
Seeq
conneqt
IEM & AI Track
#allin

Seeq

Jeff Skarda

Digital Solutions Sr. Consultant

Kimberly-Clark

conneqt

Better Care for a Better World

Kimberly-Clark

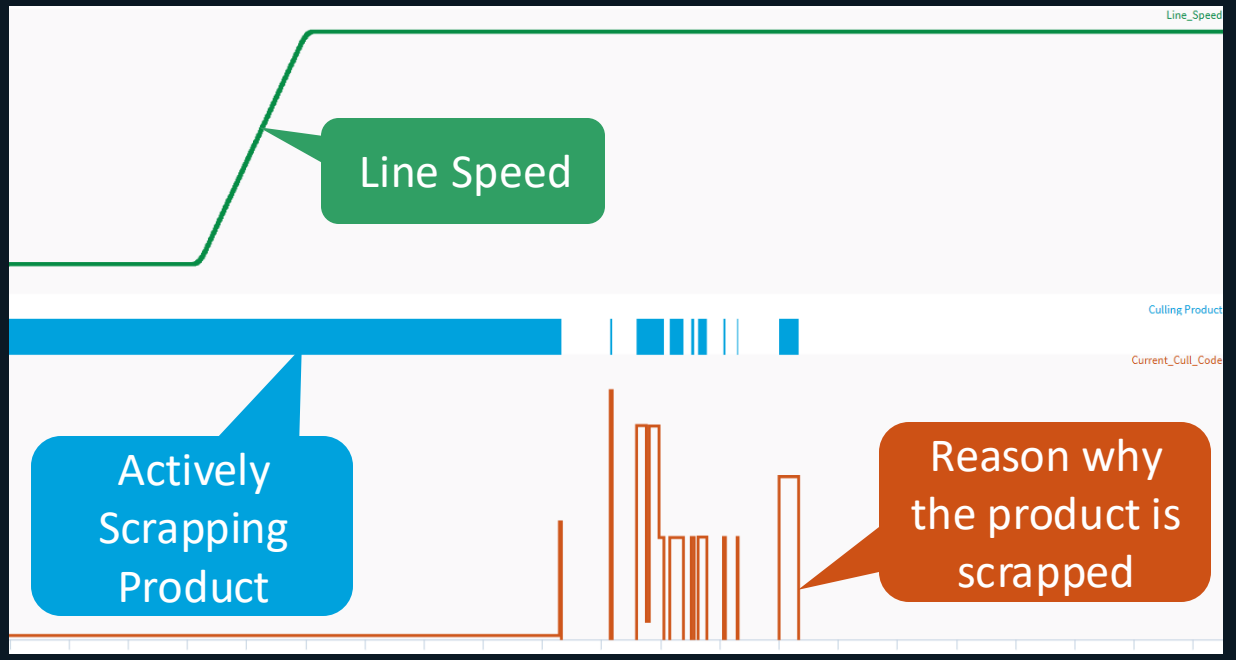
Countries where our leading brands are sold

Number of countries in which our products hold the #1 or #2 brand share

Of the world's population use one or more of our products each day

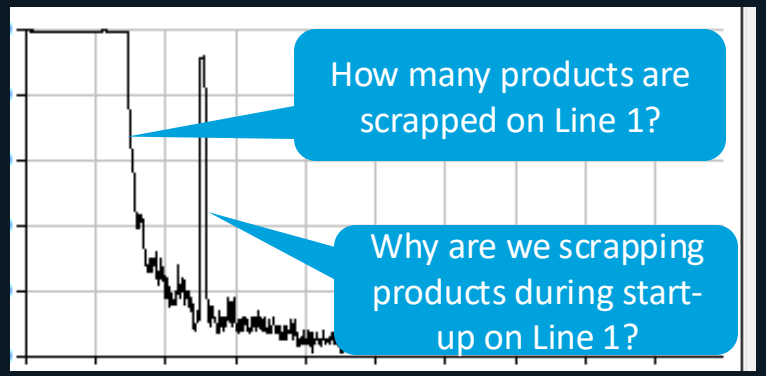
conneqt

Diaper assets scrap product during ramp up



conneqt

Analyzing startup waste using a legacy application

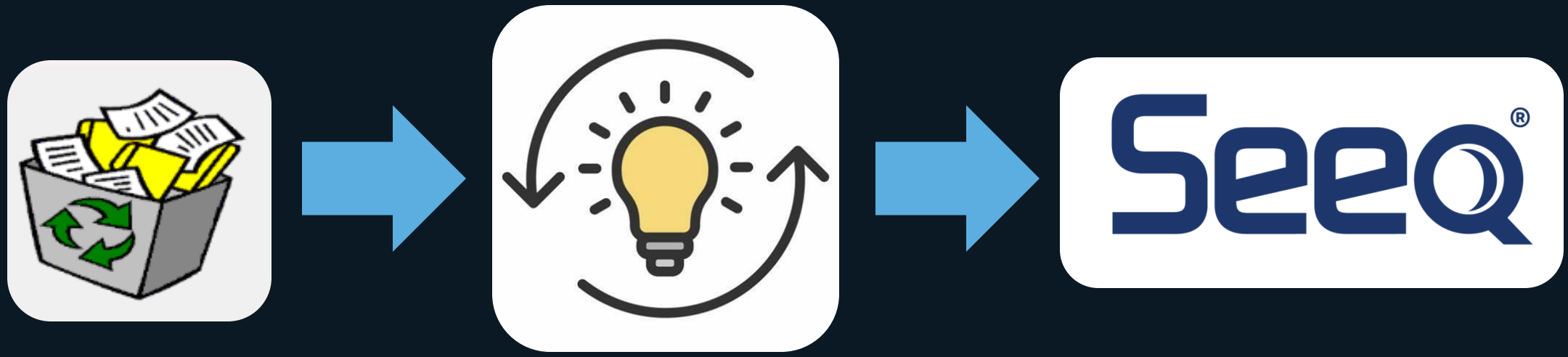


conneqt

1-2 distinct user
2-4 views per year

conneqt

The need to re-imagine



Requirements

Requirements	Response:
Easy access to MES Waste Events	**Seeq already has access**
Data processing is required to reconstruct the machine start-up waste profile based on individual on waste events	**Seeq Data lab uses Python. Let's do it in Python**
Determine and visualize best in class, worst in class	**There are lots of options for data visualization using Python Libraries. How about a bubble chart?**
Tree Map of Waste Events by Reasons	**Plotly has tree maps**
Add $$$ cost impact to bring the data to life for the team	**That's easy**
Provide an always accessible dashboard	**We can schedule the Seeq Data Lab project to pull and stage the data each night**

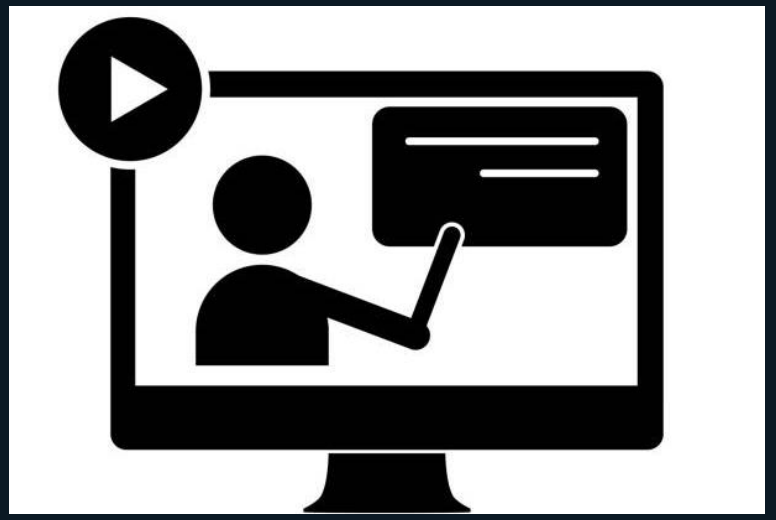
Foundational knowledge to speak the language

"An electrical engineer trying to write python code"

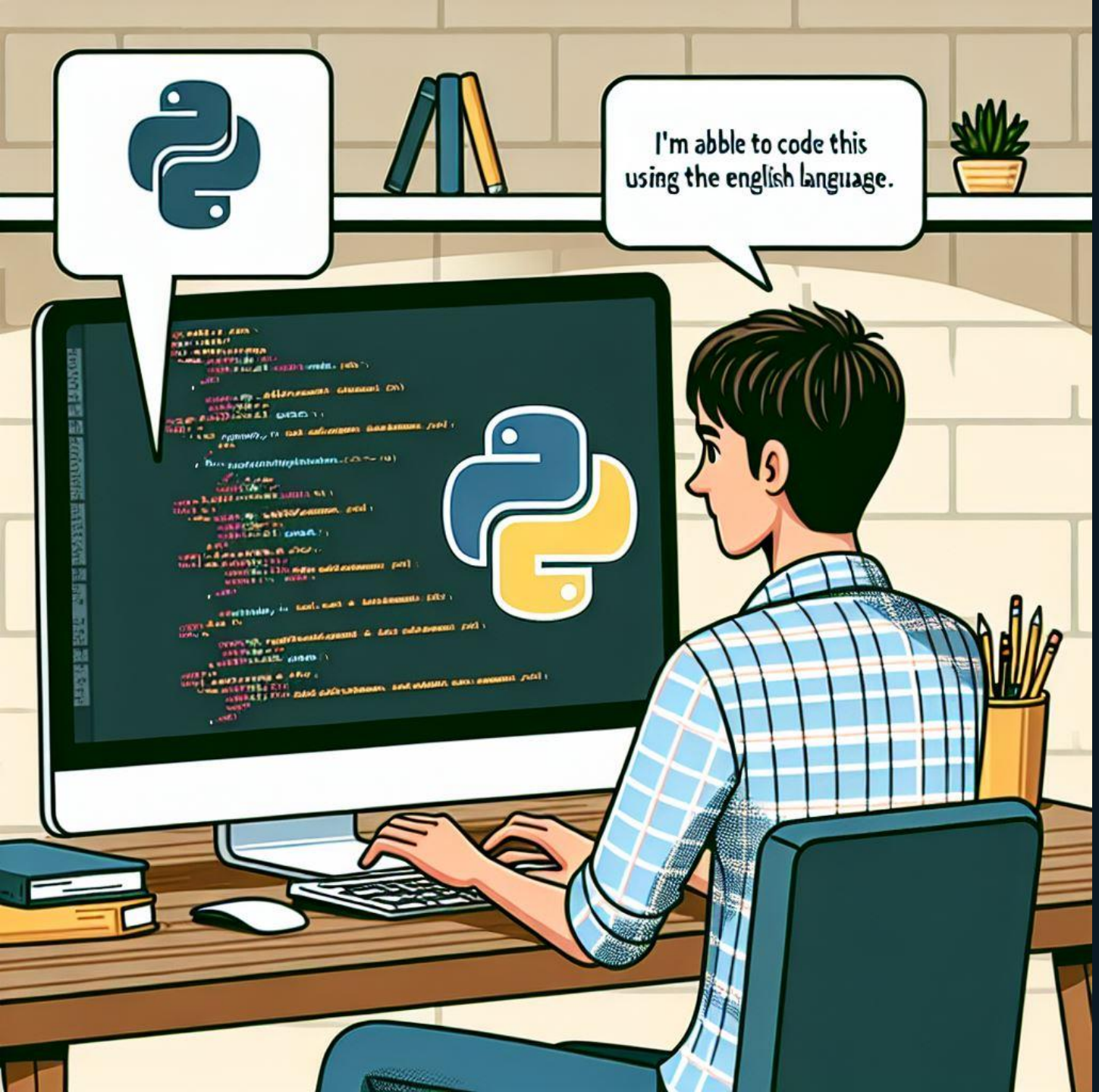
pandas

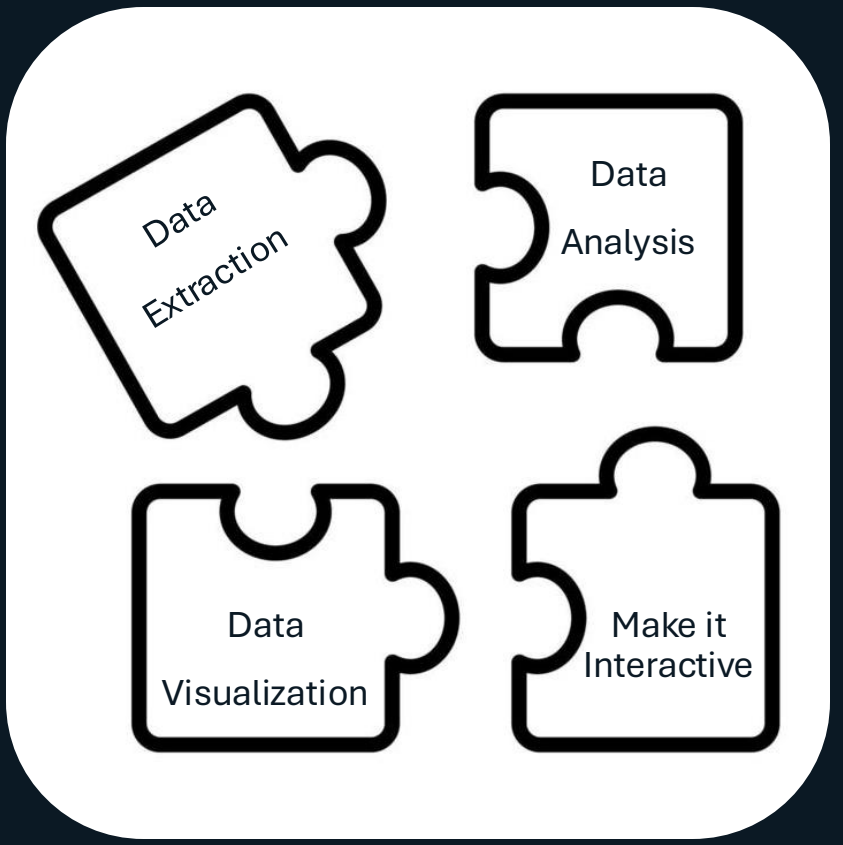
plotly

Seeq DATA LAB



conneqt

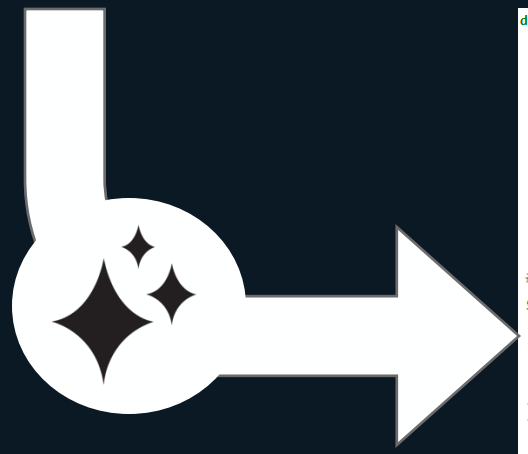

I'm abble to code this using the english language.


Data Extraction
Data Analysis
Data Visualization
Make it Interactive

conneqt

Puzzle Piece #1: Data Extraction

- “Find all tags called “Waste Events”
- “Query the last 6 months of data for each asset’s waste event”
- “Remove data where production mode is Trial or Project Startup”
- “Now re-write this code as a function called get waste events”

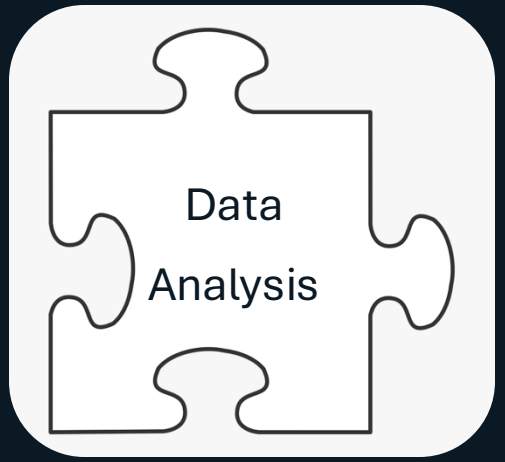


```python
def get_waste_events(asset, startDateTime, endDateTime):
    """
    Function to search for 'WasteEvents' in a specified asset path and pull the data within a given time range.

    Parameters:
    asset (str): The asset path to search within.
    startDateTime (str): The start date and time for the data pull.
    endDateTime (str): The end date and time for the data pull.

    Returns:
    pd.DataFrame: A DataFrame containing the pulled data, sorted by 'Capsule Start' and 'current_cut_count'.
    """
# Define the search query
search_query = {
    'Name': f'{asset} Waste Events',
    'DataSource': 'PIPE',
    'Type': 'StoredCondition'
}

# Execute the search query
search_results = spy.search(search_query,quiet=True)


# Pull data from the worksheet with the search query
waste_df = spy.pull(search_results, start=startDateTime, end=endDateTime,quiet=True)
```

Data Extraction

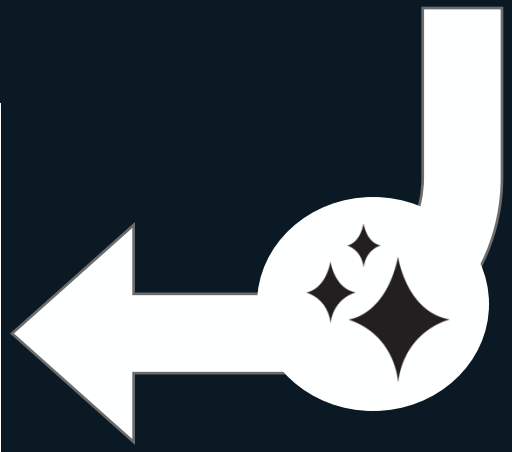
Puzzle Piece #2: Data Analysis

Data Analysis



- “Given the waste events dataframe, identify the zero-reference indicating this is a new machine start-up event”
- “Compute the number of products between the zero reference and this event”
- “Calculate the total number of good products, scrapped products and percent waste for each machine start-up”
- “Now re-write this code as a function called calculate startup waste”

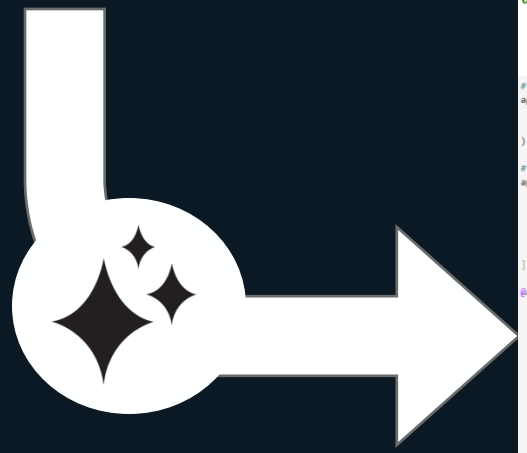
```
def calculate_startup_waste(startUp_DF, waste_df, max_cut_count):
    try:
        """
        Function to calculate startup waste and save the results to a CSV file.

        Parameters:
        startUp_DF (pd.DataFrame): The DataFrame containing startup events.
        waste_df (pd.DataFrame): The DataFrame containing waste data.
        max_cut_count (int): The maximum cut count to consider.

        Returns:
        pd.DataFrame: A DataFrame containing the calculated startup waste.
        """
 #Iterate through the data set and identify the zero reference of each cut count indicating this is a new start-up.
 for i in range(len(waste_df)):
     if waste_df.loc[i, 'machine_state'] <= 2:
         stop_event_occurred = True
         waste_df.loc[i, 'ZeroReference'] = False
     elif stop_event_occurred and waste_df.loc[i, 'machine_state'] == 3:
         stop_event_occurred = False  # Reset the flag after marking the first 3.
         waste_df.loc[i, 'ZeroReference'] = True
     else:
         waste_df.loc[i, 'ZeroReference'] = False
```



conneqt

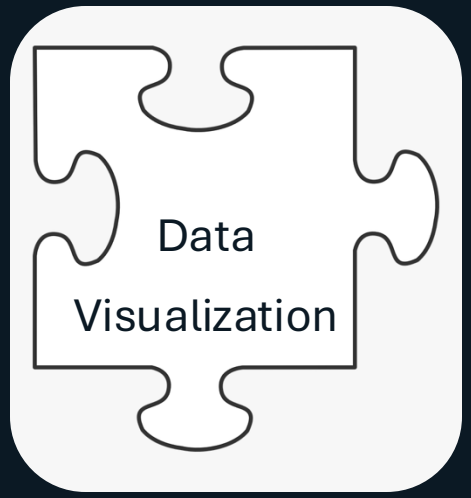
Puzzle Piece #3: Data Visualization

- “Create a Plotly Dash App used to visualize data staged as a .CSV file. This app should have 3 tabs, summary, graphs and table.”
- “Load data from the summary_data.CSV as a bubble chart”
- “Make the background red anytime start-up waste is above x%”
- “Create a tree map showing the distribution of waste reasons”



```python
def plot_distribution(combined_data_dict):
    fig = go.Figure()
    if not combined_data_dict:
        return fig

# Initialize the Dash app
app = Dash(
    "Kimblery-Clark Waste Analysis Suite",
    requests_pathname_prefix=requests_pathname_prefix
)

# Define the layout
app.layout = html.Div([
    get_layout(),
    dcc.Store(id='queried-data'),  # Store to hold queried data
    dcc.Store(id='startup-waste-data'),  # Store to hold startup_waste_df
    dcc.Store(id='selected-cell-store'),  # Store to hold clicked cell information
    dcc.Download(id="download")  # Ensure this component is included
])

@app.callback(
    [Output('queried-data', 'data'),
     Output('startup-waste-graph', 'figure'),
     Output('startup-waste-dist', 'figure'),
     Output('main-tabs', 'value'),
     Output('loading-placeholder', 'children'),],
    [Input('submit-button', 'n_clicks'),
     Input('trend-button', 'n_clicks')],
    [State('start-date', 'date'),
     State('start-time', 'value'),
     State('end-date', 'date'),
     State('end-time', 'value'),
     State('asset-selector', 'value'),
     State('include-aborted-toggle', 'value'),
     State('cut-count-slider', 'value'),
     State('summary-grid', 'selectedRows')]
)
def handle_submit(config_clicks, table_clicks, start_date, start_time, end_date, end_time, config_selected_assets, include_aborted_toggle, max_cut
```

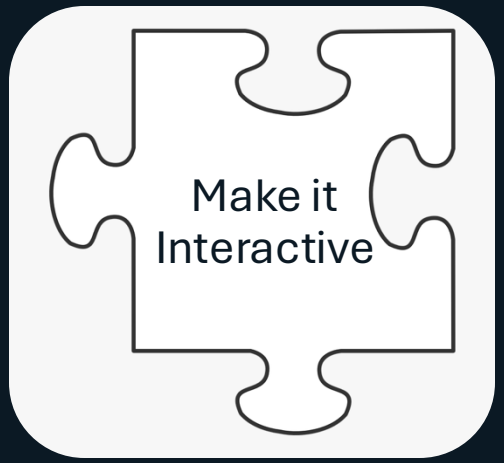
Data Visualization



conneqt

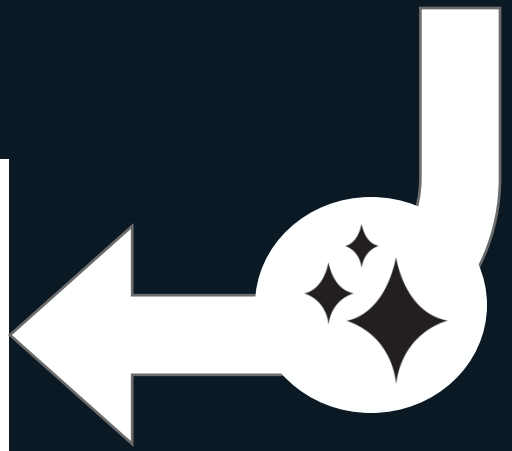
Puzzle Piece #4: Make it Interactive

Make it Interactive



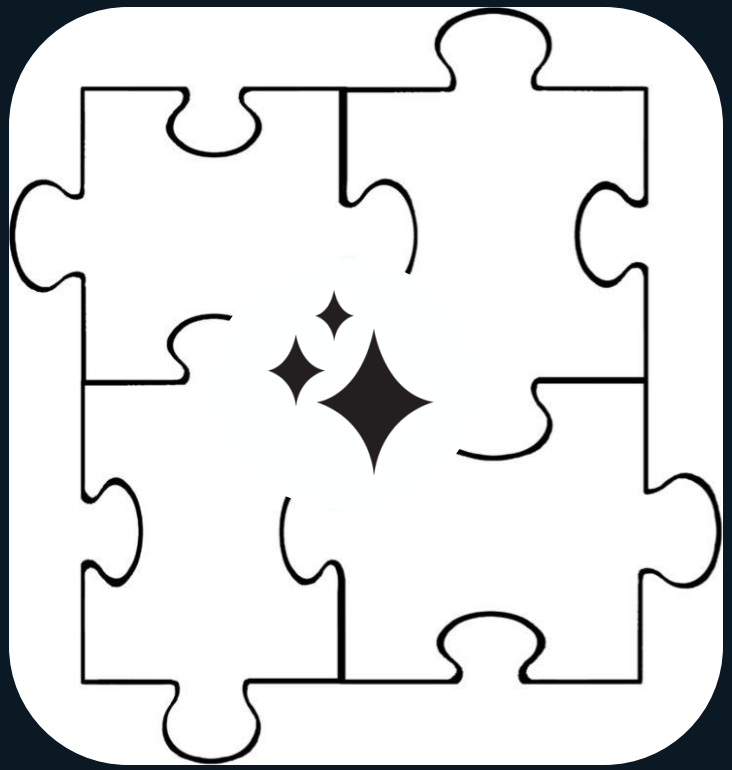
"Provide an input field to allow a user to select asset class"
"When a user selects a specific asset class, filter the bubble chart for only assets of that class"
"Allow a user to select 1 or more assets to trend and compare similar assets"
"Add an ability to download the summarized data set"

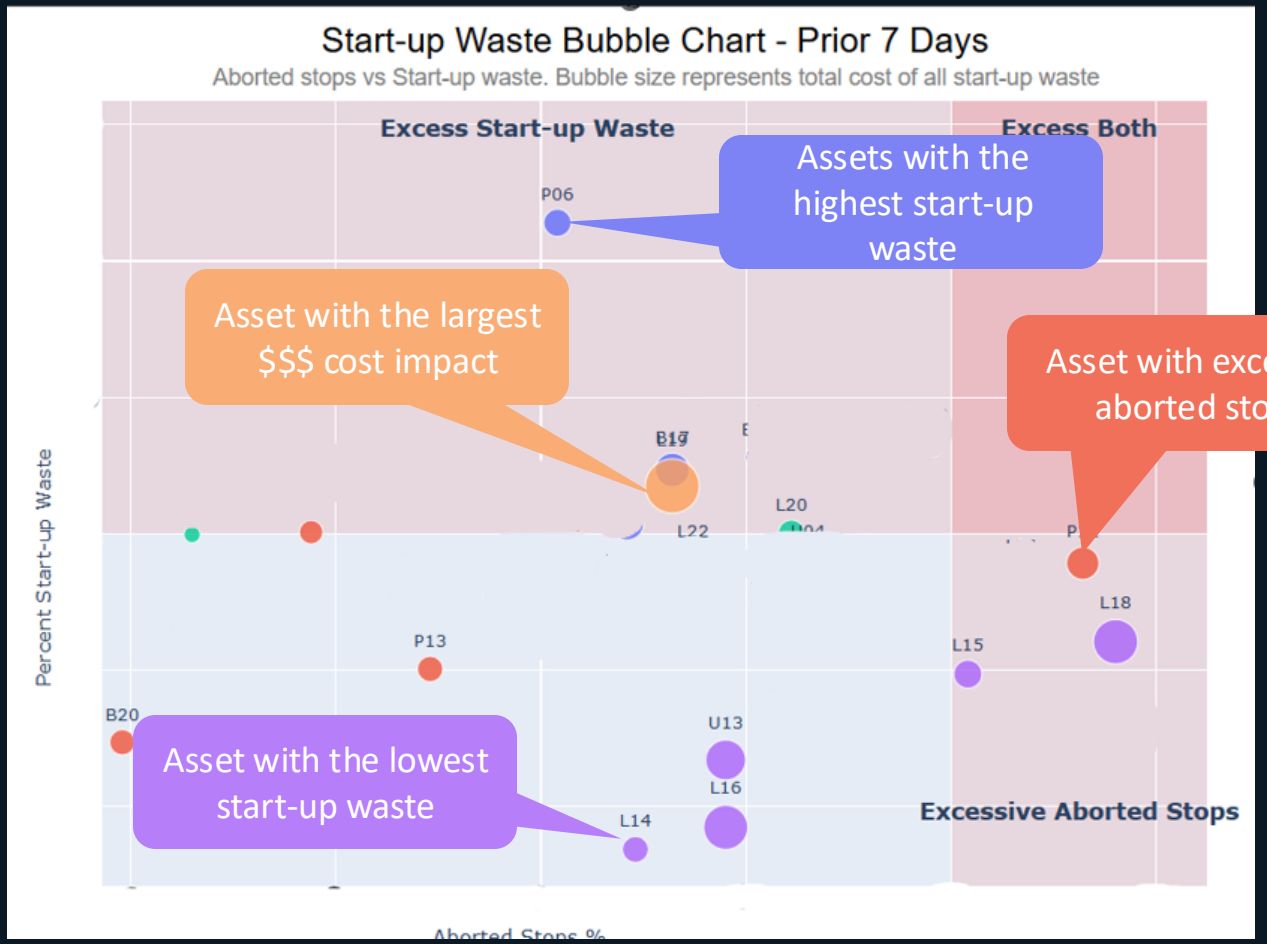
```
@app.callback(
    Output('summary-grid', 'selectedRows'),
    Input('bubble-chart', 'clickData'),
    State('summary-grid', 'rowData')
)
def handle_bubble_click_update_table(clickData, rowData):
    if not clickData or not rowData:
        return []

    # Get clicked bubble data
    point = clickData['points'][0]
    asset_name = point['text']  # Assuming 'Asset Name' is in the hover text

    # Find the corresponding row in the table
    for row in rowData:
        if row['Asset'] == asset_name:  # Match asset name
            return [row]  # Highlight this row
# Callback to download data
@app.callback(
    Output("download", "data"),
    Input("download-button", "n_clicks"),
    prevent_initial_call=True,
)
def download_data(n_clicks):
    asset_summary_df = pd.read_csv('Storage/asset_summaries.csv')
    return dcc.send_data_frame(asset_summary_df.to_csv, "asset_summary_df.csv")
```



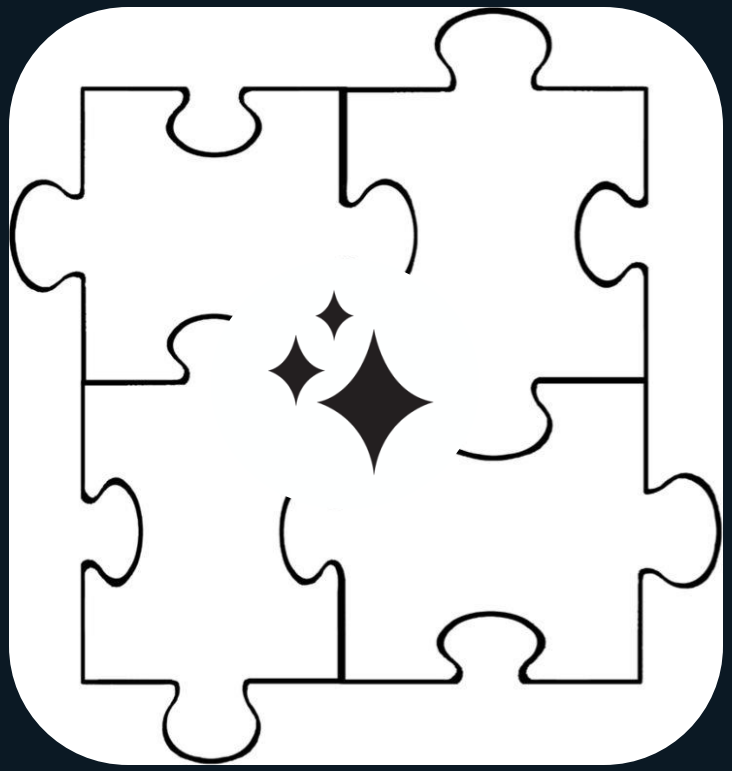
conneqt

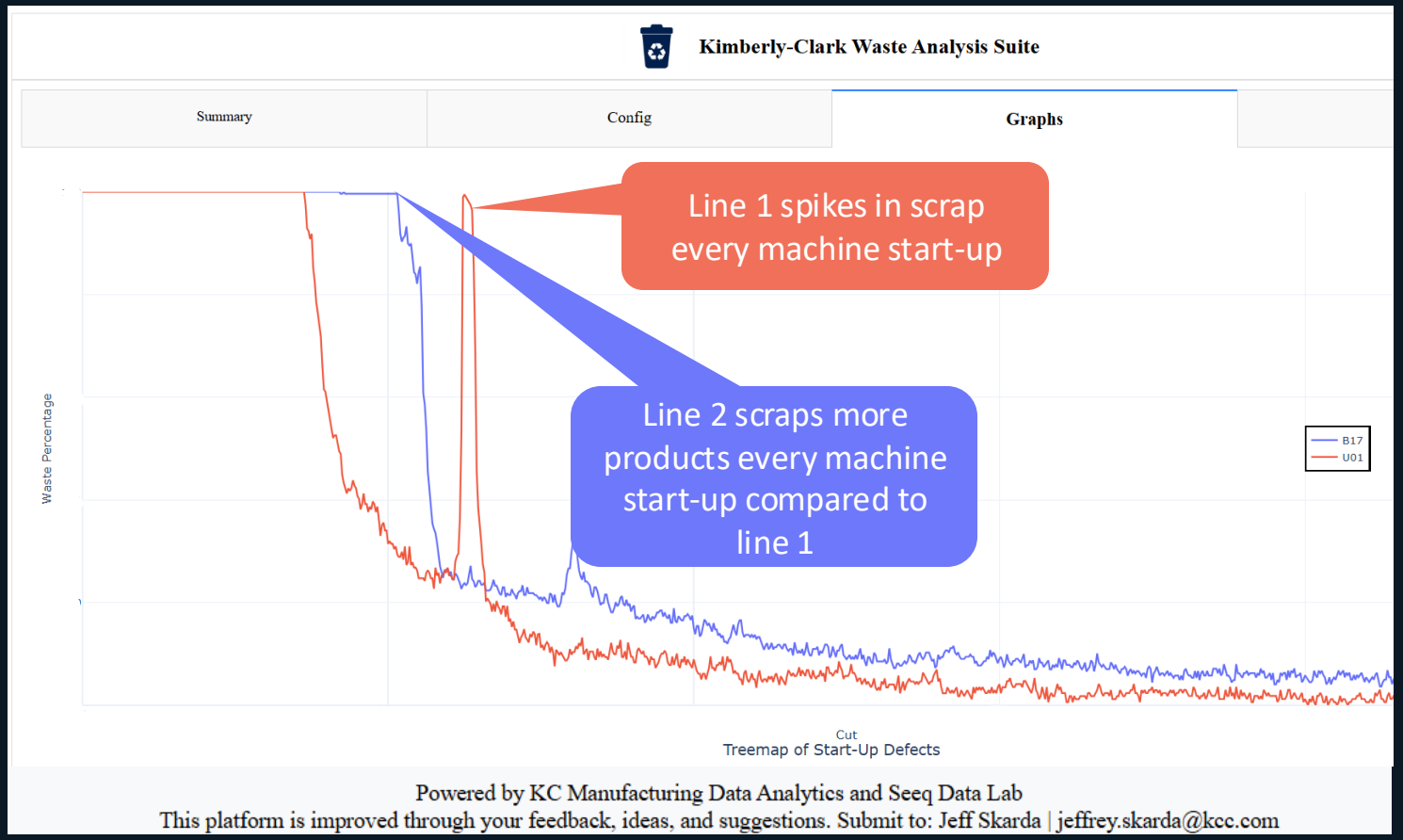
Putting the pieces of the puzzle together



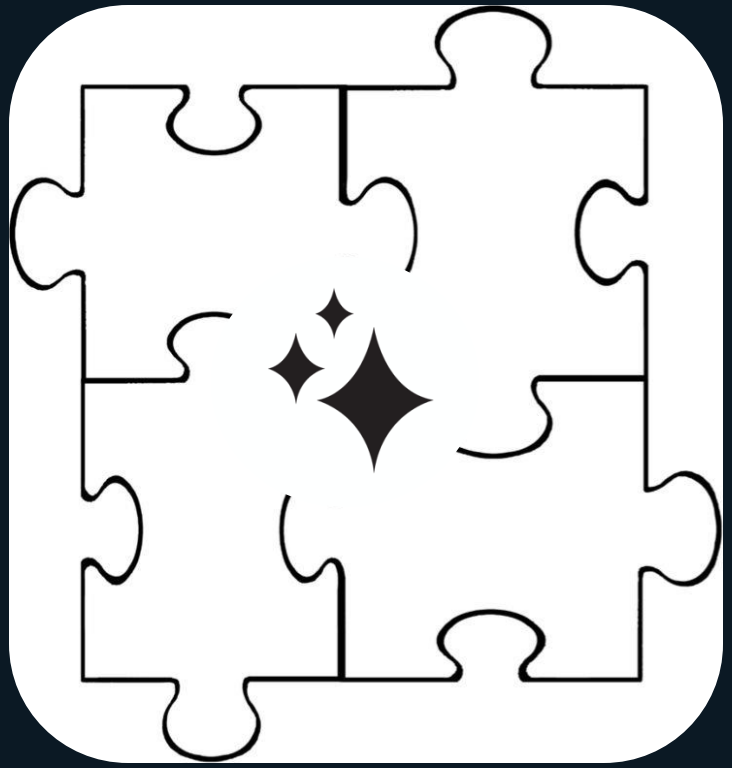


Putting the pieces of the puzzle together





Putting the pieces of the puzzle together



Focus Here

Focus Here

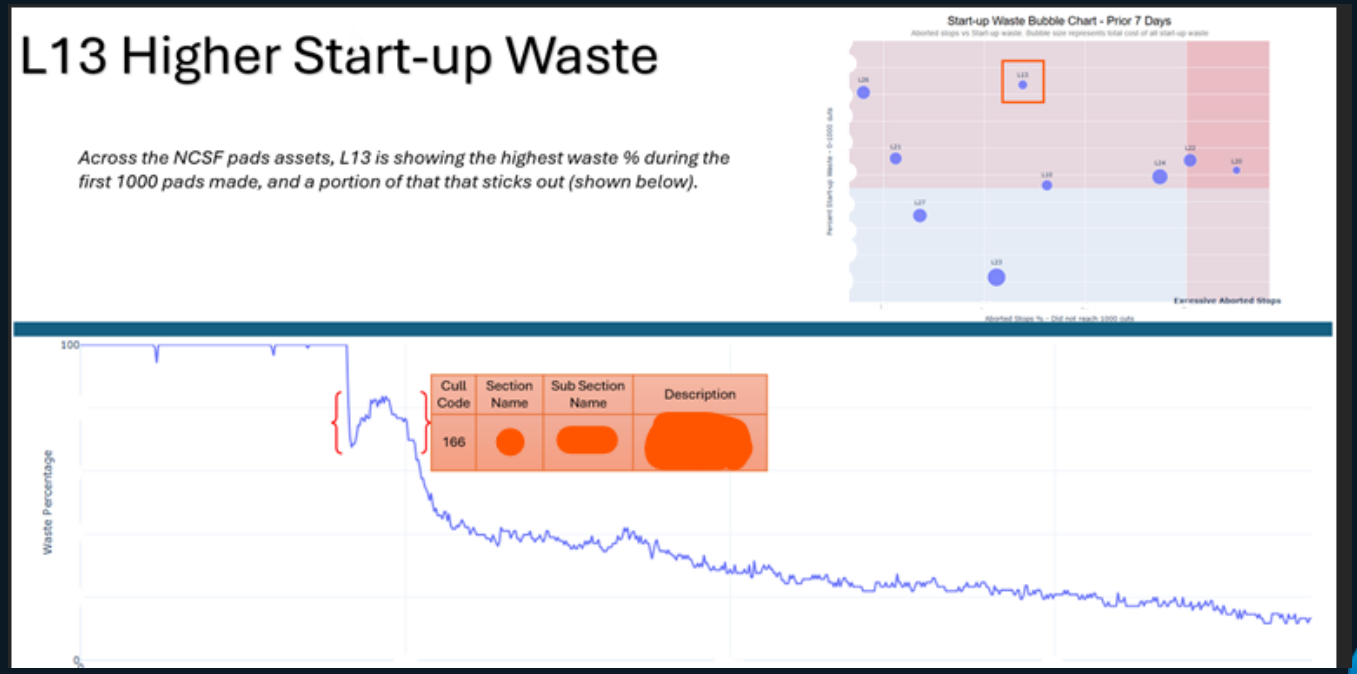
Not Here

conneqt

Democratizing start-up waste data provides quick wins

"Thanks to the tool, I was able to identify cull 414 as a problem cull on P4 vs the other assets. I shared the info with the mill so they can act."

"*This was the info I mentioned to you on the way out yesterday. I am not sure how fixable this material movement is, but I wanted to put it on your radar to get a small waste win for the year-on-year results.*"



Democratizing start-up waste data visualization

1-2 distinct users
2-4 views per year

65 distinct viewers
13 views per week

conneqt

Thank you!

Jeff Skarda

Kimberly-Clark