Map Types

Jan-Philipp Kolb

Wed Sep 16 08:22:22 2015

Outline

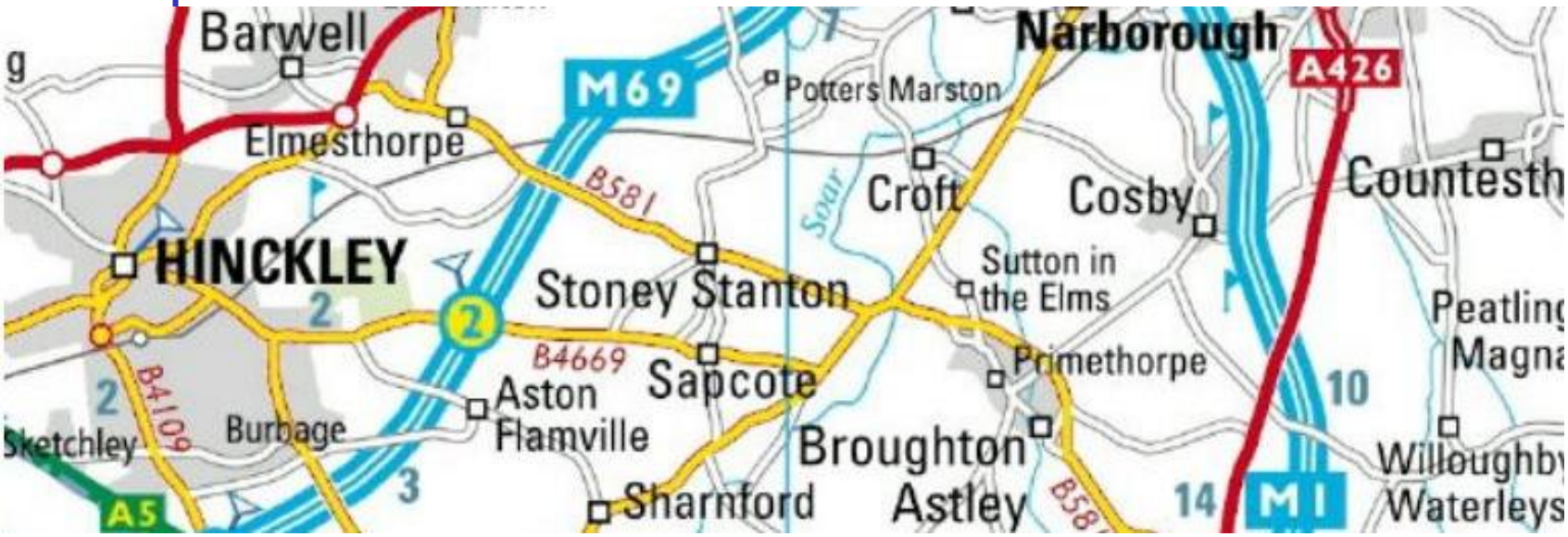
Various map types:

- Road maps
- Satellite maps
- Terrain maps/physical maps
- Abstracted maps

There are more which will be presented next week...

The R-package ggmap will be used in the following to produce different types of maps with the command qmap

Road maps



Source

A road map is one of the most widely used map types.

- These maps show major and minor highways and roads (depending on detail)
- as well as things like airports, city locations and points of interest like parks, campgrounds and monuments.
- Major highways on a road map are generally red and larger than other roads,
- while minor roads are a lighter color and a narrower line.

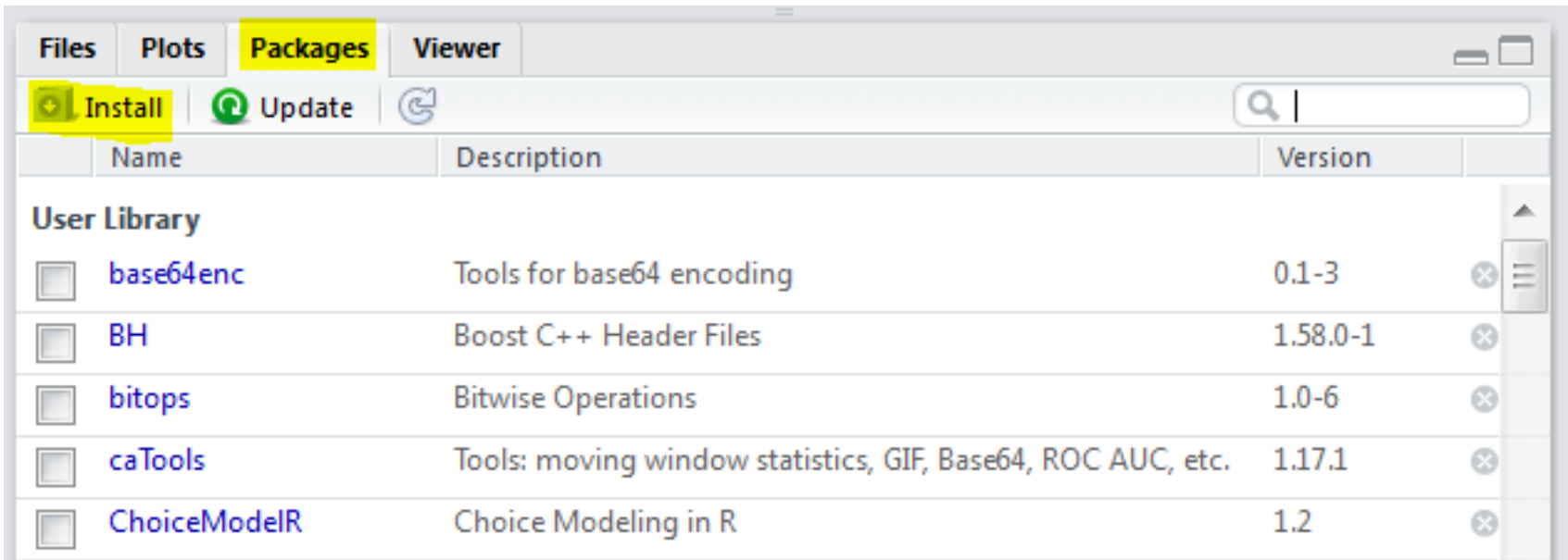
Install the library

- We'll need package ggmap - so we have to install it:

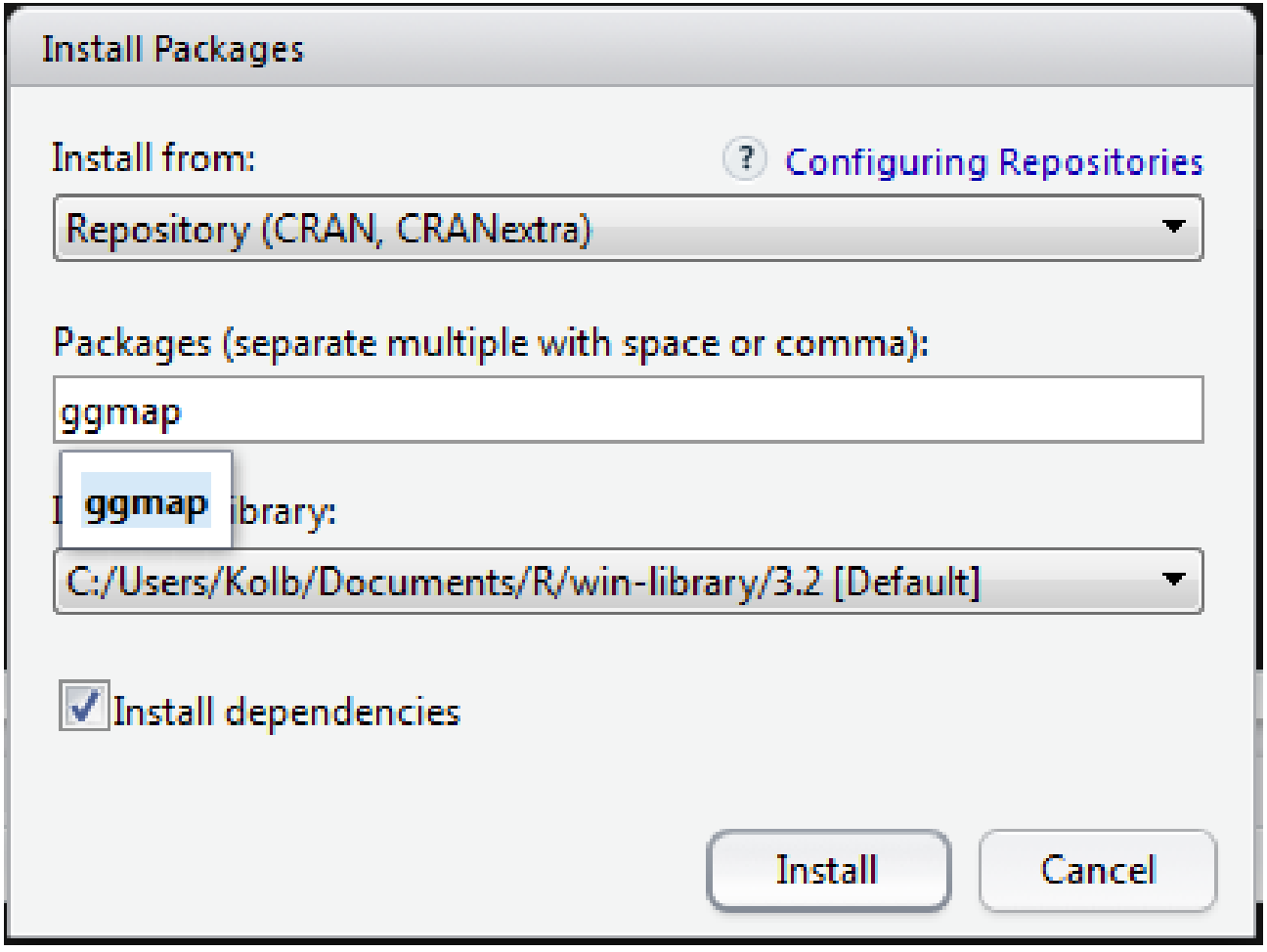
1. possibility

```
install.packages("ggmap")
```

2. possibility:



Install packages



Library ggmap - Hello world

- To load the library we use the command `library`

```
library(ggmap)
```

```
qmap("Mannheim")
```

Use another zoom level

- level 3 - continent
- level 10 - city
- level 21 - building

```
qmap("Germany", zoom = 6)
```

Get help with the questionmark

```
?qmap
```

Different components in the help

- Description
- Usage
- Arguments
- Value
- Author(s)
- See Also
- Examples

The examples section of help

Extract from the help file on qmap:

Examples

```
## Not run:
# these examples have been excluded for checking efficiency

qmap(location = "baylor university")
qmap(location = "baylor university", zoom = 14)
qmap(location = "baylor university", zoom = 14, source = "osm")
```

This examples can be directly copy-pasted to the console

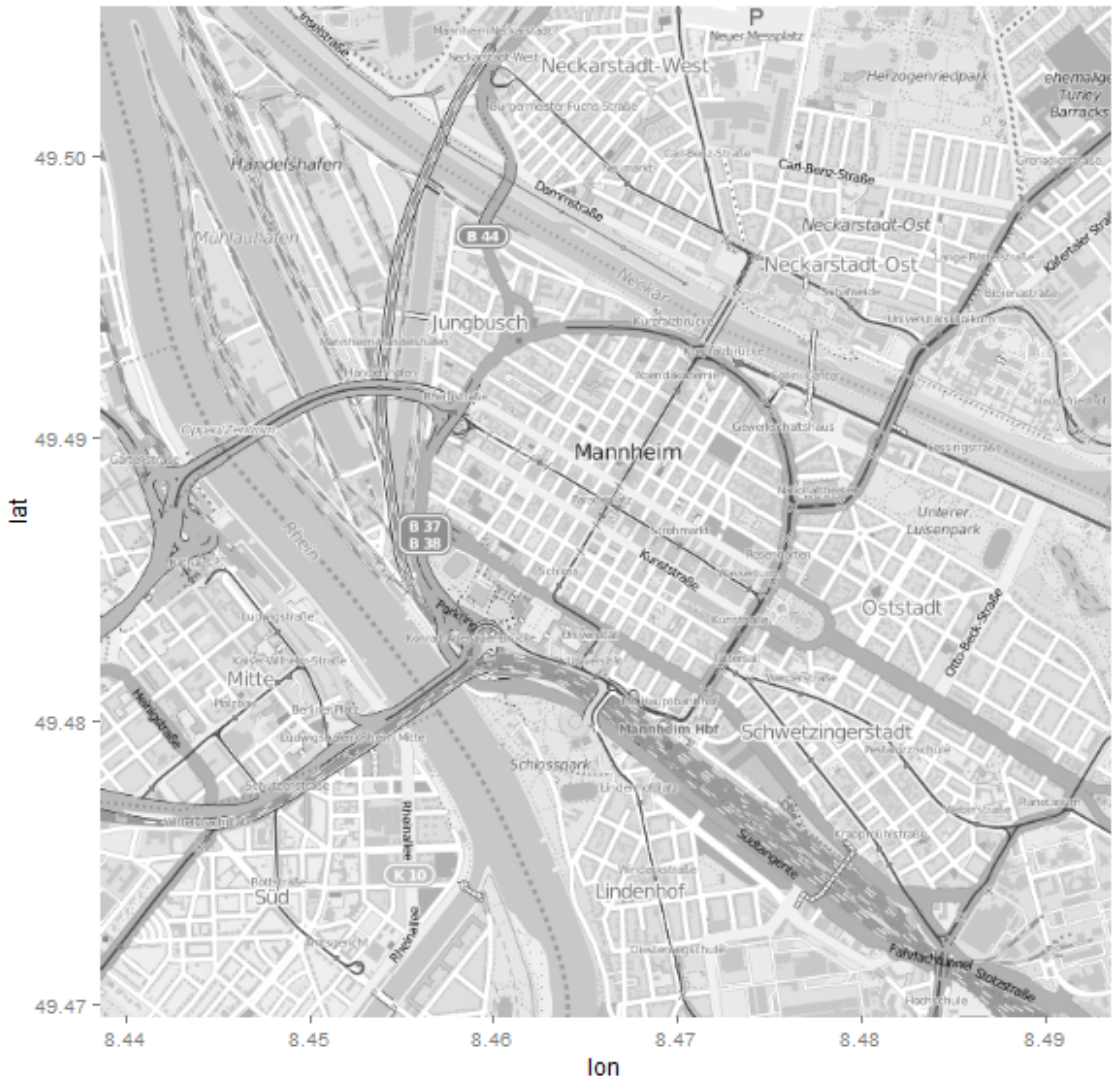
```
qmap(location = "baylor university")
qmap(location = "baylor university", zoom = 14)
# and so on
```

ggmap - source OpenStreetMap

```
qmap('Mannheim', zoom = 14, source="osm")
```

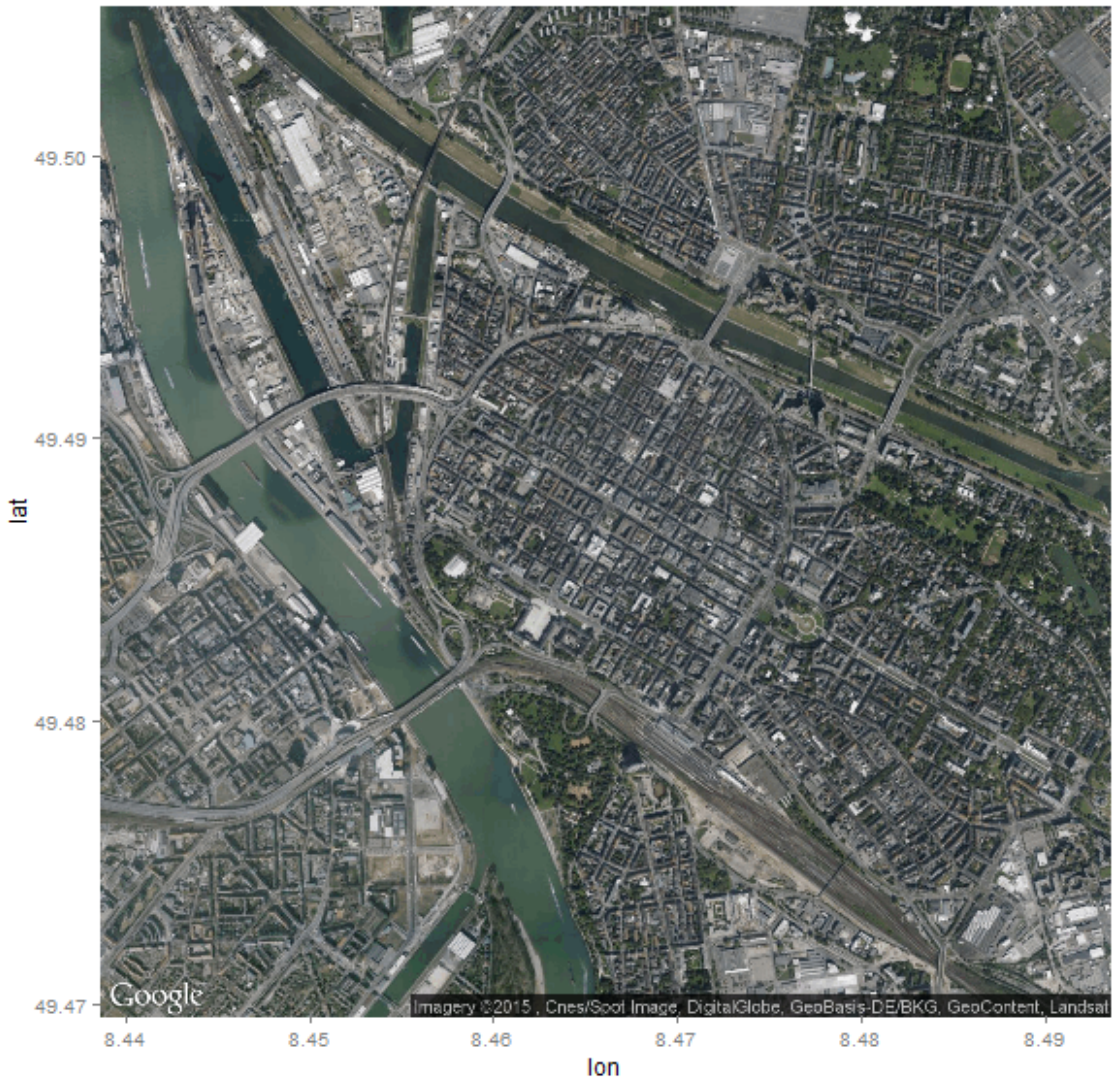
ggmap - OpenStreetMap - black/white

```
qmap('Mannheim', zoom = 14, source="osm",color="bw")
```



ggmap - maptype satellite

```
qmap('Mannheim', zoom = 14, maptype="satellite")
```



ggmap - maptype satellite zoom 21

```
qmap('Mannheim', zoom = 21, maptype="hybrid")
```

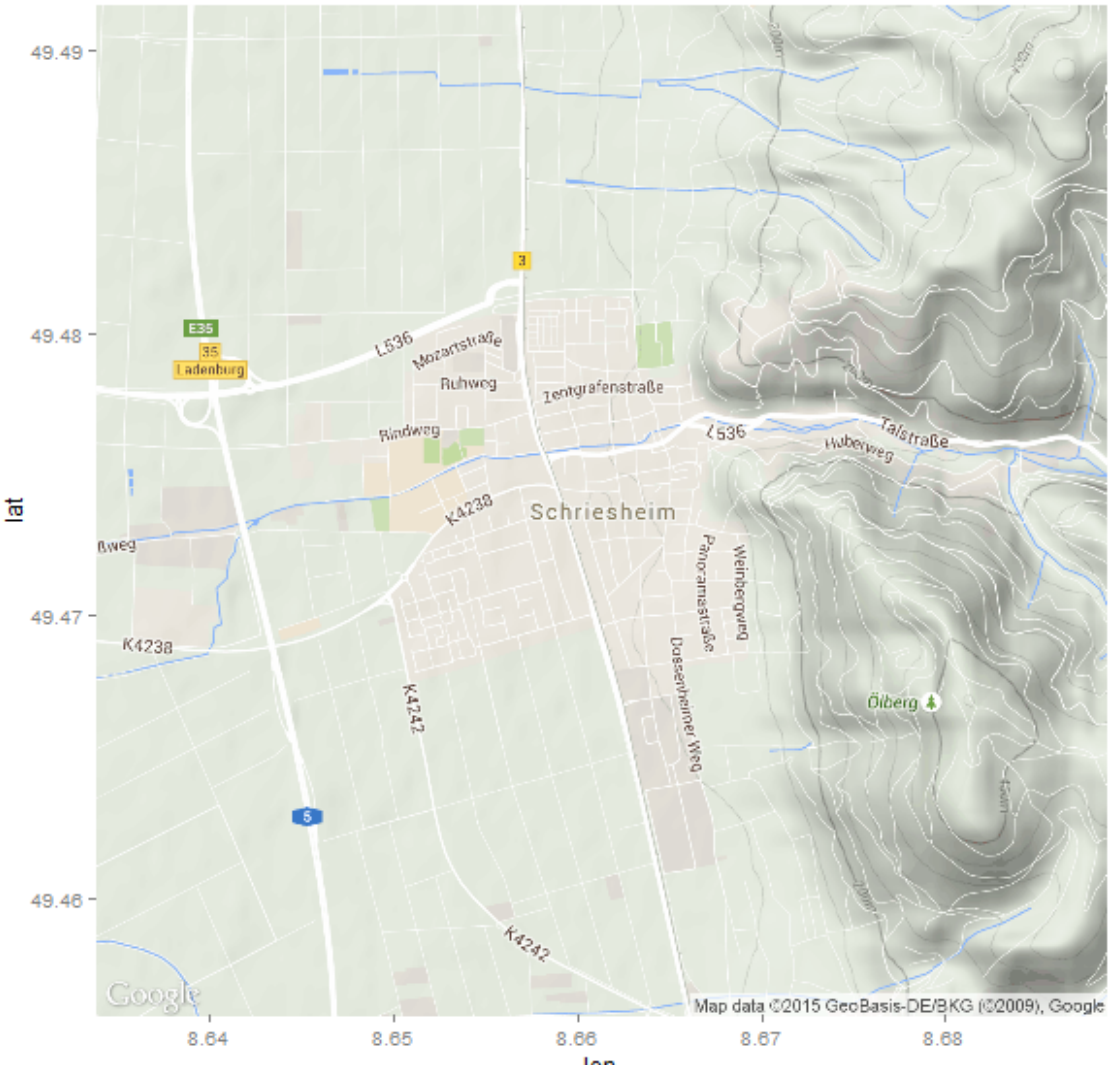
Terrain/physical maps

Physical maps illustrate the physical features of an area, such as the mountains, rivers and lakes. Colors are used to show relief differences in land elevations.

- The water is usually shown in blue.
- a lighter color is typically used at lower elevation and
- darker color indicate higher elevations.
- Contour lines can also be used to show elevation changes (they are normally spaced at regular intervals)

ggmap - terrain map

```
qmap('Schriesheim', zoom = 14,
 maptype="terrain")
```



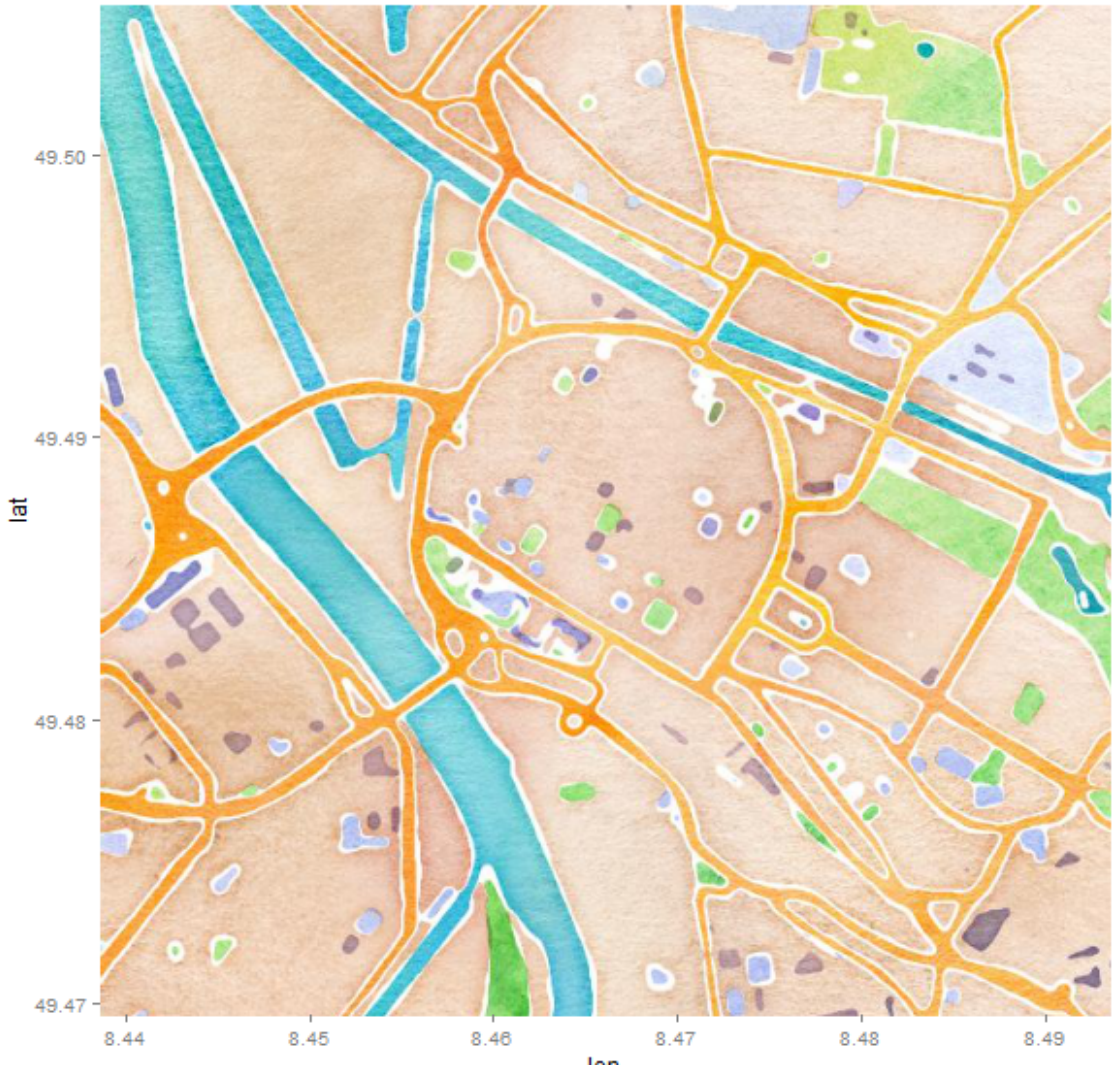
Abstracted maps

Source: Design faves

- Abstraction is used to create a map with only essential information
- Example metro maps - directions and little information for orientation is included
- In the following especially background maps

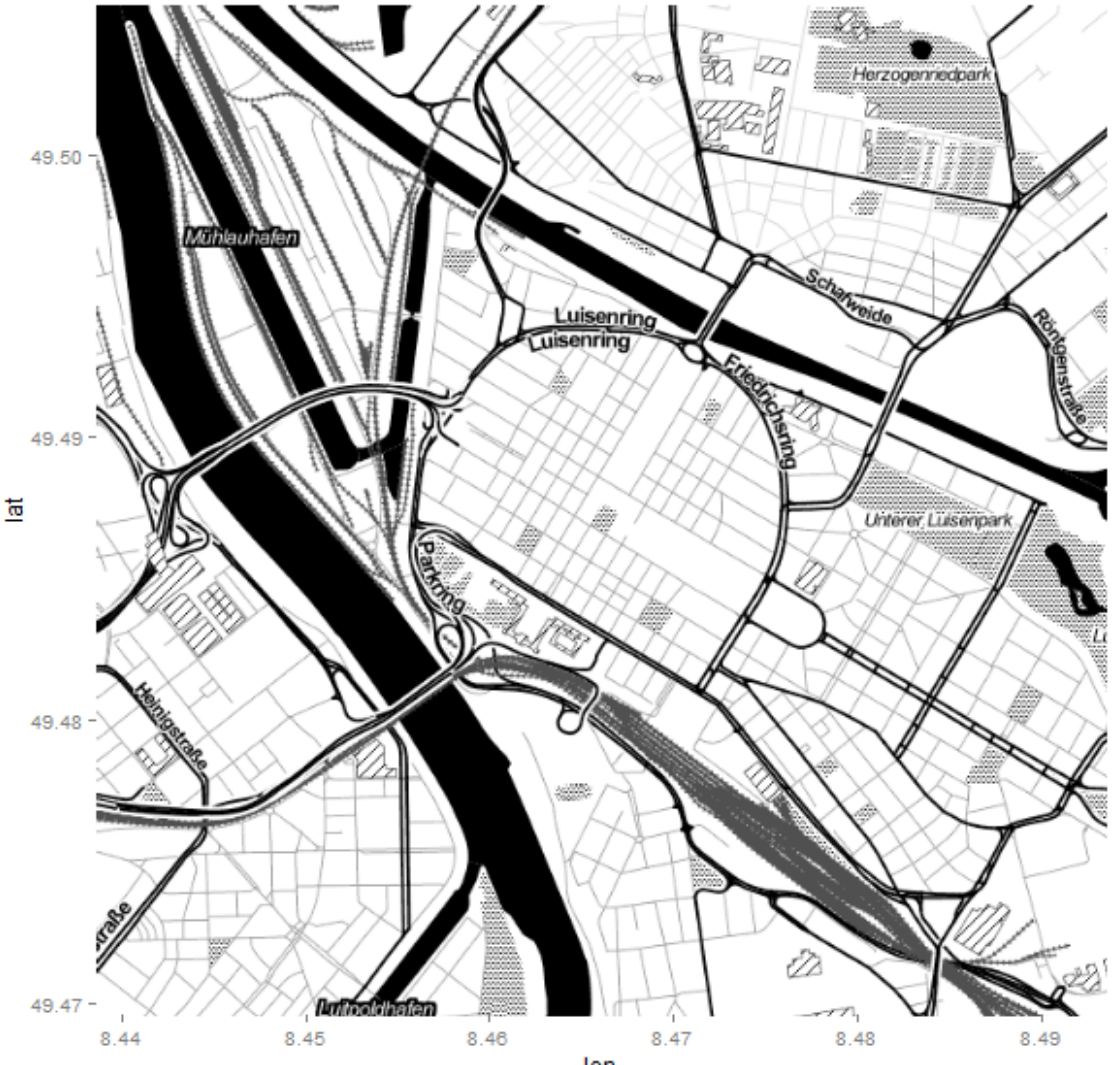
ggmap - maptype watercolor

```
qmap('Mannheim', zoom = 14,
 maptype="watercolor",source="stamen")
```



ggmap - source stamen

```
qmap('Mannheim', zoom = 14,
 maptype="toner",source="stamen")
```

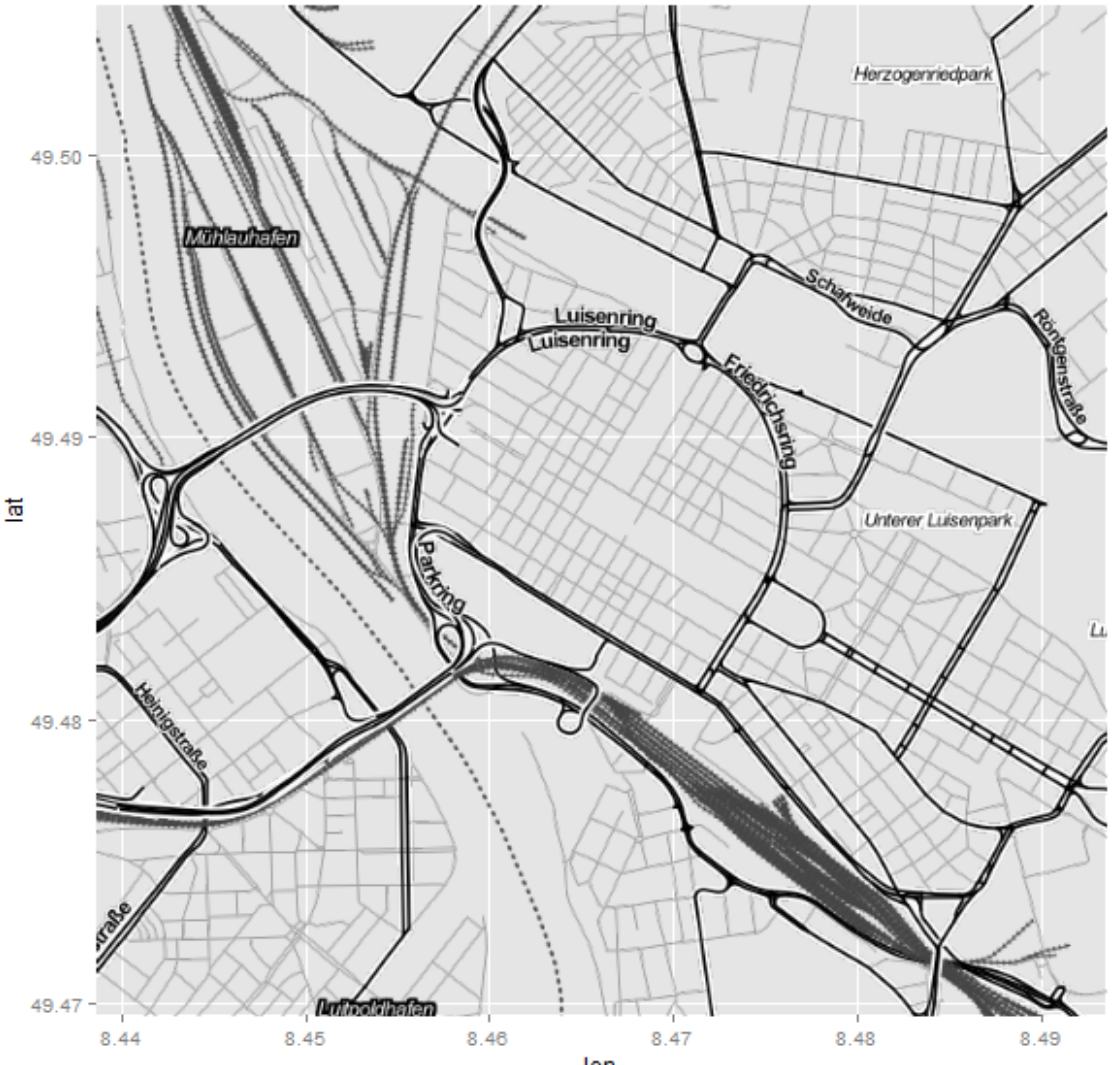


ggmap - maptype toner-lite

```
qmap('Mannheim', zoom = 14,
 maptype="toner-lite",source="stamen")
```

ggmap - maptype toner-hybrid

```
qmap('Mannheim', zoom = 14,
 maptype="toner-hybrid",source="stamen")
```



ggmap - maptype terrain-lines

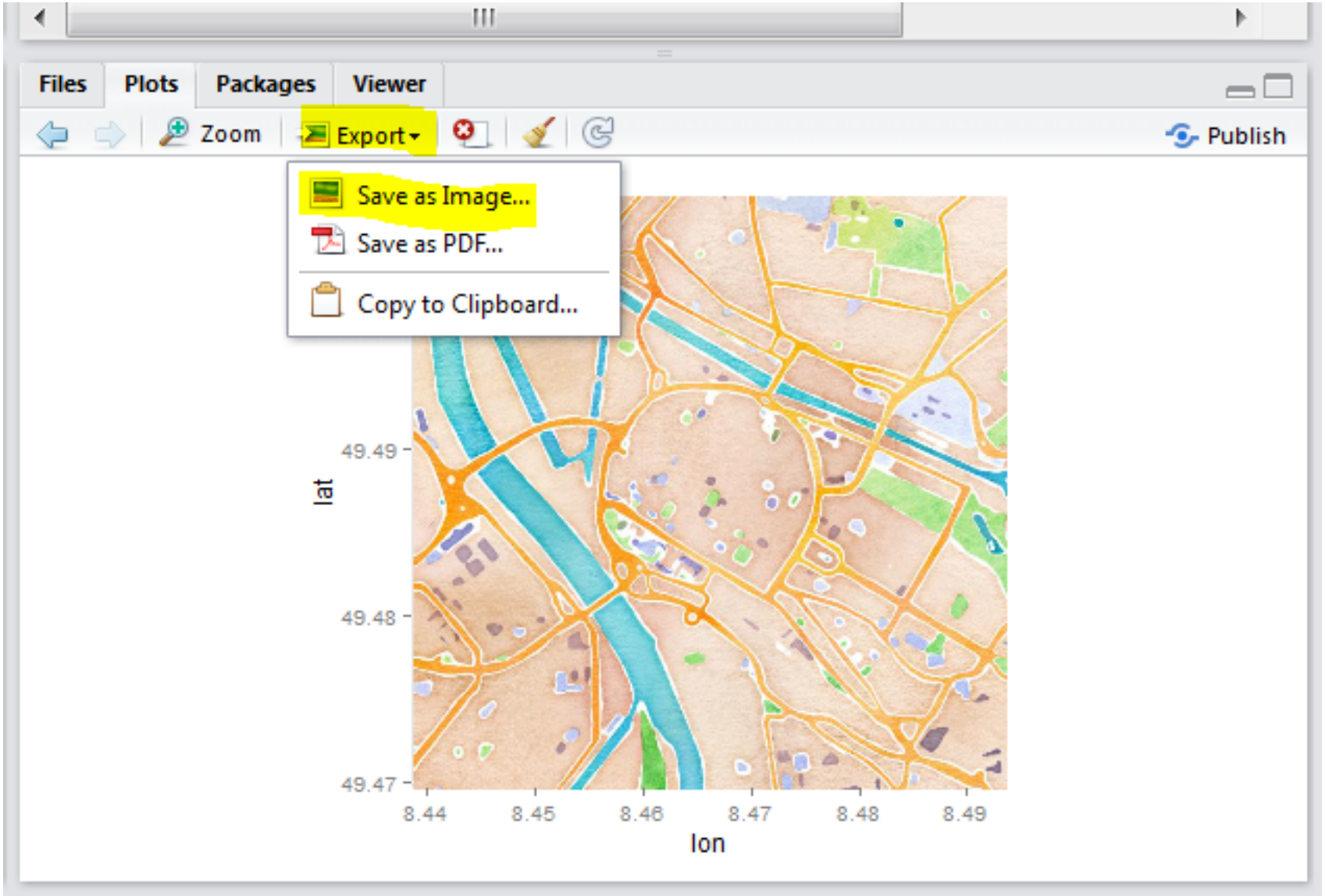
```
qmap('Mannheim', zoom = 14,
 maptype="terrain-lines",source="stamen")
```

Stamen maps

These high-contrast B+W (black and white) maps are featured in our Dotspotting project. They are perfect for data mashups and exploring river meanders and coastal zones.

Source: `http://maps.stamen.com/`

Save graphics



ggmap - create an object

- <- is an assignment operator which can be used to create an object
- This is useful if you work with several maps at the same time

```
MA_map <- qmap("Mannheim",
               zoom = 14,
               maptype="toner",
               source="stamen")
```

Geocoding

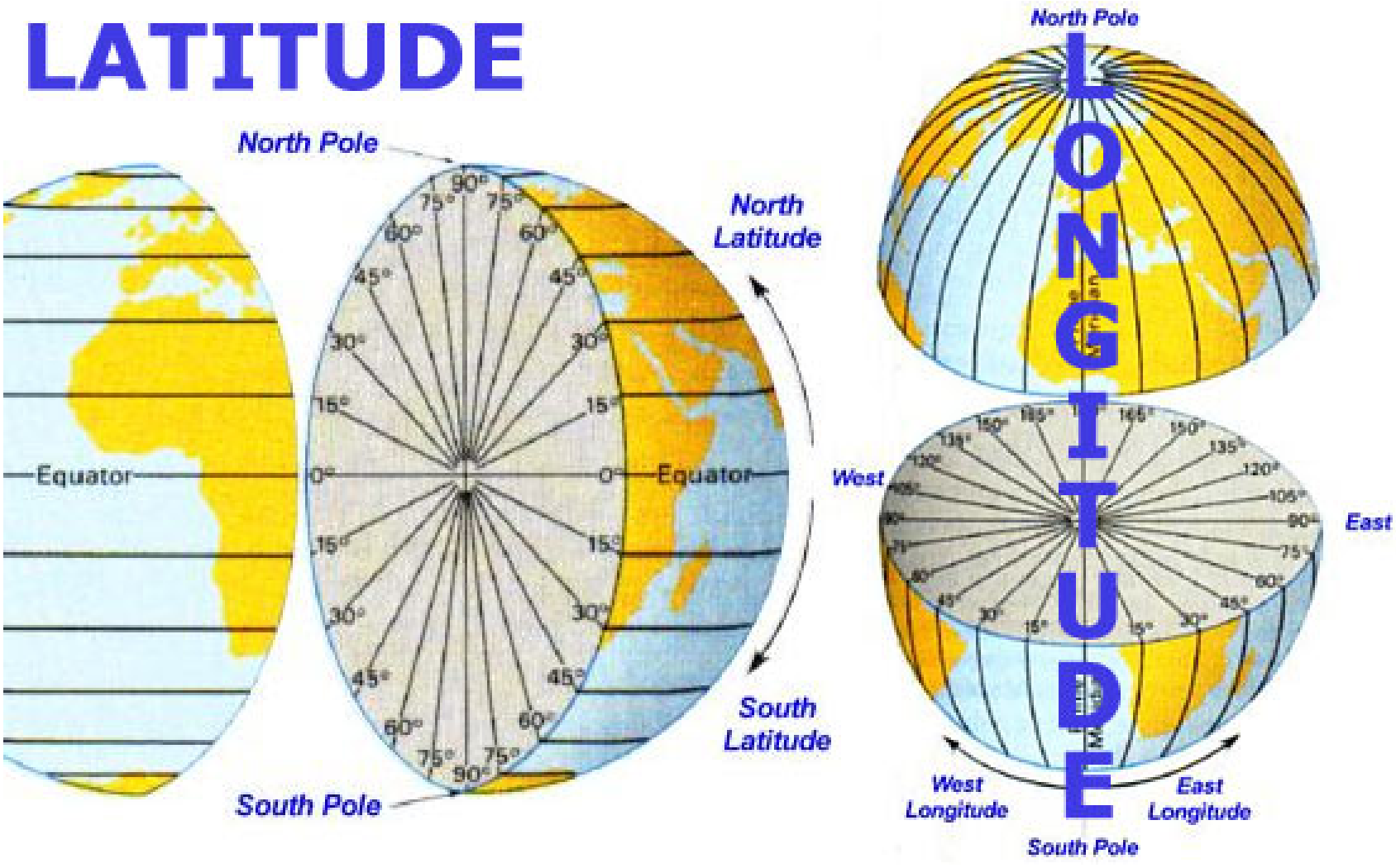
> *Geocoding (. . .) uses a description of a location, most typically a postal address or place name, to find geographic coordinates from spatial reference data . . .*

Wikipedia - Geocoding

```
library(ggmap)
geocode("Mannheim Wasserturm",source="google")
```

```
    lon      lat
1 8.473664 49.48483
```

Latitude and Longitude



Source

Reverse geocoding

> *Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighbourhoods, county, state, or country.*

Source: Wikipedia

```
revgeocode(c(48,8))
```

[1] "Qoriley Rd, Somalia"

Get the distance between 2 points

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim")
```

```
                from                  to    m     km      miles seconds
```

1 Q1, 4 Mannheim B2, 1 Mannheim 746 0.746 0.4635644 211 minutes hours 1 3.516667 0.05861111

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim",mode="walking")
```

```
                from                  to    m     km      miles seconds
```

1 Q1, 4 Mannheim B2, 1 Mannheim 546 0.546 0.3392844 420 minutes hours 1 7 0.1166667

Get another distance

```
mapdist("Q1, 4 Mannheim","B2, 1 Mannheim",mode="bicycling")
```

	from	to	m	km	miles	seconds
1	Q1, 4 Mannheim	B2, 1 Mannheim	555	0.555	0.344877	215
minutes	hours					
1	3.583333	0.05972222				

Geocoding - points of interest

```
POI1 <- geocode("B2, 1 Mannheim",source="google")
POI2 <- geocode("Hbf Mannheim",source="google")
POI3 <- geocode("Wasserturm Mannheim",source="google")
ListPOI <-rbind(POI1,POI2,POI3)
POI1;POI2;POI3
```

```
   lon      lat
```

1 8.462844 49.48569 lon lat 1 8.469879 49.47972 lon lat 1 8.473664 49.48483

Points in map

```
MA_map +
geom_point(aes(x = lon, y = lat),
data = ListPOI)
```

Points in map

```
MA_map +
geom_point(aes(x = lon, y = lat),col="red",
data = ListPOI)
```

ggmap - adding different colors

```
ListPOI$color <- c("A","B","C")
MA_map +
geom_point(aes(x = lon, y = lat,col=color),
data = ListPOI)
```

ggmap - bigger dots

```
ListPOI$size <- c(10,20,30)
MA_map +
geom_point(aes(x = lon, y = lat,col=color,size=size),
data = ListPOI)
```

Get a route from Google maps

```
from <- "Mannheim Hbf"
to <- "Mannheim B2 , 1"
route_df <- route(from, to, structure = "route")
```

More information

Draw a map with this information

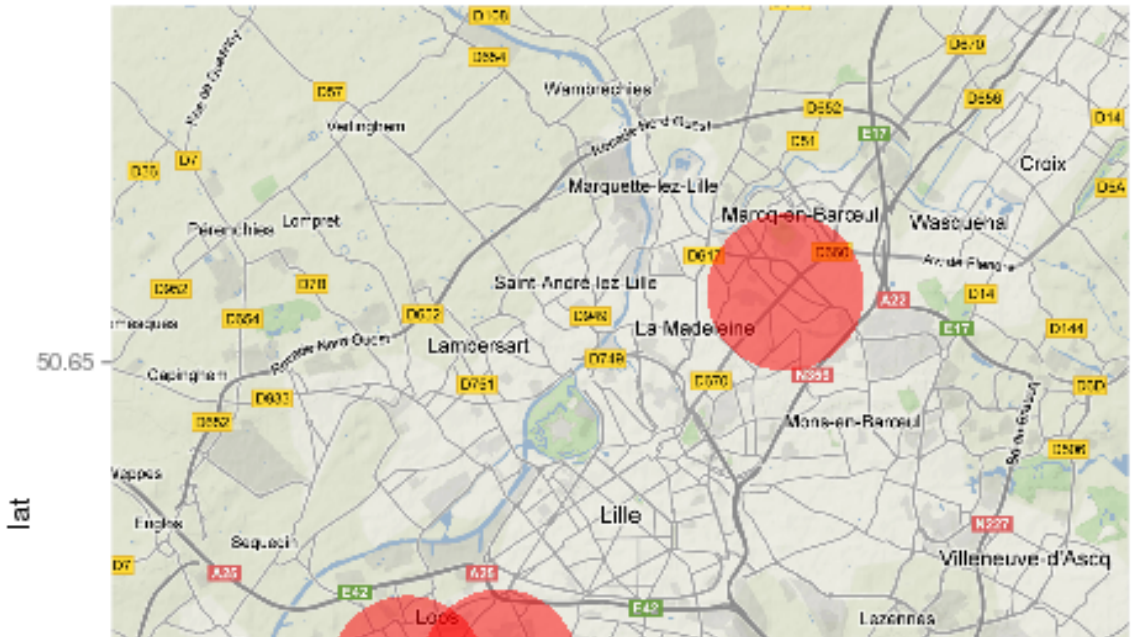
```
qmap("Mannheim Hbf", zoom = 14) +
  geom_path(
    aes(x = lon, y = lat),  colour = "red", size = 1.5,
    data = route_df, lineend = "round"
  )
```

Resources

- Article by David Kahle and Hadley Wickham on the usage of ggmap.
- Grab a map
- Making Maps in R

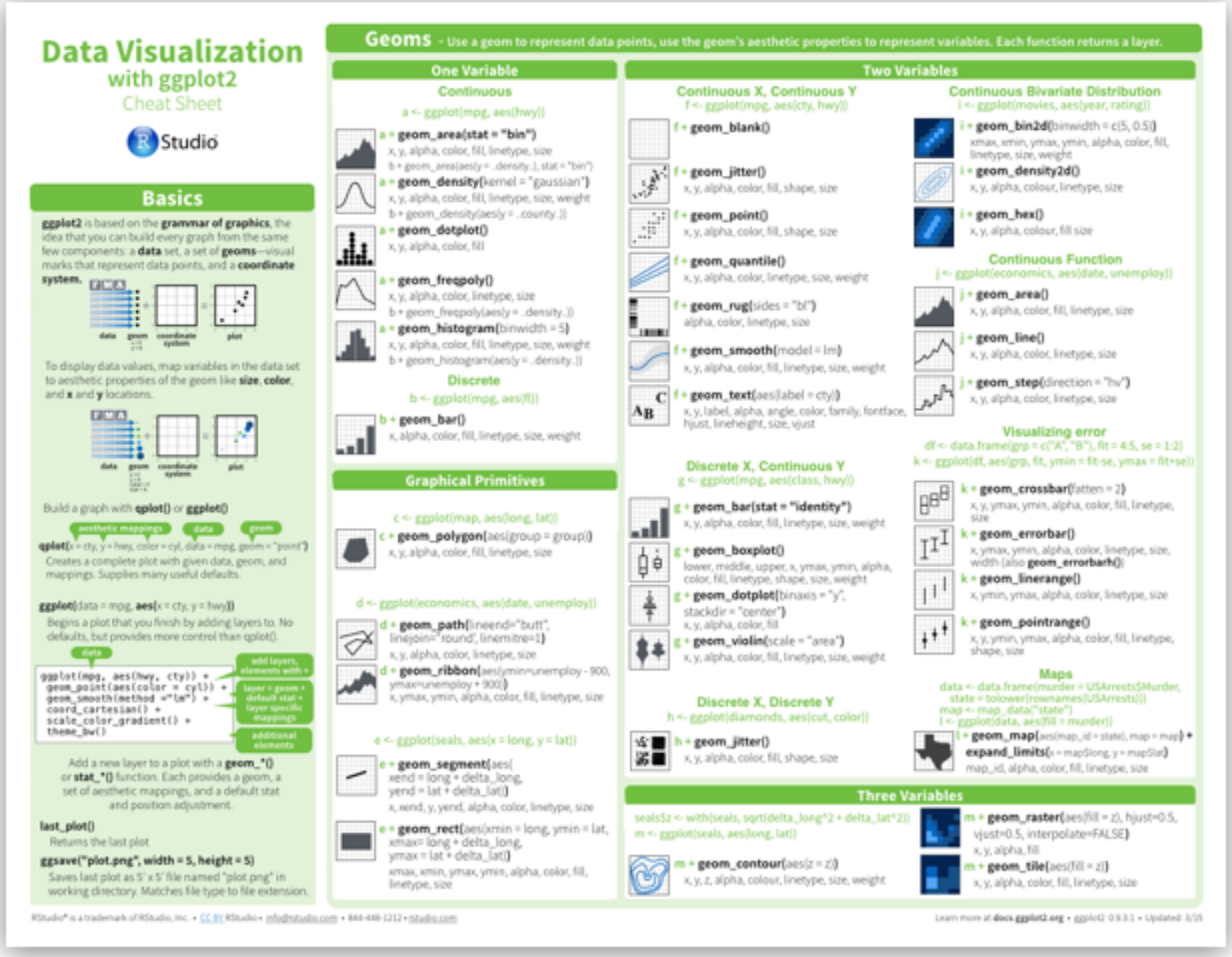
More about adding points

- Usage of geom_point
- Question on stackoverflow



Cheatsheet

- Cheatsheet on data visualisation



Data Visualization with ggplot2

Cheat Sheet

RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.

To display data values, map variables in the data set to aesthetic properties of the geom like **size**, **color**, and **x** and **y** locations.

Build a graph with **qplot()** or **ggplot()**

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**
Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method ="lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()
Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

- a + **geom_area(stat = "bin")** — x, y, alpha, color, fill, linetype, size
- a + **geom_density(**kernel = "gaussian"**)** — x, y, alpha, color, fill, linetype, size, weight
- a + **geom_dotplot()** — x, y, alpha, color, fill
- a + **geom_freqpoly()** — x, y, alpha, color, linetype, size
- a + **geom_histogram(**binwidth = 5**)** — x, y, alpha, color, fill, linetype, size, weight

Discrete

b <- ggplot(mpg, aes(fl))

- b + **geom_bar()** — x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

- c + **geom_polygon(**aes(group = group)**)** — x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

- d + **geom_path(**lineend="butt", linejoin="round", linemitre=1**)** — x, y, alpha, color, linetype, size
- d + **geom_ribbon(**aes(ymin=unemploy - 900, ymax=unemploy + 900)**)** — x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

- e + **geom_segment(**aes(xend = long + delta_long, yend = lat + delta_lat)**)** — x, xend, y, yend, alpha, color, linetype, size
- e + **geom_rect(**aes(xmin = long, ymin = lat, xmax= long + delta_long, ymax = lat + delta_lat)**)** — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

- f + **geom_blank()**
- f + **geom_jitter()** — x, y, alpha, color, fill, shape, size
- f + **geom_point()** — x, y, alpha, color, fill, shape, size
- f + **geom_quantile()** — x, y, alpha, color, linetype, size, weight
- f + **geom_rug(**sides = "bl"**)** — alpha, color, linetype, size
- f + **geom_smooth(**model = lm**)** — x, y, alpha, color, fill, linetype, size, weight
- f + **geom_text(**aes(label = cty)**)** — x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

- g + **geom_bar(stat = "identity")** — x, y, alpha, color, fill, linetype, size, weight
- g + **geom_boxplot()** — lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight
- g + **geom_dotplot(**binaxis = "y", stackdir = "center"**)** — x, y, alpha, color, fill
- g + **geom_violin(**scale = "area"**)** — x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

- h + **geom_jitter()** — x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

- i + **geom_bin2d(**binwidth = c(5, 0.5)**)** — xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight
- i + **geom_density2d()** — x, y, alpha, colour, linetype, size
- i + **geom_hex()** — x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

- j + **geom_area()** — x, y, alpha, color, fill, linetype, size
- j + **geom_line()** — x, y, alpha, color, linetype, size
- j + **geom_step(**direction = "hv"**)** — x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

- k + **geom_crossbar(**fatten = 2**)** — x, y, ymax, ymin, alpha, color, fill, linetype, size
- k + **geom_errorbar()** — x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)
- k + **geom_linerange()** — x, ymin, ymax, alpha, color, linetype, size
- k + **geom_pointrange()** — x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

- l + **geom_map(**aes(map_id = state), map = map**)** + **expand_limits(**x = map$long, y = map$lat**)** — map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

- m + **geom_contour(**aes(z = z)**)** — x, y, z, alpha, colour, linetype, size, weight
- m + **geom_raster(**aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)** — x, y, alpha, fill
- m + **geom_tile(**aes(fill = z)**)** — x, y, alpha, color, fill, linetype, size

Resources and literature

ggmap: Spatial Visualization with ggplot2

by David Kahle and Hadley Wickham