2022

Water Table Depth Analysis for Well-Based Pumped Hydro

JOHN RICE

Problem Statement

I have been developing a new method for pumped-hydro energy storage. This new method involves using aquafers as a lower reservoir in a pumped-hydro setup connecting to traditional upper reservoirs via commercial agricultural wells. One of the very first steps in developing a new technology is determining where it can be used and by who. Since the rate of energy output of this battery system is directly related to the depth of the aquafer system, it is crucial to find areas that have suitable aquafer depths.

Abstract

It was found that the top of the water table becomes deeper the further southwest you go in the United States. Moreover, it was found that roughly one-fifth of the continental U.S. might be suitable for this energy storage method. The top five states to focus resources on were found to be Idaho, Nevada, Arizona, California, and Oregon. Meanwhile, a decent portion of the states along the east coast are generally unsuitable for this method but could be considered in special circumstances.

Introduction

To determine the best locations for a well-based pumped hydro system, an analysis of water tables across the United Stated must be performed. This process will consist of finding a dataset that accurately reports on thousands of wells across the country and is publicly available, cleaning the data to focus on aspects directly related to the topic, generating results that can easily be analyzed for trends and key areas of focus.

The Dataset

The dataset chosen for this study is provided by the United States Geological Survey. The dataset, “Groundwater Data for the Nation,” consists of both federally owned and operated wells specifically for the purpose of data collecting as well as wells owned privately outfitted with additional reporting systems. The dataset is kept up to date and includes a plethora of information ranging from water quality all the way to pressure. The specific metric that was used for this study was the latitude, longitude, and depth to water (depth to water table). The data is in the form of a txt file and does not have a traditional formatting technique as this is simply provided due to the obligation of public reporting and thus the dataset was not formatted in an easy-to-use way.

Data Clean-up

After the dataset was obtained it needed to be cleaned up so it could be manipulated, and insights gathered from it. To do this Python was used to open the txt file and then pull information such as latitude, longitude, and water level. The script can be seen below.

```
import csv

'''
  Author: John Rice
  Purpose: This script opens a file pullled from USGS Groundwater Data for the
Nation that does not have a traditional format. To comabt
           this each well listed in the file is interpreted and its relating data
is added as a row in a csv file that is outputted called
           depthMapDeepest.csv
'''

# Define header
header =  'Latitude'  'Longitude'  'Depth to water (ft)'  'lastLoc'

# Open file and read lines
with open "dv"  "r"  as file1
    lines = file1 readlines

    # Get number of entries
    lengthString = lines 15
    length = lengthString 25 29
    print 'Number of Entries:'  length
    entryCount = int length

    lastDepth = -500
    lastLoc = ''

    # Create output csv file and write header
    with open 'depthMapDeepest.csv'  'w+'  encoding='UTF8'  newline=''  as
csvFile
        writer = csv writer csvFile
        writer writerow header

        # Iterate through lines and extract data
        for line in lines
            if '2021-11-25' in line
                latDeg = float line 5 7
                latMin = float line 7 9
                latSec = float line 9 11
                latitude = latDeg +  latMin *  1/60   +  latSec *  1/3600
                location = line 5 18

                lonDeg = float line 11 14
                lonMin = float line 14 16
                lonSec = float line 16 18
```

```
longitude =  lonDeg +  lonMin *  1/60   +  lonSec *  1/3600 * -1

start = line find "2021-11-25   "  + len "2021-11-25    "
end = line find "   P"
substring = line start end

if substring not in  'Eqp'  '***'  'Dis'
    depth = float substring
    data =  latitude  longitude  depth   lastLoc + 'd'

    if location == lastLoc
        if lastDepth < depth
            lastDepth = depth
            writer writerow data
    else
        writer writerow data
        lastDepth = -500

    lastLoc = location
```

This Python script is designed to read data from a file that contains groundwater data, where each well is listed in a non-traditional format, and then output the relevant data as rows in a CSV file called depthMapDeepest.csv. The script opens the file and reads its contents, extracts the necessary data from each line of the file, and then writes that data to the output CSV file.

The first thing that the script does is import the csv module. This module provides functionality for reading and writing CSV files. Next, the author defines a header for the output CSV file, which includes four columns: Latitude, Longitude, Depth to water (ft), and lastLoc.

The script then opens the input file "dv" in read mode using the "with open" statement, and reads all the lines into a list called "lines". It also extracts the number of entries from the file by searching for a specific string in line 15 and converts it to an integer.

The script then creates a new output CSV file called depthMapDeepest.csv in write mode and writes the header row to it using the csv.writer() function.

Next, the script iterates through each line of the input file and extracts the necessary data from it. If the line contains the date '2021-11-25', the script extracts the latitude and longitude coordinates, calculates their decimal degree equivalents, and stores them in variables. The script then extracts the depth to water from the line and stores it in a variable called "depth".

If the location in the current line matches the location in the previous line (which is stored in a variable called "lastLoc"), the script checks if the depth in the current line is greater than the depth in the previous line. If it is, the script updates the value of "lastDepth" to the current depth

and writes the data to the output CSV file using the csv.writer() function. If the location in the current line does not match the location in the previous line, the script writes the data to the output CSV file and sets "lastDepth" back to -500.

Finally, the script updates the value of "lastLoc" to the current location and continues to the next line. When the script has finished iterating through all the lines in the input file, it closes the input and output files.

Overall, this script seems to accomplish the task of converting the non-traditional groundwater data format into a standard CSV format that can be more easily analyzed or visualized using other software tools. However, it is important to note that this code was written with a specific file format and date in mind, so it may need to be modified if used with different data sources or date ranges.

Data Manipulation

Now that the data is formatted nicely and all relevant information is easy to retrieve, manipulations can be performed on the table produced. Specifically, one important column that could be added is a column showing the state where the well is located. Since the position of the well is known in the form of latitude a longitude, geopy can be used to find the closest address to the well and from there a state can be determined. The following Python code shows how geopy was used to determine the address and how the state was assigned to each well.

```
import csv
from                import
from tqdm import tqdm

'''
Author: John Rice
Purpose: This script opens a csv file called data.csv that contains the coordinates
of wells throughout the United States
         along with the depth to water measured at each well. Then, using geopy, the
closest address to the coordinates is
         found. Finally the state is extracted from the returned address using
getState() and added to a new column called
         'Address'
'''

def getState address
    # Returns the state that an address is in
    allStates =  "Alabama"  "Alaska"  "Arizona"  "Arkansas"  "California"
"Colorado"  "Connecticut"  "Delaware"  "Florida"  "Georgia"  "Hawaii"  "Idaho"
"Illinois"  "Indiana"  "Iowa"  "Kansas"  "Kentucky"  "Louisiana"  "Maine"
"Maryland"  "Massachusetts"  "Michigan"  "Minnesota"  "Mississippi"  "Missouri"
"Montana"  "Nebraska"  "Nevada"  "New Hampshire"  "New Jersey"  "New Mexico"  "New
York"  "North Carolina"  "North Dakota"  "Ohio"  "Oklahoma"  "Oregon"
"Pennsylvania"  "Rhode Island"  "South Carolina"  "South Dakota"  "Tennessee"
"Texas"  "Utah"  "Vermont"  "Virginia"  "Washington"  "West Virginia"  "Wisconsin"
"Wyoming"
    for state in allStates
        if state in address
            return state
geolocator =           user_agent="coordinateconverter"

# open the input and output files
with open 'data.csv'  'r'  as infile  open 'output.csv'  'w'  newline=''  as outfile
    reader = csv reader infile
    writer = csv writer outfile

    # read in the header row and add the new column name
    header = next reader
    header append 'Address'
    writer writerow header

    # iterate over each row and the address of the well
    for row in tqdm reader  desc='Processing items'  unit='item'
        location = geolocator         row 0  + ',' + row 1
        value = ''
        try
            value = getState location
        except
            value = 'Other'
            pass
        row append value
        writer writerow row

    print 'File updated successfully!'
```

The above Python code aims to add a new column called 'Address' to an existing CSV file called 'data.csv' that contains the coordinates of wells throughout the United States, along with the depth to water measured at each well. The 'Address' column contains the state where the well is located.

The code uses the following Python libraries:

- csv: provides functionality for reading and writing CSV files.
- geopy: a Python client for several popular geocoding web services, including Nominatim, which is used in this code to find the closest address to the given coordinates.
- tqdm: a progress bar library for Python that enables the user to add a progress meter to their loops in a second.

The code defines a function called getState() which takes an address as input and returns the name of the state that the address is in. This function is used later in the code to extract the state name from the address returned by Nominatim.

The code opens the 'data.csv' file in read mode and the 'output.csv' file in write mode. It reads the header row of the input file and adds the 'Address' column to it. It then writes the updated header row to the output file.

The code then iterates over each row of the input file, finds the closest address to the given coordinates using Nominatim, extracts the state name from the address using the getState() function, and appends the state name to the current row. Finally, it writes the updated row to the output file.

The tqdm library is used to add a progress bar to the loop that iterates over each row of the input file. This provides visual feedback on the progress of the script.

If an exception occurs while extracting the state name from the address, the code sets the value of the 'Address' column to 'Other'. This ‘Other’ value is filtered out later when analyzing.

In conclusion, the code efficiently uses the geopy library to find the address corresponding to the given coordinates and extracts the state information from the returned address using the getState() function. The progress bar provided by tqdm is a useful addition to the script, which enhances the user experience by providing visual feedback on the progress of the script.

Tools Used:

- Python
- csv – Python library provides functionality for reading from and writing to CSV files
- geopy – Python client for several popular geocoding web services that provides geocoding and reverse geocoding functionality to locate addresses, cities, countries, and landmarks around the world
- tqdm - A progress bar library for Python that enables the user to add a progress meter to their loops
- Tableau - A data visualization software that allows users to connect, visualize, and share data in interactive and intuitive ways.

Results

Tableau was used to visualize the water tables of each state. It was discovered that water tables further west typically had a higher potential for being useful as a lower reservoir for a well-based pumped hydro solution. In an earlier study it was found that a depth over 100 feet is desired when powering the average U.S. single family house. A list of states that meet this desire can be found. Those states are Idaho, Nevada, Arizona, California, Oregon, Missouri, Texas, Arkansas, and New Mexico. In Figure 1.1 the trend in the water table can be seen. The water table tends to be deeper the further west you go and the further south you go.

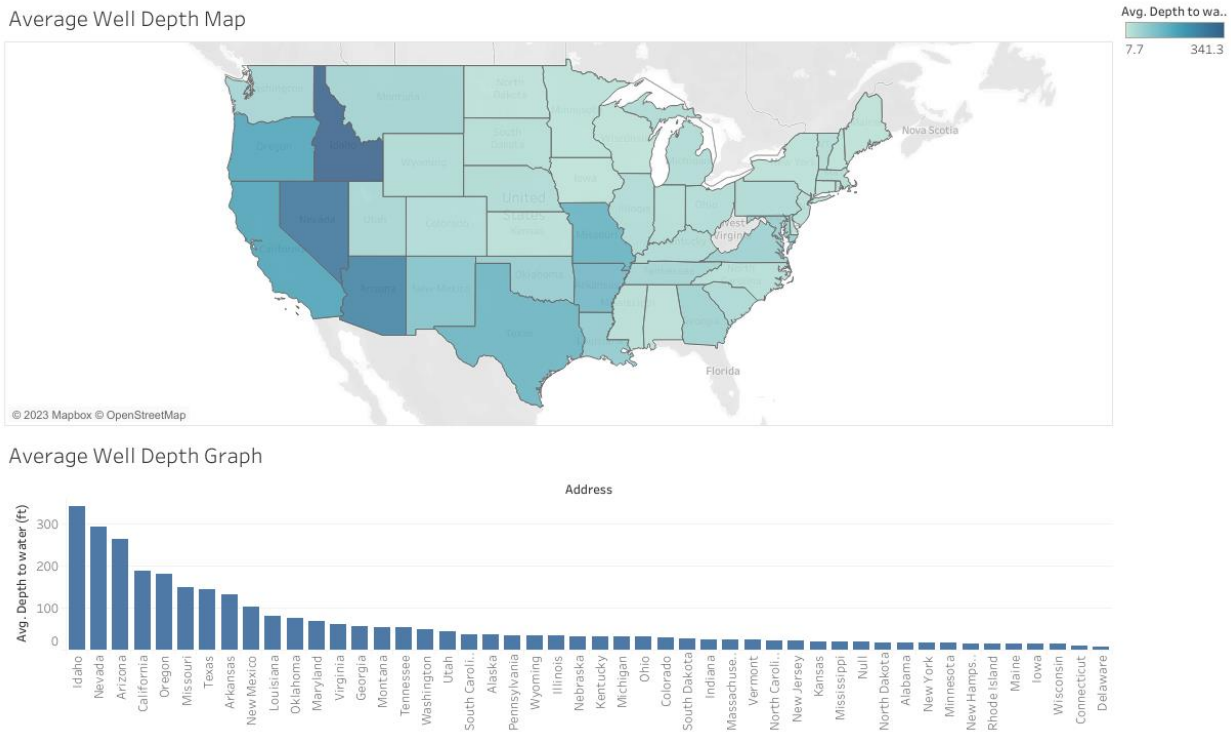

Figure 1: A geographic view of the water table in the United States of America.

Conclusion

In conclusion, given that roughly one-fifth of the U.S. meets the desired depth requirements for well-based pumped-hydro, there is significant reason to invest further research into the key states (Idaho, Nevada, Arizona, California, Oregon, and Texas) as well as the technology itself. It should be noted that a large amount of produce is produced in these areas and they may already be utilizing agricultural wells that could be retrofitted.