ECON6027
1c

Handling Vector Spatial Data

Sf objects

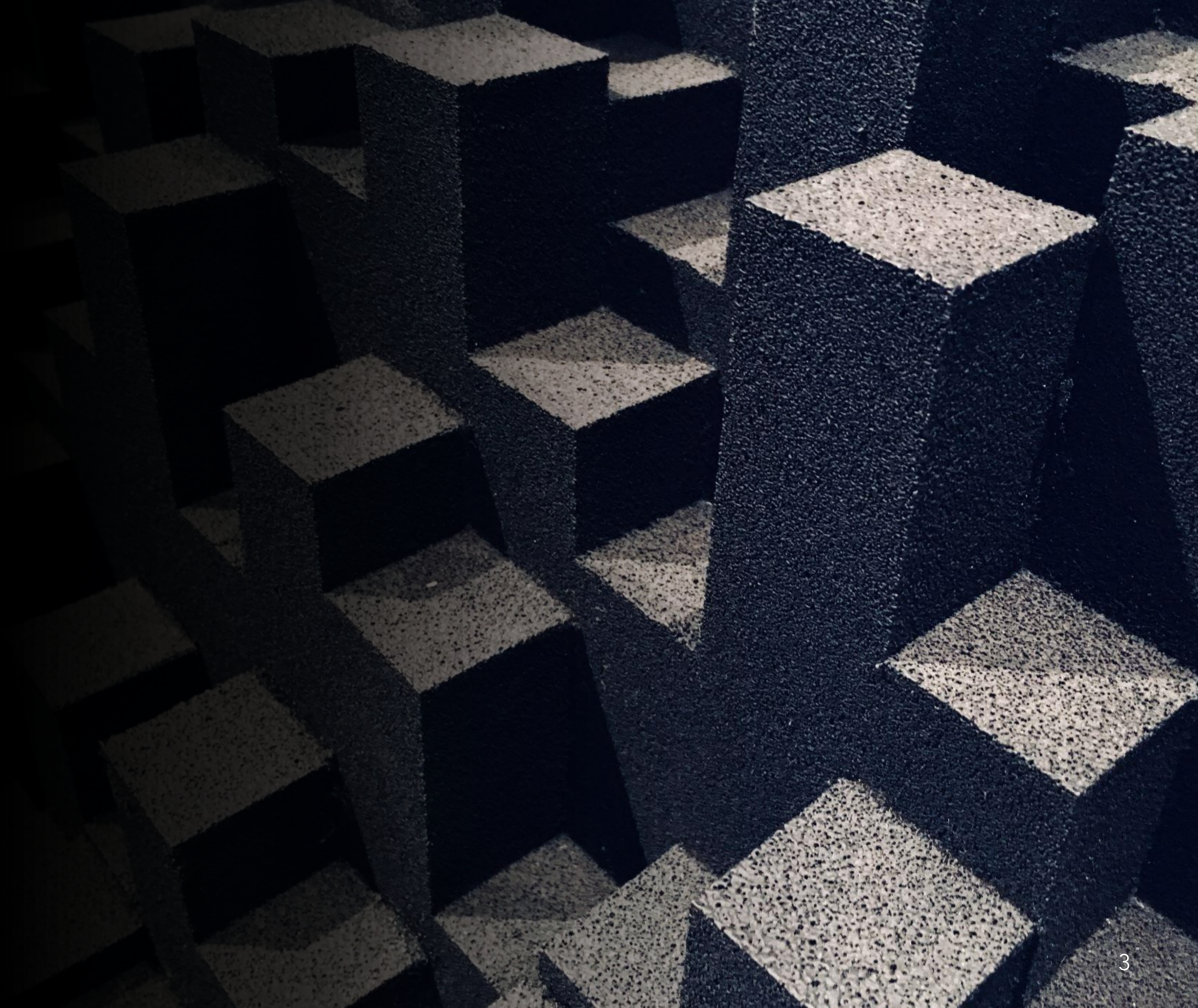
In R, sf objects are datasets that have spatial awareness. That means they know exactly "where" we are talking about!

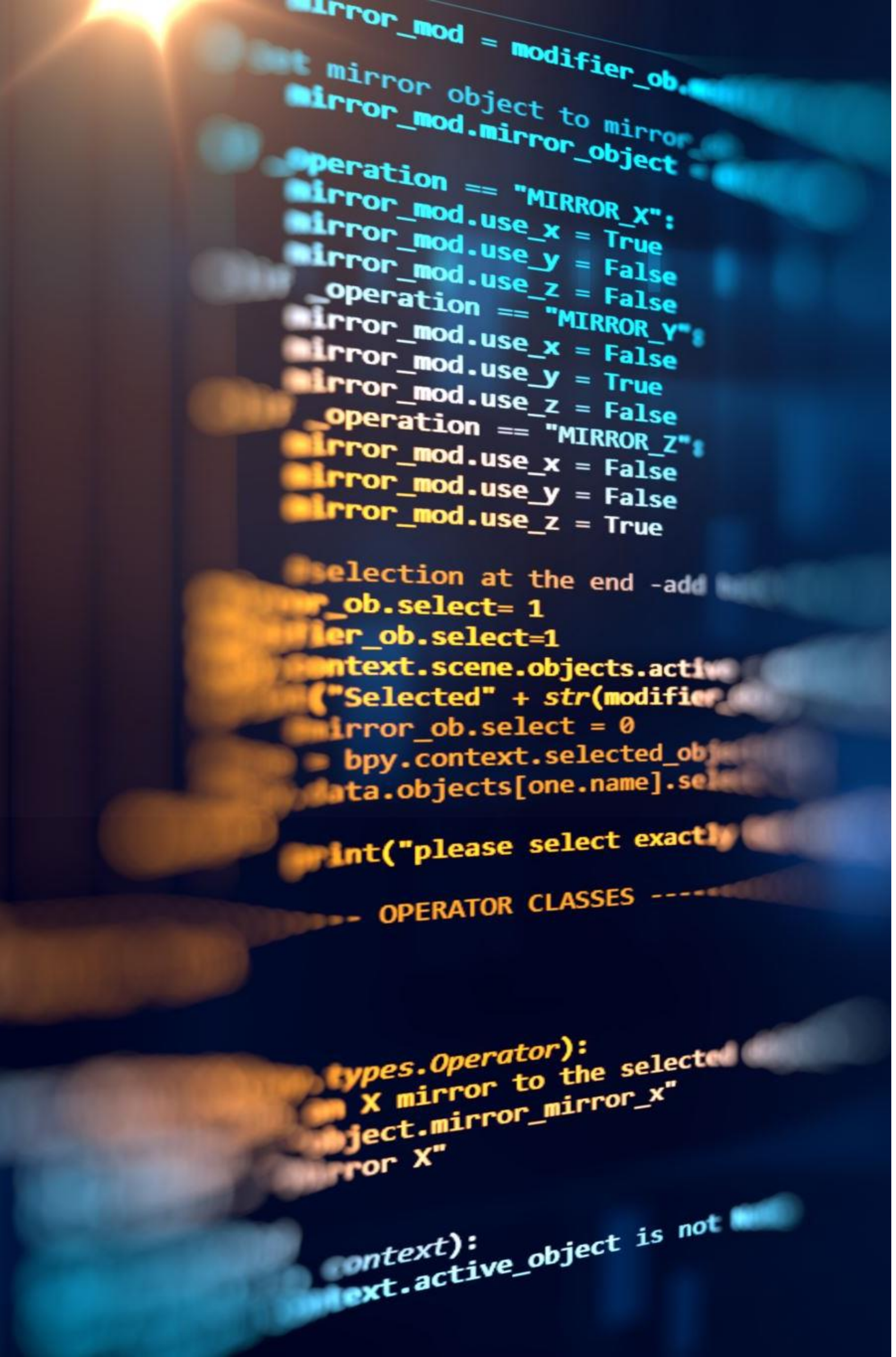
In 1b we saw how to import a geographic dataset to R and thereby create a sf object.

In this section, we will discuss explore sf objects further.

Packages you need

- sf
- spData
    - contains spatial datasets





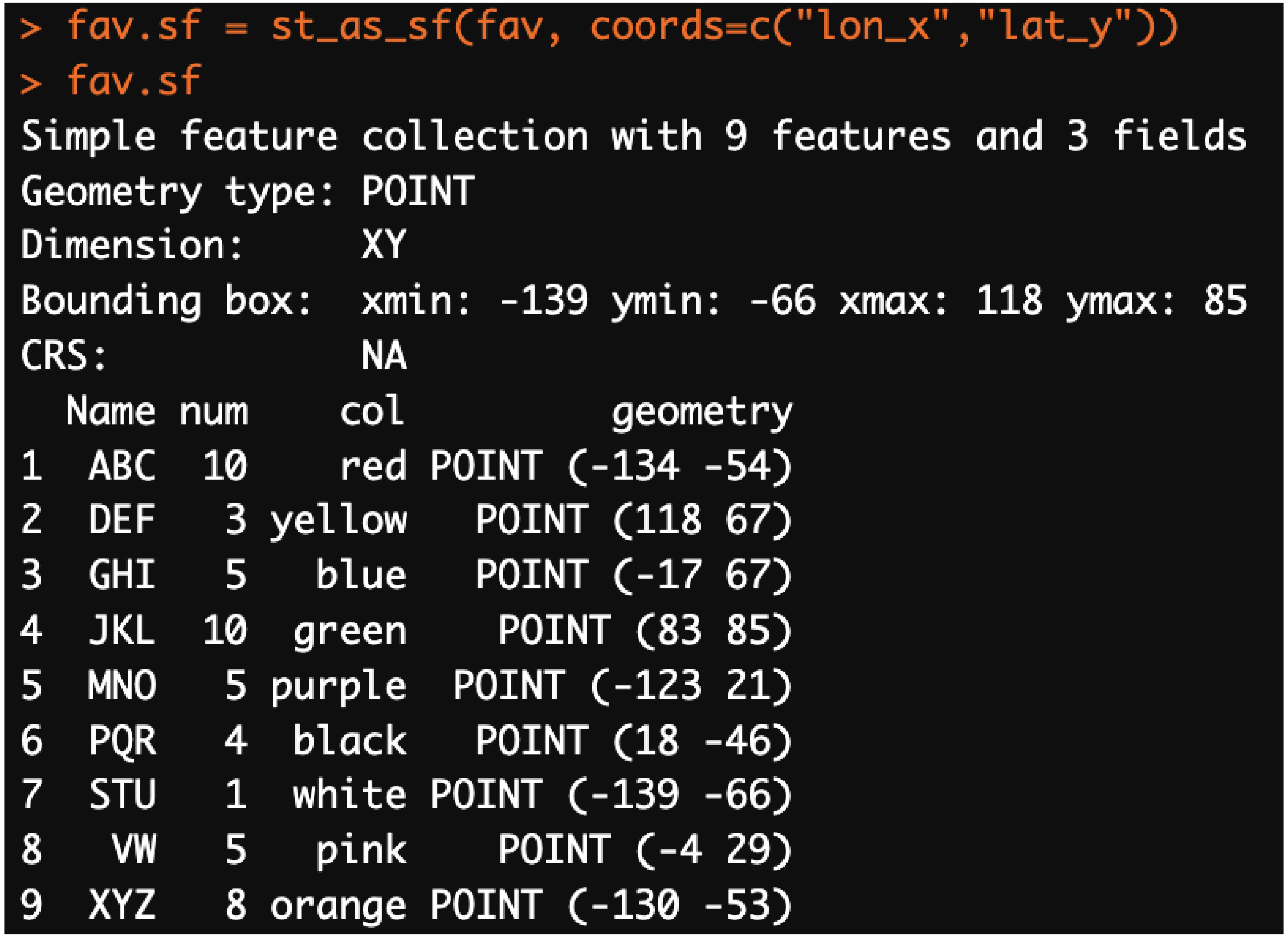
```
irror_mod = modifier_ob.
mirror object to mirror
irror_mod.mirror_object
peration == "MIRROR_X":
irror_mod.use_x = True
irror_mod.use_y = False
irror_mod.use_z = False
 operation == "MIRROR_Y":
irror_mod.use_x = False
irror_mod.use_y = True
irror_mod.use_z = False
 operation == "MIRROR_Z":
irror_mod.use_x = False
irror_mod.use_y = False
irror_mod.use_z = True

election at the end -add
_ob.select= 1
er_ob.select=1
ntext.scene.objects.active
("Selected" + str(modifier
irror_ob.select = 0
 bpy.context.selected_obj
ata.objects[one.name].sel

int("please select exactly

-- OPERATOR CLASSES ----

types.Operator):
 X mirror to the selected
ject.mirror_mirror_x"
rror X"

ontext):
ext.active_object is not
```

Import data to r

- Import the csv file to the R using read.csv() function.
- We have a spatial dataset, but this dataset has no “spatial awareness”
- Let’s give it some spatial powers!

Give spatial awareness

“coords” argument follows the cartesian coordinates convention (x,y).

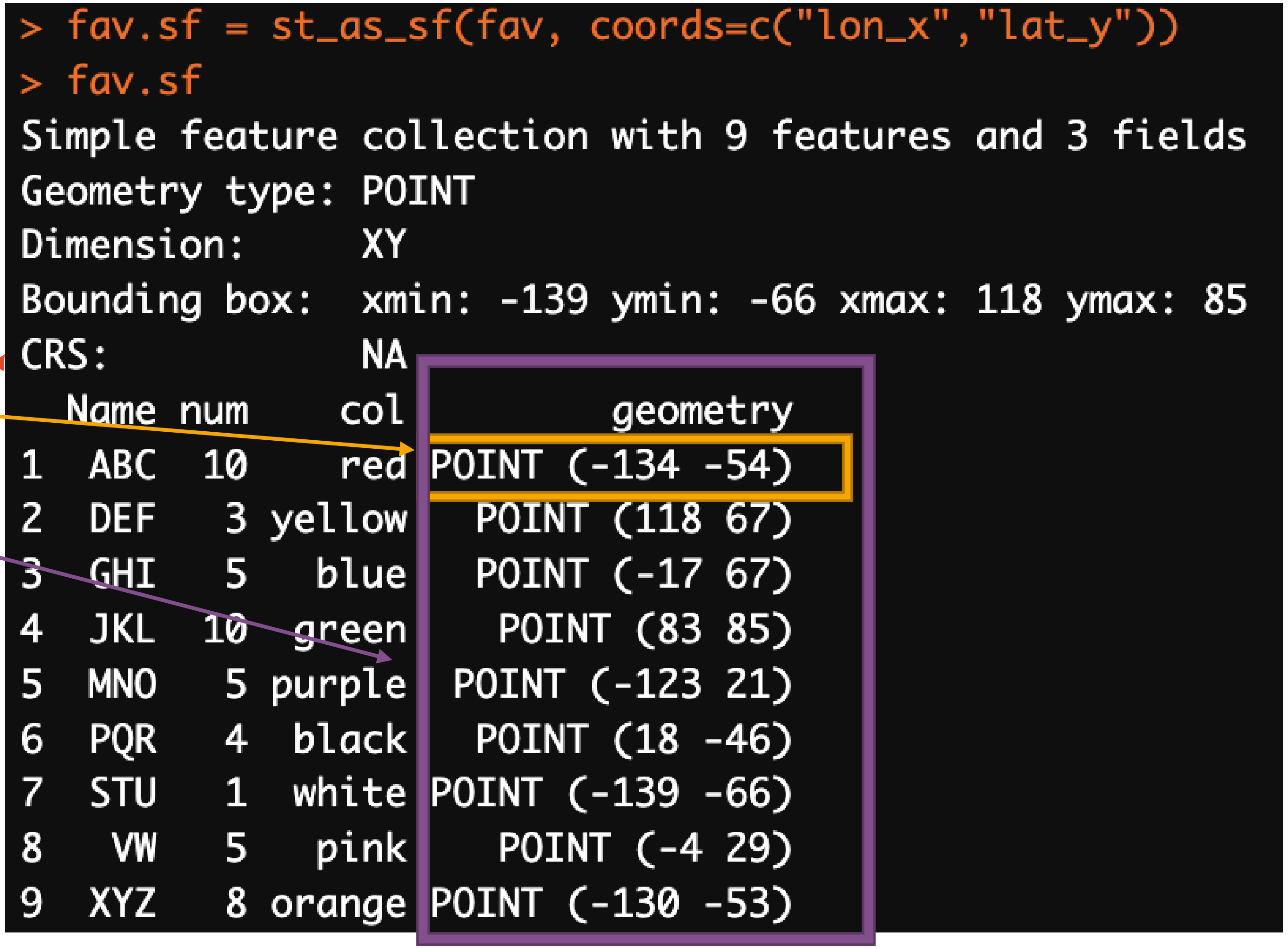


```
> fav.sf = st_as_sf(fav, coords=c("lon_x","lat_y"))
> fav.sf
Simple feature collection with 9 features and 3 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -139 ymin: -66 xmax: 118 ymax: 85
CRS:           NA
  Name num    col         geometry
1  ABC  10    red POINT (-134 -54)
2  DEF   3 yellow   POINT (118 67)
3  GHI   5   blue   POINT (-17 67)
4  JKL  10  green    POINT (83 85)
5  MNO   5 purple  POINT (-123 21)
6  PQR   4  black   POINT (18 -46)
7  STU   1  white POINT (-139 -66)
8   VW   5   pink    POINT (-4 29)
9  XYZ   8 orange POINT (-130 -53)
```

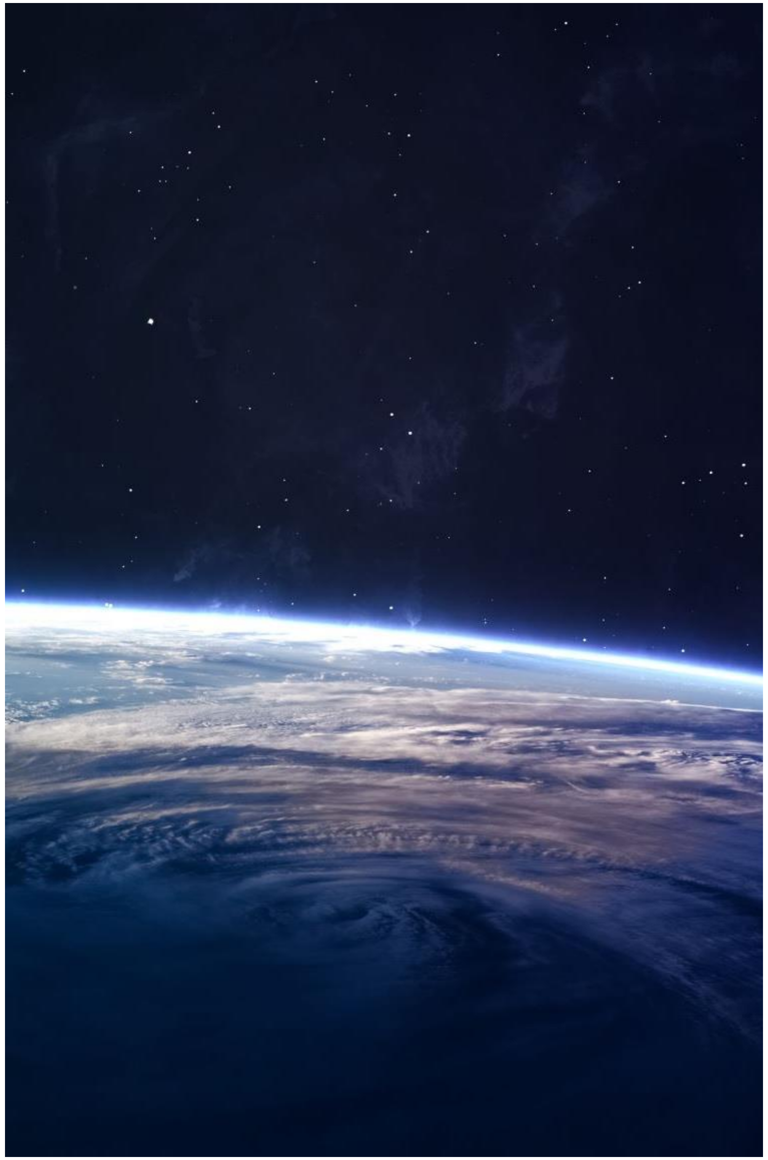
Notice the lack of CRS: R knows these are locations but not quite sure where… See Ch 4

Basic Terminology

- One sfg object contains only a single simple feature geometry.
- A simple feature geometry column (sfc) is a list of sfg objects, which is additionally able to contain information about the coordinate reference system in use. (Ch 2)
- This is important since sfc represents the geometry column in **sf** data frames



```
> fav.sf = st_as_sf(fav, coords=c("lon_x","lat_y"))
> fav.sf
Simple feature collection with 9 features and 3 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: -139 ymin: -66 xmax: 118 ymax: 85
CRS:           NA
  Name num    col          geometry
1  ABC  10    red POINT (-134 -54)
2  DEF   3 yellow   POINT (118 67)
3  GHI   5   blue   POINT (-17 67)
4  JKL  10  green    POINT (83 85)
5  MNO   5 purple  POINT (-123 21)
6  PQR   4  black   POINT (18 -46)
7  STU   1  white POINT (-139 -66)
8   VW   5   pink    POINT (-4 29)
9  XYZ   8 orange POINT (-130 -53)
```



Roll up your sleeves…

Install package “spData” and load the “world” dataset as an sf object:

```
> world = st_read(system.file("shapes/world.gpkg",
package="spData"))
```

```
# Inspect the world dataset and general sf
properties
```

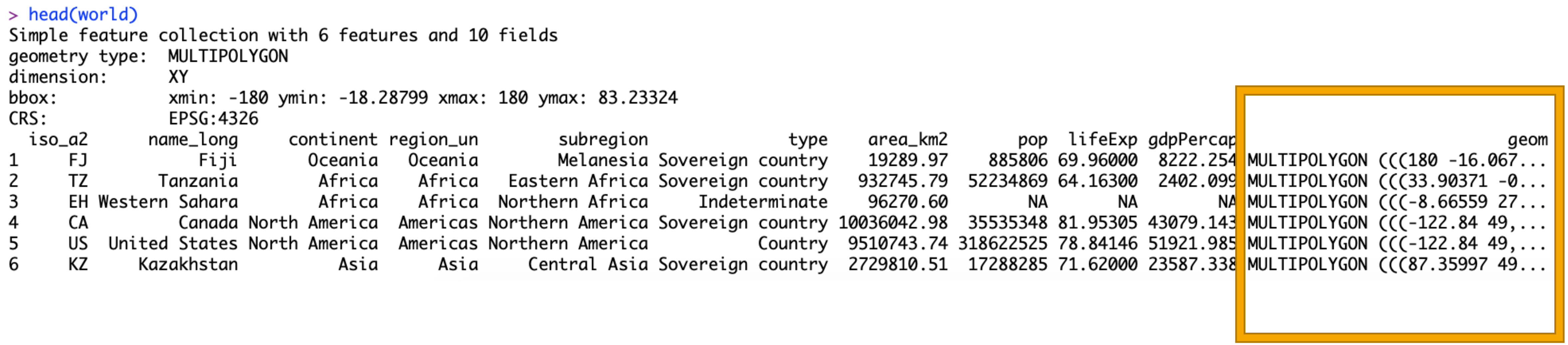
```
> class(world); dim(world); names(world);
summary(world); head(world)
```

sf class dataset

```
> class(world)
[1] "sf"        "tbl_df"     "tbl"        "data.frame"
```

Test yourself…

- Identify “attributes” and “features”.
- sf objects also have a special column to contain geometry data, can you identify it?
- What is the coordinate reference system associated with this dataset?

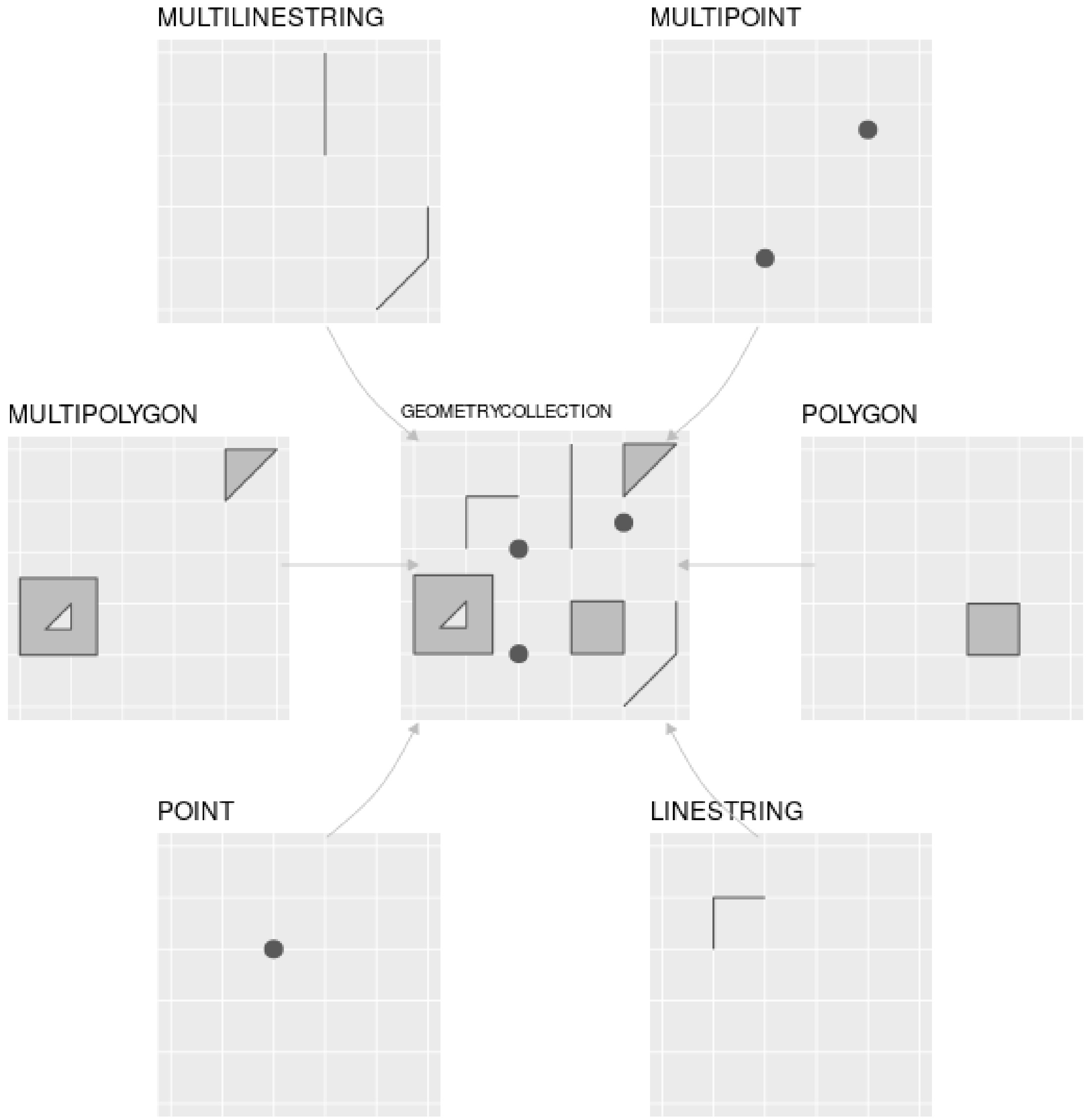


```
> head(world)
Simple feature collection with 6 features and 10 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -180 ymin: -18.28799 xmax: 180 ymax: 83.23324
CRS:            EPSG:4326
  iso_a2      name_long     continent region_un         subregion              type    area_km2       pop  lifeExp gdpPercap                           geom
1     FJ           Fiji       Oceania   Oceania         Melanesia Sovereign country    19289.97    885806 69.96000  8222.254 MULTIPOLYGON (((180 -16.067...
2     TZ       Tanzania        Africa    Africa    Eastern Africa Sovereign country   932745.79  52234869 64.16300  2402.099 MULTIPOLYGON (((33.90371 -0...
3     EH Western Sahara        Africa    Africa   Northern Africa     Indeterminate    96270.60        NA       NA        NA MULTIPOLYGON (((-8.66559 27...
4     CA         Canada North America  Americas Northern America Sovereign country 10036042.98  35535348 81.95305 43079.143 MULTIPOLYGON (((-122.84 49,...
5     US  United States North America  Americas Northern America           Country  9510743.74 318622525 78.84146 51921.985 MULTIPOLYGON (((-122.84 49,...
6     KZ     Kazakhstan          Asia      Asia      Central Asia Sovereign country  2729810.51  17288285 71.62000 23587.338 MULTIPOLYGON (((87.35997 49...
```

Attributes vs. Features

- Attribute data is “non-spatial” information associated with geographic (geometry) data. Examples:
  - A bus stop provides a simple example: its position would typically be represented by latitude and longitude coordinates (geometry data), in addition to its name. The name is an *attribute* of the feature (to use Simple Features terminology) that bears no relation to its geometry.
  - The elevation value (attribute) for a specific grid cell in raster data.

Feature types supported by sf



Geometry column

The geometry column gives the sf object its “spatial awareness”.

It is a list column that contains all the coordinates of the country polygons.

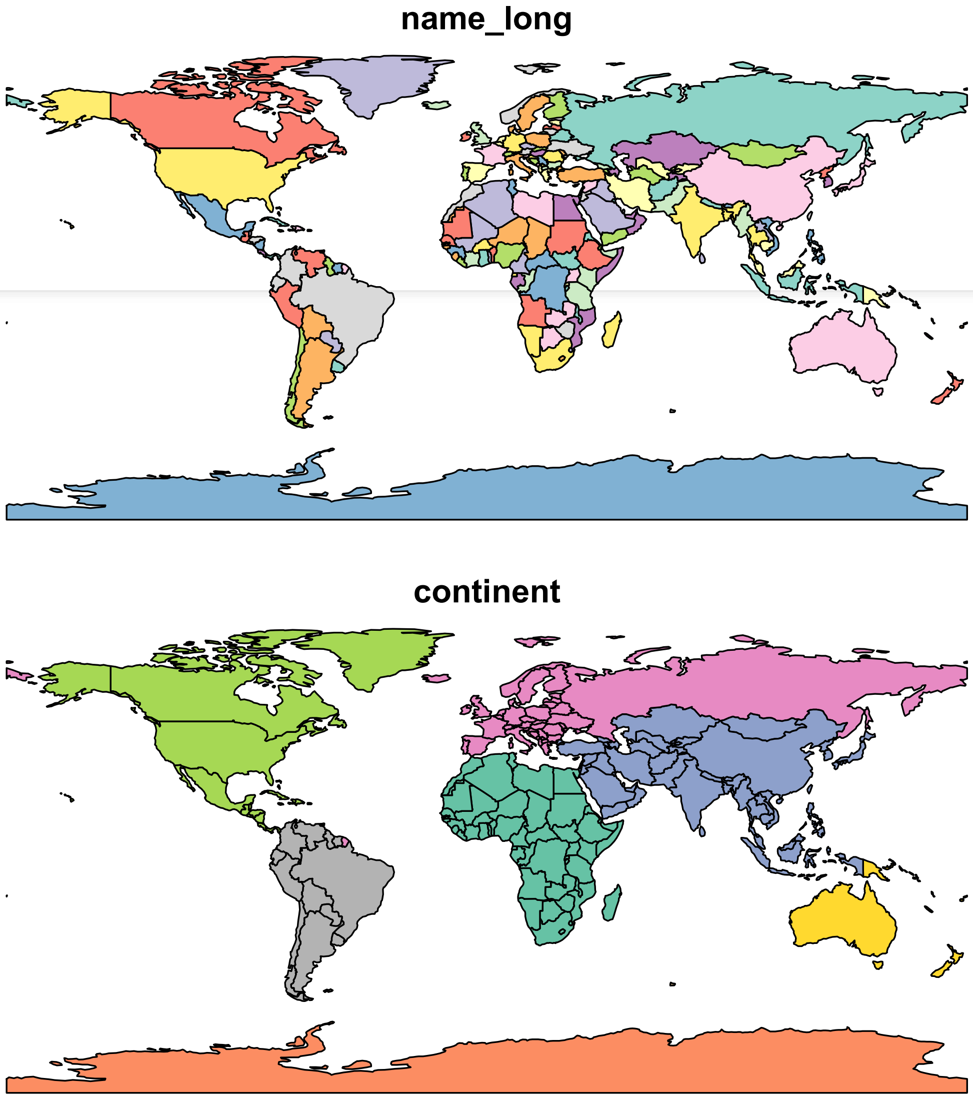
Inspect the geometry features

```
> world$geom

Geometry set for 177 features

Geometry type: MULTIPOLYGON

Dimension:     XY

Bounding box:  xmin: -180 ymin: -89.9
xmax: 180 ymax: 83.64513

Geodetic CRS:  WGS 84

First 5 geometries…
```

If you strip the geometry data from an “sf” object, it reduces to a typical “data.frame”

```
> world_df = 
st_drop_geometry(world);
class(world_df)
[1] "data.frame”
```

```
> head(world_df)
```



Visualise

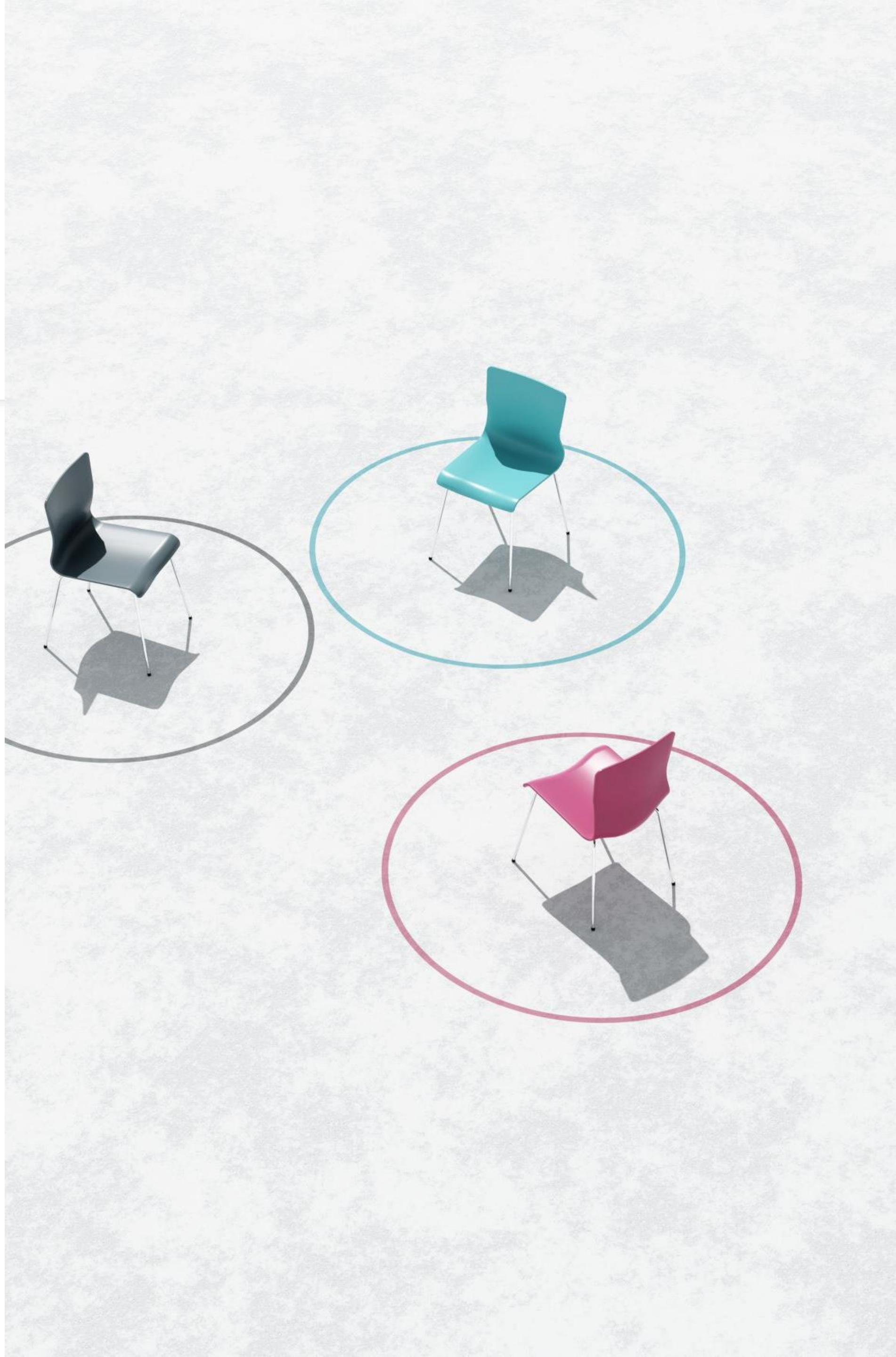
```
> plot(world) # plot all
```

```
> plot(world[c("name_long",
"continent")])
```

Manipulation of a Vector Dataset

1. Spatial (Attribute) Subsetting

- Spatial subsetting is the process of taking a spatial object and returning a new object containing only features that *relate* in space to another object.
- In other words, we will collect attributes that have some shared common element.
  - eg: collection of mall names and size in the central region



1. Subsetting Examples

Extract a numerical summary of the numerical data in columns 7 through 10:

```
> summary(world[, 7:10])
```

A subset of large countries:

```
> l_area = world$area_km2 >
5000000

> (big_countries =
world[l_area, ])

# or

> (big_countries =
subset(world, area_km2 >
500000))
```

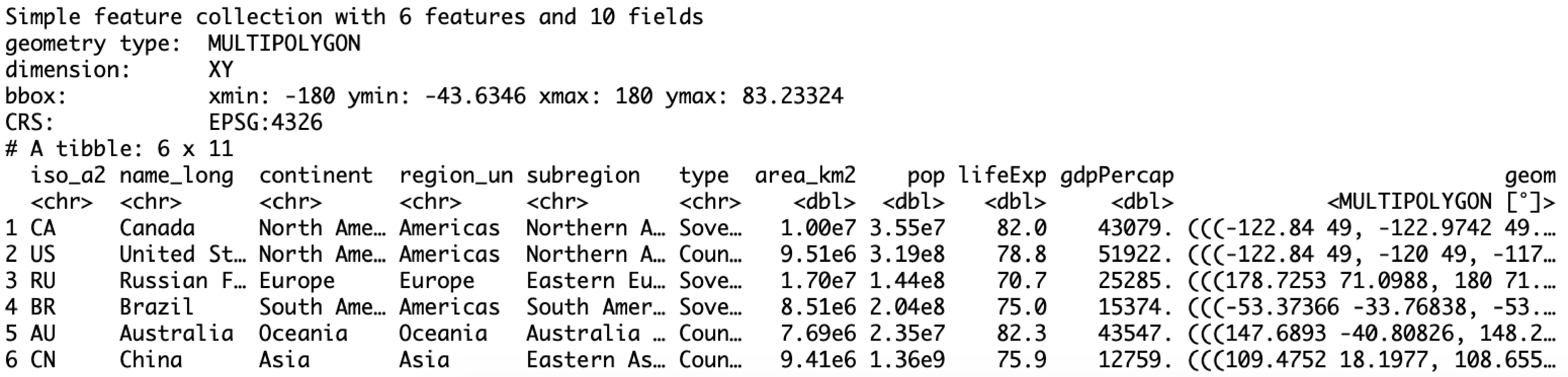
Standard comparison operators

Symbol	Name
==	Is equal to
!=	Is not equal to
<,>	Is great/less than
<=, >=	Is greater/less than or equal to
&, |, !	and, or, not

Subsetting Examples (contd.)

- Remove Antartica

```
> (big_countries = big_countries[-7, ])
```



```
Simple feature collection with 6 features and 10 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -180 ymin: -43.6346 xmax: 180 ymax: 83.23324
CRS:            EPSG:4326
# A tibble: 6 x 11
  iso_a2 name_long continent region_un subregion   type  area_km2    pop lifeExp gdpPercap                              geom
  <chr>  <chr>     <chr>     <chr>     <chr>       <chr>    <dbl>  <dbl>   <dbl>     <dbl>                <MULTIPOLYGON [°]>
1 CA     Canada    North Ame… Americas Northern A… Sove…   1.00e7 3.55e7    82.0    43079. (((-122.84 49, -122.9742 49.…
2 US     United St… North Ame… Americas Northern A… Coun…  9.51e6 3.19e8    78.8    51922. (((-122.84 49, -120 49, -117…
3 RU     Russian F… Europe    Europe    Eastern Eu… Sove…   1.70e7 1.44e8    70.7    25285. (((178.7253 71.0988, 180 71.…
4 BR     Brazil    South Ame… Americas South Amer… Sove…   8.51e6 2.04e8    75.0    15374. (((-53.37366 -33.76838, -53.…
5 AU     Australia Oceania   Oceania   Australia … Coun…   7.69e6 2.35e7    82.3    43547. (((147.6893 -40.80826, 148.2…
6 CN     China     Asia      Asia      Eastern As… Coun…   9.41e6 1.36e9    75.9    12759. (((109.4752 18.1977, 108.655…
```

EXERCISE A

1. Create a new sf object made of columns 3 to 6.
2. Create a new sf object by removing “iso_a2” and “region_un”.
3. Filter the countries with “gdpPercap” more than $50,000.
4. Try at home: Generate a subset of the “richest three” countries in ”Europe”.

Subsetting Examples (contd.)

```
> world$continent # check listed continents
```

```
> asia = world[world$continent == "Asia", ]
```

# extract Asia. Using [] is one way to subset a dataset. Notice the resulting dataset is also an sf object.

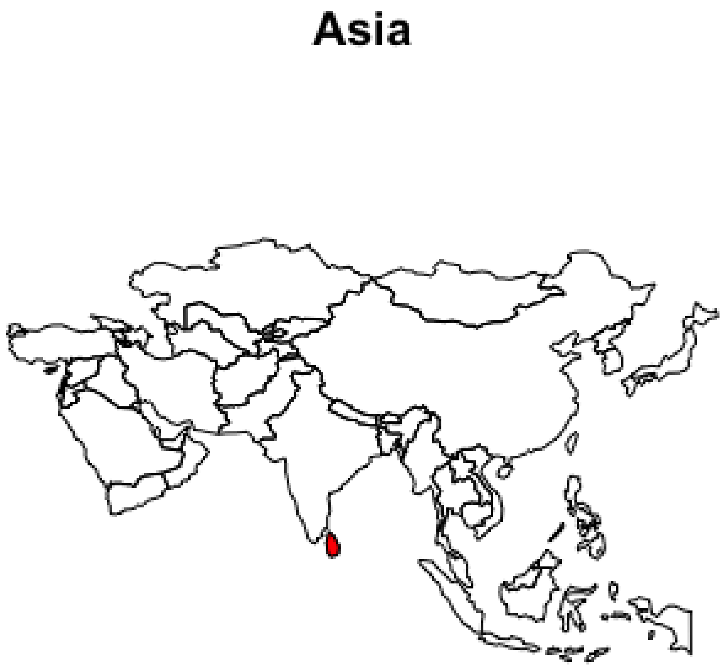
```
> plot(world["continent"], reset = F)
```

# if the first object has more than one key reset must be set to False for “add=T” option to work in the next line.

```
> plot(asia, add = T, col = "black")
```

Where do I come from?

```
> world$name_long # list of countries

> (SL = world[world$name_long == "Sri
Lanka", ]) # subsetting

> plot(st_geometry(asia), main="Asia")

> plot(st_geometry(SL), col = "red", lwd =
3, add=T)
```



# Notice [0] and plot(st_geometry()) both gives the same outcome: an outline of the geometry column.

Exercise B

- Repeat the steps above for a country of your choice. (Unfortunately, Singapore is not available in this dataset).

2. Attribute Aggregation

Aggregation operations summarise datasets by a ‘grouping variable’, typically an attribute column.

Attribute data aggregation example

Aggregate the population by continent:

1. Using aggregate function in “stats” package (invoked when using ~)

```
> (cont_pop = aggregate(pop ~ continent, FUN = sum, data = world, na.rm
= T)); class(cont_pop) # output is a dataframe
```

2. Using aggregate function in “sf” package (invoked when using an sf object and a “by” argument is given)

```
> (cont_pop2 = aggregate(world['pop'], by = list(world$continent), FUN =
sum, na.rm = T)); class(cont_pop2) # output is an sf object
```

na.rm=T allows R to exclude missing values in the calculation

EXERCISE C

1. Can you explain what the difference is between “cont_pop” and “cont_pop2”?
2. Run the command “?aggregate” and inspect the function arguments.
3. Create an object of the average population in each continent using “FUN=mean” while maintaining the geometry column of the object.

3. Attribute Joining

Joining data from different sources

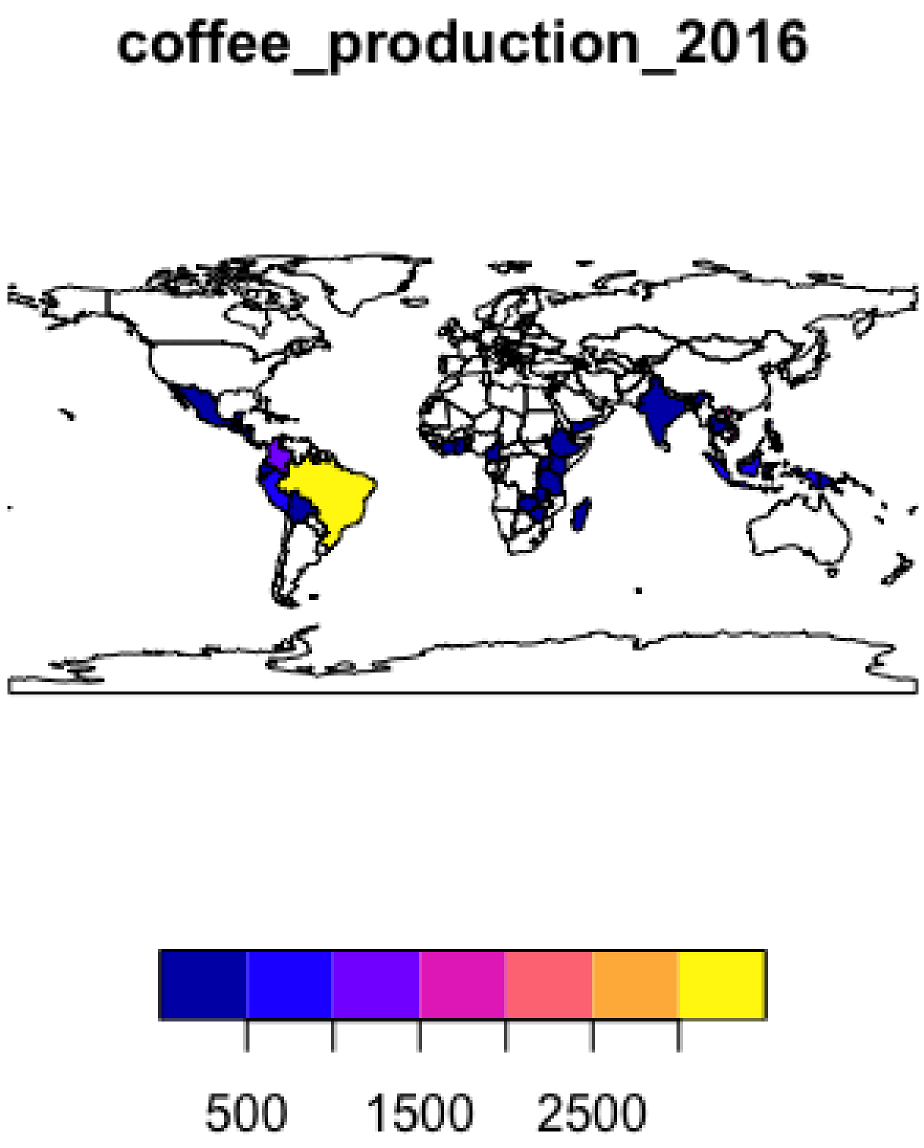
This is useful when you want to “give” spatial awareness to a dataset.

Example: attribute joining using a shared “key” variable

Combine “world” and “coffee_data” found in spData package.

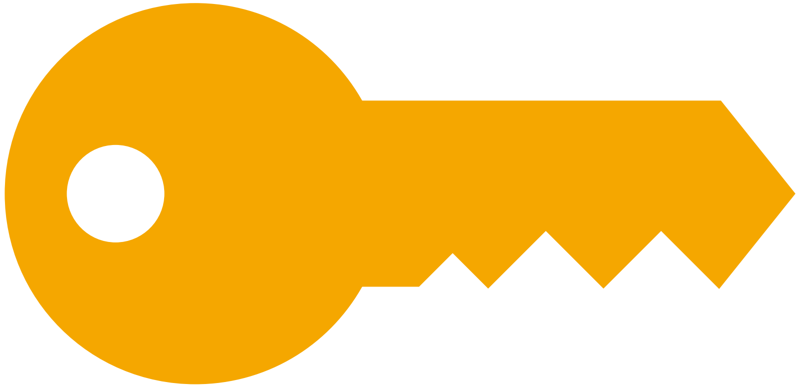
```
> library(spData)
> summary(coffee_data); class(coffee_data)


> (world_coffee = left_join(world, coffee_data))
# left_join preserves the left/first dataset.
> ?left_join  # see what other ways of joining two datasets are available.
```

Visualise the added variable



The “key variable”

- Notice the joining is done using the shared “**key**” variable “name_long”.
- By default, **dplyr** uses all variables with matching names. In this case, both world_coffee and world objects contained a variable called name_long, explaining the message Joining, by = "name_long".
- Where variable names are not the same, you have two options:
  - Rename the key variable in one of the objects so they match.
  - Use the by argument to specify the joining variables.



Other ways of mutating joins

- The mutating joins add columns from y to x, matching rows based on the keys:
  - inner_join(): includes all rows in x and y.
  - left_join(): includes all rows in x.
  - right_join(): includes all rows in y.
  - full_join(): includes all rows in x or y.
- If a row in x matches multiple rows in y, all the rows in y will be returned once for each matching row in x.

Example: attribute joining using a shared “key” variable

left_join preserves the left dataset in full. i.e., if there is an entry in the left dataset without an exact match with the right dataset, it simply inserts an <NA> entry. (Check entry 3 of world_coffee). What if we want to remove such entries completely. i.e. get an exact match of the left and right datasets?

```
> (world_coffee2 = inner_join(world, coffee_data))
# the sf object must be in front if you want to
retain the sf class
```

EXERCISE D

1. Illustrate coffee production in 2017 in a world map.
2. Conduct a left_join to create a dataset of only the coffee producing countries. Is this an sf object? How will you convert this to an sf object?

4. Create new attributes

```
> (world_coffee$prod_yoy =
(world_coffee$coffee_production_2017/world_c
offee$coffee_production_2016 - 1)*100)
> world_coffee
```

Exercise E (Challenge Exercise)

1. Load the world dataset form spData. Generate a subset that consists of Asia. Create a preliminary plot.
2. Load the “arrivals.Rdata” and extract the arrivals from Asia. Give it spatial powers by performing an appropriate joining with the world dataset. The final object must have all the attributes from both datasets. Inspect the object.
3. Extract a subset of arrivals from 2000 onward.
4. Find the average arrivals from each Asian country between 2000 to 2018. Plot it.
5. Add a population column to the average arrivals. Create a new attribute that shows the annual average arrivals as a percentage of total population. Plot this attribute.

Take home points…

- Manipulate attributes and features of an sf object
    - Subsetting
    - Aggregation
    - Joining

References

- https://geocompr.robinlovelace.net/attr.html#vector-attribute-manipulation
- https://geocompr.robinlovelace.net/spatial-operations.html#spatial-vec
- Datasets: https://nowosad.github.io/spData/