



— Parte 5. —

Mappe Tematiche

Paolo Bosetti (paolo.bosetti@unitn.it)

Indice

1	Mappe tematiche: tmap	1
2	Trentino	4

1 Mappe tematiche: tmap

È possibile creare mappe in R in numerosi modi: anche R base dispone di funzioni per gestire dati geografici.

Sull'onda della *new wave* del Tidyverse, tuttavia, sono emerse alcune librerie che sposano la stessa filosofia e si integrano bene con la gestione dati basata su *pipe* e *tibble*. In particolare sono utili *sf* che consente la gestione di dati geografici di tipo *simple feature*, e *tmap*, che adotta una grammatica simile a *GGplot2* per la creazione di mappe.

```
library(tmap)
library(sf)
```

```
## Linking to GEOS 3.8.1, GDAL 3.2.1, PROJ 7.2.1
```

```
library(spData)
library(spDataLarge)
```

Come in *GGplot2*, e mappe si costruiscono per strati, sommando differenti contributi provenienti da oggetti *sf*. A loro volta, gli oggetti *sf* sono dei data frame che contengono dati eterogenei con l'aggiunta di *feature* geometriche:

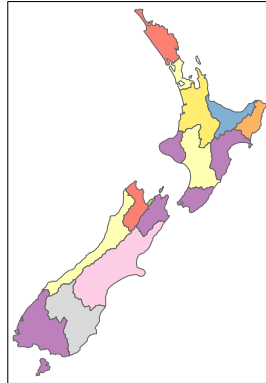
```
nz %>% str # nz è l'esempio per la Nuova Zelanda fornito da spData
```

```
## Classes 'sf' and 'data.frame':  16 obs. of  7 variables:
##  $ Name      : chr  "Northland" "Auckland" "Waikato" "Bay of Plenty" ...
##  $ Island     : chr  "North" "North" "North" "North" ...
##  $ Land_area  : num  12501 4942 23900 12071 8386 ...
##  $ Population : num  175500 1657200 460100 299900 48500 ...
##  $ Median_income: int  23400 29600 27900 26200 24400 26100 29100 25000 32700 26900 ...
##  $ Sex_ratio   : num  0.942 0.944 0.952 0.928 0.935 ...
##  $ geom        :sfc_MULTIPOLYGON of length 16; first list element: List of 1
##  ..$ :List of 1
##  .. ..$ : num [1:68, 1:2] 1745493 1740539 1733165 1720197 1709110 ...
##  ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
```

```
## - attr(*, "sf_column")= chr "geom"
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA NA NA NA NA
##   ..- attr(*, "names")= chr [1:6] "Name" "Island" "Land_area" "Population" ...
```

Plottiamo la mappa della Nuova Zelanda riempita con colori casuali generati in modo che aree adiacenti abbiano sempre colori differenti e con i bordi delle regioni interne:

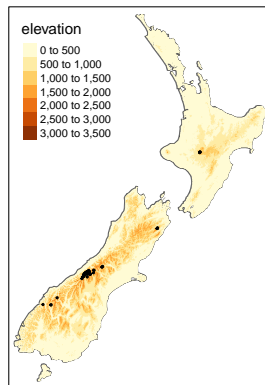
```
map_nz <- tm_shape(nz) +
  tm_fill(col="MAP_COLORS") +
  tm_borders()
print(map_nz)
```



La funzione `qtm()` semplifica le cose, e può essere combinata con altri dati `sf`; inoltre, un oggetto `sf` può contenere anche dati *raster* (altimetria, ortofoto, ecc.):

```
qtm(nz) + qtm(nz_elev) + qtm(nz_height)
```

```
## stars object downsampled to 877 by 1140 cells. See tm_shape manual (argument raster.downsample)
```



Si possono combinare livelli *raster* con livelli vettoriali:

```
map_nz1 <- map_nz +
  tm_shape(nz_elev) + tm_raster(alpha = 0.8)
print(map_nz1)
```

```
## stars object downsampled to 877 by 1140 cells. See tm_shape manual (argument raster.downsample)
```



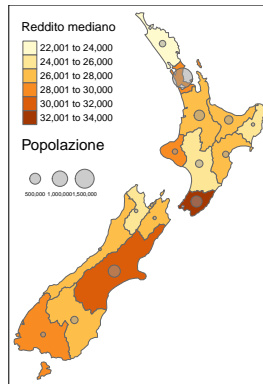
```
tmap arrange(map nz1, map nz2, map nz3)
```

The figure consists of three panels, each showing a map of New Zealand with a color-coded elevation scale. The scale ranges from 0 to 3,500 meters, with colors transitioning from light yellow (0-500m) to dark brown (3,000-3,500m). The left panel shows the raw elevation data. The middle panel shows a smoothed version of the data. The right panel shows a smoothed version of the data with black dots indicating specific locations.

È interessante aggiungere ad una mappa i metadati contenuti nell'oggetto **sf**:

```
tm_shape(nz) +
  tm_borders() +
  tm_fill(col="Median_income", title="Reddito mediano") +
  tm_bubbles(size="Population", alpha=1/2, title.size="Popolazione")
```

```
## Legend labels were too wide. The labels have been resized to 0.6, 0.6, 0.6, 0.6, 0.6, 0.6. Increase
## Legend labels were too wide. Therefore, legend.text.size has been set to 0.31. Increase legend.width
## The legend is too narrow to place all symbol sizes.
```



2 Trentino

Vediamo di realizzare alcune mappe tematiche per il Trentino, attingendo a dati GIS open in formato *shapefile* (.shp), che può essere direttamente caricato come oggetto *simple feature*, *sf*:

- Limiti amministrativi scaricati da (WebGIS)[<https://webgis.provincia.tn.it/wgt/>], in formato ESRI (.shp). Contengono sia i limiti comunali che delle comunità di valle;
- Dati COVID-19 scaricati da (FBK)[https://covid19trentino.fbk.eu/data/stato_comuni_td.csv];
- Dati estratti dal censimento 2011 per la **regione Trentino/Alto Adige**, formato tabulare senza dati geografici.

```
# Dati amministrativi
comuni <- read_sf(dsn="ammcom/ammcom.shp")
com.valle <- read_sf(dsn="ammcva/ammcva.shp")
# Da "TIONE DI TRENTO" a "Tione Di Trento":
com.valle$nome <- com.valle$nome %>% str_to_title()
# Dati COVID-19:
covid <- read_csv("https://covid19trentino.fbk.eu/data/stato_comuni_td.csv")
```

```
## Rows: 166 Columns: 9
```

```
## -- Column specification -----
```

```
## Delimiter: ","
```

```
## chr (2): nome, aggiornamento
```

```
## dbl (7): codice, contagi, guariti, decessi, dimessi, lat, lon
```

```
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
# Censimento:
```

```
stat <- read_csv2(mydata("censimento_TAA_2011.csv"))
```

```
## i Using '','' as decimal and '','' as grouping mark. Use `read_delim()` for more control.
## Rows: 2055 Columns: 154

## -- Column specification -----
## Delimiter: ";"
## chr (5): REGIONE, PROVINCIA, COMUNE, LOCALITA, ALTITUDINE
## dbl (149): CODREG, CODPRO, CODCOM, PROCOM, LOC2011, CODLOC, TIPOLOC, AMPLOC,...

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
stat %<>% filter(PROVINCIA=="Trento")
```

Ispezioniamo gli oggetti con `str()`:

```
str(comuni)
```

```
## sf [166 x 19] (S3: sf/tbl_df/tbl/data.frame)
## $ objectid : num [1:166] 2209 2210 2211 2212 2213 ...
## $ classid : chr [1:166] "AMB003_1" "AMB003_26" "AMB003_60" "AMB003_237" ...
## $ codice : num [1:166] 1 26 60 237 203 51 136 155 47 188 ...
## $ istat : chr [1:166] "22001" "22026" "22060" "22237" ...
## $ istatcat : chr [1:166] "A116" "B158" "C727" "M351" ...
## $ nome : chr [1:166] "ALA" "BRESIMO" "CIS" "AMBLAR-DON" ...
## $ supcom : num [1:166] 120 41 5.5 19.9 50.2 ...
## $ altcom : num [1:166] 180 1036 732 971 525 ...
## $ fkcomcva : chr [1:166] "AMB002_10" "AMB002_6" "AMB002_6" "AMB002_6" ...
## $ fkamprv : chr [1:166] "AMB001_1" "AMB001_1" "AMB001_1" "AMB001_1" ...
## $ struttura : chr [1:166] "S133" "S133" "S133" "S133" ...
## $ struttura_ : chr [1:166] "Servizio Catasto" "Servizio Catasto" "Servizio Catasto" "Servizio Catasto" ...
## $ fkfonte : chr [1:166] "05" "05" "05" "05" ...
## $ fktfonte_d : chr [1:166] "altre fonti" "altre fonti" "altre fonti" "altre fonti" ...
## $ fktipoelab : chr [1:166] "01" "01" "01" "01" ...
## $ fktipoel_d : chr [1:166] "manuale" "manuale" "manuale" "manuale" ...
## $ fkscala : chr [1:166] "03" "03" "03" "03" ...
## $ fkscala_d : chr [1:166] "10000" "10000" "10000" "10000" ...
## $ geometry :sfc_MULTIPOLYGON of length 166; first list element: List of 1
## ..$ :List of 1
## ...$ : num [1:1420, 1:2] 663379 663430 663474 663535 663602 ...
## ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
## - attr(*, "sf_column")= chr "geometry"
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA NA NA NA NA NA NA NA ...
## ..- attr(*, "names")= chr [1:18] "objectid" "classid" "codice" "istat" ...
```

```
str(covid)
```

```
## spec_tbl_df [166 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
## $ codice : num [1:166] 1 26 60 237 203 51 136 155 47 188 ...
## $ nome : chr [1:166] "ALA" "BRESIMO" "CIS" "AMBLAR-DON" ...
## $ contagi : num [1:166] 920 20 16 42 121 82 144 113 344 287 ...
## $ guariti : num [1:166] 781 20 16 39 110 76 120 106 328 253 ...
## $ decessi : num [1:166] 8 0 0 1 3 3 5 3 8 2 ...
## $ dimessi : num [1:166] 2 0 0 0 0 0 0 0 1 ...
## $ lat : num [1:166] 45.7 46.4 46.4 46.4 45.8 ...
## $ lon : num [1:166] 11 10.9 11 11.2 11.1 ...
## $ aggiornamento: chr [1:166] "28/12/2021" "28/12/2021" "28/12/2021" "28/12/2021" ...
```

```
## - attr(*, "spec")=
## .. cols(
## ..   codice = col_double(),
## ..   nome = col_character(),
## ..   contagi = col_double(),
## ..   guariti = col_double(),
## ..   decessi = col_double(),
## ..   dimessi = col_double(),
## ..   lat = col_double(),
## ..   lon = col_double(),
## ..   aggiornamento = col_character()
## .. )
## - attr(*, "problems")=<externalptr>
```

Dunque `comuni` è un oggetto `sf`, la cui classe è `sf`, `tbl_df`, `tbl`, `data.frame`, mentre `covid` è ovviamente un `tibble`. Siccome `comuni` eredita da `data.frame`, possiamo aggiungere e modificare le sue colonne mediante l'operatore `$`, oppure possiamo usare la più avanzata libreria `dyplr` per unire le due tabelle condividendo le stesse colonne `nome` e `codice`:

```
tn_data <- comuni %>% full_join(covid, by=c("codice", "nome"))
str(tn_data)
```

```
## sf [166 x 26] (S3: sf/tbl_df/tbl/data.frame)
## $ objectid      : num [1:166] 2209 2210 2211 2212 2213 ...
## $ classid       : chr [1:166] "AMB003_1" "AMB003_26" "AMB003_60" "AMB003_237" ...
## $ codice        : num [1:166] 1 26 60 237 203 51 136 155 47 188 ...
## $ istat         : chr [1:166] "22001" "22026" "22060" "22237" ...
## $ istatcat      : chr [1:166] "A116" "B158" "C727" "M351" ...
## $ nome          : chr [1:166] "ALA" "BRESIMO" "CIS" "AMBLAR-DON" ...
## $ supcom        : num [1:166] 120 41 5.5 19.9 50.2 ...
## $ altcom        : num [1:166] 180 1036 732 971 525 ...
## $ fkcomcva      : chr [1:166] "AMB002_10" "AMB002_6" "AMB002_6" "AMB002_6" ...
## $ fkammpv       : chr [1:166] "AMB001_1" "AMB001_1" "AMB001_1" "AMB001_1" ...
## $ struttura     : chr [1:166] "S133" "S133" "S133" "S133" ...
## $ struttura_    : chr [1:166] "Servizio Catasto" "Servizio Catasto" "Servizio Catasto" "Servizio Cat
## $ fkfonte       : chr [1:166] "05" "05" "05" "05" ...
## $ fktfonte_d    : chr [1:166] "altre fonti" "altre fonti" "altre fonti" "altre fonti" ...
## $ fktipoelab    : chr [1:166] "01" "01" "01" "01" ...
## $ fktipoel_d    : chr [1:166] "manuale" "manuale" "manuale" "manuale" ...
## $ fkscala       : chr [1:166] "03" "03" "03" "03" ...
## $ fkscala_d     : chr [1:166] "10000" "10000" "10000" "10000" ...
## $ geometry      :sfc_MULTIPOLYGON of length 166; first list element: List of 1
## ..$ :List of 1
## .. ..$ : num [1:1420, 1:2] 663379 663430 663474 663535 663602 ...
## ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
## $ contagi       : num [1:166] 920 20 16 42 121 82 144 113 344 287 ...
## $ guariti       : num [1:166] 781 20 16 39 110 76 120 106 328 253 ...
## $ decessi       : num [1:166] 8 0 0 1 3 3 5 3 8 2 ...
## $ dimessi       : num [1:166] 2 0 0 0 0 0 0 0 0 1 ...
## $ lat           : num [1:166] 45.7 46.4 46.4 46.4 45.8 ...
## $ lon           : num [1:166] 11 10.9 11 11.2 11.1 ...
## $ aggiornamento: chr [1:166] "28/12/2021" "28/12/2021" "28/12/2021" "28/12/2021" ...
## - attr(*, "sf_column")= chr "geometry"
## - attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",...: NA NA NA NA NA NA NA NA NA ...
## ..- attr(*, "names")= chr [1:25] "objectid" "classid" "codice" "istat" ...
```

Come si vede, ora `tn_data` contiene anche i dati COVID-19.

Ripetiamo lo stesso con `stat` per recuperare la popolazione. In questo caso, i dati originari contengono più di un *caso* per lo stesso comune (uno per frazione). È quindi necessario sommare tutte le frazioni (campo `LOCALITA`) di uno stesso comune (campo `PROCOM`). Successivamente importiamo in `comuni` tutti i dati in cui `PROCOM` equivale a `istat`:

```
pop_data <- stat %>% group_by(PROCOM, COMUNE) %>% summarise(Popolazione=sum(P1))

## `summarise()` has grouped output by 'PROCOM'. You can override using the `.groups` argument.

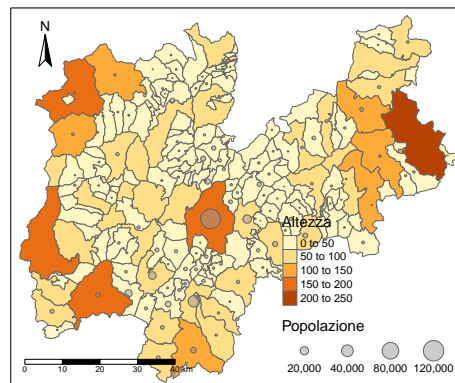
pop_data <- pop_data %>% mutate(PROCOM=as.character(PROCOM))
tn_data <- tn_data %>% left_join(pop_data, by=c("istat"="PROCOM"))
tn_data %>% filter(is.na(Popolazione)) %>% length

## [1] 28
```

Come si vede, ci sono 28 comuni il cui codice non trova corrispondenza nel censimento del 2011, per i quali quindi la popolazione non è nota. Non ci sono metodi semplici e automatici per ricostruire questi dati mancanti.

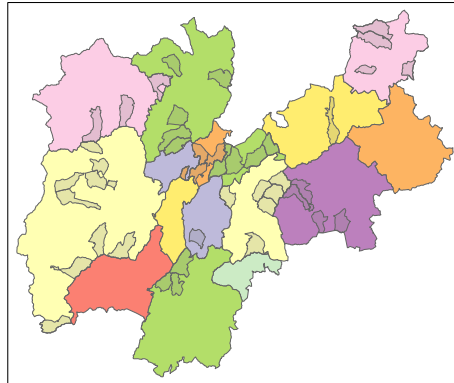
Ora inseriamo questi dati su una mappa tematica, indicando l'altezza sul mare come colore e la popolazione con un diagramma a bolle:

```
tm_shape(tn_data) +
  tm_borders() +
  tm_fill(col="supcom", title="Altezza") +
  tm_bubbles(size="Popolazione", title.size="Popolazione", alpha=1/2) +
  tm_compass(type = "arrow", position = c("left", "top")) +
  tm_scale_bar(position=c("left", "bottom"), bg.alpha=0.5)
```



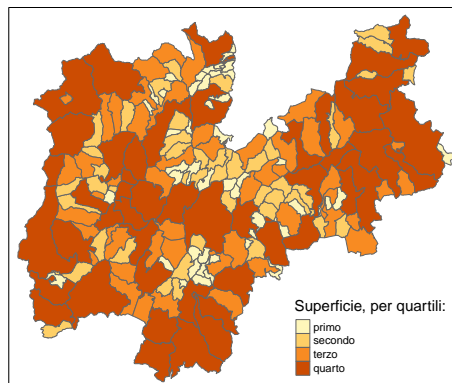
Vogliamo ora evidenziare i comuni che stanno nel primo quartile secondo la distribuzione della superficie comunale:

```
inrange <- function(x, rng) x >= min(rng) & x <= max(rng)
breaks <- quantile(tn_data$supcom)
tm_shape(com.valle) +
  tm_fill(col="MAP_COLORS") +
  tm_borders() +
  tm_shape(filter(tn_data, inrange(supcom, breaks[2:3]))) +
  tm_borders() +
  tm_fill(col=gray(0.5), alpha=0.2)
```



Possiamo anche colorare l'intera mappa secondo il quartile di appartenenza:

```
tm_shape(tn_data) +
  tm_borders() +
  tm_fill(col="supcom",
    style = "fixed",
    breaks=breaks,
    title = "Superficie, per quartili:",
    labels = c("primo", "secondo", "terzo", "quarto"))
```



Il layer `tm_basemap()` consente di aggiungere una mappa base interattiva, scaricata dinamicamente da un provider di servizi, come Open Street Map. Perché questa funzionalità sia attiva, è necessario passare dalla modalità "print" (default) alla modalità "view": l'oggetto grafico risultante è generato mediante JavaScript, è interattivo ma non è adatto alla stampa.

```
tmap_mode("view") # Per usare tm_basemap
```

```
## tmap mode set to interactive viewing
tnplus <- st_union(tn_data) %>% st_buffer(dist=2000) %>% st_cast(to="POLYGON")
tn <- tm_shape(tn_data) +
  tm_borders(col="gray") +
  tm_fill(col="contagi", title="Contagi COVID-19") +
  tm_layout(legend.outside = TRUE, legend.outside.position = "right")
tm_shape(tnplus) + tm_polygons(col="gray") +
  tn +
  tm_shape(com.valle) +
  tm_borders(col="black") +
  tm_text("nome", size=2/3) +
  tm_basemap(server="OpenStreetMap")
```