NPS Flagging Tool

Documentation for the repository to automate database flags within the National Park Service.

# Introduction

This repository contains Python and Jupyter Notebook files for automating the extraction and data entry for database flags within the National Park Service.

# Overview

Each park has a different structure in which they estimate their visitation. This structure is logged in physical excel files as well as within the IRMA database. This creates issues where either source can differ greatly from the other. It is important to keep these changes synced across both sources in an efficient manner.

# Motivation

Parks often change this structure from time to time and it can be tedious and cumbersome to ensure the calculation map matches the database. This tool assumes the calculation maps as the most up to date structure and compares it with what is currently in the database. This means it only takes a few minutes to quickly update a park.

# Required Dependencies

You will need to have both Python and Visual Studio Code (VS Code) installed as well as access to the IRMA database to run SQL scripts. While there are many helper scripts located within the repository, the only files that need to be used are ‘verification.ipynb’, ‘FlagsStatusRetrevial.csv’, as well as ‘FlagsStatusRetrevial.sql’. These require access to the IRMA database and Sharepoint drive.

**IMPORTANT NOTES:** When installing software on DOI government computers it can be quite tricky. As software and policies change, the exact installation process may differ. Here are some notes to keep in mind when having IT install and download the software.

* It is best to install R Studio even if you will be using VS Code. R Studio plays nicer with R, however VS Code has better functionality.
* When downloading the VS Code installer, it is important to download **‘Visual Studio Code’ (Blue Icon)**, NOT **‘Visual Studio’ (Purple Icon)**. These are two different types of software, VS Code is an advanced text editor, whereas Visual Studio is an IDE for software development.
* When installing both R and VS Code, always check the box when regarding anything around ‘Add to PATH variable’. If you do not do this, the software will install but there will be no way to utilize it.

# Running The Tool

The process to use this tool is comprised of three steps: ***Installing The Packages, Updating The Data*** and ***Running The Script***.

## Installing The Packages

To get the correct script, load in VS Code and click ‘File > Open Folder…’ and navigate to where the repo is. Once opened you should see something like this on the left-hand side.

A screenshot of a computer

Description automatically generated

Double click on the ‘install\_packages.cmd’ file to open it, then click the play button on the top right to run the script. This will install each of the packages individually. If you have issues, you can always open the ‘Command Prompt’ application on your computer and copy/paste the code into the terminal and run it there.

## Updating The Data

To update the data, load Microsoft SQL Server Management and navigate to the VU Stats database. Load up the ‘FlagsStatusRetrevial.sql’ script and click ‘run’. Once the script has ran, save the results as ‘FlagsStatusRetrevial.csv’ in the same directory as the repository. It is important to ensure that headers are included when saving these results. There is a setting inside Microsoft SQL Server Management to turn on the headers on export.

## Running The Script

Once we have up to date data, we can run the script. First, start by opening the ‘verification.ipynb’ file and then entering the parks 4-character UnitCode. This will automatically go into the SharePoint file structure and find the latest calc map, as well as extract its data. Once the UnitCode is changed, clicking the play button will run that section of code.

A black background with text and numbers

Description automatically generated

The next block is to view the dataFrameChanged table. This table contains only the inputs that are different between the calculation map and IRMA csv file. This is useful if you only want to see the changes that need to be made. False values are colored in red, True values are colored in green. Specific cells that need to be changed are highlighted in white.

A black screen with white text

Description automatically generated

A screen shot of a black board with white text

Description automatically generated

The next code block loads the entire dataframe and shows all inputs in the database. This is useful if you need to figure out how far down to scroll between each cell that needs to be changed. This makes it very fast to quickly scroll and count which cells and rows need to be flipped from True to False or from False to True.

A black background with white text

Description automatically generated

A screenshot of a computer screen

Description automatically generated

Once all fields have been updated in IRMA, you can save changes and do this process all over again to ensure everything was updated correctly. It is not uncommon to miss a cell or row and this second go around will expose those errors to be fixed. If done correctly, the dataFrameChanged will be empty, and dataFrame will have 0 cells that are highlighted in white. It is important to note, that you need to re-export data from the database too see the changes affected in the tool.