R Notebook

Conjunto de datos de diágnostico de cáncer de mama en Wisconsin

1.1 Importación y Carga de Datos

Para comenzar nuestro análisis es necesario realizar la carga de los datos:

```
data <- read.csv("data/data.csv")

data
```

id <int>	diagnosis <chr>	radius_mean <dbl>	texture_mean <dbl>	perimeter_mean <dbl>	area_mean <dbl>	smoothness_mean <dbl>
842302	M	17.990	10.38	122.80	1001.0	0.11840
842517	M	20.570	17.77	132.90	1326.0	0.08474
84300903	M	19.690	21.25	130.00	1203.0	0.10960
84348301	M	11.420	20.38	77.58	386.1	0.14250
84358402	M	20.290	14.34	135.10	1297.0	0.10030
843786	M	12.450	15.70	82.57	477.1	0.12780
844359	M	18.250	19.98	119.60	1040.0	0.09463
84458202	M	13.710	20.83	90.20	577.9	0.11890
844981	M	13.000	21.82	87.50	519.8	0.12730
84501001	M	12.460	24.04	83.97	475.9	0.11860

1-10 of 569 rows | 1-7 of 33 columns Previous **1** 2 3 4 5 6 … 57 Next

Instalamos los paquetes que serán necesarios durante nuestro proyecto:

- Tidyverse: Para la manipulación de datos y gráficos.
- Caret: Para el preprocesamiento y modelado Lattice es requerido por Caret
- DataExplorer: Para la exploración automatizada de los datos.
- Dplyr: Proporciona una gramática de manipulación de datos.
- Ggplot2: Personalización de gráficas.
- Psych: Para análisis estadístico
- Corrplot: Visualización de matriz de correlación.

```
#install.packages("tidyverse")
#install.packages("caret")
#install.packages("DataExplorer")
#install.packages("dplyr")
#install.packages("ggplot2")
#install.packages("lattice")
#install.packages("psych")
#install.packages("corrplot")
#install.packages("ggcorrplot")

library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
## Loading required package: lattice
```

```
library(DataExplorer)
```

```
## Warning: package 'DataExplorer' was built under R version 4.4.2
```

```
library()
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

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

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

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

```
library(ggplot2)
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.1
## ✓ readr     2.1.5
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ✖ purrr::lift()   masks caret::lift()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(lattice)
library(psych)
```

```
## Warning: package 'psych' was built under R version 4.4.2
```

```
##
## Attaching package: 'psych'
##
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.4.2
```

```
## corrplot 0.95 loaded
```

```
library(ggcorrplot)
```

```
## Warning: package 'ggcorrplot' was built under R version 4.4.2
```

Hagamos una vista inicial de los datos:

```
#primeras filas de nuestro dataset:

head(data)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
	<int>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	842302	M	17.99	10.38	122.80	1001.0	0.11840
2	842517	M	20.57	17.77	132.90	1326.0	0.08474
3	84300903	M	19.69	21.25	130.00	1203.0	0.10960
4	84348301	M	11.42	20.38	77.58	386.1	0.14250
5	84358402	M	20.29	14.34	135.10	1297.0	0.10030
6	843786	M	12.45	15.70	82.57	477.1	0.12780

6 rows | 1-8 of 34 columns

```
# Dimensión de nuestro dataset:

dim(data)
```

```
## [1] 569  33
```

Nuestro dataset cuenta con 33 columnas y 569 filas.

```
# Con str mostramos la estructura de nuestro dataframe, incluyendo los tipos de nuestras variables:

str(data)
```

```
## 'data.frame':    569 obs. of  33 variables:
##  $ id                     : int  842302 842517 84300903 84348301 84358402 843786 844359 84458202 844981 84501001
...
##  $ diagnosis              : chr  "M" "M" "M" "M" ...
##  $ radius_mean            : num  18 20.6 19.7 11.4 20.3 ...
##  $ texture_mean           : num  10.4 17.8 21.2 20.4 14.3 ...
##  $ perimeter_mean         : num  122.8 132.9 130 77.6 135.1 ...
##  $ area_mean              : num  1001 1326 1203 386 1297 ...
##  $ smoothness_mean        : num  0.1184 0.0847 0.1096 0.1425 0.1003 ...
##  $ compactness_mean       : num  0.2776 0.0786 0.1599 0.2839 0.1328 ...
##  $ concavity_mean         : num  0.3001 0.0869 0.1974 0.2414 0.198 ...
##  $ concave.points_mean    : num  0.1471 0.0702 0.1279 0.1052 0.1043 ...
##  $ symmetry_mean          : num  0.242 0.181 0.207 0.26 0.181 ...
##  $ fractal_dimension_mean : num  0.0787 0.0567 0.06 0.0974 0.0588 ...
##  $ radius_se              : num  1.095 0.543 0.746 0.496 0.757 ...
##  $ texture_se             : num  0.905 0.734 0.787 1.156 0.781 ...
##  $ perimeter_se           : num  8.59 3.4 4.58 3.44 5.44 ...
##  $ area_se                : num  153.4 74.1 94 27.2 94.4 ...
##  $ smoothness_se          : num  0.0064 0.00522 0.00615 0.00911 0.01149 ...
##  $ compactness_se         : num  0.049 0.0131 0.0401 0.0746 0.0246 ...
##  $ concavity_se           : num  0.0537 0.0186 0.0383 0.0566 0.0569 ...
##  $ concave.points_se      : num  0.0159 0.0134 0.0206 0.0187 0.0188 ...
##  $ symmetry_se            : num  0.03 0.0139 0.0225 0.0596 0.0176 ...
##  $ fractal_dimension_se   : num  0.00619 0.00353 0.00457 0.00921 0.00511 ...
##  $ radius_worst           : num  25.4 25 23.6 14.9 22.5 ...
##  $ texture_worst          : num  17.3 23.4 25.5 26.5 16.7 ...
##  $ perimeter_worst        : num  184.6 158.8 152.5 98.9 152.2 ...
##  $ area_worst             : num  2019 1956 1709 568 1575 ...
##  $ smoothness_worst       : num  0.162 0.124 0.144 0.21 0.137 ...
##  $ compactness_worst      : num  0.666 0.187 0.424 0.866 0.205 ...
##  $ concavity_worst        : num  0.712 0.242 0.45 0.687 0.4 ...
##  $ concave.points_worst   : num  0.265 0.186 0.243 0.258 0.163 ...
##  $ symmetry_worst         : num  0.46 0.275 0.361 0.664 0.236 ...
##  $ fractal_dimension_worst: num  0.1189 0.089 0.0876 0.173 0.0768 ...
##  $ X                      : logi  NA NA NA NA NA NA ...
```

```
# Con Describe podemos ver un primer resumen estadístico básico:

describe(data)
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to min; returning Inf
```

```
## Warning in FUN(newX[, i], ...): no non-missing arguments to max; returning -Inf
```

	v... <int>	n <dbl>	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>
id	1	569	3.037183e+07	1.250206e+08	9.06024e+05	7.344333e+06	6.556799e+04
diagnosis*	2	569	1.372583e+00	4.839180e-01	1.00000e+00	1.341357e+00	0.000000e+00
radius_mean	3	569	1.412729e+01	3.524049e+00	1.33700e+01	1.381991e+01	2.816940e+00
texture_mean	4	569	1.928965e+01	4.301036e+00	1.88400e+01	1.903779e+01	4.166106e+00
perimeter_mean	5	569	9.196903e+01	2.429898e+01	8.62400e+01	8.974046e+01	1.884385e+01
area_mean	6	569	6.548891e+02	3.519141e+02	5.51100e+02	6.061278e+02	2.272826e+02
smoothness_mean	7	569	9.636028e-02	1.406413e-02	9.58700e-02	9.587740e-02	1.408470e-02
compactness_mean	8	569	1.043410e-01	5.281276e-02	9.26300e-02	9.808295e-02	4.837724e-02
concavity_mean	9	569	8.879932e-02	7.971981e-02	6.15400e-02	7.725070e-02	5.998600e-02
concave.points_mean	10	569	4.891915e-02	3.880284e-02	3.35000e-02	4.400013e-02	2.985956e-02

1-10 of 33 rows | 1-8 of 14 columns Previous **1** 2 3 4 Next

```
# Tipos de datos en nuestras variables:

sapply(data, class)
```

```
##                     id              diagnosis            radius_mean
##              "integer"            "character"              "numeric"
##           texture_mean         perimeter_mean              area_mean
##              "numeric"              "numeric"              "numeric"
##        smoothness_mean       compactness_mean         concavity_mean
##              "numeric"              "numeric"              "numeric"
##     concave.points_mean          symmetry_mean fractal_dimension_mean
##              "numeric"              "numeric"              "numeric"
##                radius_se            texture_se          perimeter_se
##              "numeric"              "numeric"              "numeric"
##                 area_se         smoothness_se         compactness_se
##              "numeric"              "numeric"              "numeric"
##           concavity_se      concave.points_se           symmetry_se
##              "numeric"              "numeric"              "numeric"
##    fractal_dimension_se           radius_worst          texture_worst
##              "numeric"              "numeric"              "numeric"
##         perimeter_worst             area_worst        smoothness_worst
##              "numeric"              "numeric"              "numeric"
##       compactness_worst         concavity_worst    concave.points_worst
##              "numeric"              "numeric"              "numeric"
##          symmetry_worst fractal_dimension_worst                      X
##              "numeric"              "numeric"              "logical"
```

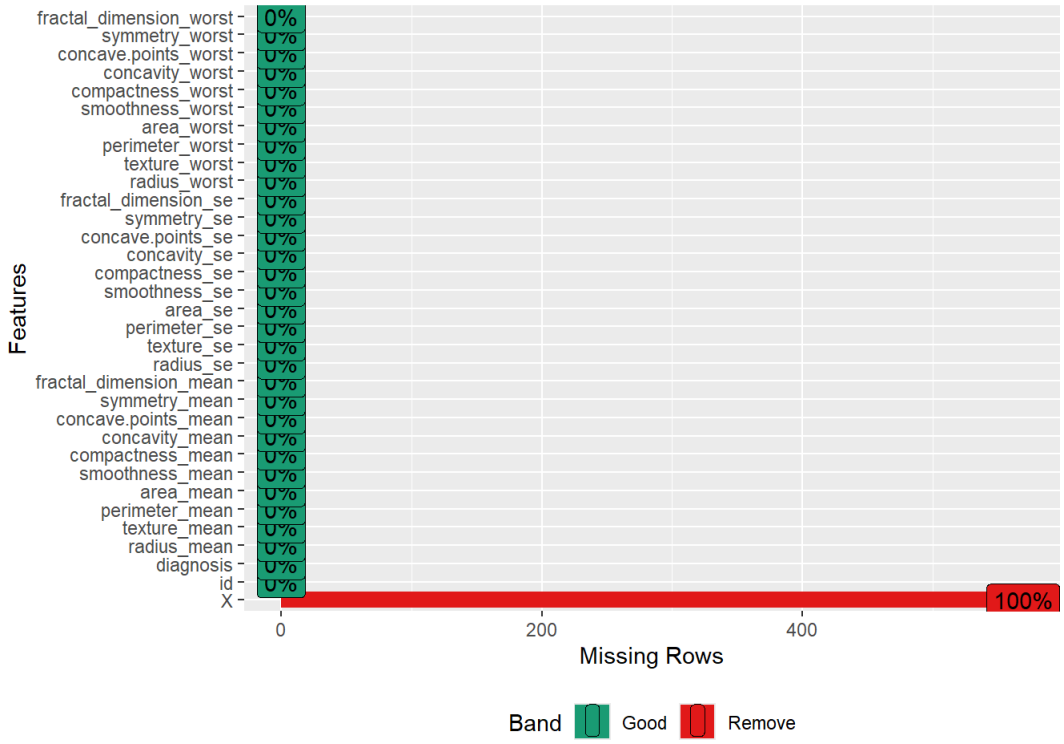
```
# Veamos si existen valores faltantes en nuestros datos:

anyNA(data)
```

```
## [1] TRUE
```

```
#Contamos el numero de valores faltantes por columna:

colSums(is.na(data))
```

```
##                      id               diagnosis             radius_mean
##                       0                       0                       0
##            texture_mean          perimeter_mean               area_mean
##                       0                       0                       0
##         smoothness_mean        compactness_mean         concavity_mean
##                       0                       0                       0
##     concave.points_mean          symmetry_mean  fractal_dimension_mean
##                       0                       0                       0
##               radius_se             texture_se           perimeter_se
##                       0                       0                       0
##                 area_se          smoothness_se         compactness_se
##                       0                       0                       0
##            concavity_se       concave.points_se            symmetry_se
##                       0                       0                       0
##    fractal_dimension_se            radius_worst           texture_worst
##                       0                       0                       0
##         perimeter_worst              area_worst        smoothness_worst
##                       0                       0                       0
##       compactness_worst         concavity_worst    concave.points_worst
##                       0                       0                       0
##          symmetry_worst fractal_dimension_worst                       X
##                       0                       0                     569
```

```
plot_missing(data)
```



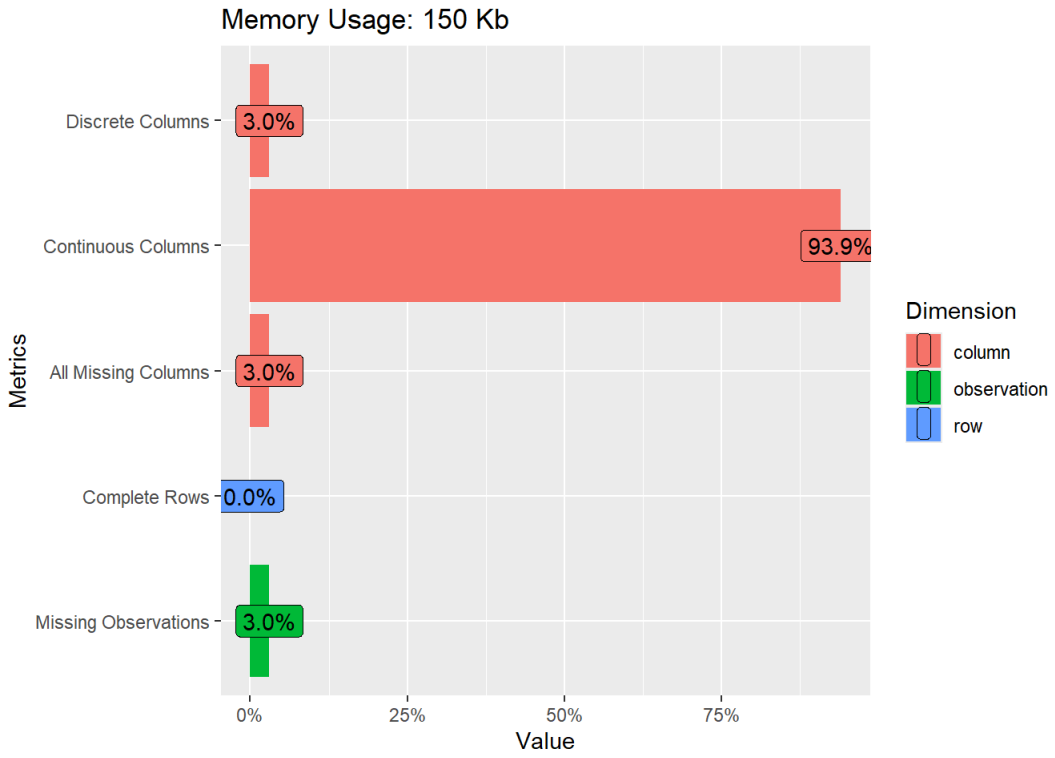
Como puede observarse, contamos con 569 valores faltantes en la última columna “X”. Más adelante veremos como tratarlo.

1.2 Análisis exploratorio de los datos

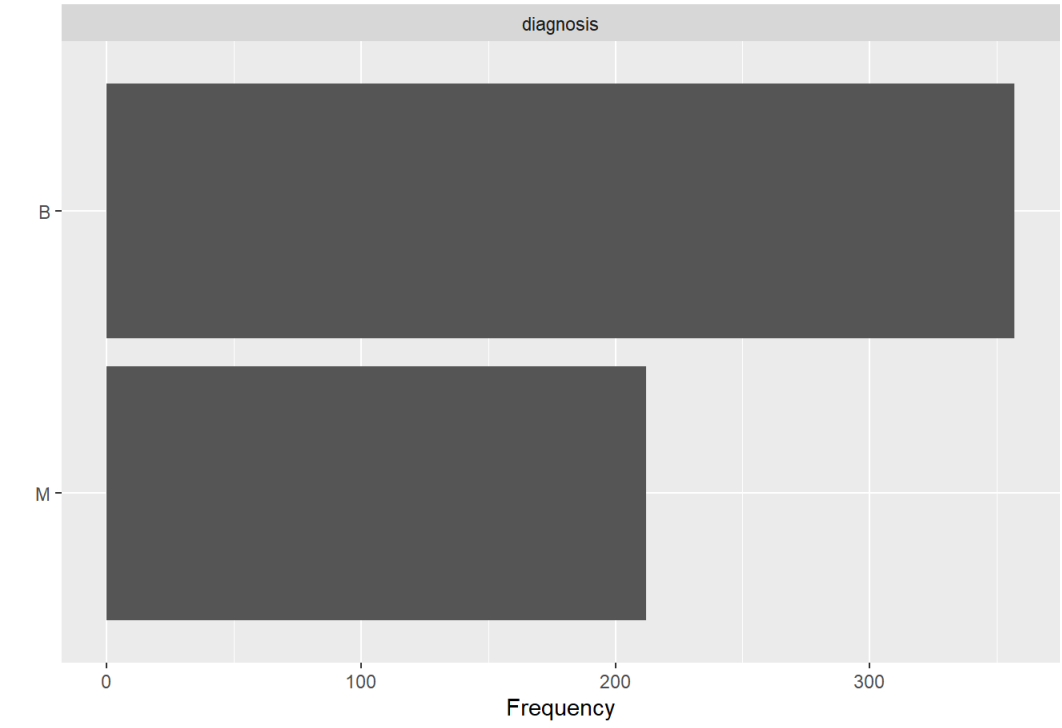
En este paso nuestro objetivo será entender la distribución y relaciones de variables.

Visualización de distribuciones y correlaciones:

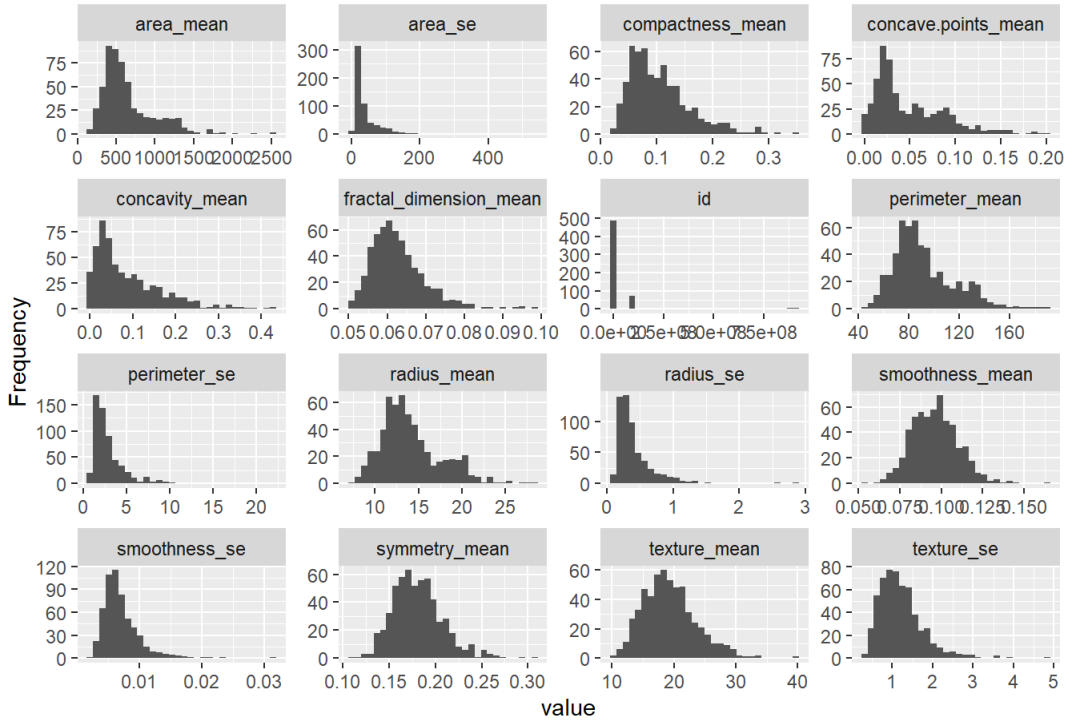
```
#Veamos un resumen gráfico general:

plot_intro(data)
```



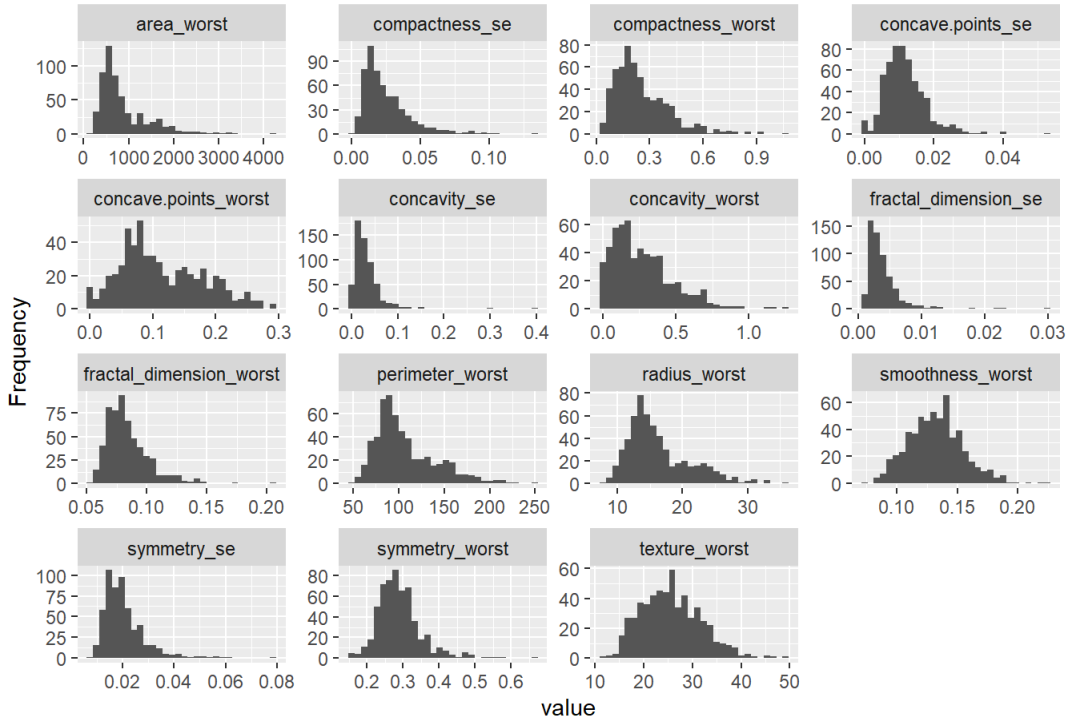

```
#Variables Categóricas:

plot_bar(data)
```




```
#Variables Numéricas
plot_histogram(data)
```



Page 1



Page 2

1.3 Preprocesamiento de los datos.

Eliminación de valores faltantes:

Para comenzar, ya sabemos que existe una columna cuyos valores son todos NA, es decir, faltantes. El primero paso en nuestro preprocesamiento será eliminar esta columna “x”:

```
data <- read.csv("data/data.csv")
data
```

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
<int>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
842302	M	17.990	10.38	122.80	1001.0	0.11840
842517	M	20.570	17.77	132.90	1326.0	0.08474
84300903	M	19.690	21.25	130.00	1203.0	0.10960

id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
<int>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
84348301	M	11.420	20.38	77.58	386.1	0.14250
84358402	M	20.290	14.34	135.10	1297.0	0.10030
843786	M	12.450	15.70	82.57	477.1	0.12780
844359	M	18.250	19.98	119.60	1040.0	0.09463
84458202	M	13.710	20.83	90.20	577.9	0.11890
844981	M	13.000	21.82	87.50	519.8	0.12730
84501001	M	12.460	24.04	83.97	475.9	0.11860

1-10 of 569 rows | 1-7 of 33 columns Previous **1** 2 3 4 5 6 … 57 Next

```
head(data)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean
	<int>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	842302	M	17.99	10.38	122.80	1001.0	0.11840
2	842517	M	20.57	17.77	132.90	1326.0	0.08474
3	84300903	M	19.69	21.25	130.00	1203.0	0.10960
4	84348301	M	11.42	20.38	77.58	386.1	0.14250
5	84358402	M	20.29	14.34	135.10	1297.0	0.10030
6	843786	M	12.45	15.70	82.57	477.1	0.12780

6 rows | 1-8 of 34 columns

```
data <- data %>% select(-X)
```

Veamos si se ha borrado correctamente la columna X y si ahora existe algún otro valor faltante:

```
colnames(data)
```

```
##  [1] "id"                    "diagnosis"
##  [3] "radius_mean"           "texture_mean"
##  [5] "perimeter_mean"        "area_mean"
##  [7] "smoothness_mean"       "compactness_mean"
##  [9] "concavity_mean"        "concave.points_mean"
## [11] "symmetry_mean"         "fractal_dimension_mean"
## [13] "radius_se"             "texture_se"
## [15] "perimeter_se"          "area_se"
## [17] "smoothness_se"         "compactness_se"
## [19] "concavity_se"          "concave.points_se"
## [21] "symmetry_se"           "fractal_dimension_se"
## [23] "radius_worst"          "texture_worst"
## [25] "perimeter_worst"       "area_worst"
## [27] "smoothness_worst"      "compactness_worst"
## [29] "concavity_worst"       "concave.points_worst"
## [31] "symmetry_worst"        "fractal_dimension_worst"
```

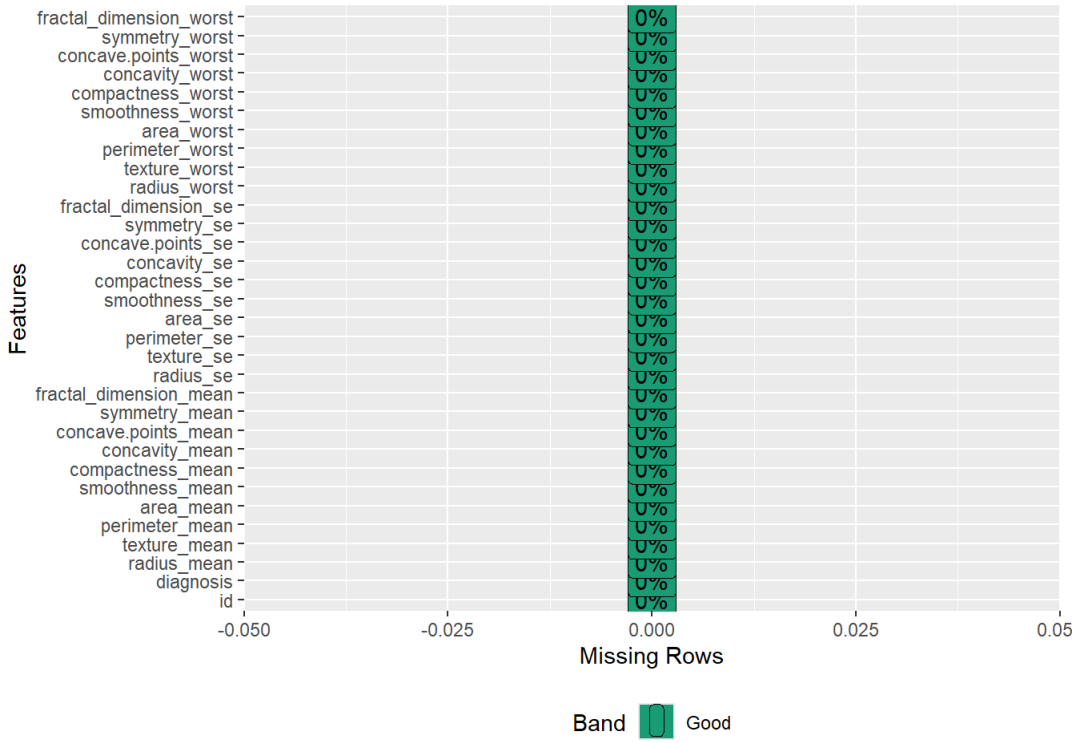
```
# Veamos si existen valores faltantes en nuestros datos:

anyNA(data)
```

```
## [1] FALSE
```

```
#Contamos el numero de valores faltantes por columna:

colSums(is.na(data))
```

```
##                      id               diagnosis             radius_mean
##                       0                       0                       0
##            texture_mean          perimeter_mean               area_mean
##                       0                       0                       0
##         smoothness_mean        compactness_mean         concavity_mean
##                       0                       0                       0
##     concave.points_mean          symmetry_mean  fractal_dimension_mean
##                       0                       0                       0
##               radius_se             texture_se           perimeter_se
##                       0                       0                       0
##                 area_se          smoothness_se         compactness_se
##                       0                       0                       0
##            concavity_se       concave.points_se            symmetry_se
##                       0                       0                       0
##    fractal_dimension_se            radius_worst           texture_worst
##                       0                       0                       0
##         perimeter_worst              area_worst        smoothness_worst
##                       0                       0                       0
##       compactness_worst         concavity_worst    concave.points_worst
##                       0                       0                       0
##          symmetry_worst fractal_dimension_worst
##                       0                       0
```

```
plot_missing(data)
```



Además de la columna con valores Faltantes, también tenemos una columna “ID” que no nos aporta ninguna información por lo que también procederemos a eliminarla:

```
data <- data %>% select(-id)
```

```
colnames(data)
```

```
##  [1] "diagnosis"               "radius_mean"
##  [3] "texture_mean"            "perimeter_mean"
##  [5] "area_mean"               "smoothness_mean"
##  [7] "compactness_mean"        "concavity_mean"
##  [9] "concave.points_mean"     "symmetry_mean"
## [11] "fractal_dimension_mean"  "radius_se"
## [13] "texture_se"              "perimeter_se"
## [15] "area_se"                 "smoothness_se"
## [17] "compactness_se"          "concavity_se"
## [19] "concave.points_se"       "symmetry_se"
## [21] "fractal_dimension_se"    "radius_worst"
## [23] "texture_worst"           "perimeter_worst"
## [25] "area_worst"              "smoothness_worst"
## [27] "compactness_worst"       "concavity_worst"
## [29] "concave.points_worst"    "symmetry_worst"
## [31] "fractal_dimension_worst"
```

```
head(data)
```

diagnosis <chr>	radius_mean <dbl>	texture_mean <dbl>	perimeter_mean <dbl>	area_mean <dbl>	smoothness_mean <dbl>	compactness_mean <dbl>
1 M	17.99	10.38	122.80	1001.0	0.11840	0.27760
2 M	20.57	17.77	132.90	1326.0	0.08474	0.07864
3 M	19.69	21.25	130.00	1203.0	0.10960	0.15990
4 M	11.42	20.38	77.58	386.1	0.14250	0.28390
5 M	20.29	14.34	135.10	1297.0	0.10030	0.13280
6 M	12.45	15.70	82.57	477.1	0.12780	0.17000

6 rows | 1-8 of 32 columns

Como podemos observar, se ha eliminado la columna “id” y se ha verificado que no existen mas valores faltantes exceptos los ya eliminados en “X”.

Después de ellos contamos con un dataset de:

```
dim(data)
```

```
## [1] 569  31
```

569 filas y 31 columnas.

Codificación de variables Categóricas

Anteriormente vimos que nuestro dataset cuenta con una única columna de valores categóricos. “Diagnosis” cuyos valores tienen el siguiente significado: + M (malignant) + B (benign)

El siguiente paso en el preprocesado de datos será pasar esta columna a numérica. Como solo se presentan dos posibles valores (M y B), se aplicará Codificación Binaria: El valor “M” pasará a ser 1 y valor “B” pasará a ser 0.

```
#M -> 1; B -> 0
data$diagnosis <- ifelse(data$diagnosis == "M", 1, 0)
```

Vamos a verificar que se ha realizado correctamente la conversión:

```
table(data$diagnosis)
```

```
##
##   0   1
## 357 212
```

```
print(data$diagnosis)
```

```
##   [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [38] 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 0 1 1 0 0 0 0 1 0 1 1
##  [75] 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 0 0 1 0 0
## [112] 0 0 0 0 0 0 0 1 1 1 0 1 1 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0
## [149] 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 0 1 1 0 0 0 0 1 0 0 1 1 1 0 1
## [186] 0 1 0 0 0 1 0 0 1 1 0 1 1 1 1 0 1 1 1 0 1 0 1 0 0 1 0 1 1 1 1 0 0 1 1 0 0
## [223] 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1 0 1 1 1 1 1 1
## [260] 1 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0
## [297] 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0 0 0 1 1 1 0 0
## [334] 0 0 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 1
## [371] 1 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0
## [408] 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 1 0 0 0 0 0 1 0 0
## [445] 1 0 1 0 0 1 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0
## [482] 0 0 0 0 0 0 1 0 1 0 0 1 0 0 0 0 0 1 1 0 1 0 1 0 0 0 0 0 1 0 0 1 0 1 0 1 1
## [519] 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [556] 0 0 0 0 0 0 0 1 1 1 1 1 1 0
```

Nos cercioramos de que no existe ningún otro valor categórico en el dataset:

```
categorical_columns <- sapply(data, is.factor) | sapply(data, is.character)
names(data)[categorical_columns]
```

```
## character(0)
```

Efectivamente, todas nuestras columnas ahora son numéricas, lo que nos da paso al siguiente punto en nuestro preprocesamiento de datos.

Estudio de correlación.

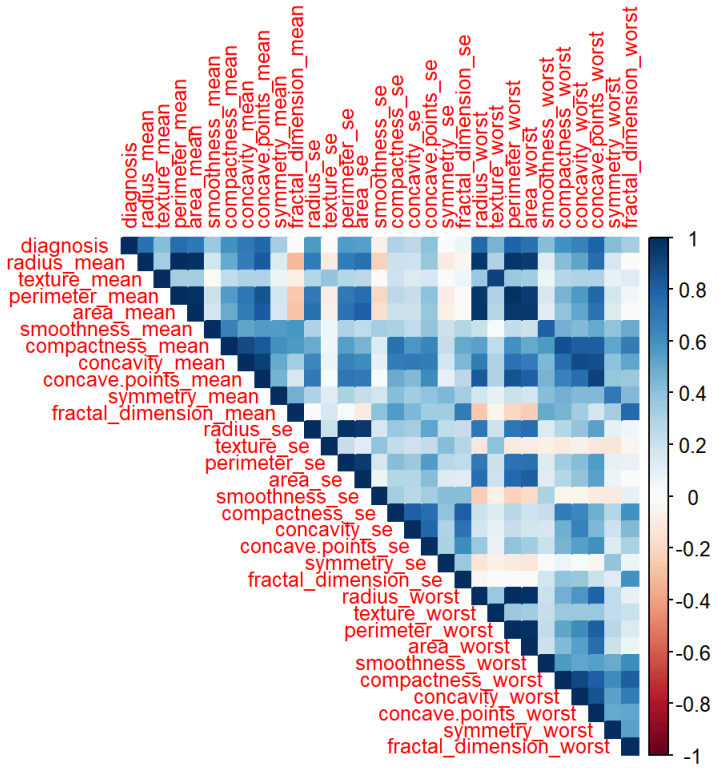
Al tener nuestro dataset limpio de NA y solo presentes variables numéricas, el siguiente paso será estudiar las posibles correlaciones de nuestro dataset.

Estudiar la correlación de los datos ayuda a identificar patrones y relaciones entre variables, lo que podría conducir a nuevas hipótesis y descubrimientos. Además, un buen estudio de correlaciones podría ser útil para seleccionar variables relevantes y construir modelos en un futuro.

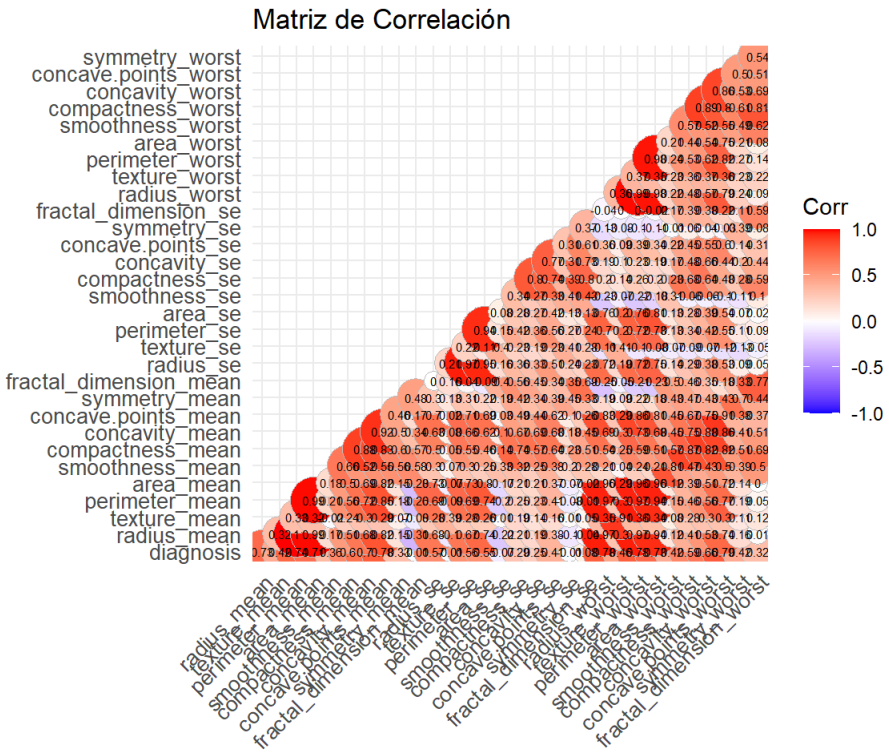
Los valores que obtendremos tendrán la siguiente interpretación:

- Correlación cercana a 1: Relación positiva fuerte.
- Correlación cercana a -1: Relación negativa fuerte.
- Correlación cercana a 0: No hay relación lineal significativa

```
# Calcular matriz de correlación
correlation_matrix <- cor(data, use = "complete.obs")  # Ignora valores faltantes


# Graficar matriz de correlación
corrplot(correlation_matrix, method = "color", type = "upper", tl.cex = 0.8)
```




```
# Graficar la matriz de correlación con valores en las celdas
ggcorrplot(correlation_matrix,
           method = "circle",  # Utilizar círculos para representar correlaciones
           type = "lower",     # Mostrar solo la mitad inferior
           lab = TRUE,         # Añadir los valores de correlación
           lab_size = 2,       # Tamaño del texto en las celdas
           colors = c("blue", "white", "red"), # Colores para las correlaciones negativas, neutrales y positivas
           title = "Matriz de Correlación",  # Título del gráfico
           tl.cex = 10)       # Tamaño de las etiquetas
```



Como puede observarse, al tratarse un conjunto de datos con 31 variables, la matriz de correlación cuesta interpretarla.

Por ello, como “diagnosis” es nuestra variable objetivo, vamos a observar cómo se correlacionan las demás variables con ella:

```r
# Calcular la correlación entre cada variable numérica y 'diagnosis'
cor_with_target <- cor(data, data$diagnosis, use = "complete.obs")

# Crear un data frame para ver las correlaciones junto con los nombres de las variables
correlation_df <- data.frame(Variable = names(data), Correlation = cor_with_target)

# Ordenar el data frame por la columna de Correlation en orden descendente
correlation_df_sorted <- correlation_df[order(-correlation_df$Correlation), ]

# Ver las correlaciones ordenadas junto con los nombres de las variables
print(correlation_df_sorted)
```

```
##                                     Variable  Correlation
## diagnosis                           diagnosis  1.000000000
## concave.points_worst       concave.points_worst  0.793566017
## perimeter_worst                 perimeter_worst  0.782914137
## concave.points_mean         concave.points_mean  0.776613840
## radius_worst                       radius_worst  0.776453779
## perimeter_mean                   perimeter_mean  0.742635530
## area_worst                           area_worst  0.733825035
## radius_mean                         radius_mean  0.730028511
## area_mean                             area_mean  0.708983837
## concavity_mean                   concavity_mean  0.696359707
## concavity_worst                 concavity_worst  0.659610210
## compactness_mean               compactness_mean  0.596533678
## compactness_worst             compactness_worst  0.590998238
## radius_se                             radius_se  0.567133821
## perimeter_se                       perimeter_se  0.556140703
## area_se                                 area_se  0.548235940
## texture_worst                     texture_worst  0.456902821
## smoothness_worst                smoothness_worst  0.421464861
## symmetry_worst                    symmetry_worst  0.416294311
## texture_mean                        texture_mean  0.415185300
## concave.points_se              concave.points_se  0.408042333
## smoothness_mean                  smoothness_mean  0.358559965
## symmetry_mean                      symmetry_mean  0.330498554
## fractal_dimension_worst fractal_dimension_worst  0.323872189
## compactness_se                    compactness_se  0.292999244
## concavity_se                        concavity_se  0.253729766
## fractal_dimension_se       fractal_dimension_se  0.077972417
## symmetry_se                         symmetry_se -0.006521756
## texture_se                           texture_se -0.008303333
## fractal_dimension_mean   fractal_dimension_mean -0.012837603
## smoothness_se                       smoothness_se -0.067016011
```

En esta tabla podemos ver la correlación de cada variable con “diagnosis” en orden descendente.

Selección de Atributos:

AQUI DEBEMOS ESTUDIAR ELIMINAR VARIABLES DE MUY ALTA CORRELACIÓN YA QUE PODRÍAN SER REDUNDANTES (>0.9) ADEMÁS TAMBIÉN AQUELLAS DE MUY BAJA CORRELACIÓN, PERO TENIENDO EN CUENTA QUE:

CORRELACION <0.3 PUEDEN ELIMINARSE SI TENEMOS MUCHAS 0.3<C<0.5 PUEDEN SER RELEVANTES Y ÚTILES PARA ALGUNOS MODELOS ~0 NORMALMENTE NO AGREGAN VALOR PREDICTIVO

IDEA:

Como no es del todo necesario eliminar variables yo lo que haría es:

-Trabajar con todas -Guardar en otro df data_preprocessed <- data %>% select(-las variables que consideremos) -Hacer lo mismo con ese nuedo df y ver que conjunto funciona mejor.