# Localgini 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);
```

Calculate the Gini coefficient, using the localgini function.

```
gini = localgini_core(logdata);
```

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

```
core_indices = find(mean(gini, 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 ensmbl 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: 60.7945%
```

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

Around 61% 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 = gini(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 | 0.6131 | 0.6166 | 0.6209 | 0.5984 | 0.6088 | 0.6028 | 0.6261 |

...

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);
```

Calculate the Gini coefficient, using the localgini function.

```
gini = localgini_core(logdata);
```

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

```
core_indices = find(mean(gini, 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 ensmbl 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: 94.4732%
```

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

Around 94% of the metabolic housekeeping genes are considered core genes. This result is surprising, as an improvement was expected, but not at this level. To check if this method is better than the previous ones, we intend to analyse this same pipeline for different datasets, and see how it performs.

### Coverage of metabolic Housekeeping genes for each sample

```
% Uses logical indexing to select corresponding rows
sel_rows = gini(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 | 0.9404 | 0.9413 | 0.9404 | 0.9292 | 0.9335 | 0.9352 | 0.9387 |

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

## Comparing the core housekeeping genes of local thresholding and localgini

The overall core housekeeping genes of the two methods are compared to see how much they match each other.

```
% Load LocalT2 gene identifiers for FPKM and TPM
core_hk_Local_FPKM = readtable('HK_core_T2_FPKM.csv');
core_hk_Local_TPM = readtable('HK_core_T2_TPM.csv');
% Load Localgini gene identifiers for FPKM and TPM
core_hk_gini_FPKM = readtable('HK_core_gini_FPKM.csv');
core_hk_gini_TPM = readtable('HK_core_gini_TPM.csv');
```

Find the common identifiers between the two, localT2 and localgini

```
%%For FPKM
% Find the gene names that are in both datasets
common_genes = intersect(core_hk_Local_FPKM, core_hk_gini_FPKM);

% Find the rows in each dataset corresponding to the common genes
rows_data_T2 = ismember(core_hk_Local_FPKM, common_genes);
rows_data_gini = ismember(core_hk_gini_FPKM, common_genes);

% Extract the data for the common genes
common_data1 = core_hk_Local_FPKM(rows_data_T2, :);
common_data2 = core_hk_gini_FPKM(rows_data_gini, :);

% Display the number of common genes
disp(['Number of common genes: ', num2str(height(common_data1))]);
```

```
Number of common genes: 576
```

A total of 592 core housekeeping genes are presented that match within a total of approximately 700.

```
%%For TPM
% Find the gene names that are in both datasets
common_genes = intersect(core_hk_Local_TPM, core_hk_gini_TPM);

% Find the rows in each dataset corresponding to the common genes
rows_data_T2 = ismember(core_hk_Local_TPM, common_genes);
rows_data_gini = ismember(core_hk_gini_TPM, common_genes);

% Extract the data for the common genes
common_data1 = core_hk_Local_TPM(rows_data_T2, :);
common_data2 = core_hk_gini_TPM(rows_data_gini, :);

% Display the number of common genes
disp(['Number of common genes: ', num2str(height(common_data1))]);
```

```
Number of common genes: 797
```

The common number of genes is 797 out of about 822, which represents a high accuracy, however, these results may be due to the large number of core housekeeping genes seen to be produced by the localgini thresholding method.