**Informe código en R- Diagnóstico de COVID-19 a partir del análisis por inteligencia artificial de espectros MALDI-TOF de hisopados nasofaríngeos**

**1.Carga de datos**

Se cargaron los archivos .RDA de cada institución. Los mismos contienen files con los espectros promediados de cada cepa y files con la metadata. Se modificaron los dataframes de la metadata de manera que cada uno de ellos contenga explícitamente la fecha de procesamiento, la institución y el equipo. Se diferencia institución y equipo ya que en el caso de INBIRS y Hospital de Clínicas, las muestras se procesaron en el MALDI del Hospital de Clínicas.

Luego, los archivos de cada institución se fusionaron de manera de tener un único file con espectros y otro con la metadata.

**2. Análisis exploratorio de datos**

**2.1 Preprocesamiento de los datos**

Utilizando la librería MALDIquant, se realizó el alineamiento y la búsqueda de picos utilizando un threshold de 0.2. Este valor permitió obtener 63 picos. Cuando se realizó la prueba con un threshold de 0.5 se obtuvieron tan sólo 19 picos.

**2.2. Estadísticas**

Se analizó cantidad de espectros positivos y negativos por equipo, la cantidad de espectros con determinada cantidad de picos, y las intensidades promedio, máxima y mínima por pico.

|  |  |  |
| --- | --- | --- |
| **institución** | **covid\_status** | **frecuencia** |
| CR | Cov.Neg | 8 |
| HdC | Cov.Neg | 120 |
| inbirs | Cov.Neg | 152 |
| Malbran | Cov.Neg | 48 |
| CR | Cov.Pos | 26 |
| HdC | Cov.Pos | 69 |
| inbirs | Cov.Pos | 73 |
| Malbran | Cov.Pos | 46 |

**Tabla 1**. Espectros positivos y negativos por institución.

En cuanto a los valores de Ct, 191 de las 542 muestras que no poseen datos, algunas tienen valores puntuales de Ct, y otras están agrupadas como low/int/high/neg clasificación que se usarán más adelante para hacer otro análisis y ver si se mejoran los resultados.

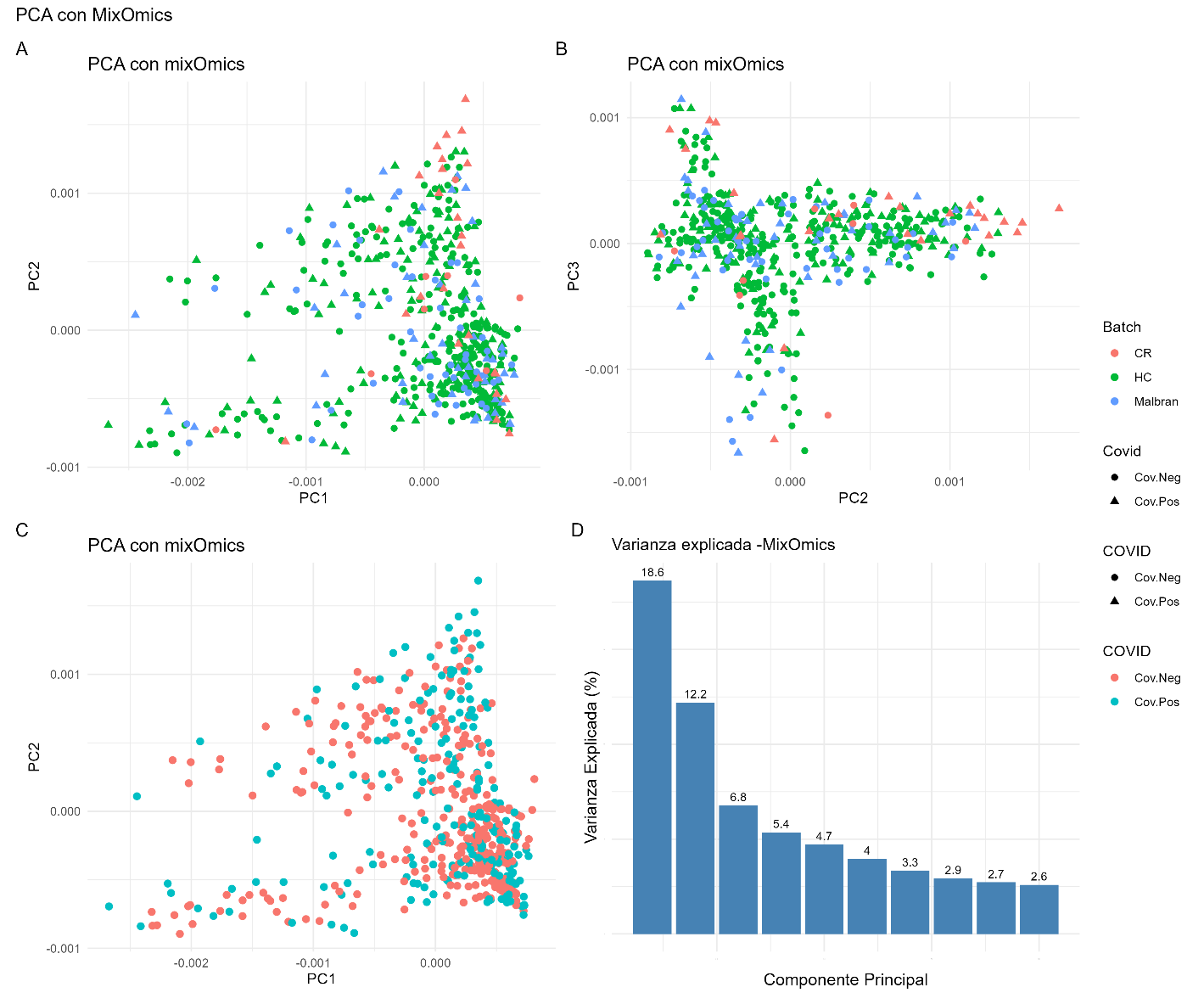
**2.3. Reducción de la dimensionalidad**

2.3.1 PCA

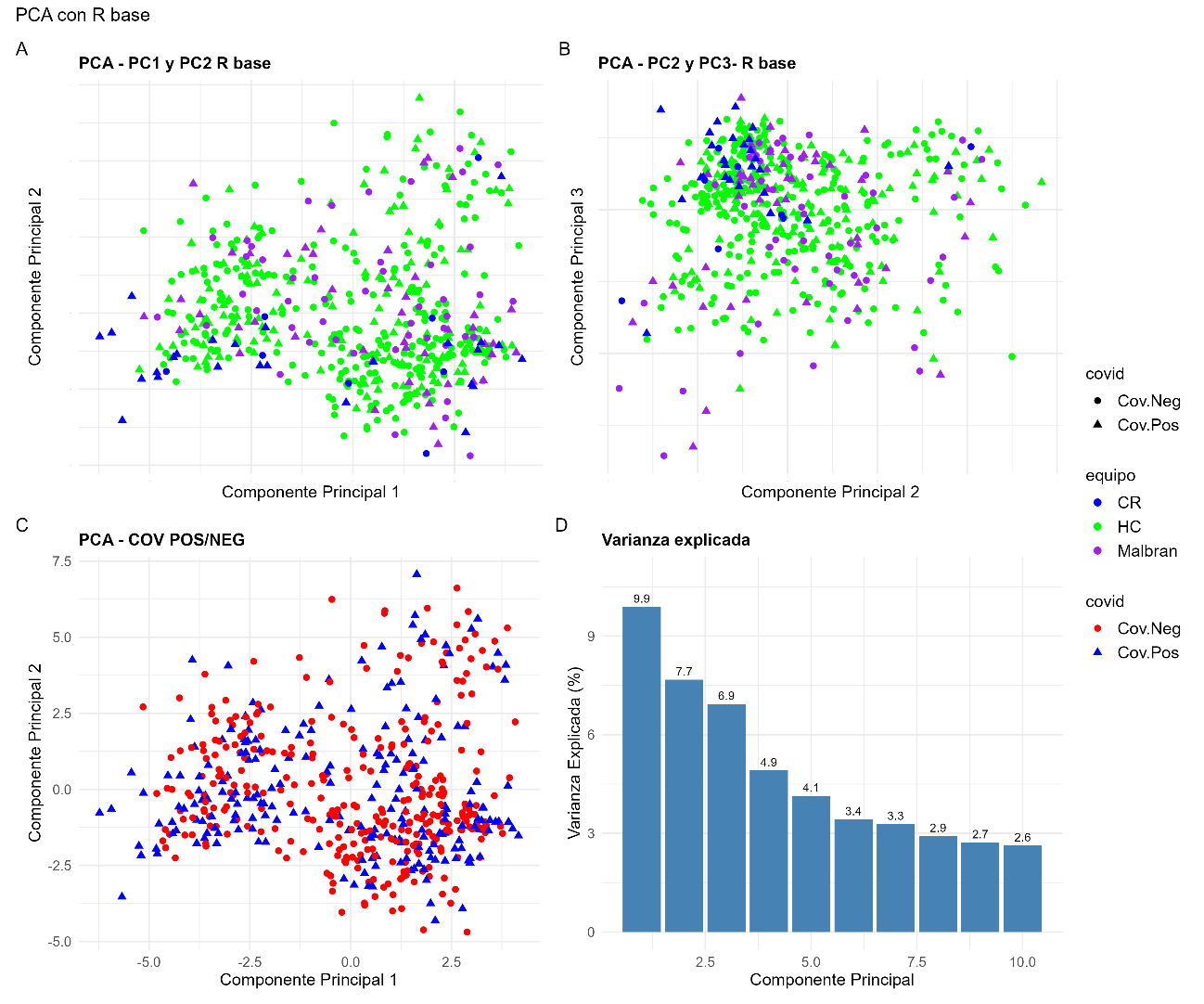
Se realizó PCA tanto con R base como con la librería MixOmics con el objetivo de identificar patrones en los datos y reducir la dimensionalidad, preservando la mayor parte de la variabilidad.

El paquete MixOmics proporciona una implementación más sofisticada y especializada de PCA, diseñada para datos multiómicos.

En la figura 1 y 2 podemos observar que las muestras se encuentran dispersas y no hay patrones entre equipos, sólo puede detectarse una aparente concentración de puntos del equipo de Costa Rica analizando PC2 y PC3 con R base. De igual forma, se puede pensar en una gran cantidad de “outliers”, que son aquellos que se alejan del concentrado de puntos. En cuanto a la variabilidad explicada, los valores son menores con R base pero en ambos casos son muy bajos para las primeras componentes.



**Figura 1.** A-Visualización de las componentes 1 y 2 del análisis de componentes principales diferenciando por equipo y por resultado de COVID. B- Visualización de las componentes 2 y 3 del análisis de componentes principales diferenciando por equipo y por resultado de COVID. C-Visualización de las componentes según el resultado de COVID. D-Histograma de varianza explicada de cada una de las componentes.



**Figura 2.** A-Visualización de las componentes 1 y 2 del análisis de componentes principales diferenciando por equipo y por resultado de COVID. B- Visualización de las componentes 2 y 3 del análisis de componentes principales diferenciando por equipo y por resultado de COVID. C-Visualización de las componentes según el resultado de COVID. D-Histograma de varianza explicada de cada una de las componentes.

2.3.2 UMAP

UMAP: Aproximación y Proyección de Manifold Uniforme para Reducción de Dimensiones.

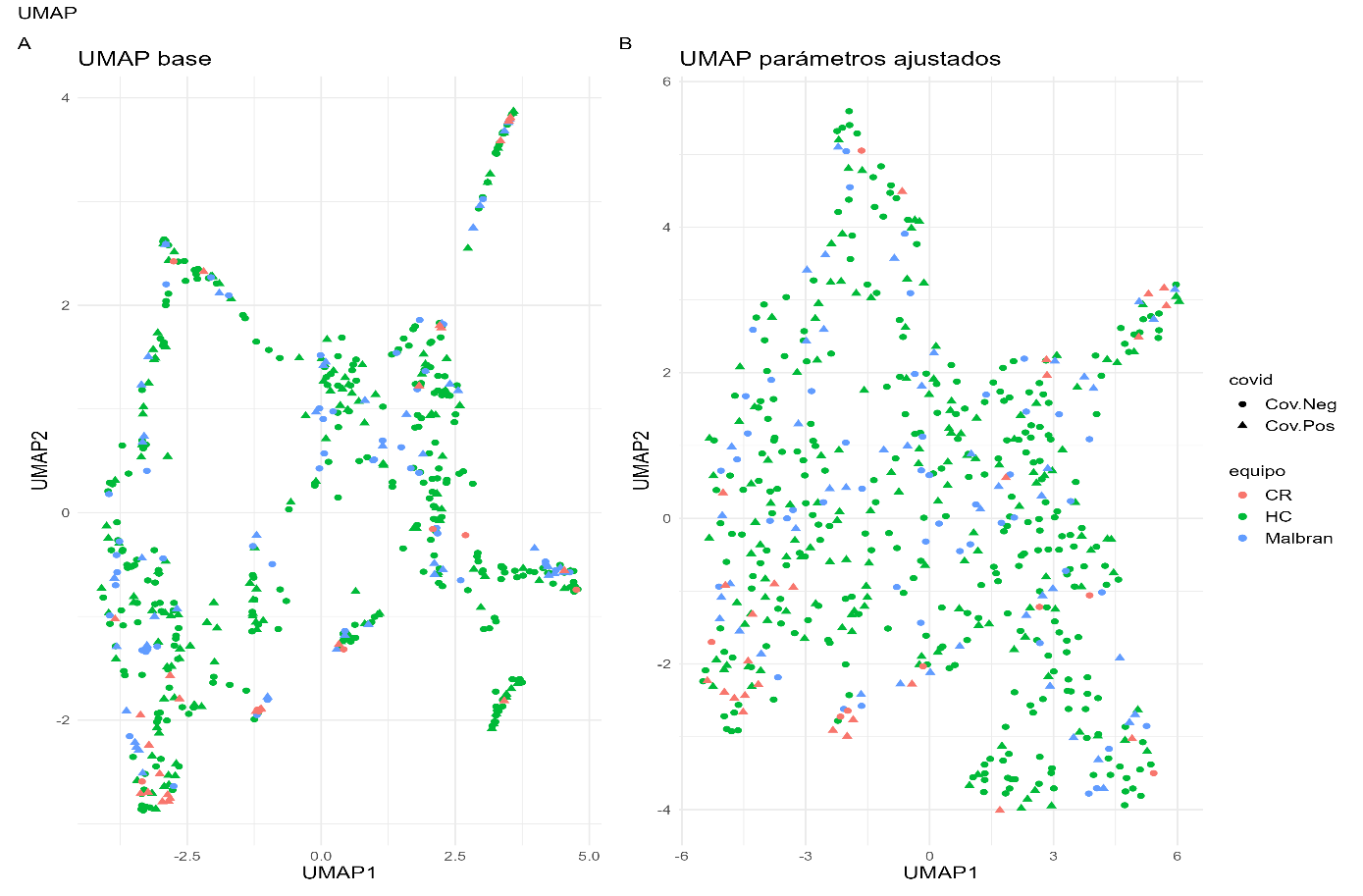
La Aproximación y Proyección de Manifold Uniforme (UMAP) es una técnica de reducción de dimensiones que se puede utilizar para visualización de manera similar a t-SNE, pero también para una reducción de dimensiones no lineal general. El algoritmo se basa en tres suposiciones sobre los datos:

-Los datos están distribuidos uniformemente en una manifold riemanniana (Un **manifold** es una estructura subyacente que organiza datos complejos de alta dimensionalidad en una forma más simple y estructurada, pero aún representativa. UMAP asume que tus datos están en un manifold y usa esta idea para proyectarlos en menos dimensiones, ayudándote a visualizarlos y entenderlos mejor).

-La métrica riemanniana es localmente constante (o puede ser aproximada como tal);

-La “manifold” es localmente conexa (, en cualquier punto de la manifold, es posible encontrar una pequeña región alrededor de ese punto que esté completamente conectada).

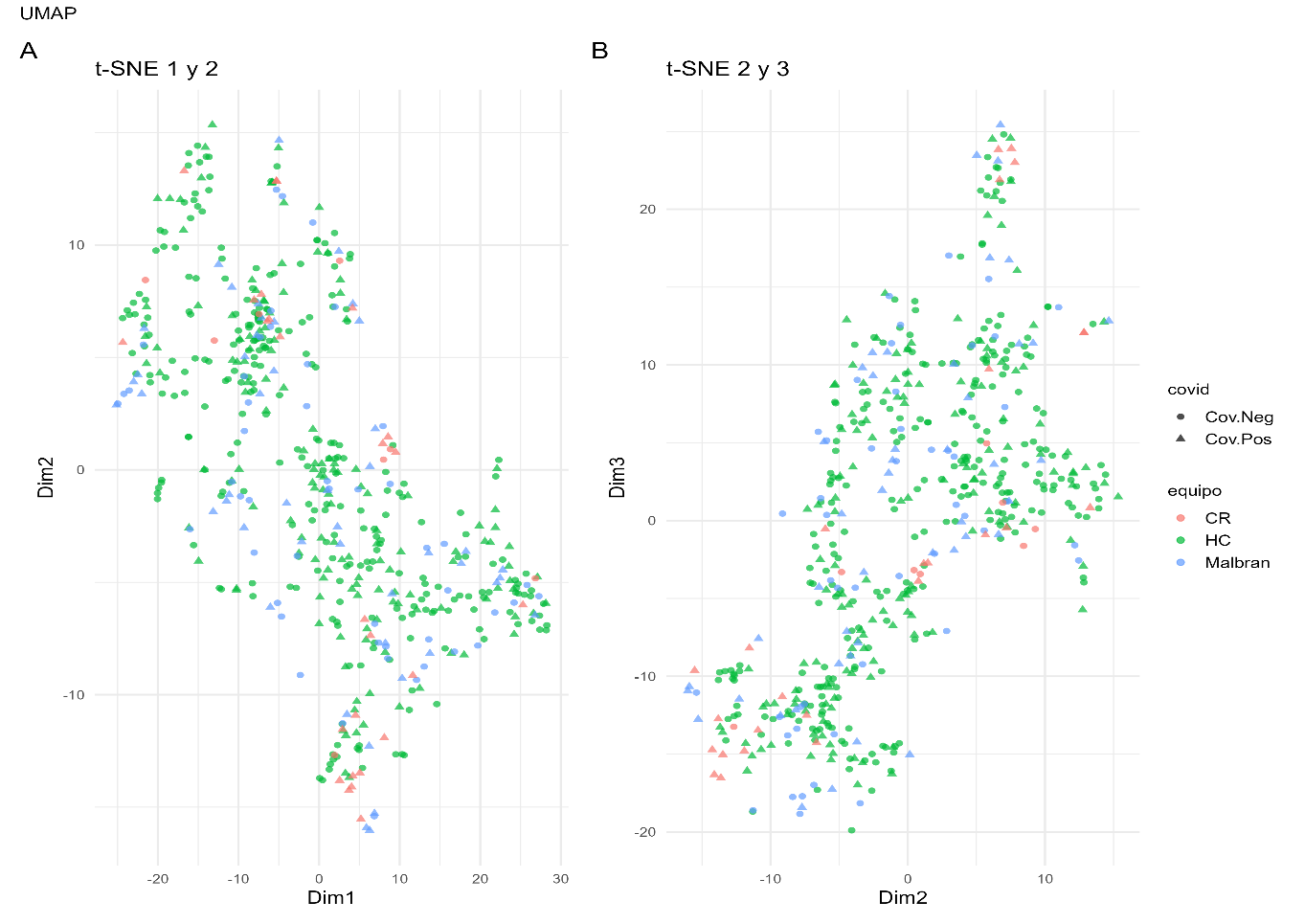
Según la verificación de las suposiciones chatGPT dice: Verificar directamente las suposiciones de UMAP sobre la manifold es un desafío técnico, pero en la práctica se puede hacer observando cómo se comportan las técnicas de reducción de dimensionalidad (como UMAP o t-SNE) con tus datos. Si los datos se agrupan de manera coherente y los puntos son conectables mediante vecinos más cercanos, es probable que las suposiciones sean razonables. Si encuentras que los datos no se agrupan bien o presentan una estructura muy dispersa, entonces tal vez las suposiciones no se cumplen tan bien y necesitarías explorar otras técnicas o transformaciones de los datos.



**Figura 3.** A-UMAP base. B- UMAP ajustado con los siguientes parámetros: n\_neighbors = 15, metric = "euclidean", min\_dist = 0.5

En estas figuras no se observa ningún patrón, ya que las muestras de los distintos equipos y las positivas y negativas se encuentran dispersas.

2.3.3 tSNE



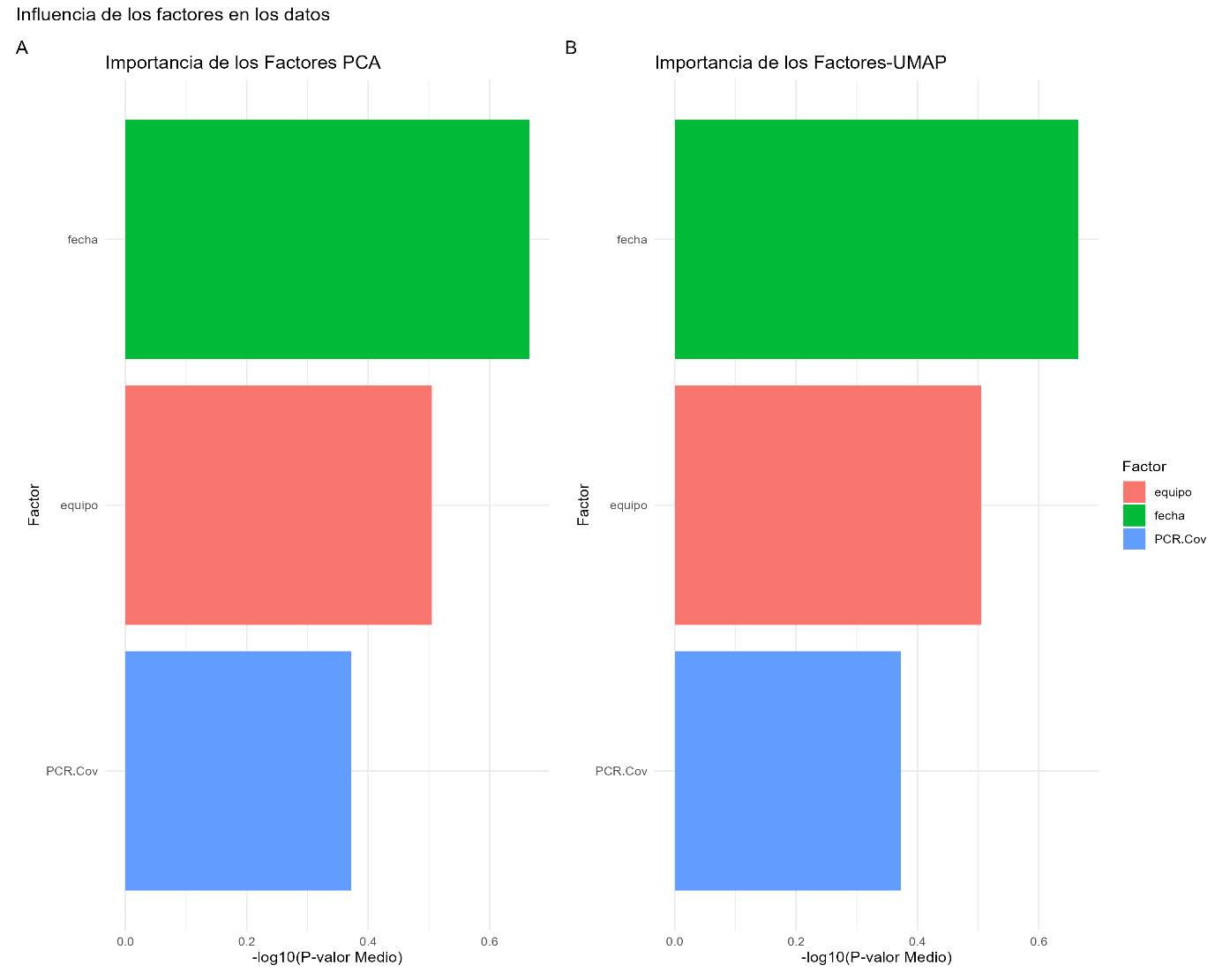
**Figura 4.** A Dimensiones 1 y 2 de t-SNE. B Dimensiones 2 y 3 de t-SNE

2.3.4 Análisis de los pesos de los factores

-Shapiro-Wilk y Levene’s Test evaluaron los supuestos de normalidad y homogeneidad de varianzas, necesarios para decidir si usar pruebas paramétricas o no paramétricas.

-Como los datos no son normales, se elige Kruskal-Wallis en lugar de ANOVA porque no asume normalidad.

Se evaluó el peso de diferentes factores (fecha, resultado de COVID y equipo) en los componentes principales obtenidos mediante PCA y UMAP, utilizando la prueba no paramétrica de Kruskal-Wallis.

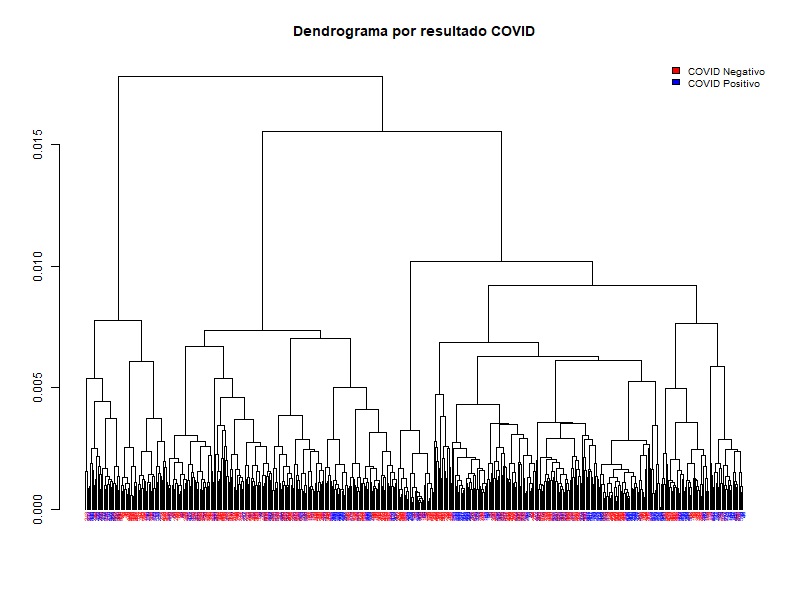


**Figura 5.** Análisis mediante Kruskal Wallis de la importancia de las variables en las componentes obtenidas por A- PCA B-UMAP.

**2.5. Análisis de Clustering para Identificación de Patrones**

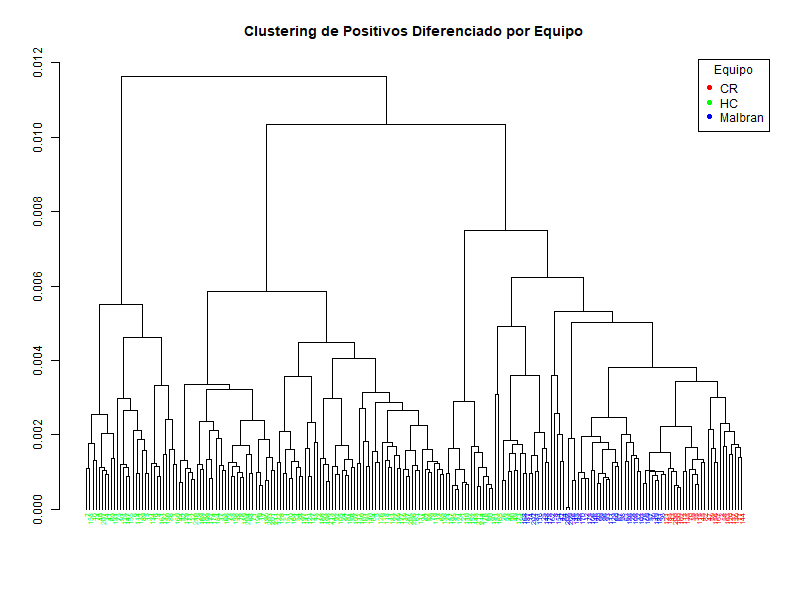
Se generó un dendrograma basado en un análisis de clustering jerárquico para evaluar si las muestras pueden agruparse en función de su resultado de COVID por PCR (positivo o negativo). En el gráfico, las muestras están etiquetadas con colores, donde el rojo representa las muestras COVID negativas y el azul las positivas.

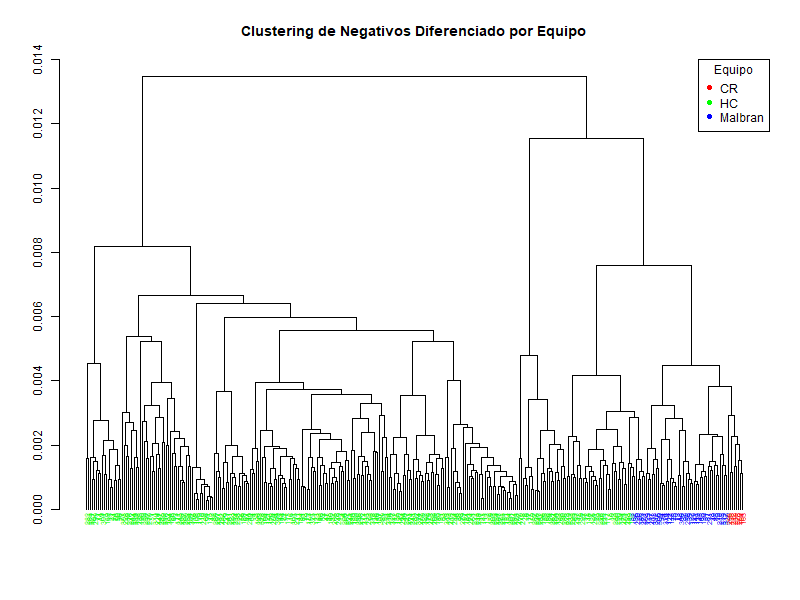
A partir de la visualización, no se observa una clara separación o patrón definido que agrupe consistentemente las muestras según el resultado de COVID. Esto sugiere que las características incluidas en el análisis no permiten discriminar de forma efectiva entre los dos grupos mediante este método de clustering.



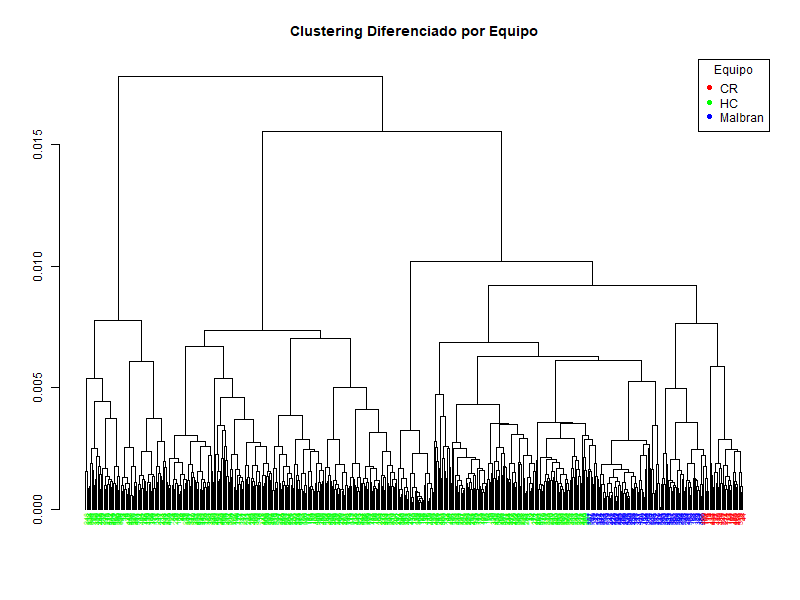
**Figura 6.** Dendrograma realizado según resultado de COVID

Luego se realizó un dendrograma teniendo en cuenta los distintos equipos donde se procesaron las muestras, y se evaluó si se observaba un patrón diferente si se diferenciaba COVID positivo o negativo.

**Figura 7.** Clusterización por equipo en muestras COVID positivas



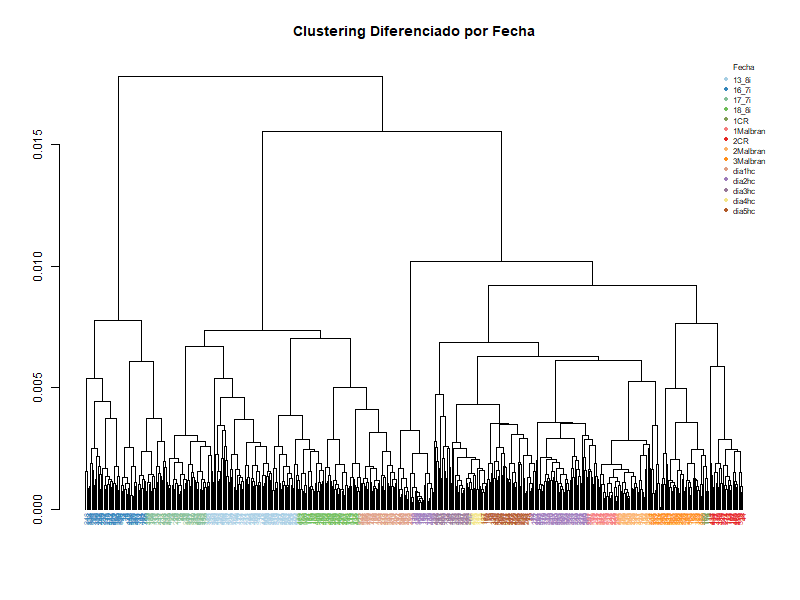
**Figura 8.** Clusterización por equipo en muestras COVID negativas



**Figura 9.** Clusterización por equipo.

Las figuras 7, 8 y 9 muestran un claro patrón de clusterización en base al equipo, independientemente del resultado de COVID obtenido por PCR.

Posteriormente se realizó la misma prueba con las distintas fechas de procesamiento de las muestras en los distintos equipos. En la figura 10 se puede observar un claro agrupamiento según fecha de largada.

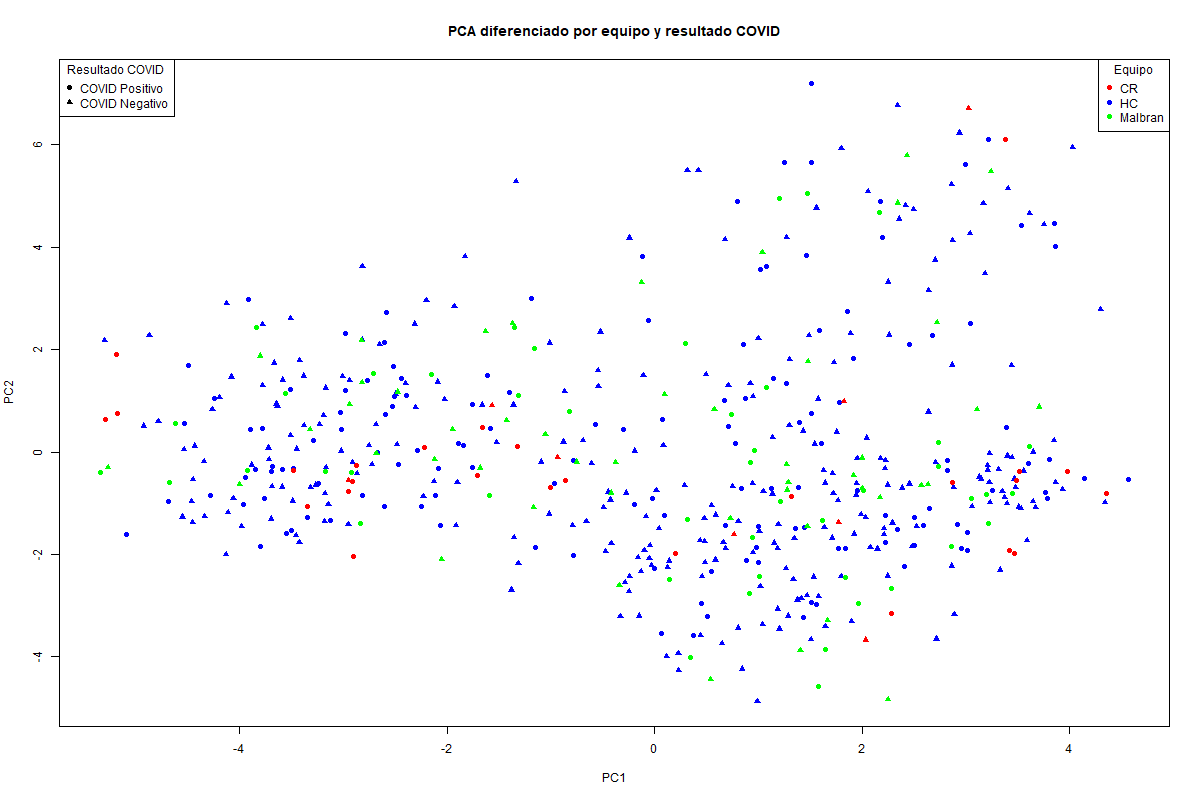


**Figura 10.** Clusterización por fecha

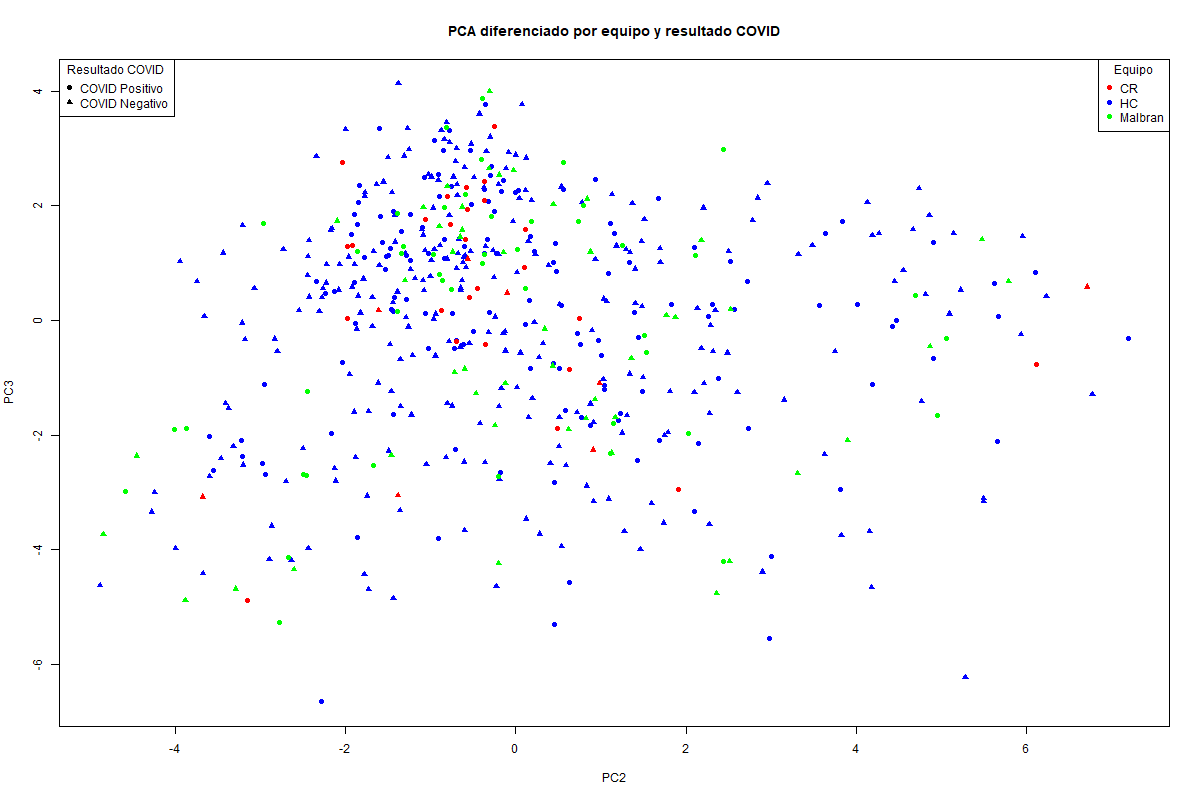
**2.6 Correcion efecto batch: COMBAT**

2.6.1 Corrección por equipo

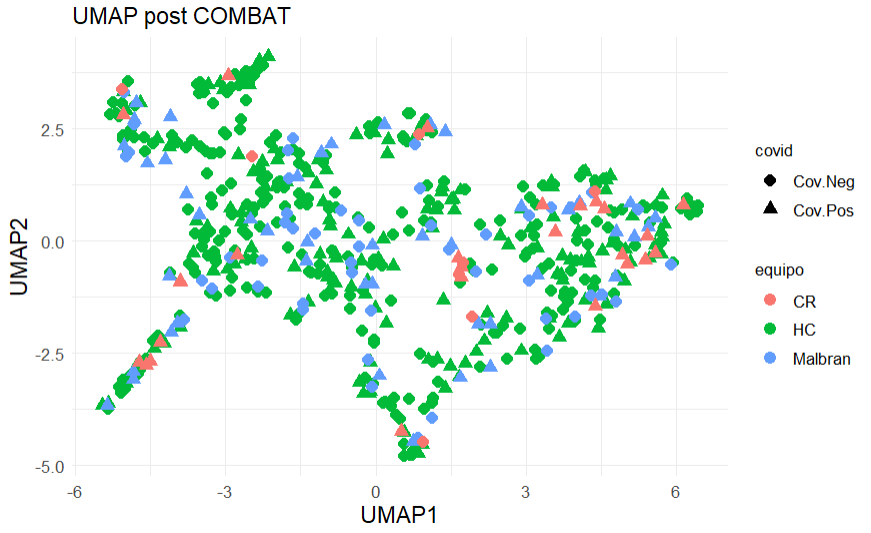
En la figura 11 se puede observar que al corregir por equipo se visualizan los puntos más separados. En la figura 12 aún se observan las muestras de CR “conglomeradas” (pero tmb observo las de HPC ASÍ) al igual que en los datos pre-Combat



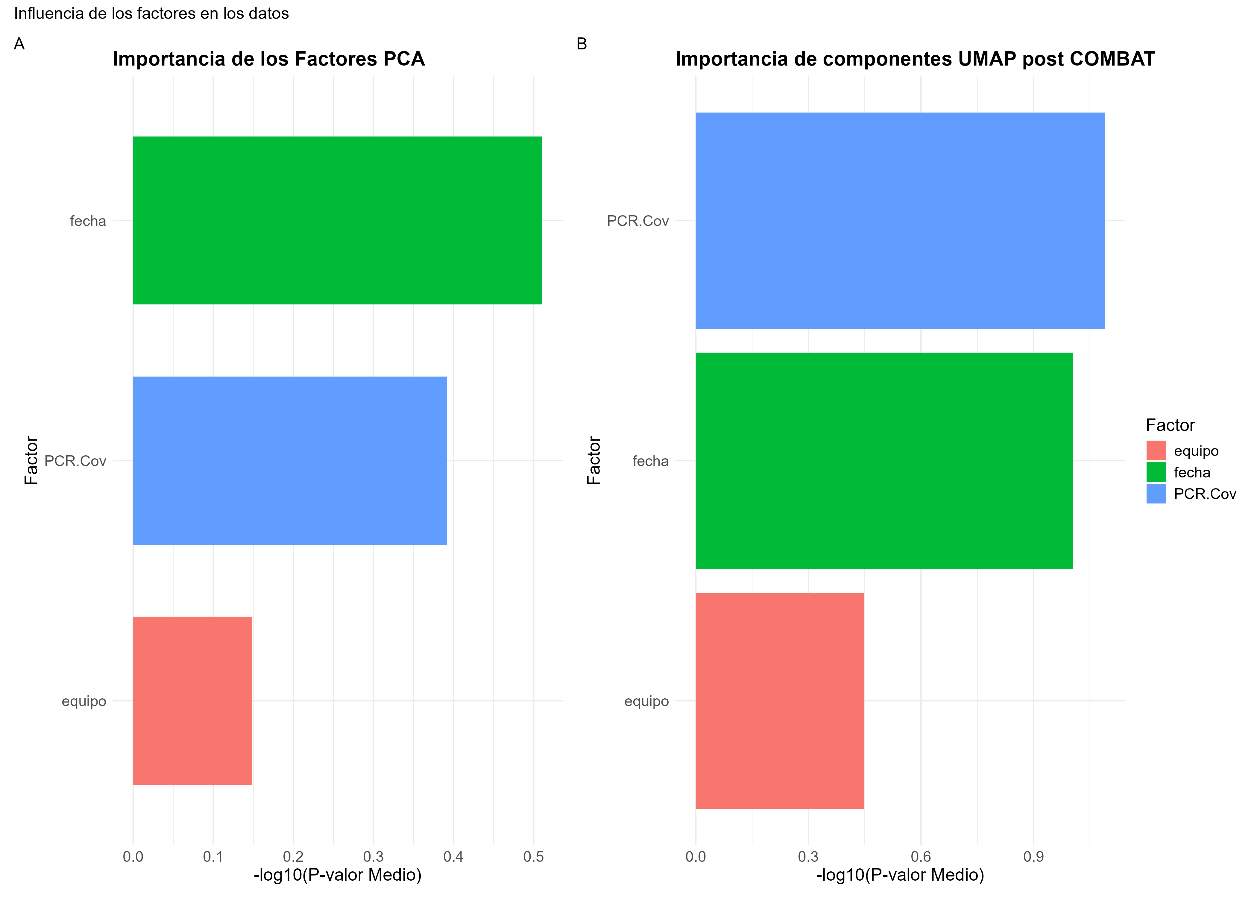
**Figura 11**. PCA post Combat PC1 y PC2



**Figura 12**. PCA post Combat PC2 y PC3

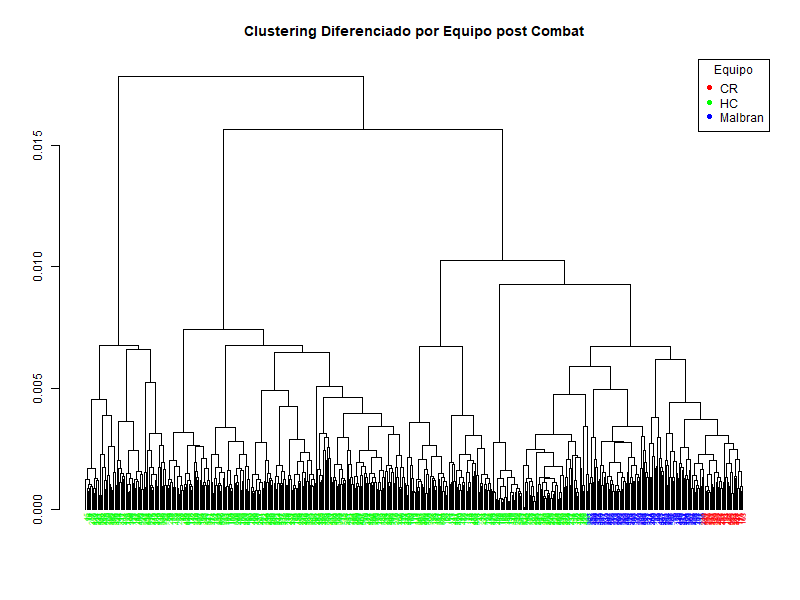


**Figura 13**. UMAP post Combat



**Figura 14.** Análisis mediante Kruskal Wallis de la importancia de las variables en las componentes obtenidas post Combat por A- PCA B-UMAP.

En la figura 14 se observan diferentes “importancias” en caso de PCA y UMAP

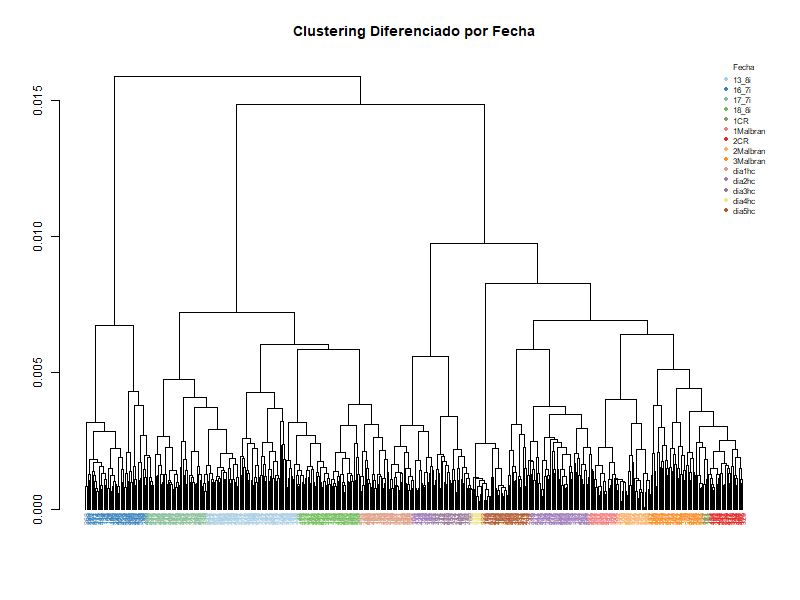


**Figura 15.** Clusterización por equipo post Combat.

En la figura 15 se observa la clusterización de los datos obtenidos luego de la corrección del efecto batch, diferenciando la proveniencia de los espectros. No se detecta mejoría respecto al dendrograma obtenido con la matriz de intensidad original.

2.6.1 Corrección por fecha

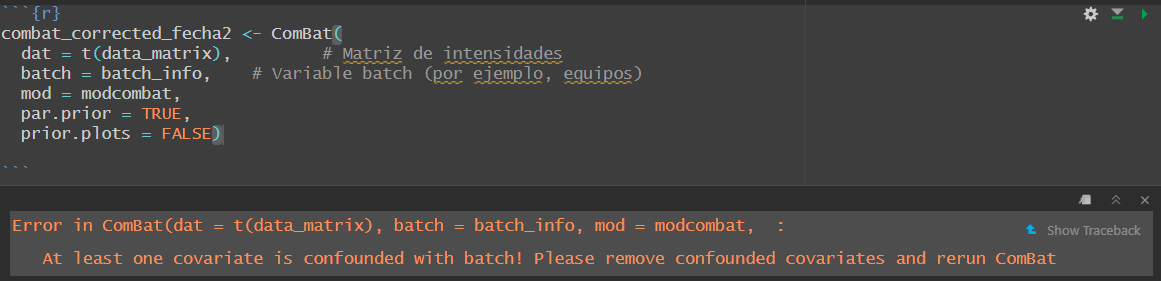
Cuando se realizó la corrección por fecha, se obtuvo este warning: Found 6 genes with uniform expression within a single batch (all zeros); these will not be adjusted for batch.



**Figura 16.** Dendrograma post corrección de efecto batch con Combat por fecha

También se realizaron los dendrogramas para combat por equipo pero todos se ven exactamente igual que los originales.

Cuando se intentó hacer corrección por batch de fecha teninedo en cuenta el equipo (modcombat <- model.matrix(~ equipo, data = Datos\_actualizados)) se obtuvo el siguiente error:



**3. Modelos**

Se tomaron las matrices de intensidad después de la corrección por combat por equipo y por fecha, dicotomizadas y no dicotomizadas, así como la matriz original dicotomizada y no dicotomizada para generar los modelos. Se probó RF, random forest con ranger, GLMNET, KNN.

Con el código que me diste probé ranger:

cuál es la diferencia entre utilizar esto que usé yo para sacar variables con varianza cero:

nzv <- nearZeroVar(featureMatrix\_dicho)

featureMatrix\_clean <- featureMatrix\_dicho[, -nzv]

y las que usaste vos:

Train.rf <- as.data.frame(trainData) %>%

dplyr::mutate(sensi = Y)

objeto\_recipe <- recipe(formula = sensi ~ .,

data = Train.rf)

objeto\_recipe <- objeto\_recipe %>%

step\_nzv(all\_predictors())

trained\_recipe <- prep(objeto\_recipe, training = Train.rf)

Train.rf <- bake(trained\_recipe, new\_data = Train.rf)

Test.r <- bake(trained\_recipe, new\_data = Test.r)

Cómo me aseguro que X1=1 y X2=2?

Otra pregunta que tengo es: la eliminación de las variables es sólo con matrices dicotomizadas, no? No hay manera de que elimine variables correlacionadas si no están dicotomizados?

3.1 Resultados de modelos con matrix corregida por efecto batch por equipo dicotomizados y no dicotomizados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo** | **Accuracy** | **Sensibilidad** | **Especificidad** | **AUC** | **kappa** |
| RF\_equipodic | 0.72 | 0.47 | 0.88 | 0.75 | 0.37 |
| GLMC\_equipodic | 0.74 | 0.58 | 0.85 | 0.76 | 0.06 |
| KNN\_equipodic | 0.63 | 0.65 | 0.62 | 0.69 | 0.26 |
| Ranger\_equipodic | 0.69 | 0.42 | 0.86 | 0.76 | 0.30 |
| RF\_equipo | 0.71 | 0.40 | 0.91 | 0.71 | 0.15 |
| GLM\_equipo | 0.62 | 0.07 | 0.98 | 0.46 | 0.06 |
| KNN\_equipo | 0.63 | 0.28 | 0.86 | 0.69 | 0.16 |
| Ranger\_equipo | 0.69 | 0.35 | 0.91 | 0.77 | 0.28 |
|  |  |  |  |  |  |

3.2 Resultados de modelos con corrección por fecha dicotomizados y no dicotomizados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo** | **Accuracy** | **Sensibilidad** | **Especificidad** | **AUC** | **kappa** |
| RF\_fechadic | 0.73 | 0.65 | 0.79 | 0.74 | 0.44 |
| GLMC\_fechadic | 0.67 | 0.40 | 0.85 | 0.72 | 0.26 |
| KNN\_fechadic | 0.71 | 0.60 | 0.77 | 0.65 | 0.38 |
| Ranger\_fechadic | 0.71 | 0.49 | 0.85 | 0.74 | 0.35 |
| RF\_fecha | 0.64 | 0.23 | 0.91 | 0.69 | 0.16 |
| GLM\_fecha | 0.61 | 0.05 | 0.97 | 0.57 | 0.02 |
| KNN\_fecha | 0.64 | 0.33 | 0.85 | 0.65 | 0.19 |
| Ranger\_fecha | 0.63 | 0.21 | 0.91 | 0.67 | 0.13 |

3.3 Resultados de modelos datos originales dicotomizados y no dicotomizados

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Modelo** | **Accuracy** | **Sensibilidad** | **Especificidad** | **AUC** | **kappa** |
| RF\_matrixdic | 0.74 | 0.53 | 0.88 | 0.78 | 0.37 |
| GLM\_matrixdic | 0.72 | 0.53 | 0.85 | 0.78 | 0.44 |
| KNN\_matrixdic | 0.76 | 0.60 | 0.86 | 0.67 | 0.26 |
| Ranger\_matrixdic | 0.69 | 0.42 | 0.86 | 0.82 | 0.30 |
| RF\_matrix | 0.64 | 0.21 | 0.92 | 0.66 | 0.15 |
| GLM\_matrix | 0.64 | 0.12 | 0.98 | 0.65 | 0.12 |
| KNN\_matrix | 0.64 | 0.33 | 0.85 | 0.67 | 0.16 |
| Ranger\_matrix | 0.69 | 0.35 | 0.91 | 0.69 | 0.28 |

**Índice Kappa**

El **índice Kappa (κ)** mide la concordancia entre las predicciones de un modelo y las etiquetas reales, ajustando por la concordancia que podría ocurrir por azar. Es útil cuando se quiere evaluar un modelo en escenarios con clases desbalanceadas o cuando la probabilidad de adivinar correctamente es alta.

* **Rango de valores**:
  + κ = 1: Concordancia perfecta.
  + κ = 0: Concordancia igual a la esperada por azar.
  + κ < 0: Concordancia peor que la esperada por azar.
* **Ventaja**: Considera el efecto del azar, lo que lo hace más confiable que la simple precisión en conjuntos de datos desbalanceados.
* **Interpretación**:
  + κ > 0.8: Concordancia muy buena.
  + 0.6 < κ ≤ 0.8: Concordancia buena.
  + 0.4 < κ ≤ 0.6: Concordancia moderada.
  + κ ≤ 0.4: Concordancia pobre.

Aclaración: cambié las combinaciones de ranger porque no me corría (se me trabó R y tuve que reinicar):

```{r}

# Specify the tunning configuration (mtry hyperparameter depends on the number of columns)

seed.rf <- 42

set.seed(seed.rf)

x <- Train.rf\_matrix[, -ncol(Train.rf\_matrix)] # se incluyen todas las columnas excepto la última

# if(ncol(x) <= 7){

#

# mtry <- c(1, 2, 3)

#

# } else {

#

# mtry <- c(1, 2, seq(4, ncol(x) \* 0.8, 2))

#

# }

mtry <- c(1, 2, 3)

min.node.size <- seq(1, 10, 2) # Reducir el rango

hiperparametros <- expand.grid(mtry = mtry,

#min.node.size = smatrix(1, 30, 2),

min.node.size=min.node.size,

splitrule = "gini")

```

```{r}

# Seeds

seed.rf <- 42

set.seed(seed.rf)

seeds <- vector(mode = "list", length = (particiones \* repeticiones) + 1)

for (i in 1:(particiones \* repeticiones)) {

seeds[[i]] <- sample.int(500, nrow(hiperparametros))

}

seeds[[(particiones \* repeticiones) + 1]] <- sample.int(500, 1)

# Training control

```

```{r}

# Training control

cross\_val <- trainControl(

method = "repeatedcv",

number = particiones,

repeats = repeticiones,

returnResamp = "final",

verboseIter = FALSE,

allowParallel = TRUE,

classProbs = TRUE,

seeds = seeds)

# Training

# Convertir los niveles de Train.rf$sensi a números también

Train.rf\_matrix$Y <- factor(as.numeric(factor(Train.rf\_matrix$Y)))

Train.rf\_matrix$Y <- factor(Train.rf\_matrix$Y, levels = c("1", "2"))

levels(Train.rf\_matrix$Y) <- make.names(levels(Train.rf\_matrix$Y))

```

```{r}

class(Train.rf\_matrix)

```

```{r}

Train.rf\_matrix <- as.data.frame(Train.rf\_matrix)

```

```{r}

# Primero aseguramos que Train.rf es un dataframe

Train.rf\_matrix <- as.data.frame(Train.rf\_matrix)

# Convertimos Y a factor

Train.rf\_matrix$Y <- as.factor(Train.rf\_matrix$Y)

# Definimos número de árboles

#n\_trees <- 500 # default

n\_trees<-200

# Establecemos semilla para reproducibilidad

set.seed(80)

# Ejecutamos el entrenamiento

results\_matrix <- caret::train(Y ~ .,

data = Train.rf\_matrix,

method = "ranger",

tuneGrid = hiperparametros,

metric = "Accuracy",

importance = "impurity",

trControl = cross\_val,

num.trees = n\_trees)

#allowParallel=FALSE)

# Vector para probar diferentes números de árboles

#num\_trees\_range <- c(10, 50, 100, 200, 500, 1000, 1500)

```