2. Data Extraction, Transformation and Loading (ETL) process (James)

NSERC GPerf2 project

Exported on 03/31/2021

Table of Contents

1 Executive Summary

This paper will explain how to extract, transform, and load raw Plaid or Client Stats data in JSON format into cleaned CSV formatted data.

2 Acronyms

JSON: JavaScript Object Notation

CSV: Comma Separated Value

npm: Node Package Manager

3 Introduction

3.1 Project Background

This project aims to explain the Extract, Transform, and Data Loading process used to convert WTFast raw data dumps. This includes the scripts used to extract them and transform the data to conform to our needs. After this, we lay out how to upload the data to the database through the database web interface and SQL Developer.

3.2 Objectives

To outline the process for future use when starting with raw data from WTFast and the steps necessary to prepare data for analysis.

Methodology

The methodology used to perform ETL operations on our data was accomplished in this order.

1. Retrieve raw JSON data dumps from WTFast.
2. Clean and Merge the JSON files into a single CSV.
3. Upload monthly data into individual tables in the database.

3.2.1 WTFast Data Dumps

We were provided data from WTFast in the form of many JSON files representing PLAID data. They are provided in JIRA tickets during sprints, with the current PLAID data linked in 2.5 Data Collection.

3.2.2 Extract and Transform

Video Tutorial: GperfCleaner Tutorial[1]
Location of video on VM: Desktop/GPERFCleanerTutorial.mkv

Using the R programing language, we clean and merge the data into a single CSV file. The cleaning process takes all UNIX timestamps and converts them into human-readable timestamps in the format of "yyyy-MM-dd hh:mm:ss:SSS." The cleaning process also includes changing any strings that show up as "NULL," "NA," or "null" are replaced with empty attributes. After this, the data is exported into a CSV file.

See Bash GPerfCleaner in Appendix 2.1: Source Code(see page 59)

3.3 *VM Credentials:*

Username: COSC436St

Password: COSC436St2020

1 https://confluence2020.okanagan.bc.ca/download/attachments/16089972/2021-02-17%2011-04-45.mkv?api=v2&modificationDate=1613589634511&version=1

3.3.1 Usage

./gperfcleaner [-conv convert] [-pd plaiddata] [-cs clientstatsdata] [-P numberofprocesses] [-if inputfile] [-o outputFolder]
[-conv] # convert input folder from json to csv]
[-cs] # specify input data as client stats data to be cleaned]
[-pd] # specify input data as plaid data to be cleaned]
[-if] # specify single input file]
[-o] # output destination]
[-P] # specify number of processes]

3.3.2 Plaid Data Convert + Clean

```
./gperfcleaner -conv -pd -P 5 -o converted/
```

As you can see, we ran the gperfcleaner script with the option to convert plaid data, using 5 CPU processes, to the directory named 'converted'. If you are dealing with many small files, it is wise to give a larger process count (eg 50).

3.3.3 Client Stats Data Convert + Clean

```
./gperfcleaner -conv -cs -P 50 -o foo/
```

In the above instance, we are running the gperfcleaner to convert client stats data, using 50 CPU processes, to the directory 'foo'.

3.4 Implementation Tasks/Requirements

- *Note: If you are using the VM provided by this project, this process is already complete. No further installation or configuration is necessary.*

Windows Command-Line: On Windows, you need to tell the computer where to find them. `R.exe` and `RScript.exe` programs to execute—that is, what is the **path** to these programs. (***Required for this tool***), a better solution is to add the folder containing these programs to your computer's **variable**[2]. This is a *system-level* variable that contains a list of folders that the computer searches when finding programs to execute. The reason the computer knows where to find the `git.exe` program when you type `git` in the command-line is that that program is "on the `PATH`". In Windows, You can add the `R.exe` and `RScript.exe` programs to your computer's `PATH` by editing your machine's **Environment Variables** through the *Control Panel*:

- Open up the "Advanced" tab of the "System Properties." In Windows 10, you can find this by searching for "environment." Click on the "Environment Variables…" button to open the settings for the Environment Variables.

2 https://en.wikipedia.org/wiki/PATH_(variable)

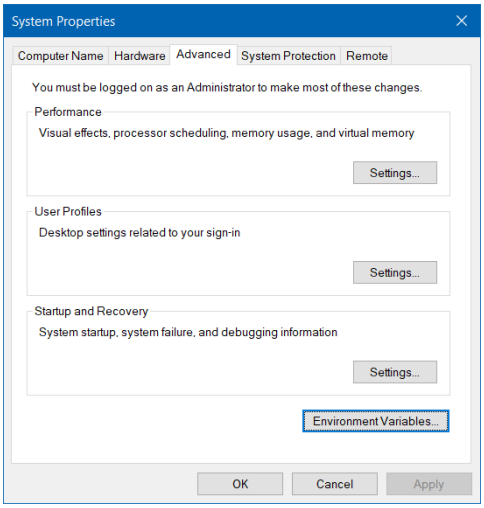


Figure 1.1 *Windows System Properties. From the advanced tab, click "Environment Variables"*

- In the window that pops up, select the “PATH” variable (either per-user or for the whole system) and click the “Edit” button below it.

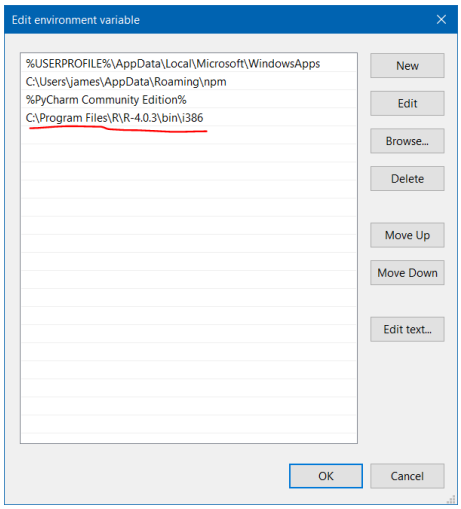


Figure 1.2 *Edit environment variable window. Above is a sample location for R*

- In the next window that pops up, click the “Browse” button to select the *folder that contains* the `R.exe` and `RScript.exe` files. See the above screenshot for one possible path.

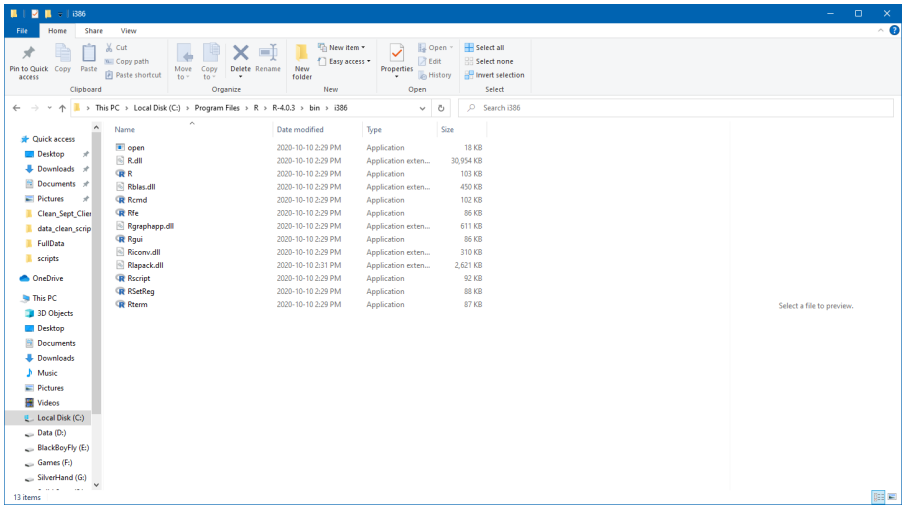


Figure 1.3 *Once R is installed, point your environment variable to its path*

- You will need to close and re-open your command-line (Git Bash) for the "PATH" changes to take effect.

3.4.1 jsonexport

```
npm install jsonexport
```

Figure 2 *Bash command to install the* jsonexport [3] *library*

If on Mac:

```
brew install gsed
```

Figure 2.1 *Bash command to install the* gsed[4] *library which is required for trimming the data*

3 https://www.npmjs.com/package/jsonexport
4 https://formulae.brew.sh/formula/gnu-sed

4 Design

4.1 How It Works

1. ***Read Arguments*** When the gperfcleaner command is received it first evaluates any arguments. If none are given, the usage instruction prompt is given. Otherwise, we continue to either the Plaid or Client stats process
2. ***Optimize JSON Data*** To improve conversion speeds, all white space and newline characters are removed from the files. Next, a newline character is added to only the end of whole JSON objects. *If Plaid data is selected, an additional step is taken to remove square bracket characters.*
3. ***Convert JSON to CSV*** This step uses the jsonexport npm library to convert the pretrimmed JSON data to CSV.
4. ***Clean Using client_Stats.R*** Runs the R script to clean client stats data.
5. ***Clean Using clientStatsAllData.R*** Runs the R script to clean the combined version of client stats data which introduces additional messy data.
6. **Clean Using monthPlaidData.R** Runs the R script to clean Plaid data.
7. ***Store Data in X Subdirectory*** Stores the cleaned data in a subdirectory.
8. ***Move All Processed Files To Destination Directory*** Moves the cleaned and original files to the desired output directory specified in the original command with the -o tag.

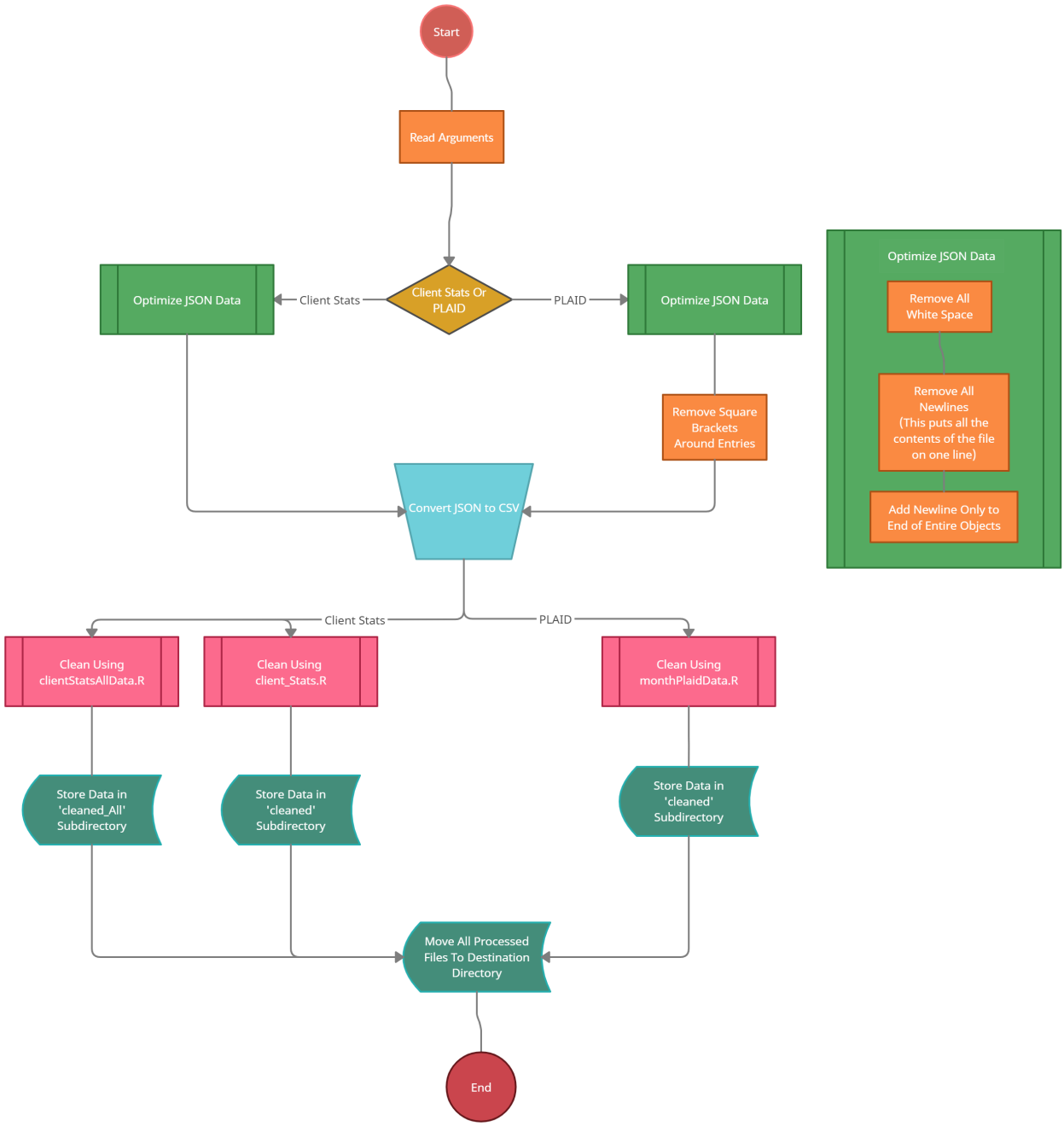


Figure 3 *Workflow of the GPerfCleaner Bash script*

5 Instruments/Tools

- R Studio [1(see page 5)]
- Bash Terminal [2]
- Jsonexport [3]

5.1 Description

This program is built to convert dynamically formatted Client Stats and Plaid data from raw JSON format into converted and cleaned CSV files. The main transformation done on the data is converting UNIX epoch timestamps into standard UTC format.

5.2 Data Collection

VMGuest files location: C://Users/cosc436st/Desktop/JamesBehnke/

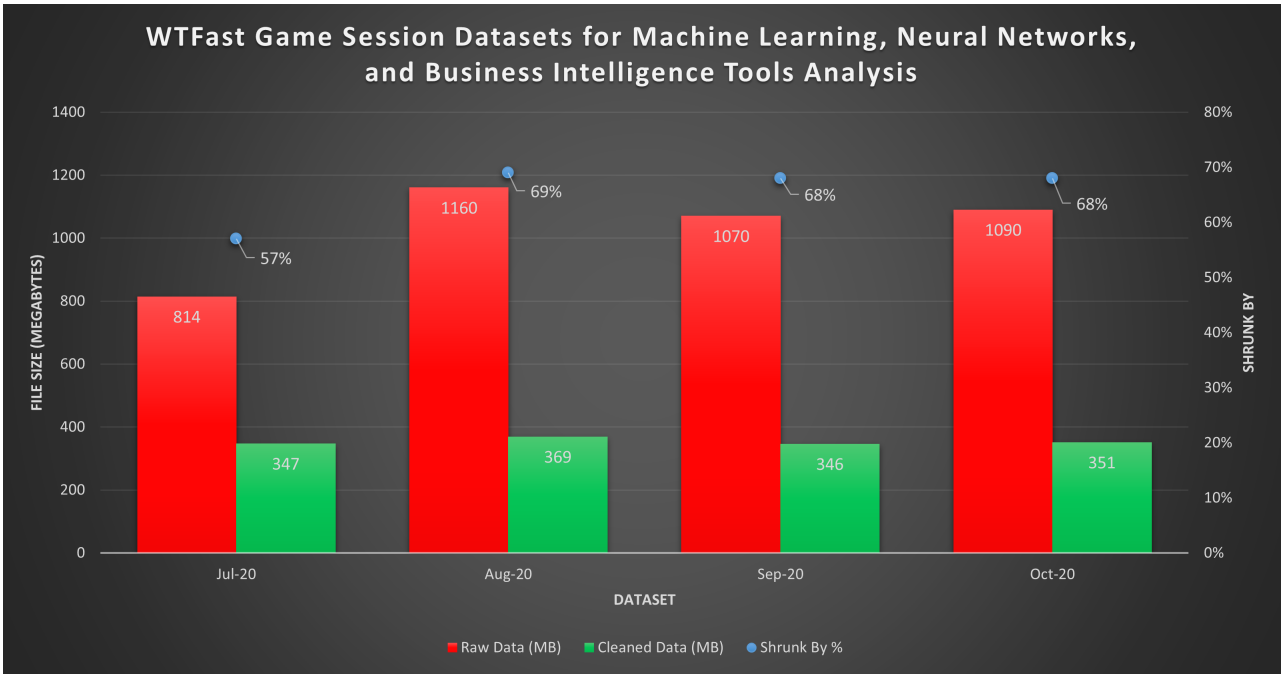
Note: Confluence file locations will be removed on March 31, 2021. After that please refer to the location on the VM.

JSON files Confluence	JSON files VMGuest location	No. of files	Rows of Data	Size (Before)	Size (After)	Shrink %	Speed (local machine)	Speed (Server)	Transformed CSV Files Confluence	VM Guest Location
July 2020 Plaid JSON Data	./JSON/PLAID/July/	1	369,479	814 MB	347 MB	57%	3.4MB/sec	8MB/sec	July 2020 Plaid CSV Data	./CleanedCSV/PLAID/July/
August 2020 Plaid Data	./JSON/PLAID/August/	61	357,151	1.16 GB	369 MB	69%	10MB/sec	19MB/s	August 2020 Plaid CSV Data	./CleanedCSV/PLAID/August/
September 2020 Plaid Data	./JSON/PLAID/September/	61	330,304	1.07 GB	346 MB	68%	7.9MB/sec	16MB/s	September 2020 Plaid CSV Data	./CleanedCSV/PLAID/September/
September 2020 Client Stats Data	./JSON/ClientStats/September/	43,200	3,659,973	54.2 GB	2.13 GB	96%	24MB/min	95MB/min	September 2020 Client Stats CSV	./CleanedCSV/ClientStats/September/
October 2020 Plaid Data	./JSON/PLAID/October/	62	339,534	1.09 GB	351 MB	68%	10MB/sec	20MB/sec	October 2020 Plaid CSV Data	./CleanedCSV/PLAID/October/
October 2020 Client Stats Data	./JSON/ClientStats/October/	44,638	3,557,641	55.2 GB	2.16 GB	96%	25MB/min	95MB/min	October 2020 Client Stats CSV Data	./CleanedCSV/ClientStats/October/

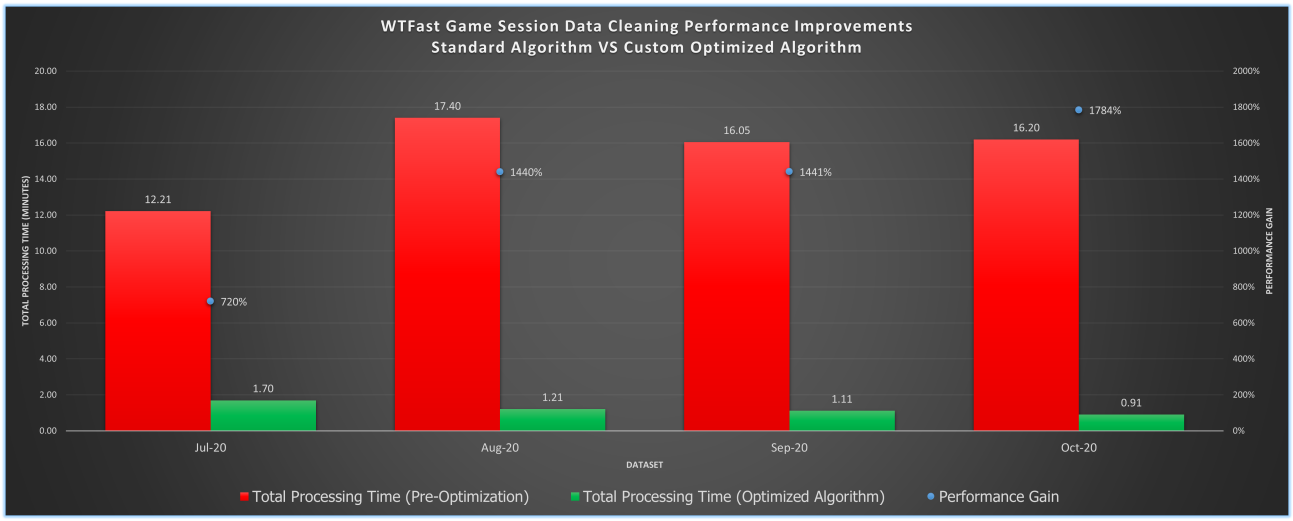
Figure 4 *Performance data of GPerfCleaner script over the different datasets provided*

5.3 Performance Results

PLAID Data:

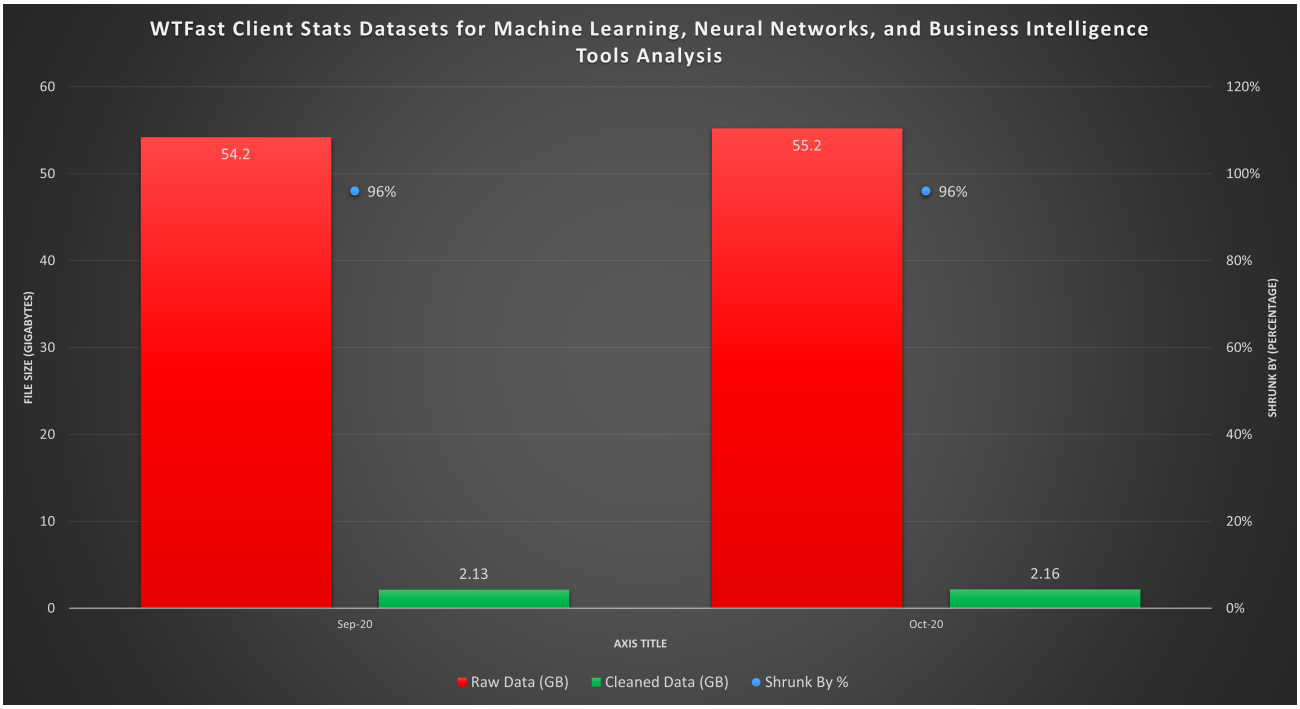


GPerfCleaner Performance Chart showing the file size differences between unzipped raw PLAID datasets and their cleaned equivalents

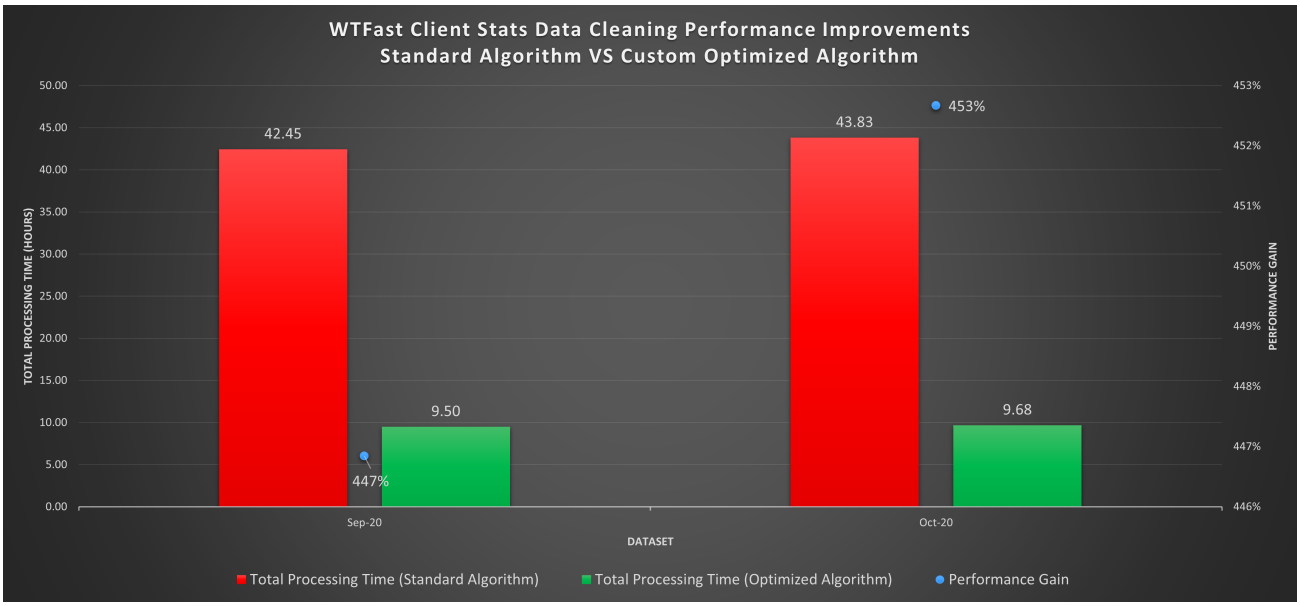


GPerfCleaner Performance Chart showing the algorithm performance differences of PLAID data using the standard JSON To CSV algorithm versus the custom algorithm I've built especially for WTFast.

Client Stats:



GPerfCleaner Performance Chart showing the file size differences between unzipped raw Client Stats datasets and their cleaned equivalents



GPerfCleaner Performance Chart showing the algorithm performance differences of Client Stats data using the standard JSON To CSV algorithm versus the custom algorithm I've built especially for WTFast.

5.4 Limitations

Data must not contain spelling errors in column names. Since the cleaning scripts clean by column name, they must be consistently named.

Data must be given proper parameters when calling the gperfcleaner script, e.g. Plaid data must be given the -pd parameter, client stats must be given -cs.

6 Results

6.1 Deliverable 1 (Extraction)

6.1.1 Plaid Data:

1. Finds all files in the current directory ending in .json
2. Makes a copy of the original JSON files and moves them to a folder named 'originalFiles.'
3. Remove all whitespace and newlines.
4. Add newlines only to the end of whole entries.
5. Remove all square brackets from entries and sub-entries.
6. Add square brackets only to the beginning and end of entire entries.
7. Move the minified file to the 'minified' subfolder.
8. Remove original file.

```
find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh -c 'cp '{}' ./
originalFiles/'{}'; sed -i -r "s/(\".*\")|\s*/\1/g" '{}'; cat '{}' | tr -d "\n" > ./minified/'{}'; sed -i
"s/}},/}},\n/g;s/.*\[{/{/;s/\]}}//g" ./minified/'{}'; rm '{}' '
```

Figure 5 *Chunk of the GPerfCleaner script responsible for trimming Plaid data*

6.1.2 Client Stats:

1. Finds all files in the current directory ending in .json
2. Makes a copy of the original JSON files and moves them to a folder named 'originalFiles.'
3. Remove all whitespace and newlines.
4. Add newlines only to the end of whole entries.
5. Move the minified file to the 'minified' subfolder.
6. Remove original file.

```
find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh -c 'cp '{}' ./
originalFiles/'{}'; sed -i -r "s/(\".*\")|\s*/\1/g" '{}'; cat '{}' | tr -d "\n" > ./minified/'{}'; sed -i
"s/}},/}},\n/g" ./minified/'{}'; rm '{}''
```

Figure 5.1 *Chunk of the GperfCleaner script responsible for trimming Client Stats data*

6.1.3 Conversion

1. Find all .json files within script-created 'minified' subfolder.
2. Use npm package jsonexport to convert files from JSON to CSV.
3. Send confirmation to the output of the completed CSV file.

```
find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh -c 'jsonexport
'{}' ./client_stats_converted/'{}.csv'; echo '{}''
```

Figure 5.2 *Chunk of the GperfCleaner script responsible for converting from JSON to CSV*

6.2 Deliverable 2 (Transformation)

6.2.1 Plaid Data:

This script is responsible for cleaning any messy data that was added during conversion and converting UNIX timestamps into human-readable UTC formatted timestamps.

See Appendix 2.2 Plaid Data Transformation[5]

6.2.2 Client Stats Transformation:

This script is responsible for cleaning any messy data that was added during conversion removing redundant rows of NA data.

See Appendix 2.3 Client Stats Transformation[6]

6.2.3 Combine X files into separate subdirectories:

This script is useful for client stats data dumps as there can be many tens of thousands of files to be processed, and inserting these into a database can be problematic and time-consuming. This script will divide them into folders of 2500 files. From there, files can be easily and efficiently combined into a single file using the next tool.

See Appendix 2.4 Combine X Files to Separate Subdirectories[7]

6.2.4 Combine all Files in a Directory:

If using this tool for Client stats, it is strongly suggested to run the above tool first as most computers won't want to handle 2GB+ CSV files. If more than ~750MB of files are combined into one, it will cause difficulty to open to view in Excel or other programs.

See Appendix 2.5 Combine All Files in a Directory[8]

5 https://confluence2020.okanagan.bc.ca/display/NGP/Appendix+4%3A+Source+Code#Appendix4:SourceCode-4.2PlaidDataTransformation
6 https://confluence2020.okanagan.bc.ca/display/NGP/Appendix+4%3A+Source+Code#Appendix4:SourceCode-4.3ClientStatsTransformation:
7 https://confluence2020.okanagan.bc.ca/display/NGP/Appendix+4%3A+Source+Code#Appendix4:SourceCode-4.4CombineXFilestoSeparateSubdirectories
8 https://confluence2020.okanagan.bc.ca/display/NGP/Appendix+4%3A+Source+Code#Appendix4:SourceCode-4.5CombineAllFilesinaDirectory

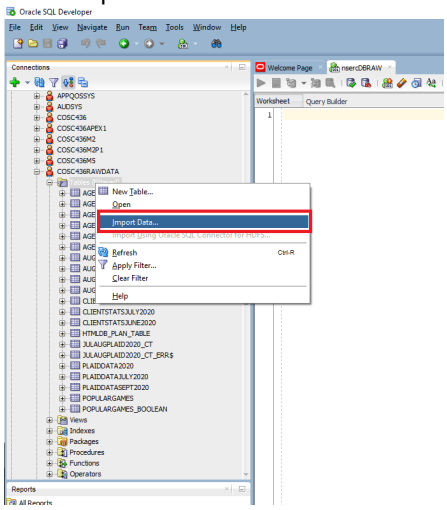
6.3 Deliverable 3 (Loading)

Using the cleaned and transformed CSV provided, we then upload this to the database. Using the credentials provided (Appendix 8(see page 0)) for the COSC436RAWDATA workspace, we login to the database web interface to upload the data. This is done in a few simple steps outlined below. This step can also be accomplished using SQL Developer.

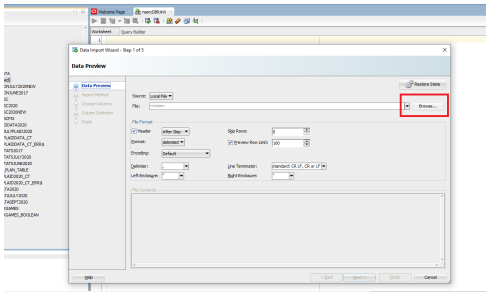
6.3.1 **Using SQLDeveloper**

Using SQLDeveloper

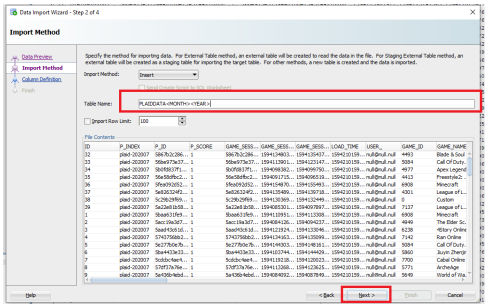
1. Login to COSC436RAWDATA schema
2. Right-click the tables folder and select Import data



3. Browse to the location of the CSV file

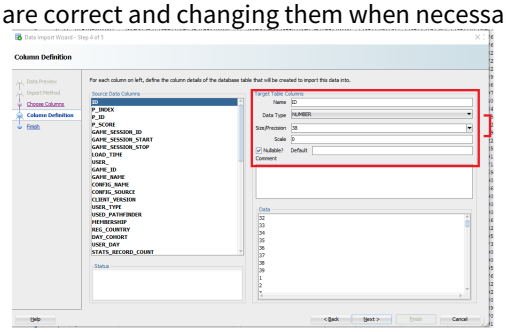


4. After data appears in the bottom table, click next and set the table name with the naming convention of **PLAIDDATA<MONTH><YEAR>** and then hit next.

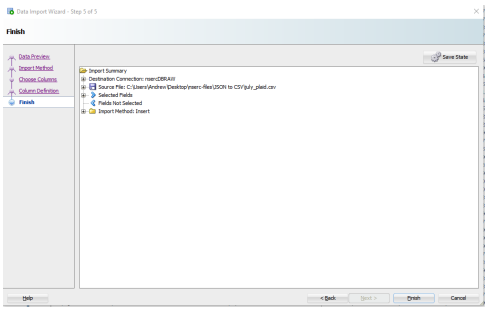


5. On the next screen, you can select which attributes you want to import. If you do not want all of them, you can remove them here. Importing all the data is generally what is wanted, so click next.

6. On this screen, we confirm the data types of the attributes to be imported. This has to be gone through one by one to make sure they are correct and changing them when necessary.



7. The final screen before uploading is to confirm all the steps taken. Click finish to upload the data. If the data is properly cleaned, it will insert without any issues.



7 Discussion + Recommendations

When running the script from Bash, during the cleaning phase, R will give you a warning error that "NAs were introduced during coercion."

You can safely ignore this warning, as later in the same R script, NA's are accounted for and discarded.

7.1 ETL Process Proposed Future Work

- Add feature to auto combine data (PLAID or Client Stats)
- Add ability to auto-upload to server.
- Add feature to auto-detect if the folder should be divided into smaller subdirectories.
- Create GUI for script

8 References

[1] R Installation. If you are only using R for this script, you can download and install only the lightweight compiler — https://mirror.rcg.sfu.ca/mirror/CRAN/

[2] Git Bash Installation. Git Bash is required on Windows machines but comes standard on Unix. The commands in these scripts must run from a bash shell — https://git-scm.com/downloads

[3] Jsonexport Installation. An npm module is called during the bash script. (Required) — https://www.npmjs.com/package/jsonexport

9 Appendixes

9.1 Appendix 1: ETL Transformation Requirements[9]

9.2 Appendix 2: Source Code

9.3 Appendix 3: Old data and documents location:

- Bitbucket: https://bitbucket2020.okanagan.bc.ca/projects/GPERF2/repos/nsercgperf2-2019/browse/RGraphing/JamesBehnke?at=refs%2Fheads%2FGPERF2Development
- ETL process using Pentaho: 1. (Jack) Business Intelligence & Visualization (DBMS/DW/BIs)#1.(Jack)BusinessIntelligence&Visualization(DBMS/DW/BIs)-Transformationsandaccommodatingdata[10]

9.4 Appendix 6: Acceptance Test

Not yet created.

9.5 Appendix 7: Scripts

BitBucket Location: nsercgperf2-2019/RGraphing/JamesBehnke/data_clean_script/
VM Location: Desktop/data_cleaning_script/

9 https://confluence2020.okanagan.bc.ca/display/NGP/%28Merged%29+ETL+Requirements

10 https://confluence2020.okanagan.bc.ca/pages/viewpage.action?pageId=16090397#id-1.(Jack)BusinessIntelligence&Visualization(DBMS/DW/BIs)-1.(Jack)BusinessIntelligence&Visualization(DBMS/DW/BIs)-Transformationsandaccommodatingdata

10 Appendix 1: JulyPlaid Data ETL Transformation (Luis)

10.1 Data Transformation Background

For modelling, Albert Wong made different requests on data transformations. Transformations consisted of changing variables to numeric (boolean), splitting, extracting (ie. extract the day of the week from a date), categorizing, and adding more variables to the table.

The purpose of the transformations is to utilize Regression and Machine learning modelling for predictions on WTF ping.

10.2 Data Extraction and Filter

Some records from the July data have WTFast ping values of 0ms, or ping values low enough that the accuracy of them is unreliable. As a result, we filtered records with less than 15ms from the JulyPlaidData. Besides zero values, there were values with ping exceeding 250ms that end up being unplayable game sessions. Records with ping values over 250 were filtered and dismissed too for this reason.

10.2.1 Filter Query

This is the data set the research team was working with for analysis of the July Plaid Data 2020. This query can be run in Apex or SQL to filter the data:

```
select * from cosc436td.PLAIDDATAJULY2020 LEFT JOIN cosc436td.POPULARGAMES on
PLAIDDATAJULY2020.GAME_ID = POPULARGAMES.X_SOURCE_GAME_ID WHERE PLAIDDATAJULY2020.USED_
PATHFINDER = 1 AND PLAIDDATAJULY2020.WTFAST_PING IS NOT NULL AND PLAIDDATAJULY2020.WTFAST
_PING BETWEEN 15 AND 250;
```

Output file: export.xlsx

The resulting dataset was derived from a left join of POPULARGAMES table and the PLAIDDATAJULY2020 table:

Query

```
select * from PLAIDDATAJULY2020 LEFT JOIN POPULARGAMES on PLAIDDATAJULY2020.GAME_ID =
POPULARGAMES.X_SOURCE_GAME_ID;
```

PLAIDDATAJULY2020:

- Number of records: 340499
- POPULARGAMES: 190

10.3 Data Transformation Request August 19, 2020

Modelling – Regression and Machine Learning

Using July 2020 Data

1. Create a file for modelling by deleting the 28974 records that have “N/A” in WTFast_Ping from the July Plaid CSV file (369478 records). The remaining file should have 340504 records.
2. Create the following indicator (dummy) variables:

Weekend = 0 if Weekday_Session_Start if not Friday, Saturday, and Sunday

Weekend = 1 if Weekday_Session_Start if it is Friday, Saturday, or Sunday

Action = 1 if RAW01 = “Action”

Action = 0 if RAW01 is not “Action”

Casual = 1 if RAW01 = “Casual”

Casual = 0 if RAW01 is not “Casual”

MassMulti = 1 if RAW01 = “Massively Multiplayer”

MassMulti = 0 if RAW01 is not “Massively Multiplayer”

AllOther = 1 if RAW01 is not “Action” or “Casual” or “Massively Multiplayer”

AllOther = 0 if RAW01 is “Action” or “Casual” or “Massively Multiplayer”

RAW01	**Action**	**Casual**	**MassMulti**	**AllOther**
Action	1	0	0	0
Casual	0	1	0	0
Massively Multiplayer	0	0	1	0
RPG	0	0	0	1
Sports	0	0	0	1
Strategy	0	0	0	1
N/A	0	0	0	1

Table 1. Dummy Variable creation for Game Genre [1]

3. Randomly select 60% of the records from the modelling file to create the training data set. We can either use an R package or create a random number (=rand() function in Excel) and then sort it to get the top 60%. The testing file should have 204302 records. The remaining 40% will form the testing data set with 136202 records.

4. Use the training data set and the variables in the table to run the following regression models:

10.3.1 **Model 1**: Multiple Regression Model

We first ran a Multiple Linear Regression Model [2] using R Studio and for the model, these were the variables that needed to be normalized.

Use wtfast_ping as the dependent variable as the following as independent variables.

Variable	**Type**	**Pre-Processing**
Weekend	I	
Action	I	
Casual	I	
MasMulti	I	
AllOther	I	
bytes_down_tcp	N	Normalized (*)
bytes_down_udp	N	Normalized (*)
bytes_per_second	N	Normalized (*)
bytes_up_tcp	N	Normalized (*)
bytes_up_udp	N	Normalized (*)
client_ip_count	N	Normalized (*)
distance	N	Normalized (*)
game_ip_count	N	Normalized (*)
internet_flux	N	Normalized (*)

internet_loss	N	Normalized (*)
internet_ping	N	Normalized (*)
internet_spke	N	Normalized (*)
socket_count_tcp	N	Normalized (*)
socket_count_udp	N	Normalized (*)

Table 2. Variables normalized for MLR

(*) normalized as follows:

```
Normalized x = (x-Min)/(Max-Min) [3]
```

Where Min (minimum value) and Max (maximum value) of the data set.

Use the resulting regression model to predict WTFast_Ping in the testing data set. If possible, produce RMSE (root mean square error) and MAPE (mean absolute percent error) as performance indicators.

10.3.2 **Model 2**: Polynomial Regression Model

For the Polynomial Regression Model [4], variables needed to be normalized as well before running the model.

Use wtfast_ping as the dependent variables and the following as the independent variables

Variable	Type	Pre-Processing
Weekend	I	
Action	I	
Casual	I	
MasMulti	I	
AllOther	I	
bytes_down_tcp	N	Normalized (*)

bytes_down_udp	N	Normalized (*)
bytes_per_second	N	Normalized (*)
bytes_up_tcp	N	Normalized (*)
bytes_up_udp	N	Normalized (*)
client_ip_count	N	Normalized (*)
distance	N	Normalized (*)
SqDist	**N**	**square of the normalized distance**
game_ip_count	N	Normalized (*)
internet_flux	N	Normalized (*)
internet_loss	N	Normalized (*)
internet_ping	N	Normalized (*)
internet_spke	N	Normalized (*)
socket_count_tcp	N	Normalized (*)
socket_count_udp	N	Normalized (*)

Table 3 - Variables Normalized for Polynomial Regression

Use the resulting regression model to predict WTFast_Ping in the testing data set. If possible, produce RMSE (root mean square error) and MAPE (mean absolute percent error) as performance indicators.

10.3.3 **Other Models**

Once the model was set, the modelling plan was the following:

- Use wtfast_ping as the dependent variables and the variable (feature) list in Model 2 as independent variables.
- Use the resulting regression model to predict WTFast_Ping in the testing data set. If possible, produce RMSE (root mean square error) and MAPE (mean absolute percent error) as performance indicators.
- Run the following machine learning models using the feature (variable) list in Model 2:
    - MLP with three hidden layers
    - Random Forest
    - Support Vector Regression

10.4 Other Data Transformation Requests

After the first variable engineering, besides working with variables, a merge problem was solved using Excel. On 09/27/20 a review and new items for the plan were set on data manipulations to follow:

ITEM
Fix problems (if any) with the transformations.
Produce descriptive statistics (5 number summary plus other percentiles) for the data set after the problems are fixed
Send statistics and file
Delete those records where Duration > 43200 to create the analysis data set
Produce descriptive statistics (5 number summary plus other percentiles) for the analysis data set
Send statistics and the analysis file
Perform the square root transformation on the following variables: (e.g., INTERNET_LOSS = Square root(INTERNET_LOSS) INTERNET_LOSS; INTERNET_SPKE; SOCKET_COUNT_TCP; SOCKET_COUNT_UDP; GAME_IP_COUNT; BYTES_PER_SECOND
Normalize the independent numerical variables except Duration
Create a categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM + STRATEGY + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY
Create three regression models (as completed in the past) using WTFAST_PING as the dependent variable and using RMSE and MAPE as performance indicators
Send regression results including the variable list and RMSE/MAPE)
Create three Random Forrest Models using WTFAST_PING as the dependent variable and using RMSE and MAPE as performance indicators (the options are: ntree=100, ntree=250, ntree=500, the default)
Send Random Forest results including the plot output and RMSE/MAPE

Excel file for new plan's items: Action items Sep 27.xlsx

10.5 Data Purposes

For modelling, variable transformations were requested (Data Transformation Request(see page 26)) to run again Regression and Machine Learning models to predict ping. These transformations were performed all using R Studio.

The considered Machine learning models that were considered are:

- Regression
- Random forest
- Multiple-Layer perceptron/Neural Network
- Support Vector Regression

10.5.1 Data Description

The raw data is from JulyPlaidData2020. It was filtered (Data Extraction(see page 26)) to get records with WTFast ping between 15 and 250.

Columns: 68

ID, P_INDEX, P_ID, P_SCORE, GAME_SESSION_START, GAME_SESSION_STOP, LOAD_TIME, GAME_ID, GAME_NAME, CONFIG_NAME, CONFIG_SOURCE, USER_TYPE, USED_PATHFINDER, MEMBERSHIP, USER_DAY, STATS_START, INTERNET_PING, INTERNET_FLUX, INTERNET_LOSS, INTERNET_SPKE, WTFAST_PING, WTFAST_FLUX, WTFAST_LOSS, WTFAST_SPKE, PING_Q, FLUX_Q, LOSS_Q, SPKE_Q, CHAIN_COUNT, BYTES_UP_TCP, BYTES_UP_UDP, BYTES_UP, BYTES_DOWN_TCP, BYTES_DOWN_UDP, BYTES_TCP, BYTES_UDP, BYTES_TOTAL, SOCKET_COUNT, SOCKET_COUNT_TCP, SOCKET_COUNT_UDP, CLIENT_IP, CLIENT_IP_COUNT, CLIENT_IP_STR, CLIENT_GEO, NODE_COUNT, NODE_IP_STR, NODE_CONTINENT, NODE_COUNTRY, NODE_REGION, NODE_COUNTRY_REGION_CITY, NODE_ISP, NODE_ORG, GAME_IP_COUNT, GAME_IP_STR, GAME_GEO, BYTES_PER_SECOND, DURATION, DAY_COHORT, ID_1, X_SOURCE_GAME_ID, CONFIG_NAME_1, MYGENRE, RAWG1, RAWG2, RAWG3, RAWG4, RAWG5, RAWG6

Observations: 64,000

10.5.2 Data Transformation

Data used to do transformations and variable creation

Input data for transformation: julyplaiddata: export_final.csv

Other files used for variable creation:

genre_merge.csv (file created on excel using lookup function to match genres from Data_PopGames.csv)

Data_PopGames.csv (file created by Gaetan to extract genres from games on julyplaid data)

Pop_Games.csv (File with genres to merge with export_final.csv, this file was created using vlookup function in Excel)

Node_Geo.csv (file with node geo latitude and longitude to calculate distance client-node-game server)

10.5.2.1 R Studio Packages

"R is a language and environment for statistical computing and graphics. It is a GNU project[11] which is similar to the S language and environment which was developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. R can be considered as a different implementation of S. There are some important differences, but much code written for S runs unaltered under R." [5]

10.5.2.2 R Studio Packages

These packages should be installed in advance to transform data:

- **tidyverse:** "A set of tools that solves a common set of problems: you need to break a big problem down into manageable pieces, operate on each piece and then put all the pieces back together." [6]
- **splitstackshape**: "The concat.split (cSplit) family of functions splits such data into separate cells." [7]
- **rgeolocate**: "IP geolocation is a powerful tool to have if you're dealing with web data, and there are a couple of R packages that provide access to specific services, such as the legacy rgeoip package or Bob Rudis's ipapi. They're all spread about and have diffing interfaces, styles and requirements." [8]
- **lubridate**: "Regularisation, decomposition and analysis of space-time series. The pastecs R package is a PNEC-Art4 and IFREMER (Benoit Beliaeff <Benoit.Beliaeff@ifremer.fr[12]>) initiative to bring PASSTEC 2000 functionalities to R." [9]
- **geosphere**: "Spherical trigonometry for geographic applications. That is, compute distances and related measures for angular (longitude/latitude) locations." [10]
- **rgl**: "Provides medium to high level functions for 3D interactive graphics, including functions modelled on base graphics (plot3d(), etc.) as well as functions for constructing representations of geometric objects (cube3d(), etc.)." [11]
- **dplyr**: "dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges" [12]

10.6 Transformation Steps

10.6.1 Step 1

10.6.1.1 Transformations

1. Merge Genre
2. Merge NODE_GEO (latitude, longitude)
3. Split GAME_SESSION_START timestamp to get WEEKEND variable (1= weekend, 0=weekday)
4. Split of CLIENTE_GEO, NODE_GEO, and GAME_GEO to calculate distance
5. New variables based on Genres: ACTION, ACT_ADV, MASS_MULTY, ACTIONRPG, ACT_RPG_MM, ACT_RPG_STR, ACT-RPG_STR_MM_ADV, ACT_SHOOTER, ACT_STR_SIM, RPG, RPGCASUAL, RPG_MM, SPORTS, STRATEGY, and OTHER.
6. Deleted all the variables that are not used on models
7. Scaled variables: BYTES_DOWN_TCP/1000000, BYTES_UP_TCP/1000000, BYTES_DOWN_UDP/1000000, BYTES_UP_UDP/1000000, TOTAL_DISTANCE/1000, BYTES_PER_SECOND/1000, and TOTAL_DISTANCE^2
8. New variables: BYTES_UP_TCP/DURATION, BYTES_DOWN_TCP/DURATION, BYTES_UP_UDP/DURATION, BYTES_DOWN_UDP/DURATION

11 http://www.gnu.org/
12 http://ifremer.fr

10.6.1.2 Deleted Records

1. After we found the issue with the merge() function, to add Genre to each record and the Node latitude and longitude we:

1. Created a vector using Excel with the Genre of each observation using “vlookup”.
2. Merged Genre vector and NODE_GEO to the dataset Changing the approach for these two issues, we avoided getting duplicate records due to the merge() error.

1. Deleted all duplicate records from the data using dstinct().
    a. 61,083 records
2. Deleted all records with GAME_ID=0

1. 57,066 records

1. Deleted records with “0” on CLIENT_LAT, GAME_LAT, and NODE_LAT. Without deleting these records, it is not possible to calculate the distance for the rest of the data.

1. 55,559 records

1. After calculating BYTES_DOWN_TCP / DURATION, we got some inf and NA values and were deleted

1. 55,517 records

1. After all transformations, all DURATION=0 values were deleted.

Dataset after transformations: 55,517

10.6.1.3 Output CSV file

Data_PopGames_Oct15.csv

10.6.2 Step 2

10.6.2.1 Deleted Records

A) Deleted all records with DURATION<=43,200
a. 52,635 records

Dataset after deleted records: 52,635

10.6.2.2 Output CSV FILE

Data_Duration_Oct015.csv

10.6.3 STEP 3 (Normalized and new variables for ML modelling CSV File)

10.6.3.1 Transformations and Normalization of Data

The square root of INTERNET_LOSS, INTERNET_SPKE, SOCKET_COUNT_TCP, SOCKET_COUNT_UDP, GAME_IP_COUNT, BYTES_PER_SECOND

Create a categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM + STRATEGY + RPG + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY

All boolean variables were converted into factors to avoid any errors while running any ML model.

Deleted variables: GAME_ID, GAME_NAME, CONFIG_NAME, ACT_RPG_STR_MM_ADV, SPORTS, ACT_RPG_MM, STRATEGY, RPG, WTFAST_LOSS, WTFAST_SPKE, WTFAST_FLUX, DURATION, DUR_BD_UDP, DUR_BU_UDP, DUR_BD_TCP, DUR_BU_TCP, DISTANCE_2, OTHER, ACTIONRPG, ACT_STR_SIM, ACT_ADV, and ACT_MASS_MULTY.

Normalization of data, except for dummy variables (using z-score normalization).

10.6.4 R Studio Scripts

Several scripts were used. The reason is that at first the "merge" function was duplicating records to the dataset and different data requests were made to fit the best regression and ML models in the future. Scripts are included by dates.

Note: On scripts using the 'merge' function we found an error, where observations get duplicated. We used the following file to merge:

Pop_Games.csv

Files location: Files Table(see page 37)

On September 22 duplicated records were deleted.

10.6.4.1 **September 19 and 22 Script (**Source Code 1**)**

Output files:

Norm_Data_Sep17.csv

Data_PopGames_Sep17.csv

Data_PopGames_Sep22.csv

Data_PopGames_Sep22.csv

10.6.4.2 **September 29 Script (**Source Code 2**)**

Output files

Data_PopGames_Sep29.csv

Data_Duration_Sep29.csv

Norm_Data_Sep29.csv

Files location: Files Table(see page 37)

10.6.4.3 **October 5 Script (**Source Code 3**)**

Note: Stopped using merge function and the column was merged after performing a vlookup function in Excel

genre_merge.csv: File where vlookup function was used (also for October 15 script)

Output files:

Data_PopGames_Oct05.csv

Data_Duration_Oct05.csv

Norm_Data_Oct05.csv

Files location: Files Table(see page 37)

10.6.4.4 **October 15 Script (**Source Code 4**)**

Note: Stopped using merge function and the column was merged after performing a vlookup function in Excel.

genre_merge.csv File where vlookup function was used (also for October 15 script).

Output files:

Data_PopGames_Oct15.csv

Data_Duration_Oct015.csv

Norm_Data_Oct15.csv

Files location: Files Table(see page 37)

10.7 Conclusion

This first approach considered JulyPlaid Data. This data set was provided by WTFast, real data to be analyzed and begging research with the foundation of ping prediction.

In order to have the correct data for analysis and prediction using Linear Regression, Multiple Layer Perceptron, Support Vector Regression, and Random Forest, data needed to be analyzed and manipulated. The previous step to analyzing data was the creation of a proper data set with the new variables, the result of the data already provided by WTFast, but also the creation of other like genres.

Not all the observations were considered from the beginning, since ping values of zero and everything above 250 could represent errors or long client sessions not from gaming. To analyze dates in which gamers play the most, transfer of data, distance, ping differences between Intenet and WTFast, geographical location, among others, all variables were modified to find the best subset. Having that in mind, all changes were done as requested, descriptive statistics were delivered for each to have a deeper look at trends and variance to do necessary changes until we finished with a definite dataset.

This dataset was used on different models as mentioned to predict WTFast ping and its performance. Further researches led to Game Satisfaction, and Geographical analysis (by Country and Continent).

10.8 References

[1] "The many different types of video games & their subgenre". [Online]. Available: https://www.idtech.com/blog/different-types-of-video-game-genres. [Accessed: 04-Mar-2021].

[2] "Multiple Linear Regression Model" [Online]. Available: https://www.investopedia.com/terms/m/mlr.asp. [Accessed: 04-Mar-2021].

[3] "Normalization. [Online]. Available: https://www.codecademy.com/articles/normalization. [Accessed: 04-Mar-2021].

[4] "Fitting a Polynomial Regresion using R". [Online]. Available: https://datascienceplus.com/fitting-polynomial-regression-r/. [Accessed: 05-Mar-2021].

[5] "What is R?". [Online]. Available: https://www.r-project.org/about.html. [Accessed: 05-Mar-2021].

[6] "Tidyverse". [Online]. Available: https://www.tidyverse.org/. [Accessed: 05-Mar-2021].

[7] "splitstackshape". [Online]. Available: https://www.r-project.org/nosvn/pandoc/splitstackshape.html. [Accessed: 05-Mar-2021].

[8] "rgeolocate: IP Address Geolocation". [Online]. Available: https://cran.r-project.org/web/packages/rgeolocate/index.html. [Accessed: 05-Mar-2021].

[9] "lubridate, part of tidyverse". [Online]. Available: https://lubridate.tidyverse.org/. [Accessed: 05-Mar-2021].

[10] "geosphere, Spherical Trigonometry". [Online]. Available: https://cran.r-project.org/web/packages/geosphere/index.html. [Accessed: 05-Mar-2021].

[11] "rgl, 3d Visualization Using OpenGL". [Online]. Available: https://cran.r-project.org/web/packages/rgl/index.html. [Accessed: 05-Mar-2021].

[12] "dplyr". [Online]. Available: https://www.rdocumentation.org/packages/dplyr/versions/0.7.8. [Accessed: 05-Mar-2021].

10.9 Appendix 1.1 Files Table

CSV FILES		
FILENAME ON CONFLUENCE	**No. of files**	**Virtual Machine Location**
export.xlsx	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
export_final.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
genre_merge.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Pop_Games.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Node_Geo.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames_Oct05.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_Duration_Oct05.csv	1	C:\Users\Edward\Documents\WTFast Backup files\Appendix 1 ETL\Input and Output CSV Files

CSV FILES		
FILENAME ON CONFLUENCE	**No. of files**	**Virtual Machine Location**
Norm_Data_Oct05.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames_Oct15.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_Duration_Oct015.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Norm_Data_Oct15.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Pop_Games.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Norm_Data_Sep17.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames_Sep17.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames_Sep22.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Norm_Data_Sep22.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_PopGames_Sep29.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Data_Duration_Sep29.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
Norm_Data_Sep29.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
genre_merge.csv	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Input and Output CSV Files
R STUDIO FILES		
FILENAME ON CONFLUENCE	**No. of files**	**Virtual Machine Location**

CSV FILES		
FILENAME ON CONFLUENCE	**No. of files**	**Virtual Machine Location**
Data_Transform_Duration_S19.R	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Scripts
Data_Transform_Duration_S19_October5.R	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Scripts
Data_Transform_Duration_S19_October15.R	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Scripts
Data_Transform_Duration_S19_SEP29.R	1	C:\Users\Edward\Documents\WTFast Backup files\A. Section 2 Appendix 1 ETL\Scripts

Table 1 - CSV files and R scripts used for transformations on this document

10.10 Appendix 1.2 Source Codes

10.10.1 Source Code 1

Data_Transform_Duration_S19.R

```
# Copyright (c) 2021 MIT License
# GPERF2 Project
# Okanagan College
#
# Object: Feature Engineering
# Version 1.0
# Author: Luis Zeuqueiro
# Creation date: 19/09/2020
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
```

```
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
# HISTORY:
# Date By Comments
# ---------- ----------- ----------------------------------------------------------
# 19/09/2020 - Luis Zequeiro - Initial Creation of version 1.0

library(tidyverse)
library(splitstackshape)
library(rgeolocate)
library(lubridate)
library(geosphere)
library(rgl)
library(dplyr)

# Set a working directory
setwd("C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and Output CSV
Files")

# Load the data
data <- read.csv("export_final.csv")
genres <- read.csv("genre_merge.csv")
names(genres)[1] <- "GENRE"
names(data)[1] <- "ID"
# data <- data[-]

NODE_GEO <- read.csv("Node_Geo.csv")
data <- cbind(data, genres)
data <- cbind(data, NODE_GEO)


# Split timestamp and get day of the week and make it boolean

data <- cSplit(data, "GAME_SESSION_START", " ")

data <- dplyr::rename(data,
GAME_SESSION_START_DATE = GAME_SESSION_START_1,
GAME_SESSION_START_TIME = GAME_SESSION_START_2,)

data$weekend <- weekdays(as.Date(data$GAME_SESSION_START_DATE))


data <- data %>% mutate(WEEKEND = recode(weekend, "Monday" = 0, "Tuesday" = 0,
"Wednesday" = 0, "Thursday" = 0, "Friday" = 1, "Saturday" = 1, "Sunday" = 1))

# Calculate distance using Client Geo and Game Geo

#CLIENT_GEO
data <- cSplit(data, "CLIENT_GEO", ",")

data <- dplyr::rename(data,
CLIENT_LAT = CLIENT_GEO_1,
CLIENT_LONG = CLIENT_GEO_2)

#GAME_GEO
data <- cSplit(data, "GAME_GEO", ",")
```

```
data <- dplyr::rename(data,
GAME_LAT = GAME_GEO_1,
GAME_LONG = GAME_GEO_2)

# NODE_GEO

data$NODE_LAT <- as.numeric(as.character(data$NODE_LAT))
data$NODE_LONG <- as.numeric(as.character(data$NODE_LONG))


data <- data[!is.na[13](data$CLIENT_LAT), ]

data <- data[!is.na[14](data$GAME_LAT), ]

data <- data[!is.na[15](data$NODE_LAT), ]

# Calculate distance from client IP to node IP
sLonLat <- data[,c(74, 73)]
dLonLat <- data[,c(68, 67)]

# create an empty vector to hold distances
dist <- vector()
#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(sLonLat)) {
dist[i] <- distm(sLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}

data$distance_h1 <- dist

# append distance column to the data frame

# dLonLat <- data[c(77, 76)]
eLonLat <- data[,c(76, 75)]


# create an empty vector to hold distances
dist2 <- vector()

#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(eLonLat)) {
dist2[i] <- distm(eLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}


# append distance column to the data frame
data$distance_h2 <- dist2
#
data$TOTAL_DISTANCE <- data$distance_h1 + data$distance_h2
#
```

13 http://is.na
14 http://is.na
15 http://is.na

```
#New Genre Columns on boolean format
data$ACTION <- as.numeric(data$GENRE == "ACTION")
data$ACT_ADV <- as.numeric(data$GENRE == "ACTIONADVENRUTE")
data$ACT_MASS_MULTY <- as.numeric(data$GENRE == "ACTIONMASSIVELY_MULTIPLAYER")
data$ACTIONRPG <- as.numeric(data$GENRE == "ACTIONRPG")
data$ACT_RPG_MM <- as.numeric(data$GENRE == "ACTIONRPGMASSIVELY_MULTIPLAYER")
data$ACT_RPG_STR <- as.numeric(data$GENRE == "ACTIONRPGSTRATEGY")
data$ACT_RPG_STR_MM_ADV <- as.numeric(data$GENRE ==
"ACTIONRPGSTRATEGYMASSIVELY_MULTIPLAYERSIMULATIONADVENTURE")
data$ACT_SHOOTER <- as.numeric(data$GENRE == "ACTIONSHOOTER")
data$ACT_STR_SIM <- as.numeric(data$GENRE == "ACTIONSTRATEGYSIMULATION")
data$RPG <- as.numeric(data$GENRE == "RPG")
data$RPGCASUAL <- as.numeric(data$GENRE == "RPGCASUAL")
data$RPG_MM <- as.numeric(data$GENRE == "RPGMASSIVELY_MULTIPLAYER")
data$SPORTS <- as.numeric(data$GENRE == "SPORTS")
data$STRATEGY <- as.numeric(data$GENRE == "STRATEGY")


data$OTHER <- (data$ACTION + data$ACT_ADV + data$ACT_MASS_MULTY + data$ACTIONRPG +
data$ACT_RPG_MM +
data$ACT_RPG_STR +data$ACT_RPG_STR_MM_ADV + data$ACT_SHOOTER + data$ACT_STR_SIM +
data$RPG + data$RPGCASUAL + data$RPG_MM + data$SPORTS + data$STRATEGY - 1) * -1


data <- select(data, -c(ID, P_ID, , P_INDEX, P_SCORE, GAME_SESSION_STOP, LOAD_TIME,
CONFIG_SOURCE, USER_TYPE, USED_PATHFINDER, MEMBERSHIP, USER_DAY, STATS_START,
PING_Q, LOSS_Q, SPKE_Q, CHAIN_COUNT, BYTES_UP, BYTES_TCP, BYTES_UDP, BYTES_TOTAL,
SOCKET_COUNT, CLIENT_IP, CLIENT_IP_STR, NODE_COUNT, NODE_IP_STR, NODE_CONTINENT,
NODE_COUNTRY, NODE_REGION, NODE_COUNTRY_REGION_CITY, NODE_ISP, NODE_ORG,
GAME_IP_STR, DAY_COHORT, GENRE,
GAME_SESSION_START_DATE, GAME_SESSION_START_TIME, weekend, CLIENT_LAT,
CLIENT_LONG, GAME_LAT, GAME_LONG, NODE_LAT, NODE_LONG, distance_h1,
distance_h2, FLUX_Q))

data <- data[-(22:31)]
data <- data[-2]

data$BYTES_DOWN_TCP <- data$BYTES_DOWN_TCP/1000000
data$BYTES_UP_TCP <- data$BYTES_UP_TCP/1000000
data$BYTES_DOWN_UDP <- data$BYTES_DOWN_UDP/1000000
data$BYTES_DOWN_TCP <- data$BYTES_DOWN_TCP/1000000
data$TOTAL_DISTANCE <- data$TOTAL_DISTANCE/1000
data$BYTES_PER_SECOND <- data$BYTES_PER_SECOND/1000
data$DISTANCE_2 <- data$TOTAL_DISTANCE^2


data <- as.data.frame(data)
#Save Data Set as CSV
write.csv(data,"C:/Users/Edward\Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and
Output CSV Files/Data_PopGames_Sep22.csv", row.names = FALSE)

# NORMALIZE THE DATA
# data <- read.csv(("C:/Users/Edward\Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and
Output CSV Files/Data_PopGames_Sep22.csv"))
library(dplyr)
```

```
data %>% distinct()
data <- distinct(data)


data[,3] <- (data[,3]-mean(data[,3]))/sd(data[,3])
data[,4] <- (data[,4]-mean(data[,4]))/sd(data[,4])
data[,5] <- (data[,5]-mean(data[,5]))/sd(data[,5])
data[,6] <- (data[,6]-mean(data[,6]))/sd(data[,6])
data[,7] <- (data[,7]-mean(data[,7]))/sd(data[,7])
data[,8] <- (data[,8]-mean(data[,8]))/sd(data[,8])
data[,9] <- (data[,9]-mean(data[,9]))/sd(data[,9])
data[,10] <- (data[,10]-mean(data[,10]))/sd(data[,10])
data[,11] <- (data[,11]-mean(data[,11]))/sd(data[,11])
data[,12] <- (data[,12]-mean(data[,12]))/sd(data[,12])
data[,13] <- (data[,13]-mean(data[,13]))/sd(data[,13])
data[,14] <- (data[,14]-mean(data[,14]))/sd(data[,14])
data[,15] <- (data[,15]-mean(data[,15]))/sd(data[,15])
data[,16] <- (data[,16]-mean(data[,16]))/sd(data[,16])
data[,17] <- (data[,17]-mean(data[,17]))/sd(data[,17])
data[,18] <- (data[,18]-mean(data[,18]))/sd(data[,18])
data[,19] <- (data[,19]-mean(data[,19]))/sd(data[,19])
data[,20] <- (data[,20]-mean(data[,20]))/sd(data[,20])

data[,22] <- (data[,22]-mean(data[,22]))/sd(data[,22])
data[,23] <- (data[,23]-mean(data[,23]))/sd(data[,23])

data[,33] <- (data[,33]-mean(data[,33]))/sd(data[,33])
data[,49] <- (data[,49]-mean(data[,49]))/sd(data[,49])

write.csv(data,"C:/Users/Edward\Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and
Output CSV Files//Norm_Data_Sep22.csv", row.names = FALSE)
```

10.10.2 Source Code 2

Data_Transform_Duration_S19_Sep29.R

```
# Copyright (c) 2021 MIT License
# GPERF2 Project
# Okanagan College
#
# Object: Feature Engineering
# Version 1.1
# Author: Luis Zeuqueiro
# Creation date: 29/09/2020
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
```

```
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
# HISTORY:
# Date By Comments
# ---------- ----------- ----------------------------------------------------------
# 29/09/2020 - Luis Zequeiro - Initial Creation of version 1.1

rm()

library(tidyverse)
library(splitstackshape)
library(rgeolocate)
library(lubridate)
library(geosphere)
library(rgl)
library(dplyr)


# Set a working directory
setwd("C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and Output CSV
Files")

# Load the data
data <- read.csv("export_final.csv")
genres <- read.csv("genre_merge.csv")
names(genres)[1] <- "GENRE"
names(data)[1] <- "ID"
# data <- data[-]

NODE_GEO <- read.csv("Node_Geo.csv")
data <- cbind(data, genres)
data <- cbind(data, NODE_GEO)



data <- data[!(data$GAME_ID==0), ]
# DELETE DUPLICATED RECORDS
data <- distinct(data)

# Change Sep 29
# data <- select(data, -c(GAME_NAME.y))

# data <- data %>% mutate(GENRE = coalesce(GENRE, 0))

# Split timestamp and get day of the week and make it boolean

data <- cSplit(data, "GAME_SESSION_START", " ")
```

```
data <- dplyr::rename(data,
GAME_SESSION_START_DATE = GAME_SESSION_START_1,
GAME_SESSION_START_TIME = GAME_SESSION_START_2,)

data$weekend <- weekdays(as.Date(data$GAME_SESSION_START_DATE))


data <- data %>% mutate(WEEKEND = recode(weekend, "Monday" = 0, "Tuesday" = 0,
"Wednesday" = 0, "Thursday" = 0, "Friday" = 1, "Saturday" = 1, "Sunday" = 1))

# Calculate distance using Client Geo and Game Geo

#CLIENT_GEO
data <- cSplit(data, "CLIENT_GEO", ",")

data <- dplyr::rename(data,
CLIENT_LAT = CLIENT_GEO_1,
CLIENT_LONG = CLIENT_GEO_2)

#GAME_GEO
data <- cSplit(data, "GAME_GEO", ",")

data <- dplyr::rename(data,
GAME_LAT = GAME_GEO_1,
GAME_LONG = GAME_GEO_2)

data$NODE_LAT <- as.numeric(as.character(data$NODE_LAT))
data$NODE_LONG <- as.numeric(as.character(data$NODE_LONG))

# data <- cbind(data, NODE_GEO)

data <- data[!is.na[16](data$CLIENT_LAT), ]

data <- data[!is.na[17](data$GAME_LAT), ]

data <- data[!is.na[18](data$NODE_LAT), ]

# Calculate distance from client IP to node IP
sLonLat <- data[,c(74, 73)] # Client s
dLonLat <- data[,c(68, 67)] # Node d

# create an empty vector to hold distances
dist <- vector()
#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(sLonLat)) {
dist[i] <- distm(sLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}

data$distance_h1 <- dist

# append distance column to the data frame

# dLonLat <- data[c(77, 76)]
eLonLat <- data[,c(76, 75)] # gAME e
```

16 http://is.na
17 http://is.na
18 http://is.na

```
# create an empty vector to hold distances
dist2 <- vector()

#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(eLonLat)) {
dist2[i] <- distm(eLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}


# append distance column to the data frame
data$distance_h2 <- dist2
#
data$TOTAL_DISTANCE <- data$distance_h1 + data$distance_h2

#New Genre Columns on boolean format
data$ACTION <- as.numeric(data$GENRE == "ACTION")
data$ACT_ADV <- as.numeric(data$GENRE == "ACTIONADVENRUTE")
data$ACT_MASS_MULTY <- as.numeric(data$GENRE == "ACTIONMASSIVELY_MULTIPLAYER")
data$ACTIONRPG <- as.numeric(data$GENRE == "ACTIONRPG")
data$ACT_RPG_MM <- as.numeric(data$GENRE == "ACTIONRPGMASSIVELY_MULTIPLAYER")
data$ACT_RPG_STR <- as.numeric(data$GENRE == "ACTIONRPGSTRATEGY")
data$ACT_RPG_STR_MM_ADV <- as.numeric(data$GENRE ==
"ACTIONRPGSTRATEGYMASSIVELY_MULTIPLAYERSIMULATIONADVENTURE")
data$ACT_SHOOTER <- as.numeric(data$GENRE == "ACTIONSHOOTER")
data$ACT_STR_SIM <- as.numeric(data$GENRE == "ACTIONSTRATEGYSIMULATION")
data$RPG <- as.numeric(data$GENRE == "RPG")
data$RPGCASUAL <- as.numeric(data$GENRE == "RPGCASUAL")
data$RPG_MM <- as.numeric(data$GENRE == "RPGMASSIVELY_MULTIPLAYER")
data$SPORTS <- as.numeric(data$GENRE == "SPORTS")
data$STRATEGY <- as.numeric(data$GENRE == "STRATEGY")

data <- select(data, -c(ID, P_ID, P_INDEX, P_SCORE, GAME_SESSION_STOP, LOAD_TIME,
CONFIG_SOURCE, USER_TYPE, USED_PATHFINDER, MEMBERSHIP, USER_DAY, STATS_START,
PING_Q, LOSS_Q, SPKE_Q, CHAIN_COUNT, BYTES_UP, BYTES_TCP, BYTES_UDP, BYTES_TOTAL,
SOCKET_COUNT, CLIENT_IP, CLIENT_IP_STR, NODE_COUNT, NODE_IP_STR, NODE_CONTINENT,
NODE_COUNTRY, NODE_REGION, NODE_COUNTRY_REGION_CITY, NODE_ISP, NODE_ORG,
GAME_IP_STR, DAY_COHORT, GENRE, GAME_SESSION_START_DATE,
GAME_SESSION_START_TIME, weekend, CLIENT_LAT,
CLIENT_LONG, GAME_LAT, GAME_LONG, NODE_LAT, NODE_LONG, distance_h1,
distance_h2, FLUX_Q))

data$OTHER <- data$OTHER[is.na[19](data$OTHER)] <- 1
data[, 23:37][is.na[20](data[ ,23:37])] <- 0

# data$OTHER <- (data$ACTION + data$ACT_ADV + data$ACT_MASS_MULTY + data$ACTIONRPG +
data$ACT_RPG_MM +
# data$ACT_RPG_STR +data$ACT_RPG_STR_MM_ADV + data$ACT_SHOOTER + data$ACT_STR_SIM +
# data$RPG + data$RPGCASUAL + data$RPG_MM + data$SPORTS + data$STRATEGY - 1) * -1
```

19 http://is.na
20 http://is.na

```
data$BYTES_DOWN_TCP <- data$BYTES_DOWN_TCP/1000000
data$BYTES_UP_TCP <- data$BYTES_UP_TCP/1000000
data$BYTES_DOWN_UDP <- data$BYTES_DOWN_UDP/1000000
data$BYTES_UP_UDP <- data$BYTES_UP_UDP/1000000
data$TOTAL_DISTANCE <- data$TOTAL_DISTANCE/1000
data$BYTES_PER_SECOND <- data$BYTES_PER_SECOND/1000
data$DISTANCE_2 <- data$TOTAL_DISTANCE^2

# Changes Sep 22 - Divide BYTES variables by DURATION
data$DUR_BU_TCP <- data$BYTES_UP_TCP/data$DURATION
data$DUR_BU_UDP <- data$BYTES_UP_UDP/data$DURATION
data$DUR_BD_TCP <- data$BYTES_DOWN_TCP/data$DURATION
data$DUR_BD_UDP <- data$BYTES_DOWN_UDP/data$DURATION

data <- as.data.frame(data)

# DELETE DUPLICATED RECORDS
data <- distinct(data)

#Save Data Set as CSV
write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_PopGames_Sep29.csv", row.names = FALSE)
#
# data <- read.csv(("/Users/luisfernandozequeirogalvez/Documents/Research Assistant OK/FILES R/
SPRINT_19_PROJECTS/Transformed_Data/Data_PopGames_Sep29.csv"))

data <- filter(data, DURATION <= 43200)

write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_Duration_Sep29.csv", row.names = FALSE)

#Changes Sep 29 Transformation square root
data$INTERNET_LOSS <- sqrt(data$INTERNET_LOSS)
data$INTERNET_SPKE <- sqrt(data$INTERNET_SPKE)
data$SOCKET_COUNT_TCP <- sqrt(data$SOCKET_COUNT_TCP)
data$SOCKET_COUNT_UDP <- sqrt(data$SOCKET_COUNT_UDP)
data$GAME_IP_COUNT <- sqrt(data$GAME_IP_COUNT)
data$BYTES_PER_SECOND <- sqrt(data$BYTES_PER_SECOND)


# Changes Sep 29 - New categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM +
# STRATEGY + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY
data$ALL_OTHER = data$ACT_RPG_STR_MM_ADV + data$SPORTS + data$ACT_RPG_MM + data$STRATEGY +
data$RPG + data$OTHER + data$ACTIONRPG + data$ACT_STR_SIM + data$ACT_ADV + data$ACT_MASS_MULTY

#Changes Sep 29 Delet columns after creation of categorical variable ALL_OTHERS
data <- select(data, -c(GAME_NAME, GAME_ID, CONFIG_NAME, ACT_RPG_STR_MM_ADV, SPORTS, ACT_RPG_MM,
STRATEGY, RPG,
WTFAST_LOSS, WTFAST_SPKE, WTFAST_FLUX, DURATION, DUR_BD_UDP, DUR_BU_UDP,
DUR_BD_TCP, DUR_BU_TCP, DISTANCE_2, OTHER, ACTIONRPG, ACT_STR_SIM, ACT_ADV,
ACT_MASS_MULTY, ID_1))
data <- data[-(15:23)]


data$ALL_OTHER[data$ALL_OTHER>1]<- 0

# NORMALIZE THE DATA
# Changes Sep 29 - Normalize all data except boolian and DURATION
data[,1] <- (data[,1]-mean(data[,1]))/sd(data[,1])
```

```
data[,2] <- (data[,2]-mean(data[,2]))/sd(data[,2])
data[,3] <- (data[,3]-mean(data[,3]))/sd(data[,3])
data[,4] <- (data[,4]-mean(data[,4]))/sd(data[,4])
data[,5] <- (data[,5]-mean(data[,5]))/sd(data[,5])
data[,6] <- (data[,6]-mean(data[,6]))/sd(data[,6])
data[,7] <- (data[,7]-mean(data[,7]))/sd(data[,7])
data[,8] <- (data[,8]-mean(data[,8]))/sd(data[,8])
data[,9] <- (data[,9]-mean(data[,9]))/sd(data[,9])
data[,10] <- (data[,10]-mean(data[,10]))/sd(data[,10])
data[,11] <- (data[,11]-mean(data[,11]))/sd(data[,11])
data[,12] <- (data[,12]-mean(data[,12]))/sd(data[,12])
data[,13] <- (data[,13]-mean(data[,13]))/sd(data[,13])
data[,14] <- (data[,14]-mean(data[,14]))/sd(data[,14])
# Weekend 15
data[,16] <- (data[,16]-mean(data[,16]))/sd(data[,16])


write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Norm_Data_Sep29.csv", row.names = FALSE)
```

10.10.3 Source Code 3

Data_Transform_Duration_S19_Oct05.R

```
# Copyright (c) 2021 MIT License
# GPERF2 Project
# Okanagan College
#
# Object: Feature Engineering
# Version 1.2
# Author: Luis Zeuqueiro
# Creation date: 05/10/2020
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
# HISTORY:
# Date By Comments
```

```
# ---------- ---------- ----------------------------------------------------------
# 05/10/2020 - Luis Zequeiro - Initial Creation of version 1.2

library(tidyverse)
library(splitstackshape)
library(rgeolocate)
library(lubridate)
library(geosphere)
library(rgl)
library(dplyr)

# Set a working directory
setwd("C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and Output CSV
Files")

# Load the data


# Notes October5
# The merge was made using VLOOKUP in Excel. merge() function presented errors
# when merging columns, duplicating and adding data.
# Also merge NODE_GEO from the beginning or the data will duplicate
# observations due to mismatch or rows
data <- read.csv("export_final.csv")
genre <- read.csv("genre_merge.csv")
NODE_GEO <- read.csv("Node_Geo.csv")

# cHANGE COLUMN NAMES (ERROR WHEN DOWNLOADING DATA TO VM)
names(data)[1] <- "ID"

NODE_GEO <- mutate_if(NODE_GEO, is.character, ~ as.numeric(as.numeric(.x)))
data <- cbind(data,genre)
data <- cbind(data, NODE_GEO)
# Deleting duplicated records from original file, including all GAME_ID=0
data <- distinct(data)

# Change Sep 29
data <- data[!(data$GAME_ID==0), ]

# Split timestamp and get day of the week and make it boolean
data <- cSplit(data, "GAME_SESSION_START", " ")

data <- dplyr::rename(data,
GAME_SESSION_START_DATE = GAME_SESSION_START_1,
GAME_SESSION_START_TIME = GAME_SESSION_START_2,)

data$weekend <- weekdays(as.Date(data$GAME_SESSION_START_DATE))


data <- data %>% mutate(WEEKEND = recode(weekend, "Monday" = 0, "Tuesday" = 0,
"Wednesday" = 0, "Thursday" = 0, "Friday" = 1, "Saturday" = 1, "Sunday" = 1))

# Calculate distance using Client Geo and Game Geo

#CLIENT_GEO
data <- cSplit(data, "CLIENT_GEO", ",")

data <- dplyr::rename(data,
CLIENT_LAT = CLIENT_GEO_1,
CLIENT_LONG = CLIENT_GEO_2)
```

```
#GAME_GEO
data <- cSplit(data, "GAME_GEO", ",")

data <- dplyr::rename(data,
GAME_LAT = GAME_GEO_1,
GAME_LONG = GAME_GEO_2)
# Notes October 5
# Delete all observations with no data on Latitude to avoid NAs and errors.
data <- data[!is.na[21](data$CLIENT_LAT), ]

data <- data[!is.na[22](data$GAME_LAT), ]

data <- data[!is.na[23](data$NODE_LAT), ]
data <- as.data.frame(data)
# Calculate distance from client IP to node IP
sLonLat <- data[,c(74, 73)] # Client s
dLonLat <- data[,c(68,67)] # Node d

# create an empty vector to hold distances
dist <- vector()
#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(sLonLat)) {
dist[i] <- distm(sLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}

data$distance_h1 <- dist

# append distance column to the data frame

# dLonLat <- data[c(77, 76)]
eLonLat <- data[,c(76, 75)] # Game e


# create an empty vector to hold distances
dist2 <- vector()

#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(eLonLat)) {
dist2[i] <- distm(eLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}


# append distance column to the data frame
data$distance_h2 <- dist2
#
data$TOTAL_DISTANCE <- data$distance_h1 + data$distance_h2
```

21 http://is.na
22 http://is.na
23 http://is.na

```
names(data)[66] <- "GENRE"
# New Genre Columns on boolean format
data$ACTION <- as.numeric(data$GENRE == "ACTION")
data$ACT_ADV <- as.numeric(data$GENRE == "ACTIONADVENRUTE")
data$ACT_MASS_MULTY <- as.numeric(data$GENRE == "ACTIONMASSIVELY_MULTIPLAYER")
data$ACTIONRPG <- as.numeric(data$GENRE == "ACTIONRPG")
data$ACT_RPG_MM <- as.numeric(data$GENRE == "ACTIONRPGMASSIVELY_MULTIPLAYER")
data$ACT_RPG_STR <- as.numeric(data$GENRE == "ACTIONRPGSTRATEGY")
data$ACT_RPG_STR_MM_ADV <- as.numeric(data$GENRE ==
"ACTIONRPGSTRATEGYMASSIVELY_MULTIPLAYERSIMULATIONADVENTURE")
data$ACT_SHOOTER <- as.numeric(data$GENRE == "ACTIONSHOOTER")
data$ACT_STR_SIM <- as.numeric(data$GENRE == "ACTIONSTRATEGYSIMULATION")
data$RPG <- as.numeric(data$GENRE == "RPG")
data$RPGCASUAL <- as.numeric(data$GENRE == "RPGCASUAL")
data$RPG_MM <- as.numeric(data$GENRE == "RPGMASSIVELY_MULTIPLAYER")
data$SPORTS <- as.numeric(data$GENRE == "SPORTS")
data$STRATEGY <- as.numeric(data$GENRE == "STRATEGY")

# Delete columns with unnecesary variables
data <- select(data, -c(ID, P_ID, P_INDEX, P_SCORE, GAME_SESSION_STOP, LOAD_TIME,
CONFIG_SOURCE, USER_TYPE, USED_PATHFINDER, MEMBERSHIP, USER_DAY, STATS_START,
PING_Q, LOSS_Q, SPKE_Q, CHAIN_COUNT, BYTES_UP, BYTES_TCP, BYTES_UDP, BYTES_TOTAL,
SOCKET_COUNT, CLIENT_IP, CLIENT_IP_STR, NODE_COUNT, NODE_IP_STR, NODE_CONTINENT,
NODE_COUNTRY, NODE_REGION, NODE_COUNTRY_REGION_CITY, NODE_ISP, NODE_ORG,
GAME_IP_STR, DAY_COHORT, GENRE,
GAME_SESSION_START_DATE, GAME_SESSION_START_TIME, weekend, CLIENT_LAT,
CLIENT_LONG, GAME_LAT, GAME_LONG, NODE_LAT, NODE_LONG, distance_h1,
distance_h2, FLUX_Q))


# data[, 23:37][is.na[24](data[ ,23:37])] <- 0

data$OTHER <- (data$ACTION + data$ACT_ADV + data$ACT_MASS_MULTY + data$ACTIONRPG +
data$ACT_RPG_MM +
data$ACT_RPG_STR +data$ACT_RPG_STR_MM_ADV + data$ACT_SHOOTER + data$ACT_STR_SIM +
data$RPG + data$RPGCASUAL + data$RPG_MM + data$SPORTS + data$STRATEGY - 1) * -1

# Change September 29 - Create a new variabla for Genres
data$OTHER <- data$OTHER[is.na[25](data$OTHER)] <- 1
# Change September 22 - Scale all variables and get Distance^2
data$BYTES_DOWN_TCP <- data$BYTES_DOWN_TCP/1000000
data$BYTES_UP_TCP <- data$BYTES_UP_TCP/1000000
data$BYTES_DOWN_UDP <- data$BYTES_DOWN_UDP/1000000
data$BYTES_UP_UDP <- data$BYTES_UP_UDP/1000000
data$TOTAL_DISTANCE <- data$TOTAL_DISTANCE/1000
data$BYTES_PER_SECOND <- data$BYTES_PER_SECOND/1000
data$DISTANCE_2 <- data$TOTAL_DISTANCE^2

# Changes Sep 22 - Divide BYTES variables by DURATION
data$DUR_BU_TCP <- data$BYTES_UP_TCP/data$DURATION
data$DUR_BU_UDP <- data$BYTES_UP_UDP/data$DURATION
```

24 http://is.na
25 http://is.na

```
data$DUR_BD_TCP <- data$BYTES_DOWN_TCP/data$DURATION
data$DUR_BD_UDP <- data$BYTES_DOWN_UDP/data$DURATION

data <- data[-(22:31)]

data <- data[!is.na[26](data$DUR_BD_UDP), ]
# CHANGE FROM OCTOBER 6 - ELIMINATE ALL VALUES ON DURATION = 0
data <- data[!(data$DURATION==0),]
#Save Data Set as CSV Total observations: 55,517
write.csv(data,"C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_PopGames_Oct05.csv", row.names = FALSE)
# ------------------------------------------------

# Filter all data using Duration <= to 43,200
data <- filter(data, DURATION <= 43200)

# Total observations: 52,635
write.csv(data,"C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_Duration_Oct05.csv", row.names = FALSE)
# ------------------------------------------------

# Changes Sep 29 Transformation square root
data$INTERNET_LOSS <- sqrt(data$INTERNET_LOSS)
data$INTERNET_SPKE <- sqrt(data$INTERNET_SPKE)
data$SOCKET_COUNT_TCP <- sqrt(data$SOCKET_COUNT_TCP)
data$SOCKET_COUNT_UDP <- sqrt(data$SOCKET_COUNT_UDP)
data$GAME_IP_COUNT <- sqrt(data$GAME_IP_COUNT)
data$BYTES_PER_SECOND <- sqrt(data$BYTES_PER_SECOND)


# Changes Sep 29 - New categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM +
# STRATEGY + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY
data$ALL_OTHER = data$ACT_RPG_STR_MM_ADV + data$SPORTS + data$ACT_RPG_MM + data$STRATEGY +
data$RPG + data$OTHER + data$ACTIONRPG + data$ACT_STR_SIM + data$ACT_ADV + data$ACT_MASS_MULTY

# Create a categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM + STRATEGY
# + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY

#Changes Sep 29 Delet columns after creation of categorical variable ALL_OTHERS
data <- select(data, -c(GAME_ID, GAME_NAME, CONFIG_NAME, ACT_RPG_STR_MM_ADV, SPORTS, ACT_RPG_MM,
STRATEGY, RPG,
WTFAST_LOSS, WTFAST_SPKE, WTFAST_FLUX, DURATION, DUR_BD_UDP, DUR_BU_UDP,
DUR_BD_TCP, DUR_BU_TCP, DISTANCE_2, OTHER, ACTIONRPG, ACT_STR_SIM, ACT_ADV,
ACT_MASS_MULTY))

data$ALL_OTHER[data$ALL_OTHER>1]<- 0

# NORMALIZE THE DATA
# Changes Sep 29 - Normalize all data except boolian and DURATION
data[,1] <- (data[,1]-mean(data[,1]))/sd(data[,1])
data[,2] <- (data[,2]-mean(data[,2]))/sd(data[,2])
data[,3] <- (data[,3]-mean(data[,3]))/sd(data[,3])
data[,4] <- (data[,4]-mean(data[,4]))/sd(data[,4])
data[,5] <- (data[,5]-mean(data[,5]))/sd(data[,5])
data[,6] <- (data[,6]-mean(data[,6]))/sd(data[,6])
data[,7] <- (data[,7]-mean(data[,7]))/sd(data[,7])
```

[26] http://is.na

```
data[,8] <- (data[,8]-mean(data[,8]))/sd(data[,8])
data[,9] <- (data[,9]-mean(data[,9]))/sd(data[,9])
data[,10] <- (data[,10]-mean(data[,10]))/sd(data[,10])
data[,11] <- (data[,11]-mean(data[,11]))/sd(data[,11])
data[,12] <- (data[,12]-mean(data[,12]))/sd(data[,12])
data[,13] <- (data[,13]-mean(data[,13]))/sd(data[,13])
data[,14] <- (data[,14]-mean(data[,14]))/sd(data[,14])
# Weekend 15
data[,16] <- (data[,16]-mean(data[,16]))/sd(data[,16])
#Genres 17-22


write.csv(data,"C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Norm_Data_Oct05.csv", row.names = FALSE)
```

Source Code 4

Data_Transform_Duration_S19_Oct15.R

```
# Copyright (c) 2021 MIT License
# GPERF2 Project
# Okanagan College
#
# Object: Feature Engineering
# Version 1.3
# Author: Luis Zeuqueiro
# Creation date: 15/10/2020
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
# HISTORY:
# Date By Comments
# ---------- ----------- ----------------------------------------------------------
# 15/10/2020 - Luis Zequeiro - Initial Creation of version 1.3

library(tidyverse)
library(splitstackshape)
```

```
library(rgeolocate)
library(lubridate)
library(geosphere)
library(rgl)
library(dplyr)

# Set a working directory
setwd("C:/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Input and Output CSV Files")

# Load the data


# Notes October5
# The merge was made using VLOOKUP in Excel. merge() function presented errors
# when merging columns, duplicating and adding data.
# Also merge NODE_GEO from the beginning or the data will duplicate
# observations due to mismatch or rows
data <- read.csv("export_final.csv")
genre <- read.csv("genre_merge.csv")
NODE_GEO <- read.csv("Node_Geo.csv")

# cHANGE COLUMN NAMES (ERROR WHEN DOWNLOADING DATA TO VM)
names(data)[1] <- "ID"

NODE_GEO <- mutate_if(NODE_GEO, is.character, ~ as.numeric(as.numeric(.x)))
data <- cbind(data,genre)
data <- cbind(data, NODE_GEO)
# Deleting duplicated records from original file, including all GAME_ID=0
data <- distinct(data)

# Change Sep 29
data <- data[!(data$GAME_ID==0), ]

# Split timestamp and get day of the week and make it boolean
data <- cSplit(data, "GAME_SESSION_START", " ")

data <- dplyr::rename(data,
GAME_SESSION_START_DATE = GAME_SESSION_START_1,
GAME_SESSION_START_TIME = GAME_SESSION_START_2,)

data$weekend <- weekdays(as.Date(data$GAME_SESSION_START_DATE))


data <- data %>% mutate(WEEKEND = recode(weekend, "Monday" = 0, "Tuesday" = 0,
"Wednesday" = 0, "Thursday" = 0, "Friday" = 1, "Saturday" = 1, "Sunday" = 1))

# Calculate distance using Client Geo and Game Geo

#CLIENT_GEO
data <- cSplit(data, "CLIENT_GEO", ",")

data <- dplyr::rename(data,
CLIENT_LAT = CLIENT_GEO_1,
CLIENT_LONG = CLIENT_GEO_2)

#GAME_GEO
data <- cSplit(data, "GAME_GEO", ",")
```

```
data <- dplyr::rename(data,
GAME_LAT = GAME_GEO_1,
GAME_LONG = GAME_GEO_2)
# Notes October 5
# Delete all observations with no data on Latitude to avoid NAs and errors.
data <- data[!is.na[27](data$CLIENT_LAT), ]

data <- data[!is.na[28](data$GAME_LAT), ]

data <- data[!is.na[29](data$NODE_LAT), ]
data <- as.data.frame(data)
# Calculate distance from client IP to node IP
sLonLat <- data[,c(74, 73)] # Client s
dLonLat <- data[,c(68,67)] # Node d

# create an empty vector to hold distances
dist <- vector()
#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(sLonLat)) {
dist[i] <- distm(sLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}

data$distance_h1 <- dist

# append distance column to the data frame

# dLonLat <- data[c(77, 76)]
eLonLat <- data[,c(76, 75)] # Game e


# create an empty vector to hold distances
dist2 <- vector()

#
# # For each row of our lat/lon, we calculate the distance.
# # use geo distance distm which gives distance in meters; convert to KM
# # insert the result into the dist vector
for (i in 1:nrow(eLonLat)) {
dist2[i] <- distm(eLonLat[i,], dLonLat[i,], fun = distHaversine) / 1000
}


# append distance column to the data frame
data$distance_h2 <- dist2
#
data$TOTAL_DISTANCE <- data$distance_h1 + data$distance_h2

names(data)[66] <- "GENRE"
# New Genre Columns on boolean format
data$ACTION <- as.numeric(data$GENRE == "ACTION")
data$ACT_ADV <- as.numeric(data$GENRE == "ACTIONADVENRUTE")
```

27 http://is.na
28 http://is.na
29 http://is.na

```
data$ACT_MASS_MULTY <- as.numeric(data$GENRE == "ACTIONMASSIVELY_MULTIPLAYER")
data$ACTIONRPG <- as.numeric(data$GENRE == "ACTIONRPG")
data$ACT_RPG_MM <- as.numeric(data$GENRE == "ACTIONRPGMASSIVELY_MULTIPLAYER")
data$ACT_RPG_STR <- as.numeric(data$GENRE == "ACTIONRPGSTRATEGY")
data$ACT_RPG_STR_MM_ADV <- as.numeric(data$GENRE ==
"ACTIONRPGSTRATEGYMASSIVELY_MULTIPLAYERSIMULATIONADVENTURE")
data$ACT_SHOOTER <- as.numeric(data$GENRE == "ACTIONSHOOTER")
data$ACT_STR_SIM <- as.numeric(data$GENRE == "ACTIONSTRATEGYSIMULATION")
data$RPG <- as.numeric(data$GENRE == "RPG")
data$RPGCASUAL <- as.numeric(data$GENRE == "RPGCASUAL")
data$RPG_MM <- as.numeric(data$GENRE == "RPGMASSIVELY_MULTIPLAYER")
data$SPORTS <- as.numeric(data$GENRE == "SPORTS")
data$STRATEGY <- as.numeric(data$GENRE == "STRATEGY")

# Delete columns with unnecessary variables
data <- select(data, -c(ID, P_ID, P_INDEX, P_SCORE, GAME_SESSION_STOP, LOAD_TIME,
CONFIG_SOURCE, USER_TYPE, USED_PATHFINDER, MEMBERSHIP, USER_DAY, STATS_START,
PING_Q, LOSS_Q, SPKE_Q, CHAIN_COUNT, BYTES_UP, BYTES_TCP, BYTES_UDP, BYTES_TOTAL,
SOCKET_COUNT, CLIENT_IP, CLIENT_IP_STR, NODE_COUNT, NODE_IP_STR, NODE_CONTINENT,
NODE_COUNTRY, NODE_REGION, NODE_COUNTRY_REGION_CITY, NODE_ISP, NODE_ORG,
GAME_IP_STR, DAY_COHORT, GENRE,
GAME_SESSION_START_DATE, GAME_SESSION_START_TIME, weekend, CLIENT_LAT,
CLIENT_LONG, GAME_LAT, GAME_LONG, NODE_LAT, NODE_LONG, distance_h1,
distance_h2, FLUX_Q))


# data[, 23:37][is.na[30](data[ ,23:37])] <- 0

data$OTHER <- (data$ACTION + data$ACT_ADV + data$ACT_MASS_MULTY + data$ACTIONRPG +
data$ACT_RPG_MM +
data$ACT_RPG_STR +data$ACT_RPG_STR_MM_ADV + data$ACT_SHOOTER + data$ACT_STR_SIM +
data$RPG + data$RPGCASUAL + data$RPG_MM + data$SPORTS + data$STRATEGY)

data$OTHER <- ifelse(data$OTHER<1 | data$OTHER==0, 1, 0)


# describe(data$OTHER)
# Change September 29 - Create a new variabla for Genres
# data$OTHER <- data$OTHER[is.na[31](data$OTHER)] <- 1
# Change September 22 - Scale all variables and get Distance^2
data$BYTES_DOWN_TCP <- data$BYTES_DOWN_TCP/1000000
data$BYTES_UP_TCP <- data$BYTES_UP_TCP/1000000
data$BYTES_DOWN_UDP <- data$BYTES_DOWN_UDP/1000000
data$BYTES_UP_UDP <- data$BYTES_UP_UDP/1000000
data$TOTAL_DISTANCE <- data$TOTAL_DISTANCE/1000
data$BYTES_PER_SECOND <- data$BYTES_PER_SECOND/1000
data$DISTANCE_2 <- data$TOTAL_DISTANCE^2

# Changes Sep 22 - Divide BYTES variables by DURATION
data$DUR_BU_TCP <- data$BYTES_UP_TCP/data$DURATION
data$DUR_BU_UDP <- data$BYTES_UP_UDP/data$DURATION
```

30 http://is.na
31 http://is.na

```
data$DUR_BD_TCP <- data$BYTES_DOWN_TCP/data$DURATION
data$DUR_BD_UDP <- data$BYTES_DOWN_UDP/data$DURATION

data <- data[-(22:31)]

data <- data[!is.na[32](data$DUR_BD_UDP), ]
# CHANGE FROM OCTOBER 6 - ELIMINATE ALL VALUES ON DURATION = 0
data <- data[!(data$DURATION==0),]
#Save Data Set as CSV Total observations: 55,517
write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_PopGames_Oct15.csv", row.names = FALSE)
# ------------------------------------------------

# Filter all data using Duration <= to 43,200
data <- filter(data, DURATION <= 43200)

# Total observations: 52,635
write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Data_Duration_Oct15.csv", row.names = FALSE)
# ------------------------------------------------

# Changes Sep 29 Transformation square root
data$INTERNET_LOSS <- sqrt(data$INTERNET_LOSS)
data$INTERNET_SPKE <- sqrt(data$INTERNET_SPKE)
data$SOCKET_COUNT_TCP <- sqrt(data$SOCKET_COUNT_TCP)
data$SOCKET_COUNT_UDP <- sqrt(data$SOCKET_COUNT_UDP)
data$GAME_IP_COUNT <- sqrt(data$GAME_IP_COUNT)
data$BYTES_PER_SECOND <- sqrt(data$BYTES_PER_SECOND)

# Changes Sep 29 - New categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM +
# STRATEGY + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY
#Changes October 13 - Remove data$OTHER vriable form ALL_OTHER
data$ALL_OTHER = data$ACT_RPG_STR_MM_ADV + data$SPORTS + data$ACT_RPG_MM +
data$STRATEGY + data$RPG + data$ACTIONRPG + data$ACT_STR_SIM +
data$ACT_ADV + data$ACT_MASS_MULTY + data$OTHER

# Note from October 18
# All boolean variables should be converted in to factor to avoid any errors while running any ML model
data$WEEKEND <- as.factor(data$WEEKEND)
data$ACTION <- as.factor(data$ACTION)
data$ACT_RPG_STR <- as.factor(data$ACT_RPG_STR)
data$ACT_SHOOTER <- as.factor(data$ACT_SHOOTER)
data$RPGCASUAL <- as.factor(data$RPGCASUAL)
data$RPG_MM <- as.factor(data$RPG_MM)
data$OTHER <- as.factor(data$OTHER)
data$ALL_OTHER <- as.factor(data$ALL_OTHER)

# Create a categorical variable ALL_OTHER = ACT_RPG_STR_MM_ADV + SPORTS + ACT_RPG_MM + STRATEGY
# + RPG + OTHER + ACTIONRPG + ACT_STR_SIM + ACT_ADV+ ACT_MASS_MULTY

#Changes Sep 29 Delet columns after creation of categorical variable ALL_OTHERS
#Changes October 13 - do not delete data$OTHER and keep it separetly
data <- select(data, -c(GAME_ID, GAME_NAME, CONFIG_NAME, WTFAST_LOSS, WTFAST_SPKE, WTFAST_FLUX,
DURATION, DUR_BD_UDP, DUR_BU_UDP, DUR_BD_TCP, DUR_BU_TCP, DISTANCE_2,
```

32 http://is.na

```
ACT_RPG_STR_MM_ADV, SPORTS, ACT_RPG_MM, STRATEGY, RPG, ACTIONRPG,
ACT_STR_SIM, ACT_ADV, ACT_MASS_MULTY, OTHER))

# data$ALL_OTHER[data$ALL_OTHER>1]<- 0

# NORMALIZE THE DATA
# Changes Sep 29 - Normalize all data except boolian and DURATION
data[,1] <- (data[,1]-mean(data[,1]))/sd(data[,1])
data[,2] <- (data[,2]-mean(data[,2]))/sd(data[,2])
data[,3] <- (data[,3]-mean(data[,3]))/sd(data[,3])
data[,4] <- (data[,4]-mean(data[,4]))/sd(data[,4])
data[,5] <- (data[,5]-mean(data[,5]))/sd(data[,5])
data[,6] <- (data[,6]-mean(data[,6]))/sd(data[,6])
data[,7] <- (data[,7]-mean(data[,7]))/sd(data[,7])
data[,8] <- (data[,8]-mean(data[,8]))/sd(data[,8])
data[,9] <- (data[,9]-mean(data[,9]))/sd(data[,9])
data[,10] <- (data[,10]-mean(data[,10]))/sd(data[,10])
data[,11] <- (data[,11]-mean(data[,11]))/sd(data[,11])
data[,12] <- (data[,12]-mean(data[,12]))/sd(data[,12])
data[,13] <- (data[,13]-mean(data[,13]))/sd(data[,13])
data[,14] <- (data[,14]-mean(data[,14]))/sd(data[,14])
# Weekend 15
data[,16] <- (data[,16]-mean(data[,16]))/sd(data[,16])
#Genres 17-22


write.csv(data,"/Users/Edward/Documents/WTFast Backup files/A. Section 2 Appendix 1 ETL/Scripts/
Norm_Data_Oct15.csv", row.names = FALSE)
```

10.11 Appendix 1.3 Copy of this Document

11 Appendix 2: Source Code

11.1

2.1 Bash GPerfCleaner

Video Tutorial: GperfCleaner Tutorial[33]

Using the R programing language, we clean and merge the data into a single CSV file. The cleaning process takes all UNIX timestamps and converts them into human-readable timestamps in the format of "yyyy-MM-dd hh:mm:ss:SSS." The cleaning process also includes changing any strings that show up as "NULL," "NA," or "null" are replaced with empty attributes. After this, the data is exported into a CSV file.

11.1.1 Usage

./gperfcleaner [-conv convert] [-pd plaiddata] [-cs clientstatsdata] [-P numberofprocesses] [-if inputfile] [-o outputFolder]
[-conv] # convert input folder from json to csv]
[-cs] # specify input data as client stats data to be cleaned]
[-pd] # specify input data as plaid data to be cleaned]
[-if] # specify single input file]
[-o] # output destination]
[-P] # specify number of processes]

11.1.2 Plaid Data Convert + Clean

```
./gperfcleaner -conv -pd -P 5 -o converted/
```

As you can see, we ran the gperfcleaner script with the option to convert plaid data, using 5 CPU processes, to the directory named 'converted'. If you are dealing with many small files, it is wise to give a larger process count (eg 50).

11.1.3 Client Stats Data Convert + Clean

```
./gperfcleaner -conv -cs -P 50 -o foo/
```

33 https://confluence2020.okanagan.bc.ca/download/attachments/16089972/2021-02-17%2011-04-45.mkv?api=v2&modificationDate=1613589634511&version=1

In the above instance, we are running the gperfcleaner to convert client stats data, using 50 CPU processes, to the directory 'foo'.

```
# Author: James Behnke
# Purpose: Automates the process of running different programs to clean and transform monthly WTFast data dumps.
# Usage: This script accepts Plaid or Client stats data and performs the associated cleaning and tranformation scripts with said data.
# License: MIT License

# Copyright (c) [2021] [James Behnke]

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

# Begin

#!/bin/bash

currentDir=$(pwd)
numProcesses=1
convert=0
plaidData=0
clientData=0
clean=0
dataFamily=0
convertOnly="find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh -c 'jsonexport '{}' ./JSONtoCSV/'{}.csv'; mv ./JSONtoCSV/ "$destinationFolder"; echo '{}' converted to CSV'"
availableFiles=$(ls *.json | wc -l)

checkDirExist () {
[ -d "$1" ] && echo "Directory "$1" exists." || mkdir "$1"
}


echo "Processing $availableFiles files"
((!$#)) && echo -e "No arguments supplied.\v" && (echo -e "usage: gperfcleaner [-conv convert] [-pd plaiddata] [-cs clientstatsdata] [-P numberofprocesses] [-o outputFolder]"
        echo -e "\t[-conv \t# convert input folder from json to csv]"
        echo -e "\t[-cs \t# specify input data as client stats data to be cleaned]"
        echo -e "\t[-pd \t# specify input data as plaid data to be cleaned]"
        echo -e "\t[-o \t# output destination]"
```

```
        echo -e "\t[-P \t# specify number of processes]")

        echo "Created By James Behnke. Last Edited January 7 2021"
        echo "For examples and tutorials visit: https://confluence2020.okanagan.bc.ca/x/CYf1 "



while [ -n "$1" ]; do # while loop starts and runs until it has handled all arguments

    case "$1" in

    -h) echo -e "usage: gperfcleaner [-conv convert] [-pd plaiddata] [-cs clientstatsdata] [-P numberofprocesses] [-o outputFolder] "
        echo -e "\t[-conv \t# convert input folder from json to csv]"
        echo -e "\t[-cs \t# specify input data as client stats data to be cleaned]"
        echo -e "\t[-pd \t# specify input data as plaid data to be cleaned]"
        echo -e "\t[-o \t# output destination]"
        echo -e "\t[-P \t# specify number of processes]"
        echo "Created By James Behnke. Last Edited January 7 2021"
        ;;

    -conv) convert=1
        ;;
    -pd)dataFamily="pd"
        clean=1
        ;;
    -cs) dataFamily="cs"
        clean=1
        ;;
    -P) numProcesses="$2"
        shift
        ;;
    -if) file="$2"
        shift
        ;;
    -id) inputDirectory="$2"
         shift
         ;;
    -o) destinationFolder="$2"
         shift
         ;;
    --)
        shift
        break
        ;;

    *) echo "Option $1 not recognized" ;;
    esac
    shift
done

if [ "$availableFiles" -gt 0 ]; then

case "$convert" in

0) case "$dataFamily" in
```

```
106
107     pd) checkDirExist "./JSONtoCSV/"
108         eval "$convertOnly"
109         ;;
110
111     cs) checkDirExist "./JSONtoCSV/"
112         eval "$convertOnly"
113         ;;
114     esac
115     ;;
116 1) case "$dataFamily" in
117
118     pd) checkDirExist "./originalFiles/"
119         checkDirExist "./minified/"
120         find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'cp '{}' ./originalFiles/'{}'; sed -i -r "s/(\".*\")|\s*/\1/g" '{}'; cat '{}' | tr -d "\n" > ./
    minified/'{}'; sed -i "s/}},/}},\n/g;s/.*\[{/{/;s/\]}}//g" ./minified/'{}'; rm '{}' '
121         cd ./minified/
122         checkDirExist "./plaid_converted/"
123         find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'jsonexport '{}' ./plaid_converted/'{}.csv'; echo "Finished trimming '{}'"'
124         cd ./plaid_converted/
125         checkDirExist "./cleaned/"
126         find . -maxdepth 1 -type f -name "*.csv" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'Rscript ../../scripts/monthPlaidData.R '{}' ./cleaned/'{}' &>> log.txt; echo "Converted and
    Cleaned '{}'" '
127         cd ../..
128         checkDirExist "$destinationFolder"
129         mv ./minified/ "$destinationFolder"
130         ;;
131
132     cs) checkDirExist "./originalFiles/"
133         checkDirExist "./minified/"
134         find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'cp '{}' ./originalFiles/'{}'; sed -i -r "s/(\".*\")|\s*/\1/g" '{}'; cat '{}' | tr -d "\n" > ./
    minified/'{}'; sed -i "s/}},/}},\n/g" ./minified/'{}'; rm '{}''
135         cd ./minified/
136         checkDirExist "./client_stats_converted/"
137         find . -maxdepth 1 -type f -name "*.json" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'jsonexport '{}' ./client_stats_converted/'{}.csv'; echo "Finished trimming '{}'"'
138         cd ./client_stats_converted/
139         checkDirExist "./cleaned/"
140         checkDirExist "./cleaned_All/"
141         find . -maxdepth 1 -type f -name "*.csv" -print0 | xargs -0 -I '{}' -P"$numProcesses" sh
    -c 'Rscript ../../scripts/client_Stats.R '{}' ./cleaned/'{}' &>> log.txt; echo "Converted and
    Cleaned '{}'" '
142         cd ../..
143         checkDirExist "$destinationFolder"
144         mv ./minified/ "$destinationFolder"
145         ;;
146
147     esac
148     ;;
149
150 esac
151
```

```
else
    echo "Error, there are no files to convert. Please add json files to this directory"
fi
```

11.2 2.2 Plaid Data Transformation

```
# Author: James Behnke
# Purpose: Transform raw WTFast monthly Plaid data into cleaned data for further processing. This script
primarily cleans
# Usage: This script runs part of the GPerfCleaner bash script which automatically runs this R script on
Plaid data when -pd parameters are given
# License: MIT License

# Copyright (c) [2021] [James Behnke]

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.


args<-commandArgs(TRUE)
file <- args[1]
fileExportLoc <- (args[2])


toDateCols <- c('game_session_start',
 'game_session_stop',
 'load_time',
 'stats_start',
 'stats_stop')

'%ni%' <- Negate('%in%')


df <- data.frame(read.csv(file))

#column names were given prepended strings as part of the JSON conversion
colnames(df) <- gsub("_source.", "", colnames(df))
colnames(df) <- gsub("X", "", colnames(df))

#convert any UNIX timestamps that are stored as Strings to numbers
#For each file, change the appropriate character columns to equivalent numeric columns
```

```
df[,c(toDateCols)] <- lapply((df[,c(toDateCols)]), as.numeric)

#convert UNIX timestamp (including milliseconds to Date eg: 1596259808483 to 2020-08-01 05:30:08 UTC)
for (i in toDateCols) {
 df[,c(i)] <- as.POSIXct(((df[,(c(i))])/1000),tz="UTC", origin="1970-01-01")
}

df <- df[df$'_index' %ni% "",]

#write a csv file of the same name ending in CSV
write.csv(df, file=fileExportLoc, na="", row.names = FALSE)
```

Figure 6 *This script is responsible for cleaning any messy data that was added during conversion and converting UNIX timestamps into human-readable UTC formatted timestamps*

11.3 2.3 Client Stats Transformation:

```
# Author: James Behnke
# Purpose: Transform raw WTFast monthly Client Stats data into cleaned data for further processing. This
script primarily cleans
# Usage: This script runs as part of the GPerfCleaner bash script which automatically runs this R script on
monthly client stats data when -cs parameter is given.
# License: MIT License

# Copyright (c) [2021] [James Behnke]

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

# Begin
args<-commandArgs(TRUE)
file <- args[1]
setwd(args[2])
fileExportLoc <- (args[2])

'%ni%' <- Negate('%in%')

colsToFix <- c('hits.',
 '_source.',
 'internet.',
 'wtfast.',
 'trace_data.',
 'hops.')

 df <- read.csv(file)
 for (title in colsToFix) {
 colnames(df) <- gsub(title, "", colnames(df))
 }
 colnames(df) <- gsub(".timestamp", "timestamp", colnames(df))
 #column names were given duplicates such as ("hits__hits___x")
 #this removes redundant columns 1-7 such as "took, timed_out etc"
```

```
notNull <- df[df$'_index' %ni% "",] #this is a very important line, it checks the first column to see if
it contains relevant info, if it doesnt, its           removed

write.csv(notNull[-c(1:7)], file =fileExportLoc, na="", row.names = FALSE)
```

Figure 6.1 *This script is responsible for cleaning any messy data that was added during conversion removing redundant rows of NA data*

11.3.1 2.4 Combine X Files to Separate Subdirectories

This script is useful for client stats data dumps as there can be many tens of thousands of files to be processed, and inserting these into a database can be problematic and time-consuming. This script will divide them into folders of 2500 files. From there, files can be easily and efficiently combined into a single file using the next tool.

Filename: moveToLargeFolders (Bash file, no extension)

```
# Author: James Behnke
# Purpose: Combine folder containing X number of files into smaller folders containing X desired files.
# Example: Monthly client stats data dumps commonly contain 50-60,000 files. Often it is helpful to combine
all these many small files into a single larger file, but you cannot necessarily combine 50-60,000 files so
this tool divides that number into 2500 files per Large file. This additionally makes uploading to database
easier.
# Usage: This script
# License: MIT License

# Copyright (c) [2021] [James Behnke]

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

# Begin

#!/bin/bash
COUNT=1
folderNum=1
filesPerFolder=2500 #change this to customize how many files per divided folder
mkdir $folderNum
for f in *.csv
do
((COUNT++))
  echo "Processing $f file..."
  if [[ "$COUNT" -gt "$filesPerFolder" ]];
  then
    ((folderNum++))
    mkdir $folderNum
    COUNT=0
  fi
  # $f value stores current file name. move file from large folder to smaller folder
  mv $f ./$folderNum/$f
done
```

Figure 7 *The moveToLargeFolders bash script*

11.4 2.5 Combine All Files in a Directory

If using this tool for Client stats, it is strongly suggested to run the above tool first as most computers won't want to handle 2GB+ CSV files. If more than ~750MB of files are combined into one, it will cause difficulty to open to view in Excel or other programs.

Filename: combineAllCSV.R

```
# Author: James Behnke
# Purpose: Combine folder containing X number of CSV files into one single CSV file
# Example: Monthly client stats data dumps commonly contain 50-60,000 files. Often it is helpful to combine
all these many small files into a single larger file. If using this tool for client stats, it is HIGHLY
recommended to run the above moveToLargeFolders script first. If using for Plaid data, all is well!
# Usage: This script takes in a directory path and combines all files ending with the .CSV extension and
combines them into a single file.
# License: MIT License

# Copyright (c) [2021] [James Behnke]

# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:

# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.

# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.

# Begin



if (!require(plyr)) install.packages('plyr')
library(plyr)

args<-commandArgs(TRUE)
fileDir <- args[1]
filenames <- list.files(fileDir, pattern = '.csv')
dataset <- ldply(filenames, read.csv, header=TRUE)
dataset <- dataset[!apply(is.na(dataset) | dataset == "", 1, all),]

write.csv(dataset, file="Combined_CSV.csv", na="", row.names = FALSE)
```

Figure 7.1 *The combineAllCSV.R file*