Paper DV08

Visualizing the Invisible: Patient Journey with Waterfall, Spider, and Swimmer Plots

Varsha Patil, MARS Group Inc, Mumbai, India
Sunita Poojary, GENINVO, Mumbai, India

ABSTRACT

Patient-level plots are essential in clinical data analysis, providing detailed insights into individual treatment responses across therapeutic areas such as oncology and neurology. These visualizations help researchers and clinicians assess variability in patient outcomes, identify trends, and make informed decisions. Waterfall plots illustrate individual patient responses by displaying changes in tumour size or clinical metrics. Spider plots track longitudinal variations, highlighting shifts over time in response to therapy. Swimmer plots visually map patient trajectories, representing treatment duration, response patterns, and key clinical events. This paper highlights the significance of these plots in evaluating treatment efficacy and disease progression. It also demonstrates their implementation using Python on Databricks platform with synthetic data, offering a practical perspective on their use in clinical trials. While patient-level plots provide valuable insights, integrating them with cohort-level analyses ensures a comprehensive interpretation of clinical trial data, ultimately enhancing patient care and improving overall outcomes.

INTRODUCTION

Clinical trials have traditionally relied on aggregate/summary statistics to evaluate treatment efficacy across patient populations. While these summaries are essential for regulatory and statistical reporting, they often obscure the nuanced and heterogeneous responses observed at the individual level. This limitation is particularly evident in therapeutic areas such as oncology and neurology, where disease progression and treatment response can vary significantly from one patient to another. To address this gap, patient-level visualizations have emerged as powerful tools that offer a more granular and insightful perspective. These plots help uncover variability, detect outliers, and visualize treatment dynamics over time, enhancing the interpretability of clinical data and supporting personalized medicine by highlighting individual trajectories that may inform tailored therapeutic decisions.

This paper focuses on the following key areas:

- Synthetic Datasets for illustration
- Python Libraries behind the visuals
- Waterfall plots
- Spider plots
- Swimmer plots
- Patient's Journey through Visualization
- Conclusion

SYNTHETIC DATASETS FOR ILLUSTRATION

A synthetic oncology dataset is developed to simulate longitudinal tumor response data across multiple patients undergoing treatment. As illustrated in Table 1, the dataset captures key variables including patient ID, visit timing, tumor measurements, percent change from baseline, treatment details, and overall response categories(CR: Complete Response, PR: Partial Response, SD: Stable Disease, PD: Progressive Disease). These variables collectively reflect the dynamic nature of disease progression and therapeutic impact. While Table 1 provides a glimpse into the dataset structure, the full dataset used for visualization comprises data from 50 patients. This comprehensive dataset (screenshot in Table1) serves as a foundational framework for generating two visualizations: waterfall plot to depict the magnitude of tumor shrinkage or growth, spider plots to illustrate temporal trends in tumor size. Independent dataset (screenshot in Table 2) will be used to demonstrate maximum possibilities of swimmer plot to represent treatment duration alongside response milestones.

Table 1 (Used for Waterfall plot & Spider plot)

PatientI	Visi	VisitDate	TumorS	PercentChange	TreatmentStartDate	TreatmentEndDate	Treatme	Treatme	Treatme	OverallResponse
P001	0	21-01-2025	72.45	0.000	21-01-2025	03-06-2025	19	Ongoing	Trt B	CR
P001	4	18-02-2025	62.93	-13.146	21-01-2025	03-06-2025	19	Ongoing	Trt B	CR
P001	8	18-03-2025	52.00	-28.227	21-01-2025	03-06-2025	19	Ongoing	Trt B	CR
P001	12	15-04-2025	35.00	-44.379	21-01-2025	03-06-2025	19	Ongoing	Trt B	PR
P001	16	13-05-2025	25.00	-51.923	21-01-2025	03-06-2025	19	Ongoing	Trt B	PR
P001	20	10-06-2025	10.00	-71.429	21-01-2025	03-06-2025	19	Ongoing	Trt B	PR
P001	24	08-07-2025	0.00	-100.000	21-01-2025	03-06-2025	19	Ongoing	Trt B	CR
P006	0	23-01-2025	46.69	0.000	23-01-2025	11-12-2025	46	Completed	Trt A	SD
P006	4	20-02-2025	68.13	45.935	23-01-2025	11-12-2025	46	Completed	Trt A	PD
P006	8	20-03-2025	35.60	-23.737	23-01-2025	11-12-2025	46	Completed	Trt A	SD
P006	12	17-04-2025	45.08	-3.449	23-01-2025	11-12-2025	46	Completed	Trt A	SD
P006	16	15-05-2025	67.95	45.549	23-01-2025	11-12-2025	46	Completed	Trt A	PD
P006	20	12-06-2025	76.08	62.950	23-01-2025	11-12-2025	46	Completed	Trt A	PD
P006	24	10-07-2025	67.57	44.730	23-01-2025	11-12-2025	46	Completed	Trt A	PD

Description of key variables:

Variable Name	Description
PatientID	Unique identifier for each patient
VisitWeek	Week number of the clinical visit relative to treatment start
VisitDate	Actual calendar date of the visit
TumorSize	Measured size of the tumor (in mm or cm², depending on context)
PercentChange	Percentage change in tumor size from baseline (Week 0)
TreatmentStartDate	Date when treatment began
TreatmentEndDate	Date when treatment ended or was last recorded
TreatmentDurationWeeks	Total duration of treatment in weeks
TreatmentStatus	Indicates whether treatment is ongoing or completed
Treatment	Type or arm of treatment assigned (e.g., Trt A, Trt B)
OverallResponse	Investigator-assessed response (CR, PR, SD, PD)

Table 2 (Used for Swimmer plot)

Subject ID	Treatment _Start_Date	Treatment _End_Date	Treatment _Start _Month	Treatment _End _Month	Disease _Stage	BOR	Response _Start _Date	Response _End _Date	Response _Start _Month	Response _end _month	Status
S001	23-01-2022	11-07-2024	1	30	Stage III	PR	18-04-2022	04-06-2022	2	9	Discontinued
S002	27-02-2022	08-11-2025	1	45	Stage II	SD	null	null	null	null	Discontinued
S003	13-02-2022	29-12-2022	1	35	Stage II	CR	23-06-2022	29-07-2022	4	6	Ongoing
S004	05-02-2022	20-04-2025	1	39	Stage II	CR	14-06-2022	10-08-2022	4	null	Ongoing
S005	10-01-2022	29-04-2025	1	28	Stage III	PD	null	null	null	null	Ongoing
S006	10-01-2022	24-01-2025	1	37	Stage II	SD	null	null	null	null	Discontinued
S007	04-01-2022	22-06-2024	1	30	Stage I	PD	null	null	null	null	Completed
S008	21-02-2022	10-07-2024	1	29	Stage II	CR	19-06-2022	01-08-2022	3	null	Ongoing
S009	06-02-2022	26-04-2024	1	27	Stage III	SD	null	null	null	null	Completed
S010	12-02-2022	26-06-2025	1	41	Stage I	SD	null	null	null	null	Completed

Description of key variables:

Variable Name	Description
SubjectID	Unique identifier for each patient
Treatment_Start_Date	Date when treatment began
Treatment_End_Date	Date when treatment ended
Treatment_Start_Month	Month number when treatment started
Treatment_End_Month	Month number when treatment ended

Disease_Stage	Stage of disease at baseline
BOR	Best overall response during treatment
Response_Start_Date	Date when response was first observed
Response_End_Date	Date when response ended
Response_Start_Month	Month number when response started
response_end_month	Month number when response ended
Status	Current status of the patient

Note: Synthetic data ensures privacy and simplifies visualization, it does not fully capture the complexity of real-world clinical datasets. Therefore, this example serves an illustrative purpose only, and conclusions drawn from synthetic data should not be interpreted as clinical evidence.

PYTHON LIBRARIES BEHIND THE VISUALS

Aspect	Waterfall Plot	Spider Plot	Swimmer Plot
Library / Package Used	Pandas, plotly.graph_objects, kaleido	Pandas, plotly.graph_objects, plotly.subplots	Pyspark.sql.functions, pandas, matplotlib.pyplot, seaborn
Type of Plot	Dynamic (interactive bar chart)	Dynamic (interactive line plot)	Static (annotated horizontal bars with markers and arrows)
Reason for Package Choice	Plotly for interactivity; Kaleido for export	Plotly for multi-panel and hover details	Matplotlib for layout control; Seaborn for color
Limitations for Dynamic Use	Manual layering for labels and legends	Subplot and legend customization is complex	No native support for arrows or mixed markers

Visualizations are developed using Python on the Databricks platform, leveraging its scalable Spark-based architecture for efficient data handling and interactive rendering of complex clinical plots.

WATERFALL PLOT

A waterfall plot is a type of bar chart commonly used in oncology clinical trials to display the percentage change in a clinical measurement, most often tumor size, from baseline for each patient. Each bar represents an individual patient, and the height of the bar indicates how much their tumor has grown or shrunk during treatment. The plot gets its name from its visual appearance: when bars are sorted from the largest decrease to the largest increase, they resemble a cascading waterfall. This layout makes it easy to see how many patients experienced tumor shrinkage (positive response) versus tumor growth (disease progression).

Waterfall plots are widely used in oncology trials to:

- Assess treatment efficacy at the individual patient level
- Identify responders and non-responders
- Visualize variability in patient outcomes
- Support regulatory submissions by providing detailed response data
- Complement statistical summaries with intuitive visual insights

These plots help researchers and clinicians quickly understand how well treatment is working across a diverse patient population and whether specific subgroups are benefiting more than others.
What the Waterfall Plot Displays in general

- X-axis: Each patient in the trial
- Y-axis: Percent change in tumour size (or another clinical metric) from baseline
- Bars below zero: Indicate tumour shrinkage (positive response)
- Bars above zero: Indicate tumour growth (disease progression)
- Colour coding (optional): Can be used to show response categories such as partial response, stable disease, or progression

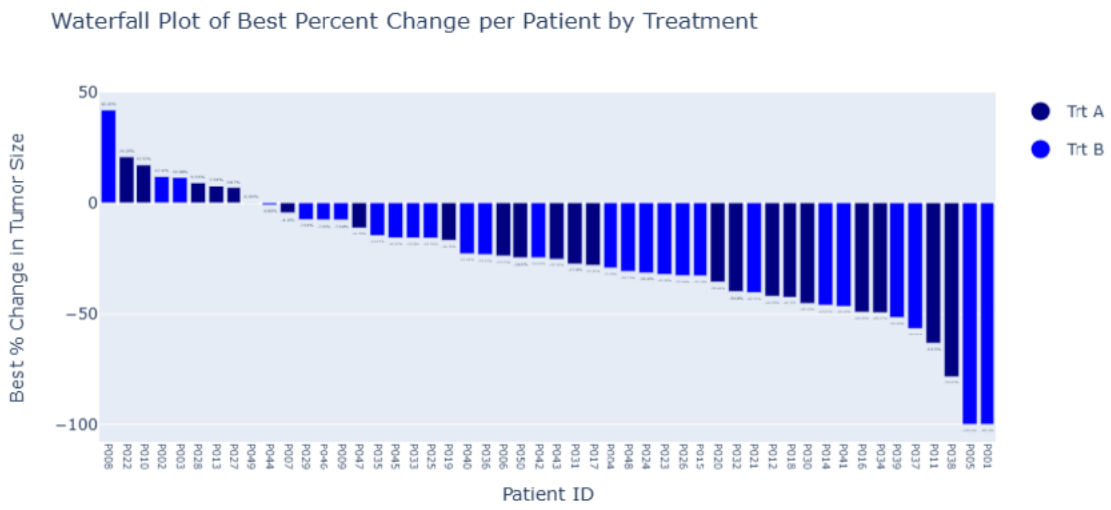
IMPLEMENTATION WITH SYNTHETIC DATA

Using synthetic data, we generated waterfall plots to visualize treatment response. The 'Best percent change' represents the maximum reduction in tumor size from baseline, calculated as the greatest decrease observed in tumor measurements over time per patient.

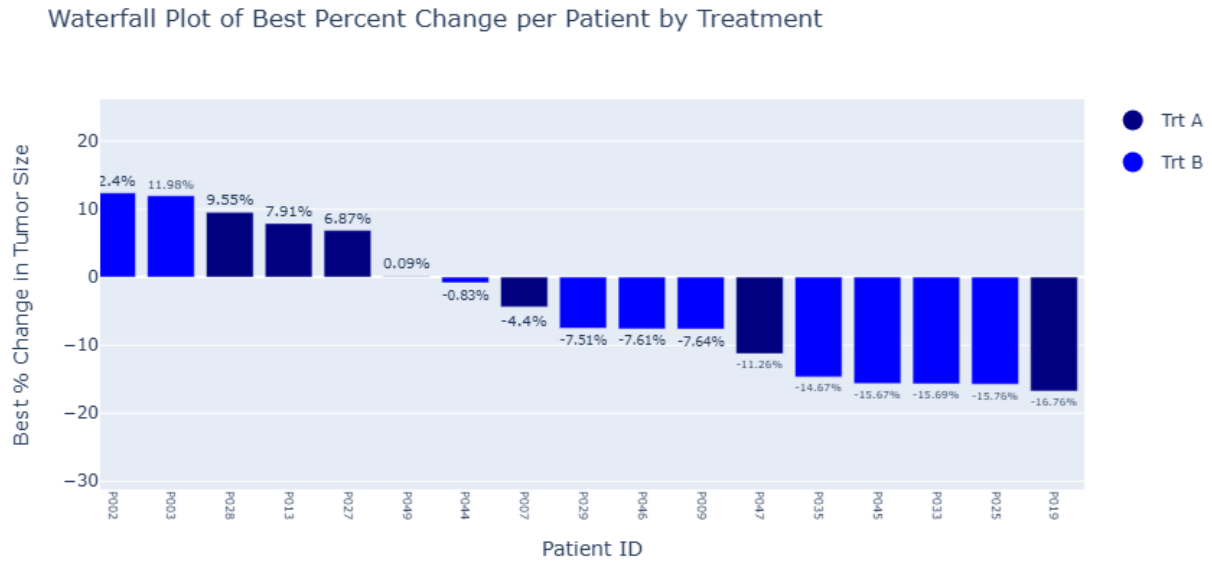
1.Waterfall plot of best percent change per patient by treatment
Python Script on Databricks platform:

```
# Databricks notebook script to generate a waterfall plot with percent change labels

# Install required packages (only needed once per cluster)
%pip install pandas==1.5.3 plotly==5.15.0 kaleido==0.2.1

# Import libraries
import pandas as pd
import plotly.graph_objects as go

# Load the dataset from the table
df = spark.table("workspace.default.oncology_dataset_final").toPandas()

# Clean column names

df.columns = df.columns.str.strip()
display(df)

# Merge BPCH with week 0 data for Treatment info
week_0 = df[df['VisitWeek'] == 0]
merged_bpch = pd.merge(bpch_df, week_0[['PatientID', 'Treatment']], on='PatientID')

# Sort by BPCH for waterfall plot in descending order
merged_bpch_sorted = merged_bpch.sort_values(by='BPCH', ascending=False)

# Define color map for Treatment
color_map = {
    'Trt A': 'navy',
    'Trt B': 'blue'
}

# Map Treatment to colors
bar_colors = merged_bpch_sorted['Treatment'].map(color_map)

# Create bar chart with BPCH
fig = go.Figure()
fig.add_trace(go.Bar(
    x=merged_bpch_sorted['PatientID'],
    y=merged_bpch_sorted['BPCH'],
    marker_color=bar_colors,
    text=merged_bpch_sorted['BPCH'].round(2).astype(str) + '%',
    textposition='outside',
    textfont=dict(size=12),
    hoverinfo='skip',
    showlegend=False
))
# Add color legends
for treatment, color in color_map.items():
    fig.add_trace(go.Scatter(
        x=[None], y=[None],
        mode='markers',
        marker=dict(size=14, color=color),
        legendgroup=treatment,
        showlegend=True,
        name=treatment
    ))
fig.update_layout(
    title='Waterfall Plot of Best Percent Change per Patient by Treatment',
    xaxis_title='Patient ID',
    yaxis_title='Best % Change in Tumor Size',
    showlegend=True,
    xaxis=dict(tickangle=90, tickfont=dict(size=8)))
fig.show()
```

Overall view:



Close view:

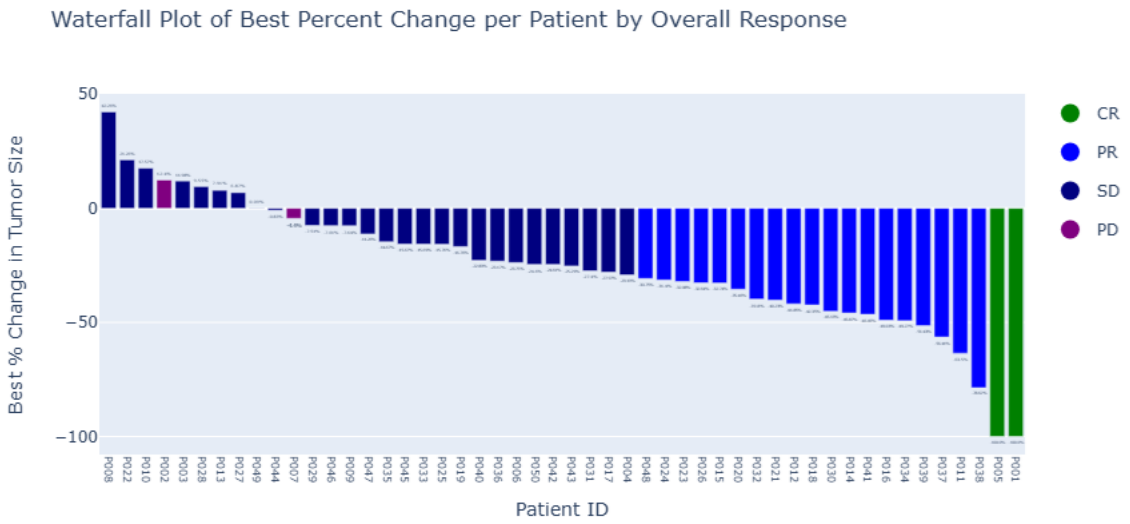


2.Waterfall plot of best percent change per patient by overall response
Python Script on Databricks platform:

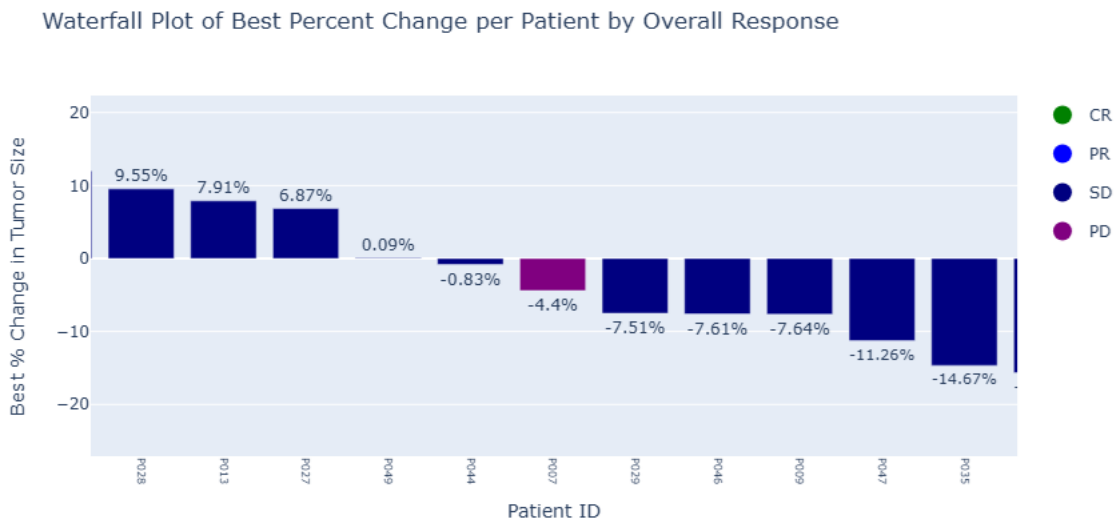
```
# Merge BPCH with week 0 data for OverallResponse info
merged_bpch = pd.merge(bpch_df, week_0[['PatientID', 'OverallResponse']], on='PatientID')

# Sort by BPCH for waterfall plot in descending order
merged_bpch_sorted = merged_bpch.sort_values(by='BPCH', ascending=False)

# Define color map for OverallResponse
response_map = {
    'CR': 'green',
    'PR': 'blue',
    'SD': 'navy',
    'PD': 'purple'}
# Map OverallResponse to colors
bar_colors = merged_bpch_sorted['OverallResponse'].map(response_map)

# Create bar chart with BPCH
fig = go.Figure()
fig.add_trace(go.Bar(
```

```
    x=merged_bpch_sorted['PatientID'],
    y=merged_bpch_sorted['BPCH'],
    marker_color=bar_colors,
    text=merged_bpch_sorted['BPCH'].round(2).astype(str) + '%',
    textposition='outside',
    textfont=dict(size=12),
    hoverinfo='skip',
    showlegend=False
))
# Add color legends for OverallResponse
for response, color in response_map.items():
    fig.add_trace(go.Scatter(
        x=[None], y=[None],
        mode='markers',
        marker=dict(size=14, color=color),
        legendgroup=response,
        showlegend=True,
        name=response
    ))

fig.update_layout(
    title='Waterfall Plot of Best Percent Change per Patient by Overall Response',
    xaxis_title='Patient ID',
    yaxis_title='Best % Change in Tumor Size',
    showlegend=True,
    xaxis=dict(tickangle=90, tickfont=dict(size=8)))
fig.show()
```

Overall view:



Close view



OBSERVATION

Two waterfall plots were generated to visualize the best percent change in tumor size from baseline for individual patients, offering complementary perspectives on treatment efficacy and clinical response. The first plot groups patients by treatment arm (Trt A and Trt B), enabling a direct comparison of therapeutic impact across cohorts. The second plot categorizes patients by overall response - Complete Response (CR), Partial Response (PR), Stable Disease (SD), and Progressive Disease (PD) - based on RECIST criteria, thereby aligning tumor shrinkage with standardized clinical outcomes. Unlike fixed-timepoint analyses, using best percent change captures each patient's maximum observed response, providing a more accurate reflection of treatment benefit. These visualizations are particularly valuable when interpreted alongside summary statistics, which in this dataset show that 2 patients achieved CR, 19 had PR, and 29 experienced SD.

Waterfall plots help bridge the gap between raw numerical data and clinical interpretation, offering intuitive insights into response heterogeneity. Response heterogeneity refers to the variation in how individual patients respond to the same treatment. In the context of these waterfall plots, it highlights that while some patients experience significant tumor shrinkage (e.g., CR or PR), others may show minimal change or even progression (SD or PD). This variability underscores the importance of personalized treatment approaches and helps identify subgroups that benefit most from a therapy. Clinically, recognizing response heterogeneity supports the development of predictive biomarkers, informs adaptive trial designs, and guides more tailored therapeutic strategies. By accounting for individual differences in treatment response, clinicians can optimize patient outcomes and improve the overall effectiveness of cancer therapies.

SPIDER PLOT

A spider plot, also known as a spaghetti plot, is a line chart commonly used in oncology and longitudinal clinical trials to visualize changes in a clinical measurement (such as tumor size or biomarker levels) over time for individual patients. Each line represents a single patient, showing their response trajectory across multiple time points during treatment. The name "spider" or "spaghetti" comes from the visual appearance of multiple overlapping lines, resembling a web or tangled strands of spaghetti. This visualization is particularly useful for understanding patterns of response, variability among patients, and identifying trends such as early responders or late progressors.

Spider plots are widely used in clinical research to:

- Track individual patient responses over time
- Identify patterns of response and progression
- Compare treatment effects across subgroups
- Complement summary statistics with detailed longitudinal data
- Support regulatory submissions by providing patient-level insights

These plots help clinicians and researchers assess whether treatment benefits are sustained, delayed, or transient, and whether certain patients deviate significantly from the overall trend.

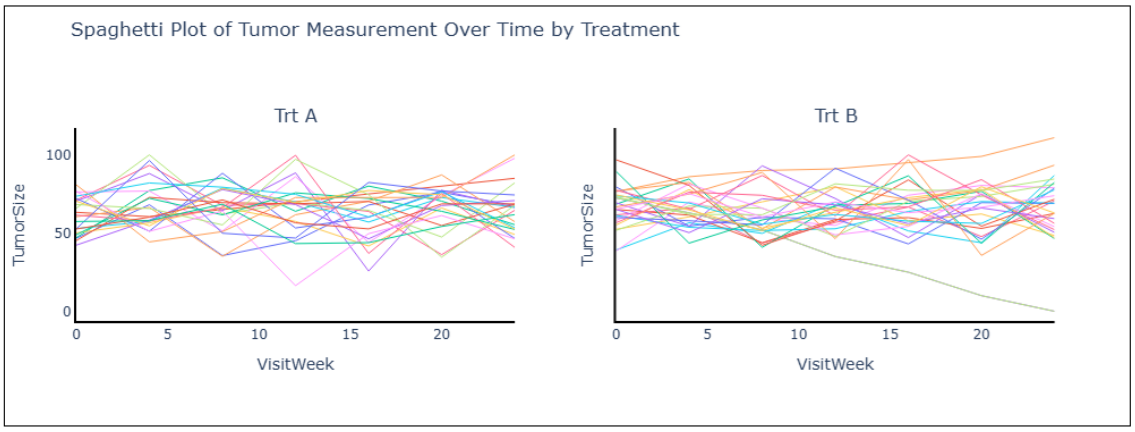
What the Spider Plot Displays

- X-axis: Time (e.g., weeks or months since treatment start)
- Y-axis: Percent change in tumor size (or another clinical metric) from baseline
- Each line: Represents an individual patient's response trajectory
- Color coding (optional): Can indicate treatment arms, response categories, or biomarker status
- Reference line at 0%: Indicates no change from baseline

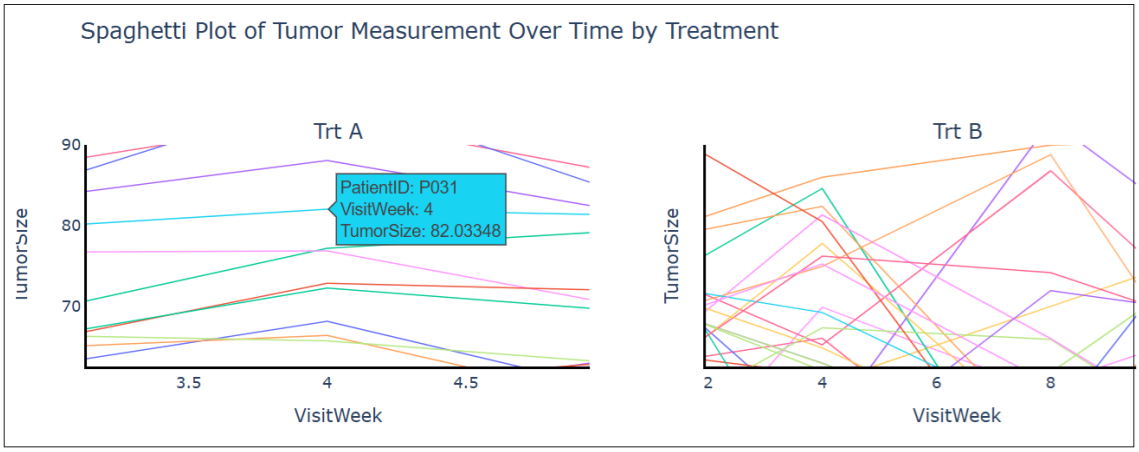
Python Script on Databricks platform:

```
import plotly.graph_objs as go
import plotly.subplots as sp

df = spark.table("workspace.default.oncology_dataset_final").select("PatientID", "VisitWeek", "TumorSize",
"Treatment", "OverallResponse")
pdf = df.toPandas()

treatments = sorted(pdf['Treatment'].dropna().unique())
n_cols = 3
n_rows = (len(treatments) + n_cols - 1) // n_cols

fig = sp.make_subplots(
    rows=n_rows,
    cols=n_cols,
    subplot_titles=[str(t) for t in treatments],
    shared_xaxes=True,
    shared_yaxes=True
)

for i, treatment in enumerate(treatments):
    row = i // n_cols + 1
    col = i % n_cols + 1
    sub_pdf = pdf[pdf['Treatment'] == treatment]
    if sub_pdf.empty:
        continue
    for pid in sub_pdf['PatientID'].dropna().unique():
        patient_data = sub_pdf[sub_pdf['PatientID'] == pid]
        if patient_data.empty:
            continue
        fig.add_trace(
            go.Scatter(
                x=patient_data['VisitWeek'],
                y=patient_data['TumorSize'],
                mode='lines',
                line=dict(width=1),
                name=str(pid),
                showlegend=False,
                customdata=patient_data['PatientID'],
                hovertemplate="PatientID: %{customdata}<br>VisitWeek: %{x}<br>TumorSize: %{y}<extra></extra>"
            ),
            row=row,
            col=col
        )

fig.update_xaxes(title_text="VisitWeek", showline=True, linecolor='black', linewidth=2)
fig.update_yaxes(title_text="TumorSize", showline=True, linecolor='black', linewidth=2)
fig.update_layout(
    height=350 * n_rows,
    width=500 * n_cols,
    title_text="Spaghetti Plot of Tumor Measurement Over Time by Treatment",
    title_y=0.98,
    plot_bgcolor='white'
)
fig.show()
```

Overall view:



Close view:



OBSERVATION

The spaghetti plot illustrates individual tumor size trajectories over time for subjects in Treatment A and Treatment B groups, highlighting substantial variability in responses. This variability reflects heterogeneous treatment effects, where patients respond differently to the same treatment due to biological, clinical, or demographic factors. While tumor size is plotted here, change from baseline can also be used to better capture relative treatment impact. Importantly, this visualization alone cannot guide clinical decisions, it must be complemented with summary statistics and inferential analyses to ensure robust conclusions. Nevertheless, the plot is essential for uncovering individual-level patterns and outliers that may be masked in aggregate data, offering valuable insights into treatment consistency and variability.

SWIMMER PLOT

A swimmer plot is a powerful visualization tool used in clinical trials to represent individual patient journeys over time. Unlike summary plots that aggregate data, swimmer plots focus on patient-level timelines, making them ideal for tracking treatment exposure, response duration, and key clinical events such as progression or dropout. Each horizontal bar in a swimmer plot represents a single patient, resembling a lane in a swimming pool—hence the name. The length of the bar indicates how long the patient remained on treatment, and additional markers or annotations can show when specific events occurred. This format allows researchers to see the full trajectory of each patient, offering insights into treatment consistency, durability of response, and variability in outcomes.

Swimmer plots are widely used in oncology trials to:

- Visualize treatment duration and timing of clinical events per patient
- Identify patterns such as early discontinuation or prolonged benefit
- Assess consistency of treatment exposure across arms
- Complement statistical summaries with detailed patient-level insights
- Support regulatory submissions by showing individual treatment journeys

These plots help researchers and clinicians understand how long patients benefit from treatment, when key events occur, and how consistent treatment effects are across the population.

What the Swimmer Plot Displays

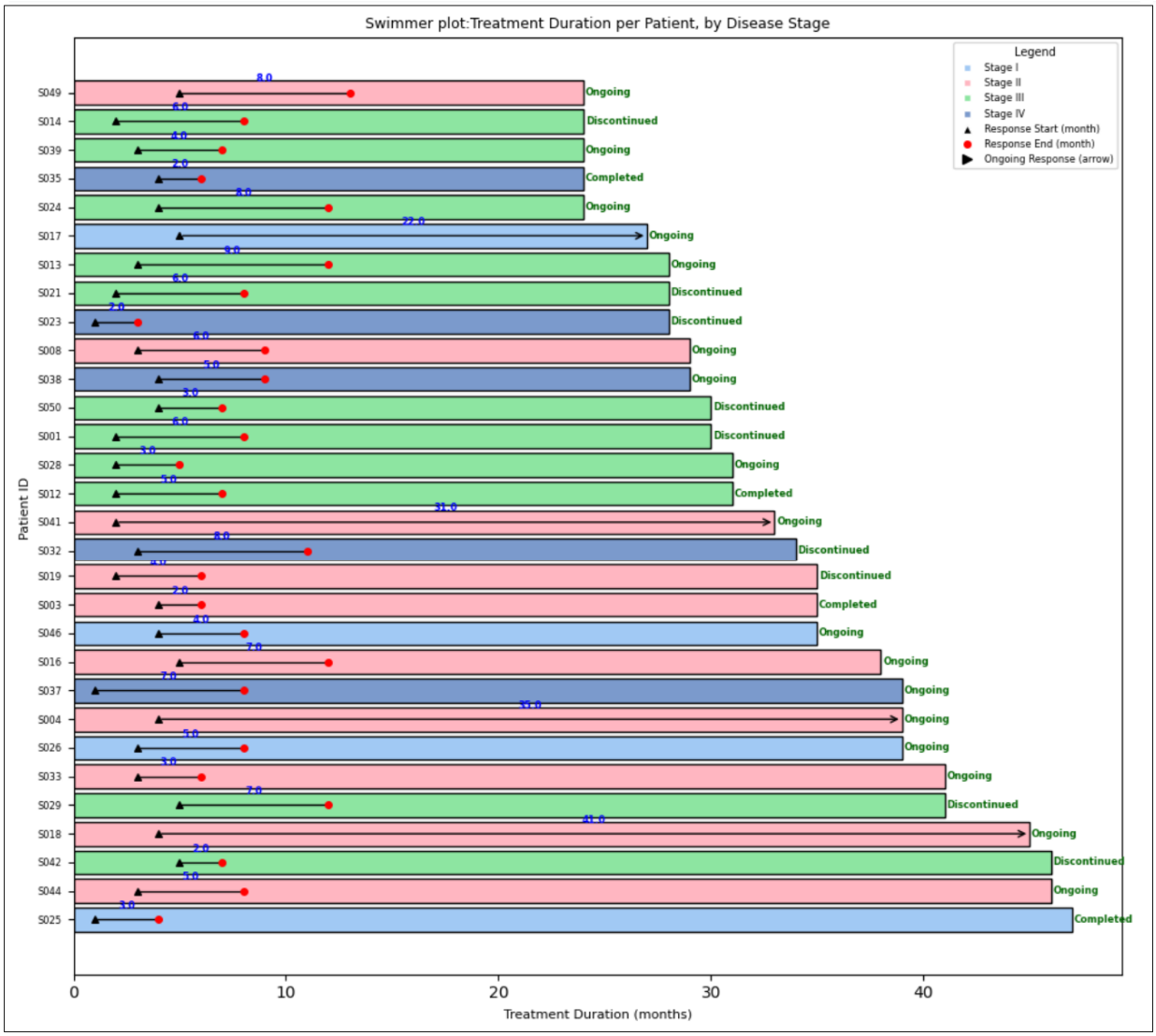
- X-axis: Time in months since treatment start (ranging approximately from 0 to 40 months).
- Y-axis: Individual patients, identified by unique Subject IDs (e.g., S001 to S025).
- Each horizontal bar: Represents a patient's treatment duration, with color coding based on disease stage:
  - Light blue = Stage I
  - Green = Stage II
  - Pink = Stage III
  - Red = Stage IV
- Black triangles: Indicate the start of treatment response (CR or PR).
- Red circles: Indicate the end of response, marking when the therapeutic effect ceased.
- Arrows: Represent ongoing responses at the time of data cutoff.
- Blue-coloured numeric annotations: Display the duration of response (in months) within each bar.
- Text labels: Indicate the current status of each patient's treatment (e.g., Ongoing, Completed, Discontinued).

Python Script on Databricks platform:

```
import matplotlib.pyplot as plt
import pandas as pd
df = spark.table("workspace.default.swimmer_plot_final")
from pyspark.sql.functions import datediff, col

# Derive trtdur as the difference in months between treatment_end_date and treatment_start_date
df_trtdur = df.withColumn(
    "trtdur",
    datediff(col("treatment_end_date"), col("treatment_start_date")) / 30
).filter(col("BOR").isin("CR", "PR"))

# Select relevant columns and convert to pandas for plotting
pdf_trtdur = df_trtdur.select(
    "subjectID", "trtdur", "disease_stage", "response_start_month", "response_end_month", "treatment_end_month",
"Status"
).toPandas()

# Sort by trtdur for better visualization
pdf_trtdur = pdf_trtdur.sort_values(by="trtdur", ascending=False).reset_index(drop=True)

# Increase bar size by increasing bar height and reducing bar spacing
bar_height = 0.5  # Increased bar height for larger bars
bar_spacing = 0.6  # Reduced spacing between bars
fig, ax = plt.subplots(figsize=(10, max(4, len(pdf_trtdur) * bar_spacing * 0.5)))

import seaborn as sns
unique_stages = pdf_trtdur["disease_stage"].unique()
palette = sns.color_palette("pastel", n_colors=len(unique_stages))
color_map = dict(zip(unique_stages, palette))

if "Stage II" in color_map:
    color_map["Stage II"] = "#ffb6c1"
if "Stage IV" in color_map:
    color_map["Stage IV"] = "#7b9acc"

colors = pdf_trtdur["disease_stage"].map(color_map)

# Adjust y positions for minimum spacing
y_positions = [i * bar_spacing for i in range(len(pdf_trtdur))]

ax.barh(y_positions, pdf_trtdur["trtdur"], color=colors, edgecolor="k", height=bar_height)

ax.set_ylabel("Patient ID", fontsize=8)
ax.set_xlabel("Treatment Duration (months)", fontsize=8)
ax.set_title("Swimmer plot:Treatment Duration per Patient, by Disease Stage", fontsize=9)
ax.set_yticks(y_positions)
ax.set_yticklabels(pdf_trtdur["subjectID"], fontsize=6)

for idx, row in pdf_trtdur.iterrows():
```

```
    y = y_positions[idx]
    if not pd.isnull(row["response_start_month"]):
        ax.scatter(
            row["response_start_month"],
            y,
            marker="^",
            color="black",
            s=16,
            label="Response Start" if idx == 0 else None,
            zorder=3
        )
        if not pd.isnull(row["response_end_month"]):
            ax.scatter(
                row["response_end_month"],
                y,
                marker="o",
                color="red",
                s=16,
                label="Response End" if idx == 0 else None,
                zorder=3
            )
            ax.plot(
                [row["response_start_month"], row["response_end_month"]],
                [y, y],
                color="black",
                linewidth=1.0,
                zorder=2
            )
            resp_dur = row["response_end_month"] - row["response_start_month"]
            if pd.notnull(resp_dur):
                ax.text(
                    (row["response_start_month"] + row["response_end_month"]) / 2,
                    y + 0.18,
                    f"{resp_dur:.1f}",
                    va="bottom",
                    ha="center",
                    fontsize=6,
                    color="blue",
                    fontweight="bold"
                )
        else:
            if not pd.isnull(row["treatment_end_month"]):
                ax.annotate(
                    "",
                    xy=(row["treatment_end_month"], y),
                    xytext=(row["response_start_month"], y),
                    arrowprops=dict(arrowstyle="->", color="black", lw=1.0),
                    va='center'
                )
                resp_dur = row["treatment_end_month"] - row["response_start_month"]
                if pd.notnull(resp_dur):
                    ax.text(
                        (row["response_start_month"] + row["treatment_end_month"]) / 2,
                        y + 0.18,
                        f"{resp_dur:.1f}",
                        va="bottom",
                        ha="center",
                        fontsize=6,
                        color="blue",
                        fontweight="bold"
                    )

for idx, row in pdf_trtdur.iterrows():
    y = y_positions[idx]
    trtdur = row["trtdur"]
    status = row.get("Status", "")
```

```
    if pd.notnull(trtdur) and status:
        ax.text(
            trtdur + 0.1,
            y,
            str(status),
            va="center",
            ha="left",
            fontsize=6,
            color="darkgreen",
            fontweight="bold"
        )

from matplotlib.lines import Line2D

stage_handles = [
    Line2D([0], [0], marker="s", color="w", markerfacecolor=color_map[stage], markersize=4, label=stage)
    for stage in unique_stages
]

marker_handles = [
    Line2D([0], [0], marker="^", color="w", markerfacecolor="black", markersize=6, label="Response Start (month)"),
    Line2D([0], [0], marker="o", color="w", markerfacecolor="red", markersize=6, label="Response End (month)"),
    Line2D([0], [0], marker=">", color="black", markerfacecolor="black", markersize=6, linestyle="None",
label="Ongoing Response (arrow)")
]

all_handles = stage_handles + marker_handles
all_labels = [h.get_label() for h in all_handles]
ax.legend(
    all_handles, all_labels, title="Legend", loc='upper right', frameon=True, fontsize=6, title_fontsize=7
)

plt.tight_layout()
fig.savefig("/tmp/swimmer_plot1.png", dpi=300, bbox_inches='tight')
display(fig)
plt.close(fig)
```



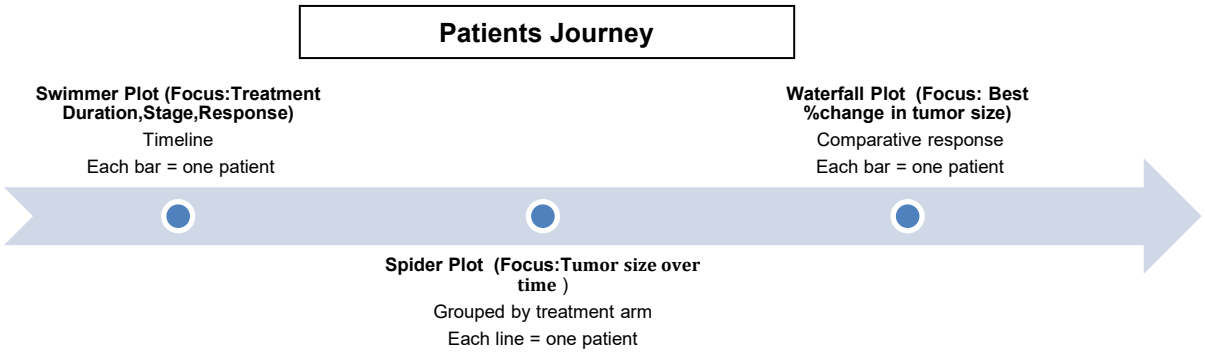
OBSERVATION

The swimmer plot provides a clear visualization of individual patient treatment durations, categorized by disease stage from Stage I to Stage IV. Each horizontal bar represents a patient, with color coding used to indicate the respective disease stage. Key clinical events are marked using distinct symbols: black triangles denote the onset of treatment response, specifically a complete response (CR) or partial response (PR), indicating the point at which a patient first showed measurable clinical improvement. Red circles represent the cessation of response, marking the end of the therapeutic effect. Arrows at the end of bars highlight ongoing responses, suggesting continued benefit or therapy at the time of data cutoff. Additionally, the duration of response is annotated within each bar using blue-coloured values (in months), helping to visually distinguish the responsive period from the overall treatment timeline.

Patients with Stage II and III disease generally exhibit longer treatment durations and more frequent ongoing responses, while Stage I patients tend to have shorter treatment periods. This stratification supports exploratory subgroup analyses, where disease stage can be considered a meaningful subgroup for evaluating treatment efficacy and response patterns. Overall, the swimmer plot effectively communicates treatment timelines, response dynamics, and disease stage-related trends across the patient cohort.

PATIENT'S JOURNEY THROUGH VISUALIZATION

In this paper, we've explored three key visualizations-Swimmer, Spider, and Waterfall plots, each offering a unique view into the patient's clinical experience. Using Python on the Databricks platform, we combined Spark-based data handling with tailored plotting libraries to bring these visuals to life. The Swimmer plot maps out each patient's treatment timeline, showing how long they stayed on therapy and when they responded. The Spider plot tracks tumor size over time, giving a dynamic view of individual patient trends. Finally, the Waterfall plot summarizes each patient's best response, offering a clear comparison across the cohort. Together, these plots help us follow the patient's journey from first dose to final outcome, capturing treatment, progression, and response in one integrated narrative.



CONCLUSION

Patient-level visualizations such as waterfall, spider, and swimmer plots offer powerful tools for uncovering individual treatment trajectories in clinical trials. These methods reveal variability, response dynamics, and outliers that aggregate summaries often mask, offering deeper insights into treatment consistency and heterogeneity. Their integration into clinical dashboards and regulatory submissions enhances interpretability and supports data-driven decision-making. The use of synthetic data and reproducible code in this paper demonstrates practical implementation strategies, making these visualizations accessible to both statisticians and clinicians. Patient-level plots will play an increasingly vital role in optimizing treatment strategies and improving patient outcomes. However, patient-level visualizations should complement, not replace aggregate summaries and inferential methods to ensure robust clinical conclusions.

ACKNOWLEDGMENTS

The author would like to express heartfelt gratitude to MARS Group Inc and PHUSE for the invaluable opportunity to contribute to this paper, as well as for their continuous support and encouragement throughout the research process.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Varsha Patil
Company: MARS Group Inc
Address: 1408 E Empire St, Bloomington, IL 61701
Phone: +919833173028
Email: Email: vdashmukhe@yahoo.co.in
Website: www.marsgroupinc.com