“By Example” introduction into the Mathematica PairwiseComparisons` Package

Konrad Kułakowski (C) 2014 - 2025
e-mail: konrad@kulakowski.org

To install the package on the computer start Mathematica and choose File/Install..., then select PairwiseComparisons.m as the package source and press OK.

To install the package at the Raspberry PI just copy them into /opt/Wolfram/WolframEngine/10.0/AddOns/ExtraPackages and restart the application

Introduction

In[1]:=
```
<< PairwiseComparisons`;
```

Alternatively you can call:

In[2]:=
```
Needs["PairwiseComparisons`"];
```

If you are going to use the package for parallel programming use ParallelNeeds

In[3]:=
```
ParallelNeeds["PairwiseComparisons`"];
```

Printing function usage

In[4]:=
```
? GeometricRank
```

Out[4]=

Symbol

GeometricRank[M] returns rank list given as geometric means of rows of the matrix M

Printing full (with implementation) function usage

In[5]:=
```
?? GeometricRank
```

Out[5]=

Symbol

GeometricRank[M] returns rank list given as geometric means of rows of the matrix M

Definitions

```
GeometricRank[PairwiseComparisons`Private`matrix_] :=
 (GeometricMean[#1] &) /@ PairwiseComparisons`Private`matrix
```

Full Name PairwiseComparisons`GeometricRank


```
GeometricRank[PairwiseComparisons`Private`matrix_] :=
 (GeometricMean[#1] &) /@ PairwiseComparisons`Private`matrix
```


Print all the public functions in the package

In[218]:=
```
Names["PairwiseComparisons`*"]
```

Out[218]=
```
{AHPA, AHPC, AIJadd, AIJadditiveWithPriorities, AIJgeom, AIJgeomWithPriorities,
 CardinalToOrdinalPCMatrix, CircularTriadsList, CircularTriadsNo,
 CIStar, CompatibilityIndex, ConsistentMatrixFromRank, COP1Check,
 COP1ViolationList, COP2Check, COP2ViolationList, CR, EigenvalueRank,
 EigenvalueRankSym, ErrorMatrix, EVM, FindAllSpanningTrees,
 FundamentalScale, GCI, GED, GEDMin, GeneralizedGeometricConsistencyIndex1,
 GeneralizedGeometricConsistencyIndex2, GeneralizedGoldenWangIndex,
 GeneralizedKendallIndex, GeneralizedKoczkodajIndex,
 GeneralizedRelativeError, GeneralizedRelativeErrorv2,
 GeneralizedSaloHamalainenIndex, GeneralizedTriadBasedInconsistencyIndexAlpha,
 GeneralizedTriadBasedInconsistencyIndexAlphaBeta,
 GeneralizedTupleBasedInconsistencyIndex1,
 GeneralizedTupleBasedInconsistencyIndex2, GeometricConsistencyIndex,
 GeometricRank, GeometricRescaledRank, GetRandomSpanningTreeFromPCMatrix,
 GGCI1, GGCI2, GGWI, GKI, GlobalDiscrepancy, GMM, GoldenWangIndex,
 GraphOfGeneralizedPCM, GraphOfPCM, GRE, GREv2, GSHI, GTBII1, GTBII2,
 GTBIIAlpha, GTBIIAlphaBeta, GWI, HarkerMatrix, HarmonicConsistencyIndex, HCI,
 HPCA, HPCC, HREConstantTermVector, HREFullRank, HREGeomConstantTermVector,
 HREGeomFullRank, HREGeomIntermediateRank, HREGeomMatrix,
 HREGeomPartialRank, HREGeomRescaledRank, HREIncompleteConstantTermsVector,
 HREIncompleteConstantTermsVectorElement, HREIncompleteFullRanking,
 HREIncompleteMatrix, HREIncompletePartialRanking, HREIncompleteRescaledRank,
 HREMatrix, HREPartialRank, HRERescaledRank, IndexOfDeterminants,
 IrreducibleMatrixQ, KendallIndex, KendallTauDistance,
 KoczkodajConsistentTriad, KoczkodajIdx, KoczkodajIdxRecip,
 KoczkodajImproveMatrixStep, KoczkodajTheMostInconsistentTriad,
 KoczkodajTheWorstTriad, KoczkodajTheWorstTriads, KoczkodajTriadIdx,
 KoczkodajTriadInconsistency, LCMatrix, ListOfPossibleRationalEntries,
 ListProduct, LLSM, LocalDiscrepancyMatrix, LowerTriangularizePCM,
 MaxInconsistentTriadsNoTies, MaxInconsistentTriadsWithTies, NGMM,
 NicePlotFraction, NormalizedKendallTauDistance, NumericalRankToOrdinalRank,
 NumericalRankToReverseOrdinalRank, PlotPCGraph, PlotUndirectedWeightedGraph,
 PrincipalEigenValue, PrincipalEigenValueSym, PrincipalEigenVector,
 RandomBoolean, RandomIndex, RandomlyDecompletedMatrix, RandomMatrix,
 RandomMatrixEntry, RandomRankingPattern, RandomRationalMatrix,
 RandomRationalMatrixEntry, RandomRationalUniformRankingPattern,
 RankBySpanningTree, RankList, RankListWithPos, RankOrder,
 RankOrderToOrdinalRank, RankOrderToReverseOrdinalRank, RankOrderWithTies,
 RankToVector, RationalRandomScaledMatrix, RationalScale, RC, RE,
 RecreateOrdinalPCMatrix, RecreatePCMatrix, RelativeError, RelativeErrorv2,
 RescaleRanking, REv2, RI, SaatyConsistencyRatio, SaatyIdx, SaatyIdxSym,
 SaloHamalainenIndex, SHI, SymbPCM, SymbPCMRec, TBII1, TBII2, TBIIAlpha,
 TriadBasedInconsistencyIndex, TriadBasedInconsistencyIndex2,
 TriadBasedInconsistencyIndexAlpha, UpperTriangularizePCM, VersionPC}
```

Assign the matrix to the variable M

In[8]:=
$$M = \begin{pmatrix} 1 & 2 & 3 \\ \frac{1}{2} & 1 & 4 \\ \frac{1}{3} & \frac{1}{4} & 1 \end{pmatrix}$$

Out[8]=
$$\left\{\{1, 2, 3\}, \left\{\frac{1}{2}, 1, 4\right\}, \left\{\frac{1}{3}, \frac{1}{4}, 1\right\}\right\}$$

Eigenvalue based method

To calculate the principal eigenvalue of the matrix M it is enough to call:

In[9]:=
```
PrincipalEigenValue[M]
```

Out[9]=
```
3.10785
```

Calculate the principal eigenvector of the matrix M

In[10]:=
```
PrincipalEigenVector[M]
```

Out[10]=
```
{0.806208, 0.558993, 0.193792}
```

Calculate the value of the Saaty (eigenvalue based) inconsistency index of M

In[11]:=
```
SaatyIdx[M]
```

Out[11]=
```
0.0539237
```

Calculate the value of the Saaty consistency ratio of M

In[12]:=
```
SaatyConsistencyRatio[M]
```

Out[12]=
```
0.102278
```

Calculate Random index for the matrix of size 3 by 3 (note that this value is Monte Carlo approximation)

In[13]:=
```
RandomIndex[3]
```

Out[13]=
```
0.527229
```

Calculate Random index for the matrix of size 3 by 3 based on 500 random PC matrices with the rational scale 1/9,1/8,...1,2,3,...,9

In[14]:=
```
RandomIndex[3, 500, 9]
```

Out[14]=
```
0.569709
```

Calculate the eigenvalue based ranking based on M

In[15]:=
```
EigenvalueRank[M]
```
Out[15]=
```
{0.517134, 0.35856, 0.124306}
```

Analytic Hierarchy Process (AHP)

Calculate the rank for the three different criteria using AHP (see Belton and Gear, On a short-coming of Saaty's method of analytic hierarchies, 1983) . To this end we will use two methods AHPA - which implements AHP on the Alternative level, and AHPC which computes the ranking on the criteria level.

In[16]:=
```
A1 = {{1, 1, 4}, {1, 1, 4}, {1/4, 1/4, 1}};
```

In[17]:=
```
EigenvalueRank[A1]
```
Out[17]=
```
{0.444444, 0.444444, 0.111111}
```

In[18]:=
```
A2 = {{1, 2, 6}, {1/2, 1, 3}, {1/6, 1/3, 1}};
```

In[19]:=
```
EigenvalueRank[A2]
```
Out[19]=
```
{0.6, 0.3, 0.1}
```

In[20]:=
```
A3 = {{1, 1/2, 1/8}, {2, 1, 1/4}, {8, 4, 1}};
```

In[21]:=
```
EigenvalueRank[A3]
```
Out[21]=
```
{0.0909091, 0.181818, 0.727273}
```

In[22]:=
```
Acriteria = {{1, 1/2, 1/4}, {2, 1, 1/2}, {4, 2, 1}};
```

In[23]:=
```
EigenvalueRank[Acriteria]
```
Out[23]=
```
{0.142857, 0.285714, 0.571429}
```

then the result is:

In[24]:=
```
AHPC[Acriteria, AHPA[A1, A2, A3]]
```
Out[24]=
```
{0.286869, 0.253102, 0.460029}
```

The above example is taken from the book Matteo Brunelli, Introduction to the Analytic Hierarchy Process, Springer 2015, http://www.springer.com/gp/book/9783319125015

Well, of course we may create another level of a hierarchy

In[25]:=
$$A_{upperLevelCriteria} = \begin{pmatrix} 1 & 4 & 4 \\ 1/4 & 1 & 3 \\ 1/4 & 1/3 & 1 \end{pmatrix};$$

and compute the AHP ranking for that hierarchy

```
In[26]:= AHPC[A_upperLevelCriteria, AHPC[A_criteria, AHPA[A_1, A_2, A_3]],
          AHPC[A_criteria, AHPA[A_2, A_3, A_1]], AHPC[A_criteria, AHPA[A_3, A_2, A_1]]]
Out[26]=
         {0.322511, 0.28831, 0.389179}
```

Geometric Mean method (Logarithmic Least Square Method)

Calculate the geometric mean based ranking based on M

```
In[27]:= N[GeometricRank[M]]
Out[27]=
         {1.81712, 1.25992, 0.43679}
```

Calculate the geometric mean based ranking based on M rescaled so that the sum of entries is 1

```
In[28]:= N[GeometricRescaledRank[M]]
Out[28]=
         {0.517134, 0.35856, 0.124306}
```

Please note that if you want to have numeric (not symbolic) output the input data also should be “numeric”.
Thus, it is not a bad idea to add a numeric conversion N@ to the input matrix. For example:

```
In[29]:= N[GeometricRank[N@M]]
Out[29]=
         {1.81712, 1.25992, 0.43679}
```

The above applies to all the methods in the package, except eigenvalue based methods such as PrincipalEigenValue,
PrincipalEigenVector, SaatyIdx and EigenvalueRank, which are “by design” numeric.

The geometric consisteny index (GCI) as defined in “Aguaron & Moreno-Jimenez, The geometric consistency index: Approximated thresholds“ can be computed by calling:

```
In[30]:= N@GCI[M, GeometricRank[M], 10]
Out[30]=
         GCI[{{1., 2., 3.}, {0.5, 1., 4.}, {0.333333, 0.25, 1.}},
          {1.81712, 1.25992, 0.43679}, 10.]
```

Hierarchical Pairwise Comparisons Method (HPCM)

AHPA and AHPC can be generalized to use any ranking method not necessarily EVM. In order to use GMM method instead of EVM one we may use the functions HPCA (Hierarchical Pairwise Comparisons on the level of Alternatives) and HPCC (Hierarchical Pairwise Comparisons on the level of

Criteria) respectively.
For example for GMM

In[31]:=
```
N@HPCC[GeometricRescaledRank, A_criteria, HPCA[GeometricRescaledRank, A_1, A_2, A_3]]
```

Out[31]=
```
{0.286869, 0.253102, 0.460029}
```

In[32]:=
```
N@HPCC[GeometricRescaledRank, A_upperLevelCriteria,
  HPCC[GeometricRescaledRank, A_criteria, HPCA[GeometricRescaledRank, A_1, A_2, A_3]],
  HPCC[GeometricRescaledRank, A_criteria, HPCA[GeometricRescaledRank, A_2, A_3, A_1]],
  HPCC[GeometricRescaledRank, A_criteria, HPCA[GeometricRescaledRank, A_3, A_2, A_1]]]
```

Out[32]=
```
{0.322511, 0.28831, 0.389179}
```

We may also mix within the same hierarchy different priority deriving methods. For example:

In[33]:=
```
HPCC[GeometricRescaledRank, A_criteria, HPCA[EigenvalueRank, A_1, A_2, A_3]]
```

Out[33]=
```
{0.286869, 0.253102, 0.460029}
```

Heuristic Rating Estimation Method (additive)

Calculate the HRE Matrix for the given matrix M where the unknown concepts are $\{c_1, c_2, c_3\}$ and the known concepts are $\{c_4 = 5, c_5 = 9\}$. It is assumed that the unknown concepts has value 0 whilst the known concepts have the values greater than 0.

Further references could be found in papers :

* Konrad Kułakowski,

Heuristic Rating Estimation Approach to The Pairwise Comparisons Method
http : // arxiv.org / abs / 1309.0386

* Konrad Kułakowski,

A heuristic rating estimation algorithm for the pairwise comparisons method
http : // dx.doi.org / 10.1007 / s10100 – 013 – 0311 – x

SymbPCM[symbol, size] - creates symbolic PC matrix based on the given symbol with the given size

In[34]:=
```
M = SymbPCM[m, 5];
```

In[35]:=
```
M // MatrixForm
```

Out[35]//MatrixForm=

$$\begin{pmatrix} 1 & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} \\ m_{2,1} & 1 & m_{2,3} & m_{2,4} & m_{2,5} \\ m_{3,1} & m_{3,2} & 1 & m_{3,4} & m_{3,5} \\ m_{4,1} & m_{4,2} & m_{4,3} & 1 & m_{4,5} \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & 1 \end{pmatrix}$$

In[36]:=
```
HREMatrix[M, {{0}, {0}, {0}, {5}, {9}}]
```

Out[36]=
$$\left\{\left\{1, -\frac{m_{1,2}}{4}, -\frac{m_{1,3}}{4}\right\}, \left\{-\frac{m_{2,1}}{4}, 1, -\frac{m_{2,3}}{4}\right\}, \left\{-\frac{m_{3,1}}{4}, -\frac{m_{3,2}}{4}, 1\right\}\right\}$$

Calculate the HRE constant term vector for M
where C_U equals $\{c_1, c_2, c_3\}$ and C_K equals $\{c_4 = 5, c_5 = 9\}$

In[37]:=
```
HREConstantTermVector[M, {{0}, {0}, {0}, {5}, {9}}]
```

Out[37]=
$$\left\{\left\{\frac{5\, m_{1,4}}{4} + \frac{9\, m_{1,5}}{4}\right\}, \left\{\frac{5\, m_{2,4}}{4} + \frac{9\, m_{2,5}}{4}\right\}, \left\{\frac{5\, m_{3,4}}{4} + \frac{9\, m_{3,5}}{4}\right\}\right\}$$

Auxiliary function that transform an upper triangle matrix into a full and reciprocal matrix

In[38]:=
```
M = RecreatePCMatrix[{{1, 3/5, 4/7, 5/8, 5/10}, {0, 1, 5/7, 5/2, 10/3}, {0, 0, 1, 7/2, 4}, {0, 0, 0, 1, 4/3}, {0, 0, 0, 0, 1}}]
```

Out[38]=
$$\left\{\left\{1, \frac{3}{5}, \frac{4}{7}, \frac{5}{8}, \frac{1}{2}\right\}, \left\{\frac{5}{3}, 1, \frac{5}{7}, \frac{5}{2}, \frac{10}{3}\right\}, \left\{\frac{7}{4}, \frac{7}{5}, 1, \frac{7}{2}, 4\right\}, \left\{\frac{8}{5}, \frac{2}{5}, \frac{2}{7}, 1, \frac{4}{3}\right\}, \left\{2, \frac{3}{10}, \frac{1}{4}, \frac{3}{4}, 1\right\}\right\}$$

In[39]:=

Defining known and unknown alternatives. It is assumed
that c_2 and c_3 are known and equal 5 and 7 correspondingly

In[40]:=
```
mk = {{0}, {5}, {7}, {0}, {0}}
```

Out[40]=
```
{{0}, {5}, {7}, {0}, {0}}
```

Calculate the HRE ranking vector for unknown alternatives i.e. for c_1, c_4 and c_5 only

In[41]:=
```
mu = N[HREPartialRank[M, mk]]
```

Out[41]=
```
{{2.52765}, {2.88338}, {2.61696}}
```

Calculate the full HRE ranking for the given input matrix M and the vector mk

In[42]:=
```
mu = N[HREFullRank[M, mk]]
```

Out[42]=
```
{2.52765, 5., 7., 2.88338, 2.61696}
```

Calculate the full HRE ranking rescaled so that all its entries sum up to 1

In[43]:=
```
mu = N[HRERescaledRank[M, mk]]
```

Out[43]=
```
{0.126206, 0.249651, 0.349511, 0.143967, 0.130665}
```

Heuristic Incomplete Rating Estimation Method (additive)

Generate incomplete and irreducible PCM and HRE auxiliary matrix and vector

In[44]:=
```
M = RandomlyDecompletedMatrix[SymbPCM[m, 7], 7, "?"];
```

In[45]:=
```
M // MatrixForm
```

Out[45]//MatrixForm=

$$\begin{pmatrix} 1 & m_{1,2} & ? & m_{1,4} & m_{1,5} & ? & m_{1,7} \\ m_{2,1} & 1 & m_{2,3} & m_{2,4} & m_{2,5} & m_{2,6} & ? \\ ? & m_{3,2} & 1 & m_{3,4} & m_{3,5} & ? & ? \\ m_{4,1} & m_{4,2} & m_{4,3} & 1 & m_{4,5} & m_{4,6} & ? \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & 1 & ? & m_{5,7} \\ ? & m_{6,2} & ? & m_{6,4} & ? & 1 & m_{6,7} \\ m_{7,1} & ? & ? & ? & m_{7,5} & m_{7,6} & 1 \end{pmatrix}$$

In[46]:=
```
vect = {0, 2, 0, 0, 3, 0, 0};
```

In[47]:=
```
MHRE = HREIncompleteMatrix[M, vect, "?"];
```

In[48]:=
```
MHRE // MatrixForm
```

Out[48]//MatrixForm=

$$\begin{pmatrix} 1 & 0 & -\frac{m_{1,4}}{4} & 0 & -\frac{m_{1,7}}{4} \\ 0 & 1 & -\frac{m_{3,4}}{3} & 0 & 0 \\ -\frac{m_{4,1}}{5} & -\frac{m_{4,3}}{5} & 1 & -\frac{m_{4,6}}{5} & 0 \\ 0 & 0 & -\frac{m_{6,4}}{3} & 1 & -\frac{m_{6,7}}{3} \\ -\frac{m_{7,1}}{3} & 0 & 0 & -\frac{m_{7,6}}{3} & 1 \end{pmatrix}$$

In[49]:=
```
VHRE = HREIncompleteConstantTermsVector[M, vect, "?"];
```

In[50]:=
```
VHRE // MatrixForm
```

Out[50]//MatrixForm=

$$\begin{pmatrix} \frac{m_{1,2}}{2} + \frac{3\,m_{1,5}}{4} \\ \frac{2\,m_{3,2}}{3} + m_{3,5} \\ \frac{2\,m_{4,2}}{5} + \frac{3\,m_{4,5}}{5} \\ \frac{2\,m_{6,2}}{3} \\ m_{7,5} \end{pmatrix}$$

Generate a numeric incomplete matrix

In[51]:=
```
M = RandomlyDecompletedMatrix[RandomMatrix[7, 7], 7, "?"];
```

In[52]:=
```
M // MatrixForm
```

Out[52]//MatrixForm=

$$\begin{pmatrix} 1 & 0.170137 & 0.05752 & 1.4726 & ? & ? & 0.224226 \\ 5.87761 & 1 & ? & 2.46802 & ? & ? & 0.106583 \\ 17.3853 & ? & 1 & 34.1492 & 39.3557 & ? & 4.71126 \\ 0.679071 & 0.405183 & 0.0292832 & 1 & 10.0236 & 2.20003 & 0.0448818 \\ ? & ? & 0.0254093 & 0.0997646 & 1 & 8.49203 & 0.143613 \\ ? & ? & ? & 0.45454 & 0.117757 & 1 & ? \\ 4.45979 & 9.38234 & 0.212257 & 22.2808 & 6.96316 & ? & 1 \end{pmatrix}$$

In[53]:=
```
vect = {{0}, {2}, {0}, {0}, {3}, {0}, {0}};
```

... and calculate numeric rankings: partial, full and full rescaled.

In[54]:=
```
HREIncompletePartialRanking[M, vect, "?"]
```

Out[54]=
```
{{10.1503}, {224.296}, {8.6642}, {2.14575}, {65.1151}}
```

In[55]:=
```
HREIncompleteFullRanking[M, vect, "?"]
```

Out[55]=
```
{10.1503, 2, 224.296, 8.6642, 3, 2.14575, 65.1151}
```

In[56]:=
```
HREIncompleteRescaledRank[M, vect, "?"]
```

Out[56]=
```
{0.0321852, 0.00634173, 0.711212, 0.027473, 0.0095126, 0.00680388, 0.206471}
```

It is worth to note that incomplete HRE is an extension of classical HRE, i.e. it works also for complete PC matrices. E.g.

In[57]:=
```
P = RandomMatrix[7, 7];
```

In[58]:=
```
P // MatrixForm
```

Out[58]//MatrixForm=

$$\begin{pmatrix} 1. & 29.4251 & 8.02527 & 5.82485 & 0.573607 & 9.82567 & 7.16983 \\ 0.0339846 & 1. & 4.38134 & 0.150961 & 0.0056832 & 0.0244627 & 0.0738056 \\ 0.124606 & 0.22824 & 1. & 0.0761037 & 0.0953373 & 0.064944 & 0.775606 \\ 0.171678 & 6.62422 & 13.14 & 1. & 4.06567 & 0.367518 & 6.9047 \\ 1.74335 & 175.957 & 10.4891 & 0.245962 & 1. & 1.0132 & 1.95257 \\ 0.101774 & 40.8786 & 15.3979 & 2.72095 & 0.986972 & 1. & 1.11905 \\ 0.139473 & 13.5491 & 1.28931 & 0.144829 & 0.512145 & 0.893617 & 1. \end{pmatrix}$$

Then we may observe the equality of rankings:

In[59]:=
```
HREIncompleteRescaledRank[P, vect, "?"]
```

Out[59]=
```
{0.570422, 0.00411872, 0.0236484, 0.155705, 0.00617808, 0.181037, 0.058891}
```

In[60]:=
```
N@HRERescaledRank[P, vect]
```

Out[60]=
```
{0.570422, 0.00411872, 0.0236484, 0.155705, 0.00617808, 0.181037, 0.058891}
```

I.e.

In[61]:=
```
HREIncompleteRescaledRank[P, vect, "?"] == N@HRERescaledRank[P, vect]
```

Out[61]=
```
True
```

Heuristic Rating Estimation Method (multiplicative/geometric)

Further references could be found in papers:
 * Konrad Kułakowski, Grobler-Dębska Katarzyna, Wąs Jarosław, Heuristic rating estimation - geometric approach, http://arxiv.org/abs/1404.6981

In[62]:=
```
HREGeomMatrix[P, {0, 0, 0, q4, q5}]
```

Out[62]=
```
{{4, -1, -1}, {-1, 4, -1}, {-1, -1, 4}}
```

Calculate the HRE constant term vector

In[63]:=
```
HREGeomConstantTermVector[P, {0, 0, 0, q4, q5}]
```

Out[63]=

$$\left\{\left\{\frac{\text{Log}[q_4\, q_5\, m_{1,2}\, m_{1,3}\, m_{1,4}\, m_{1,5}]}{\text{Log}[10]}\right\},\ \left\{\frac{\text{Log}[q_4\, q_5\, m_{2,1}\, m_{2,3}\, m_{2,4}\, m_{2,5}]}{\text{Log}[10]}\right\},\ \left\{\frac{\text{Log}[q_4\, q_5\, m_{3,1}\, m_{3,2}\, m_{3,4}\, m_{3,5}]}{\text{Log}[10]}\right\}\right\}$$

Calculate the HRE geometric ranking vector for unknown alternatives i.e. for c_1, c_4 and c_5 only

```
In[64]:= mu = N[HREGeomPartialRank[M, mk]]
```

Out[64]=

$$\left\{\left\{10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}\right\},\right.$$
$$\left\{10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right\},$$
$$\left.\left\{10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right\}\right\}$$

Calculate the full HRE geometric ranking for the given input matrix M and the vector mk

```
In[65]:= mu = N[HREGeomFullRank[M, mk]]
```

Out[65]=

$$\left\{10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])},\right.$$
$$5.,\ 7.,\ 10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])},$$
$$\left.10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right\}$$

Calculate the full HRE geometric ranking, rescaled so that all its entries sum up to 1

```
In[66]:= mu = N[HREGeomRescaledRank[M, mk]]
```

Out[66]=

$$\left\{10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}\Big/\right.$$
$$\left(12.+10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}+\right.$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}+$$
$$\left.10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right),$$
$$5.\Big/\left(12.+10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}+\right.$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}+$$
$$\left.10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right),$$
$$7.\Big/\left(12.+10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}+\right.$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}+$$
$$\left.10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right),$$
$$10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\Big/$$
$$\left(12.+10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}+\right.$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}+$$
$$\left.10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right),$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\Big/$$
$$\left(12.+10.^{-1.\,(0.0197938-0.0155105\,\text{Log}[0.108204\,?^2]-0.0775526\,\text{Log}[0.113098\,?^2])}+\right.$$
$$10.^{-1.\,(0.0197938-0.0775526\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}+$$
$$\left.\left.10.^{-1.\,(0.0989688-0.0155105\,\text{Log}[0.108204\,?^2]-0.0155105\,\text{Log}[0.113098\,?^2])}\right)\right\}$$

Show intermediate (before raisng up to the power) HRE geometric partial rank vector

In[67]:=
```
N[HREGeomIntermediateRank[M, mk]]
```

Out[67]=
$$\{\{-1.\ (0.0197938 - 0.0155105\ \text{Log}[0.108204\ ?^2] - 0.0775526\ \text{Log}[0.113098\ ?^2])\},$$
$$\{-1.\ (0.0989688 - 0.0155105\ \text{Log}[0.108204\ ?^2] - 0.0155105\ \text{Log}[0.113098\ ?^2])\},$$
$$\{-1.\ (0.0197938 - 0.0775526\ \text{Log}[0.108204\ ?^2] - 0.0155105\ \text{Log}[0.113098\ ?^2])\}\}$$

Bana e Costa and Vansnick's Condition of Order Preservation test

Methods:
COP1Check[M, mu]
COP2Check[M, mu]
COP1ViolationList[M, mu]
COP2ViolationList[M, mu]
are depricated

In[68]:=
```
M = RecreatePCMatrix[ ... ];
```
$$\begin{pmatrix} 1 & \frac{3}{5} & \frac{4}{7} & \frac{5}{8} & \frac{5}{10} \\ 0 & 1 & \frac{5}{7} & \frac{5}{2} & \frac{10}{3} \\ 0 & 0 & 1 & \frac{7}{2} & 4 \\ 0 & 0 & 0 & 1 & \frac{4}{3} \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Let calculate the eigenvalue ranking

In[69]:=
```
rank = EigenvalueRank[M]
```

Out[69]=
```
{0.119092, 0.27476, 0.356526, 0.130954, 0.118669}
```

.... and check whether the first Bana e Costa and Vansnick condition "condition of order preservation - COP" is satisfied

In[70]:=
```
COP1Check[M, mu]
```

Out[70]=
```
True
```

then check whether the second Bana e Costa and Vansnick condition (preserving intensity of preferences postulate) is satisfied

In[71]:=
```
COP2Check[M, mu]
```

Out[71]=
```
True
```

Prints the list of pairs for which the 1st COP is not satisfied

In[72]:=
```
COP1ViolationList[M, mu]
```

Out[72]=
```
{}
```

Prints the list of pairs of pairs for which the 2nd COP is not satisfied

In[73]:=
```
COP2ViolationList[M, mu]
```

Out[73]=
```
{}
```

Here are the new COP calculation methods

In[74]:=
```
M = {{1, 171/172, 86, 65/22, 121/191, 45, 199/3, 200/3, 62/101},
 {172/171, 1, 165/2, 215/87, 29/96, 127/3, 211/3, 211/4, 118/219},
 {1/86, 2/165, 1, 3/113, 1/190, 67/129, 139/196, 77/171, 2/155},
 {22/65, 87/215, 113/3, 1, 10/91, 31/2, 49/2, 199/21, 45/139},
 {191/121, 96/29, 190, 91/10, 1, 156, 138, 169/2, 31/14},
 {1/45, 3/127, 129/67, 2/31, 1/156, 1, 165/181, 1, 1/53},
 {3/199, 3/211, 196/139, 2/49, 1/138, 181/165, 1, 145/173, 3/163},
 {3/200, 4/211, 171/77, 21/199, 2/169, 1, 173/145, 1, 2/105},
 {101/62, 219/118, 155/2, 139/45, 14/31, 53, 163/3, 105/2, 1}};
```

So let us calulate all the places where $m_i > m_j$ but $w_i <= w_j$

In[75]:=
```
OrderPreservationViolationMatrix[M, EigenvalueRank@M] // MatrixForm
```

Out[75]//MatrixForm=
```
OrderPreservationViolationMatrix[{{1, 171/172, 86, 65/22, 121/191, 45, 199/3, 200/3, 62/101},
  {172/171, 1, 165/2, 215/87, 29/96, 127/3, 211/3, 211/4, 118/219},
  {1/86, 2/165, 1, 3/113, 1/190, 67/129, 139/196, 77/171, 2/155},
  {22/65, 87/215, 113/3, 1, 10/91, 31/2, 49/2, 199/21, 45/139},
  {191/121, 96/29, 190, 91/10, 1, 156, 138, 169/2, 31/14},
  {1/45, 3/127, 129/67, 2/31, 1/156, 1, 165/181, 1, 1/53},
  {3/199, 3/211, 196/139, 2/49, 1/138, 181/165, 1, 145/173, 3/163},
  {3/200, 4/211, 171/77, 21/199, 2/169, 1, 173/145, 1, 2/105},
  {101/62, 219/118, 155/2, 139/45, 14/31, 53, 163/3, 105/2, 1}},
 {0.177257, 0.152033, 0.00189828, 0.0560799, 0.406428,
  0.00336018, 0.00283829, 0.00378611, 0.196318}]
```

So the list of pairs of alternatives that violates COP can be obtained as follows

In[76]:=
```
Position[OrderPreservationViolationMatrix[M, EigenvalueRank@M], -1]
```

Out[76]=
```
{}
```

The violation index is

In[77]:=
```
OrderPreservationIdx[M, EigenvalueRank@M]
```

Out[77]=

$$\text{OrderPreservationIdx}\left[\left\{\left\{1, \frac{171}{172}, 86, \frac{65}{22}, \frac{121}{191}, 45, \frac{199}{3}, \frac{200}{3}, \frac{62}{101}\right\},\right.\right.$$
$$\left\{\frac{172}{171}, 1, \frac{165}{2}, \frac{215}{87}, \frac{29}{96}, \frac{127}{3}, \frac{211}{3}, \frac{211}{4}, \frac{118}{219}\right\},$$
$$\left\{\frac{1}{86}, \frac{2}{165}, 1, \frac{3}{113}, \frac{1}{190}, \frac{67}{129}, \frac{139}{196}, \frac{77}{171}, \frac{2}{155}\right\},$$
$$\left\{\frac{22}{65}, \frac{87}{215}, \frac{113}{3}, 1, \frac{10}{91}, \frac{31}{2}, \frac{49}{2}, \frac{199}{21}, \frac{45}{139}\right\},$$
$$\left\{\frac{191}{121}, \frac{96}{29}, 190, \frac{91}{10}, 1, 156, 138, \frac{169}{2}, \frac{31}{14}\right\},$$
$$\left\{\frac{1}{45}, \frac{3}{127}, \frac{129}{67}, \frac{2}{31}, \frac{1}{156}, 1, \frac{165}{181}, 1, \frac{1}{53}\right\},$$
$$\left\{\frac{3}{199}, \frac{3}{211}, \frac{196}{139}, \frac{2}{49}, \frac{1}{138}, \frac{181}{165}, 1, \frac{145}{173}, \frac{3}{163}\right\},$$
$$\left\{\frac{3}{200}, \frac{4}{211}, \frac{171}{77}, \frac{21}{199}, \frac{2}{169}, 1, \frac{173}{145}, 1, \frac{2}{105}\right\},$$
$$\left.\left\{\frac{101}{62}, \frac{219}{118}, \frac{155}{2}, \frac{139}{45}, \frac{14}{31}, 53, \frac{163}{3}, \frac{105}{2}, 1\right\}\right\},$$
$$\{0.177257, 0.152033, 0.00189828, 0.0560799, 0.406428,$$
$$0.00336018, 0.00283829, 0.00378611, 0.196318\}\Big]$$

Similarly we can calculate positive intensity violations i.e. when $m_i/m_j > m_p/m_q > 1$ but $w_i/w_j <= w_p/w_q$ and negative intensity violations $m_i/m_j < m_p/m_q < 1$ but $w_i/w_j >= w_p/w_q$. So quadruples that violates intesity are:

In[78]:=
```
Position[IntensityPreservationViolation4DPositive[M, EigenvalueRank@M], -1]
```

Out[78]=

$$\{\}$$

similarly

In[79]:=
```
Position[IntensityPreservationViolation4DNegative[M, EigenvalueRank@M], -1]
```

Out[79]=

$$\{\}$$

And both:

In[80]:=
```
Position[IntensityPreservationViolation4D[M, EigenvalueRank@M], -1]
```

Out[80]=

$$\{\}$$

The intensity violation index are:

In[81]:=
```
IntensityPreservationIdx[M, EigenvalueRank@M]
```

Out[81]=

$$\text{IntensityPreservationIdx}\left[\left\{\left\{1, \frac{171}{172}, 86, \frac{65}{22}, \frac{121}{191}, 45, \frac{199}{3}, \frac{200}{3}, \frac{62}{101}\right\}, \left\{\frac{172}{171}, 1, \frac{165}{2}, \frac{215}{87}, \frac{29}{96}, \frac{127}{3}, \frac{211}{3}, \frac{211}{4}, \frac{118}{219}\right\}, \left\{\frac{1}{86}, \frac{2}{165}, 1, \frac{3}{113}, \frac{1}{190}, \frac{67}{129}, \frac{139}{196}, \frac{77}{171}, \frac{2}{155}\right\}, \left\{\frac{22}{65}, \frac{87}{215}, \frac{113}{3}, 1, \frac{10}{91}, \frac{31}{2}, \frac{49}{2}, \frac{199}{21}, \frac{45}{139}\right\}, \left\{\frac{191}{121}, \frac{96}{29}, 190, \frac{91}{10}, 1, 156, 138, \frac{169}{2}, \frac{31}{14}\right\}, \left\{\frac{1}{45}, \frac{3}{127}, \frac{129}{67}, \frac{2}{31}, \frac{1}{156}, 1, \frac{165}{181}, 1, \frac{1}{53}\right\}, \left\{\frac{3}{199}, \frac{3}{211}, \frac{196}{139}, \frac{2}{49}, \frac{1}{138}, \frac{181}{165}, 1, \frac{145}{173}, \frac{3}{163}\right\}, \left\{\frac{3}{200}, \frac{4}{211}, \frac{171}{77}, \frac{21}{199}, \frac{2}{169}, 1, \frac{173}{145}, 1, \frac{2}{105}\right\}, \left\{\frac{101}{62}, \frac{219}{118}, \frac{155}{2}, \frac{139}{45}, \frac{14}{31}, 53, \frac{163}{3}, \frac{105}{2}, 1\right\}\right\}, \{0.177257, 0.152033, 0.00189828, 0.0560799, 0.406428, 0.00336018, 0.00283829, 0.00378611, 0.196318\}\right]$$

Koczkodaj's Iterative Inconsistency Reduction algorithm

Calculate the value of the Koczkodaj inconsistency index

In[82]:=
```
N[KoczkodajIdx[M]]
```

Out[82]=

```
0.583761
```

Prints the worst Koczkodaj triad in M. As we can see it is $m_{5,3} == \frac{1}{4}$, $m_{3,1} == \frac{7}{4}$, $m_{5,1} == 2$. The value of inconsistency introduced by this triad is $\frac{25}{32}$

In[83]:=
```
KoczkodajTheWorstTriad[M]
```

Out[83]=

$$\left\{\{9, 7, 2\}, \left\{\frac{163}{3}, \frac{3}{211}, \frac{219}{118}\right\}, \frac{26\,975}{46\,209}\right\}$$

Perform one step of the Koczkodaj inconsistency reduction algorithm. On the output there is a new slightly modified matrix M2 that is expected to be more consistent than M

In[84]:=
```
M2 = N[KoczkodajImproveMatrixStep[M]]
```

Out[84]=
```
{{1., 0.994186, 86., 2.95455, 0.633508, 45., 66.3333, 66.6667, 0.613861},
 {1.00585, 1., 82.5, 2.47126, 0.302083, 42.3333, 52.5141, 52.75, 0.721645},
 {0.0116279, 0.0121212, 1., 0.0265487,
  0.00526316, 0.51938, 0.709184, 0.450292, 0.0129032},
 {0.338462, 0.404651, 37.6667, 1., 0.10989, 15.5, 24.5, 9.47619, 0.323741},
 {1.57851, 3.31034, 190., 9.1, 1., 156., 138., 84.5, 2.21429},
 {0.0222222, 0.023622, 1.92537, 0.0645161, 0.00641026, 1.,
  0.911602, 1., 0.0188679}, {0.0150754, 0.0190425, 1.41007,
  0.0408163, 0.00724638, 1.09697, 1., 0.83815, 0.0137419},
 {0.015, 0.0189573, 2.22078, 0.105528, 0.0118343, 1., 1.1931, 1., 0.0190476},
 {1.62903, 1.38572, 77.5, 3.08889, 0.451613, 53., 72.7699, 52.5, 1.}}
```

In[85]:=
```
KoczkodajIdx[M2]
```

Out[85]=
```
0.580032
```

Group decision making

Let us consider three different PC matrices X, Y and Z that come from three different experts

In[86]:=
$$X = \begin{pmatrix} 1 & x_{12} & x_{13} & x_{14} \\ x_{21} & 1 & x_{23} & x_{24} \\ x_{31} & x_{32} & 1 & x_{34} \\ x_{41} & x_{42} & x_{43} & 1 \end{pmatrix};$$

In[87]:=
$$Y = \begin{pmatrix} 1 & y_{12} & y_{13} & y_{14} \\ y_{21} & 1 & y_{23} & y_{24} \\ y_{31} & y_{32} & 1 & y_{34} \\ y_{41} & y_{42} & y_{43} & 1 \end{pmatrix};$$

In[88]:=
$$Z = \begin{pmatrix} 1 & z_{12} & z_{13} & z_{14} \\ z_{21} & 1 & z_{23} & z_{24} \\ z_{31} & z_{32} & 1 & z_{34} \\ z_{41} & z_{42} & z_{43} & 1 \end{pmatrix};$$

Then it is possible to aggregate the results using appropriate functions.

Aggregate Individual Judgments additively (AIJadd):

In[89]:=
```
AIJadd[X, Y, Z] // MatrixForm
```

Out[89]//MatrixForm=
$$\begin{pmatrix} 1 & \frac{1}{3}(x_{12}+y_{12}+z_{12}) & \frac{1}{3}(x_{13}+y_{13}+z_{13}) & \frac{1}{3}(x_{14}+y_{14}+z_{14}) \\ \frac{1}{3}(x_{21}+y_{21}+z_{21}) & 1 & \frac{1}{3}(x_{23}+y_{23}+z_{23}) & \frac{1}{3}(x_{24}+y_{24}+z_{24}) \\ \frac{1}{3}(x_{31}+y_{31}+z_{31}) & \frac{1}{3}(x_{32}+y_{32}+z_{32}) & 1 & \frac{1}{3}(x_{34}+y_{34}+z_{34}) \\ \frac{1}{3}(x_{41}+y_{41}+z_{41}) & \frac{1}{3}(x_{42}+y_{42}+z_{42}) & \frac{1}{3}(x_{43}+y_{43}+z_{43}) & 1 \end{pmatrix}$$

Aggregate Individual Judgments geometrically (AIJgeom)

In[90]:=
```
AIJgeom[X, Y, Z] // MatrixForm
```

Out[90]//MatrixForm=

$$\begin{pmatrix} 1 & (x_{12}\,y_{12}\,z_{12})^{1/3} & (x_{13}\,y_{13}\,z_{13})^{1/3} & (x_{14}\,y_{14}\,z_{14})^{1/3} \\ (x_{21}\,y_{21}\,z_{21})^{1/3} & 1 & (x_{23}\,y_{23}\,z_{23})^{1/3} & (x_{24}\,y_{24}\,z_{24})^{1/3} \\ (x_{31}\,y_{31}\,z_{31})^{1/3} & (x_{32}\,y_{32}\,z_{32})^{1/3} & 1 & (x_{34}\,y_{34}\,z_{34})^{1/3} \\ (x_{41}\,y_{41}\,z_{41})^{1/3} & (x_{42}\,y_{42}\,z_{42})^{1/3} & (x_{43}\,y_{43}\,z_{43})^{1/3} & 1 \end{pmatrix}$$

Note that these two functions works also for result lists i.e.:

In[91]:=
```
AIJgeom[{x₁, x₂, x₃, x₄}, {y₁, y₂, y₃, y₄}]
```

Out[91]=

$$\left\{\sqrt{x_1\,y_1},\ \sqrt{x_2\,y_2},\ \sqrt{x_3\,y_3},\ \sqrt{x_4\,y_4}\right\}$$

In[92]:=
```
AIJadd[{x₁, x₂, x₃, x₄}, {y₁, y₂, y₃, y₄}]
```

Out[92]=

$$\left\{\frac{1}{2}\,(x_1+y_1),\ \frac{1}{2}\,(x_2+y_2),\ \frac{1}{2}\,(x_3+y_3),\ \frac{1}{2}\,(x_4+y_4)\right\}$$

The aggregated priority vector can also be computed as the result of GED (group euclidian distance) minimization. For example GED for X,Y,Z can be calculated as GED[X,Y,Z]. In this particular case GED[X,Y,Z] equals:

In[93]:=
```
√((((-w1/w2 + x12)^2 + (-w1/w3 + x13)^2 + (-w1/w4 + x14)^2 + (-w2/w1 + x21)^2 + (-w2/w3 + x23)^2 + (-w2/w4 + x24)^2 +
(-w3/w1 + x31)^2 + (-w3/w2 + x32)^2 + (-w3/w4 + x34)^2 + (-w4/w1 + x41)^2 + (-w4/w2 + x42)^2 +
(-w4/w3 + x43)^2 + (-w1/w2 + y12)^2 + (-w1/w3 + y13)^2 + (-w1/w4 + y14)^2 + (-w2/w1 + y21)^2 +
(-w2/w3 + y23)^2 + (-w2/w4 + y24)^2 + (-w3/w1 + y31)^2 + (-w3/w2 + y32)^2 + (-w3/w4 + y34)^2 +
(-w4/w1 + y41)^2 + (-w4/w2 + y42)^2 + (-w4/w3 + y43)^2 + (-w1/w2 + z12)^2 + (-w1/w3 + z13)^2 +
(-w1/w4 + z14)^2 + (-w2/w1 + z21)^2 + (-w2/w3 + z23)^2 + (-w2/w4 + z24)^2 + (-w3/w1 + z31)^2 +
(-w3/w2 + z32)^2 + (-w3/w4 + z34)^2 + (-w4/w1 + z41)^2 + (-w4/w2 + z42)^2 + (-w4/w3 + z43)^2));
```

Let us consider the following four PC matrices:

In[94]:=
```
C1 = {{1, 2, 4/5, 9/5, 9}, {1/2, 1, 1/4, 7/4, 5}, {5/4, 4, 1, 7/2, 9}, {5/9, 4/7, 2/7, 1, 5}, {1/9, 1/5, 1/9, 1/5, 1}};
```

In[95]:=
$$C2 = \begin{pmatrix} 1 & 2 & \frac{1}{4} & 1 & 9 \\ \frac{1}{2} & 1 & \frac{8}{9} & \frac{2}{3} & 8 \\ 4 & \frac{9}{8} & 1 & 3 & 9 \\ 1 & \frac{3}{2} & \frac{1}{3} & 1 & 9 \\ \frac{1}{9} & \frac{1}{8} & \frac{1}{9} & \frac{1}{9} & 1 \end{pmatrix};$$

In[96]:=
$$C3 = \begin{pmatrix} 1 & 1 & \frac{3}{8} & \frac{5}{3} & 9 \\ 1 & 1 & \frac{2}{5} & 1 & 9 \\ \frac{8}{3} & \frac{5}{2} & 1 & 6 & 9 \\ \frac{3}{5} & 1 & \frac{1}{6} & 1 & 5 \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} & \frac{1}{5} & 1 \end{pmatrix};$$

In[97]:=
$$C4 = \begin{pmatrix} 1 & 1 & \frac{3}{7} & \frac{7}{4} & 5 \\ 1 & 1 & \frac{8}{9} & 1 & 7 \\ \frac{7}{3} & \frac{9}{8} & 1 & 2 & 9 \\ \frac{4}{7} & 1 & \frac{1}{2} & 1 & 5 \\ \frac{1}{5} & \frac{1}{7} & \frac{1}{9} & \frac{1}{5} & 1 \end{pmatrix};$$

The aggregated priority vector minimizing GED can be calcualted as follows:

In[98]:=
```
GEDMin[C1, C2, C3, C4]
```

Out[98]=
```
{0.254041, 0.224838, 0.313325, 0.175409, 0.0323872}
```

For minimization a differential evolution procedure is used.

Let us look, for example, at the same matrices aggregated using AIJ + GMM

In[99]:=
```
GeometricRescaledRank@N@AIJgeom[C1, C2, C3, C4]
```

Out[99]=
```
{0.224222, 0.18592, 0.404349, 0.154636, 0.0308722}
```

Aggregate with the specified priorities (priorities do not have to sum up to 1)

In[100]:=
```
aggregatedMatrix = N@AIJgeomWithPriorities[{C1, C2, C3, C4}, {1, 2, 3, 4}];
```

In[101]:=
```
aggregatedMatrix // MatrixForm
```

Out[101]//MatrixForm=
$$\begin{pmatrix} 1. & 1.23114 & 0.393474 & 1.54631 & 7.11432 \\ 0.812252 & 1. & 0.616198 & 0.975182 & 7.49595 \\ 2.54147 & 1.62285 & 1. & 3.18925 & 9. \\ 0.6467 & 1.02545 & 0.313553 & 1. & 5.62373 \\ 0.140562 & 0.133405 & 0.111111 & 0.177818 & 1. \end{pmatrix}$$

and the priority vector is:

In[102]:=
```
GMM@aggregatedMatrix
```
Out[102]=
```
{0.21402, 0.198512, 0.397907, 0.15802, 0.0315411}
```

Obviously,

In[103]:=
```
GMM@N@AIJgeom[C1, C2, C3, C4] ==
 GMM@N@AIJgeomWithPriorities[{C1, C2, C3, C4}, {1, 1, 1, 1}]
```
Out[103]=
```
True
```

which means that AIJgeomWithPriorities[] when priorities are identical works exactly the same as AIJgeom[]

One can also use additive aggregation with priorities

In[104]:=
```
aggregatedMatrix =
  N@AIJadditiveWithPriorities[{Log@C1, Log@C2, Log@C3, Log@C4}, {1, 2, 3, 4}];
```

In[105]:=
```
Exp[aggregatedMatrix] // MatrixForm
```
Out[105]//MatrixForm=

$$\begin{pmatrix} 1. & 1.23114 & 0.393474 & 1.54631 & 7.11432 \\ 0.812252 & 1. & 0.616198 & 0.975182 & 7.49595 \\ 2.54147 & 1.62285 & 1. & 3.18925 & 9. \\ 0.6467 & 1.02545 & 0.313553 & 1. & 5.62373 \\ 0.140562 & 0.133405 & 0.111111 & 0.177818 & 1. \end{pmatrix}$$

And obviously:

In[106]:=
```
SymArr[s_, n_] := Table[s_{i,j}, {i, 1, n}, {j, 1, n}];
```

In[107]:=
```
AIJadd @@ SymArr[u, 4] // MatrixForm
```
Out[107]//MatrixForm=

$$\begin{pmatrix} \frac{1}{4}\left(u_{1,1}+u_{2,1}+u_{3,1}+u_{4,1}\right) \\ \frac{1}{4}\left(u_{1,2}+u_{2,2}+u_{3,2}+u_{4,2}\right) \\ \frac{1}{4}\left(u_{1,3}+u_{2,3}+u_{3,3}+u_{4,3}\right) \\ \frac{1}{4}\left(u_{1,4}+u_{2,4}+u_{3,4}+u_{4,4}\right) \end{pmatrix}$$

is exactly the same as

In[108]:=
```
Simplify@AIJadditiveWithPriorities[SymArr[u, 4], Table[1 / 4, 4]] // MatrixForm
```
Out[108]//MatrixForm=

$$\begin{pmatrix} \frac{1}{4}\left(u_{1,1}+u_{2,1}+u_{3,1}+u_{4,1}\right) \\ \frac{1}{4}\left(u_{1,2}+u_{2,2}+u_{3,2}+u_{4,2}\right) \\ \frac{1}{4}\left(u_{1,3}+u_{2,3}+u_{3,3}+u_{4,3}\right) \\ \frac{1}{4}\left(u_{1,4}+u_{2,4}+u_{3,4}+u_{4,4}\right) \end{pmatrix}$$

Pairwise Comparisons Matrix - Harker Method

The idea comes from P. T. Harker, Alternative Modes of Questioning in The Analytic Hierarchy Process, Math Modeling, 1987. The

Let M be an incomplete pairwise comparisons matrix.

In[109]:=
```
M = {{1, 2, □}, {1/2, 1, 2}, {□, 1/2, 1}};
```

Thus, to compute the ranking based on M we need to compute matrix A (hereinafter referred to as Harker matrix)

In[110]:=
```
A = HarkerMatrix[M];
```

In[111]:=
```
A // MatrixForm
```

Out[111]//MatrixForm=

$$\begin{pmatrix} 2 & 2 & 0 \\ \frac{1}{2} & 1 & 2 \\ 0 & \frac{1}{2} & 2 \end{pmatrix}$$

Then compute the eigenvalue based ranking and inconsistency index as usuall:

In[112]:=
```
EigenvalueRank[A]
```

Out[112]=
```
{0.571429, 0.285714, 0.142857}
```

In[113]:=
```
SaatyIdx[A]
```

Out[113]=
```
0
```

The ranking can be computed for incomplete matrix only if it is irreducible, i.e. associated adjacency graph is irreducible. To check whether matrix is irreducible we can use IrreducibleMatrixQ

In[114]:=
```
IrreducibleMatrixQ[M]
```

Out[114]=
```
True
```

As it easy to observe, the Harker method works for both complete and incomplete matrices. For complete matrices it is just a EVM method. For this reason there is another, general function EVM[M] that can handle both complete and incomplete matrices.

In[115]:=
```
EVM[M]
```

Out[115]=
```
{0.571429, 0.285714, 0.142857}
```

Incomplete Pairwise Comparisons Matrix - Geometric Mean Method for Incomplete matrices

The idea comes from Sandor et. al, On optimal completion of incomplete pairwise comparison matrices, Mathematical and Computer Modelling, 2010. Note that this method works for both complete and incomplete matrices. When the matrix is complete it simply boils down to GMM method.

Let M be an incomplete pairwise comparisons matrix.

In[116]:=

$$M = \begin{pmatrix} 1 & 2 & 3 & \square & \square \\ 1/2 & 1 & 2 & \square & \square \\ 1/3 & 1/2 & 1 & 2 & \square \\ \square & \square & 1/2 & 1 & 3 \\ \square & \square & \square & 1/3 & 1 \end{pmatrix};$$

Thus, to compute the GMM ranking based on M we need to call GMM[M]

In[117]:=

```
GMM[M]
```

Out[117]=

$$\left\{ e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \Big/ \left(e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \right), \right.$$

$$e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} \Big/ \left(e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \right),$$

$$e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} \Big/ \left(e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \right),$$

$$e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} \Big/ \left(e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \right),$$

$$\left. e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} \Big/ \left(e^{\frac{1}{25}\left(8\,\text{Log}\left[\frac{3}{2}\right]-21\,\text{Log}[3]-12\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(13\,\text{Log}\left[\frac{3}{2}\right]-6\,\text{Log}[3]-7\,\text{Log}[6]\right)} + e^{\frac{1}{25}\left(-2\,\text{Log}\left[\frac{3}{2}\right]-\text{Log}[3]+3\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+19\,\text{Log}[6]\right)} + e^{\frac{1}{75}\left(-21\,\text{Log}\left[\frac{3}{2}\right]+27\,\text{Log}[3]+44\,\text{Log}[6]\right)} \right) \right\}$$

There is also a numerical shorthand for this function:

In[118]:=
```
NGMM[M]
```
Out[118]=
```
{0.4866, 0.267786, 0.147368, 0.0736842, 0.0245614}
```

This functions also works for complete matrices, and by definition, they are equal to GMM method

In[119]:=
```
M = {{1, 2, 3, 1, 2}, {1/2, 1, 2, 1, 1}, {1/3, 1/2, 1, 2, 1}, {1, 1, 1/2, 1, 3}, {1/2, 1, 1, 1/3, 1}};
```

In[120]:=
```
N@GMM[M] == GeometricRescaledRank[M]
```
Out[120]=
```
True
```

The ranking errors and the ranking discrepancies

Let the w = $\{w_1, w_2, w_3\}$ be the ranking vector whilst M a PC matrix. An error is defined as $e_{ij} = m_{ij}(w_j / w_i)$ and it corresponds to the "discrepancy" between individual judgment m_{ij} and the ranking result. Hence, the error matrix is just E = $[e_{ij}]$. E.g.:

In[121]:=
```
M = RecreatePCMatrix[{{1, 3/5, 4/7, 5/8, 5/10}, {0, 1, 5/7, 5/2, 10/3}, {0, 0, 1, 7/2, 4}, {0, 0, 0, 1, 4/3}, {0, 0, 0, 0, 1}}];
```

In[122]:=
```
ErrorMatrix[M, EigenvalueRank[M]] // MatrixForm
```
Out[122]//MatrixForm=
$$\begin{pmatrix} 1. & 0.722398 & 0.584559 & 1.45507 & 2.00712 \\ 1.38428 & 1. & 1.07892 & 0.839258 & 0.694602 \\ 1.71069 & 0.92685 & 1. & 0.777866 & 0.751091 \\ 0.687253 & 1.19153 & 1.28557 & 1. & 0.827639 \\ 0.498227 & 1.43967 & 1.3314 & 1.20826 & 1. \end{pmatrix}$$

The error matrix is not symmetric. An attempt to symmetrization of error matrix leads to the local discrepancy matrix D = (d_{ij}) where $d_{ij} = \max\{e_{ij} - 1, 1/e_{ij} - 1\}$. Another interesting property of the local discrepancy matrix is the fact that wherever $d_{ij} = 0$ this means that the local discrepancy (error) is 0. If $d_{ij} = X$ this means that the local judgment m_{ij} differs from the ratio $\frac{w_i}{w_j}$ by 100%*X

In[123]:=
```
Chop@LocalDiscrepancyMatrix[M, EigenvalueRank[M]] // MatrixForm
```

Out[123]//MatrixForm=

0	0.384278	0.71069	0.455068	1.00712
0.384278	0	0.0789237	0.191529	0.439673
0.71069	0.0789237	0	0.285569	0.331397
0.455068	0.191529	0.285569	0	0.208257
1.00712	0.439673	0.331397	0.208257	0

The greatest entry of the local discrepancy matrix can be found by using the GlobalDiscrepancy function

In[124]:=
```
GlobalDiscrepancy[M, EigenvalueRank[M]]
```

Out[124]=
```
1.00712
```

Usefull methods built into Mathematica

Spearman Rank Correlation (compare with the example from https://en.wikipedia.org/wiki/Spearman%27s_rank_correlation_coefficient)

In[219]:=
```
Needs["MultivariateStatistics`"];
```

In[220]:=
```
N@SpearmanRankCorrelation[{86, 97, 99, 100, 101, 103, 106, 110, 112, 113},
  {0, 20, 28, 27, 50, 29, 7, 17, 6, 12}]
```

Out[220]=
```
-0.175758
```

In[127]:=
```
r_s = N@SpearmanRankCorrelation[{1.1, 1.57, 0.51, 1.1, 1.1}, {1.2, 1, 2.3, 1, 18}]
```

Out[127]=
```
-0.573539
```

In[128]:=
```
n = Length@{1.1, 1.57, 0.51, 1.1, 1.1};
```

In[129]:=
```
t = N[r_s * Sqrt[(n - 2)/(1 - r_s^2)]]
```

Out[129]=
```
-1.21268
```

Tau Corellation

In[130]:=
```
KendallTau[{86, 97, 99, 100, 101, 103, 106, 110, 112, 113},
 {0, 20, 28, 27, 50, 29, 7, 17, 6, 12}]
```

Out[130]=
$$-\frac{1}{9}$$

Rank Order (with and without Ties)

The use of RankOrder[ranking] is a way to shift from cardinal ranking as produced by EVM or GMM to the ordinal ranking. Hence, the result is a list of alternatives ordered according to the outrank relation $\prec$ introduced by the cardinal ranking. I.e. for two alternatives c_i and c_j it will holds that $c_i \prec c_j$ if $w(c_i) < w(c_j)$ and, of course $c_i \sim c_j$ if $w(c_i) = w(c_j)$. For example:

In[131]:=
```
ranking = EigenvalueRank[M]
```
Out[131]=
```
{0.119092, 0.27476, 0.356526, 0.130954, 0.118669}
```

In[132]:=
```
RankOrder[ranking]
```
Out[132]=
```
{3, 2, 4, 1, 5}
```

The above result means that the highest priority has 3rd alternative, then goes 2nd alternative, 4th, 1st and 5th at the end. It may happen, however, that some alternatives have the same priority. E.g.

In[133]:=
```
ranking = EigenvalueRank[{{1, 1, 1, 1, 3}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, {1, 1, 1, 1, 1}, {1/3, 1, 1, 1, 1}}]
```
Out[133]=
```
{0.256633, 0.194146, 0.194146, 0.194146, 0.16093}
```

In such a case RankOrderWithTies[] groups the alternatives with the similar priorities

In[134]:=
```
RankOrderWithTies[ranking]
```
Out[134]=
```
{1, {2, 3}, 4, 5}
```

The above result means that the alternatives 2, 3, and 4 took ex aequo the second position.

Sometimes it is useful to use ordinal ranking instead of cardinal one. In the ordinal ranking vector every alternative gets weight which is its position in the numerical ranking. To compute ordinal ranking from rank order we may use RankOrderToOrdinalRank

In[135]:=
```
RankOrderToOrdinalRank[{1, {2, 3, 4}, 5}]
```
Out[135]=
```
{1, 2, 2, 2, 3}
```

The same result we may get directly from the numerical ranking by using NumericalRankToOrdinalRank

In[136]:=
```
NumericalRankToOrdinalRank[ranking]
```
Out[136]=
```
{1, 2, 2, 3, 4}
```

The only inconvenience of the above method lies in the fact that the most important first alternative gets the lowest weight (first place). To remove this inconvenience we may want to assign to the alternatives their positions in the ranking in the reverse order. For this purpose we can use RankOrderToReverseOrdinalRank.

In[137]:=
```
RankOrderToReverseOrdinalRank[{1, {2, 3, 4}, 5}]
```
Out[137]=
```
{3, 2, 2, 2, 1}
```

and similarly

In[138]:=
```
NumericalRankToReverseOrdinalRank[ranking]
```
Out[138]=
```
{4, 3, 3, 2, 1}
```

Produces combination of cardinal and ordinal results together

In[139]:=
```
RankList[ranking]
```
Out[139]=
```
{{1, 0.256633}, {3, 0.194146}, {2, 0.194146}, {4, 0.194146}, {5, 0.16093}}
```

Obviously ordinal result can be obtained from rankList by

In[140]:=
```
First /@ RankList[ranking]
```
Out[140]=
```
{1, 3, 2, 4, 5}
```

and similarly cardinal result can be obtained from rankList by

In[141]:=
```
Last /@ RankList[ranking]
```
Out[141]=
```
{0.256633, 0.194146, 0.194146, 0.194146, 0.16093}
```

Montecarlo PC Matrices

Sometimes it is useful to generate random PC matrix to verify some properties of such matrix statistically
For example let us create some real random ranking of 7 alternatives with values from interval [1/15, 15]:

In[142]:=
```
ranking = RandomRankingPattern[7, 15]
```
Out[142]=
```
{0.335493, 5.41219, 7.1184, 0.6067, 10.566, 13.4996, 2.42948}
```

Then we may create fully consistent matrix based on this ranking by setting $c_{ij} = w_i / w_j$

In[143]:=
```
M = RandomMatrix[7, ranking, 1];
```

In[144]:=
```
M // MatrixForm
```

Out[144]//MatrixForm=

1.	0.0619885	0.0471304	0.552981	0.0317521	0.0248522	0.138093
16.132	1.	0.760309	8.9207	0.512225	0.400916	2.22772
21.2177	1.31525	1.	11.733	0.673706	0.527306	2.93002
1.80838	0.112099	0.0852297	1.	0.0574198	0.0449422	0.249724
31.494	1.95227	1.48433	17.4156	1.	0.782695	4.3491
40.2379	2.49429	1.89643	22.2508	1.27764	1.	5.55657
7.2415	0.44889	0.341295	4.00441	0.229933	0.179967	1.

In[145]:=
```
SaatyIdx[M]
```

Out[145]=
```
0
```

We may also disturb every entry by a factor chosen from the range [1/1.3, 1.3]. This makes the random matrix a bit inconsistent:

In[146]:=
```
M = RandomMatrix[7, ranking, 1.3];
```

In[147]:=
```
M // MatrixForm
```

Out[147]//MatrixForm=

1.	0.0509428	0.0393371	0.631832	0.0406248	0.0233109	0.149119
19.6298	1.	0.755062	8.12071	0.619257	0.505401	2.27839
25.4213	1.32439	1.	14.8069	0.832552	0.63383	2.66835
1.5827	0.123142	0.0675362	1.	0.0442987	0.0367438	0.217982
24.6155	1.61484	1.20113	22.574	1.	0.661504	4.13043
42.8985	1.97863	1.57771	27.2155	1.51171	1.	5.33342
6.70604	0.438906	0.374764	4.58753	0.242106	0.187497	1.

In[148]:=
```
SaatyIdx[M]
```

Out[148]=
```
0.00658761
```

In[149]:=
```
KoczkodajIdx[M]
```

Out[149]=
```
0.419083
```

One may want to consider disturbed rational matrix. To this end first we create rational ranking pattern of 7 alternatives unifirmly distributed over the scale 1/15 to 15:

In[150]:=
```
ranking = RandomRationalUniformRankingPattern[7, 15]
```

Out[150]=
$$\left\{10, 1, 11, \frac{1}{2}, \frac{1}{5}, 6, 14\right\}$$

Then we create fully consistent Rational PC matrix over this ranking

In[151]:=
```
M = RandomRationalMatrix[7, 15 * 15, ranking, 1];
```

In[152]:=
```
M // MatrixForm
```

Out[152]//MatrixForm=

$$\begin{pmatrix} 1 & 10 & \frac{10}{11} & 20 & 50 & \frac{5}{3} & \frac{5}{7} \\ \frac{1}{10} & 1 & \frac{1}{11} & 2 & 5 & \frac{1}{6} & \frac{1}{14} \\ \frac{11}{10} & 11 & 1 & 22 & 55 & \frac{11}{6} & \frac{11}{14} \\ \frac{1}{20} & \frac{1}{2} & \frac{1}{22} & 1 & \frac{5}{2} & \frac{1}{12} & \frac{1}{28} \\ \frac{1}{50} & \frac{1}{5} & \frac{1}{55} & \frac{2}{5} & 1 & \frac{1}{30} & \frac{1}{70} \\ \frac{3}{5} & 6 & \frac{6}{11} & 12 & 30 & 1 & \frac{3}{7} \\ \frac{7}{5} & 14 & \frac{14}{11} & 28 & 70 & \frac{7}{3} & 1 \end{pmatrix}$$

In[153]:=
```
SaatyIdx[M]
```

Out[153]=
```
0
```

In[154]:=
```
KoczkodajIdx[M]
```

Out[154]=
```
0
```

As previously we may also disturb every entry by the factor chosen from the range [1/1.3, 1.3], however, this time we have also to find the closest rational number within the set of numbers in the form $\frac{p}{q}$ where p,q belongs to $\{1,2,3,\ldots,15^2\}$. To get this matrix we have to call once again:

In[155]:=
```
M = RandomRationalMatrix[7, 15 * 15, ranking, 1.3];
```

In[156]:=
```
M // MatrixForm
```

Out[156]//MatrixForm=

$$\begin{pmatrix} 1 & \frac{145}{14} & \frac{23}{26} & 21 & 61 & \frac{175}{114} & \frac{134}{223} \\ \frac{14}{145} & 1 & \frac{9}{94} & \frac{197}{81} & \frac{36}{7} & \frac{13}{68} & \frac{17}{212} \\ \frac{26}{23} & \frac{94}{9} & 1 & \frac{149}{6} & \frac{191}{3} & \frac{25}{11} & \frac{109}{154} \\ \frac{1}{21} & \frac{81}{197} & \frac{6}{149} & 1 & \frac{194}{91} & \frac{4}{37} & \frac{7}{177} \\ \frac{1}{61} & \frac{7}{36} & \frac{3}{191} & \frac{91}{194} & 1 & \frac{3}{97} & \frac{1}{57} \\ \frac{114}{175} & \frac{68}{13} & \frac{11}{25} & \frac{37}{4} & \frac{97}{3} & 1 & \frac{62}{137} \\ \frac{223}{134} & \frac{212}{17} & \frac{154}{109} & \frac{177}{7} & 57 & \frac{137}{62} & 1 \end{pmatrix}$$

but of course this time there is some small inconsistency

In[157]:=
```
SaatyIdx[M]
```

Out[157]=
```
0.00754676
```

In[158]:=
```
N@KoczkodajIdx[M]
```
Out[158]=
```
0.438506
```

Both methods for generating random PC matrices has its short forms for default scales and patterns. They need only two parameters: size of the matrix and disturbance level

In[159]:=
```
M = RandomMatrix[5, 1.5];
```

In[160]:=
```
M // MatrixForm
```
Out[160]//MatrixForm=

1.	1.00337	0.314885	1.63372	1.0204
0.99664	1.	0.298187	1.38761	1.04596
3.17576	3.3536	1.	5.70895	3.09558
0.612101	0.720665	0.175164	1.	0.652309
0.980012	0.956058	0.323041	1.53301	1.

In[161]:=
```
M = RandomRationalMatrix[5, 1.5];
```

In[162]:=
```
M // MatrixForm
```
Out[162]//MatrixForm=

1	9	9	9	$\frac{1}{2}$
$\frac{1}{9}$	1	$\frac{9}{8}$	$\frac{3}{5}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{8}{9}$	1	$\frac{2}{3}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{5}{3}$	$\frac{3}{2}$	1	$\frac{1}{9}$
2	9	9	9	1

For example is we want to check what is the average inconsistency of the rational matrices 5 by 5 over the fundamental scale disturbed by the factor from [1/2, 2] we may just call:

In[163]:=
```
Mean@(SaatyIdx /@ Table[RandomRationalMatrix[5, 2], {100}])
```
Out[163]=
```
0.0632576
```

The above result was computed based on 100 randomly choosen rational random PC matrices.

The fully random rational matrix based on the rational scale can be obtained by RationalRandomScaledMatrix[size of a matrix, size of a scale]:

In[164]:=
```
M = RationalRandomScaledMatrix[7, 9];
```

In[165]:=
```
M // MatrixForm
```

Out[165]//MatrixForm=

$$\begin{pmatrix} 1 & \frac{1}{6} & \frac{1}{3} & 7 & \frac{1}{4} & 5 & 2 \\ 6 & 1 & 7 & 7 & \frac{1}{4} & 8 & \frac{1}{8} \\ 3 & \frac{1}{7} & 1 & \frac{1}{6} & 1 & 2 & \frac{1}{4} \\ \frac{1}{7} & \frac{1}{7} & 6 & 1 & \frac{1}{3} & 8 & \frac{1}{4} \\ 4 & 4 & 1 & 3 & 1 & 4 & 2 \\ \frac{1}{5} & \frac{1}{8} & \frac{1}{2} & \frac{1}{8} & \frac{1}{4} & 1 & \frac{1}{3} \\ \frac{1}{2} & 8 & 4 & 4 & \frac{1}{2} & 3 & 1 \end{pmatrix}$$

When we want to get randomly decompleted matrix based on the above then we call

In[166]:=
```
MD = RandomlyDecompletedMatrix[M, 7, □];
```

In[167]:=
```
MD // MatrixForm
```

Out[167]//MatrixForm=

$$\begin{pmatrix} 1 & \frac{1}{6} & \frac{1}{3} & □ & □ & □ & 2 \\ 6 & 1 & □ & □ & □ & 8 & \frac{1}{8} \\ 3 & □ & 1 & \frac{1}{6} & 1 & 2 & \frac{1}{4} \\ □ & □ & 6 & 1 & \frac{1}{3} & 8 & □ \\ □ & □ & 1 & 3 & 1 & 4 & 2 \\ □ & \frac{1}{8} & \frac{1}{2} & \frac{1}{8} & \frac{1}{4} & 1 & \frac{1}{3} \\ \frac{1}{2} & 8 & 4 & □ & \frac{1}{2} & 3 & 1 \end{pmatrix}$$

where the first paarameter is the original matrix, second - number of missing comparisons and the third is the missing comparisons symbol

Statistics

Wilcoxon test i.e. how to confirm that even for hardly disturbed PC matrix eigenvalue method and geometric mean method results in the statistically similar nummerical rankings....

In[168]:=
```
baseRanking = RandomRankingPattern[20, 15];
```

In[169]:=
```
M = RandomMatrix[20, baseRanking, 5];
```

In[170]:=
```
SaatyIdx@M
```

Out[170]=
```
0.399906
```

In[171]:=
```
evr = EigenvalueRank@M;
```

In[172]:=
```
gmr = GeometricRescaledRank@M;
```

In[173]:=
```
H = SignedRankTest[{evr, gmr}, Automatic, "HypothesisTestData"]
```

Out[173]=
```
HypothesisTestData[ Type: SignedRankTest
                    p−Value: 0.955 ]
```

In[174]:=
```
H["TestDataTable", "PValue", "TestConclusion"]
```

Out[174]=
```
{ |Statistic P-Value
  Signed-Rank|103.     0.955343 , 0.955343,

 The null hypothesis that the median difference is 0                    }
  is not rejected at the 5 percent level based on the Signed-Rank test.
```

As p-value is high we may expect that both evr and gmr are “statistically” similar.

Graph representation

Calculates the upper and lower triangularize of PC matrix. Missing elements are replaced by non-number symbol. Here it is ∞

In[175]:=
```
tmpM = SymbPCM[m, 5];
```

In[176]:=
```
tmpM // MatrixForm
```

Out[176]//MatrixForm=
$$\begin{pmatrix} 1 & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} \\ m_{2,1} & 1 & m_{2,3} & m_{2,4} & m_{2,5} \\ m_{3,1} & m_{3,2} & 1 & m_{3,4} & m_{3,5} \\ m_{4,1} & m_{4,2} & m_{4,3} & 1 & m_{4,5} \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & 1 \end{pmatrix}$$

In[177]:=
```
UpperTriangularizePCM@tmpM // MatrixForm
```

Out[177]//MatrixForm=
$$\begin{pmatrix} 1 & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} \\ \infty & 1 & m_{2,3} & m_{2,4} & m_{2,5} \\ \infty & \infty & 1 & m_{3,4} & m_{3,5} \\ \infty & \infty & \infty & 1 & m_{4,5} \\ \infty & \infty & \infty & \infty & 1 \end{pmatrix}$$

In[178]:=
```
LowerTriangularizePCM@tmpM // MatrixForm
```

Out[178]//MatrixForm=
$$\begin{pmatrix} 1 & \infty & \infty & \infty & \infty \\ m_{2,1} & 1 & \infty & \infty & \infty \\ m_{3,1} & m_{3,2} & 1 & \infty & \infty \\ m_{4,1} & m_{4,2} & m_{4,3} & 1 & \infty \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & 1 \end{pmatrix}$$

Let us prepare a ranking and a random PC matrix, which will be used for demonstration later on.

In[179]:=
```
ranking = RandomRationalUniformRankingPattern[5, 15]
```

Out[179]=
$$\left\{\frac{1}{9}, 11, \frac{1}{5}, 7, \frac{1}{14}\right\}$$

In[180]:=
```
M = RandomRationalMatrix[5, 15 * 15, ranking, 1.5];
```

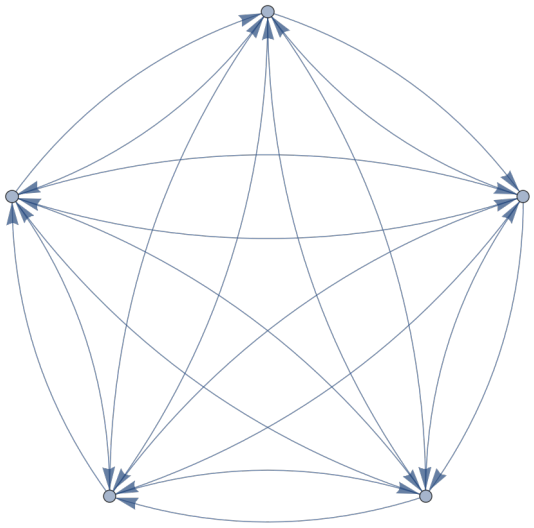
In[181]:=
```
M // MatrixForm
```

Out[181]//MatrixForm=
$$\begin{pmatrix} 1 & \frac{1}{107} & \frac{55}{128} & \frac{3}{164} & \frac{218}{119} \\ 107 & 1 & 57 & \frac{191}{138} & 174 \\ \frac{128}{55} & \frac{1}{57} & 1 & \frac{2}{103} & \frac{177}{55} \\ \frac{164}{3} & \frac{138}{191} & \frac{103}{2} & 1 & \frac{209}{2} \\ \frac{119}{218} & \frac{1}{174} & \frac{55}{177} & \frac{2}{209} & 1 \end{pmatrix}$$

We can compute a PC graph for the given PC matrix. Weights are not shown because we did not specify it directly

In[182]:=
```
g = GraphOfPCM@M
```

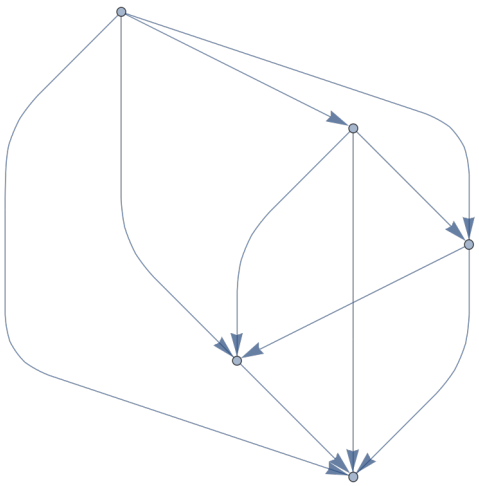
Out[182]=



Of course, we can choose to have a graph with only one edge between the pair of vertices

In[183]:=
```
GraphOfPCM@UpperTriangularizePCM@M
```

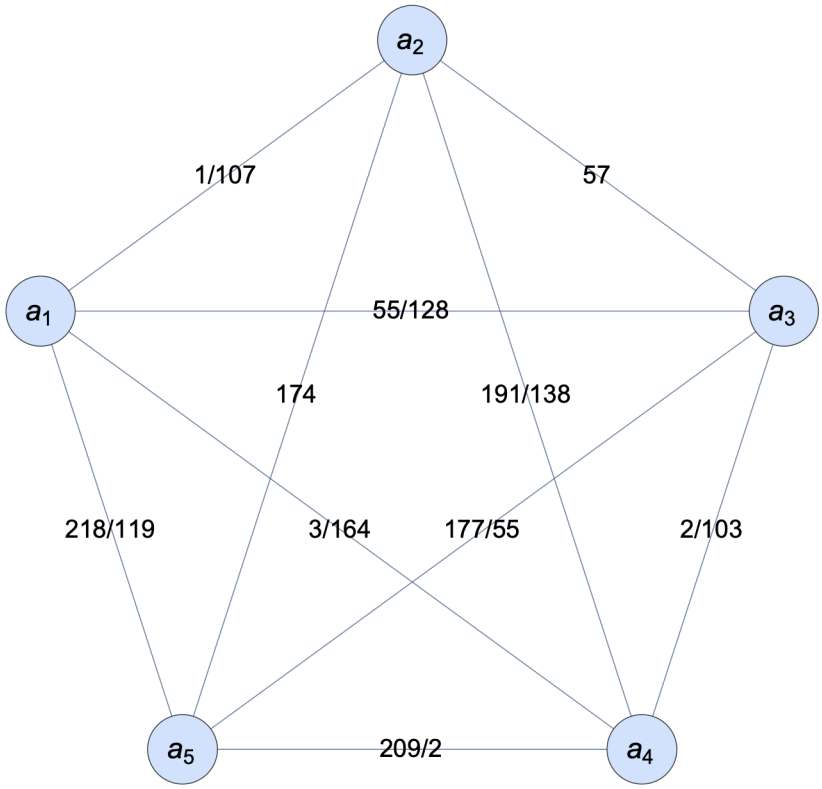
Out[183]=



For nice plotting we may use PlotPCGraph where first parametr is a M, second - flag whether the graph should be printed as directed of not, the third specifies the graph layout.

In[184]:=
```
PlotPCGraph[UpperTriangularizePCM@M, False,
 Automatic, NicePlotFraction, Function[i, aᵢ], 0.15, 13, 15]
```

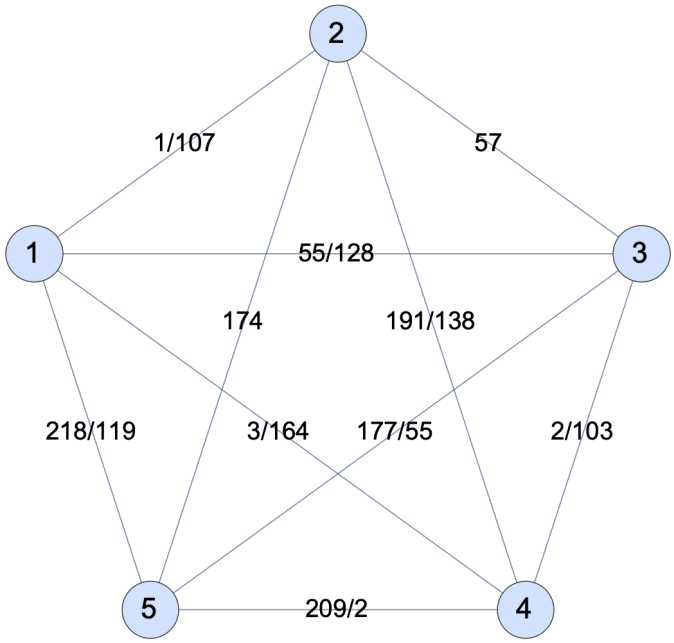
Out[184]=



Shorter form of a call is:

In[185]:=
```
PlotPCGraph[UpperTriangularizePCM@M]
```

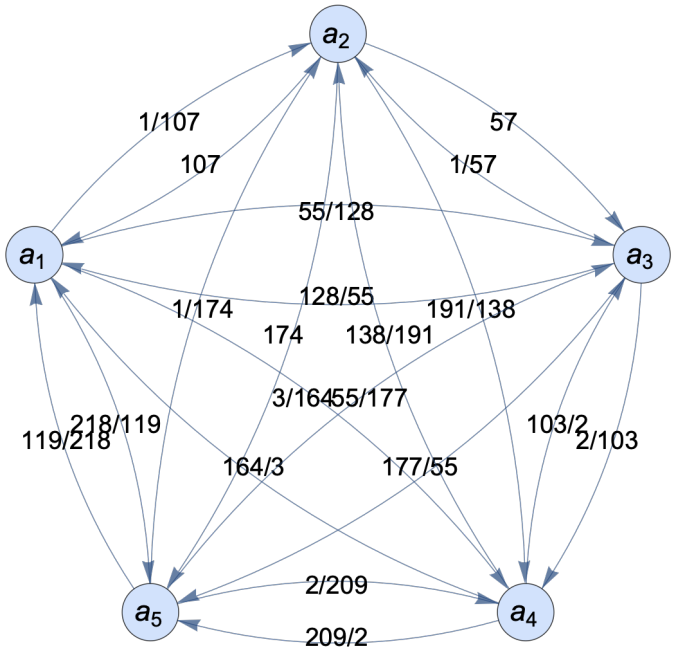
Out[185]=



We can also print out the whole directed graph

In[186]:=
```
PlotPCGraph[M, True, Automatic,
 NicePlotFraction, Function[i, aᵢ], 0.15, 13, 15]
```

Out[186]=



Let us compute all undirected spanning trees of the given PC matrix M

In[187]:=
```
st = FindAllSpanningTrees@UndirectedGraph@GraphOfPCM@UpperTriangularizePCM@M;
```

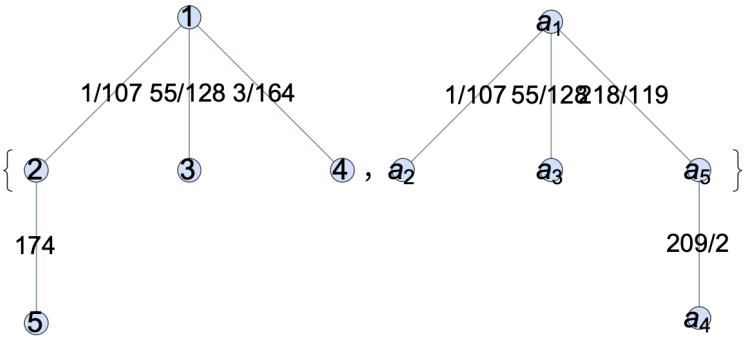
In[188]:=
```
Length@st
```
Out[188]=
```
125
```

Let us choose sample spanning trees

In[189]:=
```
sp1 = st[[2]];
```
In[190]:=
```
sp2 = st[[7]];
```

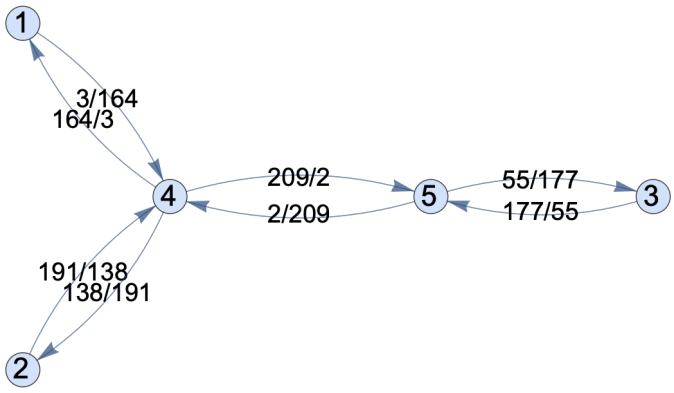
Let us plot spanning trees

In[191]:=
```
{PlotUndirectedWeightedGraph[sp1, M], PlotUndirectedWeightedGraph[sp2,
  M, Automatic, NicePlotFraction, Function[i, a_i], 0.15, 13, 15]}
```
Out[191]=



get random spanning tree from M

In[192]:=
```
rsp = GetRandomSpanningTreeFromPCMatrix[M];
```
In[193]:=
```
PlotUndirectedWeightedGraph[rsp, M]
```
Out[193]=



Every spanning tree induces some specific ranking so, let us compute these rankings

In[194]:=
```
r1 = RankBySpanningTree[sp1, M]
```
Out[194]=
$$\left\{1, 107, \frac{6313}{3190}, \frac{650\,239}{6380}, \frac{107}{174}\right\}$$

In[195]:=
```
N@RescaleRanking@r1
```
Out[195]=
```
{0.00470561, 0.5035, 0.00931239, 0.479588, 0.00289368}
```

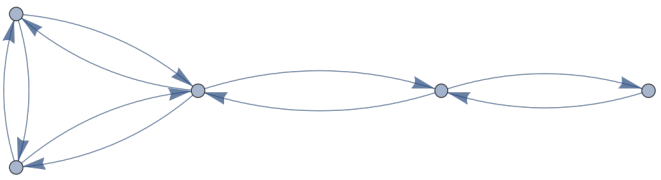
In[196]:=
```
r2 = RankBySpanningTree[sp2, M]
```
Out[196]=
$$\left\{1, 107, \frac{107}{57}, \frac{64\,735}{1062}, \frac{5885}{10\,089}\right\}$$

In[197]:=
```
N@RescaleRanking@r2
```
Out[197]=
```
{0.00583375, 0.624212, 0.0109511, 0.355601, 0.00340288}
```

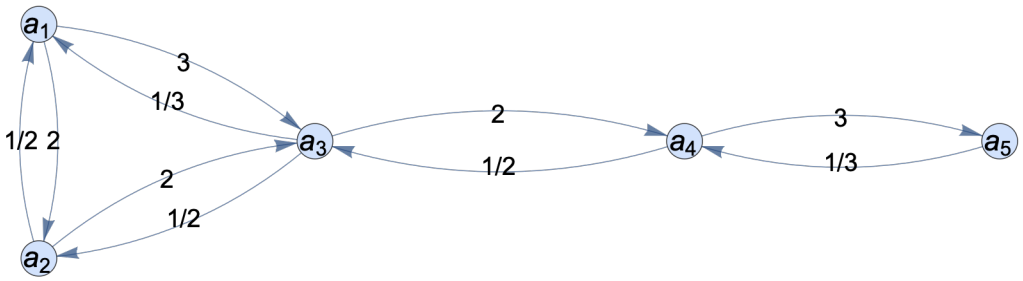
It works also for incomplete PC matrices

In[198]:=
$$M = \begin{pmatrix} 1 & 2 & 3 & \square & \square \\ 1/2 & 1 & 2 & \square & \square \\ 1/3 & 1/2 & 1 & 2 & \square \\ \square & \square & 1/2 & 1 & 3 \\ \square & \square & \square & 1/3 & 1 \end{pmatrix};$$

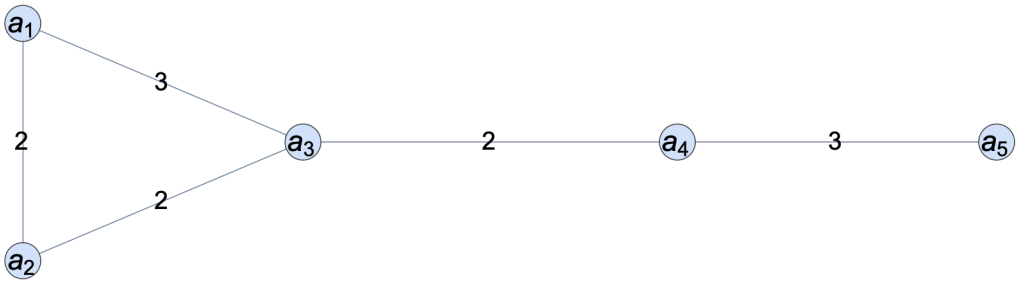
In[199]:=
```
g = GraphOfPCM@M
```
Out[199]=



In[200]:=
```
PlotPCGraph[M, True, Automatic,
 NicePlotFraction, Function[i, a_i], 0.15, 13, 15]
```
Out[200]=



In[201]:=
```
PlotPCGraph[M, False, Automatic,
 NicePlotFraction, Function[i, aᵢ], 0.15, 13, 15]
```
Out[201]=



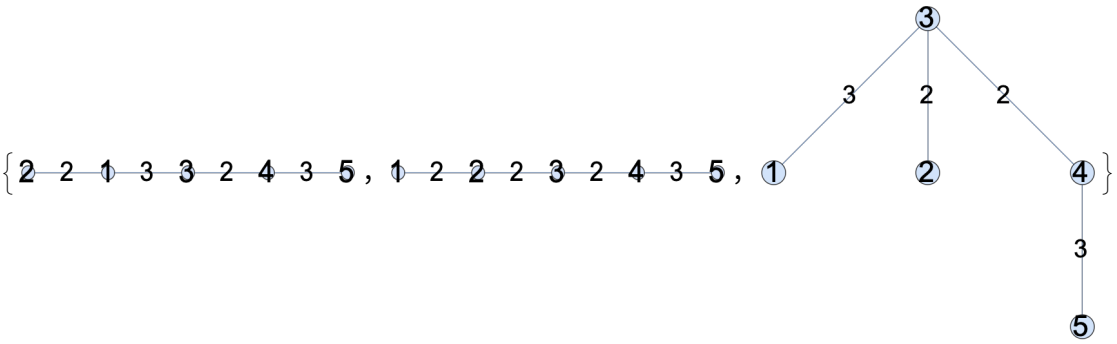
Get undirected graph of the given PC matrix M

In[202]:=
```
ug = UndirectedGraph@GraphOfPCM@UpperTriangularizePCM@M;
```
Get all spanning trees of M

In[203]:=
```
st = FindAllSpanningTrees@ug;
```
In[204]:=
```
Length@st
```
Out[204]=
```
3
```
And mark all the spanning trees in red

In[205]:=
```
Table[HighlightGraph[UndirectedGraph@g,
  st[[i]], GraphHighlightStyle → "Thick"], {i, 1, Length@st}]
```
Out[205]=

In[206]:=
```
Table[PlotUndirectedWeightedGraph[st[[i]], M], {i, 1, Length@st}]
```
Out[206]=



Layered cake matrix

Let us consider two Layered cake matrices:

In[207]:=
```
M1 = LCMatrix[5, 9];
```

In[208]:=
```
M1 // MatrixForm
```

Out[208]//MatrixForm=

$$\begin{pmatrix} 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 \\ 9 & 1 & 9 & 1 & 9 \\ 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 \\ 9 & 1 & 9 & 1 & 9 \\ 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 \end{pmatrix}$$

In[209]:=
```
M2 = LCMatrix[6, 9];
```

In[210]:=
```
M2 // MatrixForm
```

Out[210]//MatrixForm=

$$\begin{pmatrix} 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 & \frac{1}{9} \\ 9 & 1 & 9 & 1 & 9 & 1 \\ 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 & \frac{1}{9} \\ 9 & 1 & 9 & 1 & 9 & 1 \\ 1 & \frac{1}{9} & 1 & \frac{1}{9} & 1 & \frac{1}{9} \\ 9 & 1 & 9 & 1 & 9 & 1 \end{pmatrix}$$

And the ranking is:

In[211]:=
```
w1 = EigenvalueRank@M1
```

Out[211]=
```
{0.047619, 0.428571, 0.047619, 0.428571, 0.047619}
```

In[212]:=
```
w2 = EigenvalueRank@M2
```

Out[212]=
```
{0.0333333, 0.3, 0.0333333, 0.3, 0.0333333, 0.3}
```

In[213]:=
```
w1[[1]] / w1[[2]]
```

Out[213]=
```
0.111111
```

In[214]:=
```
w2[[1]] / w2[[2]]
```

Out[214]=
```
0.111111
```