Using R to read NetCDF Files

Method · September 2017

DOI: 10.17605/OSF.IO/8XCHE

Venki Uddameri
Texas Tech University

Using R to read NetCDF Files

Venki Uddameri

09/02/2017

Abstract Hydrologic data are collected in space and time. Global (Gridded) datasets for hydrometerological variables such as temperature and precipitation can now be readily downloaded. These data are generally stored in Network Common Data Format (NetCDF). NetCDF is a machine independent data format that store big datasets in a compact format. In this tutorial we shall explore how NetCDF files can be opened and analyzed in R Project for Statistical Computing environment.

Keywords Big Data · Data Interchange · Gridded Datasets

1 Introduction

Hydrometeorological (HM) data are often collected in space and time. Data for commonly measured variables such as temperature and precipitation are available worldwide and in many places have been collected over a long period of time. While HM data are often collected at irregularly spaced locations, they have been gridded to produce uniformly spaced datasets [1] particularly to support global climate change modeling efforts [2]. Gridded space-time data can be represented as a three-dimensional array (longitudes represent X-axis, latitudes the Y-axis and time the Z-axis). The netCDF is a common data format to store such data.

2 What is NetCDF

The Network Common Data Format (NetCDF) is a self-describing machine-independent data format that was developed by Unidata. Unidata is a member of the University Corporation for Atmospheric Research (UCAR) and funded by the National Science Foundation (NSF). The NetCDF data has been around since 1989. The most recent version is the NetCDF version 4.0 which was released in 2008. The NetCDF version 3 released in 1997 is also still in use today. The netCDF version 4 is somewhat compatible with the **Hierarchical Data Format version 5 (HDF5)** which is also another popular data format for working with large-array datasets.

The NetCDF is a **self-describing** which means the file contains a header which includes information on the variables used and the structure of the data. Generally, the meta-data is stored in a separate file. The data is also **platform-independent** which means data can be ported between different operating systems with out any problems. The netCDF file format is based on the common data format for scientific datasets and there is a growing effort to make it compatible with other data formats used to store scientific data [3]. NetCDF files have the extension ".nc". You can learn more about the Unidata common data format by visiting their web-site:

```
http://www.unidata.ucar.edu/software/thredds/current
    /netcdf-java/CDM/
```

3 Structure of NetCDF Data

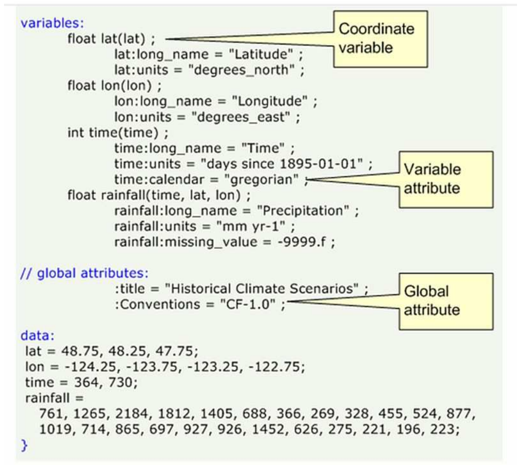
A NetCDF file stores data in arrays. A **variable** represents a hydrometeorological phenomenon of interest (e.g., temperature, precipitation). A variable collected at a single location at various times is stored as a one-dimensional array. In a similar manner, a variable collected at several locations at a single point in time is stored as a two-dimensional array. Data collected at several locations at different times is stored in a three-dimensional array. A four-dimensional data storage is adopted when data are collected at multiple geographic locations and at various depths (heights) and times at each of these locations. A netCDF file consists of following basic components - **dimensions**, **variables** and **attributes**, I will describe these terms briefly.

– **Dimensions** have names (usually lat, lon and time) and a size. The size is an integer indicating how many values are stored along each named dimension (e.g., data collected at 3 locations and 4 times would have dimensions

Venkatesh Uddameri
Civil, Environmental& Construction Eng., Texas Tech University
Tel.: 806-834-8340
E-mail: venki.uddameri@ttu.edu

- lat = 3, lon = 3, time= 4. Only one dimension can have UNLIMITED dimension. In gridded datasets, time is set to have unlimited dimension to allow addition of data as it becomes available.
- **Dimensions** represent an array of values of the same type. Variables have a name, data type and a shape that determined by the number of dimensions. Variables are used to store bulk of the data. **Coordinate Variables** are special variables which store data pertaining to dimension variables (typically - lat, lon and time).
- **Attributes** are used to store ancillary data or metadata. Attributes associated with variables are identified using both variable and attribute name (e.g., rainfall:units = mm/yr; stores the units for the rainfall variable). A **global attribute** stores information about the NetCDF file as a whole.

A network Common Data Language (CDL) is the ASCII format used to describe a NetCDF file. An illustrative example of a netCDF file structure using CDL is shown in Figure 1



```
variables:
        float lat(lat) ;
                lat:long_name = "Latitude" ;
                lat:units = "degrees_north" ;
        float lon(lon) ;
                lon:long_name = "Longitude" ;
                lon:units = "degrees_east" ;
        int time(time) ;
                time:long_name = "Time" ;
                time:units = "days since 1895-01-01" ;
                time:calendar = "gregorian" ;
        float rainfall(time, lat, lon) ;
                rainfall:long_name = "Precipitation" ;
                rainfall:units = "mm yr-1" ;
                rainfall:missing_value = -9999.f ;

// global attributes:
                :title = "Historical Climate Scenarios" ;
                :Conventions = "CF-1.0" ;

data:
 lat = 48.75, 48.25, 47.75;
 lon = -124.25, -123.75, -123.25, -122.75;
 time = 364, 730;
 rainfall =
   761, 1265, 2184, 1812, 1405, 688, 366, 269, 328, 455, 524, 877,
   1019, 714, 865, 697, 927, 926, 1452, 626, 275, 221, 196, 223;
}
```

Fig. 1 Components of a NetCDF File

4 NetCDF and R

R offers several packages to manipulate NetCDF files. **Raster**, **climates**,**RNetCDF**,**ncdf4** as well as **ncdf4.helpers** provide functionality to work with NetCDF datasets. In this tutorial we shall make use of the **ncdf4** package [4] as it can deal with both **NetCDF version 3** and **NetCDF version 4** formats and is available on both windows and Macintosh platforms. This package allows us to both read and write NetCDF files. In this tutorial our major focus will however be on reading NetCDF files.

5 Working with NetCDF File in R

The **ncdf4** package can be installed from any CRAN mirror. This package must be installed and loaded into the memory of the R session prior to its use. Some important functions for manipulating NetCDF data in **ncdf4** package are summarized in Table 1.

Table 1 Summary of Certain Important Functions in NetCDF

Purpose	**Purpose**
nc_open	Open a NetCDF File
print	Prints the built-in metadata
ncvar_get	Get data for variables
ncatt_get	Get information on attributes
nc_close	Close the netcdf datafile

6 Illustrative Example

6.1 Problem Statement

Droughts occur when there is less than normal moisture within a system, largely due to insufficient precipitation. Droughts can result in billions of dollars losses due to loss of agricultural productivity and lack of water for energy production and other municipal and industrial activities. While there are many drought indicators, the Palmer Drought Severity Index (PDSI) is one widely used metric to assess droughts. PDSI uses the concept of water budgets along with temperature and precipitation data to evaluate droughts. A limitation of PDSI is that it uses several empirical constants that can affect the accuracy of predictions and make it difficult to compare values across different locations. A self-calibrated PDSI (PDSI-sc),improves drought predictions by dynamically updating several empirical constants within the original PDSI which in turn allows comparison of PSDI across sites [5]. A global, historical PDSI-sc dataset has been made available recently (as you guessed it is in the NetCDF format). We seek to make use of this dataset and extract historical PDSI-sc data for Lubbock, TX $(33.5779^oN, 101.8552^oW)$.

6.2 Dataset

To illustrate the manipulation of NetCDF files in **R**, we shall make use of the self-calibrated Palmer Drought Severity Index (PDSI-sc). A global scale PDSI-sc dataset was developed to study the effects of global warming on soil moisture [6]. The dataset has been periodically updated and is also referred to as the Dai-PDSI dataset. It is available for download from:

```
http://www.unidata.ucar.edu/software/thredds/current
    /netcdf-java/CDM/}
```

Download the PDSI-SC dataset which has gridded monthly mean PDSI-sc values computed on a 2.5 ^oC x 2.5 ^oC grid.

```
\textbf{pdsi.mon.mean.selfcalibrated.nc}
```

There are 144 columns (longitudes in the range of -178.75^o : 178.75^o) and 55 rows (latitudes in the range of -58.75^o : 76.25^o) with negative values for longitudes in the western hemisphere and negative latitudes in the southern hemisphere. The file size is approximately 62 MB. To begin the exercise download the file and place it in a separate directory on your computer. . Although not necessary, I have renamed the file **pdsisc.nc** for simplicity.

6.3 R Code

The **R** code for performing the analysis is shown below. Each step of the code is annotated to explain what it does. The main functions are nc _open which opens the NetCDF File and ncvar _get which can be used to extract data. Note that each variable (including coordinate variables) have a defined structure (one, two, three or multidimensional arrays). Data can be subset specifying the necessary cells in the **ncvar _get** you need to specify the starting point of the extraction in the array using **start** (in our case, lon, lat and time) and **count** (in our case, the number of lon, lat and time values to be extracted). We can use -1 to extract all values in a particular dimension. For example count(1,1,-1) in our case will extract values for one longitude, one latitude and all for all values of time.

```
# A script to read NetCDF Data
# Written by Venki Uddameri, Ph.D., P.E.

# Step 1: Load Libraries and Set Working Data
library(ncdf4)
setwd('E:\\Dropbox\\CE5333-DataAnalysis\\Module1-
    Data')

# Step 2: Read ncdf file into an R object
pdsi <- nc_open('pdsisc.nc')

# Step 3: Read the metadata and capture it
metadata <- capture.output(print(pdsi))

# Step 4: Extract the longitudes and latitudes
lon <- ncvar_get(pdsi,"lon") # starts at -178.5
lat <- ncvar_get(pdsi,'lat') # starts at -58.75

# Step 5: Extract data for Lubbock
# We shall extract the data from the 2.5 x 2.5 cell
# that has Lubbock in it
# Step 5a: Set the Coordinates for Lubbock
coord.lbb <- c(-101.8552,33.5779)
# Step 5b: Identify the cell which has Lubbock
# We need to select 38 row (lat) and 32 column (lon)
idx.lon <- ceiling((coord.lbb[1]-lon[1])/2.5)+1
idx.lat <- ceiling((coord.lbb[2]-lat[1])/2.5)+1

# Step 6: Extract PDSI values for all times for
    Lubbock and close

pdsisc.lbb <- ncvar_get(pdsi,"pdsi",start=c(idx.lon,
    idx.lat,1),
                   count = c(1,1,-1))
# -1 can be used in count to obtain all values (of
    time in this case)
nc_close(pdsi) # Close the netCDF file

# Step 7: Create a monthly time-series object
pdsisc.lbb.ts <- ts(pdsisc.lbb,start=c(1850,1),
    frequency=12)

# Step 8: Plot the Time-Series Object
plot(pdsisc.lbb.ts,ylab='PDSI-sc')
grid() # Add grid for aesthetics
abline(h=-1.0,col='red',lwd=2,lty=2) # add the upper
    limit of drought
# Add a rectangle between the years 1947 - 1957
# to depict the drought of record
rect(1947,-7,1957,8, col= rgb(0.211,0.211,0.211,
    alpha=0.25))
```

7 Outputs and Results

7.1 Evaluating Metadata

Information about variables is self-contained within the NetCDF file. It is always a good idea to see this information using the **print** command. A partial listing is shown below. The print statement reveals the variables and their data structure (e.g., pdsi has the data type float and is denoted by three dimensions - lon, lat and time). The number of latitudes, longitudes and times can also be ascertained.

```
pdsisc.nc (NC_FORMAT_CLASSIC):

    1 variables (excluding dimension variables):
      float pdsi[lon,lat,time]
          statistic: Mean
          missing_value: -99999
          dataset: Dai Palmer Drought Severity Index
              : Self-calibrated
          long_name: Monthly Self-calibrated Palmer
              Drought Severity
          Index using Penman-Monteith PE
          level_desc: Surface
          var_desc: Palmer Drought Severity Index
          least_significant_digit: 2
          units: Standardized Units of Relative Dry
              and Wet
          actual_range: -8.16020011901855
           actual_range: 8.16014289855957
          valid_range: -100
           valid_range: 100

    3 dimensions:
      lon  Size:144
          units: degrees_east
```

```
        long_name: Longitude
        actual_range: -178.75
         actual_range: 178.75
        standard_name: longitude
        axis: X
    lat  Size:55
        units: degrees_north
        long_name: Latitude
        actual_range: -58.75
         actual_range: 76.25
        standard_name: latitude
        axis: Y
    time  Size:1980  *** is unlimited ***
        units: hours since 1800-01-01 00:00:0.0
        actual_range: 438288
         actual_range: 1883904
        long_name: Time
        delta_t: 0000-01-00 00:00:00
        avg_period: 0000-01-00 00:00:00
        standard_name: time
        axis: T
```

7.2 Time in NetCDF Files

The metadata obtained using **print** command shows that time is given as minutes from ”01-010-1800 00:00:00”. This is usually done so that the time data can be stored as integers and reducing the memory requirements of the file. As we know that the website that data are on monthly time-step starting at 01-01-1850, we can directly create a time-series object and plot the time-series data here (see Figure 2). We can easily convert the hours since 01-01-1800 data to dates with a few lines of code in **R**.

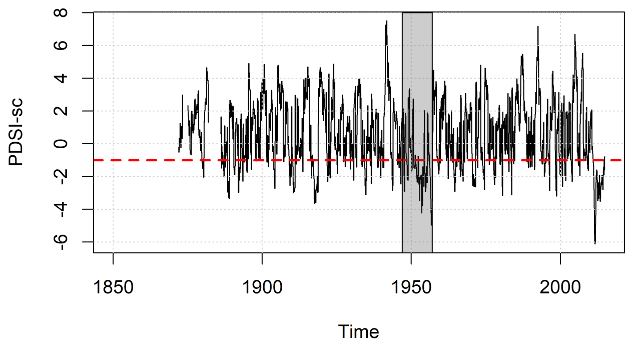


Fig. 2 Historical PDSI at Lubbock, TX (the Drought of Record is highlighted)

```
# Set start date
start.date = as.Date('1800-01-01')
# Extract time in hours since start date
times.hr <- ncvar_get(pdsi,'time')
# Convert to days
times.day <- times.hr/24
# add the days to start date create a series
datex <- start.date + times.day
```

8 Closing Remarks

In this short tutorial we saw how to open and extract data from an NetCDF file in **R**. In particular, we say the functionality in ncdf4 package. A useful feature of NetCDF format is it is easily subsettable and we do not need to load the entire data into memory, if we just want data at a few locations.

9 Acknowledgments

We acknowledge the support of NOAA/OAR/ESRL PSD, Boulder, Colorado, USA, for freely making available the Dai Palmer Drought Severity Index data at their Web site - http://www.esrl.noaa.gov/psd/

References

1. H.E. Beck, A.I. van Dijk, V. Levizzani, J. Schellekens, D.G. Miralles, B. Martens, A. de Roo, Hydrology and Earth System Sciences **21**(1), 589 (2017)
2. I. Harris, P. Jones, T. Osborn, D. Lister, International Journal of Climatology **34**(3), 623 (2014)
3. S. Nativi, J. Caron, B. Domenico, L. Bigagli, Earth Science Informatics **1**(2), 59 (2008)
4. D. Pierce, R package, URL http://CRAN. R-project. org/package= ncdf4 (2012)
5. N. Wells, S. Goddard, M.J. Hayes, Journal of Climate **17**(12), 2335 (2004)
6. A. Dai, K.E. Trenberth, T. Qian, Journal of Hydrometeorology **5**(6), 1117 (2004)