24) Survival Function and Hazard Rates

Vitor Kamada

November 2018

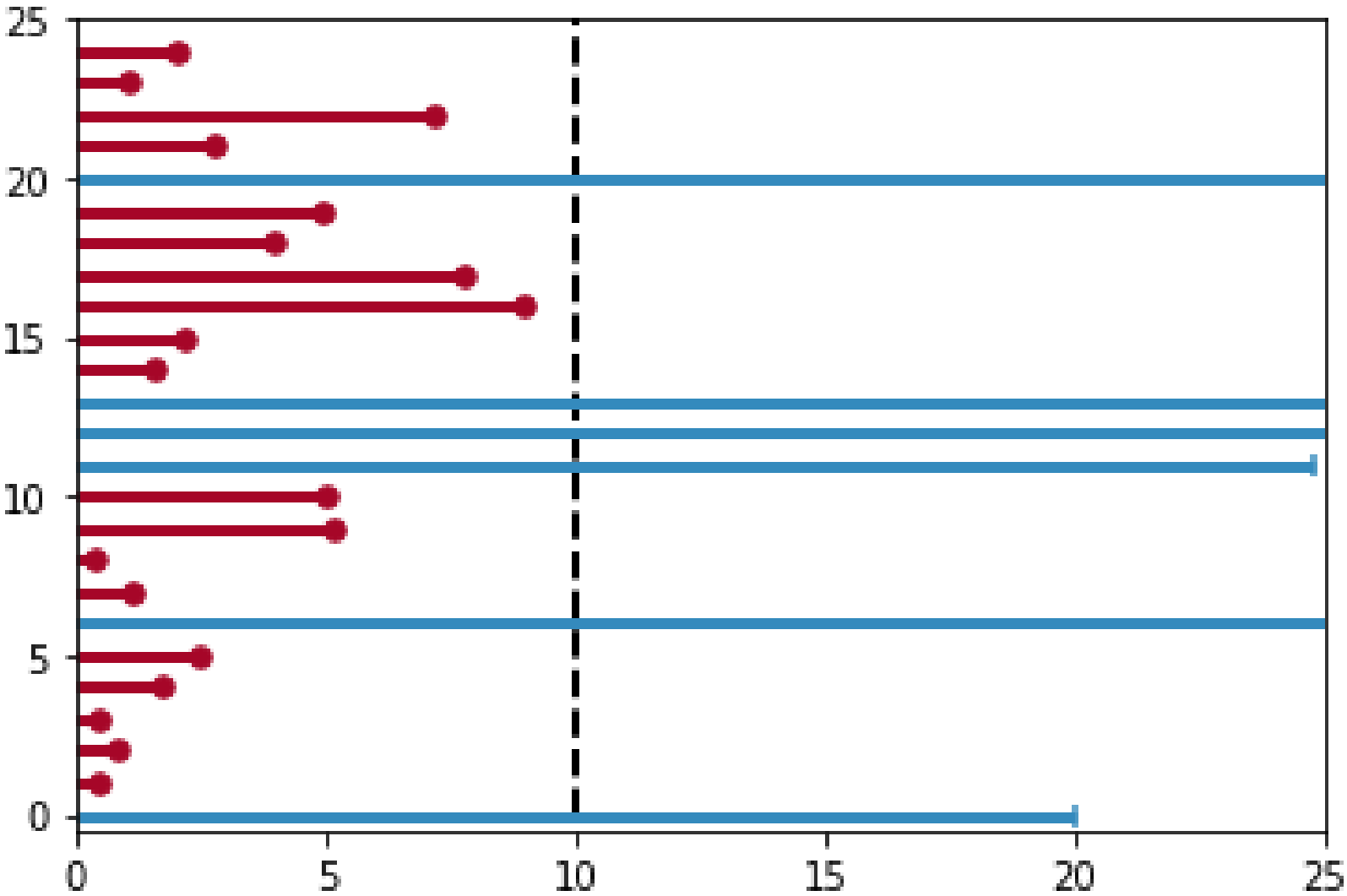
Reference

Tables, Graphics, and Figures from

https://lifelines.readthedocs.io/

Introduction to Survival Analysis

Simulated Data



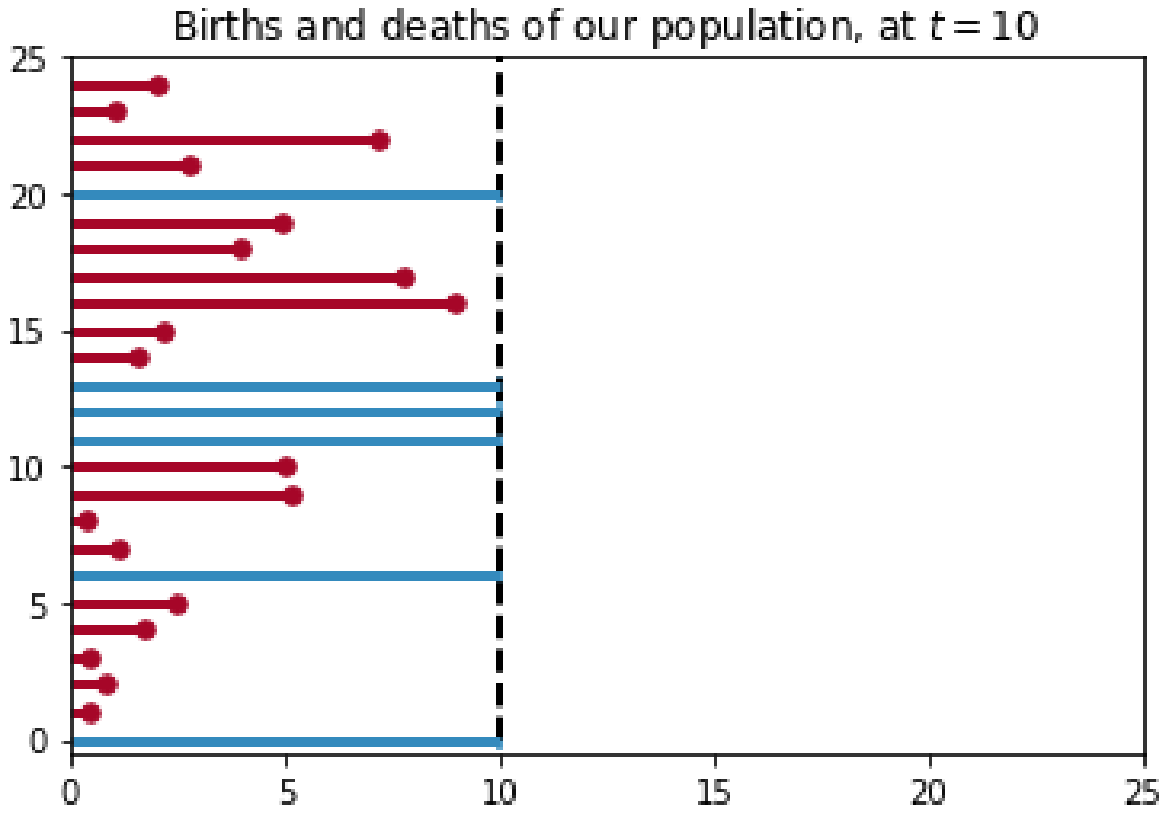
Generated Data

```
from lifelines.plotting import plot_lifetimes
from numpy.random import uniform, exponential
import numpy as np
import matplotlib.pyplot as plt
```

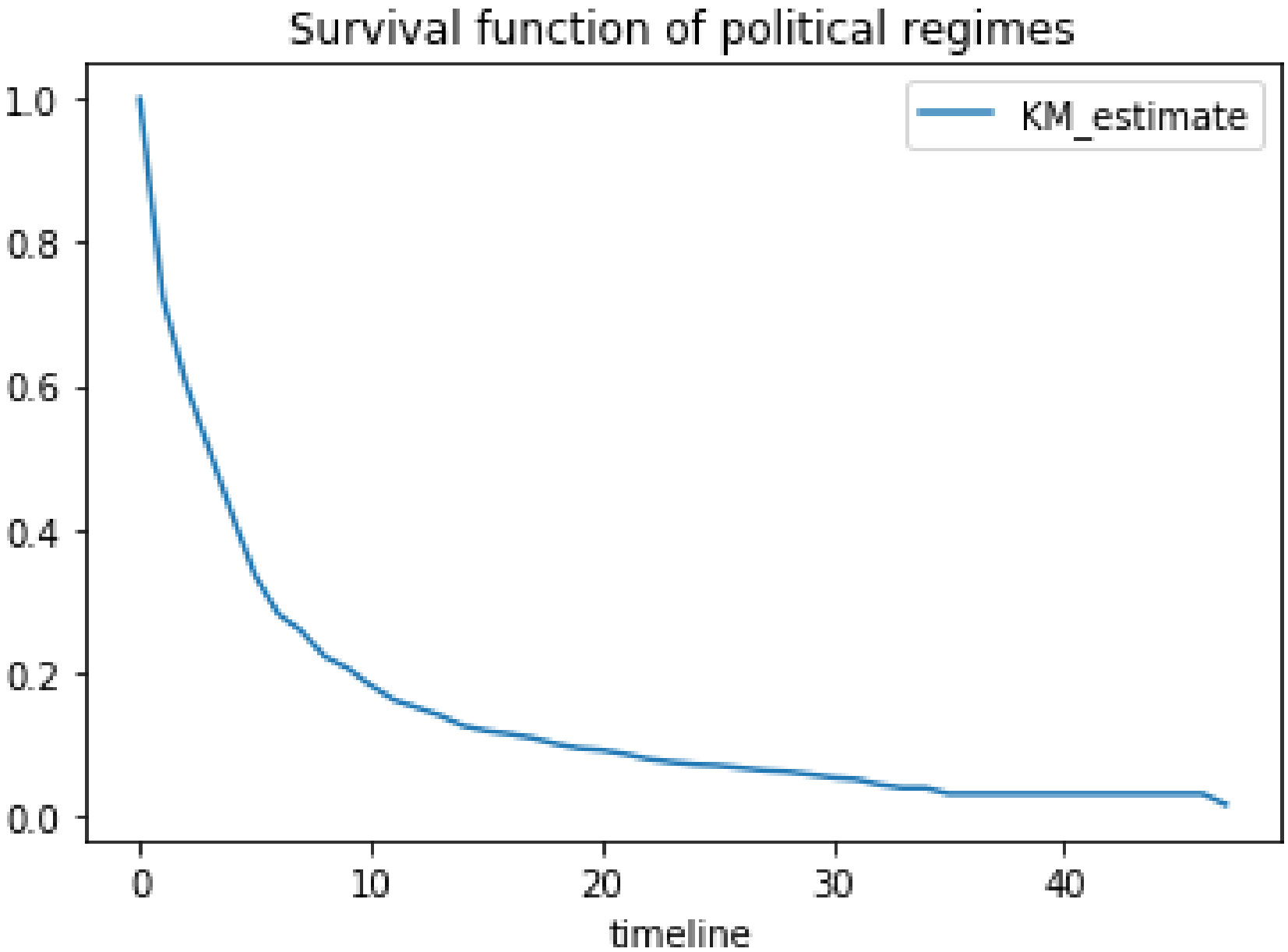
```
N = 25
current_time = 10
actual_lifetimes = np.array([[exponential(12),
  exponential(2)][uniform() < 0.5] for i in range(N)])
observed_lifetimes = np.minimum(actual_lifetimes,
                                current_time)
observed = actual_lifetimes < current_time
```

Observed Lifetimes at Time 10

```
plt.xlim(0, 25)
plt.vlines(10, 0, 30, lw=2, linestyles='--')
plt.xlabel("time")
plt.title("Births and deaths of our population, at $t=10$")
plot_lifetimes(observed_lifetimes, event_observed=observed)
```



Survival Function of Political Regimes



Survival Function

Prob. of surviving past time t

Prob. death event has not occured yet at time t

$$S(t) = Pr(T > t)$$

$$0 \leq S(t) \leq 1$$

$F(t) = 1 - S(t)$, where $F(t)$ is the CDF of T

$S(t)$ is non-increasing function of t

Kaplan-Meier Estimate of Survival Function

$$\hat{S}(t) = \prod_{t_i < t} \frac{n_i - d_i}{n_i}$$

d_i : # of death events at time t

n_i : # of subjects at risk of death just prior to time t

Import Data

```
import pandas as pd
from lifelines.datasets import load_dd
data = load_dd()
data.sample(6)
```

	leaderspellreg	democracy
369	Nikica Valentic.Croatia.1993.1994.Mixed Dem	Democracy
1645	Suleyman Demirel.Turkey.1975.1977.Parliamentar...	Democracy
827	Silvio Berlusconi.Italy.2008.2008.Parliamentar...	Democracy
1349	Armindo Vaz d'Almeida.Sao Tome and Principe.19...	Democracy
1154	Ali Saibou.Niger.1987.1992.Military Dict	Non-democracy

	regime	start_year	duration	observed
369	Mixed Dem	1993	2	1
1645	Parliamentary Dem	1975	3	1
827	Parliamentary Dem	2008	1	0
1349	Mixed Dem	1995	1	1
1154	Military Dict	1987	6	1

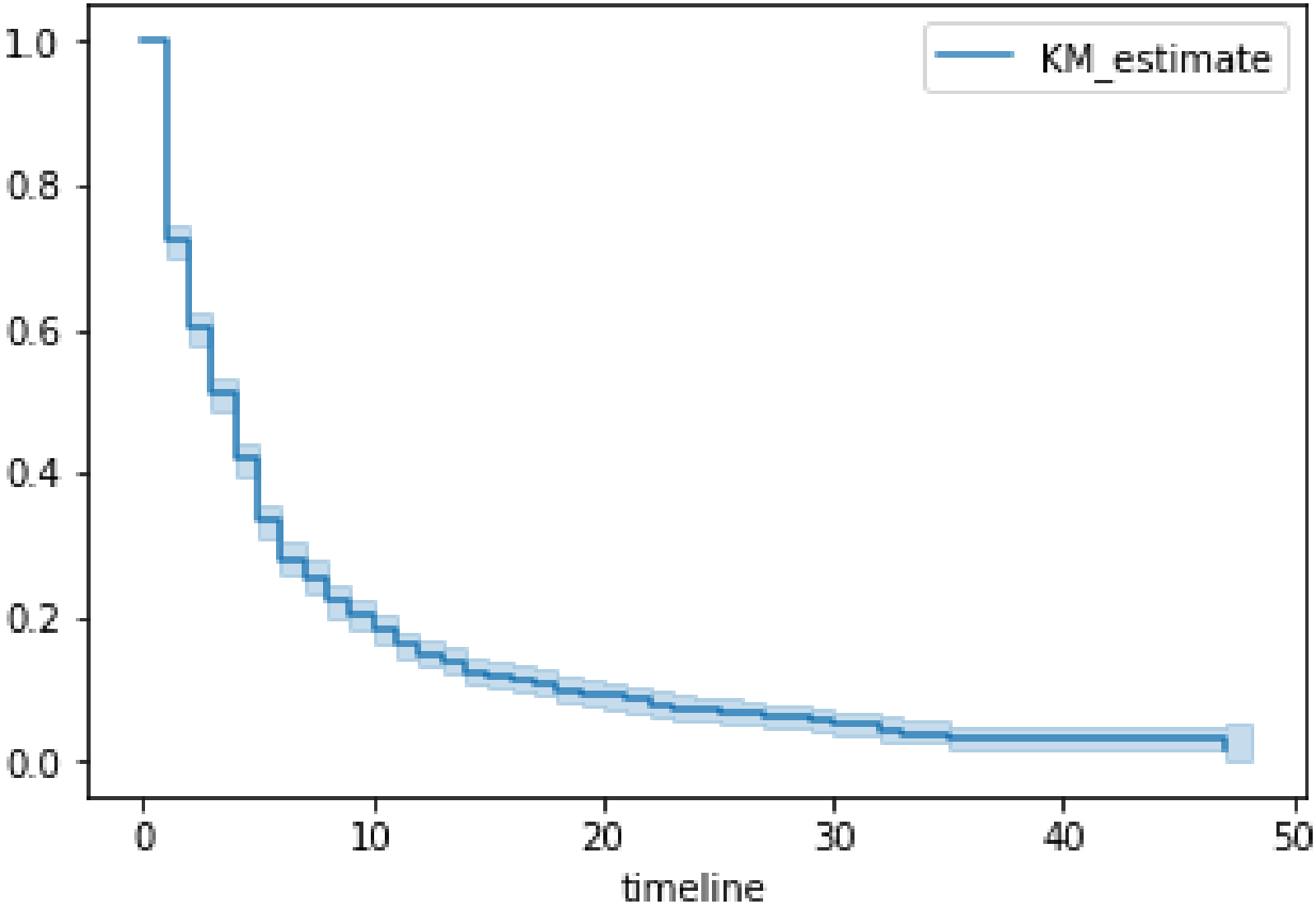
Kaplan Meier Fitter

```
from lifelines import KaplanMeierFitter
kmf = KaplanMeierFitter()

T = data["duration"]
E = data["observed"]

kmf.fit(T, event_observed=E)
```

kmf.median_

4

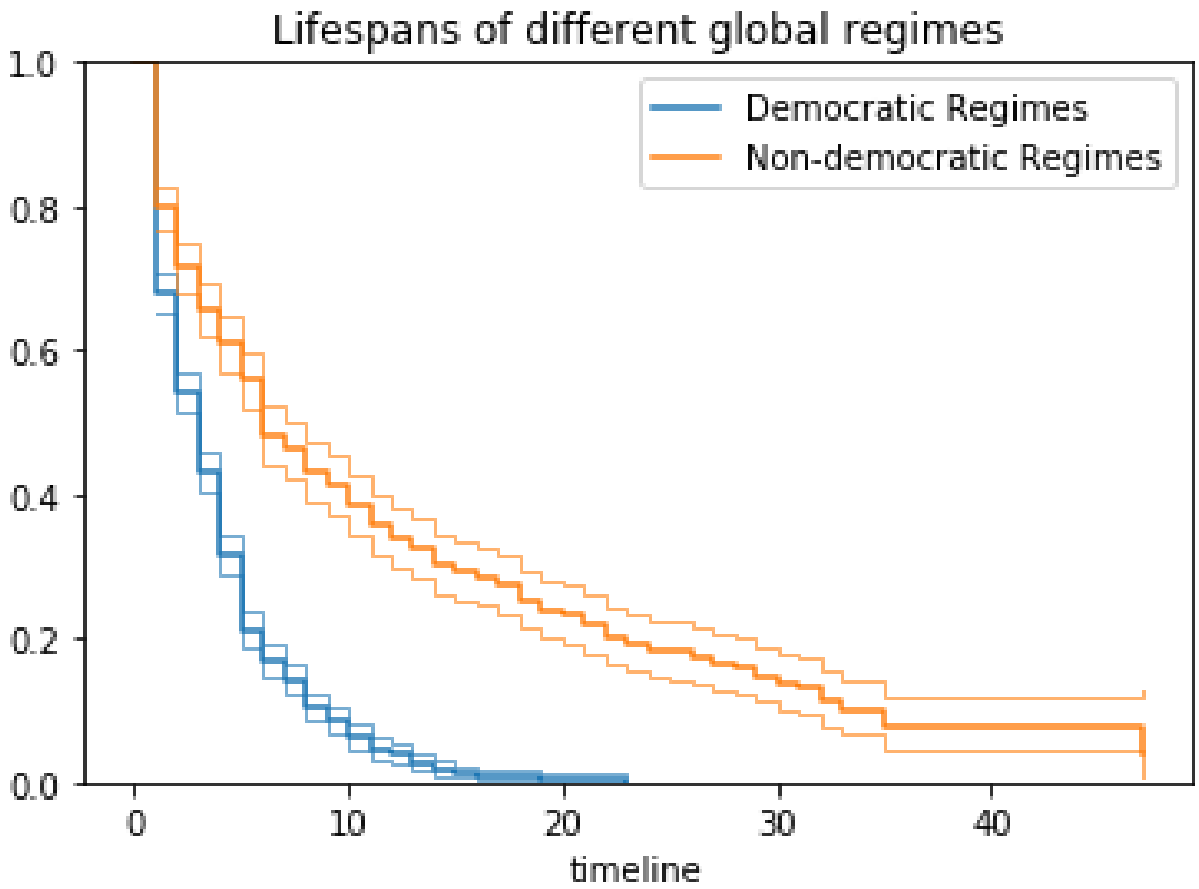
kmf.plot()



Democratic vs Non-Democratic Regimes

```
ax = plt.subplot(111)
dem = (data["democracy"] == "Democracy")

t = np.linspace(0, 50, 51)
kmf.fit(T[dem], event_observed=E[dem], timeline=t,
        label="Democratic Regimes")
ax = kmf.plot(ax=ax)
print("Median survival time of democratic:", kmf.median_)

kmf.fit(T[~dem], event_observed=E[~dem], timeline=t,
        label="Non-democratic Regimes")
ax = kmf.plot(ax=ax)
print("Median survival time of non-democratic:", kmf.median_)

plt.ylim(0,1)
plt.title("Lifespans of different global regimes");
```

Lifespans of Different Global Regimes



Median of democratic: **3.0**

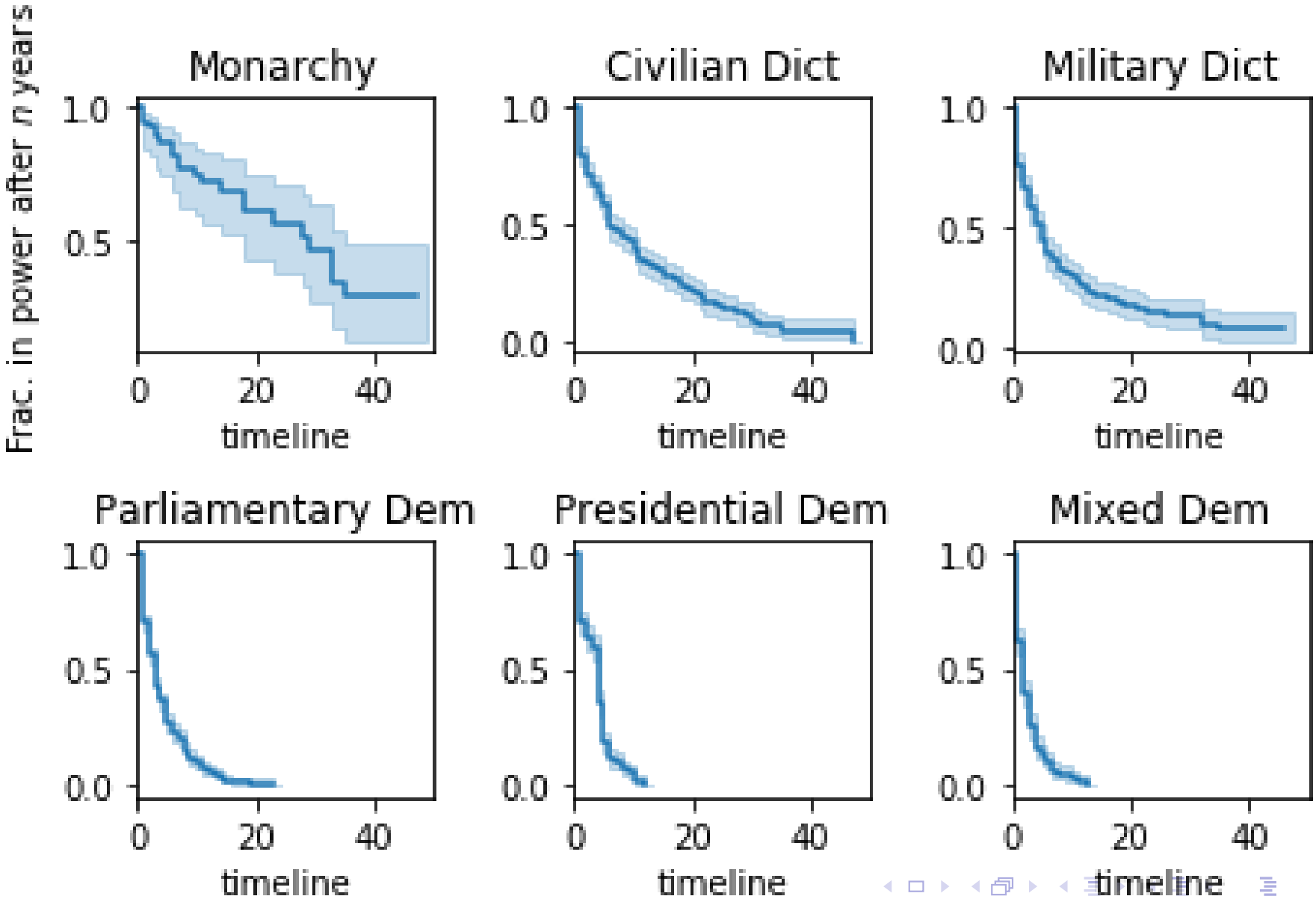
Median non-democratic: **6.0**

Logrank Test (Mantel-Cox Test)

```
from lifelines.statistics import logrank_test
results = logrank_test(T[dem], T[~dem],
                       E[dem], E[~dem], alpha=.99)
results.print_summary()
```

```
t_0=-1, alpha=0.99, null_distribution=chi squared, df=1

test_statistic      p
      260.4695 0.0000  ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' '
```

Lifespans of Regime Types



Hazard Curve

Prob. of death event occurring at time t, given that the death event has not occurred until time t

$$\lambda(t) = \lim_{\Delta t \to 0} \frac{Pr(t \leq T < t + \Delta t | T \geq t)}{\Delta t}$$

$$\lambda(t) = \frac{f(t)}{S(t)} = \frac{-S'(t)}{S(t)} = \frac{-dln(S(t))}{dt}$$

$$S(t) = exp(-\int_0^t \lambda(z)dz)$$

Estimating Hazard Rates by Nelson-Aalen Estimator

$$\Lambda(t) = \int_0^t \lambda(z)dz = -lnS(t)$$

$$\hat{\Lambda}(t) = \sum_{t_i \leq t} \frac{d_i}{n_i}$$

d_i : # of deaths at time t_i

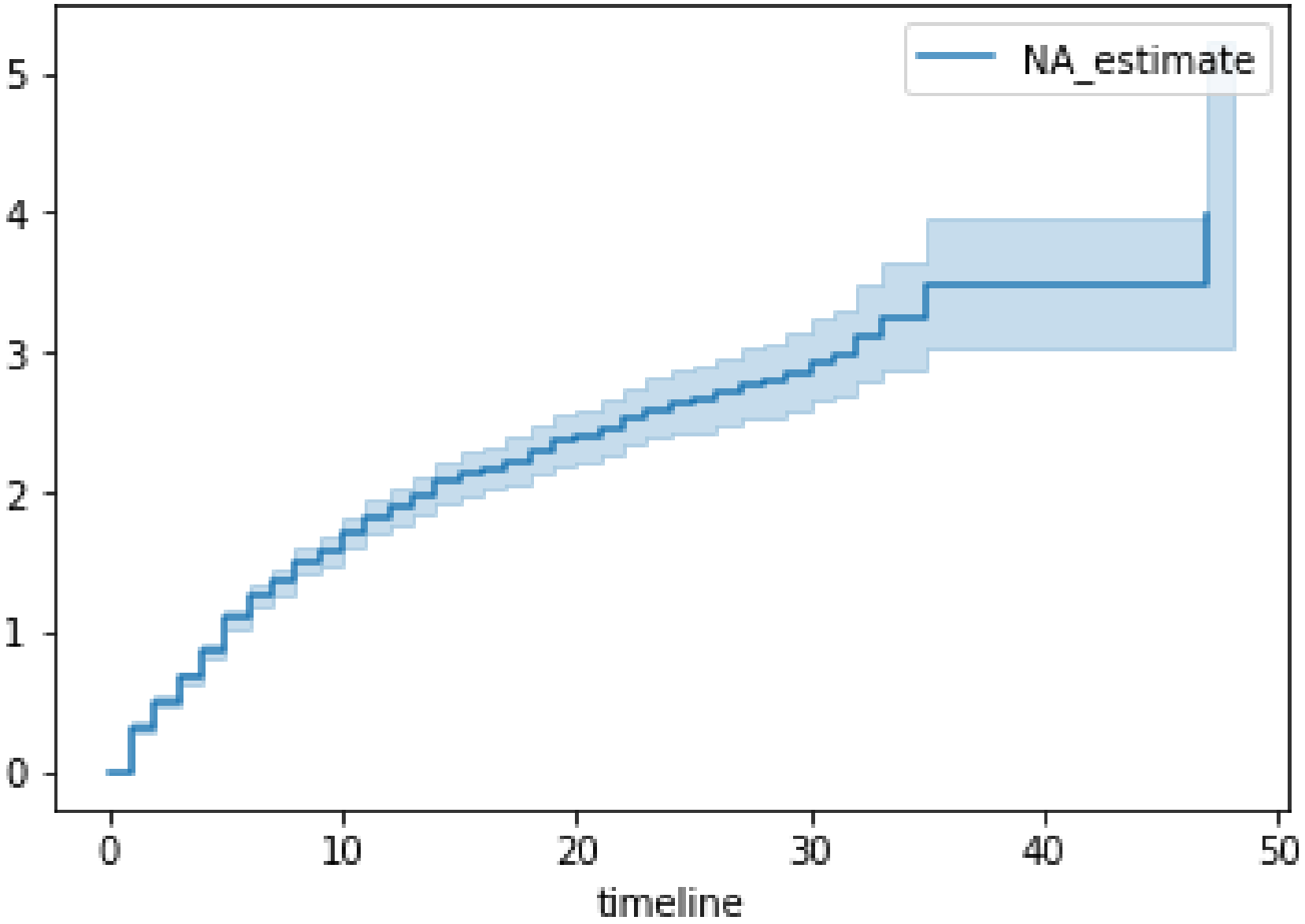
n_i : # of susceptible individuals

from lifelines import NelsonAalenFitter

```
naf = NelsonAalenFitter()
naf.fit(T,event_observed=E)
print(naf.cumulative_hazard_.head())
```

timeline	NA_estimate
0.0	0.000000
1.0	0.325912
2.0	0.507356
3.0	0.671251
4.0	0.869867

naf.plot()



Cumulative Hazard Function

```
naf.fit(T[dem], event_observed=E[dem], label="Democratic Regimes")
ax = naf.plot(loc=slice(0, 20))
naf.fit(T[~dem], event_observed=E[~dem], label="Non-democratic Regimes")
naf.plot(ax=ax, loc=slice(0, 20))
```