Universidad Zaragoza
1542

Trabajo Fin de Máster

Modelos predictivos para el análisis de la mortalidad intrahospitalaria por COVID-19 en Aragón con validación entre olas epidémicas.

Autor

Daniel Vera Tena

Directores

Juan González García

Carlos Tellería Orriols

FACULTAD DE CIENCIAS

2021

RESUMEN

Este trabajo de fin de máster está dedicado a realizar un proyecto para predecir la mortalidad intrahospitalaria en Aragón debido al COVID-19. Para ello se van a utilizar los datos proporcionados desde la plataforma de uso de datos sanitarios BIGAN (orden SAN/1355/2018). El objetivo es utilizar técnicas de aprendizaje automático, entrenando a nuestro modelo con datos de la primera ola epidémica, con el fin de realizar una predicción de mortalidad intrahospitalaria por COVID-19 en las olas posteriores. El modelo no pretende limitarse a la epidemia de COVID-19, sino que puede utilizarse como marco general para otras epidemias futuras. Todo el trabajo queda englobado dentro del área del Big Data, puesto que se maneja una gran cantidad de información para validar modelos y realizar predicciones. Mediante una amplia revisión bibliográfica, se han utilizado algoritmos que nos permitan confirmar la hipótesis de que mediante estás técnicas de aprendizaje somos capaces de llevar a cabo predicciones certeras en el ámbito epidemiológico. Tras realizar una primera revisión de literatura científica, se han considerado los modelos de árboles de clasificación, *Random Forest* y *Extreme Gradient Boosting* como modelos válidos para realizar la clasificación deseada en este trabajo. Las variables a considerar serán las utilizadas en el trabajo presentado en [1] siempre que dichas variables estén disponibles en la plataforma BIGAN. El motivo de utilizar dichas variables es que son variables analíticas frecuentes, y existe evidencia en la literatura científica que sugiere que esos factores pueden ser más o menos determinantes en la gravedad de la enfermedad.

Antes de presentar los resultados se explica brevemente el funcionamiento de los algoritmos utilizados y las técnicas estadísticas empleadas. Se mide el valor predictivo de cada uno de los modelos entrenados, y posteriormente se comentan las conclusiones extraídas de todo el trabajo. Además, el proceso de tratar, filtrar y preprocesar los datos también se expone en la memoria, pues supone una parte importante y fundamental del proyecto llevado a cabo.

ABSTRACT

This master's thesis is dedicated to carrying out a project to predict in-hospital mortality in Aragon due to COVID-19. For this, the data provided from the BIGAN health data use platform (order SAN/1355/2018) will be used. The objective is to use automatic learning techniques, training our model with data from the first epidemic wave, in order to make a prediction of in-hospital mortality from COVID-19 in subsequent waves. The model is not intended to be limited to the COVID-19 epidemic, but can be used as a general framework for other future epidemics. All the work is included within the Big Data area, since a large amount of information is handled to validate models and make predictions. Through an extensive bibliographic review, algorithms have been used that allow us to confirm the hypothesis that through these learning techniques we are capable of making accurate predictions in the epidemiological field. After conducting a first review of the scientific literature, the classification tree models, Random Forest and Extreme Gradient Boosting, have been considered as valid models to perform the desired classification in this work. The variables to be considered will be those used in the work presented in [1] provided that said variables are available on the BIGAN platform. The reason for using these variables is that they are frequent analytical variables, and there is evidence in the scientific literature that suggests that these factors may be more or less decisive in the severity of the disease.

Before presenting the results, the operation of the algorithms used and the statistical techniques used are briefly explained. The predictive value of each of the trained models is measured, and the conclusions drawn from all the work are then discussed. In addition, the process of treating, filtering and pre-processing the data is also exposed in memory, as it is an important and fundamental part of the project carried out.

Regarding the structure of the work, it has been divided into several blocks. Chapter 2 explains the operation of the algorithms and the techniques used to measure the goodness of the results, also the packages and libraries used for our purposes. In section 3.1 we explain what data we are going to use, their meaning, how the classification algorithms will be used and we will comment on the central hypothesis of the work.

Section 3.2 explains how the entire task of data exploration and preprocessing has been carried out, as well as the associated problems. In chapter 4 the results obtained with the different techniques are presented and tables and graphs are presented that give us a visual interpretation of what was obtained. In chapter 5 the results of the algorithms are compared as a whole and the conclusions and interpretations that can be drawn from the project carried out are discussed. The last part of the work is dedicated to concluding if the classification methods used are effective, with the variables used. It is important to note that this work is particularly relevant when compared to [1], as it is a more elaborate multidisciplinary work but with variables in common.

Índice

Capítulo 1

Introducción y metodología.

1.1. Introducción.

Desde la aparición de la pandemia provocada por el coronavirus SARS-Cov-2, una gran atención mediática y científica se ha dirigido hacia el estudio de la propagación del virus, así como de las consecuencias derivadas de la COVID, la enfermedad causada por dicho virus, ya sean económicas, sociales o de otra índole. En este trabajo vamos a desarrollar varios modelos de aprendizaje automático basados en diferentes algoritmos de clasificación, muy conocidos dentro del campo de conocimiento del *machine learning*. El objetivo del trabajo es analizar y predecir la mortalidad intrahospitalaria por COVID-19 en la Comunidad Autónoma de Aragón a partir de un amplio conjunto de variables clínicas y de laboratorio. Estos datos han sido obtenidos, previa solicitud de acceso y compromiso de confidencialidad, desde la plataforma de datos sanitarios BIGAN. BIGAN es la plataforma de Big Data del Departamento de Sanidad del Gobierno de Aragón, gestionada por el Instituto Aragonés de Ciencias de la Salud (IACS), que recopila de forma sistemática y pseudonimizada multitud de datos clínicos de los pacientes del Servicio Aragonés de Salud para su utilización en proyectos de investigación y política sanitaria.

La hipótesis básica de trabajo es que resulta posible entrenar un modelo de clasificación a partir de los datos de una primera ola epidemiológica, y que dicho modelo es capaz de predecir con un alto grado de precisión la mortalidad (o supervivencia) de los pacientes ingresados por COVID-19 en las sucesivas olas epidemiológicas. En el caso de que la precisión de los modelos vaya decayendo en las sucesivas olas, el análisis de los resultados podría ofrecernos indicios para saber por qué el modelo va perdiendo capacidad predictiva, y así poder mejorar el modelo de una forma dinámica.

1.2. Metodología.

Para llevar a cabo el trabajo se ha utilizado el software estadístico R, donde tenemos una vasta colección de funciones y librerías para lograr nuestro fin. Las muestras que utilizamos están desequilibradas, ya que el número de pacientes que sobreviven es, afortunadamente, mucho mayor que el número de fallecidos. Esto significa que una de las categorías de la variable respuesta utilizada en nuestros algoritmos está más representada que la otra. Esta problemática unida al tratamiento de los datos faltantes son los principales obstáculos a salvar en el trabajo.

Para el desarrollo y entrenamiento de los distintos modelos predictivos, se utilizaron los siguientes algoritmos:

- Árboles de clasificación, un método de clasificación en el cual aparecen nodos que se dividen según distintos criterios estadísticos hasta llegar a una clasificacíon final de los individuos, que son etiquetados como "Sobrevive" o "Fallece".
- *Random Forest*, basado en árboles de clasificación que se combinan para dar resultados muy precisos a la hora de etiquetar individuos de una muestra.
- *Extreme Gradient Boosting*, un método de clasifición basado en el algoritmo *Gradient Boosting* pero con una implementación que lo hace muy preciso, y también basado en árboles de clasificación.

1.3. Organización de la memoria.

En cuanto a la estructura del trabajo, se ha dividido en varios bloques. En el capítulo 2 se explica el funcionamiento de los algoritmos y las técnicas utilizadas para medir la bondad de los resultados, así como los paquetes y librerías utilizadas para nuestros fines. En la sección 3.1 se explica qué datos vamos a utilizar, su significado, cómo se utilizarán los algoritmos de clasificación y comentaremos la hipótesis central del trabajo. En la sección 3.2 se explica cómo se ha llevado a cabo toda la tarea de la exploración de los datos y su preprocesado, también la problemática asociada. En el capítulo 4 se exponen los resultados obtenidos con las diferentes técnicas y se presentan las tablas y gráficas que nos dan una interpretación visual de lo obtenido. En el capítulo 5 se comparan los resultados de los algoritmos en conjunto y se comentan las conclusiones e interpretaciones que se pueden extraer del proyecto llevado a cabo. La última parte del trabajo está dedicada a concluir si son eficaces los métodos de clasificación empleados, con las variables utilizadas.

Es importante reseñar que este trabajo cobra una especial relevancia comparándolo con [1], pues es un trabajo multidisciplinar más elaborado pero con variables en común.

Capítulo 2

Marco de trabajo.

2.1. Algoritmos.

2.1.1. Arbol de clasificación

Es un modelo predictivo cuyo objetivo es separar los individuos de un conjunto según el valor de una variable respuesta. Se explica con detalle en [11]. Este algoritmo es similar a un diagrama de flujo, donde se utiliza una regla para dividir cada conjunto de individuos. Esta regla se basa en escoger para la división la variable que mejor separa cada una de las clases de la variable objetivo. Cada regla en el diagrama corresponde a un nodo, donde se particiona en dos el conjunto. La división continúa de manera recursiva hasta que se llega al nodo terminal u hoja, donde ya no se divide más. Una vez que ha utilizado una variable para separar los datos, esta variable no se vuelve a utilizar en los nodos hijos respectivos. El modelo es simple y visual, motivo por el que se ha utilizado, intentando ser complementario a otros estudios como [1], que utilizan modelos de regresión logística, aquí no utilizados.

Es importante para crear nuestros modelos no utilizar un gran número de nodos, pues se puede sobreajustar el modelo y así obtener un descripción que no sirva para un conjunto de individuos cualquiera. El árbol tiene el objetivo de que sus nodos finales sean los más homogéneos posibles. En nuestro caso es un árbol binario, pues en cada división se forman dos conjuntos nuevos. En cada división, el árbol debe decidir qué variable utiliza para segmentar y qué valor utiliza como umbral. El criterio de partición es la impureza del nodo, de tal forma que los nodos hijos tienen que ser más puros que el nodo padre. La impureza de un nodo t es una función que definimos según la probabilidad de que una clase j se encuentre en el nodo t ($p(j/t)$ o p_j/t) como:

$$i(t) = \begin{cases} \text{máx} & \text{si } p_j/t = \text{cte } \forall j \\ 0 & \text{si } p_j/t = 0 \ \forall j \neq k \ p_k/t = 1 \end{cases}$$

En cada división se produce un decremento de impureza dado por:

$$\Delta i(t) = i(t) - \frac{n_{tl}}{n_t} i(t_l) - \frac{n_{tr}}{n_t} i(t_r) \tag{2.1}$$

donde t_l y t_r son los nodos hijos a izquierda y derecha, y n_{tl} y n_{tr} los individuos de cada nodo hijo izquierdo y derecho respectivamente. Este decremento hay que hacerlo con todas las variables. Así en cada partición se almacena la información del resto de particiones que no son la óptima, y se mide la importancia de cada variable en cada división. Es importante cuando tenemos datos faltantes. La medida de la impureza se puede hacer según varios criterios, en nuestros árboles se utiliza el índice Gini, dado por la siguiente expresión

$$i(t) = \sum_{i \neq j} p(j/t) p(i/t) \tag{2.2}$$

En este trabajo utilizamos la librería `rpart`, muy utilizada en `R` para construir árboles de clsificación, tal cómo se muestra en [4]. Aquí exponemos un ejemplo del código utilizado en R:

```
arbol <- rpart( exitus_bl ~ ., data = entrenamiento, method="class",
                control=list(  minsplit=10,
                               minbucket=2,
                               maxdepth=10,
                               xval=10
                               ))
```

donde cada uno de los parámetros tiene la siguiente función:

- *minsplit*: número mínimo de observaciones para que un nodo pueda ser dividido.
- *minbucket*: número mínimo de observaciones en un nodo terminal.
- *maxdepht*: número máximo de nodos que pueden existir en el árbol, contando el nodo raíz como profundidad 0.
- *xval*: número de validaciones cruzadas.

2.1.2. Random Forest

Es un algoritmo que utiliza distintos árboles de clasificación de forma que es un método de aprendizaje conjunto, con el fin de clasificar individuos. El algoritmo puede entenderse más detalladamente en [5]. Para entender este algoritmo es necesario conocer los árboles de clasificación descritos en 2.1.1 pues está basado

en la combinación de muchos de ellos. Cada uno de los árboles es un clasificador débil, que intentamos que arroje resultados diferentes y no todos similares pues no tendría sentido combinarlos. En general un solo árbol tiene una varianza alta, que se solucionaría aumentando la cantidad de datos de entrenamiento. Como nuestro conjunto de datos es limitado, se combinan en el Random Forest muchos de estos árboles de clasificación para crear un clasificador fuerte. Los árboles individuales son sencillos y se trabaja conjuntanmente con técnicas de *bagging* y aleatorización. La aleatorización es tanto en individuos como en variables. Los individuos que no se utilizan en cada remuestra, el algoritmo los utiliza como conjunto de validación, esto es lo que se conoce como *Out Of Bag* (OOB).

En cuanto a las técnicas de *bagging*, podemos referirnos a [9] para una explicación más profunda. Resumidamente, los predictores de bagging son un método para generar múltiples versiones de un predictor y usarlas para obtener un predictor agregado. La agregación promedia sobre las versiones al predecir un resultado numérico, y hace un voto de pluralidad al predecir una clase. Las múltiples versiones se forman haciendo réplicas *bootstrap* del conjunto de aprendizaje y utilizándolas como nuevos conjuntos de aprendizaje. Pruebas en conjuntos de datos reales y simulados utilizando clasificación, árboles de regresión y la selección de subconjuntos en regresión lineal muestran que el *bagging* puede proporcionar ganancias sustanciales en precisión. El elemento vital es la inestabilidad del método de predicción. Si perturbar el conjunto de aprendizaje puede causar cambios en el predictor construido, el *bagging* puede mejorar la precisión.

Se ha utilizado la librería `randomForest` de `R`, que nos perimite utilizar el *Random Forest* y escoger el óptimo para nuestro proyecto.

2.1.3. Extreme Gradient Boosting

Se puede consultar en profundidad el funcionamiento del algoritmo en [6]. *Extreme Gradient Boosting* es un algoritmo de aprendizaje supervisado que se fundamenta en el principio de *boosting*, es decir, en construir una secuencia de clasificadores "débiles" que tengan en cuenta los resultados del resto de clasificadores empleados, todo esto recogido en una tasa de aprendizaje a la cual se le pueden asignar valores arbitrarios. El algoritmo intenta minimizar una función objetivo; si el modelo nuevo tiene mejores resultados se selecciona para continuar con el aprendizaje, si no, se regresa al modelo anterior y se modifican variables distintas. Se lleva a cabo el proceso hasta que las diferencias entre esas funciones correspondientes a distintos modelos en tan pequeña

que no merece la pena continuar mejorándolo, o cuando el número de iteraciones llegue a un máximo permitido. Se utiliza este algoritmo para intentar mejorar los resultados del árbol de clasificación simple, pues serán aquí varios los utilizados y contaremos con una tasa de aprendizaje que penaliza los aspectos menos relevantes.

Para esquematizar como funciona un algoritmo de gradiente descendente se muestra la figura 2.1. El *Extreme Gradient Boosting* es una implementación de este método mostrado.

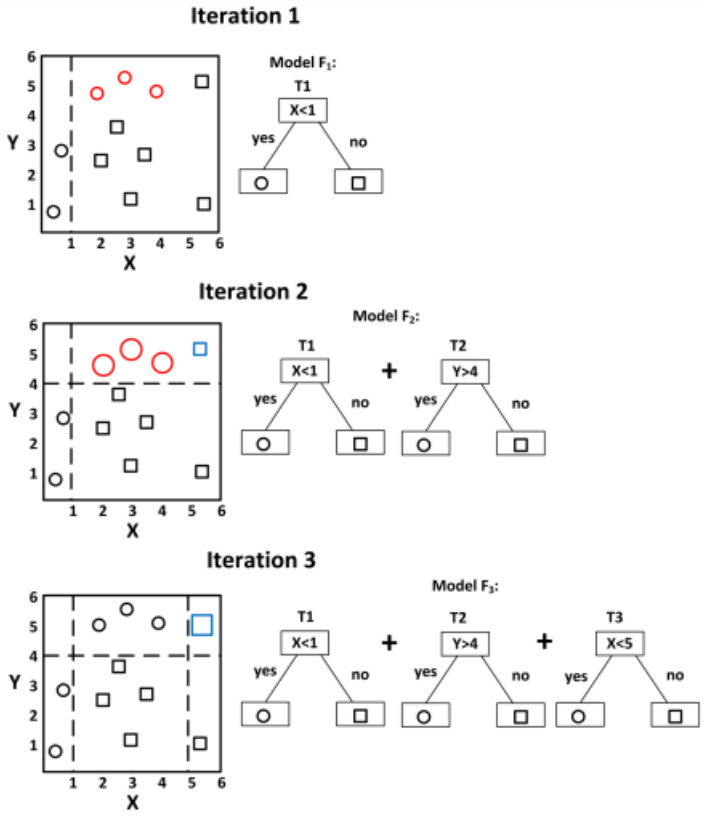


Figura 2.1: Esquema del algoritmo *Gradient Boosting*, obtenido de [10].

Se ha utilizado la librería `xgboost` de `R`.

2.2. Métricas de bondad.

2.2.1. Matriz de confusión.

Antes de avanzar en el trabajo y explicar someramente el fundamento de los algoritmos utilizados para clasificar, así como los resultados obtenidos, vamos a analizar el método del que disponemos para evaluar la calidad de nuestros resultados. Se utiliza la matriz de confusión, que sirve para detallar la calidad de la clasificación de las

predicciones con clases binarias, es decir, una clase “Positivo” y otra clase “Negativo”. Debido a que hay dos posibles valores reales y dos posibles valores de predicción, a partir de estas opciones podemos crear lo que se conoce como la matriz de confusión con 4 resultados posibles:

- **Verdadero positivo**: El valor real es positivo y la prueba predice un positivo.
- **Verdadero negativo**: El valor real es negativo y la prueba predice un positivo.
- **Falso negativo**: El valor real es positivo, y la prueba predice un negativo. También se denomina error tipo 2 en estadística.
- **Falso positivo**: El valor real es negativo, y la prueba predice un positivo. También se denomina error de tipo 1 en estadística.

El esquema de la matriz de confusión es el de la figura 2.2.

Figura 2.2: Matriz de Confusión.

Las métricas que empleamos son las siguientes:

- **Exactitud**: Es la proporción de verdaderos positivos (VP) y verdaderos negativos (VN) con los casos totales. Analíticamente sería lo siguiente:

$$\frac{VP + VN}{VP + FP + FN + VN} \tag{2.3}$$

- **Sensibilidad**: Es la tasa de verdaderos positivos. Se puede expresar como:

$$\frac{VP}{FN + VP} \tag{2.4}$$

– **Especificidad**: Es la tasa de verdaderos negativos. Es decir:

$$\frac{VN}{FP + VN} \tag{2.5}$$

– **Valor Predictivo Negativo**: Es la capacidad de clasificar correctamente la clase negativa, en nuestro caso los fallecimientos.

$$\frac{VN}{FN + VN} \tag{2.6}$$

– **Valor Predictivo Positivo**: Es la capacidad de clasificar correctamente la clase positiva, en nuestro caso la supervivencia.

$$\frac{VP}{FP + VP} \tag{2.7}$$

Para nuestro propósito, son especialmente relevantes el valor predictivo negativo y la especificidad, pues es crucial el clasificar correctamente a los pacientes que fallecen, sin importar tanto los que sobreviven. Otra métrica que aparece en `R` es el *No Information Rate*, que es un valor numérico que viene a expresar la exactitud que tendría el método si clasificáramos indistintamente a todos los individuos como clase mayoritaria. En nuestros datos, debido a que los pacientes que sobreviven son mayoritarios, el valor de este parámetro es elevado.

2.2.2. Curvas ROC

Estas curvas [12] representan la sensibilidad frente al complementario de la especificidad (1− especificidad), es decir, la tasa de verdaderos positivos en función de la tasa de falsos postivos. Si un clasificador (probabilístico) tiene una curva ROC por encima de otro, es un mejor clasificador. Estas curvas se modifican según varía el umbral de clasificación, que es un valor según el cual discriminamos entre clase positiva o negativa. El valor AUC (área bajo la curva) nos indica el grado de fiabilidad que tiene un clasificador de discernir entre la dos categorías, siendo un valor igual a 1 para una clasificación perfecta, y un valor igual a 0,5 para un modelo que no presenta ninguna capacidad diagnóstica o clasificativa. Se muestra en la figura 2.3 una ilustración de la curva ROC.

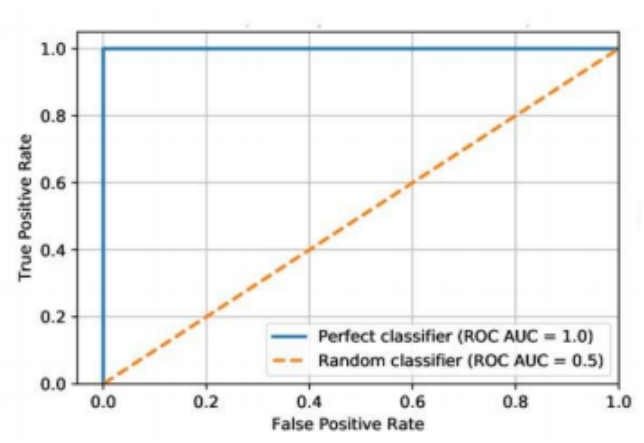


Figura 2.3: Ejemplo de curva ROC. Se ha obtenido de la asignatura de *Introducción a la Minería de Datos* del Máster Modelización e Investigación Matemática, Estadística y Computación.

Capítulo 3

Tratamiento de datos.

En este capítulo se trata el contenido central del trabajo, se explica el significado de los datos que se han utilizado y el modo en que funcionan los algoritmos para dar los resultados.

3.1. Explicación de los datos.

Los datos se han extraído de la plataforma de datos sanitarios BIGAN, organizados según una tabla de registro de los pacientes, con su referencia identitaria, fechas relevantes y una variable que refleja si el paciente sobrevive o fallece; y una tabla con las variables clínicas de los pacientes. Se han dividido las olas epidémicas según la tabla 3.1. Utilizamos las variables clínicas, mostradas en la tabla 3.2, junto con la edad y el sexo del paciente, para clasificar. En las tablas 3.3 y 3.4 se explican las columnas de las tablas extraídas de BIGAN: la tabla de registros o casos y la tabla de laboratorio.

	Comienzo	Final	Pacientes ingresados
Primera ola	03/03/2020	07/06/2020	680
Segunda ola	08/06/2020	23/08/2020	962
Tercera ola	24/08/2020	13/12/2020	3416
Cuarta ola	14/12/2020	21/03/2021	2434
Quinta ola	22/03/2021	10/06/2021	1408

Tabla 3.1: División de olas epidémicas llevadas a cabo en este trabajo.

Las olas epidémicas se dividen siguiendo las curvas de ingresos y fallecidos en Aragón, estableciendo los límites entre olas en los días de mínima incidencia (cambio de tendencia). Este criterio es el mismo que utilizó el Instituto Aragonés de Ciencias de la Salud en sus diferentes análisis de la pandemia. A fecha de 10/06/2021 cuenta como

la fecha de muerte del último paciente en el conjunto de datos obtenido de BIGAN.

Variable
Alanina aminotransferasa (ALT)(U/L)
Tiempo de tromboplastina parcial activado (aPTT)(seg)
Aspartato aminotransferasa (AST) (U/L)
Basófilos en sangre ($10^3/\mu L$)
Bilirrubina directa (mg/dL)
Bilirrubina total (mg/dL)
Proteína C Reactiva (PCR)(mg/L)
Linfocitos CD3 (%)
Creatinina(mg/dL)
Eosinófilos en sangre ($10^3/\mu L$)
Eritrocitos en sangre ($10^6/\mu L$)
Fibrinógeno (mg/dL)
Glucosa en sangre (mg/dL)
Hematocritos (%)
INR, prueba de tiempo de protrombina ({INR})
Leucocitos en sangre ($10^3/\mu L$)
Linfocitos en sangre ($10^3/\mu L$)
Hemoglobina corpuscular media (MCH) (pg)
Concentración de hemoglobina corpuscular media (MCHC) (g/dL)
Volumen corpuscular medio (MCV) (fL)
Monocitos en sangre ($10^3/\mu L$)
Plaquetas en sangre ($10^3/\mu L$)
Potasio en sangre(mmol/L)
Tiempo de protrombina(PT) (seg)
Sodio en sangre (mmol/L)

Tabla 3.2: Variables clínicas que utilizamos en este trabajo.

Las variables de la tabla 3.2 están expresadas con sus unidades entre paréntesis. En el Anexo B se exponen las gráficas de las distribuciones de cada variable. Muchas de las variables corresponden a parámetros hematológicos, procedentes de los hemogramas de los pacientes, aunque también hay determinaciones correspondientes a bioquímica básica. En la literatura científica aparecen muchas de estás variables, por ejemplo en [2] y [3], por lo que el objetivo de este trabajo está dirigido en cierta manera para validar la influencia de estos factores.

3.2. Detalles y preprocesado de los datos.

Lo primero que se ha llevado a cabo en este trabajo es un preprocesado de los datos, es decir, se ha hecho una depuración de éstos, se ha visto a qué tipo de dato corresponde cada variable (numérica, categórica, fecha...), y también otras tareas

como valorar los datos faltantes. Una vez llevada a cabo esta tarea, se hizo un primer análisis detallado de cada variable, analizando sus valores estadísticos y construyendo representaciones gráficas.

En nuestro estudio, disponemos de un conjunto de datos que se pueden dividir en otros dos subconjuntos: el subconjunto de casos y el subconjunto de datos de laboratorio. En el primero, disponemos de 8969 observaciones, cada una correspondiente a cada individuo hospitalizado. Para cada individuo se nos proporciona información sobre su edad, sexo, fecha de ingreso hospitalaria, una variable binaria indicando la supervivencia o el fallecimiento, fecha de primeros síntomas, fecha de confirmación de la enfermedad y fecha de fallecimiento, es decir la tabla de "casos", con cada columna explicada en la tabla 3.3. En el segundo conjunto disponemos de las variables clínicas de cada paciente, es decir, la tabla de "laboratorio", con sus columnas explicadas en la tabla 3.4. Con los datos proporcionados finalmente para llevar a cabo el trabajo, disponemos de un total de 25 variables incluyendo variables clínicas y hematológicas. Con cada paciente, están relacionadas una o más variables clínicas, así como su valor cuantitativo.

Para hacer la exploración de los datos, se ha utilizado el paquete de R `DataExplorer`, una herramienta muy útil para visualizar datos faltantes y resúmenes gráficos de cada una de las columnas. En la figura 3.1 se muestra la cantidad de datos faltantes para cada columna.

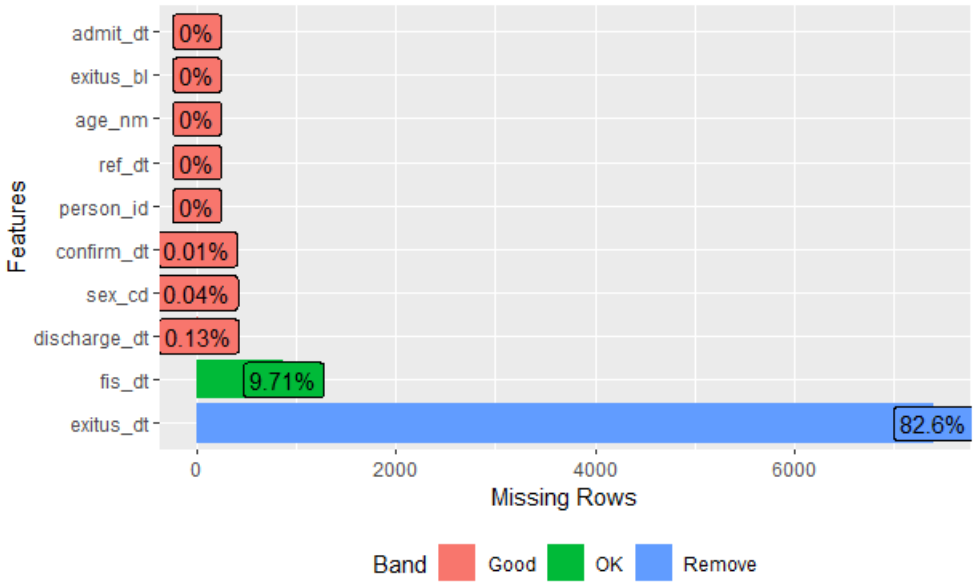


Figura 3.1: Exploración de los datos de la tabla de casos.

Variable	Significado
admit_dt	Fecha de ingreso hospitalario
exitus_bl	Variable categórica binaria: 0 (sobrevive), 1 (fallece)
age_nm	Edad del paciente
ref_dt	Fecha de referencia
person_id	Identidad del paciente
confirm_dt	Fecha de confirmación
sex_cd	Sexo del paciente: 1(femenino), 2(masculino)
discharge_dt	Fecha de fallecimiento
fis_dt	Fecha de inicio de los síntomas
exitus_dt	Fecha de fallecimiento

Tabla 3.3: Significado de cada columna de la tabla de casos.

La variable *ref_dt* no tiene mucho significado en nuestro estudio. Para cada paciente, interesa la más temprana fecha entre las variables *ref_dt* y *confirm_dt*. La variable *exitus_dt* tiene sólo algún valor cuando el paciente haya fallecido, por eso hay una gran cantidad de datos faltantes. En la variable *fis_dt*, tenemos alrededor de un 10 % de datos faltantes, ya que no en todos los casos se averiguó cuando comenzaron los primeros sintomas. En la tabla 3.4 se muestra el contenido de la tabla de laboratorio.

Variable	Significado
std_observable_st	Nombre de la variable clínica
unit_st	Unidad en la que está expresada la variable clínica
std_observable_cd	Código característico de la variable clínica
person_id	Identidad del paciente
obs_value_nm	Valor observado de la variable
request_dt	Fecha de petición de la analítica

Tabla 3.4: Significado de cada columna de la tabla de laboratorio.

Para el conjunto de datos de laboratorio, tenemos las siguientes columnas y su porcentaje de valores omitidos, que se muestran en la figura 3.2.

Cada paciente en el conjunto de datos de laboratorio puede aparecer en múltiples ocasiones, cada una de ellas con analíticas de variables diferentes. El conjunto total de datos tiene por lo tanto 173609 registros, correspondientes a 8969 pacientes, siendo cada registro alguno referente a las variables de la figura 3.2. Algunas variables están altamente correlacionadas, y por tanto no aportan información adicional sobre el paciente. Por ejemplo, la variable *INR* tiene un significado similar al *Tiempo de protrombina*, aunque ambas medidas están referidas en unidades diferentes.

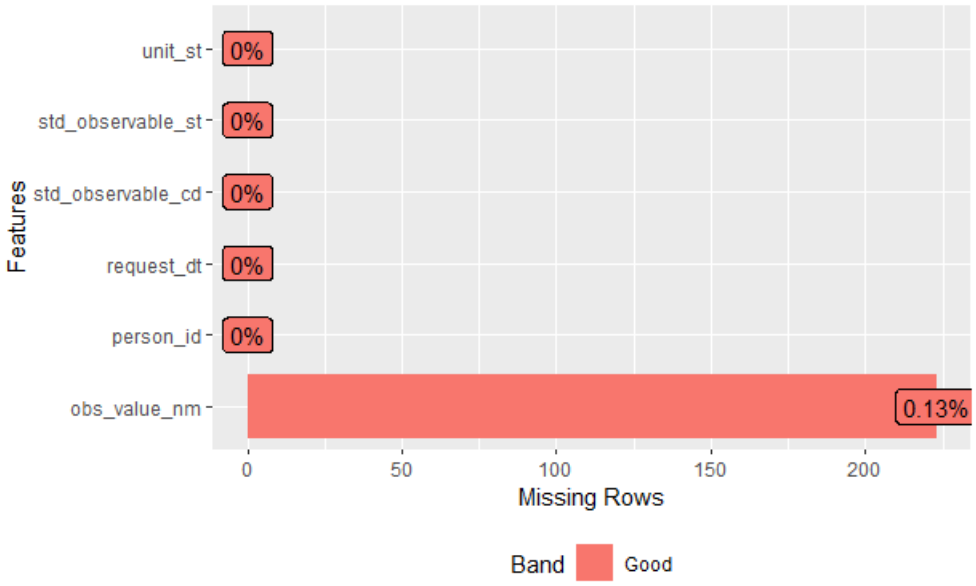


Figura 3.2: Exploración del conjunto de datos del laboratorio.

Se ha utilizado la función `dcast` de `R` para transformar una tabla larga en una tabla ancha. Es decir, se ha construido una tabla con tantas filas igual al número de pacientes, de tal forma que las sucesivas columnas representan las variables clínicas de cada paciente así como la variable respuesta de supervivencia, que es la variable objetivo de éste trabajo y puede tomar dos valores: '0' y '1'. En la figura 3.3 se muestra el número de ingresos hospitalarios, y en la figura 3.4 se muestran los fallecidos. Se pueden distinguir 5 olas epidémicas a simple vista.

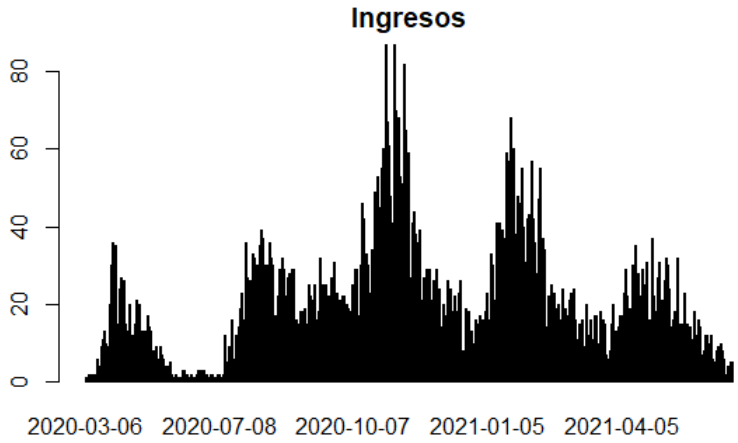


Figura 3.3: Ingresos hospitalarios en Aragón.

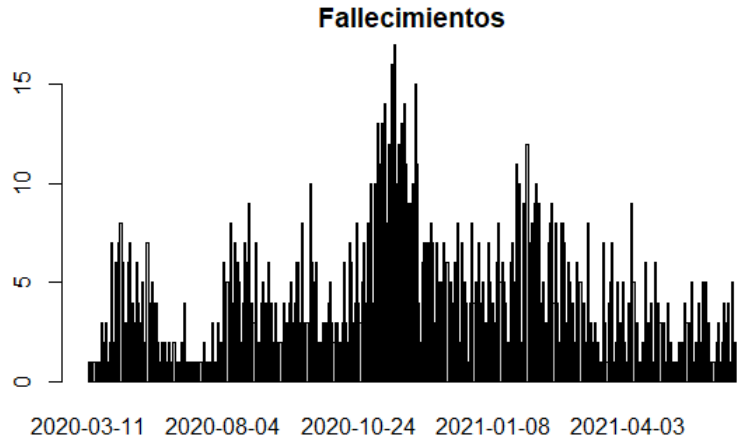


Figura 3.4: Fallecimientos hospitalarios en Aragón.

El número de datos faltantes o *missings* es de gran relevancia, ya que si tenemos una variable con muchos valores omitidos, estropea el método de clasificación, distorsiona los resultados, o directamente impide la utilización del algoritmo. Por lo tanto, una tarea fundamental es eliminar esas variables que tengan un gran número de valores perdidos. Existen 2 variables (CD3 y Bilirrubina_directa) que tienen más de un 90 % de valores perdidos, tal cómo se indica en la figura 3.5; estas variables no se incluyen en los modelos. El resto de variables, incluyendo las catalogadas como *bad* son utilizadas en el trabajo.

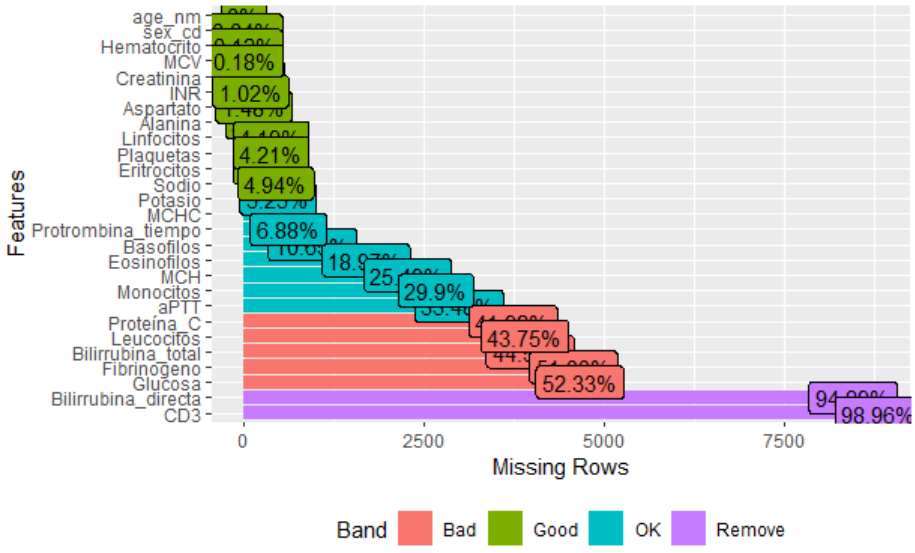


Figura 3.5: Valores perdidos o *missings*.

Cada variable tiene una distribución, que se puede observar nítidamente en histogramas. Un aspecto a considerar para la correcta visualización es la presencia de *outliers*, es decir, valores que están muy alejados del resto, ya sea por un caso atípico o por un fallo de recolección de los datos, que nos impide tener una representación ajustada de los valores. Eliminamos estos *outliers* y estudiamos cada distribución, como se muestra a modo de ejemplo en la figura 3.6. La presencia de ese valor discrepante en el 0 nos hace darnos cuenta de que hay valores que realmente son nulos, y se han recogido con valor cero (0). Estos datos han sido eliminados y tratados como valor faltante. Se hace lo mismo con el resto de variables.

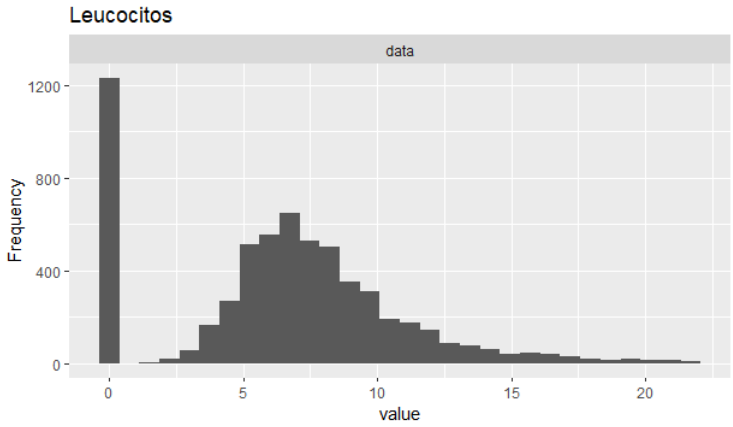


Figura 3.6: Distribución de la variable de leucocitos en sangre.

En aquellos casos en los que el algoritmo a utilizar no sea tolerante con datos faltantes (Random Forest), ha sido necesario realizar una imputación de valor a dichos registros vacíos, tal como se explica más adelante. Este mecanismo de imputación nos permite entrenar y validar esos modelos, aunque resta capacidad de comparación de la bondad de distintos algoritmos, ya que en última instancia utilizan conjuntos de datos distintos. Además, el resultado final dependerá del mecanismo de imputación de datos nulos que se utilice, y en muchos casos requerirá de una interpretación clínica de esas ausencias. No es lo mismo un dato faltante porque la determinación concreta solo se solicita a pacientes con determinadas características, o porque solo se registran los datos patológicos (fuera de rango de normalidad), y se dejan a nulo los normales, o si el dato falta por un error en la captura o el registro de la información.

Vamos a estudiar cual es la importancia relativa de las variables a la hora de clasificarlas, utilizando una función del paquete `caret`, llamada `filterVarImp`. Esta función se basa en el criterio AUC discutido en 2.2.2. De esta forma vemos que las

variables más importantes al clasificar con un árbol son la edad, los linfocitos en sangre, el INR y el PT, cómo se aprecia en la figura 3.8.

Vemos en la figura 3.7 que en las variables de INR y PT existe una correlación, por lo tanto a la hora de clasificar, utilizar las 2 variables a la vez no tiene mucho sentido. En el resto de gráficas se aprecia un nube homogénea de puntos, lo que nos indica que no hay una gran correlación entre ellas.

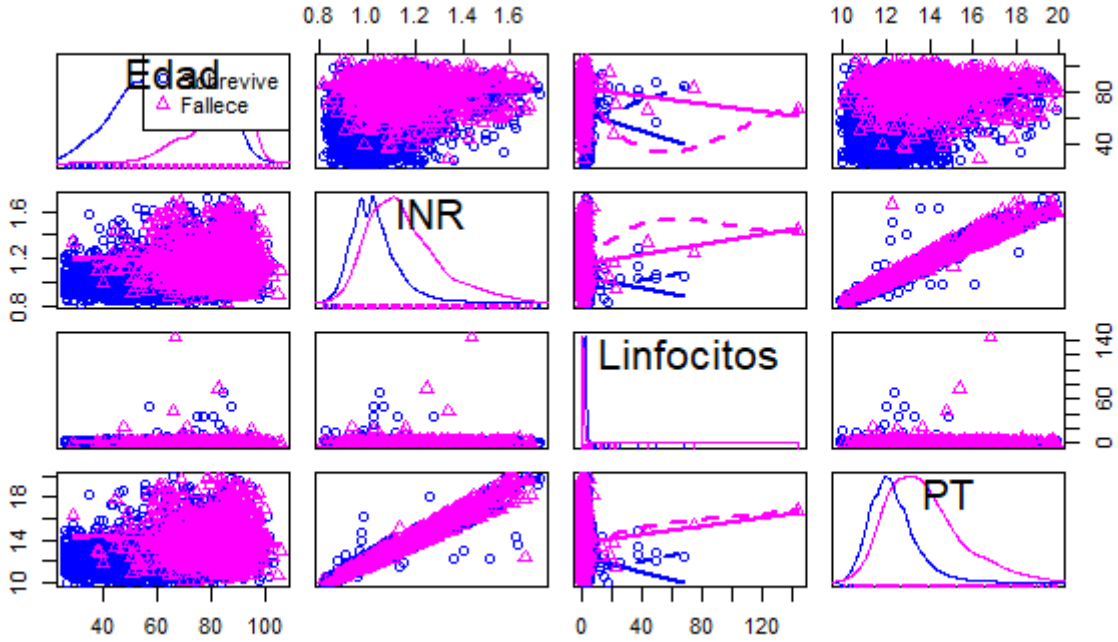


Figura 3.7: Correlación de las variables más importantes para clasificar.

Cada ola tiene unas características distintas, una es la mortalidad, que se recoge en la tabla 3.5:

	Supervivencia	Mortalidad
Primera ola	71.62 %	28.38 %
Segunda ola	79.73 %	20.27 %
Tercera ola	79.30 %	20.70 %
Cuarta ola	84.14 %	15.86 %
Quinta ola	95.31 %	4.69 %

Tabla 3.5: Mortalidad en cada ola epidémica.

En cuanto a la importancia de las variables según el crirterio AUC utilizando la función `filterVarImp`, se observan diferencias entre la primera ola (principio de la

pandemia) y la quinta ola (figuras 3.8 y 3.9). Como veremos más adelante, en la quinta ola, la campaña de vacunación comienza en Aragón y este efecto empieza a ser notable en cuanto a la mortalidad y en cuanto a las características del paciente ingresado.

aPTT	Potasio	MCH
0.5001436	0.5124427	0.5199540
Monocitos	Eosinofilos	Fibrinogeno
0.5255184	0.5465364	0.5488504
Plaquetas	Leucocitos	Proteína_C
0.5510794	0.5580162	0.5612505
Alanina	Bilirrubina_total	Sodio
0.5782841	0.5784224	0.5795342
Glucosa	Basofilos	Eritrocitos
0.5973604	0.6155749	0.6258152
MCV	Hematocrito	MCHC
0.6283580	0.6340075	0.6581854
Creatinina	Aspartato	Protrombina_tiempo
0.6607654	0.6914279	0.7303306
INR	Linfocitos	age_nm
0.7403049	0.7548116	0.7892458

Figura 3.8: Importancia de las variables en la primera ola.

Basofilos	Bilirrubina_total	Creatinina
0.5083548	0.5187023	0.5225240
Fibrinogeno	Sodio	Glucosa
0.5235458	0.5271926	0.5273280
Potasio	Monocitos	aPTT
0.5365522	0.5436650	0.5456070
MCH	Proteína_C	Leucocitos
0.5507666	0.5681818	0.5726245
Alanina	Eosinofilos	Aspartato
0.5781793	0.5959841	0.5985244
MCHC	MCV	Plaquetas
0.6456104	0.6467337	0.6573804
INR	Protrombina_tiempo	Linfocitos
0.7317267	0.7537597	0.7670709
Hematocrito	Eritrocitos	age_nm
0.7747934	0.7767974	0.8017827

Figura 3.9: Importancia de las variables en la quinta ola.

Se observa cómo según el criterio AUC la edad, los linfocitos, el INR y el tiempo de protrombina son las variables más importantes en la primera ola epidémica. Los eritrocitos y el hematocrito toman un papel más relevante a la hora de clasificar, mientras que la edad sigue siendo el factor más determinante en la quinta ola epidémica.

Estos cambios observados en las tablas son comentados más adelante. Para finalizar el tratamiento previo de los datos, es importante considerar que estamos trabajando

con dos clases desequilibradas, pues el número de fallecidos oscila en torno al 80 % de la población estudiada, y el número de pacientes que sobreviven oscila en torno al 20 %. Se hace uso de la función de `caret`: `upSample`. Para describirla brevemente, es una función que lleva a cabo un remuestreo con reemplazamiento hasta que la clase minoritaria esté equilibrada con la clase mayoritaria. Son importantes estas muestras sobremuestreadas para los entrenamientos de los algoritmos, pero no utilizamos las muestras equilibradas como conjunto test, pues ya no sería un descripción real de la población. Se explica con más detalle en [8].

Capítulo 4

Resultados

Una vez se ha realizado la tarea del preprocesado de los datos y su depuración, se lleva a cabo el análisis de los resultados para expresarlos correctamente. Los algoritmos utilizados son los explicados en 2.1.1, 2.1.2 y 2.1.3, que expresan la evaluación de los resultados así como matrices de confusión y otros valores estadísticos. Cada algoritmo lo tratamos por separado y al final del capítulo los comparamos conjuntamente. Utilizamos los datos de la primera ola epidémica para construir el árbol, y medimos su calidad con los datos de las siguientes olas. Es importante destacar, para evitar confusiones, que en este estudio la clase positiva es la supervivencia, y la clase negativa es el fallecimiento.

Los algoritmos que utilizamos se han aplicado de la siguiente manera: primero se construyen modelos a partir de un conjunto de datos de entrenamiento, extraídos de la primera ola epidémica; a continuación se utilizan el resto de olas epidémicas como conjunto de validación, obteniendo una serie de métricas de la bondad con la que se clasifican los pacientes en las sucesivas olas. En el caso de que el modelo vaya perdiendo capacidad predictiva a medida que va testándose con las sucesivas olas epidemiológicas, intentamos encontrar por qué ocurre, y tener así elementos para una mejora del modelo. Es posible, igualmente, realizar variaciones sobre el algoritmo de entrenamiento-validación, por ejemplo entrenando un modelo con las dos primeras olas juntas, y validar contra las siguientes, y así sucesivamente. Por cada uno de los procesos de validación de un modelo, obtenemos matrices de confusión, explicadas en el apartado 2.2.1, que nos exponen la capacidad de predecir clases positivas y negativas de los modelos. Empezamos utilizando un algoritmo sencillo como el árbol de clasificación, y después algoritmos más complejos basados en éste, como el *Random Forest* y el *Extreme Gradient Boosting*. En principio, sería esperable que los resultados de los modelos más complejos sean más precisos, pero podría no ser así. De hecho, en las conclusiones finales veremos algunos casos en los que, en efecto, modelos más

sencillos se comportan mejor que otros más complejos. El entrenamiento de los distintos modelos y su validación con los datos de las distintas olas epidemiológicas se ha realizado utilizando el software estadístico `R`, pues dispone de gran variedad de librerías y funciones enfocadas hacia el aprendizaje automático, los métodos de clasificación y la estadística.

4.1. Clasificación con un solo árbol.

Empezamos construyendo nuestro modelo con el algoritmo más simple de los tres que vamos a utilizar, pero también el más ilustrativo y gráfico. Se utilizó la función `rpart` para crear nuestro árbol, que necesita un conjunto de datos que sirva como entrenamiento y otro como conjunto de validación. Se ha creado una partición con un tamaño del 90 % para el entrenamiento y un 10 % para la validación, y a su vez se ha dividido el conjunto de entrenamiento en 10 subconjuntos, con los que se ha llevado a cabo el proceso de validación cruzada. El proceso de validación cruzada es necesario ya que debido a la variación que podemos tener en el diseño del árbol según escojamos un conjunto de entrenamiento u otro, tenemos que particionar el conjunto en n bloques y seleccionar cada uno de ellos para validar mientras el resto se utilizan de entrenamiento. El proceso se repite n veces y se obtiene el modelo final con el promedio de estas iteraciones y su error. Se ha utilizado un árbol donde entran todas las variables descritas en la figura 3.5 menos las células CD3 y la bilirrubina directa debido a la gran cantidad de datos faltantes. En nuestro árbol creado con `rpart` se ha dado a los parámetros los siguientes valores, explicados en la sección 2.1.1: $minsplit = 10$, $minbucket = 2$, $maxdepth = 10$ y $xval = 10$. Tenemos que escoger el árbol óptimo, nos fijamos en el parámetro de complejidad para seleccionarlo (figura 4.1).

Se escoge el que tenga un menor error, en este caso es el de tamaño 11, pero dado que el de tamaño 6 y **cp** = 0.026 tiene un error a menos de una desviación típica del mínimo, se selecciona éste. Se muestra en la figura 4.2 la representación del árbol escogido con la función `rpart.plot()`.

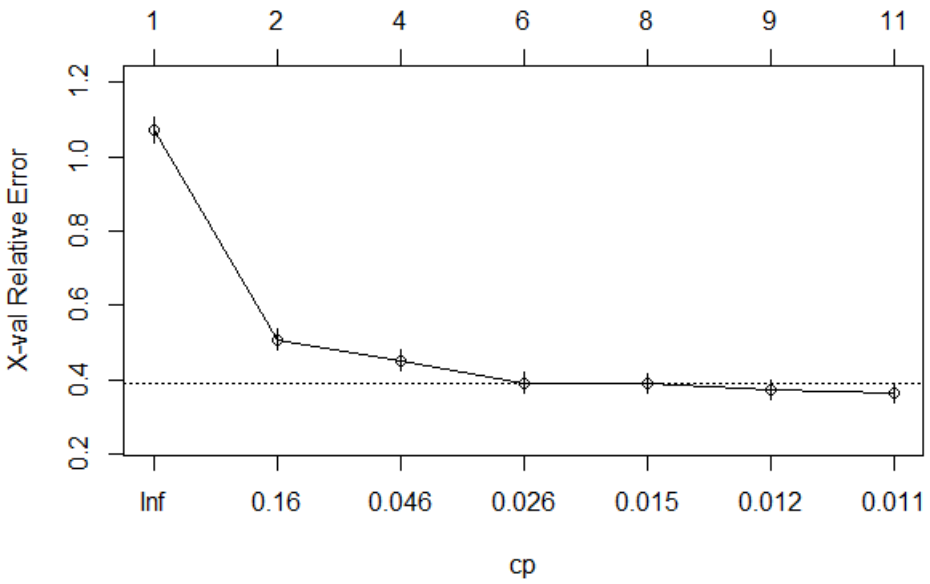


Figura 4.1: Representación del error en el árbol de clasificación en función del parámetro de complejidad (cp) y de su tamaño (eje horizontal superior). Las líneas verticales indican la desviación estándar en el error de cada árbol, siendo la línea de puntos el valor del error del árbol de mayor tamaño más la desviación estándar correspondiente a dicho árbol.

Comprobamos la calidad del modelo con el conjunto de validación que supone un 10 % del tamaño del conjunto de la primera ola. Se muestra en la tabla 4.1 la matriz de confusión:

	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	36	12
Fallece (pred)	2	46

Exactitud	0.8542
Sesibilidad	0.9474
Especifidad	0.7931
Val.Pred.Pos	0.7500
Val.Pred.Neg	0.9583

Tabla 4.1: Resultados obtenidos con el conjunto de validación, basado en un 10 % de los datos de la primera ola epidémica.

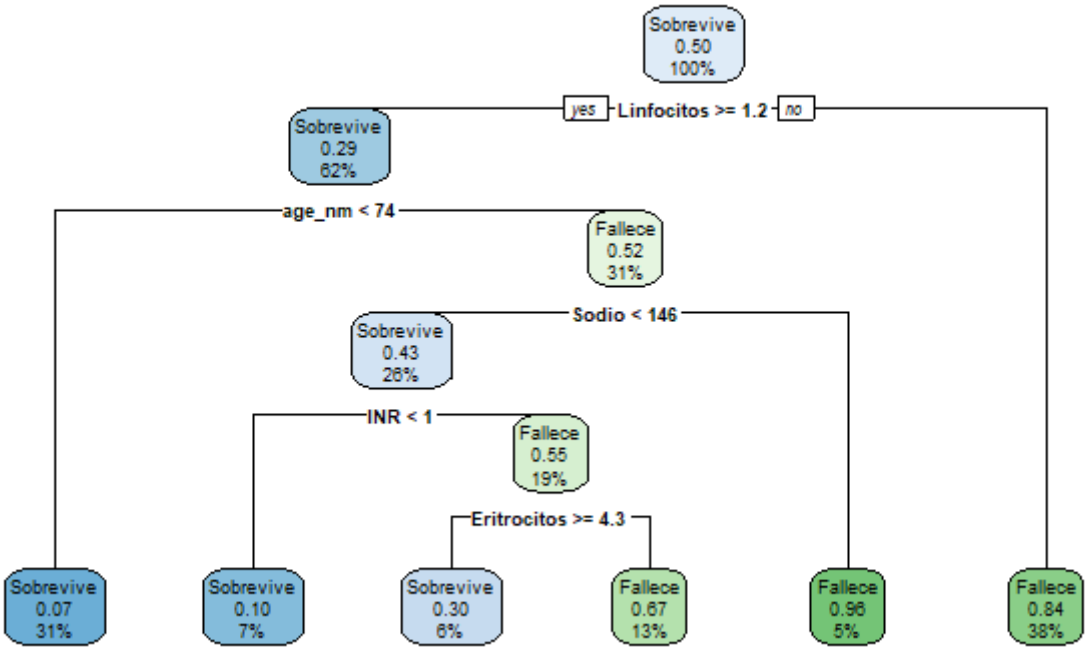


Figura 4.2: Árbol de clasificación construido a partir de los datos de la primera ola.

Se ha llevado a cabo el mismo procedimiento de validación con el resto de olas epidémicas. Los datos del conjunto de validación son los de cada ola posterior respectivamente. Se muestran los resultados en la tabla 4.2. Con estas matrices de confusión se pueden obtener las medidas de calidad del clasficador, que se muestran en la tabla 4.3.

Veamos cuales son los cambios significativos entre distintas olas epidémicas. La exactitud y la sensibilidad no sufren grandes cambios, y esta última es grande debido a que interviene el gran número de verdaderos positivos (la clase positiva hace referencia a los pacientes que sobreviven), que es muy elevado. En cuanto a la especificidad se mantiene constante en la segunda y tercera ola, disminuyendo un 10 % en la cuarta ola y llegando a un valor del 13 % en la quinta ola. Esto se refiere a que el modelo es poco eficaz en detectar muertes en los pacientes ingresados, es decir, entre todos los pacientes que fallecen los diagnostica a muy pocos de ellos, en concreto a 13 de cada 100. Otro parámetro que viene a reflejar una situación similar es el valor predictivo negativo, que disminuye un 10 % aproximadamente en la quinta ola. Más detalladamente, de entre todos los diagnosticados por el árbol como pacientes que fallecen, el 74 % terminan falleciendo realmente en la quinta ola, mientras que cerca del 84-85 % lo hacen en las restantes olas. Otro aspecto reseñable es el aumento del

2ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	592	175
Fallece (pred)	28	167

3ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	2088	621
Fallece (pred)	106	601

4ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1512	536
Fallece (pred)	60	326

5ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1034	308
Fallece (pred)	17	49

Tabla 4.2: Validación para cada ola epidémica.

	2ª ola	**3ª ola**	**4ª ola**	**5ª ola**
Exactitud	0.7890	0.7872	0.7551	0.7692
Sensibilidad	0.9548	0.9517	0.9618	0.9838
Especifidad	0.4883	0.4918	0.3782	0.1373
Val.Pred.Pos	0.7718	0.7708	0.7383	0.7705
Val.Pred.Neg	0.8564	0.8501	0.8446	0.7424
No Information Rate	0.6445	0.6423	0.6459	0.7464
Número de pacientes	962	3416	2434	1408

Tabla 4.3: Resultados de la bondad del ajuste del árbol de clasificación con los datos de cada ola epidémica.

10 % en el *No Information Rate* en la quinta ola, y así el modelo aumentaría su tasa de aciertos diez puntos porcentuales clasificando a todos los individuos en la clase positiva (paciente que sobrevive). En el capítulo 5 se comentan y se tratan las hipótesis que explican estas diferencias mostradas en este mismo capítulo.

4.2. Clasificación con Random Forest.

Se ha empleado el Random Forest (RF) para clasificar a los pacientes, con el paquete `randomForest`. Se han utilizado los datos de la primera ola para construir el modelo. Se ha separado la muestra en dos subconjuntos: entrenamiento (90 %) y validación (10 %). Con el RF tenemos una dificultad añadida respecto a los árboles de clasificación simples, y es que los individuos con valores perdidos no los utiliza para crear el modelo

(por ejemplo, si un paciente no tiene un registro de glucosa en sangre, ese paciente ya no se considera parte de la muestra.) Para solventar este problema se ha utilizado la función `rfImpute()`, que rellena los valores perdidos utilizando una media ponderada, donde los pesos son las proximidades. Los parámetros que se utilizan en nuestro modelo (número de árboles y variables para dividir en cada nodo) se han optimizado con la función `tuneRF()`.

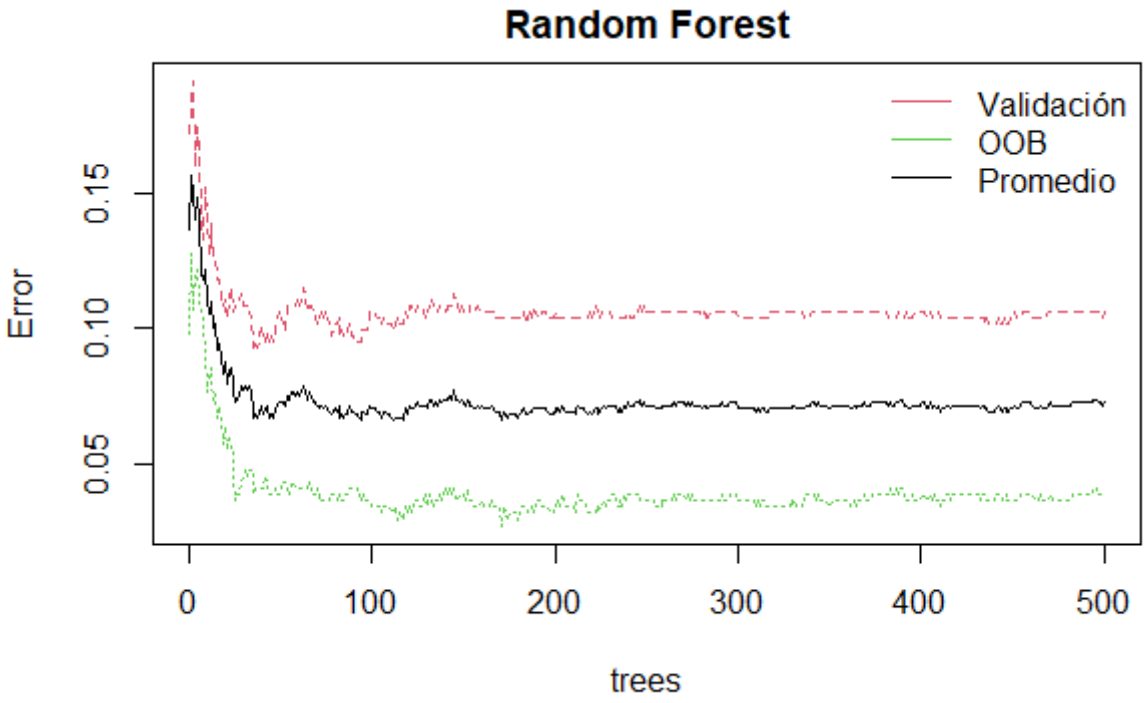


Figura 4.3: Error del Random Forest en función del número de árboles.

Vemos en la figura 4.3 que a partir de una determinada cantidad de árboles empleados el error se mantiene prácticamente constante, por lo tanto coger más de 300 árboles sería poco útil pues el tiempo de computación sería muy elevado para reducir el error una cantidad ínfima. Las medidas de calidad del algoritmo se resumen en la tabla 4.4.

2ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	731	36
Fallece (pred)	46	149

3ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	2550	159
Fallece (pred)	165	542

4ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1926	122
Fallece (pred)	94	292

5ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1308	34
Fallece (pred)	26	40

	2ª ola	**3ª ola**	**4ª ola**	**5ª ola**
Exactitud	0.9148	0.9052	0.9113	0.9574
Sensibilidad	0.9408	0.9392	0.9535	0.9805
Especifidad	0.8054	0.7732	0.7053	0.5405
Val.Pred.Pos	0.9531	0.9413	0.9404	0.9747
Val.Pred.Neg	0.7641	0.7666	0.7565	0.6061
No Information Rate	0.8077	0.7948	0.8299	0.9474
Número de pacientes	962	3416	2434	1408

Tabla 4.4: Matrices de confusión y medidas de la bondad del clasificador para cada ola epidémica con el algoritmo *Random Forest*.

Los datos utilizados para construir el modelo (primera ola) como para los datos de validación (de las olas posteriores) no son los originales, sino que han tenido un tratamiento de *missings* con la función `rfImpute()`; dada esta particularidad en el capítulo 5 se detalla la comparción los mismos datos sin *missings* para los tres algoritmos empleados en el trabajo. La importancia de las variables calculada con la función `varImpPlot` la vemos en la figura 4.4. Vemos que la proteína C reactiva tiene un elevada importancia que en el árbol de clasificación no se apreciaba.

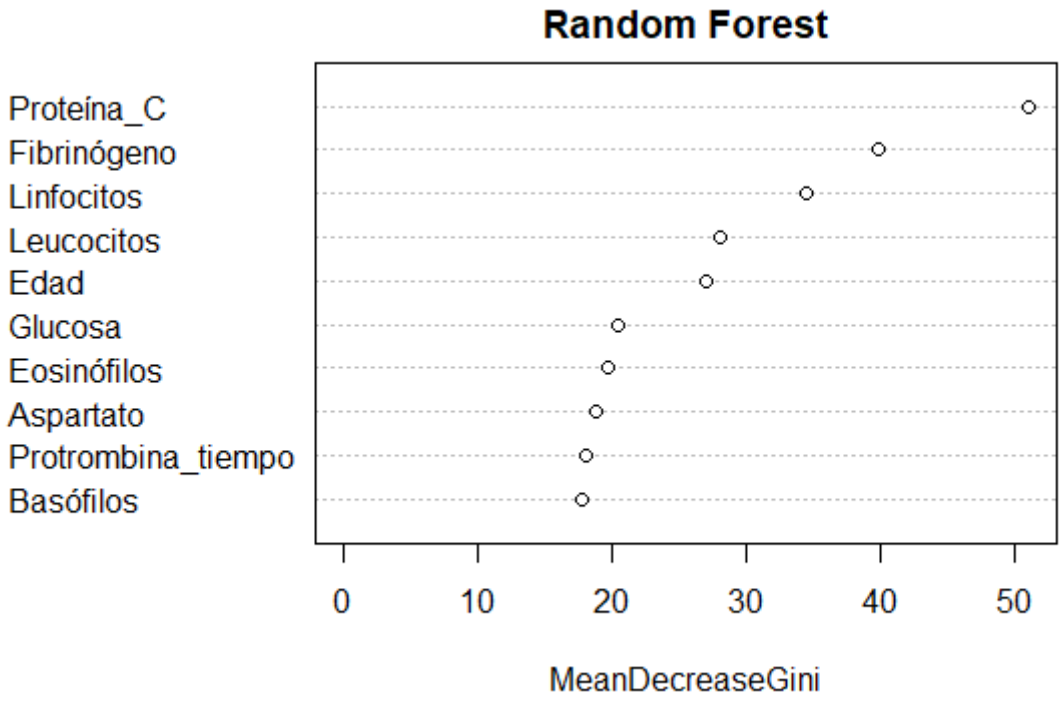


Figura 4.4: Importancia de las variables el Random Forest ya optimizado con la función `tuneRF()`.

4.3. Clasificación con Extreme Gradient Boosting.

Por último se clasificó con el paquete `xgboost`, que utiliza el algoritmo del *Extreme Gradient Boosting* (XGB). Se utilizó un conjunto de entrenamiento del 90 % del tamaño total y un conjunto de validación del 10 %. Se utilizó un número máximo de 200 árboles, un coeficiente de aprendizaje con un valor de 0.3, y una profundidad máxima de cada árbol igual a 6 (siguiendo la propuesta presentada en [6]). Los resultados obtenidos se muestran en las tablas 4.5 y 4.6. Si lo comparamos con el árbol de clasificación, pues los dos han trabajado con los datos del conjunto original (el *Random Forest* ha utilizado datos rellenos) podemos observar diferencias entre ambos algoritmos. La exactitud aumenta en el XGB, donde se mantiene en torno a un 90 % para cada ola epidémica, mientras que en el árbol de clasificación en ninguna ola superaba el 80 % de exactitud. La especifidad, es también bastante mejor en el XGB, obteniendo resultados de más del 70 % para la segunda y tercera ola, mientras que el árbol se quedaba por debajo del 50 % en cada una de estas dos olas. En cuanto al valor predictivo negativo son bastante similares los dos algoritmos, siendo en la quinta ola algo mejor el XGB (80 %) frente al árbol de clasificación (74 %). Por lo tanto confirmamos la hipótesis de que un algoritmo

más elaborado, como es el *Extreme Gradient Boosting*, ofrece una mejor bondad en el ajuste comparado con el árbol de clasificación.

2ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	703	64
Fallece (pred)	31	164

3ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	2492	217
Fallece (pred)	141	566

4ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1839	209
Fallece (pred)	70	316

5ª ola	Sobrevive (ref)	Fallece (ref)
Sobrevive (pred)	1208	134
Fallece (pred)	13	53

Tabla 4.5: Matrices de confusión para cada ola epidémica aplicando el algoritmo del *Extreme Gradient Boosting*.

	2ª ola	**3ª ola**	**4ª ola**	**5ª ola**
Exactitud	0.9012	0.8952	0.8854	0.8956
Sensibilidad	0.9578	0.9464	0.9633	0.9894
Especifidad	0.7193	0.7229	0.6019	0.2834
Val.Pred.Pos	0.9166	0.9199	0.8979	0.9001
Val.Pred.Neg	0.8410	0.8006	0.8187	0.8030
No Information Rate	0.7630	0.7708	0.7843	0.8672
Número de pacientes	962	3416	2434	1408

Tabla 4.6: Resultados de la bondad del algoritmo *Extreme Gradient Boosting*.

4.3.1. Resultados con datos rellenos.

Si se quiere comparar los 3 algoritmos para estudiar cuál clasifica mejor, necesitamos que utilicen los mismos datos, ya que perdería veracidad al utilizar datos distintos tanto para construir los modelos como para validarlos. Por esto se emplean los datos con los *missings* rellenos, tal cómo hemos descrito en el capítulo 2.1.2, con la función `rfImpute`. Se han comparado conjuntamente los resultados de cada indicador de calidad (exactitud, sensibilidad...) haciendo el promedio de las olas para cada algoritmo. Obtenemos los resultados de bondad mostrados en la tabla 4.7.

	Árbol clasificación	Random Forest	Ext Grad Boost
Exactitud	0.8318	0.9222	0.8689
Sensibilidad	0.9608	0.9535	0.9495
Especifidad	0.4384	0.7061	0.5775
Val.Pred.Pos	0.8322	0.9528	0.8972
Val.Pred.Neg	0.7920	0.7233	0.7151
NIR	0.6698	0.7338	0.7958

Tabla 4.7: Comparación de algoritmos con datos rellenados con la función `rfImpute`, de los valores promedio de todas las olas.

Capítulo 5

Conclusiones

Se ha visto la calidad de cada uno de los métodos para predecir la mortalidad intrahospitalaria en Aragón, empleando para entrenarlos los datos de la primera ola epidémica.

Se pueden extraer conclusiones muy interesantes de la tabla 4.7:

- La exactitud de los clasificadores más complejos es un 9,04 % mayor en el Random Forest y un 3,71 % mayor en el XGB respecto al árbol.
- El Random Forest tiene menor acierto a la hora de clasificar correctamente la categoría de "Fallece" (valor predictivo negativo) comparado con los otros dos algoritmos. Sin embargo tiene un 95 % de aciertos a la hora de etiquetar a un paciente que sobrevive (valor predictivo positivo).
- La tasa de pacientes que sobreviven (sensibilidad) es muy elevada en los 3 métodos, en torno al 95 %. La tasa de pacientes que fallecen (especifidad) es claramente mejor en el Random Forest (70,61 %), siendo muy baja en el árbol de clasificación (43,84 %).

Al igual que en el artículo [1], los linfocitos y la edad avanzada son variables que tienen un gran importancia a la hora de clasificar, según la función `filterVarImp`. A lo largo del trabajo se ha puesto de manifiesto que la calidad de los clasificadores empeora notablemente en la quinta ola epidémica. Existen varias hipótesis que exponemos aquí, por ejemplo, el comienzo de la campaña de vacunación, aún en fase muy temprana pero interfiriendo ya en el número de fallecidos y en el expediente clínico del paciente. Tal vez el distinto comportamiento del virus o la inmunidad que consigue la población al aumentar la cantidad de personas con defensas adquiridas contra la enfermedad sean también factores condicionantes. Los 3 algoritmos tienen exactitudes muy correctas, clasificando bien a la mayoría de individuos, pero el *No Information Rate* también es muy elevado, ya que hay una clase predominante en número de individuos, que es la de

pacientes que sobreviven (clase positiva). Pero el proyecto está dirigido esencialmente a comprobar si son útiles o no los algoritmos para etiquetar a los pacientes que fallecen en el hospital, por lo tanto será la especifidad el factor que más relevancia tiene y con él evaluamos la calidad del método clasificador para este trabajo. La especifidad es claramente superior en el *Random Forest*, etiquetando bien al 70 % de la clase negativa, que es la que fallece. Resulta lógico que sea así pues utilizamos centenares de árboles para construir nuestro modelo, aleatorizando el proceso de segmentación de cada nodo. Por lo tanto podemos concluir que el Random Forest será el algoritmo más útil para nuestros fines. Vemos que el árbol de clasificación individual tiene un valor de la especifidad por debajo del 50 %, eso quiere decir que se clasifica peor a los individuos que fallecen con el árbol que con un proceso al azar, por lo tanto no discrimina con eficacia y no sería útil, por lo tanto lo descartamos por su mala especifidad. El *Extreme Gradient Boosting* clasifica correctamente (utilizando los mismos valores de los parámetros del algoritmo que en la sección 4.3) aproximadamente un 58 % de los fallecidos, aportando ligeramente más exactitud que un proceso aleatorio, por lo tanto también sería eficaz considerarlo con estas variables que hemos utilizado a lo largo del estudio. Concluimos el trabajo proponiendo el algoritmo *Random Forest* y *Extreme Gradient Boosting* como clasificadores que en mayor o menor medida son eficaces para etiquetar a pacientes que fallecen en el hospital, habiendo utilizado datos referentes a la región de Aragón.

5.1. Trabajo futuro y valoración personal

Sería importante añadir a este final de la memoria aspectos que mejorar o que puedieran incluirse en un trabajo posterior, con objeto de completar los resultados obtenidos aquí.

Por un lado, en España y en concreto en Aragón, que es la región de la que se extraen los datos, se está llevando a cabo el proceso de vacunación; por lo tanto sería interesante incluir la información relativa a la vacunación de los pacientes como variable relevante en el análisis de riesgo de los mismos. Ya que se dispondría de la fecha de vacunación, de la fecha de ingreso y del día de fallecimiento, se podrían tener en cuenta para ver si estos periodos de tiempo son relevantes a la hora de clasificar, y probablemente eso mejoraría la predicción de la última ola analizada, y más aún la de la última ola epidémica sufrida, y no analizada en este trabajo dado que la fecha de extracción de los datos es anterior a la misma.

Otro matiz que pudiera ser interesante añadir, siempre dentro de la información de la que disponemos a través de los datos sanitarios proporcionados, sería el introducir el tipo de tratamientos o medicamentos que se le han dado a cada paciente, de esta manera se podría discernir entre cuáles son eficaces y cuáles no, de igual forma se podría comparar entre éstos.

En nuestro trabajo utilizamos tres algoritmos de clasificación, pero existen otros métodos que pudieran ser útiles a la hora de clasificar, por ejemplo la regresión logística, o realizar los análisis utilizando otras variables que, a medida que avanza la investigación sobre los mecanismos de interacción del virus con el organismo, pueden ir mostrándose como relevantes, tales como factores genéticos, tratamientos hormonales, etc.

Para terminar este capítulo y la memoria quisiera comentar mi punto de vista personal del trabajo.

Ha sido éste un trabajo desarrollado durante varios meses, que con la ayuda de los directores del trabajo se ha desarrollado con éxito. Tras haber realizado una exhausta revisión bibliográfica, mientras se conseguía el acceso a los datos sanitarios, y una larga búsqueda de artículos académicos, se comenzó el proceso de obtención de resultados y de la elaboración de la memoria. Es un proyecto realizado en el contexto de un TFM (*Trabajo de Fin de Máster*), por lo tanto ha sido necesario explicar algunos aspectos y técnicas que el lector pudiera considerar elementales o básicas. El sentido del trabajo cobra especial relevancia dado que han sido contemporáneas su redacción y el contexto social y sanitario de éste. Han aparecido dificultades y problemas que necesariamente había que superar, por ejemplo, la enorme cantidad de datos faltantes en los registros de los pacientes, o también la complicada interpretación de los resultados obtenidos. Así, como autor del trabajo, el aprendizaje no sólo lo obtengo de los resultados obtenidos sino del desarrollo de un trabajo de esta índole, es decir, construyendo la estructura, la organización, los detalles a la hora de elaborar tablas y gráficas, o incluso de la tarea de cómo enfocar un proyecto de investigación.

Bibliografía

[1] Ibrahim Abdollahpour, Isabel Aguilar-Palacio, Juan Gonzalez-Garcia, Golnaz Vaseghi, Zahra Otroj, Amirreza Manteghinejad, Azam Mosayebi, Yahya Salimi and Shaghayegh Haghjooy Javanmard. *Model Prediction for In-Hospital Mortality in Patients with COVID-19: A Case-Control Study in Isfahan, Iran.* The American journal of tropical medicine and hygiene, pages 1-9, 2021.

[2] A Rodríguez, G Moreno, J Gómez, R Carbonell, E Picó-Plana, C Benavent Bofill, R Sánchez Parrilla, S Trefler, E Pitarch, E Esteve, L Canadell, X Teixido, L Claverias and M Bodí. *Infección grave por coronavirus SARS-CoV-2: experiencia en un hospital de tercer nivel con pacientes afectados por COVID-19 durante la pandemia 2020.* Med Intensiva, 44(9):525-533, 2020.

[3] Sara Velazquez, Rodrigo Madurga, José María Castellano, Jesús Rodriguez-Pascual, Santiago Ruiz De Aguiar Diaz Obregon, Sara Jimeno, Juan Ignacio Montero, Paula Sol, Ventura Wichner and Alejandro López-Escobar. *Hemogram-derived ratios as prognostic markers of ICU admission in COVID-19.* BMC Emergency Medicine (2021)21:89.

[4] N Speybroeck. *Classification and regression trees.* Swiss School of Public Health. Int J Public Health, 57:243–246, 2011.

[5] Leo Breiman. *Random Forests.* Machine Learning **45**:5-32(2001).

[6] Tianqui Chen and Carlos Guestrin. *XGBoost: A Scalable Tree Boosting System.* 785-794. [DOI: 10.1145/2939672.2939785]

[7] Carlos Balsa Castro and Alexandre Sánchez Pla. *Un paquete R para análisis masivo de modelos predictivos de regresión logística multivariante, y sus medidas de discriminación y de clasificación asociadas.* Technical report, 2017.

[8] Max Kuhn.*Package 'caret'. Classification and Regression Training.*

[9] Leo Breiman. *Bagging predictors.* Machine Learning. **26(2)**, 123–140.

[10] ZhongxingZhang et al. *Exploring the clinical features of narcolepsy type 1 versus narcolepsy type 2 from European Narcolepsy Network database with machine learning.* Scientific Reports **8(1)**, July 2018. DOI:10.1038/s41598-018-28840-w.

[11] Terry M. Therneau and Elizabeth J. Atkinson. *An Introduction to Recursive Partitioning Using the RPART Routines.*

[12] Hanley JA and McNeil BJ. *The meaning and use of the area under a receiver operating characteristic (ROC) curve.* Radiology. 1982 Apr; **143(1)**: 29-36. DOI: 10.1148/radiology.143.1.7063747. PMID: 7063747.

Anexos

Anexo A: Código Fuente.

```
    # Librerías
library(caret)
library(xlsx)
library(tidyverse)
library(readxl)
library(readr)
library(RcmdrMisc)
library(DataExplorer)
library(lattice)
library(rpart)
library(rpart.plot)
library(Hmisc)
library(dplyr)
library(data.table)
library(car)

library(rpart)
library(rpart.plot)
library(adabag)
library(randomForest)
library("xgboost")
library(varhandle)

# PREPROCESADO DE LOS DATOS

setwd(" nombre_directorio ")
Casos <- read_delim(" nombre_tabla_casos",
    "|", escape_double = FALSE, trim_ws = TRUE)
Laboratorio <- read_delim(" nombre_tabla_lab",
```

```
    "|", escape_double = FALSE, trim_ws = TRUE)

Casos$exitus_bl <- as.factor(Casos$exitus_bl)
Casos$sex_cd <- as.factor(Casos$sex_cd)
Laboratorio$std_observable_st <- as.factor(Laboratorio$std_observable_st)
Laboratorio$unit_st <- as.factor(Laboratorio$unit_st)

introduce(Casos)
plot_intro(Casos)
plot_missing(Casos)

introduce(Laboratorio)
plot_intro(Laboratorio)
plot_missing(Laboratorio)

Casos$admit_dt <- as.Date(Casos$admit_dt, format = "%d/%m/%Y")
Casos$admit_dt<- as.factor(Casos$admit_dt)
Casos$exitus_dt <- as.Date(Casos$exitus_dt, format = "%d/%m/%Y")
Casos$exitus_dt<- as.factor(Casos$exitus_dt)

plot(Casos$admit_dt, main = "Ingresos")
plot(Casos$exitus_dt, main = "Fallecimientos")

which(Laboratorio$std_observable_st ==
    "Alanine aminotransferase [Enzymatic activity/volume]
    in Serum or Plasma" &
    Laboratorio$unit_st != "U/L")

which(Laboratorio$std_observable_st ==
    "aPTT in Blood by Coagulation assay" &
    Laboratorio$unit_st != "s")

which(Laboratorio$std_observable_st ==
    "Aspartate aminotransferase [Enzymatic
    activity/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "U/L")
```

```
which(Laboratorio$std_observable_st ==
    "Basophils [#/volume] in Blood"  &
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "Bilirubin.direct [Mass/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "mg/dL")

which(Laboratorio$std_observable_st ==
    "Bilirubin.total [Mass/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "mg/dL")

which(Laboratorio$std_observable_st ==
    "C reactive protein [Mass/volume] in Serum or Plasma"
    & Laboratorio$unit_st != "mg/L")

which(Laboratorio$std_observable_st ==
    "CD3 cells/100 cells in Blood" &
    Laboratorio$unit_st != "%")

which(Laboratorio$std_observable_st ==
    "Creatinine [Mass/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "mg/dL")

which(Laboratorio$std_observable_st ==
    "Eosinophils [#/volume] in Blood"  &
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "Erythrocytes [#/volume] in Blood"  &
    Laboratorio$unit_st != "10*6/uL")

which(Laboratorio$std_observable_st ==
    "Fibrinogen [Mass/volume] in Platelet poor plasma by
    Coagulation assay" & Laboratorio$unit_st != "mg/dL")

which(Laboratorio$std_observable_st ==
```

```
    "Glucose [Mass/volume] in Venous blood" &
    Laboratorio$unit_st != "mg/dL")

which(Laboratorio$std_observable_st ==
    "Hematocrit [Volume Fraction] of Blood" &
    Laboratorio$unit_st != "%")

which(Laboratorio$std_observable_st ==
    "INR in Blood by Coagulation assay" &
    Laboratorio$unit_st != "{INR}")

which(Laboratorio$std_observable_st ==
    "Leukocytes [#/volume] in Blood"  &
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "Lymphocytes [#/volume] in Blood"  &
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "MCH [Entitic mass]"  &
    Laboratorio$unit_st != "pg")

which(Laboratorio$std_observable_st ==
    "MCHC [Mass/volume]"   &
    Laboratorio$unit_st != "g/dL")

which(Laboratorio$std_observable_st ==
    "MCV [Entitic volume]"   &
    Laboratorio$unit_st != "fL")

which(Laboratorio$std_observable_st ==
    "Monocytes [#/volume] in Blood"   &
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "Platelets [#/volume] in Blood"   &
```

```
    Laboratorio$unit_st != "10*3/uL")

which(Laboratorio$std_observable_st ==
    "Potassium [Moles/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "mmol/L")

which(Laboratorio$std_observable_st ==
    "Prothrombin time (PT)" &
    Laboratorio$unit_st != "s")

which(Laboratorio$std_observable_st ==
    "Sodium [Moles/volume] in Serum or Plasma" &
    Laboratorio$unit_st != "mmol/L")

df_ancha <- dcast(Laboratorio,
                  person_id ~ std_observable_st,
                  value.var = "obs_value_nm")

dataset <- merge(x = Casos, y = df_ancha)

dataset$exitus_bl <- factor(dataset$exitus_bl, levels = c(0, 1),
                            labels = c("Sobrevive", "Fallece"))
dataset$sex_cd <- factor(dataset$sex_cd, levels = c("1", "2"),
                         labels = c("Hombre", "Mujer"))

introduce(dataset)
plot_bar(dataset$sex_cd)
plot_bar(dataset$exitus_bl)
plot_histogram(dataset[, c(3,4,11:35)])

plot_missing(dataset[,c(3,4, 11:35)])

describe(dataset[,-c(1,2,5:10, 15,18)])

outliersReplace <- function(data, titulo){
  variable <- data
  Q <- quantile(variable, probs=c(.5, .95), na.rm = TRUE)
```

```
  iqr <- IQR(variable, na.rm = TRUE)
  up <-  Q[2]+1.5*iqr # Rango superior
  low <- Q[1]-1.5*iqr # Rango inferior

  eliminated <- subset(data, variable > low & variable < up)
  plot_histogram(eliminated, title = titulo)
}

outliersReplace(dataset$`Alanine aminotransferase [Enzymatic
    activity/volume] in Serum or Plasma`, "Alanina");
outliersReplace(dataset$`aPTT in Blood by Coagulation     assay`, "aPTT");
outliersReplace(dataset$`Aspartate aminotransferase [Enzymatic
    activity/volume] in Serum or Plasma`, "Aspartato");
outliersReplace(dataset$`Bilirubin.total [Mass/volume]
    in Serum or Plasma`, "Bilirrubina_Total");
outliersReplace(dataset$`C reactive protein
    [Mass/volume] in Serum or Plasma`, "Proteina_C");
outliersReplace(dataset$`Creatinine [Mass/volume]
    in Serum or Plasma`, "Creatinina");
outliersReplace(dataset$`Glucose [Mass/volume]
    in Venous blood`, "Glucosa");
outliersReplace(dataset$`Hematocrit [Volume Fraction]
    of Blood`, "Hematocrito");
outliersReplace(dataset$`INR in Blood by
    Coagulation assay`, "INR");
outliersReplace(dataset$`Potassium [Moles/volume]
    in Serum or Plasma`, "Potasio");
outliersReplace(dataset$`Prothrombin time (PT)`, "PT");
outliersReplace(dataset$`Sodium [Moles/volume]
    in Serum or Plasma`, "Sodio");
outliersReplace(dataset$`Basophils [#/volume]
    in Blood`, "Basofilos");
outliersReplace(dataset$`Eosinophils [#/volume]
    in Blood`, "Eosiniofilos");
outliersReplace(dataset$`Erythrocytes [#/volume]
    in Blood`, "Eritrocitos");
outliersReplace(dataset$`Fibrinogen [Mass/volume]
```

```
    in Platelet poor plasma by Coagulation
    assay‘, "Fibrinogeno");
outliersReplace(dataset$‘Leukocytes [#/volume]
    in Blood‘, "Leucocitos");
outliersReplace(dataset$‘Lymphocytes [#/volume]
    in Blood‘, "Linfocitos");
outliersReplace(dataset$‘MCH [Entitic mass]‘, "MCH");
outliersReplace(dataset$‘MCHC [Mass/volume]‘, "MCHC");
outliersReplace(dataset$‘MCV [Entitic volume]‘, "MCV");
outliersReplace(dataset$‘Monocytes [#/volume]
    in Blood‘, "Monocitos");
outliersReplace(dataset$‘Platelets [#/volume]
    in Blood‘, "Plaquetas");

dataset$‘Leukocytes [#/volume] in Blood‘
    [which(dataset$‘Leukocytes [#/volume] in
    Blood‘ == 0)] <- NA;
dataset$‘INR in Blood by Coagulation assay‘
    [which(dataset$‘INR in
    Blood by Coagulation assay‘ == 0)] <- NA;
dataset$‘Eosinophils [#/volume] in Blood‘
    [which(dataset$‘Eosinophils [#/volume] in
    Blood‘ == 0)] <- NA;
dataset$‘Basophils [#/volume] in Blood‘
    [which(dataset$ ‘Basophils [#/volume] in
    Blood‘ == 0)] <- NA;

rocVarImp<-filterVarImp(x = dataset[, -c(1,2,4,5:10,15,18)],
                        y = as.factor(dataset$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()
prop.table(table(dataset$exitus_bl)) * 100

out <- dataset
outliers <- function(data){
  variable <- data
  Q <- quantile(variable, probs=c(.5, .95), na.rm = TRUE)
  iqr <- IQR(variable, na.rm = TRUE)
```

```
  up <-  Q[2]+1.5*iqr # Rango superior
  low <- Q[1]-1.5*iqr # Rango inferior
  data[which(variable < low | variable > up)] <- NA
  return(data)
}
out$‘Aspartate aminotransferase [Enzymatic activity/volume]
    in Serum or Plasma‘<- outliers(out$‘Aspartate aminotransferase
    [Enzymatic activity/volume] in Serum or Plasma‘);
out$‘Prothrombin time (PT)‘ <- outliers(out$‘Prothrombin time (PT)‘);
out$‘INR in Blood by Coagulation assay‘ <-
    outliers(out$‘INR in Blood by Coagulation assay‘);
out$‘Creatinine [Mass/volume] in Serum or Plasma‘ <-
    outliers(out$‘Creatinine [Mass/volume] in Serum or Plasma‘);
out$age_nm <- outliers(out$age_nm);
plot(out[,c(3,21,24,25,27,34)],
pch='*',col=c("red", "green")[(out[,5])])

names(out)[3] <- "Edad"
names(out)[11] <- "Alanina"
names(out)[12] <- "aPTT"
names(out)[13] <- "Aspartato"
names(out)[14] <- "Basofilos"
names(out)[15] <- "Bilirrubina_directa"
names(out)[16] <- "Bilirrubina_total"
names(out)[17] <- "Proteína_C"
names(out)[18] <- "CD3"
names(out)[19] <- "Creatinina"
names(out)[20] <- "Eosinofilos"
names(out)[21] <- "Eritrocitos"
names(out)[22] <- "Fibrinogeno"
names(out)[23] <- "Glucosa"
names(out)[24] <- "Hematocrito"
names(out)[25] <- "INR"
names(out)[26] <- "Leucocitos"
names(out)[27] <- "Linfocitos"
names(out)[28] <- "MCH"
names(out)[29] <- "MCHC"
```

```
names(out)[30] <- "MCV"
names(out)[31] <- "Monocitos"
names(out)[32] <- "Plaquetas"
names(out)[33] <- "Potasio"
names(out)[34] <- "PT"
names(out)[35] <- "Sodio"

scatterplotMatrix(out[,c(3,25,27,34)], groups=out$exitus_bl)

names(dataset)[11] <- "Alanina"
names(dataset)[12] <- "aPTT"
names(dataset)[13] <- "Aspartato"
names(dataset)[14] <- "Basofilos"
names(dataset)[15] <- "Bilirrubina_directa"
names(dataset)[16] <- "Bilirrubina_total"
names(dataset)[17] <- "Proteína_C"
names(dataset)[18] <- "CD3"
names(dataset)[19] <- "Creatinina"
names(dataset)[20] <- "Eosinofilos"
names(dataset)[21] <- "Eritrocitos"
names(dataset)[22] <- "Fibrinogeno"
names(dataset)[23] <- "Glucosa"
names(dataset)[24] <- "Hematocrito"
names(dataset)[25] <- "INR"
names(dataset)[26] <- "Leucocitos"
names(dataset)[27] <- "Linfocitos"
names(dataset)[28] <- "MCH"
names(dataset)[29] <- "MCHC"
names(dataset)[30] <- "MCV"
names(dataset)[31] <- "Monocitos"
names(dataset)[32] <- "Plaquetas"
names(dataset)[33] <- "Potasio"
names(dataset)[34] <- "Protrombina_tiempo"
names(dataset)[35] <- "Sodio"

cor(na.omit(dataset[,c(3,11:14,16,17,19:35)]))
```

```
dataset$admit_dt <- as.Date(dataset$admit_dt, format = "%Y-%m-%d")

Primera_ola <- dataset[which(dataset$admit_dt >= "2020-03-03"
  & dataset$admit_dt <= "2020-06-07"),]
Segunda_ola <- dataset[which(dataset$admit_dt >= "2020-06-08"
  & dataset$admit_dt <= "2020-08-23"),]
Tercera_ola <- dataset[which(dataset$admit_dt >= "2020-08-24"
  & dataset$admit_dt <= "2020-12-13"),]
Cuarta_ola <- dataset[which(dataset$admit_dt >= "2020-12-14"
  & dataset$admit_dt<= "2021-03-21"),]
Quinta_ola <- dataset[which(dataset$admit_dt >= "2021-03-22"),]

rocVarImp<-filterVarImp(x = Primera_ola[, c(3,11:14,16,17,19:35)],
  y = as.factor(Primera_ola$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()
prop.table(table(Primera_ola$exitus_bl)) * 100

rocVarImp<-filterVarImp(x = Segunda_ola[, c(3,11:14,16,17,19:35)],
  y = as.factor(Segunda_ola$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()
prop.table(table(Segunda_ola$exitus_bl)) * 100

rocVarImp<-filterVarImp(x = Tercera_ola[, c(3,11:14,16,17,19:35)],
  y = as.factor(Tercera_ola$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()
prop.table(table(Tercera_ola$exitus_bl)) * 100

rocVarImp<-filterVarImp(x = Cuarta_ola[, c(3,11:14,16,17,19:35)],
  y = as.factor(Cuarta_ola$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()
prop.table(table(Cuarta_ola$exitus_bl)) * 100

rocVarImp<-filterVarImp(x = Quinta_ola[, c(3,11:14,16,17,19:35)],
  y = as.factor(Quinta_ola$exitus_bl))
apply(rocVarImp, 1, mean) %>% sort()

prop.table(table(Segunda_ola$exitus_bl)) * 100
```

```
prop.table(table(Tercera_ola$exitus_bl)) * 100
prop.table(table(Cuarta_ola$exitus_bl)) * 100
prop.table(table(Quinta_ola$exitus_bl)) * 100

set.seed(1237)
Primera_ola_upSampled = upSample(Primera_ola[,],
  Primera_ola$exitus_bl)
table(Primera_ola_upSampled$Class)
table(Primera_ola$exitus_bl)
prop.table(table(Quinta_ola$exitus_bl)) * 100
set.seed(1237)
Segunda_ola_upSampled = upSample(Segunda_ola[, ],
  Segunda_ola$exitus_bl)
table(Segunda_ola_upSampled$Class)
table(Segunda_ola$exitus_bl)
set.seed(1237)
Tercera_ola_upSampled = upSample(Tercera_ola[, ],
  Tercera_ola$exitus_bl)
table(Tercera_ola_upSampled$Class)
table(Tercera_ola$exitus_bl)
set.seed(1237)
Cuarta_ola_upSampled = upSample(Cuarta_ola[,],
  Cuarta_ola$exitus_bl)
table(Cuarta_ola_upSampled$Class)
table(Cuarta_ola$exitus_bl)
set.seed(1237)
Quinta_ola_upSampled = upSample(Quinta_ola[, ],
  Quinta_ola$exitus_bl)
table(Quinta_ola_upSampled$Class)
table(Quinta_ola$exitus_bl)

save(list = ls(all.names = TRUE),
  file = "MisDatos.RData", envir = .GlobalEnv)

## ARBOL DE CLASIFICACION

load("MisDatos.RData")
```

```
set.seed(200323)

indtrain<-createDataPartition(Primera_ola_upSampled$exitus_bl,
  p=0.9, list = FALSE)
entrenamiento<-Primera_ola_upSampled[indtrain,]
test<-Primera_ola_upSampled[-indtrain,]

arbol <- rpart( exitus_bl ~ age_nm + sex_cd+ Alanina + aPTT + Aspartato +
  Basofilos + Bilirrubina_total + Proteína_C + Creatinina + Eosinofilos +
  Eritrocitos + Fibrinogeno + Glucosa + Hematocrito + INR + Leucocitos +
  Linfocitos + MCH + MCHC + MCV + Monocitos + Plaquetas + Potasio +
  Protrombina_tiempo + Sodio, data = entrenamiento, method="class",
  control=list(  minsplit=10,
  minbucket=2,
  maxdepth=10,
  xval=10 # k in k-fold CV)
  ))

rpart.plot(arbol)
summary(arbol)
plotcp(arbol)

arbol_optimizado<-prune(arbol, cp=0.026)
rpart.plot(arbol_optimizado)
head(summary(arbol_optimizado),60)

y_pred <- predict(arbol_optimizado, newdata = test, type = "class")
matriz_confusion <- confusionMatrix(test$exitus_bl, y_pred)
matriz_confusion
y_pred_SegundaOla <-  predict(arbol_optimizado,
  newdata = Segunda_ola, type = "class")
matriz_confusion2 <- confusionMatrix(Segunda_ola$exitus_bl, y_pred_SegundaOla)
matriz_confusion2
y_pred_TerceraOla <-  predict(arbol_optimizado,
  newdata = Tercera_ola, type = "class")
matriz_confusion3 <- confusionMatrix(Tercera_ola$exitus_bl, y_pred_TerceraOla)
matriz_confusion3
```

```
y_pred_CuartaOla <-  predict(arbol_optimizado,
  newdata = Cuarta_ola, type = "class")
matriz_confusion4 <- confusionMatrix(Cuarta_ola$exitus_bl, y_pred_CuartaOla)
matriz_confusion4
y_pred_QuintaOla <-  predict(arbol_optimizado,
  newdata = Quinta_ola, type = "class")
matriz_confusion5 <- confusionMatrix(Quinta_ola$exitus_bl, y_pred_QuintaOla)
matriz_confusion5

## RANDOM FOREST


load("MisDatos.RData")
dataset_RF <- Primera_ola_upSampled[,c(3,4,5,11:35)]
dataset_RF$exitus_bl <- as.factor(dataset_RF$exitus_bl)
n <- nrow(dataset_RF)
entrena <- sample(c(1:n), round(0.90*n))
valida <- setdiff(c(1:n), entrena)

set.seed(222)
data.impute <- rfImpute(exitus_bl ~ ., dataset_RF)

data.rf <- randomForest(exitus_bl ~ age_nm + sex_cd + Alanina +
  aPTT + Aspartato + Basofilos + Bilirrubina_total + Proteína_C +
  Creatinina + Eosinofilos + Eritrocitos + Fibrinogeno + Glucosa +
  Hematocrito + INR + Leucocitos + Linfocitos + MCH + MCHC + MCV +
  Monocitos + Plaquetas + Potasio + Protrombina_tiempo + Sodio,
  data=data.impute, subset = entrena,
  xtest=data.impute[valida,c(2:7,9,10,12:28)],
  ytest=data.impute[valida,1])
data.rf

plot(data.rf)
varImpPlot(data.rf, n.var = 10)

set.seed(444)
data.tuneRF<-tuneRF(data.impute[entrena,c(2:7,9,10,12:28)],
```

```
  data.impute[entrena,1], ntreeTry=300, mtryStart=3,
  stepFactor=1.5, plot=TRUE, trace=TRUE, doBest=TRUE)
data.tuneRF

varImpPlot(data.tuneRF, n.var = 10, main = "Random Forest")
plot(data.tuneRF, main = "Random Forest")
legend(x = "topright",legend=c("Validación", "OOB", "Promedio"),
  col=c(2, 3, "black"), lty = c(1, 1, 1), bty = "n")

set.seed(345)
predictTREE <- predict(data.tuneRF, newdata = data.impute[valida,],
  type="class")
confusionMatrix(data.impute[valida,]$exitus_bl, as.factor(predictTREE))

newdata <- Segunda_ola[,c(3,4,5,11:35)]
set.seed(222)
newdata_2 <- rfImpute(exitus_bl ~ ., newdata)
predictTREE_2 <- predict(data.tuneRF, newdata = newdata_2[,], type="class")
confusionMatrix(newdata_2$exitus_bl, as.factor(predictTREE_2))

newdata <- Tercera_ola[,c(3,4,5,11:35)]
set.seed(222)
newdata_3 <- rfImpute(exitus_bl ~ ., newdata)
predictTREE_3 <- predict(data.tuneRF, newdata = newdata_3[,], type="class")
confusionMatrix(newdata_3$exitus_bl, as.factor(predictTREE_3))

newdata <- Cuarta_ola[,c(3,4,5,11:35)]

set.seed(222)
newdata_4 <- rfImpute(exitus_bl ~ ., newdata)
predictTREE_4 <- predict(data.tuneRF, newdata = newdata_4[,], type="class")
confusionMatrix(newdata_4$exitus_bl, as.factor(predictTREE_4))

newdata <- Quinta_ola[,c(3,4,5,11:35)]
set.seed(222)
newdata_5 <- rfImpute(exitus_bl ~ ., newdata)
predictTREE_5 <- predict(data.tuneRF, newdata = newdata_5[,], type="class")
```

```
confusionMatrix(newdata_5$exitus_bl, as.factor(predictTREE_5))

save(list = ls(all.names = TRUE), file = "Datos_RF.RData",
  envir = .GlobalEnv)

## EXTREME GRADIENT BOOSTING

load("MisDatos.RData")
data <- Primera_ola_upSampled[c(3,4,5,11:14,16,17,19:35)]

data$sex_cd <- factor(data$sex_cd, levels = c("Hombre", "Mujer"),
  labels = c("0", "1"))
data$exitus_bl <- factor(data$exitus_bl,c("Sobrevive", "Fallece"),
  labels =  c(0, 1))
data$sex_cd <- unfactor(data$sex_cd)
data$exitus_bl <- unfactor(data$exitus_bl)
class(data$sex_cd)

set.seed(1919)
train_df <- sample_frac(data, size = 0.9)
test_df <- setdiff(data, train_df)

train_mat <-
  train_df %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = train_df$exitus_bl)

test_mat <-
  test_df %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = test_df$exitus_bl)

modelo_01 <- xgboost(data = train_mat,
  objective = "binary:logistic",
  nrounds = 200, max.depth = 6, eta = 0.3, nthread = 2)
```

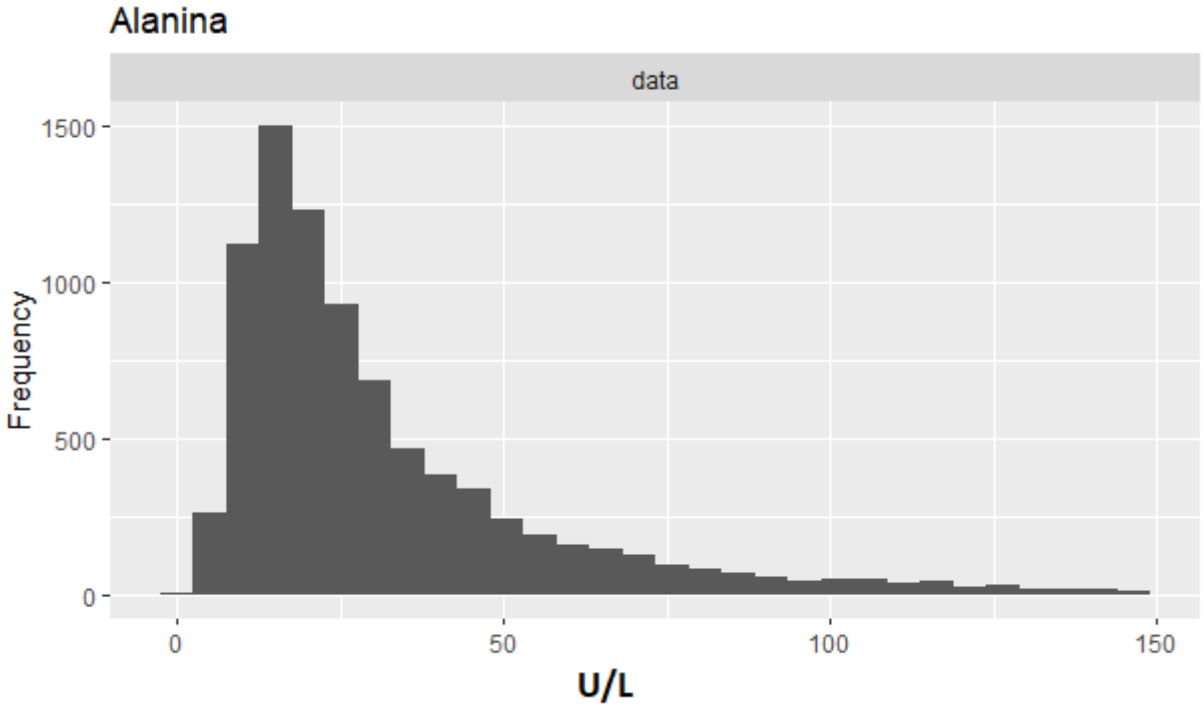
```
predict_01 <- predict(modelo_01, test_mat)

cbind(test_df$exitus_bl, predict_01 > 0.5) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()

data_2 <- Segunda_ola[c(3,4,5,11:14,16,17,19:35)]
data_2$sex_cd <- factor(data_2$sex_cd, levels = c("Hombre", "Mujer"),
  labels = c("0", "1"))
data_2$exitus_bl <- factor(data_2$exitus_bl,c("Sobrevive", "Fallece"),
  labels =  c(0, 1))
data_2$sex_cd <- unfactor(data_2$sex_cd)
data_2$exitus_bl <- unfactor(data_2$exitus_bl)
Ola_2_mat <-
  data_2 %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = data_2$exitus_bl)
predict_02 <- predict(modelo_01, Ola_2_mat)
cbind(data_2$exitus_bl,predict_02 > 0.5) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()

data_3 <- Tercera_ola[c(3,4,5,11:14,16,17,19:35)]
data_3$sex_cd <- factor(data_3$sex_cd, levels = c("Hombre", "Mujer"),
  labels = c("0", "1"))
data_3$exitus_bl <- factor(data_3$exitus_bl,c("Sobrevive", "Fallece"),
  labels =  c(0, 1))
data_3$sex_cd <- unfactor(data_3$sex_cd)
data_3$exitus_bl <- unfactor(data_3$exitus_bl)
Ola_3_mat <-
  data_3 %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
```

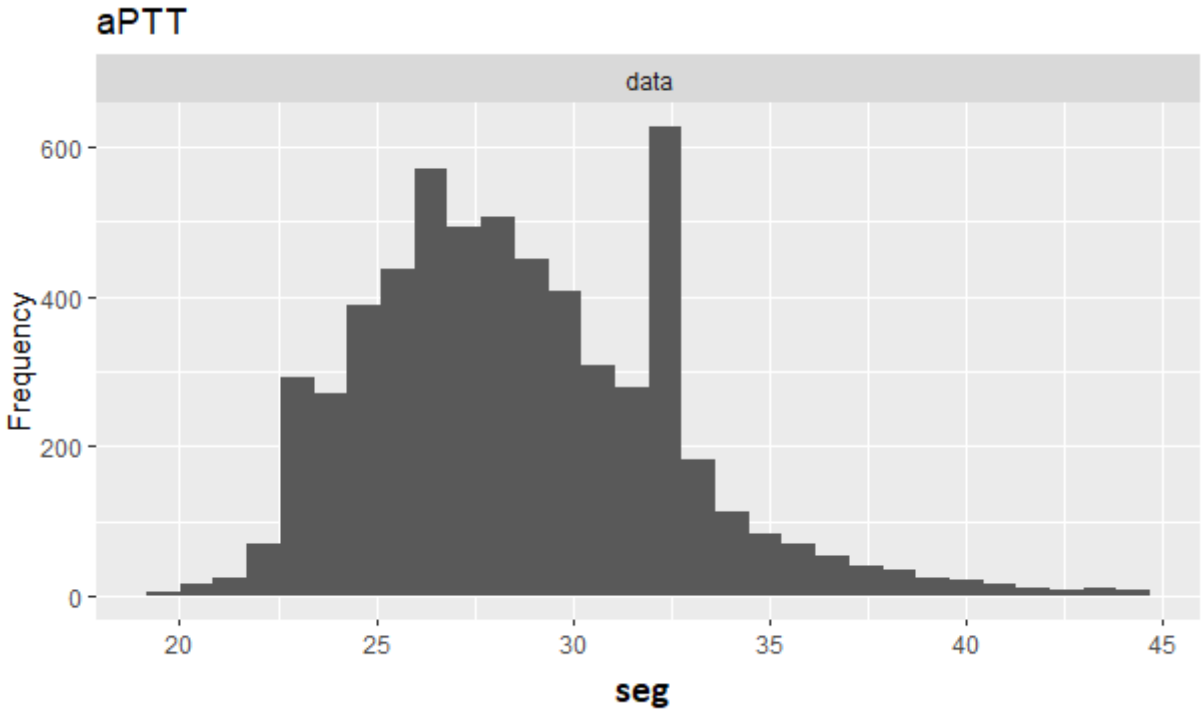
```
  xgb.DMatrix(data = ., label = data_3$exitus_bl)
predict_03 <- predict(modelo_01, Ola_3_mat)
cbind(data_3$exitus_bl,predict_03 > 0.5) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()

data_4 <- Cuarta_ola[c(3,4,5,11:14,16,17,19:35)]
data_4$sex_cd <- factor(data_4$sex_cd, levels = c("Hombre", "Mujer"),
  labels = c("0", "1"))
data_4$exitus_bl <- factor(data_4$exitus_bl,c("Sobrevive", "Fallece"),
  labels =  c(0, 1))
data_4$sex_cd <- unfactor(data_4$sex_cd)
data_4$exitus_bl <- unfactor(data_4$exitus_bl)
Ola_4_mat <-
  data_4 %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = data_4$exitus_bl)
predict_04 <- predict(modelo_01, Ola_4_mat)
cbind(data_4$exitus_bl,predict_04 > 0.5) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()

data_5 <- Quinta_ola[c(3,4,5,11:14,16,17,19:35)]
data_5$sex_cd <- factor(data_5$sex_cd, levels = c("Hombre", "Mujer"),
  labels = c("0", "1"))
data_5$exitus_bl <- factor(data_5$exitus_bl,c("Sobrevive", "Fallece"),
  labels =  c(0, 1))
data_5$sex_cd <- unfactor(data_5$sex_cd)
data_5$exitus_bl <- unfactor(data_5$exitus_bl)
Ola_5_mat <-
  data_5 %>%
  select(-exitus_bl) %>%
  as.matrix() %>%
  xgb.DMatrix(data = ., label = data_5$exitus_bl)
```

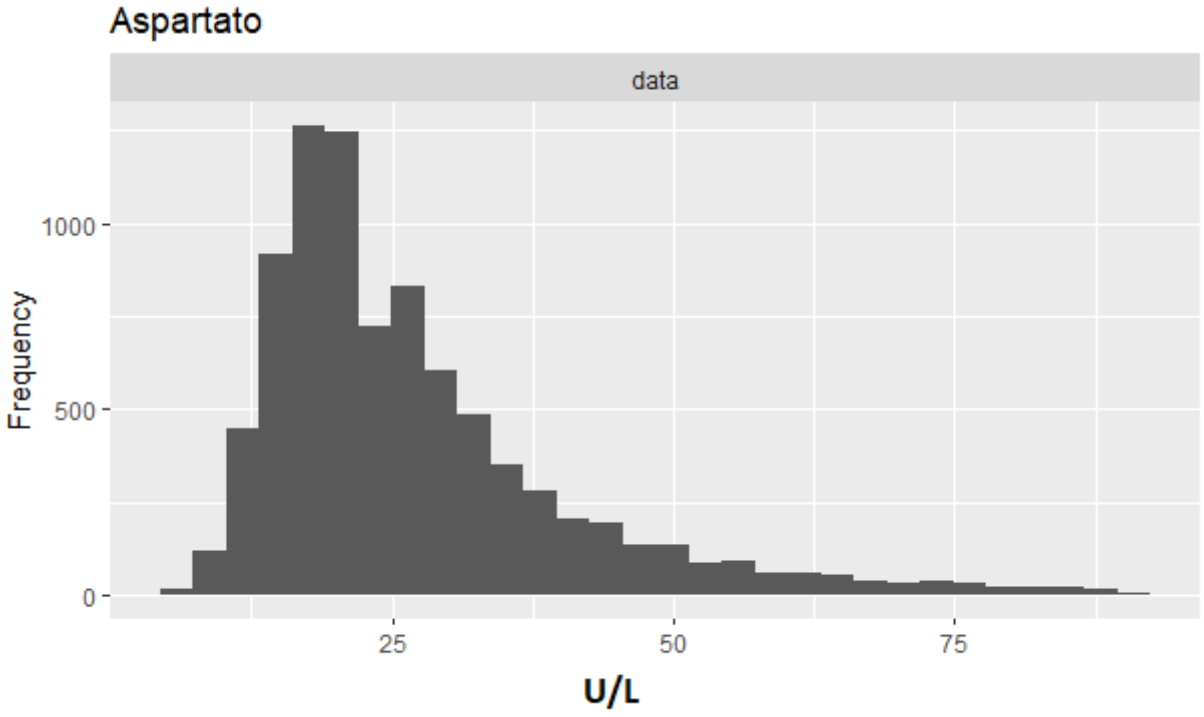
```
predict_05 <- predict(modelo_01, Ola_5_mat)
cbind(data_5$exitus_bl,predict_05 > 0.5) %>%
  data.frame() %>%
  table() %>%
  confusionMatrix()
```

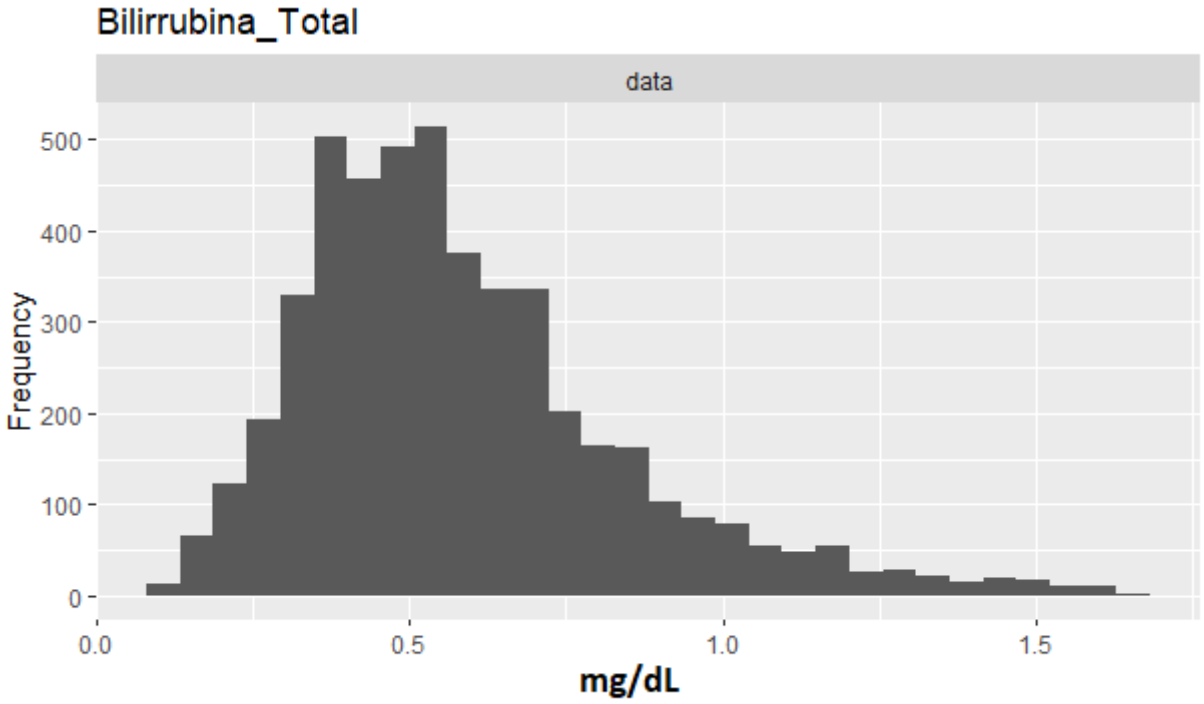
Anexo B: Gráficas.

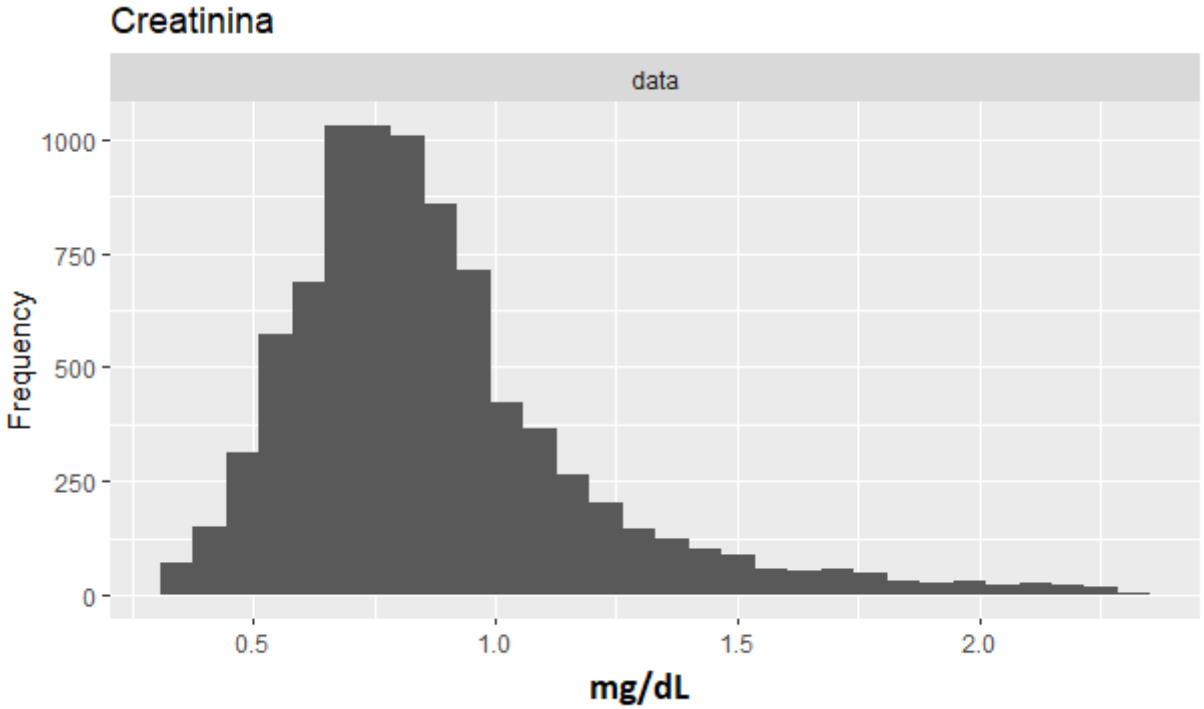
Distribución de las variables utilizadas en el trabajo:

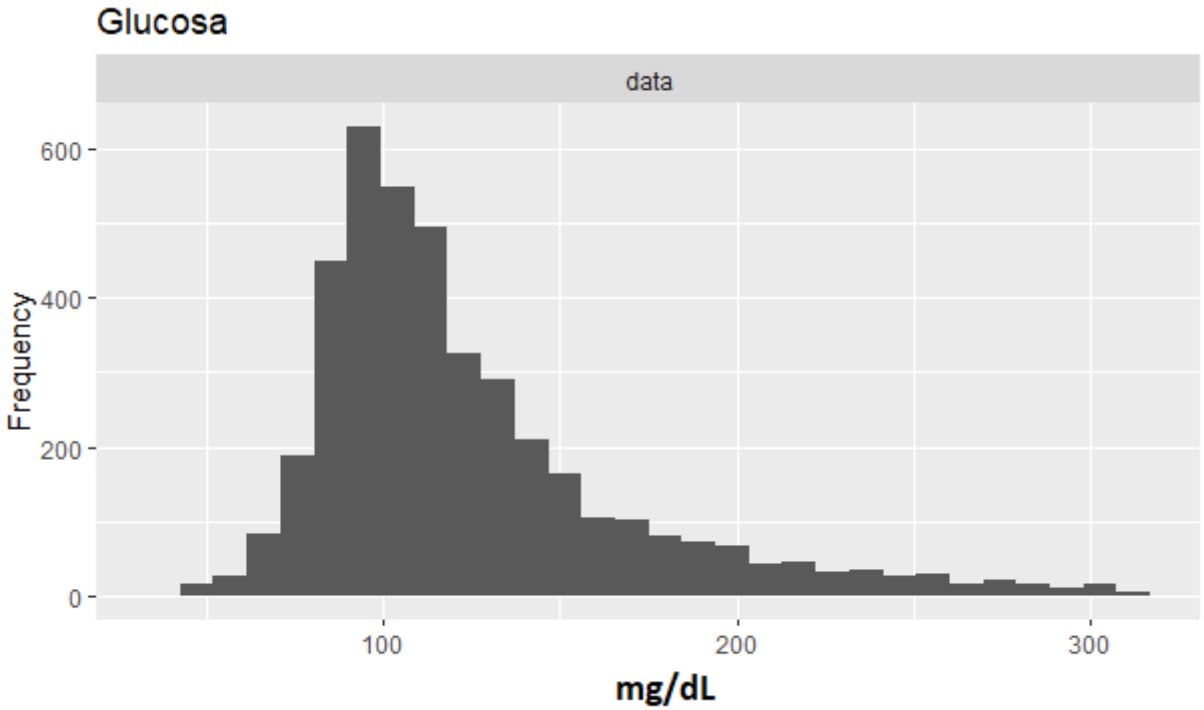


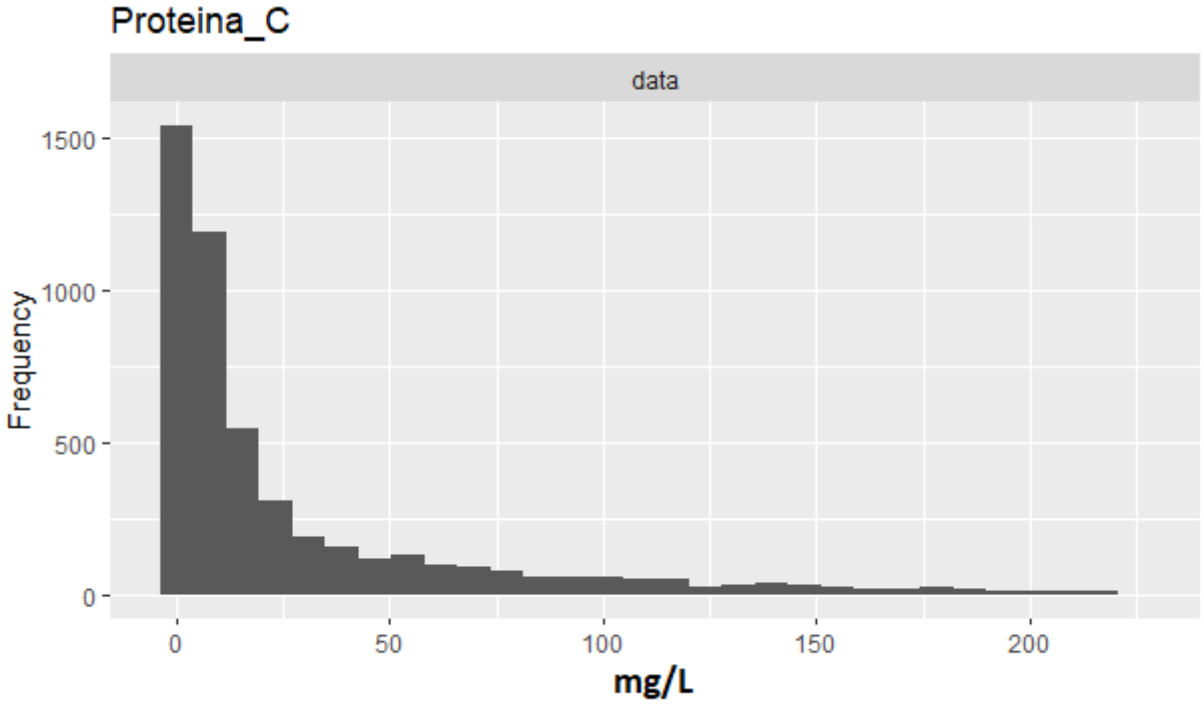

aPTT
data
600
400
200
0
Frequency
20
25
30
35
40
45
seg

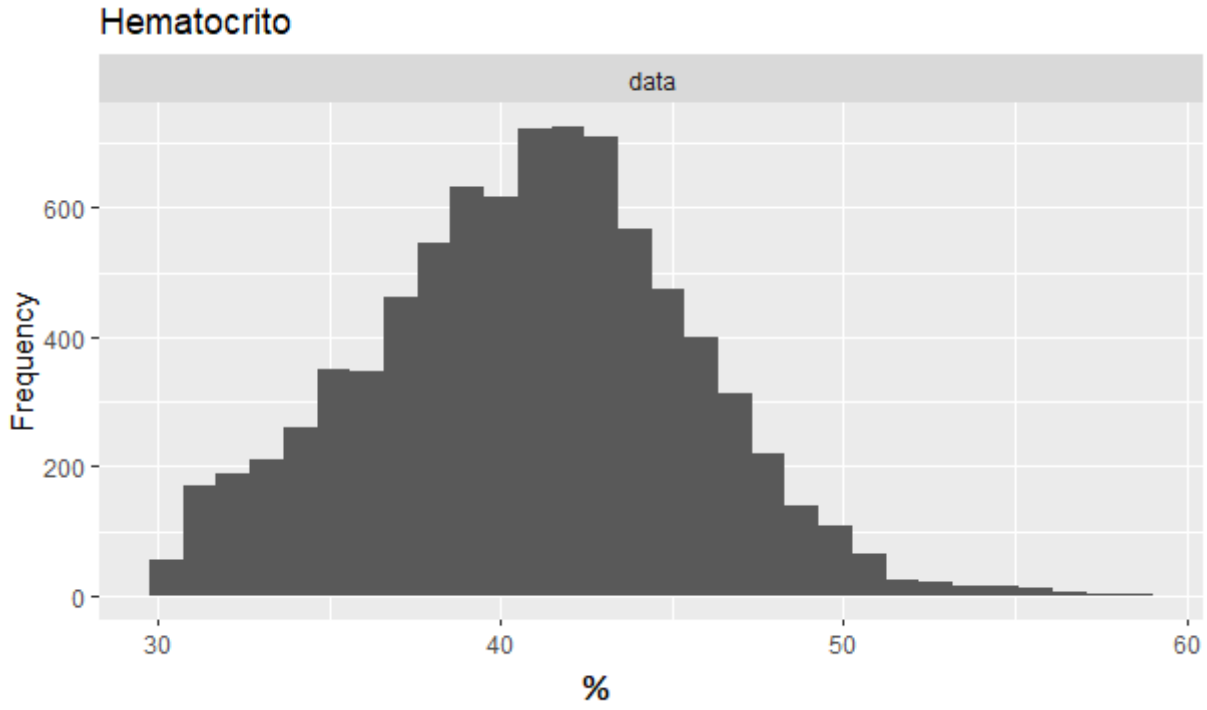

Aspartato
data
1000
500
0
Frequency
25
50
75
U/L

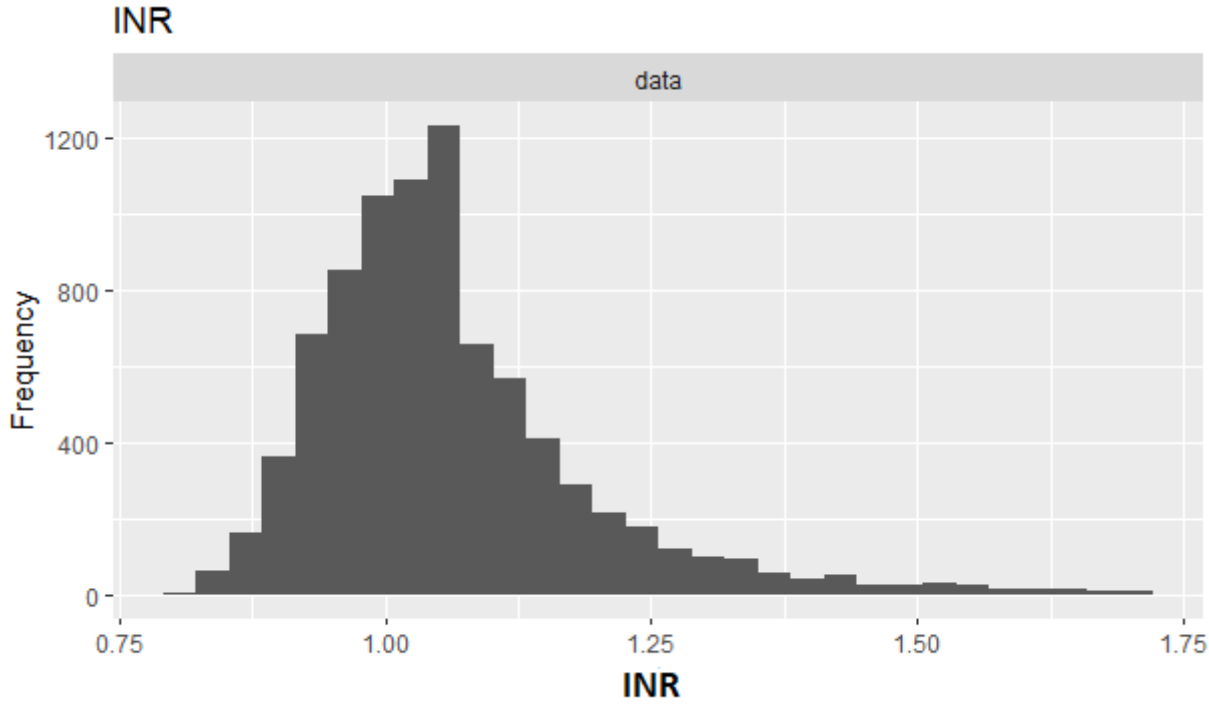

Bilirrubina_Total
data
Frequency
500
400
300
200
100
0
0.0
0.5
1.0
1.5
mg/dL

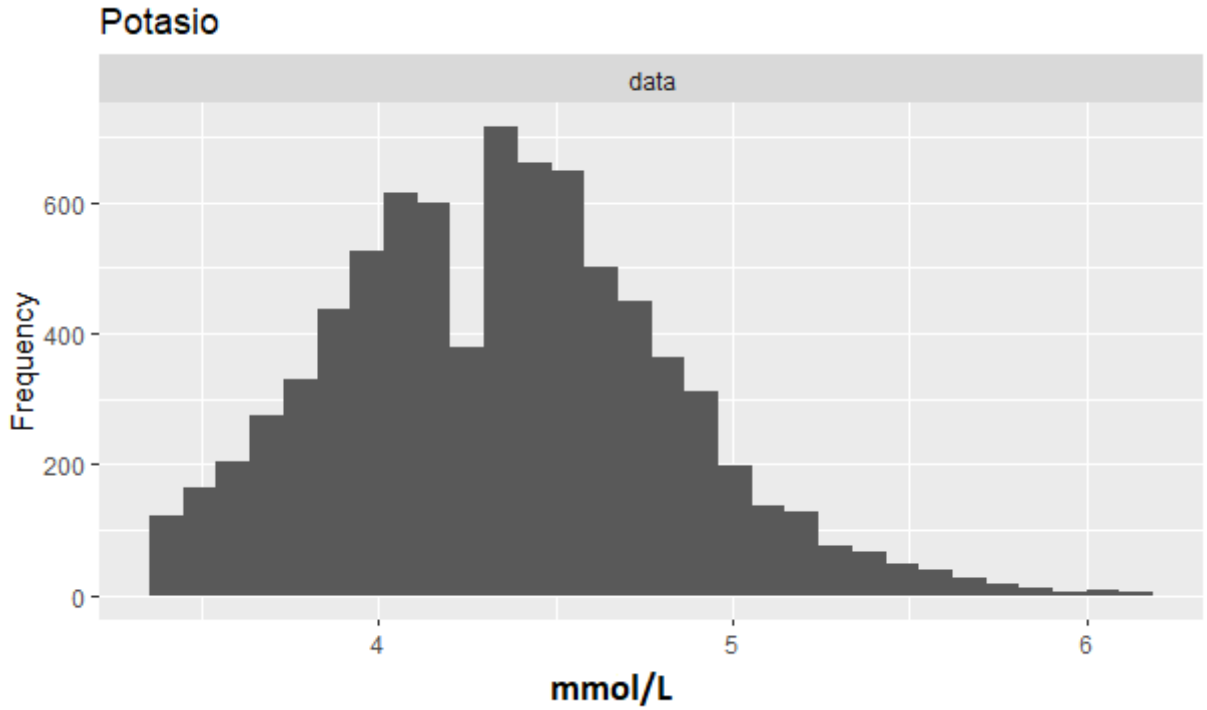

Creatinina
data
Frequency
1000
750
500
250
0
0.5
1.0
1.5
2.0
mg/dL

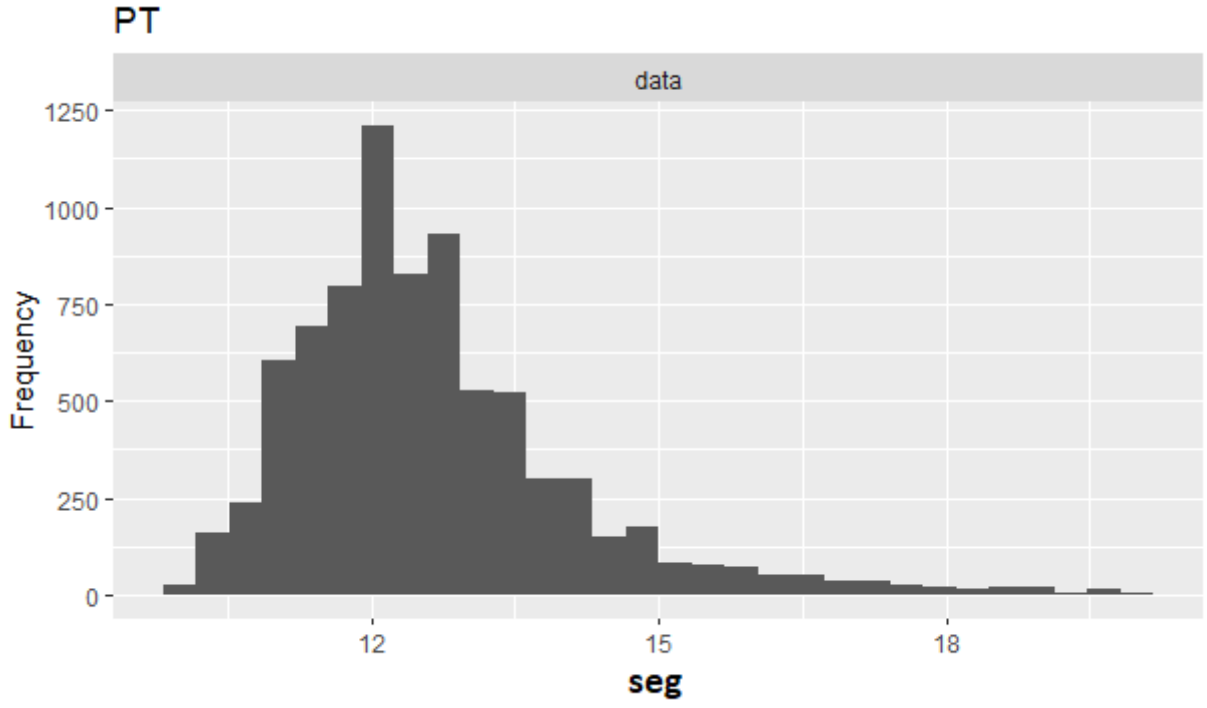

Glucosa
data
600
400
200
0
Frequency
100
200
300
mg/dL


Proteina_C
data
1500
1000
500
0
Frequency
0
50
100
150
200
mg/L

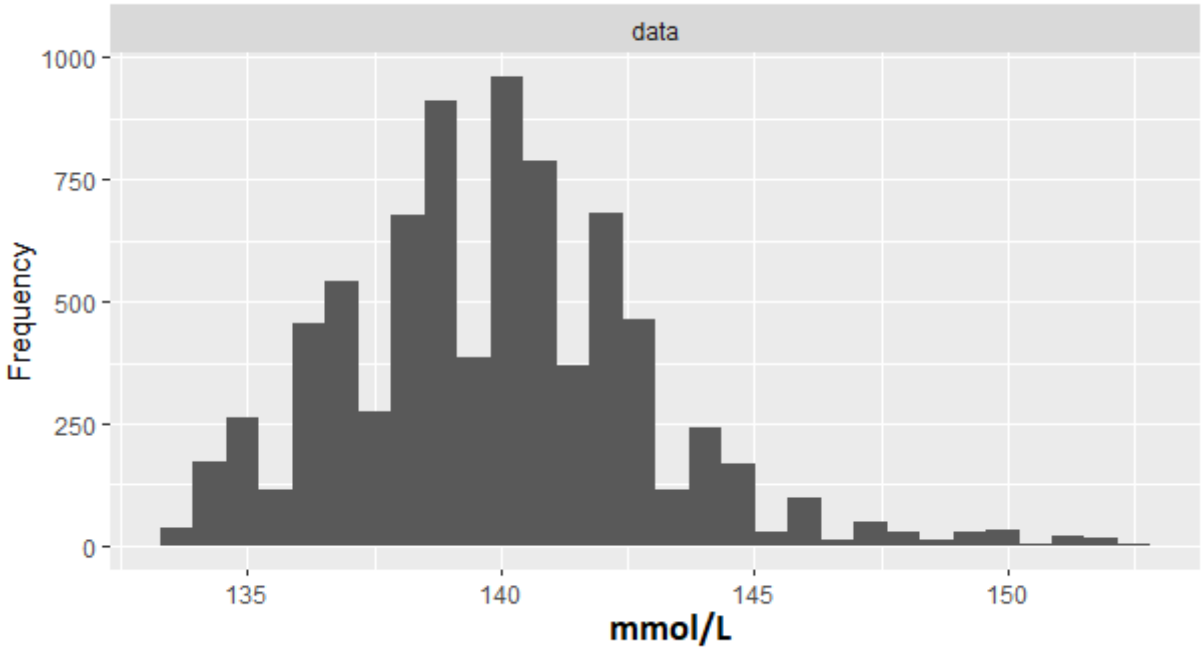

Hematocrito
data
Frequency
600
400
200
0
30
40
50
60
%


INR
data
Frequency
1200
800
400
0
0.75
1.00
1.25
1.50
1.75
INR

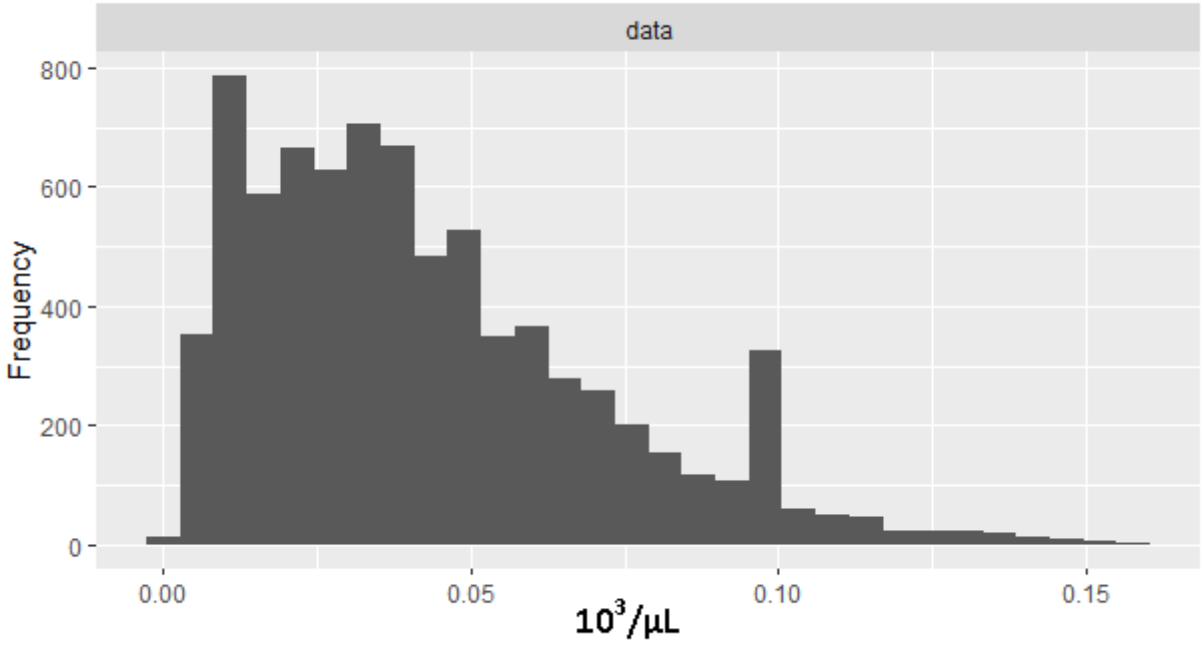

Potasio
data
Frequency
600
400
200
0
4
5
6
mmol/L

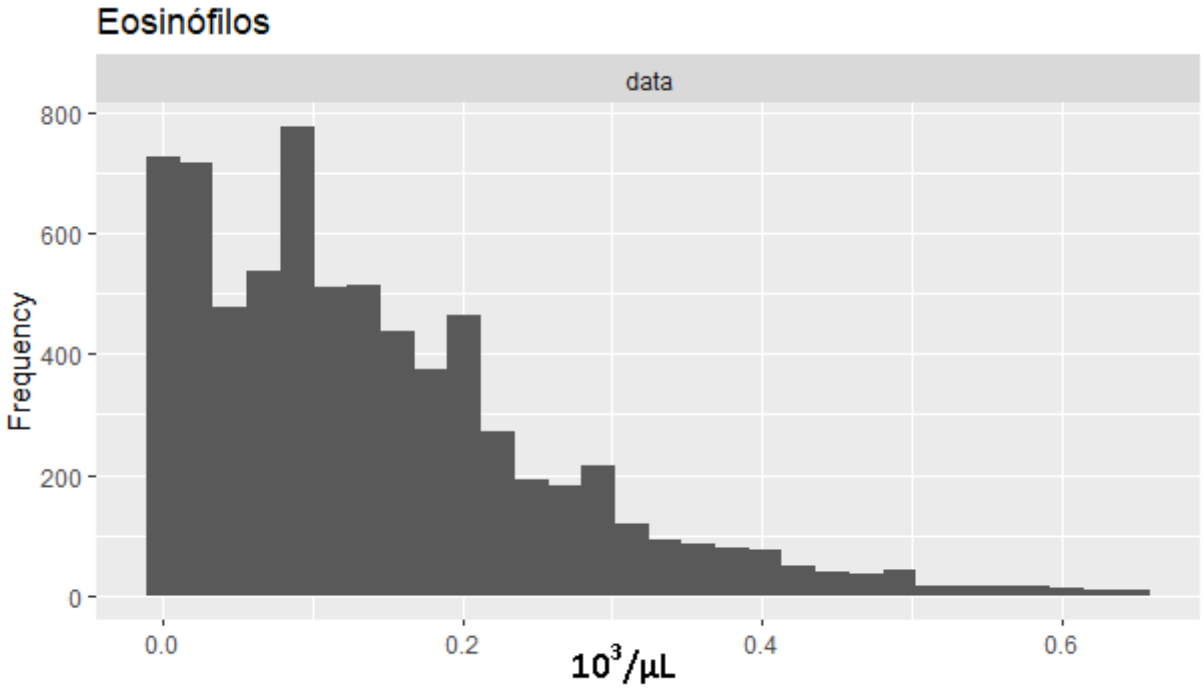

PT
data
Frequency
1250
1000
750
500
250
0
12
15
18
seg

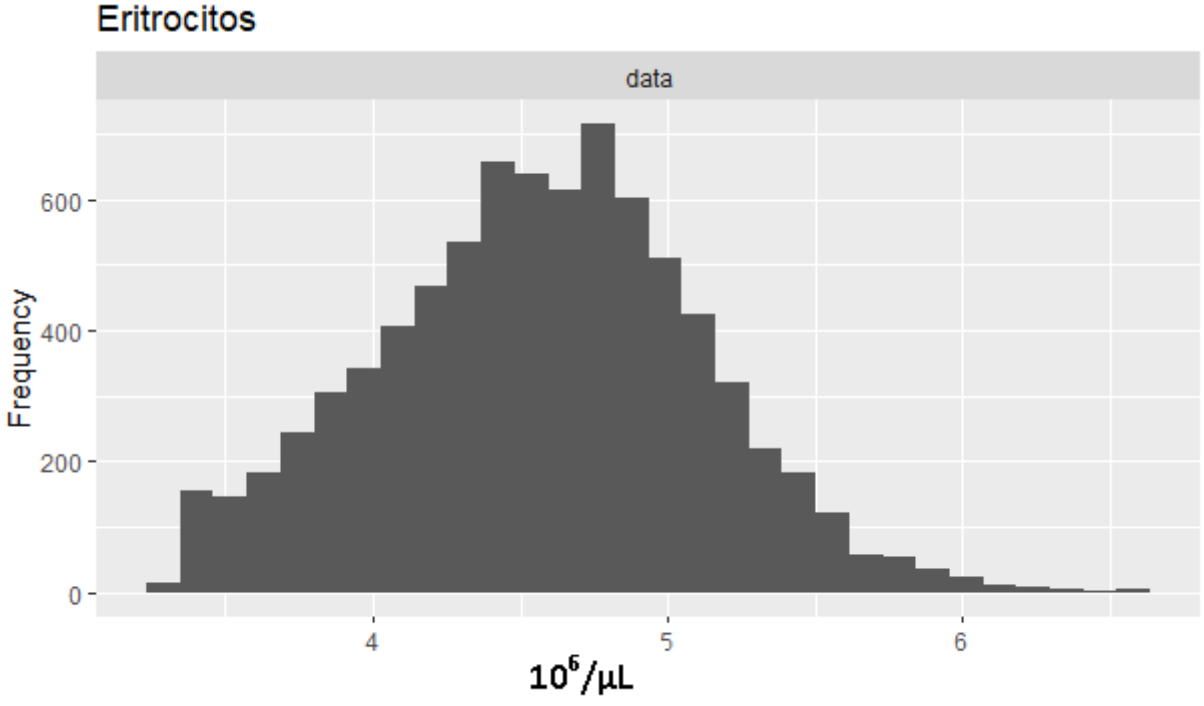
Sodio

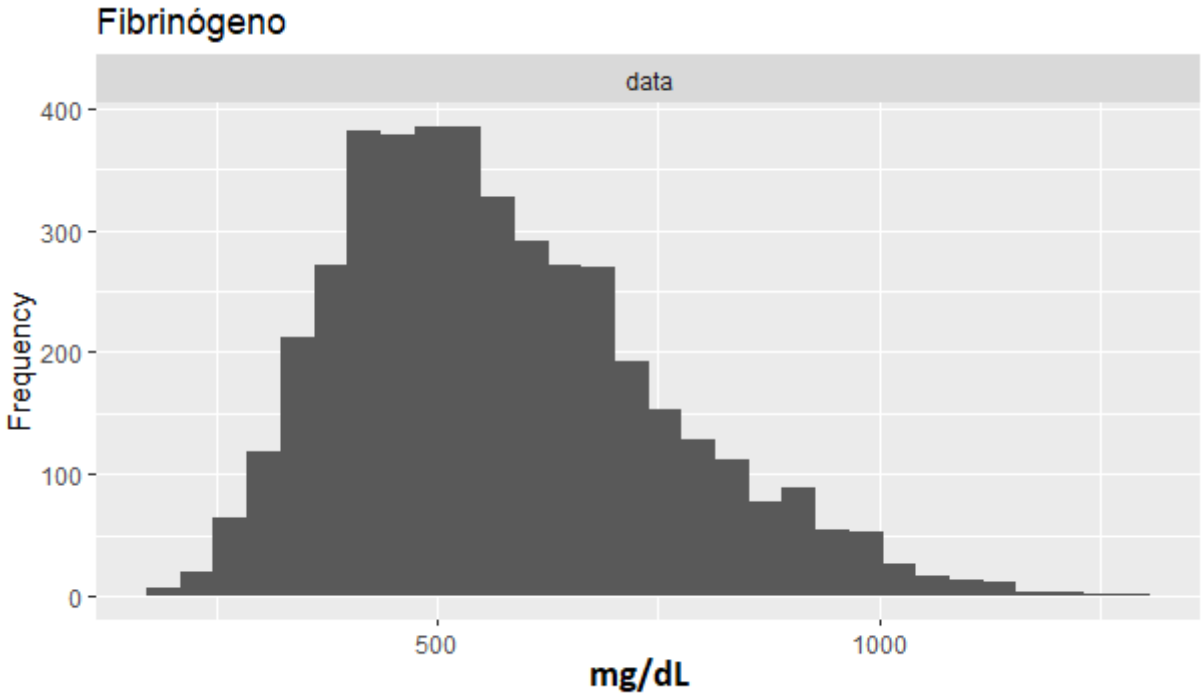


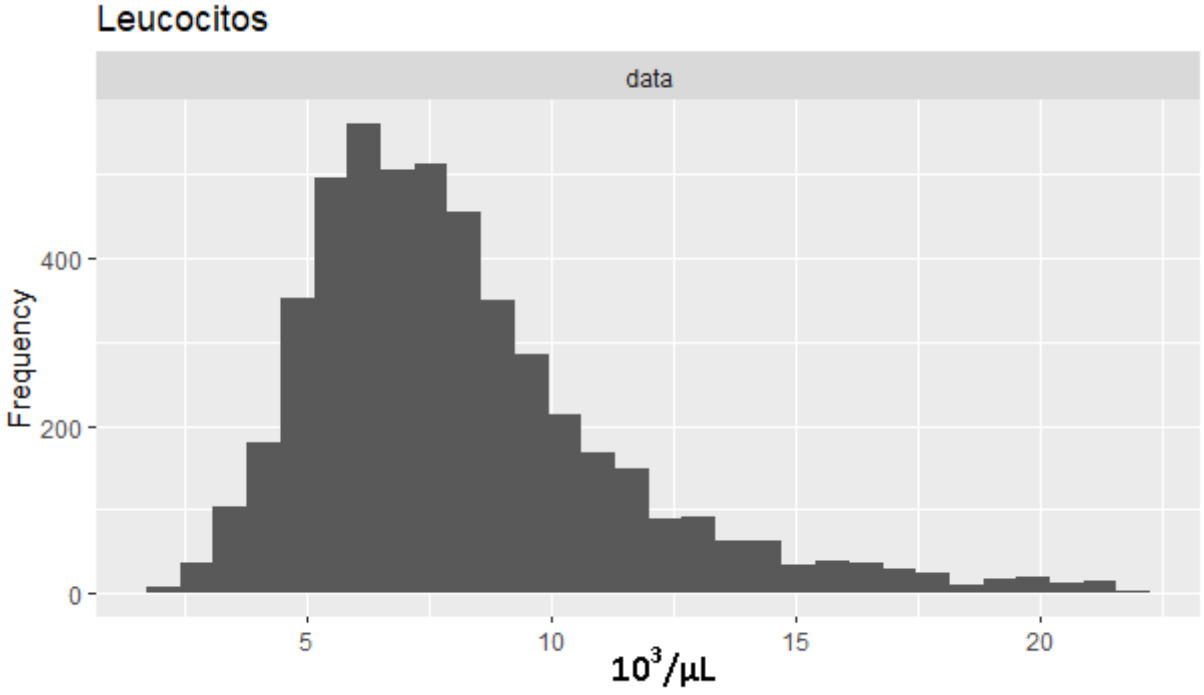
Basófilos

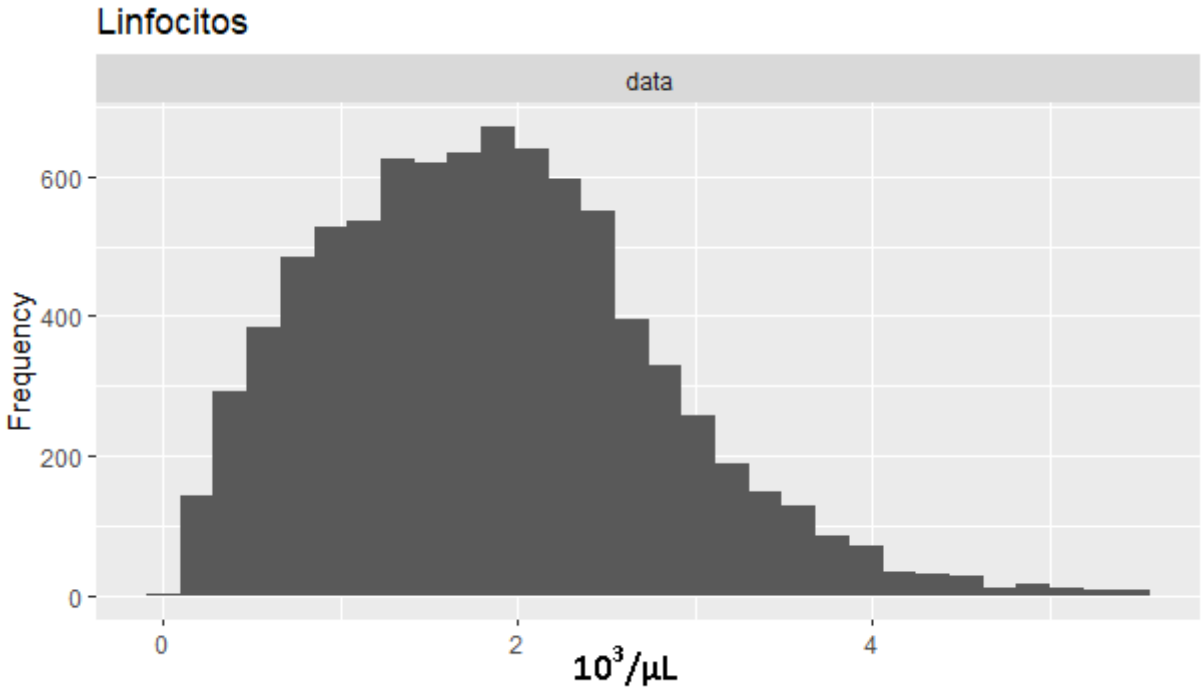


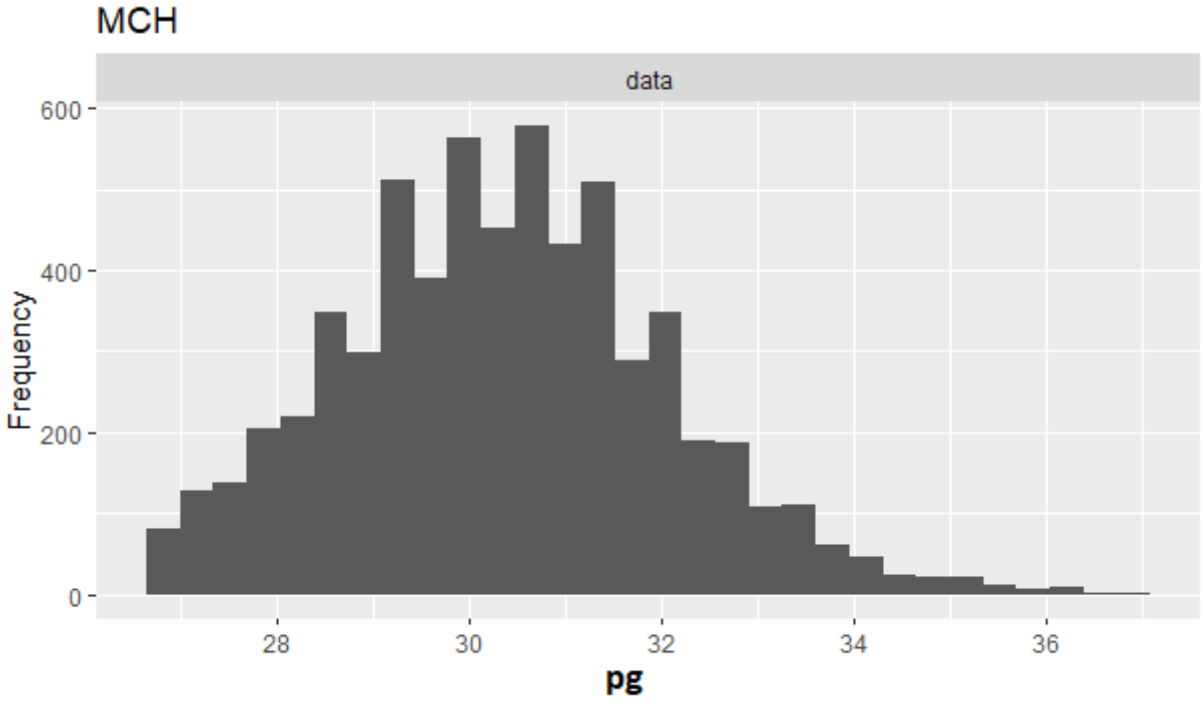

Eosinófilos
data
Frequency
800
600
400
200
0
0.0
0.2
0.4
0.6
10³/µL

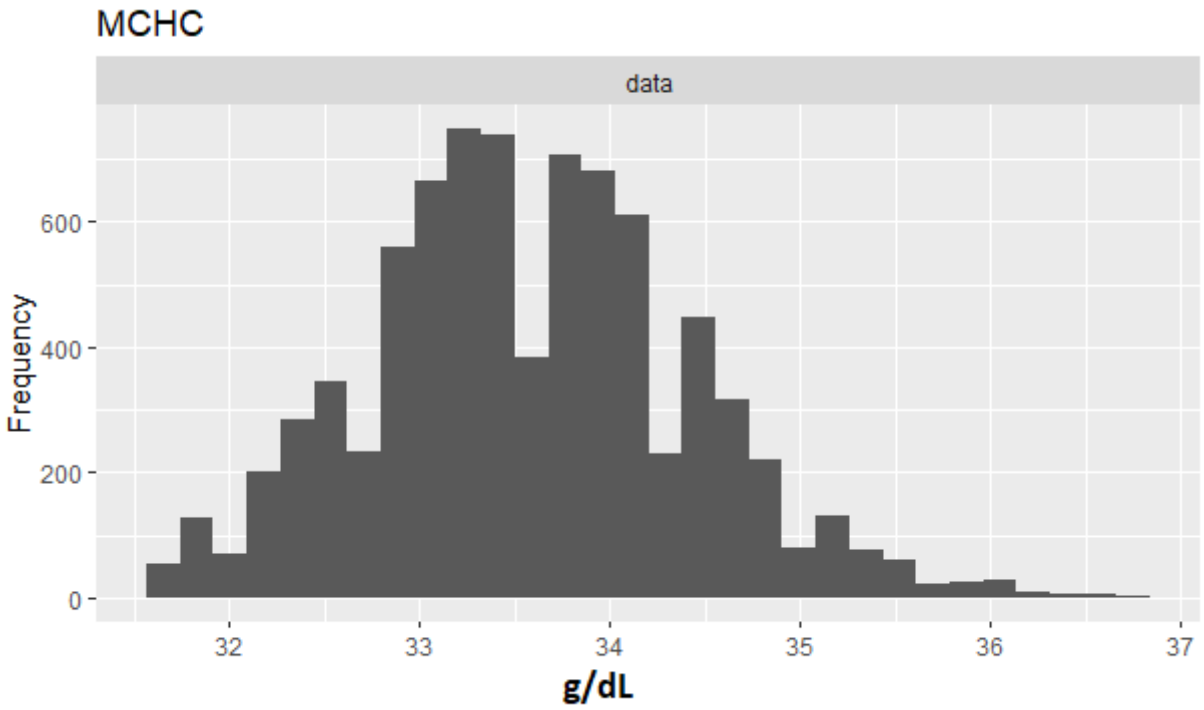

Eritrocitos
data
Frequency
600
400
200
0
4
5
6
10⁶/µL

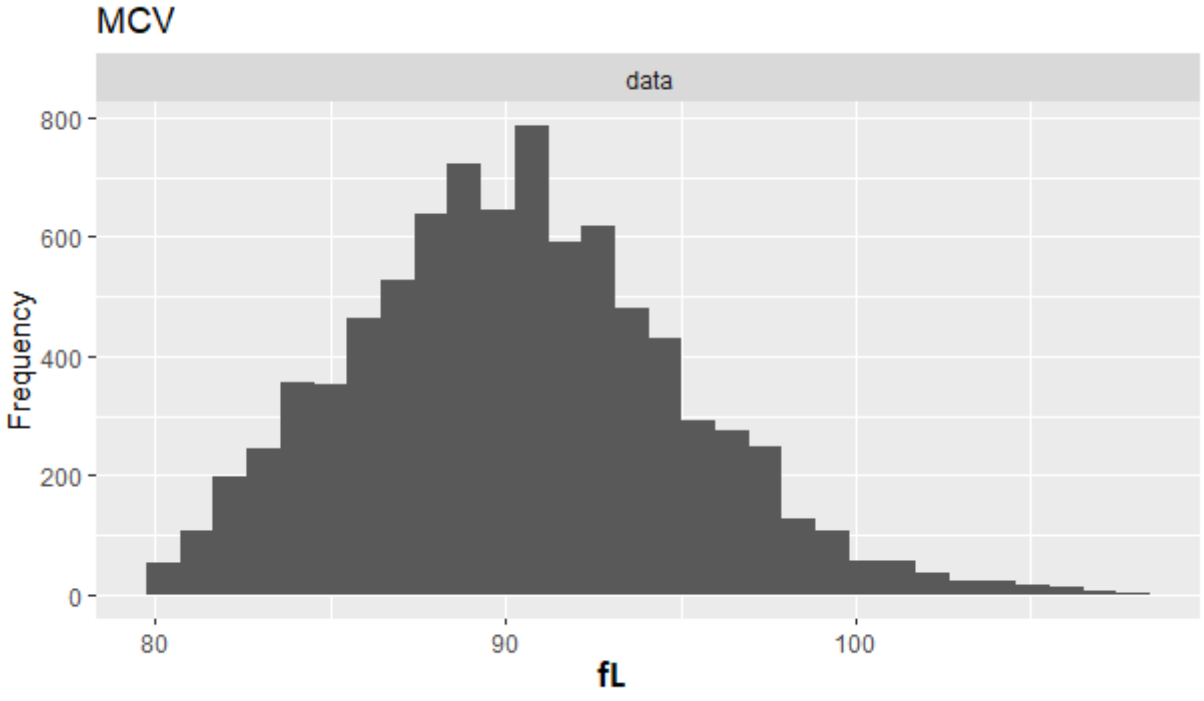

Fibrinógeno
data
Frequency
400
300
200
100
0
500
1000
mg/dL

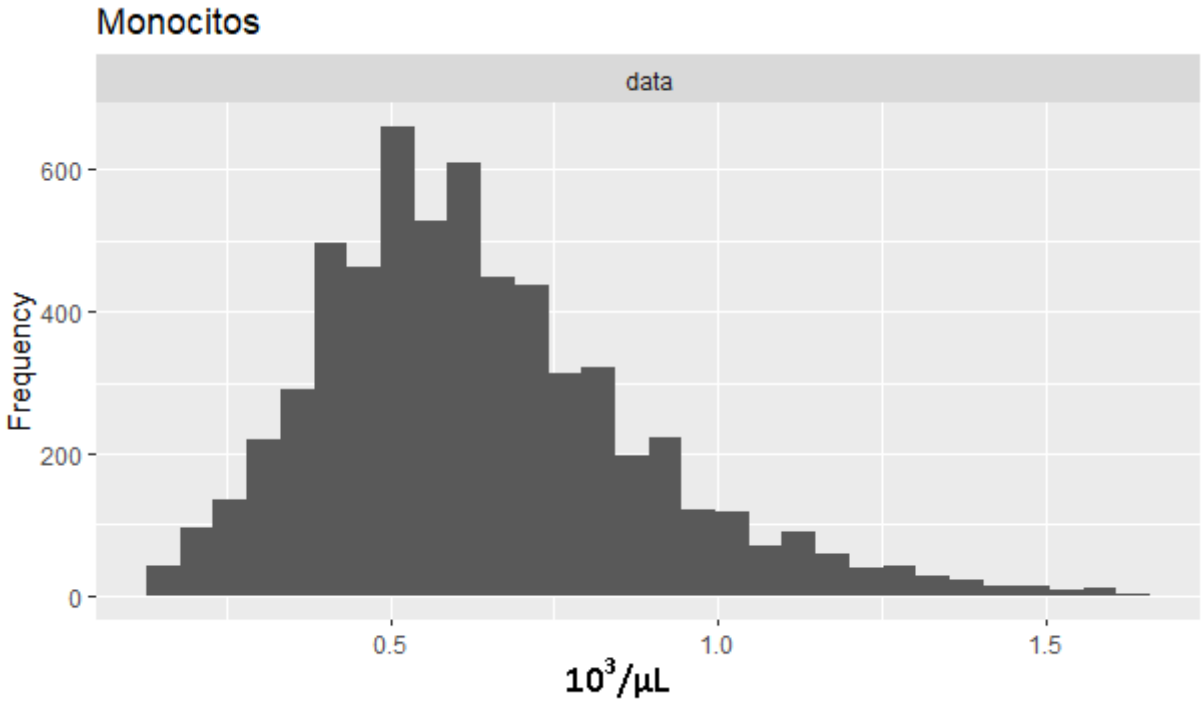

Leucocitos
data
Frequency
400
200
0
5
10
15
20
10³/µL

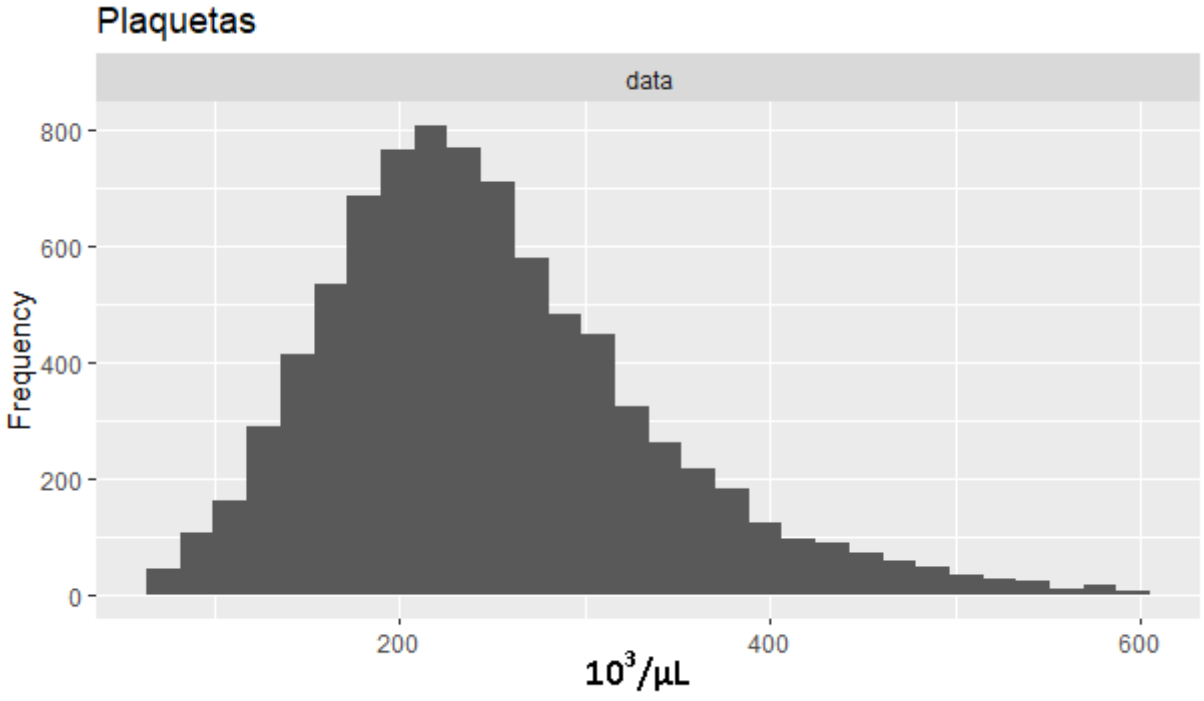

Linfocitos
data
Frequency
600
400
200
0
0
2
4
10³/µL


MCH
data
Frequency
600
400
200
0
28
30
32
34
36
pg


MCHC
data
Frequency
600
400
200
0
32
33
34
35
36
37
g/dL


MCV
data
Frequency
800
600
400
200
0
80
90
100
fL


Monocitos
data
Frequency
600
400
200
0
0.5
1.0
1.5
10³/µL


Plaquetas
data
Frequency
800
600
400
200
0
200
400
600
10³/µL