```
gurobi_setup;
```

```
The MATLAB interface for Gurobi 10.0.1 has been installed.

The directory
    C:\gurobi1001\win64\matlab\
has been added to the MATLAB path.
To use Gurobi regularly, you must save this new path definition.
To do this, type the command
    savepath
at the MATLAB prompt. Please consult the MATLAB documentation
if necessary.
```

```
initCobraToolbox(false);
```

```
      _____   _____   _____   _____     _____     |
     /  ___| /  _  \ |  _  \ |  _  \   / ___ \    |   COnstraint-Based Reconstruction and Analysis
     | |     | | | | | |_| | | |_| |  | |___| |   |   The COBRA Toolbox - 2023
     | |     | | | | |  _  { |  _  /  |  ___  |   |
     | |___  | |_| | | |_| | | | \ \  | |   | |   |   Documentation:
     \_____| \_____/ |_____/ |_|  \_\ |_|   |_|   |   http://opencobra.github.io/cobratoolbox
                                                  |

 > Checking if git is installed ...  Done (version: 2.37.1).
 > Checking if the repository is tracked using git ...  Done.
 > Checking if curl is installed ...  Done.
 > Checking if remote can be reached ...  Done.
 > Initializing and updating submodules (this may take a while)... Done.
 > Adding all the files of The COBRA Toolbox ...  Done.
 > Define CB map output... set to svg.
 > TranslateSBML is installed and working properly.
 > Configuring solver environment variables ...
   - [----] ILOG_CPLEX_PATH: --> set this path manually after installing the solver ( see instructions )
   - [*---] GUROBI_PATH: C:\gurobi1001\win64\matlab
   - [----] TOMLAB_PATH: --> set this path manually after installing the solver ( see instructions )
   - [----] MOSEK_PATH: --> set this path manually after installing the solver ( see instructions )
   Done.
 > Checking available solvers and solver interfaces ...     0

     0

Check osense*c - A'*lam - w = 0 (stationarity):
     0
     0


 > [gurobi] Primal optimality condition in solveCobraLP satisfied.
 > [gurobi] Dual optimality condition in solveCobraLP satisfied.
Warning: Cplex is not on the MATLAB path. Complete the installation as specified here: https://
opencobra.github.io/cobratoolbox/stable/installation.html#ibm-ilog-cplex
changeCobraSolver: problem initialising CPLEX object: Undefined function 'Cplex' for input arguments of type 'struct
Could not find installation of ibm_cplex, so it cannot be tested
Could not find installation of tomlab_cplex, so it cannot be tested
Original LP has 1 row, 2 columns, 1 non-zero
Objective value = 0
OPTIMAL SOLUTION FOUND BY LP PRESOLVER

 > [glpk] Primal optimality condition in solveCobraLP satisfied.Could not find installation of mosek, so it cannot b
Could not find installation of matlab, so it cannot be tested

   --------------------------------------------------------
   pdco.m                      Version pdco5 of 15 Jun 2018
   Primal-dual barrier method to minimize a convex function
   subject to linear constraints Ax + r = b,  bl <= x <= bu

   Michael Saunders       SOL and ICME, Stanford University
   Contributors:     Byunggyoo Kim (SOL), Chris Maes (ICME)
                     Santiago Akle (ICME), Matt Zahr (ICME)
                     Aekaansh Verma (ME)
   --------------------------------------------------------

The objective is linear
The matrix A is an explicit sparse matrix

m        =        1     n        =        2      nnz(A)  =        1
max |b | =        0     max |x0| =  1.0e+00      xsize   =  1.0e+00
max |y0| =        1     max |z0| =  1.0e+00      zsize   =  1.0e+00

x0min    =        1     featol   =  1.0e-06      d1max   =  1.0e-04
z0min    =        1     opttol   =  1.0e-06      d2max   =  5.0e-04
mu0      =  1.0e-01     steptol  =     0.99     bigcenter=     1000

LSMR/MINRES:
atol1    =  1.0e-10     atol2    =  1.0e-15      btol    =  0.0e+00
conlim   =  1.0e+12     itnlim   =       10      show    =        0

Method   =        2     (1 or 11=chol  2 or 12=QR  3 or 13=LSMR  4 or 14=MINRES 21=SQD(LU)  22=SQD(MA57))
Eliminating dy before dx


Bounds:
  [0,inf]  [-inf,0]  Finite bl  Finite bu  Two bnds   Fixed    Free
        0        0          0          0         0       2       0
  [0, bu]  [bl,  0]  excluding fixed variables
        0        0

Itn   mu stepx stepz  Pinf  Dinf  Cinf   Objective    nf  center       QR
  0                   -6.6 -99.0  -Inf  1.2500000e-07        1.0
  1 -1.0 1.000 1.000 -99.0 -99.0  -Inf  0.0000000e+00  1     1.0        1
  2 -3.0 1.000 1.000 -99.0 -99.0  -Inf  0.0000000e+00  1     1.0
  3 -5.0 1.000 1.000 -99.0 -99.0  -Inf  0.0000000e+00  1     1.0
  4 -7.0 1.000 1.000 -99.0 -99.0  -Inf  0.0000000e+00  1     1.0
   Converged

max |x| =     0.000    max |y| =     0.000    max |z| =     0.000   scaled
max |x| =     0.000    max |y| =     0.000    max |z| =     0.000 unscaled
max |x| and max |z| exclude fixed variables
PDitns  =         4     QRitns =         0    cputime =       0.0

Distribution of vector     x         z
[      1,     10 )         0         2
[    0.1,      1 )         0         0
[   0.01,    0.1 )         0         0
[  0.001,   0.01 )         0         0
[ 0.0001,  0.001 )         0         0
[  1e-05, 0.0001 )         0         0
[  1e-06,  1e-05 )         0         0
[  1e-07,  1e-06 )         0         0
[  1e-08,  1e-07 )         0         0
[      0,  1e-08 )         2         0
Elapsed time is 0.092347 seconds.

 > [pdco] Primal optimality condition in solveCobraLP satisfied.
 > [pdco] Dual optimality condition in solveCobraLP satisfied.
Could not find installation of quadMinos, so it cannot be tested
Could not find installation of dqqMinos, so it cannot be tested
Could not find installation of cplex_direct, so it cannot be tested
```

```
Warning: Cplex is not on the MATLAB path. Complete the installation as specified here: https://
opencobra.github.io/cobratoolbox/stable/installation.html#ibm-ilog-cplex
changeCobraSolver: problem initialising CPLEX object: Undefined function 'Cplex' for input arguments of type 'struct
Could not find installation of cplexlp, so it cannot be tested
Could not find installation of tomlab_snopt, so it cannot be tested
 Done.
 > Setting default solvers ...Could not find installation of mosek, so it cannot be tested
Could not find installation of matlab, so it cannot be tested
 Done.
 > Saving the MATLAB path ... Done.
   - The MATLAB path was saved in the default location.

 > Summary of available solvers and solver interfaces

                    Support          LP      MILP        QP      MIQP        NLP        EP
    ------------------------------------------------------------------------------
    gurobi           active               1         1         1         1         -         -
    ibm_cplex        active               0         0         0         0         -         -
    tomlab_cplex     active               0         0         0         0         -         -
    glpk             active               1         1         -         -         -         -
    mosek            active               0         -         0         -         -         0
    matlab           active               0         -         -         -         0         -
    pdco             active               1         -         1         -         -         1
    quadMinos        active               0         -         -         -         -         -
    dqqMinos         active               0         -         0         -         -         -
    cplex_direct     active               0         0         0         -         -         -
    cplexlp          active               0         -         -         -         -         -
    qpng             passive              -         -         1         -         -         -
    tomlab_snopt     passive              -         -         -         -         0         -
    lp_solve         legacy               1         -         -         -         -         -
    ------------------------------------------------------------------------------
    Total            -                    4         2         3         1         0         1

 + Legend: - = not applicable, 0 = solver not compatible or not installed, 1 = solver installed.


 > You can solve LP problems using: 'gurobi' - 'glpk' - 'pdco'
 > You can solve MILP problems using: 'gurobi' - 'glpk'
 > You can solve QP problems using: 'gurobi' - 'pdco'
 > You can solve MIQP problems using: 'gurobi'
 > You can solve NLP problems using:
 > You can solve EP problems using: 'pdco'

> Checking for available updates ... skipped
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
removing: C:\Users\PC\AppData\Roaming\MathWorks\MATLAB Add-Ons\Collections\The COnstraint-Based Reconstruction and A
```

```
% setenv('GUROBI_PATH', 'C:\gurobi1001\win64\matlab')
% changeCobraSolver('gurobi', 'all')
```

Load the data

Load the transcriptomics data, the housekeeping genes obtained in the previous script, and the metabolic model (Human1 version).

```
% Transcriptomics data
data = readtable('Mod_data.xlsx');
```

```
% Housekeeping genes with the ensembl ids
h_k_g = readtable('housekeeping_ens.csv');
% Human1 metabolic model.
model = readCbModel('Human-GEM_Cobra_v1.01.mat');
```

```
Each model.subSystems{x} is a cell array, allowing more than one subSystem per reaction.
```

Metabolic genes

Pick the genes related to metabolism

In the transcriptomics dataset there are a lot of genes, but we are interested just in those ones that take part in the metabolism, so, we are going to compare the ensembl id of both, the transcriptomics dataset and the Human1 metabolic model, and pick those ones that are present.

```
% Create a vector just with the ensembl ids of the genes from the model
model_genes = model.genes
```

```
model_genes = 3068×1 cell
'ENSG00000000419'
'ENSG00000001036'
'ENSG00000001084'
'ENSG00000001630'
'ENSG00000002549'
'ENSG00000002587'
'ENSG00000002726'
'ENSG00000002746'
'ENSG00000003137'
'ENSG00000003987'
     ⋮
```

```
% Find indexes of the genes that are present in the metabolic model.
index_names = ismember(data.Ensembl_GeneID, model_genes);
% take the rows of the dataset that match the indexes obtained before, with all the
data
data_met = data(index_names, :)
```

```
data_met = 3033×49 table
```

...

	Ensembl_GeneID	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3
1	'ENSG00000000419'	24.2420	21.2693	24.6489	22.6069	23.4380	23.1285
2	'ENSG00000001036'	36.4484	34.3146	34.8401	40.4961	40.3793	40.6007
3	'ENSG00000001084'	1.5123	1.5729	1.7464	2.0652	2.2006	2.2952
4	'ENSG00000001630'	0.5885	0.4977	0.6929	0.6954	0.8602	0.7167
5	'ENSG00000002549'	13.4967	13.8094	14.0154	14.6136	15.2707	15.5160
6	'ENSG00000002587'	0.0045	0.0065	0.0088	0.0095	0.0030	0.0120
7	'ENSG00000002726'	0	0.0207	0.0063	0.0102	0	0
8	'ENSG00000002746'	0.4244	0.4569	0.4976	0.8658	0.8617	0.9534
9	'ENSG00000003137'	0.8409	0.6320	0.5645	6.3025	6.0255	5.6638

	Ensembl_GeneID	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3
10	'ENSG00000003987'	0.3639	0.4501	0.4426	0.3488	0.3388	0.3830
11	'ENSG00000003989'	0.0468	0.0267	0.0273	0.0197	0.0154	0.0412
12	'ENSG00000004455'	18.4989	17.6697	18.0190	15.6646	15.0455	15.5210
13	'ENSG00000004468'	0.0241	0.0086	0	0.0127	0.0079	0.0106
14	'ENSG00000004478'	7.7011	8.5748	8.2320	8.7527	8.5782	8.5574

⋮

StanDep

The first step is to check the if the model contain blocked reactions that can affect the interpretation.

```
%First, in all model extraction it is good to pre_process the model to
%ensure it does not contain blocked reactions

[fluxConsistentMetBool, fluxConsistentRxnBool, fluxInConsistentMetBool,
fluxInConsistentRxnBool, ~, fluxConsistModel] = findFluxConsistentSubset(model);
model = fluxConsistModel
```

```
model = struct with fields:
                          S: [7114×11831 double]
                       mets: {7114×1 cell}
                          b: [7114×1 double]
                     csense: [7114×1 char]
                       rxns: {11831×1 cell}
                         lb: [11831×1 double]
                         ub: [11831×1 double]
                          c: [11831×1 double]
                  osenseStr: 'max'
                      genes: {2495×1 cell}
                      rules: {11831×1 cell}
                  compNames: {9×1 cell}
                      comps: {9×1 cell}
                 metCharges: [7114×1 int64]
                metFormulas: {7114×1 cell}
                   metNames: {7114×1 cell}
             metInChIString: {7114×1 cell}
                    grRules: {11831×1 cell}
                 rxnGeneMat: [11831×2495 double]
        rxnConfidenceScores: [11831×1 double]
                   rxnNames: {11831×1 cell}
                   rxnNotes: {11831×1 cell}
               rxnECNumbers: {11831×1 cell}
              rxnReferences: {11831×1 cell}
                 subSystems: {11831×1 cell}
                description: 'Human-GEM_Cobra_v1.01.mat'
                    modelID: 'Human-GEM'
                    version: '1.13.0'
       fluxConsistentMetBool: [7114×1 logical]
       fluxConsistentRxnBool: [11831×1 logical]
     fluxInConsistentMetBool: [7114×1 logical]
     fluxInConsistentRxnBool: [11831×1 logical]
```

```
model
```

```
model = struct with fields:
                          S: [7114×11831 double]
                       mets: {7114×1 cell}
                          b: [7114×1 double]
                     csense: [7114×1 char]
                       rxns: {11831×1 cell}
                         lb: [11831×1 double]
                         ub: [11831×1 double]
                          c: [11831×1 double]
                  osenseStr: 'max'
                      genes: {2495×1 cell}
                      rules: {11831×1 cell}
                  compNames: {9×1 cell}
                      comps: {9×1 cell}
                 metCharges: [7114×1 int64]
                metFormulas: {7114×1 cell}
                   metNames: {7114×1 cell}
             metInChIString: {7114×1 cell}
                    grRules: {11831×1 cell}
                 rxnGeneMat: [11831×2495 double]
        rxnConfidenceScores: [11831×1 double]
                   rxnNames: {11831×1 cell}
                   rxnNotes: {11831×1 cell}
               rxnECNumbers: {11831×1 cell}
              rxnReferences: {11831×1 cell}
                 subSystems: {11831×1 cell}
                description: 'Human-GEM_Cobra_v1.01.mat'
                    modelID: 'Human-GEM'
                    version: '1.13.0'
      fluxConsistentMetBool: [7114×1 logical]
      fluxConsistentRxnBool: [11831×1 logical]
    fluxInConsistentMetBool: [7114×1 logical]
    fluxInConsistentRxnBool: [11831×1 logical]
```

Change the name of the column that contains the Ensembl_ID to gene, and normalize the data,

```
%Ensuring that the first colum is names 'gene'
data_met.Properties.VariableNames{1} = 'gene';
%Although this is not documented clearly, standep will perform a log10 on
%the expression data. To ensure that this does not introduce negative
%numbers we add 1 everywhere.
datalog10 = data_met
```

```
datalog10 = 3033×49 table
```

...

	gene	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3
1	'ENSG0000000...	24.2420	21.2693	24.6489	22.6069	23.4380	23.1285
2	'ENSG0000000...	36.4484	34.3146	34.8401	40.4961	40.3793	40.6007
3	'ENSG0000000...	1.5123	1.5729	1.7464	2.0652	2.2006	2.2952
4	'ENSG0000000...	0.5885	0.4977	0.6929	0.6954	0.8602	0.7167
5	'ENSG0000000...	13.4967	13.8094	14.0154	14.6136	15.2707	15.5160
6	'ENSG0000000...	0.0045	0.0065	0.0088	0.0095	0.0030	0.0120

	gene	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3
7	'ENSG0000000...	0	0.0207	0.0063	0.0102	0	0
8	'ENSG0000000...	0.4244	0.4569	0.4976	0.8658	0.8617	0.9534
9	'ENSG0000000...	0.8409	0.6320	0.5645	6.3025	6.0255	5.6638
10	'ENSG0000000...	0.3639	0.4501	0.4426	0.3488	0.3388	0.3830
11	'ENSG0000000...	0.0468	0.0267	0.0273	0.0197	0.0154	0.0412
12	'ENSG0000000...	18.4989	17.6697	18.0190	15.6646	15.0455	15.5210
13	'ENSG0000000...	0.0241	0.0086	0	0.0127	0.0079	0.0106
14	'ENSG0000000...	7.7011	8.5748	8.2320	8.7527	8.5782	8.5574

⋮

```
datalog10{:,2:end} = log10(datalog10{:,2:end}+1)
```

```
datalog10 = 3033×49 table
```

	gene	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3
1	'ENSG0000000...	1.4021	1.3477	1.4091	1.3730	1.3881	1.3825
2	'ENSG0000000...	1.5734	1.5480	1.5544	1.6180	1.6168	1.6191
3	'ENSG0000000...	0.4001	0.4104	0.4388	0.4865	0.5052	0.5179
4	'ENSG0000000...	0.2010	0.1754	0.2286	0.2293	0.2696	0.2347
5	'ENSG0000000...	1.1613	1.1705	1.1765	1.1935	1.2114	1.2179
6	'ENSG0000000...	0.0020	0.0028	0.0038	0.0041	0.0013	0.0052
7	'ENSG0000000...	0	0.0089	0.0027	0.0044	0	0
8	'ENSG0000000...	0.1536	0.1634	0.1754	0.2709	0.2699	0.2908
9	'ENSG0000000...	0.2650	0.2127	0.1944	0.8635	0.8467	0.8237
10	'ENSG0000000...	0.1348	0.1614	0.1591	0.1299	0.1267	0.1408
11	'ENSG0000000...	0.0199	0.0115	0.0117	0.0085	0.0067	0.0175
12	'ENSG0000000...	1.2900	1.2711	1.2792	1.2218	1.2054	1.2180
13	'ENSG0000000...	0.0103	0.0037	0	0.0055	0.0034	0.0046
14	'ENSG0000000...	0.9396	0.9811	0.9653	0.9891	0.9813	0.9803

⋮

Define the limits of the bins:

```
%Standep requires us to define the bin boundaries. IMPORTANTLY these
%boundaries have to be in a log scale...
maxlog10 = max(max(table2array(datalog10(:,2:end))));
edgeX = linspace(0,maxlog10,11); %If we want 10 bins we need 11
edgeX = round(edgeX,1); %These are our bin bounds!
```

Pre-processing of the data for Standep

```
%Standep requires a very specific pre-processing for the inputs!
%1st, the RNA data structure
rnaData = struct();
rnaData.gene = data_met.gene;
rnaData.value = table2array(data_met(:,2:end));
rnaData.valuebyTissue = table2array(data_met(:,2:end));
rnaData.Tissue = data.Properties.VariableNames(2:end)';
```

Pre-processing of the metabolic model

```
%2nd the model data structure
modelData = getModelData(rnaData,model);
```

Pre-processing of the enzyme data

```
%3rd the enzyme data structure
spec = getSpecialistEnzymes(model);
prom = getPromEnzymes(model);
enzymeData = comparePromiscuousSpecific(spec,prom,modelData);
```

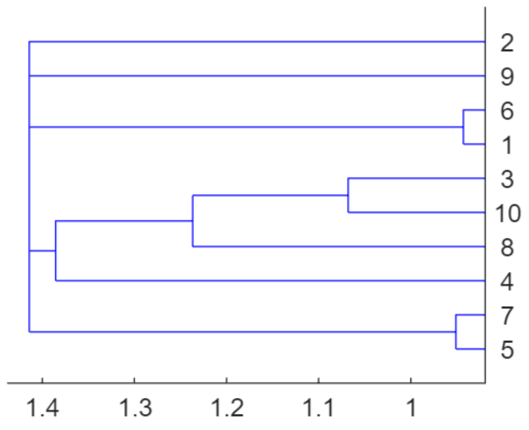
Set the parameters for the clustering analysis

```
% We then set our parameters...
distMethod = 'euclidean'; % distance method  maybe think one more robust than
euclidean.
linkageMethod = 'complete'; % linkage metric for hierarchical clustering
k = 10;
```

Clustering analysis

```
close all %This is important to ensure that all standep plots have the right number
to be saved if we wish
clustObj = geneExprDist_hierarchy(enzymeData,[],edgeX,k,distMethod,linkageMethod);
```

```
Cophenetic correlation coeffcient using complete linkage and euclidean distance = 0.8784
```



Identify core reactions and calculate ubiquity score?

```
[coreRxnMat,enzTis,cutOff,thr] =
models4mClusters1(clustObj,enzymeData.Tissue,model,edgeX,[],[],false,0,[1 1]);
%CoreRxnMat defines the core set of reactions in a general sense
```

```
Top 25th percentile for the data = 1.0211
Mean of Data = 0.1716
Std. Dev. of Data = 0.2575
fraction selected = 0.0002
fraction selected = 0.0001
fraction selected = 0.0000
fraction selected = 1.0000
fraction selected = 1.0000
fraction selected = 1.0000
fraction selected = 1.0000
fraction selected = 1.0000
fraction selected = 1.0000
fraction selected = 1.0000
```

```
[ubiScore,uScore] = getUbiquityScore_2022(clustObj,edgeX,model); %ubiScore is the
ubiquity score per rxns to use in mCadre!
```

```
Top 25th percentile for the data = 1.0211
Mean of Data = 0.1716
Std. Dev. of Data = 0.2575
coreRxnTable = 11831×48 table
```

...

	coreRxnMat1	coreRxnMat2	coreRxnMat3	coreRxnMat4	coreRxnMat5
1 MAR03905	1	1	1	1	1
2 MAR03907	0	0	0	0	0
3 MAR04097	1	1	1	1	1
4 MAR04099	0	0	0	0	0
5 MAR04108	1	1	1	1	1
6 MAR04133	1	1	1	1	1
7 MAR04137	1	1	1	1	1
8 MAR04281	1	1	1	1	1
9 MAR04388	1	1	1	1	1
10 MAR04283	1	1	1	1	1
11 MAR08357	1	1	1	1	1
12 MAR04379	1	1	1	1	1
13 MAR04301	1	1	1	1	1
14 MAR04355	1	1	1	1	1

⋮

```
hkg_met_ens = 1158×1 table
```

	gene
1	'ENSG0000000...
2	'ENSG0000000...
3	'ENSG0000000...
4	'ENSG0000000...
5	'ENSG0000000...
6	'ENSG0000000...
7	'ENSG0000000...
8	'ENSG0000000...
9	'ENSG0000000...
10	'ENSG0000000...
11	'ENSG0000000...
12	'ENSG0000000...
13	'ENSG0000000...
14	'ENSG0000000...

⋮

Thanks to StanDep, a matrix has been generated with the reactions that are considered core (1) and non-core (0), however, no solution has been found to obtain from the list of housekeeping genes, the list of reactions that they encode, since as we know, many genes come into play, and some could be considered as non housekeeping. At the same time, an attempt was also made to convert reactions into genes, in order to be able to analyse them, but genes that were not known to exist were found. Therefore, no progress was made in this field, but it is hoped that a solution will be found in the future.