DATA INCUBATOR: CAPSTONE

Healthcare prediction for Insurance

by

KJ BENJAMIN

Project Summary

For this project, I want to look at connections between health habits, occupation and age, to determine cost of healthcare as a predictor for insurance premiums. The initial analyze will look at:

1. Behavioral Risk Factor (behavior)
2. Big Cities Health (socioeconomic)
3. ACA (Insurance layout)
4. State performance (access & quality)

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURES Page

LIST OF TABLES

TABLES Page

1. Datasets

Website for data import, wrangling and visualization in R: computerworld.

1.1 Healthcare

Key website for health data: HSRIC.

1.1.1 Behavioral Risk Factor

The data can be downloaded as csv file:

1. life-quality. Health related quality of life survey.
2. nutrition. Very recent data.
3. indicators. This is a zipped file.

Note: There is metadata (json) for the above datasets.

I'm not going to want to save this data to my hard drive, at least not initially. So I can use R packages to download directly from the website.

Example:

```
library(data.table)
mydat <- fread("https://website.url.here")
```

1.1.2 Healthcare Spending

1. raw data. This is in json.
2. ACA. This is xlsx file.
3. state performance. This is also xlsx file.

1.2 Video Gaming

Changing topics to video gaming, but don't want to lose the information above.

I first download the available excel file for game rating, and convert to csv. Next, I need the sales data for these games.

Below are the datasets including game ratings and unit sales. Furthermore, I have additional datasets for franchise information and best-selling video game based on money earned and not units sold. This is a significantly smaller dataset.

1. game rankings
2. game database
3. xlsx game ranking
4. vg wiki
   (a) highest franchises
   (b) best-selling vg
   (c) most expensive vg
   (d) pokemon
   (e) sonic the hedgehog
   (f) mario
   (g) gta: grand theft auto
   (h) halo

2. Data manipulation

2.1 Units sold & ratings

The first thing I will do is use the vgcharts website to download units sold information. This along with the excel file with game ratings can be used to compare sales (units) and game rating[1].

The `project.R` script first section details R parts of the analysis. `sed` and `grep` were used to edit information.

Example:

```
grep "PlayStation" ratings.tsv | head
sed -i 's/PlayStation 4/PS4/g' ratings.tsv
```

Emacs was used to edit the much smaller NS.rating.tsv.

```
rm(list = ls()) ## Clear workspace
library(XML) ## Load XML library


ratings <- read.csv("capstone/research/gamedata.csv")
write.table(ratings, file="capstone/research/ratings.tsv", sep="\t",
            append = FALSE, row.names = FALSE, col.names = TRUE)
```

Here, I would edit the ratings file with `sed` and `grep`.

```
r.ns <- readHTMLTable(paste("http://www.gamerankings.com/",
                            "sites/1188-ign/index.html?platform=124", sep=""),
                      stringsAsFactors = FALSE)
names(r.ns) <- "NS"
write.table(r.ns, file = "capstone/research/NS.ratings.tsv", sep="\t",
            append = FALSE, row.names = FALSE, col.names = TRUE)
```

Here, I used Emacs to edit the NS (Nintendo Switch) file.

```
## Download Game Sales from website
u1 <- readHTMLTable(paste("http://www.vgchartz.com/gamedb/?name=&publisher=",
```

[1]NS - Nintendo Switch, console games is not included in the game rating, so I have add them from the IGN rating website (genre unknown).

```
                        "&platform=&genre=&minSales=0&results=1000", sep=""),
                    stringsAsFactors = FALSE)
names(u1) <- c("Header", "Space", "Table", "Void")
game.sales <- u1$Table

for (i in 2:158){
    u <- paste("http://www.vgchartz.com/gamedb/?page=",i,
               "&results=1000&name=&platform=&minSales=0&publisher=&genre=&sort=GL",
               sep="")
    tmp.page <- readHTMLTable(u, stringsAsFactors = FALSE)
    if (length(names(tmp.page))==4){
        names(tmp.page) <- c("Header", "Space", "Table", "Void")
        game.sales <- rbind(game.sales, tmp.page$Table)
    }
    else next
}

game.sales$"North America" <- as.numeric(game.sales$"North America")
game.sales$"Europe"        <- as.numeric(game.sales$"Europe")
game.sales$"Japan"         <- as.numeric(game.sales$"Japan")
game.sales$"Rest of World" <- as.numeric(game.sales$"Rest of World")
game.sales$"Global"        <- as.numeric(game.sales$"Global")

save(game.sales, file="capstone/research/vgchart_gamesales.rda") ## Save original
```

Now that the two datasets have been downloaded, my next step is to merge them. I spent time changing Platform names (i.e. Nintendo Switch to NS and PlayStation 4 to PS4). As such, I will use both Game and Platform for my overlapping `merge`.

```
## Individual game metrics
load("capstone/research/vgchart_gamesales.rda")
## game.sales$Year <- as.numeric(game.sales$Year)
game.sales[game.sales==0]  <- NA
game.sales                 <- game.sales[complete.cases(game.sales),]

ratings     <- read.delim("capstone/research/ratings.tsv", stringsAsFactors = F)
r.ns        <- read.delim("capstone/research/NS.ratings.tsv", stringsAsFactors = F)
```

```
r.ns$Score  <- as.numeric(r.ns$Score)
r.ns        <- r.ns[,c(1,3,4,2)]
new.ratings <- rbind(ratings, r.ns)

sales.rating <- merge(new.ratings, game.sales, by = c("Game", "Platform"))
sales.rating <- subset(sales.rating, select=-Genre.x)
```

After giving the data a quick look with `ggplot2`, I realize I also want to do the following:

1. Remove outliers from statistical analysis techniques like ANOVA.
2. Separate out multi-platform games
3. Separate out console developers
    - Atari
    - Microsoft
    - Nintendo
    - SEGA
    - Sony
4. Separate out Platforms by generation

2.1.1 Removing outliers

There is a package "`outliers`" that has the function, `rm.outlier`.

This did not do what I wanted it to do. I'm not sure if I should remove the outliers at all now. Maybe too complicated to figure out right now. I will work around the outliers. If I must, I can use the franchise dataset instead of this one, or remove by hand outliers. I'm just not sure it will make a whole lot of difference to the statistical analysis.

2.1.2 Multi-platform

Here, I'd like to add a column with multi-platform information (yes, no). I need a code that looks to see if a game was released in more than one platform.

- loop (for-loop) using length of Game
- reorder dataset via game title
- if game title repeats, this is a multi-platform game
    - compare n and n+1; n1==n2
    - if they equal, then mark as yes
    - but I would need to mark for both, or check up and down; n2==n1 or n2==n3

    - will need special looks at the beginning and end (n1, n)
- make as own vector, then add vector to dataset as new column

First I check to see if the data is sorted.

```
tmp <- sales.rating$Game
tmp <- as.character(tmp)
is.unsorted(tmp) ## FALSE
```

A false means that the data is sorted, so I can move on without reorganizing the dataset[2].

```
## Multi-platform check
mp <- vector()
for (i in 1:length(sales.rating$Game)){
    a = i
    b = i + 1
    c = i - 1
    if (a=="1"){
        tmp1 <- sales.rating$Game[a] == sales.rating$Game[b]
        mp <- c(mp, tmp1)
    }
    else if (a==length(sales.rating$Game)){
        tmp2 <- sales.rating$Game[a] == sales.rating$Game[c]
        mp <- c(mp, tmp2)
    }
    else {
        tmp3 <- (sales.rating$Game[a] == sales.rating$Game[b]) |
            (sales.rating$Game[a] == sales.rating$Game[c])
        mp <- c(mp, tmp3)
    }
}
mp[mp==TRUE]  <- "Yes"
mp[mp==FALSE] <- "No"
sales.rating[["Multi.Platform"]] <- mp
```

This gives me a yes/no column that I can use to subset the data later into multi-platform and single-platform release.

[2]It should be noted that I needed to transform the data into *character*, otherwise `is.unsorted` comes back TRUE. This seems to be an issue because didn't import data with `stringsAsFactors` set.

2.1.3 Console Developers

Now, I want to separate out the developers, or really group the games into 5 overall companies. First, I checked to see which platforms are in the dataset. There are 21 unique platforms:

- **Atari:** None
- **Microsoft**
    1. XB
    2. X360
    3. XOne
- **Nintendo**
    1. NES
    2. SNES
    3. N64
    4. GC
    5. Wii
    6. WiiU
    7. NS
    8. GB
    9. GBA
    10. DS
    11. 3DS
- **SEGA:** DC
- **Sony**
    1. PS
    2. PS2
    3. PS3
    4. PS4
    5. PSP
    6. PSV

I can use `grep` functions within R, `gsub` to match and replace the above[3]. On second thought, I do not want to replace the Platform, but add a new column that uses the platform names to add the developer. I would export the Platform as vector, then replace that one, and add it back as a new column. Nevermind, `gsub` does not overwrite the file; however, I keep

[3]Base package

overwriting the file.

As I want to run in a loop, I need to figure out how to use with partial matches. The best idea I can come up with, without using the command line, is to creat two platform vectors: 1) unique strings, 2) the rest. This would mean removing: PS, GB, NES, DS, and Wii.

```
## Developers
dev.names1 <- c("Nintendo", "SEGA", rep("Nintendo", 4),
                rep("Sony",5), rep("Nintendo", 2), rep("Microsoft", 3))
dev.names2 <- c(rep("Nintendo", 3), "Sony", "Nintendo")
developer <- sales.rating$Platform
plat.names1 <- sort(unique(developer))[-c(3,4,8,10,17)]
plat.names2 <- sort(unique(developer))[c(3,4,8,10,17)]

for (i in 1:length(plat.names1)){
    developer <- gsub(plat.names1[i], dev.names1[i], developer)
}

for (i in 1:length(plat.names2)){
    developer <- gsub(plat.names2[i], dev.names2[i], developer)
}

sales.rating[["Developer"]] <- developer
```

2.1.4 VG Generation

There are roughly 8 generations of console video games with the Nintendo Switch a Gen 9 release. Below is where my overlapping Platforms lay:

1. **Gen 1:** None
2. **Gen 2:** None
3. **Gen 3**
    - NES
4. **Gen 4**
    - SNES
5. **Gen 5**
    - GB
    - N64

- PS

6. **Gen 6**
    - DC
    - GBA
    - GC
    - PS2
    - XB
7. **Gen 7**
    - DS
    - PS3
    - PSP
    - Wii
    - X360
8. **Gen 8**
    - 3DS
    - PS4
    - PSV
    - WiiU
    - XOne
9. **Gen 9**
    - NS

```
## Console Generations
gen.names1 <- gen.names1 <- c("Gen 8", rep("Gen 6", 3), "Gen 5",
                              "Gen 9", "Gen 6", "Gen 7", "Gen 8",
                              "Gen 7", "Gen 8", "Gen 4", "Gen 8",
                              "Gen 7", "Gen 6", "Gen 8")
gen.names2 <- c("Gen 7", "Gen 5", "Gen 3", "Gen 5", "Gen 7")
generation <- sales.rating$Platform

for (i in 1:length(plat.names1)){
    generation <- gsub(plat.names1[i], gen.names1[i], generation)
}

for (i in 1:length(plat.names2)){
    generation <- gsub(plat.names2[i], gen.names2[i], generation)
```

```
}

sales.rating[["Generation"]] <- generation
```

This uses the `developer` defined platform names. As such, I can combine the two `for-loops`.

```
## Developers & Console Generations
dev.names1  <- c("Nintendo", "SEGA", rep("Nintendo", 4),
                 rep("Sony",5), rep("Nintendo", 2), rep("Microsoft", 3))
dev.names2  <- c(rep("Nintendo", 3), "Sony", "Nintendo")
developer   <- sales.rating$Platform

gen.names1  <- gen.names1 <- c("Gen 8", rep("Gen 6", 3), "Gen 5",
                               "Gen 9", "Gen 6", "Gen 7", "Gen 8",
                               "Gen 7", "Gen 8", "Gen 4", "Gen 8",
                               "Gen 7", "Gen 6", "Gen 8")
gen.names2  <- c("Gen 7", "Gen 5", "Gen 3", "Gen 5", "Gen 7")
generation  <- sales.rating$Platform

plat.names1 <- sort(unique(developer))[-c(3,4,8,10,17)]
plat.names2 <- sort(unique(developer))[c(3,4,8,10,17)]

for (i in 1:length(plat.names1)){
    developer  <- gsub(plat.names1[i], dev.names1[i], developer)
    generation <- gsub(plat.names1[i], gen.names1[i], generation)
}

for (i in 1:length(plat.names2)){
    developer  <- gsub(plat.names2[i], dev.names2[i], developer)
    generation <- gsub(plat.names2[i], gen.names2[i], generation)
}

sales.rating[["Developer"]]  <- developer
sales.rating[["Generation"]] <- generation
```

2.2 Franchising

2.2.1 Pokemon

3. Analysis

3.1 Units sold & reviews

3.1.1 Statistics

3.1.2 Interactive graphs

3.2 Franchising