intro to mapping in R, University of Venda

andy south

2016-11-03

This tutorial will be about you doing stuff. I will ask you to try things probably before you understand fully what they are. No need to worry.

1. It doesn't matter if you make mistakes.
2. We will come back to some of the concepts later.
3. Questions are encouraged.
4. This should at least give you a start to follow up on later.

The beauty (& sometimes otherwise) of R is that there are usually multiple ways of doing the same thing. Here I will introduce you to some, there are others.

I suggest you copy and paste code chunks from this document into the R console and then modify them.

1 the tmap package

tmap is one package for making maps in R.

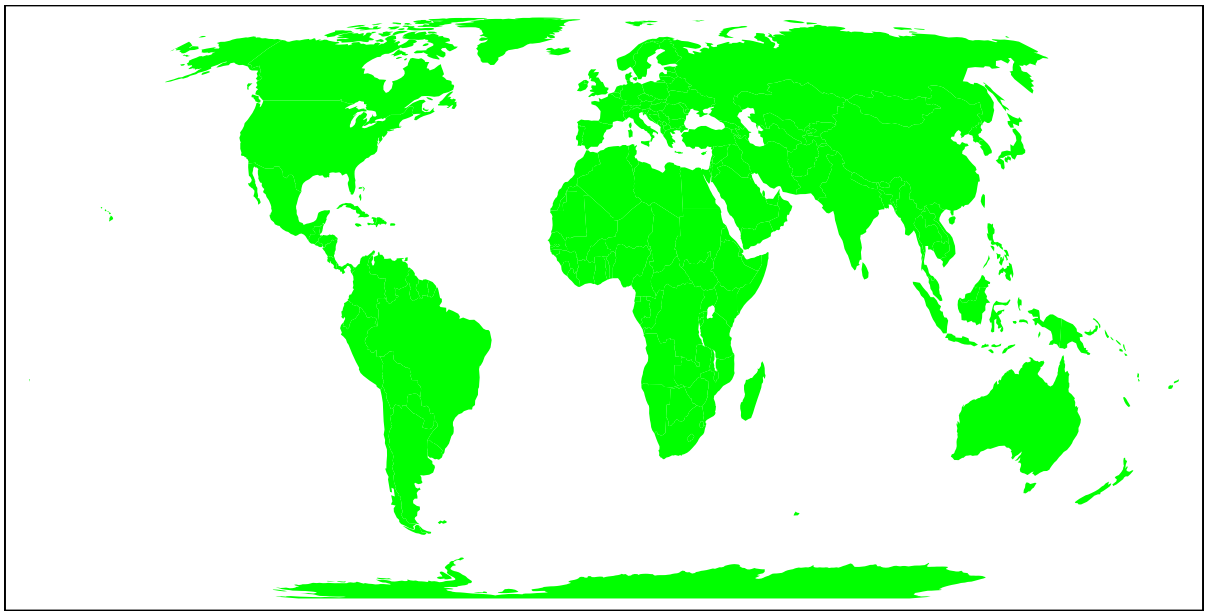
Install and load the tmap package.

```
#install if not installed already
if (require(tmap)) install.packages("tmap")
```

```
library("tmap")
```

1.1 Getting started with polygon maps

```
# load data from tmap
data(World)
# set shapes and fill them
tm_shape(World) +
  tm_fill("green")
```



```
# check what data are associated with a map using 'str(*@data)', str stands for structure
str(World@data)
```

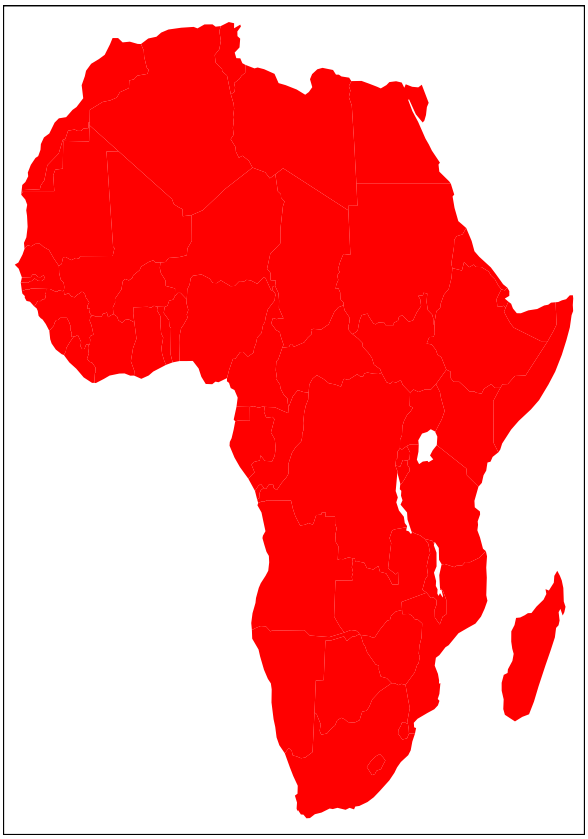
```
## 'data.frame':    177 obs. of  15 variables:
##  $ iso_a3      : Factor w/ 235 levels "ABW","AFG","AGO",..: 2 3 6 8 9 10 12 13 15 16 ...
##  $ name        : Factor w/ 241 levels "Afghanistan",..: 1 7 3 228 11 12 9 76 15 16 ...
##  $ sovereignt  : Factor w/ 200 levels "Afghanistan",..: 1 5 2 187 8 9 6 60 10 11 ...
##  $ continent   : Factor w/ 8 levels "Africa","Antarctica",..: 3 1 4 3 8 3 2 7 6 4 ...
##  $ subregion   : Factor w/ 24 levels "Antarctica","Australia and New Zealand",..: 20 11 21 23 18 23 1
##  $ area        : num  652860 1246700 27400 83600 2736690 ...
##  $ pop_est     : num  28400000 12799293 3639453 4798491 40913584 ...
##  $ pop_est_dens: num  43.5 10.3 132.8 57.4 15 ...
##  $ gdp_md_est  : num  22270 110300 21810 184300 573900 ...
##  $ gdp_cap_est : num  784 8618 5993 38408 14027 ...
##  $ economy     : Factor w/ 7 levels "1. Developed region: G7",..: 7 7 6 6 5 6 6 6 2 2 ...
##  $ income_grp  : Ord.factor w/ 5 levels "1. High income: OECD"<..: 5 3 4 2 3 4 2 2 1 1 ...
##  $ life_exp    : num  48.7 51.1 76.9 76.5 75.9 74.2 NA NA 81.9 80.9 ...
##  $ well_being  : num  4.76 4.21 5.27 7.2 6.44 ...
##  $ HPI         : num  36.8 33.2 54.1 31.8 54.1 ...
```

```
# check the contents of Factor variables using 'levels'
levels(World$continent)
```

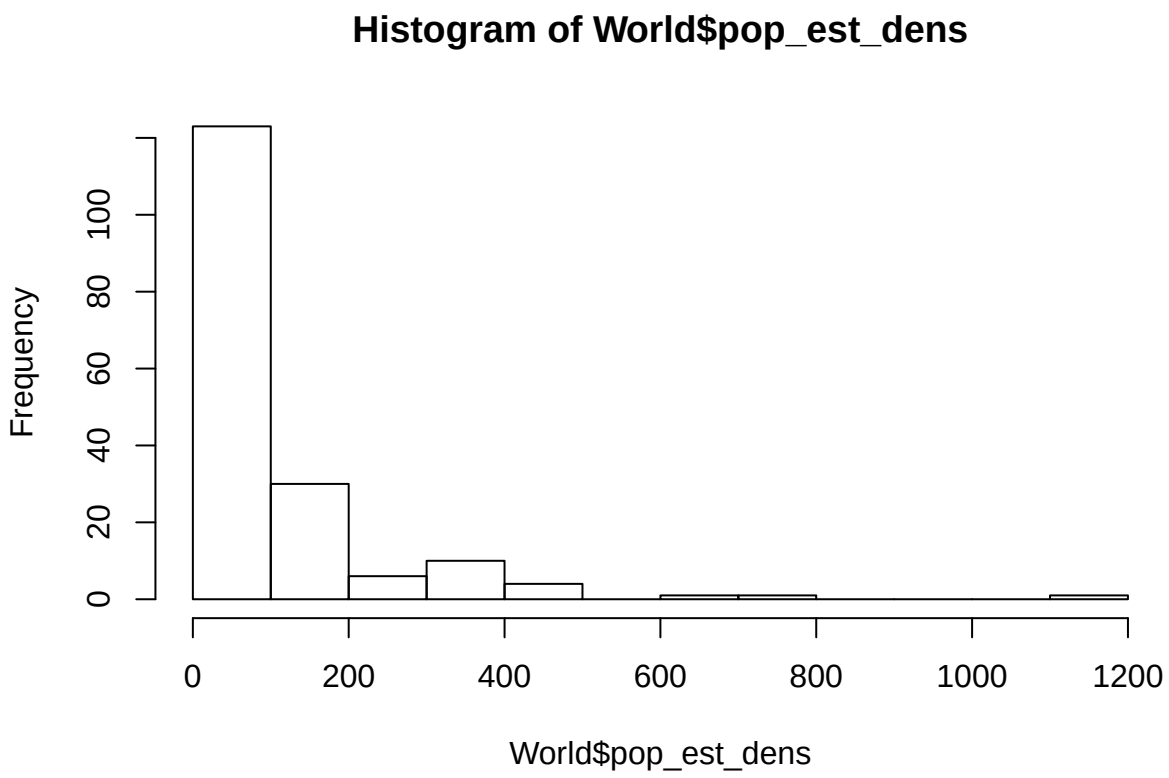
```
## [1] "Africa"                  "Antarctica"
## [3] "Asia"                    "Europe"
```

```
## [5] "North America"           "Oceania"
## [7] "Seven seas (open ocean)" "South America"
```

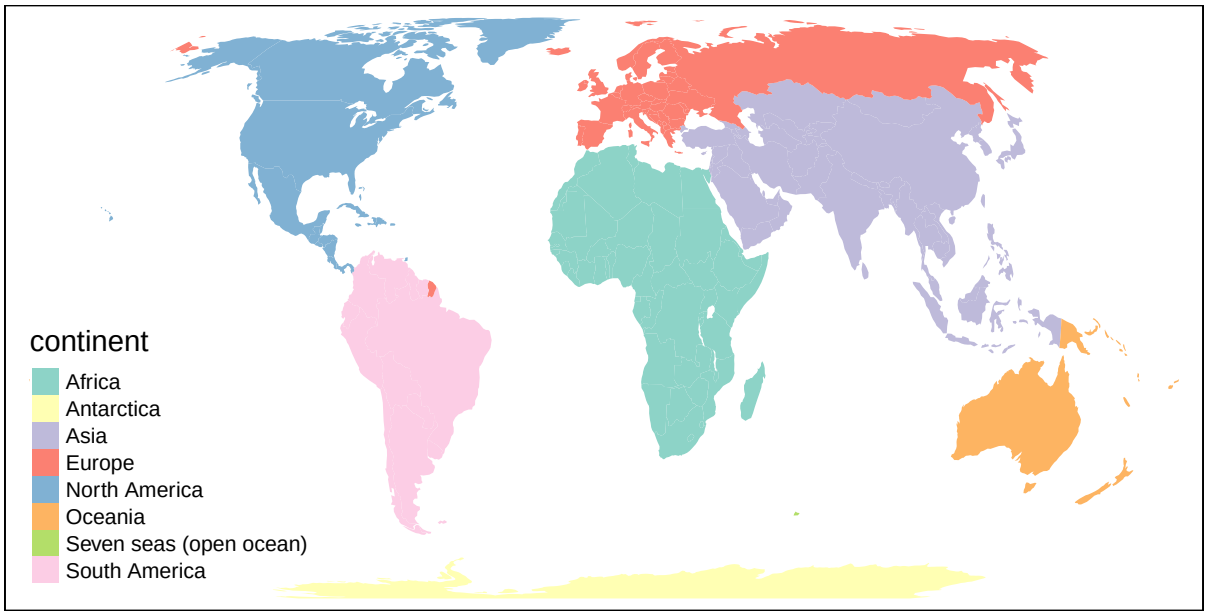
```r
# plot a subset of the map using '[which(*),]'
tm_shape(World[which(World$continent=='Africa'),]) +
  tm_fill("red")
```



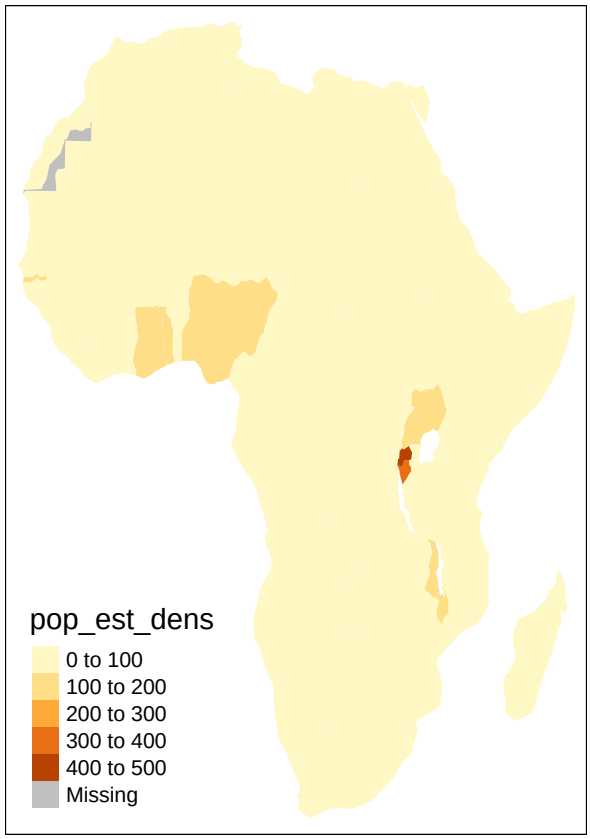
```r
# check the contents of numeric variables using 'hist'
hist(World$pop_est_dens)
```



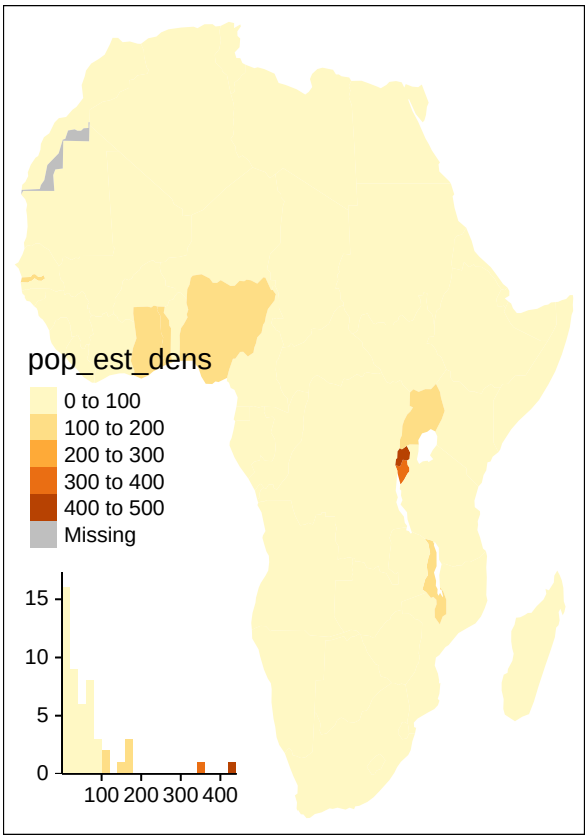

```
# fill polygons on a map based on data variables
# factor
tm_shape(World) +
  tm_fill("continent")
```



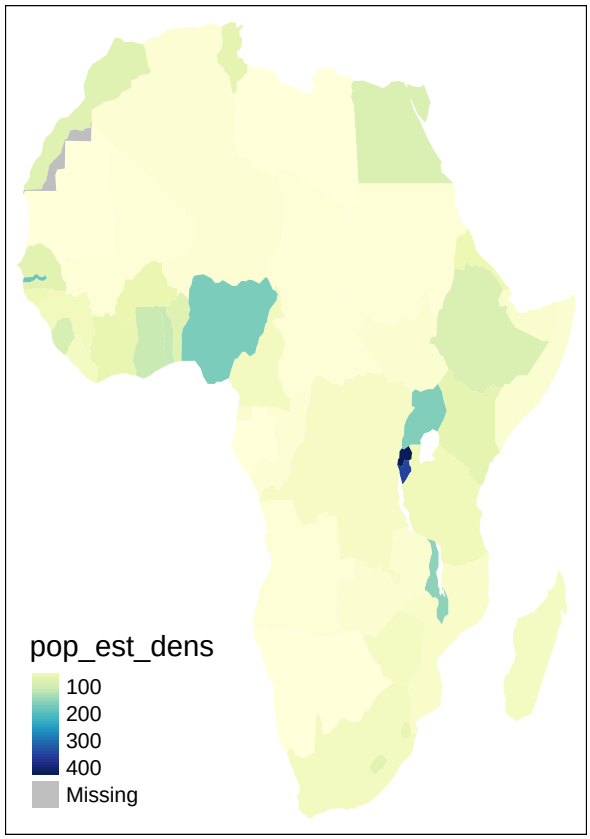

```
# numeric
tm_shape(World[which(World$continent=='Africa'),]) +
  tm_fill("pop_est_dens")
```



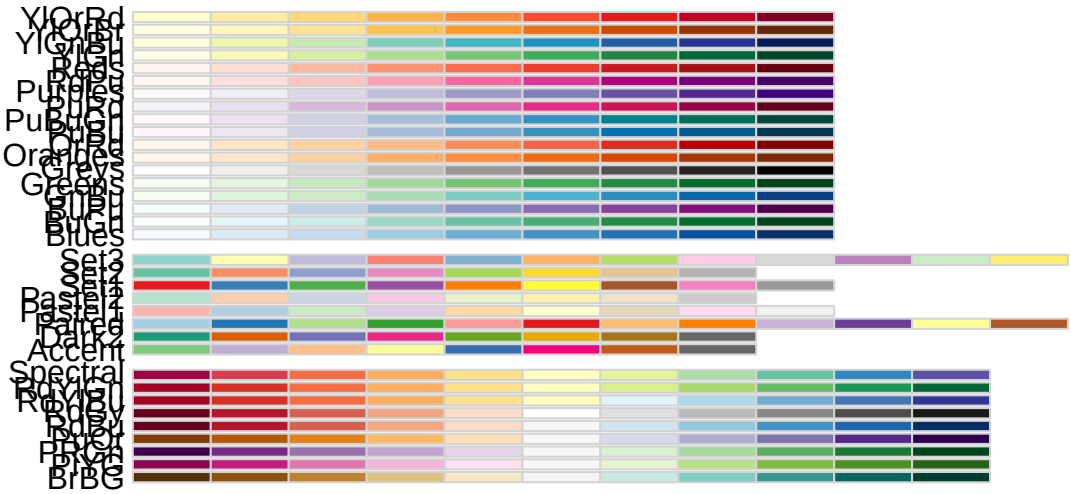

```
# to add a histogram to the legend
tm_shape(World[which(World$continent=='Africa'),]) +
  tm_fill("pop_est_dens", legend.hist = TRUE)
```




```
# the previous commands used default colours, you can modify these with 'palette'
# and make continuous with style='cont'
tm_shape(World[which(World$continent=='Africa'),]) +
  tm_fill("pop_est_dens", palette = "YlGnBu", style='cont')
```




```
# to see available palettes
RColorBrewer::display.brewer.all()
```



```
# maps are stored as a SpatialPolygonsDataFrame class from the package sp
# you can find this out by typing
class(World)
```

```
## [1] "SpatialPolygonsDataFrame"
## attr(,"package")
## [1] "sp"
```

1.2 Exercise : can you plot different areas, different variables, different colours ...

```
# to see help use '?'
?tm_fill

# syntax is like ggplot2 (e.g. + to add layers)

# start with one of the code examples above and change something
# e.g. change tm_fill("pop_est_dens") to tm_fill("area")
# or one of the other fields shown in str(World@data)
```

2.1 Getting started with point maps

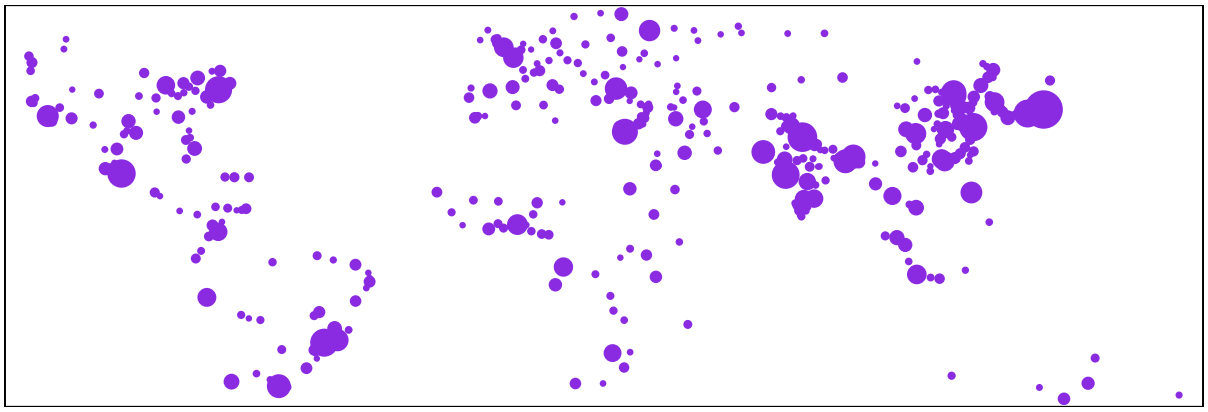
```r
# load some tmap point data for metropolitan areas
data(metro)
# ?metro gives information about the data
# this is also an sp object with associated variables
class(metro)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

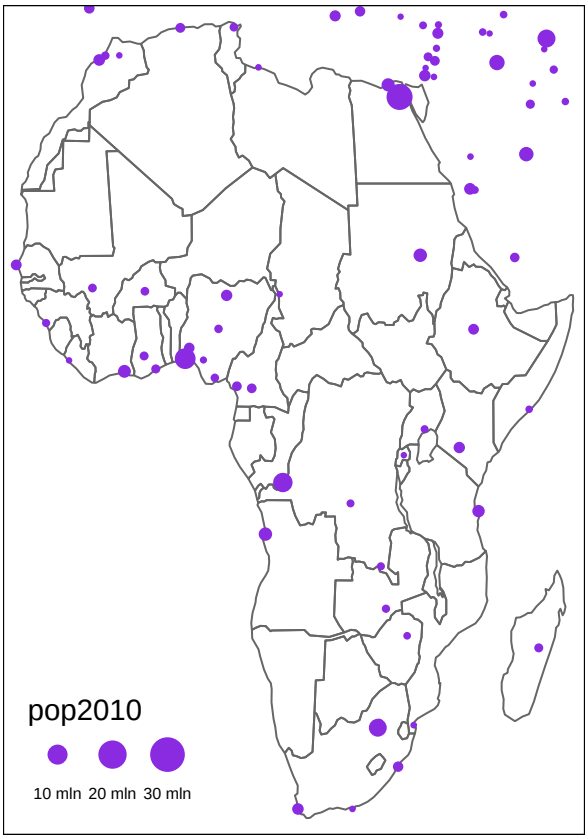
```r
str(metro@data)
```

```
## 'data.frame':    436 obs. of  12 variables:
##  $ name     : chr  "Kabul" "Algiers" "Luanda" "Buenos Aires" ...
##  $ name_long: chr  "Kabul" "El Djazair  (Algiers)" "Luanda" "Buenos Aires" ...
##  $ iso_a3   : chr  "AFG" "DZA" "AGO" "ARG" ...
##  $ pop1950  : num  170784 516450 138413 5097612 429249 ...
##  $ pop1960  : num  285352 871636 219427 6597634 605309 ...
##  $ pop1970  : num  471891 1281127 459225 8104621 809794 ...
##  $ pop1980  : num  977824 1621442 771349 9422362 1009521 ...
##  $ pop1990  : num  1549320 1797068 1390240 10513284 1200168 ...
##  $ pop2000  : num  2401109 2140577 2591388 12406780 1347561 ...
##  $ pop2010  : num  3722320 2432023 4508434 14245871 1459268 ...
##  $ pop2020  : num  5721697 2835218 6836849 15894307 1562509 ...
##  $ pop2030  : num  8279607 3404575 10428756 16956491 1718192 ...
```

```r
# plot points as bubbles
tm_shape(metro) +
    tm_bubbles("pop2010", legend.size.show = FALSE)
```



```
# plot points on top of a map
tm_shape(World[which(World$continent=='Africa'),]) +
  tm_borders() +
tm_shape(metro) +
  tm_bubbles("pop2010")
```



2.2 Exercise - can you change one of the point maps ?

...

3.1 Getting started with raster maps

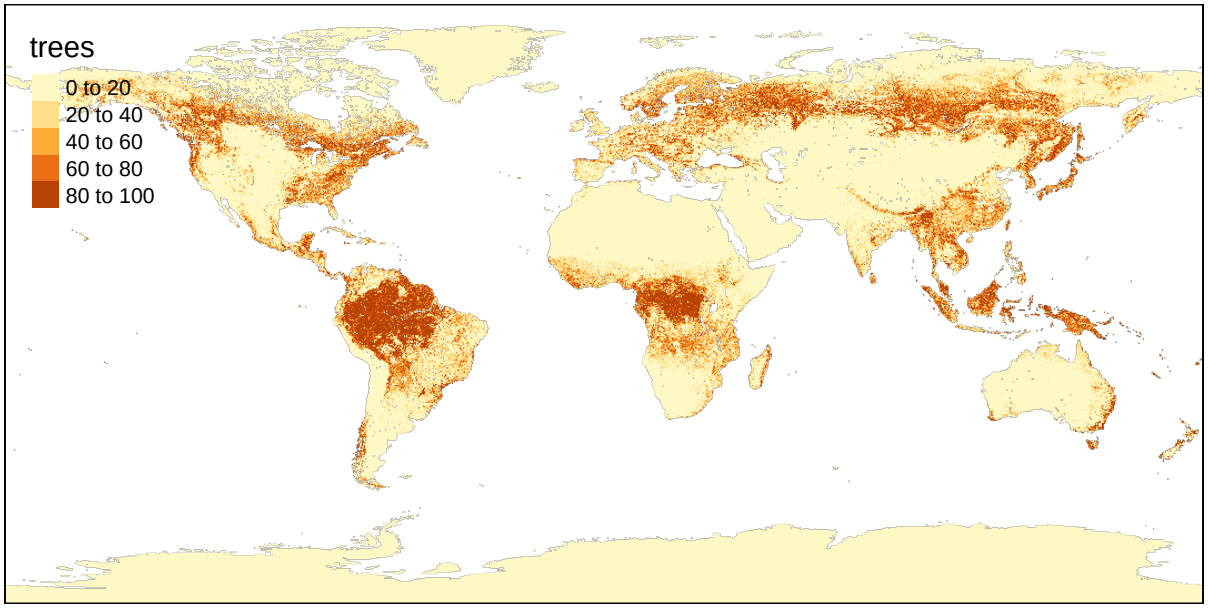
```
# load some data from tmap
data(land)

class(land)
```

```
## [1] "SpatialGridDataFrame"
## attr(,"package")
## [1] "sp"
```

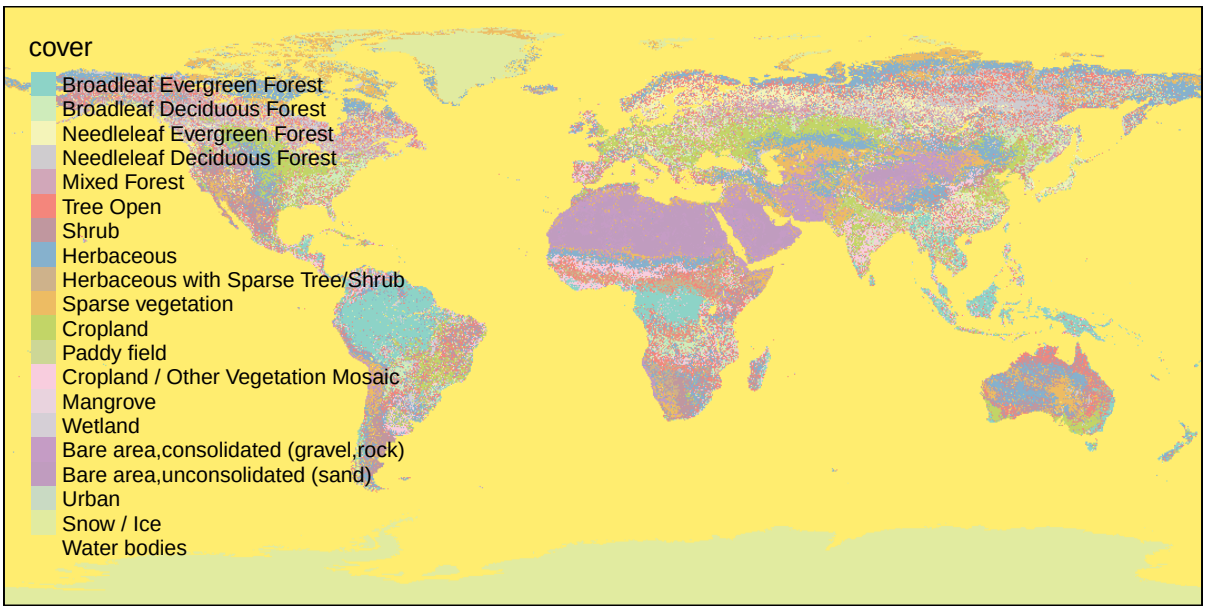
```
str(land@data)
```

```
## 'data.frame':    583200 obs. of  4 variables:
##  $ cover    : Factor w/ 20 levels "Broadleaf Evergreen Forest",..: 20 20 20 20 20 20 20 20 20 20 ...
##  $ cover_cls: Factor w/ 8 levels "Forest","Other natural vegetation",..: 8 8 8 8 8 8 8 8 8 8 ...
##  $ trees    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ elevation: int  NA NA NA NA NA NA NA NA NA NA ...
```

```
# plot land and add raster of trees
tm_shape(land) +
    tm_raster("trees", breaks=seq(0, 100, by=20) )
```

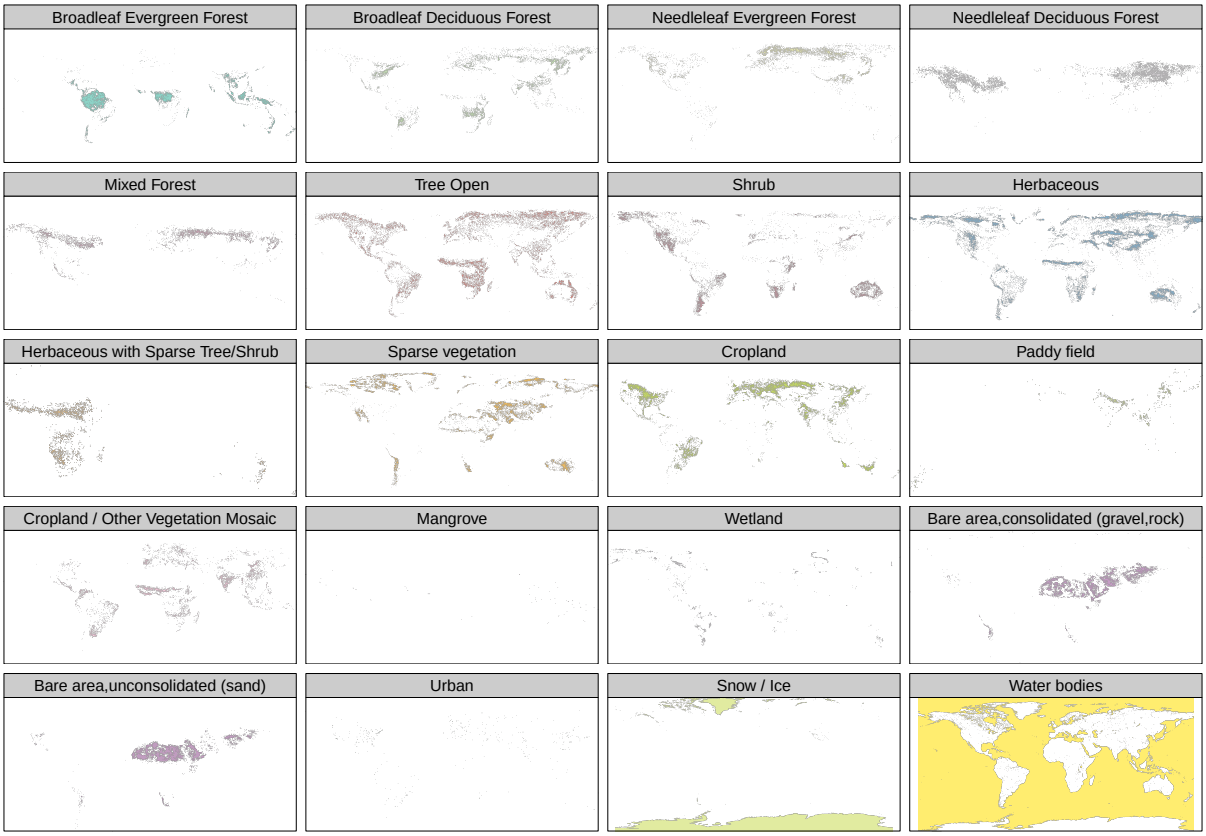



```
# plot land and add categorical land cover
tm_shape(land) +
    tm_raster("cover")
```

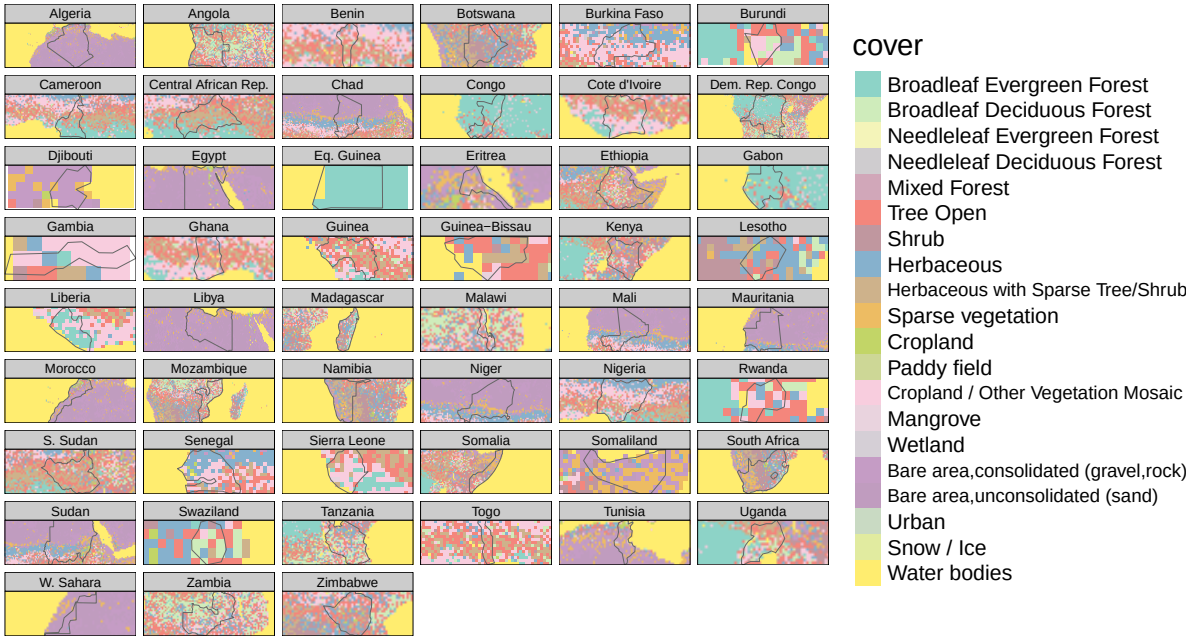



```
# just for a selected country
tm_shape(World[which(World$name=='South Africa'),], is.master=TRUE) +
    tm_borders() +
tm_shape(land) +
    tm_raster("cover")
```


```
# using faceting to create multiple plots

# facet by cover to get one map for each cover type
tm_shape(land) +
    tm_raster("cover", legend.show = FALSE) +
    tm_facets("cover", free.coords=TRUE, drop.units=TRUE)
```




```
# facet by country to get one cover map per country
tm_shape(land) +
    tm_raster("cover") +
tm_shape(World[which(World$continent=='Africa'),]) +
    tm_borders() +
    tm_facets("name", free.coords = TRUE)
```



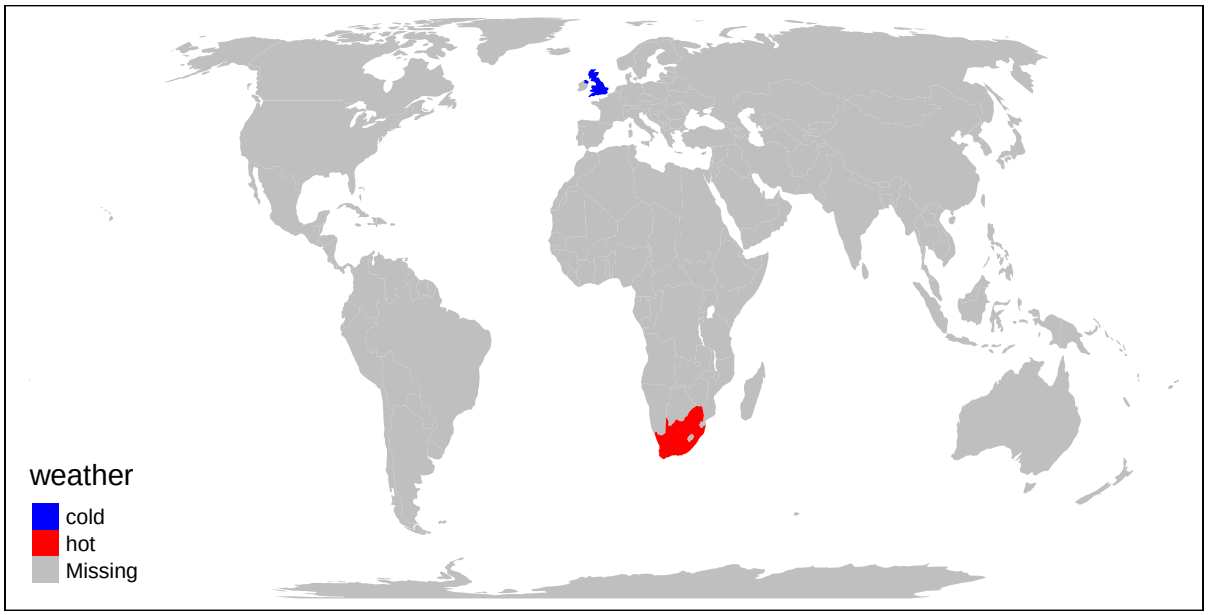
4 map your own data in tmap

You can use both your own geometry and/or attribute data in tmap and other mapping options.

```
# create a very simple data frame as an example
dF <- data.frame( country=c("South Africa", "United Kingdom"),
                  weather=c("hot", "cold") )

# tmap can join this onto a map
World_and_dat <- append_data(World, dF, key.shp="name", key.data="country")
```

```
## Under coverage. No data for 175 out of 177 polygons: Afghanistan, Angola, Albania, United Arab Emira
```

```
tm_shape(World_and_dat) +
  tm_fill("weather", palette=c("blue", "red"))
```




```
# Can you change this to plot a different map ?
# Maybe add other countries or a different variable.
```

5 Interactive web maps with leaflet

leaflet is an R package that links to a web-mapping tool also called leaflet, with this you can create interactive maps.

```
#install if not installed already
if (require(leaflet)) install.packages("leaflet")
```

```
library(leaflet)
# to see help on the package
?leaflet

# create a default map
mymap = leaflet() %>% addTiles()

# use %>% (called a 'pipe') to modify the map

# set view by a point (long, lat) and zoom level
mymap %>% setView(32, -26, zoom=10)

# set view by the edges
```

```
#fitBounds(lng1, lat1, lng2, lat2)
mymap %>% fitBounds(30, -29, 35, -24)
```

6 A short ggmap example for Thoyoyando

```
#install if not installed already
if (require(ggmap)) install.packages("ggmap")
library(ggmap)
mymap <- get_map("thoyoyando, south africa")
ggmap(mymap)

mymap <- get_map("thoyoyando, south africa", maptype='satellite', zoom=15)
ggmap(mymap)

#creating a dataframe with a point in to add to the map
tho_points <- data.frame(lon=c(30.48),lat=c(-22.88),class=c("house"))

ggmap(mymap) +
  geom_point( aes(x = lon, y = lat, colour=class), data = tho_points)
```

Going further, other useful resources and packages

tmap in a nutshell

A more advanced mapping tutorial

```
# a new package for getting world bank data
install.packages("wbstats")
# http://www.r-bloggers.com/new-r-package-to-access-world-bank-data/

# raster package, great for mapping satellite data etc.
install.packages("raster")
library(raster)

# rmapshaper for simplifying polygon boundaries
install.packages("rmapshaper")
library(rmapshaper)

# leaflet a package for creating interactive maps
install.packages("leaflet")
```

Acknowledgements :

Thankyou to all the package developers on whose work this tutorial is based.