Lab 05

Working with Tabular and Gridded Climate Data

2023-10-02

Table of contents

# 1. Intro

## 1.1 Setup

Remember to git clone to your machine, then activate and instantiate the environment, as we have done in previous labs.

using CSV  
using DataFrames  
using Dates  
using Plots  
using NCDatasets  
using Unitful

## 1.2 Resources

* [NCDatasets.jl](https://alexander-barth.github.io/NCDatasets.jl/stable/)
* [DataFramesMeta.jl](https://juliadata.github.io/DataFramesMeta.jl/stable/)

# 2. Tablular Climate Data

The provided data set contains daily precipitation data from three rain gauges in Houston, TX, from the [GHCN Daily](https://www.ncei.noaa.gov/products/land-based-station/global-historical-climatology-network-daily) dataset. First, let’s glimpse at the data:

first(CSV.read("data/rain\_gauge\_houston.csv", DataFrame), 5)

1. What are the columns?
2. What are the column names?
3. What does the date column look like?

Parse the date column into a Date type. Use the [documentation](https://docs.julialang.org/en/v1/stdlib/Dates/#Dates.DateFormat) to define a Date Format that matches the format of the dates in the data. You’ve seen this in lab 2!

# as usual, remember to add curly brackets so that this block can run  
date\_format = DateFormat(...)  
precip = CSV.read(fname, DataFrame; dateformat=date\_format)

Next, let’s add appropriate units. Explore the data documentation and multiply the precipitation column by the appropriate unit to add units to our data variable.

...

Next, let’s *reshape* our data so that each station is its own column. This sort of reshaping is common – see [the docs](https://dataframes.juliadata.org/stable/man/reshaping_and_pivoting/).

precip = unique(precip)  
precip2 = let  
 long\_df = stack(precip, :PRCP, [:STATION, :DATE], value\_name=:prcp)[!, Not(:variable)]  
 rename!(long\_df, :STATION => :station, :DATE => :date)  
 unstack(long\_df, :date, :station, :prcp)  
end  
first(precip2, 5)

1. Plot each station’s precipitation as a time series
2. Plot the mean precipitation at each station for each month
3. Plot a scatterplot for each of the three pairs of stations (A and B, B and C, A and C)
4. What do you learn from these plots?

# 3. Gridded Climate Data

We will work with the [CPC Global Unified Gauge-Based Analysis of Daily Precipitation](https://psl.noaa.gov/data/gridded/data.cpc.globalprecip.html). This is a gridded dataset on 0.5 degrees. We are working with data in the general vicinity of Houston.

ds = Dataset("data/cpc.nc")  
println(ds)

Dataset: data/cpc.nc  
Group: /  
  
Dimensions  
 lon = 10  
 lat = 10  
 time = 15341  
  
Variables  
 lon (10)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lon  
 Attributes:  
 \_FillValue = NaN  
 long\_name = Longitude  
 units = degrees\_east  
 axis = X  
 standard\_name = longitude  
 actual\_range = Float32[0.25, 359.75]  
 coordinate\_defines = center  
  
 lat (10)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lat  
 Attributes:  
 \_FillValue = NaN  
 actual\_range = Float32[89.75, -89.75]  
 long\_name = Latitude  
 units = degrees\_north  
 axis = Y  
 standard\_name = latitude  
 coordinate\_defines = center  
  
 time (15341)  
 Datatype: Union{Missing, DateTime} (Float64)  
 Dimensions: time  
 Attributes:  
 \_FillValue = NaN  
 long\_name = Time  
 axis = T  
 standard\_name = time  
 coordinate\_defines = start  
 actual\_range = [692496.0, 701232.0]  
 delta\_t = 0000-00-01 00:00:00  
 avg\_period = 0000-00-01 00:00:00  
 \_ChunkSizes = 1  
 units = hours since 1900-01-01  
 calendar = proleptic\_gregorian  
  
 precip (10 × 10 × 15341)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lon × lat × time  
 Attributes:  
 \_FillValue = -9.96921e36  
 var\_desc = Precipitation  
 level\_desc = Surface  
 statistic = Total  
 parent\_stat = Other  
 long\_name = Daily total of precipitation  
 cell\_methods = time: sum  
 avg\_period = 0000-00-01 00:00:00  
 actual\_range = Float32[0.0, 428.02423]  
 units = mm  
 valid\_range = Float32[0.0, 1000.0]  
 dataset = CPC Global Precipitation  
 \_ChunkSizes = Int32[1, 360, 720]  
 missing\_value = -9.96921e36

The data is stored as a netcdf4 file, which is a common data format for climate data. We can glimpse the preciptation data as follows:

ds[:precip]

precip (10 × 10 × 15341)  
 Datatype: Union{Missing, Float32} (Float32)  
 Dimensions: lon × lat × time  
 Attributes:  
 \_FillValue = -9.96921e36  
 var\_desc = Precipitation  
 level\_desc = Surface  
 statistic = Total  
 parent\_stat = Other  
 long\_name = Daily total of precipitation  
 cell\_methods = time: sum  
 avg\_period = 0000-00-01 00:00:00  
 actual\_range = Float32[0.0, 428.02423]  
 units = mm  
 valid\_range = Float32[0.0, 1000.0]  
 dataset = CPC Global Precipitation  
 \_ChunkSizes = Int32[1, 360, 720]  
 missing\_value = -9.96921e36

1. What are its units?
2. What are its dimensions?

You can access the attributes (i.e., metadtata) of the variable as

ds[:precip].attrib

\_FillValue = -9.96921e36  
var\_desc = Precipitation  
level\_desc = Surface  
statistic = Total  
parent\_stat = Other  
long\_name = Daily total of precipitation  
cell\_methods = time: sum  
avg\_period = 0000-00-01 00:00:00  
actual\_range = Float32[0.0, 428.02423]  
units = mm  
valid\_range = Float32[0.0, 1000.0]  
dataset = CPC Global Precipitation  
\_ChunkSizes = Int32[1, 360, 720]  
missing\_value = -9.96921e36

Now, consult the [quickstart tutorial](https://alexander-barth.github.io/NCDatasets.jl/stable/#Explore-the-content-of-a-netCDF-file), specifically the “Load a netCDF file” section, to learn how to load the data into a Julia variable.

1. Create a Julia variable called lon to hold the longitude, one called lat to hold the latitude, one called time to hold the time variable, and one called precip\_grid to hold the precipitation data.
2. The time variable is stored as a DateTime type, but we want to convert it to a Date type. Use the Dates package to do this.
3. Add the correct units to precip\_grid by multiplying it by the appropriate unit from the Unitful package

Now, let’s create some simple plots. A challenge we will face is that the latitude data in our dataset goes in the reverse direction. We can fix this by reversing the latitudes

lat = reverse(lat)  
precip\_grid = reverse(precip\_grid, dims=2)

Now we can create some plots. For now we will neglect plotting coastlines, geographical projections, etc.

heatmap(lon, lat, precip\_grid[:, :, 1]')

Note that we need to transpose the data here. There’s not a good rule of thumb that I have any luck with for remembering when I need to do this, and you need to look at your data. In this case, the blank squares correspond to ocean and I know that the Gulf of Mexico is in the South East of our domain.

Create some plots for other days in the dataset.

After we’re done reading the data, we close our link to the file.

close(ds)

closed Dataset

# 4. Comparison

Create two visualizations that compare the gridded data to the rain gauge data.

1. Create a scatter plot of the precipitation at a single gauge with the precipitation at the nearest grid point. You will need to merge the two datasets together to do this – one way is to create a DataFrame with the time column of the gridded data and a single subset of the precip\_grid data. Then you can [join](https://dataframes.juliadata.org/stable/man/joins/) that to your rain gauge data.
2. Compare the seasonality (monthly averages) of your gridded and gauge data. You may take the average of all the gauge data (watch out for missing values) or you may choose a single gauge to work with.

You may interpret these instructions as you see most appropriate. Explain your thinking and your results, and what you have learned from this comparison. Does the gridded product better represent the gauges on short or long time scales?