

# Using the statistical language R as a Geographic Information System

---

## Easy Maps

Stefan Jünger / GESIS – Leibniz Institute for the Social Sciences

November 23, 2021

DOI: 10.5281/zenodo.5717830

# Now

Time	Title
09:00-09:30	Introduction: Data Management & Geospatial Data
09:30-09:35	Exercise 1: R Warm up
09:35-10:00	Data Processing & Spatial Linking
10:00-10:30	Exercise 2: Geospatial Data Wrangling
10:30-10:45	Break
10:45-11:15	Easy Maps
11:15-11:45	Excercise 3: Build your own map
11:45-12:00	Closing, Q & A

# Fun With Flags... MAPS, It's Maps!

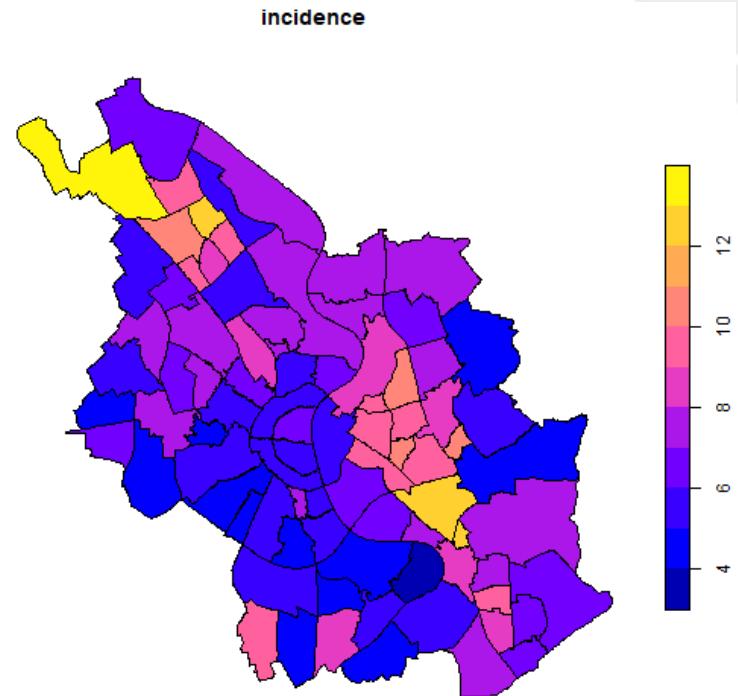


Fun with Flags by Dr. Sheldon Cooper. [Big Bang Theory](#)

# Fun With Maps

`plot()` does not allow us to manipulate the maps in an easy. But we already have the two most essential ingredients to create a nice map:

1. One vector layer like our Cologne Corona data.
2. Some interesting attributes linked with the geometries.



# What Makes a Good Map?

## Good Mapping

- reduction to most important information
- legends, scales, descriptions
- audience oriented
- adjusted for color vision deficiencies

## Bad Mapping

- overcrowding and overlapping
- unreadable information
- missing information like the legend or source
- poor choice of color palettes



# What Makes a Good Map?



Source

... but there is one other type:

## **The fast but nice map.**

- fast exploration of spatial data by visualizing the geometries and attributes
- might not be publication-ready yet, but they are more rewarding than just plotting information.

# The Choice Is Yours: R Packages for Mapping

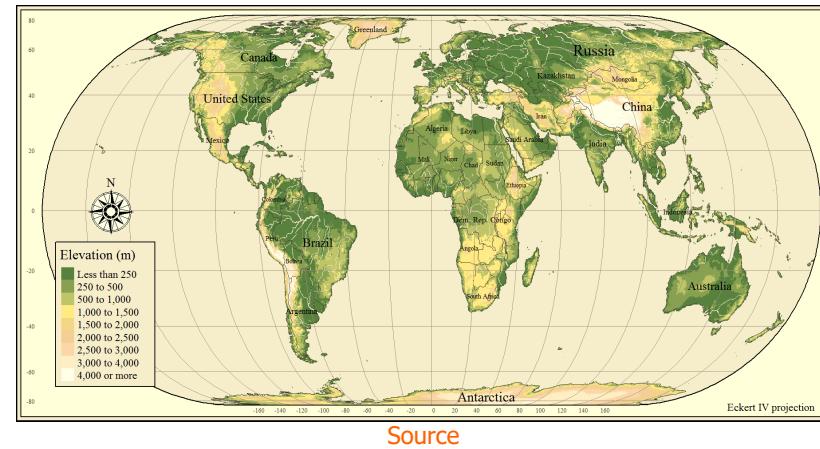
As always, R offers several ways to map spatial data, and the provided packages are various. What is out there? Just a few:

- base R graphics package: `mapdata`
- mobile-friendly interactive maps: `leaflet`
- interactive and static thematic maps based on shapefiles:
- `tmap`
- `mapview`

# Our Choice Today

Today, we'll concentrate on the package **tmap**

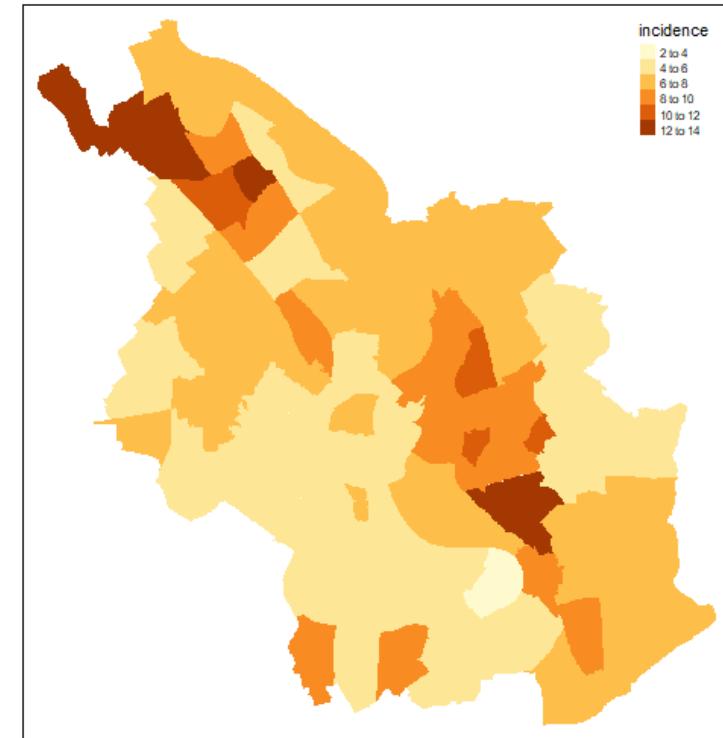
- very intuitive and makes "good" decisions for us
- the syntax is very similar to **ggplot2**\*



\*A wide-known 'secret' is that tmap creates a map based on ggplot2, so ggplot2-specific graphics manipulations will work as well.

# First Map: Our Corona Data (Again, but Prettier)

```
tm_shape(corona_cologne) +  
  tm_fill("incidence")
```



# tmap In a Nutshell

There is not much to consider when using tmap but essential two requirements:

1. Define your spatial object.
2. Choose a building block to determine how to display information.

```
# define and introduce every (new)
# geospatial data object
tm_shape() +  
  
# choose at least one building block as
# 'aesthetic layer'  
  
# for polygon layer choose from:
tm_fill() + # polygons without borders
tm_polygons() + # polygons with borders
tm_borders() + # only borders of polygons  
  
# for line layer choose:
tm_lines() +  
  
# for point layer choose:
tm_dots() +
tm_bubbles() +  
  
# for raster layer choose
tm_raster() +
tm_rgb() +  
  
...
# for all of them:
?'tmap-element'
```

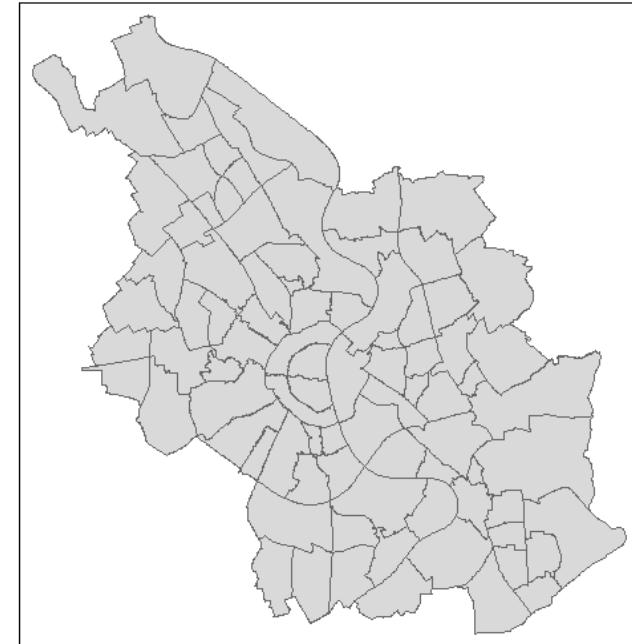
# tmap In a Nutshell: Polygon Layer

```
tm_shape(corona_cologne) +  
  tm_fill()
```



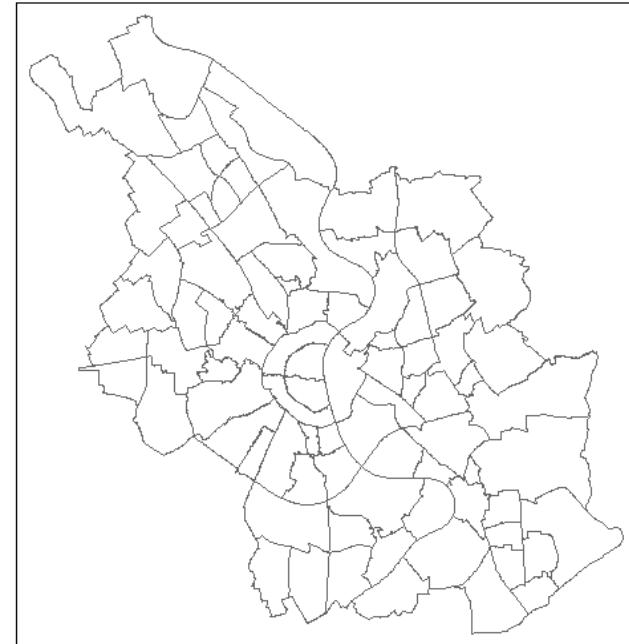
# tmap In a Nutshell: Polygon Layer

```
tm_shape(corona_cologne) +  
  tm_polygons()
```



# tmap In a Nutshell: Polygon Layer

```
tm_shape(corona_cologne) +  
  tm_borders()
```



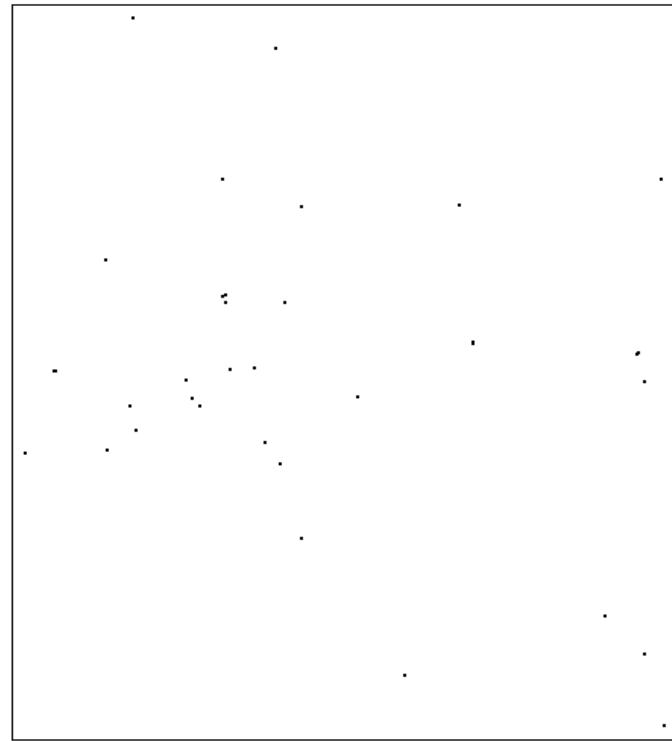
# tmap In a Nutshell: Line and Point Layer

```
tm_shape(streets_cologne) +  
  tm_lines()
```



# tmap In a Nutshell: Line and Point Layer

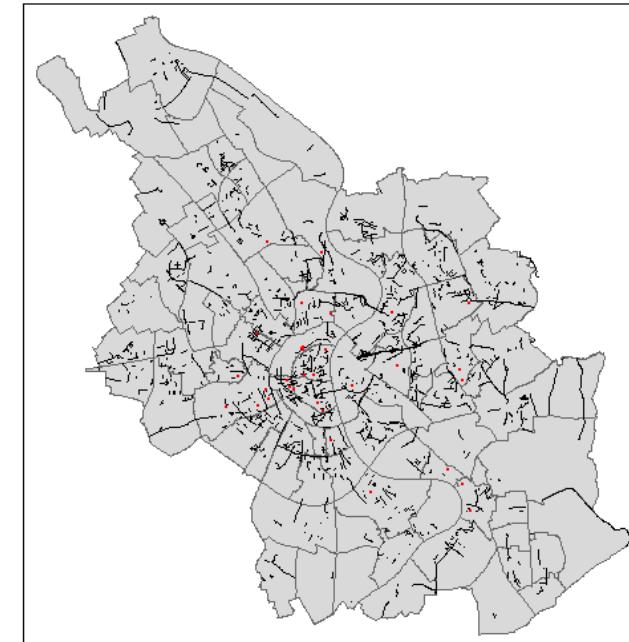
```
tm_shape(hospitals_cologne) +  
  tm_dots()
```



# tmap In a Nutshell: Put It All Together

We can map the geometric attributes as single layers, but we can also layer our map and stack the layers on each other.

```
tm_shape(corona_cologne) +  
  tm_polygons() +  
  tm_shape(streets_cologne) +  
  tm_lines() +  
  tm_shape(hospitals_cologne) +  
  tm_dots(col = "red")
```



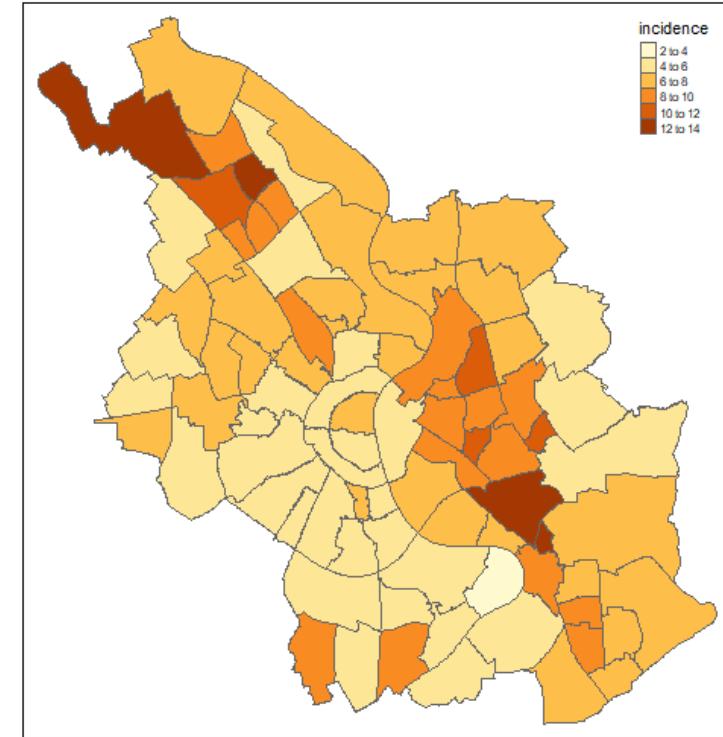
# Add Some Information

After we took care of our geometric types, we want to add some information to our data. The inner construction of each building block of `tm_elements` is the same.

1. Define the variable of interest first by stating the column name.
2. Add a name for legend title, color palette, adjust legend, scales ...

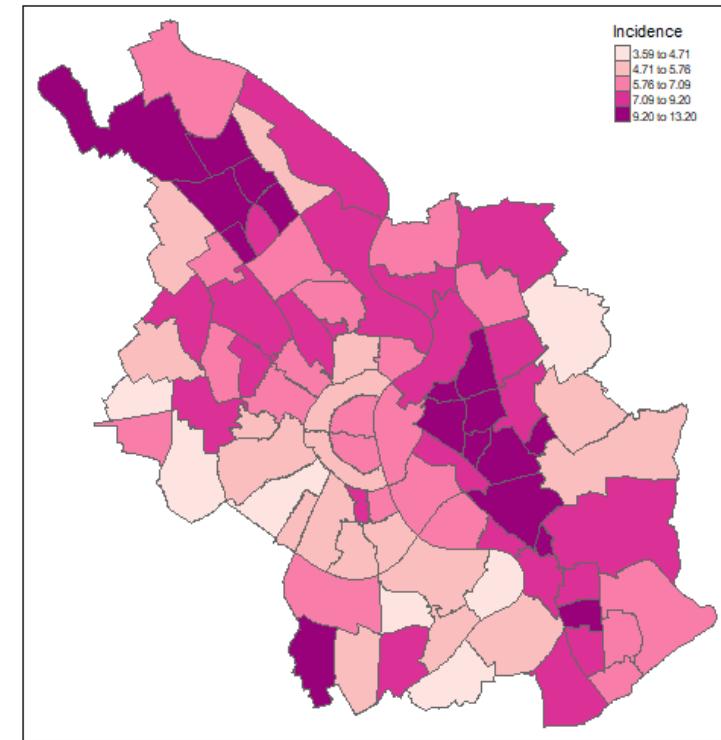
# Chosing an Attribute

```
tm_shape(corona_cologne) +  
  tm_polygons("incidence")
```



# Chosing a Color Palette

```
tm_shape(corona_cologne) +  
  tm_polygons(  
    "incidence",  
    palette = "RdPu",  
    title = "Incidence",  
    style = "kmeans"  
)
```



# Re-Placing the Legend

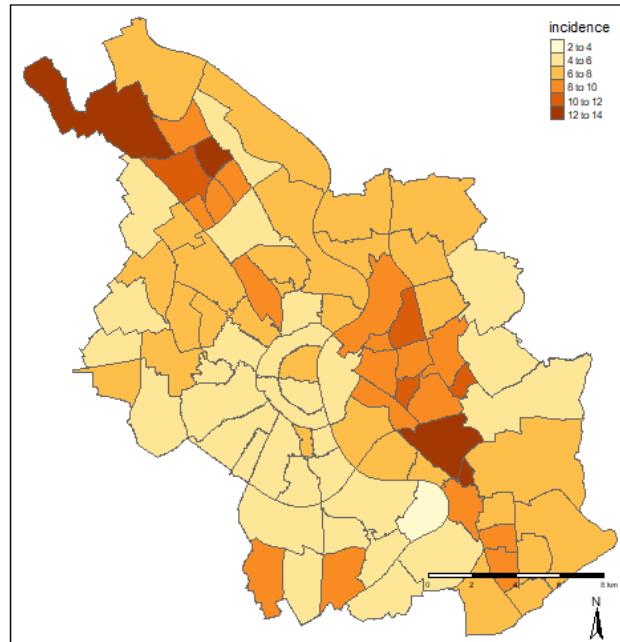
```
tm_shape(corona_cologne) +  
  tm_polygons(  
    "incidence",  
    palette = "RdPu",  
    title = "Incidence",  
    style = "kmeans"  
  ) +  
  tm_layout(  
    legend.outside = TRUE  
  )
```

# What's Left?

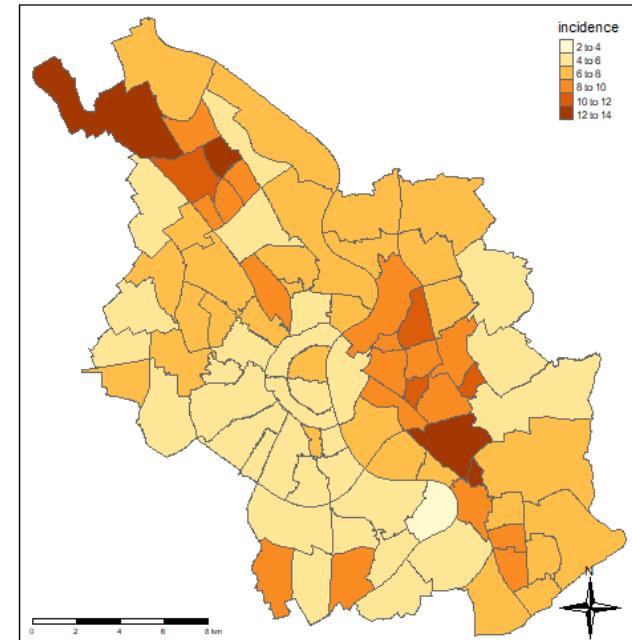


# Compasses & Scale Bars

```
tm_shape(corona_cologne) +  
  tm_polygons("incidence") +  
  tm_scale_bar() +  
  tm_compass()
```

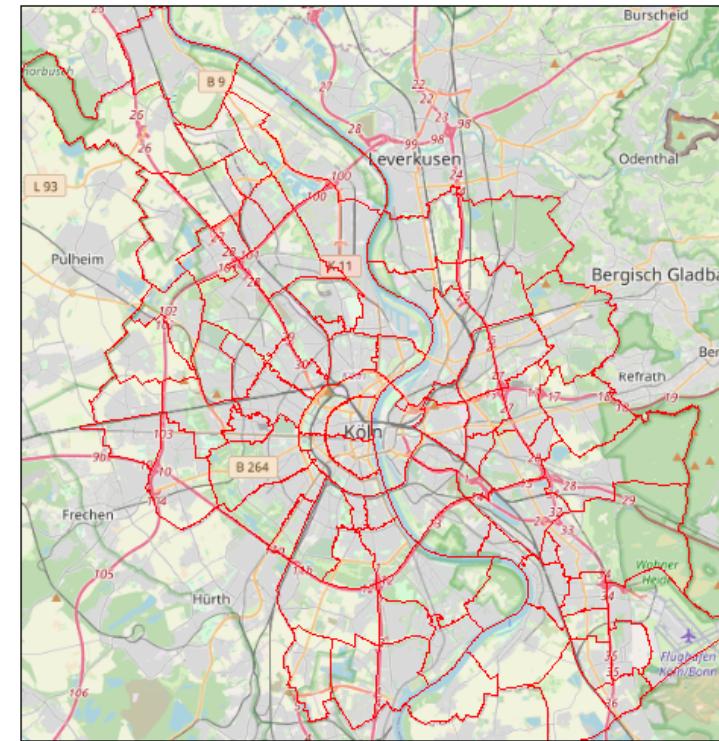


```
tm_shape(corona_cologne) +  
  tm_polygons("incidence") +  
  tm_scale_bar(position = "left") +  
  tm_compass(type = "4star")
```



# Add a Background From OpenStreetMap

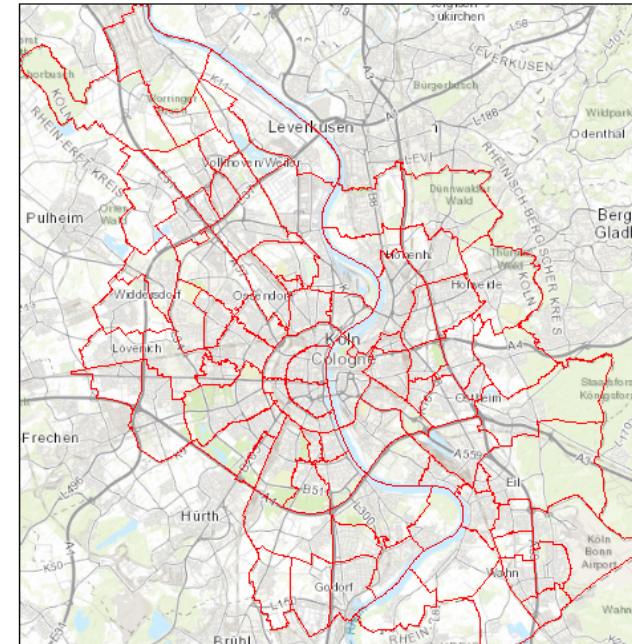
```
tmaptools::read_osm(  
  corona_cologne  
) %>%  
  tm_shape() +  
  tm_rgb() +  
  tm_shape(corona_cologne) +  
  tm_borders(col = "red")
```



# Playing With Different Map Types

Call `OpenStreetMap::getMapInfo()` for a complete list.

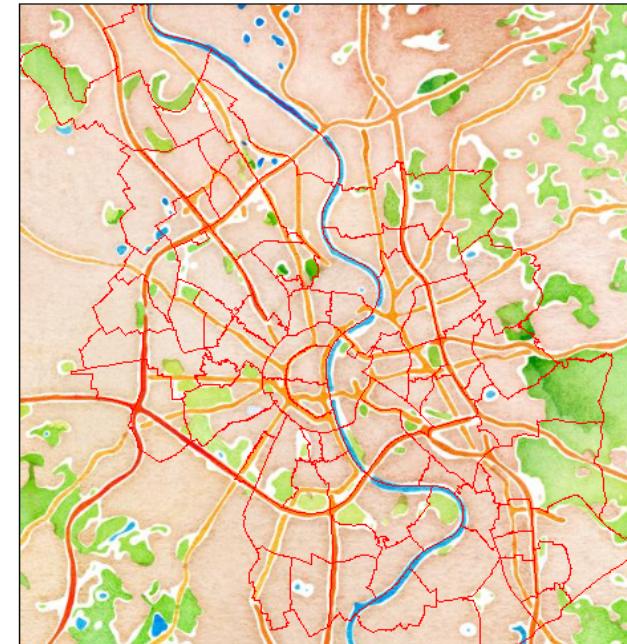
```
tmaptools::read_osm(  
  corona_cologne,  
  type = "esri-topo"  
) %>%  
  tm_shape() +  
  tm_rgb() +  
  tm_shape(corona_cologne) +  
  tm_borders(col = "red")
```



# Playing With Different Map Types

Call `OpenStreetMap::getMapInfo()` for a complete list.

```
tmaptools::read_osm(  
  corona_cologne,  
  type = "stamen-watercolor"  
) %>%  
  tm_shape() +  
  tm_rgb() +  
  tm_shape(corona_cologne) +  
  tm_borders(col = "red")
```



# Note On Mapping Responsible

**In the best cases**, maps are easy to understand and an excellent way to transport (scientific) messages.

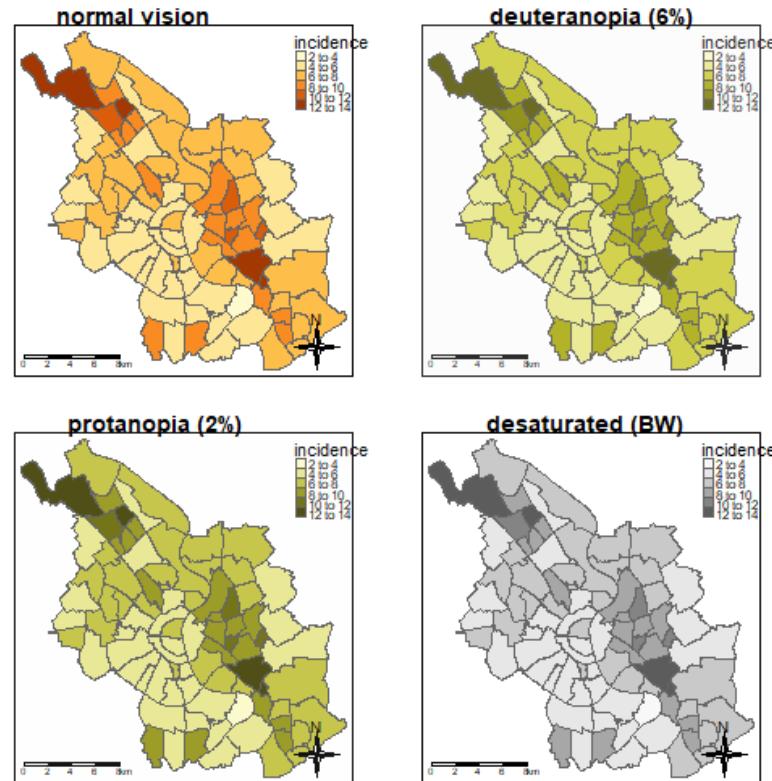
**In the worst cases**, they simplify (spurious) correlations and draw a dramatic picture of the world.

## Maps can shape narratives

- Decisions on which projection you use (remember the true size projector?),
- the segment of the world you choose,
- and the colors you add have a strong influence.

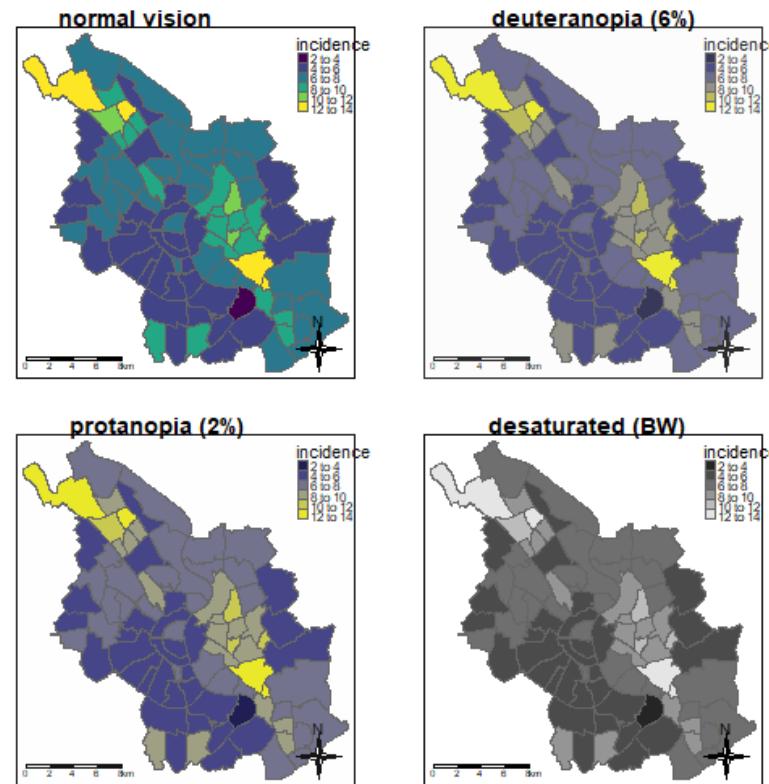
Example: [Kenneth Field's blog post](#)

# Color Vision Deficiencies: Standard Palette



Created with the package [colorBlindness](#)

# Color Vision Deficiencies: Viridis Palette



Created with the package [colorBlindness](#)

# More Resources

If you want to dive deeper into mapping with tmap check out:

- Introduction by [Thomas Lo Russo](#)
- Blogpost by [Hollie Zevross](#)

And if you want to get some inspiration, keep an eye out for the #30DayMapChallenge on Twitter. Repository of Last Year's Challenge [here](#).



# Exercise 3: Build Your Own Map

Exercise

Solution



✉ [stefan.juenger@gesis.org](mailto:stefan.juenger@gesis.org)  
🐦 [@StefanJuenger](https://twitter.com/StefanJuenger)  
🗣 [StefanJuenger](https://github.com/StefanJuenger)  
🏡 <https://stefanjuenger.github.io>



🐦 @CESSDA\_Data



Licence: CC-BY 4.0