Comment lire des données NetCDF avec R

Partie 1 : avec les packages ncdf4 et raster

Cyril Bernard, CEFE

Introduction

De nombreuses ressources satellitaires sont distribuées dans le format NetCDF. Bien souvent, il s'agit de séries temporelles, car le NetCDF est bien adapté pour ce type de données. Un seul fichier peut contenir plusieurs variables.

Le but de ce document est de montrer comment lire des données avec le package `ncdf4` pour accéder aux métadonnées ou aux données, puis avec le package `raster`, pour lire directement des données spatialisées sous forme de **grilles raster**.

Exemples de fichier NetCDF

Dans ce document, 2 jeux de données au format NetCDF sont utilisés :

- CHIRPS : une grille mondiale à la résolution 0.05° et 0.25° avec une estimation quotidienne des precipitations (https://www.chc.ucsb.edu/data/chirps) de 1981 à aujourd'hui
- CRU TS Version 4.03 : une grille mondiale à la résolution 0.5° avec 9 variables climatiques (http://www.cru.uea.ac.uk/data/) de 1901 à 2016

Données à télécharger

Pour exécuter les commandes R qui suivent, vous devez au préalable télécharger ces données :

- ftp://ftp.chg.ucsb.edu/pub/org/chg/products/CHIRPS-2.0/global_daily/netcdf/p25/chirps-v2.0.2018.days_p25.nc
- https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_4.03/cruts.1905011326.v4.03/tmp/cru_ts4.03.1981.1990.tmp.dat.nc.gz

```
# telecharger avec R et decompresser
url1 <- "ftp://ftp.chg.ucsb.edu/pub/org/chg/products/CHIRPS-2.0/global_daily/netcdf/p25/chirps-v2.0.201
url2 <- "https://crudata.uea.ac.uk/cru/data/hrg/cru_ts_4.03/cruts.1905011326.v4.03/tmp/cru_ts4.03.1981.
library("curl")
library("R.utils")
dir.create("data")
curl_download(url1, "data/chirps-v2.0.2018.days_p25.nc")
curl_download(url2, "data/cru_ts4.03.1981.1990.tmp.dat.nc.gz")
gunzip("data/cru_ts4.03.1981.1990.tmp.dat.nc.gz")
```

Comment lire un fichier NetCDF avec le package `ncdf4`

La structure interne des fichiers NetCDF (.nc) change selon les variables et l'organisation spatiale des données.

Il faut toujours commencer par lire les métadonnées du fichier pour identifier :

1. le nom et la nature des variables contenues (ex : temperature, precipitation)
2. le nombre de dimensions (exemple : x, y, t)

Exemple 1 : CHIRPS

```r
library(ncdf4)
library(raster)
library(magrittr)
f_ncdf <- "data/chirps-v2.0.2018.days_p25.nc"
nc_chirps <- nc_open(f_ncdf)

print(nc_chirps)
```

```
## File data/chirps-v2.0.2018.days_p25.nc (NC_FORMAT_NETCDF4):
##
##      1 variables (excluding dimension variables):
##         float precip[longitude,latitude,time]   (Chunking: [240,67,61])  (Compression: level 5)
##             units: mm/day
##             standard_name: convective precipitation rate
##             long_name: Climate Hazards group InfraRed Precipitation with Stations
##             time_step: day
##             missing_value: -9999
##             _FillValue: -9999
##             geostatial_lat_min: -50
##             geostatial_lat_max: 50
##             geostatial_lon_min: -180
##             geostatial_lon_max: 180
##
##      3 dimensions:
##         longitude  Size:1440
##             units: degrees_east
##             standard_name: longitude
##             long_name: longitude
##             axis: X
##         latitude  Size:400
##             units: degrees_north
##             standard_name: latitude
##             long_name: latitude
##             axis: Y
##         time  Size:365
##             units: days since 1980-1-1 0:0:0
##             standard_name: time
##             calendar: gregorian
##             axis: T
##
##     15 global attributes:
##         Conventions: CF-1.6
##         title: CHIRPS Version 2.0
##         history: created by Climate Hazards Group
##         version: Version 2.0
##         date_created: 2019-01-30
##         creator_name: Pete Peterson
##         creator_email: pete@geog.ucsb.edu
##         institution: Climate Hazards Group.  University of California at Santa Barbara
##         documentation: http://pubs.usgs.gov/ds/832/
##         reference: Funk, C.C., Peterson, P.J., Landsfeld, M.F., Pedreros, D.H., Verdin, J.P., Rowlan
##         comments:  time variable denotes the first day of the given day.  Improved October 2015.
##         acknowledgements: The Climate Hazards Group InfraRed Precipitation with Stations development
##         ftp_url: ftp://chg-ftpout.geog.ucsb.edu/pub/org/chg/products/CHIRPS-latest/
```

```
##         website: http://chg.geog.ucsb.edu/data/chirps/index.html
##         faq: http://chg-wiki.geog.ucsb.edu/wiki/CHIRPS_FAQ
```

Variables

Le jeu de données CHIRPS contient 1 seule variable : **precip**. Les métadonnées pour cette variable donne des indications intéressantes. L'unité : mm/jour. La résolution temporelle (time_step) : journalière. La valeur numérique choisie pour désigner les valeurs manquantes : -9999. Et celle pour les pixels sans valeurs (hors de la zone des données) : -9999 aussi.

Ici, les métadonnées indiquent également la portée géographique : entre 50°S et 50°N.

Comment extraire une métadonnée en particulier ?

```
# exemple : extraire "missing value" pour la variable precip
nodata_attr <- ncatt_get(nc_chirps, "precip", "missing_value")
# la liste extraite présente 2 éléments :
# présence du paramètre (hasatt) et valeur (value)
nodata_value <- nodata_attr$value
nodata_value
```

```
## [1] -9999
```

Dimensions

Les valeurs sont agencées selon 3 axes X, Y, T ou pour les désigner autrement, **3 dimensions** nommée "longitude", "latitude" et "time".

Les longitudes, latitudes et jours forment 3 vecteurs avec respectivement 1440, 400, et 365 éléments. Logique, puisque la résolution spatiale est de 0.25 degrés et la résolution temporelle de 1 jour. Pour extraire ces 3 vecteurs, utilisez la fonction `ncvar_get`.

Infos à noter pour ce jeu de données : les longitudes sont indexées de l'ouest vers l'est, et les latitudes **du sud au nord** ; le vecteur issu de "time" est une série d'entiers avec le nb de jours depuis le 01/01/1980.

```
lon <- ncvar_get(nc_chirps, "longitude")
lat <- ncvar_get(nc_chirps, "latitude")
t <- ncvar_get(nc_chirps, "time")
head(lon)
```

```
## [1] -179.875 -179.625 -179.375 -179.125 -178.875 -178.625
```

```
head(lat)
```

```
## [1] -49.875 -49.625 -49.375 -49.125 -48.875 -48.625
```

```
head(t)
```

```
## [1] 13880 13881 13882 13883 13884 13885
```

```
# pour convertir le vecteur t en dates :
t_dates <- as.Date("1980-01-01") + t
head(t_dates)
```

```
## [1] "2018-01-01" "2018-01-02" "2018-01-03" "2018-01-04" "2018-01-05"
## [6] "2018-01-06"
```

Extraire toutes les données sous la forme d'une matrice

Pour 1 variable, il est possible d'extraire l'ensemble des données sous forme de matrice 3D. Attention cependant au risque de saturation de la mémoire, si le jeu de données est conséquent. Les données extraites

d'un NetCDF avec R peuvent occuper plus de volume en mémoire que dans le fichier. Ainsi dans notre exemple, les données stockées sous forme d'entiers dans le fichier (sur 2 octets) sont converties en numérique flottant (sur 8 octets) en mémoire par R.

```
# note : le fichier chirps-v2.0.2018.days_p25.nc pèse 78 Mo
m3d <- ncvar_get(nc_chirps, varid="precip")
print(object.size(m3d), unit="MB")
```

```
## 1604 Mb
```

Extraire un sous-ensemble des données

Si l'on souhaite extraire uniquement un secteur géographique ou une saison, il est judicieux d'utiliser la fonction `ncvar_get` avec une indexation sur 3 dimensions pour extraire les valeurs recherchées.

Par exemple, la valeur indexée par (1,1,1) sera celle du pixel situées à 179.875°W, 49.875°S, jour 1. La valeur indexée par (1440, 400, 365) sera celle du pixel situées à 179.875°E, 49.875°N, jour 365.

Pour utiliser l'indexation avec la fonction `ncvar_get`, il faut indiquer les indices de départ, et le nombre de valeur à extraire pour chaque axe.

```
# quels indices correspondent à Paris dans lat et lon ?
i_lat_paris <- which(lat==48.875)
i_lon_paris <- which(lon==2.375)
# extraire 365 jours de données pour 1 pixel
precip_paris_2018 <- ncvar_get(nc_chirps, varid="precip",
                               start=c(i_lon_paris, i_lat_paris, 1),
                               count=c(1, 1, 365))
head(precip_paris_2018)
```

```
## [1] 10.380525  8.205389  0.000000  8.626737  6.985046  0.000000
```

```
# total precipitations 2018
sum(precip_paris_2018)
```

```
## [1] 661.0904
```

Comment faire pour extraire les précipitations sur une surface correspondant à l'Espagne pour une date spécifiée (prenons le 1 avril 2018) ? C'est possible avec la fonction `ncvar_get`, il faut cependant se rappeler que pour ce jeu de données, les latitudes sont indexées du sud vers le nord. Par conséquent :

- indiquer à la fonction les indices du coin au sud-ouest comme départ
- indiquer le nombre de valeurs à extraire sur les axes X et Y
- en sortie, s'attendre à une matrice dans laquelle la ligne 1 sera la limite sud (et la dernière ligne, la limite nord)

```
# etendue espagne
sp_ext <- c(e=-9.3, w=4.4, s=35.9, n=43.8)
# quel est l'indice des longitudes entre -9.3 et 4.4 ?
i_lon <- which(lon >= sp_ext[1] & lon <= sp_ext[2])
i_lat <- which(lat >= sp_ext[3] & lat <= sp_ext[4])
i_time <- which(t_dates=="2018-04-01")
# taille lon et lat
n_lat <-  length(i_lat)
n_lon <- length(i_lon)
#
precip_esp_20180401 <- ncvar_get(nc_chirps, varid="precip",
                                 start=c(i_lon[1], i_lat[1], i_time),
                                 count=c(n_lon, n_lat, 1))
```

Générer un objet `RasterLayer` à partir de la matrice extraite

On verra par la suite qu'**il est facile** avec le package `raster` de **lire des données NetCDF directement** sous forme de `RasterLayer` ou `RasterBrick`.

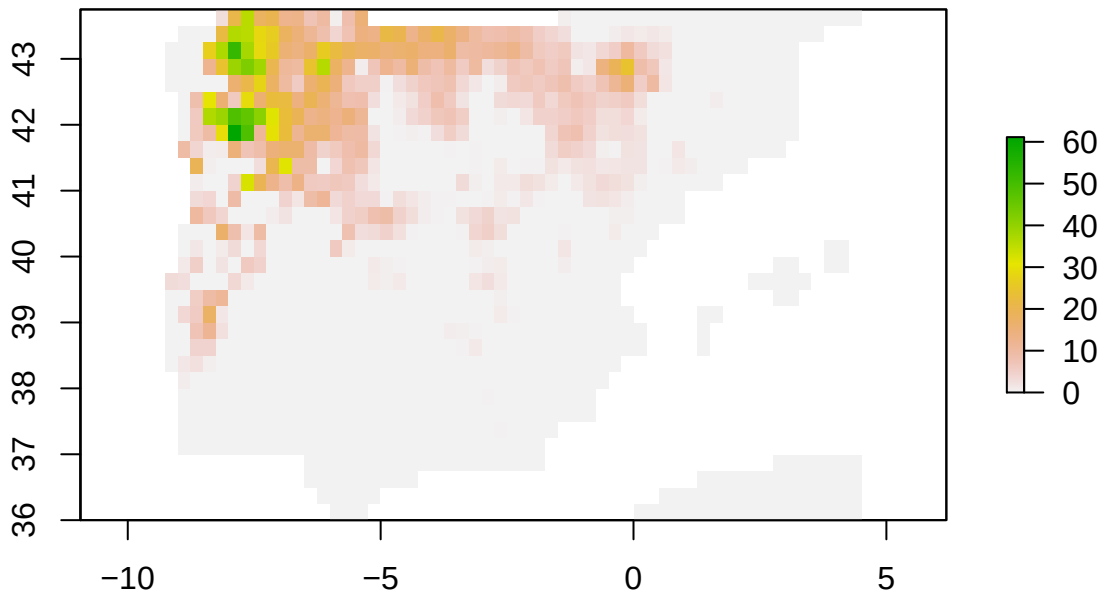
Mais à titre d'exemple, voici comment procéder pour créer un objet `RasterLayer` **à partir de la matrice** extraite précédemment :

1. Dans la matrice, les longitudes sont en lignes et les latitudes en colonnes. Il faut **chercher la transposée** de la matrice pour obtenir l'inverse.
2. Les 1ères lignes correspondent au sud et les dernières au nord. Il faut **inverser l'ordre des lignes** de manière à ce que le nord soit "en haut" de la matrice.
3. Alors, il est possible de passer la matrice en argument de la fonction `raster`. Il faut aussi donner les coordonnées des bords du raster et le système de coordonnées. Attention, les longitudes tirées du NetCDF correspondent au centre des pixels, ce qui est différent des bords (xmin, xmax, ymin, ymax) ; la dimension d'1 pixel est 0.25°, donc la distance du centre au bord est 0.125°.

```
library(raster)
dim(precip_esp_20180401)
```

```
## [1] 55 31
```

```
# transposer
precip_esp_20180401 <- t(precip_esp_20180401)
# inverser ordre des lignes
precip_esp_20180401 <- apply(precip_esp_20180401, 2, rev)
# extraire vecteur lon et lat pour l'Espagne
sp_lon <- lon[i_lon]
sp_lat <- lat[i_lat]
# créer raster
hres <- 0.25 / 2
rast_esp_20180401 <- raster(precip_esp_20180401,
                            xmn=sp_lon[1]-hres, xmx=sp_lon[n_lon]+hres,
                            ymn=sp_lat[1]-hres, ymx=sp_lat[n_lat]+hres,
                            crs="+proj=longlat +datum=WGS84")
plot(rast_esp_20180401)
```




```
# pour sauvegarder le raster au format GeoTIFF
writeRaster(rast_esp_20180401, "data/esp_20180401.tif", overwrite=T)
```

Fermer le NetCDF

Avec le package `ncdf4`, fermer le NetCDF proprement est obligatoire en cas d'écriture des données dans le fichier. C'est de toute façon une bonne habitude à prendre.

```
nc_close(nc_chirps)
```

Comment lire un fichier NetCDF directement avec le package `raster`

Exemple 1 : CHIRPS

Même exemple que dans la 1ère partie : comment faire pour extraire les précipitations sur une surface correspondant à l'Espagne pour une date spécifiée (prenons le 1 avril 2018) ?

Parce qu'on a précédemment inspecté les métadonnées du fichier, on sait qu'il contient une variable : *precip*, avec 3 dimensions : *longitude*, *latitude* et *time*.

Avec le package `raster` on peut ouvrir directement les données sous forme de `RasterBrick` puis extraire le jour et l'étendue qui nous intéresse.

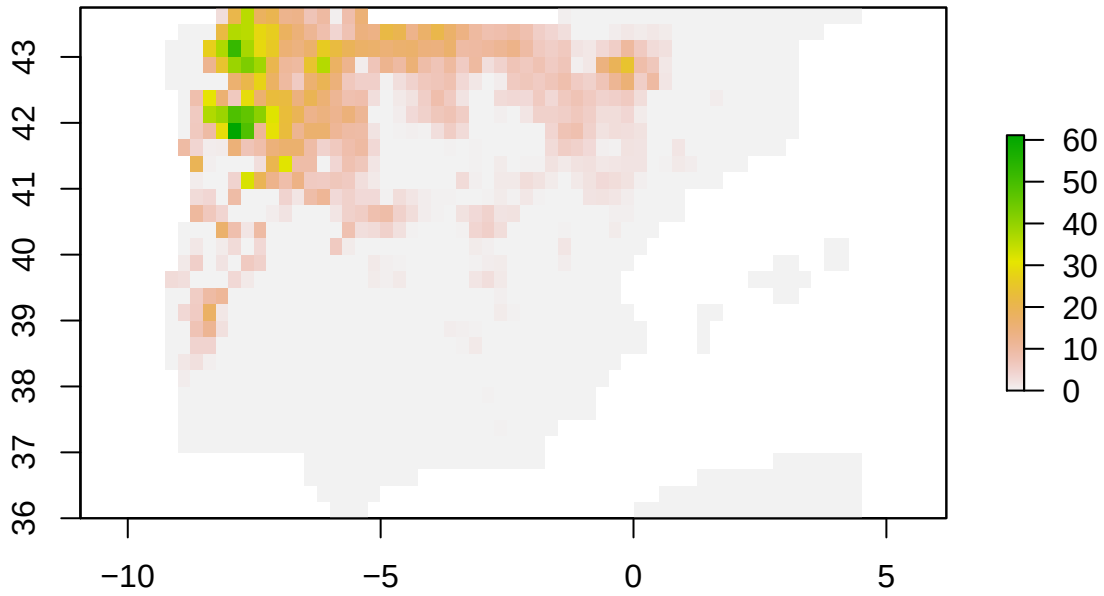
```
library(raster)
rast_chirps <- brick(f_ncdf, "precip")
# remarques :
# - comme le fichier comporte 1 seule variable, on aurait pu omettre son nom
# - les données ne sont pas encore chargées en mémoire
inMemory(rast_chirps)
```

```
## [1] FALSE
```

```
# index de la date correspondant au 01/04/2018
t_dates <- getZ(rast_chirps)
(i <- which(t_dates == "2018-04-01"))
```

```
## [1] 91
```

```
rast_0401 <- raster(rast_chirps, layer=i)
# etendue espagne > crop
sp_ext <- c(e=-9.3, w=4.4, s=35.9, n=43.8)
rast_esp_0401 <- crop(rast_0401, extent(sp_ext))
plot(rast_esp_0401)
```



Exemple 2 : CRU TS 4.03

Voyons un autre exemple avec le fichier NetCDF issu du site de CRU TS 4.03. Il s'agit des températures moyennes mensuelles pour les années 1981 à 1990, à une résolution de 0.5°.

Le but est cette fois-ci d'extraire les températures sur l'Amérique du Sud pour les mois de décembre uniquement.

```
f_ncdf2 <- "data/cru_ts4.03.1981.1990.tmp.dat.nc"
nc_cruts <- nc_open(f_ncdf2)
print(nc_cruts)
```

```
## File data/cru_ts4.03.1981.1990.tmp.dat.nc (NC_FORMAT_CLASSIC):
```

```
##
##      2 variables (excluding dimension variables):
##         float tmp[lon,lat,time]
##             long_name: near-surface temperature
##             units: degrees Celsius
##             correlation_decay_distance: 1200
##             _FillValue: 9.96920996838687e+36
##             missing_value: 9.96920996838687e+36
##         int stn[lon,lat,time]
##             description: number of stations contributing to each datum
##             _FillValue: -999
##             missing_value: -999
##
##      3 dimensions:
##         lon  Size:720
##             long_name: longitude
##             units: degrees_east
##         lat  Size:360
##             long_name: latitude
##             units: degrees_north
##         time  Size:120   *** is unlimited ***
##             long_name: time
##             units: days since 1900-1-1
##             calendar: gregorian
##
##     8 global attributes:
##         Conventions: CF-1.4
##         title: CRU TS4.03 Mean Temperature
##         institution: Data held at British Atmospheric Data Centre, RAL, UK.
##         source: Run ID = 1905011326. Data generated from:tmp.1905011321.dtb
##         history: Wed  1 May 2019 15:42:51 BST : User ianharris : Program makegridsauto.for called by
##         references: Information on the data is available at http://badc.nerc.ac.uk/data/cru/
##         comment: Access to these data is available to any registered CEDA user.
##         contact: support@ceda.ac.uk
```

Cette fois le fichier comporte 2 variables : les températures moyennes mensuelles (*tmp*) et le nombre de stations météo (*stn*) et toujours 3 dimensions : *lon*, *lat* et *time*.

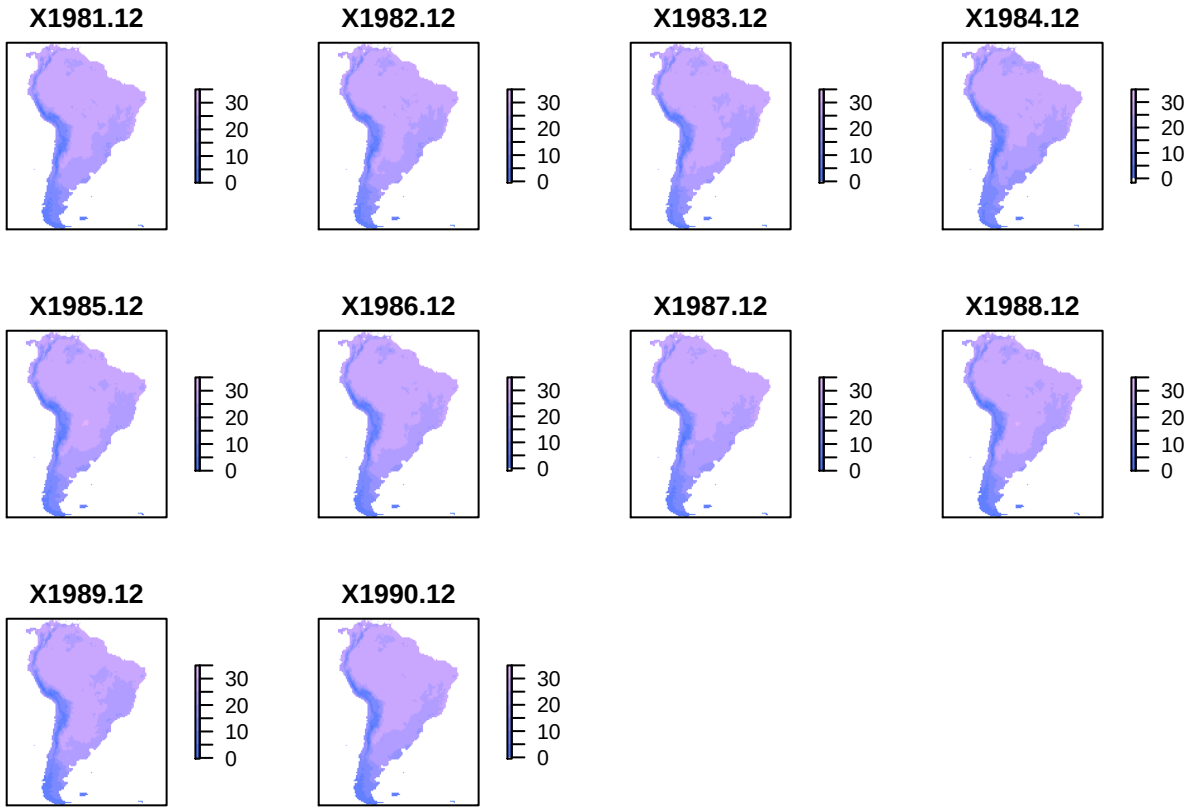
Nous cherchons l'indice des mois de décembre pour chaque année. Nous extrayons la variable **'tmp'** pour les mois recherchés sous la forme d'une liste de rasters. Nous créeons un objet `RasterBrick` à partir de cette liste.

```
# lire variable tmp
brick_cruts <- brick(f_ncdf2, varname="tmp")
# indices mois de décembre
t_months <- format(getZ(brick_cruts), "%Y-%m") # dates au format YYYY-MM
# facultatifs - changer les noms
names(brick_cruts) <- t_months
y_m <- paste0(seq(1981, 1990), "-12") # de 1981-12 à 1990-12
(i_time <- match(y_m, t_months))
```

```
##  [1]  12  24  36  48  60  72  84  96 108 120
```

```
# etendue Amerique Sud
sam_ext <- extent(c(xmin=-82, xmax=-34, ymin=-56, ymax=13))
l_rast <- lapply(i_time, function(i, r, e) {raster(r,layer=i) %>% crop(e)},
```

```
                  brick_cruts, sam_ext)
# une brick à partir de la liste des rasters
tmp_sam_dec <- brick(l_rast)
plot(tmp_sam_dec, axes=F,
     breaks=seq(0,35,5), col=colorRampPalette(c("royalblue1", "plum1"))( 8 ))
```




```
# fermeture nc_cruts (ncdf4)
nc_close(nc_cruts)
```

Liens utiles

- https://www.r-bloggers.com/a-netcdf-4-in-r-cheatsheet/
- https://disc.gsfc.nasa.gov/information/howto?title=How%20to%20Read%20Data%20in%20netCDF%20Format%20with%20R
- https://lpdaac.usgs.gov/resources/e-learning/working-appeears-netcdf-4-output-data-r/