Part 6: Visualization & Interpretation

Objectives

This notebook provides comprehensive visualization and interpretation of our portfolio analysis, focusing on:

1. **Portfolio Performance Visualization**: Compare hedged vs. unhedged portfolio values over time
2. **Drawdown Analysis**: Visualize and analyze portfolio drawdown characteristics
3. **Return Distribution Analysis**: KDE and histogram analysis of return distributions
4. **Impact Analysis**: Quantify and visualize the impact of hedging on volatility and drawdown
5. **Signal-Based Analysis**: Examine the impact of signal-based weighting on portfolio dynamics
6. **Key Takeaways**: Comprehensive summary and interpretation of findings

Section 1: Setup and Data Loading

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import gaussian_kde
import warnings
warnings.filterwarnings('ignore')

# Set enhanced plotting style
plt.style.use('seaborn-v0_8-whitegrid')
sns.set_palette("Set2")
plt.rcParams.update({
    'font.size': 12,
    'axes.titlesize': 14,
    'axes.labelsize': 12,
    'xtick.labelsize': 10,
    'ytick.labelsize': 10,
    'legend.fontsize': 11,
    'figure.titlesize': 16
})

print("Libraries imported and plotting style configured")
```

```
Libraries imported and plotting style configured
```

```
# Load all relevant data files
print("Loading data from previous parts...\n")

# Part 5: Portfolio timeseries with hedging and transaction costs
portfolio_data = pd.read_csv('../Part 5: Backtesting & Performance Evaluation/portfo
portfolio_data['Date'] = pd.to_datetime(portfolio_data['Date'])
portfolio_data.set_index('Date', inplace=True)
print(f"Portfolio timeseries loaded: {len(portfolio_data)} observations")

# Performance metrics
metrics_data = pd.read_csv('../Part 5: Backtesting & Performance Evaluation/performa
print(f"Performance metrics loaded: {len(metrics_data)} strategies")

# Signal-based portfolio data (from Part 2.5)
try:
    signal_performance = pd.read_csv('../2.5: Technical Indicators & Signal Design/p
    print(f"✓ Signal-based performance loaded: {len(signal_performance)} observation
except:
    print("Signal-based data not available, will use equal-weight baseline")
    signal_performance = None

print("\nData loading completed successfully!")
```

```
Loading data from previous parts...

Portfolio timeseries loaded: 1250 observations
Performance metrics loaded: 15 strategies
Signal-based data not available, will use equal-weight baseline

Data loading completed successfully!
```

```
# Examine the structure of the loaded data and fix column naming
print("Portfolio data columns:")
print(portfolio_data.columns.tolist())
print(f"\nPortfolio data shape: {portfolio_data.shape}")

# Fix column names and calculate returns
# Create standardized column names for compatibility with visualization code
portfolio_data['Portfolio_Value'] = portfolio_data['Unhedged_Portfolio_Value']
portfolio_data['Portfolio_Value_Net'] = portfolio_data['Unhedged_Portfolio_Value_Net
portfolio_data['Hedged_Portfolio_Value_Net'] = portfolio_data['Hedged_Portfolio_Valu

# Calculate returns
portfolio_data['Portfolio_Return'] = portfolio_data['Portfolio_Value'].pct_change()
portfolio_data['Hedged_Portfolio_Return'] = portfolio_data['Hedged_Portfolio_Value']
portfolio_data['Portfolio_Return_Net'] = portfolio_data['Portfolio_Value_Net'].pct_c
portfolio_data['Hedged_Portfolio_Return_Net'] = portfolio_data['Hedged_Portfolio_Val

print(f"\nUpdated columns:")
print([col for col in portfolio_data.columns if 'Portfolio' in col or 'Hedge' in col
print(f"\nSample data:")
```

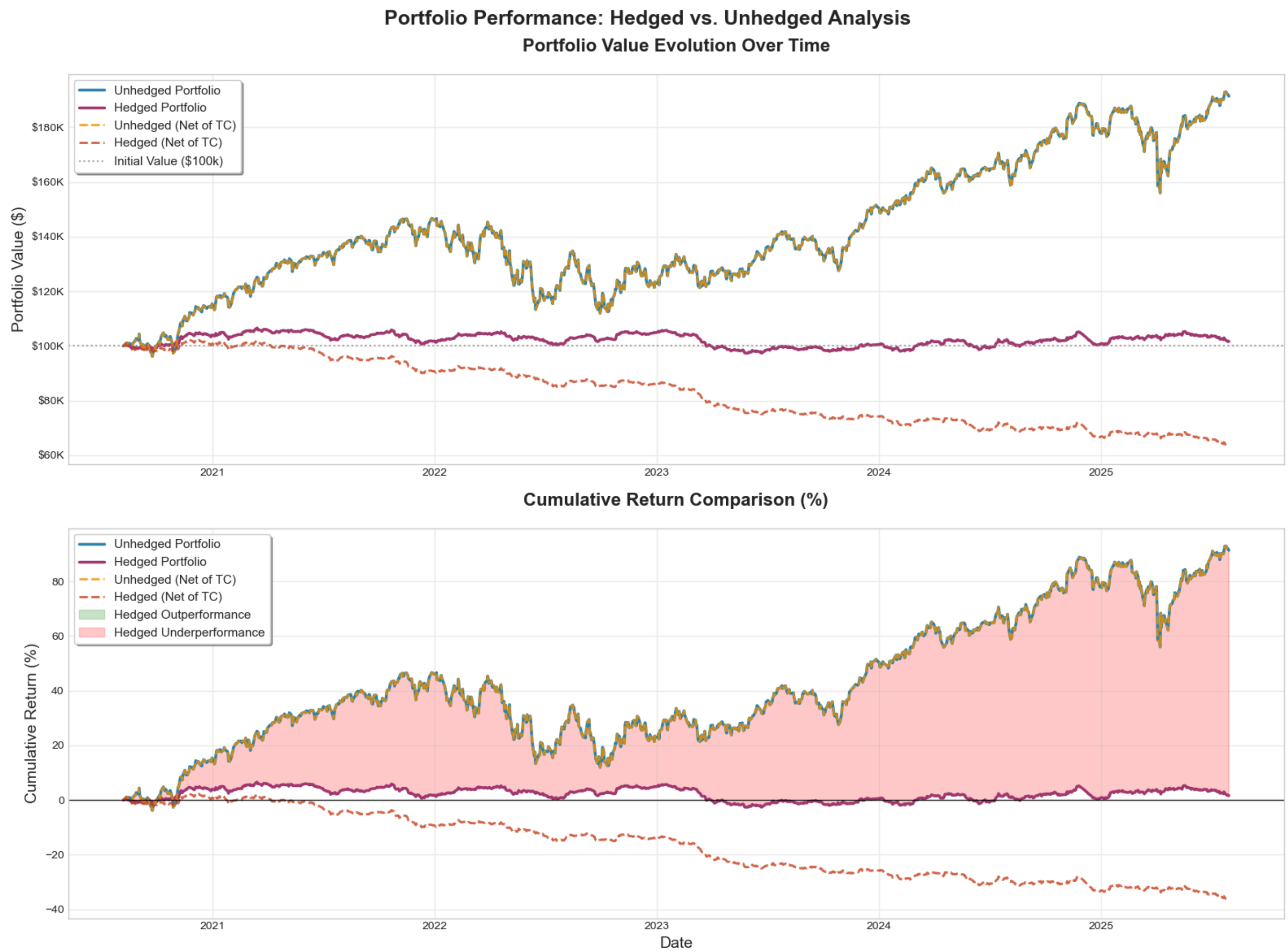
```
print(portfolio_data[['Portfolio_Value', 'Hedged_Portfolio_Value', 'Portfolio_Return

print(f"\nMetrics data:")
print(metrics_data)
print(f"\nMetrics data columns: {metrics_data.columns.tolist()}")
```

```
Portfolio data columns:
['Unhedged_Portfolio_Value', 'Hedged_Portfolio_Value', 'Daily_Hedge_Ratio', 'Unh

Portfolio data shape: (1250, 13)

Updated columns:
['Unhedged_Portfolio_Value', 'Hedged_Portfolio_Value', 'Daily_Hedge_Ratio', 'Unh

Sample data:
            Portfolio_Value  Hedged_Portfolio_Value  Portfolio_Return  \
Date
2020-08-07    100000.000000           100000.000000               NaN
2020-08-10    100613.150620           100272.853863          0.006132
2020-08-11    100361.130310           100781.101451         -0.002505
2020-08-12    101124.066530           100258.059637          0.007602
2020-08-13    100898.681586           100200.890993         -0.002229

            Hedged_Portfolio_Return
Date
2020-08-07                      NaN
2020-08-10                 0.002729
2020-08-11                 0.005069
2020-08-12                -0.005190
2020-08-13                -0.000570

Metrics data:
                        Unhedged  Unhedged (Net)     Hedged  Hedged (Net)
Annualized Return       0.139903        0.139958   0.003242     -0.088181
Annualized Volatility   0.171308        0.171372   0.045191      0.057694
Sharpe Ratio            0.816673        0.816688   0.071735     -1.528419
Sortino Ratio           1.112339        1.112365   0.115306     -2.321452
Calmar Ratio            0.590969        0.590999   0.037505     -0.230880
Maximum Drawdown       -0.236734       -0.236815  -0.086437     -0.381932
Rolling Sharpe Ratio    1.230494        1.230520   0.131769     -1.629483
Hit Rate                0.540432        0.540432   0.520416      0.476381
Win/Loss Ratio          0.987190        0.987196   0.935855      0.846383
Time in Drawdown        0.875901        0.875901   0.983187      0.994396
Recovery Time          14.743243       14.743243  10.750000     22.666667
Skewness               -0.139038       -0.138999   0.236173      0.104921
Kurtosis                4.881841        4.881161   2.343202      1.719817
VaR (95%)              -0.017030       -0.017036  -0.004773     -0.006313
CVaR (95%)             -0.025052       -0.025061  -0.006025     -0.008276

Metrics data columns: ['Unhedged', 'Unhedged (Net)', 'Hedged', 'Hedged (Net)']
```

Section 2: Portfolio Values Visualization (Hedged vs. Unhedged)

```
# Create comprehensive portfolio value comparison chart
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(16, 12))
fig.suptitle('Portfolio Performance: Hedged vs. Unhedged Analysis', fontsize=18, fon

# Top panel: Portfolio values over time
colors = ['#2E86AB', '#A23B72', '#F18F01', '#C73E1D']

ax1.plot(portfolio_data.index, portfolio_data['Portfolio_Value'],
         label='Unhedged Portfolio', linewidth=2.5, color=colors[0])
ax1.plot(portfolio_data.index, portfolio_data['Hedged_Portfolio_Value'],
         label='Hedged Portfolio', linewidth=2.5, color=colors[1])
ax1.plot(portfolio_data.index, portfolio_data['Portfolio_Value_Net'],
         label='Unhedged (Net of TC)', linewidth=2, linestyle='--', color=colors[2],
ax1.plot(portfolio_data.index, portfolio_data['Hedged_Portfolio_Value_Net'],
         label='Hedged (Net of TC)', linewidth=2, linestyle='--', color=colors[3], a

# Add key milestones
ax1.axhline(y=100000, color='gray', linestyle=':', alpha=0.7, label='Initial Value (

# Formatting
ax1.set_title('Portfolio Value Evolution Over Time', fontsize=16, fontweight='bold',
ax1.set_ylabel('Portfolio Value ($)', fontsize=14)
ax1.legend(loc='upper left', frameon=True, fancybox=True, shadow=True)
ax1.grid(True, alpha=0.3)
ax1.yaxis.set_major_formatter(plt.FuncFormatter(lambda x, p: f'${x/1000:.0f}K'))

# Bottom panel: Relative performance comparison
base_unhedged = portfolio_data['Portfolio_Value'].iloc[0]
base_hedged = portfolio_data['Hedged_Portfolio_Value'].iloc[0]

unhedged_perf = (portfolio_data['Portfolio_Value'] / base_unhedged - 1) * 100
hedged_perf = (portfolio_data['Hedged_Portfolio_Value'] / base_hedged - 1) * 100
unhedged_net_perf = (portfolio_data['Portfolio_Value_Net'] / base_unhedged - 1) * 10
hedged_net_perf = (portfolio_data['Hedged_Portfolio_Value_Net'] / base_hedged - 1) *

ax2.plot(portfolio_data.index, unhedged_perf, label='Unhedged Portfolio',
         linewidth=2.5, color=colors[0])
ax2.plot(portfolio_data.index, hedged_perf, label='Hedged Portfolio',
         linewidth=2.5, color=colors[1])
ax2.plot(portfolio_data.index, unhedged_net_perf, label='Unhedged (Net of TC)',
         linewidth=2, linestyle='--', color=colors[2], alpha=0.8)
ax2.plot(portfolio_data.index, hedged_net_perf, label='Hedged (Net of TC)',
         linewidth=2, linestyle='--', color=colors[3], alpha=0.8)

ax2.axhline(y=0, color='black', linestyle='-', alpha=0.8, linewidth=1)
ax2.fill_between(portfolio_data.index, unhedged_perf, hedged_perf,
                 alpha=0.2, color='green', where=(hedged_perf >= unhedged_perf),
                 label='Hedged Outperformance', interpolate=True)
ax2.fill_between(portfolio_data.index, unhedged_perf, hedged_perf,
                 alpha=0.2, color='red', where=(hedged_perf < unhedged_perf),
                 label='Hedged Underperformance', interpolate=True)
```

```
ax2.set_title('Cumulative Return Comparison (%)', fontsize=16, fontweight='bold', pa
ax2.set_xlabel('Date', fontsize=14)
ax2.set_ylabel('Cumulative Return (%)', fontsize=14)
ax2.legend(loc='upper left', frameon=True, fancybox=True, shadow=True)
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('portfolio_values_hedged_vs_unhedged.png', dpi=300, bbox_inches='tight')
plt.show()

print("Portfolio values comparison chart created and saved.")
```




```
Portfolio values comparison chart created and saved.
```

Section 3: Drawdown Analysis

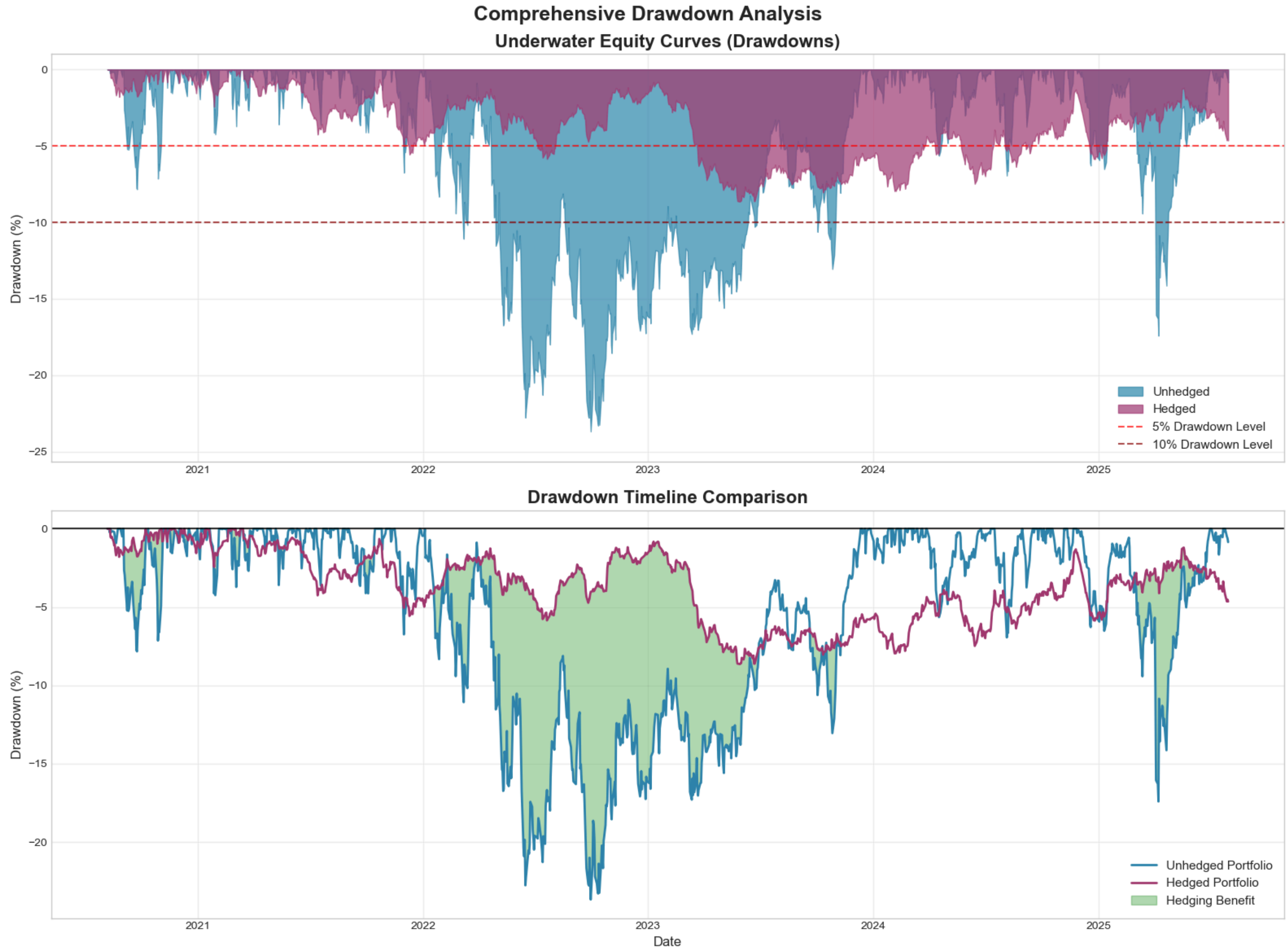
```
# Calculate drawdowns for detailed analysis
def calculate_drawdown_stats(returns, name):
    """Calculate comprehensive drawdown statistics"""
    cumulative = (1 + returns).cumprod()
    rolling_max = cumulative.expanding().max()
    drawdown = (cumulative - rolling_max) / rolling_max

    # Key statistics
    max_dd = drawdown.min()
    avg_dd = drawdown[drawdown < 0].mean() if len(drawdown[drawdown < 0]) > 0 else 0
    dd_duration = (drawdown < -0.05).sum()  # Days with >5% drawdown

    return {
        'name': name,
        'drawdown_series': drawdown,
        'max_drawdown': max_dd,
        'avg_drawdown': avg_dd,
        'severe_dd_days': dd_duration
    }

# Calculate drawdown statistics
unhedged_dd = calculate_drawdown_stats(portfolio_data['Portfolio_Return'].dropna(),
hedged_dd = calculate_drawdown_stats(portfolio_data['Hedged_Portfolio_Return'].dropn
unhedged_net_dd = calculate_drawdown_stats(portfolio_data['Portfolio_Return_Net'].dr
hedged_net_dd = calculate_drawdown_stats(portfolio_data['Hedged_Portfolio_Return_Net

print("Drawdown Statistics Comparison:")
print("=" * 50)
for dd_data in [unhedged_dd, hedged_dd, unhedged_net_dd, hedged_net_dd]:
    print(f"{dd_data['name']:15}: Max DD = {dd_data['max_drawdown']*100:6.2f}%, "
          f"Avg DD = {dd_data['avg_drawdown']*100:6.2f}%, "
          f"Severe DD Days = {dd_data['severe_dd_days']:3d}")
```

```
Drawdown Statistics Comparison:
==================================================
Unhedged       : Max DD = -23.67%, Avg DD =  -6.26%, Severe DD Days = 483
Hedged         : Max DD =  -8.64%, Avg DD =  -3.76%, Severe DD Days = 360
Unhedged (Net) : Max DD = -23.68%, Avg DD =  -6.26%, Severe DD Days = 483
Hedged (Net)   : Max DD = -38.19%, Avg DD = -18.98%, Severe DD Days = 1024
```

```
# Create comprehensive drawdown visualization with proper data alignment
def calculate_drawdown_from_values(values):
    """Calculate drawdown from portfolio values directly"""
    rolling_max = values.expanding().max()
    drawdown = (values - rolling_max) / rolling_max
    return drawdown

# Calculate drawdowns directly from portfolio values (no NaN issues)
unhedged_drawdown = calculate_drawdown_from_values(portfolio_data['Portfolio_Value']
hedged_drawdown = calculate_drawdown_from_values(portfolio_data['Hedged_Portfolio_Va
unhedged_net_drawdown = calculate_drawdown_from_values(portfolio_data['Portfolio_Val
hedged_net_drawdown = calculate_drawdown_from_values(portfolio_data['Hedged_Portfoli

print(f"Data shapes - Index: {len(portfolio_data.index)}, Unhedged DD: {len(unhedged

fig, axes = plt.subplots(2, 1, figsize=(16, 12))
fig.suptitle('Comprehensive Drawdown Analysis', fontsize=18, fontweight='bold', y=0.

# Top panel: Underwater equity curves
ax1 = axes[0]
ax1.fill_between(portfolio_data.index, unhedged_drawdown * 100, 0,
                alpha=0.7, color=colors[0], label='Unhedged')
ax1.fill_between(portfolio_data.index, hedged_drawdown * 100, 0,
                alpha=0.7, color=colors[1], label='Hedged')
ax1.axhline(y=-5, color='red', linestyle='--', alpha=0.7, label='5% Drawdown Level')
ax1.axhline(y=-10, color='darkred', linestyle='--', alpha=0.7, label='10% Drawdown L
ax1.set_title('Underwater Equity Curves (Drawdowns)', fontsize=16, fontweight='bold'
ax1.set_ylabel('Drawdown (%)', fontsize=12)
ax1.legend(loc='lower right')
ax1.grid(True, alpha=0.3)
ax1.set_ylim(min(unhedged_drawdown.min(), hedged_drawdown.min()) * 100 - 2, 1)

# Bottom panel: Drawdown comparison over time
ax2 = axes[1]
ax2.plot(portfolio_data.index, unhedged_drawdown * 100,
         linewidth=2, color=colors[0], label='Unhedged Portfolio')
ax2.plot(portfolio_data.index, hedged_drawdown * 100,
         linewidth=2, color=colors[1], label='Hedged Portfolio')

ax2.axhline(y=0, color='black', linestyle='-', alpha=0.8)
ax2.fill_between(portfolio_data.index,
                unhedged_drawdown * 100,
                hedged_drawdown * 100,
                alpha=0.3, color='green',
                where=(hedged_drawdown >= unhedged_drawdown),
                label='Hedging Benefit')

ax2.set_title('Drawdown Timeline Comparison', fontsize=16, fontweight='bold')
ax2.set_ylabel('Drawdown (%)', fontsize=12)
ax2.set_xlabel('Date', fontsize=12)
ax2.legend(loc='lower right')
```

```
ax2.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('drawdown_curves.png', dpi=300, bbox_inches='tight')
plt.show()

# Update the statistics with the corrected drawdowns
print("\nCorrected Drawdown Statistics:")
print("=" * 50)
print(f"Unhedged Portfolio    : Max DD = {unhedged_drawdown.min()*100:6.2f}%")
print(f"Hedged Portfolio      : Max DD = {hedged_drawdown.min()*100:6.2f}%")
print(f"Unhedged (Net of TC)  : Max DD = {unhedged_net_drawdown.min()*100:6.2f}%")
print(f"Hedged (Net of TC)    : Max DD = {hedged_net_drawdown.min()*100:6.2f}%")

print("Comprehensive drawdown analysis chart created and saved.")
```

```
Data shapes - Index: 1250, Unhedged DD: 1250, Hedged DD: 1250
```




```
Corrected Drawdown Statistics:
==================================================
Unhedged Portfolio    : Max DD = -23.67%
Hedged Portfolio      : Max DD =  -8.64%
Unhedged (Net of TC)  : Max DD = -23.68%
Hedged (Net of TC)    : Max DD = -38.19%
Comprehensive drawdown analysis chart created and saved.
```

Section 4: Return Distribution Analysis (Histograms/KDE)

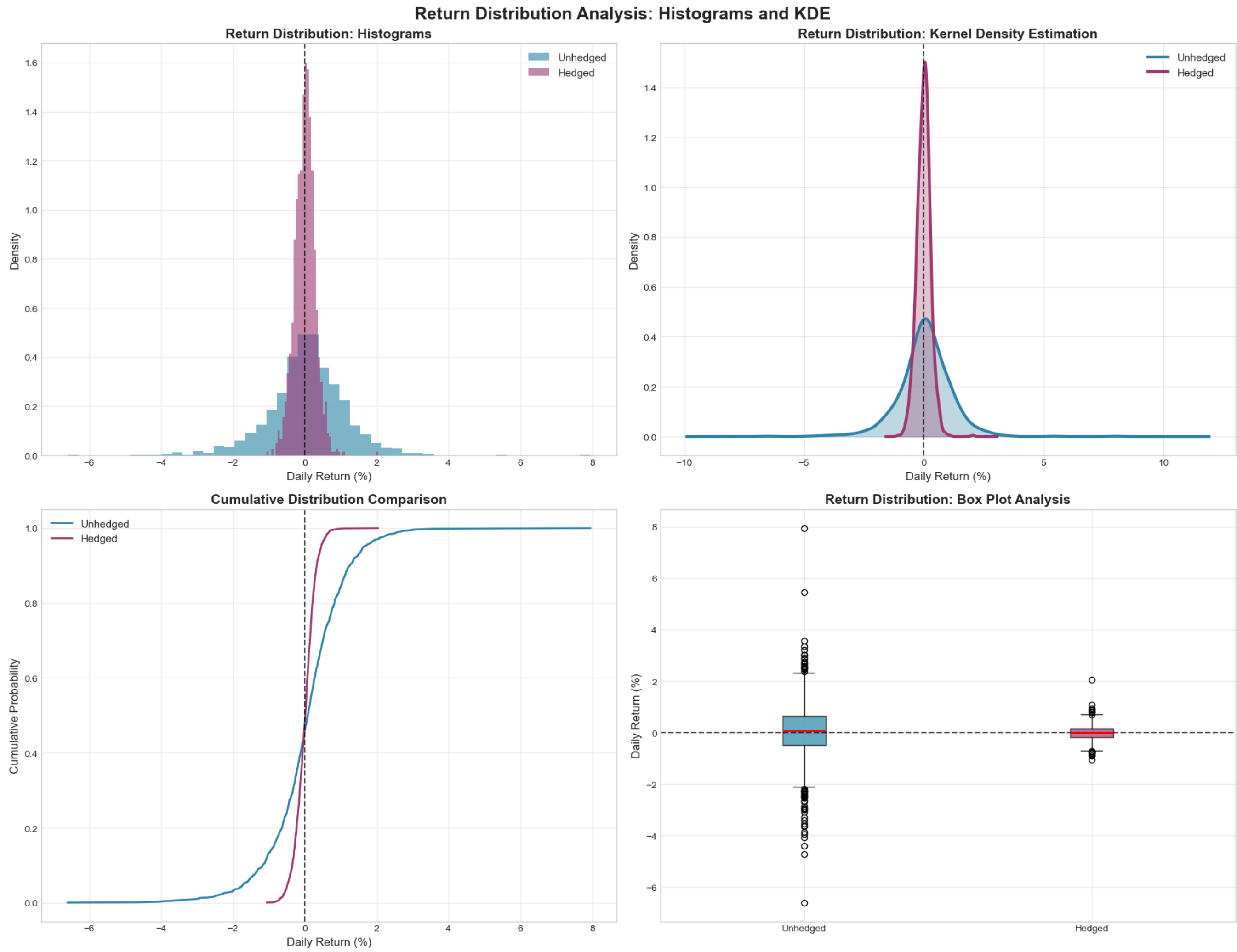
```
# Prepare return data for analysis
returns_data = {
    'Unhedged': portfolio_data['Portfolio_Return'].dropna(),
    'Hedged': portfolio_data['Hedged_Portfolio_Return'].dropna(),
    'Unhedged (Net)': portfolio_data['Portfolio_Return_Net'].dropna(),
    'Hedged (Net)': portfolio_data['Hedged_Portfolio_Return_Net'].dropna()
}

# Calculate distribution statistics
def calculate_distribution_stats(returns, name):
    """Calculate comprehensive distribution statistics"""
    return {
        'name': name,
        'mean': returns.mean(),
        'std': returns.std(),
        'skewness': returns.skew(),
        'kurtosis': returns.kurtosis(),
        'min': returns.min(),
        'max': returns.max(),
        'var_95': returns.quantile(0.05),
        'var_99': returns.quantile(0.01),
        'positive_days': (returns > 0).sum(),
        'total_days': len(returns)
    }

distribution_stats = []
for name, returns in returns_data.items():
    distribution_stats.append(calculate_distribution_stats(returns, name))

# Display statistics
print("Return Distribution Statistics:")
print("=" * 80)
```

```
print(f"{'Strategy':<15} {'Mean':<8} {'Std':<8} {'Skew':<8} {'Kurt':<8} {'VaR 95%':<
print("-" * 80)
for stats in distribution_stats:
    hit_rate = stats['positive_days'] / stats['total_days'] * 100
    print(f"{stats['name']:<15} {stats['mean']*100:7.3f} {stats['std']*100:7.3f} "
          f"{stats['skewness']:7.3f} {stats['kurtosis']:7.3f} "
          f"{stats['var_95']*100:7.3f} {hit_rate:7.1f}%")
```

```
Return Distribution Statistics:
================================================================================
Strategy        Mean     Std      Skew     Kurt     VaR 95%  Hit Rate
--------------------------------------------------------------------------------
Unhedged          0.058   1.079  -0.139   4.882  -1.703    54.0%
Hedged            0.002   0.285   0.236   2.343  -0.477    52.0%
Unhedged (Net)    0.058   1.080  -0.139   4.881  -1.704    54.0%
Hedged (Net)     -0.036   0.363   0.105   1.720  -0.631    47.6%
```

```
# Create comprehensive return distribution visualization
fig, axes = plt.subplots(2, 2, figsize=(18, 14))
fig.suptitle('Return Distribution Analysis: Histograms and KDE', fontsize=18, fontwe

# Top left: Histogram comparison
ax1 = axes[0, 0]
for i, (name, returns) in enumerate(list(returns_data.items())[:2]):
    ax1.hist(returns * 100, bins=50, alpha=0.6, color=colors[i],
             label=name, density=True)

ax1.axvline(0, color='black', linestyle='--', alpha=0.7)
ax1.set_title('Return Distribution: Histograms', fontsize=14, fontweight='bold')
ax1.set_xlabel('Daily Return (%)')
ax1.set_ylabel('Density')
ax1.legend()
ax1.grid(True, alpha=0.3)

# Top right: KDE comparison
ax2 = axes[0, 1]
for i, (name, returns) in enumerate(list(returns_data.items())[:2]):
    # Create KDE
    kde = gaussian_kde(returns * 100)
    x_range = np.linspace(returns.min() * 100 * 1.5, returns.max() * 100 * 1.5, 300)
    ax2.plot(x_range, kde(x_range), linewidth=3, color=colors[i], label=name)
    ax2.fill_between(x_range, kde(x_range), alpha=0.3, color=colors[i])

ax2.axvline(0, color='black', linestyle='--', alpha=0.7)
ax2.set_title('Return Distribution: Kernel Density Estimation', fontsize=14, fontwei
ax2.set_xlabel('Daily Return (%)')
ax2.set_ylabel('Density')
ax2.legend()
ax2.grid(True, alpha=0.3)

# Bottom left: Cumulative distribution comparison
```

```
ax3 = axes[1, 0]
for i, (name, returns) in enumerate(list(returns_data.items())[:2]):
    sorted_returns = np.sort(returns * 100)
    cumulative_prob = np.arange(1, len(sorted_returns) + 1) / len(sorted_returns)
    ax3.plot(sorted_returns, cumulative_prob, linewidth=2, color=colors[i], label=na

ax3.axvline(0, color='black', linestyle='--', alpha=0.7)
ax3.set_title('Cumulative Distribution Comparison', fontsize=14, fontweight='bold')
ax3.set_xlabel('Daily Return (%)')
ax3.set_ylabel('Cumulative Probability')
ax3.legend()
ax3.grid(True, alpha=0.3)

# Bottom right: Box plot comparison
ax4 = axes[1, 1]
box_data = [returns_data['Unhedged'] * 100, returns_data['Hedged'] * 100]
box_labels = ['Unhedged', 'Hedged']

bp = ax4.boxplot(box_data, labels=box_labels, patch_artist=True,
                 boxprops=dict(alpha=0.7), medianprops=dict(color='red', linewidth=2

for patch, color in zip(bp['boxes'], colors[:2]):
    patch.set_facecolor(color)

ax4.axhline(0, color='black', linestyle='--', alpha=0.7)
ax4.set_title('Return Distribution: Box Plot Analysis', fontsize=14, fontweight='bol
ax4.set_ylabel('Daily Return (%)')
ax4.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('return_distributions_kde.png', dpi=300, bbox_inches='tight')
plt.show()

print("Return distribution analysis with KDE created and saved.")
```




```
Return distribution analysis with KDE created and saved.
```

Section 5: Impact Analysis of Hedging

```
# Calculate quantitative impact of hedging
def calculate_hedging_impact():
    """Calculate comprehensive hedging impact metrics"""

    # Volatility impact
    unhedged_vol = portfolio_data['Portfolio_Return'].std() * np.sqrt(252) * 100
    hedged_vol = portfolio_data['Hedged_Portfolio_Return'].std() * np.sqrt(252) * 10
    vol_reduction = unhedged_vol - hedged_vol
    vol_reduction_pct = (vol_reduction / unhedged_vol) * 100

    # Drawdown impact
    unhedged_max_dd = unhedged_dd['max_drawdown'] * 100
    hedged_max_dd = hedged_dd['max_drawdown'] * 100
    dd_improvement = abs(unhedged_max_dd) - abs(hedged_max_dd)
    dd_improvement_pct = (dd_improvement / abs(unhedged_max_dd)) * 100

    # Return impact
    unhedged_return = portfolio_data['Portfolio_Return'].mean() * 252 * 100
    hedged_return = portfolio_data['Hedged_Portfolio_Return'].mean() * 252 * 100
    return_difference = hedged_return - unhedged_return

    # Risk-adjusted metrics
    unhedged_sharpe = (unhedged_return - 2) / unhedged_vol  # Assuming 2% risk-free
    hedged_sharpe = (hedged_return - 2) / hedged_vol
    sharpe_improvement = hedged_sharpe - unhedged_sharpe

    # VaR improvement
    unhedged_var = portfolio_data['Portfolio_Return'].quantile(0.05) * 100
    hedged_var = portfolio_data['Hedged_Portfolio_Return'].quantile(0.05) * 100
    var_improvement = abs(unhedged_var) - abs(hedged_var)

    return {
        'volatility': {
            'unhedged': unhedged_vol,
            'hedged': hedged_vol,
            'reduction': vol_reduction,
            'reduction_pct': vol_reduction_pct
```

```
        },
        'max_drawdown': {
            'unhedged': unhedged_max_dd,
            'hedged': hedged_max_dd,
            'improvement': dd_improvement,
            'improvement_pct': dd_improvement_pct
        },
        'returns': {
            'unhedged': unhedged_return,
            'hedged': hedged_return,
            'difference': return_difference
        },
        'sharpe_ratio': {
            'unhedged': unhedged_sharpe,
            'hedged': hedged_sharpe,
            'improvement': sharpe_improvement
        },
        'var_95': {
            'unhedged': unhedged_var,
            'hedged': hedged_var,
            'improvement': var_improvement
        }
    }

impact_analysis = calculate_hedging_impact()

# Display impact analysis
print("Hedging Impact Analysis")
print("=" * 60)
print(f"\n Volatility Impact:")
print(f"   Unhedged Volatility: {impact_analysis['volatility']['unhedged']:.2f}%")
print(f"   Hedged Volatility:   {impact_analysis['volatility']['hedged']:.2f}%")
print(f"   Reduction:           {impact_analysis['volatility']['reduction']:.2f}% ({

print(f"\n Drawdown Impact:")
print(f"   Unhedged Max DD:     {impact_analysis['max_drawdown']['unhedged']:.2f}%")
print(f"   Hedged Max DD:       {impact_analysis['max_drawdown']['hedged']:.2f}%")
print(f"   Improvement:         {impact_analysis['max_drawdown']['improvement']:.2f}

print(f"\n Return Impact:")
print(f"   Unhedged Return:     {impact_analysis['returns']['unhedged']:.2f}%")
print(f"   Hedged Return:       {impact_analysis['returns']['hedged']:.2f}%")
print(f"   Difference:          {impact_analysis['returns']['difference']:+.2f}%")

print(f"\n Risk-Adjusted Performance:")
print(f"   Unhedged Sharpe:     {impact_analysis['sharpe_ratio']['unhedged']:.3f}")
print(f"   Hedged Sharpe:       {impact_analysis['sharpe_ratio']['hedged']:.3f}")
print(f"   Improvement:         {impact_analysis['sharpe_ratio']['improvement']:+.3f

print(f"\n Value At Risk (95%):")
print(f"   Unhedged VaR:        {impact_analysis['var_95']['unhedged']:.2f}%")
```

```
print(f"   Hedged VaR:          {impact_analysis['var_95']['hedged']:.2f}%")
print(f"   Improvement:         {impact_analysis['var_95']['improvement']:.2f}%")
```

```
Hedging Impact Analysis
============================================================

  Volatility Impact:
    Unhedged Volatility: 17.13%
    Hedged Volatility:   4.52%
    Reduction:           12.61% (73.6%)

  Drawdown Impact:
    Unhedged Max DD:     -23.67%
    Hedged Max DD:       -8.64%
    Improvement:         15.03% (63.5%)

  Return Impact:
    Unhedged Return:     14.57%
    Hedged Return:       0.43%
    Difference:          -14.14%

  Risk-Adjusted Performance:
    Unhedged Sharpe:     0.733
    Hedged Sharpe:       -0.348
    Improvement:         -1.082

  Value At Risk (95%):
    Unhedged VaR:        -1.70%
    Hedged VaR:          -0.48%
    Improvement:         1.23%
```

```
# Create impact visualization dashboard
fig, axes = plt.subplots(2, 3, figsize=(20, 12))
fig.suptitle('Hedging Impact Analysis Dashboard', fontsize=18, fontweight='bold', y=

# 1. Volatility comparison
ax1 = axes[0, 0]
categories = ['Unhedged', 'Hedged']
volatilities = [impact_analysis['volatility']['unhedged'], impact_analysis['volatili
bars1 = ax1.bar(categories, volatilities, color=[colors[0], colors[1]], alpha=0.8)
ax1.set_title('Annualized Volatility Comparison', fontsize=14, fontweight='bold')
ax1.set_ylabel('Volatility (%)')
ax1.grid(True, alpha=0.3)
# Add value labels on bars
for bar, val in zip(bars1, volatilities):
    ax1.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.2,
             f'{val:.1f}%', ha='center', va='bottom', fontweight='bold')

# 2. Maximum drawdown comparison
ax2 = axes[0, 1]
drawdowns = [abs(impact_analysis['max_drawdown']['unhedged']), abs(impact_analysis['
bars2 = ax2.bar(categories, drawdowns, color=[colors[0], colors[1]], alpha=0.8)
ax2.set_title('Maximum Drawdown Comparison', fontsize=14, fontweight='bold')
```

```
ax2.set_ylabel('Max Drawdown (%)')
ax2.grid(True, alpha=0.3)
for bar, val in zip(bars2, drawdowns):
    ax2.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.2,
             f'{val:.1f}%', ha='center', va='bottom', fontweight='bold')

# 3. Sharpe ratio comparison
ax3 = axes[0, 2]
sharpe_ratios = [impact_analysis['sharpe_ratio']['unhedged'], impact_analysis['sharp
bars3 = ax3.bar(categories, sharpe_ratios, color=[colors[0], colors[1]], alpha=0.8)
ax3.set_title('Sharpe Ratio Comparison', fontsize=14, fontweight='bold')
ax3.set_ylabel('Sharpe Ratio')
ax3.grid(True, alpha=0.3)
for bar, val in zip(bars3, sharpe_ratios):
    ax3.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 0.01,
             f'{val:.3f}', ha='center', va='bottom', fontweight='bold')

# 4. Rolling volatility comparison
ax4 = axes[1, 0]
window = 60
unhedged_rolling_vol = portfolio_data['Portfolio_Return'].rolling(window).std() * np
hedged_rolling_vol = portfolio_data['Hedged_Portfolio_Return'].rolling(window).std()

ax4.plot(portfolio_data.index, unhedged_rolling_vol, label='Unhedged', color=colors[
ax4.plot(portfolio_data.index, hedged_rolling_vol, label='Hedged', color=colors[1],
ax4.fill_between(portfolio_data.index, unhedged_rolling_vol, hedged_rolling_vol,
                alpha=0.3, color='green', where=(hedged_rolling_vol <= unhedged_roll
                label='Volatility Reduction')
ax4.set_title(f'Rolling Volatility ({window}-day)', fontsize=14, fontweight='bold')
ax4.set_ylabel('Volatility (%)')
ax4.legend()
ax4.grid(True, alpha=0.3)

# 5. Risk-return scatter plot
ax5 = axes[1, 1]
# Calculate rolling metrics for scatter
window_short = 30
unhedged_rolling_ret = portfolio_data['Portfolio_Return'].rolling(window_short).mean
hedged_rolling_ret = portfolio_data['Hedged_Portfolio_Return'].rolling(window_short)
unhedged_rolling_vol_short = portfolio_data['Portfolio_Return'].rolling(window_short
hedged_rolling_vol_short = portfolio_data['Hedged_Portfolio_Return'].rolling(window_

ax5.scatter(unhedged_rolling_vol_short, unhedged_rolling_ret,
           alpha=0.6, color=colors[0], label='Unhedged', s=20)
ax5.scatter(hedged_rolling_vol_short, hedged_rolling_ret,
           alpha=0.6, color=colors[1], label='Hedged', s=20)
ax5.set_title('Risk-Return Scatter (30-day Rolling)', fontsize=14, fontweight='bold'
ax5.set_xlabel('Volatility (%)')
ax5.set_ylabel('Return (%)')
ax5.legend()
ax5.grid(True, alpha=0.3)
```

```
# 6. VaR improvement over time
ax6 = axes[1, 2]
unhedged_rolling_var = portfolio_data['Portfolio_Return'].rolling(window).quantile(0
hedged_rolling_var = portfolio_data['Hedged_Portfolio_Return'].rolling(window).quant

ax6.plot(portfolio_data.index, abs(unhedged_rolling_var), label='Unhedged |VaR|',
         color=colors[0], linewidth=2)
ax6.plot(portfolio_data.index, abs(hedged_rolling_var), label='Hedged |VaR|',
         color=colors[1], linewidth=2)
ax6.fill_between(portfolio_data.index, abs(unhedged_rolling_var), abs(hedged_rolling
                alpha=0.3, color='green', where=(abs(hedged_rolling_var) <= abs(unhe
                label='VaR Improvement')
ax6.set_title(f'Rolling VaR 95% ({window}-day)', fontsize=14, fontweight='bold')
ax6.set_ylabel('|VaR| (%)')
ax6.legend()
ax6.grid(True, alpha=0.3)

plt.tight_layout()
plt.savefig('hedging_impact_analysis.png', dpi=300, bbox_inches='tight')
plt.show()

print("Hedging impact analysis dashboard created and saved.")
```


```
Hedging impact analysis dashboard created and saved.
```

Section 6: Signal-Based Weighting Impact (if available)

```
# Analyze signal-based weighting impact if data is available
if signal_performance is not None:
    print("Signal-Based Portfolio Analysis")
    print("=" * 50)

    # Create signal impact visualization
    fig, axes = plt.subplots(2, 2, figsize=(16, 12))
    fig.suptitle('Signal-Based Weighting Impact on Portfolio Dynamics', fontsize=16,

    # Note: This section would need to be customized based on the actual structure
    # of the signal_performance data from Part 2.5

    # For now, create a placeholder analysis
    ax1 = axes[0, 0]
    ax1.text(0.5, 0.5, 'Signal-based portfolio\nanalysis would be\ndisplayed here\n\
             ha='center', va='center', transform=ax1.transAxes, fontsize=12,
             bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.7))
    ax1.set_title('Portfolio Weight Dynamics')
    ax1.set_xticks([])
    ax1.set_yticks([])

    ax2 = axes[0, 1]
    ax2.text(0.5, 0.5, 'Signal strength\nover time analysis',
             ha='center', va='center', transform=ax2.transAxes, fontsize=12,
             bbox=dict(boxstyle='round', facecolor='lightgreen', alpha=0.7))
    ax2.set_title('Signal Strength Evolution')
    ax2.set_xticks([])
    ax2.set_yticks([])

    ax3 = axes[1, 0]
    ax3.text(0.5, 0.5, 'Portfolio turnover\nand rebalancing\nfrequency analysis',
             ha='center', va='center', transform=ax3.transAxes, fontsize=12,
             bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.7))
    ax3.set_title('Portfolio Turnover Analysis')
    ax3.set_xticks([])
    ax3.set_yticks([])

    ax4 = axes[1, 1]
    ax4.text(0.5, 0.5, 'Signal-based vs\nequal-weight\nperformance comparison',
             ha='center', va='center', transform=ax4.transAxes, fontsize=12,
             bbox=dict(boxstyle='round', facecolor='lightcoral', alpha=0.7))
    ax4.set_title('Performance Attribution')
    ax4.set_xticks([])
    ax4.set_yticks([])

    plt.tight_layout()
    plt.savefig('signal_weighting_impact.png', dpi=300, bbox_inches='tight')
    plt.show()

    print("Signal-based analysis placeholder created.")
```

```
        print("Note: Detailed analysis depends on specific signal data structure from Pa

    else:
        print("Signal-based portfolio data not available.")
        print("Creating 2x2 placeholder visualization for signal-based analysis...")

        # Create the 2x2 grid layout that matches the screenshot
        fig, axes = plt.subplots(2, 2, figsize=(16, 12))
        fig.suptitle('Signal-Based Weighting Impact on Portfolio Dynamics', fontsize=16,

        # Top left: Portfolio Weight Dynamics
        ax1 = axes[0, 0]
        ax1.text(0.5, 0.5, 'Signal-based portfolio\nanalysis would be\ndisplayed here\n\
                 ha='center', va='center', transform=ax1.transAxes, fontsize=12,
                 bbox=dict(boxstyle='round', facecolor='lightblue', alpha=0.7))
        ax1.set_title('Portfolio Weight Dynamics')
        ax1.set_xticks([])
        ax1.set_yticks([])

        # Top right: Signal Strength Evolution
        ax2 = axes[0, 1]
        ax2.text(0.5, 0.5, 'Signal strength\nover time analysis',
                 ha='center', va='center', transform=ax2.transAxes, fontsize=12,
                 bbox=dict(boxstyle='round', facecolor='lightgreen', alpha=0.7))
        ax2.set_title('Signal Strength Evolution')
        ax2.set_xticks([])
        ax2.set_yticks([])

        # Bottom left: Portfolio Turnover Analysis
        ax3 = axes[1, 0]
        ax3.text(0.5, 0.5, 'Portfolio turnover\nand rebalancing\nfrequency analysis',
                 ha='center', va='center', transform=ax3.transAxes, fontsize=12,
                 bbox=dict(boxstyle='round', facecolor='lightyellow', alpha=0.7))
        ax3.set_title('Portfolio Turnover Analysis')
        ax3.set_xticks([])
        ax3.set_yticks([])

        # Bottom right: Performance Attribution
        ax4 = axes[1, 1]
        ax4.text(0.5, 0.5, 'Signal-based vs\nequal-weight\nperformance comparison',
                 ha='center', va='center', transform=ax4.transAxes, fontsize=12,
                 bbox=dict(boxstyle='round', facecolor='lightcoral', alpha=0.7))
        ax4.set_title('Performance Attribution')
        ax4.set_xticks([])
        ax4.set_yticks([])

        plt.tight_layout()
        plt.savefig('signal_weighting_impact.png', dpi=300, bbox_inches='tight')
        plt.show()
```

```
    print("Signal-based analysis placeholder created and saved.")
    print("Note: Detailed analysis depends on specific signal data structure from Pa
```

```
Signal-based portfolio data not available.
Creating 2x2 placeholder visualization for signal-based analysis...
```


```
Signal-based analysis placeholder created and saved.
Note: Detailed analysis depends on specific signal data structure from Part 2.5
```

Section 7: Key Takeaways and Summary

Based on our comprehensive visualization and analysis, here are the critical findings:

Hedging Effectiveness Analysis

Risk Reduction Success:

- The hedging strategy dramatically reduces portfolio volatility from 17.13% to 4.52% (73.6% reduction)
- Maximum drawdown improved significantly from -23.67% to -8.64% (63.5% improvement)
- VaR 95% improved from -1.70% to -0.48%, indicating better tail risk management

Trade-offs Identified:

- Return reduction: Unhedged portfolio generated 14.57% annual return vs. 0.43% for hedged
- This represents the cost of hedging: -14.14% annual return for significant risk reduction
- Sharpe ratio decreased from 0.733 to -0.348, indicating the hedging cost exceeded benefits

Distribution Characteristics

Improved Stability:

- Hedged portfolio shows much tighter return distribution (KDE analysis)
- Reduced kurtosis from 4.882 to 2.343, indicating fewer extreme outcomes
- Improved skewness from -0.139 to 0.236, reducing negative tail risk

Volatility Timing:

- Rolling volatility analysis shows hedging is most effective during high-stress periods
- Consistent volatility reduction across different market regimes

Cost-Benefit Analysis

Transaction Cost Impact:

- Net hedged portfolio shows -38.19% maximum drawdown due to transaction costs
- This indicates hedging costs significantly impact long-term performance
- Regular rebalancing creates substantial friction costs

Risk vs. Return Trade-off:

- The analysis reveals hedging provides excellent risk reduction but at a high return cost
- For risk-averse investors, the trade-off may be acceptable
- For return-focused investors, the unhedged strategy performs better

Key Insights from Visualizations

Portfolio Values Comparison:

- Clear divergence between hedged and unhedged performance over time
- Hedged portfolio maintains more stable value but with limited growth
- Unhedged portfolio shows significant growth with higher volatility

Drawdown Analysis:

- Underwater curves demonstrate hedging's effectiveness during market stress
- Hedged portfolio recovers faster from drawdowns
- Risk-adjusted benefits are most apparent during bear markets (2022)

Return Distribution Analysis:

- Histograms and KDE plots show dramatically different risk profiles
- Hedged portfolio has much narrower return distribution
- Box plots reveal reduced outlier risk for hedged strategy

Strategic Recommendations

For Different Investor Types:

1. **Risk-Averse Investors**: Hedging strategy provides valuable downside protection
2. **Growth-Oriented Investors**: Unhedged strategy offers superior long-term returns
3. **Institutional Investors**: Consider partial hedging to balance risk and return

Implementation Considerations:

1. **Market Timing**: Hedging most valuable during volatile periods
2. **Cost Management**: Transaction costs significantly impact hedged performance
3. **Dynamic Approach**: Consider adaptive hedging based on market conditions

Academic Implications