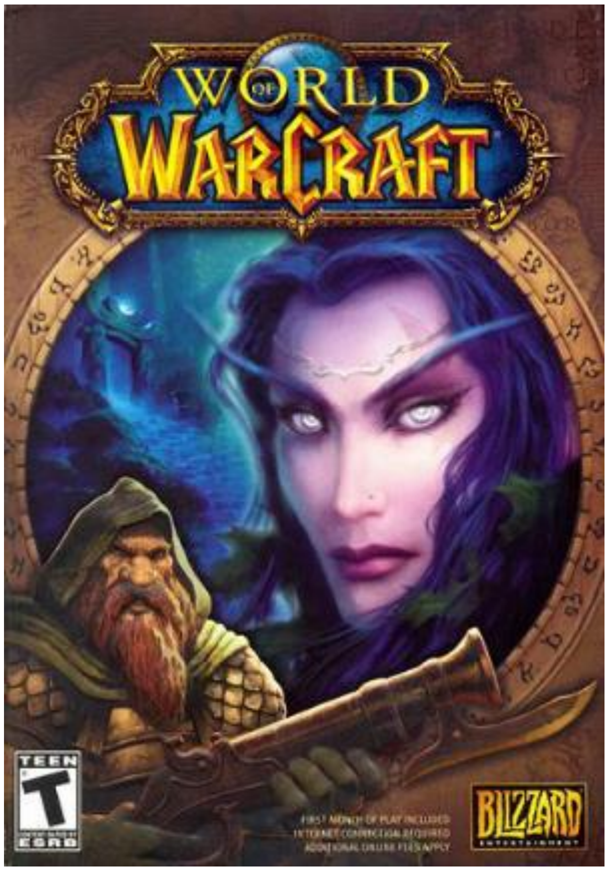


ABSTRACT
In this paper, we analyze the World of Warcraft Avatar History dataset, to help video game designers gain insight of player behaviors and habits in order to create a competitive product and drive revenue.

Morgan Blein
ITM_6899: Capstone

ANALYSIS OF VIDEO GAME DATASET

OUTLINE:

1) <u>Introduction:</u>

The video game industry was profoundly transformed by the advent of home internet and computerization of homes. Today, in the US, there are over 286 million internet users, accounting for almost 90% of the population[1]. The rise in speeds and accessibility as well as the decrease of costs allowed for people to play from the comfort of their houses, while socializing online with individuals who share the same passion for this industry. In fact, over a third of internet users now are involved in some type of social game and this share will keep growing.[2]

In this paper, we will study and analyze player characteristics from a famous Massively Multiplayer Online Role Playing Game (MMO RPG): World of Warcraft (WoW) developed by the studio Blizzard. According to a recent study, the number of monthly active MMO subscribers worldwide, as of 2014, was an estimated 23.5 million, with a revenue of 19 billion in 2011 and estimated 35 million in 2017[3]. World of Warcraft captured at some point over 12 million users[4] (Almost half of all game subscriptions).

The objective of this research is to help game designer understand their customers. How can they retain as many customers as possible and create a pleasant game experience? To do so, the game designer need to understand player interactions and choices, as it relates to in-game design. User data can be collected and analyzed to address issues such as game balance, social behavior, variation of players online, or subscription analysis. Having a clear understanding of this data can help game designer create a seamless experience for users. By doing so, they can create value for their product and generate revenue.

2) <u>Data Set</u>

2.1) Motivation:

The reason I chose to work with this dataset is that it holds many observations for a field that passionate me. Moreover, I am a World of Warcraft user so the content of the analysis can easily relate to me. The dataset available is very thorough and hold many observations.

2.2) Source:

The data set can be found at this link: http://mmnet.iis.sinica.edu.tw/dl/wowah/

All credits go to the institute of information science team for the data collection.

[1] http://www.internetlivestats.com/internet-users-by-country
[2] See exhibit A: percentage of US internet users that play social games
[3] http://www.statista.com/statistics/240987/global-online-games-revenue
[4] see exhibit B: Number of World of Warcraft subscribers:1st quarter 2010

2.3) Methodology of collection:

Thanks to the World of Warcraft API the team was able to run a script on the Taiwanese servers of the game for about 3 years, allowing for massive data collection of all users and their characteristics.[5] Queries allow the user to ask the game server about who is connected alongside them. Some attributes of these other players can be reported as well.

2.4) Original data:

The file to download is a RAR archive: 577,334,422 bytes compressed, about 3.4 gigabytes uncompressed. The original data downloaded can be summarized as such[6]:

- Timeframe: 1107 days from 01/01/2006 to 01/10/2009
- 114 samples per day
- 91065 different players
- Over 28 million observations

The uncompressed data contains over 130,000 separated text files.

2.5) Transformed data:

In order to analyze this data, I chose to modify it. To do so, I used both Python and STATA. The format of thousands of text files spread in many folders and subfolder was not conducive to the type of analysis and programs I wanted to use.

To see how I modified the data, please refer to exhibit E: Steps in Data Transformation.

After transformation, we have a single table structured file with over 28.5 million rows of observation!

2.6) Variables and Attributes description:

The fields I kept are:

Char: character individual ID. Each individual avatar recorded have their own number, no two person or characters share this ID. Basically Char=Player ID.

Guild: an integer to specify the guild (collection of players organized together). For example of guild=x, then the observation specifies that this specific character belong to guild x. This field is interesting to find corresponding traits shared between players in the same guilds.

Timestamp: a record of the date as a string at which the observation is made. The format is the following:

MM/DD/YYYY HH:MM:SS

Level: the in game level of the specified observation or character.

[5] See Exhibit C: Data collection methodology for Original data from WOWAH Dataset team.

[6] See Exhibit D: summary of original data.

RaceInGame: the race the player decided to pick. There are several races with difference characteristics players can choose from when starting the game.

Possible values: Orc, Undead, Tauren, troll.

ClassInGame: The class the payer decided to pick. Just like race, classes vary and can drastically change the game experience. Whether you choose to heal your friends, damage enemies or other, this choice will alter your role within the game.

Possible values[7]: Warrior, hunter, Rogue, Priest, Shaman, Mage, Warlock and Druid.

Both race and class observation can hold a lot of value in order to know what people prefer to pick. Game companies can resort to analyses to make sure the game is balanced in order for as many customers to have a pleasure-full experience.

3) <u>Analysis:</u>

In this section we will describe the types of analysis performed as well as their relevance to our goal of helping game designer understand their customers and address game issues.

3.1) Gameplay balance

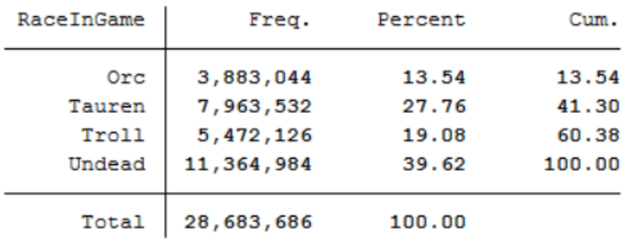
The purpose of this section is to take a closer look at what players tend to choose most out of their choices of races and classes. In a Massive Online game such as World of Warcraft, diversity is encouraged. The game is designed for all races and classes to be different, yet equal in terms of power or utilities. If we notice a greater influx of players towards a certain class or race, it could point to balance issues. Balance is regularly addressed in these types of games through balance patches. (That can be released at will, sometimes even weekly)

In order to generate these tables and graph, simply run the EDA.do file provided (See code exhibit 1). It outputs table with frequency distributions for each of the attributes of the fields. It also automatically generates pie charts for these distributions.

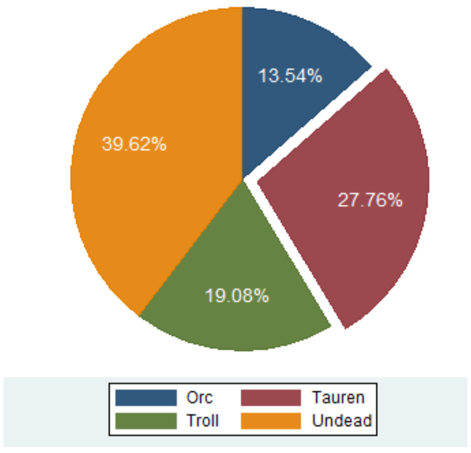
First, let’s address the balance issue using RaceinGame and ClassinGame fields:

[7] You can see an example of class and race selection exhibit F: in-game example of race and class selection

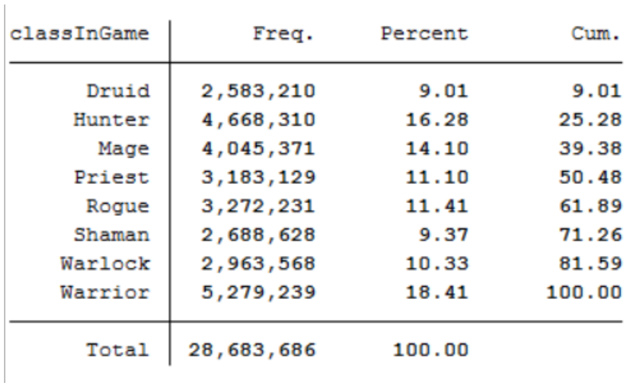
Raceingame:



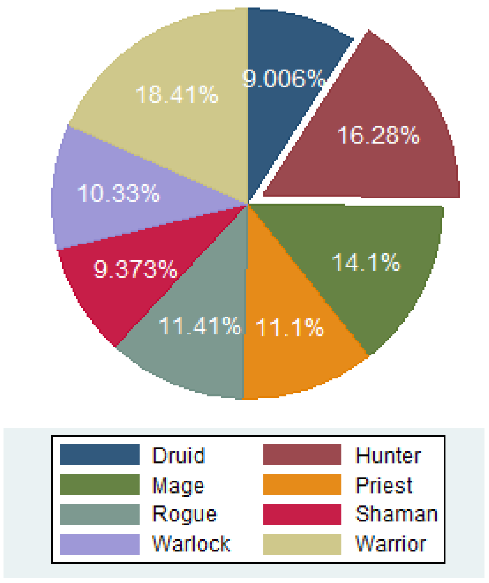
RaceInGame	Freq.	Percent	Cum.
Orc	3,883,044	13.54	13.54
Tauren	7,963,532	27.76	41.30
Troll	5,472,126	19.08	60.38
Undead	11,364,984	39.62	100.00
Total	28,683,686	100.00	



ClassIngame:



classInGame	Freq.	Percent	Cum.
Druid	2,583,210	9.01	9.01
Hunter	4,668,310	16.28	25.28
Mage	4,045,371	14.10	39.38
Priest	3,183,129	11.10	50.48
Rogue	3,272,231	11.41	61.89
Shaman	2,688,628	9.37	71.26
Warlock	2,963,568	10.33	81.59
Warrior	5,279,239	18.41	100.00
Total	28,683,686	100.00	



As seen in the tables and graph above, we can find some choices or race and classes that are a lot more popular than others. The most popular race is undead, accounting for almost 40% of the whole dataset. On the other hand, the least popular is Orc with only 13.5% of the players choosing this race.

For classes, the distribution is more even. There are 8 choices, so the average should be 100/8=12.5%. Most of the races (Priest, Rogue, Shaman, Warlock, Druid and Mage) are within a reasonably close distribution (from 9% to 14%). The only outliers, more popular than most option are Warrior and Hunter.

Blizzard (World of Warcraft creator) would want to address this un-even distribution. As a matter of fact, there has been dozens of balance patches to address these kinds of issues since 2009. The game version at the end of this analysis was Patch 3.3.0. They are now on patch 6.2.0. The second decimal represents a minor fix (glitches), the first decimal usually a balance patch and the first number a deeper change in the game[8] (graphics overhaul for example). Balance in video games is very fragile and needs close monitoring in order not to be abused. If abused, developers can lose huge shares of subscribers due to the "unfair" nature of the game.

3.2) Social behavior

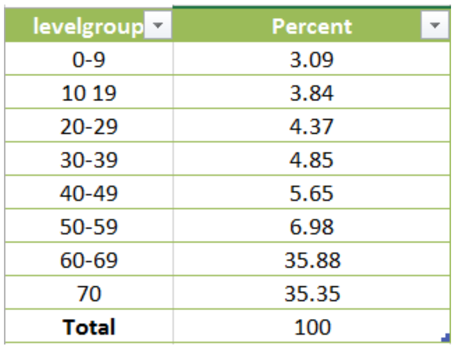
The level of the player as well as their belonging to a guild start showing us insights towards to social aspect of the game. This is my opinion what makes the game so attractive. If game designer manage to get a more efficient way for people to interact in game, and spend many hours on their product, they will drive more revenue. Today, single player games are a dying breed. People love social interaction and the challenges it creates.

First let's, take a look at the level distribution. Level can vary from 0 to 70 at the time of this dataset collection. For visualization convenience, I grouped level together such as level 0-9, level 10-19 all the way to 06-69 are grouped together. Level 70 being the maximum, it is grouped only with other level 70.

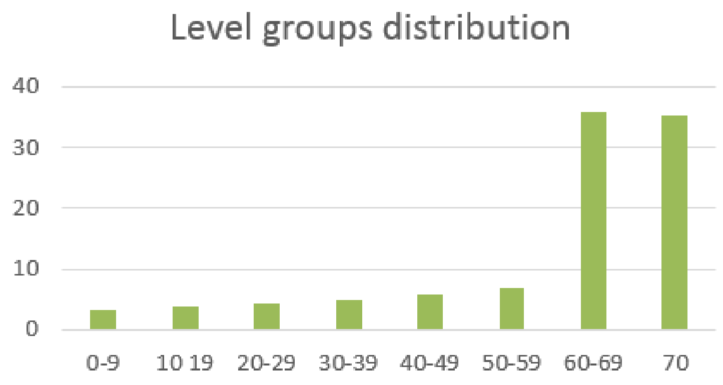
Note: the script EDA.do (code exhibit 1) created the distribution tables for us, however I do not like how the histogram look from STATA, therefore I used excel to graph.

[8] http://us.battle.net/wow/en/game/patch-notes/3-3-0

Level:



levelgroup	Percent
0-9	3.09
10 19	3.84
20-29	4.37
30-39	4.85
40-49	5.65
50-59	6.98
60-69	35.88
70	35.35
Total	100

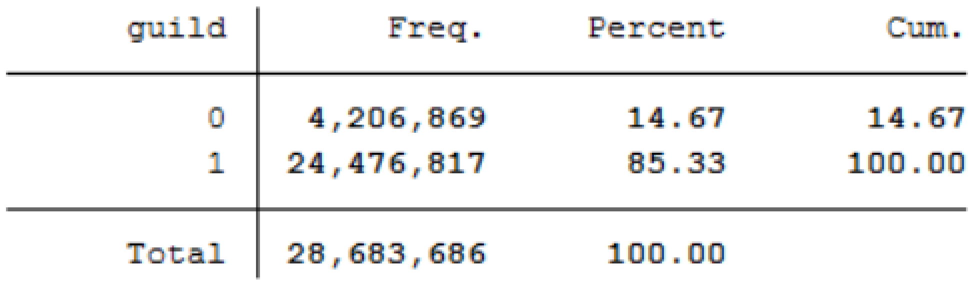


This Analysis confirm what we previously assumed. Players spend a lot of time on this game. We can see that high levels, from 60 to 70, account for over 70% of the data observed. People are interested in the type of gameplay offered at high level. Most groups will only allow high level players too, therefore to “fit-in” players are forced to spend a long time leveling up. It is a commitment and shows how passionate people who play this game are.

Guild:

The belonging to some kind of socially organized group on such a game is really what keeps people playing and coming back. Let’s now take a look at how socially involved players in this dataset are.

As described in the field’s description section of the data set, the field Guild represents the belonging to a certain group. I decided to just look whether people did belong to a guild or not. Therefore, I recoded the GUILD fields to a binary, with 0=not belonging to a guild, and 1 belonging to a guild (See code exhibit 1: EDA.do script, at the very end).



guild	Freq.	Percent	Cum.
0	4,206,869	14.67	14.67
1	24,476,817	85.33	100.00
Total	28,683,686	100.00	

The frequency table for guild belonging shows an overwhelming majority (over 85%) of the players are involved in an in-game guild. This is in my opinion a huge finding. World of Warcraft managed to create an environment where people come together and are organized for a specific purpose.

On a game design standpoint, this data points to high social interaction and high dedication to time spent on this game. In fact, some research on daily play time has been done for this dataset: "We find that 75% of gamers play longer than 1.9 hours per day on average, and 25% play longer than 4. 9 hours per day, which again indicates that the game is very attractive to the gamers."[9]

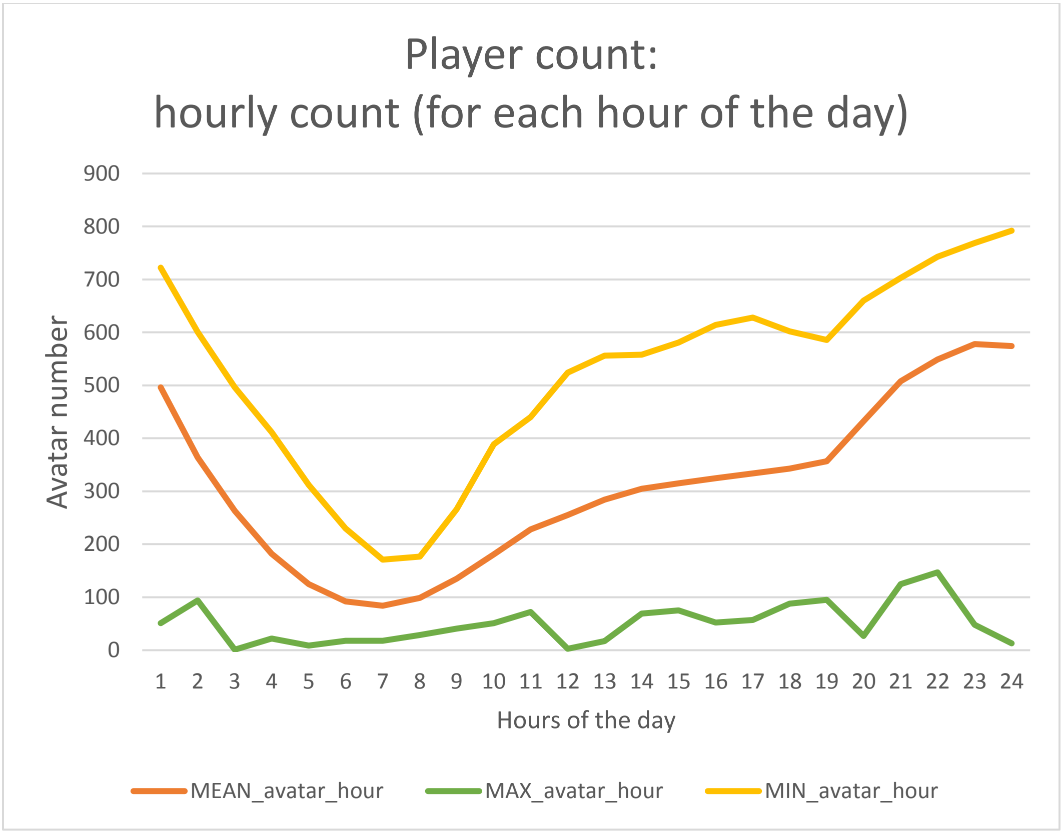
3.3) Time - Trend analysis: number of avatars online

Now that we understand how heavily players socially interact in this game and we know they spend a high volume of hours daily for this leisure, we can take a closer look at how many players are connected on this server, and at what times. The change in the count results for characters can give us insight as to when expect more or less demand for the servers. This can be a very useful thing to know for game companies, who can appropriately allocate server space for crowded times, or on the other hand provide incentive for players to connect more frequently or at different times. Predicting this demand ahead of time will allow for game designers to allocate costs with more accuracy, while still making sure their infrastructure is solid enough to handle the number of players online.

The analysis is performed in STATA. To replicate the results, run dates_and_times.do script (see code exhibit 2). It will create respective csv files with cumulative counts mean, maximum and minimum count values for the desired time frame.

[9] http://mmnet.iis.sinica.edu.tw/pub/lee11_wowah.pdf

Daily count: variation of player per hour basis:

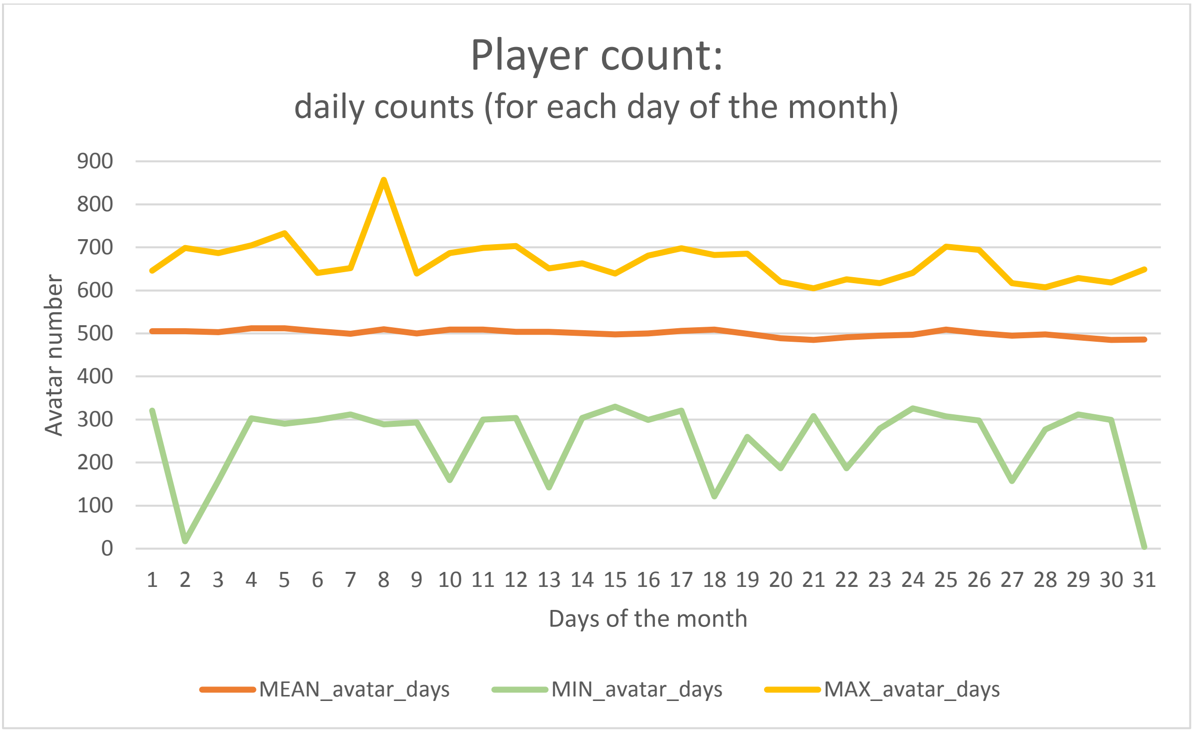


The graph above shows the mean number of player connected to the game servers (as well as minimum and maximum). We can see that variation throughout the day is huge: the mean varies from 84 players at 6AMto 578 at 10PM.

These numbers make sense: even hardcore gamers need to sleep. However we can notice that from 1 to 2 AM, there are still more people connected than during day hours. Most of the World of Warcraft users are people that work, it is interesting to point out they decide to cut hours off their sleep schedules in order to play. Most social guild activities such as donjon raids or player battles will also happen at night, giving players more reason to be connected from 7PM to 2AM. As mentioned by a similar research: “This finding shows that many people play all night, and therefore implies that the game is addictive”[10]

[10] http://mmnet.iis.sinica.edu.tw/pub/lee11_wowah.pdf

Monthly count: variation of player per Day basis:



Unlike hourly counts that held huge variations, the mean number of players connected throughout the month on a daily basis does not change much. We can see the mean is very constant, hovering around 500 players regardless of the day of the month.

The minimum and maximum curves offer a lot of variation. However, I believe the 2 occurrences when the minimum hits 0 represents days when World of Warcraft was closed for maintenance. Since these values are the minimum return by the count of every day and there is over 3 years of data, servers were bound to close at some point.

3.4) Subscription time analysis:

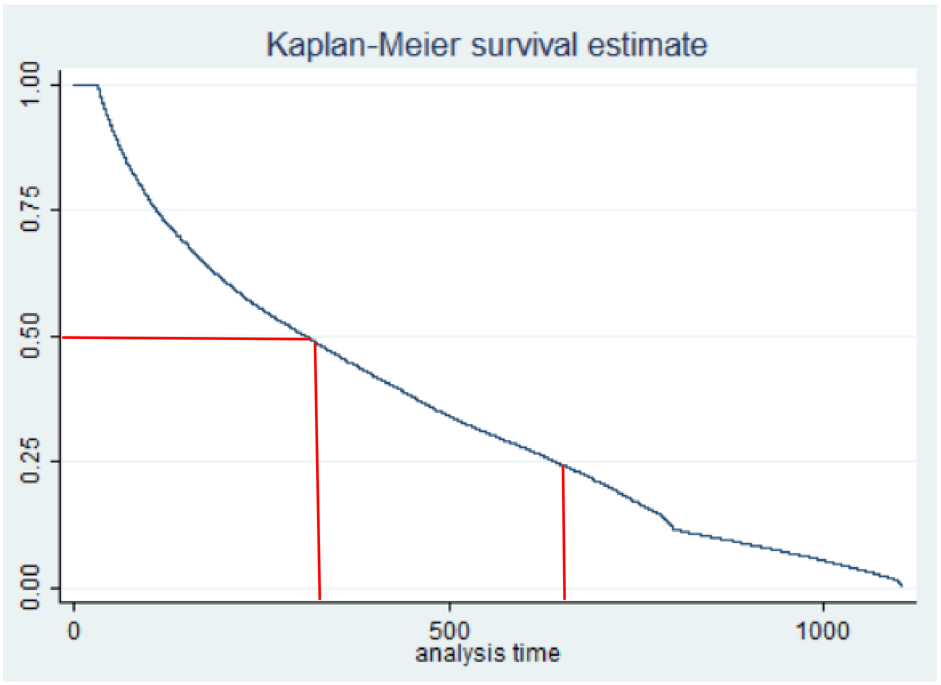
Now that we have a clear idea of the volume of players online, we can go back to individual behavior. The main stream of revenue for massively multiplayer online role-playing game is player subscription fees. Therefore, for a game to be successful, developers want to make subscription time as long as possible. Our dataset gives us query times of when specific users are online. Therefore with some analysis we can extract the subscription time of each of the players. This subscription time is calculated as the spread (in days) from first to last login. It shows how long players stay interested in the game.

To do so, Use subtime.do script. (See code exhibit 3) This script outputs a table that summarize the subscription time and gives us statistical distribution:

```
                        (mean) between
-------------------------------------------------------------
      Percentiles      Smallest
 1%           31             30
 5%           39             30
10%           52             30       Obs               20162
25%          107             30       Sum of Wgt.       20162

50%          304                      Mean           389.2973
                        Largest       Std. Dev.      312.5214
75%          638           1105
90%          848           1105       Variance       97669.63
95%         1010           1105       Skewness       .6600016
99%         1101           1105       Kurtosis       2.251081
```

As we can see from this output, the mean subscription time for paid customers is 389 days (over a year). However this data is heavily skewed. Some users are online for almost every single day throughout the 1107 day period. The 99th percentile represent very long term users: they have an estimated average subscription time of 1101 days! Again, we can see that some user get very much addicted by this fantasy world.

Another interesting output from the script is the Kaplan-Meier survival estimate curve for our subscription time. This method offers a robust approach for visualization of the subscription time:



This shows again that 50% of users will still subscribe past 380 days. The highly negative slope for the first 2 or 3 months represents most of the casual users that try many games and do not commit much. After this, we can see the curve stabilizes more to capture most of the players that have a genuine interest in the game. As a fact, 25% of the users will have subscription times higher than 630 days.

As it relates to business and the main goal of any organization to generate profit, we can see that subscription times are high for this game in particular. Conducting this type of analysis will allow a marketing department to compute average customer lifetime value (ACLV) with more accuracy. Some studies have pushed this analysis further and predicted the un-subscription dates of players, using a dataset from another game. It is mentioned: “we can see that the accuracy for predicting whether a gamer is going to leave in a month is around 82% for gamers who subscribed for more than one year, and 85% for gamers who subscribed for more than two years. The results indicate that, it is feasible to use our scheme to predict whether they are quitting from the game in the near future”[11] This study proves that we can analyze game users behavior in order to predict subscription numbers for a specific game. Again, this is very valuable information for a sales or marketing department to compute KPIs and be more accurate.

4) Conclusion:

The analysis we ran, whether Gameplay balance, Social behavior, variation in the number of players online or Subscription time analysis all can give game developers a great deal of insight on how to create a better product. By doing so, not only will they generate more profit by offering value to their customers, they can also control their costs and make solid predictions for the game’s future. In our case, this study demonstrated that World of Warcraft subscribers not only keep paying for the game for a very long time in average, they are socially tied to it. Many of the game component appeal to them on a social level: things such as the belonging to a guild, the hours spent every day (mostly late at night) tend to point to a very immersive, almost addictive experience. On a game developer stand point, we can safely say that the World of Warcraft team did a good job designing the game. However as all successful products, they are now facing fierce competition. It will be interesting to see how the game evolves to try to keep a massive share of online gaming revenues.

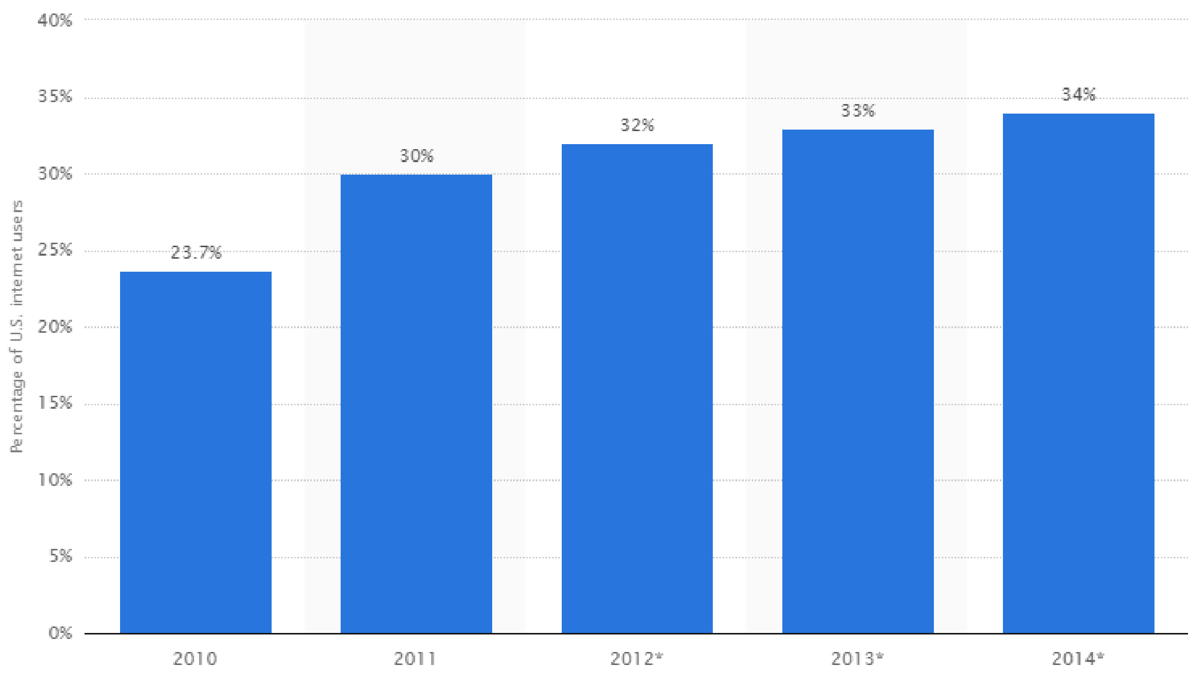
[11] http://mmnet.iis.sinica.edu.tw/publication_detail.html?key=tarng09_gamer_departure

Exhibits

Exhibit A: percentage of US internet users that play social games:

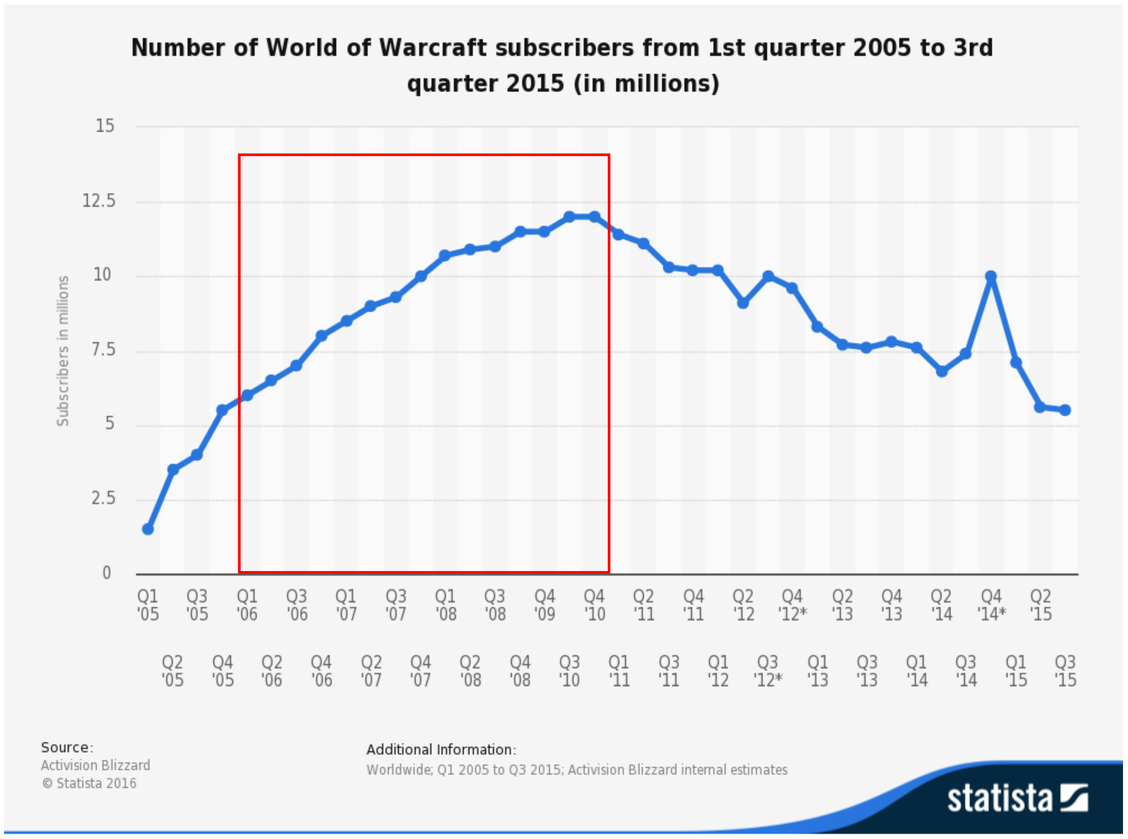
Percentage of U.S. internet users that play social games from 2010 to 2014

This forecast illustrates the percentage of internet users in the United States that play social games from 2010 to 2014. In 2014, 34 percent of U.S. internet users are expected to play games on a social network.



http://www.statista.com/statistics/234655/social-gamers-in-percent-of-internet-users-forecast/

Exhibit B: Number of World of Warcraft subscribers:



http://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/

Exhibit C: Data collection methodology for Original data from WOWAH Dataset team:

Data Collection Methodology

- Create a game character
- Use the command *'\who'*
- The command asks the game server to reply with a list of players who are currently online

- Write a specialized data-collection program (using C#, VBScript, and Lua)

Exhibit D: summary of original data.

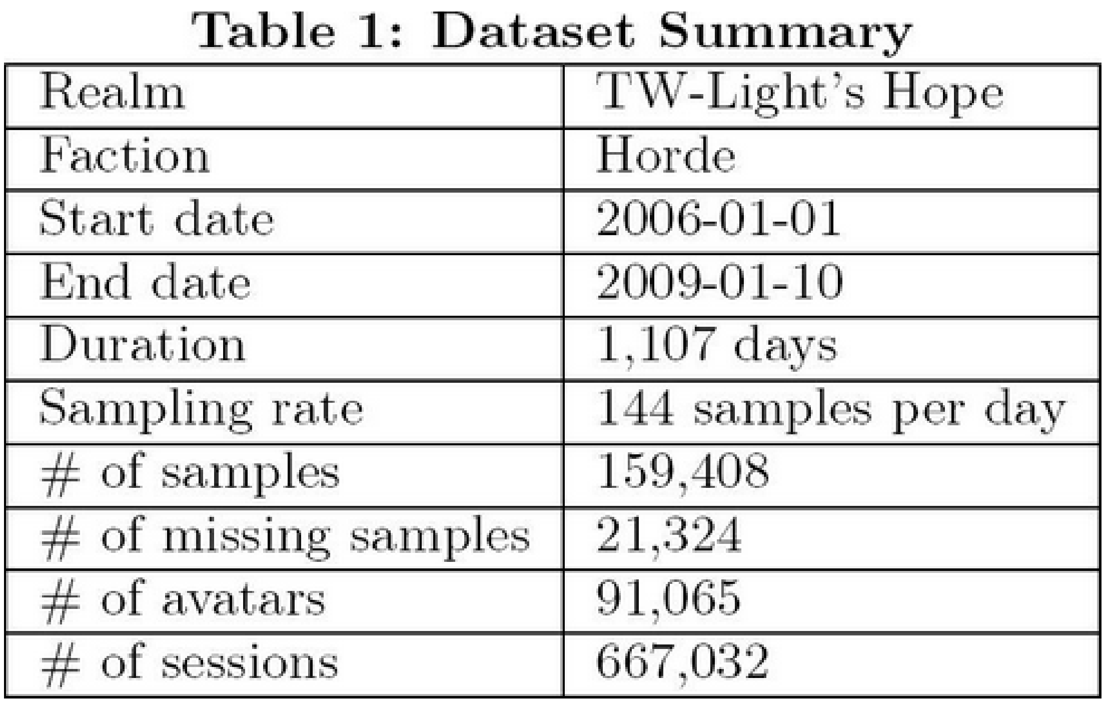
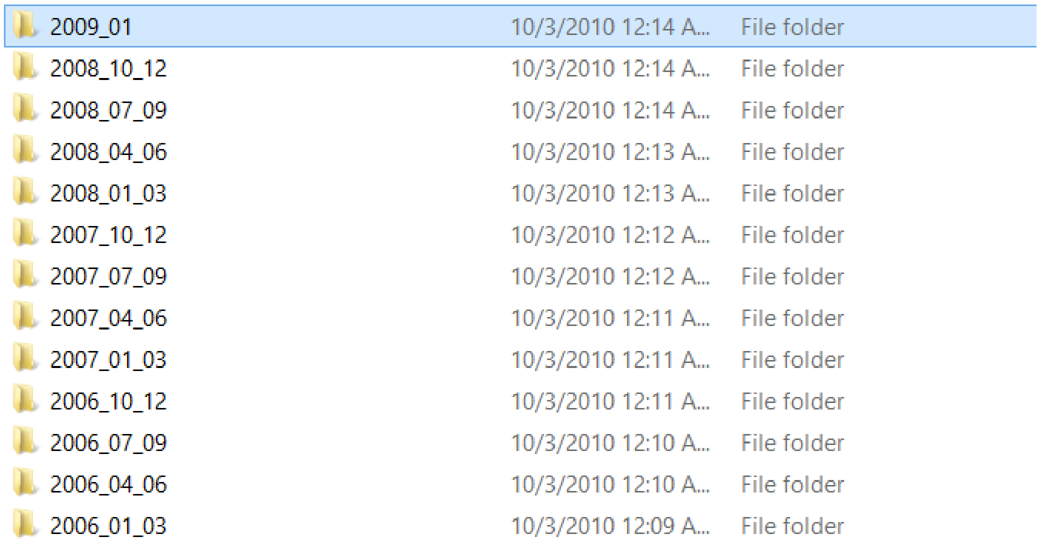


Table 1: Dataset Summary

Realm	TW-Light's Hope
Faction	Horde
Start date	2006-01-01
End date	2009-01-10
Duration	1,107 days
Sampling rate	144 samples per day
# of samples	159,408
# of missing samples	21,324
# of avatars	91,065
# of sessions	667,032

Exhibit E: Steps in data transformation:

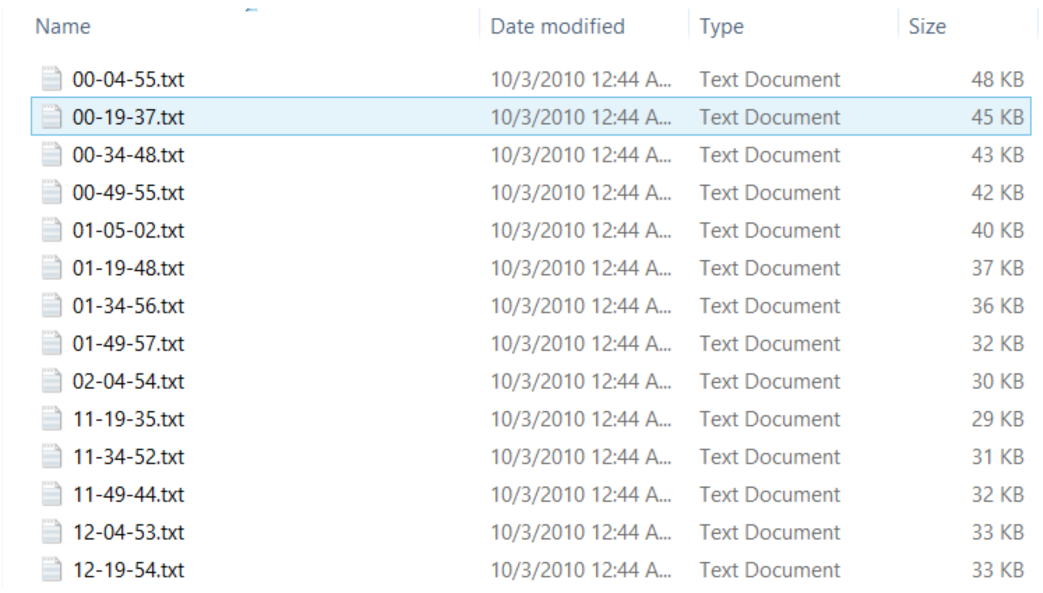
The uncompressed data contains over 130,000 text files. Each of those corresponds to a time interval. (1,144 query times a day corresponding to observations every 10 min) When extracting the data, we can see that many subfolders are created such as each subfolder represents several days of data:



2009_01	10/3/2010 12:14 A...	File folder
2008_10_12	10/3/2010 12:14 A...	File folder
2008_07_09	10/3/2010 12:14 A...	File folder
2008_04_06	10/3/2010 12:13 A...	File folder
2008_01_03	10/3/2010 12:13 A...	File folder
2007_10_12	10/3/2010 12:12 A...	File folder
2007_07_09	10/3/2010 12:12 A...	File folder
2007_04_06	10/3/2010 12:11 A...	File folder
2007_01_03	10/3/2010 12:11 A...	File folder
2006_10_12	10/3/2010 12:11 A...	File folder
2006_07_09	10/3/2010 12:10 A...	File folder
2006_04_06	10/3/2010 12:10 A...	File folder
2006_01_03	10/3/2010 12:09 A...	File folder

Such as $\wowah\2009_01 for example

Each of these subfolder will hold several folders of their own (representing all the queries for each day). These subfolder then respectively contain all text files resulting in running the queries for each day of the collection:



Name	Date modified	Type	Size
00-04-55.txt	10/3/2010 12:44 A...	Text Document	48 KB
00-19-37.txt	10/3/2010 12:44 A...	Text Document	45 KB
00-34-48.txt	10/3/2010 12:44 A...	Text Document	43 KB
00-49-55.txt	10/3/2010 12:44 A...	Text Document	42 KB
01-05-02.txt	10/3/2010 12:44 A...	Text Document	40 KB
01-19-48.txt	10/3/2010 12:44 A...	Text Document	37 KB
01-34-56.txt	10/3/2010 12:44 A...	Text Document	36 KB
01-49-57.txt	10/3/2010 12:44 A...	Text Document	32 KB
02-04-54.txt	10/3/2010 12:44 A...	Text Document	30 KB
11-19-35.txt	10/3/2010 12:44 A...	Text Document	29 KB
11-34-52.txt	10/3/2010 12:44 A...	Text Document	31 KB
11-49-44.txt	10/3/2010 12:44 A...	Text Document	32 KB
12-04-53.txt	10/3/2010 12:44 A...	Text Document	33 KB
12-19-54.txt	10/3/2010 12:44 A...	Text Document	33 KB

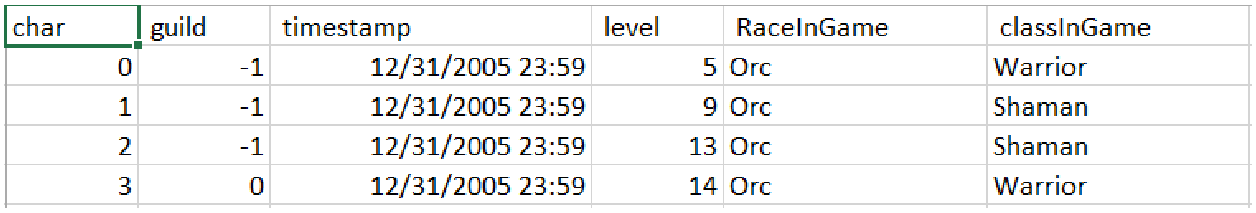
Such as $\WoWAH\2008_10_12\2008-10-01\00-04-55.txt for example

1) **Python: WOWAH_tocsv.py** (see Code exhibit 4: python program for consolidation into csv)

I used this script I edited in order to consolidate the 130, 000 text files into 1 massive CSV file.

I also used this step to screen out any part of the data that I did not want (example: dummy fields, array structures)

Here is what the data looks like after being treated by the python script:



char	guild	timestamp	level	RaceInGame	classInGame
0	-1	12/31/2005 23:59	5	Orc	Warrior
1	-1	12/31/2005 23:59	9	Orc	Shaman
2	-1	12/31/2005 23:59	13	Orc	Shaman
3	0	12/31/2005 23:59	14	Orc	Warrior

2) **STATA**

After generating the CSV using Python, I decided to use STATA to further modify my data and thereafter run my analysis.

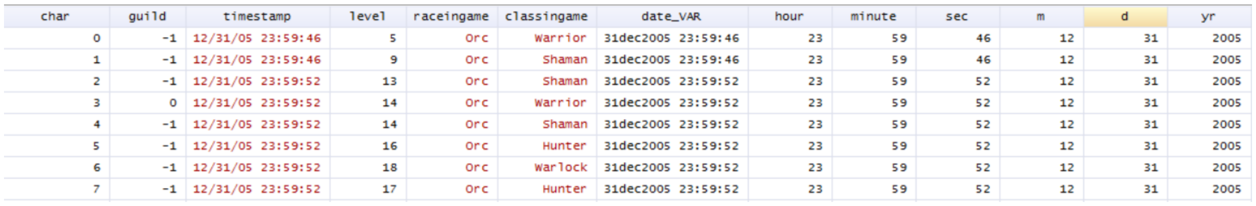
First, export the csv in STATA. (File/import/CSV) Save the data in STATA format. (.dta)

Next, run the Generatedate.do script (it is really short so I will just paste the code here):

```
*use "C:\wowah\averages.dta", clear

gen double date_VAR = clock(timestamp,"MD20Yhms")

format date_VAR %tc

gen hour = hh(date_VAR)

gen minute = mm(date_VAR)

gen sec = ss(date_VAR)

gen double date21 = date(timestamp,"MD20Yhms")

format date21 %tc

generate m=month(date21)

generate d=day(date21)

generate yr=year(date21)
```

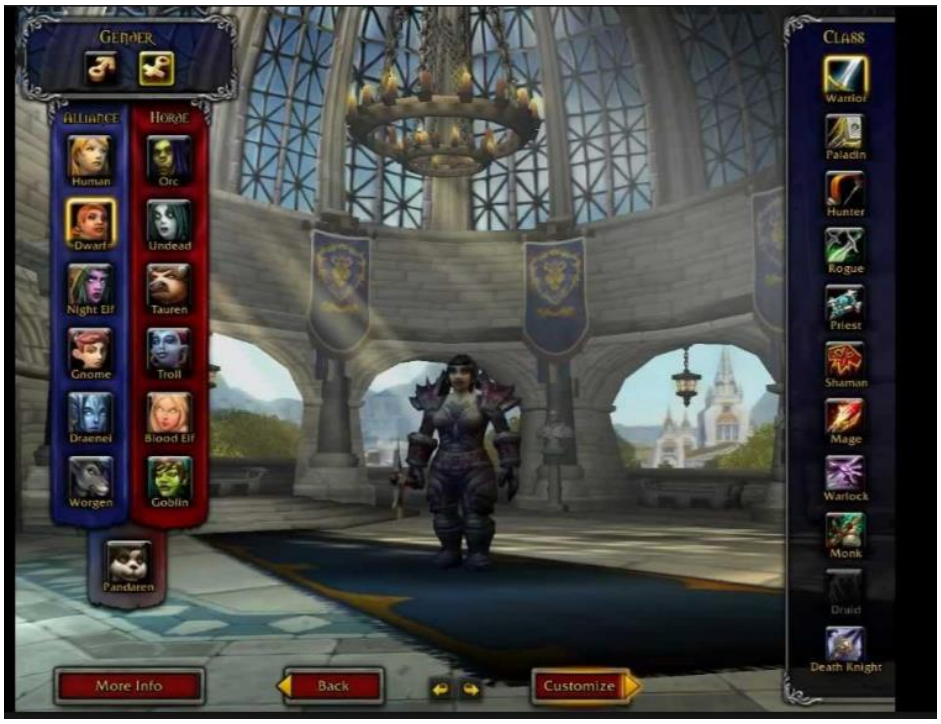
This code snippet allows me to make the timestamp string a variable DATE usable for analysis. It automatically extracts the date information from the timestamp string and return a date variable, as well as individual variables for year, month, day, hour, minute and second. These variables will be useful to perform analysis such as Daily/weekly/monthly trends.

After running the script, here is an excerpt of the data:



char	guild	timestamp	level	raceingame	classingame	date_VAR	hour	minute	sec	m	d	yr
0	-1	12/31/05 23:59:46	5	Orc	Warrior	31dec2005 23:59:46	23	59	46	12	31	2005
1	-1	12/31/05 23:59:46	9	Orc	Shaman	31dec2005 23:59:46	23	59	46	12	31	2005
2	-1	12/31/05 23:59:52	13	Orc	Shaman	31dec2005 23:59:52	23	59	52	12	31	2005
3	0	12/31/05 23:59:52	14	Orc	Warrior	31dec2005 23:59:52	23	59	52	12	31	2005
4	-1	12/31/05 23:59:52	14	Orc	Shaman	31dec2005 23:59:52	23	59	52	12	31	2005
5	-1	12/31/05 23:59:52	16	Orc	Hunter	31dec2005 23:59:52	23	59	52	12	31	2005
6	-1	12/31/05 23:59:52	18	Orc	Warlock	31dec2005 23:59:52	23	59	52	12	31	2005
7	-1	12/31/05 23:59:52	17	Orc	Hunter	31dec2005 23:59:52	23	59	52	12	31	2005

Exhibit F: in-game example of race and class selection:



Code exhibit 1: stata do file: EDA.do

```
*load the data

use "C:\wowah\CLEANED_DATA.dta", clear

*use "C:\wowah\averages.dta", clear


*create frequency table for race choice by player

tabulate  raceingame

*creates graph for the distrubtion of races chosen by players.
```

```
*with legend and percentages on the graph.
graph pie, over(raceingame) pie(2,explode) plabel(_all percent, size(*1.5) color(white))legend(on)
*save graph: edit to your path:
graph save Graph "C:\wowah\raceingame_distribution.gph

*create frequency table for race choice by player
tabulate  classingame
*creates graph for the distrubtion of races chosen by players.
graph pie, over(classingame) pie(2,explode) plabel(_all percent, size(*1.5) color(white))legend(on)
*save graph: edit to your path:
graph save Graph "C:\wowah\classingame_distribution.gph
*re-encode the level in groups for visualization
egen levelgroup1 = cut(level), at(0,10,20,30,40,50,60,70,80) label
*TO DO: rename labels.
tabulate  levelgroup
histogram levelgroup
*to do: get the pie charts.

*rework the guild field such as if char belongs to a guild, set to 1, if not, set to 0.

replace guild =1 if guild >= 0
replace guild =0 if guild < 0
*recode to binary
recast byte guild
tabulate  guild

save "C:\wowah\CLEANED_DATA_EDA.dta", replace
```

Code exhibit 2: stata do file: dates_and_time.do

```
Created by Morgan Blein
*May/June 2016.
*feel free to copy use and modify this code.

*load the data
use "C:\wowah\CLEANED_DATA.dta", clear
*from string, generate a date variable that STATA understands and extract days month and year.
gen double date_VAR = clock(timestamp,"MD20Yhms")
format date_VAR %tc
gen hour = hh(date_VAR)
gen minute = mm(date_VAR)
gen sec = ss(date_VAR)
gen double date21 = date(timestamp,"MD20Yhms")
format date21 %tc
generate month_VAR=month(date21)
generate day_VAR=day(date21)
generate year_VAR=year(date21)

*number of player online for 24h: hourly averages.
bysort hour date21 char : gen tag = _n ==1
by hour date21 : gen COUNT_avatar_hour = sum(tag)
by hour date21 : replace COUNT_avatar_hour = COUNT_avatar_hour[_N]
by hour date21 : replace tag = _n ==1

*get mean, max and min for each hour
by hour: egen MEAN_avatar_hour=mean(COUNT_avatar_hour)
```

```
replace MEAN_avatar_hour = round(MEAN_avatar_hour)

by hour: egen MIN_avatar_hour=min(COUNT_avatar_hour)

by hour: egen MAX_avatar_hour=max(COUNT_avatar_hour)



*number of player online monthly daily averages


bysort day_VAR date21 char : gen tag2 = _n ==1

by day_VAR date21 : gen COUNT_avatar_days = sum(tag2)

by day_VAR date21 : replace COUNT_avatar_days = COUNT_avatar_days[_N]

by day_VAR date21 : replace tag2 = _n ==1


*get mean, max and min for each hour

*there are 3 years. we need to divide those results by 3. we then round up to avoid decimals

by day_VAR: egen MEAN_avatar_days=mean(COUNT_avatar_days/3)

replace MEAN_avatar_days = round(MEAN_avatar_days)

by day_VAR: egen MIN_avatar_days=min(COUNT_avatar_days/3)

replace MIN_avatar_days = round(MIN_avatar_days)

by day_VAR: egen MAX_avatar_days=max(COUNT_avatar_days/3)

replace MAX_avatar_days = round(MAX_avatar_days)


*generating days of the week.

*gen days_of_week_index = dow( mdy( m, day, yr) )

*creates an index such as: 0=Sunday,...,6=Saturday.

*TODO: figure out what to do with days.


save "C:\wowah\CLEANED_DATA_DATES.dta", replace

*collapse data to export for graphs

*hours

collapse MEAN_avatar_hour MAX_avatar_hour MIN_avatar_hour, by(hour)
```

```
export delimited using "C:\wowah\HOUR_averages.csv", replace

*days:
use "C:\wowah\CLEANED_DATA_DATES.dta", clear
collapse MEAN_avatar_days MIN_avatar_days MAX_avatar_days, by(day_VAR)
export delimited using "C:\wowah\DAYS_averages.csv", replace
```

Code exhibit 3: stata do file: subtime.do

```
*load the data
use "C:\wowah\CLEANED_DATA.dta", clear
gen double date21 = date(timestamp,"MD20Y###")
format date21 %td
by char: egen lastdate2 = max(date21)
by char: egen firstdate2 = min(date21)

gen between = lastdate2 - firstdate2
save "C:\wowah\CLEANED_DATA_SUBTIME.dta", replace
collapse between , by(char)
*bysort char (between) : keep if _n == _N
*drop

drop  if between<31
*dropped the values for non paying custoemrs as the first month is free
summarize  between, detail

stset between
sts graph
```

Code exhibit 4: python program for consolidation into csv:

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 12 16:50:42 2016
This code was created using parts of Miles Oneills code: https://github.com/myles-oneill/WoWAH-parser

@author: morgan

"""




# PACKAGES
import csv
import re
import os          # This is for os.listdir
import os.path     # This is for the other dir stuff.
import string      # Maybe for directory name cycling etc.
import time                        # For timing how long it takes.

# VARIABLES
max_char = 0            # Tracks char ID for error testing.
max_guild = 0           # Tracks guild ID for error testing.s
mfc = 0                            # Tracks how many little files were counted.
write_data_loc = "C:\\wowah"                                                        # Adjust to your
dir as needed.
write_data_file = '/wowah_data_timecharguildlevelraceclass.csv'
write_data_filename = write_data_loc + write_data_file
```

```
the_dir = "C:\\wowah\\WoWAH\\"                                                    # This
is where the WoWAH folders are located, adjust as needed. Have them in their own subdir.


#REGEX
line_re = re.compile(r'^.*"[\d+],\s(.*),\s(\d+),(\d+),\s?(\d*),\s(\d+),\s(\w+),\s(\w+),.*".*$')
#                dummy  time   seq  char  guild  level   race  class



# FUNCTIONS


def get_subdirs(the_folder):
    this_list = []
    this_list = os.listdir(the_folder)
    print 'From get_subdirs, a list is: ', this_list     # Printing for error control.
    for item in this_list:
        if item.startswith('.'):
            this_list.remove(item)
    return(this_list)
# End of get_subdirs
# '.DS_Store'



def get_file_list(the_folder):
    this_list = []
    this_list = os.listdir(the_folder)
    for item in this_list:
        if item.startswith('.'):
            this_list.remove(item)
    return(this_list)
# End of get_file_list
```

```
def parse_and_write(file, output_file):
    for line in file:
#       print 'A line is: ', line
        data = line_re.match(line)
        if data is not(None):
            timestamp = data.group(1)
            char = data.group(3)
            Level = data.group(5)
            RaceInGame = data.group(6)
            classInGame = data.group(7)
            if data.group(4) is not(''):
                guild = data.group(4)
            else:
                guild = '-1'   # Note there are some missing values, i.e. errant -1.
            print timestamp    # This is so you can keep track of where it is. Max Jan 2009 IIRC.



            #zoneInGame = data.group(8)


            new_line = char + ',' + guild + ',' + timestamp + ',' + Level + ',' + RaceInGame + ',' + classInGame
+'\n'
            output_file.write(new_line)


        else:
            print "Didn't match the regex."
```

```
def read_tree(output_file):
    global the_dir
    months_folders = get_subdirs(the_dir)
    for folder in months_folders:
        folder = the_dir + folder
        day_folders = get_subdirs(folder)
        for day_folder in day_folders:
            day_folder = folder + '/' + day_folder
            file_list = get_file_list(day_folder)
            for file in file_list:
                try:
                    file = day_folder + '/' + file
                    with open(file, 'r') as f:
                        this_file = f.readlines()
                        parse_and_write(this_file, output_file)
                except IOError:
                    print 'Error opening hoped for data-text file,', str(file), ', reason: ', IOError
# End of read_tree



def main():
    #open write file here
    output_file = open(write_data_filename, 'a')
    fieldnames = ('char, guild, timestamp, level, RaceInGame, classInGame\n')
    output_file.write(fieldnames)
    start_time = time.time()
    read_tree(output_file)
    #close write file here
    output_file.close()
    spent_time = time.time() - start_time

    mins_spent = int(spent_time / 60)

    secs_remainder = int(spent_time % 60)

    print 'Time of process: ', mins_spent, ':', secs_remainder


main()
```

Bibliography:

http://www.internetlivestats.com/internet-users-by-country
http://www.statista.com/statistics/234655/social-gamers-in-percent-of-internet-users-forecast/

http://www.statista.com/statistics/240987/global-online-games-revenue
http://www.statista.com/statistics/276601/number-of-world-of-warcraft-subscribers-by-quarter/

http://mmnet.iis.sinica.edu.tw/publication_detail.html?key=lee11_wowah, pages 6 and 8.

http://us.battle.net/wow/en/game/patch-notes/3-3-0

http://mmnet.iis.sinica.edu.tw/pub/lee11_wowah.pdf

http://mmnet.iis.sinica.edu.tw/publication_detail.html?key=tarng09_gamer_departure