B7 Simple Features

Jan-Philipp Kolb

23 Oktober 2018

Themen dieses Abschnitts

- Der Import von Geodaten mit dem Paket simple features (`sf`).
- Die Verarbeitung der OSM-Daten mit dem Paket `sf`.
- Die Daten visualisieren mit `sf`

Das Paket `sf`

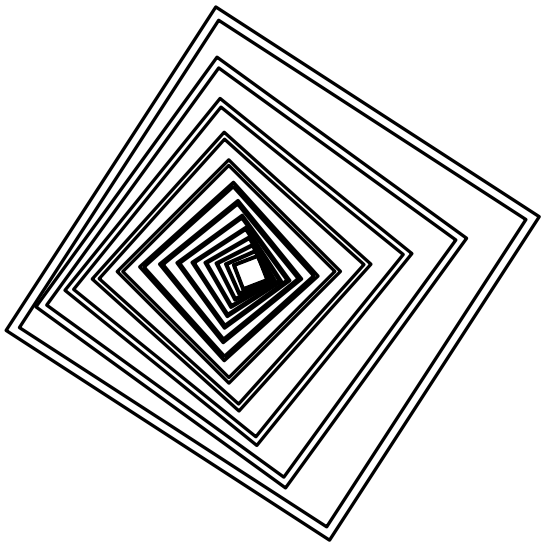
Simple Features for R

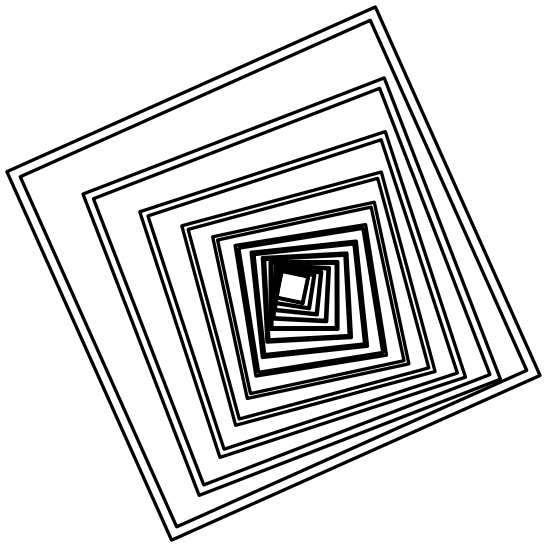
```
library(sf)
```

```
## Linking to GEOS 3.6.1, GDAL 2.2.3, proj.4 4.9.3
```

- Ein Demo ist im Paket `sf` integriert

```
demo(sf::affine)
```





Shapefiles mit `sf` importieren

```
st_layers("../data/london_sport.shp")
```

```
## Driver: ESRI Shapefile
## Available layers:
##     layer_name geometry_type features fields
## 1 london_sport       Polygon       33      4
```

```
london <- st_read("../data/london_sport.shp")
```

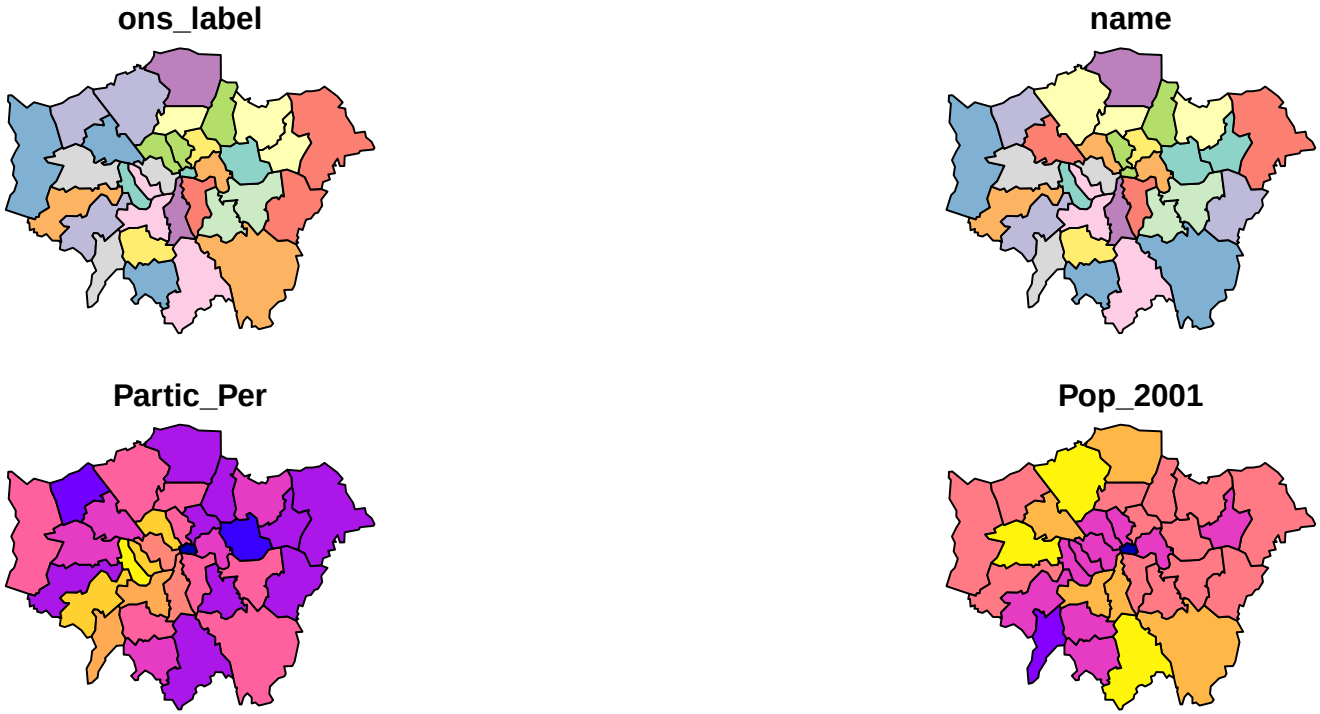
```
## Reading layer `london_sport' from data source `D:\github\ge
## Simple feature collection with 33 features and 4 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 503571.2 ymin: 155850.8 xmax: 561941.
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.999601
```

Das Shapefile plotten

```
plot(london$geometry)
```

Graphiken mit `sf`

```
plot(london)
```



Die Hilfe für die Funktion plot im `sf` Paket

```
?plot
```

tigris exported operators
(in package tigris in library D:/Eigene Dateien/Dokumente/R/win-library/3.5)
Generic X-Y Plotting
(in package graphics in library C:/Program Files/R/R-3.5.0/library)
Plot a Satellite object
(in package satellite in library D:/Eigene Dateien/Dokumente/R/win-library/3.5)
Plot a Raster* object
(in package raster in library D:/Eigene Dateien/Dokumente/R/win-library/3.5)
acs Methods for Function 'plot'
(in package acs in library D:/Eigene Dateien/Dokumente/R/win-library/3.5)
Plot sf object
(in package sf in library D:/Eigene Dateien/Dokumente/R/win-library/3.5)

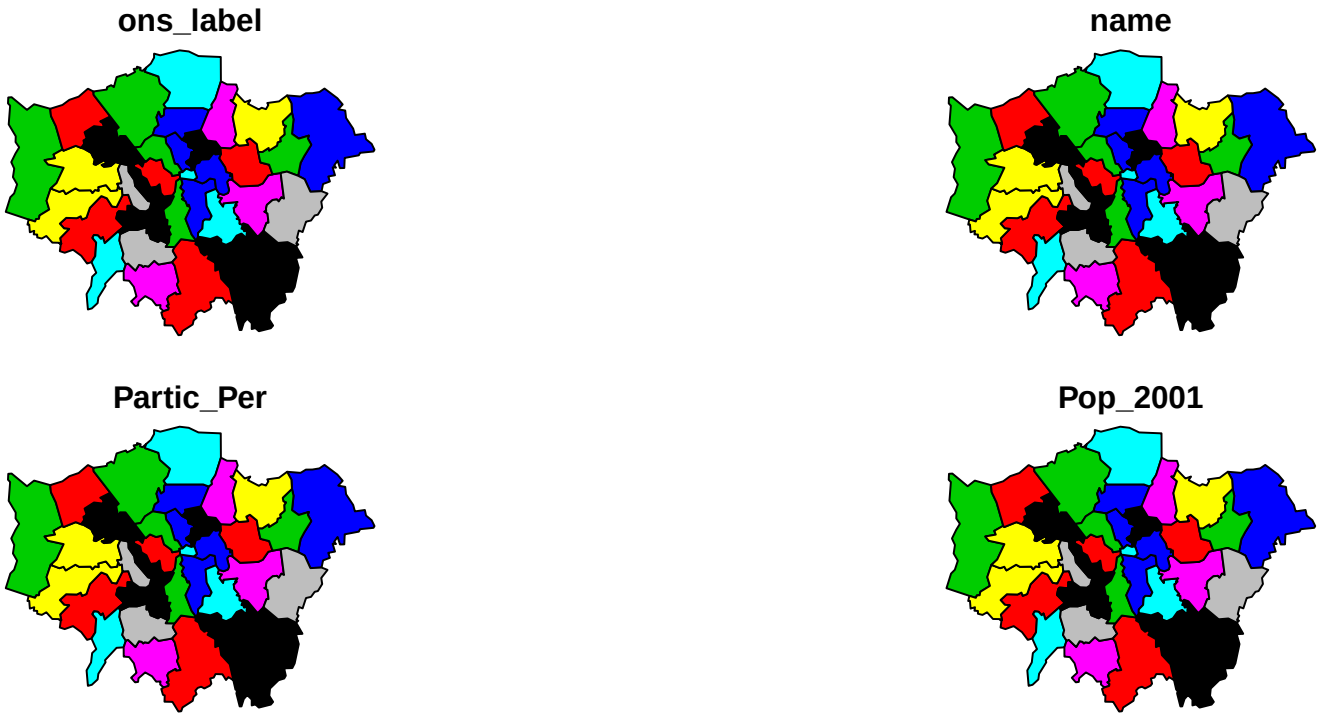
Der `london` shapefile als Beispiel

```
head(london)
```

```
## Simple feature collection with 6 features and 4 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 503571.2 ymin: 156480.8 xmax: 561941.
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.999601
##   ons_label                    name Partic_Per Pop_2001
## 1      00AF                 Bromley       21.7   295535
## 2      00BD Richmond upon Thames       26.6   172330
## 3      00AS              Hillingdon       21.5   243006
## 4      00AR                Havering       17.9   224262
## 5      00AX Kingston upon Thames       24.4   147271
## 6      00BF                  Sutton       19.3   179767
##                       geometry
## 1 POLYGON ((541177.7 173555.7
```

Die Farben verändern

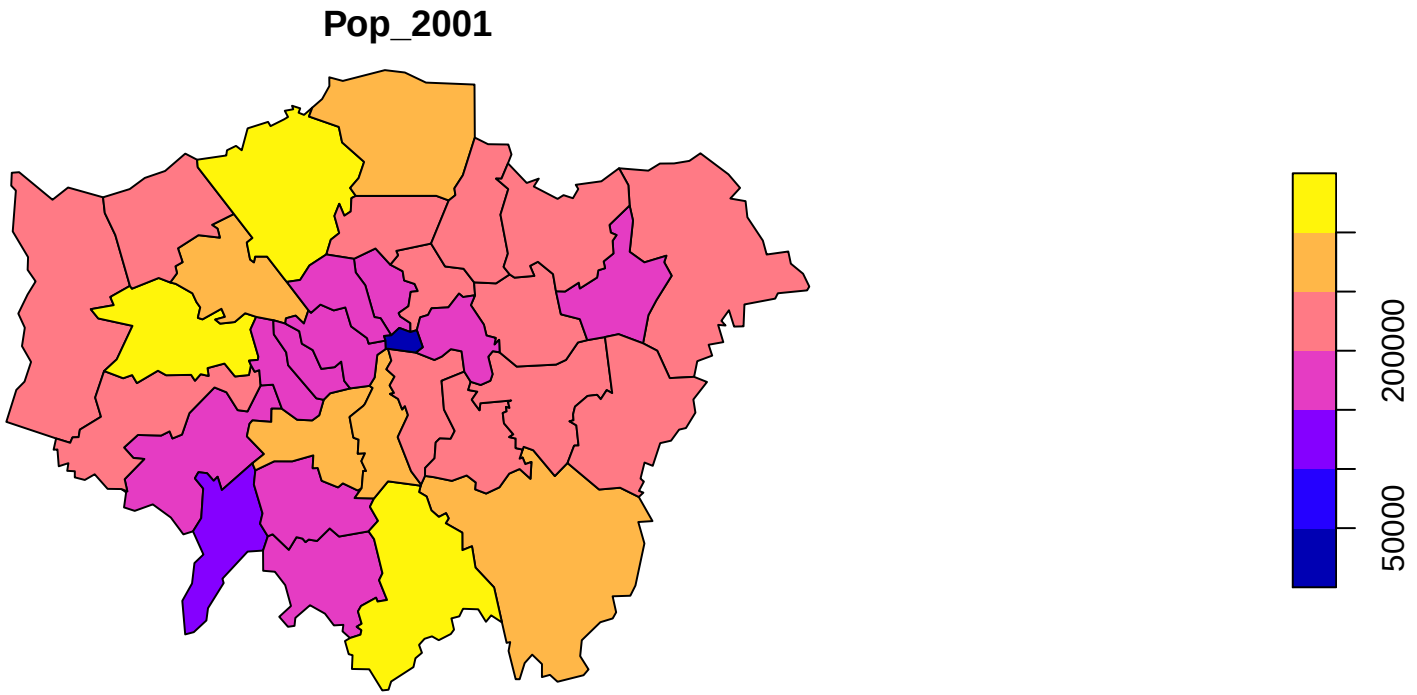
```
plot(london,col=1:20)
```



Nur eine Karte

Beispiel Bevölkerung in London‘s Stadtteilen

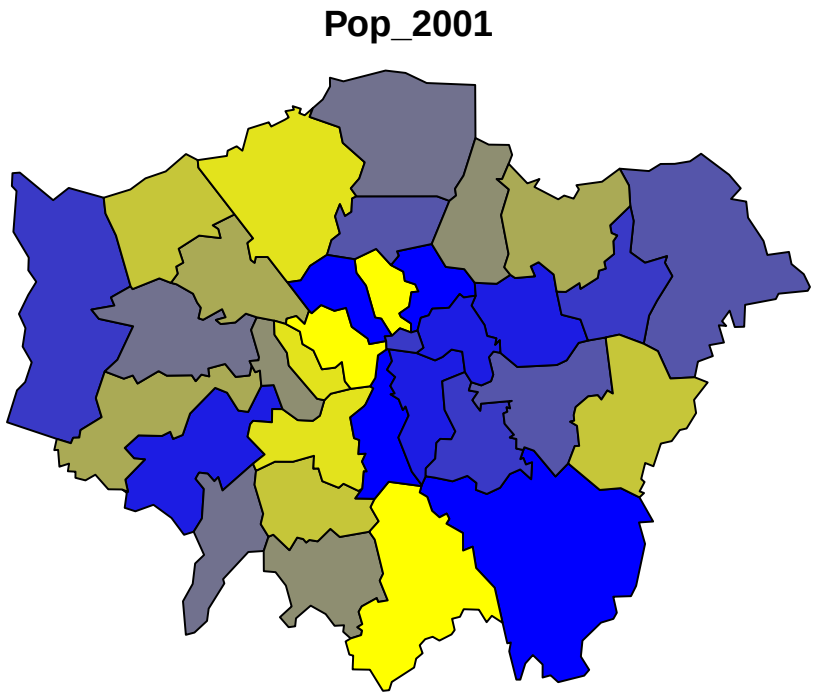
```
london2 <- london[,-(1:3)]
plot(london2)
```



Das Paket colorRampsverwenden

- Cheatsheet zum Thema **hier**

```
library("colorRamps")
plot(london2,col=blue2yellow(10))
```



Beispieldaten bekommen

```
library(osmdata)
```

```
## Data (c) OpenStreetMap contributors, ODbL 1.0. http://www.o
```

```
bb_poly <- getbb(place_name = "Amsterdam",
                 format_out = "polygon")
```

```
ls <- st_multilinestring(bb_poly)
```

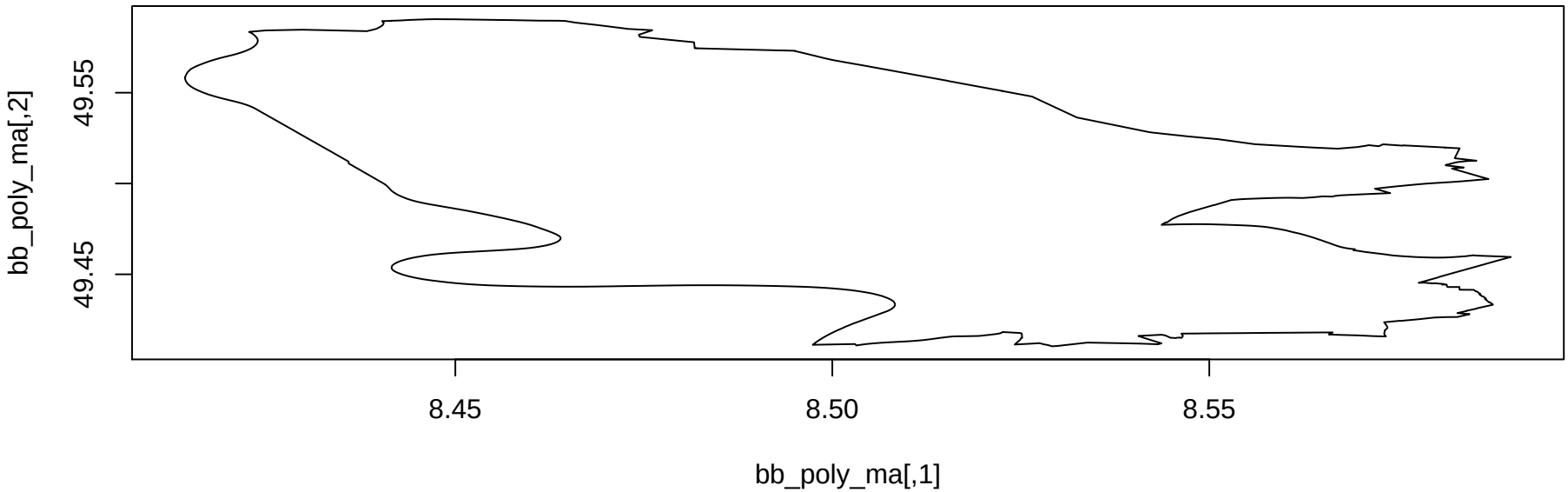
```
pol <- sf::st_polygon(bb_poly)
class(pol)
```

```
## [1] "XY"      "POLYGON" "sfg"
```

```
bb_poly_ma<-getbb(place_name="Mannheim",format_out="polygon")
```

Das Ergebnis plotten

```
plot(bb_poly_ma,type="l")
```



Eine .osm Datei importieren

- In einer .osm Datei sind verschiedene Layer vorhanden.
- Mit st_layers kann man sich anzeigen lassen, welche das sind.

```
st_layers("../data/ams_centraal.osm")
```

```
## Driver: OSM
## Available layers:
##         layer_name       geometry_type features fields
## 1           points               Point       NA     10
## 2            lines         Line String       NA      9
## 3 multilinestrings   Multi Line String       NA      4
## 4    multipolygons       Multi Polygon       NA     25
## 5  other_relations Geometry Collection       NA      4
```

Daten vom Amsterdam Beispiel

- Mit der Funktion `st_read` kann der gewünschte Layer importiert werden.

```
datm <- st_read("../data/ams_centraal.osm","multipolygons")
```

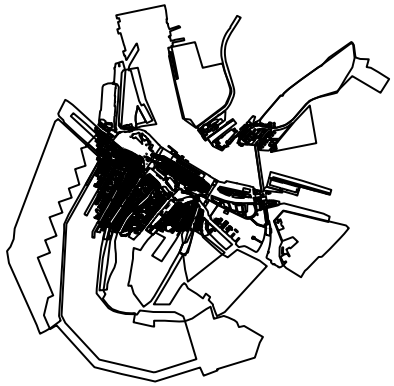
```
## Reading layer `multipolygons' from data source `D:\github\g
## Simple feature collection with 2796 features and 25 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 4.874776 ymin: 52.36088 xmax: 4.92975
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

Die Funktion st_geometry

Get, set, or replace geometry from an sf object

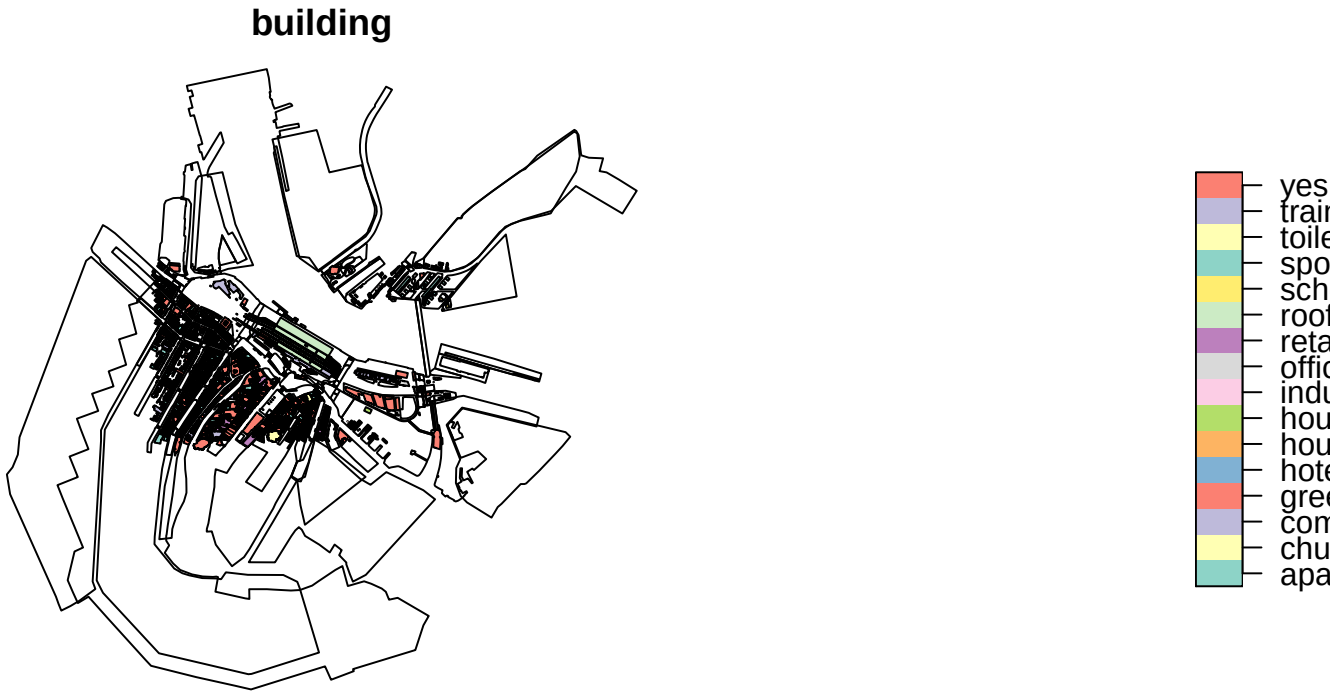
```
?st_geometry
```

```
geom_datm <- st_geometry(datm)
plot(geom_datm)
```



Die Häuser auswählen

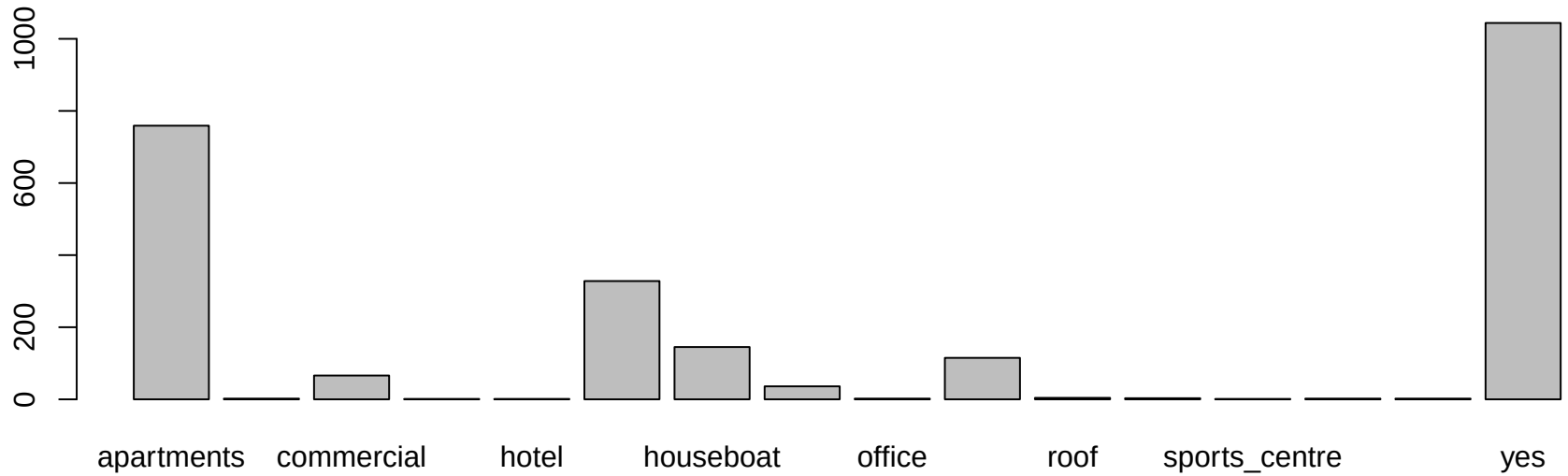
```
library(dplyr)
buis <- datm %>% select(building)
plot(buis)
```



Welche Häusertypen gibt es?

```
buis2 <- datm %>% as.data.frame %>% select(building)
```

```
datbuis <- datm[, "building", drop = TRUE]
plot(datbuis)
```



Alle Häuser herausnehmen

```
houses <- datm[datm$building %in% c("house","yes",
                                    "apartments"),]
```

- Im ersten Teil des Objekts sind allgemeine Informationen zum Geometrietyp, zur Bounding Box und zum EPSG Code enthalten.

```
Simple feature collection with 2131 features and 25 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: 4.887275 ymin: 52.37334 xmax: 4.91342 ym
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

Zweiter Teil des Objekts houses

- Im zweiten Teil sind dann spezifische Informationen zu den einzelnen Features aufgelistet.
- Es handelt sich beispielsweise um die OSM id und in der letzten Spalte die Geometrie, die wir später zum visualisieren brauchen.

	osm_id	osm_way_id	name	type	building	craft
5	3580102	<NA>	<NA>	multipolygon	apartments	<NA>
6	3580414	<NA>	<NA>	multipolygon	yes	<NA>
7	3580416	<NA>	<NA>	multipolygon	apartments	<NA>
8	3580417	<NA>	<NA>	multipolygon	apartments	<NA>
9	3580420	<NA>	<NA>	multipolygon	apartments	<NA>
10	3580421	<NA>	<NA>	multipolygon	apartments	<NA>
11	3580422	<NA>	<NA>	multipolygon	apartments	<NA>
12	3580423	<NA>	<NA>	multipolygon	apartments	<NA>
13	3580427	<NA>	<NA>	multipolygon	apartments	<NA>
14	3580428	<NA>	<NA>	multipolygon	house	<NA>

Das Objekt houses transformieren

```
class(houses)
```

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

```
class(st_geometry(houses))
```

```
## [1] "sfc_MULTIPOLYGON" "sfc"
```

Das Ergebnis visualisieren

```
library(tmap)
(map1 <- qtm(st_geometry(houses)))
```

Wohnstraßen hinzufügen

```
datl <- st_read("../data/ams_centraal.osm","lines")
```

```
## Reading layer `lines' from data source `D:\github\geocourse
## Simple feature collection with 2372 features and 9 fields
## geometry type:  LINESTRING
## dimension:      XY
## bbox:           xmin: 4.826049 ymin: 52.33891 xmax: 4.95717
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
```

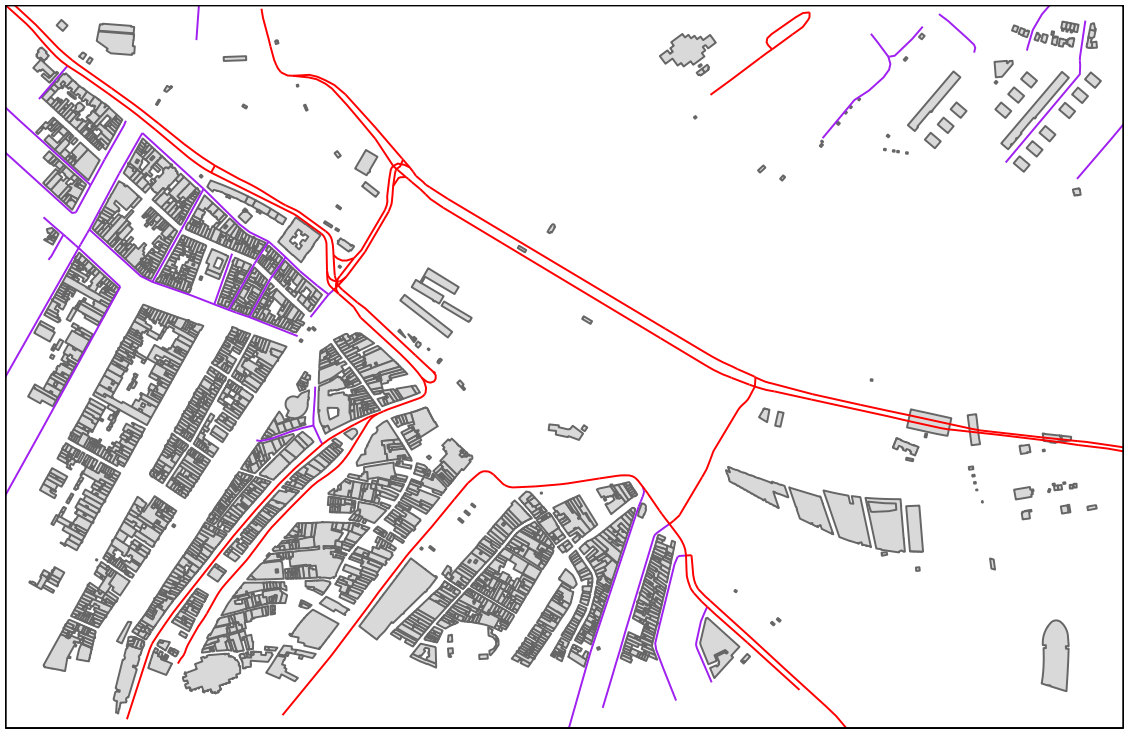
```
roads <- datl[datl$highway %in% c("residential"),]
```

Der Straßen-Typ residential

```
(map2 <- map1+qtm(st_geometry(roads),lines.col="purple"))
```

Weitere Straßen hinzufügen

```
roads2 <- datl[datl$highway %in% c("tertiary","secondary",
                                   "primary"),]
(map3 <- map2+qtm(st_geometry(roads2),lines.col="red"))
```



Eine Demonstartion von sf

Beispieldatensatz `nc`

```
demo(nc, ask = FALSE, echo = FALSE)
```

```
## Reading layer `nc.gpkg' from data source `D:\Eigene Dateien
## Simple feature collection with 100 features and 14 fields
## Attribute-geometry relationship: 0 constant, 8 aggregate, 6
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
```

Die Vignetten für das Paket `sf`

https://r-spatial.github.io/sf/reference/st_as_sf.html

https://r-spatial.github.io/sf/reference/st_read.html

https://r-spatial.github.io/sf/articles/sf1.html