University of British Columbia
Department of Statistics
Technical Report #244
May 2009

Extracting Canadian Climate Data from Environment Canada dataset

by

Reza Hosseini[1]
[1] University of British Columbia

Deaprtment of Statistics, University of British Columbia

ABSTRACT

A python package is developed to extract observed climate data from the binary files of the dataset published by Environment Canada which includes the daily temperature and precipitation from 1895-2006. The package is an extension and modification of the package developed by Bernhard Reiter (1999) and it includes python functions to write customized climate data files for several stations in the same time to use in different statistical packages. A guide to use the python package and a summary of the dataset is given.

1 Introduction

In this document, some instructions are given to use the climate data provided by environment Canada [1]. The data we are using are contained in a file, which can be downloaded from the environment Canada website

(http://www.weatheroffice.ec.gc.ca/).

"The National Climate Data and Information Archive, operated and maintained by Environment Canada, contains official climate and weather observations for Canada" (quoting from the website).

Environment Canada has published a series of climate data CDs: 1993, 1996, 2002, 2007. The newest version is the 2007 CD. The Environment Canada website also includes some other useful information, as a glossary of some useful terms in climate literature and also some information about the files. In particular, the glossary includes the definition of precipitation:

Precipitation: The sum of the total rainfall and the water equivalent of the total snowfall observed during the day.

On the 2007 CD, data are stored in a binary format in several files. The CD includes two softwares to use the data, "cdcd" and "cdex" along with manuals to use the softwares. "cdcd" is to view the data and "cdex" is to extract the data. "cdex" can only extract the data for one climate station at a time in certain formats which are not necessarily convenient to use in R (a well known statistical software) or other statistical softwares. In these formats the longitude, latitude and elevation are missing. Hence, to get the data in our desired way, we need to read the binary files using another program. Bernhard Reiter has written a code in Python [3] to get the data, which is available online at

http://www.intevation.de/∼bernhard/archiv/uwm/canadian_climate_cdformat/.

However, this code fails to get the data for a large proportion of the stations. We have modified the code to get the data for all stations. The modified code [2] is available at

http://bayes.stat.ubc.ca/∼reza/python.

After getting the data, we need to write the data in our desired formats. We have also included many new functions in Python for different extraction purposes.

There are 7802 stations from all over Canada. The available variables are:

1. maximum temperature

2. minimum temperature
3. one-day rainfall
4. one-day snowfall
5. one-day precipitation
6. snow depth on the ground

These data are available, both daily and monthly. For each station the data are available for different intervals of time.

The data are saved in 8 directories on the CD labeled $1, 2, \cdots, 8$. They correspond to different territories of Canada.

```
1 --> British Columbia

2 --> Yukon territories, Nunavut and North west territories

3 --> Alberta

4 --> Saskatchewan

5 --> Manitoba

6 --> Ontario

7 --> Quebec

8 --> Nova Scotia, New found land and Labrador
```

Each directory contains a number of data files and index files. For example, directory 3 which correspond to Alberta contains the following files: DATA.301, DATA.302,$\cdots$, DATA.308 and INDEX.301, INDEX.302,$\cdots$,INDEX.308. Each DATA file correspond to the data of a region in Alberta and the corresponding INDEX file contain the information about the stations in the given region. In Figure 1, you can see the location of available stations over Canada.

2 Using Python to extract data

In the following, we illustrate getting the data using the python module "Reza_canadian_data.py". After opening the python interface, let us import some necessary packages and tell python where the data are stored. Using sys.path.append specify the directory where Reza_canadian_data.py is stored as shown below. Also, define Topdirectory to be where the data are stored.

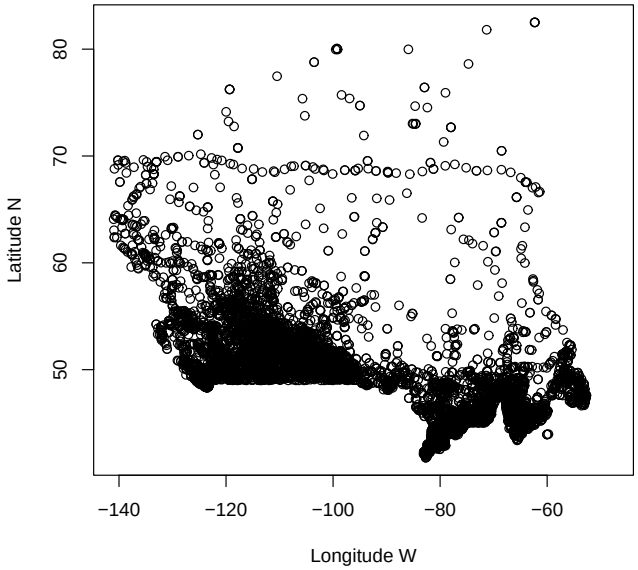

Figure 1: Canada site locations

```
>>>import sys
>>>sys.path.append("D:\School\Research\Climate\Python_code")
>>>Topdirectory="D:\Data"
>>>from Reza_canadian_data import *
>>>stations=get_station_list(Topdirectory)
```

Once you did that you can call the command get_station_list from Reza_canadian_data to get the list of the stations available on the CD. Let us see how many stations we have access to:

```
>>> len(stations)
7802
```

Let us pick a random station, say the 3000-th station and find out its id and index.

```
>>> s=stations[2436]
>>> s.stationnumber
'3025480'
>>> s.index_record
('5480', 'RED DEER A              ', 'YQF', 5211, 11354, 905, 1938,
1938, 1938, 1938, 1938, 1938, 1955, 2007, 2007, 2007, 2007, 2007,
```

```
2007, 2007, 9904)
>>> len(s.index_record)
21
```

The command "stationnumber" gives back the id of the given station on the CD. The stations in the same district start with the same numbers. For example the stations in Alberta all start with 30 and so Red Deer is in Alberta. You can use cdcd to see the list of the stations and id numbers to figure out which ids correspond to which districts.

The index_record command reads the information available for the given station. There are many values available and it is hard to understand what they mean. As you see the index has 21 components. Here is the explanation of each component:

1. The last four digits of the id
2. station name
3. Airport is the three-character airport identifier that some stations have (e.g., "YWG" for Winnipeg); if none exists for this station then the field is left blank
4. latitude
5. longitude
6. elevation
7. The first available year for max temperature
8. The first available year for min temperature
9. The first available year for mean temperature
10. The first available year for rainfall
11. The first available year for snowfall
12. The first available year for snow depth
13. The first available year for precipitation
14. The last available year for Max temperature
15. The last available year for min temperature
16. The last available year for mean temperature
17. The last available year for rainfall

18. The last available year for snowfall
19. The last available year for precipitation
20. The last available year for snow depth
21. Starting Record Number: This record is a header that contains information about the station

Hence, for example this station name is Red Deer. It has the data for precipitation from 1938 to 2007. Whenever, 9999 is recorded as the first and 55537 as the last available year for a variable, that variable is missing. As mentioned before the available data for a given station are maximum temperature, minimum temperature, one-day rainfall, one-day snowfall, one-day precipitation, snow depth. These are coded in Reza_canadian_data.py as

```
"MT" "mint" "rain" "snow" "precip" "snow_ground"
```

We have used the following procedure in python interface to create a file "stations.txt", which has the information for all the available stations. In every row the information for a stations is given. There are 22 columns, the first one is the stations id and the other 21 are as described above. Whenever, the station was not an airport station, the airport identifier was recorded as NA. Notice, how using the "if" command in below, we have separated the case where the airport identifier is blank from the case that there is an airport identifier.

```
stations=get_station_list(Topdirectory)

f=open('stations.txt','w') for s in stations:
        ind=s.index_record

        if ind[2]=='   ':
                f.write(str(s.stationid)+','+str(ind[0])+','+str(ind[1])
                +','+'NA'+','+str(ind[3])+','+str(ind[4])+','+str(ind[5])
                +','+str(ind[6])+','+str(ind[7])+','+str(ind[8])
                +','+str(ind[9])+','+str(ind[10])+','+str(ind[11])
                +','+str(ind[12])+','+str(ind[13])+','+str(ind[14])
                +','+str(ind[15])+','+str(ind[16])+','+str(ind[17])
                +','+str(ind[18])+','+str(ind[19])+','+str(ind[20])
                +'\n')
        else:
                f.write(str(s.stationid)+','+str(ind[0])+','+str(ind[1])
                +','+str(ind[2])+','+str(ind[3])+','+str(ind[4])
                +','+str(ind[5])+','+str(ind[6])+','+str(ind[7])
```

```
                +','+str(ind[8])+','+str(ind[9])+','+str(ind[10])
                +','+str(ind[11])+','+str(ind[12])+','+str(ind[13])
                +','+str(ind[14])+','+str(ind[15])+','+str(ind[16])
                +','+str(ind[17])+','+str(ind[18])
                +','+str(ind[19])+','+str(ind[20])+'\n')

f.close()
```

One of the useful commands in Reza_canadian_data.py is get_data. Let us use this command to get some data.

```
data=s.get_data(1995,``precip")
>>> len(data)
3
>>> len(data[0])
366
>>> len(data[1])
366
>>> len(data[2])
108
```

As you see the data object created has three components. The first two components each have 366 entries and the third one has 108 components. The first component of the data is the data values for each day of the year, the amount of precipitation. The second component includes the flag associated with each daily values. The third component correspond to monthly values, number of missing days for a given month and etc. Let us look at the first two components. We print the value of precipitation for the first 60 days of the year:

```
>>> for precip in data[0][0:60]:
         print "%5.1f" % precip,
  0.0   0.2   0.0   0.0   0.0   0.5   0.0   0.0   0.0   2.0   0.8
  0.0   0.2   0.4   2.2   0.4   0.6   0.0   0.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.0   0.2   0.0   0.4   0.0   0.0   0.0   0.0
  0.0   0.2   1.2   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
  0.0   3.0   0.0   0.0 -999.9
```

Everything looks OK other than the last vale -999.9. Every missing value in the dataset is shown by -999.9. In fact to see the status of a data point look the corresponding flag which is given in the second component of the data. Let us look at the flag for the first 60 days of the year as well:

```
>>> for flag in data[1][0:60]:
         print "%5.1f" % flag,


  0.0   0.0   0.0   0.0   2.0   0.0   2.0   2.0   2.0   0.0   0.0
  2.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   2.0
  2.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0   2.0   2.0   2.0
  0.0   0.0   0.0   2.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0
  2.0   0.0   2.0   0.0  14.0
```

We need to know what each flag means. Please note that the flag corresponding to -999.9 is 14. Below is a description of flags:

```
  0 ->  Observed value
  1 ->  Estimated
  2 ->  Trace. Value is reported 0
  3 ->  Precipitation occurred, amount uncertain; value is reported 0
  4 ->  Precipitation may or may not have occurred; value is reported 0
  5 ->  Accumulated amount (from past days possibly)
  6 ->  Accumulated and estimated
  7 to 13 -> unused
  14 -> This is used to denote Feb 29 in a non-leap year
  15 -> Missing data
```

In summary only a data point with 0 flag (no flag) is valid. The flag 2.0 corresponds to "trace" (as called by Environment Canada) which is a precipitation under 0.2 (mm) that can not be measured accurately and the value is reported as zero. In the above example the flag corresponding to -999.9 is 14 which is to denote Feb 29 in a non-leap year as explained above and this makes sense since the 60th day of the year correspond to Feb 29th. There are 13 points flagged 2.0 and so we have many "trace" values. For more information regrading the flags and data format please refer to 'Reza_canadianinfo.txt"

In order to extract and interpret the data, one need to read each data point as well as the flag corresponding to the data point.

3 New functions to write stations' data

In the python package, "Reza_canadian_data.py", we have also introduced some functions to write the data for a given station including the stations information as longitude latitude and elevation. Using these commands has the advantage that we do not need to worry about the flags anymore. Whenever, the data is missing we

will get NA (instead of -999.9) and also for trace values for precipitation, we get trace. The command for getting the data for a given stations is "write_station(, ,)". We need three entries for this command: stationnumber, the list of all stations (We can get that by the command stations=get_station_list(Topdirectory) as shown above.) and value ("MT","mint","rain","snow","precip").

For example,

```
>>>write_station(2436,stations,'MT')
```

The output for this command (if the data are available) is a txt file. There is also an output in the terminal. If the data is available the output is "success" and the name of the txt file created. If the data is not available then the output is simply "failure". A statement is also printed depending on the data being available or not. If the data are available, the number of years the data are available is reported and also the name of the file created. For the above example, we get

```
The data file 3025480-MT.txt created. It should contain 69 years.
('success', '3025480-MT.txt')
```

If the data were not available we would get:

```
There was not any years containing more than 100 days. No file
created. ('failure','none')
```

Also note that, we only write data for a year if it contains more than 100 days of data. You can modify this easily by modifying the write function in the module.

The data files are named by the id followed by "MT", "mint" or "precip" which stand for Max temperature, min temperature and precipitation respectively. For example, since the id for ABBOTSFORD's id is "1100030" the file containing the data for maximum temperature for ABBOTSFORD is called "1100030-MT.txt"

Each row of the data files correspond to a year. The first entry is the year and then 366 entries corresponding to the observed daily values for the given year. Whenever the actual year is 365 days only, the value corresponding to Feb 29th is recorded as NA (60th day of the 366 year).

"NA" stands for the missing data. For precipitation (only) "trace" correspond to a wet day with an amount less than 0.2 (mm).

Note that we can use this command in a "for" loop to write a bunch of stations. Suppose, we are given a list of stations. If we want to keep track of the stations that have data available, we can create the list of the files created. In the following we have done that with stations_list which contains the number of the stations that have data available. stations_list contains the name of the files created.

```
list=733,4034,2517,7467,6744,1518,2113,7269 subset=list value='MT'
stations_list=[] stations_files=[]

for i in subset:
        snum=i
        d=write_station(snum,stations,value)
        if d[0]=='success':
                stations_list.append(i)
                stations_files.append(d[1])
```

4 Concluding remarks

The software described in this report can be used to generate datasets suitable for analysis with R and other standard datasets. Moreover the tutorials and demonstrations should help users understand the process for doing so.

References

[1] Environment Canada. The climate cds. *http://www.weatheroffice.ec.gc.ca*, 2007.

[2] R. Hosseini. Python module for canadian climate data. *http://bayes.stat.ubc.ca/∼reza/python*, 2009.

[3] B. Reiter. Canadian daily climate data cd reader. *http://www.intevation.de/∼bernhard/archiv/uwm/canadian_climate_cdformat*, 1999.