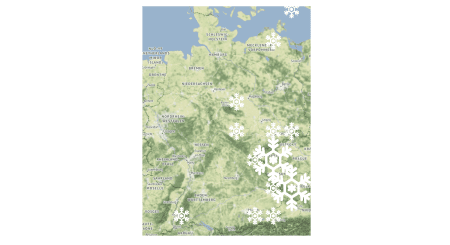
And what better way to get our hopes up, than by taking a look at the [DWD Climate Data Center’s](https://www.dwd.de/DE/klimaumwelt/cdc/cdc_node.html) historic data on the snow depth on the past ten Christmas Eves?

But how to best visualize spatial data? Other than most data types, spatial data usually calls for a very particular visualization, namely data points overlaying a map. In this way, areal data is automatically contextualized by the geographic information intuitively conveyed by a map.

The basic functionality of ggplot2 dosen’t offer the possibility to do so, but there is a package akin to ggplot2 that allows to do so: ggmap. ggmap was written by David Kahle and Hadley Wickham and combines the building blocks of ggplot2, the grammar of graphics as well as the static maps of Google Maps, OpenStreetMap, Stamen Maps or CloudMade Maps. And with all that, ggmap allows us to make really fancy visualizations:



Above-average snow depth on Christmas Eve (2008-2017)

The original functionalities of ggmap used to be somewhat more general, broad and “barrier-free”, but since those good old days – aka 2013 – some of the map suppliers changed the terms of use as well as mechanics of their APIs. At the moment, the service of Stamen Maps seems to be the most stable, while also being easily accessible – e.g. without registering for an API that requires one to provide some payment information. Therefore, we’re going to focus on Stamen Maps.

**First things first: the map**

Conveniently, ggmap employs the same theoretical framework and general syntax as ggplot2. However, ggmap requires one additional step: Before we can start plotting, we have to download a map as backdrop for our visualization. This is done with get\_stamenmap(), get\_cloudmademap(), get\_googlemap() or get\_openstreetmap() or the more general get\_map(). We’re going to use get\_stamenmap().

To determine the depicted map cutout, the left, bottom, right and top coordinates of a bounding box, have to be supplied to the argument bbox.  
Conveniently, there is no need to know the exact latitudes and longitudes of each and every bounding box of interest. The function geocode\_OSM() from the package tmaptools, returns whenever possible the coordinates of a search query consisting of an address, zip code and/or name of a city or country.

library(scales)

library(tidyverse)

library(tmaptools)

library(ggimage)

library(ggmap)

# get the bounding box

geocode\_OSM("Germany")$bbox

xmin ymin xmax ymax

5.866315 47.270111 15.041932 55.099161

The zoom level can be set via the zoom argument and can range between 0 (least detailed) and 18 (most detailed, quick disclaimer: this can take a very long time). The zoom level determines the resolution of the image as well as the amount of displayed annotations.

Depending on whether we want to highlight roads, political or administrative boundaries or bodies of water and land different styles of maps excel. The maptype argument allows to choose from different ready-made styles: "terrain", "terrain-background", "terrain-labels", "terrain-lines", "toner", "toner-2010", "toner-2011", "toner-background", "toner-hybrid", "toner-labels", "toner-lines", "toner-lite" or "watercolor".

Some further, very handy arguments of get\_stamenmap() are crop, force and color:  
As implied by the name, color defines whether a map should be in black-and-white ("bw") or when possible in color ("color").

Under the hood get\_stamenmap() downloads map tiles, which are joined to the complete map. If the map tiles should be cropped so as to only depict the specified bounding box, the crop argument can be set to TRUE.

Unless the force argument is set to TRUE, even when arguments changing the style of a map have been altered, once a map of a given location has been downloaded it will not be downloaded again.

When we’ve obtained the map of the right location and style, we can store the “map image” in an object or simply pass it along to ggmap() to plot it. [The labels, ticks etc. of axes can be controlled as usual](https://www.statworx.com/blog/using-themes-in-ggplot2/).

# getting map

plot\_map\_z7 <- get\_stamenmap(as.numeric(geocode\_OSM("Germany")$bbox),

zoom = 7,

force = TRUE,

maptype = "terrain")

# saving plotted map alone

plot1 <- ggmap(plot\_map\_z7) +

theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank())

# getting map

plot\_map\_z5 <- get\_stamenmap(as.numeric(geocode\_OSM("Germany")$bbox),

zoom = 5,

force = TRUE,

maptype = "terrain")

# saving plotted map alone

plot2 <- ggmap(plot\_map\_z5) +

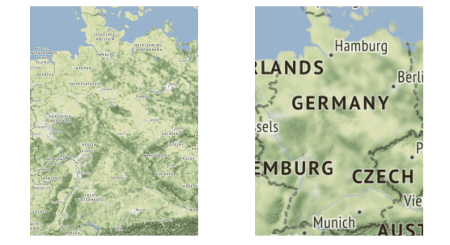
theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank())

# plotting maps together

plot <- gridExtra::grid.arrange(plot1, plot2, nrow = 1)



Example for maptype = „terrain“ with zoom = 7 (left) vs. zoom = 5 (right).

**Business as usual: layering geoms on top**

We then can layer any ggplot2 geom we’d like on top of that map, with the only requirement being that the variables mapped to the axes are within the same numeric range as the latitudes and longitudes of the depicted map. We also can use many extension packages building on ggplot2. For example, we can use the very handy package ggimage by Guangchuang Yu to make our plots extra festive:

# aggregating data per coordinate

df\_snow\_agg <- df\_snow %>%

dplyr::mutate(LATITUDE = plyr::round\_any(LATITUDE, accuracy = 1),

LONGITUDE = plyr::round\_any(LONGITUDE, accuracy = 1)) %>%

dplyr::group\_by(LATITUDE, LONGITUDE) %>%

dplyr::summarise(WERT = mean(WERT, na.rm = TRUE))

# cutting into equal intervals

df\_snow\_agg$snow <- as.numeric(cut(df\_snow\_agg$WERT, 12))

# setting below average snow depths to 0

df\_snow\_agg <- df\_snow\_agg %>%

mutate(snow = ifelse(WERT <= mean(df\_snow\_agg$WERT), 0, snow))

# adding directory of image to use

df\_snow\_agg$image <- "Snowflake\_white.png"

# getting map

plot\_map <- get\_stamenmap(as.numeric(geocode\_OSM("Germany")$bbox),

zoom = 7,

force = TRUE,

maptype = "terrain")

# plotting map + aggregated snow data with ggimage

plot <- ggmap(plot\_map) +

geom\_image(data = df\_snow\_agg,

aes(x = LONGITUDE,

y = LATITUDE,

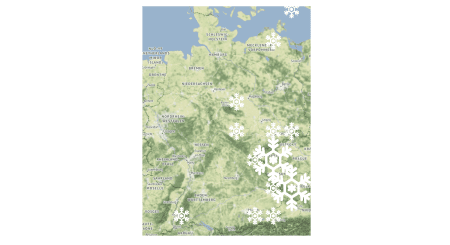
image = image,

size = I(snow/20))) + # rescaling to get valid size values

theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank())



Above-average snow depth on Christmas Eve (2008-2017)

We can get creative, playing with all sorts of geoms and map styles. Especially visualization strategies to prevent [overplotting](https://www.statworx.com/blog/how-not-to-overplot/) can be quite handy:

For example, a facetted scatter plot with alpha blending:

# getting map

plot\_map <- get\_stamenmap(as.numeric(geocode\_OSM("Germany")$bbox),

zoom = 7,

force = TRUE,

maptype = "watercolor")

# facets per year, scatter plot with alpha blending

plot <- ggmap(plot\_map) +

geom\_point(data = df\_snow,

aes(x = LONGITUDE,

y = LATITUDE,

alpha = WERT,

size = WERT),

color = "white") +

facet\_wrap(. ~ DATE, nrow = 2) +

scale\_alpha\_continuous(range = c(0.0, 0.9)) +

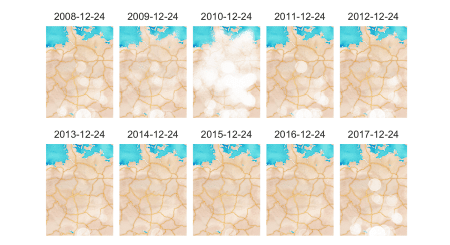
theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank(),

legend.position = "none",

strip.background = element\_rect(fill = "white"))



Snow depth on Christmas Eve per year (2008-2017)

Or density plots with either geom\_line() or geom\_polygon():

# getting map

plot\_map <- get\_stamenmap(as.numeric(geocode\_OSM("Germany")$bbox),

zoom = 7,

crop = FALSE,

force = TRUE,

maptype = "toner-background")

# density plot: lines

plot1 <- ggmap(plot\_map) +

geom\_density2d(data = df\_snow\_dens,

aes(x = LONGITUDE,

y = LATITUDE,

color = ..level..),

size = 0.7) +

scale\_color\_continuous(low = "#17775A", high = "white",

name = "",

breaks = c(0.01, 0.06),

labels = c("little snow", "much snow")) +

theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank(),

legend.position = "left")

# density plot: polygon

plot2 <- ggmap(plot\_map) +

stat\_density\_2d(data = df\_snow\_dens,

aes(x = LONGITUDE, y = LATITUDE,

alpha = ..level..),

fill = "#CFECFF",

geom = "polygon") +

scale\_alpha(name = "",

breaks = c(0.012, 0.024, 0.036, 0.048, 0.06),

labels = c("little snow", "","", "", "much snow")) +

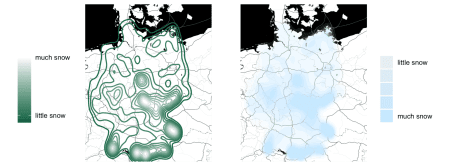
theme(axis.title = element\_blank(),

axis.ticks = element\_blank(),

axis.text = element\_blank())

plot <- gridExtra::grid.arrange(plot1, plot2, nrow = 1)

ggsave(plot = plot, filename = "density.png", width = 5.5, height = 3)



Average snow depth on Christmas Eve (2008-2017)

Our plots give a first taste of what can be done with ggmap. However, based on the historic data, there won’t be a white Christmas for the most of us… Against all odds, I still hope that outside there’s a winter wonderland waiting for you just now.

From everybody here at STATWORX happy holidays and a happy new year!

**References**

* D. Kahle and H. Wickham. ggmap: Spatial Visualization with ggplot2. The R Journal, 5(1), 144-161. <http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf>
* DWD Climate Data Center (CDC): Tägliche Stationsmessungen Schneehöhe in cm für Deutschland, Version v18.3 & recent, abgerufen am 16.12.2018.
* Map tiles by [Stamen Design](https://stamen.com/), under [CC BY 3.0.](https://creativecommons.org/licenses/by/3.0/) Data by [OpenStreetMap](https://www.statworx.com/de/blog/dreaming-of-a-white-christmas-with-ggmap-in-r/), under [CC BY SA](https://creativecommons.org/licenses/by-sa/3.0/) respectively under [ODbL](https://www.openstreetmap.org/copyright).