SPATIAL MAPS

STA465: Theory and Methods for Complex Spatial Data

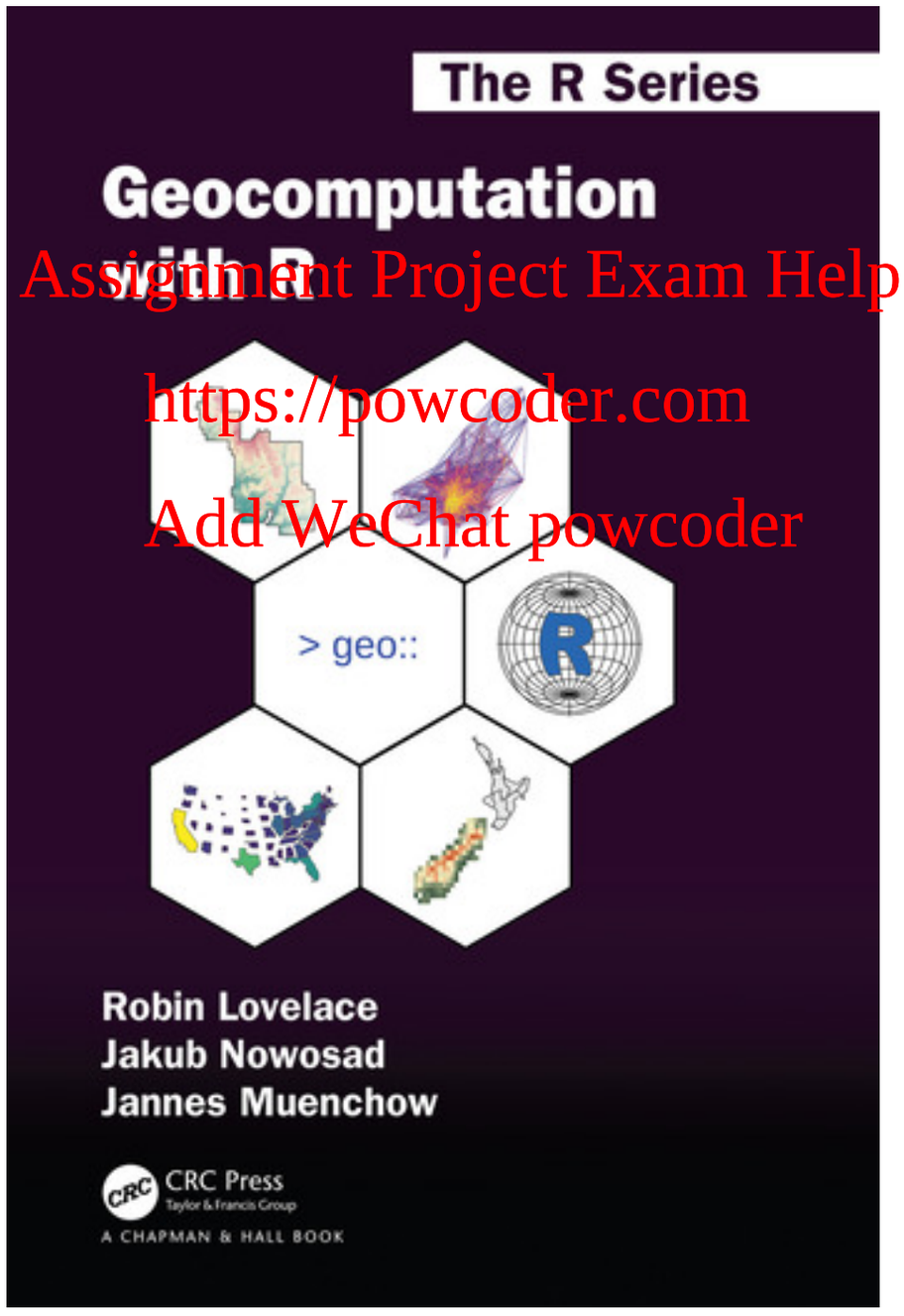
Instructor: Dr. Vianey Leos Barajas

WORKING WITH SPATIAL DATA STRUCTURES IN R

- We'll exclusively use R to work with spatial data:
  - There are two books we will use to learn how to work with spatial data structures and plot spatial data in R:
    - Geocomputation with R
    - Geospatial Health Data: Modeling and Visualization with R-INLA and Shiny
- Spatial data structures:
  - Beyond plotting points
  - Data sets will now have additional structures
  - https://r-spatial.github.io/sf/reference/index.html

GEOCOMPUTATION WITH R

PDF of book: *https://geocompr.robinlovelace.net*



R PACKAGES

install.packages("sf"): a class system for geographic vector data

install.packages("raster")

install.packages("spData")

remotes::install_github("Nowosad/spDataLarge")

library("sp")

What about ggplot2 or base R?

- *For spatial data structures, we need to go beyond that*
- *We can use it in combination with 'sf' or 'sp' structures*

VECTOR DATA

SIMPLE FEATURES (SF)

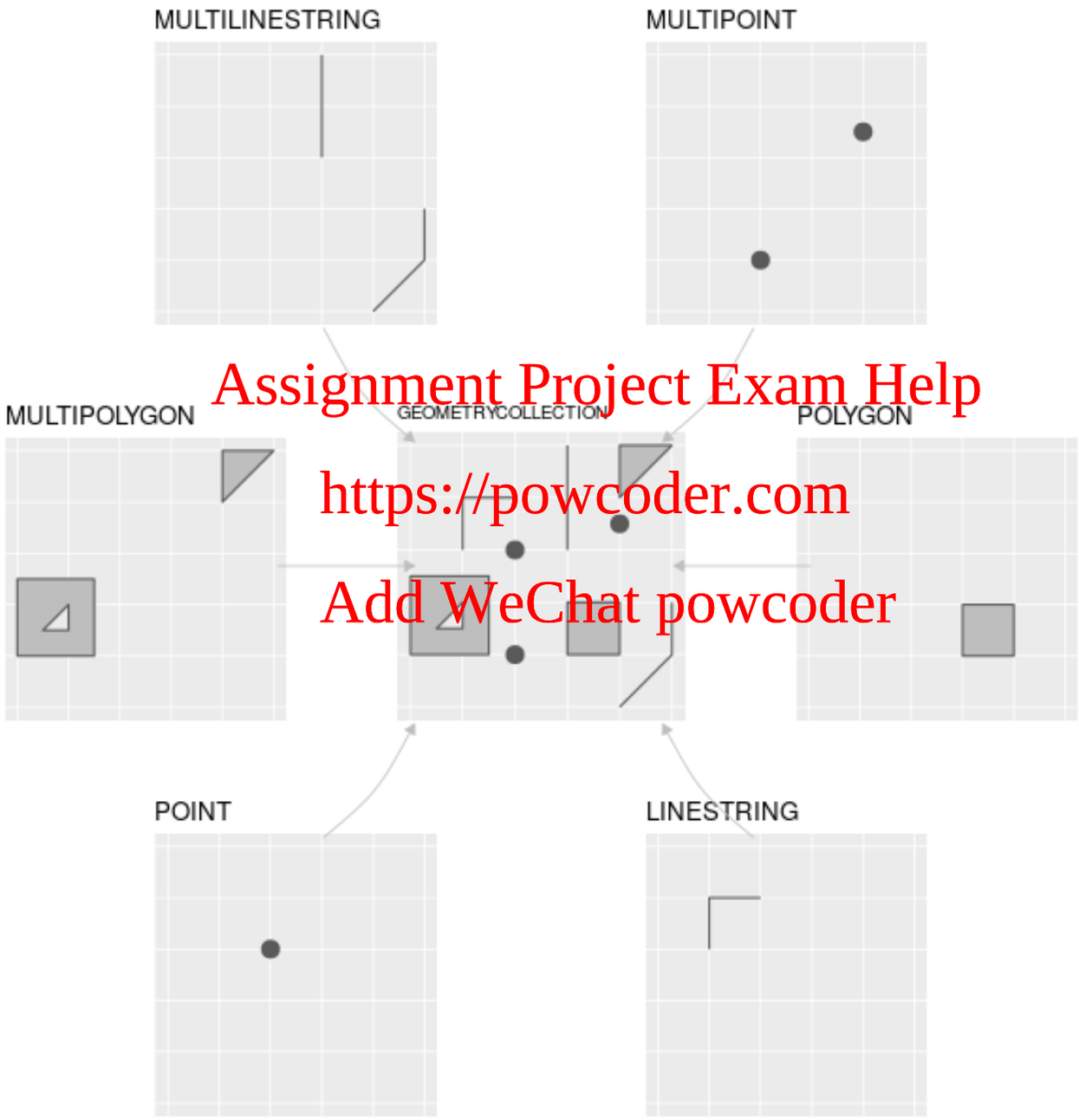


Figure from 'geocomputation with R'

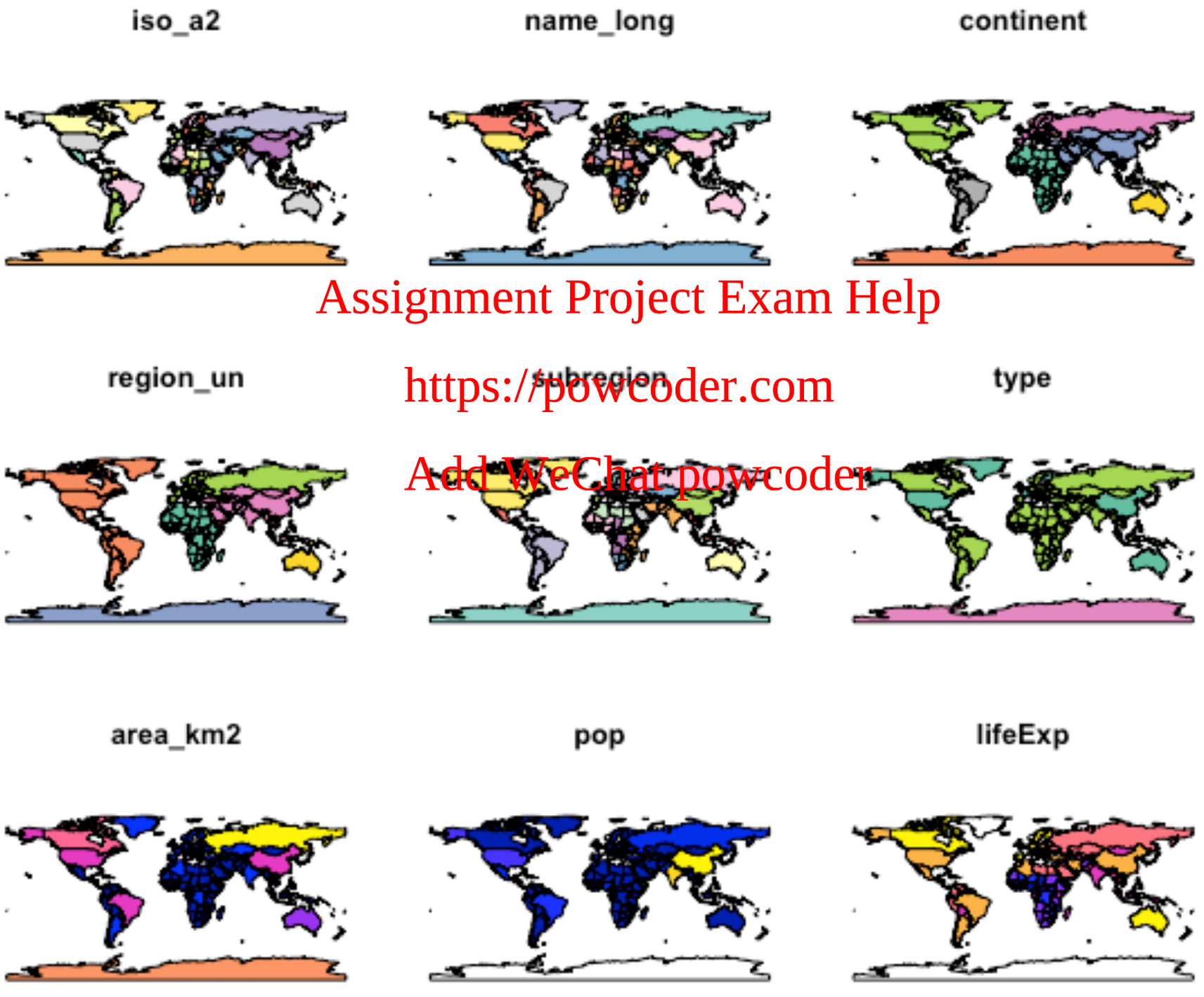
Simple feature geometry types

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

From https://r-spatial.github.io/sf/articles/sf1.html

```
## library(spData)
plot(world)
```

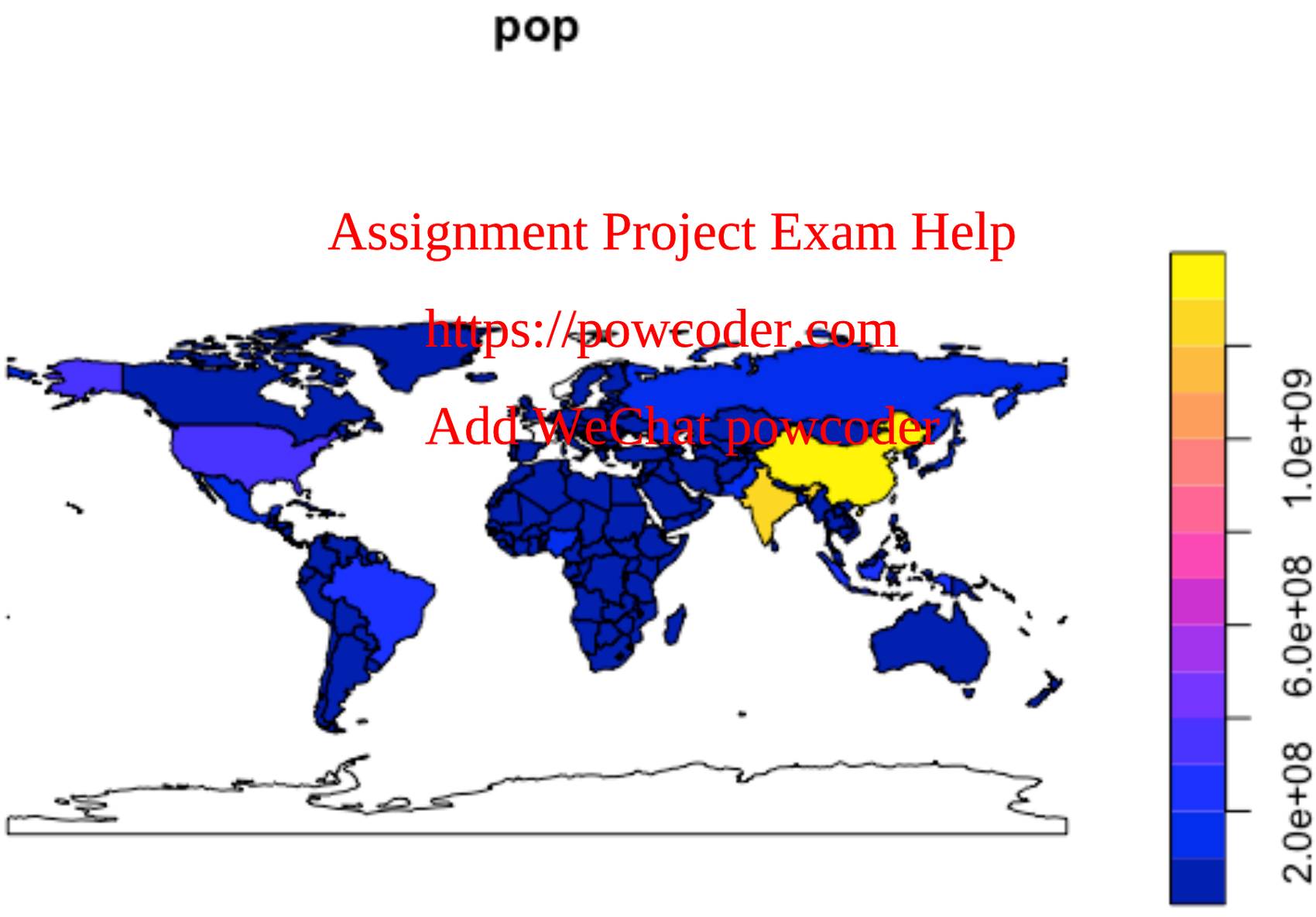


library(spData)

world

```
> world
Simple feature collection with 177 features and 10 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
geographic CRS: WGS 84
# A tibble: 177 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 FJ     Fiji       Oceania   Oceania   Melanesia Sove…   1.93e4  8.86e5    70.0     8222. (((180 -16.06713, 180 -1…
 2 TZ     Tanzania   Africa    Africa    Eastern … Sove…   9.33e5  5.22e7    64.2     2402. (((33.90371 -0.95, 34.07…
 3 EH     Western S… Africa    Africa    Northern… Unde…   9.63e4  NA        NA         NA  (((-8.66559 27.65643, -8…
 4 CA     Canada     North Am… Americas  Northern… Sove…   1.00e7  3.55e7    82.0    43079. (((-122.84 49, -122.9742…
 5 US     United St… North Am… Americas  Northern… Coun…   9.51e6  3.19e8    78.8    51922. (((-122.84 49, -120 49, …
 6 KZ     Kazakhstan Asia      Asia      Central … Sove…   2.73e6  1.73e7    71.6    23587. (((87.35997 49.21498, 86…
 7 UZ     Uzbekistan Asia      Asia      Central … Sove…   4.61e5  3.08e7    71.0     5371. (((55.96819 41.30864, 55…
 8 PG     Papua New… Oceania   Oceania   Melanesia Sove…   4.65e5  7.76e6    65.2     3709. (((141.0002 -2.600151, 1…
 9 ID     Indonesia  Asia      Asia      South-Ea… Sove…   1.82e6  2.55e8    68.9    10003. (((141.0002 -2.600151, 1…
10 AR     Argentina  South Am… Americas  South Am… Sove…   2.78e6  4.30e7    76.3    18798. (((-68.63401 -52.63637, …
# … with 167 more rows
```

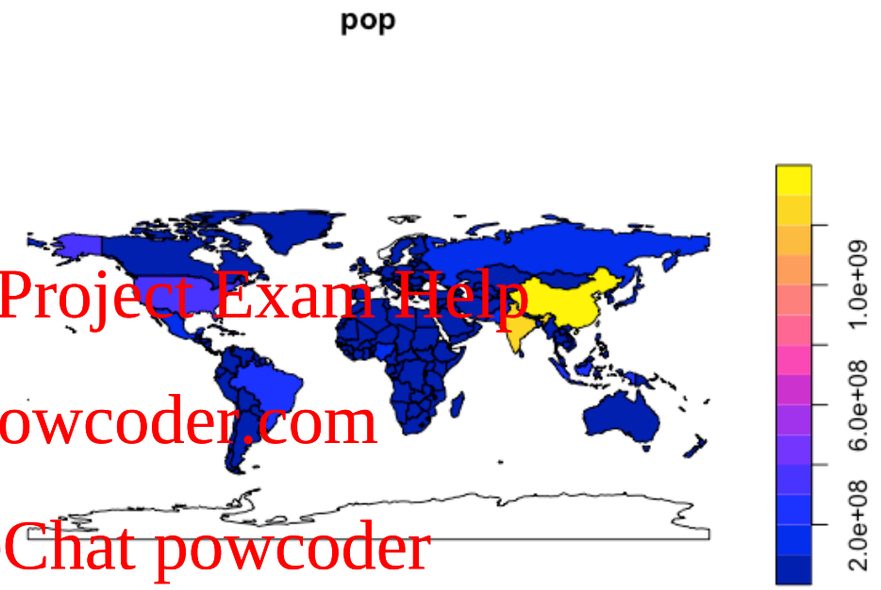
```
plot(world["pop"])
```



SPATIAL DATA

- How is the data structured here?

plot(world["pop"])



ASIA

```
> world_asia = world[world$continent == "Asia", ]
> asia = st_union(world_asia)
> plot(asia)
```

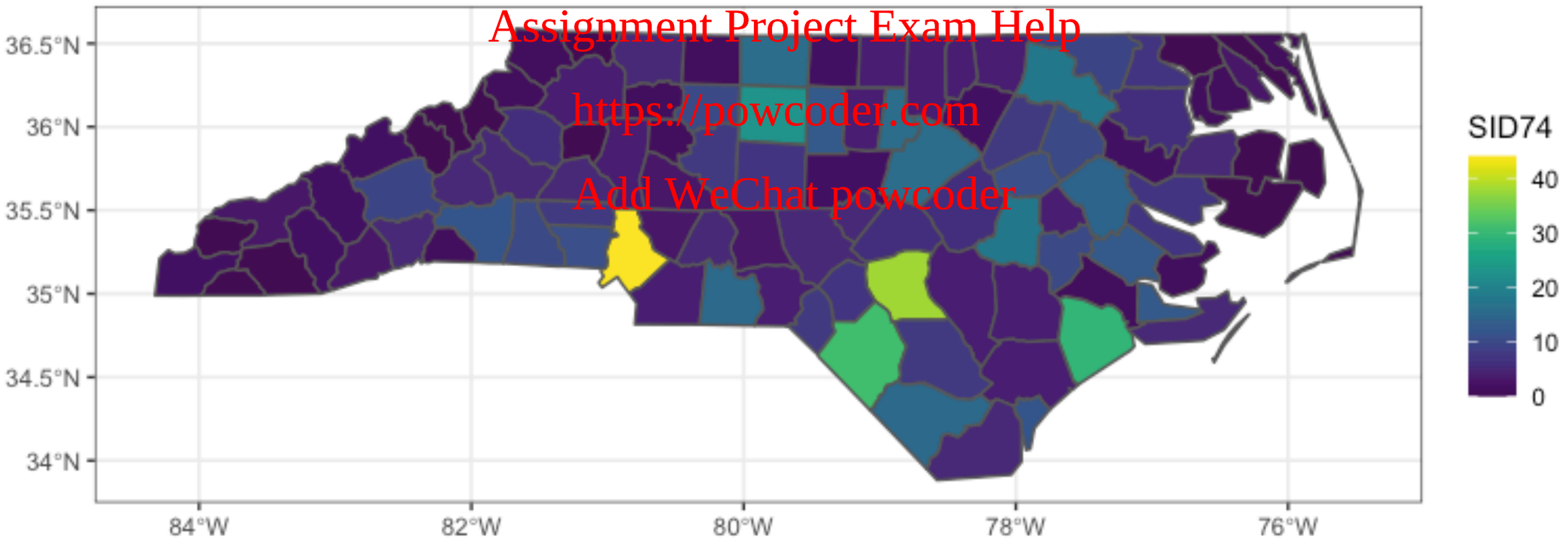

```
> world_asia = world[world$continent == "Asia", ]
> india = world[world$name_long == "India", ]
> plot(st_geometry(india), expandBB = c(0, 0.2, 0.1, 1), col = "gray", lwd =
 3)
> plot(world_asia[0], add = TRUE)
```

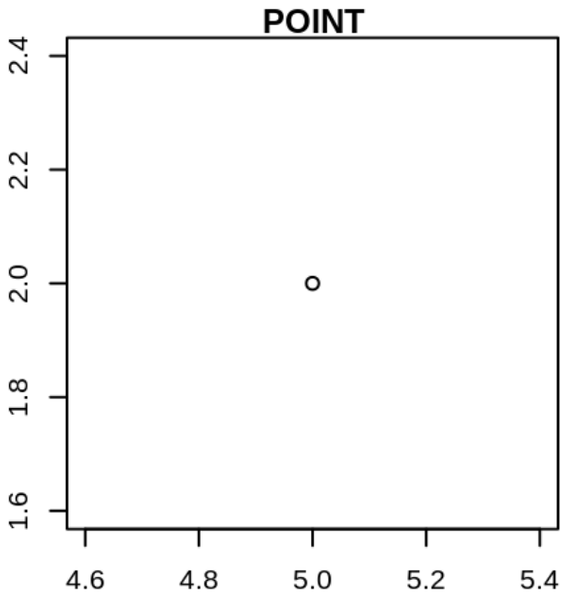

```
> library(sf)
> library(ggplot2)
> library(viridis)
> nc <- st_read(system.file("shape/nc.shp", package = "sf"),
+               quiet = TRUE
+ )
```

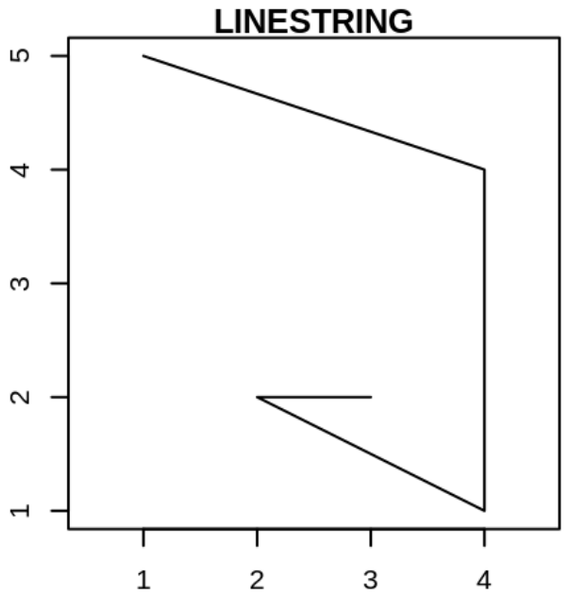


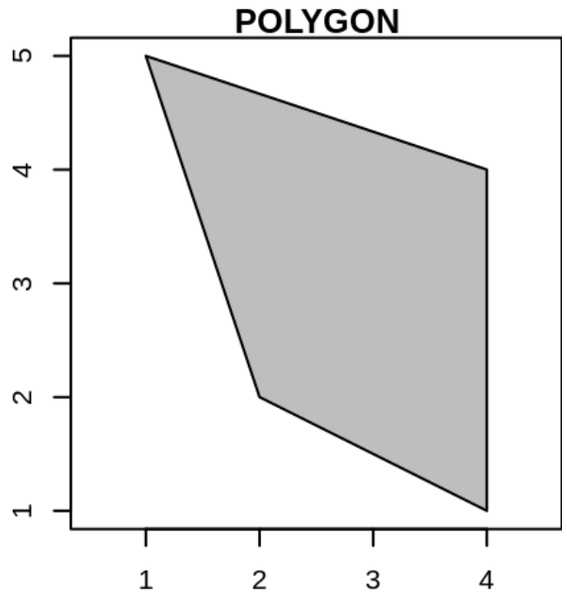
```
> nc
Simple feature collection with 100 features and 14 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
geographic CRS: NAD27
First 10 features:
    AREA PERIMETER CNTY_ CNTY_ID        NAME  FIPS FIPSNO CRESS_ID BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                       geometry
1  0.114     1.442  1825    1825        Ashe 37009  37009        5  1091     1      10  1364     0      19 MULTIPOLYGON (((-81.47276 3...
2  0.061     1.231  1827    1827   Alleghany 37005  37005        3   487     0      10   542     3      12 MULTIPOLYGON (((-81.23989 3...
3  0.143     1.630  1828    1828       Surry 37171  37171       86  3188     5     208  3616     6     260 MULTIPOLYGON (((-80.45634 3...
4  0.070     2.968  1831    1831   Currituck 37053  37053       27   508     1     123   830     2     145 MULTIPOLYGON (((-76.00897 3...
5  0.153     2.206  1832    1832 Northampton 37131  37131       66  1421     9    1066  1606     3    1197 MULTIPOLYGON (((-77.21767 3...
6  0.097     1.670  1833    1833    Hertford 37091  37091       46  1452     7     954  1838     5    1237 MULTIPOLYGON (((-76.74506 3...
7  0.062     1.547  1834    1834      Camden 37029  37029       15   286     0     115   350     2     139 MULTIPOLYGON (((-76.00897 3...
8  0.091     1.284  1835    1835       Gates 37073  37073       37   420     0     254   594     2     371 MULTIPOLYGON (((-76.56251 3...
9  0.118     1.421  1836    1836      Warren 37185  37185       93   968     4     748  1190     2     844 MULTIPOLYGON (((-78.30876 3...
10 0.124     1.428  1837    1837      Stokes 37169  37169       85  1612     1     160  2038     5     176 MULTIPOLYGON (((-80.02567 3...
```

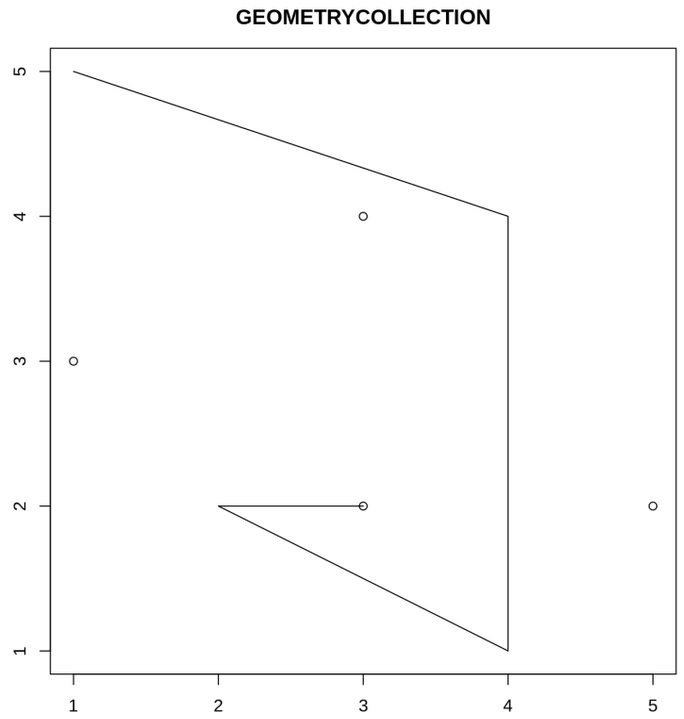
```
> ggplot(data = nc, aes(fill = SID74)) + geom_sf() +
+     scale_fill_viridis() + theme_bw()
```

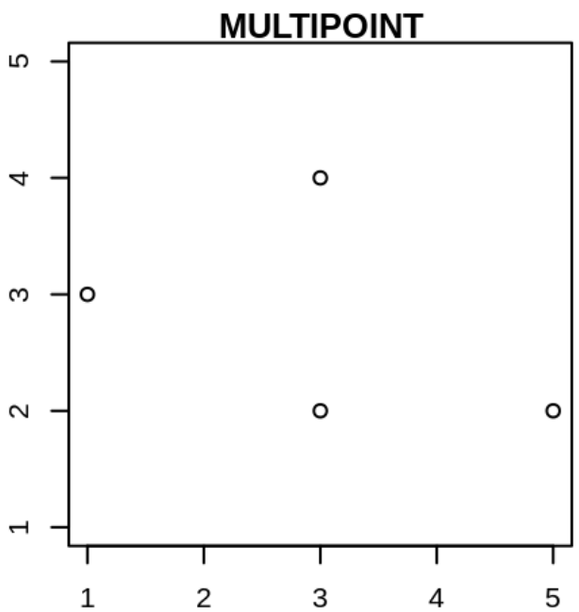



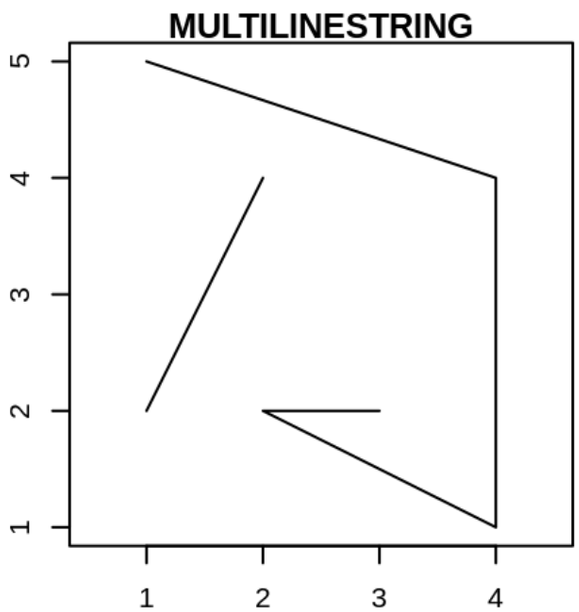


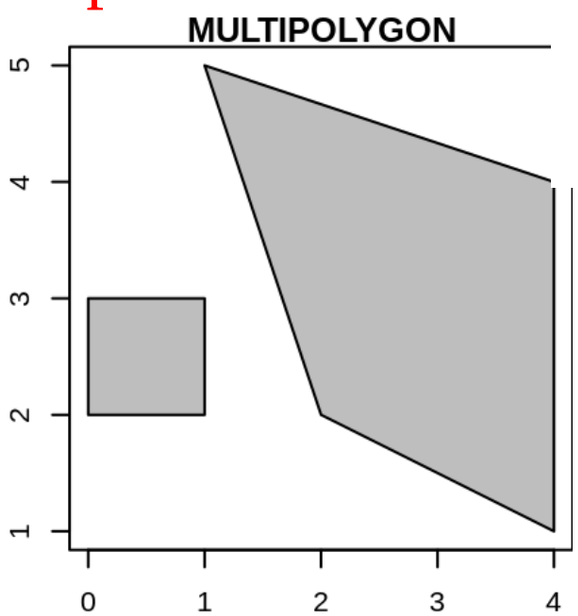












R CODE:

```r
# the rbind function simplifies the creation of matrices
## MULTIPOINT
multipoint_matrix = rbind(c(5, 2), c(1, 3), c(3, 4), c(3, 2))
st_multipoint(multipoint_matrix)
#> MULTIPOINT ((5 2), (1 3), (3 4), (3 2))
## LINESTRING
linestring_matrix = rbind(c(1, 5), c(4, 4), c(4, 1), c(2, 2), c(3, 2))
st_linestring(linestring_matrix)
#> LINESTRING (1 5, 4 4, 4 1, 2 2, 3 2)
```



```r
## POLYGON
polygon_list = list(rbind(c(1, 5), c(2, 2), c(4, 1), c(4, 4), c(1, 5)))
st_polygon(polygon_list)
#> POLYGON ((1 5, 2 2, 4 1, 4 4, 1 5))
```

R CODE:

```r
## MULTILINESTRING
multilinestring_list = list(rbind(c(1, 5), c(4, 4), c(4, 1), c(2, 2), c(3, 2)),
                            rbind(c(1, 2), c(2, 4)))
st_multilinestring((multilinestring_list))
#> MULTILINESTRING ((1 5, 4 4, 4 1, 2 2, 3 2), (1 2, 2 4))
```



```r
## MULTIPOLYGON
multipolygon_list = list(list(rbind(c(1, 5), c(2, 2), c(4, 1), c(4, 4), c(1, 5))),
                         list(rbind(c(0, 2), c(1, 2), c(1, 3), c(0, 3), c(0, 2))))
st_multipolygon(multipolygon_list)
#> MULTIPOLYGON (((1 5, 2 2, 4 1, 4 4, 1 5)), ((0 2, 1 2, 1 3, 0 3, 0 2)))
```

RASTER DATA

RASTER DATA

- Consists of a raster header and matrix — whose rows and columns represent equally spaced cells (aka pixels)
- Raster maps usually represent continuous phenomena such as elevation, population density or spectral data
- Can represent discrete features such as soil or land-cover classes with the help of a raster data model

RASTER DATA

The cell of one raster layer can only hold a single value:

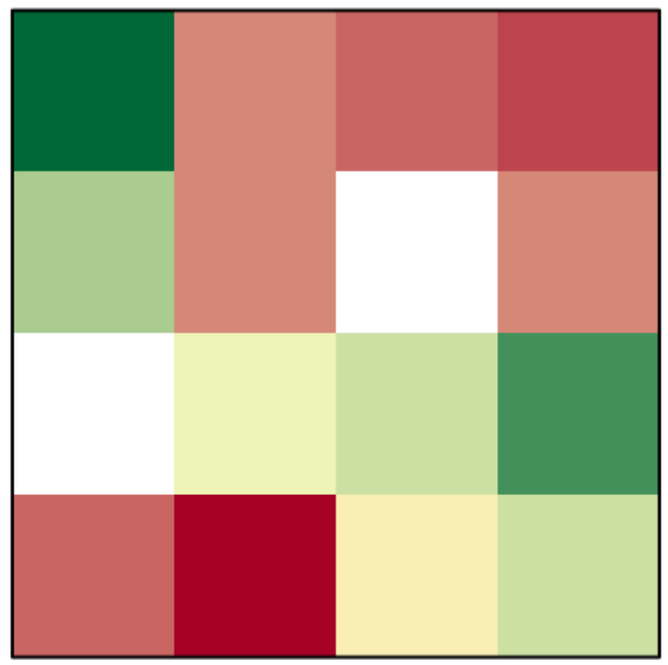
A. Cell IDs

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

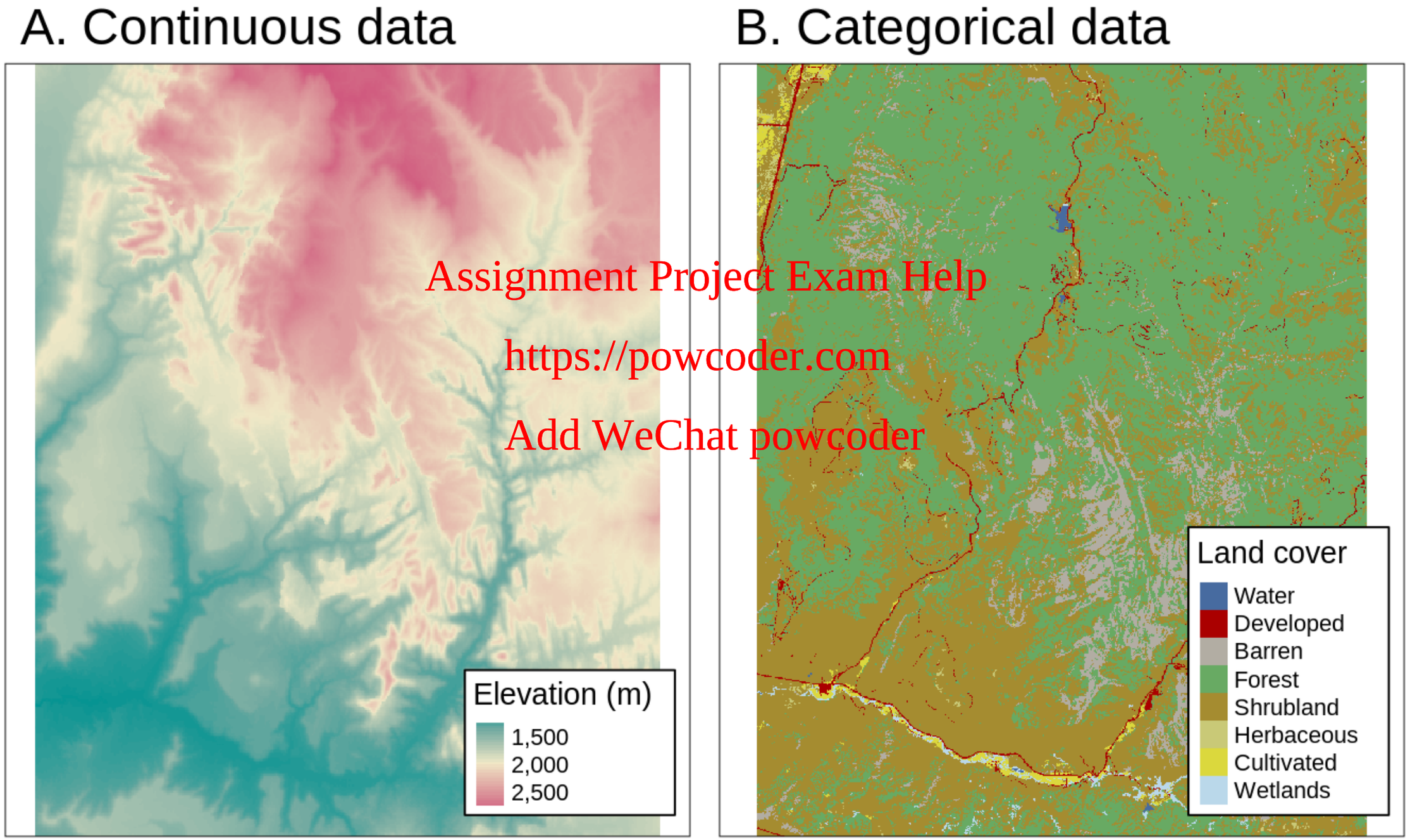
B. Cell values

100	28	22	15
73	31	NA	30
NA	59	62	91
25	6	53	66

C. Colored values



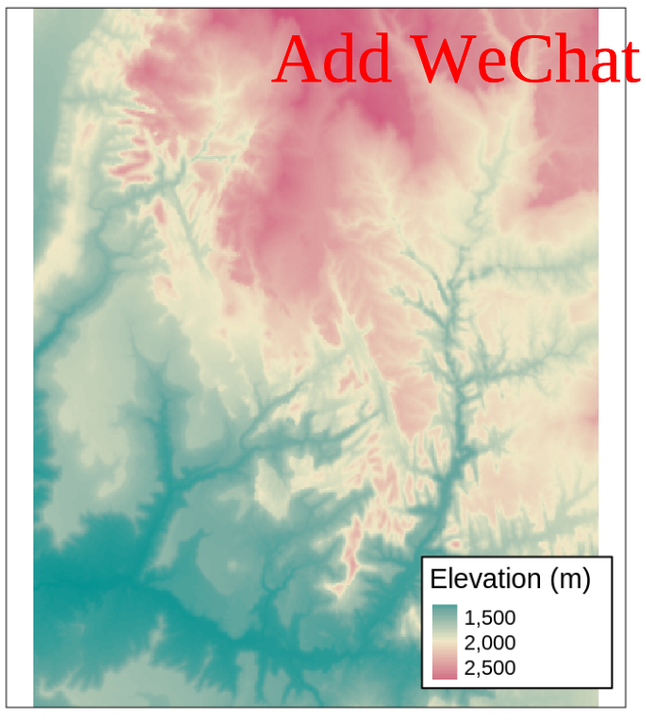
CONTINUOUS AND CATEGORICAL RASTERS



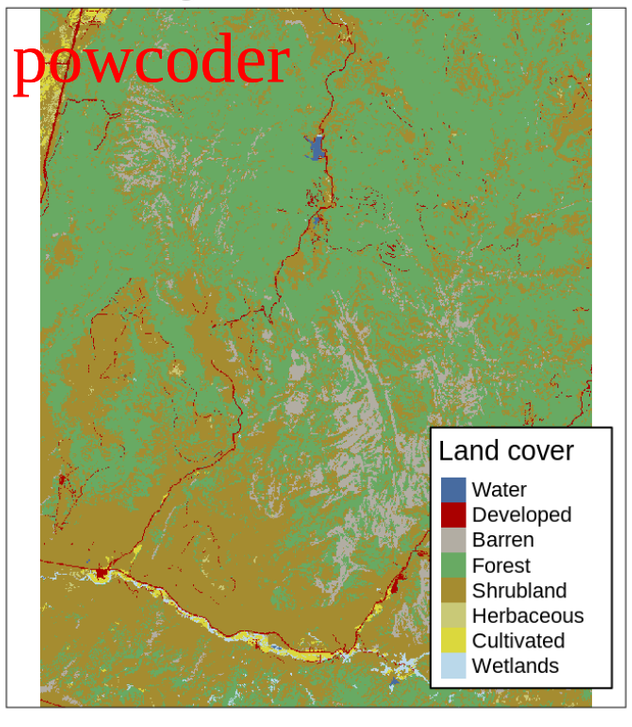
WHAT CAN WE DO?

- Given a coordinate, we can:
  - Identify elevation
  - Connect to land cover type

A. Continuous data



B. Categorical data



DATA STRUCTURE IN R

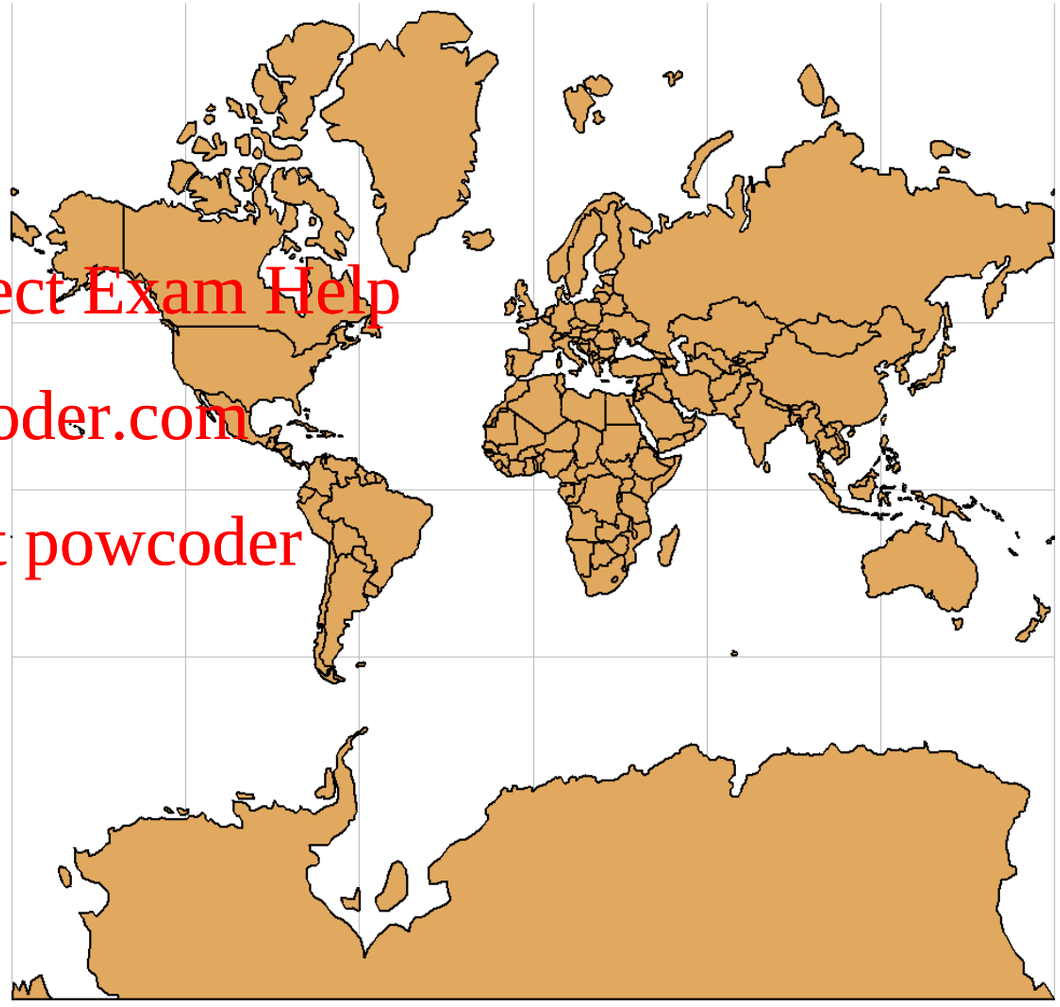
```
raster_filepath = system.file("raster/srtm.tif", package = "spDataLarge")
new_raster = raster(raster_filepath)
```



```
new_raster
#> class       : RasterLayer
#> dimensions  : 457, 465, 212505  (nrow, ncol, ncell)
#> resolution  : 0.000833, 0.000833  (x, y)
#> extent      : -113, -113, 37.1, 37.5  (xmin, xmax, ymin, ymax)
#> coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
#> data source : /home/robin/R/x86_64-pc-linux../3.5/spDataLarge/raster/srtm.tif
#> names       : srtm
#> values      : 1024, 2892  (min, max)
```

COORDINATE REFERENCE SYSTEM (CRS)

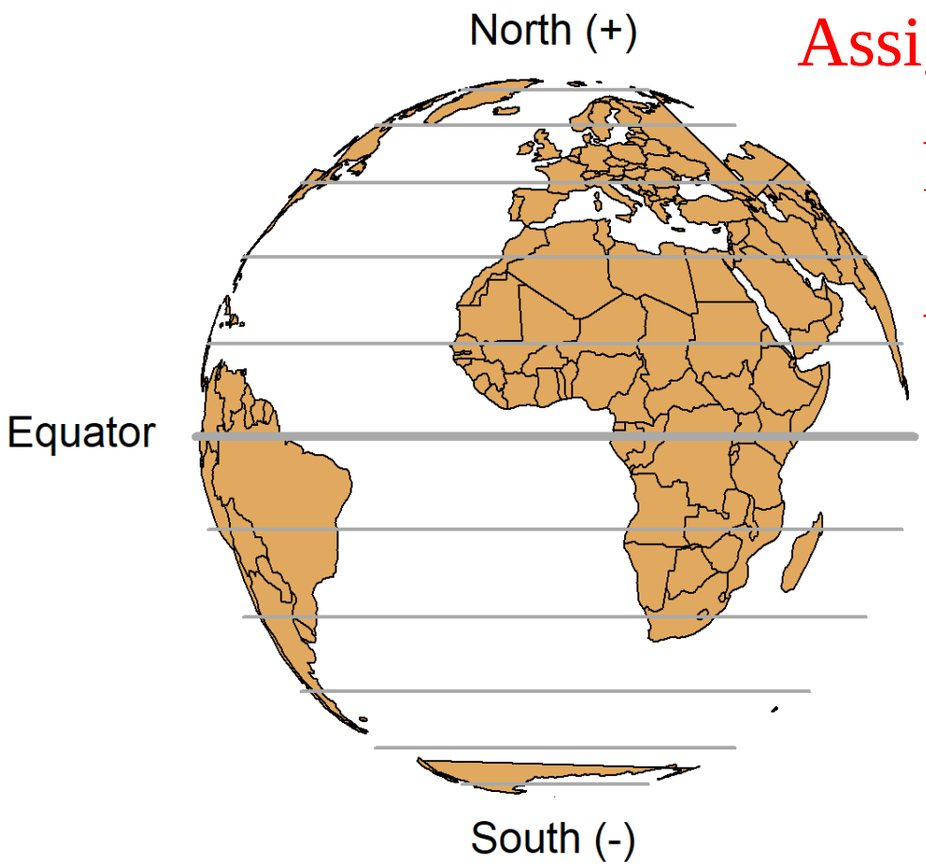
3D AND 2D EARTH SURFACES

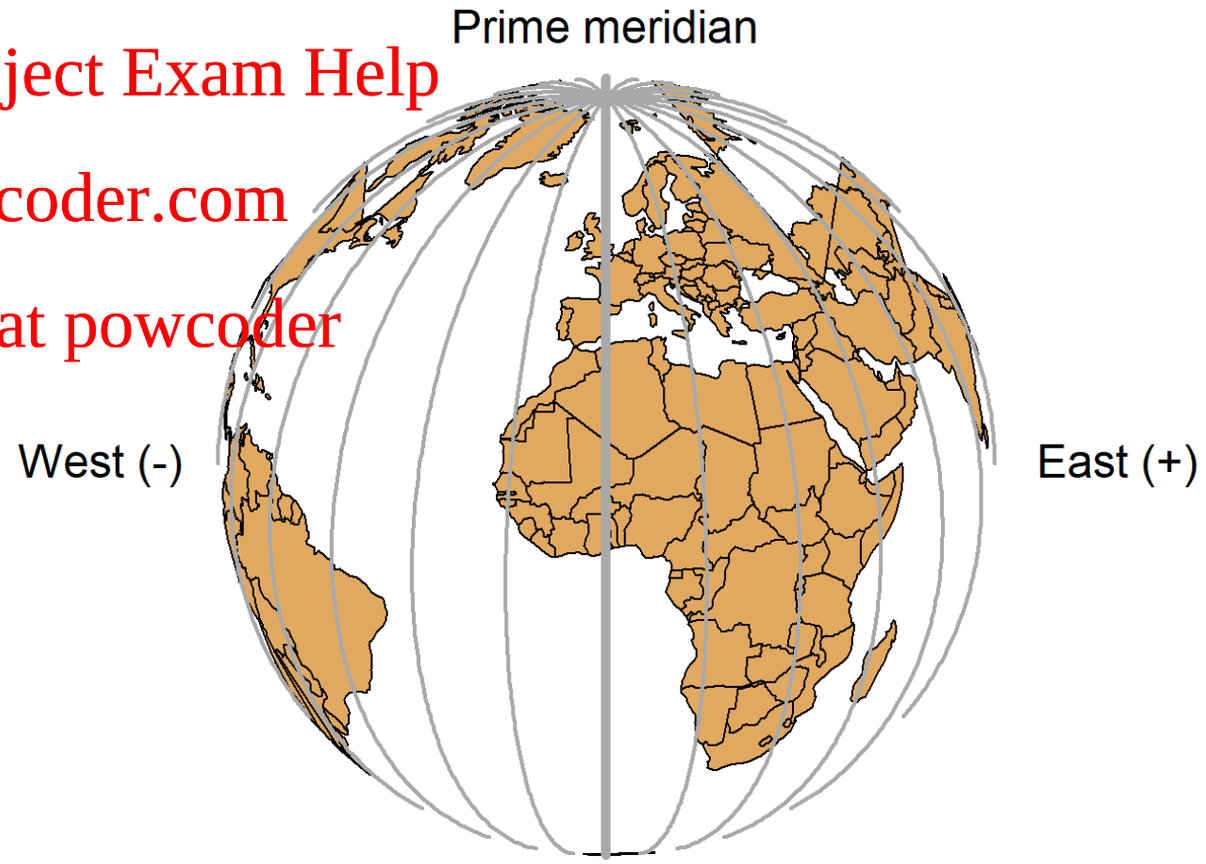


From 'geospatial health data'

GEOGRAPHIC COORDINATE SYSTEMS

A geographic coordinate system specifies locations on the Earth's three-dimensional using latitude and longitude values.





PROJECTED COORDINATE SYSTEMS

- Map projections are transformation of the Earth's three-dimensional surface as a flat two-dimensional plane
- All map projections distort the Earth's surface in some fashion. Can only preserve one to two of the following:
  - Area
  - Direction
  - Shape
  - Distance properties

PROJECTION TYPES

- Conic — projected onto a cone along a single line of tangency or two lines of tangency
    - Best suited for maps of mid-latitude areas

- Cylindrical — maps the surface onto a cylinder
    - Most often used for plotting the world

- Planar — projects data onto a flat surface touching the globe at a point or along a line of tangency
    - Typically used in mapping polar regions

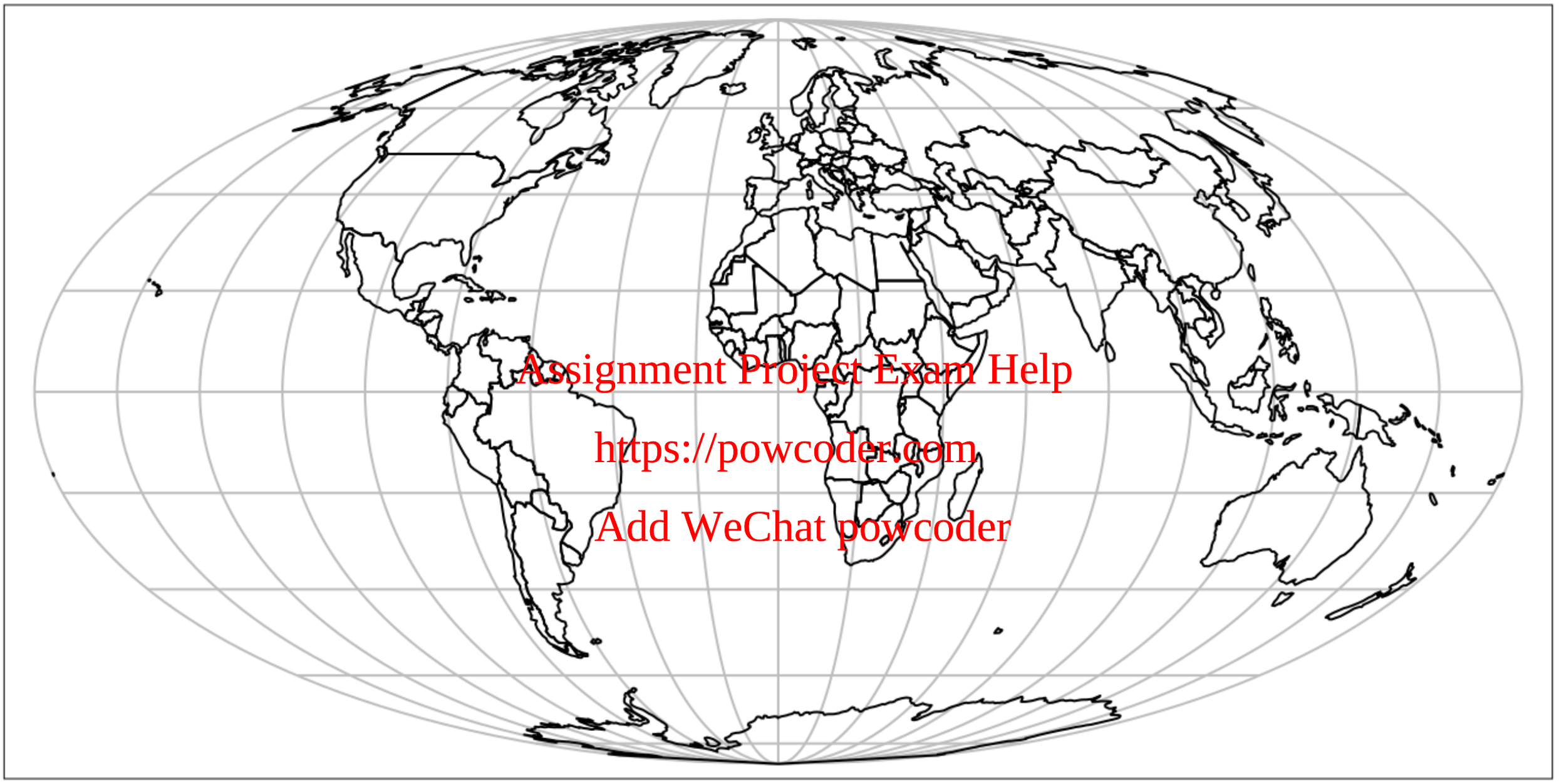


FIGURE 6.2: Mollweide projection of the world.

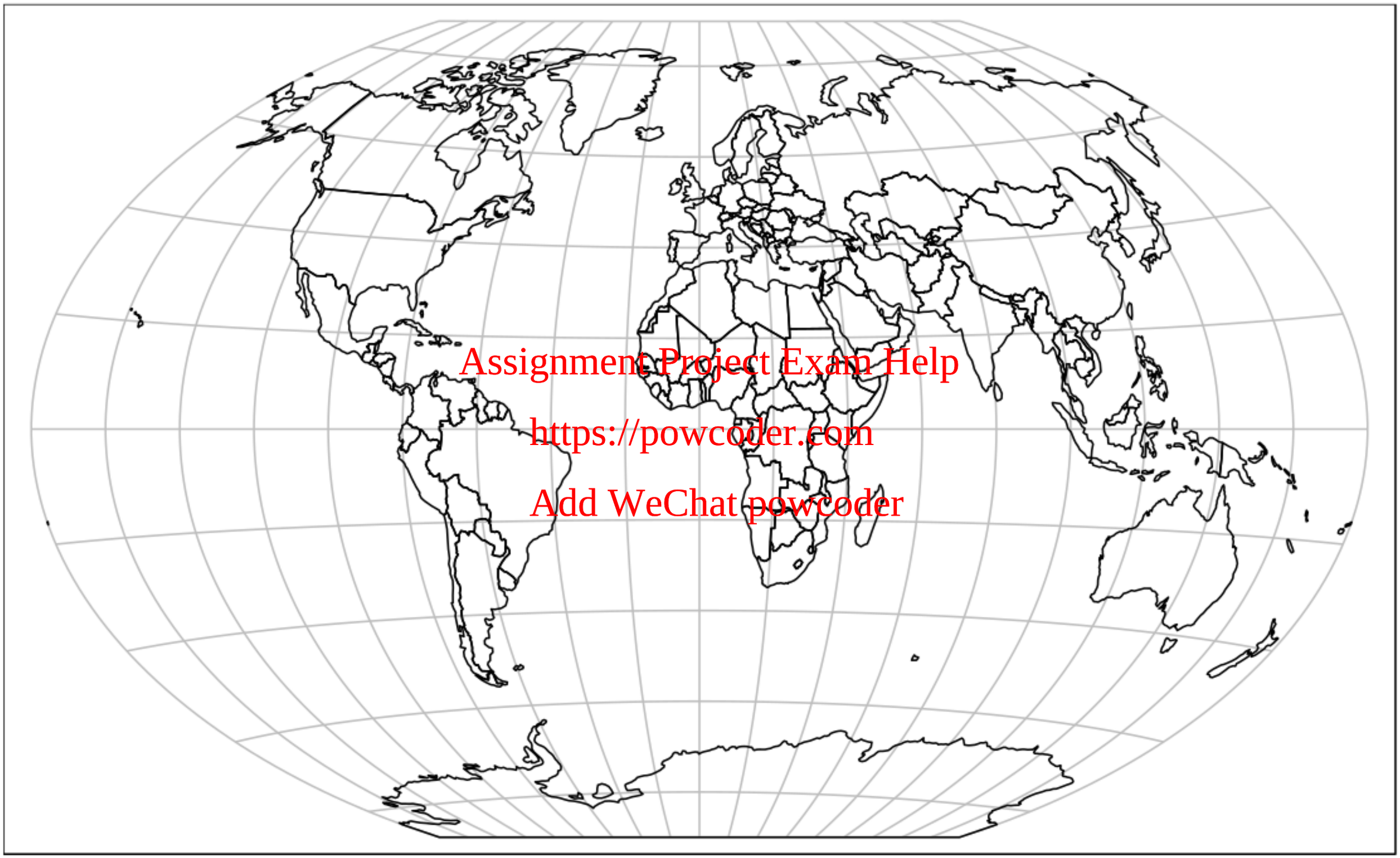


FIGURE 6.3: Winkel tripel projection of the world.

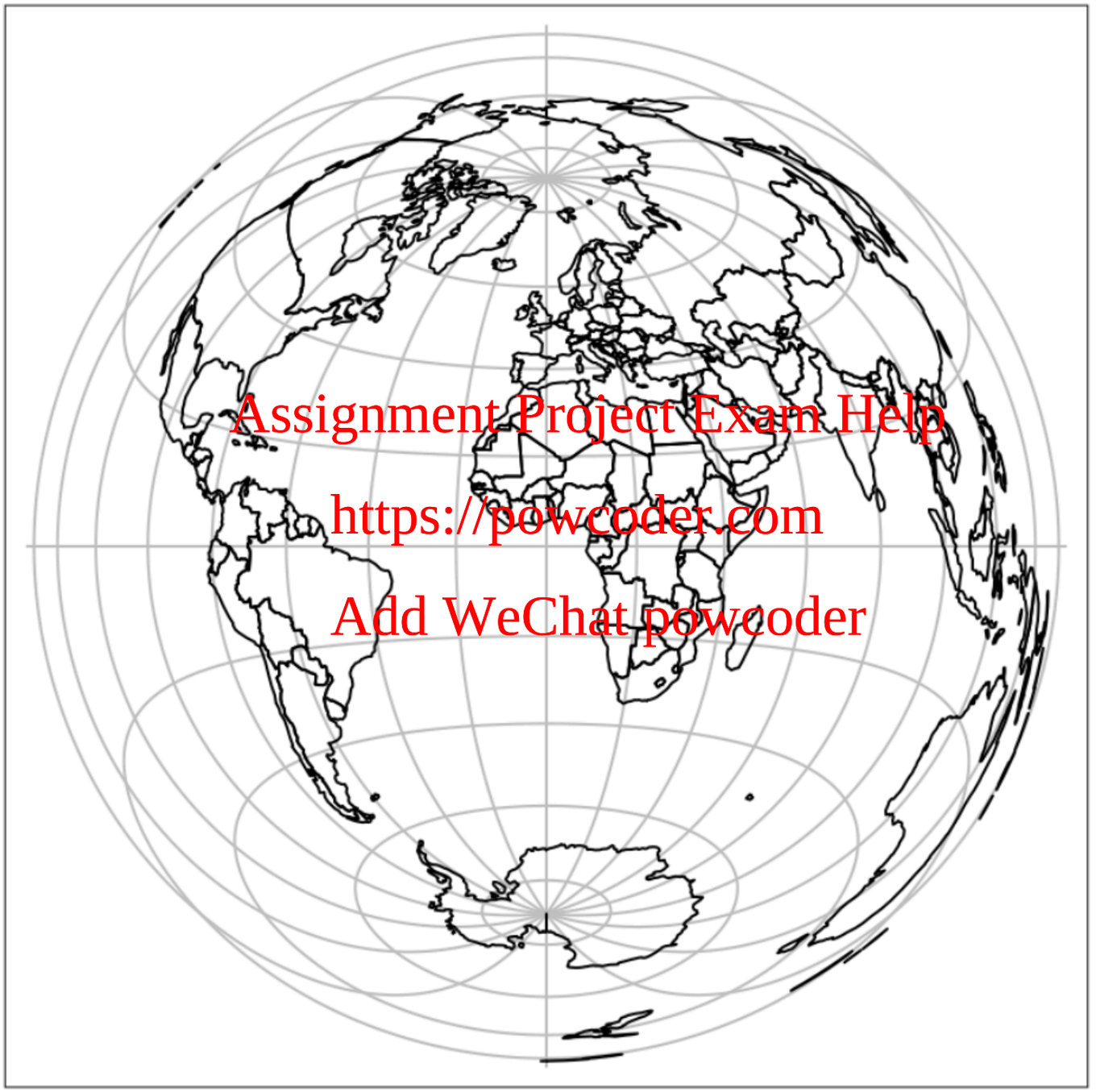


FIGURE 6.4: Lambert azimuthal equal-area projection centered on longitude and latitude of 0.

PROJECTED COORDINATE SYSTEMS

- Universal Transverse Mercator — UTM
  - Preserves local angles and shapes.
  - Divides Earth into 60 zones of 6 degrees of longitude in width
  - Each zone uses transverse Mercator projection to map a region of
- Position on Earth given by UTM zone number, hemisphere, and easting/northing coordinates in zone (measured in meters)
  - Eastings are referenced from the central meridian of each zone, and northings are referenced from the equator. The easting at the central meridian of each zone is defined to have a value of 500,000 meters

BACK TO PREVIOUS DATA SET:

```
> world
Simple feature collection with 177 features and 10 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
geographic CRS: WGS 84
> nc
Simple feature collection with 100 features and 14 fields
geometry type:  MULTIPOLYGON
dimension:      XY
bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
geographic CRS: NAD27
```



```
new_raster
#> class       : RasterLayer
#> dimensions  : 457, 465, 212505  (nrow, ncol, ncell)
#> resolution  : 0.000833, 0.000833  (x, y)
#> extent      : -113, -113, 37.1, 37.5  (xmin, xmax, ymin, ymax)
#> coord. ref. : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
#> data source : /home/robin/R/x86_64-pc-linux../3.5/spDataLarge/raster/srtm.tif
#> names       : srtm
```

CRS OPTIONS

code	note	prj4	prj_method
4265	Monte Mario	+proj=longlat +ellps=intl +no_defs +type=crs	(null)
4266	M'poraloko	+proj=longlat +ellps=clrk80ign +no_defs +type=crs	(null)
4267	NAD27	+proj=longlat +datum=NAD27 +no_defs +type=crs	(null)
4268	NAD27 Michigan	+proj=longlat +a=6378450.04754889 +b=6356826...	(null)
4269	NAD83	+proj=longlat +datum=NAD83 +no_defs +type=crs	(null)
4270	Nahrwan 1967	+proj=longlat +a=6378249.145 +rf=293.465 +no_d...	(null)
4271	Naparima 1972	+proj=longlat +ellps=intl +no_defs +type=crs	(null)


```
> View(rgdal::make_EPSG())
> unique(rgdal::make_EPSG()$prj_method)
 [1] "(null)"                                    "Transverse Mercator"
 [3] "Oblique Stereographic"                     "Transverse Mercator (South Orientated)"
 [5] "Hotine Oblique Mercator (variant B)"       "Lambert Conic Conformal (1SP)"
 [7] "Krovak"                                    "Cassini-Soldner"
 [9] "Mercator (variant A)"                      "Lambert Conic Conformal (2SP)"
[11] "Lambert Azimuthal Equal Area (Spherical)"  "Lambert Conic Conformal (West Orientated)"
[13] "Tunisia Mining Grid"                       "Lambert Conic Near-Conformal"
[15] "Hotine Oblique Mercator (variant A)"       "New Zealand Map Grid"
[17] "American Polyconic"                        "Bonne (South Orientated)"
[19] "Albers Equal Area"                         "Laborde Oblique Mercator"
[21] "Polar Stereographic (variant C)"           "Polar Stereographic (variant B)"
[23] "Lambert Azimuthal Equal Area"              "Lambert Conic Conformal (2SP Belgium)"
[25] "Hyperbolic Cassini-Soldner"                "Transverse Mercator Zoned Grid System"
[27] "Polar Stereographic (variant A)"           "Equidistant Cylindrical (Spherical)"
[29] "Equidistant Cylindrical"                   "Modified Azimuthal Equidistant"
[31] "Mercator (variant B)"                      "Lambert Cylindrical Equal Area (Spherical)"
[33] "Mercator (1SP) (Spherical)"                "Popular Visualisation Pseudo Mercator"
[35] "Guam Projection"                           "Krovak (North Orientated)"
[37] "Krovak Modified"                           "Krovak Modified (North Orientated)"
[39] "Geographic/topocentric conversions"        "Geocentric/topocentric conversions"
[41] "Lambert Conic Conformal (2SP Michigan)"    "Colombia Urban"
[43] "Lambert Cylindrical Equal Area"            "Equal Earth"
```

CRS

- Specifies how coordinates are related to locations on Earth
- CRS is either geographic or projected
- In R, CRS specified using `proj4` strings that specify:
  - Projection
  - Ellipsoid
  - Datum

SELECTING AND SETTING CRS

- Depending on the data set you work with, it may already have a defined CRS — typically comes as a `shape` file (.shp)
- What if it doesn't?
    - There will be choices to be made!
    - Don't make an assumption without thoroughly contacting people who know the data best
    - https://www.spatialreference.org
- There are a some commonly used CRS, for example:
    - WGS84/4326

SETTING AND CHANGING A CRS

```
vector_filepath = system.file("vector/zion.gpkg", package = "spDataLarge")
new_vector = st_read(vector_filepath)
```

```
st_crs(new_vector) # get CRS
#> Coordinate Reference System:
#> No EPSG code
#> proj4string: "+proj=utm +zone=12 +ellps=GRS80 ... +units=m +no_defs"
```



```
new_vector = st_set_crs(new_vector, 4326) # set CRS
#> Warning: st_crs<- : replacing crs does not reproject data; use st_transform for
#> that
```

SETTING AND CHANGING A CRS

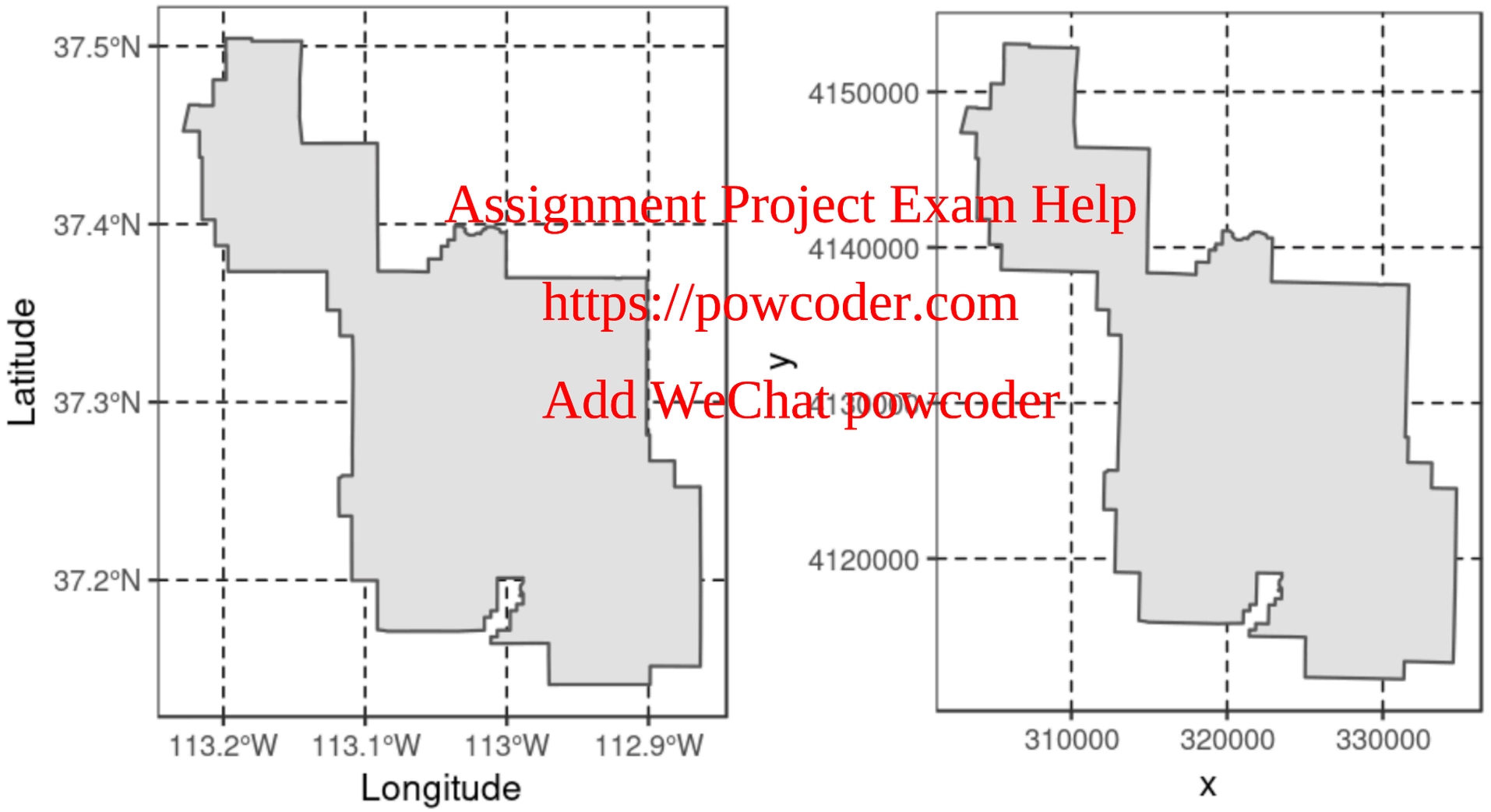


FIGURE 2.13: Examples of geographic (WGS 84; left) and projected (NAD83 / UTM zone 12N; right) coordinate systems for a vector data type.

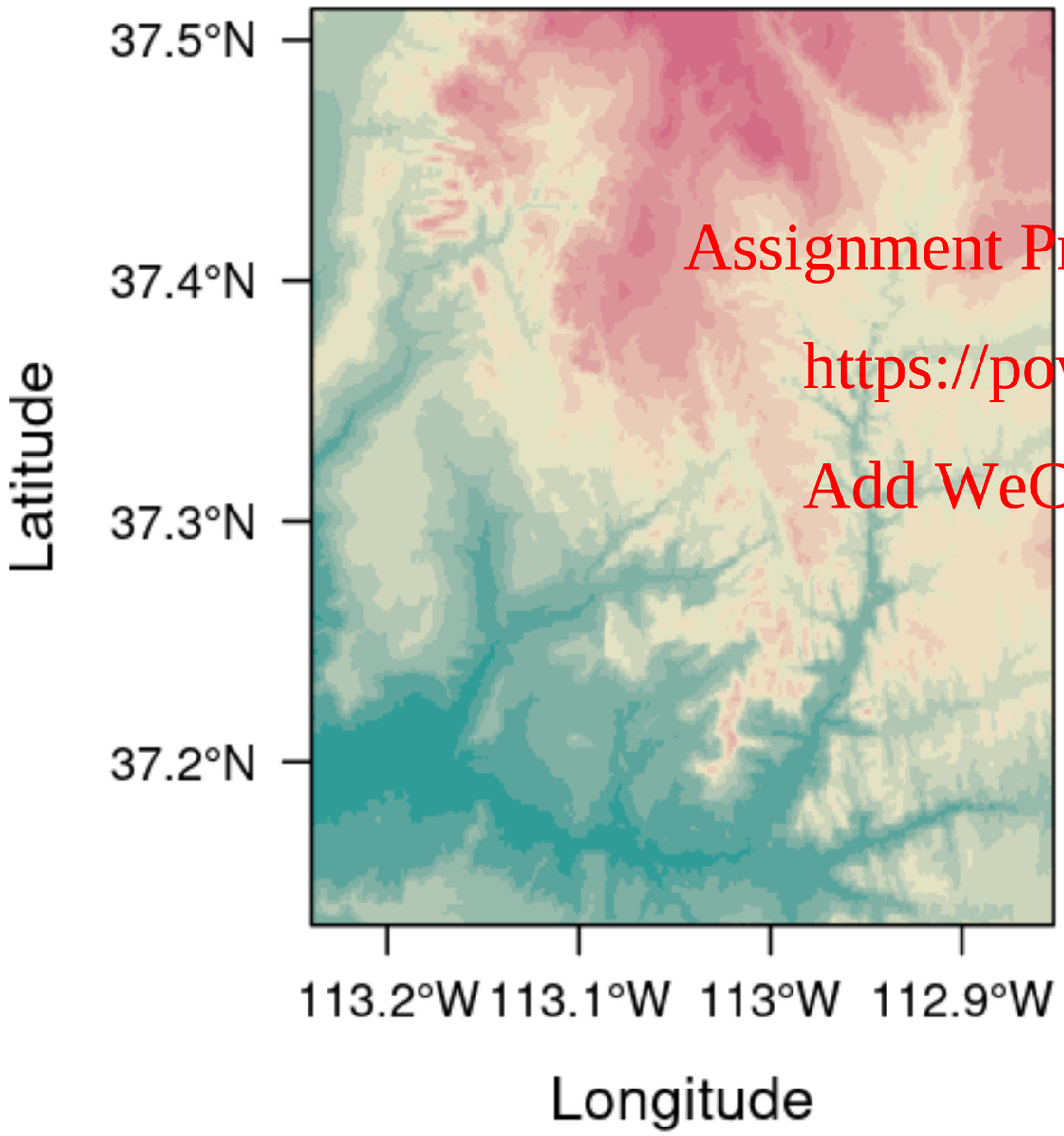
SETTING AND CHANGING A CRS

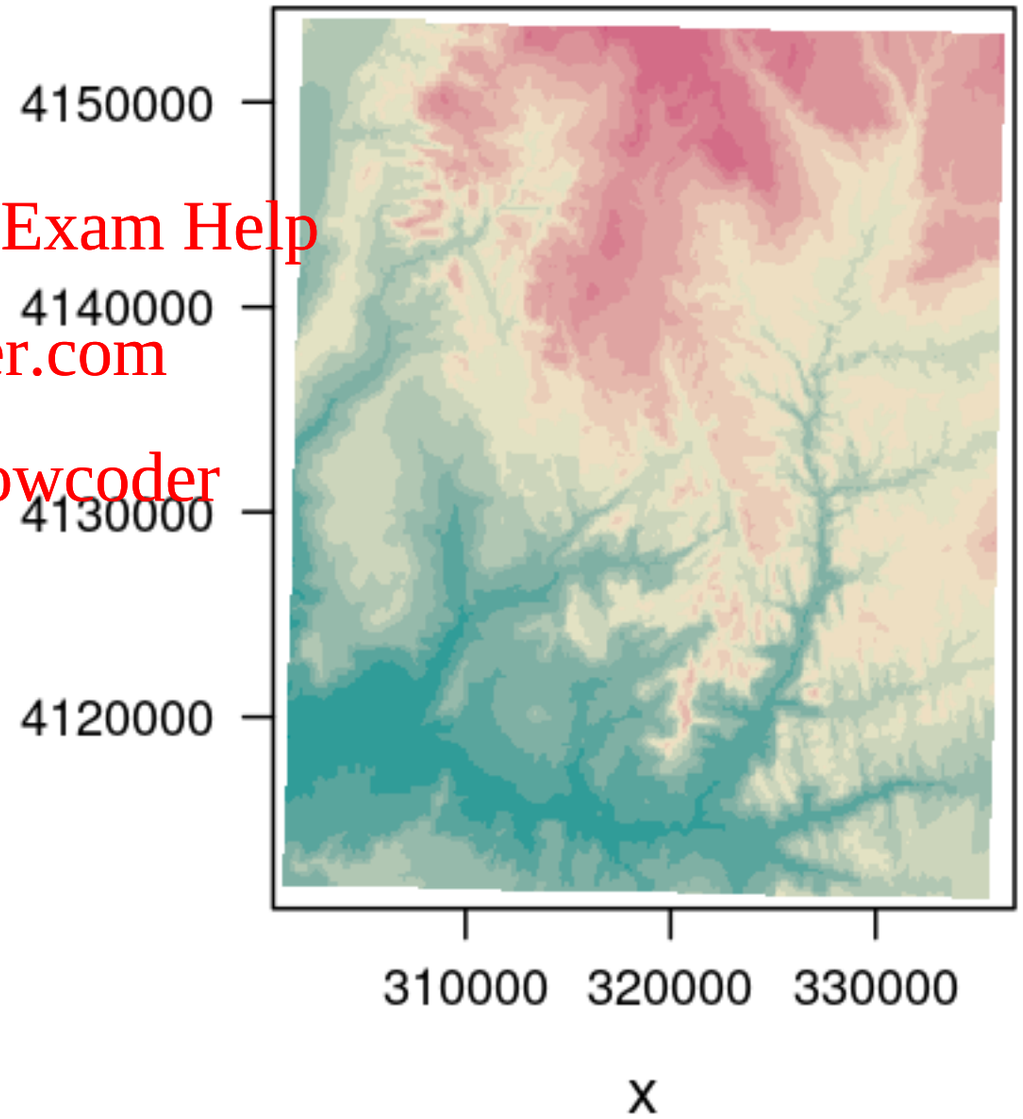
```
projection(new_raster) # get CRS
#> [1] "+proj=longat +datum=WGS84 +no_defs"
```



```
projection(new_raster) = "+proj=utm +zone=12 +ellps=GRS80 +towgs84=0,0,0,0,0,0,0
                          +units=m +no_defs" # set CRS
#> Warning in showSRID(uprojargs, format = "PROJ", multiline = "NO", prefer_proj =
#> prefer_proj): Discarded datum Unknown based on GRS80 ellipsoid in CRS definition
```

SETTING AND CHANGING A CRS





MANIPULATING SPATIAL DATA STRUCTURES

COMBINING/MODIFYING SPATIAL DATA STRUCTURES

- You want to combine two data sets — but they have different CRS…

- You have two data sets, one with points and one with multipolygons — how do you combine them?

- Overlaying multipolygons, strings, points

- Computing distances, areas from spatial data structure!