Statistiques Lol

Quelle est la meilleur methode pour gagner de l'experience dans League of Legends ?

Figure 1: Logo lol

Introduction

Dans League of legends, l'experience permet à un joueur (ou summoner) d'améliorer son niveau. Au fur et à mesure de sa montée de niveau, le joueur va avoir accès à de plus en plus de mode, l'encourageant ainsi à jouer et à progresser. Une fois tous les modes débloqués, les joueurs ont toujours accès à des bonus à chaque montée de niveau. Plus le niveau augmente, plus la quantité d'experience nécéssaire pour passer au niveau suivant est grande. Nous nous sommes donc demandé quelle était la meilleur methode pour gagner de l'experience dans League of Legends.

L'experience est souvent abréviée en **xp**.

Nos données

L'experience est calculé à la suite d'un match pour chaque joueur. Nous avons donc eu besoin d'avoir les données non pas par équipe mais par joueur. Nous aurions pu utiliser le fichier par match et le découper pour chaque joueur mais comme en plus nous avions bespoin d'ajouter des champs dans le fichier source, nous avons jugé bon de réécrire le parseur afin de ce baser sur un fichier CSV dans lequel une ligne correspond à un joueur.

Nous avons joint une copie du code de ce projet en retirant les fichiers JSON sources afin de diminuer la taille de l'archive. Nous vous invitons à vous référer au dossier **LOL_java_project** afin d'avoir un aperçu de nos modifications.

Nos règles

Pour tous nos tests impliquant l'étude d'un p-value, nous utilisons une valeur alpha de 0.05.

```
# Setup librairies and dataset
data <- read.csv("data/player_lol_data.CSV")
data_lol <- read.csv("data/player_team_lol_data.csv")
library(questionr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Analyses statistiques

```
# récupération des xp de tous les joueurs
vec_xp <- c(data[,"xp"]);
```

Etude sur l'experience accumulée par les joueurs pendant la partie

Etude de données sur l'expérience

Pour commencer, nous allons voir quelques données sur la variable **xp**, qui correspond à l'expérience engrangée par un joueur lors d'une partie. Cela permettra plus tard de mettre en contexte les observations par rapport à cette variable (par exemple, on pourra voir si la fréquence de répartion de l'**xp** a une importance dans certaines observations et peuvent mener à de faux résultats).

La première données intéressante sur cette variable serait la *moyenne*. Cela permetterait d'avoir un repère quant à l'**xp** engendrée, et de savoir si une valeur s'écarte trop de la *moyenne*. De plus, nous allons stocker la *moyenne* car elle pourrait servir plus tard.

```
moy_xp = mean(vec_xp);
print(moy_xp); # affichage de la moyenne
```

```
## [1] 944.0109
```

Maintenant que nous avons stocké la *moyenne*, nous pouvons regarder plus généralement les données concernant l'**xp** comme la *mediane* ou encore les *quartiles* grâce à la commande summary.

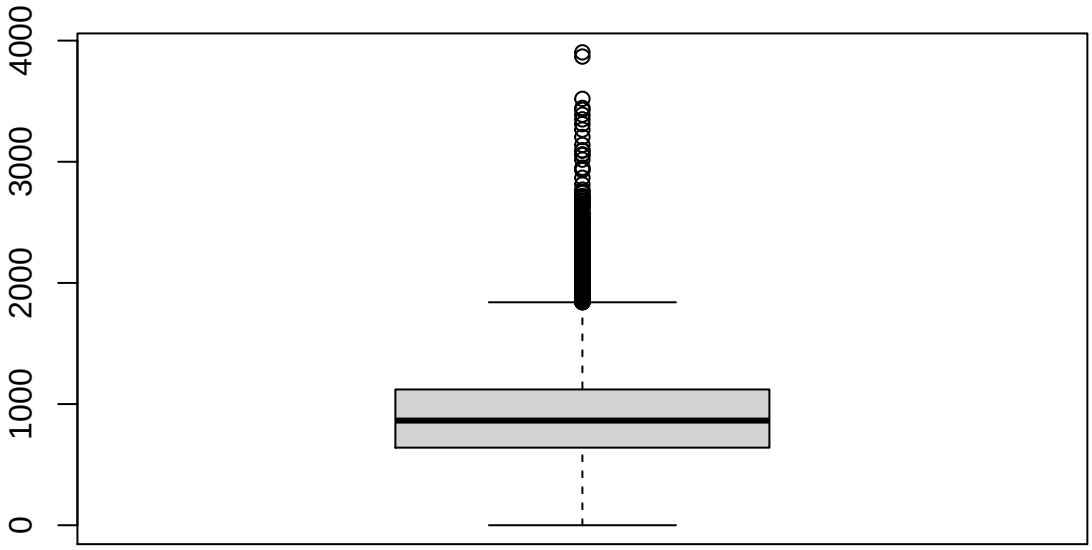
```
summary(vec_xp);
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     0.0   640.5   863.9   944.0  1120.6  3903.1
```

On peut voir que plusieurs choses : 1. Les *quartiles* ne semblent pas beaucoup s'écarter la *moyenne*, ce qui pourrait indiquer une bonne homogénéité de l'échatillon. Mais cela reste à montrer avec un calcul de la *variance*, et il se pourrait que le calcul infirme ce que notre approximation semble dire 2. La *mediane* est plus basse que la *moyenne*. Cela signifie que plus de 50% des joueurs ont une **xp** inférieure à la *moyenne* en fin de partie. Il n'est donc pas aisé de gagner plus d'**xp** que la *moyenne*. Nous pourrons tenter d'y trouver une explication grâce à de futures observations.

En attendant, nous pouvons avoir un aperçu graphique des chiffres vus plus haut grâce à un boxplot :

```
vec_outliers <- boxplot(vec_xp)$out;
```




```
vec_wo_outliers <- vec_xp[!(vec_xp %in% vec_outliers)]
```

On voit, sur le boxplot, la *mediane* (un trait noir épais), les 1er et 3ème *quartiles* (bord inférieur et supérieur du rectangle gris), et les valeurs extrêmes : *minimum* et *maximum* (symbolisées par un trait en dessous et au dessus des pointillées). Cependant, 2 choses : on voit que le *maximum* sur le boxplot est aux alentour de 2000, alors que plus haut, nous avons vu que le maximum était à **3903.1**, et on peut voir beaucoup de petits cercles au dessus du *maximum*.

Ces cercles sont ce que l'on appelle les *valeurs aberrantes*, ou en anglais : *outliers*. Ce sont des valeurs qui sont considérées comme trop loin des autres valeurs et qui pourraient donc fausser les résultats.

On a donc récupérer ces *outliers* ainsi que la liste des **xp** qui ne sont pas des *outliers* afin de pouvoir faire des observations avec et sans les *outliers*.

Test sur l'homogénéité de l'expérience

Afin de savoir si l'**xp** est une variable à forte dispertion, il faut que l'on calcule l'*écart type*. On va le faire avec toutes les valeurs de l'**xp** puis sur les valeurs des **xp sans outliers**. Le but sera de comparer leur *écart type*. Normalement, l'*écart type* du second sera plus petit que le premier et donc l'échantillon devrait être plus homogène, mais il reste intéressant de voir à quel point retirer les *outliers* a influencer l'homogénéité.

```
sd(vec_xp);
```

```
## [1] 486.5885
```

```
sd(vec_wo_outliers);
```

```
## [1] 379.0009
```

```r
sd(vec_wo_outliers) / sd(vec_xp);
```

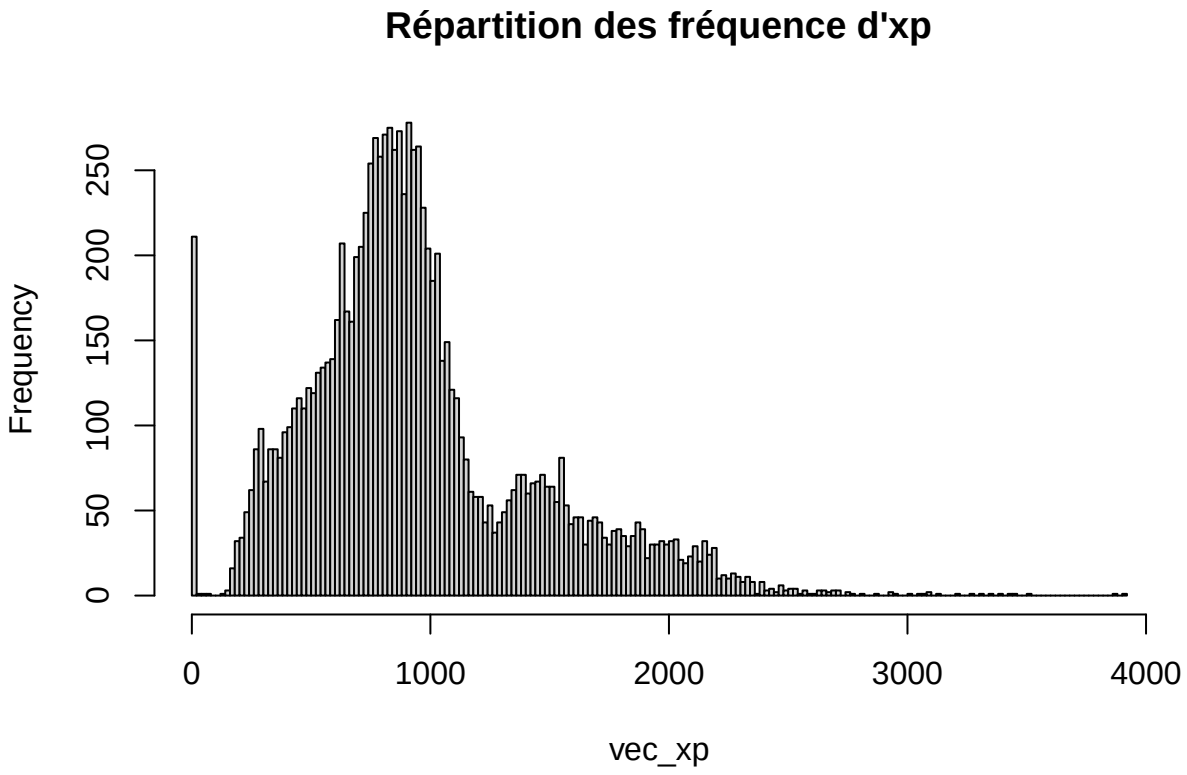
```
## [1] 0.778894
```

On voit effectivement que l'*écart type* de l'échantillon sans les *outliers* est plus petit que l'*écart type* de l'échantillon complet. Il a même diminué de presque **25%**. Le but de faire des tests ensuite sans les *outliers* est d'avoir une hétérogénéité diminuée de 25%.

Détermination d'une loi de probabilité pour représenter l'expérience

Nous allons essayer de voir avec quelle loi de probabilité nous pouvons approcher la distribution de l'**xp**. Il y a de fortes chances de penser que la distribution approchera la loi normale, nous allons donc voir un histogramme des **xp**.

```r
hist(vec_xp, breaks = 200, main = "Répartition des fréquence d'xp");
```



On voit sur ce graphique que la distribution se rapproche en effet de la loi normale. Comme nous pouvons calculer la *moyenne* et l'*écart type* de la série, nous pouvons calculer la probabilité d'avoir un certain nombre d'**xp** :

```r
1 - pnorm(2000, mean = moy_xp, sd = sd(vec_xp));
```

```
## [1] 0.01499626
```

Par exemple, ici, on a **1.5%** de chances d'obtenir plus de **2000 xp** à la fin de la partie.

Pour eviter les répétitions, on peut même créer des fonctions pour calculer les probabilités d'avoir tant d'**xp** :

```r
pxp <- function(q){
  return(pnorm(q, mean = moy_xp, sd = sd(vec_xp)));
}

dxp <- function(q){
  return(dnorm(q, mean = moy_xp, sd = sd(vec_xp)));
}

rxp <- function(n){
  return(abs(rnorm(n, mean = moy_xp, sd = sd(vec_xp))));
}
```

Voilà, maintenant, imaginons que l'on veut connaître la probabilité d'avoir moins de **500 xp** ainsi que la probabilité d'avoir exactement le nombre d'**xp** moyen, on peut utiliser ces fonctions :

```r
pxp(500) # probabilité que xp < 500
```

```
## [1] 0.1807534
```

```r
dxp(moy_xp) # probabilité que xp = moyenne d'xp
```

```
## [1] 0.0008198761
```

On a donc **18%** de chance d'avoir moins de **500 xp** à la fin d'une partie, et **0.08%** de chance d'obtenir exactement la moyenne.

On peut aussi simuler une partie par exemple en tirant **10 xp** (pour les 10 joueurs de la partie) au sort suivant la répartition des **xp**

```r
fgame <- as.data.frame(t(rxp(10)));
names(fgame) <- c("joueur1","joueur2","joueur3","joueur4","joueur5","joueur6","joueur7","joueur8","joue
fgame
```

```
##    joueur1  joueur2  joueur3  joueur4  joueur5  joueur6  joueur7  joueur8
## 1 312.8867 65.26166 1673.077 479.4864 1173.253 846.5825 444.0235 1305.293
##   joueur9 joueur10
## 1 708.895  1911.91
```

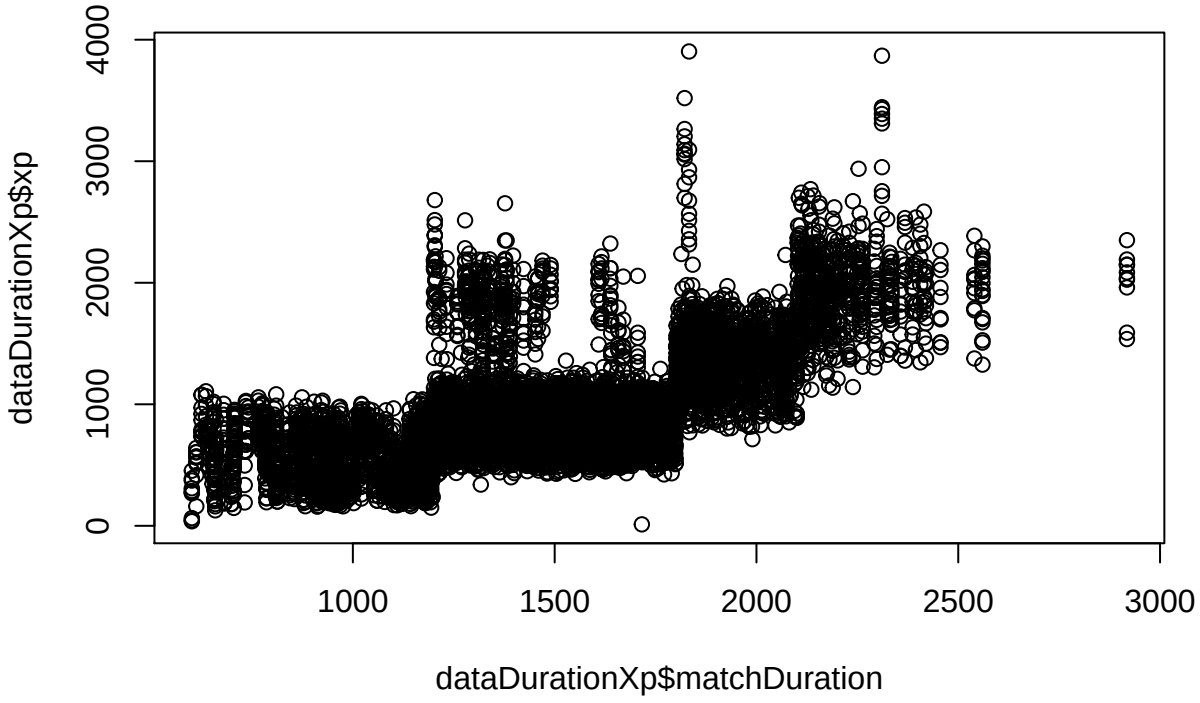
La durée des matchs et l'experience

Notee problématique concernant le gain d'experience post -match, nousnous sommes renseigné quelles caractéristiques pouvaient avoir un rôle dans le gain d'experience et il semblerait que la dur&ée des matchs ait une importance capitale. Nous retrouvons cette valeur à partir du premier jeu de donnée (par match) dans la colonne `gameDuration`.

```
# Pour la suite, nous allons extraire uniquement les deux variables dont nous avons besoin
dataDurationXp <- data[,c("matchDuration", "xp")]
# Nous retirons également les ligneqs avec une durée nulle ainsi que celles avec un xp nul, car ce sont
dataDurationXp <- filter(dataDurationXp, matchDuration != 0)
dataDurationXp <- filter(dataDurationXp, xp != 0)
```

Notre objectif est de montrer une éventuelle variation simultanée entre les variables `matchDuration` et `xp`.

```
plot(dataDurationXp$matchDuration, dataDurationXp$xp)
```



Nous pouvons déjà faire quelques remarques à partir de cette courbe. Premièrement plus les matchs durent longtemps, plus les joueurs semblent récupérer de l'expérience. Ensuite la courbe fait une sorte d'escalier ce qui pourrait indiquer qu'une fraction arrondie du temps de jeu a une influence sur l'experience. Enfin, nous pouvons constater une dépendance positive, il est cependant difficile d'affirmer qu'elle est linéaire ou non.

Les coefficients

Examinons maintenant les différents coefficients afin de mieux comprendre la dépendance entre ces deux variables

```
# Coefficient de corélation linéaire de Pearson (Compris entre -1 et 1, si la valeur absolue du coeffic
c_pearson <- cor(dataDurationXp$matchDuration, dataDurationXp$xp)
```

```r
print("Coefficient de Pearson :")
```

```
## [1] "Coefficient de Pearson :"
```

```r
print(c_pearson)
```

```
## [1] 0.678494
```

```r
# Coefficient de corélation des rangs de Spearman (Compris entre -1 et 1, si la valeur absolue du coeff
c_spearman <- cor(dataDurationXp$matchDuration, dataDurationXp$xp, method = "spearman")
print("Coefficient de Spearman :")
```

```
## [1] "Coefficient de Spearman :"
```

```r
print(c_spearman)
```

```
## [1] 0.6761558
```

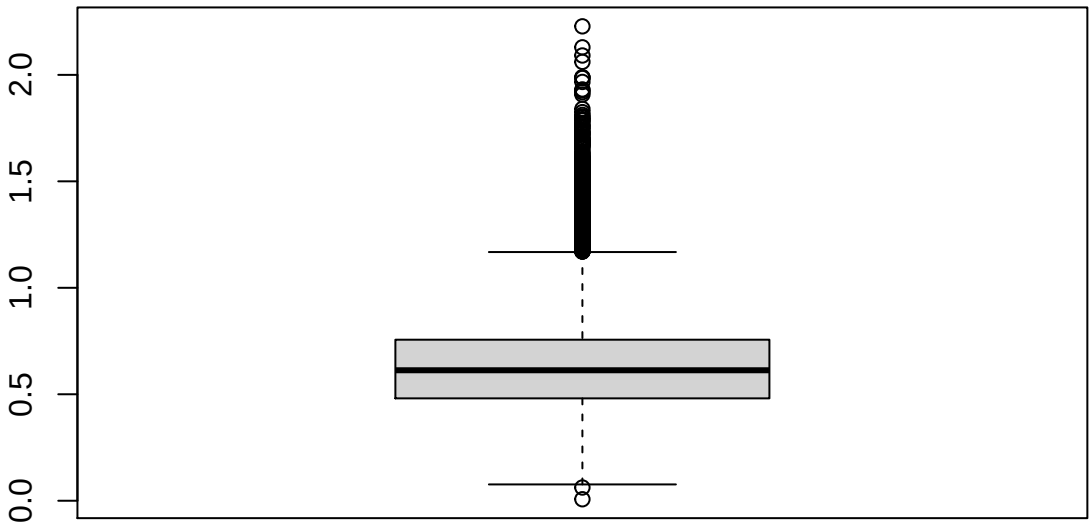
Le coefficient de Spearman est superieur à 0.5, on est donc bien en présence d'une dépendance monotone positive. De plus le coefficient de Pearson montre une linéarité entre les deux variables même si la valeur ne dépasse `0.5` de peu.

Devant ces résultats, nous pouvons aisément émettre l'hypothèse que d'autrees variables affectent l'experience ou alors que le temps de jeu n'est qu'une dérive d'une autre variable. Pour le démontrer, nous étudions le rapport entre l'experience reçu par un joueur et le temps de la partie.

Recherche d'un cefficient

Pour travailler avec les coefficients, nous ajoutons une colonne `coef` à nos données ayant pour valeur le rapport entre l'experience et la durée des matchs.

```r
d2 <- transform(dataDurationXp, coef = xp / matchDuration)
boxplot(d2$coef)
```




```
summary(d2$coef)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.006993 0.480703 0.612679 0.642302 0.756197 2.227847
```

La moyenne des coefficient est de 0.64 avec des quartiles très proches signifiant ainsi que les coefficients sont très homogènes. Cependant les valeurs s'éttendent sur une très grande plage.

On peut donc faire deux hypothèses :

- Il existe une ou plusieur autres variables permettant de calculer le total d'experience d'un joueur
- Le temps de jeu n'a pas de lien direct avec le calcul de l'experience mais il est influencé par une autre variable qui a un lien direct avec le calcul

La victoire et l'experience

Nous souhaitons maintenans évalur la statistique bidimentionelle composée de l'experience (variable quantitative) et de la victoire (variable qualitative). Notre objectif est de rechercher une éventuelle corélation entre la victoire d'une équipe et le gain d'experience des joueurs.

La variable victoirea 2 modalités, nous avons donc 2 sous populations.

```
dataWinXp <- data[,c("win", "xp")]

d_win <- filter(dataWinXp, win == 1)
d_lose <- filter(dataWinXp, win == 0)

# On aurait aussi pu obtenir les données avec la fonction `tapply` mais comme on a que 2 résultats, je
m_win <- mean(d_win$xp)
m_lose <- mean(d_lose$xp)
m_global <-mean(dataWinXp$xp)

print("Moyenne d'experience des joueurs gagnants :", quote = FALSE)
```

```
## [1] Moyenne d'experience des joueurs gagnants :
```

```
print(m_win)
```

```
## [1] 984.6953
```

```
print("Moyenne d'experience des joueurs perdants :", quote = FALSE)
```

```
## [1] Moyenne d'experience des joueurs perdants :
```

```
print(m_lose)
```

```
## [1] 903.3265
```

```
print("Moyenne de l'ensemble des joueurs :", quote = FALSE)
```

```
## [1] Moyenne de l'ensemble des joueurs :
```

```
print(m_global)
```

```
## [1] 944.0109
```

```
print("Rapport moyenne d'experience des équipe gagnante contre moyenne d'experience globale :", quote =
```

```
## [1] Rapport moyenne d'experience des équipe gagnante contre moyenne d'experience globale :
```

```
cat((m_win / m_global) * 100, "%\n")
```

```
## 104.3097 %
```

```
print("Rapport moyenne d'experience des équipe perdante contre moyenne d'experience globale :", quote =
```

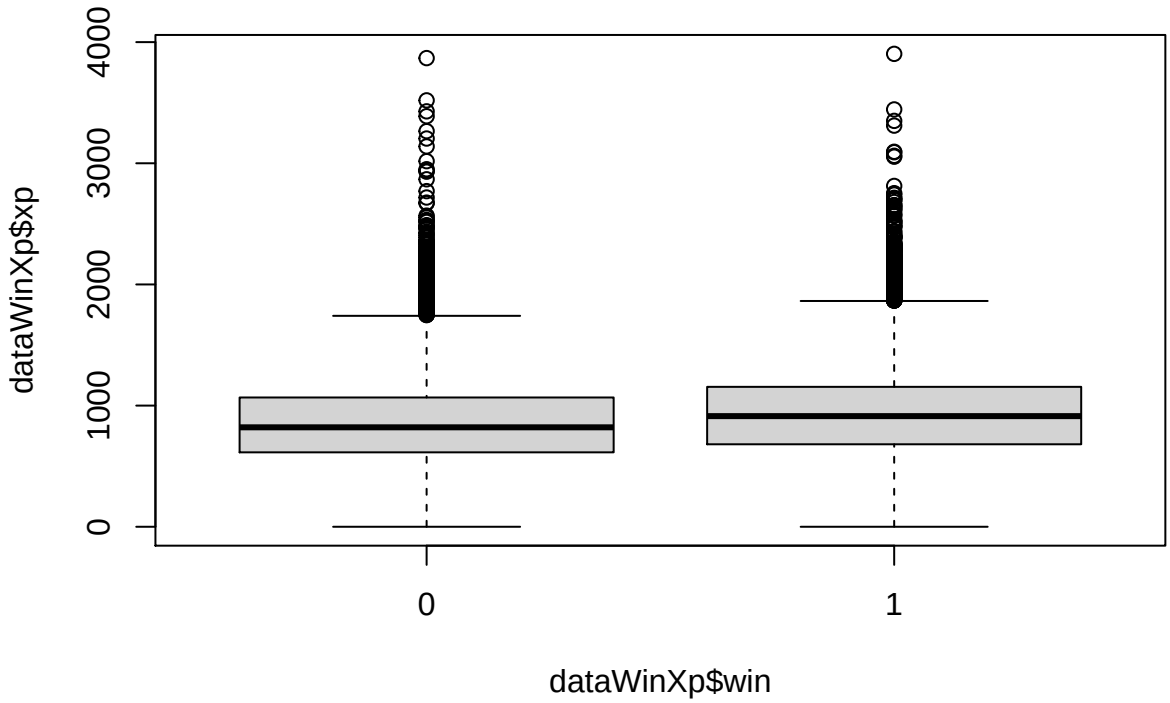
```
## [1] Rapport moyenne d'experience des équipe perdante contre moyenne d'experience globale :
```

```
cat((m_lose / m_global) * 100, "%\n")
```

```
## 95.69026 %
```

En regardant les moyennes des deux échantillons ainsi que celle de l'échantillon parent nous constatons que celles des échantillons semblent être espacées de façon égales de la moyenne globale. Nous avons donc calculé le rapport entre les moyennes et l'on optien une augmentation d'environ 5% pour les équipes gagnantes par rapport à la moyenne et une diminution de 5% pour les équipes perdantes.

```r
boxplot(dataWinXp$xp ~ dataWinXp$win)
```



Dans ce graphique ci-dessus, nous constatons que les joueurs ayant gagné une partie ont sensiblement plus d'experience en fin de partie que les joueurs ayant perdu.

Conclusion

Nous pouvons constater par l'étude des moyennes et du graphique ci-dessus qu'il existe bien une corélation entre la victoir d'une équipe et le gain d'experience des joueurs. Cette différence semble être de +-5% mais

Observations

Le nombre de dommage infligés par un joueur influe sur son total d'experience

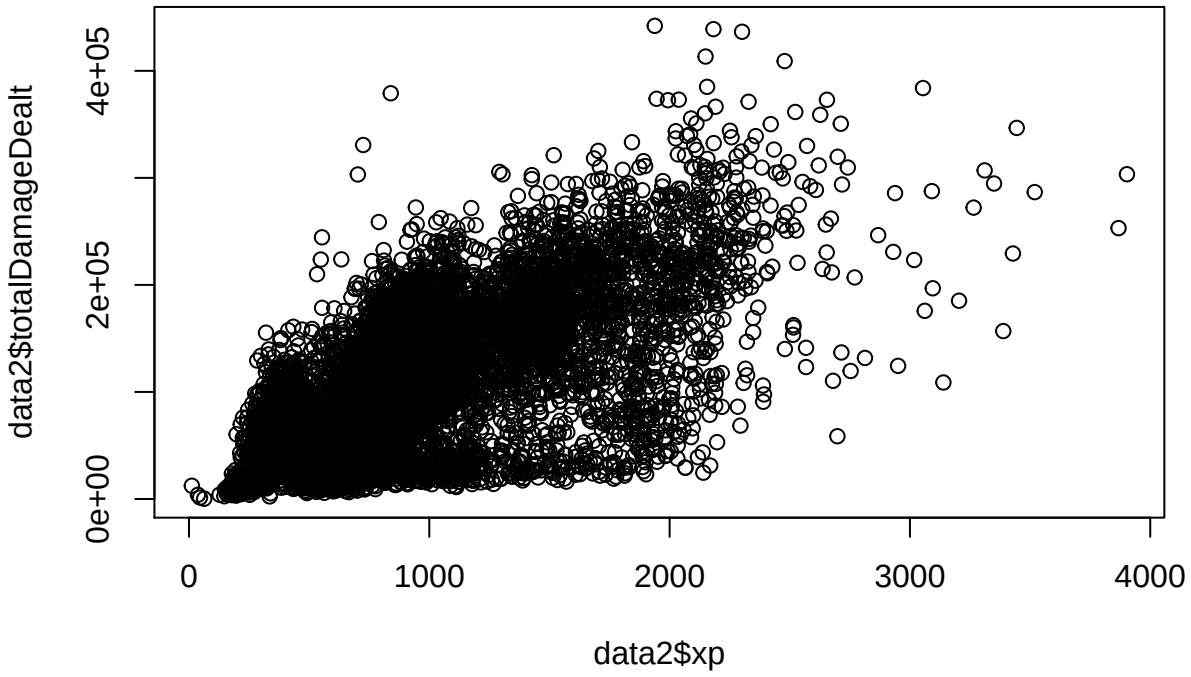
Dans la majorité des jeux comportant du combats, les dommages infligés par un joueur ont une grande importance dans sa progression. Un joueur qui n'infligerera que peu de dommages aura souvent plus de mal à avancer alors qu'un joueur en infligeant beaucoup pourra progresser avecmoins de difficulté. League of legend ne fait pas exception nous met à dispositiondes données sur la quantité de dommages infligés par joueur. Nous nous sommes donc demandés si la quantité de dommage infligés avaient une influence sur la progression des joueurs.

H0 : Plus un joueur va infligé de dommages aux ennemis, plus il recevra d'experience en fin de combat.

H1 : Le nombre de dommage infligé n'a pas d'incidence sur le total d'experience en fin de combat.

```
# Remove unnecessary values (we remove players that gains no xp and players that dealt no dammages, bec
data2 <- filter(data, xp != 0)
data2 <- filter(data2, totalDamageDealt != 0)

plot(data2$totalDamageDealt ~ data2$xp)
```



Pour réfuter H1, nous effectuons un test d'indépendance pour réfuter l'indépendance des 2 caractéritiques

```
cor.test(data2$xp, data2$totalDamageDealt)$p.value
```

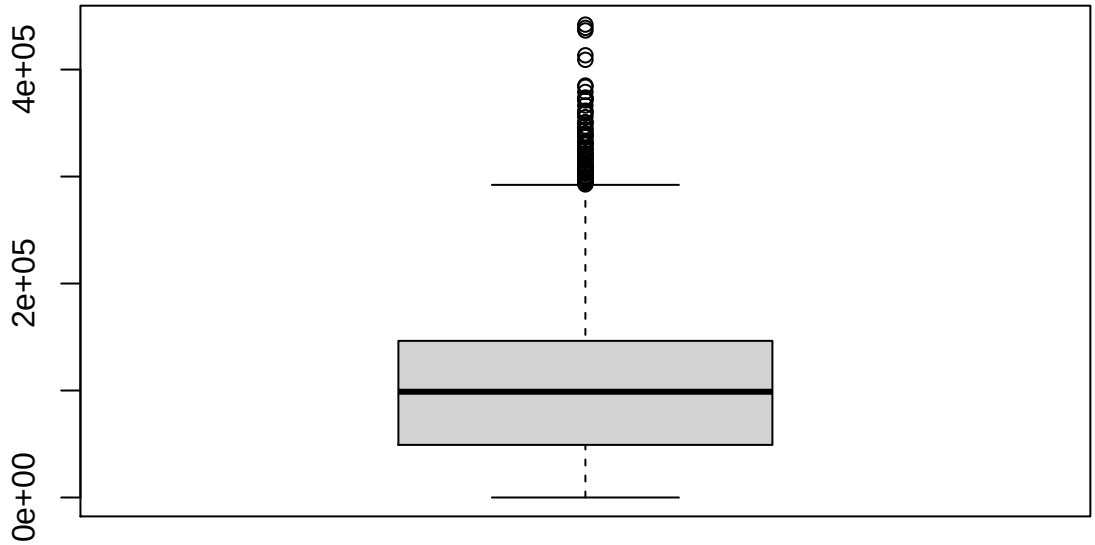
```
## [1] 0
```

Le `P-value` de *0* nous permet de réfuter H1 et donc de valider H0. De plus le plot nous permet de constater une dépendance linéaire positive entre les deux caractéristiques bien que la large répartition des points indique que le nombre de dommage infligés n'est pas le seul critère déterminant.

Nous pouvons donc conclure que le nombre de dommage influgés par un joueur a une grande influence sur la quantité d'xp qu'il recevra en fin de partie. Cela peut s'expliquer aisément car cette valeur est un bon indicateur des performances d'un joueur.

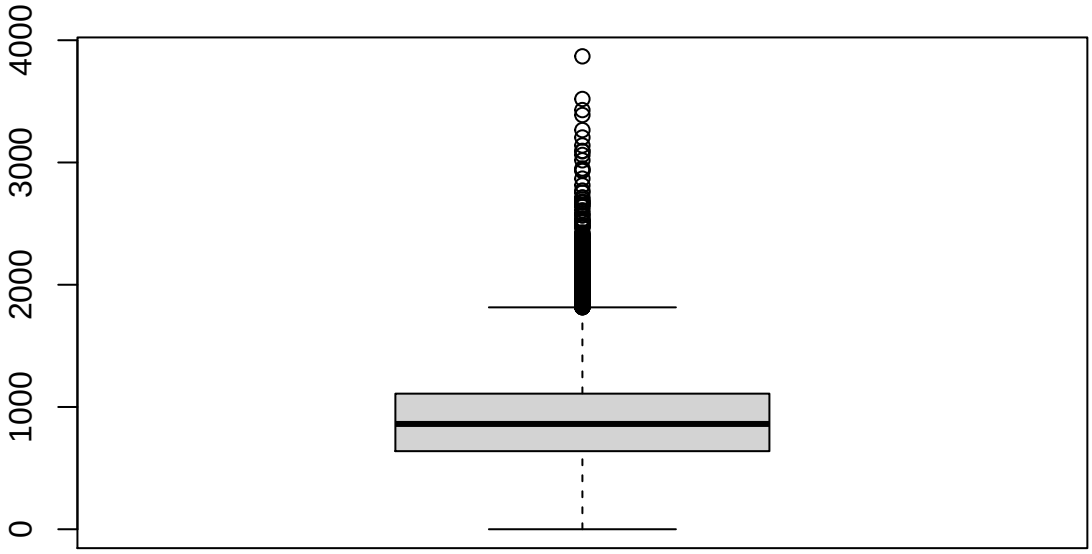
En enlevant les valeurs abérentes

```
# Remove unnecessary values (we remove players that gains no xp and players that dealt no dammages, bec
dd_a <- boxplot(data$totalDamageDealt)$out
```



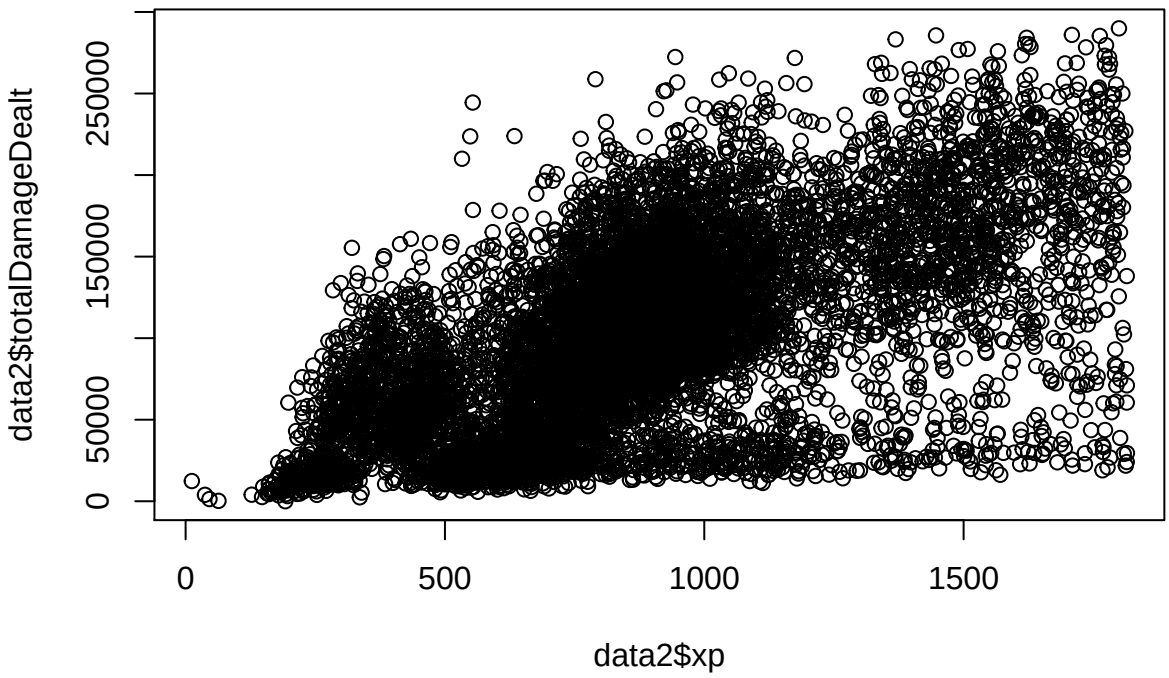

```
data2 <- subset(data, !(totalDamageDealt %in% dd_a))

xp_a <- c(0, boxplot(data2$xp)$out)
```




```
data2 <- subset(data2, !(xp %in% xp_a))

plot(data2$totalDamageDealt ~ data2$xp)
```




```
cor.test(data2$xp, data2$totalDamageDealt)$p.value
```

```
## [1] 0
```

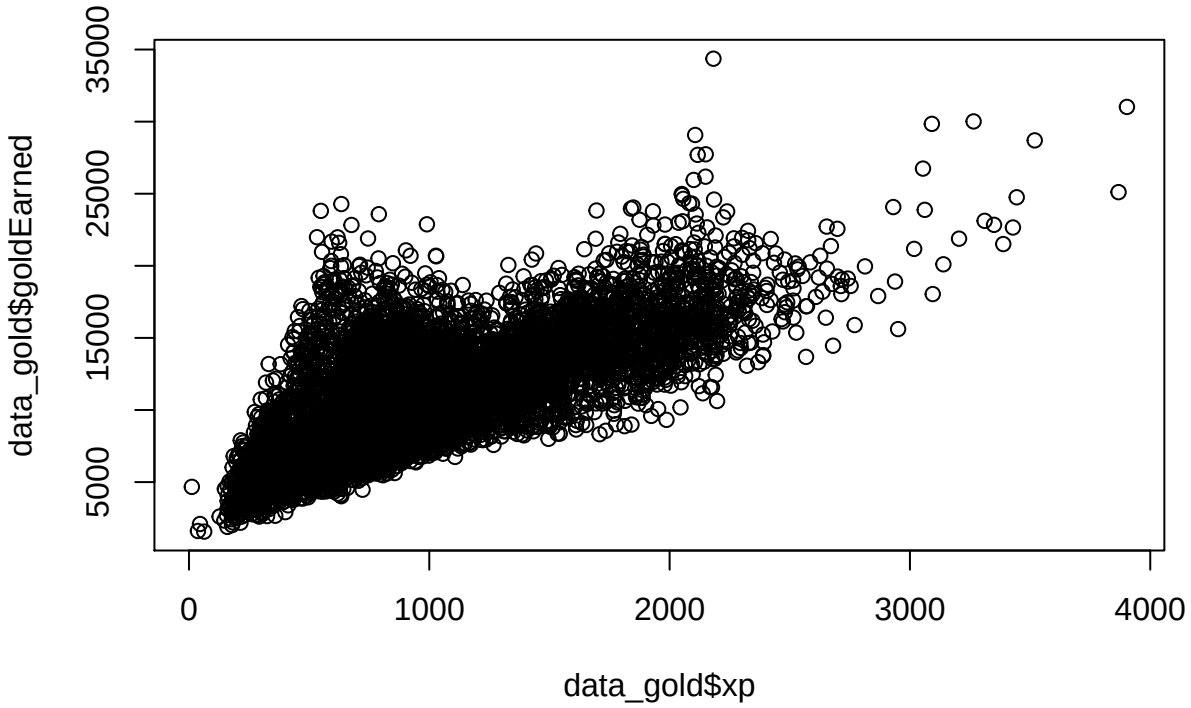
La quantité de golds récupérés par un joueur influe sur son total d'experience

Dans League of legends, les golds permettent d'acheter des objets permettant d'améliorer certaines caractéristiques de son personnage pour la durée du match ou des consommables pour regagner de la vie par exemple. Nous nous sommes demander si l'or n'avait pas un interet suplémentaire dans le gain d'experience d'un joueur.

H0 La quantité de gold récupéré a une influence sur la quantité d'experience en fin de match d'un joueur

H1 La quantité de gold récupéré n'a pas d'influence sur l'experience de fin de match

```
# Remove unnecessary values (players with 0 xp or golds because this is due to innactivity)
data_gold <- filter(data, goldEarned != 0)
data_gold <- filter(data_gold, xp != 0)
plot(data_gold$goldEarned ~ data_gold$xp)
```



Afin de réfuter H1, nous effectuons un test d'indépendance entre le nombre de gold récupérés et le total d'experience en fin de match.

```
cor.test(data_gold$goldEarned, data_gold$xp)$p.value
```

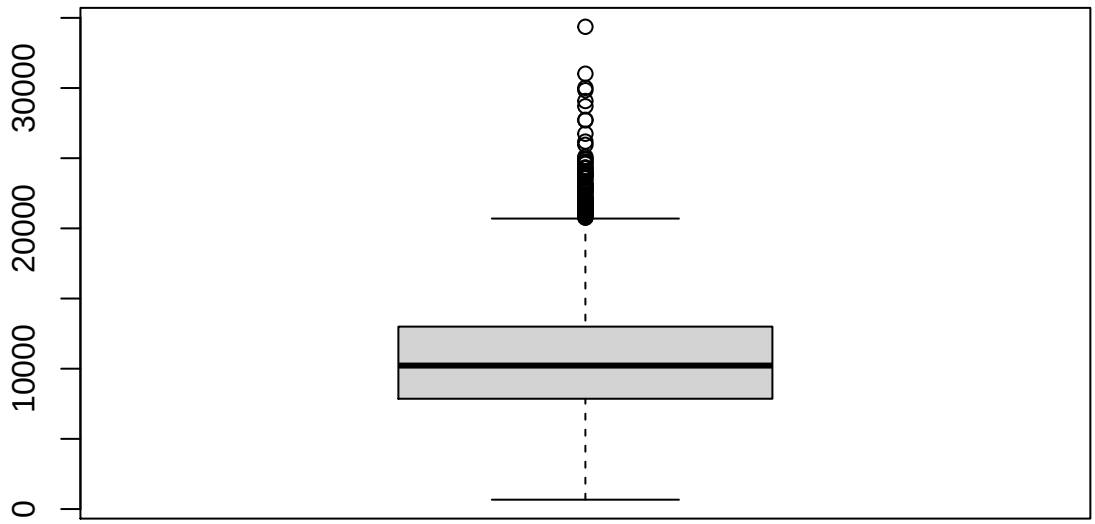
```
## [1] 0
```

Le p-value de 0 nous permet de réfuter largement H1 et donc de vérifier H0. De plus nous pouvons constater une dépendance linéaire positive entre les deux caractéristiques.

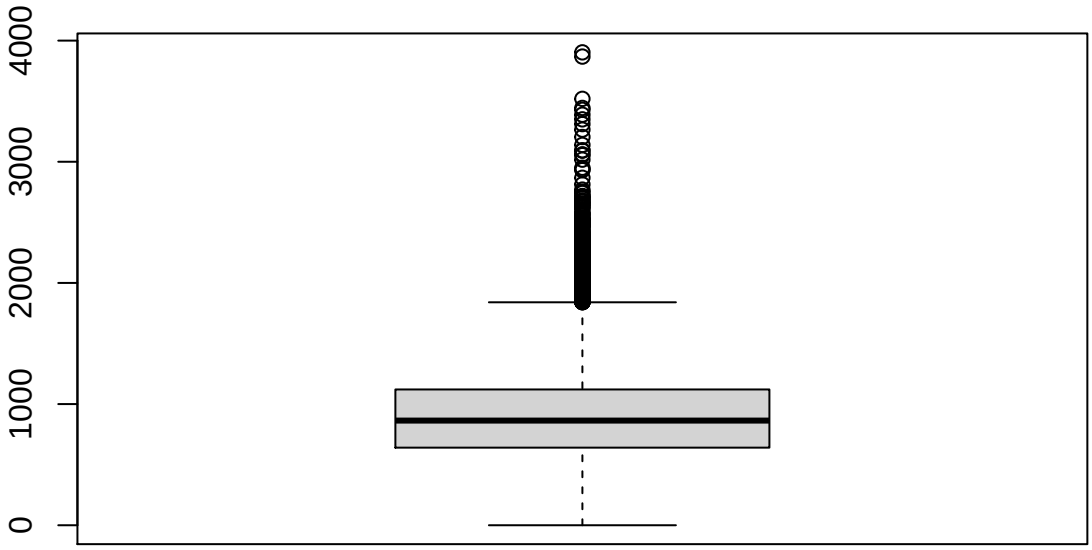
Ces résultats nous permettent largement d'affirmer que la quantité de gold récupéré par un joueur influe sur son total d'experience en fin de combat. Cela peut s'expliquer par une mention de la quantité de gold récupérer dans le calcul de l'experience mais aussi par la mention d'une caracteristque corolaire à l'or comme le nombre de minions tués par un joueur ou le temps de jeu.

On enlève les valeurs abérentes

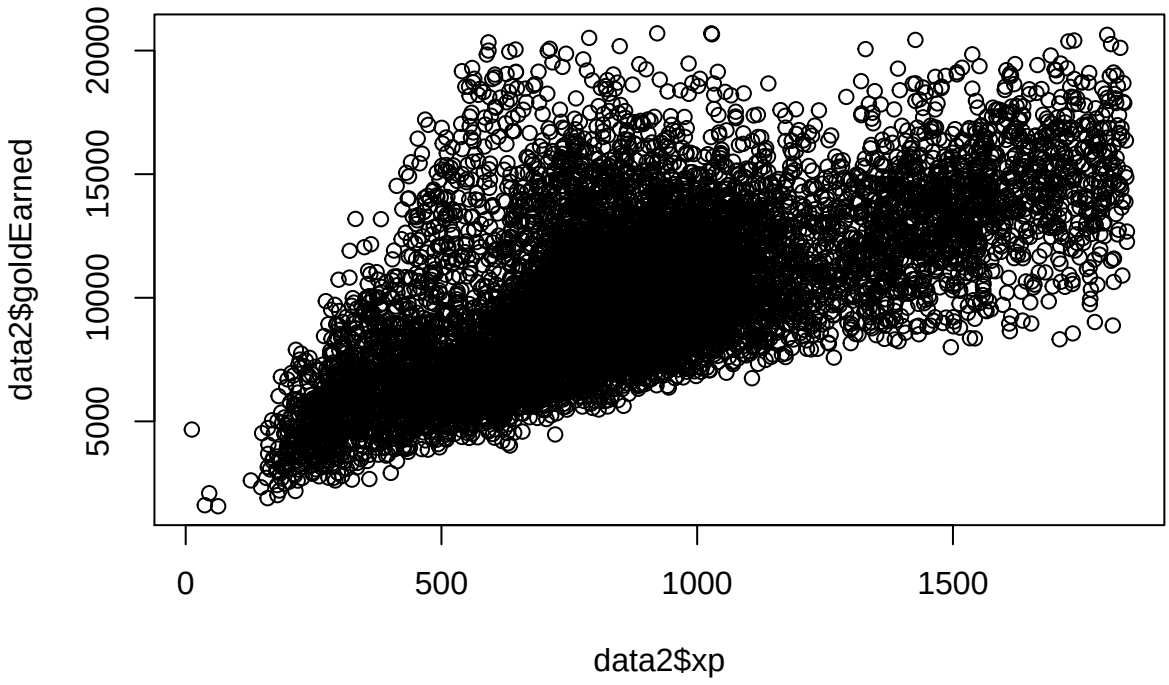
```
# Remove unnecessary values (players with 0 xp or golds because this is due to innactivity)
gold_a <- boxplot(data$goldEarned)$out
```




```
data2 <- subset(data, !(goldEarned %in% gold_a))
xp_a <- c(0, boxplot(data$xp)$out)
```




```
data2 <- subset(data2, !(xp %in% xp_a))
plot(data2$goldEarned ~ data2$xp)
```




```
cor.test(data2$goldEarned, data2$xp)$p.value
```

```
## [1] 0
```

Même en retirant les valeurs abérentes il nous reste un p-value de 0. Nous emmetons donc quelques reserves quand à ce résultat.

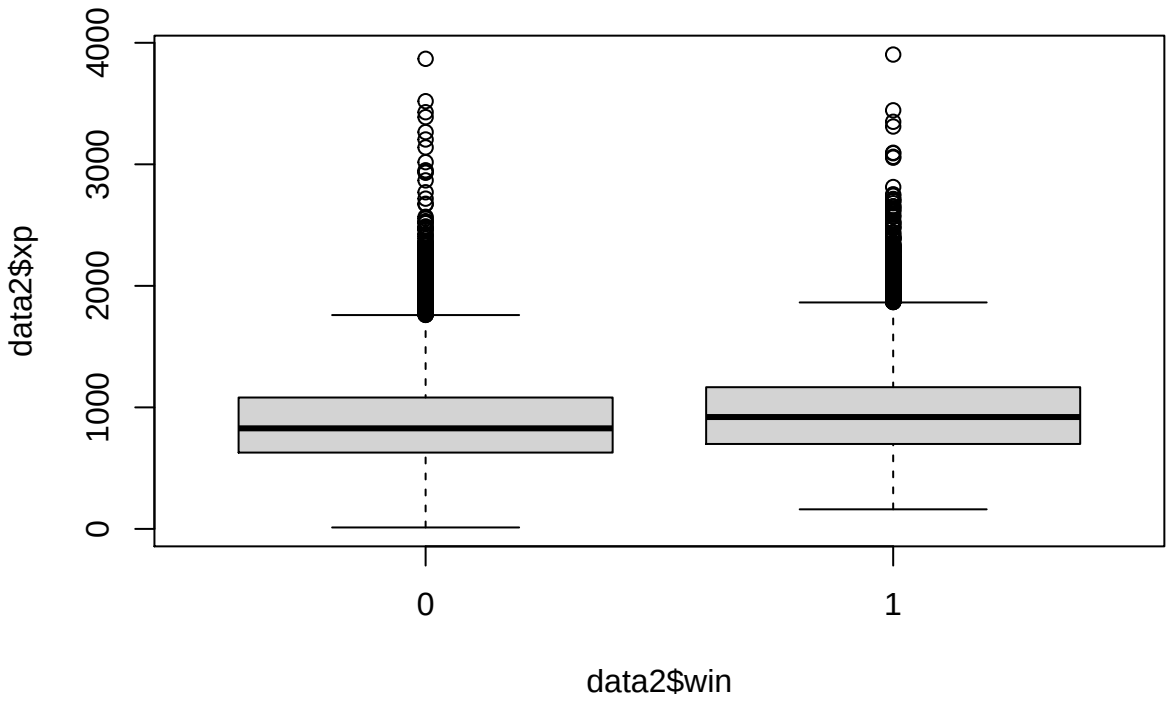
La victoire ou la défaite d'unjoueur a une influence sur son total d'xp en fin de partie

Comme n'importe quel jeu, vidéo ou non, l'objectif principal reste toujours le même. Gagner. Le gain d'une partie est dans de nombreux jeux récompensé. Soit par un bonus pour la partie suivante ou alors par l'obtention d'un objet ou de point vous glorifiant devant lers autres joueurs. C'est donc en toute logique que nous nous sommes demandé si les joueurs des équipes gagnantes de League of Legends étaient récompensés par de l'experience suplémentaire en fin de combat.

H0 La victoire d'une équipe a une influence sur la quantité d'xp récupérée par ses joueurs

H1 La victoire d'une éuipe n'a pas d'inluence sur la quantité d'experience perçue par le joueur en fin de partie

```
# Remove irrelevent players (with 0 xp)
data2 <- filter(data, xp != 0)
boxplot(data2$xp ~ data2$win )
```




```
data3 <- data2[order(data2$win),]
m = matrix(data3$xp, nrow = 2, byrow = TRUE)
# chisq.test(m)$p.value
```

Le p-value de 0 nous permet de réfuter l'hypothèse H1 et donc d'approuver H0. La victoire semble donc jouer un role très important sur le total d'experience gagné par un joueur en fin de partie. L'experience servant à améliorer un champion,

> Le résultat "0" nous semble très étrange. Quand nous regardons le boxplot ainsi que la matrice, on remarque une différence mais minime. Le p value de 0 signifie une corélation évidente entre la

victoire et la quantité d'experience remportée or le total d'experience est majoritairement calculé pendant la partie, ce n'est pas ce que nous constatons avec ce p-value

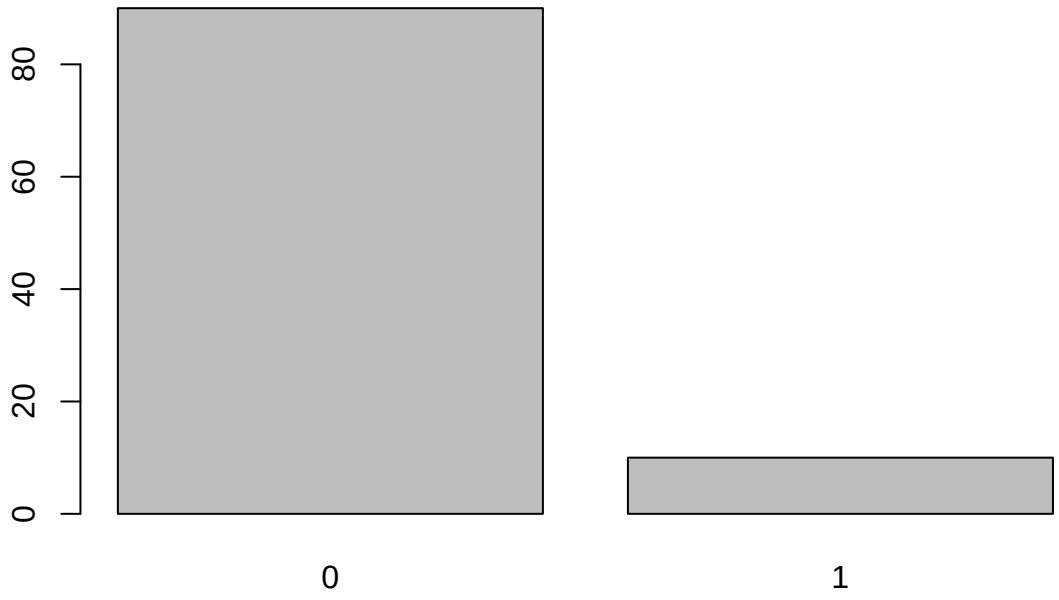
Etude de l'importance d'un firstBlood (premier kill) dans l'expérience gagnée

Etude préalable sur firstBlood

```
# extraction des données pour l'étude du firstBlood
data_firstBlood <- data.frame(c(data[,"firstBloodKill"]), c(data[,"xp"]));
# nommage des colonnes
names(data_firstBlood) <- c("fb", "xp");
# filtrage des colonnes avec un firstBlood (certains fb sont à -1 s'il n'y avait pas de données)
data_firstBlood <- subset(data_firstBlood, fb > -1)
```

Tout d'abord, nous allon voir la répartition des firstBlood. Normalement, comme il y n'y en a qu'un seul par partie, la répartition devrait être de 90% de 0 et 10% de 1. Nous allons simplement voir si les données confirment ceci. (comme on a enlever des données, cela peut très légèrement varier).

```
barplot(freq(data_firstBlood$fb)$`%`, names.arg = row.names(freq(data_firstBlood$fb)))
```



On voit clairement sur le graphique que l'on approche des **90-10** prévu, on le voit encore mieux grâce à une table des fréquences :

```
freq(x = data_firstBlood$fb)
```

```
##      n  % val%
## 0 9361 90   90
## 1 1039 10   10
```

Avec ses 2 possibilités seulement, la répartition des **firstBlood** ressemble beaucoup à une répartition selon

la loi binomiale, avec 1 chance sur 10 d'avoir un firstBlood. On peut alors, comme avec les **xp**, créer des fonction pour calculer des probabilités et simuler des matchs.

```
pfb <- function(q, n){
  return(pbinom(q, n, 1/10));
}

dfb <- function(q, n){
  return(dbinom(q, n, 1/10));
}

rfb <- function(n){
  return(rbinom(1, n, 1/10));
}
```

Maintenant, on peut, par exemple, calculer : 1. la probabilité de faire au moins un **firstBlood** dans les 3 prochaines parties 2. la probabilité de réaliser aucun **firstBlood** lors des 30 prochaines parties 3. simuler 15 parties et savoir combien de **firstBlood** on aurait eu

```
1 - pfb(0,3) # probabilité de faire au moins 1 firstBlood dans les 3 prochaines parties
```

```
## [1] 0.271
```

```
dfb(0,30) # probabilité de réaliser 0 firstBlood en 30 partie
```

```
## [1] 0.04239116
```

```
rfb(15) # simuler 15 partie (le résultat est le nombre de firstBlood que l'on aurait eu)
```

```
## [1] 2
```

Etude de la corrélation entre les firstBlood et l'experience gagné

Nous allons d'abord étudier comment les **firstBlood** et l'**xp** gagné sont corrélés. Pour cela, nous allons poser les hypothèses suivantes :

H0 : les **firstBlood** et les **xp** sont indépendant H1 : il existe une corrélation entre les 2 caractères

```
cor.test(x = data_firstBlood$xp, y = data_firstBlood$fb)$p.value;
```

```
## [1] 0.002682401
```

On obtient un p value de *0.003*, ce qui est très bien. On peut rejeter H0, et donc accepter H1, avec un taux de confiance de **99.7%**.

Conclusion

En conclusion, on peut dire avec une très grande confiance qu'il y a une corrélation entre le fait d'effectuer le **firstBlood** et le fait de gagner beaucoup d'**xp**. Cependant, cela ne veux pas dire que l'un est une conséquence de l'autre, on peut juste savoir qu'ils sont liés.

Parties personnelles

Anthony

J'ai travaillé sur le parseur en JAVA ainsi que sur les tests d'hypothèse sur le nombre de golds récupérés, l'importance de la victoire et l'importance des dommage . J'ai rencontré de nombreux problème car je n'ai trouvé aucun résultat très interessant sur les tests d'hypotheses (que des p-value de 0) et malgré le temps passé sur le jsujet, je ne trouve pas de réponse. J'ai eu bien moins de temps que Kilian pour travailler car je travaille à temps plein pour mon alternance (même pendant les vacances de Noël).

Je connais assez peu le jeu League of legends et du peu que j'y ai joué je peux juste dire que je ne l'aime pas du tout. Nous avons choisi de faire un sujet original en choisissant de travailler sur l'experience reçu par les joueurs. Comme ce sujet nous demandait d'avoir les données par joueur et non pas par équipe (à certaines exceptions près), nous avons préféré modifier la strucutre du dataset pour avoir les données par joueur.

Nos résultats sont en incohérence totale avec la formule donnée par Riot games :

- Si victoir : `TEMPS_DE_JEU * 0.11 + 6.6`
- Si défaite : `TEMPS_DE_JEU * 0.09 + 5.4`

Kilian

Personnelement, j'ai écrit l'introduction, la partie sur les données et sur les règles.

Ensuite, j'ai travaillé sur la partie avec les **firstBlood** et avec tous les **kills d'affilés**. Je trouvais cela interessant d'avoir plusieurs observations groupées sur un sujet. Cela nous a permis de voir quels **killstreak** peremttait d'avoir le plus d'expérience.

Enfin, j'ai fait toute la partie sur l'étude unidimensionnel sur la variable **xp**. Cela m'a permit de retravailler les loi de probabilités.

D'un côté personnel, c'est sur ce projet (et grâce au projet de dataviz), que j'ai vraiment pu prendre R en main et faire ce que je voulais. J'ai vraiment appris à manipuler les différentes structures de données proposées par R et les différentes fonctions implémentées dans le langage ainsi que les bibliothèques externes.