CSSS 569 Visualizing Data and Models

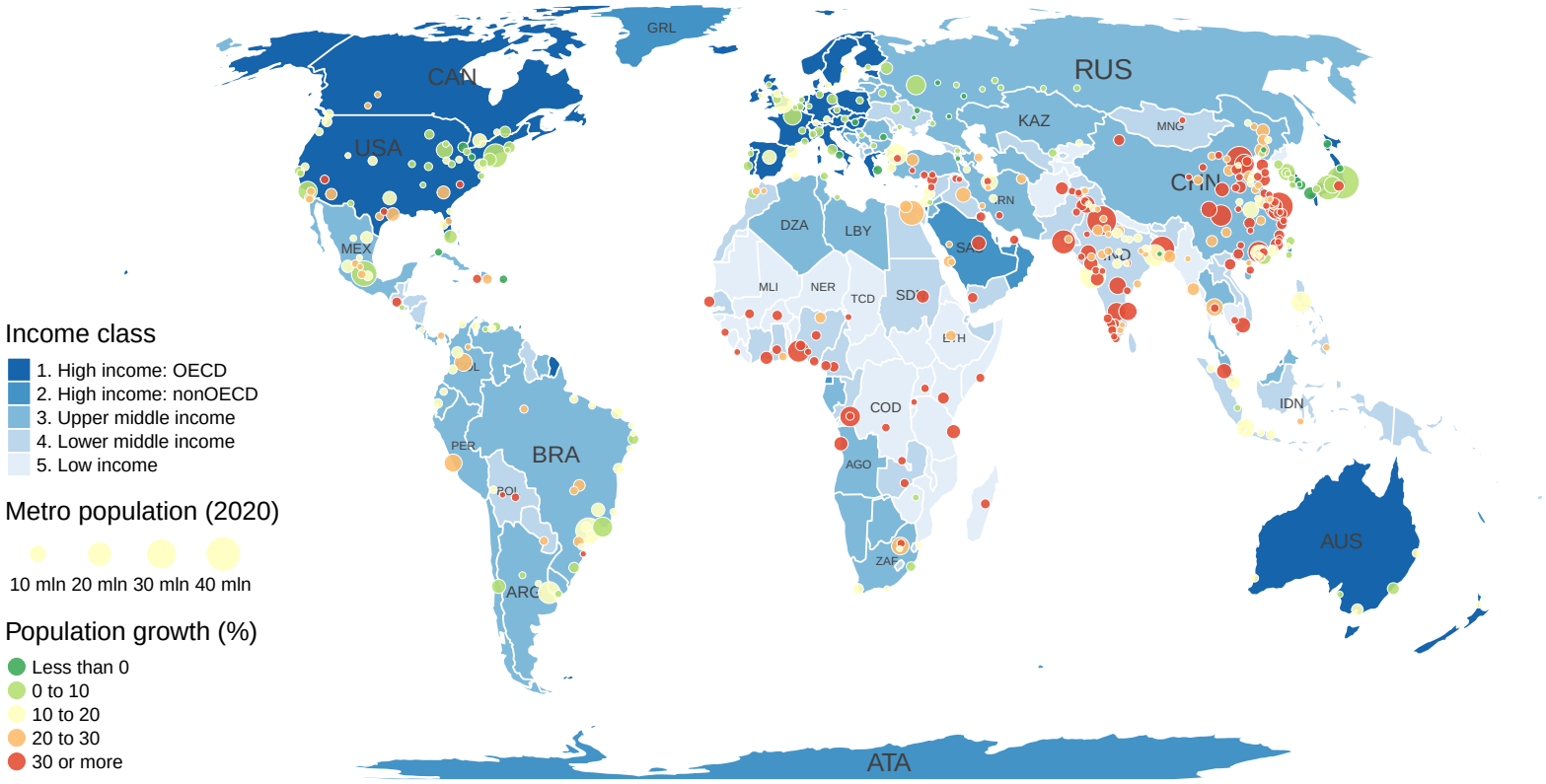
Lab 6: Visualizing Spatial Data

Brian Leung

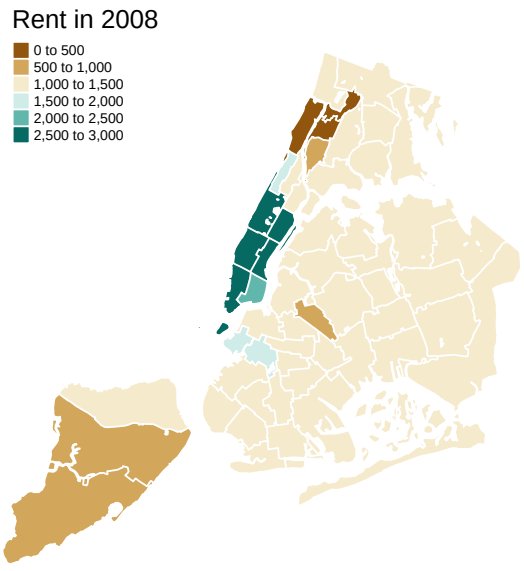
Department of Political Science, UW

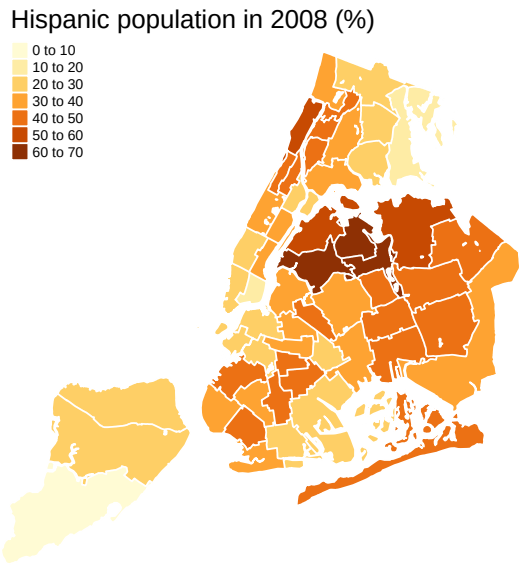
December 29, 2021

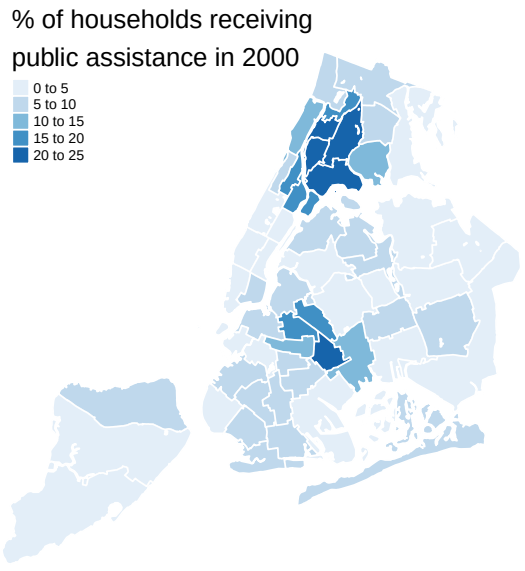
Introduction



Introduction







Dealing with and visualizing spatial data in R

- Numerous spatial data formats

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - **`.shp`** (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps
  - `tmap` allows easy visualization of static and interactive maps

Dealing with and visualizing spatial data in R

- Numerous spatial data formats
  - `.shp` (shapefile; the most common); `.geojson`, `.json`; `.gml`; `.csv`; `.tiff`...
- Countless packages to work with spatial data
  - Recent package `sf` allows geospatial data to be stored in data frames
  - Well integrated with `tidyverse`
- Many packages to draw maps
  - `tmap` allows easy visualization of static and interactive maps
  - Also employs the "grammar of graphics"

Overview of `tmap` package

	ggplot2	tmap
Data	`ggplot(...) +`	`tm_shape(...) +`
Layers	`geom_...(...) +`	`tm_...(...) +`
Small Multiples	`facet_grid(...)`	`tm_facets(...)`
Layout	`theme(...)`	`tm_layout(...)`

Prerequisite

```
install.packages(c("sf", "tmap"))
```

```
# Load packages
library(tidyverse)
library(sf)
library(tmap)

# Load data (from tmap)
data(World, metro)
```

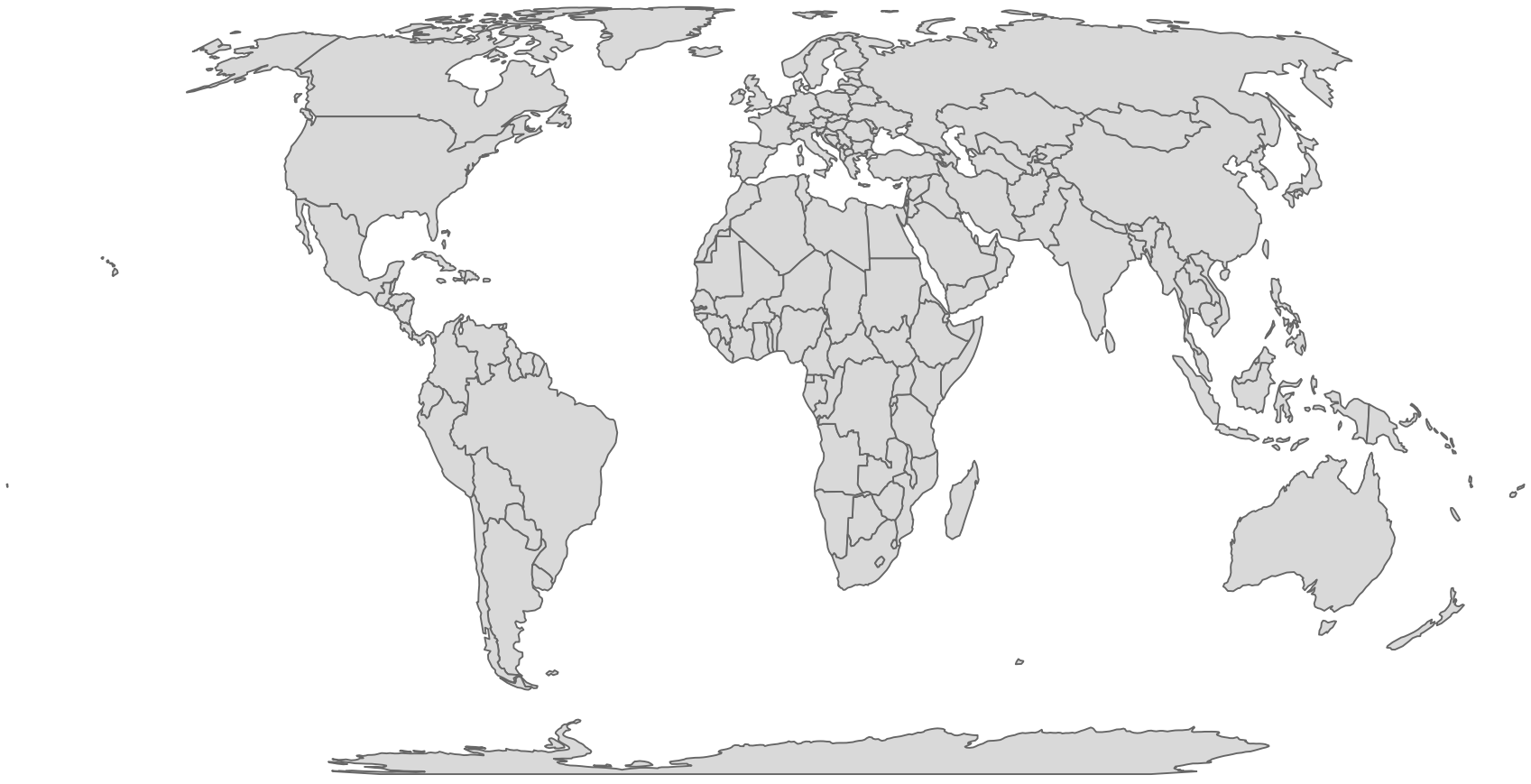
Basics tmap

```
print(World[1,])
```

```
## Simple feature collection with 1 feature and 15 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 5298517 ymin: 3762310 xmax: 6474206 ymax: 4839642
## CRS:            +proj=eck4 +lon_0=0 +x_0=0 +y_0=0 +datum=WGS84 +units=m +no_
##   iso_a3        name  sovereignt continent          area
## 1    AFG Afghanistan Afghanistan      Asia 652860 [km^2]
##    pop_est pop_est_dens                    economy
## 1 28400000      43.5009 7. Least developed region
##      income_grp gdp_cap_est life_exp well_being footprint
## 1 5. Low income    784.1549   59.668        3.8      0.79
##   inequality      HPI                        geometry
## 1  0.4265574 20.22535 MULTIPOLYGON (((5310471 451...
```

Basics tmap

```
tm_shape(World) +
  tm_polygons() +
  tm_layout(frame = FALSE)
```



Basics tmap

`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

```
tm_shape(World) +
  tm_borders() +
  tm_layout(frame = FALSE)
```

Basics tmap

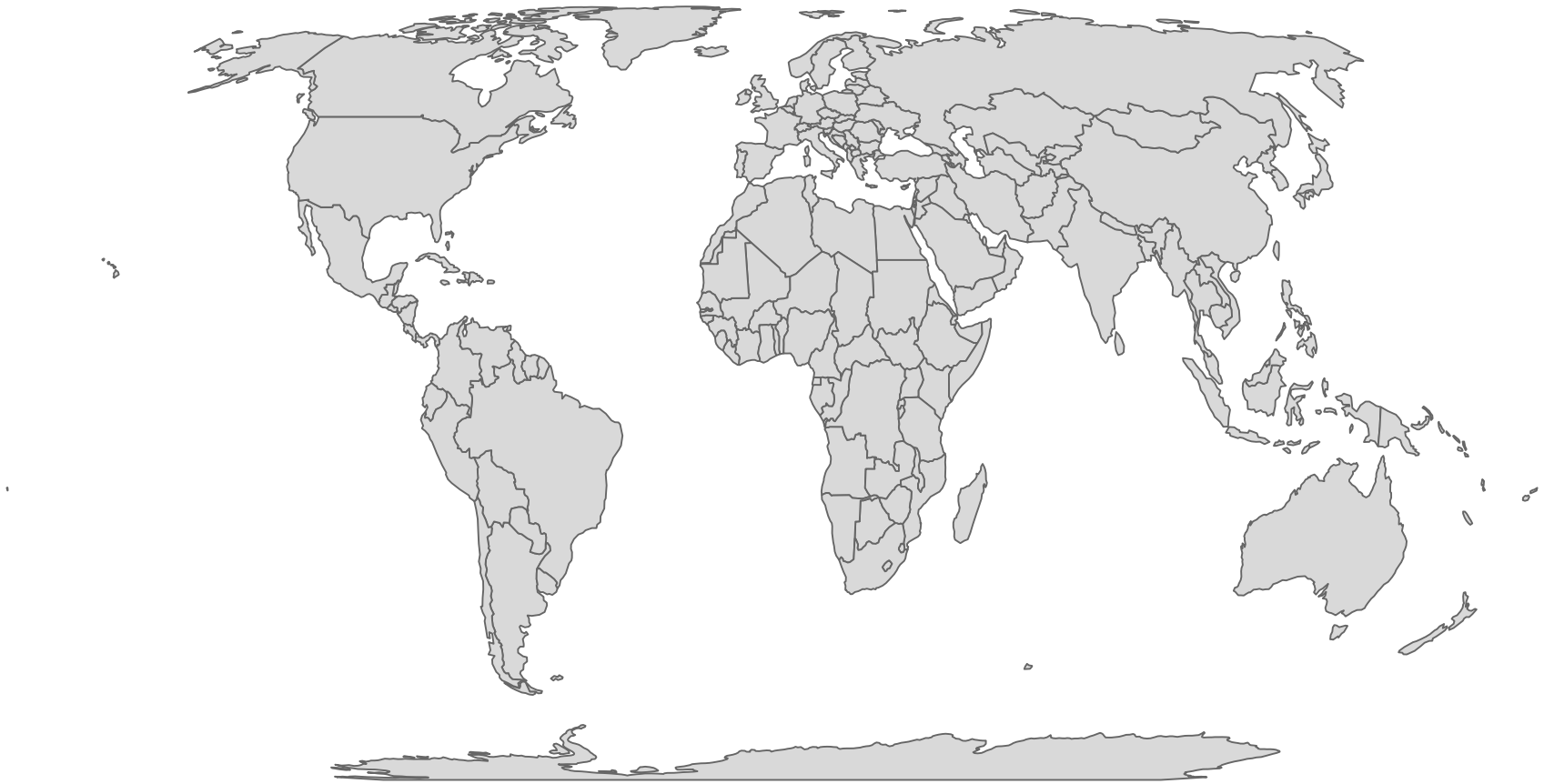
`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

```
tm_shape(World) +
  tm_fill() +
  tm_layout(frame = FALSE)
```

Basics tmap

`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

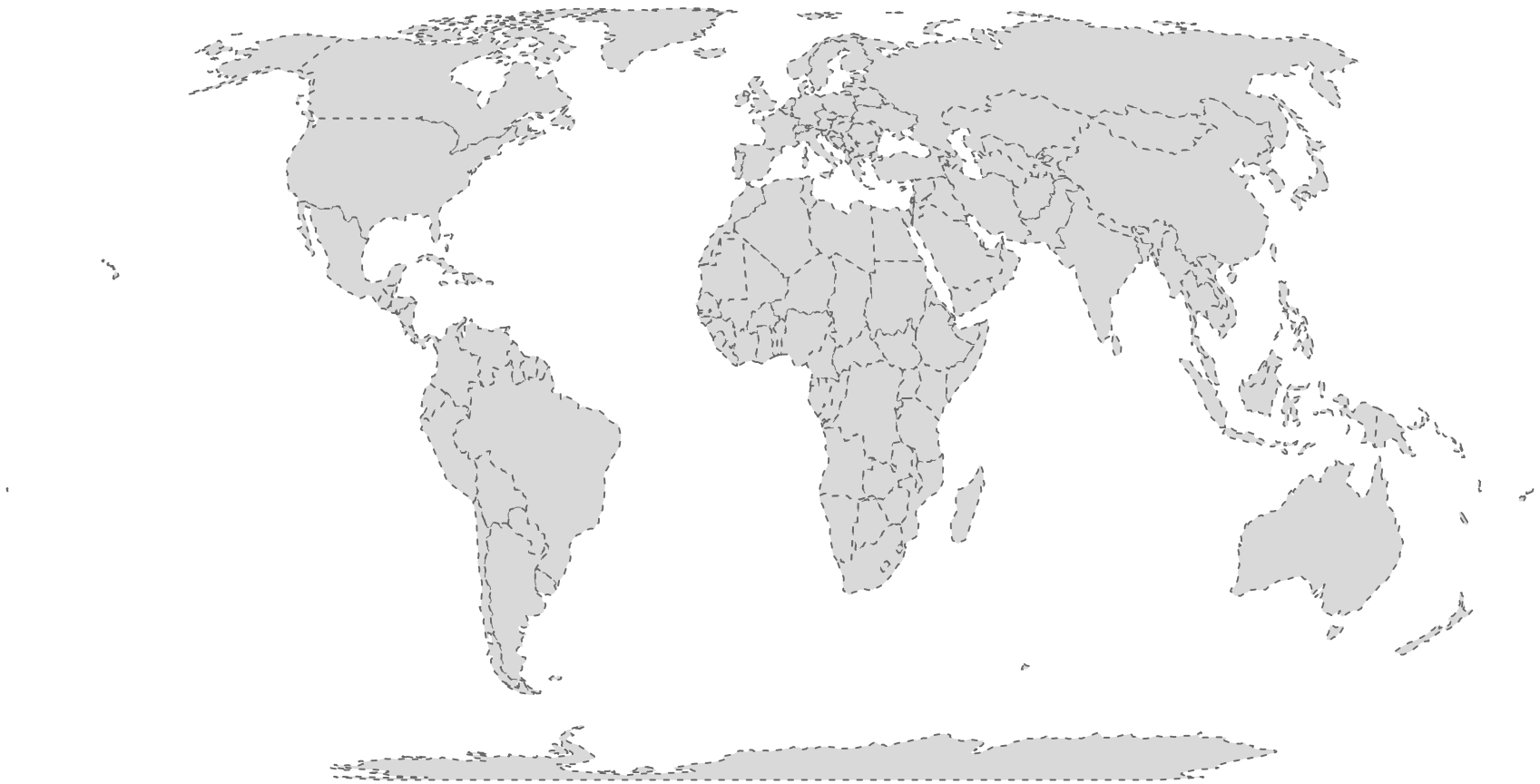
```
tm_shape(World) +
  tm_borders() +
  tm_fill() +
  tm_layout(frame = FALSE)
```



Basics tmap

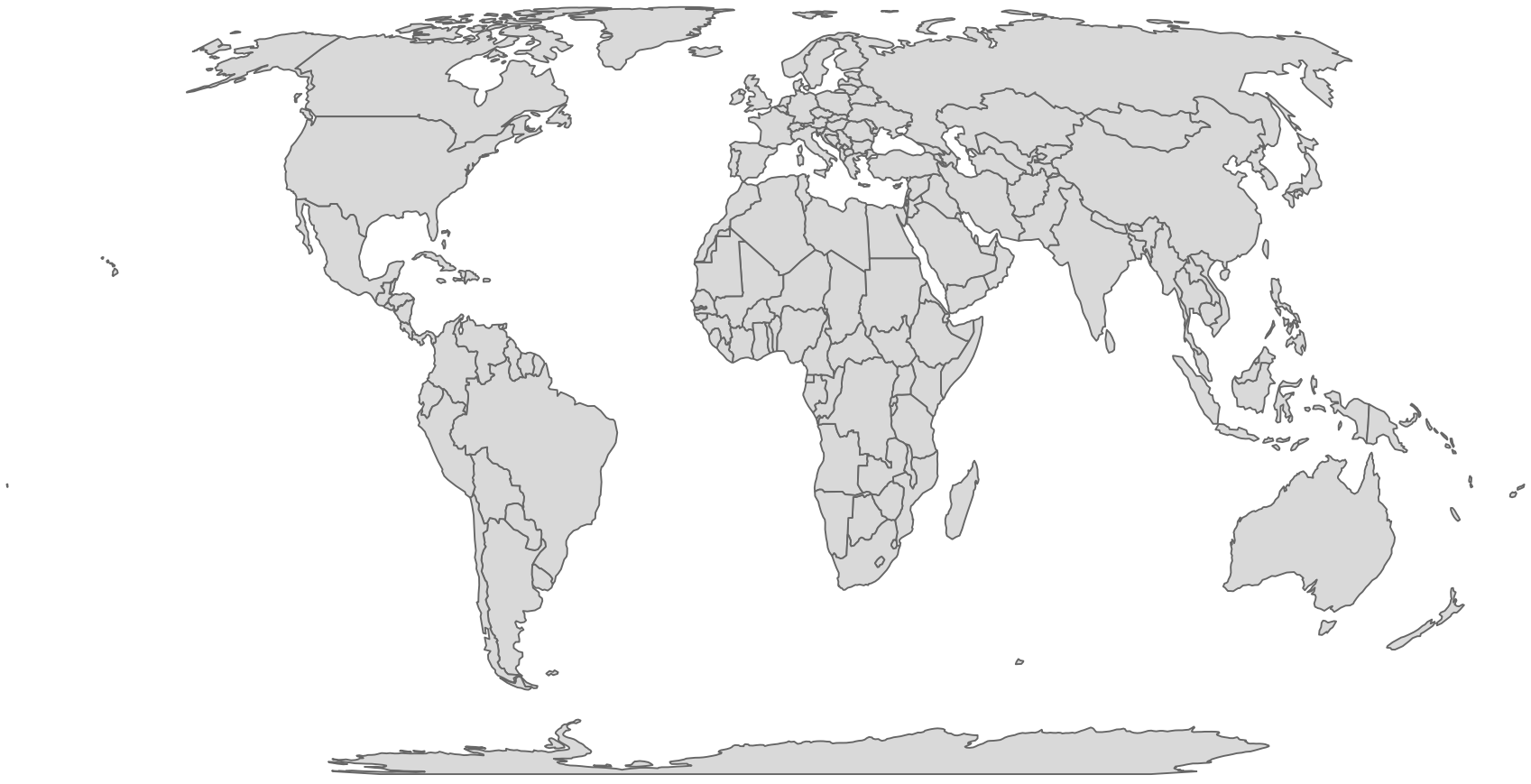
`tm_polygons()` is composed of two parts: `tm_borders()` and `tm_fill()`

```
tm_shape(World) +
  tm_borders(lty = 2) +
  tm_fill() +
  tm_layout(frame = FALSE)
```



Basics tmap

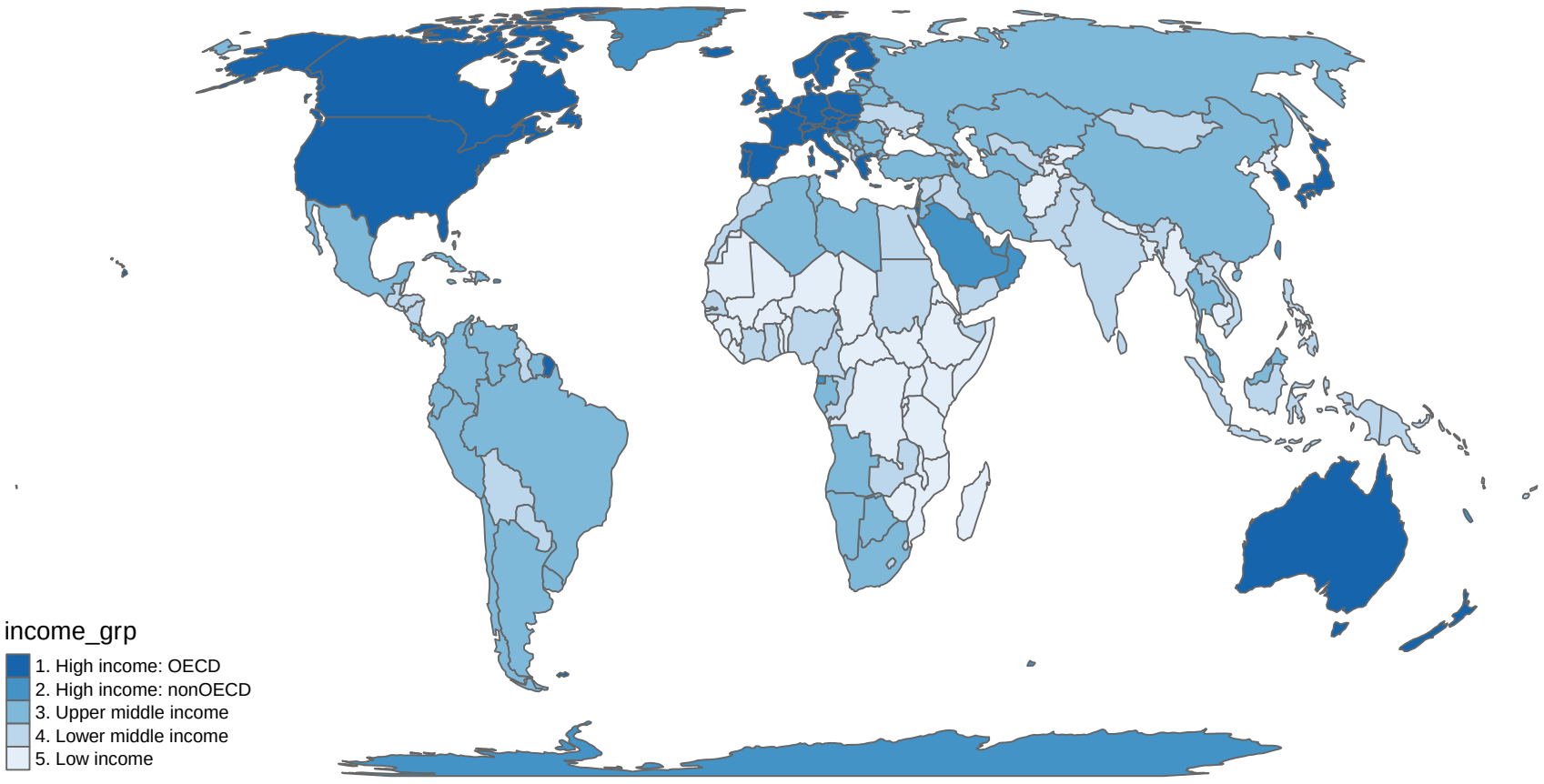
```
tm_shape(World) +
  tm_polygons() +
  tm_layout(frame = FALSE)
```



Basics tmap

All palettes from RColorBrewer are supported

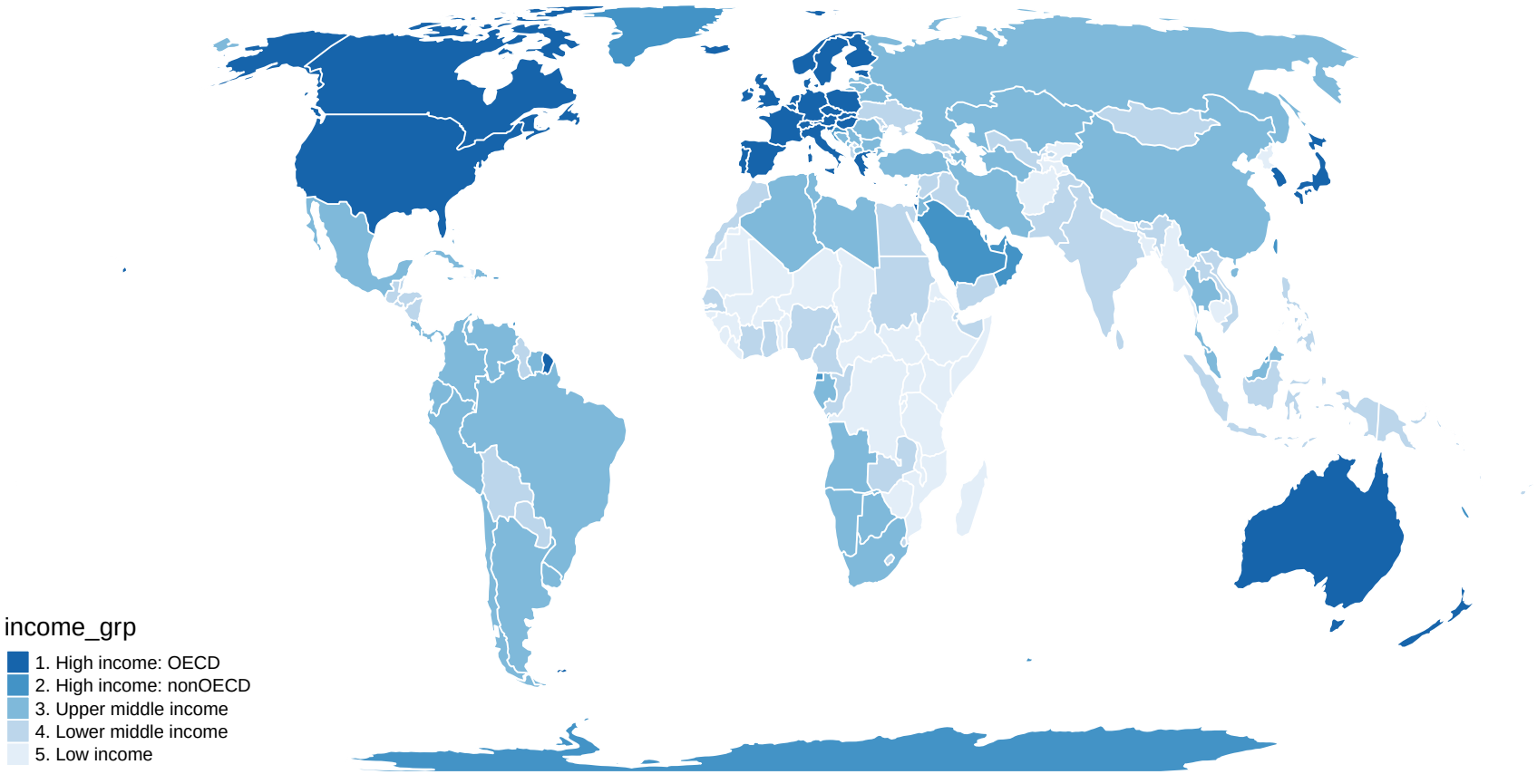
```
tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues") +
  tm_layout(frame = FALSE)
```



Basics tmap

Use white border to give it a "modern" look

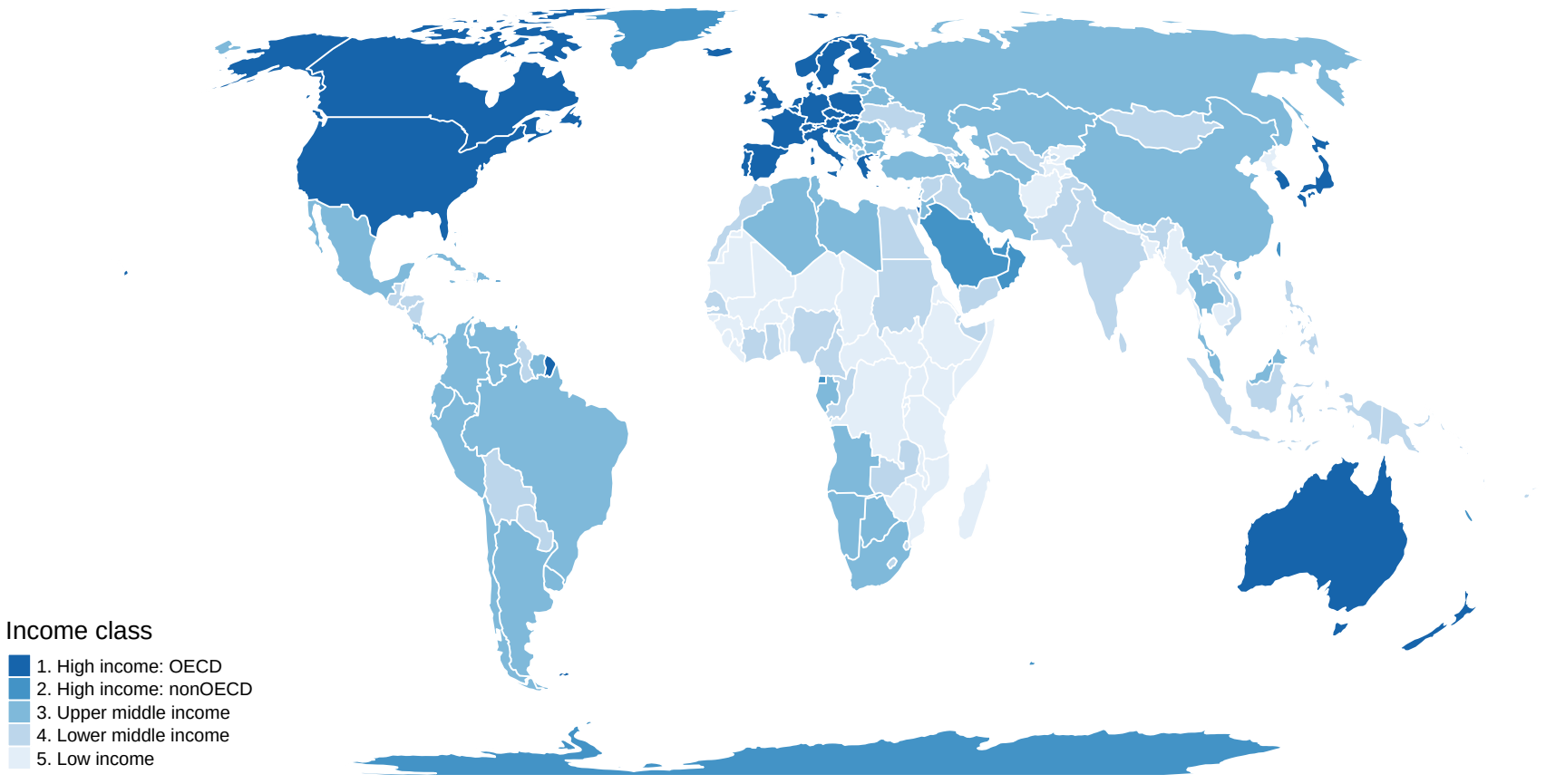
```
tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5) +
  tm_layout(frame = FALSE)
```



Basics tmap

Legend title:

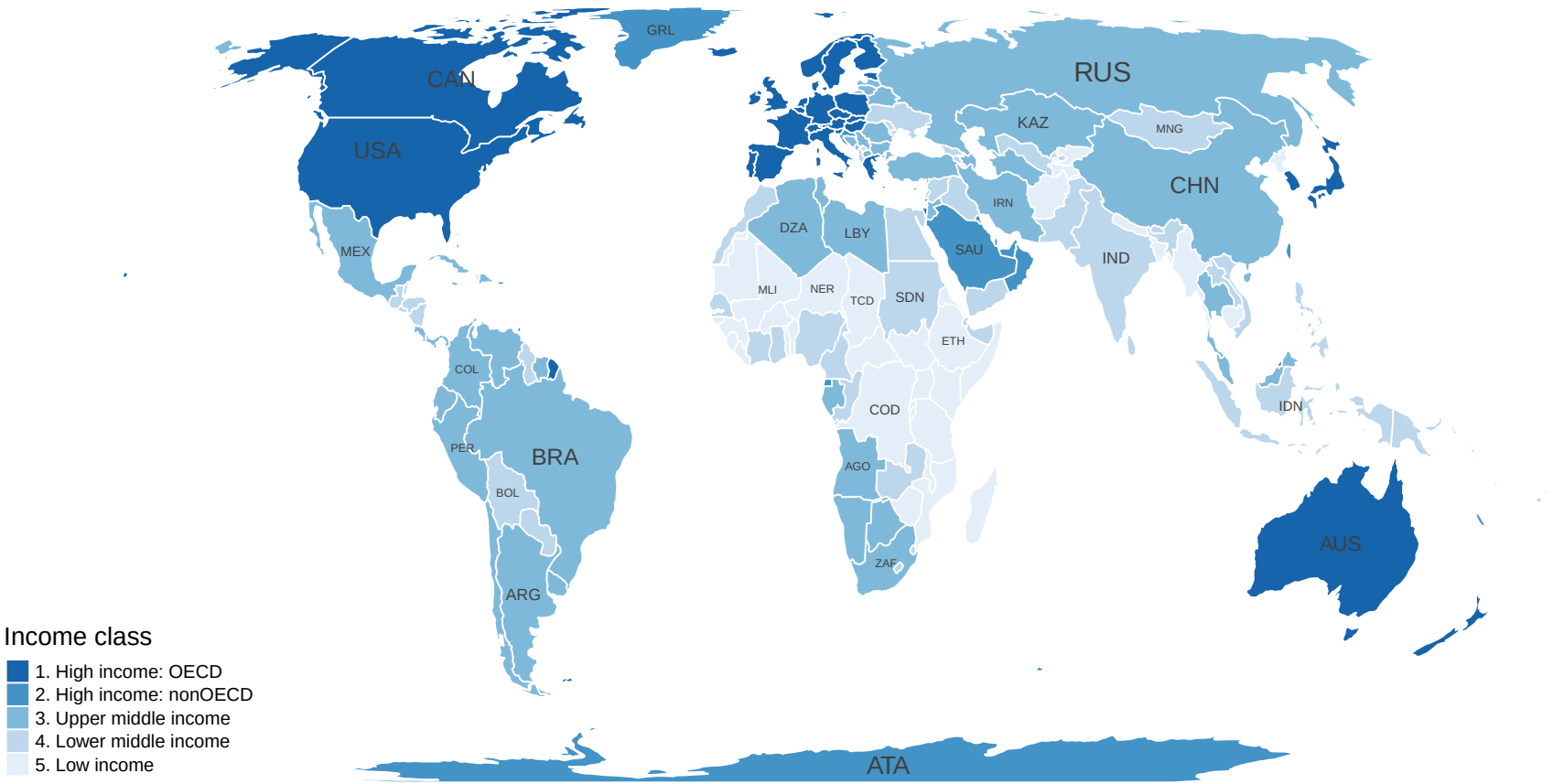
```
tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_layout(frame = FALSE)
```



Basics tmap

Add country labels as an additional layer:

```
tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_text(text = "iso_a3", size = "AREA", col = "grey25") +
  tm_layout(frame = FALSE)
```



Basics tmap

```
worldMap <-
  tm_shape(World) +
  tm_polygons(col = "income_grp", palette = "-Blues",
              border.col = "white", border.alpha = 0.5,
              title = "Income class") +
  tm_text(text = "iso_a3", size = "AREA", col = "grey25") +
  tm_layout(frame = FALSE)
```

Basics tmap

Let's add another dimension of information: cities' population

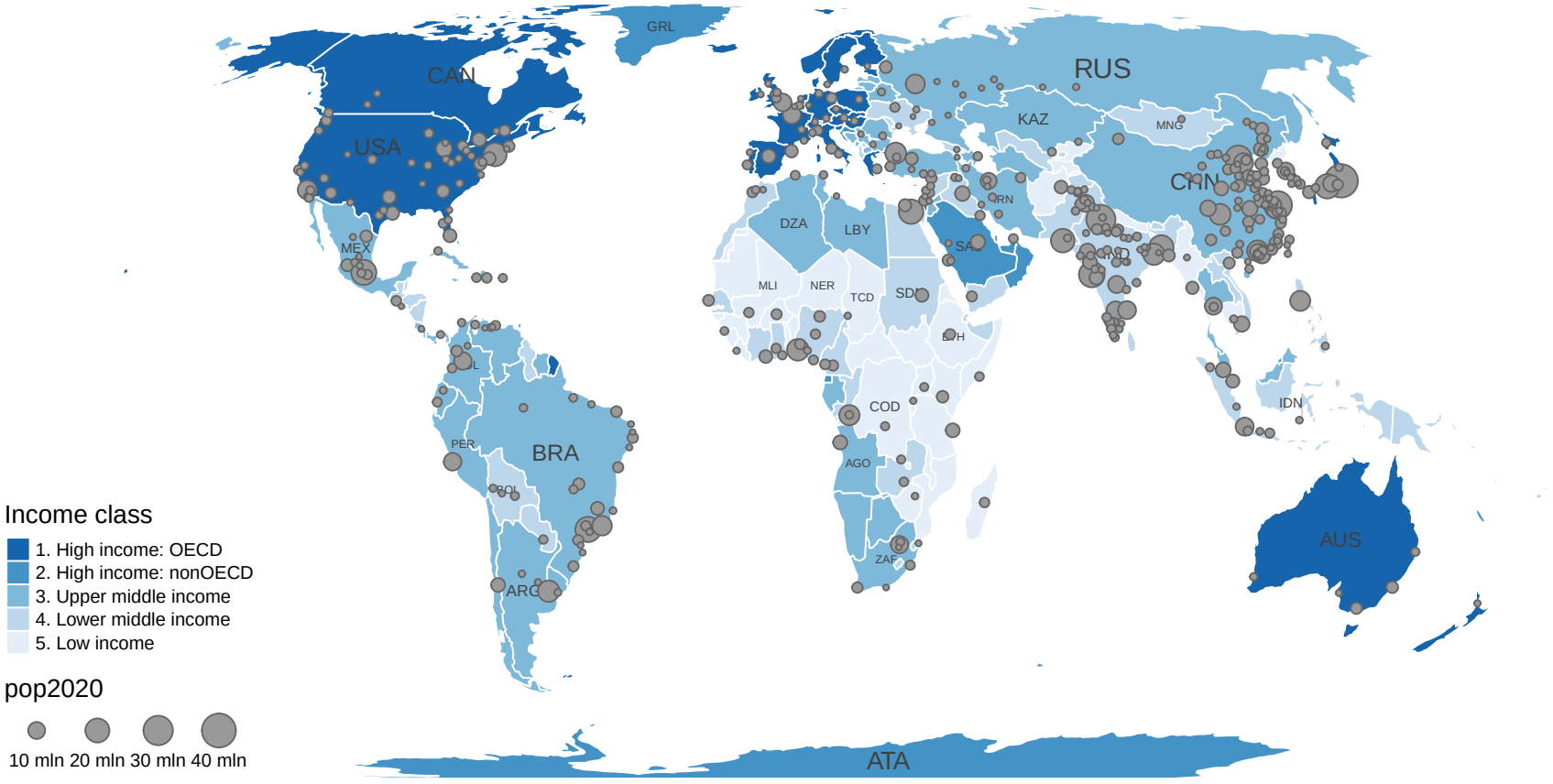
```
metro <- metro %>%
  mutate(growth = (pop2020 - pop2010) / pop2010 * 100)

print(metro[1, ])
```

```
## Simple feature collection with 1 feature and 13 fields
## geometry type:  POINT
## dimension:      XY
## bbox:           xmin: 69.17246 ymin: 34.52889 xmax: 69.17246 ymax: 34.52889
## geographic CRS: WGS 84
##    name name_long iso_a3 pop1950 pop1960 pop1970 pop1980
## 2 Kabul     Kabul    AFG  170784  285352  471891  977824
##   pop1990 pop2000 pop2010 pop2020 pop2030
## 2 1549320 2401109 3722320 5721697 8279607
##                     geometry   growth
## 2 POINT (69.17246 34.52889) 53.71319
```

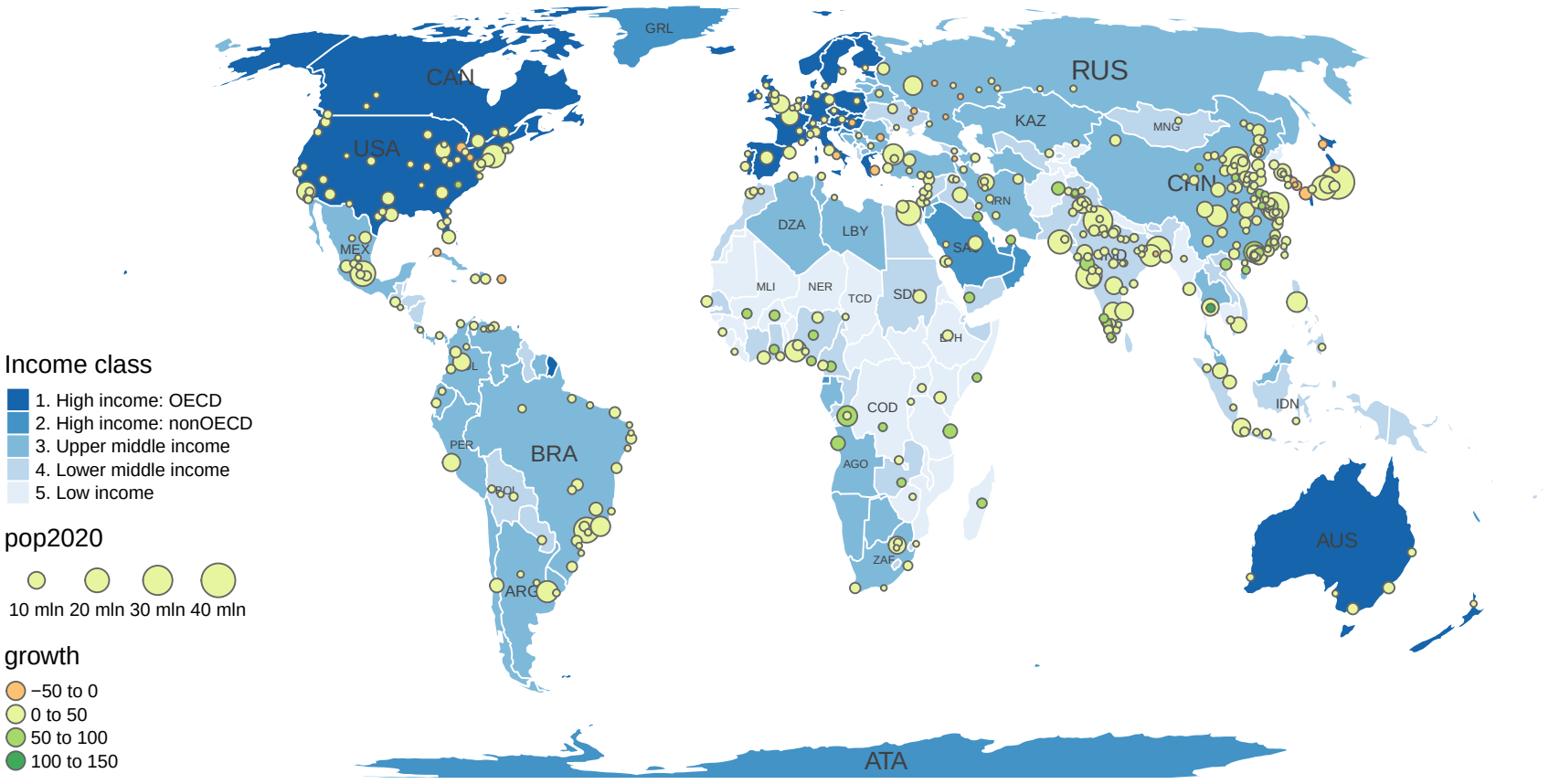
Basics tmap

```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020")
```



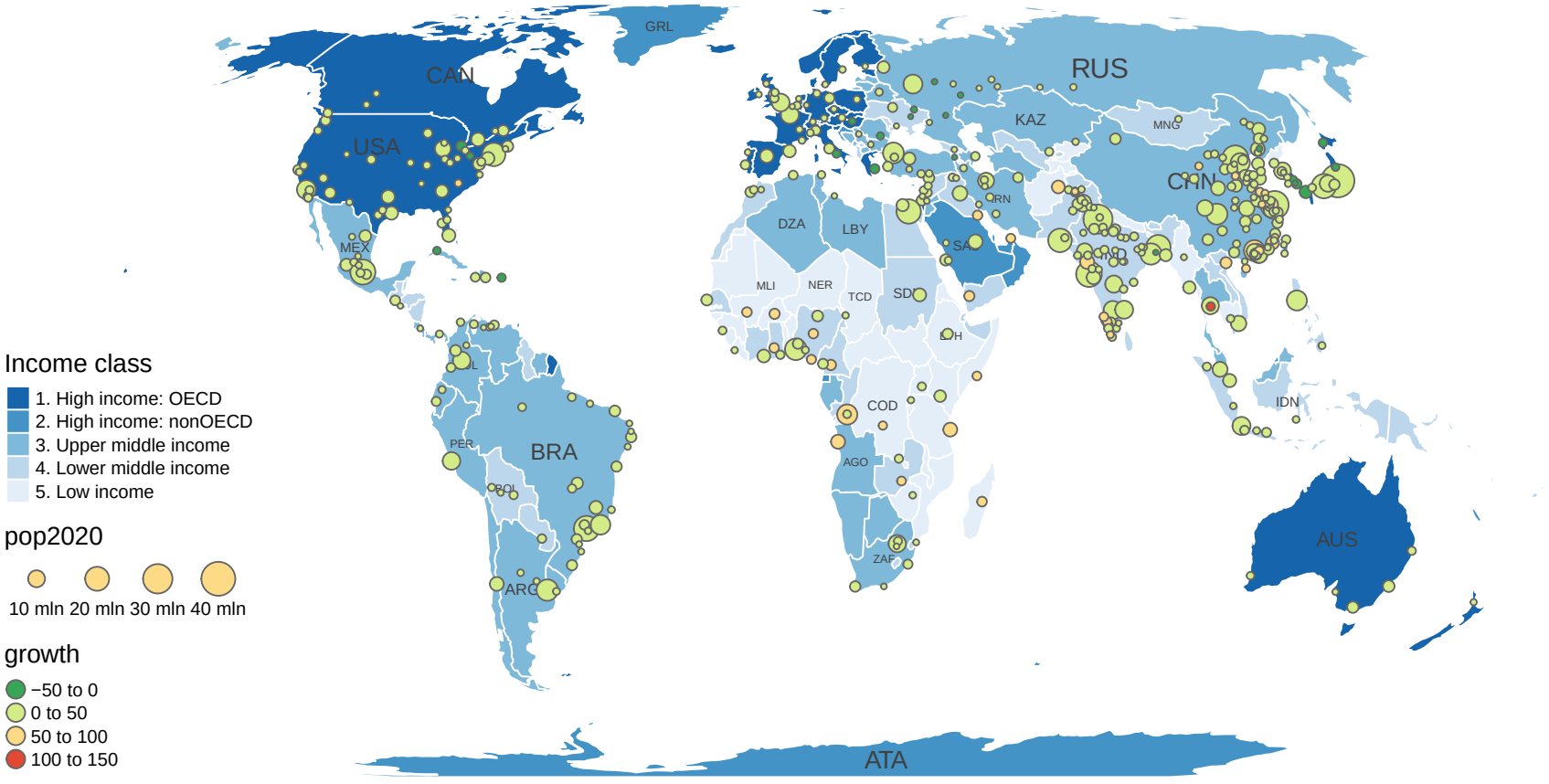
Basics tmap

```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth")
```



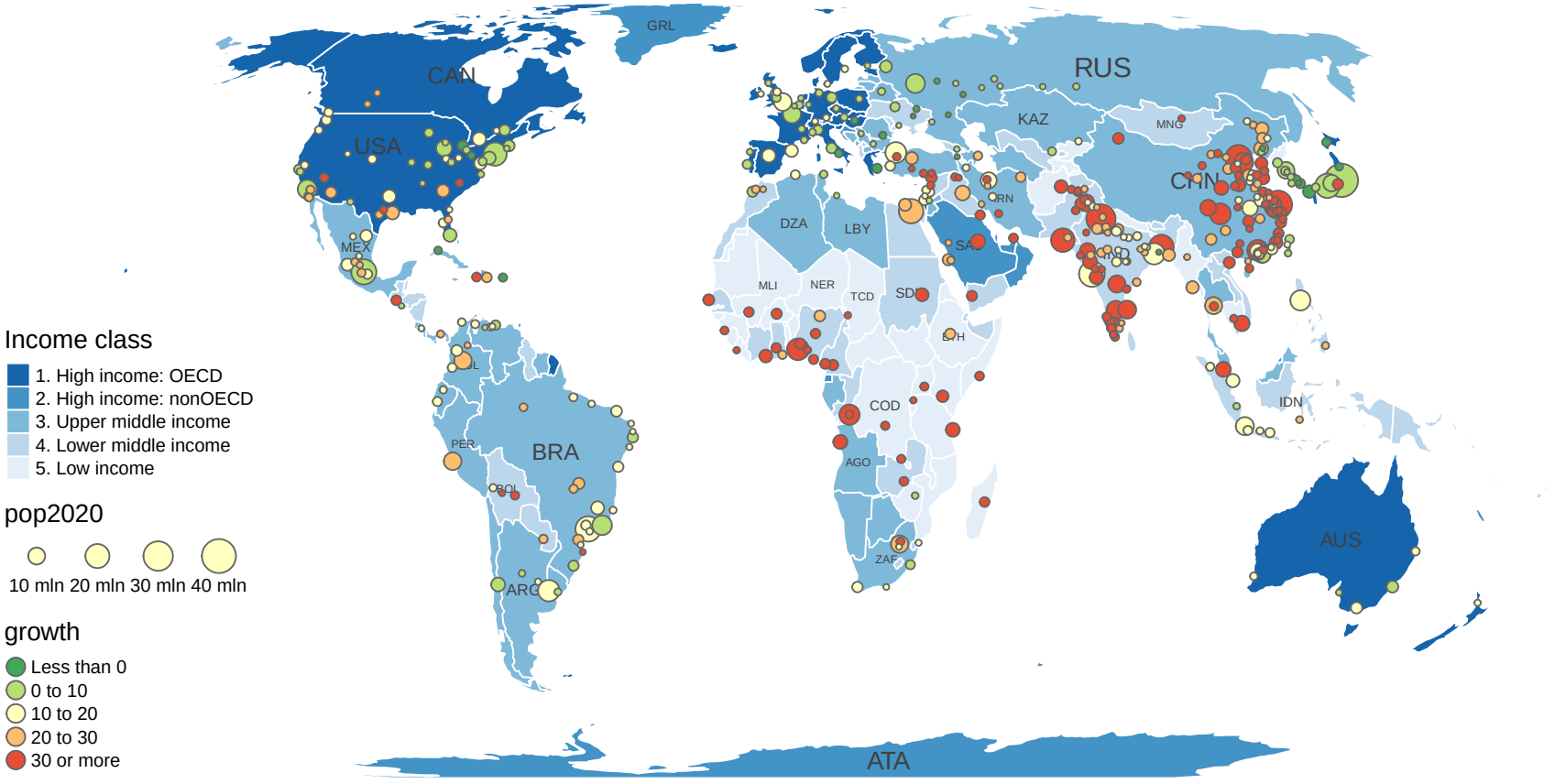
Basics tmap

```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
             palette = "-RdYlGn", midpoint = NA)
```



Basics tmap

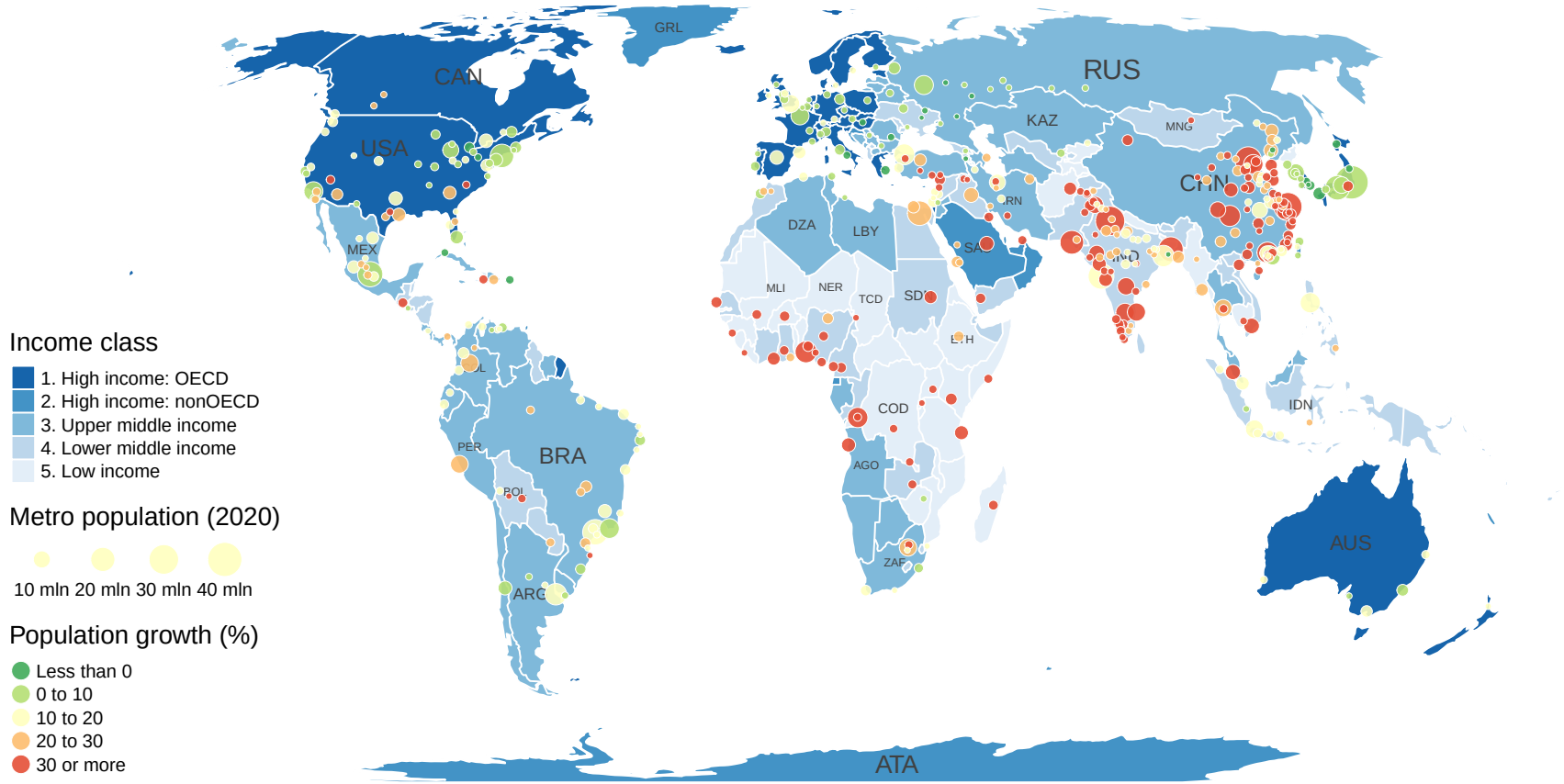
```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
             palette = "-RdYlGn", midpoint = NA,
             breaks = c(-Inf, 0, 10, 20, 30, Inf))
```



Basics tmap

```
worldMap +
  tm_shape(metro) +
  tm_bubbles(size = "pop2020", col = "growth",
             palette = "-RdYlGn", midpoint = NA,
             breaks = c(-Inf, 0, 10, 20, 30, Inf),
             alpha = 0.9,
             border.col = "white",
             border.lwd = 0.1,
             title.size = "Metro population (2020)",
             title.col = "Population growth (%)")
```

Basics tmap

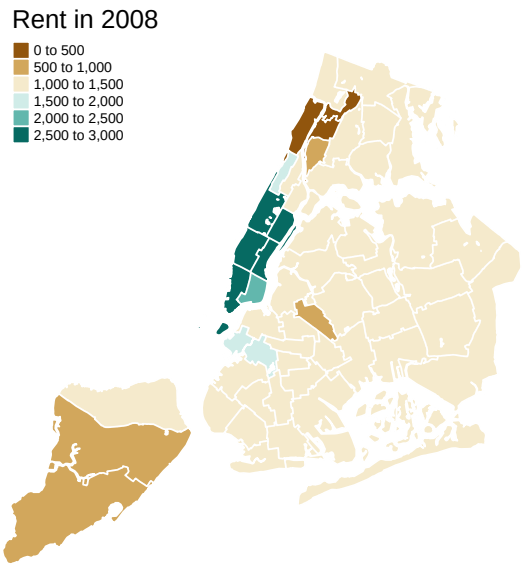

Income class
1. High income: OECD
2. High income: nonOECD
3. Upper middle income
4. Lower middle income
5. Low income
Metro population (2020)
10 mln 20 mln 30 mln 40 mln
Population growth (%)
Less than 0
0 to 10
10 to 20
20 to 30
30 or more
GRL
CAN
USA
MEX
RUS
KAZ
MNG
CHN
IRN
DZA
LBY
MLI
NER
TCD
SDN
COD
AGO
ZAF
IDN
AUS
BRA
PER
BOL
ARG
ATA

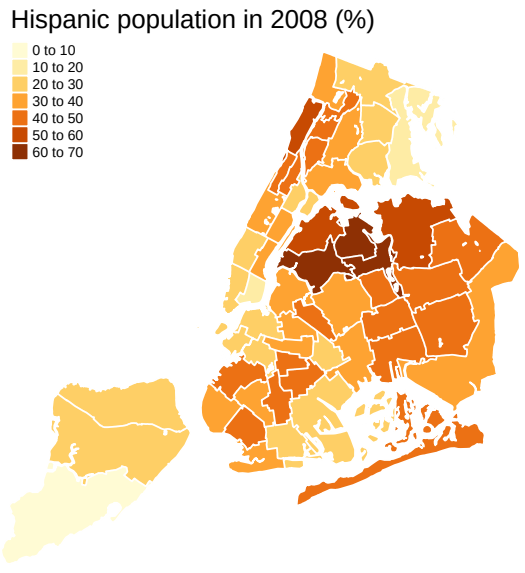
Basics tmap

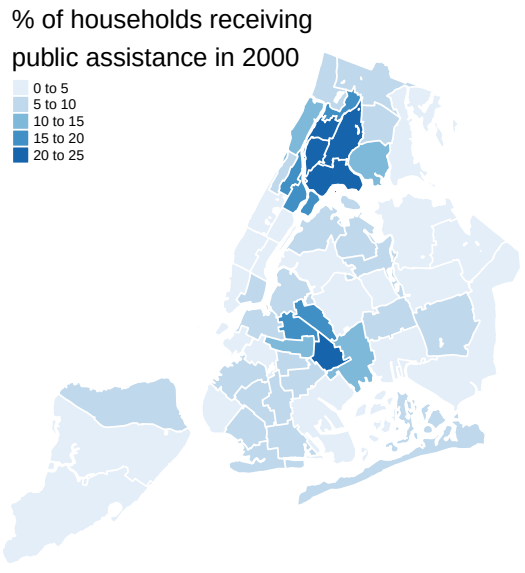
Save the resulting map using `tmap_save()`

```
tmap_save(world_map, filename = "worldMap.pdf")
```

Useful functions from `tmap`: New York example







Useful functions from `tmap`: New York example

- Prerequisite

Useful functions from `tmap`: New York example

- Prerequisite
  - Download the `.zip` data file here

Useful functions from `tmap`: New York example

- Prerequisite
  - Download the `.zip` data file here
  - Unzip it and put it in your working directory

Useful functions from tmap: New York example

- Load .shp file with sf

```
nyc.bound <- st_read("nyc/nyc.shp")
```

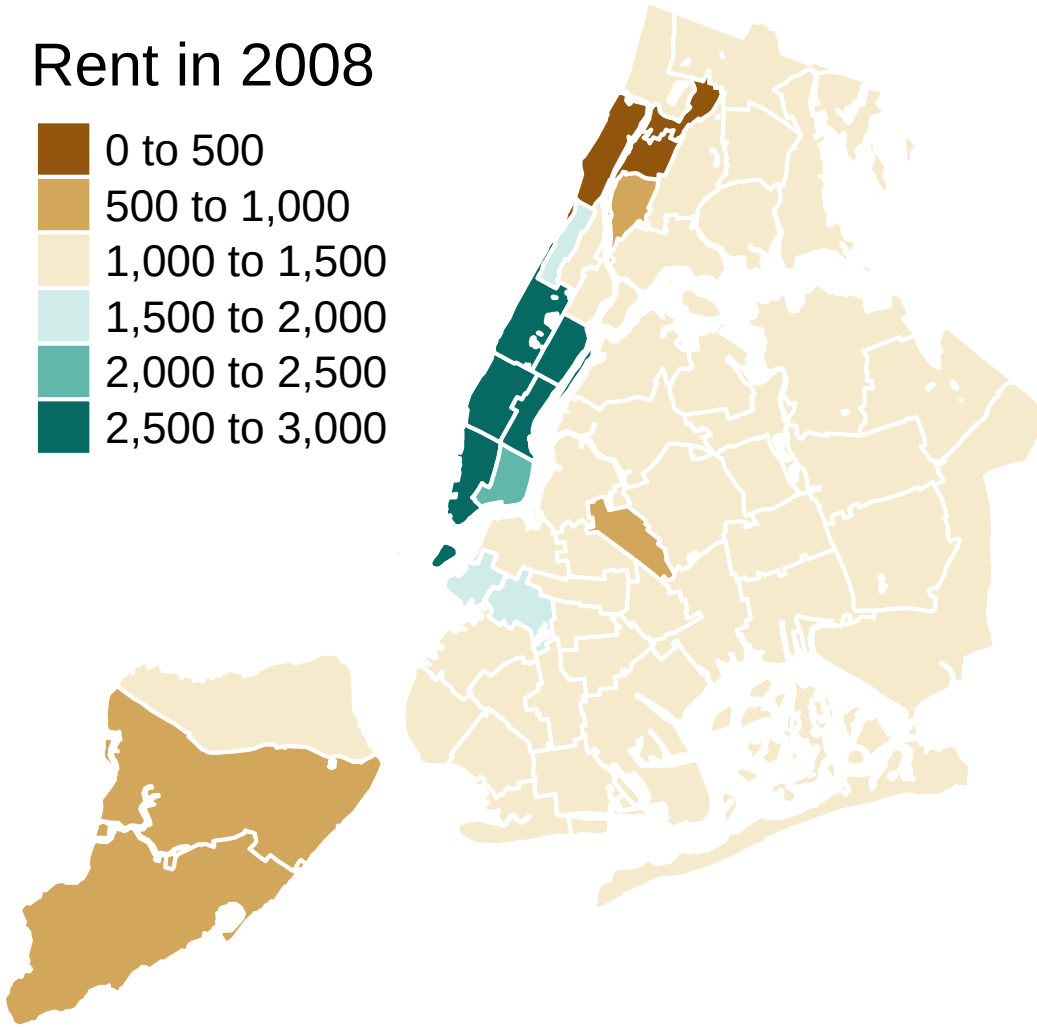
```
## Reading layer `nyc' from data source `/Users/brianleung/Desktop/21WI_CSSS569
## Simple feature collection with 55 features and 34 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 913037.2 ymin: 120117 xmax: 1067549 ymax: 272751.4
## projected CRS:  NAD83 / New York Long Island (ftUS)
```

New York example: overview

Variable	Description
rent2008	median monthly contract rent in 2008
forhis08	% of hispanic population in 2008
pubast00	% of households receiving public assistance in 2000

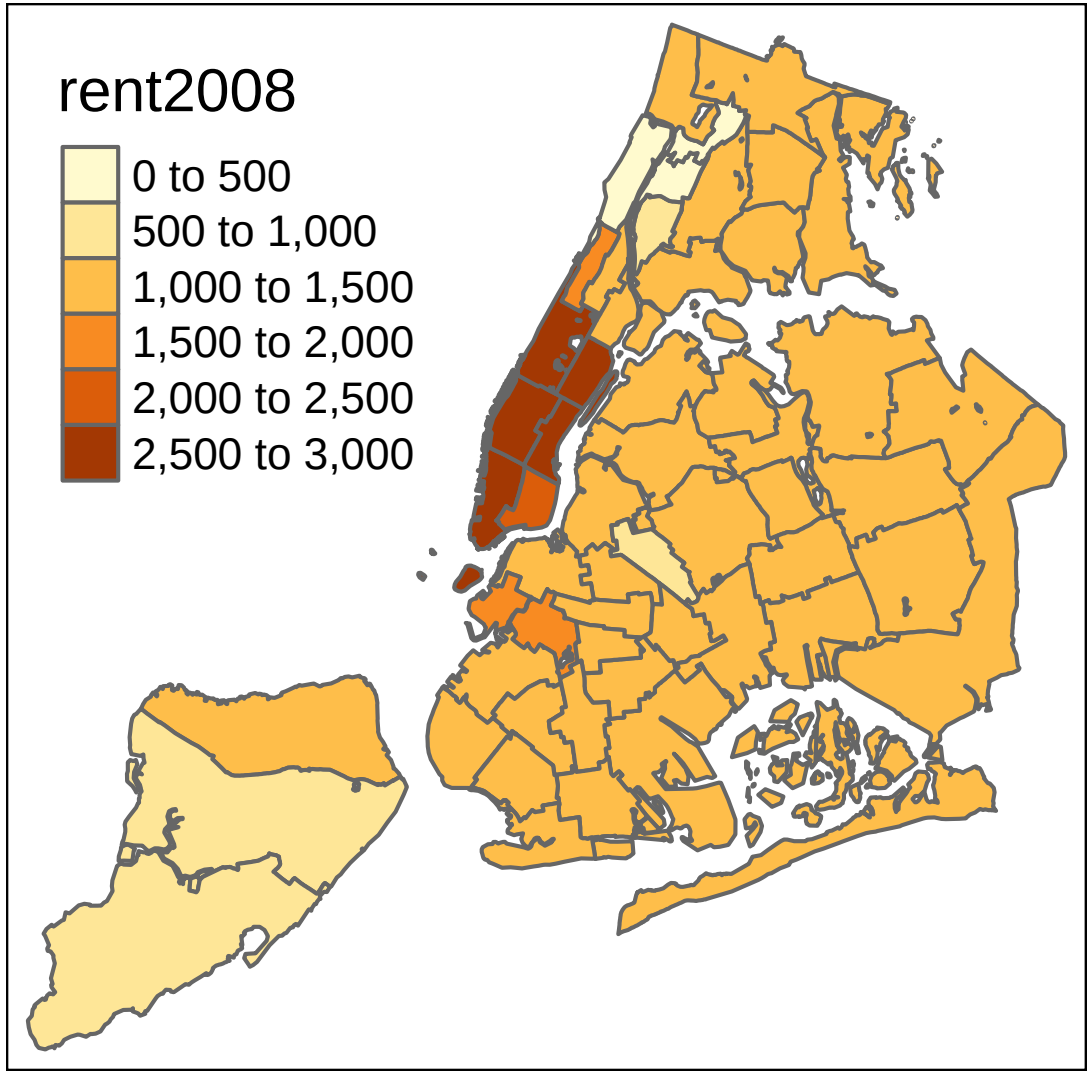
New York example: exercise 1

- Replicate the following map (or choose any palette you see fit)



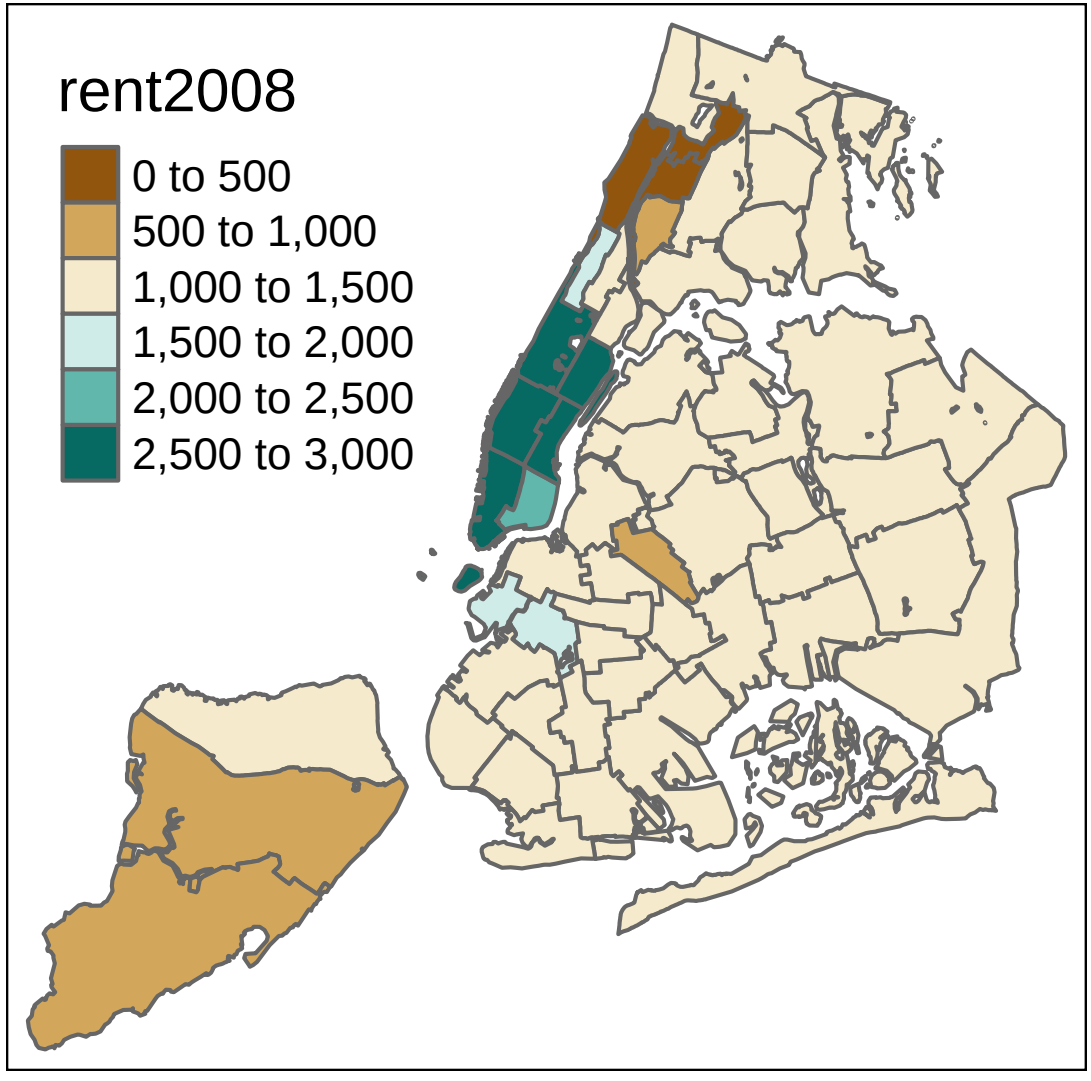
New York example: exercise 1

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008")
```



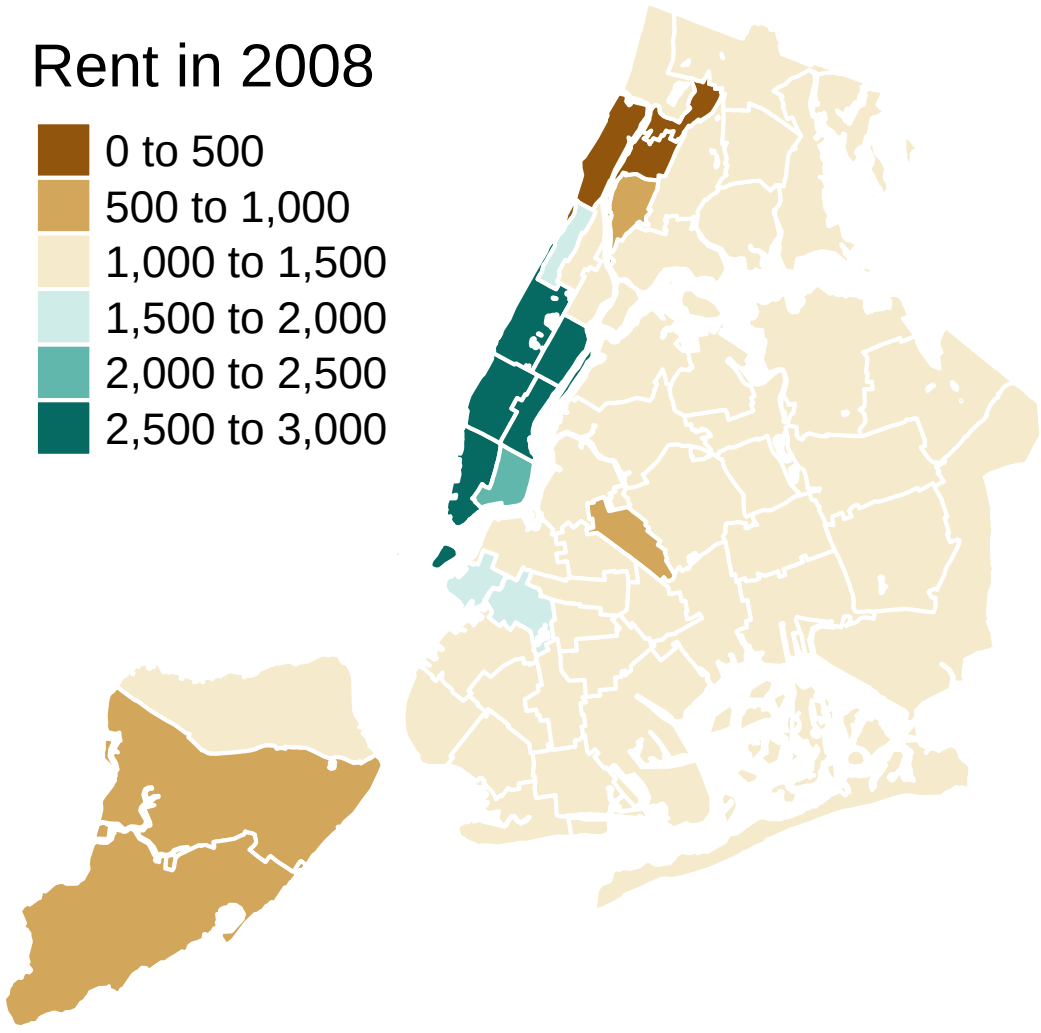
New York example: exercise 1

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG")
```



New York example: exercise 1

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              border.col = "white", title = "Rent in 2008") +
  tm_layout(frame = FALSE)
```



New York example: interactive mode

- Interactive map visualization

```
tmap_mode("view")
```

```
## tmap mode set to interactive viewing
```

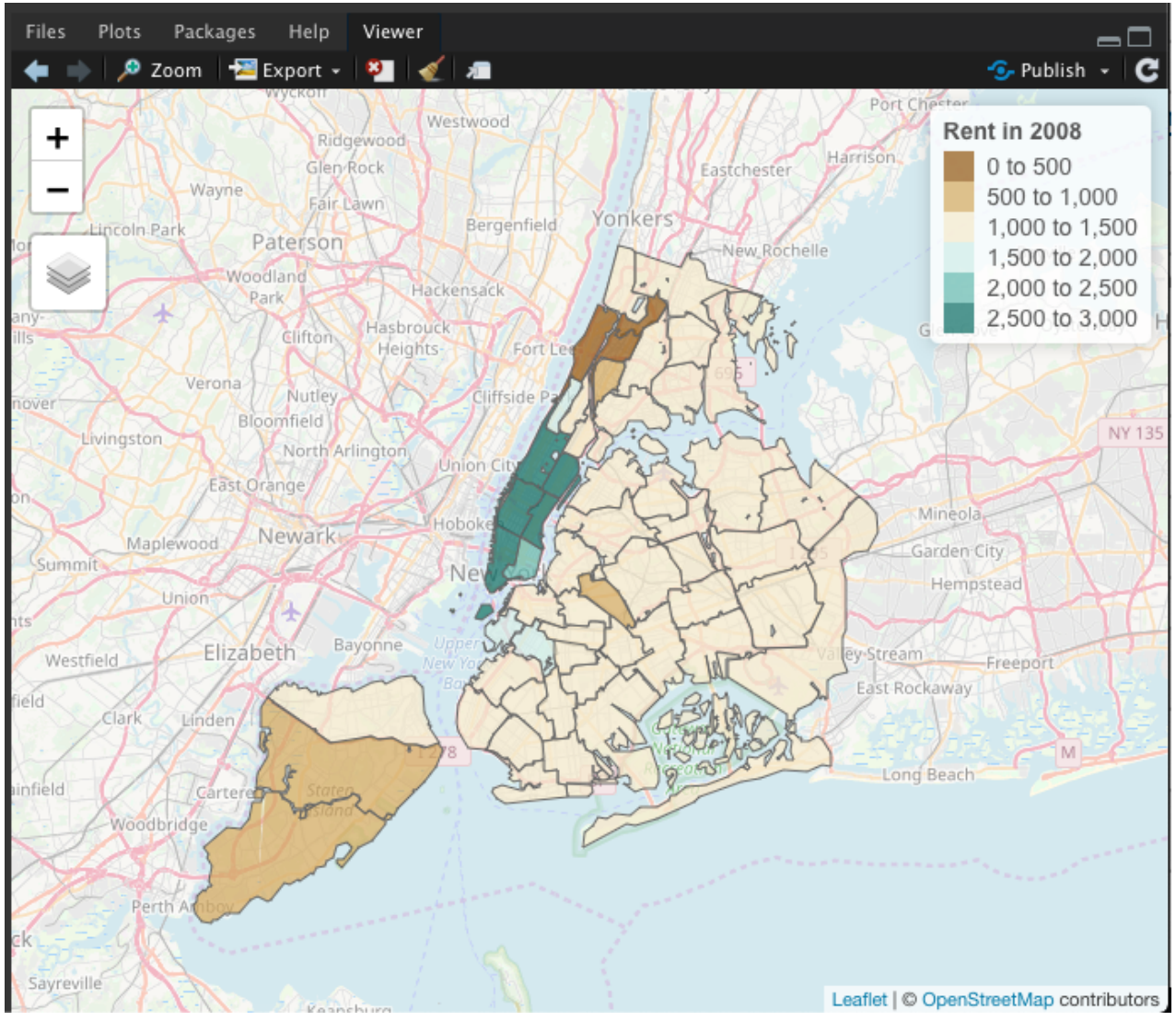
New York example: interactive mode

- Interactive map visualization

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              title = "Rent in 2008",
              alpha = 0.7) +
  tm_basemap(server = "OpenStreetMap", alpha = 0.5)
```

New York example: interactive mode

- Interactive map visualization



New York example: interactive mode

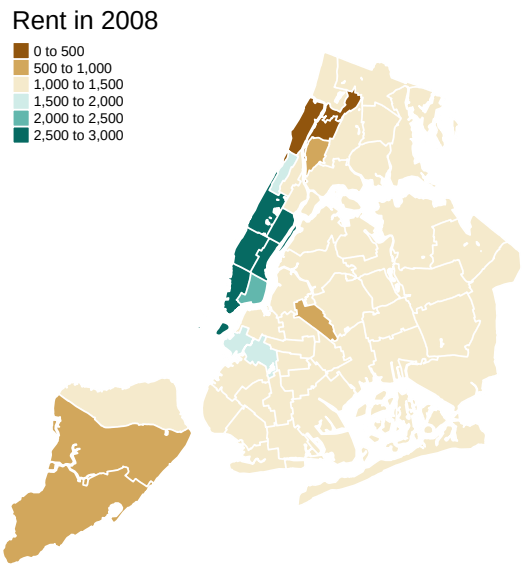
- Switching back to plotting mode

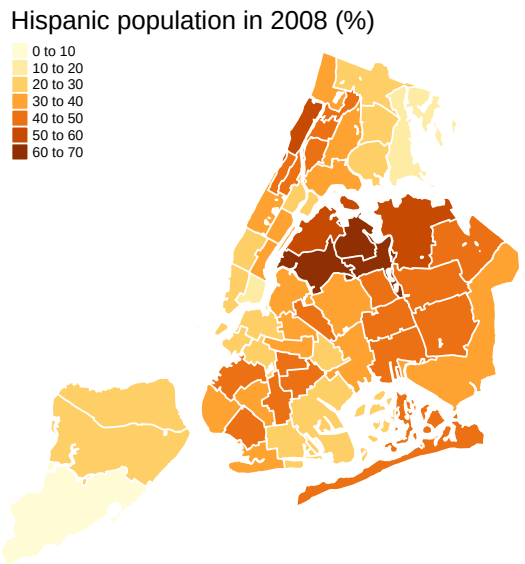
```
tmap_mode("plot")
```

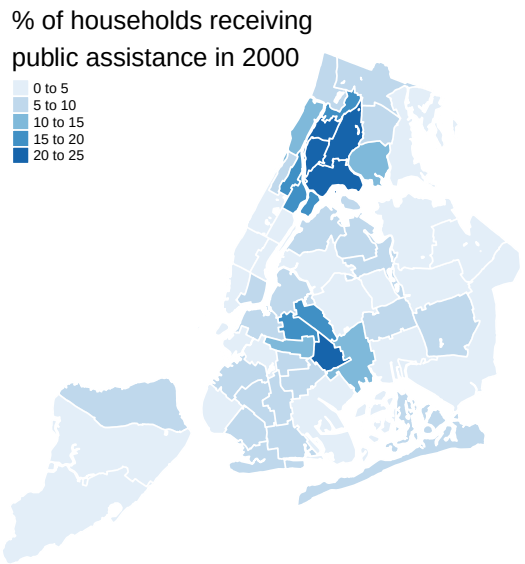
```
## tmap mode set to plotting
```

New York example: exercise 2

- Create two more maps based on `forhis08` and `pubast00`:







New York example: exercise 2

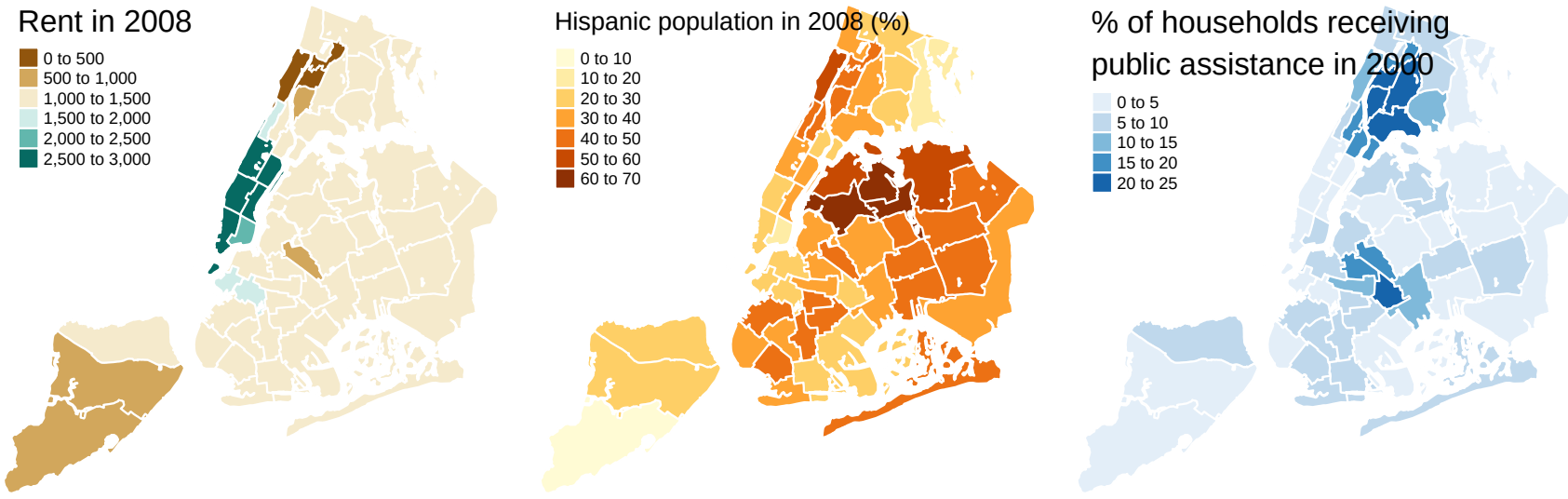
```
rentNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              border.col = "white", border.alpha = 0.5,
              title = "Rent in 2008") +
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)

hisNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "forhis08",
              border.col = "white", border.alpha = 0.5,
              title = "Hispanic population in 2008 (%)") +
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)

pubastNYC <- tm_shape(nyc.bound) +
  tm_polygons(col = "pubast00", palette = "Blues",
              border.col = "white", border.alpha = 0.5,
              title = "% of households receiving \npublic assistance in 2000")
  tm_layout(legend.text.size = 0.5,
            legend.width = 0.7,
            frame = FALSE)
```

New York example: exercise 2

```
tmap_arrange(rentNYC, hisNYC, pubastNYC, nrow = 1)
```



New York example: small multiples

- First, create some cutpoints based on `forhis08`

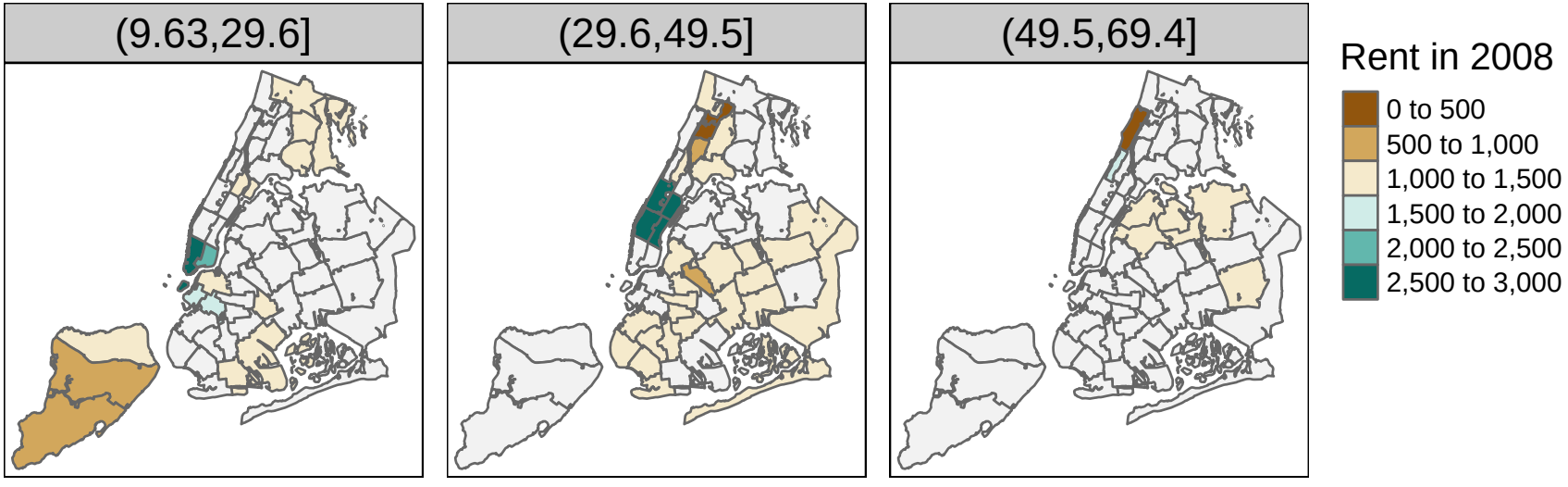
```
nyc.bound$cut.forhis <- cut(nyc.bound$forhis08,
                            breaks = 3)
print(nyc.bound$cut.forhis[1:10])
```

```
##  [1] (9.63,29.6] (9.63,29.6] (9.63,29.6] (49.5,69.4]
##  [5] (49.5,69.4] (49.5,69.4] (49.5,69.4] (29.6,49.5]
##  [9] (29.6,49.5] (49.5,69.4]
## Levels: (9.63,29.6] (29.6,49.5] (49.5,69.4]
```

New York example: small multiples

- Small multiples using `tm_facets()`

```
tm_shape(nyc.bound) +
  tm_polygons(col = "rent2008", palette = "BrBG",
              title = "Rent in 2008") +
  tm_facets(by = "cut.forhis", nrow = 1,
            free.coords = FALSE,
            drop.units = FALSE)
```



Concluding remarks

- Many more cool functions in `tmap`

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps
- Check out

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps
- Check out
  - `tmap` vignette

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps
- Check out
  - `tmap` vignette
  - Basic Mapping: R Notes

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps
- Check out
  - `tmap` vignette
  - Basic Mapping: R Notes
  - Geocomputation with R: Ch. 8:: Making maps with R

Concluding remarks

- Many more cool functions in `tmap`
  - Animation with maps
- Check out
  - `tmap` vignette
  - Basic Mapping: R Notes
  - Geocomputation with R: Ch. 8:: Making maps with R
  - Creating beautiful demographic maps in R with the tidycensus and tmap packages