On quantiles rating with multiple incommensurable criteria

Raymond Bisdorff

Université du Luxembourg
MICS ADT Course

May 12, 2020

The rating decision problem

- Besides *selecting* and *ranking*, the omnipresent decision problem nowadays appears to be the *rating* decision problem.
- The *rating* problem consists in *partitioning* the set of potential decision alternatives into several, usually ordered, performance categories, the definition of these categories being *intrinsic*.
- The essential *distinctive charateristics* of this kind of decision problem therefore lie in the actual definition of the rating categories.

The rating decision problem

- Besides *selecting* and *ranking*, the omnipresent decision problem nowadays appears to be the *rating* decision problem.
- The *rating* problem consists in *partitioning* the set of potential decision alternatives into several, usually ordered, performance categories, the definition of these categories being *intrinsic*.
- The essential *distinctive charateristics* of this kind of decision problem therefore lie in the actual definition of the rating categories.

The rating decision problem

- Besides *selecting* and *ranking*, the omnipresent decision problem nowadays appears to be the *rating* decision problem.
- The *rating* problem consists in *partitioning* the set of potential decision alternatives into several, usually ordered, performance categories, the definition of these categories being *intrinsic*.
- The essential *distinctive charateristics* of this kind of decision problem therefore lie in the actual definition of the rating categories.

Rating via supervised clustering or sorting

There exist two operational approaches for soving a rating problem:

1. The rating categories do not explicitely refer to the actual *desirability* of the decision alternatives. Many rating problems in *pattern* and *speech recognition* or *diagnosis* are of this kind and may be solved with *classification* or *supervised clustering* algorithms. Rating categories are here usually defined with *prototypical* elements and rating is operated with the help of some *proximity measures*.
2. In the outranking approach, we rely instead precisely on the *desirability* of the alternatives; e.g., a credit manager may want to isolate *good* and *bad* risks, an academic department head may wish to enroll only *good* students. A crucial problem lies here in the definition of these *relational rating categories*. Here we are going to use *quantiles sorting*, e.g. *order statistics* based sorting algorithms taking into account multiple criteria performance tableaux.

Rating via supervised clustering or sorting

There exist two operational approaches for soving a rating problem:

1. The rating categories do not explicitly refer to the actual *desirability* of the decision alternatives. Many rating problems in *pattern* and *speech recognition* or *diagnosis* are of this kind and may be solved with *classification or supervised clustering* algorithms. Rating categories are here usually defined with *prototypical* elements and rating is operated with the help of some *proximity measures*.

2. In the outranking approach, we rely instead precisely on the *desirability* of the alternatives; e.g., a credit manager may want to isolate *good* and *bad* risks, an academic department head may wish to enroll only *good* students. A crucial problem lies here in the definition of these *relational rating categories*. Here we are going to use *quantiles sorting*, e.g. *order statistics* based sorting algorithms taking into account multiple criteria performance tableaux.

Relative versus absolute rating norms

There exist two potential approaches for defining ordered rating categories:

1. *Relative* rating categories: The rating categories are only defined with respect to a given performance tableau and may change with each instance;
2. *Absolute* rating *norms*: The rating norms are defined in general and may apply to all performance tableaux of the same kind, e.g. *star rating* of hotels and restaurants, or *credit rating* of countries or companies.

Relative versus absolute rating norms

There exist two potential approaches for defining ordered rating categories:

1. *Relative* rating categories: The rating categories are only defined with respect to a given performance tableau and may change with each instance;
2. *Absolute* rating *norms*: The rating norms are defined in general and may apply to all performance tableaux of the same kind, e.g. *star rating* of hotels and restaurants, or *credit rating* of countries or companies.

Contents

Performance Quantiles

- Let X be the set of n potential decision alternatives evaluated on a single real performance criteria.
- We denote $x, y, \ldots$ the performances observed of the potential decision actions in X.
- We call quantile $q(p)$ the performance such that $p\%$ of the observed n performances in X are less or equal to $q(p)$.
- The quantile $q(p)$ is estimated by linear interpolation from the cumulative distribution of the performances in X.

Performance Quantiles

- Let X be the set of n potential decision alternatives evaluated on a single real performance criteria.
- We denote $x, y, \ldots$ the performances observed of the potential decision actions in X.
- We call quantile $q(p)$ the performance such that $p\%$ of the observed n performances in X are less or equal to $q(p)$.
- The quantile $q(p)$ is estimated by linear interpolation from the cumulative distribution of the performances in X.

Performance Quantiles

- Let X be the set of n potential decision alternatives evaluated on a single real performance criteria.
- We denote $x, y, \ldots$ the performances observed of the potential decision actions in X.
- We call quantile $q(p)$ the performance such that $p\%$ of the observed n performances in X are less or equal to $q(p)$.
- The quantile $q(p)$ is estimated by linear interpolation from the cumulative distribution of the performances in X.

Performance Quantiles

- Let X be the set of n potential decision alternatives evaluated on a single real performance criteria.
- We denote $x, y, \ldots$ the performances observed of the potential decision actions in X.
- We call quantile $q(p)$ the performance such that $p\%$ of the observed n performances in X are less or equal to $q(p)$.
- The quantile $q(p)$ is estimated by linear interpolation from the cumulative distribution of the performances in X.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, \dots q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, \dots, 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, \dots, q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $]-\infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$, for $k = 1, \dots, q-1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}); +\infty[$.
- We call q-tiles a complete series of $k = 1, \dots, q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q+1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $]-\infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$, for $k = 1, ..., q-1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}); +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $]-\infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$,for $k = 1, ..., q-1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}), +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $]-\infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$,for $k = 1, ..., q-1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}), +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $] -\infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$,for $k = 1, ..., q - 1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}), +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $] - \infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$,for $k = 1, ..., q - 1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}), +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Performance Quantile Classes

- We consider a series: $p_k = k/q$ for $k = 0, ...q$ of $q + 1$ equally spaced quantiles like
  - quartiles: $0.00, 0.25, 0.50, 0.75, 1.00,$
  - quintiles: $0.00, 0.20, 0.40, 0.60, 0.80, 1.00,$
  - deciles: $0.00, 0.10, 0.20, ..., 0.90, 1.00,$ etc
- The upper-closed q^k class corresponds to the interval $]q(p_{k-1}); q(p_k)]$, for $k = 2, ..., q$, where $q(p_q) = \max(X)$ and the first class gathers all data below p_1: $] - \infty; q(p_1)]$.
- The lower-closed q_k class corresponds to the interval $[q(p_{k-1}); q(p_k)[$,for $k = 1, ..., q - 1$, where $q(p_0) = \min(X)$ and the last class gathers all data above $q(p_{q-1})$: $[q(p_{q-1}), +\infty[$.
- We call q-tiles a complete series of $k = 1, ..., q$ upper-closed q^k, resp. lower-closed q_k, quantile classes.

Example

Let us consider the following 31 random performances:

1.10	6.93	8.59	20.97	22.16	24.18	25.39	27.13
32.10	32.23	33.53	34.59	38.65	41.41	41.89	44.87
45.03	50.72	50.96	54.43	58.53	59.82	61.68	62.48
64.82	65.65	71.99	80.73	87.84	87.89	91.56	-

measured on a real scale from 0.0 to 100.0.

5-tiles class limits:

k	p_k	$[q(p_k), -[$	$]-, q(p_k)]$
0	0.0	$[1.10 - [$	$] - \infty]$
1	0.2	$[25.74 - [$	$] - 25.74]$
2	0.4	$[39.75 - [$	$] - 39.75]$
3	0.6	$[53.04 - [$	$] - 53.04]$
4	0.8	$[65.48 - [$	$] - 65.48]$
5	1.0	$[+\infty$	$] - 91.56]$

5-tiles class contents:

q_k class	q^k class	#
$[0.8; +\infty[$	$]0.8; 1.0]$	6
$[0.6; 0.8[$	$]0.6; 0.8]$	6
$[0.4; 0.6[$	$]0.4; 0.6]$	6
$[0.2; 0.4[$	$]0.2; 0.4]$	6
$[0.0; 0.2[$	$]-\infty; 0.2]$	7

Example

Let us consider the following 31 random performances:

1.10	6.93	8.59	20.97	22.16	24.18	25.39	27.13
32.10	32.23	33.53	34.59	38.65	41.41	41.89	44.87
45.03	50.72	50.96	54.43	58.53	59.82	61.68	62.48
64.82	65.65	71.99	80.73	87.84	87.89	91.56	-

measured on a real scale from 0.0 to 100.0.

5-tiles class limits:

k	p_k	$[q(p_k), \_[$	$]\_, q(p_k)]$
0	0.0	$[1.10 - [$	$] - \infty]$
1	0.2	$[25.74 - [$	$] - 25.74]$
2	0.4	$[39.75 - [$	$] - 39.75]$
3	0.6	$[53.04 - [$	$] - 53.04]$
4	0.8	$[65.48 - [$	$] - 65.48]$
5	1.0	$[+\infty$	$] - 91.56]$

5-tiles class contents:

q_k class	q^k class	#
$[0.8; +\infty[$	$]0.8; 1.0]$	6
$[0.6; 0.8[$	$]0.6; 0.8]$	6
$[0.4; 0.6[$	$]0.4; 0.6]$	6
$[0.2; 0.4[$	$]0.2; 0.4]$	6
$[0.0; 0.2[$	$]-\infty; 0.2]$	7

Example

Let us consider the following 31 random performances:

1.10	6.93	8.59	20.97	22.16	24.18	25.39	27.13
32.10	32.23	33.53	34.59	38.65	41.41	41.89	44.87
45.03	50.72	50.96	54.43	58.53	59.82	61.68	62.48
64.82	65.65	71.99	80.73	87.84	87.89	91.56	-

measured on a real scale from 0.0 to 100.0.

5-tiles class limits:

k	p_k	$[q(p_k), \_[$	$]\_, q(p_k)]$
0	0.0	$[1.10 - [$	$] - \infty]$
1	0.2	$[25.74 - [$	$] - 25.74]$
2	0.4	$[39.75 - [$	$] - 39.75]$
3	0.6	$[53.04 - [$	$] - 53.04]$
4	0.8	$[65.48 - [$	$] - 65.48]$
5	1.0	$[+\infty$	$] - 91.56]$

5-tiles class contents:

q_k class	q^k class	#
$[0.8; +\infty[$	$]0.8; 1.0]$	6
$[0.6; 0.8[$	$]0.6; 0.8]$	6
$[0.4; 0.6[$	$]0.4; 0.6]$	6
$[0.2; 0.4[$	$]0.2; 0.4]$	6
$[0.0; 0.2[$	$]-\infty; 0.2]$	7

Example

Let us consider the following 31 random performances:

1.10	6.93	8.59	20.97	22.16	24.18	25.39	27.13
32.10	32.23	33.53	34.59	38.65	41.41	41.89	44.87
45.03	50.72	50.96	54.43	58.53	59.82	61.68	62.48
64.82	65.65	71.99	80.73	87.84	87.89	91.56	-

measured on a real scale from 0.0 to 100.0.

5-tiles class limits:

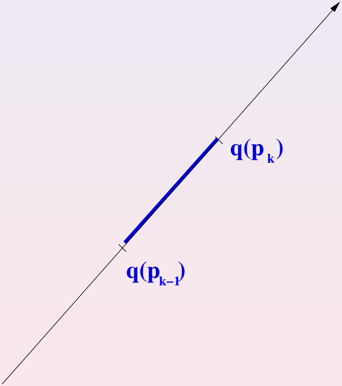
k	p_k	$[q(p_k), \_[$	$]\_, q(p_k)]$
0	0.0	$[1.10 - [$	$] - \infty]$
1	0.2	$[25.74 - [$	$] - 25.74]$
2	0.4	$[39.75 - [$	$] - 39.75]$
3	0.6	$[53.04 - [$	$] - 53.04]$
4	0.8	$[65.48 - [$	$] - 65.48]$
5	1.0	$[+\infty$	$] - 91.56]$

5-tiles class contents:

q_k class	q^k class	#
$[0.8; +\infty[$	$]0.8; 1.0]$	6
$[0.6; 0.8[$	$]0.6; 0.8]$	6
$[0.4; 0.6[$	$]0.4; 0.6]$	6
$[0.2; 0.4[$	$]0.2; 0.4]$	6
$[0.0; 0.2[$	$] - \infty; 0.2]$	7

Upper-closed q-tiles sorting on a single criterion

If x is a measured performance, we may distinguish three sorting situations:

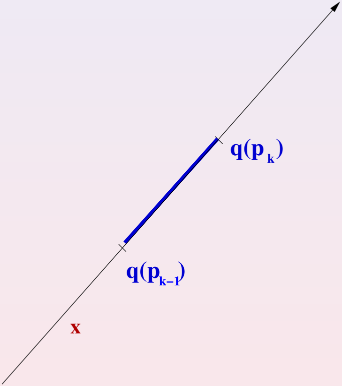


1. $x \leqslant q(p_{k-1})$ and $x < q(p_k)$
   The performance x is lower than the q^k class;
2. $x > q(p_{k-1})$ and $x \leqslant q(p_k)$
   The performance x belongs to the q^k class;
3. ($x > q(p_{k-1})$ and) $x > q(p_k)$
   The performance x is higher than the p^k class.

If the relation $<$ is the dual of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

Upper-closed q-tiles sorting on a single criterion

If x is a measured performance, we may distinguish three sorting situations:

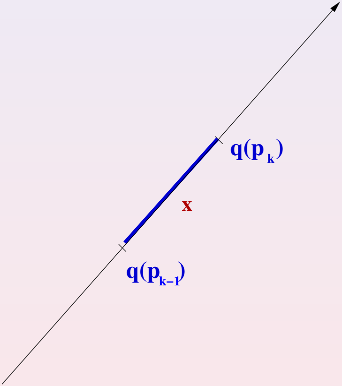


1. $x \leqslant q(p_{k-1})$ and $x < q(p_k)$
   The performance x is lower than the q^k class;
2. $x > q(p_{k-1})$ and $x \leqslant q(p_k)$
   The performance x belongs to the q^k class;
3. $(x > q(p_{k-1})$ and$)$
   $x > q(p_k)$
   The performance x is higher than the p^k class.

If the relation $<$ is the dual of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

Upper-closed q-tiles sorting on a single criterion

If x is a measured performance, we may distinguish three sorting situations:

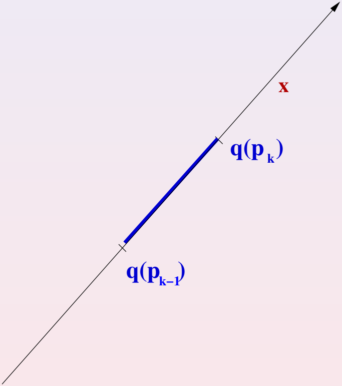


1. $x \leqslant q(p_{k-1})$ and $x < q(p_k)$
   The performance x is lower than the q^k class;
2. $x > q(p_{k-1})$ and $x \leqslant q(p_k)$
   The performance x belongs to the q^k class;
3. $(x > q(p_{k-1})$ and$)$
   $x > q(p_k)$
   The performance x is higher than the p^k class.

If the relation $<$ is the dual of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

Upper-closed q-tiles sorting on a single criterion

If x is a measured performance, we may distinguish three sorting situations:

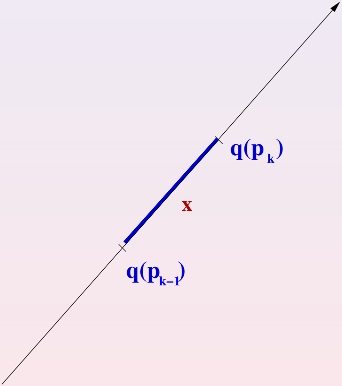


1. $x \leqslant q(p_{k-1})$ and $x < q(p_k)$
   The performance x is lower than the q^k class;
2. $x > q(p_{k-1})$ and $x \leqslant q(p_k)$
   The performance x belongs to the q^k class;
3. $(x > q(p_{k-1})$ and$)$
   $x > q(p_k)$
   The performance x is higher than the p^k class.

If the relation $<$ is the dual of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

Upper-closed q-tiles sorting on a single criterion

If x is a measured performance, we may distinguish three sorting situations:



1. $x \leqslant q(p_{k-1})$ and $x < q(p_k)$
   The performance x is lower than the q^k class;
2. $x > q(p_{k-1})$ and $x \leqslant q(p_k)$
   The performance x belongs to the q^k class;
3. $(x > q(p_{k-1})$ and$)$
   $x > q(p_k)$
   The performance x is higher than the p^k class.

If the relation $<$ is the dual of $\geqslant$, it will be sufficient to check that both, $q(p_{k-1}) \not\geqslant x$, as well as $q(p_k) \geqslant x$, are verified for x to be a member of the k-th q-tiles class.

Taking into account imprecise evaluations – I

Suppose now we acknowledge two preference discrimination thresholds:

1. An indifference threshold *ind* of 10.0 pts, modelling the maximal numerical performance difference which is considered preferentially insignificant;
2. A preference threshold *pr* of 20.0 pts ($pr > ind$), modelling the smallest numerical performance which is considered preferentially significant.

Taking into account imprecise evaluations – I

Suppose now we acknowledge two preference discrimination thresholds:

1. An indifference threshold *ind* of 10.0 pts, modelling the maximal numerical performance difference which is considered preferentially insignificant;
2. A preference threshold *pr* of 20.0 pts ($pr > ind$), modelling the smallest numerical performance which is considered preferentially significant.

Taking into account imprecise evaluations – I

Suppose now we acknowledge two preference discrimination thresholds:

1. An indifference threshold *ind* of 10.0 pts, modelling the maximal numerical performance difference which is considered preferentially insignificant;
2. A preference threshold *pr* of 20.0 pts ($pr > ind$), modelling the smallest numerical performance which is considered preferentially significant.

Taking into account imprecise evaluations – II

Example (Upper-closed 5-tiles sorting with preference threshold)

1.1	6.9	8.6	21.0	22.2	24.2	25.4	27.1
32.1	32.2	33.5	34.6	38.6	41.4	41.9	44.9
45.0	50.7	51.0	54.4	58.5	59.8	61.7	62.5
64.8	65.7	72.0	80.7	87.8	87.9	91.6	-

Adapted 5-tiles class limits:

k	p_k	$q(p_k)$
1	0.2	25.74 – 20*
2	0.4	39.75 – 20
3	0.6	53.04 – 20
4	0.8	65.48 – 20
5	1.0	91.56

*Preference threshold: 20.0

Taking into account imprecise evaluations – II

Example (Upper-closed 5-tiles sorting with preference threshold)

1.1	6.9	8.6	21.0	22.2	24.2	25.4	27.1
32.1	32.2	33.5	34.6	38.6	41.4	41.9	44.9
45.0	50.7	51.0	54.4	58.5	59.8	61.7	62.5
64.8	65.7	72.0	80.7	87.8	87.9	91.6	-

Adapted 5-tiles class limits:

k	p_k	$q(p_k)$
1	0.2	25.74 – 20*
2	0.4	39.75 – 20
3	0.6	53.04 – 20
4	0.8	65.48 – 20
5	1.0	91.56

*Preference threshold: 20.0

Taking into account imprecise evaluations – II

Example (Upper-closed 5-tiles sorting with preference threshold)

1.1	6.9	8.6	21.0	22.2	24.2	25.4	27.1
32.1	32.2	33.5	34.6	38.6	41.4	41.9	44.9
45.0	50.7	51.0	54.4	58.5	59.8	61.7	62.5
64.8	65.7	72.0	80.7	87.8	87.9	91.6	-

Adapted 5-tiles class limits:

k	p_k	$q(p_k)$
1	0.2	25.74 – 20*
2	0.4	39.75 – 20
3	0.6	53.04 – 20
4	0.8	65.48 – 20
5	1.0	91.56

*Preference threshold: 20.0

Taking into account imprecise evaluations – II

Example (Upper-closed 5-tiles sorting with preference threshold)

1.1	6.9	8.6	21.0	22.2	24.2	25.4	27.1
32.1	32.2	33.5	34.6	38.6	41.4	41.9	44.9
45.0	50.7	51.0	54.4	58.5	59.8	61.7	62.5
64.8	65.7	72.0	80.7	87.8	87.9	91.6	-

Adapted 5-tiles class limits:

k	p_k	$q(p_k)$
1	0.2	25.74 – 20*
2	0.4	39.75 – 20
3	0.6	53.04 – 20
4	0.8	65.48 – 20
5	1.0	91.56

*Preference threshold: 20.0

Resulting 5-tiles sorting:

q-tiles class	values
$]0.0 - 0.2]$	$\{1.1, 6.9, 8.6\}$
$]0.0 - 0.4]$	$\{21.0, 22.2, 24.2, 25.4\}$
$]0.2 - 0.4]$	$\{27.1\}$
$]0.2 - 0.6]$	$\{32.1, 32.2, 33.5, 34.6, 38.6\}$
$]0.4 - 0.6]$	$\{41.4, 41.9\}$
$]0.4 - 0.8]$	$\{44.9, 45.0, 50.7, 51.0\}$
$]0.6 - 0.8]$	$\{54.43\}$
$]0.6 - 1.0]$	$\{58.5, 59.8, 61.7, 62.5, 64.8\}$
$]0.8 - 1.0]$	$\{65.7, 72.0, 80.7, 87.8, 87.9, 91.6\}$

Taking into account imprecise evaluations – II

Example (Upper-closed 5-tiles sorting with preference threshold)

1.1	6.9	8.6	21.0	22.2	24.2	25.4	27.1
32.1	32.2	33.5	34.6	38.6	41.4	41.9	44.9
45.0	50.7	51.0	54.4	58.5	59.8	61.7	62.5
64.8	65.7	72.0	80.7	87.8	87.9	91.6	-

Adapted 5-tiles class limits:

k	p_k	$q(p_k)$
1	0.2	25.74 – 20*
2	0.4	39.75 – 20
3	0.6	53.04 – 20
4	0.8	65.48 – 20
5	1.0	91.56

*Preference threshold: 20.0

Resulting 5-tiles sorting:

q-tiles class	values
$]0.0 - 0.2]$	$\{1.1, 6.9, 8.6\}$
$]0.0 - 0.4]$	$\{21.0, 22.2, 24.2, 25.4\}$
$]0.2 - 0.4]$	$\{27.1\}$
$]0.2 - 0.6]$	$\{32.1, 32.2, 33.5, 34.6, 38.6\}$
$]0.4 - 0.6]$	$\{41.4, 41.9\}$
$]0.4 - 0.8]$	$\{44.9, 45.0, 50.7, 51.0\}$
$]0.6 - 0.8]$	$\{54.43\}$
$]0.6 - 1.0]$	$\{58.5, 59.8, 61.7, 62.5, 64.8\}$
$]0.8 - 1.0]$	$\{65.7, 72.0, 80.7, 87.8, 87.9, 91.6\}$

Multiple criteria extension

- $X = \{x, y, z, ...\}$ is a finite set of n objects to be sorted.
- $F = \{1, ..., m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, ...\}$ is a finite set of n objects to be sorted.
- $F = \{1, ..., m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, ...\}$ is a finite set of n objects to be sorted.
- $F = \{1, ..., m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, ...\}$ is a finite set of n objects to be sorted.
- $F = \{1, ..., m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, \ldots\}$ is a finite set of n objects to be sorted.
- $F = \{1, \ldots, m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, ...\}$ is a finite set of n objects to be sorted.
- $F = \{1, ..., m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

Multiple criteria extension

- $X = \{x, y, z, \dots\}$ is a finite set of n objects to be sorted.
- $F = \{1, \dots, m\}$ is a finite and coherent family of m performance criteria.
- For each criterion j in F, the objects are evaluated on a real performance scale $[0; M_j]$,

  supporting an indifference threshold ind_j

  and a preference threshold pr_j such that $0 \leqslant ind_j < pr_j \leqslant M_j$.
- The performance of object x on criterion j is denoted x_j.
- Each criterion j in F carries a rational significance w_j such that $0 < w_j < 1.0$ and $\sum_{j \in F} w_j = 1.0$.

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if
   2.1 a significant majority of criteria invalidates a global outranking situation between x and y, and
   2.2 [illegible] is observed on a concordant criterion.

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if
   2.1 a significant majority of criteria invalidates a global outranking situation between x and y, and
   2.2 no counter-veto is observed on a concordant criterion.

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if
   2.1 a significant majority of criteria invalidates a global outranking situation between x and y, and
   2.2 no counter-veto is observed on a concordant criterion.

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if
   2.1 a significant majority of criteria invalidates a global outranking situation between x and y, and
   2.2 no counter-veto is observed on a concordant criterion.

The bipolar outranking relation $\succsim$

From an epistemic point of view, we say that:

1. object x *outranks* object y, denoted $(x \succsim y)$, if
   1.1 a significant majority of criteria validates a global outranking situation between x and y, and
   1.2 no veto is observed on a discordant criterion,
2. object x *does not outrank* object y, denoted $(x \not\succsim y)$, if
   2.1 a significant majority of criteria invalidates a global outranking situation between x and y, and
   2.2 no counter-veto is observed on a concordant criterion.

q-tiles sorting with bipolar outrankings

Property

The bipolar characteristic of x belonging to upper-closed q-tiles class q^k, resp. lower-closed class q_k, may hence, in a multiple criteria outranking approach, be assessed as follows:

$$r(x \in q^k) = \min\Big[-r\big(\mathrm{q}(p_{k-1}) \succsim x\big),\ r\big(\mathrm{q}(p_k) \succsim x\big)\Big]$$

$$r(x \in q_k) = \min\Big[r\big(x \succsim \mathrm{q}(p_{k-1})\big),\ -r\big(x \succsim \mathrm{q}(p_k)\big)\Big]$$

The bipolar outranking relation $\succsim$, being weakly complete, verifies the coduality principle (Bisdorff 2013). Hence:

$$-r\big(\mathrm{q}(p_{k-1}) \succsim x\big) = r\big(\mathrm{q}(p_{k-1}) \not\succsim x\big) = r\big(\mathrm{q}(p_{k-1}) \prec x\big),$$

$$-r\big(x \succsim \mathrm{q}(p_k)\big) = \big(x \not\succsim \mathrm{q}(p_k)\big) = r\big(x \prec \mathrm{q}(p_k)\big).$$

q-tiles sorting with bipolar outrankings

Property

The bipolar characteristic of x belonging to upper-closed q-tiles class q^k, resp. lower-closed class q_k, may hence, in a multiple criteria outranking approach, be assessed as follows:

$$r(x \in q^k) = \min \left[-r\big(\mathrm{q}(p_{k-1}) \succsim x\big),\ r\big(\mathrm{q}(p_k) \succsim x\big) \right]$$

$$r(x \in q_k) = \min \left[r\big(x \succsim \mathrm{q}(p_{k-1})\big),\ -r\big(x \succsim \mathrm{q}(p_k)\big) \right]$$

The bipolar outranking relation $\succsim$, being weakly complete, verifies the coduality principle (Bisdorff 2013). Hence:

$$-r\big(\mathrm{q}(p_{k-1}) \succsim x\big) = r\big(\mathrm{q}(p_{k-1}) \not\succsim x\big) = r\big(\mathrm{q}(p_{k-1}) \prec x\big),$$

$$-r\big(x \succsim \mathrm{q}(p_k)\big) = \big(x \not\succsim \mathrm{q}(p_k)\big) = r\big(x \prec \mathrm{q}(p_k)\big).$$

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x)\big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x)\big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x)\big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x) \big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

1. *The complexity of the q-tiles sorting algorithm is $\mathcal{O}(nmq)$; linear in the number of decision actions (n), criteria (m) and quantile classes (q).*
2. *As $\mathcal{Q}$ represents a partition of the criteria measurement scales, i.e. the upper limits of the preceding category correspond to the lower limits of the succeeding ones, there is a potential for run time optimization.*

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x) \big)$
   2.2 if $r(x \in q^k) \;\geqslant\; 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

1. *The complexity of the q-tiles sorting algorithm is $O(nmq)$; linear in the number of decision actions (n), criteria (m) and quantile classes (q).*
2. *As $\mathcal{Q}$ represents a partition of the criteria measurement scales, i.e. the upper limits of the preceding category correspond to the lower limits of the succeeding ones, there is a potential for run time optimization.*

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x) \big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

1. *The complexity of the q-tiles sorting algorithm is $\mathcal{O}(nmq)$; linear in the number of decision actions (n), criteria (m) and quantile classes (q).*
2. *As $\mathcal{Q}$ represents a partition of the criteria measurement scales, i.e. the upper limits of the preceding category correspond to the lower limits of the succeeding ones, there is a potential for run time optimization.*

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x)\big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

1. *The complexity of the q-tiles sorting algorithm is $\mathcal{O}(nmq)$; linear in the number of decision actions (n), criteria (m) and quantile classes (q).*
2. *As $\mathcal{Q}$ represents a partition of the criteria measurement scales, i.e. the upper limits of the preceding category correspond to the lower limits of the succeeding ones, there is a potential for run time optimization.*

The multicriteria (upper-closed) q-tiles sorting algorithm

1. **Input**: a set X of n objects with a performance table on a family of m criteria and a set $\mathcal{Q}$ of $k = 1, .., q$ empty q-tiles equivalence classes.
2. **For each** object $x \in X$ **and each** q-tiles class $q^k \in \mathcal{Q}$
   2.1 $r(x \in q^k) \quad \leftarrow \quad \min\big(-r(\mathrm{q}(p_{k-1}) \succsim x), r(\mathrm{q}(p_k) \succsim x)\big)$
   2.2 if $r(x \in q^k) \geqslant 0$:
   add x to q-tiles class q^k
3. **Output**: $\mathcal{Q}$

Comment

1. *The complexity of the q-tiles sorting algorithm is $\mathcal{O}(nmq)$; linear in the number of decision actions (n), criteria (m) and quantile classes (q).*
2. *As $\mathcal{Q}$ represents a partition of the criteria measurement scales, i.e. the upper limits of the preceding category correspond to the lower limits of the succeeding ones, there is a potential for run time optimization.*

Properties of q-tiles sorting result

1. *Coherence*: Each object is always sorted into a non-empty subset of adjacent q-tiles classes.
2. *Uniqueness*: If the q-tiles classes represent a discriminated partition of the measurement scales on each criterion and $r \neq 0$, then every object is sorted into exactly one q-tiles class.
3. *Independence*: The sorting result for object x, is independent of the other object's sorting results.

Comment

The independence property gives us access to efficient parallel processing of class membership characteristics $r(x \in q^k)$ for all $x \in X$ and q^k in Q.

Properties of q-tiles sorting result

1. *Coherence*: Each object is always sorted into a non-empty subset of adjacent q-tiles classes.
2. *Uniqueness*: If the q-tiles classes represent a discriminated partition of the measurement scales on each criterion and $r \neq 0$, then every object is sorted into exactly one q-tiles class.
3. *Independence*: The sorting result for object x, is independent of the other object's sorting results.

Comment

The independence property gives us access to efficient parallel processing of class membership characteristics $r(x \in q^k)$ for all $x \in X$ and q^k in Q.

Properties of q-tiles sorting result

1. *Coherence*: Each object is always sorted into a non-empty subset of adjacent q-tiles classes.
2. *Uniqueness*: If the q-tiles classes represent a discriminated partition of the measurement scales on each criterion and $r \neq 0$, then every object is sorted into exactly one q-tiles class.
3. *Independence*: The sorting result for object x, is independent of the other object's sorting results.

Comment

The independence property gives us access to efficient parallel processing of class membership characteristics $r(x \in q^k)$ for all $x \in X$ and q^k in $\mathcal{Q}$.

Properties of q-tiles sorting result

1. *Coherence*: Each object is always sorted into a non-empty subset of adjacent q-tiles classes.
2. *Uniqueness*: If the q-tiles classes represent a discriminated partition of the measurement scales on each criterion and $r \neq 0$, then every object is sorted into exactly one q-tiles class.
3. *Independence*: The sorting result for object x, is independent of the other object's sorting results.

Comment

The independence property gives us access to efficient parallel processing of class membership characteristics $r(x \in q^k)$ *for all* $x \in X$ *and* q^k *in* $\mathcal{Q}$.

Quantiles sorting example

- 34 top European Universities;
- Assessed on five cardinal criteria (measured as *z*-scores):
  1. Teaching: quality of the learning environment ($w_T = 3$),
  2. Citations: research influence ($w_C = 3$),
  3. Research: volume, income and reputation ($w_R = 1$),
  4. International outlook ($w_I = 1$),
  5. Industry income: innovation ($w_{Ind} = 1$).
- *Source*: Times Higher Education University Rankings 2010

```
>>> from perfTabs import PerformanceTableau
>>> t = PerformanceTableau('theRanking2010')
>>> t.showHTMLPerformanceHeatmap(colorLevels=5,\
      rankingRule=None,pageTitle=\
         'Performance Tableau \'theRanking10\'')
```

Extract from an unordered heatmap view

Performance Tableau 'theRanking10'

criteria	c-T	c_C	c-Ind	c_I	c_R
weights	+3.00	+3.00	+1.00	+1.00	+1.00
DU-UK	39.80	91.90	33.90	65.70	44.10
ENSL-FR	51.10	88.80	26.10	37.60	34.40
ENSP-FR	66.80	95.70	30.70	44.90	48.20
EP-FR	57.90	91.40	NA	77.90	56.10
EPFL-CH	55.00	83.80	38.00	100.00	56.10
ETHZ-CH	77.50	83.10	NA	93.70	87.80
EUT-NL	55.40	56.90	99.80	44.90	51.70
ICL-UK	89.20	88.30	92.90	90.00	94.50
KCL-UK	48.50	72.40	44.10	85.90	54.50
KI-S	65.80	62.30	31.70	NA	72.70
KUL-BE	57.70	45.20	97.70	29.60	62.90
LSE-UK	62.40	51.60	38.40	99.50	56.20
LU-S	46.30	67.60	33.20	56.80	60.80
RHL-UK	37.70	93.20	30.50	92.90	36.20
RKU-DE	59.20	70.30	39.10	63.40	47.50
TCD-IR	47.70	84.40	31.60	84.20	45.30
TUM-DE	50.40	71.20	NA	85.30	43.20
UB-CH	42.40	78.30	45.80	91.30	37.10

The 17-tiles sorting of the THE University ranking data

```
>>> from sortingDigraphs import QuantilesSortingDigraph
>>> qs = QuantilesSortingDigraph(t,limitingQuantiles=17,LowerClosed=False)
>>> qs
*-----  Object instance description -----------*
Instance class      : QuantilesSortingDigraph
Instance name       : sorting_with_17-tile_limits
# # Actions         : 34
# Criteria          : 5
# Categories        : 17
Lowerclosed         : False
Size                : 747
Valuation domain    : [-1.00;1.00]
Determinateness (%) : 103.40
Attributes          : ['actions', 'actionsOrig', 'criteria', 'evaluation',
                       'runTimes', 'name', 'limitingQuantiles', 'LowerClosed',
                       'categories', 'criteriaCategoryLimits', 'profiles',
                       'profileLimits', 'hasNoVeto', 'valuationdomain',
                       'nbrThreads', 'relation', 'categoryContent', 'order',
                       'gamma', 'notGamma', 'quantiles']
```

The 17-tiles sorting of the THE University ranking data

```
>>> qs.showSorting()
```

]0.94 - 1.00]:	[]
]0.88 - 0.94]:	['ICL-UK']
]0.82 - 0.88]:	['ETHZ-CH', 'ICL-UK', 'UO-UK']
]0.76 - 0.82]:	['ETHZ-CH', 'EUT-NL', 'KUL-BE', 'UC-UK', 'UO-UK']
]0.71 - 0.76]:	['ENSP-FR', 'ETHZ-CH', 'EUT-NL', 'KI-S', 'KUL-BE']
]0.65 - 0.71]:	['ENSP-FR', 'EUT-NL', 'KI-S', 'KUL-BE', 'UCL-UK']
]0.59 - 0.65]:	['EUT-NL', 'KI-S', 'KUL-BE', 'UCL-UK']
]0.53 - 0.59]:	['EUT-NL', 'KI-S', 'KUL-BE', 'UCL-UK', 'UE-UK']
]0.47 - 0.53]:	['EP-FR', 'EUT-NL', 'KI-S', 'KUL-BE', 'LSE-UK', 'UE-UK', 'UG-DE']
]0.41 - 0.47]:	['EPFL-CH', 'EUT-NL', 'KI-S', 'KUL-BE', 'LSE-UK', 'UCD-IR', 'UG-DE', 'UM-DE', 'UM-UK', 'UZ-CH']
]0.35 - 0.41]:	['EUT-NL', 'KI-S', 'UCD-IR', 'UM-DE']
]0.29 - 0.35]:	['ENSL-FR', 'EUT-NL', 'KI-S', 'UB-UK', 'UCD-IR']
]0.24 - 0.29]:	['ENSL-FR', 'KI-S', 'UB-CH', 'UB-UK', 'UCD-IR', 'UY-UK']
]0.18 - 0.24]:	['DU-UK', 'ENSL-FR', 'KCL-UK', 'KI-S', 'RKU-DE', 'TUM-DE', 'UG-CH', 'UH-FI', 'USTA-UK', 'USth-UK', 'UY-UK']
]0.12 - 0.18]:	['DU-UK', 'ENSL-FR', 'KI-S', 'LU-S', 'TCD-IR', 'TUM-DE', 'UG-CH']
]0.06 - 0.12]:	['RHL-UK', 'UG-CH', 'US-UK']
]< - 0.06]:	['RHL-UK']

Ordering the *q*-tiles sorting result

We may notice that some universities like 'ETHZ' and 'KIS' are sorted into several adjacent 17-tiles classes and the sorting result leaves us hence with a more or less refined partition of the set of 35 Universities.

The upper-closed 17-tiles sorting shows here 25 such overlapping quantile classes, of which 5 contain more than 1 university (1×5, 1×3, and 3×2 universities).

For linearly ranking from *best to worst* these 25 quantile classes we may apply three strategies:

1. Optimistic: In decreasing lexicographic order of the *upper* and *lower* quantile class limits;
2. Pessimistic: In decreasing lexicographic order of the *lower* and *upper* quantile class limits;
3. *Average* (default): In decreasing numeric order of the *average* of the lower and upper quantile limits. In case of ties, we select first the *highest upper* quantile class.

Ordering the q-tiles sorting result

We may notice that some universities like 'ETHZ' and 'KIS' are sorted into several adjacent 17-tiles classes and the sorting result leaves us hence with a more or less refined partition of the set of 35 Universities.

The upper-closed 17-tiles sorting shows here 25 such overlapping quantile classes, of which 5 contain more than 1 university (1×5, 1×3, and 3×2 universities).

For linearly ranking from *best to worst* these 25 quantile classes we may apply three strategies:

1. Optimistic: In decreasing lexicographic order of the *upper* and *lower* quantile class limits;
2. Pessimistic: In decreasing lexicographic order of the *lower* and *upper* quantile class limits;
3. Average (default): In decreasing numeric order of the average of the lower and upper quantile limits. In case of ties, we select first the *highest upper* quantile class.

Ordering the q-tiles sorting result

We may notice that some universities like 'ETHZ' and 'KIS' are sorted into several adjacent 17-tiles classes and the sorting result leaves us hence with a more or less refined partition of the set of 35 Universities.
The upper-closed 17-tiles sorting shows here 25 such overlapping quantile classes, of which 5 contain more than 1 university (1×5, 1×3, and 3×2 universities).
For linearly ranking from *best to worst* these 25 quantile classes we may apply three strategies:

1. Optimistic: In decreasing lexicographic order of the *upper* and *lower* quantile class limits;
2. Pessimistic: In decreasing lexicographic order of the *lower* and *upper* quantile class limits;
3. Average (default): In decreasing numeric order of the *average* of the lower and upper quantile limits. In case of ties, we select first the *highest upper* quantile class.

Ordering the q-tiles sorting result

We may notice that some universities like 'ETHZ' and 'KIS' are sorted into several adjacent 17-tiles classes and the sorting result leaves us hence with a more or less refined partition of the set of 35 Universities.
The upper-closed 17-tiles sorting shows here 25 such overlapping quantile classes, of which 5 contain more than 1 university (1×5, 1×3, and 3×2 universities).
For linearly ranking from *best to worst* these 25 quantile classes we may apply three strategies:

1. Optimistic: In decreasing lexicographic order of the *upper* and *lower* quantile class limits;
2. Pessimistic: In decreasing lexicographic order of the *lower* and *upper* quantile class limits;
3. Average (default): In decreasing numeric order of the *average* of the lower and upper quantile limits. In case of ties, we select first the *highest upper* quantile class.

Ordering the q-tiles sorting result

We may notice that some universities like 'ETHZ' and 'KIS' are sorted into several adjacent 17-tiles classes and the sorting result leaves us hence with a more or less refined partition of the set of 35 Universities.
The upper-closed 17-tiles sorting shows here 25 such overlapping quantile classes, of which 5 contain more than 1 university (1×5, 1×3, and 3×2 universities).
For linearly ranking from *best to worst* these 25 quantile classes we may apply three strategies:

1. Optimistic: In decreasing lexicographic order of the *upper* and *lower* quantile class limits;
2. Pessimistic: In decreasing lexicographic order of the *lower* and *upper* quantile class limits;
3. Average (default): In decreasing numeric order of the *average* of the lower and upper quantile limits. In case of ties, we select first the *highest upper* quantile class.

The 17-tiles rating result

```
>>> qs.showQuantileOrdering(strategy='average')
```

quantile class	content	quantile class	content
]0.82-0.94] :	['ICL-UK']	]0.24-0.47] :	['UCD-IR']
]0.76-0.88] :	['UO-UK']	]0.24-0.35] :	['UB-UK']
]0.71-0.88] :	['ETHZ-CH']	]0.24-0.29] :	['UB-CH']
]0.76-0.82] :	['UC-UK']	]0.12-0.35] :	['ENSL-FR']
]0.65-0.76] :	['ENSP-FR']	]0.18-0.29] :	['UY-UK']
]0.41-0.82] :	['KUL-BE']	]0.18-0.24] :	['KCL-UK', 'RKU-DE', 'UH-FI', 'USTA-UK', 'USth-UK']
]0.53-0.71] :	['UCL-UK']		
]0.29-0.82] :	['EUT-NL']		
]0.47-0.59] :	['UE-UK']		
]0.47-0.53] :	['EP-FR']		
]0.41-0.53] :	['LSE-UK', 'UG-DE']	]0.12-0.24] :	['DU-UK', 'TUM-DE']
]0.41-0.47] :	['EPFL-CH', 'UM-UK', 'UZ-CH']	]0.06-0.24] :	['UG-CH']
		]0.12-0.18] :	['LU-S', 'TCD-IR']
]0.12-0.76] :	['KI-S']	]0.06-0.12] :	['US-UK']
]0.35-0.47] :	['UM-DE']	]$-\infty$ -0.12] :	['RHL-UK']

Extract from a ranked heatmap view

Performance Tableau 'theRanking10'

criteria	c-T	c_R	c-Ind	c_C	c_I
weights	+3.00	+1.00	+1.00	+3.00	+1.00
tau(*)	+0.63	+0.50	+0.37	+0.14	+0.13
ICL-UK	89.20	94.50	92.90	88.30	90.00
UO-UK	90.50	93.90	73.50	95.10	77.20
UC-UK	88.20	94.10	57.00	94.00	77.70
ETHZ-CH	77.50	87.80	NA	83.10	93.70
UCL-UK	74.00	81.60	39.00	80.60	90.80
EP-FR	57.90	56.10	NA	91.40	77.90
UG-DE	57.30	55.90	73.30	92.50	44.50
UE-UK	59.90	61.90	42.20	86.80	67.30
EPFL-CH	55.00	56.10	38.00	83.80	100.00
ENSP-FR	66.80	48.20	30.70	95.70	44.90
EUT-NL	55.40	51.70	99.80	56.90	44.90
KUL-BE	57.70	62.90	97.70	45.20	29.60
LSE-UK	62.40	56.20	38.40	51.60	99.50
UCD-IR	50.80	36.60	NA	86.30	87.00
KI-S	65.80	72.70	31.70	62.30	NA
UM-DE	59.10	57.50	40.40	76.40	43.10
UZ-CH	56.60	47.00	43.80	65.00	87.90

Contents

Normed (learned) quantiles rating

Decision problem: Rating multiple criteria performances with respect to historical order statistics, i.e. performance quantiles learned from historical data gathered in the past.

Example (How to rate two decision actions – I)

Consider below the multi-criteria performances of two potential decision actions named $a1001$ and $a1010$:

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
$a1001$	37.0	2	2	61.0	31.0	-4	-40.0
$a1010$	32.0	9	6	55.0	51.0	-4	-35.0

Both are evaluated on 7 seven performance criteria: five *Benefits* criteria: $b1$ to $b5$ (objective to *maximize*) and two *Costs* criteria: $c1$ and $c2$ (objective to *minimize*).

Normed (learned) quantiles rating

Decision problem: Rating multiple criteria performances with respect to historical order statistics, i.e. performance quantiles learned from historical data gathered in the past.

Example (How to rate two decision actions – I)

Consider below the multi-criteria performances of two potential decision actions named $a1001$ and $a1010$:

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
$a1001$	37.0	2	2	61.0	31.0	-4	-40.0
$a1010$	32.0	9	6	55.0	51.0	-4	-35.0

Both are evaluated on 7 seven performance criteria: five *Benefits* criteria: $b1$ to $b5$ (objective to *maximize*) and two *Costs* criteria: $c1$ and $c2$ (objective to *minimize*).

Normed (learned) quantiles rating

Decision problem: Rating multiple criteria performances with respect to historical order statistics, i.e. performance quantiles learned from historical data gathered in the past.

Example (How to rate two decision actions – I)

Consider below the multi-criteria performances of two potential decision actions named $a1001$ and $a1010$:

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
$a1001$	37.0	2	2	61.0	31.0	-4	-40.0
$a1010$	32.0	9	6	55.0	51.0	-4	-35.0

Both are evaluated on 7 seven performance criteria: five *Benefits* criteria: $b1$ to $b5$ (objective to *maximize*) and two *Costs* criteria: $c1$ and $c2$ (objective to *minimize*).

Absolute versus relative rating

Example (How to rate two decision actions – II)

- The performances on the *Costs* criterion $c2$ are measured on an ordinal negative scale from -10 (worst) to 0 (best), whereas the performances on the *Costs* criterion $c2$ are measured on a cardinal negative scale from -100.00 (*worst*) to 0.0 (*best*).
- The performances on the *Benefits* criteria $b2$ and $b3$ are measured on an ordinal positive scale from 0 (worst) to 10 (best), whereas the performances on the *Benefits* criteria $b1$, $b4$ and $b5$ are measured on a cardinal scale from 0.0 (*worst*) to 100.0 (*best*).
- The *importance* (sum of weights) of the 2 *Costs* criteria is equal to the *importance* (sum of weights) of the 5 *Benefits* criteria.

Absolute versus relative rating

Example (How to rate two decision actions – II)

- The performances on the *Costs* criterion $c2$ are measured on an ordinal negative scale from -10 (worst) to 0 (best), whereas the performances on the *Costs* criterion $c2$ are measured on a cardinal negative scale from -100.00 (*worst*) to 0.0 (*best*).
- The performances on the *Benefits* criteria $b2$ and $b3$ are measured on an ordinal positive scale from 0 (worst) to 10 (best), whereas the performances on the *Benefits* criteria $b1$, $b4$ and $b5$ are measured on a cardinal scale from 0.0 (*worst*) to 100.0 (*best*).
- The *importance* (sum of weights) of the 2 *Costs* criteria is equal to the *importance* (sum of weights) of the 5 *Benefits* criteria.

Absolute versus relative rating

Example (How to rate two decision actions – II)

- The performances on the *Costs* criterion $c2$ are measured on an ordinal negative scale from -10 (worst) to 0 (best), whereas the performances on the *Costs* criterion $c2$ are measured on a cardinal negative scale from -100.00 (*worst*) to 0.0 (*best*).
- The performances on the *Benefits* criteria $b2$ and $b3$ are measured on an ordinal positive scale from 0 (worst) to 10 (best), whereas the performances on the *Benefits* criteria $b1$, $b4$ and $b5$ are measured on a cardinal scale from 0.0 (*worst*) to 100.0 (*best*).
- The *importance* (sum of weights) of the 2 *Costs* criteria is equal to the *importance* (sum of weights) of the 5 *Benefits* criteria.

Absolute versus relative rating

Example (How to rate two decision actions – II)

- The performances on the *Costs* criterion $c2$ are measured on an ordinal negative scale from -10 (worst) to 0 (best), whereas the performances on the *Costs* criterion $c2$ are measured on a cardinal negative scale from -100.00 (*worst*) to 0.0 (*best*).
- The performances on the *Benefits* criteria $b2$ and $b3$ are measured on an ordinal positive scale from 0 (worst) to 10 (best), whereas the performances on the *Benefits* criteria $b1$, $b4$ and $b5$ are measured on a cardinal scale from 0.0 (*worst*) to 100.0 (*best*).
- The *importance* (sum of weights) of the 2 *Costs* criteria is equal to the *importance* (sum of weights) of the 5 *Benefits* criteria.

Absolute versus relative rating

Example (How to rate two decision actions – III)

When compared with all similar multi-criteria performances one has meanwhile already encountered, how may the multiple criteria performances of *a*1001, respectively *a*1010, now be *rated* ?

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
*a*1001	37.0	2	2	61.0	31.0	-4	-40.0
*a*1010	32.0	9	6	55.0	51.0	-4	-35.0

Excellent, good, or fair ?

Perhaps even, weak or very weak ?

Absolute versus relative rating

Example (How to rate two decision actions – III)

When compared with all similar multi-criteria performances one has meanwhile already encountered, how may the multiple criteria performances of *a*1001, respectively *a*1010, now be *rated* ?

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
*a*1001	37.0	2	2	61.0	31.0	-4	-40.0
*a*1010	32.0	9	6	55.0	51.0	-4	-35.0

Excellent, good, or fair ?

Perhaps even, weak or very weak ?

Absolute versus relative rating

Example (How to rate two decision actions – III)

When compared with all similar multi-criteria performances one has meanwhile already encountered, how may the multiple criteria performances of *a*1001, respectively *a*1010, now be *rated* ?

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
*a*1001	37.0	2	2	61.0	31.0	-4	-40.0
*a*1010	32.0	9	6	55.0	51.0	-4	-35.0

Excellent, good, or fair ?

Perhaps even, weak or very weak ?

Absolute versus relative rating

Example (How to rate two decision actions – III)

When compared with all similar multi-criteria performances one has meanwhile already encountered, how may the multiple criteria performances of *a*1001, respectively *a*1010, now be *rated* ?

Criterion	b1	b2	b3	b4	b5	c1	c2
Weight	2	2	2	2	2	5	5
*a*1001	37.0	2	2	61.0	31.0	-4	-40.0
*a*1010	32.0	9	6	55.0	51.0	-4	-35.0

Excellent, good, or fair ?
Perhaps even, weak or very weak ?

Incremental learning of quantiles

The `PerformanceQuantiles` class (see `performanceQuantiles` module) estimates performance quantiles from a performance tableau instance.

Its main components are:

- Ordered `objectives` and `criteria` dictionaries copied from the given performance tableau instance;
- A list called `quantileFrequencies`, with a complete set of quantile frequencies, like *quartiles* $[0.0, 0.25, 05, 0.75, 1.0]$, *quintiles* $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$ or *deciles* $[0.0, 0.1, 0.2, ..., 1.0]$ for instance;
- An ordered `limitingQuantiles` dictionary with so far estimated *lower* (default) or *upper* quantile class limits for each frequency per criterion;
- An ordered `historySizes` dictionary keeping track of the number of evaluations seen so far per criterion. Missing data may make these sizes vary from criterion to criterion.

Incremental learning of quantiles

The `PerformanceQuantiles` class (see `performanceQuantiles` module) estimates performance quantiles from a performance tableau instance.
Its main components are:

- Ordered `objectives` and `criteria` dictionaries copied from the given performance tableau instance;
- A list called `quantileFrequencies`, with a complete set of quantile frequencies, like *quartiles* $[0.0, 0.25, 05, 0.75, 1.0]$, *quintiles* $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$ or *deciles* $[0.0, 0.1, 0.2, ..., 1.0]$ for instance;
- An ordered `limitingQuantiles` dictionary with so far estimated *lower* (default) or *upper* quantile class limits for each frequency per criterion;
- An ordered `historySizes` dictionary keeping track of the number of evaluations seen so far per criterion. Missing data may make these sizes vary from criterion to criterion.

Incremental learning of quantiles

The PerformanceQuantiles class (see performanceQuantiles module) estimates performance quantiles from a performance tableau instance.
Its main components are:

- Ordered objectives and criteria dictionaries copied from the given performance tableau instance;
- A list called quantileFrequencies, with a complete set of quantile frequencies, like *quartiles* $[0.0, 0.25, 05, 0.75, 1.0]$, *quintiles* $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$ or *deciles* $[0.0, 0.1, 0.2, ..., 1.0]$ for instance;
- An ordered limitingQuantiles dictionary with so far estimated *lower* (default) or *upper* quantile class limits for each frequency per criterion;
- An ordered historySizes dictionary keeping track of the number of evaluations seen so far per criterion. Missing data may make these sizes vary from criterion to criterion.

Incremental learning of quantiles

The `PerformanceQuantiles` class (see `performanceQuantiles` module) estimates performance quantiles from a performance tableau instance.
Its main components are:

- Ordered `objectives` and `criteria` dictionaries copied from the given performance tableau instance;
- A list called `quantileFrequencies`, with a complete set of quantile frequencies, like *quartiles* $[0.0, 0.25, 05, 0.75, 1.0]$, *quintiles* $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$ or *deciles* $[0.0, 0.1, 0.2, ..., 1.0]$ for instance;
- An ordered `limitingQuantiles` dictionary with so far estimated *lower* (default) or *upper* quantile class limits for each frequency per criterion;
- An ordered `historySizes` dictionary keeping track of the number of evaluations seen so far per criterion. Missing data may make these sizes vary from criterion to criterion.

Incremental learning of quantiles

The `PerformanceQuantiles` class (see `performanceQuantiles` module) estimates performance quantiles from a performance tableau instance.
Its main components are:

- Ordered `objectives` and `criteria` dictionaries copied from the given performance tableau instance;
- A list called `quantileFrequencies`, with a complete set of quantile frequencies, like *quartiles* $[0.0, 0.25, 05, 0.75, 1.0]$, *quintiles* $[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]$ or *deciles* $[0.0, 0.1, 0.2, ..., 1.0]$ for instance;
- An ordered `limitingQuantiles` dictionary with so far estimated *lower* (default) or *upper* quantile class limits for each frequency per criterion;
- An ordered `historySizes` dictionary keeping track of the number of evaluations seen so far per criterion. Missing data may make these sizes vary from criterion to criterion.

Using the PerformanceQuantiles class

Example Python session:

```
>>> from performanceQuantiles import PerformanceQuantiles
>>> from randomPerfTabs import RandomCBPerformanceTableau
>>> tp = RandomCBPerformanceTableau(numberOfActions=900,\
                     numberOfCriteria=7,seed=100)
>>> pq = PerformanceQuantiles(tp, numberOfBins = 'quartiles',\
                   LowerClosed=True)
>>> pq
*------- PerformanceQuantiles instance description ------*
Instance class    : PerformanceQuantiles
Instance name     : 4-tiled_performances
# Objectives      : 2, # Criteria      : 7, # Quantiles      : 4
# History sizes   : {'c1': 887, 'b1': 888, 'b2': 891, 'b3': 895,
                    'b4': 892, 'c2': 893, 'b5': 887}
Attributes        : ['perfTabType', 'valueDigits',
                    'actionsTypeStatistics', 'objectives',
                    'BigData', 'missingDataProbability',
                    'criteria', 'LowerClosed', 'name',
                    'quantilesFrequencies', 'historySizes',
                    'limitingQuantiles', 'cdf']
```

We suppose that the decision alternatives *a*1001 and *a*1010, seen before, are indeed drawn from the same *tp* random performance tableau model.

Using the PerformanceQuantiles class – continue

The constructor parameter numberOfBins, choosing the wished number of quantile frequencies, may be either *quartiles* (4 bins), *quintiles* (5 bins), *deciles* (10 bins), *dodeciles* (20 bins) or any other integer number of quantile bins. The quantile bins may be either lower closed (default) or upper-closed.

```
>>> # showing quantile limits
>>> pq.showLimitingQuantiles(ByObjectives=True)
*----  performance quantiles -----*
Costs
criteria | weights | '0.00'   '0.25'   '0.50'   '0.75'   '1.00'
---------|-----------------------------------------------------
   'c1'  |    5    |  -10       -7       -5       -3        0
   'c2'  |    5    | -96.37   -70.65   -50.10   -30.00   -1.43
Benefits
criteria | weights | '0.00'   '0.25'   '0.50'   '0.75'   '1.00'
---------|-----------------------------------------------------
   'b1'  |    2    |  1.99    29.82    49.44    70.73    99.83
   'b2'  |    2    |   0        3        5        7       10
   'b3'  |    2    |   0        3        5        7       10
   'b4'  |    2    |  3.27    30.10    50.82    70.89    98.05
   'b5'  |    2    |  0.85    29.08    48.55    69.98    97.56
```

Using the `PerformanceQuantiles` class – continue

- The preference direction of the *Costs* criteria is *negative*; the lesser the costs the better it is, wheras all the *Benefits* criteria: $b1$ to $b5$ show *positive* preference directions, i.e. the higher the benefits the better it is.
- The columns entitled '0.00', resp. '1.00' show the quartile Q_0, resp. Q_4, ie the *worst*, resp. *best* performance observed so far on each criterion. Column '0.50' shows Q_2, the median performance observed on the criteria.
- The random performances on all criteria appear to be more or less symmetrically distributed around median scales values $(-50.0, 5, 50.0)$ with a spread of approximatively 20% of the scale's amplitude.

Generating new random data

New decision actions with random multiple criteria performance vectors from the same random performance tableau model may now be generated with ad hoc random performance generators. We provide, for experimental purpose, in the `randomPerfTabs` module a generic `RandomPerformanceGenerator` for three models of random performance tableaux:

- The *standard* `RandomPerformanceTableau` model,
- The two objectives `RandomCBPerformanceTableau` *Cost-Benefit* model, and
- The `Random3ObjectivesPerformanceTableau` model with three objectives concerning respectively *economic*, *environmental* and *societal aspects*.

```
>>> # generate 100 new random decision actions
>>> from randomPerfTabs import RandomPerformanceGenerator
>>> rpg = RandomPerformanceGenerator(tp,seed=seed)
>>> newTab = rpg.randomPerformanceTableau(100)
```

Updating the historical quantile limits

Given a new performance tableau *newTab* with 100 new decision alternatives, the so far estimated historical quantile limits may be updated as follows:

```
>>> # Updating the quartile norms shown above
>>> pq.updateQuantiles(newTab,historySize=None)
```

- Parameter `historySize` of the pq.updateQuantiles() method allows to balance the new evaluations against the historical ones.
- With `historySize = None` (the default setting), the balance in the example above is 900/1000 (90%, weight of historical data) against 100/1000 (10%, weight of the new incoming observations).
- Putting `historySize = 0`, for instance, will ignore all historical data (0/100 against 100/100) and restart building the quantile estimation with solely the new incoming data.

Showing the historical quantile limits

The updated quantile limits may be shown in a browser view.

```
>>> # browsing the updated quantile limits
>>> pq.showHTMLLimitingQuantiles(Transposed=True)
```

Performance quantiles

Sampling sizes between 986 and 995.

criterion	0.00	0.25	0.50	0.75	1.00
b1	1.99	28.77	49.63	75.27	99.83
b2	0.00	2.94	4.92	6.72	10.00
b3	0.00	2.90	4.86	8.01	10.00
b4	3.27	35.91	58.58	72.00	98.05
b5	0.85	32.84	48.09	69.75	99.00
c1	-10.00	-7.35	-5.39	-3.38	0.00
c2	-96.37	-72.22	-52.27	-33.99	-1.43

The NormedQuantilesRatingDigraph class

- For absolute rating of a newly given set of decision actions, we provide, in the sortingDigraphs module, the NormedQuantilesRatingDigraph class, a specialisation of the SortingDigraph class.
- The class constructor requires a valid PerformanceQuantiles instance (*pq*) and a compatible PerformanceTableau instance (*newTab*) or a dictionary *newActions* with compatible new decision alternatives.
- The actions dictionary in such a NormedQuantilesRatingDigraph class instance will contain not only newly given decision alternatives, but also the estimated quantile bins' performance limits from a given PerformanceQuantiles instance.

The NormedQuantilesRatingDigraph class

- For absolute rating of a newly given set of decision actions, we provide, in the sortingDigraphs module, the NormedQuantilesRatingDigraph class, a specialisation of the SortingDigraph class.
- The class constructor requires a valid PerformanceQuantiles instance (*pq*) and a compatible PerformanceTableau instance (*newTab*) or a dictionary *newActions* with compatible new decision alternatives.
- The actions dictionary in such a NormedQuantilesRatingDigraph class instance will contain not only newly given decision alternatives, but also the estimated quantile bins' performance limits from a given PerformanceQuantiles instance.

The `NormedQuantilesRatingDigraph` class

- For absolute rating of a newly given set of decision actions, we provide, in the `sortingDigraphs` module, the `NormedQuantilesRatingDigraph` class, a specialisation of the `SortingDigraph` class.
- The class constructor requires a valid `PerformanceQuantiles` instance (*pq*) and a compatible `PerformanceTableau` instance (*newTab*) or a dictionary *newActions* with compatible new decision alternatives.
- The actions dictionary in such a `NormedQuantilesRatingDigraph` class instance will contain not only newly given decision alternatives, but also the estimated quantile bins' performance limits from a given `PerformanceQuantiles` instance.

Using the NormedQuantilesRatingDigraph class

```
>>> from sortingDigraphs import NormedQuantilesRatingDigraph
>>> newActions = rpg.randomActions(10)
>>> nqr = NormedQuantilesRatingDigraph(pq, newActions,\
                                       rankingRule = 'best')
>>> nqr
*-----  Object instance description -----------*
Instance class      : NormedQuantilesRatingDigraph
Instance name       : normedRatingDigraph
# Criteria          : 7,
# Quantile profiles : 4
# New actions       : 10
Size                : 93
Determinateness     : 50.962
Attributes: ['runTimes','objectives','criteria','LowerClosed',
    'quantilesFrequencies','limitingQuantiles','historySizes',
    'cdf','name','newActions','evaluation','categories',
    'criteriaCategoryLimits','profiles','profileLimits','hasNoVeto',
    'actions','completeRelation','relation','concordanceRelation',
    'valuationdomain','order','gamma','notGamma','rankingRule',
    'rankingCorrelation','rankingScores','actionsRanking',
    'ratingCategories','ratingRelation','relationOrig',
    'rankingByBestChoosing']
```

Using the NormedQuantilesRatingDigraph class

Data input to the *NormedQuantilesRatingDigraph* class constructor provides a set, called *newActions*, of new decision alternatives generated from the same random model.

```
>>> nqr.showHTMLPerformanceTableau(actionsSubset = nqr.newActions)
```

criteria	b1	b2	b3	b4	b5	c1	c2	
weight	**2.00**	**2.00**	**2.00**	**2.00**	**2.00**	**5.00**	**5.00**	
a1001c	37.00	2.00	2.00	61.00	31.00	-4.00	-40.00	←
a1002c	27.00	5.00	4.00	54.00	63.00	-6.00	-23.00	
a1003a	24.00	8.00	2.00	74.00	61.00	-8.00	-37.00	
a1004c	16.00	3.00	1.00	25.00	48.00	-5.00	-37.00	
a1005c	42.00	3.00	6.00	28.00	30.00	-1.00	-24.00	
a1006c	33.00	3.00	3.00	20.00	39.00	-5.00	-27.00	
a1007c	39.00	6.00	2.00	20.00	16.00	-1.00	-73.00	
a1008n	64.00	5.00	6.00	49.00	96.00	-6.00	-43.00	
a1009n	42.00	4.00	6.00	44.00	57.00	-6.00	-94.00	
a1010n	32.00	9.00	6.00	55.00	51.00	-4.00	-35.00	←

Among the 10 new incoming decision actions (see above) there appear the two decision actions *a*1001 and *a*1010 mentioned in the beginning.

Using the NormedQuantilesRatingDigraph class

The NormedQuantilesRatingDigraphdigraph instance's actions dictionary also contains the closed lower limits of the four quartile classes: $m1 = [0.0 - 0.25[$, $m2 = [0.25 - 0.5[$, $m3 = [0.5 - 0.75[$, and $m4 = [0.75 - 1.0[$.

```
>>> nqr.showPerformanceTableau(actionsSubset=nqr.profiles)
*----  performance tableau -----*
criteria |   'm1'    'm2'    'm3'    'm4'
---------|-------------------------------
  'b1'   |    2.0    28.8    49.6    75.3
  'b2'   |    0.0     2.9     4.9     6.7
  'b3'   |    0.0     2.9     4.9     8.0
  'b4'   |    3.3    35.9    58.6    72.0
  'b5'   |    0.8    32.8    48.1    69.7
  'c1'   |  -10.0    -7.4    -5.4    -3.4
  'c2'   |  -96.4   -72.2   -52.3   -34.0
```

Ranking decision actions and quantile limits

- The actual rating procedure will rely on a linear ranking of the *new decision actions* and the *quantile class limits* obtained from the corresponding bipolar valued outranking digraph.
- Two *efficient and scalable* ranking rules, the Copeland rule and, its valued version, the Netflows rule may be used for this purpose.
- The `rankingRule` Parameter allows to choose one of both. With `rankingRule = 'best'` (see Line 4 above) the `NormedQuantilesRatingDigraph` constructor will choose the ranking rule that results in the highest ordinal correlation with the given outranking relation.
- In this rating example, the Copeland rule appears to be the *more appropriate* ranking rule.

Ranking decision actions and quantile limits

- The actual rating procedure will rely on a linear ranking of the *new decision actions* and the *quantile class limits* obtained from the corresponding bipolar valued outranking digraph.
- Two *efficient and scalable* ranking rules, the Copeland rule and, its valued version, the Netflows rule may be used for this purpose.
- The `rankingRule` Parameter allows to choose one of both. With `rankingRule = 'best'` (see Line 4 above) the `NormedQuantilesRatingDigraph` constructor will choose the ranking rule that results in the highest ordinal correlation with the given outranking relation.
- In this rating example, the Copeland rule appears to be the *more appropriate* ranking rule.

Ranking decision actions and quantile limits

- The actual rating procedure will rely on a linear ranking of the *new decision actions* and the *quantile class limits* obtained from the corresponding bipolar valued outranking digraph.
- Two *efficient and scalable* ranking rules, the Copeland rule and, its valued version, the Netflows rule may be used for this purpose.
- The `rankingRule` Parameter allows to choose one of both. With `rankingRule = 'best'` (see Line 4 above) the `NormedQuantilesRatingDigraph` constructor will choose the ranking rule that results in the highest ordinal correlation with the given outranking relation.
- In this rating example, the Copeland rule appears to be the *more appropriate* ranking rule.

Ranking decision actions and quantile limits

- The actual rating procedure will rely on a linear ranking of the *new decision actions* and the *quantile class limits* obtained from the corresponding bipolar valued outranking digraph.
- Two *efficient and scalable* ranking rules, the Copeland rule and, its valued version, the Netflows rule may be used for this purpose.
- The `rankingRule` Parameter allows to choose one of both. With `rankingRule = 'best'` (see Line 4 above) the `NormedQuantilesRatingDigraph` constructor will choose the ranking rule that results in the highest ordinal correlation with the given outranking relation.
- In this rating example, the Copeland rule appears to be the *more appropriate* ranking rule.

Ranking decision actions and quantile limits

```
>>> nqr.rankingRule
 'Copeland'
>>> nqr.actionsRanking
 ['m4', 'a1005', 'a1010', 'a1002', 'a1008', 'a1006',
  'a1001', 'a1003', 'm3', 'a1004', 'a1007', 'a1009',
  'm2', 'm1']
>>> nqr.showCorrelation(nqr.correlation)
 Correlation indexes:
  Crisp ordinal correlation  : +0.945
  Epistemic determination    :  0.522
  Bipolar-valued equivalence : +0.493
```

We achieve here a linear ranking without ties (from best to worst) of the new decision actions as well as the quartile limits $m1$ to $m4$, which is very close in an ordinal sense ($\tau = +0.94$) to the underlying outranking relation.

Showing rating results

The eventual rating procedure is based on the lower quantile limits, such that we may collect the quartiles' contents in increasing order of the quartiles' lower limits.

```
>>> nqr.ratingCategories
OrderedDict([
('m2', ['a1004', 'a1007', 'a1009']),
('m3', ['a1001', 'a1002', 'a1003', 'a1005', 'a1006', 'a1008', 'a1010'])
])
```

We notice above that no decision action is rated in the highest quartile class [0.75 - 1.0] or in the lowest quartile class [0.0 - 0.25[. Indeed, the rating result is shown in descending order as follows.

```
>>> nqr.showQuantilesRating()
*-------- Quantiles rating result ---------
[0.50 - 0.75[ ['a1001', 'a1002', 'a1003', 'a1005',
               'a1006', 'a1008', 'a1010']
[0.25 - 0.50[ ['a1004', 'a1007', 'a1009']
```

Reconsidering the question at the beginning of the lecture, we may now see that in view of our historical recordings, both decision action $a1001$ and $a1010$ are actually rated in quartile $Q3$ ([0.50—0.75[).

Showing rating results – continue

The same result may even more conviently be consulted in a browser view via a specialised heatmap illustration.

```
>>> nqr.showHTMLRatingHeatmap(\
          Correlations=True,\
          colorLevels=5)
```

Heatmap of quartiles rating

Ranking rule: **Copeland**; *Ranking correlation*: **0.945**

criteria	c2	b3	c1	b2	b1	b5	b4
weights	5	2	5	2	2	2	2
tau(*)	0.64	0.53	0.41	0.38	0.36	0.35	0.35
[0.75 -	-33.99	8.01	-3.38	6.72	75.27	69.75	72.00
a1005c	-24.00	6.00	-1.00	3.00	42.00	30.00	28.00
a1010n	-35.00	6.00	-4.00	9.00	32.00	51.00	55.00
a1002c	-23.00	4.00	-6.00	5.00	27.00	63.00	54.00
a1008n	-43.00	6.00	-6.00	5.00	64.00	96.00	49.00
a1006c	-27.00	3.00	-5.00	3.00	33.00	39.00	20.00
a1001c	-40.00	2.00	-4.00	2.00	37.00	31.00	61.00
a1003a	-37.00	2.00	-8.00	8.00	24.00	61.00	74.00
[0.50 -	-52.27	4.86	-5.39	4.92	49.63	48.09	58.58
a1004c	-37.00	1.00	-5.00	3.00	16.00	48.00	25.00
a1007c	-73.00	2.00	-1.00	6.00	39.00	16.00	20.00
a1009n	-94.00	6.00	-6.00	4.00	42.00	57.00	44.00
[0.25 -	-72.22	2.90	-7.35	2.94	28.77	32.84	35.91
[0.00 -	-96.37	0.00	-10.00	0.00	1.99	0.85	3.27

Color legend:

quantile	20.00%	40.00%	60.00%	80.00%	100.00%

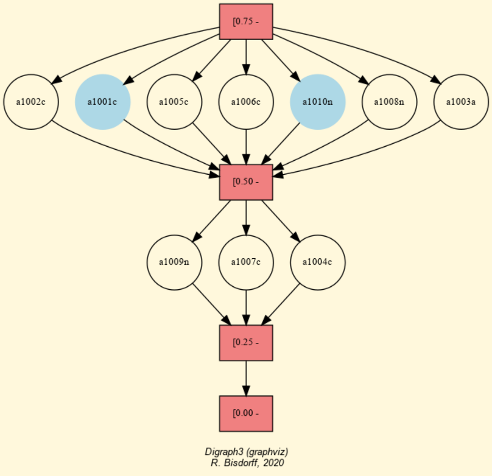
(*) tau: Ordinal (Kendall) correlation between marginal criterion and global ranking relation.

Showing the quartiles rating graph

```
>>> nqr.exportGraphViz()
```

Reconsidering the question at the beginning of the lecture, we may now see that in view of our historical recordings:

decision action 'a1001' and decision action 'a1010' are rated in quartile Q_3 ($[0.50 - 0.75[$).



Conclusion

In this presentation, we addressed the problem of rating multiple criteria performances of a set of potential decision actions with respect to empirical order statistics, i.e. performance quantiles learned from historical performance data gathered from similar decision actions observed in the past.

Example (How to rate two decision actions – continue)

Absolute Rating	Criterion Weight	b1 2	b2 2	b3 2	b4 2	b5 2	c1 5	c2 5
[0.50 – 0.75[	*a1001c*	37.0	2	2	51.00	31.00	-4	-40.00
[0.50 – 0.75[	*a10010n*	32.0	9	6	55.00	51.00	-4	-35.00

A refined deciles rating result

Heatmap of quartiles rating

Ranking rule: **Copeland**; *Ranking correlation*: **0.945**

criteria	c2	b3	c1	b2	b1	b5	b4
weights	5	2	5	2	2	2	2
tau(*)	0.64	0.53	0.41	0.38	0.36	0.35	0.35
[0.75 -	-33.99	8.01	-3.38	6.72	75.27	69.75	72.00
a1005c	-24.00	6.00	-1.00	3.00	42.00	30.00	28.00
a1010n	-35.00	6.00	-4.00	9.00	32.00	51.00	55.00
a1002c	-23.00	4.00	-6.00	5.00	27.00	63.00	54.00
a1008n	-43.00	6.00	-6.00	5.00	64.00	96.00	49.00
a1006c	-27.00	3.00	-5.00	3.00	33.00	39.00	20.00
a1001c	-40.00	2.00	-4.00	2.00	37.00	31.00	61.00
a1003a	-37.00	2.00	-8.00	8.00	24.00	61.00	74.00
[0.50 -	-52.27	4.86	-5.39	4.92	49.63	48.09	58.58
a1004c	-37.00	1.00	-5.00	3.00	16.00	48.00	25.00
a1007c	-73.00	2.00	-1.00	6.00	39.00	16.00	20.00
a1009n	-94.00	6.00	-6.00	4.00	42.00	57.00	44.00
[0.25 -	-72.22	2.90	-7.35	2.94	28.77	32.84	35.91
[0.00 -	-96.37	0.00	-10.00	0.00	1.99	0.85	3.27

Color legend:

quantile	20.00%	40.00%	60.00%	80.00%	100.00%

(*) tau: Ordinal (Kendall) correlation between marginal criterion and global ranking relation.

Heatmap of Deciles rating

Ranking rule: **NetFlows**; *Ranking correlation*: **0.960**

criteria	c2	b3	c1	b1	b5	b2	b4
weights	5	2	5	2	2	2	2
tau(*)	0.67	0.65	0.58	0.57	0.53	0.53	0.48
[0.90 -	-20.32	7.73	-2.53	86.83	82.16	7.66	82.04
[0.80 -	-29.70	7.26	-3.35	79.30	75.15	6.64	74.66
[0.70 -	-37.97	6.67	-4.14	70.95	60.20	5.88	69.76
a1005c	-24.00	6.00	-1.00	42.00	30.00	3.00	28.00
a1010n	-35.00	6.00	-4.00	32.00	51.00	9.00	55.00
a1008n	-43.00	6.00	-6.00	64.00	96.00	5.00	49.00
a1002c	-23.00	4.00	-6.00	27.00	63.00	5.00	54.00
[0.60 -	-44.23	5.92	-5.04	60.56	56.01	5.37	62.23
a1006c	-27.00	3.00	-5.00	33.00	39.00	3.00	20.00
a1001c	-40.00	2.00	-4.00	37.00	31.00	2.00	61.00
a1003a	-37.00	2.00	-8.00	24.00	61.00	8.00	74.00
[0.50 -	-52.22	4.64	-6.02	49.56	48.07	4.83	58.45
a1007c	-73.00	2.00	-1.00	39.00	16.00	6.00	20.00
a1004c	-37.00	1.00	-5.00	16.00	48.00	3.00	25.00
[0.40 -	-60.50	3.84	-6.69	39.61	40.16	4.25	49.82
a1009n	-94.00	6.00	-6.00	42.00	57.00	4.00	44.00
[0.30 -	-67.14	3.12	-7.32	30.85	34.33	3.30	40.89
[0.20 -	-77.07	2.55	-7.94	23.84	29.57	2.27	30.45
[0.10 -	-83.04	1.99	-8.48	16.64	16.91	1.58	24.78
[0.00 -	-96.37	0.00	-10.00	1.99	0.85	0.00	3.27

Color legend:

quantile	20.00%	40.00%	60.00%	80.00%	100.00%

(*) tau: Ordinal (Kendall) correlation between marginal criterion and global ranking relation.

Further reading about *q*-tiling and quantiles rating may be found in the Digraph3 tutorials :

https://digraph3.readthedocs.io/en/latest/