Fitting a Kaplan-Meier estimator

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

What is the Kaplan-Meier estimator?

A non-parametric statistic that **estimates the survival function** of **time-to-event** data.

- Also known as
  - the product-limit estimator
  - the K-M estimator
- Non-parametric: constructs a survival curve from collected data and does not assume underlying distribution

The mathematical intuition

Definitions:

- t_i: a duration time
- d_i: number of events that happened at time t_i
- n_i: number of individuals known to have survived up to time t_i

Survival function $S(t)$ is estimated with:

$$S(t) = \prod_{i: t_i \leq t} \left(1 - \frac{d_i}{n_i}\right)$$

Why is it called the product-limit estimator?

Suppose we have events at 3 times: 1, 2, 3

Survival rate for $t = 2$:

$$S(t=2) = \left(1 - \frac{d_1}{n_1}\right) * \left(1 - \frac{d_2}{n_2}\right)$$

Survival rate for $t = 3$:

$$S(t=3) = S(t=2) * \left(1 - \frac{d_3}{n_3}\right)$$

The survival rate at time t is equal to the product of the percentage chance of surviving at time t and each prior time.

Assumptions to keep in mind

- **Unambiguous events:** the event of interest happens at a clearly specified time.
- **Survival probabilities are comparable in all subjects**: individuals' survival probabilities do not depend on when they entered the study.
- **Censorship is non-informative:** censored observations have the same survival prospects as observations that continue to be followed.

Kaplan-Meier estimator with lifelines

```
from lifelines import KaplanMeierFitter
```

`KaplanMeierFitter` : a class of the `lifelines` library

```
kmf = KaplanMeierFitter()
kmf.fit(durations, event_observed)
```

The mortgage problem example

DataFrame name: `mortgage_df`

id	duration	paid_off
1	25	0
2	17	1
3	5	0
...	...	...
100	30	1

The mortgage problem example

DataFrame name: `mortgage_df`

id	duration	paid_off
1	25	0
2	17	1
3	5	0
...	...	...
100	30	1

```
from lifelines import KaplanMeierFitter
```

```
mortgage_kmf = KaplanMeierFitter()
mortgage_kmf.fit(duration=mortgage_df["duration"],
        event_observed=mortgage_df["paid_off"])
```

```
<lifelines.KaplanMeierFitter:"KM_estimate",
fitted with 100 total observations,
18 right-censored observations>
```

Using the Kaplan-Meier estimator

What is the median length of an outstanding mortgage?

```
print(mortgage_kmf.median_survival_time_)
```

```
4.0
```

What is the probability of a mortgage being outstanding every year after initiation?

```
print(mortgage_kmf.survival_function_)
```

```
          KM_estimate
timeline
0.0          1.000000
1.0          0.983267
2.0          0.950933
3.0          0.892328
```

Using the Kaplan-Meier estimator

What is the probability that a mortgage is not paid off by year 34 after initiation?

```
mortgage_kmf.predict(34)
```

```
0.037998
```

Benefits and limitations

Benefits

- Intuitive interpretation of survival probabilities.
- Flexible to use on any time-to-event data.
- Usually the first model to attempt on time-to-event data.

Limitations

- Survival curve is not smooth.
- If 50% of more of the data is censored, `.median_survival_time_` cannot be calculated.
- Not effective for analyzing the effect of covariates on the survival function.

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

Visualizing your Kaplan-Meier model

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

How to construct a Kaplan-Meier survival curve?

Toy data with $n = 5$:

duration	observed
2	1
5	0
3	1
5	1
2	0

Step 1: Arrange data in increasing order. If tied, censored data comes after uncensored data.

Step 2: For each t_i, calculate d_i, n_i, and $\left(1 - \frac{d_i}{n_i}\right)$

Step 3: For each t_i, multiply $\left(1 - \frac{d_i}{n_i}\right)$ with $\left(1 - \frac{d_{i-1}}{n_{i-1}}\right)$, $\left(1 - \frac{d_{i-2}}{n_{i-2}}\right)$, ... , $\left(1 - \frac{d_0}{n_0}\right)$

How to construct a Kaplan-Meier survival curve?

Step 1: Arrange durations in increasing order. If tied, censored data comes after uncensored data.

duration
2
5+
3
5
2+

Use "+" sign to denote censored data: 2, 5+, 3, 5, 2+

How to construct a Kaplan-Meier survival curve?

Step 1: Arrange durations in increasing order. If tied, censored data comes after uncensored data.

t_i
2, 2+
3
5, 5+

How to construct a Kaplan-Meier survival curve?

Step 2: For each t_i, calculate d_i, n_i, and $\left(1 - \frac{d_i}{n_i}\right)$

t_i
2, 2+
3
5, 5+

How to construct a Kaplan-Meier survival curve?

Step 2: For each t_i, calculate d_i, n_i, and $\left(1 - \frac{d_i}{n_i}\right)$

t_i	d_i
2, 2+	1
3	1
5, 5+	1

How to construct a Kaplan-Meier survival curve?

Step 2: For each t_i, calculate d_i, n_i, and $\left(1 - \frac{d_i}{n_i}\right)$

t_i	d_i	n_i
2, 2+	1	5
3	1	3
5, 5+	1	2

How to construct a Kaplan-Meier survival curve?

Step 2: For each t_i, calculate d_i, n_i, and $(1 - \frac{d_i}{n_i})$

t_i	d_i	n_i	$(1 - \frac{d_i}{n_i})$
2, 2+	1	5	$4/5$
3	1	3	$2/3$
5, 5+	1	2	$1/2$

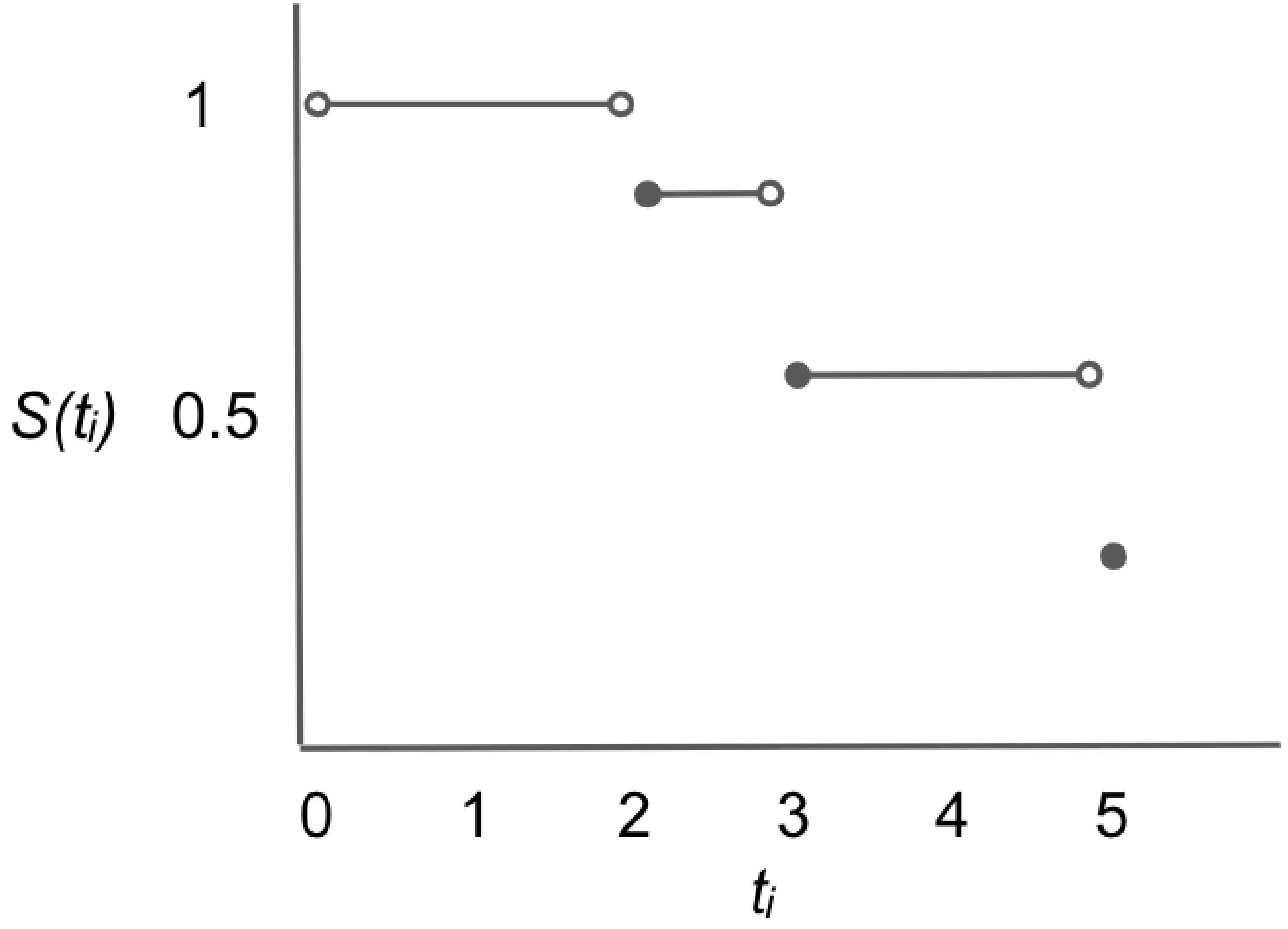
How to construct a Kaplan-Meier survival curve?

Step 3: For each t_i, multiply $\left(1 - \frac{d_i}{n_i}\right)$ with $\left(1 - \frac{d_{i-1}}{n_{i-1}}\right)$, $\left(1 - \frac{d_{i-2}}{n_{i-2}}\right)$, ... , $\left(1 - \frac{d_0}{n_0}\right)$

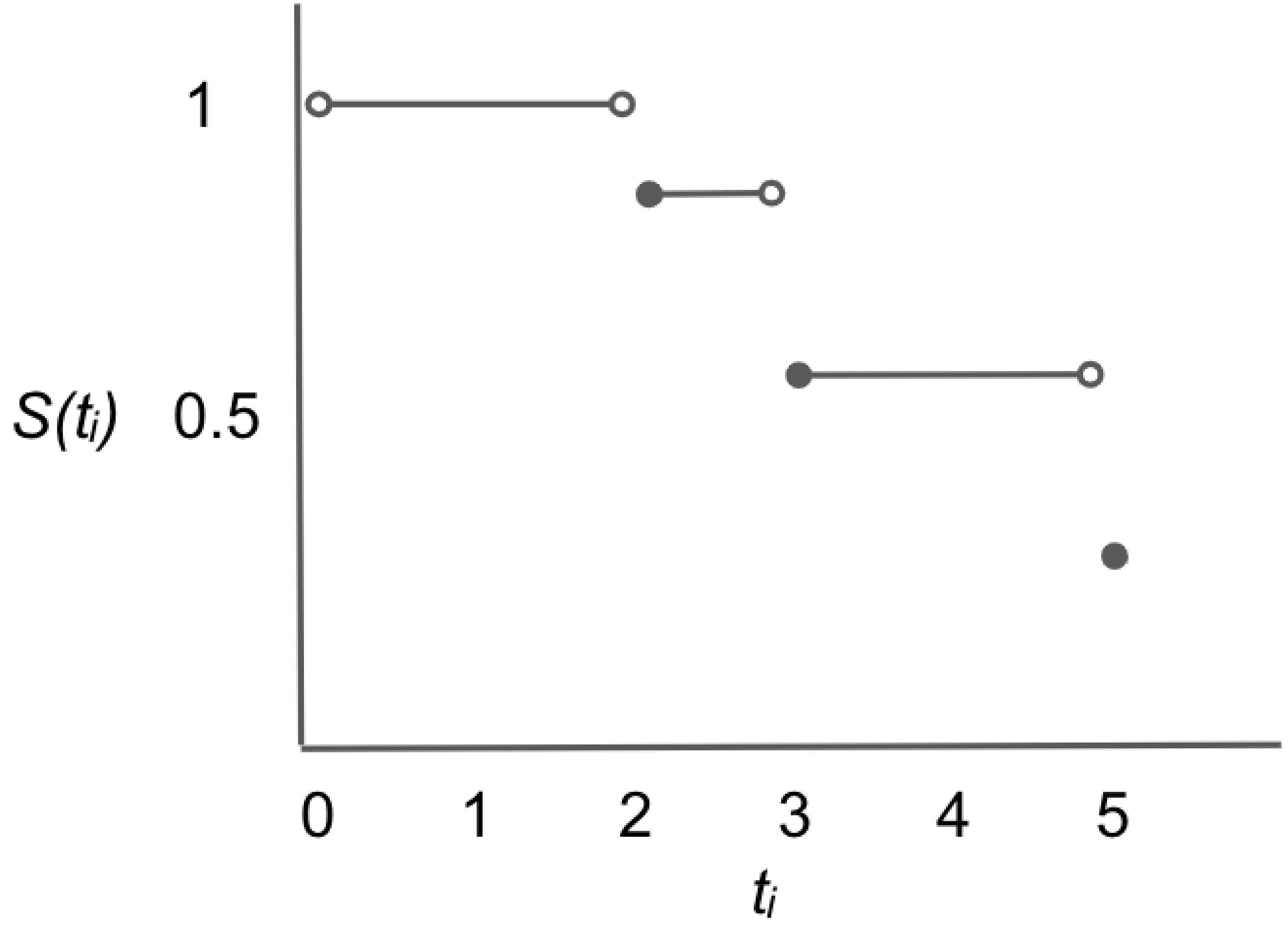
t_i	d_i	n_i	$\left(1 - \frac{d_i}{n_i}\right)$	$S(t_i)$
2, 2+	1	5	4/5	4/5 = 0.8
3	1	3	2/3	4/5 · 2/3 = 0.53
5, 5+	1	2	1/2	4/5 · 2/3 · 1/2 = 0.27

How to construct a Kaplan-Meier survival curve?

t_i	d_i	n_i	$\left(1 - \frac{d_i}{n_i}\right)$	$S(t_i)$
2, 2+	1	5	$4/5$	0.8
3	1	3	$2/3$	0.53
5, 5+	1	2	$1/2$	0.27



Interpreting the survival curve



- The survival probabilities at each time between 0 and 5.
- Common misconception: If the curve goes to 0, no subjects survived.
- The curve will drop to zero if the last observation is not censored (true event duration is known).

Plotting the Kaplan-Meier survival curve

```
from lifelines import KaplanMeierFitter
import matplotlib.pyplot as plt

kmf = KaplanMeierFitter()
kmf.fit(durations, event_observed)
```

```
kmf.survival_function_.plot()
plt.show()
```

The mortgage problem example

DataFrame name: `mortgage_df`

id	duration	paid_off
1	25	0
2	17	1
3	5	0
...	...	...
100	30	1

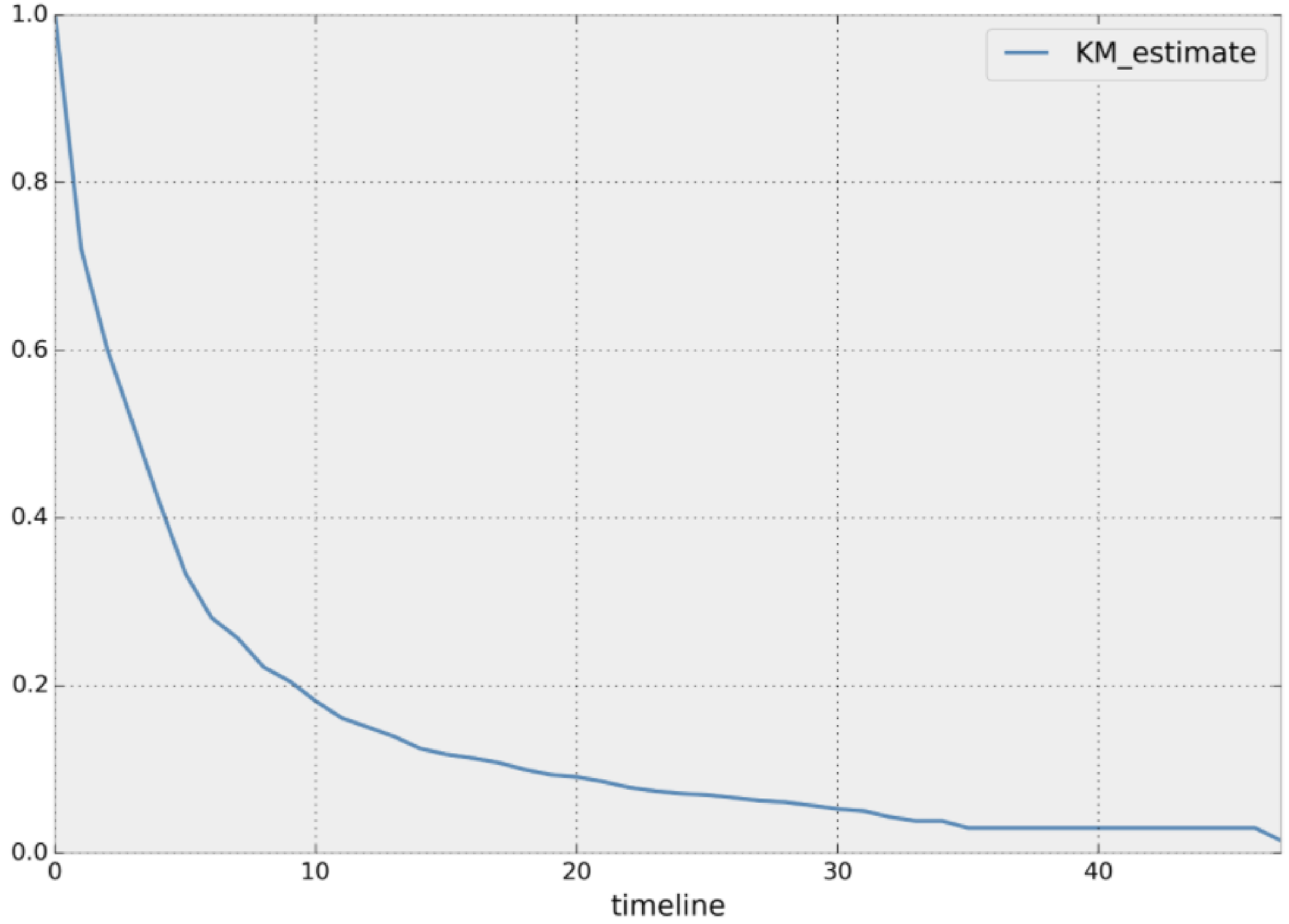
```
from lifelines import KaplanMeierFitter
from matplotlib import pyplot as plt
```

```
mortgage_kmf = KaplanMeierFitter()
mortgage_kmf.fit(duration=mortgage_df["duration"],
        event_observed=mortgage_df["paid_off"])
```

```
mortgage_kmf.survival_function_.plot()
```

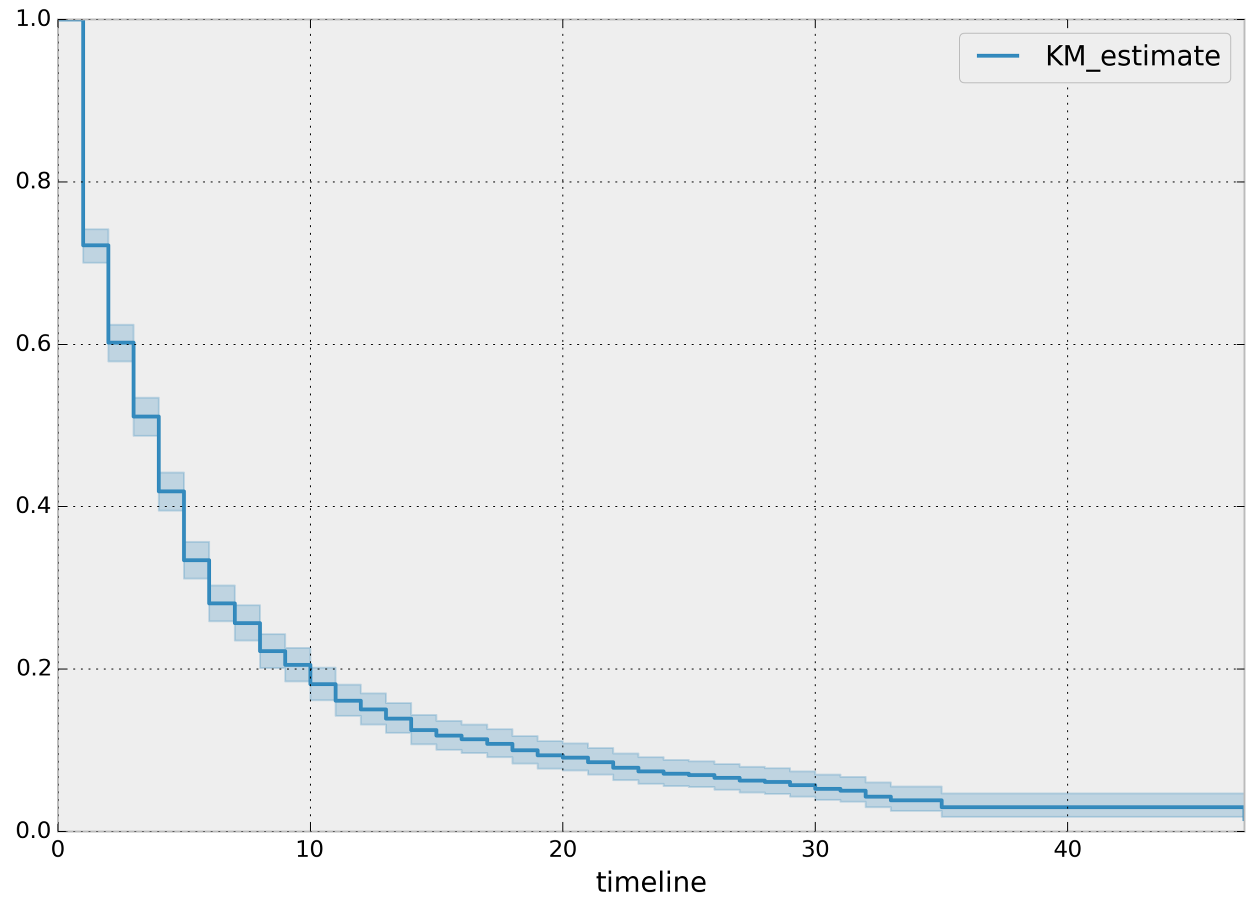
The mortgage problem example

```
plt.show()
```



Survival curve confidence interval

```
mortgage_kmf.plot_survival_function()
plt.show()
```



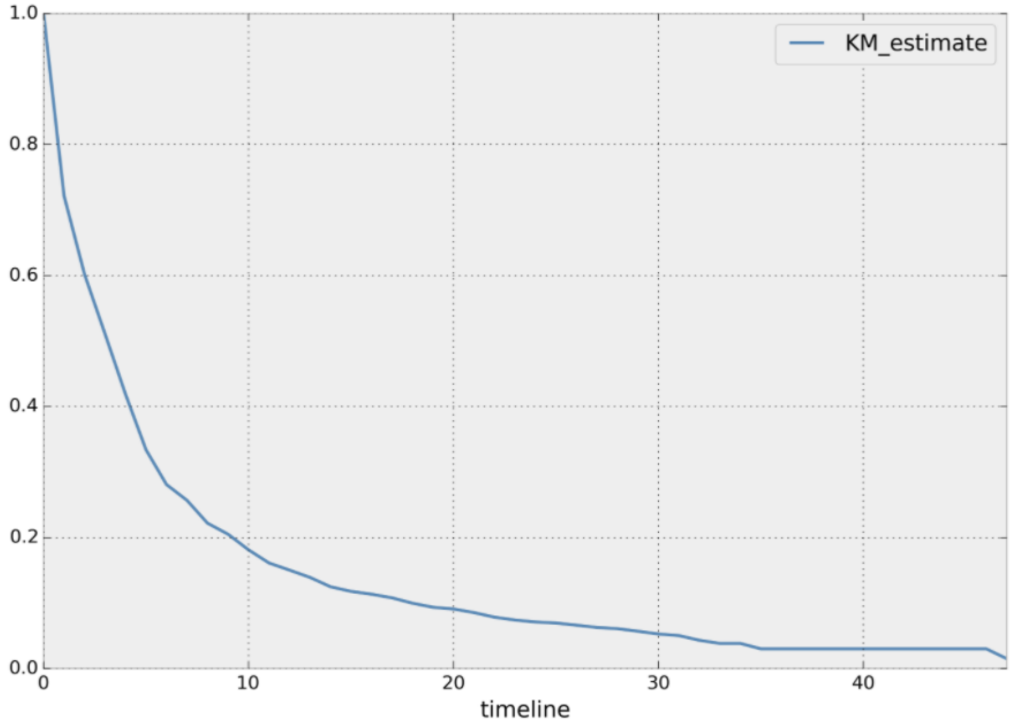
Why is the confidence interval useful?

- A way to quantify how uncertain we are about each point estimate of survival probabilities
- A **wide confidence interval** means we are **less certain**, often due to small sample size
- A **narrow confidence interval** means we are **more certain**, often due to large sample size

Ways to plot the Kaplan-Meier survival curve

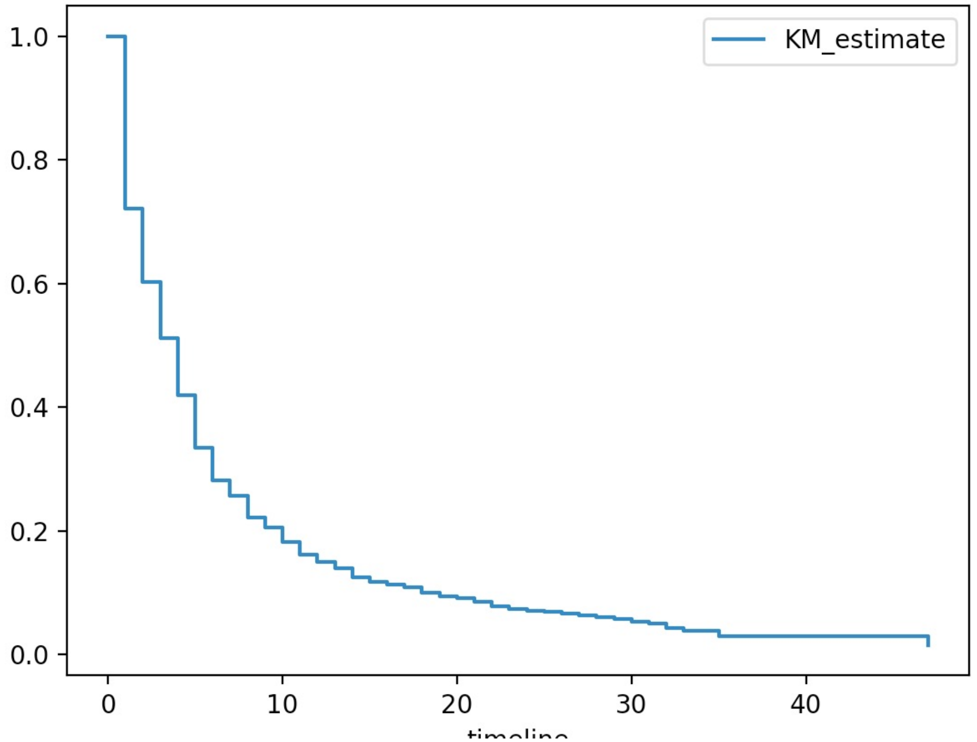
Plot survival function point estimates as a continuous line.

```
kmf.survival_function_.plot()
plt.show()
```



Plot survival function as a stepped line without the confidence interval.

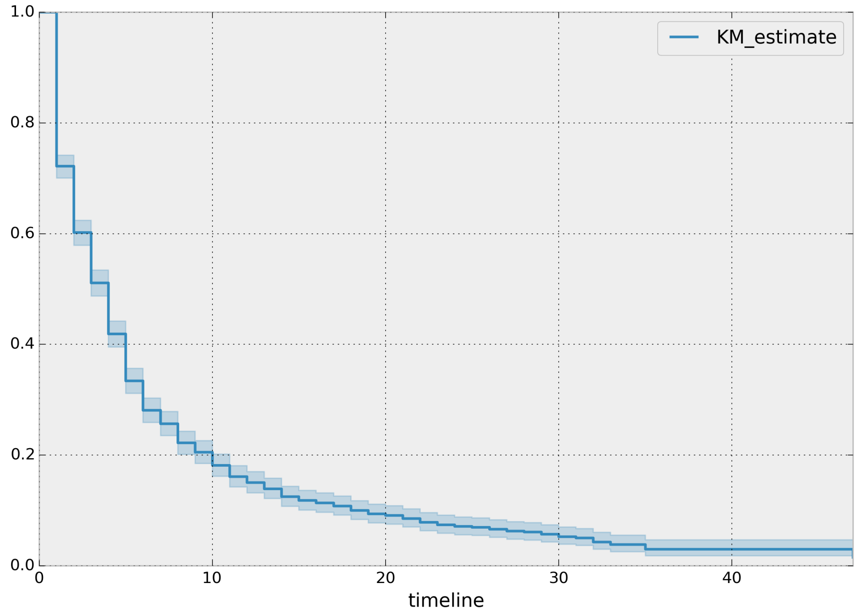
```
kmf.plot(ci_show=False)
plt.show()
```



Ways to plot the Kaplan-Meier survival curve

Plot survival function as a stepped line with the confidence interval.

```
kmf.plot()
plt.show()
```



Another way...

```
kmf.plot_survival_function()
plt.show()
```

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

Applying survival analysis to groups

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

The mortgage problem

DataFrame name: `mortgage_df`

id	property type	duration	paid_off
1	house	25	0
2	apartment	17	1
3	apartment	5	0
...	...	...	...
100	house	30	1

Property type: the type of home that's financed by the mortgage (either house or apartment)

Is there a difference in time to payoff for house versus apartment mortgages?

Comparing groups' survival distributions

We are often interested in assessing whether there are differences in survival (or event/survival probabilities) among different groups of subjects.

- **Dimensional attributes about the subjects**
  - i.e. different types of mortgages, different brands of tires
- **Different experiment groups**
  - i.e. treatment versus control groups
- **Different values for the same dimensional attribute**
  - i.e. high versus low income households

Types of survival group comparisons

1. Are any point estimates or survival statistics different?

- Compare two groups' survival probabilities at a specific time
- Compare total proportion of survived subjects across two groups

Types of survival group comparisons

2. Are the underlying distributions different?

- Requires formal hypothesis testing

Types of survival group comparisons

3. How much does an attribute affect survival?

- Requires regression-based modeling frameworks

Visualizing group differences

Fitting a Kaplan-Meier survival function to each group and visualize their survival curves side-by-side.

Benefits:

- Simple and straight-forward to use and interpret.
- Non-parametric means it is more flexible for different types of survival distributions.
- Useful illustrative tool for demonstrating differences in survival functions.

Identifying the groups

DataFrame name: `mortgage_df`

id	property type	duration	paid_off
1	house	25	0
2	apartment	17	1
3	apartment	5	0
...	...	...	...
100	house	30	1

Create a Boolean mask for each group.

```
house = (mortgage_df["property_type"]=="house")
apt = (mortgage_df["property_type"]=="apartment")
```

If there are only 2 groups, only 1 mask is necessary. The other group could be referenced using negation.

Fitting and plotting survival curves

Create one figure and instantiate a `KaplanMeierFitter` class.

```
ax = plt.subplot(111)
mortgage_kmf = KaplanMeierFitter()
```

Fit `mortgage_kmf` to the house group and plot on the figure `ax`.

```
mortgage_kmf.fit(duration=mortgage_df[house]["duration"],
                 event_observed=mortgage_df[house]["paid_off"],
                 label="Houses")
mortgage_kmf.plot_survival_function(ax=ax)
```

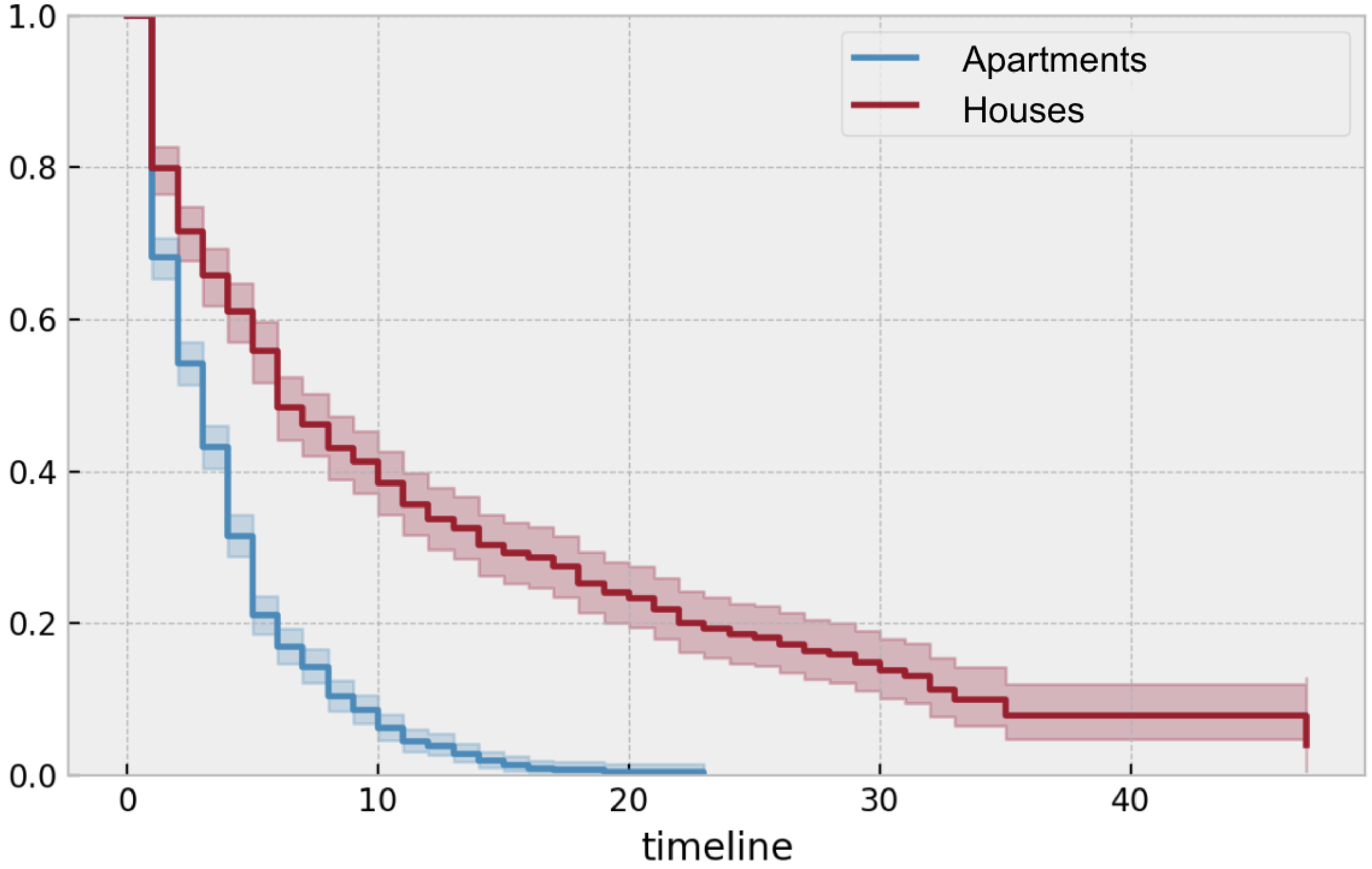
Fitting and plotting survival curves

Fit `mortgage_kmf` to the apartment group and plot on the figure `ax` .

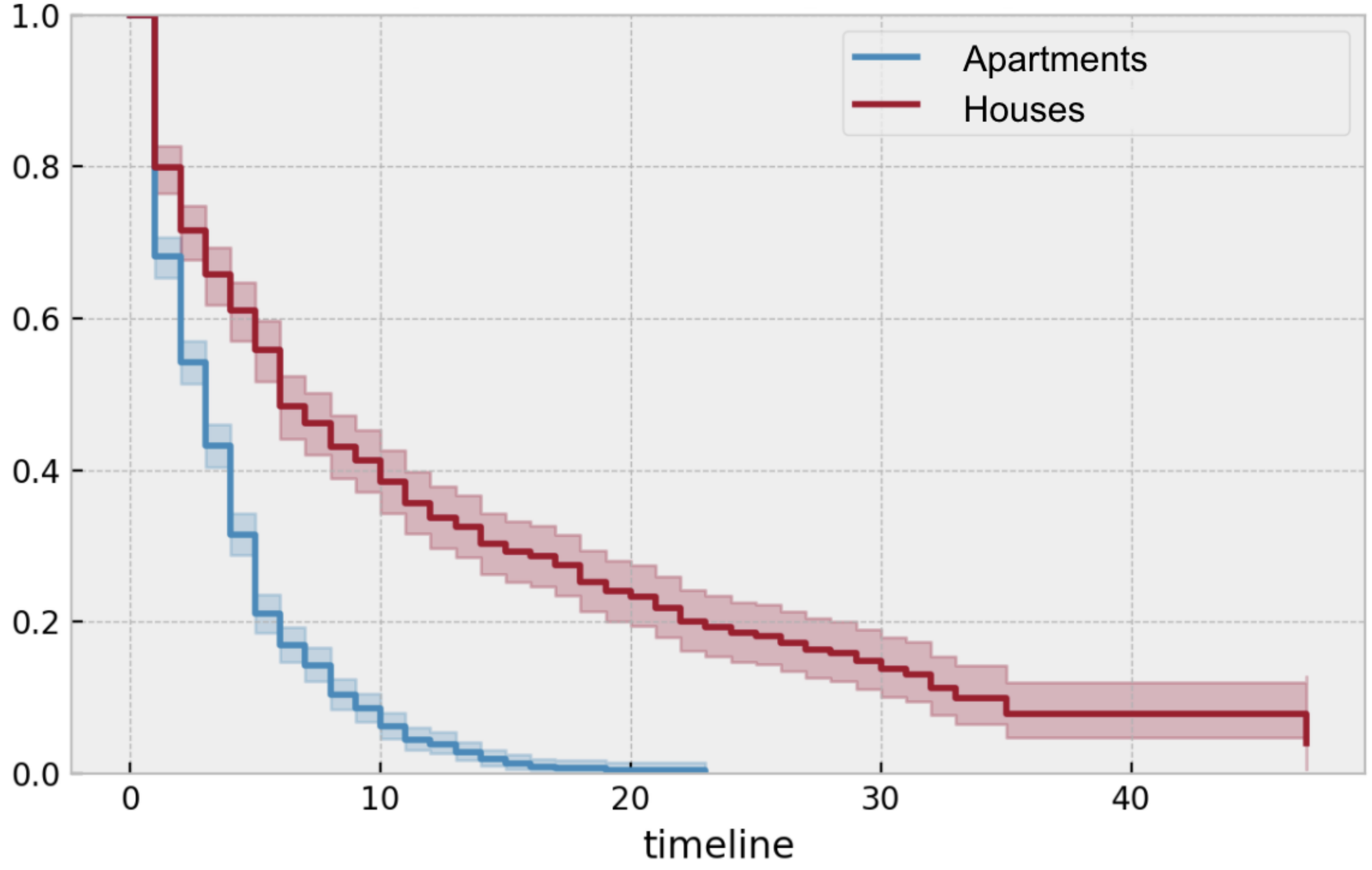
```
mortgage_kmf.fit(duration=mortgage_df[apt]["duration"],
                 event_observed=mortgage_df[apt]["paid_off"],
                 label="Apartments")
mortgage_kmf.plot_survival_function(ax=ax)
```

Visualizing side-by-side

```
plt.show()
```



Interpreting groups' survival curves



- Apartment mortgages seem to be paid off faster than house mortgages on average.
- At any given duration, a higher proportion of users pay off apartment mortgages than house mortgages.

Note: if the confidence intervals overlap at some points, it's less likely that there's a real difference between the curves.

Let's practice!

SURVIVAL ANALYSIS IN PYTHON

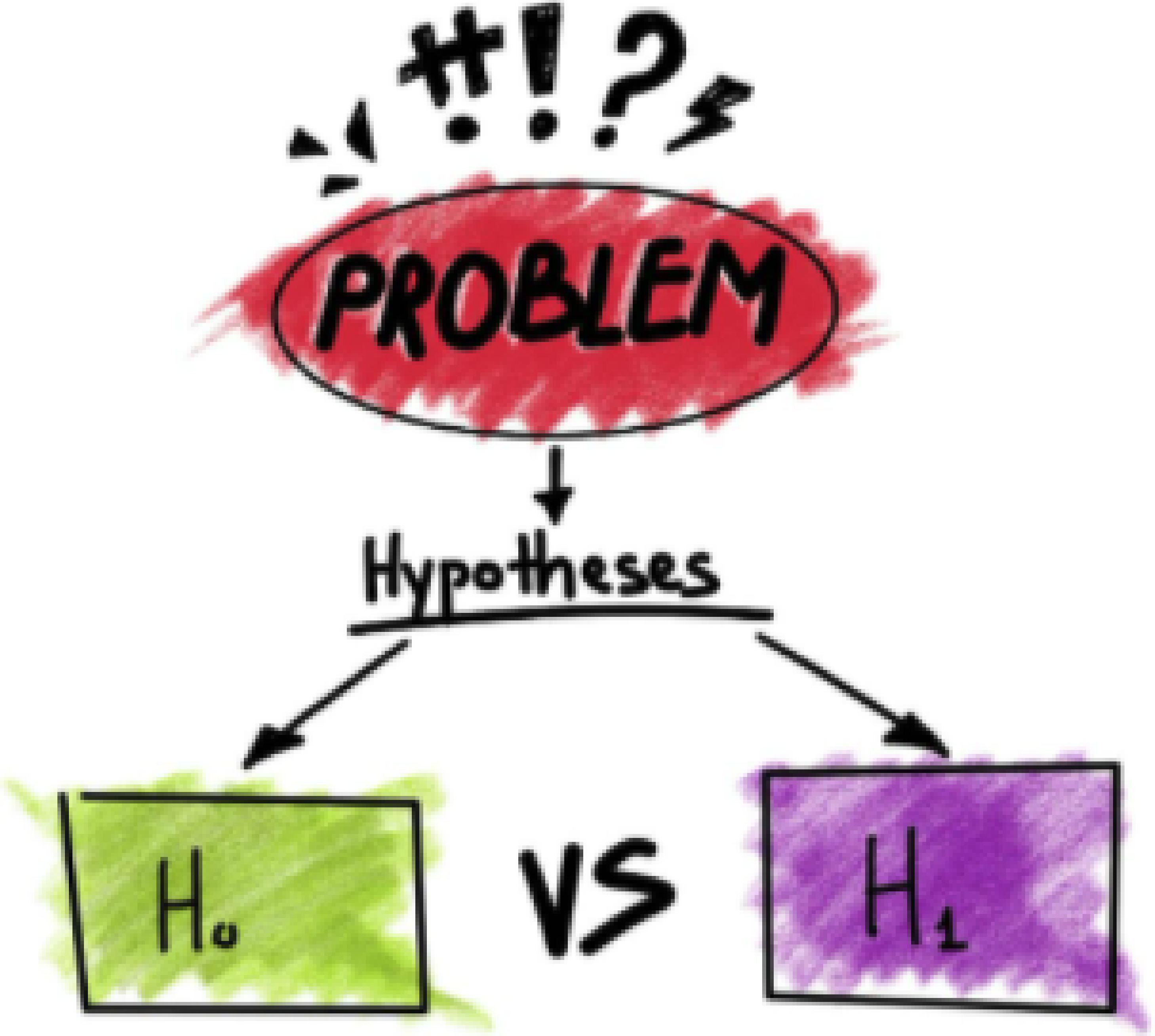
The log-rank test

SURVIVAL ANALYSIS IN PYTHON

Shae Wang
Senior Data Scientist

Hypothesis testing

- A method of statistical inference
- Null hypothesis H_0: e.g. California and Nevada residents have the same average income.
- Alternative hypothesis H_1: e.g. California and Nevada residents have different average income.
- P-value: what's the likelihood that the data would've occurred if the null hypothesis were true?



Log-rank hypothesis testing

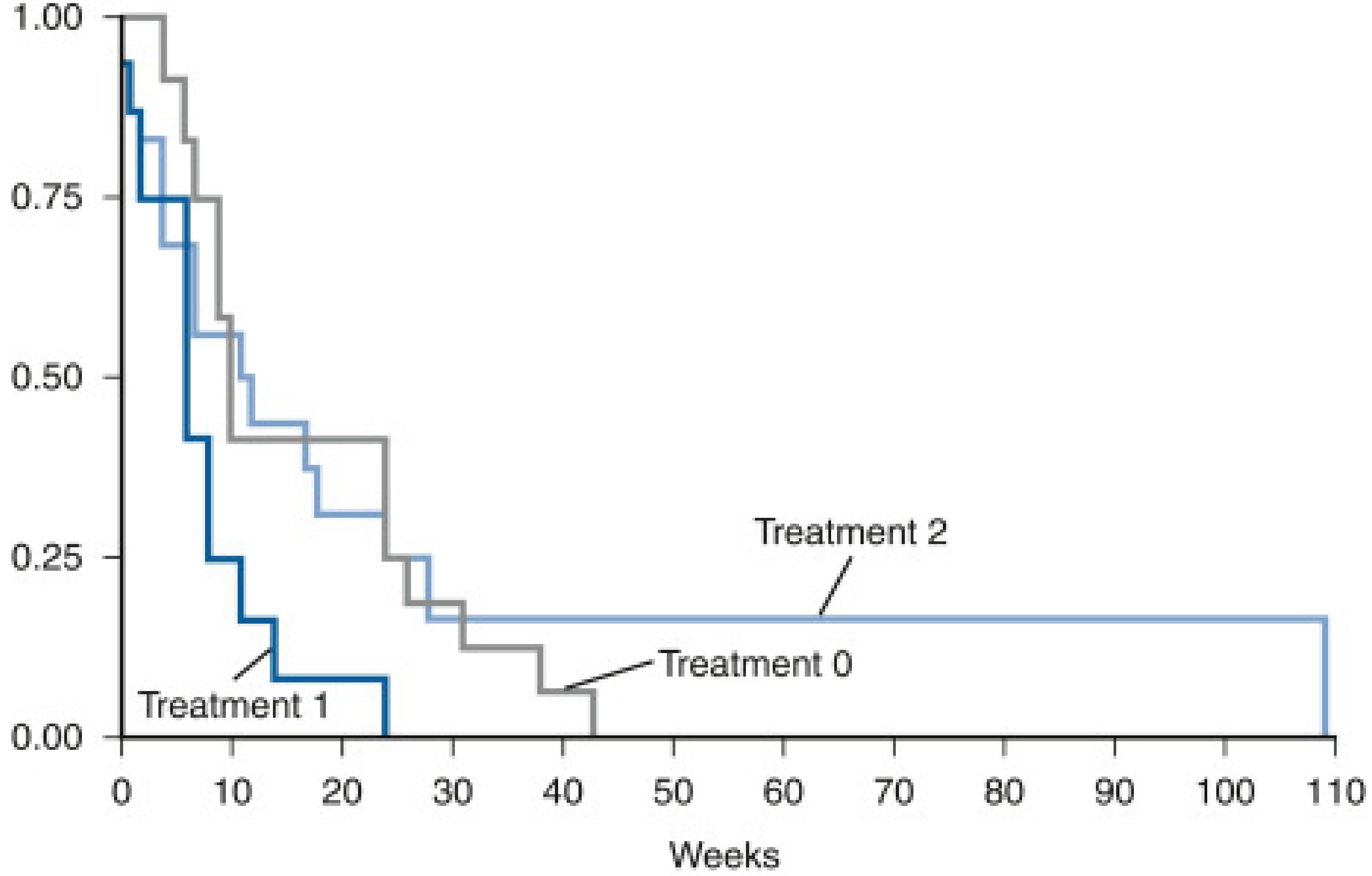
- Compares survival probabilities S_i between groups at each time t

H_0: $S_A(t) = S_B(t)$

H_1: $S_A(t) \neq S_B(t)$

- P-value: if $S_A(t) = S_B(t)$, what's the probability of our data occurring?

Multiple survival curves



Running the log-rank test

```
from lifelines.statistics import logrank_test
logrank_test(durations_A, durations_B, event_observed_A, event_observed_B)
```

- `.print_summary()`
- `.p_value`
- `.test_statistic`

Log-rank test example

Does the program change when babies start speaking?

```
t.head(2)
```

```
   id  duration  observed
0   1        12         0
1   4         6         1
```

```
c.head(2)
```

```
   id  duration  observed
0   0        11         1
1   2        14         0
```

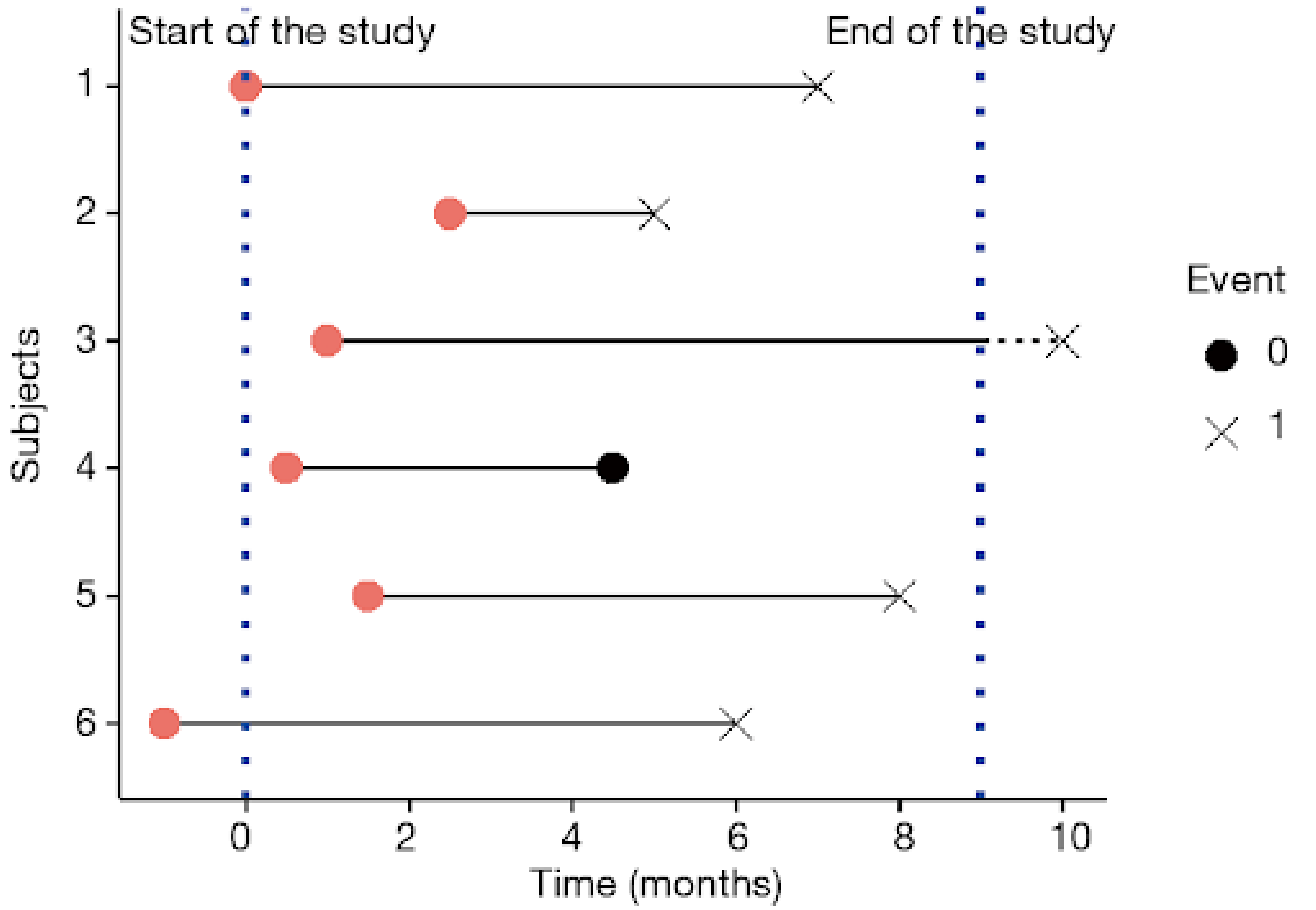
```
lrt = logrank_test(
    durations_A = t['duration'],
    durations_B = c['duration'],
    event_observed_A = t['observed'],
    event_observed_B = c['observed'])
```

```
lrt.print_summary()
```

```
<lifelines.StatisticalResult: logrank_test>
 null_distribution = chi squared
degrees_of_freedom = 1
         test_name = logrank_test
 test_statistic    p  -log2(p)
           0.09 0.77      0.38
```

Keep in mind...

- Log-rank test is a non-parametric hypothesis test
- When using `lifelines` , data must be right-censored (i.e. subject 3)
- Censorship must be non-informative
- For a log-rank test between $n > 2$ groups, use `pairwise_logrank_test()` or `multivariate_logrank_test()`



Let's practice!

SURVIVAL ANALYSIS IN PYTHON