R script to convert netCDF Climate dataset to Quicklook

1. Reading a netCDF data set using the ncdf4 package

The `ndcf4` package is used to read, write and analyze netCDF files.The `netCDF` package is available in both windows and MAC OS X and Linux and supports both older NetCDF3 format as well as netCDF4. To begin, load the ncdf4 package

```
library(ncdf4)
```

```
## Warning: package 'ncdf4' was built under R version 3.4.4
```

`fn` is the path where the file is located and cru10min30_tmp.nc is the file name.

`name` is the name of the variable that will be read in

2. Open the netCDF file

```
#Set path and filename
fn <- "F:\\IntroductionEarthData\\data_samples\\netCDF\\cru10min30_tmp.nc"
name <- "tmp"    # tmp means temperature
```

Open the NetCDF dataset and print basic information. The `print()` function applied to `nc` object provides information about the dataset

```
#open a netCDF file
nc<- nc_open(fn)
print(nc)
```

```
## File F:\IntroductionEarthData\data_samples\netCDF\cru10min30_tmp.nc (NC_FORMAT_CLASSIC):
##
##      2 variables (excluding dimension variables):
##         float time_bounds[nv,time]
##         float tmp[lon,lat,time]
##             long_name: air_temperature
##             units: degC
##             _FillValue: -99
##             source: E:\Projects\cru\data\cru_cl_2.0\nc_files\cru10min_tmp.nc
##
##      4 dimensions:
##         lon  Size:720
##             standard_name: longitude
##             long_name: longitude
##             units: degrees_east
##             axis: X
##         lat  Size:360
##             standard_name: latitude
##             long_name: latitude
##             units: degrees_north
##             axis: Y
##         time  Size:12
##             standard_name: time
##             long_name: time
##             units: days since 1900-01-01 00:00:00.0 -0:00
##             axis: T
##             calendar: standard
##             climatology: climatology_bounds
##         nv  Size:2
##
##     7 global attributes:
##         data: CRU CL 2.0 1961-1990 Monthly Averages
##         title: CRU CL 2.0 -- 10min grid sampled every 0.5 degree
##         institution: http://www.cru.uea.ac.uk/
##         source: http://www.cru.uea.ac.uk/~markn/cru05/cru05_intro.html
##         references: New et al. (2002) Climate Res 21:1-25
##         history: Wed Oct 29 11:27:35 2014: ncrename -v climatology_bounds,time_bounds cru10min30_tmp.nc
## P.J. Bartlein, 19 Jun 2005
##         Conventions: CF-1.0
```

2.1. Get Coordinate including time variables

`ncvr_get()` function is used to read the coordinate variables `longitude` and `latitude`. `head()` and `tail()` functions are used to list first few values and the number of variables can be verified using `dim()` function:

```
##get longitude and latitude
lon <- ncvar_get(nc,"lon")
nlon <- dim(lon)
head(lon)
```

```
## [1] -179.75 -179.25 -178.75 -178.25 -177.75 -177.25
```

```
lat <- ncvar_get(nc,"lat")
nlat <- dim(lat)
head(lat)
```

```
## [1] -89.75 -89.25 -88.75 -88.25 -87.75 -87.25
```

```
print(c(nlon,nlat))
```

```
## [1] 720 360
```

Time variable and its attributes are derived by `ncvar_get()` and `ncatt_get()` functions and the dimensions of the time is obtained using `dim()` function

```
##get time
time <- ncvar_get(nc,"time")
time
```

```
##  [1] 27773.5 27803.5 27833.5 27864.0 27894.5 27925.0 27955.5 27986.5
##  [9] 28017.0 28047.5 28078.0 28108.5
```

```
tunits <- ncatt_get(nc,"time","units")
nt <- dim(time)
nt
```

```
## [1] 12
```

Print the time units string. It can be noticed that the structure of the object tunits has two components hasatt (a logical variable), and tunits$value, the actual “time since” string.

```
tunits
```

```
## $hasatt
## [1] TRUE
##
## $value
## [1] "days since 1900-01-01 00:00:00.0 -0:00"
```

2.2. Get a variable

Get a variable `tmp` and its attribute and verify the size of the array

```
#get temperature

tmp_array <- ncvar_get(nc,name)
dlname <- ncatt_get(nc,name,"long_name")
dunits <- ncatt_get(nc,name,"units")
fillvalue <- ncatt_get(nc,name,"_FillValue")
dim(tmp_array)
```

```
## [1] 720 360  12
```

Get the global attributes

```
#get global attributes
title <- ncatt_get(nc,0,"title")
institution <- ncatt_get(nc,0,"institution")
datasource <- ncatt_get(nc,0,"source")
references <- ncatt_get(nc,0,"references")
history <- ncatt_get(nc,0,"history")
Conventions <- ncatt_get(nc,0,"Conventions")
```

Close the netCDF file

Check the current workspace:

```
ls()
```

```
##  [1] "Conventions" "datasource"  "dlname"      "dunits"      "fillvalue"
##  [6] "fn"          "history"     "institution" "lat"         "lon"
## [11] "name"        "nc"          "nlat"        "nlon"        "nt"
## [16] "references"  "time"        "title"       "tmp_array"   "tunits"
```

3. Reshaping from raster to rectangular

NetCDF files or data sets are naturally raster slabs (e.g. a longitude by latitude “slice”), bricks(longitude by latitude by time), or 4-d arrays(longitude by latitude by height by time) while most data analysis routines in R expect 2-d variable-by-observation data frames. In addition, time is usually stored as the CF (Climate Forecast) “time since” format that is not usually human-readable.

Install and Load the below packages

```
#load some packages
library(chron)
```

```
## Warning: package 'chron' was built under R version 3.4.4
```

```
library(lattice)
library(RColorBrewer)
```

```
## Warning: package 'RColorBrewer' was built under R version 3.4.4
```

3.1.Convert the time variable

The time variable in “time-since” units is converted into readable form. `Chron()` function is used to determine the absolute value of each time value from time origin.

```
# convert time -- split the time units string into fields
tustr <- strsplit(tunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth <- as.integer(unlist(tdstr)[2])
tday <- as.integer(unlist(tdstr)[3])
tyear <- as.integer(unlist(tdstr)[1])
chron(time,origin=c(tmonth, tday, tyear))
```

```
##  [1] (01/16/76 12:00:00) (02/15/76 12:00:00) (03/16/76 12:00:00)
##  [4] (04/16/76 00:00:00) (05/16/76 12:00:00) (06/16/76 00:00:00)
##  [7] (07/16/76 12:00:00) (08/16/76 12:00:00) (09/16/76 00:00:00)
## [10] (10/16/76 12:00:00) (11/16/76 00:00:00) (12/16/76 12:00:00)
```

3.2. Replace netCDF fillvalues with R NAs

The missing values are flagged using specific `(_FillValues)` or `(missing_value) in netCDF files. The missing values are treated by replacing unavailable data using` value.

```
# replace netCDF fill values with NA's
tmp_array[tmp_array==fillvalue$value] <- NA
```

```
length(na.omit(as.vector(tmp_array[,,1])))
```

```
## [1] 62961
```

3.3. Get a single time slice of data

NetCDF variables are read and written as one-dimensional vectors (e.g. longitudes), two-dimensional arrays or matrices (raster “slices”), or multi-dimensional arrays (raster “bricks”). In such data structures, the coordinate values for each grid point are implicit, inferred from the marginal values of, for example, longitude, latitude and time. In contrast, in R, the principal data structure for a variable is the data frame. In the kinds of data sets usually stored as netCDF files, each row in the data frame will contain the data for an individual grid point, with each column representing a particular variable, including explicit values for longitude and latitude (and perhaps time). In the example CRU data set considered here, the variables would consist of longitude, latitude and 12 columns of long-term means for each month, with the full data set thus consisting of 259200 rows (720 by 360) and 14 columns.

This particular structure of this data set can be illustrated by selecting a single slice from the temperature “brick”, turning it into a data frame with three variables and 720 by 360 rows,

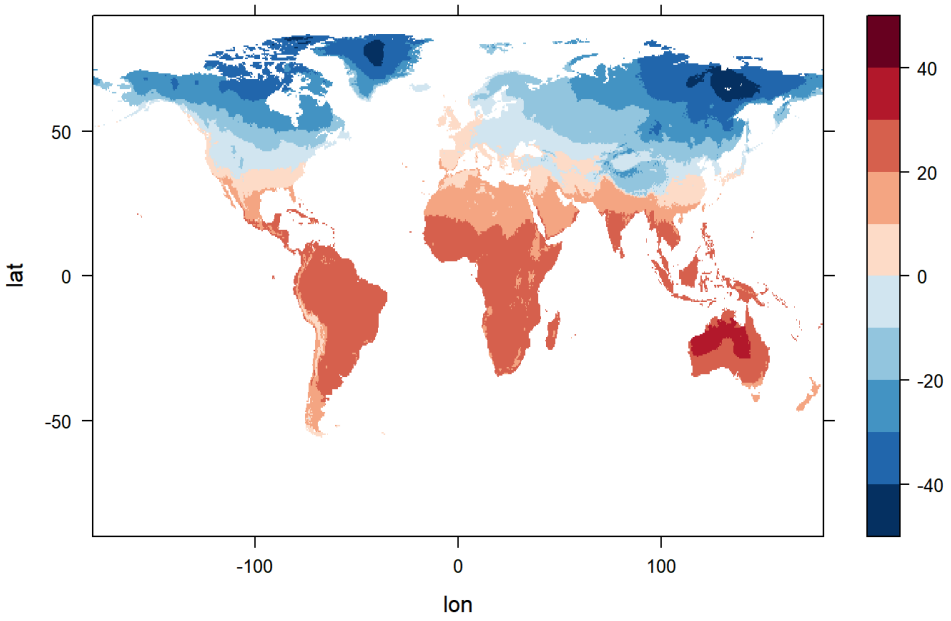
```
# get a single slice or layer (January)
m <- 1
tmp_slice <- tmp_array[,,m]
```

The dimensions of `tmp_slice`, e.g. 720, 360, can be verified using the `dim()` function.

4. Visualization

A quick look (map) of the extracted slice of data can be obtained using the `image()` function.The `expand.grid()` function is used to create a set of 720 by 360 pairs of latitude and longitude values (with latitudes varying most rapidly), one for each element in the `tmp_slice` array. Specific values of the cutpoints of temperature categories are defined to cover the range of temperature values

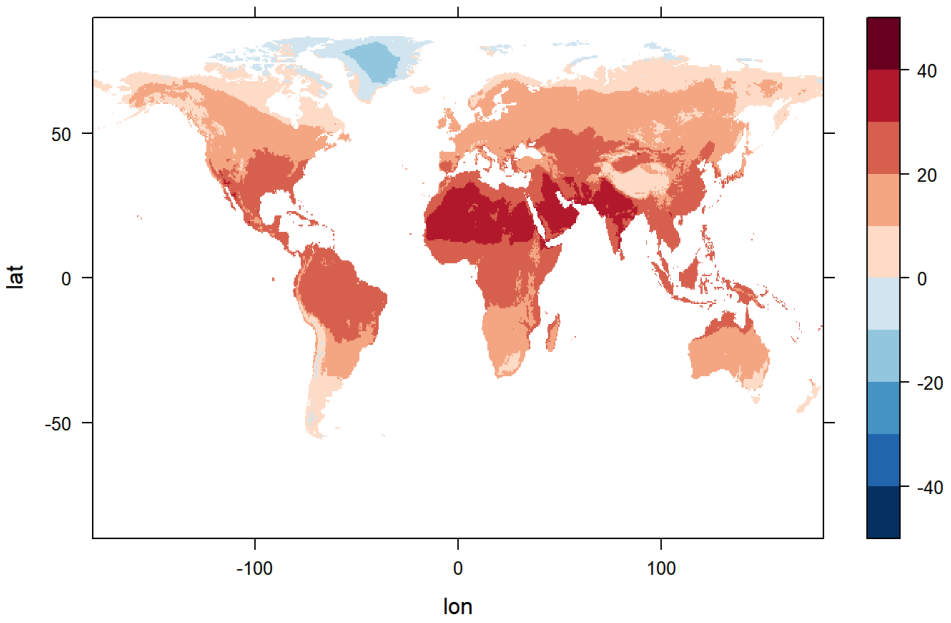
```
# quick map
grid <- expand.grid(lon=lon, lat=lat)
cutpts <- c(-50,-40,-30,-20,-10,0,10,20,30,40,50)
levelplot(tmp_slice ~ lon * lat, data=grid, at=cutpts, cuts=11, pretty=T,
col.regions=(rev(brewer.pal(10,"RdBu"))))
```



Quicklook of slice of data with different month, red color indicates the variation of increase in the temperature and Blue color indicates the winter period. Consider for example during the month `June` we can notice that most part of the world is experiencing an increase in the temperature

```
# June Month
m <- 6
tmp_slice <- tmp_array[,,m]
```

```
# quick map
grid <- expand.grid(lon=lon, lat=lat)
cutpts <- c(-50,-40,-30,-20,-10,0,10,20,30,40,50)
levelplot(tmp_slice ~ lon * lat, data=grid, at=cutpts, cuts=11, pretty=T,
  col.regions=(rev(brewer.pal(10,"RdBu"))))
```


50
0
-50
lat
-100
0
100
lon
40
20
0
-20
-40