VIDEO GAMES - Test analítica

Tatán Rufino

14/10/2019

Bootstrapping

En primer lugar vamos a cargar los paquetes necesarios

```
library(readr)        #Para leer e importar datos
library(dplyr)        #Análisis exploratorio
library(ggplot2)      #Visualizaciones
library(tidyverse)    #Utilizada para ordenar dataset
library(gsheet)       #Leer google sheets
library(readxl)
library(knitr)        #Varios propósitos
library(DT)           #libreria DataTables
library(caret)        #Variables para Correlation Matrix
library(grid)         #Plotear
library(gridExtra)    #Plotear
library(corrr)        #Var estadisticas
library(ggpubr)
library(qcc)
library(data.table)
library(janitor)
library(arules)
library(arulesViz)
library(lubridate)
library(stats)
library(samplingbook)
library(rmarkdown)
```

Posteriormente, procedemos a la apertura de los ficheros de eventos y los cargamos como un datatable csv.

```
DT <- read.csv("event_levels.csv")
```

Análisis exploratorio

En esta fase, vamos a explorar las posibilidades de nuestro dataset. Para ello, empezaremos calculando algunos totales.

```
# Numero total de usuarios que han participado en el evento
Num_Usr = as.integer(DT %>% distinct(user_id, .keep_all = TRUE)
                     %>% summarise(n=n()))
# Numero total de entradas
Num_Tot = as.integer(DT %>% summarise(n=n()))

Num_Suc = as.integer(DT %>%  filter(level==5,action=="action.success") %>% #
```

```
                              distinct(user_id, .keep_all = TRUE)
                                  %>% summarise(n=n()))
# Numero de veces que se usa un powerup
Num_PowT =as.integer(DT %>% filter(action=="action.use_powerup") %>%
        summarise(n=n()))
# Recuento de Starts
Num_Start = as.integer(DT %>% filter(action=="action.game_start")
                       %>% summarise(n=n()))
# Recuento de Fails
Num_Fail = as.integer(DT %>% filter(action=="action.fail")
                      %>% summarise(n=n()))
# Recuento de Success totales
Num_SucT = as.integer(DT %>% filter(action=="action.success")
                      %>% summarise(n=n()))
```

Ejercicio 1:

Procedamos con la primera pregunta: ¿Cuál es el porcentaje de usuarios que han completado el evento respecto de los que han participado?

```
SOL_SUCC =(Num_Suc/Num_Usr)*100

SOL_SUCC
```

```
## [1] 17.59277
```

Como podemos ver, el **17,5927% de los usuarios** lo ha superado completamente.

Análisis de datos

```
# Relacion de jugadores que ha empezado
Num_Start/Num_Usr
```

```
## [1] 13.33696
```

```
# Relacion de success totales
    Num_SucT/Num_Usr
```

```
## [1] 6.998154
```

```
# Relacion de fails por usuario
Num_Fail/Num_Usr
```

```
## [1] 5.180065
```

```
RATE_TABLE = S2 =  DT %>% group_by(level) %>%
            mutate(Total=n(),
```

```
                  Start=ifelse(action=="action.game_start",1,0),
                  Success=ifelse(action=="action.success",1,0),
                  Fail=ifelse(action=="action.fail",1,0)) %>%
           summarise(RATE_START=sum(Start)/Num_Usr,
                     RATE_SUCC=sum(Success)/Num_Usr,
                     RATE_FAIL=sum(Fail)/Num_Usr)
```

```
summary(RATE_TABLE)
```

```
##      level     RATE_START       RATE_SUCC        RATE_FAIL
##  Min.   :1   Min.   :0.2731   Min.   :0.1764   Min.   :0.06429
##  1st Qu.:2   1st Qu.:0.7737   1st Qu.:0.2661   1st Qu.:0.37152
##  Median :3   Median :1.8700   Median :0.7652   Median :0.92871
##  Mean   :3   Mean   :2.6674   Mean   :1.3996   Mean   :1.03601
##  3rd Qu.:4   3rd Qu.:3.9725   3rd Qu.:1.8519   3rd Qu.:1.78264
##  Max.   :5   Max.   :6.4477   Max.   :3.9386   Max.   :2.03291
```

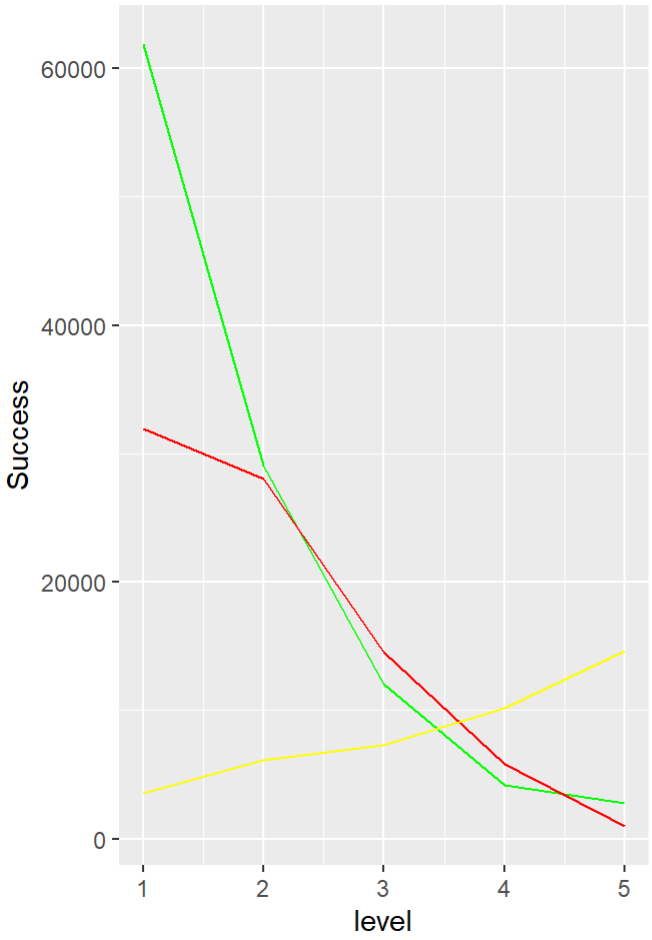
En una primera visual vemos que el ratio de success bruto y por level es mayor que el de fails.

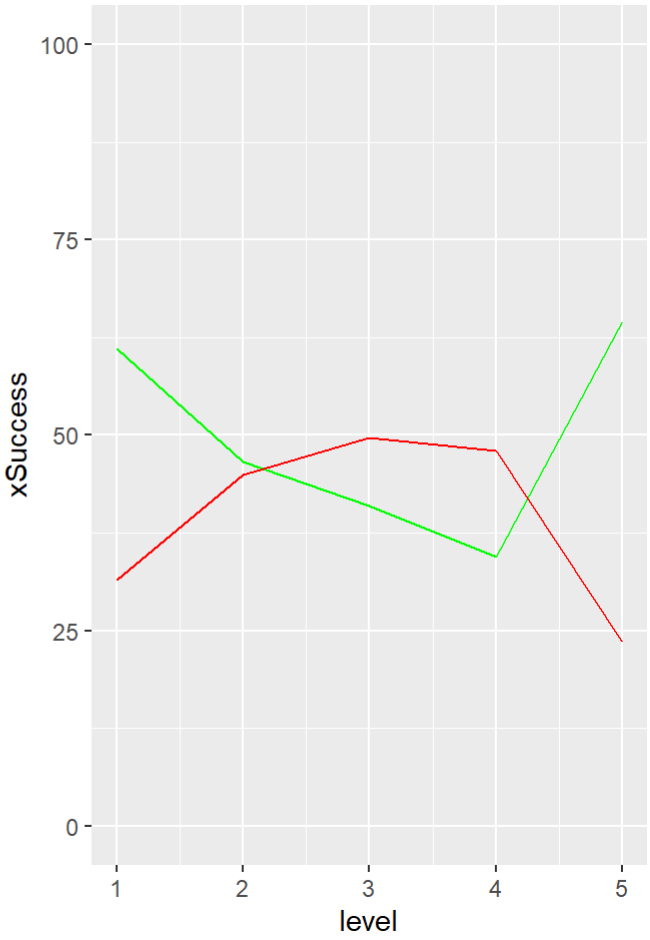
Solo el fail tiene una mediana mayor, lo que suguiere más equilibrio y menos dependencia de factores externos.

```
S2 =  DT %>% group_by(level) %>%
        mutate(Total=n(),
               Start=ifelse(action=="action.game_start",1,0),
               Success=ifelse(action=="action.success",1,0),
               Fail=ifelse(action=="action.fail",1,0),
               PowerUp=ifelse(action=="action.use_powerup",1,0)) %>%
        summarise(Success=sum(Success),
                  Fail=sum(Fail),
                  PowerUp=sum(PowerUp),
                  xSuccess=sum(Success)/sum(Start)*100,
                  xFail=sum(Fail)/sum(Start)*100)# %>%
        #adorn_totals("row")
```

Sacamos la gráfica de la tabla para que podamos analizar a fondo las evoluciones relativas y netas de cantidades:

```
# Evolución de cantidades netas

S2gA <- ggplot(S2, aes(x=level)) +
        geom_line(aes(y=Success), colour="green") +
        geom_line(aes(y=Fail), colour="red") +
        geom_line(aes(y=PowerUp), colour="yellow")

# Evolución de cantidades relativas

S2gB <-ggplot(S2, aes(level)) +
        geom_line(aes(y=xSuccess), colour="green") +
        geom_line(aes(y=xFail), colour="red") +
        ylim(0,100)

ggarrange(S2gA,S2gB)
```





En la grafica S2gA se observa una caída decreciente de la cantidad de partidas ganadas y perdidas (basándonos en la proporción relativa a partidas comenzadas), las cuales difieren principalmente en el primer nivel y después se ajustan hasta encontrarse en valores similares, lo que sugiere equidad.

Su decrecimiento es relativamente lineal (S2.R2=0.84). Una baja dispersión sugiere alta fiabilidad.

```
lm(Success~level,data=S2)
```

```
##
## Call:
## lm(formula = Success ~ level, data = S2)
##
## Coefficients:
## (Intercept)        level
##       64929       -14313
```

```
S2.R2 <- summary(lm(Success~level,data=S2))$r.squared
```

En la gráfica S2gB se puede confirmar que existe cierta equidad en la proporción de éxito y fallo durante los niveles intermedios, lo que sugiere un juego asequible.

La caida de las proporciones de fallo en el último nivel sugiere que los powerups acumulados durante todo el evento son usados principalmente en el nivel 5 del evento.

Análisis de Dificultad:

Para este análisis, vamos a calcular la relación entre el *número de fallos y los success* de nuestros jugadores. Esta medida nos permitirá ver el grado de dificultad al comparar los niveles visualmente en una gráfica, ya que, aquellos niveles que despunten (los conocidos *outliers*) deberán ser objeto de estudio con tal de suavizar las *curvas de dificultad* a los usuarios de nuestro juego.

Nuestro objetivo, es que en los primeros niveles, no haya grandes cuellos de botella que acelerarían en los primeros niveles el **churn rate**; necesitamos por tanto, acercarnos a una curva suave y mantener así la *retención* de los usuarios en un alto porcentaje.

```
SOL_DIF = round(Num_Fail/Num_SucT,2)
SOL_DIF
```

```
## [1] 0.74
```

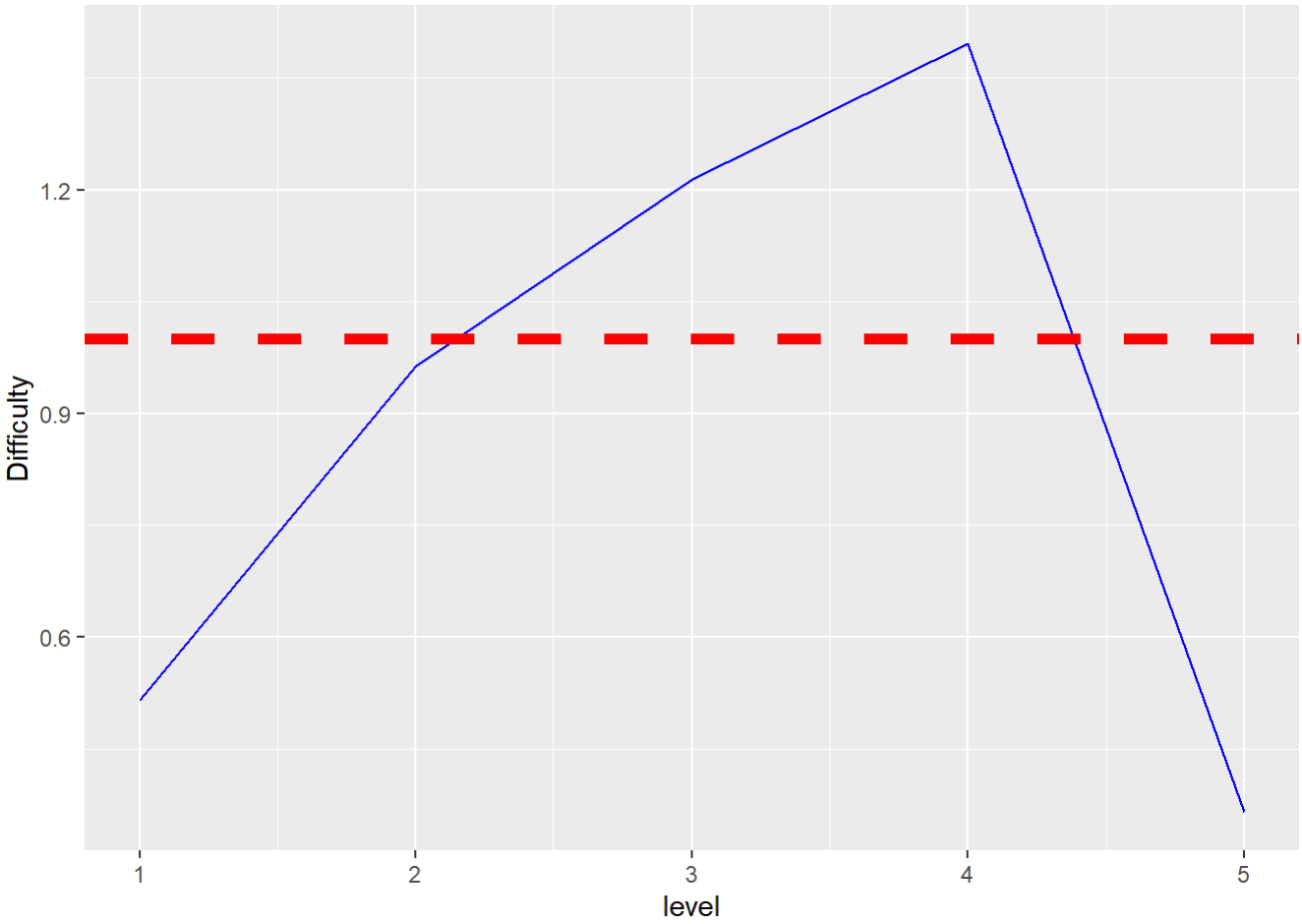
Podemos observar, como ahora mismo tratamos niveles fáciles, ya que nuestra relación se encuentra por debajo de 1. Estas métricas nos sirven en este tipo de juegos que nos ocupan (match 3) para ajustar la dificultad, observando como crece este número podemos asegurarnos de que los usuarios experimentan o no crecidas desmedidas y ajustarlos en sucesivas updates del juego.

```
S3 = as.data.table(
        DT %>% group_by(level) %>%
                mutate(Total=n(),
                       Start=ifelse(action=="action.game_start",1,0),
                       Success=ifelse(action=="action.success",1,0),
                       Fail=ifelse(action=="action.fail",1,0),
                       PowerUp=ifelse(action=="action.use_powerup",1,0)) %>%
                summarise(Success=sum(Success),
                          Difficulty=sum(Fail)/sum(Success)))
```

```
mean(S3$Difficulty)
```

```
## [1] 0.8906032
```

```
ggplot(S3, aes(level))+
        geom_line(aes(y=Difficulty), colour="blue")+
        geom_hline(yintercept=1, linetype="dashed",
                   color = "red", size=2)
```



La dificultad nos sale de **media 0.89**, En este gráfico es muy interesante visualizar la evolución de la dificultad según nivel:

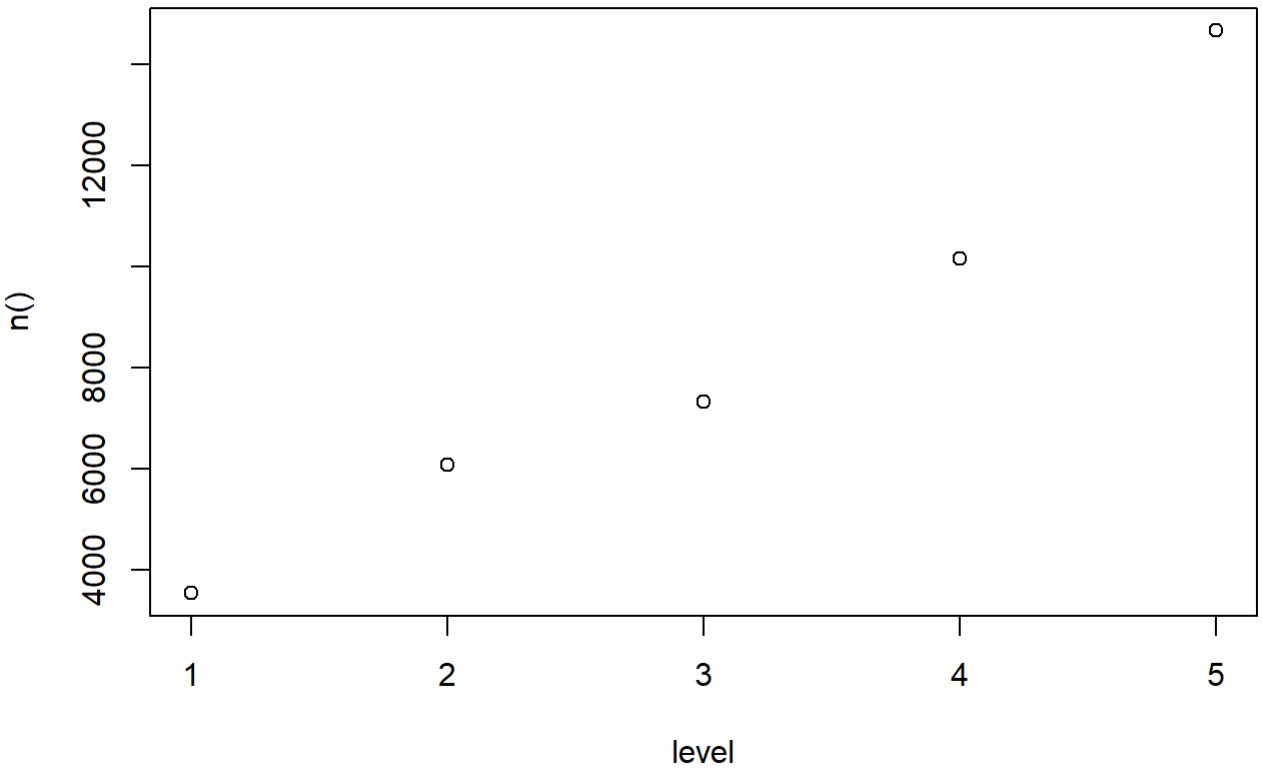
La dificultad del juego se eleva levemente durante los niveles 3 y 4 y cae de nuevo en el nivel 5. Tal como hemos visto con la variable *SOL_DIF*, en estos niveles se trata de algo normal que la dificultad no aumente mucho y se mantenga fácil, pero que deducimos que según avanzamos en niveles los usuarios también exigen una mayor dificultad, por lo que tendríamos que estar pendientes de estás métricas para ajustarlos a sus necesidades.

Por lo general, y como he podido ver en este tipo de juegos (*Candy Crush, Soda*, etc.), es una estrategia común empezar con una curva muy suave al principio; en el *late game*, los jugadores más asiduos se sienten recompensados ante la estrategia 3-7-5; en donde cada 3 niveles encontramos un pico que les cuesta superar, para luego bajar de nuevo y que les parezca que sus habilidades han mejorado, reconfortando a los jugadores y aumentando retención.

Uso de powerups y su relación con la dificultad:

Procedamos aquí a estudiar la relación de dificultad y el uso de powerups:

```
POWERUP_TABLE0 = DT %>%  filter(action=="action.use_powerup") %>% group_by(level) %>% summa
rise(n())

plot(POWERUP_TABLE0)
```



Nos resulta curioso como el uso de powerups aumenta de forma geométrica. Casa con la estrategia comentada del aumento de dificultad en la **progresión de niveles**; los usuarios conforme avanzan harán uso de más powerups cuando se queden *cerca* de pasarse un nivel, por este motivo es necesario hacer análisis de el *número de reintentos* hasta que se lo pasen (success), indicador suficiente para reconocer estrategias de gasto de powerups.

```
SOL_PU = (Num_PowT/Num_Usr)
```

Se han usado **2.66 de powerups** por cada usuario en el evento.

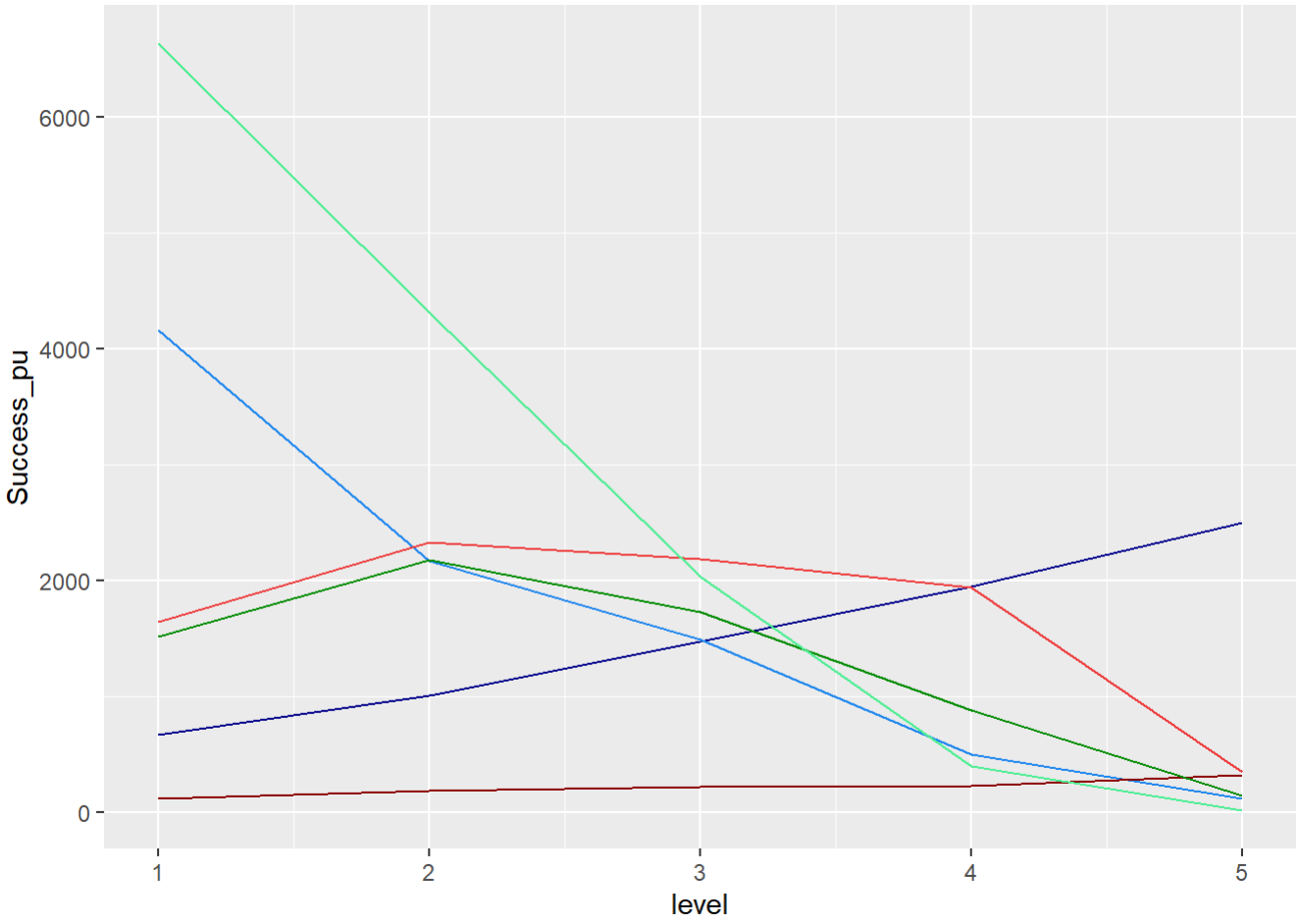
Como no existen variables que asocien directamente el success o el fail con el uso de powerups, crearemos un data table que nos ayude a clasificar ambos casos estudiando y filtrando los casos de coincidencia de variables:

```
POWERUP_TABLE = data.frame(level=c(1),
                           Success_pu=c(1),
                           Success_nopu=c(1),
                           Fail_pu=c(1),
                           Fail_nopu=c(1),
                           SucFail_pu=c(1),
                           SucFail_nopu=c(1))

POWERUP_TABLE = POWERUP_TABLE[-1,]

y=1

for(y in 1:5){

        Success_pu = as.integer(DT %>% filter(level==y) %>%
                        group_by(user_id) %>%
                        mutate(PowerUp=ifelse(action=="action.use_powerup",1,0),
```

```
                                Success=ifelse(action=="action.success",1,0),
                                Fail=ifelse(action=="action.fail",1,0))  %>%
                          summarise(PowerUp=sum(PowerUp),Success=sum(Success),Fail=sum(Fail))
 %>%
                          filter(PowerUp>0,Success>0,Fail<1) %>%
                          summarise(n=n()))

        Success_nopu = as.integer(DT %>% filter(level==y) %>%
                                        group_by(user_id) %>%
                                        mutate(PowerUp=ifelse(action=="action.use_powerup",
1,0),
                                               Success=ifelse(action=="action.success",1,0)
,
                                               Fail=ifelse(action=="action.fail",1,0))  %>%

                                        summarise(PowerUp=sum(PowerUp),Success=sum(Success)
,Fail=sum(Fail)) %>%
                                        filter(PowerUp<1,Success>0,Fail<1) %>%
                                        summarise(n=n()))

        Fail_pu = as.integer(DT %>% filter(level==y) %>%
                                          group_by(user_id) %>%
                                          mutate(PowerUp=ifelse(action=="action.use_powerup
",1,0),
                                                 Success=ifelse(action=="action.success",1,
0),
                                                 Fail=ifelse(action=="action.fail",1,0))  %
>%
                                          summarise(PowerUp=sum(PowerUp),Success=sum(Succes
s),Fail=sum(Fail)) %>%
                                          filter(PowerUp>0,Success<1,Fail>0) %>%
                                          summarise(n=n()))


        Fail_nopu = as.integer(DT %>% filter(level==y) %>%
                                          group_by(user_id) %>%
                                          mutate(PowerUp=ifelse(action=="action.use_powerup
",1,0),
                                                 Success=ifelse(action=="action.success",1,
0),
                                                 Fail=ifelse(action=="action.fail",1,0))  %
>%
                                          summarise(PowerUp=sum(PowerUp),Success=sum(Succes
s),Fail=sum(Fail)) %>%
                                          filter(PowerUp<1,Success<1,Fail>0) %>%
                                          summarise(n=n()))

        SucFail_pu = as.integer(DT %>% filter(level==y) %>%
                                       group_by(user_id) %>%
                                       mutate(PowerUp=ifelse(action=="action.use_powerup",1,0
),
                                              Success=ifelse(action=="action.success",1,0),
                                              Fail=ifelse(action=="action.fail",1,0))  %>%
                                       summarise(PowerUp=sum(PowerUp),Success=sum(Success),Fa
il=sum(Fail)) %>%
                                       filter(PowerUp>0,Success>0,Fail>0) %>%
                                       summarise(n=n()))
        SucFail_nopu = as.integer(DT %>% filter(level==y) %>%
```

```
                                        group_by(user_id) %>%
                                        mutate(PowerUp=ifelse(action=="action.use_powerup",
1,0),
                                               Success=ifelse(action=="action.success",1,0)
,
                                               Fail=ifelse(action=="action.fail",1,0))  %>%

                                        summarise(PowerUp=sum(PowerUp),Success=sum(Success)
,Fail=sum(Fail)) %>%
                                        filter(PowerUp<1,Success>0,Fail>0) %>%
                                        summarise(n=n()))

                H = data.table(t(c(y,Success_pu,Success_nopu,Fail_pu,Fail_nopu,SucFail_pu,S
ucFail_nopu)))

                names(H)<-names(POWERUP_TABLE)

                POWERUP_TABLE = data.table(rbind(POWERUP_TABLE,H))

                #POWERUP_TABLEx = data.table(mapply(c, POWERUP_TABLEx,G))

        }

POWERUP_TABLE = POWERUP_TABLE %>% adorn_totals("row")
POWERUP_TABLE$level = as.integer(POWERUP_TABLE$level) # En ocasiones se genera como factor
```

```
ggplot(POWERUP_TABLE[-6,],aes(x=level))+
        geom_line(aes(y=Success_pu), colour="blue4")+
        geom_line(aes(y=Success_nopu), colour="dodgerblue2")+
        geom_line(aes(y=Fail_pu), colour="red4")+
        geom_line(aes(y=Fail_nopu), colour="brown2")+
        geom_line(aes(y=SucFail_pu), colour="green4")+
        geom_line(aes(y=SucFail_nopu), colour="seagreen2")
```



La única variable que aumenta con el nivel es el Success_pu, es decir, la relación PowerUp - Success es la unica variable con una tasa de crecimiento mínimamente considerable.

Decir que, dadas las circunstancias, obviaremos los datos relativos a las 2 últimas variables, donde coinciden powerups, success y fail, y es difícil suponer la relación que pueden tener directamente entre unas y otras (aunque esto se podría conseguir con un algoritmo bien elaborado)

```
SOL_DIFpu = round(as.integer(POWERUP_TABLE[6,4])/as.integer(POWERUP_TABLE[6,2]),2)


SOL_DIFnopu = round(as.integer(POWERUP_TABLE[6,5])/as.integer(POWERUP_TABLE[6,3]),2)


# Absolutos

c(SOL_DIF=SOL_DIF,SOL_DIFpu=SOL_DIFpu,SOL_DIFnopu=SOL_DIFnopu)
```

```
##     SOL_DIF   SOL_DIFpu SOL_DIFnopu
##        0.74        0.14        1.00
```

Y las medias restantes:

```
summary(POWERUP_TABLE[-6,])
```

```
##      level     Success_pu    Success_nopu     Fail_pu        Fail_nopu
##  Min.   :1   Min.   : 668   Min.   : 114   Min.   :120.0   Min.   : 348
##  1st Qu.:2   1st Qu.:1009   1st Qu.: 494   1st Qu.:185.0   1st Qu.:1643
##  Median :3   Median :1476   Median :1487   Median :219.0   Median :1942
```

```
##  Mean   :3   Mean   :1519   Mean   :1685   Mean   :214.2   Mean   :1690
##  3rd Qu.:4   3rd Qu.:1946   3rd Qu.:2170   3rd Qu.:225.0   3rd Qu.:2190
##  Max.   :5   Max.   :2496   Max.   :4159   Max.   :322.0   Max.   :2329
##    SucFail_pu    SucFail_nopu
##  Min.   : 143   Min.   :  11
##  1st Qu.: 878   1st Qu.: 395
##  Median :1519   Median :2030
##  Mean   :1289   Mean   :2677
##  3rd Qu.:1728   3rd Qu.:4312
##  Max.   :2178   Max.   :6638
```

En este resumen se revela algo muy interesante y es la caida de Fails (en medias) con el uso de powerups y sin ello.

En definitiva, Se observa que la diferencia de dificultad del evento (por niveles) del juego cambia importantemente en el uso de pu, hasta en un 86%, si aceptamos las asunciones hechas hasta el momento, ya que la medida sobre exito/fallo con/sin powerups es aproximada, su uso tiene una incidencia importante.

Vamos llegando a las conclusiones siguientes:

```
POWERUP_TABLE2 = DT %>% group_by(level) %>%
        filter(action=="action.game_start" | action=="action.success" |
                    action=="action.use_powerup" | action=="action.fail") %>%
        mutate(Start=ifelse(action=="action.game_start",1,0),
               Success=ifelse(action=="action.success",1,0),
               Fail=ifelse(action=="action.fail",1,0),
               Powerup=ifelse(action=="action.use_powerup",1,0)) %>%
        summarise(Start=sum(Start),Success=sum(Success),Fail=sum(Fail),
                  Powerup=sum(Powerup),Succ_rate=round(sum(Success)/sum(Start)*100,2),
                  PU_rate=round(sum(Powerup)/sum(Start)*100,2)) %>%
        as.data.table()
```

Apoyándonos en las tablas POWERUP_TABLE 0/2 y el grafico podemos ver el ratio o incremento de uso de PUs en el último nivel, que se corresponde con el incremento de éxito y como elemento fundamental para el éxito del evento completo (subida de más del 300% del primer nivel al último)

Extra: cálculo de KPI's como propuesta basado en el dataset actual

```
ET = DT %>% filter(action=="action.game_start",level==1) %>%
mutate(DAY=day(client_time))

ET = DT %>% filter(action=="action.game_start",level==1) %>%
        mutate(DAY=day(client_time)) %>% group_by(user_id)%>%
        mutate(D1=ifelse(DAY==5,1,0),D2=ifelse(DAY==6,1,0),
               D3=ifelse(DAY==6,1,0),D4=ifelse(DAY==6,1,0)) %>%
        summarise(D1=ifelse(sum(D1)>=1,1,0),
                  D2=ifelse(sum(D2)>=1,1,0),
                  D3=ifelse(sum(D3)>=1,1,0),
                  D4=ifelse(sum(D4)>=1,1,0)) %>%
        mutate(RT1=ifelse((D1+D2+D3+D4)>1,1,0),RT2=ifelse((D1+D2+D3+D4)>2,1,0),RT3=ifelse((
D1+D2+D3+D4)>3,1,0)) %>%
        select(user_id,RT1,RT2,RT3) %>%
    summarise(RTT1=sum(RT1)/n(),RTT2=sum(RT2)/n(),RTT3=sum(RT3)/n())
```

Con esto obtenemos un mapa completo por usuario con retenciones a día 1, 2 y 3, que son el máximo de días de los que disponemos en nuestro muestreo se comprueba que la tasa de retencion a día 1 es superior al 50%, también al día 2, y finalmente cae en el día 3 al 25%

```
ET = DT %>% filter(action=="action.game_start") %>%
        mutate(DAY=day(client_time)) %>%
        group_by(user_id)%>%
        mutate(D1=ifelse(DAY==5,1,0),D2=ifelse(DAY==6,1,0),
               D3=ifelse(DAY==6,1,0),D4=ifelse(DAY==6,1,0)) %>%
        summarise(D1=ifelse(sum(D1)>=1,1,0),
                  D2=ifelse(sum(D2)>=1,1,0),
                  D3=ifelse(sum(D3)>=1,1,0),
                  D4=ifelse(sum(D4)>=1,1,0)) %>%
        mutate(RT1=ifelse((D1+D2+D3+D4)>1,1,0),RT2=ifelse((D1+D2+D3+D4)>2,1,0),RT3=ifelse((
D1+D2+D3+D4)>3,1,0)) %>%
        select(user_id,RT1,RT2,RT3) %>%
    summarise(RTT1=sum(RT1)/n(),RTT2=sum(RT2)/n(),RTT3=sum(RT3)/n())
```

¿Qué haría con datos adicionales?

- Contabilizaría a tiempo real los Success/fail que estén asociados al uso de powerups durante la partida y los que no, así como la cantidad de powerups usados en ellos. La suma de éstos (como hemos visto en la tabla que he calculado llamada POWERUP_TABLE) debería ser igual a la suma total de Success/Fail total, que es la que me habéis proporcionado, de esta forma podríamos valorar mucho mejor la influencia de los pu en la dificultad, en la retención de jugadores y gasto en revenues y compras.
- Me gustaría también poder desgranar los tipos de pu y sus costes en la partida, aunque a rasgos generales no esté mal analizar como si fueran un mismo ente (pese a que se puede jugar mucho en relación al peso económico y la incidencia que tienen)
- Datos adicionales del usuario tipo, como un historial más amplio de su actividad, con la idea de tener trazados sus movimientos

Test A/B:

```
GT <- read.csv("ab_test.csv")

GT=as.data.table(GT)

GT %>% filter(ab_group=="A") %>% summary()
```

```
##                                  user_id       total_game_starts
##  00001591-9A66-45A9-9D6D-EB2C957B7DD0:     1   Min.   :   0.0
##  0000399D-1029-4BF8-84DA-5F268F5401D3:     1   1st Qu.:  15.0
##  00007952-2A50-45F8-94E2-CB949C154908:     1   Median :  64.0
##  0000CEE0-4472-4F32-A2EA-66D52C1D776B:     1   Mean   : 104.6
##  0000FFDD-9EBE-4078-9276-69E47F7F76E9:     1   3rd Qu.: 156.0
##  00010B12-EC5E-4C0E-87AF-F814FE67E960:     1   Max.   :2764.0
##  (Other)                             :381933
##  total_pu_uses     total_revenue      total_product     ab_group
##  Min.   :   0.00   Min.   :  0.0000   Min.   : 0.0000   A:381939
##  1st Qu.:   1.00   1st Qu.:  0.0000   1st Qu.: 0.0000   B:     0
##  Median :   6.00   Median :  0.0000   Median : 0.0000
##  Mean   :  10.76   Mean   :  0.6384   Mean   : 0.1542
##  3rd Qu.:  15.00   3rd Qu.:  0.0000   3rd Qu.: 0.0000
##  Max.   :2045.00   Max.   :684.7800   Max.   :92.0000
```

```
##
```

```
GT %>% filter(ab_group=="B") %>% summary()
```

```
##                                  user_id       total_game_starts
##  00001CFC-7DB1-46E6-BA1B-EF699F13B1C1:     1   Min.   :   0.0
##  00006DA0-80CC-4C2B-A01A-B4873F42122B:     1   1st Qu.:  15.0
##  0000EC54-BF9B-40A2-9698-9EE67B398036:     1   Median :  64.0
##  00014D4A-78F6-41BB-B9CE-B488EA824BCA:     1   Mean   : 104.5
##  000234AD-8AD5-441D-875C-9AC6AA22385A:     1   3rd Qu.: 156.0
##  000296D5-B36D-422B-844E-B00995DFF6A9:     1   Max.   :2678.0
##  (Other)                              :382256
##  total_pu_uses     total_revenue      total_product      ab_group
##  Min.   :   0.00   Min.   :  0.0000   Min.   :  0.0000   A:     0
##  1st Qu.:   1.00   1st Qu.:  0.0000   1st Qu.:  0.0000   B:382262
##  Median :   6.00   Median :  0.0000   Median :  0.0000
##  Mean   :  10.78   Mean   :  0.6593   Mean   :  0.1591
##  3rd Qu.:  15.00   3rd Qu.:  0.0000   3rd Qu.:  0.0000
##  Max.   :4509.00   Max.   :610.7400   Max.   :125.0000
##
```

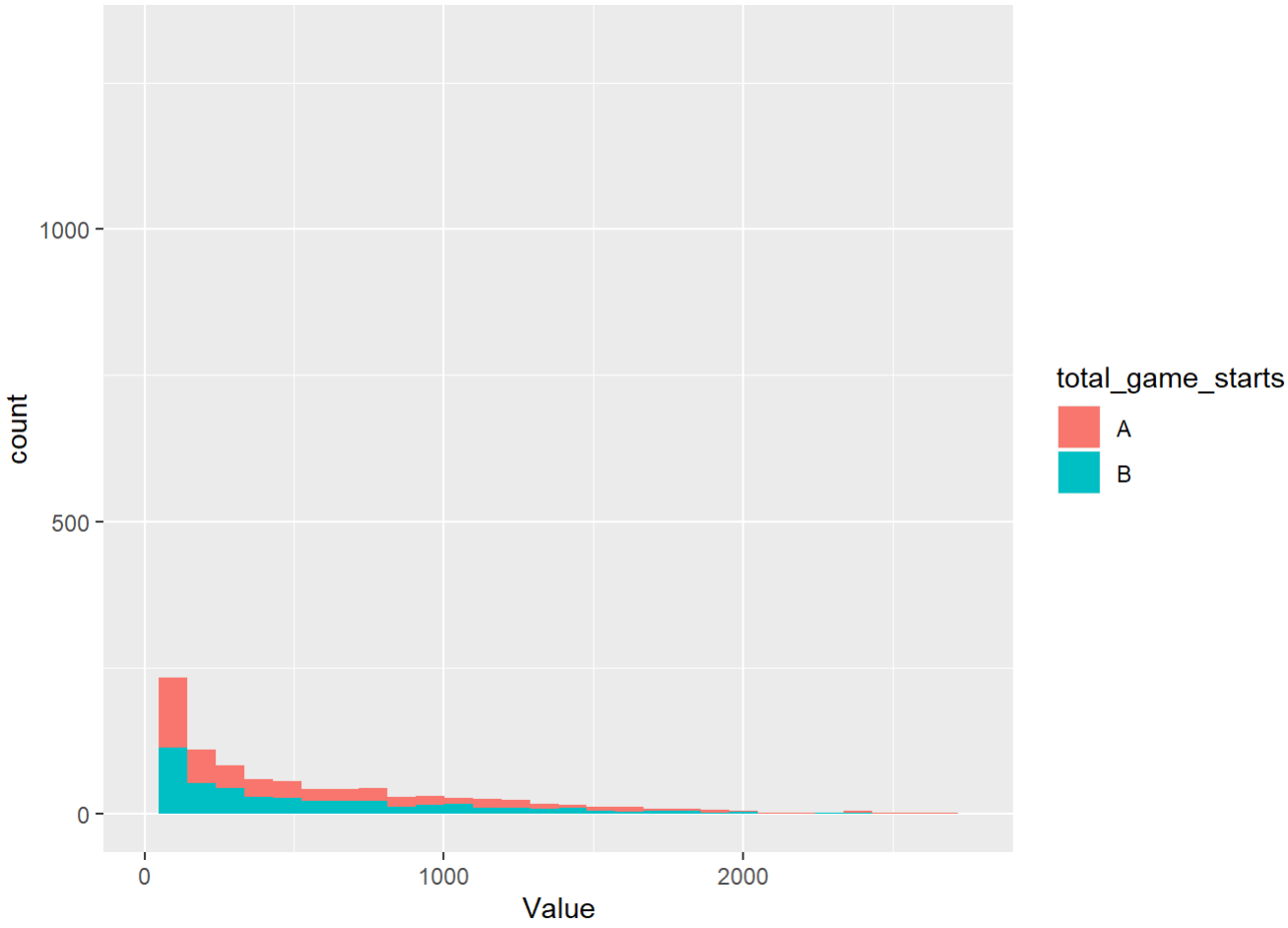
```
GT %>% group_by(ab_group) %>%
    summarise(TOT_STARTS = mean(total_game_starts),
              TOT_PU=mean(total_pu_uses),TOT_REN=mean(total_revenue),
              TOT_PROD=mean(total_product))
```

```
## # A tibble: 2 x 5
##   ab_group TOT_STARTS TOT_PU TOT_REN TOT_PROD
##   <fct>         <dbl>  <dbl>   <dbl>    <dbl>
## 1 A              105.   10.8   0.638    0.154
## 2 B              105.   10.8   0.659    0.159
```

Una primera visual ofrece muy pocas variaciones en los indicadores estándares principales, continuamos el Test A/B, de manera que hacemos el test en estilo overlap para visualizar el histograma, para los diferentes *Rates*.

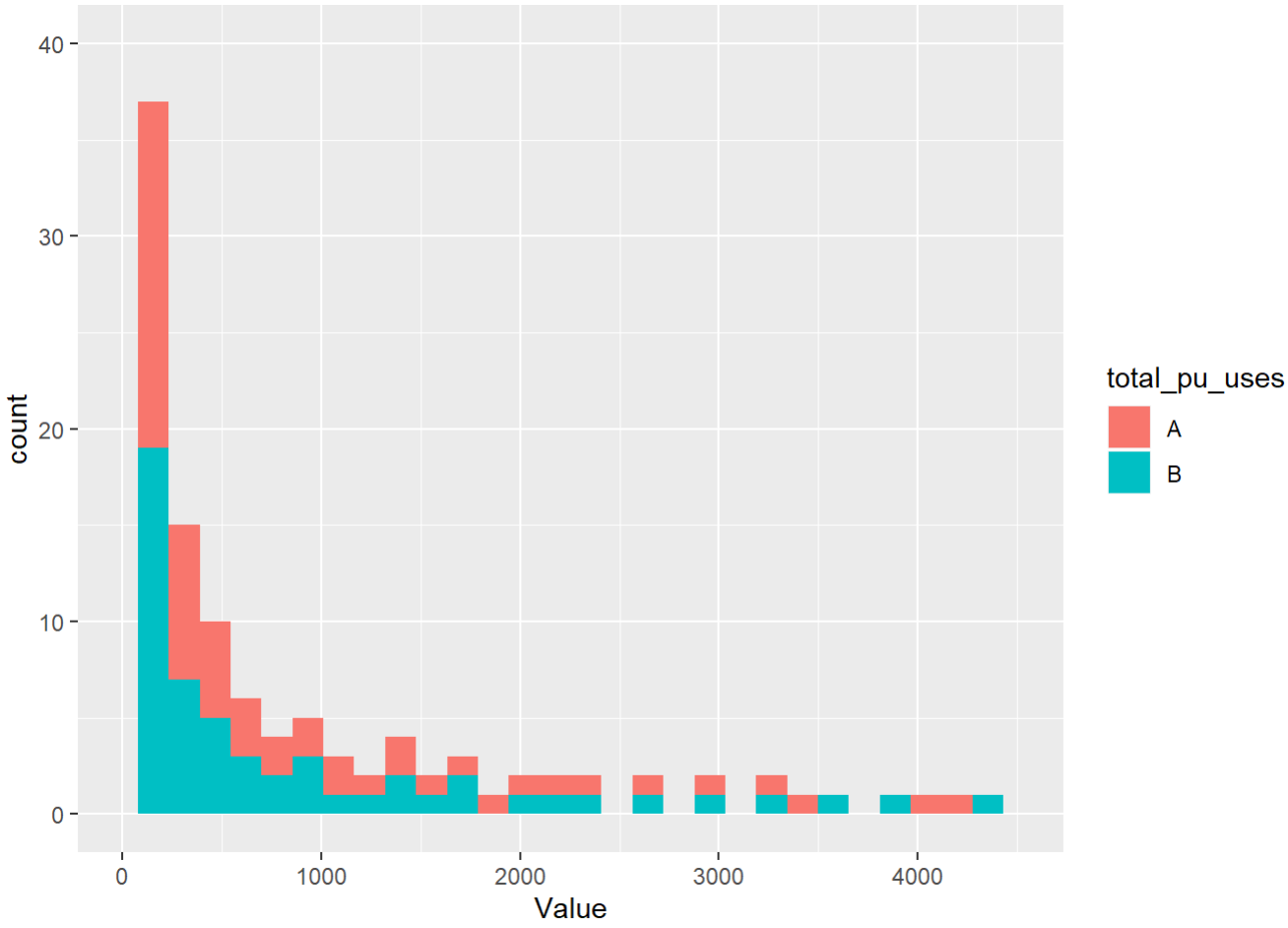
```
START_RATE =  GT %>%
    group_by(total_game_starts) %>%
    mutate(A=ifelse(ab_group=="A",1,0),B=ifelse(ab_group=="B",1,0)) %>%
    summarise(A=sum(A),B=sum(B))

        START_RATE %>%
            gather(key=total_game_starts, value=Value) %>%
            ggplot(aes(x=Value,fill=total_game_starts)) +
            geom_histogram()+
            xlim(0,max(START_RATE$total_game_starts))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



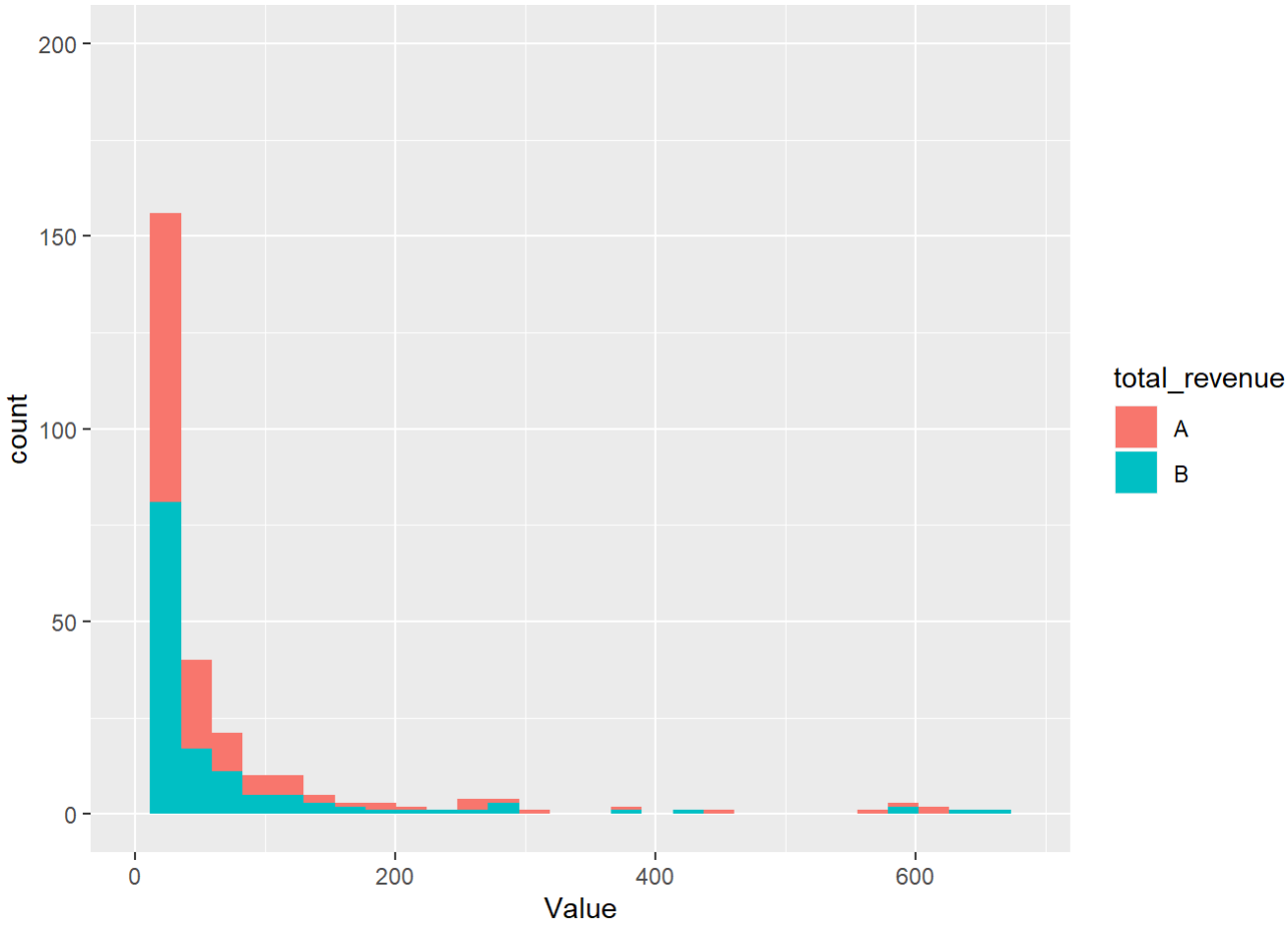

```
PU_RATE =  GT %>%
    group_by(total_pu_uses) %>%
    mutate(A=ifelse(ab_group=="A",1,0),B=ifelse(ab_group=="B",1,0)) %>%
    summarise(A=sum(A),B=sum(B))

        PU_RATE %>%
            gather(key=total_pu_uses, value=Value) %>%
            ggplot(aes(x=Value,fill=total_pu_uses)) +
            geom_histogram()+
            xlim(0,max(PU_RATE$total_pu_uses))+
            ylim(0,40)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



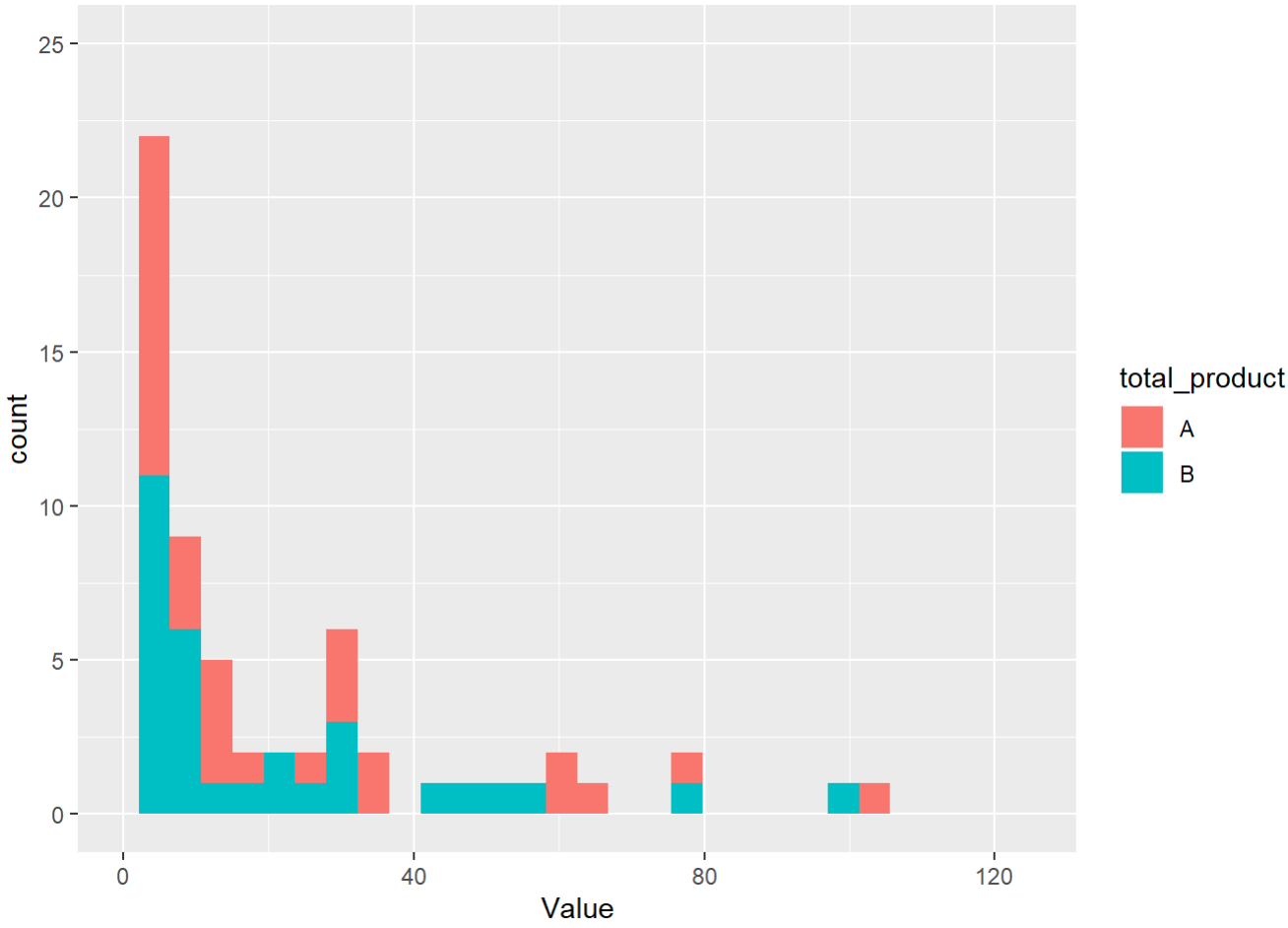

```
REN_RATE =  GT %>%
    group_by(total_revenue) %>%
    mutate(A=ifelse(ab_group=="A",1,0),B=ifelse(ab_group=="B",1,0)) %>%
    summarise(A=sum(A),B=sum(B))

        REN_RATE %>%
            gather(key=total_revenue, value=Value) %>%
            ggplot(aes(x=Value,fill=total_revenue)) +
            geom_histogram()+
            xlim(0,max(REN_RATE$total_revenue))+
            ylim(0,200)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
PROD_RATE =  GT %>%
    group_by(total_product) %>%
    mutate(A=ifelse(ab_group=="A",1,0),B=ifelse(ab_group=="B",1,0)) %>%
    summarise(A=sum(A),B=sum(B))

        PROD_RATE %>%
            gather(key=total_product, value=Value) %>%
            ggplot(aes(x=Value,fill=total_product)) +
            geom_histogram()+
            xlim(0,max(PROD_RATE$total_product)) +
            ylim(0,25)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Los histogramas superpuestos de todas las variables no permiten gráficamente sacar grandes conclusiones ya que parece que los cambios y evoluciones son más sutiles, en adelante lo vemos.

```
GT_1 = GT %>% group_by(ab_group) %>%
    summarise(USR=n(),
              STR=sum(total_game_starts),
              REN=sum(total_revenue),
              PUR=sum(total_product)) %>%
    mutate(`S/U`=STR/USR,
           `R/U`=REN/USR,
           `P/U`=PUR/USR) %>%
    adorn_totals(where="row")

sum(GT_1[,2])
```

```
## [1] 1528402
```

```
for(x in 6:8){
        GT_1[3,x]=((GT_1[2,x]-GT_1[1,x])/GT_1[1,x])*100
}
```

Una primera métrica sería establecer indicadores de densidad, o peso específico, para comprobar qué ratio de comienzos de partida, que los llamamos S/U (cantidad de partidas comenzadas por cantidad de usuarios que han participado) y R/U // P/U con cantidad de dinero inyectado y compras realizadas por usuario.

Después compruebo en estas variables el crecimiento o descenso que ha sufrido en cantidades cuando se ha pasado al grado de dificultad B, y efectivamente han aumentado, aunque no en un porcentaje muy alto (en torno al 3,3%).

```
GT_2 = GT %>% group_by(ab_group) %>%
        summarise(USR=n(),
              STR=sum(total_game_starts),
              PWU=sum(total_pu_uses),
              REN=sum(total_revenue),
              PUR=sum(total_product)) %>%
    mutate(`S/T`=STR/sum(STR),
           `PU/T`=PWU/sum(PWU),
           `R/T`=REN/sum(REN),
           `P/T`=PUR/sum(PUR))
```

```
GT_3= GT %>% group_by(ab_group) %>%
    summarise("CORpu-ren"=cor(total_pu_uses,total_revenue),
              "CORpu-prd"=cor(total_pu_uses,total_product),
              "CORpu-str"=cor(total_pu_uses,total_game_starts),
              "CORstr-ren"=cor(total_game_starts,total_revenue),
              "CORstr-prod"=cor(total_game_starts,total_product))

GT_3 = adorn_totals(GT_3,where="row")
for(x in 2:5){
        GT_3[3,x]=((GT_3[2,x]-GT_3[1,x])/GT_3[1,x])*100
}
```

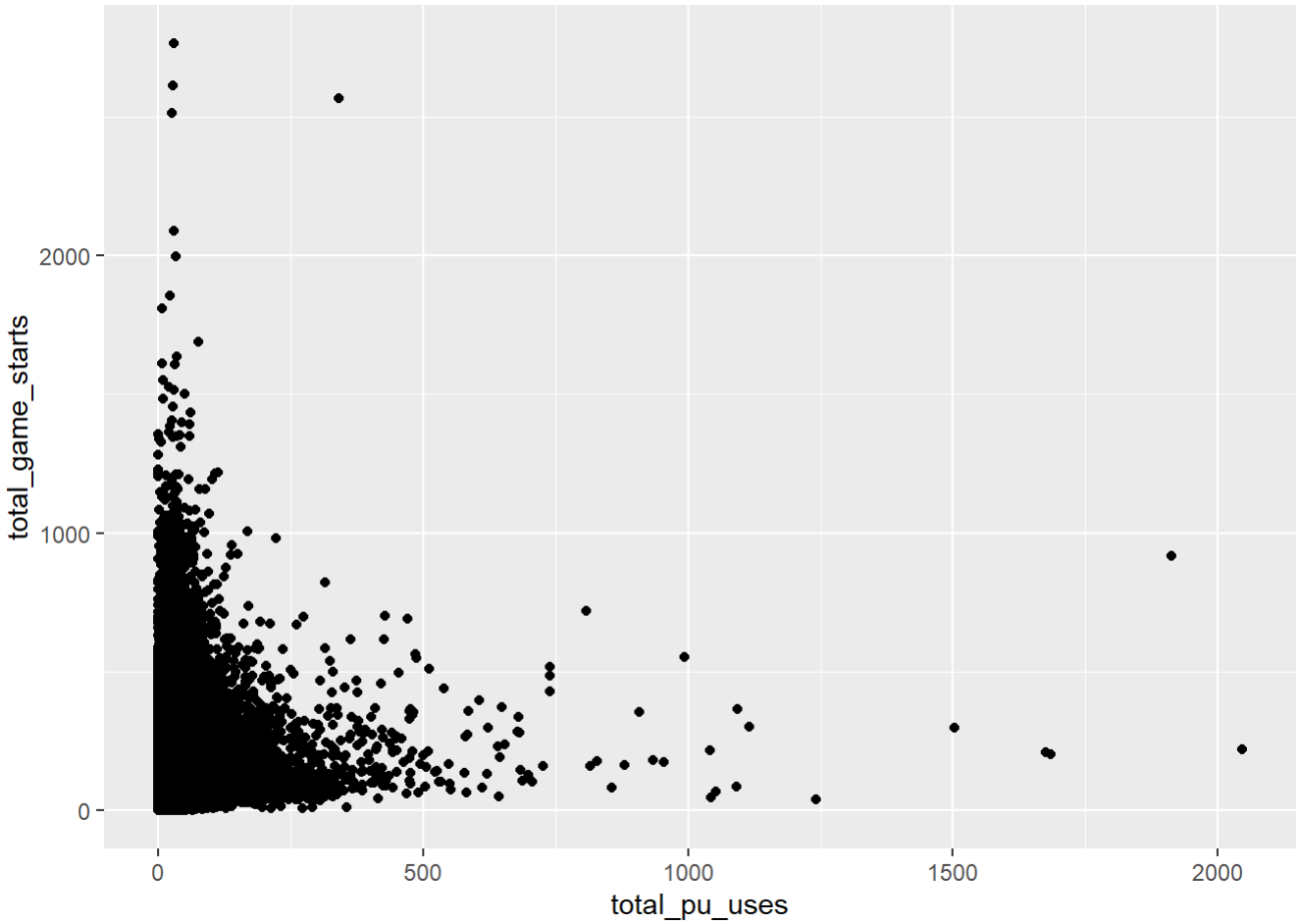
```
    median(as.double(GT_3[3,2:5]))
```

```
## [1] -7.184121
```

```
    mean(as.double(GT_3[3,2:5]))
```

```
## [1] -6.496022
```

```
GT %>% filter(ab_group=="A") %>%
    ggplot(aes(x=total_pu_uses,y=total_game_starts)) +
    geom_point()
```



En un mapa de correlaciones observamos que se ha reducido la correlación entre las diferentes variables (Start,PU,REN,PROD) Al pasar al grado de dificultad B que no es beneficiosa, puesto que se diluye un patrón reconocible.

Esto nos puede situar en una aparente “incogruencia” ya que se presupone, al ver que existe un crecimiento objetivo de los KPI relativos al aumento de compras de productos y dinero gastados así comop uso de powerups, pero sin embargo se ha reducido la relación entre estas variables.

Se intentará solventar esto clasificando por tipo de jugador:

```
GT = GT %>% rename(TS=total_game_starts,PU=total_pu_uses,RE=total_revenue,PR=total_product,
AB=ab_group)

GT_4 = GT %>% group_by(AB) %>%
    mutate(T1=ifelse(TS<max(TS)/6,1,0),
           T2=ifelse(TS %between% c(max(TS)/6, max(TS)/5),1,0),
           T3=ifelse(TS %between% c(max(TS)/5, max(TS)/4),1,0),
           T4=ifelse(TS %between% c(max(TS)/4, max(TS)/3),1,0),
           T5=ifelse(TS %between% c(max(TS)/3, max(TS)/1.5),1,0),
           T6=ifelse(TS>max(TS)/1.5,1,0)) %>%
    summarise(T1=sum(T1),
              T2=sum(T2),
              T3=sum(T3),
              T4=sum(T4),
              T5=sum(T5),
              T6=sum(T6))
```

```
GT_5 = GT %>% group_by(AB) %>%
    mutate(T1=ifelse(PU<max(PU)/6,1,0),
           T2=ifelse(PU %between% c(max(PU)/6, max(PU)/5),1,0),
```

```
          T3=ifelse(PU %between% c(max(PU)/5, max(PU)/4),1,0),
          T4=ifelse(PU %between% c(max(PU)/4, max(PU)/3),1,0),
          T5=ifelse(PU %between% c(max(PU)/3, max(PU)/1.5),1,0),
          T6=ifelse(PU>max(PU)/1.5,1,0)) %>%
    summarise(T1=sum(T1),
              T2=sum(T2),
              T3=sum(T3),
              T4=sum(T4),
              T5=sum(T5),
              T6=sum(T6))
```

```
GT_6 = GT %>% group_by(AB) %>%
    mutate(T1=ifelse(PR<mean(PR[PR!=0])*10/6,1,0),
          T2=ifelse(PR %between% c(mean(PR[PR!=0])*10/6, mean(PR[PR!=0])*10/5),1,0),T3=ife
lse(PR %between% c(mean(PR[PR!=0])*10/5, mean(PR[PR!=0])*10/4),1,0),T4=ifelse(PR %between%
c(mean(PR[PR!=0])*10/4, mean(PR[PR!=0])*10/3),1,0),T5=ifelse(PR %between% c(mean(PR[PR!=0])
*10/3, mean(PR[PR!=0])*10/1.5),1,0),T6=ifelse(PR>mean(PR[PR!=0])*10/1.5,1,0)) %>% summarise
(T1=sum(T1),T2=sum(T2),T3=sum(T3),T4=sum(T4),T5=sum(T5),T6=sum(T6))
```

Y observamos los sumarios por grupo:

```
summary(filter(GT,AB=="A"))
```

```
##                                 user_id             TS
##  00001591-9A66-45A9-9D6D-EB2C957B7DD0:     1   Min.   :   0.0
##  0000399D-1029-4BF8-84DA-5F268F5401D3:     1   1st Qu.:  15.0
##  00007952-2A50-45F8-94E2-CB949C154908:     1   Median :  64.0
##  0000CEE0-4472-4F32-A2EA-66D52C1D776B:     1   Mean   : 104.6
##  0000FFDD-9EBE-4078-9276-69E47F7F76E9:     1   3rd Qu.: 156.0
##  00010B12-EC5E-4C0E-87AF-F814FE67E960:     1   Max.   :2764.0
##  (Other)                             :381933
##        PU                RE                PR          AB
##  Min.   :   0.00   Min.   :  0.0000   Min.   : 0.0000   A:381939
##  1st Qu.:   1.00   1st Qu.:  0.0000   1st Qu.: 0.0000   B:     0
##  Median :   6.00   Median :  0.0000   Median : 0.0000
##  Mean   :  10.76   Mean   :  0.6384   Mean   : 0.1542
##  3rd Qu.:  15.00   3rd Qu.:  0.0000   3rd Qu.: 0.0000
##  Max.   :2045.00   Max.   :684.7800   Max.   :92.0000
##
```

```
summary(filter(GT,AB=="B"))
```

```
##                                 user_id             TS
##  00001CFC-7DB1-46E6-BA1B-EF699F13B1C1:     1   Min.   :   0.0
##  00006DA0-80CC-4C2B-A01A-B4873F42122B:     1   1st Qu.:  15.0
##  0000EC54-BF9B-40A2-9698-9EE67B398036:     1   Median :  64.0
##  00014D4A-78F6-41BB-B9CE-B488EA824BCA:     1   Mean   : 104.5
##  000234AD-8AD5-441D-875C-9AC6AA22385A:     1   3rd Qu.: 156.0
##  000296D5-B36D-422B-844E-B00995DFF6A9:     1   Max.   :2678.0
##  (Other)                             :382256
##        PU                RE                PR           AB
##  Min.   :   0.00   Min.   :  0.0000   Min.   :  0.0000   A:     0
##  1st Qu.:   1.00   1st Qu.:  0.0000   1st Qu.:  0.0000   B:382262
##  Median :   6.00   Median :  0.0000   Median :  0.0000
```

```
##  Mean   :  10.78   Mean   :  0.6593   Mean   :  0.1591
##  3rd Qu.:  15.00   3rd Qu.:  0.0000   3rd Qu.:  0.0000
##  Max.   :4509.00   Max.   :610.7400   Max.   :125.0000
##
```

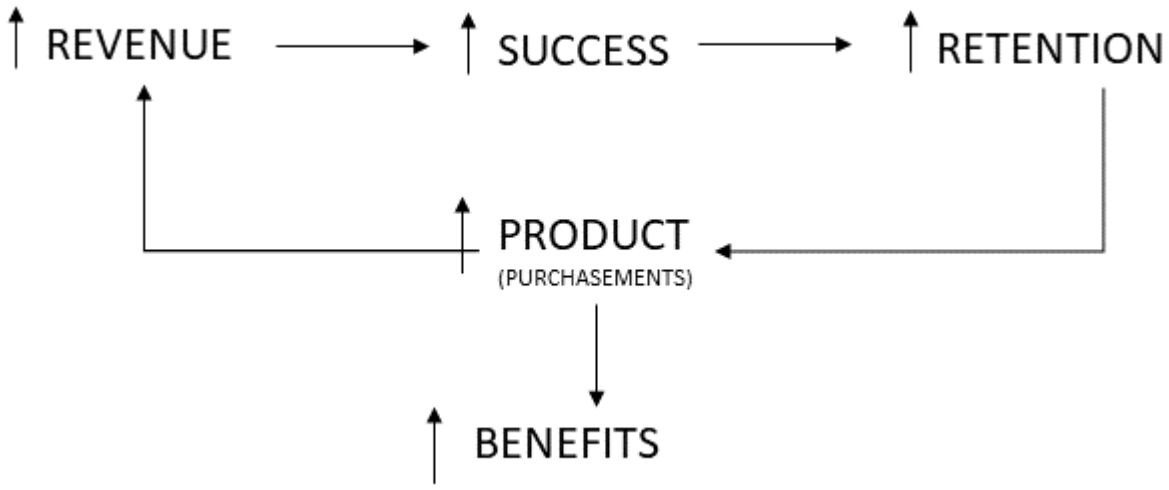
```
GT %>% group_by(AB) %>% summarise(PU = max(PU),PR = (mean(PR)*10)) %>% rbind(c())
```

```
## # A tibble: 2 x 3
##   AB       PU    PR
##   <fct> <int> <dbl>
## 1 A      2045  1.54
## 2 B      4509  1.59
```

Pero no se obtiene gran cosa más. Con los datos proporcionados podemos concluir que haber pasado al nivel de dificultad “B” ha supuesto , el consumo de los PU y PR ha aumentado como se comentaba antes, y observamos en los sumarios que el valor máximo de consumos en determinados jugadores se ha incrementado al doble en PU, y en más de un 40% en compras, lo que sugiere que se ha incrementado el grado de success, la reentrada, y gasto en productos.

Debo decir que no conozco bien las metodologías que se siguen para un test A/B, me harían falta datos de días para estudiar las retenciones o los llamados DAUs o MAUs (Daily active Users). Pero comprendo que la clave está en la relación intrínseca entre el Success, el Revenue (gasto de dinero) y la retención.

La idea es clara:D:



Caption for the picture.

De manera que concluimos que el **cambio de dificultad B ha sido benificioso.**

Usuarios Dia:

¿Cúantos usuarios harían falta para poder estimar este KPI con un nivel de confianza al 90% y con un tamaño del intervalo de confianza de 5%?

La respuesta a esta pregunta es sencilla, aunque hacemos algunas suposiciones, junto con las condiciones de contorno:

- Se supone una población universal infinita, dado la capacidad de muestreo de la que se dispone
- Se supone una probabilidad de retención a día uno del 50% (40-60%)

- Nivel de confianza del 90%
- Error máximo asumido o intervalo de confianza del 5%
- Se asume que su distribución es atribuible a una Distribucion normal N(0,1)

Y usando el principio del *Teorema central*, y simplificando, aplicamos la siguiente fórmula:

```
    n = (Z^2·p(1-p))/e^2

    Donde:
            - Z = Zalpha (para intervalo de confianza de 95%  ~1,62 interpolando)
            - p = probabilidad de éxito en la retencion a día 1
            - e = intervalo de confianza
```

```
Nsamples1 = ((1.62^2)*0.5*(0.5))/(0.05^2)
Nsamples1
```

```
## [1] 262.44
```

```
Nsamples2=sample.size.prop(e=0.05, P=0.5, N=Inf, level=0.9)
Nsamples2
```

```
##
## sample.size.prop object: Sample size for proportion estimate
## Without finite population correction: N=Inf, precision e=0.05 and expected proportion P=
0.5
##
## Sample size needed: 271
```

La muestra total para cumplir con el objetivo de confianza propuesto es de 271 muestras. Como se puede ver, calculado manualmente se obtiene una cifra parecida, en torno a las 262 muestras.

Básicamente esto nos explica, basándonos en el *Teorema central* del límite y llevando en cuenta las asunciones hechas más arriba, que para una distribución de probabilidad asumible como normal, necesitaríamos como mínimo una muestra > 270 (jugadores en nuestro caso) de la población completa para que la estimación de la probabilidad de que un sujeto que juegue un día vaya a hacerlo al día siguiente (que en nuestro sector como dice el enunciado suele tener un valor comprendido entre 40-60%) se encuentre al nivel de confianza exigido (que podríamos ver gráficamente que ocupa el espacio encerrado por la campana de Gauss) del 90%.

La fiabilidad de la *estimación de probabilidad* aumentaría con el número de muestras escogidas de la población, pero con la cantidad de 271 muestras ya estaríamos cumpliendo con las expectativas propuestas.

Conclusiones y comentarios

A día de hoy, lo interesante es el planteamiento de análisis regresivos que permitan rescatar ingentes cantidades de datos de partidas y eventos pasados que permitan alimentar *algoritmos de aprendizaje automático* para hacer posible no sólo maximizar la fiabilidad de las estimaciones de los indicadores de éxito, suceso, aumento de compras, y toda clase de KPIs que afecten al aumento de beneficios, si no la *capacidad anticipatoria* de cara a los tests A/B futuros: toda clase de cambios aplicados en los juegos para mejorar las aplicaciones de cambios en los eventos futuras.

Esto quiere decir, sería muy interesante aplicar en el futuro no sólo análisis de tipo test A/B *retroactivos*, por así decirlo, si no valorar también la fiabilidad de estos tests haciendo simulaciones con algoritmos de Machine learning que puedan predecir tanto, la combinación de variables adecuada y óptima para asegurar maximizar las variables que afecten a los beneficios que genere el juego si no para mejorar y perfeccionar las variables y metodologías usadas en los análisis estadísticos para estimar las KPIs.