This vignette builds on the [making maps chapter](https://geocompr.robinlovelace.net/adv-map.html) of [the Geocomputation with R book](https://geocompr.github.io/).  
Its goal is to demonstrate how to set and modify grids and graticules in the **tmap** package.

**Prerequisites**

The examples below assume the following packages are attached:

library(spData) # example datasets

library(tmap) # map creation (>=2.3)

library(sf) # spatial data classes

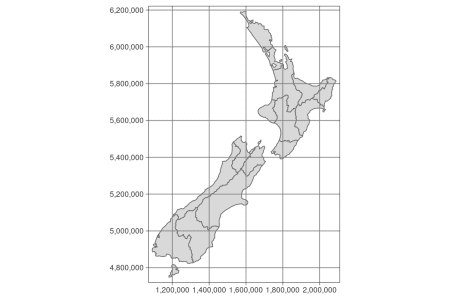
**Grids and graticules**

The **tmap** package offers two ways to draws coordinate lines – tm\_grid() and tm\_graticules().  
The role of tm\_grid() is to represent the input data’s coordinates.  
For example, the nz object uses the New Zealand Transverse Mercator projection, with meters as its units.

tm\_shape(nz) +

tm\_polygons() +

tm\_grid()

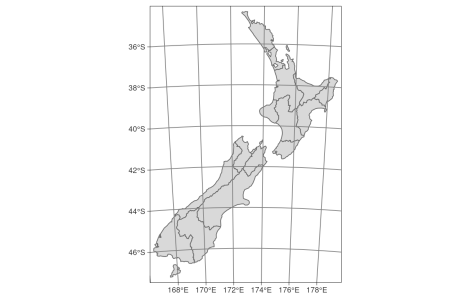


tm\_graticules() shows longitude lines (meridians) and latitude lines (parallels), with degrees as units (note the degree sign in the example below).

tm\_shape(nz) +

tm\_polygons() +

tm\_graticules()



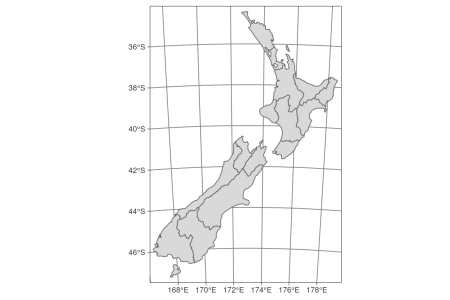
**Layers order**

Both, tm\_grid() and tm\_graticules() could be placed above or below the main spatial data.  
Its position on the map depends on its place in the code.  
When tm\_grid() or tm\_graticules() are placed after the code drawing geometry (e.g. tm\_polygons()), the grids or graticules are ploted on the top of the map.  
On the other hand, when tm\_grid() or tm\_graticules() are placed before the code drawing geometry (e.g. tm\_polygons()), the grids or graticules are plotted behind the spatial data.

tm\_shape(nz) +

tm\_graticules() +

tm\_polygons()



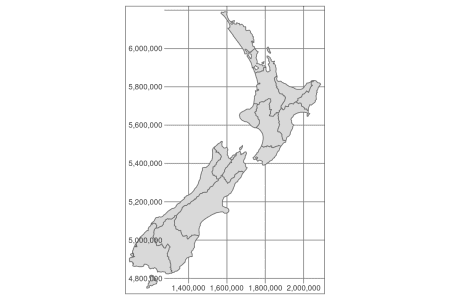
**Customization**

Grids and graticules can be easily customized in **tmap** using several arguments.  
The first one, labels.inside.frame moves the labels inside the map grid (it is set to FALSE as the default).

tm\_shape(nz) +

tm\_grid(labels.inside.frame = TRUE) +

tm\_polygons()

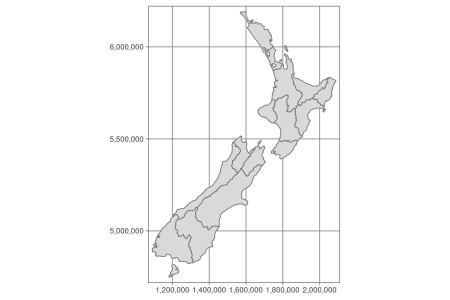


The number of horizontal (x) and vertical (y) lines can be set using the n.x and n.y arguments.  
Importantly, **tmap** rounds coordinate values to equally spaced “round” values, so the number of actual labels may be slightly different than set with n.x and n.y.

tm\_shape(nz) +

tm\_grid(n.x = 4, n.y = 3) +

tm\_polygons()

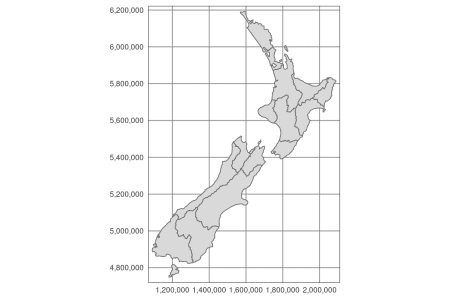


By default, tm\_grid() and tm\_graticules() shows ticks and lines.  
They can be disabled using ticks = FALSE and lines = FALSE.

tm\_shape(nz) +

tm\_grid(ticks = FALSE) +

tm\_polygons()

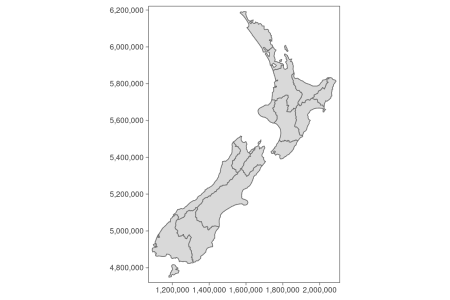


Especially, lines = FALSE could be useful when presenting raster data.

tm\_shape(nz) +

tm\_grid(lines = FALSE) +

tm\_polygons()

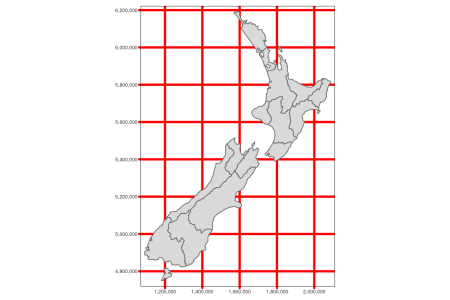


It is also possible to customize tm\_grid() and tm\_graticules() apperance, for example by chaning the lines colors (col), width (lwd) or labels size (labels.size).

tm\_shape(nz) +

tm\_grid(col = "red", lwd = 3, labels.size = 0.4) +

tm\_polygons()



The above examples uses tm\_grid(), but the same arguments apply to the tm\_graticules().

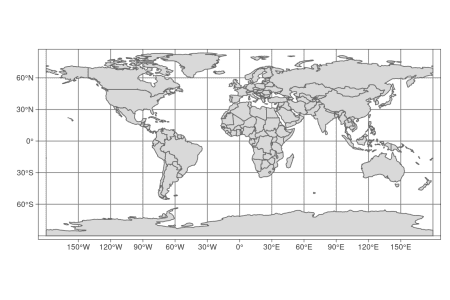
**Layout settings**

By default, **tmap** adds small inner margins between the presented data and the map frame.  
It works well in many cases, for example, see the map of New Zealand above.  
However, it does not look perfect for world maps.

tm\_shape(world) +

tm\_graticules() +

tm\_polygons()



The way to fix this is to use the tm\_layout() function and set its inner.margins argument to 0.

tm\_shape(world) +

tm\_graticules() +

tm\_polygons() +

tm\_layout(inner.margins = 0)

