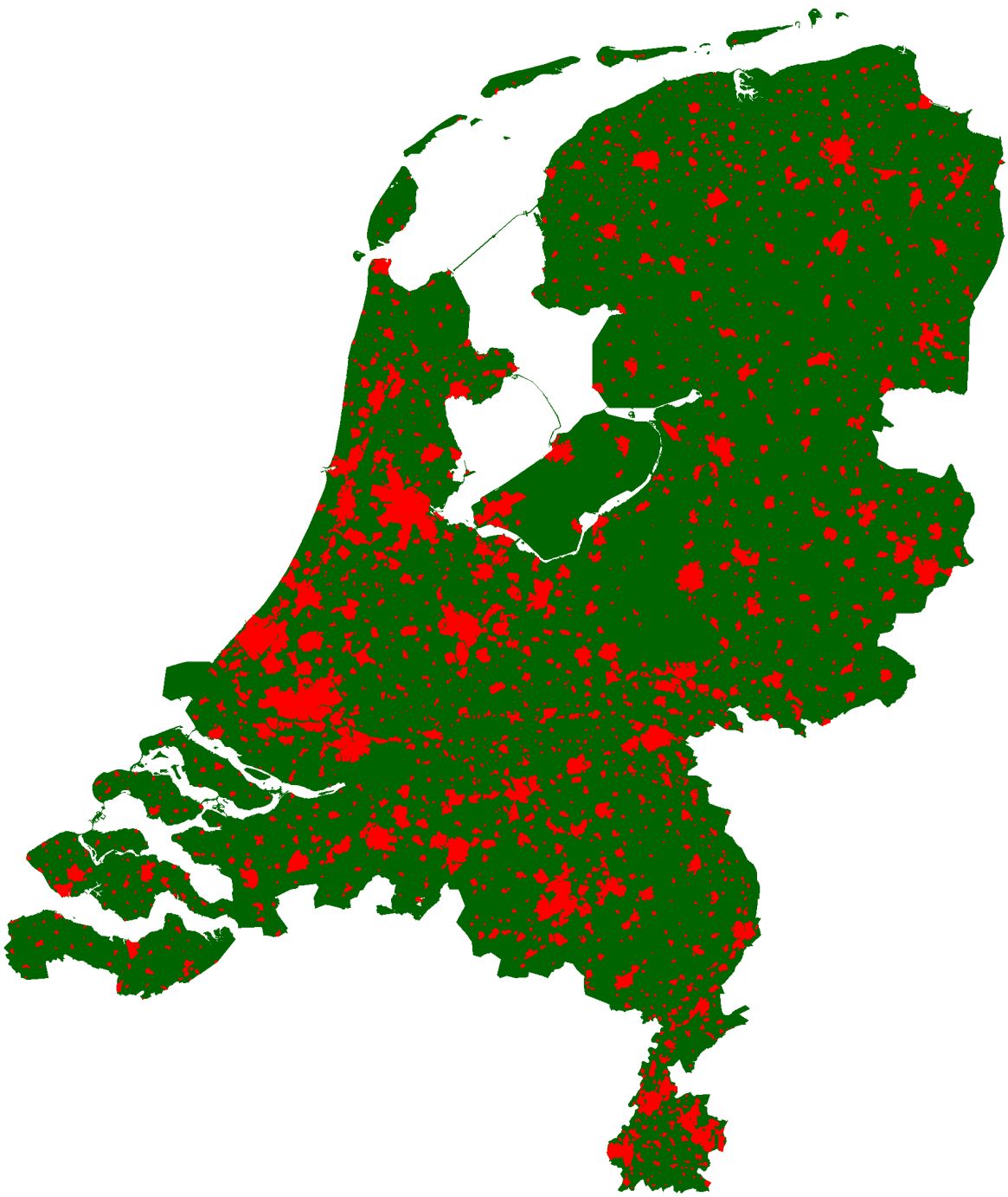
Geografische analyses met R

Versie 0.4.0 - ”Geobuzz”

Egge-Jan Pollé

23 november 2015



Dit is eigenlijk een **e-boek**...

Dit PDF document is zo opgemaakt dat het dubbelzijdig kan worden afgedrukt op A4 formaat.
Maar bedenk - voordat je gaat printen - eerst welke nadelen een gedrukte versie allemaal heeft:

- **Geen Ctrl+F:** op papier kun je veel minder makkelijk zoeken dan in een electronisch document. (Er is wel een inhoudsopgave, maar er staat geen index achter in het boek...)
- **Geen Ctrl+C/Ctrl+V:** vanaf papier kun je geen codevoorbeelden knippen en plakken; je zal alle code zelf in moeten typen. (Maar of dat nou echt een nadeel is...?)
- **Hyperlinks werken niet op papier:** dit document staat vol verwijzingen - zowel interne als externe - die in de electronische versie wel met één klik te volgen zijn.
- **Letters hebben een vaste grootte:** op papier kun je de lettergrootte niet aanpassen - en dat kan vooral voor de oudere lezertjes onder ons best een probleem zijn.

Kortom: het welgemeende advies is om dit boek te lezen met een PDF reader op je favoriete *device*. En op het moment dat je echt aan de slag gaat: gewwon op de computer waarop je ook **R** en **RStudio** hebt draaien.
(En als je dan toch gaat afdrukken, overweeg dan om dat wel in zwart-wit te doen.)

Dit is een uitgave van TWIAV - Onderzoek & Advies.

Voorwoord

Deze handleiding is speciaal samengesteld voor de Hands-on workshop **Geografische analyses met R** op de 4e jaarlijkse OSGeo.nl Dag, op de GeoBuzz, op woensdag 25 november 2015 in Den Bosch.

Het was in de zomer van 2015, in Como (Noord-Italië), in de wandelgangen van de FOSS4G-Europe 2015. Daar werd ik benaderd door Gert-Jan van der Weijden, de voorzitter van de stichting OSGeo.nl. Eerder die week had ik in Como de workshp **R for spatial data** van Robin Lovelace (University of Leeds) gevolgd. En nu kwam Gert-Jan met het verzoek of ik een soortgelijke workshop zou willen geven in Den Bosch. Daarop heb ik maar *ja* gezegd.

De afgelopen weken heb ik veel met **R** gespeeld. Een opdracht intypen op de commandoregel, en dan kijken wat er gebeurt. Want ja, om ergens een workshop over te kunnen geven, moet je er wel iets van weten. Dit document is het voorlopige resultaat van mijn speurtocht: laat ik maar meteen opschrijven wat ik geleerd heb, want anders vergeet ik het weer.

Dit is ook de plek om de Friese Woudloper, Willy Bakker te bedanken voor haar inbreng. Met haar technische adviezen heeft ze een grote bijdrage geleverd aan de totstandkoming van deze handleiding.

R heeft - als geografisch informatiesysteem - beslist voordelen ten opzichte van traditionele GIS desktop applicaties, met al die knopjes waarop je kunt klikken. Waar je in een gewoon GIS systeem de knoppenbalken en menu's op een gegeven moment wel zat bent ('*Kan dit niet sneller? kan ik dit niet automatiseren?*'), moet je in **R** al vanaf het allereerste commando goed nadenken over de syntaxis van je opdracht.

Ik hoop dan ook van harte dat mijn aantekeningen nuttig zijn voor anderen die kennis willen maken met de geografische kracht van **R**.

Overigens is dit een '*werk in uitvoering*'. Er zullen de komende tijd nog enkele nieuwe versies van deze cursushandleiding verschijnen. De meest recente versie staat altijdd hier.

Egge-Jan Pollé

Asperen - November 2015

Oproep: beoordelaars gezocht

Geen enkel goed boek kan zonder meelezers. Dus als je je geroepen voelt om (een deel van) het boek grondig te beoordelen en van commentaar voorzien - zowel tekstueel als codetechnisch - dan houdt de auteur zich van harte aanbevolen.
De beloning? Een vermelding in het voorwoord van een volgende editie van deze bundel.

Inhoudsopgave

Voor Sacha

1 Inleiding

1.1 De paden op, de lanen in, vooruit met flinke pas…

In dit boek gaan we op reis. Op reis door de werled van **R**, ja, je kunt wel zeggen de *wondere* wereld van **R**.

Ter voorbereiding op dit avontuur volgen hieronder enkele op- en aanmerkingen, handige tips en nuttige trucs.

1.2 Over dit boek

1.2.1 Voor wie?

Voor wie is dit boek bedoeld? Gewoon: voor iedereen die meer wil weten over het uitvoeren van geografische analyses met **R**. Is er specifieke voorkennis vereist? Nee (en ja), of: Ja (en nee)! Nou, toch meer 'nee' dan 'ja'. Dit document is geschreven om je te helpen iets te leren. Dus hoe minder voorkennis je hebt, hoe meer je hiervan kan leren.

R is een statistische programmeertaal, en wij gaan specifiek kijken naar geografische functionaliteit. Dus dan komen er veel terreinen bijelkaar: statistiek, informatica, GIS, geodesie, kartografie, etc., etc.. Een beetje voorkennis in één of meerdere van deze disciplines is nuttig, maar niet strikt noodzakelijk. Het belangrijkste is eigenlijk: een gezonde dosis nieuwsgierigheid, en creativiteit en doorzettingsvermogen.

Maar ik ben helemaal geen programmeur! Dat is niet erg, want dat geldt voor de meeste van ons. **R** is leuk omdat je snel mooie kaartjes en grafieken op het scherm kan toveren, en dat je dat zelf moet programmeren, dat ga je op den duur ook leuk vinden.

In elk hoofdstuk vind je codevoorbeelden met uitgebreide toelichting. Je moet deze voorbeelden niet alleen lezen, maar vooral gewoon doen: ga achter **R** zitten en probeer het zelf uit. Met de data die bij de oefeningen wordt meegeleverd of, beter nog, met je eigen geografische bestanden.

Dit cursusmateriaal leent zich voor zelfstudie, maar kan ook gebruikt worden in een klassikale omgeving, als onderdeel van een opleiding of workshop.

1.2.2 *Werk in uitvoering*

Deze bundel is een "work in progress". Dit is versie 0.4.0 - "Geobuzz". De komende tijd zal de het aantal hoofdstukken nog worden uitgebreid. Het advies is dan ook om, voor je verder leest, via deze link te controleren of je de meest recente versie hebt.

En o ja, op- en aanmerkingen zijn meer dan welkom. Heb je

- een alternatieve oplossing gevonden voor een probleem,
- suggesties voor aanvullende oefeningen,
- schrijf- en spelfouten ontdekt (???),
- of is er een *package* waarvan je vindt dat er beslist een hoofdstuk aan moet worden gewijd,

neem dan contact op met de auteur.

1.2.3 Broncode openbaar

Dit document is *open source*. De broncode is geschreven in een mengeling van R Markdown en pure LaTeX. Het PDF document is gecompileerd met **RStudio** (zie verderop in dit boek).

Het bronbestand (Geografische_analyses_met_R.Rmd) is te vinden in deze GitHub repository:

https://github.com/TWIAV/Geografische_analyses_met_R.

Dus als je wijzigingsvoorstellen direct wil doorvoeren in de bron: *Fork me on GitHub*.

1.3 Benodigdheden

1.3.1 Computer

Om deze cursus te volgen heb je een machine nodig. Ja, hè, hè. Maar wacht, dat hoeft niet per se je eigen computer te zijn, dat kan ook een virtuele machine zijn.

1.3.1.1 Virtuele machine van OSGeo-Live Wij raden je aan om eens te kijken naar de *Virtual Machine* van OSGeo-Live. Dit is een machine waarop een grote verscheidenheid aan *open source* (en dus gratis) geografische applicaties geïnstalleerd staat. Het is, zeg maar, de reclame-DVD van The Open Source Geospatial Foundation.

En ook het besturingssysteem van deze machine is *open source*, dat is namelijk Lubuntu, een Linux distributie. (Let op: de desktop omgeving van Lubuntu is redelijk gebruikersvriendelijk, dus ook een verstokte MS Windows gebruiker kan daarin de weg wel vinden.)

Je kunt de virtuele machine van **OSGeo-Live** gebruiken met behulp van VirtualBox, de virtualisatieoplossing van Oracle. Ook **VirtualBox** is gratis te downloaden.

Als deze optie aantrekkelijk klinkt, kijk dan hier voor installatieinstructies.

(**R** staat al geïnstalleerd op **OSGeo-Live**; **RStudio** moet je zelf even toevoegen.)

1.4 Cursusbestanden

De gegevensbestanden die je tijdens deze cursus nodig hebt worden aangeboden op de website www.twiav.nl of via andere, openbare bronnen, zoals de site van het Centraal Bureau voor de Statistiek.

Overigens is het downloaden van de benodigde data onderdeel van het script dat je bij elke oefening gaat schrijven. Dus je hoeft de bestanden niet zelf op te halen, dat gaat **R** voor je doen.

1.5 Meer informatie over het gebruik van R

Dit boek is geschreven voor een breed publiek, ook voor gebruikers die nog weinig of geen ervaring hebben met **R**. Maar dit is geen beginnershandleiding. De focus ligt op de geografische functionaliteit van **R**. Dit betekent dat - in de volgende hoofdstukken - sommige regels code uitvoerig worden toegelicht, terwijl andere commando's zonder verder commentaar of uitleg worden doorgegeven.

Het kan daardoor gebeuren dat je een regel code intikt, zonder dat je alle functies en parameters volledig begrijpt. Maar dat is niet erg, want om **R** te leren moet je kilometers maken, veel achter het systeem zitten, zaken uitproberen en vervolgens uitzoeken waarom iets wel (of juist niet) werkt.

Het wordt ten sterkste aangeraden om de **R** scripts in deze cursus regel voor regel uit te voeren (en niet meteen in één keer in zijn geheel). Probeer vooraf te bedenken wat er volgens jou gaat gebeuren, en controleer na elke regel of dat ook inderdaad zo is.

Door het hele boek heen vind je links naar externe bronnen voor aanvullende informatie (ja, de blauwe letters, ja). Hieronder wijzen we nog even op een aantal specifieke startpunten voor een zoektocht naar informatie over **R**.

1.5.1 *The R Manuals*

Een goed begin is de pagina met handleidingen die door leden van het *R Development Core Team* zelf wordt bijgehouden: https://cran.r-project.org/manuals.html. Hier vind je bijvoorbeeld An Introduction to R.

1.5.2 Discussiegroepen op internet

Tip: zoek - voordat je een vraag stelt in een discussiegroep - uit of deze niet al eens gesteld en beantwoord is. Die kans is namelijk groot. Als je met heel specifieke zoektermen googlet dan vind je het antwoord misschien wel in de archieven van één van de de onderstaande mailinglijsten:

- R Mailing Lists: er zijn verschillende discussiegroepen met betrekking tot **R** (zie overzicht: https://www.r-project.org/mail.html), waaronder een aantal **Special Interest Groups (SIG)**, zoals de R-sig-Geo – R Special Interest Group on using Geographical data and Mapping.
- Stack Overflow: een vraag- en antwoordsite voor programmeurs, die elkaar helpen, met bijna 5 miljoen deelnemers wereldwijd. Ook veel vragen, en nuttige antwoorden, over **R**.

2 Het Project R

2.1 Inleiding

R is een softwarepakket en een ontwikkelomgeving voor statistische berekeningen en het weergeven van grafieken. De titel van de homepage van de website van **R** (https://www.r-project.org/) luidt dan ook: **The R Project for Statistical Computing**.

R is een populaire toepassing voor onderzoekers en analisten op allerlei terreinen, en wordt vaak genoemd in relatie met termen als *Big Data* en *Data Science*.

In deze handleiding zullen we laten zien dat **R** ook heel goed overweg kan met geografische data, dat wil zeggen gegevens waar een geografisch object aan gekoppeld is.

2.1.1 Een korte geschiedenis van R

De ontwikkeling van **R** begint in de vroege jaren negentig van de vorige eeuw. De ontwikkelaars van het eerste uur zijn Ross Ihaka en Robert Gentleman. Zij ontmoeten elkaar voor het eerst in 1990 op het *Department of Statistics* van de *University of Auckland* (Nieuw-Zeeland). Zij halen een deel van hun inspiratie uit **S**, een programmeertaal die ontwikkeld is in de laboratoria van Bell. Bij wijze van grap noemen Ihaka en Gentleman hun taal **R**. Dat is immers niet alleen de letter die in het alfabet precies voor de S komt, maar tevens de eerste letter van hun beider voornamen.

Al kort na het uitbrengen van een vroege versie in 1994 sporen collega's van over zee hen aan om de broncode van deze nieuwe programmeertaal vrij te geven. Aangezien ze inschatten dat de vooruitzichten van **R** als commerciëel product slechts beperkt zijn, brengen ze hun geesteskind inderdaad uit als 'vrije software', onder de GPL licentie van de *Free Software Foundation*. Vanaf dat moment kan **R** gratis gedownload worden. Ook wordt er een discussielijst opgezet, waarop gebruikers over de verdere ontwikkeling van **R** mee kunnen praten.

En daarmee wordt een spectaculaire groei in gang gezet. Al snel kunnen Ihaka en Gentleman het niet meer samen aan en wordt het ontwikkelteam uitgebreid met een select gezelschap van programmeurs die direct toeganghebben tot de broncode - met schrijfrechten. In dit kernteam van ongeveer 20 man zitten een aantal bekende onderzoekers op het gebied van statistiek.

Op 29 februari 2000 besluit men dat de programmeertaal voldoende functionaliteit heeft, en stabiel genoeg is, om over te gaan tot het vrijgeven van versie R-1.0.0 (Ihaka, 2009).

2.2 R downloaden en installeren

Het eerste dat je moet doen om met **R** te kunnen werken is natuurlijk: het downloaden en installeren van de software.

R wordt niet op één centrale plaats aangeboden, maar is beschikbaar via een uitgebreid netwerk van zogeheten *mirror sites* onder de naam **CRAN**, het **Comprehensive R Archive Network**. Dit is een wereldwijd netwerk van servers - vaak van universiteiten - waarop identieke en actuele versies van de software staan opgeslagen.

Vanuit het **CRAN** wordt het vriendelijke verzoek gedaan om voor het downloaden een server in de buurt te gebruiken, om het netwerk niet te zwaar te belasten. Daarom hebben we gekozen voor de server van de Universiteit Utrecht: http://cran-mirror.cs.uu.nl/.

Op de downloadpagina moet je eerst een keuze maken voor **R** voor het juiste besturingssysteem. Voor het schrijven van deze handleiding maken we gebruik van een 64-bit Windows 7 machine, dus wij kiezen **R for Windows**. Vanzelfsprekend is **R** ook beschikbaar voor (Mac) OS X en voor Linux (vaak al opgenomen in de distributie).

Op de volgende pagina selecteren we de **base distribution**, want, zoals de website zegt: *dit is wat jij wilt als je **R** voor de eerste keer installeert*.

Op het moment van schrijven is versie 3.2.2 van 14 augustus 2015 de meest recente release van **R**. Maar de ontwikkelingen gaan snel, dus wellicht is er al een nieuwere versie beschikbaar als je dit leest. Zorg ervoor dat je de laatste versie downloadt.

Dubbelklik na het downloaden op het bestand **R-X.X.X-win.exe** om de installatie te starten. (Op dit moment is dat *R-3.2.2-win.exe*.)

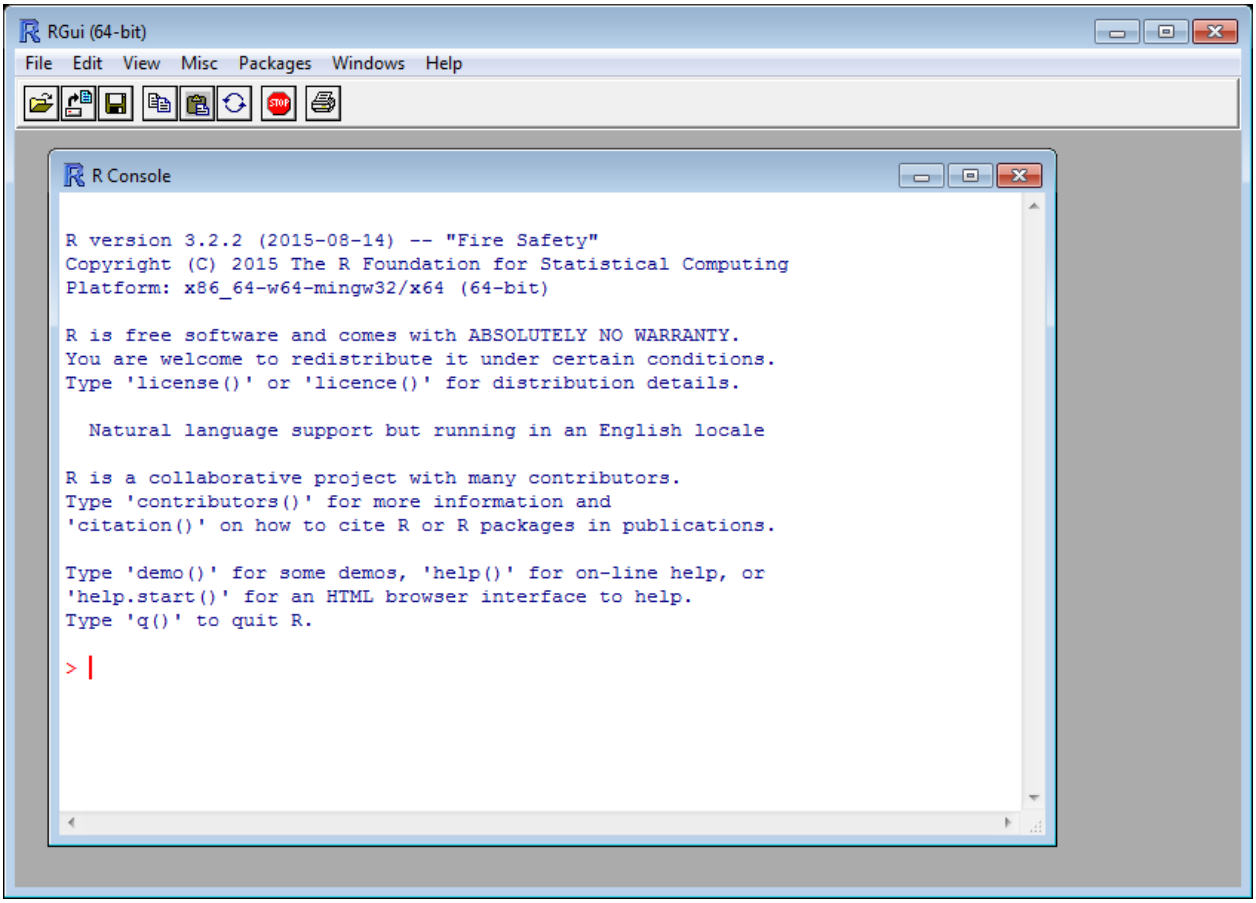
Codenaam

Elke nieuwe versie van **R** krijgt naast een nummer ook een codenaam. Zo heet versie 3.2.2 **Fire Safety**. Voorbeelden van enkele codenamen uit het verleden zijn: **Spring Dance** (3.0.3), **Security Blanket** (2.15.3), **Trick or Treat** (2.15.2) en **December Snowflakes** (2.14.1).
En ook de release die jij hebt geïnstalleerd heeft een bijzonder codenaam. Deze verschijnt in de console als je **R** opstart.

Tijdens het installatieproces kun je - afhankelijk van het besturingssysteem - kiezen of je de 32-bit of de 64-bit versie wilt installeren. Als je geen keuze maakt, dan worden beide versies geïnstalleerd.

Figuur 1: R op het bureaublad

Na het voltooien van de installatie kun je **R** opstarten. In de afbeelding hieronder zie je de **RGui** met daarin de **R Console**.



Figuur 2: De grafische gebruikersinterface van R

Voor nu kiezen we er voor om **R** direct te verlaten door *q()* te typen achter de prompt.

```
q()
```

2.3 RStudio downloaden en installeren

In de vorige paragraaf hebben we **R** direct na het opstarten weer afgesloten. En niet zonder reden. Om goed met **R** te kunnen werken gaan we eerst nog **RStudio** installeren.

RStudio is een **IDE**, een *integrated development environment* ofwel een geïntegreerde ontwikkelomgeving, die het werken met **R** op een aantal punten veel gemakkelijker maakt. Het is niet strikt noodzakelijk om **RStudio** te installeren. Je kan ook met **R** werken zonder deze toevoeging, maar zoals gezegd: het leven van een **R** programmeur wordt zoveel makkelijker met **RStudio**. Daarom zullen wij in deze handleiding gebruik maken van deze omgeving.

Productiviteit

RStudio bied een grote hoeveelheid functionaliteiten om de productiviteit van een **R** programmeur te verhogen. Twee belangrijke voorbeelden hiervan zijn:

- **Syntax Highlighting** (in het Nederlands *syntaxiskleuring* of *accentuering*): door het gebruik van verschillende kleuren tekst is de structuur van de code gemakkelijker te doorgronden, en zijn syntaxisfouten gemakkelijker op te sporen.
- **Code Completion**: RStudio biedt automatische aanvulling bij het schrijven van code, zowel voor namen van objecten die eerder gedefinieerd zijn, als voor functies en functieargumenten.

Na kennismaking met RStudio wil je nooit meer terug naar de RGui zoals we die in de vorige paragraaf hebben gezien.

Let op: **RStudio** is geen vervanging, maar draait bovenop **R**. Dus je moet er voor zorgen dat **R** eerst is geïnstalleerd voordat je de studio toevoegt.

Ga naar de website https://www.rstudio.com/ en klik op de button **Download RStudio**. We zijn op zoek naar de gratis desktop versie.

Op het moment van schrijven is versie 0.99.489 van 5 november 2015 de meest recente release van **RStudio**. Maar ook hier geldt dat er hoogstwaarschijnlijk al een nieuwere versie beschikbaar is als je dit leest.

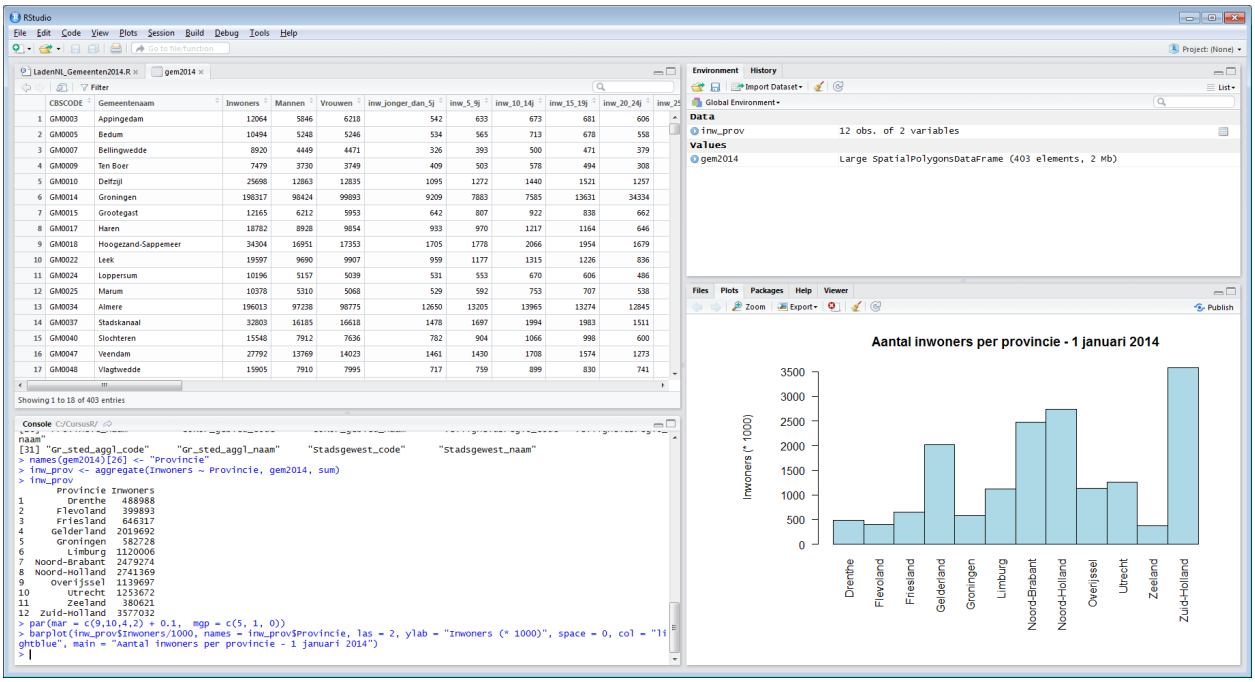
Dubbelklik na het downloaden op het bestand **RStudio-X.XX.XXX.exe** om de installatie te starten. (Op dit moment is dat *RStudio-0.99.489.exe*.)

Figuur 3: RStudio op het bureaublad

2.3.1 De RStudio gebruikersinterface

Na het voltooien van de installatie kun je **RStudio** opstarten. De interface van RStudio bestaat uit verschillende schermen (zie Figuur 4).

- **Linksonder:** hier zie je het *Console* venster. Dit is de commandoregel van **R**. Achter de prompt (**>**) kun je commando's ingeven die worden uitgevoerd door **R**, zodra je een <ENTER> hebt gegeven. En dit is ook de plek waar **R** waarschuwingen en/of foutmeldingen zal afdrukken.
- **Linksboven:** zodra je een nieuw **R** Script aanmaakt, of een bestaand script opent, dan wordt dat bestand hier in de linkerbovenhoek getoond. En ook het venster van de *Data Viewer* wordt hier getoond.
- **Rechtsboven:** op het tabblad **Environment** welke objecten en welke waarden **R** in het geheugen heeft. Ook kun je hier zien welke packages geladen zijn. Op het tabblad **History** kun je naar eerder ingevoerde commando's zoeken, en deze eventueel hergebruiken.
- **Rechtsonder:** hier in der rechteronderhoek vind je vijf verschillende tabbladen:
  - **Files:** hier kun je door het bestandssysteem bladeren
  - **Plots:** in dit venster worden je plots, dat wil zeggen: de grafieken en de kaartjes, getoond. (En RStudio houdt een geschiedenis bij van eerder gemaakte plots.)
  - **Packages:** hier kun je packages ('bibliotheken') installeren en of laden.
  - **Help:** Altijd nuttig :-)
  - **Viewer:** op dit tabblad wordt lokale webinhoud getoond (bijvoorbeeld als je - later in dit boek - met de bibliotheek `leaflet` aan de slag gaat).



Figuur 4: De grafische gebruikersinterface van RStudio

2.4 Packages, packages, packages... (ook wel: bibliotheken)

R is modulair opgebouwd. Er is het basissysteem, zoals we dat in de vorige paragrafen hebben geïnstalleerd. Maar daar bovenop zijn er vele aanvullende bibliotheken, zogeheten *packages*, beschikbaar.

Door middel van een *package* wordt extra functionaliteit toegevoegd aan **R**. Er zijn in de loop der jaren veel van dergelijke bibliotheken aan het **Comprehensive R Archive Network** toegevoegd, met bijdragen van wetenschappers en programmeurs van over de hele wereld. Ook vanuit Nederland.

Tijdens een presentatie in oktober 2009 kon Ross Ihaka nog 'opscheppen' dat er al meer dan 1.700 *packages* aan het **CRAN** waren toegevoegd (Ihaka, 2009), maar nu - op 22 november 2015, dus zes jaar later - staat de teller al op het aantal van 7.512 bijdragen (Zie: https://cran.r-project.org/web/packages/), en de teller loopt door.

Ook voor het uitvoeren van geografische analyses met **R** zijn vele bibliotheken beschikbaar. Voor een overzicht, zie: CRAN Task View: Analysis of Spatial Data.

In de volgende hoofdstukken zullen we telkens één of meer van deze bibliotheken introduceren. De rode draad in de rest van het boek is dan ook: *Packages, packages, packages...* (Overigens heeft deze handleiding niet de pretentie om alle 'geografische' bibliotheken te behandelen. Er is altijd meer dan men in één boek kan samenvatten.)

Als je op zoek bent naar bepaalde functionaliteit die niet beschikbaar is in basis-**R**, dan is de kans groot dat een aanvullende bibliotheek uitkomst biedt. En anders wordt je van harte uitgenodigd om zelf een *package* te schrijven en met de gemeenschap te delen.

2.4.1 Een bibliotheek installeren: `install.packages()`

Om een bibliotheek te kunnen gebruiken, moet je hem eerst - éénmalig - installeren. Dit kan heel eenvoudig vanuit **RStudio** zelf. Bijvoorbeeld:

```
install.packages("rgdal") # bibliotheeknaame met aanhalingstekens
```

Zoals je ziet wordt de benodigde software automatisch gedownload en geïnstalleerd.

2.4.2 Een bibliotheek laden: `library()`

Om een funcite uit een bepaalde bibliotheek te kunnen gebruiken, moet je deze bibliotheek eerst laden. En dit moet elke keer dat je **R** gebruikt opnieuw gebeuren. Bijvoorbeeld:

```
library(tmap) # bibliotheeknaame zonder aanhalingstekens
```

Als je een (fout)melding krijgt dat een bepaalde functie onbekend is, dan kan dat komen doordat je de benodigde bibliotheek niet geladen hebt.

In **R** scripts zie je vaak bovenaan één of meerdere `library()` statements staan, zodat de benodigde bibliotheken geladen worden voordat de rest van het script wordt uitgevoerd.

2.4.3 Zelf een bibliotheek bouwen: `devtools`

Het is mogelijk om zelf een **R** *package* te bouwen en dit via het **CRAN** met de weredwijde gemeenschap te delen. Misschien dat je nu denkt: 'Wat? Ik ben nog nauwelijks met **R** begonnen, en nu wil je al direct dat ik een bibliotheek ga bouwen?!' Nee, nee, nee: ik wil niks. Maar wellicht krijg je de smaak te pakken, en heb je op een dag wel de behoefte om een eigen *package* te creëren.

Overigens hebben we gezien dat er al heel veel *packages* zijn. Dus als je iets wil dat met **base R** niet lijkt te kunnen, kijk dan - voordaat je zelf iets bouwt - zeker even rond of er niet al een bibliotheek is die de gevraagde functionaliteit wel biedt.

Het *package* `devtools` biedt gereedschap om het bouwen van een **R** bibliotheek gemakkelijker te maken.

Verderop in deze handleiding zul je nog een andere reden vinden om `devtools` te installeren. Want naast de *packages* die al op het **CRAN** staan zijn er nog veel andere in ontwikkeling. En als je zo'n 'experimentele' bibliotheek wilt installeren, dan heb je daar vaak `devtools` voor nodig.

3 Aan de slag

3.1 Kennismaking met de ontwikkelomgeving

Voordat we in de volgende hoofdstukken met geografische data gaan werken volgt hier een eerste kennismaking met de IDE, *Integrated development environment* die RStudio is.

3.1.1 De werkmap (*Working Directory*): `getwd()` en `setwd()`

De *Working Directory* is de map waar **R** invoerbestanden zoekt en uitvoerbestanden wegschrijft.

Dus als je de opdracht geeft om een CSV bestand in te lezen met het commando `read.csv("test.csv")`, dan verwacht **R** dat het bestand in de werkmap staat. Met `getwd()` kun je opvragen wat de werkmap is:

```
getwd()
```

Dit is nu waarschijnlijk de standaard documentenmap.

Het wordt aangeraden om (per project) een speciale werkmap in te stellen. Dit doe je met `setwd()`. Bijvoorbeeld:

```
setwd("C:/CursusR") # Let op: gebruiek / of \\ (in plaats van \)
```

Tip: blader op de **Files** tab (in de rechteronderhoek van **RStudio**) naar de door jou ingestelde werkmap, zodat je in de gaten kunt houden welke bestanden **R** daar allemaal neerzet.

3.1.2 Oefening: een CSV bestand aanmaken met `write.table`

Voer het onderstaande script uit (regel voor regel!) en bekijk de inhoud van het uitvoerbestand `maanden.csv` dat in de werkmap staat.

```
nummer <- c(1:12)
en_lang <- month.name
en_kort <- month.abb
# months in your current locale
nl_lang <- format(ISOdate(2000, 1:12, 1), "%B")
nl_kort <- format(ISOdate(2000, 1:12, 1), "%b")
maanden <- data.frame(nummer, en_lang, nl_lang, en_kort, nl_kort)
write.table(maanden, file="maanden.csv", sep=",", row.names=FALSE)
```

3.2 Oefening: bestanden downloaden en uitpakken met R

Voor veel oefeningen in dit boek worden datasets gebruikt die als zip bestand op de website www.twiav.nl staan. Het is niet nodig om deze bestanden zelf handmatig te downloaden en uit te pakken. Met een paar regels code in het **R** script kun je dit automatisch laten doen.

Hieronder staat een voorbeeld van de 'downloadprocedure' die in veel scripts in deze cursus wordt gebruikt.

Doorloop deze procedure één of meerdere keren (en lees de commentaarregels, die beginnen met een #), en kijk na elke stap wat er gebeurd is. Gebruik ? voor uitleg over een functie. Bijvoorbeeld: `?unzip`, `?unlink` of `?rm`.

```
# Sla de URL van het te downloaden bestand op in een variabele
URL <- "http://www.twiav.nl/files/NL_Gemeenten2014.zip"
 # Extraheer de bestandsnaam uit de URL
bestand <- file.path(basename(URL))
 # Maak een submap in de werkmap (voor het opslaan van de invoerdata)
dir.create("./Data", showWarnings = FALSE)
 # Sla het huidige pad van de werkmap op in een variabele
werkmap <- getwd()
# Switch tijdelijk naar de Datamap
setwd("./Data")
# Download het bestand
download.file(URL, destfile = bestand, mode = "wb")
# Pak het bestand uit
unzip(bestand)
# Ruim een beetje op:
 # Na het uitpakken kan het zip bestand verwijderd worden
unlink(bestand)
# De variabelen 'URL' en 'bestand' zijn niet meer nodig na de download
rm(URL, bestand)
# Keer terug naar de oorspronkelijke werkmap
setwd(werkmap)
# De variabele 'werkmap' is niet meer nodig na terugkeer naar de oorspronkelijke werkmap
rm(werkmap)
```

Hoeveel bestanden staan er na deze actie in de map 'Data'?

4 Het werken met geografische gegevens in R

4.1 De bibliotheek sp

Het package sp is een belangrijk element voor het gebruik van geografische data in **R**. Deze bibliotheek biedt een aantal klassen en methoden voor het verwerken van deze gegegevens. Zo zijn er ruimtelijke datastructuren gedefinieerd voor het opslaan van punt-, lijn- en vlakobjecten, en voor rasterdata, al dan niet met daaraan gekoppelde attribuutgegevens.

De bouwstenen - dat wil zeggen de klassen en methoden - die deze bibliotheek aanlevert, worden door vele andere geografische **R** packages gebruikt, zowel direct als indirect. En dat was precies de bedoeling van de auteurs, Edzer Pebesma en Roger Bivand, toen zij rond 2003 met de ontwikkeling van deze package begonnen.

R is oorspronkelijk ontwikkeld als programmeertaal voor èn door statistici, maar omdat (bijna) elk gegeven ook een ruimtelijke component heeft (*"Waar?"*), was er al snel ook belangstelling voor deze taal vanuit geografische hoek. En door de modulaire opbouw van **R** was het mogelijk om GIS functionaliteit toe te voegen door middel van een zelf te bouwen package. Maar de ontwikkeling van deze geografische bibliotheken werd belemmerd doordat er geen gemeenschappelijk ruimtelijk raamwerk was; elke package had zijn eigen regels voor het opslaan en verwerken van geodata. Om aan deze wildgroei een einde te maken hebben Pebesma en Bivand de hadschoen opgepakt, en een aantal zaken éénduidig vastgelegd in hun bibliotheek. Hiermee hebben ze het mogelijk gemaakt dat de analyse van geografische gegevens met **R** op een veel meer coherente wijze kan worden uitgevoerd (Pebesma en Bivand, 2005).

De bocht in de Maas

Met **R** worden een aantaal oefendatasets meegeleverd, handig voor onderwijs- en demonstratiedoeleinden. Zo zul je vaak **R** oefeningen tegenkomen, waarin bijvoorbeeld de `mtcars` of de `iris` dataset wordt gebruikt.
En ook bij het sp package zit een dergelijk oefenbestand: de *Meuse river data set*. Deze set bevat gegevens over de locatie van concentraties van zware metalen (cadmium, koper, lood en zink) in de bodem van een overstromingsvlakte in een bocht van de Maas, ten westen van het Limburgse plaatsje Stein. Deze gegevens zijn in 1993 verzameld door Ruud van Rijn and Mathieu Rikken, in het kader van hun afstudeerproject Fysische Geografie aan de Universiteit Utrecht.
Deze data zijn door Edzer Pebesma - ook Fysisch Geograaf, in 1996 gepromoveerd aan dezelfde universiteit in Utrecht - toegevoegd aan 'zijn' sp bibliotheek.
Tegenwoordig is Pebesma hoogleraar aan de **Westfälische Wilhelms-Universität (WWU)** in Münster (Duitsland), waar hij directeur is van het **Institut für Geoinformatik**.

Zie voor voorbeelden waarin deze Maasdata worden gebruikt:
http://rspatial.r-forge.r-project.org/gallery/

Zie voor meer informatie over de bibliotheek sp de handleiding op het CRAN:

https://cran.r-project.org/web/packages/sp/sp.pdf.

4.2 Het aanmaken van een SpatialPointsDataFrame - 1

In deze paragraaf gaan we kennismaken met één van de hiervoor genoemde ruimtelijke datastructuren. We gaan zelf een SpatialPointsDataFrame aanmaken, op basis van gegevens in een data.frame.

Begin met het installeren van de betreffende bibliotheek en laad deze in het geheugen:

```
install.packages("sp") #éénmalig
library(sp)
```

Het invoerbestand is een lijstje met enkele bezienswaardigheden in Amsterdam, met bijbehorende x- en y-coördinaten. Dit bestand is online beschikbaar en staat op deze lokatie.

Je hoeft dit bestand niet te downloaden en op te slaan. Je kan het direct inlezen vanaf de huidige lokatie.

```
URL <- "http://www.twiav.nl/files/NL_Museums_Amsterdam.csv"
ams.df <- read.csv(URL)
ams.df
```

```
##                  Name                            Website        x        y
## 1         Rijksmuseum     https://www.rijksmuseum.nl/en 120797.3 485890.8
## 2     Van Gogh Museum    http://www.vangoghmuseum.nl/en 120520.9 485729.2
## 3    Stedelijk Museum        http://www.stedelijk.nl/en 120436.3 485678.2
## 4     Anne Frank Huis      http://www.annefrank.org/en/ 120741.5 487587.3
## 5     EYE Film Museum         https://www.eyefilm.nl/en 121887.9 488614.1
## 6 Heineken Experience http://www.heineken.com/Home.aspx 121258.7 485661.8
```

Bekijk van welke klasse ams.df is:

```
class(ams.df)
```

```
## [1] "data.frame"
```

De kolommen 3 en 4 bevatten de coordinaatinformatie (x en y). Deze kun je op de volgende wijze selecteren:

```
ams.df[3:4]
```

```
##          x        y
## 1 120797.3 485890.8
## 2 120520.9 485729.2
## 3 120436.3 485678.2
## 4 120741.5 487587.3
## 5 121887.9 488614.1
## 6 121258.7 485661.8
```

En de eerste twee Kolommen bevatten attribuutgegevens. Deze kun je desgewenst ook even bekijken: `ams.df[1:2]`

Met deze informatie kunnen we nu een geografisch object maken:

```
ams.spdf <- SpatialPointsDataFrame(coords = ams.df[3:4], data = ams.df[1:2])
ams.spdf
```

```
##            coordinates                Name
## 1 (120797.3, 485890.8)         Rijksmuseum
## 2 (120520.9, 485729.2)     Van Gogh Museum
## 3 (120436.3, 485678.2)    Stedelijk Museum
## 4 (120741.5, 487587.3)     Anne Frank Huis
## 5 (121887.9, 488614.1)     EYE Film Museum
## 6 (121258.7, 485661.8) Heineken Experience
##                              Website
## 1     https://www.rijksmuseum.nl/en
## 2    http://www.vangoghmuseum.nl/en
## 3        http://www.stedelijk.nl/en
## 4      http://www.annefrank.org/en/
## 5         https://www.eyefilm.nl/en
## 6 http://www.heineken.com/Home.aspx
```

Bekijk van welke klasse ams.spdf is:

```
class(ams.spdf)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

En vraag de samenvatting op:

```
summary(ams.spdf)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##        min      max
## x 120436.3 121887.9
## y 485661.8 488614.1
## Is projected: NA
## proj4string : [NA]
## Number of points: 6
## Data attributes:
##                   Name                                Website
##  Anne Frank Huis    :1   http://www.annefrank.org/en/     :1
##  EYE Film Museum    :1   http://www.heineken.com/Home.aspx:1
##  Heineken Experience:1   http://www.stedelijk.nl/en       :1
##  Rijksmuseum        :1   http://www.vangoghmuseum.nl/en   :1
##  Stedelijk Museum   :1   https://www.eyefilm.nl/en        :1
##  Van Gogh Museum    :1   https://www.rijksmuseum.nl/en    :1
```

Dat ziet er goed uit.

4.2.1 Plot

In de vorige paragraaf hebben we een SpatialPointsDataFrame aangemaakt, en nu gaan we dit plotten, om te zien hoe dat er uit ziet.

De snelste manier is:

```
plot(ams.spdf)
```

Dit geeft wel een erg saai kaartbeeld. Dat moet beter kunnen. Na wat experimenteren en uitzoeken zou je tot het volgende commando kunnen komen:

```
plot(ams.spdf, pch = 19, col = "blue", axes = TRUE,
     main = "Enkele bezienswaardigheden in Amsterdam")
```

Je wordt hier uitgenodigd om gebruik te gaan maken van het uitgebreide helpsysteem dat bij **R** hoort, om uit te zoeken wat de verschillende parameters precies doen.

Je kunt tekstlabels toevoegen aan de musea met onderstaande opdracht:

```
text(coordinates(ams.spdf), as.character(ams.spdf$Name),
     cex = .7, pos = 4, col = "blue")
```

Doordat alle tekstlabels rechts van het museum staan ontstaat op het Museumplein enige overlap.

Dit kunnen we oplossen door niet één generieke positie (`pos = 4`) voor alle labels op te geven, maar door een vector aan te maken waarin ieder museum een eigen labelpositie krijgt, zodat we kunnen variëren.

Maak een vector aan met (initieel) voor elk museum de waarde 4. En bekijk de inhoud van deze vector:

```
pos.vector <- rep(4, length(ams.spdf))
pos.vector
```

```
## [1] 4 4 4 4 4 4
```

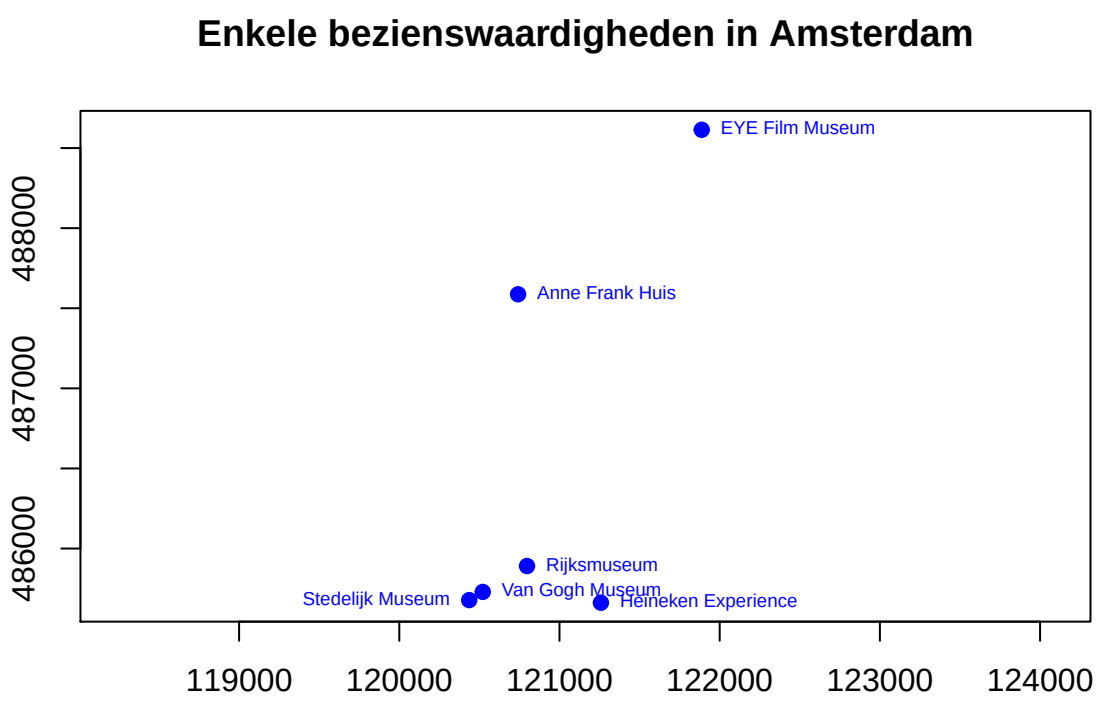
Ken nu een andere waarde toe voor het Stedelijk:

```
pos.vector[ams.spdf$Name == "Stedelijk Museum"] <- 2
pos.vector
```

```
## [1] 4 4 2 4 4 4
```

Plot opnieuw - nu met labels op goede plaats:

```
plot(ams.spdf, pch = 19, col = "blue", axes = TRUE,
     main = "Enkele bezienswaardigheden in Amsterdam")
text(coordinates(ams.spdf), as.character(ams.spdf$Name),
     cex = .6, pos = pos.vector, col = "blue")
```



4.3 Het toekennen van een CRS (*Coordinate Reference System*)

Het SpatialPointsDataFrame dat we in de vorige paragraaf (4.2) hebben aangemaakt, is nog niet helemaal compleet, omdat we er nog geen coördinaatreferentiesysteem (CRS) aan hebben toegekend. Het Frame heeft weliswaar coördinaten, maar we hebben nog niet vastgelegd hoe deze aan een plek op aarde te relateren zijn.

Er zijn twee soorten coördinaatsystemen:

- **ongeprojecteerd/geografisch**: in een dergelijk systeem wordt dor middel van lengte- en breedtegraden een positie op de aarde weergegeven,
- **geprojecteerd**: in een projectie wordt een gedeelte van het aardoppervlak weergegeven in een 2-dimensionaal vlak.

Een bekend, en tegenwoordig alom gebruikt systeem van het eerste soort is WGS84. Het officiële Nederlandse systeem - het Rijksdriehoeksstelsel (zie paragraaf 4.3.1) - is een voorbeeld van het tweede type.

Het toekennen van een CRS gaat door middel van een `proj4string`, een lange ingewikkelde string met geodetische parameters. In de samenvatting die we eerder hebben opgevraagd - met `summary(ams.spdf)` - zagen we al dat er aan onze data nog geen CRS was toegekend: `proj4string : [NA]`.

Gelukkig hebben veel projecties tegenwoorig een zogeheten *EPSG* code, waardoor het makkelijker wordt om een `proj4string` toe te kennen.

EPSG?

Het was de **European Petroleum Survey Group** die het initiatief heeft genomen de vele wereldwijd gebruikte coördinaatsystemen elk van een eigen, unieke code te voorzien, om de communicatie over locaties te vergemakkelijken. Het moge duidelijk zijn dat de (financiële) belangen van de olieindustrie hier groot zijn; men kan het zich simpelweg niet permitteren om op een verkeerde plek te gaan boren.
Hoewel de EPSG als organisatie al niet meer bestaat, wordt de EPSG Dataset nog steeds onderhouden door de International Association of Oil & Gas Producers (IOGP), tot voordeel van ons allen. Zie http://www.epsg.org/

Nu is natuurlijk de vraag: welk coördinaatsysteem gaan we aan onze data koppelen? Op die vraag is maar één antwoord mogelijk, namelijk: het juiste CRS. Je kunt immers niet zomaar een willekeurige referentie aan de gegevens hangen.

Dit betekent dat er altijd duidelijk moet zijn welk CRS bij geografische data hoort. Dit kan in de metadata staan, of het moet worden aangegeven door de dataleverancier.

De x en y van onze Amsterdamse museumdataset zijn gegeven in Nederlandse RD-coördinaten. De *EPSG* code van dit systeem is 28992.

Nu we dit weten kunnen we het CRS toekennen:

```
proj4string(ams.spdf) <- CRS("+init=epsg:28992")
summary(ams.spdf)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##        min      max
## x 120436.3 121887.9
## y 485661.8 488614.1
## Is projected: TRUE
## proj4string :
## [+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
## +lon_0=5.38763888888889 +k=0.9999079 +x_0=155000 +y_0=463000
## +ellps=bessel
## +towgs84=565.4171,50.3319,465.5524,-0.398957388243134,0.343987817378283,-1.87740163998045,4.0725
## +units=m +no_defs]
## Number of points: 6
## Data attributes:
##                   Name                                Website
##  Anne Frank Huis    :1   http://www.annefrank.org/en/     :1
##  EYE Film Museum    :1   http://www.heineken.com/Home.aspx:1
##  Heineken Experience:1   http://www.stedelijk.nl/en       :1
##  Rijksmuseum        :1   http://www.vangoghmuseum.nl/en   :1
##  Stedelijk Museum   :1   https://www.eyefilm.nl/en        :1
##  Van Gogh Museum    :1   https://www.rijksmuseum.nl/en    :1
```

4.3.1 Het Nederlandse coördinaatsysteem: het stelsel van de Rijksdriehoeksmeting

In Nederland worden de meeste geografische data uitgewisseld in RD-coördinaten. De EPSG code voor het RD-stelsel is 28992.

Het Rijksdriehoeksstelsel is het nationale geodetische triangulatiesysteem van Nederland. De basis van dit systeem is een stereografische kaartprojectie gecentreerd op de spits van de Onze Lieve Vrouwetoren in Amersfoort. Op het zo verkregen projectievlak is een cartesiaans assenstelsel geconstrueerd. De voor het RD-stelsel gebruikte stereografische projectie is gebaseerd op de ellipsoïde van Bessel (1841). Een belangrijk voordeel van de projectie is, dat deze *conform* (hoekgetrouw) is: een hoek gemeten in de kaart is gelijk aan de werkelijke hoek. Voor een klein gebied als Nederland kan men gebruik maken van een dergelijke projectie, al nemen de afstandsfouten wel toe, naarmate men verder van de oorsprong verwijderd raakt.

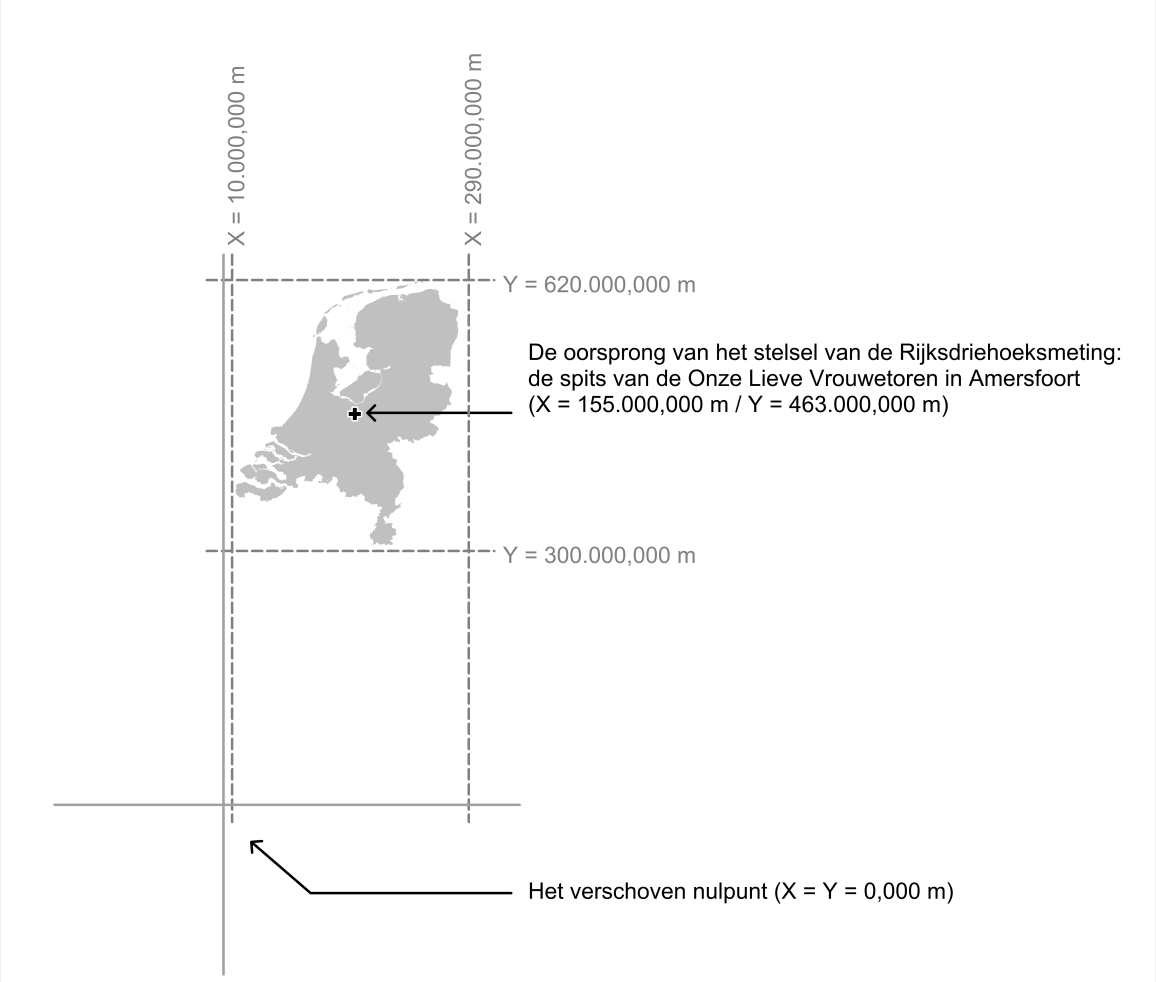
De oorsprong van het systeem ligt in Amersfoort. Aanvankelijk was deze oorsprong ook het nulpunt (X=Y=0) van het systeem. Dit betekende dat men in het noordoosten van Nederland met positieve X- en Y-coördinaten kon werken, maar dat men in de andere delen van het land rekening moest houden met een negatieve X- en/of een negatieve Y-coördinaat. Om dit probleem op te lossen heeft men in het begin van de jaren zeventig van de twintigste eeuw het nulpunt verschoven, en wel 155 kilometer naar het westen en 463 kilometer naar het zuiden. Nederland ligt nu in zijn geheel in het eerste kwadrant van het assenstelsel, en door de extreem grote Y-verschuiving is de grootst voorkomende X-waarde nog altijd kleiner dan de kleinst mogelijke Y-waarde. Daardoor zijn alle voorkomende coördinaten nu positief, en de X- en Y-waarden zijn goed uit elkaar te houden (zie figuur 5).

Het geldigheidsgebied voor de RD-coördinaten is nauwkeurig gedefinieerd. Globaal mogen deze coördinaten alleen gebruikt worden op het Nederlandse vasteland en in een kleine band daaromheen. Daar valt niet het gehele Nederlandse Continentaal Plat (het Nederlandse deel van de Noordzee) binnen. In België en Duitsland maakt men gebruik van andere coördinaatsystemen.

Voor meer achtergrondinformatie over de geodetische referentiestelsels van Nederland, zie: De Bruijne et al. (2005).

De belangrijkste kenmerken van het RD-stelsel zijn:

- stereografische projectie
- conform (hoekgetrouw)
- alleen bruikbaar voor het Nederlandse Grondgebied
- alle coördinaatgetallen zijn positief
- Y-coördinaat is altijd groter dan X-coördinaat
- meeteenheid: meters
- EPSG:28992



Figuur 5: De coördinaten van het Rijksdriehoeksstelsel

4.4 Het aanmaken van een SpatialPointsDataFrame -2

4.4.1 direct met CRS

Let op: het toekennen van het CRS hoeft niet in een aparte stap te gebeuren.Je kunt dit ook direct doen bij het aanmaken van het SpatialPointsDataFrame:

```
ams.spdf <- SpatialPointsDataFrame(coords = ams.df[3:4],
                           data = ams.df[1:2], proj4string = CRS("+init=epsg:28992"))
```

4.4.2 alternatieve syntaxis

In de voorgaande paragrafen hebben op de volgende manier een SpatialPointsDataFrame gecreëerd:

```
URL <- "http://www.twiav.nl/files/NL_Museums_Amsterdam.csv"
ams.df <- read.csv(URL)
ams.spdf <- SpatialPointsDataFrame(coords = ams.df[3:4],
                           data = ams.df[1:2], proj4string = CRS("+init=epsg:28992"))
```

In deze paragraaf gaan we het op een iets andere manier doen.

1. Het is niet noodzakelijk om twee verschillende objecten - in het voorbeeld `ams.df` en `ams.spdf` - te gebruiken. Je kunt ook een object zelf omzetten van data.frame naar SpatialPointsDataFrame,
2. We gaan een alternatieve syntaxis gebruiken voor het aanwijzen van de coördinaten.

```
URL <- "http://www.twiav.nl/files/NL_Museums_Amsterdam.csv"
mus <- read.csv(URL)
class(mus)
```

```
## [1] "data.frame"
```

```
coordinates(mus) <- ~x+y
class(mus)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
proj4string(mus) <- CRS("+init=epsg:28992")
```

Doordat het object `mus` op een andere manier is aangemaakt dan `ams.spdf`, is er 'diep onder water' een klein technisch veschil tussen de beide dataframes[1]. Maar voor praktische toepassingen zijn de objecten gelijkwaardig: het betreft in beide gevallen een geldig **SpatialPointsDataFrame**. Dit kun je controleren door de samenvatting op te vragen: `summary(ams.spdf)` en `summary(mus)`.

Welke manier beter is? Ach, dat is vaak een persoonlijke keuze van de programmeur, een kwestie van smaak. En, zoals de oude Romeinen pleegden te zeggen: *De gustibus non est disputandum*.

[1] Nieuwsgierig naar dit verschil? Ga op onderzoek uit met `all.equal(ams.spdf, mus)`

4.5 Een dataset herprojecteren met spTransform()

Het zal regelmatig voorkomen dat je een dataset met een bepaald CRS hetbt, terwijl voor de actie die je wilt uitvoeren een ander CRS noodzakelijk is. Gelukkig is het vrij eenvoudig om data te transformeren van het ene naar het andere CRS, op voorwaarde natuurlijk dat je de definitie van het doel-CRS kent.

Bijvoorbeeld: wij willen later in deze handleiding onze Amsterdamse musea op een OpenStreetMap kaart tonen (zie hoofdstuk 8). Hiervoor is het noodzakelijk dat de coördinaten worden getransformeerd naar WGS84. Gelukkig kennen wij de *EPSG* code van dit CRS, namelijk EPSG:4326.

In het voorbeeld hieronder maken we een nieuw object aan, waarbij de coördinaten uit `ams.spdf` worden overgezet naar WGS84. Bekijk de `coordinates` van `ams.spdf.wgs84`: dit zijn nu geen meters meer, maar decimale graden.

```
ams.spdf.wgs84 = spTransform(ams.spdf, CRS("+init=epsg:4326"))
summary(ams.spdf.wgs84)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##        min       max
## x  4.87983  4.900849
## y 52.35782 52.384387
## Is projected: FALSE
## proj4string :
## [+init=epsg:4326 +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84
## +towgs84=0,0,0]
## Number of points: 6
## Data attributes:
##                   Name                                Website
##  Anne Frank Huis    :1   http://www.annefrank.org/en/     :1
##  EYE Film Museum    :1   http://www.heineken.com/Home.aspx:1
##  Heineken Experience:1   http://www.stedelijk.nl/en       :1
##  Rijksmuseum        :1   http://www.vangoghmuseum.nl/en   :1
##  Stedelijk Museum   :1   https://www.eyefilm.nl/en        :1
##  Van Gogh Museum    :1   https://www.rijksmuseum.nl/en    :1
```

4.6 Handmatig een data.frame aanmaken

In de oefeningen hiervoor hebben we een data.frame verkregen door het inlezen van een bestand met `read.csv()`. In deze paragraaf maak je handmatig een data.frame aan met de functie `data.frame`.

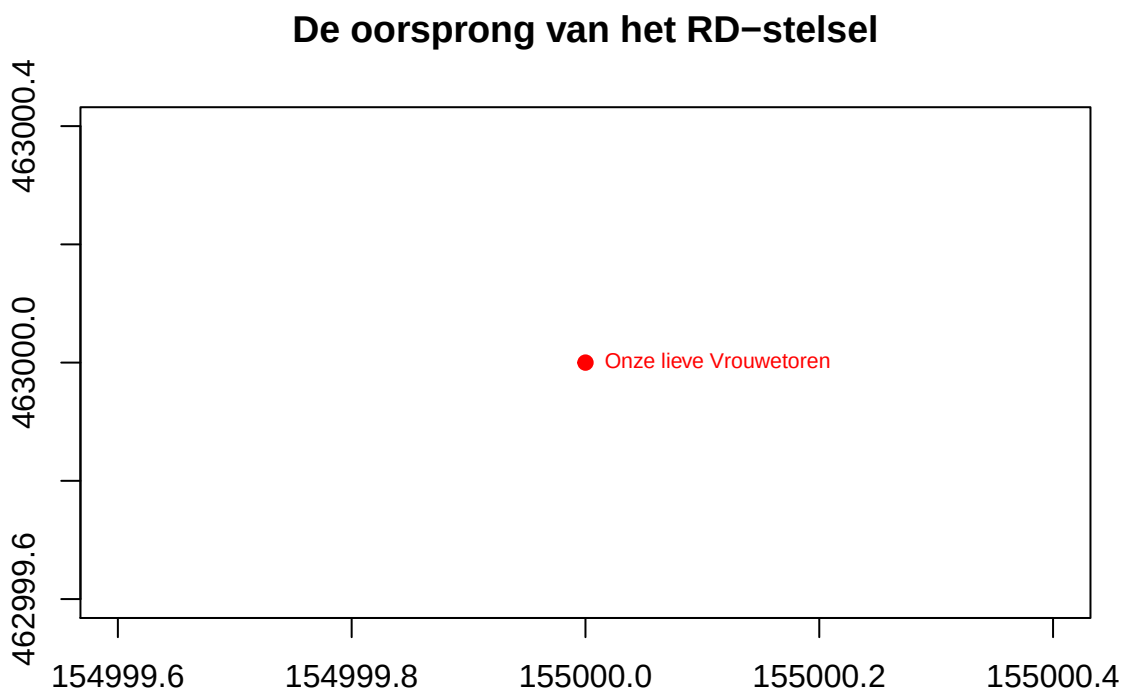
```
toren <- "Onze lieve Vrouwetoren"
X <- 155000
Y <- 463000
torens <- data.frame(toren, X, Y)
class(torens)
```

```
## [1] "data.frame"
```

```
coordinates(torens) <- ~X+Y
proj4string(torens) <- CRS("+init=epsg:28992")
summary(torens)
```

```
## Object of class SpatialPointsDataFrame
## Coordinates:
##      min    max
## X 155000 155000
## Y 463000 463000
## Is projected: TRUE
## proj4string :
## [+init=epsg:28992 +proj=sterea +lat_0=52.15616055555555
## +lon_0=5.38763888888889 +k=0.9999079 +x_0=155000 +y_0=463000
## +ellps=bessel
## +towgs84=565.4171,50.3319,465.5524,-0.398957388243134,0.343987817378283,-1.87740163998045,4.0725
## +units=m +no_defs]
## Number of points: 1
## Data attributes:
##                     toren
##  Onze lieve Vrouwetoren:1
```

```
plot(torens, pch = 19, col = "red", axes = TRUE, main = "De oorsprong van het RD-stelsel")
text(coordinates(torens), as.character(torens$toren), cex = .7, pos = 4, col = "red")
```



5 Het inlezen van geografische gegevens in R

5.1 De bibliotheek `rgdal`

In dit hoofdstuk gaan we gegevens vanuit externe GIS bestanden inlezen in **R**. Dit doen we met de functie `readOGR` uit het *package* `rgdal`.

De bibliotheek `rgdal` biedt toegang vanuit **R** tot de functionaliteit van **GDAL**. Voordat we deze installatie gaan uitvoeren, geven we eerst wat informatie over de kracht van **GDAL**.

5.1.1 GDAL

GDAL staat voor *Geospatial Data Abstraction Library*. Deze bibliotheek biedt abstracte gegevensmodellen voor geografische gegevens, één voor het raster- en één voor het vectorformaat. Daarnaast bevat **GDAL** een hoeveelheid programma's, zoals bijvoorbeeld `ogr2ogr`, `ogrinfo`, `gdaldem` en `gdal_contour`, voor het bewerken van geografische data. Deze programma's zijn zogeheten *command line utilities* (dat wil zeggen: ze moeten via de commandoregel worden aangesproken) en bieden vele opties voor onder andere:

- het inlezen en wegschrijven van geografische bestanden;
- het converteren van bestanden van het ene naar het andere geografische formaat (bijvoorbeeld van ESRI SHP naar GeoJSON);
- het herprojecteren van geografische bestanden (bijvoorbeeld van het Nederlandse RD-stelsel, EPSG:28992, naar WGS84, EPSG:4326);
- het analyseren en visualiseren van digitale hoogtemodellen en het genereren van contourlijnen;
- etc., etc.

GDAL wordt zowel gebruikt door Geo-ICT professionals, voor het - via de commandoregel - bewerken van geografische bestanden, als door programmeurs die op deze wijze geografische functionaliteit aan hun applicaties toe kunnen voegen.

De bibliotheek is oorspronkelijk ontwikkeld door Frank Warmerdam, maar tegenwoordig is de ontwikkeling en het onderhoud van **GDAL** ondergebracht bij de **Open Source Geospatial Foundation** (http://www.osgeo.org/).

Meer informatie over **GDAL** is te vinden op de website http://www.gdal.org/.

5.1.2 `install.packages("rgdal")`

`rgdal` maakt geen onderdeel uit van de basisinstallatie van **R**, dus we zullen dit *package* zelf moeten toevoegen. Typ achter de prompt het volgende commando:

```
install.packages("rgdal")
```

Je ziet nu in de console dat de gevraagde bibliotheek automatisch wordt gedownload en geïnstalleerd. Overigens maakt `rgdal` gebruik van ruimtelijke klassen die in de *package* `sp` zijn gedefiniëerd. Deze afhankelijkheid wordt automatisch gedetecteerd. Dus mocht je `sp` nog niet hebben geïnstalleerd, dan verschijnt in de console de melding: `also installing the dependency 'sp'`.

5.2 De gebruikte bestandsformaten: MapInfo TAB en ESRI SHP

We gaan in dit hoofdstuk aan de slag met de volgende populaire bestandsformaten: de MapInfo tabel (*table*) en de ESRI Shapefile, twee bekende GIS formaten voor het opslaan van vectorgegevens (punten, lijnen, vlakken) met bijbehorende attribuutgegevens.

Zowel het Mapinfo TAB als het ESRI SHP formaat bestaan al redelijk lang: ze zijn ontwikkeld in de pioniersjaren van de geografische informatietechnologie, de jaren negentig van de vorige eeuw. Toen was het opslaan van geometrie nog een kunst. Tegenwoordig kan elk zichzelf respecterend databasesysteem geometrie bevatten.

Ook in een ander opzicht zijn er overeenkomsten: we spreken wel over een ESRI Shapefile, maar feitelijk bestaat deze uit een aantal deelbestanden (minimaal een *.SHP, *.DBF en een *.SHX bestand, maar vaak nog meer). En een MapInfo TAB bestand bestaat minimaal uit een *.TAB, *.DAT, *.MAP en een *.ID bestand. In beide gevallen geldt dat alle deelbestanden dezelfde naam moeten hebben en in dezelfde map moeten staan.

Beide bestandstypen kunnen op precies dezlfde manier worden ingelezen met `readOGR()` in een spatial data frame.

Stel dat de tabel NL_Gemeenten2014.TAB in de submap `Data` in onze werkmap staat, en dat we deze in willen lezen in een object met de naam `gem2014`, dan is de syntax als volgt:

```
gem2014 <- readOGR(dsn = "Data", layer = "NL_Gemeenten2014")
```

De twee verplichte argumenten zijn:

- `dsn` (data source name): dit is de map (in het Engels *folder* of *directory*) waarin het bestand - met de bijbehorende deelbestanden - staat. Voor deze map kun je zowel een relatieve als een absolute padverwijzing gebruiken,
- `layer`: de naam van het bestand, zonder de extensie.

En voor een ESRI Shapefile is het precies hetzelfde…

Let op: het argument `dsn` is verplicht. Als het in te lezen bestand direct in de *working directory* staat (en dus niet in een submap), gebruik dan de volgende syntaxis: `dsn="."`

5.3 Het inlezen van geografische data: `readOGR()`

In deze oefening gebruiken we het bestand **NL_Gemeenten2014.TAB**. Achtergrondinformatie (metadata) over dit bestand staat in dit document: http://www.twiav.nl/files/NL_Gemeenten2014_Leesmij.pdf.

5.3.1 Stap 1: het klaarzetten van de invoerbestanden

Om het bestand te downloaden, uit te pakken en klaar te zetten in de submap 'Data' van onze werkmap gebruiken we de regels code hieronder. (Deze 'downloadprocedure' wordt toegelicht in paragraaf 3.2.)

```
URL <- "http://www.twiav.nl/files/NL_Gemeenten2014.zip"
bestand <- file.path(basename(URL))
dir.create("./Data", showWarnings = FALSE)
werkmap <- getwd()
setwd("./Data")
download.file(URL, destfile = bestand, mode = "wb")
unzip(bestand)
unlink(bestand)
rm(URL, bestand)
setwd(werkmap)
rm(werkmap)
```

Controleer nu of de Mapinfo tabel (bestaande uit meerdere deelbestanden) op de juiste plek staat.

5.3.2 Stap 2: het inlezen

De functie readOGR is onderdeel van de *package* rgdal, dus om de functie te kunnen gebruiken moet deze bibliotheek geladen zijn:

```
library(rgdal)
```

```
## rgdal: version: 1.1-1, (SVN revision 572)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 1.11.2, released 2015/02/10
##  Path to GDAL shared files: C:/Program Files/R/R-3.2.2/library/rgdal/gdal
##  GDAL does not use iconv for recoding strings.
##  Loaded PROJ.4 runtime: Rel. 4.9.1, 04 March 2015, [PJ_VERSION: 491]
##  Path to PROJ.4 shared files: C:/Program Files/R/R-3.2.2/library/rgdal/proj
##  Linking to sp version: 1.2-1
```

En dan is het inlezen zelf een fluitje van een cent:

```
gem2014 <- readOGR(dsn = "Data", layer = "NL_Gemeenten2014")
```

```
## OGR data source with driver: MapInfo File
## Source: "Data", layer: "NL_Gemeenten2014"
## with 403 features
## It has 34 fields
```

In de **Environment** tab (in de rechterbovenhoek van **RStudio**) zie je nu het object gem2014 verschijnen, als **Large SpatialPolygonsDataFrame (403 elements, 2 Mb)**

OK - de gegevens zijn geladen. In dit hoofdstuk gaan we nog even verder met het laden van andere datasets. Verderop in dit boek gaan we de attribuutgegevens van de dataset gem2014 inhoudelijk verkennen (zie hoofdstuk 6).

Toon de geometrie in de **Plots** tab:

```
plot(gem2014, col = "darkgreen", border = "lightgray")
```

Vraag de klasse op:

```
class(gem2014)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

Kijk ook naar de samenvatting (niet afgedrukt in dit boek, omdat deze bijna twee pagina's zou beslaan):

```
summary(gem2014)
```

We zien hier dat er al een CRS aan de dataset is toegekend. Ja, nogal wiedes, deze coördinaatreferentiegegevens zijn overgenomen van het invoerbestand.

Als je de `proj4string` opvraagt, dan zie je dat de dataset geprojecteerd is in RD-coördinaten (en waarom de string in dit boek van de pagina loopt, dat weet ik nog niet...):

```
proj4string(gem2014)
```

```
## [1] "+proj=stere +lat_0=52.156160556 +lon_0=5.387638889 +k=0.9999079 +x_0=155000 +y_0=463000 +ellps=bes
```

5.4 Meer gegevens inlezen: nog twee MapInfo tabellen

In deze paragraaf lezen we de tabellen **NL_Spoorwegen2015.TAB** en **NL_Stations2015.TAB** in, omdat we deze gaan gebruiken in allerlei oefeningen in de komende hoofdstukken. Achtergrondinformatie (metadata) over deze bestanden staat in dit document: http://www.twiav.nl/files/SpoorwegenEnStations_Leesmij.pdf.

5.4.1 Opnieuw stap 1: het klaarzetten van de invoerbestanden

De twee tabellen zitten samen in één zipbestand, dat we willen downloaden en uitpakken in de submap 'Data' van onze werkmap.

En daarvoor hoeven we alleen de URL te wijzigen in de eerder gebruikte 'downloadprocedure':

```
URL <- "http://www.twiav.nl/files/Spoorwegen2015.zip"
bestand <- file.path(basename(URL))
dir.create("./Data", showWarnings = FALSE)
werkmap <- getwd()
setwd("./Data")
download.file(URL, destfile = bestand, mode = "wb")
unzip(bestand)
unlink(bestand)
rm(URL, bestand)
setwd(werkmap)
rm(werkmap)
```

- *Hé, het valt op dat we telkens het commando `dir.create("./Data", showWarnings = FALSE)` meegeven. Waarom is dat?*
- *Nou, dat doen we om zeker te weten dat deze map bestaat. Als de map nog niet aanwezig is dan wordt-ie aangemaakt, en anders wordt de bestaande map gebruikt. Het is dus niet zo dat er telkens data verloren gaan omdat er opnieuw een lege map wordt aangemaakt… En o ja, we willen voorkomen dat **R** ons steeds lastigvalt met de meedeling dat de map al bestaat, vandaar `showWarnings = FALSE`*

Controleer of beide tabellen in de submap 'Data' staan.

5.4.2 Opnieuw stap 2: het inlezen

We gaan twee objecten aanmaken:

```
stations2015 <- readOGR(dsn = "Data", layer = "NL_Stations2015")
```

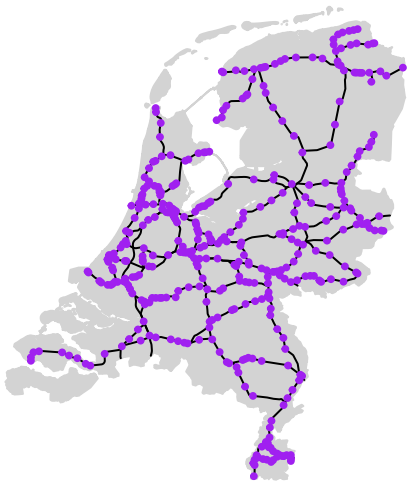
```
## OGR data source with driver: MapInfo File
## Source: "Data", layer: "NL_Stations2015"
## with 398 features
## It has 2 fields
```

```
spoorwegen2015 <- readOGR(dsn = "Data", layer = "NL_Spoorwegen2015")
```

```
## OGR data source with driver: MapInfo File
## Source: "Data", layer: "NL_Spoorwegen2015"
## with 115 features
## It has 2 fields
```

Toon de datasets in de **Plots** tab:

```
plot(gem2014, col = "lightgray", border = "lightgray")
plot(spoorwegen2015, add = TRUE)
plot(stations2015, pch = 19, cex = .4, col = "purple", add = TRUE)
```



En vraag de klasse op:

```
class(stations2015)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
class(spoorwegen2015)
```

```
## [1] "SpatialLinesDataFrame"
## attr(,"package")
## [1] "sp"
```

Bekijk - met behulp van `summary()` en/of `proj4string()` - welk CRS de datasets hebben.

5.5 En nog een keer gegevens inlezen: nu een ESRI Shapefile

In deze paragraaf lezen we het bestand **bevolkingskern_2011.shp** in. Dit is een ESRI Shapefile die door het Centraal Bureau voor de Statistiek (CBS) wordt aangeboden op deze pagina. Achtergrondinformatie (metadata) over deze bestanden staat in dit document: 2014BevolkingskerneninNederland2011pub.pdf.

5.5.1 En weer stap 1: het klaarzetten van de invoerbestanden

We moeten de eerder gebruikte 'downloadprocedure' hier op twee punten aanpassen:

- De ESRI en de Mapinfo bestanden mogen niet in dezelfde map staan (want dan geeft `readOGR()` foutmeldingen). We moeten dus een nieuwe submap aanmaken, die tevens als `dsn` zal dienen bij het inlezen. Wij hebben voor de mapnaam 'CBS' gekozen.
- De naam van het te downloaden bestand staat niet in de URL die naar dit bestand verwijst. Dus de eerder gebruikte syntaxis om de bestandsnaam te verkrijgen - `bestand <- file.path(basename(URL))` - zal hier niet werken. We moeten iets anders bedenken voor deze naam, bijvoorbeeld **tijdelijk.zip**. (Als je het bestand handmatig download, dan zul je ontdekken dat het de naam **2011-bevolkingskern-shape.zip** heeft. Maar wij gaan deze naam hier niet gebruiken. Omdat we het bestand toch direct na het uitpakken weer weggooien, kunnen we het net zo goed een tijdelijke naam geven.)

En dan kun je de shapefile als volgt downloaden en uitpakken:

```
URL <- "http://www.cbs.nl/nl-NL/menu/themas/dossiers/nederland-regionaal/links/
                                    2014-bevolkingskernen-in-nederland-2011-el.htm"
bestand <- "tijdelijk.zip"
dir.create("./CBS", showWarnings = FALSE)
werkmap <- getwd()
setwd("./CBS")
download.file(URL, destfile = bestand, mode = "wb")
unzip(bestand)
unlink(bestand)
rm(URL, bestand)
setwd(werkmap)
rm(werkmap)
```

Controleer of het bestand met de bevolkingskernen in de submap 'CBS' staat.

5.5.2 En ook nog een keer stap 2: het inlezen

We maken een object aan:

```
bevkern <- readOGR(dsn = "CBS", layer = "bevolkingskern_2011")
```

Toon de dataset in de **Plots** tab:

```
plot(gem2014, col = "darkgreen", border = "darkgreen")
plot(bevkern, col= "red", border = "red", add = TRUE)
# dit geeft de afbeelding zoals die op de voorpagina van deze bundel staat
```

En vraag de klasse op:

```
class(bevkern)
```

Bekijk - met behulp van `summary()` en/of `proj4string()` - welk CRS de dataset heeft.

6 Verkennende gegevensanalyse: *Exploratory Data Analysis* (EDA)

6.1 Bekijken

Om een eerste inzicht te krijgen in een verzameling gegevens, kun je er naar kijken.

Hiervoor biedt **RStudio** een goede mogelijkheid met de Data Viewer. Dit is een venster waarin je gegevens kan filteren, sorteren en doorzoeken.

Om de attribuutgegevens van onze eerder ingelezen dataset `gem2014` te bekijken, moet je het volgende commando geven:

```
View(gem2014)
```

(Let op: View met een hoofdletter V.)

Het venster van de **Data Viewer** verschijnt nu in de linkerbovenhoek van de grafische gebruikersinterface van RStudio (zie Figuur 4).

Experimenteer met de mogelijkheden van de **Data Viewer** door de volgende vragen te beantwoorden:

- Welke gemeente had op 1 januari 2014 de meeste inwoners? Hoeveel?
- En welke de minste?
- Hoeveel gemeenten hadden op die datum minder dan 5.000 inwoners? Welke gemeenten zijn dat?
- En hoeveel inwoners had de gemeente Súdwest-Fryslân op die dag?
- Hoeveel gemeenten liggen er in de Veiligheidsregio Haaglanden?

6.2 Selecteren

In een eerder hoofdstuk (zie hoofdstuk 5) hebben we een dataset met gemeenten ingelezen - `gem2014`. In deze paragraaf gaan we bekijken hoe we een gedeelte van deze data kunnen selecteren.

6.2.1 Selecteren met de functie `subset()`

6.2.1.1 Rijen selecteren Stel dat je alleen de gemeenten in de provincie Limburg wilt selecteren. Dat kan met de functie `subset()` als volgt (let op het dubbele isgelijkteken):

```
limburg <- subset(gem2014, Provincie_naam == "Limburg")
```

Bekijk het resultaat (niet in het boek afgedrukt):

```
plot(limburg, col = "darkgreen", border = "lightgray")
View(limburg)
```

Nu gaan we een selectie uitvoeren op een numerieke kolom. We willen namelijk een overzicht van alle grote gemeenten (hier gedefiniëerd als gemeenten met meer dan 200.000 inwoners):

```
grote_gemeenten <- subset(gem2014, Inwoners > 200000)
```

Bekijk het resultaat (niet in het boek afgedrukt):

```
plot(gem2014, col = "lightgray", border = "lightgray") # plot eerst heel nederland
plot(grote_gemeenten, col = "red", border = "red", add = TRUE) # voeg de grote gemeenten toe
View(grote_gemeenten)
```

En een selectie van middelgrote gemeenten (met niet minder dan 50.000 en niet meer dan 200.000 inwoners) ziet er dan zo uit:

```
middelgrote_gemeenten <- subset(gem2014, Inwoners >= 50000 & Inwoners <= 200000)
```

6.2.1.2 Kolommen selecteren De dataset met gemeenten heeft een groot aantal kolommen met attribuutgegevens, namelijk wel 34. Met de functie `subset()` kun je een de selectie van deze kolommen inperken.

Stel dat je alleen de gemeentenaam en de bevolkingsgegevens (de kolommen `Inwoners` tot en met `inw_80j_en_ouder`) wil hebben. Omdat dit een reeks met opeenvolgende kolommen is, kun die op de volgende twee manieren selecteren:

- met **kolomnummers**:

```
gem2014_bevolking_1 <- subset(gem2014, select = 2:14)
```

- met **kolomnamen**:

```
gem2014_bevolking_2 <- subset(gem2014, select = Gemeentenaam:inw_80j_en_ouder)
```

Bekijk het resultaat in de Data Viewer:

```
View(gem2014_bevolking_1)
```

en/of

```
View(gem2014_bevolking_2)
```

Er zijn nu nog maar 13 kolommen met attribuutgegevens.

Nota bene:

Als je op deze manier een aantal kolommen selecteert uit een `Spatial*DataFrame` object, dan wordt de geometrie meegeselecteerd. Dus het resultaat is ook een `Spatial*DataFrame` object.

Kijk maar:

```
class(gem2014_bevolking_1)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

Je kunt natuurlijk ook een selectie maken van niet-opeenvolgende kolommen (bijvoorbeeld naam, provincie en aantal inwoners). En ook dat kan op twee manieren:

- met **kolomnummers**:

```
gem2014_3kolommen_1 <- subset(gem2014, select = c(2,26,3))
```

- met **kolomnamen**:

```
gem2014_3kolommen_2 <- subset(gem2014, select = c(Gemeentenaam,Provincie_naam,Inwoners))
```

Bekijk het resultaat in de Data Viewer:

```
View(gem2014_3kolommen_1)
```

en/of

```
View(gem2014_3kolommen_2)
```

Er zijn nu nog maar 3 kolommen met attribuutgegevens.

6.2.1.3 Rijen èn Kolommen selecteren Natuurlijk kun je ook een selctie maken van rijen èn kolommen. In het voorbeeld hieronder selecteren we een aantal kolommen met betrekking tot de grote gemeenten:

```
grote_gemeenten_2 <- subset(gem2014, Inwoners > 200000,
                            select = c(Gemeentenaam,Provincie_naam,Inwoners))
View(grote_gemeenten_2)
```

6.2.2 Selecteren met indices (*bracket notation*)

Naast de hierboven gebruikte functie `subset()` bestaat er in **R** nog een manier om een selectie te maken, namelijk op basis van de indices van de dataset. De eerste index betreft de rijen en de tweede index betreft de kolommen. Bij deze selectiemethode wordt gebruik gemaakt van een notatie met vierkante haken (*square brackets*).

6.2.2.1 Rijen selecteren Stel dat je alleen de gemeenten in de provincie Groningen wilt selecteren. Dat kan met vierkenate haken op de volgende manier:

```
groningen <- gem2014[gem2014$Provincie_naam == "Groningen", ]
```

Let op de komma aan het eind! We geven in dit voorbeeld alleen een index op voor de rijen, en die voor de kolommen laten we leeg. Dat wil zeggen dat we alle attribuutgegevens (34 kolommen) willen hebben voor onze dataset `groningen`. (Als je de komma weglaat, dan krijg je een foutmelding.)

Bekijk het resultaat (niet in het boek afgedrukt):

```
plot(groningen, col = "darkgreen", border = "lightgray")
View(groningen)
```

Nog een aantal voorbeelden om zelf uit te proberen:

```
twee_provincies <- gem2014[gem2014$Provincie_naam %in% c("Drenthe","Friesland"), ]
```

```
kleine_gemeenten <- gem2014[gem2014$Inwoners >= 10000 & gem2014$Inwoners <= 200000, ]
```

```
superkleine_gemeenten <- gem2014[gem2014$Inwoners < 10000, ]
```

6.2.2.2 Kolommen selecteren

Stel dat je alleen de eerste vijf kolommen van de dataset wil selecteren:

```
gem2014_beperkt <- gem2014[ , 1:5]
View(gem2014_beperkt)
```

Let op de komma aan het begin. We geven in dit voorbeeld alleen een index op voor de kolommen, en die voor de rijen laten we leeg. Dat wil zeggen dat we alle rijen (403 records) willen hebben voor onze dataset `gem2014_beperkt`.

Overigens is de komma in dit geval - anders dan bij het selecteren van rijen - niet verplicht. Als je de komma weglaat, dan krijg je hetzelfde resultaat:

```
gem2014_beperkt2 <- gem2014[1:5]
View(gem2014_beperkt2)
```

Dit kun je als volgt testen:

```
all.equal(gem2014_beperkt,gem2014_beperkt2)
```

```
## [1] TRUE
```

Je kunt ook een aantal niet-opeenvolgende kolommen selecteren, naar keuze met kolomnummers of (in het voorbeeld hieronder) met kolomnamen:

```
gem2014_drie_kolommen <- gem2014[ , c("Gemeentenaam", "Provincie_naam", "Inwoners")]
View(gem2014_drie_kolommen)
```

6.2.2.3 Rijen èn Kolommen selecteren

Natuurlijk kun je op deze wijze ook een selctie maken van rijen èn kolommen. In het voorbeeld hieronder selecteren we een aantal kolommen met betrekking tot de wel hééééél erg kleine gemeenten:

```
super_super_kleine_gemeenten <- gem2014[gem2014$Inwoners < 2000, c(2,26,3)]
View(super_super_kleine_gemeenten)
```

6.3 Diagrammen

Gegevens hoeven niet *altijd* op een kaart gepresenteerd te worden. Soms kan een gewone staaf- of cirkeldiagram ook al heel informatief zijn.

6.3.1 Staafdiagram: `barplot()`

We gebruiken de functie `aggregate` om het aantal inwoners per provincie te berekenen:

```
inw_prov <- aggregate(Inwoners ~ Provincie_naam, gem2014, sum)
inw_prov
```

```
##    Provincie_naam Inwoners
## 1         Drenthe   488988
## 2       Flevoland   399893
## 3       Friesland   646317
## 4      Gelderland  2019692
## 5       Groningen   582728
## 6         Limburg  1120006
## 7   Noord-Brabant  2479274
## 8   Noord-Holland  2741369
## 9      Overijssel  1139697
## 10        Utrecht  1253672
## 11        Zeeland   380621
## 12   Zuid-Holland  3577032
```

Voordat we deze gegevens in een staafdigram gaan verwerken, gebruiken we de functie `par()` om een aantal grafische parameters aan te passen, zodat onze plot er netjes uit komt te zien:

```
par(mar = c(9,10,4,2) + 0.1,  mgp = c(5, 1, 0))
```

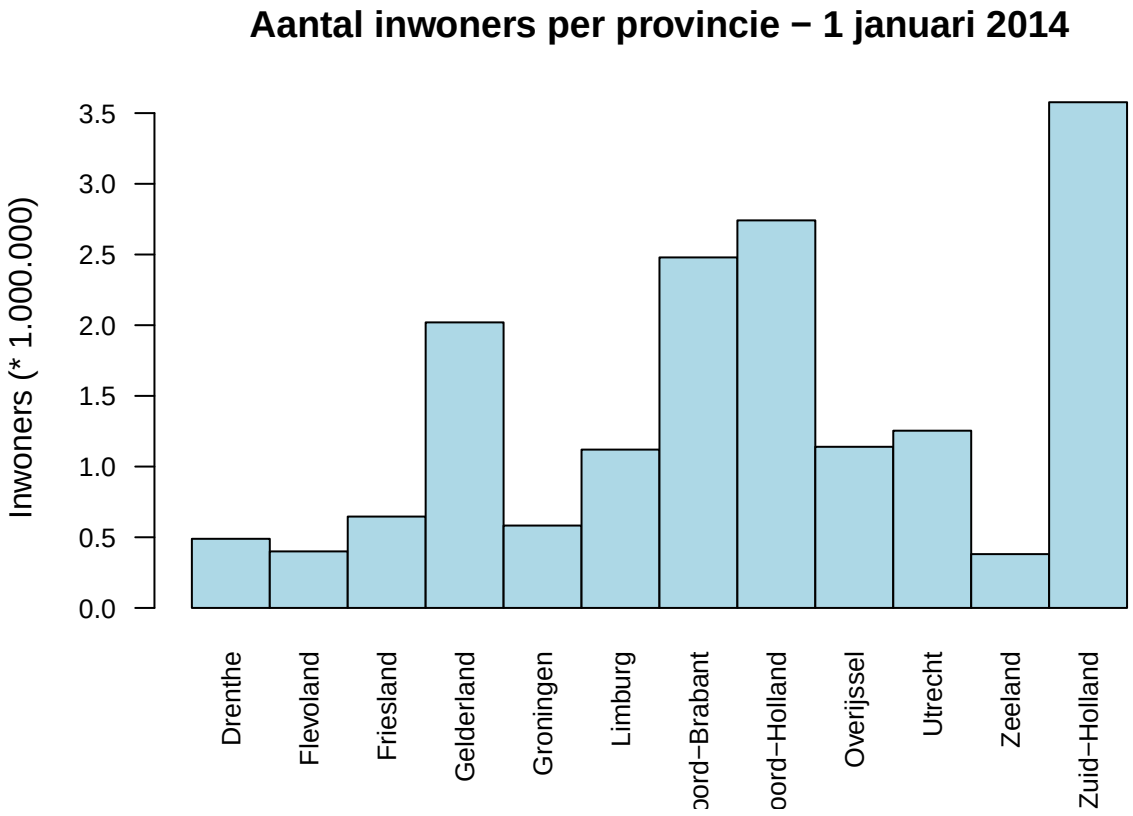
- `mar`: Een numerieke vector van de vorm `c(onder, links, boven, rechts)`, die het aantal lijnen van de marge geeft voor de vier zijden van de afdruk. De standaardinstelling is C (5, 4, 4, 2) + 0,1.
- `mgp`: De marge lijn (in mex eenheden) voor de astitel, aslabels en aslijn. De standaardinstelling is c (3, 1, 0).

Nu zijn we klaar om ons staafdiagram te produceren.

Gebruik de help om een idee te krijgen waar de verschillende parameters voor dienen. Je kunt natuurlijk ook gewoon een beetje expereimenteren met deze instellingen, om te kijken wat voor effect dat heeft.

Wij zijn tot het volgende barplot statement gekomen:

```
barplot(inw_prov$Inwoners/1000000, names = inw_prov$Provincie_naam, las = 2,
        cex.axis = .8, cex.names = .8, ylab = "Inwoners (* 1.000.000)", space = 0,
        col = "lightblue", main = "Aantal inwoners per provincie - 1 januari 2014")
```



6.3.2 Cirkeldiagram: `pie()`

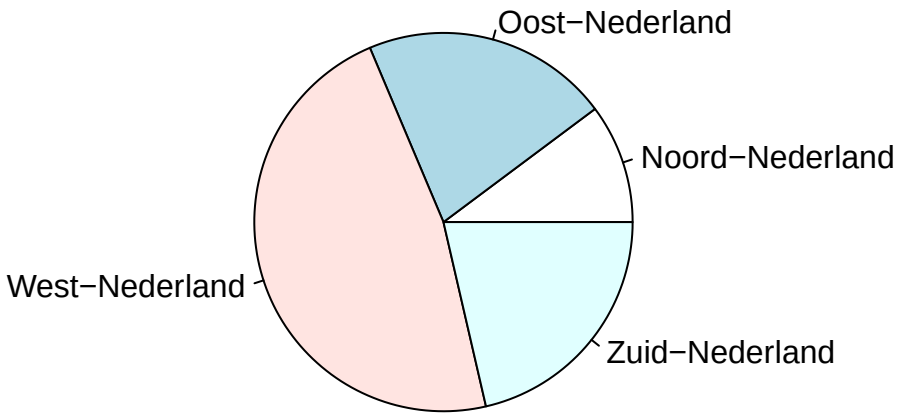
We gebruiken de functie `aggregate` om het aantal inwoners per landsdeel te berekenen:

```
inw_landsdeel <- aggregate(Inwoners ~ Landsdeel_naam, gem2014, sum)
inw_landsdeel
```

```
##    Landsdeel_naam Inwoners
## 1 Noord-Nederland  1718033
## 2  Oost-Nederland  3559282
## 3  West-Nederland  7952694
## 4  Zuid-Nederland  3599280
```

Als je snel even een indruk wil krijgen van de verhouding tussen de verschillende landsdelen, dan kun je het volgende simpele statement gebruiken:

```
pie(inw_landsdeel$Inwoners, labels = inw_landsdeel$Landsdeel_naam)
```



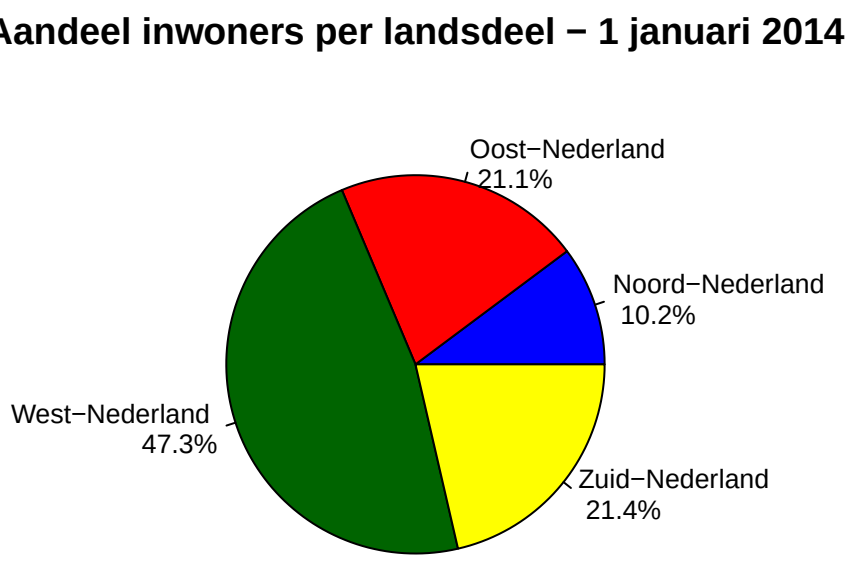
Dit resultaat is misschien niet visueel aantrekkelijk, maar helder en duidelijk genoeg voor een eerste verkennende dataanalyse.

In de onderstaande regels code berekenen we eerst een percentage dat we in de labels verwerken.

Vraag: Wat is het verschil tussen de functies `paste()` en `paste0()`?

Hiermee plotten wij een meer publicabel cirkeldiagram:

```
pct <- round(inw_landsdeel$Inwoners/sum(inw_landsdeel$Inwoners)*100, 1)
pct <- paste0(pct, "%")
lbls <- paste(inw_landsdeel$Landsdeel_naam, "\n", pct)
pie(inw_landsdeel$Inwoners, labels = lbls,
    cex = .8, col = c("blue", "red", "darkgreen", "yellow"),
    main = "Aandeel inwoners per landsdeel - 1 januari 2014")
```



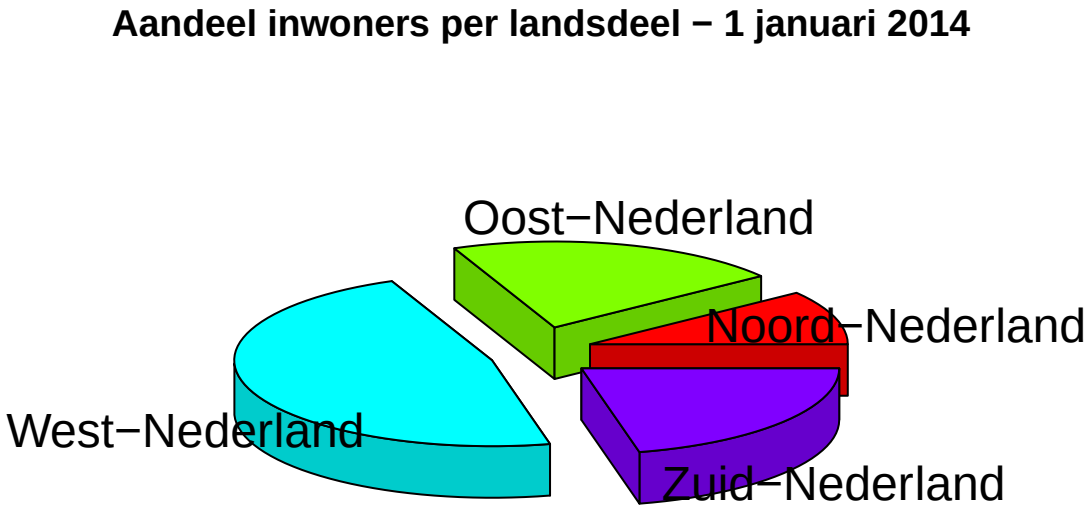
Uitdaging: Pas de labels zo aan, dat ze niet alleen het percentage, maar ook het absolute aantal inwoners per landsdeel weergeven. En zorg er daarbij voor dat deze inwoneraantallen netjes geformatteerd zijn, met een '.' tussen de duizendtallen. (**Tip:** Gebruik hiervoor de functie `format()`, met `big.mark = "."`.)

6.3.3 Driedimensionaal cirkeldiagram: `pie3D()`

Tijdens het googlen kwamen wij nog het *package* 'plotrix tegen.

Daarmee kun je leuk dit 'driedimensionale' cirkeldiagram maken:

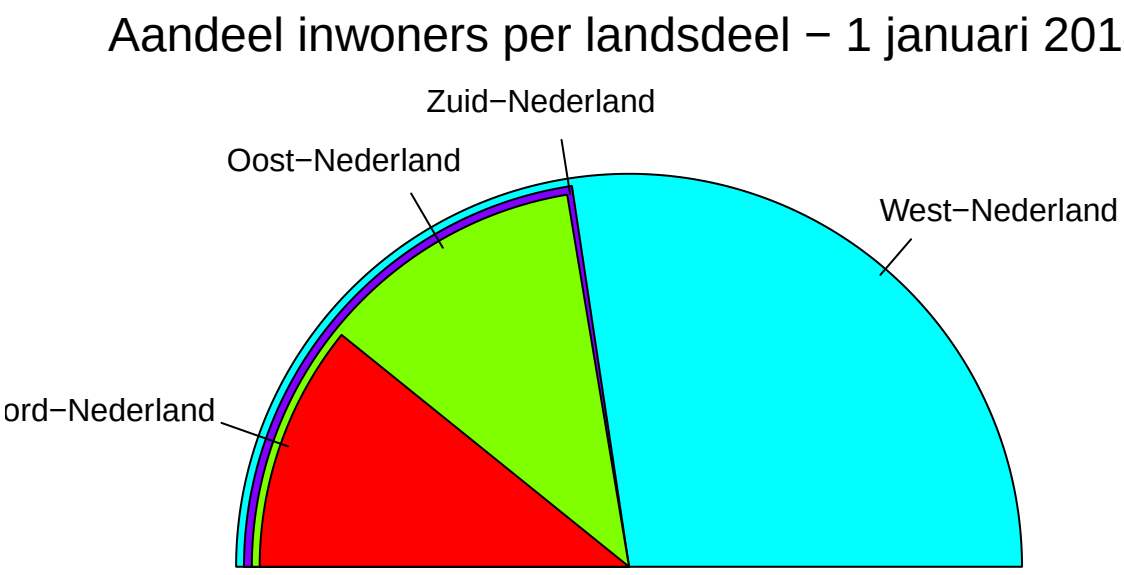
```
library(plotrix)
pie3D(inw_landsdeel$Inwoners, labels = inw_landsdeel$Landsdeel_naam,
      cex = .8, explode = .2, main = "Aandeel inwoners per landsdeel - 1 januari 2014")
```



6.3.4 Waaierdiagram: `fan.plot()`

Ook met behulp van `plotrix`:

```
fan.plot(inw_landsdeel$Inwoners,labels=as.character(inw_landsdeel$Landsdeel_naam),
         align="left", max.span=pi,
         main = "Aandeel inwoners per landsdeel - 1 januari 2014")
```



7 Thematisch presenteren

7.1 De bibliotheek `tmap`

Als je graag mooie thematische kaartjes wilt gaan maken met **R**, dan moet je beslist eens kijken naar de nieuwe *package* `tmap`. Deze bibliotheek is ontwikkeld door Martijn Tennekes, werkzaam bij het Nederlandse Centraal Bureau voor de Statistiek.

De ontwikkeling van deze package is begonnen in 2013 - de *Initial commit* in de GitHub repository dateert van 25 november van dat jaar - en op 28 mei 2015 is versie 1.0 van `tmap` vrijgegeven.

In de zomer van 2015 heeft Tennekes een presentatie gegeven, over `tmap`, op de useR! Conference 2015 in Aalborg, Denemarken.

In de volgende paragrafen gaan we theamtische kaarten maken op basis van de gemeenten (`gem2014`) en (`spoorwegen2015`) die we in een eerder hoofdstuk hebben ingelezen (zie hoofdstuk 5).

Zie voor meer informatie over de bibliotheek `tmap` de handleiding op het CRAN:

https://cran.r-project.org/web/packages/tmap/tmap.pdf

Er is ook een vignette *tmap in a nutshell*:

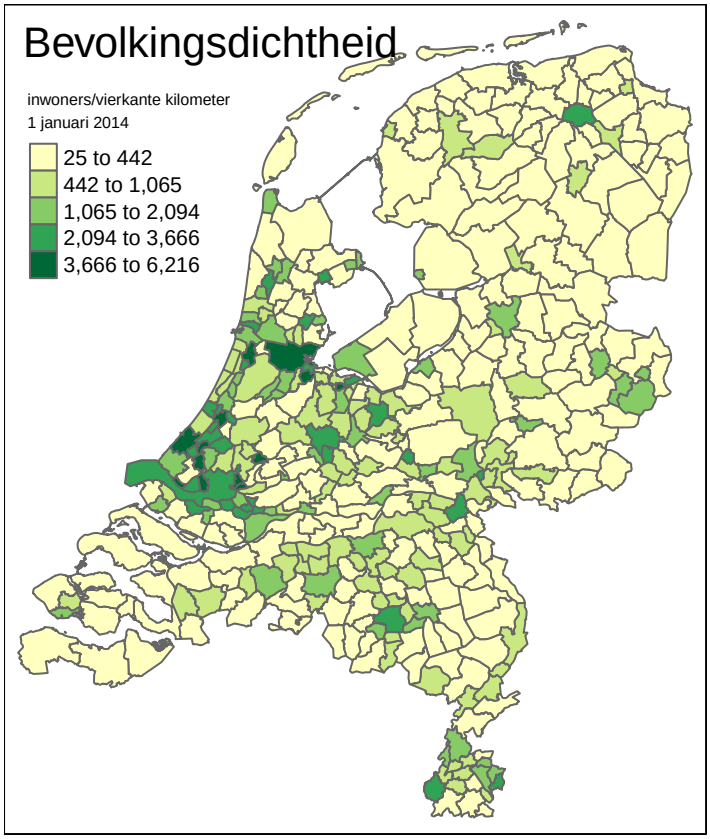
https://cran.r-project.org/web/packages/tmap/vignettes/tmap-nutshell.html

> Een 'concurrerende' *package*? `cartography`
>
> Eind oktober 2015 verscheen er een berichtje in de mailinglijst **R-sig-Geo**, waarin aandacht werd gevraagd voor een nieuwe bibliotheek: `cartography`.
> Deze is ontwikkeld door Timothée Giraud en Nicolas Lambert van de Groupe ElementR. Let op de Franse uitspraak: **ElementR** - *R pour les sciences humaines et sociales*.
> Wij hebben nog niet de tijd en de moeite genomen om deze bibliotheek grondig te bekijken, maar gaan dat in de toekomst zeker doen.
>
> - **Vingette** met voorbeelden: https://cran.r-project.org/web/packages/cartography/vignettes/cartography.html
> - **Handleiding op het CRAN:** https://cran.r-project.org/web/packages/cartography/cartography.pdf
> - **Broncode op GitHub:** https://github.com/Groupe-ElementR/cartography/

Nu snel verder met `tmap`, waarover dit hoofdstuk eigenlijk ging.

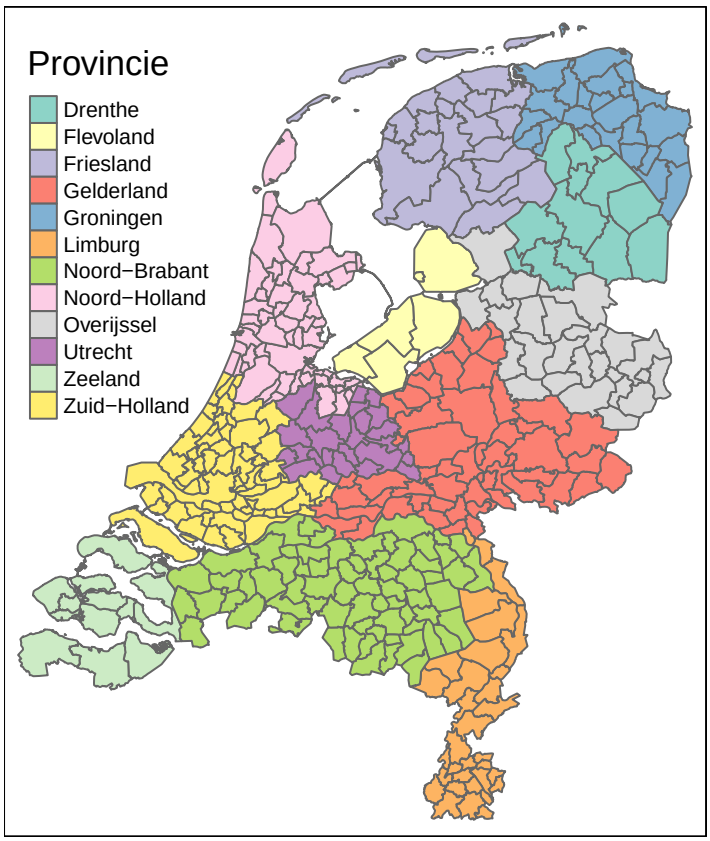
7.2 Choropleet

```
library(tmap)
 qtm(shp = gem2014, fill = "Bevolkingsdichtheid",
     fill.style = "kmeans", title = "Bevolkingsdichtheid",
     fill.title = "inwoners/vierkante kilometer\n1 januari 2014")
```



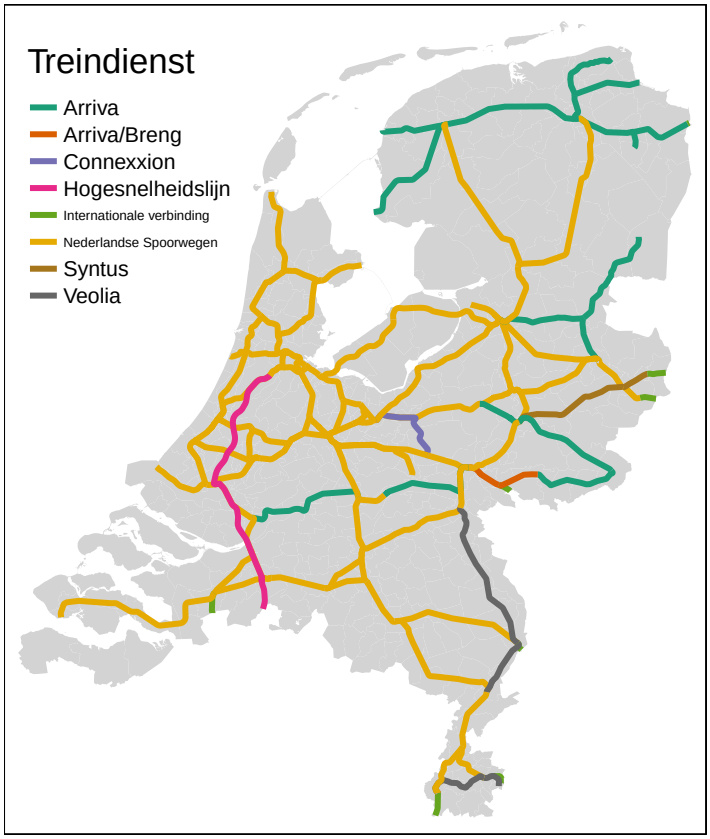
7.3 Inkleuring naar individuele waarden (vlakken)

```
library(tmap)
 qtm(shp = gem2014, fill = "Provincie_naam", fill.title = "Provincie")
```



7.4 Inkleuring naar individuele waarden (lijnen)

```
tm_shape(gem2014) +
  tm_fill("lightgray") +
  tm_shape(spoorwegen2015) +
  tm_lines(col = "Treindienst", lwd = 3)
```



8 Interactieve kaartvensters in R

8.1 De bibliotheek `leaflet`

Leaflet (http://leafletjs.com/) is een open-source JavaScript bibliotheek voor het maken van interactieve kaarten. De functionaliteit van Leaflet wordt in **R** beschikbaar gemaakt via het package `leaflet`.

Deze bibliotheek wordt onderhouden door Joe Cheng van RStudio, Inc.

In de oefeningen hieronder zullen we eerst stap voor stap een Leafletkaart maken. Daarna zullen we de *pipe operator* (%>%) introduceren, waarmee de code vereenvoudigd kan worden.

Zie voor meer informatie over de bibliotheek `leaflet`

- de handleiding op het CRAN: https://cran.r-project.org/web/packages/leaflet/leaflet.pdf.

8.2 Een interactieve kaart - 1: stap voor stap

Ter voorbereiding laden we onze Amsterdamse musea weer (zie hoofdstuk 4):

```
library(sp)
URL <- "http://www.twiav.nl/files/NL_Museums_Amsterdam.csv"
mus <- read.csv(URL)
coordinates(mus) <- ~x+y
proj4string(mus) <- CRS("+init=epsg:28992")
```

Stap 1: het aanmaken van een Leaflet map widget:

```
ikaart <- leaflet()
ikaart
```

Dit geeft een leeg kaartvenster in de **Viewer** tab.

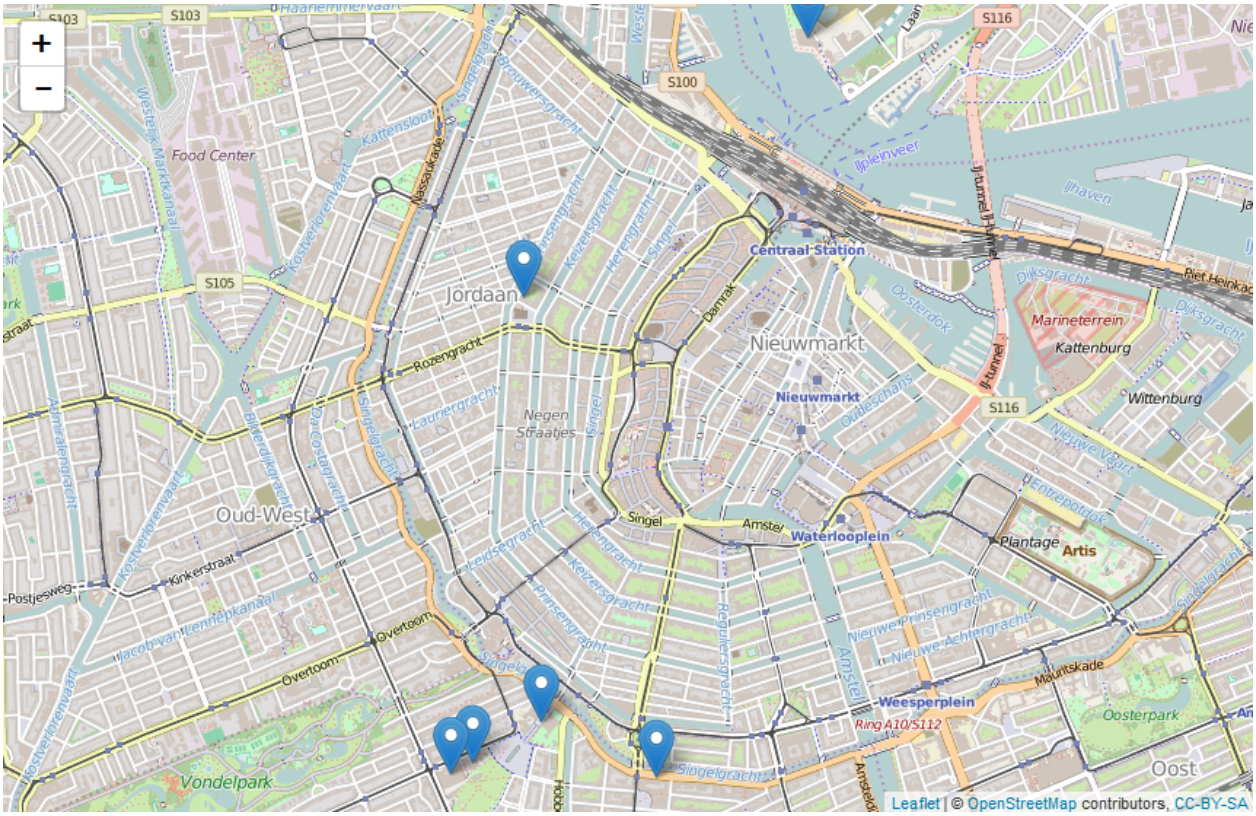
Stap 2: het toevoegen van de OpenStreetMap kaartlaag:

```
ikaart <- addTiles(ikaart)
ikaart
```

Stap 3: het toevoegen van onze eigen laag (let op: de gegevens worden *on the fly* geconverteerd van epsg:28992 naar epsg:4326):

```
ikaart <- addMarkers(ikaart,
                     data = spTransform(mus, CRS("+init=epsg:4326")), popup = mus$Name)
ikaart
```

Et voilà! De afbeelding in deze bundel is zo statisch als wat. Maar de kaart in de **Viewer** tab is echt interactief: je kunt inzoomen, uitzoomen en pannen. Vergeet ook niet op de markers te klikken om de tooltip te controleren.



8.3 Een interactieve kaart - 2: met de *pipe operator* (%>%)

In de vorige paragraaf hebben we in 3 stappen een kaart - `ikaart` - gemaakt:

```
ikaart <- leaflet()
ikaart <- addTiles(ikaart)
ikaart <- addMarkers(ikaart,
                     data = spTransform(mus, CRS("+init=epsg:4326")), popup = mus$Name)
```

Hierbij diende telkens de uitvoer van een functie als invoer voor de volgende stap: eerst wordt `ikaart` gevuld met een lege htmlwidget, vervolgens dient `ikaart` als invoer voor de functie `addTiles()` om de OSM laag toe te voegen, en dan wordt `ikaart` nog een keer ingevoerd in de functie `addMarkers`...

Deze 3 stappen kunnen worden teruggebracht tot één met behulp van de *pipe operator*, het symbool: %>%

Met deze operator kun je de uitvoer van een functie doorgeven aan de volgende, zodat je functies als het ware aan elkaar kunt reigen. En dat ziet er dan zo uit:

```
ikaart2 <- leaflet() %>%
  addTiles() %>%
  addMarkers(data = spTransform(mus, CRS("+init=epsg:4326")), popup = mus$Name)
ikaart2
```

Dit geeft dezelfde kaart als net.

Daarmee leidt het gebruik van de deze operator tot code die èn beter leesbaar is, en makkelijker te onderhouden.

De pipe operator is een relatief nieuw fenomeen in **R**; hij bestaat pas sinds 2014 en is ontwikkeld door Stefan Milton in zijn `magrittr` package. (Nee, inderdaad: je hebt dit `magrittr` package niet zelf geïnstalleerd - maar het is geïmporteerd door `leaflet`, een bibliotheek die je wel hebt draaien.)

Zie voor meer informatie over de bibliotheek `magrittr`

- de handleiding op het CRAN: https://cran.r-project.org/web/packages/magrittr/magrittr.pdf,
- deze vignette: https://cran.r-project.org/web/packages/magrittr/vignettes/magrittr.html

Figuur 6: het 'officiële' logo van het magrittr package

8.4 Een interactieve kaart - 3: een uitgebreid script

Voor de oefening in deze paragraaf verwijzen we naar het volgende blog: Maak een interactief kaartvenster in R/RStudio met leaflet.

Hier vind je een uitgebreid script - InteractiefKaartvensterInRStudio.R - waarmee meerdere lagen aan een kaart worden toegevoegd, inclusief een bijbehorende *LayersControl*.

9 Ruimtelijke analyses

9.1 Overlapanalyse (punt in polygoon): `over()`

Met de functie `over()` uit de `sp` bibliotheek kun je overlapanalyses doen. In deze paragraaf doen we wat eenvoudige oefeningen met punten in polygonen, namelijk stations in gemeenten.

De functie `over()` kijkt naar de locatie van *object x* (het station) en haalt de attribuutgegevens op van *object y* (de gemeente).

Bij de vergelijking van twee ruimtelijke objecten - een `SpatialPointsDataFrame` en een `SpatialPolygonsDataFrame` - komt de functie terug met een `data.frame` (een object zonder geometrie).

Probeer het volgende maar eens uit:

```
Gem_met_stations <- over(stations2015, gem2014)
class(Gem_met_stations)
View(Gem_met_stations)
```

Het data.frame `Gem_met_stations` bevat 398 observaties met 34 variabelen. Er zijn immers 398 stations, en voor elk station zijn alle 34 attribuutgegevens van de betreffende gemeente opgehaald. Sommige gemeenten komen meerdere keren voor, omdat er meer dan één station is.

OK - dit was om te laten zien hoe de functie werkt. Hieronder volgen twee praktische toepassingen.

9.1.1 Verrijken van een tabel met gegevens over de locatie - met `spCbind()`

We willen graag `stations2015` uitbreiden met een attribuutkolom **Gemeentenaam**.

Stap 1: genereer de attribuutkolom:

```
Gem_per_station <- over(stations2015, gem2014["Gemeentenaam"])
class(Gem_per_station)
View(Gem_per_station)
```

Nu willen we deze ene kolom toevoegen aan `stations2015`. Dit kan met `spCbind()` uit de bibliotheek `[maptools](https://cran.r-pro`
(`spCbind` verschaft *cbind-achtige* methoden voor Spatial*DataFrame objecten.)

Stap 2: voeg de kolom toe:

```
library(maptools)
stations2015 <- spCbind(stations2015, Gem_per_station)
View(stations2015)
```

En nu willen we ook graag een attribuutkolom **Provincie_naam** toevoegen.

Deze keer hebben we **beide stappen gecombineerd:**

```
stations2015 <- spCbind(stations2015, over(stations2015, gem2014["Provincie_naam"]))
View(stations2015)
```

9.1.2 Telling per gebiedseenheid - met `table()`

We willen graag weten hoeveel stations er in elke provincie liggen.

Met `over()` halen we de provincienaam bij elk station op, en met `table()` zetten we in één moeite door een telling van de voorkomens in een tabelletje:

```
stations_per_provincie <- table(over(stations2015, gem2014["Provincie_naam"]))
```

Toon het resultaat van de telling:

```
stations_per_provincie
```

```
##
##       Drenthe     Flevoland     Friesland    Gelderland     Groningen
##             9             8            24            63            29
##       Limburg Noord-Brabant Noord-Holland    Overijssel       Utrecht
##            39            35            60            36            30
##       Zeeland  Zuid-Holland
##             9            56
```

```
View(stations_per_provincie) # in de Data Viewer
```

9.2 Een geografische subset (met vierkante haken)

In **R** kun met behulp van vierkante haken selecties maken van een dataset (zie voor een uitleg van de syntaxis paragraaf 6.2.2).

Deze syntaxis kan ook gebruikt worden voor geografische selecties.

In het voorbeeld hieonder maken we eerst een subset `brabant` aan, en vervolgens selecteren we alle stations in die provincie:

```
brabant <- gem2014[gem2014$Provincie_naam == "Noord-Brabant", ]
stations_brabant <- stations2015[brabant, ]
```

De dataset `stations_brabant` bevat nu alle 35 stations stations die gelegen zijn in de Nederlandse provincie Noord-Brabant. En de dataset bevat ook de puntgeometrie van deze stations.

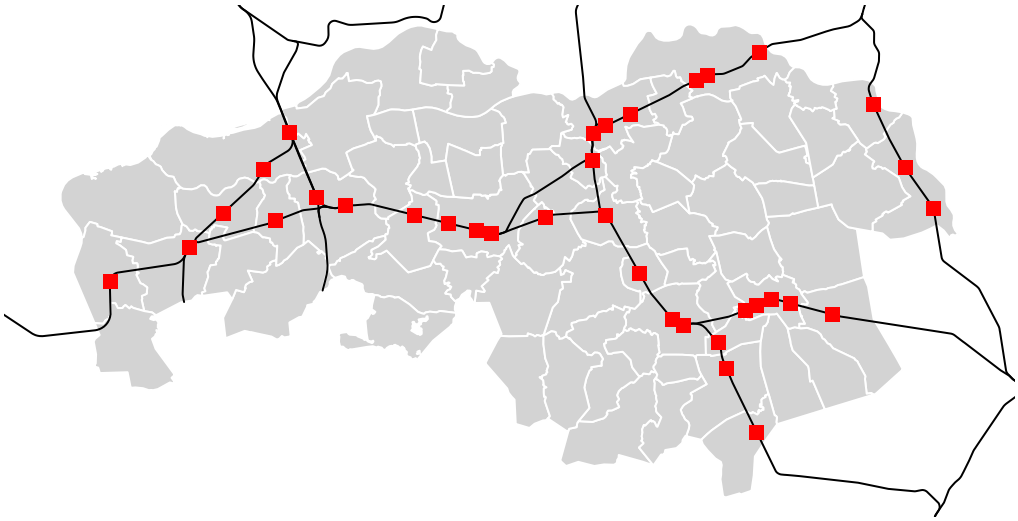
Kijk maar:

```
View(stations_brabant)
plot(stations_brabant)
class(stations_brabant)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

We kunnen dit tonen in een kaartje:

```
plot(brabant, col = "lightgrey", border= "white")
plot(spoorwegen2015, add = TRUE)
plot(stations_brabant, pch = 15, col = "red", add = TRUE )
```



10 Geocoderen

10.1 De bibliotheek `photon`

Er bestaan verschillende oplossingen voor het geocoderen van adressen en plaatsnamen met R. Vaak zijn deze oplossingen gebaseerd op commerciële diensten (bijvoorbeeld Google) en maken ze gebruik van niet-vrije data.

In deze paragraaf kijken we naar een oplossing die gebruik maakt van - vrij beschikbare - OpenStreetMap gegegevens, namelijk `photon`. Wij zijn deze bibliotheek op het spoor gekomen via deze blog: Géocoder en masse avec R et sans Google Maps.

Het *package* `photon` biedt een **R** Interface naar de Photon API.

Photon is een open source geocoder gebouwd voor OpenStreetMap data en is gebaseerd op elasticsearch. Met deze bibliotheek kun je een Photon API bevragen. Het resultaat wordt gegeven in het formaat van een data frame.

`photon` is nog in ontwikkeling, en daarom nog niet beschikbaar op het **CRAN**, maar wel op GitHub: https://github.com/rCarto/photon.

Om `photon` te kunnen installeren heb je `devtools` nodig (zie paragraaf 2.4.3):

```
install.packages("devtools")
```

Nu kun je `photon` installeren en laden:

```
require(devtools)
devtools::install_github(repo = 'rCarto/photon')
library(photon)
```

En nu kun je gaan geocoderen:

```
loc <- c("Wierdijk 12 - 22,1601 LA Enkhuizen", # Zuiderzeemuseum, Enkhuizen
         "Avenue Ceramique 250, 6221 KX Maastricht") # Bonnefantenmuseum, Maastricht
geocode(loc, limit = 1, key = "place")
```

```
## [1] "Wierdijk 12 - 22,1601 LA Enkhuizen"
## [1] "Avenue Ceramique 250, 6221 KX Maastricht"
```

```
##                                location name housenumber
## 1       Wierdijk 12 - 22,1601 LA Enkhuizen <NA>          12
## 2 Avenue Ceramique 250, 6221 KX Maastricht <NA>         250
##             street postcode       city         state         country
## 1         Wierdijk   1601LA  Enkhuizen North Holland The Netherlands
## 2 Avenue Ceramique   6221KX Maastricht       Limburg The Netherlands
##   osm_key osm_value      lon      lat  msg
## 1   place     house 5.297224 52.70379 <NA>
## 2   place     house 5.702028 50.84252 <NA>
```

11 Rasteranalyse

11.1 De bibliotheek `raster`

Met **R** kun je ook uitgebreide rasteranalyses uitvoeren.

Voor dit hoofdstuk moeten we nog een aantal eenvoudige oefeningen bedenken.

De auteur van dit boek heeft al wel deze blog geschreven: raster(): het samenvoegen van rasterbestanden in R

Literatuurlijst

- De Bruijne, Arnoud, Van Buren, Joop, Kösters, Anton & Van der Marel, Hans (2005). *De geodetische referentiestelsels van Nederland; Definitie en vastlegging van ETRS89, RD en NAP en hun onderlinge relaties*.Nederlandse Commissie voor Geodesie (NCG), Delft, maart 2005. URL: http://www.ncgeo.nl/phocadownload/43Referentie.pdf

- Frazier, Melanie (z.d.). *Overview of Coordinate Reference Systems (CRS) in R*. Geraadpleegd op 12 november 2015. URL: https://www.nceas.ucsb.edu/~frazier/RSpatialGuides/OverviewCoordinateReferenceSystems.pdf

- Ihaka, Ross (2009). *The R Project: A Brief History and Thoughts About the Future*. Presentatie gehouden op de Massey University Statistics Day, 23 oktober 2009, Massey University, Palmerston North. URL: https://www.stat.auckland.ac.nz/~ihaka/downloads/Massey.pdf

- Pebesma, Edzer & Bivand, Roger S. (2005). *Classes and Methods for Spatial Data: the sp Package*. Eén van de vignettes behorende bij de sp bibliotheek. URL: https://cran.r-project.org/web/packages/sp/vignettes/intro_sp.pdf

- Peng, Roger D. (2015). *Exploratory Data Analysis with R*. Gepubliceerd als e-book op Leanpub. Er is een versie gepubliceerd op 12 november 2015. URL: http://leanpub.com/exdata

- Venables, W. N., Smith, D. M. en het R Core Team (2015). *An Introduction to R. Notes on R: A Programming Environment for Data Analysis and Graphics*. Version 3.2.2 (2015-08-14). URL: https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf