Logistics Network Visualization with the Google Maps API

Tellef Solberg

20/02/2020

Context

Map data has become increasingly more available, however the active usage of map data for visualization and decision support is as of today quite underutilized. Opportunities in using map data actively are especially great for supply chain and logistics departments. In this project, the *Google Maps API* will be used in order to visualize and analyze map data, as Googles map data is amongst the richest in the world.

Setting up the Google Maps API and recieving the credentials can be done in the Google Cloud Console on their webpages:

https://cloud.google.com/maps-platform/

The complete R-file containing all variable declarations and data wrangling can be found on my GitHub:

https://github.com/TellSol/Logistics-and-Routing

Problem

There are four problems we are trying to solve in this project paper:

1. How can we visualize a roadmap of Germany and mark a certain city?
2. How can we mark five different locations on the Germany map?
3. How can we visualize a distribution network for a trucking company?
4. How can we visualize air-travelling routes on the world map?

All problems will be solved using the *Google Maps API* and the R-packages; *ggmap*, *mapproj* and *maps*.

Visualizations with corresponding R-Code

Some initial setups

Load in the packages: *Tidyverse, ggmap, ggthemes, mapproj, maps*,

then insert the API key using the function, Register_Google(Key = *private API key goes here*).

Now we are set up and ready for using the *Google Maps API* in *R*, and we start of with solving the first problem we stated.

1. How can we visualize a roadmap of Germany and mark a certain city?

Acquiring a map over Germany and marking a city, say Munich, consider the following code and visualization output:

the functions *geocode* and *ggmap* are used here to first obtain the longitude and lattitude of each varaible decleared, and then both are used to visualization.

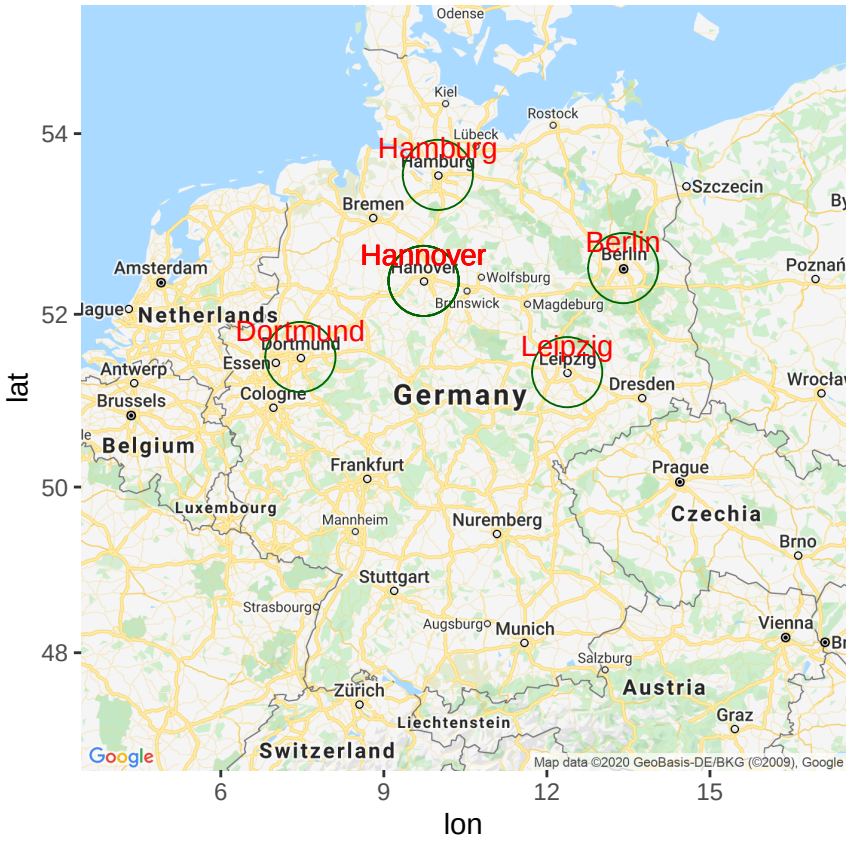
2. How can we mark five different locations on the Germany map?

For this problem, say we want to mark the German cities: Hannover, Berlin, Leipzig, Hamburg and Dortmund. Consider the following code and visualization output:

```
# VISUALIZE THE NETWORK FROM HANNOVER TO THE FOUR CITIES WITH... :

# ROADMAP (WHICH LOOKS KIND OF NOISY)
ggmap(get_map("germany", zoom = 6, maptype = "roadmap")) +
  geom_point(mapping = aes(x = lon, y = lat), color = "darkgreen",
             data = places, size = 12, shape = 21) +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "red", data = places, nudge_y = 0.3)
```

```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=germany&zoom=6&size=640x640&scale=2&ma
```

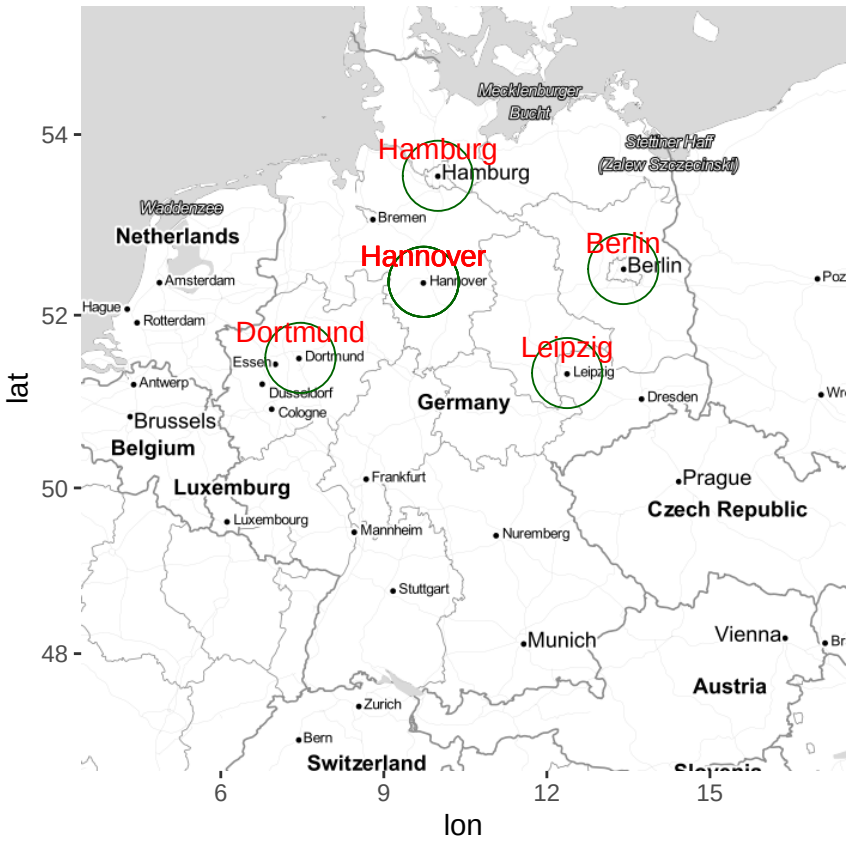
```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=germany&key=xxx
```




```
# TONER-LITE STYLE
ggmap(get_map("germany", zoom = 6, maptype = "toner-lite")) +
  geom_point(mapping = aes(x = lon, y = lat), color = "darkgreen",
```

```
            data = places, size = 12, shape = 21) +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "red", data = places, nudge_y = 0.3)
```

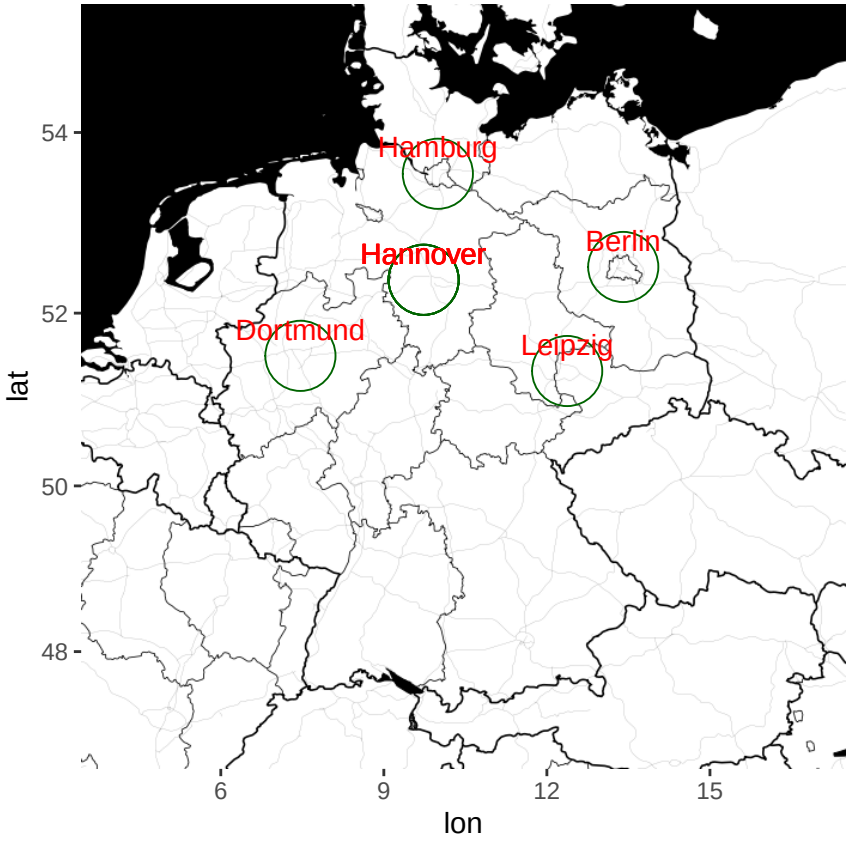
```
## maptype = "toner-lite" is only available with source = "stamen".

## resetting to source = "stamen"...

## Source : https://maps.googleapis.com/maps/api/staticmap?center=germany&zoom=6&size=640x640&scale=2&ma

## Source : https://maps.googleapis.com/maps/api/geocode/json?address=germany&key=xxx

## Source : http://tile.stamen.com/toner-lite/6/32/20.png

## Source : http://tile.stamen.com/toner-lite/6/33/20.png

## Source : http://tile.stamen.com/toner-lite/6/34/20.png

## Source : http://tile.stamen.com/toner-lite/6/35/20.png

## Source : http://tile.stamen.com/toner-lite/6/32/21.png

## Source : http://tile.stamen.com/toner-lite/6/33/21.png

## Source : http://tile.stamen.com/toner-lite/6/34/21.png

## Source : http://tile.stamen.com/toner-lite/6/35/21.png

## Source : http://tile.stamen.com/toner-lite/6/32/22.png

## Source : http://tile.stamen.com/toner-lite/6/33/22.png

## Source : http://tile.stamen.com/toner-lite/6/34/22.png

## Source : http://tile.stamen.com/toner-lite/6/35/22.png
```



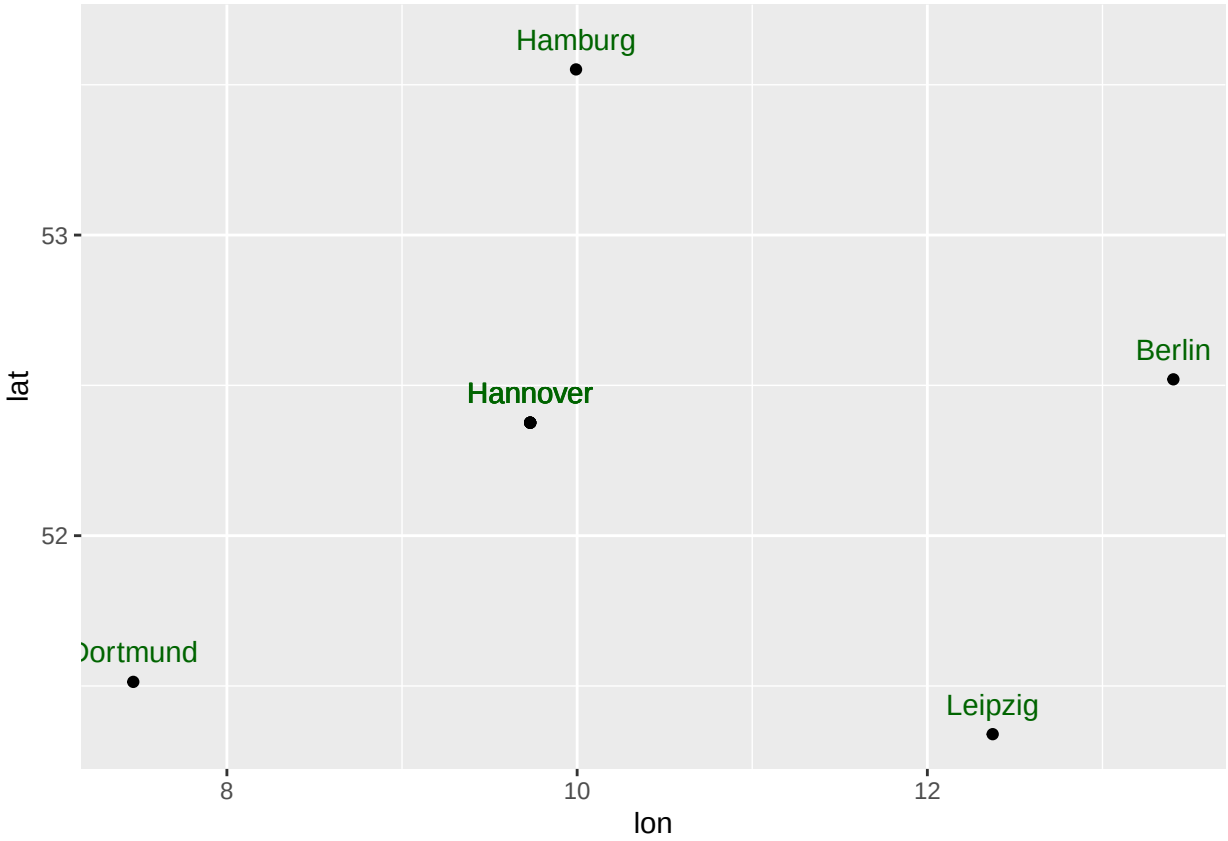

```
# TONER-BACKGROUND STYLE
ggmap(get_map("germany", zoom = 6, maptype = "toner-background")) +
  geom_point(mapping = aes(x = lon, y = lat), color = "darkgreen",
             data = places, size = 12, shape = 21) +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "red", data = places, nudge_y = 0.3)
```

```
## maptype = "toner-background" is only available with source = "stamen".

## resetting to source = "stamen"...

## Source : https://maps.googleapis.com/maps/api/staticmap?center=germany&zoom=6&size=640x640&scale=2&ma

## Source : https://maps.googleapis.com/maps/api/geocode/json?address=germany&key=xxx

## Source : http://tile.stamen.com/toner-background/6/32/20.png

## Source : http://tile.stamen.com/toner-background/6/33/20.png

## Source : http://tile.stamen.com/toner-background/6/34/20.png

## Source : http://tile.stamen.com/toner-background/6/35/20.png

## Source : http://tile.stamen.com/toner-background/6/32/21.png
```

```
## Source : http://tile.stamen.com/toner-background/6/33/21.png

## Source : http://tile.stamen.com/toner-background/6/34/21.png

## Source : http://tile.stamen.com/toner-background/6/35/21.png

## Source : http://tile.stamen.com/toner-background/6/32/22.png

## Source : http://tile.stamen.com/toner-background/6/33/22.png

## Source : http://tile.stamen.com/toner-background/6/34/22.png

## Source : http://tile.stamen.com/toner-background/6/35/22.png
```




```
# GENERIC LONGITUDE-LATITUDE (2-DIMENSIONAL PLANE WITHOUT MAP)
ggplot(mapping =aes(x = lon, y = lat, group = grp), data = places) +
  geom_point() +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "darkgreen", data = places, nudge_y = 0.1)
```



Here, the same priciple as in the first problem is used, but now only with string vectors instead of a single string. Take some time to appreciate the different visualizations.

3. How can we visualize a distribution network for a trucking company?

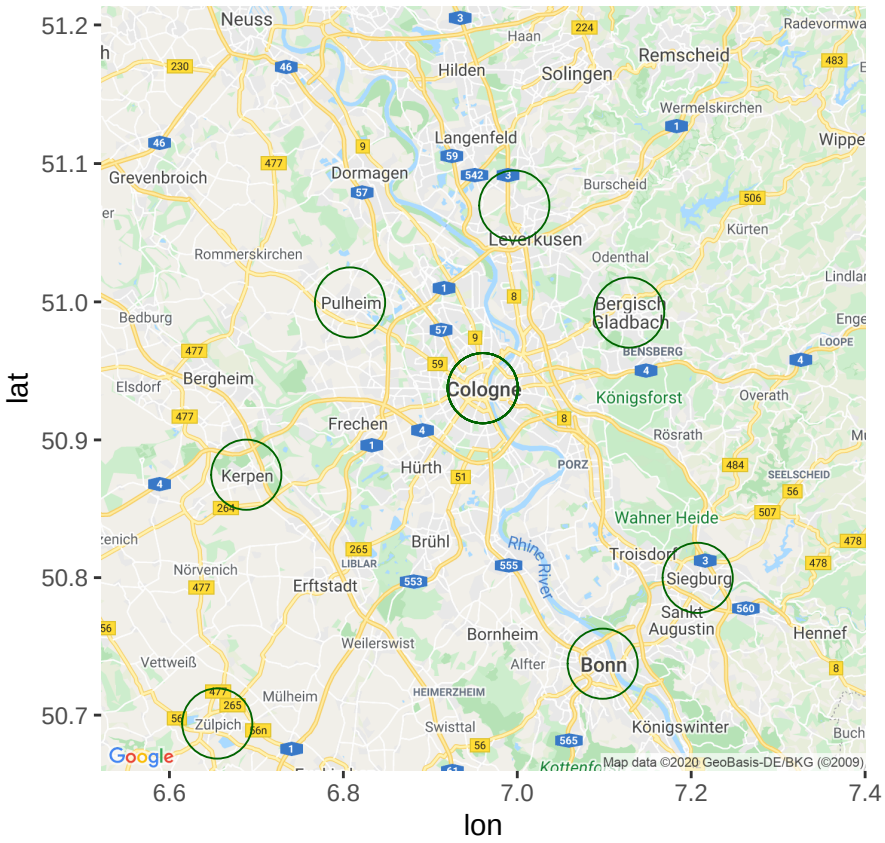
Lets assume that we have a trucking company located in Cologne. It has a northern route and a southern route. Note that inserting "Cologne" several times in the locationCenter variable is for visualization purposes. Consider the following code and visualization output:

```
# VISUALIZATION OF NETWORK WITH ROADMAP
ggmap(get_map("Cologne", zoom = 10, maptype = "roadmap")) +
  geom_point(mapping = aes(x = lon, y = lat), color = "darkgreen",
             data = logisticsNetwork, size = 12, shape = 21) +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "red", data = places, nudge_y = 0.3)
```

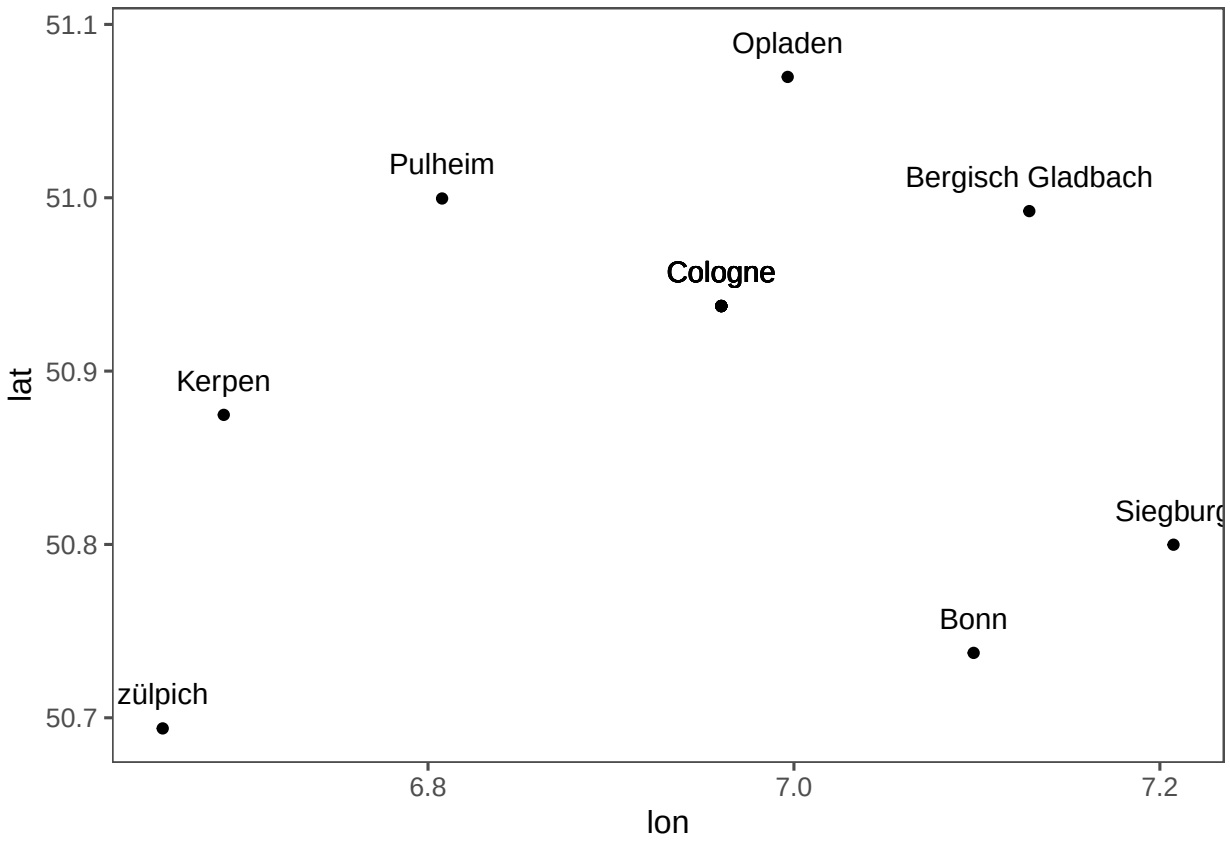
```
## Source : https://maps.googleapis.com/maps/api/staticmap?center=Cologne&zoom=10&size=640x640&scale=2&m
```

```
## Source : https://maps.googleapis.com/maps/api/geocode/json?address=Cologne&key=xxx
```

```
## Warning: Removed 9 rows containing missing values (geom_text).
```

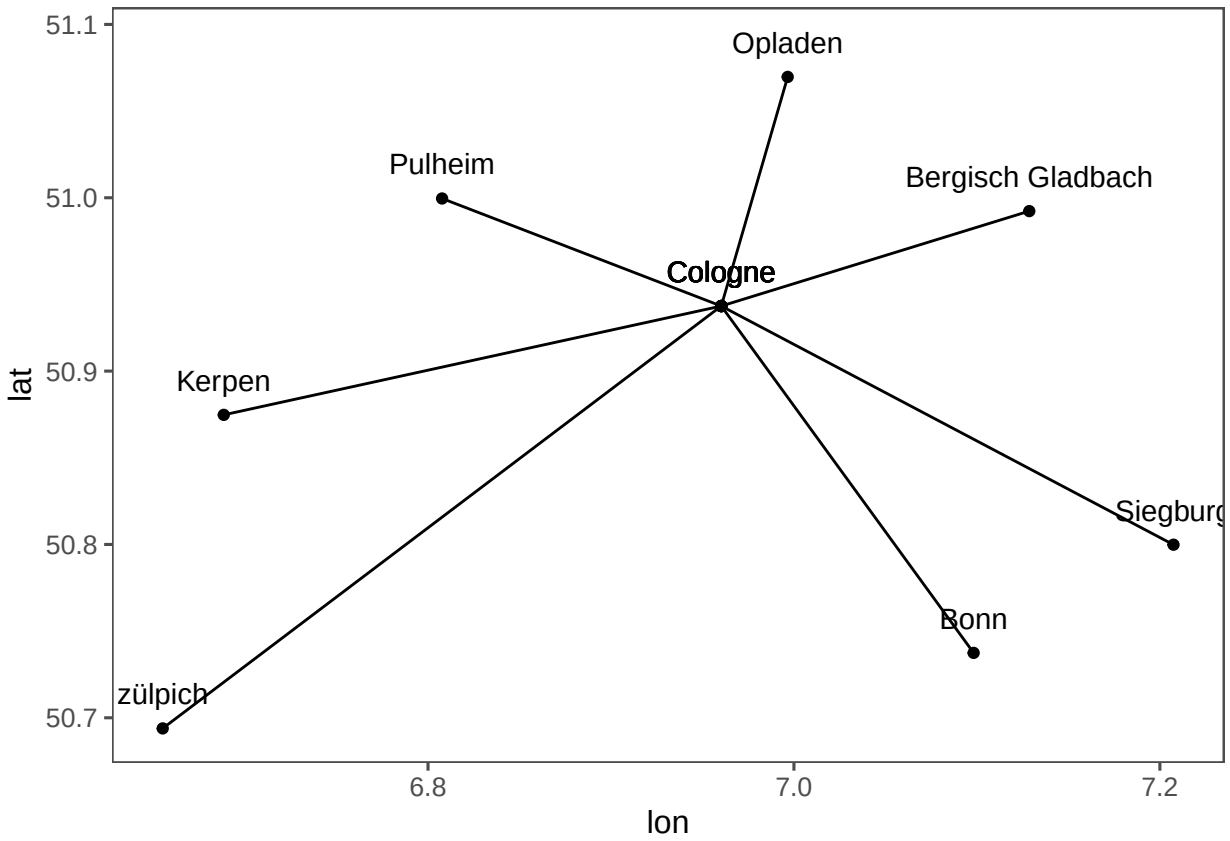



```
# GENERIC LONGITUDE LATITUDE ON A 2-DIMENSIONAL PLANE
ggplot(mapping = aes(x = lon, y = lat), data = logisticsNetwork) +
  geom_point() +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "black", data = logisticsNetwork, nudge_y = 0.02) +
  theme_few()
```



Lets consider a *distribution center driven network* where all traffic has to go through the distribution center in Cologne:

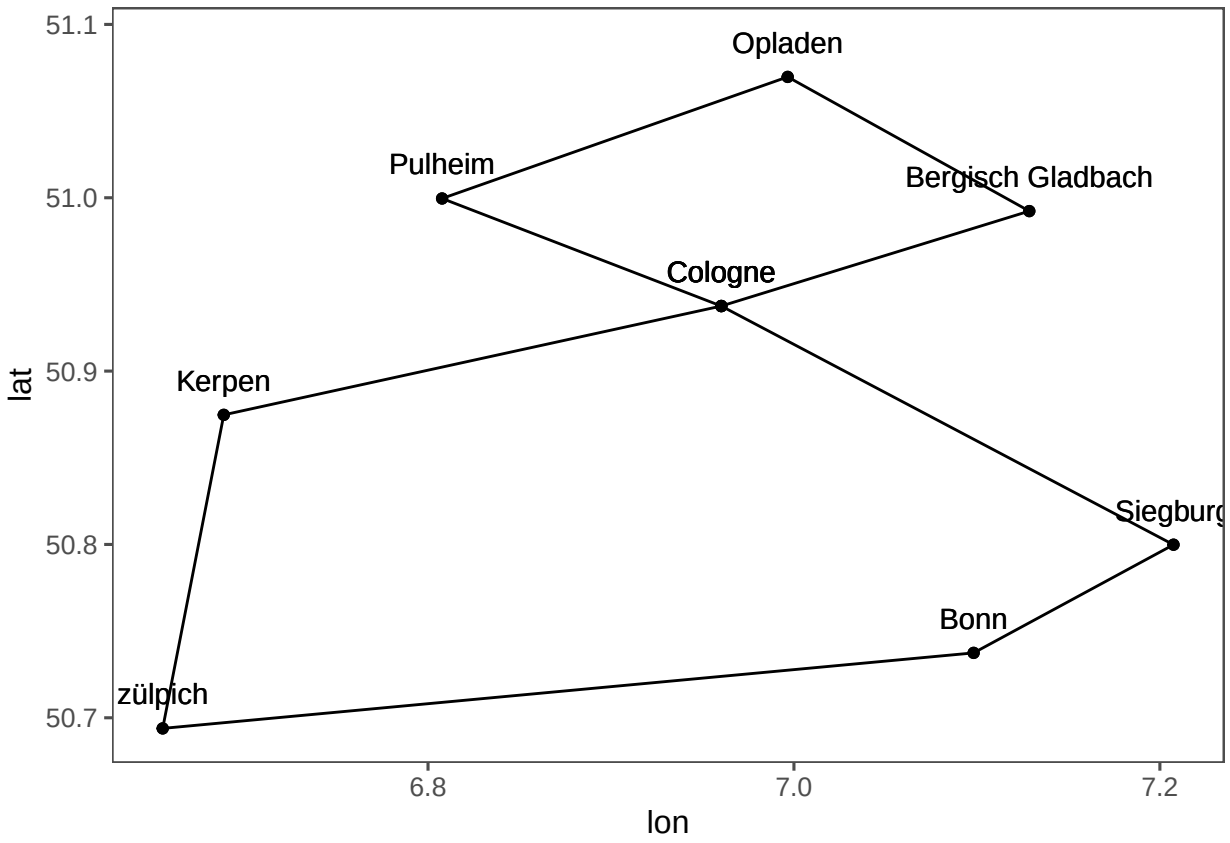
```
ggplot(mapping = aes(x = lon, y = lat, group = group), data = logisticsNetwork) +
  geom_point() + geom_line() +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "black", data = logisticsNetwork, nudge_y = 0.02) +
  theme_few()
```



Next, lets consider a *Milk run driven network* where the trucks does deliveries to several locations in one transport mission. Note that for connection pruposes, most variables are delcared again with double on several cities to make visualization possible.

Consider the following code and visualization output:

```
ggplot(mapping = aes(x = lon, y = lat, group = group), data = logisticsNetworkMilkRun) +
  geom_point() + geom_path() +
  geom_text(mapping = aes(x = lon, y = lat, label = name),
            color = "black", data = logisticsNetworkMilkRun, nudge_y = 0.02) +
  theme_few()
```

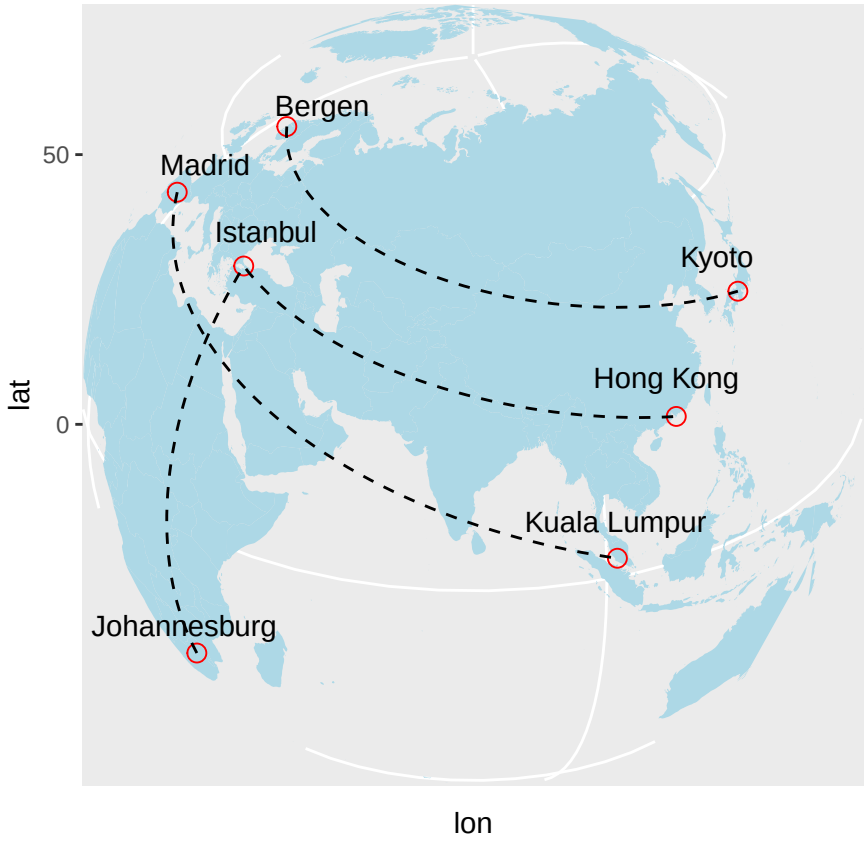


Here, the trucks deliver in a more economic way when transportation costs is concerned.

4. How can we visualize air-travelling routes on the world map?

Lets assume that we want to mark seven cities on a world map and visualize air-travelling routes between the cities. Consider the following code and corresponding output:

```
ggplot(mapping = aes(x = lon, y = lat, group = group), data = mapData1) +
  geom_polygon(data = world,  aes(long, lat, group = group),
               fill = "lightblue") +
  geom_point(data = mapData1, aes(lon, lat),
             colour = "red", size = 3, shape = 21) +
  coord_map("ortho", orientation = c(30, 80, 0)) +
  geom_text(mapping = aes(x = lon, y = lat, label = cities),
            color = "black", data = mapData, nudge_y = 6) +
  geom_line(linetype = "dashed")
```




```
  theme_void()
```

```
## List of 28
##  $ line                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ rect                      : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ text                      :List of 11
##   ..$ family       : chr ""
##   ..$ face         : chr "plain"
##   ..$ colour       : chr "black"
##   ..$ size         : num 11
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : num 0
##   ..$ lineheight   : num 0.9
##   ..$ margin       : 'margin' num [1:4] 0pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : logi FALSE
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ axis.title                : list()
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.text                 : list()
```

```
##   ..- attr(*, "class")= chr [1:2] "element_blank" "element"
##  $ axis.ticks.length         : 'unit' num 0pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ axis.ticks.length.x       : NULL
##  $ axis.ticks.length.x.top   : NULL
##  $ axis.ticks.length.x.bottom: NULL
##  $ axis.ticks.length.y       : NULL
##  $ axis.ticks.length.y.left  : NULL
##  $ axis.ticks.length.y.right : NULL
##  $ legend.key.size           : 'unit' num 1.2lines
##   ..- attr(*, "valid.unit")= int 3
##   ..- attr(*, "unit")= chr "lines"
##  $ legend.text               :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.title              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ legend.position           : chr "right"
##  $ legend.box                : NULL
##  $ panel.spacing             : 'unit' num 5.5pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ panel.ontop               : logi FALSE
##  $ plot.title                :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
```

```
##   ..$ margin       : 'margin' num [1:4] 5.5pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.subtitle             :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : NULL
##   ..$ hjust        : num 0
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.caption              :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : num 1
##   ..$ vjust        : num 1
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : 'margin' num [1:4] 5.5pt 0pt 0pt 0pt
##   .. ..- attr(*, "valid.unit")= int 8
##   .. ..- attr(*, "unit")= chr "pt"
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag                  :List of 11
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 1.2
##   ..$ hjust        : num 0.5
##   ..$ vjust        : num 0.5
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ plot.tag.position         : chr "topleft"
##  $ plot.margin               : 'unit' num [1:4] 0lines 0lines 0lines 0lines
##   ..- attr(*, "valid.unit")= int 3
##   ..- attr(*, "unit")= chr "lines"
##  $ strip.text                :List of 11
```

```
##   ..$ family       : NULL
##   ..$ face         : NULL
##   ..$ colour       : NULL
##   ..$ size         : 'rel' num 0.8
##   ..$ hjust        : NULL
##   ..$ vjust        : NULL
##   ..$ angle        : NULL
##   ..$ lineheight   : NULL
##   ..$ margin       : NULL
##   ..$ debug        : NULL
##   ..$ inherit.blank: logi TRUE
##   ..- attr(*, "class")= chr [1:2] "element_text" "element"
##  $ strip.switch.pad.grid     : 'unit' num 2.75pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  $ strip.switch.pad.wrap     : 'unit' num 2.75pt
##   ..- attr(*, "valid.unit")= int 8
##   ..- attr(*, "unit")= chr "pt"
##  - attr(*, "class")= chr [1:2] "theme" "gg"
##  - attr(*, "complete")= logi TRUE
##  - attr(*, "validate")= logi TRUE
```

Here we can observe that there is e.g. flights from Istanbul to Johannesburg and Kuala Lumpur and a flight from Bergen to Kyoto.

Final Remarks

The prupose of this project paper was to give some inspiration to how map data from the *Google Maps API* can be used in order to visualize and analyze logistics networks and supply chains. Through the four problems, some useful R-code was displayed along with visualizations to give an idea about how problems can be solved with the use of lattitude and longitude data on maps.

References

https://www.rdocumentation.org/packages/ggmap/versions/3.0.0/ https://blog.dominodatalab.com/geographic-visualization-with-rs-ggmaps/

Disclaimer: This is not an academic or commerical project, rather a way to wrap up my personal research on my repositories at www.Github.com/TellSol.