sf

SoDaTeam

2/15/2021

Simple Features

> *"A simple feature is defined by the OpenGIS Abstract specification to have both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on 2D geometry with linear interpolation between vertices."*

- A **simple feature** is a standard representation of physical objects that has spatial and non-spatial attributes
- **17** simple feature types (like *point*, *line*, *polygon*)
- Supported by many databases and software

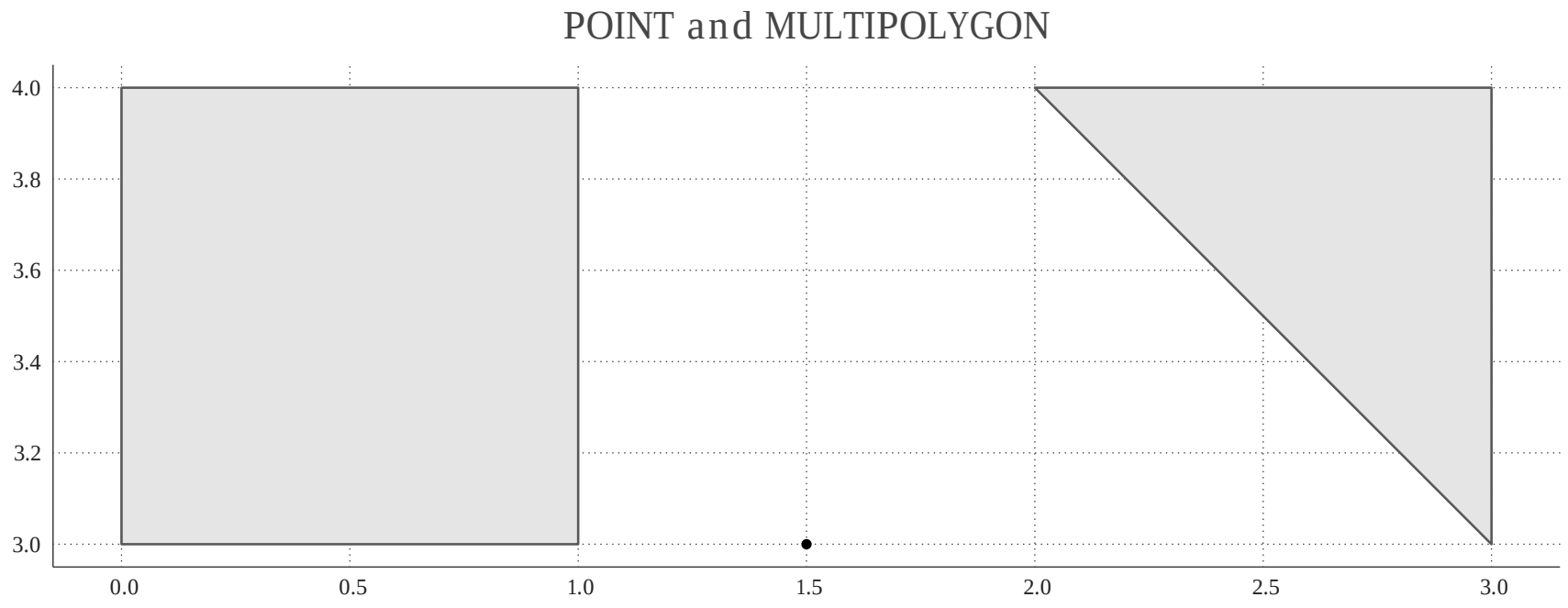
Simple Features in R (sf)

- In R simple features are implemented in the package `sf`
- Provides simple features in `data.frames` or `tibbles` with a geometry list-column
- `sf` supports all simple feature types

```
library(sf)
```

```
## Linking to GEOS 3.7.2, GDAL 2.4.2, PROJ 5.2.0
```

Simple Features: POINT and MULTIPOLYGON



Simple Features and data.frames

- The common use case is working with datasets consisting of sets of features with attributes

```
nc <- st_read(system.file("shape/nc.shp", package="sf"))
```

```
## Reading layer `nc' from data source `/Library/Frameworks/R.framework
## Simple feature collection with 100 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax:
## CRS:            4267
```

Simple Features and data.frames

```
head(nc[,c("CNTY_ID", "NAME")])
```

```
## Simple feature collection with 6 features and 2 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -81.74107 ymin: 36.07282 xmax: -75.77316 ymax:
## CRS:            4267
##   CNTY_ID        NAME                           geometry
## 1    1825        Ashe MULTIPOLYGON (((-81.47276 3...
## 2    1827   Alleghany MULTIPOLYGON (((-81.23989 3...
## 3    1828       Surry MULTIPOLYGON (((-80.45634 3...
## 4    1831   Currituck MULTIPOLYGON (((-76.00897 3...
## 5    1832 Northampton MULTIPOLYGON (((-77.21767 3...
## 6    1833    Hertford MULTIPOLYGON (((-76.74506 3...
```

Simple Features and `data.frames`

```
## Simple feature collection with 100 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## precision:      double (default; no precision model)
## First 3 features:
##   BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                              geom
## 1  1091     1      10  1364     0      19 MULTIPOLYGON(((-81.47275543...
## 2   487     0      10   542     3      12 MULTIPOLYGON(((-81.23989105...
## 3  3188     5     208  3616     6     260 MULTIPOLYGON(((-80.45634460...
```

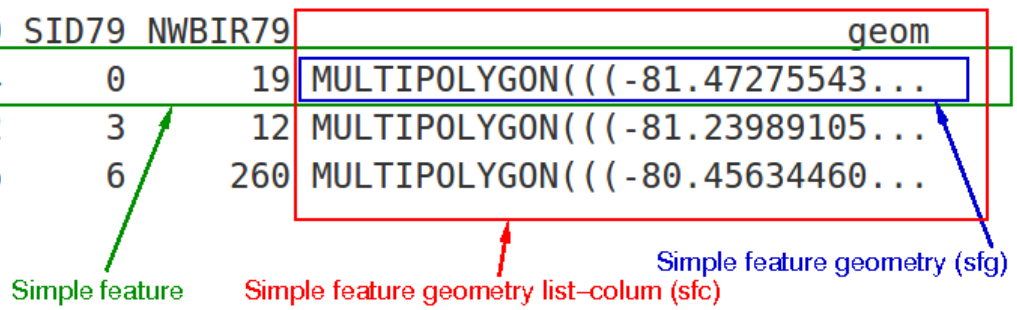


Figure 1: Source: Illustration from Edzer Pebesma (https://r-spatial.github.io/).

sf and OpenStreetMap (OSM) data

- Common use case is working with spatial or geographic data
- The osmdata R package helps with extracting data from OSM
- Potential use case of sf to extract and plot trees and shops within Utrecht

```
library("sf")
library("osmdata")
library("dplyr")
```

Example - Extract Utrecht Geometry

- Extract Utrecht boundaries (`bbox`)

```
utrecht_sf <- osmdata::opq(bbox = 'utrecht nl')

utrecht_sf$bbox
```

```
## [1] "52.026282,5.0041822,52.1356715,5.195155"
```

Example - Extract Features

- With the boundary of Utrecht it is possible to add OSM features
- **Shops**

```
shops <- utrecht_sf %>%
  osmdata::add_osm_feature(key = "shop", value = NULL) %>%
  osmdata::osmdata_sf()
```

- **Trees**

```
trees <- utrecht_sf %>%
  osmdata::add_osm_feature(key = "natural", value = "tree")
  osmdata::osmdata_sf()
```

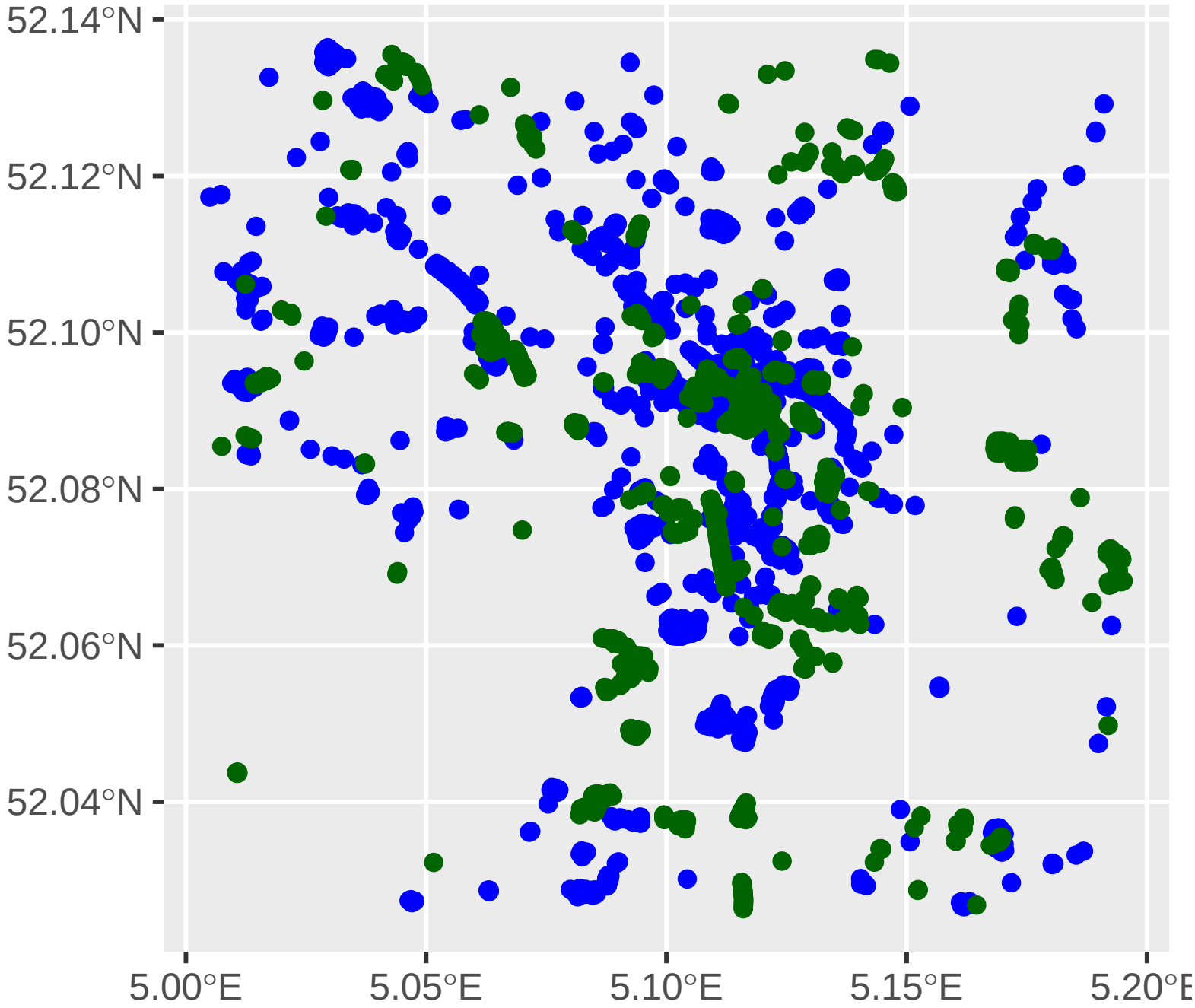
Example - Plot Features

- Use ggplot to plot the points downloaded from OSM

```
library(ggplot2)

ggplot() +
  geom_sf(
    data=shops$osm_points,
    fill="blue",
    color="blue"
    ) +
  geom_sf(
    data=trees$osm_points,
    fill="darkgreen",
    color="darkgreen"
    )
```

Example - Plot Features



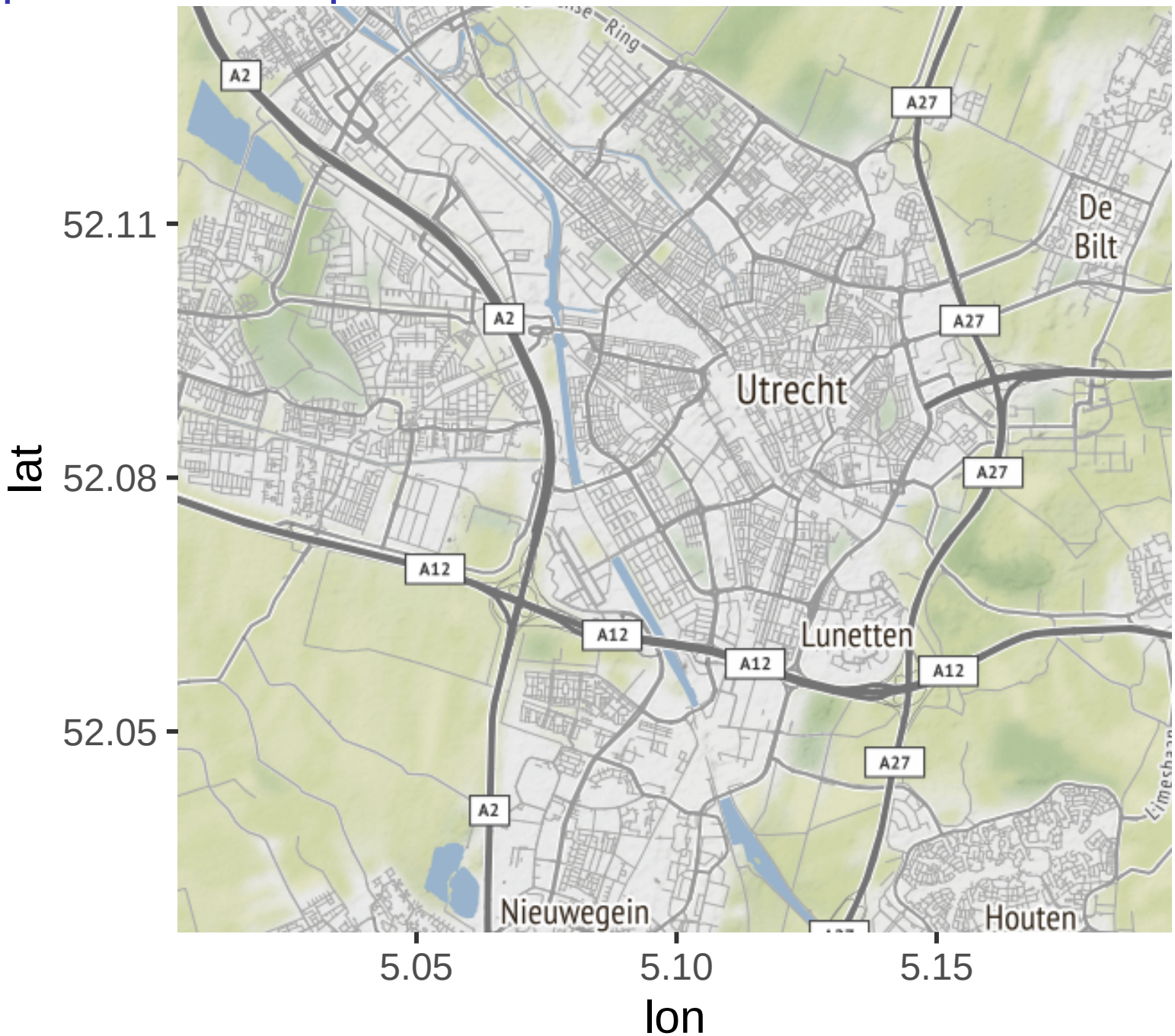
Example - Add Map of Utrecht

- A map of Utrecht improves the visualization of the position of these points

```
library(ggmap)

utrecht_map <- ggmap::get_map(
  c(bottom = 52.026282,
    left = 5.0041822,
    top = 52.1356715,
    right = 5.195155), maptype = "toner-background")
```

Example - Add Map of Utrecht

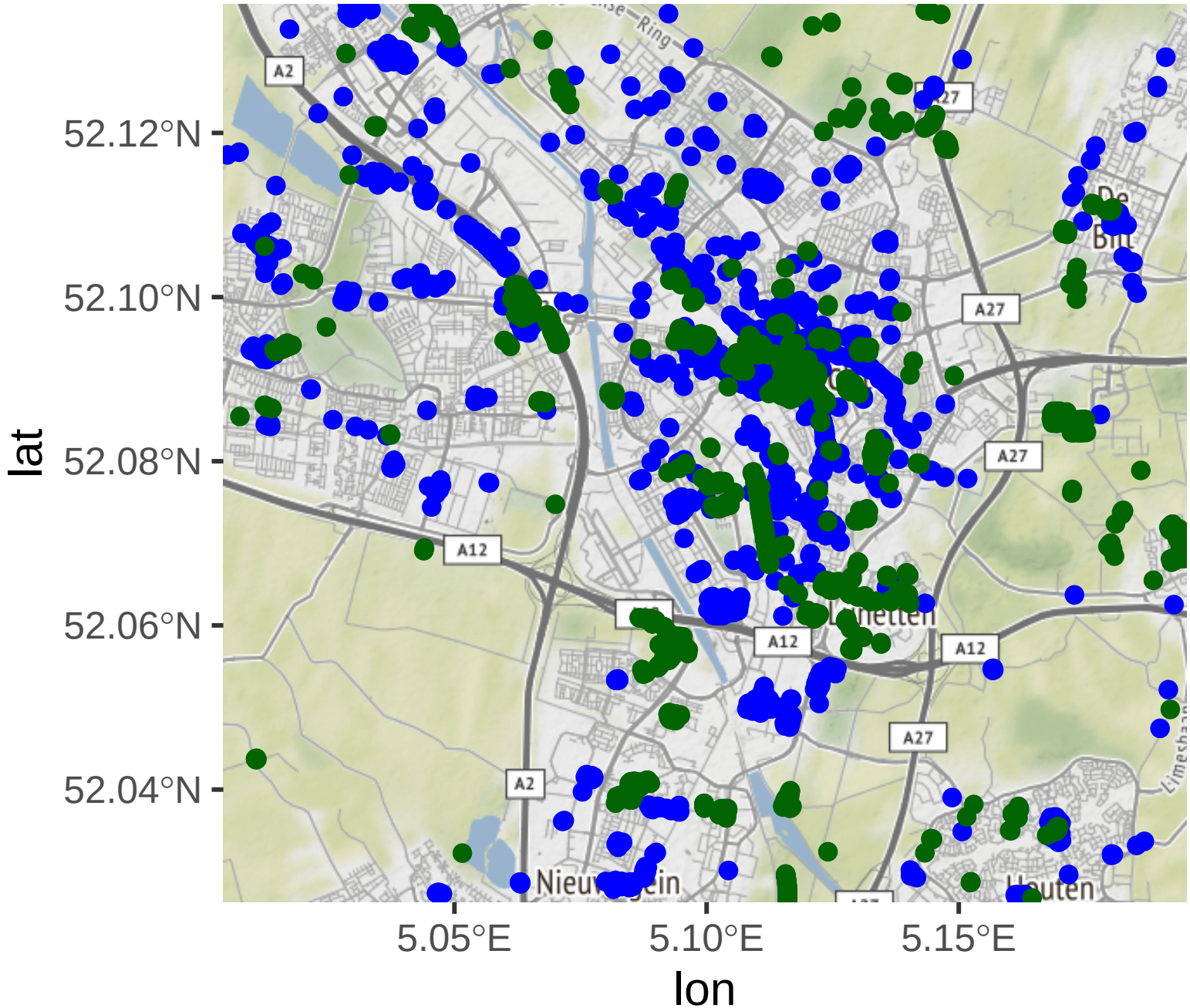


Example - Combine Map and Features

- Combining the map and the points to show the final result

```
ggmap(utrecht_map) +
  geom_sf(
    data=shops$osm_points,
    inherit.aes =FALSE,
    fill="blue",
    color="blue"
    ) +
  geom_sf(
    data=trees$osm_points,
    inherit.aes =FALSE,
    fill="darkgreen",
    color="darkgreen"
    )
```

Example - Combine Map and Features



What is missing?

- Complex workflow
- Retrieving data points in data.frame
- Distances from original points to features
- Large queries
- ...