LocalT2 Thresholding Method

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;
% 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, :);
```

Select just the transcriptomics data, and normalize it, with log10, and adding a 1, to avoid having -inf values

```
% Data just with transcriptomics data, not explanatory columns
data_f = data_met(:, 2:end);
% Normalize the data
logdata = log10(data_f + 1);
```

Use the localT2_core function. By default the 25th percentile is selected as the lower and the 75th percentile is selected as the upper percentile to calculate thresholds.

```
% Use function localT2 with 25 ant 75 percentiles as thresholds
up_percentage =75;
low_percentage =25;
% Use the localT2 function.
coreMat = localT2_core(logdata, low_percentage, up_percentage);
```

Find the core genes of the metabolism with the results of the localT2 function

```
core_indices = find(mean(coreMat, 2) > 0.5);
gene_names = data_met{:, 1};
core_genes = gene_names(core_indices);
core_genes_tab = table(core_genes);
```

Housekeeping genes

Now that we found the core genes, we have to explore the housekeeping genes. First of all is determine which of all the housekeeping genes in the list have metabolic functions. For that, we get the Ensembl id of the genes related to metabolism, and with that we can compare which ones are present

```
% Select the ensembl id of the core genes
genes_table = data_met.Ensembl_GeneID;

% See how many housekeeping genes in the list have metabolic functions.
index_names = ismember(genes_table, h_k_g.converted_alias);
hkg_met = data_met(index_names, :);

% Now just the ensembl id
hkg_met_ens = hkg_met(:, "Ensembl_GeneID");
```

We have the ensembl id of the housekeeping genes with metabolic function, now is time to check which of those are considered core genes by general grouping of the dataset.

```
correctly_identified = intersect(core_genes_tab.core_genes,
hkg_met_ens.Ensembl_GeneID);
hkg_met_ens = table2array(hkg_met(:, 'Ensembl_GeneID'));
percentage_correct = (length(correctly_identified) / length(hkg_met_ens)) * 100;
disp(['Correctly identified housekeeping genes: ', num2str(percentage_correct),
'%']);
```

```
Correctly identified housekeeping genes: 63.9896%
```

```
% Save the correctly_identified to have the genes identifiers
writecell(correctly_identified, 'HK_core_T2_FPKM.csv');
```

Around 64% of the metabolic housekeeping genes are considered core genes

Coverage of metabolic Housekeeping genes for each sample

```
% Uses logical indexing to select corresponding rows
sel_rows = coreMat(index_names == 1, :);

% Count the number of ones in each column
num_ones = sum(sel_rows);

% Calculates the total number of rows
num_rows = size(sel_rows, 1);

% Calculate the proportion of ones in each column
prop_genes = num_ones / num_rows;
```

```
% Create a table with the results
expression_col = data_met.Properties.VariableNames(2:end);
results_col_FPKM = array2table(prop_genes, 'VariableNames', expression_col)
```

```
results_col_FPKM = 1×48 table
```

...

	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3	IMR90_Y1
1	0.7098	0.7012	0.7124	0.6399	0.6736	0.6770	0.7496

Here you can see a table with the accuracy for this model with the FPKM data for each sample.

Convert the data from FPKM to TPM

Load the data again

```
% 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.
```

Convert FPKM to TPM

It was decided to explore new alternatives for the normalisation of gene expression data by switching from FPKM (Fragments Per Kilobase Million) to TPM (Transcripts Per Million), which has been shown in other studies to facilitate comparison between different samples.

```
% Extract matrix from table
data_matrix = data{:, 2:end};

% Calculate the sum of each column
column_sums = sum(data_matrix, 1);

% Normalize each element by the sum of its column and then multiply by 10^6
normalized_matrix = (data_matrix ./ column_sums) * 1e6;

% Convert the normalized matrix back to a table
normalized_table = array2table(normalized_matrix, 'VariableNames',
data.Properties.VariableNames(2:end));

% Replace the relevant columns in the original table
data(:, 2:end) = normalized_table;
```

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;
% 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, :);
```

Select just the transcriptomics data, and normalize it, with log10, and adding a 1, to avoid having -inf values

```
% Data just with transcriptomics data, not explanatory columns
data_f = data_met(:, 2:end);
% Normalize the data
logdata = log10(data_f + 1);
```

Use the localT2_core function. By default the 25th percentile is selected as the lower and the 75th percentile is selected as the upper percentile to calculate thresholds.

```
% Use function localT2 with 25 ant 75 percentiles as thresholds
up_percentage =75;
low_percentage =25;
% Use the localT2 function.
coreMat = localT2_core(logdata, low_percentage, up_percentage);
```

Find the core genes of the metabolism with the results of the localT2 function

```
core_indices = find(mean(coreMat, 2) > 0.5);
gene_names = data_met{:, 1};
core_genes = gene_names(core_indices);
core_genes_tab = table(core_genes);
```

Housekeeping genes

Now that we found the core genes, we have to explore the housekeeping genes. First of all is determine which of all the housekeeping genes in the list have metabolic functions. For that, we get the Ensembl id of the genes related to metabolism, and with that we can compare which ones are present

```
% Select the ensembl id of the core genes
genes_table = data_met.Ensembl_GeneID;

% See how many housekeeping genes in the list have metabolic functions.
index_names = ismember(genes_table, h_k_g.converted_alias);
hkg_met = data_met(index_names, :);
```

```
% Now just the ensembl id
hkg_met_ens = hkg_met(:, "Ensembl_GeneID");
```

We have the ensembl id of the housekeeping genes with metabolic function, now is time to check which of those are considered core genes by general grouping of the dataset..

```
correctly_identified = intersect(core_genes_tab.core_genes,
hkg_met_ens.Ensembl_GeneID);
hkg_met_ens = table2array(hkg_met(:, 'Ensembl_GeneID'));
percentage_correct = (length(correctly_identified) / length(hkg_met_ens)) * 100;
disp(['Correctly identified housekeeping genes: ', num2str(percentage_correct),
'%']);
```

```
Correctly identified housekeeping genes: 71.0708%
```

```
% Save the correctly_identified to have the genes identifiers
writecell(correctly_identified, 'HK_core_T2_TPM.csv');
```

Around 71% of the metabolic housekeeping genes are considered core gene

Coverage of metabolic Housekeeping genes for each sample

```
% Uses logical indexing to select corresponding rows
sel_rows = coreMat(index_names == 1, :);

% Count the number of ones in each column
num_ones = sum(sel_rows);

% Calculates the total number of rows
num_rows = size(sel_rows, 1);

% Calculate the proportion of ones in each column
prop_genes = num_ones / num_rows;

% Create a table with the results
expression_col = data_met.Properties.VariableNames(2:end);
results_col_TPM = array2table(prop_genes, 'VariableNames', expression_col)
```

```
results_col_TPM = 1×48 table
```

...

	BJ_Y1	BJ_Y2	BJ_Y3	BJ_OLD_1	BJ_OLD_2	BJ_OLD_3	IMR90_Y1
1	0.7634	0.7599	0.7746	0.6477	0.6857	0.6978	0.8031

Here you can see a table with the accuracy for this model with the TPM data for each sample.