOA die we eerder hebben gebruikt over de misdaden. We kunnen de misdaden plotten zoals we eerder hebben gedaan. We kunnen de LSOAdata er bovenop leggen.

```

#plot the crime data
tm_shape(crimes_sf_bng) +
tm_dots(size = 0.1, shape = 19, col = "red", alpha = 0.5) +
#add LSOA on top
tm_shape(LSOA) +
tm_borders()

```



Dit helpt niet echt omdat R wat je ziet in R wordt bepaald door de eerste data set die geplot wordt. Probeer eerst eens LSO te plotten en dan pas de laag misdaad.

Dit is nog steeds slechts twee lagen over elkaar heen, maar hopelijk zie je nu wel wat er gebeurt. Een algemene GIS-function is ‘point-in-polygon’ die ons in staat stelt te tellen hoeveel misdaden er in elke LSOA gerapporteerd worden.

Dat is iets dat nog steeds niet (makkelijk) in SF werkt dus we moeten het in SP doen.

```
#Laad bibliotheken
library(rgdal) #voor readOGR()

## Loading required package: sp
## rgdal: version: 1.4-4, (SVN revision 833)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 2.1.3, released 2017/20/01
## Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/gdal
## GDAL binary built with GEOS: FALSE
## Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
## Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.5/Resources/library/rgdal/proj
## Linking to sp version: 1.3-1

library(GISTools) #voor poly.counts()

## Loading required package: maptools
## Checking rgeos availability: TRUE
## Loading required package: RColorBrewer
## Loading required package: MASS
```

```

## 
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
## 
##     select

## Loading required package: rgeos

## rgeos version: 0.4-2, (SVN revision 581)
## GEOS runtime version: 3.6.1-CAPI-1.10.1
## Linking to sp version: 1.3-1
## Polygon checking: TRUE

#Lees shapefile in in SP formaat
LSOA_sp <- readOGR(".", "england_lsoa_2011")

## OGR data source with driver: ESRI Shapefile
## Source: "/Users/harriejonkman/Library/Mobile Documents/com~apple~CloudDocs/GithubHarrie/GISRotterdam
## with 298 features
## It has 3 fields

#Zet crimes SF naar sp
#(zie https://cran.r-project.org/web/packages/sf/vignettes/sf2.html#
#  conversie_naar_andere_formaten:_wkt,_wkb,_sp voor details)
crimes_sf_bng <- as(crimes_sf_bng, "Spatial")
#Calculeer het aantal misdaden in elk LSOA
crimes.count <- poly.counts(crimes_sf_bng, LSOA_sp)

## Warning in RGEOSBinPredFunc(spgeom1, spgeom2, byid, func): spgeom1 and
## spgeom2 have different proj4 strings

#Voeg dit als een nieuw veld toe aan onze LSOA data
LSOA_sp@data$crimes.count <- crimes.count

```

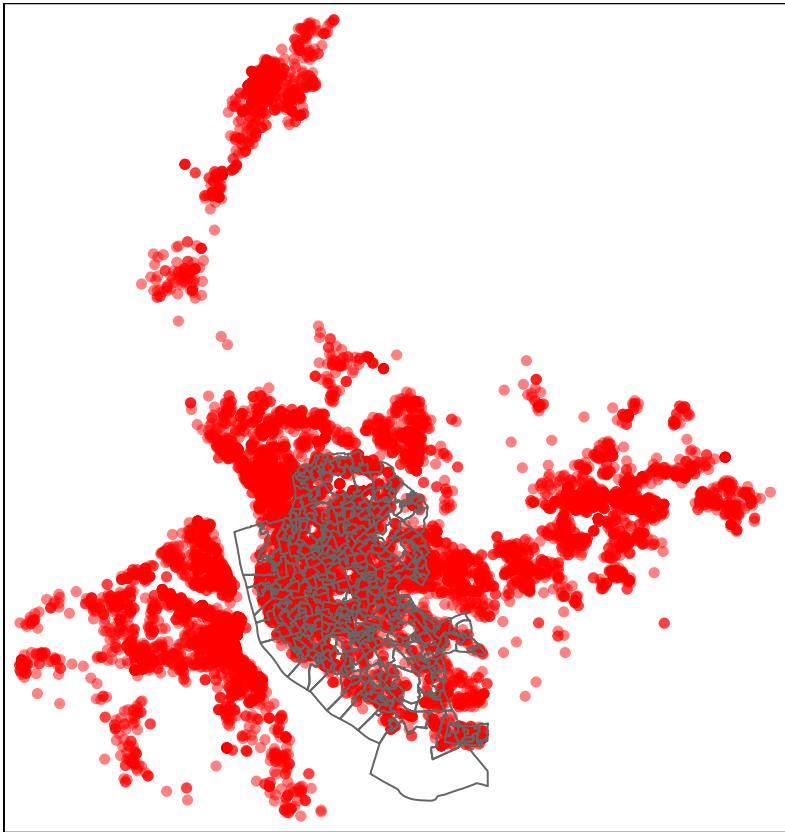
De LSOA_sp data set is nu in een groot SpatialPolygonsDataFrame (dat is de list in het venster ‘environment’ die van de SP-bibliotheek is. Dit betekent dat we de functie head() niet op dezelfde manier kunnen gebruiken in the same way. In plaats daarvan kunnen we head@data() gebruiken.

```
#Laat de eerste 6 rijen zien (precies zoals head(LSOA_sp) maar nu voor SP data)
head(LSOA_sp@data)
```

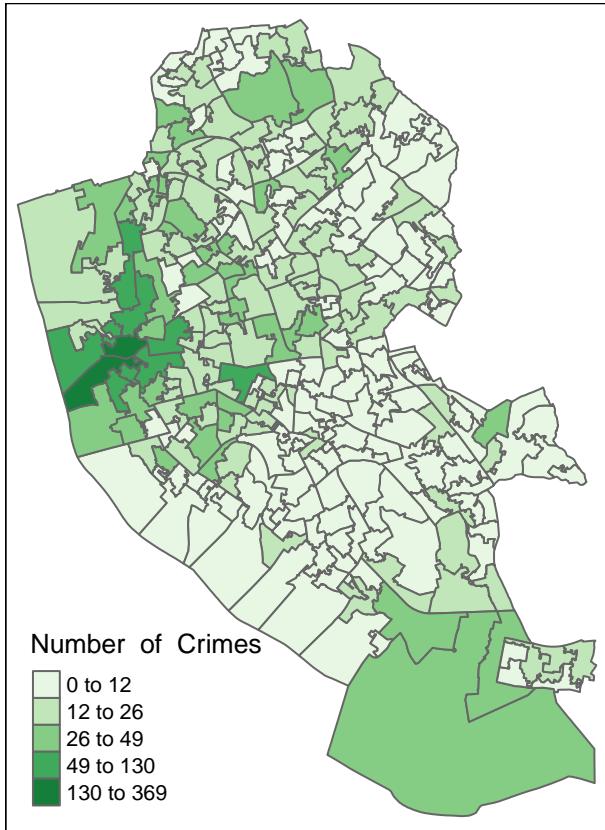
	label	name	code	crimes.count
## 0	E08000012E02006934E01033755	Liverpool	062D E01033755	14
## 1	E08000012E02006932E01033758	Liverpool	060B E01033758	30
## 2	E08000012E02001356E01033759	Liverpool	010F E01033759	25
## 3	E08000012E02006932E01033762	Liverpool	060E E01033762	65
## 4	E08000012E02001396E01032505	Liverpool	050F E01032505	2
## 5	E08000012E02001396E01032506	Liverpool	050G E01032506	10

Gelukkig, tmap kan SP of SF data plotten, dus we kunnen dezelfde code gebruiken als hiervoor:

```
#tmap
tm_shape(crimes_sf_bng) +
tm_dots(size = 0.1, shape = 19, col = "red", alpha = 0.5) +
#Plaats LSOA bovenaan
tm_shape(LSOA_sp) + tm_borders()
```



```
tm_shape(LSOA_sp) +  
  tm_polygons("crimes.count", title = "Number of Crimes", palette = "Greens", style = "jenks")
```



Shapefiles exporteren

We gebruikten `st_read` om een shapefile in te lezen en we kunnen `st_write` gebruiken om een shape file te exporteren:

```
# dataset bestaat reeds
st_write(LSOA, "LSOA-crime-count.shp")
```

Misdaadpercentage calculeren (optionele oefening)

Dit vertelt ons hoeveel misdaden er in elke LSOA zijn - niets over het percentage. Probeer het percentage misdaden per 10,000 mensen te berekenen door de bevolkingsdata van hiervoor te gebruiken.

We kunnen het percentage als volgt berekenen: aantal misdaden / bevolking) * 10000. Zie het begin van de handout om te zien hoe je deze waarden berekent. Onthoud wel dat je expliciet moet refereren naar het dataframe en de variabele (LSOACrime.count) elke keer als je een kolom benoemt, b.v. (LSOA@datacrime.count / LSOA\$AllUsualResidents) * 10000. Er is een “antwoord” script in `script-examples/script-crime-rates.R` maar probeer hier niet meteen naar te kijken!

Praktijk 4: Neem je eigen data mee

Dit is de mogelijkheid om enkele technieken waar we het over hebben gehad toe te passen op jouw eigen data. Als je geen data bij jou hebt, kun je enkele voorbeelddata gebruiken uit de links die in de trainingshandout staan.

Referenties

Hier de referenties plaatsen

Deze ‘practical’ is geschreven met R 3.5.1 (2018-07-02) en RStudio 1.1.463 door Dr. Nick Bearman (nick@geospatialtrainingsolutions.co.uk).

Het werk is gelicenseerd onder Creative Commons Attribution-ShareAlike 4.0 International License. Om een kopie van deze licentie te zien, ga dan naar <http://creativecommons.org/licenses/by-sa/4.0/deed.en>. De laatste PDF-versie kun je hier <https://github.com/nickbearman/intro-r-spatial-analysis> vinden. Deze versie is op 18 May 2019 gemaakt.