Introduction

Starting Materials

- The third tab in the Data Mining Hackathon spreadsheet (in this folder, not linked in the notebook due to issues with linking) is the EPA Strategic Goals
- Use that tab for examples of what I'm describing below, but I am trying to give an overview that does not exist in the spreadsheet!

The gist of the problem

- These goals are from 2020 and from 2014.
- These goals fall into a set of subject areas - they are not contiguous in the spreadsheet, roughly:
  - Air Quality (CAA - Clean Air Act)
  - Water Quality (CWA - Clean Water Act)
  - Toxic Substance Releases into the environment (TSCA)
  - Toxic Chemical Inventory (FIFRA)
  - Superfund Sites (CERCLA)
  - Brownfield Sites - think ugly block inside of a city
  - Leaking Unerground Storage Tanks (LUST)
  - Others too, just not listing all here
- For each of these there are metrics about things like
  - Permit processes - when applied for, when granted, etc
  - Accidental Leaks (water, air, superfund sites - all apply)
  - How many "things" (superfund sites, brownfield sites) are "ready for use" (we've finished cleaning them up) - think of these as following a "cleanup process"
- For each of the metrics, there are "dimensions" to think about
  - Date/Time range
  - In compliance vs out of compliance
  - Subject Area - mentioned above
- The best single page explaining all of these is here
- Once you dig in to this page you will see that most of the compliance areas have their own applications and databases that do the tracking for the EPA. Very typical - they probably built each app separately when that law came into effect.

The standard solution

- A Metrics Scorecard
- At all times shows:
  - A set of metrics
  - Their results across time

    - Choice boxes to change the dimensions above - change Subject Area from Air Metrics to Water Metrics, change time frame, etc.
- Under the covers there is usually:
    - Raw Data from each source, in case the drill downs from the scorecard require full detail
    - Integrated data stores - the "Event Lifecycle" or the "Accidental Emissions" - can be generalized so that you can show the emissions whether they are Air/Water/Superfund, etc.
    - Metric data stores - take the integrated data and transform it into the metrics and lay it out (usually a flat table, or a star schema if the reporting tool works with that) - so that the metrics and all the dimensional data are together for a single day - or some structure like that.
    - Some metric definition glossary that the user interface can use when it needs
    - The scorecard itself - pulling from the above sources. The scorecard would typically be done with Tableau or QlikView or PowerBI these days, or COGNOS, Web Focus, Business Objects - in older days.

Our problem - the standard solution is too boring to bother entering into a challenge.

Options to do something unique

The standard solution with all open source components

- I think this is too hard for the time we have.
- But the motivation - I'm glad you brought up Tableau the other day
- Paying for Enterprise Software drove me crazy at certain times in the last 20 years. And not being able to release a visualization to 1000's of employees without a big fat license cost - really constrained our solutions.
- And so for enterprises - a standard methodology and toolset for doing scorecards with open source - would be very valuable.
- But for our audience of the Data Visualization Society - this is not very exciting.
- So between too hard, and not directed at the Data Viz Society audience - this probably isn't a good idea.

Drama via heat maps and geographic maps

- A standard scorecard has little excitement in it.
- In other words, people may not pay enough attention to it. (Aside - in the big company they paid lots of attention if they had a bonus based on a metric - and management often did this - so you needed usable displays of information, they didn't have to be exciting).
- And so when I saw great data visualizations, I had two thoughts:
    - Wow, how cool - what a great way to make that point
    - But no one is going to watch this time lapse video every day. They need to get quickly to the metric they are trying to improve. So these super cool visualization techniques aren't good for every day work.

A "different" visualization technique for process flow diagrams

- So we need to combine the utility of the standard scorecard with the cool/wow factor of the newer data visualization techniques.
- (Making this up, we have to study the actual EPA data)
- We could create a process diagram
  - Step 1 - Fill out form abc
  - Step 2 - Form gets reviewed approval goes to Step 3, rejection back to Step 1
  - Step 3 - Requestor now carries out whatever they got permission for
  - Step 4 - the Request is closed out
- Now, we could
  - Put the dynamic numbers on it - 327 forms are on step 2, 193 requests are currently being worked
  - Put the choice boxes around it - geography, time dimension, which process so you could change to look at a different process.
- I really think this could be a great way of doing process oriented scorecards and so I think this has a lot of potential for us.

A "different" visualization technique for the accidental emissions

The sample data

- SampleData1.xlsx and 2246339817.CSV are the same data - I can see the .xlsx becoming more dictionary like, and we will have many .csv's that we will iterate over.
- Each row is about the release of Chemical Id/Name, from Facility Name, with amounts released to air, land, water.
- Each row has identifiers that probably link to other tables with more information - the Chemical Id, the Facility Id, etc.
- It looks like I only grabbed a year, we need an individual date (I might have missed selecting it, or it is in a table we would link to).

What we want to show

- Dynamic Data per Emission (and at aggregate levels)
  - What is being released
  - How much of it
  - Being released to different destinations (and in total)
- Attribute Information
  - Geography
  - Chemical Info
  - Facility Info
  - Dates

Let the user choose

- Generally same as attribute information
- Driving our data to display
  - Until their selections are at an individual row level, we will almost always be showing data at an aggregate level
  - State of Colorado, month of June 2019, all releases to water
  - So think that if they haven't made choices - we will be showing great big totals of some kind

What does our display look like and how dynamic can we make it

Option 1 - picture of pipe with junk flowing through and out (options not mutually exclusive)

- Can we show something that looks like a pipe
- Inside of it we would show chemical information
- Coming out, we would have flows to air, land and water (all visuals)
- We would show amounts on the output flows, probably a total amount (in the pipe visually)?
- We would try to make the amount show visually with thinner/thicker streams of stuff
- Again these are big totals that get refined down to smaller ones as the user chooses.
- When an individual row is reached, we will want some additional capabilities
  - popups or extra windows
  - with as much info as we can get about the incident
  - and links to the rest

Option 2 - US Map Based

- User choice stuff all generally applies
- We would be doing totals a lot, but here, we would be able to show many individual incidents at once
- We should be able to show relative size here also, but the large scale of the map might make that tough

Option 3 - Top half of screen is Opt 1 or 2, bottom half is individual rows involved, with export capabilities

Option 4 - Map on top and choice boxes on top, driving effluent picture below

Time dimension

Discussion with Emily 1/27/21

- Sankey diagrams!!!
- ipywidgets in jupyter would allow some interaction
- streamlits will allow interaction outside of notebook - streamlit.io

In [8]:
```
import json
import urllib
```

```
import pandas as pd
import numpy as np
```

In [18]:
```
import chart_studio.plotly as py
```

In [10]:
```
df = pd.read_csv("test_sankey1.csv")
df
```

Out[10]:

	Source	Target	Value
0	0	0	1
1	0	1	2
2	0	2	3
3	1	0	2
4	1	1	1
5	1	2	3

In [21]:
```
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode()
```

In [22]:
```
data_trace = dict(
    type='sankey',
    domain = dict(
      x =  [0,1],
      y =  [0,1]
    ),
    orientation = "h",
    valueformat = ".0f",
    node = dict(
      pad = 10,
      thickness = 30,
      line = dict(
        color = "black",
        width = 0.5
      ),
      #label =  refugee_df['Node, Label'].dropna(axis=0, how='any'),
      #color = refugee_df['Color']
    ),
    link = dict(
      source = df['Source'].dropna(axis=0, how='any'),
      target = df['Target'].dropna(axis=0, how='any'),
      value = df['Value'].dropna(axis=0, how='any'),
  )
)

layout =  dict(
    title = "Mick Test",
    height = 772,
    width = 950,
    font = dict(
```

```
      size = 10
    ),
)


fig = dict(data=[data_trace], layout=layout)
#fig.show()
py.offline.iplot(fig, validate=False)

#plotly.offline.iplot(...)
```

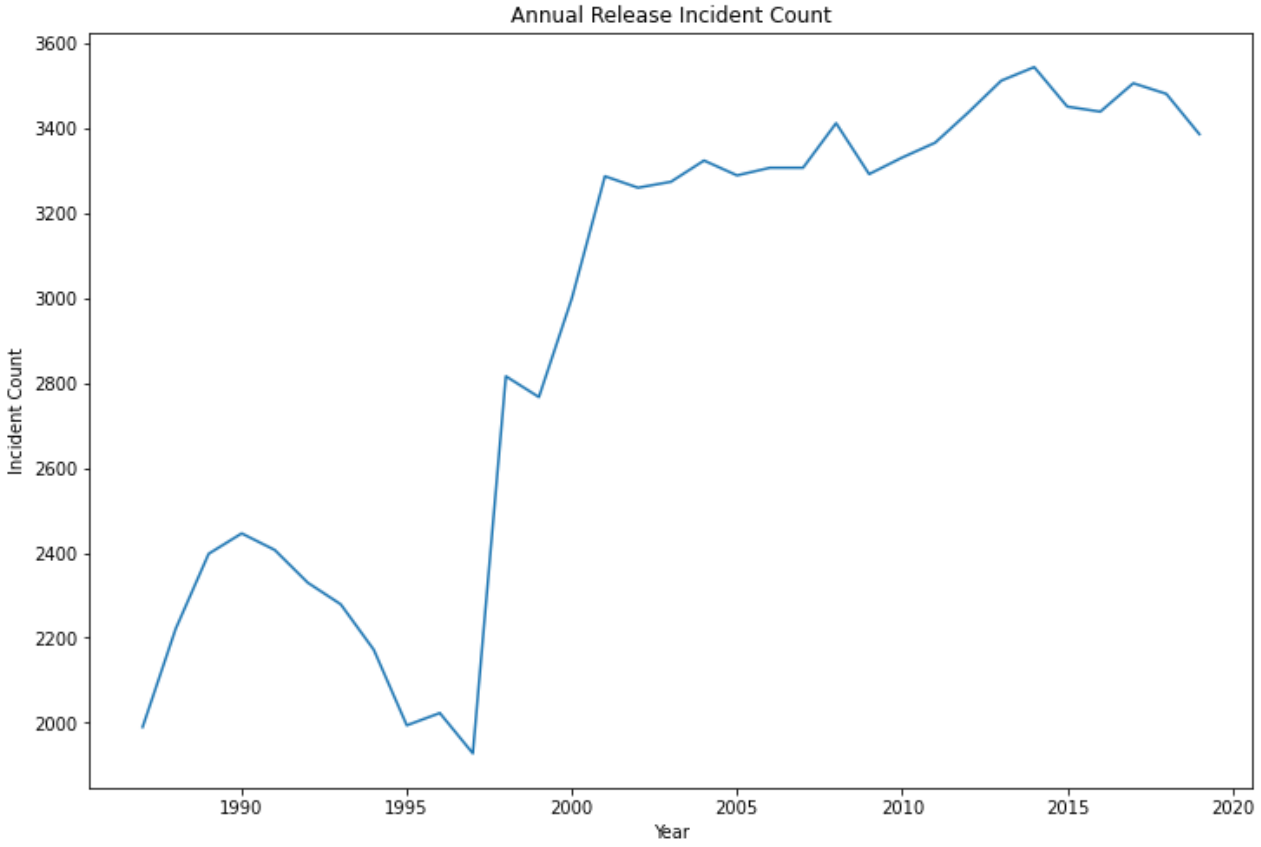
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-22-09e11416afc8> in <module>
     36 fig = dict(data=[data_trace], layout=layout)
     37 #fig.show()
---> 38 py.offline.iplot(fig, validate=False)
     39
     40 #plotly.offline.iplot(...)

AttributeError: module 'chart_studio.plotly' has no attribute 'offline'
```

In [ ]:

In [2]:

```
import streamlit
import pandas as pd
import matplotlib
matplotlib.use()
import matplotlib.pyplot as plt
import numpy as np

rls_complt_raw = pd.read_csv("Data/releases_complete2.CSV")
use_columns = ['REPORTING_YEAR', 'TOTAL_ON_OFF_SITE_RELEASE']
yrly_rls = rls_complt_raw[use_columns].groupby(['REPORTING_YEAR'], as_index=False).coun
#yrly_rls

fig, ax = plt.subplots(figsize=(12,8))

ax.set_title('Annual Release Incident Count')
ax.set_ylabel('Incident Count')
ax.set_xlabel('Year')

x = yrly_rls['REPORTING_YEAR']
y = yrly_rls['TOTAL_ON_OFF_SITE_RELEASE']

#ax.set_ylim(80, 100)
ax.plot(x, y)
#plt.savefig("Data/Output/rainfall_validation/average_monthly_rainfall.png")       # ea

plt.show()
```

```
C:\Users\MickC\Documents\virtual_envs\data_viz\lib\site-packages\IPython\core\interactiv
eshell.py:3146: DtypeWarning: Columns (11) have mixed types.Specify dtype option on impo
rt or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```



In [ ]:

Streamlits app

First build a time lapse, presentation style app

Then decide on interactivity - probably as a second app

Time Lapse Thoughts - FIrst Pass

- US Map with Totals - plain facts as title at top, but what does it mean as transition text on bottom (15)
- A single incident could be (5 * 30)
  - Example 1 - as Sankey diagrams
  - ...
  - Example N - to cover many input and output types
- Trending Analysis - several charts
- Recommendations for an Interactive Analytical App for the public
  - Map based
    - US with totals by state
    - State diagram drillable to individual release
    - Sankey with each

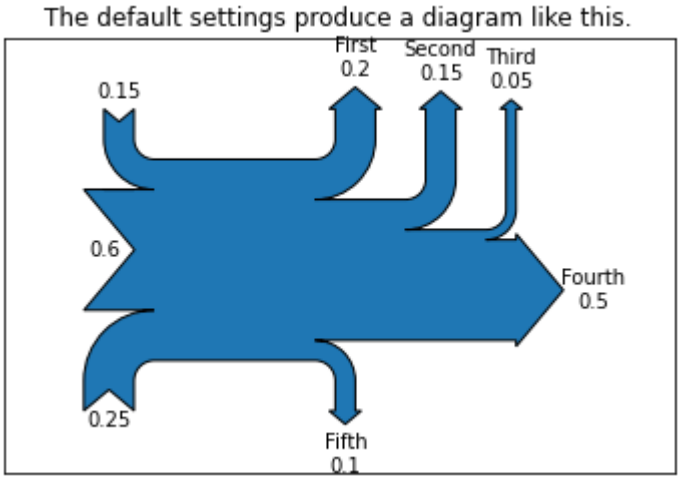
In [ ]:

Sankey diagram with matplotlib

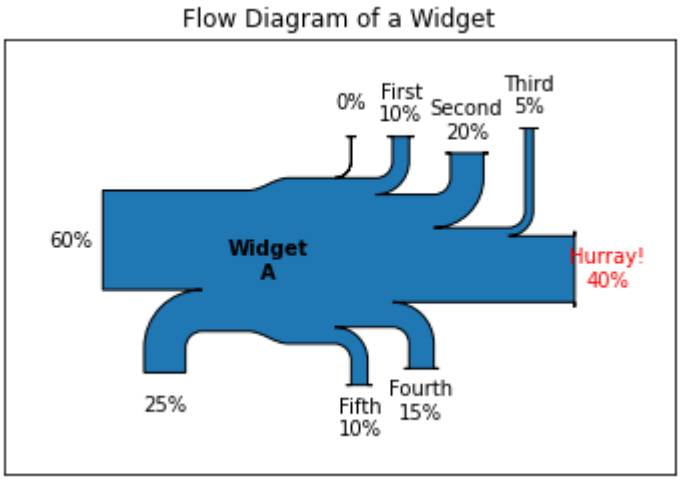
In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from matplotlib.sankey import Sankey
```

In [2]:
```
Sankey(flows=[0.25, 0.15, 0.60, -0.20, -0.15, -0.05, -0.50, -0.10],
       labels=['', '', '', 'First', 'Second', 'Third', 'Fourth', 'Fifth'],
       orientations=[-1, 1, 0, 1, 1, 1, 0, -1]).finish()
plt.title("The default settings produce a diagram like this.")
```

Out[2]: Text(0.5, 1.0, 'The default settings produce a diagram like this.')



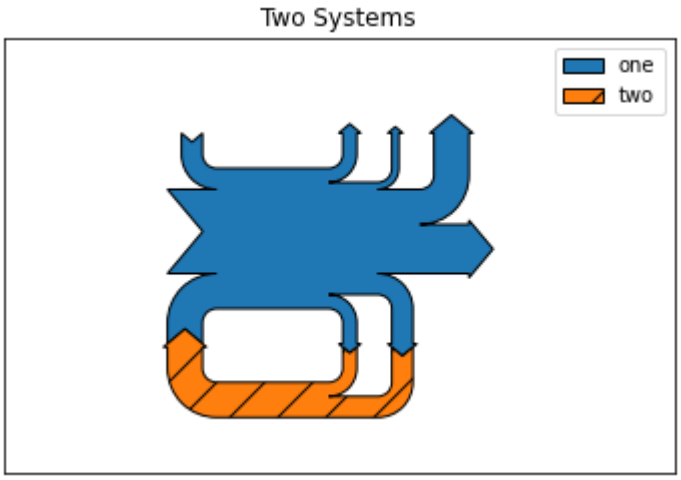
In [3]:
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, xticks=[], yticks=[],
                     title="Flow Diagram of a Widget")
sankey = Sankey(ax=ax, scale=0.01, offset=0.2, head_angle=180,
                format='%.0f', unit='%')
sankey.add(flows=[25, 0, 60, -10, -20, -5, -15, -10, -40],
           labels=['', '', '', 'First', 'Second', 'Third', 'Fourth',
                   'Fifth', 'Hurray!'],
           orientations=[-1, 1, 0, 1, 1, 1, -1, -1, 0],
           pathlengths=[0.25, 0.25, 0.25, 0.25, 0.25, 0.6, 0.25, 0.25,
                        0.25],
           patchlabel="Widget\nA")  # Arguments to matplotlib.patches.PathPatch()
diagrams = sankey.finish()
diagrams[0].texts[-1].set_color('r')
diagrams[0].text.set_fontweight('bold')
```



In [4]:

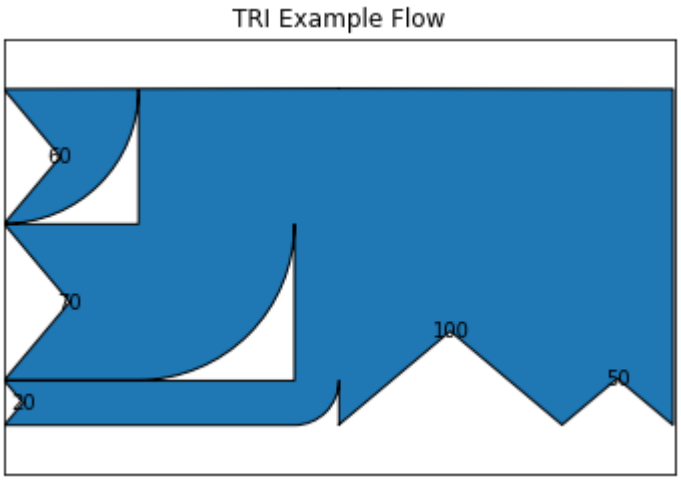
```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, xticks=[], yticks=[], title="Two Systems")
flows = [0.25, 0.15, 0.60, -0.10, -0.05, -0.25, -0.15, -0.10, -0.35]
sankey = Sankey(ax=ax, unit=None)
sankey.add(flows=flows, label='one',
           orientations=[-1, 1, 0, 1, 1, 1, -1, -1, 0])
sankey.add(flows=[-0.25, 0.15, 0.1], label='two',
           orientations=[-1, -1, -1], prior=0, connect=(0, 0))
diagrams = sankey.finish()
diagrams[-1].patch.set_hatch('/')
plt.legend()
```

Out[4]: <matplotlib.legend.Legend at 0x290fdc26e20>



In [11]:

```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1, xticks=[], yticks=[], title="TRI Example Flow")
sankey = Sankey(ax=ax)
sankey.add(flows=[100, 50, 60, 70, 20],
           orientations=[0, 0, 1, 1, 1],
           rotation=90)
diagrams = sankey.finish()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: