

Dynamic stock modeling in Python

Sebastiaan Deetman, Janneke van Oorschot & Tomer Fishman | MFA II – 03.05.2024



Universiteit
Leiden
The Netherlands

Discover the world at Leiden University

Content

1. Flow driven model
2. Stock driven model

Flow driven model



Universiteit
Leiden
The Netherlands

Discover the world at Leiden University

Flow driven model

- Excel file: MFA_II_tutorial_II.xlsx
- Python file: DSM_flow_driven.py

Lay-out python model flow driven model

1. Load required modules and Excel data

```
# %% 1. Load modules & data
import pandas as pd
import numpy as np
import scipy.stats
from os import chdir

chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/HFA/2022/Week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel('HFA_II_tutorial_II.xlsx', sheet_name='inflow_driven')

stock_flow_timeseries.set_index(['Year'])

time_max = stock_flow_timeseries.shape[0]

timesteps = np.arange(0, time_max)

# %% 2. create a single survival curve, if one wasn't supplied as input data
# comment / uncomment the snippet for the curve you want, and modify its parameters

# # Fixed lifetime survival curve
# curve_lifetime = 40
# curve_surv = np.zeros_like(timesteps)
# curve_surv[0:curve_lifetime] = 1

# # Weibull distributed survival curve
# curve_shape = 3
# curve_scale = 20
# curve_surv = scipy.stats.weibull_min.sf(timesteps, curve_shape, 0, curve_scale)

# # Normally distributed survival curve
# curve_mean = 39
# curve_sd = curve_mean / 3
# curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)

# # Geometrically distributed survival curve
# curve_depreciate = 0.05
# curve_surv = scipy.stats.geom.sf(timesteps, curve_depreciate)

# # Uniformly distributed survival curve
# curve_subtract = 0.1
# curve_surv = scipy.stats.uniform.sf(timesteps, loc=0, scale=(1 / curve_subtract))

# %% 3. create survival curve matrix
# create survival curve matrix with placeholder zeros
curve_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# populate the survival curve matrix with shifted curves, column by column using slices
for time in timesteps:
    curve_surv_matrix.loc[:, time] = curve_surv[0:time_max - time]

# %% 4. flow driven model

# create survival matrix with placeholder zeros
cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# multiply the inflow times the shifted curves to get the cohorts' behavior over time
for time in timesteps:
    cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['Inflow'].iloc[time]

# set row index to years instead of timesteps
cohort_surv_matrix.index = stock_flow_timeseries.index

# calculate flows & stocks using the cohort_surv_matrix
stock_flow_timeseries['Stock'] = cohort_surv_matrix.sum(axis=1)
stock_flow_timeseries['NAS'] = np.diff(stock_flow_timeseries['Stock'], prepend=0) # prepending 0 assumes no initial stock
stock_flow_timeseries['Outflow'] = stock_flow_timeseries['Inflow'] - stock_flow_timeseries['NAS']

# %% Export output data to Excel
cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
```

Input data

- Inflow of funky furniture per year
- From 1990 to 2050



test_data.xlsx • Last Modified: 3

File Home Insert Draw Page Layout Formulas Data Review

Cut Copy Format Painter Clipboard Font Alignment

F9 : fx

A	B	C	D	E	F	G	H
1	Year	Stock	Inflow	Outflow			
2	1990		13				
3	1991		14				
4	1992		12				
5	1993		10				
6	1994		7				
7	1995		20				
8	1996		22				
9	1997		23				
10	1998		28				
11	1999		30				
12	2000		14				
13	2001		13				
14	2002		20				
15	2003		22				
16	2004		30				
17	2005		15				
18	2006		30				
19	2007		33				
20	2008		28				
21	2009		29				
22	2010		15				
23	2011		27				
24	2012		30				
25	2013		32				
26	2014		33				
27	2015		34				
28	2016		36				
29	2017		33				
30	2018		38				
31	2019		39				
32	2020		33				

Lay-out python model flow driven model

1. Load required modules and Excel data

2. Create a single survival curve

```
# %% 1. Load modules & data
import pandas as pd
import numpy as np
import scipy.stats
from os import chdir

chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/HFA/2022/Week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel('HFA_II_tutorial_II.xlsx', sheet_name='inflow_driven')

stock_flow_timeseries = stock_flow_timeseries.set_index(['Year'])

time_max = stock_flow_timeseries.shape[0]

timesteps = np.arange(0, time_max)

# %% 2. create a single survival curve, if one wasn't supplied as input data
# comment / uncomment the snippet for the curve you want, and modify its parameters

# # Fixed lifetime survival curve
# curve_lifetime = 40
# curve_surv = np.zeros_like(timesteps)
# curve_surv[0:curve_lifetime] = 1

# # Weibull distributed survival curve
# curve_shape = 3
# curve_scale = 20
# curve_surv = scipy.stats.weibull_min.sf(timesteps, curve_shape, 0, curve_scale)

# # Normally distributed survival curve
# curve_mean = 39
# curve_sd = curve_mean / 3
# curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)

# # Geometrically distributed survival curve
# curve_depreciate = 0.05
# curve_surv = scipy.stats.geom.sf(timesteps, curve_depreciate)

# # Uniformly distributed survival curve
# curve_subtract = 0.1
# curve_surv = scipy.stats.uniform.sf(timesteps, loc=0, scale=(1 / curve_subtract))

# %% 3. create survival curve matrix
# create survival curve matrix with placeholder zeros
curve_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# populate the survival curve matrix with shifted curves, column by column using slices
for time in timesteps:
    curve_surv_matrix.loc[time:, time] = curve_surv[0:time_max - time]

# %% 4. flow driven model

# create survival matrix with placeholder zeros
cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# multiply the inflow times the shifted curves to get the cohorts' behavior over time
for time in timesteps:
    cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['Inflow'].iloc[time]

# set row index to years instead of timesteps
cohort_surv_matrix.index = stock_flow_timeseries.index

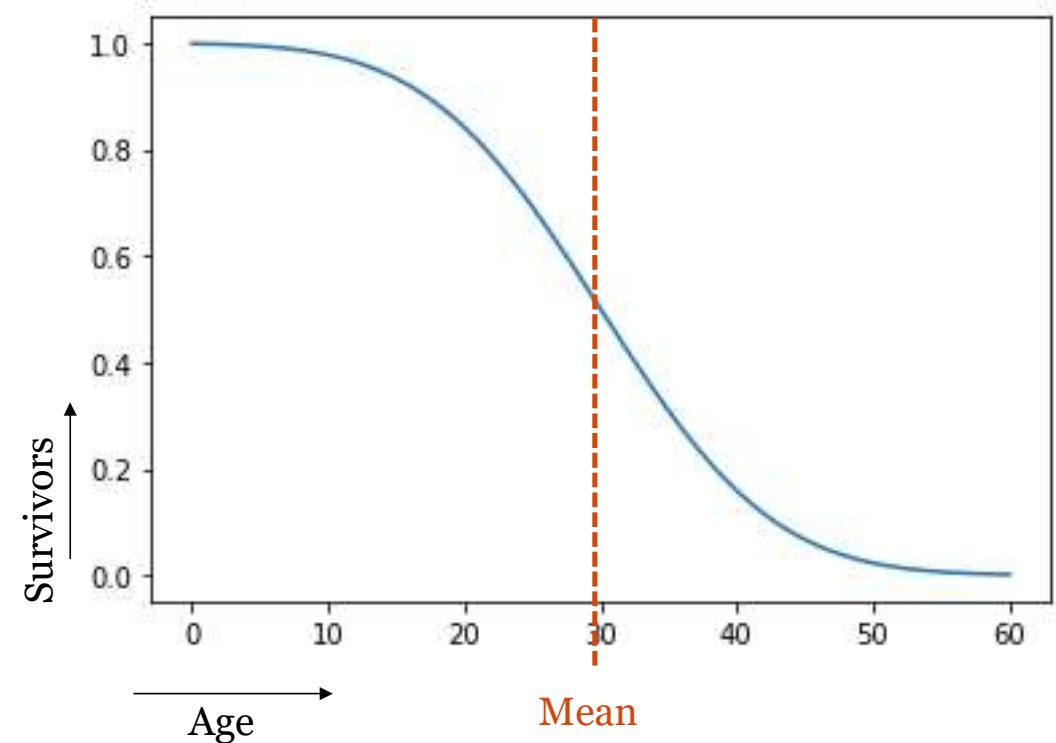
# calculate flows & stocks using the cohort_surv_matrix
stock_flow_timeseries['Stock'] = cohort_surv_matrix.sum(axis=1)
stock_flow_timeseries['NAS'] = np.diff(stock_flow_timeseries['Stock'], prepend=0) # prepending 0 assumes no initial stock
stock_flow_timeseries['Outflow'] = stock_flow_timeseries['Inflow'] - stock_flow_timeseries['NAS']

# %% Export output data to Excel
cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
```

Create/Load survival curve

- E.g. Mean lifetime = 30 years, standard deviation = 10

flow_driven_model.xlsx



age (years):	survival curve:
0	0.999
1	0.998
2	0.997
3	0.997
4	0.995
5	0.994
6	0.992
7	0.989
8	0.986
9	0.982
10	0.977
11	0.971
12	0.964
13	0.955
14	0.945
15	0.933
16	0.919
17	0.903
18	0.885
19	0.864
20	0.841

Lay-out python model flow driven model

1. Load required modules and Excel data

2. Create a single survival curve

3. Create a survival curve matrix

```
# %% 1. Load modules & data
import pandas as pd
import numpy as np
import scipy.stats
from os import chdir

chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/HFA/2022/Week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel('HFA_II_tutorial_II.xlsx', sheet_name='inflow_driven')
stock_flow_timeseries.set_index(['Year'])

time_max = stock_flow_timeseries.shape[0]
timesteps = np.arange(0, time_max)

# %% 2. create a single survival curve, if one wasn't supplied as input data
# comment / uncomment the snippet for the curve you want, and modify its parameters

## Fixed lifetime survival curve
# curve_lifetime = 40
# curve_surv = np.zeros_like(timesteps)
# curve_surv[0:curve_lifetime] = 1

## Weibull distributed survival curve
# curve_shape = 2
# curve_scale = 20
# curve_surv = scipy.stats.weibull_min.sf(timesteps, curve_shape, 0, curve_scale)

## Normally distributed survival curve
# curve_mean = 39
# curve_sd = curve_mean / 3
# curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)

## Geometrically distributed survival curve
# curve_depreciate = 0.05
# curve_surv = scipy.stats.geom.sf(timesteps, curve_depreciate)

## Uniformly distributed survival curve
# curve_subtract = 0.1
# curve_surv = scipy.stats.uniform.sf(timesteps, loc=0, scale=(1 / curve_subtract))

# %% 3. create survival curve matrix
# create survival curve matrix with placeholder zeros
curve_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# populate the survival curve matrix with shifted curves, column by column using slices
for time in timesteps:
    curve_surv_matrix.loc[:, time] = curve_surv[0:time_max - time]

# %% 4. flow driven model

# create survival matrix with placeholder zeros
cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# multiply the inflow times the shifted curves to get the cohorts' behavior over time
for time in timesteps:
    cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['Inflow'].iloc[time]

# set row index to years instead of timesteps
cohort_surv_matrix.index = stock_flow_timeseries.index

# calculate flows & stocks using the cohort_surv_matrix
stock_flow_timeseries['Stock'] = cohort_surv_matrix.sum(axis=1)
stock_flow_timeseries['NAS'] = np.diff(stock_flow_timeseries['Stock'], prepend=0) # prepending 0 assumes no initial stock
stock_flow_timeseries['Outflow'] = stock_flow_timeseries['Inflow'] - stock_flow_timeseries['NAS']

# %% Export output data to Excel
cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
```

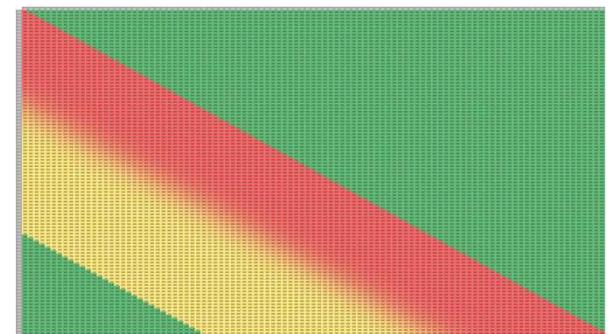
Create survival curve matrix

cohorts →

Index	0	1	2	3	4	5	6	7	8	9	10
0	0.99865	0	0	0	0	0	0	0	0	0	0
1	0.998134	0.99865	0	0	0	0	0	0	0	0	0
2	0.997445	0.998134	0.99865	0	0	0	0	0	0	0	0
3	0.996533	0.997445	0.998134	0.99865	0	0	0	0	0	0	0
4	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0	0	0	0
5	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0	0	0
6	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0	0
7	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0
8	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
9	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
10	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0
11	0.9771283	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134
12	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445
13	0.955435	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339	0.996533
14	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379	0.995339
15	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276	0.991802	0.99379
16	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276	0.991802
17	0.9032	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986697	0.989276

Format Resize Background color Column min/max Save and Close Close

stock_driven_model_surv.xlsx



$$stock(y) = \sum_{t=y_0}^y [inflow(t) \times sf(y - t)]$$

Lay-out python model flow driven model

1. Load required modules and Excel data

2. Create a single survival curve

3. Create a survival curve matrix

4. Calculate stocks and flows

5. Export results to Excel

```
# %% 1. Load modules & data
import pandas as pd
import numpy as np
import scipy.stats
from os import chdir

chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/HFA/2022/Week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel('HFA_II_tutorial_II.xlsx', sheet_name='inflow_driven')
stock_flow_timeseries.set_index(['Year'])

time_max = stock_flow_timeseries.shape[0]
timesteps = np.arange(0, time_max)

# %% 2. create a single survival curve, if one wasn't supplied as input data
# comment / uncomment the snippet for the curve you want, and modify its parameters

## Fixed lifetime survival curve
# curve_lifetime = 40
# curve_surv = np.zeros_like(timesteps)
# curve_surv[0:curve_lifetime] = 1

## Weibull distributed survival curve
# curve_shape = 2
# curve_scale = 20
# curve_surv = scipy.stats.weibull_min.sf(timesteps, curve_shape, 0, curve_scale)

## Normally distributed survival curve
# curve_mean = 39
# curve_sd = curve_mean / 3
# curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)

## Geometrically distributed survival curve
# curve_depreciate = 0.05
# curve_surv = scipy.stats.geom.sf(timesteps, curve_depreciate)

## Uniformly distributed survival curve
# curve_subtract = 0.1
# curve_surv = scipy.stats.uniform.sf(timesteps, loc=0, scale=(1 / curve_subtract))

# %% 3. create survival curve matrix
# create survival curve matrix with placeholder zeros
curve_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# populate the survival curve matrix with shifted curves, column by column using slices
for time in timesteps:
    curve_surv_matrix.loc[:, time] = curve_surv[0:time_max - time]

# %% 4. flow driven model

# create survival matrix with placeholder zeros
cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# multiply the inflow times the shifted curves to get the cohorts' behavior over time
for time in timesteps:
    cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['Inflow'].iloc[time]

# set row index to years instead of timesteps
cohort_surv_matrix.index = stock_flow_timeseries.index

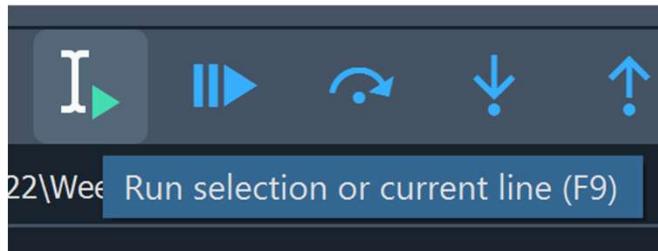
# calculate flows & stocks using the cohort_surv_matrix
stock_flow_timeseries['Stock'] = cohort_surv_matrix.sum(axis=1)
stock_flow_timeseries['NAS'] = np.diff(stock_flow_timeseries['Stock'], prepend=0) # prepending 0 assumes no initial stock
stock_flow_timeseries['Outflow'] = stock_flow_timeseries['Inflow'] - stock_flow_timeseries['NAS']

# %% Export output data to Excel
cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
```

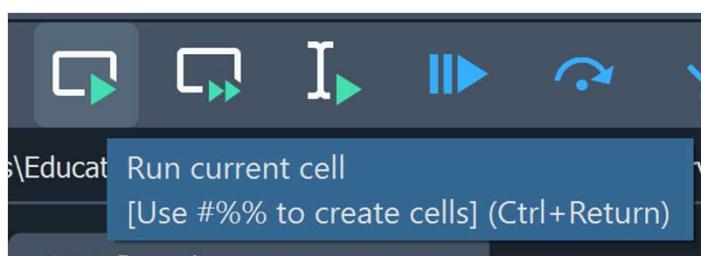
Let's run the model line by line

In case you wonder:

1. You can use the button illustrated below, or F9, to run the line of code your cursor is on, or a selection of code



2. You can use the button illustrated below, or Ctrl+Return, to run a cell (demarcated with#%%)



1. Load modules & data

- What happens in line 16 & 19?

```
10 # % 1. Load modules & data
11 import pandas as pd
12 import numpy as np
13 import scipy.stats
14 from os import chdir
15
16 chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/MFA/2022/Week_2')
17
18 # Load input data, inflow-driven model:
19 stock_flow_timeseries = pd.read_excel(r'MFA_II_tutorial_II.xlsx', sheet_name='inflow_driven')
20
```

- And in line 22?

```
22 stock_flow_timeseries = stock_flow_timeseries.set_index(['Year'])
23
```

1. Load modules & data

1. Line 22:

- Run the line, what happened?



Index	Year	Stock	Inflow	Outflow
0	1990	20	nan	nan
1	1991	21	nan	nan
2	1992	30	nan	nan
3	1993	39	nan	nan
4	1994	45	nan	nan
5	1995	50	nan	nan
6	1996	51	nan	nan
7	1997	52	nan	nan
8	1998	100	nan	nan
9	1999	113	nan	nan
10	2000	126	nan	nan
11	2001	130	nan	nan

Year	Stock	Inflow	Outflow
1990	20	nan	nan
1991	21	nan	nan
1992	30	nan	nan
1993	39	nan	nan
1994	45	nan	nan
1995	50	nan	nan
1996	51	nan	nan
1997	52	nan	nan
1998	100	nan	nan
1999	113	nan	nan
2000	126	nan	nan
2001	130	nan	nan

1. Load modules & data

1. Line 24-26

- What does time_max and timesteps refer to?

```
23  
24     time_max = stock_flow_timeseries.shape[0]  
25  
26     timesteps = np.arange(0, time_max)  
27
```

Overview			
time_max	int	1	61
timesteps	Array of int32	(61,)	[0 1 2 ... 58 59 60]

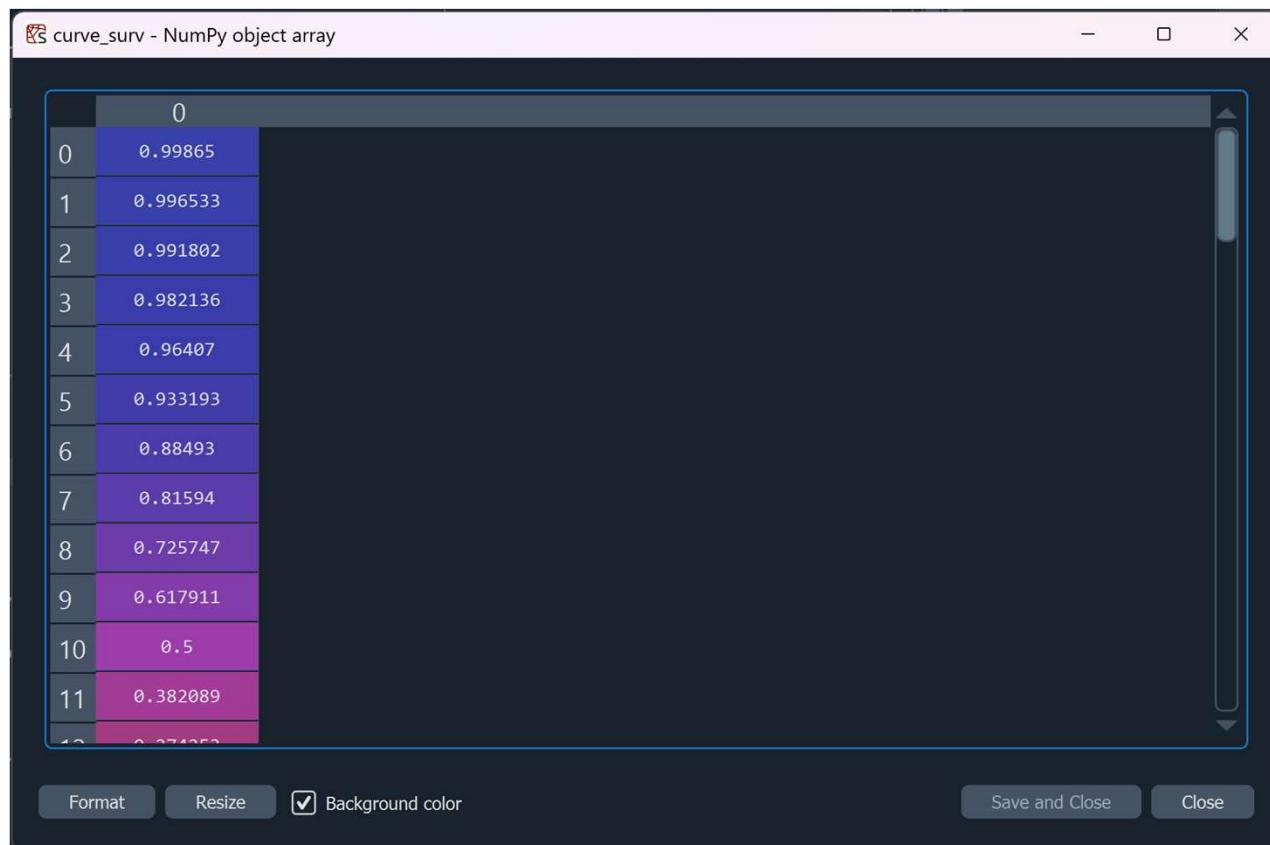
2. Create a single survival curve

1. Line 44-46

- We assume a normally distributed lifespan with a lifetime of 30 and a mean of 30/3.
- Run line 44,45,46. Explore the output in variable explorer, what does the data present? What does each row present?

```
42
43     #Normally distributed survival curve
44     curve_mean = 30
45     curve_sd = curve_mean / 3
46     curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)
```

2. Create a single survival curve



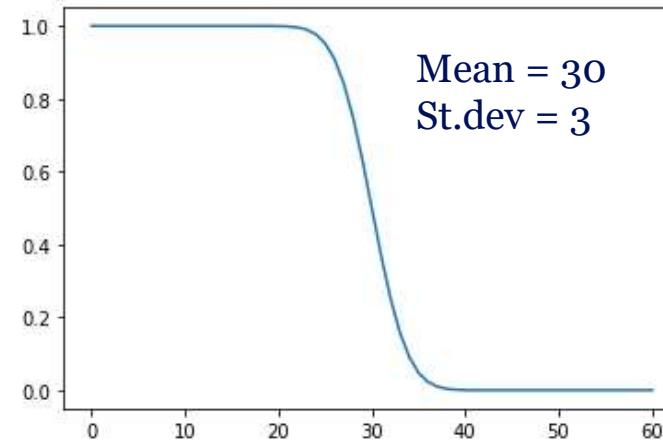
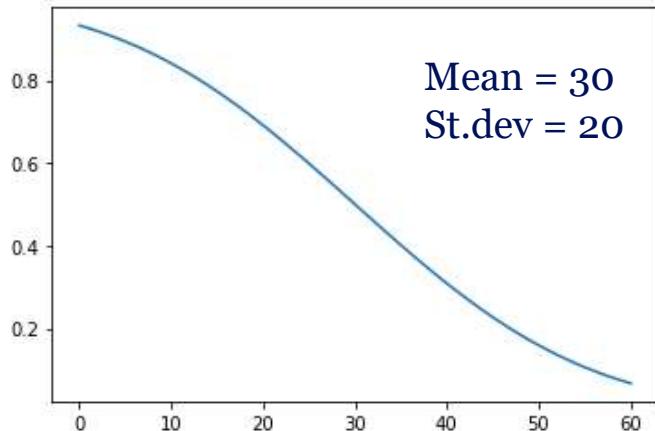
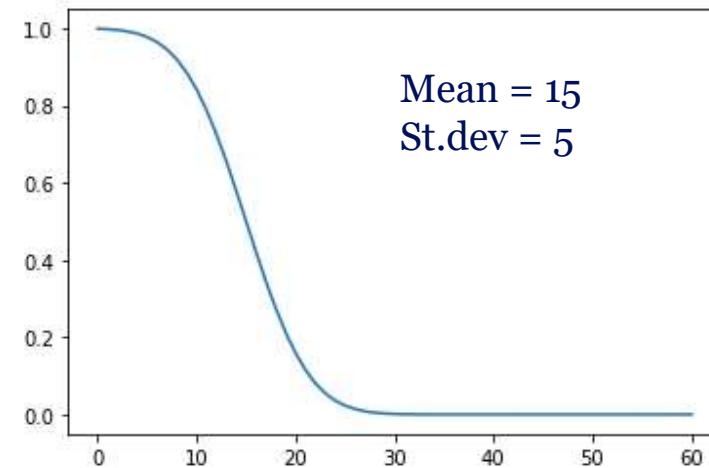
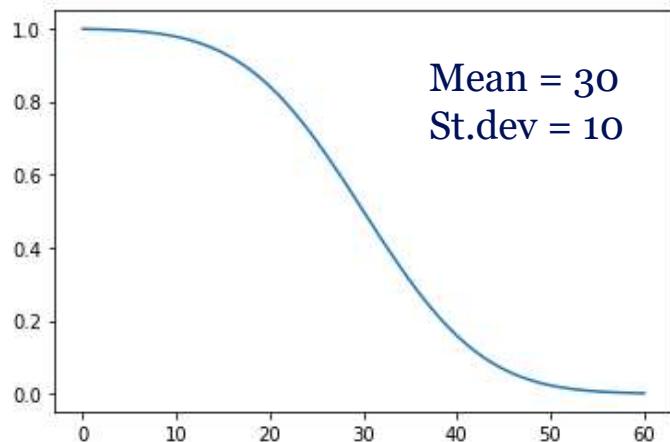
2. Create a single survival curve

1. Line 47-48

- Run line 47 and 48, explore the results. What happens when you change the curve_mean to 15? What if you change the curve_sd to 3? And to 20?
- Change the values back as illustrated below

```
42
43     #Normally distributed survival curve
44     curve_mean = 30
45     curve_sd = curve_mean / 3
46     curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)
47     plt.plot(curve_surv)
48     plt.show()
49
```

2. Create a single survival curve



3. Create survival curve matrix

1. Run line 67. What do the rows and columns present?

```
65  
66     # create survival curve  
67     curve_surv_matrix = pd.D  
68
```

The screenshot shows a Jupyter Notebook interface. On the left, a code cell contains the following Python code:

```
65  
66     # create survival curve  
67     curve_surv_matrix = pd.D  
68
```

To the right of the code cell is a DataFrame viewer window titled "curve_surv_matrix - DataFrame". The window displays a 11x11 grid of zeros. The columns are labeled from 0 to 10, and the rows are labeled from 0 to 10. The DataFrame has a "Index" column on the left.

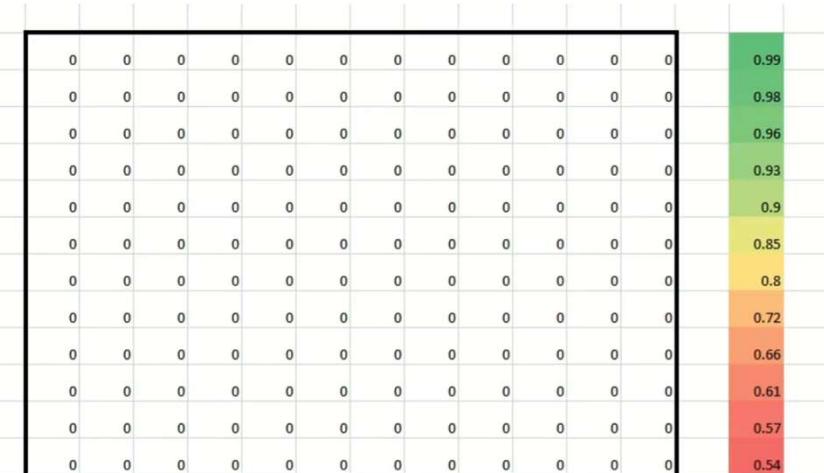
Index	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0

At the bottom of the DataFrame viewer, there are buttons for "Format", "Resize", "Background color" (checked), "Column min/max" (checked), "Save and Close", and "Close".

3. Create survival curve matrix

1. Now we need to fill the matrix with the shifted survival curves
2. Let's start with timestep 0.

```
68  
69      # populate the survival curve matrix with shifted curves, column by column using slices  
70      for time in timesteps:  
71          curve_surv_matrix.loc[time:, time] = curve_surv[0:time_max - time]  
72
```



3. Create survival curve matrix

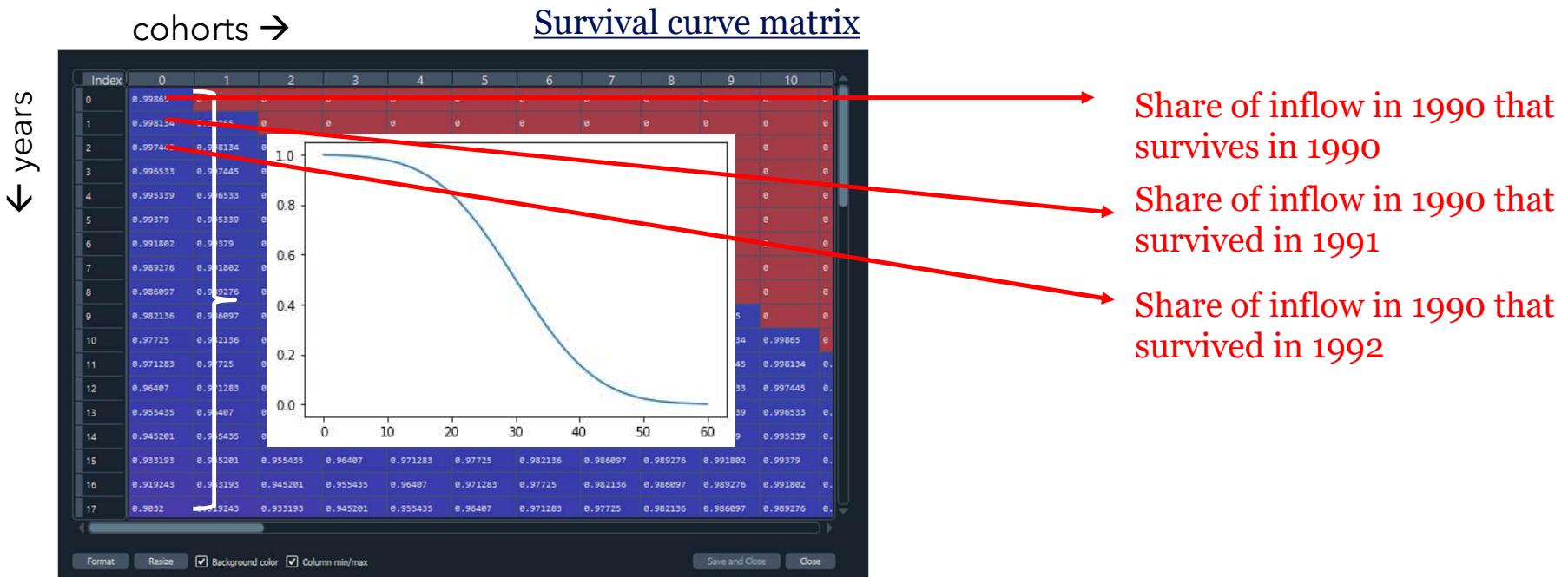
1. Let's fill the matrix with the shifted survival curves
2. Instead of filling each of the timesteps [0,1,2,3...60], we can use a for loop
3. Run line 70 & 71 and explore the results in Variable explorer.

```
curve_surv_matrix.loc[0:, 0] = curve_surv[0:time_max - 0]
```

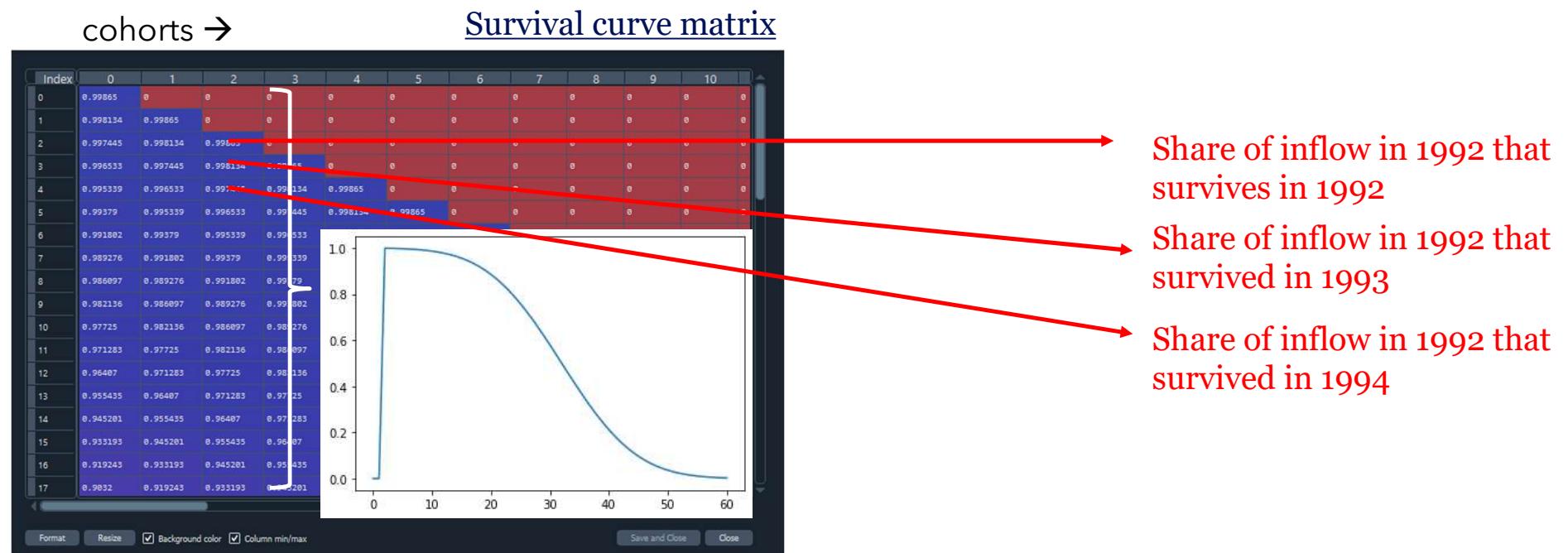


```
68  
69      # populate the survival curve matrix with shifted curves, column by column using slices  
70      for time in timesteps:  
71          curve_surv_matrix.loc[time:, time] = curve_surv[0:time_max - time]  
72
```

3. Create survival curve matrix



3. Create survival curve matrix



4. Flow driven model

1. Run line 76.

What is the difference between the cohort_surv_matrix (line 76) and the curve_surv_matrix (line 67)?

```
75      # create survival matrix with placeholder zeros  
76      cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)  
77
```

2. Select line 80 and 81 and run. What happened here?
What do the rows and columns present?

```
78      # multiply the inflow times the shifted curves to get the cohorts' behavior over time  
79  
80      for time in timesteps:  
81          cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['inflow'].iloc[time]  
82
```

3. Change the index to years instead of timesteps

```
82  
83      # set row index to years instead of timesteps  
84      cohort_surv_matrix.index = stock_flow_timeseries.index  
85
```

4. Flow driven model

$$stock(y) = \sum_{t=y_0}^y [inflow(t) \times sf(y-t)]$$

Stock flow timeseries

Year	Stock	Inflow	Outflow
1990	nan	13	nan
1991	nan	14	nan
1992	nan	12	nan
1993	nan	10	nan
1994	nan	7	nan
1995	nan	20	nan
1996	nan	22	nan
1997	nan	23	nan
1998	nan	28	nan
1999	nan	30	nan
2000	nan	14	nan
2001	nan	12	nan

Survival curve matrix

Index	0	1	2	3	4	5	6	7
0	0.99865	0	0	0	0	0	0	0
1	0.998134	0.99865	0	0	0	0	0	0
2	0.997445	0.998134	0.99865	0	0	0	0	0
3	0.996533	0.997445	0.998134	0.99865	0	0	0	0
4	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0
5	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
6	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
7	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865
8	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134
9	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445
10	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533
11	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339
12	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379
13	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882
14	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276
15	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897
16	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136
17	0.9032	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725

Cohort survival matrix

Year	0	1	2	3	4	5	6	7	8	9
1990	12.9825	0	0	0	0	0	0	0	0	0
1991	12.9757	13.9811	0	0	0	0	0	0	0	0
1992	12.9668	13.9739	11.9838	0	0	0	0	0	0	0
1993	12.9549	13.9642	11.9776	9.9865	0	0	0	0	0	0
1994	12.9394	13.9515	11.9693	9.98134	6.99055	0	0	0	0	0
1995	12.9193	13.9347	11.9584	9.97445	6.98694	19.973	0	0	0	0
1996	12.8934	13.9131	11.9441	9.96533	6.98211	19.9627	21.9703	0	0	0
1997	12.8606	13.8852	11.9255	9.95339	6.97573	19.9489	21.959	22.969	0	0
1998	12.8193	13.8499	11.9016	9.9379	6.96737	19.9307	21.9438	22.9571	27.9622	0
1999	12.7678	13.8054	11.8713	9.91802	6.95653	19.9068	21.9237	22.9412	27.9478	29.9595
2000	12.7042	13.7499	11.8332	9.89276	6.94262	19.8758	21.8975	22.9203	27.9285	29.944
2001	12.6267	13.6815	11.7856	9.86097	6.92493	19.836	21.8634	22.8928	27.9029	29.9233
2002	12.5329	13.598	11.727	9.82136	6.90268	19.7855	21.8197	22.8572	27.8695	29.896
2003	12.4206	13.497	11.6554	9.7725	6.87495	19.7219	21.7641	22.8115	27.8261	29.8602

Discover the world at Leiden University

4. Flow driven model

$$stock(y) = \sum_{t=y_0}^y [inflow(t) \times sf(y - t)]$$

Stock flow timeseries

Year	Stock	Inflow	Outflow
1990	nan	13	nan
1991	nan	14	nan
1992	nan	12	nan
1993	nan	10	nan
1994	nan	7	nan
1995	nan	20	nan
1996	nan	22	nan
1997	nan	23	nan
1998	nan	28	nan
1999	nan	30	nan
2000	nan	14	nan
2001	nan	12	nan

Survival curve matrix

Index	0	1	2	3	4	5	6	7
0	0.99865	0	0	0	0	0	0	0
1	0.998134	0.99865	0	0	0	0	0	0
2	0.997445	0.998134	0.99865	0	0	0	0	0
3	0.996533	0.997445	0.998134	0.99865	0	0	0	0
4	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0
5	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
6	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
7	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865
8	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134
9	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445
10	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533
11	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339
12	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379
13	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276	0.991882
14	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897	0.989276
15	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986897
16	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136
17	0.9032	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725

Cohort survival matrix

Year	0	1	2	3	4	5	6	7	8	9
1990	12.9825	0	0	0	0	0	0	0	0	0
1991	12.9757	13.9811	0	0	0	0	0	0	0	0
1992	12.9668	13.9739	11.9838	0	0	0	0	0	0	0
1993	12.9549	13.9642	11.9776	9.9865	0	0	0	0	0	0
1994	12.9394	13.9515	11.9693	9.98134	6.99055	0	0	0	0	0
1995	12.9193	13.9347	11.9584	9.97445	6.98694	19.973	0	0	0	0
1996	12.8934	13.9131	11.9441	9.96533	6.98211	19.9627	21.9703	0	0	0
1997	12.8606	13.8852	11.9255	9.95339	6.97573	19.9489	21.959	22.969	0	0
1998	12.8193	13.8499	11.9016	9.9379	6.96737	19.9307	21.9438	22.9571	27.9622	0
1999	12.7678	13.8054	11.8713	9.91802	6.95653	19.9068	21.9237	22.9412	27.9478	29.9595
2000	12.7042	13.7499	11.8332	9.89276	6.94262	19.8758	21.8975	22.9203	27.9285	29.944
2001	12.6267	13.6815	11.7856	9.86097	6.92493	19.836	21.8634	22.8928	27.9029	29.9233
2002	12.5329	13.598	11.727	9.82136	6.90268	19.7855	21.8197	22.8572	27.8695	29.896
2003	12.4206	13.497	11.6554	9.7725	6.87495	19.7219	21.7641	22.8115	27.8261	29.8602

Discover the world at Leiden University

4. Flow driven model

- Inflow x the shifted curves to get the cohorts' behavior over time

Stock flow timeseries

Year	Stock	Inflow	Outflow
1990	nan	13	nan
1991	nan	14	nan
1992	nan	12	nan
1993	nan	10	nan
1994	nan	7	nan
1995	nan	20	nan
1996	nan	22	nan
1997	nan	23	nan
1998	nan	28	nan
1999	nan	30	nan
2000	nan	14	nan
2001	nan	12	nan

Survival curve matrix

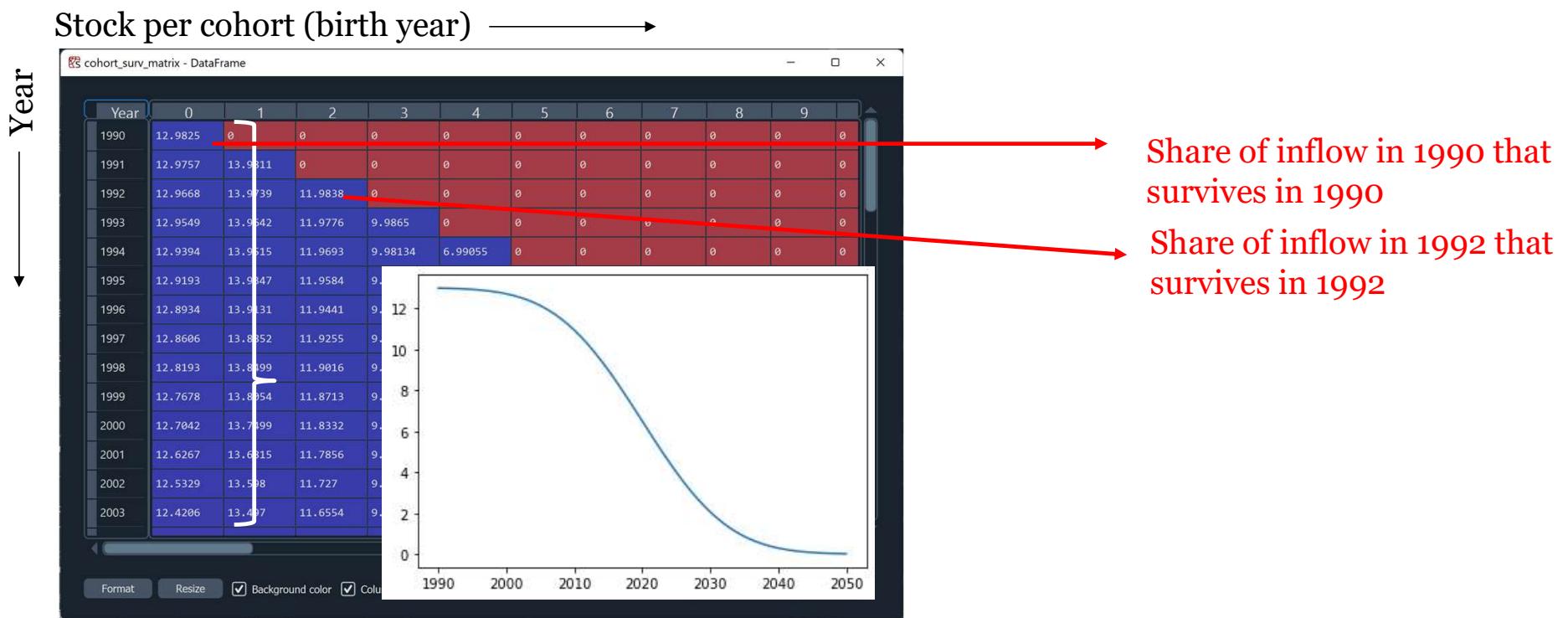
Index	0	1	2	3	4	5	6	7
0	0.99865	0	0	0	0	0	0	0
1	0.998134	0.99865	0	0	0	0	0	0
2	0.997445	0.998134	0.99865	0	0	0	0	0
3	0.996533	0.997445	0.998134	0.99865	0	0	0	0
4	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0
5	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
6	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
7	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865
8	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445	0.998134
9	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533	0.997445
10	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339	0.996533
11	0.972383	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379	0.995339
12	0.96407	0.972383	0.97725	0.982136	0.986897	0.989276	0.991882	0.99379
13	0.955435	0.96407	0.972383	0.97725	0.982136	0.986897	0.989276	0.991882
14	0.945201	0.955435	0.96407	0.972383	0.97725	0.982136	0.986897	0.989276
15	0.933193	0.945201	0.955435	0.96407	0.972383	0.97725	0.982136	0.986897
16	0.919243	0.933193	0.945201	0.955435	0.96407	0.972383	0.97725	0.982136
17	0.9032	0.919243	0.933193	0.945201	0.955435	0.96407	0.972383	0.97725

Cohort survival matrix

Year	0	1	2	3	4	5	6	7	8	9
1990	12.9825	0	0	0	0	0	0	0	0	0
1991	12.9757	13.9811	0	0	0	0	0	0	0	0
1992	12.9668	13.9739	11.9838	0	0	0	0	0	0	0
1993	12.9549	13.9642	11.9776	9.9865	0	0	0	0	0	0
1994	12.9394	13.9515	11.9693	9.98134	6.99055	0	0	0	0	0
1995	12.9193	13.9347	11.9584	9.97445	6.98694	19.973	0	0	0	0
1996	12.8934	13.9131	11.9441	9.96533	6.98211	19.9627	21.9703	0	0	0
1997	12.8606	13.8852	11.9255	9.95339	6.97573	19.9489	21.959	22.969	0	0
1998	12.8193	13.8499	11.9016	9.9379	6.96737	19.9307	21.9438	22.9571	27.9622	0
1999	12.7678	13.8054	11.8713	9.91802	6.95653	19.9068	21.9237	22.9412	27.9478	29.9595
2000	12.7042	13.7499	11.8332	9.89276	6.94262	19.8758	21.8975	22.9203	27.9285	29.944
2001	12.6267	13.6815	11.7856	9.86097	6.92493	19.836	21.8634	22.8928	27.9029	29.9233
2002	12.5329	13.598	11.727	9.82136	6.90268	19.7855	21.8197	22.8572	27.8695	29.896
2003	12.4206	13.497	11.6554	9.7725	6.87495	19.7219	21.7641	22.8115	27.8261	29.8602

4. Flow driven model

- Inflow x the shifted curves to get the cohorts' behavior over time



4. Flow driven model

Now we can...

- Calculate the stock:

$$\text{stock}(y) = \sum_{t=y_0}^y [\text{inflow}(t) \times sf(y - t)]$$

- Calculate the outflow:

$$NAS(y) = \text{stock}(y) - \text{stock}(y - 1)$$

$$\text{outflow}(y) = \text{inflow}(y) - NAS(y)$$

Cohort survival matrix

Year	0	1	2	3	4	5	6	7	8	9
1990	12.9825	0	0	0	0	0	0	0	0	0
1991	12.9757	13.9811	0	0	0	0	0	0	0	0
1992	12.9668	13.9739	11.9838	0	0	0	0	0	0	0
1993	12.9549	13.9642	11.9776	9.9865	0	0	0	0	0	0
1994	12.9394	13.9515	11.9693	9.98134	6.99055	0	0	0	0	0
1995	12.9193	13.9347	11.9584	9.97445	6.98694	19.973	0	0	0	0
1996	12.8934	13.9131	11.9441	9.96533	6.98211	19.9627	21.9703	0	0	0
1997	12.8666	13.8852	11.9255	9.95339	6.97573	19.9489	21.959	22.969	0	0
1998	12.8193	13.8499	11.9016	9.9379	6.96737	19.9387	21.9438	22.9571	27.9622	0
1999	12.7678	13.8054	11.8713	9.91802	6.95653	19.9068	21.9237	22.9412	27.9478	29.9595
2000	12.7042	13.7499	11.8332	9.89276	6.94262	19.8758	21.8975	22.9203	27.9285	29.944
2001	12.6267	13.6815	11.7856	9.86097	6.92493	19.836	21.8634	22.8928	27.9029	29.9233
2002	12.5329	13.598	11.727	9.82136	6.90268	19.7855	21.8197	22.8572	27.8695	29.896
2003	12.4206	13.497	11.6554	9.7725	6.87495	19.7219	21.7641	22.8115	27.8261	29.8602

↳ Stock (1990)

↳ Stock (1996)

4. Flow driven model

- And now in python:

```
86      # calculate flows & stocks using the cohort_surv_matrix
87      stock_flow_timeseries['stock'] = cohort_surv_matrix.sum(axis=1)
88      stock_flow_timeseries['nas'] = np.diff(stock_flow_timeseries['stock'], prepend=0) # prepending 0 assumes no initial stock
89      stock_flow_timeseries['outflow'] = stock_flow_timeseries['inflow'] - stock_flow_timeseries['nas']
90
```

- If you're happy – export data to Excel:

```
92      # %% Export output data to Excel
93
94      cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
95      stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
96
```

Stock driven model



Universiteit
Leiden
The Netherlands

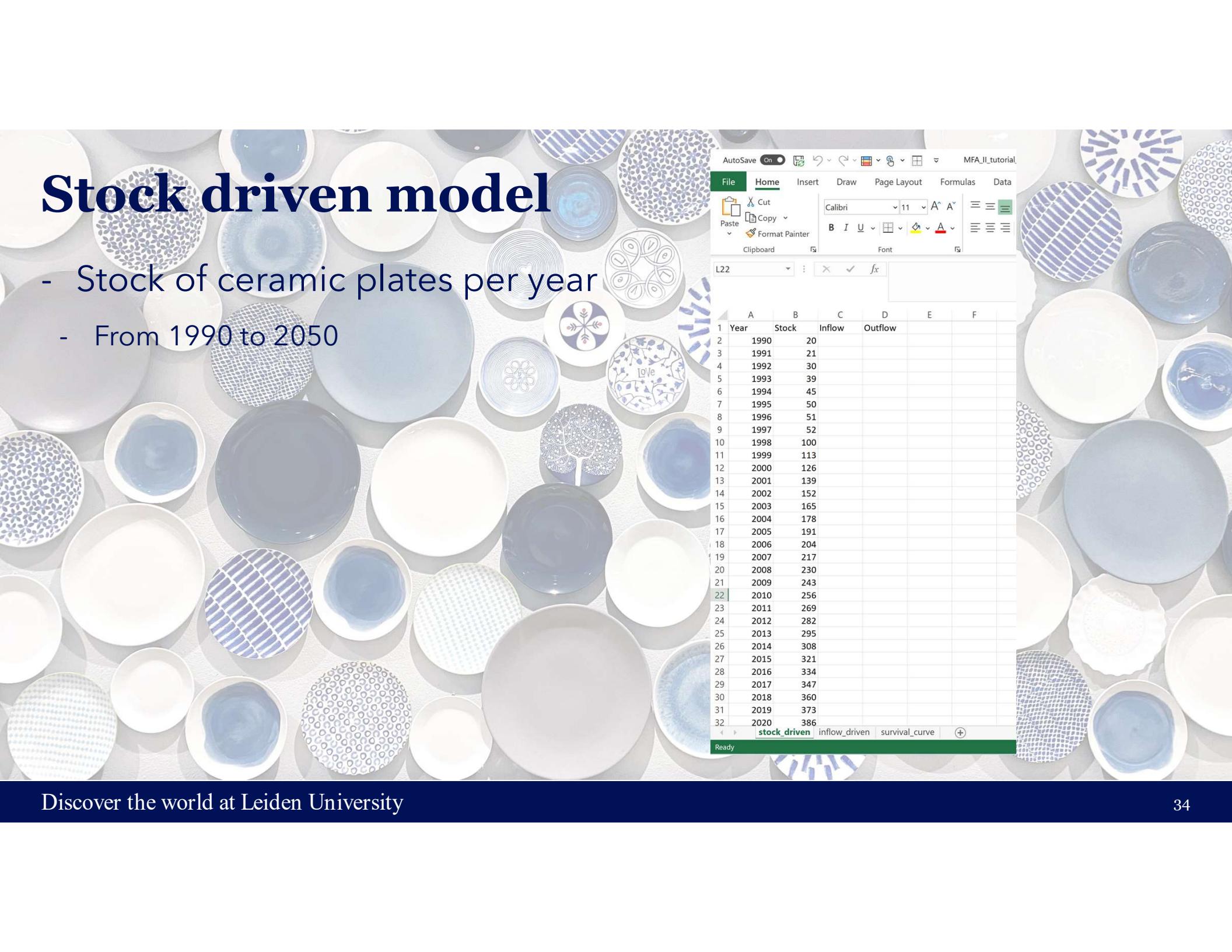
Discover the world at Leiden University

Stock driven model

- Excel file: MFA_II_tutorial_II.xlsx
- Python file: DSM_stock_driven.py

Stock driven model

- Stock of ceramic plates per year
- From 1990 to 2050



A Microsoft Excel spreadsheet titled "MFA_IITutorial" showing a stock driven model for ceramic plates from 1990 to 2020. The data is presented in a table with columns for Year, Stock, Inflow, and Outflow.

	A	B	C	D	E	F
1	Year	Stock	Inflow	Outflow		
2	1990	20				
3	1991	21				
4	1992	30				
5	1993	39				
6	1994	45				
7	1995	50				
8	1996	51				
9	1997	52				
10	1998	100				
11	1999	113				
12	2000	126				
13	2001	139				
14	2002	152				
15	2003	165				
16	2004	178				
17	2005	191				
18	2006	204				
19	2007	217				
20	2008	230				
21	2009	243				
22	2010	256				
23	2011	269				
24	2012	282				
25	2013	295				
26	2014	308				
27	2015	321				
28	2016	334				
29	2017	347				
30	2018	360				
31	2019	373				
32	2020	386				

Lay-out python model stock driven model

1. Load required modules and Excel data

2. Create a single survival curve

3. Create a survival curve matrix

4. Calculate stocks and flows

5. Export results to Excel

Almost the
same as
flow-model!
Except from

```
# NN 1. Load modules & data
import pandas as pd
import numpy as np
import scipy.stats
from os import chdir
import matplotlib.pyplot as plt

chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/NFA/2022/Week_2')

# Load input data, stock-driven model
stock_flow_timeseries = pd.read_excel(r'NFA_II_tutorial_II.xlsx', sheet_name='stock_driven')
stock_flow_timeseries.set_index(['Year'])

time_max = stock_flow_timeseries.shape[0]

timesteps = np.arange(0, time_max)

# NN 2. Create a single survival curve, if one wasn't supplied as input data
# comment / uncomment the snippet for the curve you want, and modify its parameters

## Fixed lifetime survival curve
# curve_lifetime = 40
# curve_surv = np.zeros_like(timesteps)
# curve_surv[timesteps <= curve_lifetime] = 1

## Weibull distributed survival curve
# curve_shape = 1.5
# curve_scale = 15
# curve_surv = scipy.stats.weibull_min.sf(timesteps, curve_shape, 0, curve_scale)

## Normally distributed survival curve
curve_mean = 30
curve_sd = curve_mean / 3
curve_surv = np.random.normal(sf(timesteps, loc=curve_mean, scale=curve_sd))
plt.plot(curve_surv)
plt.show()

## Geometrically distributed survival curve
# curve_deprecate = 0.05
# curve_surv = scipy.stats.geom.sf(timesteps, curve_deprecate)

## Uniformly distributed survival curve
# curve_subtract = 0.1
# curve_surv = scipy.stats.uniform.sf(timesteps, loc=0, scale=(1 / curve_subtract))

## Lognorm
# curve_mean = lognorm(0.01, 2.5)
# s = timesteps
# curve_surv = scipy.stats.lognorm.sf(s, s, loc = 0, scale = 1)
# curve_surv[0] = 1

## NN 3. Create survival curve matrix
# Create survival curve matrix with placeholder zeros
curve_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# populates the survival curve matrix with shifted curves, column by column using slices
for time in timesteps:
    curve_surv_matrix.loc[:, time] = curve_surv[0:(time_max - time)]
    curve_surv_matrix.loc[:, time] = curve_surv[0:(time_max - time)]

## NN 4. Stock driven model
# Create survival matrix with placeholder zeros
cohort_surv_matrix = pd.DataFrame(0, index=timesteps, columns=timesteps)

# iteratively calculate the inflow in stock_flow_timeseries, and
# multiply the inflow times the shifted curves to get the cohorts' behavior over time
for time in timesteps:
    stock_flow_timeseries['inflow'].loc[time] = (stock_flow_timeseries['stock'].loc[time, :].sum()) / curve_surv_matrix.loc[time, time]
    cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['inflow'].loc[time]
    cohort_surv_matrix.loc[:, time] = cohort_surv_matrix.loc[:, time].sum()

# set row index to years instead of timesteps
cohort_surv_matrix.index = stock_flow_timeseries.index

# calculate outflows and nas using the cohort_surv_matrix
stock_flow_timeseries['outflow'] = np.diff(stock_flow_timeseries['stock'], prepend=0) # prepending 0 assumes no initial stock
stock_flow_timeseries['outflow'] = stock_flow_timeseries['inflow'] - stock_flow_timeseries['nas']

## NN 5. Export output data to Excel
cohort_surv_matrix.to_excel('cohort_surv_matrix.xlsx')
stock_flow_timeseries.to_excel('stock_flow_timeseries.xlsx')
```

4. Calculate stocks & flows

- Let's move directly to cell 4 (cell 1,2,3 and 5 are identical to the flow driven model)
- We will iteratively calculate the inflow in the stock_flow_timeseries, and multiply the inflow with the shifted survival curve

```
83     for time in timesteps:  
84         stock_flow_timeseries['inflow'].iloc[time] = (stock_flow_timeseries['stock'].iloc[time] - cohort_surv_matrix.loc[time, :].sum()) / curve_surv_matrix.loc[time, time]  
85         cohort_surv_matrix.loc[:, time] = curve_surv_matrix.loc[:, time] * stock_flow_timeseries['inflow'].iloc[time]  
86
```

4. Calculate stocks & flows

- Let's try in python
- Calculate the inflow for timestep 0

```
stock_flow_timeseries['inflow'].iloc[0] = (stock_flow_timeseries['stock'].iloc[0] - cohort_surv_matrix.loc[0, :].sum()) / curve_surv_matrix.loc[0, 0]
```

$$inflow(y) = \frac{stock(y) - \sum_{t=y_0}^{y-1} [inflow(t) \times sf(y-t)]}{sf(0)}$$

4. Calculate stocks & flows

- Calculate inflow for timestep 0
- $y = 0$ (1990)

The image displays three data frames from a software interface:

- stock_flow_timeseries - DataFrame**: Shows a table with columns Year, Stock, Inflow, and Outflow. The row for 1990 has a Stock value of 20, highlighted with a red box. A white box contains the text "NAS = 20".
- curve_surv_matrix - DataFrame**: Shows a 12x8 matrix where the first column (Index 0) contains values like 0.99865, 0.998134, etc., all highlighted with red boxes. A white box contains the text "A small part of the inflow immediately becomes outflow".
- cohort_surv_matrix - DataFrame**: Shows a 12x8 matrix where every cell in the first row (Index 0) is highlighted with a red box. A white box contains the text "No survivors from previous years, because year = 0".

4. Calculate stocks & flows

- Calculate inflow for timestep 0
- $y = 0$ (1990)

$$inflow(y) = \frac{stock(y)}{sf(0)} - \sum_{t=y_0}^{y-1} [inflow(t) \times sf(y-t)]$$

$= 0$

stock_flow_timeseries - DataFrame

Year	Stock	Inflow	Outflow
1990	20	nan	nan
1991	21	nan	nan
1992	30	nan	nan
1993	39	nan	nan
1994	45	nan	nan
1995	50	nan	nan
1996	51	nan	nan
1997	52	nan	nan
1998	100	nan	nan
1999	113	nan	nan
2000	126	nan	nan
2001	120	nan	nan

Format Resize Background color Column min/max

curve_surv_matrix - DataFrame

Index	0	1	2	3	4	5	6	7
0	0.99865	0	0	0	0	0	0	0
1	0.998134	0.99865	0	0	0	0	0	0
2	0.997445	0.998134	0.99865	0	0	0	0	0
3	0.996533	0.997445	0.998134	0.99865	0	0	0	0
4	0.995339	0.996533	0.997445	0.998134	0.99865	0	0	0
5	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
6	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
7	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865
8	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134
9	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445
10	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533
11	0.973182	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339

Format Resize Background color Column min/max Save and Close Close

4. Calculate stocks & flows

- Calculate inflow for timestep 0
- $y = 0$

stock_flow_timeseries - DataFrame

Year	Stock	Inflow	Outflow
1990	20	20.027	nan
1991	21	nan	nan
1992	30	nan	nan
1993	39	nan	nan
1994	45	nan	nan
1995	50	nan	nan
1996	51	nan	nan
1997	52	nan	nan
1998	100	nan	nan
1999	113	nan	nan
2000	126	nan	nan
2001	139	nan	nan

Format Resize Background color Column min/max Save and Close Close

4. Calculate stocks & flows

- What does the line below do?

```
cohort_surv_matrix.loc[:, 0] = curve_surv_matrix.loc[:, 0] * stock_flow_timeseries['inflow'].iloc[0]
```

4. Calculate stocks & flows

- Calculate inflow for timestep 0

- $y = 0$
- Multiply the inflow times the shifted survival curve to get the cohorts' behavior over time

Year	0	1	2	3	4	5	6	7	8	9	10
1990	20	0	0	0	0	0	0	0	0	0	0
1991	19.9897	0	0	0	0	0	0	0	0	0	0
1992	19.9759	0	0	0	0	0	0	0	0	0	0
1993	19.9576	0	0	0	0	0	0	0	0	0	0
1994	19.9337	0	0	0	0	0	0	0	0	0	0
1995	19.9027	0	0	0	0	0	0	0	0	0	0
1996	19.8629	0	0	0	0	0	0	0	0	0	0
1997	19.8123	0	0	0	0	0	0	0	0	0	0
1998	19.7486	0	0	0	0	0	0	0	0	0	0
1999	19.6693	0	0	0	0	0	0	0	0	0	0
2000	19.5714	0	0	0	0	0	0	0	0	0	0
2001	0	0	0	0	0	0	0	0	0	0	0

4. Calculate stocks & flows

- Calculate inflow for timestep 1
- $y = 1$ (1991)

The screenshot shows three data frames in a software interface:

- stock_flow_timeseries - DataFrame**: A table with columns Year, Stock, Inflow, and Outflow. The Year column shows values from 1990 to 2001. The Stock column shows values 20, 21, 30, 39, 45, 50, 51, 52, 100, 113, 126, and 130. The first two rows (1990 and 1991) have Inflow and Outflow set to nan. The row for 1991 is highlighted with a red box around the value 21.
- curve_surv_matrix - DataFrame**: A table with columns Index (0 to 7) and rows Index (0 to 11). The matrix contains survival probabilities. The cell at index (1, 1) contains the value 0.99865, which is highlighted with a red box.
- cohort_surv_matrix - DataFrame**: A table with columns Year (1990 to 1992) and rows Year (1990 to 1992). The matrix contains survival probabilities. The cell at (1991, 1991) contains the value 19.9897, which is highlighted with a red box.

Checkboxes at the bottom of the interface include: Format, Resize, Background color, Column min/max, Save and Close, and Close.

4. Calculate stocks & flows

- Calculate inflow for timestep 1
- $y = 1$

stock_flow_timeseries - DataFrame

Year	Stock	Inflow	Outflow
1990	20	20.027	nan
1991	21	1.0117	nan
1992	30	nan	nan
1993	39	nan	nan
1994	45	nan	nan
1995	50	nan	nan
1996	51	nan	nan
1997	52	nan	nan
1998	100	nan	nan
1999	113	nan	nan
2000	126	nan	nan
2001	139	nan	nan

Format Resize Background color Column min/max Save and Close Close

4. Calculate stocks & flows

- Calculate inflow for timestep 1

- $y = 1$
- Multiply the inflow times the shifted survival curve to get the cohorts' behavior over time

- **And so on for $y = 2, 3, \dots$**

- Instead of manually calculating each timestep, let's use the for loop:
- Run line 83,84,85 and explore the results

Year	0	1	2	3	4	5	6	7	8	9
1990	20	0	0	0	0	0	0	0	0	0
1991	19.9897	1.01033	0	0	0	0	0	0	0	0
1992	19.9759	1.00981	0	0	0	0	0	0	0	0
1993	19.9576	1.00911	0	0	0	0	0	0	0	0
1994	19.9337	1.00819	0	0	0	0	0	0	0	0
1995	19.9027	1.00698	0	0	0	0	0	0	0	0
1996	19.8629	1.00542	0	0	0	0	0	0	0	0
1997	19.8123	1.0034	0	0	0	0	0	0	0	0
1998	19.7486	1.00085	0	0	0	0	0	0	0	0
1999	19.6693	0.997632	0	0	0	0	0	0	0	0
2000	19.5714	0.993625	0	0	0	0	0	0	0	0
2001	19.4510	0.989593	0	0	0	0	0	0	0	0

4. Calculate stocks & flows

- Calculate outflows:

$$NAS(y) = \text{stock}(y) - \text{stock}(y - 1)$$

$$\text{outflow}(y) = \text{inflow}(y) - NAS(y)$$

- What does prepend=0 do?

```
55  
90     # calculate outflows and nas using the cohort_surv_matrix  
91     stock_flow_timeseries['nas'] = np.diff(stock_flow_timeseries['stock'], prepend=0)  # prepending 0 assumes no initial stock  
92     stock_flow_timeseries['outflow'] = stock_flow_timeseries['inflow'] - stock_flow_timeseries['nas']  
93
```

4. Calculate stocks & flows

- Calculate outflows:

$$NAS(y) = \text{stock}(y) - \text{stock}(y - 1)$$

$$\text{outflow}(y) = \text{inflow}(y) - NAS(y)$$

stock_flow_timeseries - DataFrame

Year	Stock	Inflow	Outflow	NAS
1990	20	20.027	0.0270345	20
1991	21	1.0117	0.0116979	1
1992	30	9.02651	0.0265118	9
1993	39	9.03581	0.0358132	9
1994	45	6.04388	0.0438815	6
1995	50	5.05662	0.056623	5
1996	51	1.06861	0.068614	1
1997	52	1.0884	0.0883962	1
1998	100	48.1763	0.176324	48
1999	113	13.1831	0.183084	13
2000	126	13.2334	0.233369	13
2001	129	13.205	0.204092	13

Format Resize Background color Column min/max Save and Close Close



**Universiteit
Leiden**
The Netherlands

Discover the world at Leiden University