

Tarea 1

MÓDULO VI. APRENDIZAJE NO SUPERVISADO

Integrantes:

Luis Rodrigo Santamaría Rodríguez
Jonathan Vargas González
Marisol Estrada Téllez

Equipo 8

Integrantes:

- Luis Rodrigo Santamaría Rodríguez
- Jonathan Vargas González
- Marisol Estrada Téllez

Para la presente tarea se trabajará con la siguiente base de datos llamada *water_potability.csv* la cual contiene medidas físico-químicas y evaluaciones de la calidad del agua relacionadas con su potabilidad, es decir, la idoneidad para el consumo humano. Proporciona información sobre los parámetros de la calidad del agua y nos ayudará a determinar si es potable o no.

Cada renglón representa una muestra de agua con atributos específicos y la columna *Potability* indica si el agua es o no apta para el consumo humano.

Las variables medidas son:

- ph: Nivel de ph.
- Dureza (Hardness): Dureza, medida del contenido mineral.
- Sólidos (Solids): Total de sólidos disueltos.
- Cloraminas (Chloramines): Concentración de cloraminas.
- Sulfato (Sulfate): Concentración de sulfato.
- Conductividad (Conductivity): Conductividad eléctrica.
- Carbono orgánico (Organic carbon): Contenido de carbono orgánico,
- Trihalometanos (Trihalomethanes): Concentración de trihalometano,
- Turbidez (Turbidity): Nivel de turbidez, medida de la claridad.
- Potabilidad (Potability): Indica la potabilidad con valores de 1 (potable) y 0 (no potable).

El objetivo de esta tarea es usar técnicas de reducción de dimensionalidad para facilitar su análisis, visualización y comprensión. Los métodos seleccionados para este propósito son:

- **PCA** (Análisis de Componentes Principales)
- **t-SNE** (t-distributed Stochastic Neighbor Embedding)
- **UMAP** (Uniform Manifold Approximation and Projection)

Iniciamos revisando nuestro dataset y haciendo un análisis exploratorio.

Cargamos nuestra base de datos.

```
# Cargamos la base de datos
df<-read.csv("water_potability.csv")
df$Potability <- as.factor(df$Potability)
```

Veamos la dimensión de nuestra base, la cual tiene 3276 registros y 10 columnas.

```
df %>% dim()
```

```
## [1] 3276   10
```

Checamos los tipos de datos que tenemos y algunos registros.

```
df %>% glimpse()
```

```

## Rows: 3,276
## Columns: 10
## $ ph <dbl> NA, 3.716080, 8.099124, 8.316766, 9.092223, 5.584087, ~
## $ Hardness <dbl> 204.8905, 129.4229, 224.2363, 214.3734, 181.1015, 188.~
## $ Solids <dbl> 20791.32, 18630.06, 19909.54, 22018.42, 17978.99, 2874~
## $ Chloramines <dbl> 7.300212, 6.635246, 9.275884, 8.059332, 6.546600, 7.54~
## $ Sulfate <dbl> 368.5164, NA, NA, 356.8861, 310.1357, 326.6784, 393.66~
## $ Conductivity <dbl> 564.3087, 592.8854, 418.6062, 363.2665, 398.4108, 280.~
## $ Organic_carbon <dbl> 10.379783, 15.180013, 16.868637, 18.436525, 11.558279, ~
## $ Trihalomethanes <dbl> 86.99097, 56.32908, 66.42009, 100.34167, 31.99799, 54.~
## $ Turbidity <dbl> 2.963135, 4.500656, 3.055934, 4.628771, 4.075075, 2.55~
## $ Potability <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~

```

```
df %>% head()
```

```

##      ph Hardness   Solids Chloramines   Sulfate Conductivity Organic_carbon
## 1     NA 204.8905 20791.32    7.300212 368.5164      564.3087      10.379783
## 2 3.716080 129.4229 18630.06    6.635246        NA      592.8854      15.180013
## 3 8.099124 224.2363 19909.54    9.275884        NA      418.6062      16.868637
## 4 8.316766 214.3734 22018.42    8.059332 356.8861      363.2665      18.436525
## 5 9.092223 181.1015 17978.99    6.546600 310.1357      398.4108      11.558279
## 6 5.584087 188.3133 28748.69    7.544869 326.6784      280.4679      8.399735
##   Trihalomethanes Turbidity Potability
## 1     86.99097  2.963135       0
## 2     56.32908  4.500656       0
## 3     66.42009  3.055934       0
## 4    100.34167  4.628771       0
## 5    31.99799  4.075075       0
## 6    54.91786  2.559708       0

```

Podemos observar que tenemos algunos datos faltantes de pH y Sulfatos, además, la variable “Potability” es la única variable que es de tipo factor, ya que en un principio era de tipo entero.

Revisamos cuántos datos faltantes tenemos por cada columna.

```
colSums(is.na(df))
```

```

##          ph      Hardness      Solids Chloramines      Sulfate
##        491            0            0            0            781
## Conductivity Organic_carbon Trihalomethanes      Turbidity      Potability
##            0            0           162            0            0

```

Observemos que:

- Tenemos 491 datos faltantes en pH.
- Tenemos 781 datos faltantes en Sulfatos.
- Tenemos 162 datos faltantes en Trihalometanos.

Imputaremos los datos faltantes con ayuda de la función `mice` usando el método PMM (Coincidencia de Medias Predictivas).

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes
Min.	0.000	47.432	320.943	0.352	129.000	181.484	2.200	0.738
1st Qu.	6.093	176.851	15666.690	6.127	307.707	365.734	12.066	55.729
Median	7.040	196.968	20927.834	7.130	332.615	421.885	14.218	66.617
Mean	7.076	196.369	22014.093	7.122	333.605	426.205	14.285	66.334
3rd Qu.	8.055	216.667	27332.762	8.115	359.269	481.792	16.558	77.345
Max.	14.000	323.124	61227.196	13.127	481.031	753.343	28.300	124.000

```
imputed_data <- mice(df, m = 5, method = 'pmm', maxit = 5, seed = 123)
```

Vamos a poner en nuestra variable df los datos imputados.

```
df <- complete(imputed_data, 1)
```

Comprobemos que ya no tenemos datos faltantes en nuestro dataframe.

```
colSums(is.na(df))
```

```
##          ph      Hardness      Solids Chloramines      Sulfate
##          0          0          0          0          0
##  Conductivity Organic_carbon Trihalomethanes      Turbidity Potability
##          0          0          0          0          0          0
```

Análisis exploratorio de datos

A continuación creamos una tabla que nos muestra el mínimo, primer cuantil, la mediana, media, tercer cuantil y el máximo de cada una de nuestras columnas.

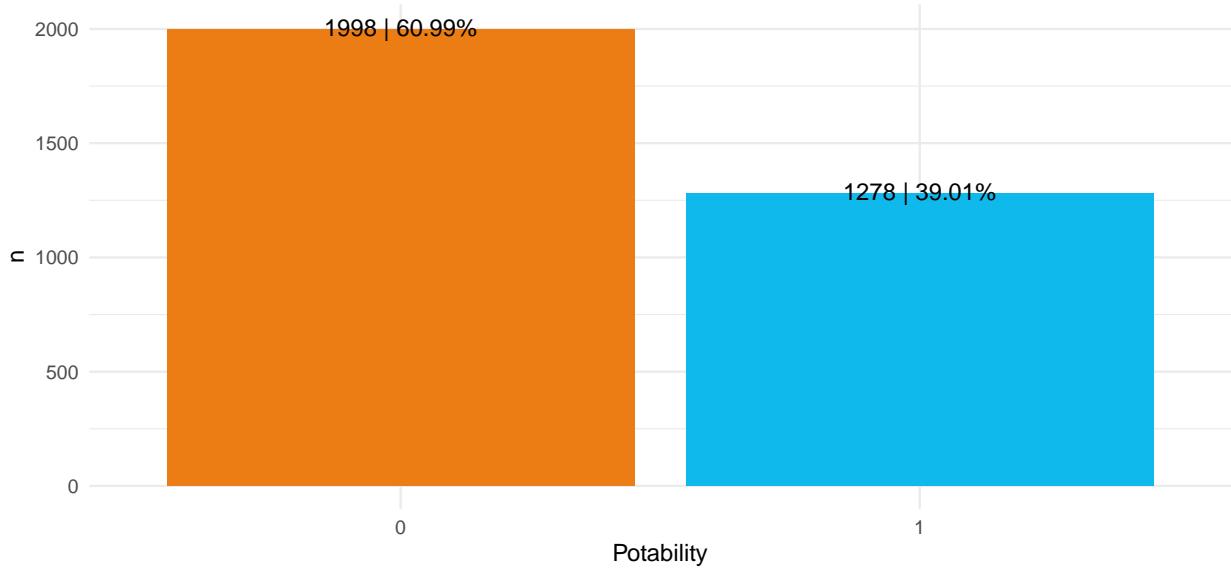
```
num.dat <- df %>% select_if(is.numeric)
# num.dat %>% glimpse()

apply(num.dat, 2, function(x) round(summary(x), 3)) %>%
  kbl() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "bordered")) %>%
  kable_paper() %>%
  scroll_box(width = "100%", height = "320px")
```

Tomamos como variable de clasificación: potability, cabe mencionar que esto lo hacemos solo de referencia ya que, en aprendizaje supervisado no tenemos una variable de salida que nos ayude a categorizar, solo lo haremos para fines ilustrativos.

Revisamos cuántos tipos de agua son adecuados para el consumo humano y cuántos no lo son.

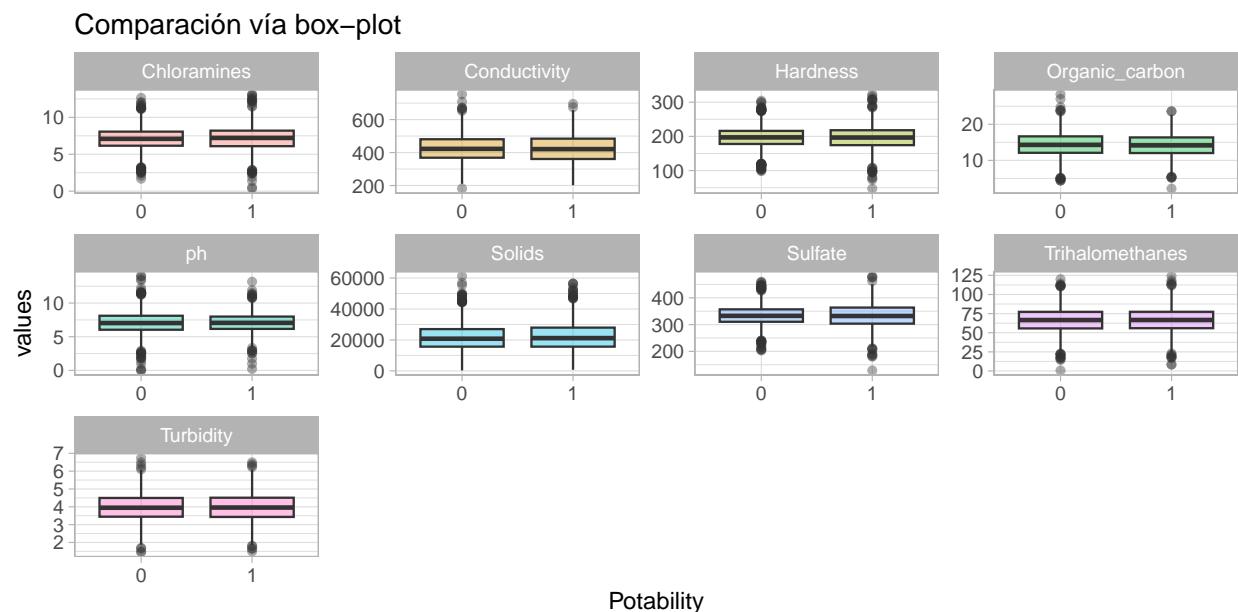
Porcentajes de agua potable y no potable



Podemos observar que hay 1998 registros que no son aptos para el consumo humano y 1278 registros sí son aptos para el consumo humano, es decir, el **60.99% de registros no son aptos para el consumo humano y un 39.01% sí lo son.**

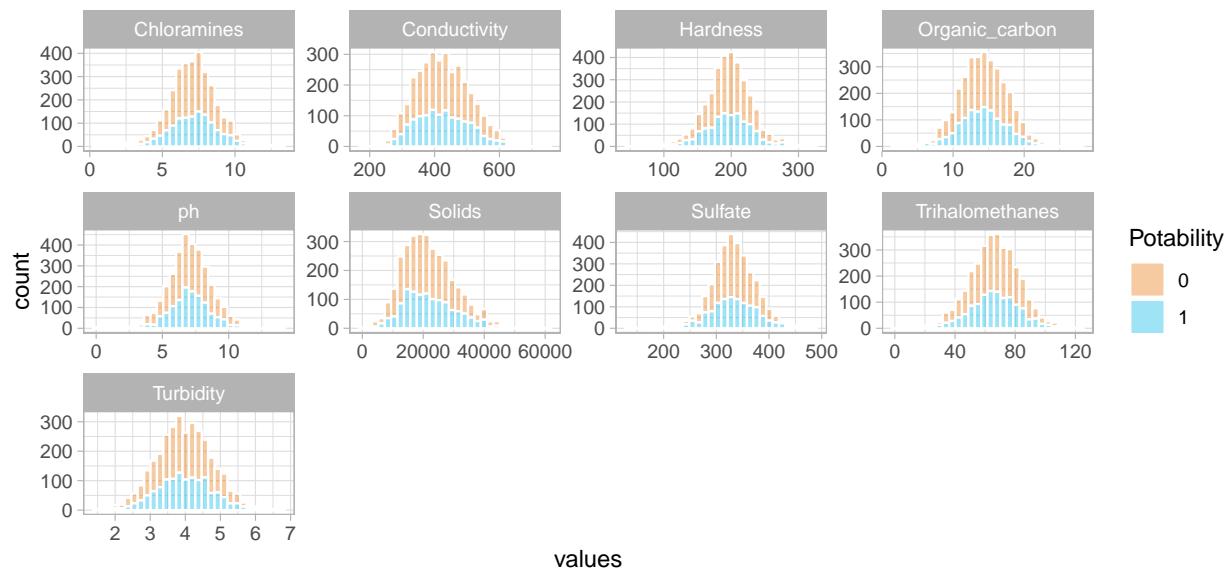
Ahora, realizamos histogramas de cada una de nuestras columnas.

Boxplots:



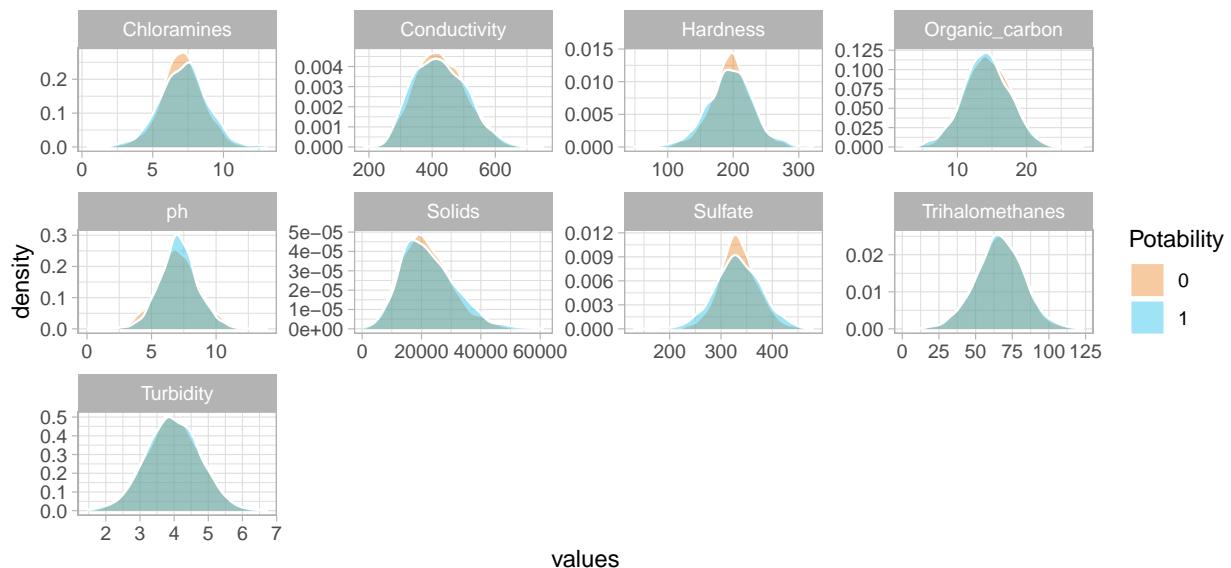
Histogramas:

Comparación vía histograma



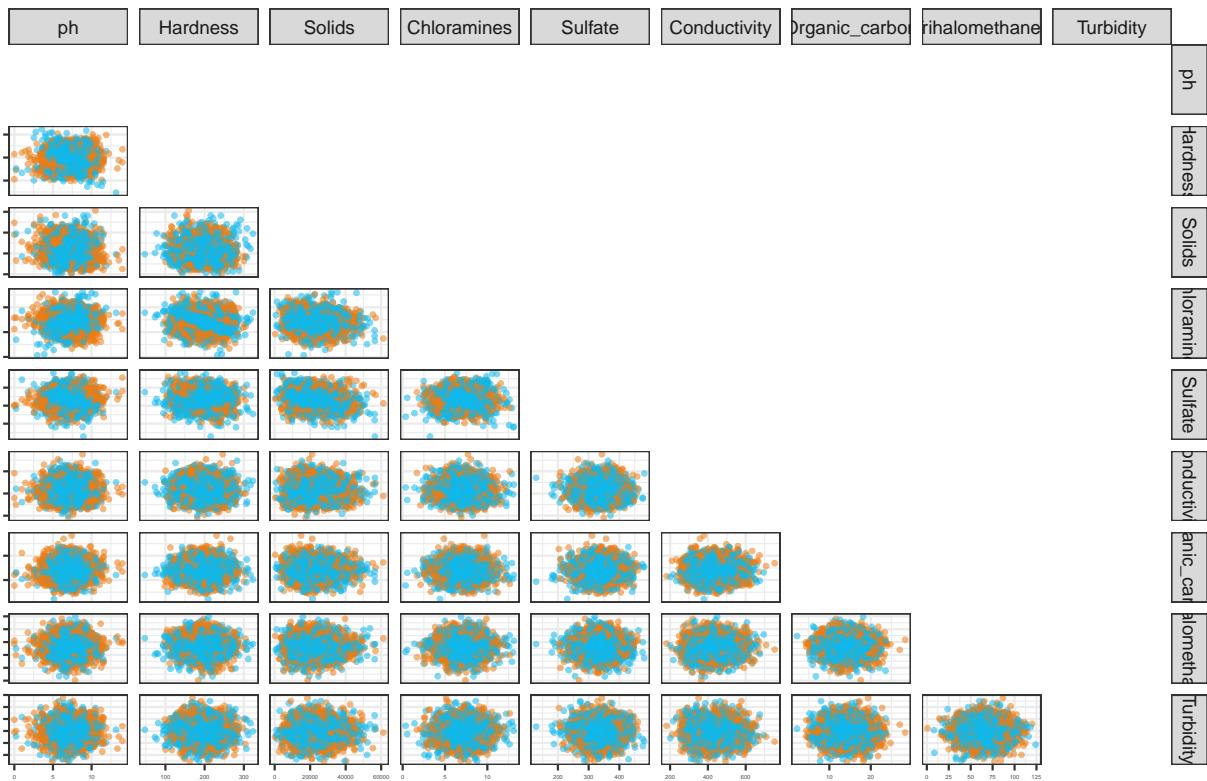
Densidades:

Comparación a través de densidad

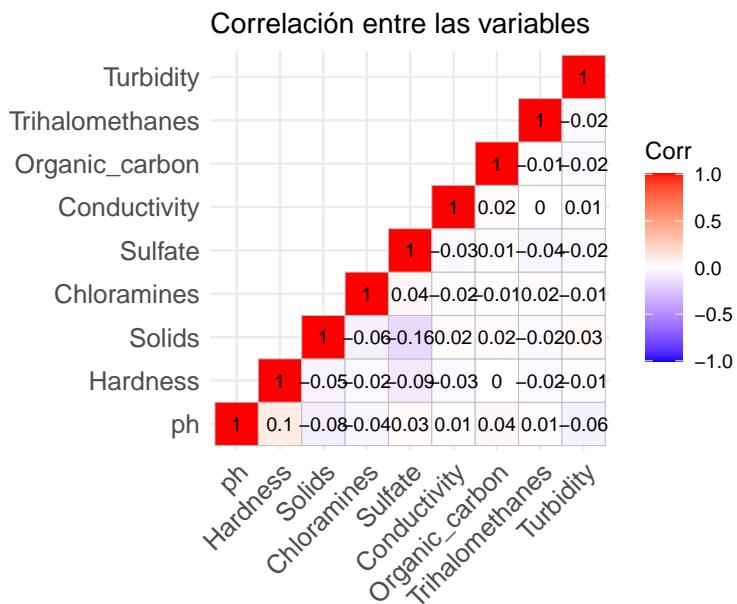


Gráficos de dispersión:

Comparación por gráficos de dispersión



Veamos la estructura de correlación de Spearman que tienen nuestras variables.



Observamos que **no hay mucha correlación en nuestras variables** por lo que podemos intuir que la **reducción de dimensionalidad no se llevará a cabo con éxito** ya que PCA, hace una reducción de dimensión mientras haya una correlación fuerte entre nuestras variables.

```

psych::KMO(cor1_data)

## Kaiser-Meyer-Olkin factor adequacy
## Call: psych::KMO(r = cor1_data)
## Overall MSA = 0.5
## MSA for each item =
##          ph      Hardness      Solids Chloramines      Sulfate
##          0.52      0.47      0.50      0.54      0.48
##  Conductivity  Organic_carbon Trihalomethanes Turbidity
##          0.49      0.50      0.43      0.56

```

Además, tomando en cuenta la medida Kaiser-Meyer-Olkin, las variables no son muy adecuadas para reducir la dimensión por estar debajo de 0.5.

PCA

Después de hacer un Análisis Exploratorio de nuestro Dataset podemos realizar la reducción de dimensión utilizando el método PCA (Componentes Principales).

Estandarizamos los datos y trataremos de reducir dimensionalidad con PCA.

```

pca_fit <- df %>%
  select(where(is.numeric)) %>%
  scale() %>% #estandarizamos los datos
  prcomp()

```

Veamos el resultado de PCA con los 9 eigenvalores:

```

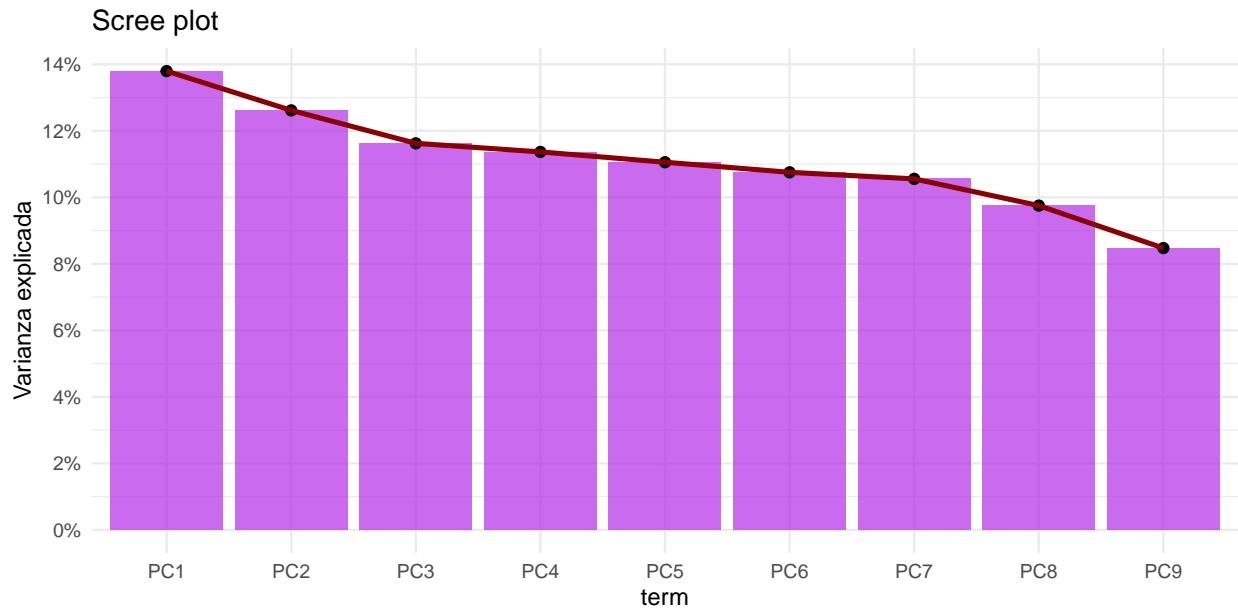
pca_fit %>%
  tidy("pcs")

##   term eigenvalues prop.variance cum.prop.variance
## 1 PC1  1.2417472    0.13797191      0.1379719
## 2 PC2  1.1354382    0.12615980      0.2641317
## 3 PC3  1.0462637    0.11625152      0.3803832
## 4 PC4  1.0230024    0.11366694      0.4940502
## 5 PC5  0.9951381    0.11057089      0.6046211
## 6 PC6  0.9678643    0.10754048      0.7121615
## 7 PC7  0.9500796    0.10556440      0.8177259
## 8 PC8  0.8777150    0.09752389      0.9152498
## 9 PC9  0.7627515    0.08475017      1.0000000

```

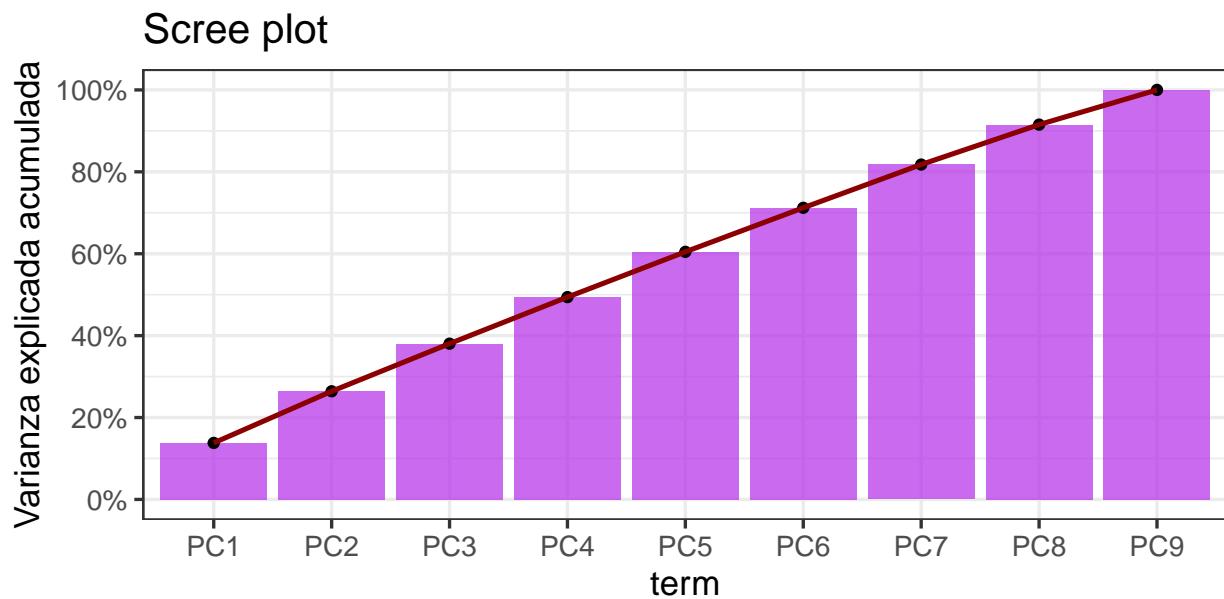
Observamos que los eigenvalores van decreciendo pero no son cercanos a cero, por lo que podemos decir que todos los eigenvalores tienen mucha varianza explicada.

Veamos una gráfica de codo la cual nos muestra como va bajando la varianza explicada por cada una de las Componentes Principales.



Cada una de las componentes tiene casi la misma varianza por lo que podemos intuir que no será posible graficar en R3 ni en R2 ya que si solo nos quedamos con 2 o 3 Componentes Principales perderemos mucha varianza de las demás componentes.

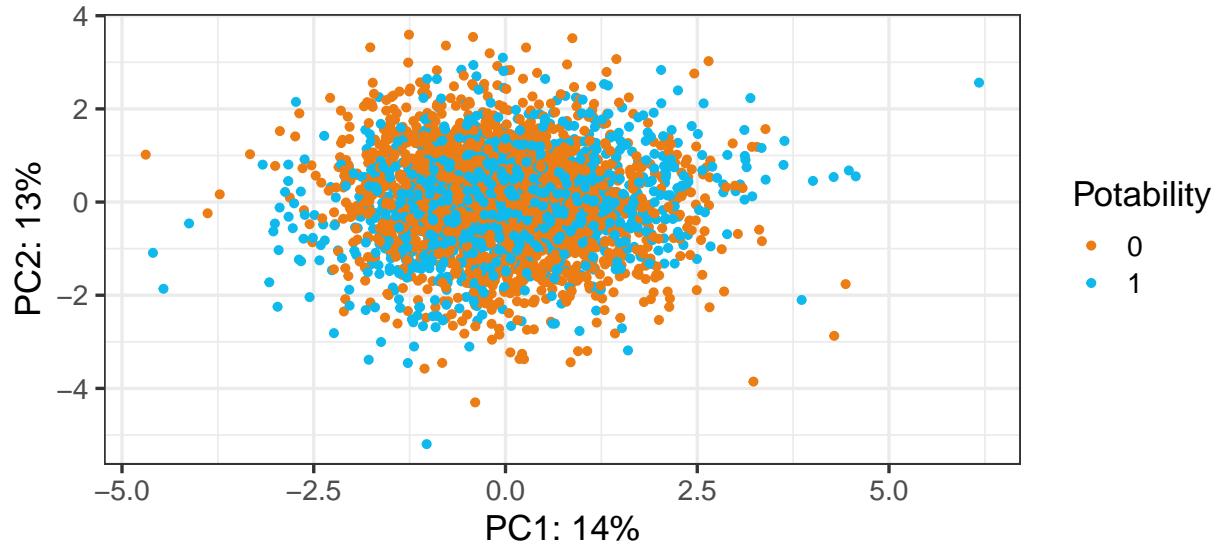
Veamos una gráfica ahora de la Varianza acumulativa.



Vamos a realizar una gráfica con los dos primeros componentes.

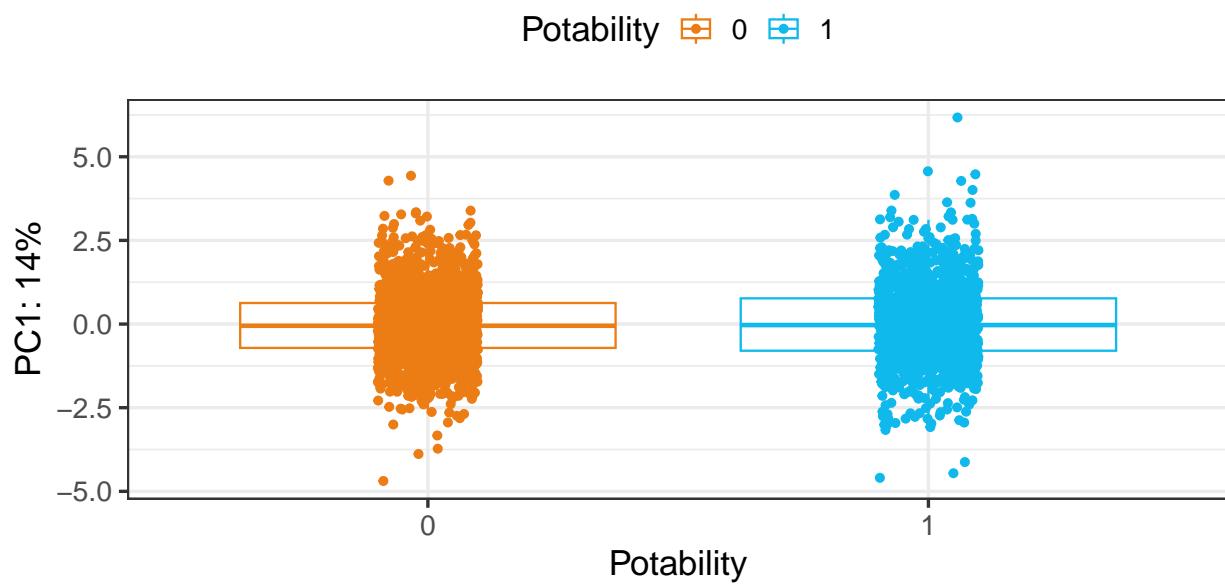
Vamos a hacer uso de nuestra variable *Potability*.

Gráfica de componentes principales

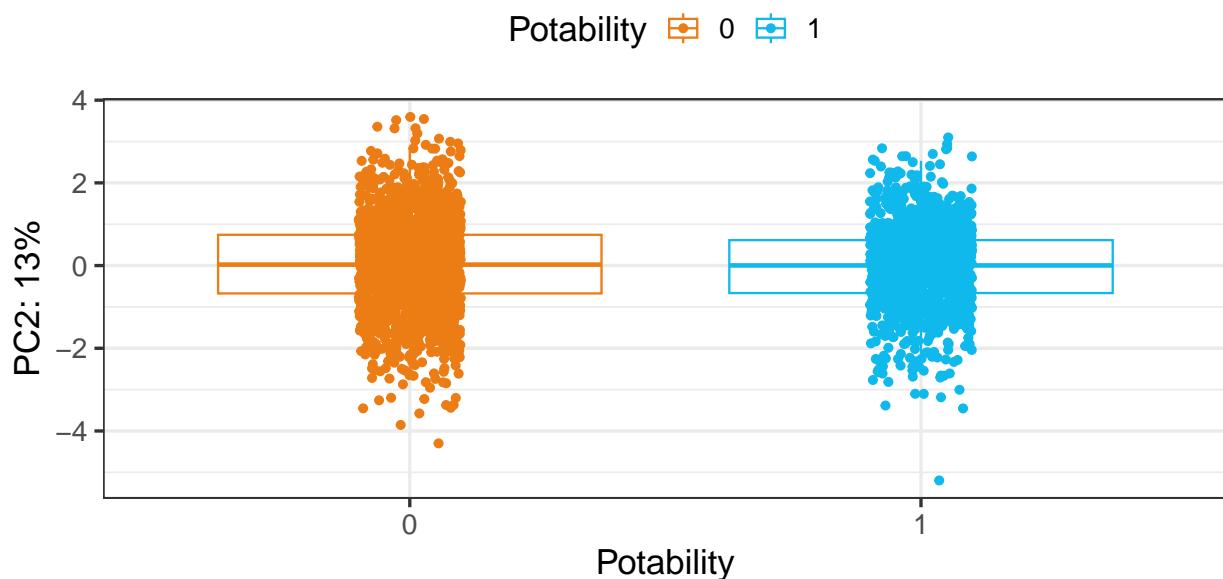


Podemos ver que no nos separan bien los datos en los grupos que tenemos de potabilidad.

Gráfica de Boxplot de la variable *Potability* con el primer componente principal.



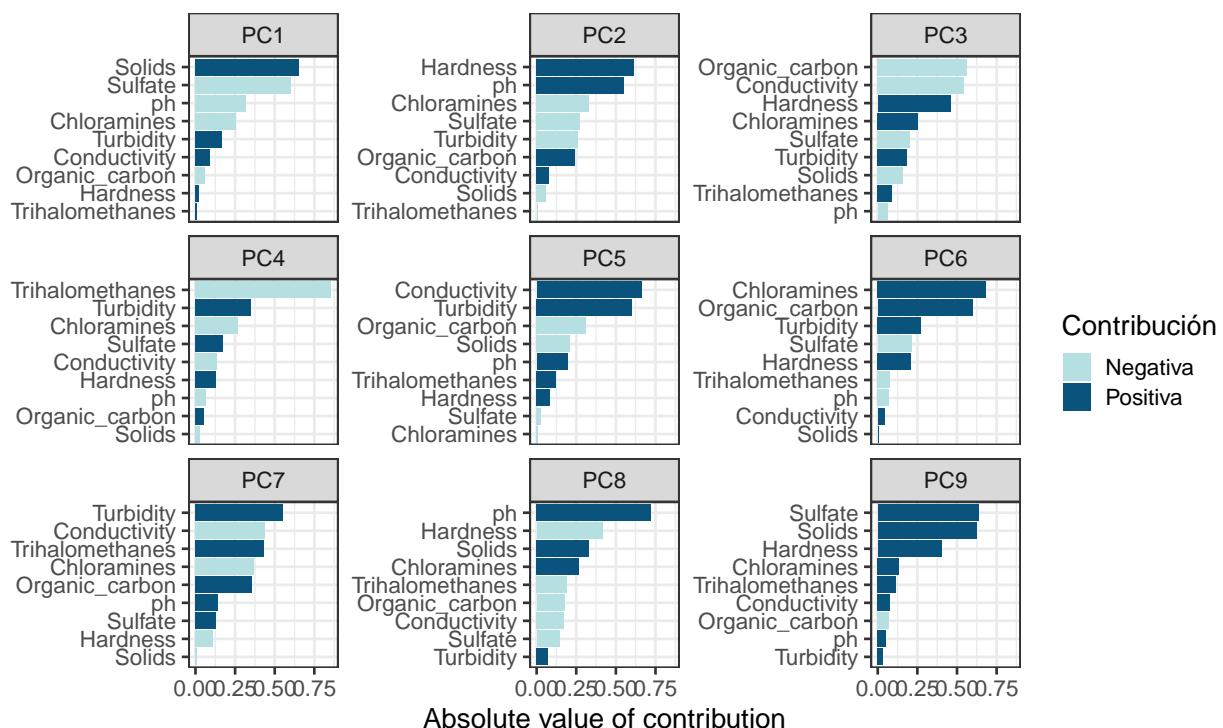
Gráfica de Boxplot de la variable *Potability* con el segundo componente principal.

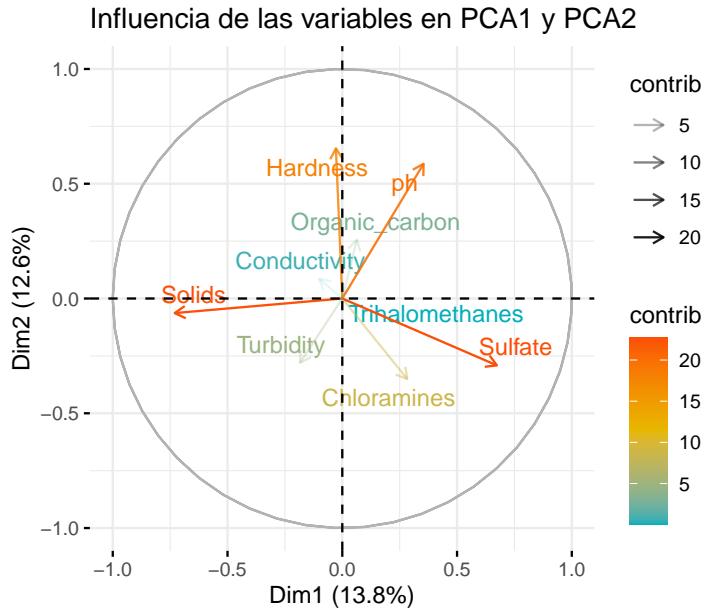


Notemos que ambas gráficas son similares los que nos indica que probablemente la diferencia en varianza explicada no es tan grande. Esto puede sugerir que ambas componentes son relevantes para entender la estructura del conjunto de datos. Pero como ya vimos anteriormente, esto ocurre también con las otras variables.

Obtenemos la carga de cada variable en el componente principal correspondiente.

Veremos la contribución de cada variable a los componentes principales: Nota: Azul oscuro para contribución positiva y azul claro para contribución negativa.





Conclusión PCA

Con este método no es posible reducir la dimensión de los datos por la baja correlación entre las variables, necesaria para aplicar PCA, y en consecuencia la alta varianza explicada por cada uno de los 9 componentes, lo que indica que no es factible reducir las dimensiones.

tSNE

A continuación vamos a hacer uso del algoritmo t-SNE con el paquete `Rtsne`.

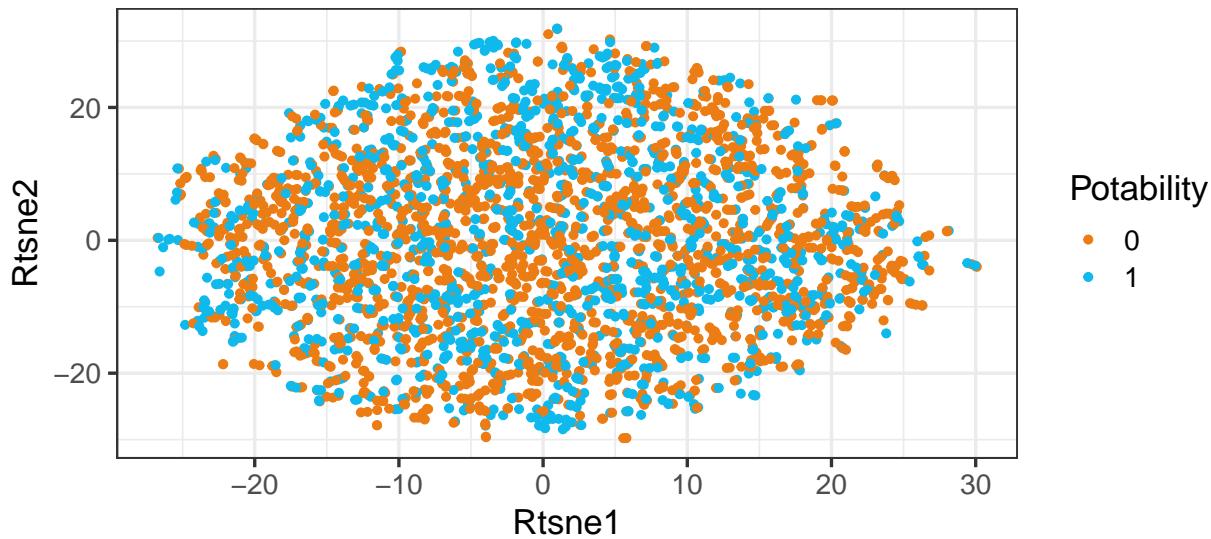
```
set.seed(123)

df_Rtsne <- df %>%
  dplyr::select(where(is.numeric)) %>%
  scale() %>%
  Rtsne()
```

Agregando los resultados de t-SNE a la base:

```
datos1<-data.frame(Rtsne1=df_Rtsne$Y[,1],Rtsne2=df_Rtsne$Y[,2],Potability=df$Potability)
```

Potabilidad: Rtsne



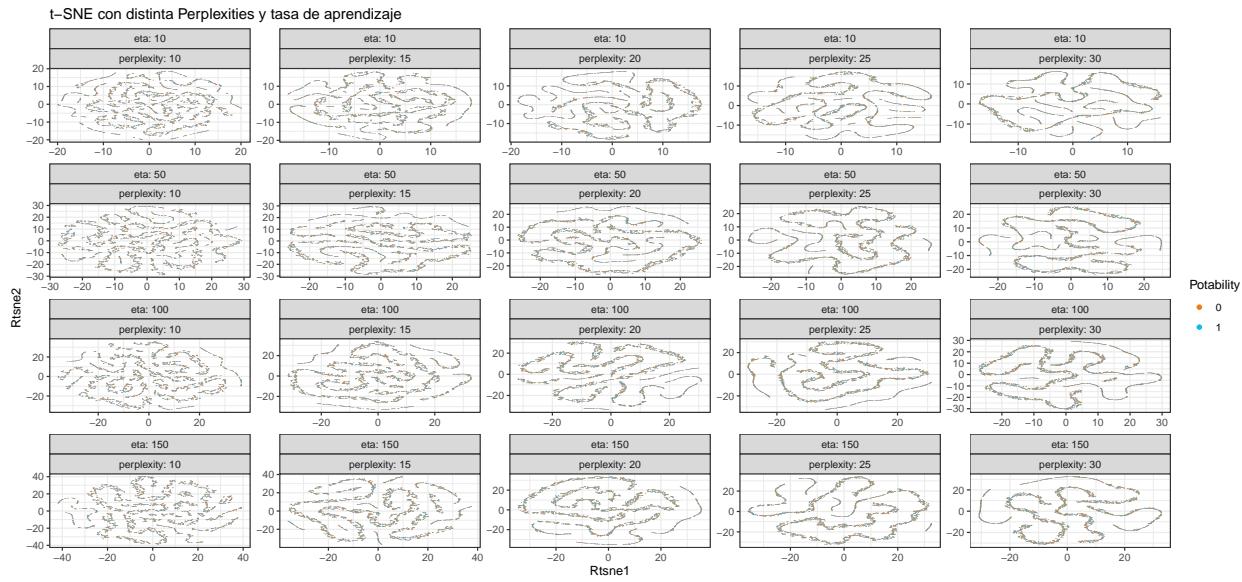
Podemos observar que aplicando t-SNE no se ve una separación de los grupos.

Veamos ahora aplicando la misma función pero modificando hiperparámetros: perplexity y la tasa de aprendizaje.

- perplexity: Con los valores de 10, 15, 20, 25, 30,
- Tasa de aprendizaje: con los valores de 10, 50, 100, 150.

```
set.seed(123)
Rtsne_params2 = expand.grid(perplexity=c(10,15,20,25,30), eta = c(10, 50, 100, 150))

Rtsne_res2 = lapply(seq(nrow(Rtsne_params2)), function(i) {
  print(i)
  Rres = Rtsne(
    X = df[,-1],
    max_iter = 500,
    verbose=TRUE,
    perplexity = Rtsne_params2$perplexity[i] ,
    check_duplicates = FALSE,
    eta = Rtsne_params2$eta[i] ,
    pca = F
  )
  return(Rres)
})
```



El gif con la animación se incluye en la carpeta de envío.

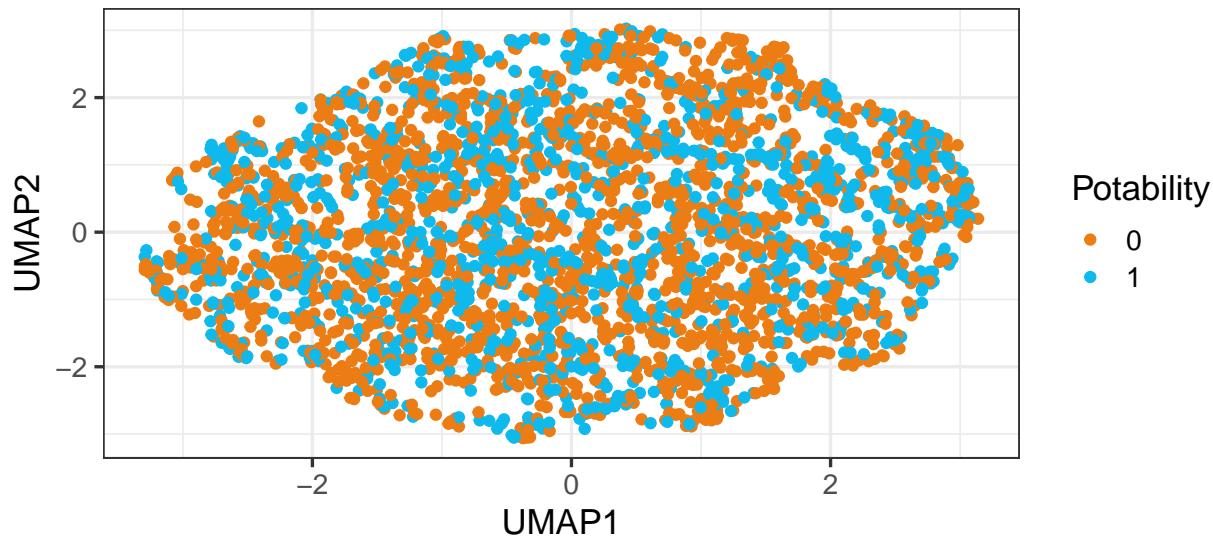
Conclusión t-SNE

En los distintos casos de hiperparámetros podemos observar que con este método de reducción de dimensiones tampoco fue posible reducir la dimensión de los datos ni separar los grupos de potabilidad.

UMAP

```
# UMAP sin valores de los hiper-parametros
df_umap <- df %>%
  dplyr::select(where(is.numeric)) %>%
  scale() %>%
  umap()
```

Potabilidad: UMAP



Argumentos de esta función:

- n_neighbors
- n_components
- metric
- n_epochs
- learning_rate

UMAP: Exploracion de

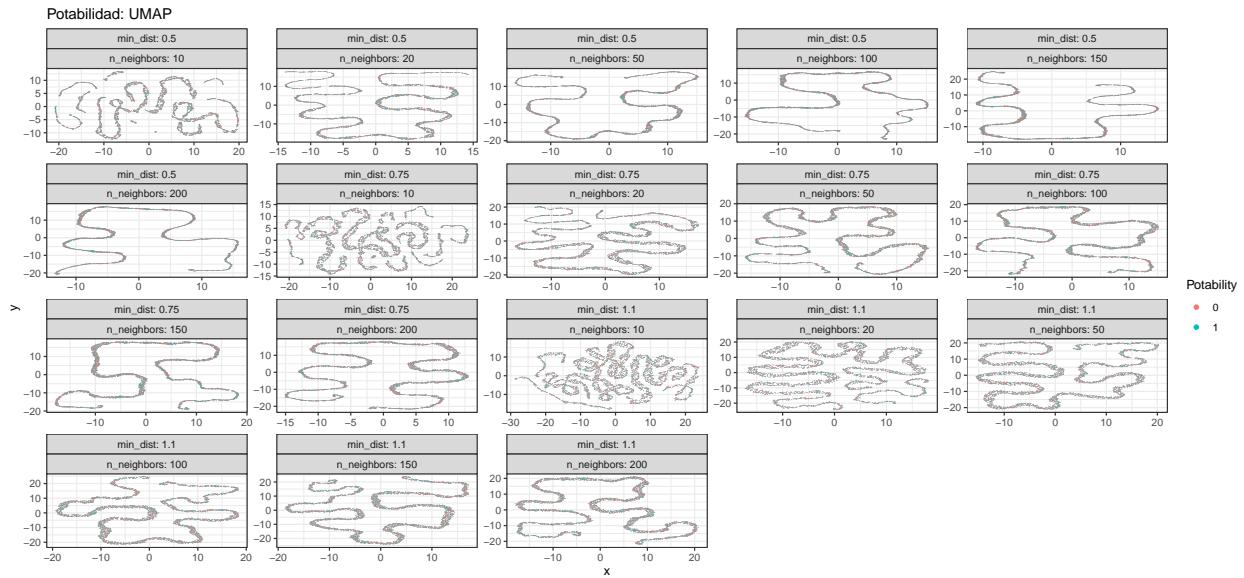
Modificaremos distintos hiperparámetros: número de vecinos y la distancia mínima.

- Número de vecinos: Con los valores de 10, 20, 50, 100, 150 y 200.
- Distancia mínima: con los valores de 0.5, 0.75 y 1.1.

```
umap_params1 = expand.grid(n_neighbors = c(10, 20, 50, 100, 150, 200), min_dist = c(0.5, 0.75, 1.1))

umaps = lapply(seq(nrow(umap_params1)), function(i) {
  emb = umap(
    X = df[,-1],
    n_neighbors = umap_params1$n_neighbors[i],
    min_dist = umap_params1$min_dist[i])

  return(emb)
})
```



El gif con la animación se incluye en la carpeta de envío.

Conclusión UMAP

En los distintos casos de hiperparámetros podemos observar que con este método de reducción de dimensiones tampoco fue posible reducir la dimensión de los datos ni separar los grupos de potabilidad.

Conclusión

En los tres métodos presentados no fue posible reducir la dimensión de los datos. Esto se puede deber a la baja correlación de las distintas variables. Esto podría ser un indicio de independencia de las variables, mostrando que cada variable aporta algo distinto e igualmente importante a la clasificación en la potabilidad.