WORKING WITH CRUTEM DATA, ECGS 2022.

Victor Nyabuti Ong'era

2022-10-17

This document concerns the **CRUTEM** tutorials we had in class. Please do check back on your personal notes and let me know if I missed or misunderstood something.Use at your own risk because this are just my own notes. There will by typos but its what its now.

I will be giving a little description on what we were encountering in class and how I interpreted it. If its weird or doesnt make sense just got to my git where I have created a repository of the class. The code will however have some notes since I took them in class.

Download the data from here and select one of the four data sets.I am using ***CRUTEM5alt*** *Land air temperature anomalies on a 5° by 5° grid, not infilled but with better representation of high-latitude stations (Osborn et al., 2021)*

working in R.

Import the data to R.

1. When clicking on the data set it will open something like this which is in .txt form. Ctrl+A > Ctrl+C
2. Open excel sheet paste the data there by Ctrl+V. You might not see anything at this point.
3. Go to data > text to column > press next in the dialog that will appear till all options are done then click finish.
4. Save this file in a folder you like. To save click FILE > Save As > select a folder > rename as GL and in the dialog below the name selct the format as CSV(comma delimited)(.csv)
5. Define your working path in R.I suggest having 2 folders , for data and for results.To define the path go to the GL file you saved right click > properties > at the location copy the path.
6. Go to R And paste it.

```
#To comment or take notes in R environment use #

#the path looks like something like this C:\Users\VICTOR_NYABUTI\Climate\data

#change the slashes to be forward facing

#To run your line just Ctrl+Enter

path1 = "C:/Users/VICTOR_NYABUTI/Climate/data/"
```

1.

Variables are like tags or labels.Used to store information. Generally give a name that is related to what that variable will store. They can be changed. We will experince that expecially when we will be working with data frames.The names can not start with a number, but can have a number in between. can not start with *-/,' "% but can have a underscore or hyphen in between. I think we get the point.

R is case sensitive so if you save your variable as *name* you can not call that variable using *Name*

To call a variable is like printing it to see what it contains.

Please note that the teacher is using -> to name variables. I am using = This is just a personal preference and I find it easier on my keyboard(am using Hungarian keyboard xd). Its also a standard way in other languages like python but generally its good practice in R to us ->. Your call. back to R.

```
values = as.matrix(read.table(paste(path1,"GL.csv", sep=""), sep = ",",dec = "."))
```

We now need to do some analyses but this data frame has missing values.If we run fro exmaple Mean we gonna get an error. So its important to clean this data.You can look at your data by clicking at the values variable in the Global environment window. top right. Alternatively you can print them in your console by calling the values variable. For the sake of illustration am gonna dispplay just the first rows. you can use the head function for that. See column V14 that has NA values

```
head (values)
```

```
##        V1     V2     V3     V4     V5     V6     V7     V8     V9    V10    V11
## [1,] 1850 -0.671 -0.296 -0.699 -0.508 -0.329 -0.237 -0.130 -0.207 -0.357 -0.442
## [2,] 1850 24.000 21.000 19.000 19.000 19.000 21.000 22.000 23.000 24.000 24.000
## [3,] 1851 -0.222 -0.328 -0.408 -0.465 -0.280 -0.247 -0.163 -0.170 -0.062 -0.053
## [4,] 1851 24.000 23.000 22.000 22.000 21.000 21.000 22.000 24.000 19.000 21.000
## [5,] 1852 -0.300 -0.445 -0.488 -0.568 -0.196 -0.056  0.068 -0.194 -0.093 -0.197
## [6,] 1852 24.000 23.000 23.000 23.000 24.000 24.000 24.000 24.000 22.000 23.000
##         V12    V13    V14
## [1,] -0.150 -0.239 -0.355
## [2,] 25.000 25.000     NA
## [3,] -0.025 -0.048 -0.206
## [4,] 19.000 21.000     NA
## [5,] -0.196  0.114 -0.213
## [6,] 23.000 26.000     NA
```

2.

Some times in the data , We have missing values maybe due to the failure of the thermometer or the thermometer malfucntioned and measure too high or too low values. From the paper which we presented in class we understand that there are inferencing methods on how that data is filled.

The values that have anormalies are not stores as NA but as -9.999. This indicates that there was a recording but anormally. Now those values are too low and can interfere with the overall evaluation therefore we deal with them by converting them to NA.

Before converting them we can see examples. Remeber you can check data table. For illustration am printing the last 6 values. Look at row 345

```
tail(values)
```

```
##          V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
## [341,] 2020  1.110  1.103  1.111  0.899  0.808  0.774  0.766  0.837  0.797
## [342,] 2020 85.000 85.000 84.000 83.000 83.000 83.000 83.000 84.000 83.000
## [343,] 2021  0.652  0.583  0.722  0.704  0.720  0.787  0.832  0.810  0.802
## [344,] 2021 82.000 82.000 82.000 82.000 82.000 82.000 82.000 83.000 83.000
## [345,] 2022  0.830  0.772  0.828  0.776  0.709  0.805  0.768 -9.999 -9.999
```

```
## [346,] 2022 81.000 81.000 82.000 81.000 81.000 80.000 81.000  0.000  0.000
##           V11    V12    V13   V14
## [341,]  0.711  0.840  0.629 0.865
## [342,] 83.000 82.000 82.000    NA
## [343,]  0.801  0.778  0.740 0.744
## [344,] 82.000 81.000 80.000    NA
## [345,] -9.999 -9.999 -9.999 0.784
## [346,]  0.000  0.000  0.000    NA
```

lets search the specific position of this values by using which function.

```
missing_values = which(values==-9.999)
```

L Print them

```
missing_values
```

```
## [1] 3113 3459 3805 4151 4497
```

As you can see this values are too huge. There is no way the temperature was that high maybe in the sun that would be possible xd.

Now lets see how you would select cells or data in a table. This is also true for matrixes. Its a bit different when we are looking at data frames.

Selecting data is called indexing. R counts numbers from 1 so the first value is 1 the second 2 etc.. indexing follows this format **variable[x,y]** where x is the row number, y is the column number.Lets for example print just the first value in our table

```
values[1,1]
```

```
##   V1
## 1850
```

This code literally translates to print from the values variable the value at row 1 column 1.If we wanted the last row which is number 346 in my case

```
values[346,]
```

```
##   V1   V2   V3   V4   V5   V6   V7   V8   V9  V10  V11  V12  V13  V14
## 2022   81   81   82   81   81   80   81    0    0    0    0    0   NA
```

which translates to print row 346 and all the columns with it.

```
values[missing_values] =  NA
```

Replace the anormalies with NA We can confirm if this worked by looking at our data

```
tail(values)
```

```
##          V1     V2     V3     V4     V5     V6     V7     V8     V9    V10
## [341,] 2020  1.110  1.103  1.111  0.899  0.808  0.774  0.766  0.837  0.797
## [342,] 2020 85.000 85.000 84.000 83.000 83.000 83.000 83.000 84.000 83.000
## [343,] 2021  0.652  0.583  0.722  0.704  0.720  0.787  0.832  0.810  0.802
## [344,] 2021 82.000 82.000 82.000 82.000 82.000 82.000 82.000 83.000 83.000
## [345,] 2022  0.830  0.772  0.828  0.776  0.709  0.805  0.768     NA     NA
## [346,] 2022 81.000 81.000 82.000 81.000 81.000 80.000 81.000  0.000  0.000
##           V11    V12    V13   V14
## [341,]  0.711  0.840  0.629 0.865
## [342,] 83.000 82.000 82.000    NA
## [343,]  0.801  0.778  0.740 0.744
## [344,] 82.000 81.000 80.000    NA
## [345,]     NA     NA     NA 0.784
## [346,]  0.000  0.000  0.000    NA
```

we can now check what are the number of rows and columns we have. we use the function length

```
n_column = length(values[1,])
n_column
```

```
## [1] 14
```

Literally translating to make a variable called *n_column* and then find the length of the first row which technically displays number of columns in it.

```
n_rows   = length(values[,1])
n_rows
```

```
## [1] 346
```

Literally translating to make a variable called *n_rows* and then find the length of the first column which technically displays number of rows in it.

The last column has means. We would like to calculate them ourselves . Therefore we remove it

```
values = values[, -n_column]
```

Which translates to: we have a variable *n_column* This variable has a value of 14. From our values variable remove in all rows, column 14. You can obeserve that in your data.

```
head(values)
```

```
##        V1     V2     V3     V4     V5     V6     V7     V8     V9    V10    V11
## [1,] 1850 -0.671 -0.296 -0.699 -0.508 -0.329 -0.237 -0.130 -0.207 -0.357 -0.442
## [2,] 1850 24.000 21.000 19.000 19.000 19.000 21.000 22.000 23.000 24.000 24.000
## [3,] 1851 -0.222 -0.328 -0.408 -0.465 -0.280 -0.247 -0.163 -0.170 -0.062 -0.053
## [4,] 1851 24.000 23.000 22.000 22.000 21.000 21.000 22.000 24.000 19.000 21.000
## [5,] 1852 -0.300 -0.445 -0.488 -0.568 -0.196 -0.056  0.068 -0.194 -0.093 -0.197
## [6,] 1852 24.000 23.000 23.000 23.000 24.000 24.000 24.000 24.000 22.000 23.000
##         V12    V13
## [1,] -0.150 -0.239
## [2,] 25.000 25.000
## [3,] -0.025 -0.048
## [4,] 19.000 21.000
## [5,] -0.196  0.114
## [6,] 23.000 26.000
```

we no longer have column V14

We now move to rows.The temp anomalies in the dataset are stored in even rows and the means in the odd rows we need to differentiate this rows.

```
even_row = seq(2,n_rows, 2)

odd_row = seq(1,n_rows, 2)
```

Which translates to select even rows that have anormalies, and odd rows that have percentages . Store them in even_row and odd_rows variables. We utilise the *seq* function. for *seq* Function the first value is from, the second value is to and the third value after a comma(,) is by.

For illusration we could create a variable that uses seq to generate number

```
seq(from = 2, to = 100, by = 20)
```

```
## [1]  2 22 42 62 82
```

```
#same as

seq(2,100,20)
```

```
## [1]  2 22 42 62 82
```

Now lets target the anomalies. We will store them in a variable called temp

```
temp = values[even_row,]
```

here we tell R to store into a **new** variable *temp* even rows of our original variable *values*

and now the percentages

```
perc = values[odd_row,]
```

here we tell R to store into a **new** variable *perc* odd rows of our original variable *values*

because in class we had peope working on different data sets, some recording were available from 1857 others 1850.The last row was also not well finished. so we remove the first seven rows and the last row too so that we can have the same length of data so to say

```
temp  = temp[-(1:7),]
```

Here we tell R that from variable *temp* remove rows 1:7

And we are going tell R to remove the last Row using **temp = temp[-length(temp[,1]),]**

But first lets break it down

first, from variable *temp* select column 1

```
temp[,1]
```

```
##   [1] 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871
##  [16] 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886
##  [31] 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901
##  [46] 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916
##  [61] 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
##  [76] 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946
##  [91] 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961
## [106] 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976
## [121] 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
## [136] 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006
## [151] 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021
## [166] 2022
```

Now measure the length of the selected column

length(temp[,1])

```
length(temp[,1])
```

```
## [1] 166
```

The length is 165. Now to remove the last row of the variable *temp* is easy because we know the index of the row.we save the result as variable *temp* this overides the value of the previous *temp*

Now if we print again our column 1 of new variable *temp*

```
temp[,1]
```

```
##   [1] 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866 1867 1868 1869 1870 1871
##  [16] 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886
##  [31] 1887 1888 1889 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901
##  [46] 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915 1916
##  [61] 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931
##  [76] 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946
##  [91] 1947 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961
## [106] 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976
## [121] 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991
## [136] 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006
## [151] 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020 2021
## [166] 2022
```

To remove it we simply put a -sighn in the code and then save the new variable as *temp* overifing the old *temp* variable.

```
temp  = temp[-length(temp[,1]),]
```

Lets check the new length

```
length(temp[,1])
```

```
## [1] 165
```

As you can see the length changed from 166 to 165

We do the same stuff on the *perc* variable

```
perc  = perc[-(1:7),]

perc  = perc[-length(perc[,1]),]
```

Now lets assighn names to the columns of variables *temp* and *perc*.

We use the function **colnames**

```
colnames(temp) = c("Year", "January" , "February", "March", "April", "May", "June", "July","Aug", "Sep"
colnames(temp)  =   c("Year", "January" , "February", "March", "April", "May", "June", "July","Aug", "S
```

Now lets export our data in form on a table. As i mentioned its good to have 2 folders one for data/input and another for results/output

```
path2 = "C:/Users/VICTOR_NYABUTI/Climate/output/"
```

Ware determined to work with the temperatures so we export the *temp* variable

```
write.table(temp,paste(path2,"Temparature_anomally.csv", sep = ""), sep = ",",col.names = TRUE, row.nam
```