A Song of Ice and Fire and Graph Theory

Introduction

Graph theory can help you understand the social landscape of the books in the series *A Song of Ice and Fire* by G. R. R. Martin. With the judicious use of Maple, you can ask yourself *really* pressing questions like

- Who is the most influential person in Westeros? How has their influence changed over the five books?
- How are Eddard Stark and Randyll Tarly connected?
- What do eigenvectors have to do with the battle for the Iron Throne, anyway?

This application uses data from the books. See Game of Thrones and Graph Theory to ask these types of questions using data for the television series.

The data attached to this workbook (originally found at the reference below) can be used to create a character interaction network for each of the five books in the series.

- Each character is a vertex.
- Two vertices are connected with an edge if the two characters appear or are referenced within 15 words of each other (this is one interaction).
- The weight of the edge connecting two vertices is the number of interactions.

You can use Maple's GraphTheory package to generate an undirected, unweighted graph from this data. You can then analyze the graph - for example, rank characters in terms of importance, visualize the relationship between the characters, determine clustering and more.

Character influence is measured by three metrics.

- Degree centrality is the simplest measure of influence, and is number of connections a vertex has.
- Betweenness centrality measures how often a vertex lies on the shortest path between two other vertices (this metric can be weighted by the number of interactions). This indicates the influence of a vertex in assisting the flow of information, assuming that

information travels along the shortest path.
- Eigenvalue centrality uses the number and importance of neighboring vertices to quantify influence.

This application is inspired by the work of Andrew Beveridge, who generated the character interaction data and authored these websites:

- Data repository: https://github.com/mathbeveridge/asoiaf (this website contains vertex-edge data for the books and the television adaptation)
- Main website: https://networkofthrones.wordpress.com/the-novels/

Packages and Predefined Graph Styles

```
> restart:
  with(GraphTheory):
  with(Statistics):
  with(ColorTools):
  with(LinearAlgebra):
  with(ArrayTools):

> style_1 := [
   vertexcolor     = Color("RGB", [0/255, 79/255, 128/255])
  ,vertexfontcolor = white
  ,vertexborder    = false
  ,edgethickness   = 0
  ,edgecolor       = Color("RGB",[191/255, 191/255, 191/255])
  ,vertexshape     = "stadium"
  ,vertexfont      = ["Arial", 12]
  ,vertexpadding   = 12
  ,edgecolor       = Color("RGB", [150/255, 40/255, 27/255])]:

  style_2 := [
   vertexcolor     = Color("RGB", [0/255, 79/255, 128/255])
  ,vertexfontcolor = white
  ,vertexborder    = false
  ,edgethickness   = 0
  ,edgecolor       = Color("RGB",[191/255, 191/255, 191/255])
  ,vertexshape     = "stadium"
  ,vertexfont      = [Arial, 18]]:

  style_3 := [
   color            = Color("RGB", [150/255, 40/255, 27/255])
  ,fontcolor        = white
  ,border           = false
  ,shape            = "stadium"
  ,font             = [Arial, 20]]:
```

Import Data and Generate Overall Graph

To analyze different books, change the filenames in the ImportMatrix command below.

A Game of Thrones	book1
A Clash of Kings	book2
A Storm of Swords	book3
A Feast for Crows	book4
A Dance with Dragons	book5

For example, to analyze A Feast for Crows, change the file names to **asoiaf-book4-nodes.csv** and **asoiaf-book4-edges.csv**.

```
> node_data := ImportMatrix("this://asoiaf-book1-nodes.csv",
  skiplines = 1);
  edge_data := ImportMatrix("this://asoiaf-book1-edges.csv",
  skiplines = 1);

  num_edges      := RowDimension(edge_data);
  num_characters := RowDimension(node_data);
```

node_data :=

"Addam-Marbrand"	"Addam Marbrand"
"Aegon-I-Targaryen"	"Aegon I Targaryen"
"Aemon-Targaryen-(Maester-Aemon)"	"Aemon Targaryen (Maester Aemon)"
"Aerys-II-Targaryen"	"Aerys II Targaryen"
"Aggo"	"Aggo"
"Albett"	"Albett"
"Alliser-Thorne"	"Alliser Thorne"
"Alyn"	"Alyn"
"Arthur-Dayne"	"Arthur Dayne"
"Arya-Stark"	"Arya Stark"
⋮	⋮

187 × 2 Matrix

edge_data :=

"Addam-Marbrand"	"Jaime-Lannister"	"Undirected"	3	1
"Addam-Marbrand"	"Tywin-Lannister"	"Undirected"	6	1
"Aegon-I-Targaryen"	"Daenerys-Targaryen"	"Undirected"	5	1
"Aegon-I-Targaryen"	"Eddard-Stark"	"Undirected"	4	1
"Aemon-Targaryen-(Maester-Aemon)"	"Alliser-Thorne"	"Undirected"	4	1
"Aemon-Targaryen-(Maester-Aemon)"	"Bowen-Marsh"	"Undirected"	4	1
"Aemon-Targaryen-(Maester-Aemon)"	"Chett"	"Undirected"	9	1
"Aemon-Targaryen-(Maester-Aemon)"	"Clydas"	"Undirected"	5	1
"Aemon-Targaryen-(Maester-Aemon)"	"Jeor-Mormont"	"Undirected"	13	1
"Aemon-Targaryen-(Maester-Aemon)"	"Jon-Snow"	"Undirected"	34	1
⋮	⋮	⋮	⋮	⋮

684 × 5 Matrix

$$num\_edges := 684$$

$$num\_characters := 187 \quad \textbf{(3.1)}$$

```
> G := Graph(node_data[.., 1], weighted);
```

$$G := Graph\ 1:\ an\ undirected\ weighted\ graph\ with\ 187\ vertices\ and\ 0\ edge(s) \quad \textbf{(3.2)}$$

```
> for i from 1 to num_edges  do
    AddEdge(G, [{edge_data[i, 1], edge_data[i, 2]}, edge_data[i,
  4]]):
  end do:
```

Visualize the overall character interaction network (this visualization needs to be big because of the shear number of vertices).

```
> DrawGraph(G, size = [2000, 2000], style = spring, showweights =
  true, showlabels = true, redraw = false, stylesheet = style_2)
```

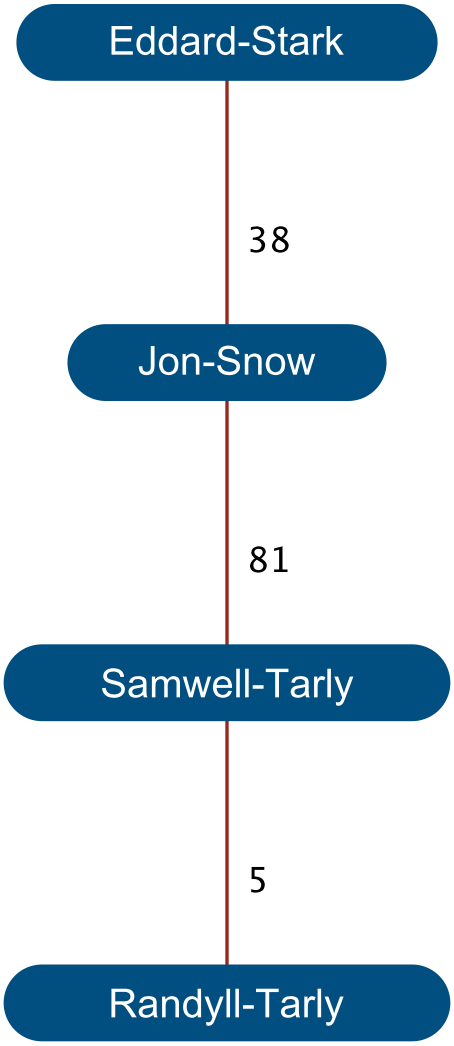
How Are Two Characters Connected?

This is the shortest path between two characters

```
> sp_char := ShortestPath(G, "Eddard-Stark", "Randyll-Tarly");
  sp := InducedSubgraph(G, sp_char):
```

sp_char := ["Eddard-Stark", "Jon-Snow", "Samwell-Tarly", "Randyll-Tarly"] **(4.1)**

```
> DrawGraph(sp, stylesheet = style_1)
```



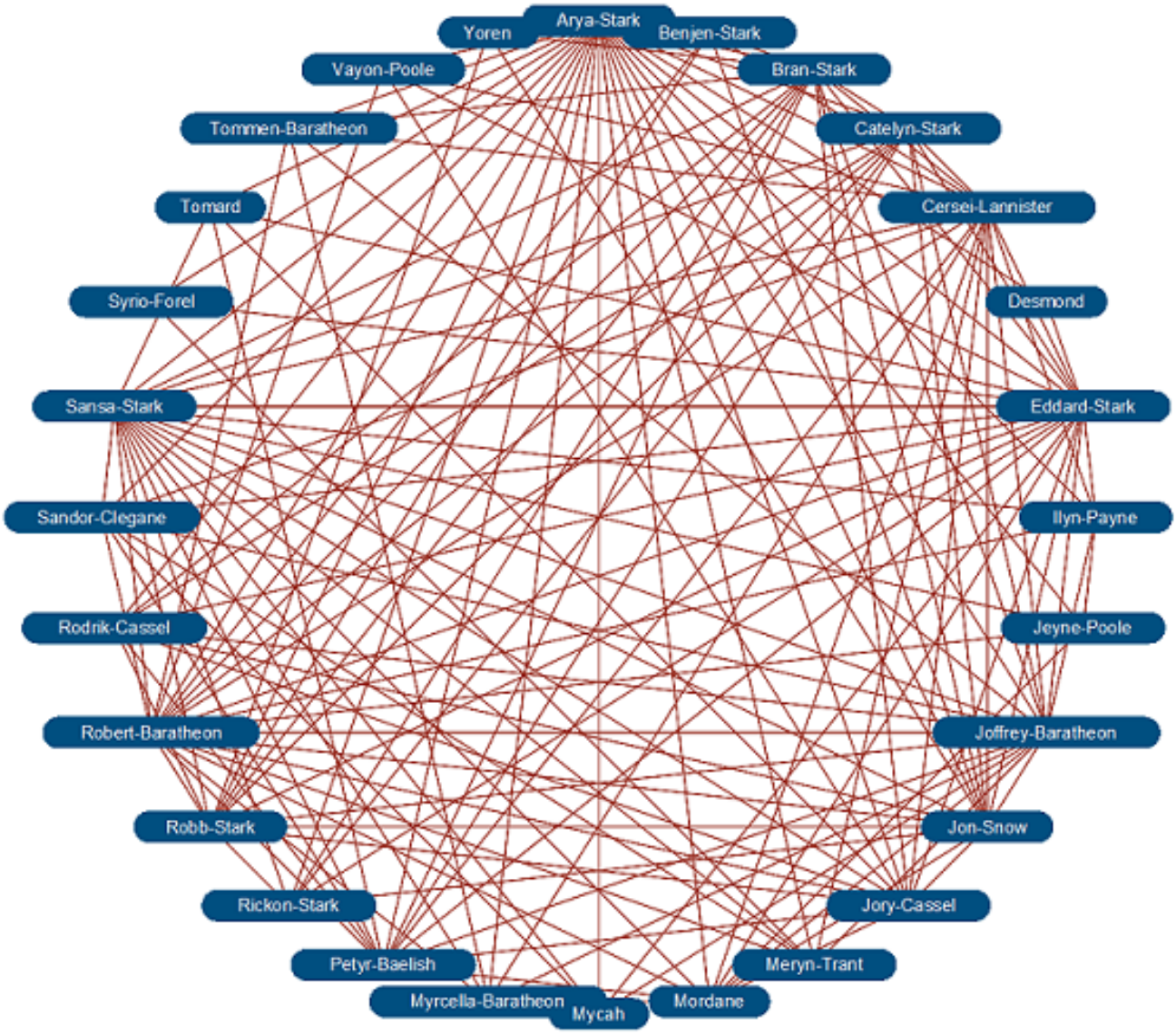
Character Neighborhoods

A neighborhood is a group of vertices that connected to a named vertex - in this case, Arya Stark.

```
> Arya_nhood := Neighborhood(G, "Arya-Stark", closed)
```

Arya_nhood := ["Arya-Stark", "Benjen-Stark", "Bran-Stark", "Catelyn-Stark", "Cersei-Lannister", "Desmond", "Eddard-Stark", "Ilyn-Payne", "Jeyne-Poole", "Joffrey-Baratheon", "Jon-Snow", "Jory-Cassel", "Meryn-Trant", "Mordane", "Mycah", "Myrcella-Baratheon", "Petyr-Baelish", "Rickon-Stark", "Robb-Stark", "Robert-Baratheon", "Rodrik-Cassel", "Sandor-Clegane", "Sansa-Stark", "Syrio-Forel", "Tomard", "Tommen-Baratheon", "Vayon-Poole", "Yoren"] **(5.1)**

```
> DrawGraph(InducedSubgraph(G, Arya_nhood), stylesheet = style_1,
  size = [1000, 1000], style = circle, thickness = 0, showweights
  = false)
```



Cliques

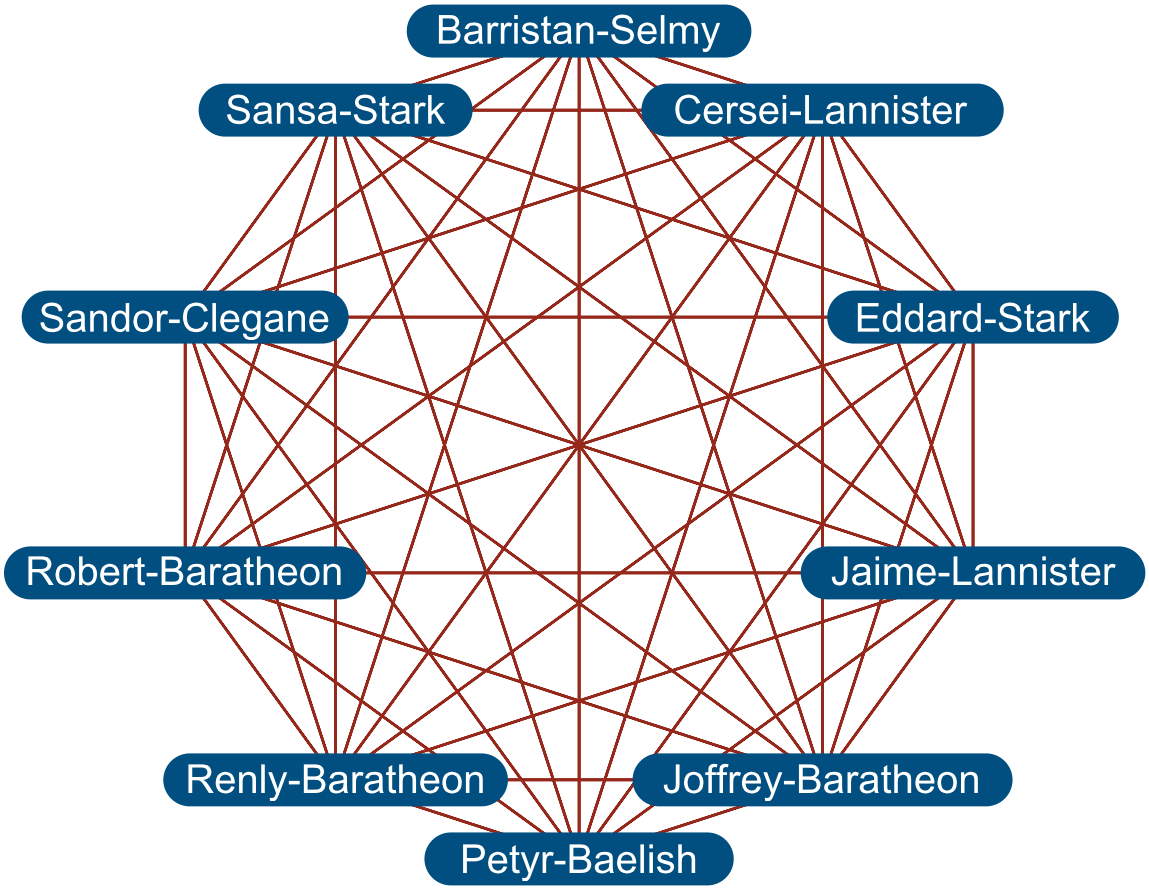
A clique is a group of vertices that are connected to every other vertex in the group

Maximum Clique

```
> clique_1   := MaximumClique(UnderlyingGraph(G));
  cliqueSubgraph_1 := InducedSubgraph(G, clique_1):
```

clique_1 := ["Barristan-Selmy", "Cersei-Lannister", "Eddard-Stark", "Jaime-Lannister", "Joffrey-Baratheon", "Petyr-Baelish", "Renly-Baratheon", "Robert-Baratheon", "Sandor-Clegane", "Sansa-Stark"] **(6.1.1)**

```
> DrawGraph(cliqueSubgraph_1, stylesheet = style_1, style =
  circle, size = [500, 500], showweights = false)
```

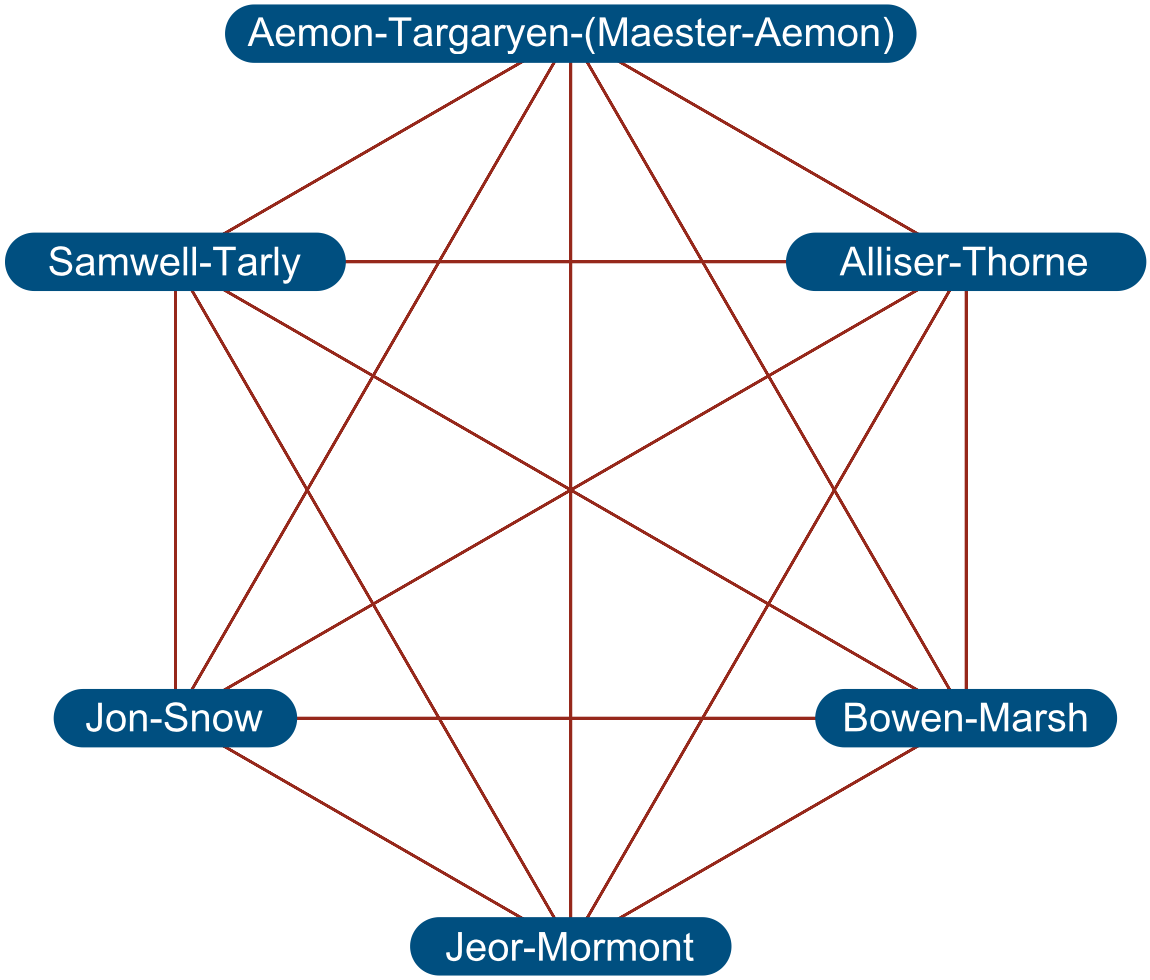


Clique of a Specific Size

```
> clique_2  := FindClique(UnderlyingGraph(G), 6);
  cliqueSubgraph_2 := InducedSubgraph(G, clique_2):
```

clique_2 := ["Aemon-Targaryen-(Maester-Aemon)", "Alliser-Thorne", "Bowen-Marsh", "Jeor-Mormont", "Jon-Snow", "Samwell-Tarly"] **(6.2.1)**

```
> DrawGraph(cliqueSubgraph_2, stylesheet = style_1, size =
  [500, 500], style = circle, showweights = false)
```



Who Are The Most Influential Characters?

Weighted Betweenness Centrality

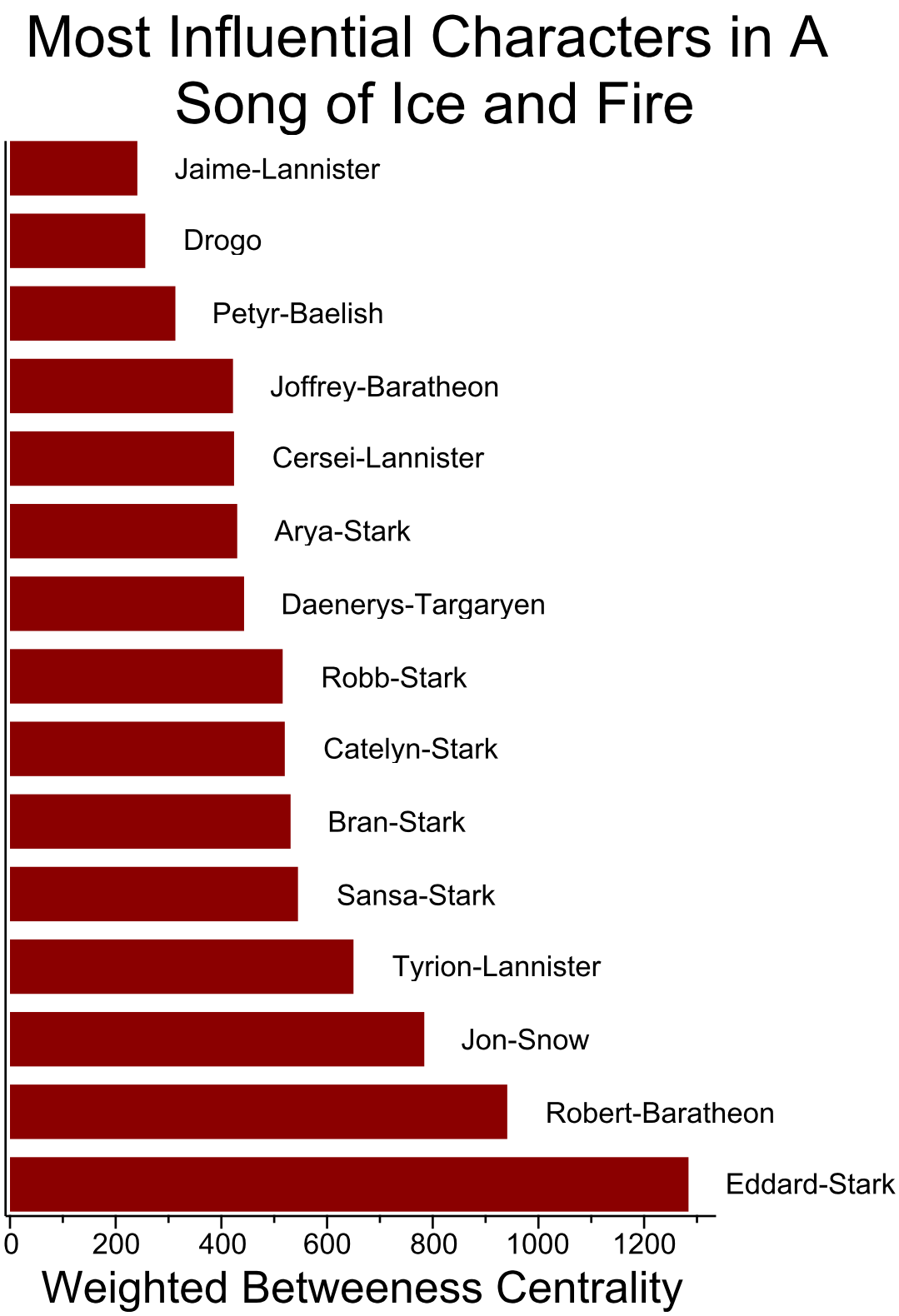
```
> ad_mat := AdjacencyMatrix(G):
  wt_mat := WeightMatrix(G):

> BetweenessCentrality_data := < node_data[1.., 1] | < seq(add
  (ad_mat[i, j] * wt_mat[i, j], j = 1.. num_characters), i = 1.
  . num_characters)> >:
  BetweenessCentrality_sorted := FlipDimension(<seq(sort([Row
  (BetweenessCentrality_data, 1..-1)], key = (x -> x[2])))>, 1)
```

;

$$BetweenessCentrality\_sorted := \begin{bmatrix} \text{"Eddard-Stark"} & 1284 \\ \text{"Robert-Baratheon"} & 941 \\ \text{"Jon-Snow"} & 784 \\ \text{"Tyrion-Lannister"} & 650 \\ \text{"Sansa-Stark"} & 545 \\ \text{"Bran-Stark"} & 531 \\ \text{"Catelyn-Stark"} & 520 \\ \text{"Robb-Stark"} & 516 \\ \text{"Daenerys-Targaryen"} & 443 \\ \text{"Arya-Stark"} & 430 \\ \vdots & \vdots \end{bmatrix} \quad \text{(7.1.1)}$$

187 × 2 Matrix

```
> BarChart([seq(BetweenessCentrality_sorted[i, 1] =
  BetweenessCentrality_sorted[i, 2], i = 1..15)], size = [800,
  800], style = surface, title = "Most Influential Characters
  in A Song of Ice and Fire", titlefont =  [Arial, 20], font =
  [Arial, 10], axesfont = [Arial, 10], labels = ["Weighted
  Betweeness Centrality",""], labelfont = [Arial, 16], axes =
  frame, size = [800,600], color = "DarkRed")
```



Normalized Degree Centrality

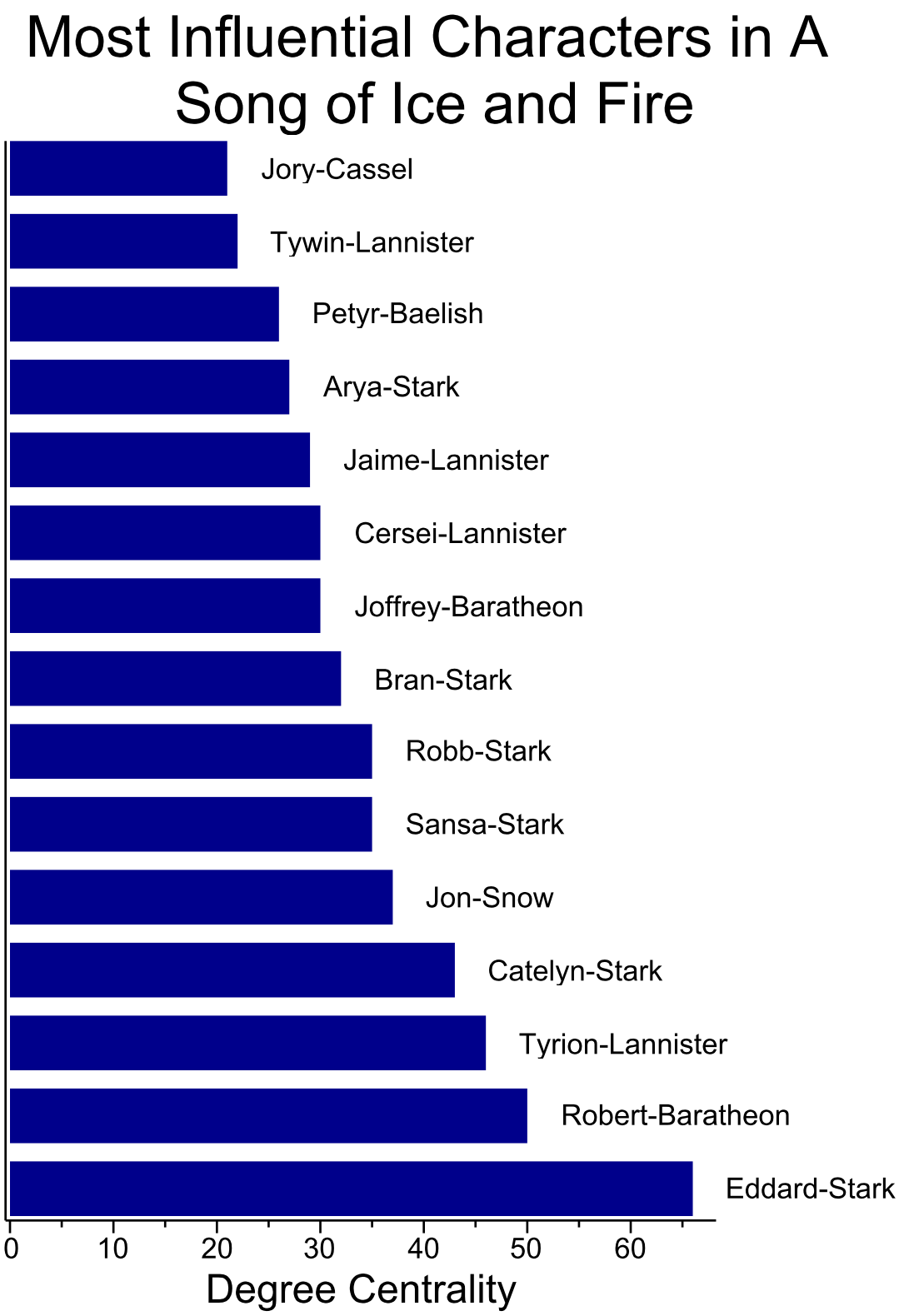
This measures the number of connections a vertex has

```
> DegreeCentrality_data   := << node_data[.., 1] | 1.0 *
  Degree~(G, node_data[.., 1]) >>:
```

```
DegreeCentrality_sorted := FlipDimension(< seq(sort([Row
(DegreeCentrality_data, 1..-1)], key = (x -> x[2]))) >, 1);
```

$$DegreeCentrality\_sorted := \begin{bmatrix} \text{"Eddard-Stark"} & 66.0 \\ \text{"Robert-Baratheon"} & 50.0 \\ \text{"Tyrion-Lannister"} & 46.0 \\ \text{"Catelyn-Stark"} & 43.0 \\ \text{"Jon-Snow"} & 37.0 \\ \text{"Sansa-Stark"} & 35.0 \\ \text{"Robb-Stark"} & 35.0 \\ \text{"Bran-Stark"} & 32.0 \\ \text{"Joffrey-Baratheon"} & 30.0 \\ \text{"Cersei-Lannister"} & 30.0 \\ \vdots & \vdots \end{bmatrix} \quad \text{187 × 2 Matrix} \qquad \textbf{(7.2.1)}$$

```
> BarChart([seq(DegreeCentrality_sorted[i, 1] =
DegreeCentrality_sorted[i, 2], i = 1..15)], size = [800,
800], style = surface, title = "Most Influential Characters
in A Song of Ice and Fire", titlefont =  [Arial, 20], font =
[Arial, 10], axesfont = [Arial, 10], labels = ["Degree
Centrality",""], labelfont = [Arial, 14],axes = frame, size =
[800, 600], color = "DarkBlue")
```



Eigenvector Centrality

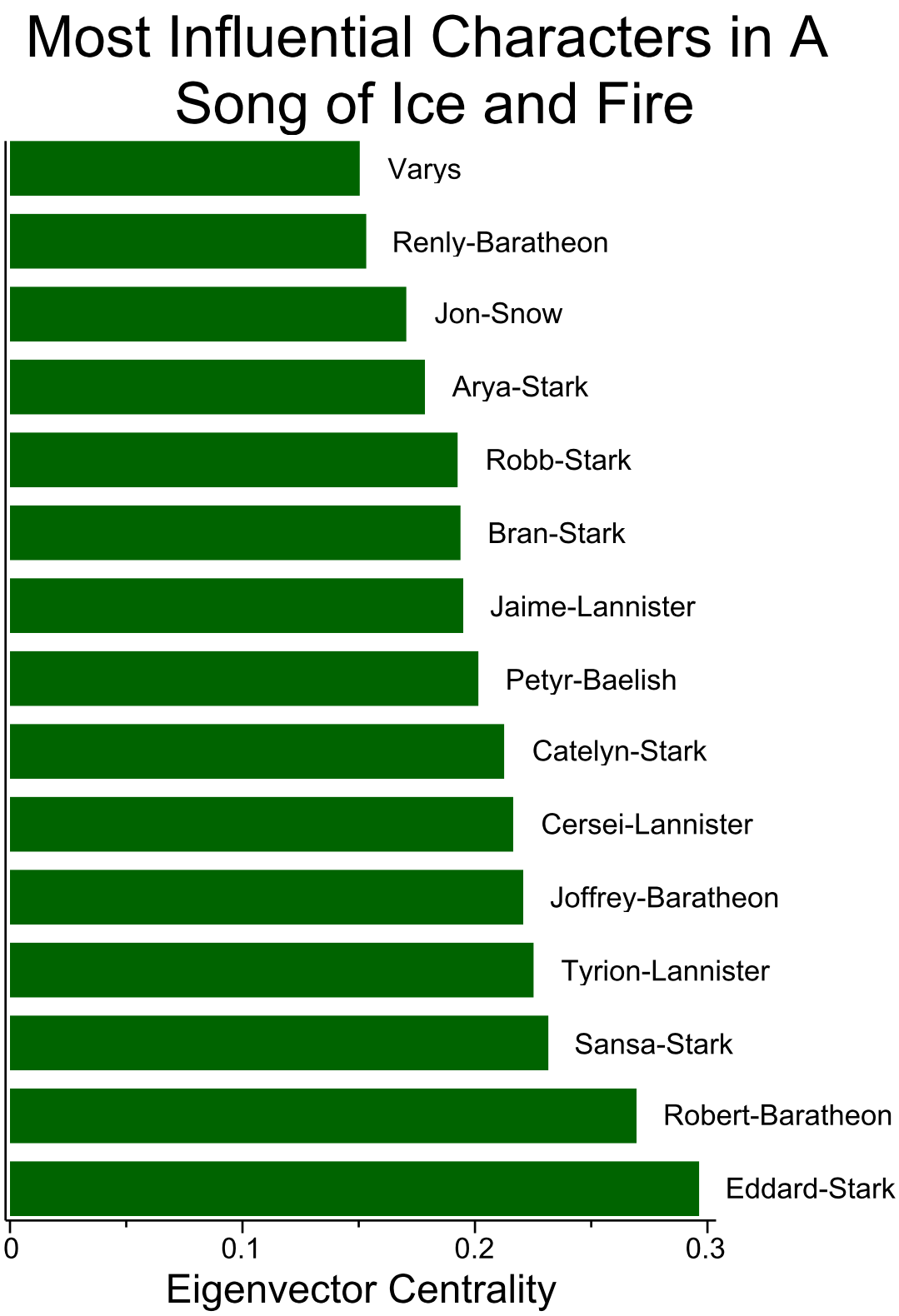
```
> DominantEigenvector := Eigenvectors(evalf(AdjacencyMatrix(G))
  )[2][..,num_characters]:
```

```
EigenvectorCentrality_data   := < node_data[1.., 1] |
DominantEigenvector * signum(DominantEigenvector[1])>:
EigenvectorCentrality_sorted := FlipDimension(<seq(sort([Row
(EigenvectorCentrality_data, 1..-1)], key = (x -> x[2])))>,
1);
```

$$EigenvectorCentrality\_sorted := \begin{bmatrix} \text{"Eddard-Stark"} & 0.296409699136637 \\ \text{"Robert-Baratheon"} & 0.269488050150604 \\ \text{"Sansa-Stark"} & 0.231551733564474 \\ \text{"Tyrion-Lannister"} & 0.225202726657263 \\ \text{"Joffrey-Baratheon"} & 0.220755262889052 \\ \text{"Cersei-Lannister"} & 0.216434674917249 \\ \text{"Catelyn-Stark"} & 0.212548763262289 \\ \text{"Petyr-Baelish"} & 0.201465596293454 \\ \text{"Jaime-Lannister"} & 0.194958997006186 \\ \text{"Bran-Stark"} & 0.193770549870058 \\ \vdots & \vdots \end{bmatrix} \quad \textbf{(7.3.1)}$$

187 × 2 Matrix

```
> BarChart([seq(EigenvectorCentrality_sorted[i, 1] =
  EigenvectorCentrality_sorted[i, 2], i = 1..15)], size = [800,
  800], style = surface, title = "Most Influential Characters
  in A Song of Ice and Fire", titlefont =  [Arial, 20], font =
  [Arial, 10], axesfont = [Arial, 10], labels = ["Eigenvector
  Centrality",""], labelfont = [Arial, 14], axes = frame, size
  = [800, 600], color = "DarkGreen")
```



Highlight the Most Influential Characters on the Overall Graph

Highlight the ten most important characters (as ranked by the Betweenness Centrality) on the graph

```
> HighlightVertex(G, convert(BetweenessCentrality_sorted[..10,
```

```
1], list), stylesheet = style_3):
DrawGraph(G, size = [2000, 2000], style = spring, showweights
= true, showlabels = true, redraw = false, stylesheet =
style_2)
```