LAB7

Hector Carvajal

2025-02-25

Lab 7

In this lab we will practice working with raster data, in this case in the context of climate models. I have given you 4 sets of data:

1. Climate Model Data_Historic - this is a NetCDF file with output from a climate model. Data is monthly average air temperature for 1920-2005
2. Climate Model Data_Future - this is a NetCDF file with climate model output for the period 2006-2080
3. Observed Temp Data - this is gridded data based on weather station and satellite data. Data is monthly for 1991-2000
4. Population data - gridded counts of population for a number of years

The first part of the lab will compare modeled and observed climate data for major cities around the world. The second part of the lab will combine the population data and future climate data to project future changes in global temperature.

Part 1

1a. Read in the historic climate model data as a SpatRaster. Use "TREFHT" (temperature at reference height) in the subds (sub-dataset) argument.

```
list.files("/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Climate Model Data_Historic
```

```
## [1] "b.e11.B20TRC5CNBDRD.f09_g16.002.cam.h0.TREFHT.192001-200512.nc"
## [2] "desktop.ini"
```

```
library(terra)
```

```
## terra 1.8.21
```

```
#file path
nc_file <- "/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Climate Model Data_Historic
file.exists(nc_file) #verifu it exists
```

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

```
#read in NetCDF file as a spatraster using "TREFHT"(given in instructions)
nc_raster <- rast(nc_file, subds = "TREFHT")
print(nc_raster)
```

```
## class       : SpatRaster
## dimensions  : 192, 288, 1032  (nrow, ncol, nlyr)
## resolution  : 1.25, 0.9424084  (x, y)
## extent      : -0.625, 359.375, -90.4712, 90.4712  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source      : b.e11.B20TRC5CNBDRD.f09_g16.002.cam.h0.TREFHT.192001-200512.nc:TREFHT
## varname     : TREFHT (Reference height temperature)
## names       : TREFHT_1, TREFHT_2, TREFHT_3, TREFHT_4, TREFHT_5, TREFHT_6, ...
## unit        :        K,        K,        K,        K,        K,        K, ...
## time (days) : 1920-02-01 to 2006-01-01
```

1b. Use ext() to see the longitude and latitude range of the SpatRaster you created. Note that the longitude goes form 0 to 360 (ish) instead of the more typical -180 to 180. This will cause a problem later on so use the rotate() function to change the longitude coordinates. Use extent again on the rotated object to check the longitude goes from -180 to 180 (ish)

```
ext(nc_raster)
```

```
## SpatExtent : -0.625, 359.375, -90.4712041884817, 90.4712041884817 (xmin, xmax, ymin, ymax)
```

```
nc_raster_rotated <- rotate(nc_raster)
ext(nc_raster_rotated)
```

```
## SpatExtent : -180.625, 179.375, -90.4712041884817, 90.4712041884817 (xmin, xmax, ymin, ymax)
```

2a. Use `rnaturalearth::ne_download()` function to get a sf object of major cities ("populated_places"). Use `vect` to coerce this to a SpatVector, and subset it to get just the 10 most populous cities based on 2020 population (POP2020 column)

```
library(rnaturalearth)
library(terra)
```

```
#major cities as an sf object
cities_sf <- ne_download(scale = 10, type = "populated_places", category = "cultural", returnclass = "s
```

```
## Reading layer 'ne_10m_populated_places' from data source
##   '/private/var/folders/95/_8c2c4nx3bdg71_0vyw5_9v40000gn/T/RtmpkHvPPl/ne_10m_populated_places.shp'
##   using driver 'ESRI Shapefile'
## Simple feature collection with 7342 features and 137 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: -179.59 ymin: -90 xmax: 179.3833 ymax: 82.48332
## Geodetic CRS:  WGS 84
```

```
cities_vect <- vect(cities_sf)
```

```
#remove NA values before sorting (better cleaning habit for me)
cities_vect_clean <- cities_vect[!is.na(cities_vect$POP2020), ]
#sort by population
cities_vect_sorted <- cities_vect_clean[order(cities_vect_clean$POP2020, decreasing = TRUE), ]
```

```r
#top 10 cities selected
top10_cities <- cities_vect_sorted[1:10, ]
#had a hard time knowing if it worked so searched up how to display the cities and found that you can d
as.data.frame(top10_cities[, c("NAME", "POP2020")])
```

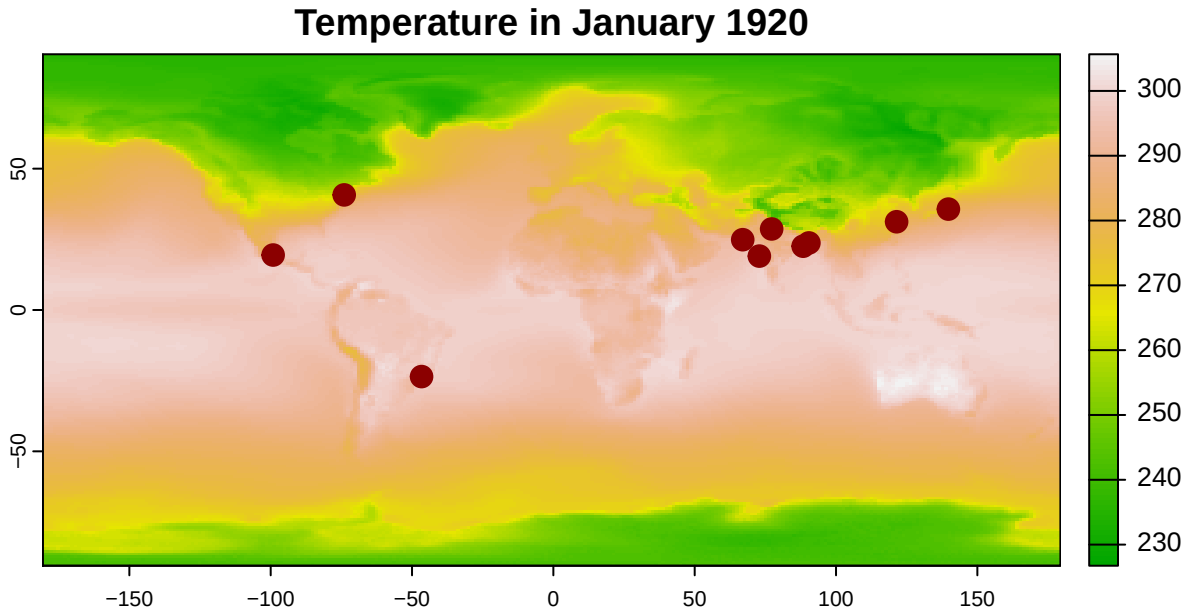
```
##           NAME POP2020
## 1        Tokyo   36371
## 2       Mumbai   21946
## 3    São Paulo   20544
## 4  Mexico City   20189
## 5     New York   19974
## 6        Delhi   18669
## 7     Shanghai   17214
## 8      Kolkata   17039
## 9        Dhaka   17015
## 10     Karachi   14855
```

2b. Make a plot of the temperature data for January 1920 and overlay the 10 major cities.

```r
#January 1920 temperature data extracted
jan1920_temp <- nc_raster_rotated[[1]]
plot(jan1920_temp, main = "Temperature in January 1920", col = terrain.colors(100))
```

```
## Warning: [is.lonlat] coordinates are out of range for lon/lat
```

```r
#overlay the 10 most populous cities
points(top10_cities, col = "darkred", pch = 19, cex = 1.5)
```



2c. What about the plot gives you confidence this is actually showing temperature data from a January? What are the units of the temperature data?

Answer

I'm confident its displaying the correct temperatures for January because the green parts displayy the colder temperautres which is to be expected (winter temps) and warmer temperatures as you approach the southern hemisphere. Units are in Kelvin since the ranges are from ~230 to ~300. Wouldn't work for Celsius since it would be impossible for earth to handle.

3a. Read in the observed temperature data as a SpatRaster, using "tmp" for the sub-dataset argument

```
list.files("/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Observed Temp Data/")
```

```
## [1] "cru_ts4.03.1991.2000.tmp.dat.nc" "desktop.ini"
```

```
#file path
obs_temp_file <- "/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Observed Temp Data/c
#observed temperature data as spatraster
obs_temp_raster <- rast(obs_temp_file, subds = "tmp")
print(obs_temp_raster)
```

```
## class       : SpatRaster
## dimensions  : 360, 720, 120  (nrow, ncol, nlyr)
## resolution  : 0.5, 0.5  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (CRS84) (OGC:CRS84)
```

```
## source      : cru_ts4.03.1991.2000.tmp.dat.nc:tmp
## varname     : tmp (near-surface temperature)
## names       :           tmp_1,           tmp_2,           tmp_3,           tmp_4,           tmp_5,
## unit        : degrees Celsius, degrees Celsius, degrees Celsius, degrees Celsius, degrees Celsius, d
## time (days) : 1991-01-16 to 2000-12-16
```

3b. Note that this climate model data is for 1920-2005 but the observation data is only from 1991-2000. Subset the climate model data to just the years 1991-2000. Also change the units to match those of the observed climate data.

```
#extracting time information from the climate model data
time_info <- time(nc_raster_rotated)

#converting the time values to years
years <- as.numeric(format(time_info, "%Y"))
#sub-setting the data to only include 1991-2000
nc_raster_subset <- nc_raster_rotated[[years >= 1991 & years <= 2000]]
nc_raster_celsius <- nc_raster_subset - 273.15 #unit conversion
print(nc_raster_celsius)
```

```
## class       : SpatRaster
## dimensions  : 192, 288, 120  (nrow, ncol, nlyr)
## resolution  : 1.25, 0.9424084  (x, y)
## extent      : -180.625, 179.375, -90.4712, 90.4712  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## varname     : TREFHT (Reference height temperature)
## names       : TREFHT_852, TREFHT_853, TREFHT_854, TREFHT_855, TREFHT_856, TREFHT_857, ...
## min values  :  -44.42032,  -48.08484,  -47.13795,  -56.85676,  -61.24125,  -66.53745, ...
## max values  :   31.83303,   33.57092,   30.72607,   32.64471,   35.56732,   38.02841, ...
## time (days) : 1991-01-01 to 2000-12-01
```

4. Use terra::extract() to produce two data-frames, one with observed and one with modeled temperature values for each city. Change the units of the modeled data so they match the units of the observed data.

```
library(terra)

#extract observed and simulated temperature data
city_obs <- terra::extract(obs_temp_raster, top10_cities)
city_sim <- terra::extract(nc_raster_celsius, top10_cities)
print(dim(city_obs))
```

```
## [1]  10 121
```

```
print(dim(city_sim))
```

```
## [1]  10 121
```

```
head(city_obs)
```

```
##   ID tmp_1 tmp_2     tmp_3 tmp_4 tmp_5 tmp_6 tmp_7 tmp_8 tmp_9 tmp_10 tmp_11
## 1  1   4.9   4.9  8.900001  14.5  18.1  23.0  25.8  25.6  23.4   17.3   11.7
## 2  2  23.1  24.5 26.800001  29.0  30.6  29.4  27.4  27.1  27.7   28.2   27.8
## 3  3  21.7  22.4 21.900000  20.6  18.5  17.7  16.1  17.4  18.3   19.8   21.3
## 4  4  11.1  12.2 15.400001  16.4  16.7  16.5  15.2  15.6  14.1   13.5   11.1
## 5  5   0.8   3.8  6.600000  12.1  19.1  22.3  24.6  24.5  19.4   14.2    8.5
## 6  6  13.0  16.9 22.300001  27.2  32.3  33.2  32.2  29.5  29.3   25.6   19.8
##   tmp_12    tmp_13 tmp_14 tmp_15    tmp_16 tmp_17 tmp_18 tmp_19 tmp_20 tmp_21
## 1    8.0  5.800000    5.4    9.1 13.900001   16.6   20.5   24.8   26.3   22.2
## 2   25.4 23.800001   23.4   26.9 28.200001   30.4   30.4   28.5   27.3   27.4
## 3   22.1 22.500000   22.8   22.3 20.500000   18.9   18.5   15.2   16.2   17.3
## 4   11.1  9.400001   11.0   14.5 14.500000   14.8   16.6   15.2   15.0   14.7
## 5    4.2  1.500000    2.1    3.8  9.400001   15.2   20.4   23.1   22.3   19.6
## 6   15.7 14.900001   15.5   22.4 28.000000   31.3   33.8   30.4   29.4   28.4
##   tmp_22 tmp_23 tmp_24 tmp_25 tmp_26 tmp_27 tmp_28 tmp_29 tmp_30 tmp_31 tmp_32
## 1   16.5   12.0    7.8    5.7    6.3    7.5   12.1   17.0   20.9   22.8   24.3
## 2   28.6   27.9   25.9   24.5   24.8   26.6   28.5   30.7   30.3   28.0   28.1
## 3   19.6   20.3   21.3   23.0   22.3   22.7   21.4   18.0   16.3   16.5   16.4
## 4   13.6   11.3   11.8   11.4   12.7   13.5   15.1   15.2   16.0   14.9   15.3
## 5   12.2    7.9    3.3    2.5   -1.3    3.3   10.8   17.3   21.7   25.7   24.3
## 6   25.8   20.3   16.3   14.2   18.8   20.4   28.2   33.1   33.9   30.4   30.5
##   tmp_33 tmp_34 tmp_35 tmp_36 tmp_37 tmp_38 tmp_39 tmp_40 tmp_41 tmp_42 tmp_43
## 1   21.4   16.0   12.8    7.2    4.6    4.9    6.9   14.4   18.9   21.7   27.3
## 2   27.6   29.2   28.3   25.9   24.7   24.3   27.9   28.5   30.7   28.9   26.9
## 3   17.9   20.4   21.0   21.5   22.1   23.9   21.7   20.4   19.4   16.5   16.4
## 4   14.7   13.9   12.0   11.5   10.9   12.4   14.4   14.9   16.4   16.2   15.7
## 5   19.7   13.1    8.2    2.6   -3.5   -1.1    4.5   12.4   15.7   23.0   26.2
## 6   28.8   26.5   21.5   16.2   15.4   16.6   24.2   27.3   33.1   33.9   30.1
##   tmp_44 tmp_45 tmp_46 tmp_47 tmp_48 tmp_49 tmp_50 tmp_51 tmp_52 tmp_53 tmp_54
## 1   28.6   24.0   18.9   12.3    7.5    4.4    5.2    8.1   13.6   18.1   20.4
## 2   27.3   27.5   28.9   27.7   25.2   23.0   24.9   26.4   28.5   30.3   31.0
## 3   16.8   18.5   20.5   21.0   22.5   23.7   23.2   22.5   21.0   18.0   17.2
## 4   14.8   14.6   14.5   13.5   12.1   11.7   13.1   14.3   15.3   16.6   16.6
## 5   23.1   19.8   13.8   10.6    5.4    3.2    0.0    6.9   10.4   15.7   21.4
## 6   29.3   28.7   25.6   20.9   16.2   12.9   16.9   21.3   27.4   33.1   35.0
##   tmp_55 tmp_56 tmp_57 tmp_58 tmp_59 tmp_60 tmp_61 tmp_62 tmp_63 tmp_64 tmp_65
## 1   25.5   28.1   22.0   18.1   10.8    5.7    4.7    4.0    7.7   11.4   17.2
## 2   28.2   28.5   28.1   29.4   27.3   26.6   24.7   25.8   28.4   29.0   30.7
## 3   18.3   19.2   18.5   19.0   21.0   21.6   23.4   23.2   22.3   21.4   18.0
## 4   15.4   14.8   15.2   13.6   12.6   11.1   10.1   12.0   12.8   14.8   16.7
## 5   25.3   25.0   19.4   15.7    6.6    0.3   -0.3    0.8    3.7   10.9   15.6
## 6   31.4   29.1   29.4   27.0   20.7   16.0   14.7   17.9   24.1   29.0   32.1
##   tmp_66 tmp_67 tmp_68 tmp_69 tmp_70 tmp_71 tmp_72 tmp_73 tmp_74 tmp_75 tmp_76
## 1   21.5   25.3   25.9   21.6   16.9   12.0    7.5    4.9    5.4    9.1   13.9
## 2   30.0   28.2   27.3   27.6   28.0   26.9   25.9   23.3   23.9   27.5   27.7
## 3   16.3   15.1   17.2   17.6   19.7   20.7   22.1   22.6   23.2   21.8   20.2
## 4   15.9   15.7   14.7   15.4   13.8   12.2   10.9   10.3   12.1   14.2   14.7
## 5   21.7   23.0   23.5   20.2   13.6    6.2    5.0    0.1    4.0    4.8    9.6
## 6   33.1   31.0   28.9   29.3   25.9   20.2   15.4   13.8   16.6   22.7   27.4
##   tmp_77 tmp_78 tmp_79 tmp_80 tmp_81 tmp_82 tmp_83 tmp_84 tmp_85 tmp_86 tmp_87
## 1   18.4   22.0   25.3   26.5   22.3   16.8   12.9    7.9    4.4    6.1    9.0
```

```
## 2   29.9   28.9   28.2   27.7   27.9   28.9   29.0   25.5   24.4   24.7   26.9
## 3   17.9   16.5   17.3   18.0   19.5   20.5   21.8   23.0   23.8   23.5   22.7
## 4   15.6   17.0   16.0   15.7   15.1   13.6   12.6   10.7   11.0   12.1   14.2
## 5   14.0   20.8   23.9   22.6   19.2   13.2    6.5    3.2    4.3    4.8    6.6
## 6   30.2   31.6   31.1   29.1   29.1   23.8   19.2   13.5   13.6   17.2   21.0
##   tmp_88 tmp_89 tmp_90 tmp_91 tmp_92 tmp_93 tmp_94 tmp_95 tmp_96 tmp_97 tmp_98
## 1   15.5   19.7   21.4   25.5   26.6   23.7   18.9   12.1    7.9    5.0    5.1
## 2   29.8   31.7   30.5   28.4   28.4   28.0   29.3   27.7   25.9   23.6   25.9
## 3   21.0   17.9   16.2   16.9   18.8   18.7   19.4   20.8   21.7   23.0   23.2
## 4   16.3   16.5   17.6   15.9   15.9   15.1   13.7   13.3   11.3   10.8   12.7
## 5   11.4   17.6   20.4   24.0   24.4   21.0   13.9    8.3    5.5    1.0    2.6
## 6   29.0   33.2   33.9   31.1   30.1   29.3   26.1   20.7   15.6   13.9   18.4
##   tmp_99 tmp_100 tmp_101 tmp_102 tmp_103 tmp_104 tmp_105 tmp_106 tmp_107
## 1    9.3    13.8    18.7    22.0    25.6    28.0    24.8    18.1    12.7
## 2   27.6    29.3    29.9    28.7    27.8    27.7    27.7    28.6    27.7
## 3   23.3    20.0    17.1    16.2    16.8    17.1    18.9    18.5    19.6
## 4   14.0    16.4    16.6    16.4    14.8    15.2    14.2    13.1    11.6
## 5    4.9    10.6    16.2    22.1    26.2    23.8    20.7    13.0    10.2
## 6   23.6    30.1    32.7    32.7    30.9    29.9    30.0    26.9    22.0
##   tmp_108 tmp_109 tmp_110 tmp_111 tmp_112 tmp_113 tmp_114 tmp_115 tmp_116
## 1     7.2     6.4     4.1     7.7    13.3    18.7    21.7    26.5    27.8
## 2    25.7    24.8    24.0    26.7    29.4    29.4    29.3    27.3    27.9
## 3    21.7    22.8    22.7    21.9    20.8    17.7    17.6    14.5    17.0
## 4     9.5    10.8    12.9    14.5    15.7    16.1    15.6    15.7    14.9
## 5     4.4    -0.5     2.4     7.9    10.0    16.4    21.2    22.3    22.7
## 6    16.3    15.0    15.3    21.8    30.6    33.6    32.8    30.3    30.1
##   tmp_117 tmp_118 tmp_119 tmp_120
## 1    24.0    17.7    12.4     7.1
## 2    28.2    29.6    28.4    26.0
## 3    17.9    21.5    20.8    21.9
## 4    15.0    13.9    12.7    10.2
## 5    18.9    13.8     7.3    -0.6
## 6    29.5    27.8    21.8    16.7
```

```
head(city_sim)
```

```
##   ID TREFHT_852  TREFHT_853 TREFHT_854 TREFHT_855 TREFHT_856 TREFHT_857
## 1  1  9.5677124  6.84594116   6.299646   8.345361   13.05013   17.59014
## 2  2 23.9092346 23.19564209  24.261133  28.055414   28.57235   27.92861
## 3  3 20.8986145 22.12758789  21.957758  20.540338   19.32577   17.88888
## 4  4  7.3596130  5.04674683   9.382104  11.671075   13.85964   13.75826
## 5  5 -0.9951233 -0.05698242  -1.134283   7.801965   10.77346   17.06130
## 6  6 14.1072327 11.25136719  18.630670  26.367639   32.86053   36.78161
##   TREFHT_858 TREFHT_859 TREFHT_860 TREFHT_861 TREFHT_862 TREFHT_863 TREFHT_864
## 1   20.79290   23.23483   24.82443   24.59228   19.09170  13.921014   8.210260
## 2   28.84109   26.58053   26.19525   26.36111   27.26751  24.708459  23.377954
## 3   15.47054   16.23275   16.84008   17.12420   18.38259  18.487817  20.581354
## 4   14.56542   13.65997   14.96304   13.89843   11.47103  10.736078   7.503992
## 5   20.76489   22.81024   20.73528   17.98135   10.78536   2.780328  -5.836646
## 6   36.98922   31.19109   29.44003   30.72225   24.93130  19.212274  15.143243
##   TREFHT_865 TREFHT_866 TREFHT_867 TREFHT_868 TREFHT_869 TREFHT_870 TREFHT_871
## 1   4.185999   4.106073   6.775323   12.06173   16.23354   19.08807   22.01650
## 2  22.233270  24.069666  27.970300   28.50198   26.92846   27.59789   25.88790
## 3  21.453058  22.474268  21.263361   18.22924   17.11611   13.92693   14.76644
```

```
## 4   6.920068   7.026422  10.663263   12.36288   13.42312   14.14739   14.03076
## 5  -6.752631  -2.256842   7.288629   13.71157   16.56756   19.15328   21.66631
## 6  13.095850  17.081964  24.062555   31.23617   34.03665   33.12911   30.77596
##    TREFHT_872 TREFHT_873 TREFHT_874 TREFHT_875 TREFHT_876 TREFHT_877 TREFHT_878
## 1   23.63503   23.10406   18.48882  13.109735   9.089197   6.179712   5.667322
## 2   25.59719   27.05389   26.95245  25.585413  23.697412  23.985651  25.640558
## 3   14.37408   17.46145   18.87707  19.860529  19.720911  20.235834  22.495416
## 4   13.66750   13.17135   13.57827  11.030786   9.975031   8.000909   7.872369
## 5   22.29299   16.41241   11.87979   5.448083   4.415948  -2.682196  -2.575507
## 6   30.53884   30.15231   25.58828  19.702753  13.777277  14.411890  18.130701
##    TREFHT_879 TREFHT_880 TREFHT_881 TREFHT_882 TREFHT_883 TREFHT_884 TREFHT_885
## 1   8.167627   11.85494   17.04699   21.46539   23.51998   24.84762   23.58154
## 2  26.418085   27.80767   26.86437   27.43307   26.28890   26.00451   26.57867
## 3  20.959924   20.04180   17.40630   15.81118   14.95028   15.44439   18.61682
## 4  11.270624   14.87182   14.81503   13.84756   13.88259   14.47561   14.70162
## 5   5.279626   12.39083   15.47726   21.29131   21.98644   20.69601   17.55566
## 6  22.598199   32.36163   34.22170   36.52618   30.99120   30.74871   29.42663
##    TREFHT_886 TREFHT_887 TREFHT_888 TREFHT_889 TREFHT_890 TREFHT_891 TREFHT_892
## 1   18.86813  13.932458  9.0870300   6.129724   4.426782   7.466943   11.84261
## 2   26.80981  25.425928 23.9366699  21.896844  23.702600  27.321100   26.46780
## 3   20.61273  22.543115 21.7111755  21.435632  23.204797  21.132776   19.76711
## 4   12.31790  10.917169  9.8026062   6.788110   9.164789  11.147974   13.97314
## 5   13.28069   4.376428 -0.6789001  -4.476416  -5.330145   2.902765   14.29632
## 6   25.71154  18.360956 12.8065430  11.868707  15.246729  26.072748   31.15429
##    TREFHT_893 TREFHT_894 TREFHT_895 TREFHT_896 TREFHT_897 TREFHT_898 TREFHT_899
## 1   16.33877   17.69167   22.01174   23.15383   22.83584   18.54958  14.537592
## 2   26.74117   27.84463   28.00558   26.49618   27.09155   27.81832  26.465204
## 3   16.51547   15.95175   15.77758   14.66830   20.08050   20.25579  21.408899
## 4   14.40533   13.94241   14.40365   14.39242   15.05874   12.47122  11.390009
## 5   16.80245   22.07504   23.28707   21.79510   17.01348   13.41122   5.879877
## 6   34.06594   33.46905   34.45248   32.17257   30.17538   25.99331  20.066461
##    TREFHT_900 TREFHT_901 TREFHT_902 TREFHT_903 TREFHT_904 TREFHT_905 TREFHT_906
## 1   7.324609  5.8558289  4.2887207   7.614496   13.39849   17.71584   21.12353
## 2  24.424493 24.8633972 25.4298645  25.601129   27.28417   27.77484   27.57021
## 3  20.869989 22.4812866 23.1112305  21.604089   17.91808   17.00973   14.91842
## 4   7.743005  7.0039917  6.3134705  11.025446   13.37615   15.17452   14.02313
## 5   1.502466 -0.7082275 -0.7616638   6.151453   12.12390   17.04501   21.28304
## 6  15.626154 14.3627869 18.5158936  21.863000   29.16915   34.08123   37.69964
##    TREFHT_907 TREFHT_908 TREFHT_909 TREFHT_910 TREFHT_911 TREFHT_912 TREFHT_913
## 1   23.69311   24.89688   24.57601   19.42114  13.251672   9.876581   6.225427
## 2   26.74865   26.47271   26.49258   27.50173  25.956232  23.742639  23.011011
## 3   14.27816   15.17324   17.07626   20.59704  20.036249  21.040125  22.062738
## 4   13.77557   14.62194   14.54748   13.47451  10.400598   7.275903   6.091180
## 5   21.80398   21.59374   17.88534   11.59850   5.203119  -0.113501  -1.474921
## 6   34.59347   31.41326   29.95046   25.13168  19.717859  16.054987  14.539453
##    TREFHT_914 TREFHT_915 TREFHT_916 TREFHT_917 TREFHT_918 TREFHT_919 TREFHT_920
## 1  4.5717712   8.391656  13.377283   17.37811   20.90762   22.99923   24.49963
## 2 25.5119568  27.301538  27.719293   27.60836   27.91442   26.76452   25.97305
## 3 22.9002014  21.094934  18.662256   16.42977   13.38445   14.18603   17.14053
## 4  8.5910278  12.801721  15.939600   14.38003   14.83337   14.46694   15.11340
## 5 -0.2336182   6.005212   9.439966   15.50958   21.24532   22.14343   20.76354
## 6 18.4531189  24.676935  32.365839   37.54901   35.63259   32.42309   32.19860
##    TREFHT_921 TREFHT_922 TREFHT_923 TREFHT_924 TREFHT_925 TREFHT_926 TREFHT_927
## 1   23.60729   18.78991  15.588251 10.5324036   5.592767   7.273004   9.414270
```

```
## 2   26.85052   27.85754  26.904382 25.0191895  23.137811  24.974603  28.007166
## 3   18.94229   20.17196  21.040643 21.5849548  21.742731  21.627374  20.972009
## 4   14.30264   13.76959  11.750452  9.0401550   6.755762   7.949213  11.257013
## 5   17.59020   13.53619   5.741785  0.5412842   1.991022   2.869714   5.920221
## 6   31.48538   28.19714  22.060602 16.7240234  12.929041  16.163660  23.220789
##   TREFHT_928 TREFHT_929 TREFHT_930 TREFHT_931 TREFHT_932 TREFHT_933 TREFHT_934
## 1   13.53344   16.61459   19.43261   23.19702   24.36346   23.86489   19.23968
## 2   28.52300   27.28195   27.66876   27.30438   26.95040   26.78369   27.67028
## 3   20.17797   16.50271   15.57580   12.70245   16.41125   17.75027   20.78701
## 4   14.33495   15.20449   14.77386   14.24673   14.05657   14.26019   12.33981
## 5   11.87081   17.53619   21.66433   23.60125   21.80734   18.38702   14.46661
## 6   29.51254   35.99340   39.78118   34.21511   29.72970   30.19589   27.05691
##   TREFHT_935 TREFHT_936 TREFHT_937 TREFHT_938 TREFHT_939 TREFHT_940 TREFHT_941
## 1  14.441827  10.695459   8.047388   6.338953   9.342615   13.75576   18.41927
## 2  26.662317  25.796045  26.616144  25.129419  26.750482   29.23367   27.21270
## 3  21.734979  20.798395  22.261255  21.417627  21.060175   18.17611   16.53359
## 4  11.666864   9.537073   8.476892   8.903070  12.097925   12.87509   13.87344
## 5   7.187402   2.051050  -3.894293  -4.864050   8.008905   11.77938   15.70693
## 6  21.255121  16.510614  16.161066  18.493341  24.754144   32.00973   35.51272
##   TREFHT_942 TREFHT_943 TREFHT_944 TREFHT_945 TREFHT_946 TREFHT_947 TREFHT_948
## 1   21.51678   24.40093   25.35983   24.05590   19.64117  13.161371  10.616602
## 2   26.71060   26.93676   26.72424   26.30917   26.97280  25.367426  25.442468
## 3   14.62289   14.11315   14.15344   18.57699   19.22338  20.906244  21.696405
## 4   13.73647   13.79275   14.28631   13.55297   12.96276  10.597040   8.771021
## 5   20.66345   23.50338   21.62768   18.01928   12.88464   3.983606   1.508203
## 6   32.73705   31.02511   31.58444   29.39425   27.41918  20.922876  15.697260
##   TREFHT_949 TREFHT_950 TREFHT_951 TREFHT_952 TREFHT_953 TREFHT_954 TREFHT_955
## 1   6.393152   6.422723   10.14791   15.88247   17.81042   21.45431   24.18868
## 2  23.452631  24.521631   24.61288   27.76995   27.68160   27.74096   27.06478
## 3  21.602838  23.068414   19.75439   19.73153   17.38033   14.26913   14.91192
## 4   9.238580   9.599603   12.94360   14.03738   14.47814   14.13152   14.22500
## 5   1.659479   7.074274   10.46337   11.84838   16.46682   21.83767   23.91708
## 6  11.795953  14.735864   22.23092   30.41256   37.53756   36.71319   30.36447
##   TREFHT_956 TREFHT_957 TREFHT_958 TREFHT_959 TREFHT_960 TREFHT_961 TREFHT_962
## 1   25.26971   23.79455   19.66470  15.401025  9.8859497   6.772363   5.159174
## 2   26.28756   26.82073   28.54284  26.340570 23.1759583  22.625513  24.851953
## 3   16.97805   18.94814   19.00771  19.905847 21.5077148  22.023920  22.295068
## 4   14.60232   13.95840   11.59295  10.175409  8.7353760   7.220728   8.474390
## 5   22.09295   19.16213   12.68795   4.560938 -0.3471436  -6.766547  -6.151160
## 6   29.73681   30.05502   26.34417  19.228693 15.0643250  14.029413  17.683252
##   TREFHT_963 TREFHT_964 TREFHT_965 TREFHT_966 TREFHT_967 TREFHT_968 TREFHT_969
## 1   9.453973   12.59490   16.32321   20.98141   23.96447   24.96322   23.97277
## 2  27.150507   26.70916   26.92397   27.30361   26.08926   25.92108   26.45535
## 3  22.251184   19.38964   17.46438   16.37390   16.31112   17.73837   21.05279
## 4  11.078058   12.85427   15.19003   14.15716   14.10525   15.15042   14.85061
## 5   1.366541   12.78796   18.44195   20.16223   21.83709   20.00164   18.86895
## 6  26.273248   29.02816   30.90078   32.16372   32.57754   31.26306   28.81869
##   TREFHT_970 TREFHT_971
## 1   19.12304  15.254297
## 2   27.02572  25.804742
## 3   22.08877  21.224359
## 4   13.33730  12.789880
## 5   12.19338   4.873712
## 6   27.47091  19.682397
```

```r
#this makes it so that the first column is only removed ONCE if it's named "ID"
if ("ID" %in% colnames(city_obs)) {
  city_obs <- city_obs[, -1]
}
if ("ID" %in% colnames(city_sim)) {
  city_sim <- city_sim[, -1]
}

#transpose to fix row-column swap
city_obs <- t(city_obs)
city_sim <- t(city_sim)

#data frame
city_obs <- as.data.frame(city_obs)
city_sim <- as.data.frame(city_sim)
print(dim(city_obs))
```

```
## [1] 120  10
```

```r
print(dim(city_sim))
```

```
## [1] 120  10
```

```r
print(colnames(city_obs))
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10"
```

```r
print(colnames(city_sim))
```

```
##  [1] "V1"  "V2"  "V3"  "V4"  "V5"  "V6"  "V7"  "V8"  "V9"  "V10"
```

We have to do a bit of data-wrangling to compare modeled and observed temperature data for each city.

5a. Add a column to both data-frames with the names of the cities using the NAME column from the city data frame

```r
#renaming the columns using city names
colnames(city_obs) <- as.character(top10_cities$NAME)
colnames(city_sim) <- as.character(top10_cities$NAME)
print(colnames(city_obs))
```

```
##  [1] "Tokyo"       "Mumbai"      "São Paulo"   "Mexico City" "New York"
##  [6] "Delhi"       "Shanghai"    "Kolkata"     "Dhaka"       "Karachi"
```

```r
print(colnames(city_sim))
```

```
##  [1] "Tokyo"       "Mumbai"      "São Paulo"   "Mexico City" "New York"
##  [6] "Delhi"       "Shanghai"    "Kolkata"     "Dhaka"       "Karachi"
```

```r
#adding the "time" column
city_obs$time <- time(obs_temp_raster)
city_sim$time <- time(nc_raster_celsius)

#verifying the new dimensions
print(dim(city_obs))
```

```
## [1] 120  11
```

```r
print(dim(city_sim))
```

```
## [1] 120  11
```

```r
print(colnames(city_obs))
```

```
##  [1] "Tokyo"       "Mumbai"      "São Paulo"   "Mexico City" "New York"
##  [6] "Delhi"       "Shanghai"    "Kolkata"     "Dhaka"       "Karachi"
## [11] "time"
```

```r
print(colnames(city_sim))
```

```
##  [1] "Tokyo"       "Mumbai"      "São Paulo"   "Mexico City" "New York"
##  [6] "Delhi"       "Shanghai"    "Kolkata"     "Dhaka"       "Karachi"
## [11] "time"
```

```r
head(city_obs)
```

```
##           Tokyo Mumbai São Paulo Mexico City New York Delhi Shanghai Kolkata
## tmp_1  4.900000   23.1      21.7        11.1      0.8  13.0      4.4    19.4
## tmp_2  4.900000   24.5      22.4        12.2      3.8  16.9      5.9    24.4
## tmp_3  8.900001   26.8      21.9        15.4      6.6  22.3      7.9    28.7
## tmp_4 14.500000   29.0      20.6        16.4     12.1  27.2     13.6    30.4
## tmp_5 18.100000   30.6      18.5        16.7     19.1  32.3     18.7    31.0
## tmp_6 23.000000   29.4      17.7        16.5     22.3  33.2     23.7    29.9
##       Dhaka Karachi       time
## tmp_1  17.8    17.4 1991-01-16
## tmp_2  22.4    19.1 1991-02-15
## tmp_3  26.9    23.8 1991-03-16
## tmp_4  28.1    28.9 1991-04-16
## tmp_5  28.1    31.6 1991-05-16
## tmp_6  28.2    33.2 1991-06-16
```

```r
head(city_sim)
```

```
##                Tokyo   Mumbai São Paulo Mexico City    New York    Delhi
## TREFHT_852  9.567712 23.90923  20.89861    7.359613 -0.99512329 14.10723
## TREFHT_853  6.845941 23.19564  22.12759    5.046747 -0.05698242 11.25137
## TREFHT_854  6.299646 24.26113  21.95776    9.382104 -1.13428345 18.63067
## TREFHT_855  8.345361 28.05541  20.54034   11.671075  7.80196533 26.36764
```

```
## TREFHT_856 13.050134 28.57235  19.32577   13.859644 10.77346191 32.86053
## TREFHT_857 17.590143 27.92861  17.88888   13.758264 17.06130371 36.78161
##             Shanghai  Kolkata    Dhaka  Karachi       time
## TREFHT_852  5.771021 21.23596 20.30871 17.68502 1991-01-01
## TREFHT_853  2.402155 18.91876 17.27410 14.15499 1991-02-01
## TREFHT_854  4.106226 22.09353 21.66500 20.54873 1991-03-01
## TREFHT_855  8.462213 25.34429 27.26278 25.56252 1991-04-01
## TREFHT_856 13.159204 26.89324 29.44360 28.34588 1991-05-01
## TREFHT_857 17.355920 27.97744 27.67037 29.79489 1991-06-01
```

5b. Use pivot_longer() from the tidyr package to turn both data-frames into tidy data-frames, with one row for each unique city-month combination

```
library(terra)
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked from 'package:terra':
##
##     extract
```

```
#extracts the observed and simulated temperature data
city_obs <- terra::extract(obs_temp_raster, top10_cities)
city_sim <- terra::extract(nc_raster_celsius, top10_cities)

#remove the ID column
if ("ID" %in% colnames(city_obs)) {
  city_obs <- city_obs[, -1]
}
if ("ID" %in% colnames(city_sim)) {
  city_sim <- city_sim[, -1]
}

city_obs <- t(city_obs)
city_sim <- t(city_sim)

#convert to data frame
city_obs <- as.data.frame(city_obs)
city_sim <- as.data.frame(city_sim)

#rename the columns with city names
colnames(city_obs) <- as.character(top10_cities$NAME)
colnames(city_sim) <- as.character(top10_cities$NAME)
#pivot the data
city_obs <- pivot_longer(
  city_obs,
  cols = everything(),
  names_to = "city",
  values_to = "observed"
)
```

```
city_sim <- pivot_longer(
  city_sim,
  cols = everything(),
  names_to = "city",
  values_to = "simulated"
)

#adds time column
city_obs$time <- rep(time(obs_temp_raster), times = 10)
city_sim$time <- rep(time(nc_raster_celsius), times = 10)
print(dim(city_obs))
```

```
## [1] 1200    3
```

```
print(dim(city_sim))
```

```
## [1] 1200    3
```

```
print(colnames(city_obs))
```

```
## [1] "city"     "observed" "time"
```

```
print(colnames(city_sim))
```

```
## [1] "city"      "simulated" "time"
```

5c. Notice that the modeled and observed rasters have used slightly different conventions for naming the months. You can see this in the "name" column of the two data frames you made in 5b. The model output uses the first of the month (e.g. 1991.02.01) whereas the observational data uses the middle of the month (e.g. 1991.01.16). This is a problem since we want to merge together the two data frames to compare observed and simulated data.

To merge the two data frames together, first we need to "chop off" the last two digits in the month ids in both data frames. One way to do this is to use the substr() function to return some subset of a character vector.

change the variable "time" from Date to "yearmon" (character)

```
library(zoo)
```

```
##
## Attaching package: 'zoo'

## The following object is masked from 'package:terra':
##
##     time<-

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
#convert time to yearmon format by extracting only YYYY-MM
city_obs$time <- as.yearmon(substr(as.character(city_obs$time), 1, 7))
city_sim$time <- as.yearmon(substr(as.character(city_sim$time), 1, 7))
print(head(city_obs$time))
```

```
## [1] "Jan 1991" "Feb 1991" "Mar 1991" "Apr 1991" "May 1991" "Jun 1991"
```

```r
print(head(city_sim$time))
```

```
## [1] "Jan 1991" "Feb 1991" "Mar 1991" "Apr 1991" "May 1991" "Jun 1991"
```

5d. Merge the observed and modeled city data into a single data-frame. In this case you could use `cbind`, but that it is safer to use `merge`

```r
#merges observed and simulated data using "city" and "time" as keys
city_data <- merge(city_obs, city_sim, by = c("city", "time"))
print(dim(city_data))
```

```
## [1] 12000     4
```

```r
print(colnames(city_data))
```

```
## [1] "city"      "time"      "observed"  "simulated"
```

```r
head(city_data)
```

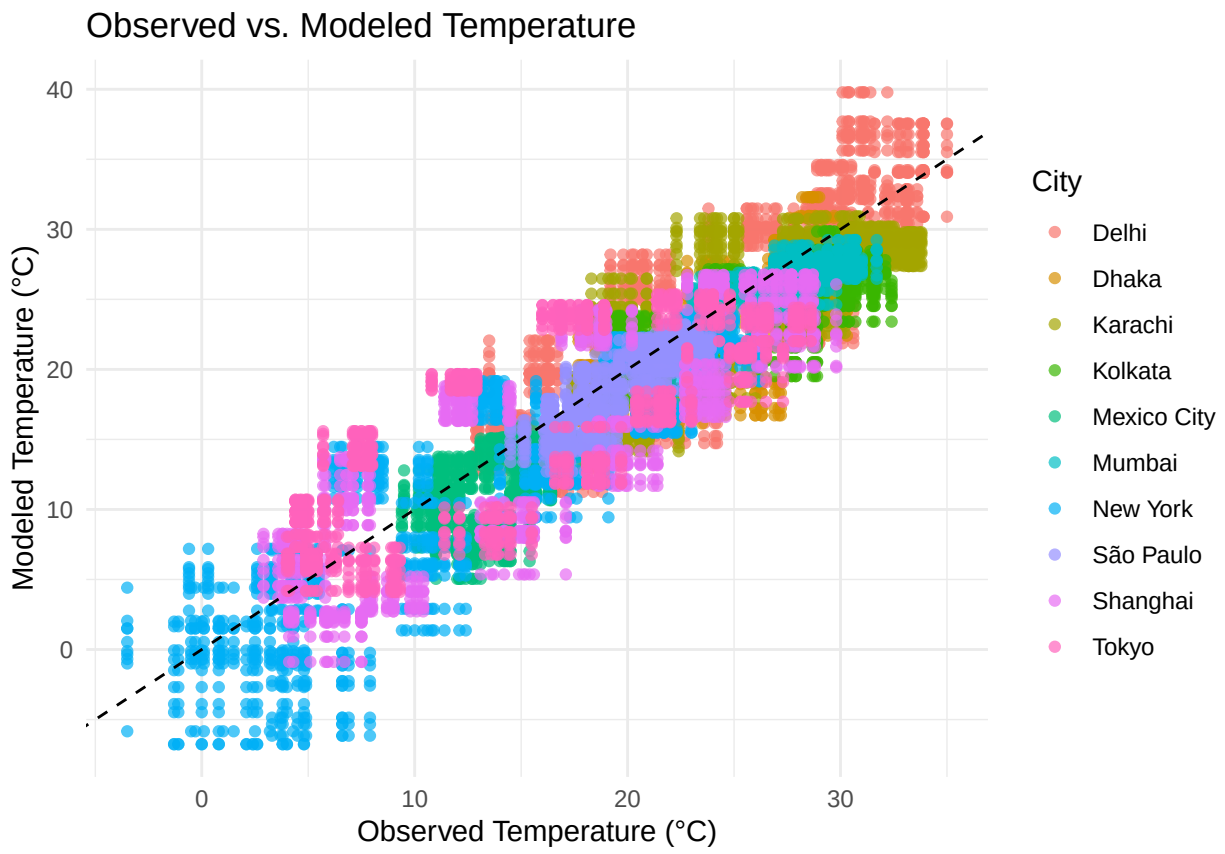
```
##    city     time observed simulated
## 1 Delhi Apr 1992     15.3  14.02941
## 2 Delhi Apr 1992     15.3  11.25137
## 3 Delhi Apr 1992     15.3  16.16107
## 4 Delhi Apr 1992     15.3  11.79595
## 5 Delhi Apr 1992     15.3  14.36279
## 6 Delhi Apr 1992     15.3  13.09585
```

5e. Make a plot showing observed vs modeled temperature for the 10 cities. Add a 1:1 line which showing the exact match between observed and modeled data. You can use base plot or ggplot.

```r
library(ggplot2)

#scatter plot
ggplot(city_data, aes(x = observed, y = simulated, color = city)) +
  geom_point(alpha = 0.7) +
  geom_abline(slope = 1, intercept = 0, linetype = "dashed", color = "black") +
  labs(


    title = "Observed vs. Modeled Temperature",
    x = "Observed Temperature (°C)",
    y = "Modeled Temperature (°C)",
    color = "City"
  ) +
  theme_minimal()
```



#Part 2

In the second part of the lab, we will use projections of future temperature change (until 2080) and a map of the distribution of population in 2020 to get global, population-weighted projected warming.

6a. Read in the netCDF file with projected climate model temperature (in the "Climate Model Data_Future" directory) as a SpatRaster. Use the rotate() function again as you did in 1b to transform the coordinates to -180 to 180 and the units to C. Use `subds="TREFHT"`. This has gridded projections of monthly global temperature between 2006 and 2020 under a high-emissions scenario (referred to as RCP8.5).

```
library(terra)

#future temperature raster loaded
future_temp_raster <- rast("/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Climate Mo
                           subds = "TREFHT")

#rotates raster to fix longitude
future_temp_raster <- rotate(future_temp_raster)
print(future_temp_raster)
```

```
## class       : SpatRaster
## dimensions  : 192, 288, 900  (nrow, ncol, nlyr)
## resolution  : 1.25, 0.9424084  (x, y)
## extent      : -180.625, 179.375, -90.4712, 90.4712  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## varname     : TREFHT (Reference height temperature)
```

```
## names       : TREFHT_1, TREFHT_2, TREFHT_3, TREFHT_4, TREFHT_5, TREFHT_6, ...
## min values  : 227.8591, 226.3425, 216.4508, 209.2771, 204.3515, 210.9332, ...
## max values  : 306.0093, 306.0674, 307.2524, 307.4044, 310.5476, 314.0392, ...
## unit        :        K,        K,        K,        K,        K,        K, ...
## time (days) : 2006-02-01 to 2081-01-01
```

6b. Compute the projected *annual* trend in global climate. Use `tapp` for this temporal aggregation.

```
#computes annual mean temperature using tapp()
annual_temp_raster <- tapp(future_temp_raster, index = "years", fun = mean)
print(annual_temp_raster)
```

```
## class       : SpatRaster
## dimensions  : 192, 288, 76  (nrow, ncol, nlyr)
## resolution  : 1.25, 0.9424084  (x, y)
## extent      : -180.625, 179.375, -90.4712, 90.4712  (xmin, xmax, ymin, ymax)
## coord. ref. : +proj=longlat +datum=WGS84 +no_defs
## source(s)   : memory
## names       :   y_2006,   y_2007,   y_2008,   y_2009,   y_2010,   y_2011, ...
## min values  : 215.2909, 217.5892, 217.3202, 217.0331, 218.8872, 218.0642, ...
## max values  : 303.1823, 303.1991, 302.6534, 303.0816, 304.2696, 303.0414, ...
## time (years): 2006 to 2081
```

7a. Read in the netCDF data on population in the "Population" directory as a SpatRaster. (There is only one variable in this netCDF, so you do not need to specify the variable name this time). This is gridded population count at 15 arc minute resolution.

```
library(terra)
pop_file <- "/Users/hectorsmacbookpro/Documents/School/ESP 106/Labs/LAB7/Data/Population/gpw_v4_populati

#reads the netCDF file as a SpatRaster
pop_raster <- rast(pop_file)
print(pop_raster)
```

```
## class       : SpatRaster
## dimensions  : 720, 1440, 20  (nrow, ncol, nlyr)
## resolution  : 0.25, 0.25  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (CRS84) (OGC:CRS84)
## source      : gpw_v4_population_count_adjusted_rev11_15_min.nc
## varname     : UN WPP-Adjusted Population Count, v4.11 (2000, 2005, 2010, 2015, 2020): 15 arc-minutes
## names       : UN WP~ter=1, UN WP~ter=2, UN WP~ter=3, UN WP~ter=4, UN WP~ter=5, UN WP~ter=6, ...
## unit        :     Persons,     Persons,     Persons,     Persons,     Persons,     Persons, ...
```

7b. We want only the 5th layer in this SpatRaster, which corresponds to population count in 2020. (Note - I know this from some associated files that came with the netCDF file. Take a look at the csv file in the directory to see this documentation). Pull out just the population in 2020.

```
#extracts the 5th layer
pop_2020 <- pop_raster[[5]]
print(pop_2020)
```

```
## class       : SpatRaster
## dimensions  : 720, 1440, 1  (nrow, ncol, nlyr)
## resolution  : 0.25, 0.25  (x, y)
## extent      : -180, 180, -90, 90  (xmin, xmax, ymin, ymax)
## coord. ref. : lon/lat WGS 84 (CRS84) (OGC:CRS84)
## source      : gpw_v4_population_count_adjusted_rev11_15_min.nc
## varname     : UN WPP-Adjusted Population Count, v4.11 (2000, 2005, 2010, 2015, 2020): 15 arc-minutes
## name        : UN WPP-Adjusted Population Cou~2020): 15 arc-minutes_raster=5
## unit        :                                                        Persons
```

8a. Now we want to eventually match the population grid to the projected temperature grid. But the problem is that the grid size of the climate model is much larger than the grid size of the population data. How many rows and columns does the climate model data have? And how many rows and columns does the population data have? Use code to show that.

```
print(dim(future_temp_raster))
```

```
## [1] 192 288 900
```

```
print(dim(pop_2020))
```

```
## [1]  720 1440    1
```

Answer From the output, the climate model data has 192 rows and 288 columns. The the population data has 720 rows and 1440 columns which means that the population data has a much finer resolution.

8b. To fix this problem we can aggregate the population raster up to the resolution of the climate model using the aggregate() function. The population data you have is the population count (i.e. number of people in each grid cell). What function should we use to aggregate to larger grid cells? What function would we use instead if we had population density data instead of population count?

Answer: For population count you can use the sum functions and for for population density you can use the mean function.

8c. Aggregate the population data to a higher level of resolution, as close as possible to the climate model data.

```
#aggregating he population data to match climate model resolution
pop_aggregated <- aggregate(pop_2020, fact = c(4,5), fun = "sum")
print(dim(pop_aggregated))
```

```
## [1] 180 288   1
```

```
print(dim(future_temp_raster))
```

```
## [1] 192 288 900
```

8d. If everything has gone according to plan, we would expect that summing up all the cells in the population SpatRaster should give us something close to the current population on the planet. Calculate that sum from your aggregated population data and compare to the total population today.

```r
#computes the total population from the aggregated raster
total_population <- global(pop_aggregated, fun = "sum", na.rm = TRUE)
print(total_population)
```

```
##
## UN WPP-Adjusted Population Count, v4.11 (2000, 2005, 2010, 2015, 2020): 15 arc-minutes_raster=5 6322
```

```r
current_world_population <- 8000000000
difference <- abs(total_population - current_world_population)
print(paste("Difference from current world population:", difference))
```

```
## [1] "Difference from current world population: 1677620693.90227"
```

Answer:

9a. Now we will use the population data to do a weighted averaging of the projected temperature data, to get the monthly temperature experienced by the average person between 2006 and 2080.

One problem is that even after the aggregation, the grids of the population data still don't quite match. Use terra::resample() to resample the aggregated population data to the climate model grid.

```r
library(terra)

#resamples the aggregated population raster
pop_resampled <- resample(pop_aggregated, future_temp_raster, method = "bilinear")
print(dim(pop_resampled))
```

```
## [1] 192 288   1
```

```r
print(dim(future_temp_raster))
```

```
## [1] 192 288 900
```

9b. Now we can use the population SpatRaster to do a weighted average of the climate model data. Use the global() function to calculate both the global and and the population-weighted average temperature for each year.

```r
global_temp <- global(future_temp_raster, fun = "mean", na.rm = TRUE)[,1]

#computes the population-weighted mean temperature
pop_weighted_temp <- (global(future_temp_raster * pop_resampled, fun = "sum", na.rm = TRUE)[,1]) /
                     (global(pop_resampled, fun = "sum", na.rm = TRUE)[,1])

# had to make sure to convert to data frame for plotting
years <- 2006:2080
temp_data <- data.frame(Year = years,
                        Global_Temperature = global_temp,
                        Pop_Weighted_Temperature = pop_weighted_temp)

head(temp_data)
```

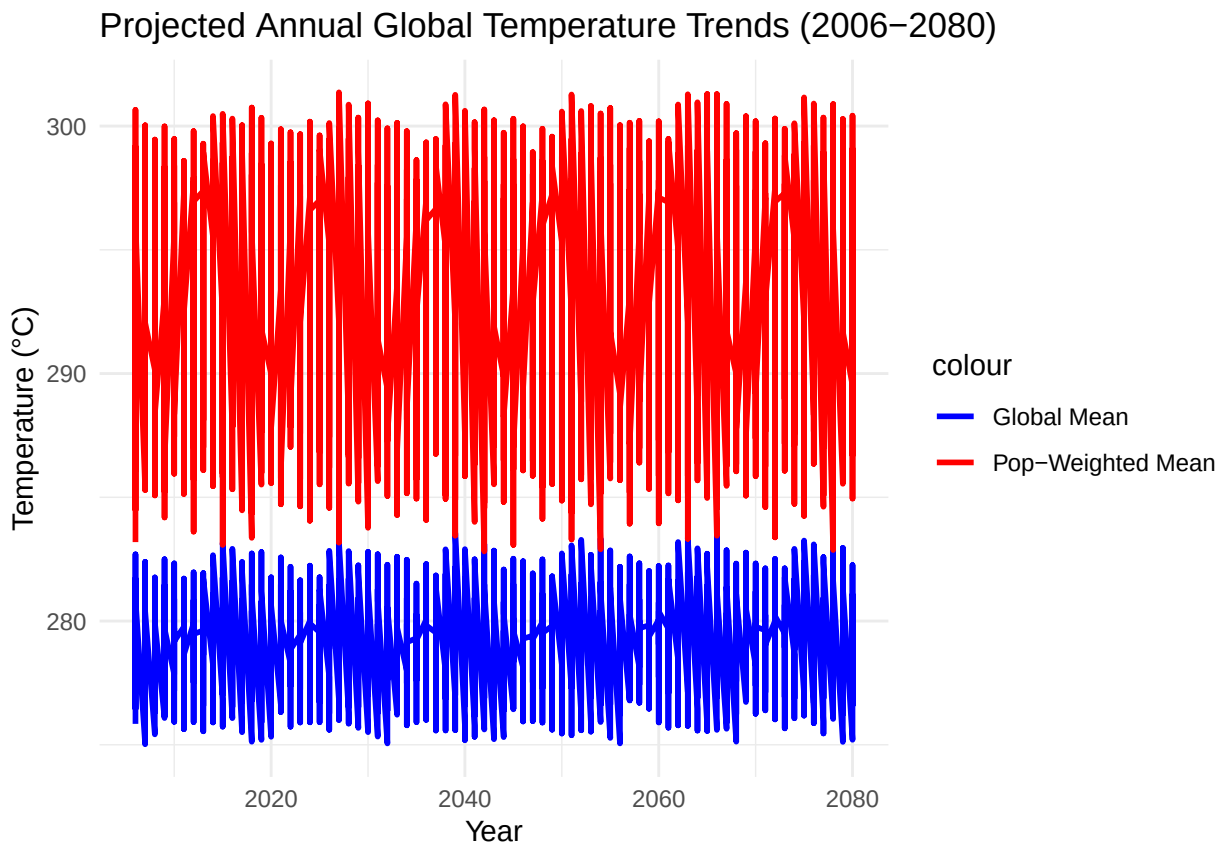
```
##   Year Global_Temperature Pop_Weighted_Temperature
## 1 2006           275.8452                 283.1898
## 2 2007           275.0193                 285.3673
## 3 2008           275.4210                 290.1538
## 4 2009           276.0805                 293.1456
## 5 2010           277.8445                 295.6844
## 6 2011           279.7656                 297.3013
```

Make a graph showing the projected annual trend in global climate. On the same graph show the temperature trend for the entire world, and weighted by population.

```r
library(ggplot2)

ggplot(temp_data, aes(x = Year)) +
  geom_line(aes(y = Global_Temperature, color = "Global Mean"), size = 1) +
  geom_line(aes(y = Pop_Weighted_Temperature, color = "Pop-Weighted Mean"), size = 1) +
  labs(title = "Projected Annual Global Temperature Trends (2006-2080)",
       x = "Year",
       y = "Temperature (°C)") +
  scale_color_manual(values = c("Global Mean" = "blue", "Pop-Weighted Mean" = "red")) +
  theme_minimal()
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```


Projected Annual Global Temperature Trends (2006−2080)
Temperature (°C)
300
290
280
2020
2040
2060
2080
Year
colour
Global Mean
Pop−Weighted Mean