EPPS/GISC 4317/6317: Social and GI Science Programming Fundamentals

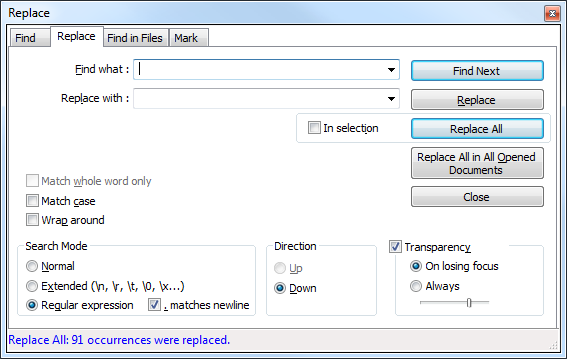
**Instructor: Dr. Bryan Chastain**

**Lab 11: Regular Expressions**

**Part 1 – Notepad++**

Regular expressions are powerful tools for searching and replacing, and they are not just limited to being used in Python. In this example we will use the free Notepad++ software to reformat latitude and longitudes in a textfile to be more easily read by GIS software.

1. Right-click on the worldcities.txt file in your Lab 11 directory and select “Edit with Notepad++”. (If you are working on your home computer, you can download Notepad++ for free from <http://notepad-plus-plus.org/>)
2. You will notice that the latitudes and longitudes are each broken up into two columns, one for degrees and one for minutes. Many GIS software processes require latitude and longitude to be in one column each. In this exercise we will be replacing all the two degree and minute columns with one column for latitude and one for longitude.
3. Go to Search>Replace to bring up the Replace window.



Notice at the bottom there is an option for searching/replacing by Regular expression. Check that option.

1. For the latitude, we know it’s generally in the format of <tab>Deg<tab>Min<space>(N or S). In regular expression, that would translate into something like: \t\d\t\d [NS]

Try typing that into the “Find what” field and pressing “Find Next”

1. It only returns a single result – the latitude for Kuala Lumpur, Malaysia. This is because Kuala Lumpur is the only city that has single digit degree and second values for latitude, which is what we specified when we just included one “\d”.

What would happen if we changed it to: \t\d\d\t\d\d [NS]? That would return only the double-digit minute and second values. We need a way of searching for both one-digit, two-digit, and three-digit values.

1. One possible solution is to include the optional character, “?”, which searches for 0 or 1 occurrence of the preceding character. Try searching for:

\t\d\d?\d?\t\d\d?\d? [NS]

1. Now it matches all latitudes in the file. Let’s modify it slightly to also include the longitudes:

\t\d\d?\d?\t\d\d?\d? [NS]\t\d\d?\d?\t\d\d?\d? [EW]

1. Now we are matching both the latitude and longitude fields. The next step is to replace them with our modified format. In order to retain the important bits when replacing we need to use back-referencing, which is done with parentheses. For example, searching for: \t(\d\d?\d?)\t\d\d?\d? [NS] and replacing with: \1 would replace all latitudes with just the degree portion of the longitude.

In our case, we are interested in the degree, minute and direction of both latitude and longitude, so we would need sets of parentheses:

\t(\d\d?\d?)\t(\d\d?\d?) ([NS])\t(\d\d?\d?)\t(\d\d?\d?) ([EW])

1. We would then refer to those numbers by \1, \2, \3, …\6. We would like the final coordinates to be in the format of 32°32’N<tab>94°45’W, so we should set our “Replace with” to be:

\t\1°\2'\3\t\4°\5'\6

1. Click “Replace All” – you should now have all latitudes and longitudes formatted into a single column for each.

**Part 2 – Regular Expressions in Python**

Regular expressions are a great way to add a lot more power and flexibility to your existing arcpy scripts. To incorporate Regular expressions into your Python scripts, you just need to include the “re” module.

This can come in handy when working with some CSV files, which have multiple data sections, each with a little header. Read these as plain text files and then parse each “minicsv” to treat that section as a file.

For example [The US DOJ Universal Crime Reporting Stats](http://www.ucrdatatool.gov/Search/Crime/State/RunCrimeStatebyState.cfm) does this. If you select more than one state you get data like (with some headers/footers removed):

Estimated crime in Alabama

Year,Population,Violent crime total

2010,4785401,18363

2011,4803689,20166

2012,4822023,21693

Estimated crime in Alaska

Year,Population,Violent crime total

2010,714146,4537

2011,723860,4416

2012,731449,4412

Estimated crime in Arizona

Year,Population,Violent crime total

2010,6413158,26528

2011,6467315,26789

2012,6553255,28108

Each state’s data is repeated in a separate little csv embedded in the larger file.

Eventually we want to create a state column to go along with the others:

State,Year,Population,Violent crime total

To do this we can split up the file and then read each of as though they were separate CSVs. After the split this ends up very similar to the approach we’d use if these were separate files in a directory.

1. Open Spyder (or Jupyter) and create a new script (or notebook) in your Lab 11 folder.
2. Download the CrimeStatebyState.csv file from eLearning and place it in your Lab 11 directory as well.
3. Enter the following code:

import re

import csv

import pandas as pd

# DOJ State Crime Estimates

csv\_file = 'CrimeStatebyState.csv'

# Variable for keeping track of current state in the file

state = ''

# Store the output (list of dictionaries, for importing into pandas)

data = []

with open(csv\_file) as f:

# Use csv module to read file

reader = csv.reader(f)

for line in reader:

# Skip blank lines

If len(line) > 0:

1. Next we need to split our code into two sections – one if we are reading in the state section header and the other for each line of actual data.:

# The section header only has one element (not comma-delimited),

# so only look at line[0], and check to see if the first 9

# characters are the word Estimated

if line[0][:9] == 'Estimated':

# Do regular expression stuff here

# Else this is the data we want to store

# However, we need to check to make sure the state has been set

# first so we don’t get any garbage at the top of the file, and

# we are not interested in the national totals, so we will skip

# all the stuff after that, and also skip the column headers.

elif state and state != 'United States-Total' and line[0] != 'Year':

state\_dict = {

'State': state,

'Year': line[0],

'Population': line[1],

'Violent crime total': line[2]

}

data.append(state\_dict)

1. Then for the regular expressions, we will search for the “Estimated crime in” text and use parentheses to grab the state name, closing with a “$” to denote the end of line:

matches = re.search('Estimated crime in (.\*)$', line[0])

state = matches.group(1)

1. Finally, just make the Data Frame and print the results (at the end, outside the loops):

df = pd.DataFrame(data)

print(df)

**HOMEWORK**

1. **Notepad++**

The universities.html file contains an ugly HTML document of DFW university information and addresses. Your task is to use regular expressions to clean up the file so that it is just: university name<tab>address. For example:

University of Texas at Dallas 2601 N Floyd Rd Richardson, TX 75080-1407

HINT: It may take multiple search & replace regular expressions to clean it up – document all of them. Try to do it in as few as possible.

Deliverable: modified file (.txt file format) and also write what regular expression(s) you used to get the result. Paste your expressions here.

**Graduate:**

**Do (a) as well as:**

1. **Python**

Using the original copy of worldcities.txt from Part 1, write Python code that uses regular expressions to convert those DMS coordinates into decimal degrees coordinates (reverse of midterm), and then map the points either using geopandas or Spatially-Enabled DataFrames.

Deliverable: Can either turn in .py file if written in Spyder, or can export Jupyter/ArcGIS Notebook to PDF and submit that.