Analisis de Estrategia de LoL

Pau Amores Giner, Sergio Catalan Ruiz, Luminita Ciobanu Borinschi, Javier Elena Navarro, Rustam Suleimanov

2024-05-12

Table of Contents

Introducción

Contexto del estudio

El estudio ha sido realizado sobre el juego de League of Legends. League of Legends (LoL) es un videojuego multijugador en línea de batalla en arena (MOBA). En cada partida, dos equipos de cinco jugadores compiten para destruir la base del equipo contrario. Los jugadores seleccionan personajes, conocidos como campeones, cada uno con habilidades únicas y funciones específicas dentro del equipo. Además, antes de comenzar la partida, los jugadores pueden bloquear ciertos campeones para evitar que los use el equipo contrario, una fase conocida como "baneo", y seleccionar hechizos y runas que mejorarán ciertas características base de un campeón.

Descripcion de la base de datos

La base de datos utilizada fue obtenida de kaggle, esta nos aportó información de de 60000 partidas de alto nivel, con más de 600 variables para cada una, pero pasaron por varios procesos de reducción, uno de ellos previo a cargarlo en R dejándonos 395 variables, 39 por cada jugador y 5 generales de la partida.

Objetivos del proyecto

Este estudio se enfoca en analizar los factores que influyen en la victoria de una partida de LoL. Utilizando técnicas estadísticas avanzadas como el Análisis de Componentes Principales (PCA), el Análisis de Correspondencias (CA), el Clustering y el análisis PLS, buscamos entender mejor como las decisiones tomadas durante la fase de selección y bloqueo de campeones, así como el rendimiento de los jugadores durante la partida, impactan en el resultado final. La base de datos utilizada para este análisis incluye registros detallados de partidas de un alto nivel de juego, proporcionando una rica fuente de información para identificar patrones y tendencias útiles para muchos jugadores inexpertos que desean mejorar jugando a este videojuego. Nos centramos en el aspecto individual porque la victoria de un equipo depende del desempeño de sus integrantes y de los del equipo rival, pero solo puedes controlar tus acciones, por eso solo exploraremos lo que debe priorizar cada jugador de manera individual para aumentar la tasa de victoria en sus partidas.

Analisis exploratorio inicial y preproceso de los datos

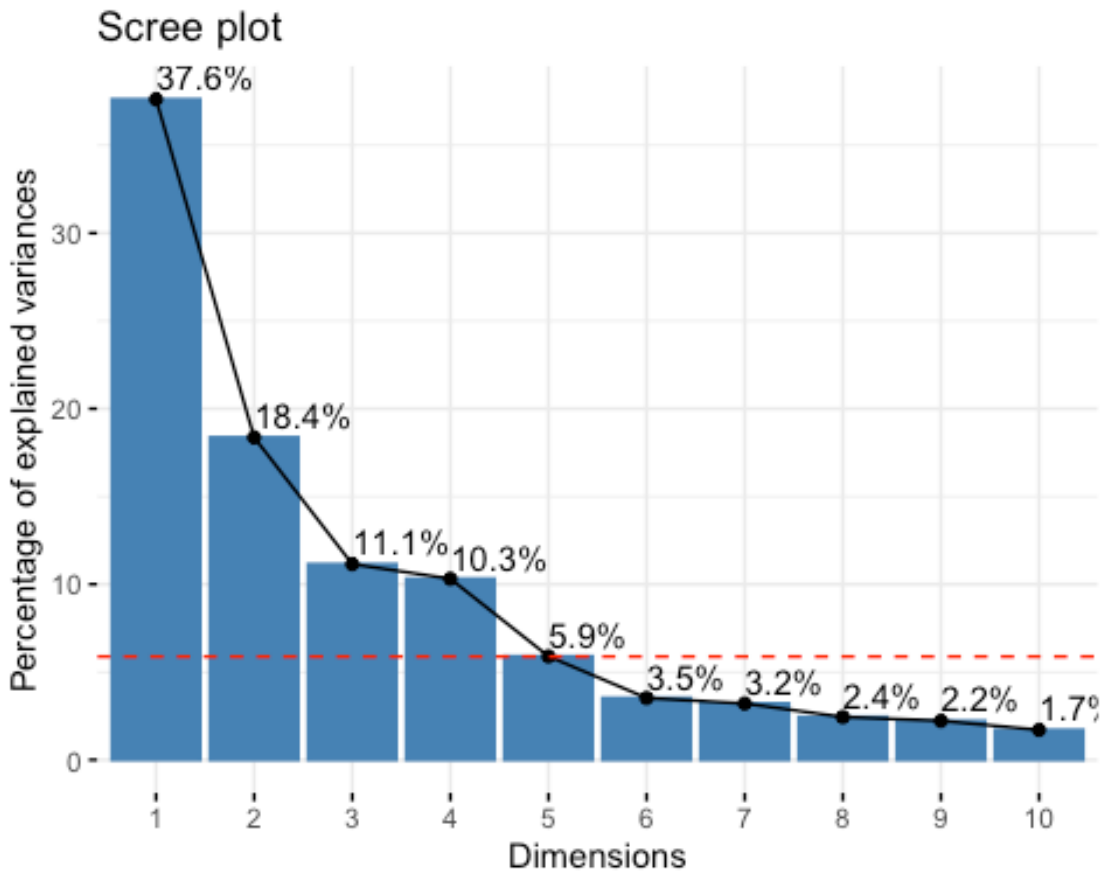
Introducimos el dataset y acabamos de eliminar las variables desechadas, también nos quedamos únicamente con 300 partidas para poder habilitar todos los estudios. Las variables que consideramos más relevantes para el estudio deseado son las siguientes, 16 numéricas y 8 categóricas: assists = cuando participas en un asesinato pero no das el golpe final, aporta oro pero en menor medida que una kill lvl = este es el nivel con el que acabas la partida, este depende de xp mayormente ganado de los minions que mueren cerca tuyo, así como de las kills y assists que tengas. El lvl máximo es el 18, y aumentar el nivel de aporta más poder. deaths = número de veces que has muerto, esto otorga oro a la persona que te ejecuta, además que te evita conseguir xp al es no estar donde mueren los minions por tener un tiempo de espera antes de reaparecer gold = este se consigue dando el golpe que ejecuta a minions y con kills y assists. Este te permite comprar objetos que te otorgan poder. kills = cada vez que ejecutas un rival, esto te otorga 300 oros y xp Gold10 y gold15 = oro que tienes al min 10 y al 15 xp10 y xp15= xp que tienes al min 10 y al 15 minions = npcs que has ejecutado, esto te otorga oro por cada uno jungla = minions neutrales, estos se comportan como los minions normales, otorgan oro, pero estos los ejecutan los junglas mayormente, los cuales se curan al hacerles daño tDMGd = daño total realizado a jugadores tDMGt = daño total recibido de jugadores heal = vida curada vision = puedes colocar guardianes de visión Duration = duración de la partida champ = id del personaje que elige para jugar la partida Rol = posición que tiene el jugador Win = si gana o no la partida ban= campeón que baneas del equipo rival Summoner1 y summoner2 = hechizos de invocador previos a la partida Runa1 y runa2 = estadísticas que eliges mejorar antes de la partida

Después de determinar las variables que eran más relevantes para el estudio realizamos un escalado y normalización de los datos, sobretodo por el hecho de que al tener partidas de tiempos muy dispares el valor de las variables numéricas se disparaba en partidas largas.

Al enfocar el estudio individual separamos las partidas por cada jugador, convirtiendo cada fila en 10, una por cada jugador, pasando de tener 300 observaciones/partidas a 3000observaciones/jugadores, reduciendo las variables de las 395 iniciales a las 24 mencionadas. En la base de datos no habían datos faltantes, además la única variable a la que se tuvo que aplicar una transformación fue a win, dado que los jugadores del equipo 2 cuando el equipo 1 ganaba ellos tenían valor 0, por lo cual invertimos los valores para el equipo 2

PCA

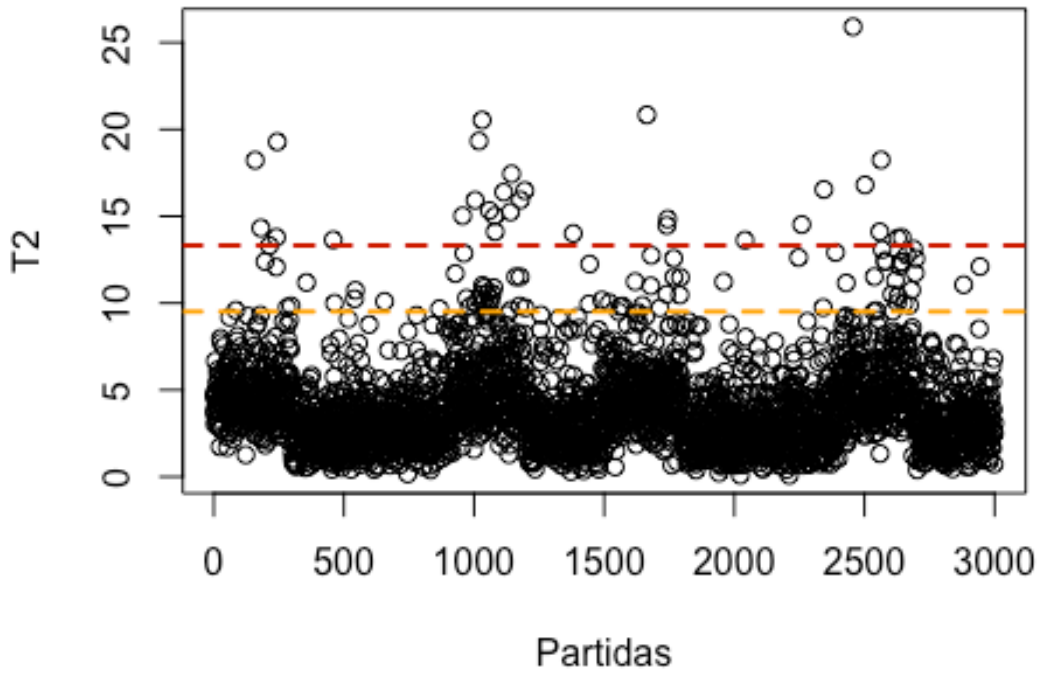
```
eig.val2 <- get_eigenvalue(res.pca2)
VPmedio = 100 * (1/nrow(eig.val2))
fviz_eig(res.pca2, addlabels = TRUE) +
  geom_hline(yintercept=VPmedio, linetype=2, color="red")
```




```
K = 4
```

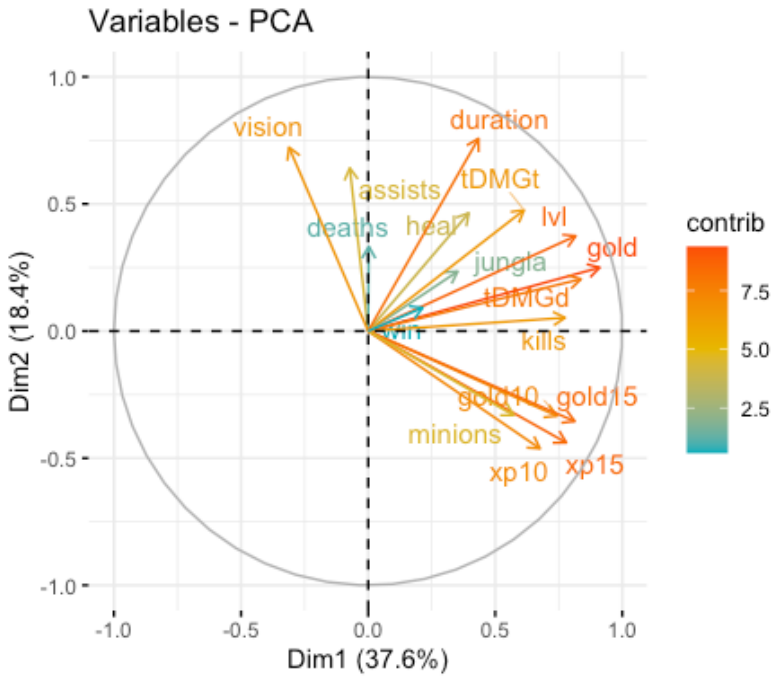
Para seleccionar el número de componentes principales más adecuad generamos el gráfico del codo, añadido la recta que indica la varianza explicada por cada PC si todas explicaran los mismo y la tabla con la varianza explicada por cada componente. Seleccionamos 4 componentes, que explican un 77% del total de variabilidad de los datos, porque cumplen tanto con el criterio del codo como con el de superar la "varianza media" explicada por cada componente.

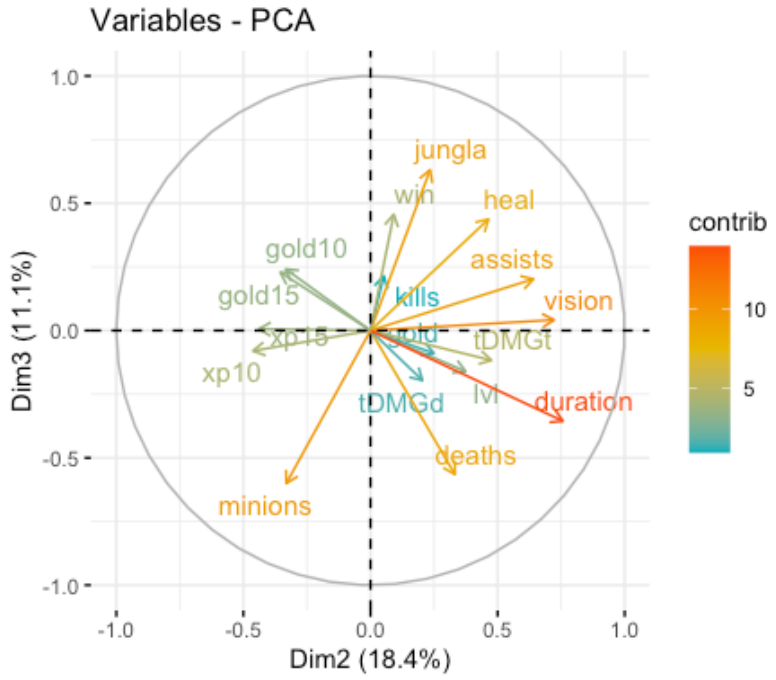
```
res.pca2 = PCA(tabla, scale.unit = TRUE, graph = FALSE, ncp = K,
               quali.sup = which(descGame$tipo == "categorical"))

misScores = res.pca2$ind$coord[,1:K]
miT2 = colSums(t(misScores**2)/eig.val2[1:K,1])
I = nrow(tabla)
F95 = K*(I**2 - 1)/(I*(I - K)) * qf(0.95, K, I-K)
F99 = K*(I**2 - 1)/(I*(I - K)) * qf(0.99, K, I-K)

plot(1:length(miT2), miT2, type = "p", xlab = "Partidas", ylab = "T2")
abline(h = F95, col = "orange", lty = 2, lwd = 2)
abline(h = F99, col = "red3", lty = 2, lwd = 2)
```

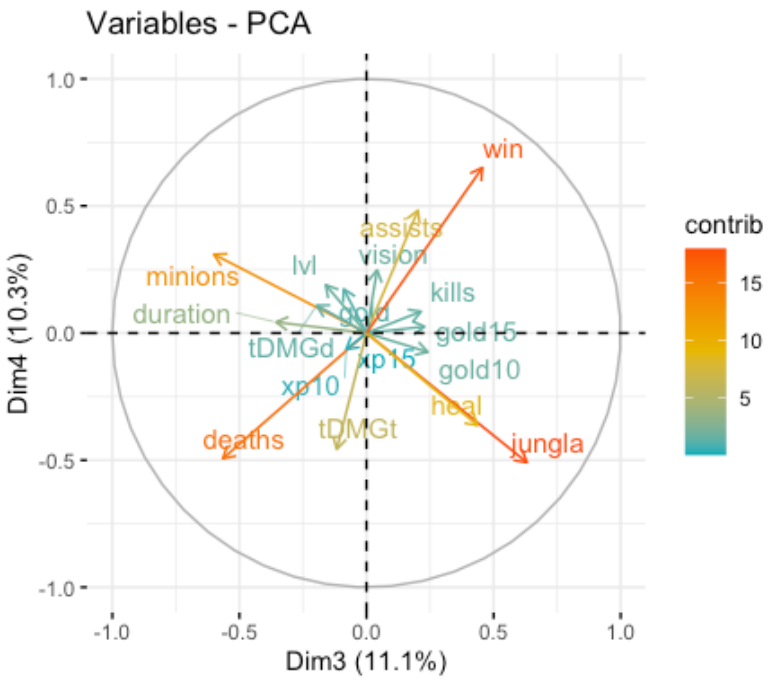



```
anomalas = which(miT2 > F95)
```

Detectamos los anómalos usando la T2 de Hotelling, y eliminamos los extremos ya que pueden condicionar nuestro PCA y resultar en interpretaciones erróneas.


Variables - PCA
vision
duration
assists
tDMGt
deaths
heal
lvl
gold
jungla
tDMGd
win
kills
gold10
gold15
minions
xp10
xp15
contrib
7.5
5.0
2.5
1.0
0.5
0.0
-0.5
-1.0
Dim2 (18.4%)
Dim1 (37.6%)


Variables - PCA
jungla
win
heal
gold10
assists
gold15
kills
vision
xp15
gold
tDMGt
xp10
tDMGd
lvl
duration
deaths
minions
contrib
10
5
1.0
0.5
0.0
-0.5
-1.0
Dim3 (11.1%)
Dim2 (18.4%)



Conclusiones

En base a estos gráficos de puntuaciones, podemos ver varias relaciones claramente diferenciadas:

las variables que más contribuyen a la primera componente son gold y lvl , las cuales además están fuertemente correlacionadas. Esta relación se da en la mayoría de partidas reales, ya que a más nivel del jugador durante la partida, más oro habrá ganado, mientras que duration, vision y assists son las que más contribuyen a la segunda componente y también están positivamente relacionadas

Las variables win, assists, heal, kills, jungla y vision están fuertemente correlacionadas, sugiriendo que estos factores pueden estar relacionados con el rendimiento general del equipo.

las variables que más contribuyen a la tercera componente son ,jungla y mninions, las cuales están negativamente correlacionadas, lo que sugiere que los minions de linea (minions) no suelen ser asesinados si se asesinan muchos minions de jungla (jungla), y viceversa. Esto parece correcto, ya que el rol de jungla en el lol se suele ocupar de asesinar todos los minions de jungla, mientras que el resto de roles suelen asessinar únicamente minions de línea.

la variable deaths (muertes) está negativamente relacionada con el oro, xp y win, lo cual es bastante intuitivo, y está positivamente relacionado con el total de daño que ha recibido el jugador, la duración de la partida y el nivel (esta última es debido a que cuanto más dura una partida, más muertes hay y por tanto, el nivel con el que acabas la partida es mayor)

la variable jungla está muy relacionada con heal y gold, lo cual parece válido, ya que los jugadores de rol jungla (que suelen asesinar minions de la jungla) se curan más que el resto de roles, y la mayoría del oro del jungla viene de los minions de la jungla, por lo que la cantidad de minions de la jungla estará relacionada con el oro.

las variables que más contribuyen a la cuarta componente son ,win jungla y deaths, y se aprecia una fuerte correlación negativa entre win y deaths, lo cual tiene sentido ya que en las partidas cuanto más mueras más ventaja tiene el equipo contrario,

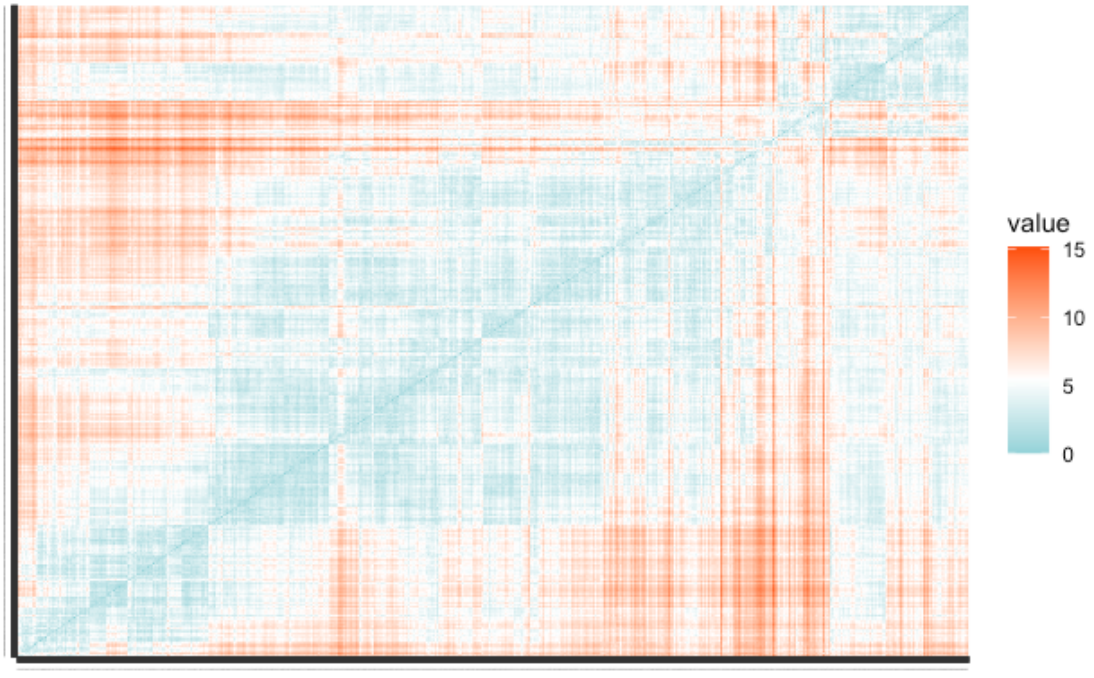
Clustering

Carga de datos

Una vez tratdos los datos en el apartado PCA y visto su resultado, se realizara un analisis de clustering que dividira un conjunto heterogéneo de elementos que en nustro caso son los jugadores de league of legends en grupos, en función de las similitudes o diferencias entre ellos.

Tendencia de agrupamiento

Antes de pasar al analisis de clusering habria que ver inicialmente si en nuestros datos existe una tendencia de agrupamiento, para ello vemos la siguiente matriz de distancias.

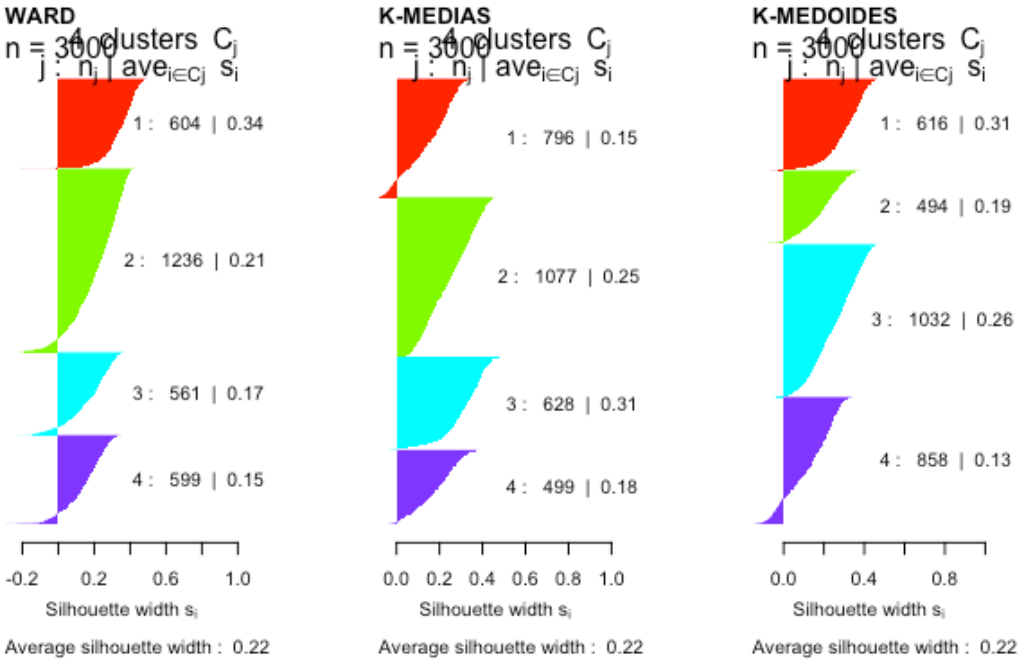


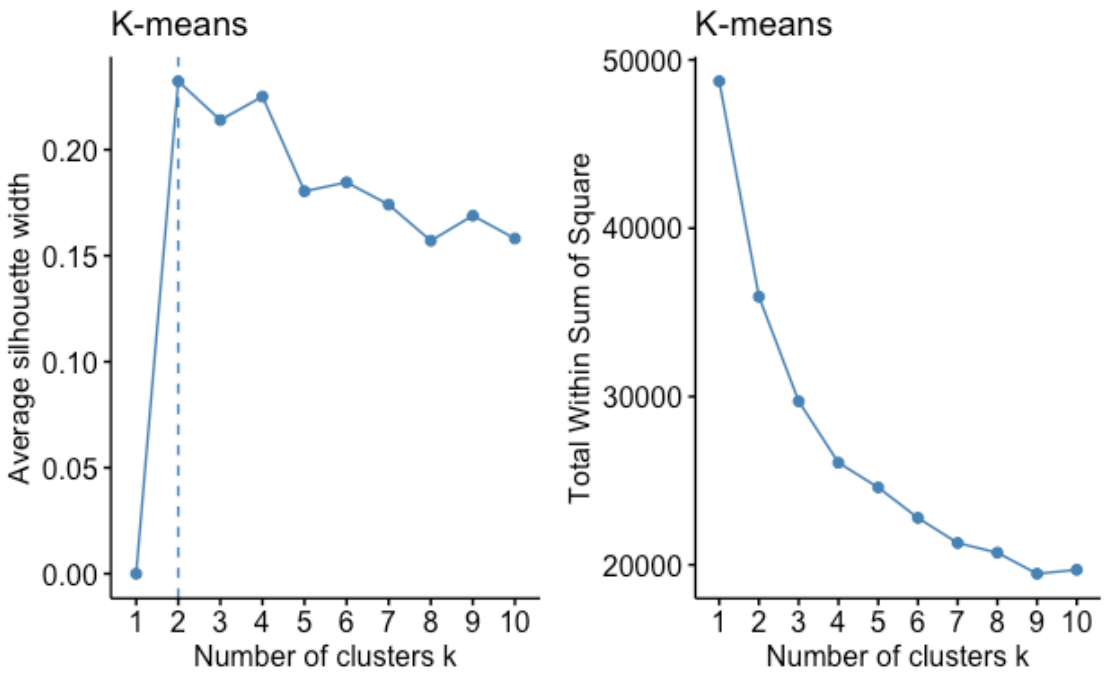
Los sombreados rojos indican una tendencia de formacion de un grupo de individuos que se puede ver a simple vista que son varios. Para confirmar dicha tendencia de agrupamiento se calcula el coeficiente de Hopkins el cual al ser mas proximo a 1

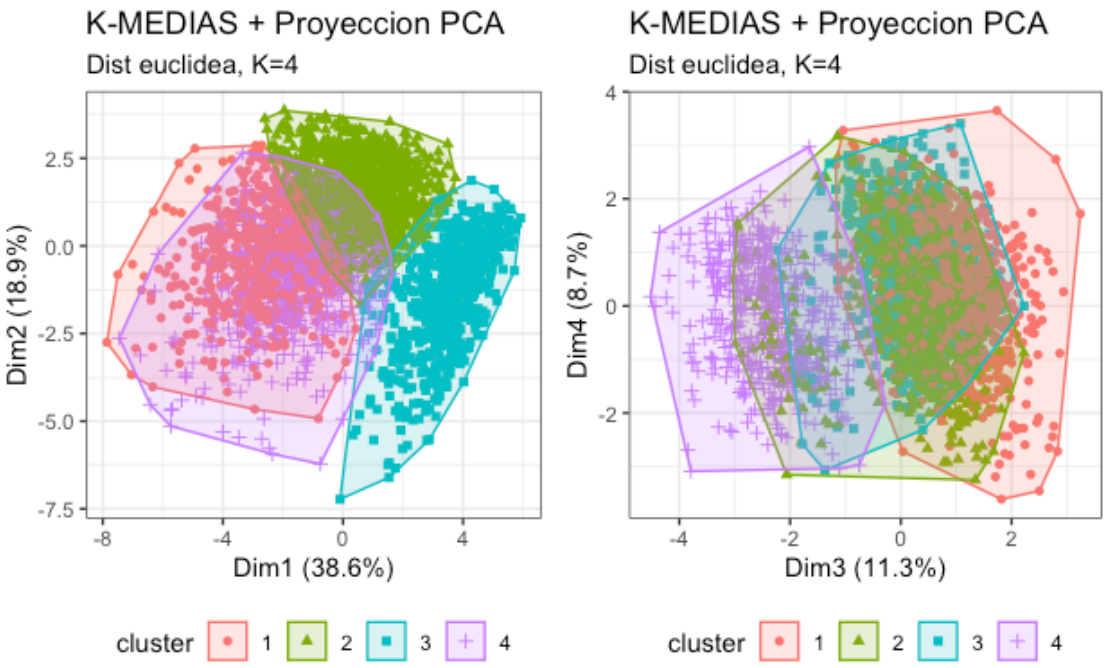
(0.878) indica una buena tendencia de agrupacion. Dicho coeficiente en estos datos es bastante alto, por ende significa una buena tendencia de agrupacion.

Una vez visto todo esto, se probaran diferentes modelos jerarquicos y de particion y se quedara con el mejor de ellos.

Seleccion del modelo

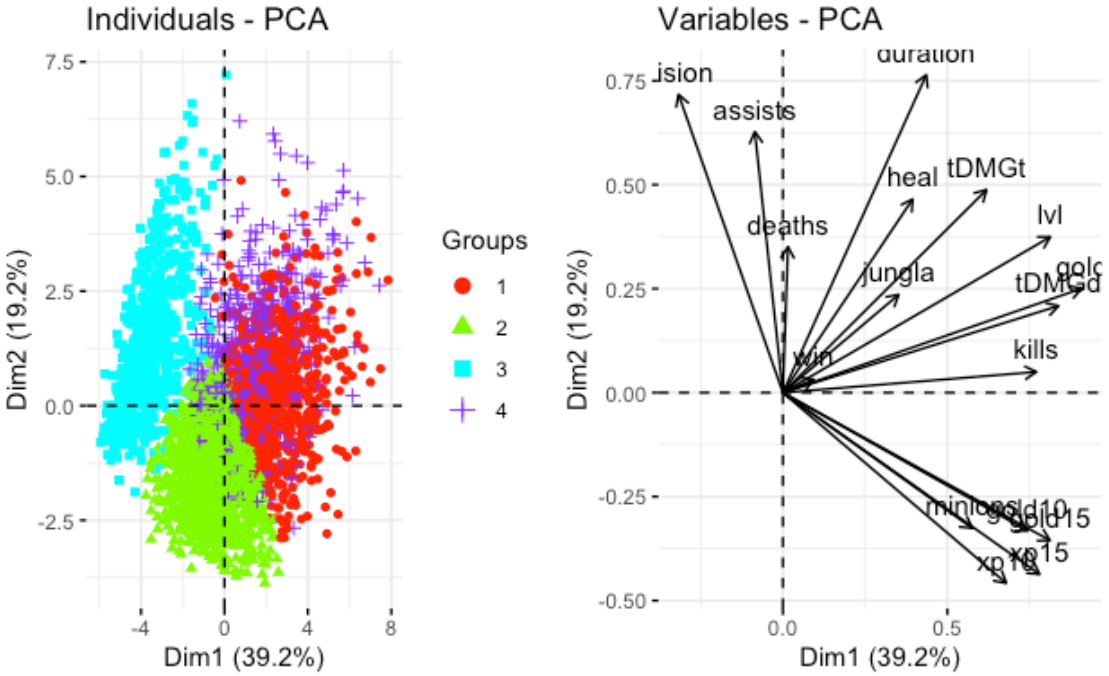






El metodo escogido en este caso ha sido el de K-medias debido a que agrupa mucho mejor los datos que el resto de los metodos. Y se han escogido 4 clusters en dicho metodo con el criterio del metodo de Codo, que se basa en escoger el cluster que mayor variabilidad explique teniendo en cuenta que se busca disminuir la SCR. Por ello, la opcion mas optima han sido 4 clusters.

Conclusiones



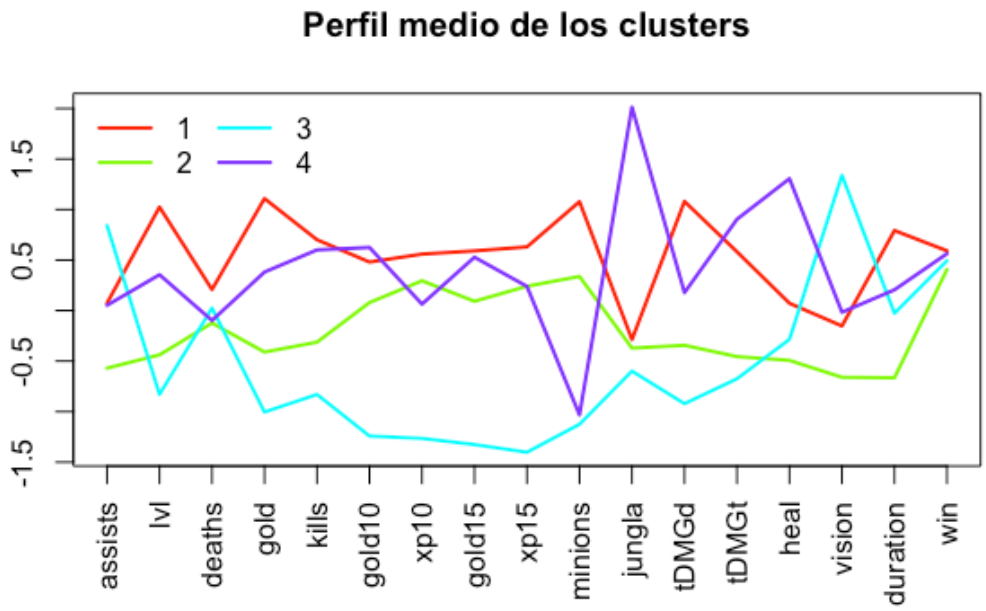
Despues de sacar este grafico la interpretacion de cada cluster es la siguiente:

- Clúster 1 (Rojo) Este grupo se caracteriza por enfocarse en el combate directo y la obtención rápida de recursos. Son jugadores agresivos, acumulando numerosas eliminaciones y recolectando oro y experiencia de manera eficiente durante todas las fases del juego. Son los encargados de realizar el daño y acaparar el oro y el nivel del equipo. Es por esta razón, que su contribución a la partida es determinante en el desarrollo de la partida. Los jugadores de este grupo suelen tener los mejores resultados.

- Clúster 2 (Verde) Este grupo se caracteriza por un estilo de juego extremadamente pasivo y de escasa contribución al equipo. Estos jugadores suelen presentar un bajo número de muertes, asistencias, daño recibido e infligido, así como una escasa provisión de visión en el mapa. Tienden a jugar de manera individualista, lo que implica que, en lugar de ser un jugador activo para el equipo, a menudo se convierten en una carga debido a su mínima participación y apoyo en las estrategias conjuntas, y por tanto, queda un poco relegado. A pesar de tener pocas muertes, su escasa contribución al juego en términos de apoyo y ejecución de objetivos los hace menos efectivo y valioso para el éxito del equipo.

- Clúster 3 (Azul claro) Este grupo coincide con un rol especifico del juego, que es el de soporte. Se caracteriza por un estilo de juego más cooperativo, de forma que se centran en apoyar a los jugadores más relevantes en la partida para que estos destaquen. El rol de soporte no necesita recursos para ser importante en la partida y este, se sacrifica para beneficiar a sus compañeros. Aunque tienden a causar poco daño, destacan por su elevado número de asistencias y puntos de visión del mapa. Este tipo de jugador puede identificarse como un jugador de apoyo. A pesar de su baja contribución en términos de daño y eliminaciones, representan un pilar fundamental en las peleas en equipo (teamfights), teniendo funciones muy diversas como iniciar peleas, curar a los aliados, paralizar al enemigo entre otras. Son esenciales para la coordinación y el éxito del equipo.

- Cluster 4(Morado) Este grupo también coincide con un rol específico del juego, que es el de jungla. Este rol tambien se centra en ayudar al equipo, pero de una forma más activa que los soportes. Contribuyen en la mayor parte de los objetivos de la partida y pueden ayudar en todos los sectores del mapa con más facilidad que los soportes. En este rol la función es bastante variable, ya que pueden acumular recursos o cederlos al resto del equipo dependiendo de la partida y el personaje. Destaca en la variable de jungla, que representa la principal forma de ganar oro y experiencia de este rol. También tienen un valor alto de la variable heal, ya que a medida que van ganando oro matando minioms en la jungla se van curando. Además, tienen un valor de minioms bajo, porque los junglas se centran en ganar oro y experiencia de una forma diferente al resto de jugadores. Estos jugadores suelen ser más decisivos en el éxito del equipo que los soportes, pero menos que los jugadores rojos.



En este grafico se ve mucho mejor la diferencia de caracteristicas de cada tipo de grupo explicado anteriormente (sobretodo rojos y morados). La conclusión de estos clusters aplicandos al contexto de league of legends está dividida en dos partes. Si el jugador es un tipo de jugador enfocado a aportar al equipo, el objetivo de este debe ser apoyar a los tipos de jugadores que más se parezcan a los del grupo rojo. Por otro lado, si el jugador es un tipo de jugador más lider, se recomienda seguir los pasos del grupo de jugadores rojos y evitar parecerse a los individuos del grupo verde.

Regresion PLS

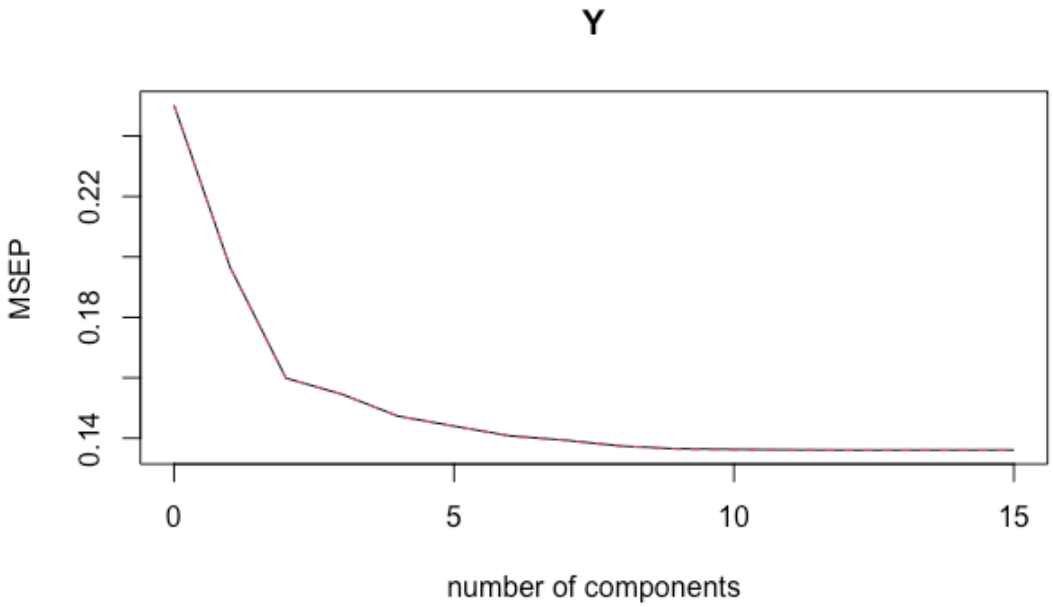
Preparacion de los datos

Y es la variable respuesta, que en este caso es el resultado del juego (win). X es la matriz de variables predictoras, que incluye estadísticas como asistencias, nivel, muertes, oro, asesinatos, etc.

Y se convierte en un vector numérico. X se asegura que sea un data.frame.

Se escalan los datos.

El gráfico generado es un “validation plot” que muestra cómo el MSEP cambia con el número de componentes en el modelo PLS.



Al principio, el MSEP desciende rápidamente a medida que se añaden más componentes, lo que sugiere que los primeros componentes capturan la mayor parte de la variación explicativa en Y. Concretamente los 2 primeros componentes.

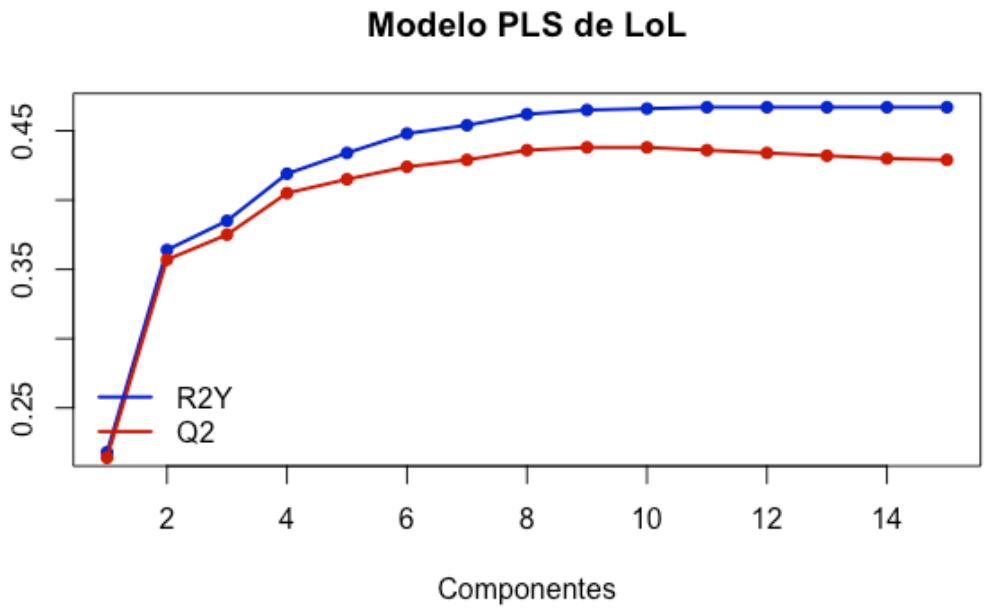
División de los datos en entrenamiento y prueba

Se realiza la división de los datos en conjuntos de entrenamiento y prueba, y luego se ajusta el modelo PLS utilizando el conjunto de entrenamiento.

Estos resultados sugieren que el modelo PLS tiene una capacidad razonable para explicar y predecir la variabilidad en la variable de respuesta.

Calculo de R2

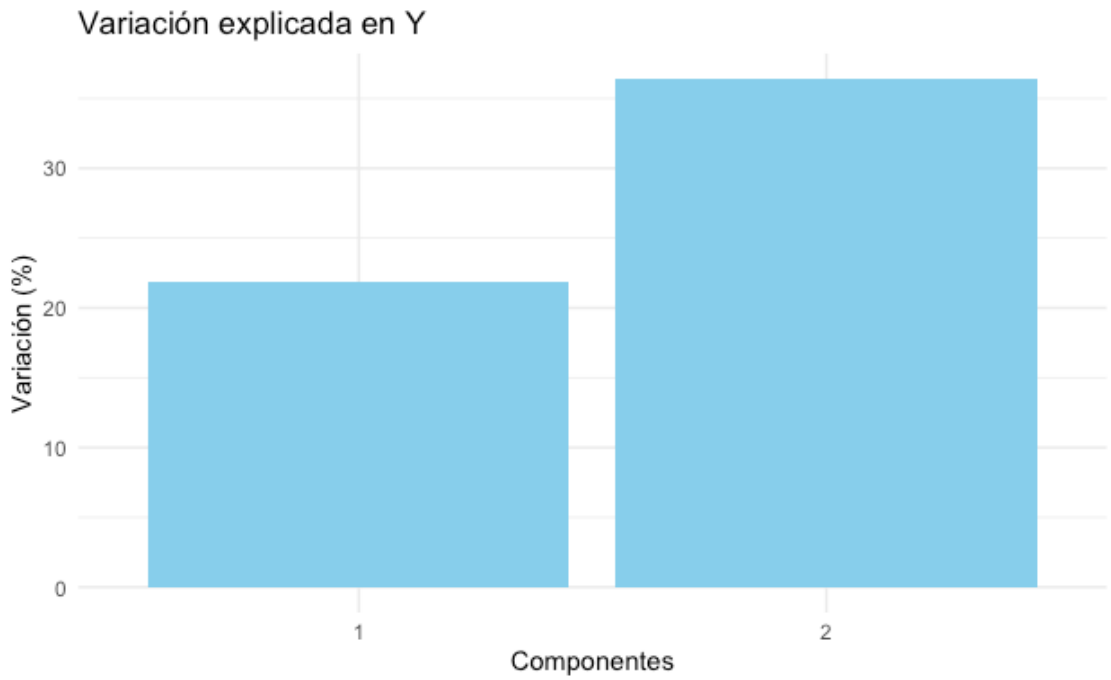
Se calculan los valores de R2 para el espacio de las respuestas y se evalúa la variación explicada por componente.



Visualizacion

Gráfico de Variación Explicada (por Componente): Muestra el porcentaje de variación en Y explicado por cada componente.

```
## PLS
## 2400 samples x 15 variables and 1 response
## standard scaling of predictors and response(s)
##       R2X(cum) R2Y(cum) Q2(cum) RMSEE pre ort pR2Y  pQ2
## Total    0.314    0.218   0.214 0.442   1   0 0.05 0.05
## PLS
## 2400 samples x 15 variables and 1 response
## standard scaling of predictors and response(s)
##       R2X(cum) R2Y(cum) Q2(cum) RMSEE pre ort pR2Y  pQ2
## Total    0.512    0.364   0.357 0.399   2   0 0.05 0.05
```



Componente 1: Explica aproximadamente un 20% de la variación en Y.

Componente 2: Explica aproximadamente un 40% de la variación en Y. Esto sugiere que añadir un segundo componente mejora significativamente la capacidad del modelo para explicar la variación en Y, incrementando la variabilidad explicada a un 40%.

El gráfico sugiere que el modelo PLS-DA con 2 componentes es bastante efectivo, explicando el 40% de la variación en Y. Añadir más componentes después del segundo no aporta significativamente a la explicación de la variación en Y.

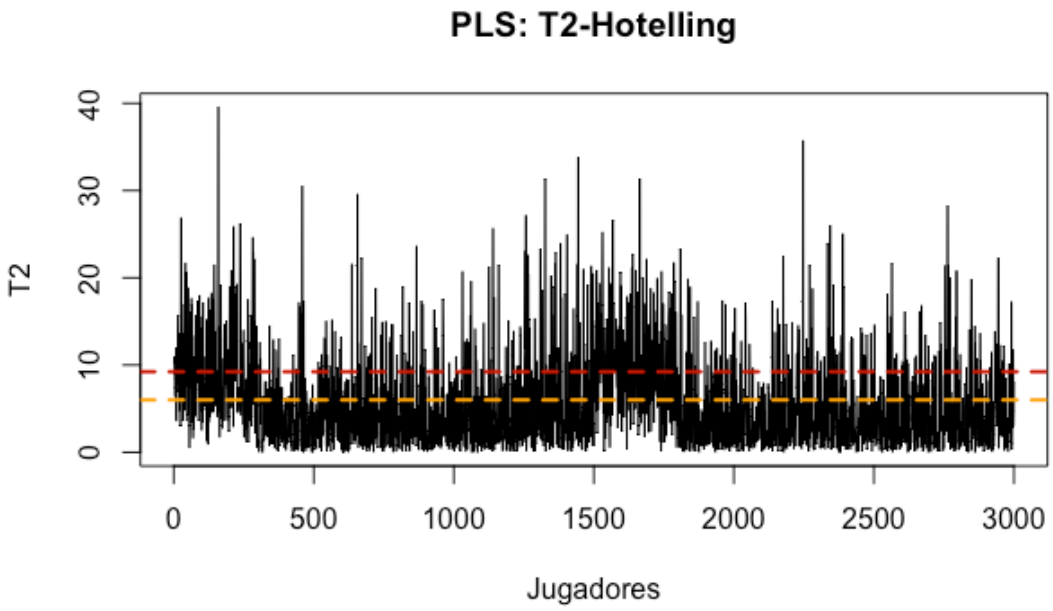
Ajuste

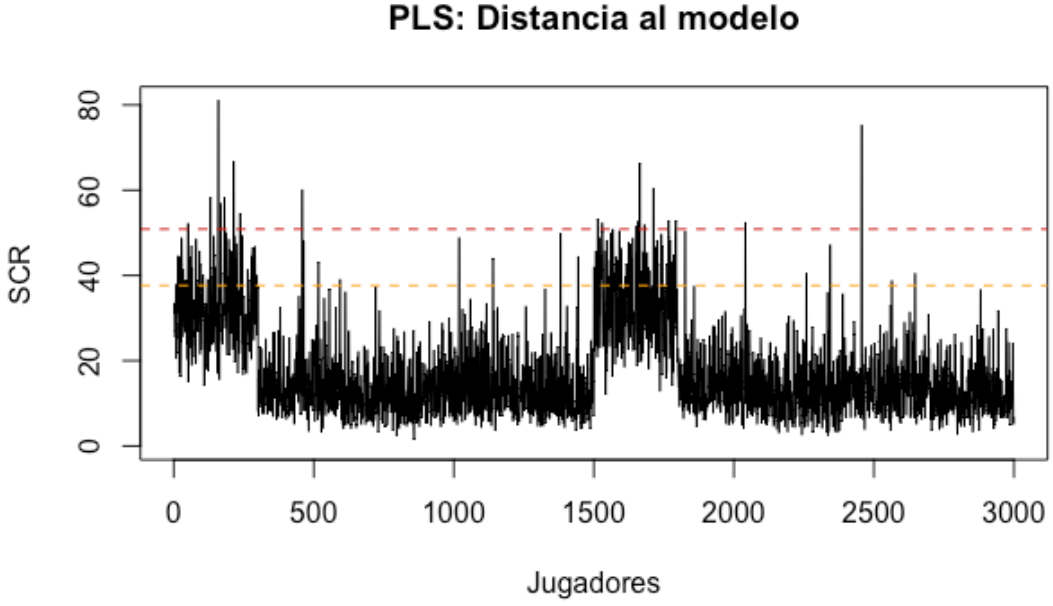
Se ajusta el modelo con el número óptimo de componentes.

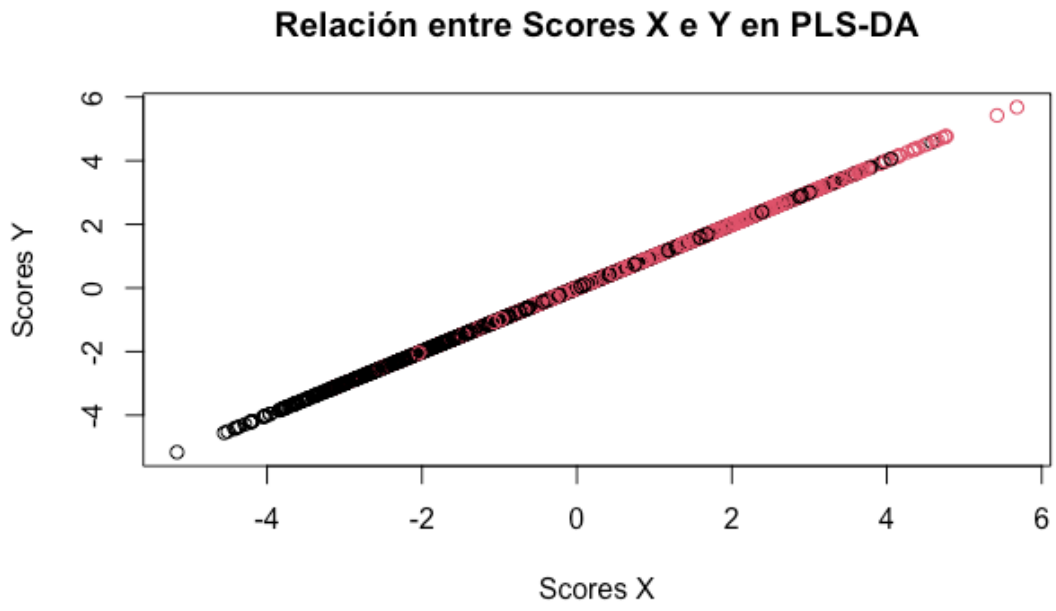
Con el número óptimo de componentes, ajustamos el modelo PLS. Los siguientes gráficos y análisis nos ayudan a evaluar la calidad del modelo.

Validacion del modelo

Detección de anómalos: Se utiliza la estadística T2 de Hotelling y la distancia al modelo (SCR). Relación entre scores: Se comparan los scores t y u para verificar la relación lineal.


PLS: T2-Hotelling
T2
0
10
20
30
40
0
500
1000
1500
2000
2500
3000
Jugadores


PLS: Distancia al modelo
SCR
0
20
40
60
80
0
500
1000
1500
2000
2500
3000
Jugadores

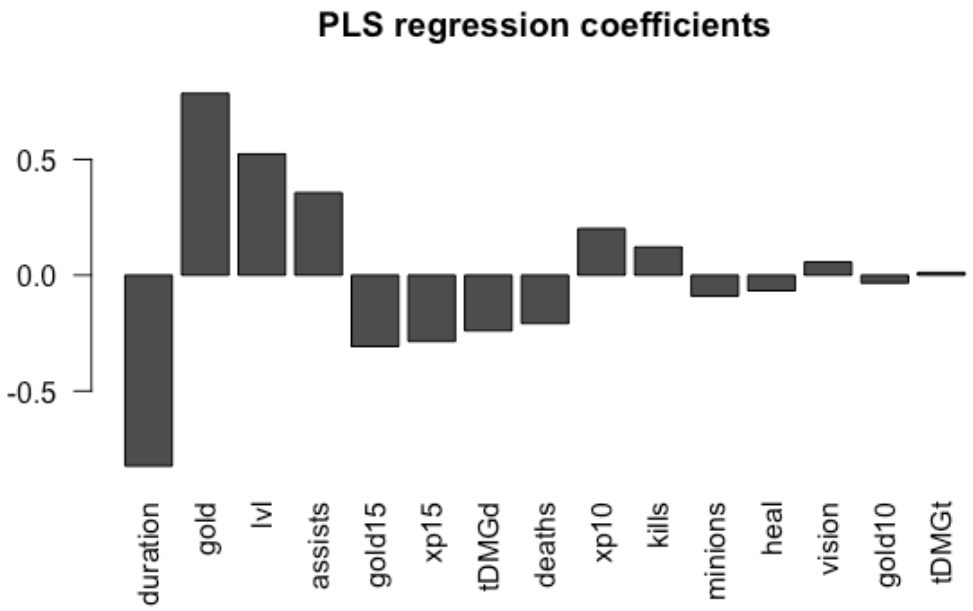


T2-Hotelling Valores por encima de F95 indican potenciales anómalos, y valores por encima de F99 indican anómalos severos.

Distancia al Modelo (SCR) Valores de SCR altos indican observaciones que son atípicas en el contexto del modelo. La línea de referencia (chi2lim) ayuda a identificar estos puntos atípicos.

Relación Lineal entre Scores La relación lineal fuerte entre los scores y Y indica que los componentes están bien alineados con la variabilidad en Y.

Coeficientes de Regresion



Este gráfico es útil para identificar las variables que tienen una mayor influencia en la predicción de la variable de respuesta. Las variables con coeficientes más altos son las que tienen una mayor importancia en el modelo. Las variables con coeficientes cercanos a cero tienen una influencia mínima en la predicción.

Conclusion

El modelo PLS-DA tiene una capacidad razonable para explicar y predecir la variabilidad en la variable de respuesta 'win' (resultado del juego). Esto se refleja en los resultados de la validación cruzada, donde se observa una disminución rápida en el error cuadrático medio de predicción (MSEP) a medida que se añaden más componentes al modelo.

Se observa que los dos primeros componentes explican aproximadamente el 40% de la variación en la variable de respuesta. Esto sugiere que estos componentes capturan una parte significativa de la variabilidad en los resultados del juego.

Se han utilizado estadísticas como T2 de Hotelling y la distancia al modelo (SCR) para validar el modelo y detectar posibles observaciones anómalas. Estas técnicas ayudan a identificar casos atípicos que pueden afectar la precisión del modelo.

La relación lineal entre los scores y la variable de respuesta indica que los componentes están bien alineados con la variabilidad en los resultados del juego. Esto sugiere que el modelo captura de manera efectiva la estructura subyacente de los datos.

Finalmente, el gráfico de coeficientes de regresión muestra las variables que tienen una mayor influencia en la predicción de los resultados del juego. Estas variables incluyen asistencias, nivel, muertes, oro, asesinatos, etc.

En resumen, el modelo PLS-DA es útil para predecir los resultados del juego en función de las variables predictoras seleccionadas.

AFC

Introduccion general

Como método opcional hemos decidido escoger el análisis factorial de correspondencias, nuestro objetivo en este apartado del proyecto será analizar todo lo relacionado con las variables que tienen que ver con las decisiones tomadas por el jugador antes de que empiece la partida. Para dar un poco más de contexto, estas variables pueden ser muy importantes en el desarollo de la partida ya que, hay personajes que tienen ventaja contra otros, o simplemente hay unas runas o hechizos de invocación elegidos previamente a la partida que tienen más importancia que otros dependiendo de la partida que se va a jugar. Por un asunto de espacio no vamos a proporcionar las conclusiones ni el procedimiento del afc múltiple. El código de este análisis será proporcionado en el anexo.

AFC Simple

En esta sección del proyecto nos centraremos en realizar un AFC simple sobre la base de datos original, sin realizar ningún tipo de transformación sobre esta. Las variables escogidas para este estudio son t1p1_champId y t1p1_ban_champId. Con estas variables intentaremos combrobar si hay algun tipo de dependencia entre el campeón seleccionado por el jugador y el personaje bloqueado de ese mismo jugador para que no pueda jugarse por el equipo rival. Estas dos variables están enfocadas en el rol de soporte del equipo.

Procedimiento

En primer lugar, cargaremos los datos y crearemos nuestra variable dataframe con los tipos de variables.

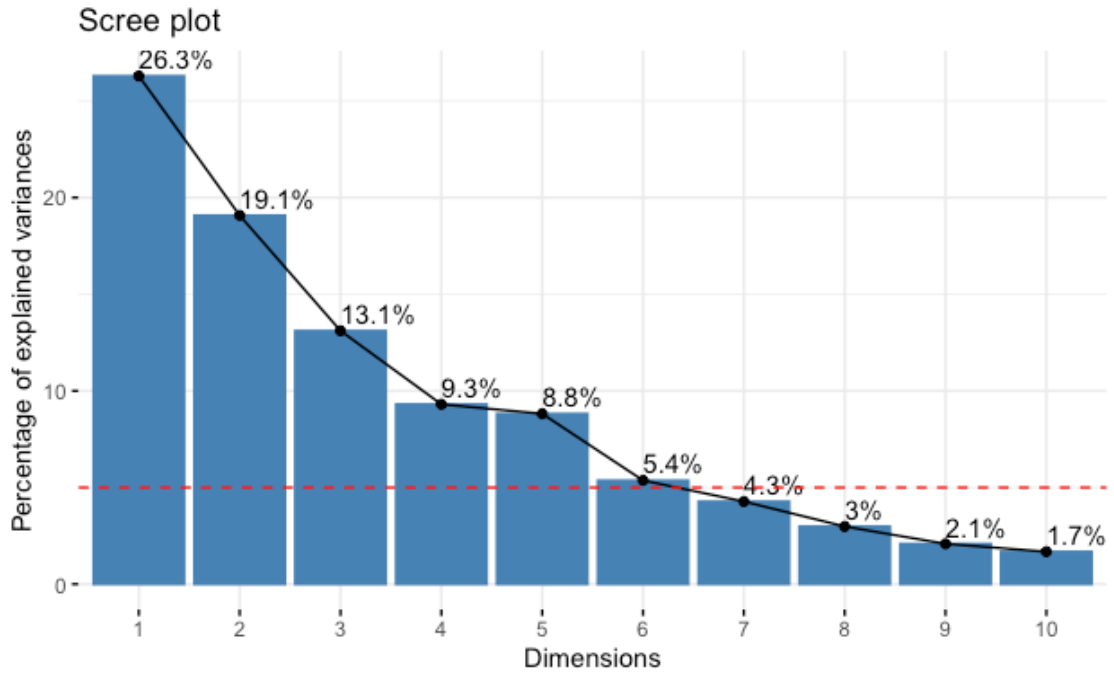
A continuación, creamos nuestras tablas de contigencia y de frecuencia junto con las variables de los valores de cada frecuencia marginal y condicional de las filas y las columnas. Por último, mostramos el valor del test de independencia. Si este valor está por debajo de 0,05 aproximadamente , entonces asegura que es útil realizar el afc porque existe algun tipo de dependencia entre estas dos variables.

El resultado del test es menor a 0,05. Podemos concluir que podria ser relevante realizar este análisis.

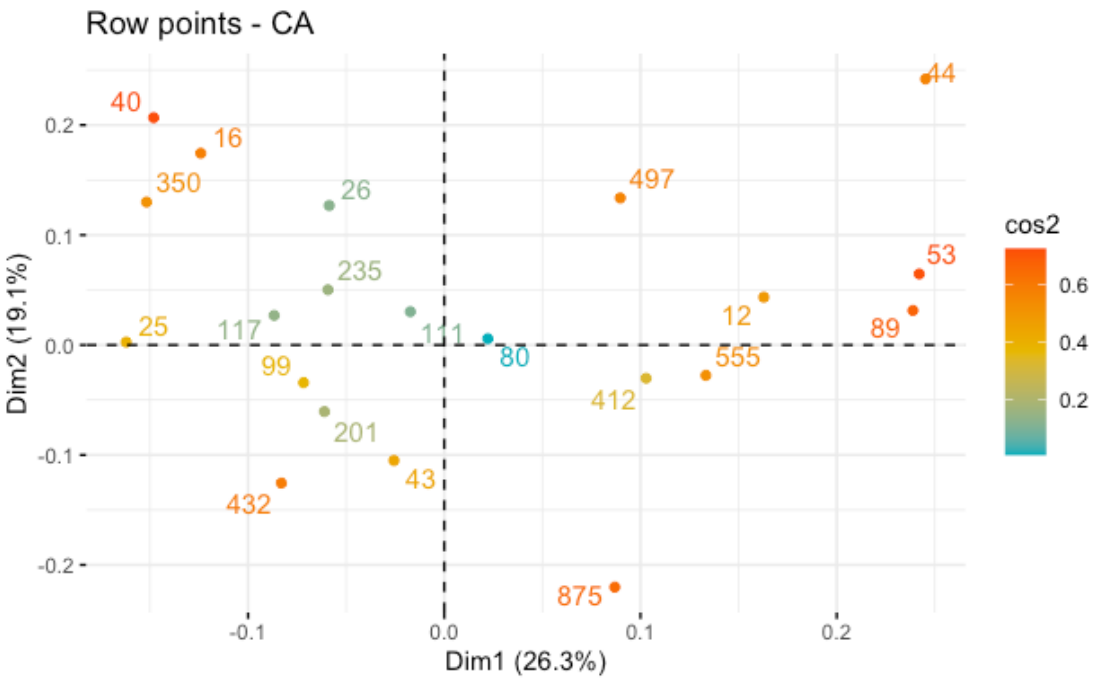
```
##
##  Pearson's Chi-squared test with simulated p-value (based on 2000
```

```
##  replicates)
##
## data:  children_filtrado
## X-squared = 2096.4, df = NA, p-value = 0.0004998
```

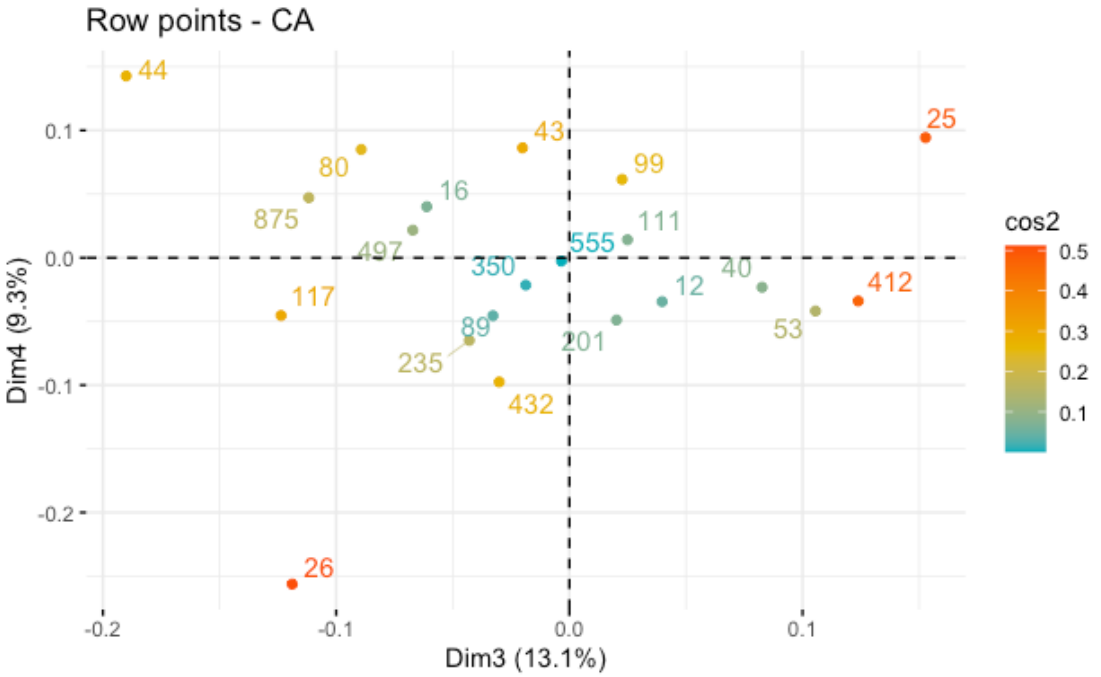
Seguidamente, vamos a utilizar las funciones CA,get_eigenvalue y fviz_eig para realizar el gráfico de las dimensiones y su porcentaje de variabilidad explicado.



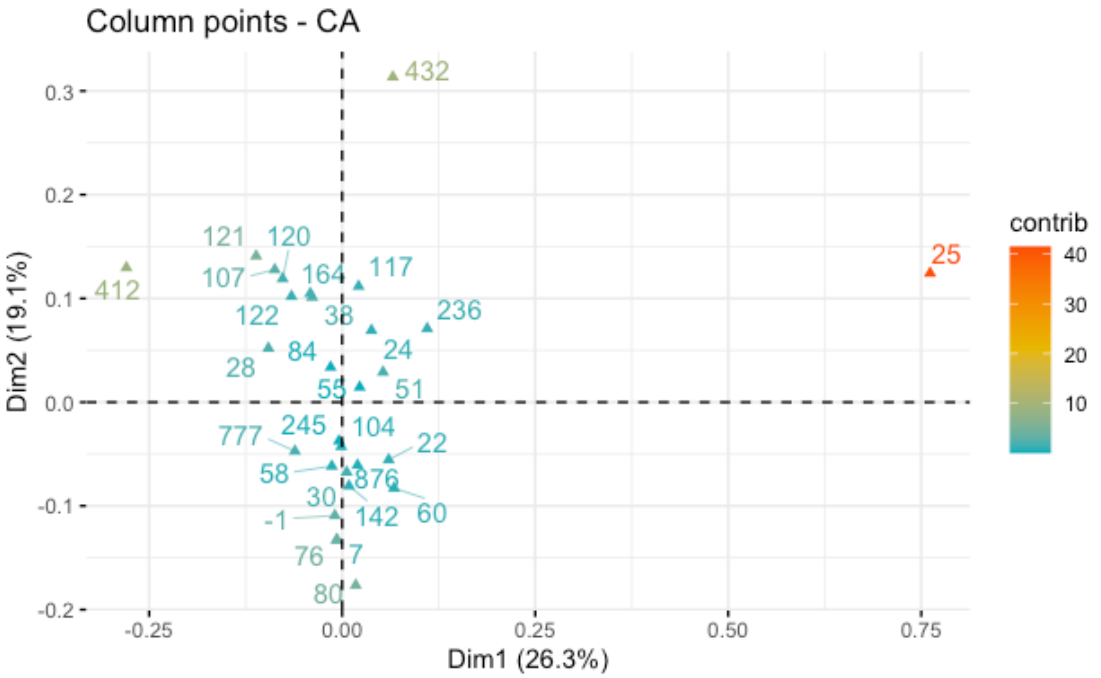
Como se aprecia en la gráfica, la varianza explicada por las dimensiones es buena, ya que tiene forma escalonada y entre las primeras dimensiones ya se recoge un gran porcentaje de variabilidad explicada. Para realizar el análisis nosotros escogeremos las cuatro primeras dimensiones, aunque también podríamos escoger 5. Ahora vamos a representar las gráficas de filas y columnas para las primeras cuatro dimensiones.



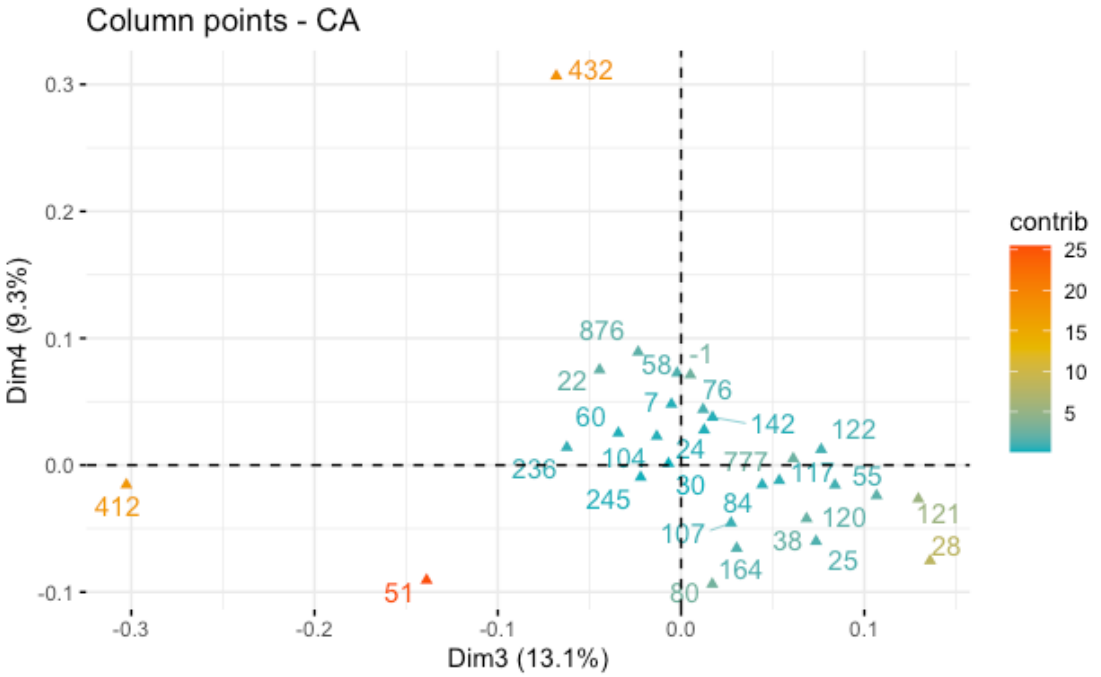
En esta gráfica, los personajes cercanos tienen patrones similares de bloqueo. Los personajes como 40 y 53, están bien representados en estas dimensiones, lo que indica que tienen perfiles de selección y bloqueo bien definidos en estas dimensiones.



Este gráfico muestra patrones adicionales de selección y bloqueo que no se observaron claramente en el primer gráfico. Personajes como 412 y 26 tienen perfiles únicos en estas dimensiones adicionales.



En las gráficas de columnas se puede apreciar una gran cantidad de valores cercanos al eje y unos pocos valores alejados del resto. Estos valores mencionados son los que más contribuyen a las dimensiones del gráfico.



Este gráfico es similar al de las dimensiones 1 y 2, y mantiene como valores aislados el 412 y el 432. En el apartado de conclusiones se explicarán estos resultados aplicados al contexto del videojuego para que se puedan entender mejor.

Conclusiones

De este análisis podemos concluir, que personajes diferentes con el rol de soporte, que tienen caracteristicas similares en el juego, también tienen un comportamiento similar a la hora de bloquear personajes del equipo rival. Un ejemplo de esto sería en el gráfico de filas los personajes 53 y 89. Estos personajes en el juego se llaman Leona y Blitzcrank. Estos personajes tienen en la partida la función de iniciar las peleas, debido a que sus habilidades son buenas en este ámbito y situan al equipo en una posicion de ventaja frente al equipo rival. Como estos personajes se parecen entre si, tambien tienen los mismos counters(personaje que lo contrarresta), y por ende, los personajes bloqueados parecidos. Un segundo ejemplo de esto serían los personajes 40 y 16, que equivalen en el juego a Soraka y Janna. Estos personajes tienen la función de ayudar al equipo proporcionandoles más vida y escudos, y esta es la razón por la cuál también van a bloquear personajes parecidos. Principalmente, personajes que cortan curaciones.

Por otro lado, gracias al gráfico de columnas hemos podido comprobar que personajes muy populares en el rol de soporte se mantienen aislados porque se bloquean independientemente del personaje escogido por el jugador. En ocasiones, cuando no sabes si tu personaje seleccionado tienen un counter específico se suele recurrir a personajes que están muy fuertes en el meta o personajes muy populares como es el caso de Morgana, que representa la variable 25 en la primera gráfica de columnas. También hay personajes que, jugados a un alto nivel decantan mucho el resultado de la partida. Estas variables son la número 412 y 432 y los personajes en partida se llaman Bardo y Tresh. Estos personajes son buenos dependiendo de las circumstancias, pero como estamos analizando partidas de alto nivel, es más probable que se den estas circumstancias, y es por esto que puede ser que aparezcan en ambas gráficas de columnas.

Por último, podemos deducir que las dimensiones 1 y 2 se centran en el meta del juego y las preferencias y estrategias generales de los jugadores. Mientras que la 3 y 4 se centran más en situaciones circumstanciales y en un contexto más específico. Todo esto debido a que en las primeras dimensiones aparecen personajes más competitivos en sus estadísticas en partida (variables 89 y 40 del gráfico de filas), y en el resto de dimensiones personajes que funcionan bien dependiendo del equipo rival y la partida en general (variable 51 gráfico de columnas y variable 26 gráfico de filas).

AFC Múltiple

Introducción

En esta segunda sección del proyecto vamos a realizar un AFC múltiple sobre una base de datos diferente a la original. En este caso hemos reducido considerablemente las partidas totales y hemos hecho una transformación de tal forma que cada fila representa un jugador. Las variables escogidas para este estudio son “summoner1” ,“summoner2”, “runa1”,“runa2”,“rol” y “win”. Las 4 primeras variables tienen que ver con elecciones de habilidades y mejoras en las caracteristicas del personaje, y las dos

últimas reflejan el tipo de funcion en el equipo y por ultimo una variable que marca si el jugador ha ganado o no. Nuestro objetivo en este caso es analizar que decisiones debemos tomar antes de empezar la partida para mejorar el porcentaje de victoria.

Conclusion final

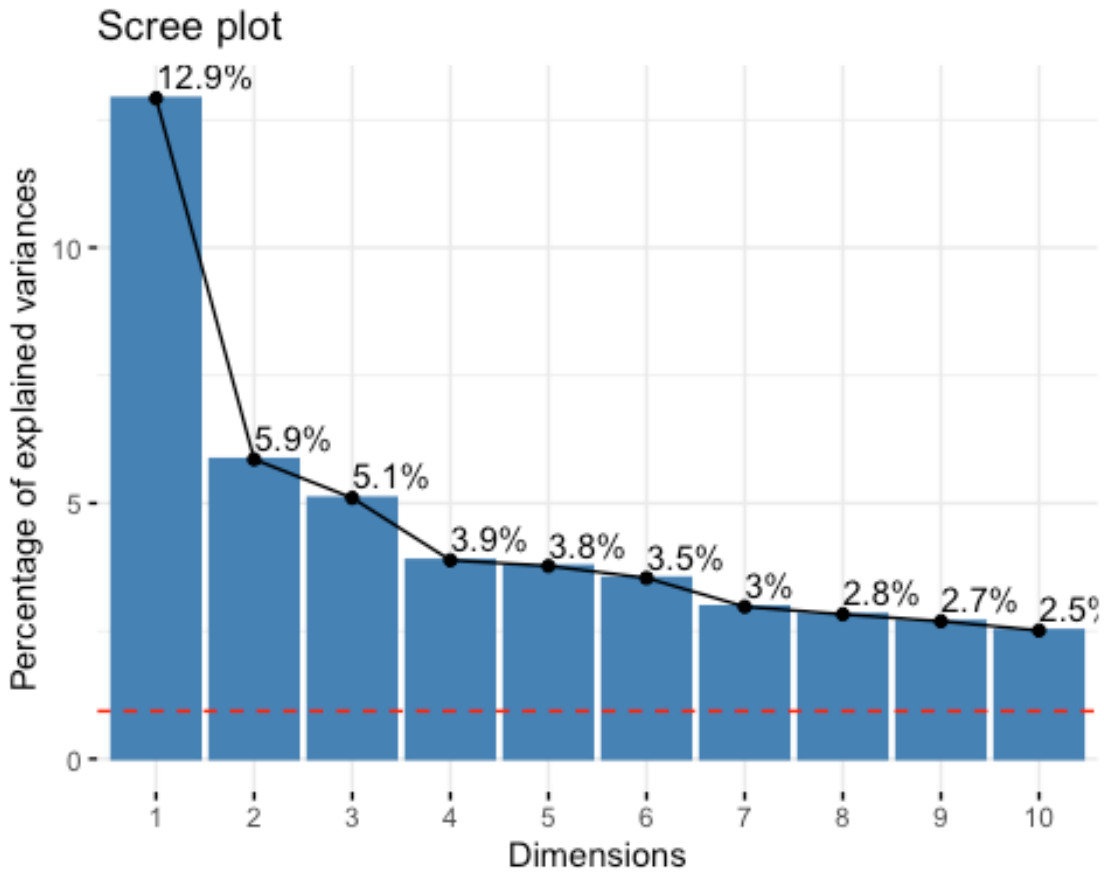
Las conclusiones del estudio éxito en League of Legends está fuertemente influenciado por el trabajo en equipo y la capacidad de acumular una ventaja tanto económica como de nivel. Los roles de support y jungla son críticos para la dinámica del equipo, con su éxito o fracaso dependiendo de que sean capaces de que a los otros jugadores les vaya bien en la partida. Es también importante saber banear los campeones que más te perjudican.

Anexos

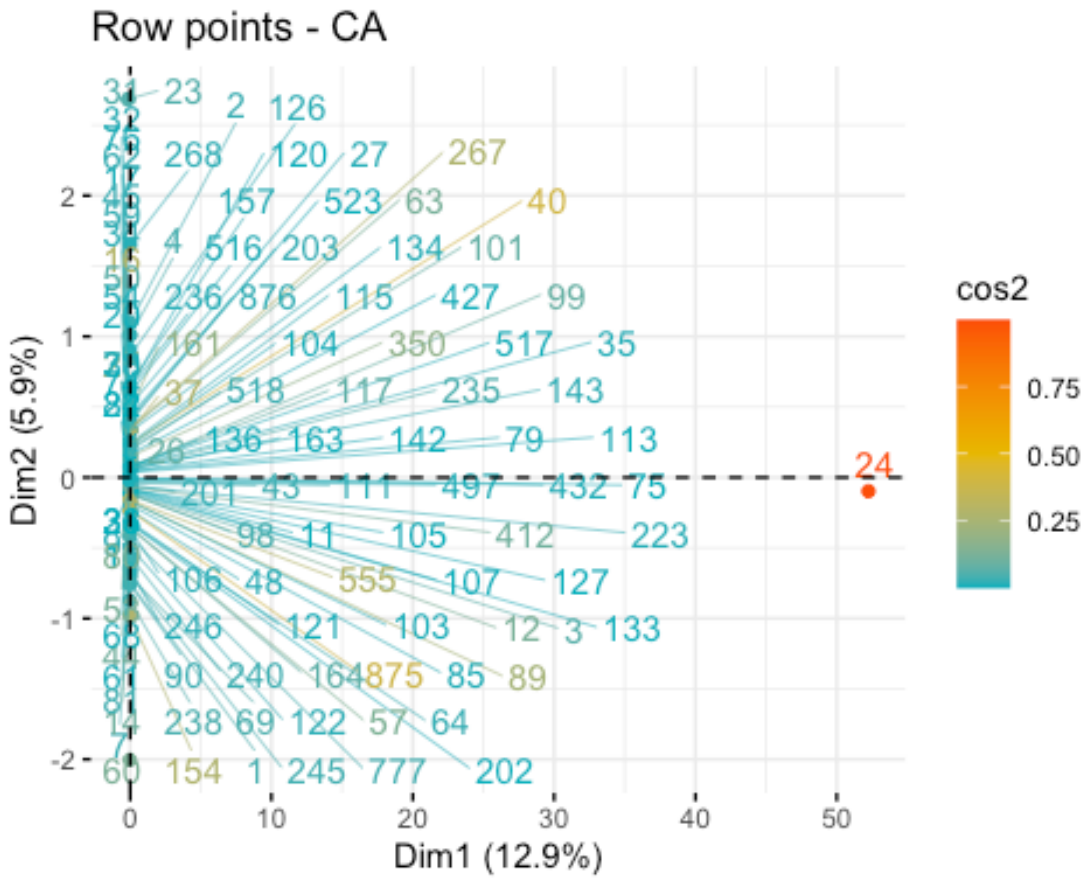
AFC

```
library(FactoMineR)
library(factoextra)
library(knitr)
datos <-
read.csv("~/Desktop/Universidad/Segundo_grado/proyecto_mdp/dataset_limpio
.csv",sep=";")
head(datos)
colnames(datos)

descToyo = data.frame("variable" = colnames(datos),
                      "tipo" = c("text",rep(c("numerical",
"categorical","categorical",
                                              rep("numerical",4),
rep("categorical",7),
                                              rep("numerical",11),
"categorical", "categorical",
rep("numerical",2),"categorical","categorical",
                                              rep("categorical",1),

rep("numerical",7)),10),"numerical"

,"categorical","numerical","categorical"), stringsAsFactors = FALSE)
rownames(descToyo) = descToyo$variable
descToyo
valores_faltantes_por_columna <- colSums(is.na(datos))
valores_faltantes_por_columna

tabla_contingencia <- table(datos$t1p1_champId, datos$t1p1_ban_champId)
tabla_contingencia
ncol(tabla_contingencia)
```

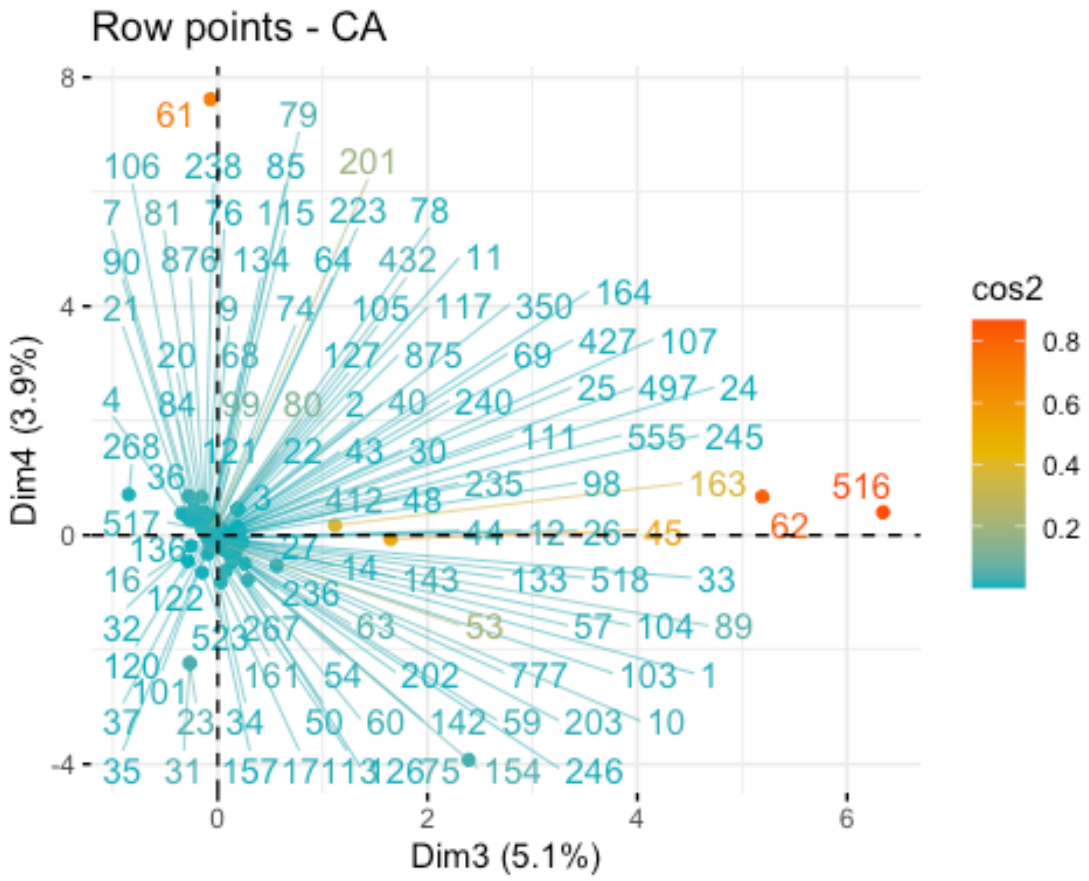
```r
children = tabla_contingencia[,1:151]
children
miF = (children/sum(children))*100; round(miF, 4)
chisq.test(children, simulate.p.value = TRUE)
margFilas = rowSums(miF);
margCols = colSums(miF);
condFilas = miF/margFilas;
condCols = t(t(miF)/margCols);


res.afc = CA(children, graph = FALSE)
eig.val <- get_eigenvalue(res.afc)
Vmedia = 100 * (1/nrow(eig.val))
fviz_eig(res.afc, addlabels = TRUE) +
  geom_hline(yintercept=Vmedia, linetype=2, color="red")
```



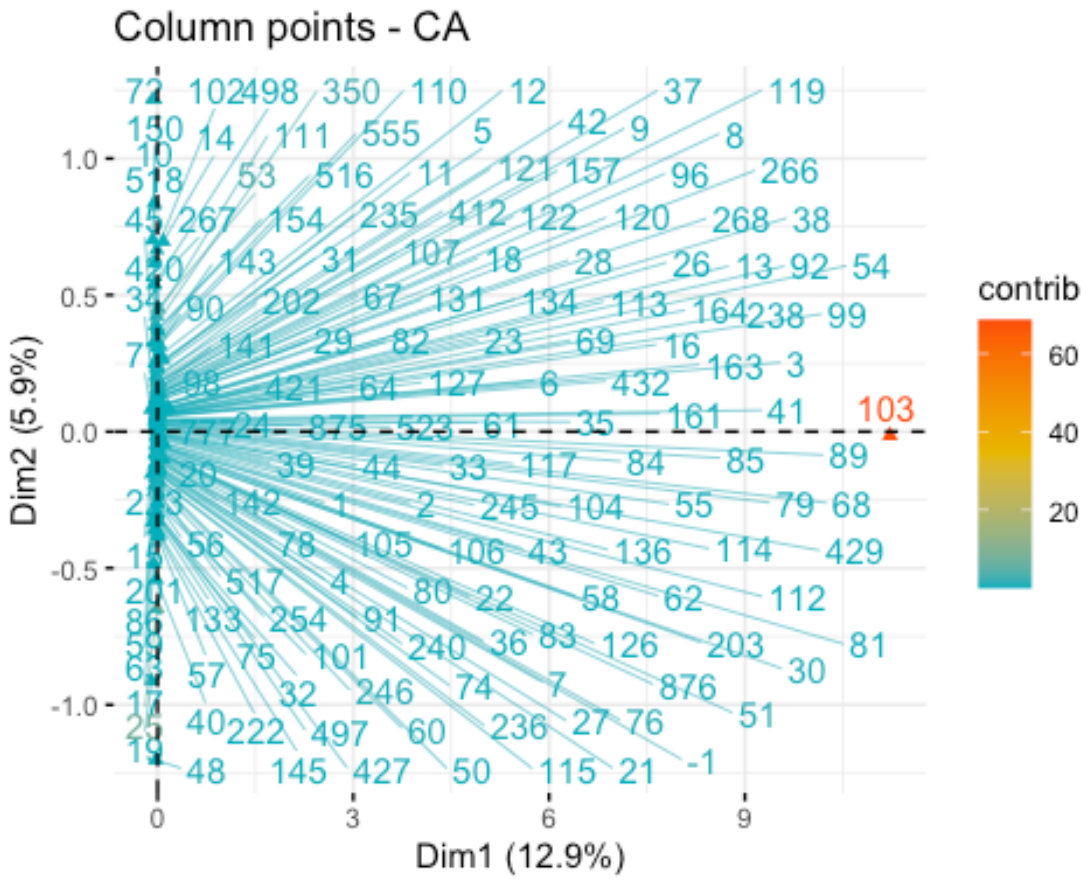

```r
kable(eig.val)
res.afc = CA(children, graph = FALSE, ncp = 4)
fviz_ca_row(res.afc, axes = c(1,2), repel = TRUE, col.row = "cos2",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```



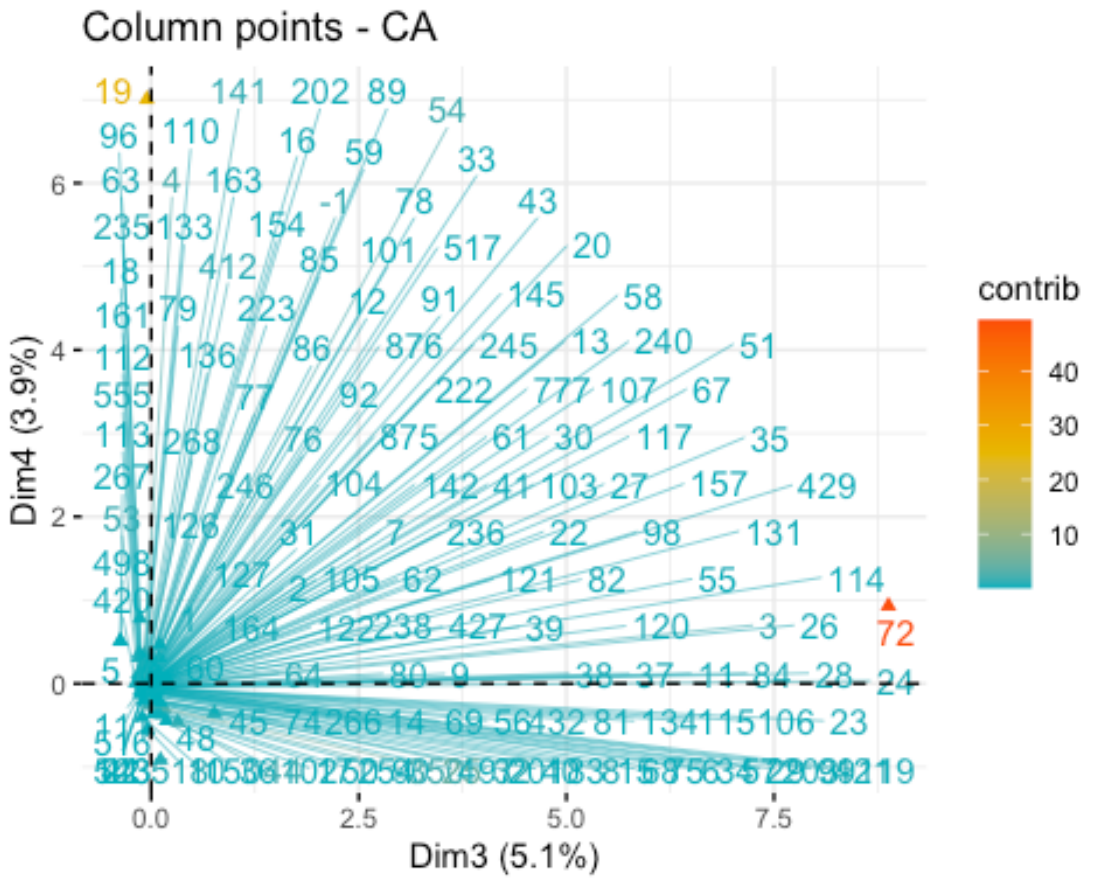

```
kable(res.afc$row$contrib)
fviz_ca_row(res.afc, axes = c(3,4), repel = TRUE, col.row = "cos2",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```




```
fviz_ca_col(res.afc, axes = c(1,2), repel = TRUE, col.col = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```

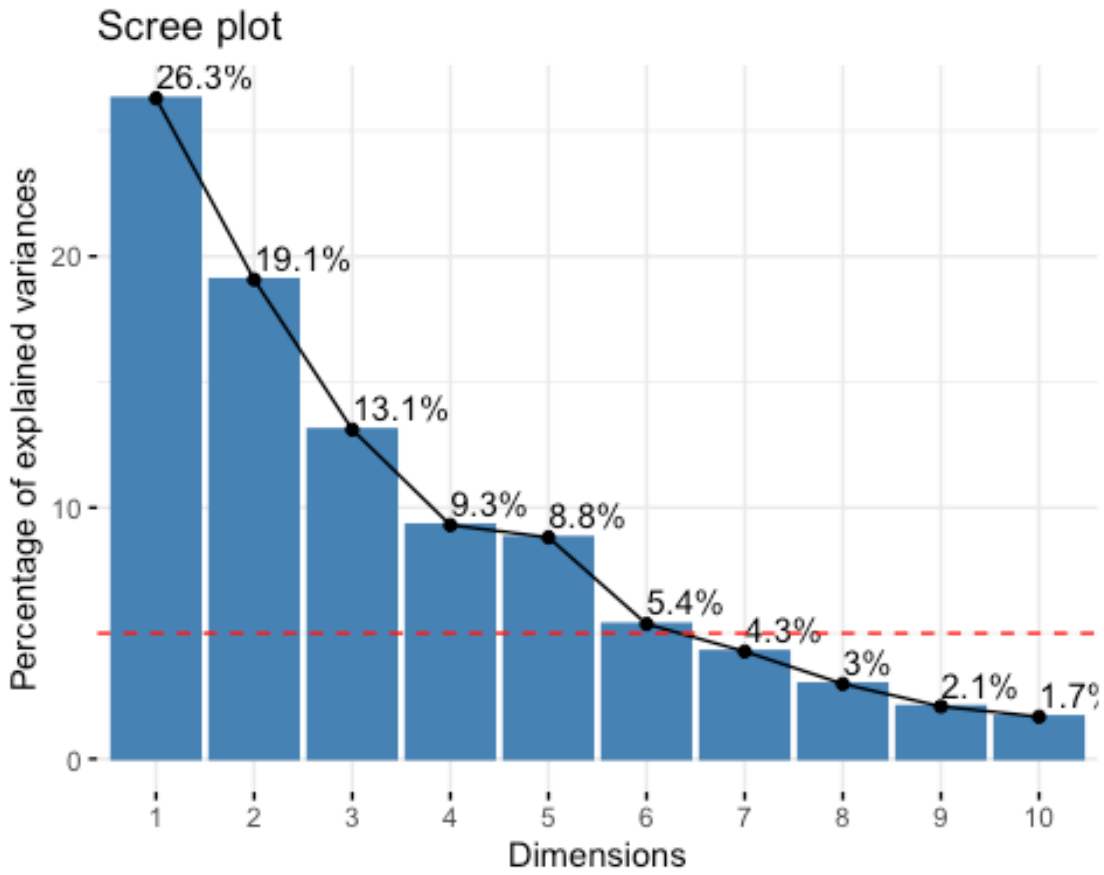



```
fviz_ca_col(res.afc, axes = c(3,4), repel = TRUE, col.col = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```

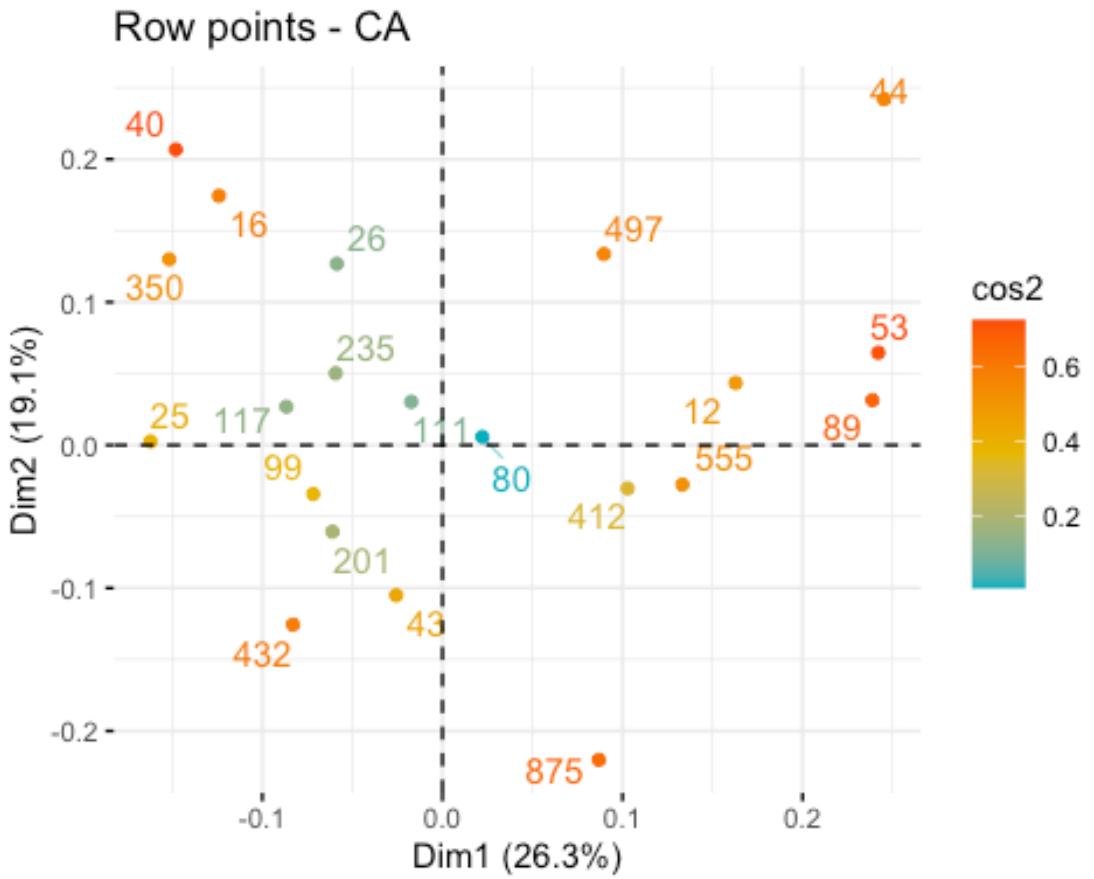



```
#Filtrado de datos, variables con mas de un 1 porciento de la informacion
total

frecuencia_filas <- rowSums(children)
porcentaje_frecuencia_filas <- (frecuencia_filas / sum(frecuencia_filas))
* 100
filas_filtradas <-
names(porcentaje_frecuencia_filas[porcentaje_frecuencia_filas >= 1])
children_filtrado <- children[filas_filtradas, ]
frecuencia_variables <- colSums(children_filtrado)
porcentaje_frecuencia_variables <- (frecuencia_variables /
sum(frecuencia_variables)) * 100
variables_filtradas <-
names(porcentaje_frecuencia_variables[porcentaje_frecuencia_variables >=
1])
children_filtrado <- children_filtrado[, variables_filtradas]



res.afc2 = CA(children_filtrado, graph = FALSE)
eig.val2 <- get_eigenvalue(res.afc2)
Vmedia2 = 100 * (1/nrow(eig.val2))
```

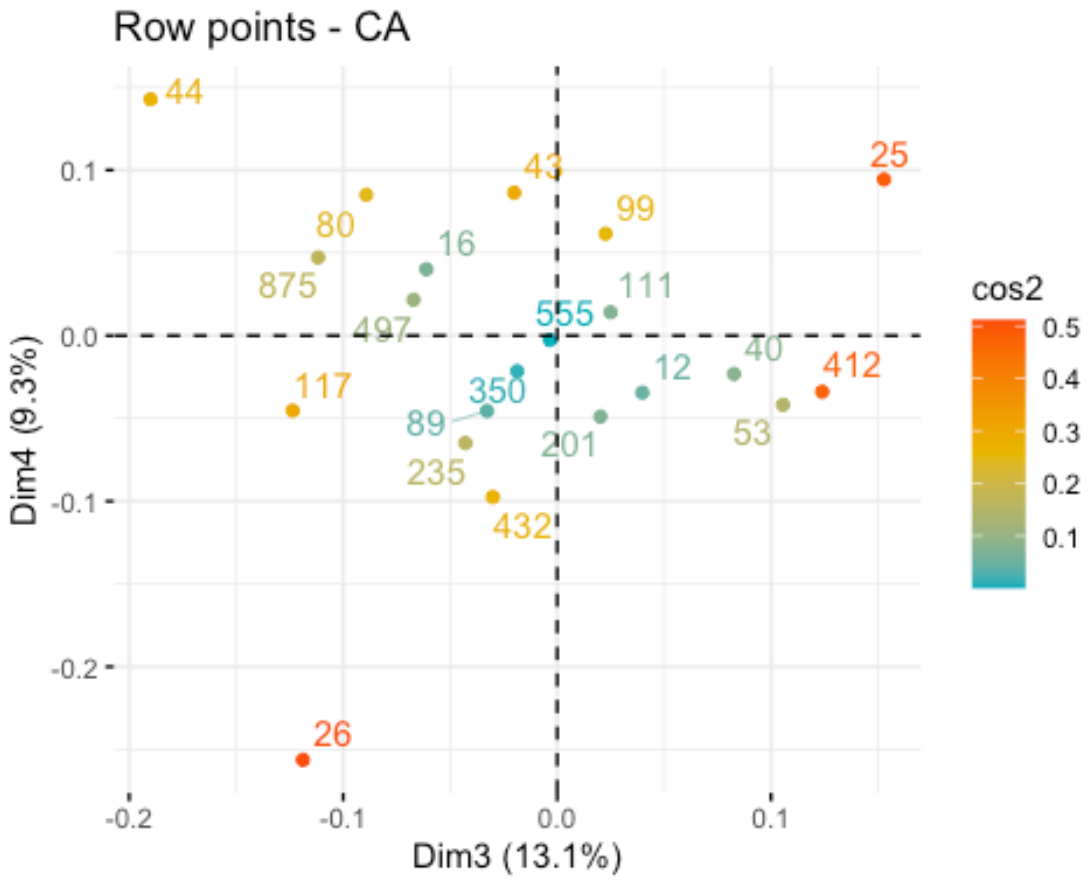
```
fviz_eig(res.afc2, addlabels = TRUE) +
  geom_hline(yintercept=Vmedia2, linetype=2, color="red")
```



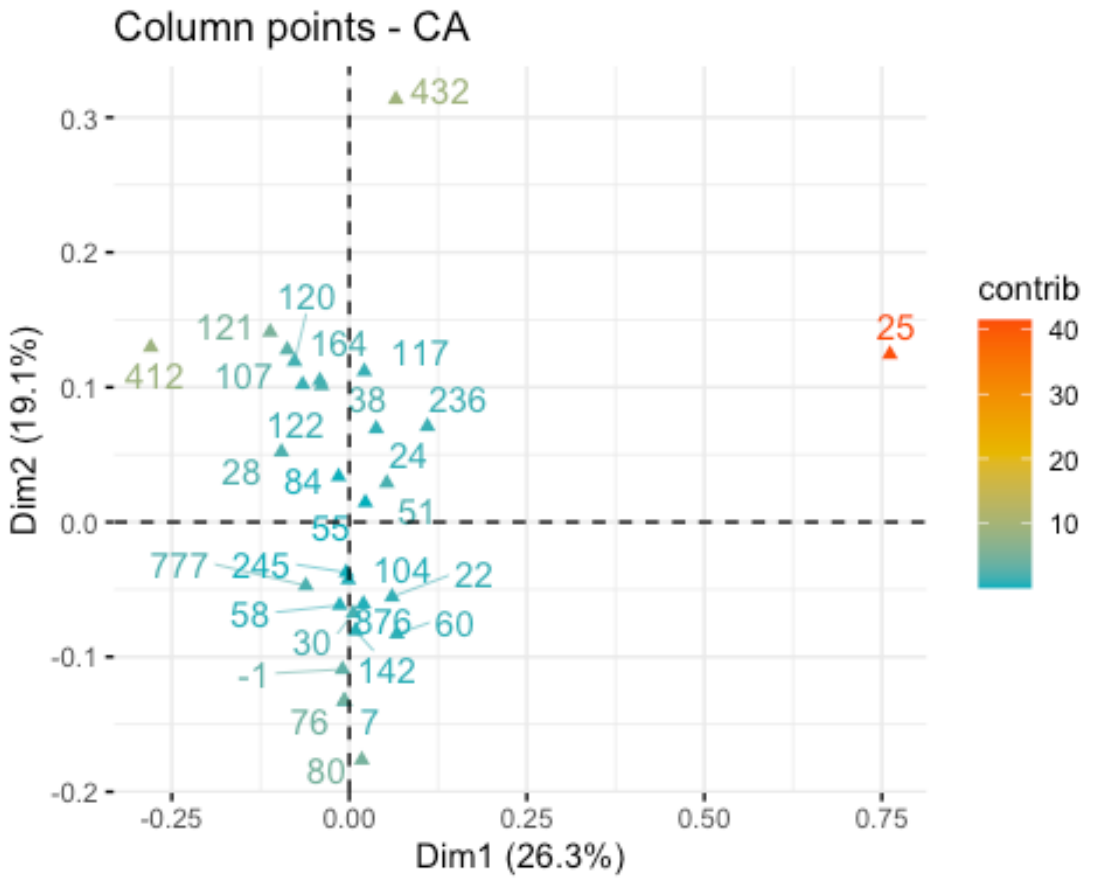

```
kable(eig.val2)
res.afc2 = CA(children_filtrado, graph = FALSE, ncp = 4)
fviz_ca_row(res.afc2, axes = c(1,2), repel = TRUE, col.row = "cos2",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```



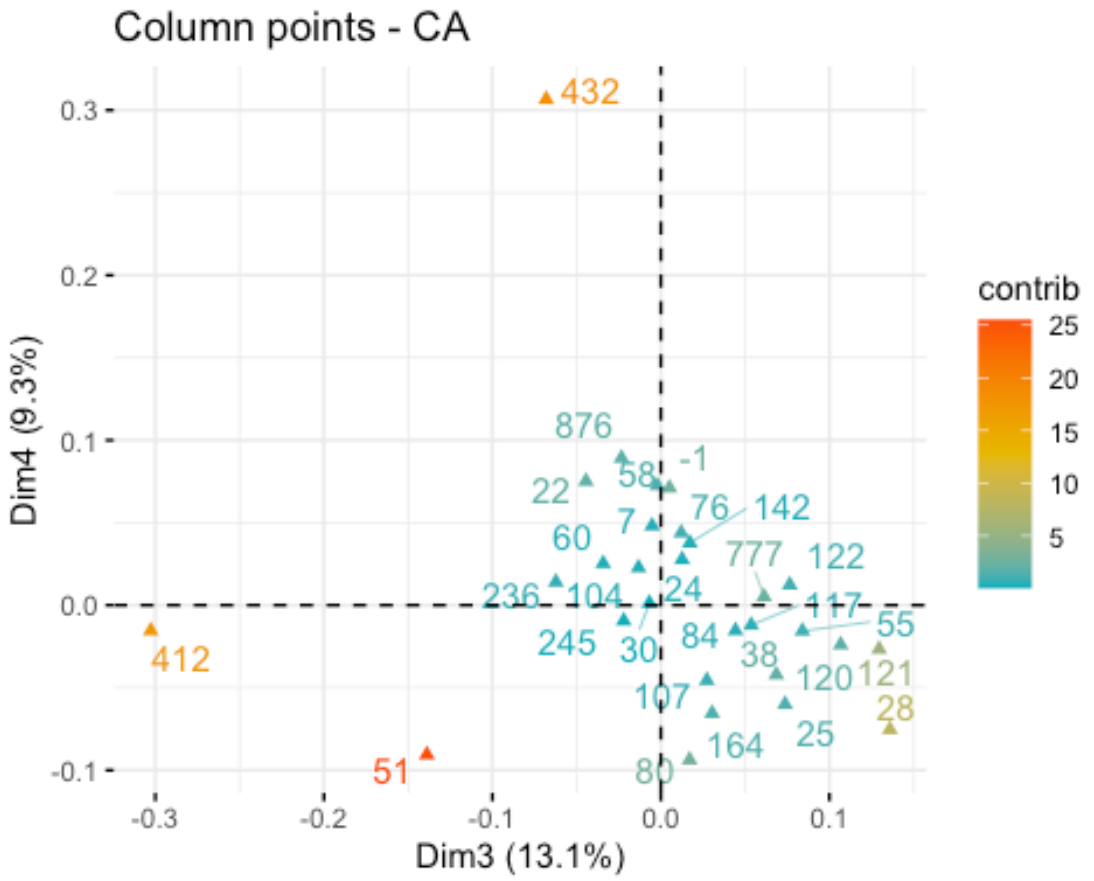

```
kable(res.afc2$row$contrib)
fviz_ca_row(res.afc2, axes = c(3,4), repel = TRUE, col.row = "cos2",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```



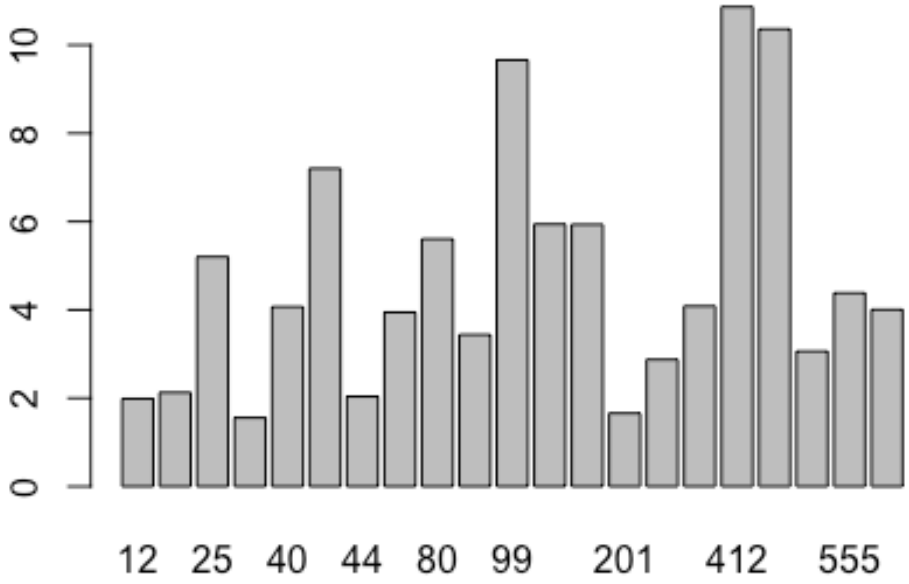

```
fviz_ca_col(res.afc2, axes = c(1,2), repel = TRUE, col.col = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```




```
fviz_ca_col(res.afc2, axes = c(3,4), repel = TRUE, col.col = "contrib",
            gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))
```




```
miF2 = (children_filtrado/sum(children_filtrado))*100; round(miF2, 4)
margFilas2 = rowSums(miF2);
margCols2 = colSums(miF2);
condFilas2 = miF2/margFilas2;
condCols2 = t(t(miF2)/margCols2);
barplot(margFilas2)
```




```r
chisq.test(children_filtrado, simulate.p.value = TRUE)

#AFC multiple

library(FactoMineR)
library(factoextra)
library(knitr)


#Afc multiple tabla shaco
#datos3=read.csv("shaco.csv")
#datos2=read.csv("tabla2.csv")
#head(datos2)
#descToyo2 = data.frame("variable" = colnames(datos3),
 #                      "tipo" = c(rep("categorical",15)))
#rownames(descToyo2) = descToyo2$variable
#descToyo2
#datos_2 <- lapply(datos3, as.factor)
#datos_categoricos3=
datos3[,c("summoner1","summoner2","runa1","runa2","rol","win")]
#datos_categoricos_3=lapply(datos_categoricos3, as.factor)
```

```
#valores_faltantes_por_columna <- colSums(is.na(datos_categoricos3))
#valores_faltantes_por_columna

# Realizar MCA
#res.mca3 <- MCA(datos_categoricos3, graph = FALSE, ncp=4)
#summary(res.mca3)

# Obtener eigenvalues
#eig.val3 <- get_eigenvalue(res.mca3)
#eig.val3

# Scree plot de eigenvalues
#fviz_eig(res.mca3, addlabels = TRUE, ylim = c(0, 50)) +
 # geom_hline(yintercept = 100 / nrow(eig.val3), linetype = 2, color =
"red")

# Visualización de las filas

#fviz_mca_ind(res.mca3,
 #            repel = TRUE,
  #           col.ind = "cos2",
   #          gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
    #         addEllipses = FALSE)

# Visualización de las columnas (variables)

#fviz_mca_var(res.mca3,
 #            repel = TRUE,
  #           col.var = "contrib",
   #          gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))

# Contribuciones de los individuos
#kable(res.mca3$ind$contrib, caption = "Contribuciones de los
Individuos")

# Contribuciones de las variables
#kable(res.mca3$var$contrib, caption = "Contribuciones de las Variables")

# Coordenadas de los individuos
#kable(res.mca3$ind$coord, caption = "Coordenadas de los Individuos")

# Coordenadas de las variables
#kable(res.mca3$var$coord, caption = "Coordenadas de las Variables")


# Contribuciones de los individuos
#ind_contrib <- res.mca3$ind$contrib
#print(ind_contrib)
```

```
# Contribuciones de las variables
#var_contrib <- res.mca3$var$contrib
#print(var_contrib)

# Visualización biplot
#fviz_mca_biplot(res.mca3, repel = TRUE,
 #               col.var = "contrib",
  #              col.ind = "cos2",
   #             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"))

#Variable Win
#var_contrib <- res.mca3$var$contrib

# Filtrar contribuciones relacionadas con "win_1"
#win_1_contrib <- var_contrib[grepl("win_1", rownames(var_contrib)), ]

# Ordenar las contribuciones para ver las variables más relacionadas
#sorted_win_1_contrib <- win_1_contrib[order(-win_1_contrib[, 1]), ]
#print(sorted_win_1_contrib)
```

PLS

Y es la variable respuesta, que en este caso es el resultado del juego (win). X es la matriz de variables predictoras, que incluye estadísticas como asistencias, nivel, muertes, oro, asesinatos, etc.

Y se convierte en un vector numérico. X se asegura que sea un data.frame.

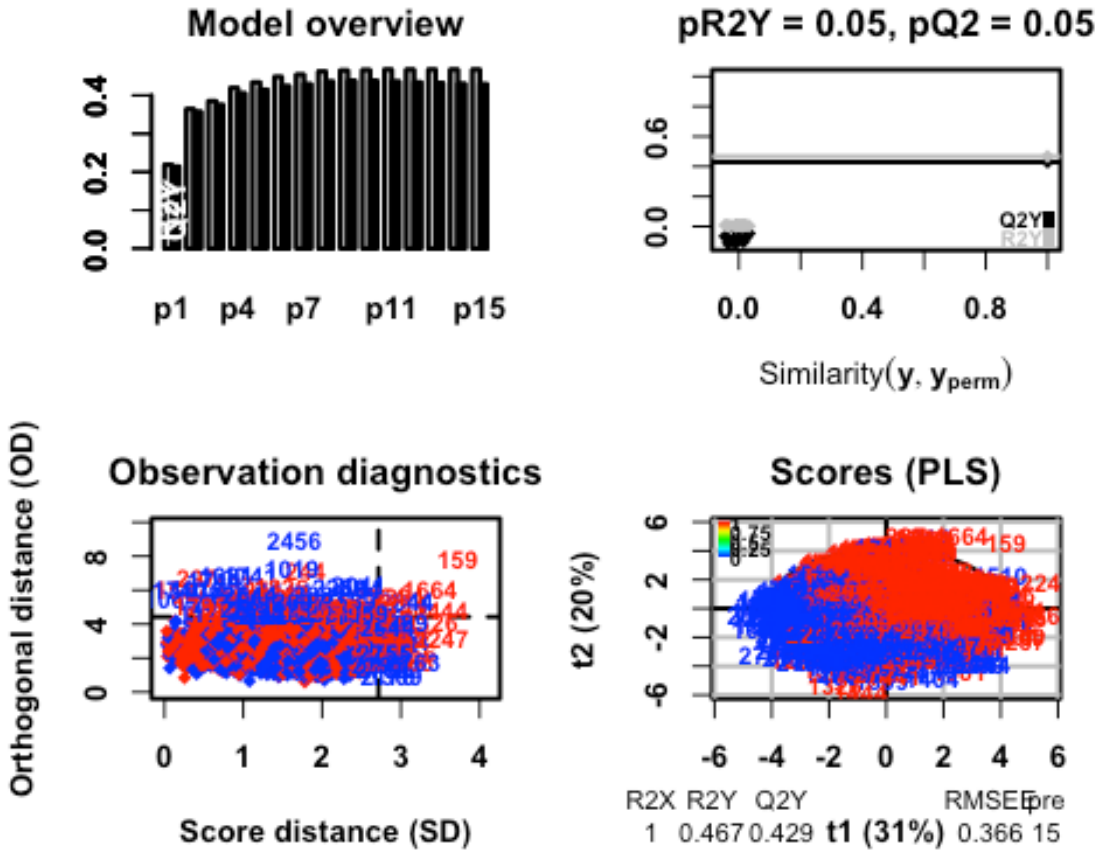
Se escalan los datos.

El gráfico generado es un “validation plot” que muestra cómo el MSEP cambia con el número de componentes en el modelo PLS.

Al principio, el MSEP desciende rápidamente a medida que se añaden más componentes, lo que sugiere que los primeros componentes capturan la mayor parte de la variación explicativa en Y. Concretamente los 2 primeros componentes.

Se realiza la división de los datos en conjuntos de entrenamiento y prueba, y luego se ajusta el modelo PLS utilizando el conjunto de entrenamiento.

Estos resultados sugieren que el modelo PLS tiene una capacidad razonable para explicar y predecir la variabilidad en la variable de respuesta.



##

PCA

```
dataset_limpio=read_xlsx("~/Desktop/Universidad/Segundo_grado/proyecto_md
p/dataset_limpio2.xlsx")
head(dataset_limpio)
columnas=colnames(dataset_limpio)
dataset_limpio= subset(dataset_limpio, select = -t1p2_accountId)
dfChamps=data.frame(dataset_limpio)

colnames(dataset_limpio)

columnas_seleccionadas = str_subset(columnas, "champId$")
columnas_seleccionadas
indices_pares <- seq_along(columnas_seleccionadas) %% 2 == 0
columnas_pares <- columnas_seleccionadas[indices_pares]
columnas_pares
champsplayed=dfChamps[,columnas_pares]
champsplayed
cadena_concatenada = paste(champsplayed, collapse = "")
champsplayed
champsplayedcol=colnames(champsplayed)
columnas_pares
```

```r
shaco <- dataset_limpio
champsplayedcol <- c(champsplayedcol, "t1_win")

descChamp2 = data.frame("variable" = colnames(dfChamps),
                        "tipo" =
c("text",rep(c("numerical",rep("categorical",2),"numerical","categorical"
,rep("numerical",2),rep("categorical",7),"numerical",rep("categorical",4)
,rep("numerical",2),"categorical",rep("numerical",3),rep("categorical",7)
,rep("numerical",4),rep("categorical",2),"numerical"),10),"numerical","ca
tegorical","categorical","categorical"), stringsAsFactors = FALSE)

indices_numericos <- which(descChamp2$tipo == "numerical")
indices_numericos= c(indices_numericos,
4,43,82,121,160,199,238,277,316,355,395)
indices_numericos=sort(indices_numericos)

shaco3=shaco[,c(indices_numericos)]
shaco3= shaco3[1:300, ]
descChampSH = data.frame("variable" = colnames(shaco3),
                         "tipo" = c(rep(c("numerical", "categorical",

rep("numerical",14)),10),"numerical","categorical"), stringsAsFactors =
FALSE)

rownames(descChampSH) = descChampSH$variable
categ = descChampSH$variable[descChampSH$tipo == "categorical"]
for (cc in categ) {
  dfChamps[,cc] = factor(dfChamps[,cc])
  }

assists=
c(shaco3$t1p1_assists,shaco3$t1p2_assists,shaco3$t1p3_assists,shaco3$t1p4
_assists,shaco3$t1p5_assists,shaco3$t2p1_assists,shaco3$t2p2_assists,shac
o3$t2p3_assists,shaco3$t2p4_assists,shaco3$t2p5_assists)
champs=
c(shaco3$t1p1_champId,shaco3$t1p2_champId,shaco3$t1p3_champId,shaco3$t1p4
_champId,shaco3$t1p5_champId,shaco3$t2p1_champId,shaco3$t2p2_champId,shac
o3$t2p3_champId,shaco3$t2p4_champId,shaco3$t2p5_champId)
lvl=
c(shaco3$t1p1_champLevel,shaco3$t1p2_champLevel,shaco3$t1p3_champLevel,sh
aco3$t1p4_champLevel,shaco3$t1p5_champLevel,shaco3$t2p1_champLevel,shaco3
$t2p2_champLevel,shaco3$t2p3_champLevel,shaco3$t2p4_champLevel,shaco3$t2p
5_champLevel)
deaths=
c(shaco3$t1p1_deaths,shaco3$t1p2_deaths,shaco3$t1p3_deaths,shaco3$t1p4_de
aths,shaco3$t1p5_deaths,shaco3$t2p1_deaths,shaco3$t2p2_deaths,shaco3$t2p3
_deaths,shaco3$t2p4_deaths,shaco3$t2p5_deaths)
gold=
c(shaco3$t1p1_goldEarned,shaco3$t1p2_goldEarned,shaco3$t1p3_goldEarned,sh
aco3$t1p4_goldEarned,shaco3$t1p5_goldEarned,shaco3$t2p1_goldEarned,shaco3
$t2p2_goldEarned,shaco3$t2p3_goldEarned,shaco3$t2p4_goldEarned,shaco3$t2p
```

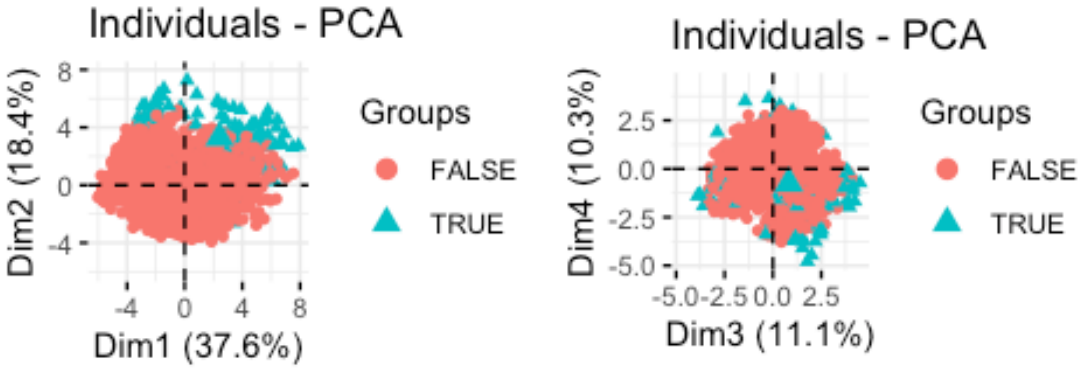
```
5_goldEarned)
kills=
c(shaco3$t1p1_kills,shaco3$t1p2_kills,shaco3$t1p3_kills,shaco3$t1p4_kills
,shaco3$t1p5_kills,shaco3$t2p1_kills,shaco3$t2p2_kills,shaco3$t2p3_kills,
shaco3$t2p4_kills,shaco3$t2p5_kills)
minions=
c(shaco3$t1p1_totalMinionsKilled,shaco3$t1p2_totalMinionsKilled,shaco3$t1
p3_totalMinionsKilled,shaco3$t1p4_totalMinionsKilled,shaco3$t1p5_totalMin
ionsKilled,shaco3$t2p1_totalMinionsKilled,shaco3$t2p2_totalMinionsKilled,
shaco3$t2p3_totalMinionsKilled,shaco3$t2p4_totalMinionsKilled,shaco3$t2p5
_totalMinionsKilled)
gold10=
c(shaco3$t1p1_min10_gold,shaco3$t1p2_min10_gold,shaco3$t1p3_min10_gold,sh
aco3$t1p4_min10_gold,shaco3$t1p5_min10_gold,shaco3$t2p1_min10_gold,shaco3
$t2p2_min10_gold,shaco3$t2p3_min10_gold,shaco3$t2p4_min10_gold,shaco3$t2p
5_min10_gold)
xp10=
c(shaco3$t1p1_min10_xp,shaco3$t1p2_min10_xp,shaco3$t1p3_min10_xp,shaco3$t
1p4_min10_xp,shaco3$t1p5_min10_xp,shaco3$t2p1_min10_xp,shaco3$t2p2_min10_
xp,shaco3$t2p3_min10_xp,shaco3$t2p4_min10_xp,shaco3$t2p5_min10_xp)
gold15=
c(shaco3$t1p1_min15_gold,shaco3$t1p2_min15_gold,shaco3$t1p3_min15_gold,sh
aco3$t1p4_min15_gold,shaco3$t1p5_min15_gold,shaco3$t2p1_min15_gold,shaco3
$t2p2_min15_gold,shaco3$t2p3_min15_gold,shaco3$t2p4_min15_gold,shaco3$t2p
5_min15_gold)
xp15=
c(shaco3$t1p1_min15_xp,shaco3$t1p2_min15_xp,shaco3$t1p3_min15_xp,shaco3$t
1p4_min15_xp,shaco3$t1p5_min15_xp,shaco3$t2p1_min15_xp,shaco3$t2p2_min15_
xp,shaco3$t2p3_min15_xp,shaco3$t2p4_min15_xp,shaco3$t2p5_min15_xp)
jgl_minions=
c(shaco3$t1p1_neutralMinionsKilled,shaco3$t1p2_neutralMinionsKilled,shaco
3$t1p3_neutralMinionsKilled,shaco3$t1p4_neutralMinionsKilled,shaco3$t1p5_
neutralMinionsKilled,shaco3$t2p1_neutralMinionsKilled,shaco3$t2p2_neutral
MinionsKilled,shaco3$t2p3_neutralMinionsKilled,shaco3$t2p4_neutralMinions
Killed,shaco3$t2p5_neutralMinionsKilled)
duration=
c(shaco3$gameDuration,shaco3$gameDuration,shaco3$gameDuration,shaco3$game
Duration,shaco3$gameDuration,shaco3$gameDuration,shaco3$gameDuration,shac
o3$gameDuration,shaco3$gameDuration,shaco3$gameDuration)
totalHeal =
c(shaco3$t1p1_totalHeal,shaco3$t1p2_totalHeal,shaco3$t1p3_totalHeal,shaco
3$t1p4_totalHeal,shaco3$t1p5_totalHeal,shaco3$t2p1_totalHeal,shaco3$t2p2_
totalHeal,shaco3$t2p3_totalHeal,shaco3$t2p4_totalHeal,shaco3$t2p5_totalHe
al)
totalDamageTaken =
c(shaco3$t1p1_totalDamageTaken,shaco3$t1p2_totalDamageTaken,shaco3$t1p3_t
otalDamageTaken,shaco3$t1p4_totalDamageTaken,shaco3$t1p5_totalDamageTaken
,shaco3$t2p1_totalDamageTaken,shaco3$t2p2_totalDamageTaken,shaco3$t2p3_to
talDamageTaken,shaco3$t2p4_totalDamageTaken,shaco3$t2p5_totalDamageTaken)
vision =
```

```
c(shaco3$t1p1_visionScore,shaco3$t1p2_visionScore,shaco3$t1p3_visionScore
,shaco3$t1p4_visionScore,shaco3$t1p5_visionScore,shaco3$t2p1_visionScore,
shaco3$t2p2_visionScore,shaco3$t2p3_visionScore,shaco3$t2p4_visionScore,s
haco3$t2p5_visionScore)
totalDamageDealtToChampions=
c(shaco3$t1p1_totalDamageDealtToChampions,shaco3$t1p2_totalDamageDealtToC
hampions,shaco3$t1p3_totalDamageDealtToChampions,shaco3$t1p4_totalDamageD
ealtToChampions,shaco3$t1p5_totalDamageDealtToChampions,shaco3$t2p1_total
DamageDealtToChampions,shaco3$t2p2_totalDamageDealtToChampions,shaco3$t2p
3_totalDamageDealtToChampions,shaco3$t2p4_totalDamageDealtToChampions,sha
co3$t2p5_totalDamageDealtToChampions)
win1=
c(shaco3$t1_win,shaco3$t1_win,shaco3$t1_win,shaco3$t1_win,shaco3$t1_win)
win2=
c(shaco3$t1_win,shaco3$t1_win,shaco3$t1_win,shaco3$t1_win,shaco3$t1_win)
win2= ifelse(win2== 0, 1, 0)
rol1 =
c(rep("support",300),rep("ADC",300),rep("middle",300),rep("jungle",300),r
ep("top",300))
rol= c(rol1,rol1)
win=c(win1,win2)

tabla <- data.frame(
 champ =champs,
  assists = assists,
  lvl = lvl,
  deaths = deaths,
  gold = gold,
  kills = kills,
  gold10 = gold10,
  xp10 = xp10,
  gold15 = gold15,
  xp15 = xp15,
  minions = minions,
  jungla = jgl_minions,
  tDMGd = totalDamageDealtToChampions,
  tDMGt = totalDamageTaken,
  heal = totalHeal,
  vision = vision,
  duration = duration,
  rol = rol,
  win = win
)

cat_tabla= tabla[,names(tabla)%in% c("rol","champ","win")]
num_tabla= tabla[,!names(tabla)%in% c("rol","champ","win")]
tabla_normalize= as.data.frame(scale(num_tabla, center= TRUE, scale=
TRUE))

tabla_normalize$win=cat_tabla$win
tabla_normalize$rol=cat_tabla$rol
```

```
tabla_normalize$champ=cat_tabla$champ
summary(tabla_normalize)
```

```
##     assists             lvl               deaths             gold
##  Min.   :-1.5849   Min.   :-2.83738   Min.   :-1.9340   Min.   :-
2.21282
##  1st Qu.:-0.8007   1st Qu.:-0.52002   1st Qu.:-0.8397   1st Qu.:-
0.75903
##  Median :-0.2126   Median :-0.05654   Median :-0.1102   Median :-
0.08377
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   :
0.00000
##  3rd Qu.: 0.5715   3rd Qu.: 0.87040   3rd Qu.: 0.6194   3rd Qu.:
0.67489
##  Max.   : 6.0604   Max.   : 1.79735   Max.   : 4.2672   Max.   :
4.54239
##      kills              gold10              xp10              gold15
##  Min.   :-1.34417   Min.   :-2.24968   Min.   :-2.51744   Min.   :-
2.36648
##  1st Qu.:-0.83572   1st Qu.:-0.65957   1st Qu.:-0.80992   1st Qu.:-
0.66175
##  Median :-0.07305   Median :-0.02353   Median :-0.01556   Median :
0.02285
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   :
0.00000
##  3rd Qu.: 0.43540   3rd Qu.: 0.65153   3rd Qu.: 0.82452   3rd Qu.:
0.65491
##  Max.   : 5.51990   Max.   : 3.49403   Max.   : 2.90913   Max.   :
3.93128
##       xp15             minions            jungla             tDMGd
##  Min.   :-2.42599   Min.   :-1.4732   Min.   :-0.66469   Min.   :-
1.6722
##  1st Qu.:-0.76007   1st Qu.:-1.0868   1st Qu.:-0.64515   1st Qu.:-
0.7645
##  Median : 0.08818   Median : 0.2537   Median :-0.46933   Median :-
0.1822
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.00000   Mean   :
0.0000
##  3rd Qu.: 0.76259   3rd Qu.: 0.8215   3rd Qu.:-0.03953   3rd Qu.:
0.5515
##  Max.   : 2.69226   Max.   : 3.8069   Max.   : 4.53196   Max.   :
5.7481
##      tDMGt              heal              vision           duration
##  Min.   :-2.0185   Min.   :-1.0784   Min.   :-1.4258   Min.   :-
1.54229
##  1st Qu.:-0.7371   1st Qu.:-0.7381   1st Qu.:-0.6796   1st Qu.:-
0.82113
##  Median :-0.1323   Median :-0.3037   Median :-0.2816   Median :-
0.05484
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   :
```

```
0.00000
##  3rd Qu.: 0.5829   3rd Qu.: 0.4771   3rd Qu.: 0.3651   3rd Qu.:
0.69139
##  Max.   : 4.5692   Max.   : 7.6403   Max.   : 5.4891   Max.   :
3.31449
##       win          rol                champ
##  Min.   :0.0   Length:3000        Min.   :  1.0
##  1st Qu.:0.0   Class :character   1st Qu.: 51.0
##  Median :0.5   Mode  :character   Median : 99.0
##  Mean   :0.5                      Mean   :172.7
##  3rd Qu.:1.0                      3rd Qu.:222.0
##  Max.   :1.0                      Max.   :876.0

tabla= tabla_normalize
tabla$win <- as.numeric(tabla$win )

descGame = data.frame("variable" = colnames(tabla_normalize),
                      "tipo" = c(

rep("numerical",17),"categorical","categorical"), stringsAsFactors =
FALSE)



rownames(descGame) = descGame$variable
categ = descGame$variable[descGame$tipo == "categorical"]
for (cc in categ) {
  tabla[,cc] = factor(tabla[,cc])
  }
```



En base a estos gráficos de puntuaciones, podemos ver varias relaciones claramente diferenciadas:

las variables que más contribuyen a la primera componente son gold y lvl , las cuales además están fuertemente correlacionadas. Esta relación se da en la mayoría de partidas reales, ya que a más nivel del jugador durante la partida, más oro habrá ganado, mientras que duration, vision y assists son las que más contribuyen a la segunda componente y también están positivamente relacionadas

Las variables win, assists, heal, kills, jungla y vision están fuertemente correlacionadas, sugiriendo que estos factores pueden estar relacionados con el rendimiento general del equipo.

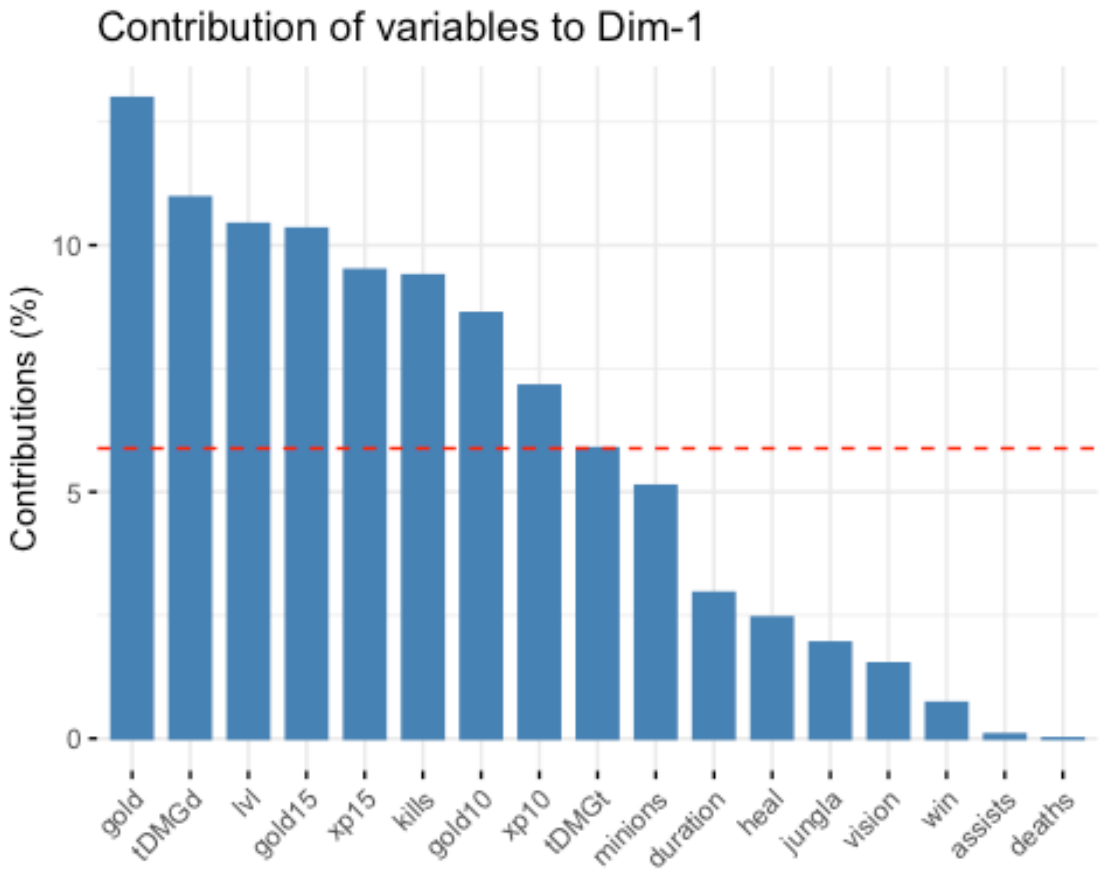
las variables que más contribuyen a la tercera componente son ,jungla y mninions, las cuales están negativamente correlacionadas, lo que sugiere que los minions de linea (minions) no suelen ser asesinados si se asesinan muchos minions de jungla (jungla), y viceversa. Esto parece correcto, ya que el rol de jungla en el lol se suele ocupar de asesinar todos los minions de jungla, mientras que el resto de roles suelen asessinar únicamente minions de línea.

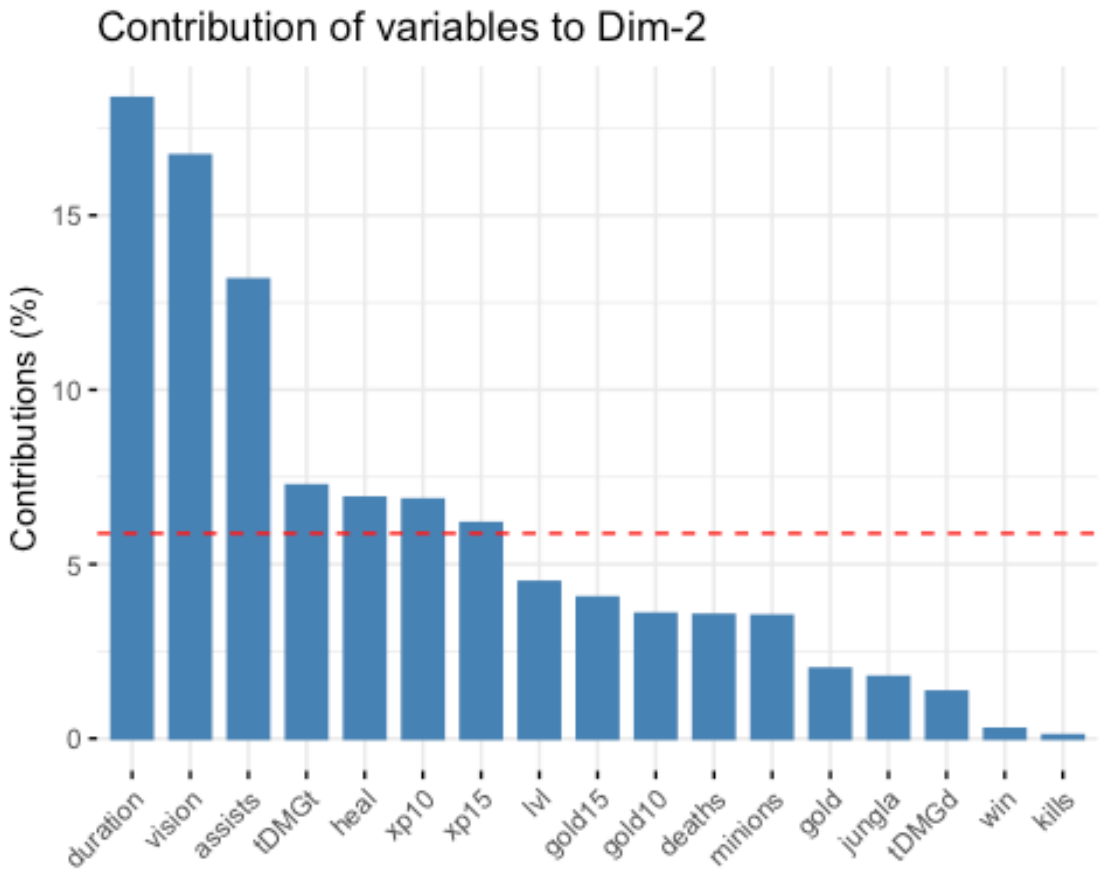
la variable deaths (muertes) está negativamente relacionada con el oro, xp y win, lo cual es bastante intuitivo, y está positivamente relacionado con el total de daño que ha recibido el jugador, la duración de la partida y el nivel (esta última es debido a que

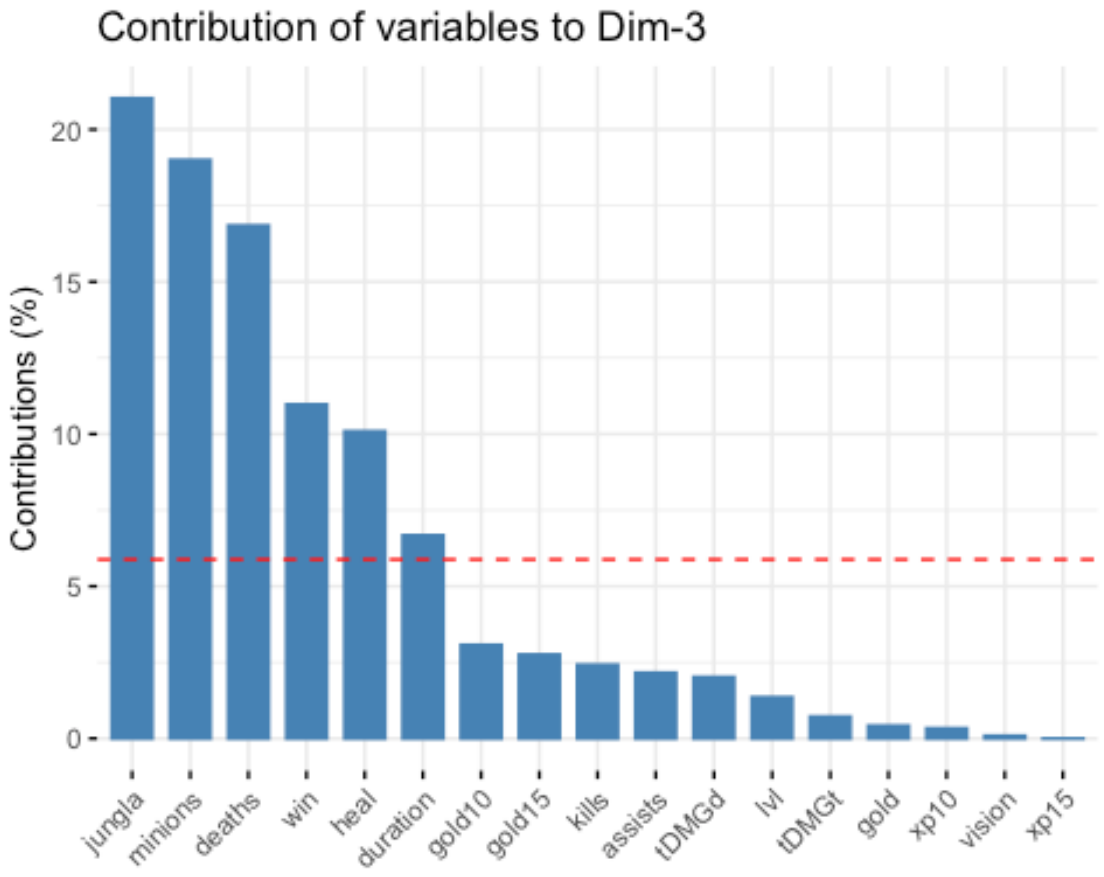
cuanto más dura una partida, más muertes hay y por tanto, el nivel con el que acabas la partida es mayor)

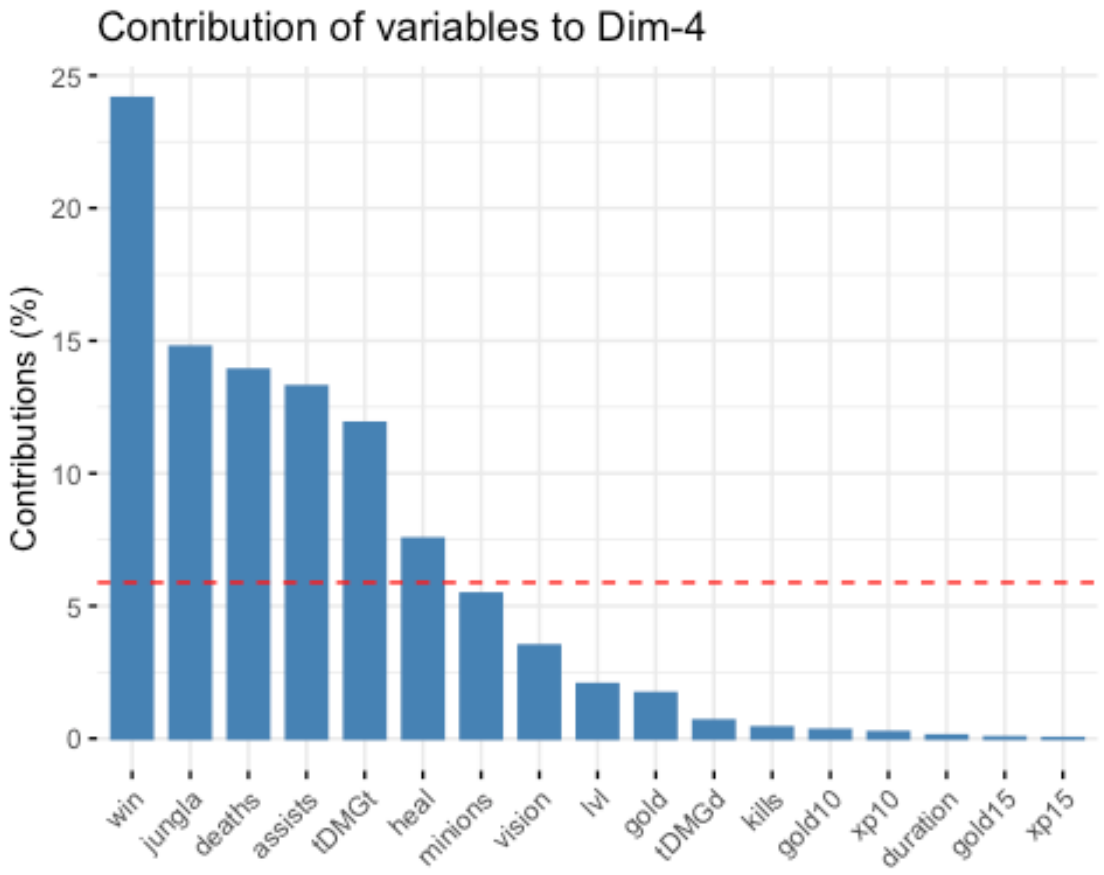
la variable jungla está muy relacionada con heal y gold, lo cual parece válido, ya que los jugadores de rol jungla (que suelen asesinar minions de la jungla) se curan más que el resto de roles, y la mayoría del oro del jungla viene de los minions de la jungla, por lo que la cantidad de minions de la jungla estará relacionada con el oro.

las variables que más contribuyen a la cuarta componente son ,win jungla y deaths, y se aprecia una fuerte correlación negativa entre win y deaths, lo cual tiene sentido ya que en las partidas cuanto más mueras más ventaja tiene el equipo contrario,




Contribution of variables to Dim-2
Contributions (%)
0
5
10
15
duration
vision
assists
tDMGt
heal
xp10
xp15
lvl
gold15
gold10
deaths
minions
gold
jungla
tDMGd
win
kills


Contribution of variables to Dim-3
Contributions (%)
20
15
10
5
0
jungla
minions
deaths
win
heal
duration
gold10
gold15
kills
assists
tDMGd
lvl
tDMGt
gold
xp10
vision
xp15




```
indices_cat <- which(descChamp2$tipo == "categorical")
shaco5=shaco[,c(indices_cat)]
shaco5=shaco5[1:300, ]
ban=
c(shaco5$t1p1_ban_champId,shaco5$t1p2_ban_champId,shaco5$t1p3_ban_champId
,shaco5$t1p4_ban_champId,shaco5$t1p5_ban_champId,shaco5$t2p1_ban_champId,
shaco5$t2p2_ban_champId,shaco5$t2p3_ban_champId,shaco5$t2p4_ban_champId,s
haco5$t2p5_ban_champId)
item0=
c(shaco5$t1p1_item0,shaco5$t1p2_item0,shaco5$t1p3_item0,shaco5$t1p4_item0
,shaco5$t1p5_item0,shaco5$t2p1_item0,shaco5$t2p2_item0,shaco5$t2p3_item0,
shaco5$t2p4_item0,shaco5$t2p5_item0)
item1=
c(shaco5$t1p1_item1,shaco5$t1p2_item1,shaco5$t1p3_item1,shaco5$t1p4_item1
,shaco5$t1p5_item1,shaco5$t2p1_item1,shaco5$t2p2_item1,shaco5$t2p3_item1,
shaco5$t2p4_item1,shaco5$t2p5_item1)
item2=
c(shaco5$t1p1_item2,shaco5$t1p2_item2,shaco5$t1p3_item2,shaco5$t1p4_item2
,shaco5$t1p5_item2,shaco5$t2p1_item2,shaco5$t2p2_item2,shaco5$t2p3_item2,
shaco5$t2p4_item2,shaco5$t2p5_item2)
item3=
c(shaco5$t1p1_item3,shaco5$t1p2_item3,shaco5$t1p3_item3,shaco5$t1p4_item3
,shaco5$t1p5_item3,shaco5$t2p1_item3,shaco5$t2p2_item3,shaco5$t2p3_item3,
```

```
shaco5$t2p4_item3,shaco5$t2p5_item3)
item4=
c(shaco5$t1p1_item4,shaco5$t1p2_item4,shaco5$t1p3_item4,shaco5$t1p4_item4
,shaco5$t1p5_item4,shaco5$t2p1_item4,shaco5$t2p2_item4,shaco5$t2p3_item4,
shaco5$t2p4_item4,shaco5$t2p5_item4)
item5=
c(shaco5$t1p1_item5,shaco5$t1p2_item5,shaco5$t1p3_item5,shaco5$t1p4_item5
,shaco5$t1p5_item5,shaco5$t2p1_item5,shaco5$t2p2_item5,shaco5$t2p3_item5,
shaco5$t2p4_item5,shaco5$t2p5_item5)
item6=
c(shaco5$t1p1_item6,shaco5$t1p2_item6,shaco5$t1p3_item6,shaco5$t1p4_item6
,shaco5$t1p5_item6,shaco5$t2p1_item6,shaco5$t2p2_item6,shaco5$t2p3_item6,
shaco5$t2p4_item6,shaco5$t2p5_item6)
runa1=
c(shaco5$t1p1_perkPrimaryStyle,shaco5$t1p2_perkPrimaryStyle,shaco5$t1p3_p
erkPrimaryStyle,shaco5$t1p4_perkPrimaryStyle,shaco5$t1p5_perkPrimaryStyle
,shaco5$t2p1_perkPrimaryStyle,shaco5$t2p2_perkPrimaryStyle,shaco5$t2p3_pe
rkPrimaryStyle,shaco5$t2p4_perkPrimaryStyle,shaco5$t2p5_perkPrimaryStyle)
runa2=
c(shaco5$t1p1_perkSubStyle,shaco5$t1p2_perkSubStyle,shaco5$t1p3_perkSubSt
yle,shaco5$t1p4_perkSubStyle,shaco5$t1p5_perkSubStyle,shaco5$t2p1_perkSub
Style,shaco5$t2p2_perkSubStyle,shaco5$t2p3_perkSubStyle,shaco5$t2p4_perkS
ubStyle,shaco5$t2p5_perkSubStyle)
summoner1=
c(shaco5$t1p1_spellId1,shaco5$t1p2_spellId1,shaco5$t1p3_spellId1,shaco5$t
1p4_spellId1,shaco5$t1p5_spellId1,shaco5$t2p1_spellId1,shaco5$t2p2_spellI
d1,shaco5$t2p3_spellId1,shaco5$t2p4_spellId1,shaco5$t2p5_spellId1)
summoner2=
c(shaco5$t1p1_spellId2,shaco5$t1p2_spellId2,shaco5$t1p3_spellId2,shaco5$t
1p4_spellId2,shaco5$t1p5_spellId2,shaco5$t2p1_spellId2,shaco5$t2p2_spellI
d2,shaco5$t2p3_spellId2,shaco5$t2p4_spellId2,shaco5$t2p5_spellId2)

tabla2 <- data.frame(

 champ =champs,
 ban= ban,
 summoner1= summoner1,
 summoner2= summoner2,
 item0= item0,
 item1= item1,
 item2= item2,
 item3= item3,
 item4= item4,
 item5= item5,
 item6= item6,
 runa1= runa1,
 runa2= runa2,
  rol = rol,
  win = win
)
```

```
write.csv(tabla2, file = "tabla2.csv", row.names = FALSE)
write.csv(tabla, file = "tabla.csv", row.names = FALSE)
```

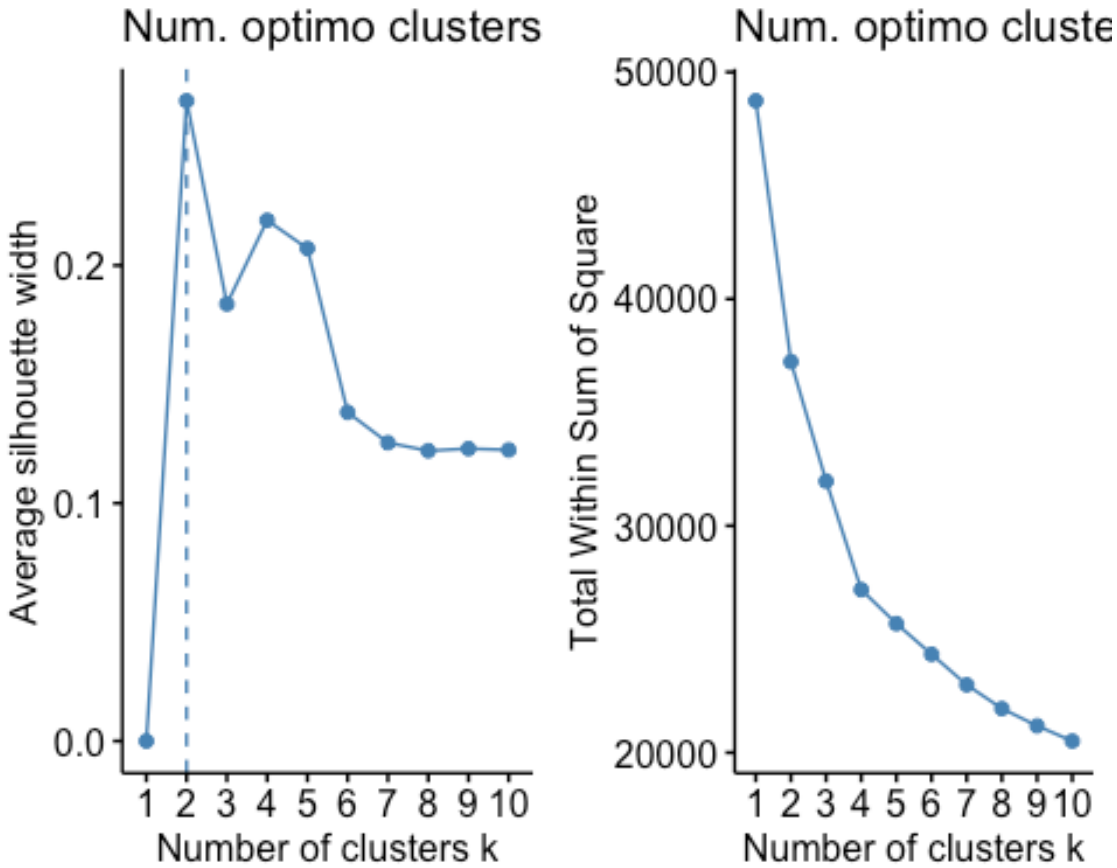
Clustering

Una vez tratdos los datos en el apartado PCA y visto su resultado, se realizara un analisis de clustering que dividira un conjunto heterogéneo de elementos que en nustro caso son los jugadores de league of legends en grupos, en función de las similitudes o diferencias entre ellos.

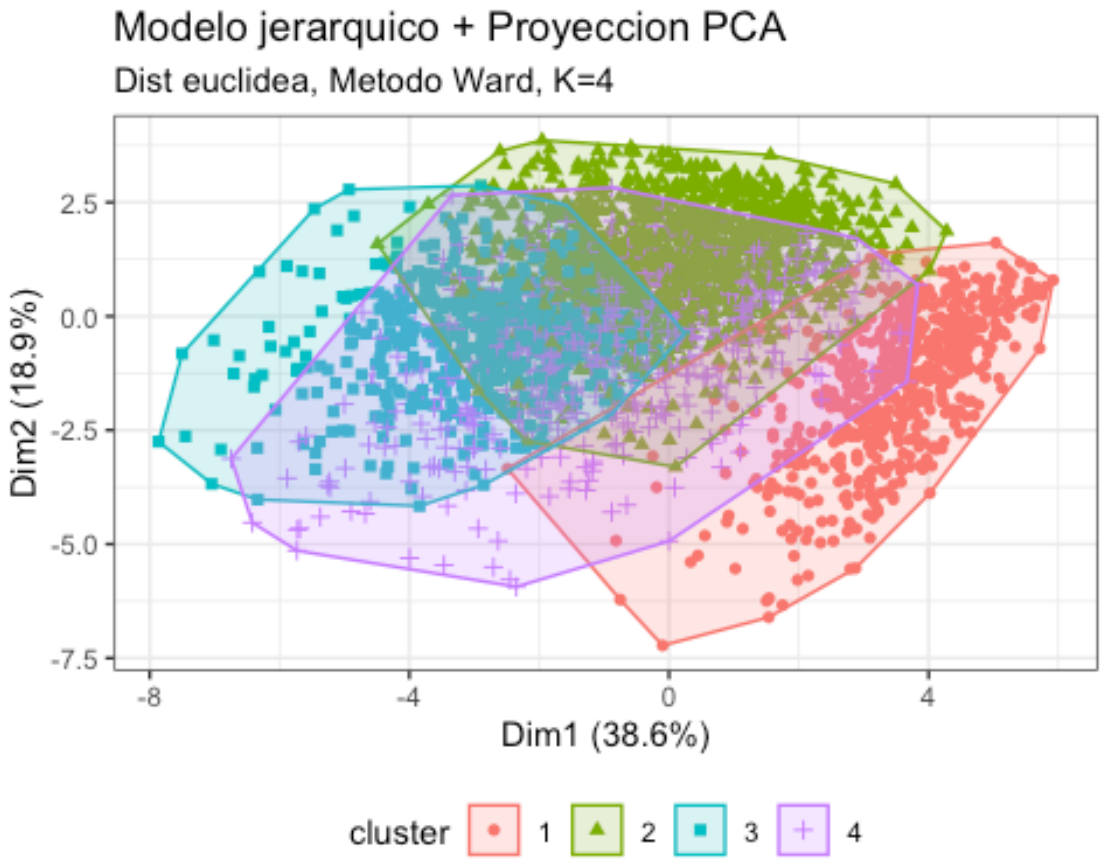
Antes de pasar al analisis de clusering habria que ver inicialmente si en nuestros datos existe una tendencia de agrupamiento, para ello vemos la siguiente matriz de distancias.

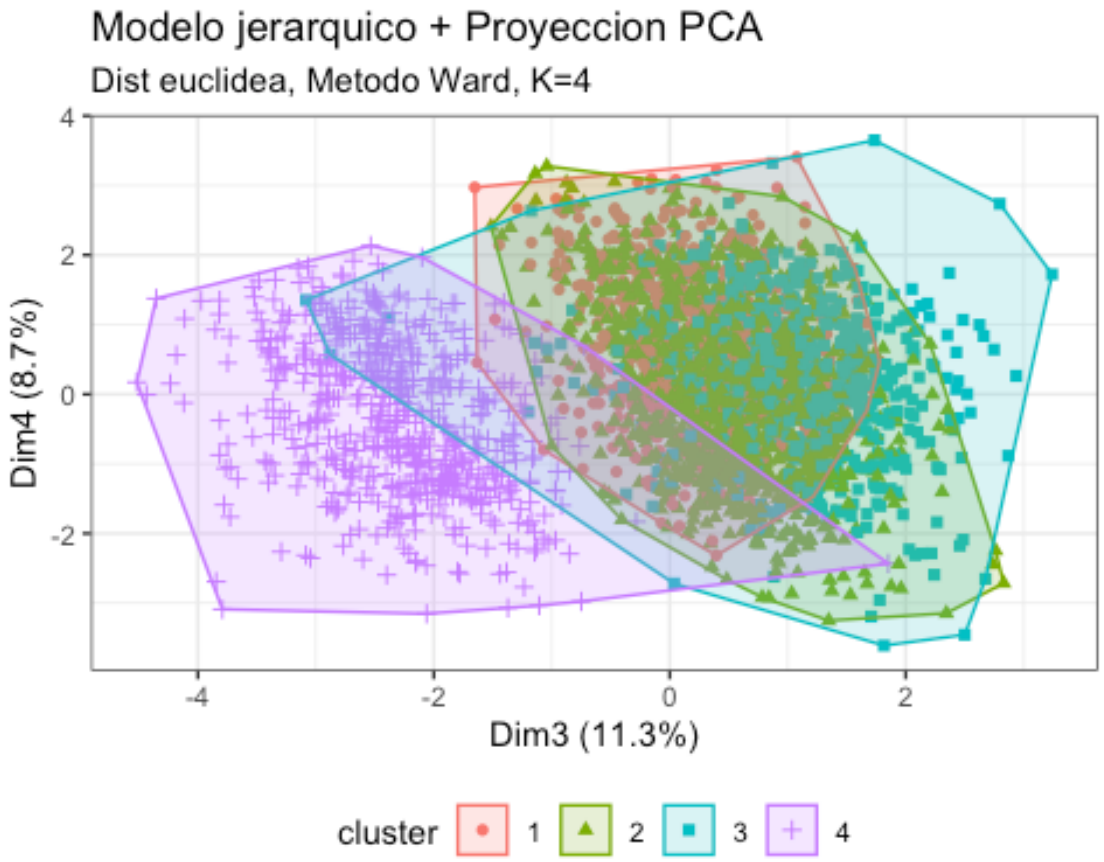
Los sombreados rojos indican una tendencia de formacion de un grupo de individuos que se puede ver a simple vista que son varios. Para confirmar dicha tendencia de agrupamiento se calcula el coeficiente de Hopkins el cual al ser mas proximo a 1 (0.878) indica una buena tendencia de agrupacion. Dicho coeficiente en estos datos es bastante alto, por ende significa una buena tendencia de agrupacion.

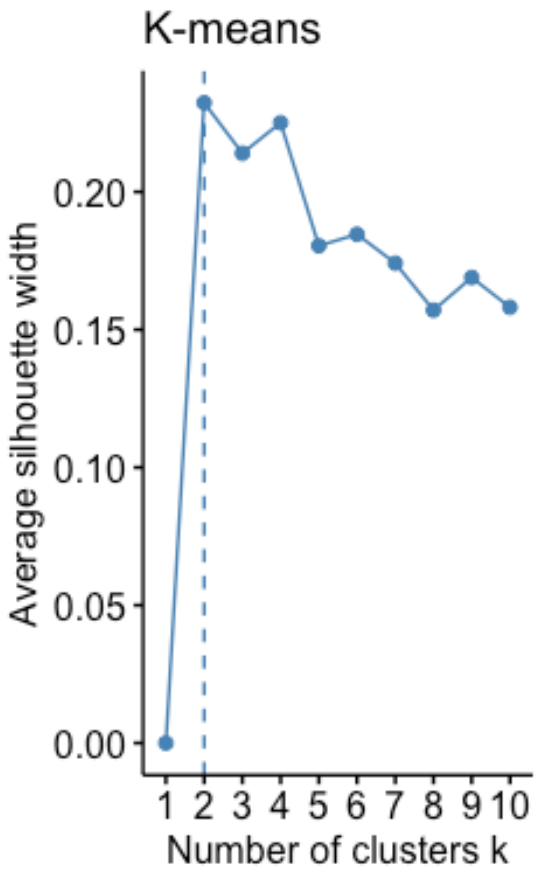
Una vez visto todo esto, se probaran diferentes modelos jerarquicos y de particion y se quedara con el mejor de ellos.

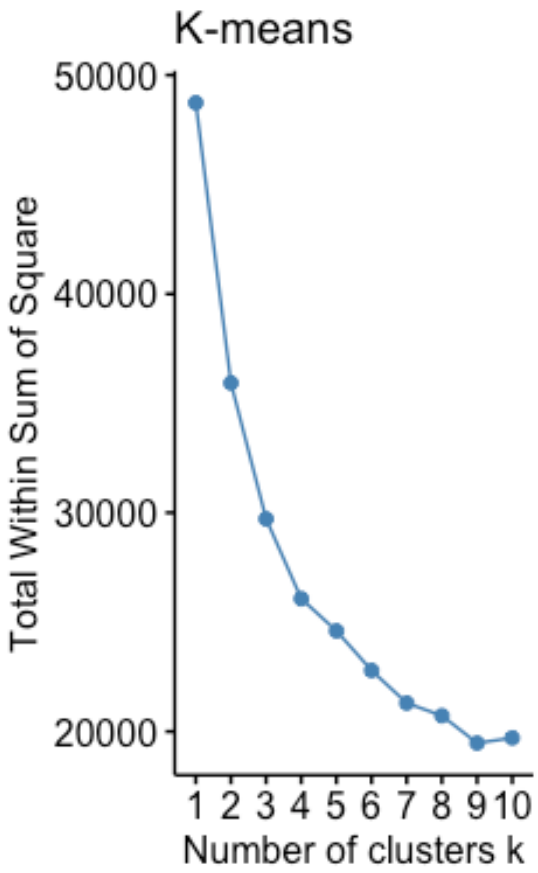


```
## grupos1
##    1    2    3    4
##  604 1236  561  599
```

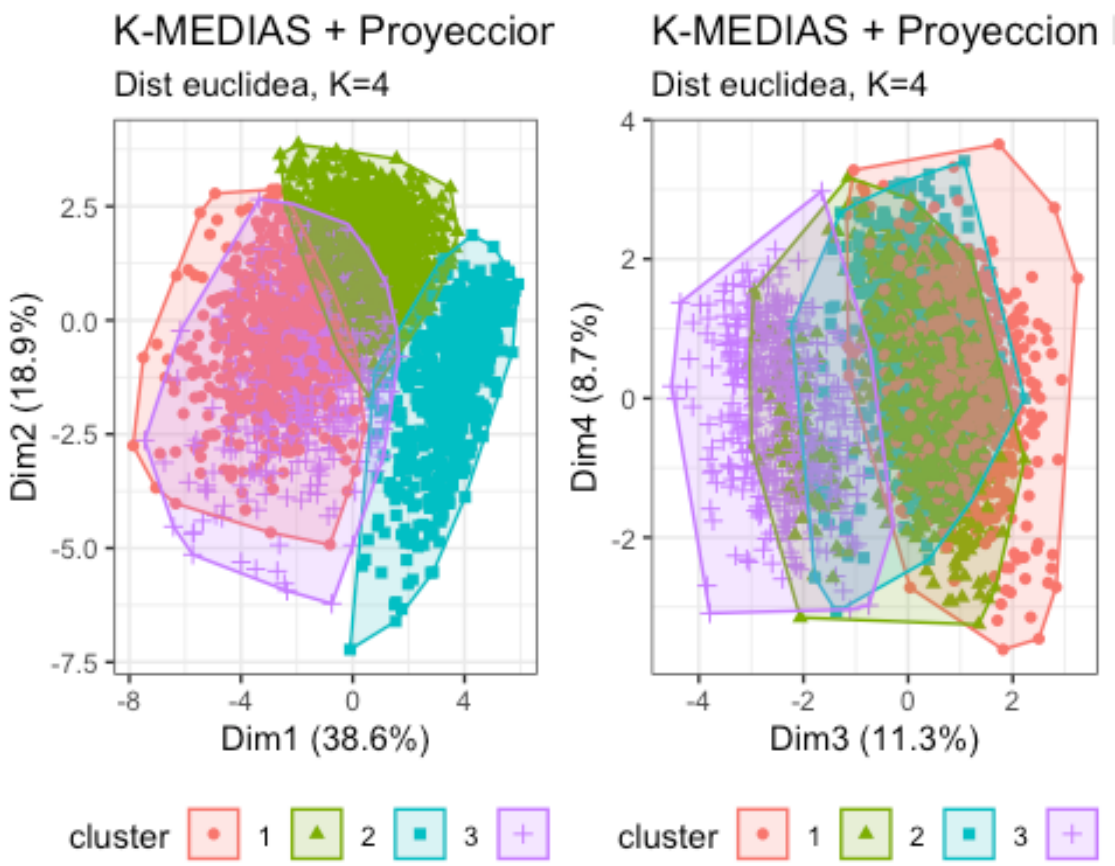

Modelo jerarquico + Proyeccion PCA
Dist euclidea, Metodo Ward, K=4
Dim2 (18.9%)
2.5
0.0
-2.5
-5.0
-7.5
-8
-4
0
4
Dim1 (38.6%)
cluster
1
2
3
4

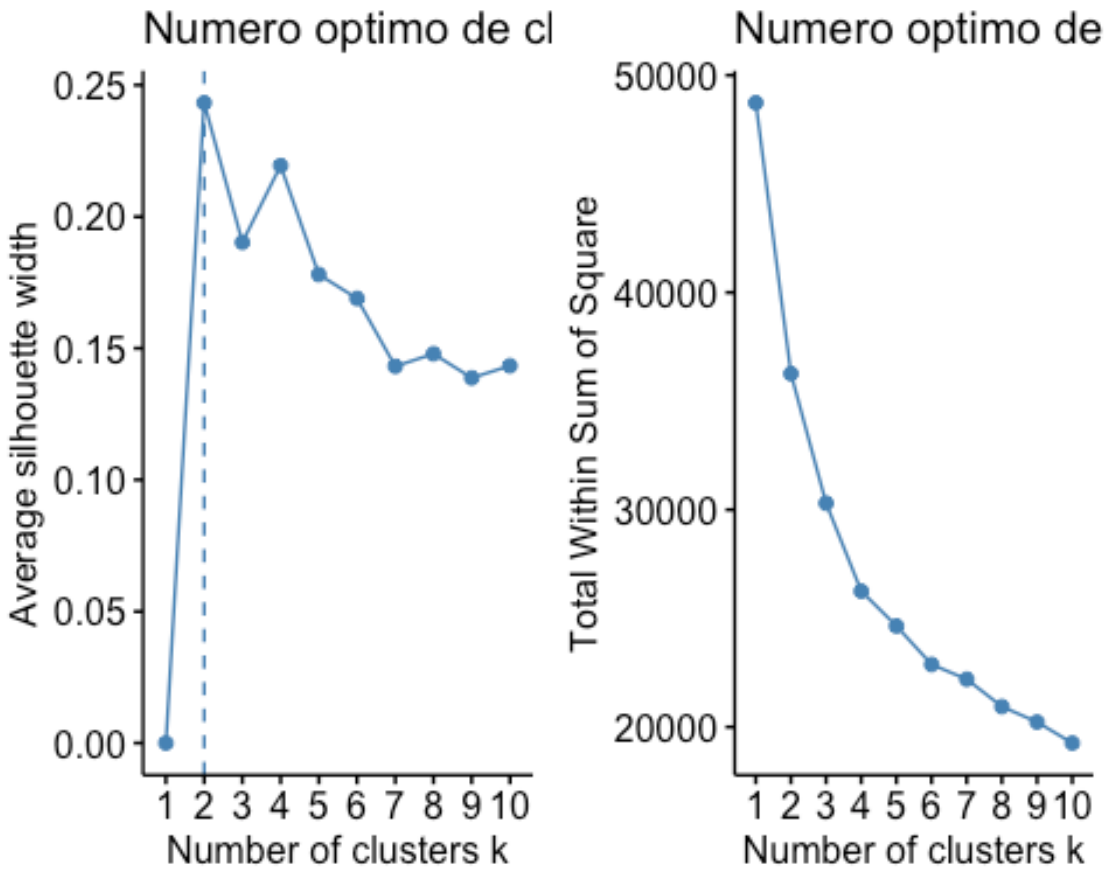

Modelo jerarquico + Proyeccion PCA
Dist euclidea, Metodo Ward, K=4
Dim4 (8.7%)
Dim3 (11.3%)
4
2
0
-2
-4
-2
0
2
cluster
1
2
3
4



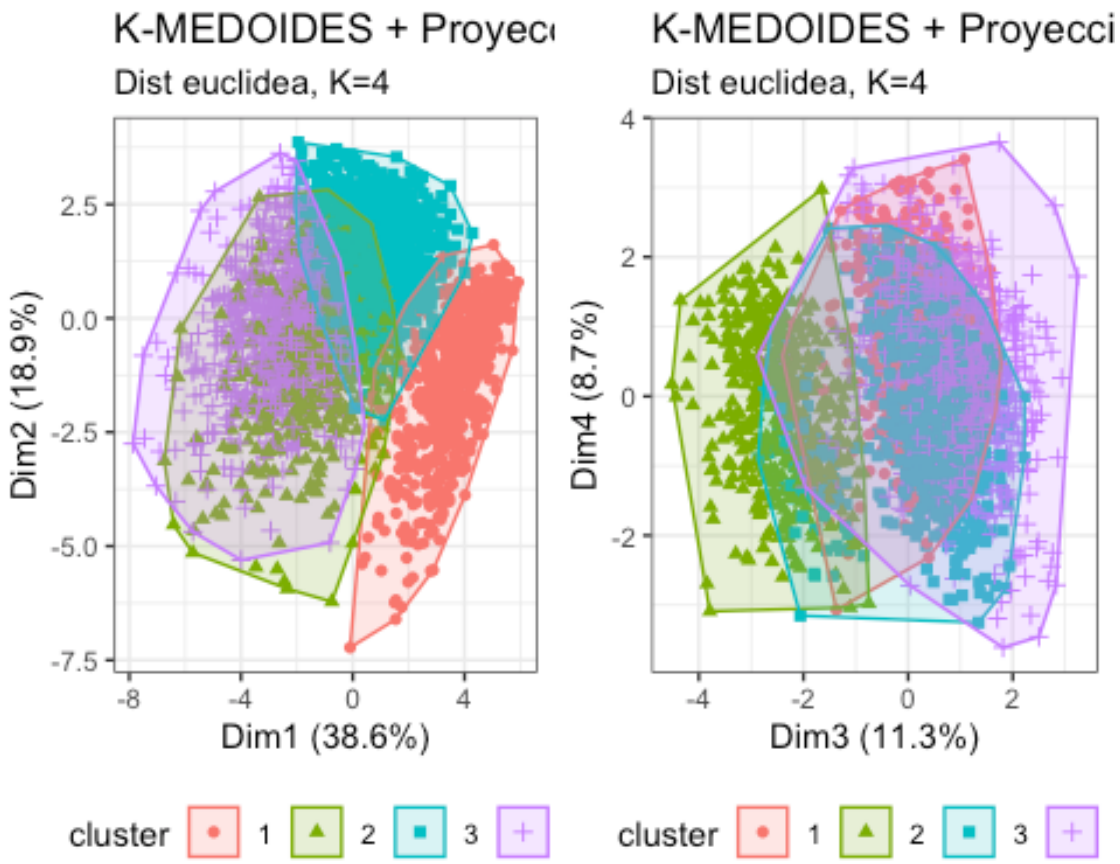


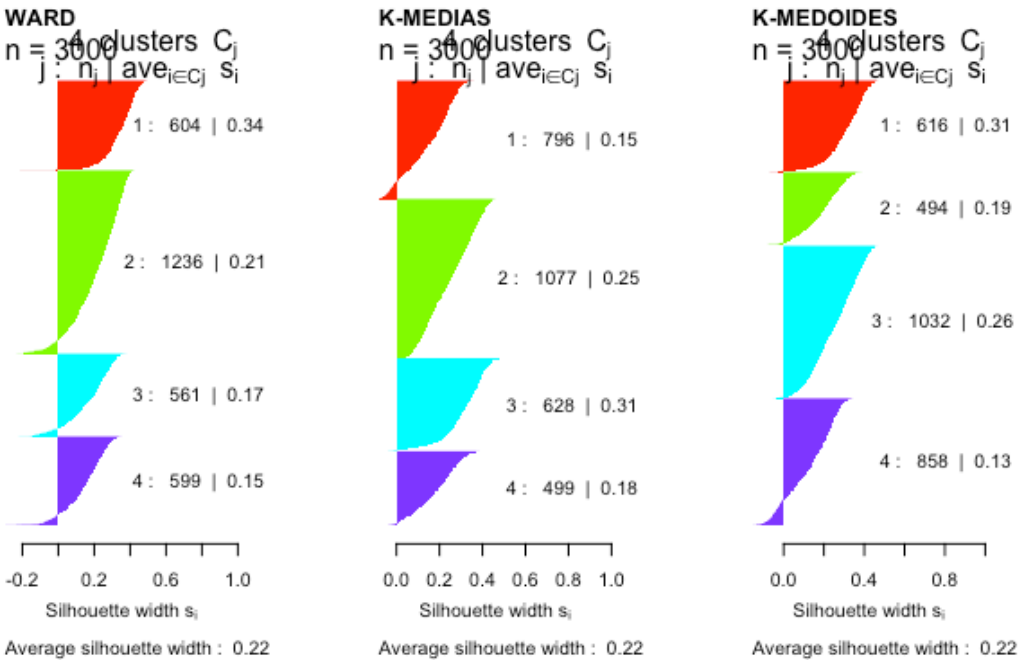

```
##
##    1    2    3    4
##  796 1077  628  499
```


K-MEDIAS + Proyeccior
Dist euclidea, K=4
Dim2 (18.9%)
Dim1 (38.6%)
2.5
0.0
-2.5
-5.0
-7.5
-8
-4
0
4
cluster
1
2
3
K-MEDIAS + Proyeccion
Dist euclidea, K=4
Dim4 (8.7%)
Dim3 (11.3%)
4
2
0
-2
-4
-2
0
2
cluster
1
2
3



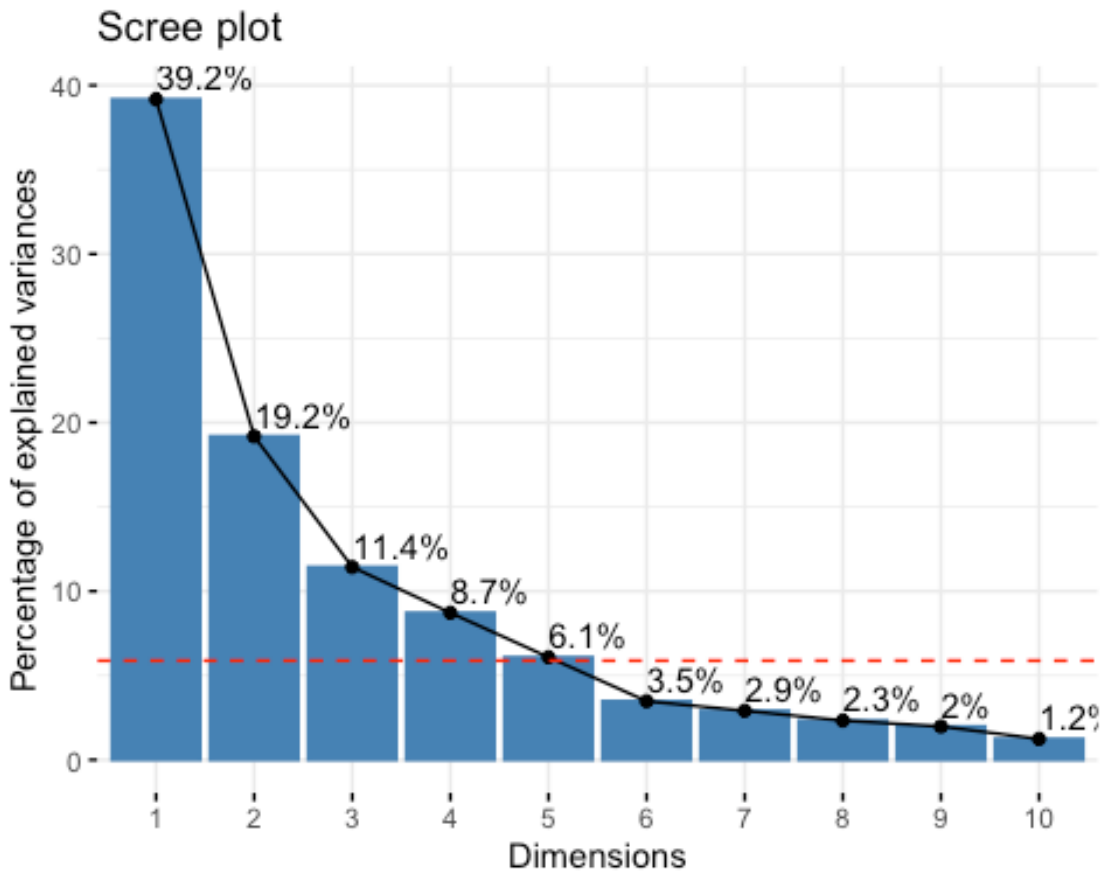

```
##
##    1    2    3    4
##  616  494 1032  858
```

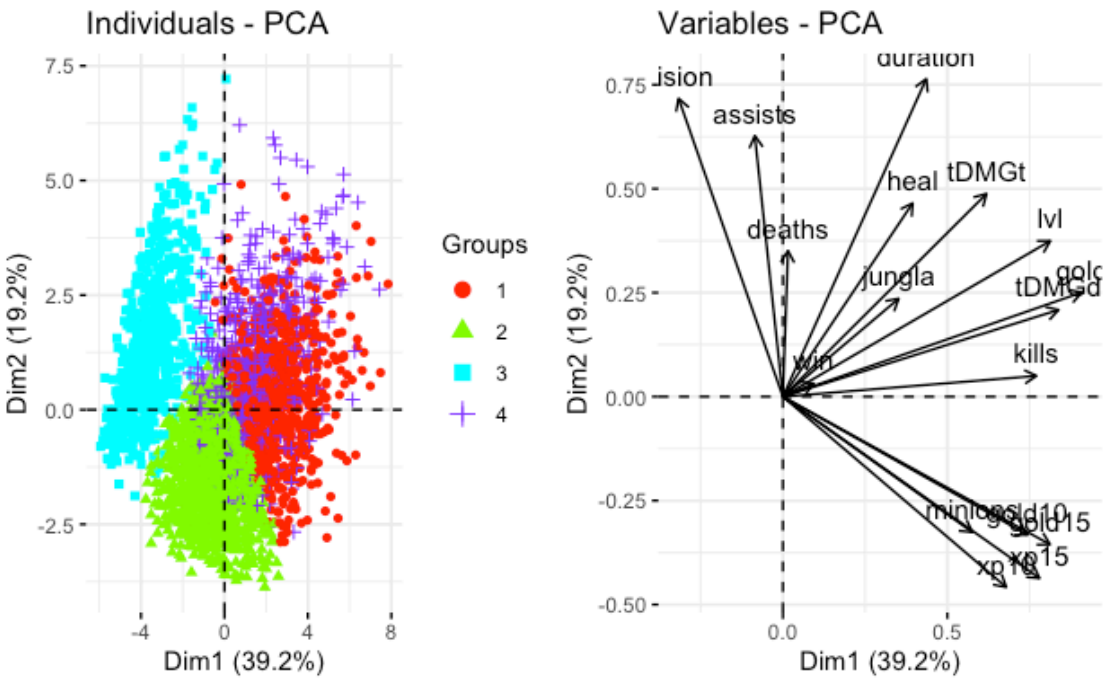




El metodo escogido en este caso ha sido el de K-medias debido a que agrupa mucho mejor los datos que el resto de los metodos. Y se han escogido 4 clusters en dicho

metodo con el criterio del metodo de Codo, que se basa en escoger el cluster que mayor variabilidad explique teniendo en cuenta que se busca disminuir la SCR. Por ello, la opcion mas optima han sido 4 clusters.





Despues de sacar este grafico la interpretacion de cada cluster es la siguiente:

- Clúster 1 (Rojo) Este grupo se caracteriza por enfocarse en el combate directo y la obtención rápida de recursos. Son jugadores agresivos, acumulando numerosas eliminaciones y recolectando oro y experiencia de manera eficiente durante todas las fases del juego. Son los encargados de realizar el daño y acaparar el oro y el nivel del equipo. Es por esta razón, que su contribución a la partida es determinante en el desarrollo de la partida. Los jugadores de este grupo suelen tener los mejores resultados.

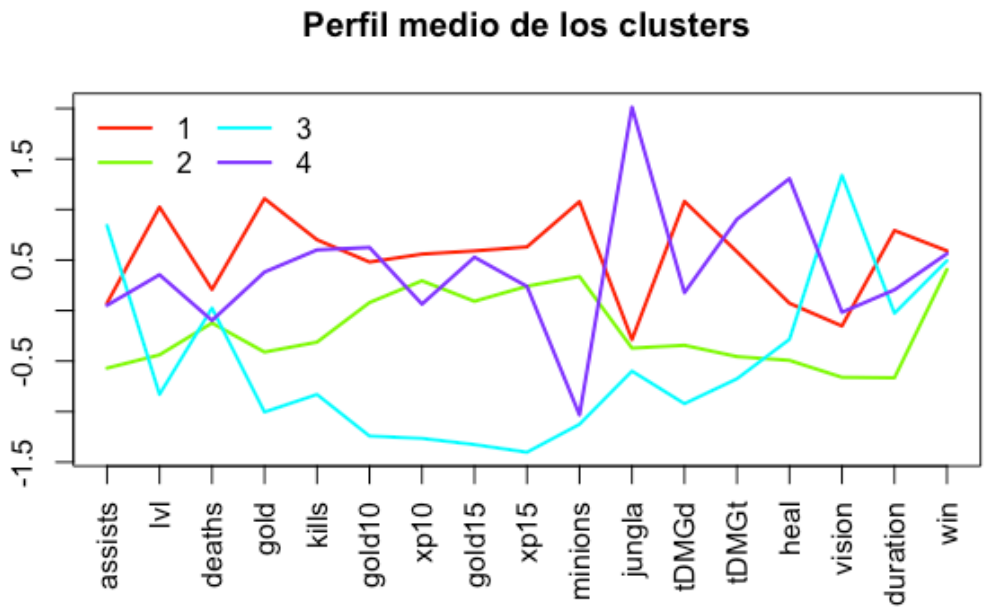
- Clúster 2 (Verde) Este grupo se caracteriza por un estilo de juego extremadamente pasivo y de escasa contribución al equipo. Estos jugadores suelen presentar un bajo número de muertes, asistencias, daño recibido e infligido, así como una escasa provisión de visión en el mapa. Tienden a jugar de manera individualista, lo que implica que, en lugar de ser un jugador activo para el equipo, a menudo se convierten en una carga debido a su mínima participación y apoyo en las estrategias conjuntas, y por tanto, queda un poco relegado. A pesar de tener pocas muertes, su escasa contribución al juego en términos de apoyo y ejecución de objetivos los hace menos efectivo y valioso para el éxito del equipo.

- Clúster 3 (Azul claro) Este grupo coincide con un rol especifico del juego, que es el de soporte. Se caracteriza por un estilo de juego más cooperativo, de forma que se centran en apoyar a los jugadores más relevantes en la partida para que estos destaquen. El rol de soporte no necesita recursos para ser importante en la partida y este, se sacrifica para beneficiar a sus compañeros. Aunque tienden a causar poco daño, destacan por su elevado número de asistencias y puntos de visión del mapa. Este tipo de jugador puede

identificarse como un jugador de apoyo. A pesar de su baja contribución en términos de daño y eliminaciones, representan un pilar fundamental en las peleas en equipo (teamfights), teniendo funciones muy diversas como iniciar peleas, curar a los aliados, paralizar al enemigo entre otras. Son esenciales para la coordinación y el éxito del equipo.

- Cluster 4(Morado) Este grupo también coincide con un rol específico del juego, que es el de jungla. Este rol tambien se centra en ayudar al equipo, pero de una forma más activa que los soportes. Contribuyen en la mayor parte de los objetivos de la partida y pueden ayudar en todos los sectores del mapa con más facilidad que los soportes. En este rol la función es bastante variable, ya que pueden acumular recursos o cederlos al resto del equipo dependiendo de la partida y el personaje. Destaca en la variable de jungla, que representa la principal forma de ganar oro y experiencia de este rol. También tienen un valor alto de la variable heal, ya que a medida que van ganando oro matando minioms en la jungla se van curando. Además, tienen un valor de minioms bajo, porque los junglas se centran en ganar oro y experiencia de una forma diferente al resto de jugadores. Estos jugadores suelen ser más decisivos en el éxito del equipo que los soportes, pero menos que los jugadores rojos.

	c1	c2	c3	c4
assists	0.07	-0.57	0.84	0.05
lvl	1.02	-0.44	-0.83	0.35
deaths	0.21	-0.12	0.02	-0.10
gold	1.11	-0.41	-1.00	0.38
kills	0.70	-0.31	-0.83	0.60
gold10	0.48	0.08	-1.24	0.62
xp10	0.56	0.29	-1.26	0.06
gold15	0.59	0.09	-1.33	0.53
xp15	0.63	0.24	-1.40	0.24
minions	1.08	0.34	-1.13	-1.03
jungla	-0.29	-0.37	-0.60	2.01
tDMGd	1.08	-0.34	-0.92	0.18
tDMGt	0.58	-0.46	-0.67	0.90
heal	0.07	-0.49	-0.28	1.31
vision	-0.15	-0.66	1.34	-0.02
duration	0.79	-0.67	-0.03	0.21
win	0.59	0.41	0.49	0.56



En este grafico se ve mucho mejor la diferencia de caracteristicas de cada tipo de grupo explicado anteriormente (sobretodo rojos y morados). La conclusión de estos clusters aplicandos al contexto de league of legends está dividida en dos partes. Si el jugador es un tipo de jugador enfocado a aportar al equipo, el objetivo de este debe ser apoyar a los tipos de jugadores que más se parezcan a los del grupo rojo. Por otro lado, si el jugador es un tipo de jugador más lider, se recomienda seguir los pasos del grupo de jugadores rojos y evitar parecerse a los individuos del grupo verde.