
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

49.S58b.VIS.01.P44b

Prompt 44b

The content is available under Creative Commons Attribution 4.0 International [CC BY 4.0](https://creativecommons.org/licenses/by/4.0/).

Opus 4 Extended: 112 Pages, July 14, 2025

Two Sets of Generations: Generation 58b1, Generation 58b2

Prompt 44b:

Based on the meta-analysis comparing the 100K-patient virtual PDAC trial with other virtual and real-world clinical trials, generate 10 visualizations with white backgrounds using 10 separate Python scripts that effectively communicate the following key comparisons and findings:

Create the following visualizations to highlight the critical insights from this comparative analysis:

- 01) Forest Plot: Display hazard ratios with 95% confidence intervals for overall survival comparing all experimental arms (virtual Triplet, virtual Doublet, MPACT Gem+nab-P, NAPOLI-1, FOLFIRINOX) versus their respective controls, showing how the virtual trial outcomes align with real-world trials
- 02) Scatter Plot with Efficiency Frontier: Plot median overall survival (x-axis) versus Grade ≥ 3 adverse events percentage (y-axis) for all treatment arms, with point sizes proportional to sample size and an efficiency frontier curve showing optimal efficacy-toxicity balance
- 03) Grouped Bar Chart: Compare median overall survival months across all experimental arms grouped by study type (virtual vs real-world), with error bars and control arm baselines shown as horizontal reference lines
- 04) Waterfall Plot: Display the Efficacy-Toxicity Score (ETS) for each experimental regimen ranked from highest to lowest, with positive scores in green and negative in red to show which treatments offer favorable risk-benefit profiles
- 05) Stacked Bar Chart: Show the distribution of ECOG performance status (0, 1, 2) across different trials to highlight the patient population discrepancy between the virtual trial (>95% ECOG 0-1) and real-world trials

- 06) Heatmap: Create a comparison matrix showing key metrics (OS benefit, PFS benefit, HR, Grade ≥ 3 AE increase, ETS) across all experimental arms with color intensity indicating magnitude of effect
- 07) Butterfly Plot: Display OS benefit (months gained) on the right and toxicity increase (% Grade ≥ 3 AE increase) on the left for each experimental arm, creating a mirror effect to visualize the trade-offs
- 08) Radar Chart: Compare the virtual Triplet, virtual Doublet, and FOLFIRINOX across multiple dimensions (OS, PFS, toxicity, patient fitness, biomarker specificity) to show their relative strengths and weaknesses
- 09) Sankey Diagram: Illustrate patient flow from baseline characteristics through treatment arms to outcomes, showing how the KRAS G12C subgroup (Archetype-05) derives enhanced benefit from Daraxonrasib-containing regimens
- 10) Timeline Visualization: Create a horizontal timeline showing the evolution of PDAC treatment standards from 2010-2025, marking when each real trial was conducted and where the virtual trial fits in the therapeutic landscape with median OS values annotated.

“Start Meta-analysis” “End Meta-analysis”

[S58.REP.02.P44]

Generation 58b1 (Formatted)

Opus 4 Extended

26162 characters, 3 minutes 44.66 seconds

I'll create 10 Python visualization scripts based on the meta-analysis data. Each script will generate a specific visualization with a white background to effectively communicate the key findings.

1. Forest Plot - Hazard Ratios for Overall Survival

Python

```
import matplotlib.pyplot as plt

import numpy as np

# Data from the meta-analysis
```

```

studies = ['FOLFIRINOX', 'NAPOLI-1', 'Virtual Triplet\n(Arm A)',
'MPACT\n(Gem+nab-P)', 'Virtual Doublet\n(Arm D)']

hazard_ratios = [0.57, 0.67, 0.69, 0.72, 0.76]

lower_ci = [0.48, 0.55, 0.60, 0.64, 0.68] # Estimated CIs

upper_ci = [0.67, 0.81, 0.79, 0.81, 0.85] # Estimated CIs

study_types = ['Real-world', 'Real-world', 'Virtual', 'Real-world', 'Virtual']


# Create figure

fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')


# Plot forest plot

y_positions = np.arange(len(studies))

colors = ['#2E86AB' if st == 'Real-world' else '#E63946' for st in study_types]


# Plot confidence intervals

for i, (hr, lower, upper, color) in enumerate(zip(hazard_ratios, lower_ci, upper_ci,
colors)):

    ax.plot([lower, upper], [i, i], 'k-', linewidth=2)

    ax.plot([lower, lower], [i-0.1, i+0.1], 'k-', linewidth=2)

    ax.plot([upper, upper], [i-0.1, i+0.1], 'k-', linewidth=2)

    ax.scatter(hr, i, s=200, color=color, edgecolor='black', linewidth=2, zorder=5)

```

```
# Add reference line at HR=1

ax.axvline(x=1, color='gray', linestyle='--', alpha=0.5)


# Customize plot

ax.set_yticks(y_positions)

ax.set_yticklabels(studies)

ax.set_xlabel('Hazard Ratio (95% CI)', fontsize=12)

ax.set_title('Overall Survival Hazard Ratios: Virtual vs Real-World PDAC Trials',
            fontsize=14, fontweight='bold')

ax.set_xlim(0.4, 1.1)

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


# Add legend

from matplotlib.patches import Patch

legend_elements = [Patch(facecolor='#2E86AB', label='Real-world Trial'),
                   Patch(facecolor='#E63946', label='Virtual Trial')]

ax.legend(handles=legend_elements, loc='upper right')


# Add HR values as text
```

```

for i, hr in enumerate(hazard_ratios):

    ax.text(hr + 0.02, i, f'{hr:.2f}', va='center', fontsize=10)


plt.tight_layout()

plt.show()

```

2. Scatter Plot with Efficiency Frontier

Python

```

import matplotlib.pyplot as plt

import numpy as np

from scipy.interpolate import make_interp_spline


# Data from meta-analysis

treatments = ['Gemcitabine\n(Control)', 'Virtual Control\n(Arm E)',
              '5-FU/LV\n(Control)',

              'MPACT\n(Gem+nab-P)', 'Virtual Doublet\n(Arm D)', 'Virtual
Triplet\n(Arm A)',

              'NAPOLI-1\n(nal-IRI+5FU)', 'FOLFIRINOX']

median_os = [6.7, 6.1, 4.2, 8.5, 8.0, 8.7, 6.2, 11.1]

grade3_ae = [50, 76.5, 46, 84, 65, 94, 76, 75] # Estimated for Doublet

sample_sizes = [430, 20000, 149, 431, 20000, 20000, 117, 171]

```

```

is_control = [True, True, True, False, False, False, False, False]

# Create figure

fig, ax = plt.subplots(figsize=(12, 9), facecolor='white')

# Normalize sample sizes for visualization

normalized_sizes = np.array(sample_sizes) / 100

normalized_sizes[normalized_sizes > 50] = 50 # Cap virtual trial sizes

# Plot points

for i, (os, ae, size, control, name) in enumerate(zip(median_os, grade3_ae,
normalized_sizes, is_control, treatments)):

    color = '#cccccc' if control else '#E63946' if 'Virtual' in name and not control
    else '#2E86AB'

    marker = 'o' if not control else 's'

    ax.scatter(os, ae, s=size*10, c=color, alpha=0.7, edgecolor='black',
linewidth=1.5, marker=marker)

# Add labels for experimental arms

if not control:

    ax.annotate(name, (os, ae), xytext=(5, 5), textcoords='offset points',
fontsize=9)

```

```
# Create efficiency frontier for experimental arms only

exp_os = [median_os[i] for i in range(len(median_os)) if not is_control[i]]

exp_ae = [grade3_ae[i] for i in range(len(grade3_ae)) if not is_control[i]]
```

```
# Sort by OS for smooth curve

sorted_indices = np.argsort(exp_os)

sorted_os = np.array(exp_os)[sorted_indices]

sorted_ae = np.array(exp_ae)[sorted_indices]
```

```
# Create smooth efficiency frontier

x_smooth = np.linspace(min(sorted_os), max(sorted_os), 300)

spl = make_interp_spline(sorted_os, sorted_ae, k=2)

y_smooth = spl(x_smooth)
```

```
ax.plot(x_smooth, y_smooth, 'g--', alpha=0.5, linewidth=2, label='Efficiency  
Frontier')
```

```
# Customize plot
```

```
ax.set_xlabel('Median Overall Survival (months)', fontsize=12)
```

```

ax.set_ylabel('Grade  $\geq 3$  Adverse Events (%)', fontsize=12)

ax.set_title('Efficacy-Toxicity Trade-off in PDAC Trials\n(Bubble size proportional
to sample size)',

            fontsize=14, fontweight='bold')

ax.grid(True, alpha=0.3)

ax.set_xlim(3, 12)

ax.set_ylim(40, 100)


# Add legend

from matplotlib.lines import Line2D

legend_elements = [Line2D([0], [0], marker='o', color='w', label='Real-world
Experimental',

                        markerfacecolor='#2E86AB', markersize=10),

                  Line2D([0], [0], marker='o', color='w', label='Virtual
Experimental',

                        markerfacecolor='#E63946', markersize=10),

                  Line2D([0], [0], marker='s', color='w', label='Control Arms',

                        markerfacecolor='cccccc', markersize=10),

                  Line2D([0], [0], color='g', linestyle='--', label='Efficiency
Frontier')]
```

ax.legend(handles=legend_elements, loc='upper left')


```
plt.tight_layout()
```

```
plt.show()
```

3. Grouped Bar Chart - Median OS by Study Type

Python

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
# Data organization
```

```
virtual_trials = {
```

```
    'names': ['Control\n(Arm E)', 'Doublet\n(Arm D)', 'Triplet\n(Arm A)'],
```

```
    'os': [6.1, 8.0, 8.7],
```

```
    'is_control': [True, False, False]
```

```
}
```

```
real_trials = {
```

```
    'names': ['Gem\n(MPACT)', 'Gem+nab-P\n(MPACT)', '5-FU\n(NAPOLI)',  
    'nal-IRI+5FU\n(NAPOLI)',
```

```
    'Gem\n(PRODIGE)', 'FOLFIRINOX\n(PRODIGE)'],
```

```
    'os': [6.7, 8.5, 4.2, 6.2, 6.8, 11.1],
```

```
    'is_control': [True, False, True, False, True, False]
```

```

}

# Create figure

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7), facecolor='white')

# Virtual trials subplot

x_virtual = np.arange(len(virtual_trials['names']))

colors_virtual = ['#cccccc' if ctrl else '#E63946' for ctrl in
virtual_trials['is_control']]

bars1 = ax1.bar(x_virtual, virtual_trials['os'], color=colors_virtual,
edgecolor='black', linewidth=1.5)

# Add control reference line

ax1.axhline(y=6.1, color='gray', linestyle='--', alpha=0.5, label='Virtual Control
Baseline')

ax1.set_xticks(x_virtual)

ax1.set_xticklabels(virtual_trials['names'])

ax1.set_ylabel('Median Overall Survival (months)', fontsize=12)

ax1.set_title('Virtual Trials', fontsize=13, fontweight='bold')

ax1.set_ylim(0, 13)

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

```

```

# Add values on bars

for bar, val in zip(bars1, virtual_trials['os']):

    ax1.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.1,

             f'{val:.1f}', ha='center', va='bottom', fontsize=10)


# Real-world trials subplot

x_real = np.arange(len(real_trials['names']))

colors_real = ['#cccccc' if ctrl else '#2E86AB' for ctrl in
real_trials['is_control']]

bars2 = ax2.bar(x_real, real_trials['os'], color=colors_real, edgecolor='black',
linewidth=1.5)


# Add multiple control reference lines

control_values = [6.7, 4.2, 6.8]

control_labels = ['Gem (MPACT/PRODIGE)', '5-FU (NAPOLI)', '']

for val, label in zip(control_values[:2], control_labels[:2]):

    ax2.axhline(y=val, color='gray', linestyle='--', alpha=0.5, linewidth=1)


ax2.set_xticks(x_real)

ax2.set_xticklabels(real_trials['names'], rotation=45, ha='right')

```

```

ax2.set_ylabel('Median Overall Survival (months)', fontsize=12)

ax2.set_title('Real-World Trials', fontsize=13, fontweight='bold')

ax2.set_ylim(0, 13)

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

# Add values on bars

for bar, val in zip(bars2, real_trials['os']):

    ax2.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.1,

             f'{val:.1f}', ha='center', va='bottom', fontsize=10)

# Overall title

fig.suptitle('Median Overall Survival: Virtual vs Real-World PDAC Trials',

             fontsize=15, fontweight='bold', y=0.98)

# Add legend

from matplotlib.patches import Patch

legend_elements = [Patch(facecolor='#E63946', label='Virtual Experimental'),

                   Patch(facecolor='#2E86AB', label='Real-World Experimental'),

                   Patch(facecolor='#cccccc', label='Control Arms')]

```

```
fig.legend(handles=legend_elements, loc='upper center', bbox_to_anchor=(0.5, 0.93),
ncol=3)
```

```
plt.tight_layout()
```

```
plt.show()
```

4. Waterfall Plot - Efficacy-Toxicity Scores

Python

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
# ETS data from meta-analysis
```

```
treatments = ['Virtual Triplet\n(Arm A)', 'NAPOLI-1\n(estimated)',
'MPACT\n(Gem+nab-P)',
```

```
              'FOLFIRINOX', 'Virtual Doublet\n(Arm D)']
```

```
ets_scores = [-0.69, -0.2, 0.00, 0.36, np.nan] # NaN for Doublet due to missing
toxicity data
```

```
# Remove NaN values
```

```
valid_indices = ~np.isnan(ets_scores)
```

```
treatments_valid = [treatments[i] for i in range(len(treatments)) if
valid_indices[i]]
```

```
ets_valid = [ets_scores[i] for i in range(len(ets_scores)) if valid_indices[i]]
```

```
# Sort by ETS score
```

```
sorted_indices = np.argsort(ets_valid)[::-1]
```

```
treatments_sorted = [treatments_valid[i] for i in sorted_indices]
```

```
ets_sorted = [ets_valid[i] for i in sorted_indices]
```

```
# Create figure
```

```
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')
```

```
# Create waterfall plot
```

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

```
colors = ['#2ecc71' if score > 0 else '#e74c3c' if score < 0 else '#95a5a6' for  
score in ets_sorted]
```

```
bars = ax.bar(x, ets_sorted, color=colors, edgecolor='black', linewidth=1.5,  
alpha=0.8)
```

```
# Add zero line
```

```
ax.axhline(y=0, color='black', linewidth=2)
```

```

# Customize plot

ax.set_xticks(x)

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

ax.set_ylabel('Efficacy-Toxicity Score (ETS)', fontsize=12)

ax.set_title('Efficacy-Toxicity Balance Across PDAC Treatment Regimens\n(Positive =
Favorable Risk-Benefit)',

             fontsize=14, fontweight='bold')

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

ax.set_ylim(-0.8, 0.5)


# Add value labels

for bar, val in zip(bars, ets_sorted):

    height = bar.get_height()

    label_y = height + 0.02 if height > 0 else height - 0.02

    va = 'bottom' if height > 0 else 'top'

    ax.text(bar.get_x() + bar.get_width()/2, label_y, f'{val:.2f}',

            ha='center', va=va, fontsize=11, fontweight='bold')


# Add interpretation text

ax.text(0.02, 0.98, 'Higher scores indicate efficacy gains outweigh toxicity
burden',

```

```

        transform=ax.transAxes, fontsize=10, va='top', style='italic')

# Add note about missing data

ax.text(0.98, 0.02, '*Virtual Doublet ETS not calculated due to missing toxicity
data',

        transform=ax.transAxes, fontsize=9, ha='right', va='bottom', style='italic')

plt.tight_layout()

plt.show()

```

5. Stacked Bar Chart - ECOG Performance Status Distribution

Python

```

import matplotlib.pyplot as plt

import numpy as np

# ECOG distribution data

trials = ['Virtual Trial\n(100K-Sim)', 'MPACT', 'NAPOLI-1', 'FOLFIRINOX\n(PRODIGE)']

ecog_0_1 = [97, 93, 100, 100] # ECOG 0-1 percentages

ecog_2 = [3, 7, 0, 0] # ECOG 2 percentages (virtual trial shows
underrepresentation)

```



```

# Create figure

fig, ax = plt.subplots(figsize=(10, 7), facecolor='white')

# Create stacked bar chart

x = np.arange(len(trials))

width = 0.6

bars1 = ax.bar(x, ecog_0_1, width, label='ECOG 0-1', color='#2ecc71',
edgecolor='black', linewidth=1.5)

bars2 = ax.bar(x, ecog_2, width, bottom=ecog_0_1, label='ECOG 2', color='#e74c3c',
edgecolor='black', linewidth=1.5)

# Add validation failure annotation for virtual trial

ax.annotate('Validation Failure:\nUnderrepresentation\nof ECOG 2',

            xy=(0, 98), xytext=(-0.3, 85),

            arrowprops=dict(arrowstyle='->', color='red', lw=2),

            fontsize=10, color='red', fontweight='bold', ha='center')

# Customize plot

ax.set_ylabel('Percentage of Patients (%)', fontsize=12)

ax.set_xlabel('Clinical Trials', fontsize=12)

```

```
ax.set_title('ECOG Performance Status Distribution:\nVirtual vs Real-World PDAC Trials',
```

```
            fontsize=14, fontweight='bold')
```

```
ax.set_xticks(x)
```

```
ax.set_xticklabels(trials)
```

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

```
ax.set_ylim(0, 105)
```

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

```
# Add percentage labels
```

```
for i, (bar1, bar2) in enumerate(zip(bars1, bars2)):
```

```
    # ECOG 0-1 label
```

```
    height1 = bar1.get_height()
```

```
    if height1 > 5: # Only show if segment is large enough
```

```
        ax.text(bar1.get_x() + bar1.get_width()/2, height1/2,
```

```
                f'{ecog_0_1[i]}%', ha='center', va='center', fontsize=11,
                fontweight='bold')
```

```
    # ECOG 2 label
```

```
    height2 = bar2.get_height()
```

```
    if height2 > 5: # Only show if segment is large enough
```

```

        ax.text(bar2.get_x() + bar2.get_width()/2, ecog_0_1[i] + height2/2,
                f'{ecog_2[i]}%', ha='center', va='center', fontsize=11,
fontweight='bold')

# Add note

ax.text(0.5, -0.15, 'Note: Real-world PDAC populations typically have 10-20% ECOG 2
patients',

        transform=ax.transAxes, fontsize=10, ha='center', style='italic')

plt.tight_layout()

plt.show()

```

6. Heatmap - Key Metrics Comparison Matrix

Python

```

import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

# Prepare data matrix

treatments = ['Virtual Triplet', 'Virtual Doublet', 'MPACT', 'NAPOLI-1',
'FOLFIRINOX']

```

```
metrics = ['OS Benefit\n(months)', 'HR vs Control', 'Grade  $\geq$ 3 AE\nIncrease (%)',  
'ETS Score']
```

```
# Data matrix (rows: treatments, columns: metrics)
```

```
data = np.array([  
  
    [2.6, 0.69, 17.5, -0.69],    # Virtual Triplet  
  
    [1.9, 0.76, np.nan, np.nan], # Virtual Doublet  
  
    [1.8, 0.72, 13.6, 0.00],    # MPACT  
  
    [2.0, 0.67, 30.0, -0.20],    # NAPOLI-1 (estimated)  
  
    [4.3, 0.57, 25.0, 0.36]     # FOLFIRINOX  
  
])
```

```
# Create figure
```

```
fig, ax = plt.subplots(figsize=(10, 6), facecolor='white')
```

```
# Create custom colormap
```

```
cmap = sns.diverging_palette(10, 150, as_cmap=True)
```

```
# Create heatmap with annotations
```

```
mask = np.isnan(data)
```

```

sns.heatmap(data, annot=True, fmt='.2f', cmap=cmap, center=0,

            xticklabels=metrics, yticklabels=treatments,

            cbar_kws={'label': 'Metric Value'}, linewidths=1,

            mask=mask, square=True, ax=ax, vmin=-1, vmax=5)

# Customize

ax.set_title('Comparative Metrics Across PDAC Treatment Regimens', fontsize=14,
fontweight='bold', pad=20)

# Rotate x-axis labels

plt.setp(ax.get_xticklabels(), rotation=0, ha='center')

plt.setp(ax.get_yticklabels(), rotation=0)

# Add asterisks for virtual trials

for i, label in enumerate(ax.get_yticklabels()):

    if 'Virtual' in label.get_text():

        label.set_text(label.get_text() + ' *')

        label.set_color('#E63946')

# Add note

```

```

ax.text(0.5, -0.15, '* Virtual Trial Arms | Lower HR is better | Higher OS benefit
is better | Lower AE increase is better | Higher ETS is better',

        transform=ax.transAxes, fontsize=9, ha='center', style='italic')

plt.tight_layout()

plt.show()

```

7. Butterfly Plot - OS Benefit vs Toxicity Trade-off

Python

```

import matplotlib.pyplot as plt

import numpy as np

# Data

treatments = ['FOLFIRINOX', 'Virtual Triplet\n(Arm A)', 'NAPOLI-1', 'Virtual
Doublet\n(Arm D)', 'MPACT']

os_benefit = [4.3, 2.6, 2.0, 1.9, 1.8] # Months gained

toxicity_increase = [25.0, 17.5, 30.0, 8.0, 13.6] # % increase in Grade ≥3 AEs
(estimated for Doublet)

# Create figure

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

```

```

# Y-axis positions

y_pos = np.arange(len(treatments))


# Create butterfly plot

# Left side - Toxicity (negative values for visualization)

bars_left = ax.barh(y_pos, -np.array(toxicity_increase), color='#e74c3c', alpha=0.7,
                    edgecolor='black', linewidth=1.5, label='Toxicity Increase')


# Right side - OS Benefit

bars_right = ax.barh(y_pos, os_benefit, color='#2ecc71', alpha=0.7,
                    edgecolor='black', linewidth=1.5, label='OS Benefit')


# Add center line

ax.axvline(x=0, color='black', linewidth=2)


# Customize plot

ax.set_yticks(y_pos)

ax.set_yticklabels(treatments, fontsize=11)

ax.set_xlabel('← Grade ≥3 AE Increase (%) | OS Benefit (months) →', fontsize=12)

ax.set_title('Treatment Trade-offs: Toxicity vs Survival Benefit in PDAC',

```

```

        fontsize=14, fontweight='bold')

ax.set_xlim(-35, 5)

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

# Add value labels

for i, (tox, os) in enumerate(zip(toxicity_increase, os_benefit)):

    # Toxicity labels

    ax.text(-tox/2, i, f'{tox:.1f}%', ha='center', va='center', fontsize=10,
fontweight='bold')

    # OS benefit labels

    ax.text(os/2, i, f'{os:.1f} mo', ha='center', va='center', fontsize=10,
fontweight='bold')

# Highlight virtual trials

for i, treatment in enumerate(treatments):

    if 'Virtual' in treatment:

        ax.get_yticklabels()[i].set_color('#E63946')

        ax.get_yticklabels()[i].set_fontweight('bold')

# Add interpretation zones

```



```

ax.text(-30, len(treatments), 'Higher Toxicity', fontsize=10, ha='center',
style='italic', color='#e74c3c')

ax.text(4, len(treatments), 'Greater Benefit', fontsize=10, ha='center',
style='italic', color='#2ecc71')

plt.tight_layout()

plt.show()

```

8. Radar Chart - Multi-dimensional Comparison

Python

```

import matplotlib.pyplot as plt

import numpy as np

# Categories

categories = ['Overall\nSurvival', 'Progression-Free\nSurvival',
'Tolerability\n(inverse toxicity)',

              'Patient\nFitness', 'Biomarker\nSpecificity']

# Data for three key regimens (normalized to 0-10 scale)

virtual_triplet = [7.5, 6.0, 2.0, 9.5, 8.0] # High efficacy, poor tolerability,
very fit patients, KRAS G12C specific

virtual_doublet = [7.0, 5.5, 7.0, 9.5, 8.0] # Moderate efficacy, better
tolerability, very fit patients, KRAS G12C specific

```

```
folfirinox = [10.0, 8.0, 4.0, 10.0, 2.0]    # Highest efficacy, poor tolerability,  
fit patients only, no biomarker
```

```
# Number of variables
```

```
num_vars = len(categories)
```

```
# Compute angle for each axis
```

```
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
```

```
# Complete the circle
```

```
virtual_triplet += virtual_triplet[:1]
```

```
virtual_doublet += virtual_doublet[:1]
```

```
folfirinox += folfirinox[:1]
```

```
angles += angles[:1]
```

```
# Create figure
```

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

```
# Plot data
```

```
ax.plot(angles, virtual_triplet, 'o-', linewidth=2, color='#E63946', label='Virtual Triplet')
```

```
ax.fill(angles, virtual_triplet, alpha=0.25, color='#E63946')
```

```
ax.plot(angles, virtual_doublet, 'o-', linewidth=2, color='#F77F00', label='Virtual Doublet')
```

```
ax.fill(angles, virtual_doublet, alpha=0.25, color='#F77F00')
```

```
ax.plot(angles, folfirinox, 'o-', linewidth=2, color='#2E86AB', label='FOLFIRINOX')
```

```
ax.fill(angles, folfirinox, alpha=0.25, color='#2E86AB')
```

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

```
ax.set_theta_offset(np.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=11)
```

```
# Set y-axis limits and labels
```

```
ax.set_ylim(0, 10)
```

```

ax.set_yticks([2, 4, 6, 8, 10])

ax.set_yticklabels(['2', '4', '6', '8', '10'], size=9)


# Add gridlines

ax.grid(True, linestyle='--', alpha=0.5)


# Add title and legend

ax.set_title('Multi-dimensional Comparison of PDAC Treatment Regimens',

             fontsize=14, fontweight='bold', pad=30)

ax.legend(loc='upper right', bbox_to_anchor=(1.15, 1.15))


# Add interpretation note

fig.text(0.5, 0.02, 'Note: Higher values indicate better performance in each
dimension',

        ha='center', fontsize=10, style='italic')


plt.tight_layout()

plt.show()

```

9. Sankey Diagram - Patient Flow and KRAS G12C Benefit

[illegible]

```

ax.text(0.125, 0.5, '100K Virtual\nPatients\n\n91% KRAS mut\n5% KRAS G12C\n5% KRAS
WT',

        ha='center', va='center', fontsize=10, fontweight='bold')


# Draw treatment arms

arms = [

    ('Triplet\n(Arm A)', 0.85, '#E63946'),

    ('Doublet\n(Arm D)', 0.55, '#F77F00'),

    ('Control\n(Arm E)', 0.25, '#95a5a6')

]


for name, y, color in arms:

    arm_box = patches.FancyBboxPatch((0.35, y-0.08), 0.15, 0.12,

                                     boxstyle="round,pad=0.02",

                                     facecolor=color, alpha=0.7, edgecolor='black',
linewidth=2)

    ax.add_patch(arm_box)

    ax.text(0.425, y, name, ha='center', va='center', fontsize=10,
fontweight='bold')


# Draw outcome boxes with KRAS G12C highlight

```

```

outcomes = [

    ('KRAS G12C\nEnhanced Benefit\nOS: ~11 months', 0.85, '#2ecc71'),

    ('Overall Benefit\nOS: 8.7 months', 0.65, '#3498db'),

    ('Standard Benefit\nOS: ~8.0 months', 0.45, '#3498db'),

    ('Baseline\nOS: 6.1 months', 0.25, '#bdc3c7')

]

for text, y, color in outcomes:

    outcome_box = patches.FancyBboxPatch((0.65, y-0.08), 0.25, 0.12,

                                         boxstyle="round,pad=0.02",

                                         facecolor=color, alpha=0.7,
edgecolor='black', linewidth=2)

    ax.add_patch(outcome_box)

    ax.text(0.775, y, text, ha='center', va='center', fontsize=9)


# Draw arrows showing patient flow

# From population to arms

for y in [0.85, 0.55, 0.25]:

    ax.arrow(0.20, 0.5, 0.13, y-0.5, head_width=0.02, head_length=0.01,

            fc='gray', ec='gray', alpha=0.5, linewidth=2)

```

```
# From arms to outcomes
```

```
arrows = [
```

```
    (0.50, 0.85, 0.13, 0, '#E63946'), # Triplet to KRAS G12C
```

```
    (0.50, 0.85, 0.13, -0.2, '#E63946'), # Triplet to overall
```

```
    (0.50, 0.55, 0.13, -0.1, '#F77F00'), # Doublet
```

```
    (0.50, 0.25, 0.13, 0, '#95a5a6') # Control
```

```
]
```

```
for x, y, dx, dy, color in arrows:
```

```
    ax.arrow(x, y, dx, dy, head_width=0.02, head_length=0.01,
```

```
            fc=color, ec=color, alpha=0.7, linewidth=2)
```

```
# Add key finding annotation
```

```
ax.annotate('KEY FINDING:\nKRAS G12C patients\n(~5% of population)\nachieve ~2.5\nmonth\nadditional OS benefit',
```

```
          xy=(0.775, 0.85), xytext=(0.9, 0.7),
```

```
          arrowprops=dict(arrowstyle='->', color='#2ecc71', lw=2),
```

```
          fontsize=10, color='#2ecc71', fontweight='bold',
```

```
          bbox=dict(boxstyle="round,pad=0.3", facecolor='white',  
                    edgecolor='#2ecc71'))
```



```

# Add title

ax.text(0.5, 0.95, 'Patient Flow Analysis: KRAS G12C Subgroup Benefit in Virtual
PDAC Trial',

        ha='center', va='center', fontsize=14, fontweight='bold')


# Set axis limits

ax.set_xlim(0, 1)

ax.set_ylim(0, 1)


plt.tight_layout()

plt.show()

```

10. Timeline Visualization - Evolution of PDAC Treatments

```

Python

import matplotlib.pyplot as plt

import matplotlib.patches as patches

import numpy as np


# Timeline data

trials = [

```

```

('PRODIGE 4\n(FOLFIRINOX)', 2011, 11.1, 'Real', '#2E86AB'),

('MPACT\n(Gem+nab-P)', 2013, 8.5, 'Real', '#2E86AB'),

('NAPOLI-1\n(nal-IRI+5FU)', 2016, 6.2, 'Real', '#2E86AB'),

('Digital Twin\nValidation', 2024, 8.5, 'Virtual', '#9B59B6'),

('100K Virtual\nTriplet', 2025, 8.7, 'Virtual', '#E63946'),

('100K Virtual\nDoublet', 2025, 8.0, 'Virtual', '#F77F00')

]

# Create figure

fig, ax = plt.subplots(figsize=(14, 8), facecolor='white')

# Timeline

start_year = 2010

end_year = 2026

ax.plot([start_year, end_year], [0, 0], 'k-', linewidth=3)

# Add year markers

for year in range(start_year, end_year + 1, 2):

    ax.plot([year, year], [-0.1, 0.1], 'k-', linewidth=2)

    ax.text(year, -0.3, str(year), ha='center', va='top', fontsize=10)

```

```

# Plot trials

for i, (name, year, os, trial_type, color) in enumerate(trials):

    y_offset = 0.5 + (i % 2) * 0.5 # Alternate heights

    # Draw stem

    ax.plot([year, year], [0, y_offset], color=color, linewidth=2, alpha=0.7)

    # Draw box

    box = patches.FancyBboxPatch((year-0.4, y_offset), 0.8, 0.3,

                                boxstyle="round,pad=0.02",

                                facecolor=color, alpha=0.8, edgecolor='black',
linewidth=1.5)

    ax.add_patch(box)

    # Add text

    ax.text(year, y_offset + 0.15, name, ha='center', va='center',

            fontsize=9, fontweight='bold', color='white')

    ax.text(year, y_offset + 0.35, f'OS: {os} mo', ha='center', va='bottom',

            fontsize=9, fontweight='bold')

```

```
# Add era labels

ax.text(2012.5, -0.8, 'Chemotherapy Era', ha='center', fontsize=11,

        style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ecf0f1'))

ax.text(2020, -0.8, 'Precision Medicine Era', ha='center', fontsize=11,

        style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ecf0f1'))

ax.text(2024.5, -0.8, 'Virtual Trial Era', ha='center', fontsize=11,

        style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ffeaa7'))


# Add trend line for OS

years = [2011, 2013, 2016, 2025, 2025]

os_values = [11.1, 8.5, 6.2, 8.7, 8.0]

z = np.polyfit(years, os_values, 2)

p = np.poly1d(z)

years_smooth = np.linspace(2011, 2025, 100)

os_smooth = p(years_smooth)

ax.plot(years_smooth, os_smooth/10 - 0.2, 'g--', alpha=0.5, linewidth=2)


# Customize plot

ax.set_xlim(start_year - 0.5, end_year + 0.5)
```

```

ax.set_ylim(-1, 1.5)

ax.set_title('Evolution of PDAC Treatment Standards: 2010-2025\n(Timeline shows
introduction of key trials with median OS outcomes)',

             fontsize=14, fontweight='bold')

ax.axis('off')

# Add legend

legend_elements = [patches.Patch(facecolor='#2E86AB', label='Real-World Trial'),

                   patches.Patch(facecolor='#9B59B6', label='Virtual Validation'),

                   patches.Patch(facecolor='#E63946', label='Virtual
Hypothesis-Generating')]

ax.legend(handles=legend_elements, loc='upper left')

# Add key insight

ax.text(0.98, 0.02, 'Key Insight: Virtual trials (2024-2025) achieve OS comparable
to real standards,\nopening new avenues for rapid hypothesis testing in PDAC',

       transform=ax.transAxes, ha='right', va='bottom', fontsize=10,

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

plt.tight_layout()

plt.show()

```

These visualizations effectively communicate the key findings from the meta-analysis:

1. **Forest Plot:** Shows the virtual trial HRs are comparable to real trials
2. **Scatter Plot:** Reveals the efficacy-toxicity trade-offs across regimens
3. **Grouped Bar Chart:** Directly compares OS between virtual and real trials
4. **Waterfall Plot:** Highlights which regimens offer favorable risk-benefit profiles
5. **Stacked Bar Chart:** Exposes the patient population discrepancy in the virtual trial
6. **Heatmap:** Provides a comprehensive metric comparison at a glance
7. **Butterfly Plot:** Visualizes the opposing forces of benefit vs toxicity
8. **Radar Chart:** Shows multi-dimensional strengths/weaknesses of key regimens
9. **Sankey Diagram:** Illustrates the KRAS G12C subgroup benefit flow
10. **Timeline:** Places the virtual trial in historical context of PDAC treatment evolution

Each visualization has a white background as requested and effectively communicates specific aspects of the meta-analysis findings.

Generation 58b1 (Plain Text)

Opus 4 Extended

26383 characters, 3 minutes 44.66 seconds

I'll create 10 Python visualization scripts based on the meta-analysis data. Each script will generate a specific visualization with a white background to effectively communicate the key findings.

1. Forest Plot - Hazard Ratios for Overall Survival

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

Data from the meta-analysis
studies = ['FOLFIRINOX', 'NAPOLI-1', 'Virtual Triplet\n(Arm A)', 'MPACT\n(Gem+nab-P)', 'Virtual Doublet\n(Arm D)']
hazard_ratios = [0.57, 0.67, 0.69, 0.72, 0.76]
lower_ci = [0.48, 0.55, 0.60, 0.64, 0.68] # Estimated CIs
upper_ci = [0.67, 0.81, 0.79, 0.81, 0.85] # Estimated CIs
study_types = ['Real-world', 'Real-world', 'Virtual', 'Real-world', 'Virtual']

Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

Plot forest plot
y_positions = np.arange(len(studies))
```

```

colors = ['#2E86AB' if st == 'Real-world' else '#E63946' for st in study_types]

Plot confidence intervals
for i, (hr, lower, upper, color) in enumerate(zip(hazard_ratios, lower_ci, upper_ci, colors)):
 ax.plot([lower, upper], [i, i], 'k-', linewidth=2)
 ax.plot([lower, lower], [i-0.1, i+0.1], 'k-', linewidth=2)
 ax.plot([upper, upper], [i-0.1, i+0.1], 'k-', linewidth=2)
 ax.scatter(hr, i, s=200, color=color, edgecolor='black', linewidth=2, zorder=5)

Add reference line at HR=1
ax.axvline(x=1, color='gray', linestyle='--', alpha=0.5)

Customize plot
ax.set_yticks(y_positions)
ax.set_yticklabels(studies)
ax.set_xlabel('Hazard Ratio (95% CI)', fontsize=12)
ax.set_title('Overall Survival Hazard Ratios: Virtual vs Real-World PDAC Trials', fontsize=14, fontweight='bold')
ax.set_xlim(0.4, 1.1)
ax.grid(axis='x', alpha=0.3)

Add legend
from matplotlib.patches import Patch
legend_elements = [Patch(facecolor='#2E86AB', label='Real-world Trial'),
 Patch(facecolor='#E63946', label='Virtual Trial')]
ax.legend(handles=legend_elements, loc='upper right')

Add HR values as text
for i, hr in enumerate(hazard_ratios):
 ax.text(hr + 0.02, i, f'{hr:.2f}', va='center', fontsize=10)

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

```

## ## 2. Scatter Plot with Efficiency Frontier

```

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import make_interp_spline

```

```

# Data from meta-analysis
treatments = ['Gemcitabine\n(Control)', 'Virtual Control\n(Arm E)', '5-FU/LV\n(Control)',
              'MPACT\n(Gem+nab-P)', 'Virtual Doublet\n(Arm D)', 'Virtual Triplet\n(Arm A)',
              'NAPOLI-1\n(naI-IRI+5FU)', 'FOLFIRINOX']
median_os = [6.7, 6.1, 4.2, 8.5, 8.0, 8.7, 6.2, 11.1]
grade3_ae = [50, 76.5, 46, 84, 65, 94, 76, 75] # Estimated for Doublet
sample_sizes = [430, 20000, 149, 431, 20000, 20000, 117, 171]
is_control = [True, True, True, False, False, False, False, False]

# Create figure
fig, ax = plt.subplots(figsize=(12, 9), facecolor='white')

# Normalize sample sizes for visualization
normalized_sizes = np.array(sample_sizes) / 100
normalized_sizes[normalized_sizes > 50] = 50 # Cap virtual trial sizes

# Plot points
for i, (os, ae, size, control, name) in enumerate(zip(median_os, grade3_ae, normalized_sizes, is_control, treatments)):
    color = '#cccccc' if control else '#E63946' if 'Virtual' in name and not control else '#2E86AB'
    marker = 'o' if not control else 's'
    ax.scatter(os, ae, s=size*10, c=color, alpha=0.7, edgecolor='black', linewidth=1.5, marker=marker)

# Add labels for experimental arms
if not control:
    ax.annotate(name, (os, ae), xytext=(5, 5), textcoords='offset points', fontsize=9)

# Create efficiency frontier for experimental arms only
exp_os = [median_os[i] for i in range(len(median_os)) if not is_control[i]]
exp_ae = [grade3_ae[i] for i in range(len(grade3_ae)) if not is_control[i]]

# Sort by OS for smooth curve
sorted_indices = np.argsort(exp_os)
sorted_os = np.array(exp_os)[sorted_indices]
sorted_ae = np.array(exp_ae)[sorted_indices]

# Create smooth efficiency frontier
x_smooth = np.linspace(min(sorted_os), max(sorted_os), 300)
spl = make_interp_spline(sorted_os, sorted_ae, k=2)
y_smooth = spl(x_smooth)

ax.plot(x_smooth, y_smooth, 'g--', alpha=0.5, linewidth=2, label='Efficiency Frontier')

```



```

# Customize plot
ax.set_xlabel('Median Overall Survival (months)', fontsize=12)
ax.set_ylabel('Grade  $\geq 3$  Adverse Events (%)', fontsize=12)
ax.set_title('Efficacy-Toxicity Trade-off in PDAC Trials\n(Bubble size proportional to sample size)',
            fontsize=14, fontweight='bold')
ax.grid(True, alpha=0.3)
ax.set_xlim(3, 12)
ax.set_ylim(40, 100)

# Add legend
from matplotlib.lines import Line2D
legend_elements = [Line2D([0], [0], marker='o', color='w', label='Real-world Experimental',
                        markerfacecolor='#2E86AB', markersize=10),
                  Line2D([0], [0], marker='o', color='w', label='Virtual Experimental',
                        markerfacecolor='#E63946', markersize=10),
                  Line2D([0], [0], marker='s', color='w', label='Control Arms',
                        markerfacecolor='cccccc', markersize=10),
                  Line2D([0], [0], color='g', linestyle='--', label='Efficiency Frontier')]
ax.legend(handles=legend_elements, loc='upper left')

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

```

3. Grouped Bar Chart - Median OS by Study Type

```

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

Data organization
virtual_trials = {
 'names': ['Control\n(Arm E)', 'Doublet\n(Arm D)', 'Triplet\n(Arm A)'],
 'os': [6.1, 8.0, 8.7],
 'is_control': [True, False, False]
}

real_trials = {
 'names': ['Gem\n(MPACT)', 'Gem+nab-P\n(MPACT)', '5-FU\n(NAPOLI)', 'nal-IRI+5FU\n(NAPOLI)',
 'Gem\n(PRODIGE)', 'FOLFIRINOX\n(PRODIGE)'],

```

```
'os': [6.7, 8.5, 4.2, 6.2, 6.8, 11.1],
'is_control': [True, False, True, False, True, False]
}
```

```
Create figure
```

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(14, 7), facecolor='white')
```

```
Virtual trials subplot
```

```
x_virtual = np.arange(len(virtual_trials['names']))
```

```
colors_virtual = ['#cccccc' if ctrl else '#E63946' for ctrl in virtual_trials['is_control']]
```

```
bars1 = ax1.bar(x_virtual, virtual_trials['os'], color=colors_virtual, edgecolor='black', linewidth=1.5)
```

```
Add control reference line
```

```
ax1.axhline(y=6.1, color='gray', linestyle='--', alpha=0.5, label='Virtual Control Baseline')
```

```
ax1.set_xticks(x_virtual)
```

```
ax1.set_xticklabels(virtual_trials['names'])
```

```
ax1.set_ylabel('Median Overall Survival (months)', fontsize=12)
```

```
ax1.set_title('Virtual Trials', fontsize=13, fontweight='bold')
```

```
ax1.set_ylim(0, 13)
```

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

```
Add values on bars
```

```
for bar, val in zip(bars1, virtual_trials['os']):
```

```
 ax1.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.1,
 f'{val:.1f}', ha='center', va='bottom', fontsize=10)
```

```
Real-world trials subplot
```

```
x_real = np.arange(len(real_trials['names']))
```

```
colors_real = ['#cccccc' if ctrl else '#2E86AB' for ctrl in real_trials['is_control']]
```

```
bars2 = ax2.bar(x_real, real_trials['os'], color=colors_real, edgecolor='black', linewidth=1.5)
```

```
Add multiple control reference lines
```

```
control_values = [6.7, 4.2, 6.8]
```

```
control_labels = ['Gem (MPACT/PRODIGE)', '5-FU (NAPOLI)', '']
```

```
for val, label in zip(control_values[:2], control_labels[:2]):
```

```
 ax2.axhline(y=val, color='gray', linestyle='--', alpha=0.5, linewidth=1)
```

```
ax2.set_xticks(x_real)
```

```
ax2.set_xticklabels(real_trials['names'], rotation=45, ha='right')
```

```
ax2.set_ylabel('Median Overall Survival (months)', fontsize=12)
```

```
ax2.set_title('Real-World Trials', fontsize=13, fontweight='bold')
```

```

ax2.set_ylim(0, 13)
ax2.grid(axis='y', alpha=0.3)

Add values on bars
for bar, val in zip(bars2, real_trials['os']):
 ax2.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.1,
 f'{val:.1f}', ha='center', va='bottom', fontsize=10)

Overall title
fig.suptitle('Median Overall Survival: Virtual vs Real-World PDAC Trials',
 fontsize=15, fontweight='bold', y=0.98)

Add legend
from matplotlib.patches import Patch
legend_elements = [Patch(facecolor='#E63946', label='Virtual Experimental'),
 Patch(facecolor='#2E86AB', label='Real-World Experimental'),
 Patch(facecolor='#cccccc', label='Control Arms')]
fig.legend(handles=legend_elements, loc='upper center', bbox_to_anchor=(0.5, 0.93), ncol=3)

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

```

#### ## 4. Waterfall Plot - Efficacy-Toxicity Scores

```

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

# ETS data from meta-analysis
treatments = ['Virtual Triplet\n(Arm A)', 'NAPOLI-1\n(estimated)', 'MPACT\n(Gem+nab-P)',
              'FOLFIRINOX', 'Virtual Doublet\n(Arm D)']
ets_scores = [-0.69, -0.2, 0.00, 0.36, np.nan] # NaN for Doublet due to missing toxicity data

# Remove NaN values
valid_indices = ~np.isnan(ets_scores)
treatments_valid = [treatments[i] for i in range(len(treatments)) if valid_indices[i]]
ets_valid = [ets_scores[i] for i in range(len(ets_scores)) if valid_indices[i]]

# Sort by ETS score
sorted_indices = np.argsort(ets_valid)[-1:]

```

```

treatments_sorted = [treatments_valid[i] for i in sorted_indices]
ets_sorted = [ets_valid[i] for i in sorted_indices]

# Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

# Create waterfall plot
x = np.arange(len(treatments_sorted))
colors = ['#2ecc71' if score > 0 else '#e74c3c' if score < 0 else '#95a5a6' for score in ets_sorted]

bars = ax.bar(x, ets_sorted, color=colors, edgecolor='black', linewidth=1.5, alpha=0.8)

# Add zero line
ax.axhline(y=0, color='black', linewidth=2)

# Customize plot
ax.set_xticks(x)
ax.set_xticklabels(treatments_sorted, rotation=45, ha='right')
ax.set_ylabel('Efficacy-Toxicity Score (ETS)', fontsize=12)
ax.set_title('Efficacy-Toxicity Balance Across PDAC Treatment Regimens\n(Positive = Favorable Risk-Benefit)',
             fontsize=14, fontweight='bold')
ax.grid(axis='y', alpha=0.3)
ax.set_ylim(-0.8, 0.5)

# Add value labels
for bar, val in zip(bars, ets_sorted):
    height = bar.get_height()
    label_y = height + 0.02 if height > 0 else height - 0.02
    va = 'bottom' if height > 0 else 'top'
    ax.text(bar.get_x() + bar.get_width()/2, label_y, f'{val:.2f}',
           ha='center', va=va, fontsize=11, fontweight='bold')

# Add interpretation text
ax.text(0.02, 0.98, 'Higher scores indicate efficacy gains outweigh toxicity burden',
       transform=ax.transAxes, fontsize=10, va='top', style='italic')

# Add note about missing data
ax.text(0.98, 0.02, '**Virtual Doublet ETS not calculated due to missing toxicity data',
       transform=ax.transAxes, fontsize=9, ha='right', va='bottom', style='italic')

plt.tight_layout()

```

```
plt.show()
```

```
'''
```

5. Stacked Bar Chart - ECOG Performance Status Distribution

```
'''python
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
# ECOG distribution data
```

```
trials = ['Virtual Trial\n(100K-Sim)', 'MPACT', 'NAPOLI-1', 'FOLFIRINOX\n(PRODIGE)']
```

```
ecog_0_1 = [97, 93, 100, 100] # ECOG 0-1 percentages
```

```
ecog_2 = [3, 7, 0, 0] # ECOG 2 percentages (virtual trial shows underrepresentation)
```

```
# Create figure
```

```
fig, ax = plt.subplots(figsize=(10, 7), facecolor='white')
```

```
# Create stacked bar chart
```

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

```
width = 0.6
```

```
bars1 = ax.bar(x, ecog_0_1, width, label='ECOG 0-1', color='#2ecc71', edgecolor='black', linewidth=1.5)
```

```
bars2 = ax.bar(x, ecog_2, width, bottom=ecog_0_1, label='ECOG 2', color='#e74c3c', edgecolor='black', linewidth=1.5)
```

```
# Add validation failure annotation for virtual trial
```

```
ax.annotate('Validation Failure:\nUnderrepresentation\nof ECOG 2',
```

```
          xy=(0, 98), xytext=(-0.3, 85),
```

```
          arrowprops=dict(arrowstyle='->', color='red', lw=2),
```

```
          fontsize=10, color='red', fontweight='bold', ha='center')
```

```
# Customize plot
```

```
ax.set_ylabel('Percentage of Patients (%)', fontsize=12)
```

```
ax.set_xlabel('Clinical Trials', fontsize=12)
```

```
ax.set_title('ECOG Performance Status Distribution:\nVirtual vs Real-World PDAC Trials',  
            fontsize=14, fontweight='bold')
```

```
ax.set_xticks(x)
```

```
ax.set_xticklabels(trials)
```

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

```
ax.set_ylim(0, 105)
```

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

```

# Add percentage labels
for i, (bar1, bar2) in enumerate(zip(bars1, bars2)):
    # ECOG 0-1 label
    height1 = bar1.get_height()
    if height1 > 5: # Only show if segment is large enough
        ax.text(bar1.get_x() + bar1.get_width()/2, height1/2,
                f'{ecog_0_1[i]}%', ha='center', va='center', fontsize=11, fontweight='bold')

    # ECOG 2 label
    height2 = bar2.get_height()
    if height2 > 5: # Only show if segment is large enough
        ax.text(bar2.get_x() + bar2.get_width()/2, ecog_0_1[i] + height2/2,
                f'{ecog_2[i]}%', ha='center', va='center', fontsize=11, fontweight='bold')

# Add note
ax.text(0.5, -0.15, 'Note: Real-world PDAC populations typically have 10-20% ECOG 2 patients',
        transform=ax.transAxes, fontsize=10, ha='center', style='italic')

plt.tight_layout()
plt.show()
'''

```

6. Heatmap - Key Metrics Comparison Matrix

```

'''python
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Prepare data matrix
treatments = ['Virtual Triplet', 'Virtual Doublet', 'MPACT', 'NAPOLI-1', 'FOLFIRINOX']
metrics = ['OS Benefit\n(months)', 'HR vs Control', 'Grade ≥3 AE\nIncrease (%)', 'ETS Score']

# Data matrix (rows: treatments, columns: metrics)
data = np.array([
    [2.6, 0.69, 17.5, -0.69], # Virtual Triplet
    [1.9, 0.76, np.nan, np.nan], # Virtual Doublet
    [1.8, 0.72, 13.6, 0.00], # MPACT
    [2.0, 0.67, 30.0, -0.20], # NAPOLI-1 (estimated)
    [4.3, 0.57, 25.0, 0.36] # FOLFIRINOX
])

```

```

# Create figure
fig, ax = plt.subplots(figsize=(10, 6), facecolor='white')

# Create custom colormap
cmap = sns.diverging_palette(10, 150, as_cmap=True)

# Create heatmap with annotations
mask = np.isnan(data)
sns.heatmap(data, annot=True, fmt='.2f', cmap=cmap, center=0,
            xticklabels=metrics, yticklabels=treatments,
            cbar_kws={'label': 'Metric Value'}, linewidths=1,
            mask=mask, square=True, ax=ax, vmin=-1, vmax=5)

# Customize
ax.set_title('Comparative Metrics Across PDAC Treatment Regimens', fontsize=14, fontweight='bold', pad=20)

# Rotate x-axis labels
plt.setp(ax.get_xticklabels(), rotation=0, ha='center')
plt.setp(ax.get_yticklabels(), rotation=0)

# Add asterisks for virtual trials
for i, label in enumerate(ax.get_yticklabels()):
    if 'Virtual' in label.get_text():
        label.set_text(label.get_text() + ' *')
        label.set_color('#E63946')

# Add note
ax.text(0.5, -0.15, '* Virtual Trial Arms | Lower HR is better | Higher OS benefit is better | Lower AE increase is better | Higher ETS is better',
       transform=ax.transAxes, fontsize=9, ha='center', style='italic')

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

```

7. Butterfly Plot - OS Benefit vs Toxicity Trade-off

```

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

```

```

Data
treatments = ['FOLFIRINOX', 'Virtual Triplet\n(Arm A)', 'NAPOLI-1', 'Virtual Doublet\n(Arm D)', 'MPACT']
os_benefit = [4.3, 2.6, 2.0, 1.9, 1.8] # Months gained
toxicity_increase = [25.0, 17.5, 30.0, 8.0, 13.6] # % increase in Grade ≥3 AEs (estimated for Doublet)

Create figure
fig, ax = plt.subplots(figsize=(12, 8), facecolor='white')

Y-axis positions
y_pos = np.arange(len(treatments))

Create butterfly plot
Left side - Toxicity (negative values for visualization)
bars_left = ax.barh(y_pos, -np.array(toxicity_increase), color='#e74c3c', alpha=0.7,
 edgecolor='black', linewidth=1.5, label='Toxicity Increase')

Right side - OS Benefit
bars_right = ax.barh(y_pos, os_benefit, color='#2ecc71', alpha=0.7,
 edgecolor='black', linewidth=1.5, label='OS Benefit')

Add center line
ax.axvline(x=0, color='black', linewidth=2)

Customize plot
ax.set_yticks(y_pos)
ax.set_yticklabels(treatments, fontsize=11)
ax.set_xlabel('← Grade ≥3 AE Increase (%) | OS Benefit (months) →', fontsize=12)
ax.set_title('Treatment Trade-offs: Toxicity vs Survival Benefit in PDAC',
 fontsize=14, fontweight='bold')
ax.set_xlim(-35, 5)
ax.grid(axis='x', alpha=0.3)

Add value labels
for i, (tox, os) in enumerate(zip(toxicity_increase, os_benefit)):
 # Toxicity labels
 ax.text(-tox/2, i, f'{tox:.1f}%', ha='center', va='center', fontsize=10, fontweight='bold')
 # OS benefit labels
 ax.text(os/2, i, f'{os:.1f} mo', ha='center', va='center', fontsize=10, fontweight='bold')

Highlight virtual trials

```



```

for i, treatment in enumerate(treatments):
 if 'Virtual' in treatment:
 ax.get_yticklabels()[i].set_color('#E63946')
 ax.get_yticklabels()[i].set_fontweight('bold')

Add interpretation zones
ax.text(-30, len(treatments), 'Higher Toxicity', fontsize=10, ha='center', style='italic', color='#e74c3c')
ax.text(4, len(treatments), 'Greater Benefit', fontsize=10, ha='center', style='italic', color='#2ecc71')

plt.tight_layout()
plt.show()
'''

```

## ## 8. Radar Chart - Multi-dimensional Comparison

```

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

Categories
categories = ['Overall\nSurvival', 'Progression-Free\nSurvival', 'Tolerability\n(inverse toxicity)',
 'Patient\nFitness', 'Biomarker\nSpecificity']

Data for three key regimens (normalized to 0-10 scale)
virtual_triplet = [7.5, 6.0, 2.0, 9.5, 8.0] # High efficacy, poor tolerability, very fit patients, KRAS G12C specific
virtual_doublet = [7.0, 5.5, 7.0, 9.5, 8.0] # Moderate efficacy, better tolerability, very fit patients, KRAS G12C specific
folfirinox = [10.0, 8.0, 4.0, 10.0, 2.0] # Highest efficacy, poor tolerability, fit patients only, no biomarker

Number of variables
num_vars = len(categories)

Compute angle for each axis
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

Complete the circle
virtual_triplet += virtual_triplet[:1]
virtual_doublet += virtual_doublet[:1]
folfirinox += folfirinox[:1]
angles += angles[:1]

Create figure

```

```

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

Plot data
ax.plot(angles, virtual_triplet, 'o-', linewidth=2, color='#E63946', label='Virtual Triplet')
ax.fill(angles, virtual_triplet, alpha=0.25, color='#E63946')

ax.plot(angles, virtual_doublet, 'o-', linewidth=2, color='#F77F00', label='Virtual Doublet')
ax.fill(angles, virtual_doublet, alpha=0.25, color='#F77F00')

ax.plot(angles, folfirinox, 'o-', linewidth=2, color='#2E86AB', label='FOLFIRINOX')
ax.fill(angles, folfirinox, alpha=0.25, color='#2E86AB')

Fix axis to go in the right order and start at 12 o'clock
ax.set_theta_offset(np.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=11)

Set y-axis limits and labels
ax.set_ylim(0, 10)
ax.set_yticks([2, 4, 6, 8, 10])
ax.set_yticklabels(['2', '4', '6', '8', '10'], size=9)

Add gridlines
ax.grid(True, linestyle='--', alpha=0.5)

Add title and legend
ax.set_title('Multi-dimensional Comparison of PDAC Treatment Regimens',
 fontsize=14, fontweight='bold', pad=30)
ax.legend(loc='upper right', bbox_to_anchor=(1.15, 1.15))

Add interpretation note
fig.text(0.5, 0.02, 'Note: Higher values indicate better performance in each dimension',
 ha='center', fontsize=10, style='italic')

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

```

## ## 9. Sankey Diagram - Patient Flow and KRAS G12C Benefit

```
```python
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib.sankey import Sankey

# Create figure
fig, ax = plt.subplots(figsize=(14, 10), facecolor='white')

# Hide axes
ax.axis('off')

# Define positions
y_levels = [0.8, 0.5, 0.2] # Top, Middle, Bottom
x_positions = [0.1, 0.4, 0.7] # Left, Center, Right

# Draw patient population box
pop_box = patches.FancyBboxPatch((0.05, 0.3), 0.15, 0.4,
                                  boxstyle="round,pad=0.02",
                                  facecolor='#ecf0f1', edgecolor='black', linewidth=2)
ax.add_patch(pop_box)
ax.text(0.125, 0.5, '100K Virtual\nPatients\n\n91% KRAS mut\n5% KRAS G12C\n5% KRAS WT',
        ha='center', va='center', fontsize=10, fontweight='bold')

# Draw treatment arms
arms = [
    ('Triplet\n(Arm A)', 0.85, '#E63946'),
    ('Doublet\n(Arm D)', 0.55, '#F77F00'),
    ('Control\n(Arm E)', 0.25, '#95a5a6')
]

for name, y, color in arms:
    arm_box = patches.FancyBboxPatch((0.35, y-0.08), 0.15, 0.12,
                                      boxstyle="round,pad=0.02",
                                      facecolor=color, alpha=0.7, edgecolor='black', linewidth=2)
    ax.add_patch(arm_box)
    ax.text(0.425, y, name, ha='center', va='center', fontsize=10, fontweight='bold')

# Draw outcome boxes with KRAS G12C highlight
outcomes = [
```

```

('KRAS G12C\nEnhanced Benefit\nOS: ~11 months', 0.85, '#2ecc71'),
('Overall Benefit\nOS: 8.7 months', 0.65, '#3498db'),
('Standard Benefit\nOS: ~8.0 months', 0.45, '#3498db'),
('Baseline\nOS: 6.1 months', 0.25, '#bdc3c7')
]

for text, y, color in outcomes:
    outcome_box = patches.FancyBboxPatch((0.65, y-0.08), 0.25, 0.12,
                                          boxstyle="round,pad=0.02",
                                          facecolor=color, alpha=0.7, edgecolor='black', linewidth=2)
    ax.add_patch(outcome_box)
    ax.text(0.775, y, text, ha='center', va='center', fontsize=9)

# Draw arrows showing patient flow
# From population to arms
for y in [0.85, 0.55, 0.25]:
    ax.arrow(0.20, 0.5, 0.13, y-0.5, head_width=0.02, head_length=0.01,
            fc='gray', ec='gray', alpha=0.5, linewidth=2)

# From arms to outcomes
arrows = [
    (0.50, 0.85, 0.13, 0, '#E63946'), # Triplet to KRAS G12C
    (0.50, 0.85, 0.13, -0.2, '#E63946'), # Triplet to overall
    (0.50, 0.55, 0.13, -0.1, '#F77F00'), # Doublet
    (0.50, 0.25, 0.13, 0, '#95a5a6') # Control
]

for x, y, dx, dy, color in arrows:
    ax.arrow(x, y, dx, dy, head_width=0.02, head_length=0.01,
            fc=color, ec=color, alpha=0.7, linewidth=2)

# Add key finding annotation
ax.annotate('KEY FINDING:\nKRAS G12C patients\n(~5% of population)\nachieve ~2.5 month\nadditional OS benefit',
            xy=(0.775, 0.85), xytext=(0.9, 0.7),
            arrowprops=dict(arrowstyle='->', color='#2ecc71', lw=2),
            fontsize=10, color='#2ecc71', fontweight='bold',
            bbox=dict(boxstyle="round,pad=0.3", facecolor='white', edgecolor='#2ecc71'))

# Add title
ax.text(0.5, 0.95, 'Patient Flow Analysis: KRAS G12C Subgroup Benefit in Virtual PDAC Trial',
        ha='center', va='center', fontsize=14, fontweight='bold')

```

```
# Set axis limits
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)

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

10. Timeline Visualization - Evolution of PDAC Treatments

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

Timeline data
trials = [
 ('PRODIGE 4\n(FOLFIRINOX)', 2011, 11.1, 'Real', '#2E86AB'),
 ('IMPACT\n(Gem+nab-P)', 2013, 8.5, 'Real', '#2E86AB'),
 ('NAPOLI-1\n(nal-IRI+5FU)', 2016, 6.2, 'Real', '#2E86AB'),
 ('Digital Twin\nValidation', 2024, 8.5, 'Virtual', '#9B59B6'),
 ('100K Virtual\nTriplet', 2025, 8.7, 'Virtual', '#E63946'),
 ('100K Virtual\nDoublet', 2025, 8.0, 'Virtual', '#F77F00')
]

Create figure
fig, ax = plt.subplots(figsize=(14, 8), facecolor='white')

Timeline
start_year = 2010
end_year = 2026
ax.plot([start_year, end_year], [0, 0], 'k-', linewidth=3)

Add year markers
for year in range(start_year, end_year + 1, 2):
 ax.plot([year, year], [-0.1, 0.1], 'k-', linewidth=2)
 ax.text(year, -0.3, str(year), ha='center', va='top', fontsize=10)

Plot trials
for i, (name, year, os, trial_type, color) in enumerate(trials):
```

```

y_offset = 0.5 + (i % 2) * 0.5 # Alternate heights

Draw stem
ax.plot([year, year], [0, y_offset], color=color, linewidth=2, alpha=0.7)

Draw box
box = patches.FancyBboxPatch((year-0.4, y_offset), 0.8, 0.3,
 boxstyle="round,pad=0.02",
 facecolor=color, alpha=0.8, edgecolor='black', linewidth=1.5)
ax.add_patch(box)

Add text
ax.text(year, y_offset + 0.15, name, ha='center', va='center',
 fontsize=9, fontweight='bold', color='white')
ax.text(year, y_offset + 0.35, f'OS: {os} mo', ha='center', va='bottom',
 fontsize=9, fontweight='bold')

Add era labels
ax.text(2012.5, -0.8, 'Chemotherapy Era', ha='center', fontsize=11,
 style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ecf0f1'))
ax.text(2020, -0.8, 'Precision Medicine Era', ha='center', fontsize=11,
 style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ecf0f1'))
ax.text(2024.5, -0.8, 'Virtual Trial Era', ha='center', fontsize=11,
 style='italic', bbox=dict(boxstyle="round,pad=0.3", facecolor='#ffeaa7'))

Add trend line for OS
years = [2011, 2013, 2016, 2025, 2025]
os_values = [11.1, 8.5, 6.2, 8.7, 8.0]
z = np.polyfit(years, os_values, 2)
p = np.poly1d(z)
years_smooth = np.linspace(2011, 2025, 100)
os_smooth = p(years_smooth)
ax.plot(years_smooth, os_smooth/10 - 0.2, 'g--', alpha=0.5, linewidth=2)

Customize plot
ax.set_xlim(start_year - 0.5, end_year + 0.5)
ax.set_ylim(-1, 1.5)
ax.set_title('Evolution of PDAC Treatment Standards: 2010-2025\n(Timeline shows introduction of key trials with median OS outcomes)',
 fontsize=14, fontweight='bold')
ax.axis('off')

```

```

Add legend
legend_elements = [patches.Patch(facecolor='#2E86AB', label='Real-World Trial'),
 patches.Patch(facecolor='#9B59B6', label='Virtual Validation'),
 patches.Patch(facecolor='#E63946', label='Virtual Hypothesis-Generating')]
ax.legend(handles=legend_elements, loc='upper left')

Add key insight
ax.text(0.98, 0.02, 'Key Insight: Virtual trials (2024-2025) achieve OS comparable to real standards,\nopening new
avenues for rapid hypothesis testing in PDAC',
 transform=ax.transAxes, ha='right', va='bottom', fontsize=10,
 bbox=dict(boxstyle="round,pad=0.3", facecolor='yellow', alpha=0.3))

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

```

These visualizations effectively communicate the key findings from the meta-analysis:

1. **Forest Plot**: Shows the virtual trial HRs are comparable to real trials
2. **Scatter Plot**: Reveals the efficacy-toxicity trade-offs across regimens
3. **Grouped Bar Chart**: Directly compares OS between virtual and real trials
4. **Waterfall Plot**: Highlights which regimens offer favorable risk-benefit profiles
5. **Stacked Bar Chart**: Exposes the patient population discrepancy in the virtual trial
6. **Heatmap**: Provides a comprehensive metric comparison at a glance
7. **Butterfly Plot**: Visualizes the opposing forces of benefit vs toxicity
8. **Radar Chart**: Shows multi-dimensional strengths/weaknesses of key regimens
9. **Sankey Diagram**: Illustrates the KRAS G12C subgroup benefit flow
10. **Timeline**: Places the virtual trial in historical context of PDAC treatment evolution

Each visualization has a white background as requested and effectively communicates specific aspects of the meta-analysis findings.

## **Research Summary Generation 58b1: 363 Words**

The user wants me to create 10 Python visualization scripts based on the comprehensive meta-analysis comparing the 100K-patient virtual PDAC trial with other virtual and real-world clinical trials. Each visualization should have a white background and effectively communicate specific comparisons and findings from the analysis.

Let me go through each requested visualization and extract the relevant data from the document:

1. **Forest Plot** - HR for OS comparing experimental arms vs controls
  - Virtual Triplet (Arm A): HR ~0.69
  - Virtual Doublet (Arm D): HR ~0.76
  - MPACT Gem+nab-P: HR 0.72
  - NAPOLI-1: HR 0.67
  - FOLFIRINOX: HR 0.57
2. **Scatter Plot with Efficiency Frontier** - OS vs Grade  $\geq 3$  AEs
  - Need median OS and AE % for all arms
  - Point sizes proportional to sample size
3. **Grouped Bar Chart** - Median OS by study type
  - Virtual vs Real-world grouping
  - Show control baselines
4. **Waterfall Plot** - ETS scores ranked
  - Virtual Triplet: -0.69
  - MPACT: ~-0.00
  - FOLFIRINOX: +0.36
  - Virtual Doublet: N/A (no toxicity data)
5. **Stacked Bar Chart** - ECOG PS distribution
  - Virtual trial: >95% ECOG 0-1
  - MPACT: ~93% ECOG 0-1, 7% ECOG 2
  - Others mostly 100% ECOG 0-1
6. **Heatmap** - Comparison matrix of key metrics
  - OS benefit, PFS benefit, HR, AE increase, ETS
7. **Butterfly Plot** - OS benefit vs toxicity increase
  - Show trade-offs visually

I'll focus on the remaining visualization approaches, highlighting multi-dimensional comparisons and treatment evolution. The radar chart will capture nuanced performance metrics across different treatment strategies, while the Sankey diagram will illustrate patient flow and molecular targeting benefits. The timeline visualization will provide a comprehensive view of PDAC treatment advancements, marking key trial milestones and survival outcomes.

The radar chart will compare Virtual Triplet, Virtual Doublet, and FOLFIRINOX across critical dimensions like overall survival, progression-free survival, toxicity profiles, patient fitness, and biomarker specificity. This visualization will offer a holistic view of treatment performance.

The Sankey diagram will specifically showcase the patient flow and enhanced benefits for KRAS G12C mutations, with a focus on the Archetype-05 trial's unique insights. This will help visualize molecular targeting precision.

The final timeline visualization will map the evolution of PDAC treatments from 2010 to 2025, highlighting significant trial dates and corresponding overall survival values, providing a historical and predictive perspective on treatment advancements.



## Generation 58b2 (Formatted)

### Opus 4 Extended

27431 characters, 3 minutes 43.91 seconds

I'll create 10 Python scripts to generate visualizations that effectively communicate the key findings from this PDAC meta-analysis. Each script will produce a clear, white-background visualization highlighting specific aspects of the comparison between virtual and real-world trials.

## 1. Forest Plot - Hazard Ratios for Overall Survival

Python

```
import matplotlib.pyplot as plt

import numpy as np

Data for forest plot

studies = ['100K Virtual Triplet', '100K Virtual Doublet', 'MPACT (Gem+nab-P)',
 'NAPOLI-1 (nal-IRI+5FU)', 'FOLFIRINOX']

hazard_ratios = [0.69, 0.76, 0.72, 0.67, 0.57]

lower_ci = [0.62, 0.68, 0.62, 0.56, 0.48] # Estimated CIs

upper_ci = [0.77, 0.85, 0.83, 0.80, 0.67]

study_types = ['Virtual', 'Virtual', 'Real-World', 'Real-World', 'Real-World']

Create figure

fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

Plot forest plot
```

```

y_positions = np.arange(len(studies))

colors = ['#3498db' if st == 'Virtual' else '#e74c3c' for st in study_types]

Plot confidence intervals

for i, (hr, lower, upper, color) in enumerate(zip(hazard_ratios, lower_ci, upper_ci,
colors)):

 ax.plot([lower, upper], [i, i], color=color, linewidth=2, alpha=0.7)

 ax.scatter(hr, i, color=color, s=200, zorder=5, edgecolor='black', linewidth=1)

Add vertical line at HR=1

ax.axvline(x=1, color='gray', linestyle='--', alpha=0.5)

ax.axvline(x=0.7, color='green', linestyle=':', alpha=0.3, label='Clinically
meaningful benefit')

Customize plot

ax.set_yticks(y_positions)

ax.set_yticklabels(studies)

ax.set_xlabel('Hazard Ratio (95% CI)', fontsize=12, fontweight='bold')

ax.set_title('Forest Plot: Overall Survival Hazard Ratios\nVirtual vs Real-World
PDAC Trials',

 fontsize=14, fontweight='bold', pad=20)

```

```

ax.set_xlim(0.4, 1.1)

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

Add legend

virtual_patch = plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor='#3498db',

 markersize=10, label='Virtual Trial')

real_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#e74c3c',

 markersize=10, label='Real-World Trial')

ax.legend(handles=[virtual_patch, real_patch], loc='lower right')

Add text annotations

ax.text(0.5, -1.5, 'Favors Experimental', ha='center', fontsize=10, style='italic')

ax.text(0.95, -1.5, 'Favors Control', ha='center', fontsize=10, style='italic')

plt.tight_layout()

plt.savefig('01_forest_plot_hazard_ratios.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

## 2. Scatter Plot with Efficiency Frontier

Python

```
import matplotlib.pyplot as plt

import numpy as np

from scipy.interpolate import interp1d

Data

treatments = ['100K Triplet', '100K Doublet', '100K Control', 'Gem+nab-P',
 'Gemcitabine',

 'nal-IRI+5FU', '5-FU/LV', 'FOLFIRINOX', 'Gem (PRODIGE)']

median_os = [8.7, 8.0, 6.1, 8.5, 6.7, 6.2, 4.2, 11.1, 6.8]

grade3_ae = [94.0, np.nan, 76.5, 84.0, 70.4, 76.0, 46.0, 75.0, 50.0]

sample_sizes = [20000, 20000, 20000, 431, 430, 117, 149, 171, 171]

study_types = ['Virtual', 'Virtual', 'Virtual', 'Real', 'Real', 'Real', 'Real',
 'Real', 'Real']

Create figure

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

Plot scatter points

for i, (os, ae, size, stype, name) in enumerate(zip(median_os, grade3_ae,
 sample_sizes, study_types, treatments)):

 if not np.isnan(ae):
```

```

color = '#3498db' if stype == 'Virtual' else '#e74c3c'

marker_size = np.sqrt(size) * 0.5 # Scale size

ax.scatter(os, ae, s=marker_size, c=color, alpha=0.7, edgecolor='black',
linewidth=1)

Add labels

if name in ['100K Triplet', 'FOLFIRINOX', 'Gem+nab-P', '100K Doublet']:

 ax.annotate(name, (os, ae), xytext=(5, 5), textcoords='offset points',
fontSize=9)

Plot efficiency frontier

frontier_x = [4.2, 6.2, 8.5, 11.1]

frontier_y = [46.0, 76.0, 84.0, 75.0]

sorted_indices = np.argsort(frontier_x)

frontier_x_sorted = np.array(frontier_x)[sorted_indices]

frontier_y_sorted = np.array(frontier_y)[sorted_indices]

Smooth the frontier

x_smooth = np.linspace(min(frontier_x_sorted), max(frontier_x_sorted), 100)

f = interp1d(frontier_x_sorted, frontier_y_sorted, kind='cubic')

y_smooth = f(x_smooth)

```

```
ax.plot(x_smooth, y_smooth, 'g--', linewidth=2, alpha=0.7, label='Efficiency Frontier')
```

```
Customize plot
```

```
ax.set_xlabel('Median Overall Survival (months)', fontsize=12, fontweight='bold')
```

```
ax.set_ylabel('Grade ≥ 3 Adverse Events (%)', fontsize=12, fontweight='bold')
```

```
ax.set_title('Efficacy-Toxicity Trade-off in PDAC Treatments\n(Bubble size proportional to sample size)',
```

```
 fontsize=14, fontweight='bold', pad=20)
```

```
ax.grid(True, alpha=0.3)
```

```
ax.set_xlim(3, 12)
```

```
ax.set_ylim(40, 100)
```

```
Add legend
```

```
virtual_patch = plt.Line2D([0], [0], marker='o', color='w',
markerfacecolor='#3498db',
```

```
 markersize=10, label='Virtual Trial Arms')
```

```
real_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#e74c3c',
```

```
 markersize=10, label='Real-World Trial Arms')
```

```
ax.legend(handles=[virtual_patch, real_patch], loc='lower right')
```

```

Add ideal direction arrow

ax.annotate('', xy=(11.5, 45), xytext=(10.5, 55),

 arrowprops=dict(arrowstyle='->', color='green', lw=2))

ax.text(11, 50, 'Ideal\nDirection', ha='center', fontsize=10, color='green',
weight='bold')

plt.tight_layout()

plt.savefig('02_scatter_efficiency_frontier.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

### 3. Grouped Bar Chart - Median OS Comparison

Python

```

import matplotlib.pyplot as plt

import numpy as np

Data

virtual_treatments = ['100K Triplet', '100K Doublet', '100K Control']

virtual_os = [8.7, 8.0, 6.1]

```

```
real_treatments = ['FOLFIRINOX', 'Gem+nab-P', 'nal-IRI+5FU', 'Gemcitabine',
'5-FU/LV']
```

```
real_os = [11.1, 8.5, 6.2, 6.7, 4.2]
```

```
Control baseline references
```

```
control_baselines = {'100K': 6.1, 'MPACT': 6.7, 'NAPOLI': 4.2, 'PRODIGE': 6.8}
```

```
Create figure
```

```
fig, ax = plt.subplots(figsize=(12, 8), facecolor='white')
```

```
Set positions
```

```
x_virtual = np.arange(len(virtual_treatments))
```

```
x_real = np.arange(len(real_treatments)) + len(virtual_treatments) + 0.5
```

```
width = 0.35
```

```
Plot bars
```

```
bars1 = ax.bar(x_virtual, virtual_os, width, label='Virtual Trials',
color='#3498db', alpha=0.8)
```

```
bars2 = ax.bar(x_real, real_os, width, label='Real-World Trials', color='#e74c3c',
alpha=0.8)
```



```
Add control baselines
```

```
ax.axhline(y=6.1, color='#3498db', linestyle='--', alpha=0.5, label='Virtual Control
(6.1 mo)')
```

```
ax.axhline(y=6.7, color='#e74c3c', linestyle=':', alpha=0.5, label='Gem Control (6.7
mo)')
```

```
Add value labels on bars
```

```
for bar in bars1 + bars2:
```

```
 height = bar.get_height()
```

```
 ax.text(bar.get_x() + bar.get_width()/2., height + 0.1,
```

```
 f'{height:.1f}', ha='center', va='bottom', fontweight='bold')
```

```
Customize plot
```

```
ax.set_ylabel('Median Overall Survival (months)', fontsize=12, fontweight='bold')
```

```
ax.set_title('Median Overall Survival Across Virtual and Real-World PDAC Trials',
```

```
 fontsize=14, fontweight='bold', pad=20)
```

```
ax.set_xticks(np.concatenate([x_virtual, x_real]))
```

```
ax.set_xticklabels(virtual_treatments + real_treatments, rotation=45, ha='right')
```

```
ax.legend(loc='upper left')
```

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

```
ax.set_ylim(0, 12.5)
```

```

Add separating line

ax.axvline(x=len(virtual_treatments)-0.25, color='gray', linestyle='--', alpha=0.3)

plt.tight_layout()

plt.savefig('03_grouped_bar_median_os.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

## 4. Waterfall Plot - Efficacy-Toxicity Scores

Python

```

import matplotlib.pyplot as plt

import numpy as np

Data

treatments = ['FOLFIRINOX', 'Gem+nab-P', '100K Doublet*', 'NAPOLI-1', '100K
Triplet']

ets_scores = [0.36, 0.00, np.nan, -0.30, -0.69] # Estimated for NAPOLI-1

colors = ['green' if score > 0 else 'red' if not np.isnan(score) else 'gray'

 for score in ets_scores]

```

```
Remove NaN for plotting
```

```
valid_indices = [i for i, score in enumerate(ets_scores) if not np.isnan(score)]
```

```
valid_treatments = [treatments[i] for i in valid_indices]
```

```
valid_scores = [ets_scores[i] for i in valid_indices]
```

```
valid_colors = [colors[i] for i in valid_indices]
```

```
Sort by score
```

```
sorted_indices = np.argsort(valid_scores)[::-1]
```

```
sorted_treatments = [valid_treatments[i] for i in sorted_indices]
```

```
sorted_scores = [valid_scores[i] for i in sorted_indices]
```

```
sorted_colors = [valid_colors[i] for i in sorted_indices]
```

```
Create figure
```

```
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')
```

```
Create waterfall plot
```

```
x_pos = np.arange(len(sorted_treatments))
```

```
bars = ax.bar(x_pos, sorted_scores, color=sorted_colors, alpha=0.8,
edgecolor='black', linewidth=1.5)
```

```
Add zero line
```

```
ax.axhline(y=0, color='black', linestyle='-', linewidth=2)
```

```
Add value labels
```

```
for bar, score in zip(bars, sorted_scores):
```

```
 height = bar.get_height()
```

```
 label_y = height + 0.02 if height > 0 else height - 0.02
```

```
 va = 'bottom' if height > 0 else 'top'
```

```
 ax.text(bar.get_x() + bar.get_width()/2., label_y,
```

```
 f'{score:.2f}', ha='center', va=va, fontweight='bold', fontsize=10)
```

```
Customize plot
```

```
ax.set_xticks(x_pos)
```

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

```
ax.set_ylabel('Efficacy-Toxicity Score (ETS)', fontsize=12, fontweight='bold')
```

```
ax.set_title('Efficacy-Toxicity Score Ranking\nHigher scores indicate better
benefit-risk profile',
```

```
 fontsize=14, fontweight='bold', pad=20)
```

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

```
ax.set_ylim(-0.9, 0.5)
```

```
Add annotations
```

```
ax.text(0.5, 0.4, 'Favorable\nBenefit-Risk', ha='center', fontsize=10,
 color='green', weight='bold', transform=ax.transAxes)
```

```
ax.text(0.5, 0.1, 'Unfavorable\nBenefit-Risk', ha='center', fontsize=10,
 color='red', weight='bold', transform=ax.transAxes)
```

```
Add note about doublet
```

```
ax.text(0.02, 0.98, '*100K Doublet: ETS not calculated (toxicity data unavailable)',
 transform=ax.transAxes, fontsize=9, va='top', style='italic')
```

```
plt.tight_layout()
```

```
plt.savefig('04_waterfall_ets_scores.png', dpi=300, bbox_inches='tight',
 facecolor='white')
```

```
plt.show()
```

## 5. Stacked Bar Chart - ECOG Performance Status Distribution

Python

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
Data
```

```
trials = ['100K Virtual\n(All Arms)', 'MPACT', 'NAPOLI-1', 'FOLFIRINOX']
```

```
ecog_0_1 = [97, 93, 100, 100] # Percentage
```

```
ecog_2 = [0, 7, 0, 0] # Percentage (100K had <5% which rounds to 0)
```

```
ecog_missing = [3, 0, 0, 0] # Remainder
```

```
Create figure
```

```
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')
```

```
Create stacked bars
```

```
x_pos = np.arange(len(trials))
```

```
width = 0.6
```

```
p1 = ax.bar(x_pos, ecog_0_1, width, label='ECOG 0-1', color='#2ecc71', alpha=0.8)
```

```
p2 = ax.bar(x_pos, ecog_2, width, bottom=ecog_0_1, label='ECOG 2', color='#f39c12',
alpha=0.8)
```

```
p3 = ax.bar(x_pos, ecog_missing, width, bottom=np.array(ecog_0_1)+np.array(ecog_2),
 label='Not Specified', color='#95a5a6', alpha=0.8)
```

```
Add percentage labels
```

```

for i, (bar1, bar2) in enumerate(zip(p1, p2)):

 # ECOG 0-1 label

 height1 = bar1.get_height()

 if height1 > 5:

 ax.text(bar1.get_x() + bar1.get_width()/2., height1/2,

 f'{ecog_0_1[i]}%', ha='center', va='center', fontweight='bold',
fontsize=12)

 # ECOG 2 label

 height2 = bar2.get_height()

 if height2 > 0:

 ax.text(bar2.get_x() + bar2.get_width()/2., ecog_0_1[i] + height2/2,

 f'{ecog_2[i]}%', ha='center', va='center', fontweight='bold',
fontsize=11)

 # Add validation failure annotation

 ax.annotate('VALIDATION\nFAILURE', xy=(0, 97), xytext=(-0.5, 85),

 arrowprops=dict(arrowstyle='->', color='red', lw=2),

 fontsize=10, color='red', weight='bold', ha='center')

 # Customize plot

```

```

ax.set_ylabel('Percentage of Patients (%)', fontsize=12, fontweight='bold')

ax.set_title('ECOG Performance Status Distribution Across Trials\nHighlighting\nVirtual Trial Patient Selection Bias',

 fontsize=14, fontweight='bold', pad=20)

ax.set_xticks(x_pos)

ax.set_xticklabels(trials)

ax.legend(loc='upper right')

ax.set_ylim(0, 105)

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

Add note

ax.text(0.5, -0.15, 'Note: 100K Virtual Trial underrepresented ECOG 2 patients\ncompared to real-world data',

 transform=ax.transAxes, ha='center', fontsize=10, style='italic')

plt.tight_layout()

plt.savefig('05_stacked_bar_ecog_distribution.png', dpi=300, bbox_inches='tight',
 facecolor='white')

plt.show()

```

## 6. Heatmap - Key Metrics Comparison



Python

```
import matplotlib.pyplot as plt

import numpy as np

import seaborn as sns

Data matrix

treatments = ['100K Triplet', '100K Doublet', 'FOLFIRINOX', 'Gem+nab-P',
 'nal-IRI+5FU']

metrics = ['OS Benefit\n(months)', 'HR vs Control', 'Grade ≥3 AE\nIncrease (%)',
 'ETS Score']

Create data matrix

data = np.array([

 [2.6, 0.69, 17.5, -0.69], # 100K Triplet

 [1.9, 0.76, np.nan, np.nan], # 100K Doublet

 [4.3, 0.57, 25.0, 0.36], # FOLFIRINOX

 [1.8, 0.72, 13.6, 0.00], # Gem+nab-P

 [1.9, 0.67, 30.0, -0.30] # nal-IRI+5FU (estimated)

])

Create figure

fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')
```

```
Create custom colormap
```

```
cmap = plt.cm.RdYlGn_r # Reversed so lower HR is green
```

```
Plot heatmap
```

```
sns.heatmap(data.T, annot=True, fmt='.2f', cmap=cmap, center=0,

 xticklabels=treatments, yticklabels=metrics,

 cbar_kws={'label': 'Standardized Value'},

 mask=np.isnan(data.T), square=True, linewidths=1,

 annot_kws={'fontsize': 11, 'fontweight': 'bold'})
```

```
Customize plot
```

```
ax.set_title('Comparative Metrics Heatmap: Virtual vs Real-World PDAC Trials',

 fontsize=14, fontweight='bold', pad=20)

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

```
Add annotations for interpretation
```

```
ax.text(0.5, -0.15, 'Green = Favorable | Red = Unfavorable | Gray = Not Reported',

 transform=ax.transAxes, ha='center', fontsize=10, style='italic')
```

```
plt.tight_layout()

plt.savefig('06_heatmap_metrics_comparison.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()
```

## 7. Butterfly Plot - OS Benefit vs Toxicity Trade-off

Python

```
import matplotlib.pyplot as plt

import numpy as np

Data

treatments = ['100K Triplet', '100K Doublet', 'FOLFIRINOX', 'Gem+nab-P',
'nal-IRI+5FU']

os_benefit = [2.6, 1.9, 4.3, 1.8, 1.9] # Months gained

toxicity_increase = [17.5, np.nan, 25.0, 13.6, 30.0] # % increase in Grade ≥3 AEs

Create figure

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

Set up positions

y_pos = np.arange(len(treatments))
```

```

Plot toxicity on the left (negative values)

toxicity_valid = []

y_pos_valid = []

for i, (tox, treat) in enumerate(zip(toxicity_increase, treatments)):

 if not np.isnan(tox):

 toxicity_valid.append(-tox) # Negative for left side

 y_pos_valid.append(i)

bars_left = ax.barh(y_pos_valid, toxicity_valid, color='#e74c3c', alpha=0.8,
label='Toxicity Increase')

Plot OS benefit on the right

bars_right = ax.barh(y_pos, os_benefit, color='#2ecc71', alpha=0.8, label='OS
Benefit')

Add center line

ax.axvline(x=0, color='black', linewidth=2)

Add value labels

for bar in bars_left:

```

```
width = bar.get_width()
```

```
ax.text(width - 1, bar.get_y() + bar.get_height()/2,
```

```
 f'{-width:.1f}%', ha='right', va='center', fontweight='bold')
```

```
for bar in bars_right:
```

```
 width = bar.get_width()
```

```
 ax.text(width + 0.1, bar.get_y() + bar.get_height()/2,
```

```
 f'{width:.1f} mo', ha='left', va='center', fontweight='bold')
```

```
Customize plot
```

```
ax.set_yticks(y_pos)
```

```
ax.set_yticklabels(treatments)
```

```
ax.set_xlabel('← Toxicity Increase (%) | OS Benefit (months) →', fontsize=12,
fontweight='bold')
```

```
ax.set_title('Butterfly Plot: Treatment Benefit vs Toxicity Trade-off\n100K Virtual
Trial vs Real-World Standards',
```

```
 fontsize=14, fontweight='bold', pad=20)
```

```
ax.set_xlim(-35, 5)
```

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

```
Add annotations
```

```

ax.text(-17.5, -0.7, 'Higher Toxicity', ha='center', fontsize=10, color='#e74c3c',
weight='bold')

ax.text(2.5, -0.7, 'Greater Benefit', ha='center', fontsize=10, color='#2ecc71',
weight='bold')

Add note for missing data

ax.text(0.5, 0.02, '*100K Doublet: Toxicity data not reported',

 transform=ax.transAxes, fontsize=9, style='italic')

plt.tight_layout()

plt.savefig('07_butterfly_plot_benefit_toxicity.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

## 8. Radar Chart - Multi-dimensional Comparison

Python

```

import matplotlib.pyplot as plt

import numpy as np

Data

categories = ['OS Benefit\n(normalized)', 'Low Toxicity\n(inverse)',
'Patient\nFitness',

```

```
'Biomarker\nSpecificity', 'Clinical\nFeasibility']
```

```
Scores (0-100 scale)
```

```
triplet_scores = [60, 20, 95, 90, 40] # High biomarker specificity, high toxicity
```

```
douplet_scores = [45, 70, 95, 90, 80] # Moderate OS, lower toxicity assumed
```

```
folfirinox_scores = [100, 40, 100, 20, 60] # Highest OS, high toxicity, no
biomarker
```

```
Number of variables
```

```
num_vars = len(categories)
```

```
Compute angle for each axis
```

```
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()
```

```
triplet_scores += triplet_scores[:1]
```

```
douplet_scores += douplet_scores[:1]
```

```
folfirinox_scores += folfirinox_scores[:1]
```

```
angles += angles[:1]
```

```
Create figure
```

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

```
Plot data
```

```
ax.plot(angles, triplet_scores, 'o-', linewidth=2, label='100K Triplet',
color='#3498db')
```

```
ax.fill(angles, triplet_scores, alpha=0.25, color='#3498db')
```

```
ax.plot(angles, doublet_scores, 'o-', linewidth=2, label='100K Doublet',
color='#9b59b6')
```

```
ax.fill(angles, doublet_scores, alpha=0.25, color='#9b59b6')
```

```
ax.plot(angles, folfirinox_scores, 'o-', linewidth=2, label='FOLFIRINOX',
color='#e74c3c')
```

```
ax.fill(angles, folfirinox_scores, alpha=0.25, color='#e74c3c')
```

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

```
ax.set_theta_offset(np.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=11, weight='bold')
```



```
Set y-axis limits and labels

ax.set_ylim(0, 100)

ax.set_yticks([20, 40, 60, 80, 100])

ax.set_yticklabels(['20', '40', '60', '80', '100'], size=8)

ax.grid(True)

Add legend and title

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

plt.title('Multi-dimensional Comparison of PDAC Treatment Approaches',

 size=14, weight='bold', pad=30)

Add explanatory note

fig.text(0.5, 0.02, 'Higher scores indicate better performance in each dimension',

 ha='center', fontsize=10, style='italic')

plt.tight_layout()

plt.savefig('08_radar_chart_multidimensional.png', dpi=300, bbox_inches='tight',
 facecolor='white')

plt.show()
```

## 9. Sankey Diagram - KRAS G12C Patient Flow

Python

```
import matplotlib.pyplot as plt

import matplotlib.patches as patches

from matplotlib.patches import FancyBboxPatch, ConnectionPatch

import numpy as np

fig, ax = plt.subplots(figsize=(14, 10), facecolor='white')

Define positions

col1_x, col2_x, col3_x = 0.1, 0.5, 0.9

y_positions = np.linspace(0.8, 0.2, 5)

Column 1: Patient populations

populations = ['All PDAC\nPatients\n(100%)', 'KRAS Mutant\n(91%)', 'KRAS WT\n(9%)',

 'KRAS G12C\n(5%)', 'Other KRAS\n(86%)']

pop_sizes = [100, 91, 9, 5, 86]

Column 2: Treatment arms

treatments = ['100K Triplet\n(Dara+Mita+nal-IRI)', '100K Doublet\n(Dara+Mita)',

 'Control\n(nal-IRI+5FU)']
```

```
Column 3: Outcomes
```

```
outcomes = ['Enhanced Benefit\n(HR ~0.5)', 'Moderate Benefit\n(HR ~0.7-0.8)',
 'Minimal Benefit\n(HR ~0.9-1.0)']
```

```
Draw boxes
```

```
Population boxes
```

```
for i, (pop, size) in enumerate(zip(populations[:3], pop_sizes[:3])):
```

```
 height = size/100 * 0.6
```

```
 y = 0.2 + (i * 0.25)
```

```
 color = '#3498db' if 'KRAS' in pop and 'WT' not in pop else '#95a5a6'
```

```
 box = FancyBboxPatch((col1_x-0.08, y), 0.16, height,
```

```
 boxstyle="round,pad=0.01",
```

```
 facecolor=color, alpha=0.7, edgecolor='black')
```

```
 ax.add_patch(box)
```

```
 ax.text(col1_x, y+height/2, pop, ha='center', va='center', fontweight='bold',
 fontsize=10)
```

```
KRAS subtypes
```

```
g12c_box = FancyBboxPatch((col1_x-0.05, 0.15), 0.1, 0.05,
```

```

 boxstyle="round,pad=0.01",

 facecolor='#e74c3c', alpha=0.8, edgecolor='black')

ax.add_patch(g12c_box)

ax.text(col1_x, 0.175, 'KRAS G12C\n(5%)', ha='center', va='center',
fontweight='bold', fontsize=9)

Treatment boxes

treatment_colors = ['#2ecc71', '#f39c12', '#95a5a6']

for i, (treat, color) in enumerate(zip(treatments, treatment_colors)):

 y = 0.3 + (i * 0.2)

 box = FancyBboxPatch((col2_x-0.08, y), 0.16, 0.15,

 boxstyle="round,pad=0.01",

 facecolor=color, alpha=0.7, edgecolor='black')

 ax.add_patch(box)

 ax.text(col2_x, y+0.075, treat, ha='center', va='center', fontweight='bold',
 fontsize=10)

Outcome boxes

outcome_colors = ['#27ae60', '#f39c12', '#c0392b']

for i, (outcome, color) in enumerate(zip(outcomes, outcome_colors)):

 y = 0.3 + (i * 0.2)

```



```

 shrinkA=5, shrinkB=5, mutation_scale=20,

 fc="green", alpha=0.6, linewidth=10)

ax.add_artist(con1)

Other connections (thinner lines for moderate/minimal benefit)

con2 = ConnectionPatch((col2_x+0.08, 0.575), (col3_x-0.08, 0.575),

 "data", "data", arrowstyle="->",

 shrinkA=5, shrinkB=5, mutation_scale=20,

 fc="orange", alpha=0.5, linewidth=6)

ax.add_artist(con2)

Title and labels

ax.text(col1_x, 0.95, 'Patient\nCharacteristics', ha='center', fontsize=12,
 fontweight='bold')

ax.text(col2_x, 0.95, 'Treatment\nAssignment', ha='center', fontsize=12,
 fontweight='bold')

ax.text(col3_x, 0.95, 'Treatment\nOutcome', ha='center', fontsize=12,
 fontweight='bold')

ax.set_xlim(0, 1)

ax.set_ylim(0, 1)

```

```

ax.axis('off')

plt.title('Patient Flow: KRAS G12C Subgroup Drives Enhanced Benefit from
Daraxonrasib-Containing Regimens',

 fontsize=14, fontweight='bold', pad=20)

Add key finding

ax.text(0.5, 0.05, 'Key Finding: KRAS G12C patients (Archetype-05) showed enhanced
response to Daraxonrasib combinations',

 ha='center', fontsize=11, style='italic', transform=ax.transAxes)

plt.tight_layout()

plt.savefig('09_sankey_kras_patient_flow.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

## 10. Timeline Visualization - Evolution of PDAC Treatment

Python

```

import matplotlib.pyplot as plt

import matplotlib.patches as patches

import numpy as np

```

```

fig, ax = plt.subplots(figsize=(14, 10), facecolor='white')

Timeline data

years = [2010, 2011, 2013, 2015, 2016, 2020, 2024, 2025]

events = [

 'Gemcitabine Era',

 'FOLFIRINOX\n(OS: 11.1 mo)',

 'MPACT: Gem+nab-P\n(OS: 8.5 mo)',

 'NAPOLI-1: nal-IRI\n(2nd line, OS: 6.2 mo)',

 'Continued refinement',

 'Precision medicine\napproaches emerge',

 'KRAS G12C inhibitors\nenter trials',

 '100K Virtual Trial\n(Triplet OS: 8.7 mo)'

]

os_values = [6.8, 11.1, 8.5, 6.2, None, None, None, 8.7]

trial_types = ['standard', 'breakthrough', 'improvement', 'second-line',

 'era', 'era', 'era', 'virtual']

Color mapping

```



```
color_map = {

 'standard': '#95a5a6',

 'breakthrough': '#27ae60',

 'improvement': '#3498db',

 'second-line': '#f39c12',

 'era': '#ecf0f1',

 'virtual': '#9b59b6'

}
```

```
Draw timeline
```

```
timeline_y = 0.5
```

```
ax.plot([2009, 2026], [timeline_y, timeline_y], 'k-', linewidth=3)
```

```
Add events
```

```
for i, (year, event, os, ttype) in enumerate(zip(years, events, os_values,
trial_types)):
```

```
 # Draw marker
```

```
 marker_size = 15 if ttype != 'era' else 8
```

```
 ax.scatter(year, timeline_y, s=marker_size**2, c=color_map[ttype],
```

```
 edgecolor='black', linewidth=2, zorder=5)
```

```
Add event label
```

```
y_offset = 0.15 if i % 2 == 0 else -0.15
```

```
ax.text(year, timeline_y + y_offset, event, ha='center', va='center',
```

```
 fontweight='bold', fontsize=10,
```

```
 bbox=dict(boxstyle="round,pad=0.3", facecolor='white', alpha=0.8))
```

```
Add OS value if available
```

```
if os is not None:
```

```
 os_y = timeline_y + y_offset + (0.08 if y_offset > 0 else -0.08)
```

```
 ax.text(year, os_y, f'OS: {os} mo', ha='center', va='center',
```

```
 fontsize=9, style='italic', color='darkred')
```

```
Connect to timeline
```

```
ax.plot([year, year], [timeline_y, timeline_y + y_offset*0.8],
```

```
 'k--', alpha=0.3, linewidth=1)
```

```
Add era labels
```

```
ax.text(2010, 0.85, 'Single-Agent Era', fontsize=12, fontweight='bold', ha='center')
```

```
ax.text(2013, 0.85, 'Combination Era', fontsize=12, fontweight='bold', ha='center')
```

```
ax.text(2022, 0.85, 'Precision/Virtual Era', fontsize=12, fontweight='bold',
ha='center')
```

```
Add progression arrow
```

```
ax.annotate('', xy=(2025, 0.15), xytext=(2010, 0.15),
 arrowprops=dict(arrowstyle='->', color='green', lw=3))
```

```
ax.text(2017.5, 0.1, 'Increasing Treatment Complexity & Personalization',
 ha='center', fontsize=11, color='green', style='italic')
```

```
Highlight virtual trial
```

```
virtual_box = patches.Rectangle((2024.5, 0.3), 1, 0.4,
 linewidth=2, edgecolor='purple',
 facecolor='purple', alpha=0.1)
```

```
ax.add_patch(virtual_box)
```

```
Customize plot
```

```
ax.set_xlim(2009, 2026)
```

```
ax.set_ylim(0, 1)
```

```
ax.set_xlabel('Year', fontsize=12, fontweight='bold')
```

```
ax.set_title('Evolution of PDAC Treatment Standards (2010-2025)\nMedian Overall
Survival Progression',
```

```
fontsize=14, fontweight='bold', pad=20)
```

```
Remove y-axis
```

```
ax.get_yaxis().set_visible(False)
```

```
ax.spines['left'].set_visible(False)
```

```
ax.spines['right'].set_visible(False)
```

```
ax.spines['top'].set_visible(False)
```

```
Add legend
```

```
legend_elements = [
```

```
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#27ae60',
 markersize=10, label='Breakthrough Trial'),
```

```
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#3498db',
 markersize=10, label='Incremental Advance'),
```

```
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#9b59b6',
 markersize=10, label='Virtual Trial')
```

```
]
```

```
ax.legend(handles=legend_elements, loc='upper left')
```

```
Add note
```

```

ax.text(0.5, -0.1, 'Note: 100K Virtual Trial demonstrates comparable efficacy to
current standards, suggesting validity of simulation approach',

 transform=ax.transAxes, ha='center', fontsize=10, style='italic')

plt.tight_layout()

plt.savefig('10_timeline_pdac_evolution.png', dpi=300, bbox_inches='tight',
facecolor='white')

plt.show()

```

These visualizations effectively communicate the key findings from the meta-analysis:

1. **Forest Plot** shows the virtual trial HRs align well with real-world trials
2. **Efficiency Frontier** reveals the virtual triplet's poor toxicity-efficacy balance
3. **Grouped Bar Chart** demonstrates comparable OS outcomes between virtual and real trials
4. **Waterfall Plot** ranks treatments by their efficacy-toxicity scores
5. **ECOG Distribution** highlights the critical patient selection bias in the virtual trial
6. **Heatmap** provides an at-a-glance comparison of all key metrics
7. **Butterfly Plot** visually contrasts benefit gains against toxicity costs
8. **Radar Chart** shows multi-dimensional strengths/weaknesses of each approach
9. **Sankey Diagram** illustrates how KRAS G12C patients drive the virtual triplet's benefit
10. **Timeline** contextualizes the virtual trial within PDAC treatment evolution

Each visualization has a clean white background and uses color strategically to highlight key insights from the meta-analysis.

## **Generation 58b2 (Plain Text)**

### **Opus 4 Extended**

#### **27652 characters, 3 minutes 43.91 seconds**

I'll create 10 Python scripts to generate visualizations that effectively communicate the key findings from this PDAC meta-analysis. Each script will produce a clear, white-background visualization highlighting specific aspects of the comparison between virtual and real-world trials.

## 1. Forest Plot - Hazard Ratios for Overall Survival

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

# Data for forest plot
studies = ['100K Virtual Triplet', '100K Virtual Doublet', 'MPACT (Gem+nab-P)',
           'NAPOLI-1 (nal-IRI+5FU)', 'FOLFIRINOX']
hazard_ratios = [0.69, 0.76, 0.72, 0.67, 0.57]
lower_ci = [0.62, 0.68, 0.62, 0.56, 0.48] # Estimated CIs
upper_ci = [0.77, 0.85, 0.83, 0.80, 0.67]
study_types = ['Virtual', 'Virtual', 'Real-World', 'Real-World', 'Real-World']

# Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

# Plot forest plot
y_positions = np.arange(len(studies))
colors = ['#3498db' if st == 'Virtual' else '#e74c3c' for st in study_types]

# Plot confidence intervals
for i, (hr, lower, upper, color) in enumerate(zip(hazard_ratios, lower_ci, upper_ci, colors)):
    ax.plot([lower, upper], [i, i], color=color, linewidth=2, alpha=0.7)
    ax.scatter(hr, i, color=color, s=200, zorder=5, edgecolor='black', linewidth=1)

# Add vertical line at HR=1
ax.axvline(x=1, color='gray', linestyle='--', alpha=0.5)
ax.axvline(x=0.7, color='green', linestyle=':', alpha=0.3, label='Clinically meaningful benefit')

# Customize plot
ax.set_yticks(y_positions)
ax.set_yticklabels(studies)
```

```

ax.set_xlabel('Hazard Ratio (95% CI)', fontsize=12, fontweight='bold')
ax.set_title('Forest Plot: Overall Survival Hazard Ratios\nVirtual vs Real-World PDAC Trials',
             fontsize=14, fontweight='bold', pad=20)
ax.set_xlim(0.4, 1.1)
ax.grid(axis='x', alpha=0.3)

# Add legend
virtual_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#3498db',
                           markersize=10, label='Virtual Trial')
real_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#e74c3c',
                        markersize=10, label='Real-World Trial')
ax.legend(handles=[virtual_patch, real_patch], loc='lower right')

# Add text annotations
ax.text(0.5, -1.5, 'Favors Experimental', ha='center', fontsize=10, style='italic')
ax.text(0.95, -1.5, 'Favors Control', ha='center', fontsize=10, style='italic')

plt.tight_layout()
plt.savefig('01_forest_plot_hazard_ratios.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()

```

2. Scatter Plot with Efficiency Frontier

```

```python
import matplotlib.pyplot as plt
import numpy as np
from scipy.interpolate import interp1d

Data
treatments = ['100K Triplet', '100K Doublet', '100K Control', 'Gem+nab-P', 'Gemcitabine',
 'nal-IRI+5FU', '5-FU/LV', 'FOLFIRINOX', 'Gem (PRODIGE)']
median_os = [8.7, 8.0, 6.1, 8.5, 6.7, 6.2, 4.2, 11.1, 6.8]
grade3_ae = [94.0, np.nan, 76.5, 84.0, 70.4, 76.0, 46.0, 75.0, 50.0]
sample_sizes = [20000, 20000, 20000, 431, 430, 117, 149, 171, 171]
study_types = ['Virtual', 'Virtual', 'Virtual', 'Real', 'Real', 'Real', 'Real', 'Real', 'Real']

Create figure
fig, ax = plt.subplots(figsize=(12, 8), facecolor='white')

Plot scatter points

```

```

for i, (os, ae, size, stype, name) in enumerate(zip(median_os, grade3_ae, sample_sizes, study_types, treatments)):
 if not np.isnan(ae):
 color = '#3498db' if stype == 'Virtual' else '#e74c3c'
 marker_size = np.sqrt(size) * 0.5 # Scale size
 ax.scatter(os, ae, s=marker_size, c=color, alpha=0.7, edgecolor='black', linewidth=1)

 # Add labels
 if name in ['100K Triplet', 'FOLFIRINOX', 'Gem+nab-P', '100K Doublet']:
 ax.annotate(name, (os, ae), xytext=(5, 5), textcoords='offset points', fontsize=9)

Plot efficiency frontier
frontier_x = [4.2, 6.2, 8.5, 11.1]
frontier_y = [46.0, 76.0, 84.0, 75.0]
sorted_indices = np.argsort(frontier_x)
frontier_x_sorted = np.array(frontier_x)[sorted_indices]
frontier_y_sorted = np.array(frontier_y)[sorted_indices]

Smooth the frontier
x_smooth = np.linspace(min(frontier_x_sorted), max(frontier_x_sorted), 100)
f = interp1d(frontier_x_sorted, frontier_y_sorted, kind='cubic')
y_smooth = f(x_smooth)

ax.plot(x_smooth, y_smooth, 'g--', linewidth=2, alpha=0.7, label='Efficiency Frontier')

Customize plot
ax.set_xlabel('Median Overall Survival (months)', fontsize=12, fontweight='bold')
ax.set_ylabel('Grade ≥3 Adverse Events (%)', fontsize=12, fontweight='bold')
ax.set_title('Efficacy-Toxicity Trade-off in PDAC Treatments\n(Bubble size proportional to sample size)',
 fontsize=14, fontweight='bold', pad=20)
ax.grid(True, alpha=0.3)
ax.set_xlim(3, 12)
ax.set_ylim(40, 100)

Add legend
virtual_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#3498db',
 markersize=10, label='Virtual Trial Arms')
real_patch = plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#e74c3c',
 markersize=10, label='Real-World Trial Arms')
ax.legend(handles=[virtual_patch, real_patch], loc='lower right')

Add ideal direction arrow

```



```

ax.annotate("", xy=(11.5, 45), xytext=(10.5, 55),
 arrowprops=dict(arrowstyle='->', color='green', lw=2))
ax.text(11, 50, 'Ideal\nDirection', ha='center', fontsize=10, color='green', weight='bold')

plt.tight_layout()
plt.savefig('02_scatter_efficiency_frontier.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()
'''

```

### ## 3. Grouped Bar Chart - Median OS Comparison

```

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

Data
virtual_treatments = ['100K Triplet', '100K Doublet', '100K Control']
virtual_os = [8.7, 8.0, 6.1]

real_treatments = ['FOLFIRINOX', 'Gem+nab-P', 'nal-IRI+5FU', 'Gemcitabine', '5-FU/LV']
real_os = [11.1, 8.5, 6.2, 6.7, 4.2]

Control baseline references
control_baselines = {'100K': 6.1, 'MPACT': 6.7, 'NAPOLI': 4.2, 'PRODIGE': 6.8}

Create figure
fig, ax = plt.subplots(figsize=(12, 8), facecolor='white')

Set positions
x_virtual = np.arange(len(virtual_treatments))
x_real = np.arange(len(real_treatments)) + len(virtual_treatments) + 0.5
width = 0.35

Plot bars
bars1 = ax.bar(x_virtual, virtual_os, width, label='Virtual Trials', color='#3498db', alpha=0.8)
bars2 = ax.bar(x_real, real_os, width, label='Real-World Trials', color='#e74c3c', alpha=0.8)

Add control baselines
ax.axhline(y=6.1, color='#3498db', linestyle='--', alpha=0.5, label='Virtual Control (6.1 mo)')
ax.axhline(y=6.7, color='#e74c3c', linestyle=':', alpha=0.5, label='Gem Control (6.7 mo)')

```

```

Add value labels on bars
for bar in bars1 + bars2:
 height = bar.get_height()
 ax.text(bar.get_x() + bar.get_width()/2., height + 0.1,
 f'{height:.1f}', ha='center', va='bottom', fontweight='bold')

Customize plot
ax.set_ylabel('Median Overall Survival (months)', fontsize=12, fontweight='bold')
ax.set_title('Median Overall Survival Across Virtual and Real-World PDAC Trials',
 fontsize=14, fontweight='bold', pad=20)
ax.set_xticks(np.concatenate([x_virtual, x_real]))
ax.set_xticklabels(virtual_treatments + real_treatments, rotation=45, ha='right')
ax.legend(loc='upper left')
ax.grid(axis='y', alpha=0.3)
ax.set_ylim(0, 12.5)

Add separating line
ax.axvline(x=len(virtual_treatments)-0.25, color='gray', linestyle='-', alpha=0.3)

plt.tight_layout()
plt.savefig('03_grouped_bar_median_os.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()

```

#### ## 4. Waterfall Plot - Efficacy-Toxicity Scores

```

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

# Data
treatments = ['FOLFIRINOX', 'Gem+nab-P', '100K Doublet*', 'NAPOLI-1', '100K Triplet']
ets_scores = [0.36, 0.00, np.nan, -0.30, -0.69] # Estimated for NAPOLI-1
colors = ['green' if score > 0 else 'red' if not np.isnan(score) else 'gray'
          for score in ets_scores]

# Remove NaN for plotting
valid_indices = [i for i, score in enumerate(ets_scores) if not np.isnan(score)]
valid_treatments = [treatments[i] for i in valid_indices]
valid_scores = [ets_scores[i] for i in valid_indices]
valid_colors = [colors[i] for i in valid_indices]

```

```

# Sort by score
sorted_indices = np.argsort(valid_scores)[::-1]
sorted_treatments = [valid_treatments[i] for i in sorted_indices]
sorted_scores = [valid_scores[i] for i in sorted_indices]
sorted_colors = [valid_colors[i] for i in sorted_indices]

# Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

# Create waterfall plot
x_pos = np.arange(len(sorted_treatments))
bars = ax.bar(x_pos, sorted_scores, color=sorted_colors, alpha=0.8, edgecolor='black', linewidth=1.5)

# Add zero line
ax.axhline(y=0, color='black', linestyle='-', linewidth=2)

# Add value labels
for bar, score in zip(bars, sorted_scores):
    height = bar.get_height()
    label_y = height + 0.02 if height > 0 else height - 0.02
    va = 'bottom' if height > 0 else 'top'
    ax.text(bar.get_x() + bar.get_width()/2., label_y,
            f'{score:.2f}', ha='center', va=va, fontweight='bold', fontsize=10)

# Customize plot
ax.set_xticks(x_pos)
ax.set_xticklabels(sorted_treatments, rotation=45, ha='right')
ax.set_ylabel('Efficacy-Toxicity Score (ETS)', fontsize=12, fontweight='bold')
ax.set_title('Efficacy-Toxicity Score Ranking\nHigher scores indicate better benefit-risk profile',
            fontsize=14, fontweight='bold', pad=20)
ax.grid(axis='y', alpha=0.3)
ax.set_ylim(-0.9, 0.5)

# Add annotations
ax.text(0.5, 0.4, 'Favorable\nBenefit-Risk', ha='center', fontsize=10,
        color='green', weight='bold', transform=ax.transAxes)
ax.text(0.5, 0.1, 'Unfavorable\nBenefit-Risk', ha='center', fontsize=10,
        color='red', weight='bold', transform=ax.transAxes)

# Add note about doublet

```

```
ax.text(0.02, 0.98, '*100K Doublet: ETS not calculated (toxicity data unavailable)',
        transform=ax.transAxes, fontsize=9, va='top', style='italic')
```

```
plt.tight_layout()
plt.savefig('04_waterfall_ets_scores.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()
...
```

5. Stacked Bar Chart - ECOG Performance Status Distribution

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

Data
trials = ['100K Virtual\n(All Arms)', 'MPACT', 'NAPOLI-1', 'FOLFIRINOX']
ecog_0_1 = [97, 93, 100, 100] # Percentage
ecog_2 = [0, 7, 0, 0] # Percentage (100K had <5% which rounds to 0)
ecog_missing = [3, 0, 0, 0] # Remainder

Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

Create stacked bars
x_pos = np.arange(len(trials))
width = 0.6

p1 = ax.bar(x_pos, ecog_0_1, width, label='ECOG 0-1', color='#2ecc71', alpha=0.8)
p2 = ax.bar(x_pos, ecog_2, width, bottom=ecog_0_1, label='ECOG 2', color='#f39c12', alpha=0.8)
p3 = ax.bar(x_pos, ecog_missing, width, bottom=np.array(ecog_0_1)+np.array(ecog_2),
 label='Not Specified', color='#95a5a6', alpha=0.8)

Add percentage labels
for i, (bar1, bar2) in enumerate(zip(p1, p2)):
 # ECOG 0-1 label
 height1 = bar1.get_height()
 if height1 > 5:
 ax.text(bar1.get_x() + bar1.get_width()/2., height1/2,
 f'{ecog_0_1[i]}%', ha='center', va='center', fontweight='bold', fontsize=12)

ECOG 2 label
```

```

height2 = bar2.get_height()
if height2 > 0:
 ax.text(bar2.get_x() + bar2.get_width()/2., ecog_0_1[i] + height2/2,
 f'{ecog_2[i]}%', ha='center', va='center', fontweight='bold', fontsize=11)

Add validation failure annotation
ax.annotate('VALIDATION\nFAILURE', xy=(0, 97), xytext=(-0.5, 85),
 arrowprops=dict(arrowstyle='->', color='red', lw=2),
 fontsize=10, color='red', weight='bold', ha='center')

Customize plot
ax.set_ylabel('Percentage of Patients (%)', fontsize=12, fontweight='bold')
ax.set_title('ECOG Performance Status Distribution Across Trials\nHighlighting Virtual Trial Patient Selection Bias',
 fontsize=14, fontweight='bold', pad=20)
ax.set_xticks(x_pos)
ax.set_xticklabels(trials)
ax.legend(loc='upper right')
ax.set_ylim(0, 105)
ax.grid(axis='y', alpha=0.3)

Add note
ax.text(0.5, -0.15, 'Note: 100K Virtual Trial underrepresented ECOG 2 patients compared to real-world data',
 transform=ax.transAxes, ha='center', fontsize=10, style='italic')

plt.tight_layout()
plt.savefig('05_stacked_bar_ecog_distribution.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()

```

## ## 6. Heatmap - Key Metrics Comparison

```

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

# Data matrix
treatments = ['100K Triplet', '100K Doublet', 'FOLFIRINOX', 'Gem+nab-P', 'nal-IRI+5FU']
metrics = ['OS Benefit\n(months)', 'HR vs Control', 'Grade ≥3 AE\nIncrease (%)', 'ETS Score']

# Create data matrix

```

```

data = np.array([
    [2.6, 0.69, 17.5, -0.69], # 100K Triplet
    [1.9, 0.76, np.nan, np.nan], # 100K Doublet
    [4.3, 0.57, 25.0, 0.36], # FOLFIRINOX
    [1.8, 0.72, 13.6, 0.00], # Gem+nab-P
    [1.9, 0.67, 30.0, -0.30] # nal-IRI+5FU (estimated)
])

# Create figure
fig, ax = plt.subplots(figsize=(10, 8), facecolor='white')

# Create custom colormap
cmap = plt.cm.RdYlGn_r # Reversed so lower HR is green

# Plot heatmap
sns.heatmap(data.T, annot=True, fmt='.2f', cmap=cmap, center=0,
            xticklabels=treatments, yticklabels=metrics,
            cbar_kws={'label': 'Standardized Value'},
            mask=np.isnan(data.T), square=True, linewidths=1,
            annot_kws={'fontsize': 11, 'fontweight': 'bold'})

# Customize plot
ax.set_title('Comparative Metrics Heatmap: Virtual vs Real-World PDAC Trials',
            fontsize=14, fontweight='bold', pad=20)
ax.set_xticklabels(ax.get_xticklabels(), rotation=45, ha='right')

# Add annotations for interpretation
ax.text(0.5, -0.15, 'Green = Favorable | Red = Unfavorable | Gray = Not Reported',
        transform=ax.transAxes, ha='center', fontsize=10, style='italic')

plt.tight_layout()
plt.savefig('06_heatmap_metrics_comparison.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()

```

7. Butterfly Plot - OS Benefit vs Toxicity Trade-off

```

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

```

```

Data
treatments = ['100K Triplet', '100K Doublet', 'FOLFIRINOX', 'Gem+nab-P', 'nal-IRI+5FU']
os_benefit = [2.6, 1.9, 4.3, 1.8, 1.9] # Months gained
toxicity_increase = [17.5, np.nan, 25.0, 13.6, 30.0] # % increase in Grade ≥3 AEs

Create figure
fig, ax = plt.subplots(figsize=(12, 8), facecolor='white')

Set up positions
y_pos = np.arange(len(treatments))

Plot toxicity on the left (negative values)
toxicity_valid = []
y_pos_valid = []
for i, (tox, treat) in enumerate(zip(toxicity_increase, treatments)):
 if not np.isnan(tox):
 toxicity_valid.append(-tox) # Negative for left side
 y_pos_valid.append(i)

bars_left = ax.barh(y_pos_valid, toxicity_valid, color='#e74c3c', alpha=0.8, label='Toxicity Increase')

Plot OS benefit on the right
bars_right = ax.barh(y_pos, os_benefit, color='#2ecc71', alpha=0.8, label='OS Benefit')

Add center line
ax.axvline(x=0, color='black', linewidth=2)

Add value labels
for bar in bars_left:
 width = bar.get_width()
 ax.text(width - 1, bar.get_y() + bar.get_height()/2,
 f'{-width:.1f}%', ha='right', va='center', fontweight='bold')

for bar in bars_right:
 width = bar.get_width()
 ax.text(width + 0.1, bar.get_y() + bar.get_height()/2,
 f'{width:.1f} mo', ha='left', va='center', fontweight='bold')

Customize plot
ax.set_yticks(y_pos)
ax.set_yticklabels(treatments)

```

```

ax.set_xlabel('← Toxicity Increase (%) | OS Benefit (months) →', fontsize=12, fontweight='bold')
ax.set_title('Butterfly Plot: Treatment Benefit vs Toxicity Trade-off\n100K Virtual Trial vs Real-World Standards',
 fontsize=14, fontweight='bold', pad=20)
ax.set_xlim(-35, 5)
ax.grid(axis='x', alpha=0.3)

Add annotations
ax.text(-17.5, -0.7, 'Higher Toxicity', ha='center', fontsize=10, color='#e74c3c', weight='bold')
ax.text(2.5, -0.7, 'Greater Benefit', ha='center', fontsize=10, color='#2ecc71', weight='bold')

Add note for missing data
ax.text(0.5, 0.02, '*100K Doublet: Toxicity data not reported',
 transform=ax.transAxes, fontsize=9, style='italic')

plt.tight_layout()
plt.savefig('07_butterfly_plot_benefit_toxicity.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()
...

```

## ## 8. Radar Chart - Multi-dimensional Comparison

```

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

# Data
categories = ['OS Benefit\n(normalized)', 'Low Toxicity\n(inverse)', 'Patient\nFitness',
            'Biomarker\nSpecificity', 'Clinical\nFeasibility']

# Scores (0-100 scale)
triplet_scores = [60, 20, 95, 90, 40] # High biomarker specificity, high toxicity
doublet_scores = [45, 70, 95, 90, 80] # Moderate OS, lower toxicity assumed
folfirinox_scores = [100, 40, 100, 20, 60] # Highest OS, high toxicity, no biomarker

# Number of variables
num_vars = len(categories)

# Compute angle for each axis
angles = np.linspace(0, 2 * np.pi, num_vars, endpoint=False).tolist()

triplet_scores += triplet_scores[:1]
doublet_scores += doublet_scores[:1]

```



```

folfirinox_scores += folfirinox_scores[:1]
angles += angles[:1]

# Create figure
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw=dict(projection='polar'), facecolor='white')

# Plot data
ax.plot(angles, triplet_scores, 'o-', linewidth=2, label='100K Triplet', color='#3498db')
ax.fill(angles, triplet_scores, alpha=0.25, color='#3498db')

ax.plot(angles, doublet_scores, 'o-', linewidth=2, label='100K Doublet', color='#9b59b6')
ax.fill(angles, doublet_scores, alpha=0.25, color='#9b59b6')

ax.plot(angles, folfirinox_scores, 'o-', linewidth=2, label='FOLFIRINOX', color='#e74c3c')
ax.fill(angles, folfirinox_scores, alpha=0.25, color='#e74c3c')

# Fix axis to go in the right order and start at 12 o'clock
ax.set_theta_offset(np.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=11, weight='bold')

# Set y-axis limits and labels
ax.set_ylim(0, 100)
ax.set_yticks([20, 40, 60, 80, 100])
ax.set_yticklabels(['20', '40', '60', '80', '100'], size=8)
ax.grid(True)

# Add legend and title
ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1.1))
plt.title('Multi-dimensional Comparison of PDAC Treatment Approaches',
          size=14, weight='bold', pad=30)

# Add explanatory note
fig.text(0.5, 0.02, 'Higher scores indicate better performance in each dimension',
         ha='center', fontsize=10, style='italic')

plt.tight_layout()
plt.savefig('08_radar_chart_multidimensional.png', dpi=300, bbox_inches='tight', facecolor='white')

```

```
plt.show()
'''
```

9. Sankey Diagram - KRAS G12C Patient Flow

```
'''python
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from matplotlib.patches import FancyBboxPatch, ConnectionPatch
import numpy as np

fig, ax = plt.subplots(figsize=(14, 10), facecolor='white')

# Define positions
col1_x, col2_x, col3_x = 0.1, 0.5, 0.9
y_positions = np.linspace(0.8, 0.2, 5)

# Column 1: Patient populations
populations = ['All PDAC\nPatients\n(100%)', 'KRAS Mutant\n(91%)', 'KRAS WT\n(9%)',
               'KRAS G12C\n(5%)', 'Other KRAS\n(86%)']
pop_sizes = [100, 91, 9, 5, 86]

# Column 2: Treatment arms
treatments = ['100K Triplet\n(Dara+Mita+nal-IRI)', '100K Doublet\n(Dara+Mita)',
              'Control\n(nal-IRI+5FU)']

# Column 3: Outcomes
outcomes = ['Enhanced Benefit\n(HR ~0.5)', 'Moderate Benefit\n(HR ~0.7-0.8)',
            'Minimal Benefit\n(HR ~0.9-1.0)']

# Draw boxes
# Population boxes
for i, (pop, size) in enumerate(zip(populations[:3], pop_sizes[:3])):
    height = size/100 * 0.6
    y = 0.2 + (i * 0.25)
    color = '#3498db' if 'KRAS' in pop and 'WT' not in pop else '#95a5a6'
    box = FancyBboxPatch((col1_x-0.08, y), 0.16, height,
                          boxstyle="round,pad=0.01",
                          facecolor=color, alpha=0.7, edgecolor='black')
    ax.add_patch(box)
    ax.text(col1_x, y+height/2, pop, ha='center', va='center', fontweight='bold', fontsize=10)
```



```

        shrinkA=5, shrinkB=5, mutation_scale=20,
        fc="green", alpha=0.6, linewidth=10)
ax.add_artist(con1)

# Other connections (thinner lines for moderate/minimal benefit)
con2 = ConnectionPatch((col2_x+0.08, 0.575), (col3_x-0.08, 0.575),
        "data", "data", arrowstyle="->",
        shrinkA=5, shrinkB=5, mutation_scale=20,
        fc="orange", alpha=0.5, linewidth=6)
ax.add_artist(con2)

# Title and labels
ax.text(col1_x, 0.95, 'Patient\nCharacteristics', ha='center', fontsize=12, fontweight='bold')
ax.text(col2_x, 0.95, 'Treatment\nAssignment', ha='center', fontsize=12, fontweight='bold')
ax.text(col3_x, 0.95, 'Treatment\nOutcome', ha='center', fontsize=12, fontweight='bold')

ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
ax.axis('off')

plt.title('Patient Flow: KRAS G12C Subgroup Drives Enhanced Benefit from Daraxonrasib-Containing Regimens',
        fontsize=14, fontweight='bold', pad=20)

# Add key finding
ax.text(0.5, 0.05, 'Key Finding: KRAS G12C patients (Archetype-05) showed enhanced response to Daraxonrasib
combinations',
        ha='center', fontsize=11, style='italic', transform=ax.transAxes)

plt.tight_layout()
plt.savefig('09_sankey_kras_patient_flow.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()
...

```

10. Timeline Visualization - Evolution of PDAC Treatment

```

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

fig, ax = plt.subplots(figsize=(14, 10), facecolor='white')

```

```

Timeline data
years = [2010, 2011, 2013, 2015, 2016, 2020, 2024, 2025]
events = [
 'Gemcitabine Era',
 'FOLFIRINOX\n(OS: 11.1 mo)',
 'MPACT: Gem+nab-P\n(OS: 8.5 mo)',
 'NAPOLI-1: nal-IRI\n(2nd line, OS: 6.2 mo)',
 'Continued refinement',
 'Precision medicine\napproaches emerge',
 'KRAS G12C inhibitors\nenter trials',
 '100K Virtual Trial\n(Triplet OS: 8.7 mo)'
]

os_values = [6.8, 11.1, 8.5, 6.2, None, None, None, 8.7]
trial_types = ['standard', 'breakthrough', 'improvement', 'second-line',
 'era', 'era', 'era', 'virtual']

Color mapping
color_map = {
 'standard': '#95a5a6',
 'breakthrough': '#27ae60',
 'improvement': '#3498db',
 'second-line': '#f39c12',
 'era': '#ecf0f1',
 'virtual': '#9b59b6'
}

Draw timeline
timeline_y = 0.5
ax.plot([2009, 2026], [timeline_y, timeline_y], 'k-', linewidth=3)

Add events
for i, (year, event, os, ttype) in enumerate(zip(years, events, os_values, trial_types)):
 # Draw marker
 marker_size = 15 if ttype != 'era' else 8
 ax.scatter(year, timeline_y, s=marker_size**2, c=color_map[ttype],
 edgecolor='black', linewidth=2, zorder=5)

 # Add event label
 y_offset = 0.15 if i % 2 == 0 else -0.15

```

```

ax.text(year, timeline_y + y_offset, event, ha='center', va='center',
 fontweight='bold', fontsize=10,
 bbox=dict(boxstyle="round,pad=0.3", facecolor='white', alpha=0.8))

Add OS value if available
if os is not None:
 os_y = timeline_y + y_offset + (0.08 if y_offset > 0 else -0.08)
 ax.text(year, os_y, f'OS: {os} mo', ha='center', va='center',
 fontsize=9, style='italic', color='darkred')

Connect to timeline
ax.plot([year, year], [timeline_y, timeline_y + y_offset*0.8],
 'k--', alpha=0.3, linewidth=1)

Add era labels
ax.text(2010, 0.85, 'Single-Agent Era', fontsize=12, fontweight='bold', ha='center')
ax.text(2013, 0.85, 'Combination Era', fontsize=12, fontweight='bold', ha='center')
ax.text(2022, 0.85, 'Precision/Virtual Era', fontsize=12, fontweight='bold', ha='center')

Add progression arrow
ax.annotate("", xy=(2025, 0.15), xytext=(2010, 0.15),
 arrowprops=dict(arrowstyle='->', color='green', lw=3))
ax.text(2017.5, 0.1, 'Increasing Treatment Complexity & Personalization',
 ha='center', fontsize=11, color='green', style='italic')

Highlight virtual trial
virtual_box = patches.Rectangle((2024.5, 0.3), 1, 0.4,
 linewidth=2, edgecolor='purple',
 facecolor='purple', alpha=0.1)
ax.add_patch(virtual_box)

Customize plot
ax.set_xlim(2009, 2026)
ax.set_ylim(0, 1)
ax.set_xlabel('Year', fontsize=12, fontweight='bold')
ax.set_title('Evolution of PDAC Treatment Standards (2010-2025)\nMedian Overall Survival Progression',
 fontsize=14, fontweight='bold', pad=20)

Remove y-axis
ax.get_yaxis().set_visible(False)
ax.spines['left'].set_visible(False)

```

```

ax.spines['right'].set_visible(False)
ax.spines['top'].set_visible(False)

Add legend
legend_elements = [
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#27ae60',
 markersize=10, label='Breakthrough Trial'),
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#3498db',
 markersize=10, label='Incremental Advance'),
 plt.Line2D([0], [0], marker='o', color='w', markerfacecolor='#9b59b6',
 markersize=10, label='Virtual Trial')
]
ax.legend(handles=legend_elements, loc='upper left')

Add note
ax.text(0.5, -0.1, 'Note: 100K Virtual Trial demonstrates comparable efficacy to current standards, suggesting validity of
simulation approach',
 transform=ax.transAxes, ha='center', fontsize=10, style='italic')

plt.tight_layout()
plt.savefig('10_timeline_pdac_evolution.png', dpi=300, bbox_inches='tight', facecolor='white')
plt.show()
...

```

These visualizations effectively communicate the key findings from the meta-analysis:

1. **Forest Plot** shows the virtual trial HRs align well with real-world trials
2. **Efficiency Frontier** reveals the virtual triplet's poor toxicity-efficacy balance
3. **Grouped Bar Chart** demonstrates comparable OS outcomes between virtual and real trials
4. **Waterfall Plot** ranks treatments by their efficacy-toxicity scores
5. **ECOG Distribution** highlights the critical patient selection bias in the virtual trial
6. **Heatmap** provides an at-a-glance comparison of all key metrics
7. **Butterfly Plot** visually contrasts benefit gains against toxicity costs
8. **Radar Chart** shows multi-dimensional strengths/weaknesses of each approach
9. **Sankey Diagram** illustrates how KRAS G12C patients drive the virtual triplet's benefit
10. **Timeline** contextualizes the virtual trial within PDAC treatment evolution

Each visualization has a clean white background and uses color strategically to highlight key insights from the meta-analysis.

## **Research Summary Generation 58b2: 326 Words**

The user is asking me to create 10 Python scripts to generate visualizations based on the meta-analysis data from the document. I need to extract the relevant data from the tables and text to create meaningful visualizations. Let me analyze what data is available:

From Table 1 (In-Silico PDAC Trials):

- 100K Triplicate Control (Arm E): OS 6.1 mo, Grade  $\geq 3$  AE 76.5%
- 100K Triplicate Triplet (Arm A): OS 8.7 mo, HR  $\sim 0.69$ , Grade  $\geq 3$  AE 94.0%
- 100K Doublet (Arm D): OS  $\sim 8.0$  mo (calculated), HR  $\sim 0.76$
- Digital Twin Study: OS  $\sim 6.7$  mo (control) /  $\sim 8.5$  mo (experimental), HR  $\sim 0.72$
- AI Simulation: N/R for OS/PFS

From Table 2 (Virtual vs Real-World):

- MPACT: Gem+nab-P OS 8.5 mo, HR 0.72, Grade  $\geq 3$  AE 84%
- MPACT Control: Gem OS 6.7 mo, Grade  $\geq 3$  AE  $\sim 70.4\%$
- NAPOLI-1: nal-IRI + 5-FU OS 6.2 mo, HR 0.67, Grade  $\geq 3$  AE  $\sim 76\%$
- NAPOLI-1 Control: 5-FU OS 4.2 mo, Grade  $\geq 3$  AE  $\sim 46\%$
- FOLFIRINOX: OS 11.1 mo, HR 0.57, Grade  $\geq 3$  AE  $\sim 75\%$
- FOLFIRINOX Control: Gem OS 6.8 mo, Grade  $\geq 3$  AE  $\sim 50\%$

I'll now examine the ETS Scores and ECOG Performance Status data. The ETS Scores reveal interesting variations across different treatment approaches. The 100K Triplet shows a negative ETS of  $-0.69$ , while FOLFIRINOX demonstrates a positive ETS of  $+0.36$ . Most treatments cluster around neutral to slightly negative ETS values. For ECOG Performance Status, the 100K Virtual study indicates over 95% of patients were in the 0-1 range, suggesting a relatively healthy patient population for the trials.

The sample sizes provide important context for these findings. The 100K Virtual study has a substantial 20,000 patients per arm, which contrasts with smaller clinical trials like NAPOLI-1 with 417 total patients and FOLFIRINOX with 342 total participants. The MPACT trial included 861 total patients, offering a mid-range sample size for comparative analysis.