What Popularity tells us about a Wikipedia Sub-Graph?

About this work

Studying the popularity time series of two Wikipedia pages and their correlation, can we say if two pages are linked? And how strong is their link?

- **Characteristics of The Wikipedia Sub-Graph**
  The graph in analysis is extracted starting from a central page and considering only English pages. We considered a recent event that involves the central page. We considered all the out-links until the second neighbors, and we selected all the links with at least 2 repetitions. Multiple links are very common and we used this as a measure of the strength of the connection, the ***Links Multiplicity***. We obtained a weighted directed Graph of 3126 Edges and 2164 Vertices.
- **General Characteristics of Popularity Time Series**
  Looking at the time series we observed a global behavior, we studied an independent set of 10000 pages in a time window of two months. We saw a ***weekly effect***, there is weekly fluctuation of the number of visitors, about 20-30%. We calculated 7 daily coefficient in order to correct Popularity and reduce the overestimation of correlations. Looking at an independent set of 1000 pages in a time window of a year, we observed a *seasonal effect*, but in our time window this is negligible.
- **What happens when there is an Event? Does Popularity diffuse?**
  I chose a recent event with a peak around the release date of the movie "Star Trek Into Darkness" (May 16th 2013). I built the weighted sub-graph of Wikipedia English pages, starting from the page of the movie. I collected all the time series of this pages in a time window of two month, around the Event. Looking at the time series we observed that, at this daily resolution, there is ***no propagation of Popularity***, there is no delay between two time series. With this data we can't consider to study a dynamic. And also we can't study causal relations. For this reason we decided to study the relation between different pages Popularity with the Pearson Correlation.
- **Can we predict the real connections of the Graph from Popularity?**
  The correlations between two time series imply a symmetric Graph:
  N_Links (v , u) ~ 1 / (1 - Correlation (v , u))
  We observed an interesting relation between the total number of links of a page and its average Popularity:
  N_Links(v) ~ Popularity (v)^0.5
  Our hypothesis is a prediction rule of this form:
  Multiplicity (u , v) ~ [1 / (1 - Correlation (u , v)]^a [Popularity (u)]^b [Popularity (v)]^c
  We tested our prediction rules with different coefficients and we defined an error measure.

The Wikigraph

Useful Functions

Starting from an initial Wikipedia Page, this group of functions built the weighted graph of depth 2 (we consider only links with multiplicity > 1). Multiplicity is our weight.

Built the Graph

```
initialTitle = "Star_Trek_Into_Darkness";
```

```
Needs["GraphUtilities`"]
```

```
cleanlinks[links_] := Module[
    {cleanedlinks},
    cleanedlinks = Select[
        links,
        StringMatchQ[#, "http://en.wikipedia.org/wiki/"~~Except[":"]..] &
    ];
    cleanedlinks = Select[ Gather[DeleteCases[cleanedlinks,"http://en.wikipedia.org/wi
    Tally@Flatten[cleanedlinks]
    (*First /@ cleanedlinks*)
]
(*this function will consider links multiplicity and links with at least 2 repetitions
```

```
importFirstLevelLinks[init_]:= cleanlinks @
    Import["http://en.wikipedia.org/wiki/"<>init, "Hyperlinks"]
(* this output is a list of weighted links with their multiplicity *)
```

```
createFirstLevelWeightedEdges[init_,weightedlinks_]:=
{init,Last@StringSplit[#[[1]],"/"],#[[2]]}& /@ weightedlinks
```

```
findFirstNeighbors[firstLevelLinks_]:= Last@StringSplit[#,"/"]& /@firstLevelLinks[[All
```

```
importSecondLevelLinks[firstLevelLinks_]:= (cleanlinks @ Import[#,"Hyperlinks"])& /@ f
```

```
createSecondLevelWeightedEdges[firstNeighbors_,secondLevelLinks_]:=
    Module[{secondNeighbors,secondNeighborsWeighted},
        secondNeighbors=Map[Last[StringSplit[#,"/"]]&,secondLevelLinks[[All,All,1]],{2
        secondNeighborsWeighted=Transpose[{secondNeighbors[[#]],secondLevelLinks[[#,Al
        Flatten[Outer[
            Flatten[{#1,#2}]&,{firstNeighbors[[#]]},{secondNeighborsWeighted[[#]]},2]&
]
```

```
createTheGraph[init_]:=Module[{firstLevel,secondLevel},
    firstLevel=importFirstLevelLinks[init];
    secondLevel=importSecondLevelLinks[firstLevel];
    Flatten[{createFirstLevelWeightedEdges[init,firstLevel],createSecondLevelWeightedE


]
```

Remove Self Loops

Correct Multiplicity: remove Multi Edges and update Weights

Operations on the Graph

Static visualization with Weights or Ranks

```
FancyRankingGraph[list_,ranks_,title_,center_]:=
    Graph[DirectedEdge@@@ list,
        VertexSize→ranks,GraphHighlight→{center},
        PlotLabel→Framed[Style[title,15,"Arial"],
        Background→LightBlue],VertexSize→1,ImageSize→700,GraphLayout→"SpringElectrica
        VertexLabels→{center→Framed[Style[center,15,"Arial"],Background→White]}
]
```

```
NormRankings[rankings_]:=
With[{mean=Mean[rankings[[All,2]]]},

(#→N[#2/mean]& @@@ rankings)
]
```

```
ExtractWeightedAjdacencyMatrix[listWeightedEdges_, listVertices_]:=
With[{l=Position[StarTrekNet,{listVertices[[#[[1]]]],listVertices[[#[[2]]]],__}]},
    If[Length[l]≠0,StarTrekNet[[l[[1,1]],3]],0]]&/@ Tuples[Range[1,Length[listVertices
```

Dynamic visualization with Popularity

```
AssignDynamicWeights[Vertices_,TimeSeries_]:=
    MapThread[#1 → Flatten[#2[[All,All,2]]] &, {Vertices,TimeSeries}]
```

```
PopWeightedGraph[list_,weights_,title_,center_]:=Graph[DirectedEdge@@@ list,
    VertexSize→weights,GraphHighlight→{center},
    PlotLabel→Framed[Style[title,15,"Arial"],
    Background→LightBlue],VertexSize→1,ImageSize→700,GraphLayout→"SpringElectricalEmb
    VertexLabels→{center→Framed[Style[center,15,"Arial"],Background→White]}
]
```

```
DynamicPopularity[Edges_,popweights_]:=Manipulate[
    With[
        {mean = Mean[Flatten[popweights[[All, 2]]]]},
        PopWeightedGraph[Edges,#1→N[#2[[w]]/ mean]&@@@ popweights
        ,"Avatar, popularity","Avatar_%282009_film%29"]
    ]
    ,{{w,1},1, Length[popweights[[1,2]]], 1}
]
```

Wikigraph of our Data Set

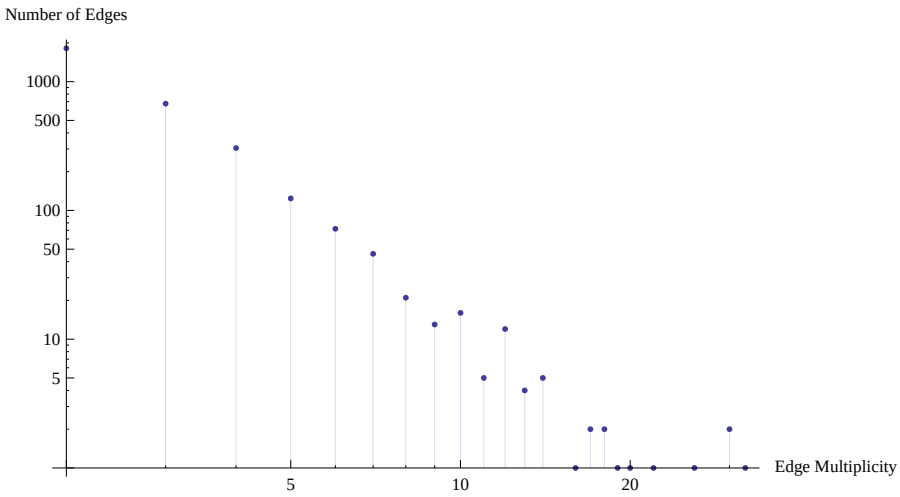
Star Trek Wikigraph: 3126 Edges, 2164 Vertices

We select the most relevant links in each page, starting from "Star Trek Into Darkness", and we take pages until its second neighbors. We consider multiplicity of links as a measure of their importance. We filter pages using Multiplicity > 1.

```
StarTrekNet = NewGraphWithCorrectMultiplicity[
   Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/
      allStarTrekWeighted.tsv", "TSV"]];
```

Edge Multiplicity

```
ListLogLogPlot[Tally[StarTrekNet[[All, 3]]], Filling → Axis,
 PlotRange → All, AxesLabel → {"Edge Multiplicity", "Number of Edges"}]
```



I consider the unweighted graph:

```
UnweightedST = StarTrekNet[[All, ;; 2]];
```

```
STGraph = Graph[DirectedEdge @@@ UnweightedST];
```

```
STVertices = VertexList[STGraph];
```

K-Cores decomposition

```
Length[VertexList[STGraph]]
```

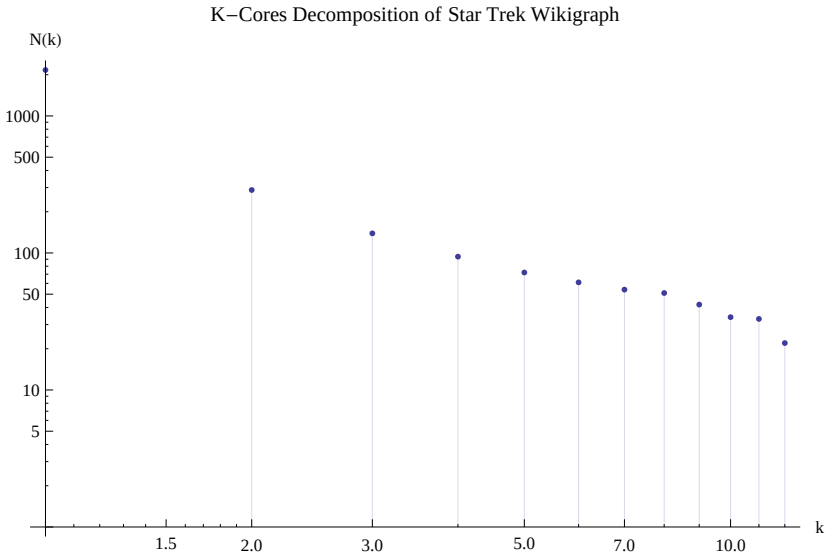
```
2164
```

```
kcores = Map[KCoreComponents[STGraph, #] &, Range[1, 12]];
```

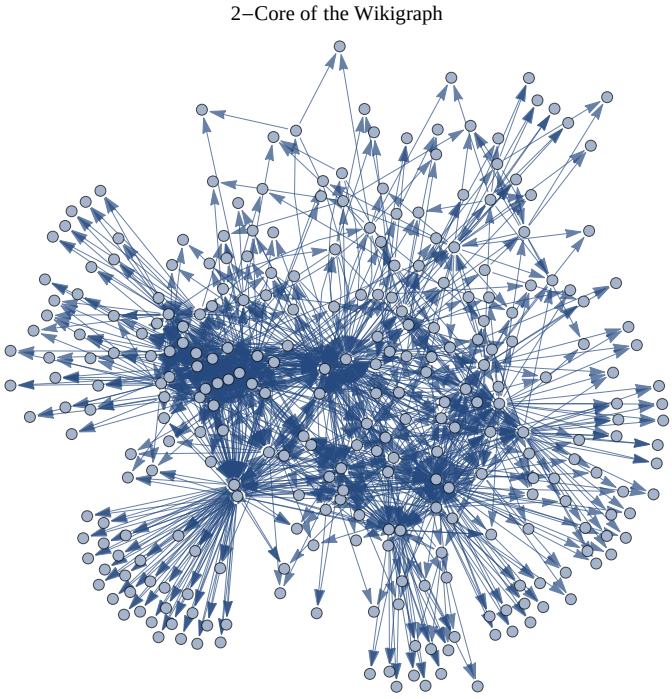
```
Length[Flatten[#]] & /@ kcores
```

```
{2164, 288, 139, 94, 72, 61, 54, 51, 42, 34, 33, 22}
```

```
ListLogLogPlot[Transpose[{Range[1, 12], Length[#] & @@@ kcores}],
 PlotLabel → "K-Cores Decomposition of Star Trek Wikigraph",
 PlotRange → All, AxesLabel → {"k", "N(k)"}, Filling → Axis]
```




```
Subgraph[STGraph, KCoreComponents[STGraph, 2],
 PlotLabel → "2-Core of the Wikigraph"]
```



Page Rank

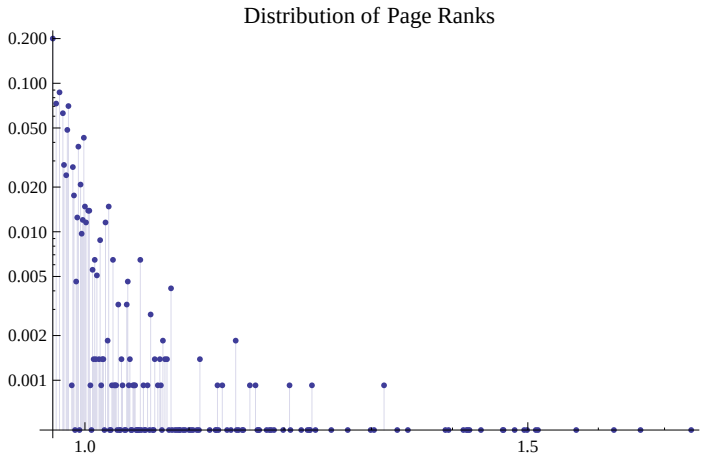
```
Needs["GraphUtilities`"]
```

```
STRanks = PageRanks[STGraph];
```

```
meanRanks = Mean[STRanks[[All, 2]]]
```

```
0.000462107
```

```
ListLogLogPlot[Transpose[
  {Tally[N[Round[(1000 * STRanks[[All, 2]] / meanRanks)] / 1000]][[All, 1]],
   Tally[N[Round[(1000 * STRanks[[All, 2]] / meanRanks)] / 1000]][[All, 2]] /
    2164.}], Filling → Axis,
 PlotLabel → "Distribution of Page Ranks", PlotRange → All]
```

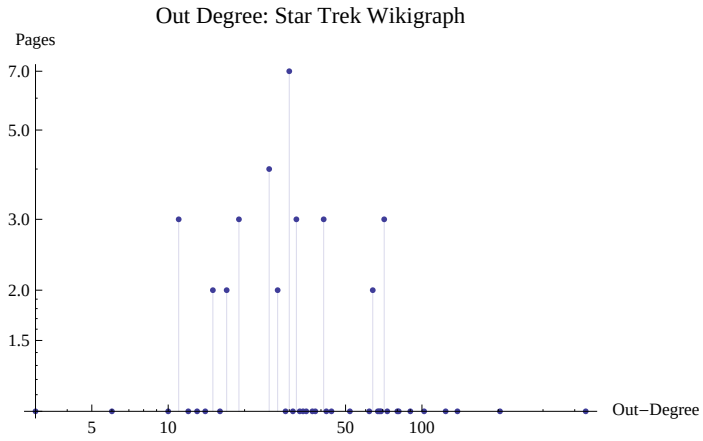



```
STNormRanks = NormRankings[STRanks];
```

```
STVertices = VertexList[STGraph];
```

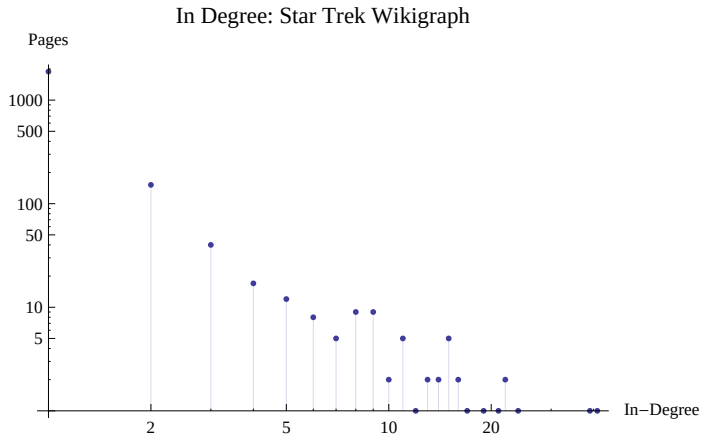
IN Degree and OUT Degree in the Star Trek Wikigraph

```
ListLogLogPlot[Tally[VertexOutDegree[STGraph]], Filling → Axis,
 PlotRange → All, PlotLabel → "Out Degree: Star Trek Wikigraph",
 AxesLabel → {"Out-Degree", "Pages"}]
```



This depends on how the graph was built, but we can roughly see a power law behavior in the InDegree plot.

```
ListLogLogPlot[Tally[VertexInDegree[STGraph]], Filling → Axis,
 PlotRange → All, PlotLabel → "In Degree: Star Trek Wikigraph",
 AxesLabel → {"In-Degree", "Pages"}]
```



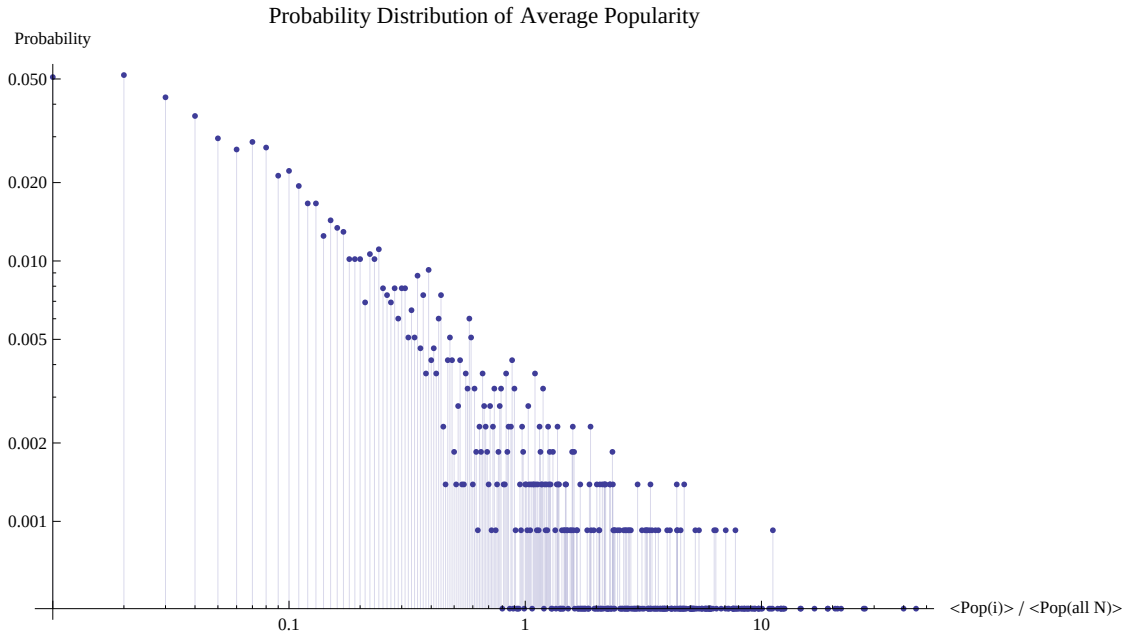
Average Popularity of each page

```
STPop = N[cleanedSTTimeSeries[[All, ;; 2, All, 2]] /
    Mean[Flatten[cleanedSTTimeSeries[[All, ;; 2, All, 2]]]]];

meansSTPop = N@Mean[Flatten[cleanedSTTimeSeries[[#, ;; 2, All, 2]]]] & /@
   Range[1, Length[cleanedSTTimeSeries]];

bigmean = Mean[meansSTPop]
```

```
1807.9
```

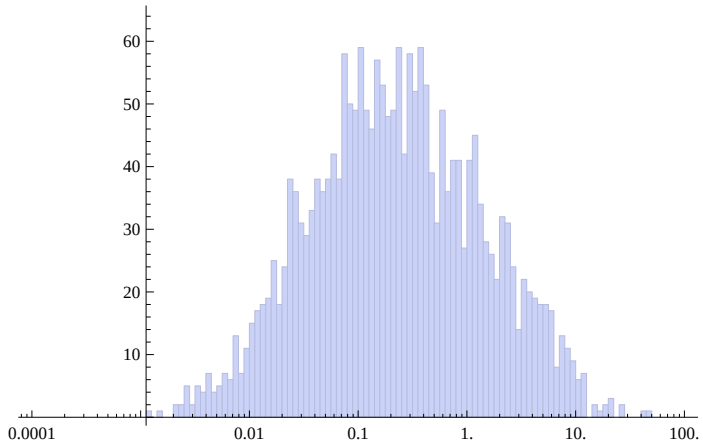
```
meansSTPopNorm = meansSTPop / bigmean;

Export[
  "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meansSTPopNorm.tsv",
  meansSTPopNorm, "TSV"];

meansSTPop = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meanSTPopNorm2Months.
     tsv", "TSV"];

STPopNorm = MapThread[# → #2 &, {STVertices, meansSTPop}];
```

```
ListLogLogPlot[
 Transpose[{Tally[N[Round[(100 * meansSTPop / bigmean)] / 100]][[All, 1]],
   Tally[N[Round[(100 * meansSTPop / bigmean)] / 100]][[All, 2]] / 2164.}],
 PlotLabel → "Probability Distribution of Average Popularity", PlotRange → All,
 Filling → Axis, AxesLabel → {"<Pop(i)> / <Pop(all N)>", "Probability"}]
```



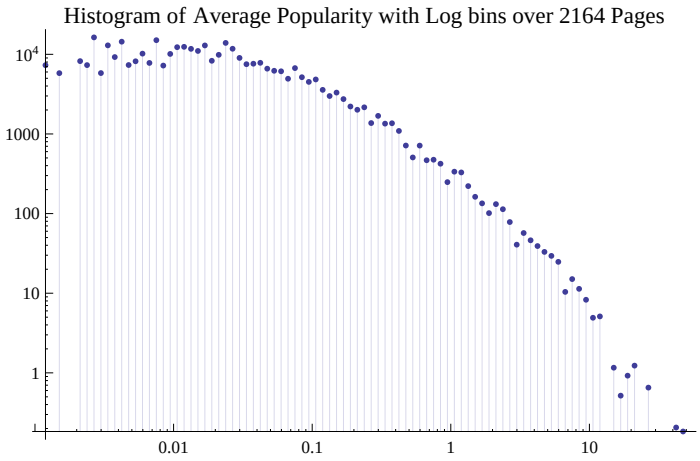
Histogram of Popularity with logarithmic bins

```
binMin = 0.0001;
binMax = 100;
binWidthLog = 0.05;
bins = {10^Range[Log[10, binMin], Log[10, binMax], binWidthLog]};

Histogram[meansSTPopNorm, {"Log", bins}]
```



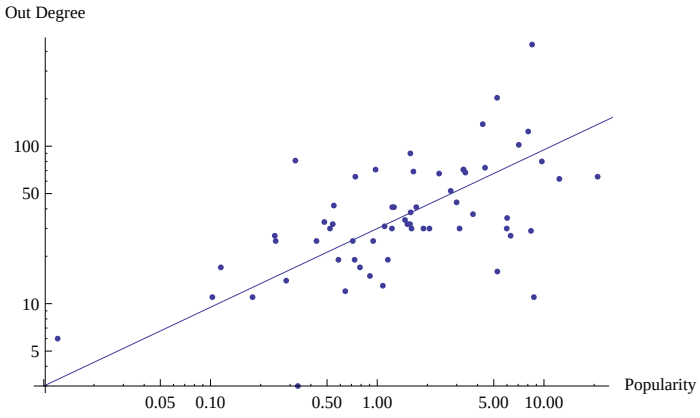

```
list = HistogramList[meansSTPopNorm, {"Log", bins}];
```

```
ListLogLogPlot[Transpose[{MovingAverage[#1, 2], #2 / Differences[#1]} & @@ list],
 PlotLabel → "Histogram of Average Popularity with Log bins over 2164 Pages",
 Filling → Axis]
```

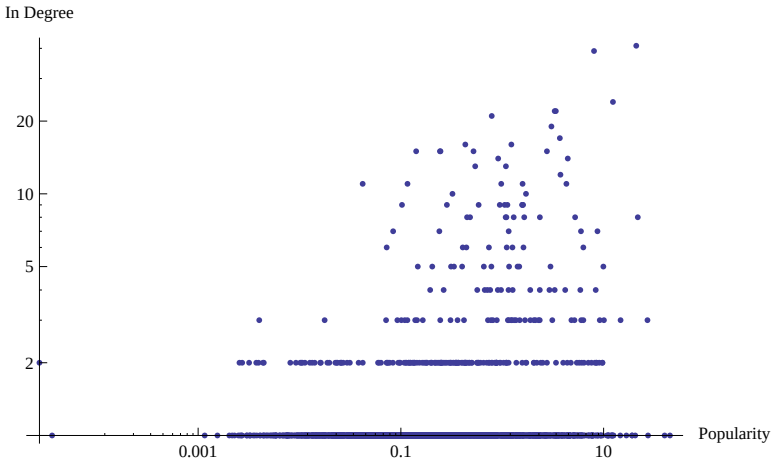


Are the number of links of a page related to its popularity?

```
Show[ListLogLogPlot[Transpose[{meansSTPopNorm, VertexOutDegree[STGraph]}],
  PlotRange → All, AxesLabel → {"Popularity", "Out Degree"}],
 LogLogPlot[30 x^0.5, {x, 0.01, 10 000}]]
```



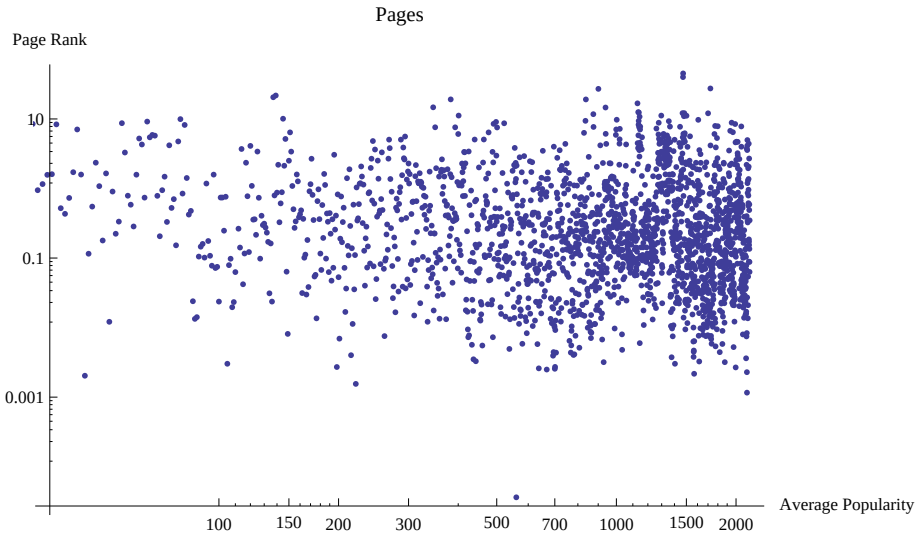

```
ListLogLogPlot[Transpose[{meansSTPopNorm, VertexInDegree[STGraph]}],
 PlotRange → All, AxesLabel → {"Popularity", "In Degree"}]
```



Page Rank VS Average Popularity

Confront between Average Popularity (from May 1st to June 30th 2013) and Page Rank of the Wikigraph.

```
ListLogLogPlot[{meansSTPopNorm, STNormRanks}, PlotRange → All,
 AxesLabel → {"Average Popularity", "Page Rank"}, PlotLabel → "Pages"]
```

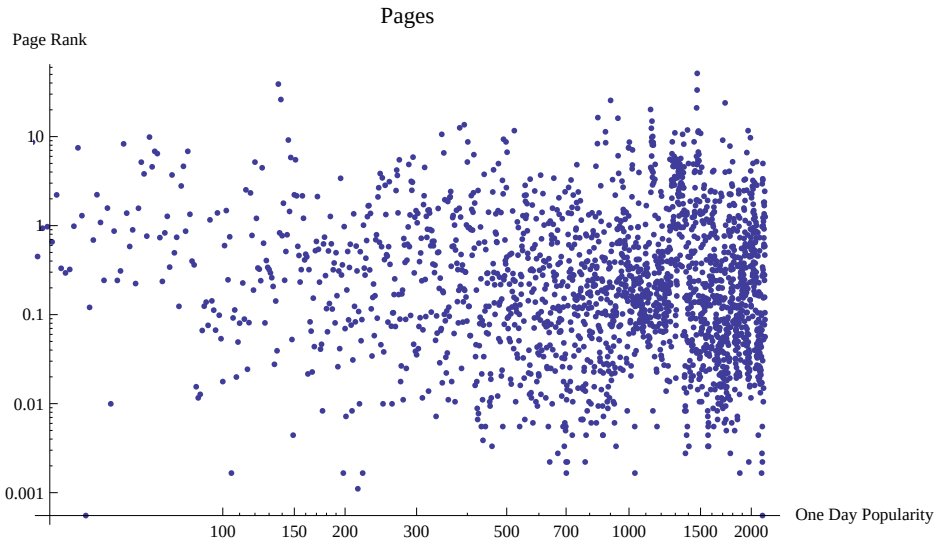


Page Rank VS One Day Popularity

Confront between One Day Popularity (June 27th 2013) and Page Rank of the Wikigraph.

```
OneDayPop = STPop[[All, 2, 27]];
```

```
ListLogLogPlot[{OneDayPop, STNormRanks}, PlotRange → All,
 AxesLabel → {"One Day Popularity", "Page Rank"}, PlotLabel → "Pages"]
```



Popularity Time Series

Useful Functions

Where? http : // stats.grok.se/json/en/

Given a list of Titles of Wikipedia Pages this function extract the time series for the chosen dates (in months), and print the results on a json file.

```
importingOnFile[dates_List,pages_,outputfile_]:=
Do[
    (Import["http://stats.grok.se/json/en/"<>#1<>"/"<>pages[[i]], "JSON"]>>>
        "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/"<>outputfile) & /@ dat
    ,
    {i,Length[pages]}]
(*dates_List = {"200910","200911","200910",..} in months, in crescent order, pages = l
```

Process and Plot Time Series

This function read from file the time series, knowing how many months.

```
readingTimeSeries[file_,months_]:= ReadList["/Users/Levantina/Documents/WOLFRAM/PROJEC
```

```
CleanTimeSeries[series_]:={FromDigits /@ StringSplit[#,"-"],#2}& @@@ series[[1,2]]
```

This function cleans imported data and makes them ready to be plotted.

```
ExtractTimeSeries[imported_List,Nmonths_Integer]:=Partition[CleanTimeSeries[#]& /@ imp
```

This function is useful to study the average behavior of a random sample of pages.

```
averageTimeSeries[cleanedTS_]:= N@Mean[Flatten[cleanedTS[[#,All,All,2]]]& /@ Range[1,L
```

This function plots the popularity for the selected Vertex:

```
PlotTimeSeries[series_,opts___]:= DateListPlot[series,Joined →True,opts]
```

```
ShowTimeSeries[index_,vertices_,imported_List,Nmonths_Integer]:=
    PlotTimeSeries[
        ExtractTimeSeries[imported,Nmonths][[index]],
            PlotRange→All,PlotLabel→vertices[[index]
        ]
    ]
```

This function correct the weekly fluctuation effect:

```
Corrections[daily_,timeseries_]:=
    Map[#/Flatten[Table[daily,{Round[(Length[#]/7. )+ 1]}]][[;;Length[#]]] &, timeseri
```

What about the Popularity behavior?

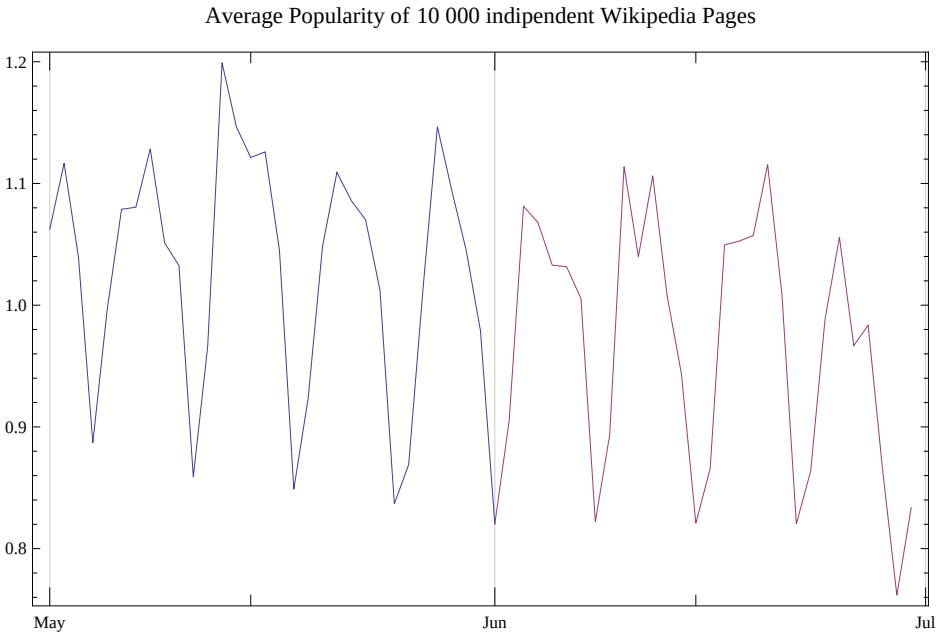
Download the Time Series

```
indipendentlinks = Flatten[Table[Import[
     "http://en.wikipedia.org/wiki/Special:Random", "Hyperlinks"], {10 000}]];

indipendentclean = Select[DeleteDuplicates[indipendentlinks],
   StringMatchQ[#, "http://en.wikipedia.org/wiki/" ~~ Except[":"] ..] &];

indipendentSet = RandomSample[indipendentclean, 10 000 ];

indipendentSet >>>
 "/Users/Levantina/Documents/WOLFRAM/PROJECT/inidpendentlinks.txt"

time = {"201305", "201306"};

indipendentlinkNameSet = Last @StringSplit[#, "/"] & /@ indipendentSet;

importingLinksOnFile[time, indipendentlinkNameSet, "bigrandomlinksseries.txt"];

bigGE = readingTimeSeries["bigrandomlinksseries.txt", 2];

cleanBigGE = ExtractTimeSeries[bigGE, 2];

averages = N@
   Mean[Flatten[cleanBigGE[[#, All, All, 2]]] & /@ Range[1, Length[cleanBigGE]]];

firstaverages = N@Mean[Flatten[cleanBigGE[[#, All, All, 2]]] & /@ Range[1, 5000]];

secondaverages =
  N@Mean[Flatten[cleanBigGE[[#, All, All, 2]]] & /@ Range[5000, 10 000]];

normfirst = firstaverages / N@Mean[firstaverages];

normsecond = secondaverages / N@Mean[secondaverages];

normaverages = averages / N@Mean[averages];

weeks = Partition[normaverages, 7];
```

10 000 independent Time Series: Weekly Periodicity

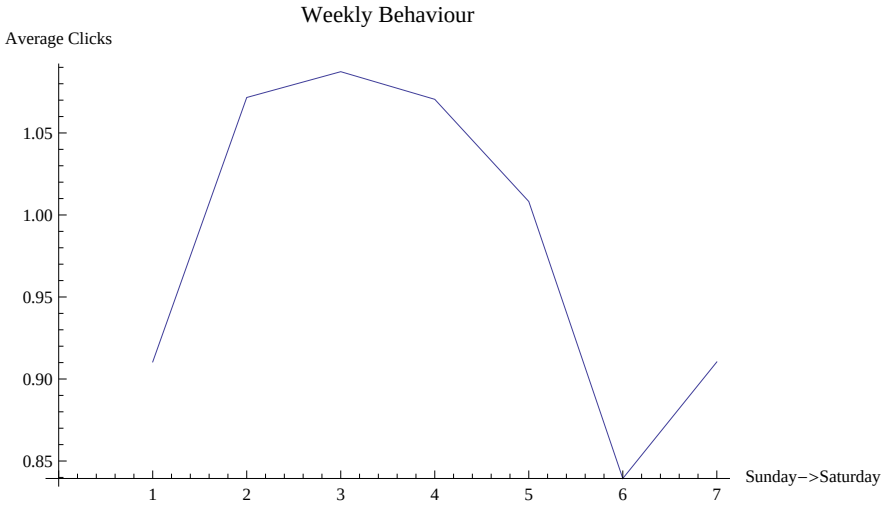
We obtain the average behavior per day in two months. What we see is that there is a general effect, due to a weekly periodicity, and we are going to quantify this fluctuation with 7 coefficients. With this coefficients we are going to reduce this effect over the data set that we are going to study. We can say that the Seasonal effect is negligible for a time window of two months, and we assume the effect to be constant over two months, and due to a daily behavior, as we can see in the graphics.

Two Months Average Behavior



Confirm of the Weekly Period and Coefficients

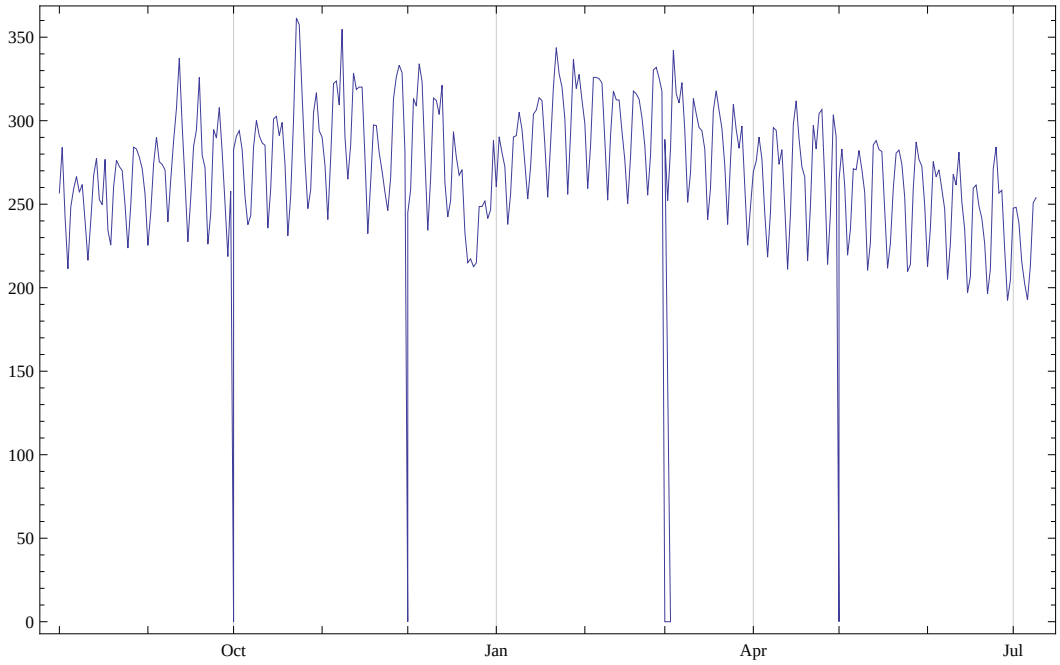
Weekly Average Behavior



1000 Average Time Series over a year

We can see that the average clicks decrease near Summer, and increase in Autumn. And also we can see a minimum near December/January holidays.

AVERAGE CLICKS PER DAY: 1000 Random Wikipedia Pages (from 2012−08 to 2013−07)



Popularity Time Series of our Data Set

Import Data

```
importedSTTimeSeries = readingTimeSeries["startrekTimeSeries.json", 2];
```

```
cleanedSTTimeSeries = ExtractTimeSeries[importedSTTimeSeries, 2];
```

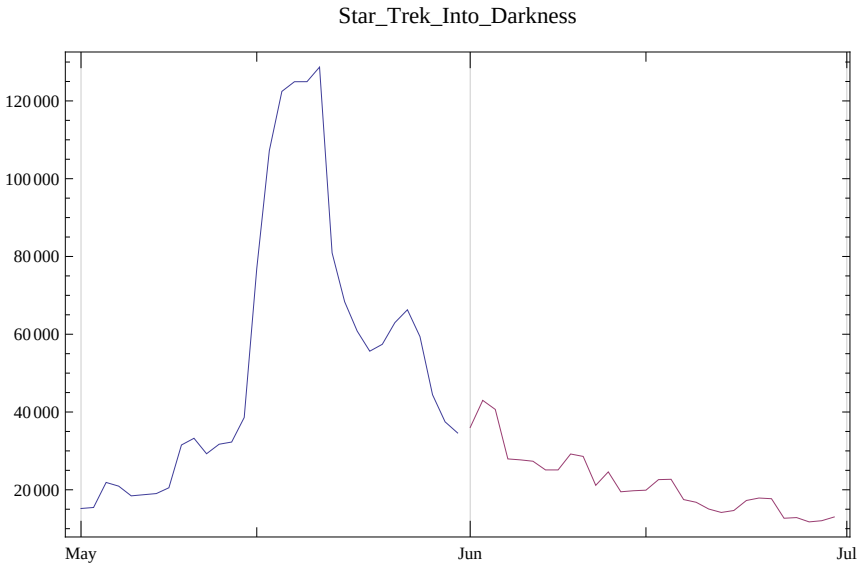
Interesting Event

The movie Star Trek Into Darkness reached a high peak of visitors on May 20th 2013, near the Release Date May 16th 2013.

```
Max[importedSTTimeSeries[[1, 1, 2, All, 2]]]
```

```
128 702
```

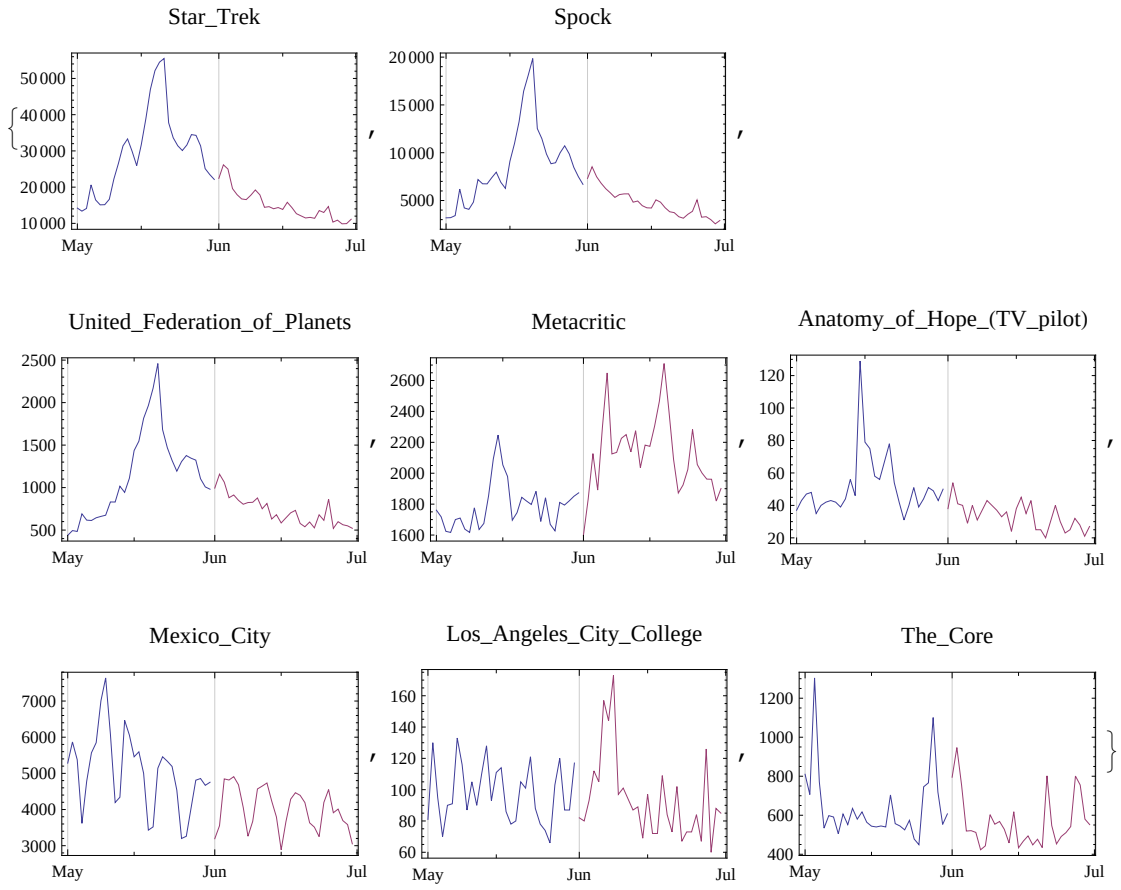
```
ShowTimeSeries[1, STVertices, importedSTTimeSeries, 2]
```



Time Series of the others vertices in the Wikigraph

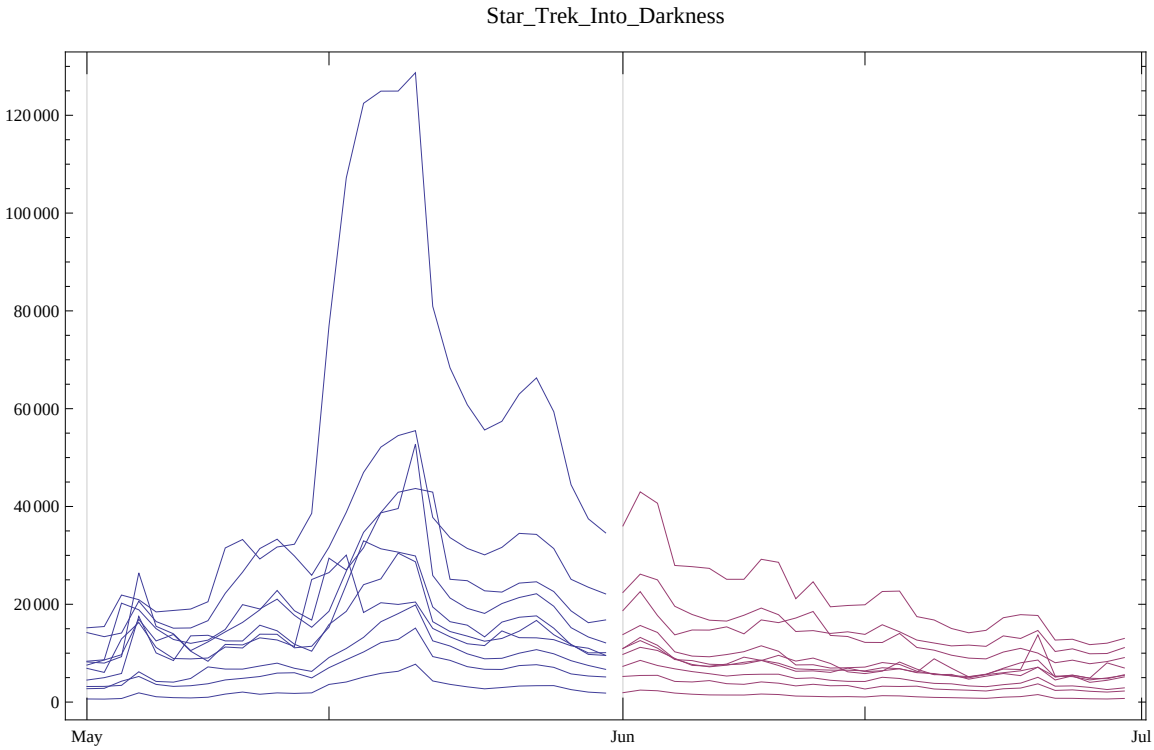
With the Vertex Index we can know its time series:

```
ShowTimeSeries[#, STVertices, importedSTTimeSeries, 2] & /@
 {7, 3    0, 40, 50, 100, 150, 200, 500}
```



Given an event, are Popularities Delayed in time?

Popularities for neighbors of “Star Trek Into Darkness”, “”Star Trek (film)”, “Star Trek”, “Spock”, ““James T. Kirk”. They don’t seem delayed in time. For this reason to study correlations we will use the Pearson Correlation.



Instant Behavior

Our resolution is of one day, we can’t see a diffusion in this behavior, we can’t consider a dynamic. And also we can’t study causal relations. We can see correlations, but we can’t say if the correlation between two pages exists because of the structure of the network, or because there is a causal connection between the two pages, beyond the network. This two mechanism probably coexist but we can’t see with these data if one is predominant.

Correlations between Time Series

Useful Functions

Fast Correlation

```
FastCorrelations[timeseries_]:=Module[{matrix,length},
    length=Length[timeseries[[1]]];
    matrix = If[
                StandardDeviation[#]>0, N[(#-Mean[#])/(StandardDeviation[#]*Sqrt[lengt
                Table[0.,{Length[#]}]] &/@timeseries
    matrix.Transpose[matrix]
]

FastCorrelations2[timeseries_]:= Module[
    {matrix, correlationmatrix, minlengthmatrix, trimmedtimeseries, timeserieslength},
    trimmedtimeseries = Cases[{#}, {Longest[Repeated[0., {0,Infinity}]],x:___} :> x] &
    timeserieslength = Length /@ trimmedtimeseries;
    trimmedtimeseries = If[
        StandardDeviation[#]>0,
        (# - Mean[#])/Sqrt[Mean[(#-Mean[#])^2]],
        Table[0., {Length[#]}]
    ]& /@ trimmedtimeseries;
    trimmedtimeseries = N@PadLeft[trimmedtimeseries];
    minlengthmatrix = Outer[Min, timeserieslength, timeserieslength];
    correlationmatrix = trimmedtimeseries.Transpose[trimmedtimeseries];
    correlationmatrix/minlengthmatrix
]

realLength[list_] := Length[Cases[{list}, {Longest[Repeated[0., {0,Infinity}]],x:___}

realCorrelation[list1_, list2_] := Module[
    {reallength},
    reallength = Min[realLength[list1], realLength[list2]];
    Correlation[Take[list1, -reallength], Take[list2, -reallength]]
];

NotFastCorrelations[timeseries_]:= Outer[realCorrelation, timeseries, timeseries, 1];
```

We can transform the correlation in a distance in the graph:
d(i,j) = Sqrt[2*(1-corr(i,j))]
d= {0 correlated, Sqrt[2] not correlated, 2 anti correlated}

```
DistancesFromCorrelationMatrix[matrix_]:= Sqrt[2*(1-#)] & /@ matrix
```

Adjacency Matrix

Put 0s on the diagonal (for Array Plot Visualization)

```
removingSelfLoops[length_]:=(-1)*(IdentityMatrix[length]-1)
```

Visualize with the function Graph, and put Infinity to disconnect vertices. Transform a matrix of multiplicity in a matrix with distances.

```
MyInversion[n_]:=If[n==0,Infinity,1./n]
SetAttributes[MyInversion,Listable]
```

```
TransformAdjacencyMatrixForGraph[multiplicityMatrix_]:= MyInversion[multiplicityMatrix
```

```
InfiniteDiagonal = Map[ If[#==1,Infinity,1]& ,IdentityMatrix[65],{2}];
```

Extract the adjacency matrix from a a list of weighted edges, using the listVertices order.

```
ExtractWeightedAjdacencyMatrix2[listWeightedEdges_, listVertices_]:=
Module[{position, length},
        length= Length[listVertices];
        (position=Position[StarTrekNet,{STVertices[[#[[1]]]],STVertices[[#[[2]]]],__}]
            If[Length[position]≠0,StarTrekNet[[position[[1,1]],3]],0]) &/@ Tuples[Rang
```

```
ExtractWeightedAjdacencyMatrix[listWeightedEdges_, listVertices_]:= Module[
    {f, adjmatrix},
    MapIndexed[Set[f[#1], Sequence@@#2 ] &, listVertices];
    adjmatrix=ConstantArray[0, {Length[listVertices], Length[listVertices]}];
    Set[adjmatrix[[f[#1], f[#2]]] , #3] & @@@ listWeightedEdges;
    adjmatrix
]
```

```
mat = ExtractWeightedAjdacencyMatrix[ StarTrekNet, STVertices];
```

Evaluation with the daily correction

Daily correction for Time Series

We noticed a weekly fluctuation of the visitors on Wikipedia, we found a weekly periodicity. We have a correction for each day of the week. And during the evaluation of the correlations we will consider this 7 coefficients. In order to reduce the overestimating of the correlations.

```
averagesDaily = {1.0861167208459364`,
  1.070509192186192`, 1.0082207459429013`, 0.8393580386204945`,
  0.910410111634681`, 1.0716210490409843`, 1.0873540902600047`}
```

```
{1.08612, 1.07051, 1.00822, 0.839358, 0.91041, 1.07162, 1.08735}
```

```
rightSTPop = Flatten[cleanedSTTimeSeries[[#, ;; 2, All, 2]]] & /@
   Range[1, Length[cleanedSTTimeSeries[[All, ;; 2, All, 2]]]];
```

```
Corrections[daily_,timeseries_]:=
Module[{days=Length[timeseries[[1]]]},
    Map[#/Flatten[Table[daily,{Round[days/7]+1}]][[;;Length[#]]] &, timeseries,{1}]
]
```

```
PopularityNormal = rightSTPop;
```

```
PopularityCorrected = Corrections[averagesDaily, rightSTPop];
```

Pearson Correlation with Popularity and Corrected Popularity

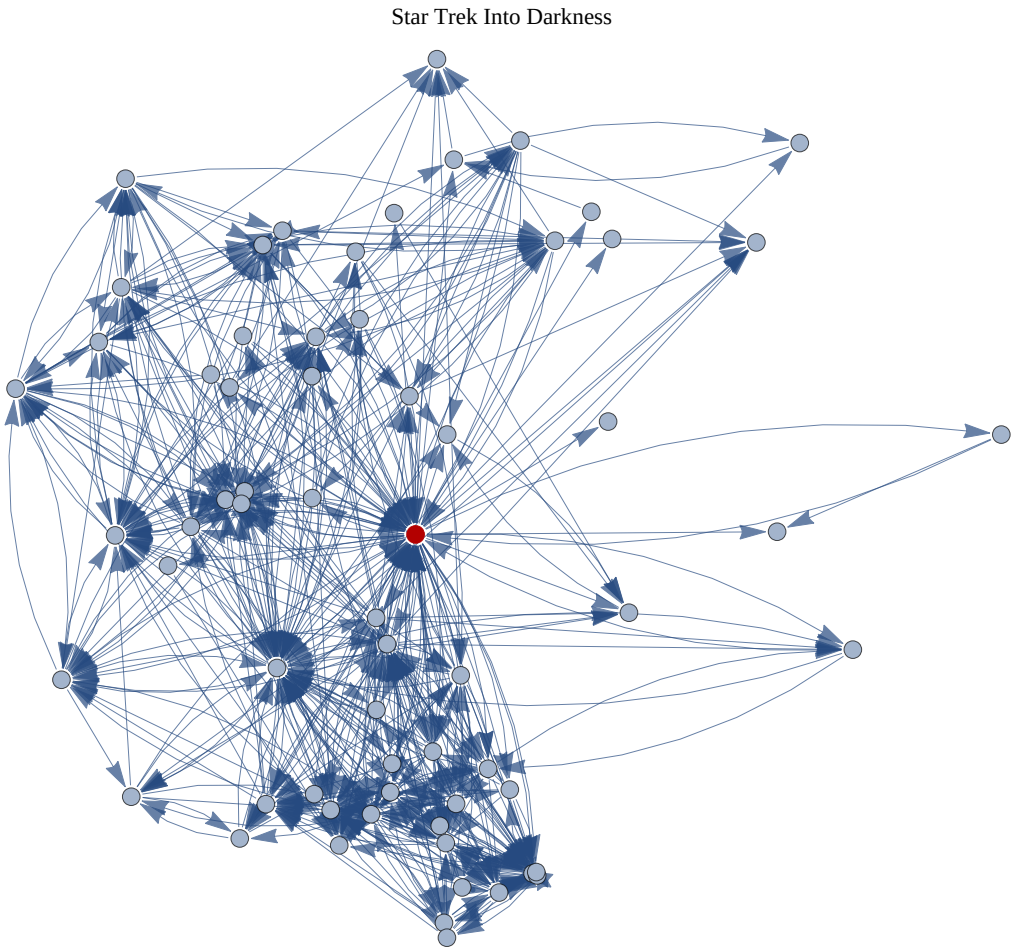
Confront Correlations

How can we reproduce the real connections in the Graph?

The Neighborhood Graph of ‘Star Trek Into Darkness’: first neighbors.

The Real Sub-Graph

This is what we want to reproduce.



Data

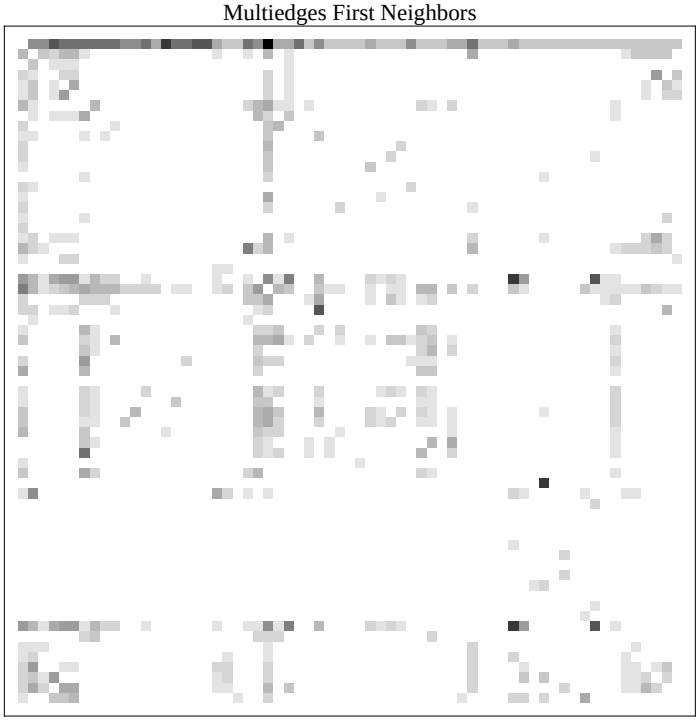
The Real Adjacency Matrix

We build the adjacency matrix of the graph (multiplicity of links will be a measure of their strength):

```
distanceMatrixCorrected =
  Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/
     distanceMaxtrixCorrected.tsv", "TSV"];

distanceMatrix65 = distanceMatrixCorrected[[ ;; 65, ;; 65]];
```

This matrix represent the links that we want to reproduce.

Multiedges First Neighbors



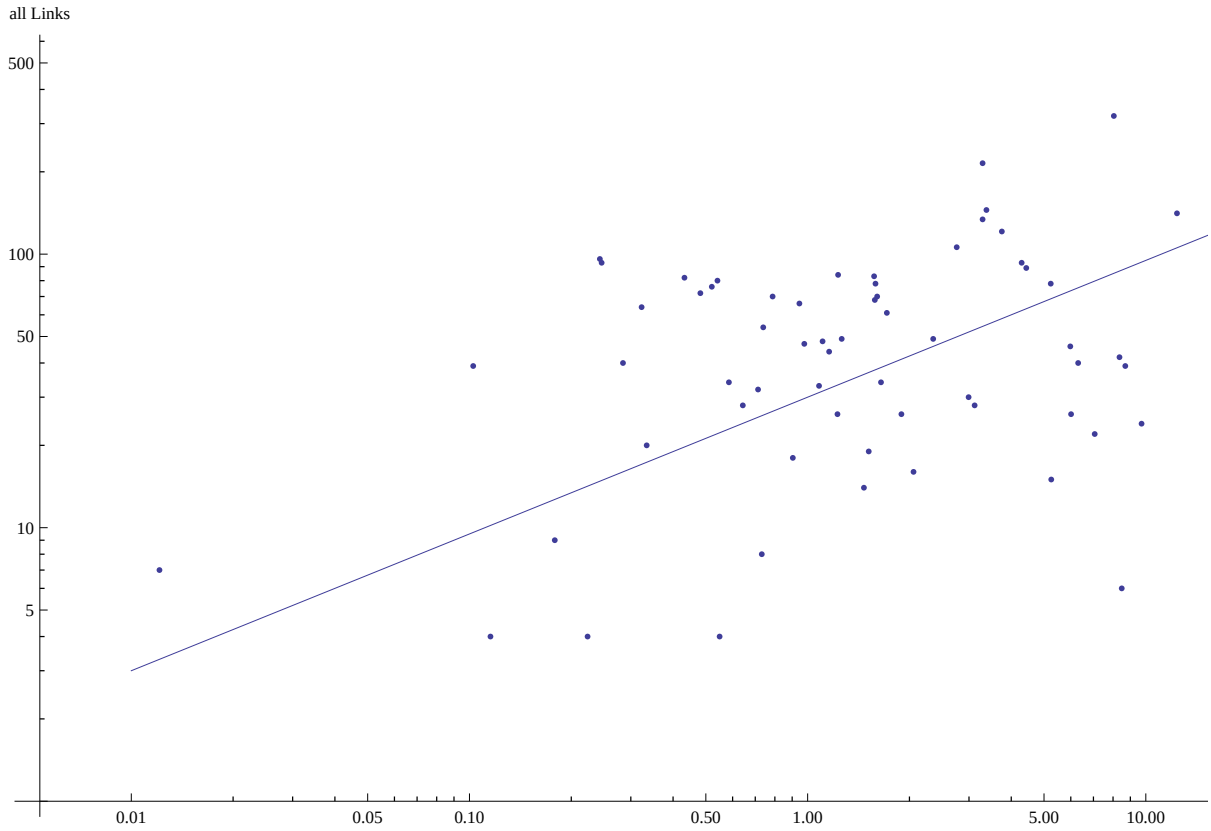
Total Number of Real Links VS Popularity

```
AdjMatr65 = Import[
    "/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/adjMatrix65.tsv"
    , "TSV"];
```

```
AdjMatr65 = Partition[Flatten[AdjMatr65], 65]; allLinks =
 (Plus @@ AdjMatr65[[All, #]] + Plus @@ AdjMatr65[[#, All]] ) & /@ Range[1, 65];
Max[allLinks]
```

```
meansSTPopNorm = Flatten@Import[
     "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meansSTPopNorm.tsv"
     , "TSV"];
```

I find an interesting behavior: Links(v) ~ Popularity(v)^0.5

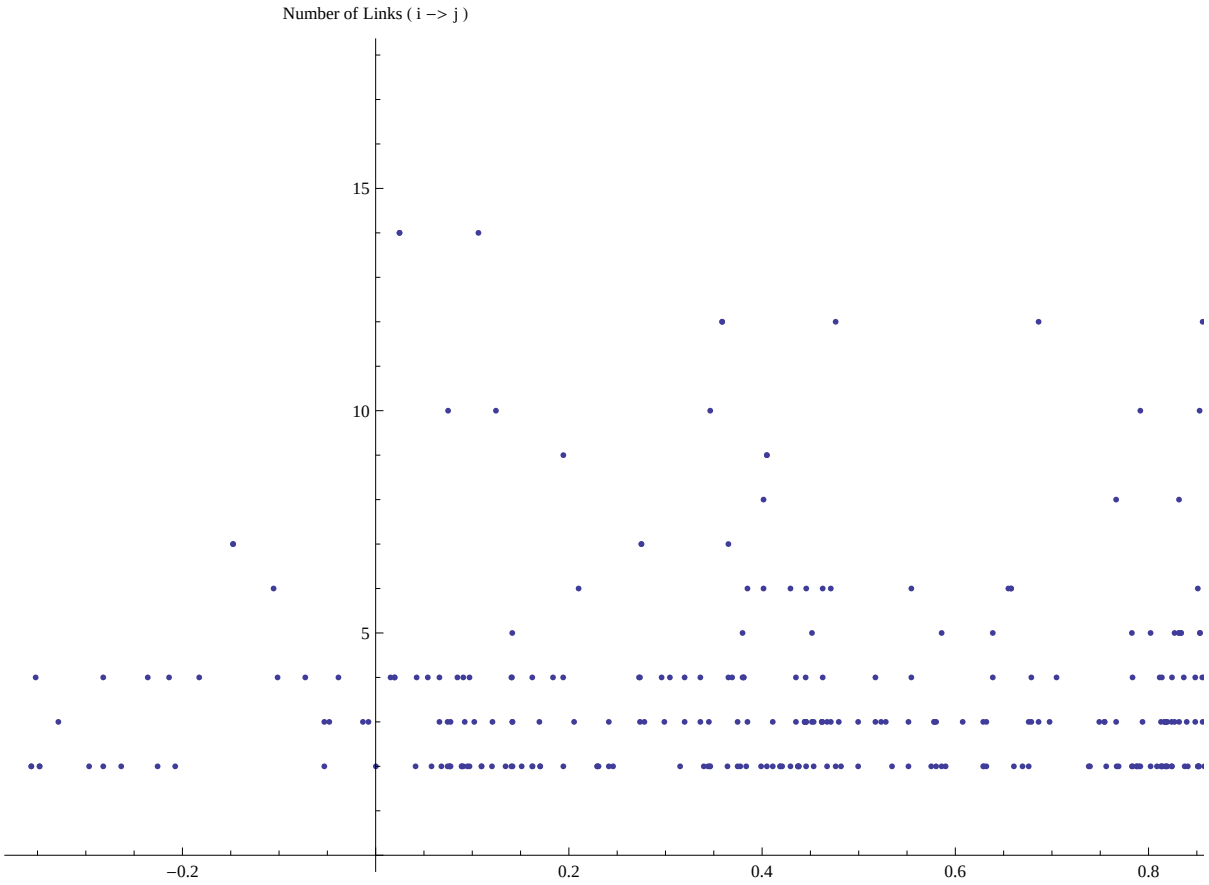


We are going to consider this behavior to predict the number of links that connect two pages.

Number of Real Links VS Correlation

```
corrMatrix = Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/
     CorrelationsSTCorrectedTimeSeries.tsv", "TSV"];

corrMatrixNoSelfCorr =
  corrMatrix * (-1) * (IdentityMatrix[Length[First[corrMatrix]]] - 1);

corrMatrix65 = corrMatrixNoSelfCorr[[ ;; 65, ;; 65]];

corrMatrix65 = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/corrMatrix65.tsv",
   "TSV"];

meansSTPopNorm = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meanSTPopNorm2Months.
     tsv", "TSV"];

Show[Histogram[Cases[
   Transpose[{Flatten[Round[100 * corrMatrix65] / 100.], Flatten[AdjMatr65]}],
   Except[{0., 0}]], PlotRange → All], LogLogPlot[1.5 + 5. x^0.7, {x, 0.001, 2}]]
```

```
$Aborted
```



From Correlations to a Graph

All Correlations Matrix

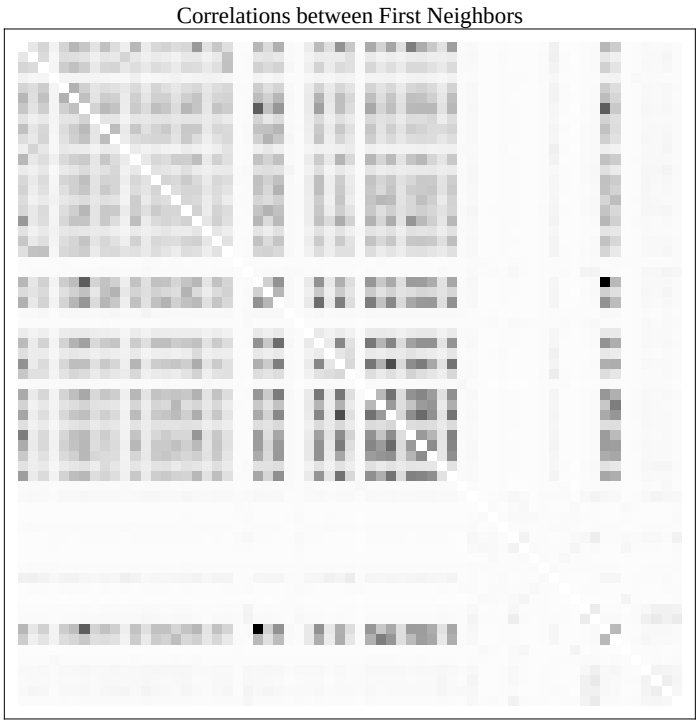
We build the adjacency matrix with all the distances d between the vertices (a measure of the correlation):

d = (1 - corr [i , j])
corr [i , j] = 1 - d

```
distanceMatrixCorrected =
  Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/
     distanceMaxtrixCorrected.tsv", "TSV"];

distanceMatrix65 = distanceMatrixCorrected[[ ;; 65, ;; 65]];
```

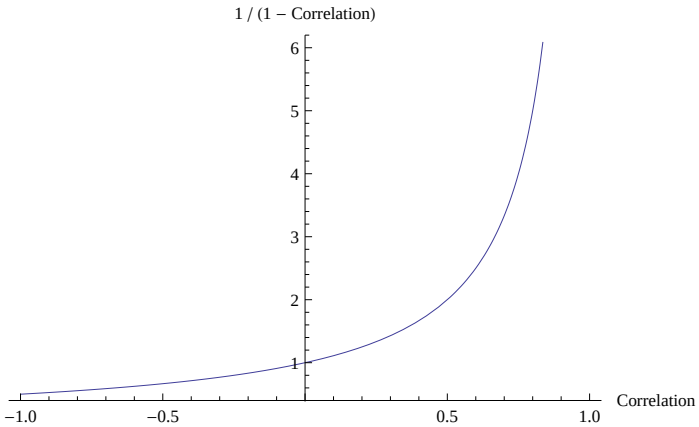
How the “correlation” matrix look like? We consider:
Multiplicity (i , j) ~ 1 / (1 - corr [i , j])

```
ArrayPlot[1. / distanceMatrix65 * removingSelfLoops,
 PlotLabel -> "Correlations between First Neighbors"]
```



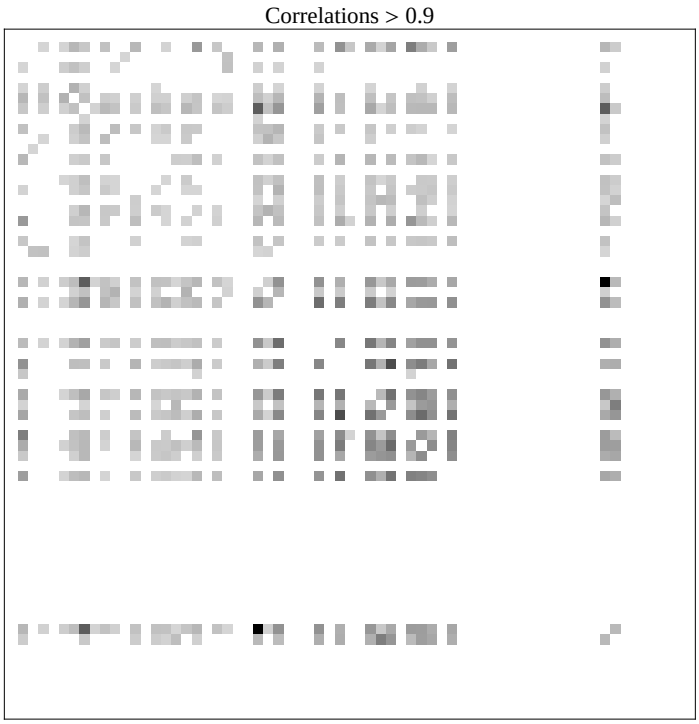
Correlation Matrix with a Threshold

```
Plot[1. / (1 - x), {x, -1., 1},
 AxesLabel → {"Correlation", "1 / (1 - Correlation)"}]
```




```
thresholdInverseDistanceMatrix65 =
  Map[If[# < 10, 0, #] &, (1. / distanceMatrix65) * removingSelfLoops, {2}];
```

```
ArrayPlot[thresholdInverseDistanceMatrix65, PlotLabel → "Correlations > 0.9"]
```



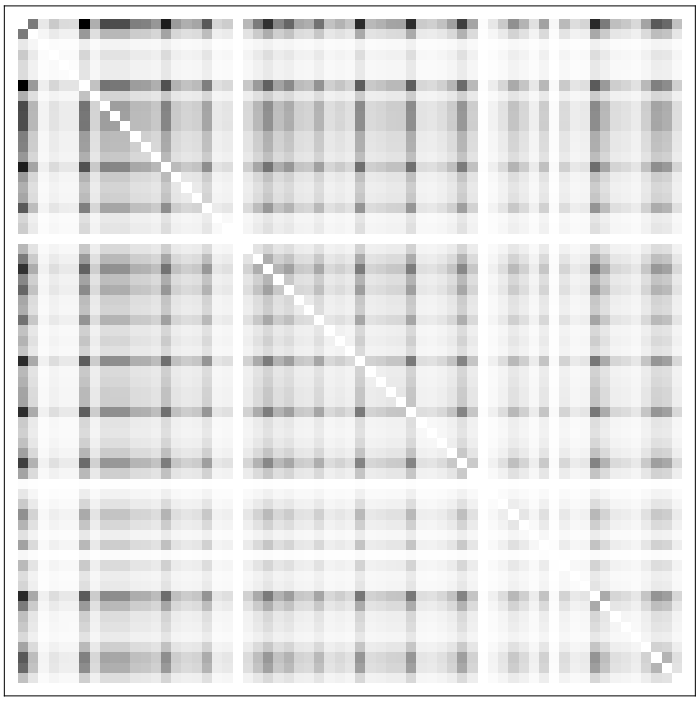
Matrix of the Square Root of the Popularity Product between each Page

How the popularity matrix look like? If we consider:

Multiplicity (i , j) ~ [Pop (i) * Pop (j)] ^ 0.5

```
PopMatrixSqrt =
  Outer[Sqrt[#1 #2] &, meansSTPopNorm[[ ;; 65]], meansSTPopNorm[[ ;; 65]]];
```

```
ArrayPlot[PopMatrixSqrt * removingSelfLoops]
```



Link Prediction

From the correlation between two pages and their popularity value, can we say if there is a link or not? We are going to predict the multiplicity of a link, and we are going to assign a error to this prediction.

Distance [u , v] = (1 - corr (u , v))

Adjacency Matrix: 1 / Distances

Multiplicity (u , v) ~ [1 / (1 - corr (u , v)]^a [Pop (u)]^b [Pop (v)]^c

Functions

Prediction Rules

```
LinkPrediction[d_,popi_,popj_]:= Module[
    {DistThreshold = 20., PopThreshold = 90.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        0.5*d*Sqrt[popi*popj],
        0.
    ]
]
```

```
LinkPredictionLeft[d_,popi_,popj_]:= Module[
    {DistThreshold = 25., PopThreshold = 80.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        0.5*d*Sqrt[popi],
        0.
    ]
]
```

```
LinkPredictionRight[d_,popi_,popj_]:= Module[
    {DistThreshold = 30., PopThreshold = 90.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        0.5*d*Sqrt[popj],
        0.
    ]
]
```

```
LinkPredictionSquare[d_,popi_,popj_]:= Module[
    {DistThreshold = 16., PopThreshold = 80.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        0.5*Sqrt[d]*Sqrt[popi*popj],
        0.
    ]
]
```

```
LinkPredictionSquareLeft[d_,popi_,popj_]:= Module[
    {DistThreshold = 14., PopThreshold = 60.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        0.6*Sqrt[d]*Sqrt[popi],
        0.
    ]
]
```

```
LinkPredictionSquareRight[d_,popi_,popj_]:= Module[
    {DistThreshold = 14., PopThreshold = 60.},
    If[d > DistThreshold || popi*popj>PopThreshold,
        1.*Sqrt[d]*Sqrt[popj],
        0.
    ]
]
```

Graph Prediction

```
GraphPrediction[adjMatrix_,popularities_,DistThreshold_,PopThreshold_]:=Map[LinkPredic
popularities[[#[[1]]]],popularities[[#[[2]]]],DistThreshold,PopThreshold]&,
Table[{i,j},{i,1,Length[adjMatrix[[1]]]},{j,1,Length[adjMatrix[[1]]]},{2}];
```

```
GraphPrediction1[correlationMatrix_, popularities_] := Module[
{predictedAdjencyMatrix, a, threshold},
    a = .05;
    threshold = 1.;
    predictedAdjencyMatrix = a/(1.-correlationMatrix)*KroneckerProduct[popularities^0.
    Threshold[predictedAdjencyMatrix, threshold]
];
```

```
GraphPrediction2[correlationMatrix_, popularities_] := Module[
{predictedAdjencyMatrix, a, threshold},
    a = 5;
    threshold = 10.;
    predictedAdjencyMatrix = a/(1.-correlationMatrix)*Sqrt[popularities];
    Threshold[predictedAdjencyMatrix, threshold]
];
```

```
GraphPrediction3[correlationMatrix_, popularitiesMatrix_] := Module[
{predictedAdjencyMatrix, a, threshold},
    a = 5;
    threshold = 20;
    predictedAdjencyMatrix = a/(1.-correlationMatrix)*Sqrt[popularitiesMatrix];
    Threshold[predictedAdjencyMatrix, threshold]
];
```

Error Measure

```
predictionError[realMatrix_, predictedMatrix_] := Sqrt[Mean[Flatten[predictedMatrix-re

predictionError2[realMatrix_, predictedMatrix_] := Sqrt[Mean[Flatten[(predictedMatrix-
```

Test The Prediction Rules

This prediction rules consider Correlation and Popularity of the pages. If two pages are strongly correlated we put a weighted link between them, proportional to their proximity and the square root of their popularity. Otherwise, if they are not strongly correlated, we consider just their popularity, if their popularity is large, we put a weighted link between them.

```
corrMatrix = Import["/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/
     CorrelationsSTCorrectedTimeSeries.tsv", "TSV"];
```

```
corrMatrixNoSelfCorr =
  corrMatrix * (-1) * (IdentityMatrix[Length[First[corrMatrix]]] - 1);
```

```
corrMatrix65 = corrMatrixNoSelfCorr[[ ;; 65, ;; 65]];
```

```
corrMatrix65 = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/corrMatrix65.tsv",
   "TSV"];
```

```
meansSTPopNorm = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meansSTPopNorm.tsv",
   "TSV"];
```

```
Dimensions[corrMatrix65]
```

```
{65, 65}
```

```
Length[meansSTPopNorm]
```

```
2164
```

```
Export[
  "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meanSTPopNorm65.tsv",
  meansSTPopNorm[[ ;; 65]], "TSV"];
```

```
/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meanSTPopNorm65.tsv
```

```
Clear[GraphPrediction1]
```

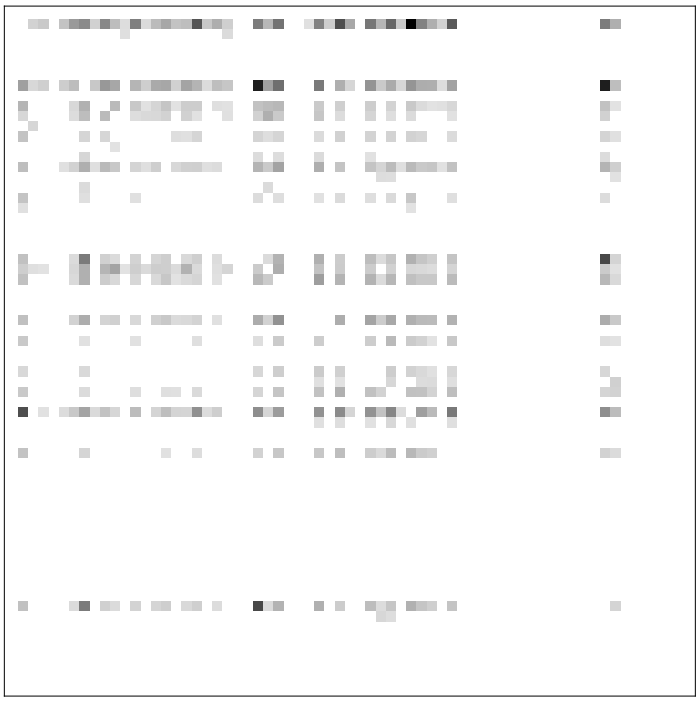
```
predicted = GraphPrediction1[corrMatrix65, Flatten@meansSTPopNorm[[ ;; 65]]];
```

```
predictionError[AdjMatr65, predicted]
```

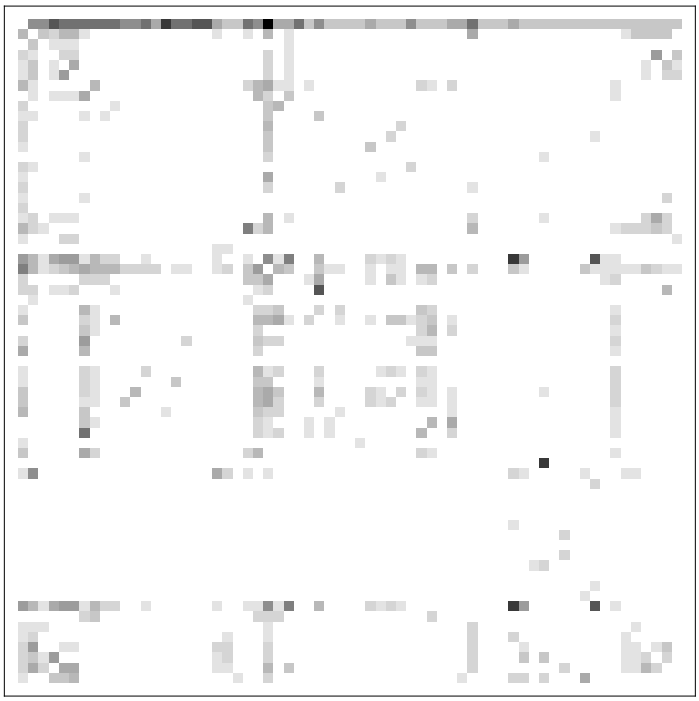
```
Mean[Flatten[predicted - AdjMatr65]]
```

```
-0.117821
```

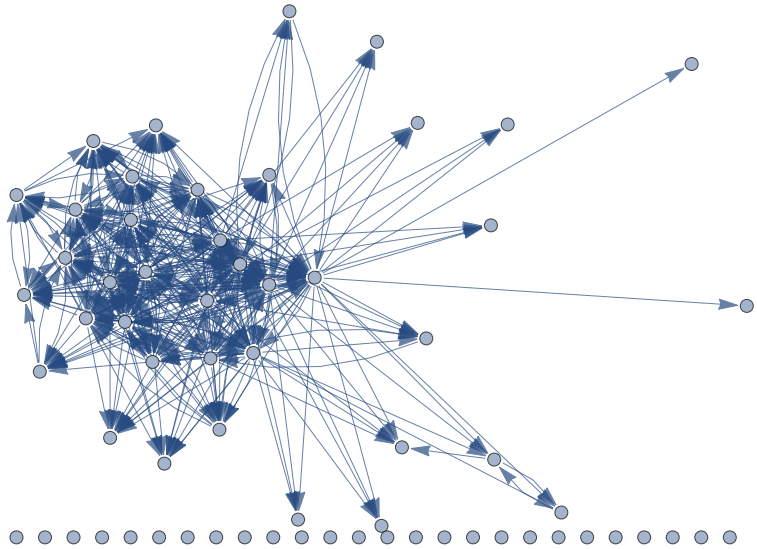
```
ArrayPlot[predicted]
```



```
ArrayPlot[AdjMatr65]
```



```
ArrayPlot[predicted - AdjMatr65, ColorFunction → "LakeColors"]
```



Live Prediction

Prediction Rule

```
LinkPrediction1[i_,j_,correlationMatrix_, popularities_] := Module[
{predictedLink, a, threshold},
    a = .05;
    threshold = 1.;
    predictedLink = a/(1.-correlationMatrix[[i,j]])*popularities[[i]]^0.5*popularities
    Threshold[predictedLink, threshold]
];
```

Data

```
corrMatrix65 = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/corrMatrix65.tsv",
   "TSV"];
meansSTPopNorm65 = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/Timeseries/meanSTPopNorm65.tsv",
   "TSV"];
AdjMatr65 = Import[
   "/Users/Levantina/Documents/WOLFRAM/PROJECT/startrekNetwork/adjMatrix65.tsv"
   , "TSV"];
AdjMatr65 = Partition[Flatten[AdjMatr65], 65];
```

Prediction

```
STVertices[[ ;; 20]]
```

```
{Star_Trek_Into_Darkness, J._J._Abrams, Bryan_Burk,
 Damon_Lindelof, Alex_Kurtzman, Roberto_Orci, Star_Trek,
 Gene_Roddenberry, Chris_Pine, Zachary_Quinto, Zoe_Saldana,
 Karl_Urban, Simon_Pegg, John_Cho, Benedict_Cumberbatch, Anton_Yelchin,
 Bruce_Greenwood, Peter_Weller, Alice_Eve, Michael_Giacchino}
```

```
LinkPrediction1[1, 12, corrMatrix65, meansSTPopNorm65]
```

```
{4.3823}
```

```
AdjMatr65[[1, 12]]
```

```
8
```

Refine the Wikipedia Graph