```mathematica
In[◦]:= SetDirectory[
    "C:/Users/serha/OneDrive/Masaüstü/MyRepo/master_thesis_MMT003/210714_finalising/
      fxd_bounds"];
```

```mathematica
In[◦]:= Get["../../algoritm_packages/SingleNetworks-algorithm-package-2.wl"]
    (* ?SingleNetworks`* *)
```

```mathematica
In[◦]:= stoichioforhomosapiens =
    Drop[Import["../../210324_disc_time_windows_and_OR_model/iAT_PLT_636_stoichiomat.csv",
      HeaderLines → 1], None, {1}];
    SparseArray@stoichioforhomosapiens
```

Out[◦]= SparseArray[⊞ [Specified elements: 4006 | Dimensions: {738, 1008}]]

```mathematica
In[◦]:= stoichiometricmatrix = stoichioforhomosapiens;
    metabolites = 738;
    fluxexchanges = 1008;
    steadystatevector = ConstantArray[{0, 0}, metabolites];
    first[a_] := First /@ GatherBy[Ordering@a, a[[#]] &] // Sort;
```

```mathematica
In[◦]:= case = "bounds";
    intvalues = {2, 4};
    interval2 = "-5+5_quadrupled";

    interval = "(" <> ToString@intvalues[[1]] <> "," <> ToString@intvalues[[2]] <> ")";
    subsetpositionsforsequences = Import["../cases/subsetpositionsforsequences.mx"];
    boundaries =
      Import["../cases/boundaries_for_deleted_reaction_series_-5and5_quadrupled.mx"];
    boundariespos0 = Table[Position[boundaries[[i]], {0, 0}], {i, 10}];
    boundariesposval = Table[Position[boundaries[[i]], {-5, 5}], {i, 10}];
    boundariesa = Table[ReplacePart[
        (Table[ReplacePart[ConstantArray[{-500, 500}, fluxexchanges], MapThread[#1 → #2 &,
            {boundariespos0[[i]], ConstantArray[{0, 0}, Length@boundariespos0[[i]]]}]],
          {i, 10}])[[j]], MapThread[#1 → #2 &, {boundariesposval[[j]],
          ConstantArray[{-5, 5}, Length@boundariesposval[[j]]]}]], {j, 10}];
```

```mathematica
In[◦]:= syntheticseqgenerator[stoichiometricmatrix_,
      steadystatevector_, boundaries_, fluxexchanges_, subsetpositions_] :=
    Module[{coefficients, objectivefunctions, solutionvectors},
      coefficients = Table[RandomReal[intvalues, Length@subsetpositions], 50];
      objectivefunctions = Table[ReplacePart[ConstantArray[0., fluxexchanges],
          MapThread[#1 → #2 &, {subsetpositions, coefficients[[i]]}]], {i, 50}];
      solutionvectors = Chop[Table[LinearProgramming[-objectivefunctions[[i]],
          stoichiometricmatrix, steadystatevector, boundaries],
          {i, Length@objectivefunctions}], 10^-5];
      {objectivefunctions, solutionvectors}]
```

```
In[ ]:= (*AbsoluteTiming[resultset=
           Table[Quiet@Table[syntheticseqgenerator[stoichiometricmatrix,steadystatevector,
               j,fluxexchanges,i],{i,subsetpositionsforsequences}],{j,boundariesa}];]*)
```

```
In[ ]:= (*Export["C:/Users/serha/NonDrive/OR_model-25.06.2021/solution_vectors/"<>
           interval<>"solutionvectors_fxd"<>case<>"_-5and5_quadrupled.mx",
         Table[Flatten[resultset[[i]][[All,2]],1],{i,10}]]

          Export["C:/Users/serha/NonDrive/OR_model-25.06.2021/objective_functions/"<>
            interval<>"objfunc_fxd"<>case<>"_-5and5_quadrupled.mx",
          Table[Flatten[resultset[[i]][[All,1]],1],{i,10}]]*)
```

```
In[ ]:= (*solutionvectorslist=Table[Flatten[resultset[[i]][[All,2]],1],{i,10}];
        objfunctionslist=Table[Flatten[resultset[[i]][[All,1]],1],{i,10}];*)
```

```
In[ ]:= solutionvectorslist =
          Import["C:/Users/serha/NonDrive/OR_model-25.06.2021/solution_vectors/" <>
            interval <> "solutionvectors_fxd" <> case <> "_-5and5_quadrupled.mx"];
        objfunctionslist = Import[
            "C:/Users/serha/NonDrive/OR_model-25.06.2021/objective_functions/" <>
            interval <> "objfunc_fxd" <> case <> "_-5and5_quadrupled.mx"];
```

```
In[ ]:= AbsoluteTiming[featuredatalist =
          Table[MapThread[Dot, {objfunctionslist[[j]], solutionvectorslist[[j]]}], {j, 10}];]
```
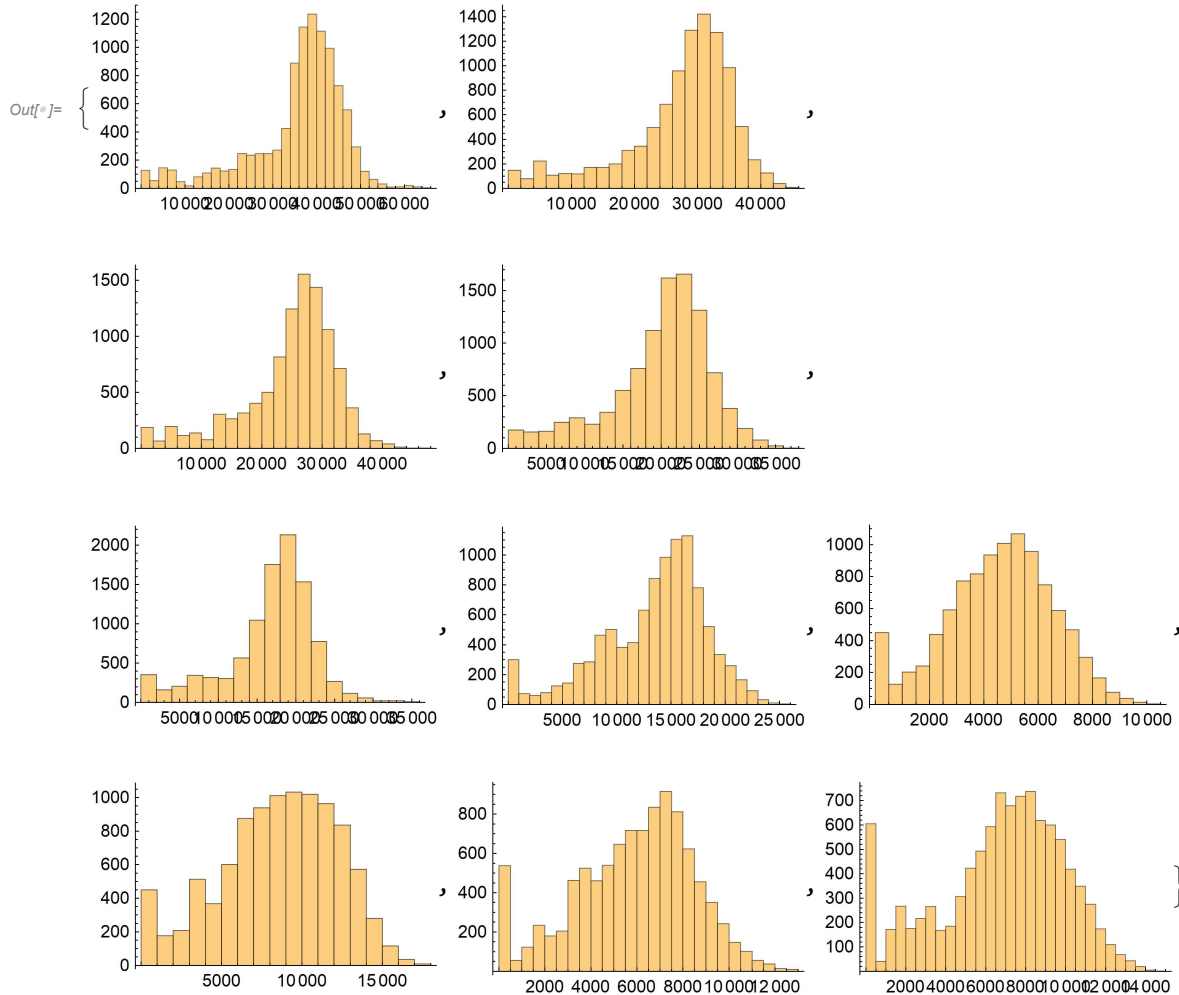
```
Out[ ]= {2.18884, Null}
```

```
In[ ]:= datafulllist = Table[Join[Partition[Range@10000, 1],
        Partition[Flatten@Table[ConstantArray[i, 50], {i, 200}], 1],
        Partition[featuredatalist[[j]], 1], 2], {j, 10}];
    Table[Histogram@datafulllist[[i]][[All, 3]], {i, 10}]
```



```
In[ ]:= thread = {{1, 1600}, {2, 1100}, {3, 1100}, {4, 850},
        {5, 850}, {6, 620}, {7, 260}, {8, 420}, {9, 310}, {10, 350}};
    Mean@thread[[All, 2]]

Out[ ]= 746
```

```
In[ ]:= thread = Thread[{Range@10, 635}]

Out[ ]= {{1, 635}, {2, 635}, {3, 635}, {4, 635},
        {5, 635}, {6, 635}, {7, 635}, {8, 635}, {9, 635}, {10, 635}}
```

```
In[ ]:= AbsoluteTiming[widthdataFixedstep2 =
        Table[snetworkdatabinned[3, i[[2]], datafulllist[[i[[1]]]]], {i, thread}];]

Out[ ]= {7.24309, Null}
```

```
In[ ]:= graphsandnodenumbers12 = Table[snetworkgraph[widthdataFixedstep2[[i]][[1]],
            widthdataFixedstep2[[i]][[2]], 2, 7, 400, Green], {i, 10}];
        graphsandnodenumbers12[[All, 2]]

Out[ ]= {102, 70, 71, 56, 56, 41, 17, 29, 21, 24}


In[ ]:= modularityvalues12 = Table[N@GraphAssortativity[graphsandnodenumbers12[[i]][[1]],
            FindGraphCommunities[graphsandnodenumbers12[[i]][[1]]], "Normalized" → False],
          {i, Length@graphsandnodenumbers12}];


In[ ]:= singlerandomgraphsdegfxd12 =
          Table[randomizinggraphdegfxd[i], {i, graphsandnodenumbers12[[All, 1]]}];
        singlerandomerdrenmodularityvalues12 =
          Table[N@GraphAssortativity[singlerandomgraphsdegfxd12[[i]],
              FindGraphCommunities[singlerandomgraphsdegfxd12[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphsdegfxd12}];
        singlerandomgraphscomm12 = Table[randomizinggraphmod[i],
            {i, graphsandnodenumbers12[[All, 1]]}];
        singlerandomcommmodularityvalues12 =
          Table[N@GraphAssortativity[singlerandomgraphscomm12[[i]],
              FindGraphCommunities[singlerandomgraphscomm12[[i]]], "Normalized" -> False],
            {i, Length@singlerandomgraphscomm12}];


In[ ]:= AbsoluteTiming[Zscoresmodularity12 =
          Table[zscorefunctionfortwonullmodels[i], {i, graphsandnodenumbers12[[All, 1]]}];]

Out[ ]= {108.014, Null}


In[ ]:= bucketnode12 = graphsandnodenumbers12[[All, 2]]

Out[ ]= {102, 70, 71, 56, 56, 41, 17, 29, 21, 24}
```
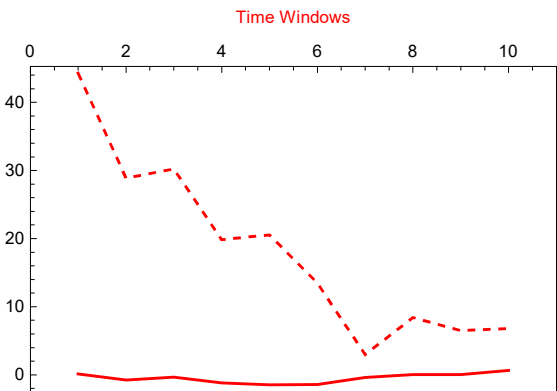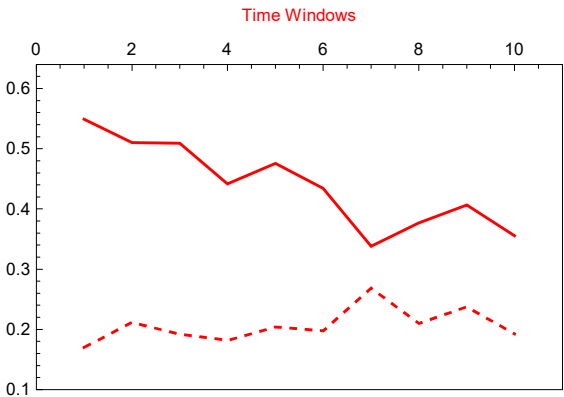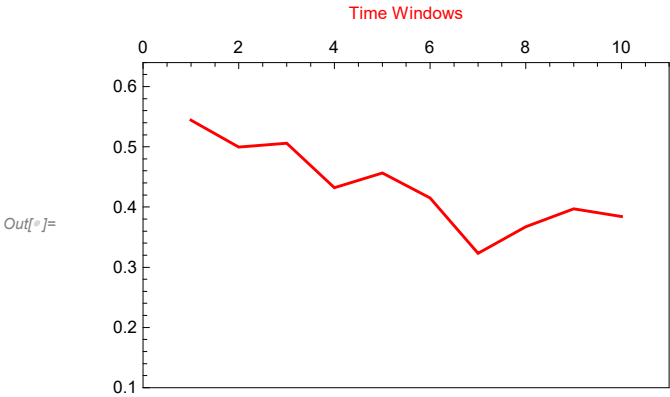
```
In[ ]:= modularityvaluestimewinsmall = modularityvalues12;
       randommodtimewinsmalldegreefxd = singlerandomerdrenmodularityvalues12;
       randommodtimewinsmallcomm = singlerandomcommmodularityvalues12;
       Zscoretimewinsmall = Zscoresmodularity12;
       modularityplotrange = {0.1, 0.64};
       (*MinMax[{modularityvalues1,singlerandomcommmodularityvalues1,
         singlerandomerdrenmodularityvalues1,modularityvalues12}]*)
       padding = 38;
       win2 = 10;
       Row[{ListLinePlot[Thread[{Range@win2, modularityvaluestimewinsmall}],
          Frame → True, ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
          FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}}, PlotStyle → Red,
          ImageSize → 350, PlotRange → {{0, win2 + 1}, modularityplotrange}],
         Row[{ListLinePlot[{Thread[{Range@win2, randommodtimewinsmalldegreefxd}],
             Thread[{Range@win2, randommodtimewinsmallcomm}]}, Frame → True,
            ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
            FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}},
            PlotStyle → {{Dashed, Red}, Red}, ImageSize → 350,
            PlotRange → {{0, win2 + 1}, modularityplotrange}],
          ListLinePlot[{Thread[{Range@win2, Zscoretimewinsmall[[All, 1]]}],
             Thread[{Range@win2, Zscoretimewinsmall[[All, 2]]}]}, Frame → True,
            ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
            FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}},
            PlotStyle → {{Dashed, Red}, Red}, ImageSize → 350,
            PlotRange → {{0, win2 + 1}, MinMax[Flatten[Zscoretimewinsmall], 1]}]}],
        LineLegend[{Dashed, Black}, {"Degrees Fixed N.M.", "Modularity N.M."},
         LegendMargins → 0, LegendMarkerSize → {20, 20}], Spacer@0.1}]
```

Out[◦]=

Time Windows

Time Windows

Time Windows

--- Degrees Fixed N.M.

— Modularity N.M.

In[◦]:= **AbsoluteTiming[widthdataFixedbucket2 =**
   Table[snetworkdatafxdbucket[3, bucketnode12[[i]], datafulllist[[i]]], {i, 10}];]

Out[◦]= {2.70495, Null}

```
In[ ]:= graphsandnodenumbers32 = Table[snetworkgraph[widthdataFixedbucket2[[i]][[1]],
            widthdataFixedbucket2[[i]][[2]], 1.5, 7, 400, Green], {i, 10}];
      modularityvalues32 = Table[N@GraphAssortativity[graphsandnodenumbers32[[i]][[1]],
            FindGraphCommunities[graphsandnodenumbers32[[i]][[1]]], "Normalized" → False],
          {i, Length@graphsandnodenumbers32}];
```

```
In[ ]:= singlerandomgraphsdegfxd32 =
        Table[randomizinggraphdegfxd[i], {i, graphsandnodenumbers32[[All, 1]]}];
      singlerandomerdrenmodularityvalues32 =
        Table[N@GraphAssortativity[singlerandomgraphsdegfxd32[[i]],
            FindGraphCommunities[singlerandomgraphsdegfxd32[[i]]], "Normalized" -> False],
          {i, Length@singlerandomgraphsdegfxd32}];
      singlerandomgraphscomm32 = Table[randomizinggraphmod[i],
          {i, graphsandnodenumbers32[[All, 1]]}];
      singlerandomcommmodularityvalues32 =
        Table[N@GraphAssortativity[singlerandomgraphscomm32[[i]],
            FindGraphCommunities[singlerandomgraphscomm32[[i]]], "Normalized" -> False],
          {i, Length@singlerandomgraphscomm32}];
```

```
In[ ]:= AbsoluteTiming[Zscoresmodularity32 =
        Table[zscorefunctionfortwonullmodels[i], {i, graphsandnodenumbers32[[All, 1]]}];]
```
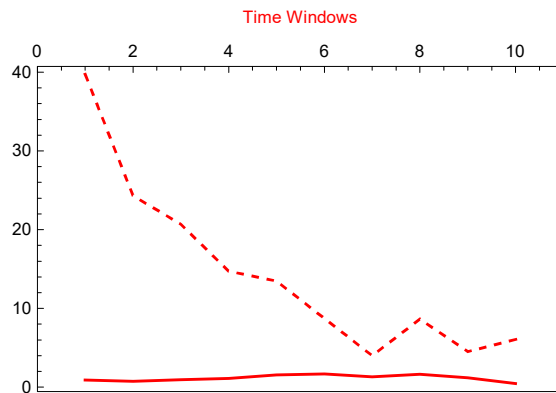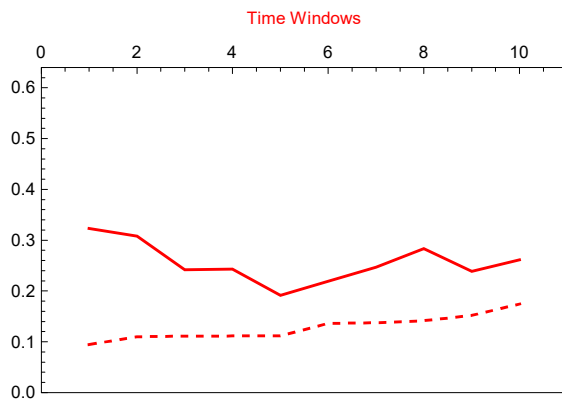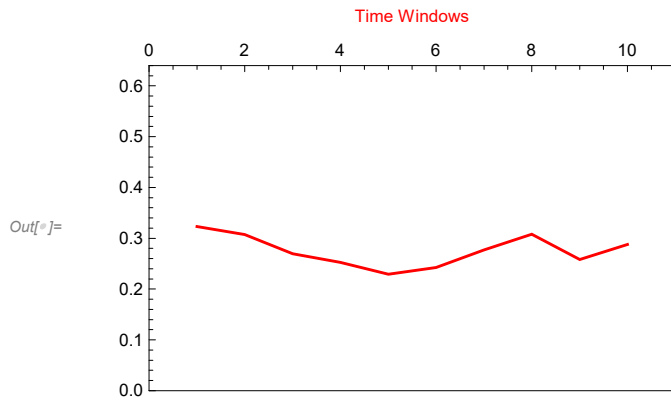
```
Out[ ]= {275.167, Null}
```

```
In[*]:= modularityvaluestimewinsmall = modularityvalues32;
       randommodtimewinsmalldegreefxd = singlerandomerdrenmodularityvalues32;
       randommodtimewinsmallcomm = singlerandomcommmodularityvalues32;
       Zscoretimewinsmall = Zscoresmodularity32;
       modularityplotrange = {0, 0.64};
       (*MinMax[{modularityvalues1,singlerandomcommmodularityvalues1,
         singlerandomerdrenmodularityvalues1,modularityvalues12}]*)
       padding = 38;
       win2 = 10;
       Row[{ListLinePlot[Thread[{Range@win2, modularityvaluestimewinsmall}],
           Frame → True, ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
           FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}}, PlotStyle → Red,
           ImageSize → 350, PlotRange → {{0, win2 + 1}, modularityplotrange}],
         Row[{ListLinePlot[{Thread[{Range@win2, randommodtimewinsmalldegreefxd}],
               Thread[{Range@win2, randommodtimewinsmallcomm}]}, Frame → True,
             ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
             FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}},
             PlotStyle → {{Dashed, Red}, Red}, ImageSize → 350,
             PlotRange → {{0, win2 + 1}, modularityplotrange}],
           ListLinePlot[{Thread[{Range@win2, Zscoretimewinsmall[[All, 1]]}],
               Thread[{Range@win2, Zscoretimewinsmall[[All, 2]]}]}, Frame → True,
             ImagePadding → padding, FrameTicks → {{All, None}, {None, All}},
             FrameLabel → {{None, None}, {None, Style["Time Windows", Red]}},
             PlotStyle → {{Dashed, Red}, Red}, ImageSize → 350,
             PlotRange → {{0, win2 + 1}, MinMax[Flatten[Zscoretimewinsmall], 1]}]}],
         LineLegend[{Dashed, Black}, {"Degrees Fixed N.M.", "Modularity N.M."},
          LegendMargins → 0, LegendMarkerSize → {20, 20}], Spacer@0.1}]
```

Out[ ]=

--- Degrees Fixed N.M.

— Modularity N.M.

```
In[ ]:= Export["plot_values/fxd_" <> case <> "/" <> interval <>
          "_" <> interval2 <> "-modularityvalues-fss.mx", modularityvalues12]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <>
          "-singrand-erd-modularityvalues-fss.mx", singlerandomerdrenmodularityvalues12]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <>
          "-singrand-comm-modularityvalues-fss.mx", singlerandomcommmodularityvalues12]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <> "-zscores-fss.mx",
          Zscoresmodularity12]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <>
          interval2 <> "-modularityvalues-fbs.mx", modularityvalues32]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <>
          "-singrand-erd-modularityvalues-fbs.mx", singlerandomerdrenmodularityvalues32]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <>
          "-singrand-comm-modularityvalues-fbs.mx", singlerandomcommmodularityvalues32]
       Export["plot_values/fxd_" <> case <> "/" <> interval <> "_" <> interval2 <> "-zscores-fbs.mx",
          Zscoresmodularity32]
```

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-modularityvalues-fss.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-singrand-erd-modularityvalues-fss.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-singrand-comm-modularityvalues-fss.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-zscores-fss.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-modularityvalues-fbs.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-singrand-erd-modularityvalues-fbs.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-singrand-comm-modularityvalues-fbs.mx

Out[ ]= plot_values/fxd_bounds/(2,4)_-5+5_quadrupled-zscores-fbs.mx