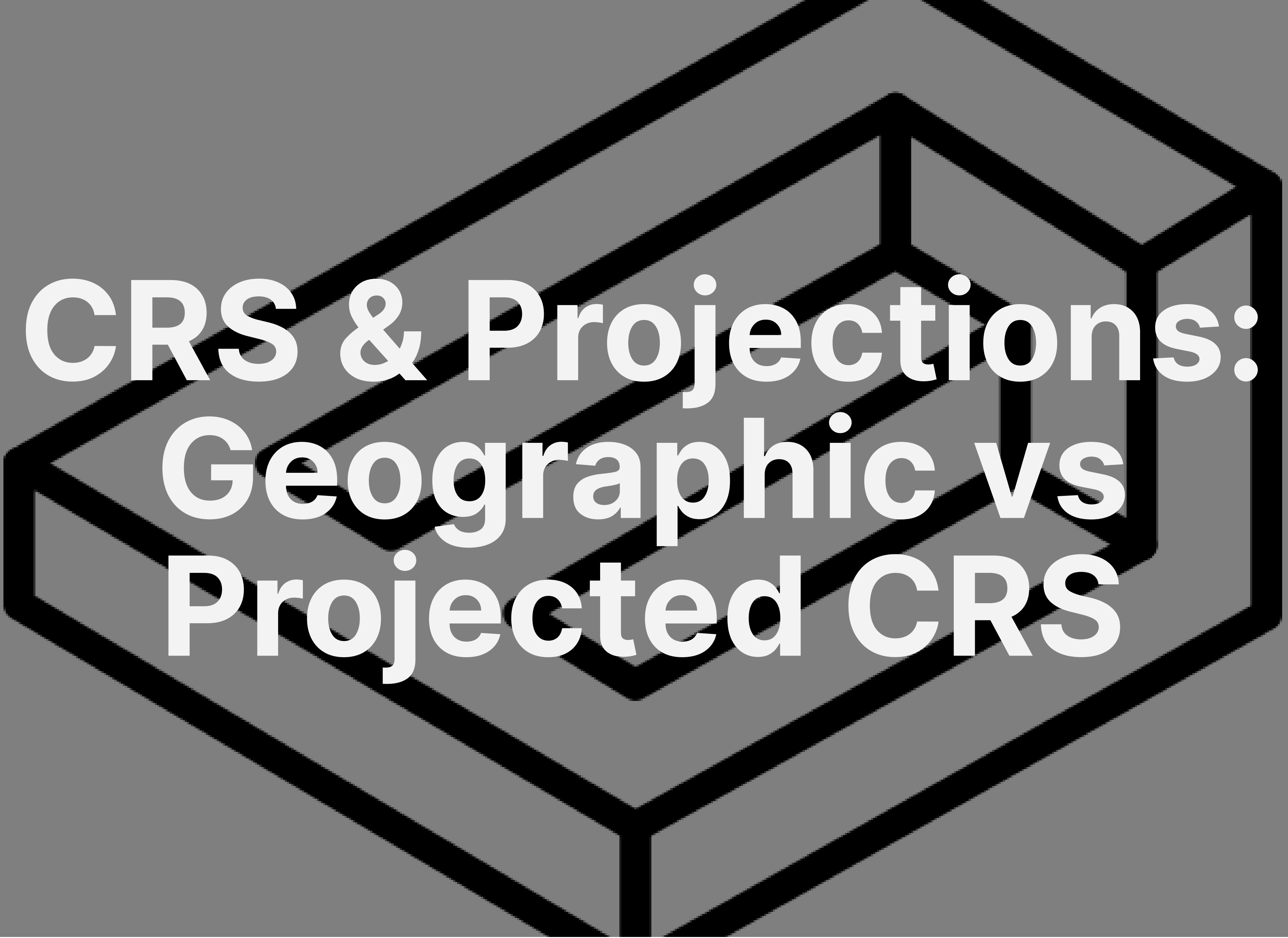
CRS & Projections: Geographic vs Projected CRS



CRS & Projections: Geographic vs Projected CRS

All maps are wrong...

To squash the surface of the Earth onto a map we need to **project** from a sphere's surface to a 2D flat map.

Geographic CRS vs Projected CRS

Geographic CRS (including 4326)

- Points are specified as longitudes and latitudes.
- Distances are measured in non-physical units: degrees.

Projected CRS

- Points are specified as (x, y) coordinates.
- Distances are measured in **physical units**, e.g. metres.

Geographic CRS vs Projected CRS

Geographic CRS (including 4326)

- Points are specified as longitudes and latitudes.
- Distances are measured in non-physical units: degrees.

Used for **visualisations**.

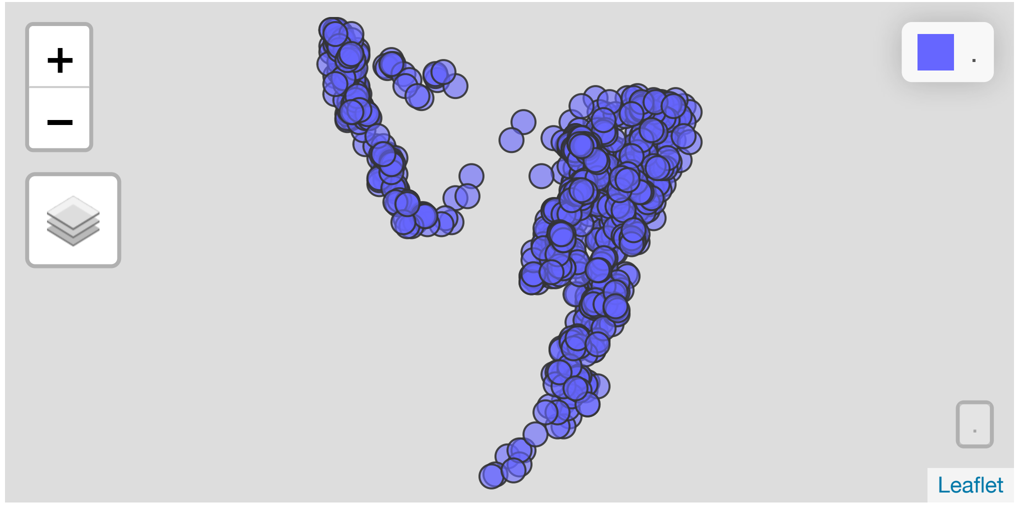
Projected CRS

- Points are specified as (x, y) coordinates.
- Distances are measured in **physical units**, e.g. metres.

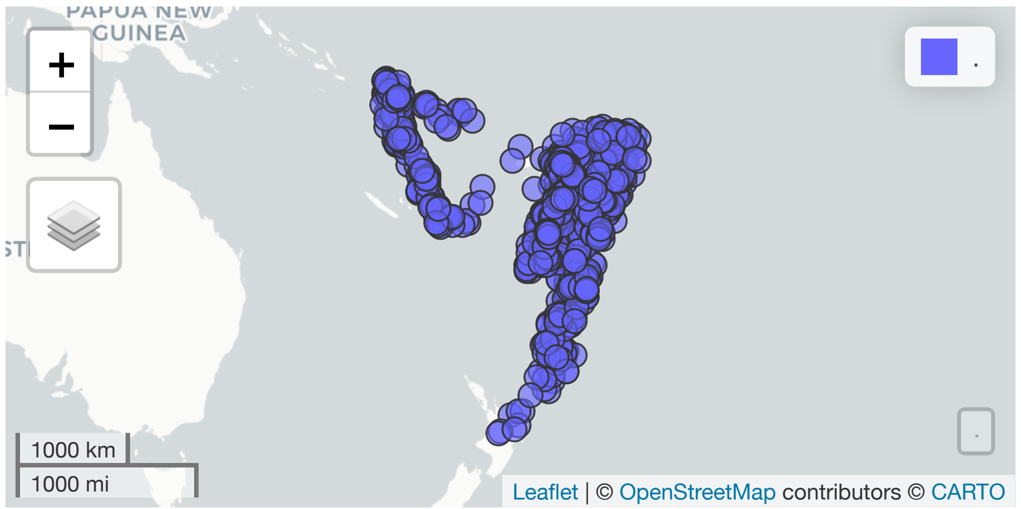
Used for **geometric calculations and GIS analysis**.

mapview needs a CRS to work

```r
quakes_no_crs <- quakes %>%
  st_as_sf(coords = c("long", "lat"))
quakes_no_crs %>% mapview()
```



```r
quakes_4326_crs <- quakes %>%
  st_as_sf(coords = c("long", "lat"),
                crs = 4326)
quakes_4326_crs %>% mapview()
```



CRS codes are EPSG codes

CRS 4326 is the **E**uropean **P**etroleum **S**urvey **G**roup (EPSG) code 4326.

EPSG.io makes it simple to find EPSG codes.

> Unfortunately, not all CRS have an EPSG code

CRS vs WKT

Only a small subset of possible Coorodinate Reference Systems *have* "CRS codes".

The most general purpose and flexible way to refer to (and store) projections is in the **Well-Known Text (WKT)** representation.

WKT include all the complexity of GIS systems (I)

We're smoothing over a lot of complicated mathematics required to build CRS.

The benefit of using WKT is that they can fully describe all of the details required for more complex GIS you or your colleagues might do in the future.

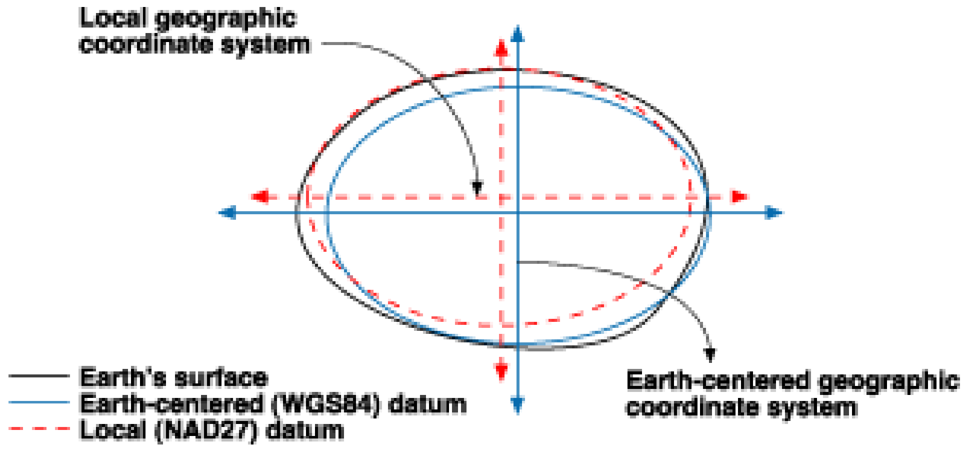

Local geographic coordinate system
Earth's surface
Earth-centered (WGS84) datum
Local (NAD27) datum
Earth-centered geographic coordinate system

Image credit: desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/datums.

WKT include all the complexity of GIS systems (II)

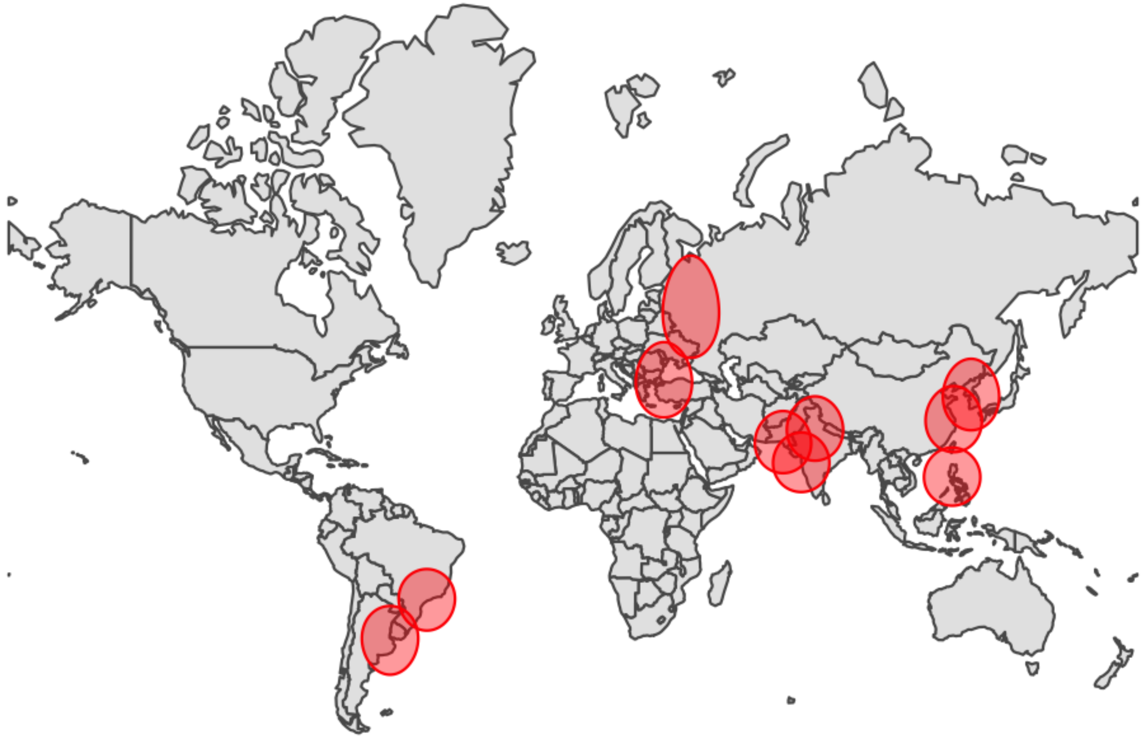
WKT can be recognised by the following features:

- Most WKT examples you'll use begin with `GEODCRS` or `PROJCS`.
- It contains nested square brackets

```
GEOGCS["WGS 84",
    DATUM["WGS_1984",
        SPHEROID["WGS 84",6378137,298.25722
            AUTHORITY["EPSG","7030"]],
        AUTHORITY["EPSG","6326"]],
    PRIMEM["Greenwich",0,
        AUTHORITY["EPSG","8901"]],
    UNIT["degree",0.0174532925199433,
        AUTHORITY["EPSG","9122"]],
    AUTHORITY["EPSG","4326"]]
```

RStudio Coding Slide

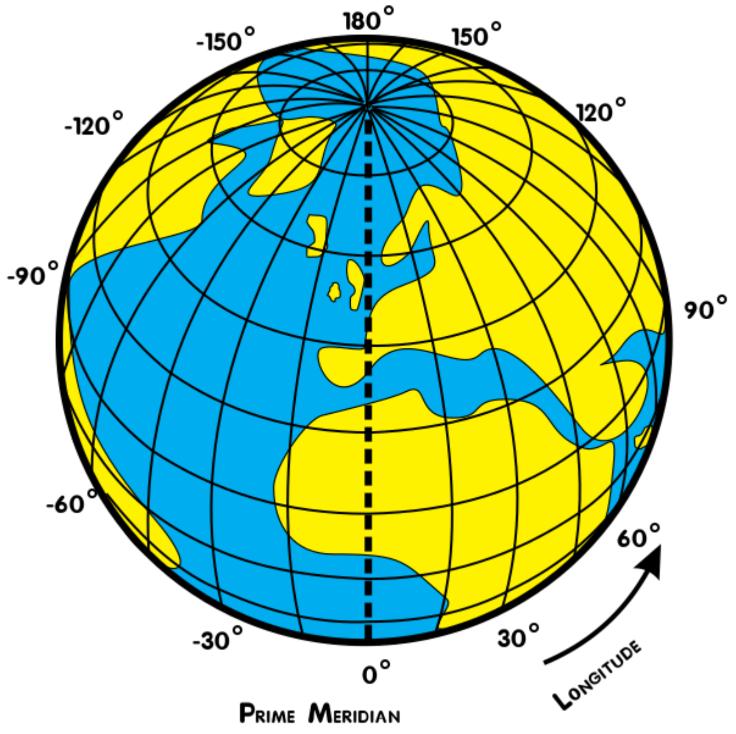
Don't use geographic CRS for analysis



Distance metrics are complicated in CRS 4326

Distances between points in geographic CRS are measured in degrees of **longitude** and **latitude**.

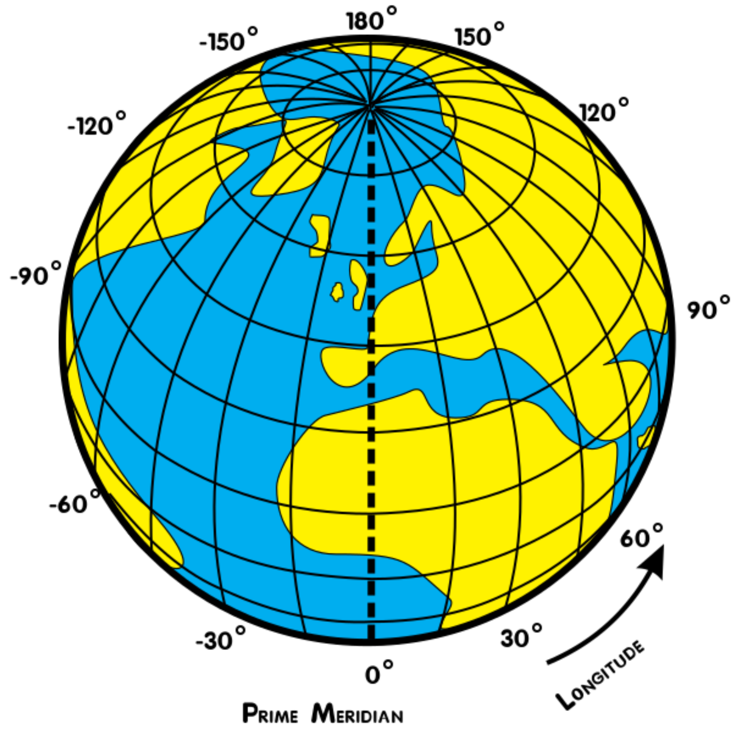
This globe shows the longitude lines (meridians) centred on the Prime Meridian in London.



Distance metrics are complicated in CRS 4326

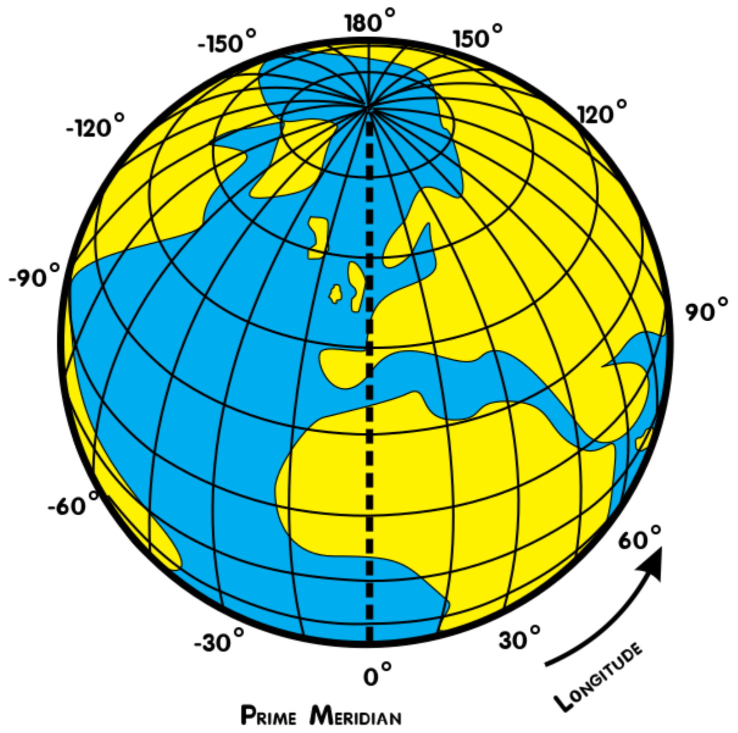
The distance between the longitude lines **varies** greatly dependent on how far points are away from the equator.

- At the equator a degree of longitude is 111.32 km.
- Moscow is at a latitude 55 where a degree of longitude is 62.64 km.



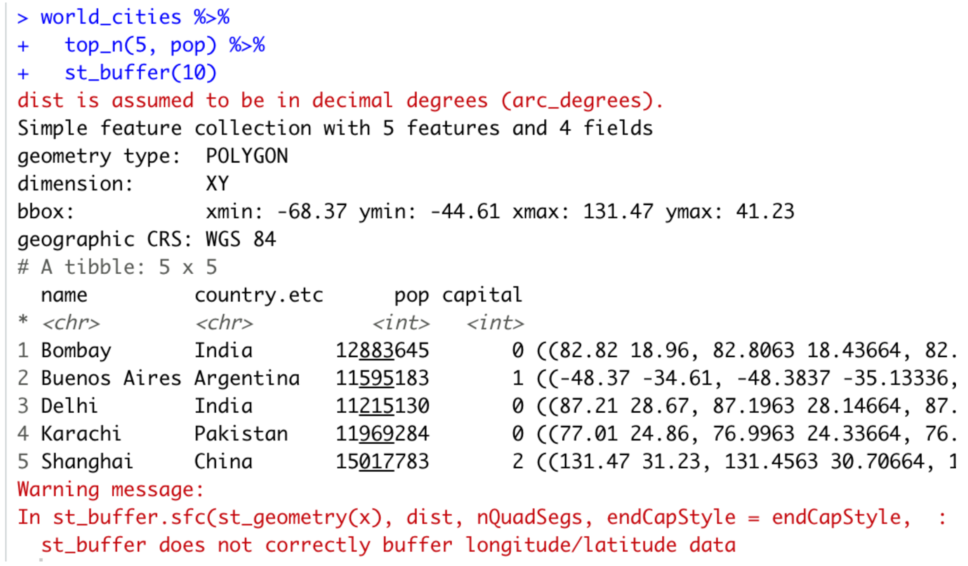
For the mathematics behind these calculations see here.

Geographic CRS distort GIS operations



{sf} has your back

{sf} will warn you about misusing geographic CRS



```
> world_cities %>%
+   top_n(5, pop) %>%
+   st_buffer(10)
dist is assumed to be in decimal degrees (arc_degrees).
Simple feature collection with 5 features and 4 fields
geometry type:  POLYGON
dimension:      XY
bbox:           xmin: -68.37 ymin: -44.61 xmax: 131.47 ymax: 41.23
geographic CRS: WGS 84
# A tibble: 5 x 5
  name         country.etc      pop capital
* <chr>        <chr>          <int>   <int>
1 Bombay       India       12883645       0 ((82.82 18.96, 82.8063 18.43664, 82.
2 Buenos Aires Argentina   11595183       1 ((-48.37 -34.61, -48.3837 -35.13336,
3 Delhi        India       11215130       0 ((87.21 28.67, 87.1963 28.14664, 87.
4 Karachi      Pakistan    11969284       0 ((77.01 24.86, 76.9963 24.33664, 76.
5 Shanghai     China       15017783       2 ((131.47 31.23, 131.4563 30.70664, 1
Warning message:
In st_buffer.sfc(st_geometry(x), dist, nQuadSegs, endCapStyle = endCapStyle,  :
  st_buffer does not correctly buffer longitude/latitude data
```

st_is_longlat() will return TRUE for geographic CRS.

```
world_cities %>%
  st_is_longlat()
```

```
## [1] TRUE
```