

# bfsMaps - Swiss Maps Themakart

by Andri Signorell

Helsana Versicherungen AG, Health Sciences, Zurich

HWZ University of Applied Sciences in Business Administration, Zurich

andri@signorell.net

March, 22<sup>th</sup>, 2020

The representation of map material with R is powerful and flexible, but also quite technically organized, and therefore challenging for occasional users. In order to reduce the complexity of the task, this R package has been created to allow Swiss cantons, districts and municipal information to be displayed more quickly and easily. Highquality maps for Switzerland are available free of charge from the Federal Office of Statistics BfS.

bfsMaps uses functions from the packages "maptools", "sp", "rgdal", "spdep", "rlang", "DescTools" available on CRAN, which must be installed additionally:

```
install.packages(c("maptools", "sp", "rgeos", "spdep", "rlang"))
```

Questions, comments and error messages are welcome! Thank you.

<b>1</b>	<b>Installation .....</b>	<b>2</b>
<b>2</b>	<b>Plot Switzerland.....</b>	<b>4</b>
<b>3</b>	<b>Plot Swiss Cantons .....</b>	<b>5</b>
<b>4</b>	<b>Cantons with capitals.....</b>	<b>6</b>
<b>5</b>	<b>Numeric Based Colorcoding for Cantons .....</b>	<b>7</b>
<b>6</b>	<b>Combine Cantons with a Dotplot .....</b>	<b>8</b>
<b>7</b>	<b>Metropolitan Regions .....</b>	<b>9</b>
<b>8</b>	<b>Language Regions .....</b>	<b>10</b>
<b>9</b>	<b>MS-Regionen .....</b>	<b>11</b>
<b>10</b>	<b>Premium Regions.....</b>	<b>12</b>
<b>11</b>	<b>Vegetation Area.....</b>	<b>13</b>
<b>12</b>	<b>Further Information in shape-files .....</b>	<b>14</b>
<b>13</b>	<b>Single Cantons .....</b>	<b>15</b>
<b>14</b>	<b>Determine neighbours .....</b>	<b>16</b>
<b>15</b>	<b>Combining geographical areas .....</b>	<b>17</b>
<b>16</b>	<b>Swiss-Locator.....</b>	<b>18</b>

Note: For all the examples in this document, `library(bfsMaps)` must be declared.

# 1 Installation

ThemaKart, the FSO cartographic centre, offers numerous map bases (geometries) for most official geographic divisions, which are used in maps and atlases of the Federal Statistical Office. An annually updated set of these geometries is freely available.

Federal Statistical Office Service ThemaKart  
Espace de l'Europe 10  
CH-2010 Neuchâtel  
Switzerland

engl:

<https://www.bfs.admin.ch/bfs/en/home/statistics/regional-statistics/base-maps/cartographic-bases.html>

german:

<https://www.bfs.admin.ch/bfs/de/home/statistiken/regionalstatistik/kartengrundlagen/basisgeometrien.html>

The Cartographic bases set contains the latest years and the state for the Population Censuses of 2000 and 2010 for the major geographic divisions, plus additional years for communes and districts, polygons for districts within the 17 largest Swiss cities. Total areas are shown in polygons which come in two versions: total area surfaces (gf, Gesamtfläche) and vegetation surfaces (vf, Vegetationsfläche). Additionally, the topographic information is enriched by numerous layers for rivers.

There is also a detailed information brochure with more information on the contents and references behind this offer, as well as a list of nomenclatures used and the application of copyrights (brochure available only in German and French).

For the use with this package the maps must be downloaded separately and unpacked into any folder.

The root folder must then be declared as an option in the R options, e.g.:

```
options(bfsMaps.base="C:/Users/andri/Documents/MapData/ThemaKart2020")
```

The bfsMaps package uses a meta file containing the path information for accessing the maps. This file is named maps.csv and can either be located in the root directory of the mapdata or in the /extdata directory of the package (usually ~/extdata, where ~ is the library path returned by system.file(package="bfsMaps")). The RequireMap() function will search for it along these two paths, taking the first file it finds.

The file consists of an optional shortname for the maps and the according path to the shapefile (with the extension .shp).

	A	B
1	name_x	path_x
2	fluss1.map	00_TOPO/K4_flusyyyyymmdd/k4flusyyyyymmdd11_ch2007.shp
3	fluss2.map	00_TOPO/K4_flusyyyyymmdd/k4flusyyyyymmdd22_ch2007.shp
4	fluss3.map	00_TOPO/K4_flusyyyyymmdd/k4flusyyyyymmdd33_ch2007.shp
5	fluss4.map	00_TOPO/K4_flusyyyyymmdd/k4flusyyyyymmdd44_ch2007.shp
6	fluss5.map	00_TOPO/K4_flusyyyyymmdd/k4flusyyyyymmdd55_ch2007.shp
7	see1.map	00_TOPO/K4_seenyyyyymmdd/k4seenyyyyymmdd11_ch2007Poly.shp
8	see2.map	00_TOPO/K4_seenyyyyymmdd/k4seenyyyyymmdd22_ch2007Poly.shp
9	stkt.pnt	00_TOPO/K4_stkt19970101/k4stkt19970101kk_ch2007Pnts.shp
10		01_INST/Gesamtfläche_gf/K4_bezk20001205_gf/K4bezk20001205gf_ch2007Poly.shp
11		01_INST/Gesamtfläche_gf/K4_bezk20001205_gf/K4bezk20001205zg_ch2007Pnts.shp
12		01_INST/Gesamtfläche_gf/K4_bezk20101231_gf/K4bezk20101231gf_ch2007Poly.shp

For the specific functions PlotKant(), PlotPolg(), etc. to work, the according shortnames must be found in the maps.csv file.

These are:

type	stem	map	pnt	vf.map	vf.pnt	used by
Rivers 1 (large)	fluss1	x				AddRivers()
Rivers 2	fluss2	x				AddRivers()
Rivers 3	fluss3	x				AddRivers()
Rivers 4	fluss4	x				AddRivers()
Rivers 5 (small)	fluss5	x				AddRivers()
Lakes (large)	see1	x				AddLakes()
Lakes (small)	see2	x				AddLakes()
Capitals of cantons	stkt		x	x	x	
Switzerland	ch	x	x	x	x	PlotCH()
Greater Regions	greg	x	x	x	x	PlotGreg()
Cantons	kant	x	x	x	x	PlotKant()
MS Regions	msre	x	x	x	x	PlotMSRe()
Districts	bezg	x	x	x	x	PlotBezg()
Municipalities	polg	x	x	x	x	PlotPolg()
Metropolitan areas	metr	x	x	x	x	PlotMetr()

vf.map denotes the maps with vegetational area, pnt are the centroid points of the specific area, map is the general map.

A default maps.csv file is included in the package, based on the filenames of ThemaKart 2020 ©.

So earlier data can be displayed as well with the same functions, but then the user must ensure himself, that the maps.csv is updated correctly and pointing to the right maps.

So the required installation steps for using the bfsMaps package are:

1. Download maps zip file from [www.bfs.admin.ch](http://www.bfs.admin.ch) (see above).
2. Unzip and place maps somewhere in your filesystem.
3. Set R options(bfsMaps.base="your directory where you placed the maps")
4. Make sure maps.csv in system.file(package="bfsMaps")/extdata is correct
5. Use package...

Switching between maps of different years can so be performed by setting the bfsMaps.base option to the required map library.

Have fun and success!

## 2 Plot Switzerland

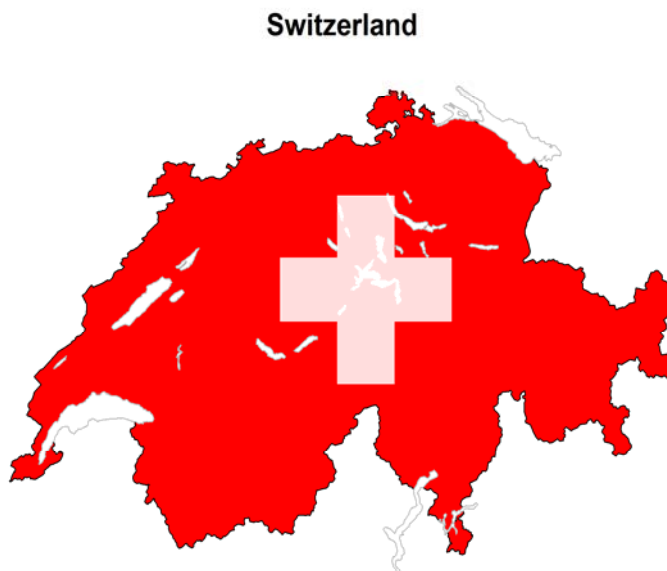
The first example represents the whole of Switzerland without any small-scale division. The waters can either be included by setting the argument `panel.first(AddLakes())` or be omitted in the `PlotCH()` function for the time being so that they can be added later by the functions `AddLakes()` and `AddRivers()`, if e.g. other colors are required.

```
library(bfsMaps)
# set the path to the map root directory
options(bfsMaps.base="C:/Users/andri/Documents/MapData/ThemaKart2020")

# Plot Switzerland map in borders of 1848
PlotCH(col="red")

sw <- 15000; xc <- 2671975; yc <- 1200600; ccol <- rgb(1,1,1,0.85)
rect(xleft=xc-sw, ytop=yc-sw, xright=xc+sw, ybottom=yc+sw, col=ccol, border=NA)
rect(xleft=(xc-2*sw)-sw, ytop=yc-sw, xright=(xc-2*sw)+sw, ybottom=yc+sw, col=ccol,
border=NA)
rect(xleft=(xc+2*sw)-sw, ytop=yc-sw, xright=(xc+2*sw)+sw, ybottom=yc+sw, col=ccol,
border=NA)
rect(xleft=xc-sw, ytop=(yc-2*sw)-sw, xright=xc+sw, ybottom=(yc-2*sw)+sw, col=ccol,
border=NA)
rect(xleft=xc-sw, ytop=(yc+2*sw)-sw, xright=xc+sw, ybottom=(yc+2*sw)+sw, col=ccol,
border=NA)

title(main="Switzerland")
```



### 3 Plot Swiss Cantons

Plot of the cantons with the canton names in the (area) centroid of the canton. The area centroids of the cantons are available in the dataset `kant.pnt` as x, y coordinates. This allows texts to be written directly on the map.

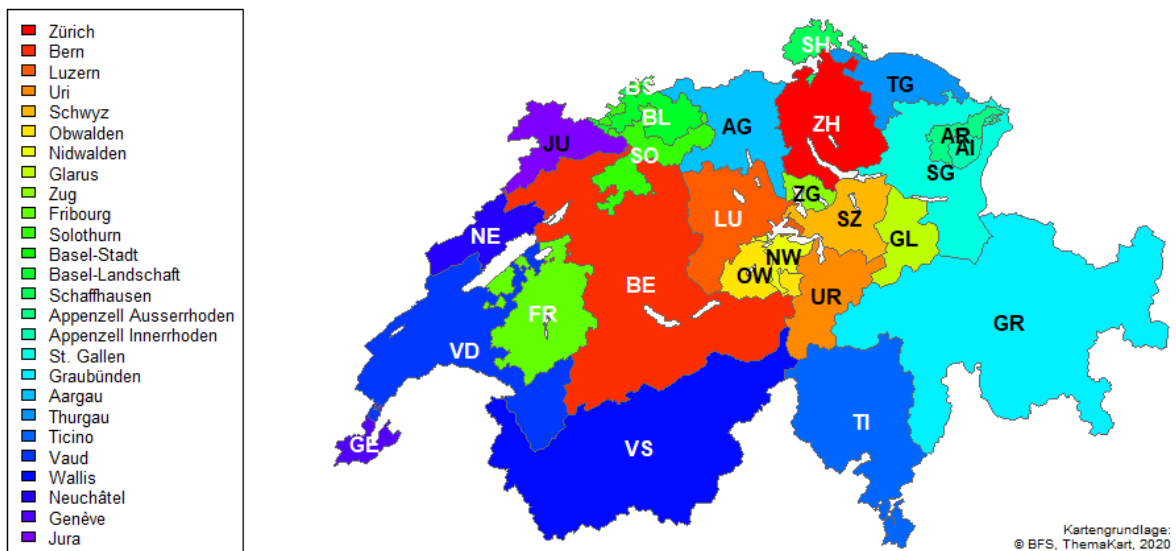
- Do everything by hand:

```
# Plot cantons in colorful way
RequireMap(c("kant.map", "kant.pnt"))
par(mar=c(5.1, 10.1, 5.1, 2.1), xpd=TRUE)
cols <- rainbow(start = 0, end = 0.75, n=26)
plot(tkart$kant.map, usePolypath=TRUE, col=cols)
legend( x="left", inset=-0.25, legend=tkart$kant.map@data$ID1,
        fill=cols, bg="white", cex=0.8, xpd=NA )

# Put text on the centroids of the cantons
text(tkart$kant.pnt, labels=unique(d.bfsrg$kt_x), col=TextContrastColor(cols), font=1)
```

- Use specialized function `bfsMaps::PlotKant()`:

```
Mar(left=15)
b <- PlotKant(col = cols, border = "grey40")
text(b, labels = kt, col=TextContrastColor(cols), font=2)
legend( x="left", inset=-0.325, legend=tkart$kant.map@data$ID1,
        fill=cols, bg="white", cex=0.8, xpd=NA )
```



## 4 Cantons with capitals

Plot of the cantons with display of their capitals. The coordinates of the capitals are stored in the file `stkt.pnt` and can be extracted with `stkt.pnt@coords`.

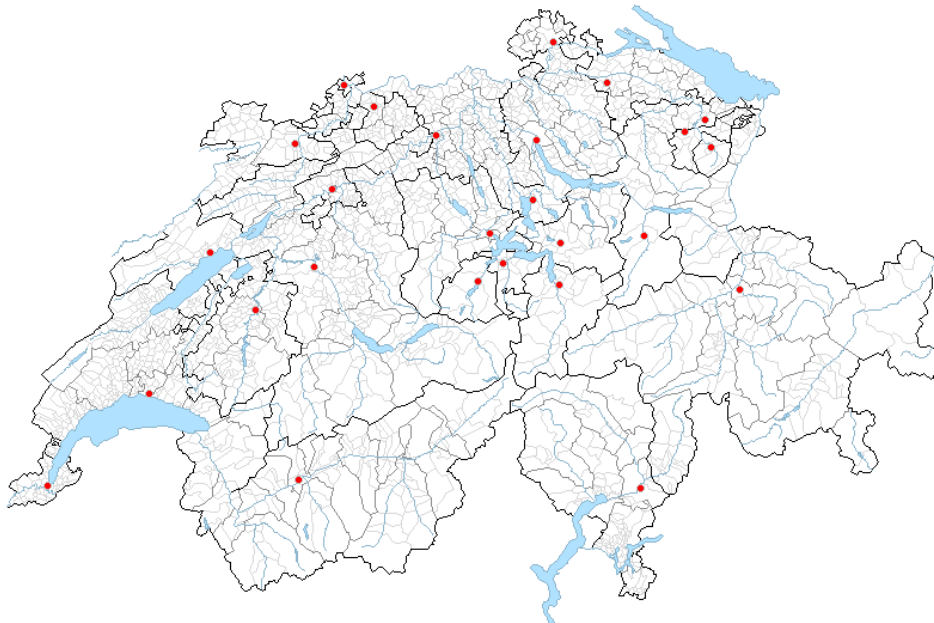
The chart is constructed so, that first the smallest spatial units (municipalities) are drawn, then the next larger ones (districts) and finally the top units (cantons). Capitals and waters are drawn over the map last.

```
RequireMap(c("polg.map", "bezg.map", "kant.map", "stkt.pnt"))
```

```
# Plot municipalities
plot(tkart$polg.map, border="grey85" )
plot(tkart$bezg.map, border="grey55", add=TRUE )
plot(tkart$kant.map, border="black", lwd=1, add=TRUE )
```

```
AddLakes()
AddRivers()
```

```
# Kantons-Hauptorte
points(tkart$stkt.pnt@coords, pch=21, col="grey", bg="red")
```



## 5 Numeric Based Colorcoding for Cantons

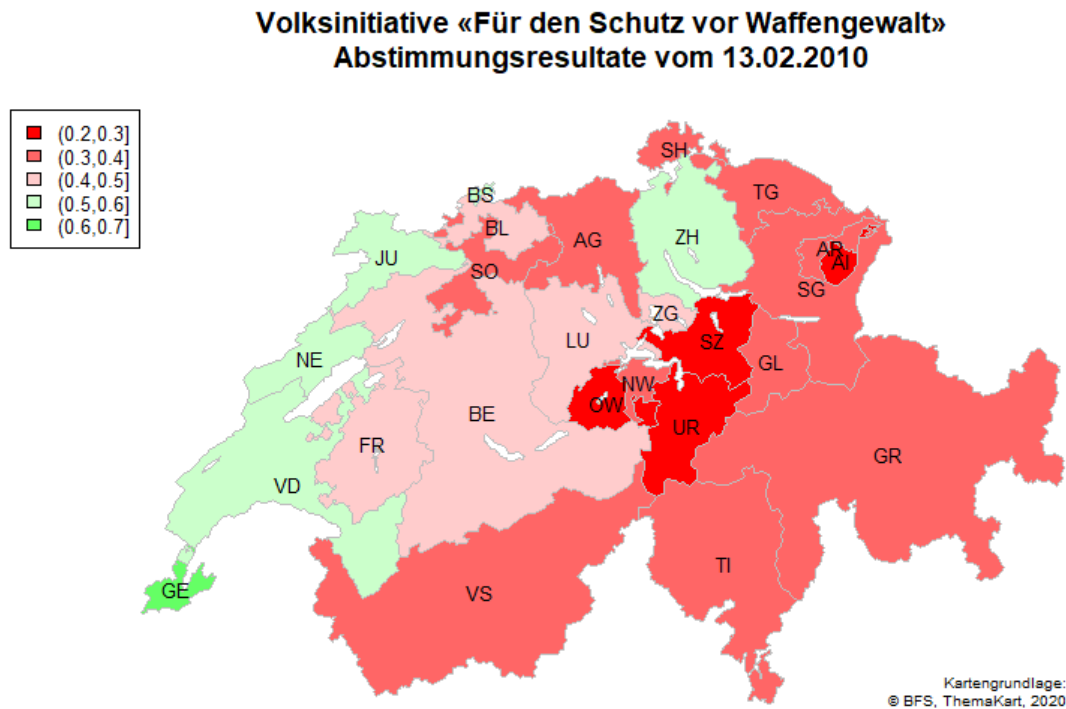
Plot of the cantons with a colour-coded variable and table in the same plot.

```
# Combine map with tabular results
waffen <- c(0.504,0.406,0.401,0.294,0.291,0.281,0.312,0.302,0.428,0.411,0.350,0.589,
0.452,0.390,0.378,0.277,0.392,0.350,0.386,0.345,0.365,0.537,0.381,0.532,0.610,0.520)
names(waffen) <- levels(d.bfsrg$kt_x)
cols <- colorRampPalette(c("red", "white", "green"), space = "rgb")(6)[
  cut(waffen, breaks=c(seq(0.2,0.8,0.1))) ]

layout( matrix(c(1,2), nrow=2, byrow=TRUE), heights=c(2,0.5), TRUE)

usr <- par(mar=c(0,4.1,4.1,2.1))
PlotKant(col=cols, main="Volksinitiative «Für den Schutz vor
Waffengewalt»\nAbstimmungsergebnisse vom 13.02.2010" )
text(tkart$kant.pnt, labels=unique(d.bfsrg$kt_x), cex=0.8)
legend(x="topleft", inset=0.02, cex=0.8,
      fill=colorRampPalette(c("red", "white", "green"), space = "rgb")(6)[1:5],
      legend=levels( cut(waffen, breaks=c(seq(0.2,0.8,0.1))))[1:5] )
par(usr)

usr <- par(mar=c(0,0,0,0))
out <- capture.output(round(waffen[1:13],3), round(waffen[14:26],3))
plot( 1, axes=FALSE, frame.plot=FALSE, type="n", xlab="", ylab="" )
text( labels=out, x=1, y= 0.85 + rev(1:length(out)) * strheight( "S", cex=1.0 ) * 1.3,
      adj=c(0.5,0.5), family="mono", cex=0.8 )
par(usr)
layout(1)
```



```
[1] 0.504 0.406 0.401 0.294 0.291 0.281 0.312 0.302 0.428 0.411 0.350 0.589 0.452
[1] 0.390 0.378 0.277 0.392 0.350 0.386 0.345 0.365 0.537 0.381 0.532 0.610 0.520
```

## 6 Combine Cantons with a Dotplot

A presentation of cantonal data that has proven to be very useful in practice is the simultaneous presentation of maps and a dotplot in which the numerical value to be depicted can be made well visible with possible confidence intervals. The graphic area is divided for this purpose with `layout()`. This process is encapsulated in the function `PlotMapDot()` and simplifies the coding.

```
# get some proportions and simulated confidence intervals
set.seed(1964)
ptab <- data.frame(val=runif(n=26, max = 0.9))
ptab$lci <- ptab$val - runif(n = 26, 0.01, 0.1)
ptab$uci <- ptab$val + runif(n = 26, 0.01, 0.1)
rownames(ptab) <- kt

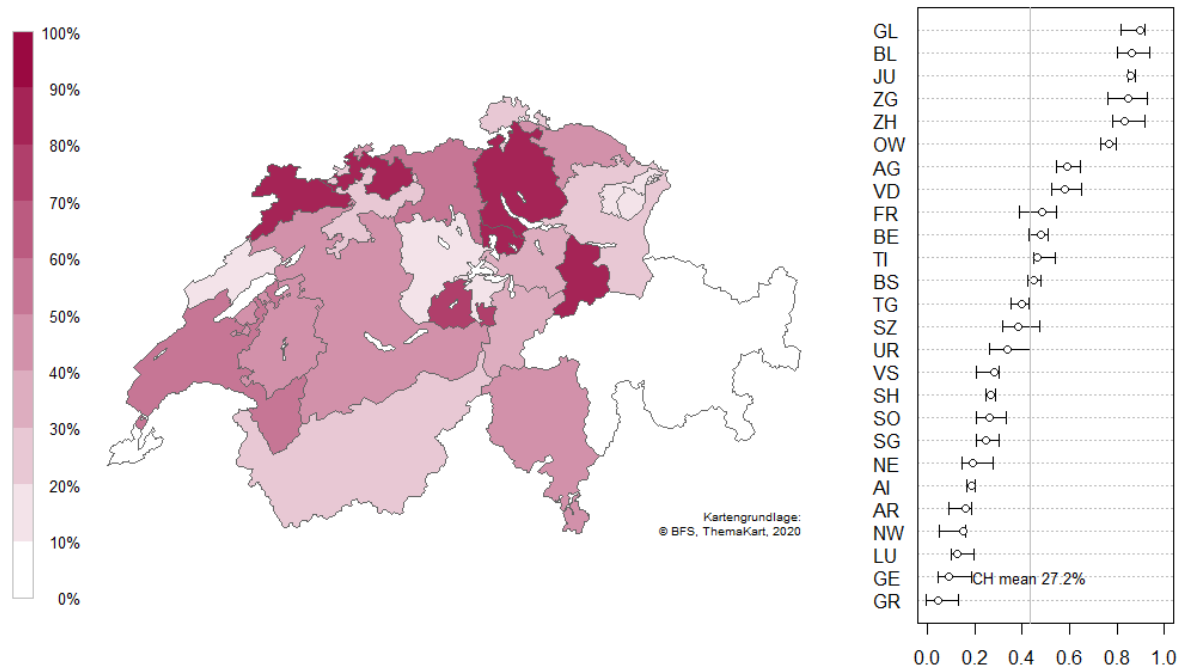
# ... and the map
PlotMapDot()
cols <- colorRampPalette(colors = c("white", hred))(10)

PlotKant(rownames(ptab), col=FindColor(ptab$val, cols = cols, min.x=0, max.x=1),
border="grey40")
ColorLegend(x="left", width=10000, labels=paste(seq(0, 100, 10), "%", sep=""),
cols=cols, cex=0.8, adj=c(1,0.5), frame="grey", inset=c(-0.09, 0))

ptab <- Sort(ptab, decreasing=TRUE)
PlotDot(ptab$val, labels = rownames(ptab),
args.errbars = list(from=ptab$lci, to=ptab$uci, mid=ptab$val),
cex=1, xlim=c(0, 1), bg="white", pch=21, col="grey10")

abline(v=mean(ptab$val), col="grey")
text(x=mean(ptab$val), y=2, "CH mean 27.2%", cex=0.8)
```

Any extrapolated distribution CH in 2020



## 7 Metropolitan Regions

Plot metropolitan regions Switzerland.

```
# Swiss metropolitan areas
RequireMap("metr.map")

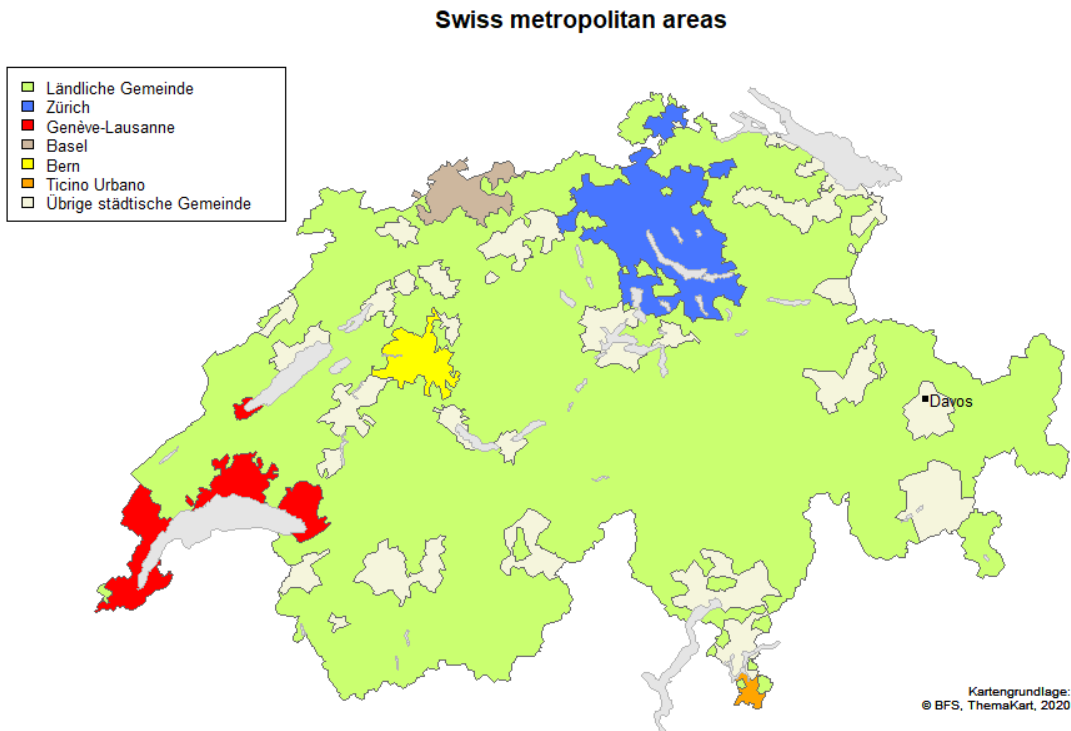
tkart$metr.map@data$ID1 <-
  c("Ländliche Gemeinde", "Zürich", "Genève-Lausanne", "Basel", "Bern",
    "Ticino Urbano", "Übrige städtische Gemeinde")

# we have to prepare the background here, for some reasons...
PlotCH(col="beige", border=NA)
plot(tkart$metr.map, add=TRUE, border="grey40",

col=c("darkolivegreen1", "royalblue1", "red", "bisque3", "yellow", "orange", "wheat1")[as.numeri
c(tkart$metr.map@data$ID0)+1])
AddLakes(col="grey90", border="grey70")
legend( x="topleft", legend=tkart$metr.map@data$ID1,
  fill=c("darkolivegreen1", "royalblue1", "red", "bisque3", "yellow", "orange", "beige"),
  bg="white", cex=0.8, xpd=TRUE )

title(main="Swiss metropolitan areas")
BfSStamp()
```

```
# Add some single points by coordinates
points(2782783,1185993,pch=15, cex=0.8 )
text(2782783,1185993," Davos", cex=0.8, adj=c(0,0.5) )
```



## 8 Language Regions

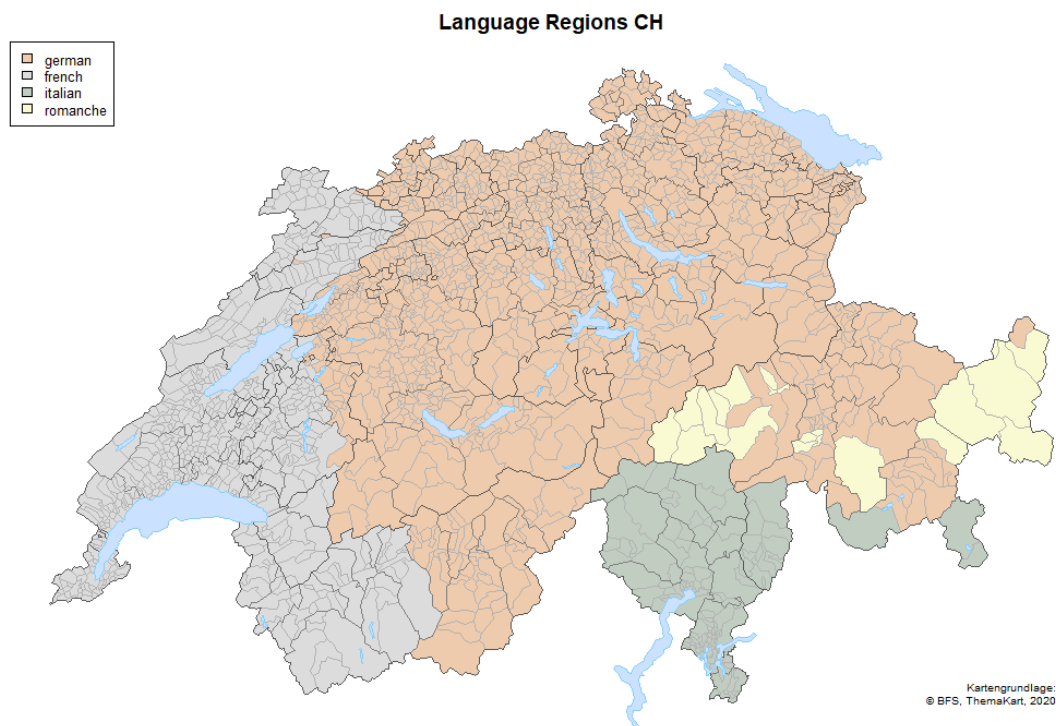
Display language regions or any other data from the attached recordset `d.bfsrg`.

```
cols <- c("peachpuff2","gainsboro","honeydew3","lightgoldenrodyellow")

# Use Raumregions data
PlotPolg(d.bfsrg$bfs_nr, col=cols[d.bfsrg$sprgeb_c],
        border="grey70", main="Language Regions CH" )

RequireMap(c("bezk.map", "kant.map"))
plot(tkart$bezk.map, border="grey50", add=TRUE)
plot(tkart$kant.map, border="grey35", add=TRUE)

AddLakes(col="lightsteelblue1", border="lightskyblue" )
legend(x="topleft", legend=c("german", "french", "italian", "romanche"),
      bg="white", cex=0.8, fill=cols)
```



## 9 MS-Regionen

Plot MS regions in Switzerland, a special, artificial spatial partitioning of the Swiss Federal Statistical Office.

```
RequireMap(c("msre.map", "msre.pnt", "kant.map"))

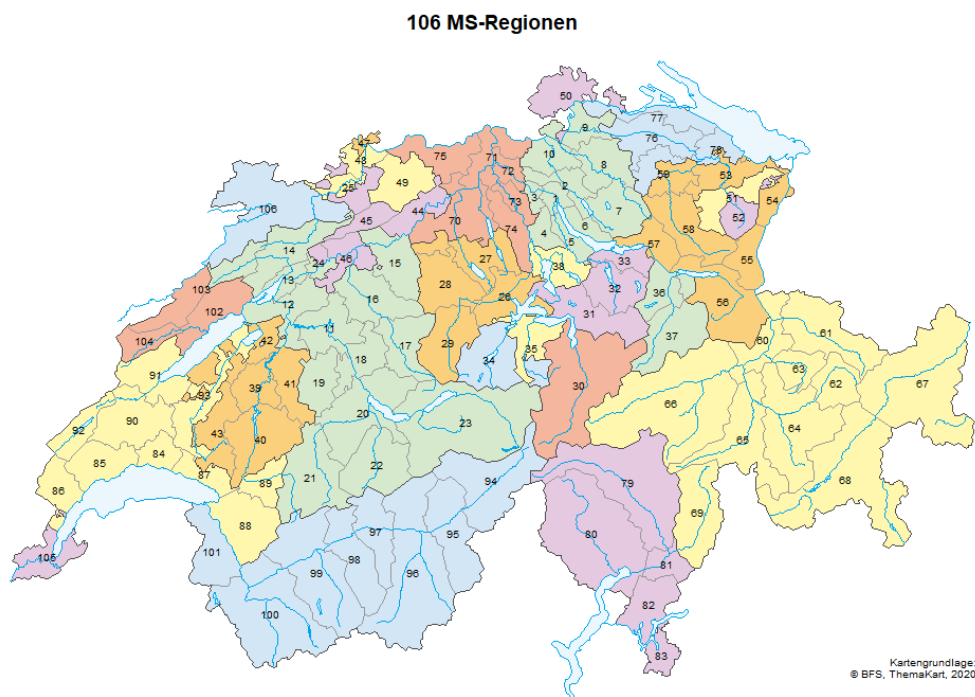
cols <- c(y=rgb(255,247,174,max=255), o=rgb(251,208,124,max=255),
v=rgb(228,201,224,max=255),
      b=rgb(211,230,246,max=255), g=rgb(215,233,205,max=255),
r=rgb(244,182,156,max=255),
      p=rgb(255,248,236,max=255))

PlotKant(1:26, col=cols[c("g","g","o","r","v","b","y","g","y","o","v","o","y","v",
      "y","v","o","y","r","b","v","y","b","r","v","b")],
      border="grey20", lwd=1, pbg=cols["p"], main="106 MS-Regionen")

plot(tkart$msre.map, add=TRUE, border="grey60")
plot(tkart$kant.map, add=TRUE, border="grey30")

AddLakes(1:2, col=rgb(235,247,253, max=255), border=rgb(0,166,235,max=255))
AddRivers(1:5, col=rgb(0,166,235,max=255))

text(x=tkart$msre.pnt@coords[,1], y=tkart$msre.pnt@coords[,2],
      labels=tkart$msre.pnt@data$ID0, cex=0.6)
```



## 10 Premium Regions

For plotting the premium regions as own areas, we have first to combine the polygons of the municipalities. The data is made available by the Swiss Federal Office of Public Health FOPH – BAG.

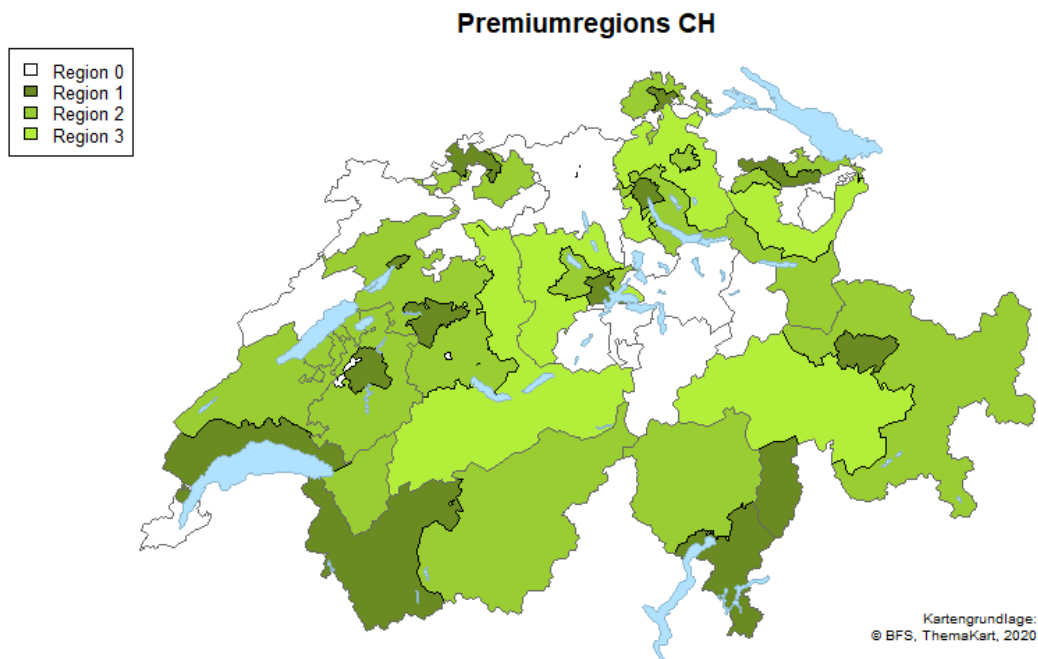
It is included in the internal dataset d.bfsrg of the package bfsMaps.

```
# start with a white the background
PlotCH(col="white", main="Premiumregions CH")

# Combine the municipalities to the premium regions
plot(CombinePolg(id=d.bfsrg$gem_id, g=d.bfsrg$preg_c),
     col=c("white", "olivedrab4", "olivedrab3", "olivedrab2"), add=TRUE)

legend(x="topleft", fill=c("white", "olivedrab4", "olivedrab3", "olivedrab2"), cex=0.8,
      legend=c("Region 0", "Region 1", "Region 2", "Region 3") )

# overplot the canton borders
PlotKant(col=NA, border="grey40", add=TRUE)
# and add waters
AddLakes()
```



## 11 Vegetation Area

For the vegetation area, uninhabited, vegetation-free areas (lakes, glaciers, rock) are subtracted from the total area and not shown, thus achieving a more realistic spatial representation for most statistical results. In the map below the unfertile area is shown in dark.

```
PlotCH(col="wheat3", col.vf="wheat", border="wheat3", main="CH Vegetation Area")
AddRivers()
AddLakes()
PlotKant(col=NA, border="wheat4", add=TRUE, lwd=1)
```



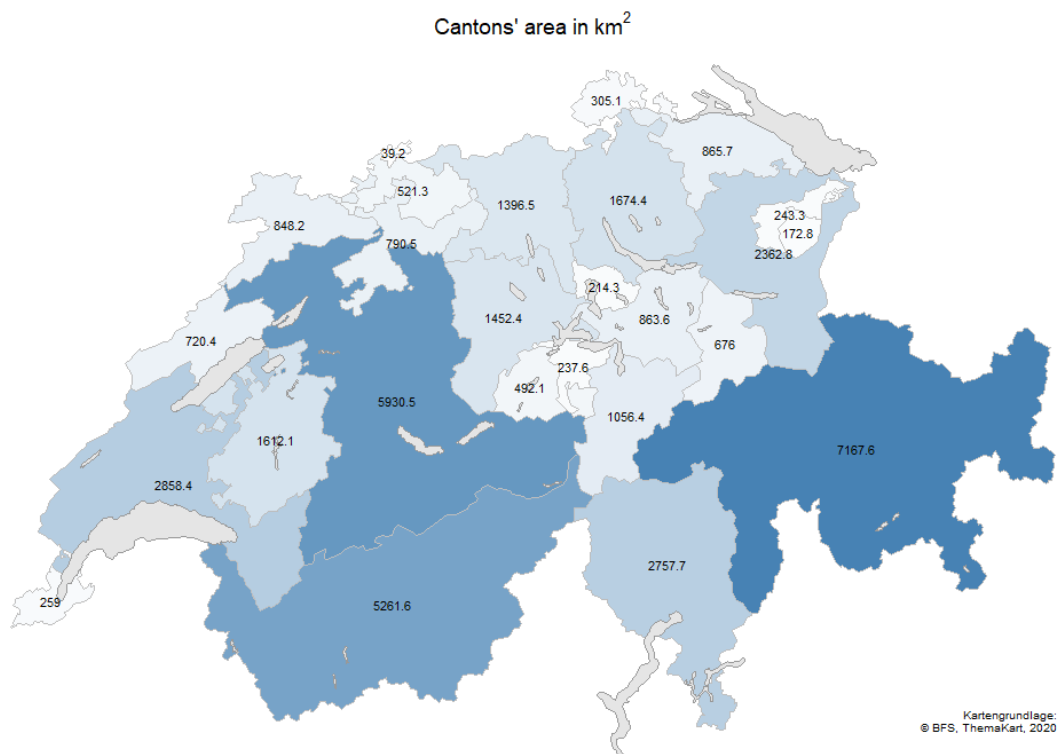
## 12 Further Information in shape-files

In the shape files you will find further information that can be extracted from the slots, however in a rather complicated way. Here is an example for the area of the cantons.

```
# Use data out of shape files slots (here: cantons' area)
acant <- sapply(slot(tkart$kant.map, "polygons"), function(x) slot(x, "area"))

# plot cantons
PlotKant(1:26, col=colorRampPalette(c("white", "steelblue"),
                                     space = "rgb")(720)[trunc(acant / 10000000)],
        main= expression(paste( "Cantons' area in ", km^2)) )

AddLakes(col="grey90", border="grey60")
text(t(sapply( slot(tkart$kant.map, "polygons"), function(x) slot(x, "labpt"))),
      labels=round(acant/1E6,1), cex=0.7)
```

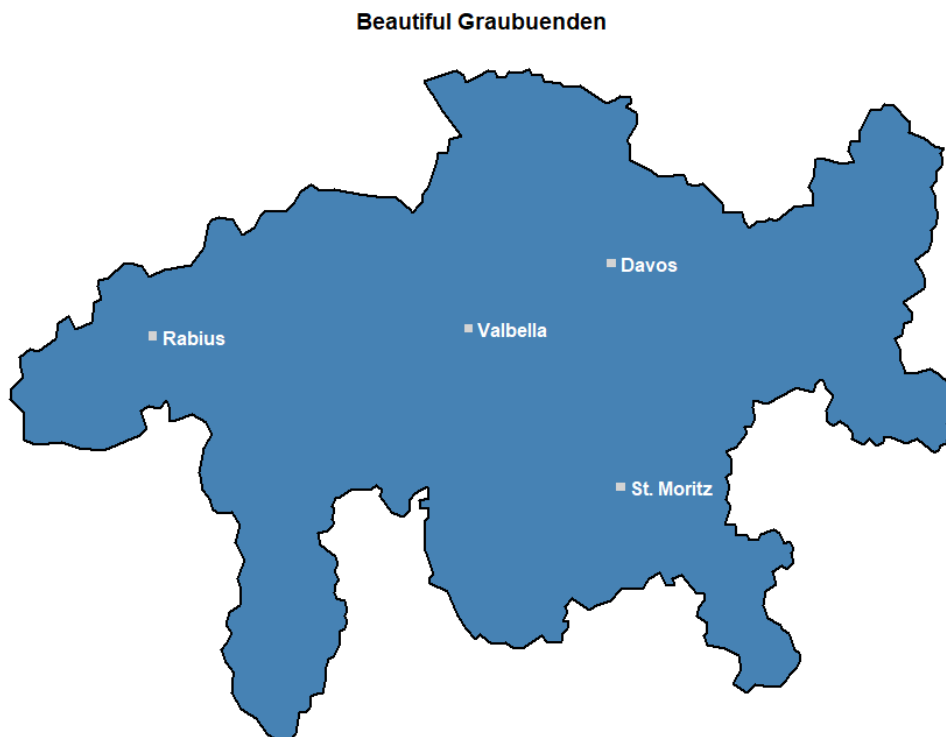


## 13 Single Cantons

We can also plot only single cantons.

```
# Plot single cantons
kant.gr <- slot(tkart$kant.map,"polygons")[[18]]@Polygons[[1]]@coords
plot(kant.gr, asp=1, axes=FALSE, frame.plot=FALSE, type="n", xlab="", ylab="",
main="Beautiful Graubünden" )
polygon(kant.gr, col="steelblue", lwd=2 )

loctext <- function(x, y, text){
  points(x, y, pch=15, col="lightgrey" )
  text(x, y, text, adj=c(0,0.5), col="white", font=2)
}
loctext(2782783,1185993," Davos")
loctext(2761412,1176112," Valbella")
loctext(2784192,1152424," St. Moritz")
loctext(2714275,1175027," Rabius")
```

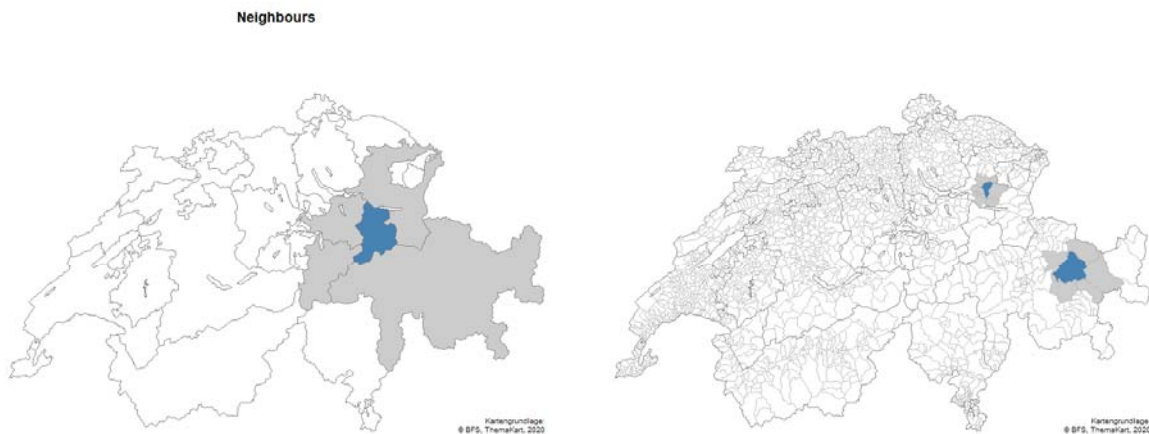


## 14 Determine neighbours

To determine the neighbours adjacent to a region, the `Neighbours()` function can be used. The function requires the specification of the map and a vector with the IDs of the regions whose neighbours are to be determined. The result consists of a list. For each region a vector with the IDs of the direct neighbours is returned.

```
par(mfrow=c(1,2))

# Determine neighbours, here for the canton Glarus (id=8)
nbs <- Neighbours(map=tkart$kant.map, id=8)[[1]]
PlotKant(c(8, nbs), col=c("steelblue", rep("grey80", length(nbs) )),
        main="Neighbours")
PlotKant(border="grey50", add=TRUE)
```



The determination is not limited to cantons, but can be used with all spatial divisions. If neighbours are sought for several municipalities, several ids (municipality numbers) can be passed to the function as vectors. Also here `Neighbours()` returns the neighbours organized as a list.

```
# works as well for communities and for a vector of ids
polg_id <- c(3851, 3352)
nbs <- Neighbours(tkart$polg.map, polg_id)

PlotPolg(c(polg_id, unlist(nbs)), col=c(rep("steelblue", 2), rep("grey80",
length(unlist(nbs))))),
        border = "grey70")

PlotKant(border="grey50", add=TRUE)
```

## 15 Combining geographical areas

With the functions `CombinePolg()` and `CombineKant()`, cantonal or municipal areas can be combined into new units. This is useful, for example, for the display of premium regions or hospital service areas. Here it is sufficient to pass the region ID as an argument to the function and to provide grouping information for each region ID. The function then compiles the regions and returns them as plotable map objects.

The determination is not limited to cantons or municipalities, but can be done with all spatial divisions using the more general function `CombinePolygons()`.

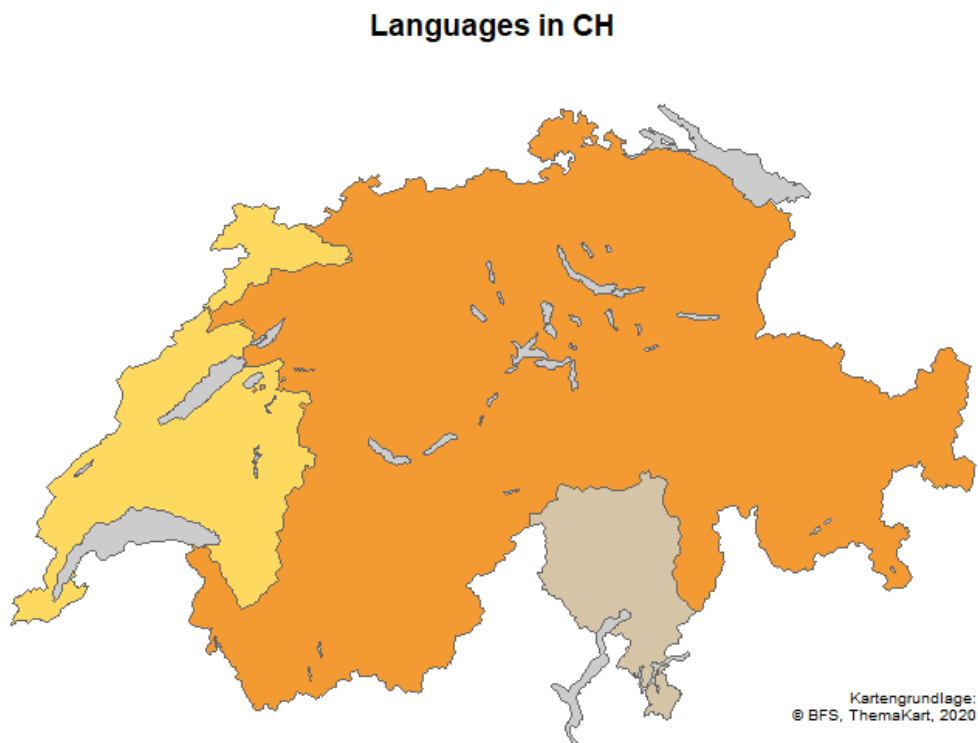
```
# the language of the municipalities, aggregated by cantons

tkc <- table(d.bfsrg$kt_c, d.bfsrg$sprgeb_x)

# Zusammenfassung nach Mehrheit pro Kanton
grp <- levels(d.bfsrg$sprgeb_x)[apply(tkc, 1, which.max)]

# Kantone kombinieren und plotten
plot(CombineKant(rownames(tkc), grp), col=SetAlpha(c(horange, hyellow, hecru), 0.8),
     border="grey40", main="Languages in CH")

# copyright is mandatory
BfSStamp()
# waters make the map more plastic...
AddLakes(col = "grey80", border = "grey40")
```



## 16 Swiss-Locator

Particularly in data cleansing at the municipal level, it happens that municipal areas remain white and one wants to know quickly which municipality(s) are involved.

The function `SwissLocator()` helps here. You can interactively click on any point on the CH map and get back information about the coordinates, canton and municipality affiliation etc. as a result.

```
# start the locator
SwissLocator()
```

[... do some clicks on the map and finish by pressing [ESC] ...]

	x	y	bfs_nr	gemeinde_name_x	bezirk_x
ms_reg_x kt_x					
319 700985.8	179998.5	584	Lauterbrunnen	Verwaltungskreis Interlaken-Oberhasli	
Oberland-Ost	BE				
431 666420.5	174423.4	784	Innertkirchen	Verwaltungskreis Interlaken-Oberhasli	
Oberland-Ost	BE				
631 635200.3	158255.8	1216	Silenen		Kanton Uri
Uri	UR				