Implementation DoBr 7.4 (Python)

1 Introduction

The DoBr tool is a collection of Python modules to calculate several Key Performance Indicators (KPI's) for the (R, s, nQ), (R, s, S), and (R, s, S, nQ) inventory policies. The tool is developed at TU/e by Rob Broekmeulen and Karel van Donselaar as an open source project. It uses the short, permissive MIT license to promote valorization of our research.

Six basic KPI's:

1. Fill rate P_2;
2. Discrete ready rate P_3^{Discr};
3. Expected back orders $E[BO]$;
4. Expected inventory on hand $E[I^{OH}]$;
5. Expected number of order lines per review period $E[OL]$;
6. Expected order size per order line $E[OS]$.

And several additional KPI's which are relevant for retail operations, such as:

- Expected outdating $E[W]$;
- Expected sojourn time of the units sold from stock $E[ST]$;
- Expected number of unit load arrivals per review period $E[UA]$;
- Probability of having overflow on-hand inventory in the backroom just after a potential delivery and after direct stacking P_V;
- Expected amount of overflow in the backroom $E[I^{BR}]$;
- Expected number of idle replenishments $E[NIR]$.

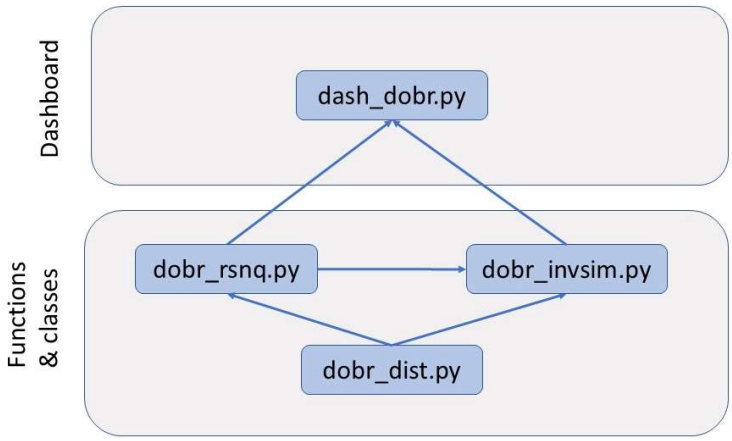
The tool can handle many situations encountered in practice. For details of the used definitions and expressions for the KPI's, the user should consult Van Donselaar & Broekmeulen (2014) and Broekmeulen *et al* (2017). Compared to earlier Excel versions, the Python version only implements published results!

Version 7.4 consists of several Python modules:

- Module `dobr_rsnq`, which contains the code to calculate the KPI's using analytic functions.
- Module `dobr_dist`, which contains the code to deal with two-moment fitted discrete demand distributions.
- Module `dobr_invsim`, which contains the code to determine the KPI's using discrete event simulation. This module also imports the modules `dobr_rsnq` and `dobr_dist`.
- Module `dash_dobr`, which contains the code to present a graphical user interface or dashboard to the user for the calculation of the KPI's, both analytic (using module `dobr_rsnq`) and with simulation (using module `dobr_invsim`).

Please note that the dashboard is only an interface to the analytic and/or simulation routines.

The dashboard can be configured by the parameters in the file `dash_config.json`. If this file is not found or present, the dashboard has a default configuration that is aimed at the TU/e courses Stochastic Operations Management and Production & Inventory Control. It is restricted to situations with backordering and non-perishables when the (R, s, nQ) inventory policy is applied.



In the module `dobr_rsnq`, an object hierarchy is implemented to accommodate the different inventory control systems, (R, s, nQ), (R, s, S), and (R, s, S, nQ), in combination with the demand distribution (Gamma, Normal or Discrete) and the out-of-stock situation: backordering (BO) or lost sales (LS).

- *InvSys:*
  - *InvSysGammaBO*
  - *InvSysNormalBO*
  - *InvSysDiscreteBO*
    - *InvSysDiscreteLS*

Since all functions to calculate the KPI's are implemented as class methods, the relevant subclass must be initialized before a method can be called, The choice for one of the four subclasses depends on the assumed demand distribution and out-of-stock situation. For Gamma and Normal distributed demand, only (R, s, nQ) with backordering is available. The lost sales situation can be handled with the InvSysDiscreteLS subclass. Note that when the dashboard is used, the logic in the dashboard takes care of these decisions.

2 Installation and use

The DoBr tool is distributed as a zip-file containing the following files:

- Python modules `dobr_rsnq.py`, `dobr_dist.py`, `dobr_invsim.py`, and `dash_dobr.py`;
- Auxiliary files `Tue-favicon-32px.ico` and `zreg_fifo.json`.

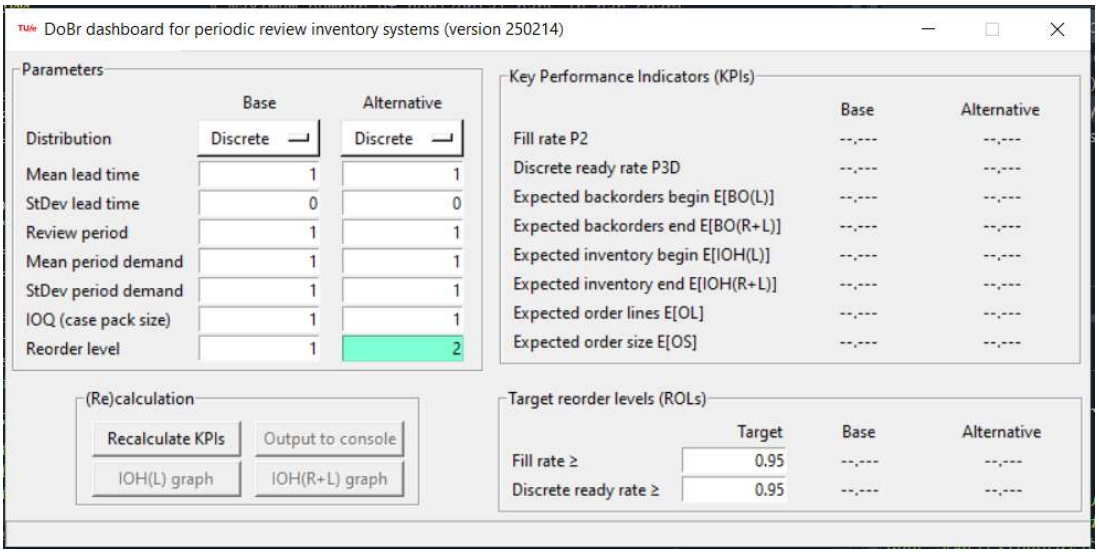
Before using DoBr, these files must be unpacked in a single folder. Depending on your operating system and python installation, you can either execute one of the dashboard modules from your command prompt or run the relevant dashboard module in IPython (available in Spyder).

3 Using the default dashboard

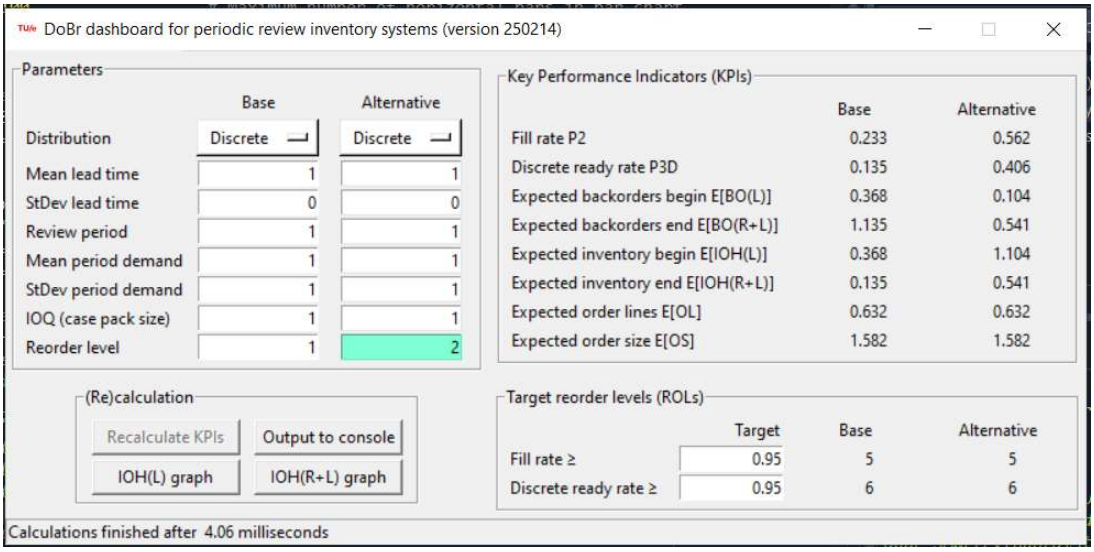
When using the command prompt, enter the following command for opening the dashboard:

In[1]: %run dash_dobr.py

The dashboard is now visible as a separate window.



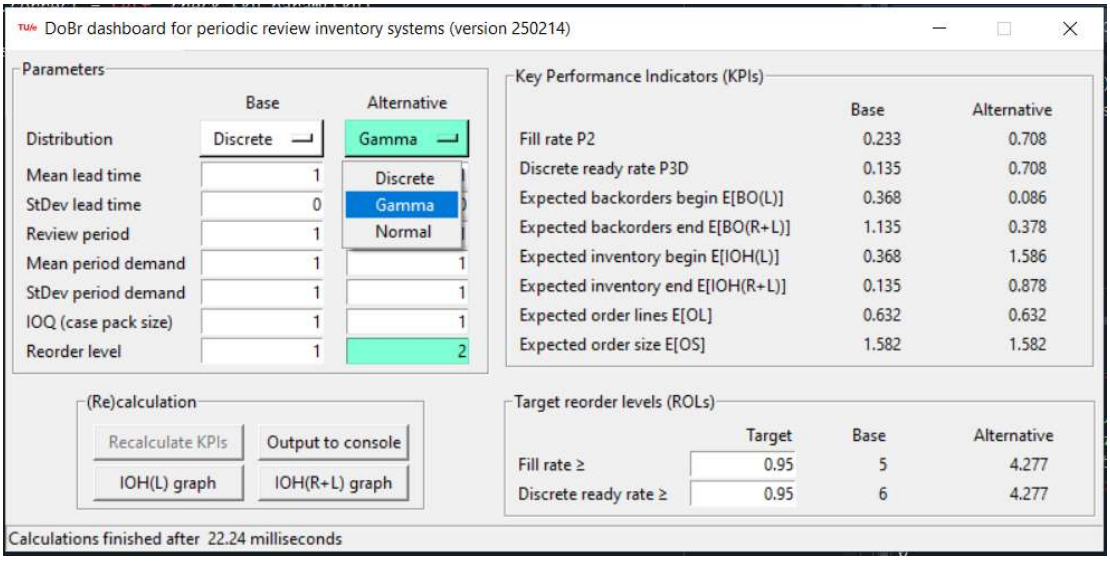
On the left, we see a frame with two sets of input parameters to evaluate two scenarios: Base and Alternative. The differences between the two sets of input parameters are highlighted in green in the Alternative column. In the screenshot, this is the reorder level. To (re)calculate the KPI's, you need to click the button "Recalculate KPIs". This results in the following screen:



Exact results are written in black, approximations in blue and unavailable KPI's are omitted or presented as "~~.~~~". In the default dashboard configuration, all KPI's are exact for deterministic lead times, since we deal here with non-perishables and backordering. Note that the $E[BO]$ and $E[I^{OH}]$ KPI's have two values: at the begin of the potential delivery cycle

(just after a potential delivery with lead time L) and at the end of the potential delivery cycle (just before a potential delivery, after a review period R plus lead time L).

The user can select a distribution by clicking the list box button (see example below).



In the "Target reorder levels (ROLs)" frame (at the right-hand bottom of the window), the tool calculated the reorder levels that are required to meet the target values for the fill rate and the discrete ready rate. Note that for continuous demand distributions, such as Gamma and Normal, the reorder level can be non-integer.

4 Input parameters for the KPI's and the simulation

For the calculations, the input parameters must be within certain ranges to be valid. The following list of parameters is not available in every dashboard configuration.

Basic input parameters (used in the default dashboard configuration):

- **Distribution**: the user can choose between three demand distributions: Discrete, Gamma and Normal. In case of discrete demand, the single period demand distribution is fitted with the procedure of Adan *et al* (1995). By default, demand distributions for other period lengths are derived by convolution whenever possible, otherwise the tool uses the two-moment fit procedure of Adan *et al* (1995).
- **Mean lead time** L: must be non-negative.
- **Stdev lead time**: must be non-negative. For deterministic lead times, the standard deviation of the lead time must be zero.
- **Review period** R: must be positive (no continuous review possible).
- **Mean period demand** μ: must be non-negative. The demand is expressed in units per period, such as consumer units per day or case packs per week. All other input parameters expect the same dimensions of units and period.
- **StDev period demand** σ: must be positive. If the standard deviation is unknown, the user can input -1. In that case, the standard deviation is estimated using an approximation based on the mean period (daily) demand: $\sigma = 1.18 \cdot \mu^{0.77}$. This is a result derived for daily demand for perishables in supermarkets (Broekmeulen & van Donselaar, 2019).

- $\boldsymbol{IOQ}$ (= Incremental Order Quantity): for continuous demand distributions this must be a non-negative value. For discrete demand distributions, this must be a positive integer value. In retail, the IOQ is equal to the case pack size.
- **Reorder level** s: for discrete demand distributions, this must be an integer value. For lost sales situations, this must be non-negative. Note that an order is generated if the inventory position is strictly below the reorder level.
- **Target**: must be a value between 0 and 1.

Other input parameters (visible in the dashboard configuration with extra KPIs):

- $\boldsymbol{MOQ}$ (=Minimum Order Quantity): this must be a non-negative value. For continuous demand distributions, the IOQ must be equal to the MOQ. If the MOQ is supposed to be equal to the IOQ, the user can enter the value zero for the MOQ and the tool will make sure that this happens. The tool can only handle MOQ values greater than the IOQ for discrete demand distributions and when the MOQ is an integer multiple of the IOQ. Therefore, the MOQ must be an integer for discrete distributions.
- **OOS situation** (OOS = out of stock): either lost sales (checked) or backordering. Note that for the approximations for perishables, we require that the lost sales OOS situation is selected.
- **Shelf life** m: this parameter indicates the remaining product lifetime upon entering the stock point. The value of the shelf life for a perishable item must be between $R + 1$ and 30. We have a newsvendor situation if $R = m > 0$, but this is not covered in the tool. For values larger than 30 or the value 0, the item is assumed to be non-perishable. The default value for this optional parameter is 0.
- **FIFO fraction**: this value must be between 0 and 1. Note that the approximations in the tool can only handle 100% FIFO withdrawal, i.e. FIFO=1.0 and the EWA policy.
- **Modify IP**: by selecting EWA (only applicable for perishables), the inventory position (IP) before ordering is modified based on the Estimated Withdrawal and Aging during L+R-1 periods if L+R-1>0. The default value is False.
- **Modify order**: in the situation with lost sales the order size is restricted based on the following equation: $\|max(s \cdot R/(R+L), MOQ)/IOQ\| \cdot IOQ$. The default value is False.
- **Unit load capacity** U: this must be a non-negative integer value. This value indicates the maximum number of units that fit on a unit load such as a pallet. If the U is supposed to be equal to the IOQ, the user can enter the value zero and the tool will make sure that this happens. For discrete demand distributions, the unit load capacity must be an integer multiple of the IOQ. The default value for this optional parameter is 0, assuming $U = IOQ$.
- **Shelf space** V: must be non-negative. A value of zero indicates ample shelf space (storage capacity at the stock point) and no need for a backroom. The default value for this optional parameter is 0.
- **Backroom replenishments**: by selecting Concurrent (only relevant for positive shelf space), the inventory is replenished during the sales period (=concurrent) and not just during idle time. The default value is True.

The values of the IOQ and the MOQ determine the type of inventory control system.

- (R, s, nQ) system: $IOQ \geq 1$ and $MOQ \leq IOQ$.
- (R, s, S) system: $IOQ = 1$ and $MOQ > 1$.
- (R, s, S, nQ) system: $IOQ > 1$ and $MOQ = n \cdot IOQ$ with $n > 1$.

In the special case with $IOQ = MOQ = 1$, the (R, s, nQ) system reduces to the (R, S) system. In the case of a system with an order up to level S we assume that $S = s - 1 + MOQ$.

The default parameter settings are stored in human-readable json-type files. After calculation of the KPI's, the parameters of the Base scenario are stored in `dobr_defaults.json`. In the next session, these stored parameters are loaded and presented. By deleting these files, the parameters in the dashboard return to the default values.

The simulation run parameters are stored in `dobr_simdefaults.json`: The following three parameters are visible in the dashboard configuration with simulation:

- **Max number of subruns**: the simulation run can terminate earlier if at least 10 subruns are executed and the required precision is reached.
- **Subrun warmup**: the number of periods after which the statistics are recorded.
- **Subrun length**: the number of periods for which the statistics are recorded.

The other parameters are available in the json-file:

- **targetkpi**: the KPI for which we want to reach a certain precision (default is Fillrate).
- **targetprecision**: the desired absolute precision of the target KPI (default is 0.001).
- **poissonprocess**: the demand arrivals follow a Poisson process (default is False).
- **weekpattern**: the demand follows a week pattern (default is False).
- **reportpmfs**: the simulation registers the PMFs of some KPIs (default is True). Setting this to False will speed up the simulation run.

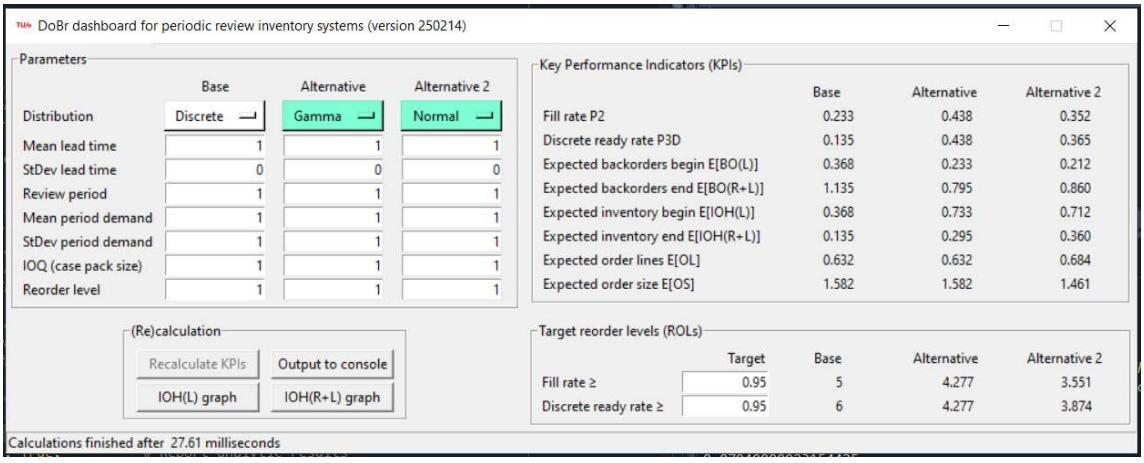
The simulation is setup in such a way that each subrun (or repeat) uses unique seeds for the random number generator, one for each stochastic process. Each simulation experiment will use the same set of seeds for the subruns. We strive for an absolute precision for the fill rate $P_2 \pm 0.001$ with 95% confidence. For more info, see Law & Kelton (2000).

5 Managing the dashboard configuration

The earlier screenshots were for the default configuration. The configuration can be changed with changing the dictionary values in the `dash_config.json` file. The default settings for this file are:

```
{"nr_skus": 2, "analytic": true, "frame_targetrol": true, "retops":
false, "simulation": false, "sim_columns": 1, "weekpattern": false}
```

The number of scenarios can be increased to a maximum of five by updating the “nr_skus” key in the dash_config file. Below an example with “nr_skus”: 3.

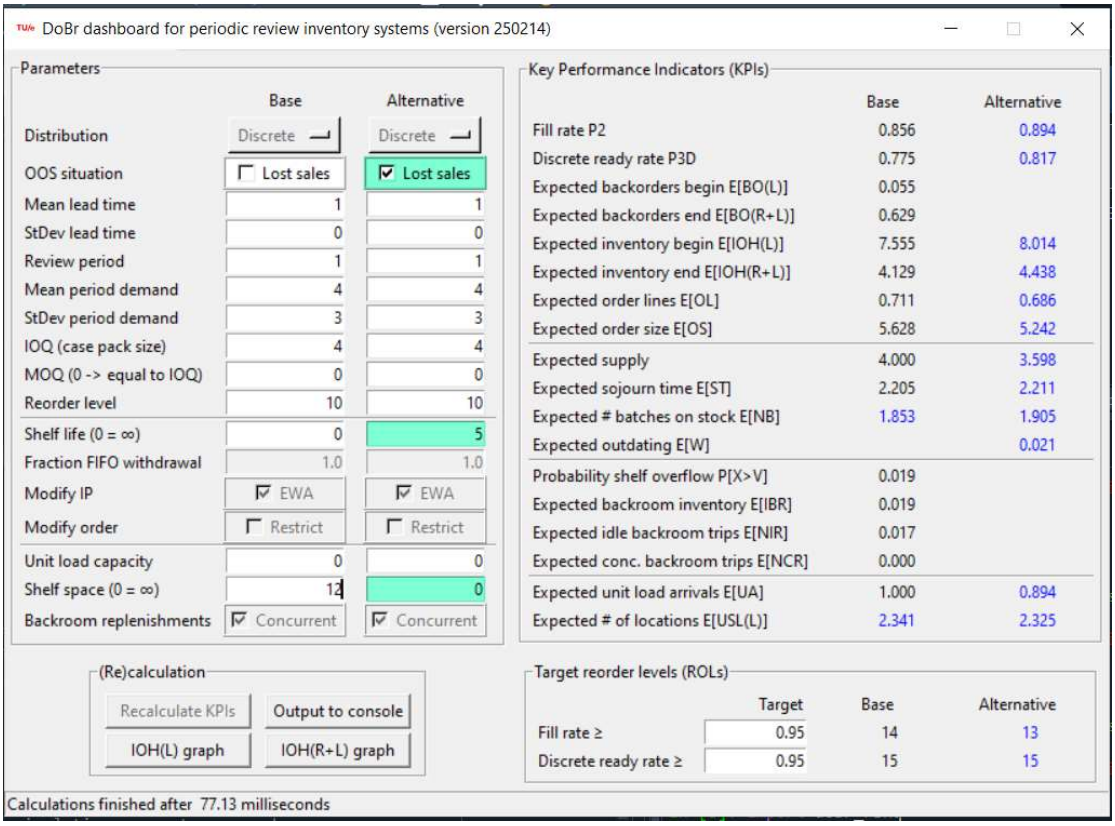


Below the parameters and allowed settings are described:

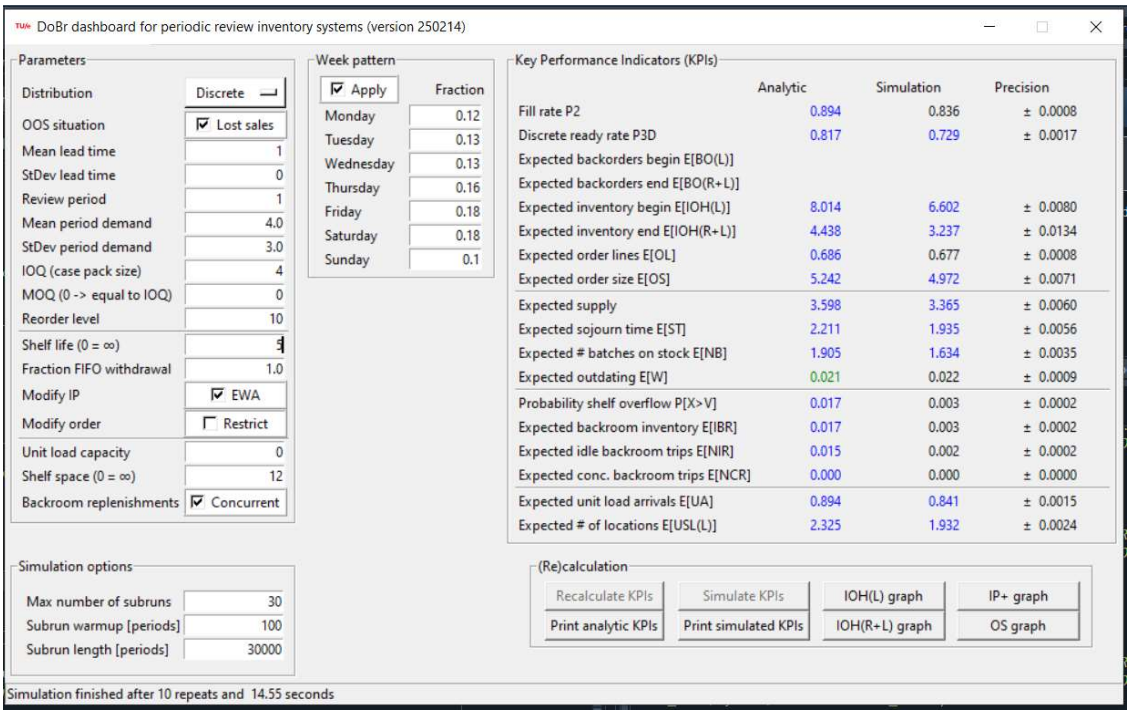
- “`nr_skus`” : This value can range between 1 and 5. It determines the number of scenarios.
- “`analytic`” : Either `true` or `false`. When true, the dashboard calculates the KPI’s based on analytic expressions.
- “`retops`” : Either `true` or `false`. When true, the dashboard shows perishables and capacity-related KPI’s, together with their required input parameters.
- “`simulation`” : Either `true` or `false`. When true, the dashboard calculates the KPI’s based on simulation.
- “`sim_columns`” : This value can be 1, 2 or 3. When the value is 2, the simulation results have an additional column with the precision of the reported values. When the value is 3, the simulation results have also an additional column with the standard deviation of the reported values.
- “`frame_targetrol`” : Either `true` or `false`. When true, the dashboard shows the frame for calculating the target reorder levels based on analytic expressions. This frame is not available when “simulation” is true.
- “`weekpattern`” : Either `true` or `false`. When true, the dashboard shows the input parameters relevant for a week pattern in demand.

Note that not both “analytic” and “simulation” can be false. If “retops” is true, but “simulation” is false, the dashboard does not allow the calculation of the KPI’s for continuous distributions. Also other input parameters are fixed and disabled, since for some combinations, there are no approximations available yet.

With “retops” is true, we get the following dashboard:



And below with “simulation” and “weekpattern” set to true.



6 Using the tool without the dashboard

First, you need to import the `dobr_rsnq` module. Help on a class or method can be obtained by placing the name of the class or function between the brackets of the help command.

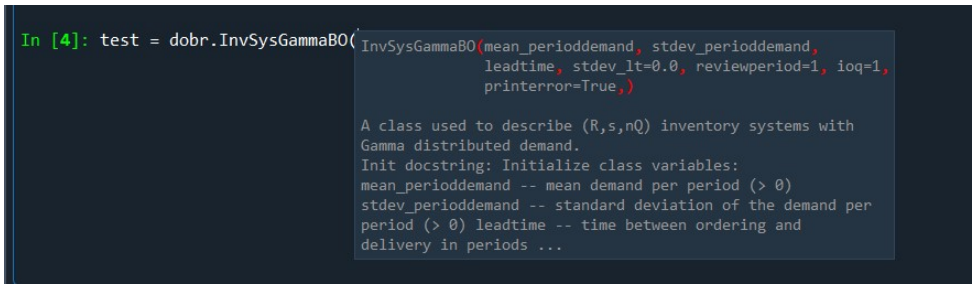
```
In [2]: import dobr_rsnq as dobr

In [3]: help(dobr.InvSysGammaBO.__init__)
Help on function __init__ in module dobr_rsnq:

__init__(self, mean_perioddemand, stdev_perioddemand, leadtime, stdev_lt=0.0, reviewperiod=1,
ioq=1, printerror=True)
    Initialize class variables:
    mean_perioddemand  -- mean demand per period (> 0)
    stdev_perioddemand -- standard deviation of the demand per period (> 0)
    leadtime           -- time between ordering and delivery in periods

    Keyword arguments:
    stdev_lt     -- the standard deviation of the leadtime (default 0.0)
    reviewperiod -- the time between two reviews (default 1.0)
    ioq          -- the incremental order quantity (default 1)
    printerror   -- print error messages to the console (default True)
```

In Spyder, the required and optional key parameters are exposed by pressing Crtl+i in front of the object (or writing a left parenthesis next to an object.



As an example, to initialize a class for Gamma distributed demand with mean 5, standard deviation 3, lead time 1, review period 2, and ioq 4 we would enter:

```
In [5]: test = dobr.InvSysGammaBO(5, 3, 1, reviewperiod=2, ioq=4)
```

When using the class methods instead of the dashboards to calculate the KPI's, these methods return error codes (starting from -9900) in case of invalid input values. For a list of the error codes and their corresponding error messages, see appendix A1.

The class InvSys and its subclasses expose the functions (methods) for the six basic KPI's (including the variants):

1. fillrate
2. readyrate
3. ebo_l and ebo_rl
4. eioh_l and eioh_rl
5. eol
6. eos

With the exception of eol and eos with backordering, they require `reorderlevel` as (positional) parameter.

To calculate the fill rate and the $E[OL]$ for our example with a reorder level of 15, we enter:

```
In [6]: test.fillrate(15)
Out[6]: 0.8677400383190685

In [7]: test.eol()
Out[7]: 0.9916069084912658

In [8]:
```

To calculate all KPI's for a given reorder level, we can use the `print_kpis` function, which also gives a header with a summary of the input:

```
In [8]: test.print_kpis(15)
*= DoBr analytic output ========================================*
 KPI's for a periodic review inventory system assuming back ordering
  and Gamma distributed demand (Mean=5, StDev=3)
  L=1 (fixed), R=2, IOQ=4.0, and s=15.0
 Fill rate (P2)                    :        0.868
 Discrete ready rate (P3D)         :        0.681
 Expected backorders begin E[BO(L)] :        0.008
 Expected backorders end E[BO(R+L)] :        1.330
 Expected inventory begin E[IOH(L)] :       12.008
 Expected inventory end E[IOH(R+L)] :        3.330
 Expected order lines E[OL]        :        0.992
 Expected order size E[OS]         :       10.085
 Expected supply                   :       10.000
 Expected sojourn time E[ST]       :        2.214
 Expected transfer units E[TU]     :        2.500
*================================================================*

In [9]:
```

The '~' behind the value of a KPI indicates that this value is an approximation.

When we initialize the subclasses for discrete demand, we have two additional keyword parameters to determine how the demand distributions are fitted and calculated.

- `usecp` : use Compound Poisson to fit the single period demand distribution on the mean and standard deviation if the variance-to-mean ratio (VTM) is greater than 1, otherwise use the two moment fit procedure of Adan *et al* (1995) (default False);
- `empiricalpmf` : the empirical pmf for the single period (default None).

In case of Compound Poisson, the order size distribution is fitted with a shifted geometric distribution with parameter $p = 2/(VTM + 1)$ and adjusted Poisson arrival rate $\lambda = p \cdot \mu$. Note that convolution is only possible with integer period lengths and deterministic lead time.

Let us illustrate the use of the `empiricalpmf` option. Assume we have discrete demand with the following empirical demand distribution for a single period:

x	0	1	2
P(X=x)	1/2	1/4	1/4

The values for the mean and variance will be derived from the empirical distribution, therefore we enter 1 (or any other value) for the first two positional parameters when we initialize the InvSysDiscreteBO class with an `empiricalpmf` parameter:

```
In [9]: test = dobr.InvSysDiscreteBO(1, 1, 1, empiricalpmf=[0.5, 0.25, 0.25], reviewperiod=2,
ioq=3)

In [10]: test.mean
Out[10]: 0.75

In [11]: test.variance
Out[11]: 0.6875

In [12]: test.fillrate(2)
Out[12]: 0.10069444444444449
```

The `dobr_rsnq` module has also functions to find the reorder level that satisfies a required service level:

- `targetfillrate(x)`: function to find the minimal reorder level that satisfies the target fill rate *x*.
- `targetreadyrate(x)`: function to find the minimal reorder level that satisfies the target discrete ready rate *x*.

In case of continuous demand, the returned reorder level can be a real (non-integer) number. To improve the speed of the bisection search, the user can enter the following keyword parameters:

- `xguess`: Initial guess for the reorder level;
- `yguess`: Value for the service rate at the initial guess;
- `xtol`: Absolute interval for the returned reorder level (default = 0.001).

To find the reorder level at minimal cost, the user must supply a cost function to the `mincost` function. This procedure is based on bisection and assumes that we have a single minimum reorder level (unimodality). With the keyword parameter `xlow`, the user can specify a minimum reorder level (default is 1). The code below illustrates how to use these optimization functions with a user supplied cost function `trc`:

```
 8  import dobr_rsnq as dobr
 9
10  def trc(invsys, rol):
11      """ Calculate the Total Relevant Costs (TRC)."""
12      # Average inventory on hand
13      avg_ioh = (invsys.eioh_l(rol) + invsys.eioh_rl(rol))/2
14      # Backorders during review period
15      bo_rp = (invsys.ebo_rl(rol) - invsys.ebo_l(rol))
16      # Holding costs
17      h = 1
18      # Backorder penalty costs
19      p = 20
20      return h*avg_ioh + p*bo_rp
21
22  # Create an inventory system
23  test = dobr.InvSysGammaBO(5, 3, 1, reviewperiod=2, ioq=4)
24
25  # Find the reorder level with 95% fill rate
26  min_rol = test.targetfillrate(0.95)
27  print(f"Minimum reorder level {min_rol: 7.3f}"
28        + f" with costs {trc(test, min_rol): 10.3f}")
29
30  # Find the cost optimal reorderlevel with at least 95% fill rate
31  opt_rol = test.mincost(trc, xlow=min_rol)
32  print(f"Optimal reorder level {opt_rol: 7.3f}"
33        + f" with costs {trc(test, opt_rol): 10.3f}")
34
```

With the `dobr_dist` module, we can also fit a discrete distribution based on the first two moments using the method of Adan et al (1995). We first initialize a class object with the mean and standard deviation, using the method `two_moment_fit`. Next, the following functions are available for that fitted distribution (with X the random variable):

- `pmf(x)`: returns $P[X = x]$, the probability that X is equal to x;
- `cdf(x)`: returns $P[X \leq x]$, the probability that X will that a value less or equal to x;
- `sf(x)`: returns $P[X \geq x]$ or survival function for x;
- `loss(x)`: returns $E[(X - x)^+]$, the expected loss for x;
- `rloss(x)`: returns $E[(x - X)^+]$, the expected reverse loss for x;

As an example, we initialize a class object with mean 5 and standard deviation 3 and report several function values for the input value x=4.

```
In [2]: import dobr_dist

In [3]: test = dobr_dist.two_moment_fit(5, 3)

In [4]: test.pmf(4)
Out[4]: 0.14251889193190975

In [5]: test.cdf(4)
Out[5]: 0.4908485005088309

In [6]: test.sf(4)
Out[6]: 0.6516703914230788

In [7]: test.loss(4)
Out[7]: 1.6797302408808759
```

We can do the same for an empirical distribution, by initializing an instance of the class `EmpiricalDistArray`.

```
In [2]: import dobr_dist

In [3]: test = dobr_dist.EmpiricalDistArray([0.4, 0.3, 0.2, 0.1])

In [4]: test.mean
Out[4]: 1.0

In [5]: test.cdf(2)
Out[5]: 0.8999999999999999
```

7 Capabilities and limitations

The values of the basic KPI's that the tool returns are exact for the situations with stationary demand, backordering, deterministic lead times, and non-perishable products.

The subclasses InvSysGammaBO and InvSysNormalBO are specific for their respective continuous demand distributions and both assume backordering and $MOQ = IOQ$, i.e. only the (R, s, nQ) inventory system. Besides the basic KPI's, they can also calculate $E[UA]$ and $E[ST]$. This last KPI is only exact for a review period equal to 1.

With discrete demand distributions, more KPI's are available and also for situations with lost sales. The subclass InvSysDiscreteBO assumes backordering and can handle $MOQ > IOQ$. For $IOQ = 1$, the tool uses the procedure described by Zheng & Federgruen (1991) to

determine the distribution of the inventory position just after ordering. For $MOQ > IOQ > 1$, the tool uses the procedure described by Hill (2006). The KPI's related to positive shelf space assume concurrent replenishment from the backroom in case of out of stocks. For deterministic lead times, the results are exact, again with the exception of the KPI $E[ST]$ for review periods greater than 1.

The subclass InvSysDiscreteLS assumes discrete demand distributions and lost sales. In the case of non-perishables and deterministic lead times less or equal to the review period, the results are exact, using Discrete Time Markov Chains (DTMC). For lead times greater than the review period, all results are approximations. As an example, the tool implements the procedure of Bijvank & Johansen (2012) for (R, S) systems.

8 References

Adan I.J.B.F., M.J.A. van Eenige, and J.A.C. Resing (1995). Fitting discrete distributions on the first two moments. *Probability in the Engineering and Informational Sciences* 9; 623-632.

Bijvank, M. and S. J. Johansen (2012). Periodic review lost-sales inventory models with compound Poisson demand and constant lead times of any length. *European Journal of Operational Research* 220, 106-114.

Broekmeulen, R.A.C.M., M.G. Sternbeck, K.H. van Donselaar and H. Kuhn (2017). Decision support for selecting the optimal product unpacking location in a retail supply chain. *European Journal of Operational Research* 259, 84-99.

Broekmeulen, R.A.C.M., van Donselaar, K.H. (2019), Quantifying the potential to improve on food waste, freshness and sales for perishables in supermarkets. *International Journal of Production Economics*, 209, 265-273.

Donselaar, K.H. van, and R.A.C.M. Broekmeulen (2012). Approximations for the relative outdating of perishable products by combining stochastic modeling, simulation and regression modeling. *International Journal of Production Economics* 140(2), 660-669.

Donselaar, K.H. van, and R.A.C.M. Broekmeulen (2014). Stochastic inventory models for a single item at a single location: Lecture Notes and Toolbox for the course Stochastic Operations Management 1CV60, *Beta Working Paper 447*, Eindhoven.

Hill, R.M. (2006). Inventory control with indivisible units of stock transfer. *European Journal of Operational Research* 175, 593-601.

Law, A.M., Kelton W.D. (2000). Simulation modeling and analysis (third ed.). Boston: McGraw-Hill.

Zheng, Y-S, and Federgruen A. (1991) Finding optimal (s,S) policies is about as simple as evaluating a single policy. *Operations Research*, 39(4), 654-665.

A1: Error codes

The following error codes can be returned:

- ✓ -9900 : "Internal inconsistency",
- ✓ -9901 : "Negative values in PMF",
- ✓ -9902 : "Sum PMF <> 1!",
- ✓ -9903 : "StDev of demand too low for corresponding mean",
- ✓ -9904 : "Negative lead time not allowed!",
- ✓ -9905 : "Negative stdev lead time not allowed!",
- ✓ -9906 : "Zero or negative review period not allowed!",
- ✓ -9907 : "Zero or negative mean period demand not allowed!",
- ✓ -9908 : "Zero variance period demand not allowed!",
- ✓ -9909 : "Lead time must be less or equal to review period",
- ✓ -9911 : "Negative IOQ not allowed!",
- ✓ -9912 : "Non-integer IOQ not allowed!",
- ✓ -9913 : "Zero or negative Minimal Order Quantity (MOQ) not allowed!",
- ✓ -9914 : "Non-integer Minimal Order Quantity (MOQ) not allowed!",
- ✓ -9915 : "MOQ must be a integer multiple of the IOQ!",
- ✓ -9925 : "Unit loads less than zero are not allowed!",
- ✓ -9926 : "Unit loads must be a integer multiple of the IOQ!",
- ✓ -9927 : "Negative capacity not allowed!",
- ✓ -9928 : "Zero or negative target not allowed!",
- ✓ -9929 : "Target must be less than 1.0!",
- ✓ -9980 : "No zreg_fifo file found",
- ✓ -9981 : "Singular matrix for IP distribution",
- ✓ -9982 : "Singular matrix for IOH distribution",
- ✓ -9930 : "Shelflife must be greater than the review period!",
- ✓ -9931 : "Non-integer shelflife not allowed!",
- ✓ -9999 : "Not yet implemented/available"

A2: MIT license