Final Project - Game Of Thrones

Shai Ben Ami 203340153 & Avraham Raviv 204355390

Assigned: 10.07.2020

Contents

Introduction

Research question

Tv shows, especially viral ones, are this decade's biggest hype. What if we could predict a show's end? Is it possible at all?

Specifically, we will focus on Game Of Thrones. Game Of Thrones was the most talked series these days last year, and perhaps even the most popular series ever watched. The last season just aired, And hundreds of millions of people were watching with great suspense towards the end.

In our project we want to explore data and try predict the end of the eighth season (and the whole series) based on the first seven seasons. So our research question is: "Who will sit on the iron throne in the end of the series And which house will be most dominant at the end?".

It might be difficult to predict who will be the one (or ones) to sit on the iron throne, but perhaps some other questions that can much more easily addressed will be helpful to dare answering such mysterious question. In addition, even though it might be difficult to predict who will be the winner, it'd be more simple to predict which house they are from, and we will also address this issue in the research.

We assume a model **based on PCA** that predicts the end according to the following parameters - **screen time, dominance in battles, the most dominant house and the audience's Opinion**. Therefore, we will focus on the following questions:

1. Which character has the most battlefield experience, and which House has won more fights?
2. Which character has the most screen time in the show?
3. Does the audience's opinion affect show's plot (in general)? if so, who is the most likeable character, and who do people around the globe - and especially around social media - think will win at the end?
4. Which House is the most dominant, in terms of land and territory, but also in terms of connections and influence (moeny, etc.)?

By analyzing data and hopefully answering these questions, we hope to create a statistic model by which we will be able to predict (or even say we couldn't predict) who will be the one to rule the seven kingdoms.

Cases

Our information is divided into two - analysis of data related to events during the series, such as battles, tactics, family ties and more, and analysis of external data, such as screen time, audience's opinions etc.

We have some resources that we already collected - screen time for each character, from IMDb.

In addition, we found database for most of the characters, including their family connections, status and more, and also battles that took place and which houses took part in them. We will add and edit data of "Defeating Rank" by the already-collected data of the battles.

Type of study

The study has two types:

a. The study was made on data from the first seven seasons; and also public opinion poll, so it is observational.

b. Also, for the part of screen time (and maybe for characters data as well) we use some other data for 'training'. We explore the training method in our proposal - Harry Potter screen time will be our training for question "Does characters with more screen time have correlation with last survivers and winners?"

Data Source

for the first method, we are collecting from:

https://github.com/mathbeveridge/asoiaf/tree/master/data

https://www.strawpoll.me/10765241/r

https://www.imdb.com/list/ls027460372/ (for harry potter time screen)

https://www.imdb.com/title/tt0455275/fullcredits?ref_=tt_cl_sm#cast (for Prison break time screen)

and more.

For the survey, we have found some polls published during the break between seasons 7 and 8.

EDA - Exploratory data analysis

PCA - Principal Components Analysis

The selection of parameters that we presented at the opening is not arbitrary, but is based on the PCA algorithm.

```
# drop columns with non-numeric data, take the relevant params and replace Na with 0
sub_pred<-pred_data[,c(8,17:21,27,29,30,32)]
sub_pred[is.na(sub_pred)]<-0
res.pca<-prcomp(sub_pred,  scale = TRUE)
fviz_pca_var(res.pca, col.var="contrib")
```

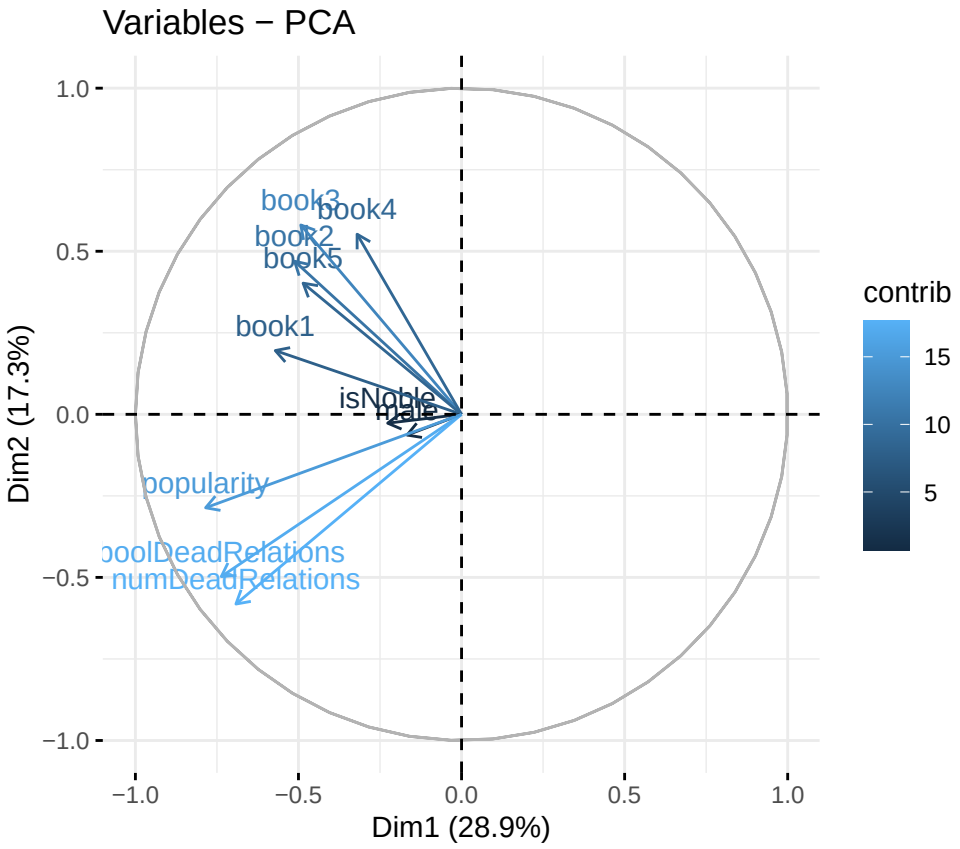


Figure 1: Fig 0.1: PCA analysis

PCA reduces the dimensionality of multivariate data, to two or three that can be visualized graphically with minimal loss of information. fviz_pca_var() helped us to plot the variables as a vectors, while the closer the vectors to each other, the higher the correlation between them. Also, the longer the vectors, the stronger the correlation.

We can see that from relevant columns, popularity, number of dead relations and number of blood dead relations explain the majority of the variable in our data.

So we chose to trace the four parameters we presented at the opening, which represent the factors presented by the PCA.

Screen time

first of all let us observe the screen time of the characters and houses in the show. As we will see, there is possibly a relation between the screen time a character has and its chances to survive and even "win". Thus, this is a very significant paramter when trying to foresee a show's end.

The data in this section is continuous and one dimensional, so there is no meaning to look at it in terms of clustering.

Analysis of the data for the characters:

```
#now lets merge all of them:

got.melt <- melt(character, id.vars='character')

par(mar=c(10,3,3,3))

allseasons <- ggplot(got.melt,
     aes(x = reorder(character, value, FUN = sum), y = value,
         fill=forcats::fct_rev(variable))) +
   geom_bar(stat = "identity",width = 0.7) + coord_flip() +
   theme_minimal(base_size=9) +
   labs(title="Screen time of characters in the Game of Thrones",
        subtitle="By seasons",
        x="characters",
        y="Screen time (minutes)", col="Season",
        fill=" ") + theme(axis.text=element_text(size=7)) + theme_grey(base_size = 6)+
  theme(axis.text.x = element_text(color = "grey20", size = 14, angle = 0, hjust = .5, vjust = .5, face
        axis.text.y = element_text(color = "grey20", size = 8, angle = 0, hjust = 1, vjust = 0, face =
        axis.title.x = element_text(color = "grey20", size = 12, angle = 0, hjust = .5, vjust = 0, face
        axis.title.y = element_text(color = "grey20", size = 12, angle = 90, hjust = .5, vjust = .5, fa
#top 10:
top10<-allseasons[[1]][1:10,1]
top10<-as.data.frame(top10)
top10
```

```
##                 top10
## 1            Jon Snow
## 2    Tyrion Lannister
## 3  Daenerys Targaryen
## 4         Sansa Stark
## 5    Cersei Lannister
## 6          Arya Stark
## 7     Jaime Lannister
## 8       Samwell Tarly
## 9       Jorah Mormont
## 10      Theon Greyjoy
```

We can also show the entire cast, in order to get a good view about the average screen time of characters.

```
allseasons
```

Presented above is the distribution of the screen time for characters and houses. It can be easily seen that Jon and Tyrion have the longest screen time, and so do the Starks and the Lannisters, who appear to be the most dominant Houses (by screen time).

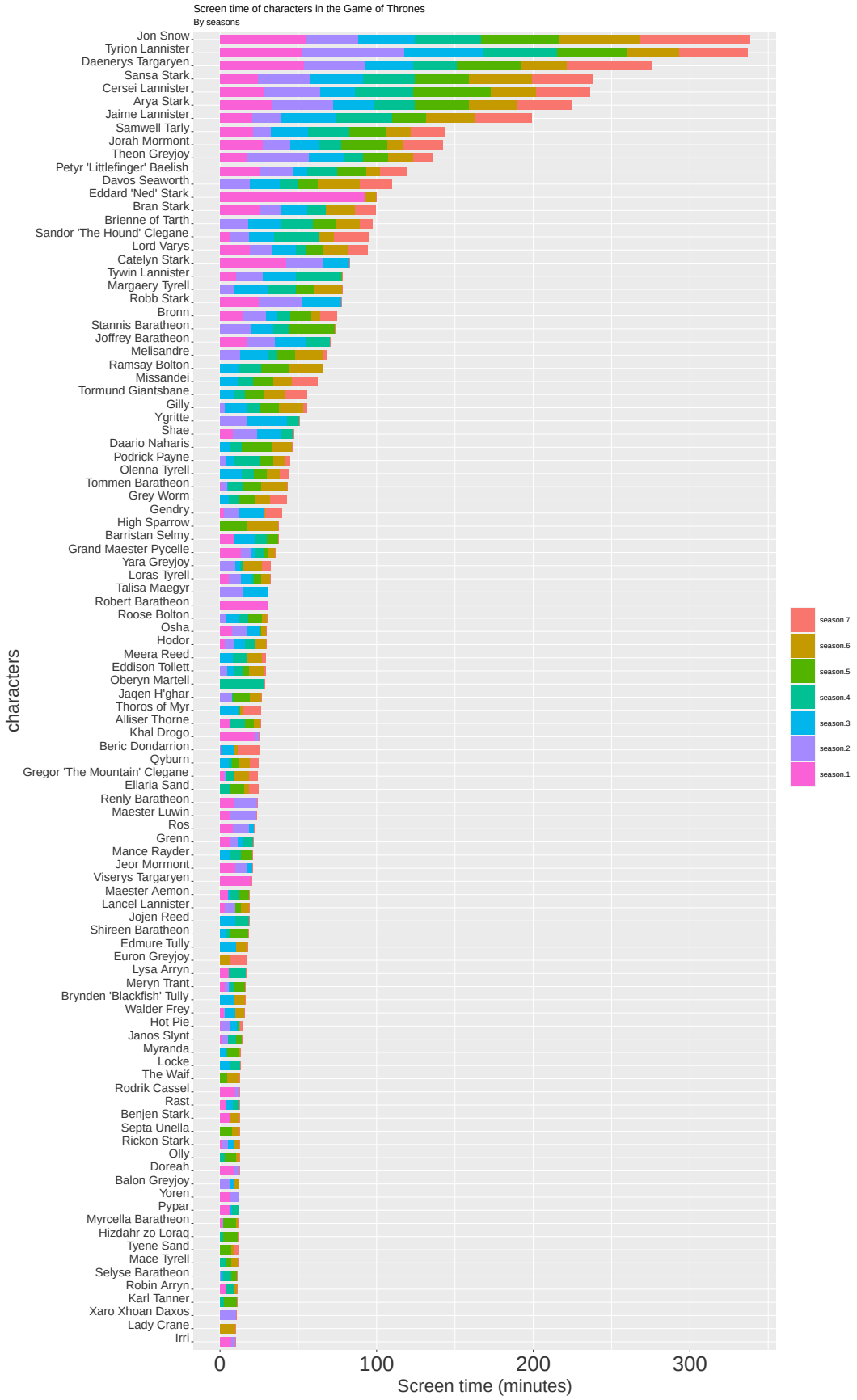


Figure 2: Fig 1.1: Time Screen per Character over the seasons - all characters

Let's try and quantify this information, and see more precisely the distribution of the time screen. But before we dig deep into the numbers, let's just bring some motivation to our analysis first - screen time in other shows. If we find this trend in other shows, we may be able to project this idea to our show as well.

We will use Harry Potter and Prison Break, as 'training'. We took data of Harry Potter (8 parts) and explored the time screen about top ten time screen's characters.

```
#Add BOOLEAN line that check if the character is dead
alive<-c(TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, FALSE, FALSE)
HarryTime<-cbind(HarryTime, alive)

#Add line of sum of screen time
sumoftime<-c(sum(as.numeric(HarryTime[1,])), sum(as.numeric(HarryTime[2,])), sum(as.numeric(HarryTime[3
HarryTime<-cbind(HarryTime, sumoftime)

#sort the data over the time screen and plot
HarryTime[order(HarryTime$sumoftime),c(1,10,11)]
```

```
##              Character alive sumoftime
## 10 Bellatrix Lestrange FALSE        12
## 7   Minerva Mcgonagall  TRUE        31
## 8         Draco Malfoy  TRUE        34
## 5       Lord Voldemort FALSE        42
## 9        Severus Snape FALSE        51
## 6        Rubeus Hagrid  TRUE        54
## 4     Albus Dubmledore FALSE        74
## 3     Hermione Granger  TRUE       194
## 2          Ron Weasley  TRUE       233
## 1         Harry Potter  TRUE       543
```

```
par(mfrow = c(1,1), mgp=c(0, 1, 0), las=2)
par(mar=c(5,4,3,4))
plot(HarryTime$sumoftime, HarryTime$alive, main="Screen Time for Harry Potter", yaxt = "n",
ylab="Is Alive?", xlab ="screen time [M] ",col = ifelse(HarryTime$alive > 0,'darkcyan','bisque4'), pch=9
legend(par('usr')[2], par('usr')[4], bty='n', xpd=NA,
       c("alive", "dead"), col = c("darkcyan", "bisque4"), pch = 9)
plot.axes={axis(2,at=seq(0,1,1), labels = c("FALSE", "TRUE"))}
text(HarryTime$sumoftime[which(HarryTime$alive == "TRUE")], HarryTime$alive[which(HarryTime$alive == "TR
text(HarryTime$sumoftime[which(HarryTime$alive == "FALSE")], HarryTime$alive[which(HarryTime$alive == "F
grid()
```

It's easy to see that high screen time can somehow predict survival and even "winning" the series - Harry, Ron and Hermione all have the highest time screen, and they "won" and defeated the Death Eaters and Lord Voldemort.

Let's explore the same point in another series - Prison Break (5 seasons):

```
#Add BOOLEAN line that check if the character is dead
alive<-c(TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, TRUE, TRUE, FALSE, FALSE)
PrisonBreakTime<-cbind(PrisonBreakTime, alive)

#sort the data over the time screen and plot
PrisonBreakTime[order(PrisonBreakTime$sumoftime),]
```

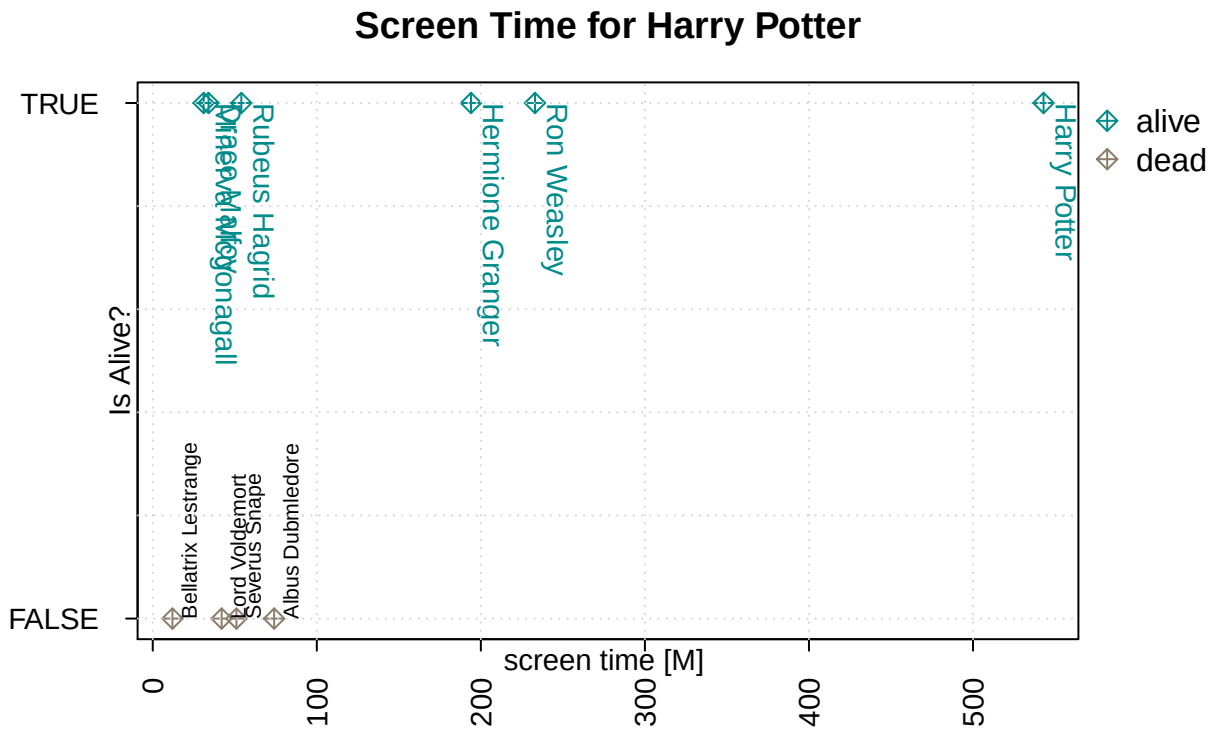


Figure 3: Fig 1.2: screen time and finish position in the series of Harry Potter. Top is alive, bottom is dead

```
##                          Character sumoftime alive
## 10               christina scofield        86 FALSE
## 9                   jonathan krantz        98 FALSE
## 8  Benjamin Miles 'C-Note' Franklin       121  TRUE
## 7                    Fernando Sucre       154  TRUE
## 6                      Brad Bellick       176 FALSE
## 5           Alexander 'Alex' Mahone       189 FALSE
## 4                     Sara Tancredi       210  TRUE
## 3          Theodore 'T-Bag' Bagwell       378  TRUE
## 2            Lincoln 'Linc' Burrows       438  TRUE
## 1                  Michael Scofield       561  TRUE
```

```
par(mfrow = c(1,1), mgp=c(0, 1, 0), las=2)
par(mar=c(5,4,3,4))
plot(PrisonBreakTime$sumoftime, PrisonBreakTime$alive, main="Screen Time for Prison Break", yaxt = "n",
ylab="Is Alive?", xlab ="screen time [M] ",col = ifelse(PrisonBreakTime$alive > 0,'darkcyan','bisque4')
legend(par('usr')[2], par('usr')[4], bty='n', xpd=NA,
       c("alive", "dead"), col = c("darkcyan", "bisque4"), pch = 9)
plot.axes={axis(2,at=seq(0,1,1), labels = c("FALSE", "TRUE"))}
text(PrisonBreakTime$sumoftime[which(PrisonBreakTime$alive == "TRUE")], PrisonBreakTime$alive[which(Pris
text(PrisonBreakTime$sumoftime[which(PrisonBreakTime$alive == "FALSE")], PrisonBreakTime$alive[which(Pr
grid()
```

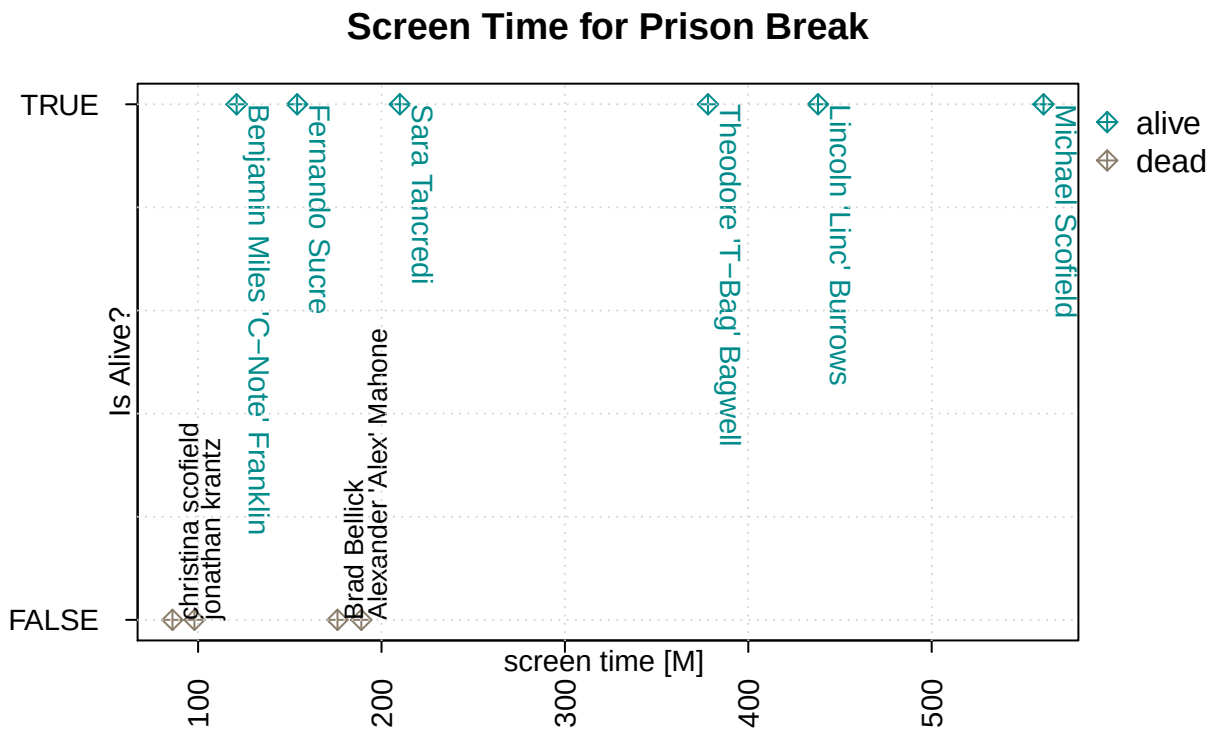


Figure 4: Fig 1.3: screen time and finish position in the series of Prison Break. Top is alive, bottom is dead

Here, too, we can see the same trend - those who received the most time screen, are those who eventually

stayed to the end and won the series (the brothers, Michael and Lincoln).

Now let's try to Quantify this correlation, and before using the tests, give some background and explain them:

A correlation coefficient gives us some knowledge about the extent to which two variables tend to change together. The change can be either linear, or sometimes simpler – just monotonic. The result can be ranged from -1 to 1, Where -1 means a perfect negative association and +1 indicates a perfect positive association. While *Pearson* evaluates the linear correlation between continuous variables, the *Spearman* coefficient evaluates the monotonic relationship between two ordinal variables as well.

Another correlation test is called *Point-Biseral*, which measures the strength of association between variables, where one of them is binary (dichotomous variables as well).

Below we use the Spearman coefficient (over Pearson) since we have ordinal set of variables. And we even have only two – alive or dead – so we checked Point-Biseral as well.

Null hypothesis: H_0 - there is NO correlation between screen time and chance to stay alive. Alternative hypothesis: H_a - there is correlation between screen time and chance to stay alive, and even to win.

```
HarryCor<-cor.test(HarryTime$sumoftime, as.numeric(HarryTime$alive), method = "spearman")
HarryCor
```

```
##
##  Spearman's rank correlation rho
##
## data:  HarryTime$sumoftime and as.numeric(HarryTime$alive)
## S = 106.37, p-value = 0.3136
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.3553345
```

```
PrisonBreakCor<-cor.test(PrisonBreakTime$sumoftime, as.numeric(PrisonBreakTime$alive), method = "spearma
PrisonBreakCor
```

```
##
##  Spearman's rank correlation rho
##
## data:  PrisonBreakTime$sumoftime and as.numeric(PrisonBreakTime$alive)
## S = 71.192, p-value = 0.08636
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5685352
```

For Harry Potter and Prison Break, we got, respectively, a p-value of 0.313 and 0.086.

Now let's use Biserial correlation (we have to implement the function too):

```
sd.pop = function(x){sd(x)*sqrt((length(x)-1)/length(x))}

biserial.cor.new =
function (x, y, use = c("all.obs", "complete.obs"), level = 2)
{
```

```
    if (!is.numeric(x))
        stop("'x' must be a numeric variable.\n")
    y <- as.factor(y)
    if (length(levs <- levels(y)) > 2)
        stop("'y' must be a dichotomous variable.\n")
    if (length(x) != length(y))
        stop("'x' and 'y' do not have the same length")
    use <- match.arg(use)
    if (use == "complete.obs") {
        cc.ind <- complete.cases(x, y)
        x <- x[cc.ind]
        y <- y[cc.ind]
    }
    ind <- y == levs[level]
    diff.mu <- mean(x[ind]) - mean(x[!ind])
    prob <- mean(ind)
    diff.mu * sqrt(prob * (1 - prob))/sd.pop(x)
}

HarryCor2<-biserial.cor.new(HarryTime$sumoftime, as.numeric(HarryTime$alive))
HarryCor2<-HarryCor2*(-100)
# HarryCor2
PrisonBreakCor2<-biserial.cor.new(PrisonBreakTime$sumoftime, as.numeric(PrisonBreakTime$alive))
PrisonBreakCor2<-PrisonBreakCor2*(-100)
# PrisonBreakCor2
Hd<-round(c(HarryCor$p.value, HarryCor$estimate, HarryCor2), digits = 3)
Pd<-round(c(PrisonBreakCor$p.value, PrisonBreakCor$estimate, PrisonBreakCor2), digits = 3)
cordf<-rbind(Hd, Pd)
rownames(cordf)<-c("Harry cor", "Prison Break cor")
colnames(cordf)<-c("P value", "rho", "Biserial cor")
cordf
```

```
##                  P value   rho Biserial cor
## Harry cor          0.314 0.355      -43.153
## Prison Break cor   0.086 0.569      -55.394
```

Usually, both tests aren't enough to reject the null hypothesis, but we can see some correlation between screen time and chance to stay alive.

In conclusion, we see that the screen time of characters can teach us a lot about their chances to survive and being the "winners" of the series. **of course it can also be coincidence, but it is still a factor to take under consideration when we do our study**.

In Game Of Thrones, we explore the data for every season and also for all of them togather. we plot the top 10 of characters and all houses.

In the graphs below we can see who are the most significant characters, screen-time wise, and also which house is the most dominant one. **Top ten of characters and houses are:**

```
top10<-c(1:10, 101:110, 201:210, 301:310, 401:410, 501:510, 601:610)
allseasonsTop10 <- ggplot(got.melt[top10,],
     aes(x = reorder(character, value, FUN = sum), y = value,
         fill=forcats::fct_rev(variable))) +
   geom_bar(stat = "identity",width = 0.7) + coord_flip() +
```

```
   theme_minimal(base_size=9) +
   labs(title="Screen time of characters in the Game of Thrones",
        subtitle="By seasons",
        x="characters",
        y="Screen time (minutes)", col="Season",
        fill=" ") + theme(axis.text=element_text(size=7)) + theme_grey(base_size = 6)+
  theme(axis.text.x = element_text(color = "grey20", size = 14, angle = 0, hjust = .5, vjust = .5, face
        axis.text.y = element_text(color = "grey20", size = 8, angle = 0, hjust = 1, vjust = 0, face =
        axis.title.x = element_text(color = "grey20", size = 12, angle = 0, hjust = .5, vjust = 0, face
        axis.title.y = element_text(color = "grey20", size = 12, angle = 90, hjust = .5, vjust = .5, fa
  theme(plot.title = element_text(size = 12, face = "bold"),
          plot.subtitle = element_text(size = 8),
          legend.text=element_text(size=9))
allseasonsTop10
```

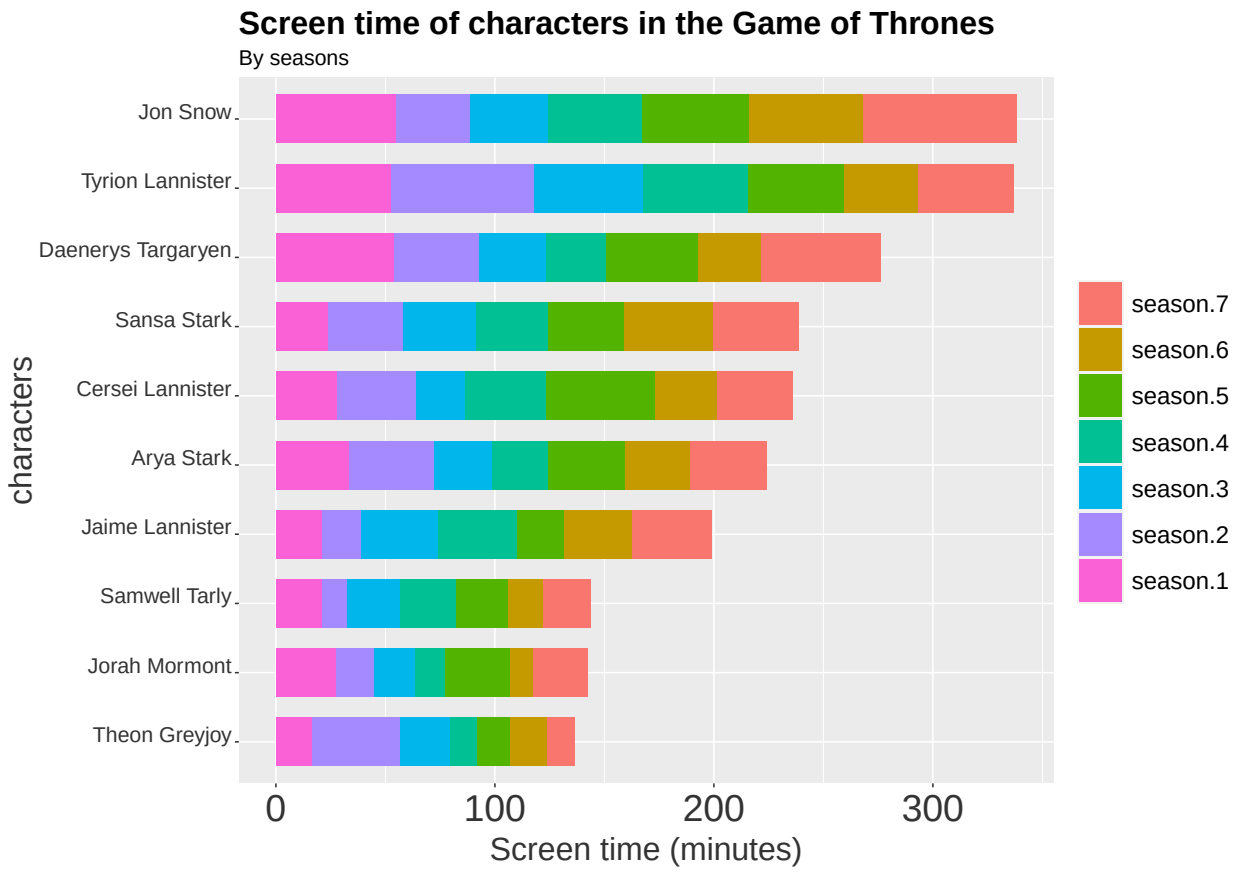


Figure 5: Fig 1.4: the top characters in Game of Thrones, screen-time wise

```
house.melt <- melt(house, id.vars='house')
```

```
## Warning in melt(house, id.vars = "house"): The melt generic in data.table has
## been passed a data.frame and will attempt to redirect to the relevant reshape2
## method; please note that reshape2 is deprecated, and this redirection is now
## deprecated as well. To continue using melt methods from reshape2 while both
## libraries are attached, e.g. melt.list, you can prepend the namespace like
```

```
## reshape2::melt(house). In the next version, this warning will become an error.
```

```
houseallseasons <- ggplot(house.melt,
       aes(x = reorder(house, value, FUN = sum), y = value,
           fill=forcats::fct_rev(variable))) +
  geom_bar(stat = "identity",width = 0.8) +
  coord_flip() +
  theme_minimal(base_size=10) +
  labs(title="Screen time of houses in the Game of Thrones",
       subtitle="By season",
       x="House",
       y="Time [m]",
       fill=" ") + theme(axis.text=element_text(size=11)) + theme_gray()
houseallseasons
```

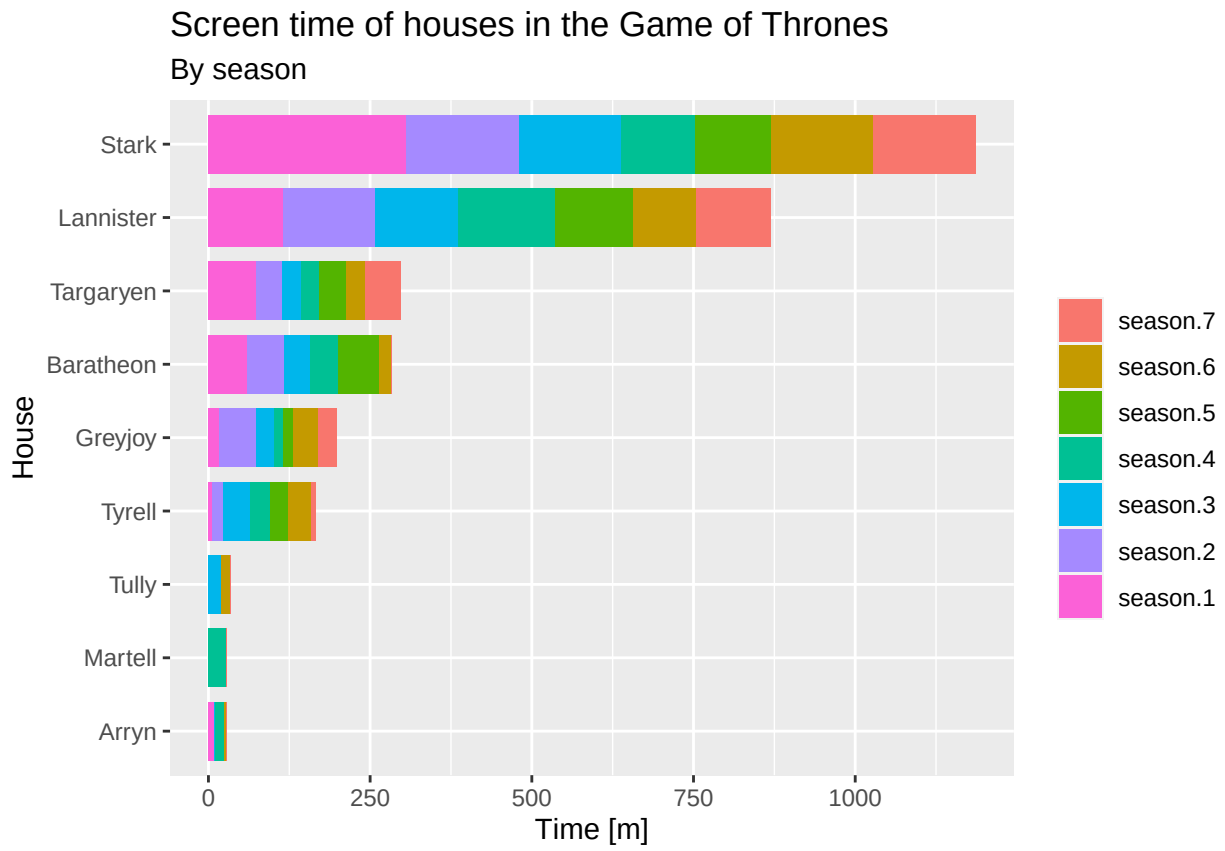


Figure 6: Fig 1.5: the top Houses in Game of Thrones, screen-time wise

Let's take this rating, and add it up to other sub-chapters and finally create our model, as we will soon explain.

Dominance in battles

It is plausible to assume that every "war-spirit" series would come to an end with some glorious battles. Therefore, a significant element in examining the chances of victory is the merit and ability to fight. To

do this, we will review the battles that have occurred in the first 7 seasons. We built a table of all the battles based on Game_of_Thrones_Wiki, and we took some significant data from each battle: Who is the attacker, who is the defender, who is the target of battle, who is victorious and more. Using this data, we will explore who has the best combat capabilities, and thus we will assess who has the best odds of winning in the end.

Let's see a few battles to get a feeling:

```
knitr::kable(battlesData[1:5,])
```

year	outcome_attack	aid_defender	defender_leader	aid_attacks	attacker_leader	name
298	1	Tully	Robb Stark	Lannister	Joffrey/Tommen Baratheon	Battle
298	1	Baratheon	Robb Stark	Lannister	Joffrey/Tommen Baratheon	Battle
298	1	Tully	Robb Stark	Lannister	Joffrey/Tommen Baratheon	Battle
298	-1	Frey	Joffrey/Tommen Baratheon	Stark	Robb Stark	Battle
298	1	Lannister	Joffrey/Tommen Baratheon	Stark	Robb Stark	Battle

There are plenty of interesting questions that come to mind, but not all of them are relevant to our topic. For example, you might be interested in whether the series has an advantage for attackers or defenders. This question is really interesting and can be investigated, but it does not contribute to our research question, since it is impossible to know who will eventually be the attacker and who will defend it.

In addition, it can be fascinating to look at how battles have evolved over the years, from 298 to 304, but again, this question doesn't really advance our research question.

So, we will focus on two main points that will help us with our research question:

A. Who is the character with the greatest winning ratio?

B. Who is the character with the most significant battle experience?

Let's start from the second point - who participated in the most battles:

```
#create ranking for experience
experience_atk<-as.vector(battlesData$attacker_leader)
experience_def<-as.vector(battlesData$defender_leader)
experience<-c(experience_atk,experience_def)
experience<-as.data.frame(table(experience))
experience<-experience[-1,]
experience<-experience[order(experience$Freq, decreasing = TRUE),]
colnames(experience)<-c("characters", "Nubmers of Battles")
```

And these are the most experienced fighters (What a pity most of them died . . .):

```
knitr::kable(experience[1:5,], row.names = FALSE)
```

characters	Nubmers of Battles
Joffrey/Tommen Baratheon	27
Robb Stark	24
Euron Greyjoy	12
Stannis Baratheon	9
Daenarys Targaryan	5

```
p <- ggplot(data = experience[1:5,], aes(x = experience[1:5,1], y = experience[1:5,2])) +
    geom_bar(stat="identity", fill="#56B4E9") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(experience[1:5,1])) +
    ggtitle("most experienced fighters")+
    labs(x="character", y="number of battles", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

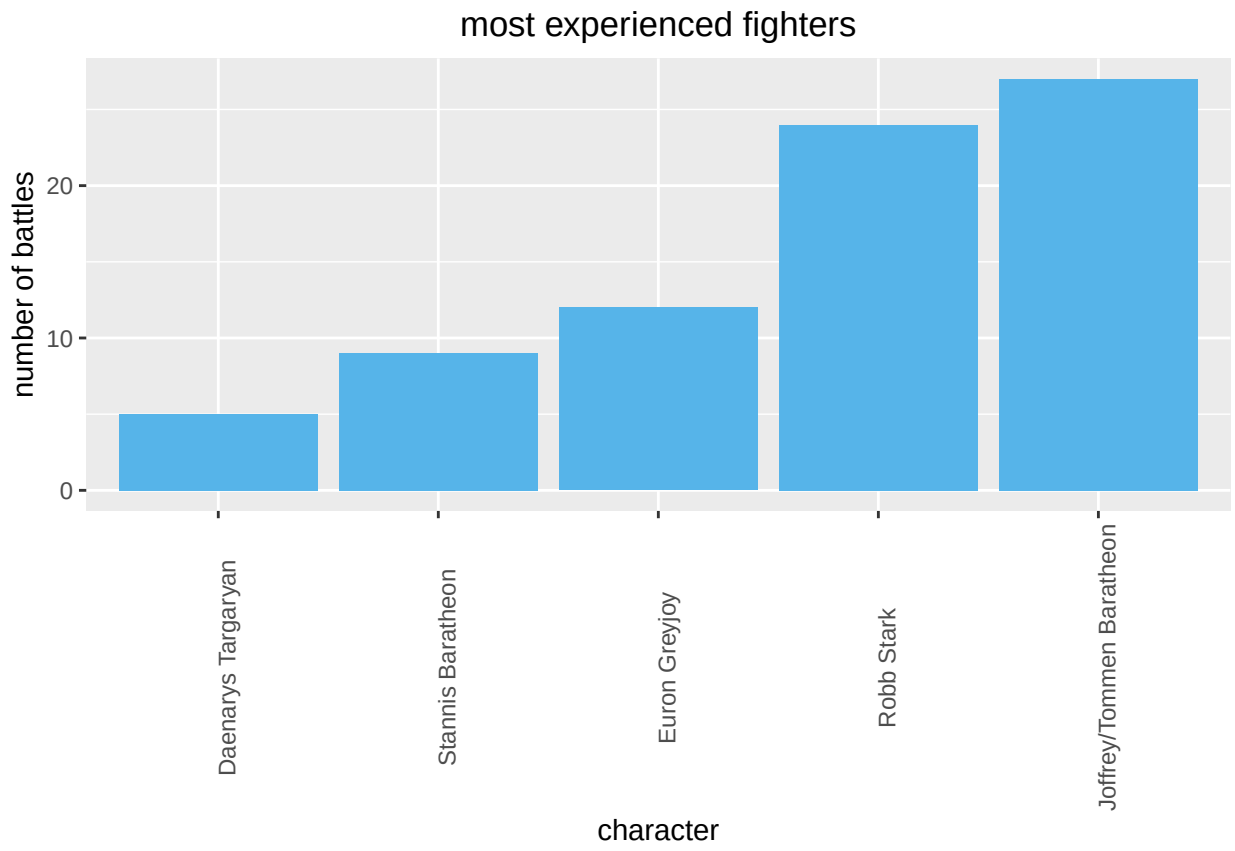


Figure 7: Fig 2.1: The most experienced fighters in Game of Thrones

Now let's see who has the best defense/attack ratio. We need to sort the data, and check out how many wins/losses each character has:

```
battlesMatrix<-matrix(0,126,3)
battlesMatrix[1:63,1]<-as.vector(battlesData$attacker_leader)
battlesMatrix[64:126,1]<-as.vector(battlesData$defender_leader)
colnames(battlesMatrix)<-c("character", "wins", "losses")
#merge losses and wins
for (i in 1:63){
  if (as.vector(battlesData$outcome_attack)[i] == 1)
  {
    battlesMatrix[i,2]<- 1
    battlesMatrix[i+63,3]<- 1
  }
```

```
  else
  {
    battlesMatrix[i,3]<- 1
    battlesMatrix[i+63,2]<- 1
  }
}

#create ranking for winners
battlesWin<-as.data.frame(battlesMatrix[,1:2])
battlesWin<-battlesWin[!(battlesWin$wins == "0"),]
battlesRankWin<-as.data.frame(table(battlesWin))
battlesRankWin<-battlesRankWin[!(battlesRankWin$wins == "0"),]
battlesRankWin<-battlesRankWin[,c(1,3)]
battlesRankWin<-battlesRankWin[order(battlesRankWin$Freq, decreasing = TRUE),]
battlesRankWin<-battlesRankWin[-8,] #delete null
colnames(battlesRankWin)<-c("character", "wins")

#create ranking for lossers
battleslose<-as.data.frame(battlesMatrix[,c(1,3)])
battleslose<-battleslose[!(battleslose$losses == "0"),]
battlesRankLose<-as.data.frame(table(battleslose))
battlesRankLose<-battlesRankLose[!(battlesRankLose$losses == "0"),]
battlesRankLose<-battlesRankLose[,c(1,3)]
battlesRankLose<-battlesRankLose[order(battlesRankLose$Freq, decreasing = TRUE),]
#delete null
battlesRankLose<-battlesRankLose[-4,]
colnames(battlesRankLose)<-c("character", "losses")
```

Top loosers:

```
knitr::kable(battlesRankLose[1:5,], row.names = FALSE)
```

character	losses
Robb Stark	15
Joffrey/Tommen Baratheon	10
Stannis Baratheon	6
Euron Greyjoy	4
Jaime Lannister	2

```
p <- ggplot(data = battlesRankLose[1:5,], aes(x = battlesRankLose[1:5,1], y = battlesRankLose[1:5,2])) +
    geom_bar(stat="identity", fill="#177ab2") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(battlesRankLose[1:5,1])) +
    ggtitle("Top loosers") +
    labs(x="character", y="number of losses", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

And top Winners:

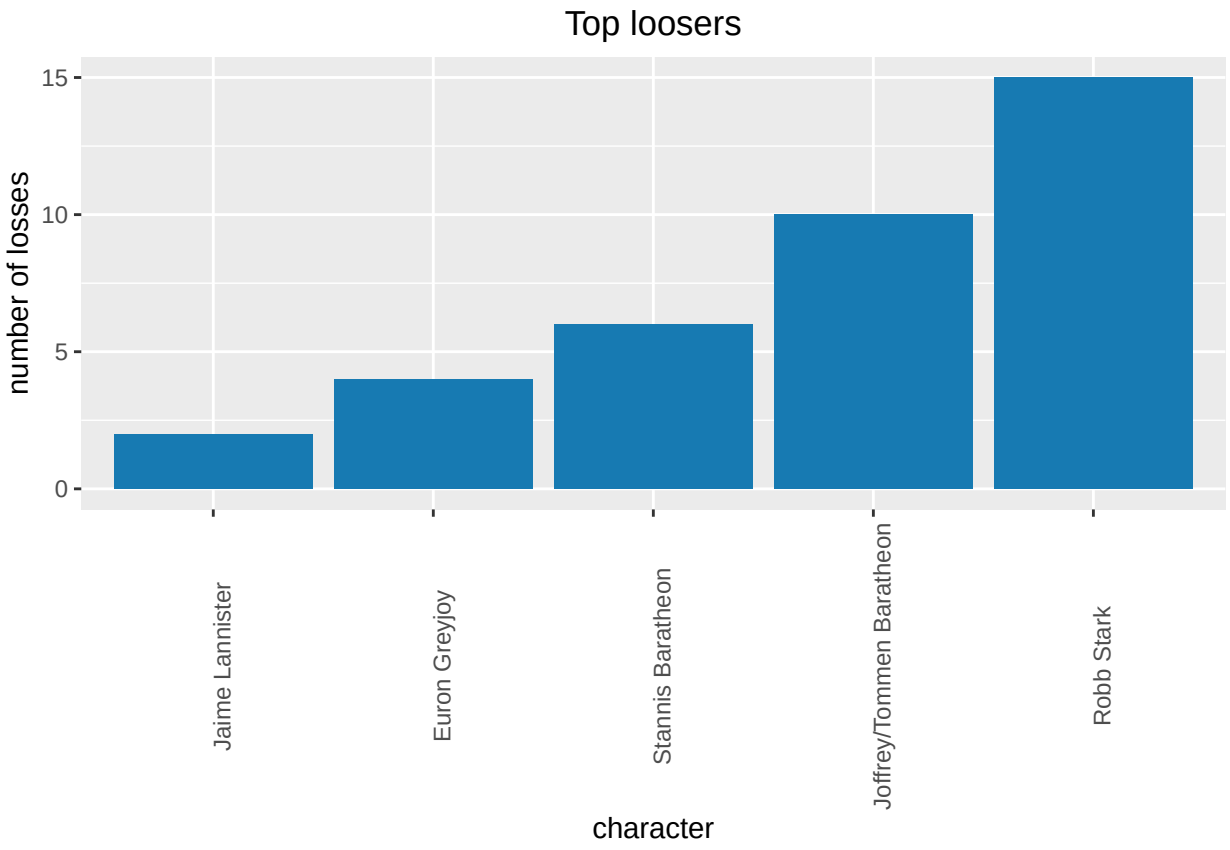


Figure 8: Fig 2.2: The characters who lost the most battles in the show

```
knitr::kable(battlesRankWin[1:5,], row.names = FALSE)
```

character	wins
Joffrey/Tommen Baratheon	17
Robb Stark	9
Euron Greyjoy	8
Daenarys Targaryan	5
Jon Snow	3

```
p <- ggplot(data = battlesRankWin[1:5,], aes(x = battlesRankWin[1:5,1], y = battlesRankWin[1:5,2])) +
    geom_bar(stat="identity", fill="#229de2") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(battlesRankWin[1:5,1])) +
    ggtitle("Top winners") +
    labs(x="character", y="number of wins", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

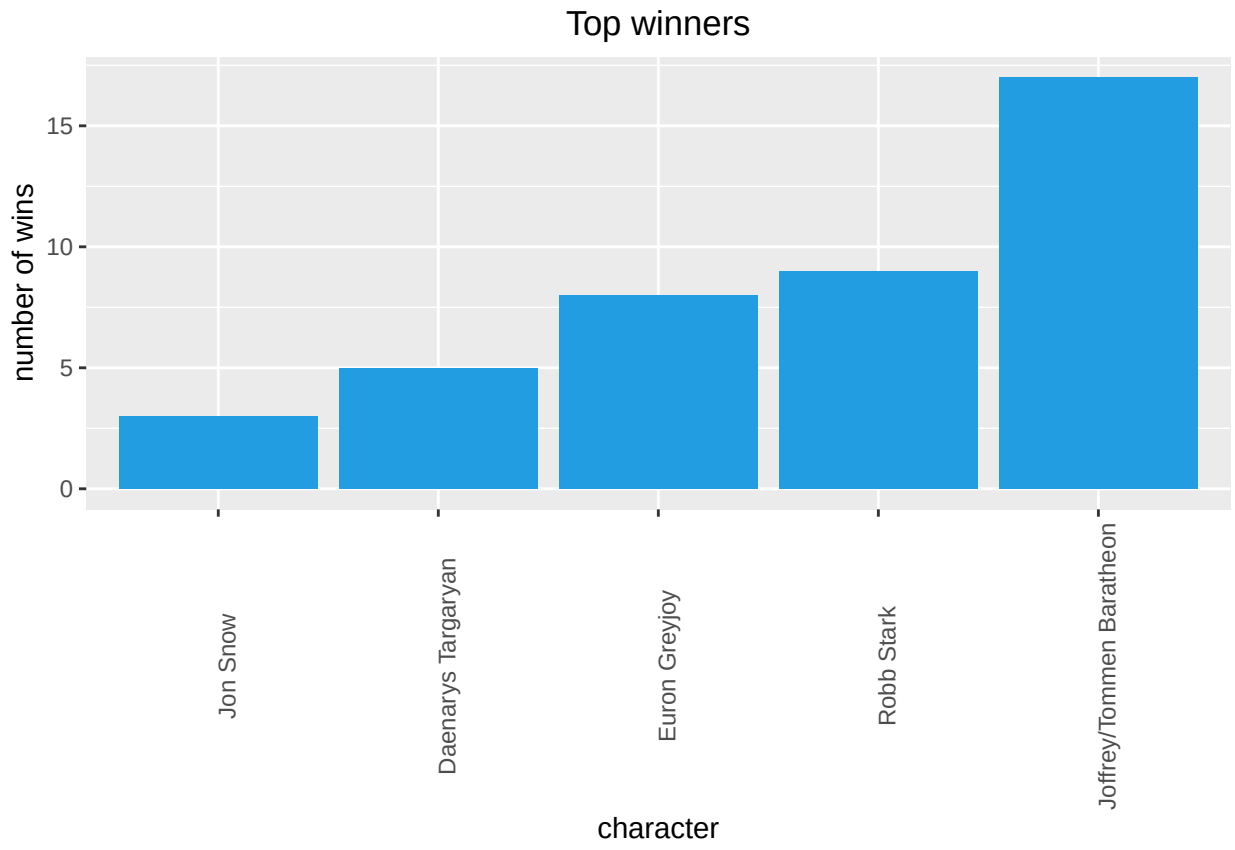


Figure 9: Fig 2.3: The characters who won the most battles in the show

Before we calculate the ratio, We have to take a moment of humor - Probably not enough to win 17 fights if one takes one sip of poisoned wine. . . :

```
par(bty = 'n', xaxt = 'n', yaxt = 'n', mar=c(8,8,1,8))
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_joffery,1,1,2,2)
```

Well, let's calculate ratio of wins/losses for every character:

```
#merge ranking of winners and lossers
battlesRank<-merge(battlesRankWin, battlesRankLose, by=0, all=TRUE)
battlesRank[is.na(battlesRank)] <- 0
battlesRank<-battlesRank[c(2:3,5)]
battlesRank[,4]<-0
colnames(battlesRank)<-c("character", "wins", "losses", "ratio")
#calculate the ratio
for (i in 1:length(battlesRank[,1])){
    #if the character has no losses, we have to put a number to avoid get NaN (number/0)
  if (battlesRank[i,3] != 0){
    battlesRank[i,4]<-round(battlesRank[i,2]/battlesRank[i,3], digits = 1)
  }
  else{
    battlesRank[i,4]<-battlesRank[i,2]
  }
}

#sort the ranking (by ratio and than by wins)
battlesRank<-battlesRank[order(battlesRank$ratio, battlesRank$wins, decreasing = TRUE),]
```

Top 10 of figthers by ratio wins/losses:

```
knitr::kable(battlesRank[1:10,-c(2,3)], row.names = FALSE)
```

character	ratio
Daenarys Targaryan	5.0
Jon Snow	3.0
Night King	3.0
Euron Greyjoy	2.0
Gryff Whitehill	2.0
Joffrey/Tommen Baratheon	1.7
Jaime Lannister	1.0
Arya Stark	1.0
Brynden Tully	1.0
Mance Rayder	1.0

```
top10exp<-matrix(0,20,3)
top10exp<-as.data.frame(top10exp)
colnames(top10exp)<-c("characters", "count", "wins/losses")
top10exp[1:10,1]<-as.vector(experience$characters[1:10])
top10exp[11:20,1]<-as.vector(experience$characters[1:10])
for (i in 1:10){
  top10exp[i,2]<-battlesRank[which(battlesRank$character == top10exp[i,1]),2]
  top10exp[i+10,2]<-battlesRank[which(battlesRank$character == top10exp[i,1]),3]
}
top10exp[1:10,3]<-"wins"
top10exp[11:20,3]<-"losses"
```

```
#plot using ggplot
p <- ggplot(data = top10exp, aes(x = top10exp$character, y = top10exp[,2], fill = top10exp[,3])) +
    geom_bar(stat="identity", ) + scale_fill_brewer(palette="Blues") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(top10exp[1:10,1])) +
    ggtitle("Ratio of wins/losses")+
    labs(x="character", y="number of battles", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

Now let's check the correlation between our two tables - experience and winnig ratio. To do this, we will check the rank of each character in each table, and than calculate the correlation between the ranks.

```
winRatio<-battlesRank[,-c(2,3)]
#matrix of ranks
ranks<-matrix(0,32,3)
colnames(ranks)<-c("character", "winnig ratio rank", "experience rank")
ranks[,1]<-as.vector(battlesRank[,1])
for (i in 1:32){
  char<-ranks[i,1]
  #winnig ratio
  ranks[i,2]<-which(winRatio$character == char)
  #experience rank
```

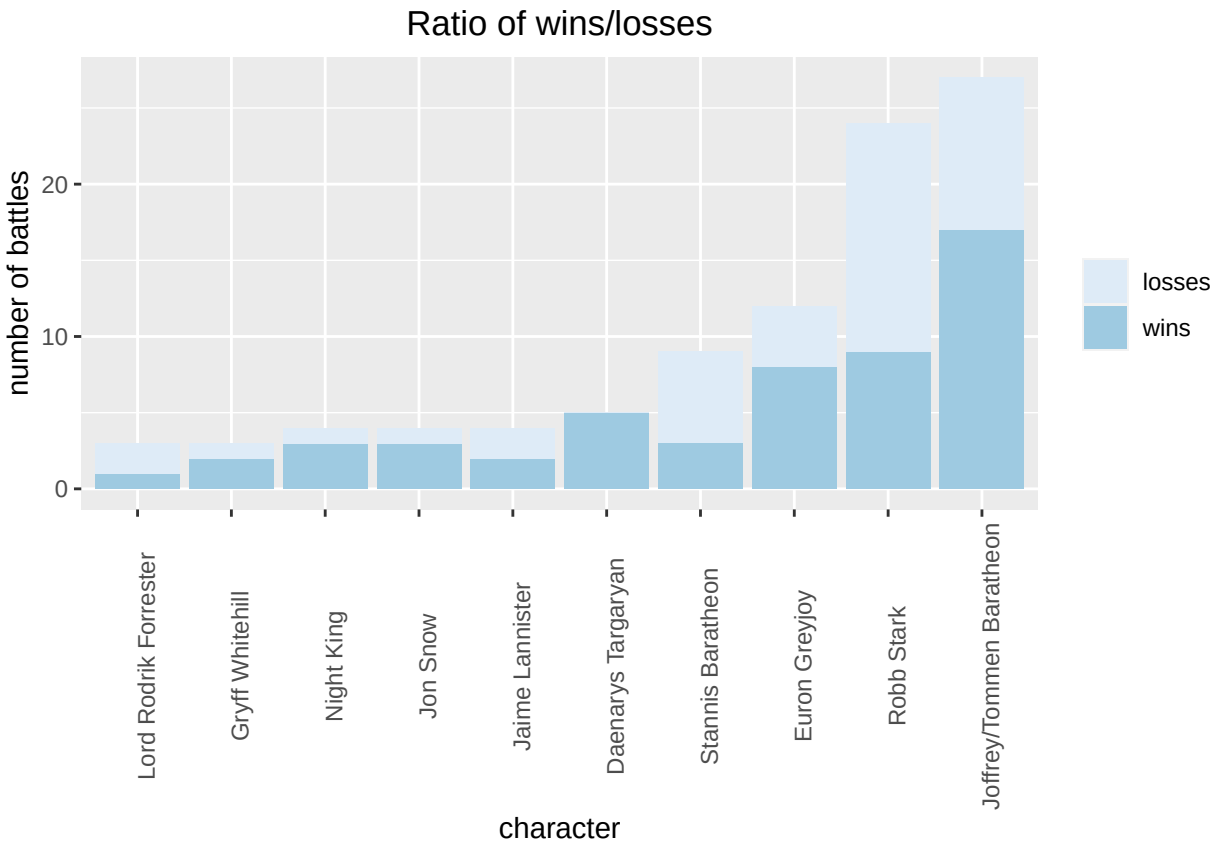


Figure 10: Fig 2.4: Ratio of the winning and loosing of characters in the show, sorted by number of battles

```
  ranks[i,3]<-which(experience$characters == char)
}


#calculate the correlation
ranksCor<-0

#Because we have long vectors, and we want to see the trend and not necessarily see that they are compl
#we will allow some differences (no more than 10)
for (i in 1:32){
  if (abs(as.numeric(ranks[i,2])-as.numeric(ranks[i,3])) <= 10)
  {
    ranksCor<-ranksCor + 1
  }
  else
  {
    ranksCor<-ranksCor - 1
  }
}

#and than the ratio of the correlation are:
ranksCor<-ranksCor/length(ranks[,1])

#we got 65% of correlation, and it's very well for the trend. this number allow us to combine the vecto
ranksTotal<-matrix(0,32,2)
colnames(ranksTotal)<-c("character", "grade")
ranksTotal[,1]<-as.vector(ranks[,1])
ranksTotal[,2]<-as.numeric(ranks[,2]) + as.numeric(ranks[,3])

#and finally, sort by ranking
ranksTotal<-ranksTotal[order(as.numeric(ranksTotal[,2])),]
ranksTotal<-as.data.frame(ranksTotal)
```

The top 10 according to an analysis of the series battles, taking into consideration both factors:

```
temp<-ranksTotal[,1]
ranksTotal[,1]<-ranksTotal[,2]
ranksTotal[,2]<-temp
colnames(ranksTotal)<-c("rank", "character")
ranksTotal[,1]<-1:length(ranks[,1])

knitr::kable(ranksTotal[1:10,])
```

rank	character
1	Daenarys Targaryan
2	Euron Greyjoy
3	Joffrey/Tommen Baratheon
4	Jon Snow
5	Night King
6	Jaime Lannister
7	Gryff Whitehill
8	Robb Stark

rank	character
9	Brynden Tully
10	Stannis Baratheon

```
bat1<-ranksTotal
```

```
p <- ggplot(data = ranksTotal[1:10,], aes(x = ranksTotal[1:10,2], y = ranksTotal[1:10,1])) +
    geom_bar(stat="identity", fill="cyan4") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits = rev(ranksTotal[1:10,2]))+
    ggtitle("Top 10") +
    labs(x="character", y="rank", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

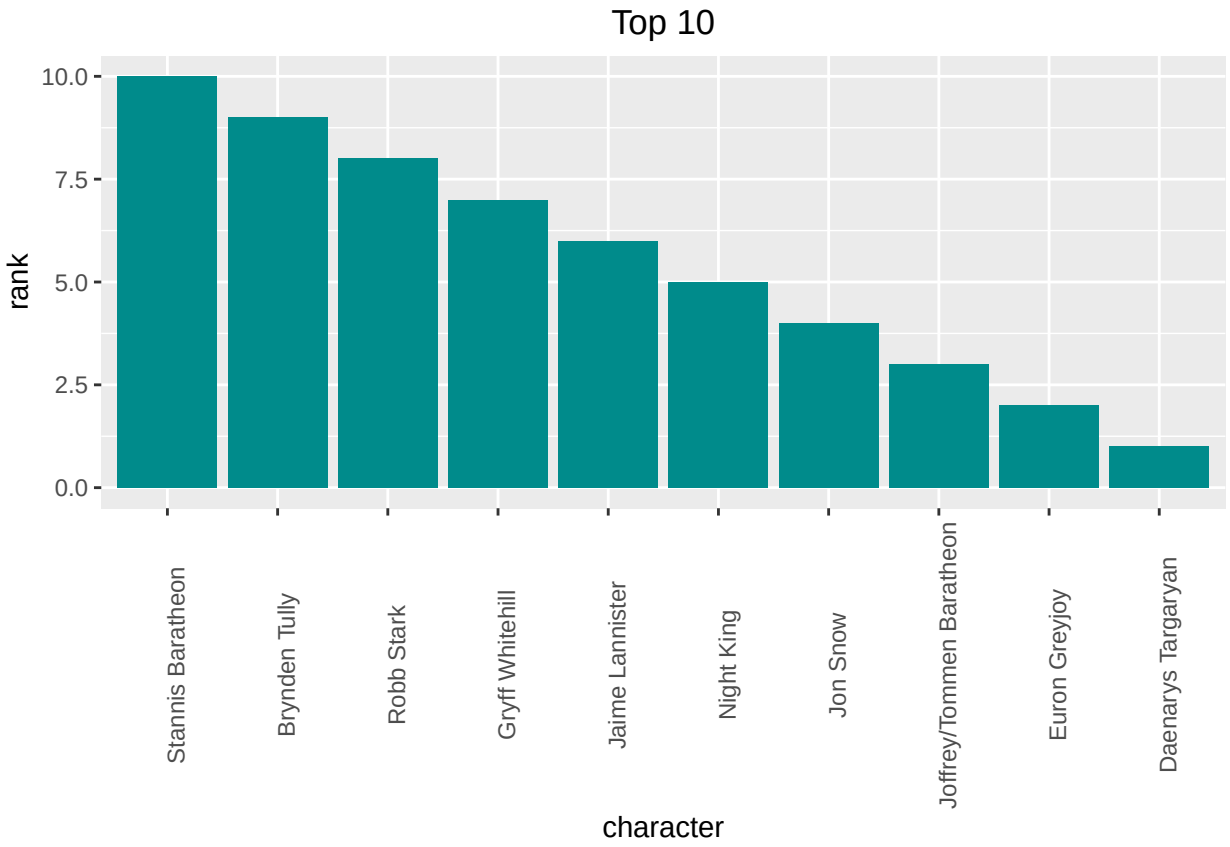


Figure 11: Fig 2.5: Top 10 characters taking into consideration both factors

We can go back one step and get a better correlation by using an algorithm similar to the ranking Blackman-TUkey ranking algorithm. This algorithm proposes to cut the "noisy" edges and thus improve the correlation. It is a clever algorithm to deal with outliers. When we say "noisy" we mean characters in which the ranking in one parameter is completely different from the ranking in the other parameter, which indicates an error in ranking this figure in at least one of the parameters, if not both (a detailed example will be given at the end, when we can see the difference between the naive ranking and the improved ranking). In fact, if we do this, we can see if some of the ratings we gave are not good. Let's look at this:

```
ranks<-cbind(ranks, 0)

#Our "Blackman-Tukey window" will be differnce of 10 or more between the two metrices
for (i in 1:length(ranks[,1])){
  if (abs(as.numeric(ranks[i,2])-as.numeric(ranks[i,3])) > 10){
    ranks[i,4]<-1
  }
}

ranks<-ranks[-c(which(ranks[,4] == "1")),]
ranks<-ranks[,-4]


#new correlation, with lower the band to 8
ranksCor<-0
for (i in 1:length(ranks[,1])){
  if (abs(as.numeric(ranks[i,2])-as.numeric(ranks[i,3])) <= 8)
  {
    ranksCor<-ranksCor + 1
  }
  else
  {
    ranksCor<-ranksCor - 1
  }
}
ranksCor<-ranksCor/length(ranks[,1])
```

Now, with Blackman-Tukey's improvement, we can even lower our margins and we achived correlation higher than 84% (instead of 65% before using Blackman-Tukey window)! much better! Let's see if it affects the final rating:

```
ranksTotal<-matrix(0,length(ranks[,1]),2)
colnames(ranksTotal)<-c("character", "grade")
ranksTotal[,1]<-as.vector(ranks[,1])
ranksTotal[,2]<-as.numeric(ranks[,2]) + as.numeric(ranks[,3])

#and finally, sort by ranking
ranksTotal<-ranksTotal[order(as.numeric(ranksTotal[,2])),]
ranksTotal<-as.data.frame(ranksTotal)
```

The top 10, after an analysis with Blackman-Tukey's improvement:

```
temp<-ranksTotal[,1]
ranksTotal[,1]<-ranksTotal[,2]
ranksTotal[,2]<-temp
colnames(ranksTotal)<-c("rank", "character")
ranksTotal[,1]<-1:length(ranks[,1])

bat2<-ranksTotal[1:10,2]
batdf<-cbind(bat1[1:10,], bat2)
colnames(batdf)<-c("rank", "character (before BT)", "character (after BT)")
knitr::kable(batdf)
```

rank	character (before BT)	character (after BT)
1	Daenarys Targaryan	Daenarys Targaryan
2	Euron Greyjoy	Euron Greyjoy
3	Joffrey/Tommen Baratheon	Joffrey/Tommen Baratheon
4	Jon Snow	Jon Snow
5	Night King	Night King
6	Jaime Lannister	Jaime Lannister
7	Gryff Whitehill	Gryff Whitehill
8	Robb Stark	Brynden Tully
9	Brynden Tully	Arya Stark
10	Stannis Baratheon	Lord Rodrik Forrester

```
#plot
p <- ggplot(data = ranksTotal[1:10,], aes(x = ranksTotal[1:10,2], y = ranksTotal[1:10,1])) +
    geom_bar(stat="identity", fill="cyan4") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(ranksTotal[1:10,2])) +
    ggtitle("Top 10 (using Blackman-Tukey algorithm)")+
    labs(x="character", y="rank", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

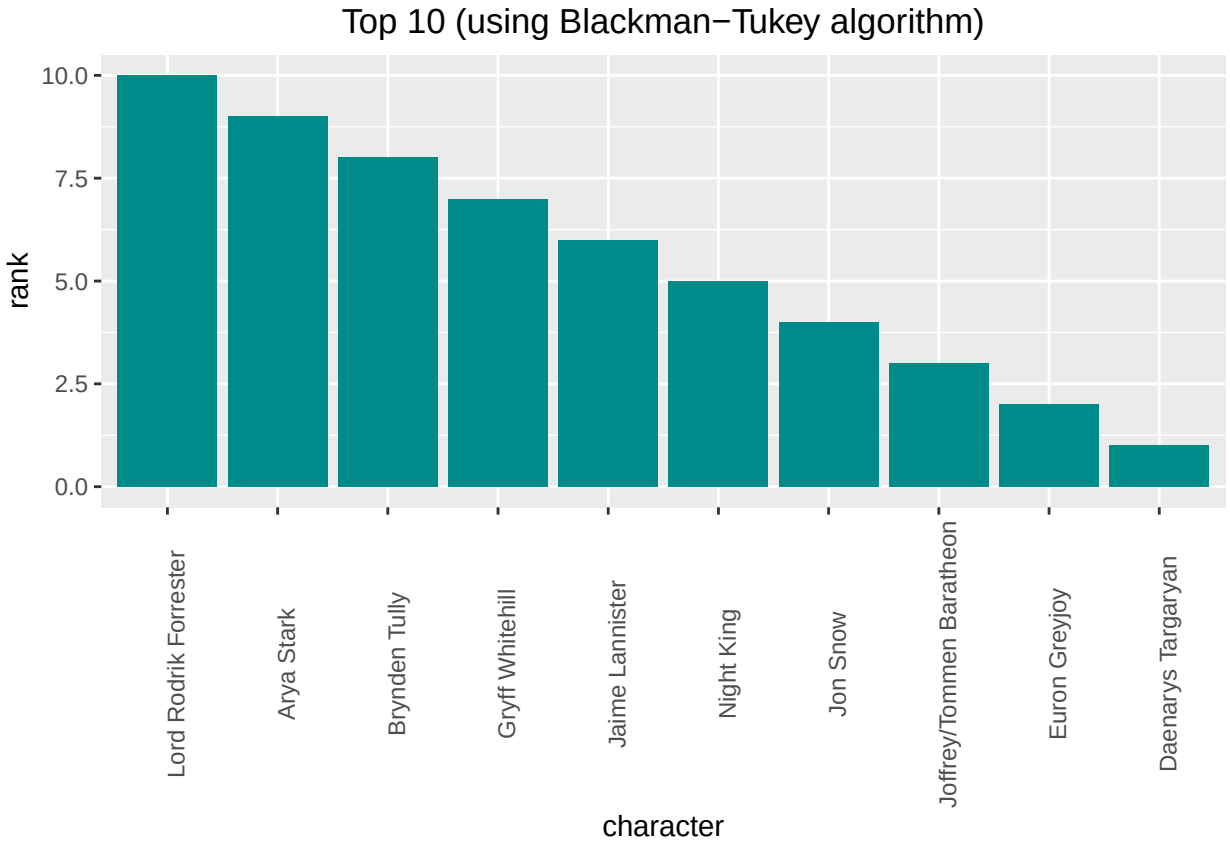


Figure 12: Fig 2.6: Top 10 characters taking into consideration both factors, with Blackman-Tukey algorithm

When we come to look at the difference we got between the initial ranking and the ranking made using the correlation enhancement algorithm, we can note that Robb Stark comes out of the top ten. This difference is well understood when you look at his ranking - He has a lot of experience in battles, which ranks him second in this category, but at the same time a lot of them are not victorious, so his wins / losses ratio is pretty low. Therefore, we have two rankings for him, but they are very far from each other, which makes us suspect that at least one is unreliable. This suspicion is actually expressed directly through the Blackman-Tukey algorithm, and Robb is deleted from the list.

Let's do the same investigation to the houses, and check what house is the most dominant:

```
#experience
experience_atk_houses<-as.vector(battlesData$aid_attacks)
experience_def_houses<-as.vector(battlesData$aid_defender)
experience_houses<-c(experience_atk_houses,experience_def_houses)
experience_houses<-as.data.frame(table(experience_houses))
experience_houses<-experience_houses[-1,]
experience_houses<-experience_houses[order(experience_houses$Freq, decreasing = TRUE),]
colnames(experience_houses)<-c("house", "Nubmers of Battles")
experience_houses<-experience_houses[-which(experience_houses$`Nubmers of Battles` == 1),]
experience_houses<-experience_houses[-2,]

knitr::kable(experience_houses[1:11,], row.names = FALSE)
```

house	Nubmers of Battles
Stark	24
Lannister	19
Baratheon	12
Greyjoy	12
Bolton	6
Tully	6
Frey	5
Tyrell	3
Forrester	2
Mormont	2
Targaryan	2

```
p <- ggplot(data = experience_houses[1:11,], aes(x = experience_houses[1:11,1], y = experience_houses[1
    geom_bar(stat="identity", fill="magenta3") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(experience_houses[1:11,1])) +
    ggtitle("most experienced houses")+
    labs(x="house", y="number of battles", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

```
#winning ratio
ratioMatrix<-matrix(0,22,3)
colnames(ratioMatrix)<-c("house", "num of wins/losses", "wins/losses")
ratioMatrix<-as.data.frame(ratioMatrix)
ratioMatrix[1:11,1]<-as.character(experience_houses[1:11,1])
ratioMatrix[12:22,1]<-as.character(experience_houses[1:11,1])
```

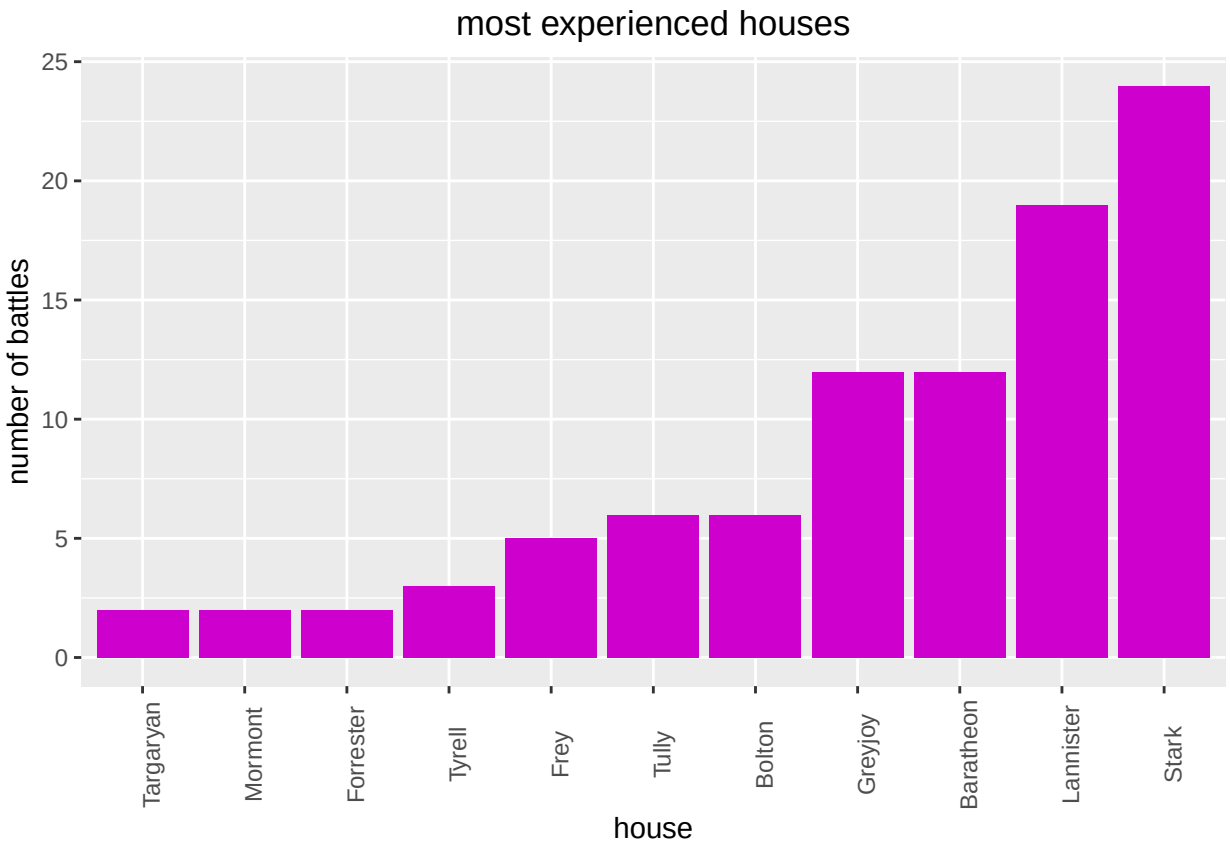


Figure 13: Fig 2.7: The most battle-experienced Houses in the show

```
for (i in 1:11){
  ratioMatrix[i,2]<-sum(battlesData$outcome_attack[(battlesData$aid_attacks == ratioMatrix[i,1])] == 1)
}
for (i in 1:11){
  ratioMatrix[i+11,2]<-sum(battlesData$outcome_attack[(battlesData$aid_attacks == ratioMatrix[i,1])] ==
}
ratioMatrix[1:11,3]<-"wins"
ratioMatrix[12:22,3]<-"losses"

p <- ggplot(data = ratioMatrix, aes(x = ratioMatrix$house, y = ratioMatrix[,2], fill = ratioMatrix[,3]))
    geom_bar(stat="identity", ) + scale_fill_brewer(palette="PuRd") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_x_discrete(limits=rev(ratioMatrix[1:11,1])) +
    ggtitle("Ratio of wins/losses per house")+
    labs(x="house", y="number of battles", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

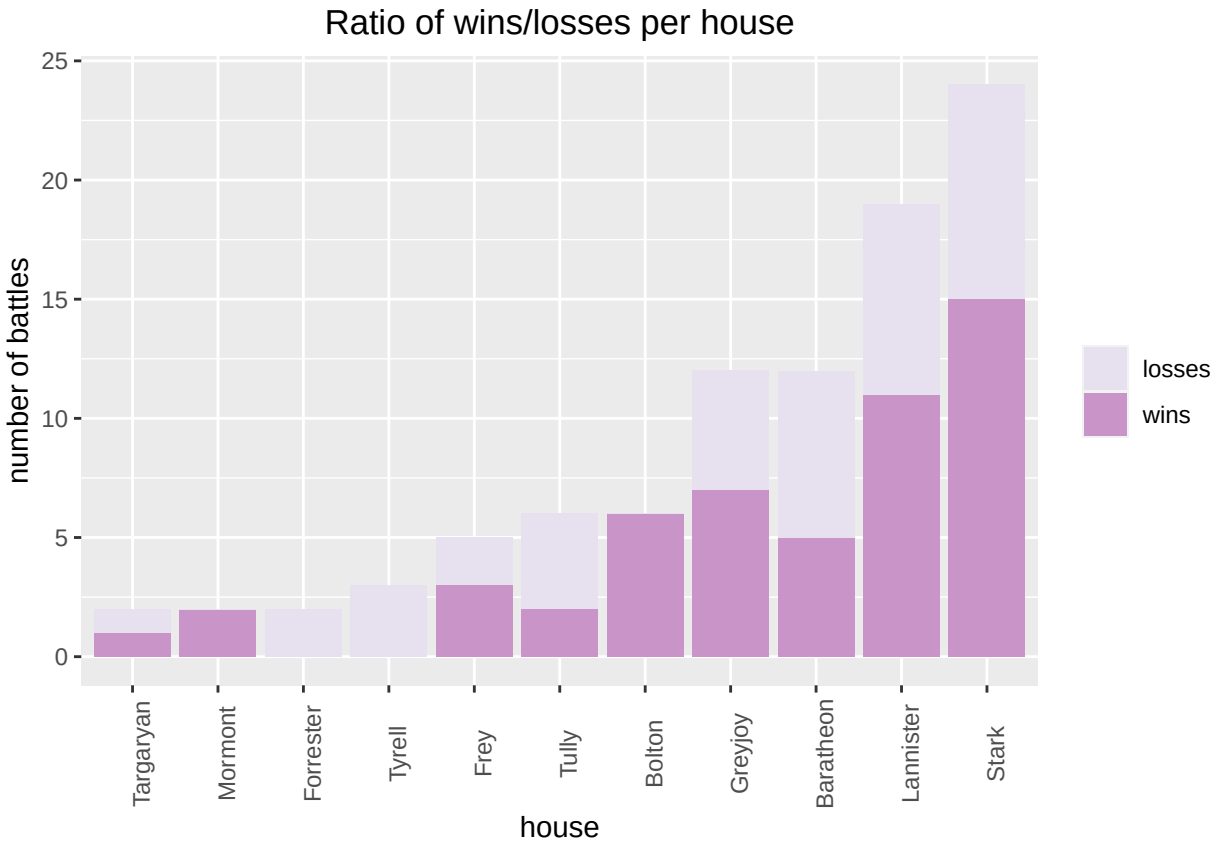


Figure 14: Fig 2.7: ration of win-lose for houses

In summary, there are a number of fascinating questions that arise when you want to analyze the information about the battles, but most are irrelevant to our research question. For the research question, we focused on two issues - experience in battles, and the ratio between wins and losses. When we came to weigh both parameters, we initially did it naively, and then improved the result with inspiration from Blackman-Tukey algorithm to filter out incorrect correlation values.

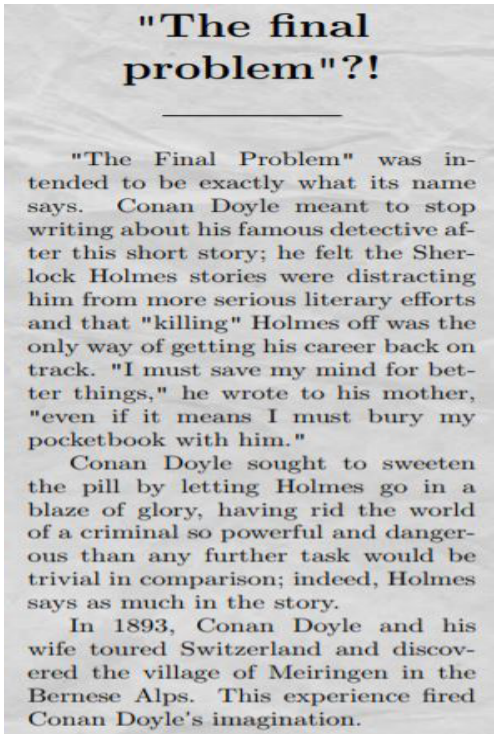
The audience's Opinion

```
par(bty = 'n', xaxt = 'n', yaxt = 'n')
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_survey,1,1,2,2)
```

Before we show our analysis, we will add two interesting documents that show that public opinion has an influence on the creators and the course of events, and therefore one can infer from the wisdom of the crowd about the future that has not yet been discovered in the series.

The first document concerns the Sherlock Holmes series, and we can learn that Sherlock Holmes resurrection in the wake of public opinion:

```
par(bty = 'n', xaxt = 'n', yaxt = 'n', mfrow=c(1,2), mar = c(1, 0.5, 1, 2))
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_sherlock1,1,1,2,2)
plot(2:1, type='n',ylab = '', xlab = '')
rasterImage(img_sherlock2,1,1,2,2)
```

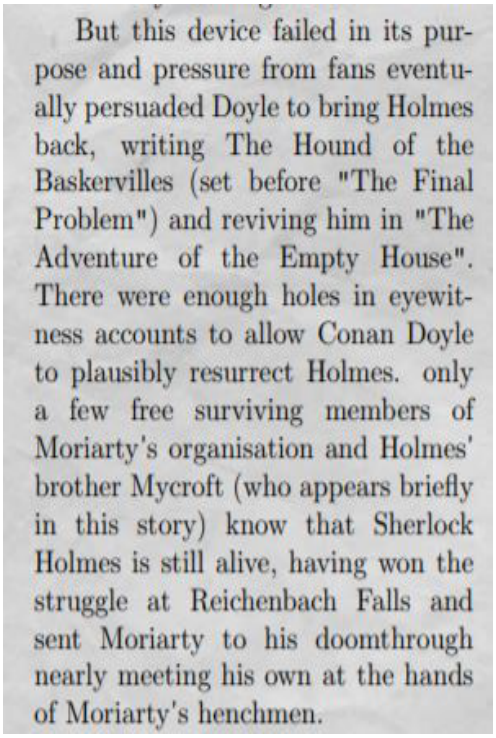


"The final problem"?!

"The Final Problem" was intended to be exactly what its name says. Conan Doyle meant to stop writing about his famous detective after this short story; he felt the Sherlock Holmes stories were distracting him from more serious literary efforts and that "killing" Holmes off was the only way of getting his career back on track. "I must save my mind for better things," he wrote to his mother, "even if it means I must bury my pocketbook with him."

Conan Doyle sought to sweeten the pill by letting Holmes go in a blaze of glory, having rid the world of a criminal so powerful and dangerous than any further task would be trivial in comparison; indeed, Holmes says as much in the story.

In 1893, Conan Doyle and his wife toured Switzerland and discovered the village of Meiringen in the Bernese Alps. This experience fired Conan Doyle's imagination.



But this device failed in its purpose and pressure from fans eventually persuaded Doyle to bring Holmes back, writing The Hound of the Baskervilles (set before "The Final Problem") and reviving him in "The Adventure of the Empty House". There were enough holes in eyewitness accounts to allow Conan Doyle to plausibly resurrect Holmes. only a few free surviving members of Moriarty's organisation and Holmes' brother Mycroft (who appears briefly in this story) know that Sherlock Holmes is still alive, having won the struggle at Reichenbach Falls and sent Moriarty to his doomthrough nearly meeting his own at the hands of Moriarty's henchmen.

Well, apparently “pressure from fans” is not a trivial matter.

We want to show another example, from a series we have already met - Prison Break.

Sarah Wayne Callies, better known by its name in the series - Dr. Sara Tancredi, was forced to die in the 3rd season due to her pregnancy (or according to another version - conflict with production).

However, following the protest of the series’ fans, she was returned to the fourth season, and she was once again one of the most central characters. for example, you can read the report here - http://whatculture.com/tv/10-outrageous-ways-famous-tv-characters-came-back-from-the-dead?page=11:

```
par(bty = 'n', xaxt = 'n', yaxt = 'n')
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_Sara,1,1,2,2)
```

She Was Dead...: In season three Sara has been kidnapped by the evil company chasing the brothers. They use her as bait to make the two men do their bidding, but when they fail a task she is beheaded as punishment. Her head is sent in a box to the brothers in episode four, and Michael swears he'll avenge her death.

...But She Got Better: It turns out he didn't need to make the effort, as season four reveals her death was faked. She escaped and a corpse was used for the head in a box. It was a real stroke of luck that the head looked exactly like Sara too.

Sara's abrupt death was in response to a contract dispute with actress Sarah Wayne Callies that couldn't be worked out, but due to a huge fan backlash the show brought her back a season later with her head reattached.

We will now analyze the survey (with 944 responders). Taking into consideration the examples we have seen and with the understanding that the viewers have an influence, we will try to get some predictions concerning Game Of Thrones.

In the survey there were two main questions that will concern us. Therefore, our data contains two numbers for each character (its ranking). We will explain later how we intend to analyze this data.

A. Who do you think will sit on the iron throne at the end of the series?

```
sortWinner<-sort(summary(surveyData$winner[1:944]))
sortWinner<-data.frame(sortWinner)
sortWinner<-data.frame(t(sortWinner))
labels = colnames(sortWinner)
rownames(sortWinner)<-"Votes"

knitr::kable(sortWinner[,-1])
```

	Jaime.Lannister	Bran.Stark	Sansa.Stark	Arya.Stark	Cersei.Lannnister	Night.King	Tyrion.Lannister	D
Votes	40	45	65	86	91	94	119	

```
par(mfrow = c(1,1), mgp=c(8, 1, 0), las=2, mar=c(10,3,3,3))
plot(factor(sortWinner), sortWinner, main="survey: who will win the series?", type='h',lwd=33, col = "pu
ylab="Votes for", xaxt = "n", xlab = "character")
axis(1, at=1:10, labels=labels)
```

```
grid(col = "blue")
```

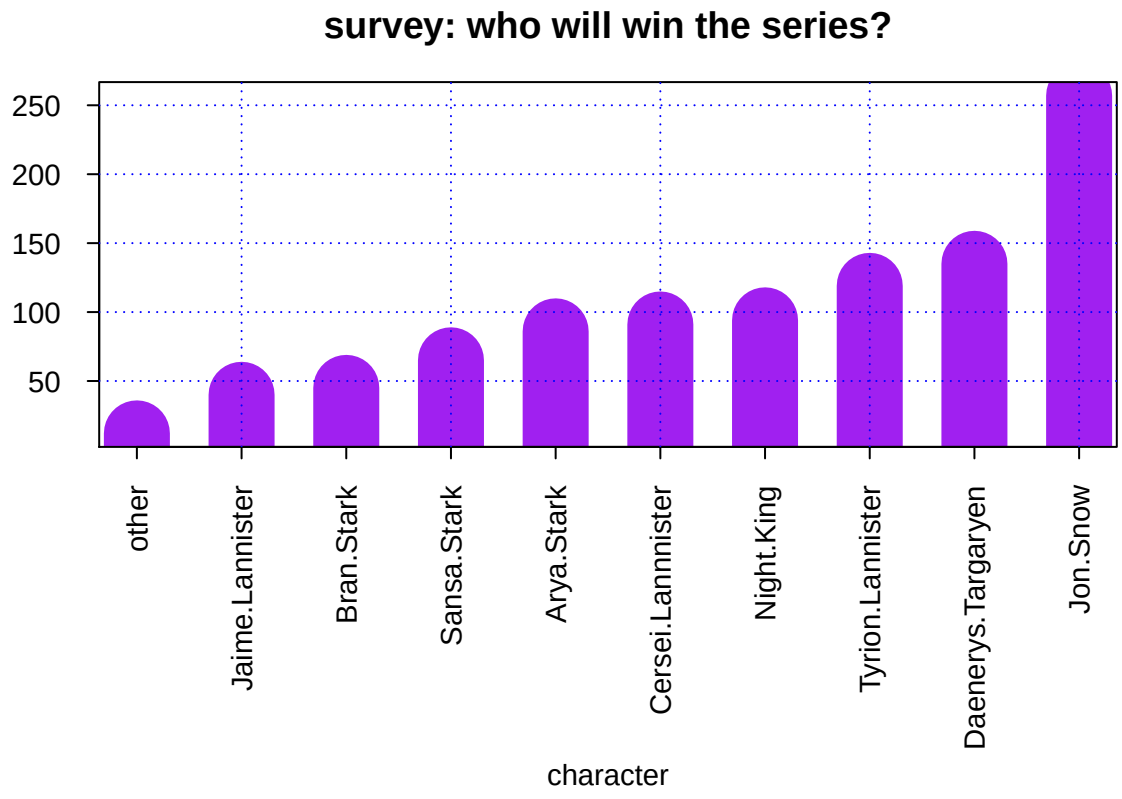


Figure 15: Fig 3.1: Results of the survey, of who the audience thinks will win the show

Wow, Jon Snow is definitely the character with the greatest chances to win according to public opinion!

in the second and third place, we can find Daenerys Targaryen and Tyrion Lannister, respectively.

B. For the main characters, do you think they will survive or will they die?

```
#Add a row that summarizes the results about the question of survival
surveyData[945,2:14]<-apply(surveyData[1:944,2:14], 2, sum)
#Normalizied (divide by number of answers - 944, and multiply by 100 to getting the answer by percent)
surveyData[946,2:14]<-surveyData[945,2:14]*100/944

sortSurvey<-sort(surveyData[946,2:14])
labels<-colnames(sortSurvey)
rownames(sortSurvey)<-"Percentage survival by votes"

knitr::kable(round(sortSurvey, digits = 2))
```

	Cersei.Lannnister	Night.King	Daenerys.Targaryen	Jaime.Lannister	Bran.Stark	Ser.
Percentage survival by votes	22.78	27.65	36.12	48.2	51.17	

```
par(mfrow = c(1,1), mgp=c(9, 1, 0), las=2, mar=c(10,3,3,3))
plot(factor(sortSurvey), sortSurvey, main="survey: who will survive?", type='h',lwd=33, col = "purple",
ylab="Votes", xaxt = "n", xlab = "character name")
axis(1, at=1:13, labels=labels)
grid(col = "blue")
```

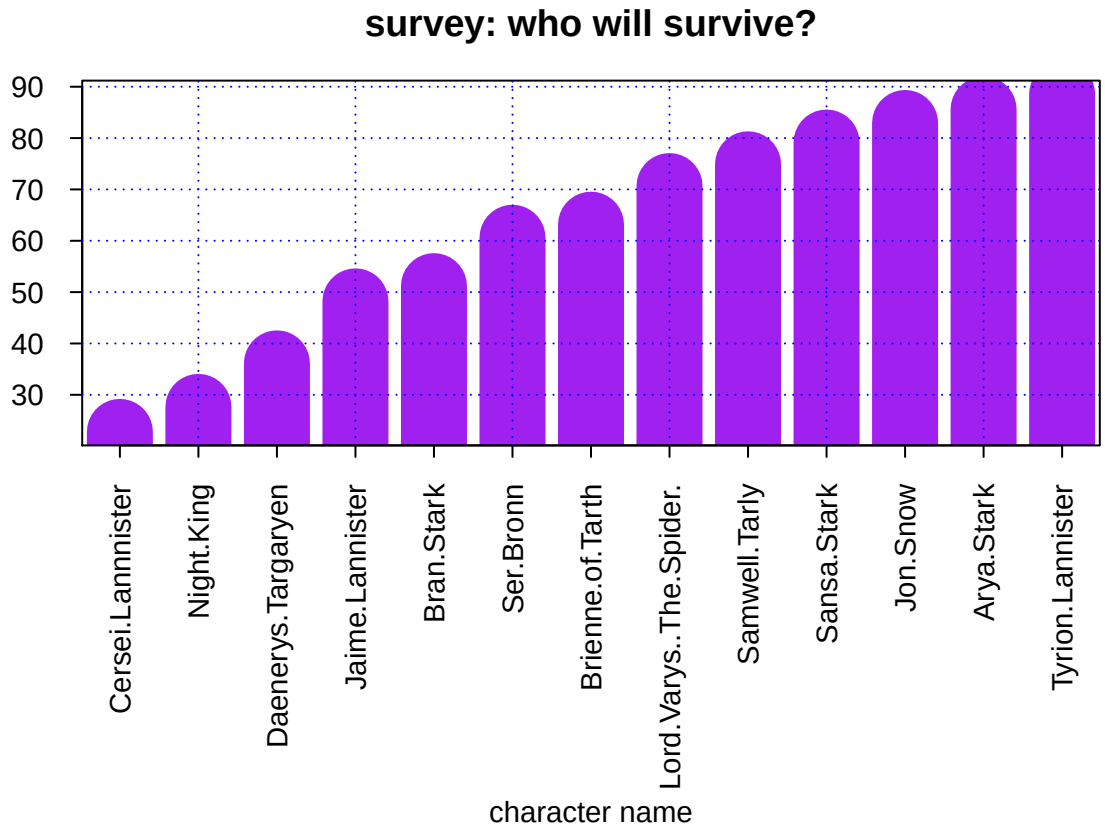


Figure 16: Fig 3.2: Results of the survey, of who the audience thinks will survive at all in the eighth season

Now let's try to combine those two graphs, and first explain how we do this with an example:

Jon Snow is the character most people think will win (1/9), but only third in the second category (3/13), therefore, he will receive 9 points from the first parameter and 11 from the second parameter then his weighted score is 20 (+epsilon, as we shown in the comments of the following code).

```
#delete "other"
sortWinner<-sortWinner[1,2:10]
#add values
sortWinner[1,1:9]<-c(1:9)
sortSurvey[1,1:13]<-c(1:13)

#moving the "others" in second survey
sortSurvey[1,6:9]<-sortSurvey[1,10:13]

#calculate the sum
sum<-data.frame(sortWinner)
```

```
sum$Jaime.Lannister<-sum$Jaime.Lannister+sortSurvey$Jaime.Lannister
sum$Bran.Stark<-sum$Bran.Stark+sortSurvey$Bran.Stark
sum$Sansa.Stark<-sum$Sansa.Stark+sortSurvey$Sansa.Stark
sum$Arya.Stark<-sum$Arya.Stark+sortSurvey$Arya.Stark
sum$Cersei.Lannnister<-sum$Cersei.Lannnister+sortSurvey$Cersei.Lannnister
sum$Night.King<-sum$Night.King+sortSurvey$Night.King
sum$Tyrion.Lannister<-sum$Tyrion.Lannister+sortSurvey$Tyrion.Lannister
sum$Daenerys.Targaryen<-sum$Daenerys.Targaryen+sortSurvey$Daenerys.Targaryen
sum$Jon.Snow<-sum$Jon.Snow+sortSurvey$Jon.Snow

#since Jon and Tyrion have both the same rank, we will separate by epsilon
epsilon<-0.5
sum$Jon.Snow<-sum$Jon.Snow+epsilon
sum$Tyrion.Lannister<-sum$Tyrion.Lannister-epsilon

sum<-sort(sum)
labels<-colnames(sum)
rownames(sortSurvey)<-"Final score"

knitr::kable(sum)
```

	Jaime.Lannister	Cersei.Lannnister	Bran.Stark	Night.King	Daenerys.Targaryen	Sansa.Stark	Arya.Stark
Votes	5	6	7	8	11	13	16

```
par(mfrow = c(1,1), mgp=c(6, 1, 0), las=2, mar=c(10,3,3,3))
plot(factor(sum), sum, main="sum of survey", type='h',lwd=33, col = "purple",
ylab="Votes", xaxt = "n", xlab = "")
axis(1, at=1:9, labels=labels)
grid(col = "blue")
```

Familiy ties

To complete the image, we'd like to add another chapter based on previous work from the site Shirin's playgRound. Shirin investigated the subject of family ties in the series by building a network, and in this chapter we present the main points.

You can find the full work here:

https://shiring.github.io/networks/2017/05/15/got_final

The premise is that the more relationships one has, the more important and dominant it is. The question is how to check the quantity and quality of those connections.

We take several parameters, most of them are derived from Graph Theory:

- The first significant parameter is how many nodes are connected directly to a given node? (its degree)
- Another significant parameter is what are the nodes to which the character is connected?
- Third parameter is "Betweenness" - measure of centrality in a graph based on shortest paths. Practically it is the number of "shortest paths" going through a given node.

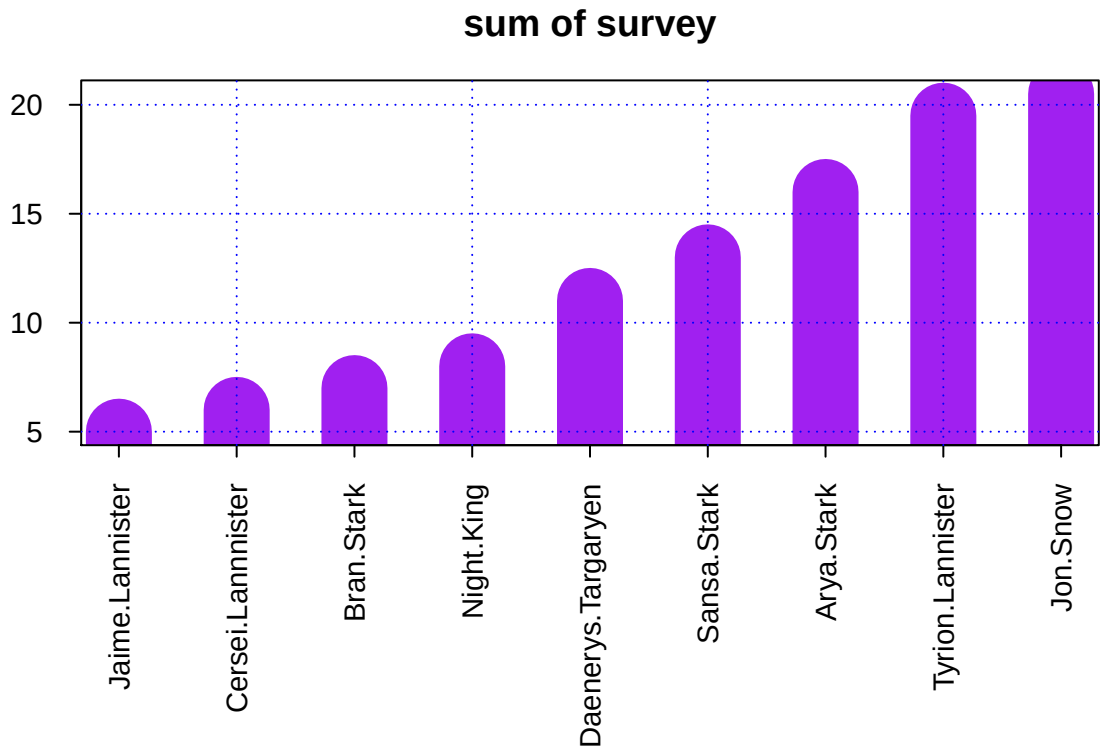


Figure 17: Fig 3.3: final Results of the survey we cunducted

- Last parameter is the eigenvectors of the networks, which could tell us about the importance of a character.

```
par(bty = 'n', xaxt = 'n', yaxt = 'n')
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_famaily,1,1,2,2)
```

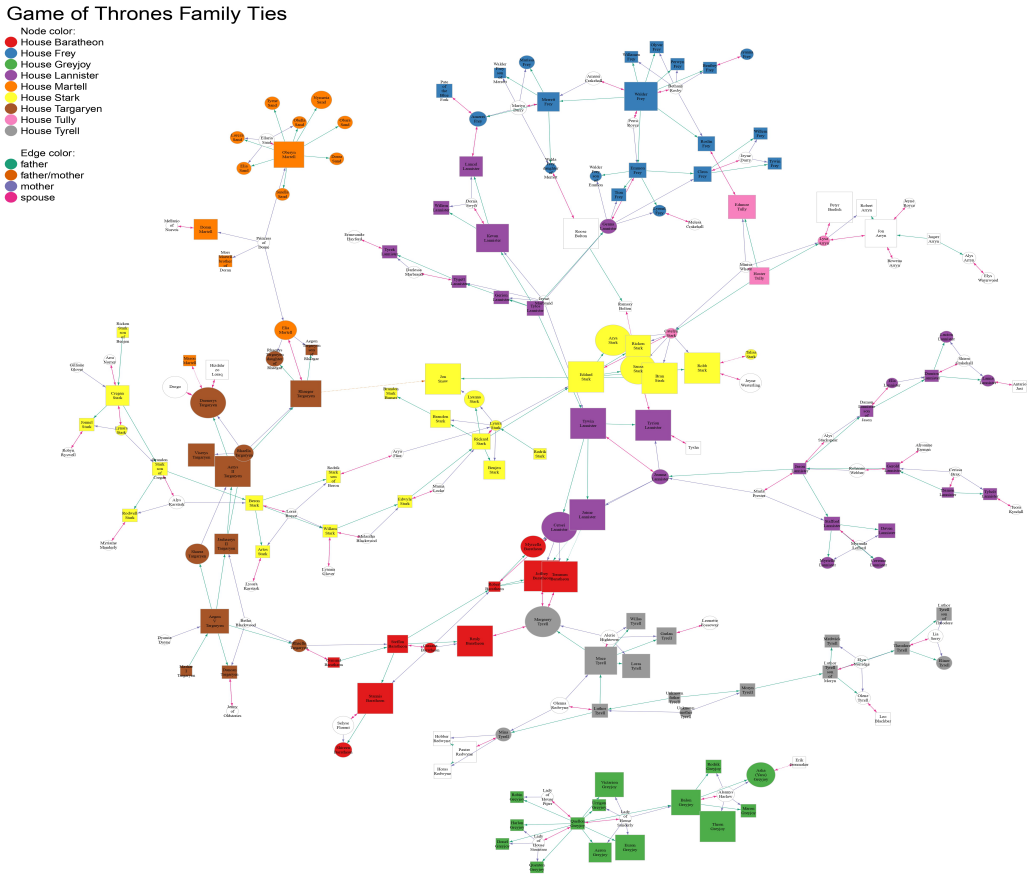


Figure 18: Fig 4.1: family trees and connection of the Game Of Thrones Houses

We can see above the general tree of families, and how they connect. The size of the node represents the character's popularity, and edge color shows different interception types.

Now, there's a principal, called "Centrality", by which we can persume which character is more central. it is done by evaluating the nodes with many connections. We can do it for the entire network, or for a specific node. In practice, it measures how many offsprings and spouses a character has. It desribes the amount of shortest paths between nodes, which can assume those nodes are key connection between families.

There's another factor that's called "Closeness" - which means the distance to all other nodes. the assumption is that a node with the highest "closeness" is more central and can spread information and influence easily. We can evaluate that Sansa Stark and Tyrion Lannister are with the highest closeness.

```
par(bty = 'n', xaxt = 'n', yaxt = 'n')
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_betweenness,1,1,2,2)
```

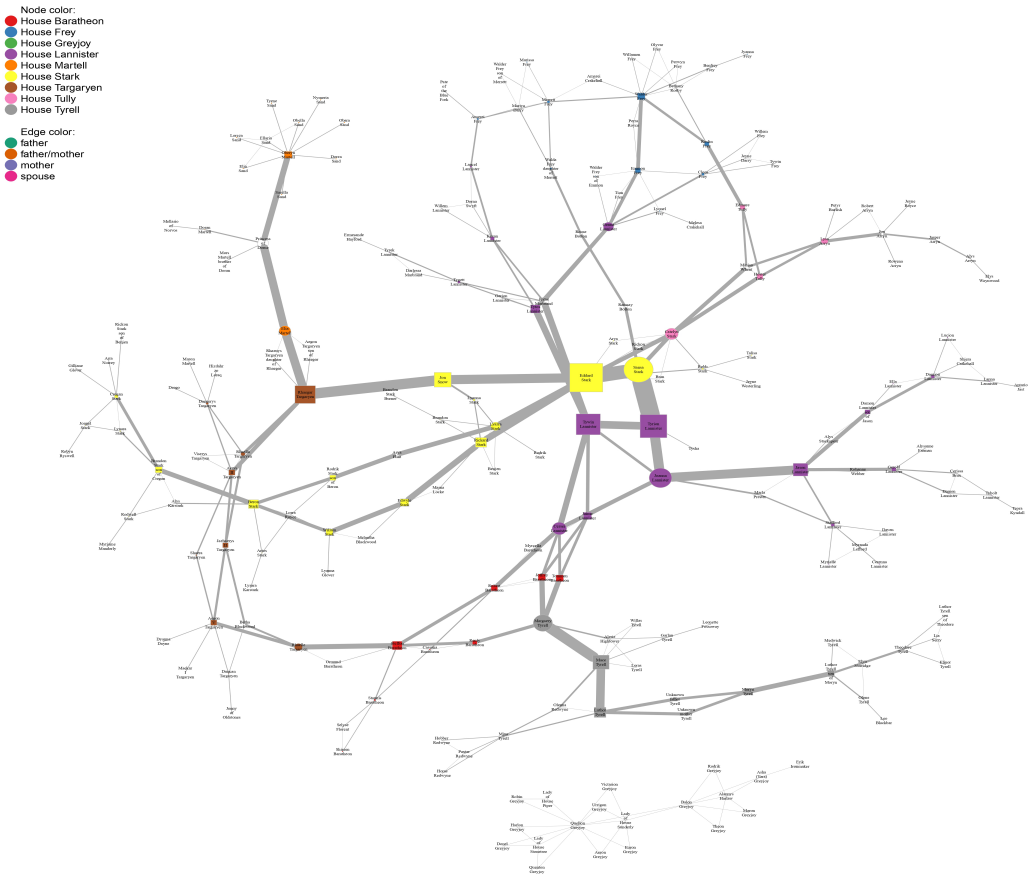


Figure 19: Fig 4.2: Betweenes graph for characters

we can conclude from Fig 4.2, that Ned Stark is the character with the highest Betweeness. (Too bad he died...) This actually does make sense, since his children - Sansa specifically - got married to oter families with high central positions.

in addition, the researchers who conducted this work took the database and put it in a matrix. whenever there was an edge between two nodes, there was a '1' in that field in the matrix. From that, they calculated eigenvalues, and eigenvectors, where the eigenvectors with the highest eigenvalue (in absolute value) means it's more connected.

When we consider eigevnector centrality - we can see that Tywin Lannsiter scores highest.

We can conclude all those factors, and more, to one graph.

```
par(bty = 'n', xaxt = 'n', yaxt = 'n')
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_centrality,1,1,2,2)
```

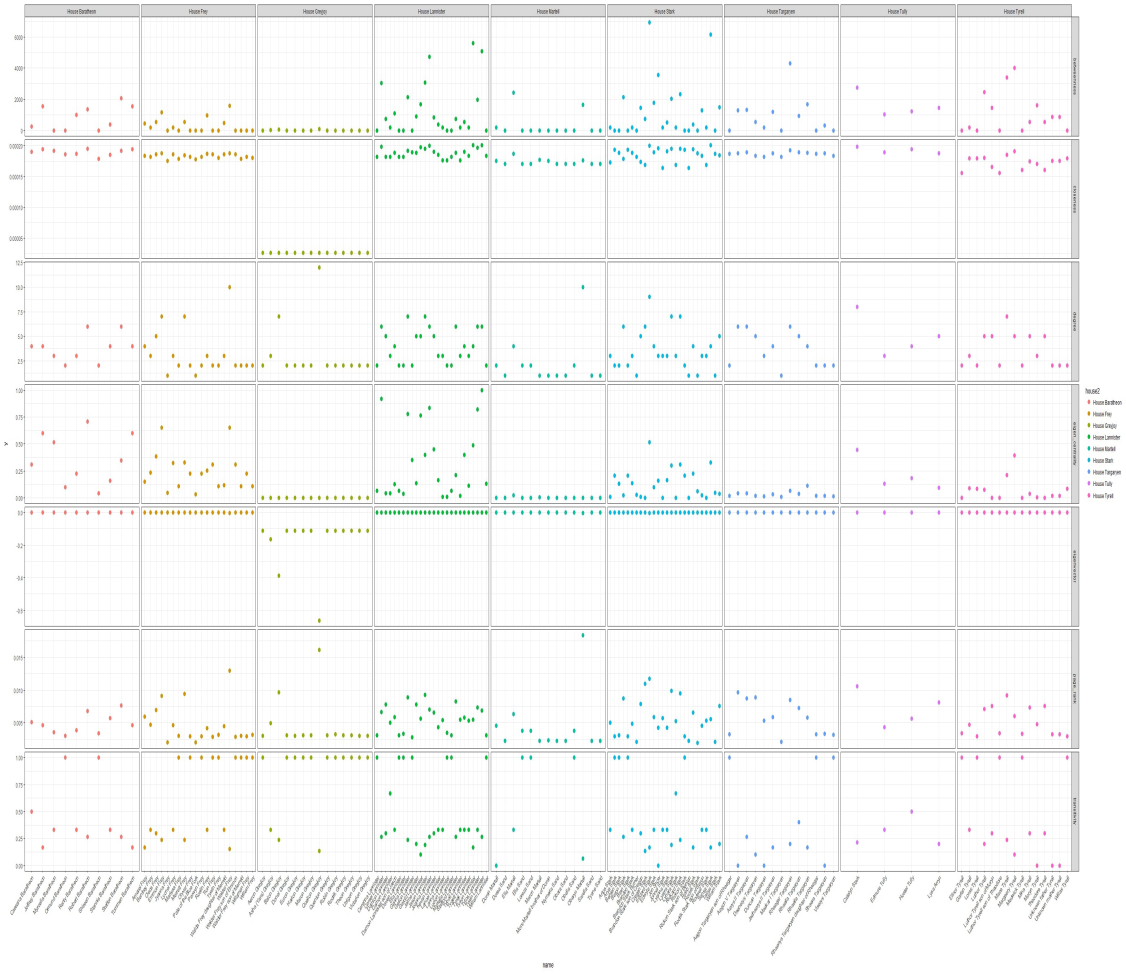


Figure 20: Fig 4.3: All Above described factors in one graph

As seen in Fig 4.3, when all taken together, we can say firmly that house Stark (and specifically late Ned, and Sansa) and House Lannister (especially Tyrion) are the most important family connections in Game of Thrones.

Building Our Model

When we come to build the model, we have two options:

A. Build a simple model that makes an average of the results we've seen so far.

B. Build a "learning" model, as we learned in the class on the subject of linear regression and in another course we learned a little different version called "Cross Validation"

Before we start, we must arrange the data. It is difficult and even impossible to give precise weight to each chapter we analyzed, so for the sake of building the model we give equal weight to all parts. We build a matrix divided by the chapters when we took the 9 main characters as a result of the data and in each row, we ranked the characters from 1 to 9 according to the analysis we performed earlier.

But, with our learning model, we'll try to fix the weights and we'll show how with the course's techniques we can analyze the data more accurately than the simple non-sophisticated model (as we will explain later).

```
Model<-matrix(0,5,9)
labels<-c("Tyrion Lannister", "Jon Snow", "Aria Stark", "Sansa Stark", "Daenerys Targaryen", "Night king
rownames(Model)<-c("Time Screen", "Battles", "Survey", "Family Ties", "Sum")
colnames(Model)<-labels
#As we have seen in the chapter of screen time, Jon has the longest screen time, followed by order - Ty
#Bran and finally the Night king
Model["Time Screen", ]<-c(8,9,4,6,7,1,2,5,3)

#Battles:
#according our ranksTotal:
#Daenerys, Jon, night king, Jaime, Arya, Sansa and Tyrion. both Bran and Cersei has no battles and get
Model["Battles", ]<-c(3,8,5,4,9,7,2,2,6)

#Survey: As you may recall, we analyzed a two-part survey, and finally, we combined the answers into on
#The order is (Note: Jon and Tyrion have the same rank, then both will get 9):
Model["Survey", ]<-c(9,9,7,6,5,4,3,2,1)

#Family Ties: let's use Shirin's resaults:
Model["Family Ties", ]<-c(8,7,6,9,3,1,2,5,4)

#sum the resaults
Model["Sum", ]<-apply(Model,2,sum)

kable(head(Model[,1:9]), format = "markdown")
```

	Tyrion Lannister	Jon Snow	Aria Stark	Sansa Stark	Daenerys Targaryen	Night king	Bran Stark	Cersei Lannister	Jaime Lannister
Time Screen	8	9	4	6	7	1	2	5	3
Battles	3	8	5	4	9	7	2	2	6
Survey	9	9	7	6	5	4	3	2	1
Family Ties	8	7	6	9	3	1	2	5	4
Sum	28	33	22	25	24	13	9	14	14

Once we have arranged the data we have analyzed, we can build the summary model.

First let's see the simple model, as seen in Fig 5.1:

```
#sort
ModelPlot<-Model[,order(Model[nrow(Model),])]
ModelPlot<-t(ModelPlot)
ModelPlot<-data.frame(ModelPlot[,5])
ModelPlot<-t(ModelPlot)
ModelPlot<-as.data.frame(ModelPlot)
```

```r
rownames(ModelPlot)<-"Total score"
ModelPlot<-rbind(ModelPlot, colnames(ModelPlot))
rownames(ModelPlot)<-c("Total score", "character")
ModelPlot<-t(ModelPlot)
ModelPlot<-as.data.frame(ModelPlot)
```

```r
#plot the resaults of basic Model
p <- ggplot(data = ModelPlot, aes(x = ModelPlot$character, y = ModelPlot$`Total score`)) +
    geom_bar(stat="identity", fill="3") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_y_discrete(limits = ModelPlot[1:9,1]) +
    scale_x_discrete(limits = ModelPlot[1:9,2]) +
    ggtitle("Results") +
    labs(x="character", y="Score", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

```
## Warning: Use of `ModelPlot$character` is discouraged. Use `character` instead.
```

```
## Warning: Use of `ModelPlot$`Total score`` is discouraged. Use `Total score`
## instead.
```

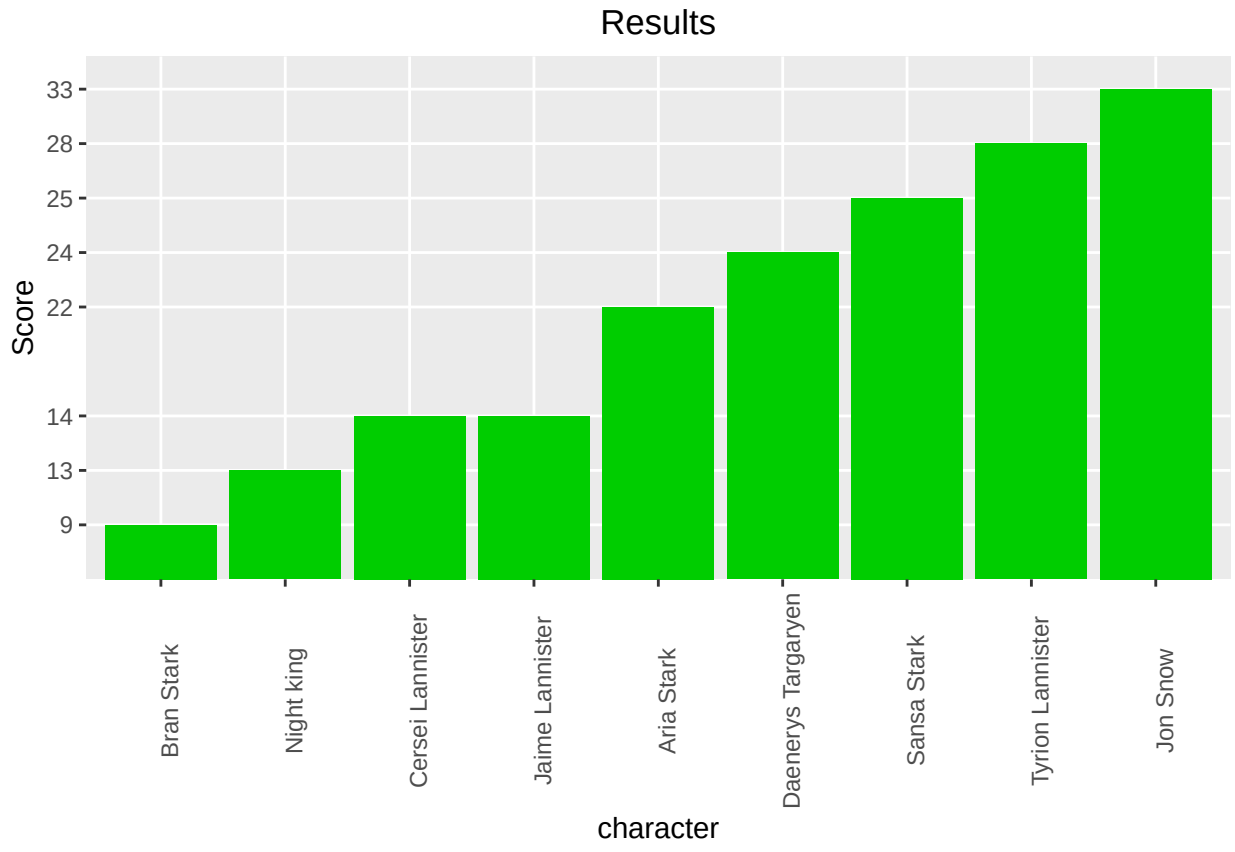


Figure 21: Fig 5.1: Answer to our research question, according to our simple model

Now let's build our "learning model", using "Cross Validation". A short explanation:

Since we have four lines of information (which is actually a bit of data), any mistake or measurement error can have a significant impact. To solve this problem, one should use "Cross Validation" algorithm. The algorithm says that if we have K data vectors (in our case K = 4), then every time we take K-1 data and build a simple model from them. We will compare new simple model to the data we did not use, and thus we will see the correlation between that data and the rest of the data. This is how you can discover information items that are not highly correlated with the other information items, and then you can say that they are not representative and therefore give them low weight.

In the professional language, the $K-1$ vectors are "training data set", and the last vector is the "test set".

Visual example for the first step:

```
par(bty = 'n', xaxt = 'n', yaxt = 'n', mar=c(1,2,1,2))
plot(1:2, type='n',ylab = '', xlab = '')
rasterImage(img_CrossValidation,1,1,2,2)
```

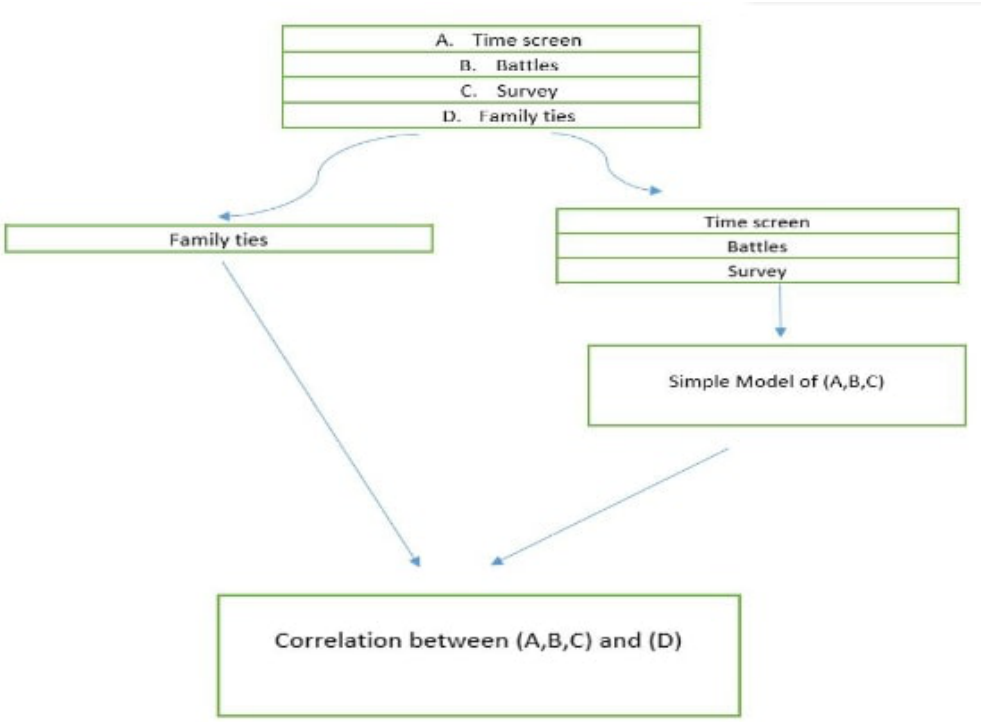


Figure 22: Fig 5.2: Cross validation method to improve our model

If we will see a character with high correlation between (A,B,C) and (D), we can assume that his D's value are correct.

We will write a general code which checks changes - and will run it in two different variations. first of all we check if there are any slots in the table, which their value is different within at least 5 points from the average of the character's other slots. Second, we will restrict ourself, and check if there is a slot which is different by 3 points from the average of its other parameters.

Once we find these exceptions, we can classify them as "significant changes" and take them out of the table, and by that we may be able to get some change in the final rating we recieve.

First check (5 points):

```
ModelCV<-Model[-5,]
topics<-c("Time screen", "Battles", "Survey", "Family ties")
for (k in 1:4){
  suspectWrong<-matrix(0,1,9)
  print("training dataset:")
  Model1<-ModelCV[-k,]
  Model1<-rbind(Model1, 1:9)
  rownames(Model1)<-c(rownames(ModelCV[-k,]), "Sum")
  Model1["Sum", ]<-apply(Model1[1:3,],2,sum)
  print(Model1)
  cat("\n")

  print("test data:")
  print(ModelCV[k,])
  cat("\n")

  for (i in 1:9){
    if(abs((Model1[4,i]/3) - ModelCV[k,i]) >= 5){
      suspectWrong[1,i]<-(colnames(Model1)[i])
    }
  }
  if (any(suspectWrong[1,] != 0)){
    cat("characters with bad correlation between", topics[k])
    cat(" and the rest:\n")
    for (i in 1:9){
    if(suspectWrong[1,i] != 0){
      print(suspectWrong[1,i])
    }
  }
  }
  else{
    cat(" Didn't find bad correlation!\n")
  }
  cat("\n")
}
```

```
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Battles                    3        8          5           4                  9
## Survey                     9        9          7           6                  5
## Family Ties                8        7          6           9                  3
## Sum                       20       24         18          19                 17
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Battles              7          2                2               6
## Survey               4          3                2               1
## Family Ties          1          2                5               4
## Sum                 12          7                9              11
##
## [1] "test data:"
```

```
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  8                  9                  4                  6
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  7                  1                  2                  5
##    Jaime Lannister
##                  3
##
##  Didn't find bad correlation!
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Survey                     9        9          7           6                  5
## Family Ties                8        7          6           9                  3
## Sum                       25       25         17          21                 15
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Survey               4          3                2               1
## Family Ties          1          2                5               4
## Sum                  6          7               12               8
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  3                  8                  5                  4
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  9                  7                  2                  2
##    Jaime Lannister
##                  6
##
## characters with bad correlation between Battles and the rest:
## [1] "Tyrion Lannister"
## [1] "Night king"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Battles                    3        8          5           4                  9
## Family Ties                8        7          6           9                  3
## Sum                       19       24         15          19                 19
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Battles              7          2                2               6
## Family Ties          1          2                5               4
## Sum                  9          6               12              13
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  9                  9                  7                  6
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  5                  4                  3                  2
##    Jaime Lannister
##                  1
##
##  Didn't find bad correlation!
```

```
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Battles                    3        8          5           4                  9
## Survey                     9        9          7           6                  5
## Sum                       20       26         16          16                 21
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Battles              7          2                2               6
## Survey               4          3                2               1
## Sum                 12          7                9              10
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  8                  7                  6                  9
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  3                  1                  2                  5
##    Jaime Lannister
##                  4
##
##  Didn't find bad correlation!
```

We can see that ranks of Tyrion and the night king in the category of battles is significantly different from their ranking in the other categories. Let's change their ranks in this category to the average of their ranks in the other categories, and see if there is a change in overall ranking:

```
modelAfterFirstImprove<-as.data.frame(ModelCV)
modelAfterFirstImprove["Battles","Tyrion Lannister"]<-round(sum(modelAfterFirstImprove[-2,"Tyrion Lannis
modelAfterFirstImprove["Battles","Night king"]<-round(sum(modelAfterFirstImprove[-2,"Night king"])/3, di
modelAfterFirstImprove<-rbind(modelAfterFirstImprove, 1:9)
rownames(modelAfterFirstImprove)<-c(rownames(modelAfterFirstImprove[1:4,]), "Sum")
modelAfterFirstImprove["Sum", ]<-apply(modelAfterFirstImprove[1:4,],2,sum)
modelAfterFirstImprove<-modelAfterFirstImprove[,order(modelAfterFirstImprove[nrow(modelAfterFirstImprove
modelAfterFirstImprove<-t(modelAfterFirstImprove)
modelAfterFirstImprove<-data.frame(modelAfterFirstImprove[,5])
modelAfterFirstImprove<-t(modelAfterFirstImprove)
modelAfterFirstImprove<-as.data.frame(modelAfterFirstImprove)
rownames(modelAfterFirstImprove)<-"Total score"
modelAfterFirstImprove<-rbind(modelAfterFirstImprove, colnames(modelAfterFirstImprove))
rownames(modelAfterFirstImprove)<-c("Total score", "character")
modelAfterFirstImprove<-t(modelAfterFirstImprove)
modelAfterFirstImprove<-as.data.frame(modelAfterFirstImprove)
```

```
p <- ggplot(data = modelAfterFirstImprove, aes(x = modelAfterFirstImprove$character, y = modelAfterFirst
    geom_bar(stat="identity", fill="#00C0AF") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_y_discrete(limits = modelAfterFirstImprove[1:9,1]) +
    scale_x_discrete(limits = modelAfterFirstImprove[1:9,2]) +
    ggtitle("Results using Cross Validaion") +
    labs(x="character", y="Score", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

```
## Warning: Use of `modelAfterFirstImprove$character` is discouraged. Use
## `character` instead.
```

```
## Warning: Use of `modelAfterFirstImprove$`Total score`` is discouraged. Use
## `Total score` instead.
```

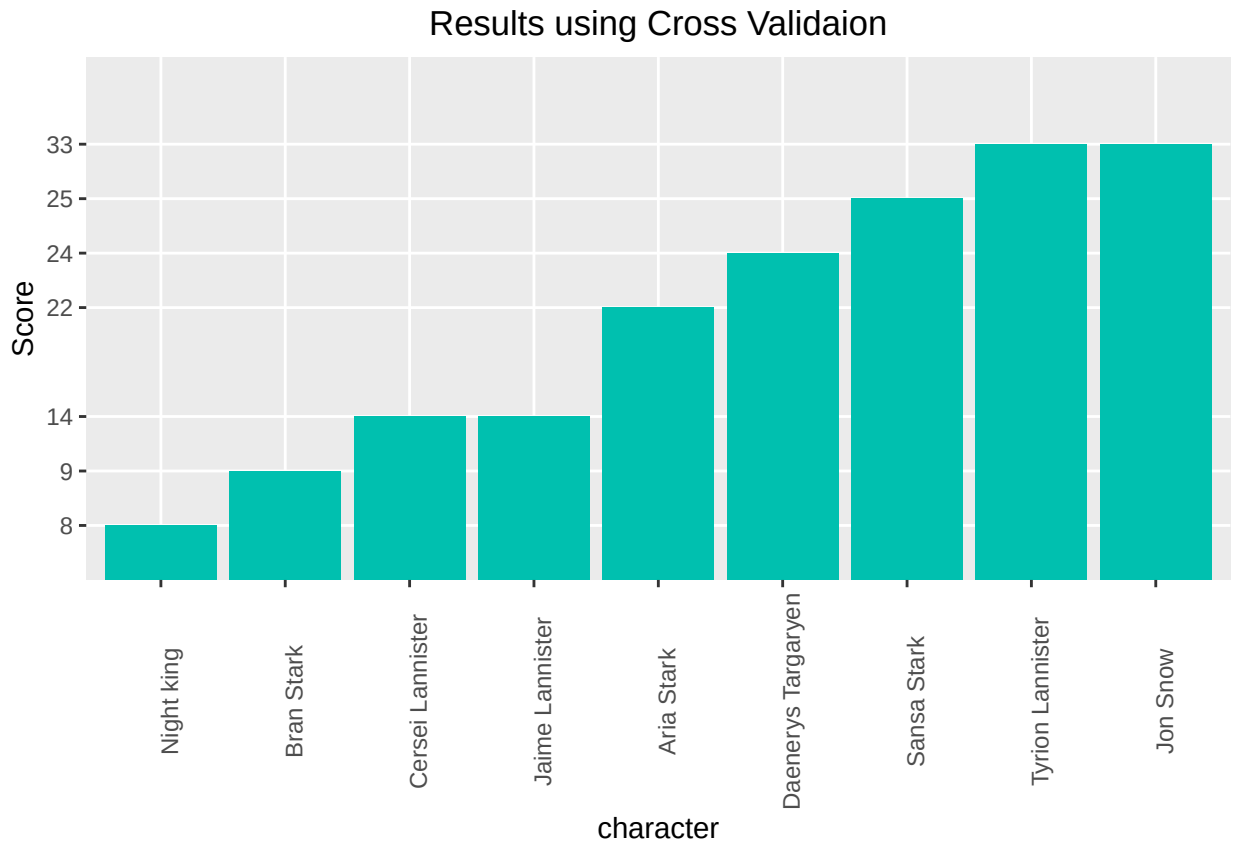


Figure 23: Fig 5.3: answer to our research question, according to our progressed model, taking cross validation into consideration

It can be seen from Fig 5.3 that Tyrion's rating was improved and got equal to Jon in the top of the list, whilst the Night King went down to bottom. We don't see very extreme changes, but it is completely not the naive model we had (in the conclusion part we will compare it to the real results - the knowledge we happen to have from the eighth season, and observe that this model was actually really better)

Now we restrict our demand for the corelation, and expect to be getting more fluctuations. We will run our code for the change's test, and test how bad are those fluctuations.

Second check (3 points):

```
topics<-c("Screen Time", "Battles", "Survey", "Family ties")
for (k in 1:4){
  suspectWrong<-matrix(0,1,9)
  print("training dataset:")
  Model1<-ModelCV[-k,]
  Model1<-rbind(Model1, 1:9)
  rownames(Model1)<-c(rownames(ModelCV[-k,]), "Sum")
```

```
  Model1["Sum", ]<-apply(Model1[1:3,],2,sum)
  print(Model1)
  cat("\n")

  print("test data:")
  print(ModelCV[k,])
  cat("\n")

  for (i in 1:9){
    if(abs((Model1[4,i]/3) - ModelCV[k,i]) >= 3){
      suspectWrong[1,i]<-(colnames(Model1)[i])
    }
  }
  if (any(suspectWrong[1,] != 0)){
    cat("characters with bad correlation between", topics[k])
    cat(" and the rest:\n")
    for (i in 1:9){
    if(suspectWrong[1,i] != 0){
      print(suspectWrong[1,i])
    }
  }
  }
  else{
    cat(" Didn't find bad correlation!\n")
  }
  cat("\n")
}
```

```
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Battles                    3        8          5           4                  9
## Survey                     9        9          7           6                  5
## Family Ties                8        7          6           9                  3
## Sum                       20       24         18          19                 17
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Battles              7          2                2               6
## Survey               4          3                2               1
## Family Ties          1          2                5               4
## Sum                 12          7                9              11
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  8                  9                  4                  6
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  7                  1                  2                  5
##    Jaime Lannister
##                  3
##
## characters with bad correlation between Screen Time and the rest:
## [1] "Night king"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
```

```
## Time Screen                8        9          4           6                  7
## Survey                     9        9          7           6                  5
## Family Ties                8        7          6           9                  3
## Sum                       25       25         17          21                 15
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Survey               4          3                2               1
## Family Ties          1          2                5               4
## Sum                  6          7               12               8
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  3                  8                  5                  4
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  9                  7                  2                  2
##    Jaime Lannister
##                  6
##
## characters with bad correlation between Battles and the rest:
## [1] "Tyrion Lannister"
## [1] "Sansa Stark"
## [1] "Daenerys Targaryen"
## [1] "Night king"
## [1] "Jaime Lannister"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Battles                    3        8          5           4                  9
## Family Ties                8        7          6           9                  3
## Sum                       19       24         15          19                 19
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Battles              7          2                2               6
## Family Ties          1          2                5               4
## Sum                  9          6               12              13
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  9                  9                  7                  6
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  5                  4                  3                  2
##    Jaime Lannister
##                  1
##
## characters with bad correlation between Survey and the rest:
## [1] "Jaime Lannister"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Battles                    3        8          5           4                  9
## Survey                     9        9          7           6                  5
## Sum                       20       26         16          16                 21
```

```
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5               3
## Battles              7          2                2               6
## Survey               4          3                2               1
## Sum                 12          7                9              10
##
## [1] "test data:"
##   Tyrion Lannister           Jon Snow         Aria Stark        Sansa Stark
##                  8                  7                  6                  9
## Daenerys Targaryen         Night king         Bran Stark   Cersei Lannister
##                  3                  1                  2                  5
##    Jaime Lannister
##                  4
##
## characters with bad correlation between Family ties and the rest:
## [1] "Sansa Stark"
## [1] "Daenerys Targaryen"
## [1] "Night king"
```

We test the suspicious value we've got:

Daenarys has got twice a suspicios value. Her set of values is 3,5,7,9 - and therefore the value 3,9 are accordingly suspicious ones, and ultimately will compensate each other. This is why we don't need to have any fix to her rating. The Night King got two suspicious values beside the suspicious value he has got from the last run. Those values are identical - 1 - so we don't need to fix them as well.

The other suspicious values are for Jaime and Sansa, two values for each. Here as well, the value are from either sides of their median (one high and one low), but they are not exactly in the middle so we need to fix them.

Let's enter the results and observe the final rating:

```
modelAfterSecondImprove<-as.data.frame(ModelCV)
modelAfterSecondImprove["Battles","Tyrion Lannister"]<-round(sum(modelAfterSecondImprove[-2,"Tyrion Lan
modelAfterSecondImprove["Battles","Night king"]<-round(sum(modelAfterSecondImprove[-2,"Night king"])/3,
modelAfterSecondImprove["Family Ties","Sansa Stark"]<-round(sum(modelAfterSecondImprove[-4,"Sansa Stark
modelAfterSecondImprove["Battles","Sansa Stark"]<-round(sum(ModelCV[-2,"Sansa Stark"])/3, digits = 0)
modelAfterSecondImprove["Family Ties","Jaime Lannister"]<-round(sum(modelAfterSecondImprove[-4,"Jaime La
modelAfterSecondImprove["Battles","Jaime Lannister"]<-round(sum(modelAfterSecondImprove[-2,"Jaime Lanni
modelAfterSecondImprove<-rbind(modelAfterSecondImprove, 1:9)
rownames(modelAfterSecondImprove)<-c(rownames(modelAfterSecondImprove[1:4,]), "Sum")
modelAfterSecondImprove["Sum", ]<-apply(modelAfterSecondImprove[1:4,],2,sum)
modelAfterSecondImprove<-modelAfterSecondImprove[,order(modelAfterSecondImprove[nrow(modelAfterSecondImp
modelAfterSecondImprove<-t(modelAfterSecondImprove)
modelAfterSecondImprove<-data.frame(modelAfterSecondImprove[,5])
modelAfterSecondImprove<-t(modelAfterSecondImprove)
modelAfterSecondImprove<-as.data.frame(modelAfterSecondImprove)
rownames(modelAfterSecondImprove)<-"Total score"
modelAfterSecondImprove<-rbind(modelAfterSecondImprove, colnames(modelAfterSecondImprove))
rownames(modelAfterSecondImprove)<-c("Total score", "character")
modelAfterSecondImprove<-t(modelAfterSecondImprove)
modelAfterSecondImprove<-as.data.frame(modelAfterSecondImprove)
```

```
p <- ggplot(data = modelAfterSecondImprove, aes(x = modelAfterSecondImprove$character, y = modelAfterSe
    geom_bar(stat="identity", fill="#619CFF") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_y_discrete(limits = modelAfterSecondImprove[1:9,1]) +
    scale_x_discrete(limits = modelAfterSecondImprove[1:9,2]) +
    ggtitle("Results using Cross Validaion (more restricted)") +
    labs(x="character", y="Score", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

```
## Warning: Use of `modelAfterSecondImprove$character` is discouraged. Use
## `character` instead.
```

```
## Warning: Use of `modelAfterSecondImprove$`Total score`` is discouraged. Use
## `Total score` instead.
```

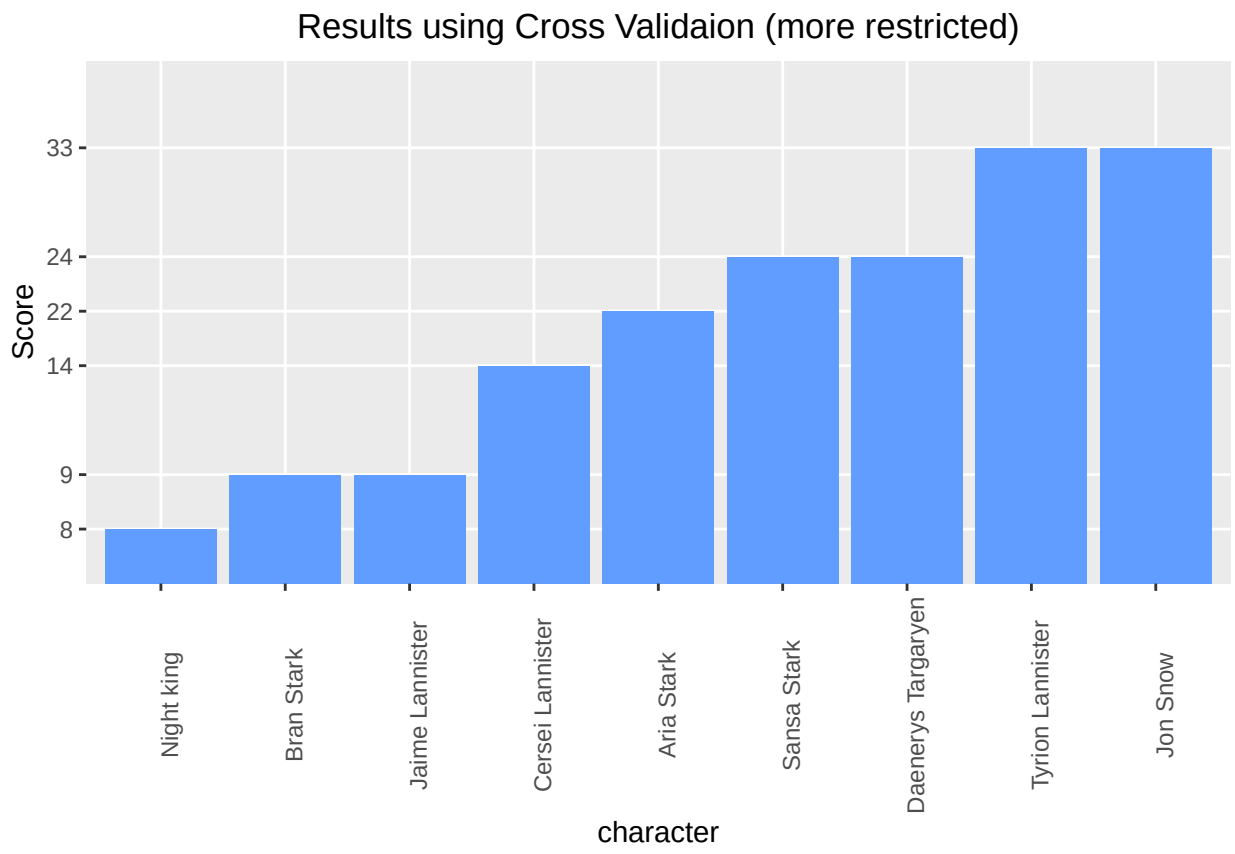


Figure 24: Fig 5.4: answer to our research question, according to our progressed model, taking cross validation into consideration (more restricted)

Let's test the changes in view of our previous results. We see in Fig 5.4 that Jaime Lannister's rating went down, and now it's equal to Bran - before last. In addition, Sansa's rank went down a little bit as well, and now she is sharing the 3rd-4th places together with Daenarys, while beforehand she was standing by herself in the 3rd place.

When taking a further look on the last run we ran, it can be clearly seen that more than half of the values in Battles category are not well correlated with the values in the other categories. It may be a result of miss-analysis on our behalf, but at the same time it could be that this parameter (i.e battles) had not been given the right weight it should have been given, or even should have been ignored in the first place. This situation raises a dilemma whether we can improve our model, and if so - how? Ultimately, we chose to ignore completely this parameter of Battles, and to execute our algorithm (the restricted one) with the table we have got left. Since we already described and explained our algorithm above, we could just simply run our script without any further ado.

```
modelAfterThirdImprove<-as.data.frame(ModelCV)
modelAfterThirdImprove<-modelAfterThirdImprove[-2,]
topics<-c("Screen Time", "Survey", "Family ties")
for (k in 1:3){
  suspectWrong<-matrix(0,1,9)
  print("training dataset:")
  Model1<-modelAfterThirdImprove[-k,]
  Model1<-rbind(Model1, 1:9)
  rownames(Model1)<-c(rownames(modelAfterThirdImprove[-k,]), "Sum")
  Model1["Sum", ]<-apply(Model1[1:2,],2,sum)
  print(Model1)
  cat("\n")

  print("test data:")
  print(modelAfterThirdImprove[k,])
  cat("\n")

  for (i in 1:9){
    if(abs((Model1[3,i]/2) - modelAfterThirdImprove[k,i]) >= 3){
      suspectWrong[1,i]<-(colnames(Model1)[i])
    }
  }
  if (any(suspectWrong[1,] != 0)){
    cat("characters with bad correlation between", topics[k])
    cat(" and the rest:\n")
    for (i in 1:9){
    if(suspectWrong[1,i] != 0){
      print(suspectWrong[1,i])
    }
  }
  }
  else{
    cat(" Didn't find bad correlation!\n")
  }
  cat("\n")
}
```

```
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Survey                    9        9          7           6                  5
## Family Ties               8        7          6           9                  3
## Sum                      17       16         13          15                  8
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Survey               4          3                2               1
## Family Ties          1          2                5               4
```

```
## Sum                  5          5                7              5
##
## [1] "test data:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5              3
##
## characters with bad correlation between Screen Time and the rest:
## [1] "Daenerys Targaryen"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Family Ties                8        7          6           9                  3
## Sum                       16       16         10          15                 10
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5              3
## Family Ties          1          2                5              4
## Sum                  2          4               10              7
##
## [1] "test data:"
##        Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Survey                9        9          7           6                  5
##        Night king Bran Stark Cersei Lannister Jaime Lannister
## Survey          4          3                2              1
##
## characters with bad correlation between Survey and the rest:
## [1] "Night king"
## [1] "Cersei Lannister"
##
## [1] "training dataset:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Time Screen                8        9          4           6                  7
## Survey                     9        9          7           6                  5
## Sum                       17       18         11          12                 12
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Time Screen          1          2                5              3
## Survey               4          3                2              1
## Sum                  5          5                7              4
##
## [1] "test data:"
##             Tyrion Lannister Jon Snow Aria Stark Sansa Stark Daenerys Targaryen
## Family Ties                8        7          6           9                  3
##             Night king Bran Stark Cersei Lannister Jaime Lannister
## Family Ties          1          2                5              4
##
## characters with bad correlation between Family ties and the rest:
## [1] "Sansa Stark"
## [1] "Daenerys Targaryen"
```

As before, we will change the values according to the correlations we have got (we will ignore the suspicious values we have got for Daenarys, since they compensate each other, as explained above):

```
modelAfterThirdImprove<-as.data.frame(ModelCV)
modelAfterThirdImprove<-modelAfterThirdImprove[-2,]
modelAfterThirdImprove["Survey","Cersei Lannister"]<-5
modelAfterThirdImprove["Survey","Night king"]<-1
modelAfterThirdImprove["Family Ties","Sansa Stark"]<-6
modelAfterThirdImprove<-rbind(modelAfterThirdImprove, 1:9)
rownames(modelAfterThirdImprove)<-c(rownames(modelAfterThirdImprove[1:3,]), "Sum")
modelAfterThirdImprove["Sum", ]<-apply(modelAfterThirdImprove[1:3,],2,sum)
modelAfterThirdImprove<-modelAfterThirdImprove[,order(modelAfterThirdImprove[nrow(modelAfterThirdImprove
modelAfterThirdImprove<-t(modelAfterThirdImprove)
modelAfterThirdImprove<-data.frame(modelAfterThirdImprove[,4])
modelAfterThirdImprove<-t(modelAfterThirdImprove)
modelAfterThirdImprove<-as.data.frame(modelAfterThirdImprove)
rownames(modelAfterThirdImprove)<-"Total score"
modelAfterThirdImprove<-rbind(modelAfterThirdImprove, colnames(modelAfterThirdImprove))
rownames(modelAfterThirdImprove)<-c("Total score", "character")
modelAfterThirdImprove<-t(modelAfterThirdImprove)
modelAfterThirdImprove<-as.data.frame(modelAfterThirdImprove)

p <- ggplot(data = modelAfterThirdImprove, aes(x = modelAfterThirdImprove$character, y = modelAfterThir
    geom_bar(stat="identity", fill="royalblue") +
    theme(axis.text.x = element_text(angle = 90)) +
    scale_y_discrete(limits = modelAfterThirdImprove[1:9,1]) +
    scale_x_discrete(limits = modelAfterThirdImprove[1:9,2]) +
    ggtitle("Results using Cross validaion after delete Battles") +
    labs(x="character", y="Score", fill = "") +
    theme(plot.title = element_text(hjust = 0.5))
p
```

```
## Warning: Use of `modelAfterThirdImprove$character` is discouraged. Use
## `character` instead.
```

```
## Warning: Use of `modelAfterThirdImprove$`Total score`` is discouraged. Use
## `Total score` instead.
```

Here we finish all the stages to create our model, which is based over the entire analysis we have done with so much hard work. Let's show all of models in one plot:

```
finalTotal<-matrix(0,4,9)
ModelTypes<-c("basic model", "first Cross-Validation model", "second Cross-Validation model", "complex
rownames(finalTotal)<-ModelTypes
colnames(finalTotal)<-ModelPlot[,2]
finalTotal["basic model",]<-as.numeric(as.character(ModelPlot[,1]))
finalTotal["first Cross-Validation model",]<-as.numeric(as.character(modelAfterFirstImprove[colnames(fi
finalTotal["second Cross-Validation model",]<-as.numeric(as.character(modelAfterSecondImprove[colnames(
finalTotal["complex Cross-Validation model",]<-as.numeric(as.character(modelAfterThirdImprove[colnames(

#plot
par(mar=c(8,4,4,4), las=2)
barplot(finalTotal,
main = "Combine all models",
col = rainbow(4, start = 0.5, end = 0.75),
```

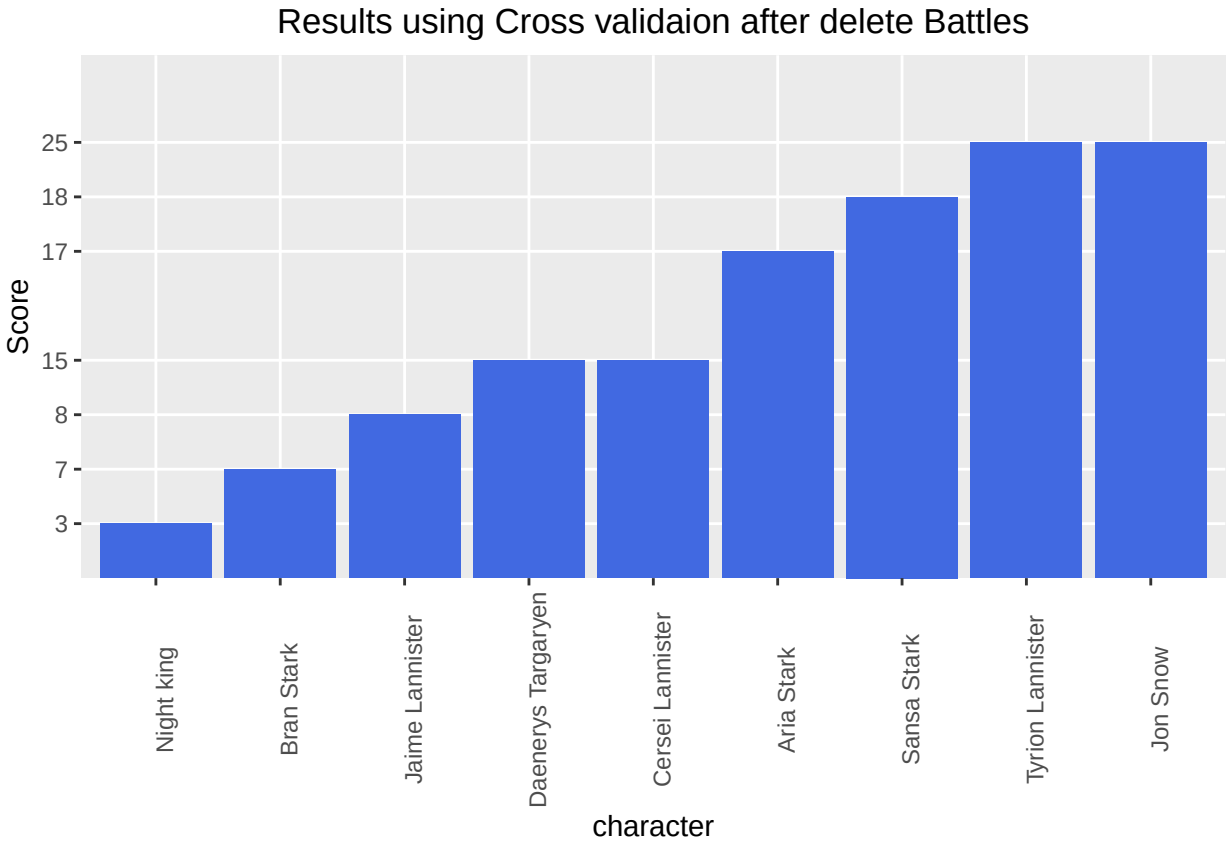


Figure 25: Fig 5.5: answer to our research question, according to our most advanced model, taking cross validation into consideration (restricted)

```
beside = TRUE
)
legend("topleft",
ModelTypes,
fill = rainbow(4, start = 0.5, end = 0.75),
cex = 0.75,
box.lwd = 0,box.col = "white",bg = "white"
)
title(ylab="Score", line = 2, cex.lab=1.2)
title(xlab="Character", line = 7, cex.lab=1.2)
grid()
```

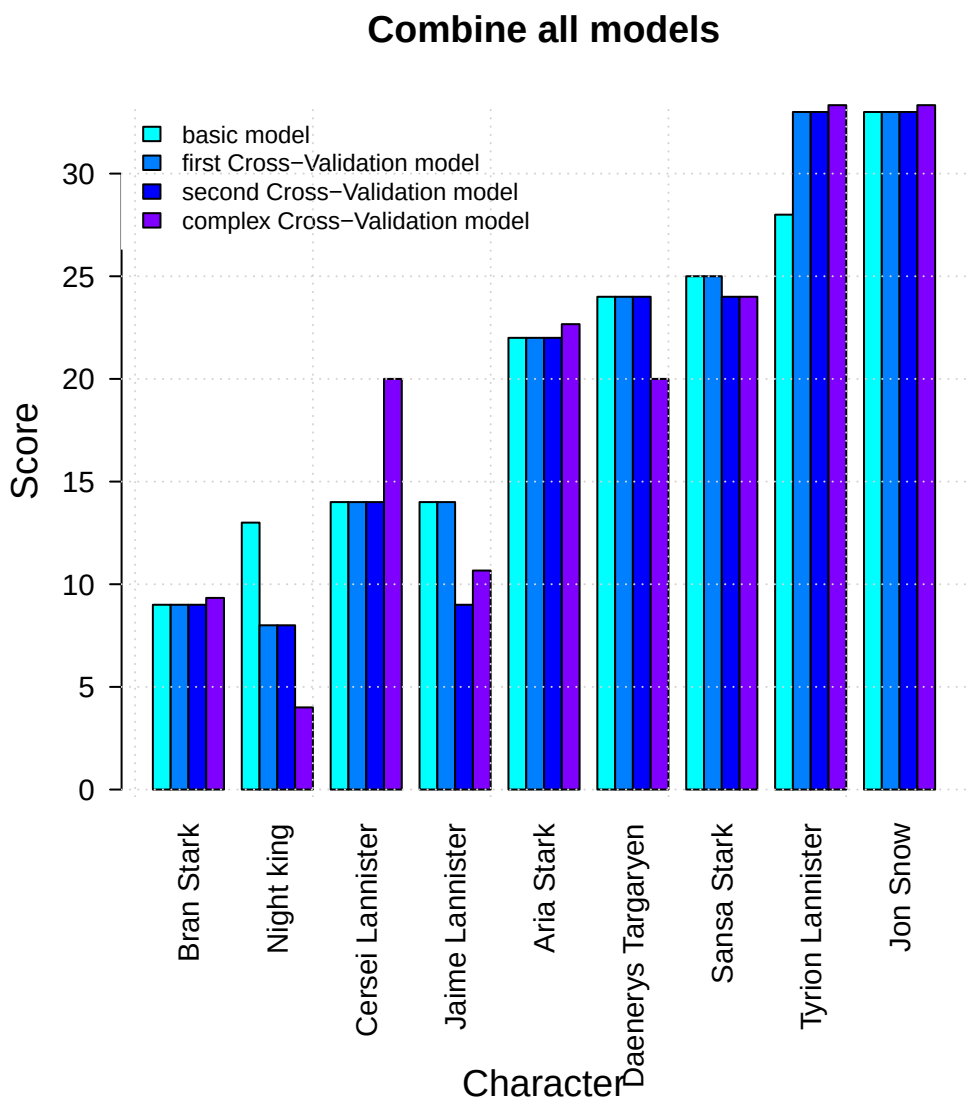


Figure 26: Fig 5.6: All models together

In the conclusion part we will add some reference to the real-results as seen in the last season of the show, and we will check if our analysis is somehow close to what actually happened, or maybe the show's creators

have surprised us all and created something that any foresight could never possibly see.

Summary & Conclusions

Our research question was dealing with the winner of season 8 of Game of Thrones, based on the last seven seasons. In order to do so, we've analysed a few parameters based on a very large-scaled data. During the analysis we used several algorithms - ones we learned in the course, and some others. We ultimately created a table which adds up all those parameters, from which we tried to create a model which will hopefully foresee our research question's answer. In conclusion. we can say the according to our project it seems that Jon, Tyrion and Sansa are most likely to survive the last season, and perhaps even "win" it. On the other hand, Bran and the Night King are the characters with the least chances to do so.

Ultimately, since we can compare our results to the "real" results of the eighth season - the throne itself has been destroyed, and the character who was crowned to king was actually Bran, whilst Jon's fate was very different than expected (Even though he survived the season's finale).

Despite this, it can be understood that for most of the characters there is a good correlation between their ranking and the way the show ended for them. The top 3 ranked characters (I.e Jon, Tyrion and Sansa) survived, and even contributed in a way to the crowning of the King; whilst the bottom 3 characters (besides Bran) have died, and we can even see that the lower the character was ranked the faster its farewell have arrived. Daenarys who is ranked 4th died, though not exactly as expected, rather by her beloved Jon, which could be interpreted as intentional plot twist and therefore not statistically predictable, by definition. Arya, who is ranked 5th, stayed alive and even had the privilege to kill the nemesis Night King. Overall, we can fairly say that the fate of the characters that were ranked high by us was merely good, and on other hand the characters that were placed in the bottom of our table have died. The only exception is Bran, who was ranked at the bottom and was eventually crowned as king. If we take a better look on the differences between the models, we can see a trend regarding 3 characters – the status of the Night King and Jaime Lannister decreases, whilst Tyrion's status went up, and the other characters stayed fairly at their spot. As we said before, we see it fits the plot, and that ultimately our models had indeed an insight of the things that actually occurred on the show.

In adition, there is something we can honestly say we analysed well, which is the question of the dominance of the Houses. in all our parameters, House Stark stood out and seemed to be the most significant and dominant house. And as it happened to be, the king is actually a Stark, and the other Starks remained alive. Lannister House, on the other hand, was less significant, and indeed two main character who are Lannisters have died.

In conclusion, the gap we see between our model and the real results of the show can be explained either by an insufficient analysis work on our side and maybe not taking some other important factors into consideration; or, alternatively, that the minds of the show's producers in Hollywood are too complex to mathematically foresee and maybe (intentionally) unwilling to follow the laws of statistics.