CS5346 Project Report

Team 01

Wu Fan (A0196314A), Xu Cong (A0091599L),

Zhang Zhaoqi (A0178320J), Zhao Mengdan (A0105695Y)

1 Introduction

Winter is coming very soon – fantasy drama television series Game of Thrones (GoT) is scheduled to premiere its eighth and final season at April 14, 2019. Despite it feeling like almost a lifetime since the screening of the last season, the thrilling story, which is replete with honor and blood, is never to be forgotten.

As the most award-winning television series in history, Game of Thrones consists of intertwined character relationships, meticulously choreographed screenplays, hundreds of locations and complex stories that are full of unexpected twists and turns. To help viewers recall the previous seasons, sort out the complex relationships, facilitate the understanding and possibly offer an insight into the future plots, we resort to visualization tool D3.js to reveal some interesting discoveries that are encoded in various datasets on GoT.

In total, we have prepared 6 visualizations to answer 5 analytical questions.

1. **Where do the scenes in GoT take place?** GoT Scene Map – Xu Cong
2. **Who has been killed and by whom?** Character Kill Tree – Wu Fan
3. **Who appeared together?**
   a) Co-occurrence Chord Matrix – Zhao Mengdan
   b) Co-occurrence Heat Map – Zhao Mengdan
4. **Where have the characters been to?** Character Travel Map – Zhang Zhaoqi
5. **Who will be on the throne?** Throne Prediction Chart – Zhao Mengdan

The rest of this report is organized as follows. Section 2 gives a brief introduction to the data sources which our visualizations are built upon. Section 3 offers a comprehensive explanation to our visualizations each from several perspectives, mainly including 1) background and motivation for creating this visualization; 2) visual encoding that explains the components within; 3) insights that we derive from it. For visualization #1 and #2, charting logic is provided that goes into detail in elaborating each step in creating the visualization. Throughout the report, we strived to demonstrate the considerations and deliver the achievements in a methodical manner.

Four team members have made an equal amount of effort in preparing the visualizations, report, video demo and poster.

The video demo of all visualizations can be found at https://youtu.be/9rbR0BhkV2c.

2 Data Sources

We leveraged multiple data sources, of which most are JSON format as follows:

episodes.json This dataset contains detailed episode information including season, episode number, title, list of scenes, description, and so one. It is used in visualization: 1, 3a, 3b, 4, 5.

characters.json This file contains detailed information for each character, such as characterName, houseName, Image, family member relationship, and killed-by relationship. It is used in visualization: 2, 3a and 3b.

Characters-group.json This file contains information about which house(group) a character belongs to. It is used in visualization: 2, 3a, 3b.

lands-of-ice-and-fire.json This dataset contains the geographic information of the land, countries, places and cities in the GoT world. It is used in visualization: 1 and 5.

characters-include.json This dataset is used to filter characters to be included in the visualization as there are several hundreds of characters appeared in the TV series. It is used in visualization: 3a and 4.

3 Visualizations

#1 – Where do the scenes in GoT take place?

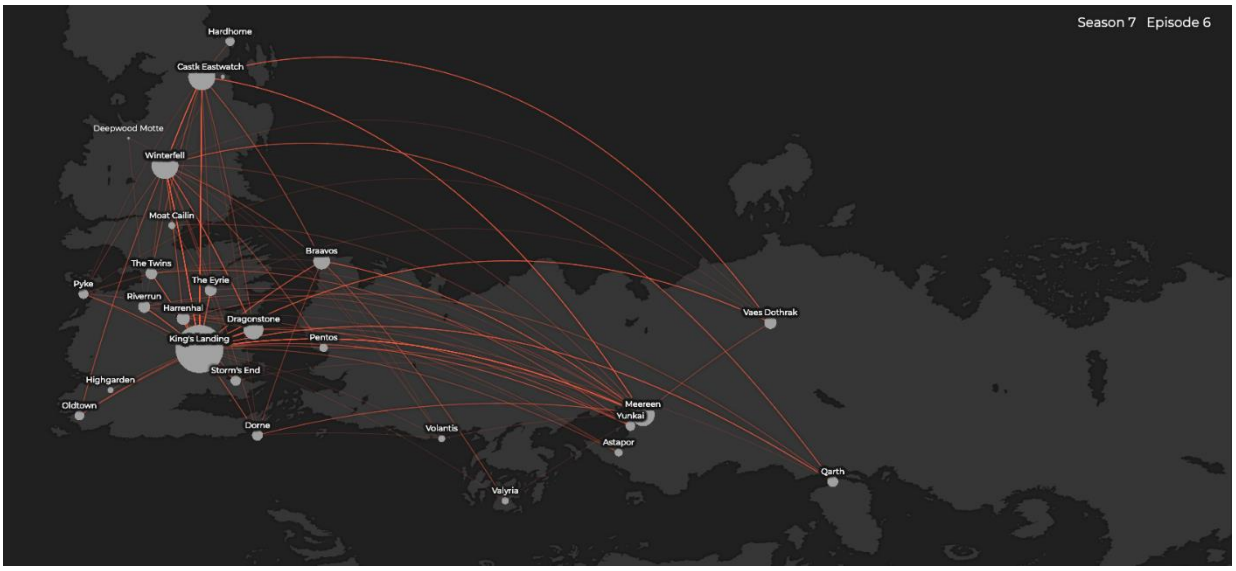


Figure 1: GoT Scene Map

GoT consists of complex stories and carefully choreographed screenplays that take place in hundreds of locations. This visualization illustrates where and when scenes in GoT take place and the connections between different scenes that help us to discover potential patterns and trends.

a. Data Processing

In this part, we used two data sources, lands-of-ice-and-fire.json, and episodes.json.

The first data source, lands-of-ice-and-fire.json, contains geometric data of points/arcs and coordinates of locations, which we used in D3 to render the map and calculate the distance between two given locations.

The second one, episodes.json, also contains information of scenes that happen in this episode, such as start and end time, location and sublocation and characters involved in this scene. Using the scene data provided in this dataset we can calculate the number of scenes for each location and the accumulated duration of scenes at a given season/episode. Details on how we process this data will be covered in the following section.

b. Visual Encoding

The map in the background is the fictional continents of Westeros and Essos. Each grey bubble on the map represents a location where the scene takes place. The orange link between bubbles represents the connection between scenes – if one scene takes place after another, there will be a link between the locations of these two scenes. The size of the bubble corresponds to the accumulated number of scenes happen there. The higher the number, the larger the bubble. The size of the bubbles, as well as the change of links between bubbles are animated over time from season 1 to season 7. The caption on the top right corner indicates the current season and episode number.

c. Charting Logic

To build this visualization, we first draw the map. The data in lands-of-ice-and-fire.json is already in the format that can be used directly by D3.js to render the map. The paths that is used to draw the map are extracted from *objects.land* field of the json file which contains geometric data of arcs on the map.

There are several types of map projections in D3. When drawing a plane world map, the *equirectangular projection* and the *spherical Mercator projection* are most commonly used. Despite that equirectangular can concisely portray relationship between the locations on the map projection, the distortions it introduces in render it useless in navigation or cadastral mapping. Considering that the locations are provided to fit in a planar graph, the idea of taking the trouble of repositioning it to fit in a 3D setting is not a favorable choice. Therefore, we take up the spherical Mercator projection method.

The second step is to prepare a list of scene-object from all episodes. We iterated through the json objects in episodes.json file to pick up all scenes in the sequence of the episode. Then, we used the location information in each scene to look up its corresponding coordinates in the other json file. By

using the scene start and end timestamps we calculated the duration for this scene. For each scene-object, we constructed a snapshot of accumulated scene time for all locations on the map at the time when this scene happened. In total the list contains some 2000 scene-objects from all episodes. By iterating through this list in sequence, we are able to illustrate the change in accumulated scene time for all locations on the map.

With the help of this scene-object list, we are able to draw the flying arcs between the locations on the map. Since each scene-object contains the coordinates of the location where the scene happens, we can calculate a trajectory that goes through all 2000+ locations in the list. To animate this trajectory, we first calculate the total length of the flying path. Then by utilizing the built-in *timer ()* function in D3, we are able to draw the full path with small increments at a given frequency, hence animating the flying path.

As the trajectory goes through each location, we also need to draw the bubble that represents the accumulated scene time for this location. To achieve this, we first calculate the ratio of which the trajectory has been drawn based on the total length of the path, the drawing speed and the time that has passed. For example, if the ratio is calculated to be 0.54, that means roughly 0.54 * 2000 scene-objects have been covered. To get the accumulated scene time of all location at this point of time, we just need to look at the snapshot that is stored in 0.54 * 2000 = 1080^{th} scene-object. We repeat this calculation at the same frequency of which the trajectory is being drawn so that the animation of the flying path and the bubble size change are roughly in sync.

The scene-object also contains the season and episode in formation. By using the same technique, we are able to display the current season/episode number and title in the visualization as well.

d. Insights

From the visualization we can clearly see that most of the scenes in GoT take place in King's Landing, Castle Eastwatch, Winterfell and Meereen. From season 4 onwards there is an obvious shift in scene locations towards the east, particularly towards the city of Meereen, while the previous three seasons are mostly telling stories that happen in the west of the Narrow Sea. From the density of the orange links we can tell that there are a lot of scene transitions between King's Landing and Meereen/Yunkai, which suggests that a lot of stories happen simultaneously or in close sequence in these two places.

#2 – Who has been killed and by whom?

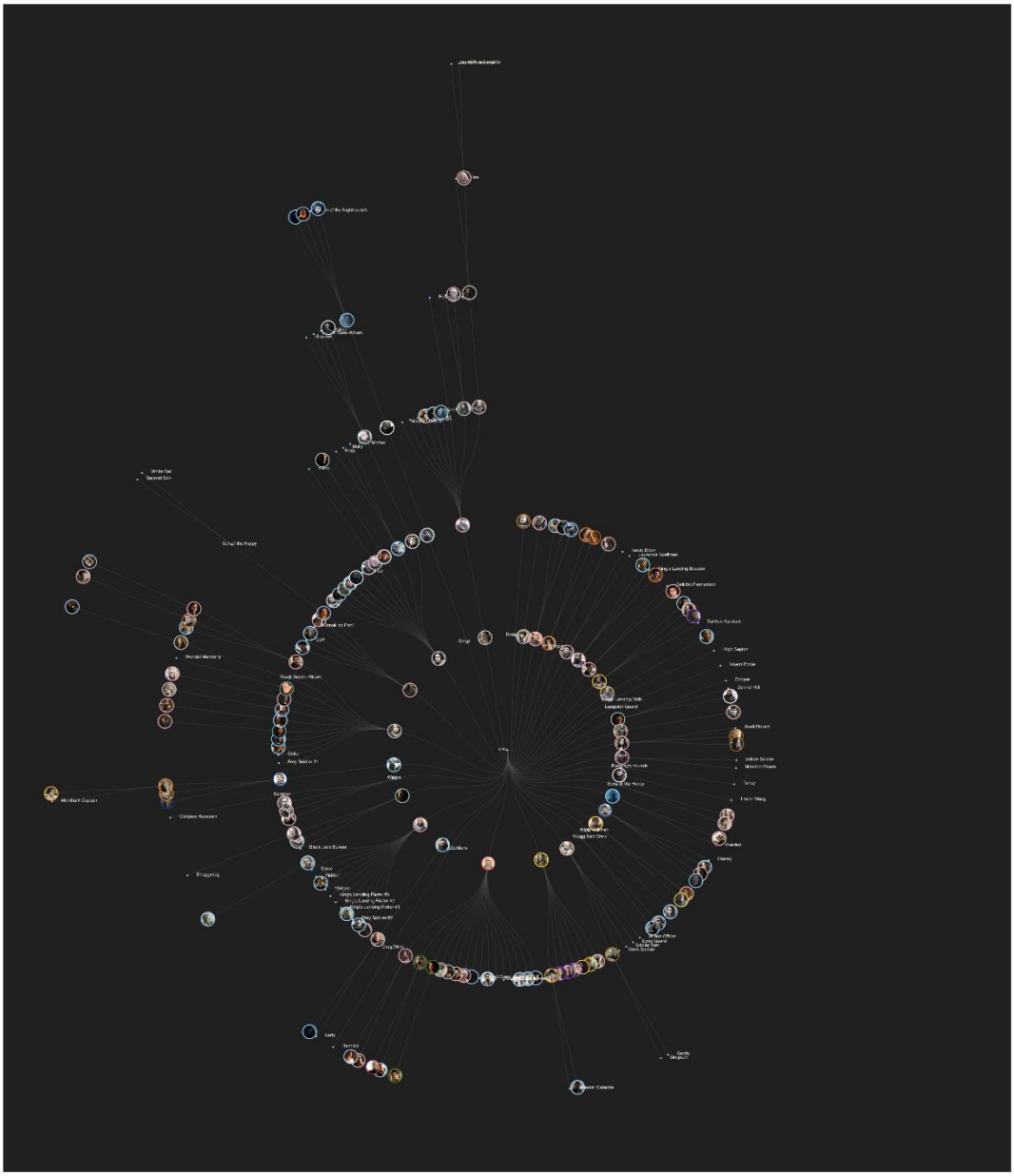


Figure 2: Character Kill Tree

It is known that death is an important component of Game of Thrones, and death scenes take up a considerable proportion of screen time. Characters died due to thousands of distinct reasons, strangled to death, shot in the head, slashed by a sword, starving to death, etc. Across the years, fanatic fans have collected a lot of statistics with regard to the deaths in TV series and provided some interesting analytic results, most of them centering around the death number in each episode, how and where the death

takes place, and how long the character lived before he died. In this big picture, we come up with a more insightful question – how is the "food chain" like in this game?

a. Data Processing

1. Problem Analysis

As we want to model the relationship of killing between all characters, the most efficient way is to represent it as a directed graph, where each vertex in the graph is a character, and each edge connects two vertex, denoting that one person killed another. We further noticed that, this graph contains no circles out of the natural property of this kind of relationship. Therefore, it is a tree. More precisely, it is a forest with several trees, and the root of each tree is the one that remains alive. Another possible situation is that this person commits suicide. In our counting, cases like this are in the minority.

In pursuit of consistency, we decided to overlook the cases of suicide; instead, we will introduce in another vertex as the new root of the tree and connects all the roots in the original forest to this new vertex. In this way, we eventually come up with a bigger tree which represents the killing relationship in Game of Thrones. More formally, it is a *Hierarchical Node-Link Diagram* with following components:

2. Data Preparation

We can directly retrieve the "killed-by" information from our data source, but that is not enough. To be able to generate the *Hierarchical Node-Link Diagram* using d3, we resort to the existing implementation – *Radial Tidy Tree*, which requires that the data be prepared in a certain format.

For example, if we have information such as

```
(Tom, Jason), (Tom, Jimmy), (Jason, Mary), (Jason, Sandy), (Sandy, Jim)
```

Then the data shall look like:

```
Tom
Tom.Jason
Tom.Jimmy
Tom.Jason.Mary
Tom.Jason.Sandy
Tom.Jason.Sandy.Jim
```

In order to get data of this format, we performed a *Depth First Search* with *Memory Mechanism*. Starting from each vertex, we explore up to its parent vertex (the one who killed him), until when we reach a vertex who either has no parent (the one that is alive) or has already been explored before (the one with known information). In the *backtracking* step, we then complete the information for all the vertex that has been traversed on the path in a *top-down approach.*

b. Visual Encoding

We modeled it as a *Tree Graph* in the family of *Hierarchical Node-Link Diagram.* There are several major components in the visualization:

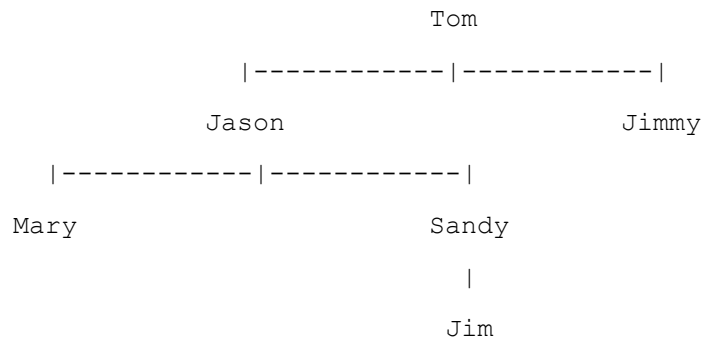
- Vertex
  - Root (at depth 0): serves the function of connection all parts together (no symbolic meaning);
  - Children of the root: characters that are still alive in the TV series;
  - Leaves: characters that have not killed anyone;
  - Other internal nodes: characters that have previously killed someone, and have already been killed by other people;
- Edge: Each edge connects two nodes, one in depth d, and the other in depth d+1, denoting that the character at depth d killed the character at depth d+1;
- Color: Each vertex has a color, denoting the group of the character.

c. Charting Logic

With the data generated above, we already have a well-defined relationship between all vertex. Using d3.tree, the data is again transformed into the following format:

```
Root: Tom
      |-Child_1: Jason
            |-Child_1: Mary
            |-Child_2: Sandy
                        |-Child_1: Jim
      |-Child_2: Jimmy
```

As we can see, each node linked in this hierarchical relationship corresponds to a vertex in the graph. This is mainly achieved by using the function *stratify()* and self-defined comparing rule, through which we derive a set of relationships represented like above. With this structure, calling *.descendants()* gives us all the subtrees of the current node. Viewing it layer by layer, we already have a hierarchical structure on hand.



But there is some problem with this setting. When suiting relationship into this architecture, we found that there are too many vertices in the first layer (depth 1, assuming that root is at depth 0), because living people are in the majority after all. Therefore, we have a tree of which the width is much larger than its height, which hurts the visual effect to a large extent. Thus, we considered switching to another setting which twist the tree into concentric rings, vertex at the same depth in the tree lying in the same ring.

We use a different example here because the original one is far too easy to display certain properties. In this example. Jason is at depth 0, Mary, Sandy, Tommy, Cindy, Bob, Tom at depth 1, Jim, Lucy, Jimmy, John and Nancy at depth 2.

Note that, the nature of this tree – the hierarchical relationship – remains the same across different representations. The only difference is that the location of nodes are computed in a different way. More specifically, when we want a layout of *concentric rings*, we calculated the coordinates using the angle and radius, compared with calculating the x and y coordinate in the case of *vertical* or *horizontal layout*.

With this down, we were only left with some minor work, such as adding the images for each characters to make the visualization more informative to the audience, and marking the characters as belonging to different groups to discover certain patterns such as which groups hold grudges against which group.

d. Insights

From the visualization we can spot an interesting phenomenon: majority of the people who killed the most are still alive. This can be easily perceived once we notice that vertex at outer rings mostly have no more than 3 descendants, while several of the vertex in the inner ring has a large number of descendants.

Among those who killed the most, Jon Snow and Daenerys Targaryen are well worth our attention. Titled as “King of the North” and the “Night’s Watch” at the same time, there is no wonder that Jon Snow killed many. Daenerys Targaryen, in pursuit of the notable career of liberating the slaves, killed a lot of slave owners, and is therefore distinguished as “Breaker of Chains”. As to the group of the living people, of them, do not have specific affiliation.

#3 – Who appeared together?

Other than the killed-by relationship, character co-occurrence could also be an interesting area to look at. For most of GoT fans, understanding which character appears most frequently with their favorite character is always exciting. We have chosen two types of visualizations to represent the relationship: Chord graph and Heatmap. Chord graph is usually more effective through interactive channels, such as web pages or mobile platforms, while heatmap is very commonly used in static means such as papers or images.

a. Data Processing

A matrix has been calculated to represent the co-occurrence relationship. As shown in the sample matrix below, an hh:mm:ss format data represents the length of total screen time for two characters has appeared in the same scene before.

	Character 1	Character 2	Character 3	...
Character 1	01:17:22	01:03:41	0	
Character 2	0	02:00:00	0	
Character 3	00:13:05	0	00:55:20	
...				…

Figure 3, Co-occurrence Relationship Matrix

b. Visualization 3a - Chord Graph

In this graph, we only selected the top 45 popular characters as more characters will contribute to the complexity of the graph and may even cause difficulties for users to extract information.

I. Visual Encoding

Character names are arranged radially around a circle with the co-occurrence relationships between each character drawn as arcs connecting them. Color is used to differentiate between characters. Moreover, if users hover their mouse on the ring for a certain character, his/her co-occurrence arcs will be highlighted.

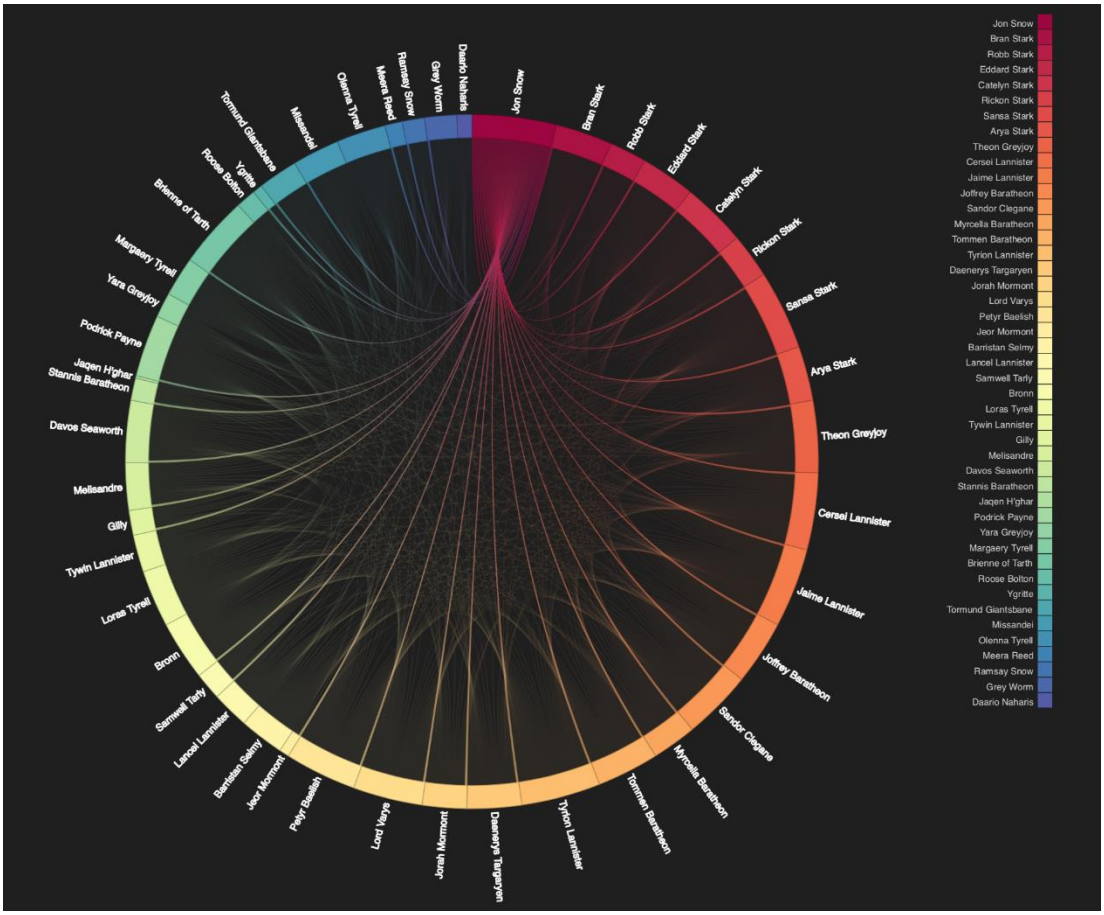


Figure 4: Chord Graph

II. Insights

As one of the main characters, Jon Snow possesses the most connection arcs with other characters. However, the other main character Daenerys Targaryen doesn't possess as many connections as Jon Snow. This could because Daenerys' main storyline is mostly in Essos instead of the mainland Westeros. Therefore, she has a lesser chance to interact with the other main characters. Another character that's worth look at is Brienne of Tarth. Even though she is not the most important character, she has almost the same number of arcs as Jon Snow. The reason behind could be that as a trustworthy knight, she has been appointed to protect many important characters.

c. Visualization 3b - Heatmap

Heatmap uses a similar dataset but can visualize more information at once compared to chord graph. Heatmap is also more flexible in grouping data. This visualization supports 3 types of sorting mechanism: order by name, frequency and cluster. While sorted by frequency, the co-occurrences with a longer duration will appear on the top left corner. While sorted by cluster, the co-occurrences will be grouped by houses, as shown in Figure 5.

I. Visual Encoding

In Figure 5, x and y axis contain GoT characters from different houses(groups). If two characters have appeared in the same scene, the corresponding square on the matrix will be color-coded. If both characters come from the same house, a special color that denotes that house will be used. While the user hovering the cursor on a particular square in the matrix, a tooltip with details about the co-occurrence will be displayed. For example, “Tyrion Lannister and Jon Snow are onscreen together for 01:04:13”.

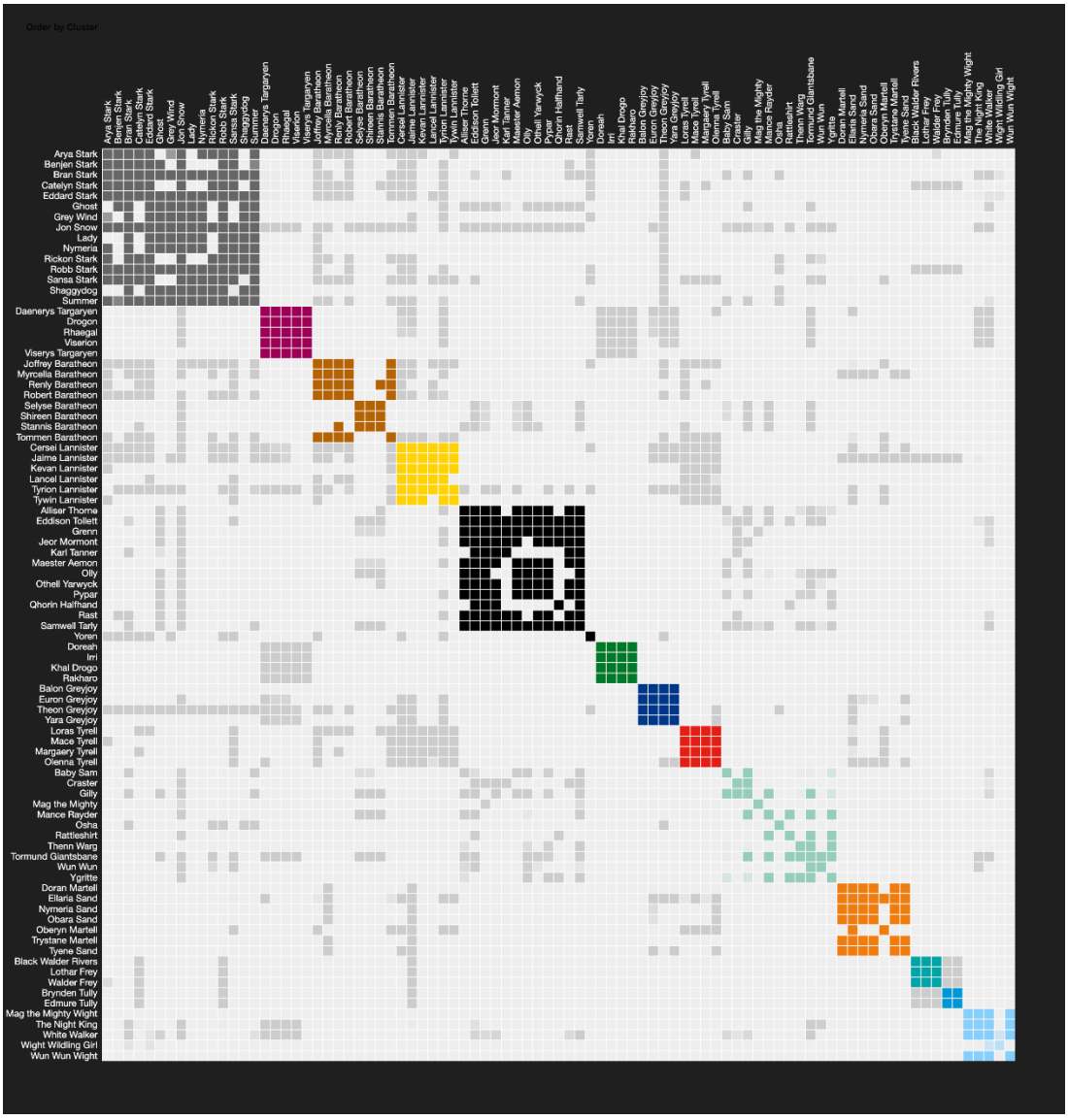


Figure 5: Heatmap

II. Insights

From the Figure 5, we know that House Stark and Night’s Watch are the two groups with most co-occurrence times, followed by House Baratheon, House Lannister and House Targaryen. This is no surprise as most of the main characters are from these families. However, it is interesting to find that

group Wildling also has a sparse but big correlation matrix (light green colored). This means there is no high co-occurrence within the Wildling group members, but they are frequently appearing with other characters in different scenes.

#4 – Where have the characters been to?

Figure 6: Character Travel Map

a. Data Processing

As mentioned at visualization 1, we used two data sources, one is Geometries data of points/arcs and coordinates of locations coming from lands-of-ice-and-fire.json and another is scenes information that happens in this episode from episodes.json.

The first step that we did is, we drew the GoT map background as the Figure 6. Then, since we want to figure out where this character has traveled, we picked up each character's all scenes, and then extract location information from them. Because one character might appear in multiple continuous scenes, so we filtered them out and only kept exact one location for this situation. Besides the location, we also want to show which season this travel happens, so we recorded its season information along with location as a pair. After getting these location-season pairs, we can draw each character's travel trace point by point in the map. Moreover, because We only selected the top 45 popular characters as more characters will contribute to the complexity of the graph and may even cause difficulties for users to extract information.

b. Visual Encoding

Map and the arc lines are used to represent the travel trace of each character. First, users can select the character they are interest in, then, the arc lines will link the locations this character has traveled to point by point. At the same time, the text beyond the map will tell uses, which season this travel line happens. Also, at this map visualization, four shapes are using as legend, the square represents city, triangle for castle, large dot for town, small dot for ruin.

c. Insights

It's typically hard for a reader to track a specific character along with the seven seasons, especially some of the seasons have happened several years ago. However, the trace the character has traveled is important to understand this character's experiences as well as his characteristics. For example, as the Figure 6 shows, Jorah Mormont, a Northern lord exiled from Westeros previously living in Essos. At the first season, his first show was at the wedding between Daenerys Targaryen and Khal Drogo where he fell in love with this bravery girl, then he followed Daenerys to Vaes Dothrak, and he saved her from being poisoned there. At the end of the first season, Jorah pledged fealty to Daenerys and witness the hatchlings of her three dragons. After that, Jorah traveled from Essos to Westeros during the following five seasons. Finally, he arrived at The Wall last season and must will play an important role at the coming winter.

#5 – Who will be on the throne?

As the storyline of Game of Throne is always unpredictable, guessing who will eventually win the Iron Throne is one of the hottest topics for GoT fans. Therefore, this visualization is aiming to provide GoT fans with a Quantitative view for the estimation.

a. Data Processing

In our prediction, we believe there should exist a positive correlation between characters' screen time and the possibility of wining the Iron Throne. Therefore, we computed the linear regression of screen time per episode for each character using below formula. As shown below, a is the intercept and b is the slop, while x and y represents episode index and screen time in the episode.

$$a = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}$$

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

Figure 7: Formulas for Intercept and Slop

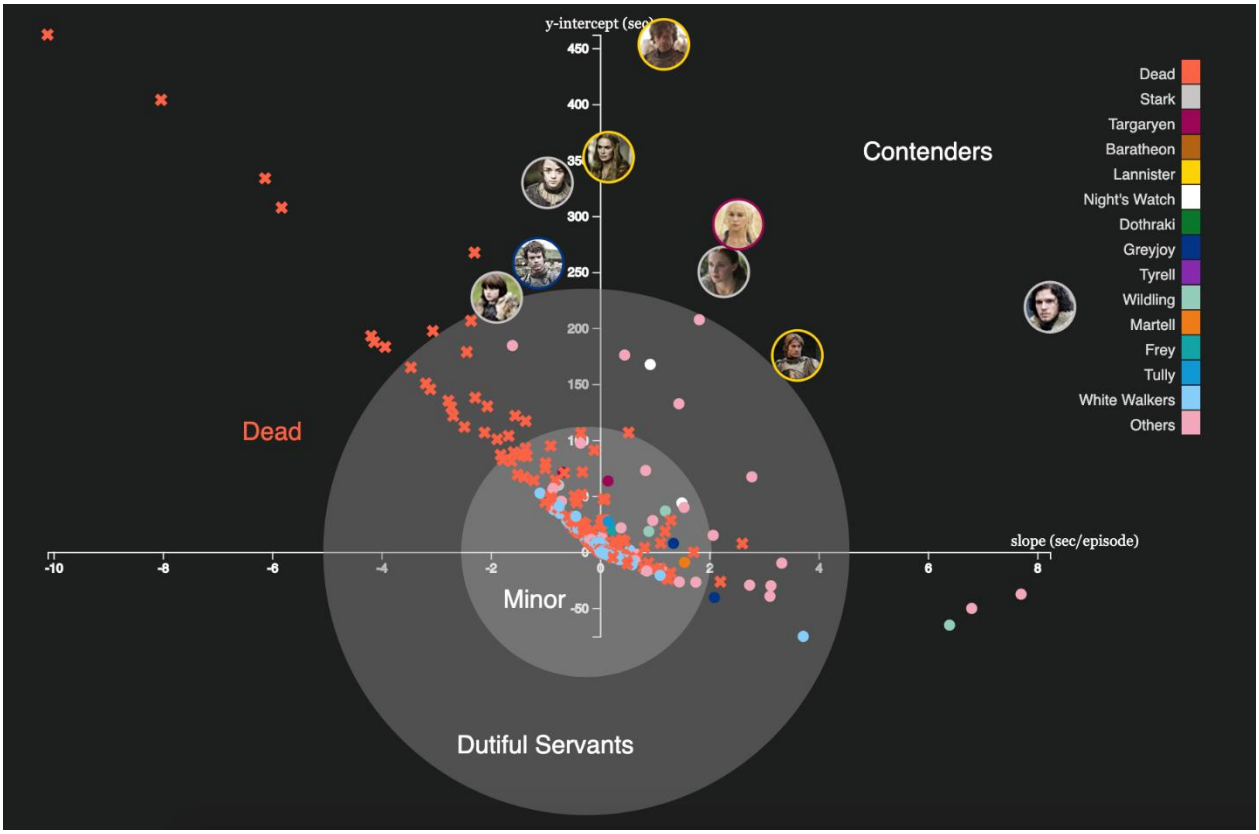


Figure 8: Throne Prediction Chart

b. Visual Encoding

Since we need to plot two quantitative data: y-intercept and slope, scatter plot is the best fit. In the graph below, color encoding is used to differentiate force groups (Houses) for each character. On the background, two concentric circles are placed to indicate the character's state: Dead, Minor, Duitiful Servants or Contenders. The closer a point to the center indicates the lower chance he could win. To emphasize on characters who falls under Contenders, their corresponding dots are enlarged to show their thumbnail images. And to differentiate characters who are already dead, a cross symbol is used instead of dots. Moreover, for each dot on the graph, there is a tooltip added to show its name.

c. Insights

As shown in the graph below, Jon Snow and Tyrion Lannister are the two strong competitors for the Iron Throne. Daenerys Targaryen, Cersei Lannister, Arya Stark and Sansa Stark have lower possibility than them.

4 Conclusions

In this project, we try to address several interesting questions related to the popular television series Game of Thrones. We are honored to base our work on an existing extensive corpus, and, through the observation and analyses of which, propose some new questions that we think might be the concerns of a large number of GoT fans. Using the visualization tool D3.js, we come up with 6 elaborately designed visualizations to address 5 questions, and, offer additional insights and new perspectives apart from answering the questions. We sincerely hope this work can strike a chord with GoT fans and bring the glamour of GoT to a larger scope of audiences.