Netcdf: combining spatial and temporal data with metadata

Dasapta Erwin Irawan, R. Willem Vervoort & Gene Melzack

22 January 2018

Storing and sharing spatio-temporal data

- Spatial and temporal data are challenging to store
- It is 2 or 3 dimensional and can even be 4 dimensional (for example groundwater data)
- We could use complex spreadsheets to share data, or build interconnected text files (by site, by time)
  - This is not necessarily easy to share, you would need an additional read me file to describe the data (i.e. it is not self-describing)
  - Is not necessarily system independent (proprietry software)
- Solution: NetCDF formats

NetCDF: history and principles

- NetCDF is the most widely used file format in climate and global studies
  - Almost all Global climate change model data is in NetCDF
  - Specifically good for multi-dimensional arrays
- Key strenghts are that NetCDF files are *self-describing* and *machine independent*
- Libraries and protocols are maintained by Unidata
- The latest protocol is NetCDF4, but older versions NetCDF3 are still around
- The typical file extension used for NetCDF files is **.nc**
- We will use the package `ncdf4` in R to read and create NetCDF4 files

NetCDF: Self-describing

- Self-describing means that within the file the metadata are included
- Here is an example from ET data from NCI Thredds server ET data
- The next slide is output from the AWRA model

netcdf metadata AWRA model

```
library(ncdf4)

nc_example <- nc_open("C:/Users/rver4657/ownCloud/working/S
print(nc_example)
```

```
## File C:/Users/rver4657/ownCloud/working/SSEAC/opendataw
##
##      1 variables (excluding dimension variables):
##         float actual_evapotranspiration[lon,lat,time]
##             _FillValue: -999
##             long_name: Daily evapotranspiration, 50th pe
##             cell_methods: day
##             grid_mapping: crs
##             standard_name: actual_evapotranspiration
##             coordinates: time lat lon
##             units: mm day-1
##             _ChunkSizes: 1
##               ChunkSizes: 681
```

NetCDF: Self-describing

- The meta data and the description of the data are included in the file
- Each variable has to have information about units and a description

```
# Longitude
lon <- ncvar_get(nc_example,"lon")
nlon <- dim(lon)
head(lon)
```

```
## [1] 145.00 145.05 145.10 145.15 145.20 145.25
```

```
# units of time
tunits <- ncatt_get(nc_example,"time","units")
tunits$value
```

```
## [1] "seconds since 1970-01-01 00:00:00"
```

NetCDF: Machine independent

- Because the algorithms to write and read netcdf are written in C and maintained by unitdata it can be used with any operating system.
- It is open source, so you can recompile if needed
- However, most scripting languages have tools to extract and manage netcdf files.

An example: creating a netcdf file using R

- We will be using the palæo-channel dataset that we have introduced on day 1
- To review: this dataset consists of samples taken on a single date on 38 locations on a transect
- To start of, we will transfer the particle size data to a netcdf (see below)

```
PSAdata <- read_csv("OriginalDataFolder/Willem/Soilparticle
pander(PSAdata[1:5,1:5])
```

Lat	Long	Distance	Depth_top	Depth_bottom
149.7	-29.27	1	0	1
149.7	-29.27	1	1	2
149.7	-29.27	1	2	3
149.7	-29.27	1	3	4
149.7	-29.27	1	4	5

Defining the spatial and temporal dimensions

```
library(ncdf4)

# define dimensions
londim <- ncdim_def("lon","degrees_east",
                    as.double(unique(PSAdata$Long)))
latdim <- ncdim_def("lat","degrees_north",
                    as.double(unique(PSAdata$Lat)))
timedim <- ncdim_def("time","days sine 2001-01-21",
                     as.double(1))
depthdim <- ncdim_def("Depth","m",
                      as.double(unique(PSAdata$Depth_top)))
```

- We actually don't need the time dimension, but I am putting it in as an example

define variables

create netCDF file and put arrays

- Need to first put data into arrays