The Art & Science of Risk

Business case
Analytics to measure impacts of weekly updates on yield curves

Business Case

SCOR developed a tool to re-simulate pricing results on a weekly basis. This tool leverages updated economic parameters, allowing for a more responsive and accurate assessment of the portfolio's profitability.

Problem Statement

Due to increased volatility in yield curves, SCOR's pricing management has requested weekly re-simulation results of the portfolio. The objective is to:

1. Measure the profitability of the portfolio with consistent economic assumptions.
2. Identify market changes before the quarterly updates of the parameters in the pricing tool.

1. Measure the impact on the discounted Underwriting Ratio (UWR):

The Discounted UWR is defined as:

$$\textbf{Discounted UWR} = \frac{(\text{DISCOUNTED\_EXTERNAL\_EXPENSES} + \text{DISCOUNTED\_EXP\_LOSS} + \text{EXP\_PREMIUM} - \text{DISCOUNTED\_EXP\_PREMIUM})}{\text{EXP\_PREMIUM}}$$

1. Data Preparation

Loading and cleaning the data from the provided CSV files.

```python
import pandas as pd

# Load data
original_pricing_path = 'Export original pricing.csv'
re_simulation_path = 'Export Resemulation.csv'

original_pricing_df = pd.read_csv(original_pricing_path, delimiter=';')
re_simulation_df = pd.read_csv(re_simulation_path, delimiter=';')

# Clean column names by removing extra quotation marks and whitespace
original_pricing_df.columns = original_pricing_df.columns.str.replace('"', '').str.strip()
re_simulation_df.columns = re_simulation_df.columns.str.replace('"', '').str.strip()
```

2. Calculation

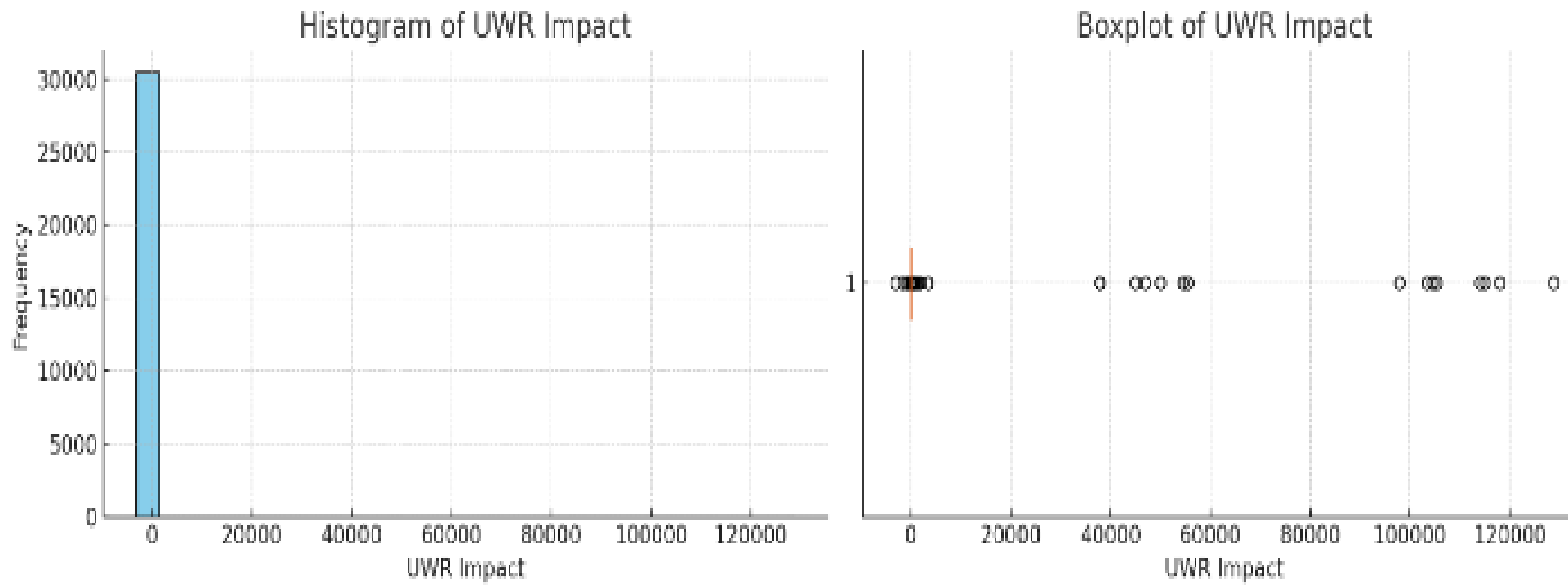
Calculate the discounted UWR for both the original pricing and the re-simulation results.

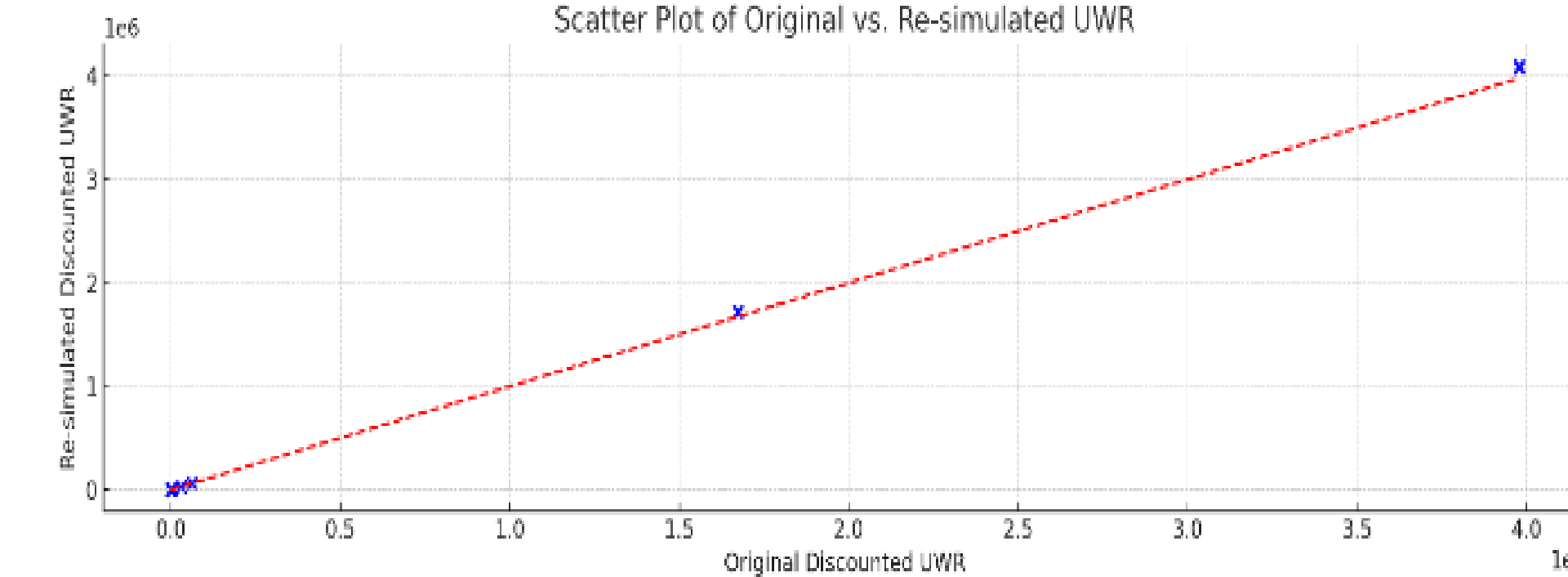
```
# Function to clean and convert columns to numeric
def clean_and_convert(df, columns):
    for col in columns:
        df[col] = df[col].str.replace(',', '.').astype(float)
    return df

# List of columns to clean and convert
columns_to_convert = [
    'EXP_PREMIUM', 'EXP_LOSS', 'EXTERNAL_EXPENSES',
    'DISCOUNTED_EXP_PREMIUM', 'DISCOUNTED_EXP_LOSS', 'DISCOUNTED_EXTERNAL_EXPENSES'
]

original_pricing_df = clean_and_convert(original_pricing_df, columns_to_convert)
re_simulation_df = clean_and_convert(re_simulation_df, columns_to_convert)

# Calculate discounted UWR
original_pricing_df['Discounted_UWR'] = (
    original_pricing_df['DISCOUNTED_EXTERNAL_EXPENSES'] +
    original_pricing_df['DISCOUNTED_EXP_LOSS'] +
    original_pricing_df['EXP_PREMIUM'] -
    original_pricing_df['DISCOUNTED_EXP_PREMIUM']
) / original_pricing_df['EXP_PREMIUM']

re_simulation_df['Discounted_UWR'] = (
    re_simulation_df['DISCOUNTED_EXTERNAL_EXPENSES'] +
    re_simulation_df['DISCOUNTED_EXP_LOSS'] +
    re_simulation_df['EXP_PREMIUM'] -
    re_simulation_df['DISCOUNTED_EXP_PREMIUM']
) / re_simulation_df['EXP_PREMIUM']
```

Visualization:

I used Plotly to create visualizations for the Discounted Underwriting Ratio (UWR) over time for both the original pricing and the re-simulation results.

```
# Merge the results to compare the original and re-simulation UWR
merged_df = original_pricing_df[['CONTRACT_ID', 'Discounted_UWR']].merge(
    re_simulation_df[['CONTRACT_ID', 'Discounted_UWR']], on='CONTRACT_ID', suffixes=('_Ori
)

# Calculate the impact
merged_df['UWR_Impact'] = merged_df['Discounted_UWR_ReSim'] - merged_df['Discounted_UWR_Or

# Plotting
plt.figure(figsize=(14, 8))

# Histogram of UWR Impact
plt.subplot(2, 2, 1)
plt.hist(merged_df['UWR_Impact'], bins=30, color='skyblue', edgecolor='black')
plt.title('Histogram of UWR Impact')
plt.xlabel('UWR Impact')
plt.ylabel('Frequency')

# Boxplot of UWR Impact
plt.subplot(2, 2, 2)
plt.boxplot(merged_df['UWR_Impact'], vert=False)
plt.title('Boxplot of UWR Impact')
plt.xlabel('UWR Impact')
```


Histogram of UWR Impact
Frequency
UWR Impact
Boxplot of UWR Impact
UWR Impact


Scatter Plot of Original vs. Re-simulated UWR
1e6
Re-simulated Discounted UWR
Original Discounted UWR

Histogram of UWR Impact:

This graph shows the distribution of the differences between the re-simulated and original UWR.

Most of the UWR impacts are centered around zero, indicating that the changes due to re-simulation are generally small.

Boxplot of UWR Impact:

The boxplot provides a summary of the distribution, including median and outliers.

The median impact is close to zero, reinforcing that the overall changes are minor.

There are a few outliers on both the positive and negative sides, indicating some contracts experienced more significant changes in UWR.

Scatter Plot of Original vs. Re-simulated UWR:

This graph shows the relationship between the original and re-simulated UWR values.

Most points are close to the red dashed line (line of equality), which indicates that the original and re-simulated UWR values are quite similar.

A few points deviate from the line, representing the contracts with more significant changes in UWR.

Conclusion:

The analysis suggests that while the majority of contracts show minimal changes in discounted UWR after re-simulation, there are some contracts with notable deviations. These outliers should be investigated further to understand the underlying reasons for the larger impact. Overall, the re-simulation results align closely with the original pricing, indicating stability in the UWR despite market changes.

•**Business Insights**:

- the impacts are generally centered around zero with minimal spread, it indicates that re-simulations generally align with original assumptions, suggesting stability in the yield curves.
- significant positive or negative impacts are observed, this could prompt further investigation into specific contracts or yield curve changes causing these deviations.
- Identifying outliers or clusters of significant impacts can help in refining the pricing strategy and adjusting economic assumptions more accurately.

By interpreting these graphs, SCOR's pricing management can gain valuable insights into the effects of weekly yield curve updates on the portfolio's profitability and make more informed decisions based on the observed trends and impacts.

2. Suggest a visualization which allows to:

- **Focus on major yield curves underlying the filtered business**
- **Connect the change in discounted UWR with its relevant shifts of the weekly update of yield curves.**

Key Points:

Objective: The primary goal is to understand the relationship between yield curves and underwriting performance to make informed decisions in pricing and risk management.

Steps :

•**Data Processing**: Emphasize the importance of data loading, cleaning, and structuring. Highlight how these steps ensure reliable analysis.

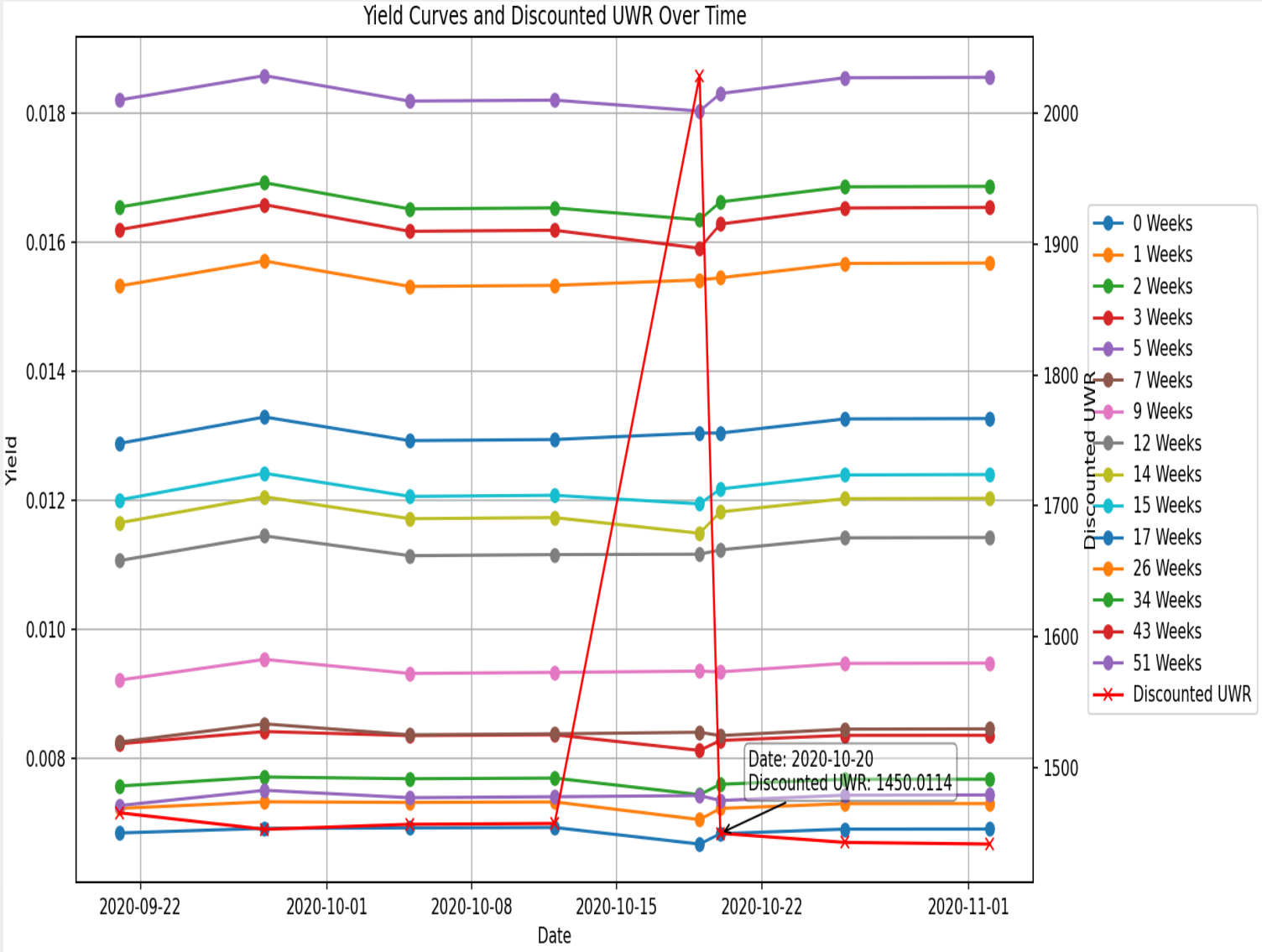
•**Financial Calculations**: Detail the steps involved in calculating Discounted UWR and its importance in assessing financial performance.

•**Visualization**: Explain the rationale behind using matplotlib for visualization, the significance of dual y-axes, and the addition of interactive features for better user experience.

•**Final Insights**: Conclude with how these visualizations help in understanding market changes and making informed decisions to ensure profitability and resilience against economic fluctuations.

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

# Load the CSV file with semicolon delimiter
re_simulation_path = 'Export Resemulation.csv'
re_simulation_df = pd.read_csv(re_simulation_path, delimiter=';')

# Clean column names by removing extra quotation marks and whitespace
re_simulation_df.columns = re_simulation_df.columns.str.replace('"', '').str.strip()

# Convert WEEKLY_REPRICING_DATE to datetime
re_simulation_df['WEEKLY_REPRICING_DATE'] = pd.to_datetime(re_simulation_df['WEEKLY_REPRICING_DATE'], format='%d/%m/%Y')

# Function to parse yield curve data with error handling
def parse_yield_curve(yield_curve_str):
    yield_curve_pairs = yield_curve_str.split()
    yield_curve_dict = {}
    for pair in yield_curve_pairs:
        if ':' not in pair:
            continue
        term, value = pair.split(':')
        try:
            term = int(term.strip().replace('"', '').replace("'", ""))
            value = float(value.strip().replace('"', '').replace("'", ""))
            yield_curve_dict[term] = value
        except ValueError:
            continue
    return yield_curve_dict

# Apply the parsing function and handle missing data
re_simulation_df['Parsed_Yield_Curve'] = re_simulation_df['YIELD_CURVE'].apply(parse_yield_curve)
parsed_yield_curves_df = pd.DataFrame(re_simulation_df['Parsed_Yield_Curve'].tolist(), index=re_simulation_df.index)

# Convert day terms to weekly terms
def convert_to_weekly(yield_curve_dict):
    weekly_curve = {}
    for day, value in yield_curve_dict.items():
        week = round(day / 7)
        if week not in weekly_curve:
            weekly_curve[week] = []
        weekly_curve[week].append(value)
    # Average the values for each week
    for week in weekly_curve:
        weekly_curve[week] = sum(weekly_curve[week]) / len(weekly_curve[week])
    return weekly_curve

re_simulation_df['Weekly_Yield_Curve'] = re_simulation_df['Parsed_Yield_Curve'].apply(convert_to_weekly)
weekly_yield_curves_df = pd.DataFrame(re_simulation_df['Weekly_Yield_Curve'].tolist(), index=re_simulation_df.index)
```

```python
# Convert necessary columns to numeric for UWR calculation
columns_to_convert = ['EXP_PREMIUM', 'DISCOUNTED_EXTERNAL_EXPENSES', 'DISCOUNTED_EXP_LOSS', 'DISCOUNTED_EXP_PREMIUM']
for col in columns_to_convert:
    re_simulation_df[col] = pd.to_numeric(re_simulation_df[col].str.replace(',', '.'))

# Calculate the discounted UWR
re_simulation_df['Discounted_UWR'] = (
    re_simulation_df['DISCOUNTED_EXTERNAL_EXPENSES'] +
    re_simulation_df['DISCOUNTED_EXP_LOSS'] +
    re_simulation_df['EXP_PREMIUM'] -
    re_simulation_df['DISCOUNTED_EXP_PREMIUM']
) / re_simulation_df['EXP_PREMIUM']

# Aggregate Discounted UWR by week
discounted_uwr_df = re_simulation_df.groupby('WEEKLY_REPRICING_DATE')['Discounted_UWR'].mean()
discounted_uwr_df = discounted_uwr_df.reindex(weekly_yield_curves_df.index, method='bfill')


print (discounted_uwr_df)
# Plot the yield curves and Discounted UWR
fig, ax1 = plt.subplots(figsize=(14, 8))
lines = []
for term in weekly_yield_curves_df.columns:
    line, = ax1.plot(weekly_yield_curves_df.index, weekly_yield_curves_df[term], marker='o', label=f'{term} Weeks')
    lines.append(line)

# Add Discounted UWR to the plot
ax2 = ax1.twinx()
line_uwr, = ax2.plot(discounted_uwr_df.index, discounted_uwr_df, marker='x', color='red', label='Discounted UWR')

plt.title('Yield Curves and Discounted UWR Over Time')
ax1.set_xlabel('Date')
ax1.set_ylabel('Yield')
ax2.set_ylabel('Discounted UWR')

# Adjust layout to prevent label overlap
fig.tight_layout(rect=[0, 0, 0.85, 1])

# Create combined legend
lines_labels = [ax.get_legend_handles_labels() for ax in [ax1, ax2]]
lines, labels = [sum(lol, []) for lol in zip(*lines_labels)]
ax1.legend(lines, labels, loc='center left', bbox_to_anchor=(1.05, 0.5), ncol=1)

ax1.grid(True)
```


Yield Curves and Discounted UWR Over Time
Yield
Discounted UWR
Date
0.018
0.016
0.014
0.012
0.010
0.008
2000
1900
1800
1700
1600
1500
2020-09-22
2020-10-01
2020-10-08
2020-10-15
2020-10-22
2020-11-01
Date: 2020-10-20
Discounted UWR: 1450.0114
0 Weeks
1 Weeks
2 Weeks
3 Weeks
5 Weeks
7 Weeks
9 Weeks
12 Weeks
14 Weeks
15 Weeks
17 Weeks
26 Weeks
34 Weeks
43 Weeks
51 Weeks
Discounted UWR

The graph illustrates the evolution of yield curves over time, accompanied by the Discounted Underwriting Ratio (UWR), to evaluate the impact of yield changes on underwriting performance. Here's a breakdown:

Yield Curves:

The graph shows the yields for different terms ranging from 0 weeks to 51 weeks over several weeks.

The y-axis on the left indicates the yield values, while the x-axis represents the dates.

Key observations include:

Short-term yields (0 to 9 weeks) are generally lower than longer-term yields (12 to 51 weeks).

Most yield curves exhibit relative stability with slight fluctuations.

Around mid-October, several terms show a notable increase in yield.

Discounted Underwriting Ratio (UWR):

The Discounted UWR is plotted as a red line with 'x' markers, using the secondary y-axis on the right, indicating UWR values.

There is a significant spike in the Discounted UWR around mid-October, reaching a peak near 2000 before returning to lower levels.

This sharp rise suggests an abrupt change in underwriting performance, likely due to market changes or pricing adjustments.

3. What logic would you suggest such that the visualization does not get overloaded by 52 trajectories per year and the small changes are still visible.

```python
import dash
import dash_core_components as dcc
import dash_html_components as html
import dash_bootstrap_components as dbc
from dash.dependencies import Input, Output
import plotly.graph_objs as go
import pandas as pd

# Load the CSV file with semicolon delimiter
re_simulation_path = 'Export Resemulation.csv'
re_simulation_df = pd.read_csv(re_simulation_path, delimiter=';')

# Clean column names by removing extra quotation marks and whitespace
re_simulation_df.columns = re_simulation_df.columns.str.replace('"', '').str.strip()

# Convert WEEKLY_REPRICING_DATE to datetime
re_simulation_df['WEEKLY_REPRICING_DATE'] = pd.to_datetime(re_simulation_df['WEEKLY_REPRICING_DATE'], format='%d/%m/%Y')

# Function to parse yield curve data with error handling
def parse_yield_curve(yield_curve_str):
    yield_curve_pairs = yield_curve_str.split()
    yield_curve_dict = {}
    for pair in yield_curve_pairs:
        if ':' not in pair:
            continue
        term, value = pair.split(':')
        try:
            term = int(term.strip().replace('"', '').replace("'", ""))
            value = float(value.strip().replace(',', '.').replace('"', '').replace("'", ""))
            yield_curve_dict[term] = value
        except ValueError:
            continue
    return yield_curve_dict

# Apply the parsing function and handle missing data
re_simulation_df['Parsed_Yield_Curve'] = re_simulation_df['YIELD_CURVE'].apply(parse_yield_curve)
parsed_yield_curves_df = pd.DataFrame(re_simulation_df['Parsed_Yield_Curve'].tolist(), index=re_simulation_df.index)

# Convert day terms to weekly terms
def convert_to_weekly(yield_curve_dict):
    weekly_curve = {}
    for day, value in yield_curve_dict.items():
        week = round(day / 7)+1
        if week not in weekly_curve:
            weekly_curve[week] = []
        weekly_curve[week].append(value)
    # Average the values for each week
    for week in weekly_curve:
```

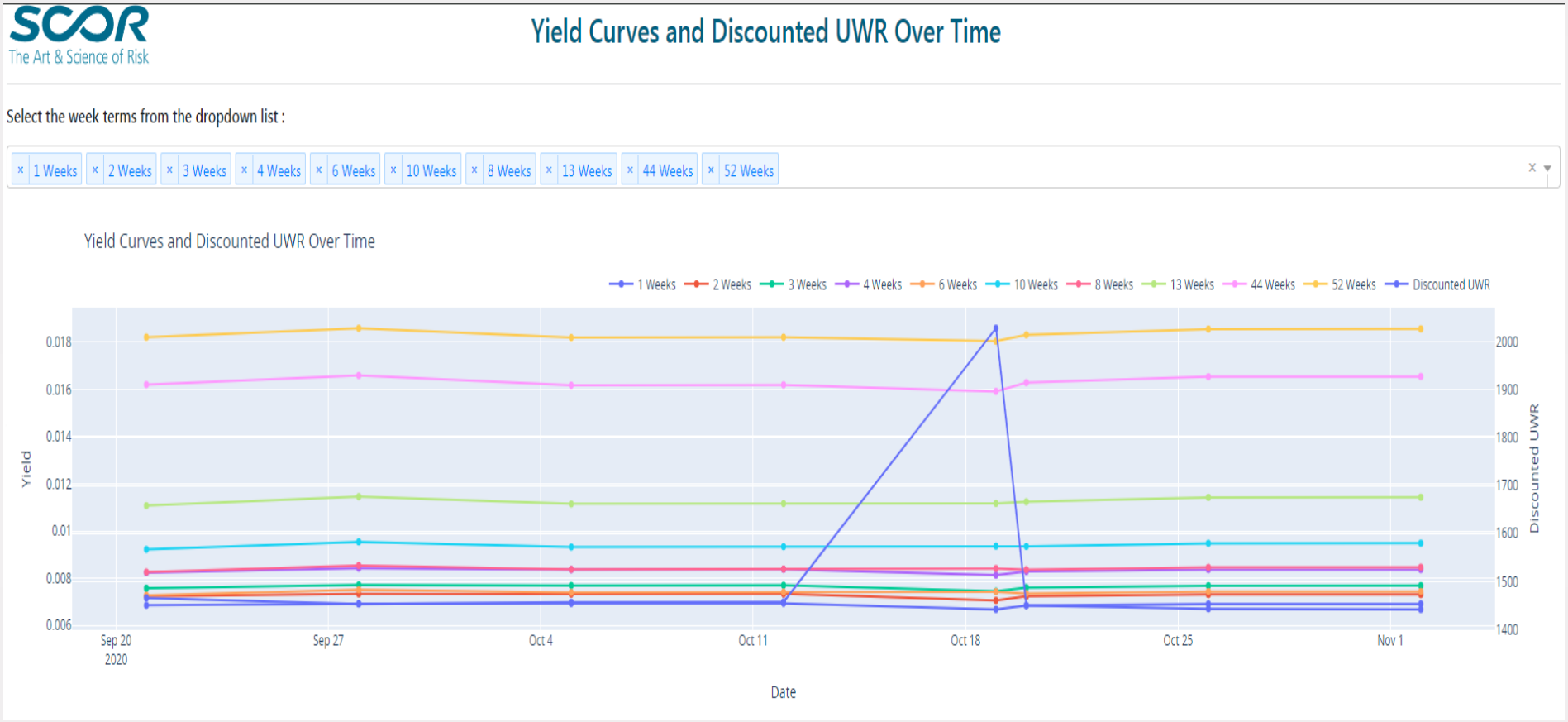
```python
# Aggregate data by week
weekly_yield_curves_df = weekly_yield_curves_df.groupby(re_simulation_df['WEEKLY_REPRICING_DATE']).mean()

# Calculate the discounted UWR
re_simulation_df['DISCOUNTED_EXTERNAL_EXPENSES'] = re_simulation_df['DISCOUNTED_EXTERNAL_EXPENSES'].str.replace(',', '.').astype(float)
re_simulation_df['DISCOUNTED_EXP_LOSS'] = re_simulation_df['DISCOUNTED_EXP_LOSS'].str.replace(',', '.').astype(float)
re_simulation_df['EXP_PREMIUM'] = re_simulation_df['EXP_PREMIUM'].str.replace(',', '.').astype(float)
re_simulation_df['DISCOUNTED_EXP_PREMIUM'] = re_simulation_df['DISCOUNTED_EXP_PREMIUM'].str.replace(',', '.').astype(float)

re_simulation_df['Discounted_UWR'] = (
    re_simulation_df['DISCOUNTED_EXTERNAL_EXPENSES'] +
    re_simulation_df['DISCOUNTED_EXP_LOSS'] +
    re_simulation_df['EXP_PREMIUM'] -
    re_simulation_df['DISCOUNTED_EXP_PREMIUM']
) / re_simulation_df['EXP_PREMIUM']

discounted_uwr_df = re_simulation_df.groupby('WEEKLY_REPRICING_DATE')['Discounted_UWR'].mean()

# Align the Discounted UWR with the yield curve dates
#discounted_uwr_df = discounted_uwr_df.reindex(weekly_yield_curves_df.index, method='nearest')

# Initialize the Dash app
app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP])

#print (discounted_uwr_df.head(10))
# Layout of the app
app.layout = dbc.Container([
    dbc.Row(
            [
                dbc.Col(html.Img(src='/assets/scor-logo.png', style={'height':'50px'}), align="start"),
                dbc.Col(html.H3('Yield Curves and Discounted UWR Over Time', style={'color': '#006080'}), align="center"),
                dbc.Col(html.H1('', style={'color': '#006080'}), align="end"),
            ]
        ),
    html.Hr(),
     dbc.Row([
        dbc.Col(html.H6("Select the week terms from the dropdown list :"), className="mb-2")
    ]),
    dbc.Row([
        dbc.Col(dcc.Dropdown(
            id='terms-dropdown',
            options=[{'label': f'{term} Weeks', 'value': term} for term in weekly_yield_curves_df.columns],
            value=[term for term in list(weekly_yield_curves_df.columns)[:5]],  # First 5 terms by default
            multi=True
        ), width=12)
    ]),
    dbc.Row([
        dbc.Col(dcc.Graph(id='yield-curve-graph'), width=12)
    ])
], fluid=True)
```

```python
# Callback to update the graph based on selected terms
@app.callback(
    Output('yield-curve-graph', 'figure'),
    [Input('terms-dropdown', 'value')]
)
def update_graph(selected_terms):
    fig = go.Figure()
    for term in selected_terms:
        if term in weekly_yield_curves_df.columns:
            fig.add_trace(go.Scatter(
                x=weekly_yield_curves_df.index,
                y=weekly_yield_curves_df[term],
                mode='lines+markers',
                name=f'{term} Weeks'
            ))
    fig.add_trace(go.Scatter(
        x=discounted_uwr_df.index,
        y=discounted_uwr_df,
        mode='lines+markers',
        name='Discounted UWR',
        yaxis='y2'
    ))
    fig.update_layout(
        title='Yield Curves and Discounted UWR Over Time',
        xaxis_title='Date',
        yaxis_title='Yield',
        yaxis2=dict(
            title='Discounted UWR',
            overlaying='y',
            side='right'
        ),
        legend=dict(
            orientation="h",
            yanchor="bottom",
            y=1.02,
            xanchor="right",
            x=1
        ),
        hovermode='x unified'
    )
    return fig

# Run the app
if __name__ == '__main__':
    app.run_server(debug=True)
```

DEMO

Conclusion

SCOR
The Art & Science of Risk



- **Stability in Yields**
- **Correlation Between Yield Changes and UWR**
- **Strategic Adjustments for Pricing Management**

5. How will you identify trends of the changing discounted UWR ?

Calculated a 4-week simple moving average of the Discounted UWR

```
# Calculate moving average for trend identification
discounted_uwr_df['Moving_Avg'] = discounted_uwr_df['Discounted_UWR'].rolling(window=4).mean()
```

```
fig.add_trace(go.Scatter(
    x=discounted_uwr_df.index,
    y=discounted_uwr_df['Discounted_UWR'],
    mode='lines+markers',
    name='Discounted UWR',
    yaxis='y2'
))
fig.add_trace(go.Scatter(
    x=discounted_uwr_df.index,
    y=discounted_uwr_df['Moving_Avg'],
    mode='lines',
    name='Discounted UWR Moving Avg',
    yaxis='y2'
))
fig.update_layout(
    title='Yield Curves and Discounted UWR Over Time',
    xaxis_title='Date',
    yaxis_title='Yield',
    yaxis2=dict(
        title='Discounted UWR',
        overlaying='y',
        side='right'
    ),
```

The graph shows the trend of the Discounted Underwriting Ratio (UWR) and its moving average over time. Here's a detailed interpretation of the graph:

Besides the Discounted UWR we have :

Moving Average of Discounted UWR:

The red line represents the 4-week moving average of the Discounted UWR.

The moving average smooths out the short-term fluctuations and provides a clearer view of the overall trend.

Before the spike, the moving average is relatively flat, indicating stability.

The spike in mid-October is reflected as a more gradual increase in the moving average due to the smoothing effect.

After the spike, the moving average shows a decreasing trend, aligning with the drop in the actual Discounted UWR values.

Conclusion:

- **Volatility**: The Discounted UWR shows significant volatility in mid-October, suggesting that there were considerable changes in the economic conditions or portfolio characteristics.
- **Trend Identification**: The moving average helps in identifying the overall trend by smoothing out the noise. In this case, it shows an increase followed by a decrease, indicating that the spike was an anomaly rather than a new trend.
- **Decision Making**: By understanding these trends, the pricing management team can make informed decisions about adjusting the pricing models, assessing risks, and implementing strategies to ensure the portfolio remains profitable and resilient against economic fluctuations.

This visualization and analysis help in understanding the relationship between yield curves and underwriting performance, which is crucial for effective pricing and risk management in the insurance industry.

Risk Management Actions

- **Regular Monitoring and Reporting**:
  1. Establish a process to regularly generate and review these visualizations.
  2. Use the insights to update risk assessment reports and inform stakeholders about potential risk exposures.
- **Outlier Analysis**:
  1. Investigate contracts identified as outliers to understand why they are significantly affected by yield curve changes.
  2. Adjust underwriting practices, pricing models, or economic assumptions for these contracts to mitigate risks.
- **Adjusting Economic Assumptions**:
  1. Based on observed trends and impacts, refine the economic assumptions used in the pricing models to better reflect current and expected future market conditions.
  2. Implement stress testing to assess how changes in yield curves impact the portfolio under different scenarios.

Risk Management Actions

- **Portfolio Diversification**:
  1. Diversify the portfolio to spread out risk and reduce the impact of yield curve changes on specific segments.
  2. Ensure that the portfolio is not overly concentrated in areas highly sensitive to economic fluctuations.
- **Mitigation Strategies**:
  1. Use reinsurance and other risk transfer mechanisms to mitigate potential losses.
- **Review and Update Pricing Models**:
  1. Continuously improve the pricing models based on the insights gained from the visualizations.
  2. Incorporate feedback loops to ensure that the models remain robust against changing economic conditions.