GRIDDED DATA FROM CRUTEM5

Victor Nyabuti Ong'era

2022-10-26

Lesson 1

We are now looking at gridded data .This is another way climatological data can be provided. As the name suggests the data is provided onto regular grids.

So far what we were doing was reanalysis. So we will need to reproduce climatilogical features of reanalysis fields in the past

This time our focus will not be globally but only over a selected grid

After that we are going to run the same analysis we run on the previous excertation

After that we are going to consinder any other data sets which can be downloaded from the Copernicus Climate Data Store

https ://cds climate copernicus eu/#!/ home

for now lets download our data from

https ://crudata uea ac uk/cru/data/temperature

The file is

CRUTEM5, netCDF

https ://cds climate copernicus eu/#!/ home

so lets define the path

```
knitr::opts_chunk$set(error = TRUE)# iignore this line its just for an error
#am getting when creating the document
```

Then install all packages we need

```
#install.packages("ncdf4","lubridate"  )
```

which are responsible for raeding ncdf files and for finding leap years respetively

Now lets load them

```
library(ncdf4)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

open the file by allocating it to a variable *nc*. we use the function **paste**

```
nc <- nc_open(paste(path1,"HadCRUT.5.0.1.0.anomalies.ensemble_mean.nc",sep=""))
```

inspect the variable

```
nc
```

```
## File C:/Users/VICTOR_NYABUTI/Climate/climate_change_global_adaptation/data/HadCRUT.5.0.1.0.anomalies
##
##      6 variables (excluding dimension variables):
##         double tas_mean[longitude,latitude,time]   (Chunking: [72,36,1])  (Compression: shuffle,level
##             _FillValue: -1e+30
##             long_name: blended air_temperature_anomaly over land with sea_water_temperature_anomaly
##             units: K
##             cell_methods: area: mean (interval: 5.0 degrees_north 5.0 degrees_east) time: mean (inter
##             coordinates: realization
##         double time_bnds[bnds,time]   (Contiguous storage)
##         double latitude_bnds[bnds,latitude]   (Contiguous storage)
##         double longitude_bnds[bnds,longitude]   (Contiguous storage)
##         8 byte int realization[]   (Contiguous storage)
##             bounds: realization_bnds
##             units: 1
##             standard_name: realization
##         8 byte int realization_bnds[bnds]   (Contiguous storage)
##
##      4 dimensions:
##         time  Size:2073
##             axis: T
##             bounds: time_bnds
##             units: days since 1850-01-01 00:00:00
##             standard_name: time
##             long_name: time
##             calendar: gregorian
##         latitude  Size:36
##             axis: Y
##             bounds: latitude_bnds
##             units: degrees_north
##             standard_name: latitude
##             long_name: latitude
##         longitude  Size:72
##             axis: X
##             bounds: longitude_bnds
##             units: degrees_east
##             standard_name: longitude
##             long_name: longitude
##         bnds  Size:2 (no dimvar)
##
##     9 global attributes:
```

```
##         comment: 2m air temperature over land blended with sea water temperature at a depth of 20cm
##         history: Data set built at: 2022-10-20T13:33:18+00:00
##         institution: Met Office Hadley Centre / Climatic Research Unit, University of East Anglia
##         licence: HadCRUT5 is licensed under the Open Government Licence v3.0 except where otherwise
##         reference: C. P. Morice, J. J. Kennedy, N. A. Rayner, J. P. Winn, E. Hogan, R. E. Killick, R
##         source: CRUTEM.5.0.1.0 HadSST.4.0.0.0
##         title: HadCRUT.5.0.1.0 blended land air temperature and sea-surface temperature anomaly data
##         version: HadCRUT.5.0.1.0
##         Conventions: CF-1.7
```

Lets extract this values so that we can work on the data. we need variables like the temperature means,longitude, latitude and even time . we use the function **ncvar_get()**

```
data <-  ncvar_get(nc,"tas_mean") #where we get a 3 dimensional matrix
```

lets extract time,longitude and latitude

```
longitude <- ncvar_get(nc,"longitude") #Vector
latitude <- ncvar_get(nc,"latitude") #Vector
time <- ncvar_get(nc,"time") #Vector
```

to get the number of months

```
nmonth <- length(time)
```

This data is daily. We have leap and non leap years meaning that they do not have the same number of days

```
#Julian days for a non-leap years. The day in the mid of the month is considered
day <- c(15.5,45,74.5,105,135.5,166,196.5,227.5,258,288.5,319,349.5)
#Julian days for a leap years. The day in the mid of the month is considered
day_leap <- c(15.5,45.5,75.5,106,136.5,167,197.5,228.5,259,289.5,320,350.5)
```

We use the lubridate package that is necessary to find if a year is leap. Bacially returns tru or false if the year is leap or non leap repsectively from the very first year ti the very last year

```
year_leap <- leap_year(1850:2022)
start_year <- 1850
end_year <- 2022
years <- c(1850:2022)
n_year <- end_year-start_year+1 # +1 ro include the very first year

summary(year_leap)
```

```
##    Mode   FALSE    TRUE
## logical     131      42
```

```
start_year
```

```
## [1] 1850
```

```
end_year
```

```
## [1] 2022
```

```
n_year
```

```
## [1] 173
```

```
for(i in 1:n_year){
  #if we are considering the first year
  if(i==1){
    day_sequence <- matrix(ncol=3,nrow=12)# empty matrix
    day_sequence[,1] <- years[i] #for column 1 put value of i from the years
    #variable
    day_sequence[,2] <- c(1:12) # for column 2 put values in a sequence 1 to 12 i.e 1,2,3...12
    if(year_leap[i]==FALSE) day_sequence[,3] <- day #If the year is not leap assighn it to variable day
    if(year_leap[i]==TRUE) day_sequence[,3] <- day_leap #If the year is leap assighn it to variable day_
    if(year_leap[i]==FALSE) count_day <- 365
    if(year_leap[i]==TRUE) count_day <- 366
  }
  #if we are not considering the first year
  if(i!=1){
    day_sequence_x <- matrix(ncol=3,nrow=12)
    day_sequence_x[,1] <- years[i]
    day_sequence_x[,2] <- c(1:12)
    if(year_leap[i]==FALSE) day_sequence_x[,3] <- day+count_day
    if(year_leap[i]==TRUE) day_sequence_x[,3] <- day_leap+count_day
    day_sequence <- rbind(day_sequence,day_sequence_x)
    if(year_leap[i]==FALSE) count_day <- count_day+365
    if(year_leap[i]==TRUE) count_day <- count_day+366
  }
}
```

Lesson 2

This data contains all the possible times from the beginning to the end my it is not mandatory that all these times are present in our data

we create a vector names time_ok. If zou just run the variable **time** you can see all possible times. From that varibale we define our new vector time_ok.

```
for(i in 1:length(time)){
  time_ok[i] <- which(time[i]==day_sequence[,3])
}
```

```
## Error in eval(expr, envir, enclos): object 'time_ok' not found
```

generally we are taking time_ok data from the 3rd column of the varibale *day_sequence*

```
day_sequence <- day_sequence[time_ok,]
```

```
## Error in eval(expr, envir, enclos): object 'time_ok' not found
```

Now the variable *day_sequence* has data just for the right period *time_ok*

we do not need the last year 2022 because its incomplete.so we delete it

```
day_sequence <- day_sequence[-which(day_sequence[,1]==2022),]
```

This matrix has data for all availlable grid points in time okay. But we need to work on a specific grid. In class we selected france. To know the right lat and lon, we need to go to google earth.

If you have a donloaded version of goooogle earth search an area in our case france.Turn on the grids by going to the view > check grids. i have highlighted in yellow the grids section and also a place where you can read in real time the coordinates of the area we are intrested in

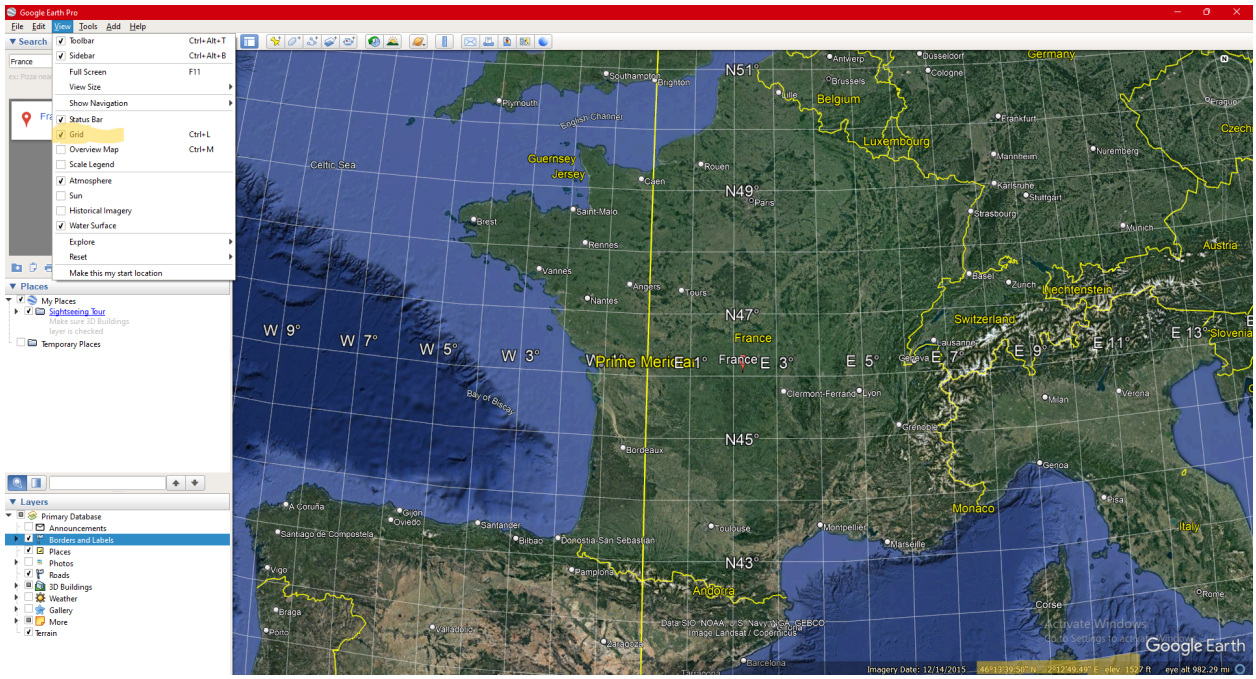
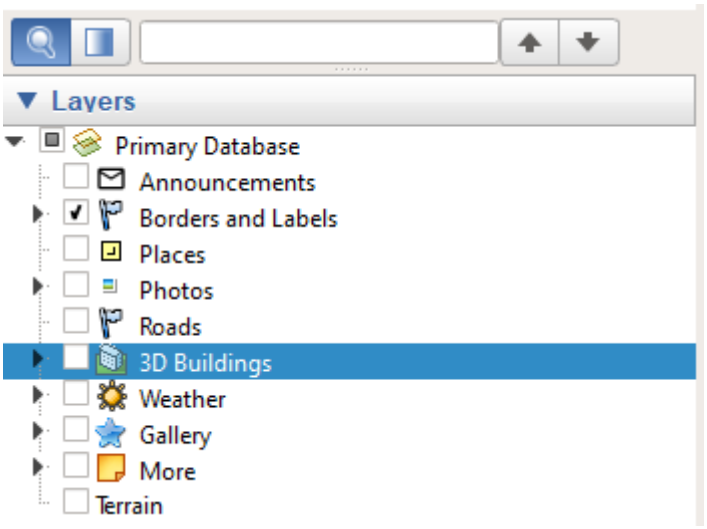


Figure 1: A caption

sometimes when zooming in you might end up not see the grids clearlz so it can be beneficial to turn off other layers at the lower left part of the screen



so lets inspect france. It comes lovely the area covered by this coordinates

```
#The matrix with the data includes the monthly values for all the available grid-points
#I cut a part and I calculate the monthly mean for the selected area
area_lon_min <- 5
area_lon_max <- 19
area_lat_min <- 35
area_lat_max <- 47
```

from our data lets search this area's data

```
lat_selected <- which(latitude>=area_lat_min & latitude<=area_lat_max)
lon_selected <- which(longitude>=area_lon_min & longitude<=area_lon_max)

#Number of point included in the selected area
npoint_selected <- length(lat_selected)*length(lon_selected)
npoint_selected
```

```
## [1] 6
```

```
#Coordinates of the selected from-points
grid_selected <- matrix(ncol=2,nrow=npoint_selected)#empty matrix
grid_selected
```

```
##      [,1] [,2]
## [1,]   NA   NA
## [2,]   NA   NA
## [3,]   NA   NA
## [4,]   NA   NA
## [5,]   NA   NA
## [6,]   NA   NA
```

```
colnames(grid_selected) <- c("Latitude","Longitude") #renaming the columns
grid_selected
```

```
##      Latitude Longitude
## [1,]       NA        NA
## [2,]       NA        NA
## [3,]       NA        NA
## [4,]       NA        NA
## [5,]       NA        NA
## [6,]       NA        NA
```

```
#filling the matrix
ii <- 0
for(i in 1:length(lat_selected)){
  for(j in 1:length(lon_selected)){
    ii <- ii+1
    grid_selected[ii,"Latitude"] <- latitude[lat_selected[i]]
    grid_selected[ii,"Longitude"] <- longitude[lon_selected[j]]
  }
}


#check it out
grid_selected
```

```
##      Latitude Longitude
## [1,]     37.5       7.5
## [2,]     37.5      12.5
## [3,]     37.5      17.5
## [4,]     42.5       7.5
## [5,]     42.5      12.5
## [6,]     42.5      17.5
```

```
#exported the matrix as a table
write.table(grid_selected,paste(path3,"Grid_points_",area_lon_min,"_",
                                area_lon_max,"_",area_lat_min,"_",
                                area_lat_max,".csv",sep=""),sep=";",
            col.names=TRUE,row.names=FALSE,quote=FALSE)


#For each month I extract the temperature anomaly data of the selected area
# and I calculate the area mean
#
temp_selected <- vector()#empty vector then we fill it
for(i in 1:length(day_sequence[,1])){
  time_ok <- which(time==day_sequence[i,3])
  data_selected <- data[lon_selected,lat_selected,time_ok]
  temp_selected[i] <- mean(data_selected,na.rm=TRUE)
}

#we can peak at it
head(temp_selected)
```

```
## [1] -3.5565952  0.5011614 -1.7601439 -0.1476181 -0.6301746  0.7099497
```

```
#Now we bind/ add other data into the vector. we add day sequence
temp_selected <- cbind(day_sequence,temp_selected)

#we can peak at it
head(temp_selected)
```

```
##                   temp_selected
## [1,] 1850 1  15.5    -3.5565952
## [2,] 1850 2  45.0     0.5011614
## [3,] 1850 3  74.5    -1.7601439
## [4,] 1850 4 105.0    -0.1476181
## [5,] 1850 5 135.5    -0.6301746
## [6,] 1850 6 166.0     0.7099497
```

```
#rename the columns

colnames(temp_selected) <- c("Year","Month","Julian_day","Mean")

#we can peak at it
head(temp_selected)
```

```
##      Year Month Julian_day       Mean
## [1,] 1850     1       15.5 -3.5565952
## [2,] 1850     2       45.0  0.5011614
## [3,] 1850     3       74.5 -1.7601439
## [4,] 1850     4      105.0 -0.1476181
## [5,] 1850     5      135.5 -0.6301746
## [6,] 1850     6      166.0  0.7099497
```

```
#we expport it as a table
write.table(temp_selected,paste(path3,"Temperature_anomaly_mean_",
                                area_lon_min,"_",area_lon_max,"_",area_lat_min,
                                "_",area_lat_max,".csv",
                                sep=""),,sep=";",col.names=TRUE,row.names=FALSE,
            quote=FALSE)
```

We now have data but instead of global the data is for a given area we can now run analysis on the data just as we ddi from our earlier lessons