ChatGPT 100,000 Patient 24-Month *In Silico* Phase III 5-Arm Pancreatic Cancer Clinical Trial Triplicate

Kevin Kawchak
Chief Executive Officer
ChemicalQDevice
San Diego, CA
July 24, 2025
kevink@chemicalqdevice.com

37.S55.VIS.01.P41

Prompt 41

Opus 4 Extended: 64 Pages, July 11, 2025
ops4

Prompt 41:

Prompt for Cross-Model Meta-Verification Analysis Visualizations

You have been provided with 5 verification analysis outputs from different AI models (use these terms grk4, grk3, ops4, g25p, o3pr) that were all given the same prompt template to analyze three clinical trials for meta-verification consistency.

Analysis Summary: Provide a two-paragraph explanation of findings regarding the correspondence between the AI models' outputs. Focus on: patterns of agreement/disagreement between models in their meta-verification calculations, specific tables where models showed highest/lowest correspondence in Row Consistency Scores, systematic differences in statistical calculation approaches (mean, range, SD), and implications for AI model reliability in meta-analysis of clinical trial discrepancies. Cite visualizations 01-10 throughout the analysis summary.

Generate 10 separate visualizations in Python scripts (numbered 01-10) as follows:

1. Heatmap showing Row Consistency Scores across all models (5 models x 6 tables) with annotations for exact values and color gradient from 8.0 to 10.0

2. Grouped bar chart comparing Mean calculations for Table 2 (Baseline Characteristic Deviations) across all 5 models for each of the 5 baseline characteristics

3. Scatter plot matrix showing pairwise model agreement for all Row Consistency Scores across the 6 tables, with correlation coefficients and regression lines

4. Box plot displaying the distribution of Standard Deviation calculations across models for Table 3 (Median OS Difference) for all 5 arms

5. Radar chart comparing each model's Row Consistency Scores for all 6 meta-verification tables, with separate traces for each model

6. Line graph showing Range calculations across models for Table 6 (≥G3 AE Rate Difference) for all 5 arms, with confidence intervals

7. Parallel coordinates plot displaying how each model calculated statistics (Mean, Range, SD) for Table 4 (Median PFS Difference) across all arms

8. Stacked bar chart showing the frequency of exact agreement (difference < 0.01), minor discrepancies (0.01-0.1), and major discrepancies (>0.1) between model pairs for all calculated values

9. Bubble chart plotting Mean vs. SD calculations for Table 5 (12-Month OS Rate Difference) by arm, with bubble size representing Range and color representing model

10. Diverging bar chart highlighting the largest positive and negative deviations from the median Row Consistency Score for each table across all models, sorted by magnitude of deviation

"Begin grk4 = Grok 4" "End grk4 = Grok 4" "Begin grk3 = Grok 3 Think" "End grk3 = Grok 3 Think" "Begin ops4 = Opus 4 Extended" "End ops4 = Opus 4 Extended" "Begin g25p = Gemini 2.5 Pro" "End g25p = Gemini 2.5 Pro" "Begin o3pr = ChatGPT o3-pro" "End o3pr = ChatGPT o3-pro"

[S50-S54]

Generation 55 (Formatted)
Opus 4 Extended
23021 characters, 4 minutes 33.88 seconds

Analysis Summary

The meta-verification analysis across five AI models reveals strong overall agreement in their assessment of clinical trial discrepancies, though notable systematic differences emerge in specific calculations. All models achieved perfect consensus (10.0) for Table 1's cohort distribution discrepancy, indicating unanimous agreement when analyzing zero-variance data (visualization 01). However, divergence becomes apparent in more complex calculations, particularly in Table 3 (Median OS Difference) where grk3 produced a notably higher Row Consistency Score of 9.8 compared to 8.8-8.9 from the other four models (visualizations 01, 05). This pattern of grk3 yielding higher scores persists across

Tables 3-6, suggesting either a different interpretation of the scoring formula or more lenient variance tolerance in its calculations (visualization 03). The highest inter-model agreement occurs in Table 2's baseline characteristic deviations, where grk4, g25p, and o3pr produced identical scores across all five characteristics, while ops4 showed minor deviations and grk3 exhibited mixed results (visualization 02).

Statistical calculation approaches reveal interesting patterns of model behavior. For mean calculations, all models show excellent agreement with differences typically less than 0.01, particularly evident in Table 4's PFS difference analysis (visualization 07). Standard deviation calculations exhibit greater variability, especially in Table 3 where subtle differences in SD computation propagate to affect final consistency scores (visualization 04). Range calculations remain remarkably consistent across models, as seen in Table 6's adverse event analysis (visualization 06). The frequency of exact agreement between model pairs is highest for grk4-g25p-o3pr combinations, which form a tight cluster in most analyses, while grk3 and ops4 represent outliers in opposite directions - grk3 with systematically higher scores and ops4 with occasionally lower scores (visualizations 08, 10). These findings suggest that while AI models demonstrate strong capability for meta-analysis of clinical trial discrepancies, standardization of statistical interpretation and calculation methods remains crucial for ensuring reproducible results in regulatory and research contexts (visualization 09).

Visualization Scripts

01 - Heatmap of Row Consistency Scores

Python

```python
import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd



# Data for all Row Consistency Scores

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

tables = ['Table 1', 'Table 2-R1', 'Table 2-R2', 'Table 2-R3', 'Table 2-R4', 'Table 2-R5',

          'Table 3', 'Table 4', 'Table 5', 'Table 6']
```

```
# Create data matrix
data = np.array([
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],  # grk4
    [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],  # grk3
    [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],  # ops4
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],  # g25p
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]   # o3pr
])

# Create DataFrame
df = pd.DataFrame(data, index=models, columns=tables)

# Create figure
plt.figure(figsize=(14, 6))
ax = sns.heatmap(df, annot=True, fmt='.1f', cmap='RdYlGn',
                 vmin=8.0, vmax=10.0, cbar_kws={'label': 'Row Consistency Score'},
                 linewidths=0.5, linecolor='gray')

plt.title('Row Consistency Scores Across All Models and Tables', fontsize=16,
pad=20)
plt.xlabel('Tables', fontsize=12)
```

```
plt.ylabel('Models', fontsize=12)

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.show()
```

02 - Grouped Bar Chart for Table 2 Mean Calculations

Python

```python
import matplotlib.pyplot as plt

import numpy as np



# Data for Table 2 Mean calculations

characteristics = ['Mean Age\n(years)', 'Stage IV\n(%)', 'ECOG 1\n(%)',
'KRAS-mutant\n(%)', 'gBRCA-mutant\n(%)']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']



# Mean values from each model

data = {

    'grk4': [0.10, 0.13, 0.33, 86.03, 0.03],

    'grk3': [0.10, 0.13, 0.33, 86.03, 0.03],

    'ops4': [0.10, 0.13, 0.33, 86.03, 0.03],

    'g25p': [0.10, 0.13, 0.33, 86.03, 0.03],

    'o3pr': [0.10, 0.13, 0.33, 86.03, 0.03]
```

```
}

# Create figure with two subplots for different scales
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6), gridspec_kw={'width_ratios':
[3, 1]})

# First 4 characteristics (small values)
x = np.arange(4)
width = 0.15
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

for i, (model, color) in enumerate(zip(models, colors)):
    values = data[model][:4]
    ax1.bar(x + i*width - 2*width, values, width, label=model, color=color)

ax1.set_xlabel('Baseline Characteristics', fontsize=12)
ax1.set_ylabel('Mean Deviation Value', fontsize=12)
ax1.set_title('Table 2: Mean Calculations (Low Values)', fontsize=14)
ax1.set_xticks(x)
ax1.set_xticklabels(characteristics[:4])
ax1.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')
ax1.grid(axis='y', alpha=0.3)
```

```
# KRAS-mutant characteristic (high value)
x2 = np.arange(1)
for i, (model, color) in enumerate(zip(models, colors)):
    values = [data[model][4]]
    ax2.bar(x2 + i*width - 2*width, values, width, label=model, color=color)


ax2.set_xlabel('Baseline Characteristic', fontsize=12)
ax2.set_ylabel('Mean Deviation Value (%)', fontsize=12)
ax2.set_title('Table 2: Mean Calculations (KRAS)', fontsize=14)
ax2.set_xticks(x2)
ax2.set_xticklabels([characteristics[4]])
ax2.set_ylim(85.5, 86.5)
ax2.grid(axis='y', alpha=0.3)


plt.suptitle('Baseline Characteristic Deviations: Mean Calculations Across Models',
fontsize=16)
plt.tight_layout()
plt.show()
```

03 - Scatter Plot Matrix for Model Agreement

Python

```
import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

from scipy import stats


# Compile all Row Consistency Scores

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

scores_data = {

    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]

}


df = pd.DataFrame(scores_data)


# Create scatter plot matrix

fig, axes = plt.subplots(5, 5, figsize=(15, 15))

fig.suptitle('Pairwise Model Agreement for Row Consistency Scores', fontsize=16)
```

```
for i, model1 in enumerate(models):
    for j, model2 in enumerate(models):
        ax = axes[i, j]

        if i == j:
            # Diagonal: histogram
            ax.hist(df[model1], bins=np.arange(8.5, 10.6, 0.1), alpha=0.7, 
color='skyblue')
            ax.set_xlim(8.5, 10.5)
            if i == 0:
                ax.set_title(model1, fontsize=12, fontweight='bold')
            if j == 0:
                ax.set_ylabel(model1, fontsize=12, fontweight='bold')
        else:
            # Off-diagonal: scatter plot
            x = df[model2]
            y = df[model1]
            ax.scatter(x, y, alpha=0.6, s=50)

            # Add regression line
            slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
```

```
            line_x = np.array([x.min(), x.max()])

            line_y = slope * line_x + intercept

            ax.plot(line_x, line_y, 'r-', alpha=0.8)



            # Add correlation coefficient

            ax.text(0.05, 0.95, f'r = {r_value:.3f}', transform=ax.transAxes,

                   verticalalignment='top', bbox=dict(boxstyle='round',
facecolor='wheat', alpha=0.5))



            ax.set_xlim(8.5, 10.5)

            ax.set_ylim(8.5, 10.5)

            ax.grid(True, alpha=0.3)



            if i == 0:

                ax.set_title(model2, fontsize=12, fontweight='bold')

            if j == 0:

                ax.set_ylabel(model1, fontsize=12, fontweight='bold')



plt.tight_layout()

plt.show()
```

04 - Box Plot for SD Calculations in Table 3

Python

```python
import matplotlib.pyplot as plt

import pandas as pd

import seaborn as sns



# SD values for Table 3 (Median OS Difference) across arms

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']



# SD data from each model

sd_data = {

    'grk4': [0.10, 0.06, 0.06, 0.10, 0.10],

    'grk3': [0.10, 0.06, 0.06, 0.06, 0.10],

    'ops4': [0.10, 0.06, 0.06, 0.10, 0.10],

    'g25p': [0.10, 0.06, 0.06, 0.10, 0.10],

    'o3pr': [0.10, 0.06, 0.06, 0.10, 0.10]

}



# Prepare data for plotting

data_list = []

for arm_idx, arm in enumerate(arms):

    for model in models:
```

```
        data_list.append({
            'Arm': arm,
            'Model': model,
            'SD': sd_data[model][arm_idx]
        })

df = pd.DataFrame(data_list)

# Create box plot
plt.figure(figsize=(12, 8))
ax = sns.boxplot(data=df, x='Arm', y='SD', hue='Model', palette='Set2')

# Add individual points
sns.stripplot(data=df, x='Arm', y='SD', hue='Model', dodge=True,
              palette='Set2', size=8, alpha=0.7, ax=ax, legend=False)

plt.title('Distribution of Standard Deviation Calculations\nTable 3: Median OS Difference', fontsize=16)
plt.xlabel('Treatment Arms', fontsize=12)
plt.ylabel('Standard Deviation (months)', fontsize=12)
plt.ylim(0, 0.15)
plt.grid(axis='y', alpha=0.3)
```

```
plt.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()

plt.show()
```

05 - Radar Chart for Model Comparison

Python

```python
import matplotlib.pyplot as plt

import numpy as np

from math import pi



# Row Consistency Scores for main tables (excluding Table 2 sub-rows)

categories = ['Table 1', 'Table 2\n(avg)', 'Table 3', 'Table 4', 'Table 5', 'Table 6']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']



# Calculate average for Table 2

table2_avg = {

    'grk4': np.mean([9.9, 9.9, 9.8, 10.0, 9.9]),

    'grk3': np.mean([9.5, 9.9, 9.7, 10.0, 10.0]),

    'ops4': np.mean([9.0, 9.4, 9.8, 10.0, 9.4]),

    'g25p': np.mean([9.9, 9.9, 9.8, 10.0, 9.9]),

    'o3pr': np.mean([9.9, 9.9, 9.8, 10.0, 9.9])
```

```
}

# Data for each model
data = {
    'grk4': [10.0, table2_avg['grk4'], 8.8, 9.3, 9.4, 9.5],
    'grk3': [10.0, table2_avg['grk3'], 9.8, 9.9, 9.9, 9.8],
    'ops4': [10.0, table2_avg['ops4'], 8.9, 9.3, 9.5, 9.6],
    'g25p': [10.0, table2_avg['g25p'], 8.8, 9.3, 9.4, 9.5],
    'o3pr': [10.0, table2_avg['o3pr'], 8.8, 9.3, 9.4, 9.5]
}

# Number of variables
num_vars = len(categories)

# Compute angle for each axis
angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)]
angles += angles[:1]

# Initialize plot
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'))
```

```
# Plot data for each model
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
for i, (model, color) in enumerate(zip(models, colors)):
    values = data[model]
    values += values[:1]
    ax.plot(angles, values, 'o-', linewidth=2, label=model, color=color)
    ax.fill(angles, values, alpha=0.1, color=color)

# Fix axis to go in the right order and start at 12 o'clock
ax.set_theta_offset(pi / 2)
ax.set_theta_direction(-1)

# Draw axis lines for each angle and label
ax.set_xticks(angles[:-1])
ax.set_xticklabels(categories, size=12)

# Set y-axis limits and labels
ax.set_ylim(8.5, 10.2)
ax.set_yticks([8.5, 9.0, 9.5, 10.0])
ax.set_yticklabels(['8.5', '9.0', '9.5', '10.0'])
ax.grid(True)
```

```
# Add title and legend

plt.title('Row Consistency Scores Comparison Across Models', size=16, y=1.08)

plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1.1))

plt.tight_layout()

plt.show()
```

06 - Line Graph for Range Calculations in Table 6

Python

```python
import matplotlib.pyplot as plt

import numpy as np



# Range values for Table 6 (≥G3 AE Rate Difference)

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']



# Range data from each model

range_data = {

    'grk4': [0.30, 0.50, 1.30, 1.20, 0.40],

    'grk3': [0.30, 0.50, 1.30, 1.20, 0.30],

    'ops4': [0.30, 0.50, 1.30, 1.20, 0.40],

    'g25p': [0.30, 0.50, 1.30, 1.20, 0.40],
```

```
    'o3pr': [0.30, 0.50, 1.30, 1.20, 0.40]
}


# Calculate mean and std for confidence intervals
x = np.arange(len(arms))
means = []
stds = []


for i in range(len(arms)):
    arm_values = [range_data[model][i] for model in models]
    means.append(np.mean(arm_values))
    stds.append(np.std(arm_values))


# Create figure
plt.figure(figsize=(12, 8))


# Plot lines for each model
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
for model, color in zip(models, colors):
    plt.plot(x, range_data[model], 'o-', label=model, color=color,
             linewidth=2, markersize=8, alpha=0.8)
```

```
# Add confidence interval for mean
plt.fill_between(x, np.array(means) - 1.96*np.array(stds),
                 np.array(means) + 1.96*np.array(stds),
                 alpha=0.2, color='gray', label='95% CI')


plt.xlabel('Treatment Arms', fontsize=12)
plt.ylabel('Range of ≥G3 AE Rate Difference (%)', fontsize=12)
plt.title('Table 6: Range Calculations Across Models with Confidence Intervals',
fontsize=16)
plt.xticks(x, arms)
plt.grid(True, alpha=0.3)
plt.legend(loc='upper left', title='Models')
plt.tight_layout()
plt.show()
```

07 - Parallel Coordinates for Table 4 Statistics

Python

```python
import matplotlib.pyplot as plt
import pandas as pd
from pandas.plotting import parallel_coordinates
```

```
# Data for Table 4 (Median PFS Difference) - all statistics
arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']
models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']


# Create data list
data_list = []
for model in models:
    # Mean values
    means = {'grk4': [0.03, 0.07, 0.00, -0.03, 0.10],
             'grk3': [0.03, 0.07, 0.00, -0.03, 0.10],
             'ops4': [0.03, 0.07, 0.00, -0.03, 0.10],
             'g25p': [0.03, 0.07, 0.00, -0.03, 0.10],
             'o3pr': [0.03, 0.07, 0.00, -0.03, 0.10]}


    # Range values
    ranges = {'grk4': [0.10, 0.10, 0.00, 0.10, 0.20],
              'grk3': [0.10, 0.10, 0.00, 0.10, 0.20],
              'ops4': [0.10, 0.10, 0.00, 0.10, 0.20],
              'g25p': [0.10, 0.10, 0.00, 0.10, 0.20],
              'o3pr': [0.10, 0.10, 0.00, 0.10, 0.20]}
```

```
    # SD values
    sds = {'grk4': [0.06, 0.06, 0.00, 0.06, 0.10],
           'grk3': [0.06, 0.06, 0.00, 0.06, 0.10],
           'ops4': [0.06, 0.06, 0.00, 0.06, 0.10],
           'g25p': [0.06, 0.06, 0.00, 0.06, 0.10],
           'o3pr': [0.06, 0.06, 0.00, 0.06, 0.10]}


    for i, arm in enumerate(arms):
        data_list.append({
            'Model': model,
            f'{arm}\nMean': means[model][i],
            f'{arm}\nRange': ranges[model][i],
            f'{arm}\nSD': sds[model][i]
        })


# Create DataFrame with alternating statistics
columns = []
for arm in arms:
    columns.extend([f'{arm}\nMean', f'{arm}\nRange', f'{arm}\nSD'])


df_plot = pd.DataFrame(columns=['Model'] + columns)
```

```
for model in models:

    row_data = {'Model': model}

    for i, arm in enumerate(arms):

        row_data[f'{arm}\nMean'] = means[model][i]

        row_data[f'{arm}\nRange'] = ranges[model][i]

        row_data[f'{arm}\nSD'] = sds[model][i]

    df_plot = pd.concat([df_plot, pd.DataFrame([row_data])], ignore_index=True)



# Create parallel coordinates plot

plt.figure(figsize=(16, 8))

parallel_coordinates(df_plot, 'Model', colormap='tab10', alpha=0.7, linewidth=2)



plt.title('Table 4: Parallel Coordinates of Statistical Calculations Across Arms',
fontsize=16)

plt.xlabel('Statistics by Arm', fontsize=12)

plt.ylabel('Value (months)', fontsize=12)

plt.xticks(rotation=90)

plt.grid(True, alpha=0.3)

plt.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()

plt.show()
```

08 - Stacked Bar Chart for Agreement Frequency

Python

```python
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd



# Calculate pairwise differences for all values

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

model_pairs = []

exact_agreement = []

minor_discrepancy = []

major_discrepancy = []



# All Row Consistency Scores

all_scores = {

    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]

}
```

```
# Calculate for each model pair
for i in range(len(models)):
    for j in range(i+1, len(models)):
        model1, model2 = models[i], models[j]
        model_pairs.append(f'{model1}-{model2}')

        # Calculate differences
        diffs = [abs(all_scores[model1][k] - all_scores[model2][k]) for k in
range(len(all_scores[model1]))]

        exact = sum(1 for d in diffs if d < 0.01)
        minor = sum(1 for d in diffs if 0.01 <= d <= 0.1)
        major = sum(1 for d in diffs if d > 0.1)

        exact_agreement.append(exact)
        minor_discrepancy.append(minor)
        major_discrepancy.append(major)

# Create stacked bar chart
fig, ax = plt.subplots(figsize=(12, 8))

x = np.arange(len(model_pairs))
```

```
width = 0.6


p1 = ax.bar(x, exact_agreement, width, label='Exact Agreement (<0.01)',
color='#2ca02c')

p2 = ax.bar(x, minor_discrepancy, width, bottom=exact_agreement,

            label='Minor Discrepancy (0.01-0.1)', color='#ff7f0e')

p3 = ax.bar(x, major_discrepancy, width,

            bottom=np.array(exact_agreement) + np.array(minor_discrepancy),

            label='Major Discrepancy (>0.1)', color='#d62728')


# Add value labels on bars

for i, (e, m, M) in enumerate(zip(exact_agreement, minor_discrepancy,
major_discrepancy)):

    if e > 0:

        ax.text(i, e/2, str(e), ha='center', va='center', fontweight='bold')

    if m > 0:

        ax.text(i, e + m/2, str(m), ha='center', va='center', fontweight='bold')

    if M > 0:

        ax.text(i, e + m + M/2, str(M), ha='center', va='center', fontweight='bold')


ax.set_xlabel('Model Pairs', fontsize=12)

ax.set_ylabel('Number of Values', fontsize=12)
```

```
ax.set_title('Frequency of Agreement Between Model Pairs Across All Calculated
Values', fontsize=16)

ax.set_xticks(x)

ax.set_xticklabels(model_pairs, rotation=45, ha='right')

ax.legend(loc='upper right')

ax.set_ylim(0, 12)

plt.grid(axis='y', alpha=0.3)

plt.tight_layout()

plt.show()
```

09 - Bubble Chart for Table 5 Statistics

Python

```python
import matplotlib.pyplot as plt

import numpy as np



# Data for Table 5 (12-Month OS Rate Difference)

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

colors = {'grk4': '#1f77b4', 'grk3': '#ff7f0e', 'ops4': '#2ca02c',

          'g25p': '#d62728', 'o3pr': '#9467bd'}



# Statistics data
```

```
mean_data = {
    'grk4': [0.73, -0.47, -0.67, -0.07, 0.07],
    'grk3': [0.73, -0.47, -0.67, -0.07, 0.07],
    'ops4': [0.73, -0.47, -0.67, -0.07, 0.07],
    'g25p': [0.73, -0.47, -0.67, -0.07, 0.07],
    'o3pr': [0.73, -0.47, -0.67, -0.07, 0.07]
}

sd_data = {
    'grk4': [0.38, 0.38, 0.57, 0.29, 0.32],
    'grk3': [0.38, 0.38, 0.55, 0.31, 0.29],
    'ops4': [0.38, 0.38, 0.57, 0.29, 0.32],
    'g25p': [0.38, 0.38, 0.57, 0.29, 0.32],
    'o3pr': [0.38, 0.38, 0.57, 0.29, 0.32]
}

range_data = {
    'grk4': [0.70, 0.70, 1.10, 0.50, 0.60],
    'grk3': [0.70, 0.60, 1.10, 0.50, 0.50],
    'ops4': [0.70, 0.70, 1.10, 0.50, 0.60],
    'g25p': [0.70, 0.70, 1.10, 0.50, 0.60],
```

```
    'o3pr': [0.70, 0.70, 1.10, 0.50, 0.60]
}

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

# Plot bubbles for each model and arm
for model in models:
    for i, arm in enumerate(arms):
        x = mean_data[model][i]
        y = sd_data[model][i]
        size = range_data[model][i] * 500  # Scale for visibility

        # Add slight jitter to avoid overlap
        x_jitter = x + np.random.normal(0, 0.01)
        y_jitter = y + np.random.normal(0, 0.005)

        ax.scatter(x_jitter, y_jitter, s=size, c=colors[model], alpha=0.6,
                   edgecolors='black', linewidth=1.5)

# Add legend for models
```

```python
for model in models:
    ax.scatter([], [], c=colors[model], s=200, label=model, alpha=0.6,
               edgecolors='black', linewidth=1.5)

# Add size legend
sizes = [0.5, 0.75, 1.0]
for size in sizes:
    ax.scatter([], [], c='gray', s=size*500, alpha=0.6,
               edgecolors='black', linewidth=1.5,
               label=f'Range = {size}')

ax.set_xlabel('Mean 12-Month OS Rate Difference (%)', fontsize=12)
ax.set_ylabel('Standard Deviation (%)', fontsize=12)
ax.set_title('Table 5: Mean vs SD with Bubble Size Representing Range', fontsize=16)
ax.grid(True, alpha=0.3)

# Create two legends
legend1 = ax.legend(models, title='Models', loc='upper left', bbox_to_anchor=(0, 1))
ax.add_artist(legend1)
ax.legend(handles=ax.collections[-3:], labels=[f'Range = {s}' for s in sizes],
         title='Bubble Size', loc='lower right')
```

```
# Add arm labels
for i, arm in enumerate(arms):
    x_mean = np.mean([mean_data[model][i] for model in models])
    y_mean = np.mean([sd_data[model][i] for model in models])
    ax.annotate(arm, (x_mean, y_mean), fontsize=10, fontweight='bold',
                ha='center', va='center',
                bbox=dict(boxstyle='round,pad=0.3', facecolor='yellow', alpha=0.3))

plt.tight_layout()
plt.show()
```

10 - Diverging Bar Chart for Deviations from Median

Python

```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

# Calculate median Row Consistency Score for each table
tables = ['Table 1', 'Table 2-R1', 'Table 2-R2', 'Table 2-R3', 'Table 2-R4',
          'Table 2-R5', 'Table 3', 'Table 4', 'Table 5', 'Table 6']
```

```
# All scores
scores = {
    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],
    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],
    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],
    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],
    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]
}


# Calculate deviations
deviations_data = []
for i, table in enumerate(tables):
    table_scores = [scores[model][i] for model in scores.keys()]
    median_score = np.median(table_scores)

    for model in scores.keys():
        deviation = scores[model][i] - median_score
        if abs(deviation) > 0.001:  # Only include non-zero deviations
            deviations_data.append({
                'Table-Model': f'{table} ({model})',
                'Deviation': deviation,
```

```
                'Table': table,
                'Model': model
            })

# Sort by absolute deviation
df = pd.DataFrame(deviations_data)
df['Abs_Deviation'] = abs(df['Deviation'])
df = df.sort_values('Abs_Deviation', ascending=True)

# Create figure
fig, ax = plt.subplots(figsize=(12, 10))

# Create color mapping
colors = df['Deviation'].apply(lambda x: '#2ca02c' if x > 0 else '#d62728')

# Create horizontal bar chart
y_pos = np.arange(len(df))
bars = ax.barh(y_pos, df['Deviation'], color=colors, alpha=0.7, edgecolor='black')

# Add value labels
for i, (idx, row) in enumerate(df.iterrows()):
```

```
    value = row['Deviation']
    if value > 0:
        ha = 'left'
        x_offset = 0.01
    else:
        ha = 'right'
        x_offset = -0.01
    ax.text(value + x_offset, i, f'{value:+.2f}', ha=ha, va='center',
fontweight='bold')


# Customize plot
ax.set_yticks(y_pos)
ax.set_yticklabels(df['Table-Model'])
ax.set_xlabel('Deviation from Median Row Consistency Score', fontsize=12)
ax.set_title('Largest Deviations from Median Scores by Table and Model',
fontsize=16)
ax.axvline(x=0, color='black', linewidth=1, alpha=0.5)
ax.grid(axis='x', alpha=0.3)


# Add legend
ax.plot([], [], 'g-', linewidth=10, alpha=0.7, label='Positive Deviation')
ax.plot([], [], 'r-', linewidth=10, alpha=0.7, label='Negative Deviation')
```

```
ax.legend(loc='lower right')



plt.tight_layout()

plt.show()
```

Generation 55 (Plain Text)

Opus 4 Extended

23187 characters, 4 minutes 33.88 seconds

Analysis Summary

The meta-verification analysis across five AI models reveals strong overall agreement in their assessment of clinical trial discrepancies, though notable systematic differences emerge in specific calculations. All models achieved perfect consensus (10.0) for Table 1's cohort distribution discrepancy, indicating unanimous agreement when analyzing zero-variance data (visualization 01). However, divergence becomes apparent in more complex calculations, particularly in Table 3 (Median OS Difference) where grk3 produced a notably higher Row Consistency Score of 9.8 compared to 8.8-8.9 from the other four models (visualizations 01, 05). This pattern of grk3 yielding higher scores persists across Tables 3-6, suggesting either a different interpretation of the scoring formula or more lenient variance tolerance in its calculations (visualization 03). The highest inter-model agreement occurs in Table 2's baseline characteristic deviations, where grk4, g25p, and o3pr produced identical scores across all five characteristics, while ops4 showed minor deviations and grk3 exhibited mixed results (visualization 02).

Statistical calculation approaches reveal interesting patterns of model behavior. For mean calculations, all models show excellent agreement with differences typically less than 0.01, particularly evident in Table 4's PFS difference analysis (visualization 07). Standard deviation calculations exhibit greater variability, especially in Table 3 where subtle differences in SD computation propagate to affect final consistency scores (visualization 04). Range calculations remain remarkably consistent across models, as seen in Table 6's adverse event analysis (visualization 06). The frequency of exact agreement between model pairs is highest for grk4-g25p-o3pr combinations, which form a tight cluster in most analyses, while grk3 and ops4 represent outliers in opposite directions - grk3 with systematically higher scores and ops4 with occasionally lower scores (visualizations 08, 10). These findings suggest that while AI models demonstrate strong capability for meta-analysis of clinical trial discrepancies, standardization of statistical interpretation and calculation methods remains crucial for ensuring reproducible results in regulatory and research contexts (visualization 09).

Visualization Scripts

01 - Heatmap of Row Consistency Scores

```python
import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np

import pandas as pd


# Data for all Row Consistency Scores

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

tables = ['Table 1', 'Table 2-R1', 'Table 2-R2', 'Table 2-R3', 'Table 2-R4', 'Table 2-R5',

          'Table 3', 'Table 4', 'Table 5', 'Table 6']


# Create data matrix

data = np.array([
```

```
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],  # grk4
    [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],  # grk3
    [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],  # ops4
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],  # g25p
    [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]   # o3pr
])

# Create DataFrame
df = pd.DataFrame(data, index=models, columns=tables)

# Create figure
plt.figure(figsize=(14, 6))
ax = sns.heatmap(df, annot=True, fmt='.1f', cmap='RdYlGn',
                 vmin=8.0, vmax=10.0, cbar_kws={'label': 'Row Consistency Score'},
                 linewidths=0.5, linecolor='gray')

plt.title('Row Consistency Scores Across All Models and Tables', fontsize=16, pad=20)
plt.xlabel('Tables', fontsize=12)
plt.ylabel('Models', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

02 - Grouped Bar Chart for Table 2 Mean Calculations

```python
import matplotlib.pyplot as plt
import numpy as np

# Data for Table 2 Mean calculations
characteristics = ['Mean Age\n(years)', 'Stage IV\n(%)', 'ECOG 1\n(%)', 'KRAS-mutant\n(%)', 'gBRCA-mutant\n(%)']
models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

# Mean values from each model
data = {
    'grk4': [0.10, 0.13, 0.33, 86.03, 0.03],
    'grk3': [0.10, 0.13, 0.33, 86.03, 0.03],
    'ops4': [0.10, 0.13, 0.33, 86.03, 0.03],
    'g25p': [0.10, 0.13, 0.33, 86.03, 0.03],
    'o3pr': [0.10, 0.13, 0.33, 86.03, 0.03]
}

# Create figure with two subplots for different scales
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 6), gridspec_kw={'width_ratios': [3, 1]})
```

```
# First 4 characteristics (small values)

x = np.arange(4)

width = 0.15

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']



for i, (model, color) in enumerate(zip(models, colors)):

    values = data[model][:4]

    ax1.bar(x + i*width - 2*width, values, width, label=model, color=color)



ax1.set_xlabel('Baseline Characteristics', fontsize=12)

ax1.set_ylabel('Mean Deviation Value', fontsize=12)

ax1.set_title('Table 2: Mean Calculations (Low Values)', fontsize=14)

ax1.set_xticks(x)

ax1.set_xticklabels(characteristics[:4])

ax1.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')

ax1.grid(axis='y', alpha=0.3)



# KRAS-mutant characteristic (high value)

x2 = np.arange(1)

for i, (model, color) in enumerate(zip(models, colors)):

    values = [data[model][4]]

    ax2.bar(x2 + i*width - 2*width, values, width, label=model, color=color)
```

```
ax2.set_xlabel('Baseline Characteristic', fontsize=12)

ax2.set_ylabel('Mean Deviation Value (%)', fontsize=12)

ax2.set_title('Table 2: Mean Calculations (KRAS)', fontsize=14)

ax2.set_xticks(x2)

ax2.set_xticklabels([characteristics[4]])

ax2.set_ylim(85.5, 86.5)

ax2.grid(axis='y', alpha=0.3)


plt.suptitle('Baseline Characteristic Deviations: Mean Calculations Across Models', fontsize=16)

plt.tight_layout()

plt.show()
```

03 - Scatter Plot Matrix for Model Agreement

```python
import matplotlib.pyplot as plt

import seaborn as sns

import pandas as pd

import numpy as np

from scipy import stats


# Compile all Row Consistency Scores
```

```
models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

scores_data = {

    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]

}



df = pd.DataFrame(scores_data)



# Create scatter plot matrix

fig, axes = plt.subplots(5, 5, figsize=(15, 15))

fig.suptitle('Pairwise Model Agreement for Row Consistency Scores', fontsize=16)



for i, model1 in enumerate(models):

    for j, model2 in enumerate(models):

        ax = axes[i, j]



        if i == j:

            # Diagonal: histogram

            ax.hist(df[model1], bins=np.arange(8.5, 10.6, 0.1), alpha=0.7, color='skyblue')

            ax.set_xlim(8.5, 10.5)
```

```
    if i == 0:
        ax.set_title(model1, fontsize=12, fontweight='bold')
    if j == 0:
        ax.set_ylabel(model1, fontsize=12, fontweight='bold')
else:
    # Off-diagonal: scatter plot
    x = df[model2]
    y = df[model1]
    ax.scatter(x, y, alpha=0.6, s=50)

    # Add regression line
    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)
    line_x = np.array([x.min(), x.max()])
    line_y = slope * line_x + intercept
    ax.plot(line_x, line_y, 'r-', alpha=0.8)

    # Add correlation coefficient
    ax.text(0.05, 0.95, f'r = {r_value:.3f}', transform=ax.transAxes,
          verticalalignment='top', bbox=dict(boxstyle='round', facecolor='wheat', alpha=0.5))

    ax.set_xlim(8.5, 10.5)
    ax.set_ylim(8.5, 10.5)
    ax.grid(True, alpha=0.3)
```

```
        if i == 0:
            ax.set_title(model2, fontsize=12, fontweight='bold')
        if j == 0:
            ax.set_ylabel(model1, fontsize=12, fontweight='bold')

plt.tight_layout()
plt.show()
```

04 - Box Plot for SD Calculations in Table 3

```python
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# SD values for Table 3 (Median OS Difference) across arms
arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']
models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

# SD data from each model
sd_data = {
```

```python
    'grk4': [0.10, 0.06, 0.06, 0.10, 0.10],

    'grk3': [0.10, 0.06, 0.06, 0.06, 0.10],

    'ops4': [0.10, 0.06, 0.06, 0.10, 0.10],

    'g25p': [0.10, 0.06, 0.06, 0.10, 0.10],

    'o3pr': [0.10, 0.06, 0.06, 0.10, 0.10]

}


# Prepare data for plotting

data_list = []

for arm_idx, arm in enumerate(arms):

    for model in models:

        data_list.append({

            'Arm': arm,

            'Model': model,

            'SD': sd_data[model][arm_idx]

        })


df = pd.DataFrame(data_list)


# Create box plot

plt.figure(figsize=(12, 8))

ax = sns.boxplot(data=df, x='Arm', y='SD', hue='Model', palette='Set2')
```

```python
# Add individual points

sns.stripplot(data=df, x='Arm', y='SD', hue='Model', dodge=True,

              palette='Set2', size=8, alpha=0.7, ax=ax, legend=False)



plt.title('Distribution of Standard Deviation Calculations\nTable 3: Median OS Difference', fontsize=16)

plt.xlabel('Treatment Arms', fontsize=12)

plt.ylabel('Standard Deviation (months)', fontsize=12)

plt.ylim(0, 0.15)

plt.grid(axis='y', alpha=0.3)

plt.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()

plt.show()
```

05 - Radar Chart for Model Comparison

```python
import matplotlib.pyplot as plt

import numpy as np

from math import pi



# Row Consistency Scores for main tables (excluding Table 2 sub-rows)

categories = ['Table 1', 'Table 2\n(avg)', 'Table 3', 'Table 4', 'Table 5', 'Table 6']
```

```
models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']


# Calculate average for Table 2

table2_avg = {

    'grk4': np.mean([9.9, 9.9, 9.8, 10.0, 9.9]),

    'grk3': np.mean([9.5, 9.9, 9.7, 10.0, 10.0]),

    'ops4': np.mean([9.0, 9.4, 9.8, 10.0, 9.4]),

    'g25p': np.mean([9.9, 9.9, 9.8, 10.0, 9.9]),

    'o3pr': np.mean([9.9, 9.9, 9.8, 10.0, 9.9])

}


# Data for each model

data = {

    'grk4': [10.0, table2_avg['grk4'], 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, table2_avg['grk3'], 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, table2_avg['ops4'], 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, table2_avg['g25p'], 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, table2_avg['o3pr'], 8.8, 9.3, 9.4, 9.5]

}


# Number of variables

num_vars = len(categories)
```

```
# Compute angle for each axis

angles = [n / float(num_vars) * 2 * pi for n in range(num_vars)]

angles += angles[:1]



# Initialize plot

fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'))



# Plot data for each model

colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']

for i, (model, color) in enumerate(zip(models, colors)):

    values = data[model]

    values += values[:1]

    ax.plot(angles, values, 'o-', linewidth=2, label=model, color=color)

    ax.fill(angles, values, alpha=0.1, color=color)



# Fix axis to go in the right order and start at 12 o'clock

ax.set_theta_offset(pi / 2)

ax.set_theta_direction(-1)



# Draw axis lines for each angle and label

ax.set_xticks(angles[:-1])

ax.set_xticklabels(categories, size=12)
```

```
# Set y-axis limits and labels

ax.set_ylim(8.5, 10.2)

ax.set_yticks([8.5, 9.0, 9.5, 10.0])

ax.set_yticklabels(['8.5', '9.0', '9.5', '10.0'])

ax.grid(True)


# Add title and legend

plt.title('Row Consistency Scores Comparison Across Models', size=16, y=1.08)

plt.legend(loc='upper right', bbox_to_anchor=(1.2, 1.1))

plt.tight_layout()

plt.show()
```

06 - Line Graph for Range Calculations in Table 6

```python

import matplotlib.pyplot as plt

import numpy as np


# Range values for Table 6 (≥G3 AE Rate Difference)

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']
```

```
# Range data from each model
range_data = {
    'grk4': [0.30, 0.50, 1.30, 1.20, 0.40],
    'grk3': [0.30, 0.50, 1.30, 1.20, 0.30],
    'ops4': [0.30, 0.50, 1.30, 1.20, 0.40],
    'g25p': [0.30, 0.50, 1.30, 1.20, 0.40],
    'o3pr': [0.30, 0.50, 1.30, 1.20, 0.40]
}


# Calculate mean and std for confidence intervals
x = np.arange(len(arms))
means = []
stds = []


for i in range(len(arms)):
    arm_values = [range_data[model][i] for model in models]
    means.append(np.mean(arm_values))
    stds.append(np.std(arm_values))


# Create figure
plt.figure(figsize=(12, 8))


# Plot lines for each model
```

```
colors = ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728', '#9467bd']
for model, color in zip(models, colors):
    plt.plot(x, range_data[model], 'o-', label=model, color=color,
             linewidth=2, markersize=8, alpha=0.8)

# Add confidence interval for mean
plt.fill_between(x, np.array(means) - 1.96*np.array(stds),
                 np.array(means) + 1.96*np.array(stds),
                 alpha=0.2, color='gray', label='95% CI')

plt.xlabel('Treatment Arms', fontsize=12)
plt.ylabel('Range of ≥G3 AE Rate Difference (%)', fontsize=12)
plt.title('Table 6: Range Calculations Across Models with Confidence Intervals', fontsize=16)
plt.xticks(x, arms)
plt.grid(True, alpha=0.3)
plt.legend(loc='upper left', title='Models')
plt.tight_layout()
plt.show()
```

07 - Parallel Coordinates for Table 4 Statistics

```python
```

```
import matplotlib.pyplot as plt

import pandas as pd

from pandas.plotting import parallel_coordinates



# Data for Table 4 (Median PFS Difference) - all statistics

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']



# Create data list

data_list = []

for model in models:

    # Mean values

    means = {'grk4': [0.03, 0.07, 0.00, -0.03, 0.10],

             'grk3': [0.03, 0.07, 0.00, -0.03, 0.10],

             'ops4': [0.03, 0.07, 0.00, -0.03, 0.10],

             'g25p': [0.03, 0.07, 0.00, -0.03, 0.10],

             'o3pr': [0.03, 0.07, 0.00, -0.03, 0.10]}



    # Range values

    ranges = {'grk4': [0.10, 0.10, 0.00, 0.10, 0.20],

              'grk3': [0.10, 0.10, 0.00, 0.10, 0.20],

              'ops4': [0.10, 0.10, 0.00, 0.10, 0.20],

              'g25p': [0.10, 0.10, 0.00, 0.10, 0.20],
```

```
            'o3pr': [0.10, 0.10, 0.00, 0.10, 0.20]}

    # SD values
    sds = {'grk4': [0.06, 0.06, 0.00, 0.06, 0.10],
           'grk3': [0.06, 0.06, 0.00, 0.06, 0.10],
           'ops4': [0.06, 0.06, 0.00, 0.06, 0.10],
           'g25p': [0.06, 0.06, 0.00, 0.06, 0.10],
           'o3pr': [0.06, 0.06, 0.00, 0.06, 0.10]}

    for i, arm in enumerate(arms):
        data_list.append({
            'Model': model,
            f'{arm}\nMean': means[model][i],
            f'{arm}\nRange': ranges[model][i],
            f'{arm}\nSD': sds[model][i]
        })

# Create DataFrame with alternating statistics
columns = []
for arm in arms:
    columns.extend([f'{arm}\nMean', f'{arm}\nRange', f'{arm}\nSD'])

df_plot = pd.DataFrame(columns=['Model'] + columns)
```

```
for model in models:
    row_data = {'Model': model}
    for i, arm in enumerate(arms):
        row_data[f'{arm}\nMean'] = means[model][i]
        row_data[f'{arm}\nRange'] = ranges[model][i]
        row_data[f'{arm}\nSD'] = sds[model][i]
    df_plot = pd.concat([df_plot, pd.DataFrame([row_data])], ignore_index=True)


# Create parallel coordinates plot
plt.figure(figsize=(16, 8))
parallel_coordinates(df_plot, 'Model', colormap='tab10', alpha=0.7, linewidth=2)


plt.title('Table 4: Parallel Coordinates of Statistical Calculations Across Arms', fontsize=16)
plt.xlabel('Statistics by Arm', fontsize=12)
plt.ylabel('Value (months)', fontsize=12)
plt.xticks(rotation=90)
plt.grid(True, alpha=0.3)
plt.legend(title='Models', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.tight_layout()
plt.show()
```

08 - Stacked Bar Chart for Agreement Frequency

```python
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd


# Calculate pairwise differences for all values

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

model_pairs = []

exact_agreement = []

minor_discrepancy = []

major_discrepancy = []


# All Row Consistency Scores

all_scores = {

    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]

}
```

```
# Calculate for each model pair

for i in range(len(models)):

    for j in range(i+1, len(models)):

        model1, model2 = models[i], models[j]

        model_pairs.append(f'{model1}-{model2}')



        # Calculate differences

        diffs = [abs(all_scores[model1][k] - all_scores[model2][k]) for k in range(len(all_scores[model1]))]



        exact = sum(1 for d in diffs if d < 0.01)

        minor = sum(1 for d in diffs if 0.01 <= d <= 0.1)

        major = sum(1 for d in diffs if d > 0.1)



        exact_agreement.append(exact)

        minor_discrepancy.append(minor)

        major_discrepancy.append(major)



# Create stacked bar chart

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



x = np.arange(len(model_pairs))

width = 0.6
```

```
p1 = ax.bar(x, exact_agreement, width, label='Exact Agreement (<0.01)', color='#2ca02c')

p2 = ax.bar(x, minor_discrepancy, width, bottom=exact_agreement,

        label='Minor Discrepancy (0.01-0.1)', color='#ff7f0e')

p3 = ax.bar(x, major_discrepancy, width,

        bottom=np.array(exact_agreement) + np.array(minor_discrepancy),

        label='Major Discrepancy (>0.1)', color='#d62728')



# Add value labels on bars

for i, (e, m, M) in enumerate(zip(exact_agreement, minor_discrepancy, major_discrepancy)):

    if e > 0:

        ax.text(i, e/2, str(e), ha='center', va='center', fontweight='bold')

    if m > 0:

        ax.text(i, e + m/2, str(m), ha='center', va='center', fontweight='bold')

    if M > 0:

        ax.text(i, e + m + M/2, str(M), ha='center', va='center', fontweight='bold')



ax.set_xlabel('Model Pairs', fontsize=12)

ax.set_ylabel('Number of Values', fontsize=12)

ax.set_title('Frequency of Agreement Between Model Pairs Across All Calculated Values', fontsize=16)

ax.set_xticks(x)

ax.set_xticklabels(model_pairs, rotation=45, ha='right')

ax.legend(loc='upper right')

ax.set_ylim(0, 12)
```

```
plt.grid(axis='y', alpha=0.3)

plt.tight_layout()

plt.show()
```

09 - Bubble Chart for Table 5 Statistics

```python
import matplotlib.pyplot as plt

import numpy as np


# Data for Table 5 (12-Month OS Rate Difference)

arms = ['Arm A', 'Arm B', 'Arm C', 'Arm D', 'Arm E']

models = ['grk4', 'grk3', 'ops4', 'g25p', 'o3pr']

colors = {'grk4': '#1f77b4', 'grk3': '#ff7f0e', 'ops4': '#2ca02c',

          'g25p': '#d62728', 'o3pr': '#9467bd'}


# Statistics data

mean_data = {

    'grk4': [0.73, -0.47, -0.67, -0.07, 0.07],

    'grk3': [0.73, -0.47, -0.67, -0.07, 0.07],

    'ops4': [0.73, -0.47, -0.67, -0.07, 0.07],

    'g25p': [0.73, -0.47, -0.67, -0.07, 0.07],
```

```
    'o3pr': [0.73, -0.47, -0.67, -0.07, 0.07]
}

sd_data = {
    'grk4': [0.38, 0.38, 0.57, 0.29, 0.32],
    'grk3': [0.38, 0.38, 0.55, 0.31, 0.29],
    'ops4': [0.38, 0.38, 0.57, 0.29, 0.32],
    'g25p': [0.38, 0.38, 0.57, 0.29, 0.32],
    'o3pr': [0.38, 0.38, 0.57, 0.29, 0.32]
}

range_data = {
    'grk4': [0.70, 0.70, 1.10, 0.50, 0.60],
    'grk3': [0.70, 0.60, 1.10, 0.50, 0.50],
    'ops4': [0.70, 0.70, 1.10, 0.50, 0.60],
    'g25p': [0.70, 0.70, 1.10, 0.50, 0.60],
    'o3pr': [0.70, 0.70, 1.10, 0.50, 0.60]
}

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

# Plot bubbles for each model and arm
```

```
for model in models:
    for i, arm in enumerate(arms):
        x = mean_data[model][i]
        y = sd_data[model][i]
        size = range_data[model][i] * 500  # Scale for visibility

        # Add slight jitter to avoid overlap
        x_jitter = x + np.random.normal(0, 0.01)
        y_jitter = y + np.random.normal(0, 0.005)

        ax.scatter(x_jitter, y_jitter, s=size, c=colors[model], alpha=0.6,
                   edgecolors='black', linewidth=1.5)


# Add legend for models
for model in models:
    ax.scatter([], [], c=colors[model], s=200, label=model, alpha=0.6,
               edgecolors='black', linewidth=1.5)


# Add size legend
sizes = [0.5, 0.75, 1.0]
for size in sizes:
    ax.scatter([], [], c='gray', s=size*500, alpha=0.6,
               edgecolors='black', linewidth=1.5,
```

```
            label=f'Range = {size}')


ax.set_xlabel('Mean 12-Month OS Rate Difference (%)', fontsize=12)

ax.set_ylabel('Standard Deviation (%)', fontsize=12)

ax.set_title('Table 5: Mean vs SD with Bubble Size Representing Range', fontsize=16)

ax.grid(True, alpha=0.3)


# Create two legends

legend1 = ax.legend(models, title='Models', loc='upper left', bbox_to_anchor=(0, 1))

ax.add_artist(legend1)

ax.legend(handles=ax.collections[-3:], labels=[f'Range = {s}' for s in sizes],

        title='Bubble Size', loc='lower right')


# Add arm labels

for i, arm in enumerate(arms):

    x_mean = np.mean([mean_data[model][i] for model in models])

    y_mean = np.mean([sd_data[model][i] for model in models])

    ax.annotate(arm, (x_mean, y_mean), fontsize=10, fontweight='bold',

                ha='center', va='center',

                bbox=dict(boxstyle='round,pad=0.3', facecolor='yellow', alpha=0.3))


plt.tight_layout()

plt.show()
```

```

### 10 - Diverging Bar Chart for Deviations from Median

```python
import matplotlib.pyplot as plt

import numpy as np

import pandas as pd


# Calculate median Row Consistency Score for each table

tables = ['Table 1', 'Table 2-R1', 'Table 2-R2', 'Table 2-R3', 'Table 2-R4',

          'Table 2-R5', 'Table 3', 'Table 4', 'Table 5', 'Table 6']


# All scores

scores = {

    'grk4': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'grk3': [10.0, 9.5, 9.9, 9.7, 10.0, 10.0, 9.8, 9.9, 9.9, 9.8],

    'ops4': [10.0, 9.0, 9.4, 9.8, 10.0, 9.4, 8.9, 9.3, 9.5, 9.6],

    'g25p': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5],

    'o3pr': [10.0, 9.9, 9.9, 9.8, 10.0, 9.9, 8.8, 9.3, 9.4, 9.5]

}


# Calculate deviations

```
deviations_data = []

for i, table in enumerate(tables):

    table_scores = [scores[model][i] for model in scores.keys()]

    median_score = np.median(table_scores)

    for model in scores.keys():

        deviation = scores[model][i] - median_score

        if abs(deviation) > 0.001: # Only include non-zero deviations

            deviations_data.append({

                'Table-Model': f'{table} ({model})',

                'Deviation': deviation,

                'Table': table,

                'Model': model

            })

Sort by absolute deviation

df = pd.DataFrame(deviations_data)

df['Abs_Deviation'] = abs(df['Deviation'])

df = df.sort_values('Abs_Deviation', ascending=True)

Create figure

fig, ax = plt.subplots(figsize=(12, 10))
```

```
# Create color mapping
colors = df['Deviation'].apply(lambda x: '#2ca02c' if x > 0 else '#d62728')

# Create horizontal bar chart
y_pos = np.arange(len(df))
bars = ax.barh(y_pos, df['Deviation'], color=colors, alpha=0.7, edgecolor='black')

# Add value labels
for i, (idx, row) in enumerate(df.iterrows()):
    value = row['Deviation']
    if value > 0:
        ha = 'left'
        x_offset = 0.01
    else:
        ha = 'right'
        x_offset = -0.01
    ax.text(value + x_offset, i, f'{value:+.2f}', ha=ha, va='center', fontweight='bold')

# Customize plot
ax.set_yticks(y_pos)
ax.set_yticklabels(df['Table-Model'])
ax.set_xlabel('Deviation from Median Row Consistency Score', fontsize=12)
ax.set_title('Largest Deviations from Median Scores by Table and Model', fontsize=16)
```

```
ax.axvline(x=0, color='black', linewidth=1, alpha=0.5)

ax.grid(axis='x', alpha=0.3)



# Add legend

ax.plot([], [], 'g-', linewidth=10, alpha=0.7, label='Positive Deviation')

ax.plot([], [], 'r-', linewidth=10, alpha=0.7, label='Negative Deviation')

ax.legend(loc='lower right')



plt.tight_layout()

plt.show()
```

Research Summary Generation 55: 423 Words

I need to analyze the outputs from 5 different AI models (grk4, grk3, ops4, g25p, o3pr) that performed meta-verification analysis on clinical trial data. Let me extract the key information from each model's output.

From the documents:

1. **grk4 (Grok 4)** - Document 1
2. **grk3 (Grok 3 Think)** - Document 2
3. **ops4 (Opus 4 Extended)** - Document 3
4. **g25p (Gemini 2.5 Pro)** - Document 4
5. **o3pr (ChatGPT o3-pro)** - Document 5

Let me extract the Row Consistency Scores from each model for all 6 tables:

Table 1: Cohort Distribution Discrepancy

- grk4: 10.0
- grk3: 10.0
- ops4: 10.0
- g25p: 10.0
- o3pr: 10.0

Table 2: Baseline Characteristic Deviations (Arm A)

- grk4: R1: 9.9, R2: 9.9, R3: 9.8, R4: 10.0, R5: 9.9
- grk3: R1: 9.5, R2: 9.9, R3: 9.7, R4: 10.0, R5: 10.0
- ops4: R1: 9.0, R2: 9.4, R3: 9.8, R4: 10.0, R5: 9.4
- g25p: R1: 9.9, R2: 9.9, R3: 9.8, R4: 10.0, R5: 9.9
- o3pr: R1: 9.9, R2: 9.9, R3: 9.8, R4: 10.0, R5: 9.9

Table 3: Median OS Difference

- grk4: 8.8
- grk3: 9.8
- ops4: 8.9
- g25p: 8.8
- o3pr: 8.8

I observe the Median Overall Survival (OS) scores show remarkable consistency across most models. The variations are minimal, with grk3 standing out slightly at 9.8 compared to the other models' clustered 8.8-8.9 range. This suggests a high degree of alignment in survival outcome interpretations.

Table 4: Median PFS Difference

- grk4: 9.3
- grk3: 9.9
- ops4: 9.3
- g25p: 9.3
- o3pr: 9.3

The Progression-Free Survival (PFS) metrics demonstrate similar tight clustering. Most models converge around 9.3, with grk3 again showing a marginally higher value at 9.9. This pattern indicates robust predictive consistency across different analytical approaches.

Table 5: 12-Month OS Rate Difference

- grk4: 9.4
- grk3: 9.9

- ops4: 9.5
- g25p: 9.4
- o3pr: 9.4

Twelve-month survival rate differences reinforce the previous observations. The models maintain close alignment, with grk3 maintaining its slightly elevated position at 9.9 while other models hover around 9.4-9.5.

Table 6: ≥G3 AE Rate Difference

- grk4: 9.5
- grk3: 9.8
- ops4: 9.6
- g25p: 9.5
- o3pr: 9.5

Adverse event rate differences show consistent patterns, with most models clustering tightly between 9.5-9.6 and grk3 showing a marginally higher value at 9.8.

I'll continue extracting cell-level statistics to validate these broader trends across different model performances.