tmap in a nutshell

With the tmap package, thematic maps can be generated with great flexibility. The syntax for creating plots is similar to that of `ggplot2`. The add-on package tmaptools contains tool functions for reading and processing shape files.

Static plot and interactive view

Each map can be plotted as a static map and shown interactively. These two modes, called the `"plot"` mode and the `"view"` mode respectively, are described in `vignette("tmap-modes")`. In the remainder of this vignette, the `"plot"` mode is used.

Shape objects

We refer to **shape objects** as objects from the class `Spatial` or `Raster`, respectively from the `sp` and the `raster` package. The supported subclasses are:

	Without data	With data
Polygons	SpatialPolygons	SpatialPolygonsDataFrame
Points	SpatialPoints	SpatialPointsDataFrame
Lines	SpatialLines	SpatialLinesDataFrame
Raster	SpatialGrid	SpatialGridDataFrame
Raster	SpatialPixels	SpatialPixelsDataFrame
Raster		RasterLayer
Raster		RasterBrick
Raster		RasterStack

Also simple features from the new `sf` package are supported. Obviously, shape objects with data (the right-hand side column) are recommended, since data is what we want to show.

Load shape object of Europe (contained in this package):

```
data(Europe)
```

The `tmaptools` package contains functions to read ESRI shape files and process them. Many of these processing steps can also be done directly in `tmap`, e.g. setting to a different map projection.

Quick thematic map

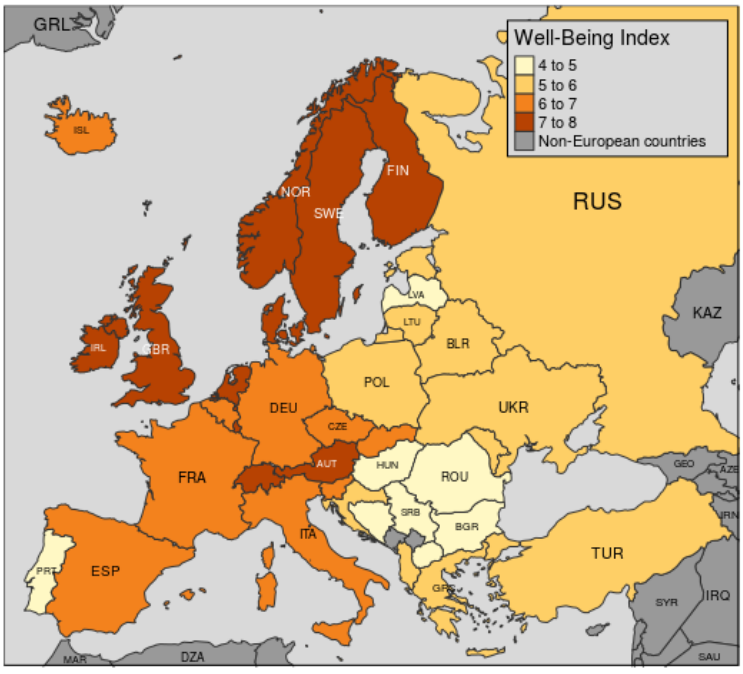
The plotting syntax is based on that of `ggplot2`. The `qtm` function is `tmap`'s equivalent to `ggplot2`'s `qplot`. The first, and only required argument is a shape object:

```
qtm(Europe)
```

So, by default, the polygons (in case the shape object is a SpatialPolygonsDataFrame) are filled with light grey, and the polygon borders are drawn in dark grey.

A choropleth is created with the following code:

```
qtm(Europe, fill="well_being", text="iso_a3", text.size="AREA", format="Europe", style="gray",
    text.root=5, fill.title="Well-Being Index", fill.textNA="Non-European countries")
```



In this code, `fill`,`text`, and `text.size` serve as aesthetics. Both `well_being` and `iso_a3` are variables of the data contained in the shape object `Europe`. A color palette, in this case the qualitative palette from yellow to brown, is mapped to the values of `well_being`. The variable `iso_a3` contains the text labels, in this case the country codes. The value `"AREA"` is a derived variable that contains the polygon area sizes. So text is sized increasingly with country area size.

The two arguments `format` and `style` are predefined layout settings (see layout).

The other arguments are passed on to the layer functions, which are described below. The argument `text.root` determines how text size is increased; in this case, the fifth root of the area sizes are taken. The result is that the text label for Russia does not dominate the other text labels. The `fill.title` argument is the title for the `fill`-legend. The argument `fill.textNA` is the legend text for missing values.

The function `qtm` offers the same flexibility as the main plotting method (to be described next). However, for more complex plots, the main plotting method is recommended for its readability.

Plotting with tmap elements

The main plotting method, the equivalent to `ggplot2`'s `ggplot`, consists of elements that start with `tm_`. The first element to start with is `tm_shape`, which specifies the shape object. Next, one, or a combination of the following drawing layers should be specified:

Drawing layer	Description	Aesthetics
Base layer		
`tm_polygons`	Draw polygons	col
`tm_symbols`	Draws symbols	size, col, shape
`tm_lines`	Draws polylines	col, lwd
`tm_raster`	Draws a raster	col
`tm_text`	Add text labels	text, size, col
Derived layer		
`tm_fill`	Fills the polygons	see tm_polygons
`tm_borders`	Draws polygon borders	none
`tm_bubbles`	Draws bubbles	see tm_symbols
`tm_squares`	Draws squares	see tm_symbols
`tm_dots`	Draws dots	see tm_symbols
`tm_markers`	Draws markers	see tm_symbols and tm_text
`tm_iso`	Draws iso/contour lines	see tm_lines and tm_text

Each aesthetic can take a constant value or a data variable name. For instance, `tm_fill(col="blue")` colors all polygons blue, while `tm_fill(col="var1")`, where `"var1"` is the name of a data variable in the shape object, creates a choropleth. If a vector of constant values or variable names are provided, small multiples are created.

The following layers are map attributes:

Attribute layer	Description
`tm_grid`	Add coordinate grid lines
`tm_credits`	Add credits text label
`tm_compass`	Add map compass
`tm_scale_bar`	Add scale bar

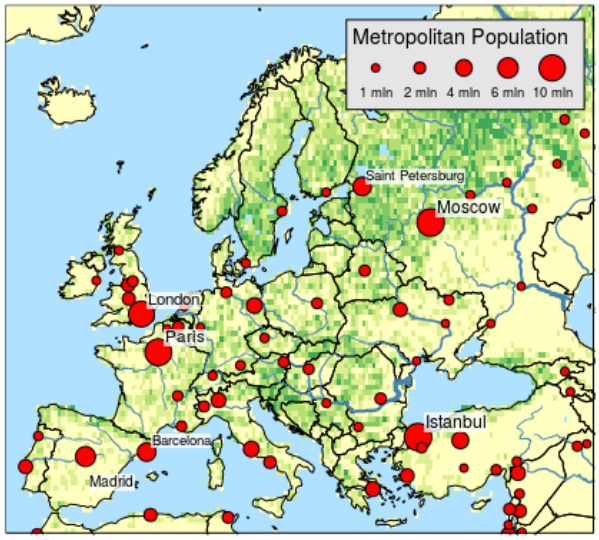
The last plot is reproduced as follows:

```
tm_shape(Europe) +
    tm_polygons("well_being", textNA="Non-European countries", title="Well-Being Index") +
    tm_text("iso_a3", size="AREA", root=5) +
tm_format_Europe() +
tm_style_grey()
```

We refer to `tm_shape` and its subsequent drawing layers as a **group**. Multiple groups can be stacked. To illustrate this, let's create a topographic map of Europe:

```
data(land, rivers, metro)

tm_shape(land) +
    tm_raster("trees", breaks=seq(0, 100, by=20), legend.show = FALSE) +
tm_shape(Europe, is.master = TRUE) +
    tm_borders() +
tm_shape(rivers) +
    tm_lines(lwd="strokelwd", scale=5, legend.lwd.show = FALSE) +
tm_shape(metro) +
    tm_bubbles("pop2010", "red", border.col = "black", border.lwd=1,
        size.lim = c(0, 11e6), sizes.legend = c(1e6, 2e6, 4e6, 6e6, 10e6),
        title.size="Metropolitan Population") +
    tm_text("name", size="pop2010", scale=1, root=4, size.lowerbound = .6,
        bg.color="white", bg.alpha = .75,
        auto.placement = 1, legend.size.show = FALSE) +
tm_format_Europe() +
tm_style_natural()
```



Things to learn from this code:

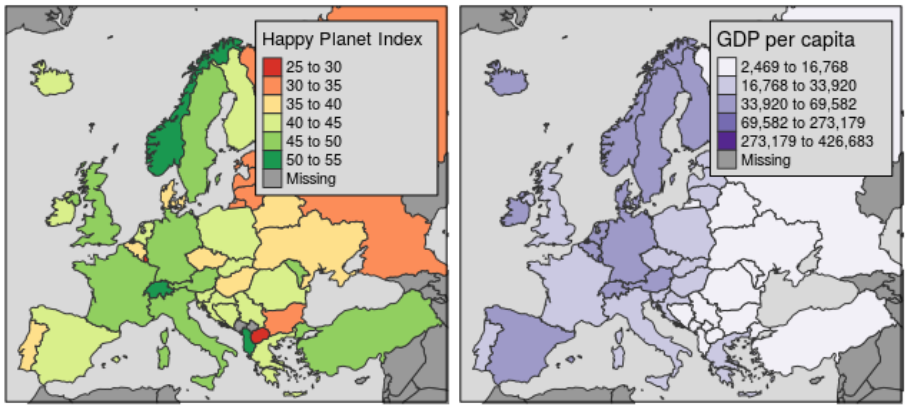
- This plot has 4 groups of layers, respectively from the shape objects land, Europe, rivers, and metro. The order of (groups of) layers corresponds to the plotting order.
- The shape objects can have different projections, and can also cover different areas (bounding boxes). Both the projection and the covered area are by default taken from shape object defined in the first `tm_shape`, but in this case in the second `tm_shape` since `is.master=TRUE`. Notice that the other shapes, i.e. `land`, `rivers`, and `metro` also contains outside Europe: see for instance `qtm(rivers)`.
- The element `tm_layout` controls all layout options such as fonts, legends, margins, and colors. The element `tm_format_Europe` is a wrapper function with some other defaults that are tailored for Europe: for instance, the legend is placed top right. The element `tm_layout_natural` is another wrapper function of `tm_layout` used to specify map-independent layout layout settings, such as default colors. See also layout.

Small multiples

Small multiples are generated in three ways:

1. By assigning multiple values to at least one of the aesthetic arguments:

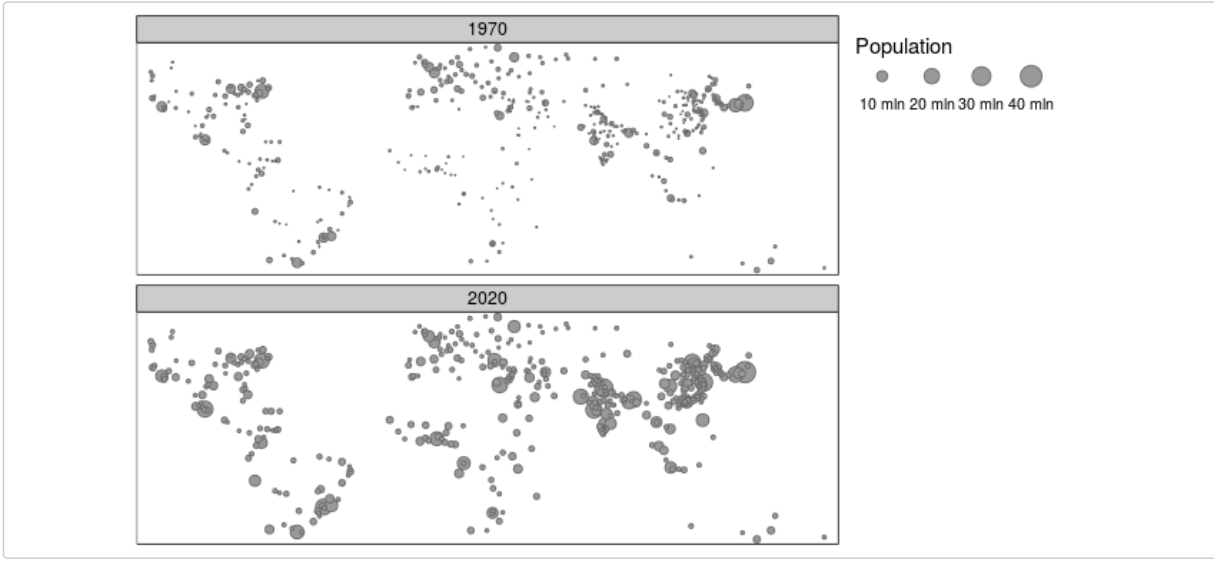
```
tm_shape(Europe) +
    tm_polygons(c("HPI", "gdp_cap_est"),
        style=c("pretty", "kmeans"),
        palette=list("RdYlGn", "Purples"),
        auto.palette.mapping=FALSE,
        title=c("Happy Planet Index", "GDP per capita")) +
tm_format_Europe() +
tm_style_grey()
```



In this case, two independent maps are created, with different scales. All arguments of the layer functions can be vectorized, one for each small multiple. Arguments that normally can take a vector, such as `palette` should be placed in a `list`.

This method is normally used to show two totally different variables, such as in this example Happy Planet Index and GDP. However, it is also possible to show variables that are related, as if they are subsets from the same data:

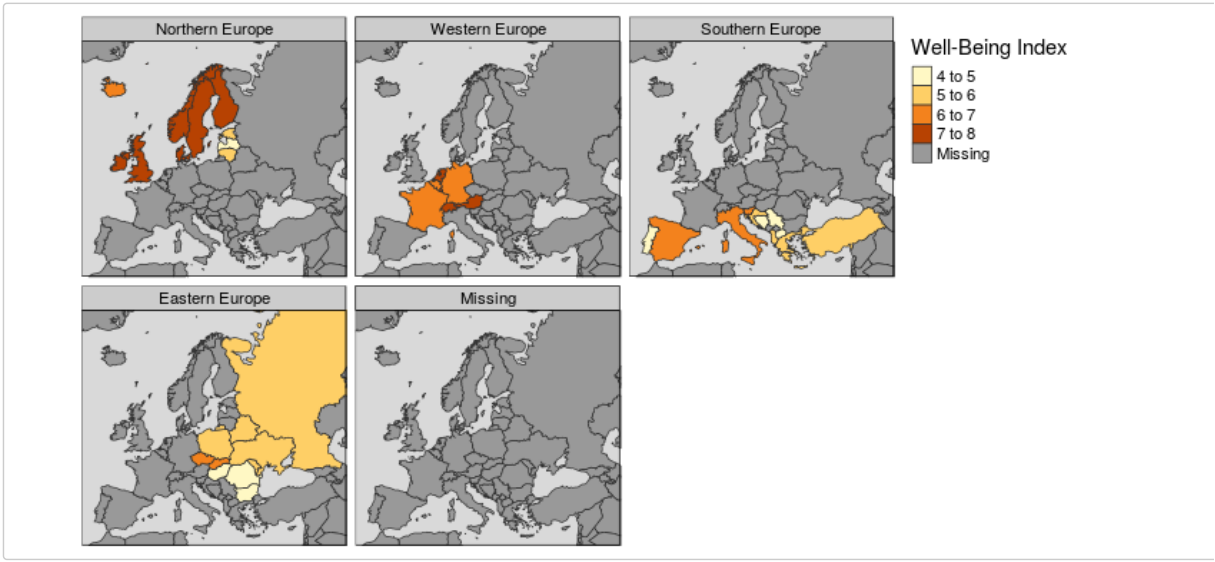
```
tm_shape(metro) +
    tm_bubbles(size=c("pop1970", "pop2020"), title.size="Population") +
    tm_facets(free.scales=FALSE) +
tm_layout(panel.labels=c("1970", "2020"))
```



Notice that this plot uses panels and that the common legend is plot outside of the maps.

2. By defining a group-by variable in `tm_facets`:

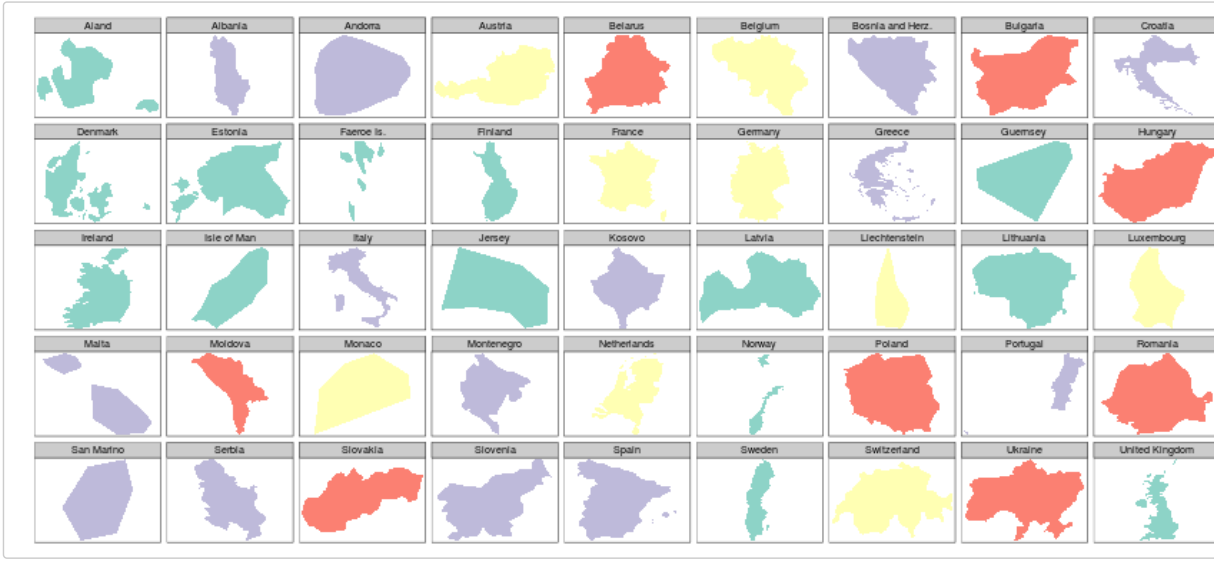
```
tm_shape(Europe) +
    tm_polygons("well_being", title="Well-Being Index") +
    tm_facets("part", free.coords=FALSE) +
tm_style_grey()
```



This plot also uses the panel layout with the common legend drawn outside the maps. These options can be changed with the arguments `panel.show` and `legend.outside` of `tm_layout`. By default, the panel/external legend layout is used when the group-by variable is specified, since in that case, the multiples share a common legend.

The scales of each aesthetic argument can be set to either fixed or free, and also, the coordinate ranges of the small multiples. By default, the latter is set to `TRUE`:

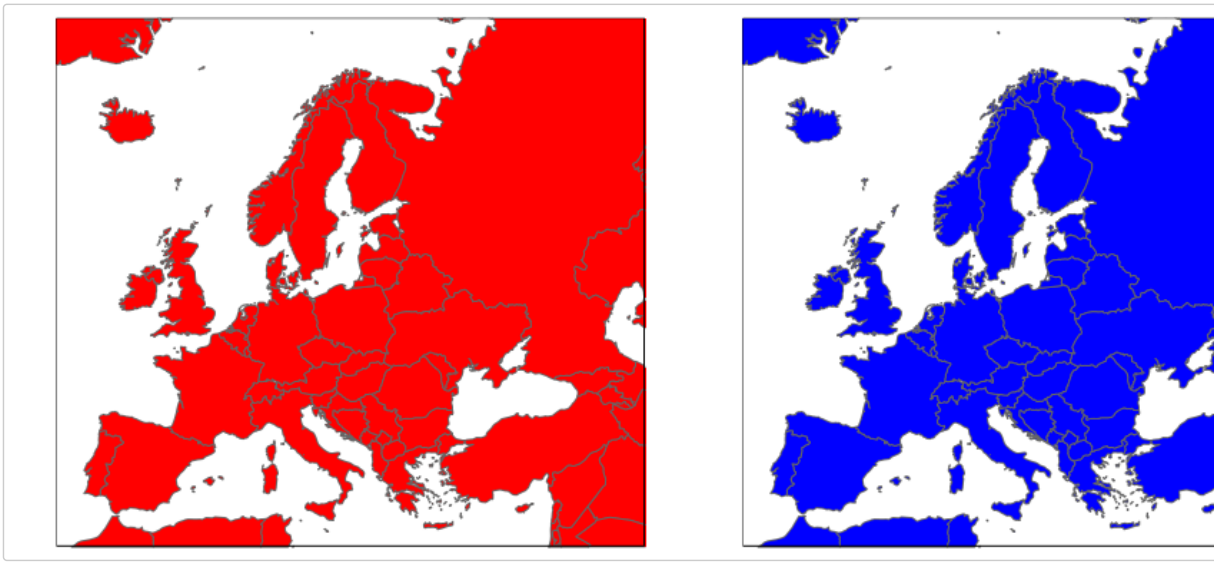
```
tm_shape(Europe[Europe$continent=="Europe",]) +
    tm_fill("part", legend.show = FALSE) +
    tm_facets("name")
```



The argument `drop.units` is used to drop all non-selected spatial units. If `drop.shapes=FALSE` then neighboring countries are also visible.

3. By creating multiple stand-alone maps with `tmap_arrange`:

```
tm1 <- qtm(Europe, fill = "red")
tm2 <- qtm(Europe, fill = "blue")
tmap_arrange(tm1, tm2, asp = NA)
```

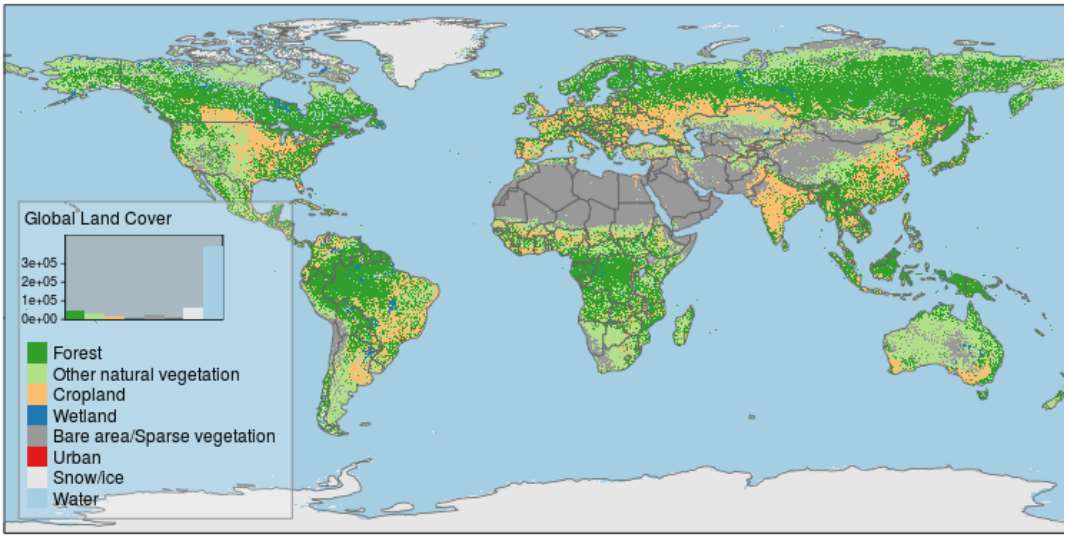


Map layout

The layout of the thematic map can be changed with `tm_layout` or one of its wrapper functions. In the next example we use two of these wrapper functions, one for the overall format of world maps, and one for the legend.

```
data(land)
data(World)
pal8 <- c("#33A02C", "#B2DF8A", "#FDBF6F", "#1F78B4", "#999999", "#E31A1C", "#E6E6E6", "#A6CEE3")
tm_shape(land, ylim = c(-88,88), relative=FALSE) +
    tm_raster("cover_cls", palette = pal8, title="Global Land Cover", legend.hist=TRUE,
legend.hist.z=0) +
tm_shape(World) +
    tm_borders() +
tm_format_World(inner.margins=0) +
tm_legend(text.size=1,
    title.size=1.2,
    position = c("left","bottom"),
    bg.color = "white",
    bg.alpha=.2,
    frame="gray50",
    height=.6,
    hist.width=.2,
    hist.height=.2,
```

```
    hist.bg.color="gray60",
    hist.bg.alpha=.5)
```

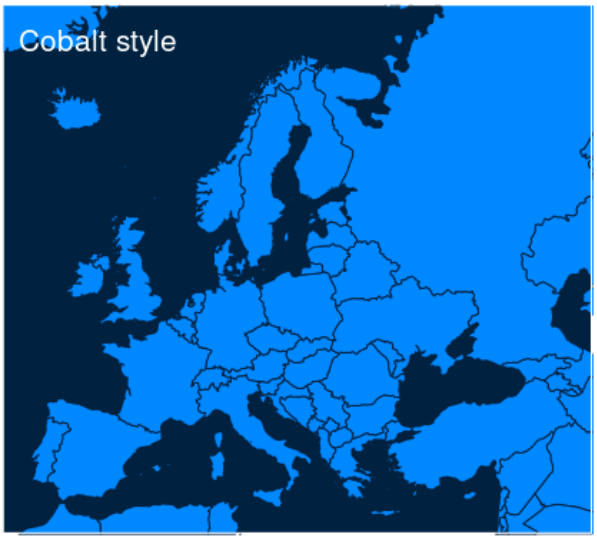


The wrapper functions starting with `tm_format_` specify the format for a specific shape. In the tmap package, a couple of them are included, for instance `tm_format_World` that is tailored for world maps. It's also possible to create your own wrapper function for shapes that you will use frequently.

Besides the shape-dependent `tm_format_` wrapper functions, tmap also contains wrapper functions for shape-independent styles.

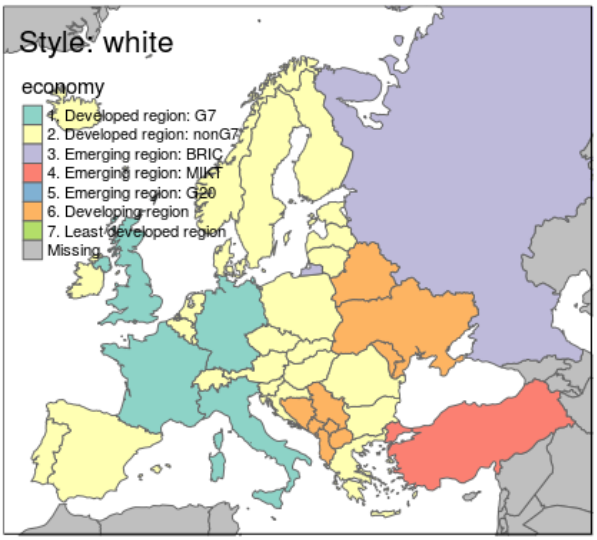
```
qtm(Europe, style="natural", title="Natural style") # equivalent to: qtm(Europe) +
tm_style_natural(title="Natural style")
```


```
qtm(Europe, style="cobalt", title="Cobalt style") # equivalent to: qtm(Europe) +
tm_style_cobalt(title="Cobalt style")
```



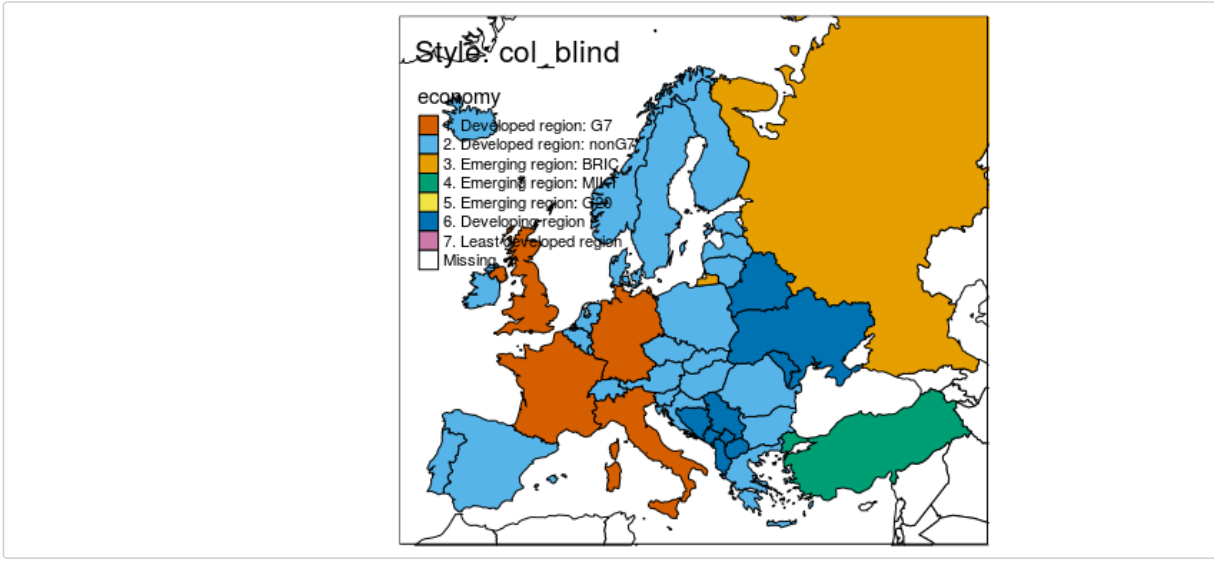
Run `style_catalogue()` to create an extensive catalogue of the available styles. The default style is `tm_style_white`. This default can be changed with the global option called `tmap.style`, which can be get and set with `tmap_style`:

```
# make a categorical map
qtm(Europe, fill="economy", title=paste("Style:", tmap_style()))
## current tmap style is "white"
```




```
# change to color-blind-friendly style
current_style <- tmap_style("col_blind")
## tmap style set to "col_blind"

# make a categorical map
qtm(Europe, fill="economy", title=paste("Style:", tmap_style()))
## current tmap style is "col_blind"
```



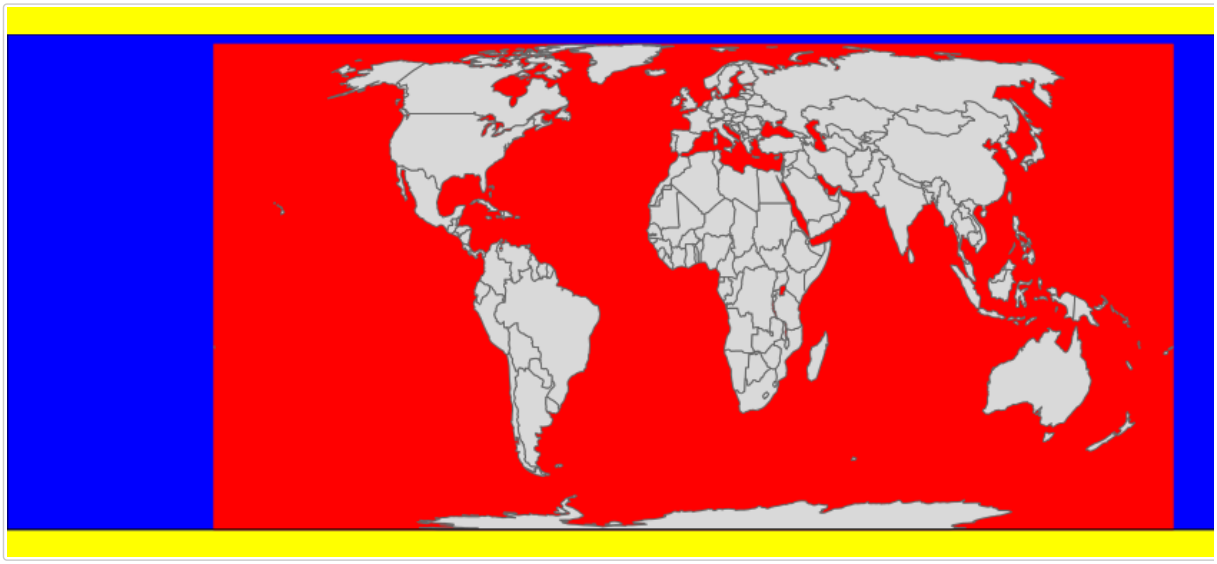

```
# change back
tmap_style(current_style)
## tmap style set to "white"
```

Also, the outer and inner margins as well as the aspect ratio are determined with `tm_layout`:

```
(tm <- qtm(World) +
tm_layout(outer.margins=c(.05,0,.05,0),
    inner.margins=c(0,0,.02,0), asp=0))
```

The behaviour of `outer.margins`, `inner.margins`, and `asp` are correlated. To see the rectangles that these arguments determine, the design mode can be enabled:

```
tm + tm_layout(design.mode=TRUE)
## ----------------aspect ratios-----------------------
## | specified (asp argument of tm_layout)   0.000000 |
## | device (yellow)                         2.500000 |
## | frame (blue)                            2.777778 |
## | master shape, World, (red)              1.979637 |
## ----------------------------------------------------
```



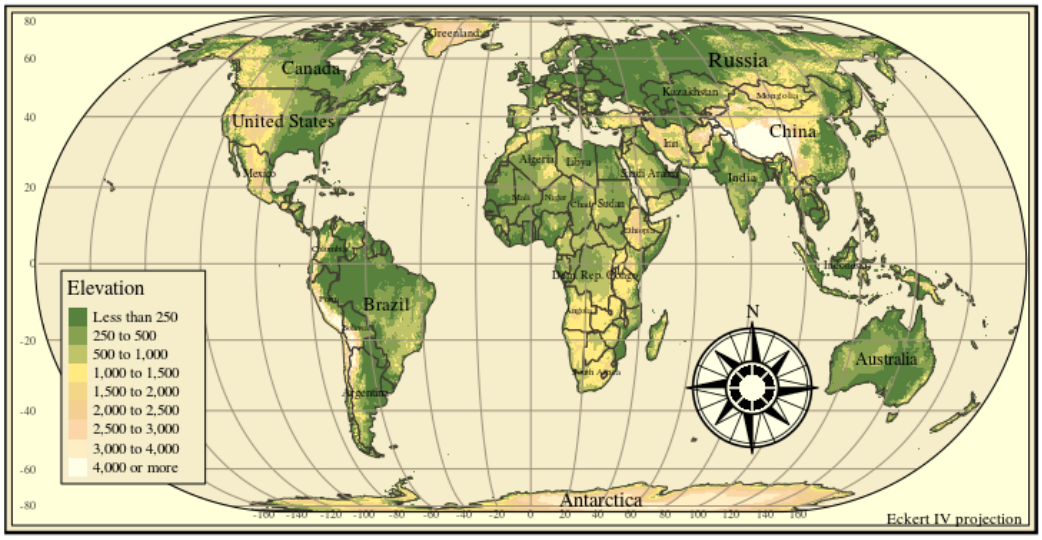
The red rectangle is the bounding box of the shape object. Both `inner.margins` and `asp` determine the measurements of the frame, indicated by the blue rectangle. Setting the left inner margin is useful to have extra space for the legend.

Setting the aspect ratio is handy when the plot is saved to an image with a specific resolution. For instance, to save a thematic World map as a png image of 1920 by 1080 pixels, the setting `outer.margins=0, asp=1920/1080` can be used. When `asp=0`, as in the example above, the aspect ratio of the device (given the outer margins) is taken. See `save_tmap`, which uses these tricks under the hood.

Map attributes

The following demo shows how a world map can be enhanced with map attributes such as grid lines and a map compass.

```
tm_shape(land, projection="eck4") +
    tm_raster("elevation", breaks=c(-Inf, 250, 500, 1000, 1500, 2000, 2500, 3000, 4000, Inf),
        palette = terrain.colors(9), title="Elevation", auto.palette.mapping=FALSE) +
tm_shape(World) +
    tm_borders("grey20") +
    tm_grid(projection="longlat", labels.size = .5) +
    tm_text("name", size="AREA") +
tm_compass(position = c(.65, .15), color.light = "grey90") +
tm_credits("Eckert IV projection", position = c(.85, 0)) +
tm_style_classic(inner.margins=c(.04,.03, .02, .01), legend.position = c("left", "bottom"),
    legend.frame = TRUE, bg.color="lightblue", legend.bg.color="lightblue",
    earth.boundary = TRUE, space.color="grey90")
```



Saving maps

A handy function for saving maps is `save_tmap`:

```
tm <- tm_shape(World) +
    tm_fill("well_being", id="name", title="Well-being") +
    tm_format_World()
```

```
save_tmap(tm, "World_map.png", width=1920, height=1080)
```

This function can also save interactive maps to stand-alone HTML files:

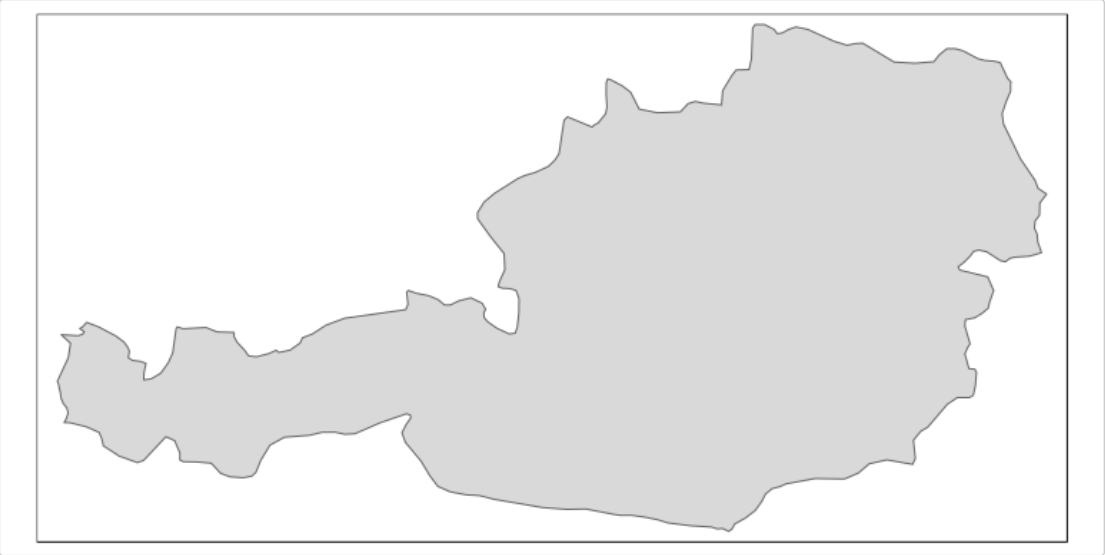
```
save_tmap(tm, "World_map.html")
```

See `vignette("tmap-modes")` for more on interactive maps.

Tips n' tricks

1. Selections can be made by treating the data.frame of the shape object:

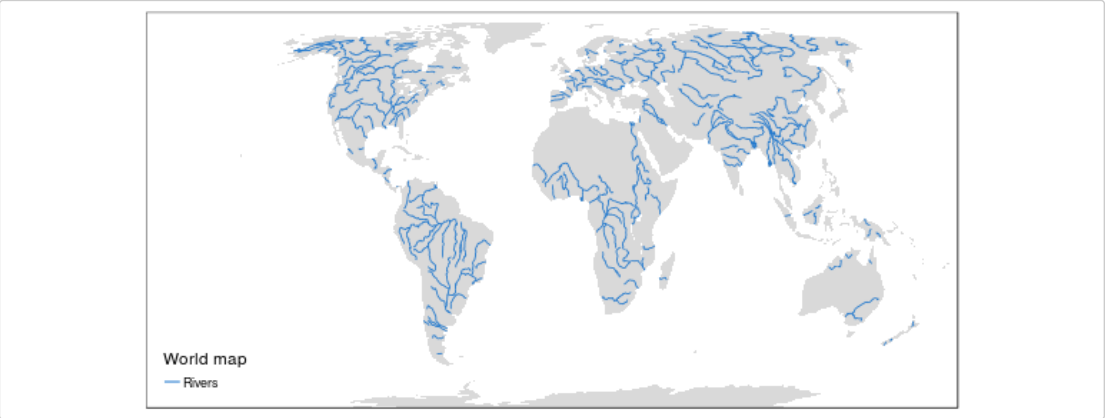
```
tm_shape(Europe[Europe$name=="Austria", ]) +
    tm_polygons()
```



2. A manual legend can be generated `tm_add_legend`:

```
data(World)

tm_shape(World) +
    tm_fill() +
tm_shape(rivers) +
    tm_lines(col="dodgerblue3") +
    tm_add_legend(type="line", col="dodgerblue3", labels = "Rivers", title="World map") +
tm_format_World()
```



3. Each drawing element has a scalar argument called `scale`. The overall scaling and font sizes can be set by the `scale` argument in `tm_layout`.
4. Arguments of the bounding box function `bb` can be passed directly to `tm_shape`:

```
tm_shape(World, bbox = "India") +
    tm_polygons("MAP_COLORS", palette="Pastel2") +
tm_shape(metro) +
    tm_bubbles("pop2010", title.size = "Population") +
```

```
    tm_text("name", size = "pop2010", legend.size.show = FALSE, root=8, size.lowerbound = .7,
auto.placement = TRUE)
```