B2 Das `osmdata` Paket

Jan-Philipp Kolb

22 Oktober 2018

Themen dieses Abschnitts

- Die Overpass API

- Downlaod von OSM-Daten mit dem Paket `osmdata`, das auf der **Overpass API** beruht.
- Verarbeitung der OSM-Daten mit dem Paket `sf`
- Das Plotten der Ergebnisse

Overpass Turbo

Ausführen | Teilen | Export | Wizard | Speichern | Laden | Einstellungen | Hilfe

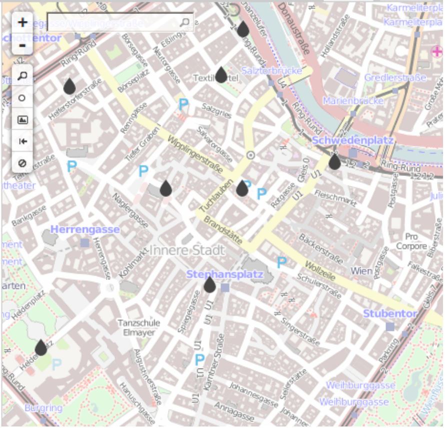
overpass turbo

Flattr this!

Karte | Daten

```
<!--
This has been generated by the overpass-turbo wizard.
The original search was:
“Trinkbrunnen”
-->
<osm-script output="json" timeout="25">
  <!-- gather results -->
  <union>
    <!-- query part for: “Trinkbrunnen” -->
    <query type="node">
      <has-kv k="amenity" v="drinking_water"/>
      <bbox-query {{bbox}}/>
    </query>
  </union>
  <!-- print results -->
  <print mode="body"/>
  <recurse type="down"/>
  <print mode="skeleton" order="quadtile"/>
</osm-script>

{{style:
node[amenity=drinking_water] {
  icon-image: url("icons/maki/water-24.png");
  icon-width: 24;
}
}}
```



Query Overpass

```
node
  [amenity=bar]
  ({{bbox}});
out;
```

Das `osmdata` Paket

Mark Padgham - Import 'OpenStreetMap' Data as Simple Features or Spatial Objects

Das osmdata Paket

- Mit dem Paket kann man Daten von OpenStreetMap importieren
- Die OSM Daten sind unter **ODbL licence** zu haben

```
install.packages("osmdata")
```

```
library(osmdata)
```

```
citation("osmdata")
```

Einen Rahmen definieren um Daten zu bekommen

- Der Rahmen kann entweder erstellt werden, indem die Koordinaten angegeben werden:

```
q <- opq(bbox = c(52.3, 4.7, 52.4, 5.1))
```

- oder indem man den Befehl `getbb` verwendet:

```
bb <- getbb('Ladenburg')
```

- In `bb` sind nun vier Werte gespeichert, die den Rahmen definieren
- Befehl `opq` - eine Overpass Anfrage erstellen

```
q <- opq(bbox = bb)
```

Die Grenze von Mannheim

- Erst mit dem Argument `format_out=polygon` Befehl `getbb` erhält man das Polygon:

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

- Das Ergebnis ist sind zwei Vektoren mit den Longitude und Latitude Koordinaten.

```
##           [,1]     [,2]
## [1,] 8.569720 49.49107
## [2,] 8.569858 49.49101
## [3,] 8.569999 49.49096
## [4,] 8.570342 49.49085
```

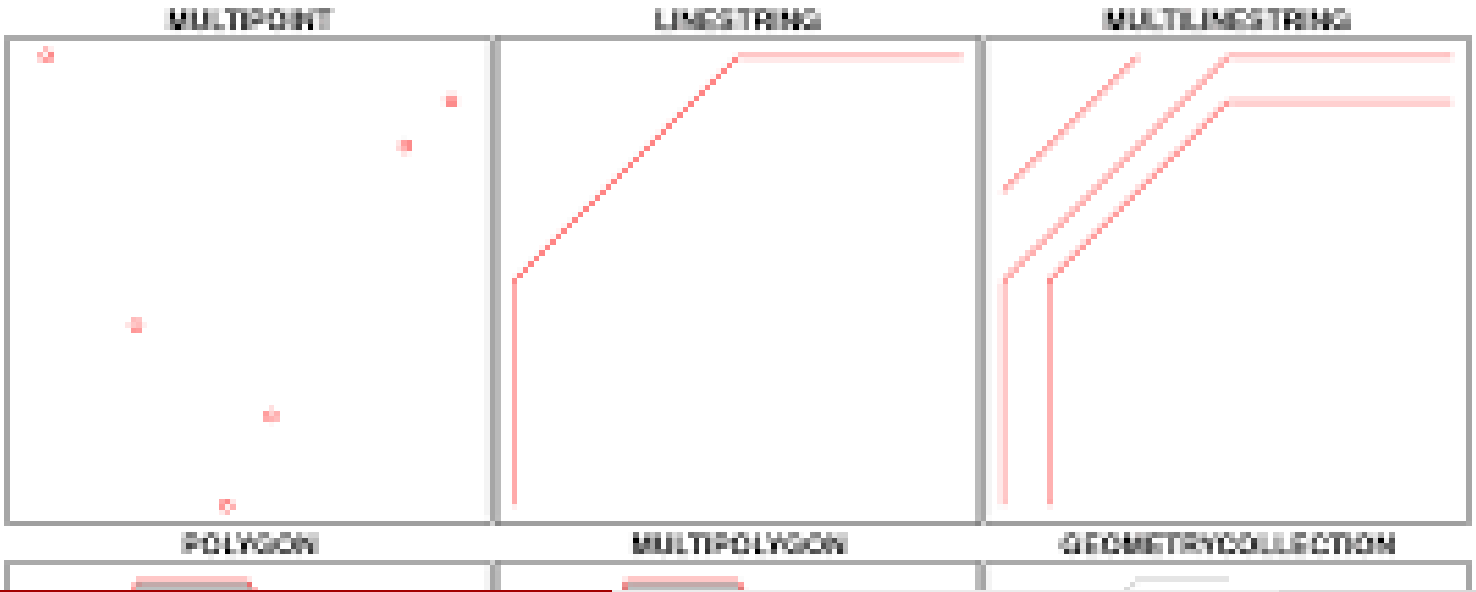
Das Paket für simple feature (`sf`)

Simple Features for R

- Das Paket `sf` ist ein Paket um geometrische Operationen durchzuführen.

```
library(sf)
```

- **Vignette für das Paket `sf`**



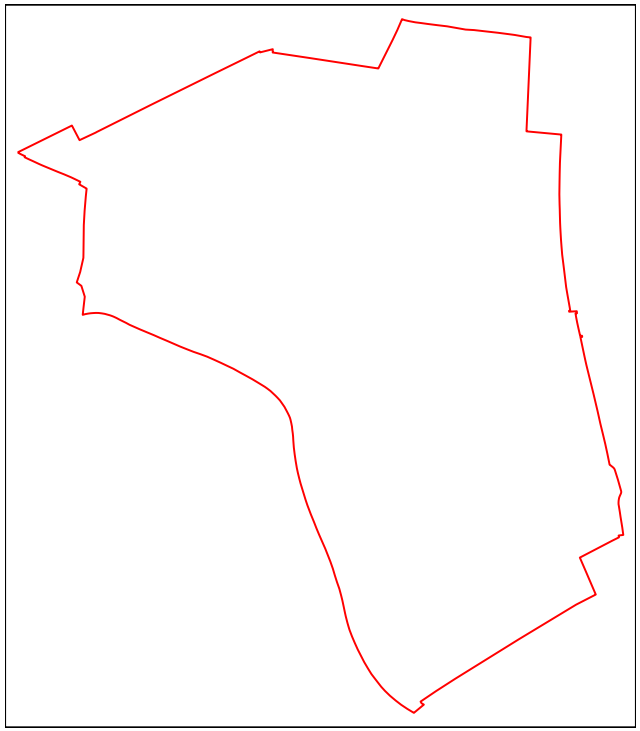
Die Funktion `st_linestring`

Create simple feature from a numeric vector, matrix or list

```
library(sf)
ls <- st_linestring(bb_poly)
sfc <- st_sfc(ls)
```

Den `linestring` plotten

```
library(tmap)
qtm(sfc)
```



Einrichtungen (amenity)

OSM map features

- Alle benannten Objekte findet man, wenn man OSM map features in eine Suchmaschine eingibt.
- Achtung, wenn man bspw. alle Objekte mit dem Schlüssel `amenity` für eine Stadt heraussucht, bekommt man einen recht großen Datensatz

```
q <- add_osm_feature (q, key = 'amenity')
osmdata_xml(q, '../data/Ladenburg_amenity.osm')
```

Was dahinter steckt

Die Funktion osmdata_sf

- Die Funktion osmdata_sf gibt ein osmdata ObjeKt im sf Format.

```
library(magrittr)
dat1 <- opq(bbox = 'Ladenburg') %>%
    add_osm_feature(key = 'shop', value = 'bakery') %>%
    osmdata_sf ()
```

```
unlist(lapply(dat1,nrow))
```

```
##          osm_points           osm_lines       osm_polygons     os
##                  16                   0                   0
## osm_multipolygons
##                   0
```

Alles in eine Karte plotten

**Der Start mit dem Paket `tmap`

```
library(tmap)
tm_shape(sfc)
tm_bubbles(dat, size=2)
```

Beispiel Fahrradparkplätze

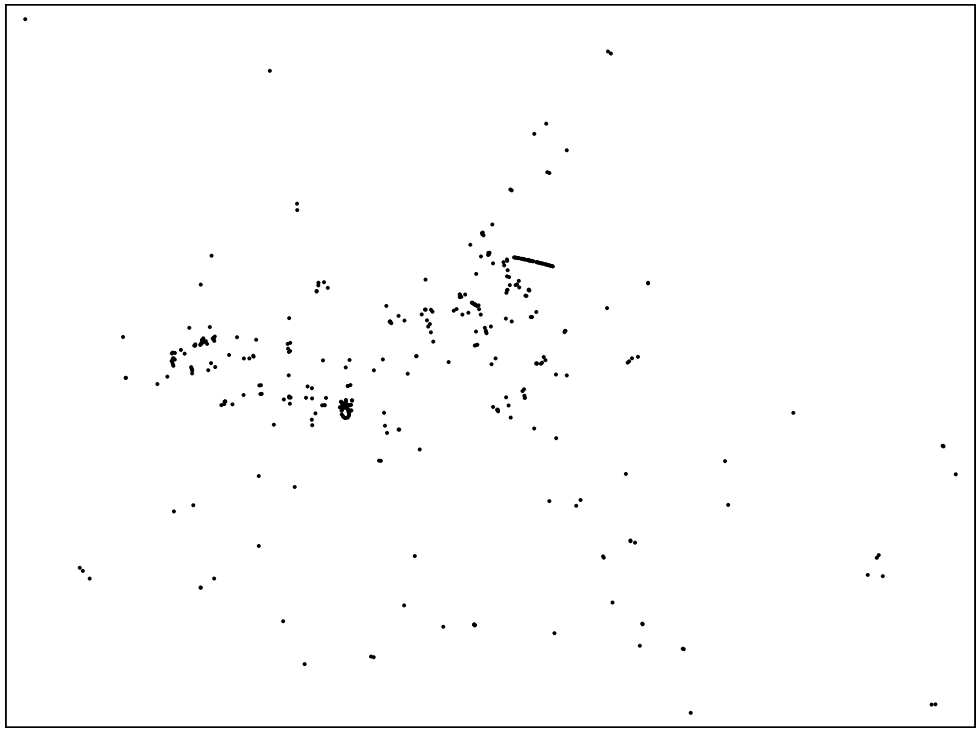
- **OSM map features**

```
q <- add_osm_feature (q, key = 'amenity',value = "bicycle_park
osmdata_xml(q, '../data/Amsterdam_amenity_bicycle_parking.osm'
```

```
dat <- sf::st_read ('../data/Amsterdam_amenity_bicycle_parking
                    layer = 'points',
                    quiet = TRUE)
```

Die Daten plotten

```
library(tmap)
qtm(dat)
```



Sehen was dahinter steckt

```
dat <- sf::st_read ('../data/Amsterdam_amenity.osm',
                    layer = 'points',
                    quiet = TRUE)
```

```
nrow(dat)
names(dat)
```

Bar's in Mannheim

```
?add_osm_feature
```

```
q <- opq (bbox = 'Mannheim')
q <- add_osm_feature (q, key ="amenity",value = 'bar')
osmdata_xml (q, 'data/Mannheim_bar.osm')
```

```
dat_bar <- sf::st_read ('../data/Mannheim_bar.osm',
                        layer = 'points', quiet = TRUE)
```

Bus stations in Amsterdam

```
q <- opq (bbox = 'Amsterdam')
q <- add_osm_feature (q, key ="amenity",
                         value = 'bus_station')
osmdata_xml (q, 'data/Amsterdam_bus_station.osm')
```

```
dat_bus <- sf::st_read ('../data/Amsterdam_bus_station.osm',
                        layer = 'points', quiet = TRUE)
nrow(dat_bus)
```

```
?sf::st_read
```

An alternative

- Main vignette osmdata
- OpenStreetMap Data Structure

```
q <- opq (bbox = 'Amsterdam')
q <- add_osm_feature (q, key ="public_transport",
                         value = 'station')
osmdata_xml (q, '../data/Amsterdam_bus_pubtrans.osm')
```

```
dat_bus <- sf::st_read ('../data/Amsterdam_bus_pubtrans.osm',
                        layer = 'points', quiet = TRUE)
nrow(dat_bus)
```

Further information about public transport

Stop area

```
dat3 <- opq(bbox = 'Amsterdam') %>%
    add_osm_feature(key = 'railway',
                    value = 'tram_stop') %>%
    osmdata_sf ()
```

```
dat3$osm_points$geometry
```

Plotting the result

```
# install.packages("osmplotr")
library("osmplotr")
bbox <- getbb("Amsterdam")
dat_pa <- extract_osm_objects(key='highway',
                              value="primary",
                              bbox=bbox)
dat_sa <- extract_osm_objects(key='highway',
                              value="secondary",
                              bbox=bbox)
dat_ta <- extract_osm_objects(key='highway',
                              value="tertiary",
                              bbox=bbox)
```

```
map <- osm_basemap(bbox = bbox, bg = c("#F5F5DC"))
map <- add_osm_objects(map, dat_pa, col = c("#00008B"))
map <- add_osm_objects(map, dat_sa, col = "green")
```

Get an overview of the available features

```
features <- available_features()
head(features,n=20)
```

```
##  [1] "4wd only"                    "abandoned"
##  [3] "abutters"                    "access"
##  [5] "addr"                        "addr:city"
##  [7] "addr:conscriptionnumber"     "addr:country"
##  [9] "addr:district"               "addr:flats"
## [11] "addr:full"                   "addr:hamlet"
## [13] "addr:housename"              "addr:housenumber"
## [15] "addr:inclusion"              "addr:interpolation"
## [17] "addr:place"                  "addr:postcode"
## [19] "addr:province"               "addr:state"
```

Changing the API

```
api_list <- c('http://overpass-api.de/api/interpreter',
              'https://lz4.overpass-api.de/api/interpreter',
              'https://z.overpass-api.de/api/interpreter',
              'https://overpass.kumi.systems/api/interpreter')

api_to_use <- sample(1:length(api_list), 1)

set_overpass_url(api_list[api_to_use])
```

Die wichtigsten Funktionen im Paket `osmdata`

```
# https://rdrr.io/cran/osmdata/man/osmdata_sp.html
?osmdata_sp
```

Links

- Github repo of the `osmdata` package
- Vignette for the package `osmdata` on github
- osmdata Homepage
- Overpass API - query form
- Overpass API/Language Guide
- Overpass Turbo
- **`osmplotr` tutorial
- **Geocomputation with R**
- **osmar - JOS**

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