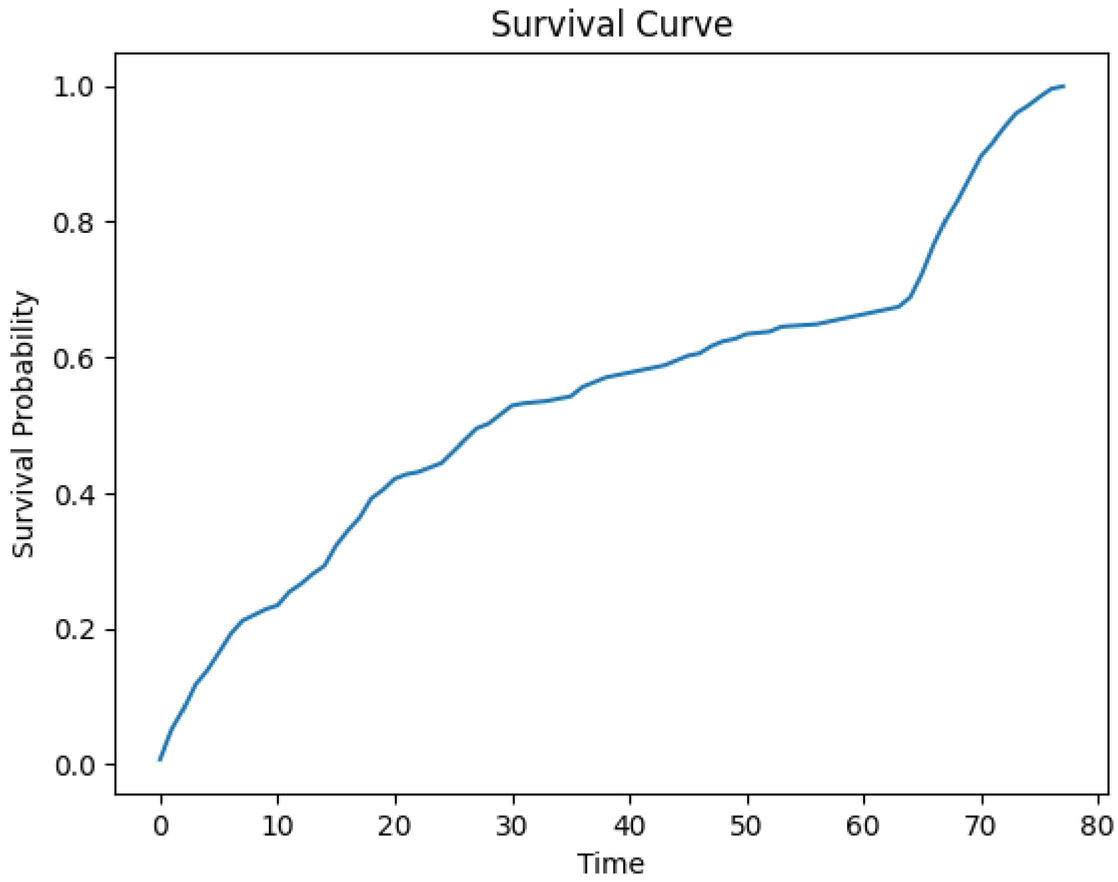
In [1]:

```
import pandas as pd
import numpy as np
from lifelines import CoxPHFitter
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

data = pd.read_excel('data1.xlsx') # Load the data from the .xlsx file

# Preprocess the data: Drop any rows with missing values in the columns of interest
data = data.dropna(subset=['Months', 'DEATH', 'AGE', 'SEX', 'CompositeStage', 'LNInvolment', 'Comorbidity', 'FamiliyHistoryOfC

# Handle missing values in other columns
imputer = SimpleImputer(strategy='median')
data[['DEATH', 'AGE', 'CompositeStage', 'LNInvolment', 'Comorbidity']] = imputer.fit_transform(data[['DEATH', 'AGE', 'Composit

# Standardize the covariates
scaler = StandardScaler()
data[['DEATH', 'AGE', 'CompositeStage', 'LNInvolment', 'Comorbidity']] = scaler.fit_transform(data[['DEATH', 'AGE', 'Composite

# Create a new DataFrame with the required columns for the Buckley-James estimator
buckley_james_data = data[['Months', 'DEATH', 'AGE', 'SEX', 'CompositeStage', 'LNInvolment', 'Comorbidity', 'FamiliyHistoryOfC

# Fit the Buckley-James model with custom options
cph = CoxPHFitter(penalizer=0.1)  # Set the penalizer parameter to control overfitting
cph.fit(buckley_james_data, 'Months', 'DEATH', show_progress=True)  # Set the step_size parameter to control the convergence s

# Print the estimated coefficients (summary)
print(cph.summary)

# Access other properties of the fitted model (e.g., hazard ratios, p-values)
# For example, to get the hazard ratios:
print(cph.hazard_ratios_)

# Calculate AIC and BIC
n = len(buckley_james_data)
llf = cph.log_likelihood_
k = cph.params_.shape[0]
aic = -2 * llf + 2 * k
```

```
bic = -2 * llf + k * np.log(n)

# Print AIC and BIC
print("AIC:", aic)
print("BIC:", bic)

# Make predictions using the fitted model
# For example, to predict the survival probability at a specific time point for a new patient:
new_patient_data = pd.DataFrame({'AGE': [90], 'SEX': [1], 'CompositeStage': [2], 'LNInvolment': [1], 'Comorbidity': [0], 'Fami
partial_hazard = cph.predict_partial_hazard(new_patient_data)
survival_prob = 1 - cph.baseline_survival_

# Plot the survival curve
plt.plot(cph.baseline_survival_.index, survival_prob.values)
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.title('Survival Curve')
plt.show()

# Perform other analyses or visualizations as needed
```

```
Iteration 1: norm_delta = 0.66384, step_size = 0.9500, log_lik = -1663.17959, newton_decrement = 46.04648, seconds_since_start
= 0.0
Iteration 2: norm_delta = 0.03630, step_size = 0.9500, log_lik = -1620.53093, newton_decrement = 0.19362, seconds_since_start =
0.0
Iteration 3: norm_delta = 0.00176, step_size = 0.9500, log_lik = -1620.33817, newton_decrement = 0.00043, seconds_since_start =
0.0
Iteration 4: norm_delta = 0.00000, step_size = 1.0000, log_lik = -1620.33774, newton_decrement = 0.00000, seconds_since_start =
0.0
Convergence success after 4 iterations.
                            coef  exp(coef)  se(coef)  coef lower 95%  \
covariate
AGE                     0.019975   1.020175  0.055896       -0.089580
SEX                     0.027013   1.027381  0.106745       -0.182203
CompositeStage          0.531571   1.701603  0.061434        0.411162
LNInvolment            -0.275748   0.759004  0.053051       -0.379725
Comorbidity            -0.034023   0.966549  0.054884       -0.141594
FamiliyHistoryOfCancer  0.003465   1.003471  0.156806       -0.303870

                        coef upper 95%  exp(coef) lower 95%  \
covariate
AGE                           0.129529             0.914315
SEX                           0.236229             0.833432
CompositeStage                0.651980             1.508570
LNInvolment                  -0.171771             0.684049
Comorbidity                   0.073548             0.867974
FamiliyHistoryOfCancer        0.310800             0.737957

                        exp(coef) upper 95%  cmp to         z             p  \
covariate
AGE                                1.138292     0.0  0.357349  7.208303e-01
SEX                                1.266464     0.0  0.253064  8.002191e-01
CompositeStage                     1.919337     0.0  8.652682  5.030319e-18
LNInvolment                        0.842172     0.0 -5.197833  2.016254e-07
Comorbidity                        1.076320     0.0 -0.619903  5.353217e-01
FamiliyHistoryOfCancer             1.364517     0.0  0.022100  9.823684e-01

                         -log2(p)
covariate
AGE                      0.472268
SEX                      0.321533
```

```
CompositeStage           57.464056
LNInvolment              22.241820
Comorbidity               0.901522
FamiliyHistoryOfCancer    0.025664
AIC: 3252.6754729222544
BIC: 3275.70185560525
```



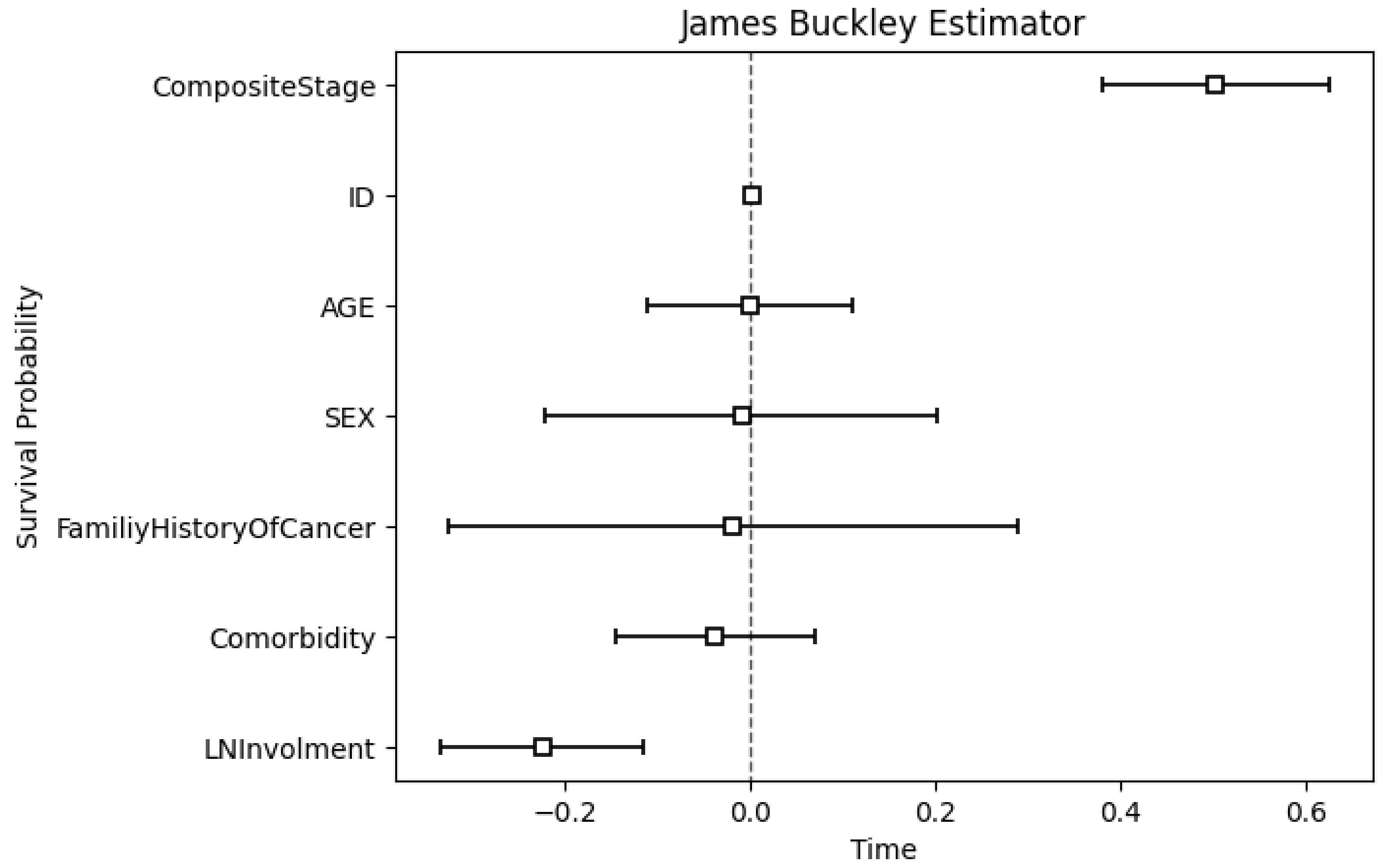
In [12]:

```
import pandas as pd
import numpy as np
from lifelines import CoxPHFitter
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
```

```
# Load the data from the .xlsx file
data = pd.read_excel('data1.xlsx')

# Preprocess the data: Drop any rows with missing values in the columns of interest
data = data.dropna(subset=['Months', 'DEATH', 'AGE', 'SEX', 'CompositeStage', 'LNInvolment', 'Comorbidity', 'FamiliyHistoryOfC

# Handle missing values in other columns
imputer = SimpleImputer(strategy='median')
data[['DEATH', 'AGE', 'CompositeStage', 'LNInvolment', 'Comorbidity']] = imputer.fit_transform(data[['DEATH', 'AGE', 'Composit

# Standardize the covariates
scaler = StandardScaler()
data[['DEATH', 'AGE', 'CompositeStage', 'LNInvolment', 'Comorbidity']] = scaler.fit_transform(data[['DEATH', 'AGE', 'Composite

# Fit the James Buckley model
cph = CoxPHFitter(penalizer=0.1)
cph.fit(data, duration_col='Months', event_col='DEATH', show_progress=True)
print(cph.summary)
# Plot the James Buckley estimator survival curve
cph.plot()
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.title('James Buckley Estimator')
plt.show()
```

```
Iteration 1: norm_delta = 0.62460, step_size = 0.9500, log_lik = -1663.17959, newton_decrement = 49.21114, seconds_since_start
= 0.0
Iteration 2: norm_delta = 0.04476, step_size = 0.9500, log_lik = -1616.79249, newton_decrement = 0.24023, seconds_since_start =
0.0
Iteration 3: norm_delta = 0.00197, step_size = 0.9500, log_lik = -1616.55380, newton_decrement = 0.00046, seconds_since_start =
0.0
Iteration 4: norm_delta = 0.00000, step_size = 1.0000, log_lik = -1616.55334, newton_decrement = 0.00000, seconds_since_start =
0.0
Convergence success after 4 iterations.
                            coef  exp(coef)  se(coef)  coef lower 95%  \
covariate
ID                      0.001450   1.001451  0.000523        0.000424
AGE                    -0.001136   0.998865  0.056504       -0.111881
SEX                    -0.009393   0.990651  0.107755       -0.220589
CompositeStage          0.502219   1.652384  0.062538        0.379647
LNInvolment            -0.224674   0.798776  0.056159       -0.334745
Comorbidity            -0.037540   0.963156  0.054946       -0.145231
FamiliyHistoryOfCancer -0.018981   0.981198  0.156943       -0.326584

                        coef upper 95%  exp(coef) lower 95%  \
covariate
ID                            0.002476             1.000424
AGE                           0.109610             0.894150
SEX                           0.201804             0.802046
CompositeStage                0.624792             1.461769
LNInvolment                  -0.114604             0.715521
Comorbidity                   0.070152             0.864822
FamiliyHistoryOfCancer        0.288622             0.721384

                        exp(coef) upper 95%  cmp to         z             p  \
covariate
ID                                 1.002479     0.0  2.770679  5.593948e-03
AGE                                1.115842     0.0 -0.020102  9.839624e-01
SEX                                1.223608     0.0 -0.087165  9.305400e-01
CompositeStage                     1.867857     0.0  8.030624  9.697825e-16
LNInvolment                        0.891719     0.0 -4.000652  6.316819e-05
Comorbidity                        1.072671     0.0 -0.683213  4.944721e-01
FamiliyHistoryOfCancer             1.334587     0.0 -0.120940  9.037386e-01

                         -log2(p)
```

```
covariate
ID                        7.481917
AGE                       0.023325
SEX                       0.103860
CompositeStage           49.873188
LNInvolment              13.950442
Comorbidity               1.016039
FamiliyHistoryOfCancer    0.146023
```




```
In [ ]: # Choose a single variable for univariate analysis
variable_of_interest = 'AGE'

# Fit the Cox proportional hazards model with the chosen variable
cph_univariate = CoxPHFitter(penalizer=0.1)
```

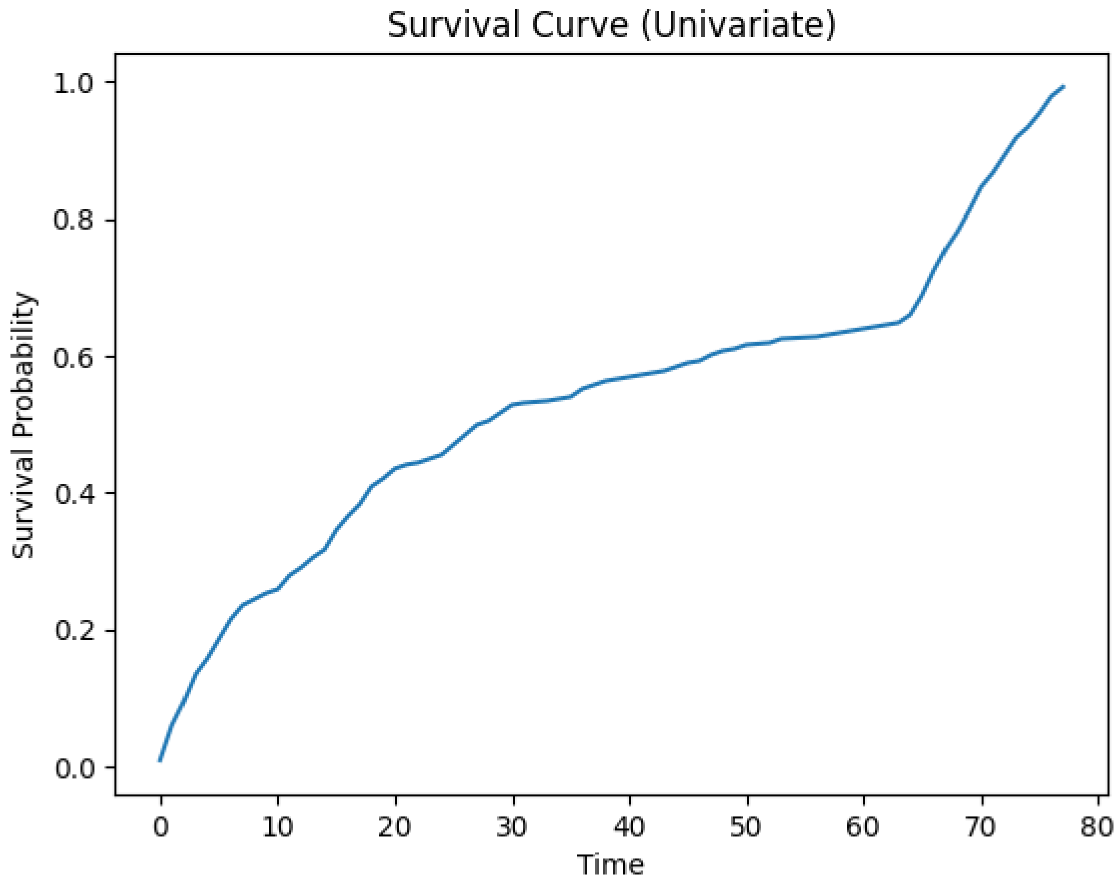
```
cph_univariate.fit(buckley_james_data[[variable_of_interest, 'Months', 'DEATH']], 'Months', 'DEATH', show_progress=True)

# Print the estimated coefficients (summary)
print(cph_univariate.summary)

# Access other properties of the fitted model (e.g., hazard ratios, p-values)
# For example, to get the hazard ratios:
print(cph_univariate.hazard_ratios_)

# Calculate AIC and BIC
n_univariate = len(buckley_james_data)
llf_univariate = cph_univariate.log_likelihood_
k_univariate = cph_univariate.params_.shape[0]
aic_univariate = -2 * llf_univariate + 2 * k_univariate
bic_univariate = -2 * llf_univariate + k_univariate * np.log(n_univariate)

# Print AIC and BIC
print("AIC (univariate):", aic_univariate)
print("BIC (univariate):", bic_univariate)

# Make predictions using the univariate model
# For example, to predict the survival probability at a specific time point for a new patient:
new_patient_data_univariate = pd.DataFrame({variable_of_interest: [90], 'Months': [12], 'DEATH': [0]})
partial_hazard_univariate = cph_univariate.predict_partial_hazard(new_patient_data_univariate)
survival_prob_univariate = 1 - cph_univariate.baseline_survival_

# Plot the survival curve for the univariate model
plt.plot(cph_univariate.baseline_survival_.index, survival_prob_univariate.values)
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.title('Survival Curve (Univariate)')
plt.show()
```

In [14]:

```
# Choose a single variable for univariate analysis
variable_of_interest = 'AGE'

# Fit the Cox proportional hazards model with the chosen variable
cph_univariate = CoxPHFitter(penalizer=0.1)
cph_univariate.fit(buckley_james_data[[variable_of_interest, 'Months', 'DEATH']], 'Months', 'DEATH', show_progress=True)
```

```
# Print the estimated coefficients (summary)
print(cph_univariate.summary)

# Access other properties of the fitted model (e.g., hazard ratios, p-values)
# For example, to get the hazard ratios:
print(cph_univariate.hazard_ratios_)

# Calculate AIC and BIC
n_univariate = len(buckley_james_data)
llf_univariate = cph_univariate.log_likelihood_
k_univariate = cph_univariate.params_.shape[0]
aic_univariate = -2 * llf_univariate + 2 * k_univariate
bic_univariate = -2 * llf_univariate + k_univariate * np.log(n_univariate)



# Make predictions using the univariate model
# For example, to predict the survival probability at a specific time point for a new patient:
new_patient_data_univariate = pd.DataFrame({variable_of_interest: [90], 'Months': [12], 'DEATH': [0]})
partial_hazard_univariate = cph_univariate.predict_partial_hazard(new_patient_data_univariate)
survival_prob_univariate = 1 - cph_univariate.baseline_survival_

# Plot the survival curve for the univariate model
plt.plot(cph_univariate.baseline_survival_.index, survival_prob_univariate.values)
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.title('Survival Curve (Univariate)')
plt.show()

# Print AIC and BIC
print("AIC (univariate):", aic_univariate)
print("BIC (univariate):", bic_univariate)
```

```
Iteration 1: norm_delta = 0.01879, step_size = 0.9500, log_lik = -1663.17959, newton_decrement = 0.06380, seconds_since_start =
0.0
Iteration 2: norm_delta = 0.00085, step_size = 0.9500, log_lik = -1663.11614, newton_decrement = 0.00013, seconds_since_start =
0.0
Iteration 3: norm_delta = 0.00004, step_size = 0.9500, log_lik = -1663.11600, newton_decrement = 0.00000, seconds_since_start =
0.1
Iteration 4: norm_delta = 0.00000, step_size = 1.0000, log_lik = -1663.11600, newton_decrement = 0.00000, seconds_since_start =
0.1
Convergence success after 4 iterations.
               coef  exp(coef)  se(coef)  coef lower 95%  coef upper 95%  \
covariate
AGE       -0.018672   0.981501  0.052274       -0.121127        0.083782

           exp(coef) lower 95%  exp(coef) upper 95%  cmp to         z  \
covariate
AGE                   0.885921             1.087392     0.0 -0.357205

                  p  -log2(p)
covariate
AGE        0.720938  0.472052
covariate
AGE    0.981501
Name: exp(coef), dtype: float64
```




```
AIC (univariate): 3328.2320093107332
BIC (univariate): 3332.0697397578992
```

In [ ]: