Dynamic stock modeling in Python

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

Universiteit Leiden
The Netherlands

Content

1. Flow driven model
2. Stock driven model

Flow driven model

Universiteit
Leiden
The Netherlands

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

```python
# %% 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/MFA/2022/week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel(r'MFA_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)
```

```python
# %% 2. create a single survival curve, if one wasn't supplied as input data

# comment / uncomment the snipet 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 = 5
# curve_scale = 20
# 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 = 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))
```

```python
# %% 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]
```

```python
# %% 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']
```

```python
# %% 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

Year	Stock	Inflow	Outflow
1990		13	
1991		14	
1992		12	
1993		10	
1994		7	
1995		20	
1996		22	
1997		23	
1998		28	
1999		30	
2000		14	
2001		13	
2002		20	
2003		22	
2004		30	
2005		15	
2006		30	
2007		33	
2008		28	
2009		29	
2010		15	
2011		27	
2012		30	
2013		32	
2014		33	
2015		34	
2016		36	
2017		33	
2018		38	
2019		39	
2020		33	

stock_driven | inflow_driven

Lay-out python model flow driven model

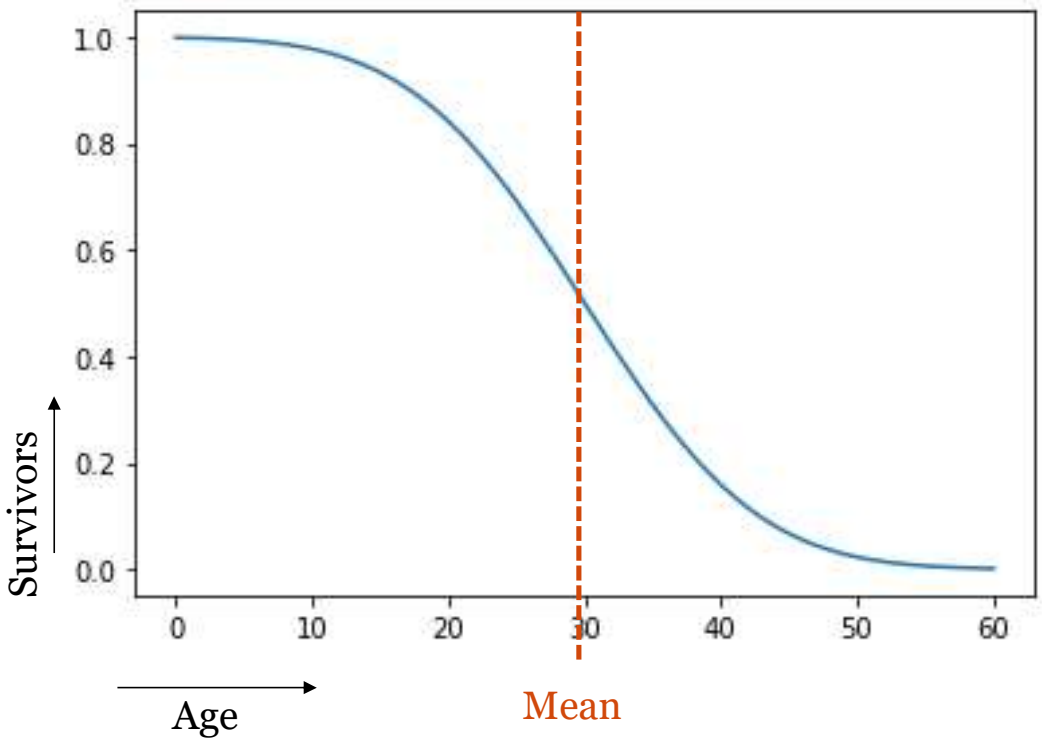
1. Load required modules and Excel data

2. Create a single survival curve

```python
# %% 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/MFA/2022/week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel(r'MFA_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 snipet 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 = 5
# curve_scale = 20
# 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 = 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

```python
# %% 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/MFA/2022/week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel(r'MFA_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 snipet 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 = 5
# curve_scale = 20
# 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 = 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 survival curve matrix

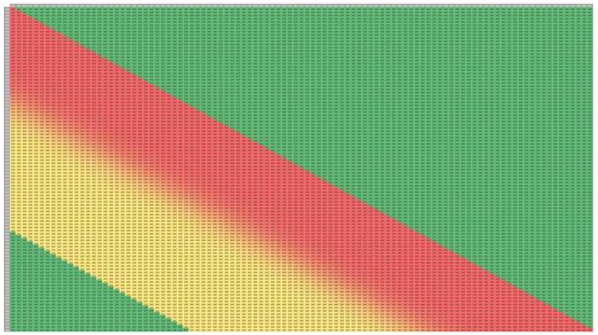
cohorts →

↓ years

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	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0	0
9	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865	0
10	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134	0.99865
11	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445	0.998134
12	0.96407	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533	0.997445
13	0.955435	0.96407	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339	0.996533
14	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379	0.995339
15	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802	0.99379
16	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986097	0.989276	0.991802
17	0.9032	0.919243	0.933193	0.945201	0.955435	0.96407	0.971283	0.97725	0.982136	0.986097	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

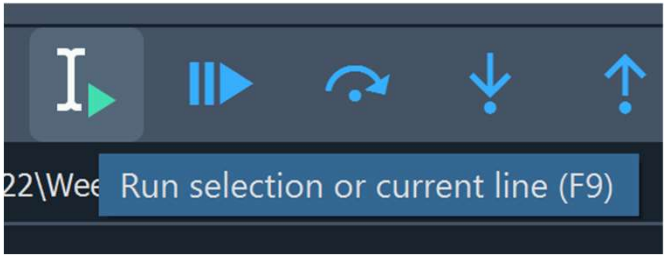
1. Load required modules and Excel data
2. Create a single survival curve
3. Create a survival curve matrix
4. Calculate stocks and flows
5. Export results to Excel

```python
# %% 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/MFA/2022/week_2')

# Load input data, inflow-driven model:
stock_flow_timeseries = pd.read_excel(r'MFA_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 snipet 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 = 5
# curve_scale = 20
# 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 = 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')
```

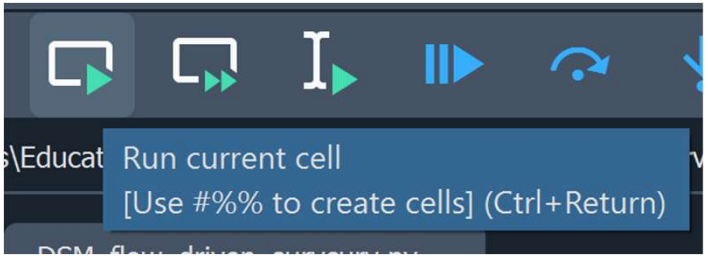
Let's run the model line by line

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 (demarked with#%%)



1. Load modules & data

- What happens in line 16 & 19?

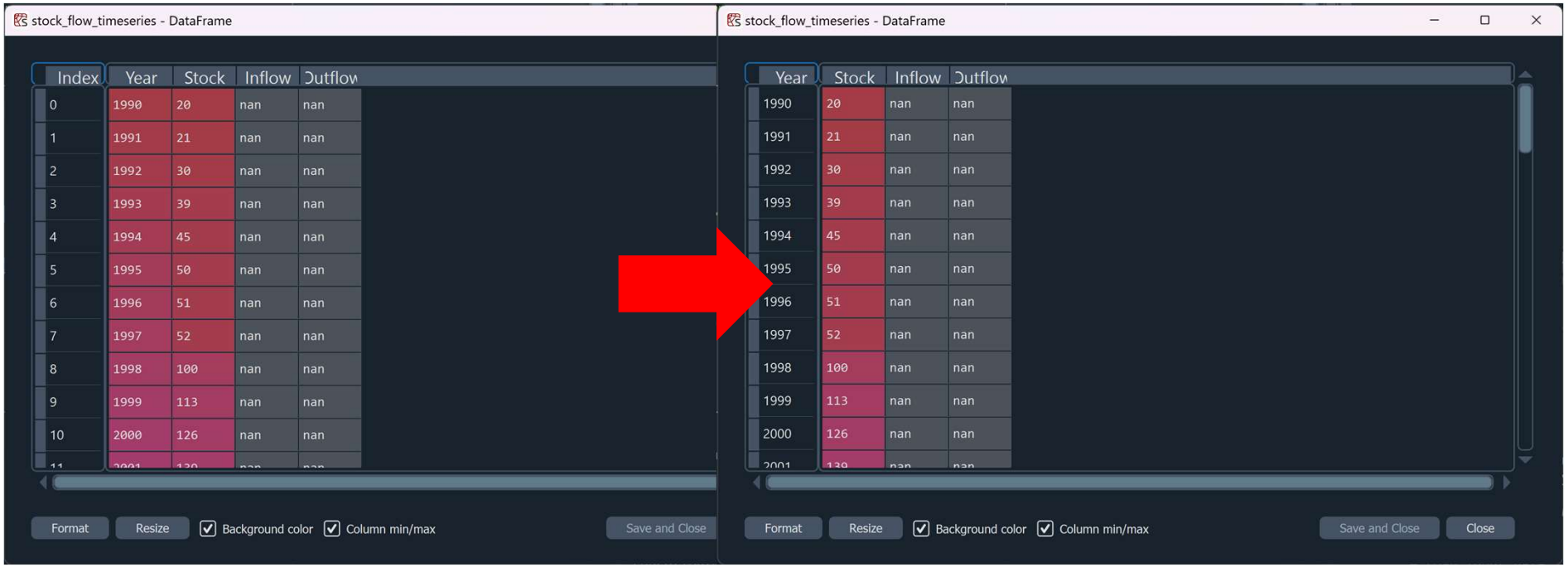
```
10  # %% 1. Load modules & data
11  import pandas as pd
12  import numpy as np
13  import scipy.stats
14  from os import chdir
15
16  chdir('C:/Users/jvano/OneDrive - Universiteit Leiden/Files/Education/MFA/2022/Week_2')
17
18  # Load input data, inflow-driven model:
19  stock_flow_timeseries = pd.read_excel(r'MFA_II_tutorial_II.xlsx',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

- And in line 22?

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

1. Load modules & data

1. Line 22:
   - Run the line, what happened?



1. Load modules & data

1. Line 24-26
   - What does time_max and timesteps refer to?

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

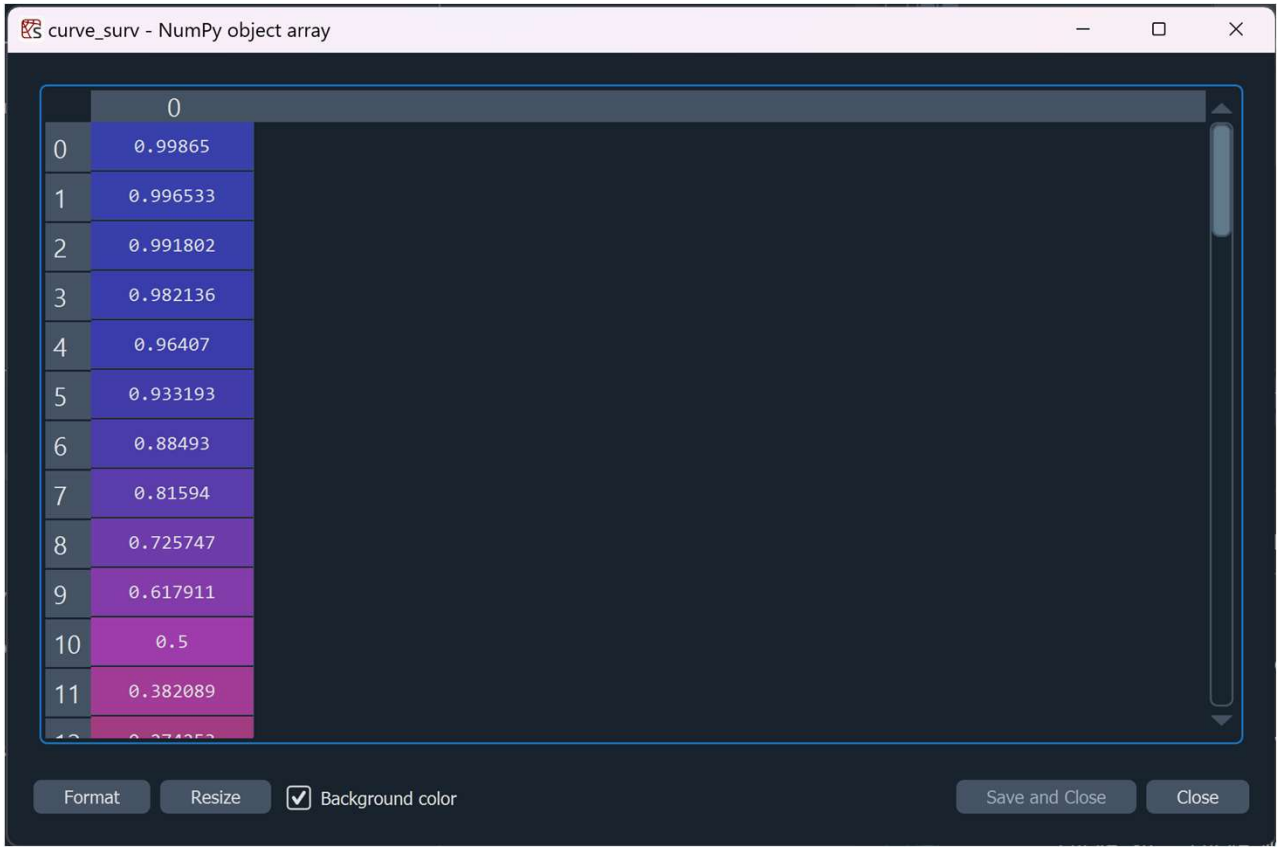
time_max	int	1	61
timesteps	Array of int32	(61,)	[0 1 2 ... 58 59 60]

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?

```
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

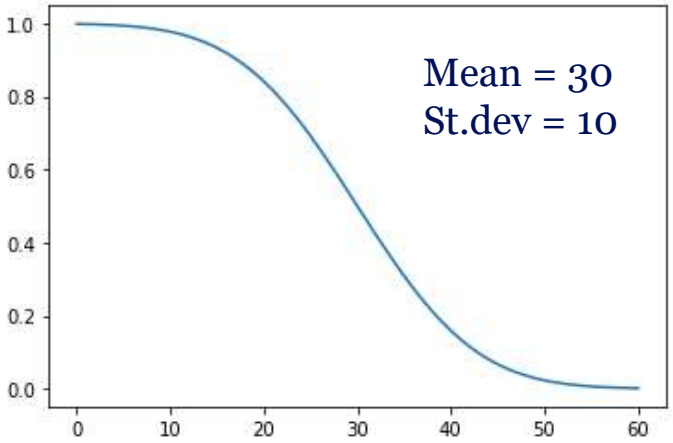


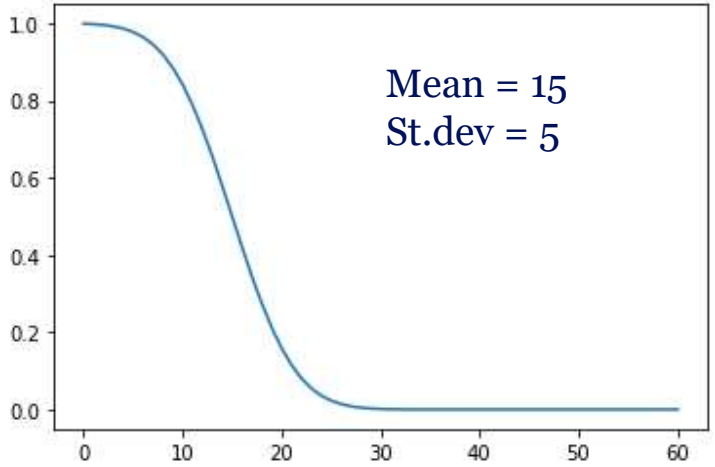
2. Create a single survival curve

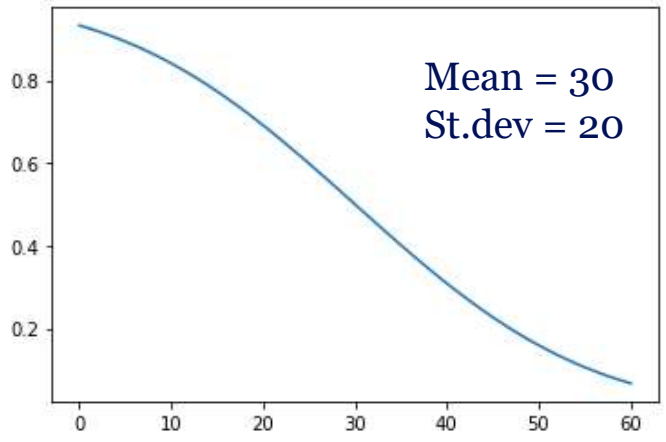
1. Line 47-48
   - Run line 47 and 48, explore the results. What happens when you change the curve_mean to 15? What if you change the curve_sd to 3? And to 20?
   - Change the values back as illustrated below

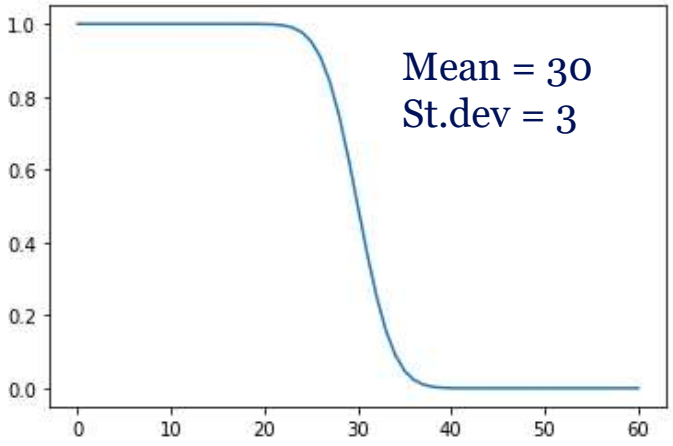
```
43  #Normally distributed survival curve
44  curve_mean = 30
45  curve_sd = curve_mean / 3
46  curve_surv = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)
47  plt.plot(curve_surv)
48  plt.show()
```

2. Create a single survival curve





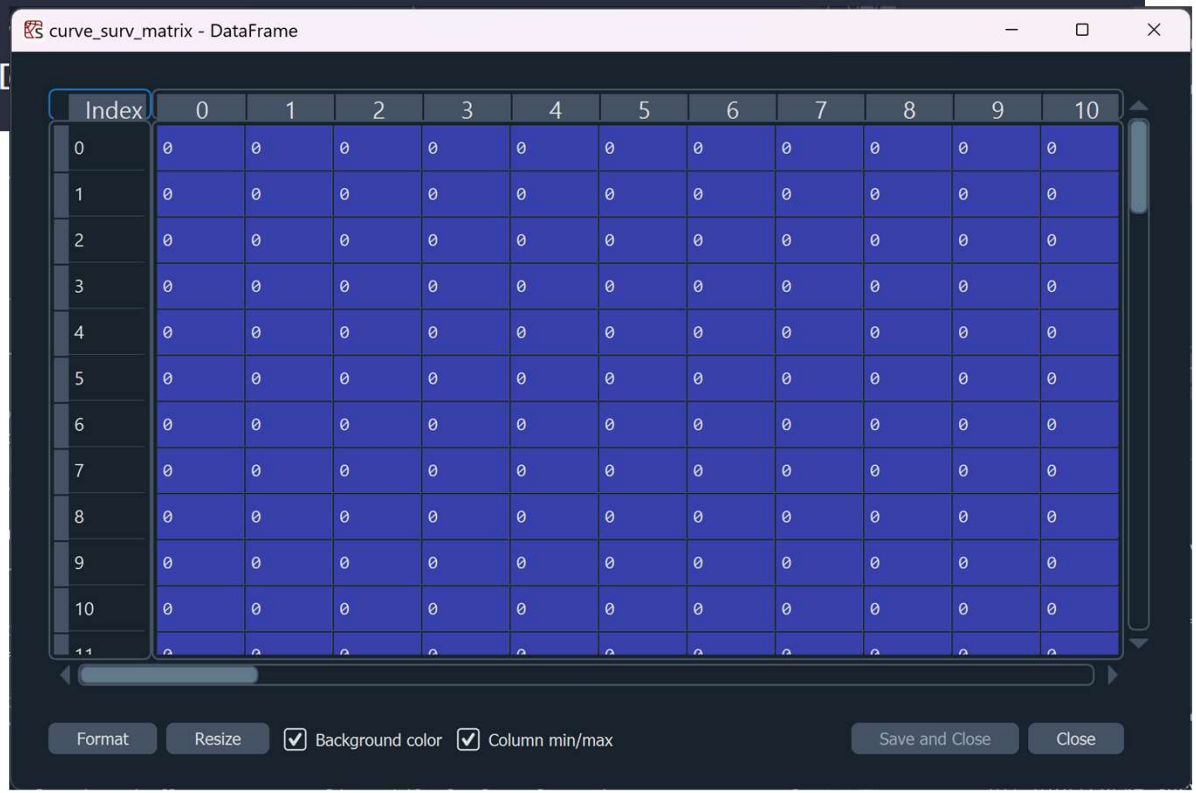




3. Create survival curve matrix

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

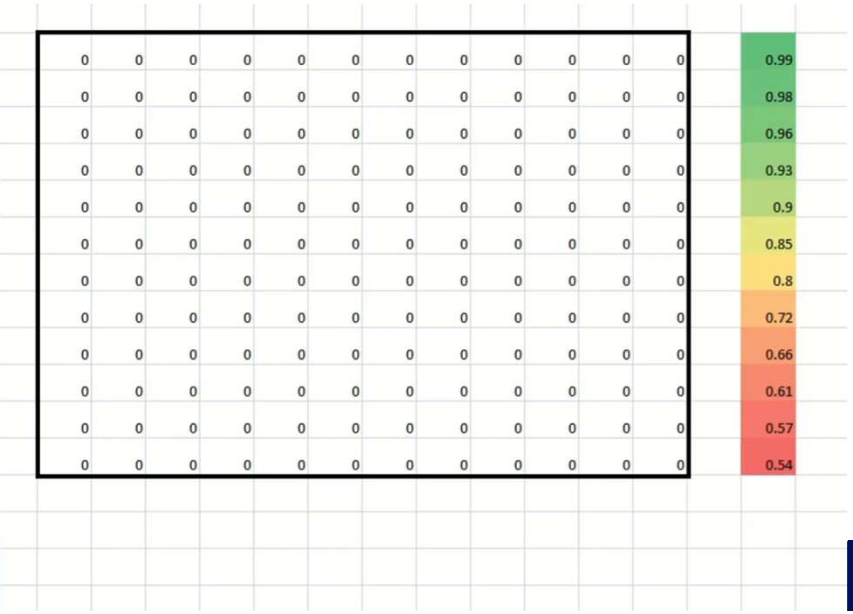
```
66  # create survival curve
67  curve_surv_matrix = pd.D
68
```



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

cohorts →

Survival curve matrix

← years

Share of inflow in 1990 that survives in 1990

Share of inflow in 1990 that survived in 1991

Share of inflow in 1990 that survived in 1992

3. Create survival curve matrix

cohorts →

Survival curve matrix

← years

Share of inflow in 1992 that survives in 1992

Share of inflow in 1992 that survived in 1993

Share of inflow in 1992 that survived in 1994

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)
```

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]
```

3. Change the index to years instead of timesteps

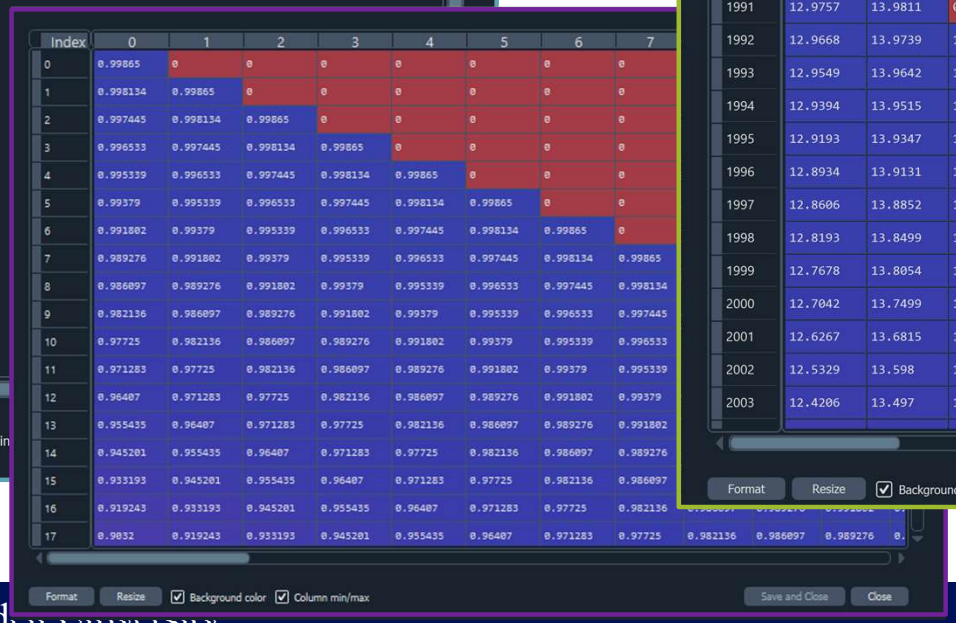
```
83    # set row index to years instead of timesteps
84    cohort_surv_matrix.index = stock_flow_timeseries.index
85
```

4. Flow driven model

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

Stock_flow_timeseries

Survival curve matrix

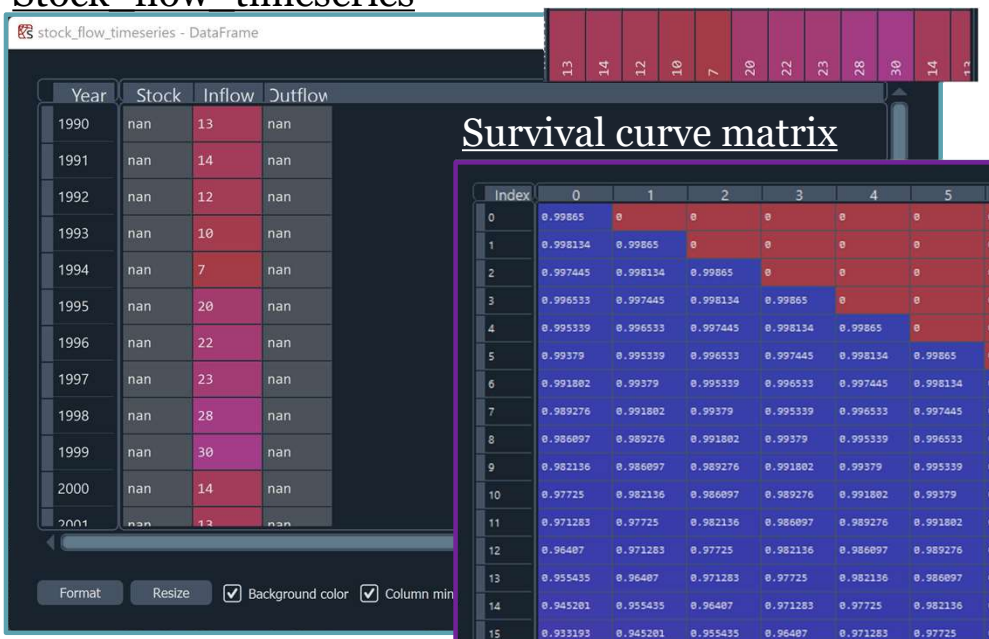


Cohort survival matrix

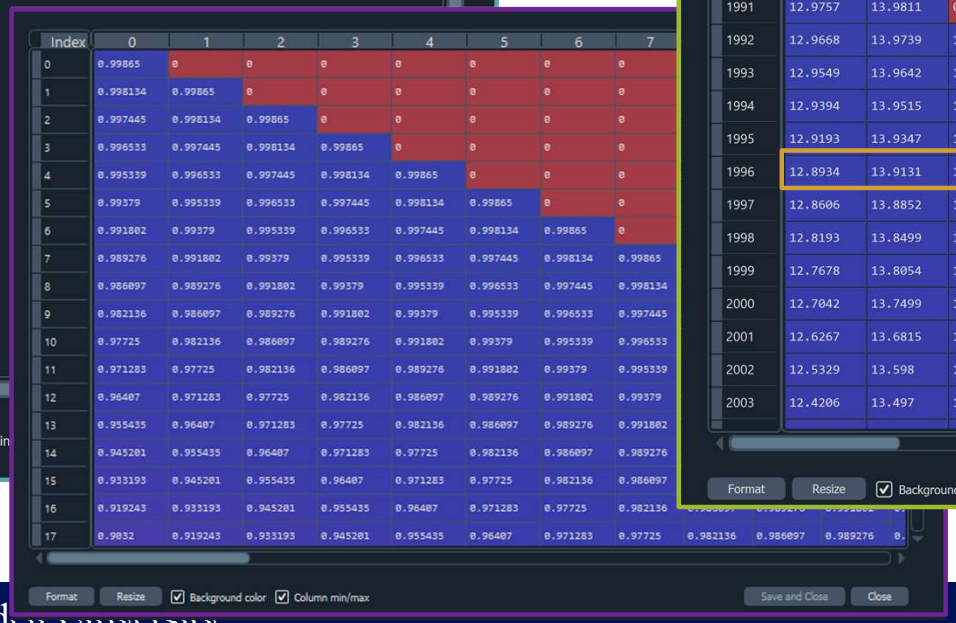
4. Flow driven model

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

Stock_flow_timeseries



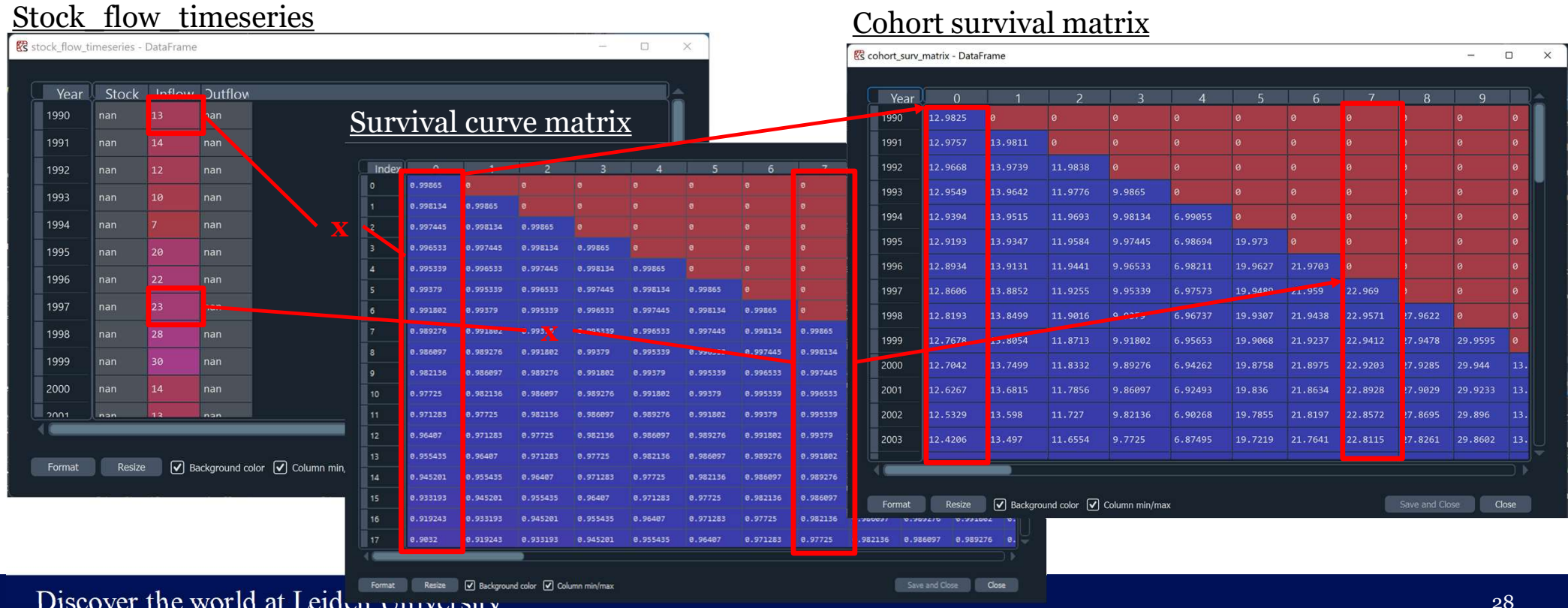
Survival curve matrix



Cohort survival matrix

4. Flow driven model

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

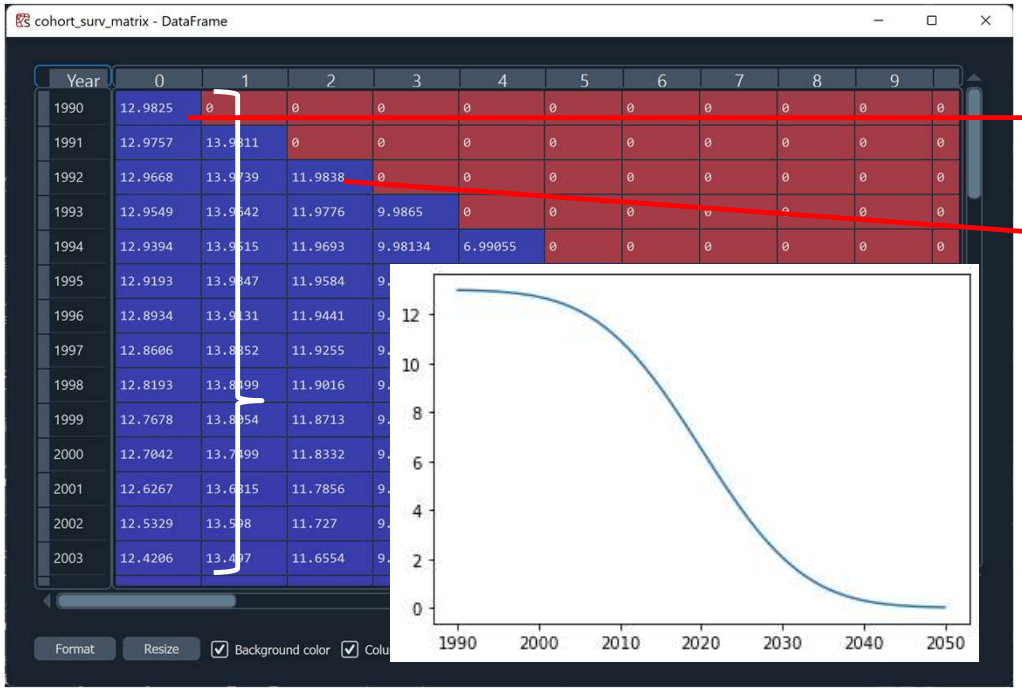


4. Flow driven model

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

Stock per cohort (birth year) →

Year ↓



Share of inflow in 1990 that survives in 1990

Share of inflow in 1992 that survives in 1992

4. Flow driven model

Now we can…

- Calculate the stock:

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

- Calculate the outflow:

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

$$outflow(y) = inflow(y) - NAS(y)$$

Cohort survival matrix

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

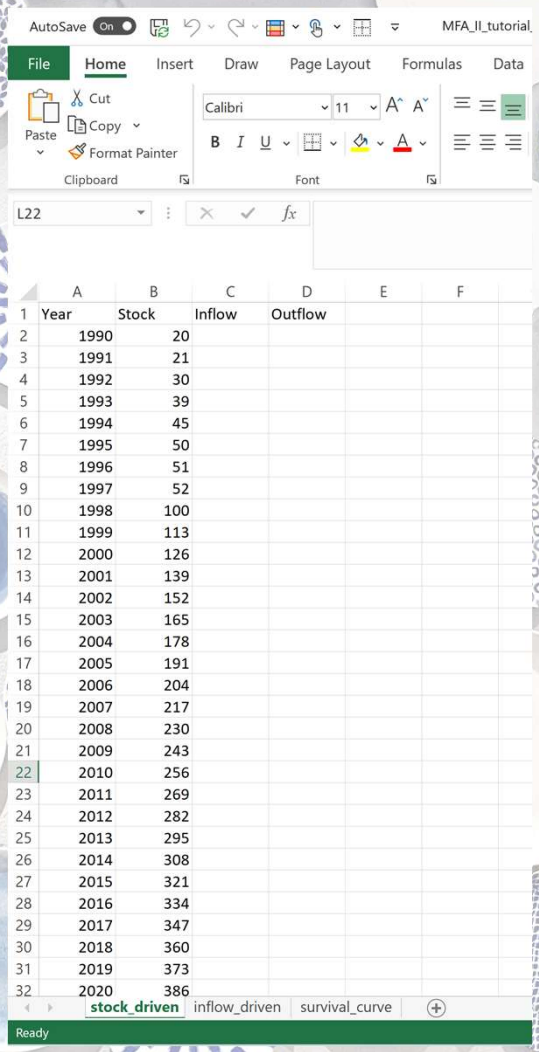
Universiteit
Leiden
The Netherlands

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	B	C	D
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

```python
# %% 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/MFA/2022/week_2')

# Load input data, stock-driven model:
stock_flow_timeseries = pd.read_excel(r'MFA_II_tutorial_II.xlsx', sheet_name='stock_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 snipet 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 = 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 = scipy.stats.norm.sf(timesteps, loc=curve_mean, scale=curve_sd)
plt.plot(curve_surv)
plt.show()

# # 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))

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

curve_surv_matrix.loc[1:, 1] = curve_surv[0:time_max - 1]

# %% 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'].iloc[time] = (stock_flow_timeseries['stock'].iloc[time] - cohort_surv_matrix.loc[time, :].sum()) / curve_surv_matrix.loc[time, time]
    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 outflows and nas using the cohort_surv_matrix
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']

# %% 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

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

4. Calculate stocks & flows

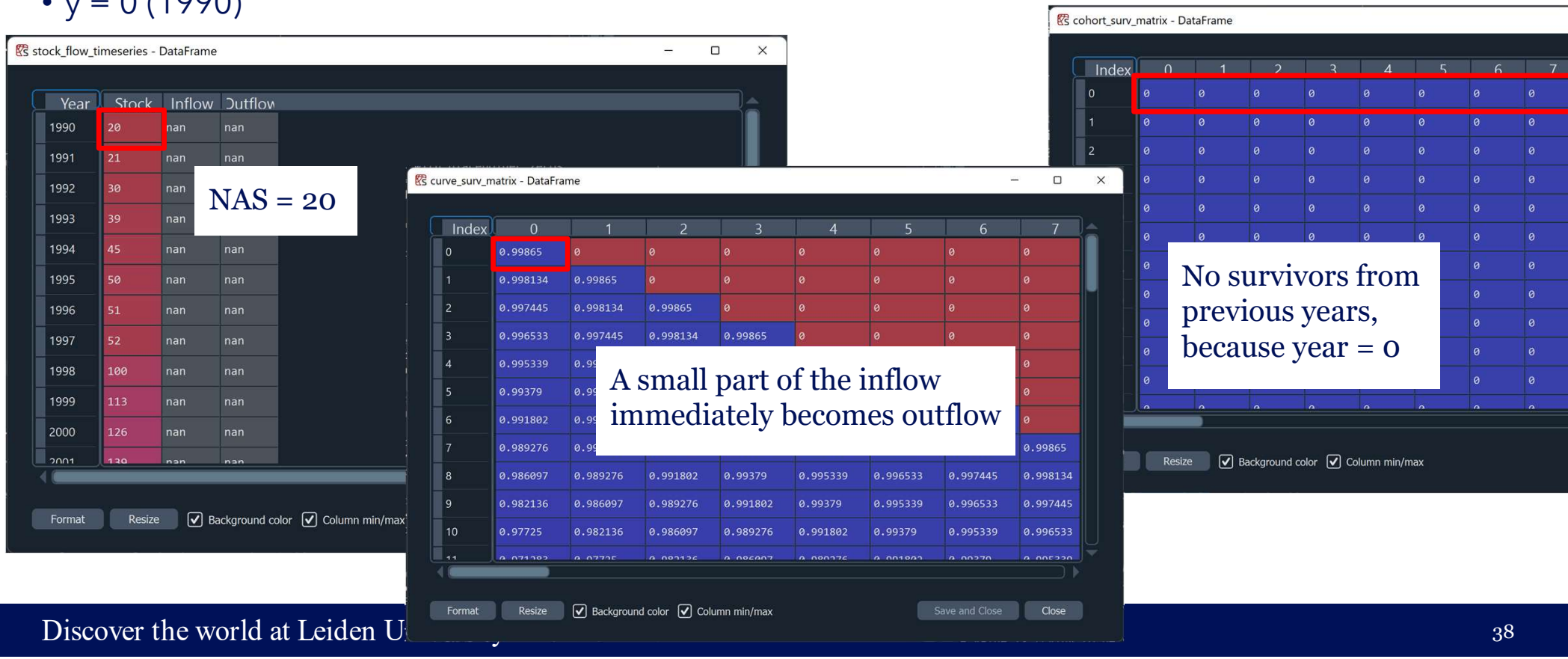
- **Let's try in python**
- Calculate the inflow for timestep 0

```
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)



4. Calculate stocks & flows

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

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

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	139	nan	nan

Format Resize Background color Column min/max

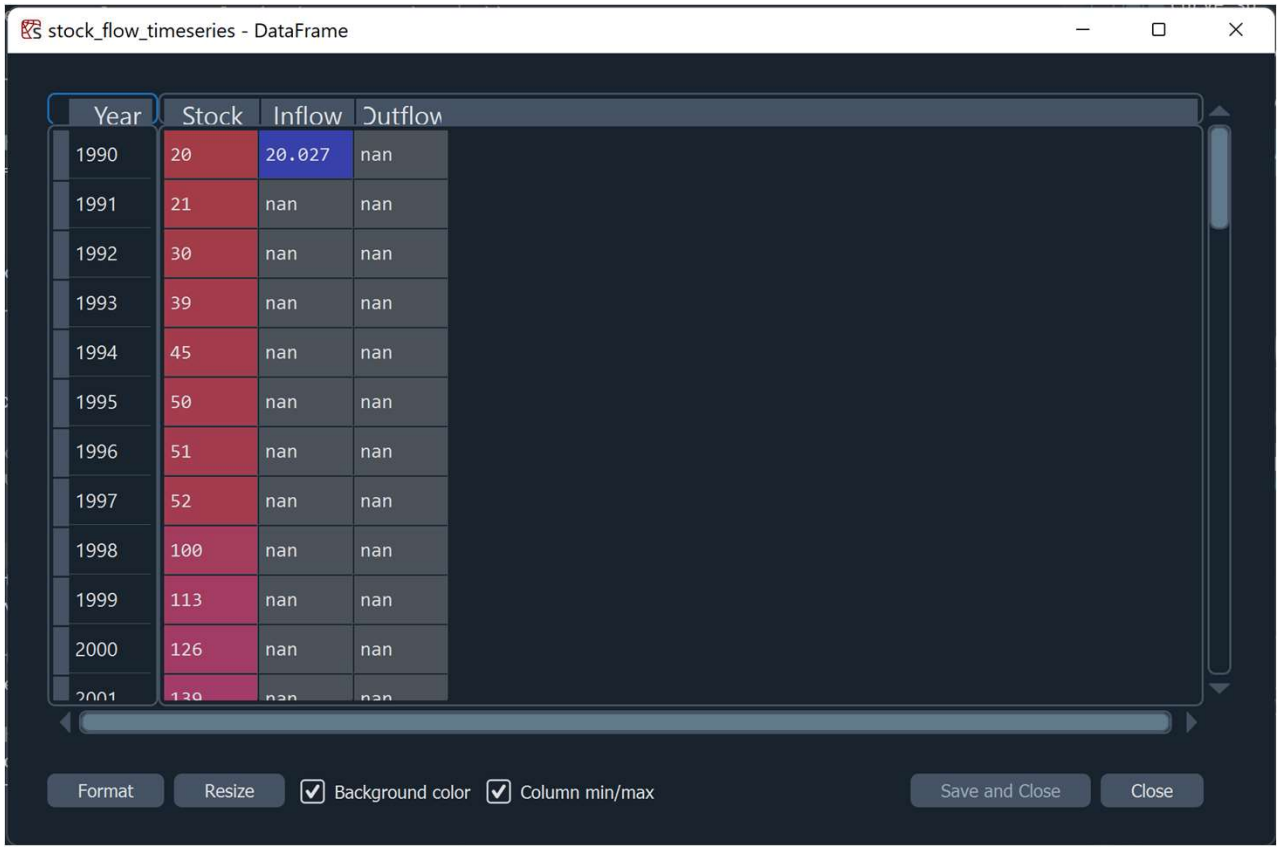
curve_surv_matrix - DataFrame

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

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

4. Calculate stocks & flows

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



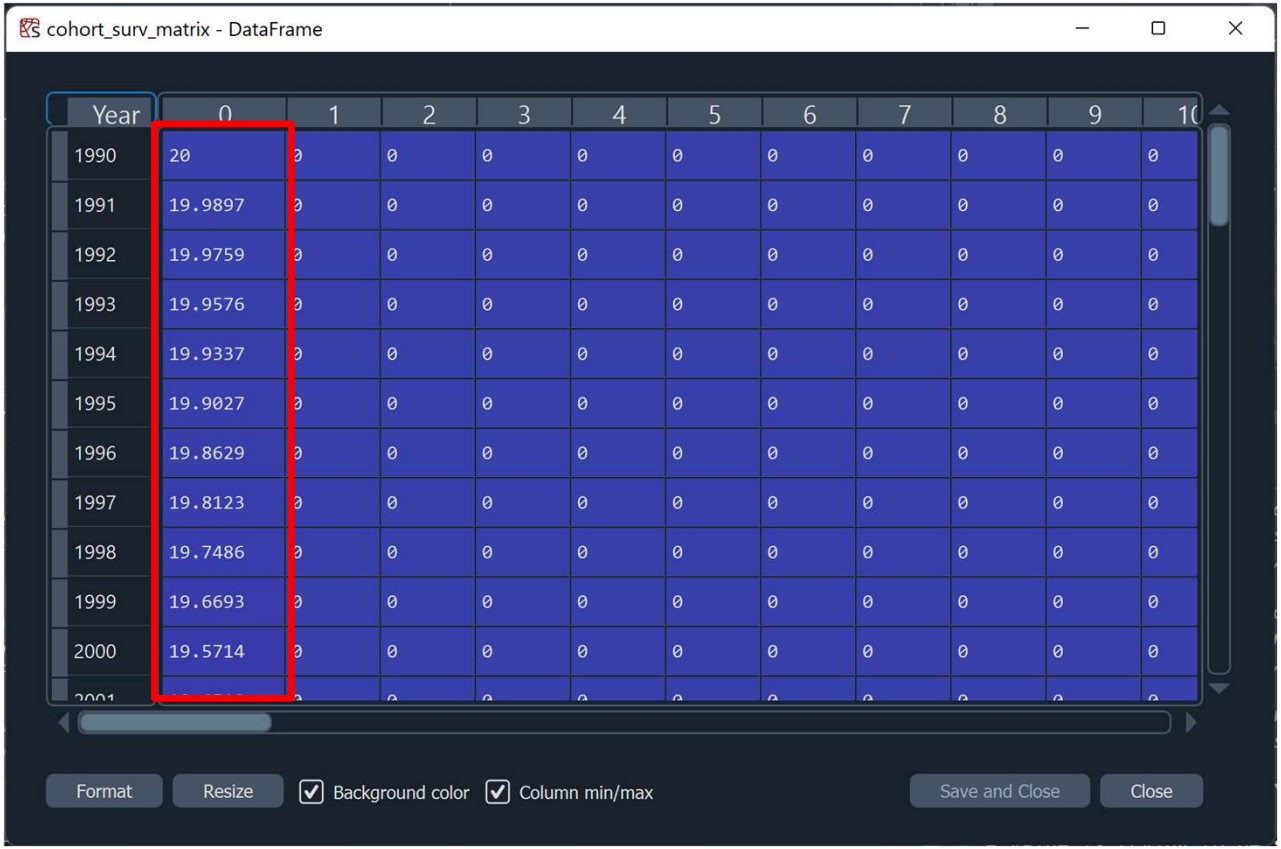
4. Calculate stocks & flows

- What does the line below do?

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

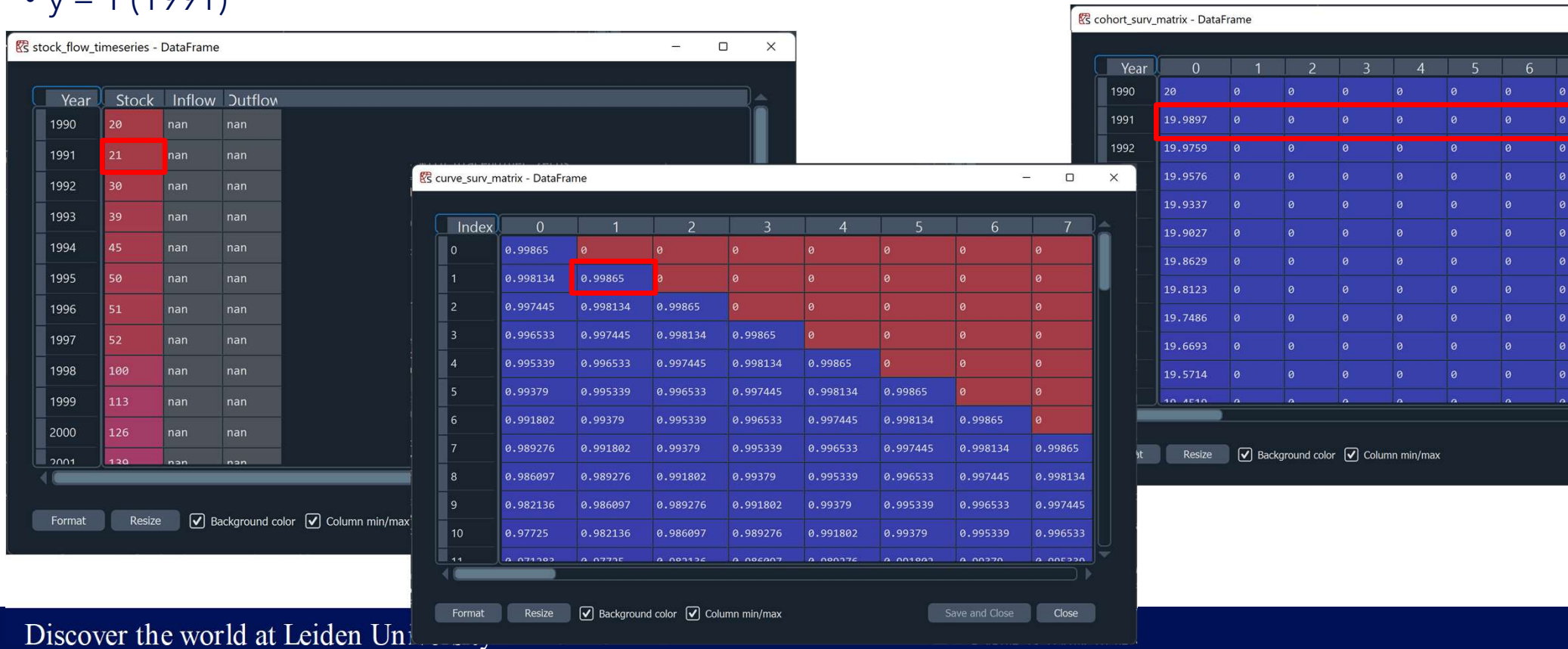
4. Calculate stocks & flows

- **Calculate inflow for timestep 0**
- y = 0
- Multiply the inflow times the shifted survival curve to get the cohorts' behavior over time



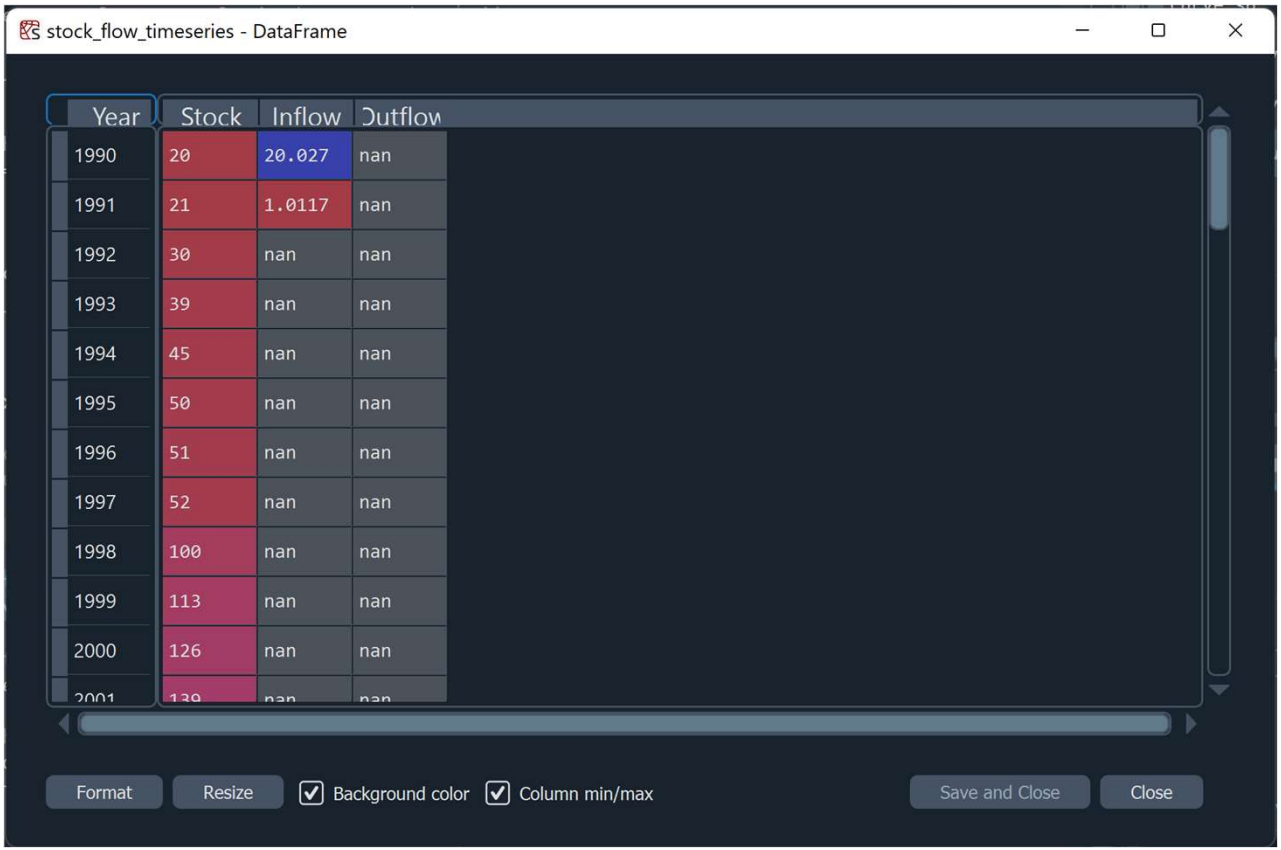
4. Calculate stocks & flows

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



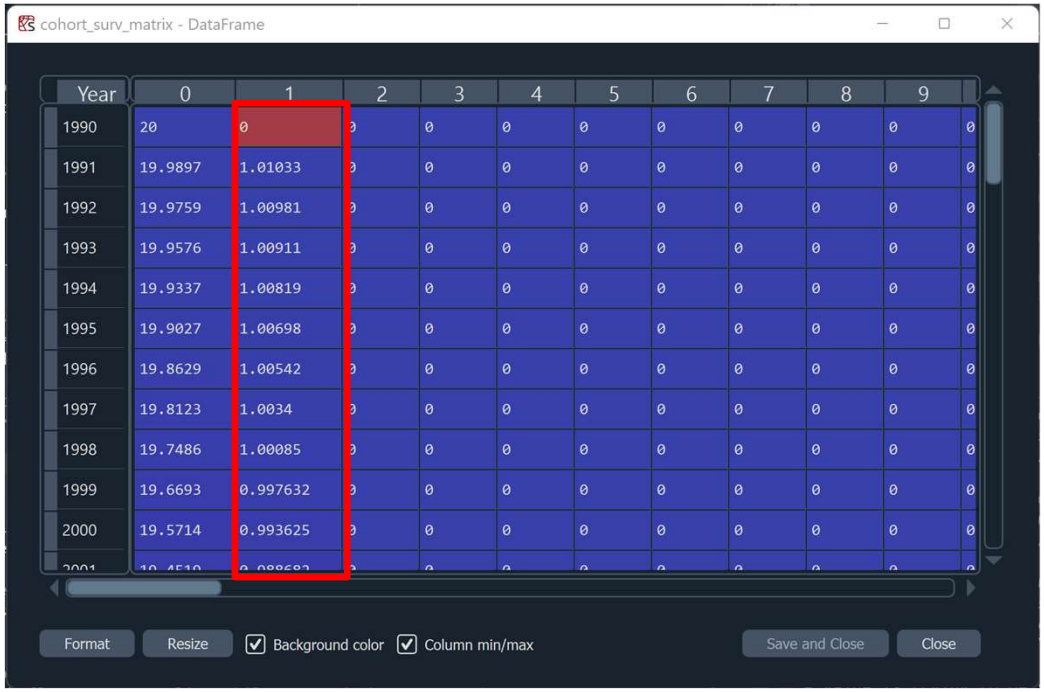
4. Calculate stocks & flows

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



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



4. Calculate stocks & flows

- Calculate outflows:

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

$$outflow(y) = inflow(y) - NAS(y)$$

- What does prepend=0 do?

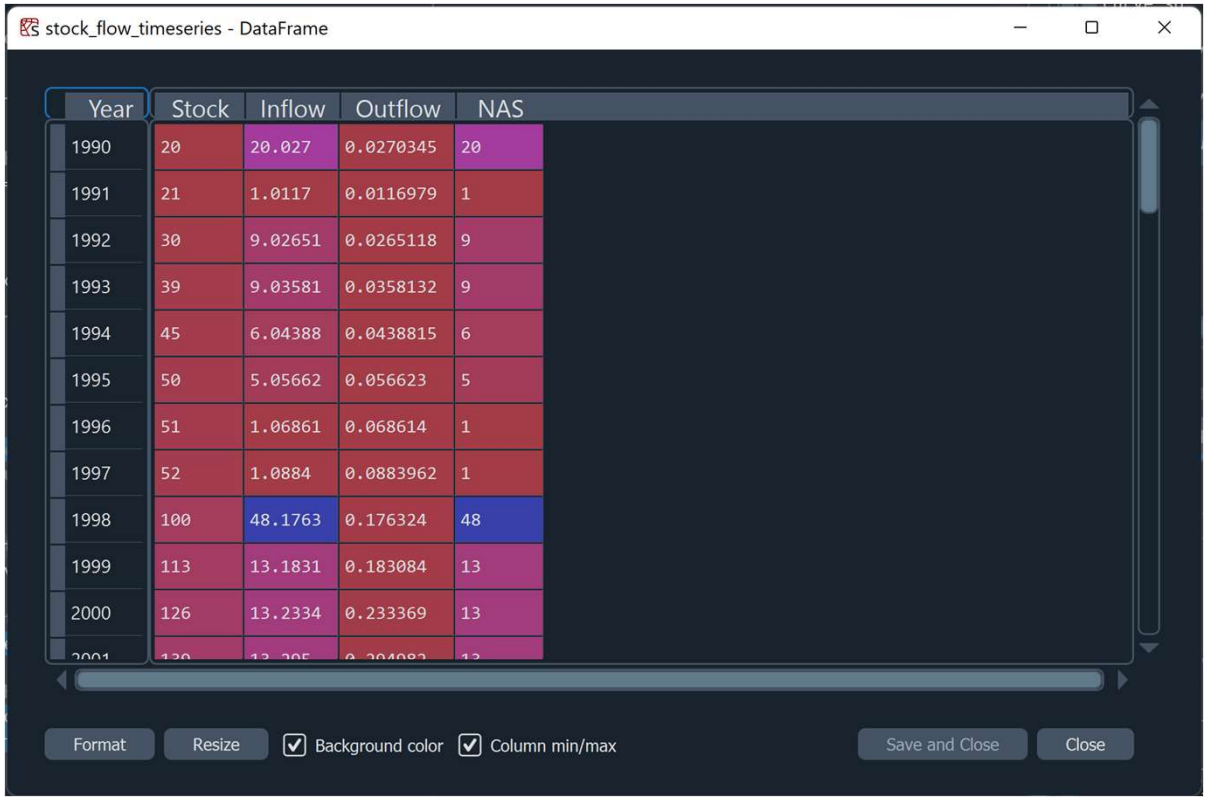
```
90  # calculate outflows and nas using the cohort_surv_matrix
91  stock_flow_timeseries['nas'] = np.diff(stock_flow_timeseries['stock'], prepend=0)  # prepending 0 assumes no initial stock
92  stock_flow_timeseries['outflow'] = stock_flow_timeseries['inflow'] - stock_flow_timeseries['nas']
93
```

4. Calculate stocks & flows

- Calculate outflows:

$$NAS(y) = stock(y) - stock(y-1)$$

$$outflow(y) = inflow(y) - NAS(y)$$



Universiteit
Leiden
The Netherlands