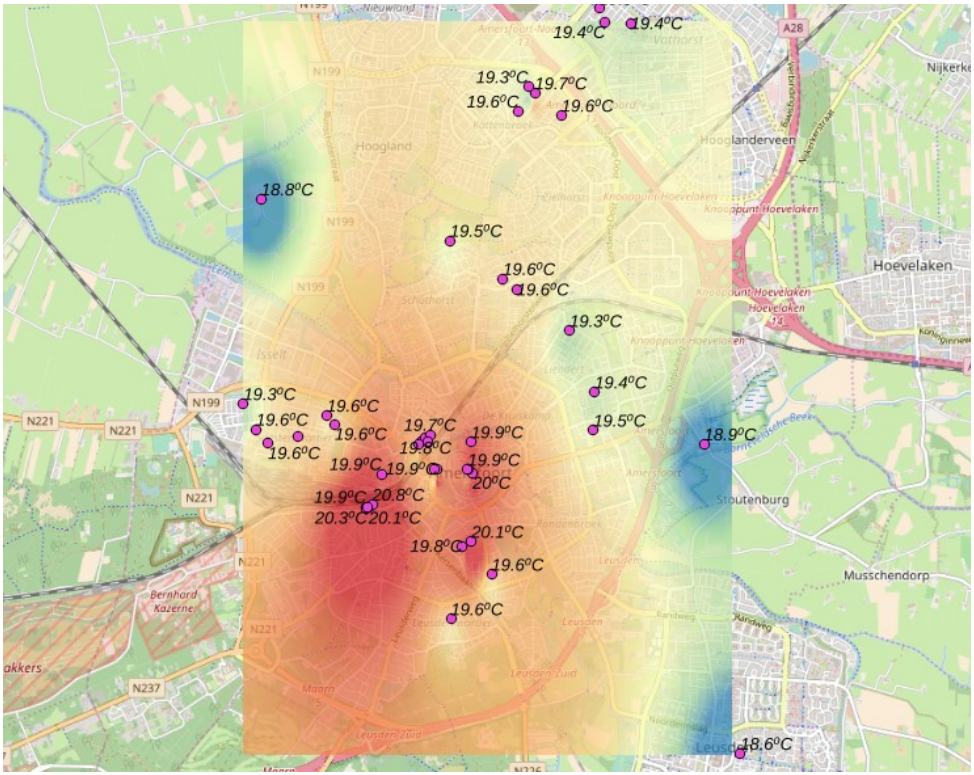
COÖPERATIEVE UNIVERSITEIT AMERSFOORT

Meet je stad!

Instructie dataverwerking



Inhoud

Inleiding

Deze handleiding is geschreven als onderdeel van het Meet Je Stad! Project van de Universiteit Amersfoort. In dit project worden effecten en ervaringen omtrent klimaatverandering in kaart gebracht.

Eén van de manieren waarop dat gebeurt is door het meten van diverse klimaat-gerelateerde waarden op verschillende plekken in de stad. Hiermee kunnen trends en verschillen binnen de stad in kaart gebracht worden.

De verzamelde data worden opgeslagen in een database. Deze handleiding geeft een introductie in het verwerken van deze gegevens tot betekenisvolle grafieken en kaarten.

Voorbereiding

Voor deze workshop heb je een computer nodig waarop we twee programma's zullen gebruiken: R en QGIS.

R

R is een open source softwarepakket en **programmeertaal** ontwikkeld voor statistiek en data-analysedoeleinden. Zie www.r-project.org

1. Download en installeer R via deze pagina: cran.r-project.org
2. Start R

Je krijgt nu een **terminalscherm** waarbinnen je commando's kunt typen, gevolgd door een Enter. Bijvoorbeeld:

```
> 1 + 1                 werkt R met dezelfde wiskunde als jij?
> r <- 5                     maak een variabele r met waarde 5
> pi * r^2              geef oppervlak van cirkel met straal 5
```

Meer **voorbeelden en tutorials** zijn hier te vinden:

- tryr.codeschool.com (leuke online R beginnerscursus)
- www.cyclismo.org/tutorial/R/input.html (toegankelijke beginnerstutorial)
- http://www.r-tutor.com/r-introduction (nog tutorial)
- www.sr.bham.ac.uk/~ajrs/R/r-getting_started.html (overzicht van commando's)

We gaan in deze workshop de **grafische module** 'ggplot2' gebruiken, deze kun je installeren door het volgende commando te typen:

```
> install.packages("ggplot2")
```

Je kunt R weer **afsluiten** door het het volgende commando te typen:

```
> quit()
```

Een ggplot **cheatsheet** vind je hier:

- www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

QGIS

QGIS is een open source **geografisch informatiesysteem** (GIS) om geografische gegevens bekijken, bewerken en analyseren. Zie www.qgis.org

1. Download en installeer QGIS via deze pagina: www.qgis.org/en/site/forusers/download.html
2. Start QGIS

Je krijgt een scherm met heel veel knoppen, toeters en bellen. Schrik niet, die hoef je niet allemaal te kennen. We gaan er in deze workshop maar een paar gebruiken.

Naast de basisfuncties die zitten ingebouwd in QGIS hebben we een aantal extra **plugins** nodig, die je kunt installeren via het menu 'Plugins > Manage and Install Plugins...'

1. Installeer de plugin genaamd 'QuickMapServices'
2. Installeer de plugin genaamd 'Interpolation plugin'
3. Ga nu naar 'Web > QuickMapServices > Settings'
4. Vink het vakje aan achter 'Enable OTF EPSG:3857 on tiled layer add:'

Alles geïnstalleerd? Gefeliciteerd, je alles is nu gereed voor je eerste analyse van meetgegevens van Meet je stad!

Wil je later meer leren over QGIS, dan zijn dit een paar **handige websites**:

- docs.qgis.org/2.18/en/docs/training_manual/ (toegankelijke tutorial voor beginners)
- http://docs.qgis.org/2.18/en/docs/user_manual/ (handleiding QGIS)

Meetreeksen bekijken en vergelijken

We gaan nu meetreeksen inlezen en omzetten in grafieken. Dat doen we in de programmeertaal R.

Geen paniek!
Het kan zijn dat je wiskunde of code tegenkomt die je niet begrijpt. Dat is geen reden om je af te laten schrikken. Het beste kun je proberen het te lezen, kijken welke delen ervan je begrijpt en een gokje wagen over wat je denkt dat het stukje doet. Als je wat meer gewend raakt aan de code of de wiskunde zul je beter gokken.

Om de code beter te kunnen lezen is het handig om eerst even vijf minuten het begin van de **online cursus** R te doen, tot en met hoofdstuk 1.4:
tryr.codeschool.com/levels/1/challenges/1

In dit hoofdstukje staat steeds een regel code met een uitleg wat deze doet. Een regel met een **#** ervoor is een **commentaarregel**. Hier wordt door R niets mee gedaan. Hij staat er alleen als geheugensteuntje of uitleg voor onszelf.

Een regel met een **>** ervoor is een regel **code**. Deze voer je in in de commandline van R.

Tip: Je kunt de tekst overtypen, maar je kunt de code ook vinden op meetjestad.net/data/handleiding.r en daaruit knippen en plakken.
Tip: Je kunt het programmaatje regel voor regel invoeren maar ook een heel blok code in één keer in R plakken.

Als je wilt weten wat alle commando's betekenen en wat er er verder mee kunt doen dan kun je meer informatie online vinden, bijvoorbeeld op de webpagina's die genoemd zijn in het hoofdstukje 'Voorbereiding'.

Toch paniek?
Wanneer je door de bomen het bos niet meer ziet kan het helpen om nog eens een stukje van de online cursus te doen. Ook helpt het vaak om een aantal stappen terug te gaan en het gewoon nog een keer te doen, net als je een moeilijke tekst soms een paar keer moet lezen voor je 'm gaat begrijpen.

Een meetreeks bekijken

Eerst gaan we eens kijken hoe we de meetreeks van een sensor uit de database kunnen halen en weergeven als grafiek.

Kies een sensor

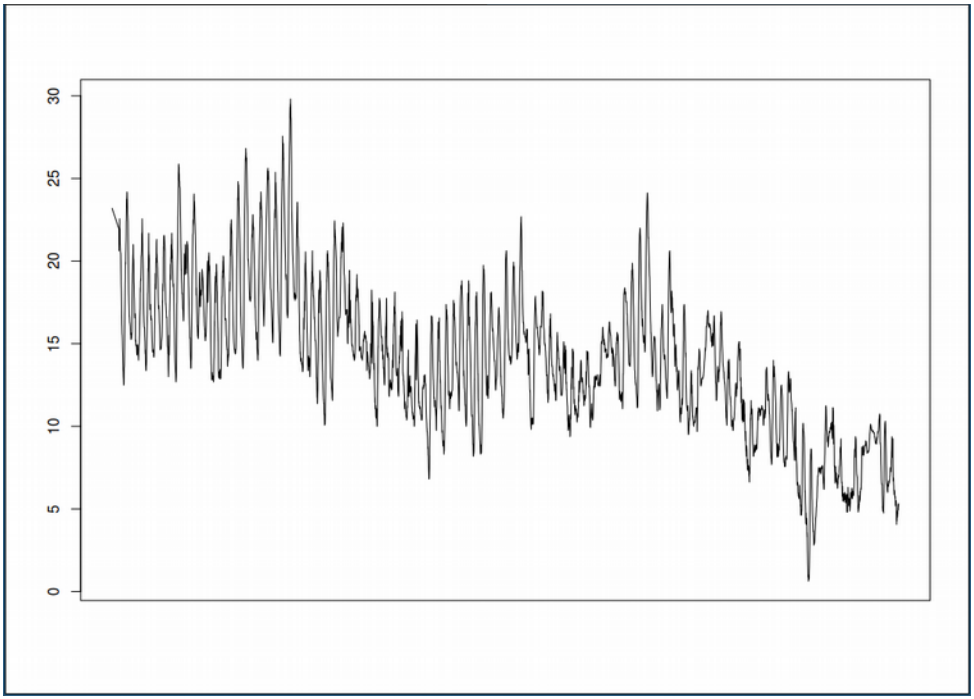
```
> id<-81
```

Haal de data op van de server van Meet je Stad

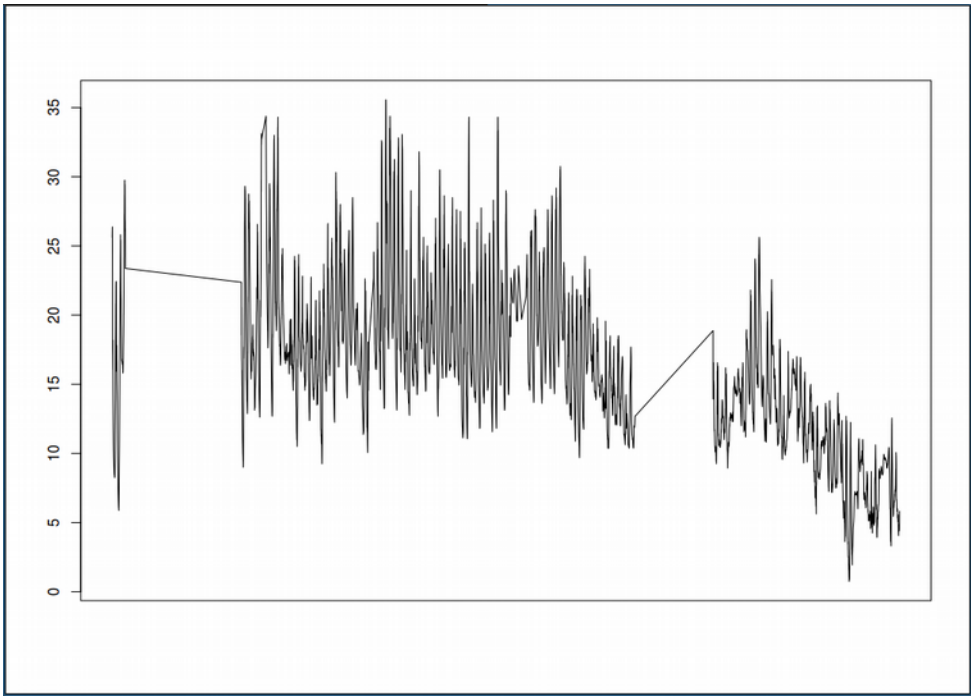
```
> data<-read.table(paste("http://meetjestad.net/data?
type=sensors&ids=",id,"&format=csv", sep=""), sep="\t",
header=T)
```

Teken een lijngrafiek

```
> plot(as.POSIXct(data$timestamp), data$temperature,
type="l")
```



Wat zie je in deze grafiek. Vergelijk dat eens met sensor 65. Wat zie je daar? Hoe verklaar je de verschillen?



De gegevens van één sensor beter bekijken

Om een meetreeks wat beter te kunnen bekijken hebben we wat meer code nodig. We gaan ook een andere, krachtiger, functie gebruiken om de grafieken te tekenen.

laad de library die R gebruikt om grafieken weer te geven

```
> library(ggplot2)
```

maak een variabele genaamd 'start', stop hier het beginmoment van je grafiek in

```
> start<-"2017-11-16,00:00"
```

maak een variabele genaamd 'eind', met het eindmoment van je grafiek

```
> end<-"2017-11-18,00:00"
```

maak variabele 'ids', met het nummer van de sensor die je wilt bekijken

voorbeelden: "13", "13,14", "1-50", "1,2,10-20", etc.

```
> ids<-"13"
```

haal de data op van de server van Meet je Stad

```
> data<-read.table(paste("http://meetjestad.net/data?
type=sensors&start=",start,"&end=",end,"&ids=",ids,"&format=
csv", sep=""), sep="\t", header=T)
```

op de server zijn alle metingen opgeslagen in Universal Time Code. Laten we deze globale tijd omrekenen naar onze lokale tijd

```
> ts<-as.POSIXct(data$timestamp, origin="1970-01-01",
tz="UTC")
> data$localtime<-as.POSIXlt(ts, "CET")
> attributes(data$localtime)$tzone <- "CET"
```

zoek de hoogste en de laagste temperatuur van de opgehaalde datareeks

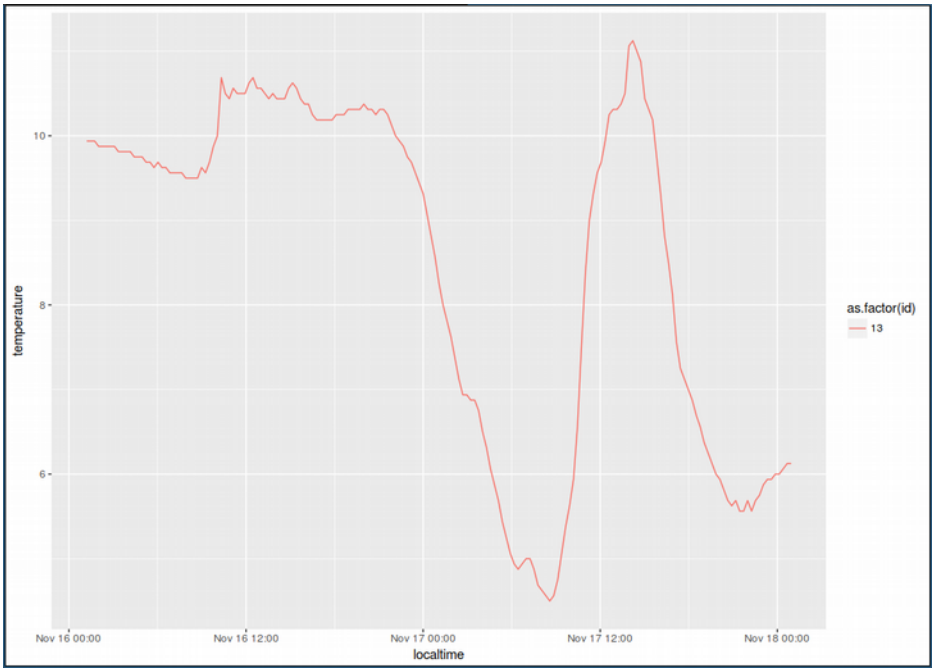
we gebruiken de onderste 5% en bovenste 5% van de meetwaarden, en trekken daar 1 graad of of tellen het erbij

```
> ylow<-quantile(data$temperature,0.05)-1
> yhigh<-quantile(data$temperature,0.95)+1
```

maak een grafiek van de data (lijnen)

```
> ggplot(data=data, aes(localtime,temperature, colour =
  as.factor(id))) +
  geom_line(aes(group = as.factor(id)) ) +
  ylim(ylow,yhigh)
```

Als het goed is heb je nu een grafiek die er ongeveer zo uit ziet:



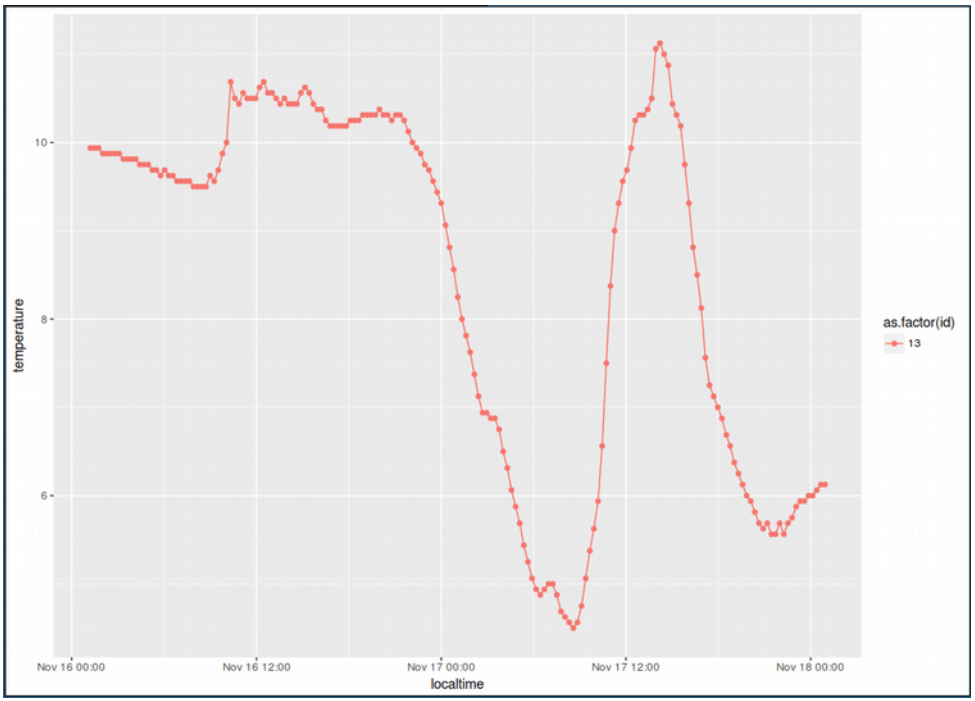
alternatieve berekening van de minima en maxima (welke werkt beter?)

```
> ylow<-boxplot(data$temperature, data=data)$stats[1]
> yhigh<-boxplot(data$temperature, data=data)$stats[5]

> ggplot(data=data, aes(localtime,temperature, colour =
  as.factor(id))) +
  geom_line(aes(group = as.factor(id)) ) +
  ylim(ylow,yhigh)
```

Maak ook een grafiek zonder de 'ylim' (schaal vande y-as). Bedenk zelf hoe je dit doet...

maak een grafiek van de data (lijnen en punten) ter vervanging van de vorige grafiek

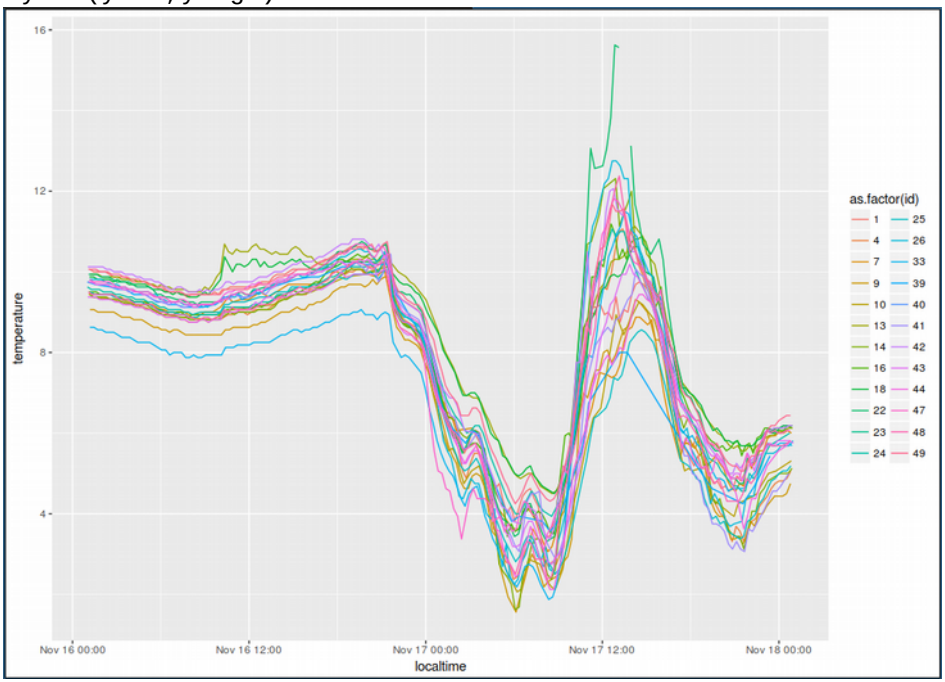
```
> ggplot(data=data, aes(localtime,temperature, colour =
  as.factor(id))) +
  geom_line(aes(group = as.factor(id)) ) +
  geom_point(aes(group = as.factor(id)) ) +
  ylim(ylow,yhigh)
```



Meetgegevens van meer sensoren vergelijken

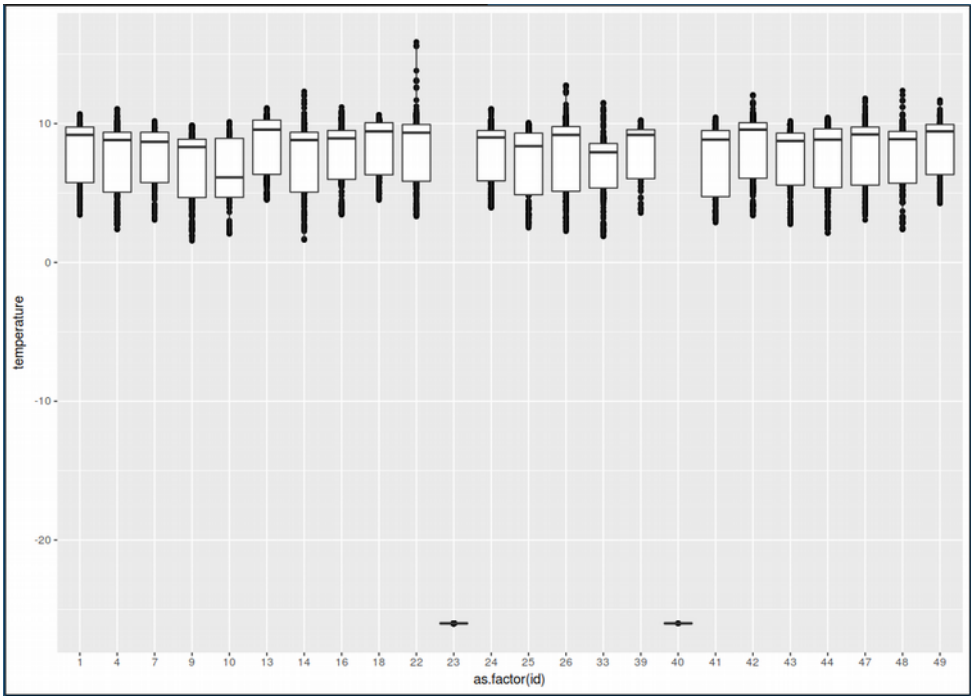
laten de we eerste 50 meetstations eens onderzoeken

```
> ids<-"1-50"

> data<-read.table(paste("http://meetjestad.net/data?
type=sensors&start=",start,"&end=",end,"&ids=",ids,"&format=
csv", sep=""), sep="\t", header=T)

> ts<-as.POSIXct(data$timestamp, origin="1970-01-01",
tz="UTC")
> data$localtime<-as.POSIXlt(ts, "CET")
> attributes(data$localtime)$tzone <- "CET"

> ylow<-quantile(data$temperature,0.05)-1
> yhigh<-quantile(data$temperature,0.95)+1

> ggplot(data=data, aes(localtime,temperature, colour =
  as.factor(id))) +
  geom_line(aes(group = as.factor(id)) ) +
  ylim(ylow,yhigh)
```



Een andere manier van weergeven is een boxplot van de metingen per station

```
> ggplot(data,
  aes(as.factor(id), temperature )) +
  geom_point() + geom_boxplot()
```



Wat zie je eigenlijk in deze grafiek, en wat valt je daarin op?

Nog een boxplot

```
> boxplot(temperature~id, data=data, xlab="station", las=3,
  ylab="temperatuur", cex.axis=0.7 )
```

Meet je stad-metingen vergelijken met KNMI-metingen

zet de starttijd en eindtijd om in het formaat van de KNMI API

```
> knmistart<-unlist(strsplit(start, ","))[1]
> knmistart<-gsub("-", "", knmistart)
> knmiend<-unlist(strsplit(end, ","))[1]
> knmiend<-gsub("-", "", knmiend)
```

haal de data op van het KNMI

```
> knmidata<-read.table(paste("http://projects.knmi.nl/klimat
ologie/uurgegevens/getdata_uur.cgi?
stns=260&vars=T:U&start=",knmistart,"01&end=",knmiend,"24",s
ep=""), sep=",", header=F)
```

voeg kolomnamen toe

```
> colnames(knmidata) <- c("station", "day", "hour",
"temperature", "humidity")
```

zet de KNMI UTC tijd om naar lokale tijd

```
> tsknmi<-as.POSIXct(strptime(paste(knmidata$day,knmidata$ho
ur, sep=""), "%Y%m%d%H"), tz="UTC")
```

verschuif de KNMI tijd 30 minuten naar voren, omdat de meetwaarden het gemiddelde zijn over het voorgaande uur

```
> knmidata$localtime <- as.POSIXlt(tsknmi, "CET", tz="CET")-
30*60
> attributes(knmidata$localtime)$tzone <- "CET"
```

voeg een kolom toe met het station 'id': knmi

```
> knmidata$id<-rep("knmi",dim(knmidata)[1])
```

deel de temperatuur door 10 om dezelfde schaal als MJS te krijgen

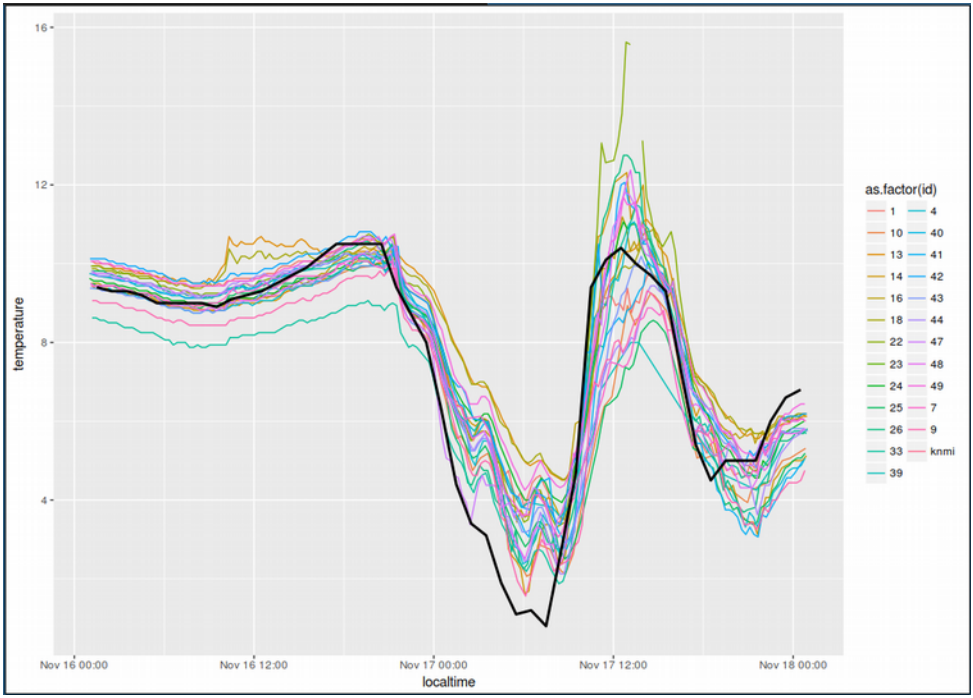
```
> knmidata$temperature<- knmidata$temperature/10
```

voeg de MJS en KNMI dataframes samen

```
> library(plyr)
> mjsknmidata<-rbind.fill(data, knmidata)
```

Plot!

```
> ggplot(data=mjsknmidata,
  aes(localtime,temperature, colour = as.factor(id))) +
  geom_line(aes(group = as.factor(id)) ) +
  geom_line(data=subset(mjsknmidata,id=="knmi"),
  colour="black", size=1 ) +
  ylim(ylow,yhigh)
```



Alle data op een meetstokje

De meetstations sturen elk kwartier hun metingen door, maar wanneer precies is afhankelijk van op welk tijdstip ze ooit zijn aangezet. Om de metingen onderling te kunnen vergelijken is het handig om ze op een vast grid te zetten. 12:00, 12:15, 12:30, 13:00 enzovoorts.

Om dat voor elkaar te krijgen moeten we per sensor de meetgegevens interpoleren. Daarna kunnen we de aangepaste dataset gebruiken om vergelijkingen te maken tussen sensoren.

maak een vector met de vroegste en laatste tijd

```
> tsrange <- range(data$localtime)
```

rond deze start- en eindtijd af naar het dichtstbijzijnde kwartier

```
> startkwartier<-as.POSIXlt(round(as.double(tsrange[1])/
(15*60))*(15*60),origin=(as.POSIXlt('1970-01-01')))
```

eindkwartier<-as.POSIXlt(round(as.double(tsrange[2])/

(15*60))*(15*60),origin=(as.POSIXlt('1970-01-01')))

maak een reeks van het beginkwartier tot het eindkwartier in stappen van 15 minuten

```
> tssequence <- seq(startkwartier+15*60,eindkwartier-15*60,
by=15*60)
```

maak een data frame met een lege matrix voor alle stations en kwartieren

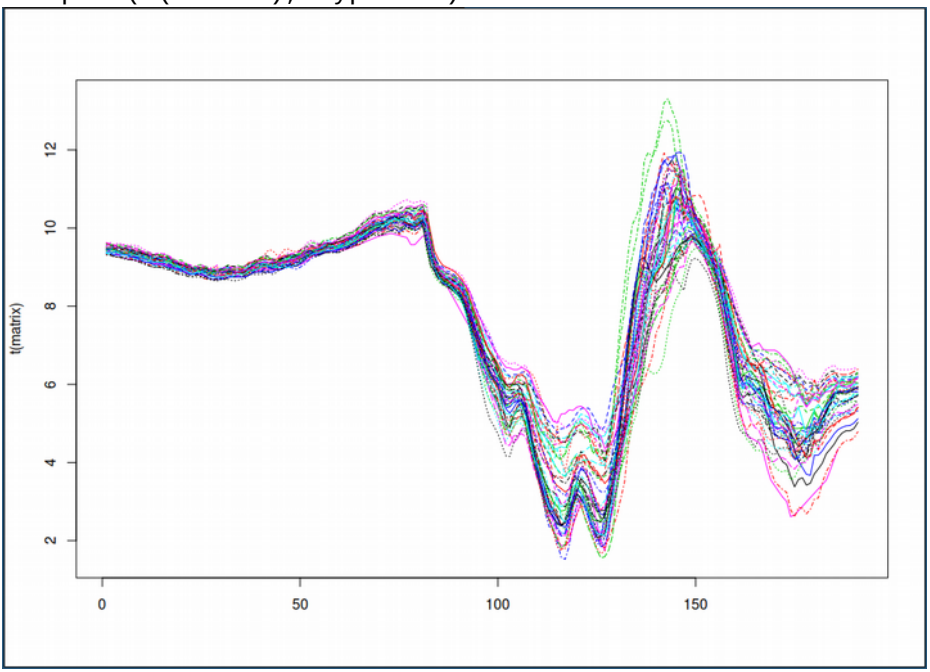
```
> matrix<-data.frame(matrix(,nrow=length(levels(factor(data$
id))),ncol=length(tssequence) ) )
```

van iedere reeks metingen van een station, berekenen we de lineaire interpolatie naar de kwartier tijden

```
j<-0
for (i in as.numeric(levels(factor(data$id))) ) {
  tree <- subset(data, id==i)
  ltime <- tree$localtime
  if (length(ltime) > 1) {
    j<-j+1
    interpol<-approx( as.numeric(ltime),tree$temperature,
xout=as.numeric(tssequence), rule=1, method = "linear",
ties=mean)
    matrix[j,] <- c(interpol$y)
  }
}
```

geeft de kolommen en rijen namen en maak een simpele plot van de (getransponeerde) matrix

```
> colnames(matrix)<-tssequence
> rownames(matrix)<-levels(factor(data$id))
> matplot(t(matrix), type="l")
```



Nu we alle data vergelijkbaar hebben gemaakt kunnen we er wat analyses mee gaan doen.

Bereken de gemiddelde temperatuur per station. Vervang 'mean' door 'min' of 'max' om de minima en maxima te zien

```
> apply(matrix,1,mean, na.rm=T)
```

Bereken de gemiddelde temperatuur per kwartier. Vervang 'mean' door 'min' of 'max' om de minima en maxima te zien. Gebruik 'median' voor de mediaan (stabieler dan het gemiddelde)

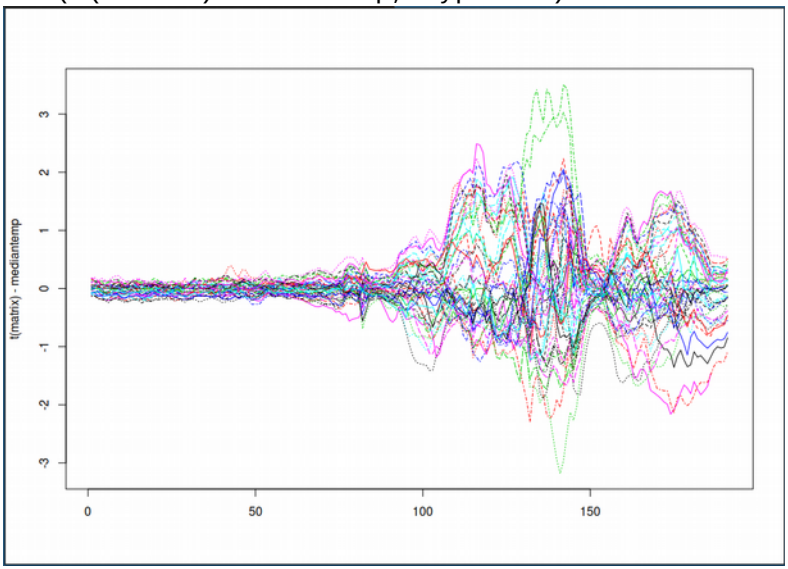
```
> apply(matrix,2,mean, na.rm=T)
```

Plot de modale temperatuur

```
> mediantemp <- as.numeric(apply(matrix,2,median, na.rm=T))
plot(tssequence, mediantemp, ylab="temperatuur",
xlab="tijd", type="b", col="blue")
```

Plot het verschil met de modale temperatuur voor alle stations

```
> matplot(t(matrix)-mediantemp, type="l")
```



Bereken de maximale afwijking van het gemiddelde per station

```
> apply(t(matrix)-mediantemp,2,max, na.rm=T)
```

Zoek uit welke stations meer dan 10 graden afwijken!

```
> slechtestations <- abs(apply(t(matrix)-mediantemp,2,max,
na.rm=T)) > 10
> which(slechtestations==T)
```

Maak een matrix met alleen de goede stations

```
> goedematrix <- matrix[!slechtestations,]
```

Plot de goede stations!

```
> matplot(t(goedematrix), type="l")
```

Plot het verschil tussen 2 stations (zorg dat ze in de data frame zitten!)

```
> station1<-13
> station2<-14
> plot(tssequence, as.numeric(matrix[levels(factor(data$id))
==station1,] - matrix[levels(factor(data$id))==station2,]),
ylab="temperatuur", xlab="tijd", type="b", col="blue")
```

Zoek de maximale standaard deviatie tussen de stations

```
> max(apply(goedematrix,2,sd, na.rm=T))

# Zoek het tijdstip met de maximale standaard deviatie
tussen de stations

> which.max(apply(goedematrix,2,sd, na.rm=T))

# Plot de standaard deviatie

> plot(tssequence, apply(goedematrix,2,sd, na.rm=T),
ylab="sd temperatuur", xlab="tijd", type="b", col="blue")
```

Floradata bekijken en analyseren

De flora-observaties van Meet je stad zitten anders in elkaar dan de sensordata. We moeten dus een ander bestand inladen, en ook over andere manieren nadenken om de data te visualiseren. In dit voorbeeld maken we een stippenplot, die eenvoudigweg in een tijdlijn per soort aangeeft wanneer er een observatie van die soort is gemeld. We gebruiken daarvoor alleen de datum van de melding en de Nederlandse naam.

Laad de library in die R gebruikt om grafieken weer te geven

```
> library(ggplot2)
```

schrijf begin- en eindmoment van je grafiek in de variabelen 'start' en 'end'

```
> start<-"2016-01-01,00:00"
> end<-"2017-11-17,00:00"
```

haal de data op van de server van Meet je Stad

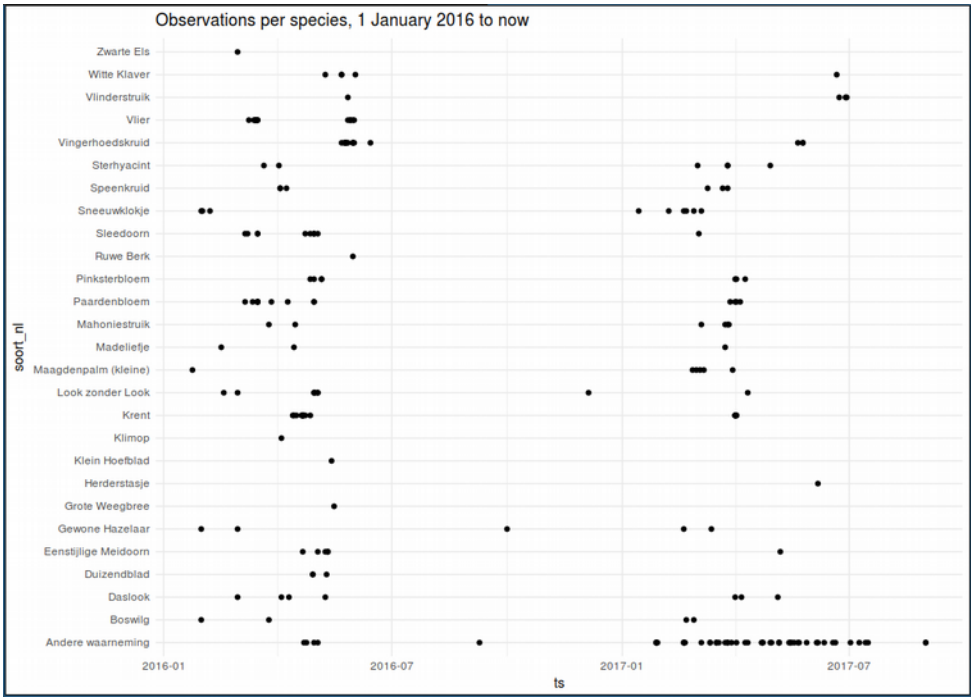
```
> data<-read.table(paste("http://meetjestad.net/data?
type=observations&start=",start,"&end=",end,"&format=csv",
sep=""), sep="\t", header=T)
```

vertel het programma hoe de datum in elkaar zit

```
> ts<-as.POSIXct(data$datum, origin="1970-01-01", tz="UTC")
```

maak een stippenplot van de data

```
> ggplot(data, aes(ts,soort_nl )) +
  geom_point( aes(ts,soort_nl))+ theme_minimal() +
  labs(title = "Observations per species, 1 January 2016 to
now")
```



Dit geeft al een mooie plot, waaraan je al kunt zien welke observaties zeker niet kloppen. Ook zie je als je twee jaren bekijkt een vorm die terugkeert. Je kunt een beetje spelen met de begin-en -einddatum en je eigen onderzoekje starten.

Er zijn nog meer gegevens over de metingen. Zo weten we ook op welke plek de plant staat.

Wat zou je nog meer willen weten? Hoe zou je dat het beste kunnen visualiseren? In een grafiek? Wat voor soort grafiek kun je daarvoor het beste tekenen? Of op een kaart? En met wat voor andere data zou je de flora-observaties willen vergelijken?

Een hittekaart maken

Download databestand

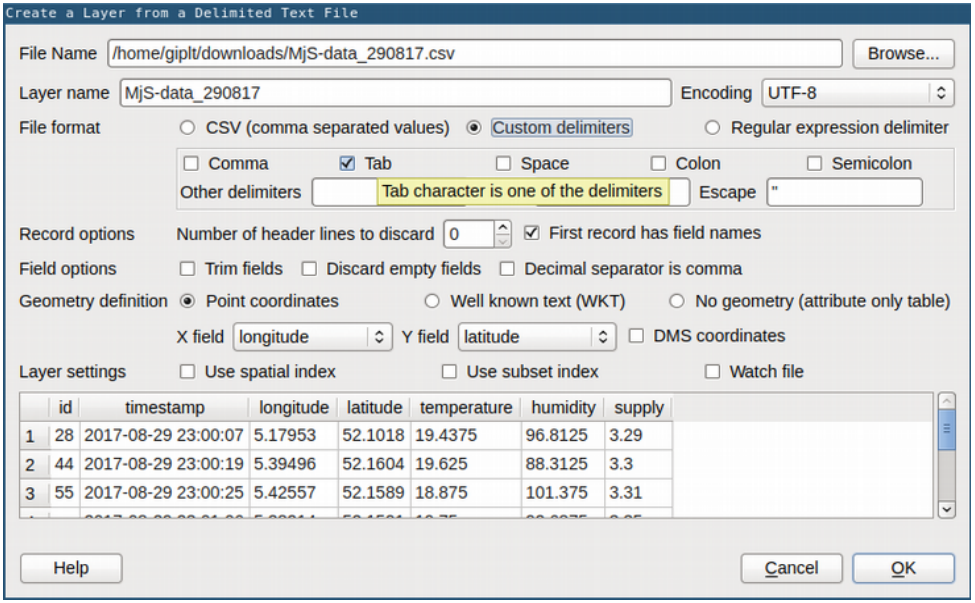
1. Ga naar meetjestad.net/data
2. Bij 'Soort data': kies 'Metingen'
3. Bij 'Periode': selecteer 15 minuten op een dag na half augustus 2017 (toen de grootschalige metingen van Meet je stad begonnen).
4. Bij 'Sensor(en)': selecteer 'Amersfoort – wijkvergelijking'
5. Download als CSV bestand

Je hebt nu een bestand met gegevens die één momentopname vormen. In het bestand staat één grote tabel, met rijen door tabs gescheiden waarden. In de eerste rij staan de namen van de waarden, ondermeer tijdstip en locatie van de meting, en temperatuur. Je kunt het bestand bekijken in een tekst-editor of spreadsheetprogramma.

Importeer databestand

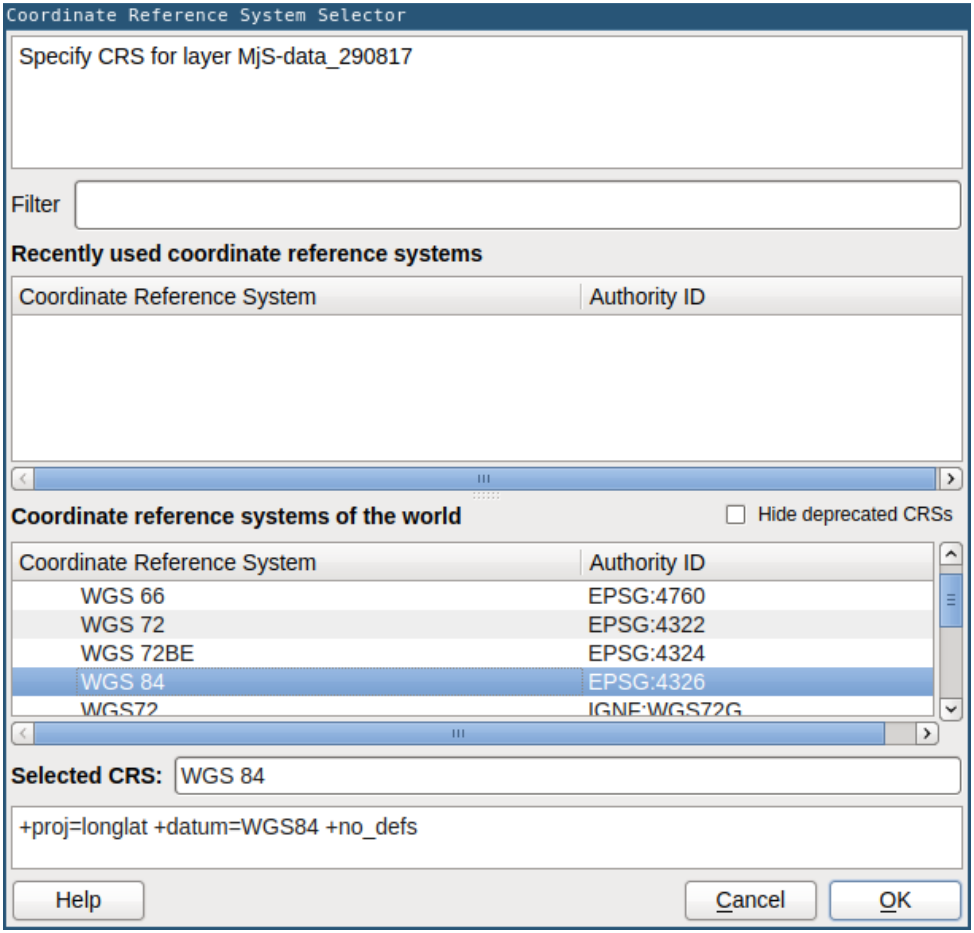
Om van de metingen een hittekaart te maken gebruiken we het programma QGIS. In dat programma kun je verschillende soorten geografische gegevens inladen, bewerken en visualiseren in kaartlagen.

1. Open QGIS
2. Zorg dat je 'Layers panel' zichtbaar is. Hierin komen de verschillende kaartlagen te staan. Dit lijstje kun je aan en uit zetten via 'View > Panels > Layers Panel'.
3. Ga nu naar 'Layer > Add Layer > Add Delimited Text Layer...'



4. Bij 'File Name': kies het bestand dat je zojuist hebt gedownload
5. Bij 'File format': kies 'Custom delimiters' en vink 'Tab' aan
6. Bij 'Geometry definition': controleer dat 'X field' staat op 'longitude', en 'Y field' op 'latitude'.
7. Klik OK

In het volgende scherm moet je nog een coördinatenstelsel kiezen. Om een geografische locatie te beschrijven zijn er vele mogelijkheden. Elk land heeft z'n eigen manier, sommige landen meerdere. De GPS-gegevens van de Meet je stad sensoren worden opgeslagen in het universele WGS84 formaat.



8. Selecteer als Coordinate Reference System: 'WGS 84'
9. Klik OK

In het Layers panel is nu een kaartlaag toegevoegd. Klik rechts en dan op 'Rename' om de naam te wijzigen, bijvoorbeeld in 'MjS Sensoren'

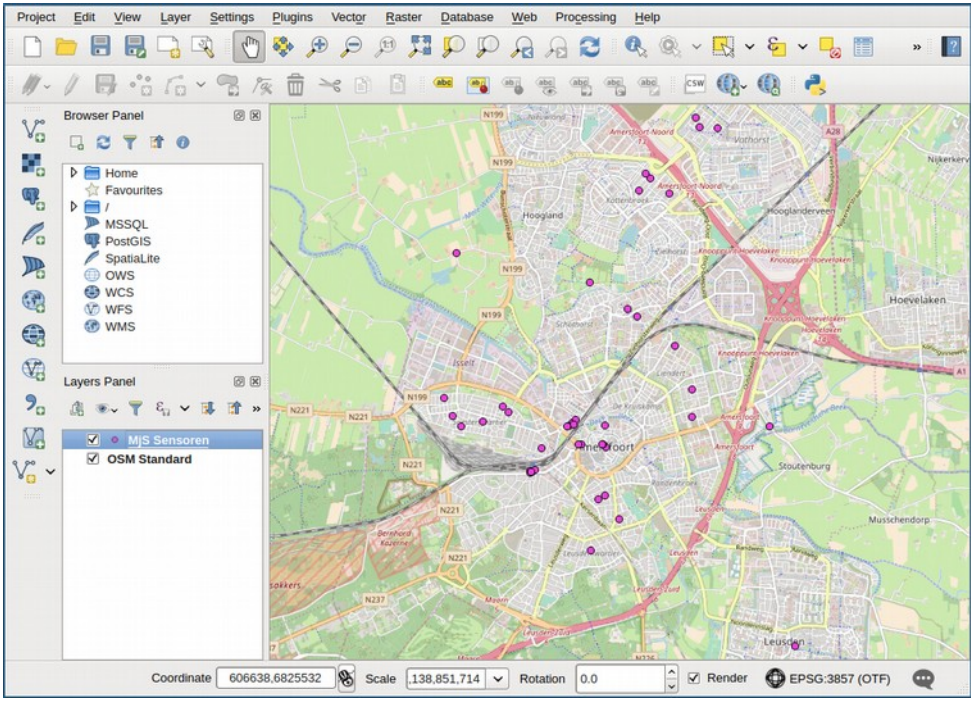
Laad achtergrondkaart

Als alles goed is gegaan zie je nu een hoop stipjes op een witte achtergrond. Deze stipjes geven de locaties aan van de meetstations.

Tip: Een handige functie in QGIS is de mogelijkheid om automatisch te schalen naar een vergroting waarbij alle objecten in een kaartlaag zichtbaar zijn. Klik rechts op 'MjS Sensoren' in het Layers panel en selecteer 'Zoom to layer'.

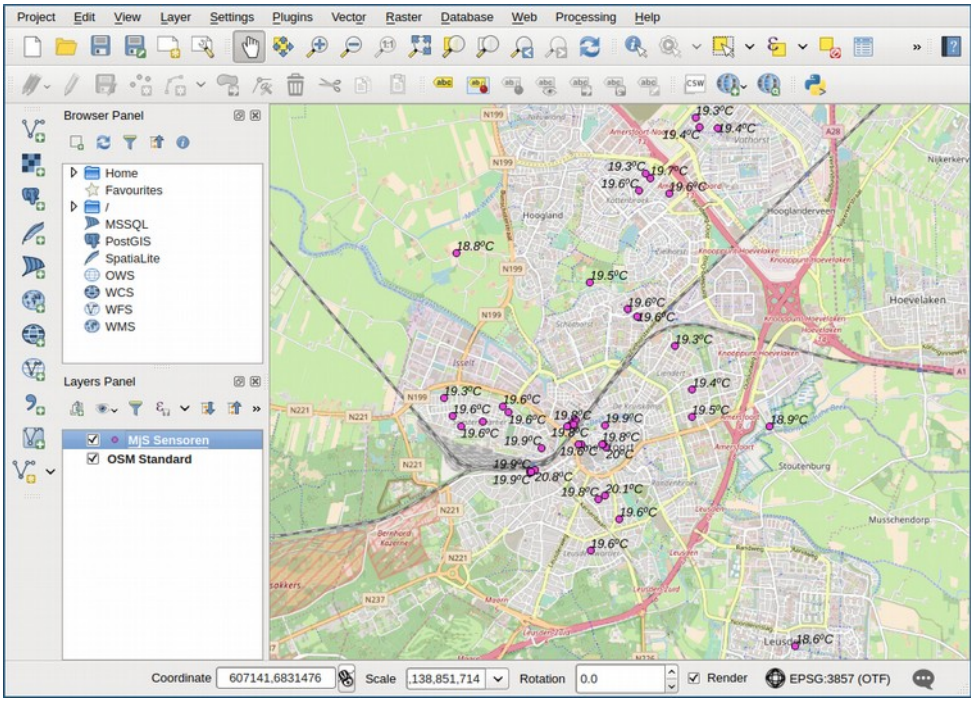
Zonder extra informatie is een wit scherm met stippen echter niet zo betekenisvol. We gaan daarom een extra kaartlaag toevoegen, de kaart van OpenStreetMap.

1. Ga naar 'Web > QuickMapServices > OSM > OSM Standard'
2. Zorg dat de achtergrondkaart onder de datakaart ligt, dat kan door de kaartlaag 'MjS Sensoren' in deze lijst boven de kaartlaag 'OSM Standard' te plaatsen.



Voeg labels toe

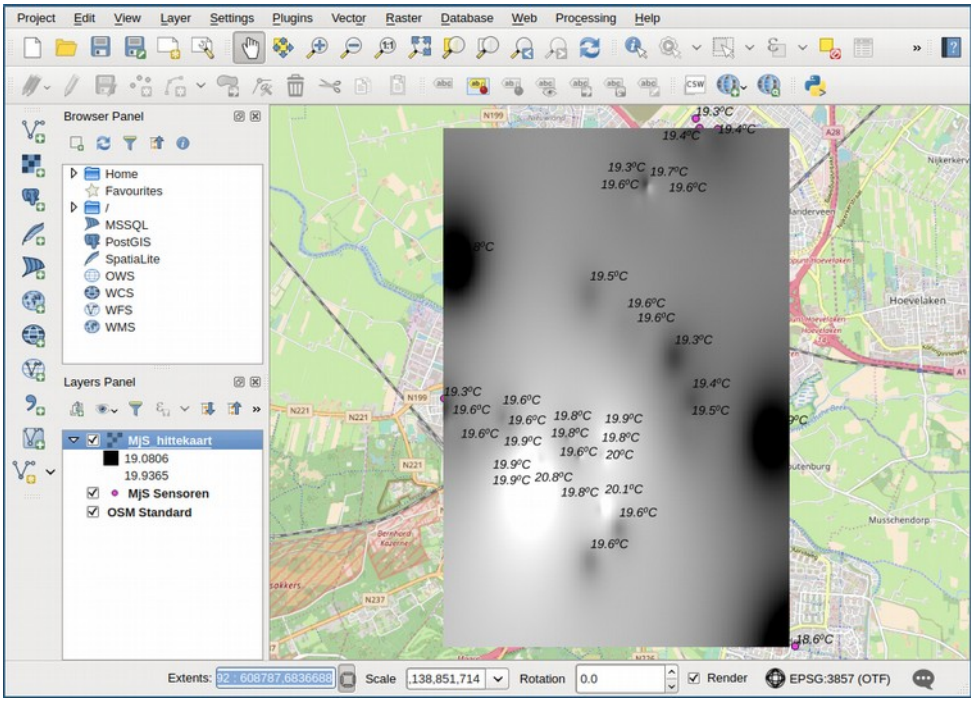
1. Selecteer 'MjS Sensoren' in het layers panel
2. Ga naar 'Layer > Labeling'
3. Kies 'Show labels for this layer'
4. Bij 'Label with': kies 'temperature'
5. Klik OK



Tip: Om de labels mooier te maken kun je een formule samenstellen. Vul bij 'Label with:' bijvoorbeeld eens in “concat(round(temperature, 1), '°C')”. Kun je begrijpen wat er in deze formule gebeurt?

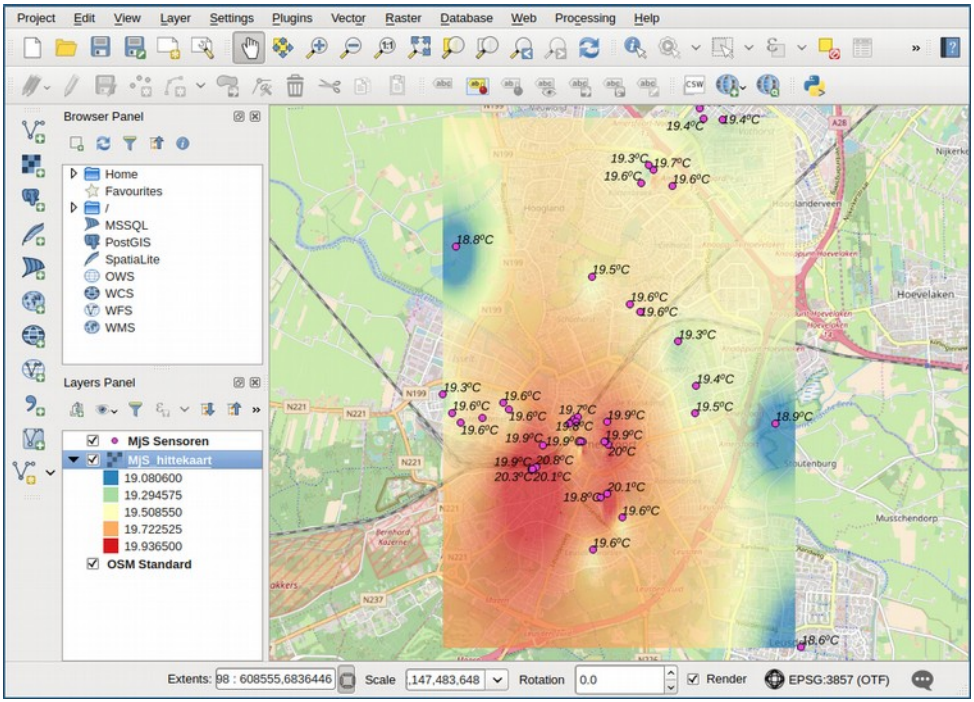
Maak hittekaart

1. Ga naar 'Raster > Interpolation > Interpolation'
2. Bij 'Input \ Vector layers': kies 'MjS Sensoren'
3. Bij 'Input \ Interpolation attribute': kies ' temperature'
4. Klik 'Add'
5. Bij 'Output \ Interpolation method': kies ' Inverse Distance Weighting'
6. Bij 'Output \ Output file': kies een bestandsnaam, bijvoorbeeld 'MjS_hittekaart.tif', en een locatie door op '...' te klikken
7. Klik OK



Maak kleurgradiënt in hittekaart

1. Klik rechts op 'MjS_hittekaart' in het layers panel, kies 'Properties' en selecteer het tabblad 'Style'
2. Bij 'Band rendering \ Render type': kies ' Singeband pseudocolor'
3. Bij 'Band rendering \ Generate new color map': kies 'Spectral' en vink 'invert' aan
4. Klik op 'Classify' en dan op 'Apply'
5. Selecteer het tabblad 'Transparency'
6. Bij 'Global Transparency': zet de fader op 30% en klik OK
7. Zorg dat de hittekaart tussen de achtergrondlaag en de datakaart ligt



Dit waren de voorbeelden... het eigenlijke werk begint nu!

Je hebt in deze workshop kennis gemaakt met een aantal manieren om meetgegevens om te zetten in meer betekenisvolle plaatjes die inzicht geven in wat we nu eigenlijk hebben gemeten.

Welke ideeën heb je gekregen om nog verder uit te proberen?

Welke vragen zou je graag onderzoeken?

Kun je (ongeveer) bedenken hoe je dat aan zou pakken?

Einde

Heb je nog vragen, of vindt je het leuk om een bijdrage te leveren aan het project? Stuur dan een mailtje naar meetgroep@meetjestad.net, of komt eens langs in onze Riot chatroom: https://riot.im/app/#/room/#meetjestad:matrix.org.