Mozhgan A. Farahani

University of Colorado Denver

pyTIER: Development of a Python knowledge-based weather station interpolation algorithm

Mentors: Andrew Newman, Seth McGinnis and Rachel McCrary

July 27, 2022

NCAR | RESEARCH APPLICATIONS LABORATORY

This material is based upon work supported by the National Center for Atmospheric Research, which is a major facility sponsored by the National Science Foundation under Cooperative Agreement No. 1852977.

Outlines

- TIER Model Concept
- Motivation: Why Conversion to Python?
- Strategies for Converting Matlab to Python
- Challenges and Lesson Learned
- Apply Information Theory to Model
- Conclusion and Future Work

Topographically InformEd Regression (TIER) Model

- Interpolations Model: Converting point-based observations from irregularly placed weather stations to values on a uniform high-resolution grid.
- Incorporate knowledge of atmospheric physics into relatively simple underlying statistical models
- Described in Daly et al., 1994, 2000, 2002, 2007, 2008

- Uses terrain attributes in a *regression framework* to distribute in situ observations of precipitation and temperature to a grid.
- Enables our understanding of complex atmospheric processes to be encoded into a statistical model in an easy-to-understand manner.
- This system consisting of a terrain pre-processing; station selection and weighting and post-processing.

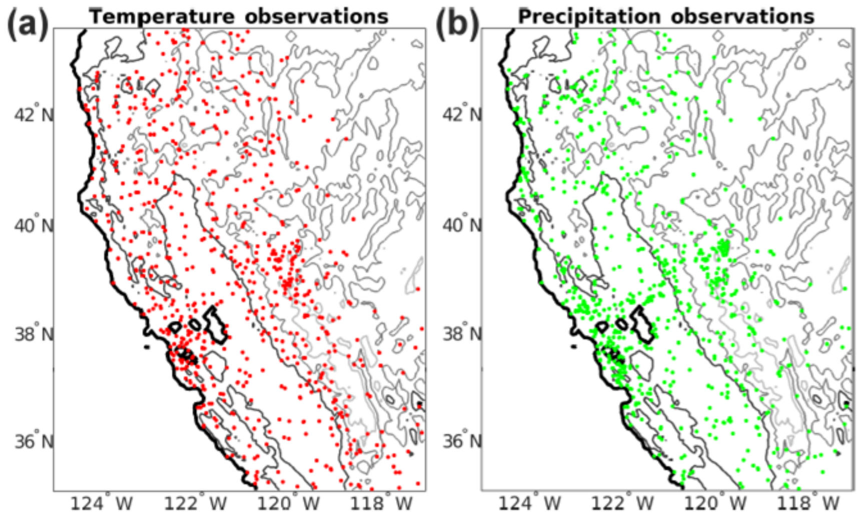


Figure 6. The TIER test domain with **(a)** the temperature station distribution and **(b)** the precipitation station distribution. Contours indicate the 0, 500, 1500, and 2500 m elevation contours moving from black to light gray.

https://doi.org/10.5194/gmd-13-1827-2020

TIER terrain pre-processing

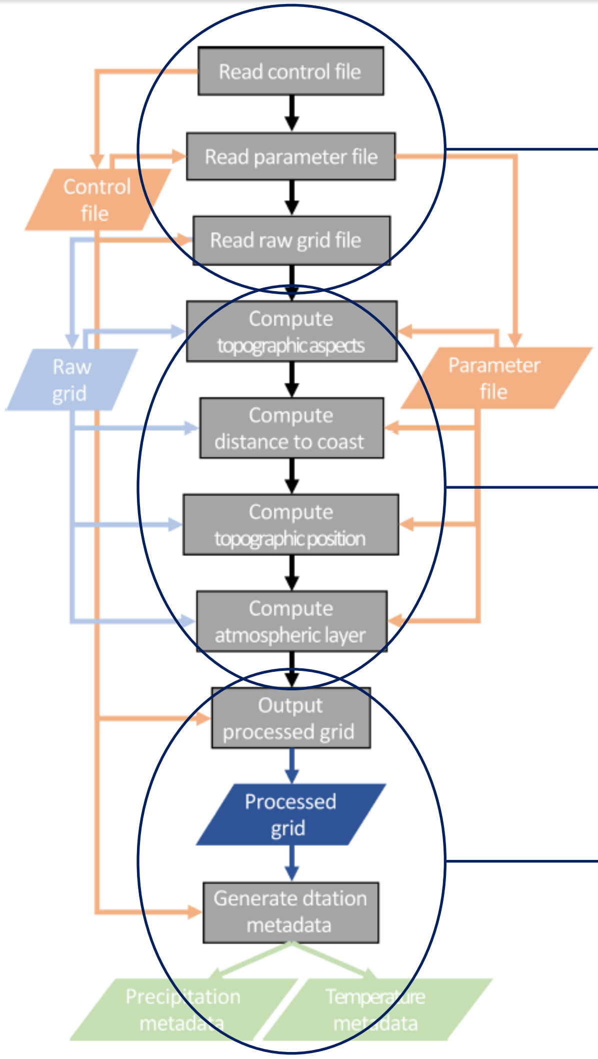


Figure 1. Flowchart describing the TIER pre-processing system. Processes are shaded gray, input files are orange, topographic inputs and outputs are shades of blue, and outputs are various shades of green.

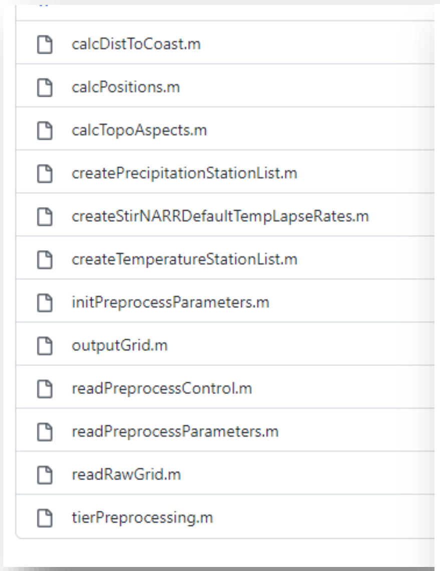
A parameter and control file specifies model parameters, and NetCDF input/output (IO) directories and files

Process the digital elevation model (DEM) into:

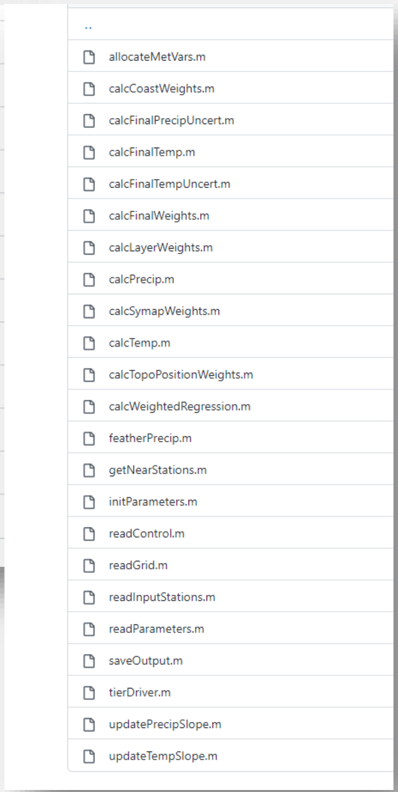
- Topographic facets
- Distance to the coast
- Topographic position
- Two-layer atmosphere

After the input domain grid file has been processed, the preprocessing routine generates station metadata files for all precipitation and temperature stations that will be used in the regression model.

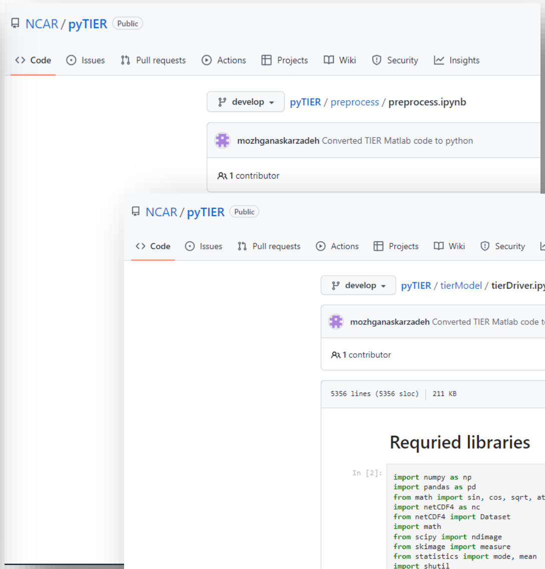
pyTIER GitHub repository



..
calcDistToCoast.m
calcPositions.m
calcTopoAspects.m
createPrecipitationStationList.m
createStirNARRDefaultTempLapseRates.m
createTemperatureStationList.m
initPreprocessParameters.m
outputGrid.m
readPreprocessControl.m
readPreprocessParameters.m
readRawGrid.m
tierPreprocessing.m



..
allocateMetVars.m
calcCoastWeights.m
calcFinalPrecipUncert.m
calcFinalTemp.m
calcFinalTempUncert.m
calcFinalWeights.m
calcLayerWeights.m
calcPrecip.m
calcSymapWeights.m
calcTemp.m
calcTopoPositionWeights.m
calcWeightedRegression.m
featherPrecip.m
getNearStations.m
initParameters.m
readControl.m
readGrid.m
readInputStations.m
readParameters.m
saveOutput.m
tierDriver.m
updatePrecipSlope.m
updateTempSlope.m



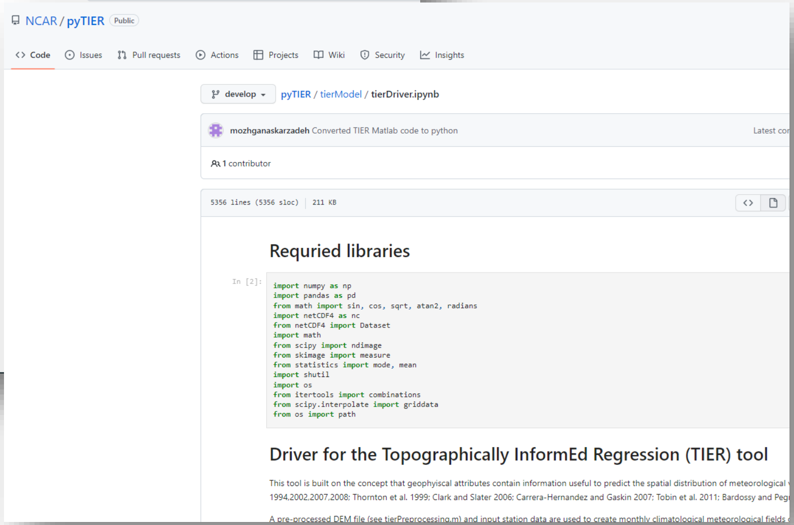
NCAR / pyTIER Public

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights

develop · pyTIER / preprocess / preprocess.ipynb

mozhganaskarzadeh Converted TIER Matlab code to python

1 contributor



NCAR / pyTIER Public

Code · Issues · Pull requests · Actions · Projects · Wiki · Security · Insights

develop · pyTIER / tierModel / tierDriver.ipynb

mozhganaskarzadeh Converted TIER Matlab code to python — Latest co

1 contributor

5356 lines (5356 sloc) | 211 KB

Requried libraries

```
In [2]:
import numpy as np
import pandas as pd
from math import sin, cos, sqrt, atan2, radians
import netCDF4 as nc
from netCDF4 import Dataset
import math
from scipy import ndimage
from skimage import measure
from statistics import mode, mean
import shutil
import os
from itertools import combinations
from scipy.interpolate import griddata
from os import path
```

Driver for the Topographically InformEd Regression (TIER) tool

This tool is built on the concept that geophyiscal attributes contain information useful to predict the spatial distribution of meteorological
1994,2002,2007,2008; Thornton et al. 1999; Clark and Slater 2006; Carrera-Hernandez and Gaskin 2007; Tobin et al. 2011; Bardossy and Peg

A pre-processed DEM file (see tierPreprocessing.m) and input station data are used to create monthly climatological meteorological fields

- **11 Preprocessing functions written in Matlab converted to one python script**
- **22 Main functions written in Matlab converted to one python script**

pyTIER GitHub repository:
https://github.com/NCAR/pyTIER/tree/develop

SCAN ME

Motivation: Why Conversion to Python?

MATLAB → python™

- **Technical:** General-purpose programming
- **Freedom:** Run your code without being locked-in with a given provider
- **Social:** Strong Python community

PYCON US 2022 PyData

Financial: Free and have access to many free open-source packages

LICENSE FEE

Strategies for Converting Matlab to Python

Either do it ***manually*** or take the help of some ***tools***

1. SMOP (Small Matlab and Octave to Python compiler)

```
git clone https://github.com/victorlei/smop
cd smop
python setup.py install --user
```

```
smop file.m
```

2. matlab2python

```
git clone https://github.com/ebranlard/matlab2python
cd matlab2python
pip install -r requirements.txt
```

```
python matlab2python.py file.m -o file.py
```

3. OMPC (Open-Source Matlab-To-Python Compiler)

```
wget https://www.bitbucket.org/juricap/ompc/get/tip.bz2
tar xvfj tip.bz2
cd ompc-2f62b3a16cd5
```

- ***Useful for short scripts***
- ***Find specific functions***

How to know which module I should use?
The SciPy stack

NumPy

Adds support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays

SciPy

A collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics, interpolation and much more.

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

Offers data structures and operations for manipulating numerical tables and time series

matplotlib

Plotting package and API for NumPy data, that provides publication-quality 2D plotting as well as rudimentary 3D plotting

Lesson Learned

MATLAB ⟹ python™

- In Matlab, several functions are loaded by default.
- Python has a different philosophy where few functions are loaded by default and most of them are separated into modules that you import into your code.
- MATLAB has the function of the matrix, and Python can use NumPy, and the library can achieve similar results

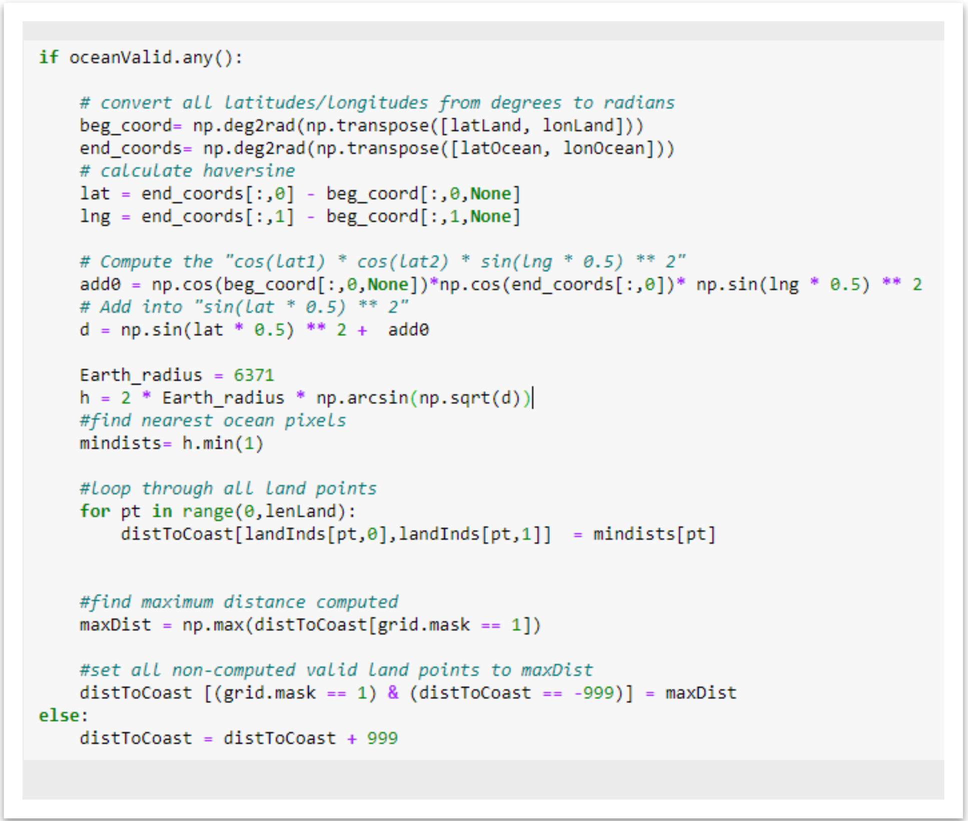
Matlab executes faster
This point might open for a great deal of debate, but my impression is that Matlab code *by default* executes faster than Python, simply because many Matlab functions are pre-compiled by default, while the user has to do this manually in Python.

Challenges: Solution for Distance function

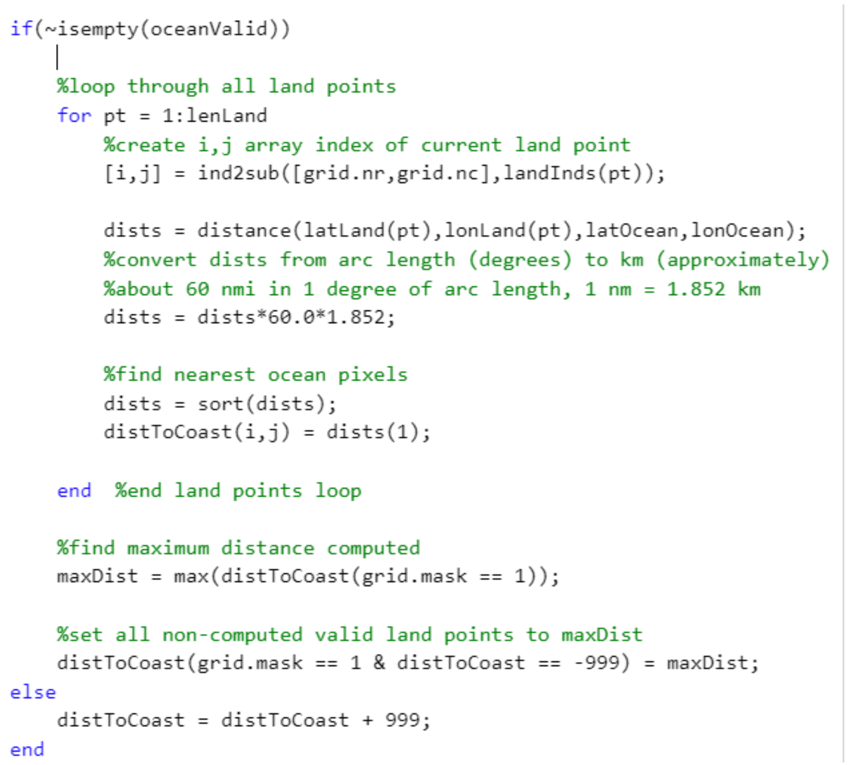
- Here's a vectorized method leveraging the NumPy ufuncs (haversine func) to replace math-module funcs
- we are enabling to operate on entire arrays in one go

Matlab distance Function:
Distance between points on sphere or ellipsoid

[arclen,az] = distance(lat1,lon1,lat2,lon2)



```python
if oceanValid.any():

    # convert all latitudes/longitudes from degrees to radians
    beg_coord= np.deg2rad(np.transpose([latLand, lonLand]))
    end_coords= np.deg2rad(np.transpose([latOcean, lonOcean]))
    # calculate haversine
    lat = end_coords[:,0] - beg_coord[:,0,None]
    lng = end_coords[:,1] - beg_coord[:,1,None]

    # Compute the "cos(lat1) * cos(lat2) * sin(lng * 0.5) ** 2"
    add0 = np.cos(beg_coord[:,0,None])*np.cos(end_coords[:,0])* np.sin(lng * 0.5) ** 2
    # Add into "sin(lat * 0.5) ** 2"
    d = np.sin(lat * 0.5) ** 2 +  add0

    Earth_radius = 6371
    h = 2 * Earth_radius * np.arcsin(np.sqrt(d))
    #find nearest ocean pixels
    mindists= h.min(1)

    #loop through all land points
    for pt in range(0,lenLand):
        distToCoast[landInds[pt,0],landInds[pt,1]]  = mindists[pt]


    #find maximum distance computed
    maxDist = np.max(distToCoast[grid.mask == 1])

    #set all non-computed valid land points to maxDist
    distToCoast [(grid.mask == 1) & (distToCoast == -999)] = maxDist
else:
    distToCoast = distToCoast + 999
```



```matlab
if(~isempty(oceanValid))

    %loop through all land points
    for pt = 1:lenLand
        %create i,j array index of current land point
        [i,j] = ind2sub([grid.nr,grid.nc],landInds(pt));

        dists = distance(latLand(pt),lonLand(pt),latOcean,lonOcean);
        %convert dists from arc length (degrees) to km (approximately)
        %about 60 nmi in 1 degree of arc length, 1 nm = 1.852 km
        dists = dists*60.0*1.852;

        %find nearest ocean pixels
        dists = sort(dists);
        distToCoast(i,j) = dists(1);

    end  %end land points loop

    %find maximum distance computed
    maxDist = max(distToCoast(grid.mask == 1));

    %set all non-computed valid land points to maxDist
    distToCoast(grid.mask == 1 & distToCoast == -999) = maxDist;
else
    distToCoast = distToCoast + 999;
end
```

Apply Information Theory to TIER

The information content of single variable is estimated using the concept of entropy (H) (Shannon 1948), described as:

$$H(X) = -\sum_{i=1}^{m} P(x_i) \log P(x_i)$$

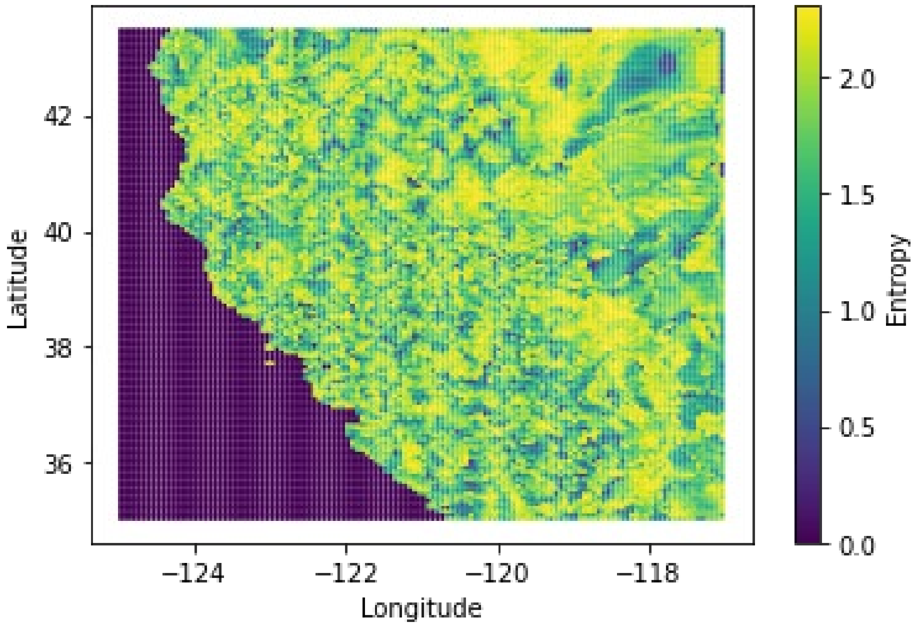
Entropy is a measure of the uncertainty associated with the occurrence of a certain event.

- ✓ How much information each station provides to each grid cell?
- ✓ How many stations do we need to consider?
- ✓ And quantify and prioritize the amount of information contained in a single station and shared among 2 or more stations?

Apply Information Theory to TIER

Quantify and prioritize the amount of information contained in a single station and shared among 2 or more stations?

- Start with NmaxNear = 16 nearest stations, which leads to some grid cells estimate of precipitation: P_16.
- Taking out the farthest stations 1 by 1 ,we get estimates of P_15, P_14, P_13, …, P_3, for each grid cell.
- This leads to some distribution of P values to calculate Entropy for a grid cell.
- A high entropy for a grid cell would indicate that the number of stations matters a lot, while a low entropy would mean that it doesn't matter too much.
- Take those "sensitive" regions and use those to choose some threshold of NmaxNear.



Conclusion and Future work

pyTIER

Developed a python version of the TIER weather station interpolation algorithm

- ✓ Enable users to better contribute to the TIER codebase
- ✓ Using a more popular, and free programming language
- ✓ make TIER more efficient for large spatiotemporal domain processing

- This project enables future developers to easily add parallelization for use on clusters and other high performance computing environments.
- Add additional interpolation methodologies in a user-friendly manner
- Potential applications of Information Theory to further improve our methodological understanding of these types of algorithms.

Thank you!

Questions?

Mozhgan.askarzadeh@ucdenver.edu

Thanks to Andrew Newman, Seth McGinnis, Rachel McCrary and SIParCS team!