

make_maps.R

joebrew

Thu Aug 27 10:41:05 2015

```
#####
```

```
# PACKAGES
```

```
#####
```

```
library(rgdal)
```

```
## Loading required package: sp
## rgdal: version: 1.0-4, (SVN revision 548)
## Geospatial Data Abstraction Library extensions to R successfully loaded
## Loaded GDAL runtime: GDAL 1.11.2, released 2015/02/10
## Path to GDAL shared files: /usr/share/gdal/1.11
## Loaded PROJ.4 runtime: Rel. 4.8.0, 6 March 2012, [PJ_VERSION: 480]
## Path to PROJ.4 shared files: (autodetected)
## Linking to sp version: 1.1-1
```

```
library(readr)
library(readxl)
library(ggplot2)
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'
```

```
##
```

```
## The following object is masked from 'package:stats':
```

```
##
```

```
##     filter
```

```
##
```

```
## The following objects are masked from 'package:base':
```

```
##
```

```
##     intersect, setdiff, setequal, union
```

```
#####
```

```
# DIRECTORIES
```

```
#####
```

```
#wd <- getwd() # Be in the mortality_map/code directory -- change only if needed
```

```
#####
```

```
# LOAD CHEN'S DATA
```

```
#####
```

```
df <- read_excel('..../data/dataofcountries.xls')
```

```
## DEFINEDNAME: 00 00 00 05 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 53 68 65 65 74 3b 00 00 00 00 c2 00 00 00
## DEFINEDNAME: 00 00 00 05 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 53 68 65 65 74 3b 00 00 00 00 c2 00 00 00
## DEFINEDNAME: 00 00 00 05 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 53 68 65 65 74 3b 00 00 00 00 c2 00 00 00
## DEFINEDNAME: 00 00 00 05 0b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 53 68 65 65 74 3b 00 00 00 00 c2 00 00 00
```

```

#####  

# LOAD IN THE WORLD SHAPEFILE  

#####  

world <- readOGR('../data/countries_shape_file/', 'world_borders')

## OGR data source with driver: ESRI Shapefile
## Source: "../data/countries_shape_file/", layer: "world_borders"
## with 3784 features
## It has 5 fields

#####  

# EXPLORE THE SHAPEFILE  

#####  

# Examine the format
head(world@data)

```

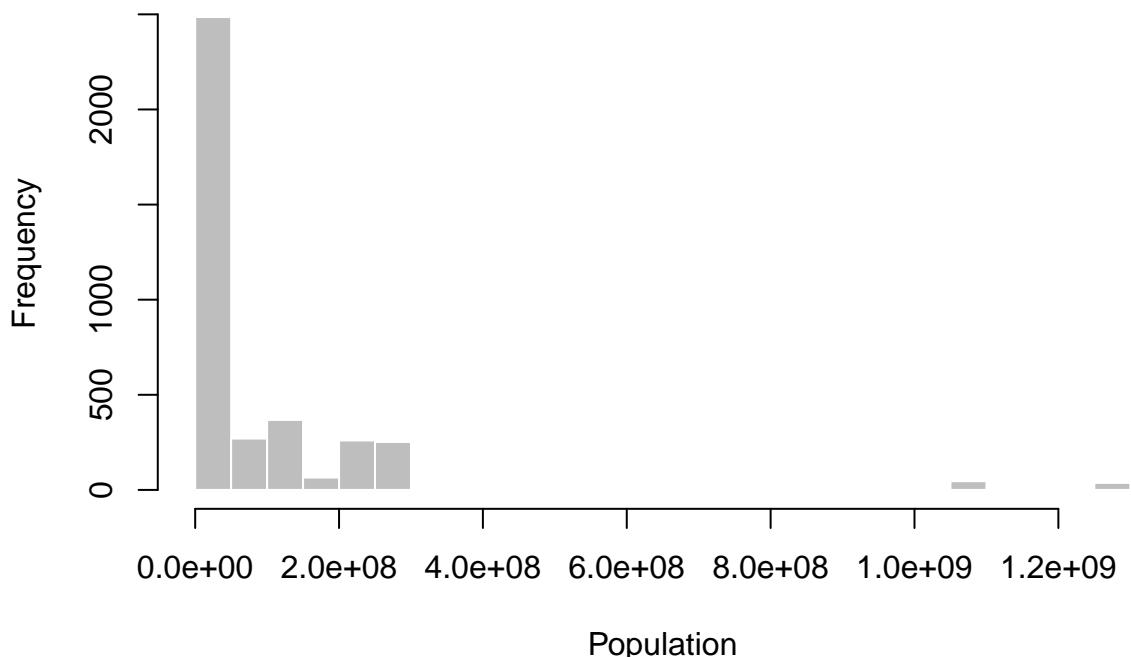
```

##   CAT FIPS_CNTRY      CNTRY_NAME     AREA POP_CNTRY
## 0   1       AA        Aruba      193    71218
## 1   2       AC  Antigua and Barbuda    443    68320
## 2   2       AC  Antigua and Barbuda    443    68320
## 3   4       AG        Algeria 2381740  32129324
## 4   5       AJ      Azerbaijan  86600  7868385
## 5   6       AL      Albania    28748  3544808

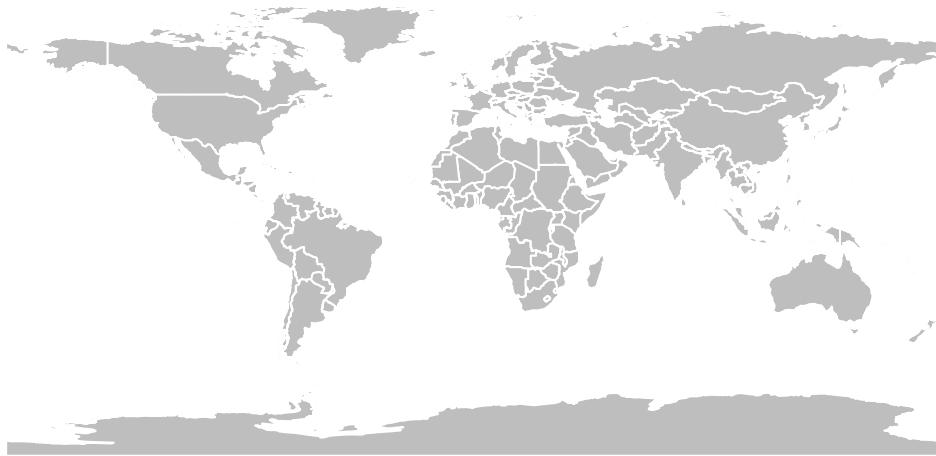
# Look at distribution of one variable
hist(world@data$POP_CNTRY, breaks = 30, col = 'grey', border = 'white',
     xlab = 'Population', main = 'Distribution of countries\' population')

```

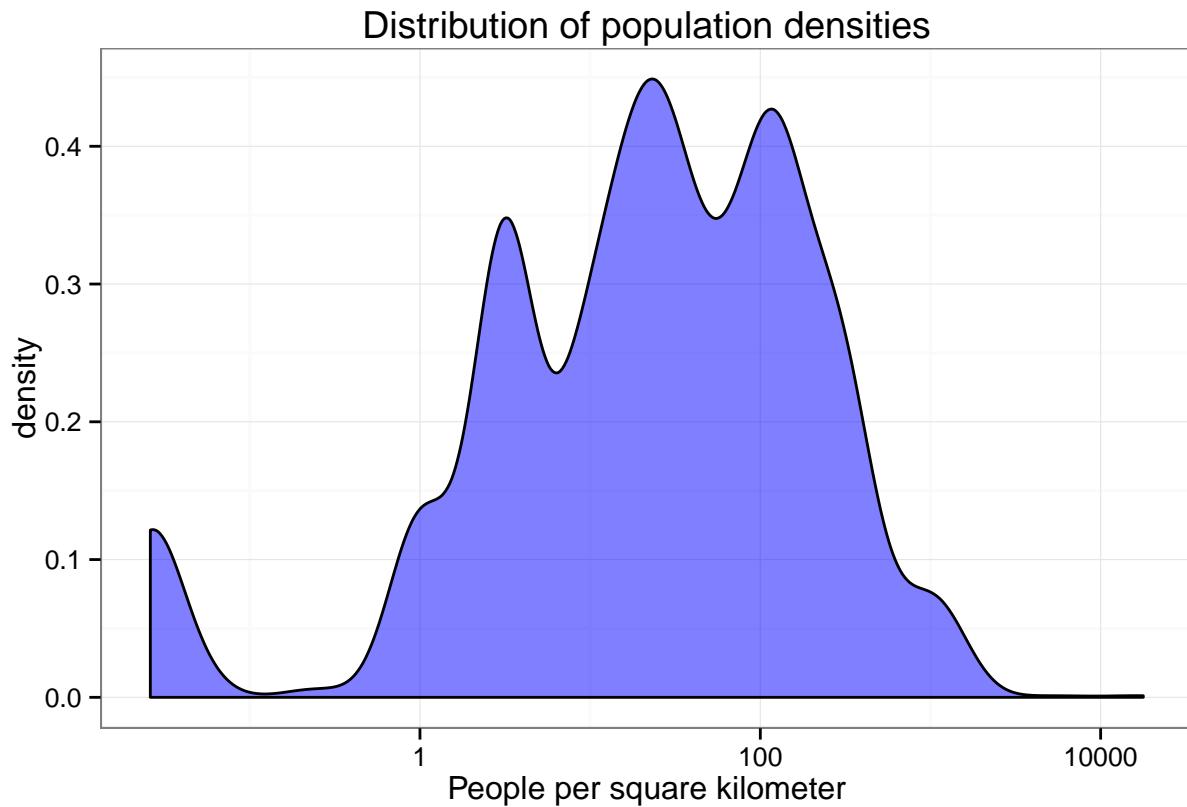
Distribution of countries' population



```
# Make a simple map  
plot(world, col = 'grey', border = 'white')
```

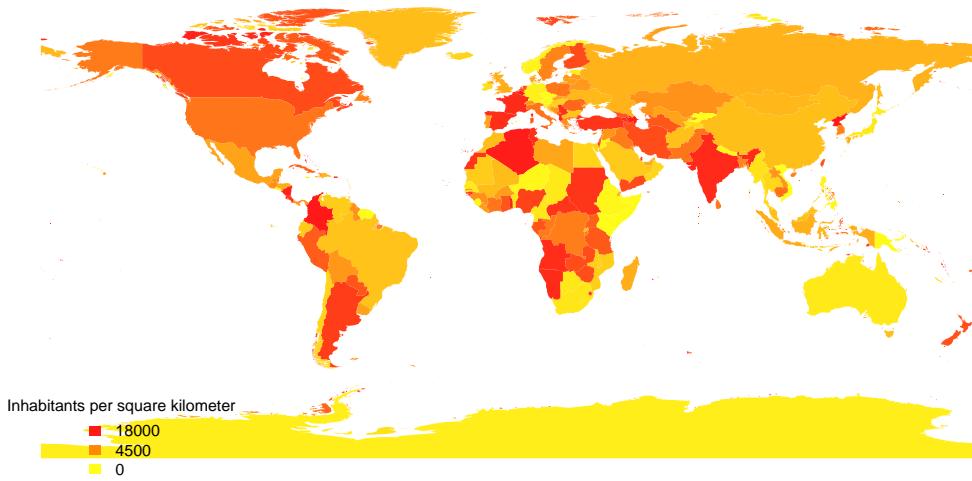


```
# Make a new variable in world: population density  
# this will be each countries' population divided by its area  
world@data$POP_DEN <- world@data$POP_CNTRY / world@data$AREA  
  
# Examine the distribution of population densities (this time using ggplot instead of base graphics)  
ggplot(data = world@data, aes(x = POP_DEN)) +  
  geom_density(fill = 'blue', alpha = 0.5) +  
  theme_bw() +  
  xlab('People per square kilometer') +  
  ggtitle('Distribution of population densities') +  
  scale_x_log10() # put on log 10 scale
```



```
# Make a choropleth in which each country is shaded by the square root of its population density
world@data$POP_DEN_SQRT <- ceiling(sqrt(world@data$POP_DEN))
colors <- colorRampPalette(c('red', 'yellow'))(max(world@data$POP_DEN_SQRT)) %>%
  adjustcolor(alpha.f = 0.9)
plot(world, col = colors, border = NA, main = 'Population density')
legend('bottomleft',
  fill = colors[c(1, length(colors)/2, length(colors))],
  legend = rev(round(c(1, length(colors)/2, length(colors)) ^ 2, digits = -2)),
  title = 'Inhabitants per square kilometer',
  bty = 'n',
  cex = 0.5,
  border = NA)
```

Population density



```
#####
# MERGE THE MORTALITY AND LIFE EXPECTANCY DATA (df) WITH THE WORLD SHAPEFILE
#####
```

```
# We will join on the 'Country' column in df
# and the 'CNTRY_NAME' column in world
# However, since these are unlikely to be identically named
# (and we don't want to take the time to manually change them)
# we'll write a "fuzzy matching" algorithm, which will
# create a column in df with the 'CNTRY_NAME' from world@data
# which is most similar to that row's 'Country' column
```

```
fuzzy_match <- function(x = world@data$CNTRY_NAME,
                        y = df$Country){
  # Create matrix of "string distance" between names
  distances <- adist(x = x, y = y)
  # For each row, get the index of the lowest (best-matching) column
  best_matches <-
    apply(distances, 1, function(z){
      possibs <- which.min(z)
      return(possibs[1])
    })
  # Return the actual names of the best matches
  names <- y[best_matches]
  distance <- apply(distances, 1, min)
  return_object <- list('name' = names, 'distance' = distance)
  return(return_object)
}

# Now, use our algorithm to populate a new column in "world"
# with the names from "df" (so that we can correctly merge)
results <- fuzzy_match()
world@data$Country <- results$name
```

```

# We can also populate a column which shows us the "string distance"
# (ie, how closely matched the terms were)
# high numbers are bad, low numbers are good.  0 is a perfect match
world@data$accuracy <- results$distance

# Now let's take another look at our world shapefile
head(world@data)

```

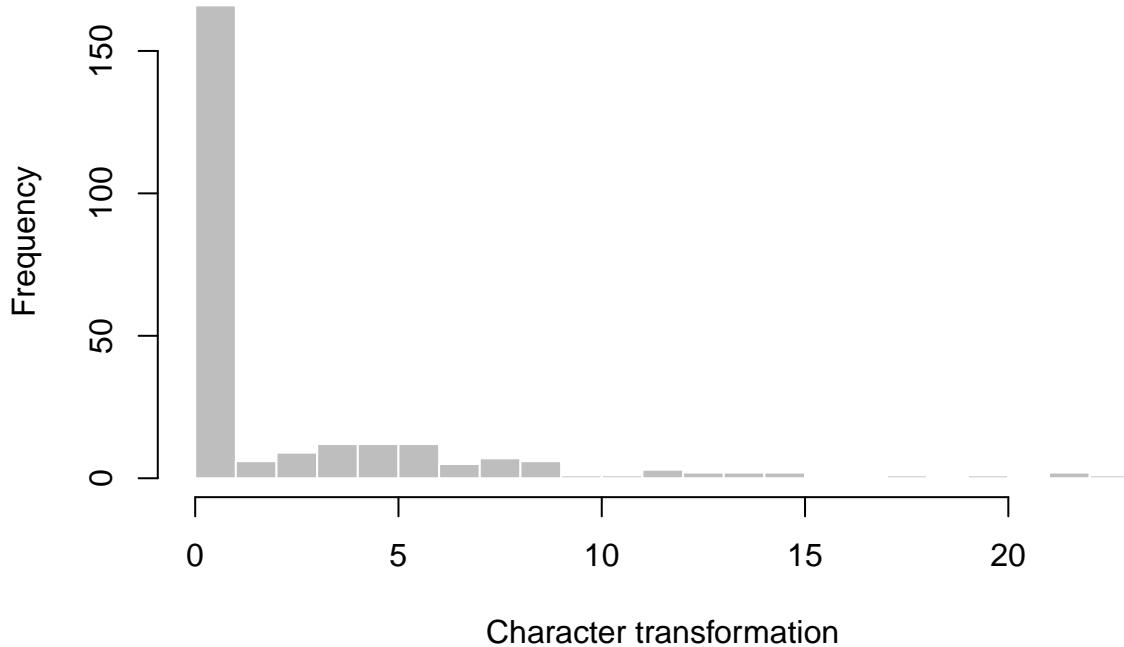
##	CAT	FIPS_CNTRY	CNTRY_NAME	AREA	POP_CNTRY	POP_DEN
## 0	1	AA	Aruba	193	71218	369.00518
## 1	2	AC	Antigua and Barbuda	443	68320	154.22122
## 2	2	AC	Antigua and Barbuda	443	68320	154.22122
## 3	4	AG	Algeria	2381740	32129324	13.48985
## 4	5	AJ	Azerbaijan	86600	7868385	90.85895
## 5	6	AL	Albania	28748	3544808	123.30625
##		POP_DEN_SQRT	Country accuracy			
## 0		20	Cuba	2		
## 1		13	Antigua and Barbuda	0		
## 2		13	Antigua and Barbuda	0		
## 3		4	Algeria	0		
## 4		10	Azerbaijan	0		
## 5		12	Albania	0		

```

# And examine the accuracy results
hist(world@data[!duplicated(world@data$FIPS_CNTRY),]$accuracy, breaks = 30,
     xlab = 'Character transformation',
     main = 'Accuracy (0 = perfect)',
     col = 'grey',
     border = 'white')

```

Accuracy (0 = perfect)



```
# So, our algorithm was pretty good, but there are still a few countries with
# pretty bad results. For example, let's just look at those
# countries who took more than 10 character transformations
world@data[which(world@data$accuracy >= 10 & !duplicated(world@data$FIPS_CNTRY)),
           c('CNTRY_NAME', 'Country')]
```

	CNTRY_NAME	Country
##		
## 1189	Cocos (Keeling) Islands	Cook Islands
## 1208	Northern Mariana Islands	Marshall Islands
## 1407	Falkland Islands (Islas Malvinas)	Bosnia and Herzegovina
## 1417	Federated States of Micronesia	United States of America
## 1489	French Southern & Antarctic Lands	South Africa
## 1820	Heard Island & McDonald Islands	Marshall Islands
## 2158	British Indian Ocean Territory	Trinidad and Tobago
## 2274	Juan De Nova Island	Cook Islands
## 2813	Pacific Islands (Palau)	Cook Islands
## 3187	St. Pierre and Miquelon	Sierra Leone
## 3280	South Georgia and the South Sandwich Islands	Saint Vincent and the Grenadines
## 3331	Turks and Caicos Islands	Marshall Islands
## 3387	Tanzania, United Republic of	Iran (Islamic Republic of)
## 3392	United Kingdom	Montenegro
## 3723	British Virgin Islands	Marshall Islands
## 3766	Wallis and Futuna	Saint Lucia

```
# Clearly, the above isn't good!
```

```
# However, the algorithm did pretty good on most
world@data[which(world@data$accuracy <=2 & !duplicated(world@data$FIPS_CNTRY)),
           c('CNTRY_NAME', 'Country')][1:10,]
```

```

##          CNTRY_NAME      Country
## 0          Aruba          Cuba
## 1  Antigua and Barbuda Antigua and Barbuda
## 3          Algeria        Algeria
## 4        Azerbaijan    Azerbaijan
## 5          Albania        Albania
## 8          Armenia        Armenia
## 11         Angola         Angola
## 19         Argentina      Argentina
## 25        Australia      Australia
## 261        Bahrain        Bahrain

# For now, we'll use the algorithm. If you want a perfect match,
# you'll need to go through your spreadsheet (dataofcountries.xls)
# and rename "Country" so that each name is one of the
# elements of the "correct_countries" vector
correct_countries <- unique(sort(world@data$CNTRY_NAME))

# (I've printed the entire vector at the end of this document)
# and then re-run the code

#####
# MERGE
#####
# Anyway, now that MOST of the data is matched, we'll merge
world@data <- left_join(x = world@data, y = df)

## Joining by: "Country"

# Now we can see that we've brought the data in from df into world@data
head(world@data)

```

```

##   CAT FIPS_CNTRY      CNTRY_NAME     AREA POP_CNTRY  POP_DEN
## 1   1       AA          Aruba     193  71218 369.00518
## 2   2       AC  Antigua and Barbuda  443  68320 154.22122
## 3   2       AC  Antigua and Barbuda  443  68320 154.22122
## 4   4       AG          Algeria 2381740 32129324 13.48985
## 5   5       AJ        Azerbaijan 86600  7868385 90.85895
## 6   6       AL          Albania 28748 3544808 123.30625
##   POP_DEN_SQRT      Country accuracy MR2000_100000population
## 1           20          Cuba        2            564.2
## 2           13  Antigua and Barbuda  0             NA
## 3           13  Antigua and Barbuda  0             NA
## 4           4          Algeria  0            935.8
## 5           10        Azerbaijan  0            1027.4
## 6           12          Albania  0            986.5
##   MR2012_100000population LE2000_year LE2012_year
## 1           500.3        77        79
## 2             NA        73        75
## 3             NA        73        75
## 4           862.0        70        72
## 5           768.8        66        72
## 6           766.1        70        74

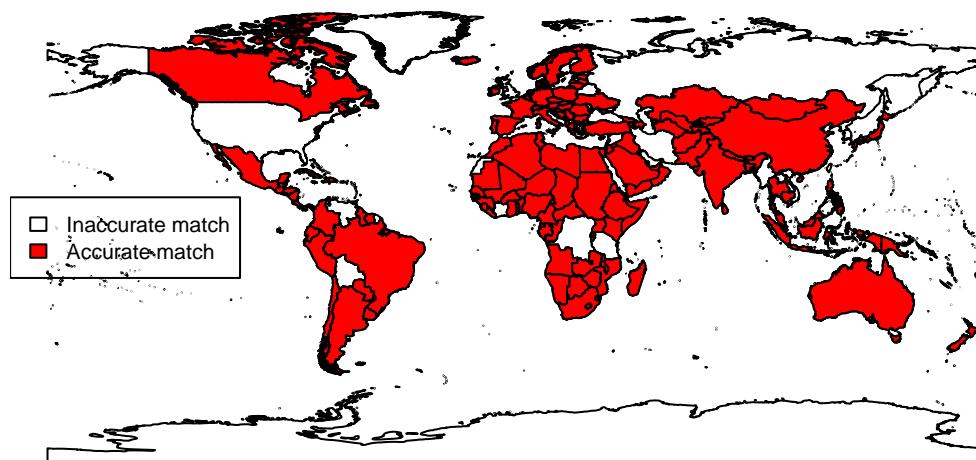
```

```

# Since the algorithm wasn't able to match ALL the countries,
# we're only going to keep those perfect accuracy
world_accurate <- world[which(world@data$accuracy == 0),]

# Let's see which countries we were able to keep
plot(world)
plot(world_accurate, col = 'red', add = TRUE)
legend('left',
       fill = c('white', 'red'),
       legend = c('Inaccurate match', 'Accurate match'),
       cex = 0.7)

```



```

#####
# PLOT
#####

# Let's define a simple function for plotting a choropleth
choro <- function(shape = world_accurate,
                    background = world,
                    colors = c('red', 'yellow', 'blue'),
                    variable = 'MR2000_100000population'){
variables <- c('MR2000_100000population', 'MR2012_100000population',
             'LE2000_year', 'LE2012_year')
if(!variable %in% variables){
  stop(paste0('variable must be one of ', paste0(variables, collapse = ', ')))
}
plot(background, col = 'grey', border = NA)

# Get title information
tit <- paste0(
  ifelse(substr(variable, 1,2) == 'MR', 'Morality', 'Life expectancy'),
  ' in ',
  substr(variable, 3,6)
)

# Create a var for plotting
var <- ceiling(shape@data[,variable])
cols <- colorRampPalette(colors)(max(var, na.rm = TRUE))

```

```

# (reverse color scheme if mortality)
if(substr(variable, 1, 2) == 'MR'){
  cols <- rev(cols)
}
var_cols <- cols[var]

# Plot
plot(shape, col = var_cols, add = TRUE, border = NA)
title(main = tit)

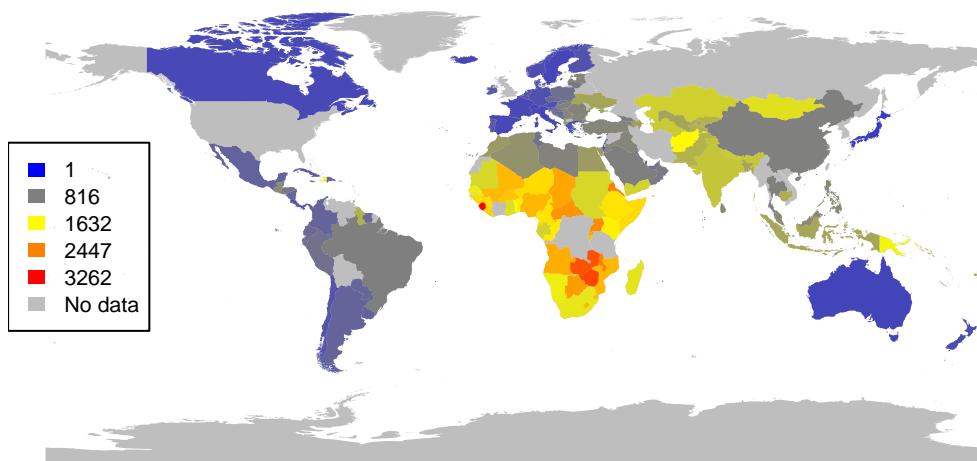
# Legend
legend_vec <- 1:length(cols)
legend_quant <- round(as.numeric(quantile(legend_vec, na.rm = TRUE)))
legend('left',
       fill = c(cols[legend_quant], 'grey'),
       legend = c(legend_quant, 'No data'),
       cex = 0.7,
       border = NA)
}

# Now let's plot all of our variables
variables <- c('MR2000_100000population', 'MR2012_100000population',
              'LE2000_year', 'LE2012_year')

choro(variable = variables[1])

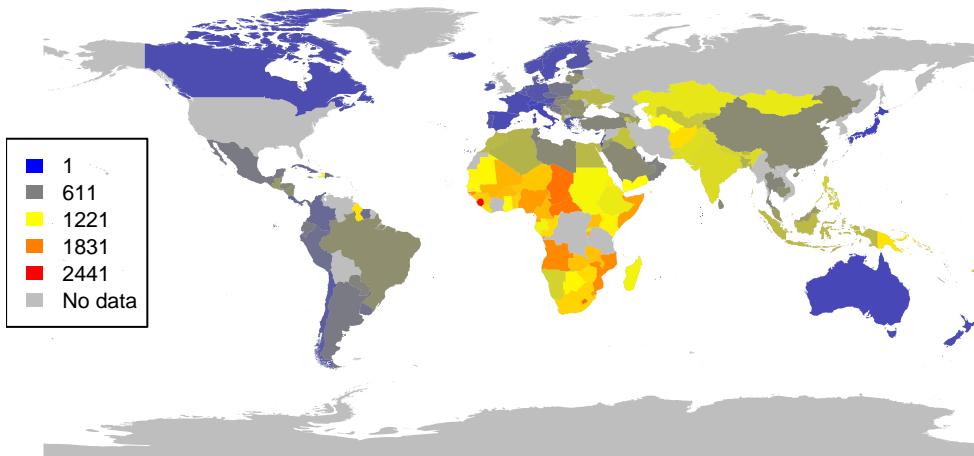
```

Morality in 2000



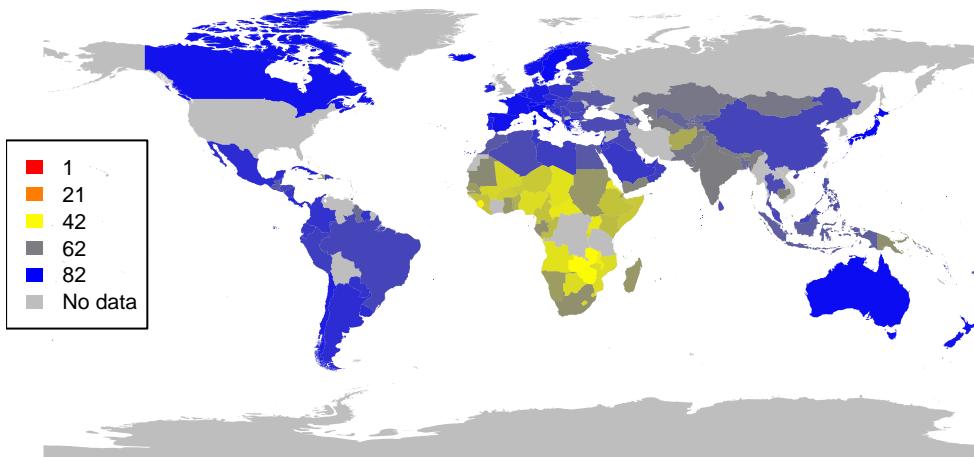
```
choro(variable = variables[2])
```

Morality in 2012



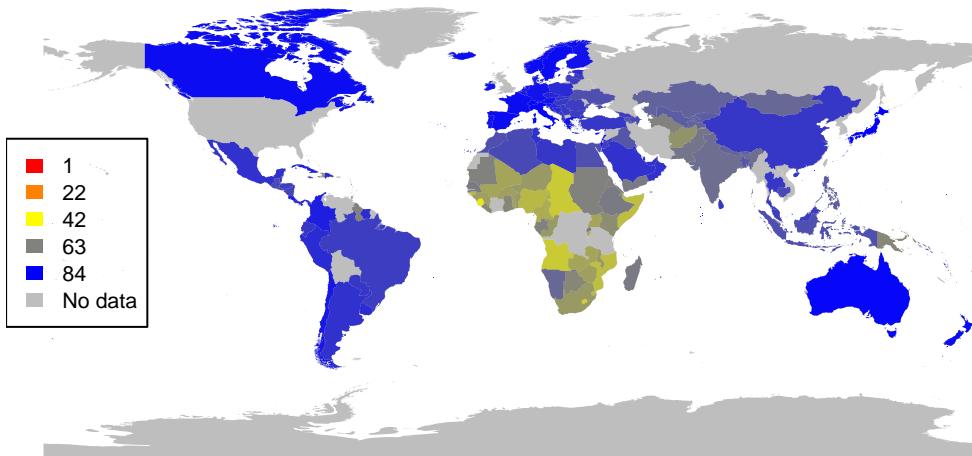
```
choro(variable = variables[3])
```

Life expectancy in 2000



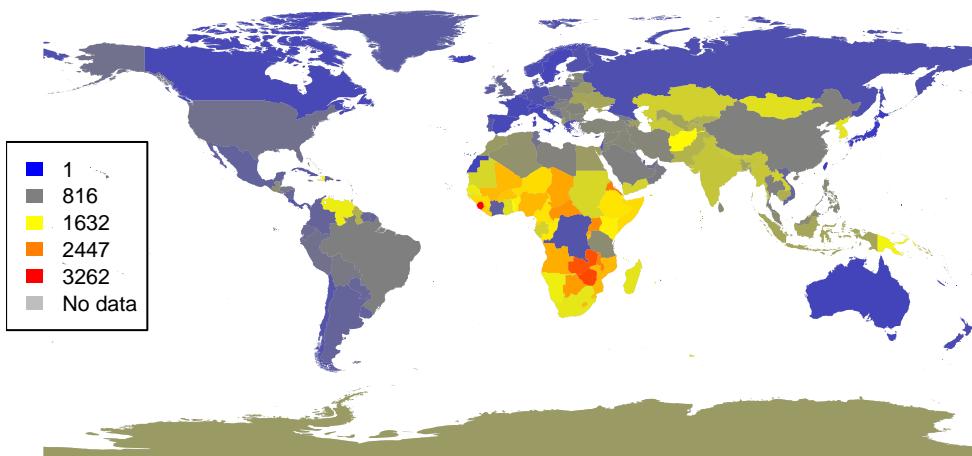
```
choro(variable = variables[4])
```

Life expectancy in 2012



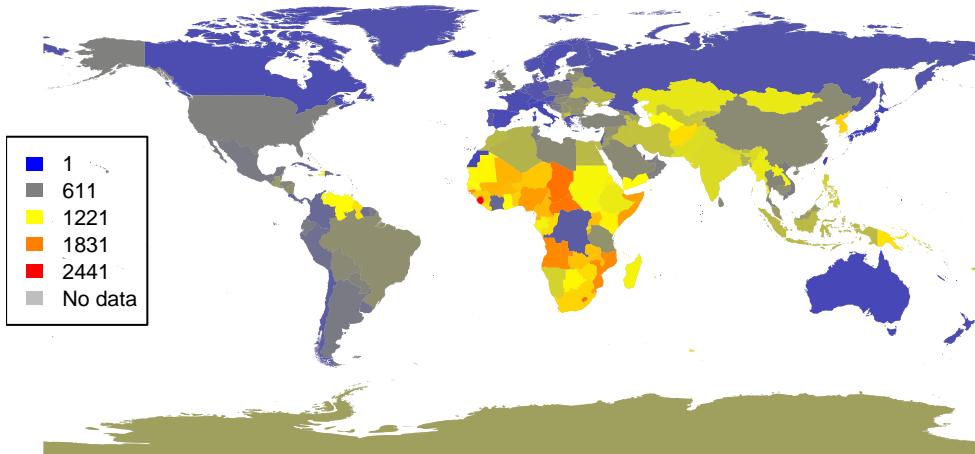
```
# Alternatively, we could plot all the countries (even those we know weren't correctly matched)
choro(variable = variables[1], shape = world)
```

Morality in 2000



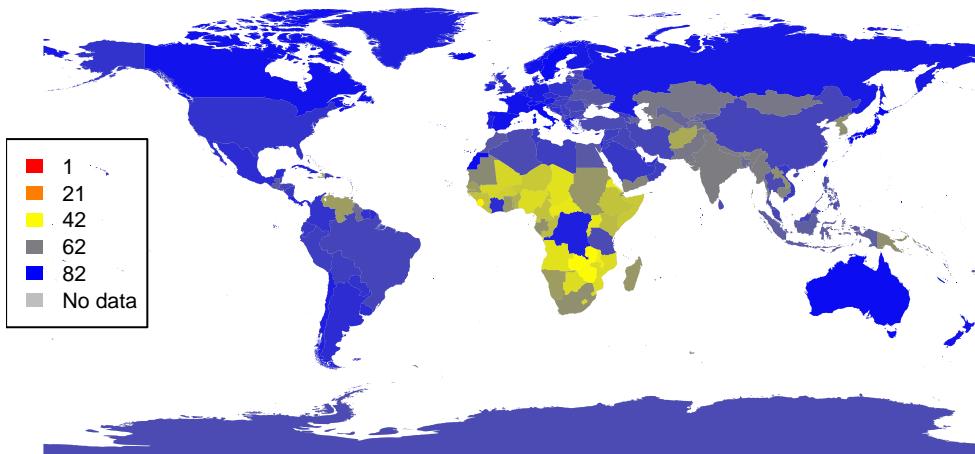
```
choro(variable = variables[2], shape = world)
```

Morality in 2012



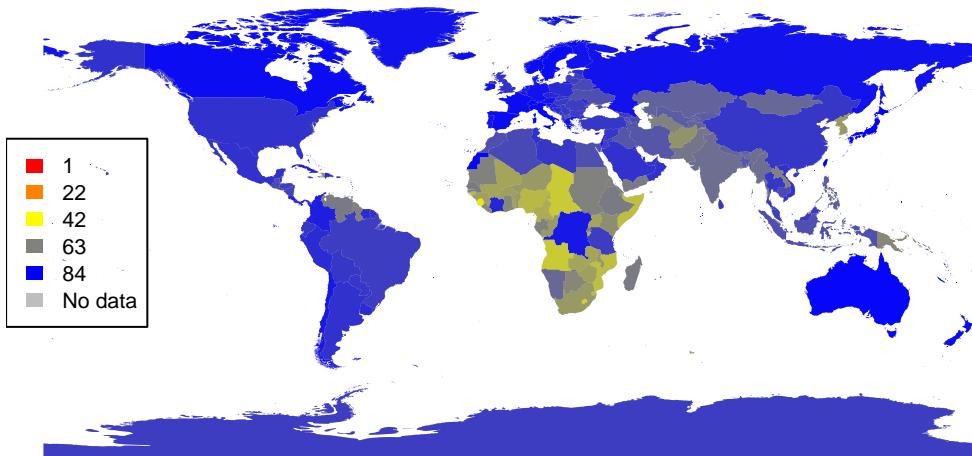
```
choro(variable = variables[3], shape = world)
```

Life expectancy in 2000



```
choro(variable = variables[4], shape = world)
```

Life expectancy in 2012



```
#####
```

```
# NEXT STEPS
```

```
#####
```

```
# In order to get the matches perfect,  
# go through the spreadsheet with your data  
# and rename all the countries to one of the following:  
print(correct_countries)
```

```
## [1] Afghanistan          Albania  
## [3] Algeria              American Samoa  
## [5] Andorra              Angola  
## [7] Anguilla             Antarctica  
## [9] Antigua and Barbuda Argentina  
## [11] Armenia             Aruba  
## [13] Australia            Austria  
## [15] Azerbaijan          Bahamas, The  
## [17] Bahrain              Baker Island  
## [19] Bangladesh           Barbados  
## [21] Belgium              Belize  
## [23] Benin                Bermuda  
## [25] Bhutan               Bolivia  
## [27] Bosnia and Herzegovina Botswana  
## [29] Bouvet Island        Brazil  
## [31] British Indian Ocean Territory British Virgin Islands  
## [33] Brunei               Bulgaria  
## [35] Burkina Faso        Burundi  
## [37] Belarus              Cambodia  
## [39] Cameroon             Canada  
## [41] Cape Verde            Cayman Islands  
## [43] Central African Republic Chad  
## [45] Chile                China  
## [47] Christmas Island     Cocos (Keeling) Islands  
## [49] Colombia             Comoros  
## [51] Congo                Cook Islands
```

## [53] Costa Rica	Croatia
## [55] Cuba	Cyprus
## [57] Czech Republic	Denmark
## [59] Djibouti	Dominica
## [61] Dominican Republic	Ecuador
## [63] Egypt	El Salvador
## [65] Equatorial Guinea	Eritrea
## [67] Estonia	Ethiopia
## [69] Falkland Islands (Islas Malvinas)	Faroe Islands
## [71] Federated States of Micronesia	Fiji
## [73] Finland	France
## [75] French Guiana	French Polynesia
## [77] French Southern & Antarctic Lands	Gabon
## [79] Gambia, The	Gaza Strip
## [81] Georgia	Germany
## [83] Ghana	Gibraltar
## [85] Glorioso Islands	Greece
## [87] Greenland	Grenada
## [89] Guadeloupe	Guam
## [91] Guatemala	Guernsey
## [93] Guinea	Guinea-Bissau
## [95] Guyana	Haiti
## [97] Heard Island & McDonald Islands	Honduras
## [99] Howland Island	Hungary
## [101] Iceland	India
## [103] Indonesia	Iran
## [105] Iraq	Ireland
## [107] Israel	Italy
## [109] Ivory Coast	Jamaica
## [111] Jan Mayen	Japan
## [113] Jarvis Island	Jersey
## [115] Johnston Atoll	Jordan
## [117] Juan De Nova Island	Kazakhstan
## [119] Kenya	Kiribati
## [121] Kuwait	Kyrgyzstan
## [123] Laos	Latvia
## [125] Lebanon	Lesotho
## [127] Liberia	Libya
## [129] Liechtenstein	Lithuania
## [131] Luxembourg	Macau
## [133] Macedonia	Madagascar
## [135] Malawi	Malaysia
## [137] Maldives	Mali
## [139] Malta	Man, Isle of
## [141] Marshall Islands	Martinique
## [143] Mauritania	Mauritius
## [145] Mayotte	Mexico
## [147] Midway Islands	Moldova
## [149] Monaco	Mongolia
## [151] Montenegro	Montserrat
## [153] Morocco	Mozambique
## [155] Myanmar (Burma)	Namibia
## [157] Nauru	Nepal
## [159] Netherlands	Netherlands Antilles

```

## [161] New Caledonia
## [163] Nicaragua
## [165] Nigeria
## [167] Norfolk Island
## [169] North Korea
## [171] Oman
## [173] Pakistan
## [175] Papua New Guinea
## [177] Paraguay
## [179] Philippines
## [181] Poland
## [183] Puerto Rico
## [185] Reunion
## [187] Russia
## [189] San Marino
## [191] Saudi Arabia
## [193] Serbia
## [195] Sierra Leone
## [197] Slovakia
## [199] Solomon Islands
## [201] South Africa
## [203] South Korea
## [205] Spratly Islands
## [207] St. Helena
## [209] St. Lucia
## [211] St. Vincent and the Grenadines
## [213] Suriname
## [215] Swaziland
## [217] Switzerland
## [219] Taiwan
## [221] Tanzania, United Republic of
## [223] Togo
## [225] Tonga
## [227] Tunisia
## [229] Turkmenistan
## [231] Tuvalu
## [233] Ukraine
## [235] United Kingdom
## [237] Uruguay
## [239] Vanuatu
## [241] Vietnam
## [243] Wake Island
## [245] West Bank
## [247] Western Samoa
## [249] Zaire
## [251] Zimbabwe
## 251 Levels: Afghanistan Albania Algeria American Samoa Andorra ... Zimbabwe
New Zealand
Niger
Niue
Northern Mariana Islands
Norway
Pacific Islands (Palau)
Panama
Paracel Islands
Peru
Pitcairn Islands
Portugal
Qatar
Romania
Rwanda
Sao Tome and Principe
Senegal
Seychelles
Singapore
Slovenia
Somalia
South Georgia and the South Sandwi
Spain
Sri Lanka
St. Kitts and Nevis
St. Pierre and Miquelon
Sudan
Svalbard
Sweden
Syria
Tajikistan
Thailand
Tokelau
Trinidad and Tobago
Turkey
Turks and Caicos Islands
Uganda
United Arab Emirates
United States
Uzbekistan
Venezuela
Virgin Islands
Wallis and Futuna
Western Sahara
Yemen
Zambia

```