```
from google.colab import drive
drive.mount("/content/drive")
```

```
Mounted at /content/drive
```

```
!pwd
```

```
/content
```

```
!ls /content/drive/MyDrive/Dataset/Levine_32dim.fcs.csv
```

```
/content/drive/MyDrive/Dataset/Levine_32dim.fcs.csv
```

```
import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/Dataset/Levine_32dim.fcs.csv")
```

```
df.head()
```

	Event	Time	Cell_length	DNA1	DNA2	CD45RA	CD133	CD19	CD22	CD11b	...	CD117	CD49d	HLA-DR	CD64	CD41	Viability	file_number
0	1	2693.0	22	4.391057	4.617262	0.162691	-0.029585	-0.006696	0.066388	-0.009184	...	0.053050	0.853505	1.664480	-0.005376	-0.001961	0.648429	3.627711
1	2	3736.0	35	4.340481	4.816692	0.701349	-0.038280	-0.016654	0.074409	0.808031	...	0.089660	0.197818	0.491592	0.144814	0.868014	0.561384	3.627711
2	3	7015.0	32	3.838727	4.386369	0.603568	-0.032216	0.073855	-0.042977	-0.001881	...	0.046222	2.586670	1.308337	-0.010961	-0.010413	0.643337	3.627711
3	4	7099.0	29	4.255806	4.830048	0.433747	-0.027611	-0.017661	-0.044072	0.733698	...	0.066470	1.338669	0.140523	-0.013449	-0.026039	-0.026523	3.627711
4	5	7700.0	25	3.976909	4.506433	-0.008809	-0.030297	0.080423	0.495791	1.107627	...	-0.006223	0.180924	0.197332	0.076167	-0.040488	0.283287	3.627711

5 rows × 42 columns

```
df.columns
```

```
Index(['Event', 'Time', 'Cell_length', 'DNA1', 'DNA2', 'CD45RA', 'CD133',
       'CD19', 'CD22', 'CD11b', 'CD4', 'CD8', 'CD34', 'Flt3', 'CD20', 'CXCR4',
       'CD235ab', 'CD45', 'CD123', 'CD321', 'CD14', 'CD33', 'CD47', 'CD11c',
       'CD7', 'CD15', 'CD16', 'CD44', 'CD38', 'CD13', 'CD3', 'CD61', 'CD117',
       'CD49d', 'HLA-DR', 'CD64', 'CD41', 'Viability', 'file_number',
       'event_number', 'label', 'individual'],
      dtype='object')
```

```
df.info()
```

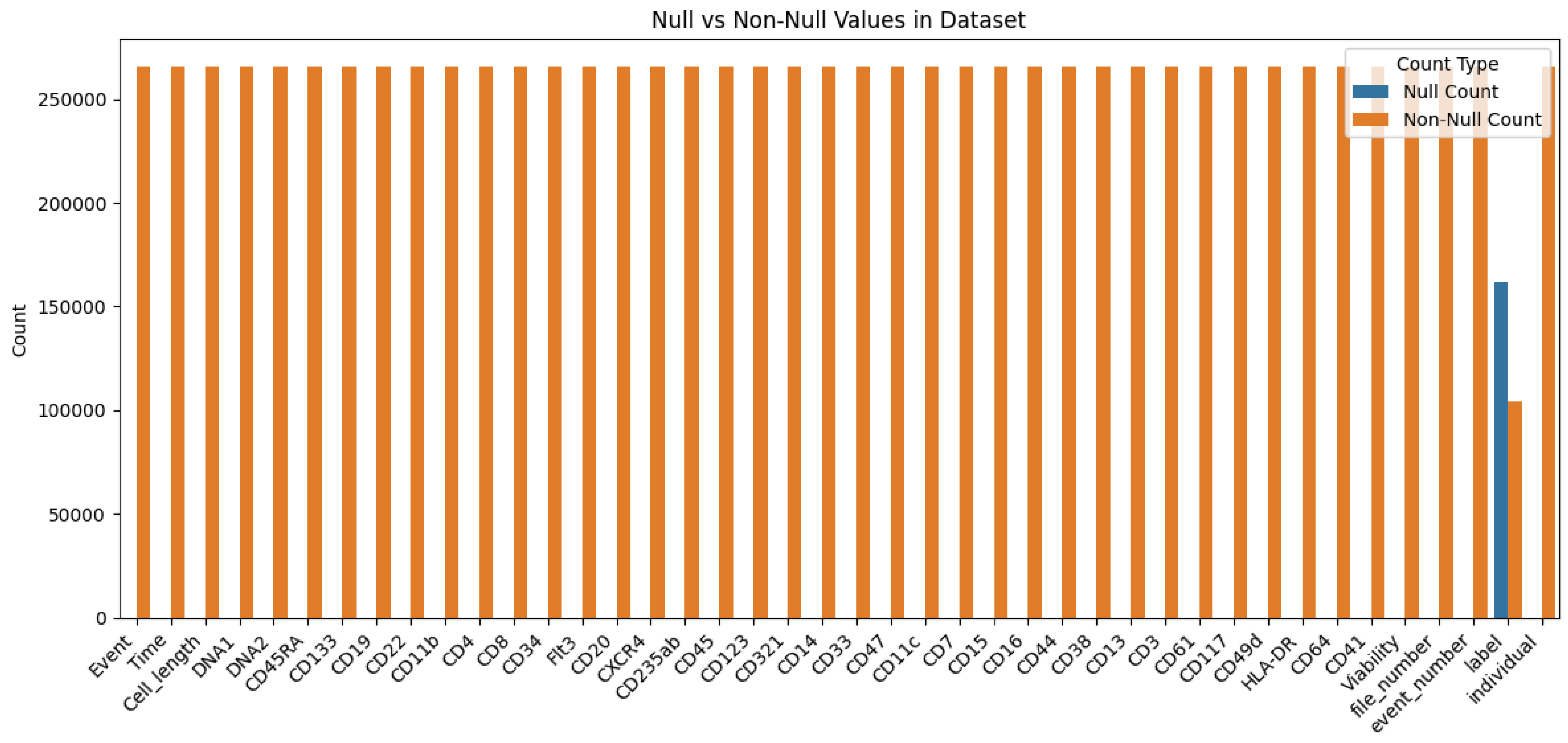
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 265627 entries, 0 to 265626
Data columns (total 42 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   Event         265627 non-null  int64  
 1   Time          265627 non-null  float64
 2   Cell_length   265627 non-null  int64  
 3   DNA1          265627 non-null  float64
 4   DNA2          265627 non-null  float64
 5   CD45RA        265627 non-null  float64
 6   CD133         265627 non-null  float64
 7   CD19          265627 non-null  float64
 8   CD22          265627 non-null  float64
 9   CD11b         265627 non-null  float64
 10  CD4           265627 non-null  float64
 11  CD8           265627 non-null  float64
 12  CD34          265627 non-null  float64
 13  Flt3          265627 non-null  float64
 14  CD20          265627 non-null  float64
 15  CXCR4         265627 non-null  float64
 16  CD235ab       265627 non-null  float64
 17  CD45          265627 non-null  float64
 18  CD123         265627 non-null  float64
 19  CD321         265627 non-null  float64
 20  CD14          265627 non-null  float64
 21  CD33          265627 non-null  float64
 22  CD47          265627 non-null  float64
 23  CD11c         265627 non-null  float64
 24  CD7           265627 non-null  float64
 25  CD15          265627 non-null  float64
 26  CD16          265627 non-null  float64
 27  CD44          265627 non-null  float64
 28  CD38          265627 non-null  float64
 29  CD13          265627 non-null  float64
 30  CD3           265627 non-null  float64
 31  CD61          265627 non-null  float64
 32  CD117         265627 non-null  float64
 33  CD49d         265627 non-null  float64
 34  HLA-DR        265627 non-null  float64
 35  CD64          265627 non-null  float64
 36  CD41          265627 non-null  float64
 37  Viability     265627 non-null  float64
 38  file_number   265627 non-null  float64
 39  event_number  265627 non-null  int64  
 40  label         104184 non-null  float64
 41  individual    265627 non-null  int64  
dtypes: float64(38), int64(4)
memory usage: 85.1 MB
```

```
df.isnull().sum()
```

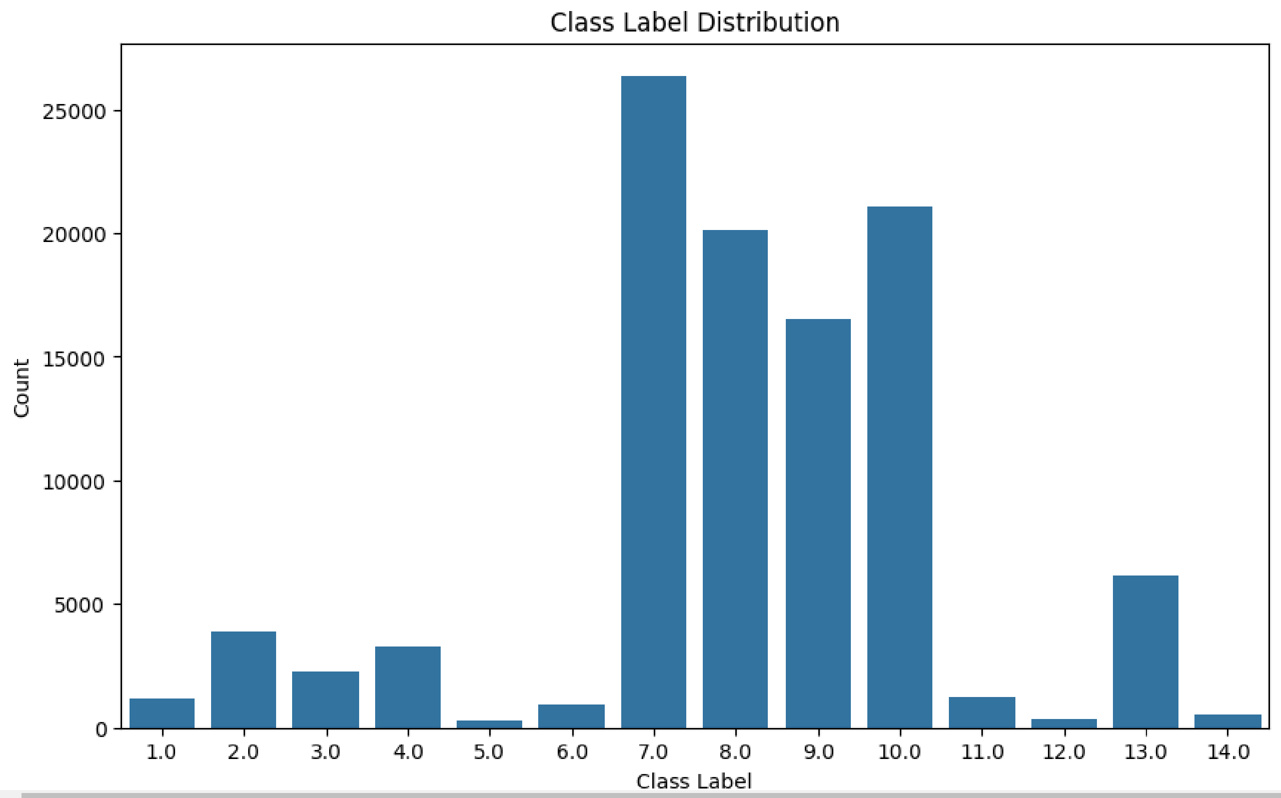
	0
Event	0
Time	0
Cell_length	0
DNA1	0
DNA2	0
CD45RA	0
CD133	0
CD19	0
CD22	0
CD11b	0
CD4	0
CD8	0
CD34	0
Flt3	0
CD20	0
CXCR4	0
CD235ab	0
CD45	0
CD123	0
CD321	0
CD14	0
CD33	0
CD47	0
CD11c	0
CD7	0
CD15	0
CD16	0
CD44	0
CD38	0
CD13	0

```
         CD3        0
        CD61        0
       CD117        0
       CD49d        0
      HLA-DR        0
        CD64        0
        CD41        0
   Viability        0
 file_number        0
event_number        0
       label   161443
  individual        0
```

```
import seaborn as sns
import matplotlib.pyplot as plt
null_counts = df.isnull().sum()
non_null_counts = df.notnull().sum()

null_data = pd.DataFrame({
    'Column': df.columns,
    'Null Count': null_counts,
    'Non-Null Count': non_null_counts
})

null_data_melted = null_data.melt(id_vars='Column', var_name='Count Type', value_name='Count')

plt.figure(figsize=(12, 6))

sns.barplot(data=null_data_melted, x='Column', y='Count', hue='Count Type')

plt.xticks(rotation=45, ha='right')
plt.title('Null vs Non-Null Values in Dataset')
plt.ylabel('Count')
plt.xlabel('Columns')
plt.legend(title='Count Type')
plt.tight_layout()
plt.show()
```



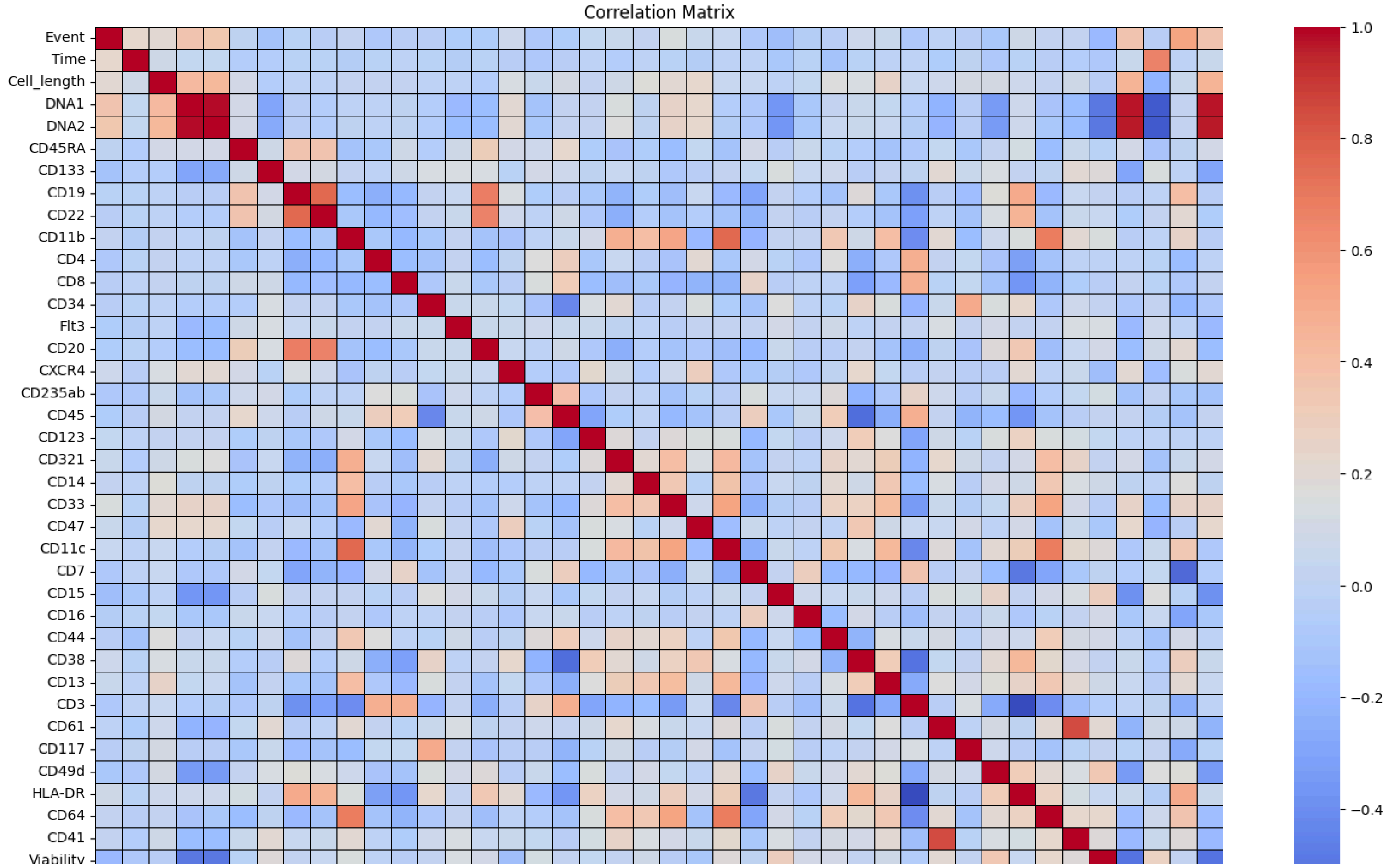
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
label_distribution = df['label'].value_counts(dropna=False)
sns.barplot(x=label_distribution.index, y=label_distribution.values)
plt.xlabel('Class Label')
plt.ylabel('Count')
plt.title('Class Label Distribution')
plt.show()
```

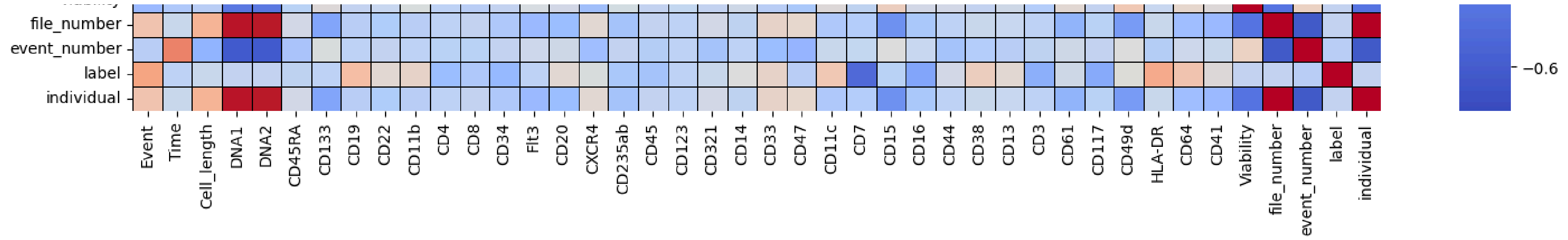


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame

# Calculate the correlation matrix
corr_matrix = df.corr()

# Set the threshold for high correlation (e.g., 0.8)
threshold = 0.8

# Find pairs of highly correlated columns
high_corr_pairs = [(col1, col2, corr_matrix.loc[col1, col2])
                   for col1 in corr_matrix.columns
                   for col2 in corr_matrix.columns
                   if col1 != col2 and abs(corr_matrix.loc[col1, col2]) > threshold]

# Remove duplicate pairs (e.g., (A, B) and (B, A))
unique_high_corr_pairs = []
```

```
    seen = set()
    for col1, col2, corr in high_corr_pairs:
        if (col2, col1) not in seen:
            unique_high_corr_pairs.append((col1, col2, corr))
            seen.add((col1, col2))

    # Print the highly correlated pairs
    print("Highly Correlated Pairs (Threshold > 0.8):")
    for col1, col2, corr in unique_high_corr_pairs:
        print(f"{col1} and {col2}: {corr:.2f}")

    # Plot the correlation matrix heatmap
    plt.figure(figsize=(18, 12))
    sns.heatmap(corr_matrix, annot=False, fmt='.2f', cmap='coolwarm', linewidths=0.5, linecolor='black')
    plt.title('Correlation Matrix')
    plt.show()
```

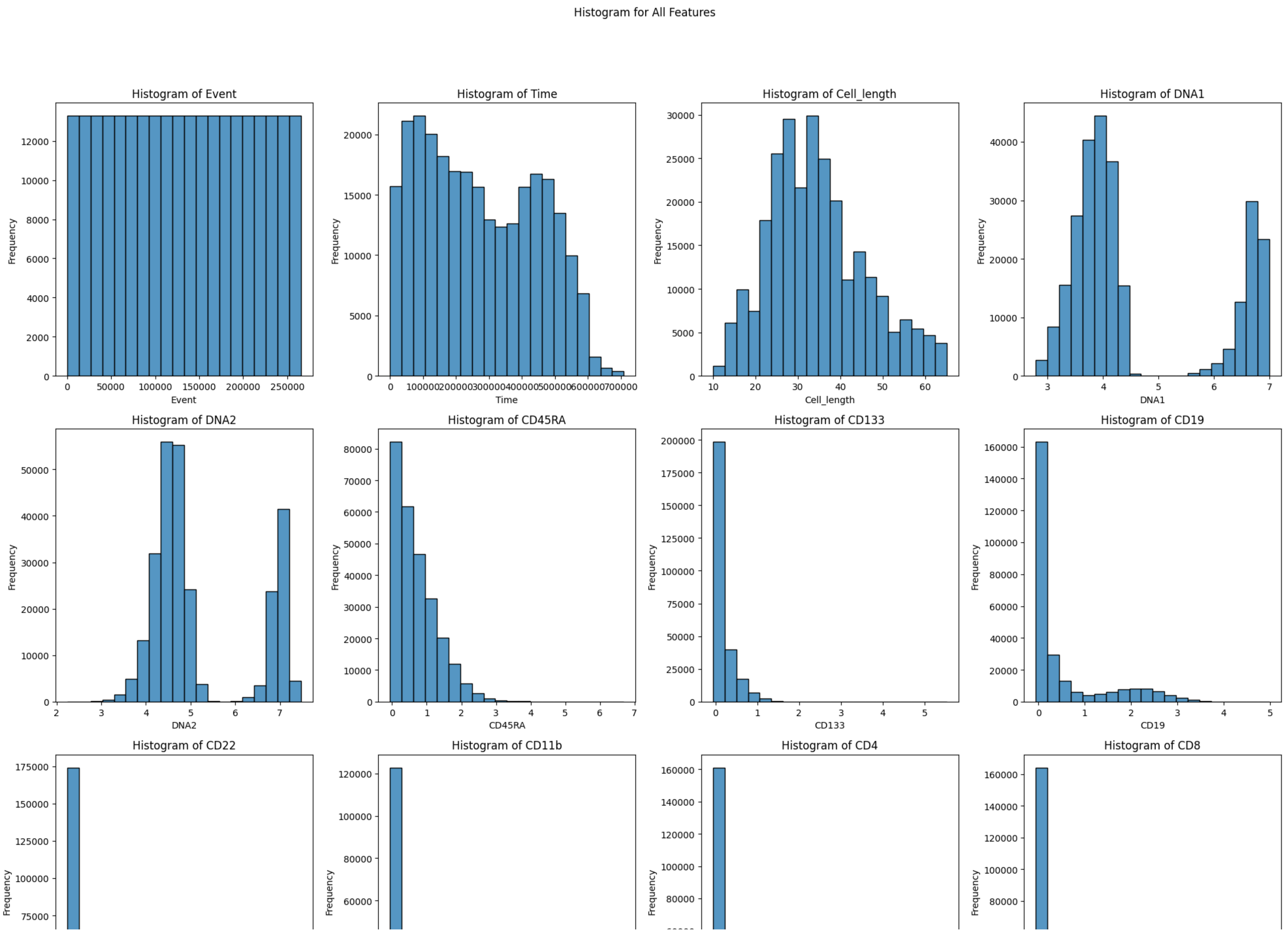
```
Highly Correlated Pairs (Threshold > 0.8):
DNA1 and DNA2: 0.98
DNA1 and file_number: 0.97
DNA1 and individual: 0.97
DNA2 and file_number: 0.96
DNA2 and individual: 0.96
CD61 and CD41: 0.84
file_number and individual: 1.00
```

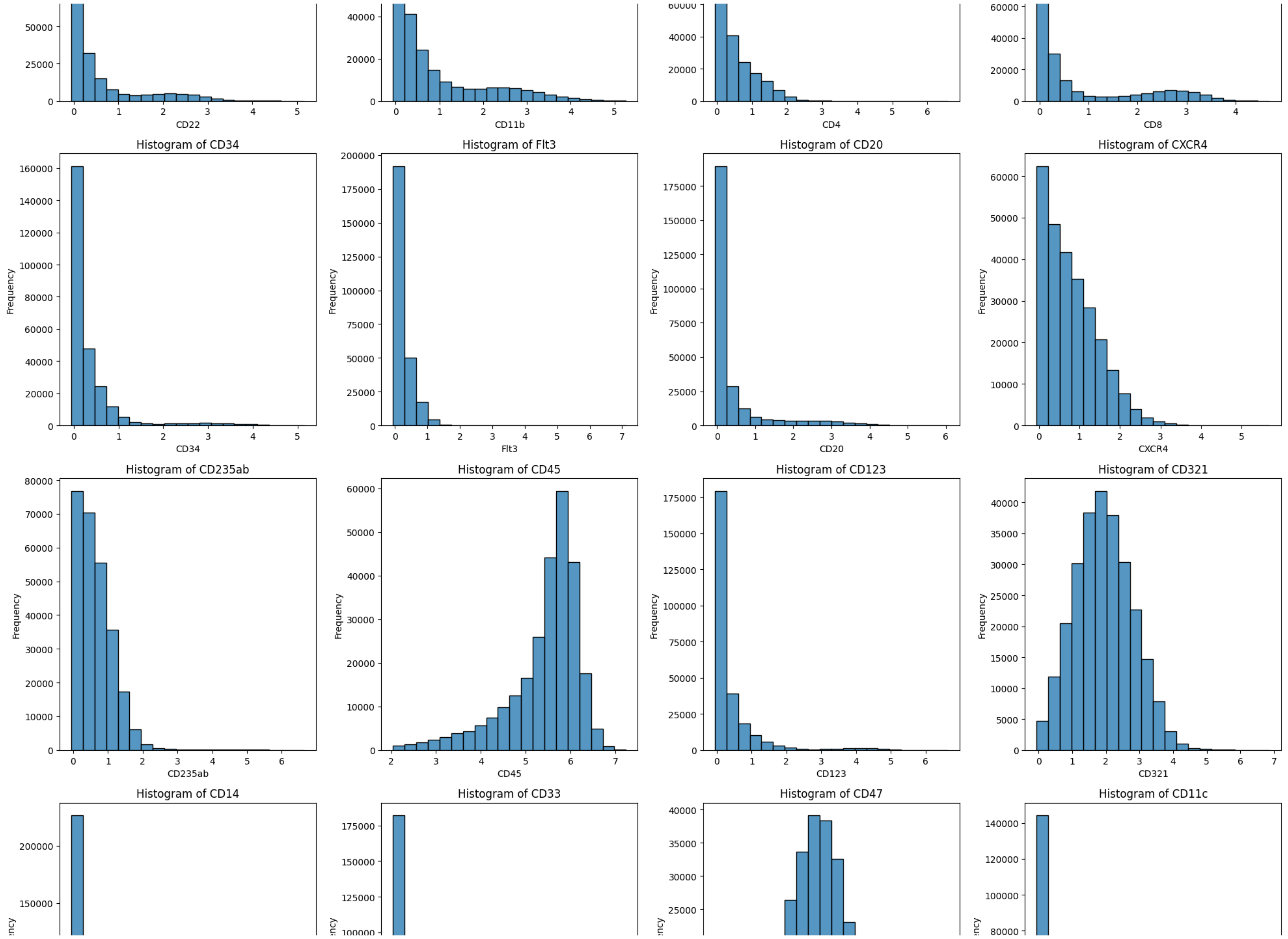


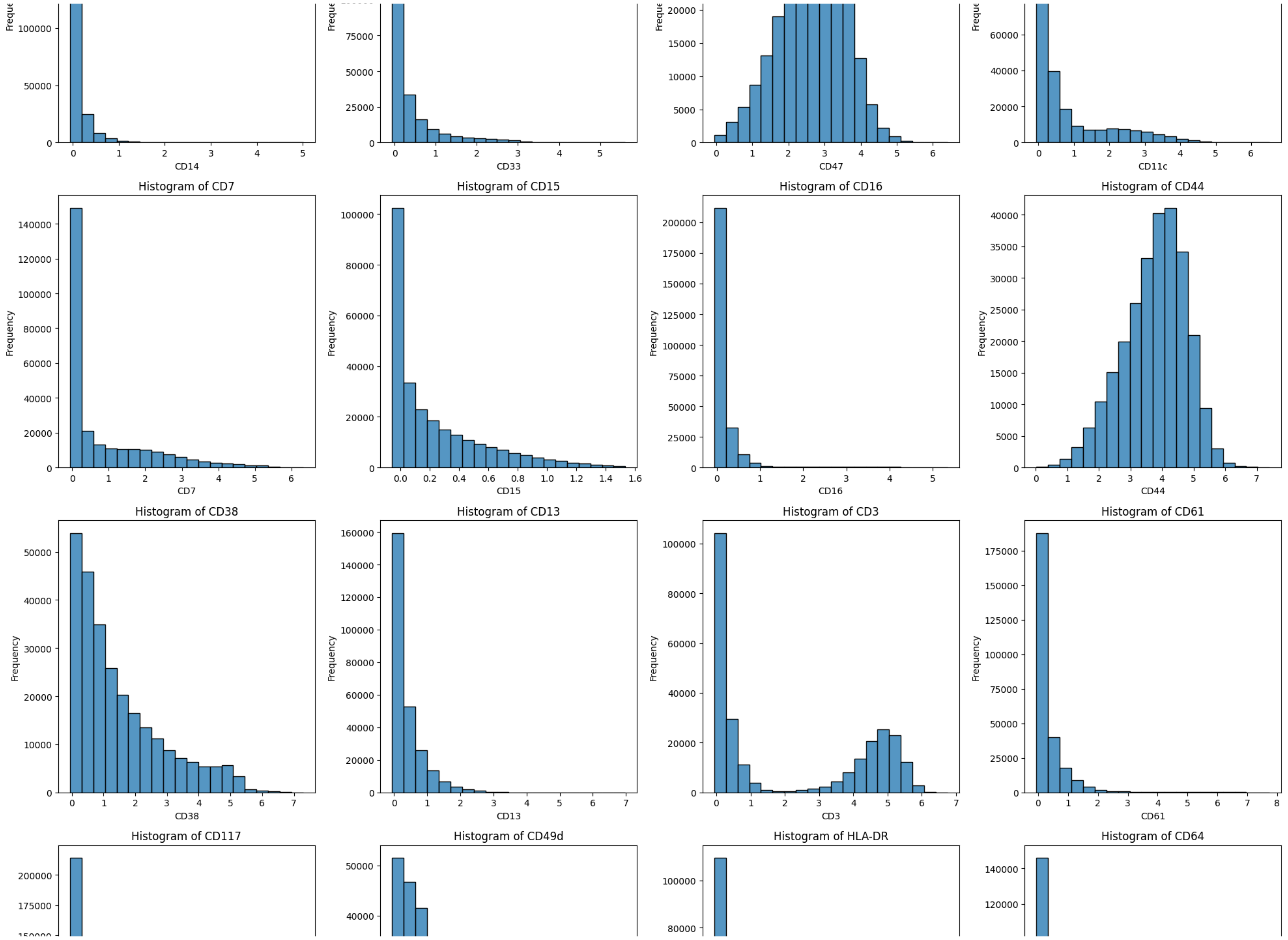


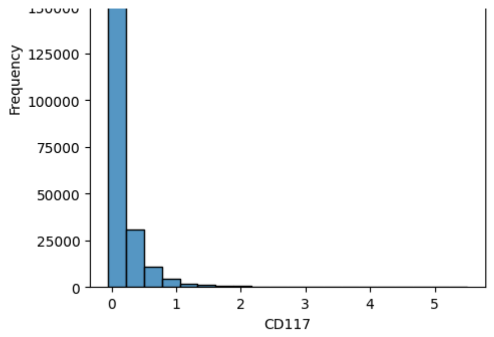
Start coding or generate with AI.

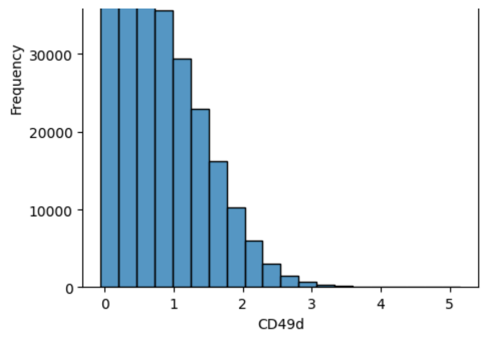
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Assuming df is your DataFrame

# Number of columns in the DataFrame
num_columns = len(df.columns)

# Calculate the number of rows needed for the subplot grid
num_rows = (num_columns + 3) // 4  # 4 histograms per row

# Create subplots
fig, axes = plt.subplots(num_rows, 4, figsize=(20, num_rows * 5))
axes = axes.flatten()

# Plot histogram for each feature
for i, col in enumerate(df.columns):
    sns.histplot(df[col], bins=20, kde=False, ax=axes[i], edgecolor='black')
    axes[i].set_title(f'Histogram of {col}')
    axes[i].set_xlabel(col)
    axes[i].set_ylabel('Frequency')

# Remove any empty subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.suptitle('Histogram for All Features', y=1.02)
plt.show()
```

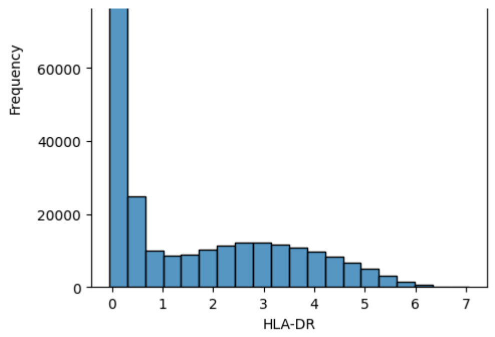

Histogram for All Features
Histogram of Event
Histogram of Time
Histogram of Cell_length
Histogram of DNA1
Histogram of DNA2
Histogram of CD45RA
Histogram of CD133
Histogram of CD19
Histogram of CD22
Histogram of CD11b
Histogram of CD4
Histogram of CD8
Frequency
Event
Time
Cell_length
DNA1
DNA2
CD45RA
CD133
CD19

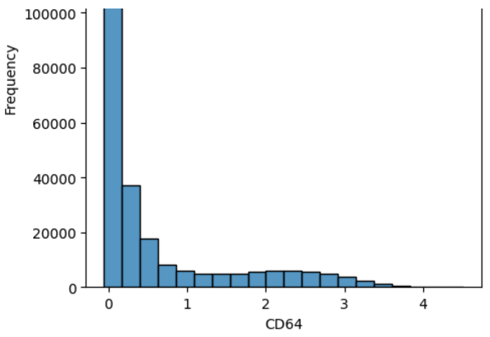

CD22
CD11b
CD4
CD8
Histogram of CD34
CD34
Histogram of Flt3
Flt3
Histogram of CD20
CD20
Histogram of CXCR4
CXCR4
Histogram of CD235ab
CD235ab
Histogram of CD45
CD45
Histogram of CD123
CD123
Histogram of CD321
CD321
Histogram of CD14
Histogram of CD33
Histogram of CD47
Histogram of CD11c
Frequency

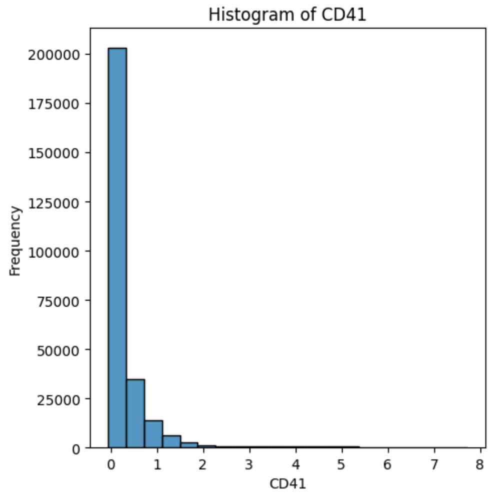

Freque
CD14
CD33
CD47
CD11c
Histogram of CD7
Histogram of CD15
Histogram of CD16
Histogram of CD44
Frequency
CD7
CD15
CD16
CD44
Histogram of CD38
Histogram of CD13
Histogram of CD3
Histogram of CD61
CD38
CD13
CD3
CD61
Histogram of CD117
Histogram of CD49d
Histogram of HLA-DR
Histogram of CD64

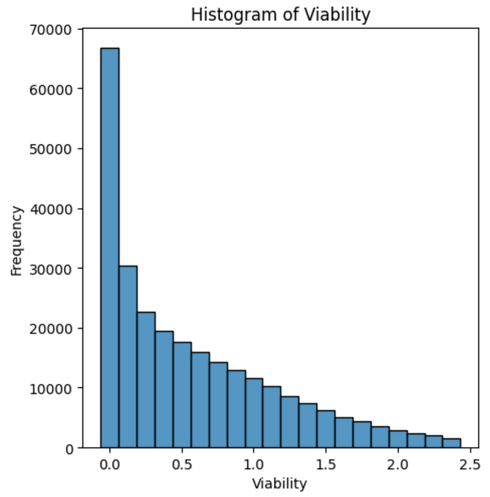

Frequency
CD117

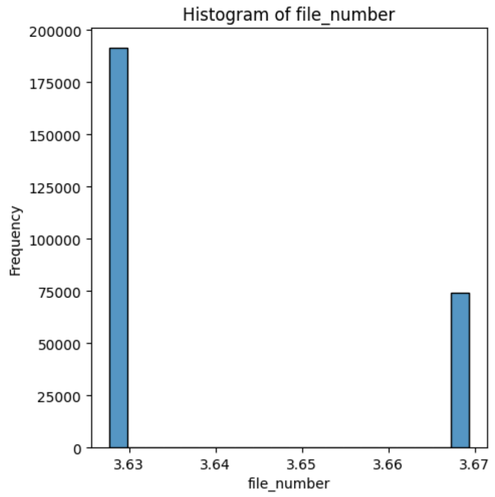

Frequency
CD49d

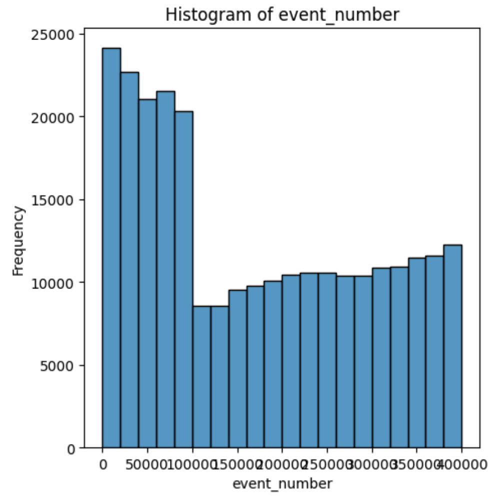

Frequency
HLA-DR

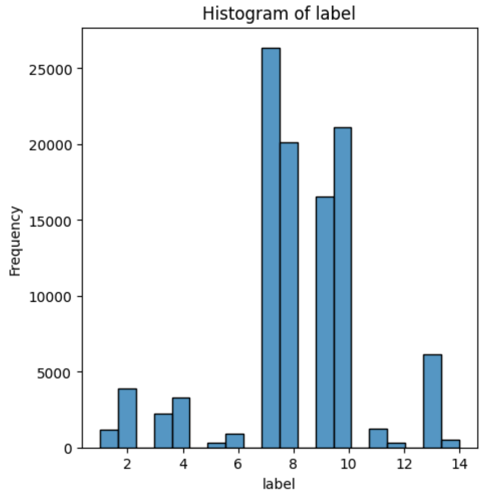

Frequency
CD64

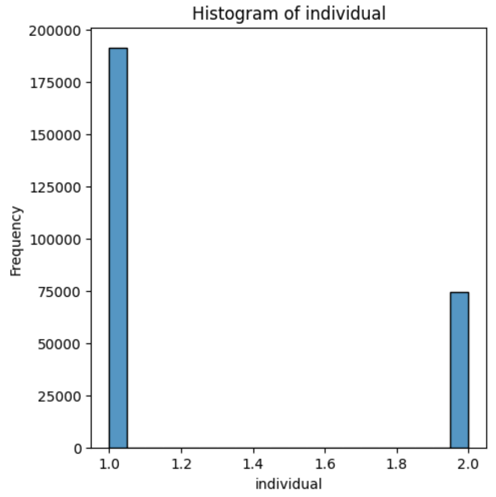

Histogram of CD41
Frequency
CD41


Histogram of Viability
Frequency
Viability

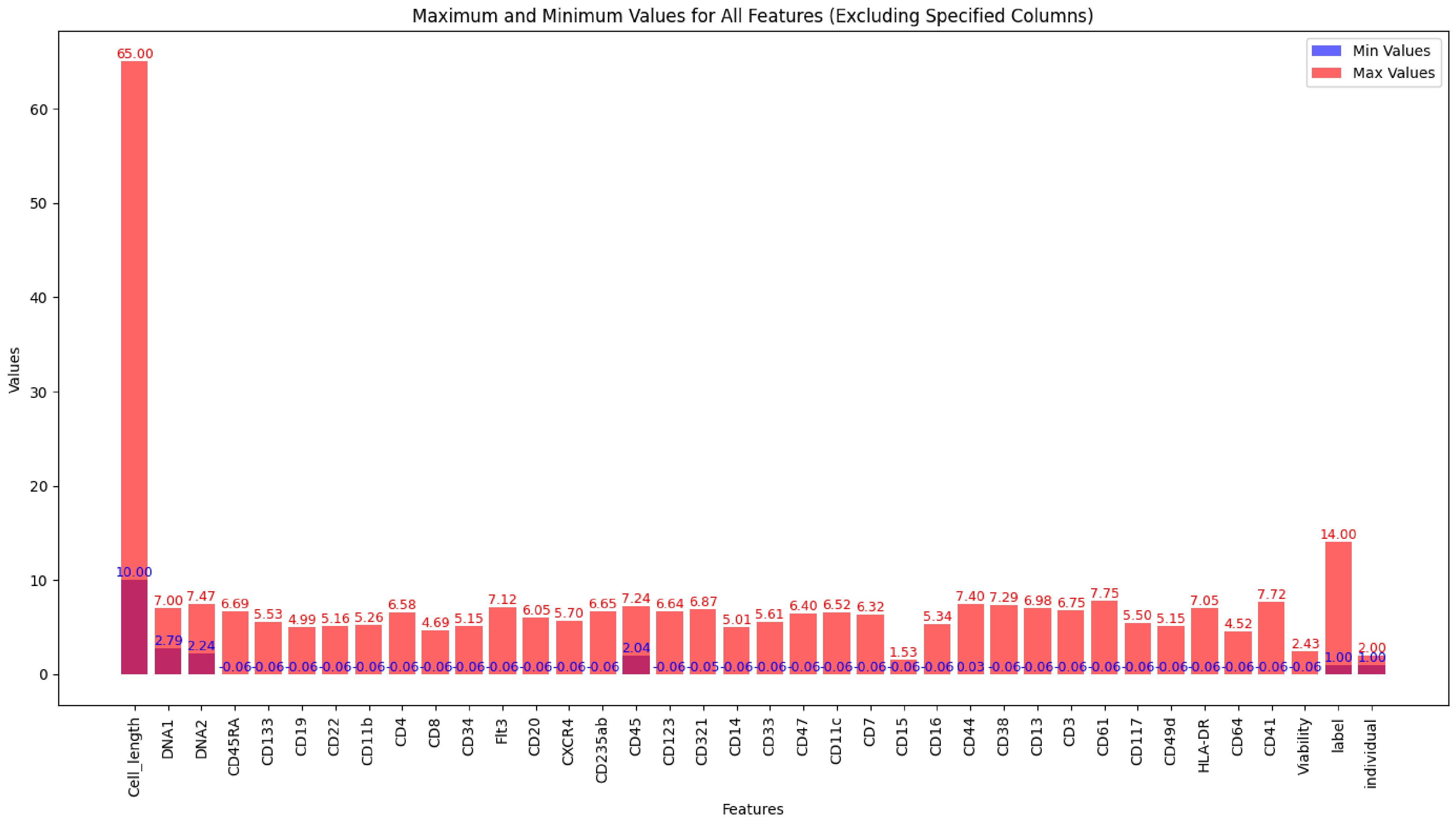

Histogram of file_number
Frequency
file_number


Histogram of event_number
Frequency
event_number


Histogram of label
Frequency
label


Histogram of individual
Frequency
individual

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

# Assuming df is your DataFrame

# Remove specified columns
columns_to_remove = ['Event', 'Time', 'event_number', 'file_number']
df_cleaned = df.drop(columns=columns_to_remove)

# Calculate the max and min values for all features in the cleaned DataFrame
max_values = df_cleaned.max()
min_values = df_cleaned.min()


# Plotting the max and min values
plt.figure(figsize=(14, 8))

# Plotting Min values
plt.bar(min_values.index, min_values.values, alpha=0.6, label='Min Values', color='b')

# Plotting Max values
plt.bar(max_values.index, max_values.values, alpha=0.6, label='Max Values', color='r')

# Annotating Min values
for i, val in enumerate(min_values.values):
    plt.text(i, val, f'{val:.2f}', ha='center', va='bottom', fontsize=9, color='blue')

# Annotating Max values
for i, val in enumerate(max_values.values):
    plt.text(i, val, f'{val:.2f}', ha='center', va='bottom', fontsize=9, color='red')

plt.title('Maximum and Minimum Values for All Features (Excluding Specified Columns)')
plt.xlabel('Features')
plt.ylabel('Values')
plt.xticks(rotation=90)
plt.legend()
plt.tight_layout()
plt.show()
```



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

# Assuming df is your DataFrame

# Remove specified columns
```

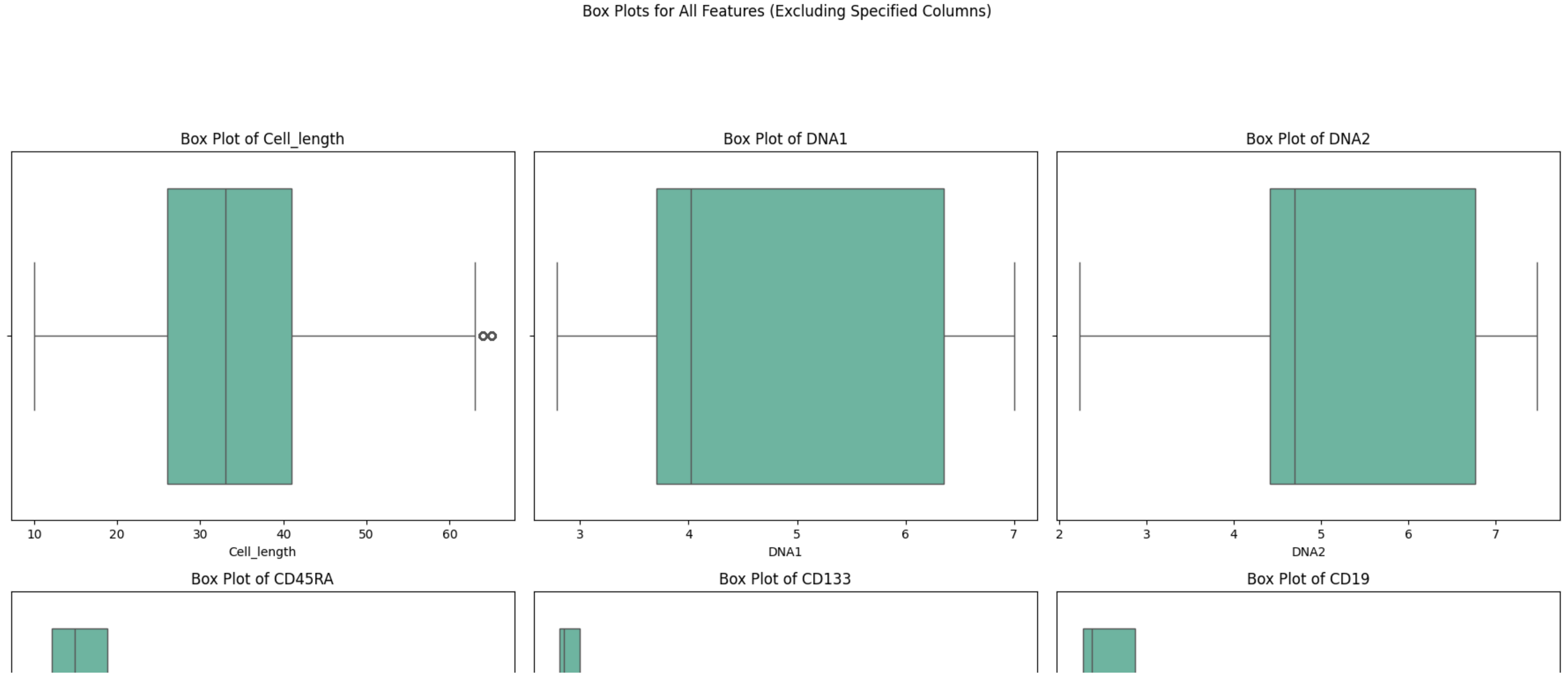
```
    columns_to_remove = ['Event', 'Time', 'event_number', 'file_number']
    df_cleaned = df.drop(columns=columns_to_remove)

    # Calculate number of features
    num_features = len(df_cleaned.columns)

    # Set up the number of rows needed (3 plots per row)
    num_rows = (num_features + 2) // 3  # Use integer division to get rows

    # Create subplots
    fig, axes = plt.subplots(num_rows, 3, figsize=(18, num_rows * 5))
    axes = axes.flatten()  # Flatten the 2D array of axes for easy indexing

    # Create box plot for each feature
    for i, col in enumerate(df_cleaned.columns):
        sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
        axes[i].set_title(f'Box Plot of {col}')
        axes[i].set_xlabel(col)

    # Remove any empty subplots
    for j in range(i + 1, len(axes)):
        fig.delaxes(axes[j])

    plt.tight_layout()
    plt.suptitle('Box Plots for All Features (Excluding Specified Columns)', y=1.02)
    plt.show()
```

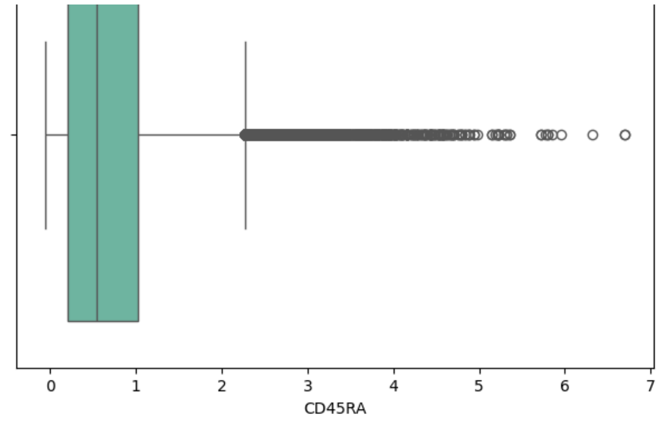
```
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:
```

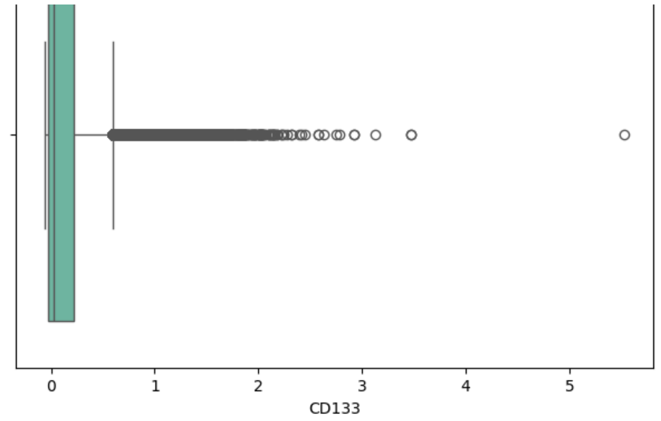
```
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:
```

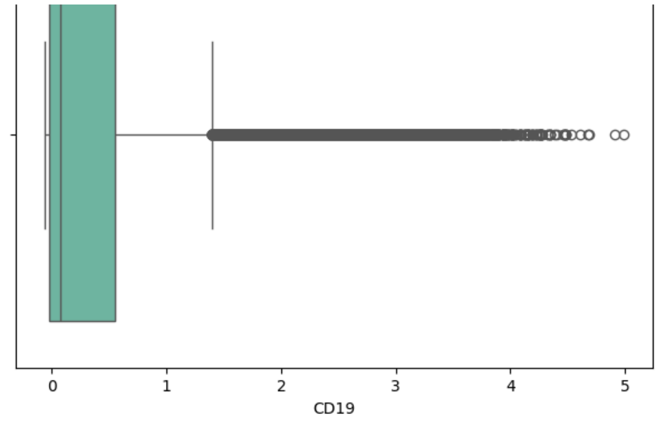
```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:
```

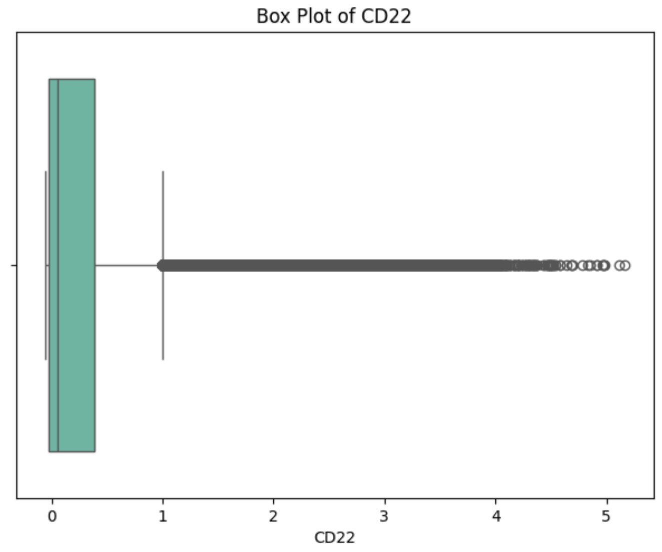
```
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
<ipython-input-14-7f365e39511a>:23: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.boxplot(x=df_cleaned[col], palette='Set2', ax=axes[i])
```

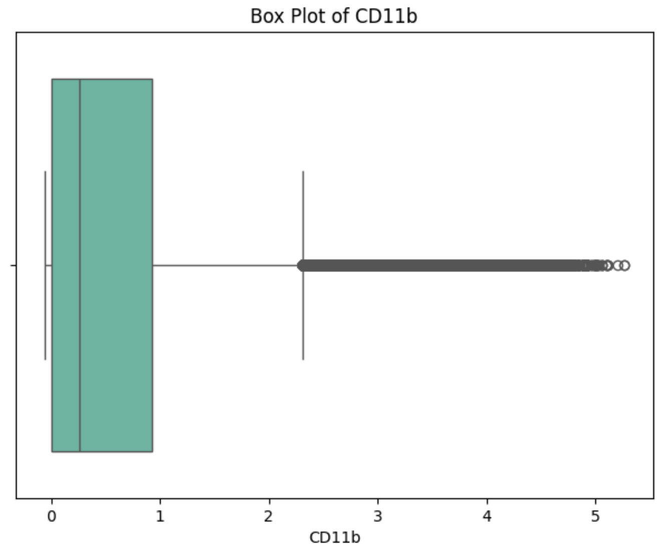


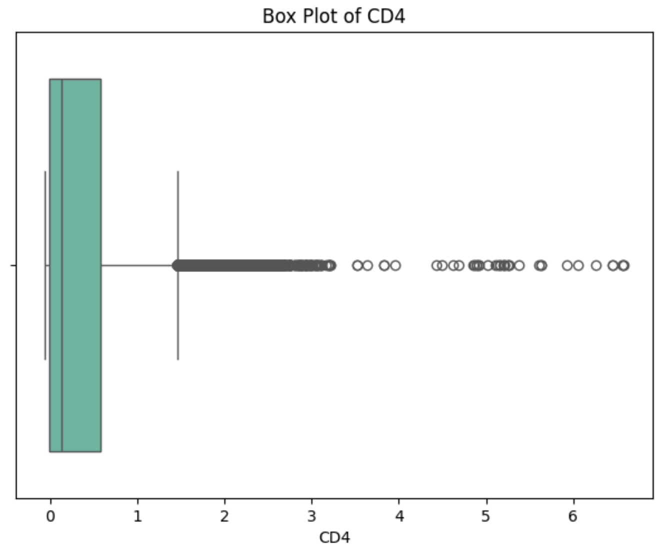

CD45RA
0 1 2 3 4 5 6 7

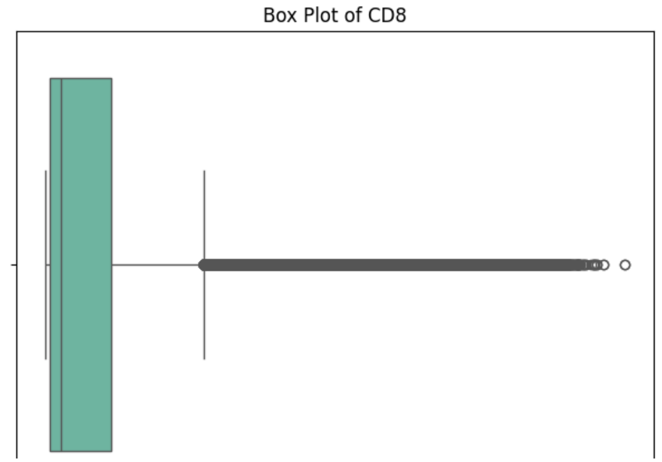

CD133
0 1 2 3 4 5

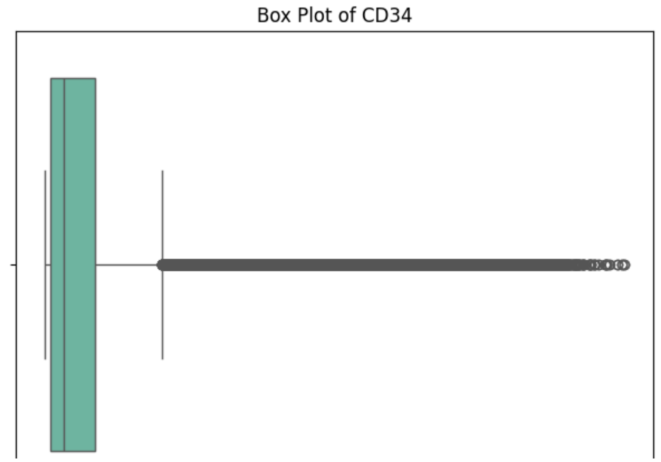

CD19
0 1 2 3 4 5

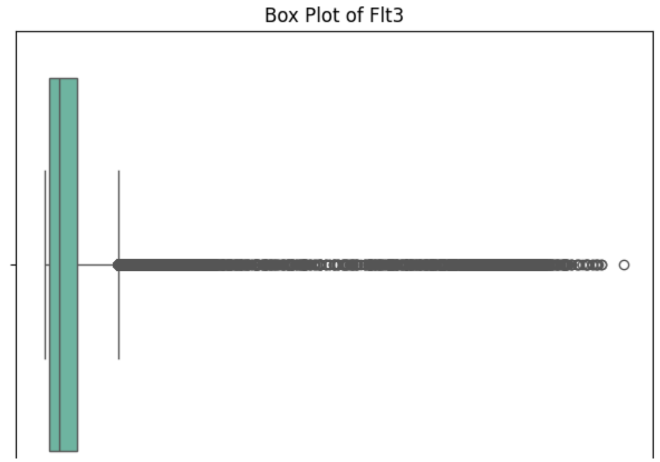

Box Plot of CD22
CD22
0 1 2 3 4 5

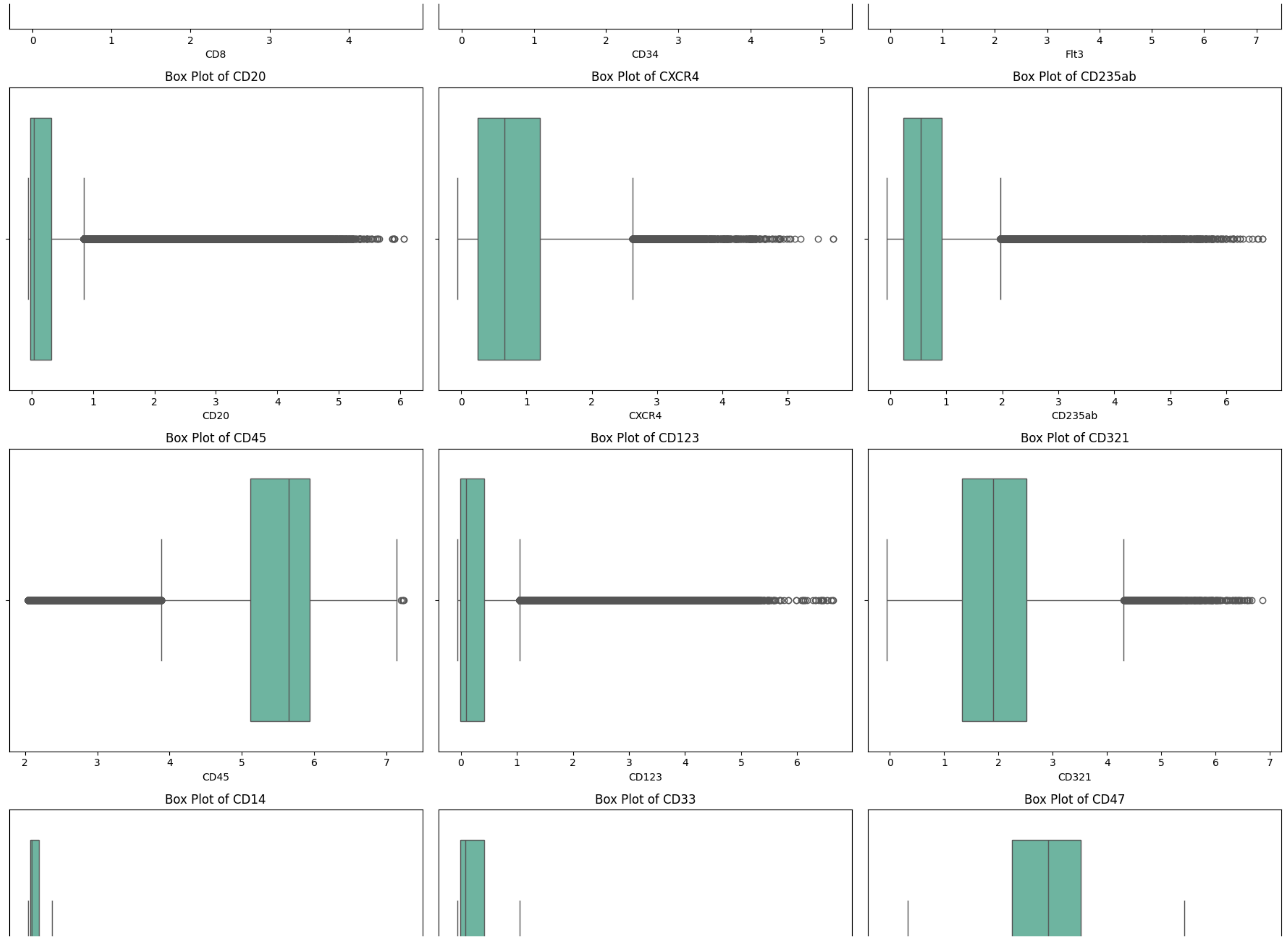

Box Plot of CD11b
CD11b
0 1 2 3 4 5

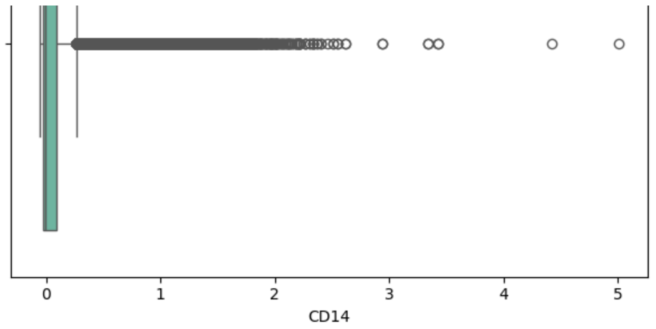

Box Plot of CD4
CD4
0 1 2 3 4 5 6

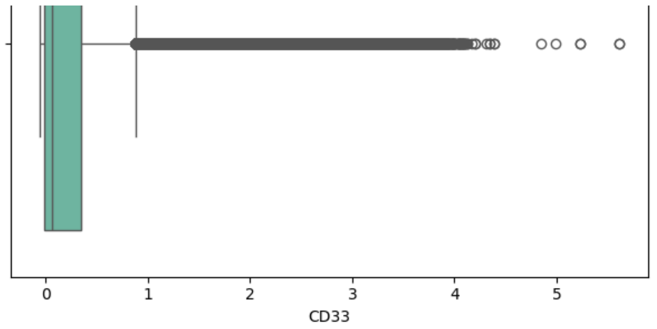

Box Plot of CD8

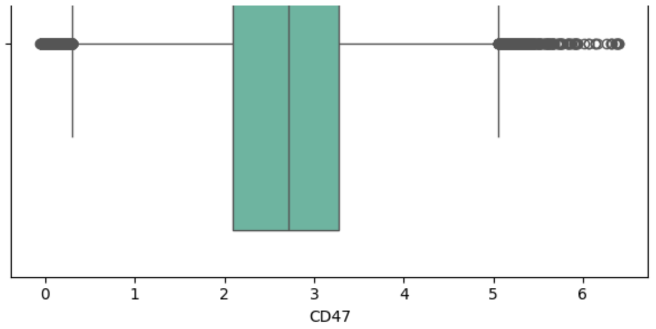

Box Plot of CD34


Box Plot of Flt3

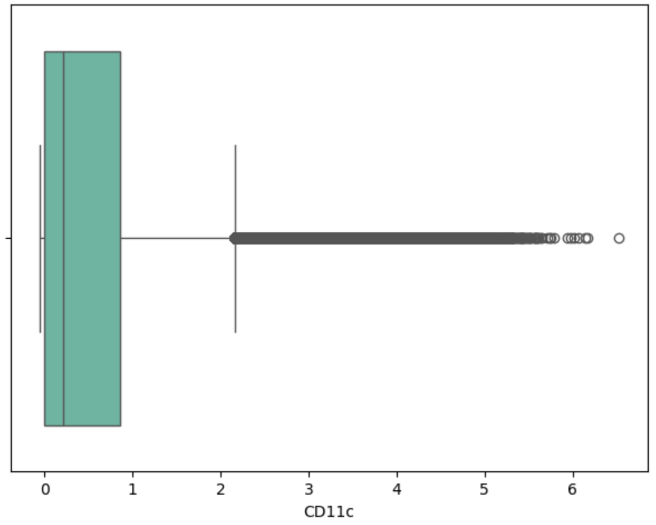

CD8
CD34
Flt3
Box Plot of CD20
Box Plot of CXCR4
Box Plot of CD235ab
CD20
CXCR4
CD235ab
Box Plot of CD45
Box Plot of CD123
Box Plot of CD321
CD45
CD123
CD321
Box Plot of CD14
Box Plot of CD33
Box Plot of CD47



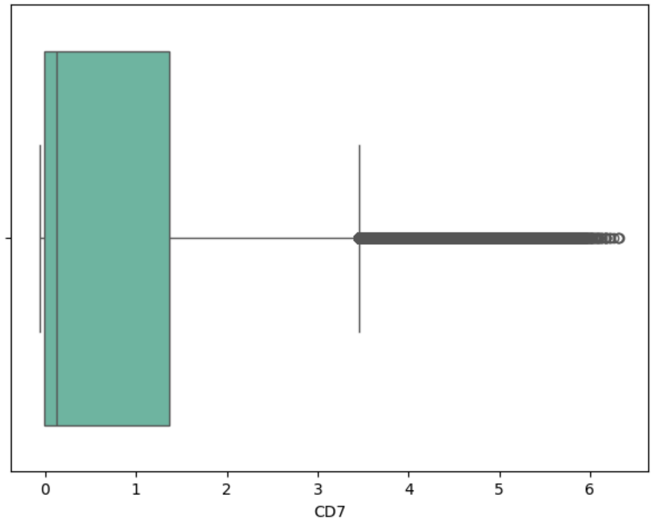




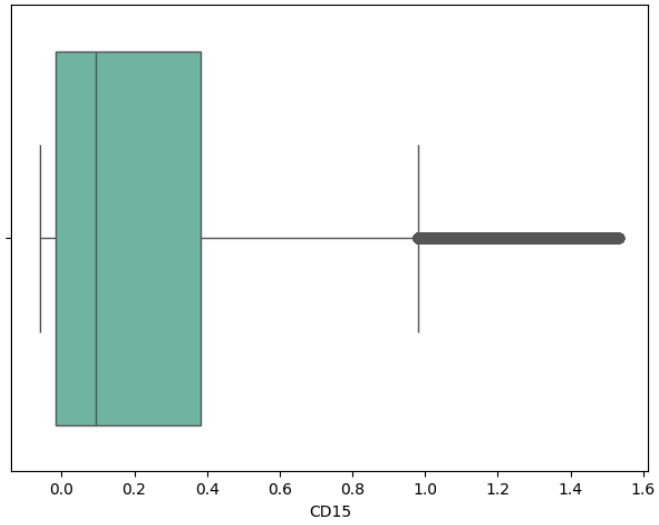
Box Plot of CD11c



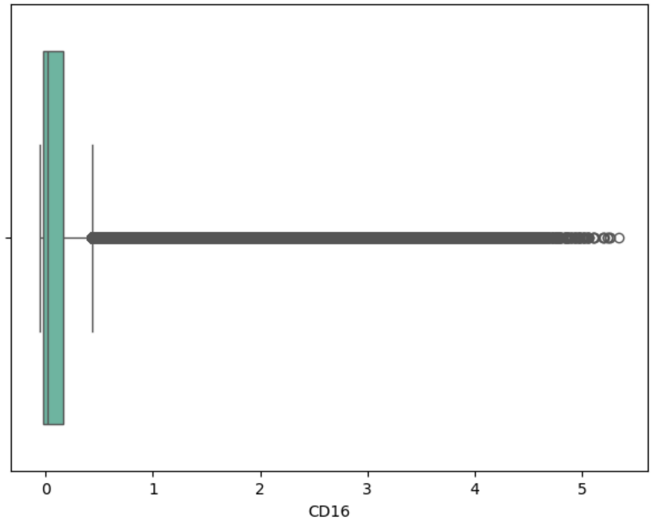
Box Plot of CD7



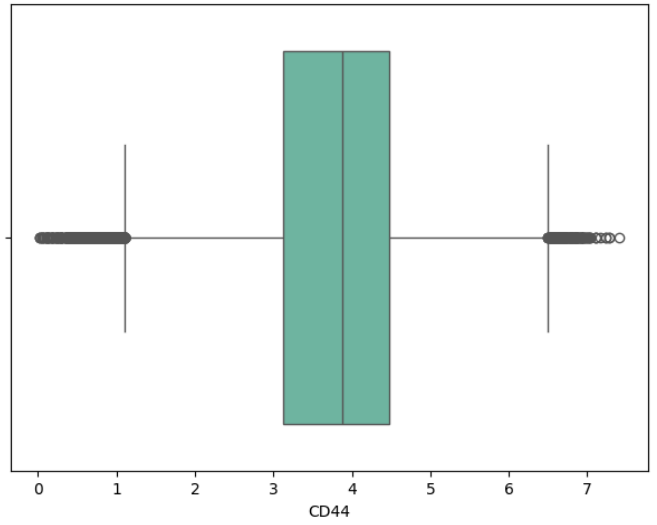
Box Plot of CD15



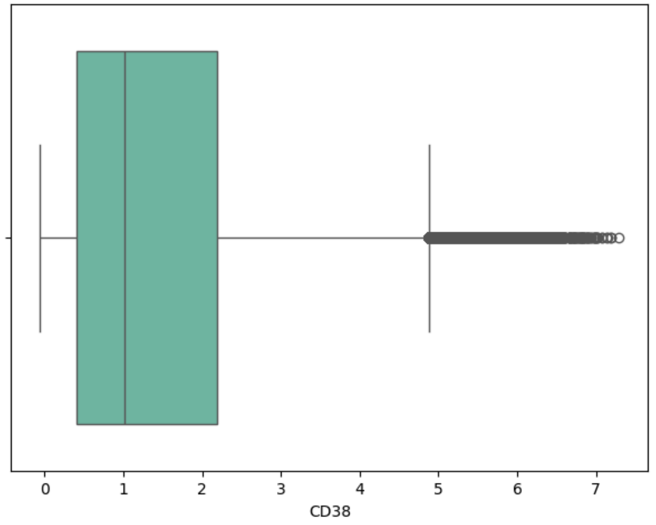
Box Plot of CD16

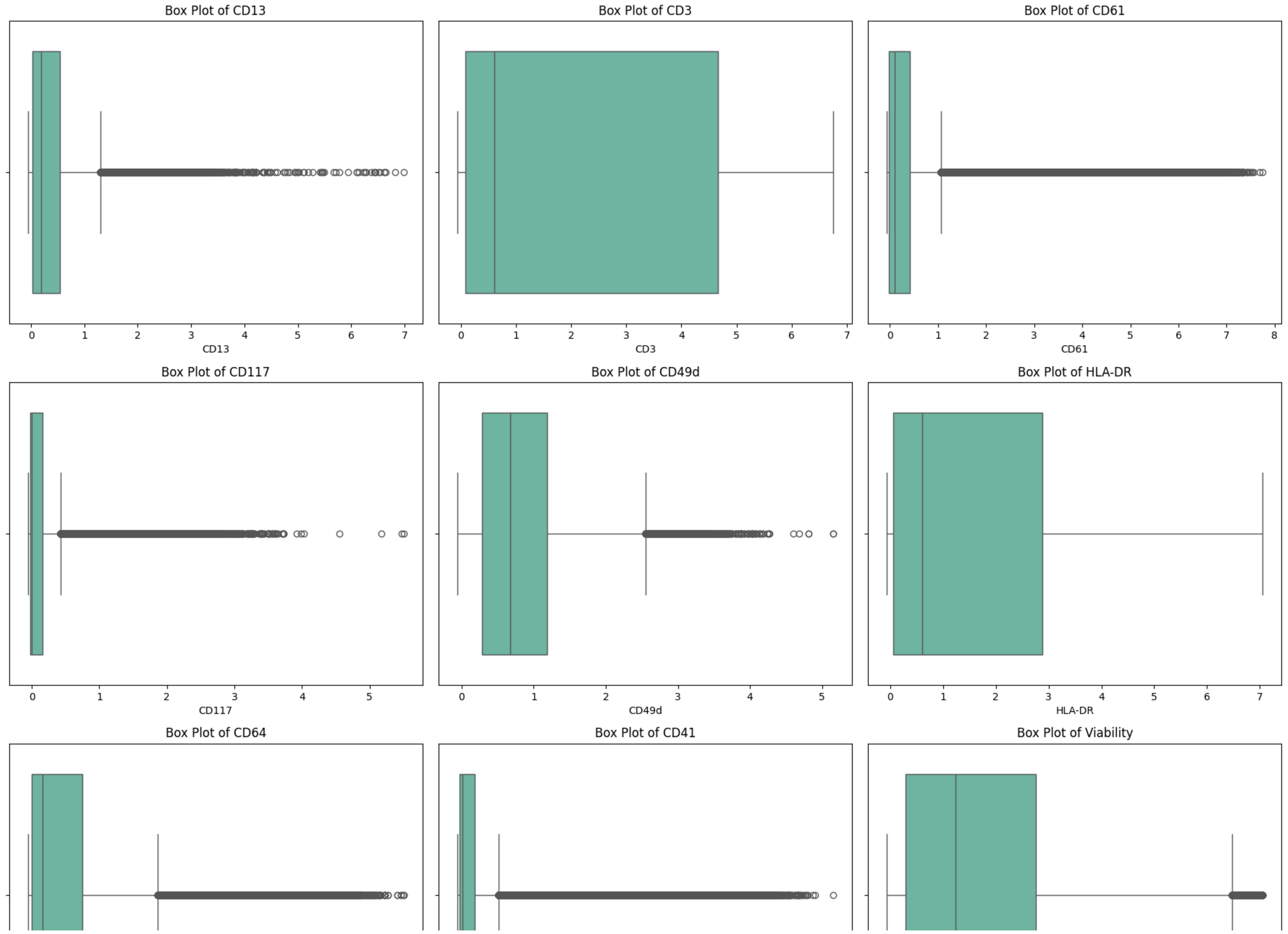


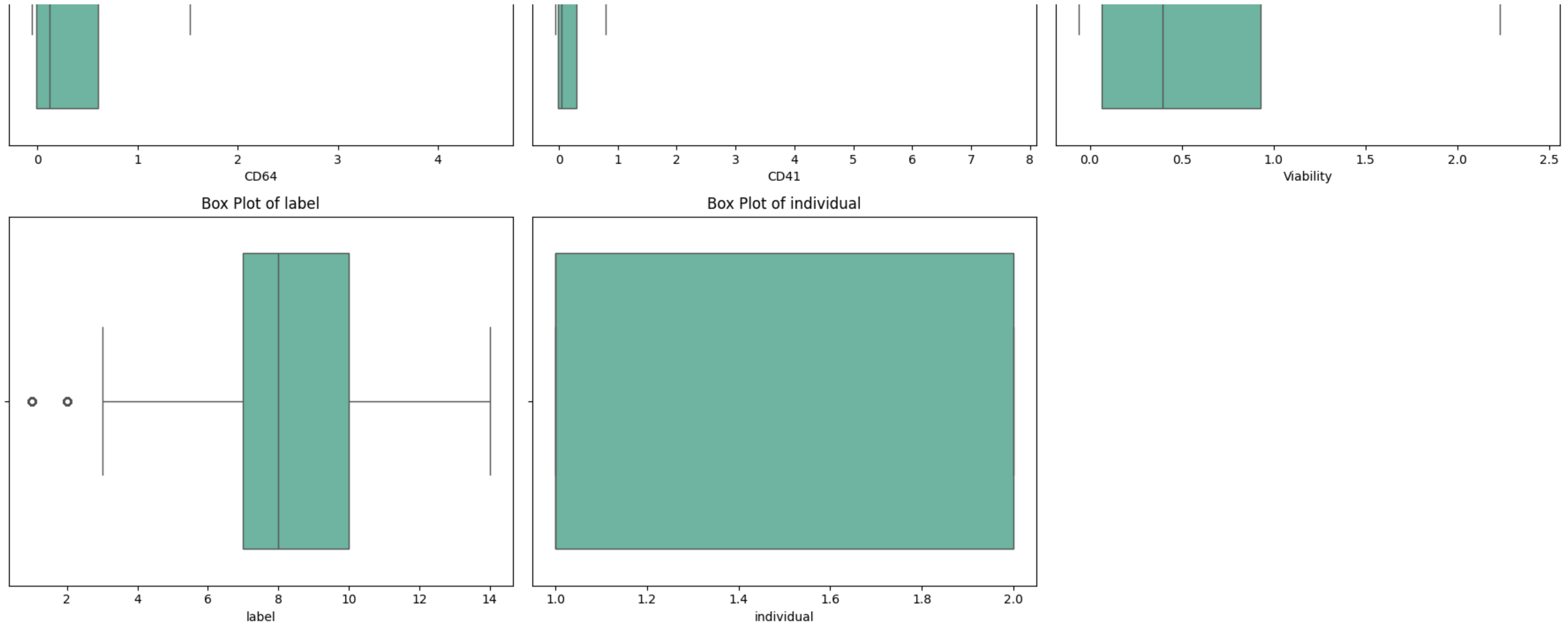
Box Plot of CD44



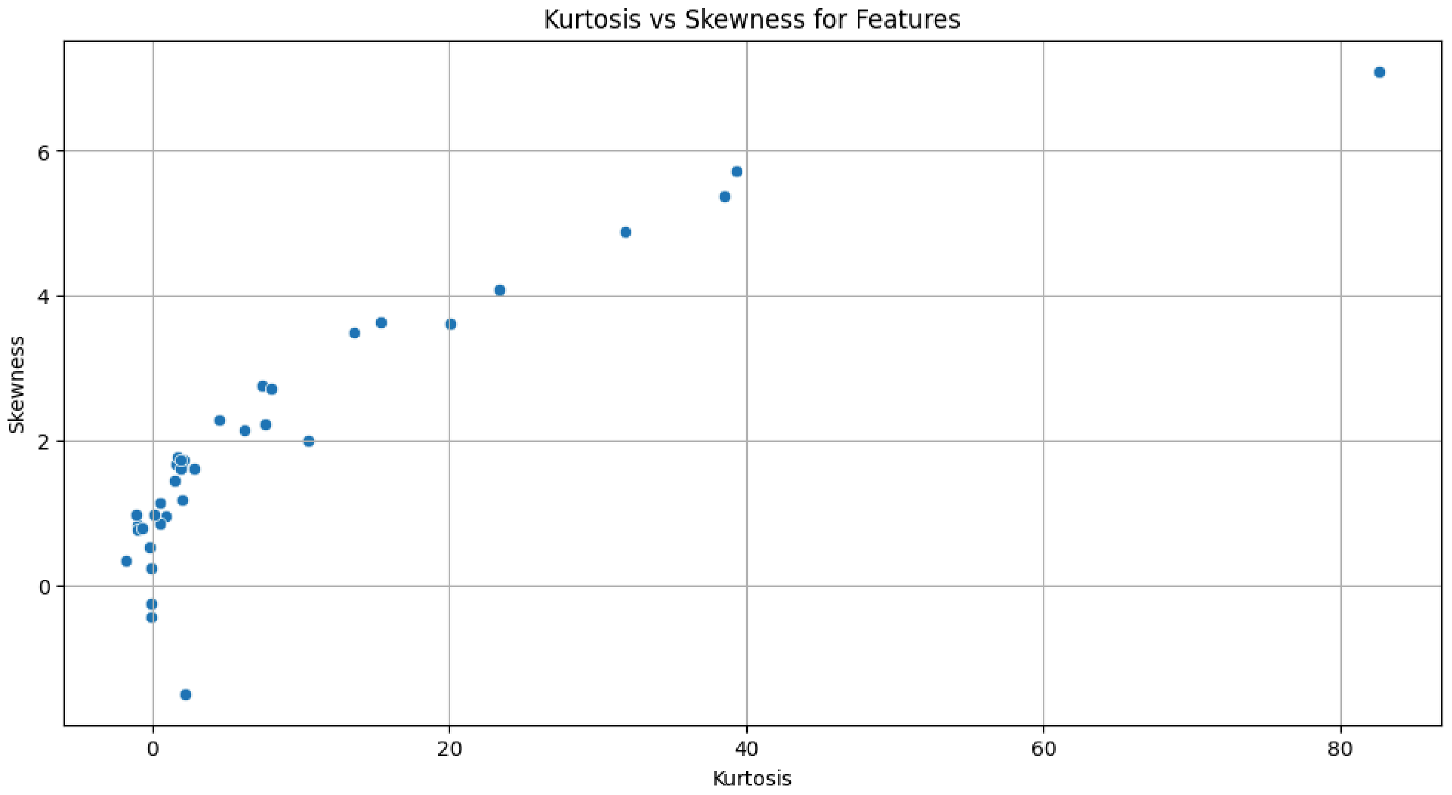
Box Plot of CD38

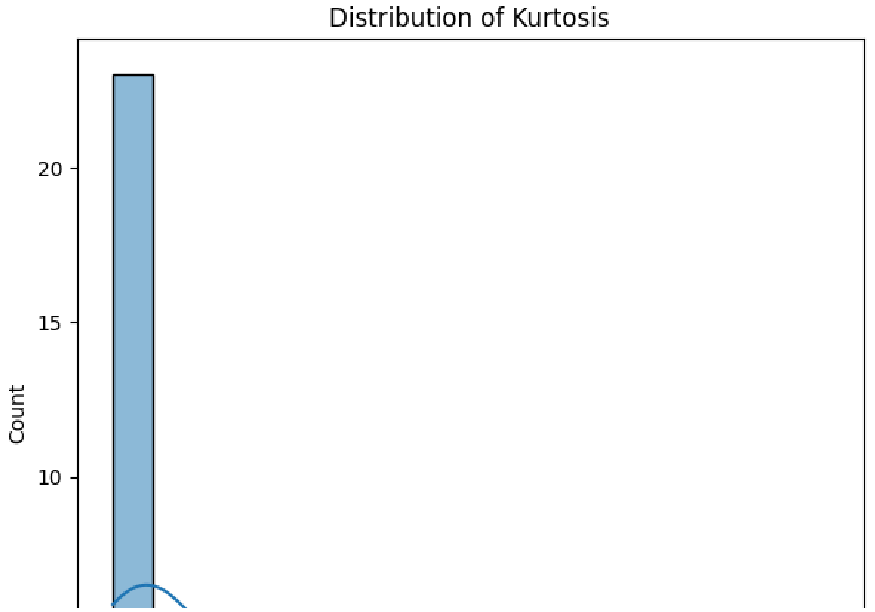


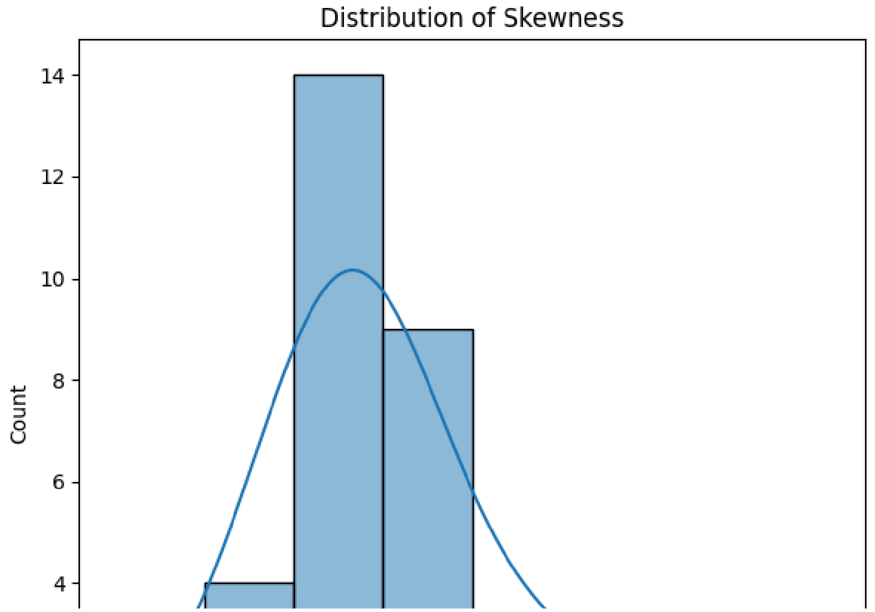

Box Plot of CD13
CD13
Box Plot of CD3
CD3
Box Plot of CD61
CD61
Box Plot of CD117
CD117
Box Plot of CD49d
CD49d
Box Plot of HLA-DR
HLA-DR
Box Plot of CD64
Box Plot of CD41
Box Plot of Viability

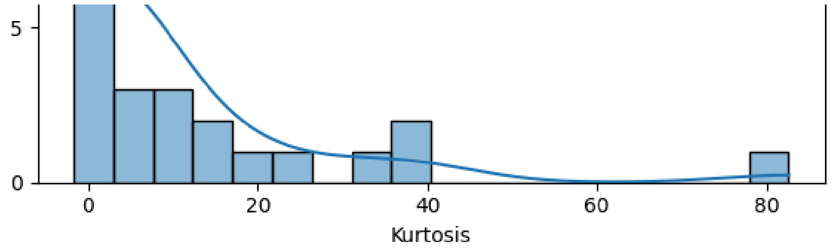

0
1
2
3
4
CD64
0
1
2
3
4
5
6
7
8
CD41
0.0
0.5
1.0
1.5
2.0
2.5
Viability
Box Plot of label
2
4
6
8
10
12
14
label
Box Plot of individual
1.0
1.2
1.4
1.6
1.8
2.0
individual

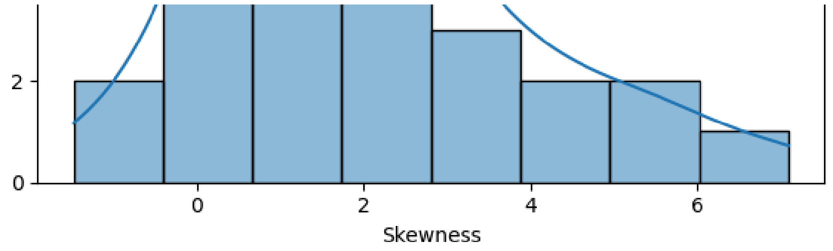
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import kurtosis, skew

columns_to_remove = ['Event', 'Time', 'event_number', 'file_number', 'label']
df_cleaned = df.drop(columns=columns_to_remove)

# Calculate kurtosis and skewness for each column
kurtosis_values = df_cleaned.apply(kurtosis)
skewness_values = df_cleaned.apply(skew)

# Create a DataFrame to store kurtosis and skewness
kurtosis_skewness_df = pd.DataFrame({'Kurtosis': kurtosis_values, 'Skewness': skewness_values})

# Plot Kurtosis and Skewness
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Kurtosis', y='Skewness', data=kurtosis_skewness_df)
plt.xlabel('Kurtosis')
plt.ylabel('Skewness')
plt.title('Kurtosis vs Skewness for Features')
plt.grid(True)
plt.show()

# You can also plot them separately
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
sns.histplot(kurtosis_values, kde=True)
plt.xlabel('Kurtosis')
plt.title('Distribution of Kurtosis')

plt.subplot(1, 2, 2)
sns.histplot(skewness_values, kde=True)
plt.xlabel('Skewness')
plt.title('Distribution of Skewness')

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


Kurtosis vs Skewness for Features
Skewness
Kurtosis
0
2
4
6
0
20
40
60
80


Distribution of Kurtosis
Count
10
15
20


Distribution of Skewness
Count
4
6
8
10
12
14


5
0
0
20
40
60
80
Kurtosis

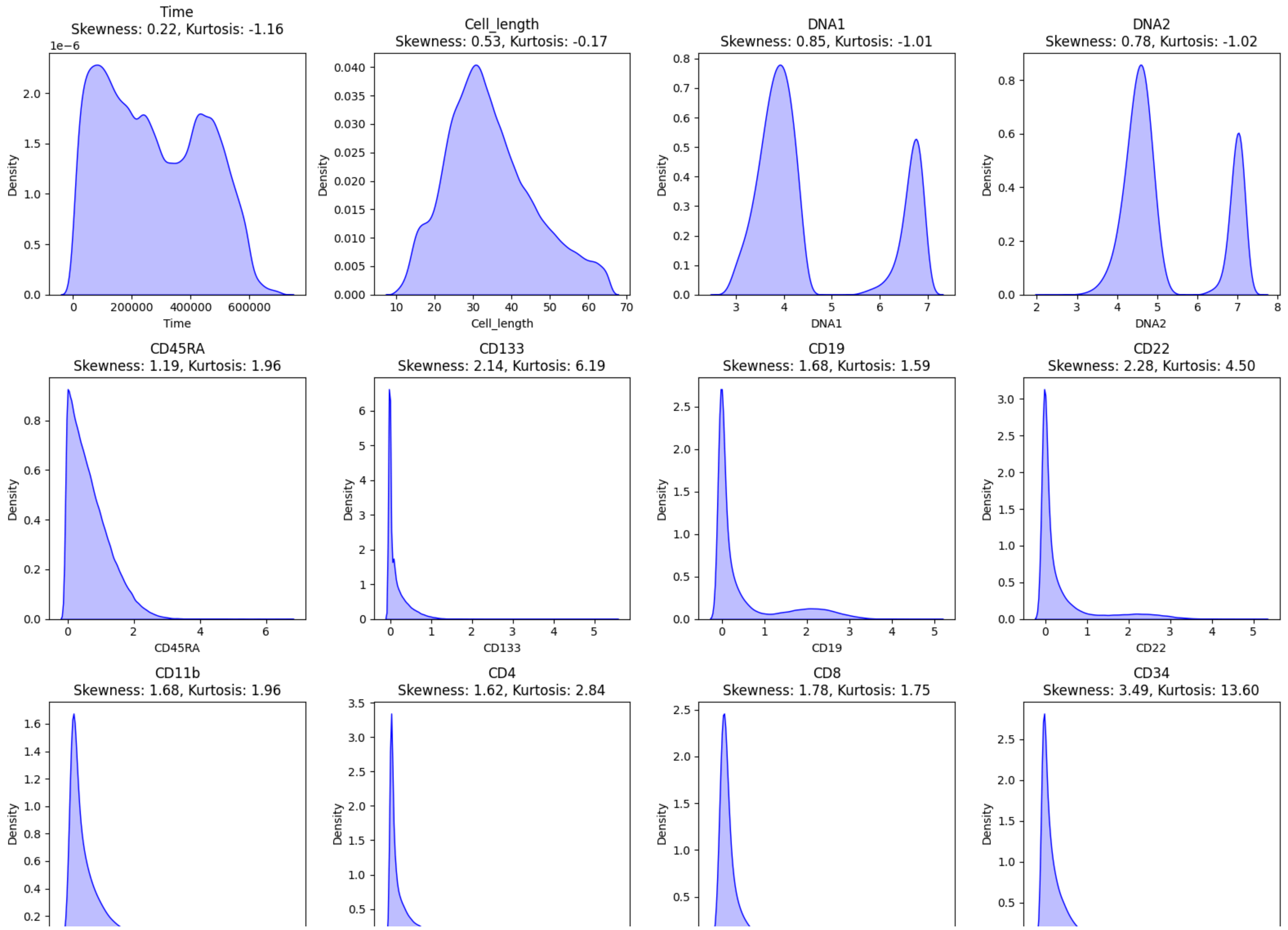

2
0
0
2
4
6
Skewness

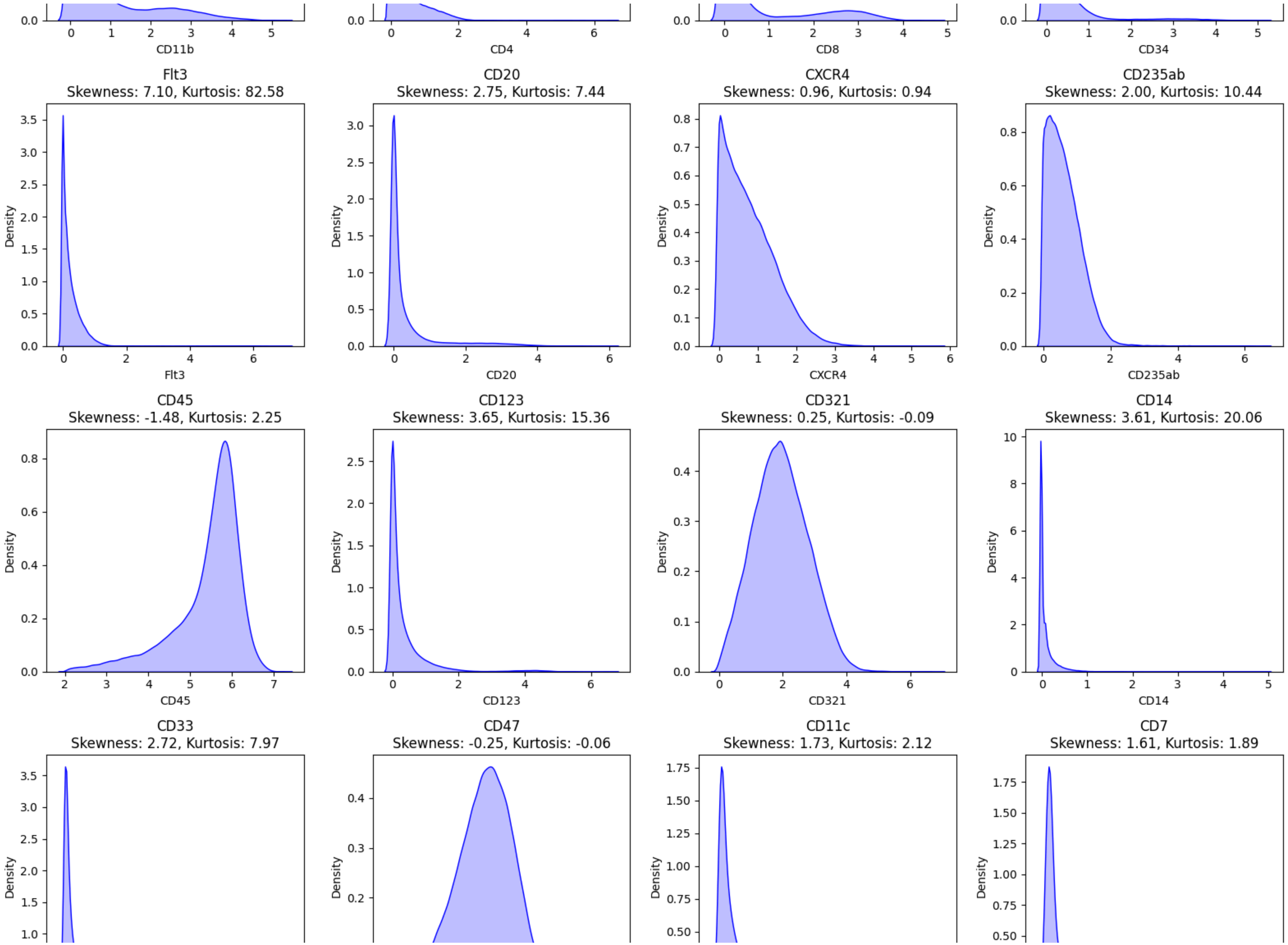
```
# prompt: Generate skewness and kurtosis for each and every feature

# Assuming df_cleaned is your DataFrame with relevant features

# Calculate skewness and kurtosis for each column
skewness_values = df_cleaned.skew()
kurtosis_values = df_cleaned.kurtosis()

# Create a DataFrame to store the results
skewness_kurtosis_df = pd.DataFrame({'Skewness': skewness_values, 'Kurtosis': kurtosis_values})

# Display the results
print(skewness_kurtosis_df)

# You can also plot the skewness and kurtosis if needed
# ... (plotting code as shown in the previous response)
```

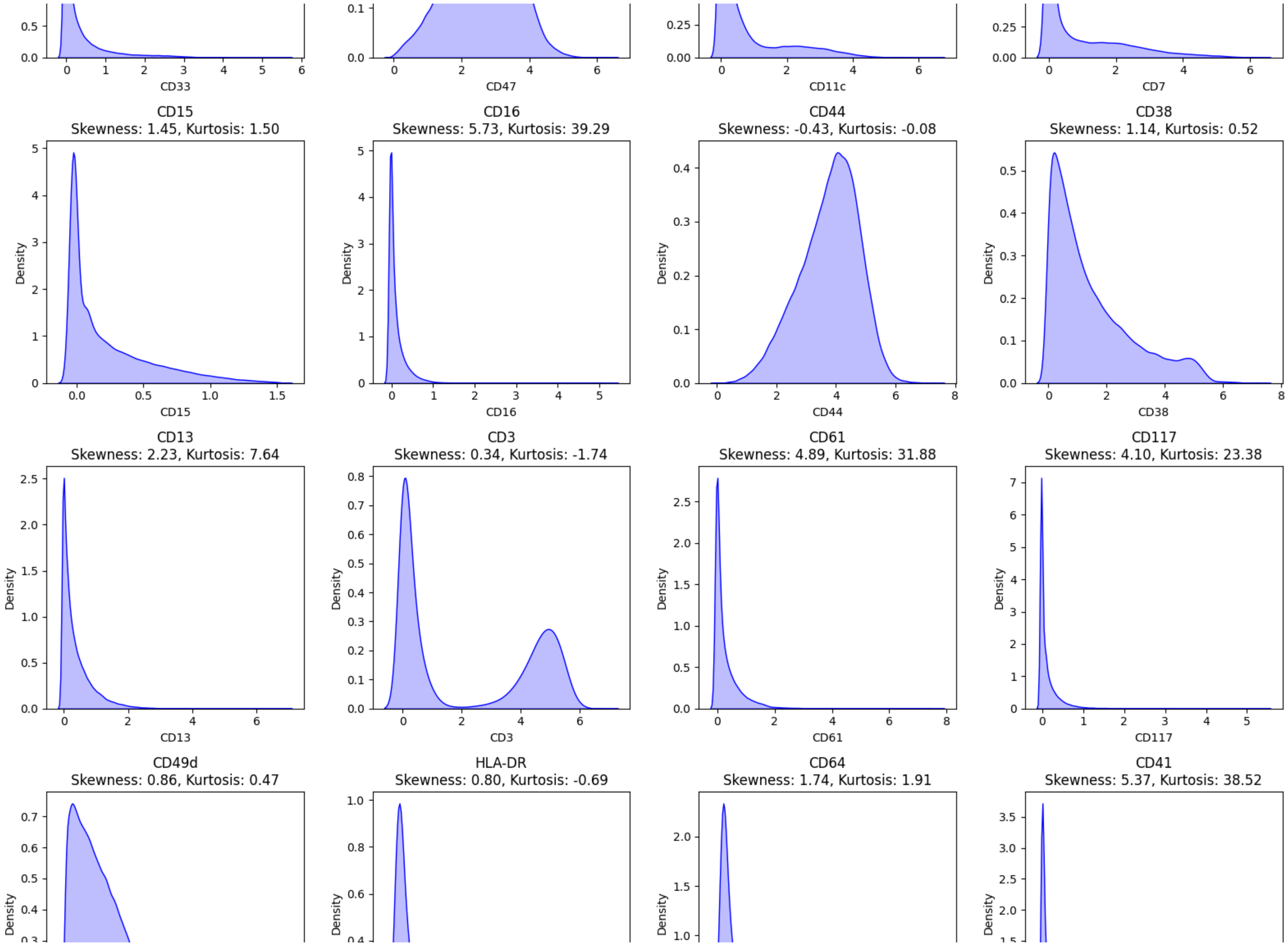
```
             Skewness   Kurtosis
Cell_length  0.527835  -0.165948
DNA1         0.845015  -1.005960
DNA2         0.779171  -1.024975
CD45RA       1.191601   1.964331
CD133        2.141965   6.190205
CD19         1.682619   1.590940
CD22         2.283194   4.500331
CD11b        1.679098   1.964555
CD4          1.622053   2.844337
CD8          1.775723   1.745831
CD34         3.492457  13.596694
Flt3         7.098191  82.585111
CD20         2.754715   7.435612
CXCR4        0.955347   0.936348
CD235ab      2.001491  10.440805
CD45        -1.484832   2.246835
CD123        3.648911  15.361529
CD321        0.247098  -0.085386
CD14         3.609026  20.062935
CD33         2.724993   7.967709
CD47        -0.250324  -0.056144
CD11c        1.733898   2.117218
CD7          1.606537   1.885173
CD15         1.445155   1.504438
CD16         5.733236  39.288511
CD44        -0.431592  -0.081187
CD38         1.141488   0.521222
CD13         2.234324   7.637731
CD3          0.342241  -1.735398
CD61         4.894735  31.878642
CD117        4.097531  23.375571
CD49d        0.856810   0.468151
```

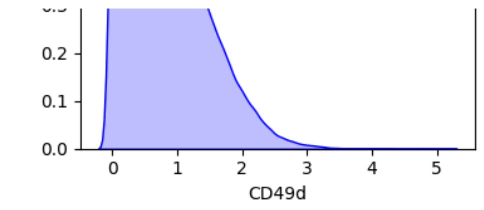
```
HLA-DR       0.795364  -0.690066
CD64         1.743742   1.910689
CD41         5.366344  38.521861
Viability    0.985422   0.156961
individual   0.982035  -1.035615
```

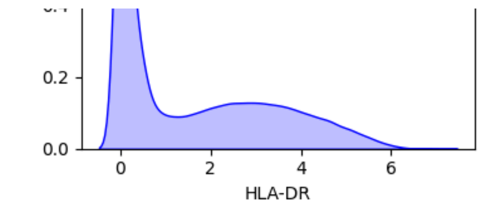
```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew, kurtosis
import numpy as np
excluded_columns = ['Event', 'event_time', 'file_number', 'event_number']
relevant_columns = [col for col in df.columns if col not in excluded_columns]

num_cols = 4
num_rows = (len(relevant_columns) + num_cols - 1) // num_cols

excluded_columns = ['Event', 'event_time', 'file_number', 'event_number']
relevant_columns = [col for col in df.columns if col not in excluded_columns]

num_rows = int(np.ceil(len(relevant_columns) / num_cols))

fig, axs = plt.subplots(num_rows, num_cols, figsize=(16, num_rows * 4))

axs = axs.flatten()

for i, col in enumerate(relevant_columns):
    sns.kdeplot(df[col], ax=axs[i], fill=True, color='blue')

    skewness_value = skew(df[col].dropna())
    kurtosis_value = kurtosis(df[col].dropna())

    axs[i].set_title(f'{col}\nSkewness: {skewness_value:.2f}, Kurtosis: {kurtosis_value:.2f}')

for j in range(i + 1, len(axs)):
    fig.delaxes(axs[j])

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

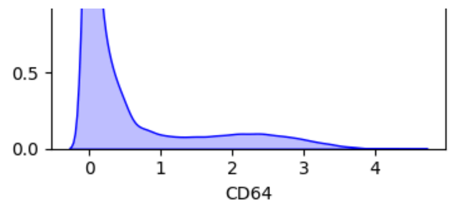

Time
Skewness: 0.22, Kurtosis: -1.16
Cell_length
Skewness: 0.53, Kurtosis: -0.17
DNA1
Skewness: 0.85, Kurtosis: -1.01
DNA2
Skewness: 0.78, Kurtosis: -1.02
CD45RA
Skewness: 1.19, Kurtosis: 1.96
CD133
Skewness: 2.14, Kurtosis: 6.19
CD19
Skewness: 1.68, Kurtosis: 1.59
CD22
Skewness: 2.28, Kurtosis: 4.50
CD11b
Skewness: 1.68, Kurtosis: 1.96
CD4
Skewness: 1.62, Kurtosis: 2.84
CD8
Skewness: 1.78, Kurtosis: 1.75
CD34
Skewness: 3.49, Kurtosis: 13.60
Density

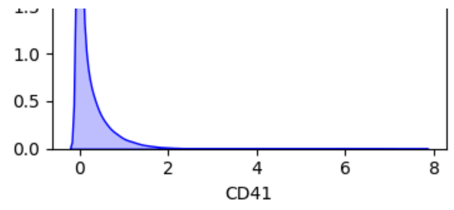

CD11b
CD4
CD8
CD34
Flt3
Skewness: 7.10, Kurtosis: 82.58
CD20
Skewness: 2.75, Kurtosis: 7.44
CXCR4
Skewness: 0.96, Kurtosis: 0.94
CD235ab
Skewness: 2.00, Kurtosis: 10.44
CD45
Skewness: -1.48, Kurtosis: 2.25
CD123
Skewness: 3.65, Kurtosis: 15.36
CD321
Skewness: 0.25, Kurtosis: -0.09
CD14
Skewness: 3.61, Kurtosis: 20.06
CD33
Skewness: 2.72, Kurtosis: 7.97
CD47
Skewness: -0.25, Kurtosis: -0.06
CD11c
Skewness: 1.73, Kurtosis: 2.12
CD7
Skewness: 1.61, Kurtosis: 1.89
Density


CD33
CD47
CD11c
CD7
CD15
Skewness: 1.45, Kurtosis: 1.50
CD16
Skewness: 5.73, Kurtosis: 39.29
CD44
Skewness: -0.43, Kurtosis: -0.08
CD38
Skewness: 1.14, Kurtosis: 0.52
CD13
Skewness: 2.23, Kurtosis: 7.64
CD3
Skewness: 0.34, Kurtosis: -1.74
CD61
Skewness: 4.89, Kurtosis: 31.88
CD117
Skewness: 4.10, Kurtosis: 23.38
CD49d
Skewness: 0.86, Kurtosis: 0.47
HLA-DR
Skewness: 0.80, Kurtosis: -0.69
CD64
Skewness: 1.74, Kurtosis: 1.91
CD41
Skewness: 5.37, Kurtosis: 38.52
Density

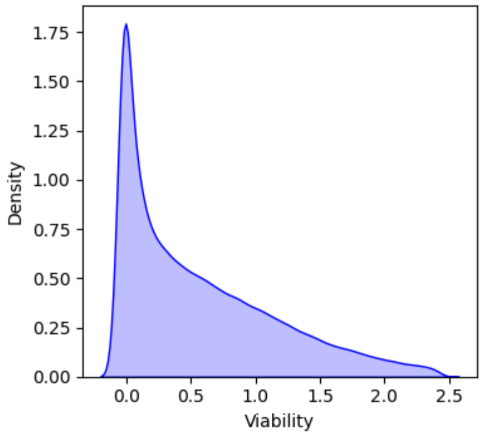




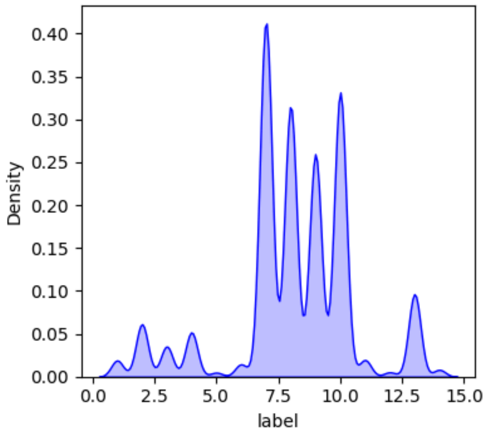




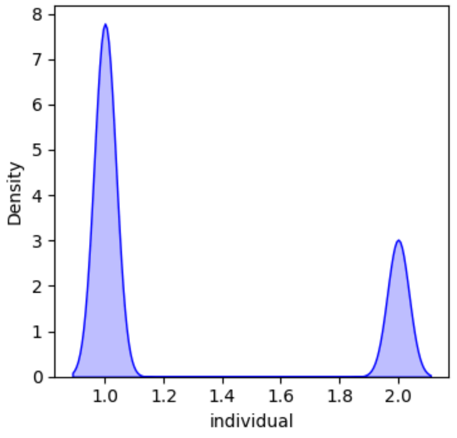
Viability
Skewness: 0.99, Kurtosis: 0.16



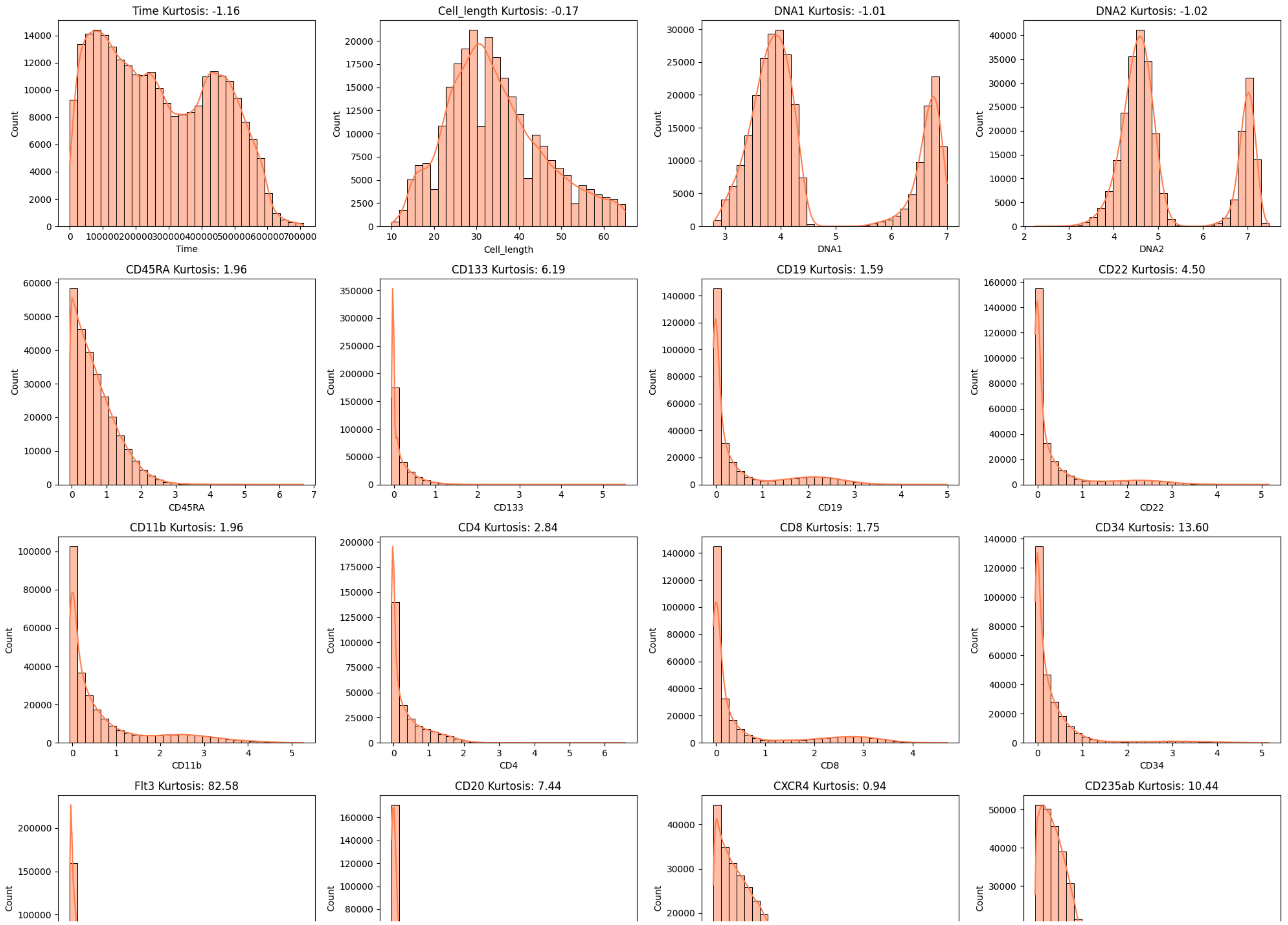
label
Skewness: -0.58, Kurtosis: 1.13

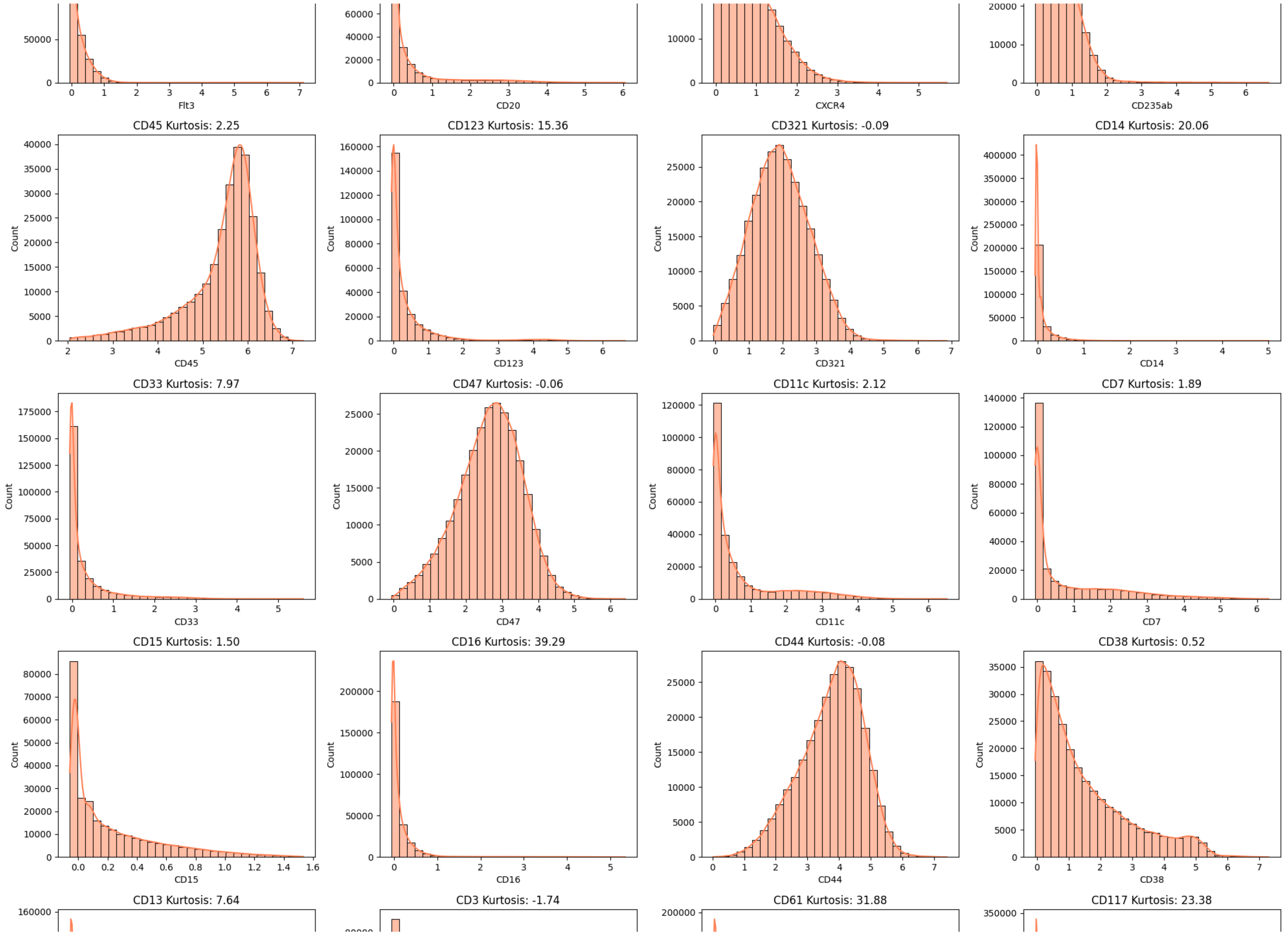


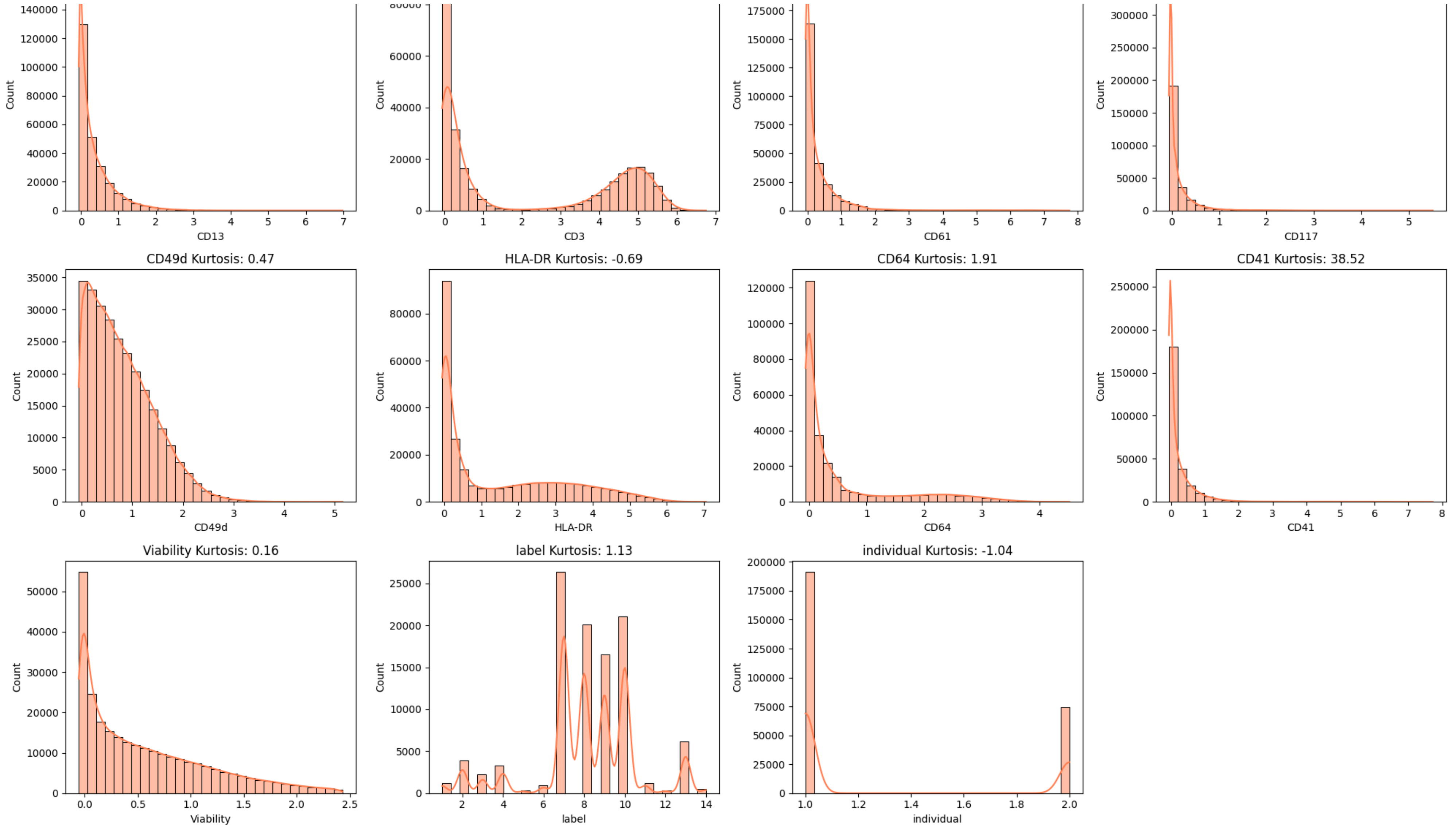
individual
Skewness: 0.98, Kurtosis: -1.04



```
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import kurtosis

excluded_columns = ['Event', 'event_time', 'file_number', 'event_number']
relevant_columns = [col for col in df.columns if col not in excluded_columns]

num_cols = 4
num_rows = (len(relevant_columns) + num_cols - 1) // num_cols

fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(20, num_rows * 4))
axes = axes.flatten()

# Set your desired color
plot_color = 'coral'  # Change this to any color you like

for i, col in enumerate(relevant_columns):
    sns.histplot(df[col], kde=True, ax=axes[i], bins=30, color=plot_color)
    axes[i].set_title(f'{col} Kurtosis: {kurtosis(df[col].dropna()):.2f}')

# Adjust layout and remove unused subplots
for j in range(i + 1, len(axes)):
    fig.delaxes(axes[j])

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

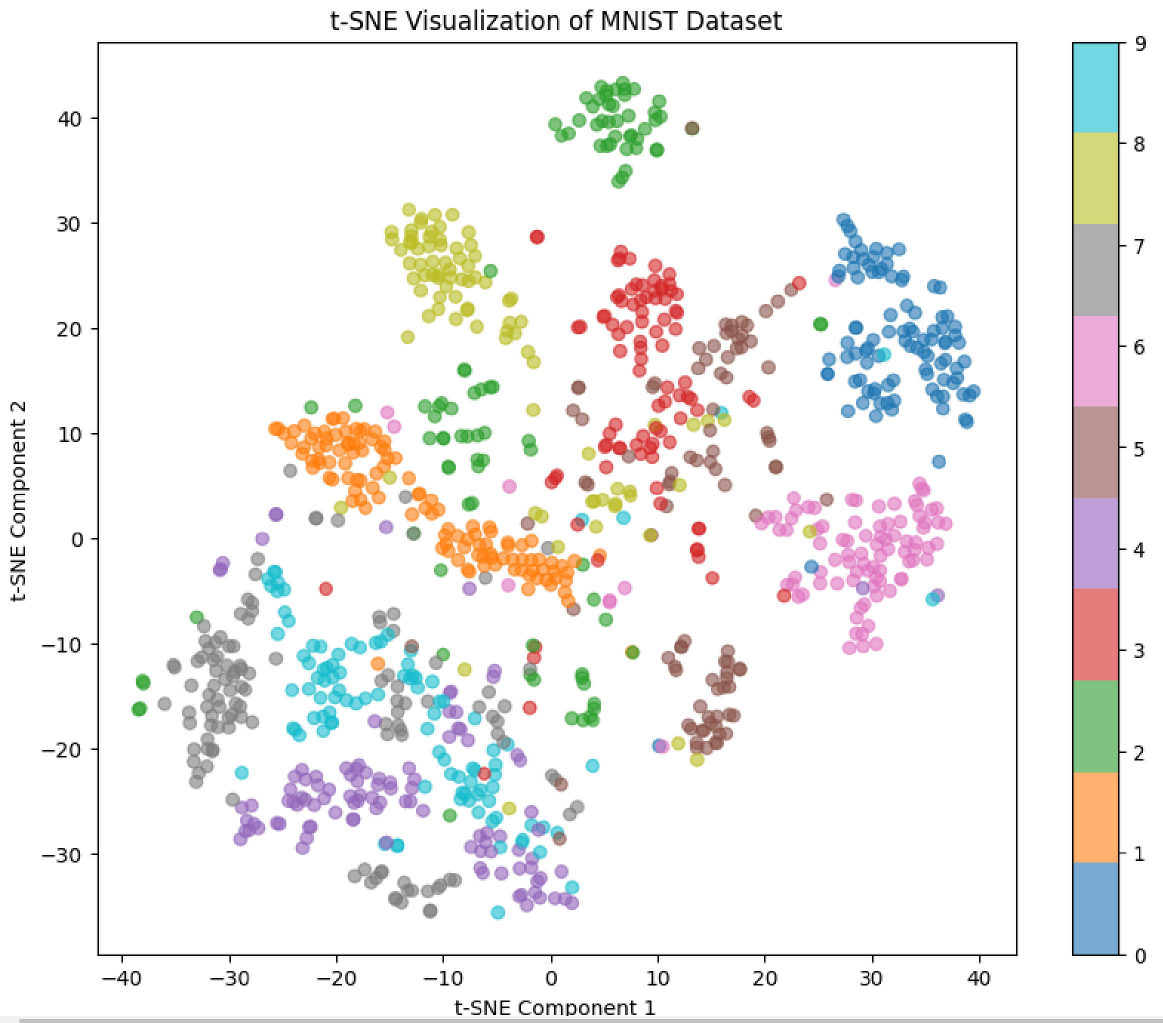

Time Kurtosis: -1.16
Time
Count
Cell_length Kurtosis: -0.17
Cell_length
DNA1 Kurtosis: -1.01
DNA1
DNA2 Kurtosis: -1.02
DNA2
CD45RA Kurtosis: 1.96
CD45RA
CD133 Kurtosis: 6.19
CD133
CD19 Kurtosis: 1.59
CD19
CD22 Kurtosis: 4.50
CD22
CD11b Kurtosis: 1.96
CD11b
CD4 Kurtosis: 2.84
CD4
CD8 Kurtosis: 1.75
CD8
CD34 Kurtosis: 13.60
CD34
Flt3 Kurtosis: 82.58
CD20 Kurtosis: 7.44
CXCR4 Kurtosis: 0.94
CD235ab Kurtosis: 10.44


Flt3
CD20
CXCR4
CD235ab
CD45 Kurtosis: 2.25
CD45
CD123 Kurtosis: 15.36
CD123
CD321 Kurtosis: -0.09
CD321
CD14 Kurtosis: 20.06
CD14
CD33 Kurtosis: 7.97
CD33
CD47 Kurtosis: -0.06
CD47
CD11c Kurtosis: 2.12
CD11c
CD7 Kurtosis: 1.89
CD7
CD15 Kurtosis: 1.50
CD15
CD16 Kurtosis: 39.29
CD16
CD44 Kurtosis: -0.08
CD44
CD38 Kurtosis: 0.52
CD38
CD13 Kurtosis: 7.64
CD3 Kurtosis: -1.74
CD61 Kurtosis: 31.88
CD117 Kurtosis: 23.38
Count


Count
CD13
CD3
CD61
CD117
CD49d Kurtosis: 0.47
HLA-DR Kurtosis: -0.69
CD64 Kurtosis: 1.91
CD41 Kurtosis: 38.52
CD49d
HLA-DR
CD64
CD41
Viability Kurtosis: 0.16
label Kurtosis: 1.13
individual Kurtosis: -1.04
Viability
label
individual

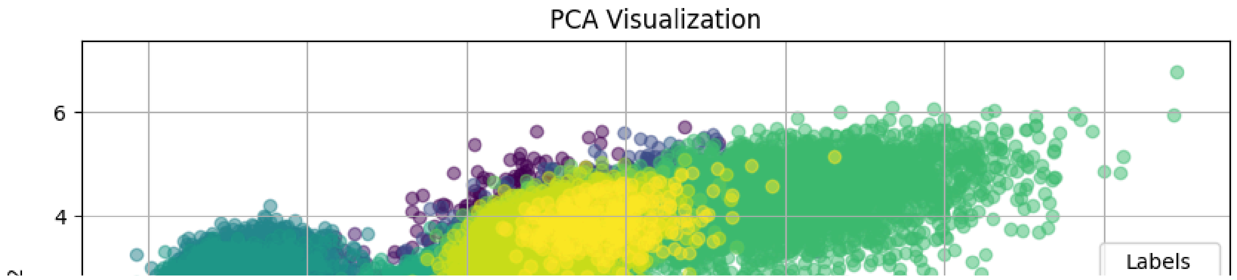
```
import tensorflow as tf
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
import numpy as np

(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()
train_images = train_images.astype('float32') / 255.0
test_images = test_images.astype('float32') / 255.0

n_samples = 1000
train_images_flat = train_images[:n_samples].reshape(n_samples, -1)
train_labels_subset = train_labels[:n_samples]

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
train_images_tsne = tsne.fit_transform(train_images_flat)

# plot the results
plt.figure(figsize=(10, 8))
scatter = plt.scatter(train_images_tsne[:, 0], train_images_tsne[:, 1], c=train_labels_subset, cmap='tab10', alpha=0.6)
plt.colorbar(scatter, ticks=range(10))
plt.title('t-SNE Visualization of MNIST Dataset')
plt.xlabel('t-SNE Component 1')
plt.ylabel('t-SNE Component 2')
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
```



```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Assume df is your DataFrame and relevant_columns contains the features to be standardized

# Standardize the data
scaler = StandardScaler()
scaled_data = scaler.fit_transform(df[relevant_columns].dropna())
```

```
# Apply PCA
pca = PCA(n_components=2)  # Reduce to 2 dimensions for visualization
pca_results = pca.fit_transform(scaled_data)

# Create a DataFrame for PCA results
pca_df = pd.DataFrame(pca_results, columns=['Principal Component 1', 'Principal Component 2'])

# Add labels or categories if available
pca_df['Label'] = df['label'].dropna().values  # Replace with your label column

# Plotting the PCA results with different colors
plt.figure(figsize=(10, 6))
unique_labels = pca_df['Label'].unique()  # Get unique labels for coloring
colors = plt.cm.viridis(np.linspace(0, 1, len(unique_labels)))  # Choose a colormap

for i, label in enumerate(unique_labels):
    subset = pca_df[pca_df['Label'] == label]
    plt.scatter(subset['Principal Component 1'], subset['Principal Component 2'],
                alpha=0.5, color=colors[i], label=label)

plt.title('PCA Visualization')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.grid()
plt.legend(title='Labels')
plt.show()

# Optional: Explained variance
explained_variance = pca.explained_variance_ratio_
print(f'Explained variance by component: {explained_variance}')
```


PCA Visualization
6
4
Labels