London Business School

Data Visualisation AM10 - Session 2
Geospatial Visualisations

Kostis Christodoulou
10 Nov 2021

London Business School

Contents

- Visualising geospatial data
  - ggplot + the simple features ***sf*** package
  - *An interlude: functional programming-* ***purrr:map()***
  - Geocoding – Opencage
  - Adding data on maps: Points and Lines
  - GIS in R using the simple features ***sf*** package

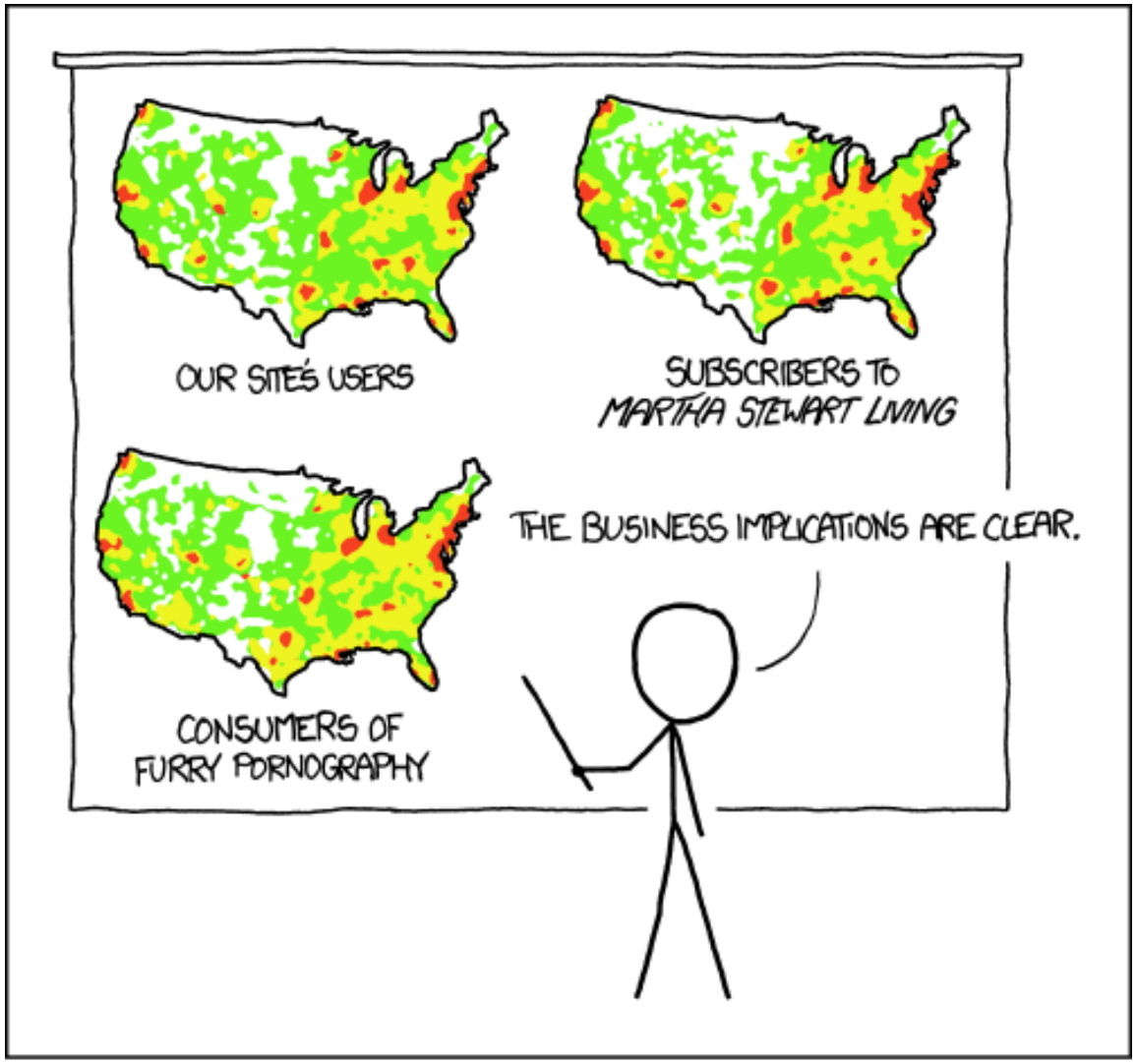
- Introduction to interactive graphs: Plotly and Shiny

Do I really need a map?

- Ask whether creating a spatial visualization adds value to your data.
- Beware of fake and/or junk maps.

https://www.fastcompany.com/90230916/the-next-great-fake-news-threat-bot-designed-maps
https://www.tasteatlas.com/europe/map

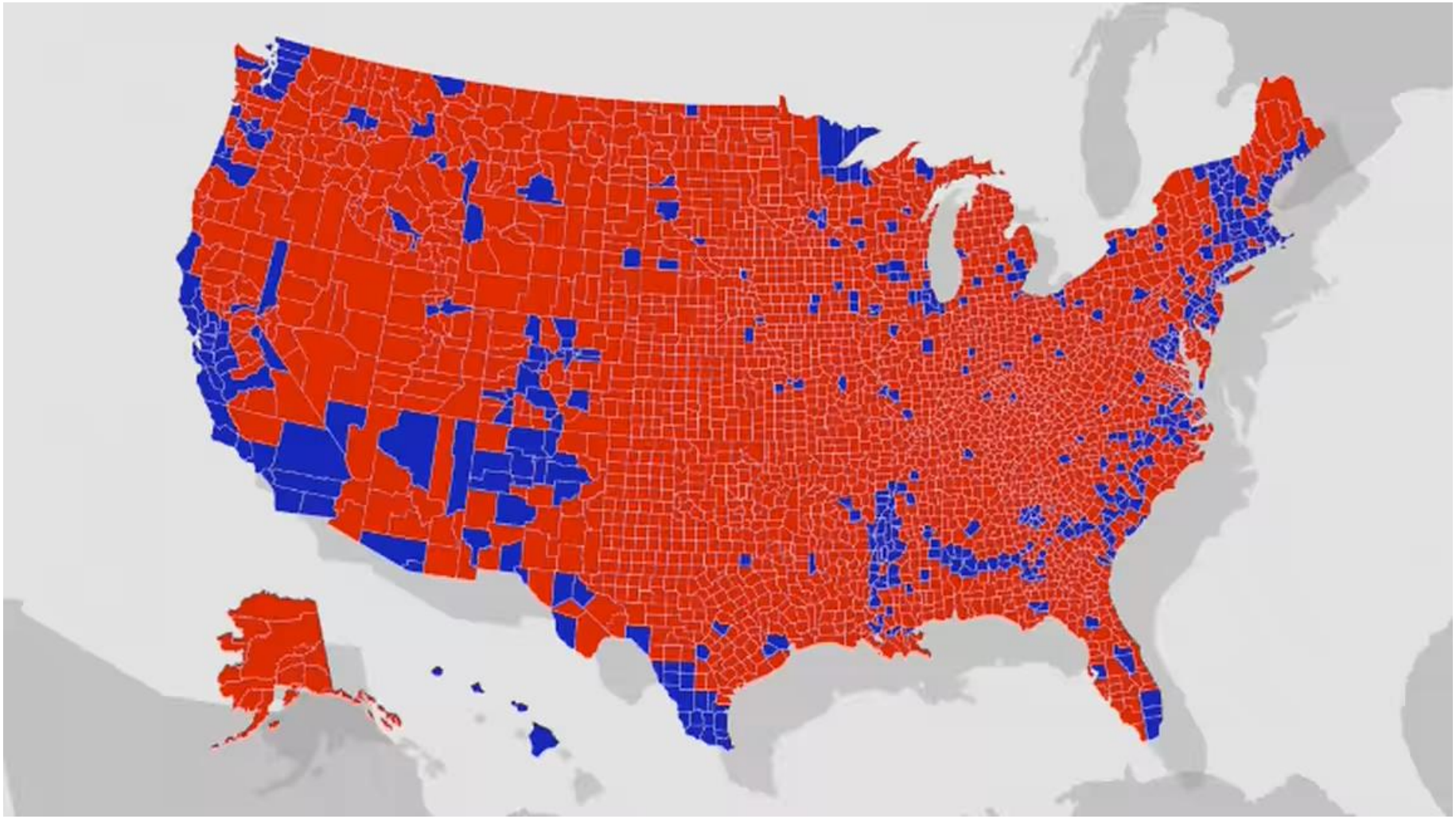
Is the map telling you something *new*?



https://www.explainxkcd.com/wiki/index.php/1138:_Heatmap

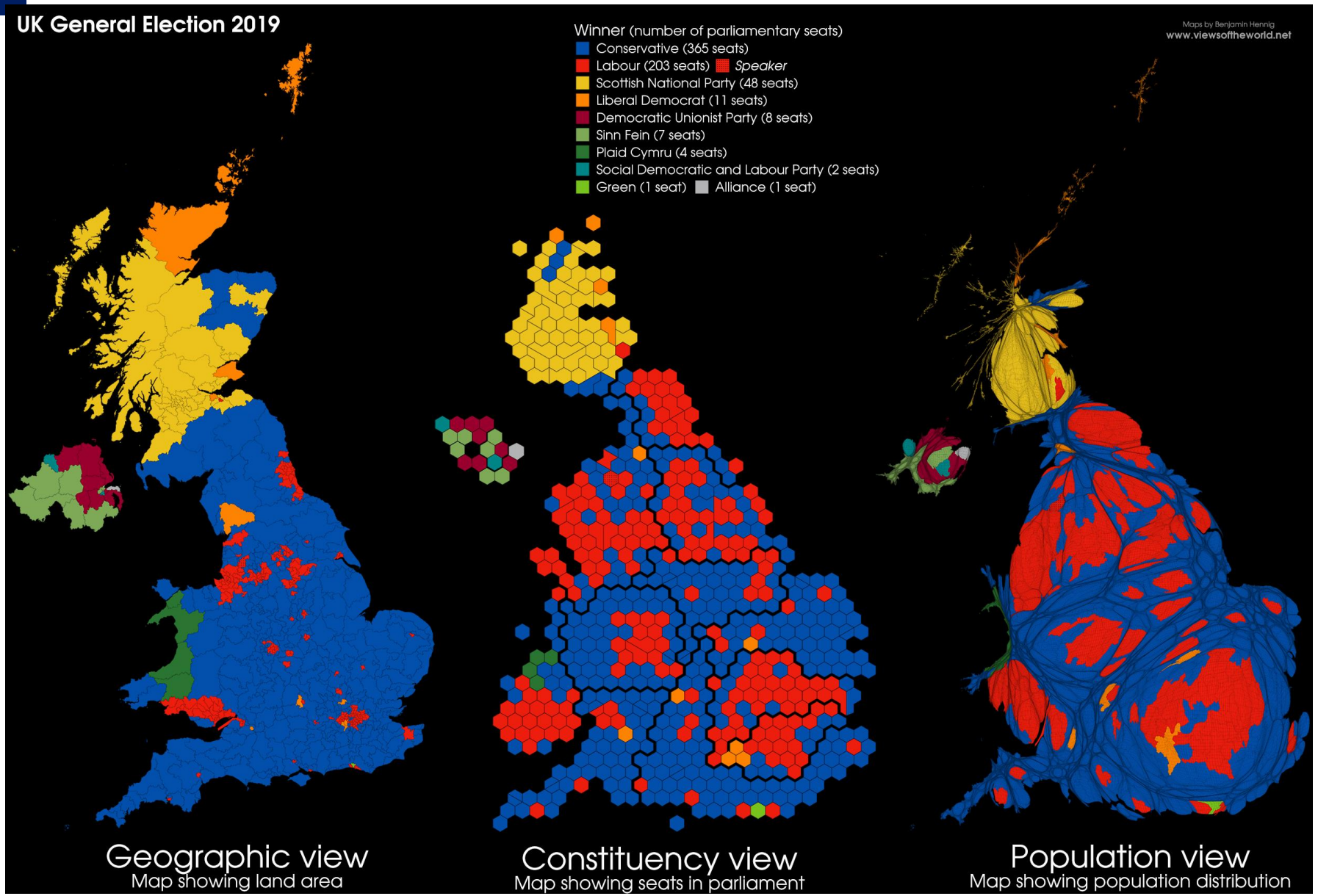
London Business School

Land does not vote in the US…



https://www.fastcompany.com/90572489/u-s-election-maps-are-wildly-misleading-so-this-designer-fixed-them
https://twitter.com/karim_doueib/status/1325208911054397441

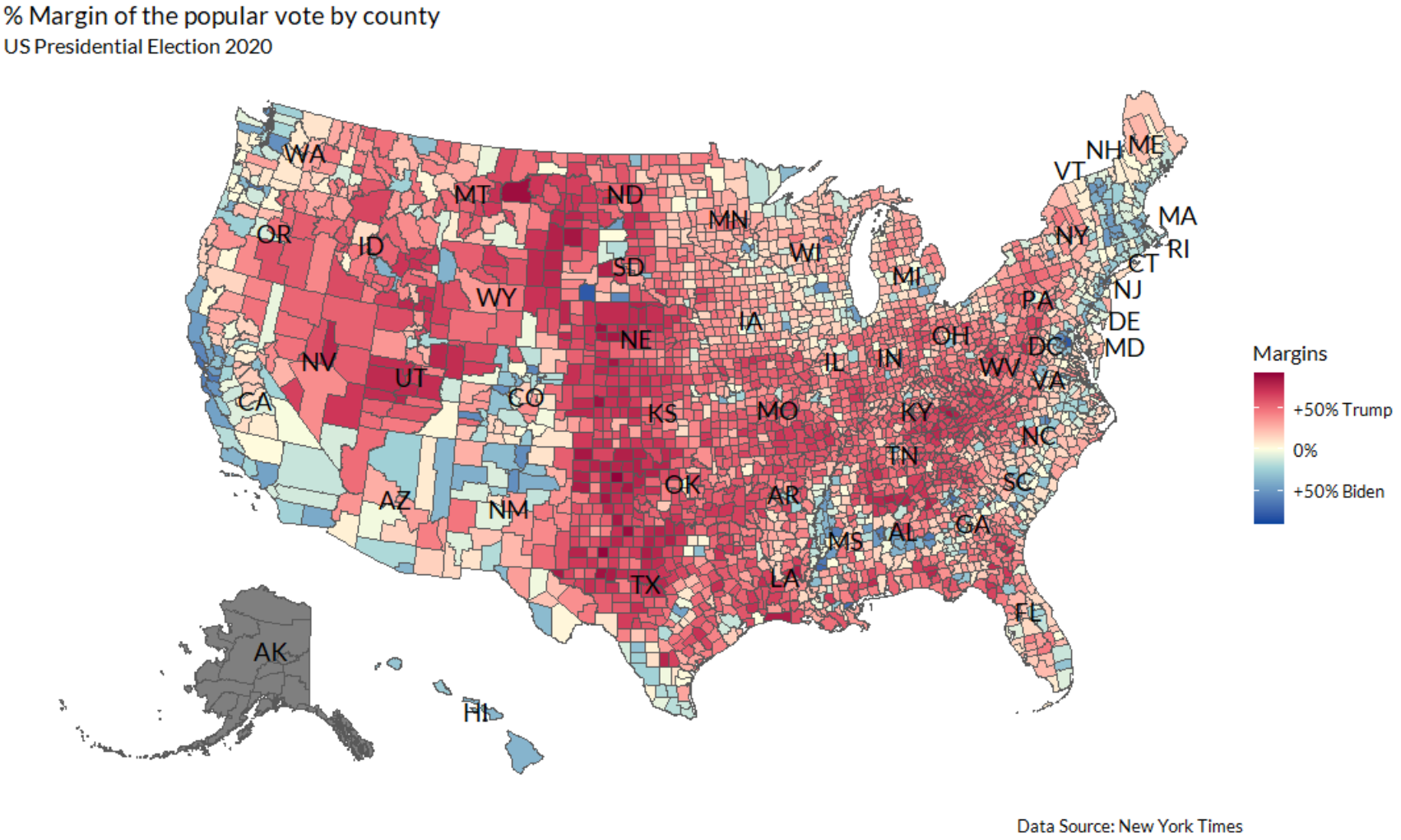
…nor does land vote in the UK



https://www.viewsoftheworld.net/?p=5755

London Business School

Challenge 1



Visualise election results w.r.t. county GDP

brookings.edu/blog/the-avenue/2020/11/09/biden-voting-counties-equal-70-of-americas-economy-what-d...

Brookings community and the public on our response to the coronavirus (COVID-19) »
Brookings scholars about the global response to coronavirus (COVID-19) »

oting counties equal 70% of America's economy. What does this mean for the s political-economic divide?

Table 1. Candidates' counties won and share of GDP in 2016 and 2020

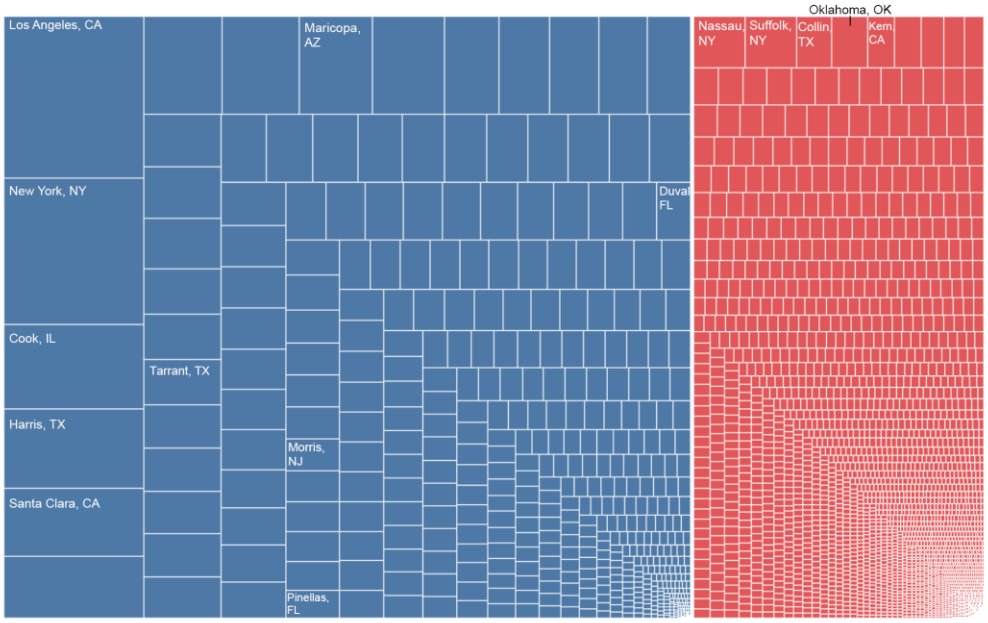
Year	Candidate	Counties won	Total votes	Aggregate share of US GDP
2016	Hillary Clinton	472	65,853,625	64%
	Donald Trump	2,584	62,985,106	36%
2020	Joe Biden	477	75,602,458	70%
	Donald Trump	2,497	71,216,709	29%

Note: 2020 figures reflect unofficial results from 96% of counties

Source: Brookings analysis of data from the Bureau of Economic Analysis, Dave Leip's Atlas of U.S. Presidential Elections, The New York Times, and Moody's Analytics

Figure 1. Candidate's share of 2018 Gross Domestic Product (GDP) by county in the 2020 presidential election

The less-than-500 counties won by **Joe Biden** generated **70 percent** of America's GDP in 2018
The more-than-2400 counties won by **Donald Trump** generated **29 percent** of America's GDP in 2018



Source: Brookings analysis of data from BEA, Dave Liep's Election Atlas, and The New York Times.

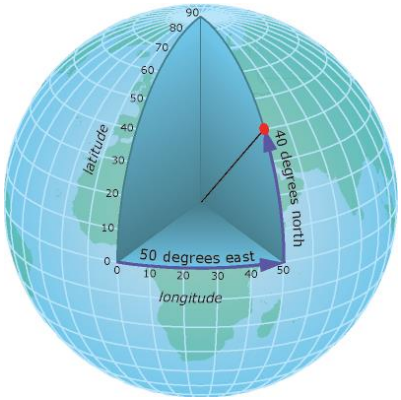
Metropolitan Policy Program at BROOKINGS

- https://www.brookings.edu/blog/the-avenue/2020/11/09/biden-voting-counties-equal-70-of-americas-economy-what-does-this-mean-for-the-nations-political-economic-divide/
- https://www.bea.gov/data/gdp/gdp-county-metro-and-other-areas

London Business School

What is spatial data?

- Data is associated with locations that are described by coordinates & a coordinate reference system (CRS)
- Common CRS: (latitude, longitude, altitude) describes locations on the surface of the Earth.
- *Latitude first, Longitude next.* International Maritime Organization standard
- Latitude, longitude, and altitude are specified relative to a reference system called the datum. One widely used datum is the *World Geodetic System (WGS) 84*, which is used by the Global Positioning System (GPS).



	type	date	part_of_a_policing_operation	policing_operation	lat	lng
1	Person search	2019-08-31 23:00:00	FALSE	*NA*	51.48535	-0.177498
2	Person search	2019-08-31 23:00:00	FALSE	*NA*	51.49349	-0.048342
3	Person search	2019-08-31 23:04:00	FALSE	*NA*	51.50088	-0.230044
4	Person search	2019-08-31 23:05:00	FALSE	*NA*	51.37918	-0.103201

https://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/geographic-coordinate-system.htm

London Business School

Coordinate reference systems

- A place on earth is specified by a latitude and longitude or X/Y coordinates
- Coordinates are based on a mathematical model of the shape of
- the earth; Some mathematical formulas involve a transformation from the 3D globe to a 2D map

- An **unprojected CRS** uses latitude and longitude coordinates and references the earth as a three-dimensional object
- A **projected CRS** uses X and Y coordinates as a two-dimensional representation of the earth
- https://observablehq.com/@d3/projection-transitions

- Spatial data files, typically **shapefiles**, have a CRS and the file's metadata usually tells you the CRS used
- Different shapefiles may have different CRS– you can transform from one CRS to another and for our purposes, we will use EPSG:4326, or the World Geodetic System 1984 used in GPS, defined by a latitude/longitude pair

http://www.epsg.org/, https://epsg.io/

Different coordinate reference systems

International Association of Oil & Gas Producers

Regis

Geomatics home | EPSG home | EPSG Dataset

What is new?

Online Registry

The current version of the Online Registry which includes all recent updates (version 9.8.3, 2019-10-11)

Download Dataset

Download Dataset (version 9.8.2, 2019-09-19)

Read about planned Upgrade of EPSG Dataset data model

About the EPSG Dataset

The IOGP's EPSG Geodetic Parameter Dataset is a collection of definitions of coordinate reference systems and coordinate transformations which may be global, regional, national or local in application. The primary EPSG Dataset is maintained in the online registry, from which data may be accessed through a graphic user interface or through a service interface. The online registry contains the most current data. Registry users may query and view the data, generate printable reports and create Well-known Text (WKT)

http://www.epsg.org/, https://epsg.io/

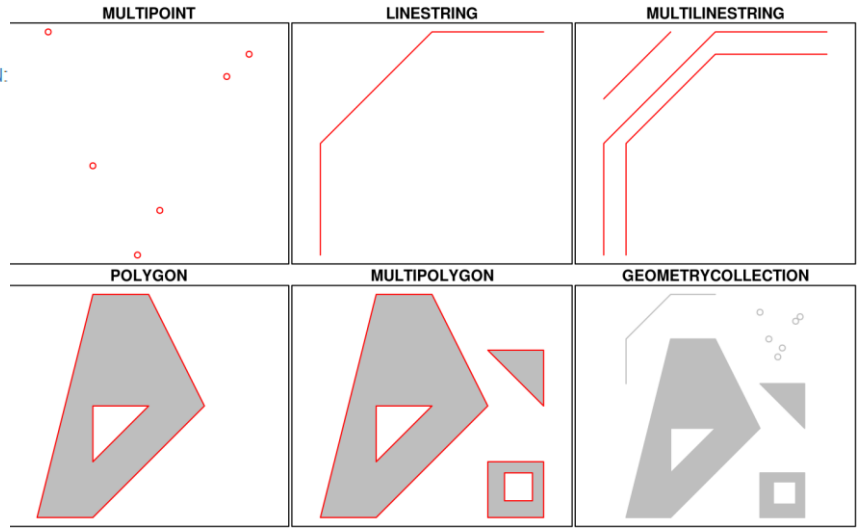
Simple features and the *sf* package

Simple features refers to how objects in the real world can be represented in computers, with emphasis on the spatial geometry of these objects.

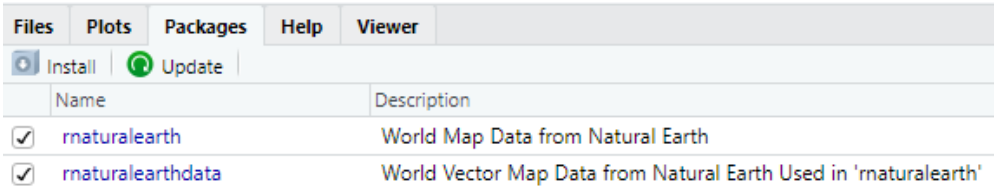
Simple feature geometry types

The following seven simple feature types are the most common, and are for instance the only ones used for GeoJSON:

type	description
`POINT`	zero-dimensional geometry containing a single point
`LINESTRING`	sequence of points connected by straight, non-self intersecting line pieces; one-dimensional geometry
`POLYGON`	geometry with a positive area (two-dimensional); sequence of points form a closed, non-self intersecting ring; the first ring denotes the exterior ring, zero or more subsequent rings denote holes in this exterior ring
`MULTIPOINT`	set of points; a MULTIPOINT is simple if no two Points in the MULTIPOINT are equal
`MULTILINESTRING`	set of linestrings
`MULTIPOLYGON`	set of polygons
`GEOMETRYCOLLECTION`	set of geometries of any type except GEOMETRYCOLLECTION



- Packages for data handling: ***sf*** for vectors and ***raster*** for grids; ***dplyr***
- Packages for plotting : ***ggplot2, tmap***
- Other useful packages: ***rnaturalearth***, ***rnaturalearthdata*** world vector data



https://cran.r-project.org/web/packages/sf/vignettes/sf1.html

London Business School

Simple features and the *sf* package

01_crs_projections.R

Vector spatial objects (***sf*** objects) are tidy data frames

Can use standard tools like *dplyr* to wrangle data

These data frames have some special properties:

- They include spatial metadata like the EPSG coordinate reference system (CRS)
- The **geometry** is stored in a *list column*

```
# get a medium resolution vector map of world countries excl. Antarctica
world <- ne_countries(scale = "medium", returnclass = "sf") %>%
  filter(name != "Antarctica")
```

```
> st_geometry(world) # what is the geometry of the world sf?
Geometry set for 240 features
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -180 ymin: -58.49229 xmax: 180 ymax: 83.59961
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
First 5 geometries:
MULTIPOLYGON (((-69.89912 12.452, -69.8957 12.4...
MULTIPOLYGON (((74.89131 37.23164, 74.84023 37....
MULTIPOLYGON (((14.19082 -5.875977, 14.39863 -5...
MULTIPOLYGON (((-63.00122 18.22178, -63.16001 1...
MULTIPOLYGON (((20.06396 42.54727, 20.10352 42....
```

Multipolygon for 240 countries

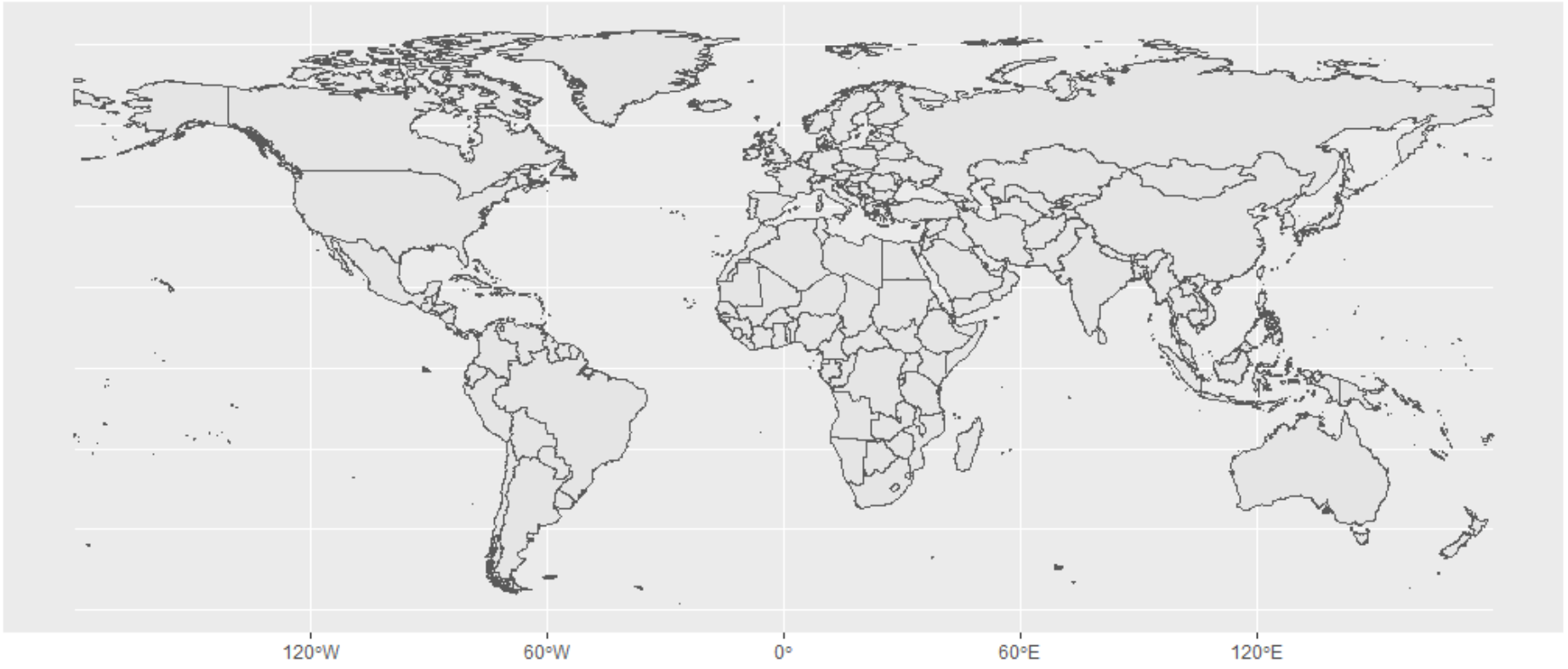
Bounding Box for vector map

CRS used: 4326, corresponding to WGS84, latitude/longitude

Plotting a world map with two lines of code

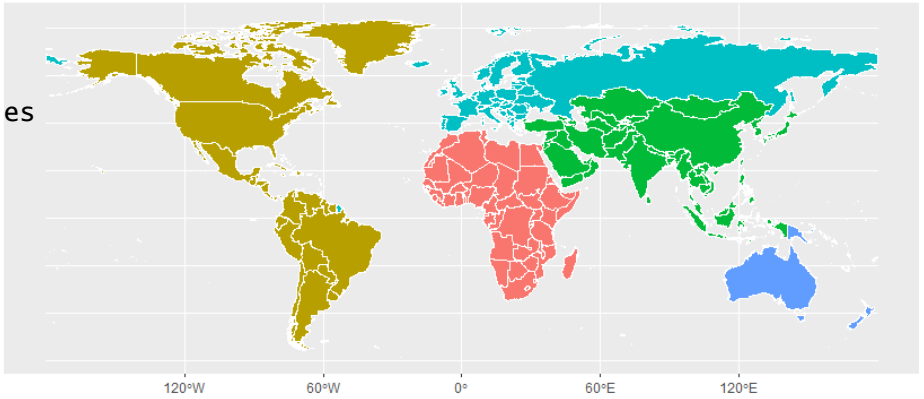
01_crs_projections.R

```
library(tidyverse)
library(rnaturalearth)

world <- ne_countries(scale = "medium", returnclass = "sf") %>%
  filter(name != "Antarctica")

ggplot(data = world) +
  geom_sf()
```

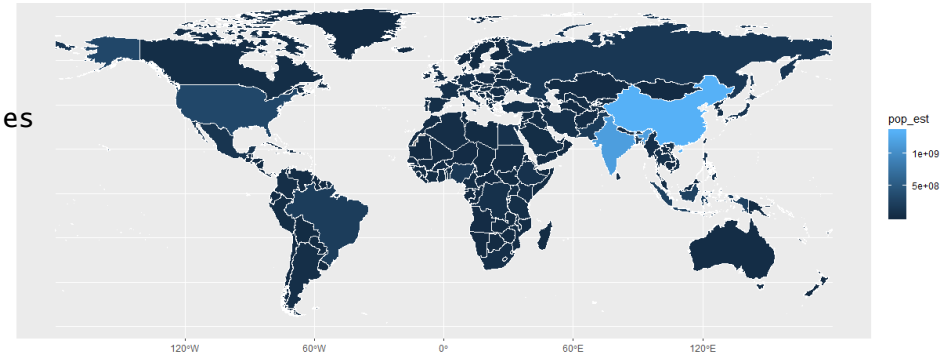


Plotting maps with *ggplot* and *sf*

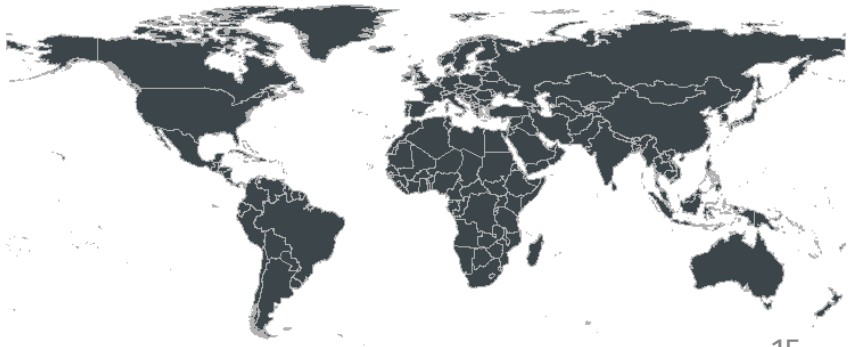
```
ggplot(data = world) +
  geom_sf(
    mapping = aes(
      geometry = geometry, # use Natural Earth World boundaries
      fill = region_un     # fill colour= country's region
    ),
    colour = "white",      # white borders between regions
    show.legend = FALSE    # no legend
  )
```




```
ggplot(data = world) +
  geom_sf(
    mapping = aes(
      geometry = geometry, #use Natural Earth World boundaries
      fill = pop_est  #fill colour = population estimate
    ),
    colour = "white",      # white borders between regions
)
```



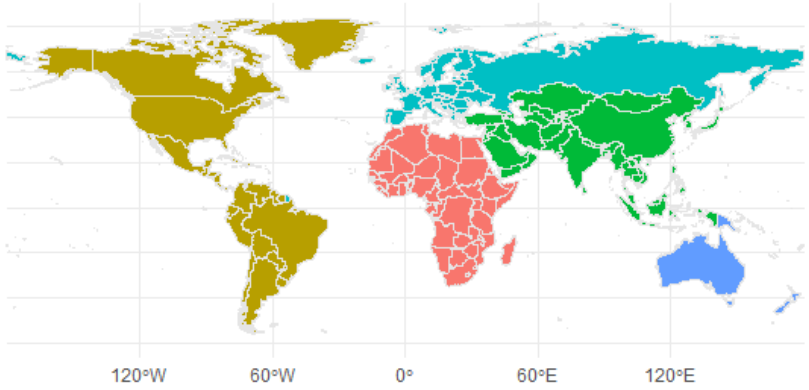

```
ggplot() +
  geom_sf(
    data = world, colour = "#b2b2b2", fill = "#3B454A"
  )+
  coord_sf(datum = NA)+
  theme_minimal()
```



Different projections

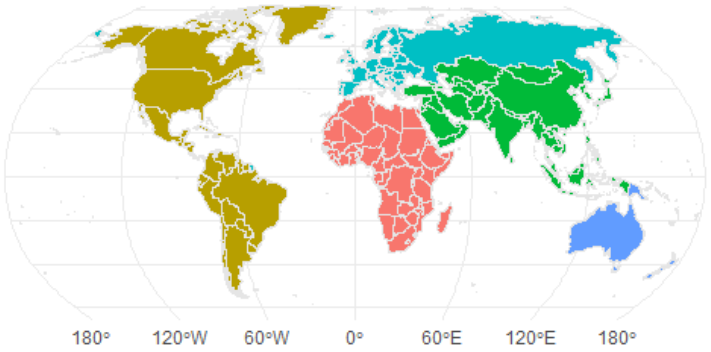
Longitude-latitude

crs = "+proj=longlat +ellps=WGS84"



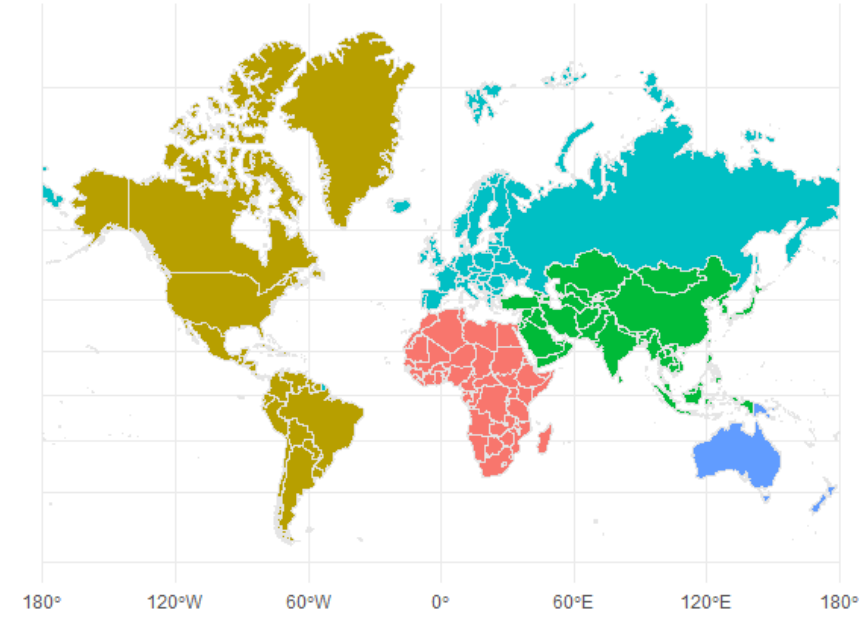
Robinson

crs = "+proj=robin"



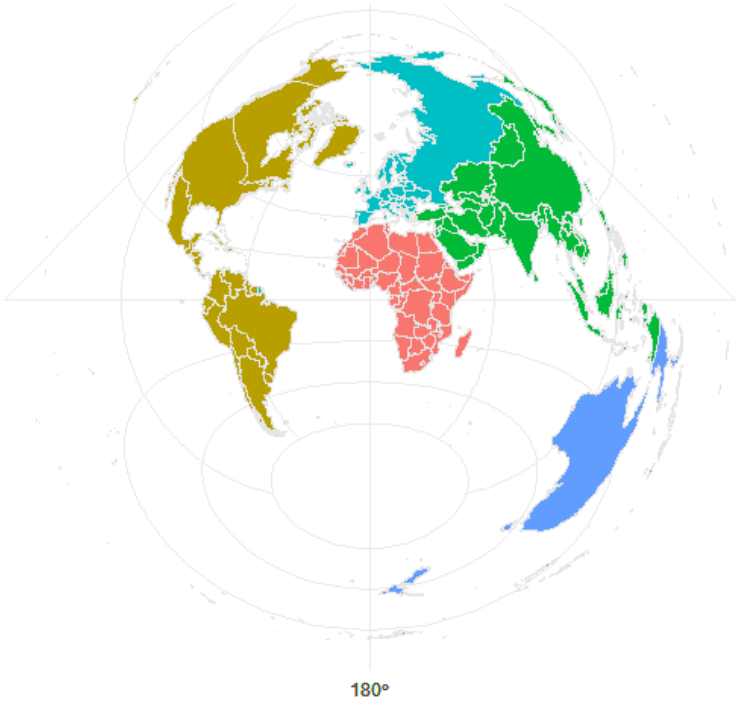
Mercator

crs = "+proj=merc"



Azimuthal Equidistant

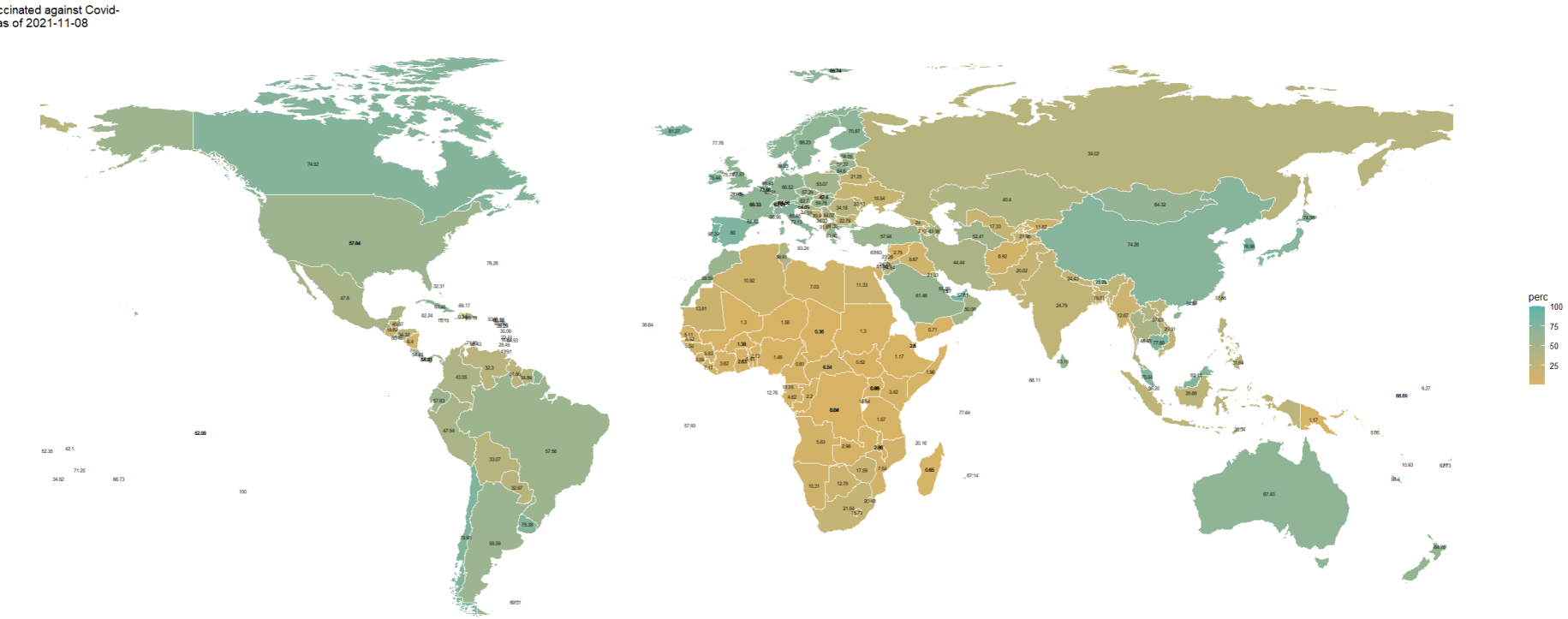
crs = +proj=aeqd

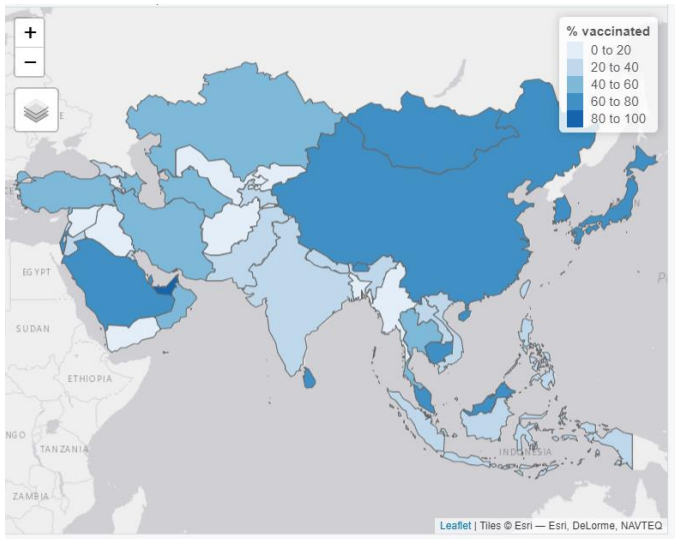


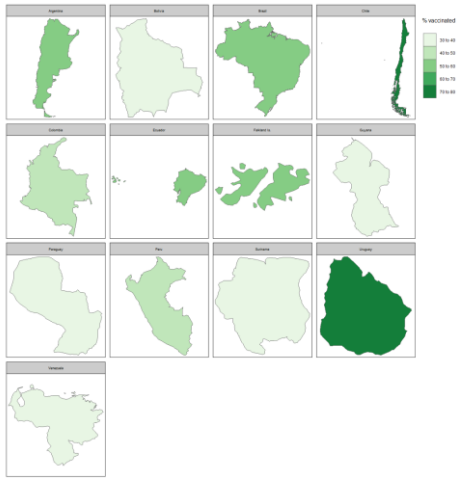
https://observablehq.com/@d3/projection-transitions

Plotting latest COVID vaccinations

01_world_covid_vaccines.R







Source: https://github.com/owid/covid-19-data/tree/master/public/data

London Business School

HIC SUNT DRACONES

Here be dragons

From Wikipedia, the free encyclopedia

This article is about a phrase used on maps. For other uses, see Here be dragons (disambiguation).

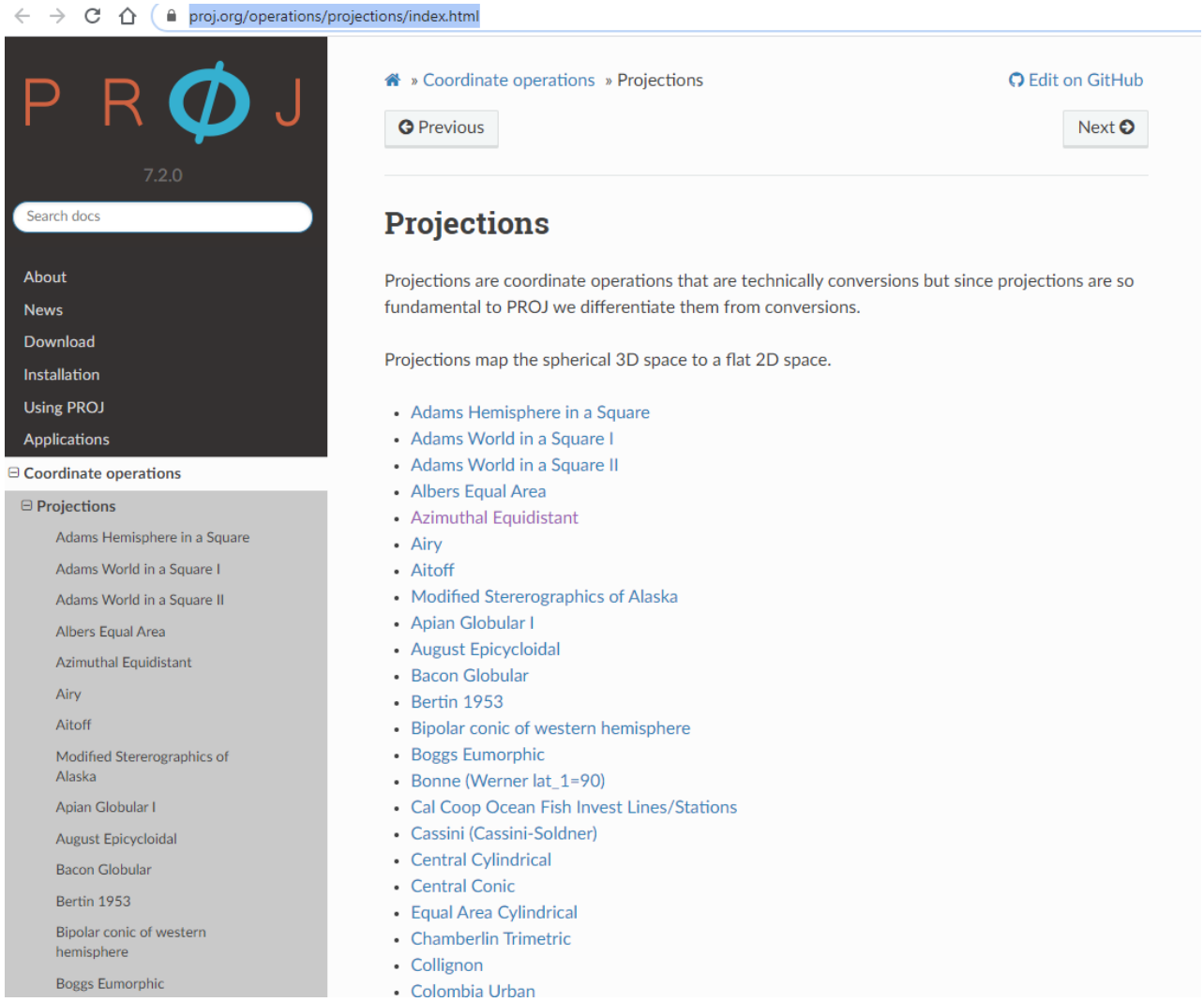
"**Here be dragons**" (*hic sunt dracones* in Latin) means dangerous or unexplored territories, in imitation of a medieval practice of putting illustrations of dragons, sea monsters and other mythological creatures on uncharted areas of maps where potential dangers were thought to exist.[1][2]

Contents [hide]

The text 'Hic Sunt Dracones' on the Hunt–Lenox Globe, dating from 1504

https://en.wikipedia.org/wiki/Here_be_dragons

Different projections




```
map_robinson <- base_map +
  coord_sf(crs = "+proj=robin") +
```

https://proj.org/operations/projections/index.html

London Business School

Contents

- Visualising geospatial data
  - ggplot + the simple features ***sf*** package
  - *An interlude: functional programming-* ***purrr:map()***
  - Geocoding – Opencage
  - Adding data on maps: Points and Lines
  - GIS in R using the simple features ***sf*** package

- Introduction to interactive graphs: Plotly and Shiny

London Business School

Vectors and Lists

purrr functional programming

Of course someone has to write loops. It doesn't have to be you.
--Jenny Bryan

London Business School

Vectors

Vectors have one dimension

- c(1, 3, 5)
- c(TRUE, FALSE, TRUE, TRUE)
- c("red", "blue")

Vectors have a length

- length(c("blue", "red"))

Vectors have a type

- Numeric/double
- Integer
- Factor — forcats
- Character — stringr
- Logical
- Dates — lubridate

Vectors

Define a vector

```
1:3
## [1] 1 2 3

c(1, 2, 3)
## [1] 1 2 3

rep(1, 3)
## [1] 1 1 1

seq(from = 1, to = 3, by = .5)
## [1] 1.0 1.5 2.0 2.5 3.0
```

Subset a vector

Subset vectors with [] or [[]]

```
x <- c(1, 5, 7)
x[2]
## [1] 5

x[[2]]
## [1] 5

x[c(FALSE, TRUE, FALSE)]
## [1] 5
```

Modify elements

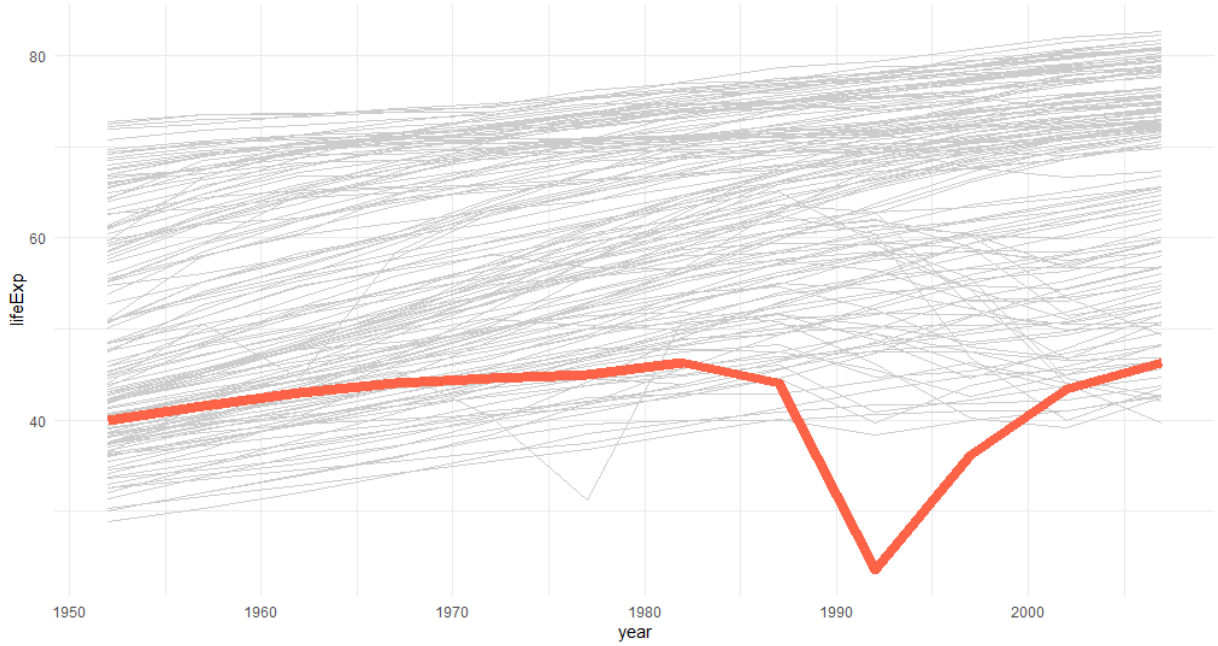
```
x
## [1] 1 5 7

x[2] <- 100
x
## [1] 1 100 7
```

London Business School

Vectors

02_vectors_and_lists.R

```
4  # vectors - one dimensional, one type
5  my_colours <- c("grey80", "tomato")
6
7  # subset and get the first element from this character vector
8  my_colours[1]
9  # [1] "grey80"
10
11 gapminder %>%
12   mutate(rwanda = ifelse(country == "Rwanda", TRUE, FALSE)) %>%
13   ggplot(aes(x = year, y = lifeExp, colour = rwanda, group = country)) +
14   geom_line(
15
16     # the data is a function that takes the original df, and filter(x,!rwanda)
17     data = function(x) filter(x, !rwanda),
18
19     # all lines where rwanda==FALSE, will be coloured 'grey80'
20     colour = my_colours[1]
21   ) +
22   geom_line(
23     data = function(x) filter(x, rwanda),
24
25     # all lines where rwanda==TRUE, will be coloured 'tomato' and be size=3
26     colour = my_colours[2],
27     size = 3
28   ) +
29   theme_minimal()
```



Lists

Lists can contain objects of any type

```
fubar <- list(a = "Hello world", b = 1, c = TRUE, d = rnorm(100), e = mean)

glimpse(fubar)
# List of 5
# $ a: chr "hello"
# $ b: num 1
# $ c: logi TRUE
# $ d: num [1:100] 3.361 1.246 0.549 1.334 -1.274 ...
# $ e:function (x, ...)

skimr::skim(fubar)
# Error in as.data.frame.default(x[[i]], optional = TRUE) :
#   cannot coerce class '"function"' to a data.frame
```

```
grades <- list(
  a = rnorm(20, mean = 78, sd = 8.3),
  b = rnorm(35, mean = 87, sd = 5.6),
  c = rnorm(19, mean = 82, sd = 6.7)
)
> glimpse(grades)
List of 3
 $ a: num [1:20] 84.9 74.1 81.8 77.9 76.1 ...
 $ b: num [1:35] 80.9 81 90.5 85.9 80 ...
 $ c: num [1:19] 93.7 81.5 74.3 85.4 79.5 ...
```

London Business School

Subset a list

Subset lists with either
- **[ ]** returns a list
- **$, [[ ]]** both return a vector

```
> grades[1]
$a
 [1] 84.85153 74.09278 81.78509 77.87915 76.08848 75.22978 76.90725 86.98110 65.39163 71.13352
[11] 75.17001 81.36820 78.29134 83.28183 73.96641 68.01873 88.11229 76.70735 88.32831 66.49468

> grades$a
 [1] 84.85153 74.09278 81.78509 77.87915 76.08848 75.22978 76.90725 86.98110 65.39163 71.13352
[11] 75.17001 81.36820 78.29134 83.28183 73.96641 68.01873 88.11229 76.70735 88.32831 66.49468
> grades[[1]]
 [1] 84.85153 74.09278 81.78509 77.87915 76.08848 75.22978 76.90725 86.98110 65.39163 71.13352
[11] 75.17001 81.36820 78.29134 83.28183 73.96641 68.01873 88.11229 76.70735 88.32831 66.49468
>
> class(grades[1])
[1] "list"
> class(grades$a)
[1] "numeric"
> class(grades[[1]])
[1] "numeric"
```

Dataframes/Tibbles are lists

```
> # dataframes/tibbles are lists, so when you subset with [] you get a tibble
> head(gapminder["pop"])
# A tibble: 6 x 1
       pop
     <int>
1  8425333
2  9240934
3 10267083
4 11537966
5 13079460
6 14880372
>
> # when you subset with [[]] or $ you get a vector
> head(gapminder[["pop"]])
[1]  8425333  9240934 10267083 11537966 13079460 14880372
> head(gapminder$pop)
[1]  8425333  9240934 10267083 11537966 13079460 14880372
```

Vectorized operations work on vectors...

... but not on lists

```
> # vectorized operations work on vectors
> sum(grades$a)
[1] 1587.826
> sum(grades[[1]])
[1] 1587.826
>
> # vectorized operations don't work on lists
> sum(grades[1])
Error in sum(grades[1]) : invalid 'type' (list) of argument
```

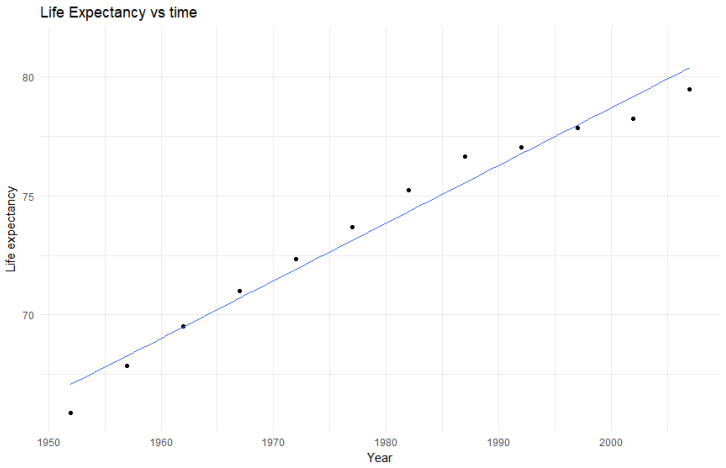
London Business School

- Vectors and Lists in R
- Functional programming using **purrr**

Life expectancy over time (1/2)

Pick one country and explore relationship between life expectancy and time

```
> tempCountry <- "Greece"  # Just a random example
> tempData <- subset(gapminder, country == tempCountry)  # temporary data file with country selected
> tempData
# A tibble: 12 x 6
   country continent  year lifeExp      pop gdpPercap
   <fct>   <fct>     <int>   <dbl>    <int>     <dbl>
 1 Greece  Europe     1952    65.9  7733250     3531.
 2 Greece  Europe     1957    67.9  8096218     4916.
 3 Greece  Europe     1962    69.5  8448233     6017.
 4 Greece  Europe     1967    71    8716441     8513.
 5 Greece  Europe     1972    72.3  8888628    12725.
 6 Greece  Europe     1977    73.7  9308479    14196.
 7 Greece  Europe     1982    75.2  9786480    15268.
 8 Greece  Europe     1987    76.7  9974490    16121.
 9 Greece  Europe     1992    77.0 10325429    17541.
10 Greece  Europe     1997    77.9 10502372    18748.
11 Greece  Europe     2002    78.3 10603863    22514.
12 Greece  Europe     2007    79.5 10706290    27538.

> tempModel1 <- lm(lifeExp~year,data=tempData)
> msummary(tempModel1)
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -406.095     25.773   -15.8  2.2e-08
year           0.242      0.013    18.6  4.3e-09

Residual standard error: 0.778 on 10 degrees of freedom
Multiple R-squared:  0.972,     Adjusted R-squared:  0.969
F-statistic:  347 on 1 and 10 DF,  p-value: 4.32e-09
```

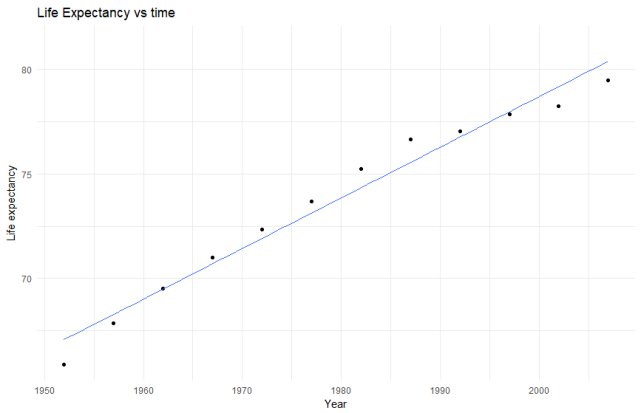


Value of intercept: Did Greek people really have a life expectancy of **-406** years in year 0?

Life expectancy over time (2/2)

- Sanity check of model fit. It makes more sense for the intercept to correspond to life expectancy in 1952, the earliest date in our dataset, rather than year 0.
- Find the minimum year in the dataset and rerun the regression

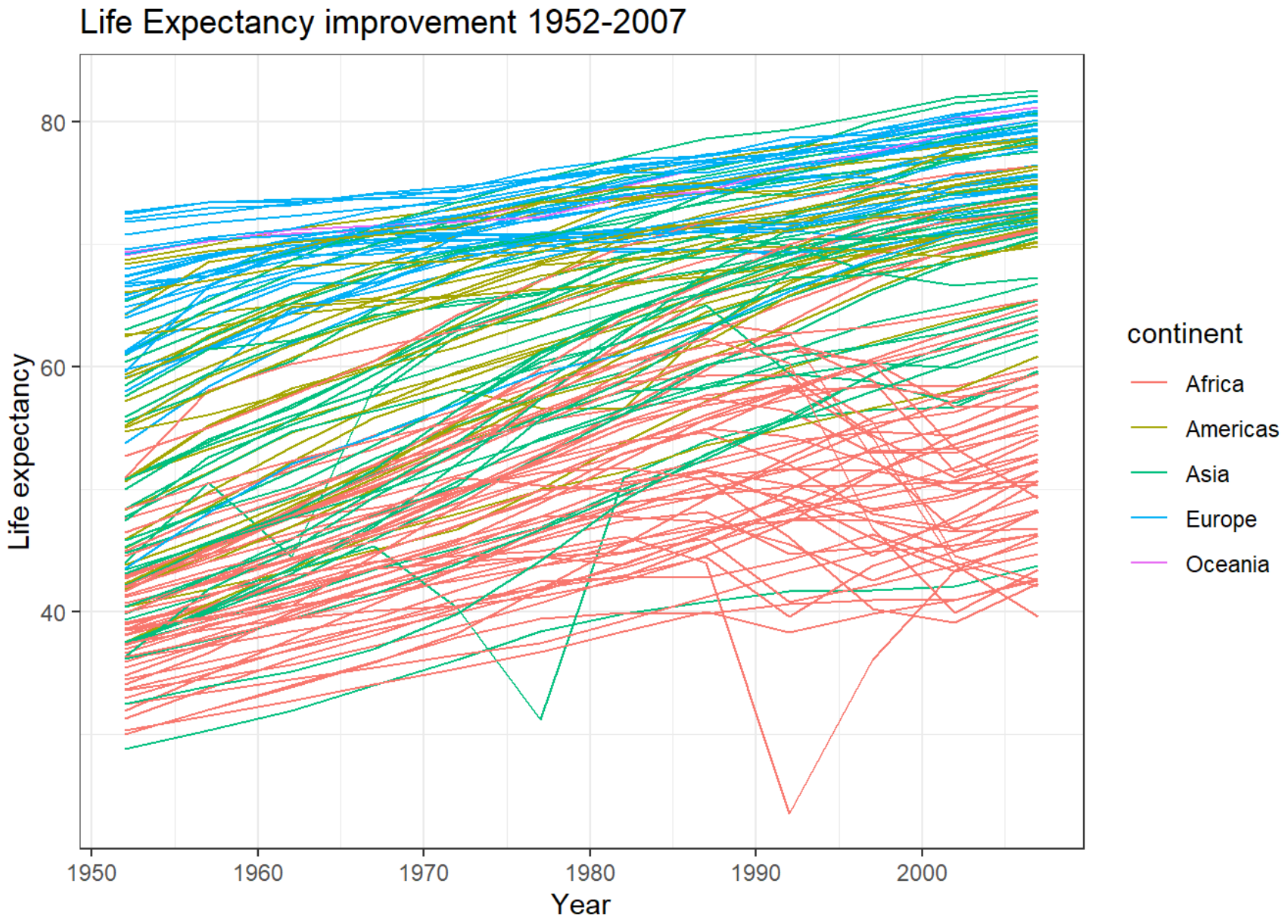
```
> yearMin <- min(gapminder$year)
>
> tempModel2 <- lm(lifeExp ~ I(year - yearMin), data=tempDa
> summary(tempModel2)

Call:
lm(formula = lifeExp ~ I(year - yearMin), data = tempData)

Residuals:
    Min      1Q  Median      3Q     Max
-1.207 -0.543  0.143  0.457  1.119

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)         67.067      0.423   158.7  < 2e-16
I(year - yearMin)    0.242      0.013    18.6  4.3e-09

Residual standard error: 0.778 on 10 degrees of freedom
Multiple R-squared:  0.972,     Adjusted R-squared:  0.969
F-statistic:  347 on 1 and 10 DF,  p-value: 4.32e-09
```



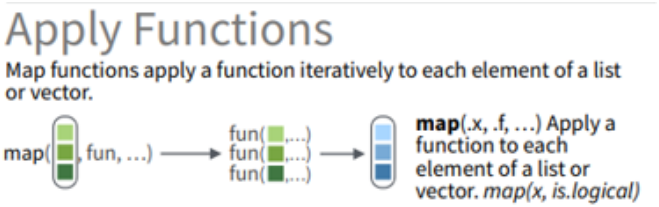
67 (value of intercept) was the life expectancy in 1952

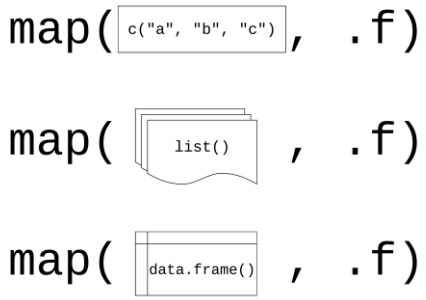
London Business School

Life expectancy regression for all countries (1/3)



Functional programming *purr::map()*






```
library(purrr)
set.seed(42)

x_list <- list(x = rnorm(100), # generates 100 random numbers from N(0,1)
               y = rnorm(100),
               z = rnorm(100))

map(x_list, mean) # take the list and apply the function 'mean' to each item on list
```

```
$x
[1] 0.03251482

$y
[1] -0.08748371

$z
[1] -0.01036817
```

```
# find the standard deviation (sd) for all numeric variables in gapminder
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map(sd)
```

```
$year
[1] 17.26533

$lifeExp
[1] 12.91711

$pop
[1] 106157897

$gdpPercap
[1] 9857.455
```

Functional programming ***purr::map()***

03_functional_programming_gapminder.Rmd

Three ways to pass functions to **map()**

1. pass directly to **map()**

```
map(.x,
      mean, na.rm = TRUE)
```

2. use an anonymous function

```
map(.x,
      function(.x){
         mean(.x,
         na.rm = TRUE)
      }
)
```

3. use **~**

```
map(.x,
      ~mean(.x,
      na.rm = TRUE)
)
```

```
many_models <- gapminder %>%
  # create a dataframe containing a separate dataframe for each country
  group_by(country,continent) %>%

  # nesting creates a list-column of data frames;
  # we will use the list to fit a regression model for every country
  nest()  %>%

  # Run a simple regression model for every country in the dataframe
  mutate(simple_model = data %>%
           map(~lm(lifeExp ~ I(year - yearMin), data = .))) %>%

  # extract coefficients and model details with broom::tidy
  mutate(coefs = simple_model %>% map(~ tidy(., conf.int = TRUE)),
         details = simple_model %>% map(glance)) %>%
  ungroup()
```

London Business School

purr::map() to run 142 regressions for all countries

```
library(broom)
library(forcats)
many_models <- gapminder %>%
  # create a dataframe containing a separate dataframe for each country
  group_by(country,continent) %>%
  nest()  %>%

  # Run a simple regression model for every country in the dataframe
  mutate(simple_model = data %>%
           map(~lm(lifeExp ~ I(year - yearMin), data = .))) %>%

  # extract coefficients and model details with broom::tidy
  mutate(coefs = simple_model %>% map(~ tidy(., conf.int = TRUE)),
         details = simple_model %>% map(glance)) %>%
  ungroup()

head(many_models, 10)
```

```
## # A tibble: 10 x 6
##    country     continent         data simple_model coefs       details
##    <fct>       <fct>     <list<df[,4> <list>       <list>      <list>
##  1 Afghanist~ Asia          [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  2 Albania    Europe        [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  3 Algeria    Africa        [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  4 Angola     Africa        [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  5 Argentina  Americas      [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  6 Australia  Oceania       [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  7 Austria    Europe        [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  8 Bahrain    Asia          [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
##  9 Bangladesh Asia          [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
## 10 Belgium    Europe        [12 x 4] <lm>         <tibble [2~ <tibble [1 x~
```

London Business School

Get 142 intercepts (life expectancy in 1952)

```
intercepts <-
  many_models %>%
  unnest(coefs) %>%
  filter(term == "(Intercept)") %>%
  arrange(estimate) %>%
  mutate(country = fct_inorder(country)) %>%
  select(country, continent, estimate, std.error, conf.low, conf.high)

# let us look at the first 20 intercepts, or life expectancy in 1952
head(intercepts,20)
```

```
## # A tibble: 20 x 6
##    country           continent estimate std.error conf.low conf.high
##    <fct>             <fct>        <dbl>     <dbl>    <dbl>     <dbl>
##  1 Gambia            Africa        28.4     0.623     27.0      29.8
##  2 Afghanistan       Asia          29.9     0.664     28.4      31.4
##  3 Yemen, Rep.       Asia          30.1     0.861     28.2      32.0
##  4 Sierra Leone      Africa        30.9     0.448     29.9      31.9
##  5 Guinea            Africa        31.6     0.649     30.1      33.0
##  6 Guinea-Bissau     Africa        31.7     0.349     31.0      32.5
##  7 Angola            Africa        32.1     0.764     30.4      33.8
##  8 Mali              Africa        33.1     0.262     32.5      33.6
##  9 Mozambique        Africa        34.2     1.24      31.4      37.0
## 10 Equatorial Guinea Africa        34.4     0.178     34.0      34.8
## 11 Nepal             Asia          34.4     0.502     33.3      35.5
```

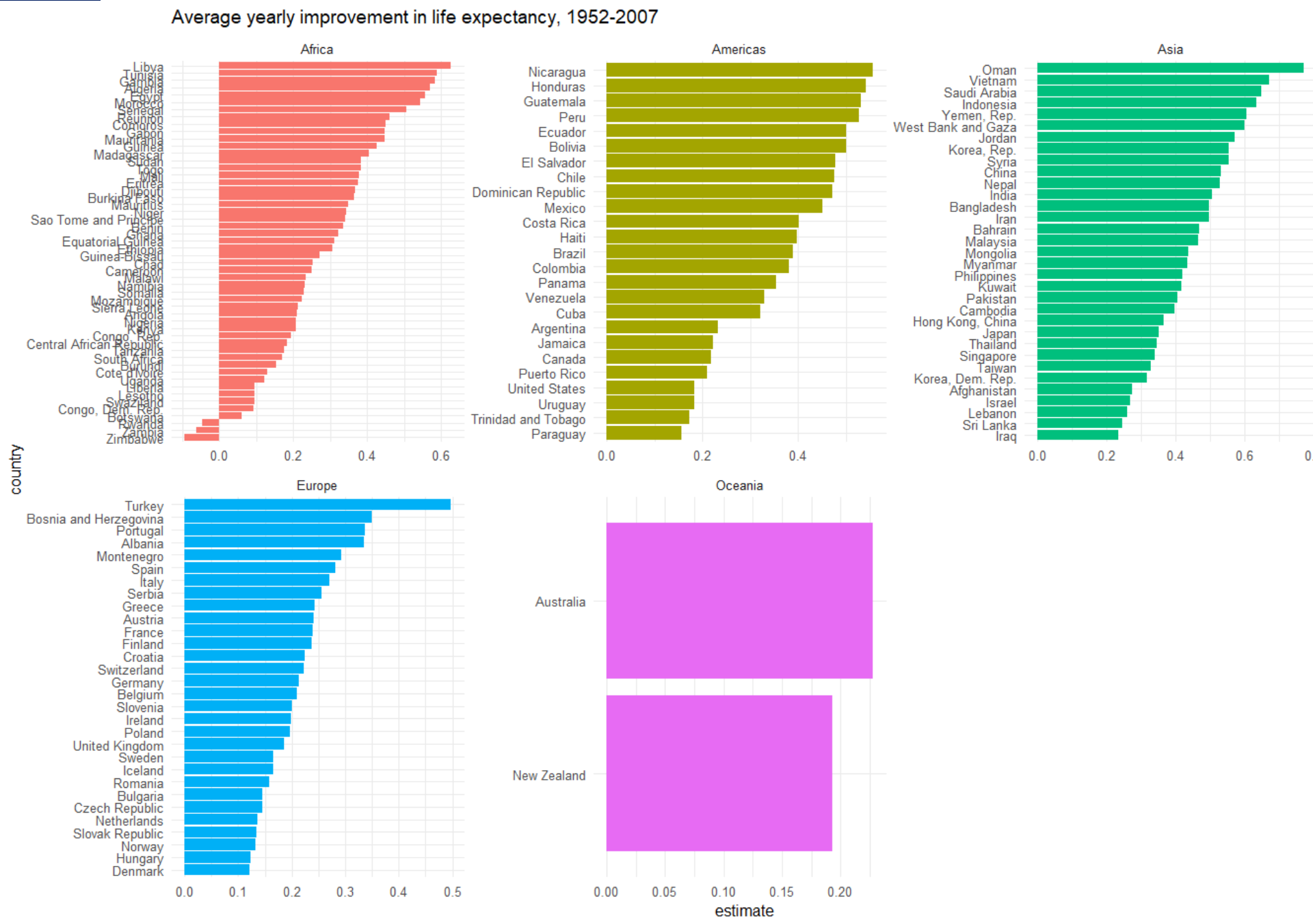
London Business School

Get 142 slopes (average yearly improvement in life expectancy 1952-2007)

```
slopes <- many_models %>%
  unnest(coefs) %>%
  filter(term == "I(year - yearMin)") %>%
  arrange(estimate) %>%
  mutate(country = fct_inorder(country)) %>%
  select(country, continent, estimate, std.error, conf.low, conf.high)

# let us look at the first 20 slopes, or average improvement in life exepctancy per year
head(slopes,20)
```

```
## # A tibble: 20 x 6
##    country          continent estimate std.error conf.low conf.high
##    <fct>            <fct>        <dbl>     <dbl>    <dbl>     <dbl>
##  1 Zimbabwe         Africa     -0.0930   0.121   -0.362       0.175
##  2 Zambia           Africa     -0.0604   0.0757  -0.229       0.108
##  3 Rwanda           Africa     -0.0458   0.110   -0.290       0.199
##  4 Botswana         Africa      0.0607   0.102   -0.167       0.288
##  5 Congo, Dem. Rep. Africa      0.0939   0.0406   0.00338     0.184
##  6 Swaziland        Africa      0.0951   0.111   -0.153       0.343
##  7 Lesotho          Africa      0.0956   0.0992  -0.126       0.317
##  8 Liberia          Africa      0.0960   0.0297   0.0299      0.162
##  9 Denmark          Europe      0.121    0.00667  0.106       0.136
## 10 Uganda           Africa      0.122    0.0533   0.00280     0.240
## 11 Hungary          Europe      0.124    0.0199   0.0794      0.168
## 12 Cote d'Ivoire    Africa      0.131    0.0657  -0.0157      0.277
## 13 Norway           Europe      0.132    0.00819  0.114       0.150
## 14 Slovak Republic  Europe      0.134    0.0217   0.0856      0.182
## 15 Netherlands      Europe      0.137    0.00582  0.124       0.150
## 16 Czech Republic   Europe      0.145    0.0138   0.114       0.176
## 17 Bulgaria         Europe      0.146    0.0420   0.0522      0.239
```


Life expectancy in 1952
Africa
Americas
Asia
Europe
Oceania
country
Mauritius
Zimbabwe
Reunion
Botswana
South Africa
Sao Tome and Principe
Zambia
Lesotho
Congo, Rep.
Namibia
Kenya
Swaziland
Cote d'Ivoire
Tunisia
Uganda
Ghana
Algeria
Tanzania
Nigeria
Morocco
Libya
Congo, Dem. Rep.
Cameroon
Togo
Egypt
Burundi
Mauritania
Comoros
Liberia
Chad
Benin
Gabon
Central African Republic
Sudan
Rwanda
Malawi
Senegal
Madagascar
Djibouti
Ethiopia
Eritrea
Niger
Burkina Faso
Somalia
Equatorial Guinea
Mozambique
Mali
Angola
Guinea-Bissau
Guinea
Sierra Leone
Gambia
Canada
United States
Puerto Rico
Uruguay
Argentina
Jamaica
Paraguay
Cuba
Trinidad and Tobago
Costa Rica
Panama
Venezuela
Chile
Colombia
Mexico
Brazil
Ecuador
Dominican Republic
El Salvador
Peru
Nicaragua
Honduras
Guatemala
Haiti
Bolivia
Israel
Japan
Hong Kong, China
Singapore
Taiwan
Sri Lanka
Lebanon
Kuwait
Korea, Dem. Rep.
Bahrain
Thailand
Malaysia
Iraq
Korea, Rep.
Philippines
China
Syria
Iran
Jordan
Mongolia
West Bank and Gaza
Pakistan
Myanmar
Saudi Arabia
India
Vietnam
Oman
Cambodia
Indonesia
Bangladesh
Nepal
Yemen, Rep.
Afghanistan
Norway
Iceland
Netherlands
Sweden
Denmark
Switzerland
United Kingdom
Belgium
France
Germany
Ireland
Czech Republic
Greece
Slovak Republic
Italy
Spain
Finland
Austria
Slovenia
Hungary
Bulgaria
Poland
Romania
Croatia
Montenegro
Serbia
Portugal
Albania
Bosnia and Herzegovina
Turkey
New Zealand
Australia
0
20
40
60

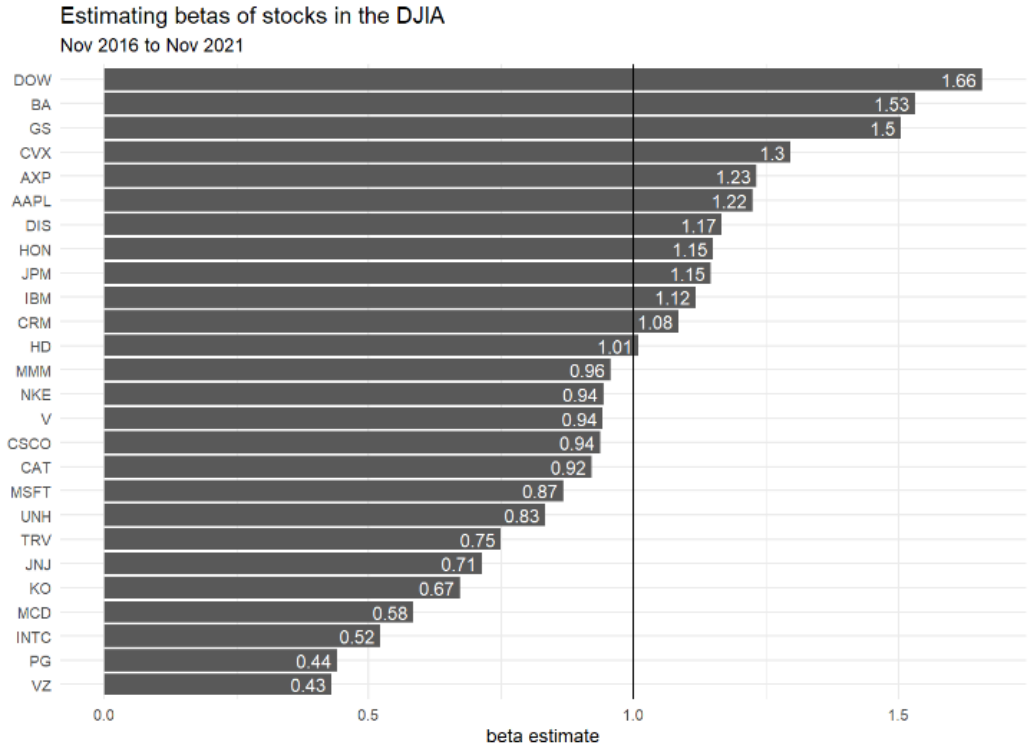

Average yearly improvement in life expectancy, 1952-2007
Africa
Libya
Tunisia
Gabon
Algeria
Egypt
Morocco
Senegal
Reunion
Comoros
Gabon
Mauritania
Guinea
Madagascar
Sudan
Togo
Mali
Eritrea
Djibouti
Burkina Faso
Mauritius
Niger
Sao Tome and Principe
Benin
Ghana
Equatorial Guinea
Ethiopia
Guinea-Bissau
Chad
Cameroon
Malawi
Namibia
Somalia
Mozambique
Sierra Leone
Angola
Nigeria
Kenya
Congo, Rep.
Central African Republic
Tanzania
South Africa
Burundi
Cote d'Ivoire
Uganda
Liberia
Lesotho
Swaziland
Congo, Dem. Rep.
Botswana
Rwanda
Zambia
Zimbabwe
0.0
0.2
0.4
0.6
Americas
Nicaragua
Honduras
Guatemala
Peru
Ecuador
Bolivia
El Salvador
Chile
Dominican Republic
Mexico
Costa Rica
Haiti
Brazil
Colombia
Panama
Venezuela
Cuba
Argentina
Jamaica
Canada
Puerto Rico
United States
Uruguay
Trinidad and Tobago
Paraguay
0.0
0.2
0.4
Asia
Oman
Vietnam
Saudi Arabia
Indonesia
Yemen, Rep.
West Bank and Gaza
Jordan
Korea, Rep.
Syria
China
Nepal
India
Bangladesh
Iran
Bahrain
Malaysia
Mongolia
Myanmar
Philippines
Kuwait
Pakistan
Cambodia
Hong Kong, China
Japan
Thailand
Singapore
Taiwan
Korea, Dem. Rep.
Afghanistan
Israel
Lebanon
Sri Lanka
Iraq
0.0
0.2
0.4
0.6
country
Europe
Turkey
Bosnia and Herzegovina
Portugal
Albania
Montenegro
Spain
Italy
Serbia
Greece
Austria
France
Finland
Croatia
Switzerland
Germany
Belgium
Slovenia
Ireland
Poland
United Kingdom
Sweden
Iceland
Romania
Bulgaria
Czech Republic
Netherlands
Slovak Republic
Norway
Hungary
Denmark
0.0
0.1
0.2
0.3
0.4
0.5
Oceania
Australia
New Zealand
0.00
0.05
0.10
0.15
0.20
estimate
Source: Gapminder package

London Business School

CAPM for all DJIA stocks

04_many_CAPM.Rmd

```
many_models <- DJIA_data %>%
  # create a dataframe containing a separate dataframe for each stock
  group_by(symbol) %>%
  nest()  %>%

  # Run a simple regression model for every stock in the dataframe
  mutate(simple_capm_model = data %>%
           map(~lm(monthly_return ~  SPY_return, data = .))) %>%

  # extract coefficients and model details with broom::tidy
  mutate(coefs = simple_capm_model %>%
           map(~ tidy(., conf.int = TRUE)),
         details = simple_capm_model %>% map(glance)) %>%
  ungroup()

# pull intercepts, or alphas
intercepts <-
  many_models %>%
  unnest(coefs) %>%
  filter(term == "(Intercept)") %>%
  arrange(estimate) %>%
  mutate(symbol = fct_inorder(symbol)) %>%
  select(symbol, estimate, std.error, conf.low, conf.high)
```



CAPM for DJIA stocks

symbol	beta	se_beta	beta_low	beta_high	alpha	r.squared	residual_se
DOW	1.658	0.235	1.178	2.138	-0.016	0.616	0.068
BA	1.532	0.310	0.910	2.153	-0.007	0.292	0.105
GS	1.505	0.161	1.183	1.826	-0.006	0.597	0.054
CVX	1.296	0.175	0.945	1.647	-0.012	0.481	0.059
AXP	1.232	0.138	0.956	1.508	0.001	0.575	0.047
AAPL	1.224	0.193	0.838	1.611	0.014	0.405	0.065
DIS	1.167	0.177	0.813	1.520	-0.004	0.425	0.060
HON	1.150	0.103	0.943	1.357	-0.002	0.677	0.035
JPM	1.146	0.140	0.865	1.427	0.000	0.530	0.047
IBM	1.118	0.148	0.821	1.414	-0.014	0.491	0.050
CRM	1.085	0.195	0.696	1.474	0.010	0.345	0.066
HD	1.008	0.129	0.750	1.266	0.006	0.508	0.044
MMM	0.957	0.125	0.706	1.208	-0.010	0.497	0.042
NKE	0.943	0.164	0.615	1.270	0.009	0.360	0.055
V	0.941	0.126	0.688	1.194	0.004	0.484	0.043
CSCO	0.937	0.164	0.609	1.266	0.001	0.356	0.055
CAT	0.921	0.180	0.560	1.281	0.005	0.307	0.061
MSFT	0.868	0.106	0.656	1.079	0.018	0.533	0.036
UNH	0.833	0.155	0.523	1.144	0.010	0.328	0.052
TRV	0.750	0.131	0.488	1.011	-0.002	0.357	0.044
JNJ	0.714	0.112	0.490	0.938	-0.002	0.407	0.038
KO	0.671	0.112	0.447	0.896	-0.001	0.378	0.038
MCD	0.585	0.122	0.340	0.829	0.007	0.279	0.041
INTC	0.522	0.205	0.112	0.932	0.003	0.099	0.069
PG	0.440	0.116	0.207	0.673	0.006	0.195	0.039
VZ	0.429	0.122	0.186	0.673	0.000	0.174	0.041

London Business School

CAPM for all SP500 stocks

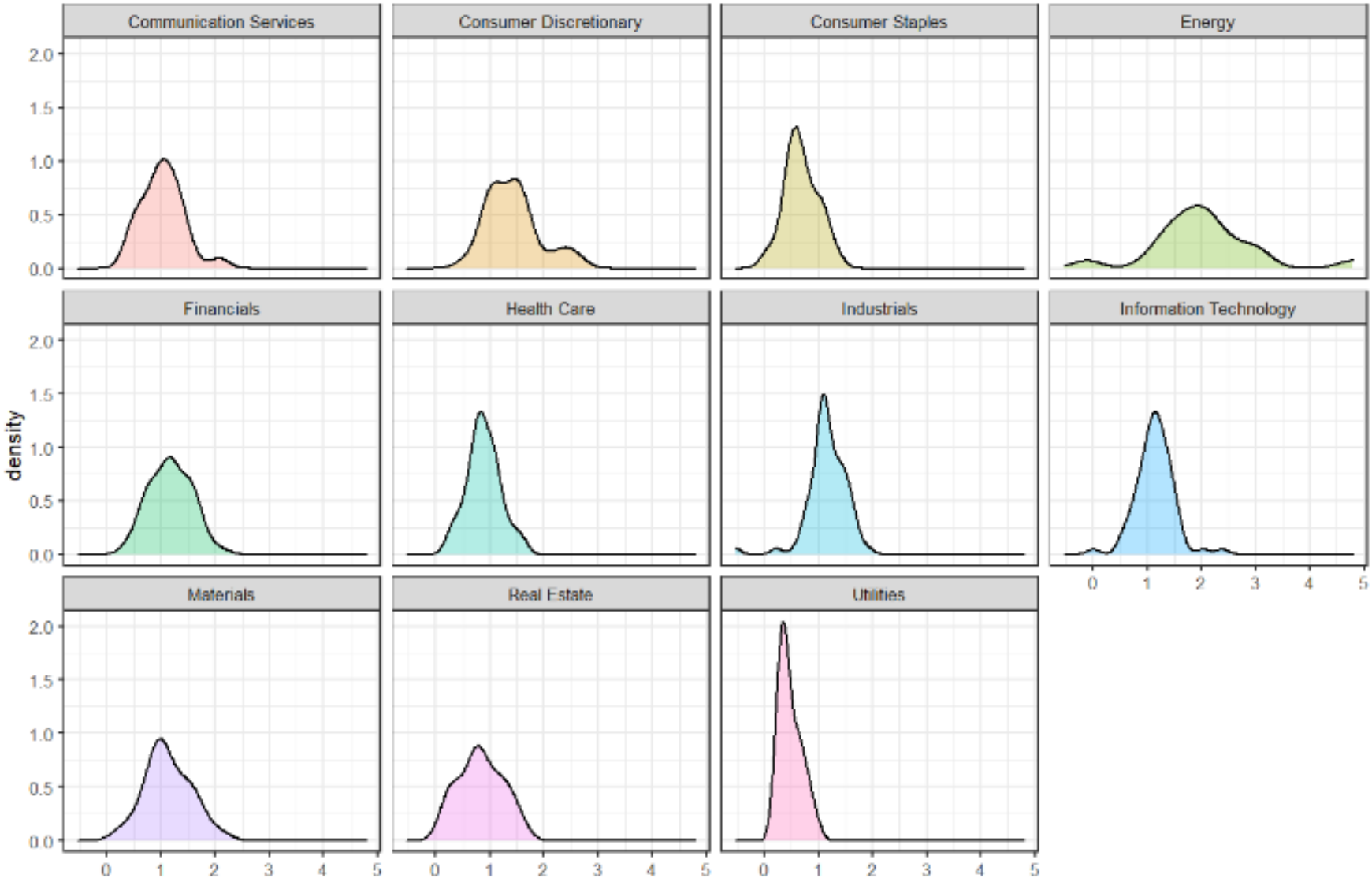
04_many_CAPM.Rmd

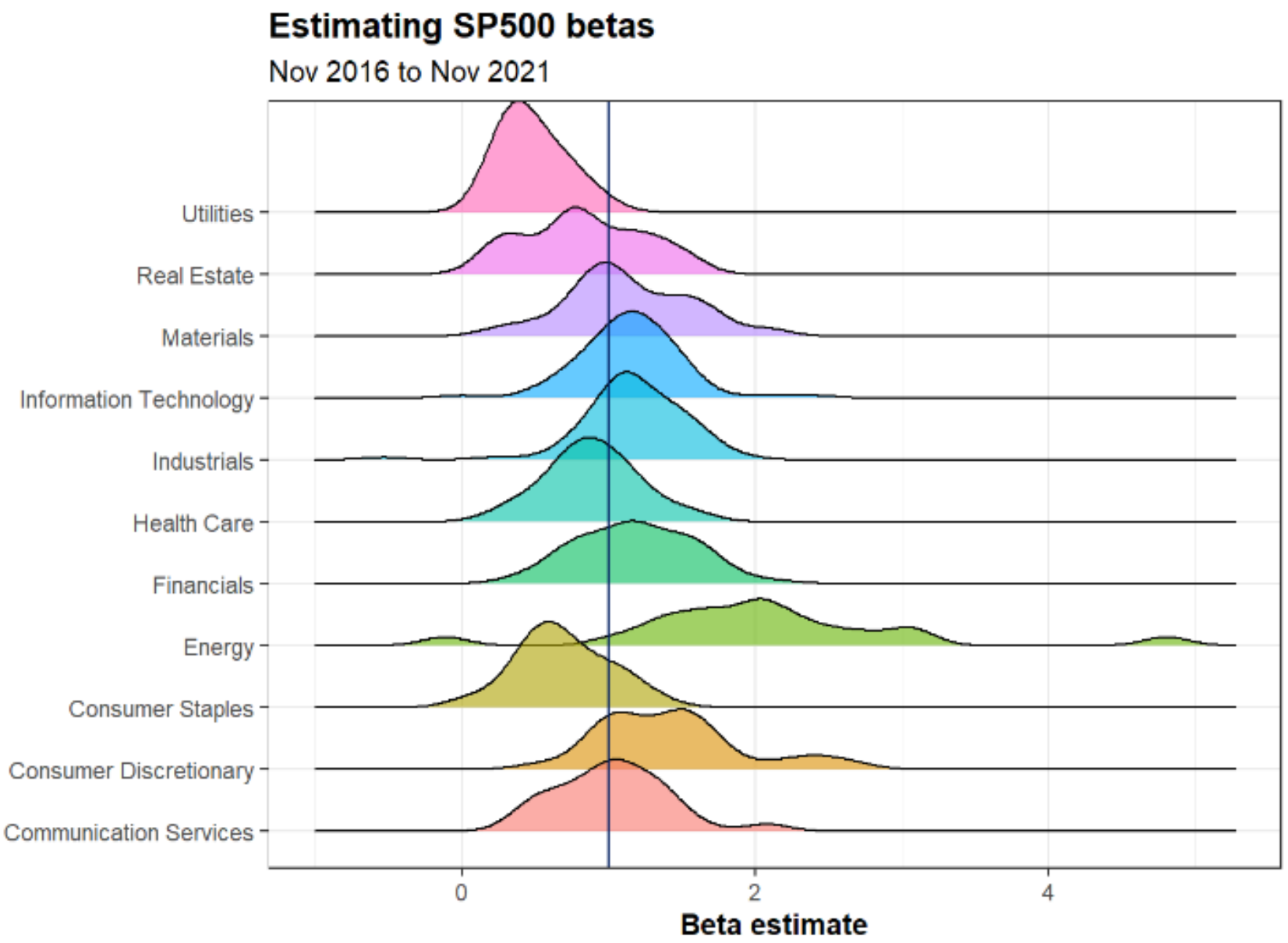
```
too_many_models <- sp500_data %>%
  # create a list containing a separate dataframe for each stock
  group_by(symbol) %>%
  nest()  %>%

  # Run a simple regression model for every country in the dataframe
  mutate(simple_capm_model = data %>%
           map(~lm(monthly_return ~  SPY_return, data = .))) %>%

  # extract regression coefficients and model details with broom::tidy()
  mutate(coefs = simple_capm_model %>%
           map(~ tidy(., conf.int = TRUE)),
         details = simple_capm_model %>%
           map(glance)) %>%
  ungroup()


# pull intercepts, or alphas
intercepts <-
  too_many_models %>%
  unnest(coefs) %>%
  filter(term == "(Intercept)") %>%
  arrange(estimate) %>%
  mutate(symbol = fct_inorder(symbol)) %>%
  select(symbol, estimate, std.error, conf.low, conf.high)

# let us look at the intercepts, or alphas
# intercepts %>%
#   rename(alpha = estimate) %>%
#   kable()%>%
#   kable_styling(bootstrap_options = c("striped", "hover", "condensed"))


# pull slopes, or betas
slopes <- too_many_models %>%
  unnest(coefs) %>%
  filter(term == "SPY_return") %>%
  arrange(estimate) %>%
  mutate(symbol = fct_inorder(symbol)) %>%
  select(symbol, estimate, std.error, conf.low, conf.high)
```

London Business School

Estimating betas of stocks

Nov 2016 to Nov 2021

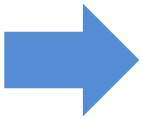



Estimating SP500 betas
Nov 2016 to Nov 2021
Utilities
Real Estate
Materials
Information Technology
Industrials
Health Care
Financials
Energy
Consumer Staples
Consumer Discretionary
Communication Services
0
2
4
Beta estimate

London Business School

Contents

- Visualising geospatial data
  - ggplot + the simple features ***sf*** package
  - *An interlude: functional programming-* ***purrr:map()***
  - Geocoding – Opencage
  - Adding data on maps: Points and Lines
  - GIS in R using the simple features ***sf*** package



- Introduction to interactive graphs: Plotly and Shiny

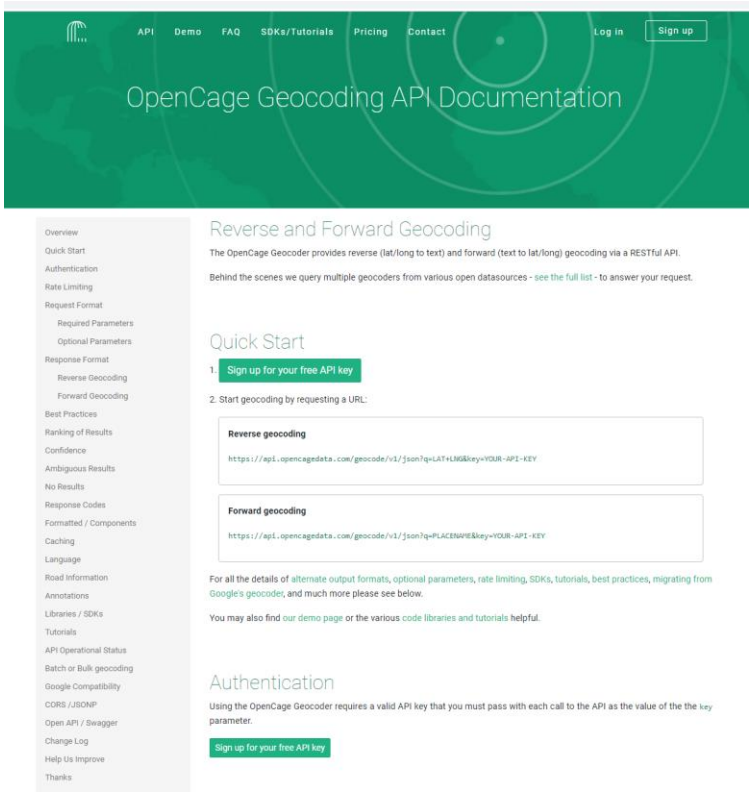
Geocoding with *opencage* (1/3)

Get an API key on https://opencagedata.com/api

To save your API key, you must create (or edit) ***.Renviron*** a hidden file that lives in your home directory.

usethis::edit_r_environ()

- Add a line
  OPENCAGE_KEY="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
- Before you exit, make sure your .Renviron ends with a blank line, then save and close it.
- Restart RStudio after modifying .Renviron in order to load the API key into memory.
- To check everything worked, go to console and type **Sys.getenv("OPENCAGE_KEY")**



https://opencagedata.com/api

Geocoding with *opencage* (2/3)

1. opencage returns a list
2. First element in the list (*results*) contains all geocoding info, but we will use *geometry.lng, geometry.lat*
3. opencage will try to return as many matches as it can when it geocodes an address
4. Typically, the first result is your best bet

```
> address <- "London Business School"
> geocoded_address <- opencage_forward(address)
>
> geocoded_address
$results
# A tibble: 9 x 81
  annotations.DMS~ annotations.DMS~ annotations.MGRS annotations.Mai~ annotations.Mer~ annotations.Mer~ annotations.OSM~
  <chr>            <chr>            <chr>           <chr>            <chr>            <chr>            <chr>
1 51° 31' 35.0148~ 0° 9' 40.12992'~ 30UXC9692012185  IO91wm06pi       -17938.826       6681466.496      https://www.ope~
2 51° 31' 16.5799~ 0° 9' 36.92988'~ 30UXC9700411618  IO91wm05sc       -17839.874       6680552.688      https://www.ope~
3 51° 31' 29.5694~ 0° 9' 44.64612'~ 30UXC9684012013  IO91wm05mx       -18078.472       6681196.561      https://www.ope~
4 51° 31' 36.4839~ 0° 9' 43.89948'~ 30UXC9684612227  IO91wm06mk       -18055.39        6681539.317      https://www.ope~
5 51° 31' 28.7842~ 0° 9' 42.40080'~ 30UXC9688411991  IO91wm05ow       -18009.045       6681157.632      https://www.ope~
6 51° 31' 40.2693~ 0° 9' 57.56508'~ 30UXC9657812334  IO91wm06bq       -18477.952       6681726.981      https://www.ope~
7 51° 31' 25.8013~ 0° 9' 40.47876'~ 30UXC9692411900  IO91wm05pr       -17949.614       6681009.775      https://www.ope~
8 51° 31' 34.0356~ 0° 9' 42.99120'~ 30UXC9686612152  IO91wm06ng       -18027.301       6681417.946      https://www.ope~
9 45° 8' 56.90400~ 11° 49' 14.0520~ 32TQR2172603388  JN55vd85it       1315859.833      5614726.846      NA
# ... with 74 more variables: annotations.OSM.note_url <chr>, annotations.OSM.url <chr>,
#   annotations.UN_M49.regions.EUROPE <chr>, annotations.UN_M49.regions.GB <chr>,
#   annotations.UN_M49.regions.NORTHERN_EUROPE <chr>, annotations.UN_M49.regions.WORLD <chr>,
#   annotations.UN_M49.statistical_groupings <chr>, annotations.callingcode <chr>,
#   annotations.currency.decimal_mark <chr>, annotations.currency.html_entity <chr>, annotations.currency.iso_code <chr>,
#   annotations.currency.iso_numeric <chr>, annotations.currency.name <chr>,
#   annotations.currency.smallest_denomination <chr>, annotations.currency.subunit <chr>,
#   annotations.currency.subunit_to_unit <chr>, annotations.currency.symbol <chr>,
#   annotations.currency.symbol_first <chr>, annotations.currency.thousands_separator <chr>, annotations.flag <chr>,
#   annotations.geohash <chr>, annotations.qibla <chr>, annotations.roadinfo.drive_on <chr>,
#   annotations.roadinfo.road <chr>, annotations.roadinfo.speed_in <chr>, annotations.sun.rise.apparent <chr>,
#   annotations.sun.rise.astronomical <chr>, annotations.sun.rise.civil <chr>, annotations.sun.rise.nautical <chr>,
#   annotations.sun.set.apparent <chr>, annotations.sun.set.astronomical <chr>, annotations.sun.set.civil <chr>,
#   annotations.sun.set.nautical <chr>, annotations.timezone.name <chr>, annotations.timezone.now_in_dst <chr>,
#   annotations.timezone.offset_sec <chr>, annotations.timezone.offset_string <chr>,
#   annotations.timezone.short_name <chr>, annotations.what3words.words <chr>, annotations.wikidata <chr>,
#   bounds.northeast.lat <chr>, bounds.northeast.lng <chr>, bounds.southwest.lat <chr>, bounds.southwest.lng <chr>,
#   `components.ISO_3166-1_alpha-2` <chr>, `components.ISO_3166-1_alpha-3` <chr>, components._category <chr>,
#   components._type <chr>, components.city <chr>, components.continent <chr>, components.country <chr>,
#   components.country_code <chr>, components.county <chr>, components.county_code <chr>, components.postcode <chr>,
#   components.road <chr>, components.state <chr>, components.state_code <chr>, components.state_district <chr>,
#   components.suburb <chr>, components.university <chr>, confidence <chr>, formatted <chr>, geometry.lat <dbl>,
#   geometry.lng <dbl>, components.building <chr>, components.house_number <chr>, components.highway <chr>,
#   annotations.UN_M49.regions.IT <chr>, annotations.UN_M49.regions.SOUTHERN_EUROPE <chr>,
#   components.local_administrative_area <chr>, components.political_union <chr>, components.town <chr>, query <chr>

$total_results
[1] 9

$time_stamp
[1] "2020-11-11 11:16:58 +03"

$rate_info
# A tibble: 1 x 3
  limit remaining reset
  <int>     <int> <dttm>
1  2500      2473 2020-11-12 03:00:00
```

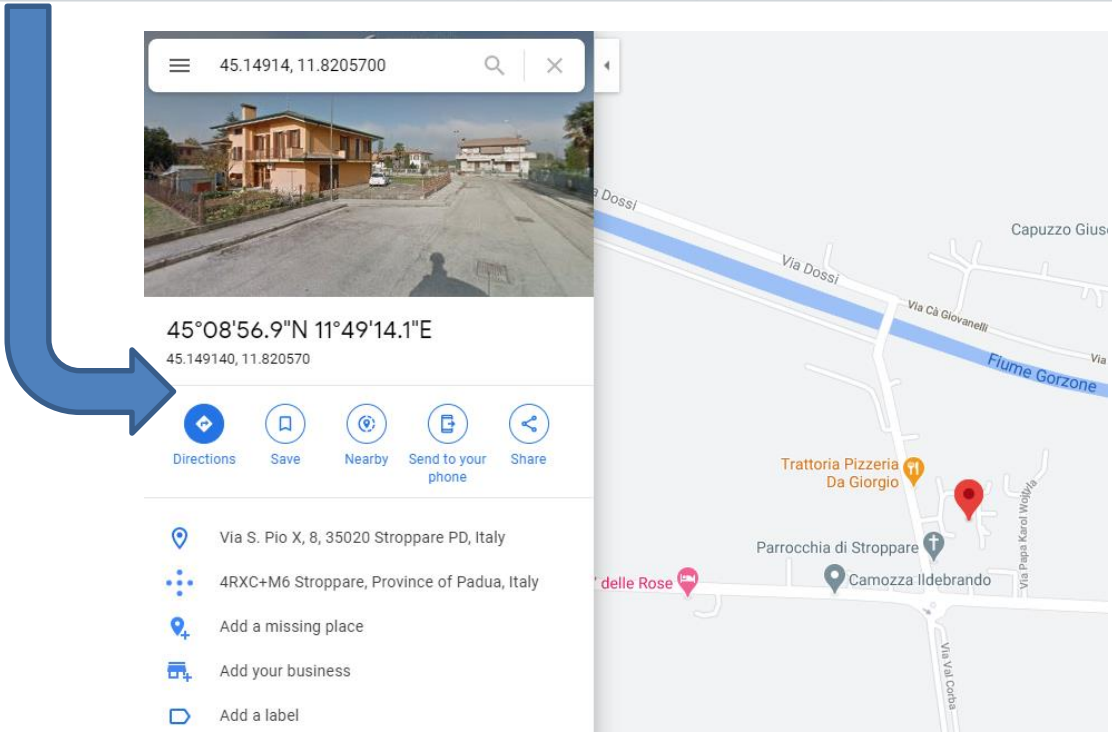
5. If you want just one result, use `geocoded_address <- opencage_forward(address, limit=1)`

Geocoding with *opencage* (3/3)

```
geocoded_address$results %>%
  select(geometry.lng, geometry.lat)
```

```
# A tibble: 9 x 2
  geometry.lng geometry.lat
         <dbl>        <dbl>
1       -0.161         51.5
2       -0.160         51.5
3       -0.162         51.5
4       -0.162         51.5
5       -0.162         51.5
6       -0.166         51.5
7       -0.161         51.5
8       -0.162         51.5
9       11.8           45.1
```

formatted	geometry.lat	geometry.lng	components.building
London Business School, Kent Terrace, London NW1 4RP, U...	51.52639	-0.1611472	*NA*
London Business School, Upper Montagu Street, London W...	51.52127	-0.1602583	*NA*
London Business School, Linhope Street, London NW1 6HT, ...	51.52488	-0.1624017	*NA*
London Business School, 27 Sussex Place, London NW1 4RG...	51.52680	-0.1621943	London Business School
London Business School, 17 Linhope Street, London NW1 6...	51.52466	-0.1617780	London Business School
London Business School, 25-27 Lorne Close, London NW8 7...	51.52785	-0.1659903	London Business School
London Business School, 15-17 Huntsworth Mews, London ...	51.52383	-0.1612441	London Business School
London Business School, Park Road, London NW1 6XU, Unit...	51.52612	-0.1619420	*NA*
Stroppare, Padua, Italy	45.14914	11.8205700	*NA*



Geocoding list of countries visited

05_countries_visited.R

```
20 googlesheets4::gs4_auth() # google sheets authorisation
21
22 #load countries_visited googlesheets
23 countries_visited <- read_sheet("https://docs.google.com/spreadsheets/d/14k4xrwrMRfabnyqQ2y_mT
24 ")
25
26 geocoded <- countries_visited %>%
27   mutate(
28     address_geo = purrr::map(country, opencage_forward, limit=1) # the beauty of purrr:map()
29   ) %>%
30   unnest_wider(address_geo) %>% # opencage returns a list, hence we unnest it...
31   unnest(results) %>% # look inside the results that opencage returns
32   rename(lat = geometry.lat, # rename latitude/longitude to lat/lng
33          lng = geometry.lng) %>%
34   select(country, lat, lng) # just select country, latitude, longitude
```

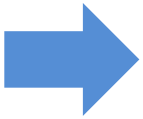
	country
1	Argentina
2	Austria
3	Belgium
4	Bulgaria
5	Canada
6	China
7	Cyprus
8	Czechia
9	Denmark
10	France
11	Germany
12	Greece
13	Italy
14	Liechtenstein
15	Mexico
16	Monaco
17	Nigeria

	country	lat	lng
1	Argentina	-34.996496	-64.967282
2	Austria	47.200034	13.199959
3	Belgium	50.640281	4.666715
4	Bulgaria	42.607397	25.485662
5	Canada	61.066692	-107.991707
6	China	35.000074	104.999927
7	Cyprus	34.982302	33.145128
8	Czechia	49.816700	15.474954
9	Denmark	55.670249	10.333328
10	France	46.603354	1.888334
11	Germany	51.083420	10.423447
12	Greece	38.995368	21.987713
13	Italy	42.638426	12.674297
14	Liechtenstein	47.141631	9.553153
15	Mexico	22.500048	-100.000038
16	Monaco	43.732349	7.427683
17	Nigeria	9.600036	7.999972

London Business School

Contents

- Visualising geospatial data
  - ggplot + the simple features ***sf*** package
  - *An interlude: functional programming-* ***purrr:map()***
  - Geocoding – Opencage
  - Adding data on maps: Points and Lines
  - GIS in R using the simple features ***sf*** package



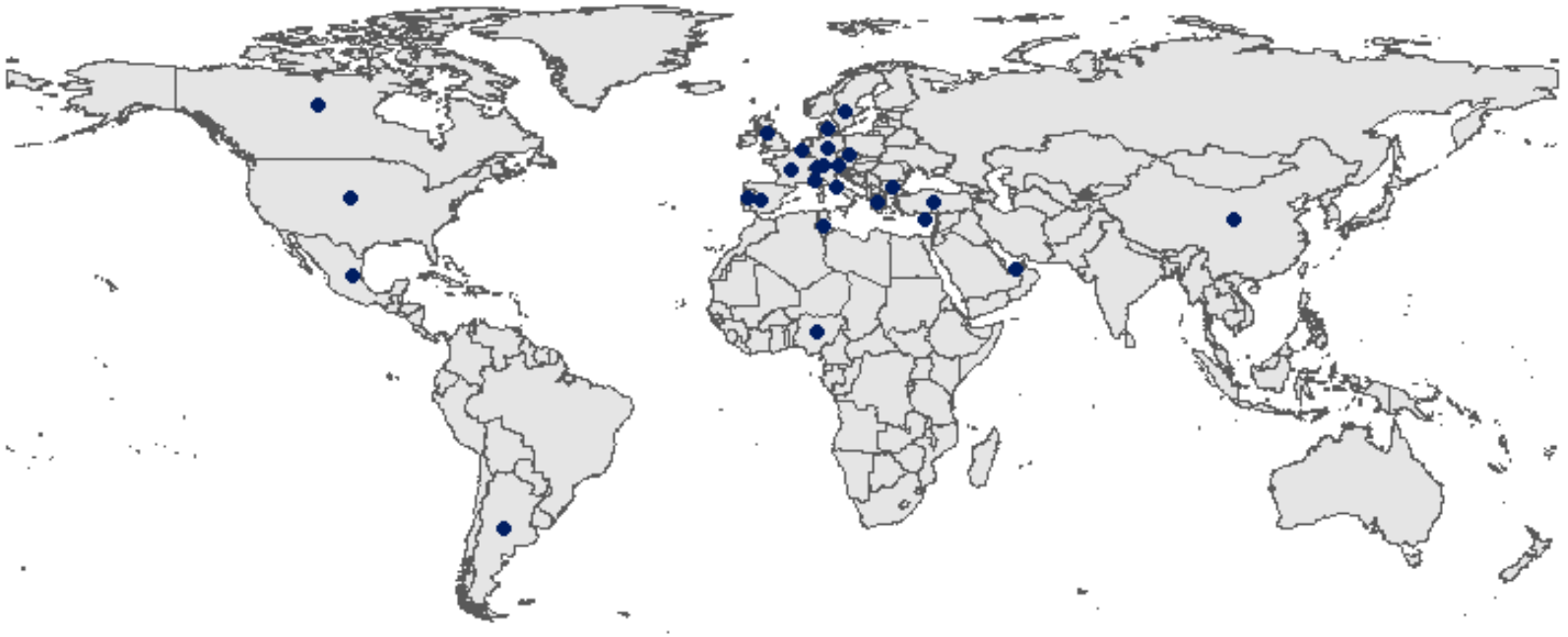
- Introduction to interactive graphs: Plotly and Shiny

London Business School

Plotting countries visited (1/2)

05_countries_visited.R

```
# we will use the rnatural earth package to get a medium resolution
# vector map of world countries excl. Antarctica
world <- ne_countries(scale = "medium", returnclass = "sf") %>%
  filter(name != "Antarctica")

st_geometry(world) # what is the geometry?
# CRS:            +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0

ggplot(data = world) +
  geom_sf() + # the first two lines just plot the world shapefile
  geom_point(data = geocoded, # then we add points
             aes(x = lng, y = lat),
             size = 2,
             colour = "#001e62") +
  theme_void()
```



Plotting countries visited (2/2)

05_countries_visited.R

```
world_visited <- left_join(world, geocoded,  by=c("admin" = "country" )) %>%
    mutate(visited = if_else (!is.na(lat), "visited", "not visited")
  )

ggplot(world_visited)+
  geom_sf(aes(fill=visited),
          show.legend = FALSE)+     # no legend
  scale_fill_manual(values=c('#f0f0f0', '#3182bd'))+
  coord_sf(datum = NA) +
  theme_void()+
  labs(title="Which countries have I travelled to?")+
  theme_ipsum_rc(grid="", strip_text_face = "bold") +
#  theme_ft_rc(grid="", strip_text_face = "bold") +
  NULL
```

London Business School

Adding lines to a map

06_london_flights.R

We will use the Eurostat package to get data on flights out of the UK

```
# Air passenger transport between the main airports of the United Kingdom
# and their main partner airports (routes data)
# https://ec.europa.eu/eurostat/web/products-datasets/-/avia_par_uk

flights_UK <- get_eurostat(id="avia_par_uk",
                           select_time = "M") #get monthly data


# filter passengers only
flights <- flights_UK %>%
  dplyr::filter(unit == "PAS") %>%
  mutate(
    origin = str_sub(airp_pr, 1, 7),
    destination = str_sub(airp_pr, -7),
    from = str_sub(origin, 4,7),
    to = str_sub(destination, 4,7),
    year = year(time),
    month_name = month(time, label = TRUE)
  )
```

```
london_airports <- c("UK_EGLL","UK_EGKK","UK_EGSS","UK_EGGW")

london_flights <- flights %>%
  dplyr::filter(origin %in% london_airports) %>%
  distinct()
```

```
londonflights2018 <- london_flights %>%
  filter(year==2018) %>%
  group_by(year, destination, origin, from, to) %>%
  summarise(totalpassengers = sum(values)) %>%
  arrange(desc(totalpassengers)) %>%

  #geocode origin (from_x, from_y)
  mutate(
    origin_geo = purrr::map(from, opencage_forward, limit=1)
  ) %>%
  unnest_wider(origin_geo) %>%
  unnest(results) %>%
  rename(from_y = geometry.lat,
         from_x = geometry.lng) %>%
  select(destination, from, to, totalpassengers, from_x, from_y) %>%

  #geocode destination (to_x, to_y)
  mutate(
    destination_geo = purrr::map(to, opencage_forward, limit=1)
  ) %>%
  unnest_wider(destination_geo) %>%
  unnest(results) %>%
  rename(to_y = geometry.lat,
         to_x = geometry.lng) %>%
  select(destination, from, to, totalpassengers, from_x, from_y, to_x, to_y)
```

London Business School

Adding lines to a map

06_london_flights.R

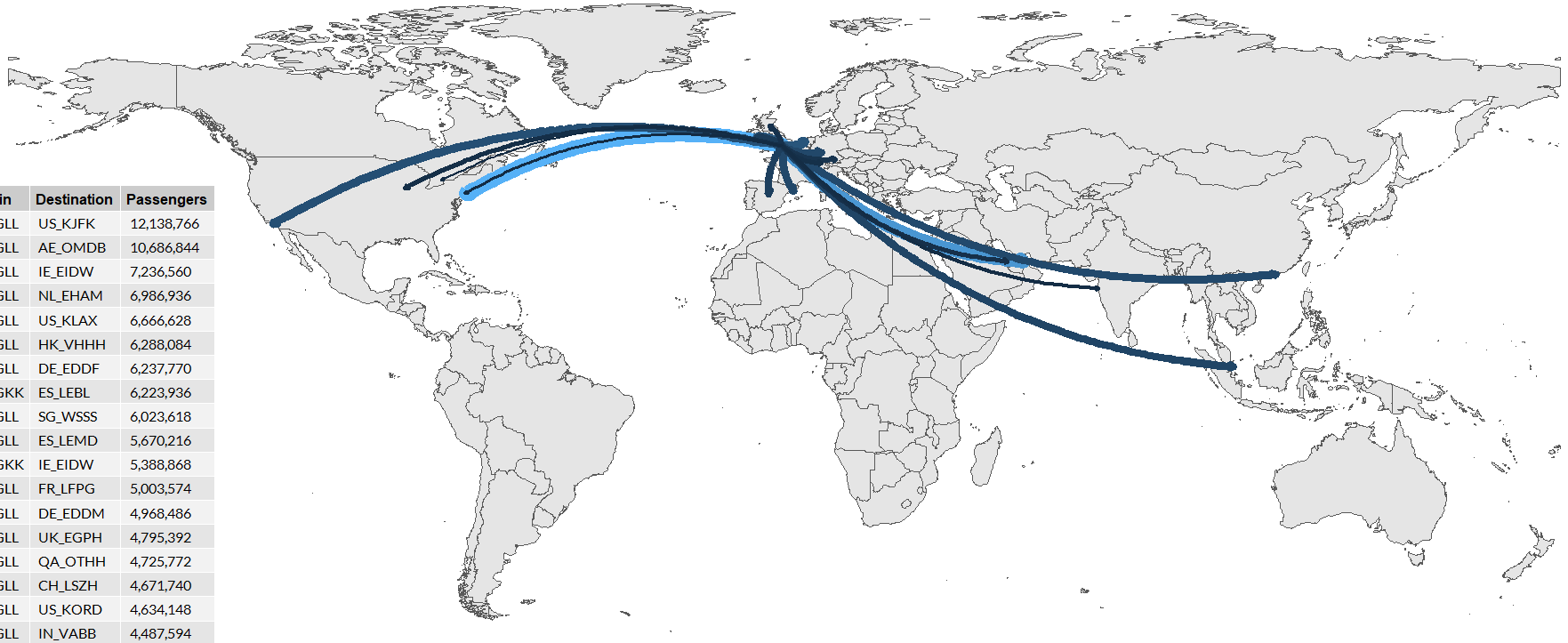
	year	origin	destination	from	to	totalpassengers	from_x	from_y	to_x	to_y
1	2018	UK_EGLL	US_KJFK	EGLL	KJFK	12138766	-0.4587801	51.46774	-73.7793734	40.642948
2	2018	UK_EGLL	AE_OMDB	EGLL	OMDB	10686844	-0.4587801	51.46774	55.3685409	25.251417
3	2018	UK_EGLL	IE_EIDW	EGLL	EIDW	7236560	-0.4587801	51.46774	-6.2475925	53.428803
4	2018	UK_EGLL	NL_EHAM	EGLL	EHAM	6986936	-0.4587801	51.46774	4.7414525	52.326401
5	2018	UK_EGLL	US_KLAX	EGLL	KLAX	6666628	-0.4587801	51.46774	-118.4213930	33.942167
6	2018	UK_EGLL	HK_VHHH	EGLL	VHHH	6288084	-0.4587801	51.46774	113.9173517	22.312599
7	2018	UK_EGLL	DE_EDDF	EGLL	EDDF	6237770	-0.4587801	51.46774	8.5249382	50.022944
8	2018	UK_EGKK	ES_LEBL	EGKK	LEBL	6223936	-0.1803876	51.15411	2.0790474	41.296944
9	2018	UK_EGLL	SG_WSSS	EGLL	WSSS	6023618	-0.4587801	51.46774	103.9880128	1.354884
10	2018	UK_EGLL	ES_LEMD	EGLL	LEMD	5670216	-0.4587801	51.46774	-3.5740806	40.494838
11	2018	UK_EGKK	IE_EIDW	EGKK	EIDW	5388868	-0.1803876	51.15411	-6.2475925	53.428803
12	2018	UK_EGLL	FR_LFPG	EGLL	LFPG	5003574	-0.4587801	51.46774	2.5710604	49.006875
13	2018	UK_EGLL	DE_EDDM	EGLL	EDDM	4968486	-0.4587801	51.46774	11.7780115	48.353767
14	2018	UK_EGLL	UK_EGPH	EGLL	EGPH	4795392	-0.4587801	51.46774	-3.3594791	55.950122
15	2018	UK_EGLL	QA_OTHH	EGLL	OTHH	4725772	-0.4587801	51.46774	51.6068054	25.274637
16	2018	UK_EGLL	CH_LSZH	EGLL	LSZH	4671740	-0.4587801	51.46774	8.5532047	47.463549
17	2018	UK_EGLL	US_KORD	EGLL	KORD	4634148	-0.4587801	51.46774	-87.9093214	41.977985
18	2018	UK_EGLL	IN_VABB	EGLL	VABB	4487594	-0.4587801	51.46774	72.8637032	19.090131
19	2018	UK_EGLL	US_KEWR	EGLL	KEWR	4376446	-0.4587801	51.46774	-74.1772549	40.689064
20	2018	UK_EGLL	CA_CYYZ	EGLL	CYYZ	4366566	-0.4587801	51.46774	-79.6291291	43.678524

Adding lines to a map

06_london_flights.R

Top 20 destinations for London flights in 2018

Data source: Eurostat , id=avia_par_uk



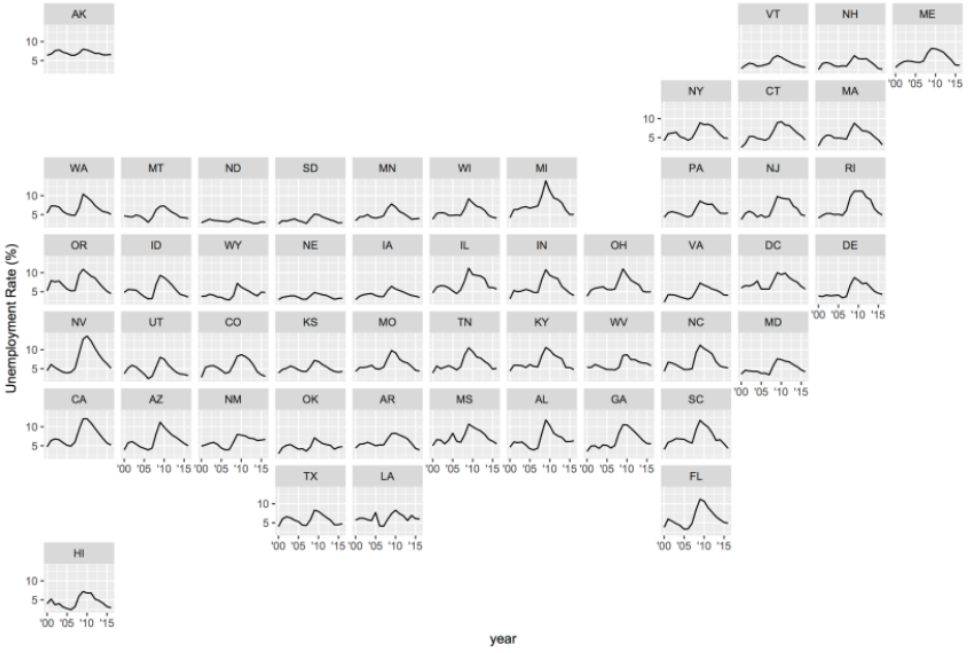
Origin	Destination	Passengers
UK_EGLL	US_KJFK	12,138,766
UK_EGLL	AE_OMDB	10,686,844
UK_EGLL	IE_EIDW	7,236,560
UK_EGLL	NL_EHAM	6,986,936
UK_EGLL	US_KLAX	6,666,628
UK_EGLL	HK_VHHH	6,288,084
UK_EGLL	DE_EDDF	6,237,770
UK_EGKK	ES_LEBL	6,223,936
UK_EGLL	SG_WSSS	6,023,618
UK_EGLL	ES_LEMD	5,670,216
UK_EGKK	IE_EIDW	5,388,868
UK_EGLL	FR_LFPG	5,003,574
UK_EGLL	DE_EDDM	4,968,486
UK_EGLL	UK_EGPH	4,795,392
UK_EGLL	QA_OTHH	4,725,772
UK_EGLL	CH_LSZH	4,671,740
UK_EGLL	US_KORD	4,634,148
UK_EGLL	IN_VABB	4,487,594
UK_EGLL	US_KEWR	4,376,446
UK_EGLL	CA_CYYZ	4,366,566

```
141 ggplot() +
142   geom_sf(data = world, size = 0.125) +
143   geom_curve(
144     data = londonflights2018 %>% head(20),
145     aes(x = from_x, y = from_y, xend = to_x, yend = to_y,
146         size= totalpassengers, colour = totalpassengers),
147     curvature = 0.2, arrow = arrow(length = unit(3, "pt"), type = "closed"),
148   )+
149   theme_void()+
```

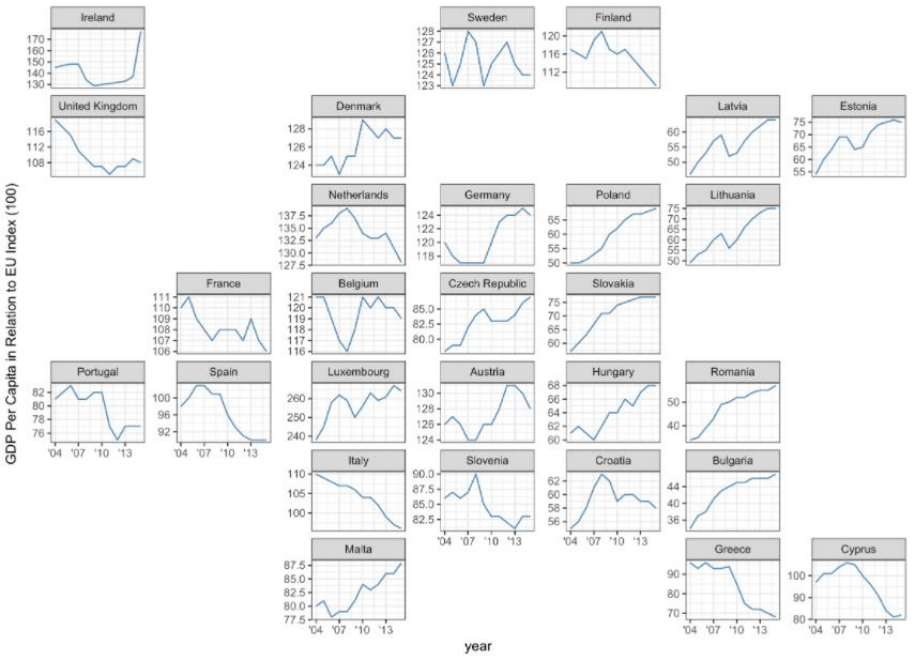
London Business School

geofacet::facet_geo()

```
ggplot(state_unemp, aes(year, rate)) +
  geom_line() +
  facet_geo(~ state, grid = "us_state_grid2") +
  scale_x_continuous(labels = function(x) paste0("'", substr(x, 3, 4))) +
  ylab("Unemployment Rate (%)")
```




```
ggplot(eu_gdp, aes(year, gdp_pc)) +
  geom_line(color = "steelblue") +
  facet_geo(~ name, grid = "eu_grid1", scales = "free_y") +
  scale_x_continuous(labels = function(x) paste0("'", substr(x, 3, 4))) +
  ylab("GDP Per Capita in Relation to EU Index (100)") +
  theme_bw()
```

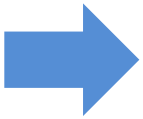


https://hafen.github.io/geofacet/

London Business School

Contents

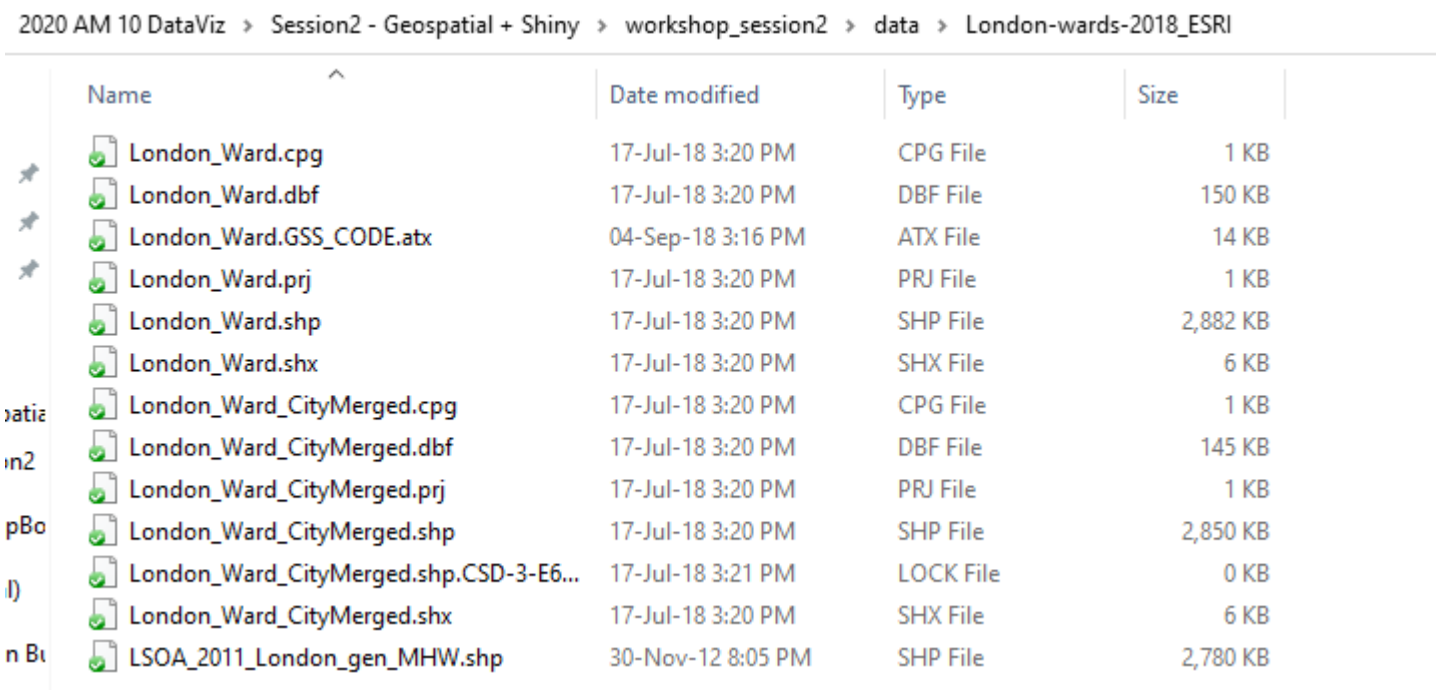
- Visualising geospatial data
  - ggplot + the simple features ***sf*** package
  - *An interlude: functional programming-* ***purrr:map()***
  - Geocoding – Opencage
  - Adding data on maps: Points and Lines
  - GIS in R using the simple features ***sf*** package



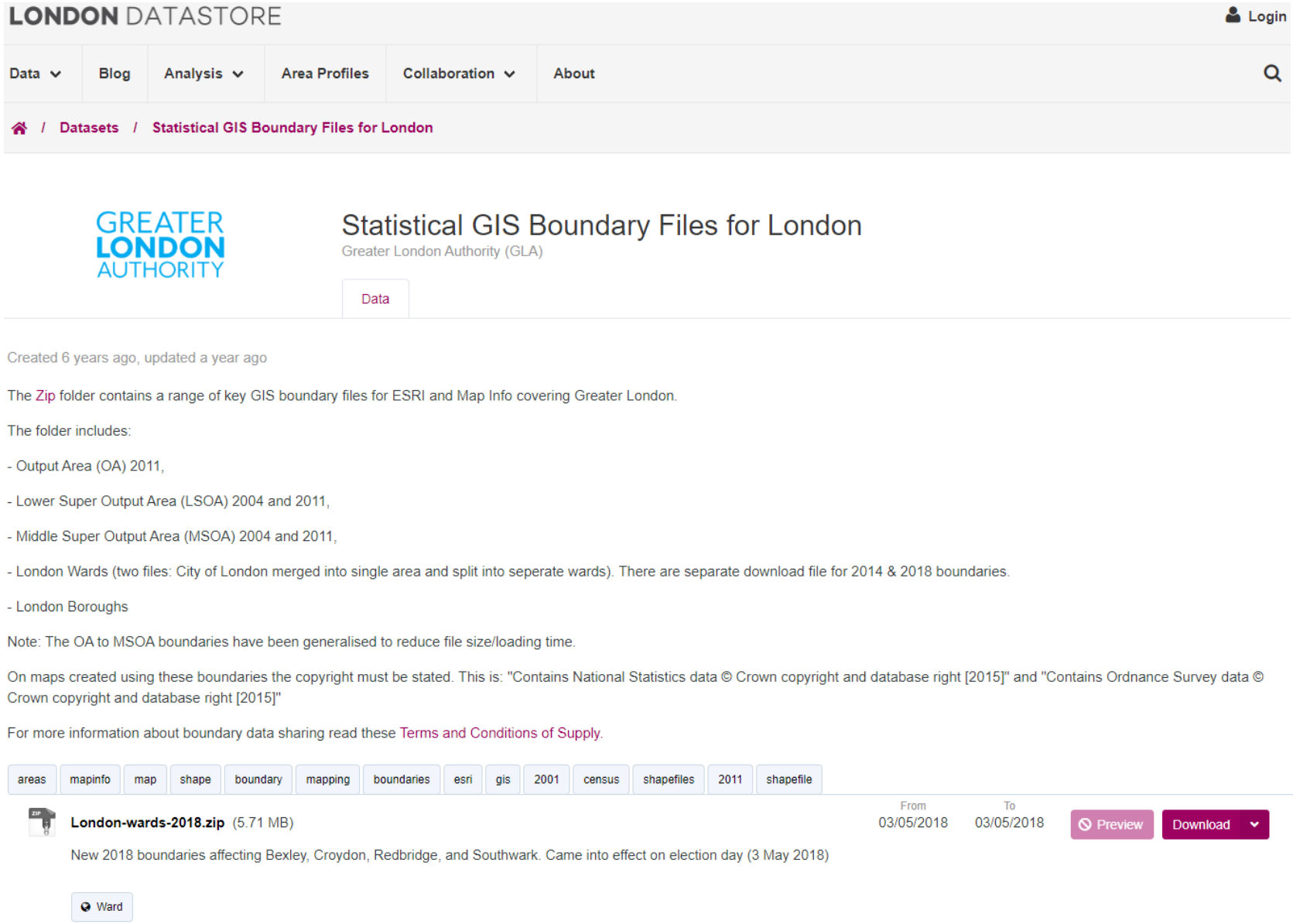
- Introduction to interactive graphs: Plotly and Shiny

For GIS analysis, you need **shapefiles**

- Shapefiles are not like your normal CSV or data files
- They come usually as a zipped file and you need all of them, not just the **.shp** file
- Below is the shapefile with all of London Wards



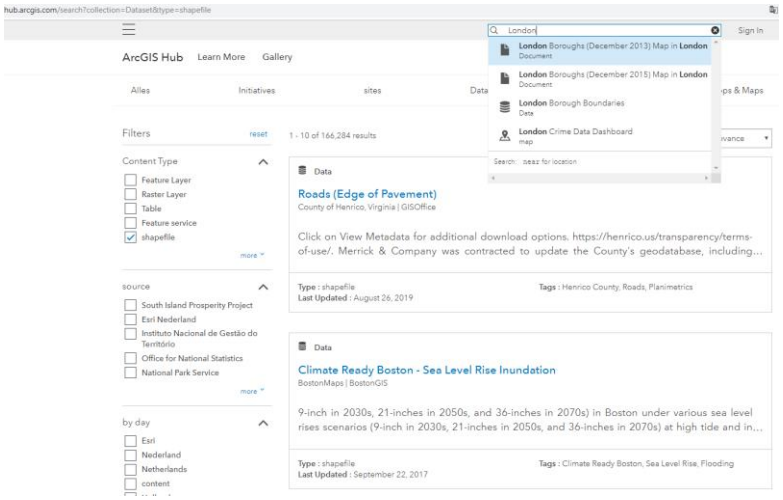
London GIS Boundary shapefile



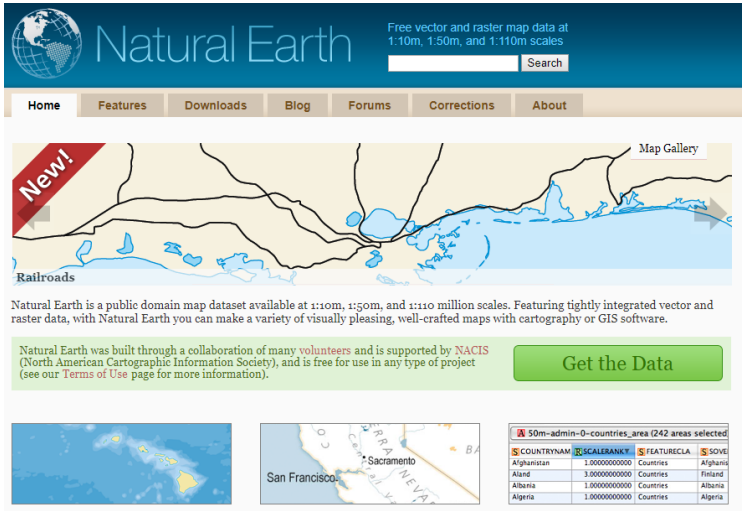
https://data.london.gov.uk/dataset/statistical-gis-boundary-files-london

Where do we find vector (or shape) files?

https://hub.arcgis.com/search?collection=Dataset



https://www.naturalearthdata.com/
Used with the ***rnaturalearth*** package
Good source of international maps



Scale (resolution) of maps

The **m** refers to millions, not metres!

Large scale data, 1:10m

Cultural | Physical | Raster

The most detailed. Suitable for making zoomed-in maps of countries and regions. Show the world on a large wall poster.

1:10,000,000
1″ = 158 miles
1 cm = 100 km

Medium scale data, 1:50m

Cultural | Physical | Raster

Suitable for making zoomed-out maps of countries and regions. Show the world on a tabloid size page.

1:50,000,000
1″ = 790 miles
1 cm = 500 km

Small scale data, 1:110m

Cultural | Physical

Suitable for schematic maps of the world on a postcard or as a small locator globe.

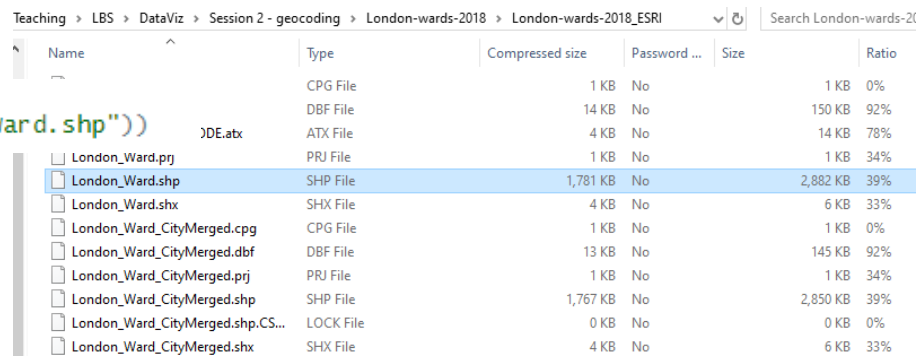
1:110,000,000
1″ = 1,736 miles
1 cm = 1,100 km

https://www.naturalearthdata.com/downloads/

Reading shapefiles

07_mapping_London_stop_search.R

```
#use sf::read_sf() to read in London Wards shapefile
london_wards_sf <- read_sf(here("London-wards-2018_ESRI","London_Ward.shp"))
```




```
> glimpse(london_wards_sf)
Rows: 657
Columns: 7
$ NAME       <chr> "Chessington South", "Tolworth and Hook Rise", "Berrylands", "Alexandra", "Beverley", "Coombe Hill"...
$ GSS_CODE   <chr> "E05000405", "E05000414", "E05000401", "E05000400", "E05000402", "E05000406", "E05000404", "E050004...
$ DISTRICT   <chr> "Kingston upon Thames", "Kingston upon Thames", "Kingston upon Thames", "Kingston upon Thames", "Ki...
$ LAGSSCODE  <chr> "E09000021", "E09000021", "E09000021", "E09000021", "E09000021", "E09000021", "E09000021", "E090000...
$ HECTARES   <dbl> 755.173, 259.464, 145.390, 268.506, 187.821, 442.170, 192.980, 166.482, 180.016, 137.578, 192.034, ...
$ NONLD_AREA <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ...
$ geometry   <POLYGON [m]> POLYGON ((516401.6 160201.8..., POLYGON ((519553 164295.6, ..., POLYGON ((518107.5 167303.4...
>
> # what type of geometry does our shapefile have?
> st_geometry(london_wards_sf)
Geometry set for 657 features
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: 503568.2 ymin: 155850.8 xmax: 561957.5 ymax: 200933.9
projected CRS:  OSGB 1936 / British National Grid
First 5 geometries:
POLYGON ((516401.6 160201.8, 516407.3 160210.5,...
POLYGON ((519553 164295.6, 519508.1 164322.9, 5...
POLYGON ((518107.5 167303.4, 518114.3 167307.5,...
POLYGON ((520336.7 165105.5, 520332.2 165106.6,...
POLYGON ((521201.2 169275.5, 521204.3 169274.7,...
```

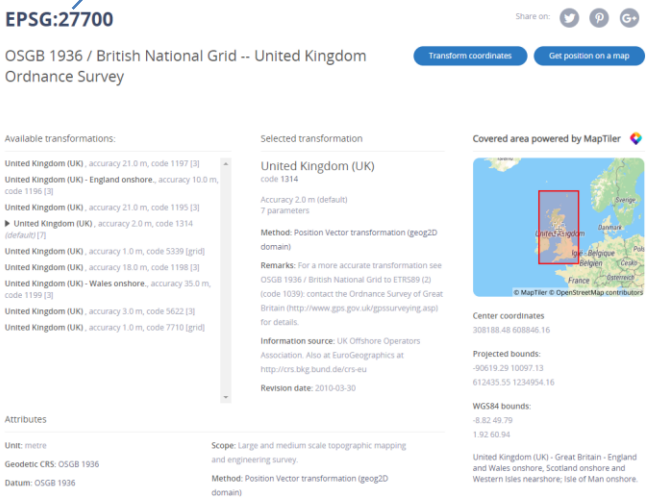
⟸ Magic stuff

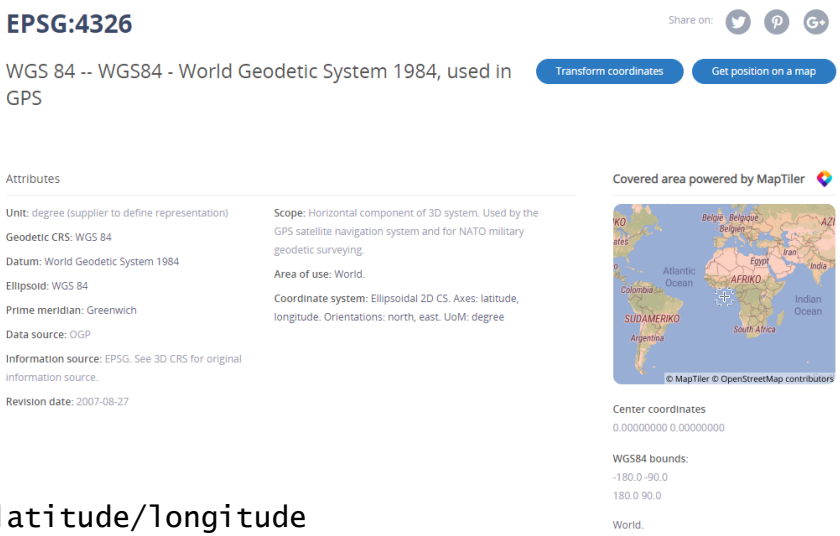
If we use multiple dataframes, they should all have the same CRS

To simplify things, we will use the EPSG:4326, or the World Geodetic System 1984 used in GPS, defined by a *latitude/longitude* pair

Transforming CRS in shapefiles

```
> london_wards_sf$geometry
Geometry set for 657 features
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: 503568.2 ymin: 155850.8 xmax: 561957.5 ymax: 200933.9
epsg (SRID):    27700
proj4string:    +proj=tmerc +lat_0=49 +lon_0=-2 +k=0.9996012717 +x_0=400000 +y_0=-100000 +ellps=airy +towgs84=446.448,-125.157,
2.06,0.15,0.247,0.842,-20.489 +units=m +no_defs
First 5 geometries:
POLYGON ((516401.6 160201.8, 516407.3 160210.5,...
POLYGON ((519553 164295.6, 519508.1 164322.9, 5...
POLYGON ((518107.5 167303.4, 518114.3 167307.5,...
POLYGON ((520336.7 165105.5, 520332.2 165106.6,...
POLYGON ((521201.2 169275.5, 521204.3 169274.7,...
```






```
# Transform CRS to 4326, or pairs of latitude/longitude
london_wgs84 <-  london_wards_sf %>%
  st_transform(4326)
```

```
> london_wgs84$geometry
Geometry set for 657 features
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: -0.5103751 ymin: 51.28676 xmax: 0.3340155 ymax: 51.69187
epsg (SRID):    4326
proj4string:    +proj=longlat +datum=WGS84 +no_defs
First 5 geometries:
POLYGON ((-0.3306791 51.32901, -0.3305944 51.32...
POLYGON ((-0.2840949 51.36515, -0.2847304 51.36...
POLYGON ((-0.3038497 51.39249, -0.3037506 51.39...
POLYGON ((-0.2725691 51.37227, -0.2726334 51.37...
POLYGON ((-0.2587332 51.40956, -0.2586889 51.40...
```

https://epsg.io/

London Business School

Convert dataframe to an *sf* object

07_mapping_London_stop_search.R

```
> glimpse(sep20_offence)
Rows: 3,687
Columns: 15
$ type                                <chr> "Person search", "Person search", "Person and Vehicle search", "Perso...
$ date                                <dttm> 2020-08-31 23:15:00, 2020-08-31 23:22:00, 2020-08-31 23:50:00, 2020-...
$ part_of_a_policing_operation        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,...
$ policing_operation                  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ lat                                 <dbl> 51.50923, 51.53663, 51.53160, 51.55719, 51.49208, 51.52502, 51.52425,...
$ lng                                 <dbl> -0.117576, 0.034527, 0.043214, -0.335518, 0.054987, -0.035504, 0.0396...
$ gender                              <chr> "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male...
$ age_range                           <fct> 10-17, 18-24, 25-34, 18-24, 18-24, 25-34, 10-17, 18-24, over 34, 10-1...
$ self_defined_ethnicity              <chr> "Black/African/Caribbean/Black British - Caribbean", "Asian/Asian Bri...
$ officer_defined_ethnicity           <fct> Black, Asian, Asian, Asian, Black, Asian, White, Black, Black, Black,...
$ legislation                         <chr> "Police and Criminal Evidence Act 1984 (section 1)", "Misuse of Drugs...
$ object_of_search                    <fct> Stolen goods, Controlled drugs, Controlled drugs, Controlled drugs, O...
$ outcome                             <chr> "Arrest", "Community resolution", "Arrest", "Arrest", "Arrest", "Arre...
$ outcome_linked_to_object_of_search  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ removal_of_more_than_just_outer_clothing <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

Turn any dataframe to sf with **st_as_sf()**

```
71  # NB: make sure to transform to a  common CRS.
72  # Here we retrieve and apply the CRS of london_wgs84
73  sep20_offence_sf <-  st_as_sf(sep20_offence,
74                                coords=c('lng', 'lat'),
75                                crs=st_crs(london_wgs84))
76
77  # Alternatively, you can explicitly set the CRS to, e.g, 4326, or WGS84
78  sep20_offence_sf <-  st_as_sf(sep20_offence,
79                                coords=c('lng', 'lat'),
80                                crs=4326)
```

London Business School

sf::st_as_sf()

```
> glimpse(sep20_offence)
Rows: 3,687
Columns: 15
$ type                                <chr> "Person search", "Person search", "Person and Vehicle search", "Perso...
$ date                                <dttm> 2020-08-31 23:15:00, 2020-08-31 23:22:00, 2020-08-31 23:50:00, 2020-...
$ part_of_a_policing_operation        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,...
$ policing_operation                  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ lat                                 <dbl> 51.50923, 51.53663, 51.53160, 51.55719, 51.49208, 51.52502, 51.52425,...
$ lng                                 <dbl> -0.117576, 0.034527, 0.043214, -0.335518, 0.054987, -0.035504, 0.0396...
$ gender                              <chr> "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male...
$ age_range                           <fct> 10-17, 18-24, 25-34, 18-24, 18-24, 25-34, 10-17, 18-24, over 34, 10-1...
$ self_defined_ethnicity              <chr> "Black/African/Caribbean/Black British - Caribbean", "Asian/Asian Bri...
$ officer_defined_ethnicity           <fct> Black, Asian, Asian, Asian, Black, Asian, White, Black, Black, Black,...
$ legislation                         <chr> "Police and Criminal Evidence Act 1984 (section 1)", "Misuse of Drugs...
$ object_of_search                    <fct> Stolen goods, Controlled drugs, Controlled drugs, Controlled drugs, O...
$ outcome                             <chr> "Arrest", "Community resolution", "Arrest", "Arrest", "Arrest", "Arre...
$ outcome_linked_to_object_of_search  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ removal_of_more_than_just_outer_clothing <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
```

```
> glimpse(sep20_offence_sf)
Rows: 3,687
Columns: 14
$ type                                <chr> "Person search", "Person search", "Person and Vehicle search", "Perso...
$ date                                <dttm> 2020-08-31 23:15:00, 2020-08-31 23:22:00, 2020-08-31 23:50:00, 2020-...
$ part_of_a_policing_operation        <lgl> FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE, FALSE,...
$ policing_operation                  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ gender                              <chr> "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male", "Male...
$ age_range                           <fct> 10-17, 18-24, 25-34, 18-24, 18-24, 25-34, 10-17, 18-24, over 34, 10-1...
$ self_defined_ethnicity              <chr> "Black/African/Caribbean/Black British - Caribbean", "Asian/Asian Bri...
$ officer_defined_ethnicity           <fct> Black, Asian, Asian, Asian, Black, Asian, White, Black, Black, Black,...
$ legislation                         <chr> "Police and Criminal Evidence Act 1984 (section 1)", "Misuse of Drugs...
$ object_of_search                    <fct> Stolen goods, Controlled drugs, Controlled drugs, Controlled drugs, O...
$ outcome                             <chr> "Arrest", "Community resolution", "Arrest", "Arrest", "Arrest", "Arre...
$ outcome_linked_to_object_of_search  <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ removal_of_more_than_just_outer_clothing <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N...
$ geometry                            <POINT [°]> POINT (-0.117576 51.50923), POINT (0.034527 51.53663), POINT (0...
```

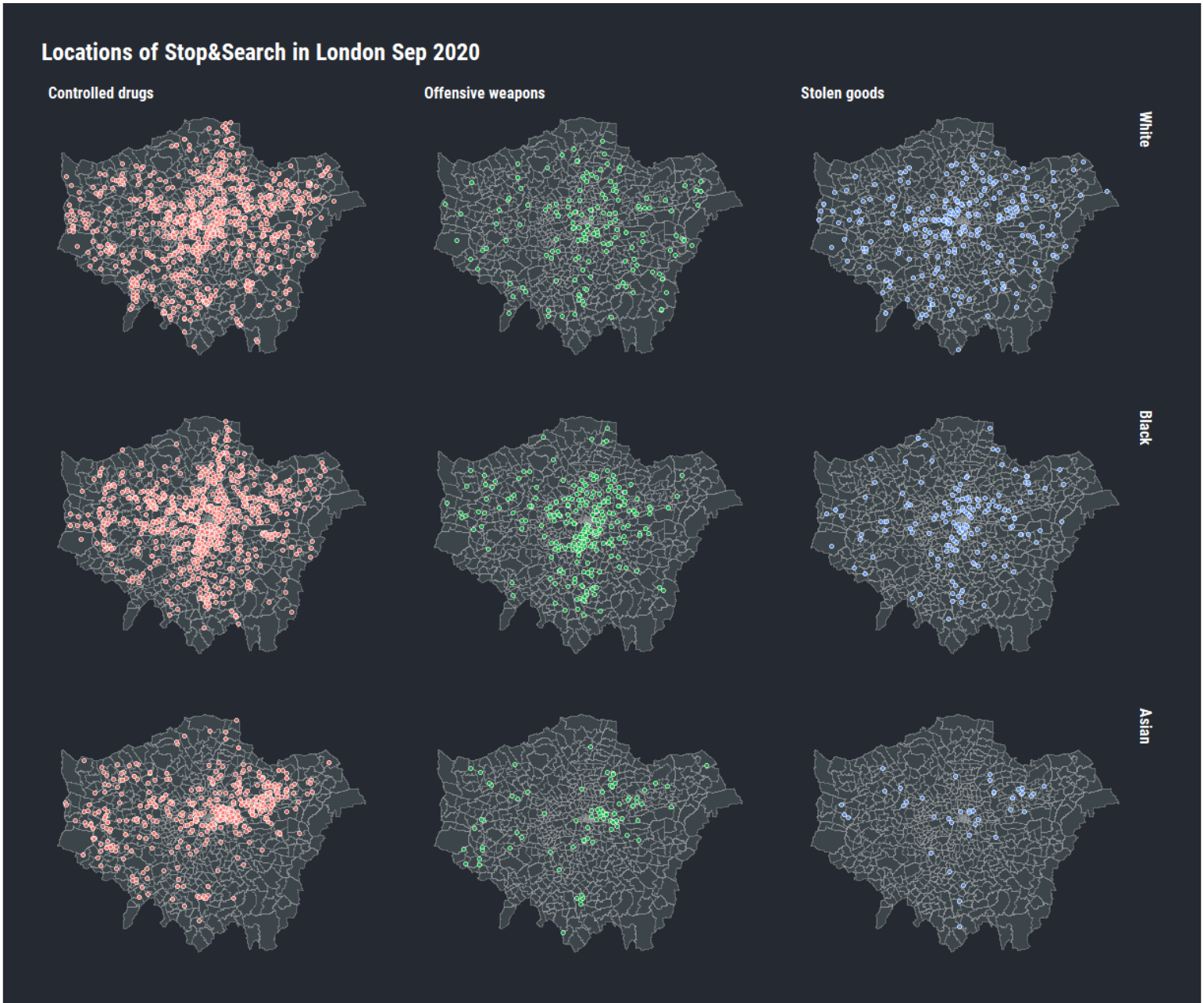
Adding points to polygons

07_mapping_London_stop_search.R

```
85  ggplot() +
86    # draw polygons from London wards shapefile
87    geom_sf(data = london_wgs84, fill = "#3B454A", size = 0.125, colour = "#b2b2b277") +
88
89    # add points from stop-and-search shapefile
90    geom_sf(
91      data = sep20_offence_sf, aes(fill = object_of_search),
92      color = "white", size = 1.5, alpha = 0.7, shape = 21,
93      show.legend = FALSE
94    ) +
95    theme_minimal()+
96    coord_sf(datum = NA) + #remove coordinates
97    facet_wrap(~object_of_search) +
98    labs(title = "Locations of Stop&Search in London Sep 2020") +
99    hrbrthemes::theme_ft_rc(grid="", strip_text_face = "bold") +
100   theme(axis.text = element_blank()) +
101   theme(strip.text = element_text(color = "white"))+
102   NULL
```

London Business School

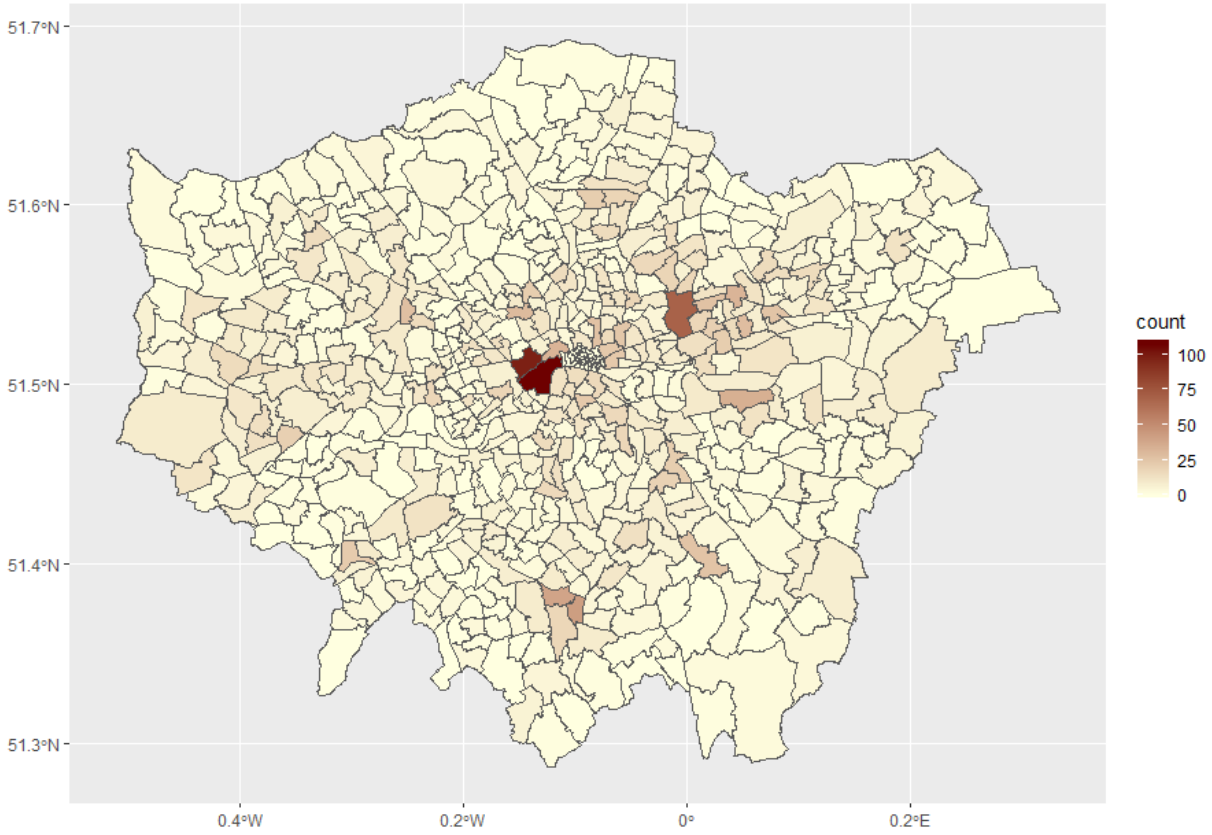
facet_grid(officer_defined_ethnicity ~ object_of_search)



sf for all GIS operations

Besides drawing maps, adding points, and calculating distances between points, we can count observations inside a given polygon (e.g., ward, county, country, etc.)

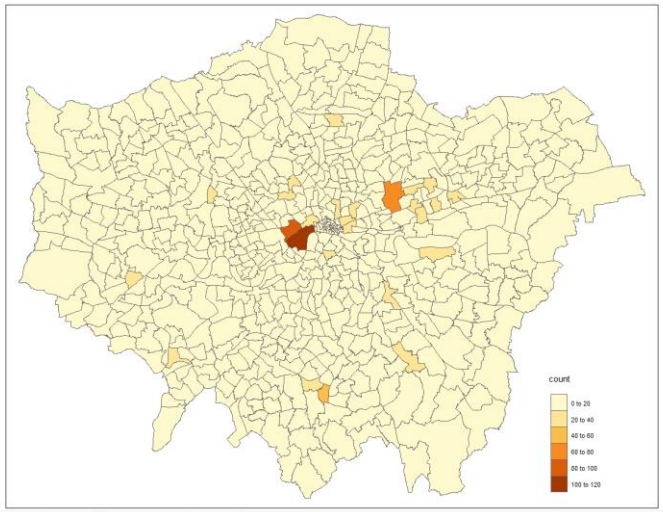
```
125 # Count how many S&S happened inside each ward
126 london_wgs84 <- london_wgs84 %>%
127   mutate(count = lengths(
128     st_contains(london_wgs84,
129                 sep20_offence_sf)))
130
131
132 ggplot(data = london_wgs84, aes(fill = count)) +
133    geom_sf() +
134    scale_fill_gradient(low = "#ffffe0", high = "#6e0000")
```



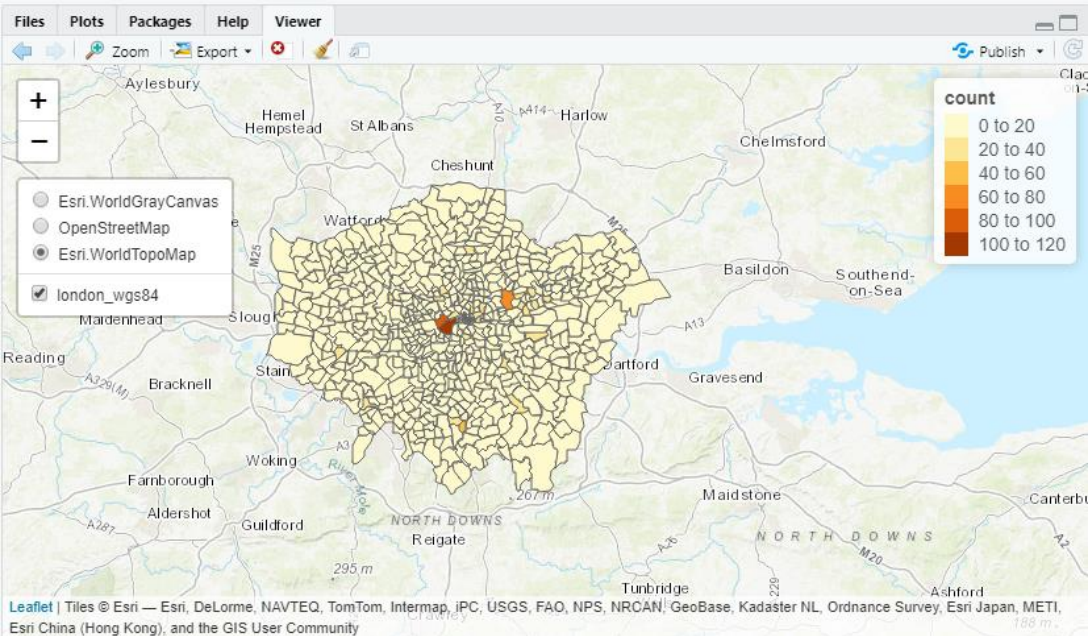
Static and interactive maps with *tmap*

```
london_wgs84 <- london_wgs84 %>%
  mutate(count = lengths(
    st_contains(london_wgs84,
                sep19_offence_sf)))
```

```
> glimpse(london_wgs84)
Observations: 657
Variables: 8
$ NAME       <chr> "Chessington South", "Tolworth and Hook Rise", "Berrylands", "Alexandra", "Beverley", "Coor
$ GSS_CODE   <chr> "E05000405", "E05000414", "E05000401", "E05000400", "E05000402", "E05000406", "E05000404",
$ DISTRICT   <chr> "Kingston upon Thames", "Kingston upon Thames", "Kingston upon Thames", "Kingston upon Thar
$ LAGSSCODE  <chr> "E09000021", "E09000021", "E09000021", "E09000021", "E09000021", "E09000021", "E09000021",
$ HECTARES   <dbl> 755.173, 259.464, 145.390, 268.506, 187.821, 442.170, 192.980, 166.482, 180.016, 137.578, 1
$ NONLD_AREA <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
$ geometry   <POLYGON [°]> POLYGON ((-0.3306791 51.329..., POLYGON ((-0.2840949 51.365..., POLYGON ((-0.30384
$ count      <int> 0, 3, 5, 2, 3, 9, 2, 0, 4, 6, 22, 3, 9, 0, 0, 2, 5, 4, 17, 1, 0, 2, 1, 0, 8, 1, 4, 3, 2, 39
```




```
tmap::tmap_mode("plot")
tmap::tm_shape(london_wgs84) +
  tm_polygons("count")
```




```
tmap::tmap_mode("view")
tmap::tm_shape(london_wgs84) +
  tm_polygons("count")
```

London Business School

osmdata

docs.ropensci.org/osmdata/index.html

osmdata v0.1.4.002 | Search... | Home

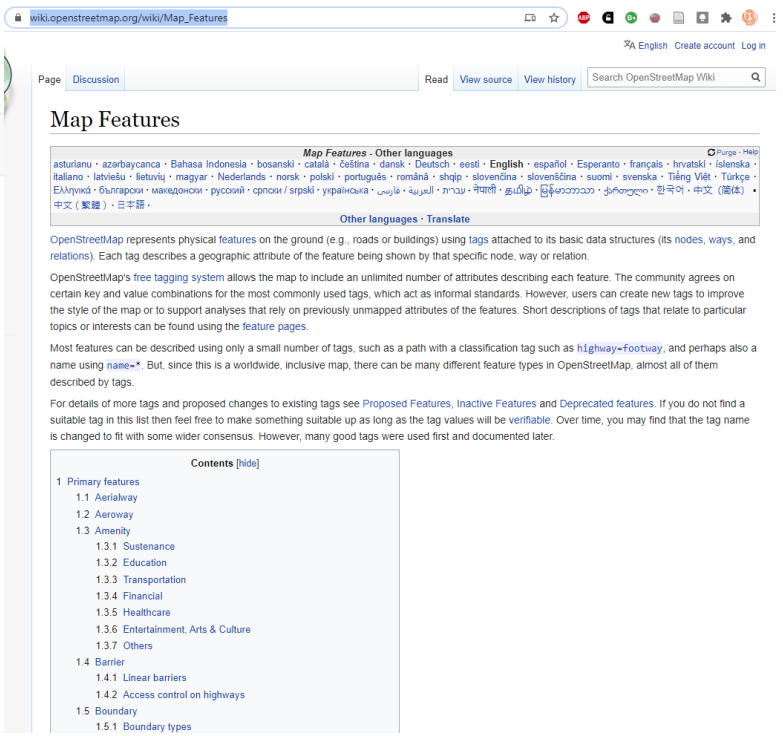
osmdata

`osmdata` is an R package for accessing the data underlying OpenStreetMap (OSM), delivered via the Overpass API. (Other packages such as `OpenStreetMap` can be used to download raster tiles based on OSM data.) Overpass is a read-only API that extracts custom selected parts of OSM data. Data can be returned in a variety of formats, including as Simple Features (`sf`), Spatial (`sp`), or Silicate (`sc`) objects. The package is designed to allow access to small-to-medium-sized OSM datasets (see `geofabrik` for an approach for reading-in bulk OSM data extracts).

Installation

To install latest CRAN version:

```
install.packages("osmdata")
```



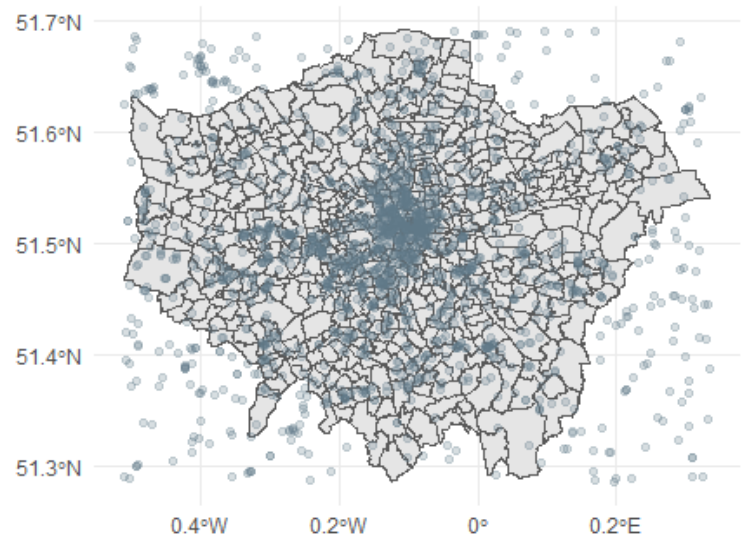
https://docs.ropensci.org/osmdata/index.html
https://wiki.openstreetmap.org/wiki/Map_Features

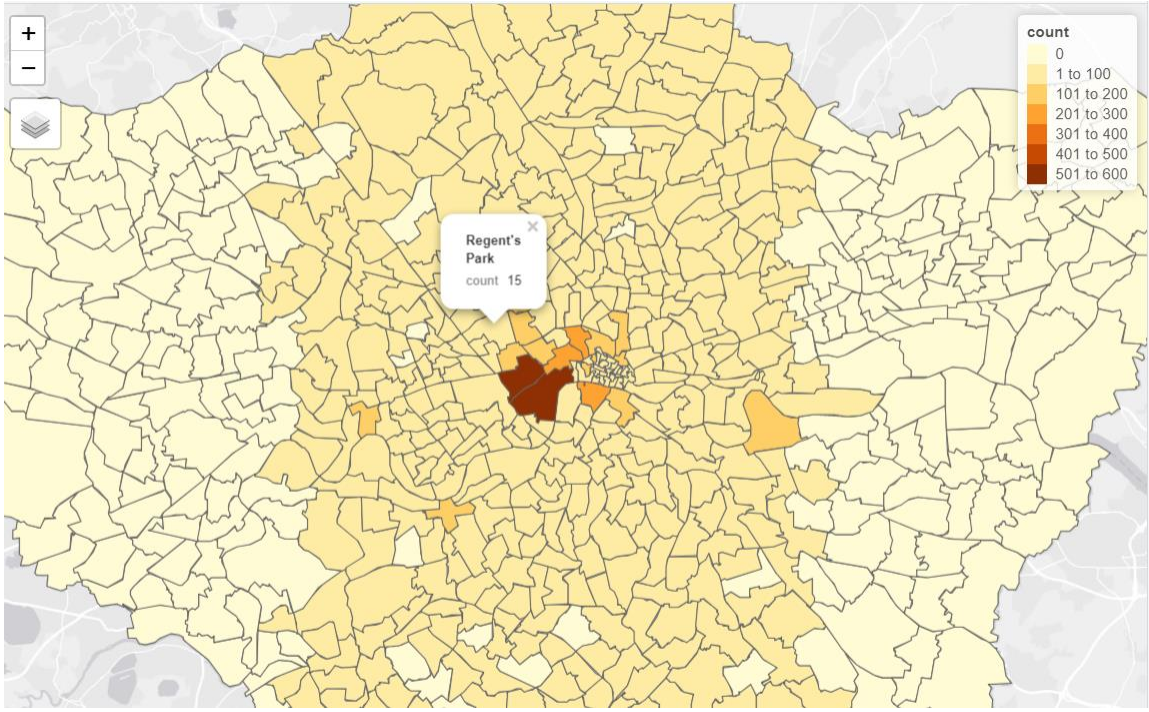
London Business School

Osmdata

08_osm_pubs.R

```r
library(osmdata)
library(here)
library(tidyverse)
library(sf)

pubs_osm <- opq ("London UK") %>%
  add_osm_feature(key = "amenity", value = "pub") %>%
  # return the data in Simple Features (sf) format
  osmdata_sf()

pubs <- pubs_osm$osm_points %>%
  drop_na(name) # drop pubs with  no name

st_geometry(pubs)
# geographic CRS: WGS 84
```



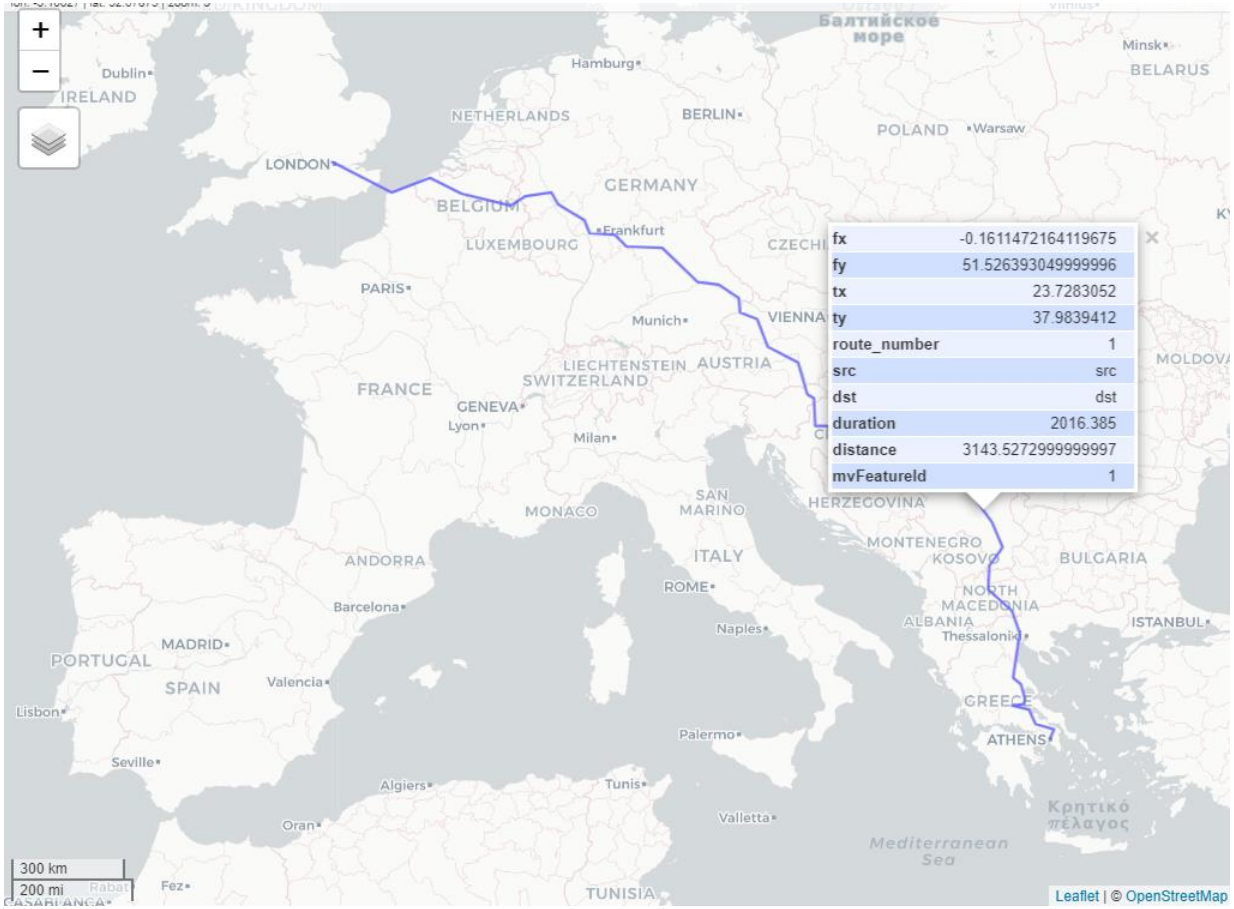


https://docs.ropensci.org/osmdata/index.html

OSM Routing

08_osm_routing.R

```
library(osrm)
library(osmdata)
library(stplanr)
library(mapview)

#> Data: (c) OpenStreetMap contributors, ODbL 1.0
#> Routing: OSRM - http://project-osrm.org/

trip <- route(
  from = "London Business School",
  to = "Athens",
  route_fun = osrmRoute,
  returnclass = "sf"
)


#> Most common output is sf
mapview::mapview(trip)
```



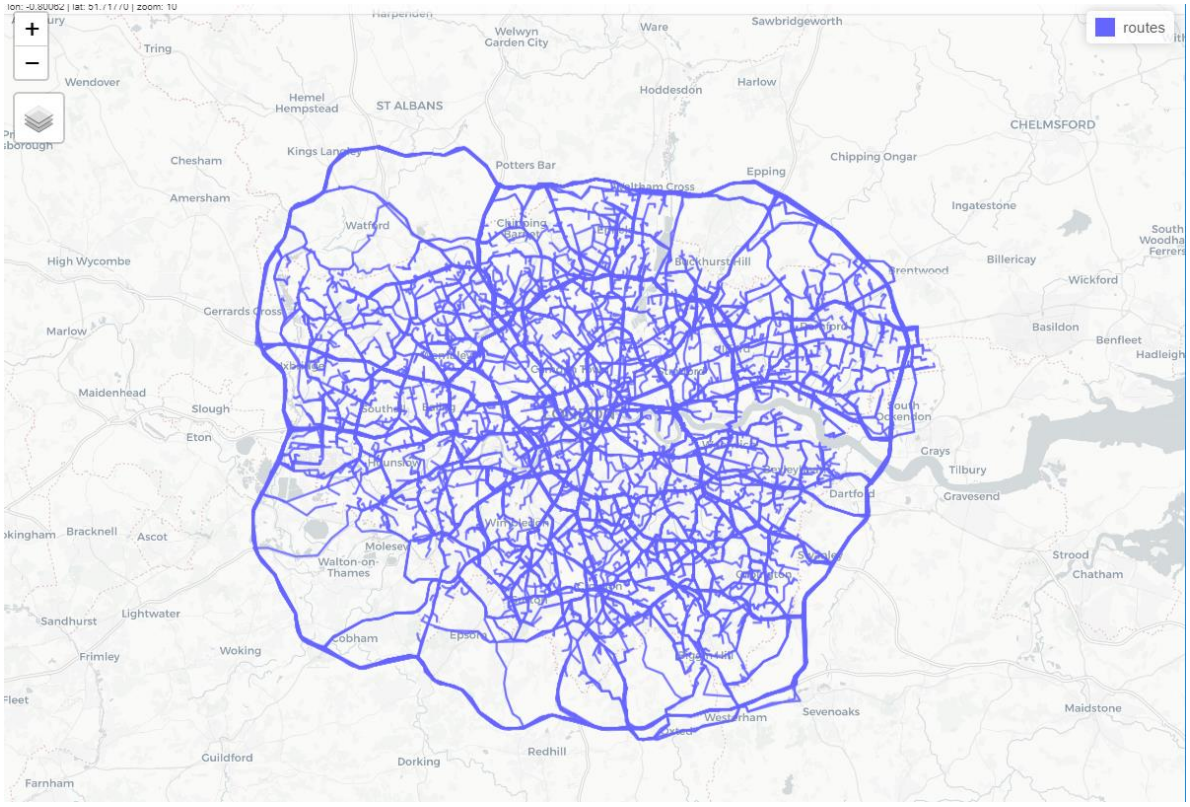
https://docs.ropensci.org/osmdata/index.html

London Business School

OSM Routing

08_osm_find_major_routes.R

```r
## find the main streets of a city:
# do random 1000 x 1000 routes and plot thickness of overlapping lines

library(sf)
library(dplyr)
library(osrm)
library(stplanr)
library(leaflet)
library(htmltools)
library(mapview)
library(viridis)

# get 1000 random routings

#use sf::read_sf() to read in London Wards shapefile
london_wards_sf <- read_sf(here("data/London-wards-2018_ESRI/London_Ward.shp"))

# transfrom CRS to 4326, or pairs of latitude/longitude numbers
london <-  london_wards_sf %>%
  st_transform(4326) # transform CRS to WGS84, latitude/longitude

# pick 100 random starting and ending points
start_points <- st_sample(london,size=1000) %>% st_as_sf
end_points <- st_sample(london,size=1000) %>% st_as_sf

# the routing will take a few minutes, as it needs to find 1000 routes
routes <- route(from = start_points,
                to = end_points,
                route_fun = osrmRoute,
                returnclass = "sf")

# Just do a quick visualisation of the 1000 routes
mapview(routes)
```



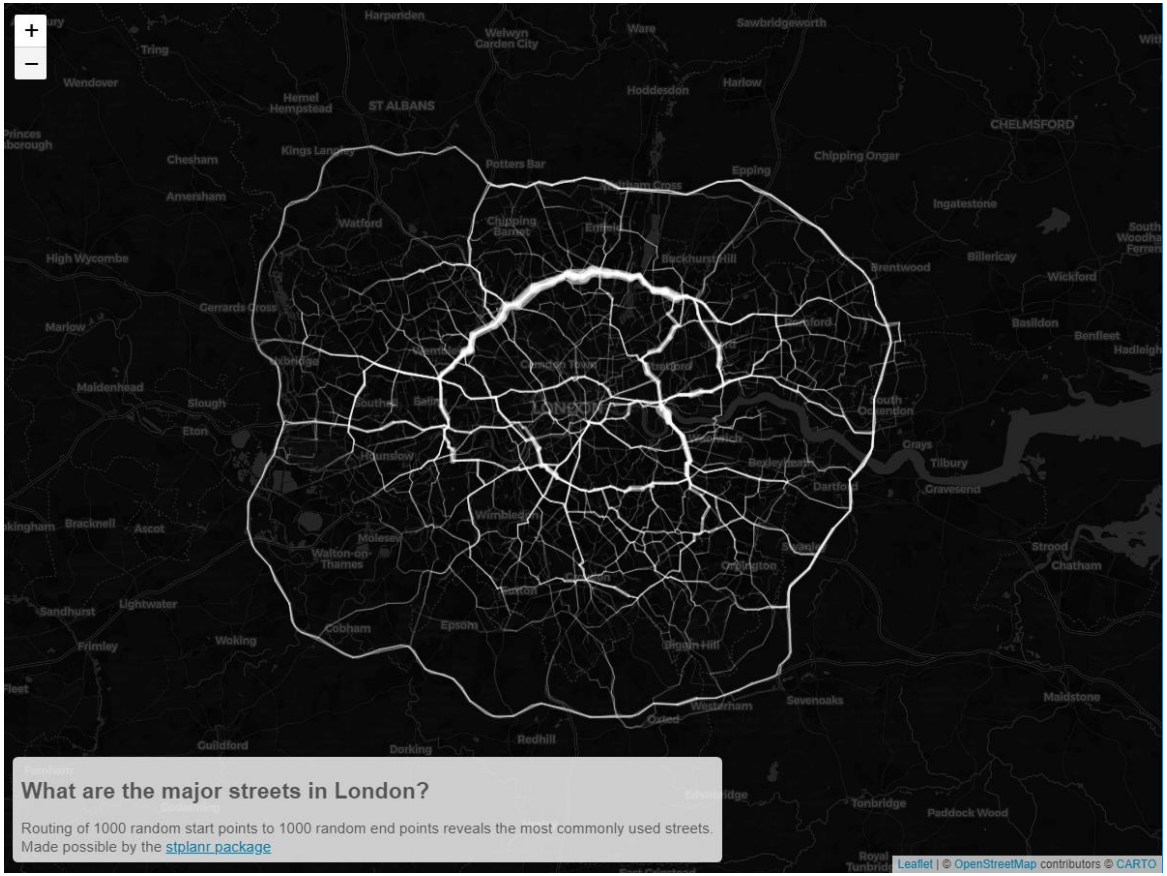
https://github.com/ropensci/stplanr

OSM Routing

08_osm_find_major_routes.R

```
routes <- routes %>%
  mutate(count = 1) # add a number that keeps track of each unique path

# stplanr::overline() takes a series of overlapping lines and converts them into a single route network.
overlapping_segments <- overline(routes,
                                 attrib = "count") # attrib: column names in sl to be aggregated
```

	count	geometry
47	73	c(-0.118839994072914, -0.109019994735718, 51.612430572...
46	58	c(-0.192819997668266, -0.170059993863106, 51.590099334...
45	55	c(-0.109019994735718, -0.0967399999499321, 51.61224746...
44	54	c(-0.0070699998177588, 0.000189999991562217, 51.505630...
43	52	c(-0.0967399999499321, -0.0870599970221519, -0.0770699...
42	51	c(-0.213129997253418, -0.207649990916252, 51.579769134...
42.1	51	c(-0.00122999993618578, 0.0128399999812245, 51.5999298...
41	48	c(-0.170059993863106, -0.148149996995926, 51.602798461...
40	47	c(-0.261059999465942, -0.254339993000031, 51.551757812...
40.1	47	c(-0.221849992871284, -0.213129997253418, 51.573909759...
39	44	c(-0.0770699977874756, -0.059919998049736, -0.04671999...
38	42	c(-0.127930000424385, -0.118839994072914, 51.614799499...
37	40	c(-0.249899998307228, -0.221849992871284, 51.565277099...
36	39	c(-0.207649990916252, -0.192819997668266, 51.587989807...
36.1	39	c(-0.171089991927147, -0.136439993977547, 51.602588653...
35	38	c(-0.015990000218153, -0.0015499999281019, 51.60083007...
34	34	c(-0.192570000886917, -0.171089991927147, 51.590309143...
34.1	34	c(-0.0159999988973141, -0.00122999993618578, 51.600719...
34.2	34	c(-0.0015499999281019, 0.0123399998992682, 51.60012054...
33	33	c(-0.254339993000031, -0.250019997358322, 51.553367614...
33.1	33	c(-0.088249996304512, -0.0769199952483177, -0.06408999...
33.2	33	c(0.000189999991562217, 0.00239999988116324, 51.50154...
32	32	c(0.0319500006735325, 0.0443099997937679, 51.51705932...
31	31	c(-0.128209993243217, -0.118839994072914, 51.614757537...



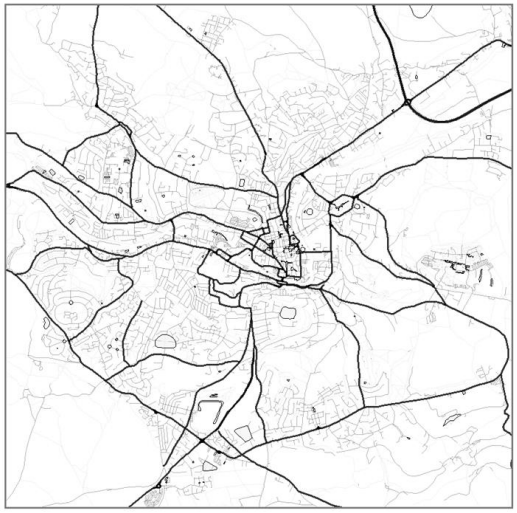
https://github.com/ropensci/stplanr

London Business School

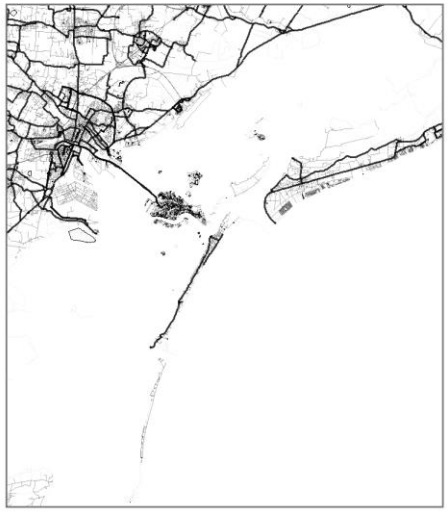
Osmdata

09_make_sexy_map.R

```r
# remotes::install_github("lina2497/Giftmap")
library(Giftmap)

# all of the actual R code can be found at
# https://github.com/lina2497/Giftmap/blob/master/R/make_sexy_map.r

# make_sexy_map(location, short_name, monochrome = FALSE)
#
# Arguments
# location: A character string describing the target, gets passed to osmdata::getbb().
# short_name: A character string to be used as a caption.
# monochrome: Logical, if TRUE output is black and white.

make_sexy_map("Bath UK",
              "Bath",
              monochrome = TRUE)
              #monochrome = FALSE)

make_sexy_map("Venice, Italy",
              "Venice",
              monochrome = TRUE)
              #monochrome = FALSE)
```



BATH

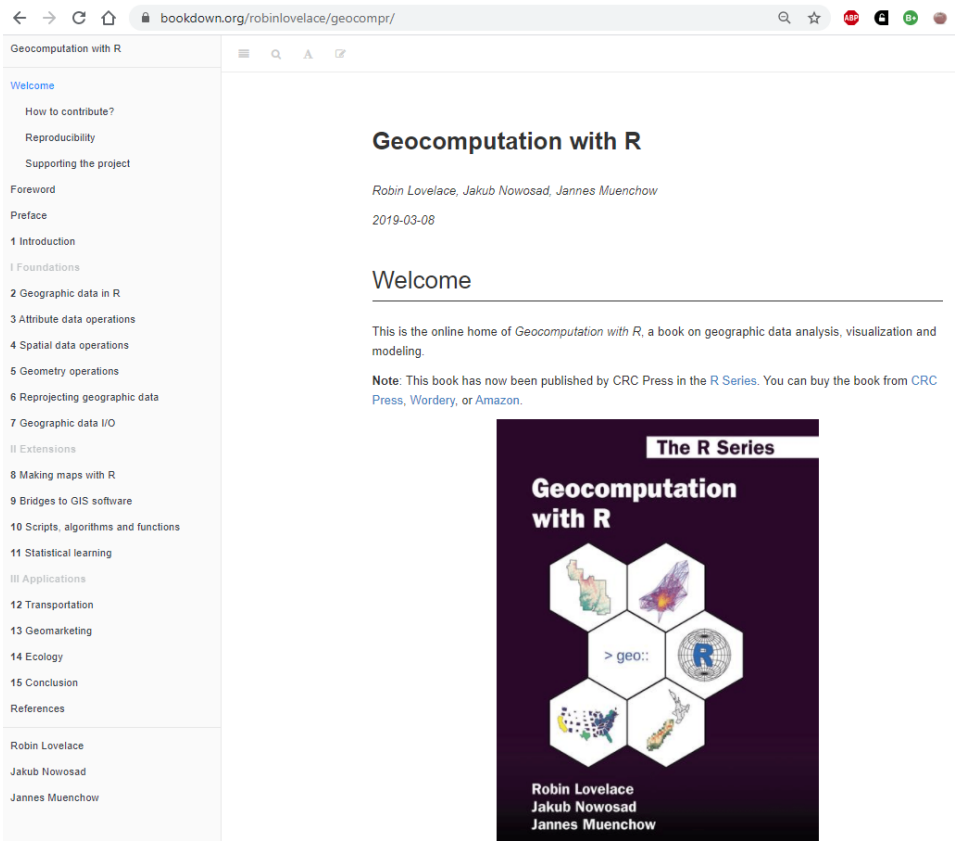


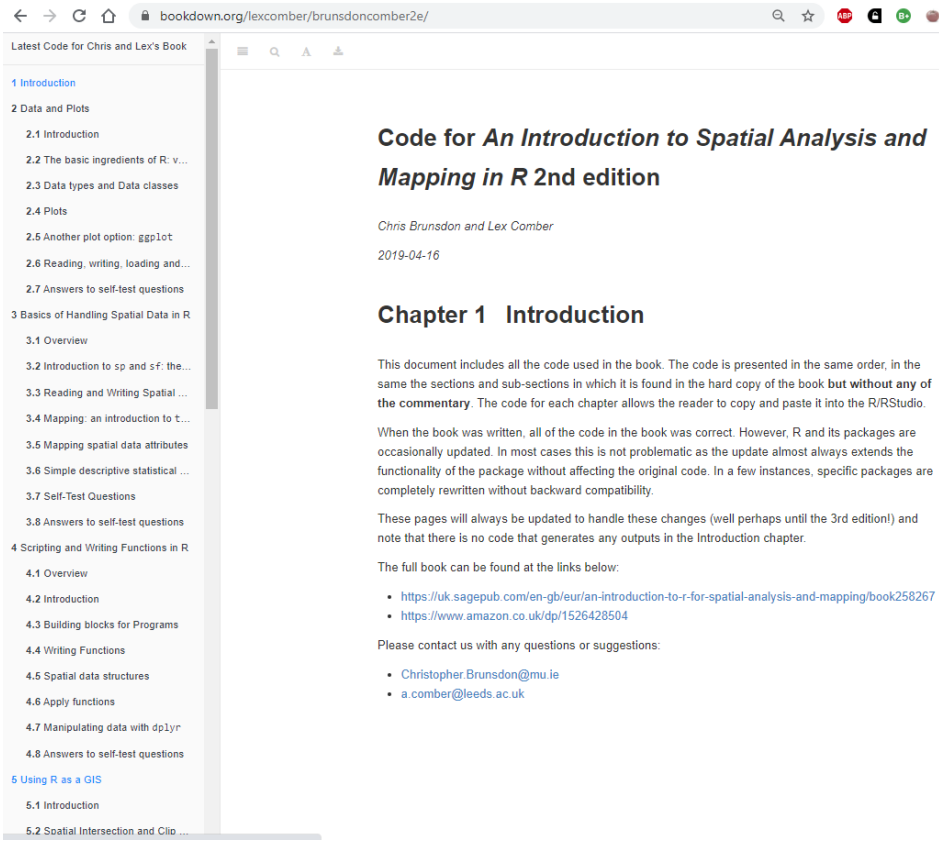
VENICE

https://github.com/lina2497/Giftmap/blob/master/R/make_sexy_map.r

Resources

Ignore all (older) tutorials/resources that use `geom_map()` or `ggmap()`
`geom_sf()` is the de-facto standard for spatial data manipulation in R



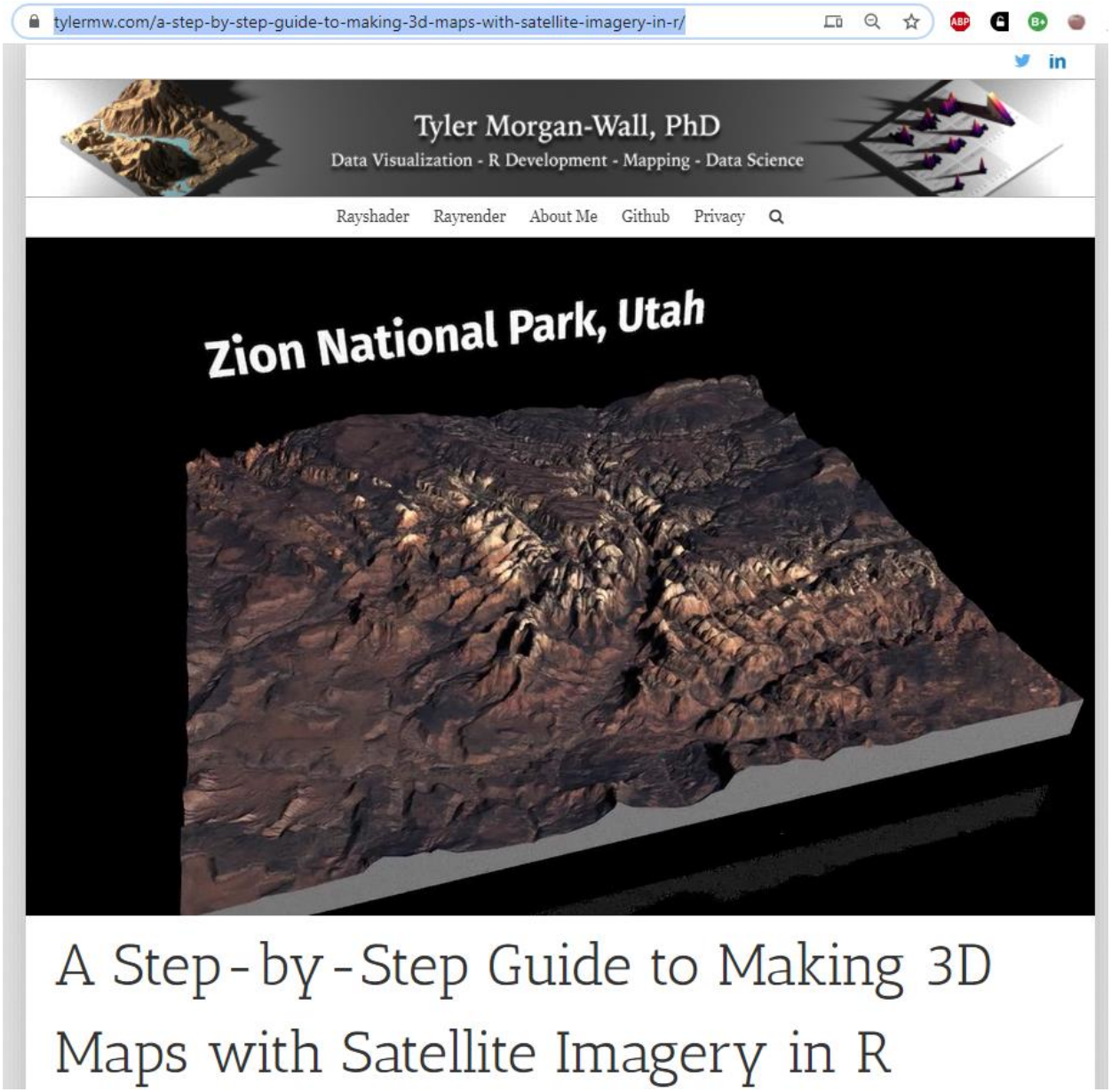


https://bookdown.org/robinlovelace/geocompr/

https://bookdown.org/lexcomber/brunsdoncomber2e/

London Business School

And finally…



https://www.tylermw.com/a-step-by-step-guide-to-making-3d-maps-with-satellite-imagery-in-r/

Session Overview – Next Steps

12_read_many_CSV.R

1. Spatial visualisations geom_sf()
2. Interactivity with plotly, flexdashboard, shiny
3. Individual assignment for next week
   - Problem set in R: Stop and Search in London, part 2
     - You will make a series of 3-4 static visualization plots about Stop and Search in London in Oct 2018 - Sep 2021
4. **Group project proposal**

```
 8  # read many CSV files
 9  # Adapted from https://www.gerkelab.com/blog/2018/09/import-directory-csv-purrr-readr/
10
11  # assuming all your files are within a directory called 'data/stop-search'
12  data_dir <- "data/stop-search"
13
14  files <- fs::dir_ls(path = data_dir, regexp = "\\.csv$", recurse = TRUE)
15  #recurse=TRUE will recursively look for files further down into any folders
16
17  files
18  s
19  #read them all in using vroom::vroom()
20  stop_search_data <- vroom(files, id = "source")
```