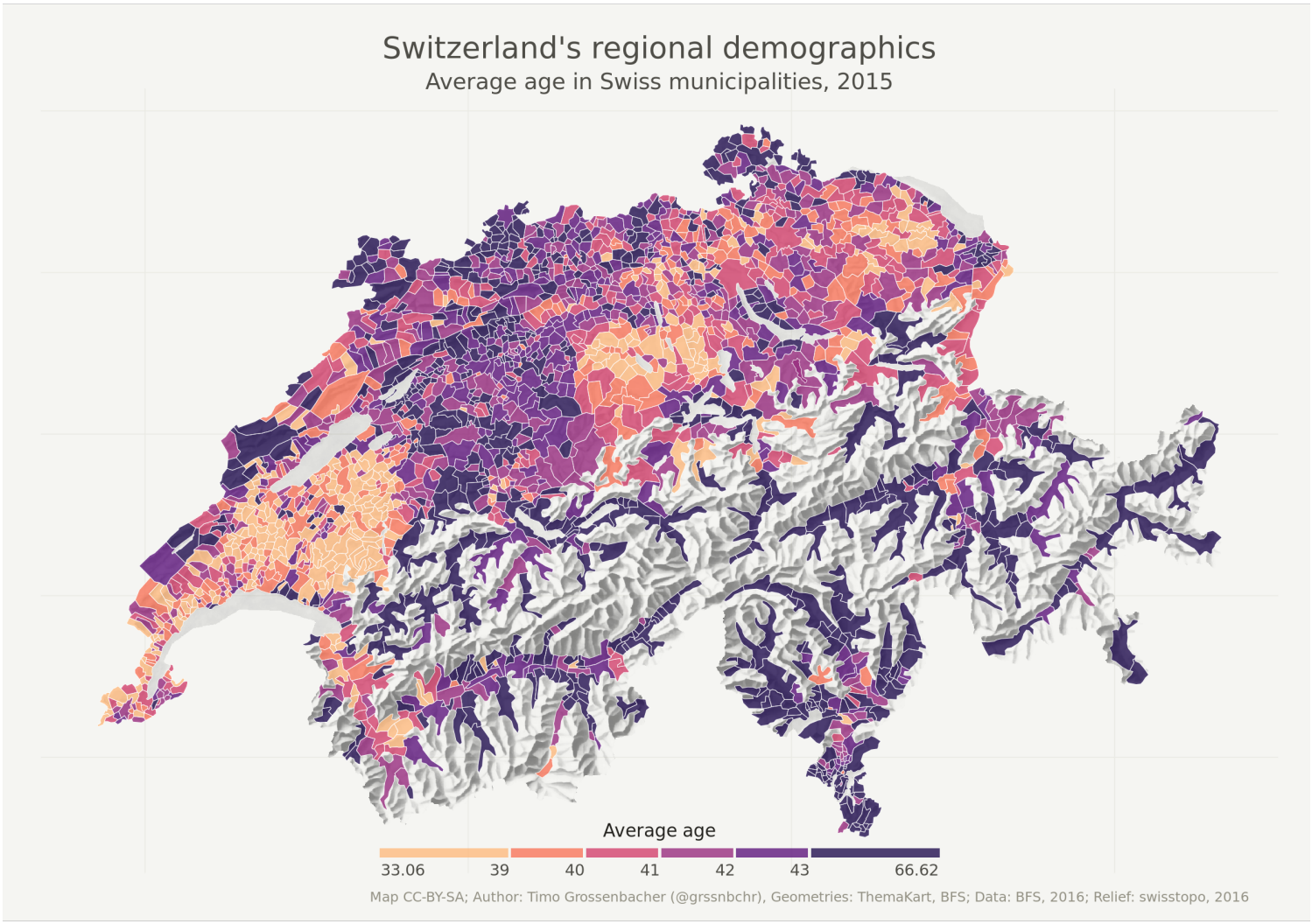
Timo Grossenbacher



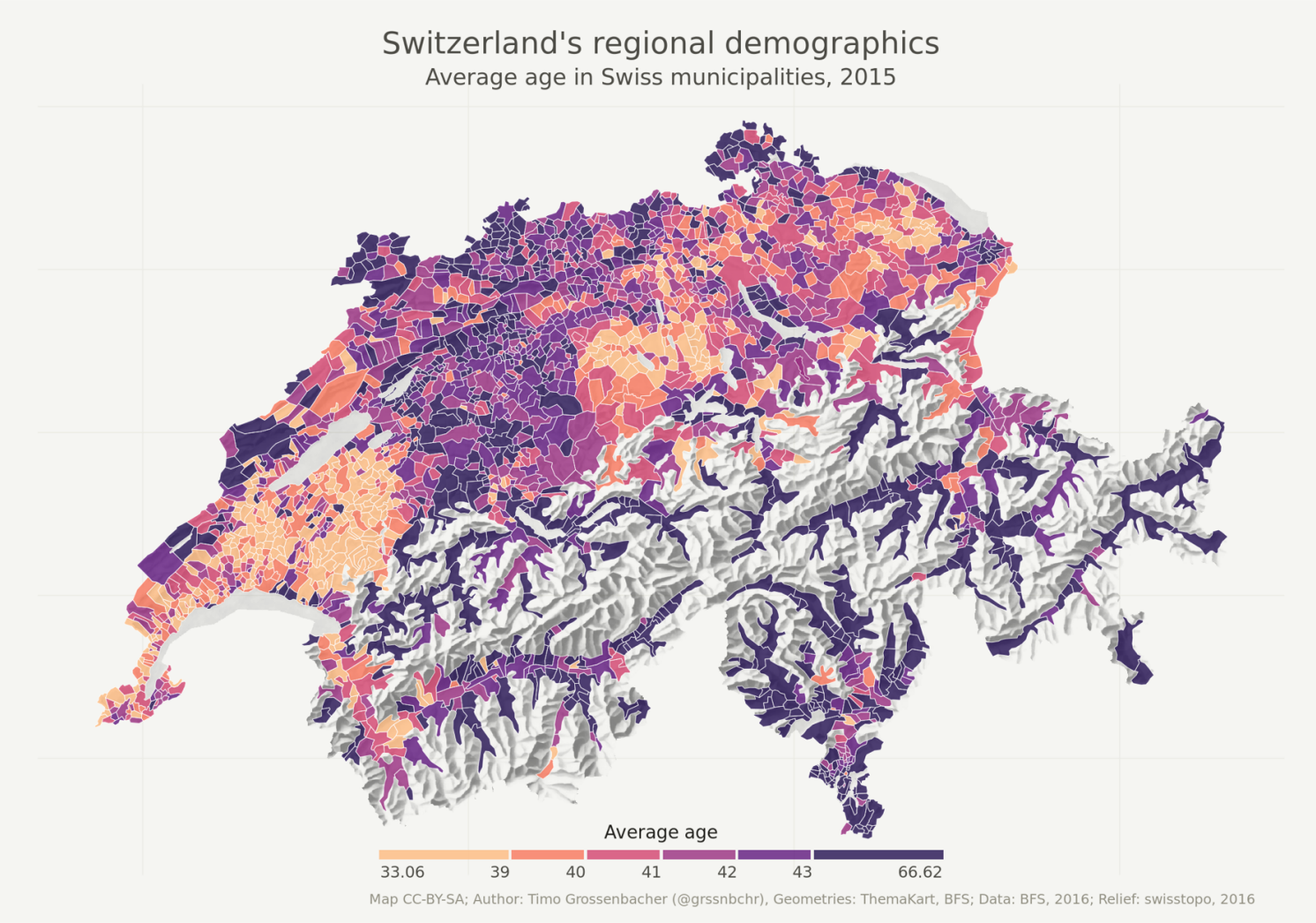

December 26, 2016 / comments 60

# Beautiful thematic maps with ggplot2 (only)

**Important update** (2019-04-19): This blog post still attracts dozens of readers each day – which is great! However, the methodology is now outdated. Please read my new blog post on producing such (and other) maps with even easier methods.

- Beautiful thematic maps with ggplot2 (only)

# Beautiful thematic maps with ggplot2 (only)




The above *choropleth* was created with `ggplot2` (2.2.0) only. Well, almost. Of course, you need the usual suspects such as `rgdal` and `rgeos` when dealing with geodata, and `raster` for the relief. But apart from that: nothing fancy such as `ggmap` or the like. The imported packages are kept to an absolute minimum.

In this blog post, I am going to explain step by step how I (eventually) achieved this result – from a very basic, useless, ugly, default map to the publication-ready and (in my opinion) highly aesthetic choropleth.

## Reproducibility

As always, you can reproduce, reuse and remix everything you find here, just go to this repository and clone it. All the needed input files are in the `input` folder, and the main file to execute is `index.Rmd`. Right now, knitting it produces an `index.md` that I use for my blog post on timogrossenbacher.ch, but you can adapt the script to produce an HTML file, too. The PNGs produced herein are saved to `wp-content/uploads/2016/12` so I can display them directly in my blog, but of course you can also adjust this.

# Preparations

## Clear workspace and install necessary packages

This is just my usual routine: Detach all packages, remove all variables in the global environment, etc, and then load the packages. Saves me a lot of headaches.

```
knitr::opts_chunk$set(
    out.width = "100%",
    dpi = 300,
    fig.width = 8,
    fig.height = 6,
    fig.path = 'https://timogrossenbacher.ch/wp-content/uploads/2016/12/tm-',
    strip.white = T,
    dev = "png",
    dev.args = list(png = list(bg = "transparent"))
)

remove(list = ls(all.names = TRUE))

detachAllPackages <- function() {
  basic.packages.blank <-  c("stats",
                             "graphics",
                             "grDevices",
                             "utils",
                             "datasets",
                             "methods",
                             "base")
  basic.packages <- paste("package:", basic.packages.blank, sep = "")

  package.list <- search()[ifelse(unlist(gregexpr("package:", search())) == 1,
                                  TRUE,
```

```
                                    FALSE)]

  package.list <- setdiff(package.list, basic.packages)

  if (length(package.list) > 0)  for (package in package.list) {
    detach(package, character.only = TRUE)
    print(paste("package ", package, " detached", sep = ""))
  }
}

detachAllPackages()


if (!require(rgeos)) {
  install.packages("rgeos", repos = "http://cran.us.r-project.org")
  require(rgeos)
}
if (!require(rgdal)) {
  install.packages("rgdal", repos = "http://cran.us.r-project.org")
  require(rgdal)
}
if (!require(raster)) {
  install.packages("raster", repos = "http://cran.us.r-project.org")
  require(raster)
}
if(!require(ggplot2)) {
  install.packages("ggplot2", repos="http://cloud.r-project.org")
  require(ggplot2)
}
if(!require(viridis)) {
  install.packages("viridis", repos="http://cloud.r-project.org")
  require(viridis)
}
if(!require(dplyr)) {
  install.packages("dplyr", repos = "https://cloud.r-project.org/")
  require(dplyr)
}
if(!require(gtable)) {
  install.packages("gtable", repos = "https://cloud.r-project.org/")
  require(gtable)
}
if(!require(grid)) {
  install.packages("grid", repos = "https://cloud.r-project.org/")
  require(grid)
}
if(!require(readxl)) {
  install.packages("readxl", repos = "https://cloud.r-project.org/")
```

```
  require(readxl)
}
if(!require(magrittr)) {
  install.packages("magrittr", repos = "https://cloud.r-project.org/")
  require(magrittr)
}
```

## General ggplot2 theme for map

First of all, I define a generic theme that will be used as the basis for all of the following steps. It's based on `theme_minimal` and basically resets all the axes. It also defined a very subtle grid and a warmgrey background, which gives it some sort of paper map feeling, I find.

The font used here is `Ubuntu Regular` – adapt to your liking, but the font must be installed on your OS.

```
theme_map <- function(...) {
  theme_minimal() +
  theme(
    text = element_text(family = "Ubuntu Regular", color = "#22211d"),
    axis.line = element_blank(),
    axis.text.x = element_blank(),
    axis.text.y = element_blank(),
    axis.ticks = element_blank(),
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    # panel.grid.minor = element_line(color = "#ebebe5", size = 0.2),
    panel.grid.major = element_line(color = "#ebebe5", size = 0.2),
    panel.grid.minor = element_blank(),
    plot.background = element_rect(fill = "#f5f5f2", color = NA),
    panel.background = element_rect(fill = "#f5f5f2", color = NA),
    legend.background = element_rect(fill = "#f5f5f2", color = NA),
    panel.border = element_blank(),
    ...
  )
}
```

## Data sources

For this choropleth, I used **three** data sources:

- Thematic data: Average age per municipality as of end of 2015. The data is freely available from The Swiss Federal Statistical Office (FSO) and included in the `input` folder.
- Municipality geometries: The geometries do not show the political borders of Swiss municipalities, but the so-called "productive" area, i.e., larger lakes and other "unproductive" areas such as mountains are excluded. This has two advantages: 1) The relatively sparsely populated but very large municipalities in the Alps don't have too much visual weight and 2) it allows us to use the beautiful raster relief of the Alps as a background. The data are also from the FSO, but not freely available. You could also use the freely available political boundaries of course. I was allowed to republish the Shapefile for this educational purpose (also included in the `input` folder). Please stick to that policy.
- Relief: This is a freely available GeoTIFF from The Swiss Federal Office of Topography (swisstopo).

## Read in data and preprocess

```
data <- read.csv("input/avg_age_15.csv", stringsAsFactors = F)
```

## Read in geodata

Here, the geodata is loaded using `rgeos` / `rgdal` standard procedures. It is then *"fortified"*, i.e. transformed into a ggplot2-compatible data frame (the `fortify`-function is part of `ggplot2`). Also, the thematic data is joined using the `bfs_id` field (each municipality has a unique one).

```
gde_15 <- readOGR("input/geodata/gde-1-1-15.shp", layer = "gde-1-1-15")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "input/geodata/gde-1-1-15.shp", layer: "gde-1-1-15"
## with 2324 features
## It has 2 fields
```

```
# set crs to ch1903/lv03, just to make sure  (EPSG:21781)
crs(gde_15) <- "+proj=somerc +lat_0=46.95240555555556
+lon_0=7.439583333333333 +k_0=1 +x_0=600000 +y_0=200000
+ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no_defs"
# fortify, i.e., make ggplot2-compatible
map_data_fortified <- fortify(gde_15, region = "BFS_ID") %>%
  mutate(id = as.numeric(id))
# now we join the thematic data
map_data <- map_data_fortified %>% left_join(data, by = c("id" = "bfs_id"))

# whole municipalities
gde_15_political <- readOGR("input/geodata/g1g15.shp", layer = "g1g15")
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "input/geodata/g1g15.shp", layer: "g1g15"
## with 2328 features
## It has 20 fields
```

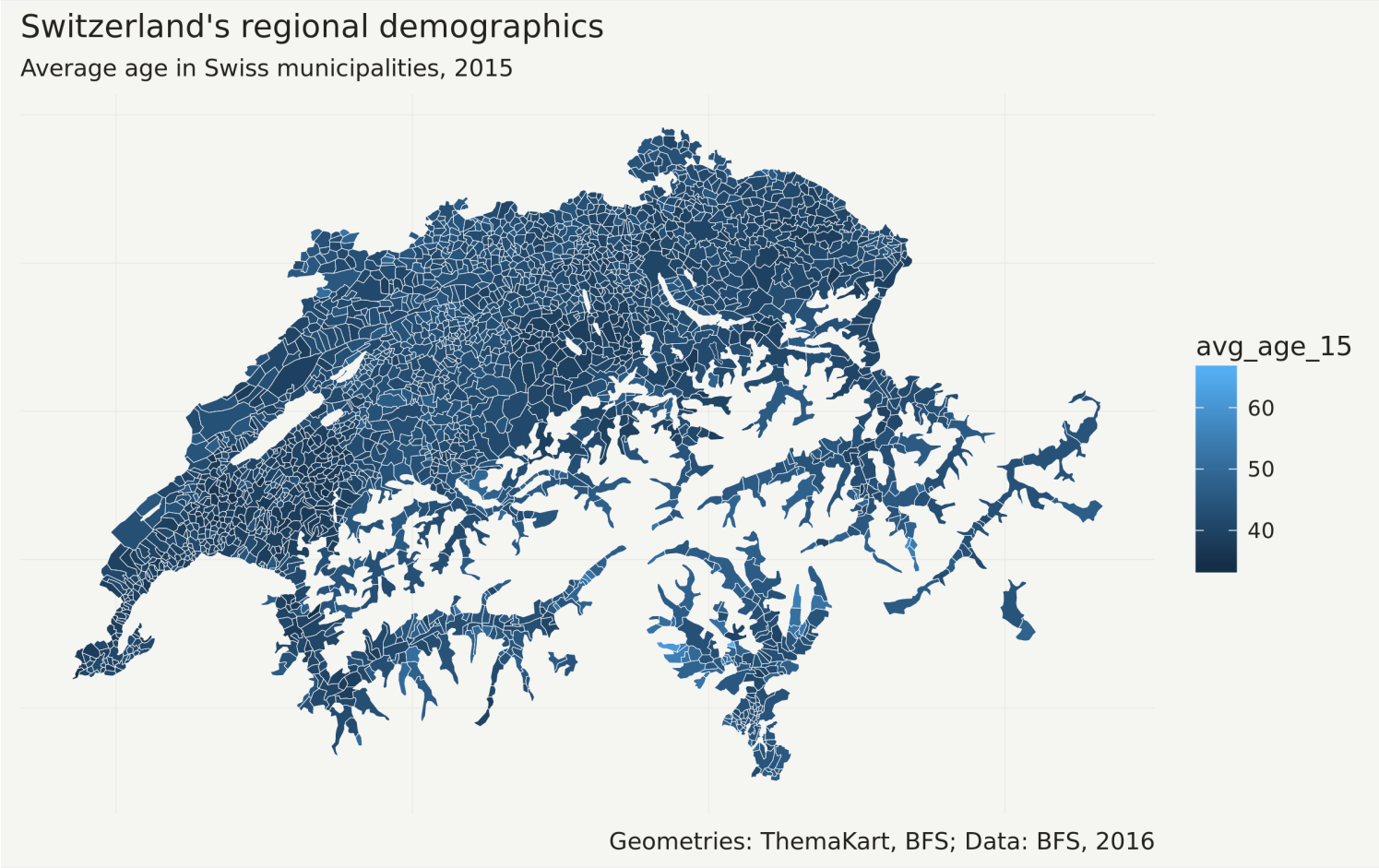
```
crs(gde_15_political) <- "+proj=somerc +lat_0=46.95240555555556
+lon_0=7.439583333333333 +k_0=1 +x_0=600000 +y_0=200000
+ellps=bessel +towgs84=674.374,15.056,405.346,0,0,0,0 +units=m +no_defs"
map_data_political_fortified <- fortify(gde_15_political, region = "GMDNR") %>%
  mutate(id = as.numeric(id))
map_data_political <- map_data_political_fortified %>% left_join(data, by = c("id" = "b
map_data_political <- map_data_political[complete.cases(map_data_political),]
# read in background relief
relief <- raster("input/geodata/02-relief-georef-clipped-resampled.tif")
relief_spdf <- as(relief, "SpatialPixelsDataFrame")
# relief is converted to a very simple data frame,
# just as the fortified municipalities.
# for that we need to convert it to a
# SpatialPixelsDataFrame first, and then extract its contents
# using as.data.frame
relief <- as.data.frame(relief_spdf) %>%
  rename(value = `X02.relief.georef.clipped.resampled`)
# remove unnecessary variables
rm(relief_spdf)
rm(gde_15)
rm(map_data_fortified)
rm(map_data_political_fortified)
```

# A very basic map

What follows now is a very basic map with the municipalities rendered with `geom_polygon` and their outline with `geom_path`. I don't even define a color scale here, it just uses ggplot2's default continuous color scale, because `avg_age_15` is a continuous variable.

Because the geodata are in a projected format, it is important to use `coord_equal()` here, if not, Switzerland would be distorted.

```
p <- ggplot() +
    # municipality polygons
    geom_polygon(data = map_data, aes(fill = avg_age_15,
                                      x = long,
                                      y = lat,
                                      group = group)) +
    # municipality outline
    geom_path(data = map_data, aes(x = long,
                                   y = lat,
                                   group = group),
              color = "white", size = 0.1) +
    coord_equal() +
    # add the previously defined basic theme
    theme_map() +
    labs(x = NULL,
         y = NULL,
         title = "Switzerland's regional demographics",
         subtitle = "Average age in Swiss municipalities, 2015",
         caption = "Geometries: ThemaKart, BFS; Data: BFS, 2016")
p
```

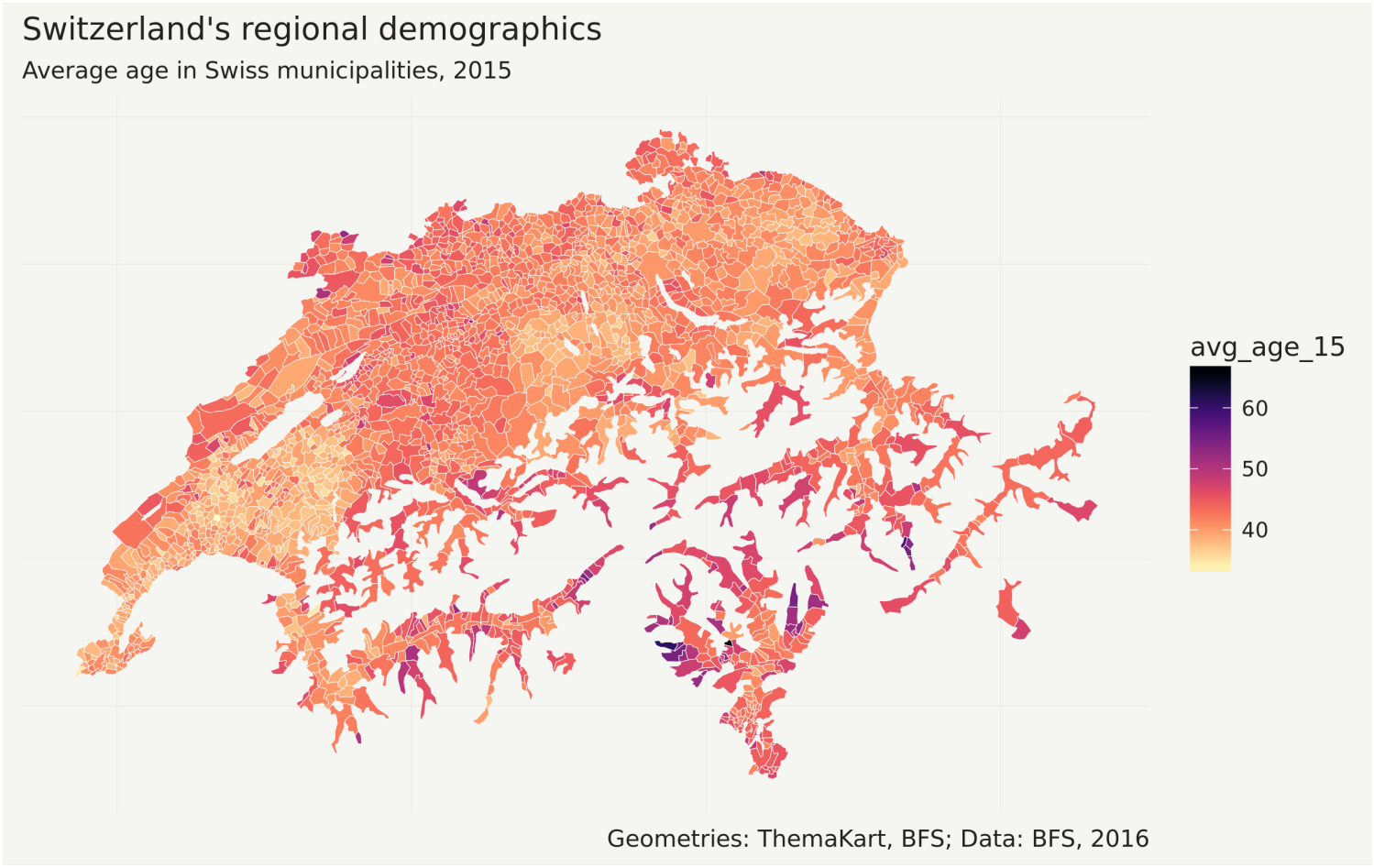



How ugly! The color scale is not very sensitive to the data at hand, i.e., regional patterns cannot be detected at all.

## A better color scale

See how I reuse the previously defined `p`-object and just add the continuous `viridis` scale from the same named package. All of a sudden the map looks more aesthetic and regional patterns are already visible in this linear scale. For example one can see that the municipalities in the south and in the Alps (where there are a lot of gaps, the unproductive areas I talked about) seem to have an older-than-average population (mainly because young people move to the cities for work etc.).

```
q <- p + scale_fill_viridis(option = "magma", direction = -1)
q
```

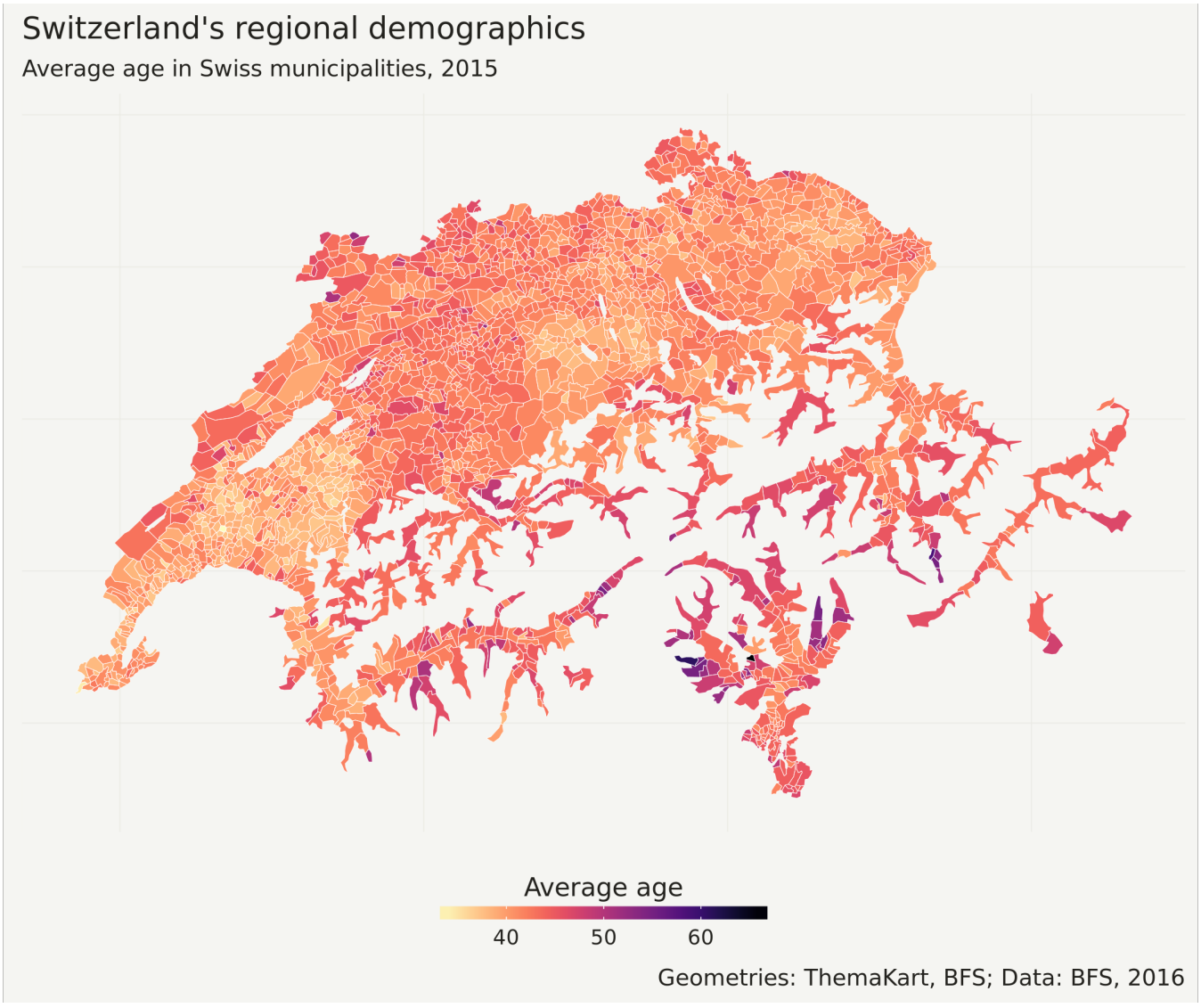



## Horizontal legend

Also I think one could save some space by using a horizontal legend at the bottom of the plot.

```
q <- p +
  # this is the main part
  theme(legend.position = "bottom") +
  scale_fill_viridis(
    option = "magma",
    direction = -1,
    name = "Average age",
```

```
    # here we use guide_colourbar because it is still a continuous scale
    guide = guide_colorbar(
      direction = "horizontal",
      barheight = unit(2, units = "mm"),
      barwidth = unit(50, units = "mm"),
      draw.ulim = F,
      title.position = 'top',
      # some shifting around
      title.hjust = 0.5,
      label.hjust = 0.5
  ))
q
```



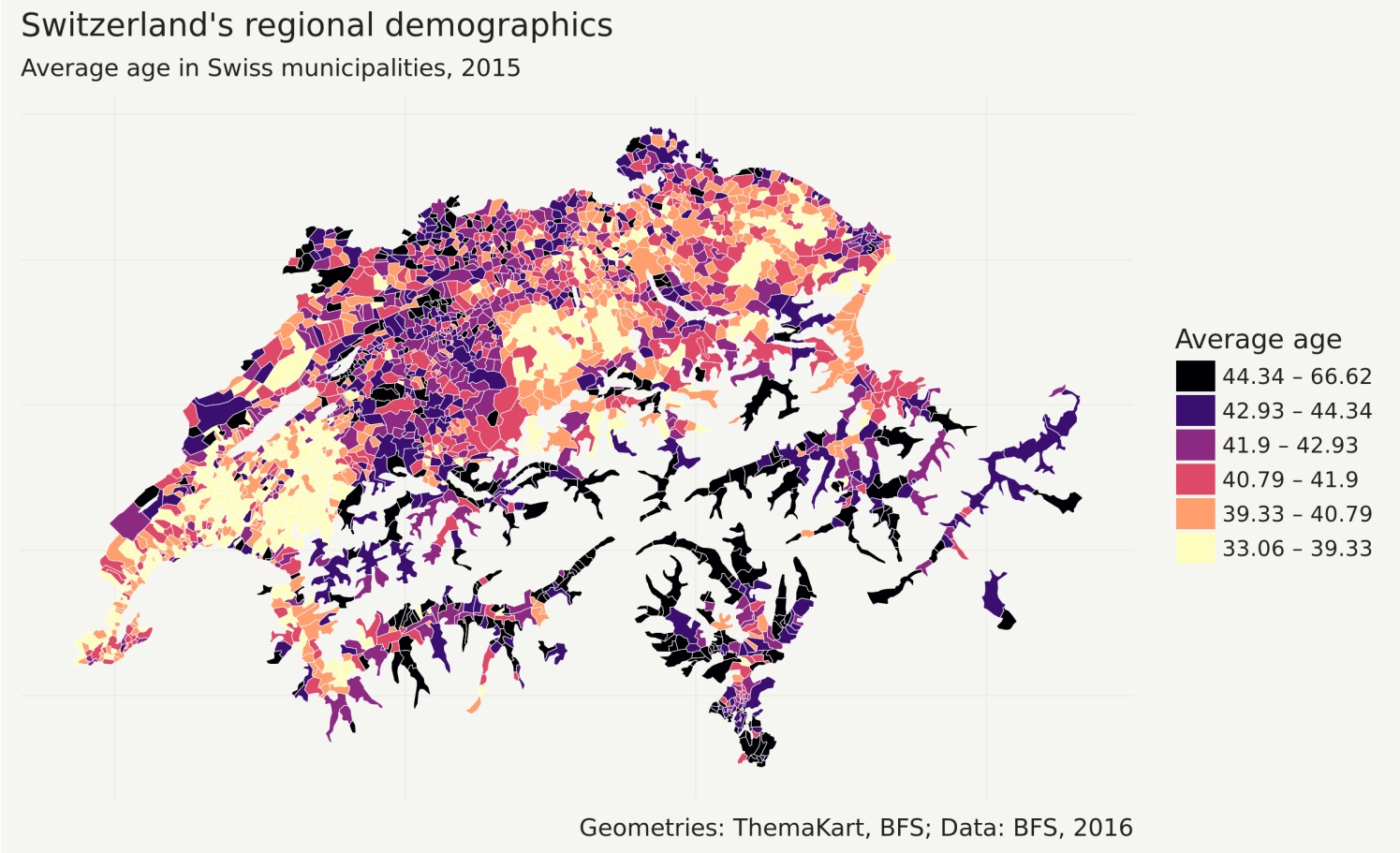

Well, the plot now has a weird aspect ratio, but okay...

## Discrete classes with quantile scale

I am still not happy with the color scale because I think regional patterns could be made more clearly visible. For that I break up the continuous `avg_age_15` variable into 6 quantiles (remember your statistics class?). The effect of that is that I now have about the same number of municipalities in each class.

```
no_classes <- 6
labels <- c()

quantiles <- quantile(map_data$avg_age_15,
                      probs = seq(0, 1, length.out = no_classes + 1))

# here I define custom labels (the default ones would be ugly)
labels <- c()
for(idx in 1:length(quantiles)){
  labels <- c(labels, paste0(round(quantiles[idx], 2),
                             " – ",
                             round(quantiles[idx + 1], 2)))
}
# I need to remove the last label
# because that would be something like "66.62 - NA"
labels <- labels[1:length(labels)-1]

# here I actually create a new
# variable on the dataset with the quantiles
map_data$avg_age_15_quantiles <- cut(map_data$avg_age_15,
                                     breaks = quantiles,
                                     labels = labels,
                                     include.lowest = T)

p <- ggplot() +
    # municipality polygons (watch how I
   # use the new variable for the fill aesthetic)
    geom_polygon(data = map_data, aes(fill = avg_age_15_quantiles,
                                      x = long,
                                      y = lat,
                                      group = group)) +
    # municipality outline
    geom_path(data = map_data, aes(x = long,
                                   y = lat,
                                   group = group),
              color = "white", size = 0.1) +
```

```
    coord_equal() +
    theme_map() +
    labs(x = NULL,
         y = NULL,
         title = "Switzerland's regional demographics",
         subtitle = "Average age in Swiss municipalities, 2015",
         caption = "Geometries: ThemaKart, BFS; Data: BFS, 2016") +
  # now the discrete-option is used,
  # and we use guide_legend instead of guide_colourbar
  scale_fill_viridis(
    option = "magma",
    name = "Average age",
    discrete = T,
    direction = -1,
    guide = guide_legend(
     keyheight = unit(5, units = "mm"),
     title.position = 'top',
     reverse = T
  ))
p
```



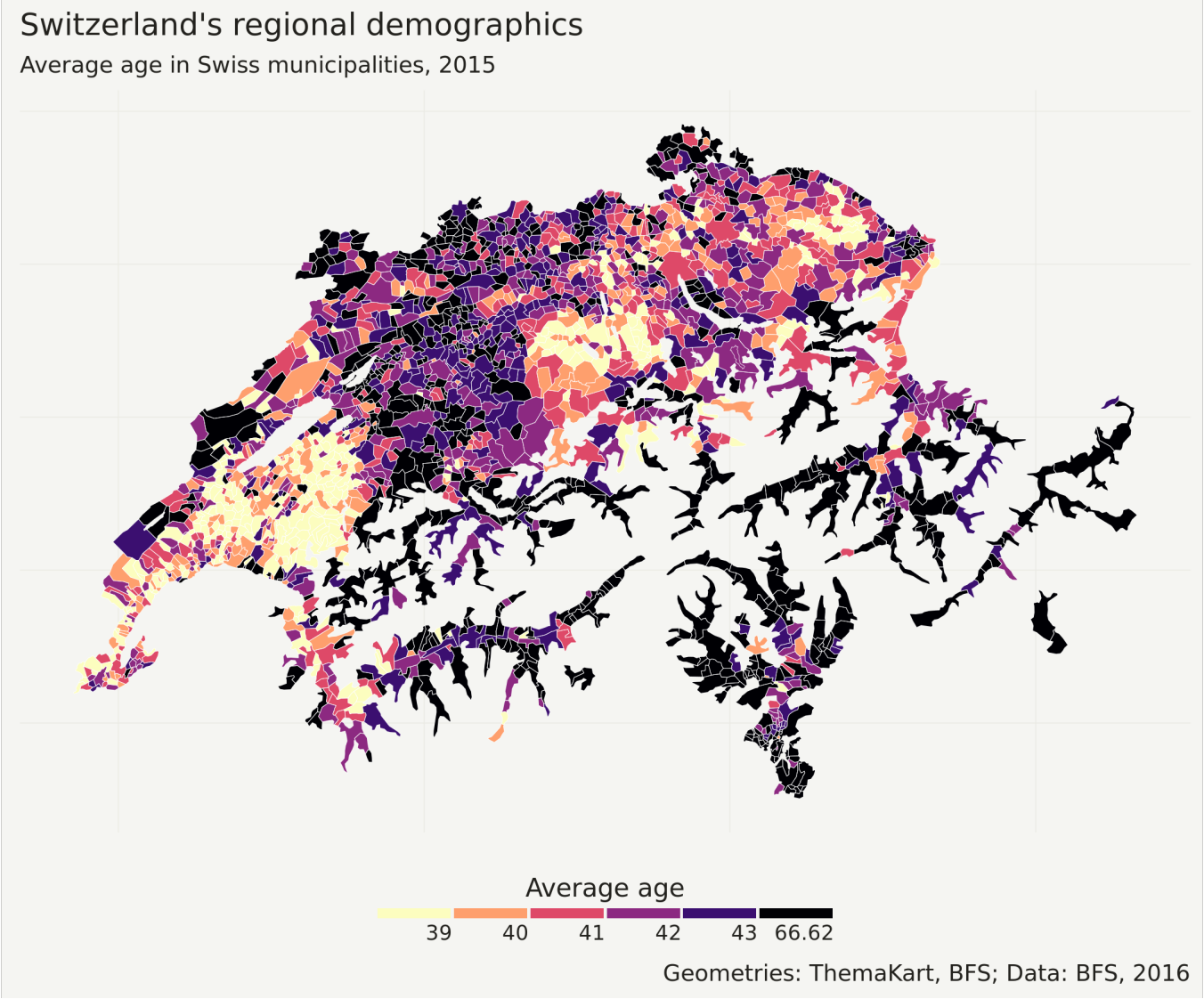

Wow! Now that is some regional variability ;-). But there is still a huge caveat: In my opinion, quantile scales are optimal at showing intra-dataset-variability, but sometimes this variability can be exaggerated. Most of the municipalities here are in the region between 39 and 43 years. The second caveat is that the legend looks somehow ugly with all these decimals, and that people are probably having problems interpreting such differently sized classes. That's why I am trying "pretty breaks" in the next step, and this is basically also what you see in almost all choropleths used for (data-)journalistic purposes.

## Discrete classes with pretty breaks

```
# here I define equally spaced pretty breaks -
```

```
# they will be surrounded by the minimum value at
# the beginning and the maximum value at the end.
# One could also use something like c(39,39.5,41,42.5,43),
# this totally depends on the data and your personal taste.
pretty_breaks <- c(39,40,41,42,43)
# find the extremes
minVal <- min(map_data$avg_age_15, na.rm = T)
maxVal <- max(map_data$avg_age_15, na.rm = T)
# compute labels
labels <- c()
brks <- c(minVal, pretty_breaks, maxVal)
# round the labels (actually, only the extremes)
for(idx in 1:length(brks)){
  labels <- c(labels,round(brks[idx + 1], 2))
}

labels <- labels[1:length(labels)-1]
# define a new variable on the data set just as above
map_data$brks <- cut(map_data$avg_age_15,
                     breaks = brks,
                     include.lowest = TRUE,
                     labels = labels)

brks_scale <- levels(map_data$brks)
labels_scale <- rev(brks_scale)

p <- ggplot() +
    # municipality polygons
    geom_polygon(data = map_data, aes(fill = brks,
                                      x = long,
                                      y = lat,
                                      group = group)) +
    # municipality outline
    geom_path(data = map_data, aes(x = long,
                                   y = lat,
                                   group = group),
              color = "white", size = 0.1) +
    coord_equal() +
    theme_map() +
    theme(legend.position = "bottom") +
    labs(x = NULL,
         y = NULL,
         title = "Switzerland's regional demographics",
         subtitle = "Average age in Swiss municipalities, 2015",
         caption = "Geometries: ThemaKart, BFS; Data: BFS, 2016")
q <- p +
    # now we have to use a manual scale,
```

```
    # because only ever one number should be shown per label
    scale_fill_manual(
          # in manual scales, one has to define colors, well, manually
          # I can directly access them using viridis' magma-function
          values = rev(magma(6)),
          breaks = rev(brks_scale),
          name = "Average age",
          drop = FALSE,
          labels = labels_scale,
          guide = guide_legend(
            direction = "horizontal",
            keyheight = unit(2, units = "mm"),
            keywidth = unit(70 / length(labels), units = "mm"),
            title.position = 'top',
            # I shift the labels around, the should be placed
            # exactly at the right end of each legend key
            title.hjust = 0.5,
            label.hjust = 1,
            nrow = 1,
            byrow = T,
            # also the guide needs to be reversed
            reverse = T,
            label.position = "bottom"
          )
      )

q
```



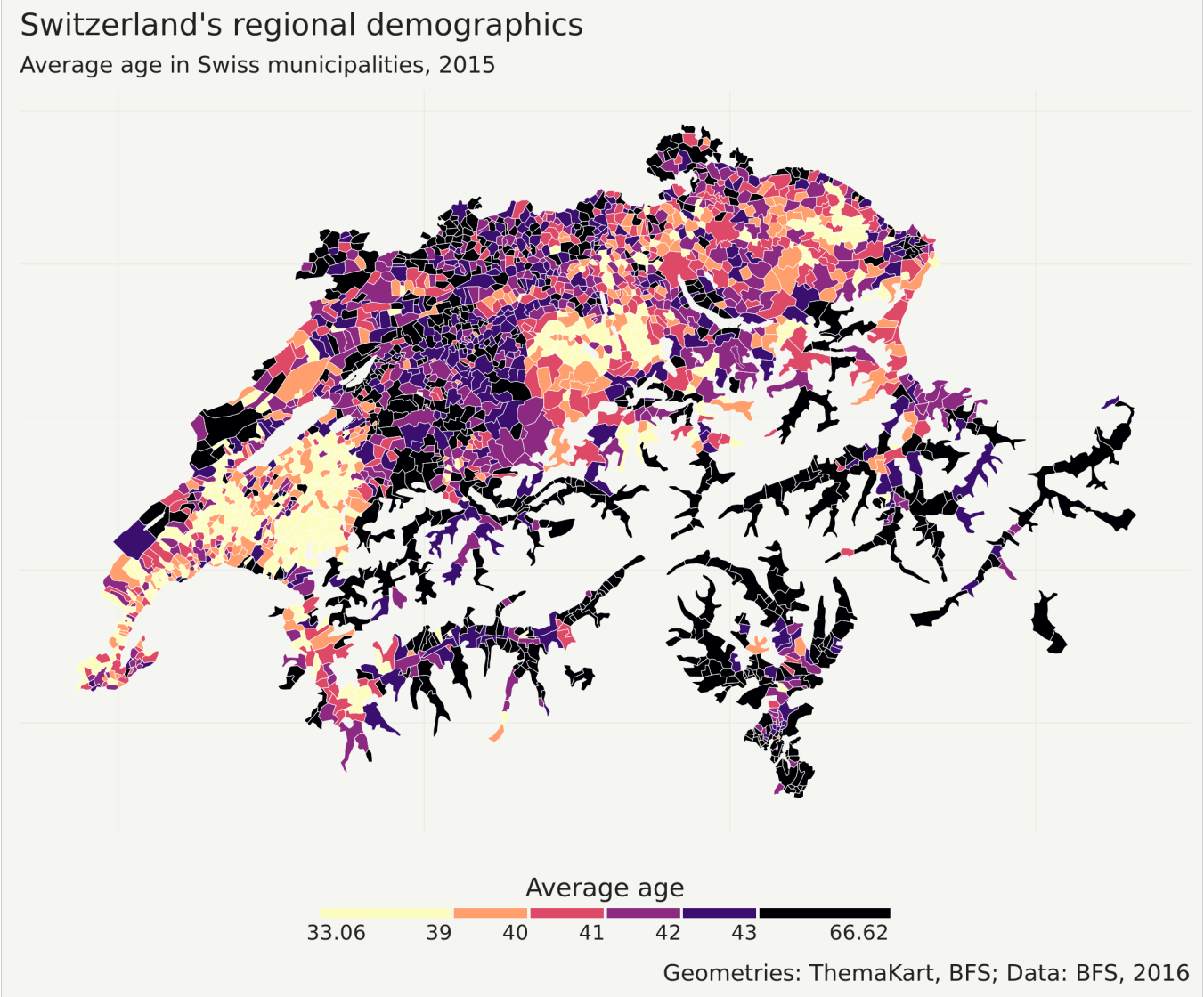

Now we have classes with the ranges 33.06 to 39, 39 to 40, 40 to 41, and so on... So four classes are of the same size and the two classes with the extremes are differently sized. One option to communicate this is to make their respective legend keys wider than usual. `ggplot2` doesn't have a standard option for that, so I had to dig deep into the underlying `grid` package and extract the relevant `grobs` and change their widths. All of the following numbers are the result of trying and trying around. I have not yet fully understood how that system actually works and certainly, it could be made more versatile. Something for next christmas...

## More intuitive legend

```
extendLegendWithExtremes <- function(p){
```

```
p_grob <- ggplotGrob(p)
legend <- gtable_filter(p_grob, "guide-box")
legend_grobs <- legend$grobs[[1]]$grobs[[1]]
# grab the first key of legend
legend_first_key <- gtable_filter(legend_grobs, "key-3-1-1")
legend_first_key$widths <- unit(2, units = "cm")
# modify its width and x properties to make it longer
legend_first_key$grobs[[1]]$width <- unit(2, units = "cm")
legend_first_key$grobs[[1]]$x <- unit(0.15, units = "cm")

# last key of legend
legend_last_key <- gtable_filter(legend_grobs, "key-3-6-1")
legend_last_key$widths <- unit(2, units = "cm")
# analogous
legend_last_key$grobs[[1]]$width <- unit(2, units = "cm")
legend_last_key$grobs[[1]]$x <- unit(1.02, units = "cm")

# grab the last label so we can also shift its position
legend_last_label <- gtable_filter(legend_grobs, "label-5-6")
legend_last_label$grobs[[1]]$x <- unit(2, units = "cm")

# Insert new color legend back into the combined legend
legend_grobs$grobs[legend_grobs$layout$name == "key-3-1-1"][[1]] <-
  legend_first_key$grobs[[1]]
legend_grobs$grobs[legend_grobs$layout$name == "key-3-6-1"][[1]] <-
  legend_last_key$grobs[[1]]
legend_grobs$grobs[legend_grobs$layout$name == "label-5-6"][[1]] <-
  legend_last_label$grobs[[1]]

# finally, I need to create a new label for the minimum value
new_first_label <- legend_last_label$grobs[[1]]
new_first_label$label <- round(min(map_data$avg_age_15, na.rm = T), 2)
new_first_label$x <- unit(-0.15, units = "cm")
new_first_label$hjust <- 1

legend_grobs <- gtable_add_grob(legend_grobs,
                                new_first_label,
                                t = 6,
                                l = 2,
                                name = "label-5-0",
                                clip = "off")
legend$grobs[[1]]$grobs[1][[1]] <- legend_grobs
p_grob$grobs[p_grob$layout$name == "guide-box"][[1]] <- legend

# the plot is now drawn using this grid function
grid.newpage()
grid.draw(p_grob)
```

```
}
extendLegendWithExtremes(q)
```

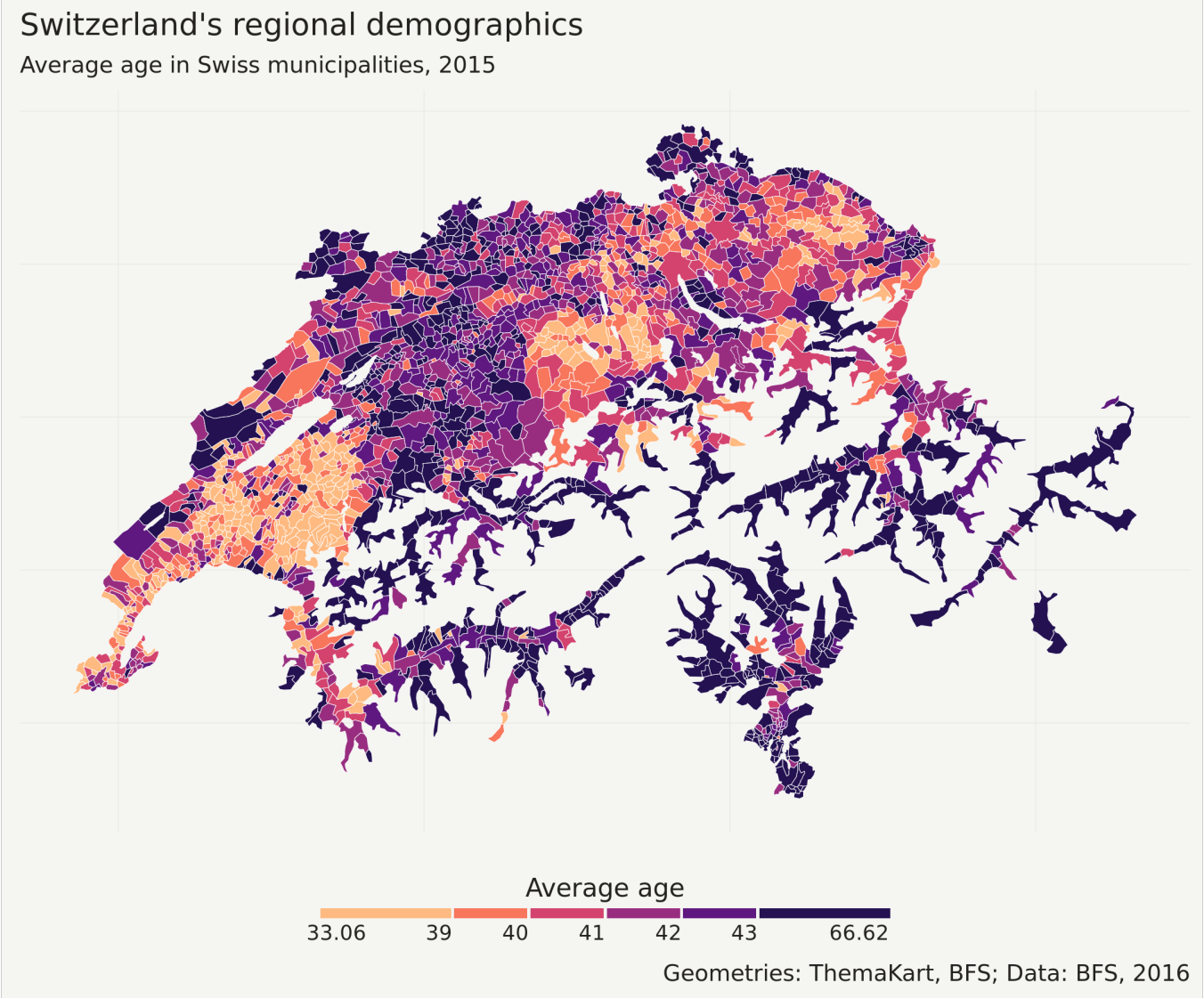



## Better colors for classes

Almost perfect. What I still don't like is the very bright yellow color of the first class. It makes it difficult to see the borders of the municipalities with that color. Also I find the color of the last class a bit too dark. That's why I now use the `magma` function with 8 classes and strip of the first and last class.

```
p <- p + scale_fill_manual(
  # magma with 8 classes
  values = rev(magma(8)[2:7]),
  breaks = rev(brks_scale),
```

```
  name = "Average age",
  drop = FALSE,
  labels = labels_scale,
  guide = guide_legend(
    direction = "horizontal",
    keyheight = unit(2, units = "mm"),
    keywidth = unit(70/length(labels), units = "mm"),
    title.position = 'top',
    title.hjust = 0.5,
    label.hjust = 1,
    nrow = 1,
    byrow = T,
    reverse = T,
    label.position = "bottom"
  )
)
# reapply the legend modification from above
extendLegendWithExtremes(p)
```



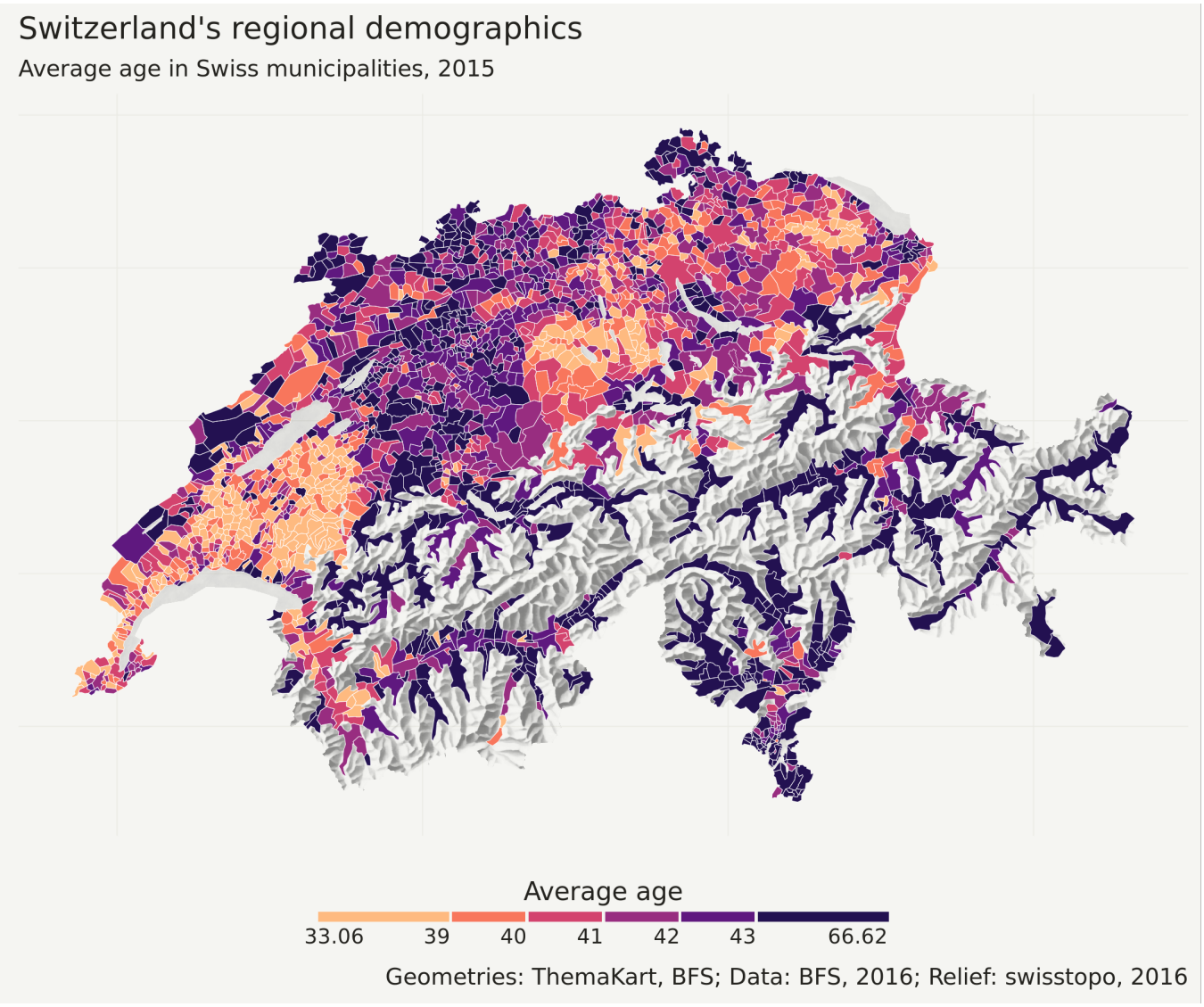

A beauty!

## Relief

What's needed now to give it a boost of aesthetic value is the relief of the Swiss Alps. Every mountain lover will appreciate that.

I add the relief with `geom_raster`. Now the problem is that I can't use the `fill` aesthetic because it (or its scale) is already in use by the `geom_polygon` layer. The workaround is using the `alpha` aesthetic which works fine here because the relief should be displayed with a greyscale anyway.

```
p <- ggplot() +
    # raster comes as the first layer, municipalities on top
    geom_raster(data = relief, aes(x = x,
                                   y = y,
                                   alpha = value)) +
    # use the "alpha hack"
    scale_alpha(name = "", range = c(0.6, 0), guide = F)  +
    # municipality polygons
    geom_polygon(data = map_data, aes(fill = brks,
                                      x = long,
                                      y = lat,
                                      group = group)) +
    # municipality outline
    geom_path(data = map_data, aes(x = long,
                                   y = lat,
                                   group = group),
              color = "white", size = 0.1) +
    # apart from that, nothing changes
    coord_equal() +
    theme_map() +
    theme(legend.position = "bottom") +
    labs(x = NULL,
         y = NULL,
         title = "Switzerland's regional demographics",
         subtitle = "Average age in Swiss municipalities, 2015",
         caption = "Geometries: ThemaKart, BFS; Data: BFS, 2016; Relief: swisstopo, 201
    scale_fill_manual(
      values = rev(magma(8)[2:7]),
      breaks = rev(brks_scale),
      name = "Average age",
      drop = FALSE,
      labels = labels_scale,
      guide = guide_legend(
        direction = "horizontal",
        keyheight = unit(2, units = "mm"),
        keywidth = unit(70/length(labels), units = "mm"),
        title.position = 'top',
        title.hjust = 0.5,
        label.hjust = 1,
        nrow = 1,
        byrow = T,
        reverse = T,
        label.position = "bottom"
      )
    )
extendLegendWithExtremes(p)
```




## Final map

What follows are a couple of adjustments concerning:

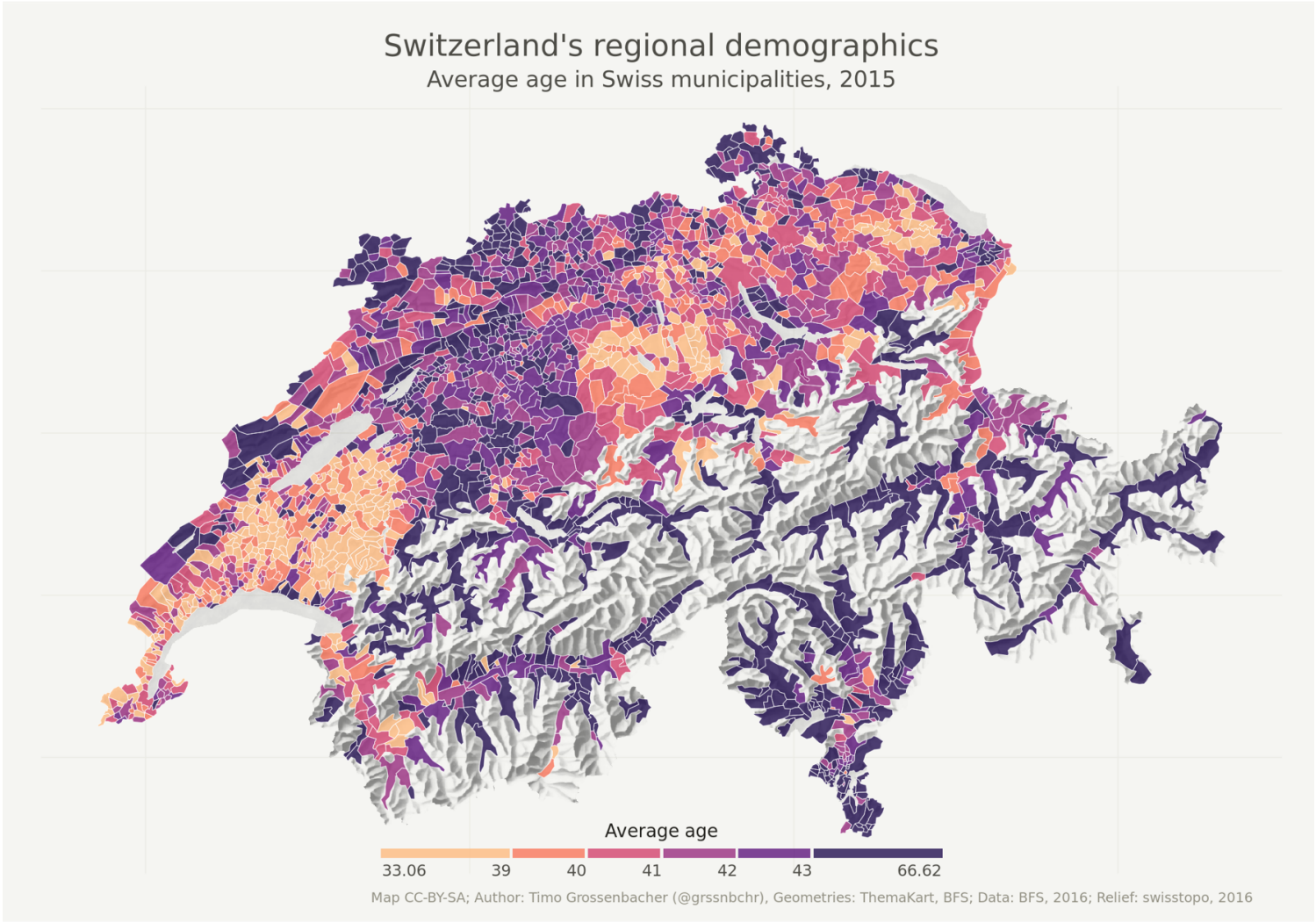
- font colors
- the position of the title
- the plot margins, i.e.: how to make better use of the available space and show the map as big as possible
- smaller and less prominent caption at the bottom

Most of that happens in the additional `theme` specifications. Again, this is just tediously trying out values after values after values...

To my great joy I also discovered that there is an `alpha` argument to the `magma` function, which gives the colors a certain pastel tone and make the map look even more geo-hipsterish (if you ask me).

```
p <- ggplot() +
    # municipality polygons
    geom_raster(data = relief, aes_string(x = "x",
                                          y = "y",
                                          alpha = "value")) +
    scale_alpha(name = "", range = c(0.6, 0), guide = F)  +
    geom_polygon(data = map_data, aes(fill = brks,
                                      x = long,
                                      y = lat,
                                      group = group)) +
    # municipality outline
    geom_path(data = map_data, aes(x = long,
                                   y = lat,
                                   group = group),
              color = "white", size = 0.1) +
    coord_equal() +
    theme_map() +
    theme(
      legend.position = c(0.5, 0.03),
      legend.text.align = 0,
      legend.background = element_rect(fill = alpha('white', 0.0)),
      legend.text = element_text(size = 7, hjust = 0, color = "#4e4d47"),
      plot.title = element_text(hjust = 0.5, color = "#4e4d47"),
      plot.subtitle = element_text(hjust = 0.5, color = "#4e4d47",
                                   margin = margin(b = -0.1,
                                                   t = -0.1,
                                                   l = 2,
                                                   unit = "cm"),
                                   debug = F),
      legend.title = element_text(size = 8),
      plot.margin = unit(c(.5,.5,.2,.5), "cm"),
      panel.spacing = unit(c(-.1,0.2,.2,0.2), "cm"),
      panel.border = element_blank(),
      plot.caption = element_text(size = 6,
                                  hjust = 0.92,
                                  margin = margin(t = 0.2,
                                                  b = 0,
                                                  unit = "cm"),
```

```
                                    color = "#939184")
    ) +
    labs(x = NULL,
         y = NULL,
         title = "Switzerland's regional demographics",
         subtitle = "Average age in Swiss municipalities, 2015",
         caption = "Map CC-BY-SA; Author: Timo Grossenbacher (@grssnbchr), Geometries: 
    scale_fill_manual(
      values = rev(magma(8, alpha = 0.8)[2:7]),
      breaks = rev(brks_scale),
      name = "Average age",
      drop = FALSE,
      labels = labels_scale,
      guide = guide_legend(
        direction = "horizontal",
        keyheight = unit(2, units = "mm"),
        keywidth = unit(70/length(labels), units = "mm"),
        title.position = 'top',
        title.hjust = 0.5,
        label.hjust = 1,
        nrow = 1,
        byrow = T,
        reverse = T,
        label.position = "bottom"
      )
    )
extendLegendWithExtremes(p)
```




Thanks for reading, I hope you learned something. Producing high-quality graphics like these with pure `ggplot2` is sometimes more an art than a science and veeeeeeeryyyyy tedious, and it would probably be way easier to export the map at an early stage and make adjustments in Illustrator or another vector editor. But then, I just like the thought of a fully automagic, reproducible workflow, it's almost an obsession. The big challenge here is to put everything into a more versatile function, or even a package, that can produce maps like these with arbitrary scales (discrete, continuous, quantiles, pretty breaks, whatever) and arbitrary geo data (for the US, for example).

If you think this example can be improved in any way, please use the comment function below. I'd also be very happy to see this map adapted to other geographic regions and/or other datasets.

As always: Follow me on Twitter!

## Update, January 2nd, 2017

This blog post has gone quite through the roof. For example, it was featured on the Revolution Analytics blog. One guy even printed the map and hung it on the wall!

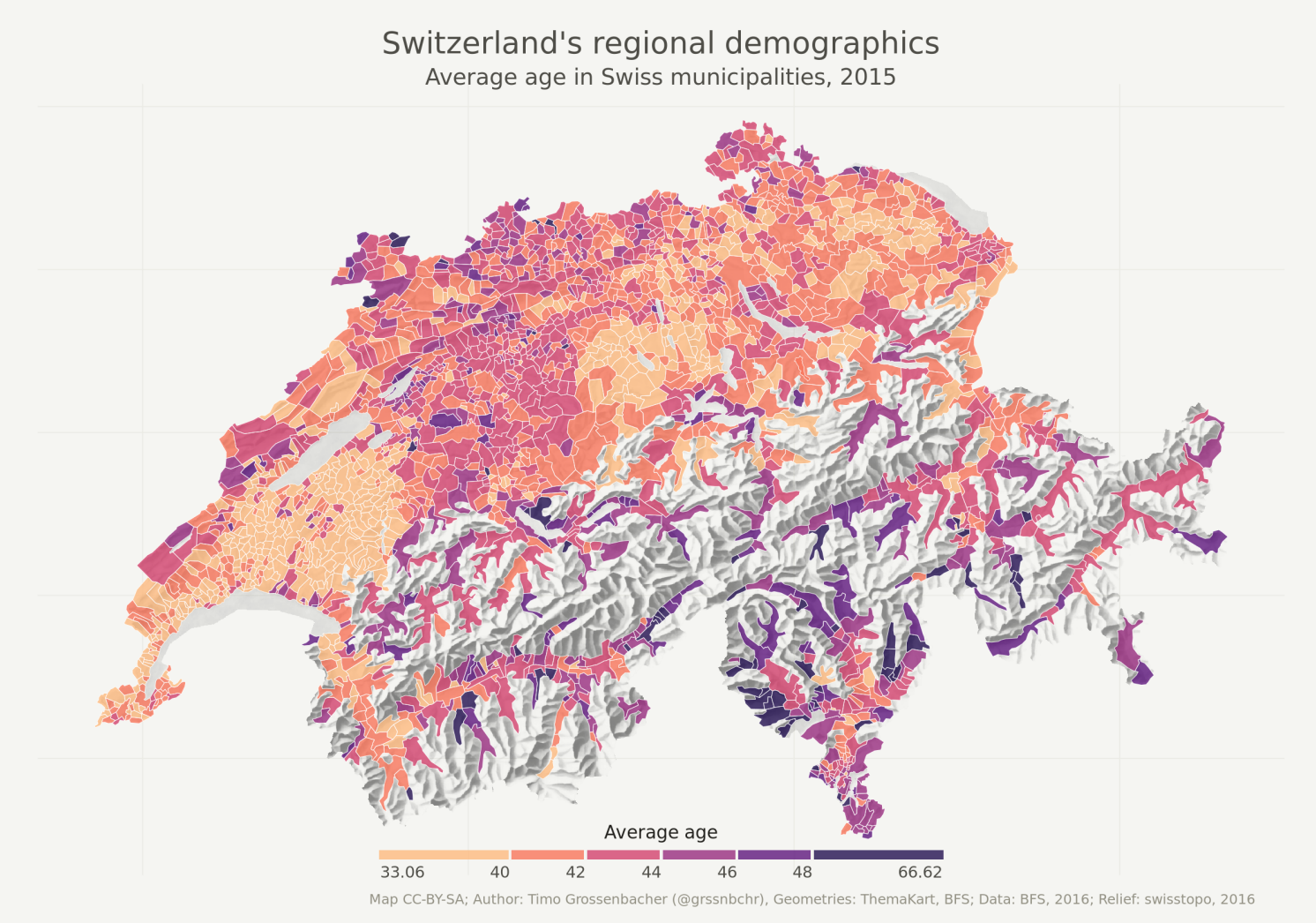
I have also received a lot of constructive feedback in the meantime. I especially appreciated the discussions on the RStats Subreddit, particularly the one about the legend / color scale.

Based on that discussion I decided to make a slightly altered version of the color scale so one can compare the visual effect.

```
# same code as above but different breaks
pretty_breaks <- c(40,42,44,46,48)
# find the extremes
minVal <- min(map_data$avg_age_15, na.rm = T)
maxVal <- max(map_data$avg_age_15, na.rm = T)
# compute labels
labels <- c()
brks <- c(minVal, pretty_breaks, maxVal)
# round the labels (actually, only the extremes)
for(idx in 1:length(brks)){
  labels <- c(labels,round(brks[idx + 1], 2))
}

labels <- labels[1:length(labels)-1]
# define a new variable on the data set just as above
map_data$brks <- cut(map_data$avg_age_15,
                     breaks = brks,
                     include.lowest = TRUE,
                     labels = labels)

brks_scale <- levels(map_data$brks)
labels_scale <- rev(brks_scale)

p <- ggplot() +
    # municipality polygons
    geom_raster(data = relief, aes_string(x = "x",
                                          y = "y",
```

```
                                        alpha = "value")) +
  scale_alpha(name = "", range = c(0.6, 0), guide = F)  +
  geom_polygon(data = map_data, aes(fill = brks,
                                    x = long,
                                    y = lat,
                                    group = group)) +
  # municipality outline
  geom_path(data = map_data, aes(x = long,
                                 y = lat,
                                 group = group),
            color = "white", size = 0.1) +
  coord_equal() +
  theme_map() +
  theme(
    legend.position = c(0.5, 0.03),
    legend.text.align = 0,
    legend.background = element_rect(fill = alpha('white', 0.0)),
    legend.text = element_text(size = 7, hjust = 0, color = "#4e4d47"),
    plot.title = element_text(hjust = 0.5, color = "#4e4d47"),
    plot.subtitle = element_text(hjust = 0.5, color = "#4e4d47",
                                 margin = margin(b = -0.1,
                                                 t = -0.1,
                                                 l = 2,
                                                 unit = "cm"),
                                 debug = F),
    legend.title = element_text(size = 8),
    plot.margin = unit(c(.5,.5,.2,.5), "cm"),
    panel.spacing = unit(c(-.1,0.2,.2,0.2), "cm"),
    panel.border = element_blank(),
    plot.caption = element_text(size = 6,
                                hjust = 0.92,
                                margin = margin(t = 0.2,
                                                b = 0,
                                                unit = "cm"),
                                color = "#939184")
  ) +
  labs(x = NULL,
       y = NULL,
       title = "Switzerland's regional demographics",
       subtitle = "Average age in Swiss municipalities, 2015",
       caption = "Map CC-BY-SA; Author: Timo Grossenbacher (@grssnbchr), Geometries:
  scale_fill_manual(
    values = rev(magma(8, alpha = 0.8)[2:7]),
    breaks = rev(brks_scale),
    name = "Average age",
    drop = FALSE,
    labels = labels_scale,
```

```
      guide = guide_legend(
        direction = "horizontal",
        keyheight = unit(2, units = "mm"),
        keywidth = unit(70/length(labels), units = "mm"),
        title.position = 'top',
        title.hjust = 0.5,
        label.hjust = 1,
        nrow = 1,
        byrow = T,
        reverse = T,
        label.position = "bottom"
      )
    )
extendLegendWithExtremes(p)
```

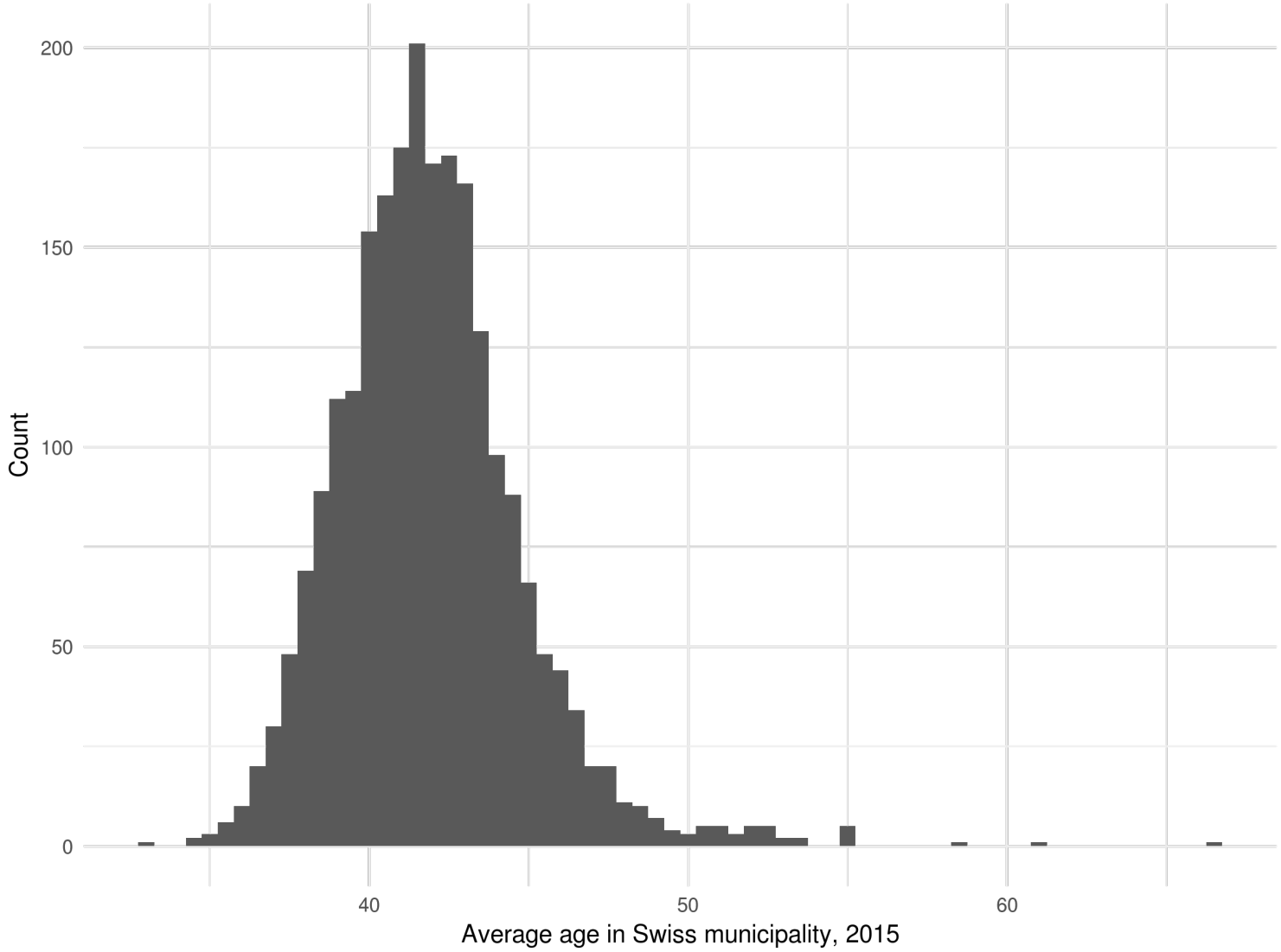



Notice that I extended the range of the first class from 33.06-39 to 33.06-40 and that, now, the class-

es in the "middle" have a range of two years rather than one year. This has the advantage that both "extreme" classes' ranges are now a bit more similar, but of course, the first is still a lot smaller than the last. I would say the disadvantage of this approach is that now some "visual balance" between both extremes is lost, mostly due to the fact that a lot of municipalities have an average age below 40 years. However, it has the other advantage that the really "old" municipalities at the far-right of the scale can now be more easily identified.

At this point, it might make sense to look at the histogram of the municipalities:

```
ggplot(data = data, aes(x = avg_age_15)) +
  geom_histogram(binwidth = 0.5) +
  theme_minimal() +
  xlab("Average age in Swiss municipality, 2015") +
  ylab("Count")
```




As you can see, the municipalities are almost normally distributed, with most municipalities being in the range between 39 and 43 years (>75%, look at the quantiles computation below). From that perspective, the first class configuration might still be "closer" to the data.

```
quantile(data$avg_age_15)
```

```
##       0%      25%      50%      75%     100% 
## 33.05566 39.99845 41.65980 43.37581 66.61538
```

But what do I know.

No, really: This is a very difficult problem. The choice of a certain color scale greatly alters the visual

perception of the underlying spatial patterns. I remember from my Geography studies that there are guidelines on how to handle this (anyone got a good link, by the way?), but there is no wrong or right. It'd be nice if you posted your opinion about that in the comments!

One last note: Some people seem to have had problems with the `maptools` package. In case you're wondering, here is the setup I used to run the script in the first place:

```
R version 3.3.1 (2016-06-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.1 LTS

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C               LC_TIME=de_CH.UTF-8        LC
 [5] LC_MONETARY=de_CH.UTF-8    LC_MESSAGES=en_US.UTF-8    LC_PAPER=de_CH.UTF-8       LC
 [9] LC_ADDRESS=C               LC_TELEPHONE=C             LC_MEASUREMENT=de_CH.UTF-8 LC

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] gtable_0.2.0  dplyr_0.5.0   viridis_0.3.4 ggplot2_2.2.0 raster_2.5-8  rgdal_1.1-10

loaded via a namespace (and not attached):
 [1] Rcpp_0.12.7      knitr_1.14       magrittr_1.5     maptools_0.8-40  munsell_0.4.3
 [8] R6_2.1.3         plyr_1.8.4       tools_3.3.1      DBI_0.5-1        digest_0.6.10
[15] tibble_1.2       gridExtra_2.2.1  formatR_1.4      labeling_0.3     scales_0.4.1
```

## Data in a barchart

```
rgs <- read_excel("input/be-b-00.04-rgs-15.xls", skip = 16, col_names = F) %>%
    select(bfs_id = X__1, NAME = X__2)
data_sorted <- data %>% left_join(rgs) %>% arrange(desc(avg_age_15))
```

```
## Joining, by = "bfs_id"
```

```
data_to_plot <- data %>% left_join(rgs)
```

```
## Joining, by = "bfs_id"
```

```
data_to_plot %<>% mutate(NAME = factor(NAME, levels = data_sorted$NAME))

p <- ggplot(data_to_plot, aes(y = avg_age_15, x = NAME)) +
  geom_bar(stat = "identity") +
  labs(x = "Gemeinde", y = "Durchschnittsalter 2015") +
  theme_minimal() +
  theme(axis.text = element_text(size = 7))+
  xlim(data_sorted[2315:2324,]$NAME) +
  ylim(c(0,70)) +
  theme(axis.text.x = element_text(angle = 60, hjust = 1))

ggsave(p, filename = "output/unteres_extrem.png", width = 8, height = 3)
```

```
## Warning: Removed 2314 rows containing missing values (position_stack).
```

https://github.com/grssnbchr/thematic-maps-ggplot2/master/index.md Markdown with by ♥ wp-gfm



Filed under: Data Visualization, Geoinformation

Tagged: Data Journalism, Data visualization, ggplot2, Mapping, Maps, R, Switzerland, Visualization

« PREVIOUS POST NEXT POST »

## 60 Comments

Fr.

Superb example.

I hope that you will turn what you did with the legend into a set of handy functions. To me, that's the part of your code that I could most make use of (the rest of your post depends either on good data sources or on smart manipulation of quantiles; of course, you could also produce some good code about these aspects: an interface to your data sources, or smarter 'cut' functions for choosing quantiles—but I would prioritize the very nice work you did on the legend).

December 28, 2016 / Reply

**Pingback:** Combine choropleth data with raster maps using R - Use-R!Use-R!

**Pingback:** Combine choropleth data with raster maps using R – Cloud Data Architect

**Pingback:** Combine choropleth data with raster maps using R – Mubashir Qasim

### Jose Manuel vera

Great Work! Nice and pretty map. My suggestion is to use pacman to manage packages instead all the "require-install" part.
https://cran.r-project.org/web/packages/pacman/index.html

Easy an neat code in pacman will be:

if (!require(pacman)) install.packages("pacman")

p_load("rgeos", "rgdal", "raster", "ggplot2", "viridis", "dplyr", "gtable", "grid)

Regards

December 29, 2016 / Reply


### Matt Sandy

Well damn, this is beautiful.

December 30, 2016 / Reply

### Oscar Perpiñán Lamigueiro

Really nice maps!
I have tried to imitate them with a lattice approach (without ggplot2). Here is the code if you are interested: https://gist.github.com/oscarperpinan/63f6e4ab95ca90e31a350e4392663b12

December 30, 2016 / Reply

### Timo Grossenbacher

Wow, this is very cool!

January 7, 2017 / Reply

### Thomas

Very nice work indeed! I had to load “maptools” as well in order to run the code and get rid of the following message: Error in get(“rgeos”, envir = .MAPTOOLS_CACHE) : object ‘rgeos’ not found

December 30, 2016 / Reply

**Enrique**

Effectively, great job. I had the same problem than Thomas. I resolve with maptools and broom packages, function tidy, following the instructions in this web: https://github.com/tidyverse/ggplot2/issues/1447

Happy new year, dear mappers

December 31, 2016 / Reply

**Timo Grossenbacher**

Mh, for me it works, with the following packages:

```
attached base packages:
[1] grid stats graphics grDevices utils datasets methods base

other attached packages:
[1] gtable_0.2.0 dplyr_0.5.0 viridis_0.3.4 ggplot2_2.2.0 raster_2.5-8 rgdal_1.1-
10 sp_1.2-3 rgeos_0.3-21

loaded via a namespace (and not attached):
[1] Rcpp_0.12.7 knitr_1.14 magrittr_1.5 maptools_0.8-40 munsell_0.4.3 color-
space_1.2-6 lattice_0.20-34
```

```
[8] R6_2.1.3 plyr_1.8.4 tools_3.3.1 DBI_0.5-1 digest_0.6.10 lazyeval_0.2.0 as-
sertthat_0.1
[15] tibble_1.2 gridExtra_2.2.1 formatR_1.4 labeling_0.3 scales_0.4.1 for-
eign_0.8-66
```

Maybe it works because of ggplot 2.2.0? Somehow maptools seems to be loaded automatically via namespace.

January 2, 2017 / Reply

### Enrico Spinielli

Timo,
great post!
A question: is there a reason why you choose to install some packages from “http://cran.us.r-project.org” and pthers from “http://cloud.r-project.org”?

January 4, 2017 / Reply

### Timo Grossenbacher

I knew this question would come 😉 No, there is no reason, it’s just because I drew the code together from different of my scripts, and in some of those, cloud is used, in others cran.us. I should unify this, there is no reason to use one over the other, is there?

January 4, 2017 / Reply

**Pingback:** Bonnes résolutions 2017, édition « science des données » | Polit’bistro : des

politiques, du café

**Pingback:** Exasperated Calculator » Blog Archive » Pretty Mapping of Australia using R

**tim**

I have also made a version without ggplot2 here: http://rgeomatic.hypotheses.org/1086
I tried a different (let's call it statistical) legend layout. Sorry, the post is in French…

January 6, 2017 / Reply

**Timo Grossenbacher**

I could understand it 😉 I really like the legend, very cool idea.

January 7, 2017 / Reply

**Pingback:** Web Picks (week of 9 January 2017) | DataMiningApps

**Sothy**

Hi Timo, really nice work and a great tutorial!
I was wondering if there a Municipality shapefile exists with swiss postalcodes and only the productive areas.
I was trying to recreate your work with the LV95/PLZO_PLZ.shp files, because my dataset consists PLZ-Codes instead of BFS_ID.

It would be great, if you could point me in the right direction.
—
Best Regards

January 27, 2017 / Reply

### Timo Grossenbacher

Thanks and you're welcome! Unfortunately I don't know of any such file. But I'm sure you'll find a file with mappings between PLZ and BFS_ID (maybe by the BFS itself – keyword "Amtliches Gemeindeverzeichnis"). I did a quick search but couldn't find anything, but with some digging you'll surely find what you're looking for… would be great if you could share your findings here!

January 29, 2017 / Reply

### Toby

You can find the mapping here: https://www.cadastre.ch/content/cadastre-internet/de/services/service/plz.html

February 19, 2017 / Reply

### Richard Eberle

Hello Timo!
Great work you did there:)
Do you have summarized the municipalities into cantons?
So that data can be plotted for each canton?

I would like to use it for a project for a course (data visualization) of my master program.

Thank you very much
Richard

March 8, 2017 / Reply

### Timo Grossenbacher

Hello, thank you! No, I haven't summarized it, but it shouldn't be a problem to get the same data on the level of cantons, see FSO. Also, there are shapefiles of cantons available from Swisstopo. Some googling will do 😉

March 14, 2017 / Reply

### Stefan Schmutz

Hi Richard,

it's probably too long ago. But since I stumbled across this now and found a shapefile, here is the link:
Search for "Switzerland" download the "Shapefile", the "gadm36_CHE_1.shp" file is what we're looking for.
https://gadm.org/download_country_v3.html

I crated a Gist of a simple example which plots the cantons:
https://gist.github.com/sschmutz/bf0ad51ce2c810e323cbed4348fe8c84

Best
Stefan

January 25, 2019 / Reply

**Pingback:** Como hacer un mapa muy bonito de España en ggplot2 | Pybonacci

### Ruslan Shumilov

The final map is indeed beautiful! I am surprised how one can create maps with R too, didn`t know that…
So many explanations, wow – great respect for your work 👍

May 24, 2017 / Reply

### Luis Urbina

Great job! I am reproducing this for Perú and US. Thank for posting this resource.

July 14, 2017 / Reply

### Nitin Shukla

Is it possible to plot the same using python? If anyone has done it, will you please enlighten me. Thanks in advance!

July 18, 2017 / Reply

### David Zumbach

gorgeous

August 8, 2017 / Reply

### Xavier

Hi Timo, Thanks for the tutorial, your map is truly gorgeous!

I tried to play with the code but got blocked with the relief data. I did not find a file named 02-relief-georef-clipped-resampled.tif on the swisstopo website. One TIF seemed to only contains relief (02-relief-georef-clipped-resampled) but it shows a big portion of europe (very beautiful actually but my CPU nearly had a heart attack). Do you remember where you found 02-relief-georef-clipped-resampled.tif?

(I saw it on your github but was wondering about the original source)

Thanks a lot!!

September 11, 2017 / Reply

### Timo Grossenbacher

Thank you! Mhh, so if it's not in that Swiss Map Raster 1000 (which I thought it was, but maybe they changed it), I must have gotten it directly from Swisstopo (it's already a while ago). I probably georeferenced the TIFF myself, that's why it has the `georef` part in the name. I'm sorry about this, it's probably better if I delete the link. You could try and ask the Swisstopo guys though, maybe they'll give you something more "original".

September 11, 2017 / Reply

**Pingback:** R Tips – The Tea House

**Pingback:** First steps with MRF smooths – Cloud Data Architect

**Pingback:** First steps with MRF smooths – Mubashir Qasim

**kishore**

ERROR: An error has occurred. Check your logs or contact the app author for clarification when i tried using ggplot2 to plot choropleth maps in rshiny i get this error.please help me what to do

July 22, 2018 / Reply

**Pingback:** Como hacer un mapa muy bonito de España en ggplot2 - Lab News

**Deepak**

Hi Timo,

Thanks a lot for this awesome tutorial. I have followed this in my work and made awesome graph. I am facing an issue which I believe you can help with. The issue I have is that I cannot move individual labels around, the below code (I replicated your code) won't just work for me.

```
# grab the last label so we can also shift its position
legend_last_label <- gtable_filter(legend_grobs, "label-5-6")
legend_last_label$grobs[[1]]$x <- unit(2, units = "cm")
```

If you could help me with this I would be grateful (I have also posted on stackoverflow for my data if you wish to take a look https://stackoverflow.com/questions/52435521/customize-label-element-position-in-ggplot-using-gtable-grob?noredirect=1#comment91814643_52435521)

Thanks in advance.

September 23, 2018 / Reply

### Timo Grossenbacher

Hey!
Sad to see you're having problems with the script. I suspect that different packages might be the source of the problem. I cannot really help you other than telling you to try and install exactly the package versions that are specified at the end of the post. The label positioning was and is a bit "hacky" anyway. Maybe a heads-up: I plan to "renew" the post with ggplot2 3.0.0, the sf package, and make it more reproducible. Hope I still find time this year.
Thanks for your patience.

October 11, 2018 / Reply

### Fabian Heimsch

Hi Timo

I would like to say that your work really looks awesome. I may have missed something, but is the *input* folder which you regularly mention in your article available? I sort of constantly fail if I

try to reproduce the codes with my own data / shapefiles.

I would be greatly thankful with you for a hint

Best wishes

Fabian

October 9, 2018 / Reply

### Timo Grossenbacher

Hello Fabian
Thank you! Yes, it comes with the repository: https://github.com/grssnbchr/thematic-maps-ggplot2/tree/master/input
Can you tell me more about the error message you get? Don't think it's because of a missing input folder.
Timo

October 11, 2018 / Reply

### Fabian

Hi Timo

I am NOT a newbie to R, actually, I am using STATA and R both at a professional level. What I think is that R, as good as it is, has some characteristics that are super cumbersome for users, e.g. your very first line of code, the part with "readOGR" does not work for me on Windows 10, no matter how long I am trying, which shapefile I use (different ones), whether I leave file extensions away or not, whether I use small or capital letters for network drives, whether I suppress

the backslash at the end of the path, and and and …

my shape file is located in:
“C:/Users/my.user.name/Desktop/test.shape.shp”
I am giving R:
readOGR(dsn=”C:/Users/my.user.name”, layer = “test_shape”).

As said, I was testing all explanations from stack overflow, but nothing works. It is super frustrating…

Best

Fabian

December 18, 2018 / Reply

### Timo Grossenbacher

Hey Fabian. Sorry it doesn’t work. But it looks like your missing “Desktop” in your path. Also, Windows 10 uses backslashes instead of slashes. Also, if your shapefile is actually named “test.shape.shp”, your layer should be “test.shape” instead of “test_shape”, so I guess your line should be
readOGR(dsn=”C:\Users\my.user.name\Desktop\test.shape.shp”, layer = “test.shape”)

But it’s hard to debug this from here. I don’t think it’s an R problem though.. 😉

December 19, 2018 / Reply

### Fabian

Hi Timo – Many thanks for your fast and kind answer – well, the problem in most cases is in front of the computer 🙂 – thanks a lot for your help, anyway, I actually did exactly as you wrote already, but I will fight on.

December 19, 2018 /

### Jake Range

Hi Timo,

Love your work. This has got to be one of my favourite maps.
I am trying to make a legend similar to yours, however, the white-space gaps between each class in my legend are too big. Do you have any idea how to reduce the space between each colour in the legend?

Thanks in advance for any help!
Jake

October 23, 2018 / Reply

### Timo Grossenbacher

Hey Jake
Thanks. I can only tell you what I told another person with similar problems: I suspect that different packages might be the source of the problem. I cannot really help you other than telling you to try and install exactly the package versions that are specified at the end of the post. The label positioning was and is a bit "hacky" anyway. Maybe a heads-up: I plan to "renew" the post with ggplot2 3.0.0, the sf package, and make it more reproducible. Hope I still find time this year.
Thanks for your patience and sorry there's no better solution atm.
Timo

October 24, 2018 / Reply

**Ricardo Esparza**

Hi Timo,
Beautiful visualization! Thanks so much for sharing and for the walk-through and tutorial. I just have a question, how do you get to get a geom_polygon with such a high resolution?
I plotted a similar map for the US, yet the quality of the rendering was not as great as yours; because of the size of the shapefile, I had to simplify it and then use the fortify function to get the polygons. Do you think this was the reason? Is there any specific setting you have that makes it look so crisp and clear?
Thank you!
Best,
Ricardo

November 28, 2018 / Reply

**Timo Grossenbacher**

You're welcome. Mh, crispness shouldn't be a problem with shapefiles, as they are in the vector format. If you want you can post a link to a screenshot of your map, so I can have a look at it and make a guess?

November 29, 2018 / Reply

**Veronica Southerland**

Thank you so much for this tutorial! Very helpful and the notations make it great for learning

ggplot.

Perhaps Ricardo is referring to when exporting? I used dpi 320 to specify higher dpi and it looks much better exported.

ggsave(p, filename = “asthmaa.png”, width = 8, height = 3, dpi=320)

March 4, 2019 / Reply

### Marina

Wonderful maps! The best example of socio-political data visualization I’ve ever seen. Thanks a lot, Timo!

May 24, 2019 / Reply

**Pingback:** Como hacer un mapa muy bonito de España en ggplot2 – Pybonacci

### Thea

Thank you for this inspiration!! Its great to see what is possible. I have incorporated several of your methods into my own maps . I am curious about applying the `extendLegendWithExtremes()` function you wrote to my own maps, but can’t seem to get it to work with my data. (I am decent with ggplot but very new to gtable)

I added :
`extendLegendWithExtremes <- function(p, aDf, aVariable){...}`

to the function to replace your `map_data$avg_age_15` with `aDf$aVariable`

Can you help me identify the other parts of the function that need to be replaced with variables to make this a generic function?

Best,
Thea

February 20, 2020 / Reply

### Timo Grossenbacher

You're welcome. Well, I would have to see your code. Do you have a Git Gist or something?

February 20, 2020 / Reply

**Pingback:** Colour in mapping – MapAction

**Pingback:** Web Links for March 2020 – stephenhucker.com

### U.

…. bin sehr begeistert ….
…. ich frage mich gerade… das Relief ist dann nicht sichtbar, wenn eine magma-Farbe darüber liegt, oder?
…..d.h. es ist nur sichtbar, wenn Löcher in den bunten Farben da sind, oder?

Lg

Ulrich

May 12, 2020 / Reply

### Timo Grossenbacher

Nein, das “Magma” überlagert das Relief lediglich. Man kann die Transparenz anpassen, und zwar mit dem Alpha-Parameter (vgl. Linie mit `values = rev(magma(8, alpha = 0.8)`). Wenn man genau hinschaut, schimmert das Relief auch schon jetzt leicht durch (Transparenz 80%). LG

May 13, 2020 / Reply

### U.

…prima, danke Ihnen herzlich

May 14, 2020 / Reply

### U.

…. muss noch mal nachfassen …
d.h. man kann kein Relief darstellen innerhalb der magma-Felder, oder?

May 12, 2020 / Reply

**Pingback:** Song about R Géomatique | hypotheses.org love - Love Songs

**Pingback:** Instrumental R Géomatique | hypotheses.org Music - Instrumental Music