

B4 Overpass

Jan-Philipp Kolb

23 Oktober 2018

Themen dieses Abschnitts

- Die **Overpass API** von Roland Olbricht wird vorgestellt.
- Die API **Overpass Turbo**
- Wie man die OSM Daten graphisch darstellen kann.

Die Overpass API

- Die von Roland Olbricht geschriebene Overpass API ermöglicht es Entwicklern, kleine Auszüge von benutzergenerierten Inhalten von Openstreetmap nach vorgegebenen Kriterien herunterzuladen.
- Overpass ist eine read-only API, die durch den Benutzer ausgewählte Teile der OSM-Daten bereitstellt.
- Overpass kann als eine Datenbank über das Internet verstanden werden.
- Die API eignet sich besonders gut, wenn man nach ganz speziellen Map Features sucht.

Overpass Turbo

AusführenTeilenExportWizardSpeichernLadenEinstellungenHilfeoverpass turboKarteDaten

```
1  /*
2  This is an example Overpass query.
3  Try it out by pressing the Run button above!
4  You can find more examples with the Load
5  tool.
6  */
7  node
8    [highway=bus_stop]
9    ({{bbox}});
10 out;
```

geladen – Nodes: 39, Ways: 0, Relations: 0
angezeigt – POIs: 39, Linien: 0, Polygone: 0


Figure 1: <https://overpass-turbo.eu/>

Query Overpass

- In der folgenden Abfrage wird bei Overpass Turbo nach Bars im ausgewählten Fenster gesucht.

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

Export bei Overpass

Exportieren 


▼ Daten


Speichere/Kopiere als GeoJSON


Speichere/Kopiere als GPX

Speichere/Kopiere als KML

Speichere/Kopiere als OSM Rohdaten

Rohdaten direkt von Overpass API 

In einen OSM-Editor laden: JOSM, Level0 

GeoJSON als gist  speichern

► Karte

Bei Export von Overpass

- GeoJSON
- GPX
- KML
- OSM Rohdaten

Import von Daten

```
library(XML)
dat <- xmlParse("../data/bus_stop_amsterdam.kml")
```

```
xmltop <- xmlRoot(dat)
xmltop[[1]][[1]]
```

```
## <name>overpass-turbo.eu export</name>
```


Xpath Abfragesprache

Beispiel: xpath wikipedia

```
xpathApply(dat, "Document")
```

```
## list()
```

```
## attr("class")
```

```
## [1] "XMLNodeSet"
```

JSON importieren

```
install.packages("rjson")  
library(rjson)
```

```
library(jsonlite)  
dat<-jsonlite::fromJSON("../data/amsterdam_busstop.geojson")  
typeof(dat)
```

```
## [1] "list"
```

```
names(dat)
```

```
## [1] "type"          "generator" "copyright" "timestamp" "featur
```

Wie sehen die Daten aus

```
DT::datatable(dat$features$properties)
```

Show 10 ▾ entries

Search:

	@id	highway	name	public_transport	zone	cxx:code	cxx:order
1	node/447840083	bus_stop	Leidseplein	platform	5700		
2	node/495568909	bus_stop	Dam	platform	5700	57002550	3
3	node/502341044	bus_stop	Elandsgracht	stop_position			
4	node/534026003	bus_stop	Centraal Station / Nicolaaskerk		5700		
5	node/700343182	bus_stop	Prins Hendrikkade	platform	5700		
6	node/724232554	bus_stop	Dam	platform	5700	57002560	3
7	node/1079768926	bus_stop	Prins Hendrikkade	platform	5700		
8	node/1079768989	bus_stop	IJ tunnel	platform	5700		

GPX file importieren

```
library(plotKML)
```

```
## plotKML version 0.5-8 (2017-05-12)
```

```
## URL: http://plotkml.r-forge.r-project.org/
```

```
dat_gpx <- readGPX("../data/Amsterdam_busstop.gpx")  
head(dat_gpx$waypoints)
```

```
##           lon      lat           name  
## 1 4.880870 52.36213      Leidseplein  
## 2 4.891237 52.37438          Dam  
## 3 4.877558 52.36953      Elandsgracht  
## 4 4.900331 52.37670 Centraal Station / Nicolaaskerk  
## 5 4.905498 52.37395      Prins Hendrikkade  
## 6 4.890181 52.37310          Dam  
##  
## 1
```

Daten verbinden - Beispiel Bäckereien in Berlin

- Quelle für die folgenden Daten ist:



Geodaten von **OpenStreetMap**

die freie Weltkarte

OpenStreetMap.org

Lizenz



OSM als Datenquelle

- Zum Download habe ich die **Overpass API** verwendet

```
load("../data/info_bar_Berlin.RData")
```

	addr.postcode	addr.street	name	
79675952	13405	Scharnweberstra��e	Albert's	52
86005430	NA	NA	Newton Bar	52
111644760	NA	NA	No Limit Shishabar	52
149607257	NA	NA	en passant	52
248651127	10115	Bergstra��e	Z-Bar	52
267780050	10405	Christburger Stra��e	Immertreu	52

Verwendung des Pakets gosmd

```
devtools::install_github("Japhilko/gosmd")
```

```
library("gosmd")  
pg_MA <- get_osm_nodes(object="leisure=playground", "Mannheim")  
pg_MA <- extract_osm_nodes(pg_MA, value='playground')
```

Matching

```
tab_plz <- table(info_be$addr.postcode)
```

```
load("../data/BE.RData")
```

```
ind <- match(BE@data$PLZ99_N, names(tab_plz))  
ind
```

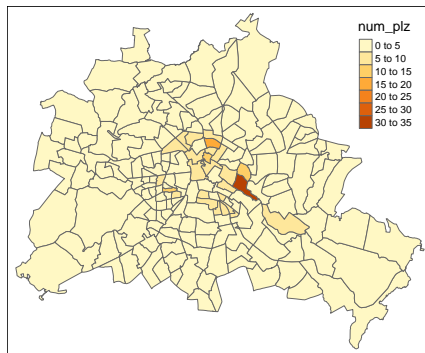
```
##      [1]   1   2   3   4   5   6   7   8 NA   9 NA NA NA NA NA 10 11 12  
##    [24] 17 18 19 20 21 22 23 24 25 NA 26 27 28 29 NA NA NA NA  
##    [47] 34 35 NA NA 36 37 38 39 40 41 42 43 44 45 46 47 48 49  
##    [70] NA 54 55 NA NA NA 56 57 58 59 60 NA NA NA NA NA 61 NA  
##    [93] NA NA NA NA NA NA NA NA 63 NA NA 64 NA 65 NA NA NA 66 NA  
##   [116] NA 68 NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA NA  
##   [139] NA 69 70 NA 71 72 73 74 75 NA 76 NA NA NA NA NA NA NA  
##   [162] 77 NA 78 79 NA NA NA NA 80 NA NA NA NA 81 NA 82 83 84  
##   [185] NA NA NA 85 NA NA
```


Daten anspielen

```
BE@data$num_plz <- tab_plz[ind]
```

Eine Karte von Berlin mit dem Paket tmap

```
BE@data$num_plz[is.na(BE@data$num_plz)] <- 0  
tmap::qtm(BE, fill = "num_plz")
```



Mehr Informationen einbinden

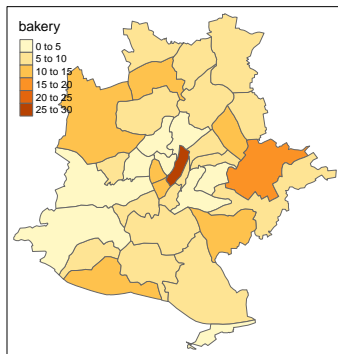
- Der folgende Datensatz ist eine Kombination aus den vorgestellten PLZ-Shapefiles und OSM-Daten die über Overpass heruntergeladen wurden:

```
load("../data/osmsa_PLZ_14.RData")
```

	PLZ99	PLZ99_N	PLZORT99	nname	EWZ_gem	area_d	EWZ_gemplz	place_id	osm_type	osm_id
0	01067	1067	Dresden	Dresden, Stadt	512354	0.000860248222975774	20494.16	144969068	relation	191645
1	01069	1069	Dresden	Dresden, Stadt	512354	0.000681890231341799	20494.16	144969068	relation	191645
2	01097	1097	Dresden	Dresden, Stadt	512354	0.000438158611364389	20494.16	144969068	relation	191645
3	01099	1099	Dresden	Dresden, Stadt	512354	0.00677395240491571	20494.16	144969068	relation	191645
4	01109	1109	Dresden	Dresden, Stadt	512354	0.00349732653109023	20494.16	144969068	relation	191645
5	01127	1127	Dresden	Dresden, Stadt	512354	0.000362617756392607	20494.16	144969068	relation	191645
				Dresden,						

OSM-Daten - Bäckereien in Stuttgart

```
tmap::qtm(PLZ_SG, fill="bakery")
```



DATASCIENCE TOOLKIT

Welcome to the Data Science Toolkit

Truly open tools for data.

Incorporates the city-level TwoFishes geocoder written by David Blackman at Foursquare.

Notice: Recent Outages

DSTK has been receiving a large amount of traffic lately, exceeding the limits of our servers. If you are using DSTK for business-critical applications, we strongly advise you host your own server so that you are not affected by outages.

- Data Science Toolkit API

```
library("RDSTK")
```

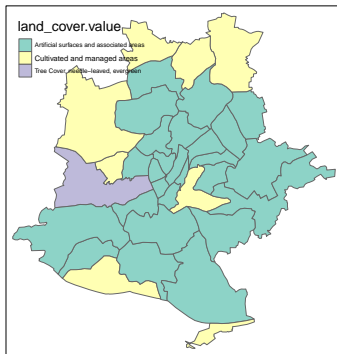
Die Daten für Stuttgart

Type_landcover	Freq
Artificial surfaces and associated areas	26
Cultivated and managed areas	8
Tree Cover, needle-leaved, evergreen	1

Eine Karte der Flächenbedeckung erstellen

- Daten von European Commission Land Resource Management Unit Global Land Cover 2000.

```
tmap::qtm(PLZ_SG, fill="land_cover.value")
```



Die Höhe in Stuttgart

- Daten von **NASA** and the **CGIAR Consortium for Spatial Information**

```
tmap::qtm(PLZ_SG, fill="elevation.value")
```

