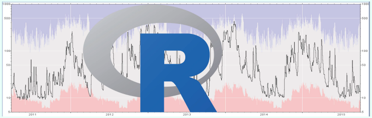
Extracting netCDF climate data for hydrological analyses

EGU short course
Using R in Hydrology

Louise Slater
l.slater@lboro.ac.uk
11 April 2018

Loughborough University



Aims

The aim of this presentation is to illustrate how to create a time series of basin-averaged daily precipitation by extracting data from a netCDF file.

The example focusses on the British Isles, because the same data will be used for streamflow modelling in the following presentations.

The same methods can however be used with other datasets.

Why use netCDF?

Network Common Data Form (netCDF) is:

- an interface for storing and accessing data in the form of arrays (n-dimensional rectangular structure)
- widely used for archiving and distributing climate data (e.g. NOAA, EUMETSAT, NASA, NCAR, COLA, CEH..)
- common to many different disciplines
- self-describing: it contains its own metadata (lat, lon, names, units, attributes), reducing the likelihood of errors
- accessible via a broad range of programs

See An Introduction to NetCDF for further info.

Why do this in R?

- R can read and write netCDF easily
- Many different packages available: ncdf, ncdf4, raster, RNetCDF
- Easy to automate the whole workflow
- Free and reproducible...

We will use the ncdf4 package by David Pierce

Before starting...

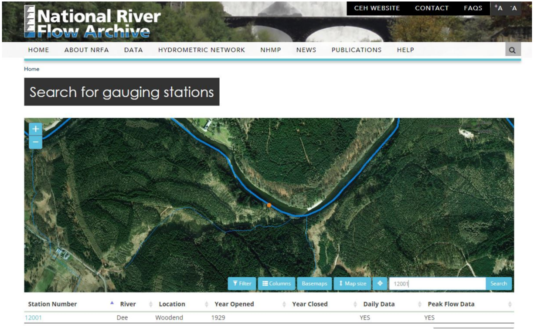
You will need to install the following packages if you wish to follow along:

```
install.packages("ncdf4") # to load the netCDF data
install.packages("lubridate") # to manipulate dates
install.packages("rgdal") # to load shapefiles
install.packages("raster") # to create rasters
install.packages("ggplot2") # to create maps/graphs
```

Load the packages:

```
library("ncdf4")
library("lubridate")
library("rgdal")
library("raster")
library("ggplot2")
```

Before starting...



We are using data from the River Dee at Woodend (NRFA gauging station number 12001). Sample gridded data for 2015-06 can be downloaded directly by clicking here and the catchment shapefile can be downloaded from here. All data are open-access from the Centre for Ecology and Hydrology (CEH), as referenced below.

CHESS data

We will use CHESS (Climate hydrology and ecology research support system) data.

These are 1 km gridded estimates of daily rainfall for the UK, for 1890-2015 (CEH-GEAR; Robinson et al. 2017).

Open and check the contents of your file:

```
nc <- nc_open("./data/chess_precip_201506.nc", auto_GMT = TRUE)
print(nc)
```

```
## File ./data/chess_precip_201506.nc (NC_FORMAT_NETCDF4):
##
##      4 variables (excluding dimension variables):
##         int crs[]   (Contiguous storage)
##             long_name: coordinate_reference_system
##             grid_mapping_name: transverse_mercator
##             semi_major_axis: 6377563.396
##             semi_minor_axis: 6356256.91
##             inverse_flattening: 299.3249646
##             latitude_of_projection_origin: 49
##             longitude_of_projection_origin: -2
##             false_easting: 4e+05
##             false_northing: -1e+05
##             scale_factor_at_projection_origin: 0.9996012717
##             EPSG_code: EPSG:27700
##         float lat[x,y]   (Contiguous storage)
##             _FillValue: -99999
##             long_name: latitude of grid box centre
##             standard_name: latitude
##             units: degrees_north
##         float lon[x,y]   (Contiguous storage)
##             _FillValue: -99999
##             long_name: longitude of grid box centre
##             standard_name: longitude
##             units: degrees_east
##         float precip[x,y,time]   (Contiguous storage)
##             _FillValue: -99999
##             standard_name: precipitation
##             units: kg m-2 s-1
##             long_name: CEH Gridded Estimates of Areal Rainfall
##             comment: The estimated rainfall amount per second (kg m-2 s-1) is equivalent to the rainfall de
##
##      3 dimensions:
##         y  Size:1057
##             _FillValue: -99999
##             units: m
```

Extract the lon, lat and time data

The ncdf description tells us the variables are: x (longitude - length 656), y (latitude - length 1057), and time (30 days):

```
lon <- ncvar_get(nc, "x")
lat <- ncvar_get(nc, "y")
time <- ncvar_get(nc, "time")
```

Check the dimensions of those variables:

```
nlon <- dim(lon)
nlat <- dim(lat)
ntime <- dim(time)
print(c(nlon, nlat, ntime))


## [1]  656 1057   30
```

Import the precipitation data

Import the actual data (units: kg m-2 s-1):

```
data <- ncvar_get(nc, "precip")
```

Check the dimensions:

```
c(nlon, nlat, ntime) == dim(data)

## [1] TRUE TRUE TRUE
```

Check the time units in the data

```
timeunits <- ncatt_get(nc, "time", "units")
timeunits$value # the starting date

## [1] "days since 1961-01-01"
```

So the dates in the netCDF are just a list of numbers:

```
time[1:5]

## [1] 19874 19875 19876 19877 19878
```

Convert the dates to something meaningful

We need to convert the dates from a list of numbers to a list of dates (using the lubridate package):

```
startDate <- ymd("1961-01-01")
myDates <- startDate + days(time)
myDates[1:5]

## [1] "2015-06-01" "2015-06-02" "2015-06-03"
## [4] "2015-06-04" "2015-06-05"
```

Assemble the data

(1) Prepare the precip data

```
precipdata <- as.matrix(data, ncol = 1)
```

(2) Prepare the lon, lat, and time data

```
# Note the expand.grid() function creates a data frame that is the cartesian produc
lonlatdat <- expand.grid(lon, lat, myDates)
head(lonlatdat)

##   Var1 Var2       Var3
## 1  500  500 2015-06-01
## 2 1500  500 2015-06-01
## 3 2500  500 2015-06-01
## 4 3500  500 2015-06-01
## 5 4500  500 2015-06-01
## 6 5500  500 2015-06-01
```

(3) Bind the two

```
precip_df <- cbind(lonlatdat, precipdata)
```

(4) Give the data some names

```
names(precip_df) <- c("lon", "lat", "date", "precip")
head(precip_df)

##    lon lat       date precip
## 1  500 500 2015-06-01     NA
## 2 1500 500 2015-06-01     NA
## 3 2500 500 2015-06-01     NA
## 4 3500 500 2015-06-01     NA
## 5 4500 500 2015-06-01     NA
## 6 5500 500 2015-06-01     NA
```

```
# Convert the units from kg m-2 s-1 to mm/day
precip_df$precip <- precip_df$precip * 86400

# Check the data
head(precip_df)

##    lon lat       date precip
## 1  500 500 2015-06-01     NA
## 2 1500 500 2015-06-01     NA
## 3 2500 500 2015-06-01     NA
## 4 3500 500 2015-06-01     NA
## 5 4500 500 2015-06-01     NA
## 6 5500 500 2015-06-01     NA
```

Choose one day

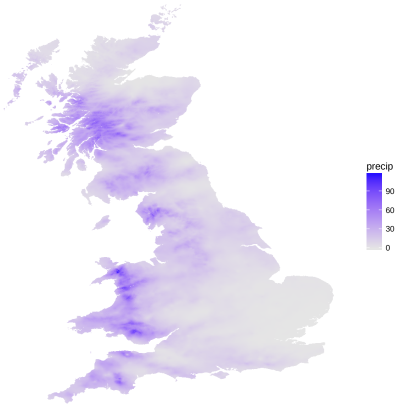
Subset the data, to make a basic visualisation:

```
df_oneday <- precip_df[precip_df$date == ymd("2015-06-01"), ]
```

Map the data

Very basic plot with ggplot:

```
ggplot(data=df_oneday)+  theme_void()+
   geom_raster(aes(x=lon, y=lat, fill=precip))+
   scale_fill_continuous(low="grey90", high="blue", na.value="white")
```



Focus on one catchment

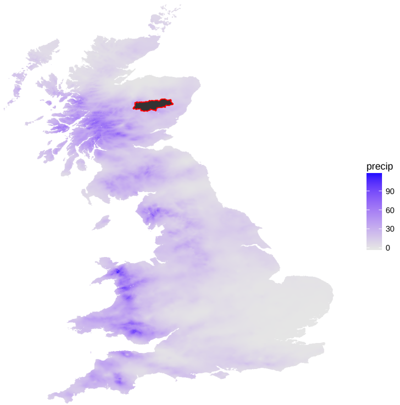
Load one of the National River Flow Archive's catchment shapefiles using the rgdal package (note: the shapefile that we use below can be downloaded here)

```
Basinname <- "12001"
Basin <- readOGR("./data", Basinname)

## OGR data source with driver: ESRI Shapefile
## Source: "C:\...
## with 1 features
## It has 4 fields
```

```
plot(Basin)
```

Check where the catchment is located

```
ggplot(data=df_oneday)+  theme_void()+
   geom_raster(aes(x=lon, y=lat, fill=precip))+
   scale_fill_continuous(low="grey90", high="blue", na.value="white")+
   geom_polygon(data = Basin, aes(x = long, y = lat), col = "red")
```



Extract daily precipitation

We wish to extract data from the precipitation raster, using the catchment outline as a cookie cutter

Convert the precipitation data to raster format

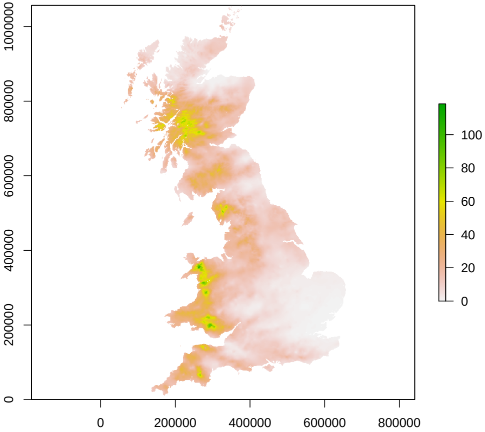
```
# remote the date (to keep just long, lat, and the values)
x <- subset(df_oneday, select = -date)

# promote to SpatialPointsDataFrame
coordinates(x) <- ~ lon + lat
```

```
# check the projection (from the netCFD file)
# ncatt_get(nc,"crs")$EPSG_code # this tells us the CRS is epsg:27700 (type ncatt_g

# assign the correct projection (British National Grid)
proj4string(x)<- CRS("+init=epsg:27700")
```

```
# promote to SpatialPixelsDataFrame
gridded(x) <- TRUE

# create a raster
precip_raster <- raster(x)
```

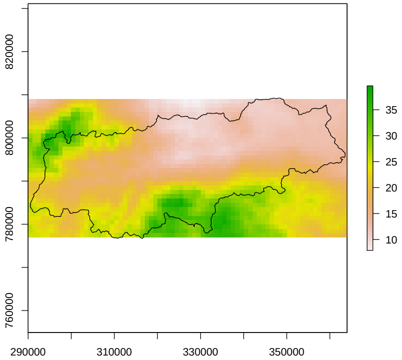
Check the raster and extract the basin-averaged value

```r
plot(precip_raster)
```



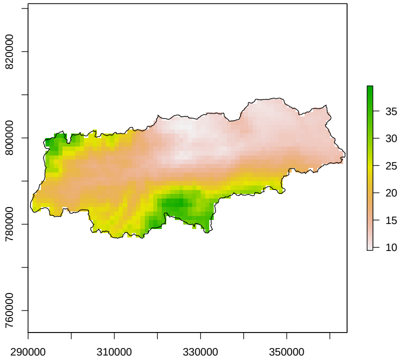
Crop the raster using the catchment outline

```
croppedraster <- crop(precip_raster, Basin)
plot(croppedraster)
plot(Basin, add = T)
```



Tailor the data to the catchment outline

```
prismraster <- mask(croppedraster, Basin)
plot(prismraster)
plot(Basin,add=T)
```



Extract the data values for that basin

Method 1- extract all data points:

```
allpoints <- rasterToPoints(prismraster)
df <- data.frame(allpoints)
head(df)

##        x      y   precip
## 1 343500 808500 10.39645
## 2 344500 808500 10.77874
## 3 345500 808500 11.13980
## 4 346500 808500 11.24395
## 5 347500 808500 11.60727
## 6 348500 808500 11.85691

mean(df$precip)

## [1] 18.56258
```

Extract the data values for that basin

Method 2- use the 'extract' function from the raster package to calculate the weighted mean:

```
raster::extract(precip_raster, Basin, weights = TRUE, fun = mean)

##          [,1]
## [1,] 18.54497

# weights = TRUE includes *any* partially covered cells
# weights = FALSE includes *only* those whose centroid is within the polygon

# note: I use 'raster::extract' so R knows it should use the raster package,
# because otherwise it can conflict with the tidyverse package.
```

Generate a time series

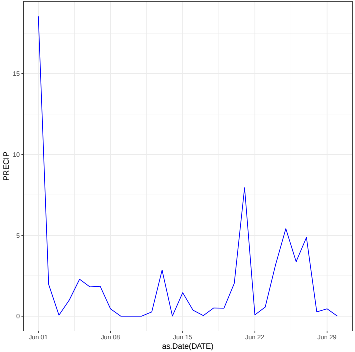
Note: there are different ways to automate the extraction for every day, to generate a time series. A loop is just one such way...

```
unique_days <- unique(precip_df$date)
ndays <- length(unique_days)
mylist <- list()

for (i in 1:ndays) {
  df_oneday2 <- precip_df[precip_df$date == ymd(unique_days[i]), ]
  x <- subset(df_oneday2, select = -date)
  coordinates(x) <- ~ lon + lat # SpatialPointsDataFrame
  proj4string(x) <- CRS("+init=epsg:27700") # Assign projection
  gridded(x) <- TRUE # promote to SpatialPixelsDataFrame
  x <- raster(x) # create raster
  mylist[[i]] <- data.frame(
    "DATE" = as.character(unique_days[i]),
    "PRECIP" = as.numeric(raster::extract(x, Basin, weights = TRUE, fun = mean)),
    stringsAsFactors = FALSE # this prevents the conversion to factors
  ) # output a dataframe with the date and the basin-averaged precip
  rm(df_oneday2, x) # remove the objects
}

df2 <- data.frame(do.call(rbind,mylist))
# sapply(df2,class)
df2$DATE <-as.Date(df2$DATE)
```

Plot the time series

One month of basin-averaged precipitation data:

```
ggplot(df2) + theme_bw() +
  geom_line(data = df2, aes(x = as.Date(DATE), y = PRECIP), col = "blue")
```



Thank you for listening!