Workshops for Ukraine

Introducing Power Bpy a python package for creating Power BI dashboards

May 22nd, 2025

Russell Shean

Outline

- What is this workshop series?
- Who am I?
- Building dashboards with Power BI
  - What is Power BI?
  - My old job
- A history of Power BI file structures
  - Implications of new file structures

Outline

- Demonstration: create a new power BI dashboard with python
- How it works
  - Theory and development process
  - Existing functions
    - Create new dashboard
    - Load data
      - TMDL
      - Local csv
      - csv on ADLS
    - Add new page
    - Add shape map

Outline

- Development setup (GitHub, documentation, PyPI, etc)
- How you can help!

Workshops for Ukraine

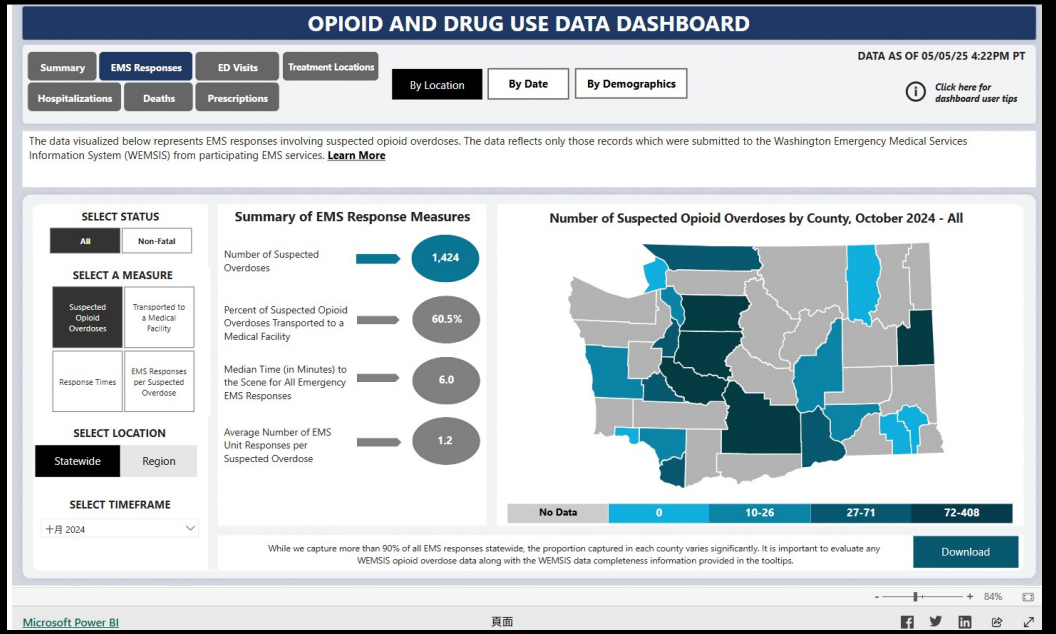
- What is this workshop series?
  - Series of technical workshops
- Why is it important?
  - Upskill!
  - Support Ukraine

About Me

- Freelance data scientist/data engineer
- Background in public health
  - Rhode Island Department of Health
  - Washington State Department of Health
- Taipei based

Dashboards at WADOH

- The visualization section builds a variety of dashboards
- Data cleaning and aggregation in R
- Dashboards in Power BI



My job at WADOH

- At start: Data processing pipelines in R
- Later: Automation, cloud transition, process improvement
  - R pipelines are already fairly automated and optimized
  - How can we speed up Power BI process too?

What is Power BI?

- Microsoft product for building dashboards and business intelligence
- Data processing, modeling and visualization
- Included with many organizations' Microsoft subscriptions
- Ties into other Microsoft products such as Fabric
- Low/no code tool

Power BI Advantages

- Bundled with existing Microsoft product subscriptions
- Supported publishing
- Existing style guide, trained staff and workflows
- Low/no code

Power BI Challenges

- Low/no code
- No version control
- Not reproducible
- Difficult to use data processing
- Difficult to automate

How to make Power BI dashboards

- Normal way: Power BI desktop
- Pause slides for live tour of Power BI desktop

Power BI file structure: A history

- First iteration: .pbix
  - A binary file
  - Similar to .zip file, multiple files inside
  - Possible to extract internal files and make changes
  - Difficult to put back together
    - Security bindings file
  - Pause slides: for live demo of extraction

Power BI file structure: A history

- Second iteration: .pbip
- Report and data specifications stored as plain text
- Mostly json
- Better, but not ideal for version control
- Pause slides for demonstration

Power BI file structure: A history

- Final iteration: .pbir
- Unnested json
- TMDL: new format for specifying data
  - Sort of yaml-like….(ish)
- Pause for demo

Power BI file structure: A history

- Data format: Tabular Model Definition Language (TMDL)
- Human-readable, version-controllable, separate files for different objects
  - https://learn.microsoft.com/en-us/analysis-services/tmdl/tmdl-overview?view=asallproducts-allversions
  - https://powerbi.microsoft.com/zh-tw/blog/tmdl-in-power-bi-desktop-developer-mode-preview/
- Pause for demo

What we have so far

- No python yet… sorry 😅
- Visual elements are defined in individual folders and files for each page and visual object
- Visual elements are defined with json
- Data has 2 parts
  - Column definitions
  - M code used for loading and processing
- Throughout: machine generated unique UUID
  - No name conflicts, but super not human readable

What does this mean?

- Thing 1: Text files can be edited directly without opening Power BI Desktop --- Great for quick fixes
- Thing 2: Changes to dashboards can easily be tracked in Git diffs
- Thing 3: Theoretically, you could write python functions to create and modify dashboards!!!!
  - Treat json as dictionaries
  - String manipulation for TMDL, M code and file structure

Queue the Python!….Finally😅

- I’m assuming you already have python installed
- To install package:
  - py -m pip install powerbpy
- Test file (may need to change data locations):
https://github.com/Russell-Shean/powerbpy/blob/main/examples/create_example_dashboard.py
- Pause for demonstration of dashboard creation script

How does it work?

- Individual functions for adding different types of datasets
- Individual functions for each visual element
- Some visual elements are more complicated and call other functions from the package
  - e.g. the map uses the functions for adding text boxes and slicers
  - the new page function adds a text box
- User supplied ids instead of machine generated UUID
- Pause to show file structure

Development process

- Step 1: commit dashboard to git
- Step 2: make change in Power BI Desktop
- Step 3: commit change to git
- Step 4: Use diff to figure out how to reverse engineer changes and create python function
- Pause for demo

Power Bpy

- Collection of functions to create elements of Power BI dashboards
- Pypi: https://pypi.org/project/powerbpy/
- Github: https://github.com/Russell-Shean/powerbpy
- Website: https://www.russellshean.com/powerbpy/

Create new dashboard

- Function has to create multiple new files
- Default files copied from dashboard_resources folder inside PyPI package
  - from importlib import resources
  - traversable = resources.files("powerbpy.dashboard_resources")
  - with resources.as_file(traversable) as path:
    - shutil.copytree(path, project_folder_path)
  - Files are modified to add report name and unique UUID to template
- Pause for demo

Repeated pattern

- Load json as a dictionary
- Modify key-value pairs in the dictionary with user provided arguments
- Write out the modified dictionary as json

Load data

- Trickier because not specified using json
- Uses TMDL and M code
- For example: csv
  - Load csv as pandas dataframe
  - Loop through columns and write out specifications as TMDL
  - Reverse engineer and write out M code to load data in Power BI
- Central assumption: You want to do data prep in python not power BI
  - Power Bpy functions very picky about format
  - It’s easier to write python cleaning scripts than reverse engineer M code for edge cases

Load data

- Limited supported types
  - Again, easier to prep data and change format in python
  - Local csv, azure blob storage csv, TMDL file

Load data: TMDL

- Easiest
- Already in the correct format so you can
  - Copy tmdl file to semanticmodel folder
  - Update diagramlayout (update_diagramLayout)
  - Update model (update_model_file)
- Gotchya: sometimes the M code has hard-coded file paths to local files….
- Pause for demo

Load data: local csv

- More difficult
- Need to
  - Load csv as dataframe
  - generate all TMDL code from dataframe
  - generate M code to load csv file
  - Update diagramlayout and model
- Pause for demo

Load data: csv from ADLS

- Azure Data Lake Storage (ADLS) is a cloud data storage system from Microsoft
- Can directly link a power BI dashboard to an ADLS blob
  - Can update data in ADLS, Power BI will use new data automatically
- Power Bpy Function:
  - Attempts login to ADLS
  - Reads data from ADLS into pandas
  - Creates TMDL and M code to establish ADLS Power BI link

ADLS login

- 3 options
  - Browser authentication (user provides tenant id)
  - SAS url (user provides SAS for time limited file-scoped access)
  - Storage account key
    - On local computer: user is prompted to provide key interactively and key is stored in windows credential manager
    - In cloud environment: user passes credential directly (presumably from the cloud provider’s secret manager NOT HARD-CODED)
- Pause for demo of token storage

Add new page

- add_new_page(dashboard_path, page_name, title = None, subtitle = None)
- Creates a new folder for a new page, and new json for the page
- User provides a human readable id (page_name)
- User can provide a title and subtitle
  - Function calls add_text_box() function under the hood
- Pause for demo

add_text_box

- Example of json schema
  - Function arguments are passed into json
  - Json complexity can scale up and down
  - Optional arguments are added using key value pairs later
    - Example: background color
  - Pause to show code

add_shape_map

- data_source: The name of the dataset you want to use to build the map.
- shape_file_path: A path to a shapefile that you want to use to build the map.
- location_var: The name of the column in data_source that you want to use for the location variable on the map
- color_var: The name of the column in data_source that you want to use for the color variable on the map
- filtering_var: Optional. The name of a column in data source that you want to use to filter the color variable on the map.

add_shape_map

- color_palatte: A list of hex codes to use to color your data.
- percentile_bin_breaks: This should be a list of percentiles between 0 and 1 that you want to us to create bins in your data. If provided, a filtering_var must also be provided. This will create power BI measures that dynamically update when the data is filtered by things such as slicers.
-

```
## Add a map to page 3 ------------------------------------------------------------------

PBI.add_shape_map(dashboard_path = dashboard_path,
                  page_id = "page3",
                  map_id = "bigfoots_by_county_map",
                  data_source = "wa_bigfoot_by_county",
                  shape_file_path = "C:/Users/rshea/Downloads/2019_53_WA_Counties9467365124727016.json",

                  map_title = "Washington State Bigfoot Sightings by County",
                  #map_title = "",
                  location_var = "county",
                  color_var = "count",
                  filtering_var = "season",
                  #static_bin_breaks = [0, 15.4, 30.8, 46.2, 61.6, 77.0],
                  percentile_bin_breaks = [0,0.2,0.4,0.6,0.8,1],
                  color_palette = ["#efb5b9",  "#e68f96","#de6a73","#a1343c", "#6b2328"],
                  height = 534,
                  width = 816,
                  x_position = 75,
                  y_position = 132,
                  z_position = 2000,
                  add_legend = True
                  #add_legend = False
                  )
```

add_shape_map

- Pause to show map
- Multiple pieces:
  - Slicer to filter data by season
  - Legend that updates with filtered data
  - Choropleth map that updates with filtered data
  - Adds “measures” (essentially reactive functions) to the dataset
- Pause to show code

Development setup

- Code on GitHub
- Documentation
  - quartodoc: https://machow.github.io/quartodoc/get-started/overview.html
  - Hosted on GitHub pages
- GitHub actions:
  - Update documentation
  - Run (nonexistent) unit tests
  - Build and upload distributions to PyPI

Inspired?

- How you can help:
  - Python pull requests
  - Complaints
    - No unit tests, no oop, bad code design etc
  - Power BI feature requests
    - No python needed!
    - Description of what you want
    - Best: GitHub diff showing what you want

Get in touch

- Happy to chat about Power Bpy for free
- Also looking for freelance clients
  - Data pipelines
  - Cloud devops (Azure, Databricks, Posit)
  - Automation
  - Visualizations (maps, shiny dashboards, etc)
  - Weird projects like this one
- Contact info on next slide!

Contact

- LinkedIn: https://www.linkedin.com/in/russell-shean/
- GitHub: https://github.com/Russell-Shean
- Email: russshean@gmail.com
- Bluesky: http://rshean.bsky.social/