Diablo 3 - Season 6

INFO 550 Final Project

Sara Khan

Table of Contents

Research Questions

1. Are there any player statistic that correlates to the player rank?
2. What item sets do most players use at the top of the leaderboard?

Data cleaning

Part A. Getting data.

Let's get some data. I'm querying the first 1000 people on the leaderboard for season 6.

First, I built an access string to query data from the BattleNet API. I redacted my actual APP/KEY.

I saved the results into an "RData" dataset called "top1000"

```
## build an access string
SARAKEY<-"SARA'S-UNIQUE-KEY"
url<-paste0("https://us.api.battle.net/data/d3/season/6/leaderboard/achievement-points?access_token=",SARAKEY)

## Saving the actual API
top1000<-fromJSON(url)
```

Part B.

Now I'll create a function that cleans up the data from the JSON format into a usable data frame.

This is a function that pulls out the top 1000 user names and their hero ID. This will allow me to query data to create unique URLs from the Battlenet API for later. I'm also saving the Hero Class for later.

```
results <- plyr::ldply(1:1000, function(i) {

  tempHero<-(top1000$row$player[[i]])$data[[1]] %>%
    filter(id %in% c("HeroBattleTag","HeroClass","HeroId"))  %>%
    select(string,number)

 (c(i,tempHero[1,1],tempHero[3,2],tempHero[2,1]))
})

colnames(results)<-c("Rank","HeroBattleTag","HeroId","HeroClass")

head(results)
```

Rank	HeroBattleTag	HeroId	HeroClass
1	Knightmare#1642	50291904	demon hunter
2	Cursewords#1359	75503053	demon hunter

3	wby#1325	75508223	demon hunter
4	Blacksheep#1512	75438396	demon hunter
5	Beldox#1259	75506076	wizard
6	Tarzimal#1145	75510116	demon hunter

I used “ldply” to retrieve the results into a data frame. I found the “list” to be overkill for what I was trying to do, because I only wanted a simple data frame with the unique “Username” aka HeroBattleTag, with the username’s “Hero” aka HeroId for the first 1000 in the leaderboard.

Part C. Weirdness.

Now I’m going to create unique URLs from the “results” data frame to query data for the API.

While exploring the “results” data frame, I noticed some weird usernames…

	Rank	HeroBattleTag	HeroId	HeroClass
30	30	Ben#1328	65342964	demon hunter
31	31	#3656	65992925	wizard
32	32	Davidoff#3704	75520759	demon hunter
33	33	Kaizen#1773	75333396	witch doctor
34	34	putman82#1425	75508028	wizard
35	35	unhealthydog#1674	75523343	demon hunter
36	36	#3291	75521053	demon hunter
37	37	Zath#1848	9075104	demon hunter
38	38	UnDauNt3d520#1414	75585119	demon hunter
39	39	G473R#1729	75512535	demon hunter
40	40	Virsta#1257	75508280	witch doctor

Notice the odd unicode characters in Rank 31 and 36?

	Rank	HeroBattleTag	HeroId	HeroClass
31	31	#3656	65992925	wizard

	Rank	HeroBattleTag	HeroId	HeroClass

36	36	#3291	75521053	demon hunter

I wondered if it would impact my ability to query data from the API. I tested it into a URL like so and got an error.

```
fromJSON(paste0("https://us.api.battle.net/d3/profile/<U+904A><U+4FE0>%233113/hero/69485497?locale=en_US&apikey=",key))
```

I kept getting errors. I decided to get into the matter further, and found that the unicode listing actually appeared within the dataframe.

Example: when I queried the data out of the dataset by specifying the actual username, I would get the correct output, like so:

“斷罪之誓%3656” “遊俠%3291”

```
fromJSON(paste0("https://us.api.battle.net/d3/profile/<U+904A><U+4FE0>%233113/hero/69485497?locale=en_US&apikey=",key))
```

Now that I figured out the unicode issue, I’ll now create unique URLs to get data back from the API.

```
heroItemURL <- lapply(1:1000, function(i) {

  paste0("https://us.api.battle.net/d3/profile/",
         results$HeroBattleTag[i],
         "/hero/",
         results$HeroId[i],
         "?locale=en_US&apikey=",
         key)
})
```

To be safe, I created a function to only make URLS first. Then I’ll wrap them into a “fromJSON” function.

This would give me urls as so:

“https://us.api.battle.net/d3/profile/流星追月#3113/hero/69485497?locale=en_US&apikey=SARAKEY (https://us.api.battle.net/d3/profile/流星追月#3113/hero/69485497?locale=en_US&apikey=SARAKEY)”

However, I kept getting error messages from the API. This led me to check the URL correctly…

```
actualURL<-("https://us.api.battle.net/d3/profile/流星追月%233113/hero/69485497?lo
cale=en_US&apikey=SARAKEY")


testURL<-paste0("https://us.api.battle.net/d3/profile/",
                results$HeroBattleTag[965],
                "/hero/",
                results$HeroId[965],
                "?locale=en_US&apikey=",
                "key")

identical(testURL,actualURL)
```

```
## [1] FALSE
```

Why was I getting a FALSE url for the identical function? I looked closer, and found that I needed to sub out the “#” for a “%” and add a “23” before the HeroId.

```
results$HeroBattleTag<-gsub("#","%23", results$HeroBattleTag)
```

Now that the URL should be correct, let’s test it again.

```
actualURL<-("https://us.api.battle.net/d3/profile/流星追月%233113/hero/69485497?lo
cale=en_US&apikey=SARAKEY")


testURL<-paste0("https://us.api.battle.net/d3/profile/",
                results$HeroBattleTag[965],
                "/hero/",
                results$HeroId[965],
                "?locale=en_US&apikey=",
                "key")

identical(testURL,actualURL)
```

```
## [1] FALSE
```

A true statement! Great. That means we can finally grab all the URLs from the API.

```
heroItems<-lapply(1:1000, function(i) {
  fromJSON(heroItemURL[[i]])
})

head(heroItems[[965]])
```

```
## $id
## [1] 69485497
##
## $name
## [1] "é£ç©è¥¿æ¯"
##
## $class
## [1] "witch-doctor"
##
## $gender
## [1] 1
##
## $level
## [1] 70
##
## $kills
## elites
##  35448
```

Looks like it worked. I now have all 1000 players! I went ahead and saved the output to save my API calls.

```
save(heroItems,file="heroItems.RData")
```

Step 2

Now let's see what kind of interesting data is in the "heroItems" dataset. I know it is a list of 1000 elements. Each element corresponds to the person in order of rank. For example, heroItem[[1]] would bring up the data for the first person on the leaderboard.

```
summary(heroItems[[1]])
```

	Length	Class	Mode

id	1	-none-	numeric
name	1	-none-	character
class	1	-none-	character
gender	1	-none-	numeric
level	1	-none-	numeric
kills	1	-none-	numeric
paragonLevel	1	-none-	numeric
hardcore	1	-none-	logical
seasonal	1	-none-	logical
seasonCreated	1	-none-	numeric
skills	2	-none-	list
items	13	-none-	list
followers	2	-none-	list
legendaryPowers	3	-none-	list
stats	31	-none-	numeric
progression	5	-none-	list
dead	1	-none-	logical
last-updated	1	-none-	numeric

What I'm interested in is the "statistics" of the player. For example, this will answer the question based on the differences in damage by the rank of the player.

Step 3

Let's start to clean up the data and put it in a usable format to look at unique items that people use.

I'd like a data frame that has usable information in a format like so:

Hypothetical list of the 6 classes: Demon Hunter - Name of combo-set - Number of people from top 1000 with set - Number of people from top 100 with set Wizard - Name of combo-set - Number of people from top 1000 with set - Number of people from top 100 with set …. etc

From that, I'll be able to create charts and a queryable application for people to look at the top sets used for each person.

```
summary(heroItems[[1]]$items)
```

	Length	Class	Mode
head	6	-none-	list
torso	7	-none-	list
feet	6	-none-	list
hands	6	-none-	list
shoulders	6	-none-	list
legs	6	-none-	list
bracers	5	-none-	character
mainHand	5	-none-	character
offHand	5	-none-	character
waist	5	-none-	character
rightFinger	6	-none-	list
leftFinger	6	-none-	list
neck	5	-none-	character

So it's yet another list of other lists. That's fine - I only will need the names of the items, like so:

```
heroItems[[1]]$items$torso$name
```

```
## [1] "Marauder's Carapace"
```

```
heroItems[[4]]$items$torso$name
```

```
## [1] "Cage of the Hellborn"
```

Another way I'll need to subset the unique users is by their class. For example, you can pick one of the 6 following classes:

```
(unique(results$HeroClass))
```

```
## [1] "demon hunter" "wizard"       "monk"         "witch doctor"
## [5] "barbarian"    "crusader"
```

Part A

First. I'll have to use the "heroItems" list and extract within the nested list of nested lists to find the unique sets that each player has.

```
setItems<- lapply(1:1000, function(i) {
  (heroItems[[i]]["items"][[1]][1][[1]]["setItemsEquipped"][[1]])
})
```

Now that I pulled out the unique set items, I can further clean and process the data to create a data frame that contains

```
  - the rank
  - hero class
  - the unique combination of set items
```

```
uniqueSetItems <- plyr::ldply(1:1000, function(i) {
return(c(i,  # Rank of player
         results$HeroClass[i],   # class of player
         paste(setItems[[i]], sep=" ", collapse=",") # Set Item: squish the set items
into one vector
         )
      )
})

# setting the column names to the data frame
colnames(uniqueSetItems)<-c("rank","class","uniqueSet")
```

Great! Now that it's in a usable format, let's look at the unique number of cominbations that people are using in the game from the top 1000 players.

It looks like there are only 107 unique sets out of 1000. Not bad. Out of 6 classes, that means there must be some uniqueness to the combination of sets people are using to get into the top 1000.

Part B.

Now, we need to get the number of unique items in the set.

```
########################################################## have to grab unique items fr
om the "uniqueSets"
# First I'm splitting the "uniqueSet" string by the commas to grab unique items
# then I'm unlisting the items to get a character object of all the unique items
# then i'm looking at the unique values
# then turning that into a usable matrix
uniqueItems<-as.matrix(unique(unlist(strsplit(uniqueSetItems$uniqueSet,","))))

# getting rid of any NA values
uniqueItems<-uniqueItems[!uniqueItems[,1] == "NA", ]

head(uniqueItems)
```

```
## [1] "Unique_Chest_Set_07_x1"    "Unique_Pants_Set_07_x1"
## [3] "Unique_Gloves_Set_07_x1"   "Unique_Shoulder_Set_07_x1"
## [5] "Unique_Boots_Set_07_x1"    "Unique_Helm_Set_07_x1"
```

Next, we're going to use the API again to query back details about the items from BattleNet's API.

First, I create a function to paste together URLs that will be used to get the JSON data back from…

```
itemURLs <- lapply(1:144, function(i) {

  paste0("https://us.api.battle.net/d3/data/item/",
         uniqueItems[i],
         "?locale=en_US&apikey=",
         key)
})
```

Next, I use those urls to grab the JSON format back using "fromJSON".

```
itemData<-lapply(1:144, function(i) {
  fromJSON(itemURLs[[i]])
})

head(itemData[[45]])
```

```
## $id
## [1] "Unique_Pants_Set_06_x1"
##
## $name
## [1] "Firebird's Down"
##
## $icon
## [1] "unique_pants_set_06_x1_demonhunter_male"
##
## $displayColor
## [1] "green"
##
## $tooltipParams
## [1] "item/firebirds-down"
##
## $requiredLevel
## [1] 70
```

Looks great! Now let's join the two data points.

I'm going to now first get the number of unique sets from the uniqueSetItems objects and casting it into a long format to make analysis easier.

```
betterUniqueSetNames<-cSplit(uniqueSetItems, "uniqueSet", sep=",")
betterUniqueSetNames<-betterUniqueSetNames %>% gather(clothing,itemID,uniqueSet_1:uni
queSet_7)
```

Next, I'll pull out the unique "set names" that correspond to each item name.

```
itemsWithSet<- plyr::ldply(1:144, function(i) {
  c(itemData[[i]]$id,itemData[[i]]$name,(itemData[[i]]$set$name))
})
colnames(itemsWithSet)<-c("itemID","itemName","setName")
```

Finally, I'll do an inner join to link each item that a player has with the set name.

```
heroSetDataFinal<-inner_join(betterUniqueSetNames,itemsWithSet, by=c("itemID"))
head(heroSetDataFinal)
```

rank	class	clothing	itemID	itemName	setName
1	demon hunter	uniqueSet_1	Unique_Chest_Set_07_x1	Marauder's Carapace	Embodiment of the Marauder
2	demon	uniqueSet_1	Unique_Helm_Set_03_p2	Accursed Visage	Unhallowed Essence

	hunter				
3	demon hunter	uniqueSet_1	Unique_Chest_Set_07_x1	Marauder's Carapace	Embodiment of the Marauder
4	demon hunter	uniqueSet_1	Unique_Helm_Set_03_p2	Accursed Visage	Unhallowed Essence
5	wizard	uniqueSet_1	Unique_Amulet_007_x1	Tal Rasha's Allegiance	Tal Rasha's Elements
6	demon hunter	uniqueSet_1	Unique_Helm_Set_03_p2	Accursed Visage	Unhallowed Essence

Now, I want to look at the most frequent sets that each class has. This will help me answer research question 2.

I'll create a function called "setClean" to loop through the 6 classes and pull out the most frequent sets for each class.

```
setClean<-function(x){
as.data.frame(xtabs(~ setName + class , x))[!as.data.frame(xtabs(~ setName + class ,
x))$Freq==0, ]
}

classSets1000<-setClean(heroSetDataFinal) #all 1000 top players
classSets100<-setClean(filter(heroSetDataFinal, (rank%in%(1:100)))) #only the top 100
ranked players

head(classSets1000)
```

	setName	class	Freq
6	Immortal King's Call	barbarian	112
8	Might of the Earth	barbarian	46
17	The Legacy of Raekor	barbarian	522
23	Wrath of the Wastes	barbarian	30
25	Armor of Akkhan	crusader	98
36	Roland's Legacy	crusader	6

Results

Research Question 1.

- What player statistics are correlated to the rank of the player?

```
r1 <- rPlot(damage ~ Rank, data = heroStats, type = 'point', color = 'HeroClass')
r1$addControls('x', 'rank', names(heroStats))
save(r1)
```

Research Question 2.

- What are the unique sets that players are using? Does it differ between the top 1000 and the top 100? *

```
library(rCharts)


sets1000Plot <- nPlot(
  Freq ~ class,
  group = "setName",
  data = classSets1000,
  type= "multiBarChart"
)

sets100Plot <- nPlot(
  Freq ~ class,
  group = "setName",
  data = classSets100,
  type= "multiBarChart"
)
```