Supplementary File 1

Selcen Ari

7 Nisan 2019

Defined functions for ceRNA models and workflow of method

We defined the functions that can be used with R programming. Briefly, these functions process a given miRNA:gene dataset and convert to graph object. All values that are significant in miRNA:target interactions are stored in edge variables and processed with formulations that are given in previous section. The functions and steps of approach are explained as following (Figure 1) :

Convertion of dataset: priming_graph function processes the given dataset that includes competing elements in first variable and repressive element in second variable. If the affinity and/or degradation factors are specified in the function, factors are taken into account, are processed with defaults in vice versa. The formulations that are given in equations (1-4) are performed in this function. This step gives the graph object which contains efficiency values of miRNA:competing target pairs in steady-state in terms of amount. It is assumed that the initial target amounts in the dataset is observed after the reppressive activity of miRNAs in steady-state.

Transition of variables in graph: In the previous step, the calculations are performed in the edge variables of the graph object. However, the graph object allows to use node variables, while the node features are handled to the graph. In this direction, update_nodes function carries the amount values to node variables This step must be applied with "once" option because it is primary process.

Trigger change in graph: The dataset are assumed as steady-state in previous step and the efficieny coefficients are calculated according to this acceptance. In the network that is found in steady-state conditions, the change is applied to the graph object for disturbtion of steady-state. To provide the disturbtion in the network the workflow offer two methods: update_variables and update_how. The first, a new dataset that is contained competing and repressive element names and current values of these can be processed with update_variables. The second option, the amount of the given node name in update_how function can be changed according to "how" argument.

Updating current values of variables: After variables updating in edge varibles, these are carried to node variables. Current and previous values of variables are stored as node variables with update_variables function.

Simulation of competing behavior of targets: After the change in the steady-state conditions, the network elements try to gain steady-state again. This process progresses as repeating of regulations after the spreading the changes in the network. In this step, simulation of regulations according to given cycle count in simulate function is applied. After each simulation cycle, the miRNA repression values are re-calculated and the current values of competing elements are found and saved. The process is performed in the edge data and at the same time outputs of the calculations are carried from edge to node data.

The node elements in the dataset are handled as two type; repressives (miRNAs) and competings (targets). It is assumed in approach that while targets are degrading or inhibiting by miRNAs continuously, miRNAs reversibly used. If the trigger of the network is a miRNA, it maintains the current value of amount that provides by user. On the contrary, it tries to help this process to provide steady-state through the regulations on its amount, if a competing element is used as a trigger. The functions that are used in the approach are developed with R programming so as can be used with other packages. These are can be found in the github repository ceRNAnetsim github page and improved with contributions of others.

```r
#install.packages("devtools")
#devtools::install_github("selcenari/ceRNAnetsim")
library(ceRNAnetsim)
```

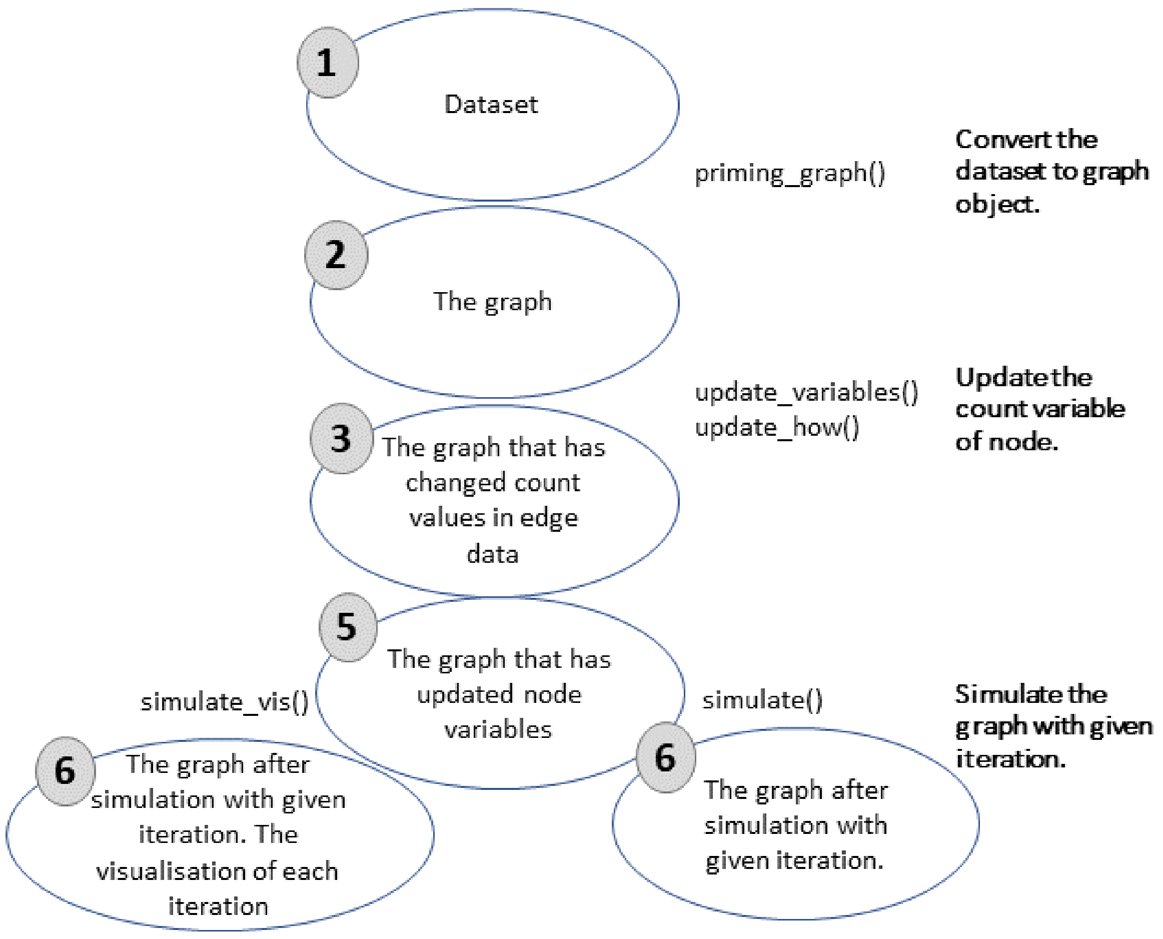

1
Dataset
priming_graph()
Convert the dataset to graph object.
2
The graph
update_variables()
update_how()
Update the count variable of node.
3
The graph that has changed count values in edge data
5
The graph that has updated node variables
simulate_vis()
simulate()
Simulate the graph with given iteration.
6
The graph after simulation with given iteration. The visualisation of each iteration
6
The graph after simulation with given iteration.

Figure 1: Workflow for simulation of competing endogenous RNA regulations. Graph object in steps 2-6 is saved and updated continuously.

- load *minsamp* data

```
data("minsamp")

minsamp
```

```
##   competing miRNA Competing_expression miRNA_expression seed_type region
## 1     Gene1  Mir1                10000             1000      0.43   0.30
## 2     Gene2  Mir1                10000             1000      0.43   0.01
## 3     Gene3  Mir1                 5000             1000      0.32   0.40
## 4     Gene4  Mir1                10000             1000      0.23   0.50
## 5     Gene4  Mir2                10000             2000      0.35   0.90
## 6     Gene5  Mir2                 5000             2000      0.05   0.40
## 7     Gene6  Mir2                10000             2000      0.01   0.80
##   energy
## 1    -20
## 2    -15
## 3    -14
## 4    -10
## 5    -12
## 6    -11
## 7    -25
```

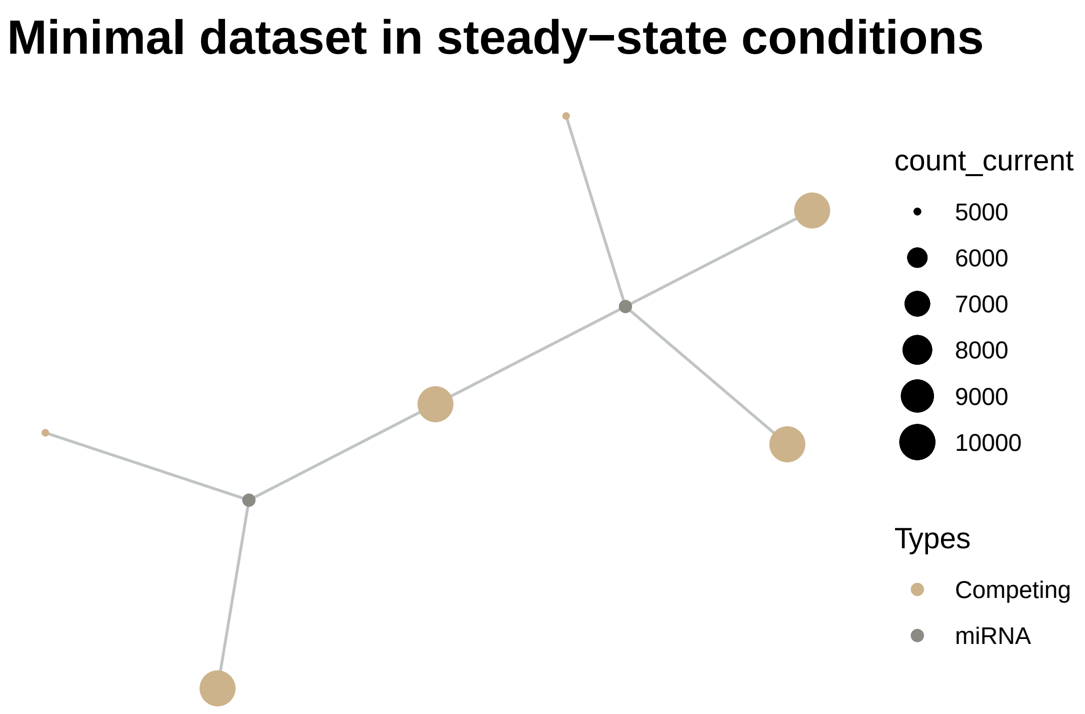
competing	miRNA	Competing_expression	miRNA_expression	seed_type	region	energy
Gene1	Mir1	10000	1000	0.43	0.30	-20
Gene2	Mir1	10000	1000	0.43	0.01	-15
Gene3	Mir1	5000	1000	0.32	0.40	-14
Gene4	Mir1	10000	1000	0.23	0.50	-10
Gene4	Mir2	10000	2000	0.35	0.90	-12
Gene5	Mir2	5000	2000	0.05	0.40	-11
Gene6	Mir2	10000	2000	0.01	0.80	-25

minsamp dataset analysis in lack of interaction factors.

Firstly, we have analysed minimal data without interaction factors between miRNA:target.

- 1. We have evaluated graph in the steady state conditions as followings:

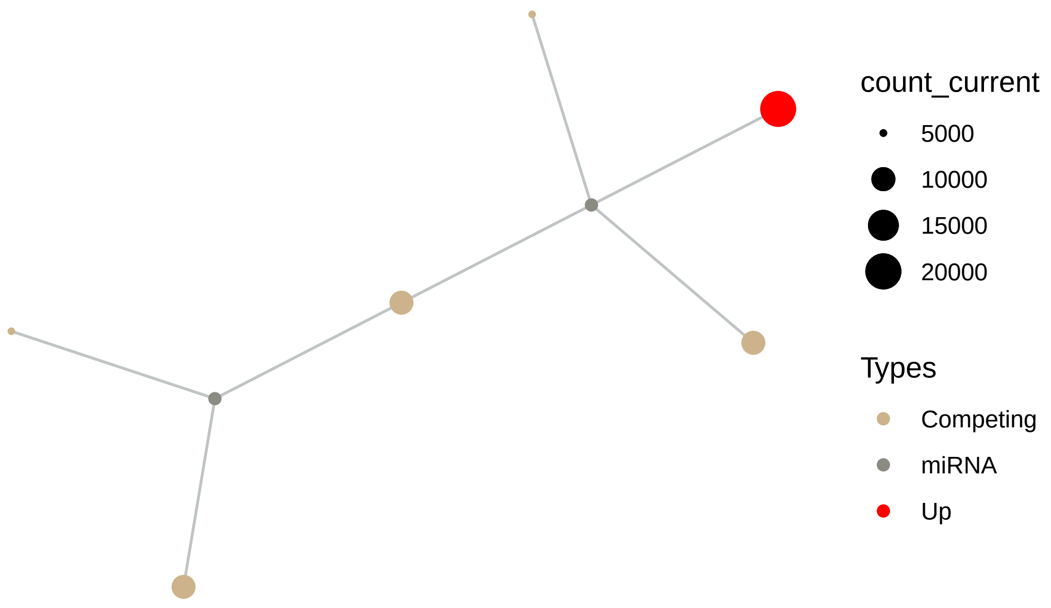
```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression)%>%
  vis_graph(Competing_color = "navajowhite3", mirna_color = "ivory4", title = "Minimal dataset in stead
```



- 2. We have obtained graph after change on Gene2 expression as followings:

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression)%>%
  update_how("Gene2", 2)%>%
  vis_graph(Competing_color = "navajowhite3", mirna_color = "ivory4", Upregulation = "red", title = "Ge
```

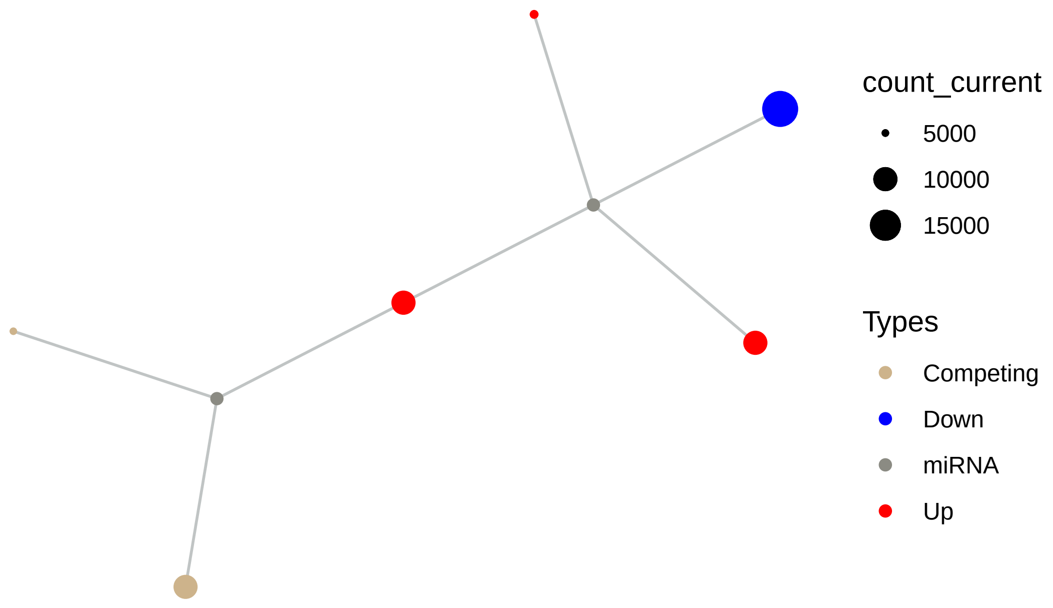
Gene2 Upregulation without interaction factors



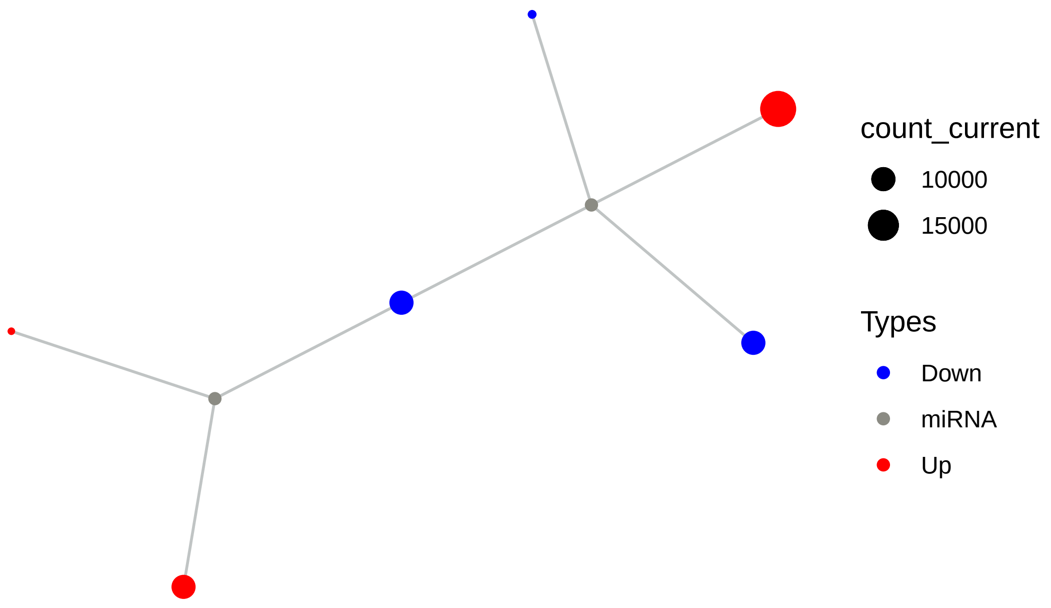
- 3. We have determined regulations after Gene2 Upregulation:

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression)%>%
  update_how("Gene2", 2)%>%
  simulate_vis(Competing_color = "navajowhite3", mirna_color = "ivory4", Upregulation = "red", Downregul
```

Regulations after Gene2 Upregulation – 1



Regulations after Gene2 Upregulation − 2



```
## # A tbl_graph: 8 nodes and 7 edges
## #
## # A rooted tree
## #
## # Node Data: 8 x 7 (active)
##   name  type  node_id initial_count count_pre count_current
##   <chr> <chr>   <int>         <dbl>     <dbl>         <dbl>
## 1 Gene1 Comp~       1         10000    10063.        10062.
## 2 Gene2 Comp~       2         10000    19841.        19845.
## 3 Gene3 Comp~       3          5000     5032.         5031.
## 4 Gene4 Comp~       4         10000    10063.        10059.
## 5 Gene5 Comp~       5          5000     5000          5001.
## 6 Gene6 Comp~       6         10000    10000         10002.
## # ... with 2 more rows, and 1 more variable: changes_variable <chr>
## #
## # Edge Data: 7 x 20
##    from    to Competing_name miRNA_name Competing_expre~ miRNA_expression
##   <int> <int> <chr>          <chr>                 <dbl>            <dbl>
## 1     1     7 Gene1          Mir1                  10000             1000
## 2     2     7 Gene2          Mir1                  10000             1000
## 3     3     7 Gene3          Mir1                   5000             1000
## # ... with 4 more rows, and 14 more variables: dummy <dbl>,
## #   afff_factor <dbl>, degg_factor <dbl>, comp_count_list <list>,
## #   comp_count_pre <dbl>, comp_count_current <dbl>,
## #   mirna_count_list <list>, mirna_count_pre <dbl>,
## #   mirna_count_current <dbl>, mirna_count_per_dep <dbl>,
## #   effect_current <dbl>, effect_pre <dbl>, effect_list <list>,
## #   mirna_count_per_comp <dbl>
```

Note that the regulations are colored according to expression changes of present and a previous value. So, it can be observed that whole gene expressions increase in comparison of initial steady-state. The overall regulations of gene expressions are as followings:

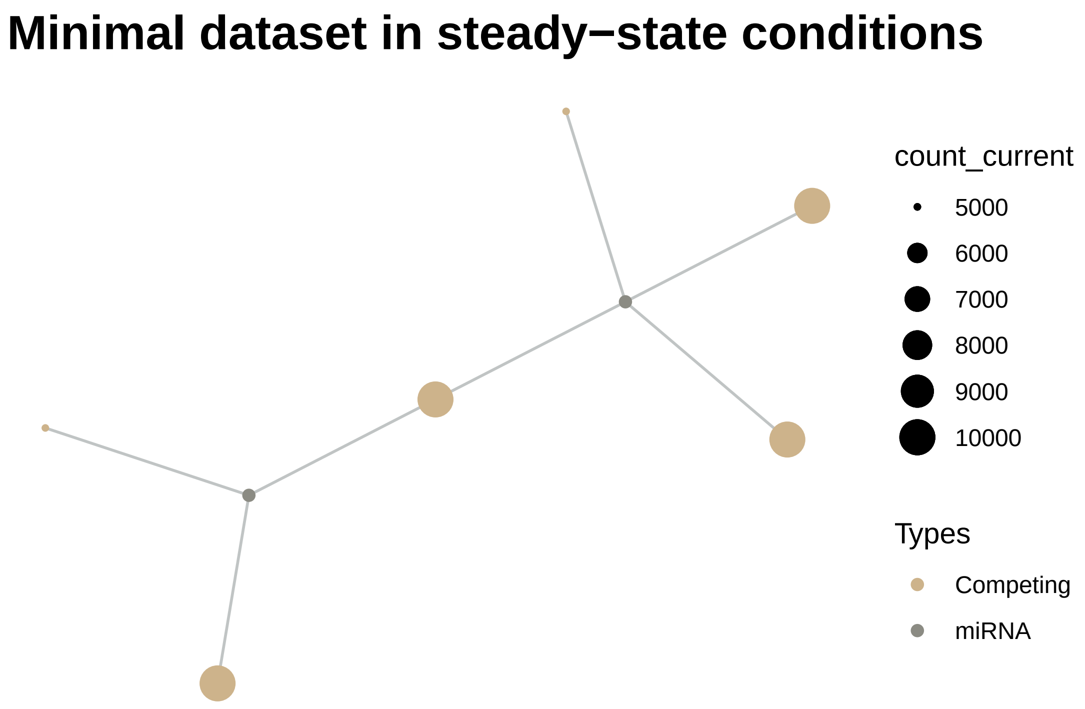
```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression)%>%
  update_how("Gene2", 2)%>%
  simulate(2)%>%
  activate(edges)%>%
  as_tibble()%>%
  select(Competing_name,comp_count_list,effect_list)%>%
  unnest()
```

```
## # A tibble: 21 x 3
##    Competing_name comp_count_list effect_list
##    <chr>                    <dbl>       <dbl>
##  1 Gene1                   10000         286.
##  2 Gene1                   10063.        222.
##  3 Gene1                   10062.        224.
##  4 Gene2                   10000         286.
##  5 Gene2                   19841.        444.
##  6 Gene2                   19845.        441.
##  7 Gene3                    5000         143.
##  8 Gene3                    5032.        111.
##  9 Gene3                    5031.        112.
## 10 Gene4                   10000         286.
## # ... with 11 more rows
```

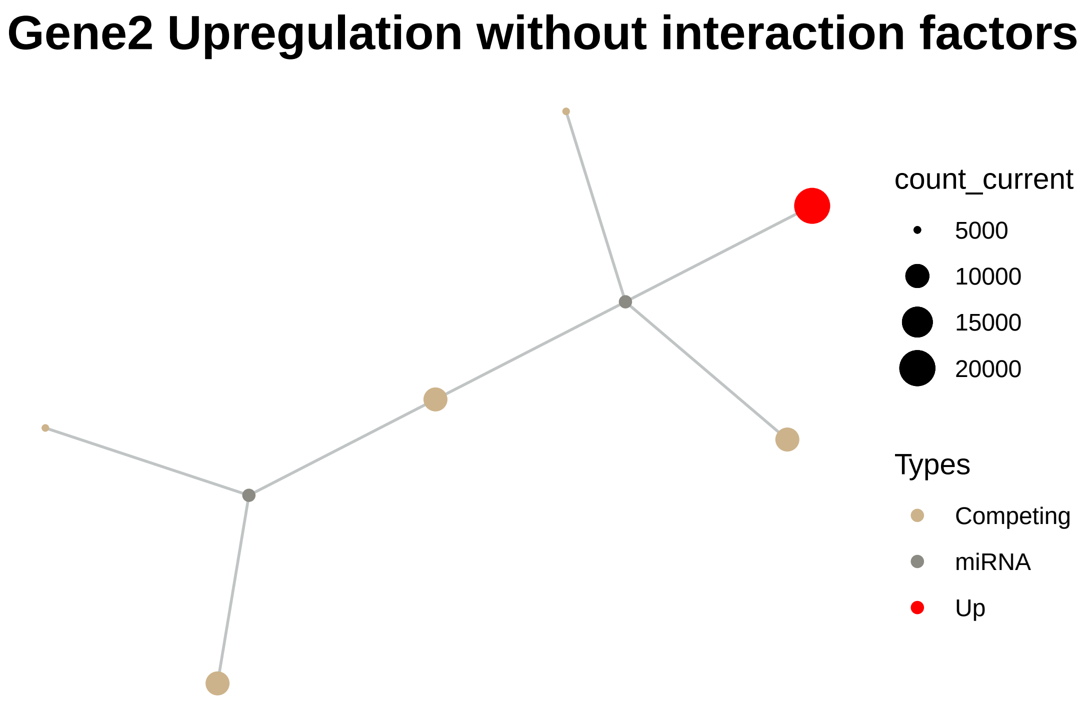
minsamp dataset analysis with interaction factors.

We have made the same analysis in present of interaction factors.

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto
  vis_graph(Competing_color = "navajowhite3", mirna_color = "ivory4", title = "Minimal dataset in stead
```

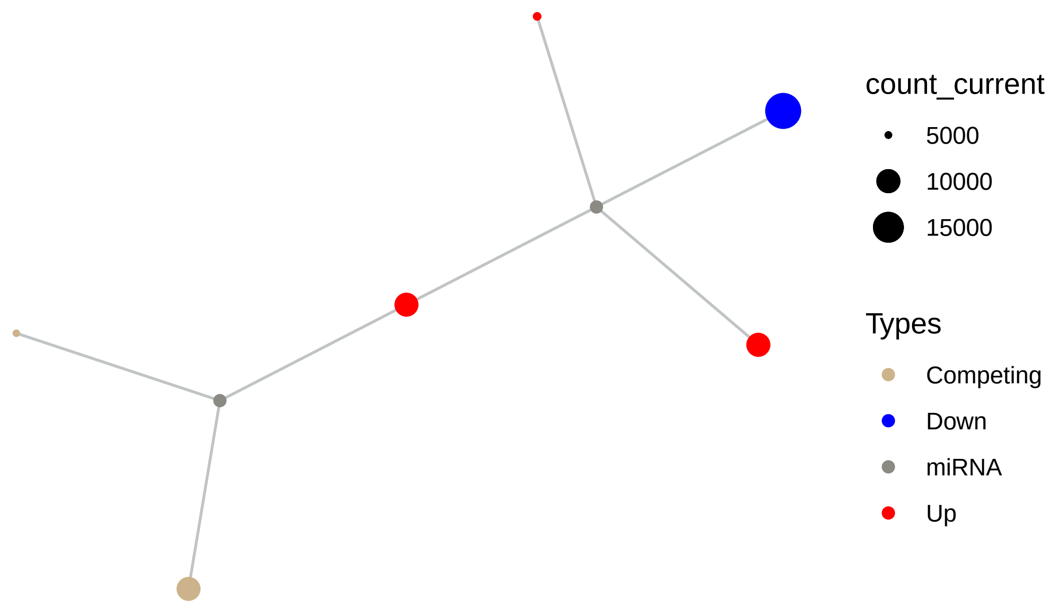


```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto
  update_how("Gene2", 2)%>%
  vis_graph(Competing_color = "navajowhite3", mirna_color = "ivory4", Upregulation = "red", title = "Ge
```

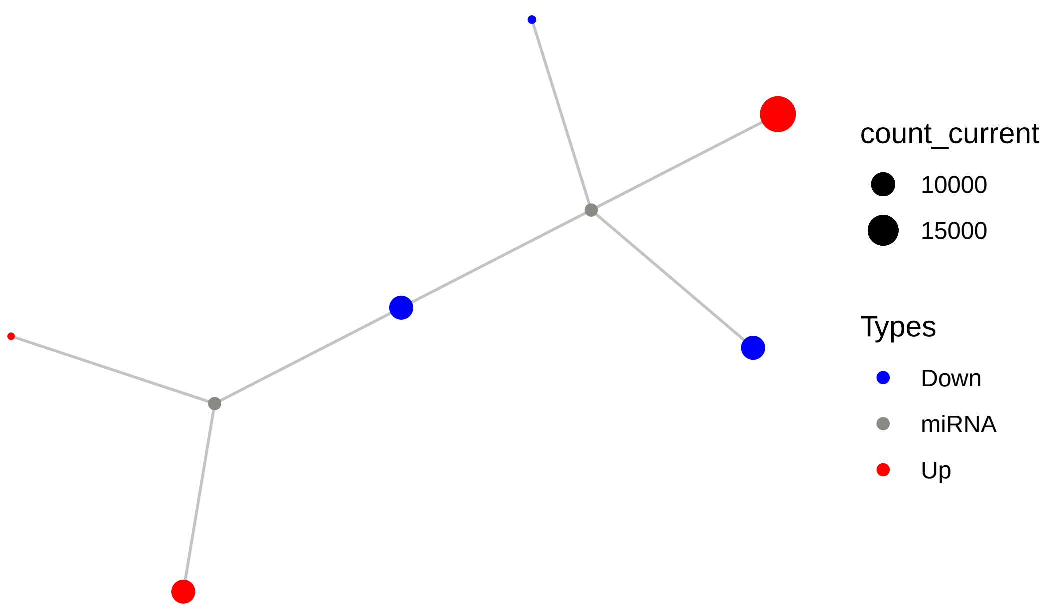


```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_fact
  update_how("Gene2", 2)%>%
  simulate_vis(Competing_color = "navajowhite3", mirna_color = "ivory4", Upregulation = "red", title =
```

Gene2 Upregulation without interaction factors − 1



Gene2 Upregulation without interaction factors − 2



```
## # A tbl_graph: 8 nodes and 7 edges
## #
## # A rooted tree
## #
## # Node Data: 8 x 7 (active)
##   name  type  node_id initial_count count_pre count_current
##   <chr> <chr>   <int>         <dbl>     <dbl>         <dbl>
## 1 Gene1 Comp~       1         10000    10065.        10064.
## 2 Gene2 Comp~       2         10000    19997.        19997.
## 3 Gene3 Comp~       3          5000     5023.         5023.
## 4 Gene4 Comp~       4         10000    10029.        10028.
## 5 Gene5 Comp~       5          5000     5000          5000.
## 6 Gene6 Comp~       6         10000    10000         10000.
## # ... with 2 more rows, and 1 more variable: changes_variable <chr>
## #
## # Edge Data: 7 x 23
##    from    to Competing_name miRNA_name Competing_expre~ miRNA_expression
##   <int> <int> <chr>          <chr>                 <dbl>            <dbl>
## 1     1     7 Gene1          Mir1                  10000             1000
## 2     2     7 Gene2          Mir1                  10000             1000
## 3     3     7 Gene3          Mir1                   5000             1000
## # ... with 4 more rows, and 17 more variables: energy <dbl>,
## #   seed_type <dbl>, region <dbl>, dummy <dbl>, afff_factor <dbl>,
## #   degg_factor <dbl>, comp_count_list <list>, comp_count_pre <dbl>,
## #   comp_count_current <dbl>, mirna_count_list <list>,
## #   mirna_count_pre <dbl>, mirna_count_current <dbl>,
## #   mirna_count_per_dep <dbl>, effect_current <dbl>, effect_pre <dbl>,
## #   effect_list <list>, mirna_count_per_comp <dbl>
```

When the graphs which were resulted from analyses were examined, it was observed that behaviours were same. But, when the results were analysed in terms of expression values, the regulation differences can be observed.

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto
  update_how("Gene2", 2)%>%
  simulate(3)%>%
  activate(edges)%>%
  as_tibble()%>%
  select(Competing_name,comp_count_list,effect_list)%>%
  unnest()
```

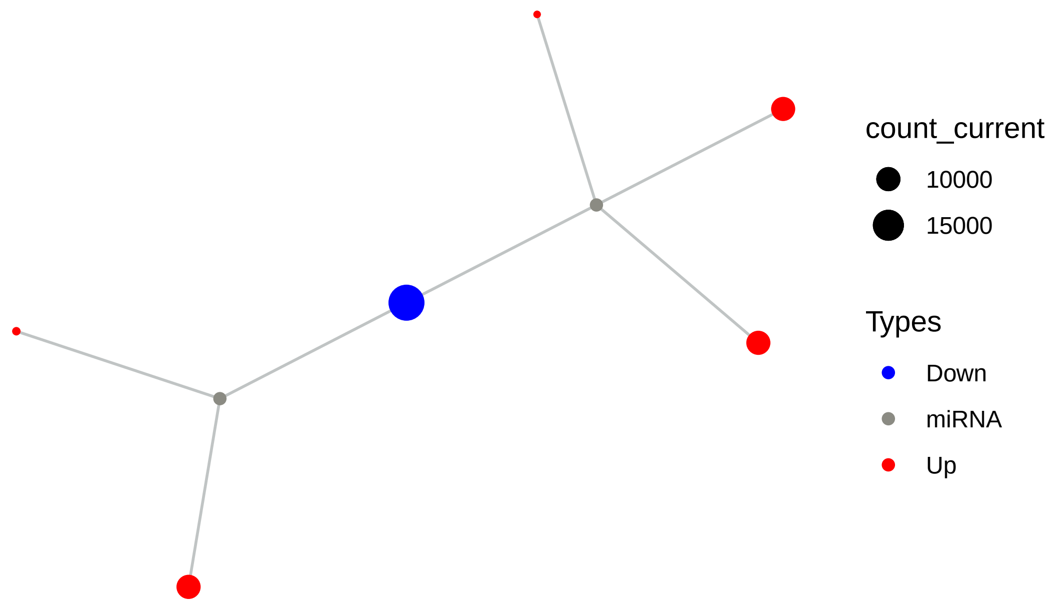
```
## # A tibble: 28 x 3
##    Competing_name comp_count_list effect_list
##    <chr>                    <dbl>       <dbl>
##  1 Gene1                    10000       263.
##  2 Gene1                    10065.      198.
##  3 Gene1                    10064.      199.
##  4 Gene1                    10064.      199.
##  5 Gene2                    10000         6.58
##  6 Gene2                    19997.        9.91
##  7 Gene2                    19997.        9.88
##  8 Gene2                    19997.        9.88
##  9 Gene3                     5000        91.5
## 10 Gene3                     5023.       68.8
## # ... with 18 more rows
```

Common target perturbation in *minsamp* dataset.

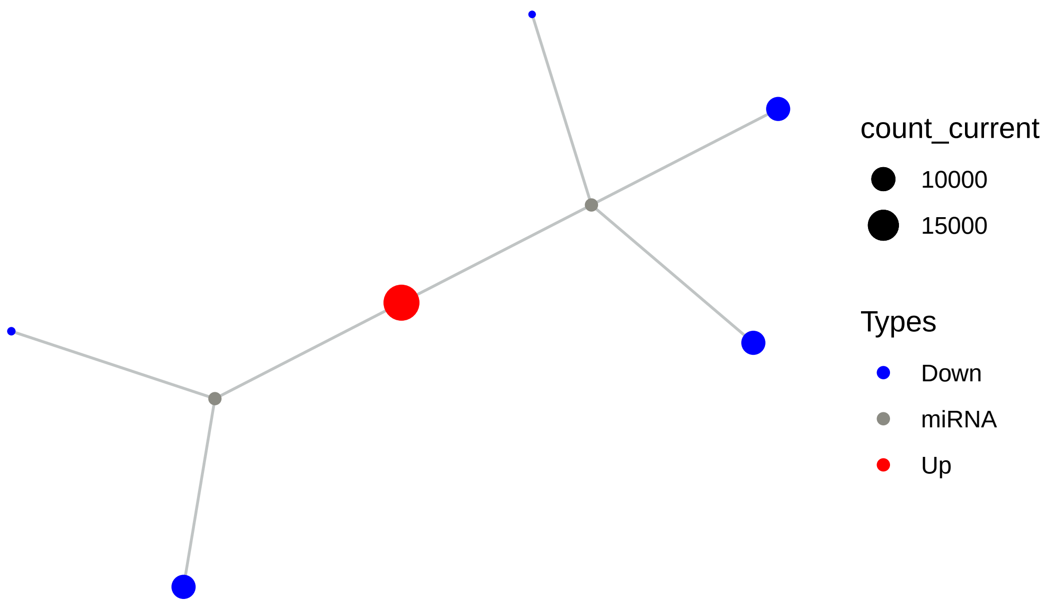
There are hundreds of defined miRNAs for human, so this results in presence of common targets of miRNAs in cells. Therefore, we have analysed perturbation efficiency of common target in *minsamp* dataset.

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto
  update_how("Gene4", 2)%>%
  simulate_vis(Competing_color = "navajowhite3", mirna_color = "ivory4", Upregulation = "red", title =
```

Gene2 Upregulation without interaction factors − 1



Gene2 Upregulation without interaction factors − 2



```
## # A tbl_graph: 8 nodes and 7 edges
## #
## # A rooted tree
## #
## # Node Data: 8 x 7 (active)
##   name  type  node_id initial_count count_pre count_current
##   <chr> <chr>   <int>         <dbl>     <dbl>         <dbl>
## 1 Gene1 Comp~       1         10000    10028.        10027.
## 2 Gene2 Comp~       2         10000    10001.        10001.
## 3 Gene3 Comp~       3          5000     5010.         5009.
## 4 Gene4 Comp~       4         10000    19803.        19806.
## 5 Gene5 Comp~       5          5000     5024.         5024.
## 6 Gene6 Comp~       6         10000    10044.        10044.
## # ... with 2 more rows, and 1 more variable: changes_variable <chr>
## #
## # Edge Data: 7 x 23
##    from    to Competing_name miRNA_name Competing_expre~ miRNA_expression
##   <int> <int> <chr>          <chr>                 <dbl>            <dbl>
## 1     1     7 Gene1          Mir1                  10000             1000
## 2     2     7 Gene2          Mir1                  10000             1000
## 3     3     7 Gene3          Mir1                   5000             1000
## # ... with 4 more rows, and 17 more variables: energy <dbl>,
## #   seed_type <dbl>, region <dbl>, dummy <dbl>, afff_factor <dbl>,
## #   degg_factor <dbl>, comp_count_list <list>, comp_count_pre <dbl>,
## #   comp_count_current <dbl>, mirna_count_list <list>,
## #   mirna_count_pre <dbl>, mirna_count_current <dbl>,
## #   mirna_count_per_dep <dbl>, effect_current <dbl>, effect_pre <dbl>,
## #   effect_list <list>, mirna_count_per_comp <dbl>
```

The common target perturbation (increasing to two fold at Gene4 expression in presence of interaction factors) resulted in more prominent efficiency at the same conditions (shown in following).

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto
  update_how("Gene4", 2)%>%
  simulate(3)%>%
  activate(edges)%>%
  as_tibble()%>%
  select(Competing_name,comp_count_list,effect_list)%>%
  unnest()
```

```
## # A tibble: 28 x 3
##    Competing_name comp_count_list effect_list
##    <chr>                    <dbl>       <dbl>
##  1 Gene1                    10000       263.
##  2 Gene1                    10028.      236.
##  3 Gene1                    10027.      237.
##  4 Gene1                    10027.      237.
##  5 Gene2                    10000         6.58
##  6 Gene2                    10001.        5.89
##  7 Gene2                    10001.        5.90
##  8 Gene2                    10001.        5.90
##  9 Gene3                     5000        91.5
## 10 Gene3                     5010.       81.9
## # ... with 18 more rows
```

Determination of perturbation efficiencies efficiencies of elements in system.

```
priming_graph(minsamp, competing_count = Competing_expression, miRNA_count = miRNA_expression, aff_facto

find_node_perturbation(sample_graph, how = 2, cycle = 3, limit = 0.1)
```

```
## # A tibble: 8 x 9
##   name  type  node_id initial_count count_pre count_current
##   <chr> <chr>   <int>         <dbl>     <dbl>         <dbl>
## 1 Gene1 Comp~       1         10000     10000         10000
## 2 Gene2 Comp~       2         10000     10000         10000
## 3 Gene3 Comp~       3          5000      5000          5000
## 4 Gene4 Comp~       4         10000     10000         10000
## 5 Gene5 Comp~       5          5000      5000          5000
## 6 Gene6 Comp~       6         10000     10000         10000
## 7 Mir1  miRNA       7          1000      1000          1000
## 8 Mir2  miRNA       8          2000      2000          2000
## # ... with 3 more variables: changes_variable <chr>,
## #   perturbation_efficiency <dbl>, perturbed_count <dbl>
```