Exercise #3 Survival Analysis Exercise

DA6233 Kilger

Name: ***Dan Schumacher***

Imports

In [21]:
```
import pandas as pd
from lifelines import KaplanMeierFitter
import matplotlib.pyplot as plt
from IPython.display import IFrame
import seaborn as sns
from lifelines.statistics import logrank_test, multivariate_logrank_test
```

Load and Clean Data

In [22]:
```
# read in
short_df = pd.read_csv('./data/short_survival_data.csv')

# drop column that was accidentally added with all NAs
short_df.drop(short_df.columns[-1], axis=1, inplace=True)

# rename Status column to represent the positive class
short_df.rename(columns={short_df.columns[-1]:'dead', 'survival time in days': 'days'}

# let's look
short_df.head(16)
```

Out[22]:

	subject	days	dead
0	A	1	1
1	B	4	1
2	C	5	0
3	D	5	1
4	E	7	1
5	F	10	0
6	G	10	0
7	H	12	1
8	I	14	1
9	J	16	0

In [23]:
```
# Second database needed
long_df = pd.read_csv('./data/long_events_data.csv')

#rename group column
```

```
long_df.rename(columns={'Group#':'Group'}, inplace=True)

# Let's look
long_df.head()
```

Out[23]:

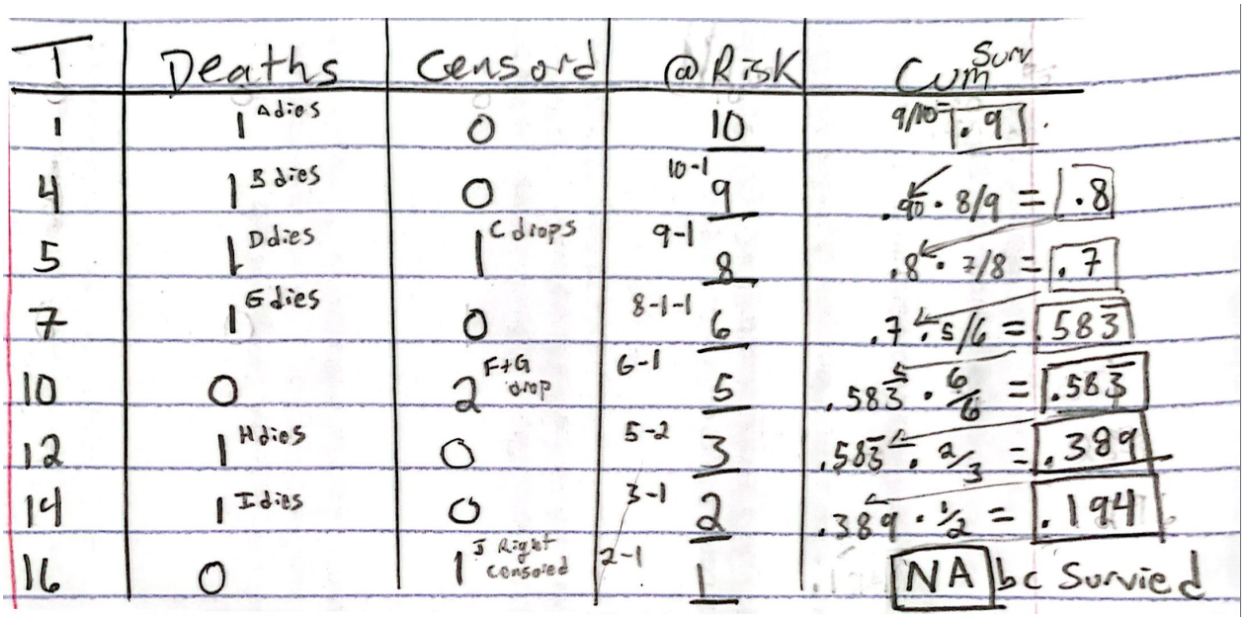
	Group	time	event
0	1	681	0
1	1	602	0
2	1	996	0
3	1	1162	0
4	1	833	0

1. You are analyzing a 16 day study of patients who have Krusty the Clown disease. Your data includes a subject ID, their survival time and a status. A status of 0 means they are alive while a status of 1 means they died of the disease.

a. Construct a survival table similar to the one in the lecture for the data. You will show your work within the table – if there is some division like 1/5 then show that like 1/5=.2 and if there is some multiplication then show the numbers being multiplied and the result e.g. .1 * 4 =0.4

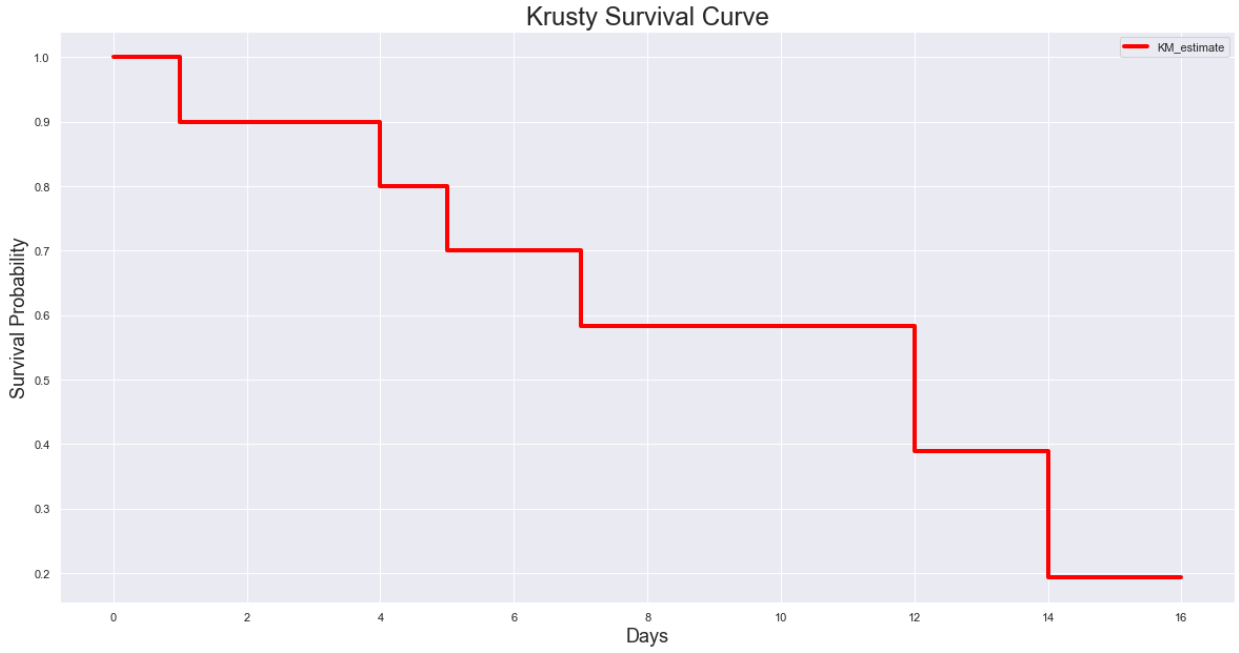
In [25]:
```
from IPython.display import Image

# Display an image with its filename
Image(filename='./data/surv_an.png')
```

Out[25]:



b. Plot a survival curve for the probabilities you generate in part a.

In [20]:
```
#this looks better
sns.set(style="darkgrid")
plt.figure(figsize=(20,10))

# Instantiate
kmf = KaplanMeierFitter()

# Fit
kmf.fit(durations=short_df['days'], event_observed=short_df['dead'])

# Plot
kmf.plot_survival_function(ci_show=False, color='red', lw=4)
plt.title('Krusty Survival Curve', fontsize=24)
plt.xlabel('Days', fontsize=18)
plt.ylabel('Survival Probability',fontsize=18)
plt.show();
```

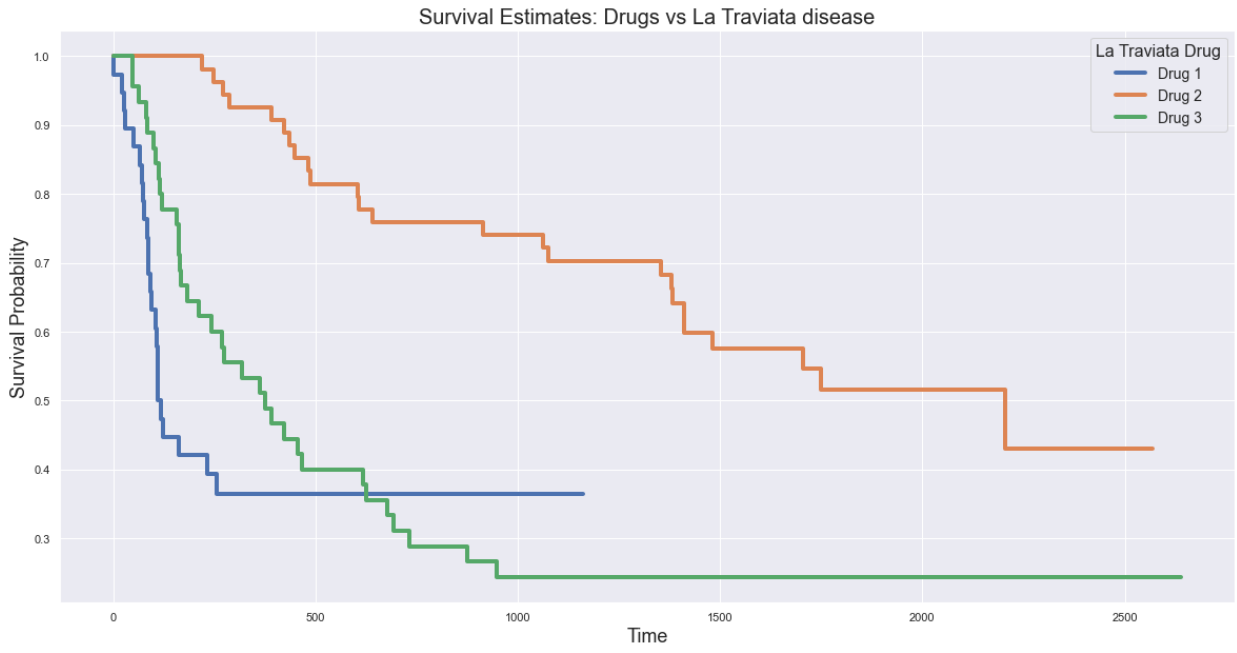


2. You are studying three different new drugs that may help slow the progress of La Traviata disease which compels people to sing opera until they exhaust themselves and die. Do the following:

a. Draw a survival plot that shows the survival curves for all three drugs.

In [15]:
```
# make look nice
plt.figure(figsize=(20,10))
sns.set(style="darkgrid")
```

```
#instantiate 3 dif
kmf1 = KaplanMeierFitter()
kmf2 = KaplanMeierFitter()
kmf3 = KaplanMeierFitter()

# split drugs up
group1 = long_df[long_df['Group'] == 1]
group2 = long_df[long_df['Group'] == 2]
group3 = long_df[long_df['Group'] == 3]

# Fit
kmf1.fit(durations=group1['time'], event_observed=group1['event'])
kmf2.fit(durations=group2['time'], event_observed=group2['event'])
kmf3.fit(durations=group3['time'], event_observed=group3['event'])

# Plot
ax = kmf1.plot_survival_function(ci_show=False, label='Drug 1',lw=4)
kmf2.plot_survival_function(ax=ax, ci_show=False, label='Drug 2',lw=4)
kmf3.plot_survival_function(ax=ax, ci_show=False, label='Drug 3',lw=4)

# Add legend + Customize
plt.xlabel('Time', fontsize=18)
plt.ylabel('Survival Probability', fontsize=18)
plt.title('Survival Estimates: Drugs vs La Traviata disease', fontsize=20)
plt.legend(title='La Traviata Drug', fontsize=14, title_fontsize=16)


plt.show();
```



b. Test to see if overall there is an effect of any of the drugs on survival taken as a global set.

In [18]:
```
# Multivar logrank

print('H0: Groups have identical hazard functions')


result = multivariate_logrank_test(
```

```
    event_durations=long_df['time'],
    groups=long_df['Group'],
    event_observed=long_df['event']
)

alpha = .05

print(f'\ntest stat: {result.test_statistic:.3}')
print(f'p val: {result.p_value:.3}')

if result.p_value < alpha:
    print('\nReject H0. Groups DO NOT have identical hazard functions')
else:
    print('Cannot Reject H0')
```

```
H0: Groups have identical hazard functions

test stat: 25.7
p val: 2.67e-06

Reject H0. Groups DO NOT have identical hazard functions
```

c. Compare the survival curves for each of the three drugs with each other (three comparisons) and see if any if the curves are different from each other. Note that you should be sure to adjust for multiple group comparisons.

In [19]:

```
# Log-rank test
print('H0: Groups have identical hazard functions\n')
compare_1_2 = logrank_test(group1['time'], group2['time'], event_observed_A=group1['ev
compare_1_3 = logrank_test(group1['time'], group3['time'], event_observed_A=group1['ev
compare_2_3 = logrank_test(group2['time'], group3['time'], event_observed_A=group2['ev

# grab p-values
p_values = pd.Series([compare_1_2.p_value, compare_1_3.p_value, compare_2_3.p_value],

# Adjusting for multiple comparison
p_adjusted = p_values * 3  # Multiplying by the number of comparisons

#  results
print("Adjusted p-values for multiple comparisons:\n")
print(p_adjusted)
print('\nReject H0 in all cases. Each group has a unique hazard function')
```

```
H0: Groups have identical hazard functions

Adjusted p-values for multiple comparisons:

1 vs 2    9.824754e-07
1 vs 3    1.141928e+00
2 vs 3    1.210205e-04
dtype: float64

Reject H0 in all cases. Each group has a unique hazard function
```