

Trabajo introducción a la ciencia de datos

Francisco Pertíñez Perea

Tabla de contenidos

1 Dataset WIZMIR	3
2 Análisis exploratorio de datos	3
2.1 Definición del problema	3
2.1.1 Planteamiento de hipótesis	4
2.2 Preparación de los datos	4
2.2.1 Estructura del dataset	4
2.2.2 Eliminación de variables	5
2.2.3 Instancias duplicadas	5
2.2.4 Valores faltantes	6
2.3 Estadística descriptiva:	6
2.3.1 Análisis de tendencia central	6
2.3.2 Análisis de dispersión	7
2.3.3 Análisis de Skewness y Kurtosis	9
2.3.4 Normalidad de los datos	14
2.4 Visualización de datos	18
2.4.1 Exploración de distribuciones:	19
2.4.2 Exploración de relaciones entre variables	32
2.5 Ingeniería de características.	40
2.5.1 Creación de variables:	40
2.5.2 Transformación de variables	44
2.6 Conclusiones finales	58
2.6.1 Comprobación de las hipótesis	58
2.6.2 Resumen del conjunto de datos	58
3 Regresión	58
3.1 Modelos lineales simples	58
3.2 Modelos lineales múltiples	66
3.2.1 Sin interacción entre variables	66
3.2.2 Con interacción entre variables y transformaciones	73

3.3	KNN en problemas de regresión	75
3.4	Comparativa de algoritmos	77
3.4.1	Comparativas por pares de LM y KNN (test de Wilcoxon)	77
3.4.2	Comparativa general entre distintos algoritmos	79
4	Dataset VOWEL	80
5	Análisis exploratorio de datos	80
5.1	Definición del problema	80
5.1.1	Planteamiento de hipótesis	80
5.2	Preparación de los datos	81
5.2.1	Estructura del dataset	81
5.2.2	Eliminación de variables	82
5.2.3	Instancias duplicadas	83
5.2.4	Valores faltantes	83
5.3	Estadística descriptiva:	83
5.3.1	Análisis de tendencia central (variables numéricas)	83
5.3.2	Análisis de dispersión (variables numéricas)	84
5.3.3	Análisis de Skewness y Kurtosis	86
5.3.4	Normalidad de los datos	91
5.3.5	Tablas de contingencia (variables categóricas)	95
5.4	Visualización de datos	96
5.4.1	Exploración de relaciones entre variables	111
5.4.2	Transformación de variables	117
5.5	Conclusiones finales	117
5.5.1	Comprobación de las hipótesis	117
5.5.2	Resumen del conjunto de datos	134
6	Clasificación	134
6.1	LDA	138
6.2	QDA	141
6.3	Comparación de resultados:	142

1 Dataset WIZMIR

2 Análisis exploratorio de datos

2.1 Definición del problema

Lo primero que vamos a hacer es buscar información sobre el dataset y el problemas que debemos resolver sobre este.

Buscando información en la web, se ha encontrado en la página <https://sci2s.ugr.es/keel/dataset.php?cod=78> una descripción sobre el problema:

- **Descripción del dataset:** Conjunto de datos en el que cada instancia representa la información del tiempo de Izmir durante un determinado día (desde 01/01/1994 hasta 31/12/1997).
- **Objetivo:** predecir la temperatura media que hubo cada día.

En base al objetivo descrito sabemos que nos encontramos ante un *Problema de Regresión*.

Leyendo el fichero del dataset podemos obtener un listado con las variables que lo componen, estas son:

Regresores / variables independientes

- Max_temperature (real): máxima temperatura que hubo ese día.
- Min_temperature (real): mínima temperatura que ha hubo ese día.
- Dewpoint (real): corresponde con el punto de rocío, este es la temperatura la que, durante un proceso de enfriamiento, empieza a condensarse el vapor de agua contenido en el aire, produciendo rocío.
- Precipitation (real): variable relacionada con la cantidad de lluvia que habido durante ese día.
- Sea_level pressure (real): presión a nivel del mar.
- Standard_pressure (real): presión estándar.
- Visibility (real): variable relacionada con el nivel de visibilidad que hubo ese día.
- Wind_speed (real): variable relacionada con velocidad del viento que hubo ese día
- Max_wind_speed (real): máxima velocidad del viento que hubo ese día

Variable objetivo / variable dependiente

- Mean temperature (real): temperatura media que hubo ese día.

2.1.1 Planteamiento de hipótesis

Con toda esta información ya estamos en disposición de plantear ciertas hipótesis sobre el problema:

- ¿Las variables relacionadas con la temperatura son buenas candidatas para predecir la variable objetivo?
- ¿Existe una fuerte correlación entre las variables relacionadas con la temperatura?

2.2 Preparación de los datos

En este apartado nuestro objetivo es obtener una visión clara de la estructura del dataset, para así poder limpiarlo y prepararlo para posteriores análisis gráficos o a través de estadística descriptiva.

2.2.1 Estructura del dataset

A continuación vamos a leer el conjunto de datos para hacernos una idea sobre la estructura del dataset:

```
library(tidyverse)

-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr     1.1.3     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.4     v tibble    3.2.1
v lubridate  1.9.3     v tidyr    1.3.0
v purrr     1.0.2
-- Conflicts ----- tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()   masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become non-conflicting

wizmir <- read.csv("wizmir/wizmir.dat", comment.char="@", header = FALSE)
names(wizmir) <- c("max_temperature", "min_temperature", "dewpoint",
                    "precipitation", "sea_level_pressure", "standard_pressure",
                    "visibility", "wind_speed", "max_wind_speed", "mean_temperature")

str(wizmir)
```

```
'data.frame': 1461 obs. of 10 variables:
$ max_temperature : num 88.2 88 91.6 64.4 94.1 ...
$ min_temperature : num 57.2 58.6 62.1 42.8 72.3 ...
$ dewpoint       : num 53.6 54.9 60.4 37.4 46.8 ...
$ precipitation   : num 0 0 0 0.2 0 0 0.2 0 0 0 ...
$ sea_level_pressure: num 30 29.8 29.8 30.1 29.9 ...
$ standard_pressure : num 7.3 7.3 7.2 7.8 7.2 6.8 7.9 7.6 7.7 7.3 ...
$ visibility      : num 9.09 10.7 8.29 21.1 17.2 21.2 17 16.1 9.09 7.37 ...
$ wind_speed       : num 16.1 18.3 18.3 27.5 25.3 27.5 25.3 20.8 16.1 20.8 ...
$ max_wind_speed   : num 34.3 34.3 34.3 34.3 34.3 ...
$ mean_temperature : num 74.3 75.2 76.1 47.1 83.9 70.8 59.7 44.9 47.8 77.8 ...
```

A partir de la función `str` podemos obtener la siguiente información:

- Efectivamente tenemos las variables descritas en el propio fichero del conjunto de datos.
- Número de instancias: 1461.
- Número de variables: 10.
- Tipado de las variables: todas son de tipo real. Además, dicho tipado es el correcto teniendo en cuenta el significado de las variables.

2.2.2 Eliminación de variables

Revisando cada una de las variables que compone el dataset para ver si existen algunas que no aporten información útil, concluimos que todas las variables contienen información que puede ser relevante a la hora de predecir la variable objetivo. Todas son mediciones sobre un determinado aspecto del tiempo que hubo en cada determinado día, por lo que la información que nos aporten puede ser muy valiosa.

2.2.3 Instancias duplicadas

Comprobemos si hay instancias duplicadas en el dataset:

```
wizmir[duplicated(wizmir), ]
```

	max_temperature	min_temperature	dewpoint	precipitation	sea_level_pressure
1289	96.8	64.2	43.3	0	29.89
	standard_pressure	visibility	wind_speed	max_wind_speed	mean_temperature
1289	7.3	18.3	25.3	34.28	86.2

Como podemos observar, tenemos únicamente una instancia duplicada. Dado que es una sola y teniendo en cuenta el tamaño del dataset, podemos eliminarla sin que haya ninguna repercusión tanto en el EDA como a la hora de aplicar regresión.

```
wizmir <- unique(wizmir)
```

2.2.4 Valores faltantes

Comprobemos si existen valores faltantes en el conjunto de datos:

```
wizmir %>% filter(rowSums(across(.cols = everything(), .fns = is.na)) > 0)

[1] max_temperature    min_temperature    dewpoint          precipitation
[5] sea_level_pressure standard_pressure  visibility         wind_speed
[9] max_wind_speed     mean_temperature
<0 rows> (or 0-length row.names)
```

Como podemos observar, no tenemos ningún valor faltante.

2.3 Estadística descriptiva:

En este apartado nuestro objetivo es describir el conjunto de datos desde el punto de vista de la estadística descriptiva.

2.3.1 Análisis de tendencia central

Comenzaremos nuestro análisis usando medidas de tendencia central, estas son la media y la mediana

```
means <- apply(wizmir, 2, mean)
medians <- apply(wizmir, 2, median)

central_tendencies <- data.frame(Mean = means,
                                    Median = medians)
central_tendencies
```

	Mean	Median
max_temperature	72.20732877	70.70
min_temperature	50.73102740	50.00
dewpoint	46.62583562	48.20
precipitation	0.09263014	0.00
sea_level_pressure	29.97116438	29.95
standard_pressure	7.19671233	7.30
visibility	11.15313699	10.50
wind_speed	19.80800000	19.81
max_wind_speed	34.28036986	34.28
mean_temperature	61.49068493	59.95

En general, podemos ver que para toda variable del dataset, el valor de la media no difiere mucho de el de la mediana. Este es un indicativo de que la distribución que siguen las variables son bastante simétricas, además de que no haya una alta presencia de valores anómalos. Como sabemos si hubiese ambas cosas resultaría en que los valores de la media diferiría mucho de la mediana, al ser la media muy sensible tanto a distribuciones asimétricas como valores anómalos, y la mediana robusta a esto.

2.3.2 Análisis de dispersión

A continuación, vamos a realizar un estudio de la variabilidad de los datos.

Mínimo, Máximo y Rango:

```
minimums <- apply(wizmir, 2, min)
maximums <- apply(wizmir, 2, max)

data.frame(Minimum = minimums,
           Maximum = maximums,
           Range = (maximums - minimums))
```

	Minimum	Maximum	Range
max_temperature	36.70	105.00	68.30
min_temperature	15.80	78.60	62.80
dewpoint	13.60	64.40	50.80
precipitation	0.00	7.60	7.60
sea_level_pressure	29.26	30.48	1.22
standard_pressure	2.30	10.10	7.80
visibility	0.92	29.10	28.18
wind_speed	4.72	68.80	64.08

```

max_wind_speed      16.11   55.24 39.13
mean_temperature    29.40   89.90 60.50

```

Observando los resultados, podemos darnos cuenta que las variables relacionadas con la temperatura están medidas en la escala Fareheit, ya que tanto los máximos como los mínimos de estas variables son demasiado altos como para que estén en la escala Celsius. Tambien nos damos cuenta de que los rangos de valores son muy diferentes entre variables salvo las relacionadas con la temperatura, algo lógico pues cada variable mide aspectos diferentes del tiempo.

Desviación estándar, varianza y desviación absoluta media:

```

standard_deviations <- apply(wizmir, 2, sd)
variances <- apply(wizmir, 2, var)
median_absolute_deviations <- apply(wizmir, 2, mad)

data.frame(Standard_deviation = standard_deviations,
           Variance = variances,
           Median_absolute_deviation = median_absolute_deviations)

```

	Standard_deviation	Variance	Median_absolute_deviation
max_temperature	15.9191646	253.41980232	21.349440
min_temperature	13.2259150	174.92482828	16.012080
dewpoint	9.3473412	87.37278676	8.895600
precipitation	0.3529134	0.12454784	0.000000
sea_level_pressure	0.1677330	0.02813435	0.163086
standard_pressure	0.6851826	0.46947513	0.296520
visibility	5.4052821	29.21707432	6.360354
wind_speed	7.1362480	50.92603534	5.500446
max_wind_speed	2.4426623	5.96659904	0.000000
mean_temperature	14.3666111	206.39951564	18.458370

Podemos observar que tenemos tanto variables que presentan cierta variabilidad en los datos (max_temperature, min_temperature, dewpoint, mean_temperature, wind_speed, y visibility) como variables que casi no presentan nada de variabilidad (sea_level_pressure, standard_pressure y max_wind_speed)

Teniendo en cuenta los resultados obtenidos para las distintas medidas de dispersión y los rangos de valores que tiene cada una de las variables podemos decir que en general, las medidas de tendencia central son representativas de la variable a la que corresponden.

Cuantiles:

```
Q1s <- apply(wizmir, 2, quantile, probs = c(0.25))
Q3s <- apply(wizmir, 2, quantile, probs = c(0.75))
```

```
data.frame(Q1 = Q1s,
           Q2 = Q3s,
           IQR = (Q3s - Q1s))
```

	Q1	Q2	IQR
max_temperature	59.00	86.950	27.950
min_temperature	40.10	62.125	22.025
dewpoint	41.30	53.600	12.300
precipitation	0.00	0.000	0.000
sea_level_pressure	29.85	30.080	0.230
standard_pressure	7.10	7.600	0.500
visibility	6.56	15.325	8.765
wind_speed	16.10	23.000	6.900
max_wind_speed	34.28	34.280	0.000
mean_temperature	49.60	75.200	25.600

Podemos observar que el rango intercuartílico no es muy grande teniendo en cuenta el rango de las variables, esto es un indicio de que los valores estan agrupados en la parte central de la distribución.

2.3.3 Análisis de Skewness y Kurtosis

Vamos a realizar una series de test estadísticas para analizar tanto la simetría como las kurtosis de la distribución de las distintas variables.

Skewness:

Para analizar la simetría de la distribución vamos a usar el test de Agostino.

```
library(moments)

agostino.test(wizmir$max_temperature)

D'Agostino skewness test

data: wizmir$max_temperature
skew = 0.027396, z = 0.429446, p-value = 0.6676
alternative hypothesis: data have a skewness
```

```
agostino.test(wizmir$min_temperature)
```

D'Agostino skewness test

```
data: wizmir$min_temperature  
skew = 0.032997, z = 0.517200, p-value = 0.605  
alternative hypothesis: data have a skewness
```

```
agostino.test(wizmir$dewpoint)
```

D'Agostino skewness test

```
data: wizmir$dewpoint  
skew = -0.69036, z = -9.84100, p-value < 2.2e-16  
alternative hypothesis: data have a skewness
```

```
agostino.test(wizmir$precipitation)
```

D'Agostino skewness test

```
data: wizmir$precipitation  
skew = 9.8957, z = 41.0475, p-value < 2.2e-16  
alternative hypothesis: data have a skewness
```

```
agostino.test(wizmir$sea_level_pressure)
```

D'Agostino skewness test

```
data: wizmir$sea_level_pressure  
skew = 0.30649, z = 4.70059, p-value = 2.594e-06  
alternative hypothesis: data have a skewness
```

```
agostino.test(wizmir$standard_pressure)
```

```
D'Agostino skewness test

data: wizmir$standard_pressure
skew = -2.6812, z = -24.4740, p-value < 2.2e-16
alternative hypothesis: data have a skewness

agostino.test(wizmir$visibility)

D'Agostino skewness test

data: wizmir$visibility
skew = 0.38712, z = 5.86523, p-value = 4.485e-09
alternative hypothesis: data have a skewness

agostino.test(wizmir$wind_speed)

D'Agostino skewness test

data: wizmir$wind_speed
skew = 1.6331, z = 18.5349, p-value < 2.2e-16
alternative hypothesis: data have a skewness

agostino.test(wizmir$max_wind_speed)

D'Agostino skewness test

data: wizmir$max_wind_speed
skew = 0.94377, z = 12.66276, p-value < 2.2e-16
alternative hypothesis: data have a skewness

agostino.test(wizmir$mean_temperature)

D'Agostino skewness test

data: wizmir$mean_temperature
skew = 0.070808, z = 1.108782, p-value = 0.2675
alternative hypothesis: data have a skewness
```

Observaciones:

- Variables que muestran evidencia significativa de la presencia de sesgo: dewpoint, precipitation, sea_level_pressure, standard_pressure, visibility, wind_speed, max_wind_speed.
- Variables que no muestran evidencia significativa de la presencia de sesgo: max_temperature, min_temperature, mean_temperature.

Kurtosis:

Para analizar la kurtosis vamos a usar el test de Anscombe.

```
anscombe.test(wizmir$max_temperature)
```

Anscombe-Glynn kurtosis test

```
data: wizmir$max_temperature
kurt = 1.7564, z = -41.2506, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(wizmir$min_temperature)
```

Anscombe-Glynn kurtosis test

```
data: wizmir$min_temperature
kurt = 2.027, z = -16.653, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(wizmir$dewpoint)
```

Anscombe-Glynn kurtosis test

```
data: wizmir$dewpoint
kurt = 2.977589, z = -0.083039, p-value = 0.9338
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(wizmir$precipitation)
```

```
Anscombe-Glynn kurtosis test

data: wizmir$precipitation
kurt = 164.70, z = 25.57, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3

anscombe.test(wizmir$sea_level_pressure)

Anscombe-Glynn kurtosis test

data: wizmir$sea_level_pressure
kurt = 2.95558, z = -0.26178, p-value = 0.7935
alternative hypothesis: kurtosis is not equal to 3

anscombe.test(wizmir$standard_pressure)

Anscombe-Glynn kurtosis test

data: wizmir$standard_pressure
kurt = 14.801, z = 17.064, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3

anscombe.test(wizmir$visibility)

Anscombe-Glynn kurtosis test

data: wizmir$visibility
kurt = 2.3554, z = -7.5702, p-value = 3.73e-14
alternative hypothesis: kurtosis is not equal to 3

anscombe.test(wizmir$wind_speed)

Anscombe-Glynn kurtosis test

data: wizmir$wind_speed
kurt = 10.655, z = 15.050, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(wizmir$max_wind_speed)

Anscombe-Glynn kurtosis test

data: wizmir$max_wind_speed
kurt = 25.437, z = 19.749, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(wizmir$mean_temperature)
```

```
Anscombe-Glynn kurtosis test

data: wizmir$mean_temperature
kurt = 1.7648, z = -39.5224, p-value < 2.2e-16
alternative hypothesis: kurtosis is not equal to 3
```

Observaciones:

- Variables que muestran evidencia significativa de tener kurtosis diferente de 3: max_temperature, min_temperature, precipitation, standard_pressure, visibility, wind_speed, max_wind_speed y mean_temperature.
- No hay evidencia significativa para rechazar la hipótesis nula de que la kurtosis es igual a 3: dewpoint y sea_level_pressure.

2.3.4 Normalidad de los datos

Si bien con el test D'Agostino y el test de Anscombe-Glynn podemos hacernos una idea de qué variables siguen aproximadamente una distribución normal (distribución simétrica y kurtosis de 3 son características de distribuciones normales), vamos a utilizar otros mecanismos para cerciorarnos de estos, en concreto realizaremos el test de Shapiro-Wilk's y un gráfico QQ-plot.

Test de Shapiro-Wilk's

```
shapiro.test(wizmir$max_temperature)
```

```
Shapiro-Wilk normality test

data: wizmir$max_temperature
W = 0.95292, p-value < 2.2e-16
```

```
shapiro.test(wizmir$min_temperature)
```

Shapiro-Wilk normality test

```
data: wizmir$min_temperature  
W = 0.97592, p-value = 6.515e-15
```

```
shapiro.test(wizmir$dewpoint)
```

Shapiro-Wilk normality test

```
data: wizmir$dewpoint  
W = 0.95984, p-value < 2.2e-16
```

```
shapiro.test(wizmir$precipitation)
```

Shapiro-Wilk normality test

```
data: wizmir$precipitation  
W = 0.27468, p-value < 2.2e-16
```

```
shapiro.test(wizmir$sea_level_pressure)
```

Shapiro-Wilk normality test

```
data: wizmir$sea_level_pressure  
W = 0.98819, p-value = 1.652e-09
```

```
shapiro.test(wizmir$standard_pressure)
```

Shapiro-Wilk normality test

```
data: wizmir$standard_pressure  
W = 0.73596, p-value < 2.2e-16
```

```
shapiro.test(wizmir$visibility)
```

Shapiro-Wilk normality test

```
data: wizmir$visibility  
W = 0.96969, p-value < 2.2e-16
```

```
shapiro.test(wizmir$wind_speed)
```

Shapiro-Wilk normality test

```
data: wizmir$wind_speed  
W = 0.88849, p-value < 2.2e-16
```

```
shapiro.test(wizmir$visibility)
```

Shapiro-Wilk normality test

```
data: wizmir$visibility  
W = 0.96969, p-value < 2.2e-16
```

```
shapiro.test(wizmir$wind_speed)
```

Shapiro-Wilk normality test

```
data: wizmir$wind_speed  
W = 0.88849, p-value < 2.2e-16
```

```
shapiro.test(wizmir$max_wind_speed)
```

Shapiro-Wilk normality test

```
data: wizmir$max_wind_speed  
W = 0.40843, p-value < 2.2e-16
```

```
shapiro.test(wizmir$mean_temperature)
```

Shapiro-Wilk normality test

```
data: wizmir$mean_temperature  
W = 0.95212, p-value < 2.2e-16
```

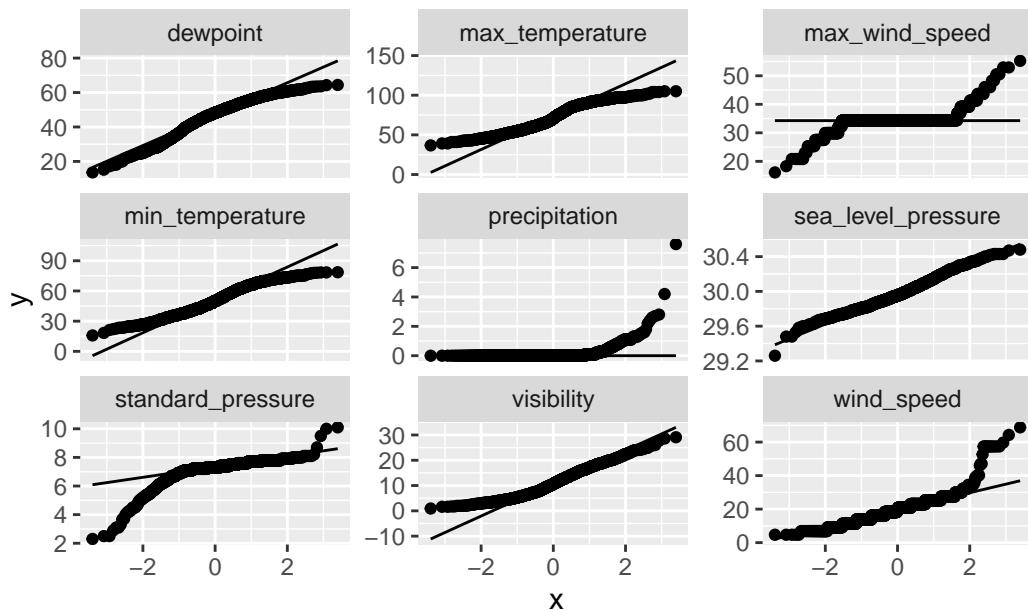
En todos los casos, el p-valor es extremadamente pequeño, lo que sugiere que hay evidencia significativa para rechazar la hipótesis nula de que los datos provienen de una distribución normal. En otras palabras, según el test de Shapiro-Wilk, ninguna de las variables parece seguir una distribución normal.

QQ-plot

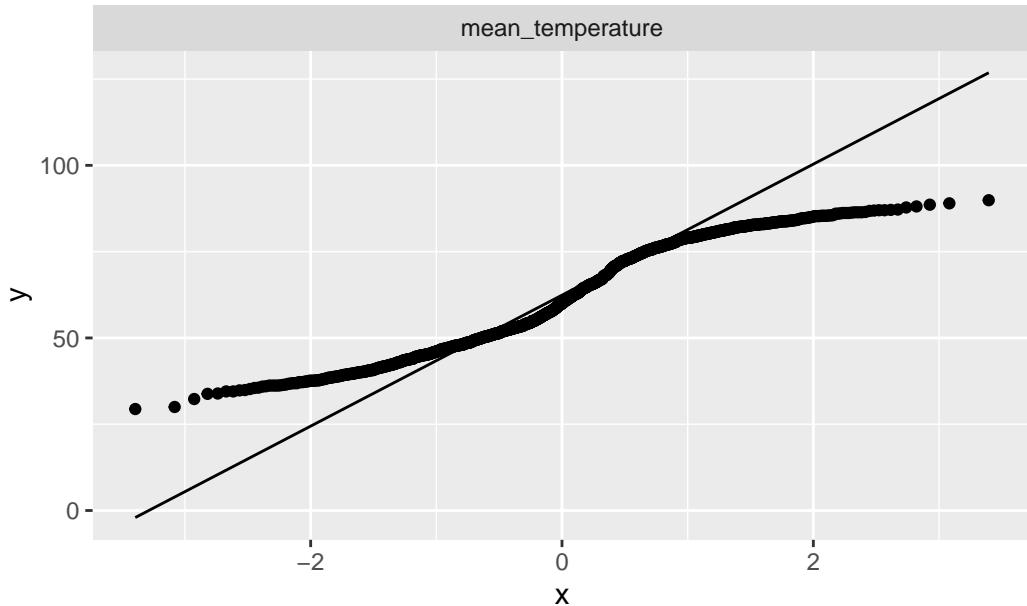
```
library(DataExplorer)
```

```
Warning: package 'DataExplorer' was built under R version 4.3.2
```

```
plot_qq(wizmir)
```



Page 1



Page 2

A partir del QQ-plot podemos sacar las siguientes conclusiones:

- La variable sea_level_pressure se asemeja casi completamente a la distribución normal.
- Todas las variables relacionadas con la temperatura (menos dewpoint) parecen seguir un distribución muy similar, teniendo lo que parecen dos modas (cabezas) en la distribución, estas están más pronunciadas sobre todo en max_temperature y mean_temperature, no tanto en min_temperature.
- Todas las variables aunque no siguen completamente la línea que marca la distribución normal, la mayoría de estas la siguen en gran parte, además parecen tener una sola cabeza las distribuciones.

En resumen, si bien el test de Shapiro-Wilk sugiere que hay evidencia significativa para rechazar la hipótesis nula de que los datos provienen de una distribución normal, mirando el gráfico QQ-plot nos damos cuenta de que en general nuestras variables tienen cierto parecido a una distribución normal, lo cual es bueno de cara a las asunciones estadísticas relacionadas con la normalidad de los datos que tienen en cuenta muchos modelos predictivos.

2.4 Visualización de datos

Además de usar estadística descriptiva para mostrar los rasgos que presentan las variables del conjunto de datos, realizar distintos tipos de visualizaciones es interesante para no solo obtener información nueva, sino también contrastar la información que hemos obtenido mediante la estadística descriptiva.

2.4.1 Exploración de distribuciones:

Histogramas

En primer lugar vamos a pintar un histograma para cada variable para hacernos una idea de la distribución que siguen las variables. Además, también pintaremos la media y la mediana para comprobar que ambas no estan muy alejadas entre sí.

```
library(ggplot2)

plot_hist <- function(data) {
  sapply(names(data), function(col_name) {

    p <- data %>% ggplot(aes_string(x = col_name)) +
      geom_histogram(color = "darkblue", fill = "lightblue") +
      labs(title = paste("Curva de densidad - ", col_name)) +
      geom_vline(xintercept = median(data[[col_name]]),
                 color = "red", linetype = "dashed", size = 1.5) +
      geom_vline(xintercept = mean(data[[col_name]]), color = "blue",
                 linetype = "dotted", size = 1.5)
    print(p)
  })
}

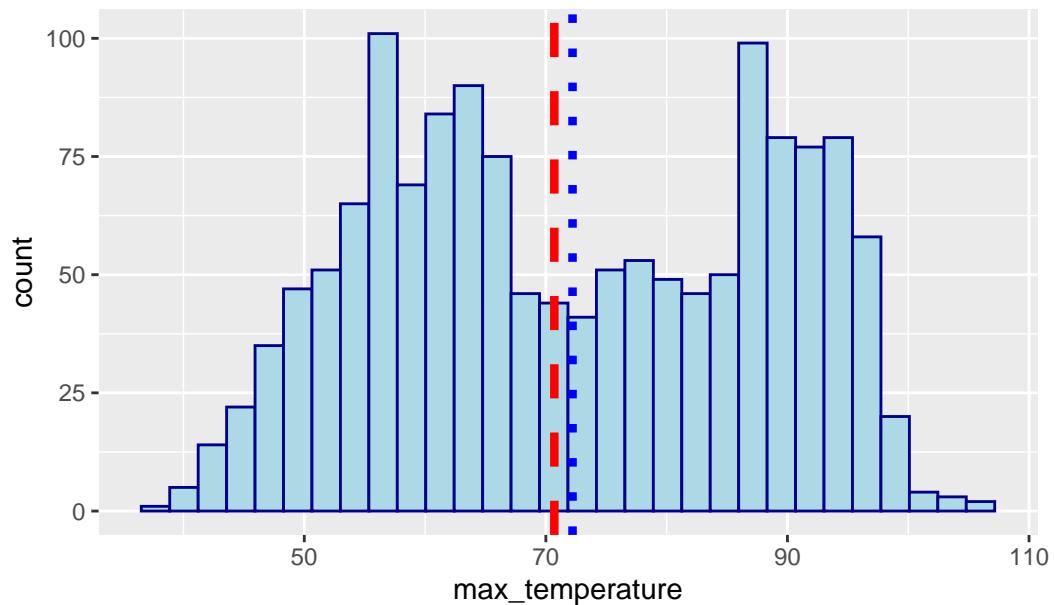
plot_hist(wizmir)
```

```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
  i Please use tidy evaluation idioms with `aes()``.
  i See also `vignette("ggplot2-in-packages")` for more information.
```

```
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
  i Please use `linewidth` instead.
```

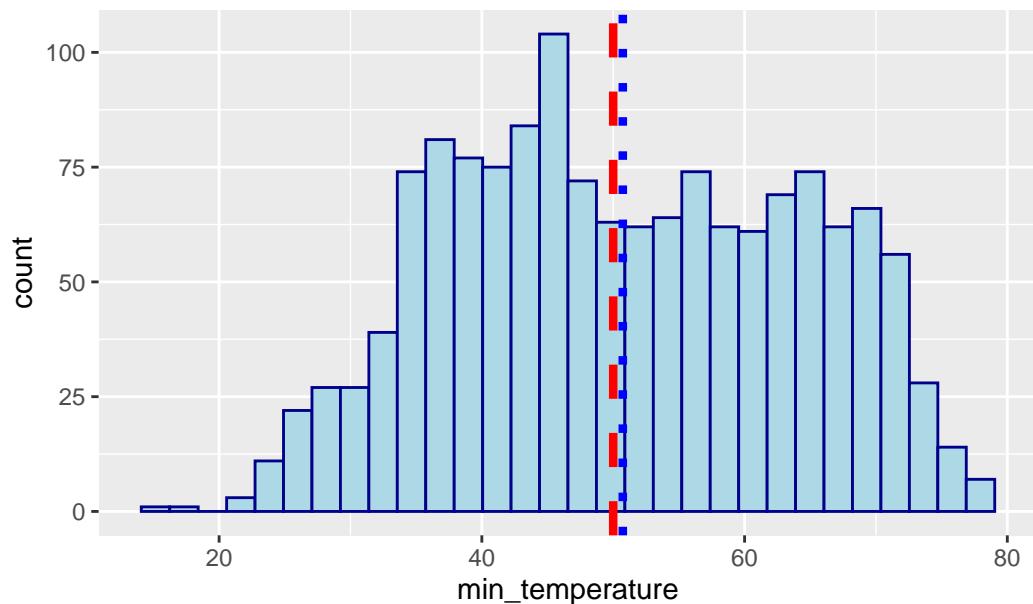
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – max_temperature



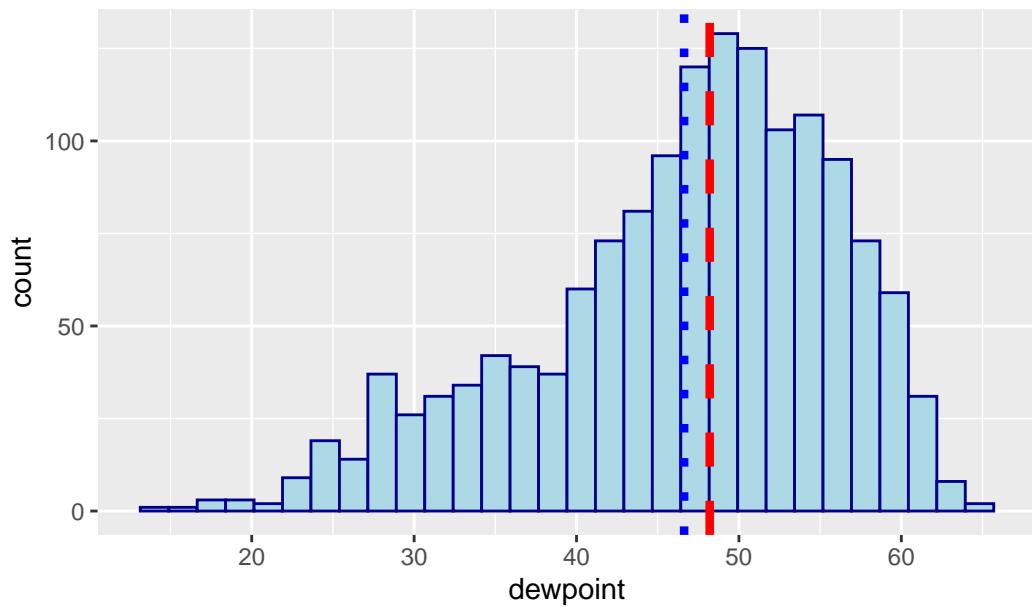
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – min_temperature



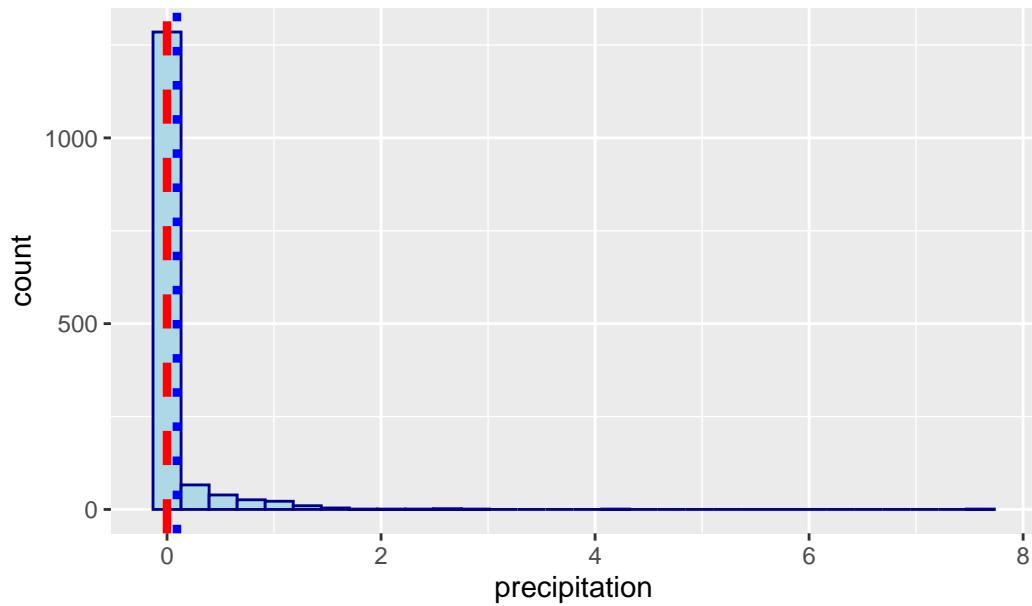
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – dewpoint



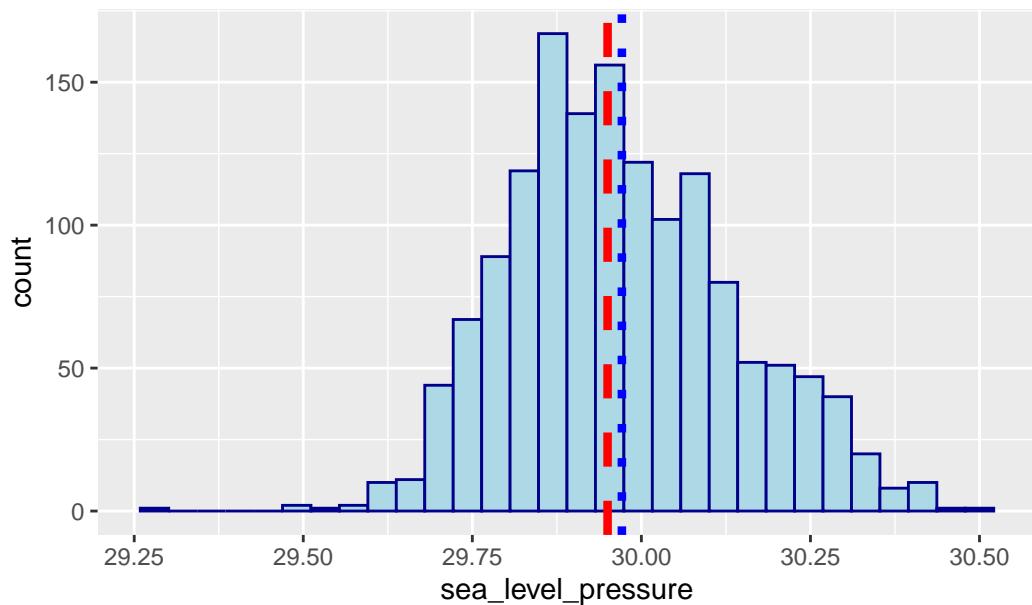
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – precipitation



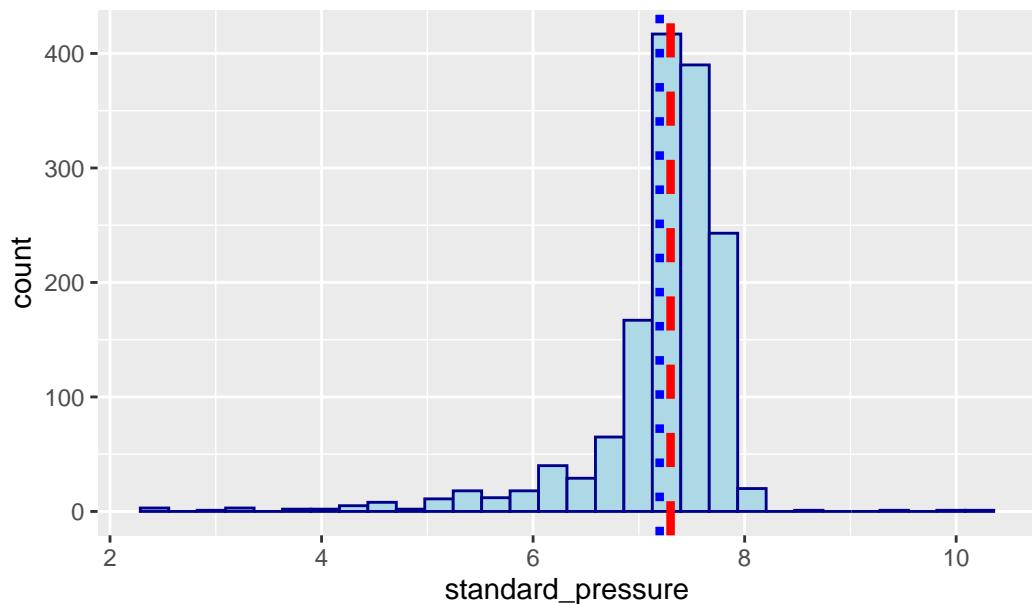
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – sea_level_pressure



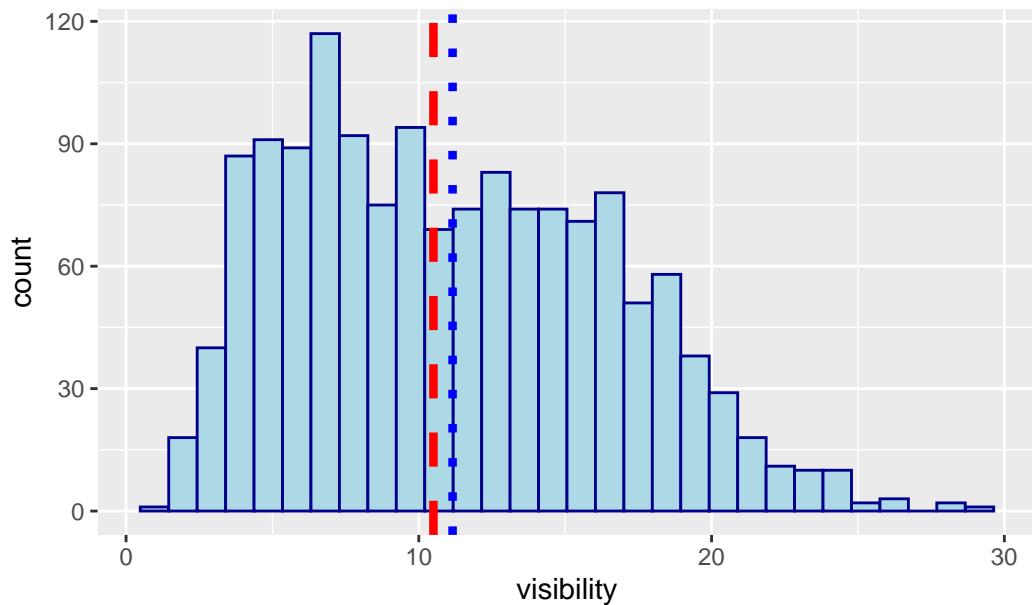
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – standard_pressure



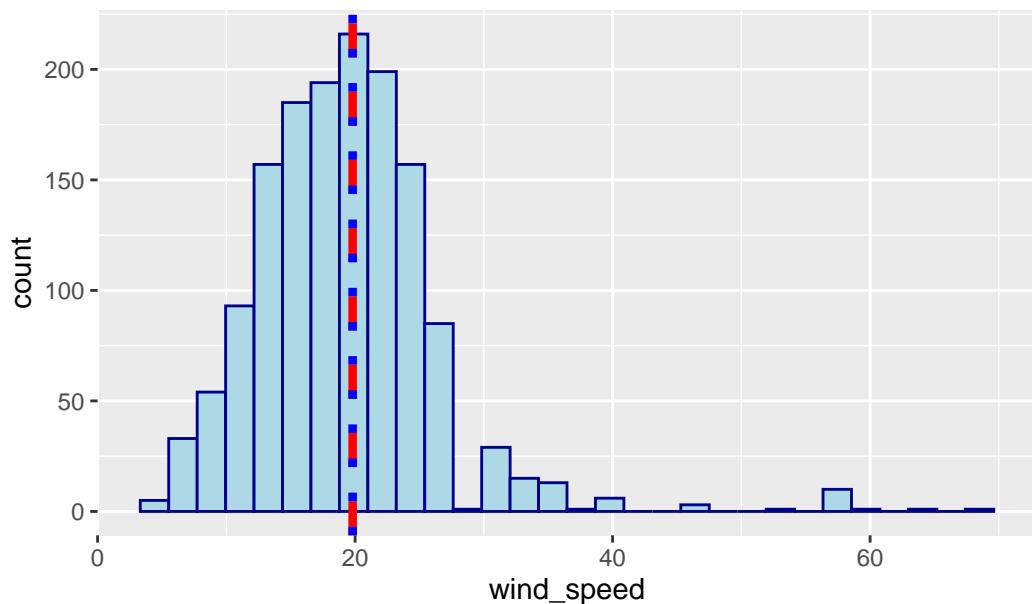
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – visibility



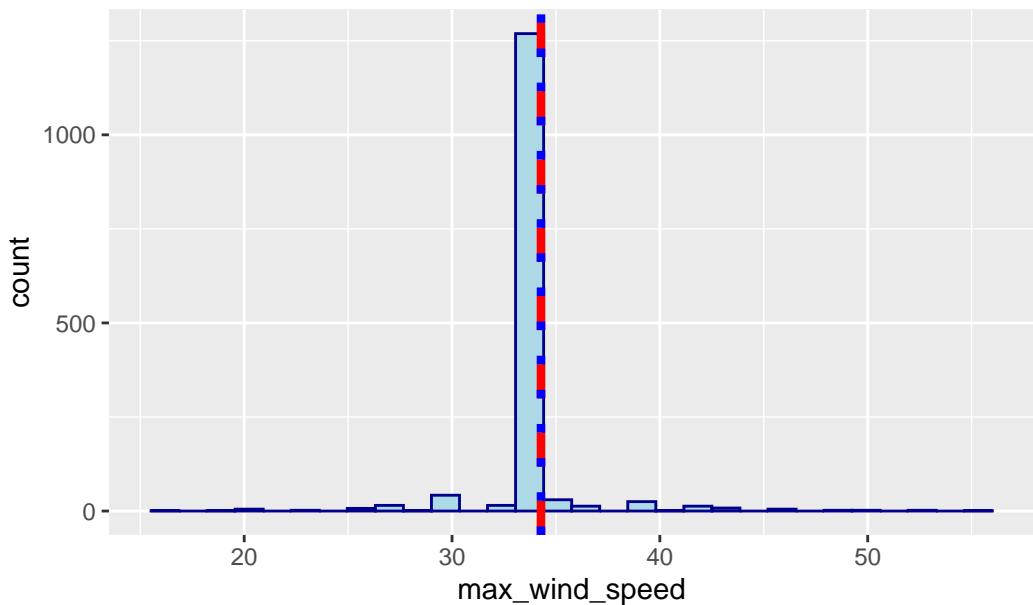
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – wind_speed



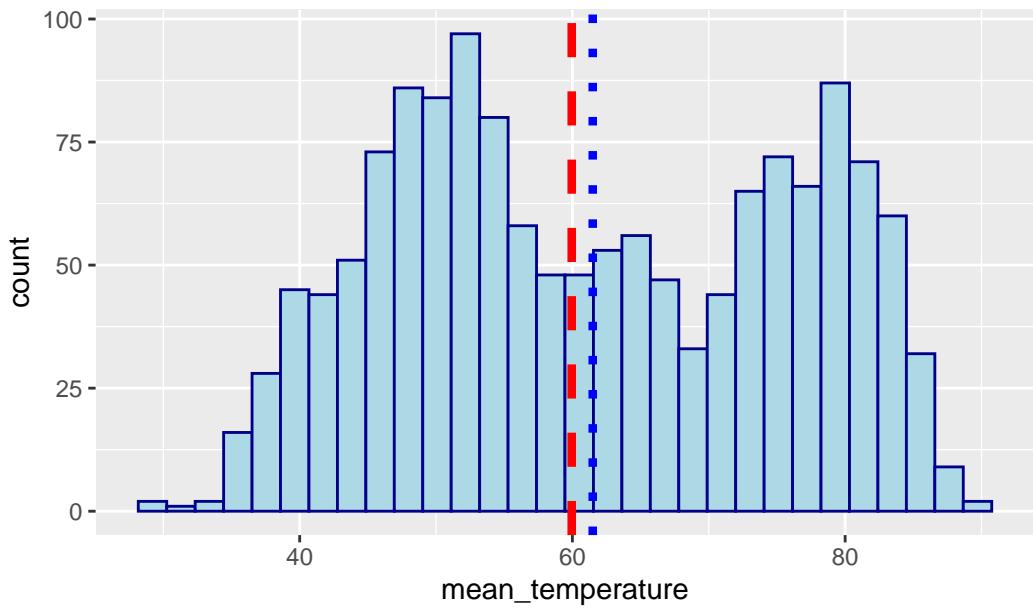
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – max_wind_speed



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – mean_temperature



```

      max_temperature min_temperature dewpoint      precipitation
data      data.frame,10   data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~max_temperature ~min_temperature ~dewpoint ~precipitation
theme     list,0         list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?             ?
labels    list,5         list,5         list,5         list,5
                  sea_level_pressure standard_pressure visibility
data      data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~sea_level_pressure ~standard_pressure ~visibility
theme     list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?
labels    list,5         list,5         list,5
                  wind_speed      max_wind_speed mean_temperature
data      data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~wind_speed    ~max_wind_speed ~mean_temperature
theme     list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?
labels    list,5         list,5         list,5

```

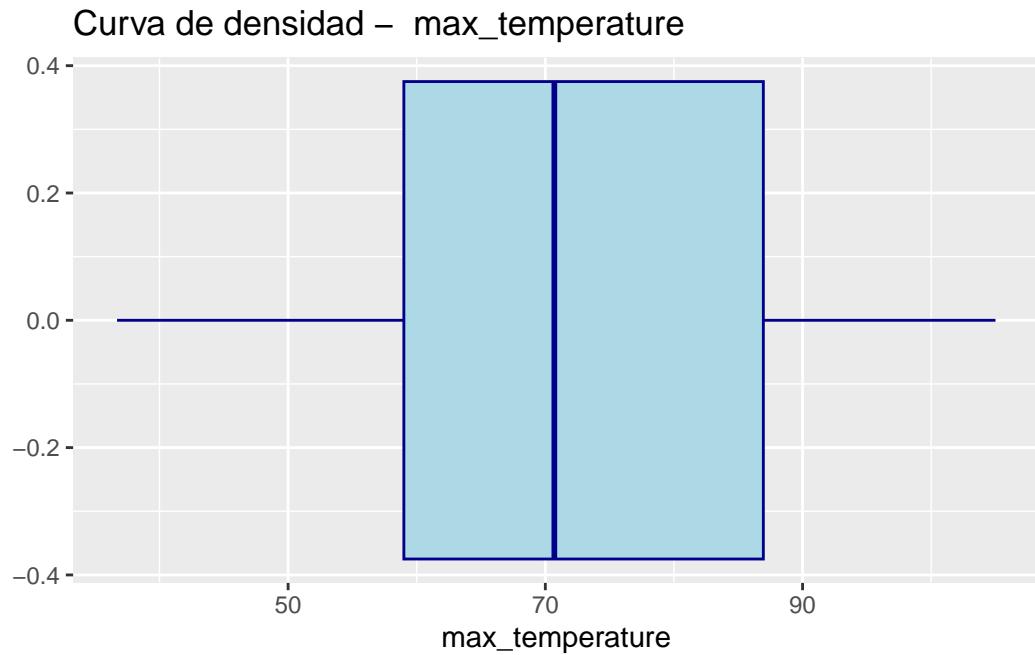
A partir de los histogramas podemos sacar las siguientes conclusiones:

- Tal y como nos indicaban las medidas de tendencia central y dispersión, la media y la mediana son bastante similares y además son representativas para cada una de las variables del dataset.
- En el histograma podemos apreciar claramente las dos cabezas que presentan las variables max_temperature, min_temperature y mean_temperature. Además como habíamos detectado en el gráfico QQ-plot estas variables siguen distribuciones muy similares.

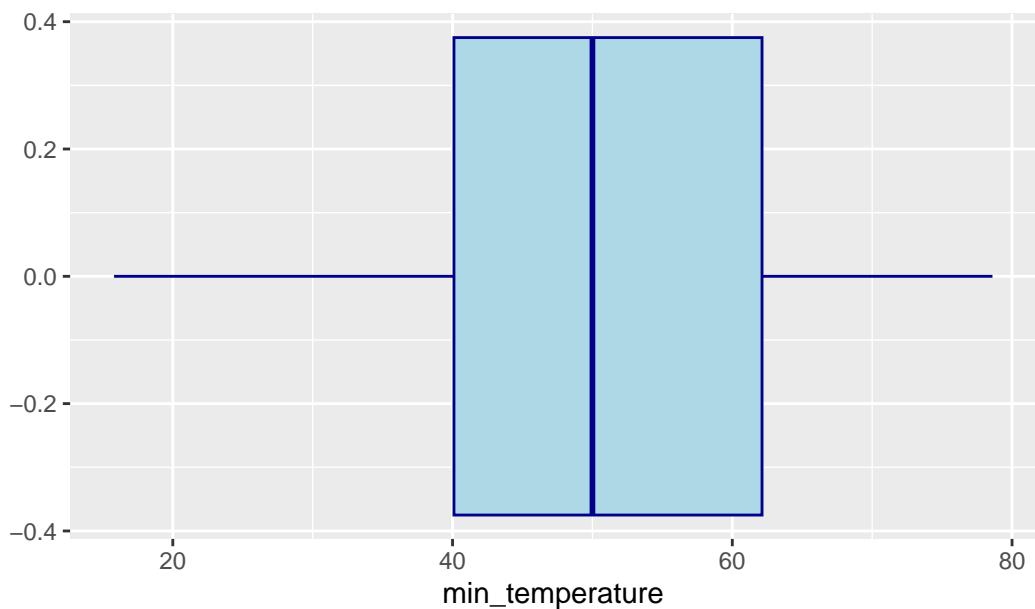
Gráfico de cajas

A continuación vamos a mostrar diagramas de caja para cada variable, principalmente con el objeto de detectar outliers en las variables

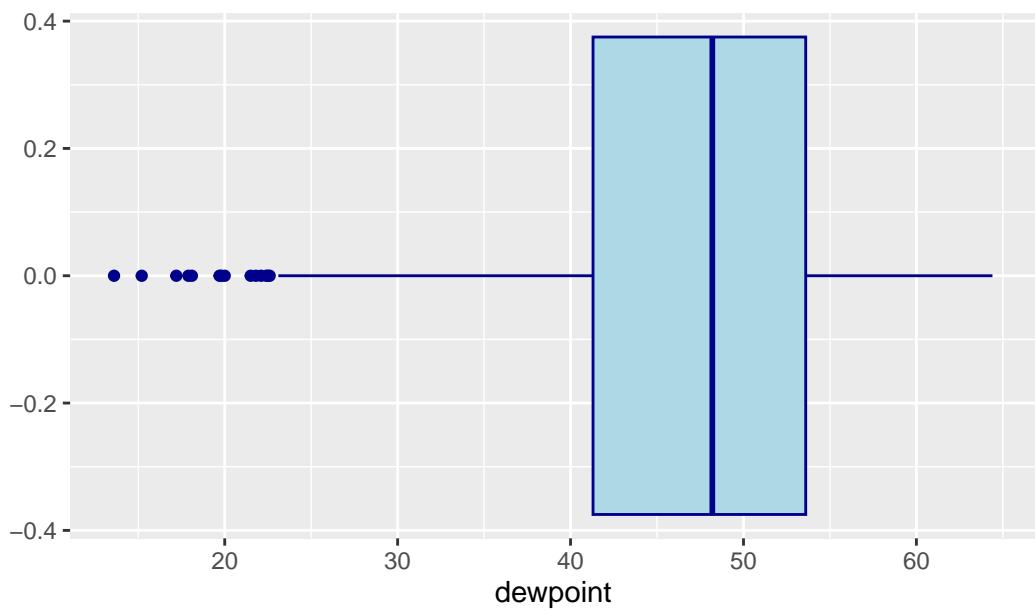
```
plot_boxplots <- function(data) {  
  sapply(names(data), function(col_name) {  
    p <- data %>% ggplot(aes_string(x = col_name)) +  
      geom_boxplot(color = "darkblue", fill = "lightblue") +  
      labs(title = paste("Curva de densidad - ", col_name))  
    print(p)  
  })  
}  
  
plot_boxplots(wizmir)
```



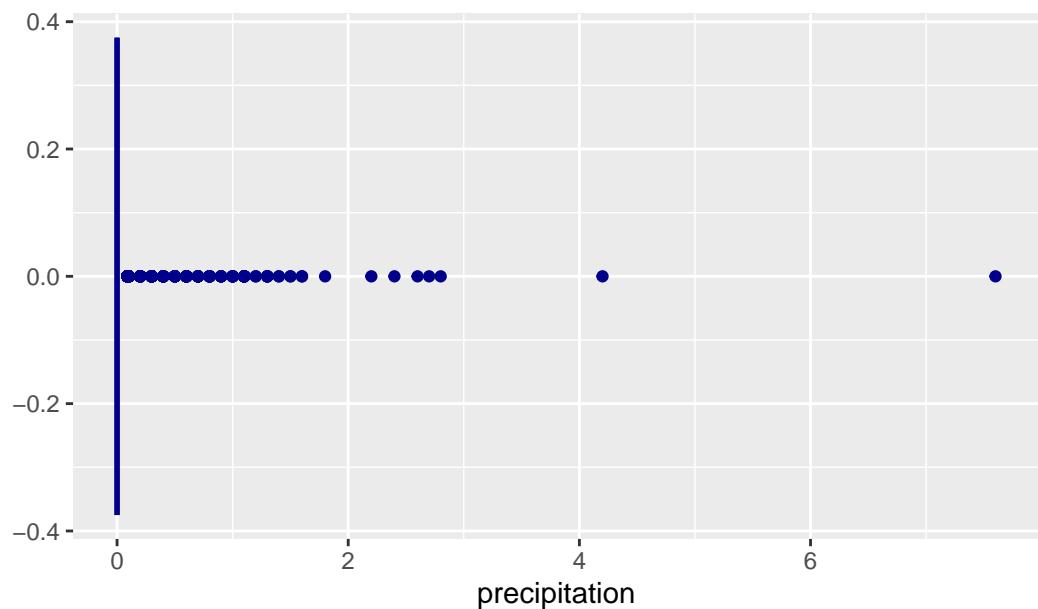
Curva de densidad – min_temperature



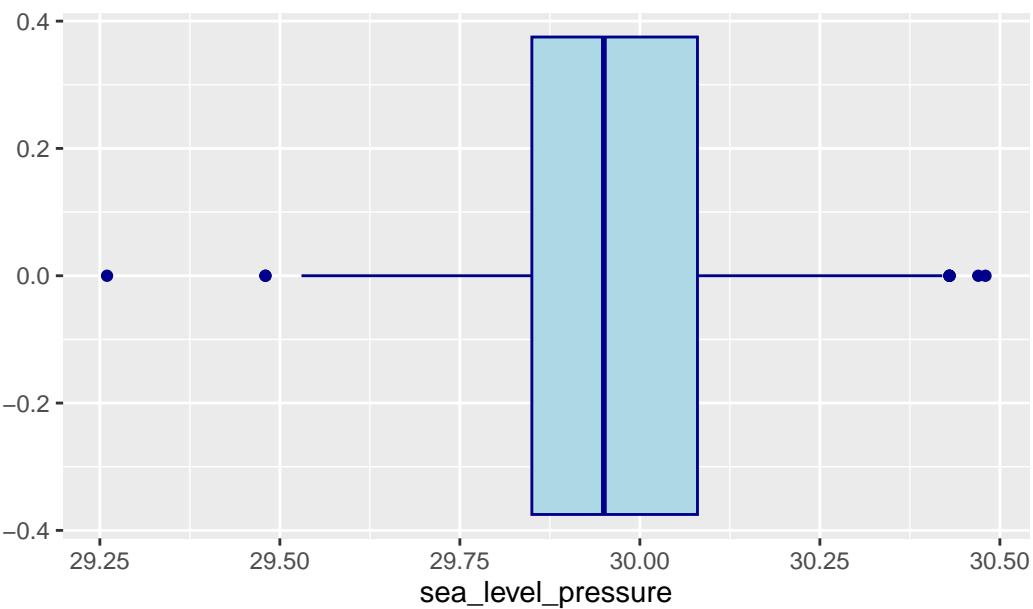
Curva de densidad – dewpoint



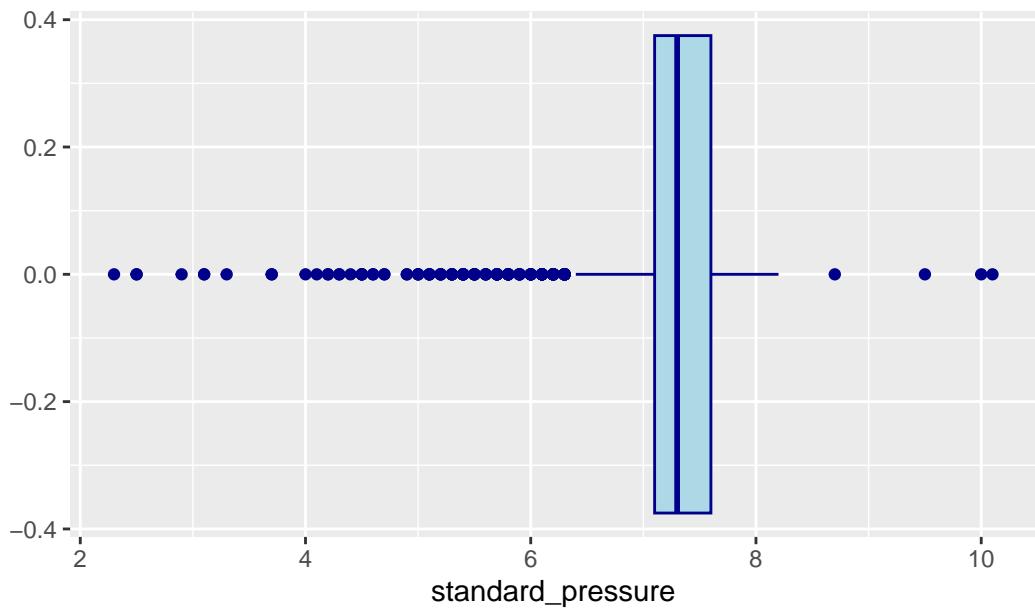
Curva de densidad – precipitation



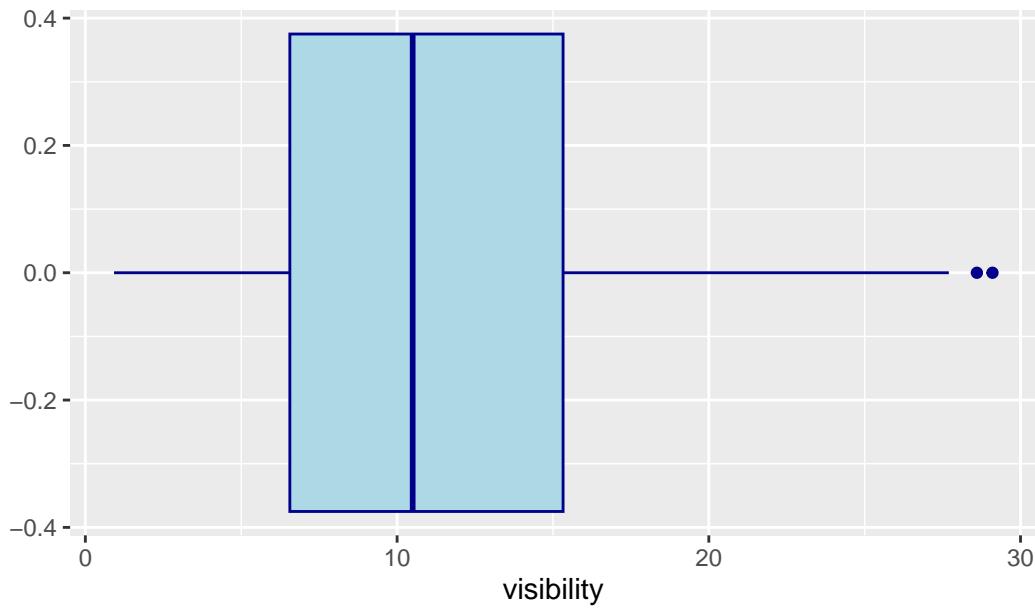
Curva de densidad – sea_level_pressure



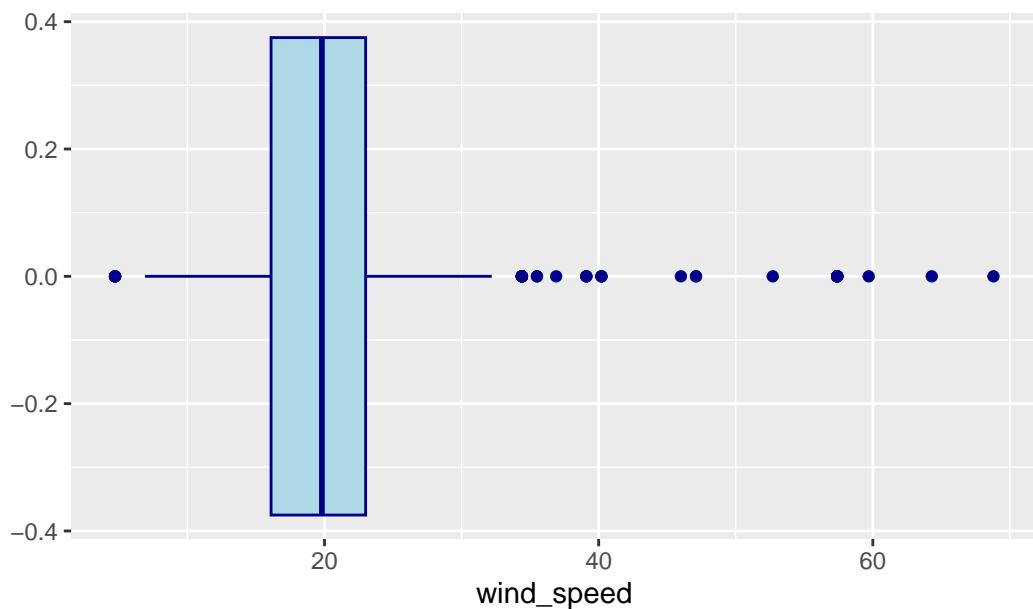
Curva de densidad – standard_pressure



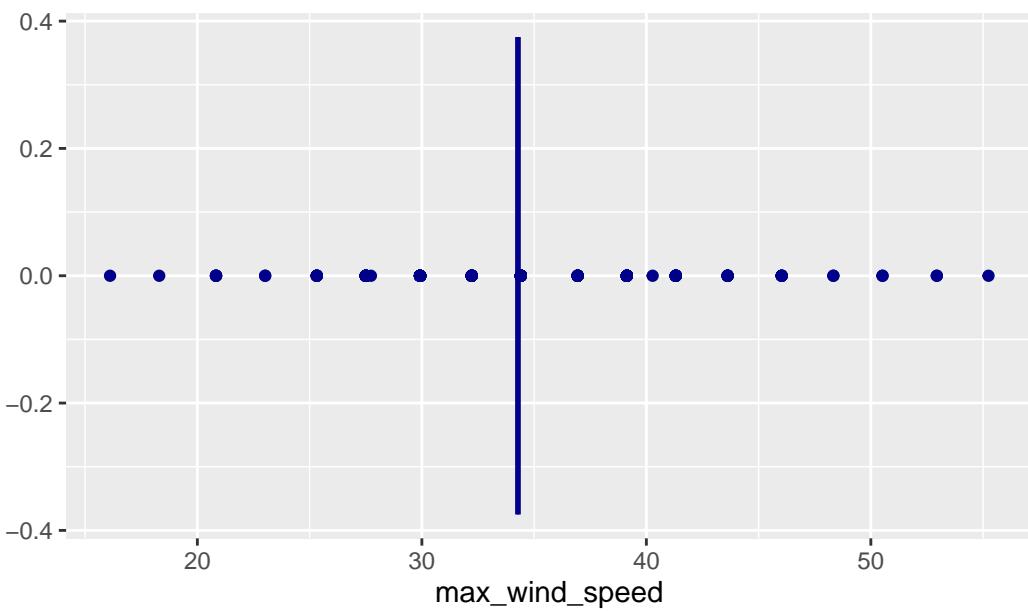
Curva de densidad – visibility



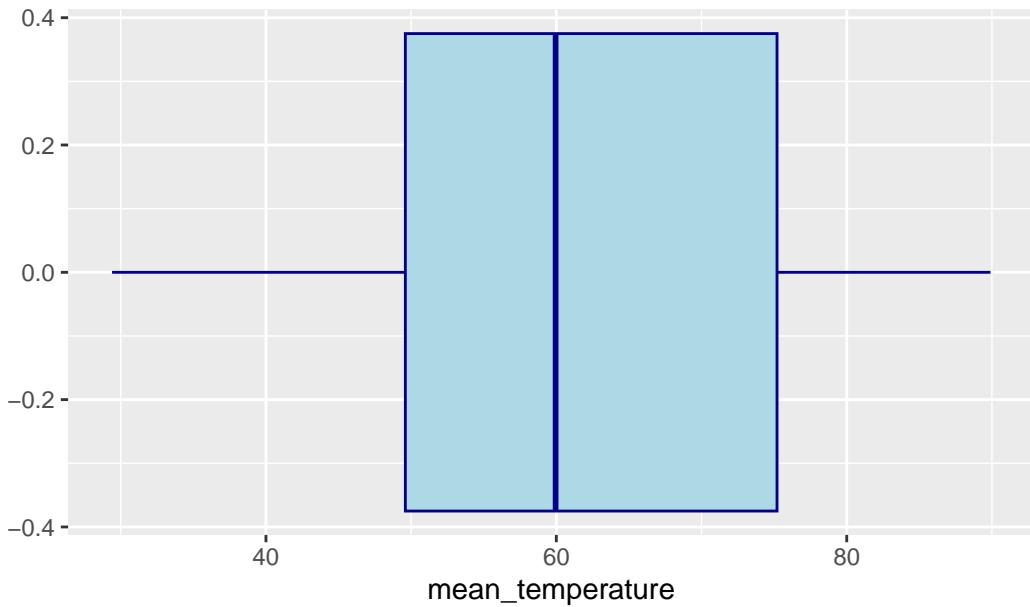
Curva de densidad – wind_speed



Curva de densidad – max_wind_speed



Curva de densidad – mean_temperature



```

      max_temperature min_temperature dewpoint      precipitation
data      data.frame,10   data.frame,10   data.frame,10   data.frame,10
layers    list,1          list,1          list,1          list,1
scales    ScalesList,2   ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~max_temperature ~min_temperature ~dewpoint   ~precipitation
theme     list,0          list,0          list,0          list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2   FacetNull,2
plot_env   ?              ?              ?              ?
labels     list,2          list,2          list,2          list,2
                  sea_level_pressure standard_pressure visibility
data      data.frame,10   data.frame,10   data.frame,10
layers    list,1          list,1          list,1
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~sea_level_pressure ~standard_pressure ~visibility
theme     list,0          list,0          list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2
plot_env   ?              ?              ?
labels     list,2          list,2          list,2
                  wind_speed      max_wind_speed  mean_temperature
data      data.frame,10   data.frame,10   data.frame,10
layers    list,1          list,1          list,1

```

```

scales      ScalesList,2      ScalesList,2      ScalesList,2
mapping     ~wind_speed       ~max_wind_speed ~mean_temperature
theme       list,0           list,0           list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet       FacetNull,2      FacetNull,2      FacetNull,2
plot_env    ?                ?                ?
labels      list,2          list,2          list,2

```

Podemos ver como varias de las variables presentar valores anómalos, algunas como max_wind_speed, wind_speed, precipitation y standard_pressure presentan en gran cantidad, la razón de esto es que sus distribuciones presentan la mayoría de los valores muy juntos en un rango de valores muy pequeño, por lo que cualquier valor que se aleje aunque sea un poco de ese rango será considerado como valor atípico.

2.4.2 Exploración de relaciones entre variables

De cara a la elección de las mejores variables para utilizar en un modelo lineal en Regresión, vamos a estudiar la interacción entre variables.

2.4.2.1 Gráfico de puntos variable independiente vs dependiente

A continuación vamos a graficar cada una de las variables independientes frente la variable dependiente, esto lo hacemos con el objeto de buscar aquellas variables candidatas para formar un modelo lineal en base a ellas.

```

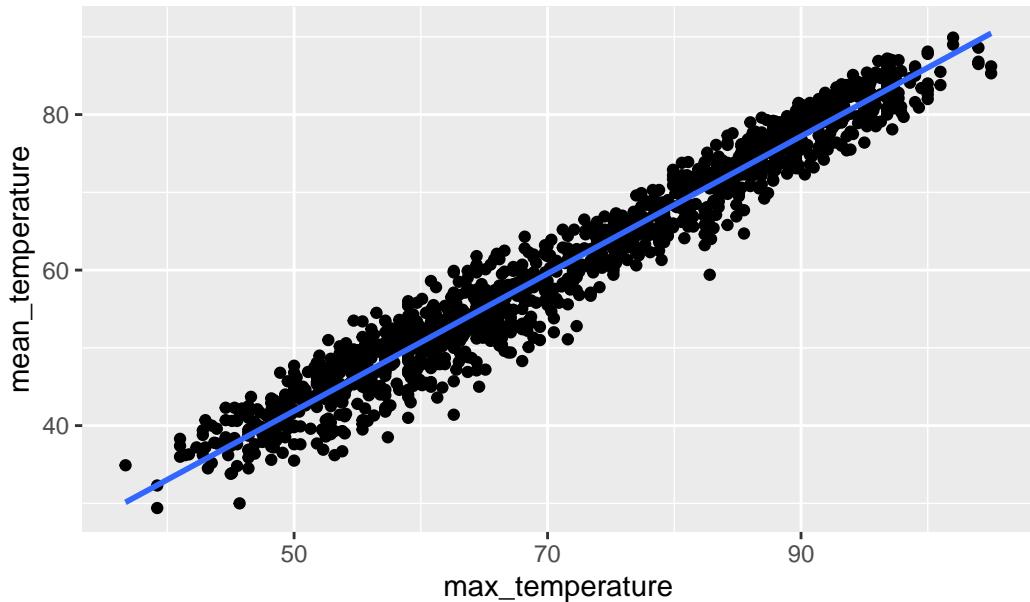
plot_corrs_with <- function(data, X, y) {
  sapply(X, function(col) {
    p <- data %>% ggplot(aes_string(x = col, y = y)) +
      geom_point() +
      geom_smooth(method = "lm") +
      labs(title = paste("Correlación entre", col, "y", y))
    print(p)
  })
}

plot_corrs_with(wizmir, X = colnames(wizmir[, !names(wizmir) %in% "mean_temperature"]),
                y = "mean_temperature")

`geom_smooth()` using formula = 'y ~ x'

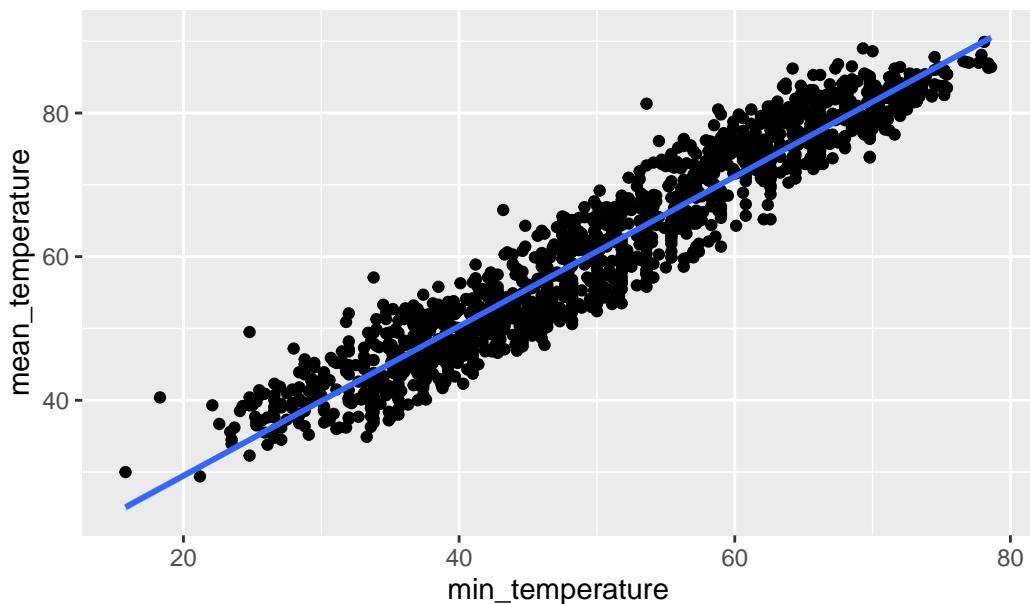
```

Correlación entre max_temperature y mean_temperature



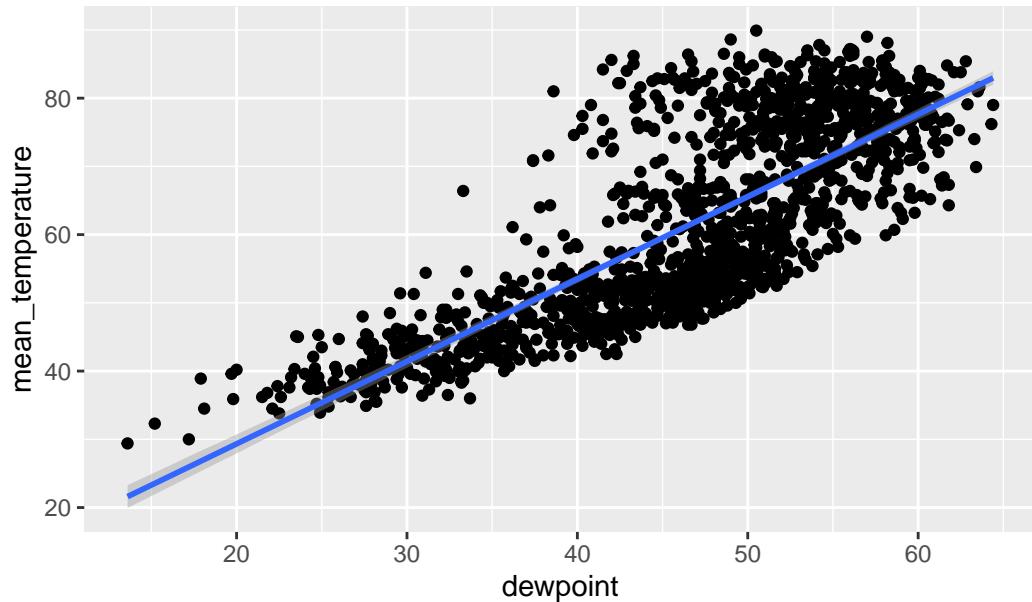
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre min_temperature y mean_temperature



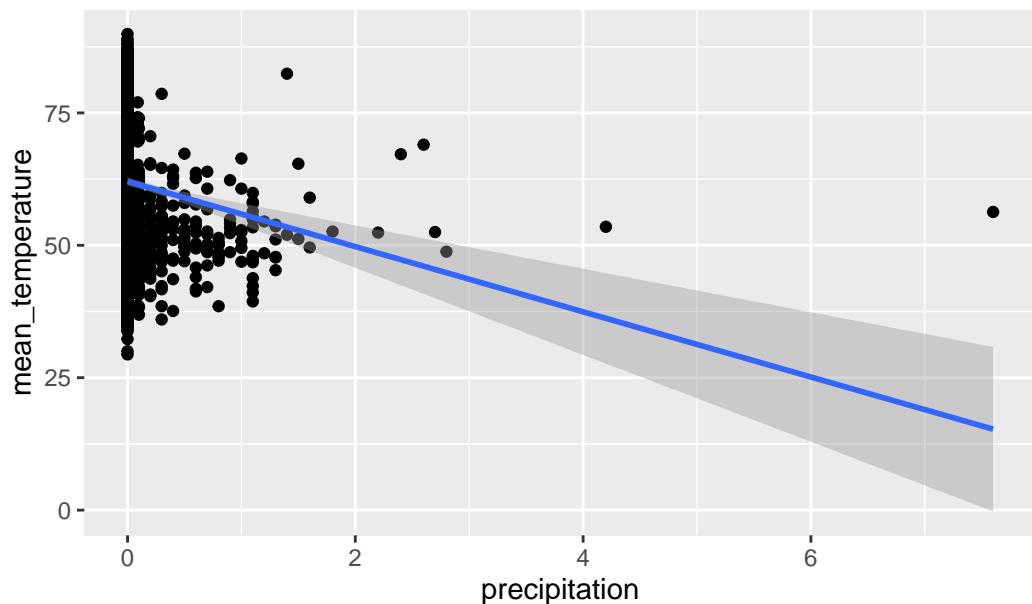
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre dewpoint y mean_temperature



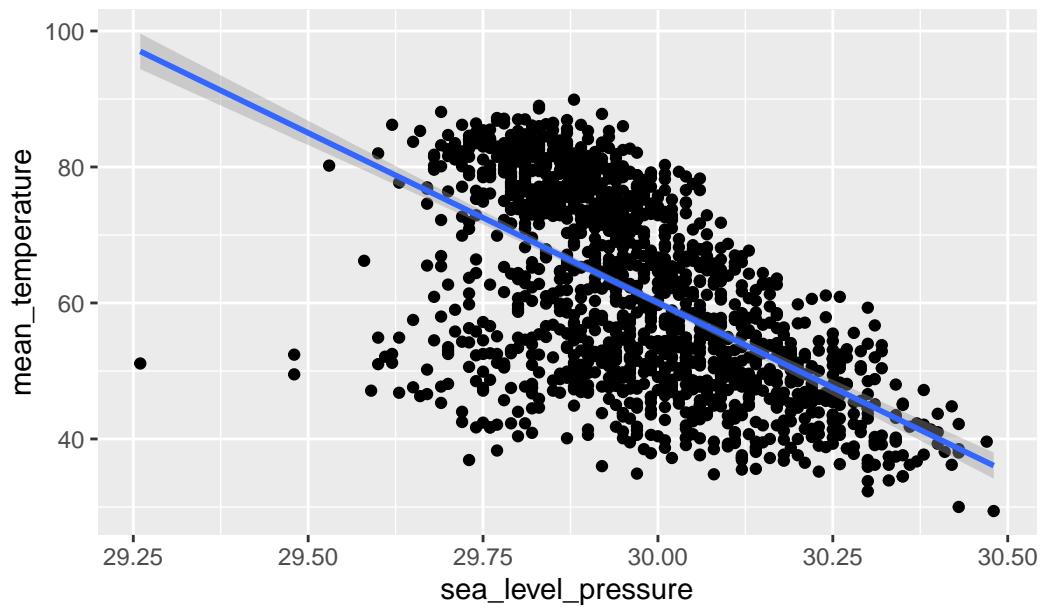
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre precipitation y mean_temperature



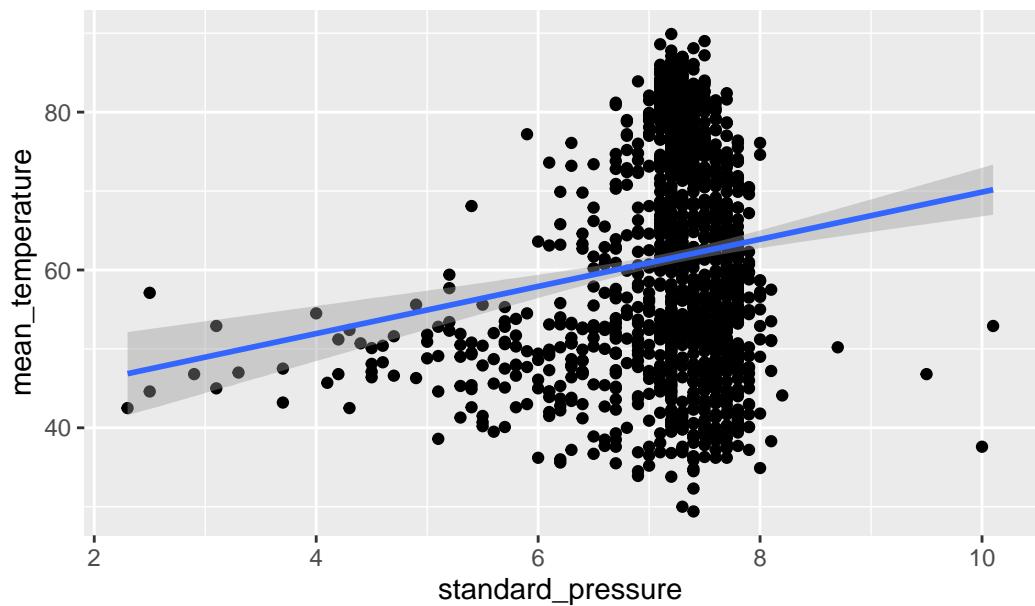
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre sea_level_pressure y mean_temperature



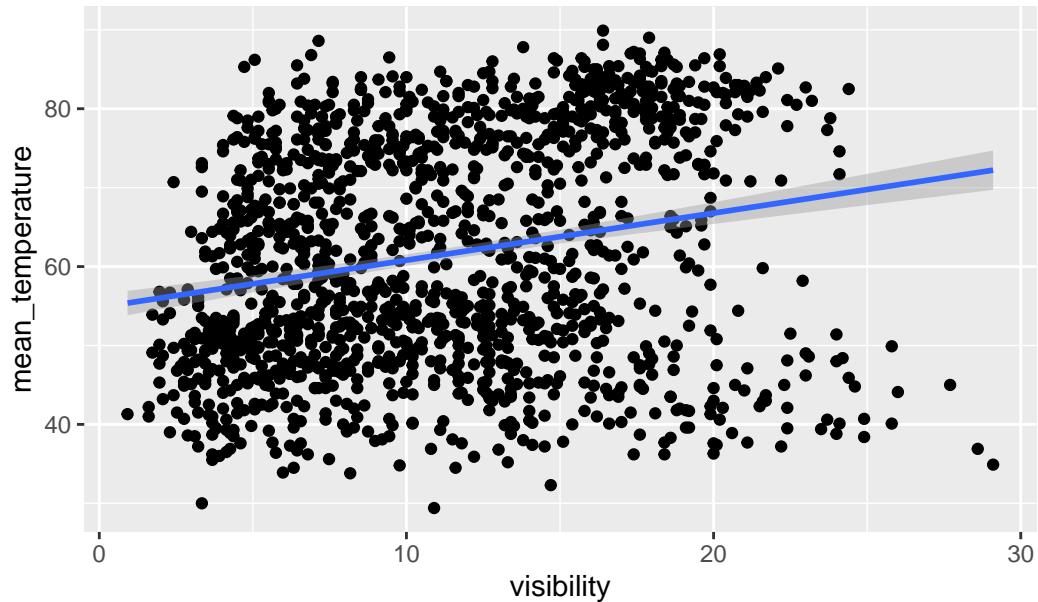
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre standard_pressure y mean_temperature



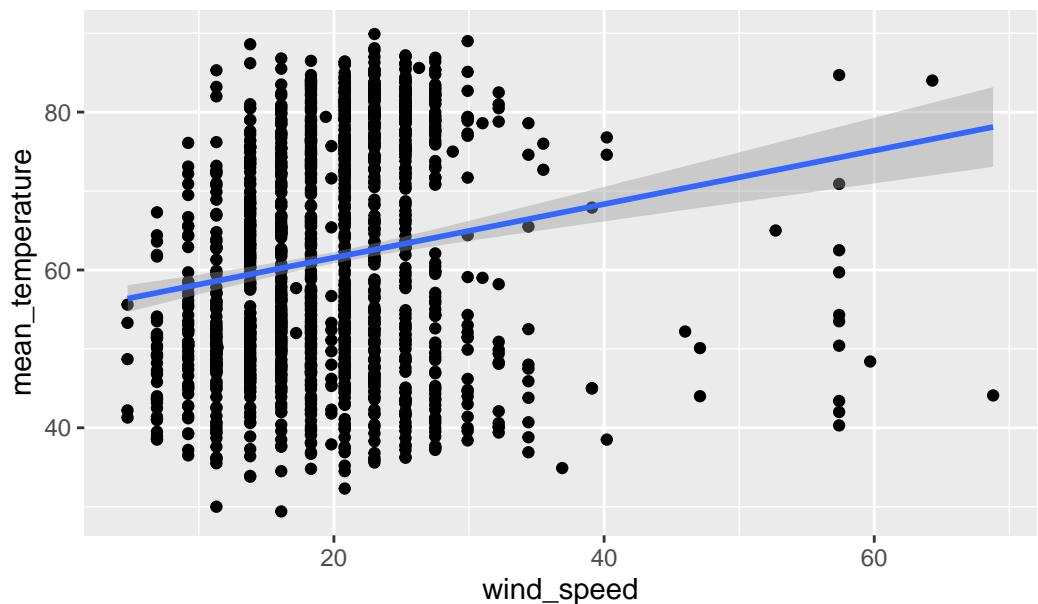
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre visibility y mean_temperature



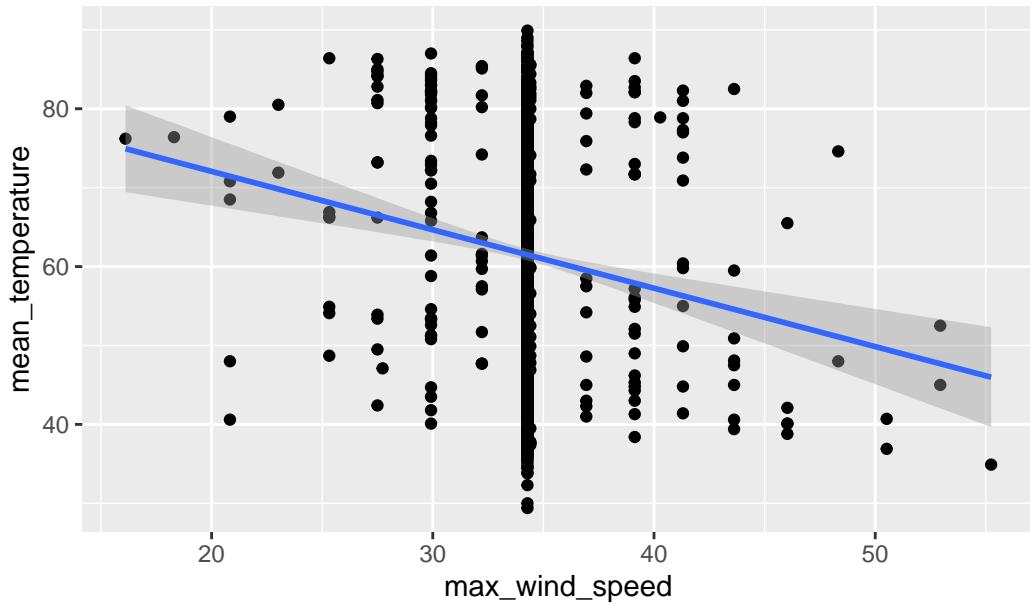
```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre wind_speed y mean_temperature



```
`geom_smooth()` using formula = 'y ~ x'
```

Correlación entre max_wind_speed y mean_temperature



```

      max_temperature min_temperature dewpoint precipitation
data      data.frame,10  data.frame,10  data.frame,10  data.frame,10
layers     list,2        list,2        list,2        list,2
scales    ScalesList,2  ScalesList,2  ScalesList,2  ScalesList,2
mapping   uneval,2      uneval,2      uneval,2      uneval,2
theme      list,0        list,0        list,0        list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2   FacetNull,2
plot_env    ?            ?            ?            ?
labels      list,3        list,3        list,3        list,3
                           sea_level_pressure standard_pressure visibility
data      data.frame,10  data.frame,10  data.frame,10
layers     list,2        list,2        list,2
scales    ScalesList,2  ScalesList,2  ScalesList,2
mapping   uneval,2      uneval,2      uneval,2
theme      list,0        list,0        list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2
plot_env    ?            ?            ?
labels      list,3        list,3        list,3
                           wind_speed      max_wind_speed
data      data.frame,10  data.frame,10
layers     list,2        list,2

```

```

scales      ScalesList,2      ScalesList,2
mapping     uneval,2         uneval,2
theme       list,0          list,0
coordinates CoordCartesian,5 CoordCartesian,5
facet       FacetNull,2      FacetNull,2
plot_env    ?                ?
labels      list,3          list,3

```

A partir de las nubes de punto podemos sacar las siguientes conclusiones:

- Las variables max_temperature y min_temperature presentan un relación lineal muy fuerte con la variable objetivo. Son las principales candidatas para realizar un modelo lineal.
- Dewpoint y sea_level pressure presentan una cierta relación lineal con la variable objetivo, sin embargo esta relación no es tan buena como la que tiene con max_temperature y min_temperature de cara a realizar un modelo lineal en base a ellas.
- Las demás variables no presentan ninguna relación clara con la variable objetivo.

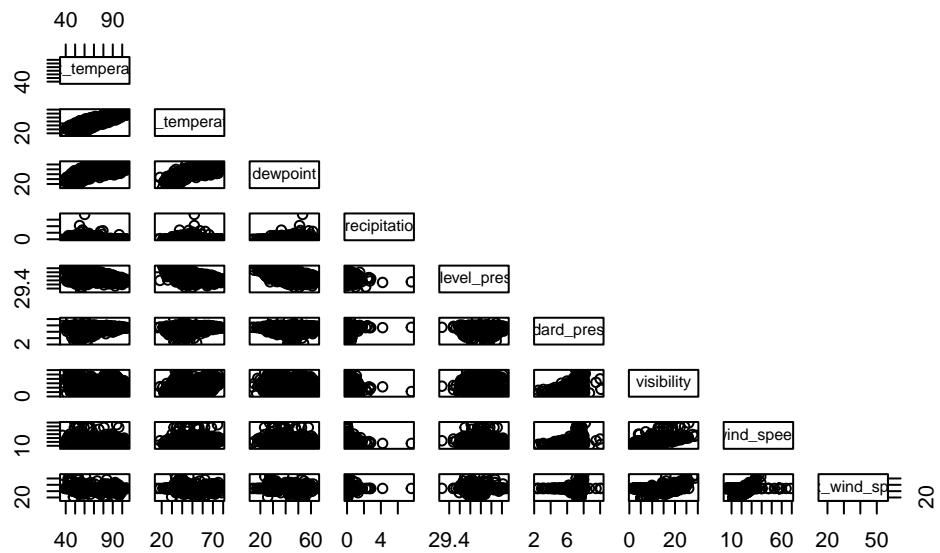
2.4.2.2 Gráfico de puntos variable independiente vs independiente

En este caso vamos a graficar cada variable independiente frente a las demás independiente, todo esto con el objeto de detectar correlaciones entre variables y así saber qué variables se podrían no usar en un modelo lineal ya que otra contiene la misma información. Mostraremos tanto una nube de puntos como un gráfico de correlaciones, pues este último es muy útil para ver la correlación entre dos variables.

```
library(Sleuth2)
```

```
Warning: package 'Sleuth2' was built under R version 4.3.2
```

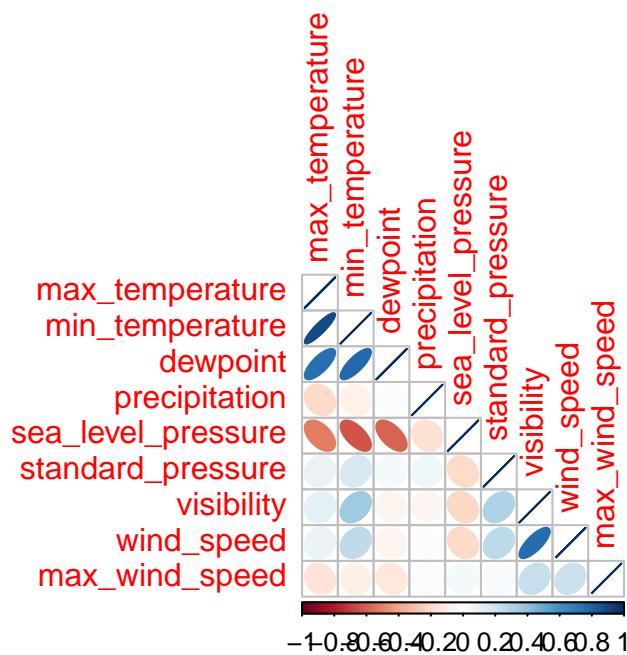
```
pairs(wizmir[!names(wizmir) %in% "mean_temperature"], upper.panel = NULL)
```



```
library(corrplot)
```

```
corrplot 0.92 loaded
```

```
corrplot(cor(wizmir[, !names(wizmir) %in% "mean_temperature"]),
         method = "ellipse", type = "lower")
```



Observaciones:

- Existe una muy fuerte correlación entre todas las variables relacionadas con la temperatura (min_temperature, max_temperature, dewpoint). Este es un indicio de que estas las variables contienen información redundante entre ellas.
- También existe una fuerte correlación entre la variable sea_level_pressure y las variables relacionadas con la temperatura.
- Existe una fuerte correlación entre la variable wind_speed y la variable visibility.
- Las demás correlaciones no son suficientemente fuertes como para tenerlas en cuenta.

2.5 Ingeniería de características.

2.5.1 Creación de variables:

Como hemos visto en el anterior apartado, todas las variables relacionadas con la temperatura presentan alta correlación entre estas. Como sabemos, utilizar varias variables con alta correlación en un modelo de regresión puede dar lugar al problema conocido como la multicolinealidad. La multicolinealidad se refiere a la alta correlación entre dos o más variables predictoras en un modelo de regresión múltiple.

Este problema se puede solucionar de distintas maneras (utilizar solo una de las variables correlacionadas, realizar PCA, ...). Desde el punto de vista de la ingeniería de características,

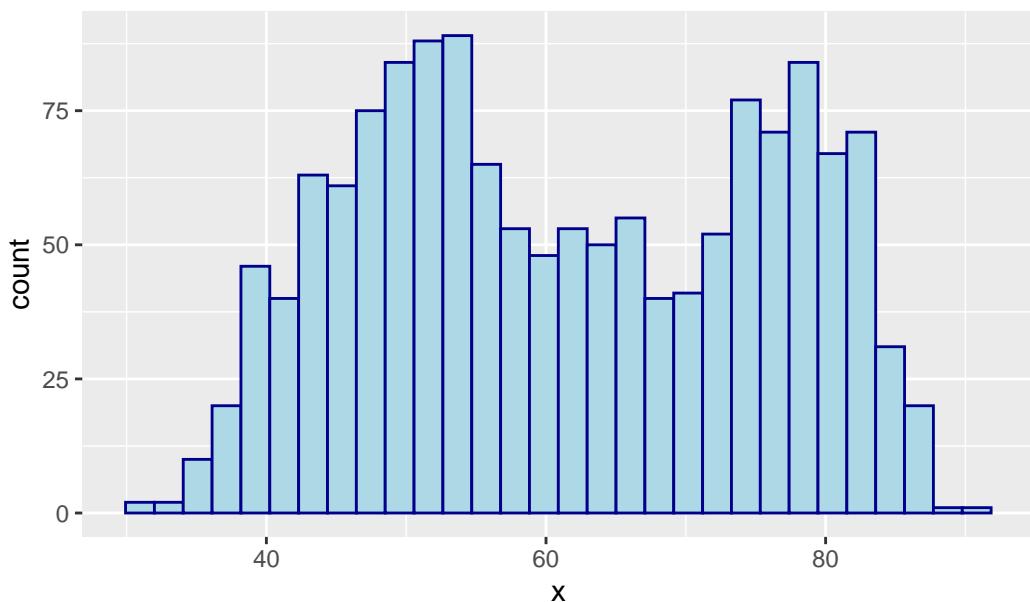
se ha optado por realizar una combinación de alguna de las variables relacionadas con la temperatura, en concreto se ha creado la variable mean_min_max_temperature, la cual se obtiene haciendo la media entre la temperatura máxima y mínima.

```
new_wizmir <- wizmir %>%
  mutate(mean_min_max_temperature = (max_temperature + min_temperature)/2)

new_wizmir %>% ggplot(aes_string(x = new_wizmir$mean_min_max_temperature)) +
  geom_histogram(color = "darkblue", fill = "lightblue") +
  labs(title = paste("Curva de densidad - r_temperature"))

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – r_temperature



Como podemos observar, la distribución de esta nueva variable está mitad de camino entre la distribución de min_temperature y max_temperature.

Vamos a hacer un gráfico de puntos de esta nueva variable frente a la variable objetivo para ver si es una buena variable para usar en el modelo lineal.

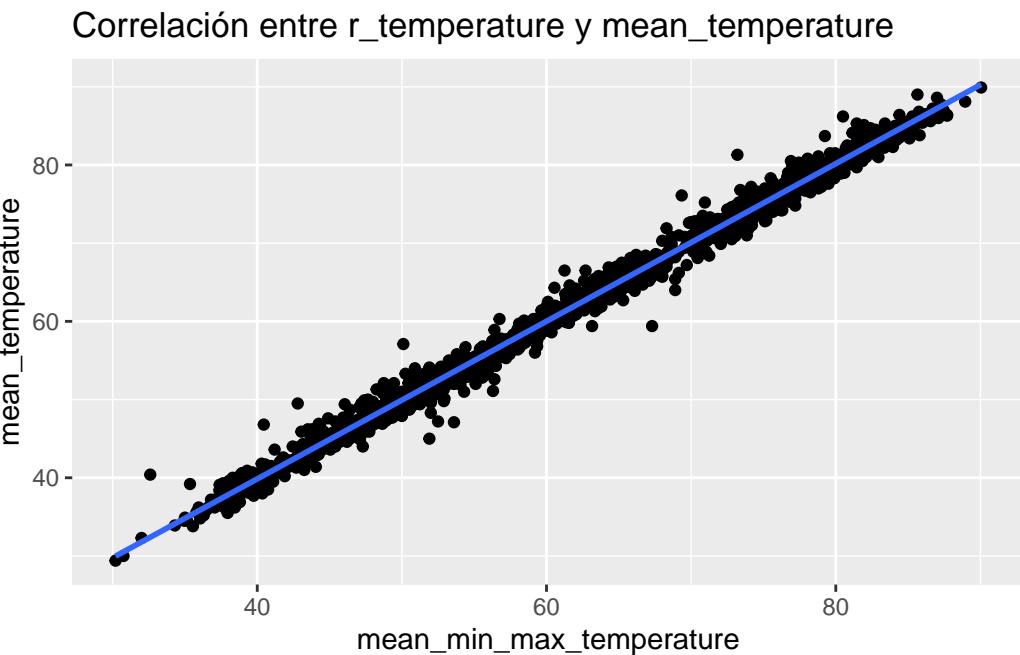
```
new_wizmir %>% ggplot(aes(x = mean_min_max_temperature, y = mean_temperature)) +
  geom_point()
```

```

geom_smooth(method = "lm") +
labs(title = paste("Correlación entre r_temperature y mean_temperature"))

`geom_smooth()` using formula = 'y ~ x'

```

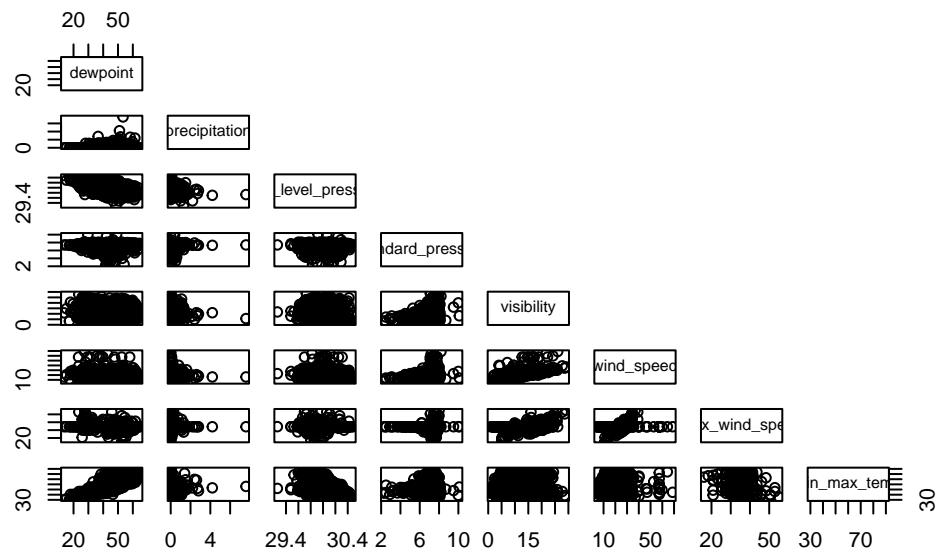


Como podemos observar, la nueva variable obtenida tiene la relación lineal más fuerte con la variable objetivo que cualquier de las variables originales del dataset. Esto tiene sentido pues la media entre la temperatura mínima y máxima que ha habido durante un día parece ser una buena aproximación de la temperatura media que hubo ese día. Seguramente realizando un modelo lineal en base a esta variable únicamente nos dará uno grandes resultados a la hora de predecir la variable objetivo. Además, al no tener que usar las variables max_temperature y min_temperature pues esta nueva resume ambas, nos quitamos en gran parte el problema de correlación entre variables.

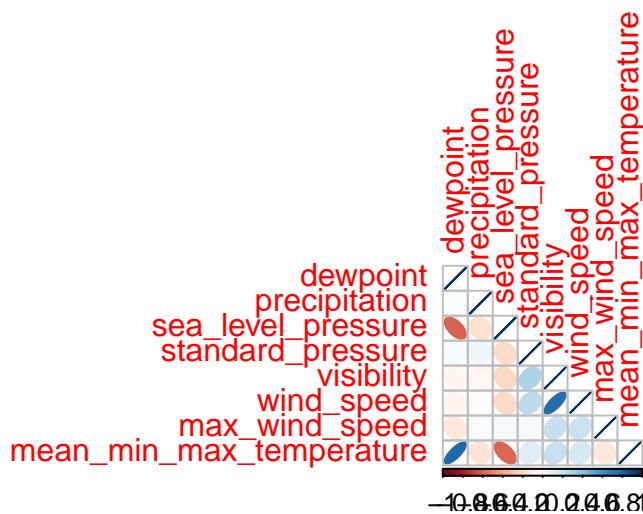
```

pairs(new_wizmir[, !names(new_wizmir) %in% c("min_temperature",
                                              "max_temperature",
                                              "mean_temperature")],
      upper.panel = NULL)

```



```
corrplot(cor(new_wizmir[, !names(new_wizmir) %in% c("min_temperature",
                                                 "max_temperature",
                                                 "mean_temperature")]),
        method = "ellipse", type = "lower")
```



Tal y como podemos observar, el número de variables que tenían correlación ha disminuido considerablemente.

2.5.2 Transformación de variables

En este apartado vamos a aplicar distintas transformaciones a las variables para ver si alguna le hace bien a la distribución de las variables, bien sea haciéndola más simétrica, reduciendo su kurtosis, etc.

Transformación logarítmica:

```
wizmir_log <- wizmir %>%
  mutate(across(where(is.numeric), ~log(.)))

head(wizmir_log)

max_temperature min_temperature dewpoint precipitation sea_level_pressure
1      4.479607        4.046554 3.981549      -Inf      3.399863
2      4.477337        4.070735 4.005513      -Inf      3.395850
3      4.517431        4.128746 4.100989      -Inf      3.393165
4      4.165114        3.756538 3.621671     -1.609438      3.406185
```

```

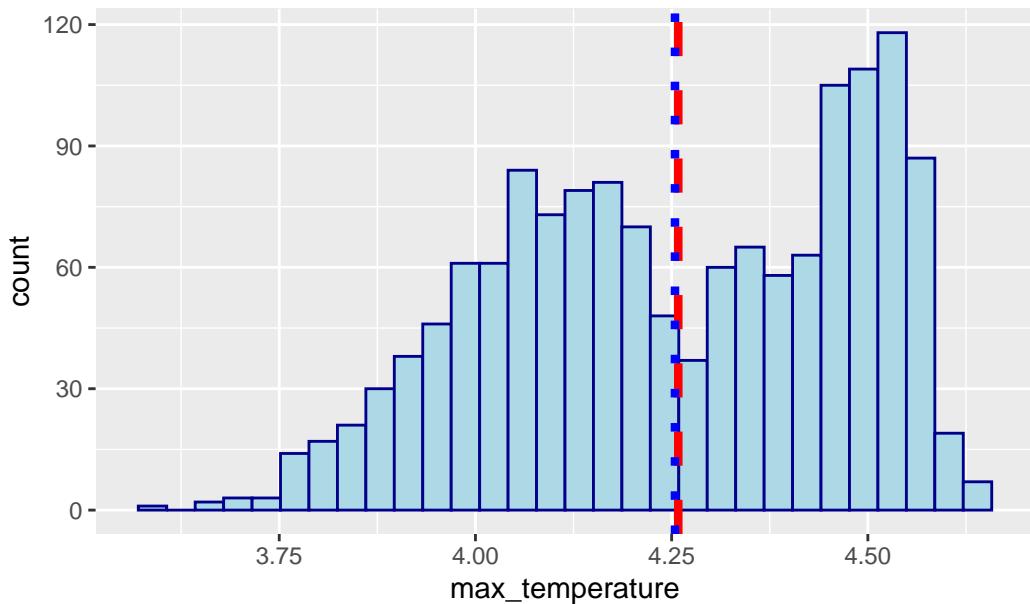
5      4.544358      4.280824 3.845883      -Inf      3.396520
6      4.398146      4.136765 3.621671      -Inf      3.401531
standard_pressure visibility wind_speed max_wind_speed mean_temperature
1      1.987874    2.207175 2.778819      3.534562      4.308111
2      1.987874    2.370244 2.906901      3.534562      4.320151
3      1.974081    2.115050 2.906901      3.534562      4.332048
4      2.054124    3.049273 3.314186      3.534562      3.852273
5      1.974081    2.844909 3.230804      3.534562      4.429626
6      1.916923    3.054001 3.314186      3.534562      4.259859

```

```
plot_hist(wizmir_log)
```

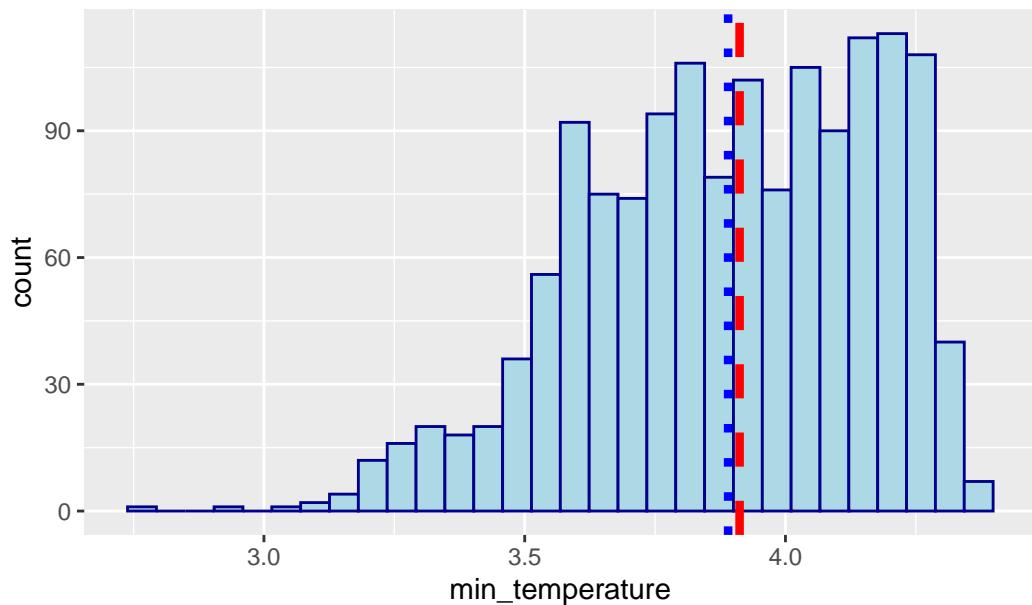
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – max_temperature



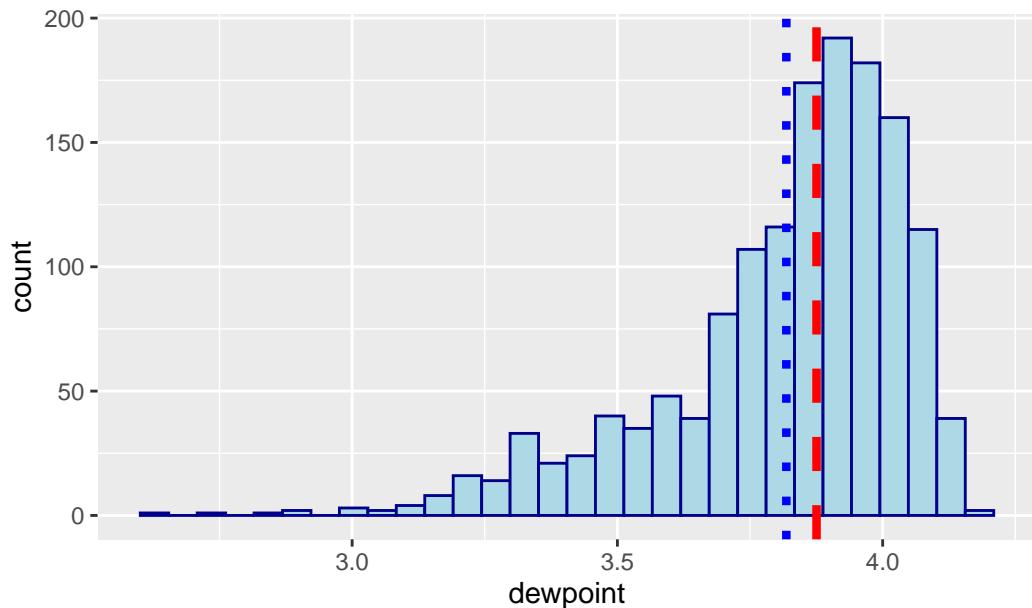
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – min_temperature



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

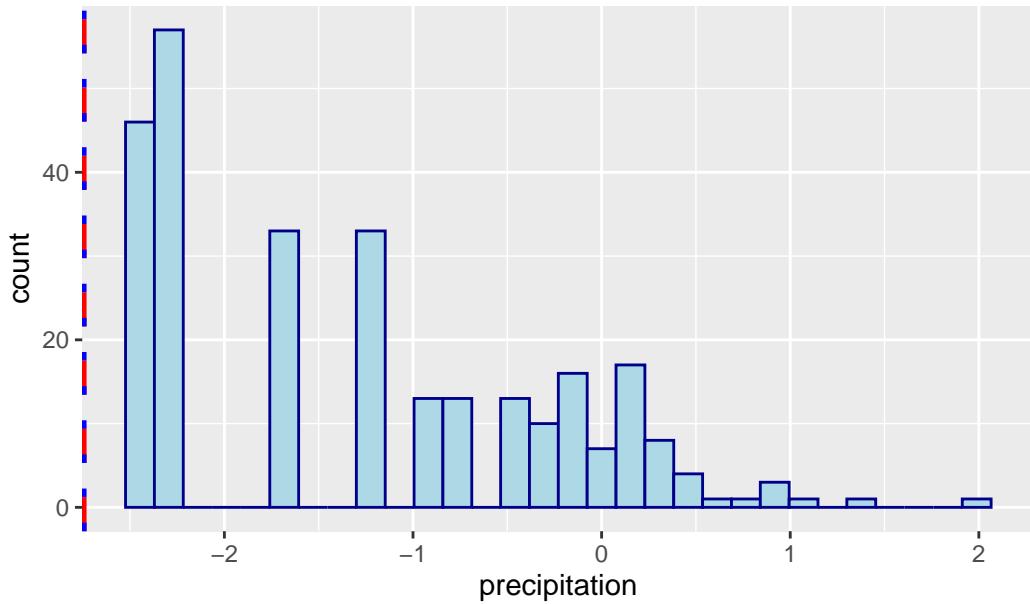
Curva de densidad – dewpoint



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

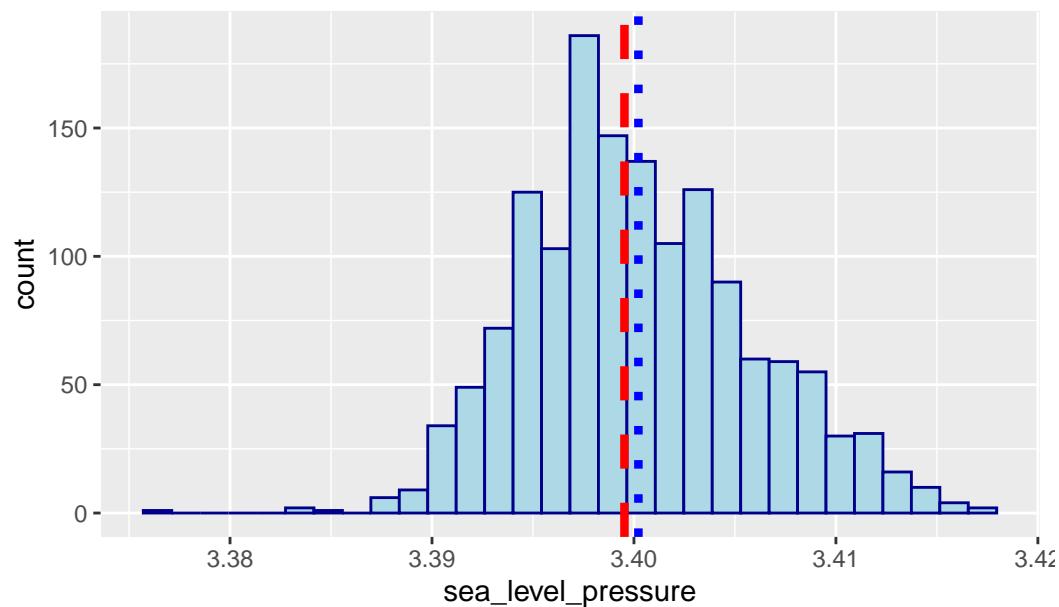
Warning: Removed 1182 rows containing non-finite values (`stat_bin()`).

Curva de densidad – precipitation



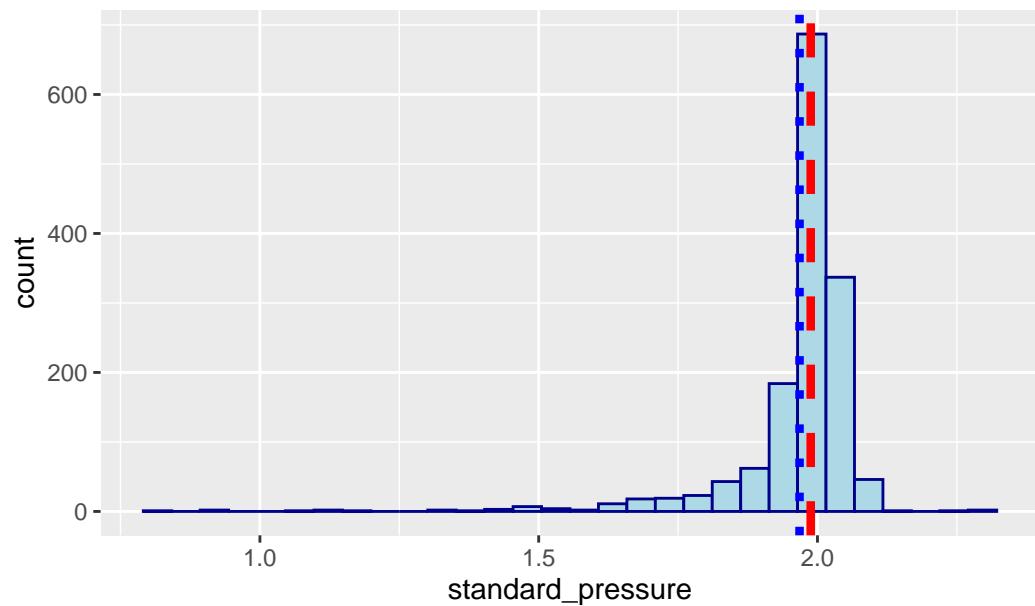
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – sea_level_pressure



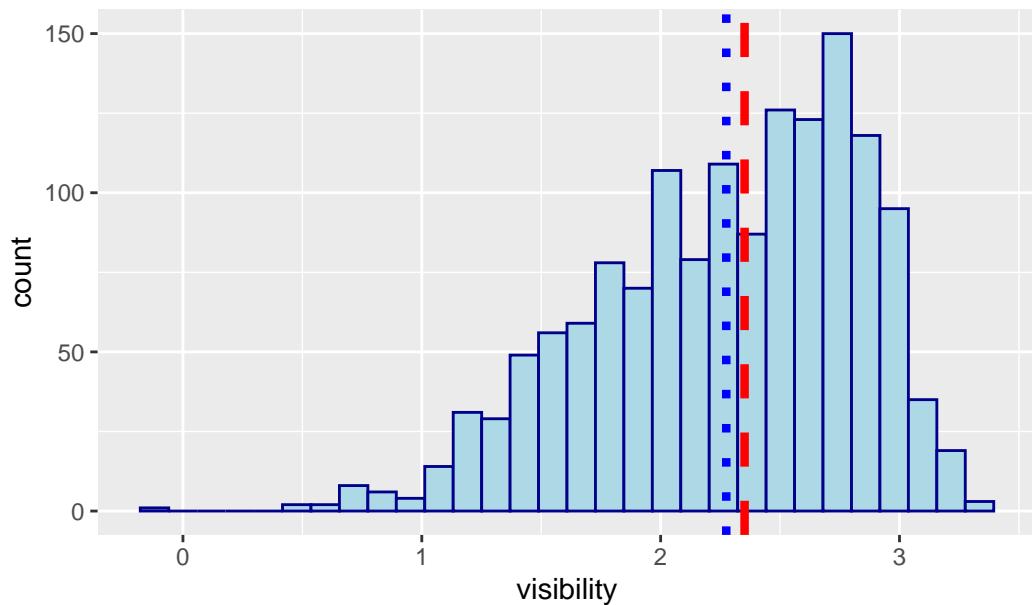
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – standard_pressure



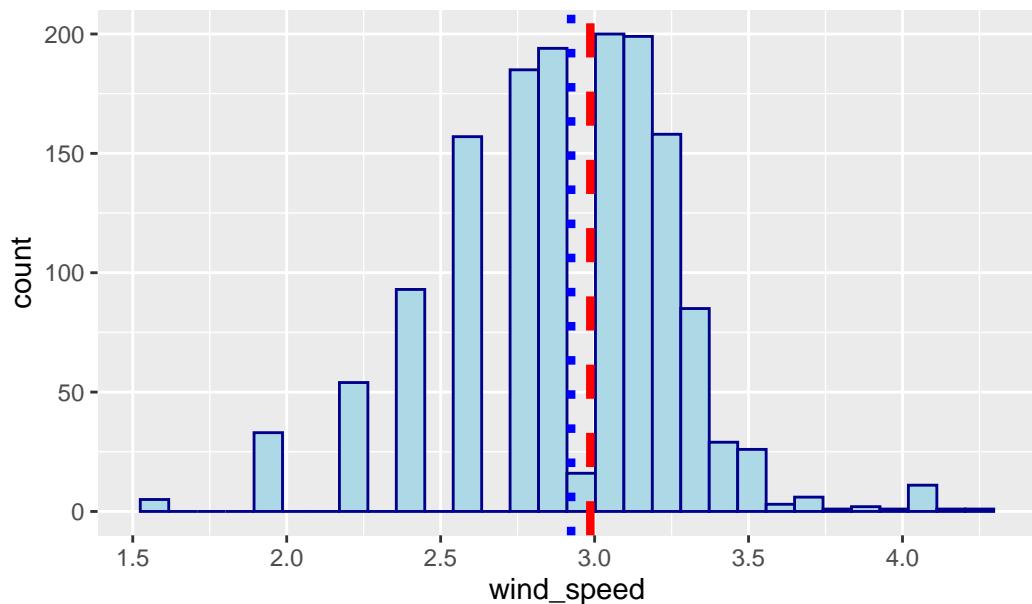
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – visibility



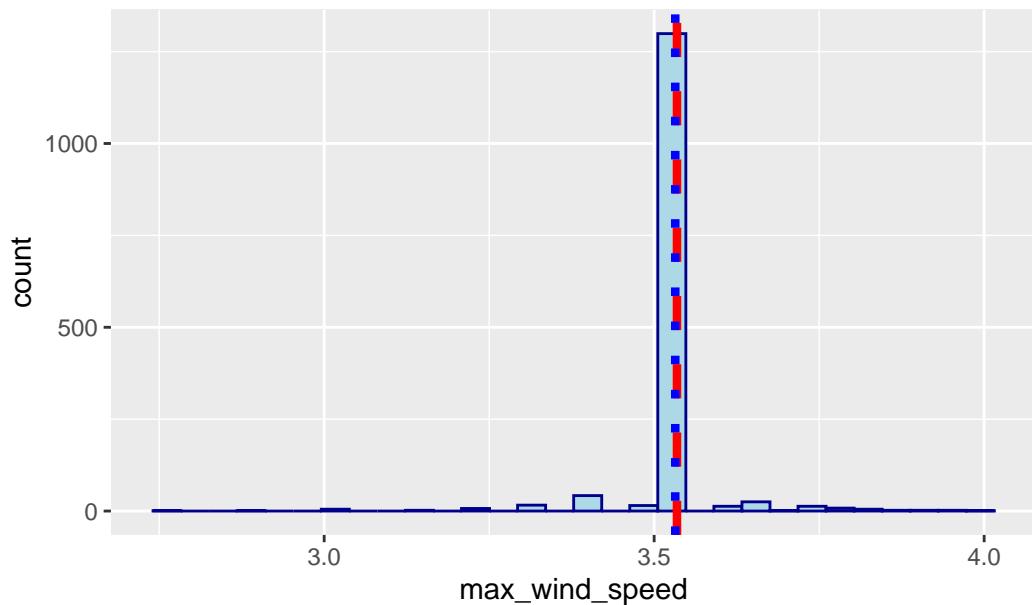
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – wind_speed



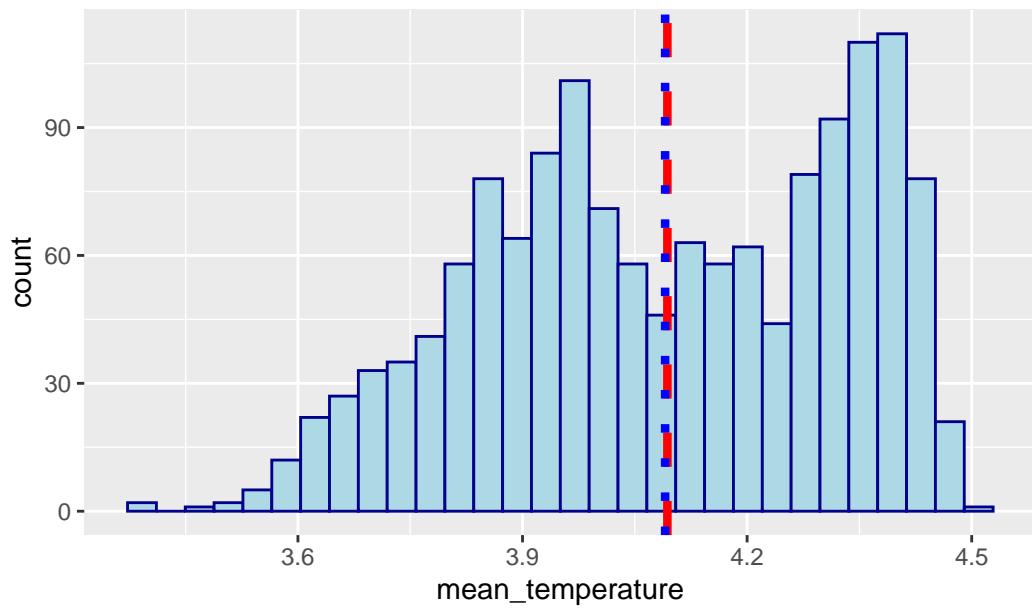
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – max_wind_speed



`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – mean_temperature



```

      max_temperature min_temperature dewpoint      precipitation
data      data.frame,10   data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~max_temperature ~min_temperature ~dewpoint ~precipitation
theme     list,0         list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?             ?
labels    list,5         list,5         list,5         list,5
                  sea_level_pressure standard_pressure visibility
data      data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~sea_level_pressure ~standard_pressure ~visibility
theme     list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?
labels    list,5         list,5         list,5
                  wind_speed      max_wind_speed mean_temperature
data      data.frame,10   data.frame,10   data.frame,10
layers    list,3         list,3         list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~wind_speed    ~max_wind_speed ~mean_temperature
theme     list,0         list,0         list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet     FacetNull,2   FacetNull,2   FacetNull,2
plot_env  ?             ?             ?
labels    list,5         list,5         list,5

```

Observaciones:

- La transformación logarítmica hace más semejante a una distribución normal las variables wind_speed

Transformación raíz cuadrada:

```

wizmir_sqrt <- wizmir %>%
  mutate(across(where(is.numeric), ~sqrt(.)))

head(wizmir_sqrt)

```

```

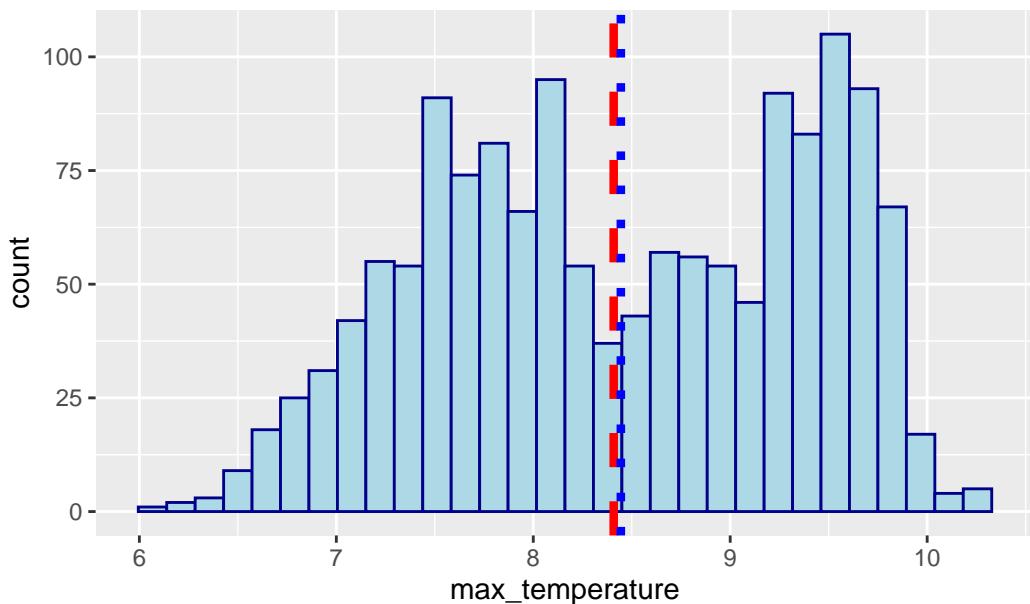
max_temperature min_temperature dewpoint precipitation sea_level_pressure
1      9.391486      7.563068 7.321202      0.0000000      5.473573
2      9.380832      7.655064 7.409453      0.0000000      5.462600
3      9.570789      7.880355 7.771744      0.0000000      5.455273
4      8.024961      6.542171 6.115554      0.4472136      5.490902
5      9.700515      8.502941 6.841053      0.0000000      5.464430
6      9.016651      7.912016 6.115554      0.0000000      5.478138
standard_pressure visibility wind_speed max_wind_speed mean_temperature
1      2.701851     3.014963 4.012481      5.854912     8.619745
2      2.701851     3.271085 4.277850      5.854912     8.671793
3      2.683282     2.879236 4.277850      5.854912     8.723531
4      2.792848     4.593474 5.244044      5.854912     6.862944
5      2.683282     4.147288 5.029911      5.854912     9.159694
6      2.607681     4.604346 5.244044      5.854912     8.414274

```

```
plot_hist(wizmir_sqrt)
```

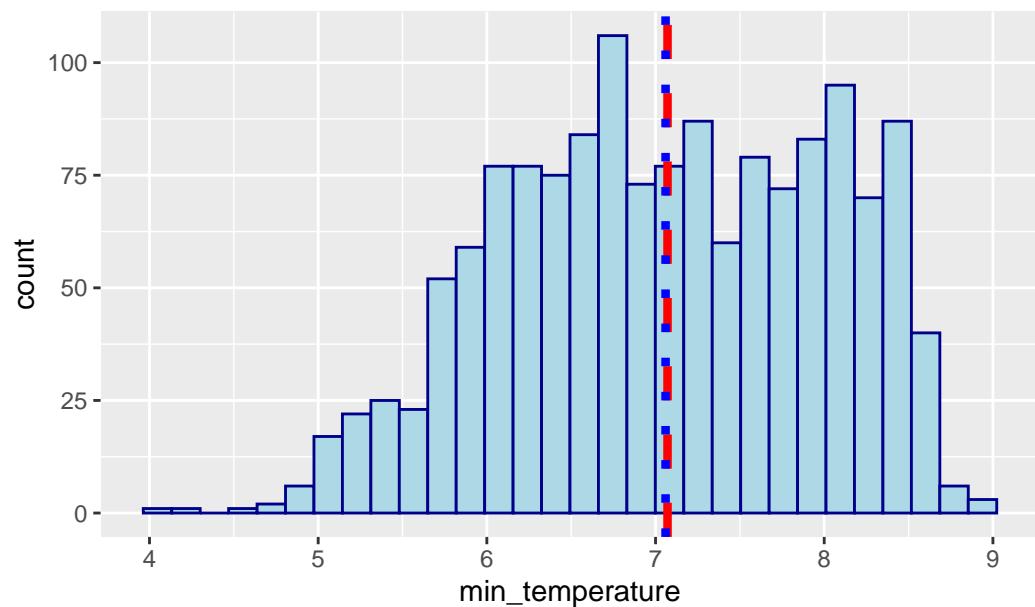
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – max_temperature



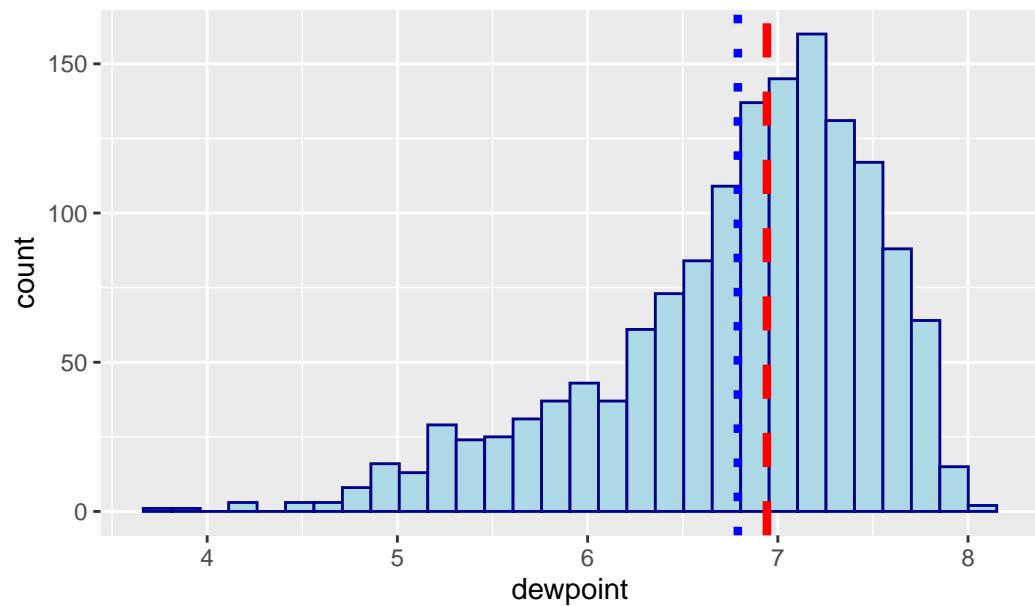
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – min_temperature



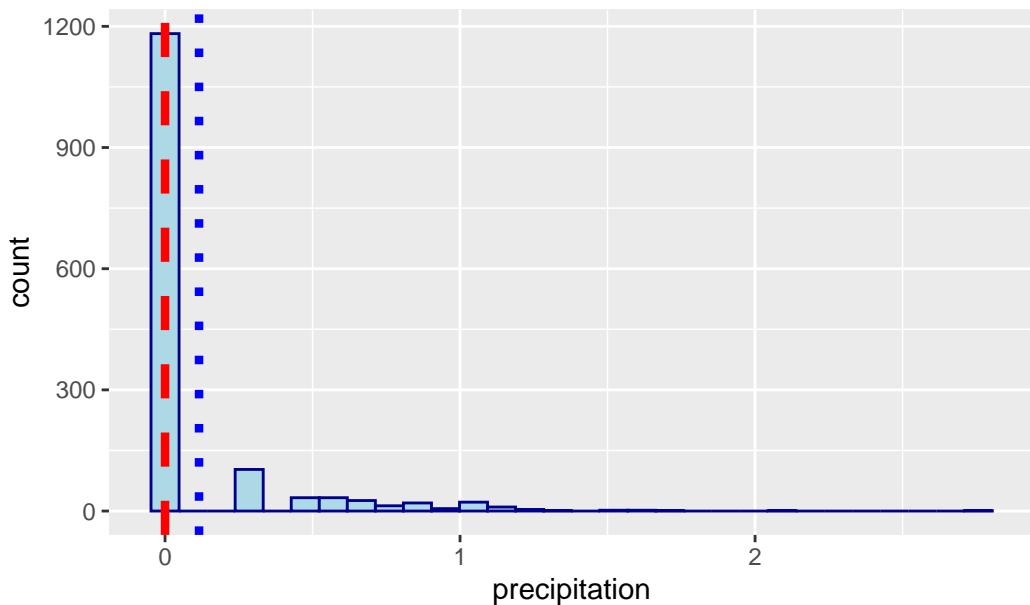
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – dewpoint



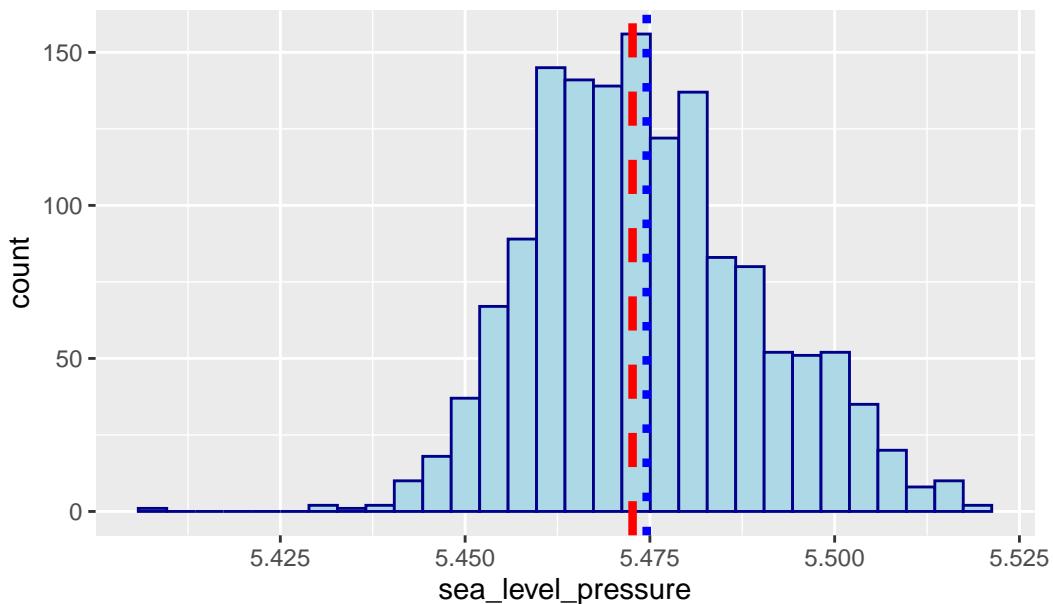
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – precipitation



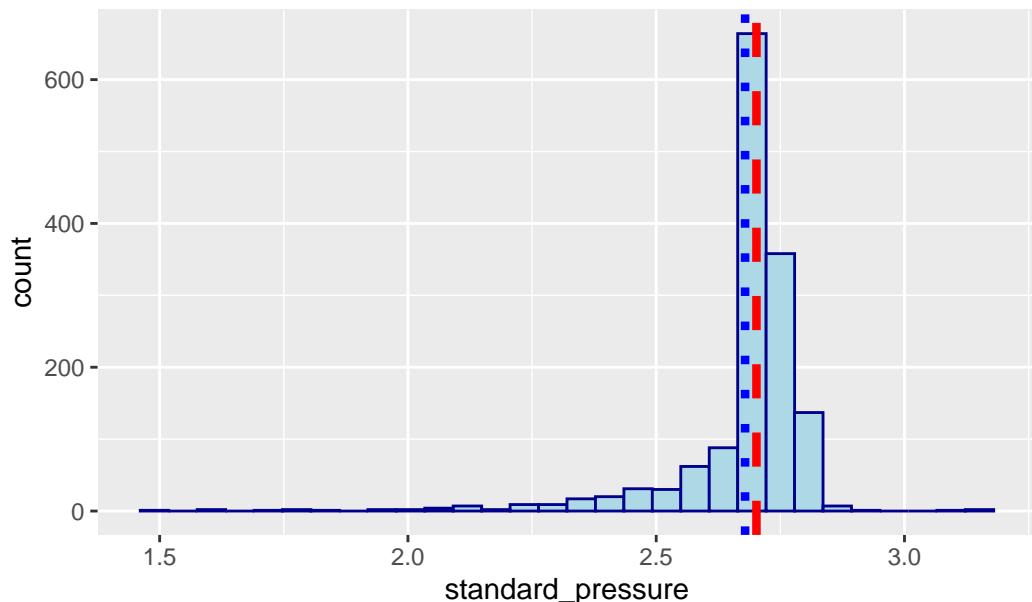
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – sea_level_pressure



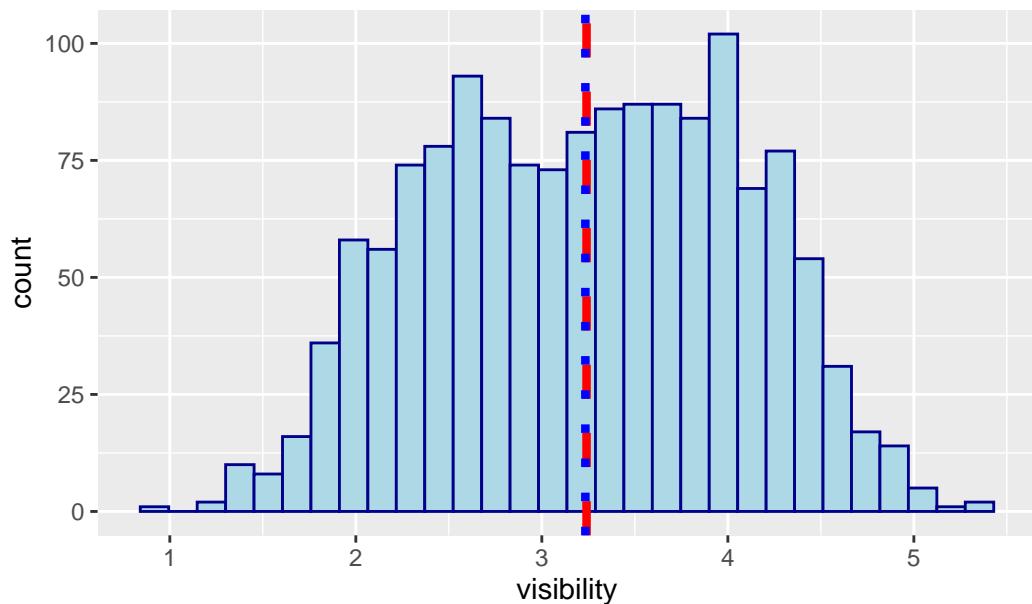
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – standard_pressure



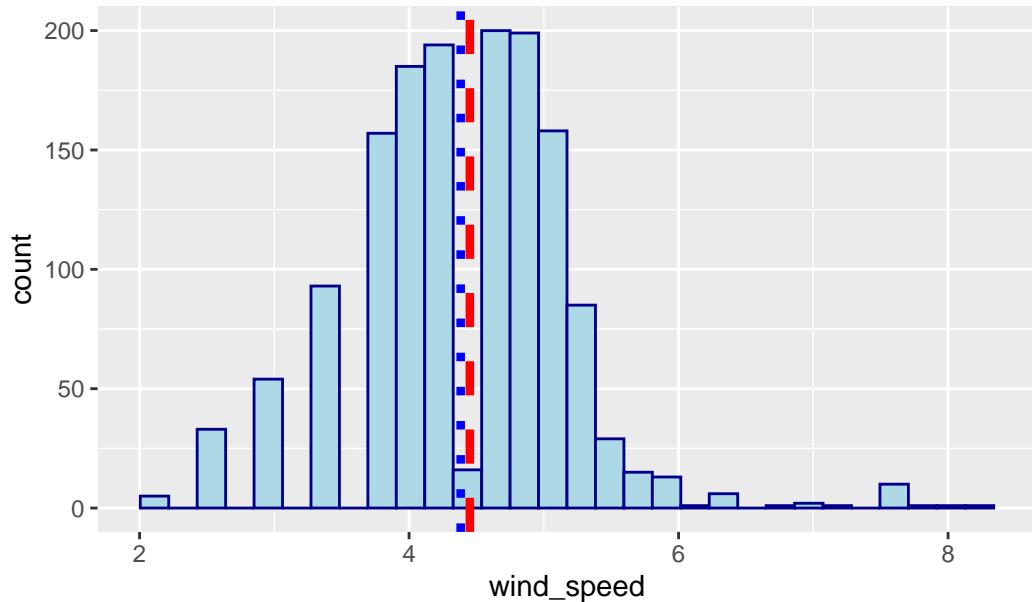
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – visibility



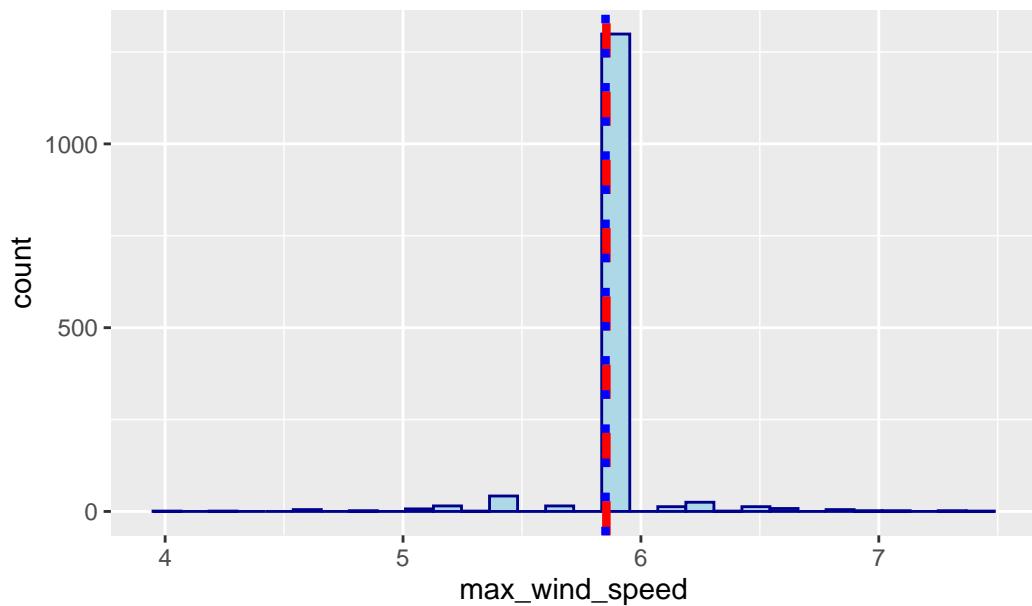
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – wind_speed



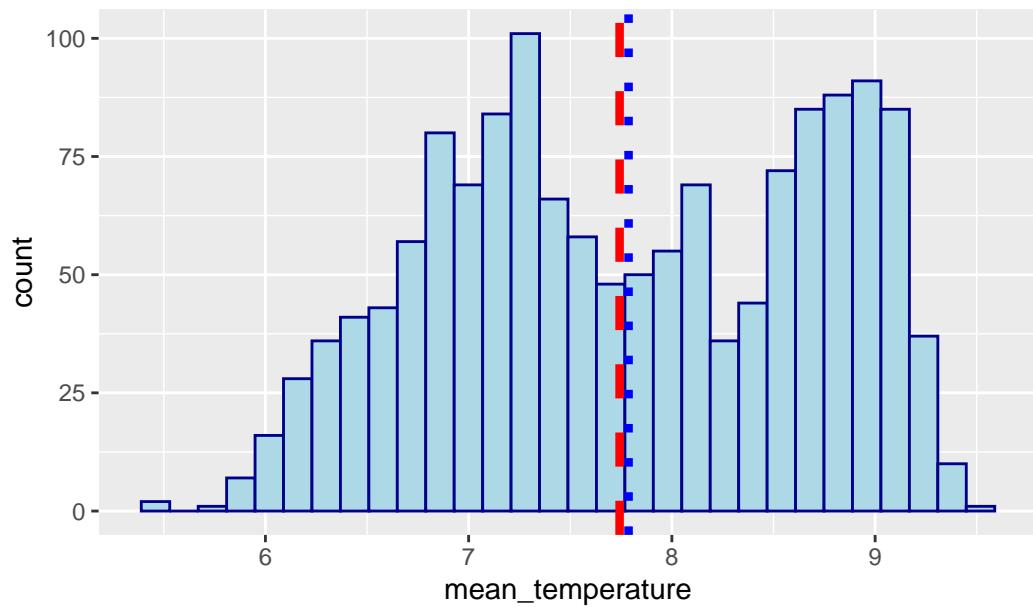
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – max_wind_speed



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – mean_temperature



```

      max_temperature min_temperature dewpoint precipitation
data      data.frame,10   data.frame,10   data.frame,10   data.frame,10
layers     list,3        list,3        list,3        list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~max_temperature ~min_temperature ~dewpoint ~precipitation
theme     list,0        list,0        list,0        list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2   FacetNull,2
plot_env    ?            ?            ?            ?
labels     list,5        list,5        list,5        list,5
          sea_level_pressure standard_pressure visibility
data      data.frame,10   data.frame,10   data.frame,10
layers     list,3        list,3        list,3
scales    ScalesList,2   ScalesList,2   ScalesList,2
mapping   ~sea_level_pressure ~standard_pressure ~visibility
theme     list,0        list,0        list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet      FacetNull,2   FacetNull,2   FacetNull,2
plot_env    ?            ?            ?
labels     list,5        list,5        list,5
          wind_speed      max_wind_speed  mean_temperature
data      data.frame,10   data.frame,10   data.frame,10
layers     list,3        list,3        list,3

```

```

scales      ScalesList,2      ScalesList,2      ScalesList,2
mapping     ~wind_speed       ~max_wind_speed ~mean_temperature
theme       list,0           list,0           list,0
coordinates CoordCartesian,5 CoordCartesian,5 CoordCartesian,5
facet       FacetNull,2      FacetNull,2      FacetNull,2
plot_env    ?                ?                ?
labels      list,5          list,5          list,5

```

Observaciones:

- La transformación raíz cuadrada hace más semejante a una distribución normal a la variable visibility

2.6 Conclusiones finales

2.6.1 Comprobación de las hipótesis

- ¿Las variables relacionadas con la temperatura son buenas candidatas para predecir la variable objetivo?
- ¿Existe una fuerte correlación entre las variables relacionadas con la temperatura?

Ambas hipótesis hemos comprobado que son ciertas mediante la exploración de relaciones entre variables.

2.6.2 Resumen del conjunto de datos

Tenemos un conjunto de datos con variables muy buenas para predecir la variable objetivo. Probablemente únicamente realizando un modelo lineal que solo consista en una de las variables relacionadas con la temperatura (max_temperature, min_temperature, dewpoint y sobre todo mean_min_max_temperature) será más que suficiente para obtener unos buenos resultados predictivos.

3 Regresión

3.1 Modelos lineales simples

En este primer apartado vamos a buscar las 5 mejores variables del conjunto de datos con las que hacer un modelo lineal simple, y de ellas buscaremos nos dé el mejor resultado en base a la métrica R^2 ajustado

A partir de la exploración de relaciones entre variables realizada en el EDA, donde mostramos cada variable independiente frente a la dependiente, sabemos que las variables candidatas para el ajuste lineal simple son: max_temperature, min_temperature, dewpoint, sea_level_pressure y visibility

Obtención dichos modelos lineales:

```
fit1 <- lm(mean_temperature ~ max_temperature, data = wizmir)
summary(fit1)
```

Call:

```
lm(formula = mean_temperature ~ max_temperature, data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-11.6062	-1.9017	0.3496	2.1409	7.4706

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-2.277746	0.359879	-6.329	3.27e-10 ***
max_temperature	0.883130	0.004867	181.446	< 2e-16 ***

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1 '	'	'	1

Residual standard error: 2.96 on 1458 degrees of freedom

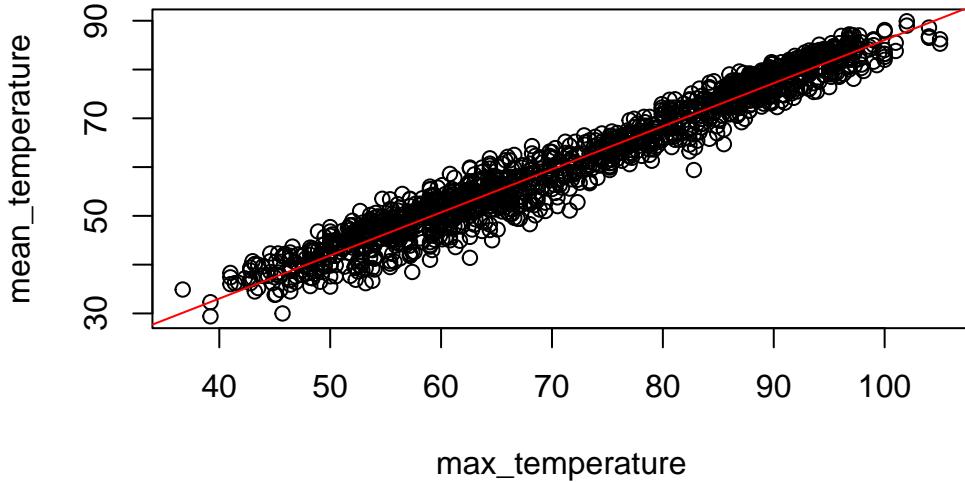
Multiple R-squared: 0.9576, Adjusted R-squared: 0.9576

F-statistic: 3.292e+04 on 1 and 1458 DF, p-value: < 2.2e-16

```
sqrt(sum(fit1$residuals ^ 2) / length(fit1$residuals))
```

[1] 2.957511

```
plot(mean_temperature ~ max_temperature, wizmir)
abline(fit1, col = "red")
```



```
confint(fit1)
```

	2.5 %	97.5 %
(Intercept)	-2.9836822	-1.571811
max_temperature	0.8735822	0.892677

```
fit2 <- lm(mean_temperature ~ min_temperature, data = wizmir)
summary(fit2)
```

Call:

```
lm(formula = mean_temperature ~ min_temperature, data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-9.1040	-3.0423	-0.2477	3.1461	16.8214

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	8.656385	0.423838	20.42	<2e-16 ***
min_temperature	1.041459	0.008085	128.82	<2e-16 ***

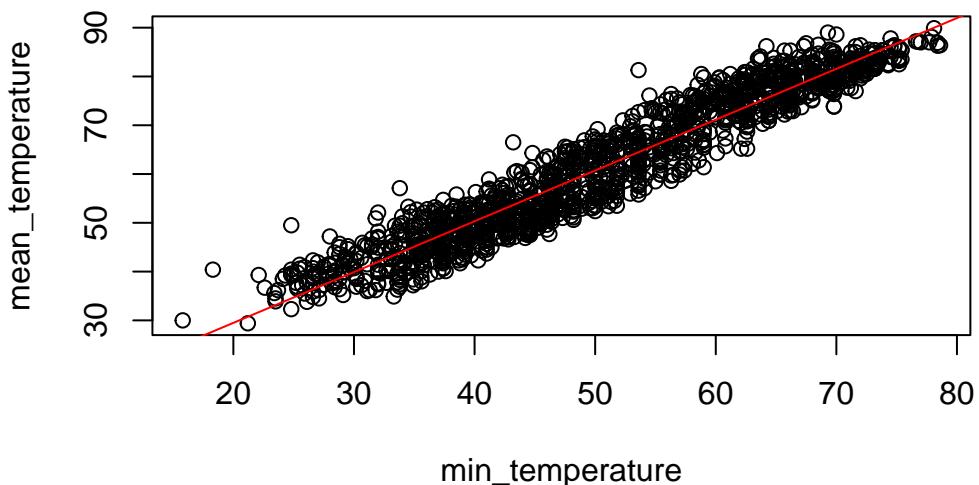
```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 4.084 on 1458 degrees of freedom
Multiple R-squared:  0.9192,    Adjusted R-squared:  0.9192
F-statistic: 1.659e+04 on 1 and 1458 DF,  p-value: < 2.2e-16
```

```
sqrt(sum(fit2$residuals ^ 2) / length(fit2$residuals))
```

```
[1] 4.081431
```

```
plot(mean_temperature ~ min_temperature, wizmir)
abline(fit2, col = "red")
```



```
confint(fit2)
```

	2.5 %	97.5 %
(Intercept)	7.824987	9.487783
min_temperature	1.025601	1.057318

```
fit3 <- lm(mean_temperature ~ dewpoint, data = wizmir)
summary(fit3)

Call:
lm(formula = mean_temperature ~ dewpoint, data = wizmir)

Residuals:
    Min      1Q  Median      3Q     Max 
-15.517 -6.963 -1.099  5.442 29.696 

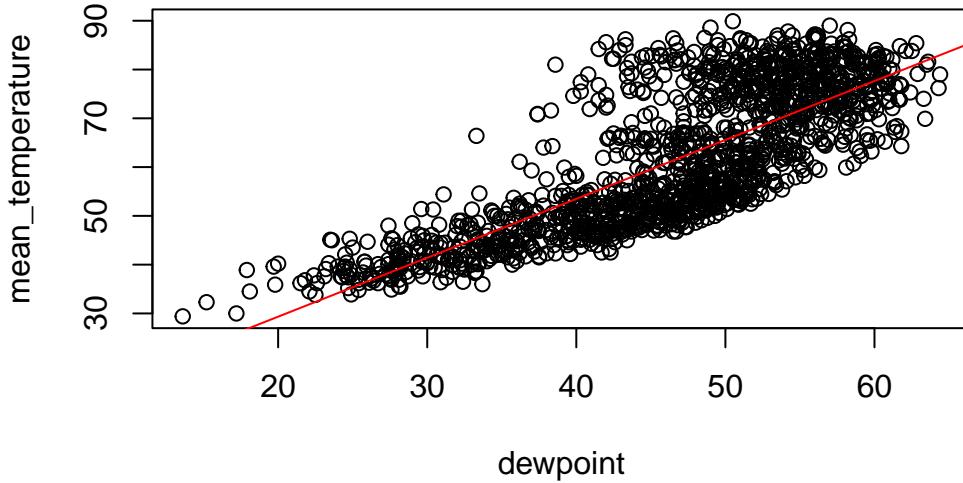
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 5.1796     1.1839   4.375  1.3e-05 ***
dewpoint    1.2077     0.0249  48.510 < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 8.889 on 1458 degrees of freedom
Multiple R-squared:  0.6174,    Adjusted R-squared:  0.6172 
F-statistic: 2353 on 1 and 1458 DF,  p-value: < 2.2e-16
```

```
sqrt(sum(fit3$residuals ^ 2) / length(fit3$residuals))
```

```
[1] 8.882801
```

```
plot(mean_temperature ~ dewpoint, wizmir)
abline(fit3, col = "red")
```



```
confint(fit3)
```

	2.5 %	97.5 %
(Intercept)	2.857342	7.501938
dewpoint	1.158886	1.256558

```
fit4 <- lm(mean_temperature ~ sea_level_pressure, data = wizmir)
summary(fit4)
```

Call:

```
lm(formula = mean_temperature ~ sea_level_pressure, data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-45.903	-8.244	1.149	9.456	23.857

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	1558.100	54.623	28.52	<2e-16 ***
sea_level_pressure	-49.935	1.822	-27.40	<2e-16 ***

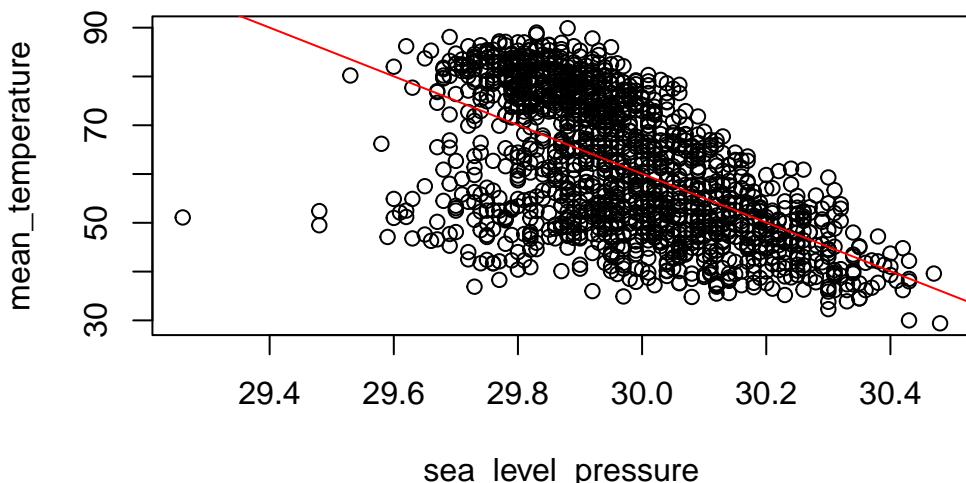
```
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 11.68 on 1458 degrees of freedom
Multiple R-squared:  0.3399,    Adjusted R-squared:  0.3394
F-statistic: 750.7 on 1 and 1458 DF,  p-value: < 2.2e-16
```

```
sqrt(sum(fit4$residuals ^ 2) / length(fit4$residuals))
```

```
[1] 11.66847
```

```
plot(mean_temperature ~ sea_level_pressure, wizmir)
abline(fit4, col = "red")
```



```
confint(fit4)
```

	2.5 %	97.5 %
(Intercept)	1450.95137	1665.24765
sea_level_pressure	-53.50994	-46.35997

```
fit5 <- lm(mean_temperature ~ visibility, data = wizmir)
summary(fit5)

Call:
lm(formula = mean_temperature ~ visibility, data = wizmir)

Residuals:
    Min      1Q  Median      3Q     Max 
-37.314 -10.872 - 0.644 13.124 29.507 

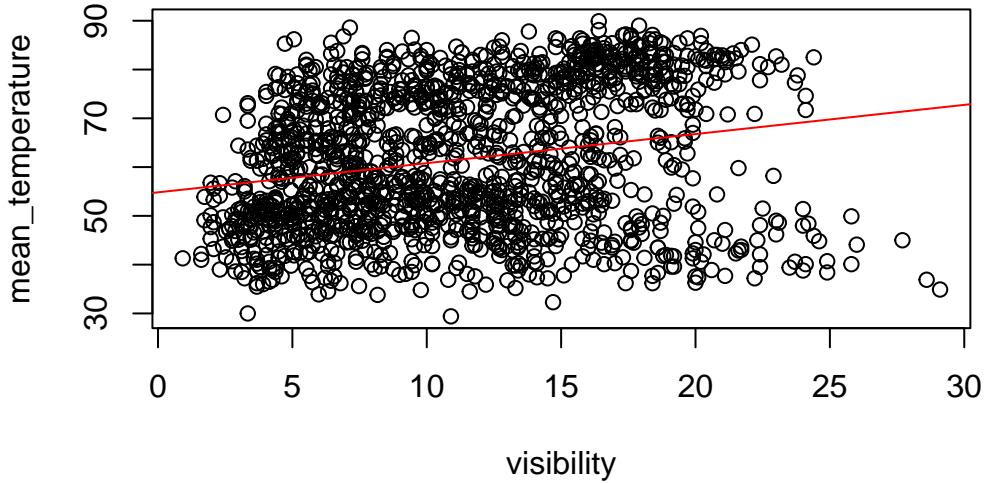
Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 54.82671   0.84058 65.225 <2e-16 ***
visibility   0.59750   0.06783  8.809 <2e-16 ***  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 14 on 1458 degrees of freedom
Multiple R-squared:  0.05054, Adjusted R-squared:  0.04988 
F-statistic: 77.6 on 1 and 1458 DF, p-value: < 2.2e-16
```

```
sqrt(sum(fit5$residuals ^ 2) / length(fit5$residuals))
```

```
[1] 13.99409
```

```
plot(mean_temperature ~ visibility, wizmir)
abline(fit5, col = "red")
```



```
confint(fit5)
```

	2.5 %	97.5 %
(Intercept)	53.1778406	56.4755777
visibility	0.4644508	0.7305449

Entre todos los modelos, el modelo creado a partir de la variable max_temperature es el que obtiene el mejor valor de R^2 ajustado, por tanto este es el elegido.

3.2 Modelos lineales múltiples

3.2.1 Sin interacción entre variables

En este apartado vamos a obtener distintos modelos lineales múltiples. Comenzaremos con el modelo que utiliza todas las variables e iremos quitando las variables que aporten menos al modelo hasta obtener un modelo que tenga un compromiso entre interpretabilidad (número de variables) y calidad.

```
mulfit1 <- lm(mean_temperature ~ ., data = wizmir)
summary(mulfit1)
```

```

Call:
lm(formula = mean_temperature ~ ., data = wizmir)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2318 -0.7033 -0.0361  0.7099  6.1892 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 56.958796   8.183942   6.960 5.14e-12 ***
max_temperature 0.573791   0.005739  99.988 < 2e-16 ***
min_temperature 0.367818   0.009225  39.870 < 2e-16 ***
dewpoint        0.045905   0.007271   6.314 3.61e-10 ***
precipitation   -0.068914   0.101707  -0.678 0.498153  
sea_level_pressure -1.920036   0.266726  -7.199 9.74e-13 ***
standard_pressure 0.064853   0.050837   1.276 0.202264  
visibility       0.084878   0.012237   6.936 6.06e-12 ***
wind_speed       0.002793   0.007200   0.388 0.698129  
max_wind_speed   -0.047163   0.014104  -3.344 0.000847 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.25 on 1450 degrees of freedom
Multiple R-squared:  0.9925,    Adjusted R-squared:  0.9924 
F-statistic: 2.126e+04 on 9 and 1450 DF,  p-value: < 2.2e-16

```

Teniendo en cuenta los códigos de significancia, las dos variables que menos importantes para el modelo son wind_speed y visibility. Eliminamos visibility.

```

mulfit2 <- lm(mean_temperature ~ . - visibility, data = wizmir)
summary(mulfit2)

```

```

Call:
lm(formula = mean_temperature ~ . - visibility, data = wizmir)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.3018 -0.7170 -0.0416  0.7108  6.5685 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 54.785824   8.309633   6.593 6.02e-11 ***

```

```

max_temperature      0.556678  0.005265 105.741 < 2e-16 ***
min_temperature     0.406469  0.007471  54.408 < 2e-16 ***
dewpoint            0.024330  0.006677   3.644 0.000278 ***
precipitation       -0.146782 0.102713  -1.429 0.153207
sea_level_pressure -1.854955 0.270854  -6.849 1.10e-11 ***
standard_pressure   0.110035  0.051230   2.148 0.031889 *
wind_speed          0.033531  0.005766   5.815 7.44e-09 ***
max_wind_speed     -0.031903 0.014156  -2.254 0.024367 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 1.27 on 1451 degrees of freedom
Multiple R-squared: 0.9922, Adjusted R-squared: 0.9922
F-statistic: 2.316e+04 on 8 and 1451 DF, p-value: < 2.2e-16

Como vemos, no se ha producido casi ningun empeoramiento en la métrica R^2 por lo que ha sido una buena eliminación de variable. La siguiente que vamos a quitar es la variable precipitation pues es la que tiene el peor código de significancia en este nuevo modelo.

```
mulfit3 <- lm(mean_temperature ~ . - visibility - precipitation, data = wizmir)
summary(mulfit3)
```

Call:
`lm(formula = mean_temperature ~ . - visibility - precipitation,
 data = wizmir)`

Residuals:

Min	1Q	Median	3Q	Max
-8.2996	-0.7259	-0.0331	0.7129	6.5646

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	52.481821	8.154648	6.436	1.66e-10 ***
max_temperature	0.558812	0.005050	110.652	< 2e-16 ***
min_temperature	0.405686	0.007453	54.431	< 2e-16 ***
dewpoint	0.023192	0.006632	3.497	0.000485 ***
sea_level_pressure	-1.780034	0.265827	-6.696	3.05e-11 ***
standard_pressure	0.105754	0.051160	2.067	0.038900 *
wind_speed	0.033960	0.005761	5.895	4.64e-09 ***
max_wind_speed	-0.031730	0.014161	-2.241	0.025196 *

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 1.27 on 1452 degrees of freedom
Multiple R-squared:  0.9922,    Adjusted R-squared:  0.9922
F-statistic: 2.645e+04 on 7 and 1452 DF,  p-value: < 2.2e-16

```

Como vemos, el R^2 ajustado no ha variado, por lo que esa variable no aportaba nada. La siguiente variable que vamos a eliminar es standard_pressure, pues es la que tiene el peor código de significancia junto con max_wind_speed.

```

mulfit4 <- lm(mean_temperature ~ . - visibility - precipitation - standard_pressure,
               data = wizmir)
summary(mulfit4)

```

Call:

```
lm(formula = mean_temperature ~ . - visibility - precipitation -
   standard_pressure, data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-8.2114	-0.7133	-0.0320	0.7122	6.6009

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	54.816397	8.085152	6.780	1.74e-11 ***
max_temperature	0.558204	0.005047	110.595	< 2e-16 ***
min_temperature	0.407392	0.007416	54.936	< 2e-16 ***
dewpoint	0.021915	0.006611	3.315	0.000939 ***
sea_level_pressure	-1.832846	0.264894	-6.919	6.79e-12 ***
wind_speed	0.035603	0.005712	6.233	5.98e-10 ***
max_wind_speed	-0.031916	0.014176	-2.251	0.024512 *

Signif. codes:	0 '***'	0.001 '**'	0.01 '*'	0.05 '.'
	0.1 '	' 1		

```

Residual standard error: 1.272 on 1453 degrees of freedom
Multiple R-squared:  0.9922,    Adjusted R-squared:  0.9922
F-statistic: 3.079e+04 on 6 and 1453 DF,  p-value: < 2.2e-16

```

De nuevo, no se produce empeoramiento en el R^2 ajustado, por lo que esa variable no aportaba nada. Siguiendo la lógica que estamos siguiendo ahora eliminamos max_wind_speed

```

mulfit5 <- lm(mean_temperature ~ . - visibility - precipitation - standard_pressure - max_
               data = wizmir)
summary(mulfit5)

Call:
lm(formula = mean_temperature ~ . - visibility - precipitation -
    standard_pressure - max_wind_speed, data = wizmir)

Residuals:
    Min      1Q  Median      3Q     Max 
-8.2425 -0.7285 -0.0256  0.7145  6.5851 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 53.908307   8.086374   6.667 3.71e-11 ***
max_temperature 0.559174   0.005036  111.037 < 2e-16 ***
min_temperature 0.407230   0.007426   54.840 < 2e-16 ***
dewpoint        0.021755   0.006620   3.286  0.00104 **  
sea_level_pressure -1.839180   0.265249  -6.934 6.14e-12 ***
wind_speed       0.033057   0.005607   5.896 4.62e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.274 on 1454 degrees of freedom
Multiple R-squared:  0.9922,    Adjusted R-squared:  0.9921 
F-statistic: 3.684e+04 on 5 and 1454 DF,  p-value: < 2.2e-16

```

Como podemos observar, no obtenemos casi nada de empeoramiento en el R^2 ajustado, por lo que podemos eliminarla sin problemas, La siguiente que vamos a eliminar es dewpoint pues es la variable con peor código de significancia.

```

mulfit6 <- lm(mean_temperature ~ . - visibility - precipitation - standard_pressure - max_
               data = wizmir)
summary(mulfit6)

```

```

Call:
lm(formula = mean_temperature ~ . - visibility - precipitation -
    standard_pressure - max_wind_speed - dewpoint, data = wizmir)

Residuals:

```

```

      Min       1Q   Median     3Q      Max
-8.2372 -0.7275 -0.0426  0.7432  6.7789

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) 61.124762  7.808707  7.828 9.49e-15 ***
max_temperature 0.560262  0.005042 111.121 < 2e-16 ***
min_temperature 0.417444  0.006767  61.691 < 2e-16 ***
sea_level_pressure -2.060969  0.257383 -8.007 2.38e-15 ***
wind_speed      0.025405  0.005118   4.964 7.71e-07 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.278 on 1455 degrees of freedom
Multiple R-squared: 0.9921, Adjusted R-squared: 0.9921
F-statistic: 4.574e+04 on 4 and 1455 DF, p-value: < 2.2e-16

```

De nuevo, no se produce empeoramiento en el R^2 ajustado, por lo que esa variable no aportaba nada que no tenga ya el modelo. Las variables que nos quedan tiene todas altos códigos de significancia por lo que son importantes, sin embargo a partir del EDA que hemos realizado anteriormente, sabemos que existe una muy alta correlación entre max_temperature, min_temperature y sea_level_pressure, por lo que si eliminamos una de ellas posiblemente no haya mucho empeoramiento, y además paliamos el problema de multicolinearidad, lo probamos eliminando sea_level_pressure:

```

mulfit7 <- lm(mean_temperature ~ . - visibility - precipitation - standard_pressure - max_
                     data = wizmir)
summary(mulfit7)

```

```

Call:
lm(formula = mean_temperature ~ . - visibility - precipitation -
    standard_pressure - max_wind_speed - dewpoint - sea_level_pressure,
    data = wizmir)

Residuals:
      Min       1Q   Median     3Q      Max
-8.2215 -0.7719 -0.0673  0.7451  7.1202

```

```

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.385537  0.186452 -7.431 1.83e-13 ***
max_temperature 0.555133  0.005108 108.672 < 2e-16 ***

```

```

min_temperature 0.439213  0.006329  69.391 < 2e-16 ***
wind_speed      0.025741  0.005227  4.924 9.42e-07 ***
---
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Residual standard error: 1.305 on 1456 degrees of freedom
Multiple R-squared: 0.9918, Adjusted R-squared: 0.9917
F-statistic: 5.844e+04 on 3 and 1456 DF, p-value: < 2.2e-16

Como hemos podido observar, casi no hemos obtenido empeoramiento aun teniendo la variable un alto código de significancia, esto es debido a la correlación que existe entre las variables. Probamos a eliminar también min_temperature debido a la correlación:

```

mulfit8 <- lm(mean_temperature ~ . - visibility - precipitation - standard_pressure - max_
               data = wizmir)
summary(mulfit8)

```

Call:

```

lm(formula = mean_temperature ~ . - visibility - precipitation -
    standard_pressure - max_wind_speed - dewpoint - sea_level_pressure -
    min_temperature, data = wizmir)

```

Residuals:

Min	1Q	Median	3Q	Max
-10.7467	-1.6456	0.1211	1.8106	8.1050

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-5.133731	0.370235	-13.87	<2e-16 ***
max_temperature	0.876528	0.004470	196.07	<2e-16 ***
wind_speed	0.168250	0.009972	16.87	<2e-16 ***

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.708 on 1457 degrees of freedom
Multiple R-squared: 0.9645, Adjusted R-squared: 0.9645
F-statistic: 1.981e+04 on 2 and 1457 DF, p-value: < 2.2e-16

En este caso el empeoramiento ha sido más alto, entre un 2 y un 3%. Dado que este empeoramiento es considerable en comparación con los anteriores paramos aquí y nos quedamos con el modelo mulfit7. Este es un modelo que usa sólo 3 variables, lo cual significa que tienen una muy alta interpretabilidad, y además tiene unas buenas métricas.

3.2.2 Con interacción entre variables y transformaciones

El modelo que hemos obtenido anteriormente (mulfit7) tiene dos variables que tienen una alta correlación entre ambas, además sabemos por el EDA que estas por sí solas tienen una relación lineal muy fuerte con la variable objetivo. Si hacemos una interacción entre ambas seguramente obtengamos un modelo muy bueno para la tarea de predicción, y además nos quitamos el problema de la correlación entre las dos variables. Tal y como decíamos en la parte de ingeniería de características del EDA, si creamos una variable a partir de la media de la temperatura máxima y la mínima obtendremos una variable que estime bastante bien la temperatura media. Lo probamos.

```
mulfit9 <- lm(mean_temperature ~ I((max_temperature+min_temperature)/2),
               data = wizmir)
summary(mulfit9)
```

Call:

```
lm(formula = mean_temperature ~ I((max_temperature + min_temperature)/2),
   data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-7.9670	-0.8384	-0.0812	0.7008	8.0035

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.457655	0.157366	-2.908	0.00369
I((max_temperature + min_temperature)/2)	1.007795	0.002494	404.007	< 2e-16
(Intercept)		**		
I((max_temperature + min_temperature)/2)		***		

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.352 on 1458 degrees of freedom

Multiple R-squared: 0.9911, Adjusted R-squared: 0.9911

F-statistic: 1.632e+05 on 1 and 1458 DF, p-value: < 2.2e-16

Tal como decíamos, hemos obtenido un modelo con una sola variable que nos da prácticamente los mismo resultados que obteníamos con el modelo fitmul7.

Vamos a probar otra interacción. A partir del EDA hemos visto que la transformación logarítmica ayuda a la variable wind_speed a tener una distribución más parecida a una

distribución normal. Lo probamos con el modelo fitmul9 aplicando el logaritmo y sin aplicarlo a la variable wind_speed

```
mulfit10 <- lm(mean_temperature ~ I((max_temperature+min_temperature)/2) + wind_speed,  
                 data = wizmir)  
summary(mulfit10)
```

Call:

```
lm(formula = mean_temperature ~ I((max_temperature + min_temperature)/2) +  
    wind_speed, data = wizmir)
```

Residuals:

Min	1Q	Median	3Q	Max
-7.9159	-0.8366	-0.0957	0.7167	7.9870

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	-0.540799	0.173776	-3.112	0.00189
I((max_temperature + min_temperature)/2)	1.007320	0.002530	398.211	< 2e-16
wind_speed	0.005672	0.005031	1.127	0.25974

(Intercept) **
I((max_temperature + min_temperature)/2) ***
wind_speed

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.352 on 1457 degrees of freedom

Multiple R-squared: 0.9912, Adjusted R-squared: 0.9911

F-statistic: 8.163e+04 on 2 and 1457 DF, p-value: < 2.2e-16

```
mulfit11 <- lm(mean_temperature ~ I((max_temperature+min_temperature)/2) + I(log(wind_spee  
data = wizmir)  
summary(mulfit11)
```

Call:

```
lm(formula = mean_temperature ~ I((max_temperature + min_temperature)/2) +  
    I(log(wind_speed)), data = wizmir)
```

Residuals:

```

      Min       1Q   Median     3Q      Max
-7.8476 -0.8431 -0.1027  0.7298  7.9775

Coefficients:
                                         Estimate Std. Error t value Pr(>|t|)
(Intercept)                         -1.032120  0.302251 -3.415 0.000656
I((max_temperature + min_temperature)/2) 1.006489  0.002559 393.261 < 2e-16
I(log(wind_speed))                  0.223935  0.100644  2.225 0.026232

(Intercept)                         ***
I((max_temperature + min_temperature)/2) ***
I(log(wind_speed))                  *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.35 on 1457 degrees of freedom
Multiple R-squared:  0.9912,    Adjusted R-squared:  0.9912
F-statistic: 8.183e+04 on 2 and 1457 DF,  p-value: < 2.2e-16

```

Como podemos ver obtenemos una ligera mejora usando el logaritmo frente a no usarlo en la variable wind_speed.

3.3 KNN en problemas de regresión

En este apartado vamos a usar algoritmo KNN para regresión. Vamos crear un modelo con todas las variables tanto para KNN como para regresión y vamos a comparar el error cuadrático medio que obtenemos tanto en entrenamiento como en test.

```

library(kknn)

nombre <- "wizmir/wizmir"
run_knn_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@", header=FALSE)
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@", header=FALSE)
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
}

```

```

if (tt == "train") {
  test <- x_tra
}
else {
  test <- x_tst
}
fitMulti=kknn(Y~.,x_tra,test)
yprime=fitMulti$fitted.values
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}

knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))
knnMSEtrain

```

[1] 2.538487

```
knnMSEtest
```

[1] 6.065217

```

nombre <- "wizmir/wizmir"
run_lm_fold <- function(i, x, tt = "test") {
  file <- paste(x, "-5-", i, "tra.dat", sep="")
  x_tra <- read.csv(file, comment.char="@", header=FALSE)
  file <- paste(x, "-5-", i, "tst.dat", sep="")
  x_tst <- read.csv(file, comment.char="@", header=FALSE)
  In <- length(names(x_tra)) - 1
  names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tra)[In+1] <- "Y"
  names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
  names(x_tst)[In+1] <- "Y"
  if (tt == "train") {
    test <- x_tra
  }
  else {
    test <- x_tst
  }
  fitMulti=lm(Y~.,x_tra)
}
```

```

yprime=predict(fitMulti,test)
sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}

lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))

lmMSEtrain

[1] 1.563401

lmMSEtest

[1] 1.600421

```

Podemos observar que para este problema, el modelo lineal múltiple obtiene un mejor desempeño que el modelo usando KNN.

3.4 Comparativa de algoritmos

Este apartado consiste en utilizar las particiones 5-fcv para su dataset con k-NN y con la regresión lineal múltiple con todas las variables. Debemos Reemplazar los resultados de las tablas disponibles en regr_test_alumnos.csv y en regr_train_alumnos.csv para el dataset wizmir.

3.4.1 Comparativas por pares de LM y KNN (test de Wilcoxon)

Leemos la tabla con los errores medios tanto de test como de entrenamiento.

```

resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]

```

```

##lm (other) vs knn (ref)
# + 0.1 porque wilcox R falla para valores == 0 en la tabla
difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
head(wilc_1_2)

      out_test_lm out_test_kknn
[1,] 0.1909091   0.1000000
[2,] 0.1000000   1.0294118
[3,] 0.1000000   0.4339071
[4,] 0.1000000   0.3885965
[5,] 0.1548506   0.1000000
[6,] 0.1000000   0.3061057

```

Se aplica test y se interpreta los resultados.

```

LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic

Rmas

```

V
78

Rmenos

V
93

pvalue

[1] 0.7660294

No existen diferencias significativas entre ambos algoritmos, solo un 23.4% de confianza de que sean distintos

3.4.2 Comparativa general entre distintos algoritmos

Usaremos Friedman y como post-hoc Holm (los rankings se calculan por posiciones de los algoritmos para cada problema y no hace falta normalización)

```
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
```

```
Friedman rank sum test

data: as.matrix(tablatst)
Friedman chi-squared = 8.4444, df = 2, p-value = 0.01467
```

Existen diferencias significativas al menos entre un par de algoritmos

```
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)
```

```
Pairwise comparisons using Wilcoxon signed rank exact test

data: as.matrix(tablatst) and groups

 1     2
2 0.580 -
3 0.081 0.108

P value adjustment method: holm
```

Existen diferencias significativas a favor de M5' (3vs1 0.081 y 3vs2 0.108, con aprox. 90% de confianza) mientras que los otros dos pueden ser considerados equivalentes

4 Dataset VOWEL

5 Análisis exploratorio de datos

5.1 Definición del problema

Lo primero que vamos a hacer es buscar información sobre el dataset y el problemas que debemos resolver sobre este.

Buscando información en la web, se ha encontrado en la página <https://sci2s.ugr.es/keel/dataset.php?cod=113> una descripción sobre el problema:

- **Descripción del dataset:** Esta base de datos contiene información sobre el reconocimiento independiente del hablante de las once vocales en estado estacionario del inglés británico utilizando un conjunto de entrenamiento específico de relaciones de área logarítmica derivadas de lpc.
- **Objetivo:** predecir que vocal es cada instancia del dataset.

También se ha encontrado en la página <https://www.openml.org/search?type=data&sort=runs&id=58&status=active> una descripción más detallada de lo que son cada una de las variables del dataset.

En base al objetivo descrito sabemos que nos encontramos ante un *Problema de Clasificación*.

Leyendo el fichero del dataset podemos obtener un listado con las variables que lo componen, estas son:

Regresores / variables independientes

- TT (categórica): Dice si dicha instancia pertenece al conjunto de entrenamiento o test
- SpeakerNumber (categórica): Identificador del hablante.
- Sex (categórica): Sexo del hablante
- F0 - F9 (real): Son distintas características que identifican la vocal que ha pronunciado el hablante.
- Class (categórica): como variable objetivo, identifica la vocal que es cada instancia.

5.1.1 Planteamiento de hipótesis

Con toda esta información ya estamos en disposición de plantear ciertas hipótesis sobre el problema:

- ¿El sexo influyen en las distribuciones de las variables numéricas?
- ¿La clase objetivo influye en las distribuciones de las variables numéricas?
- ¿Y la clase y el sexo juntos?

5.2 Preparación de los datos

En este apartado nuestro objetivo es obtener una visión clara de la estructura del dataset, para así poder limpiarlo y prepararlo para posteriores análisis de tipo gráfico o a través de estadística descriptiva.

5.2.1 Estructura del dataset

A continuación vamos a leer el conjunto de datos para hacernos una idea sobre la estructura del dataset:

```
library(tidyverse)

vowel <- read.csv("vowel/vowel.dat", comment.char="@", header = FALSE)
names(vowel) <- c("tt", "speaker_number", "sex",
                   "f0", "f1", "f2", "f3",
                   "f4", "f5", "f6", "f7", "f8", "f9", "class")

str(vowel)

'data.frame': 990 obs. of 14 variables:
 $ tt          : int 0 0 0 0 0 0 0 0 0 ...
 $ speaker_number: int 0 0 0 0 0 0 0 0 0 ...
 $ sex         : int 0 0 0 0 0 0 0 0 0 ...
 $ f0          : num -3.64 -3.33 -2.12 -2.29 -2.6 ...
 $ f1          : num 0.418 0.496 0.894 1.809 1.938 ...
 $ f2          : num -0.67 -0.694 -1.576 -1.498 -0.846 ...
 $ f3          : num 1.779 1.365 0.147 1.012 1.062 ...
 $ f4          : num -0.168 -0.265 -0.707 -1.053 -1.633 ...
 $ f5          : num 1.627 1.933 1.559 1.06 0.764 ...
 $ f6          : num -0.388 -0.363 -0.579 -0.567 0.394 0.217 0.322 -0.435 -0.512 -0.466 ...
 $ f7          : num 0.529 0.51 0.676 0.235 -0.15 -0.246 0.45 0.992 0.928 0.702 ...
 $ f8          : num -0.874 -0.621 -0.809 -0.091 0.277 0.238 0.377 0.575 -0.167 0.06 ...
 $ f9          : num -0.814 -0.488 -0.049 -0.795 -0.396 -0.365 -0.366 -0.301 -0.434 -0.830 ...
 $ class       : int 0 1 2 3 4 5 6 7 8 9 ...
```

A partir de la función `str` podemos obtener la siguiente información:

- Efectivamente tenemos las variables descritas en el propio fichero del conjunto de datos.
- Número de instancias: 990.

- Número de variables: 14.
- Tipado de las variables: el tipado no es el correcto, pues las variables categóricas estan como numéricas, lo cambiamos:

```
vowel$tt <- factor(vowel$tt, c(0,1), c("train", "test"))
vowel$speaker_number <- factor(vowel$speaker_number, 0:14, as.character(0:14))
vowel$sex <- factor(vowel$sex, c(0,1), c("male", "female"))
vowel$class <- factor(vowel$class, 0:10, as.character(0:10))
```

5.2.2 Eliminación de variables

Revisando cada una de las variables que compone el dataset para ver si existen algunas que no aporten información útil, concluimos que:

- Eliminamos la variable TT: esta da información de si esa instancia pertenece al conjunto de test o entrenamiento, por tanto no aporta ninguna información útil, más bien puede introducir información que confunda al modelo. Además a la hora de aplicar los modelos de clasificación crearemos nuestros propios conjuntos de entrenamiento y test.
- Eliminamos la variable speaker_number: no queremos que el modelo aprenda a distinguir vocales en base al hablante. Lo que queremos es distinguir las vocales en base a las características del sonido que realiza una persona cualquiera al la hora de pronunciar una vocal.

Las demás variables nos aportan información útil de cara a nuestro objetivo.

Dicho esto eliminamos las variables que no queremos:

```
vowel <- vowel[,-(1:2)]  
  
str(vowel)  
  
'data.frame': 990 obs. of 12 variables:  
 $ sex : Factor w/ 2 levels "male","female": 1 1 1 1 1 1 1 1 1 1 ...  
 $ f0  : num -3.64 -3.33 -2.12 -2.29 -2.6 ...  
 $ f1  : num 0.418 0.496 0.894 1.809 1.938 ...  
 $ f2  : num -0.67 -0.694 -1.576 -1.498 -0.846 ...  
 $ f3  : num 1.779 1.365 0.147 1.012 1.062 ...  
 $ f4  : num -0.168 -0.265 -0.707 -1.053 -1.633 ...  
 $ f5  : num 1.627 1.933 1.559 1.06 0.764 ...  
 $ f6  : num -0.388 -0.363 -0.579 -0.567 0.394 0.217 0.322 -0.435 -0.512 -0.466 ...  
 $ f7  : num 0.529 0.51 0.676 0.235 -0.15 -0.246 0.45 0.992 0.928 0.702 ...
```

```
$ f8    : num  -0.874 -0.621 -0.809 -0.091 0.277 0.238 0.377 0.575 -0.167 0.06 ...
$ f9    : num  -0.814 -0.488 -0.049 -0.795 -0.396 -0.365 -0.366 -0.301 -0.434 -0.836 ...
$ class: Factor w/ 11 levels "0","1","2","3",...: 1 2 3 4 5 6 7 8 9 10 ...
```

5.2.3 Instancias duplicadas

Comprobemos si hay instancias duplicadas en el dataset:

```
vowel[duplicated(vowel), ]
```

```
[1] sex   f0    f1    f2    f3    f4    f5    f6    f7    f8    f9    class
<0 rows> (or 0-length row.names)
```

Como podemos observar, no tenemos instancias duplicadas.

5.2.4 Valores faltantes

Comprobemos si existen valores faltantes en el conjunto de datos:

```
vowel %>% filter(rowSums(across(.cols = everything(), .fns = is.na)) > 0)
```

```
[1] sex   f0    f1    f2    f3    f4    f5    f6    f7    f8    f9    class
<0 rows> (or 0-length row.names)
```

Como podemos observar, no tenemos ningún valor faltante.

5.3 Estadística descriptiva:

En este apartado nuestro objetivo es describir el conjunto de datos desde el punto de vista de la estadística descriptiva.

5.3.1 Análisis de tendencia central (variables numéricas)

Comenzaremos nuestro análisis usando medidas de tendencia central, estas son la media y la mediana

```

means <- apply(vowel[,2:11], 2, mean)
medians <- apply(vowel[,2:11], 2, median)

central_tendencies <- data.frame(Mean = means,
                                   Median = medians)
central_tendencies

```

	Mean	Median
f0	-3.203740404	-3.1455
f1	1.881763636	1.8765
f2	-0.507769697	-0.5725
f3	0.515482828	0.4335
f4	-0.305657576	-0.2990
f5	0.630244444	0.5520
f6	-0.004364646	0.0220
f7	0.336552525	0.3280
f8	-0.302975758	-0.3025
f9	-0.071339394	-0.1565

En general, podemos ver que para toda variable del conjunto de datos, el valor de la media no difiere mucho de el de la mediana. Este es un indicativo de que la distribución que siguen las variables son bastante simétricas, además de que no haya una alta presencia de valores anómalos. Como sabemos si hubiese ambas cosas resultaría en que los valores de la media diferiría mucho de la mediana, al ser la media muy sensible tanto a distribuciones asimétricas como valores anómalos, en contraposición a la mediana que es robusta a esto.

5.3.2 Análisis de dispersión (variables numéricas)

A continuación, vamos a realizar un estudio de la variabilidad de los datos.

Mínimo, Máximo y Rango:

```

minimums <- apply(vowel[,2:11], 2, min)
maximums <- apply(vowel[,2:11], 2, max)

data.frame(Minimum = minimums,
           Maximum = maximums,
           Range = (maximums - minimums))

```

Minimum Maximum Range

```

f0 -5.211 -0.941 4.270
f1 -1.274  5.074 6.348
f2 -2.487  1.431 3.918
f3 -1.409  2.377 3.786
f4 -2.127  1.831 3.958
f5 -0.836  2.327 3.163
f6 -1.537  1.403 2.940
f7 -1.293  2.039 3.332
f8 -1.613  1.309 2.922
f9 -1.680  1.396 3.076

```

En base a los resultados obtenidos podemos ver que prácticamente todas las variables están en un rango de valores similares, esto puede debido a la transformación logarítmica que decían que se había aplicado en la descripción del dataset.

Desviación estándar, varianza y desviación absoluta media:

```

standard_deviations <- apply(vowel[,2:11], 2, sd)
variances <- apply(vowel[,2:11], 2, var)
median_absolute_deviations <- apply(vowel[,2:11], 2, mad)

data.frame(Standard_deviation = standard_deviations,
           Variance = variances,
           Median_absolute_deviation = median_absolute_deviations)

```

	Standard_deviation	Variance	Median_absolute_deviation
f0	0.8689872	0.7551388	0.9377445
f1	1.1752720	1.3812643	1.2505731
f2	0.7119483	0.5068703	0.6582744
f3	0.7592613	0.5764778	0.8287734
f4	0.6646023	0.4416962	0.6968220
f5	0.6038711	0.3646603	0.6063834
f6	0.4619268	0.2133764	0.4418148
f7	0.5733020	0.3286752	0.6412245
f8	0.5701616	0.3250843	0.5952639
f9	0.6039855	0.3647985	0.6530853

Podemos observar que todas las variables tienen una variabilidad similar.

Teniendo en cuenta los resultados obtenidos para las distintas medidas de dispersión y los rangos de valores que tiene cada una de las variables podemos decir que en general, las medidas de tendencia central son representativas de la variable a la que corresponden.

Cuantiles:

```
Q1s <- apply(vowel[,2:11], 2, quantile, probs = c(0.25))
Q3s <- apply(vowel[,2:11], 2, quantile, probs = c(0.75))

data.frame(Q1 = Q1s,
           Q2 = Q3s,
           IQR = (Q3s - Q1s))
```

	Q1	Q2	IQR
f0	-3.88775	-2.60250	1.28525
f1	1.05150	2.73800	1.68650
f2	-0.97575	-0.06875	0.90700
f3	-0.06550	1.09600	1.16150
f4	-0.76900	0.16950	0.93850
f5	0.19600	1.02850	0.83250
f6	-0.30700	0.29650	0.60350
f7	-0.09575	0.77000	0.86575
f8	-0.70400	0.09375	0.79775
f9	-0.54800	0.37100	0.91900

Observamos que el rango intercuartílico no es muy grande teniendo en cuenta el rango de las variables, esto es un indicio de que los valores están agrupados en la parte central de la distribución.

5.3.3 Análisis de Skewness y Kurtosis

Vamos a realizar una serie de test estadísticos para analizar tanto la simetría como las kurtosis de la distribución de las distintas variables.

Skewness:

Para analizar la simetría de la distribución vamos a usar el teste de Agostino.

```
library(moments)

agostino.test(vowel$f0)

D'Agostino skewness test

data: vowel$f0
```

```
skew = 0.066297, z = 0.857022, p-value = 0.3914
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f1)
```

D'Agostino skewness test

```
data: vowel$f1
skew = -0.042698, z = -0.552300, p-value = 0.5807
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f2)
```

D'Agostino skewness test

```
data: vowel$f2
skew = 0.23522, z = 3.00417, p-value = 0.002663
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f3)
```

D'Agostino skewness test

```
data: vowel$f3
skew = 0.12874, z = 1.65937, p-value = 0.09704
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f4)
```

D'Agostino skewness test

```
data: vowel$f4
skew = 0.016474, z = 0.213175, p-value = 0.8312
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f5)
```

D'Agostino skewness test

```
data: vowel$f5
skew = 0.35652, z = 4.48115, p-value = 7.424e-06
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f6)
```

D'Agostino skewness test

```
data: vowel$f6
skew = -0.20584, z = -2.63689, p-value = 0.008367
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f7)
```

D'Agostino skewness test

```
data: vowel$f7
skew = 0.0059484, z = 0.0769764, p-value = 0.9386
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f8)
```

D'Agostino skewness test

```
data: vowel$f8
skew = 0.053788, z = 0.695571, p-value = 0.4867
alternative hypothesis: data have a skewness
```

```
agostino.test(vowel$f9)
```

D'Agostino skewness test

```
data: vowel$f9
skew = 0.29532, z = 3.74459, p-value = 0.0001807
alternative hypothesis: data have a skewness
```

Observaciones:

- Variables que muestran evidencia significativa de la presencia de sesgo: f2, f5, f6 y f9.
- Variables que no muestran evidencia significativa de la presencia de sesgo: f0, f1, f2, f3, f4, f7 y f8

Kurtosis:

Para analizar la kurtosis vamos a usar el test de Anscombe.

```
anscombe.test(vowel$f0)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f0
kurt = 2.5025, z = -4.2172, p-value = 2.474e-05
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f1)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f1
kurt = 2.6007, z = -3.1326, p-value = 0.001733
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f2)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f2
kurt = 2.84244, z = -0.99403, p-value = 0.3202
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f3)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f3
kurt = 2.2356, z = -8.2413, p-value = 2.22e-16
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f4)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f4
kurt = 2.7227, z = -1.9746, p-value = 0.04832
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f5)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f5
kurt = 2.7090, z = -2.0955, p-value = 0.03613
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f6)
```

Anscombe-Glynn kurtosis test

```
data: vowel$f6
kurt = 3.14561, z = 0.99741, p-value = 0.3186
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f7)
```

```
Anscombe-Glynn kurtosis test
```

```
data: vowel$f7
kurt = 2.5070, z = -4.1647, p-value = 3.118e-05
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f8)
```

```
Anscombe-Glynn kurtosis test
```

```
data: vowel$f8
kurt = 2.5392, z = -3.7935, p-value = 0.0001485
alternative hypothesis: kurtosis is not equal to 3
```

```
anscombe.test(vowel$f9)
```

```
Anscombe-Glynn kurtosis test
```

```
data: vowel$f9
kurt = 2.2400, z = -8.1566, p-value = 4.441e-16
alternative hypothesis: kurtosis is not equal to 3
```

Observaciones:

- Variables que muestran evidencia significativa de tener kurtosis diferente de 3: f0, f1, f3, f4, f5, f7, f8 y f9.
- Variables que no muestran evidencia significativa de tener kurtosis diferente de 3: f2 y f6

5.3.4 Normalidad de los datos

Si bien con el test D'Agostino y el test de Anscombe-Glynn podemos hacernos una idea de qué variables siguen aproximadamente una distribución normal (distribución simétrica y kurtosis de 3 son características de distribuciones normales), vamos a utilizar otros mecanismos para cerciorarnos de estos, en concreto realizaremos el test de Shapiro-Wilk's y un gráfico QQ-plot.

Test de Shapiro-Wilk's

```
shapiro.test(vowel$f0)
```

```
Shapiro-Wilk normality test

data: vowel$f0
W = 0.9928, p-value = 9.807e-05

shapiro.test(vowel$f1)

Shapiro-Wilk normality test

data: vowel$f1
W = 0.9966, p-value = 0.0312

shapiro.test(vowel$f2)

Shapiro-Wilk normality test

data: vowel$f2
W = 0.99217, p-value = 4.245e-05

shapiro.test(vowel$f3)

Shapiro-Wilk normality test

data: vowel$f3
W = 0.98364, p-value = 4.324e-09

shapiro.test(vowel$f4)

Shapiro-Wilk normality test

data: vowel$f4
W = 0.9971, p-value = 0.07073

shapiro.test(vowel$f5)
```

```
Shapiro-Wilk normality test

data: vowel$f5
W = 0.98632, p-value = 5.435e-08

shapiro.test(vowel$f6)

Shapiro-Wilk normality test

data: vowel$f6
W = 0.9964, p-value = 0.0225

shapiro.test(vowel$f7)

Shapiro-Wilk normality test

data: vowel$f7
W = 0.99397, p-value = 0.0005086

shapiro.test(vowel$f8)

Shapiro-Wilk normality test

data: vowel$f8
W = 0.99467, p-value = 0.001437

shapiro.test(vowel$f9)

Shapiro-Wilk normality test

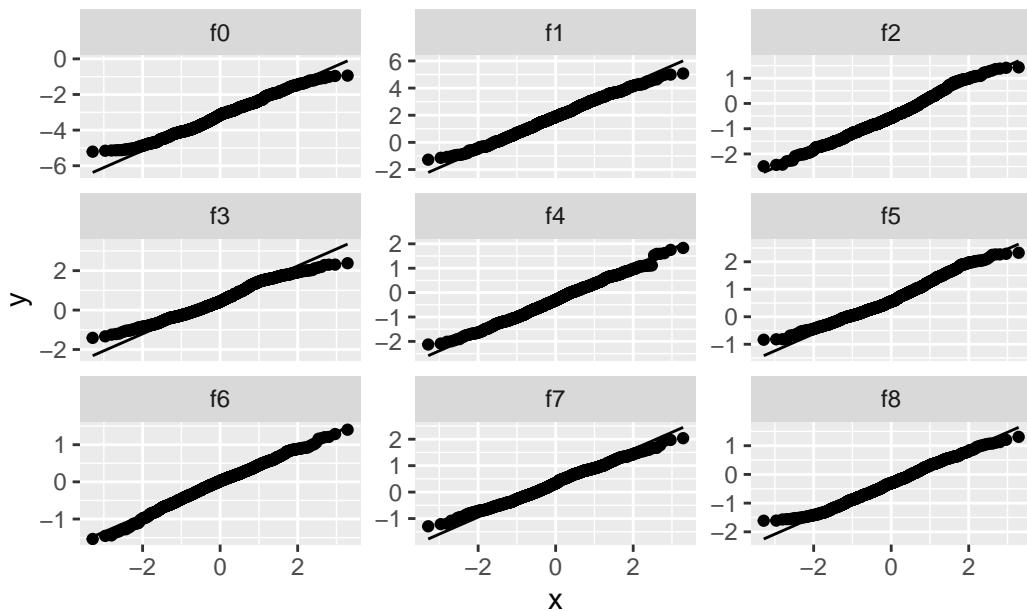
data: vowel$f9
W = 0.97377, p-value = 2.208e-12
```

Salvo f4, para las demás variables el p-valor es extremadamente pequeño, lo que sugiere que hay evidencia significativa para rechazar la hipótesis nula de que los datos provienen de una distribución normal.

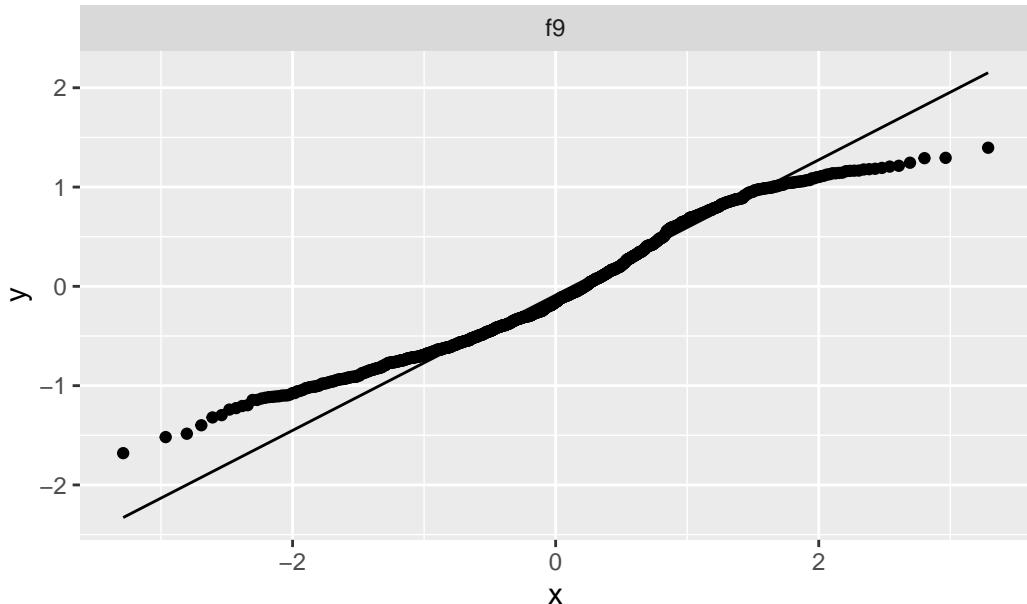
QQ-plot

```
library(DataExplorer)
```

```
plot_qq(vowel)
```



Page 1



Page 2

A partir del gráfico QQ-plot podemos ver como todas las variables siguen casi completamente la línea que marca la distribución normal, por lo que a diferencia de lo que nos decía el test de Shapiro-Wilk si que parece que las distribuciones de dichas variables se asemejan en gran manera a una distribución normal, esto lo verificaremos con los histogramas.

5.3.5 Tablas de contingencia (variables categóricas)

Vamos a mostrar las tablas de contingencia para las variables categóricas, con esto podemos ver si existe desequilibrio entre las clases:

```
apply(vowel[,-(2:11)], 2, table)
```

```
$sex
```

female	male
462	528

```
$class
```

0	1	10	2	3	4	5	6	7	8	9
90	90	90	90	90	90	90	90	90	90	90

Observaciones:

- variable sex: existen mas instancias de la clase male que female, pero ese desequilibrio no es muy grave de cara a la aplicación de los modelos de clasificación.
- variable objetivo: tenemos igual número de instancias para cada clase, lo cual es lo ideal para aplicar modelos de clasificación ya que no tenemos problemas de desequilibrio de clases.

5.4 Visualización de datos

Además de usar estadística descriptiva para mostrar los rasgos que presentan las variables del conjunto de datos, realizar distintos tipos de visualizaciones es interesante para no solo obtener información nueva, sino también contrastar la información que hemos obtenido mediante la estadística descriptiva.

Histogramas (variables numéricas)

En primer lugar vamos a pintar un histograma para cada variable numérica para hacernos una idea de la distribución que siguen las variables. Además, también pintaremos la media y la mediana para comprobar que ambas no estan muy alejadas entre sí.

```
library(ggplot2)

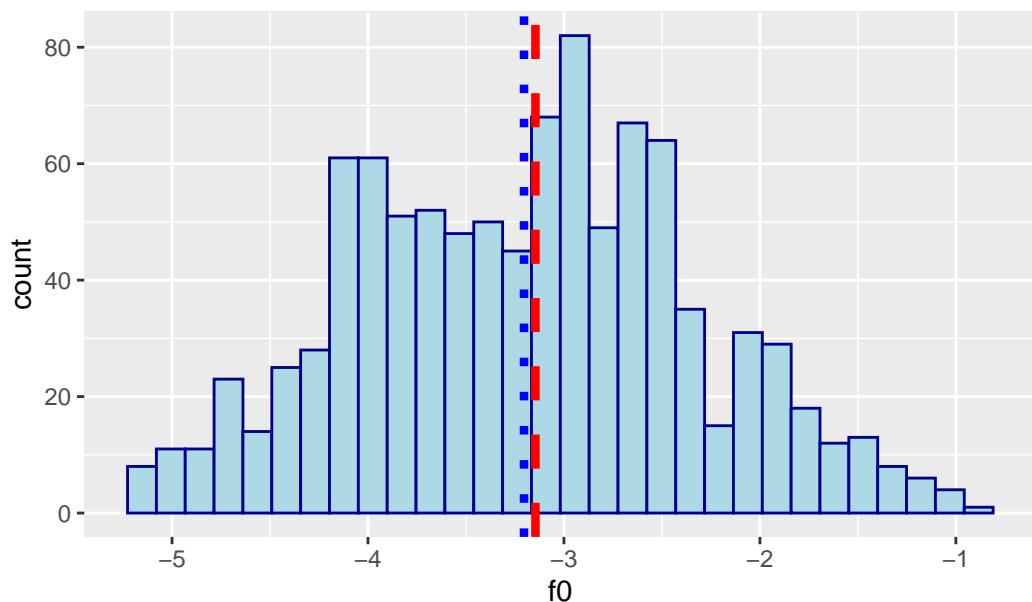
plot_hist <- function(data) {
  sapply(names(data), function(col_name) {

    p <- data %>% ggplot(aes_string(x = col_name)) +
      geom_histogram(color = "darkblue", fill = "lightblue") +
      labs(title = paste("Curva de densidad - ", col_name)) +
      geom_vline(xintercept = median(data[[col_name]]), color = "red",
                 linetype = "dashed", size = 1.5) +
      geom_vline(xintercept = mean(data[[col_name]]), color = "blue",
                 linetype = "dotted", size = 1.5)
    print(p)
  })
}

plot_hist(vowel[,2:11])

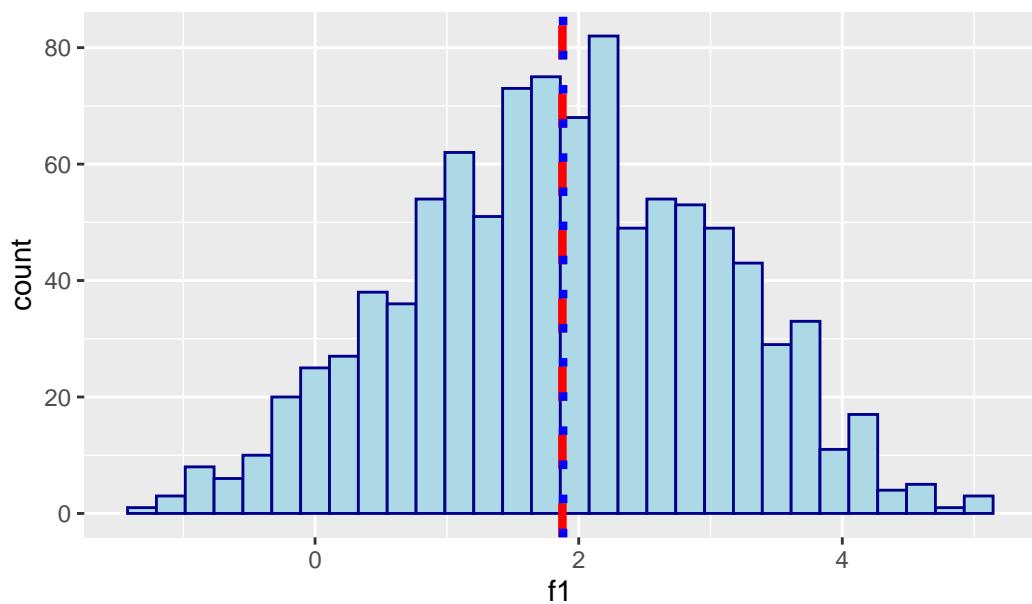
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f0



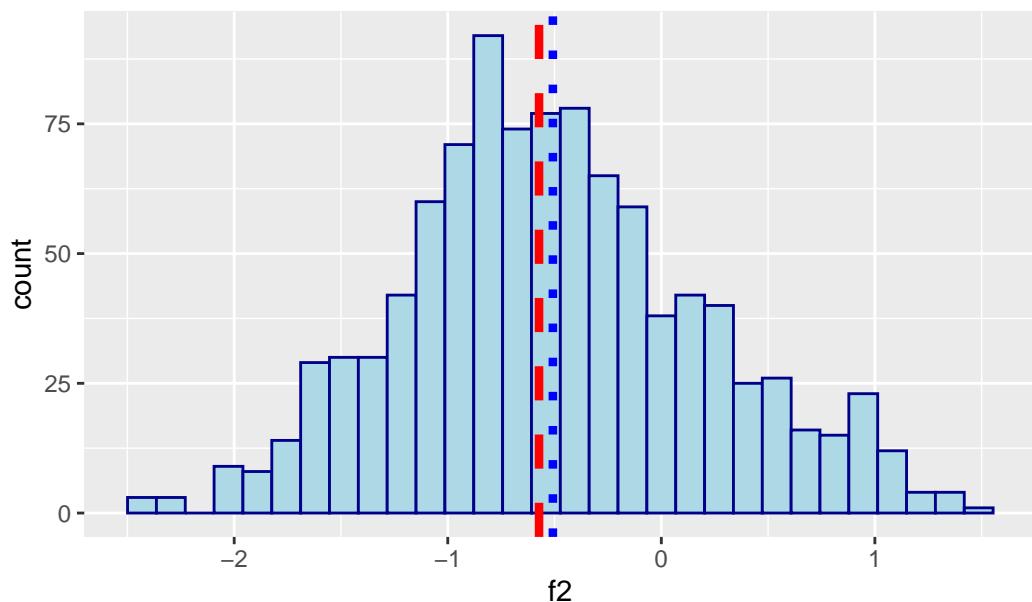
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f1



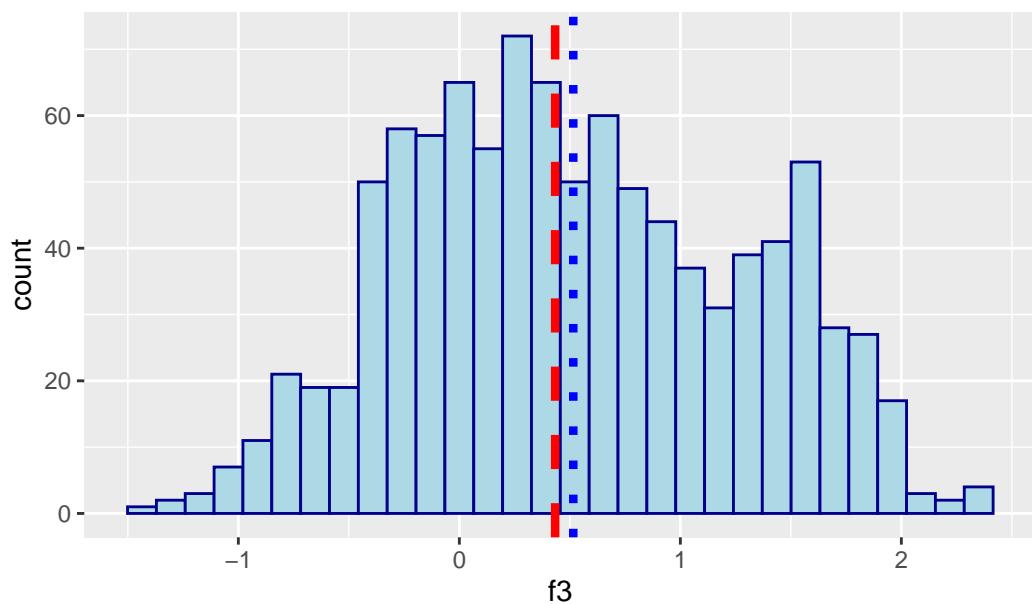
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f2



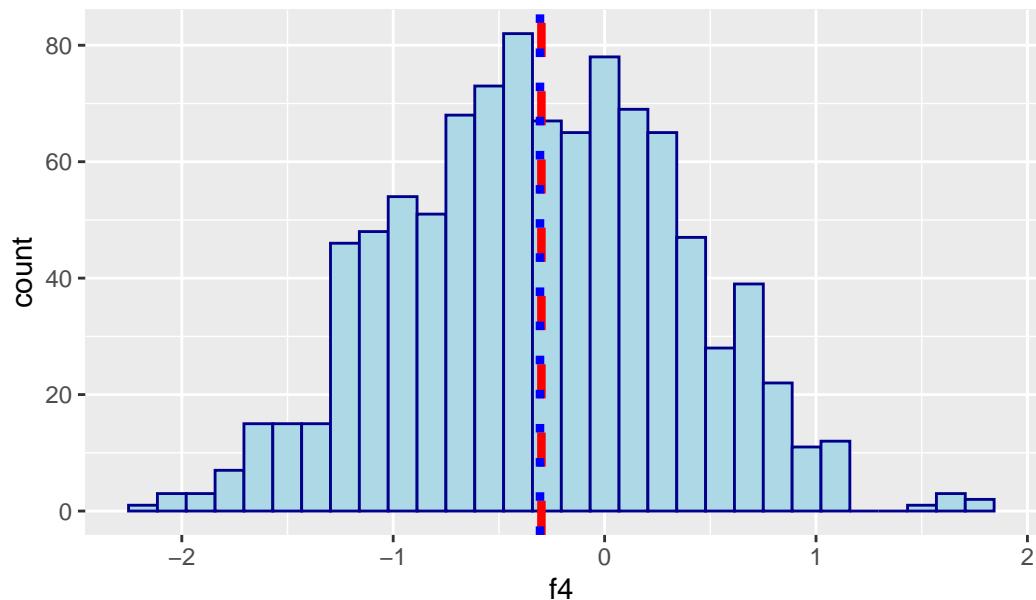
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – f3



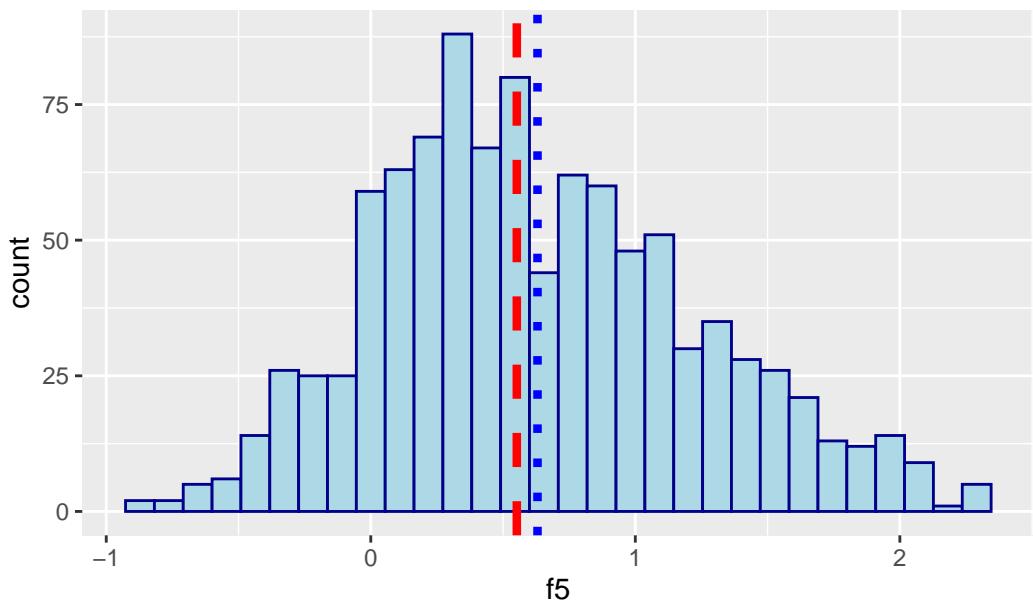
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Curva de densidad – f4



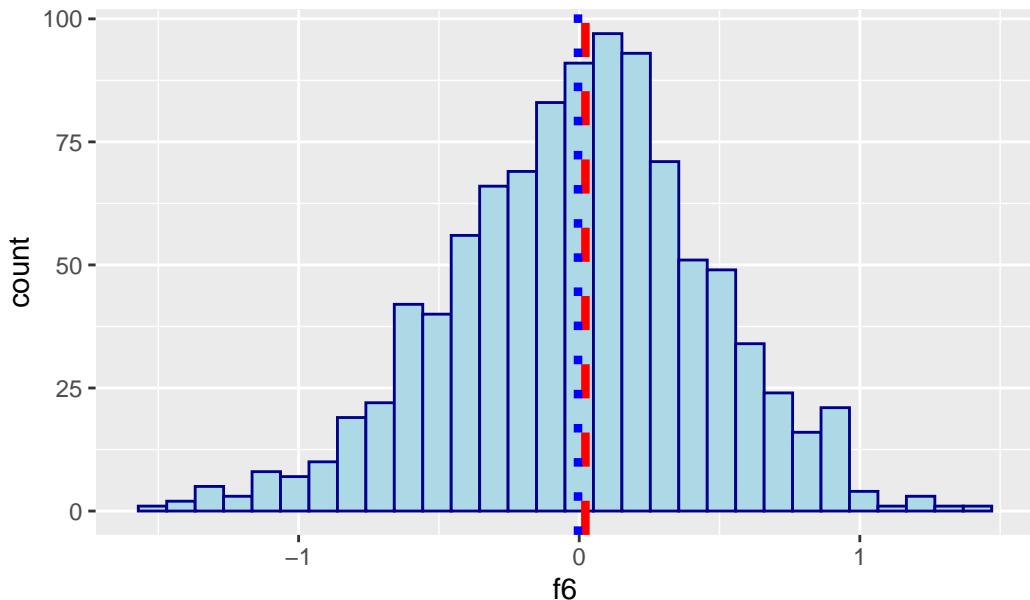
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f5



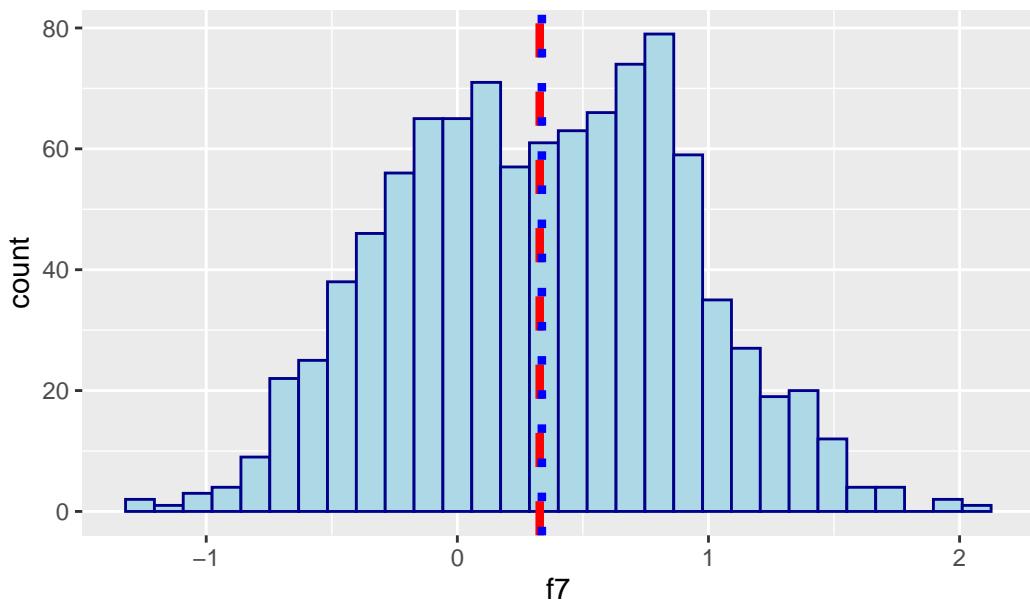
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f6



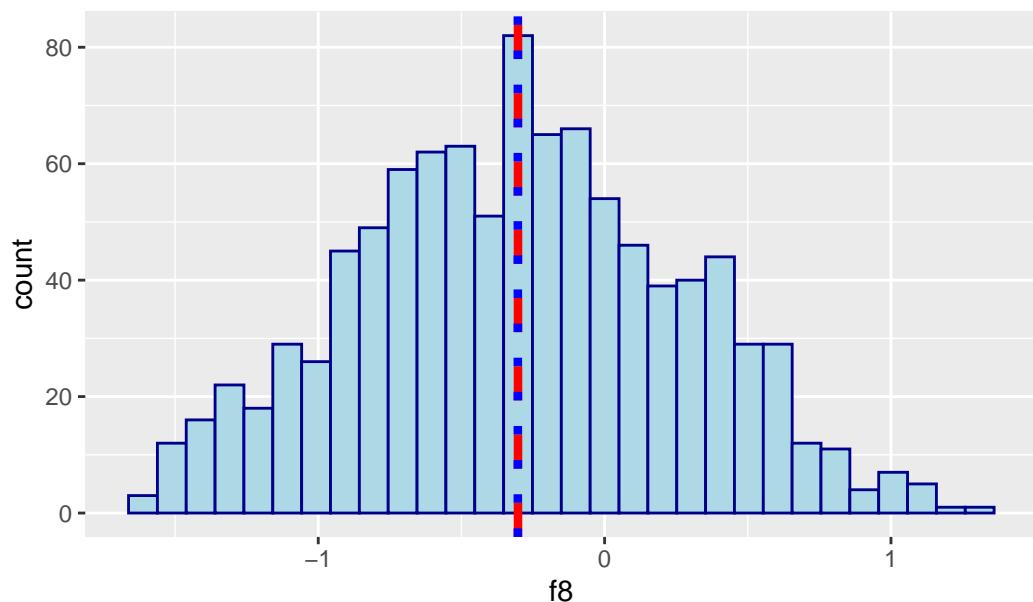
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f7



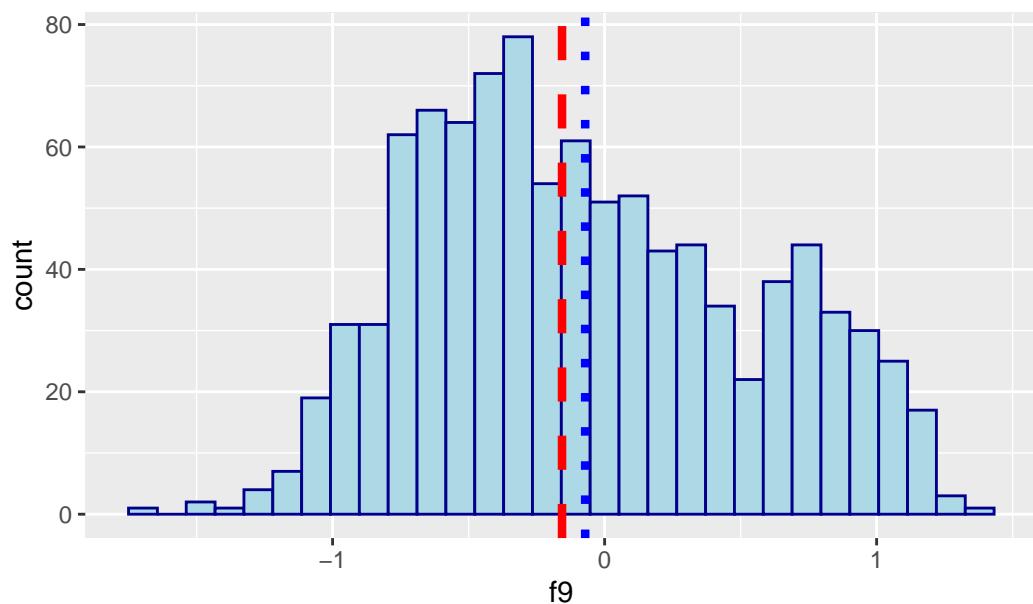
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f8



```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Curva de densidad – f9



	f0	f1	f2	f3
data	data.frame,10	data.frame,10	data.frame,10	data.frame,10
layers	list,3	list,3	list,3	list,3
scales	ScalesList,2	ScalesList,2	ScalesList,2	ScalesList,2
mapping	~f0	~f1	~f2	~f3
theme	list,0	list,0	list,0	list,0
coordinates	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5
facet	FacetNull,2	FacetNull,2	FacetNull,2	FacetNull,2
plot_env	?	?	?	?
labels	list,5	list,5	list,5	list,5
	f4	f5	f6	f7
data	data.frame,10	data.frame,10	data.frame,10	data.frame,10
layers	list,3	list,3	list,3	list,3
scales	ScalesList,2	ScalesList,2	ScalesList,2	ScalesList,2
mapping	~f4	~f5	~f6	~f7
theme	list,0	list,0	list,0	list,0
coordinates	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5
facet	FacetNull,2	FacetNull,2	FacetNull,2	FacetNull,2
plot_env	?	?	?	?
labels	list,5	list,5	list,5	list,5
	f8	f9		
data	data.frame,10	data.frame,10		
layers	list,3	list,3		
scales	ScalesList,2	ScalesList,2		
mapping	~f8	~f9		
theme	list,0	list,0		
coordinates	CoordCartesian,5	CoordCartesian,5		
facet	FacetNull,2	FacetNull,2		
plot_env	?	?		
labels	list,5	list,5		

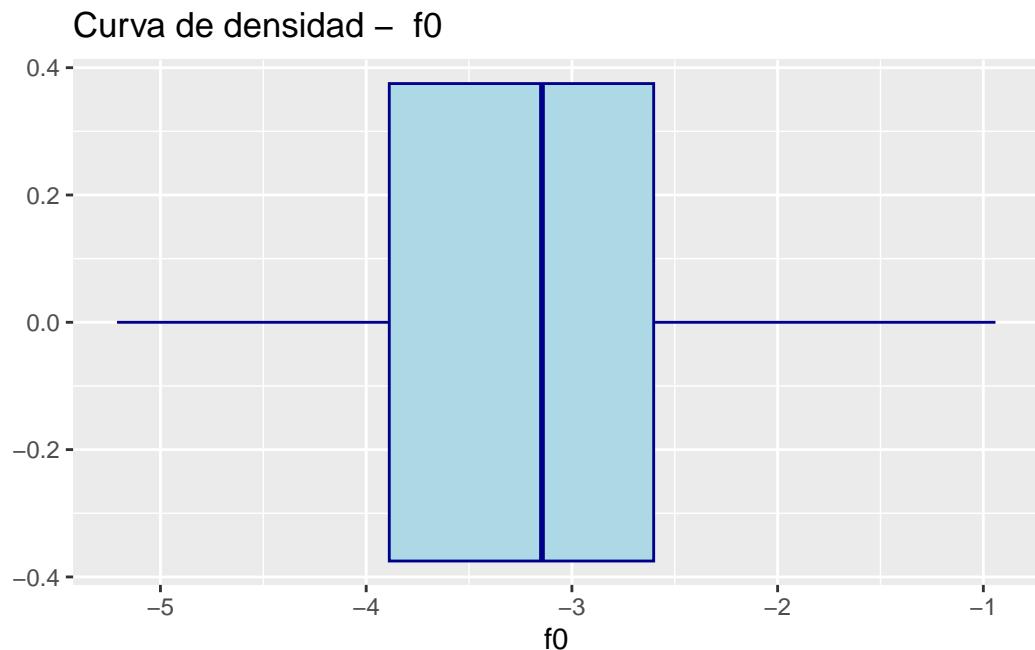
A partir de los histogramas podemos sacar las siguientes conclusiones:

- Tal y como nos indicaban las medidas de tendencia central y dispersión, la media y la mediana son bastante similares y además son representativas para cada una de las variables del dataset.
- Como veíamos a través del gráfico QQ-plot, todas las variables numéricas presentan una distribución parecida a la que tendría una distribución normal. Lo cual es muy importante pues gran parte de las asunciones estadísticas que siguen modelos de clasificación necesitan que la distribución de las variables sea normal.

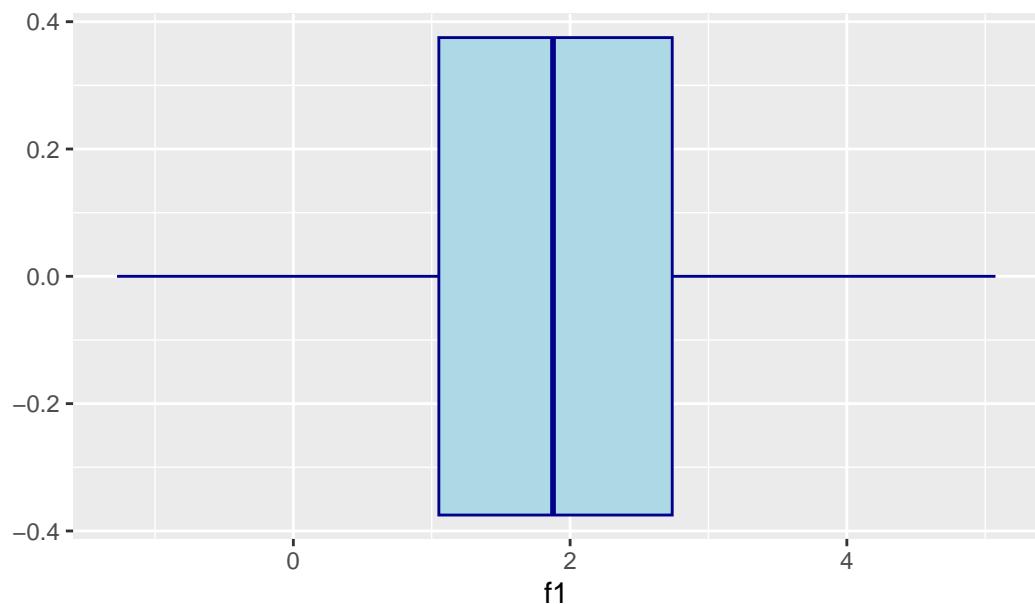
Gráfico de cajas

A continuación vamos a mostrar diagramas de caja para cada variable, principalmente con el objeto de detectar outliers en las variables

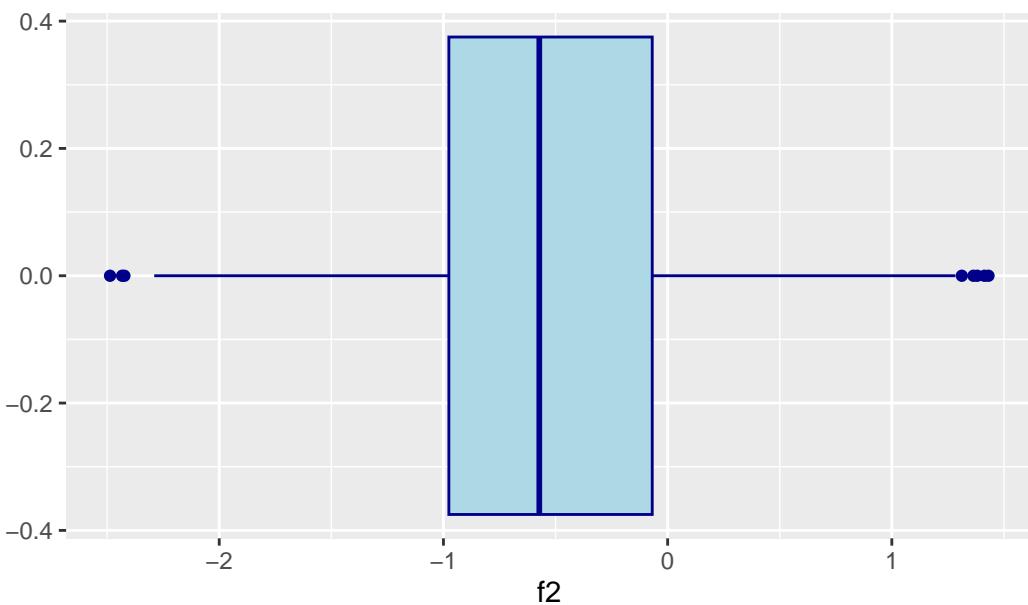
```
plot_boxplots <- function(data) {  
  sapply(names(data), function(col_name) {  
    p <- data %>% ggplot(aes_string(x = col_name)) +  
      geom_boxplot(color = "darkblue", fill = "lightblue") +  
      labs(title = paste("Curva de densidad - ", col_name))  
    print(p)  
  })  
}  
  
plot_boxplots(vowel[,2:11])
```



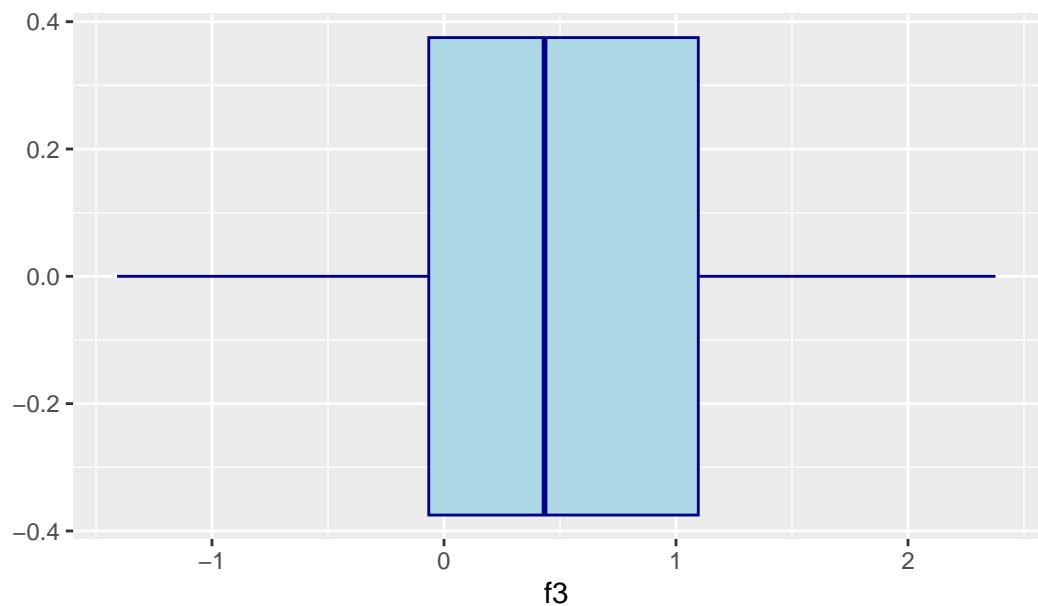
Curva de densidad – f1



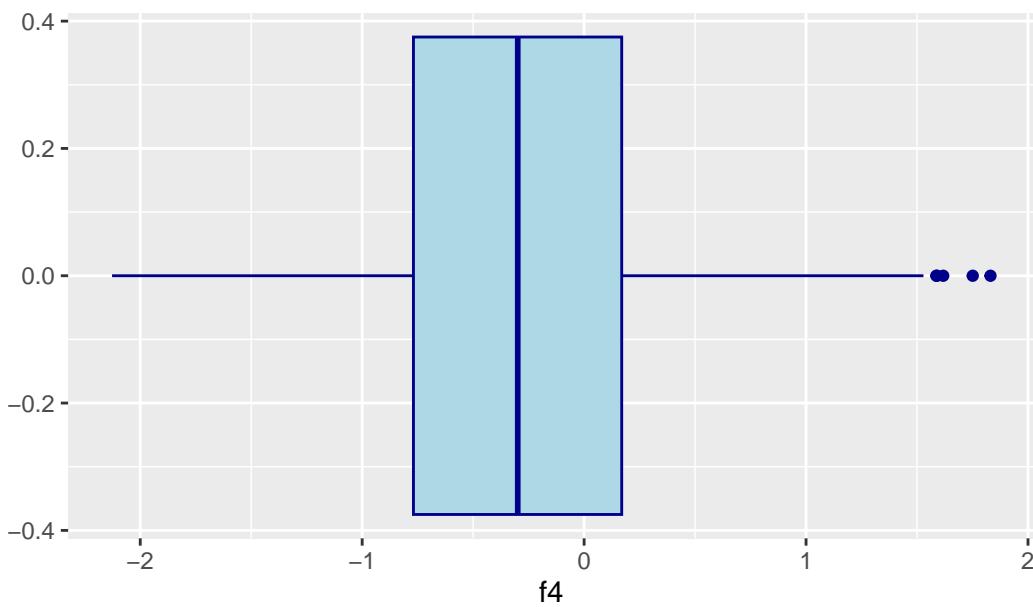
Curva de densidad – f2



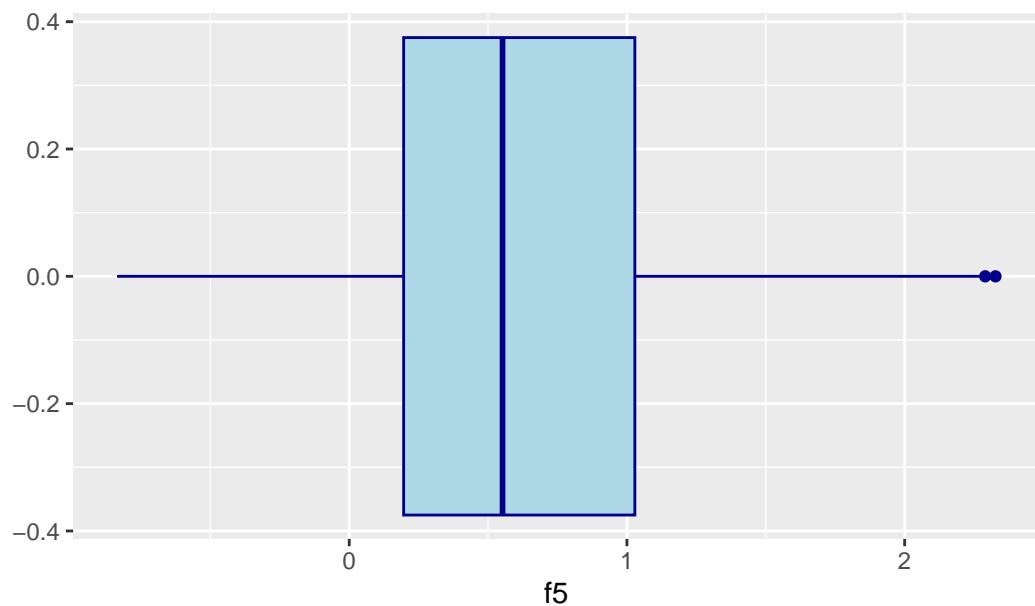
Curva de densidad – f3



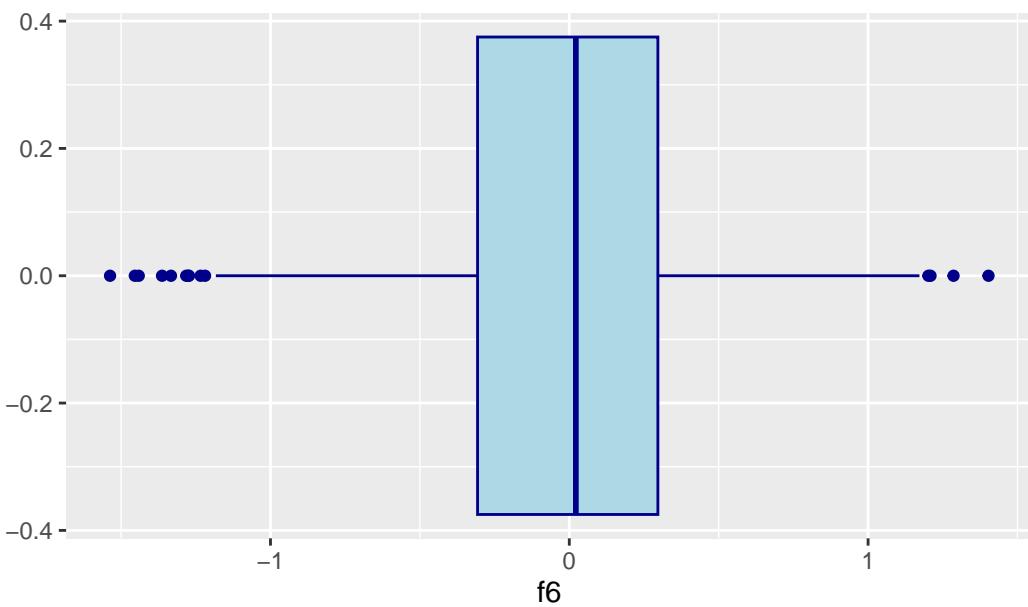
Curva de densidad – f4



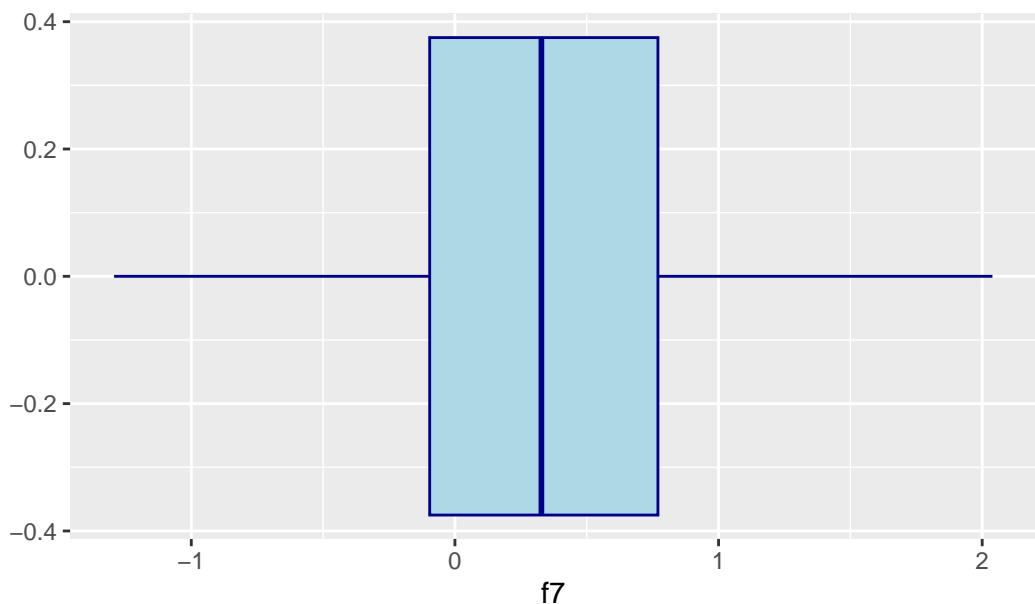
Curva de densidad – f5



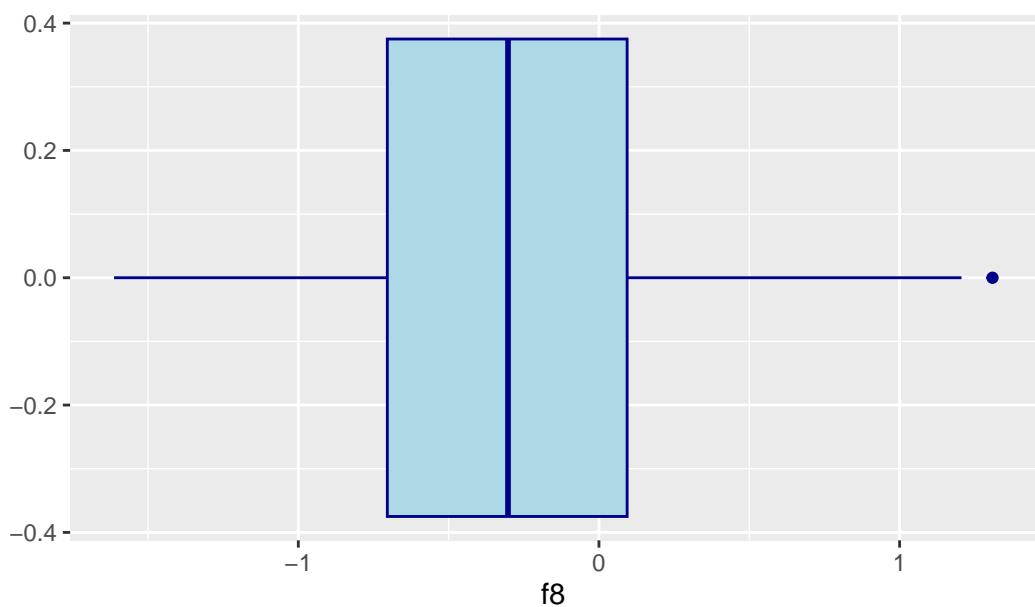
Curva de densidad – f6



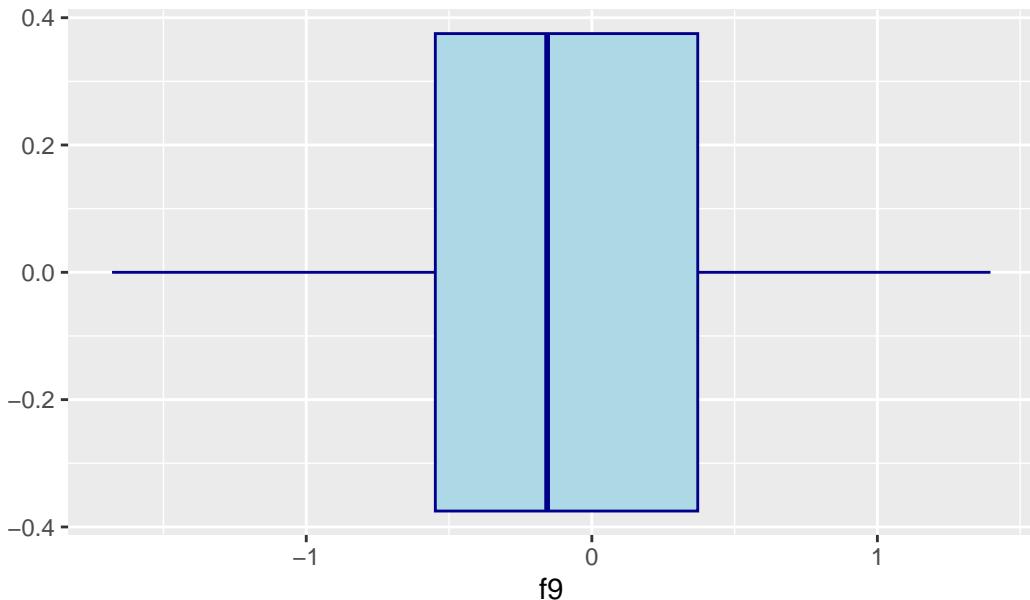
Curva de densidad – f7



Curva de densidad – f8



Curva de densidad – f9



	f0	f1	f2	f3
data	data.frame,10	data.frame,10	data.frame,10	data.frame,10
layers	list,1	list,1	list,1	list,1
scales	ScalesList,2	ScalesList,2	ScalesList,2	ScalesList,2
mapping	~f0	~f1	~f2	~f3
theme	list,0	list,0	list,0	list,0
coordinates	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5
facet	FacetNull,2	FacetNull,2	FacetNull,2	FacetNull,2
plot_env	?	?	?	?
labels	list,2	list,2	list,2	list,2
	f4	f5	f6	f7
data	data.frame,10	data.frame,10	data.frame,10	data.frame,10
layers	list,1	list,1	list,1	list,1
scales	ScalesList,2	ScalesList,2	ScalesList,2	ScalesList,2
mapping	~f4	~f5	~f6	~f7
theme	list,0	list,0	list,0	list,0
coordinates	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5	CoordCartesian,5
facet	FacetNull,2	FacetNull,2	FacetNull,2	FacetNull,2
plot_env	?	?	?	?
labels	list,2	list,2	list,2	list,2
	f8	f9		
data	data.frame,10	data.frame,10		
layers	list,1	list,1		

```

scales      ScalesList,2      ScalesList,2
mapping     ~f8                  ~f9
theme       list,0              list,0
coordinates CoordCartesian,5  CoordCartesian,5
facet        FacetNull,2      FacetNull,2
plot_env    ?                  ?
labels      list,2              list,2

```

Podemos ver como solo unas pocas variables presentan outliers, además en poca cantidad. Esto se corresponde cuando decíamos que la media y la mediana eran muy similares debido la simetría de las ditribuciones (que también se aprecia en estos gráficos) y la ausencia de valores atípicos.

Gráfico de barras (variables categóricas)

Para visualizar mejor las variables categóricas vamos a realizar un gráfico de barras

```

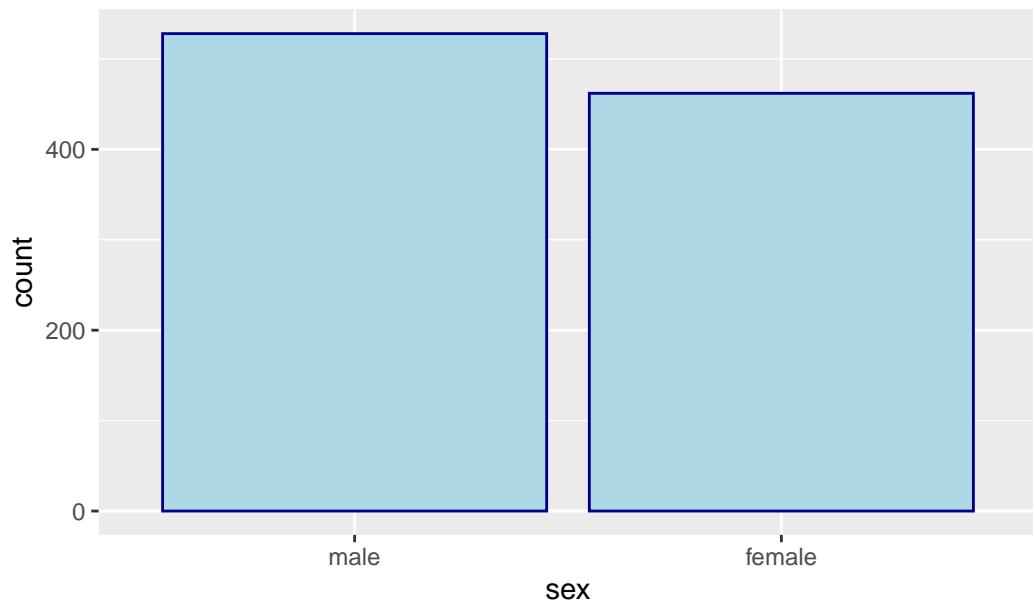
plot_bar <- function(data) {
  sapply(names(data), function(col_name) {

    p <- data %>% ggplot(aes_string(x = col_name)) +
      geom_bar(color = "darkblue", fill = "lightblue") +
      labs(title = paste("Curva de densidad - ", col_name))
    print(p)
  })
}

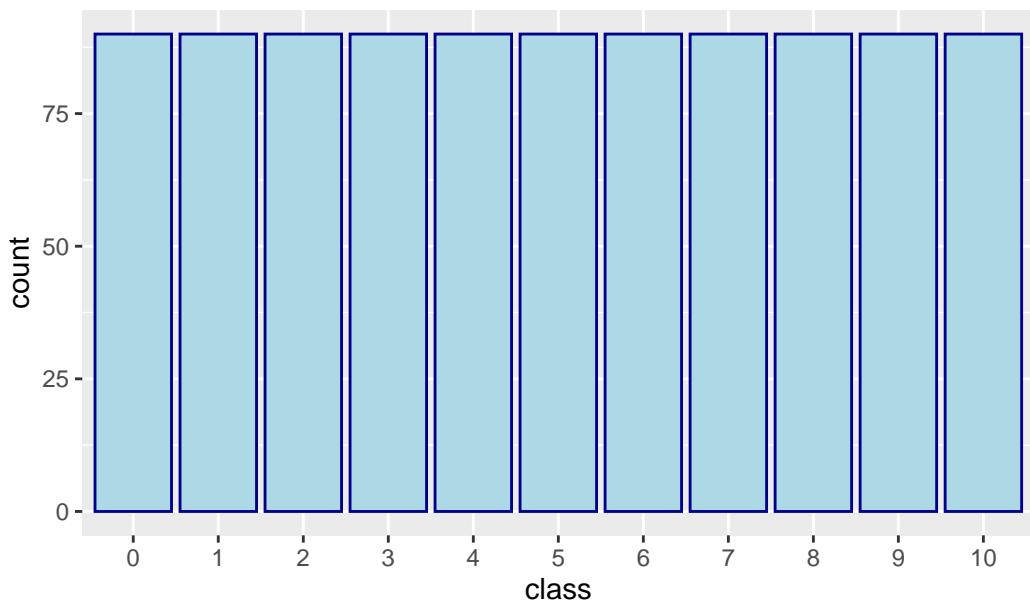
plot_bar(vowel[,-(2:11)])

```

Curva de densidad – sex



Curva de densidad – class



```
          sex           class  
data   data.frame,2   data.frame,2
```

```

layers      list,1          list,1
scales      ScalesList,2    ScalesList,2
mapping     ~sex            ~class
theme       list,0          list,0
coordinates CoordCartesian,5 CoordCartesian,5
facet       FacetNull,2    FacetNull,2
plot_env    ?               ?
labels      list,4          list,4

```

Con esto podemos ver de manera más visual lo que ya nos decía las tablas de contingencia.

5.4.1 Exploración de relaciones entre variables

5.4.1.1 Separabilidad de las variables por clases

Lo primero que vamos a comprobar es si existe cierta separabilidad en las clases de las instancias, esto es útil pues dependiendo de la separabilidad utilizaremos un determinado modelo para nuestro conjunto de datos

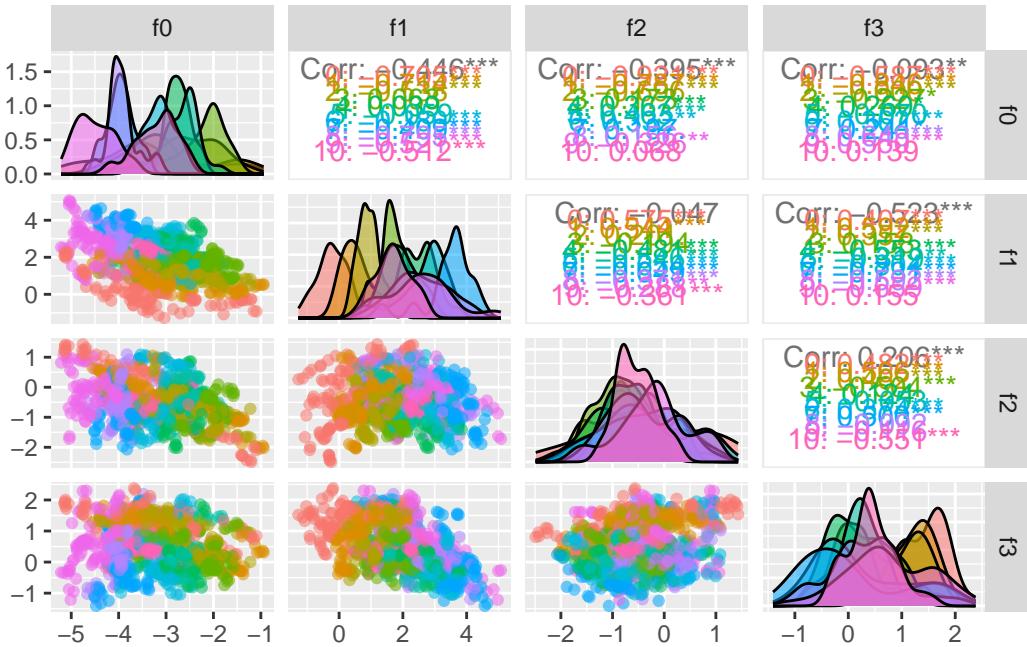
Nota: en el gráfico solo hemos mostrado las variables de f0 a f3 pues si ponemos todas no se puede apreciar, pero se ha mirado con todas.

```
library(GGally)
```

```
Warning: package 'GGally' was built under R version 4.3.2
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
p <- ggpairs(vowel, columns = 2:5, aes(color = class, alpha = 0.5))
print(p)
```



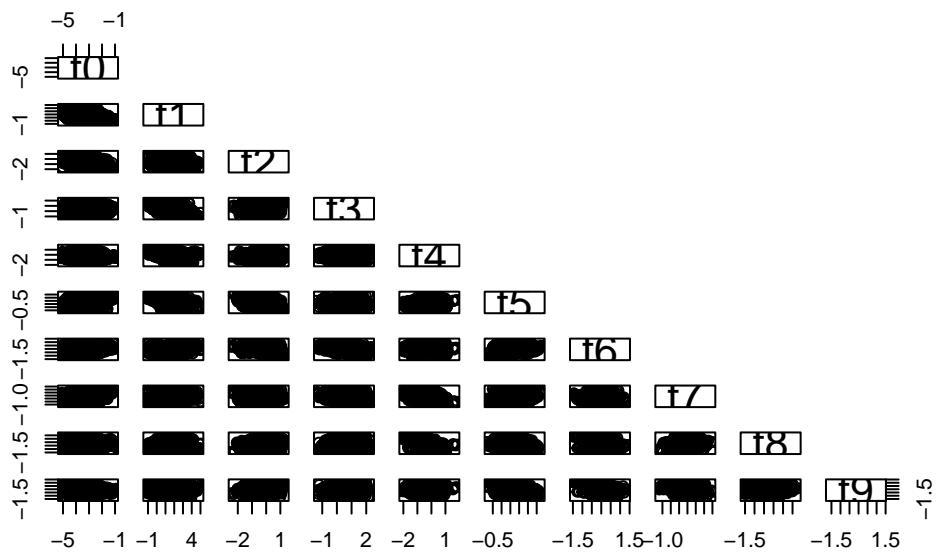
Como podemos observar, no existe una clara separabilidad entre las clases para las distintas variables, esto se tendrá en cuenta a la hora de elegir el modelo de clasificación.

5.4.1.2 Gráfico de puntos variable independiente vs independiente

En este caso vamos a graficar cada variable numérica independiente frente a las demás independiente, todo esto con el objeto de detectar correlaciones entre variables y así saber qué variables se podrían no usar en un modelo de clasificación ya que otra contiene la misma información. Mostraremos tanto una nube de puntos como un gráfico de correlaciones, pues este último es muy útil para ver la correlación entre dos variables.

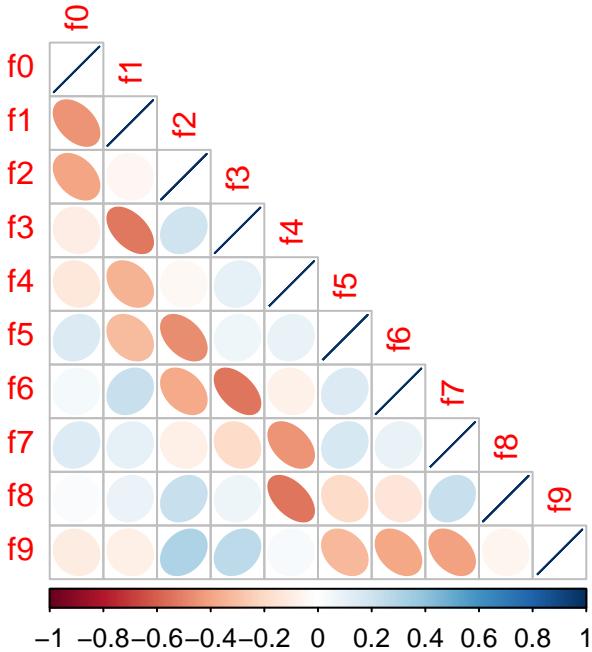
```
library(Sleuth2)

pairs(vowel[,2:11], upper.panel = NULL)
```



```
library(corrplot)

corrplot(cor(vowel[,2:11]), method = "ellipse", type = "lower")
```



Podemos observar que existe correlación entre varias variables, lo cual es un indicio que varias de las variables comparten información.

5.4.1.3 PCA.

Dado que existe una correlación generalizada en el conjunto de datos, y además con el objeto de saber cuáles son las características más importantes de f0 a f9, vamos a realizar PCA.

```
library("FactoMineR")
library("factoextra")
```

Welcome! Want to learn more? See two factoextra-related books at <https://goo.gl/ve3WBa>

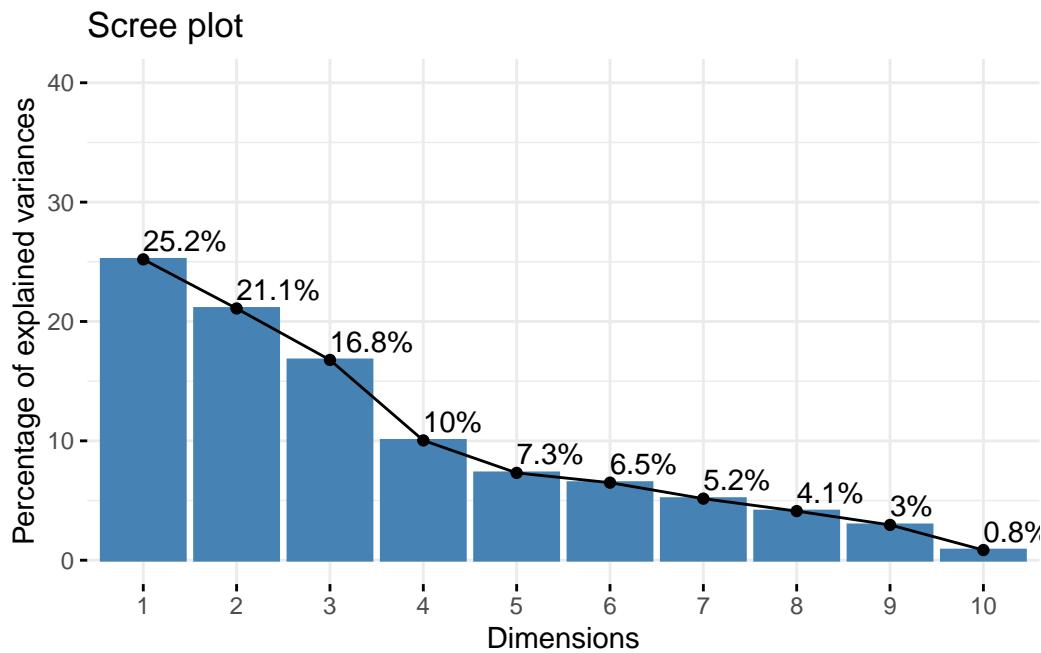
```
vowel_pca = PCA(vowel[,2:11], scale.unit = TRUE, ncp = 10, graph = FALSE)
```

```
get_eigenvalue(vowel_pca)
```

	eigenvalue	variance.percent	cumulative.variance.percent
Dim.1	2.52039555	25.2039555	25.20396
Dim.2	2.10900541	21.0900541	46.29401

Dim.3	1.67794284	16.7794284	63.07344
Dim.4	1.00355247	10.0355247	73.10896
Dim.5	0.73145507	7.3145507	80.42351
Dim.6	0.64976609	6.4976609	86.92117
Dim.7	0.51569467	5.1569467	92.07812
Dim.8	0.41154869	4.1154869	96.19361
Dim.9	0.29567396	2.9567396	99.15035
Dim.10	0.08496525	0.8496525	100.00000

```
fviz_eig(vowel_pca, addlabels = TRUE, ylim = c(0, 40))
```

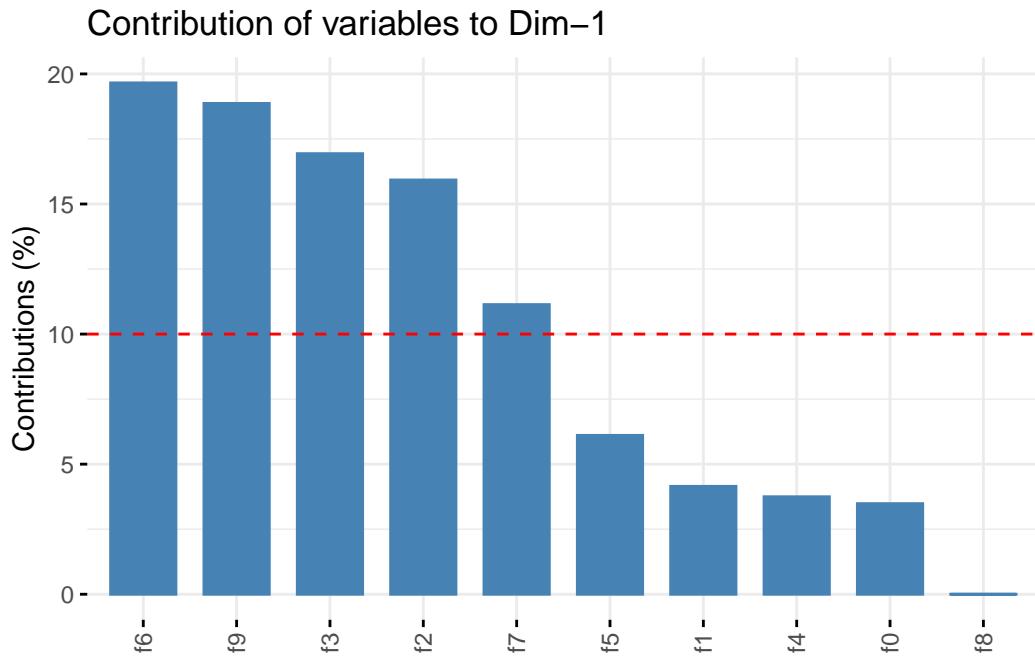


Observando el porcentaje de la varianza explicada, no parece un PCA muy bueno pues las 2 primeras dimensiones solo explican un 46.2% de la varianza, cuando un buen PCA debería explicar sobre un 70% de la varianza con las 2 primeras componentes. Sin embargo, si que podemos obtener información de las variables más importantes para cada componente y en sí para el propio conjunto de datos. Vamos a revisar las 2 primeras componentes.

Primera componente:

```
fviz_contrib(vowel_pca,
            choice = "var",
            axes = 1,
```

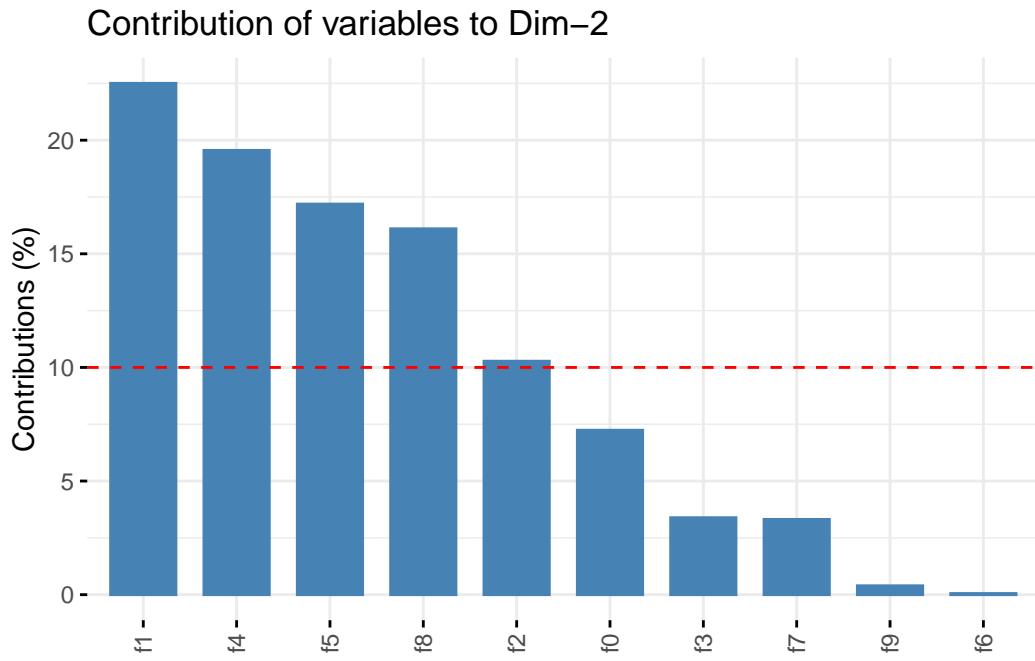
```
xtickslab.rt = 90,  
top=10)
```



Podemos observar como las variables f6, f9, f3, f2, y f7 son las relevantes para dicha dimensión.

Segunda componente:

```
fviz_contrib(vowel_pca,  
choice = "var",  
axes = 2,  
xtickslab.rt = 90,  
top=10)
```



Podemos observar como las variables f1, f4, f5, f8, y f2 son las relevantes para dicha dimensión.

5.4.2 Transformación de variables

En el caso de este problema no vamos probar ninguna transformación sobre las variables ya que mirando los histogramas de estas, tal y como están ya tienen distribuciones bastante similares a una distribución normal.

5.5 Conclusiones finales

5.5.1 Comprobación de las hipótesis

- ¿El sexo influyen en las distribuciones de las variables numéricas?

```
hist_by_sex <- function(variable) {
  ggplot(vowel, aes(x = .data[[variable]])) +
    geom_histogram(binwidth = 0.5, position = "identity", alpha = 0.7) +
    labs(title = paste("Histograma de", variable, "por Sexo"),
         x = variable, y = "Frecuencia") +
```

```

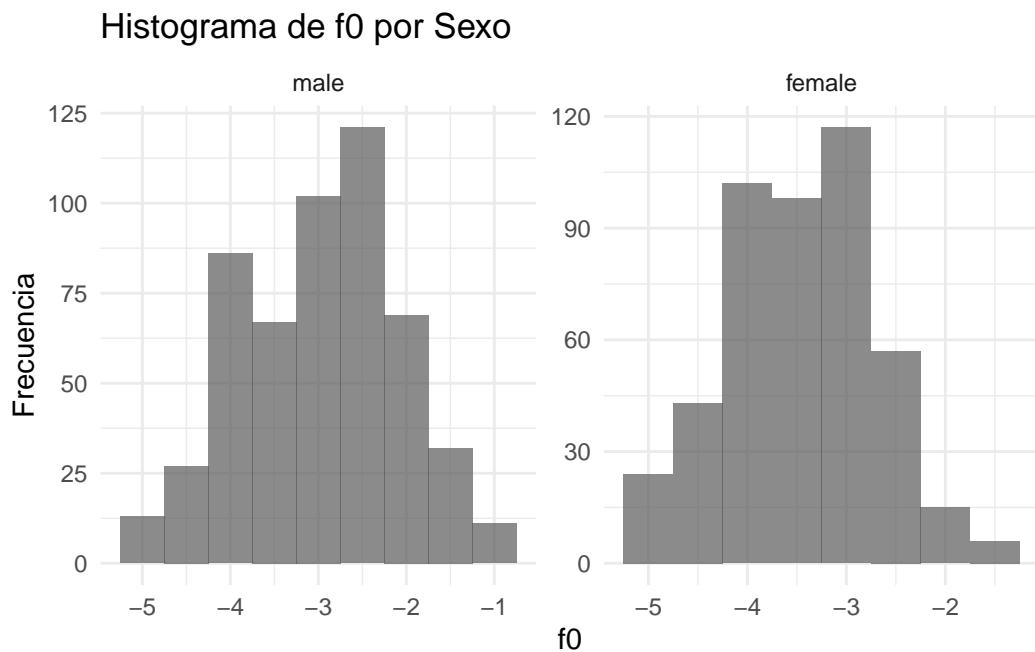
    scale_fill_manual(values = c("blue", "pink")) +
    facet_wrap(~ sex, scales = "free") +
    theme_minimal()
}

variables_to_plot <- c("f0", "f1", "f2", "f3", "f4", "f5", "f6", "f7", "f8", "f9")

lapply(variables_to_plot, hist_by_sex)

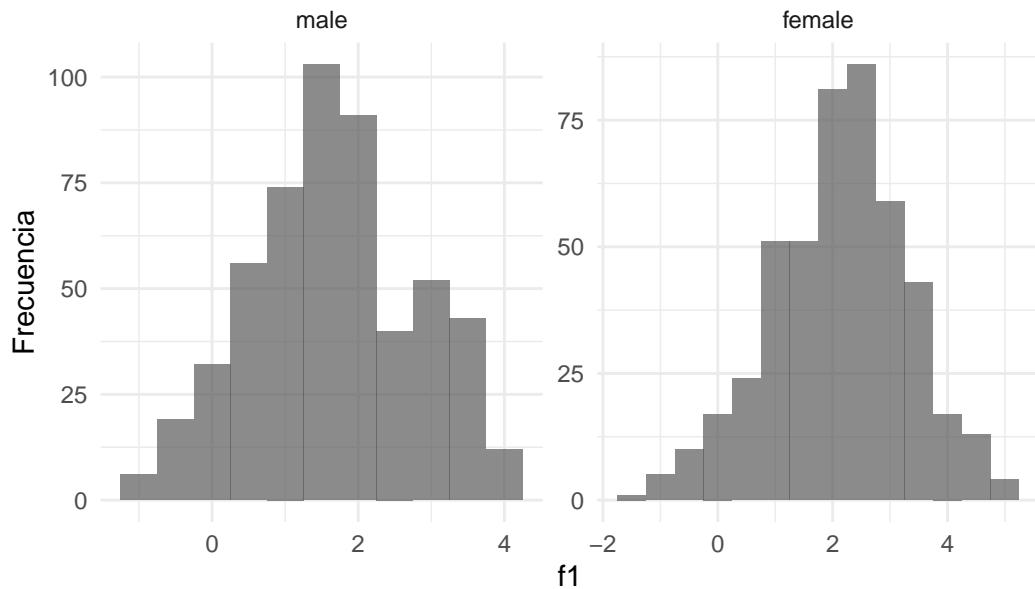
```

[[1]]



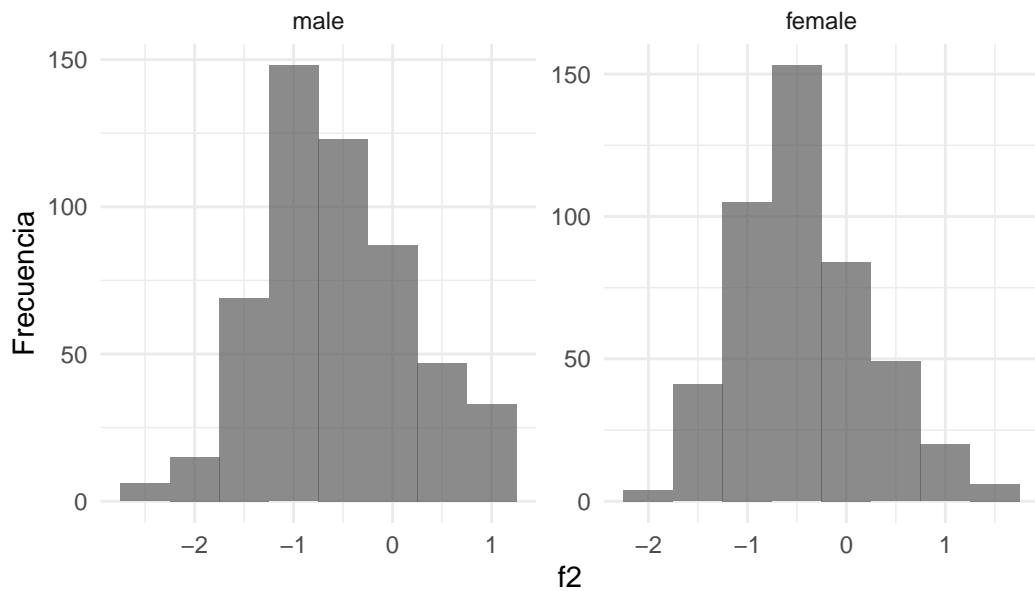
[[2]]

Histograma de f1 por Sexo



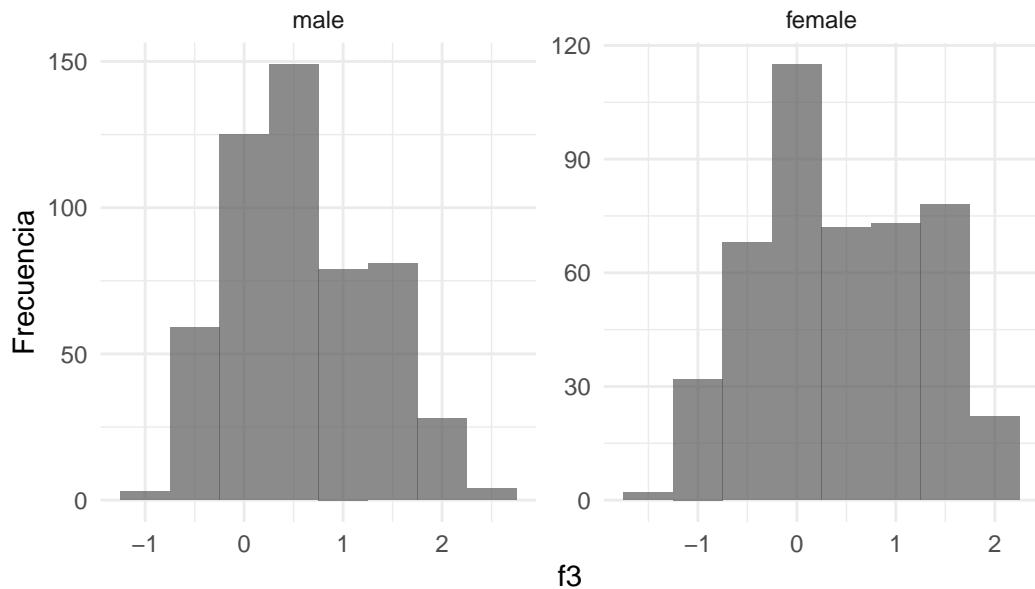
[[3]]

Histograma de f2 por Sexo



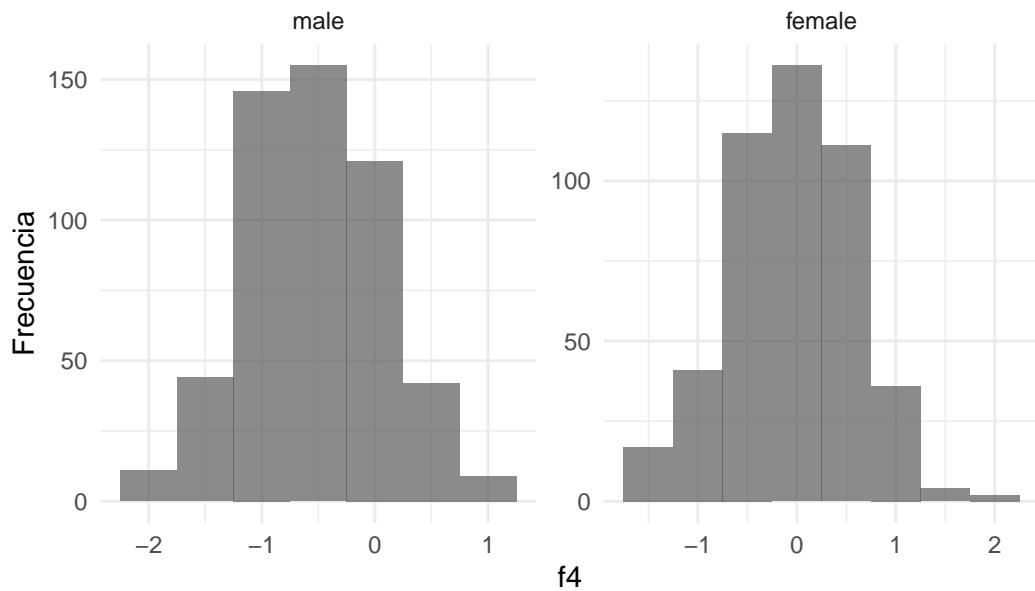
[[4]]

Histograma de f3 por Sexo



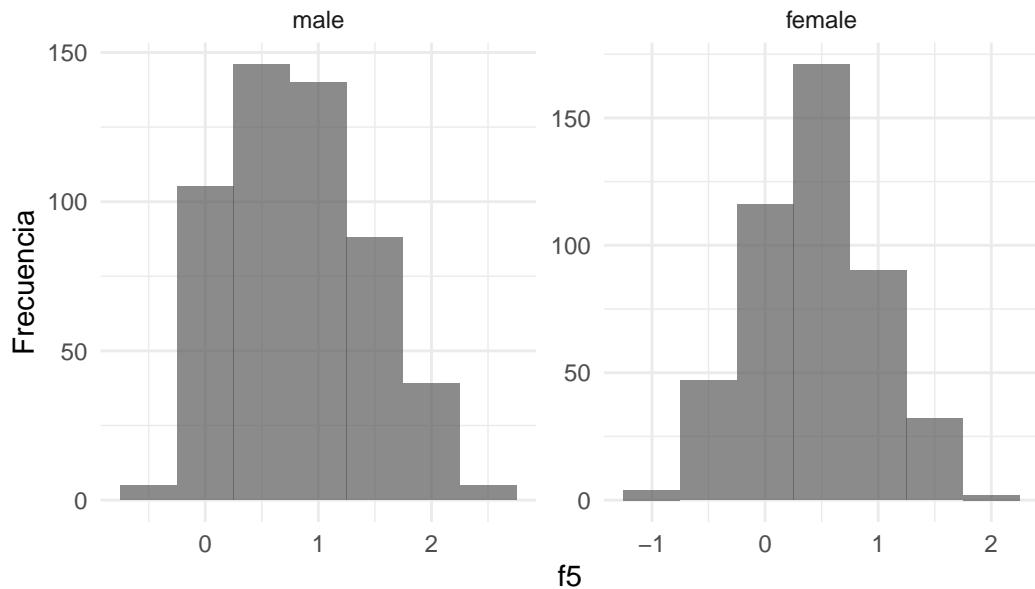
[[5]]

Histograma de f4 por Sexo



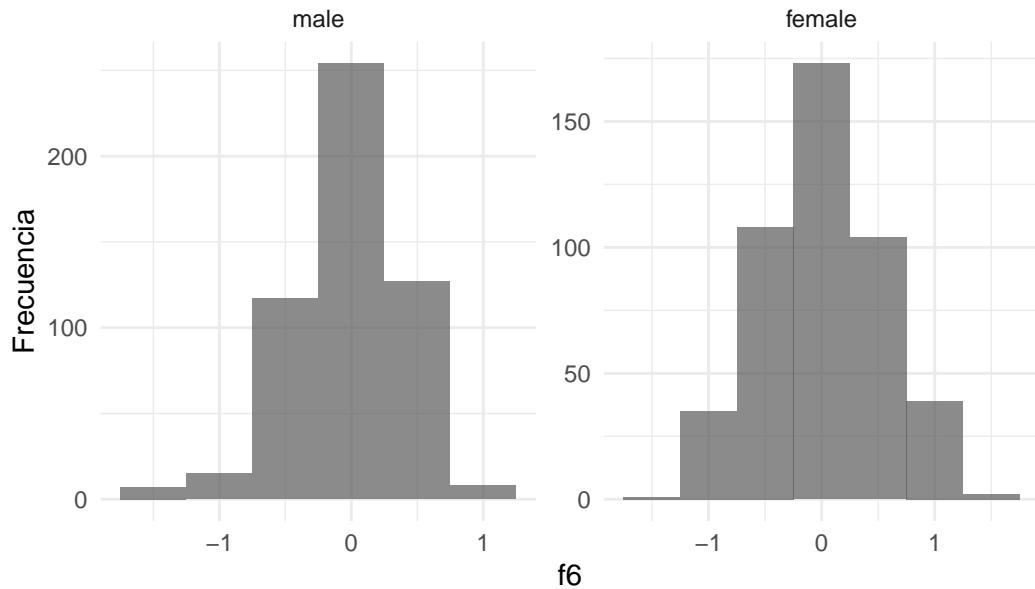
[[6]]

Histograma de f5 por Sexo



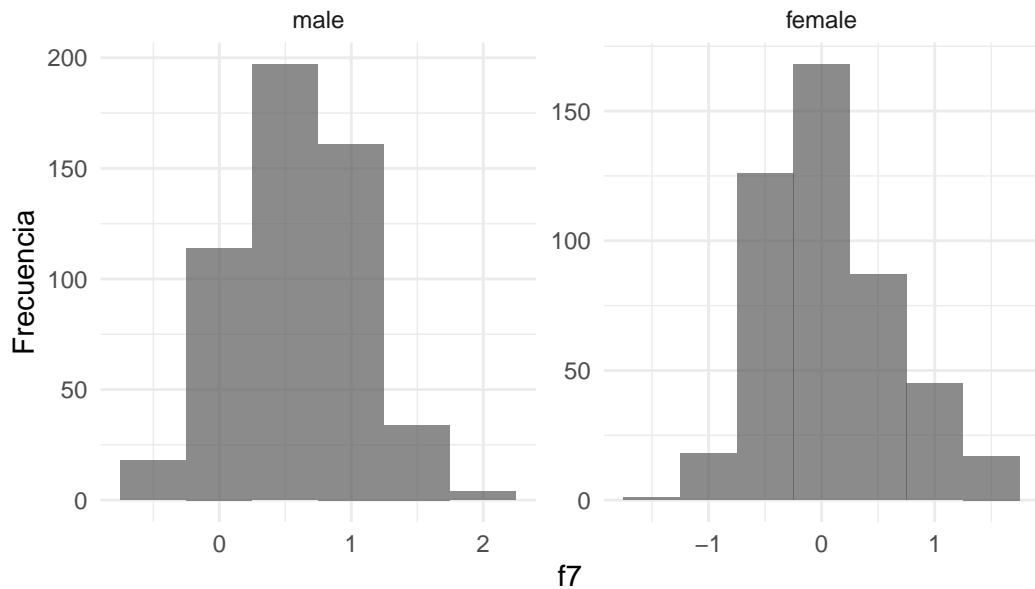
[[7]]

Histograma de f6 por Sexo



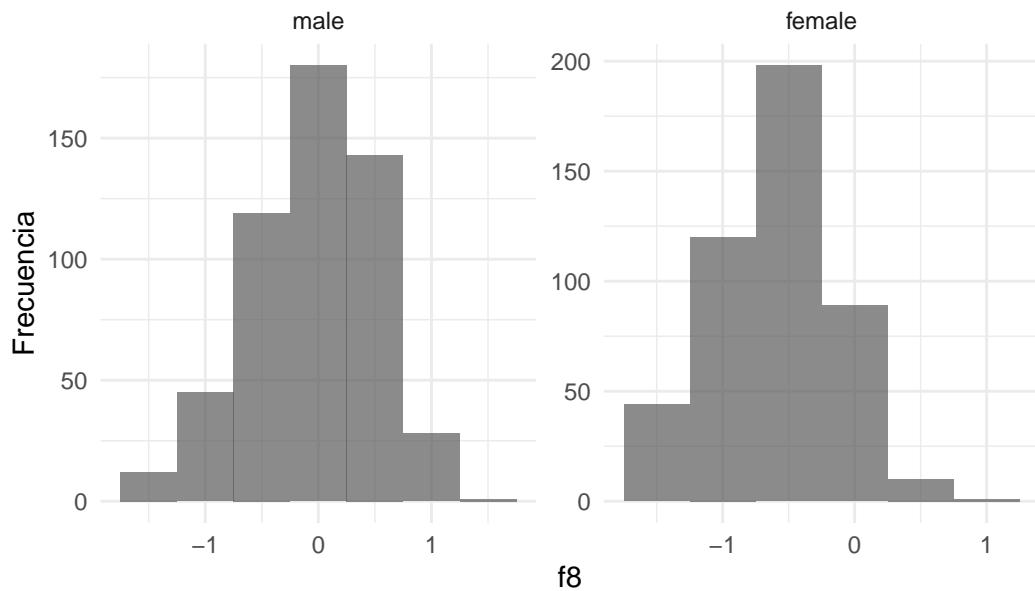
[[8]]

Histograma de f7 por Sexo



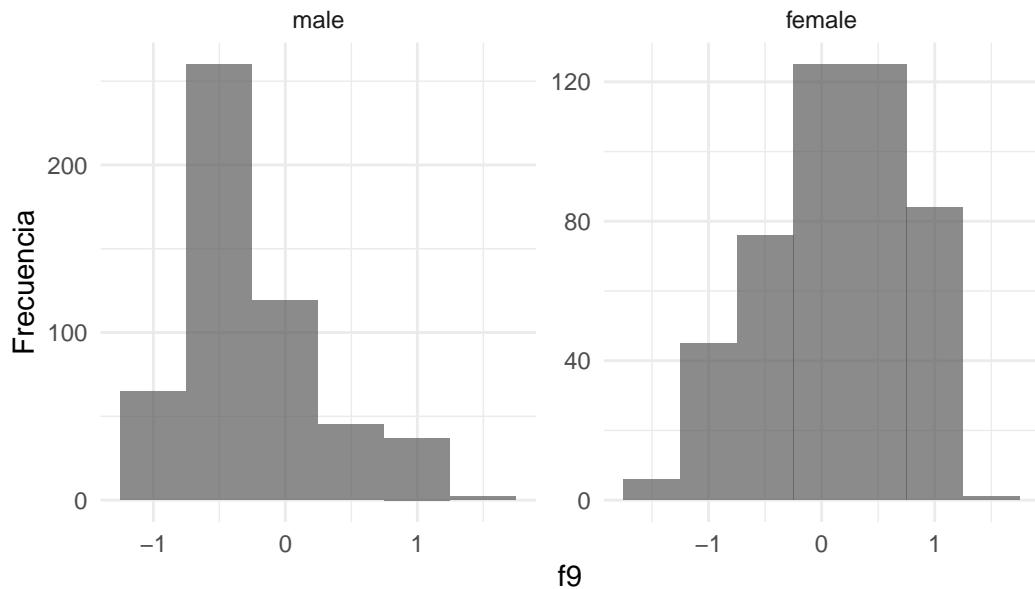
[[9]]

Histograma de f8 por Sexo



[[10]]

Histograma de f9 por Sexo

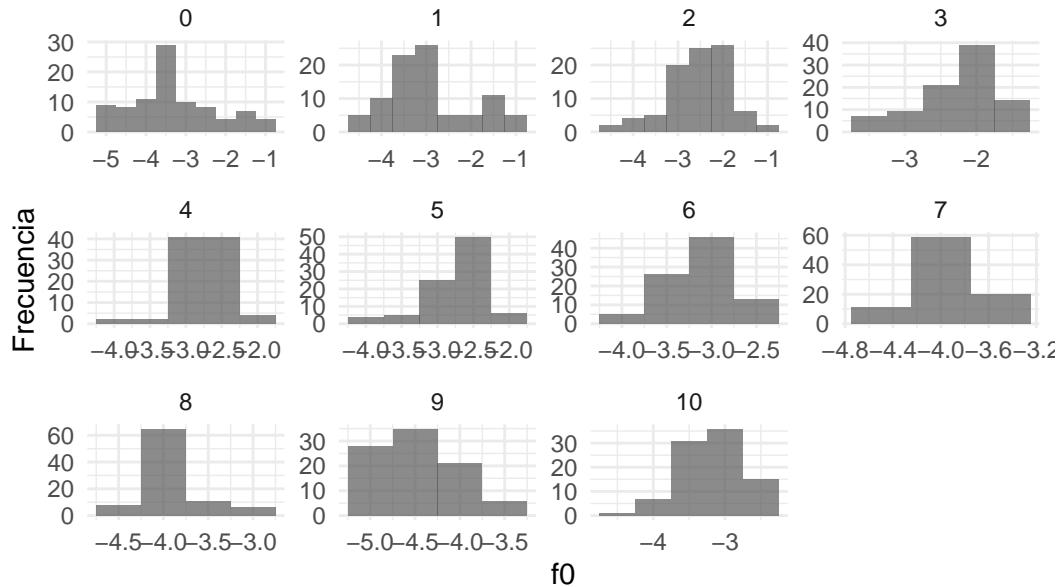


Como podemos ver, dependiendo del sexo la distribución de cada variable sufre una cierta modificación, mayor o menor dependiendo de la variable. Este es un indicio de que el sexo es información útil a la hora de la clasificación.

- ¿La clase objetivo influye en las distribuciones de las variables numéricas?

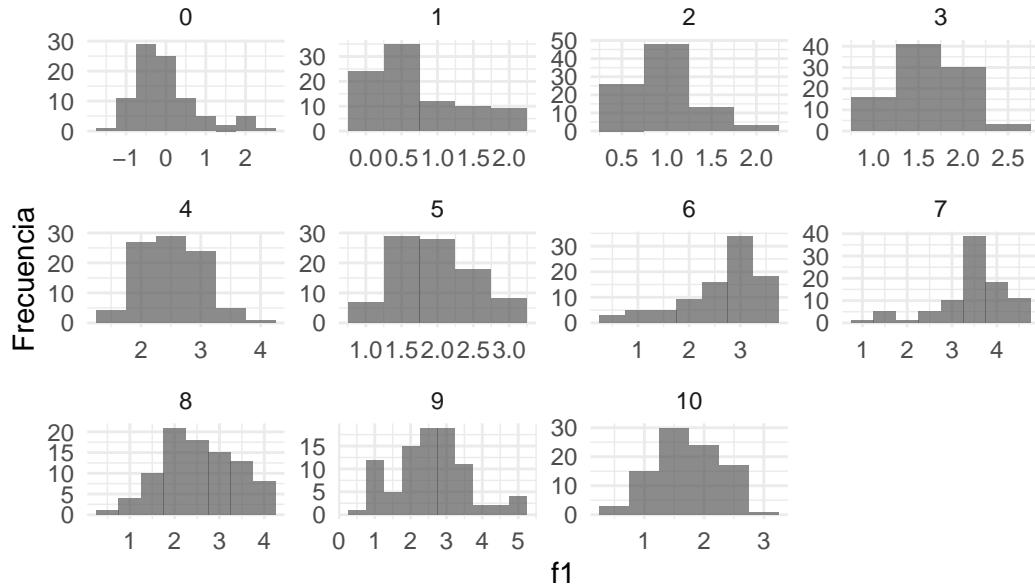
```
hist_by_class <- function(variable) {  
  ggplot(vowel, aes(x = .data[[variable]])) +  
    geom_histogram(binwidth = 0.5, position = "identity", alpha = 0.7) +  
    labs(title = paste("Histograma de", variable, "por class"),  
         x = variable, y = "Frecuencia") +  
    scale_fill_manual(values = c("blue", "pink")) +  
    facet_wrap(~ class, scales = "free") +  
    theme_minimal()  
}  
  
variables_to_plot <- c("f0", "f1", "f2", "f3", "f4", "f5", "f6", "f7", "f8", "f9")  
  
lapply(variables_to_plot, hist_by_class)  
  
[[1]]
```

Histograma de f0 por class



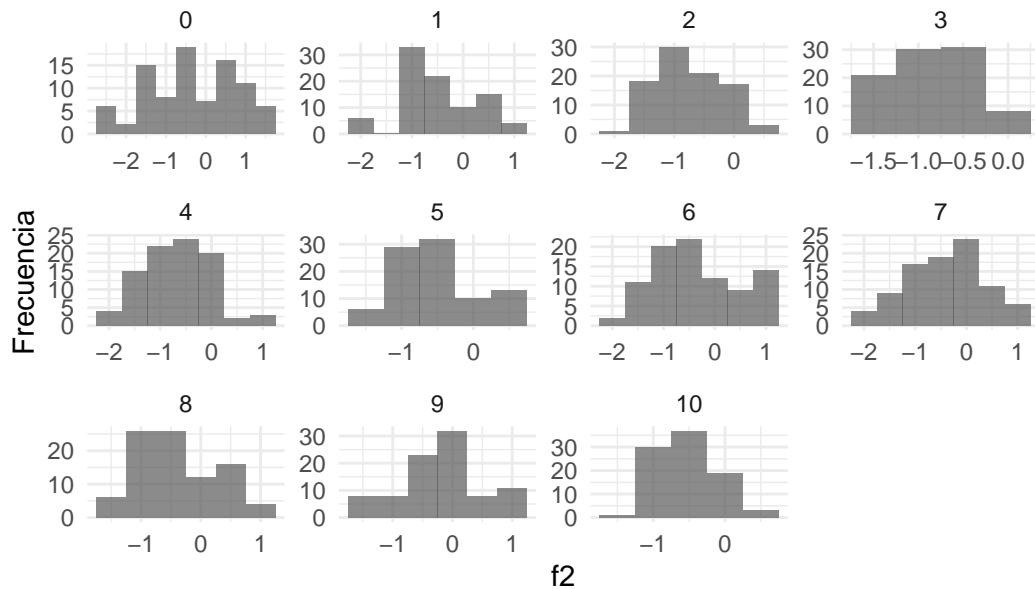
[[2]]

Histograma de f1 por class



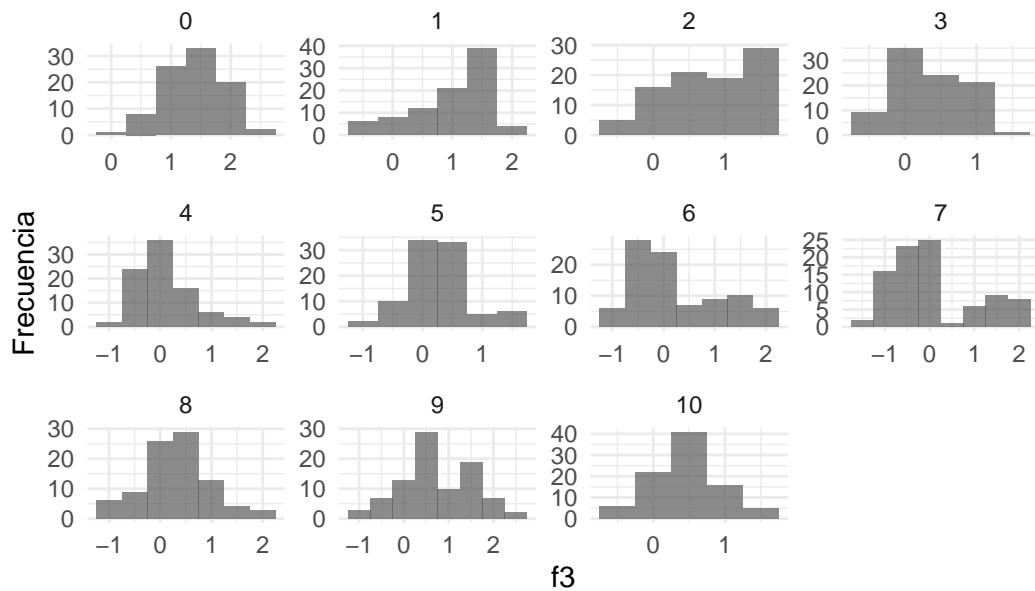
[[3]]

Histograma de f2 por class



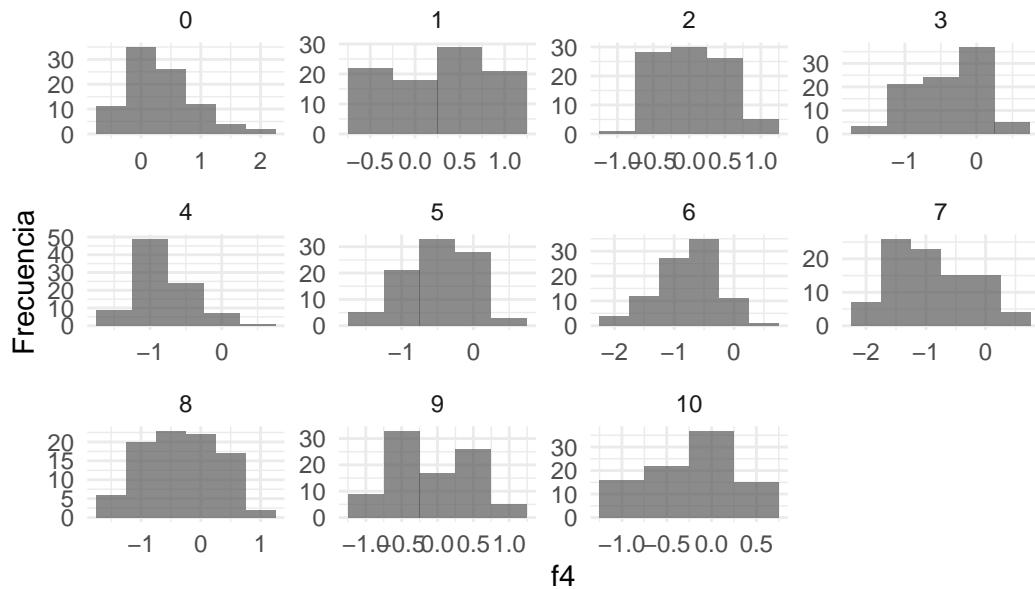
[[4]]

Histograma de f3 por class



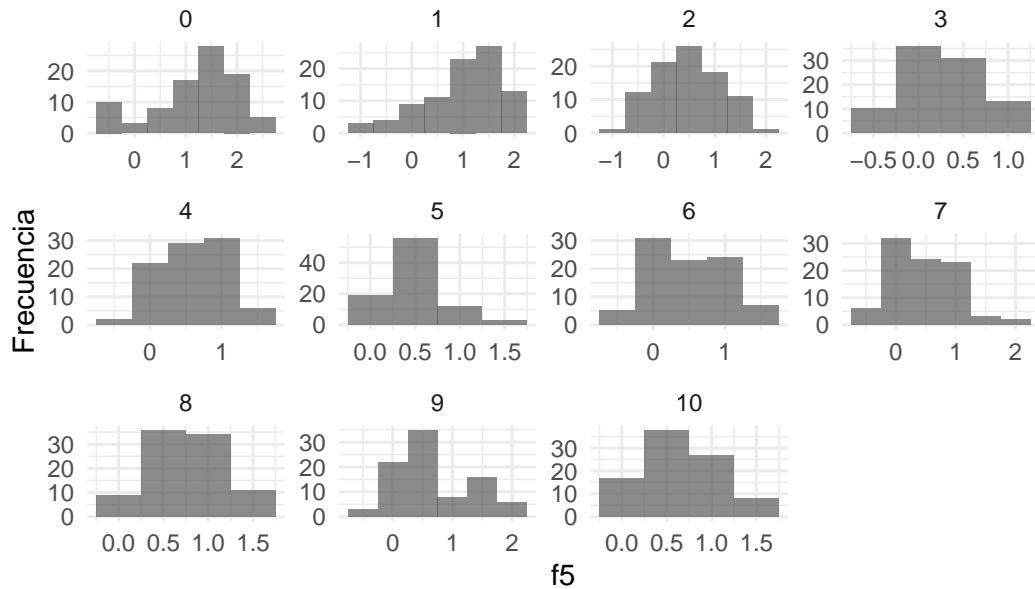
[[5]]

Histograma de f4 por class



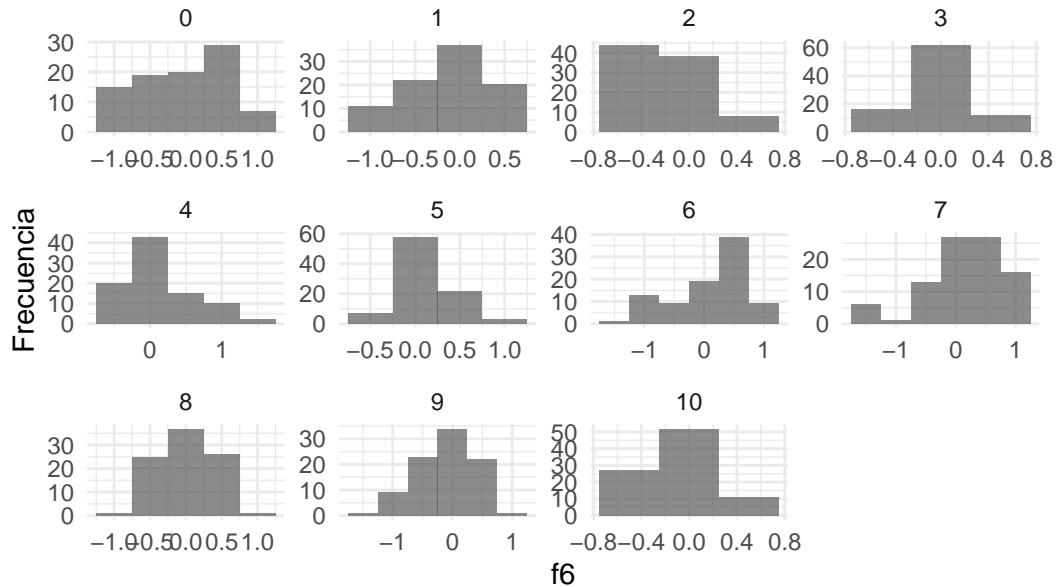
[[6]]

Histograma de f5 por class



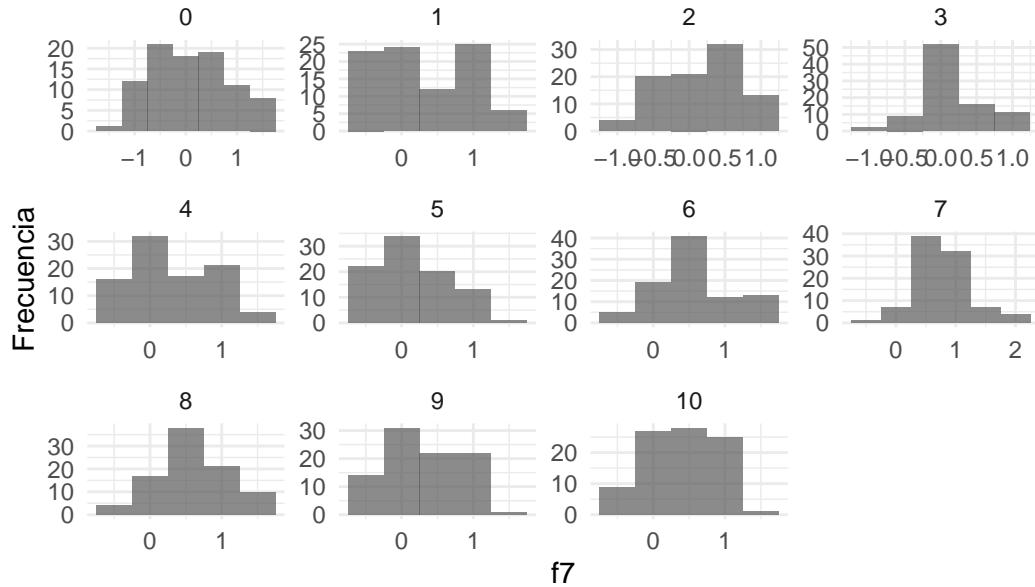
[[7]]

Histograma de f6 por class



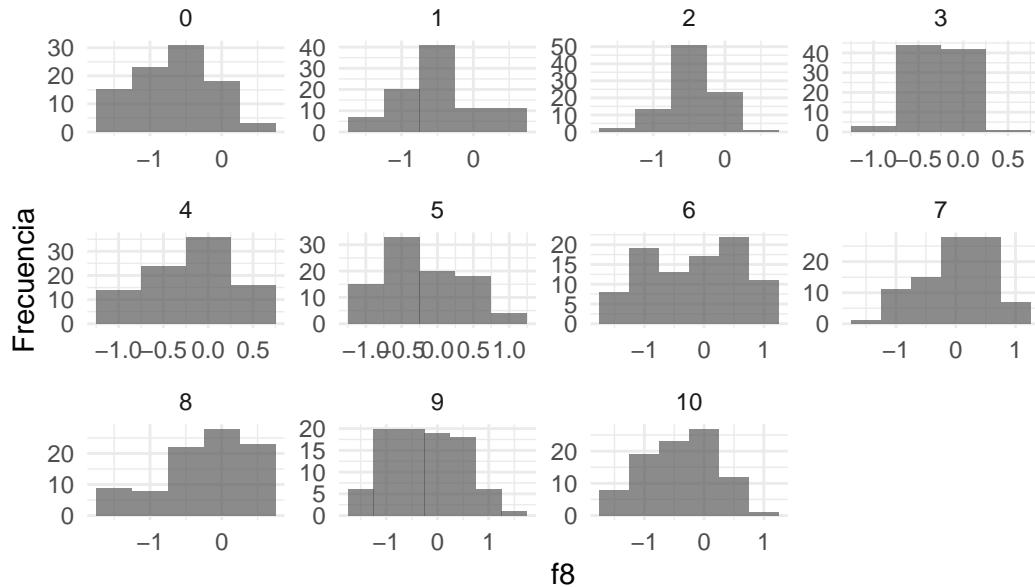
[[8]]

Histograma de f7 por class



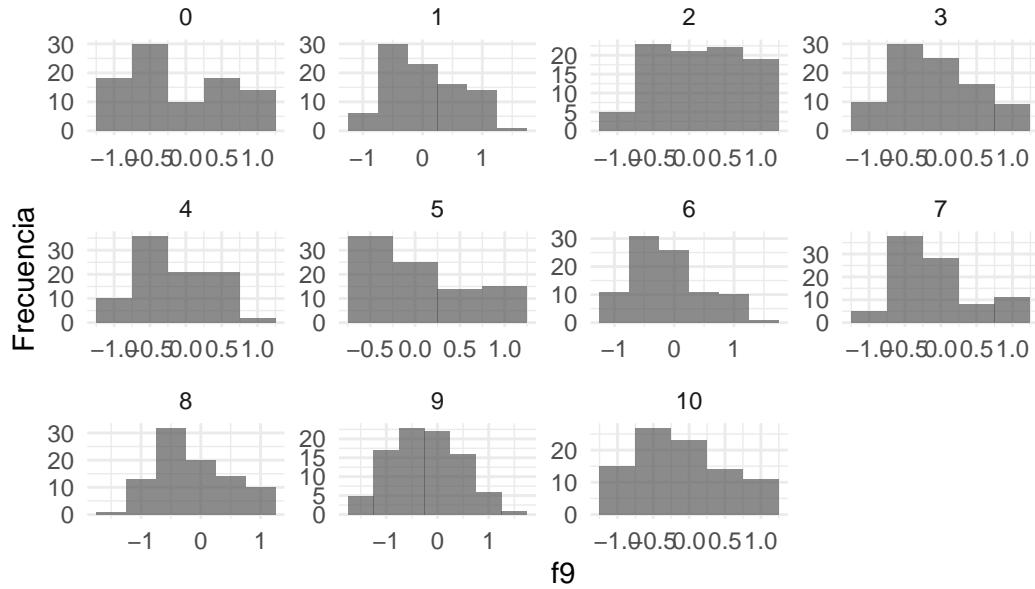
[[9]]

Histograma de f8 por class



[[10]]

Histograma de f9 por class



Como podemos observar, dependiendo del la clase a la que pertenece cada instancia, la distribución de cada variable numérica difiere bastante.

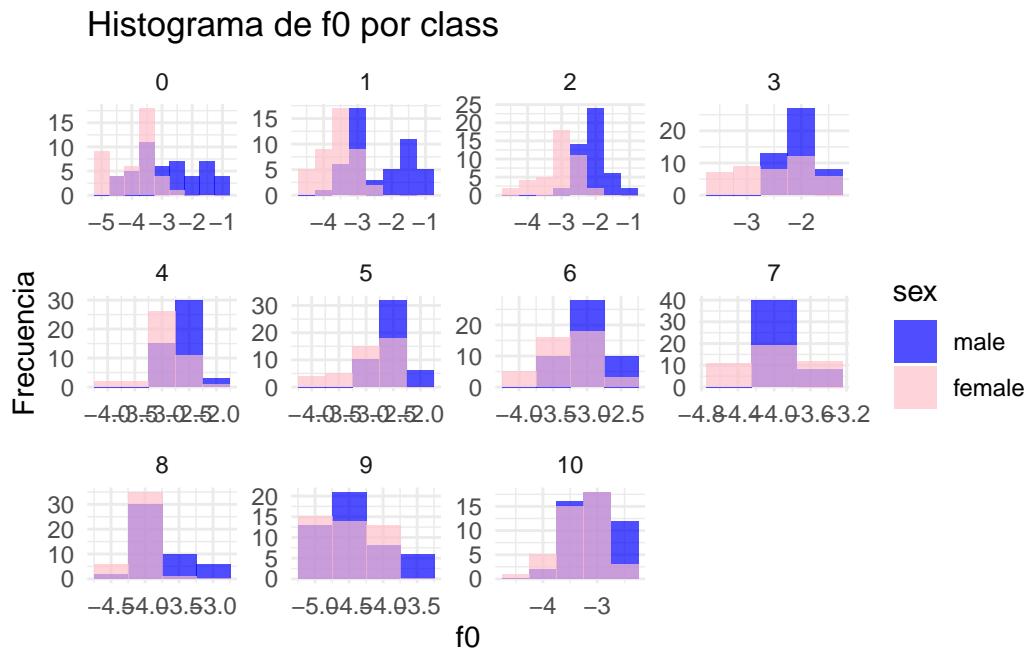
- ¿Y la clase y el sexo juntos?

```
hist_by_class_and_sex <- function(variable) {
  ggplot(vowel, aes(x = .data[[variable]], fill = sex)) +
    geom_histogram(binwidth = 0.5, position = "identity", alpha = 0.7) +
    labs(title = paste("Histograma de", variable, "por class"),
         x = variable, y = "Frecuencia") +
    scale_fill_manual(values = c("blue", "pink")) +
    facet_wrap(~ class, scales = "free") +
    theme_minimal()
}

variables_to_plot <- c("f0", "f1", "f2", "f3", "f4", "f5", "f6", "f7", "f8", "f9")

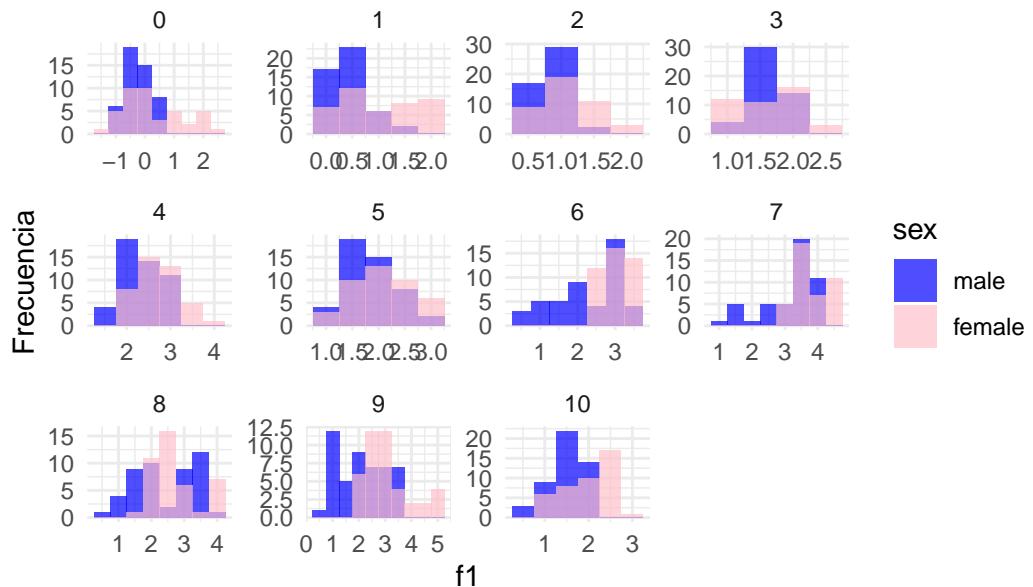
lapply(variables_to_plot, hist_by_class_and_sex)
```

[[1]]



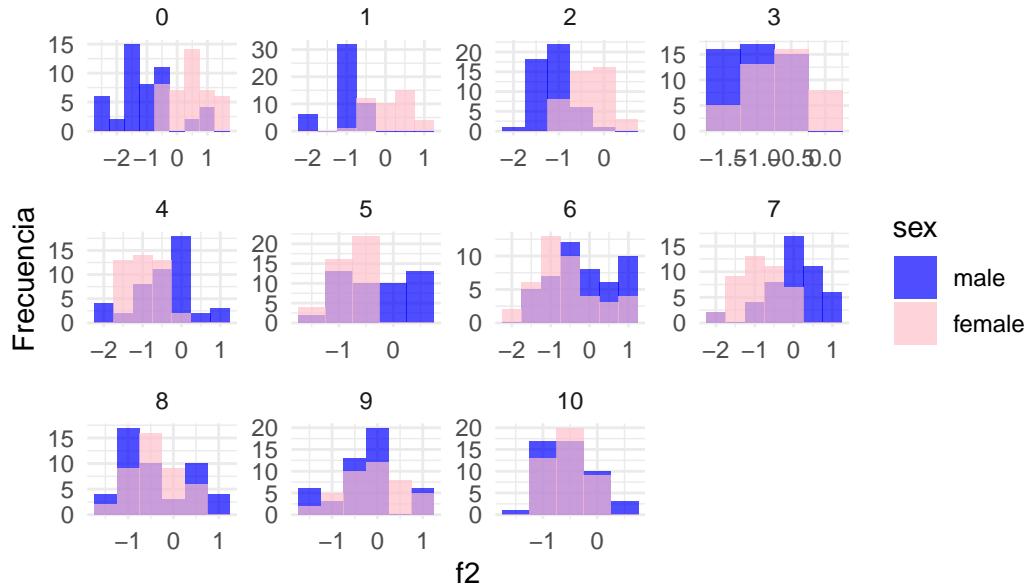
[[2]]

Histograma de f1 por class



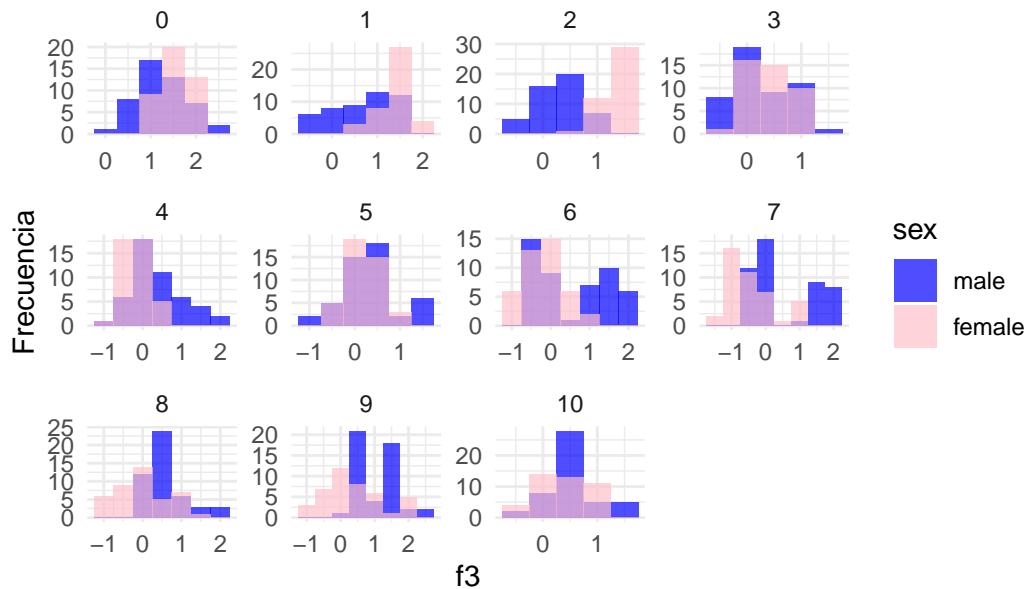
[[3]]

Histograma de f2 por class



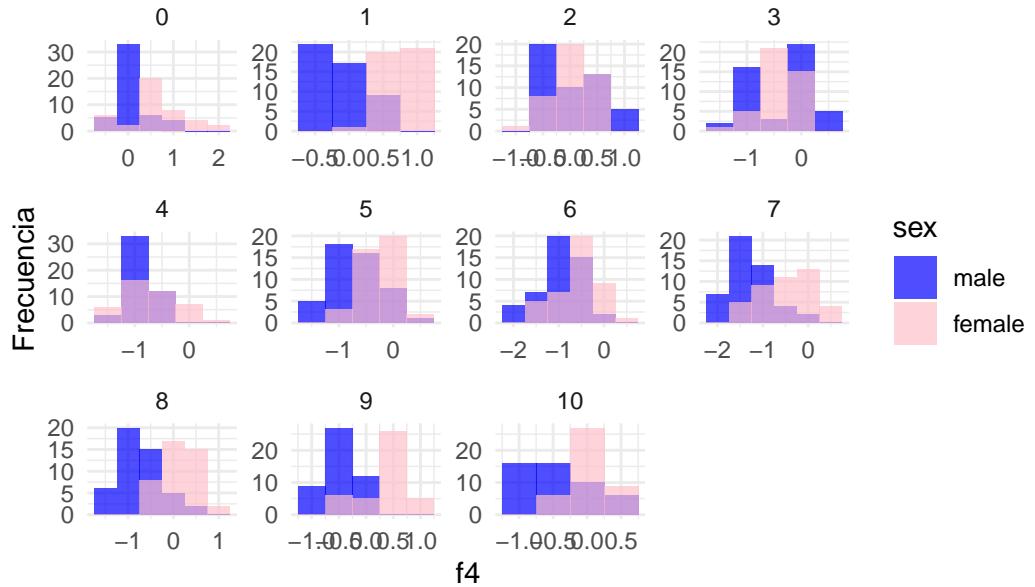
[[4]]

Histograma de f3 por class



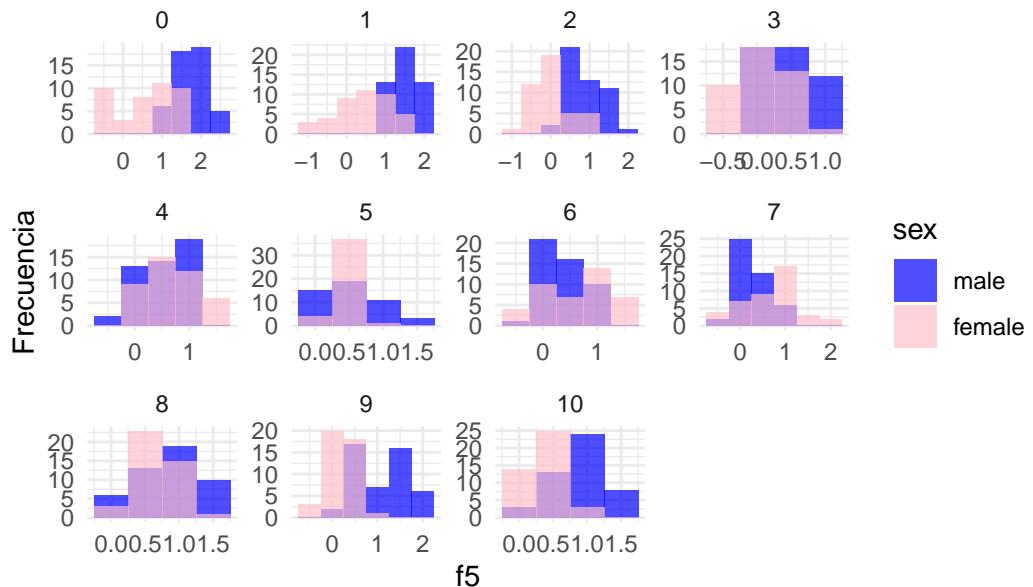
[[5]]

Histograma de f4 por class



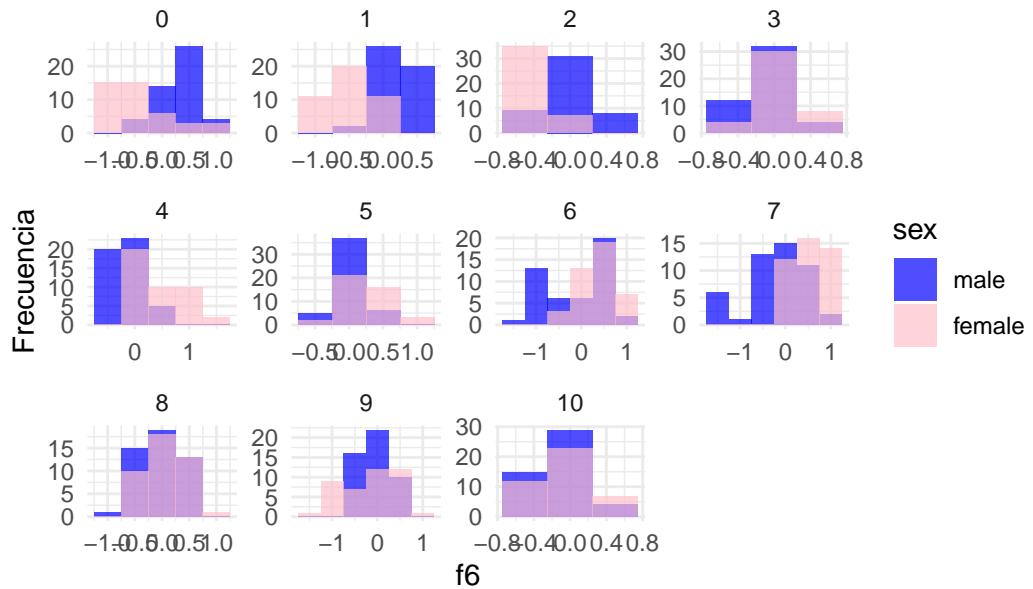
[[6]]

Histograma de f5 por class



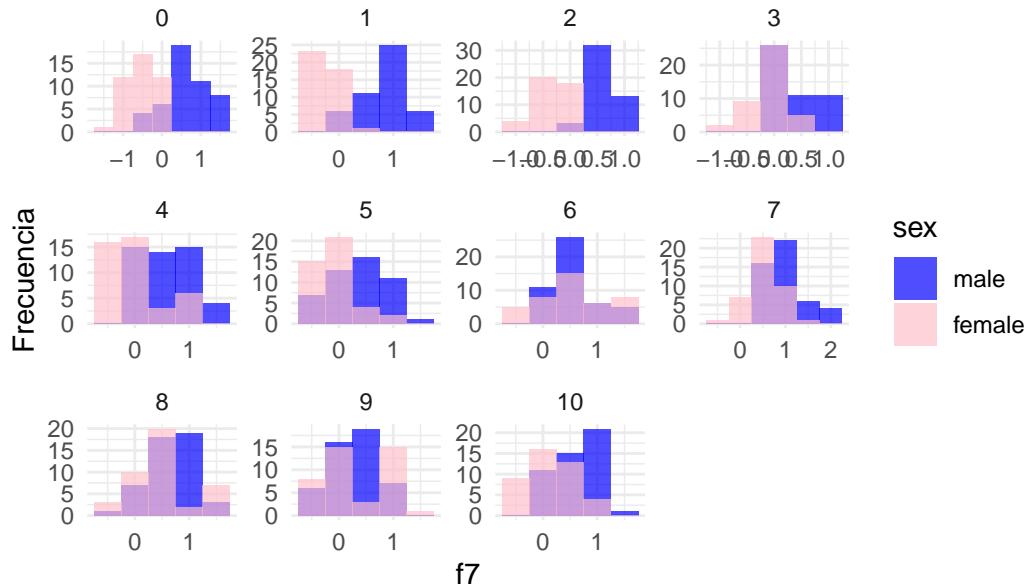
[[7]]

Histograma de f6 por class



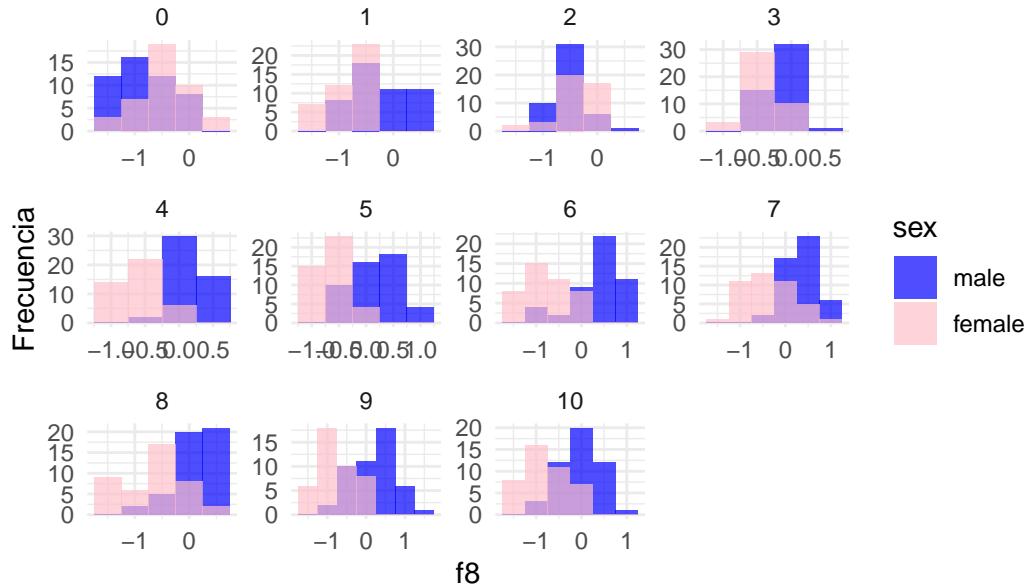
[[8]]

Histograma de f7 por class

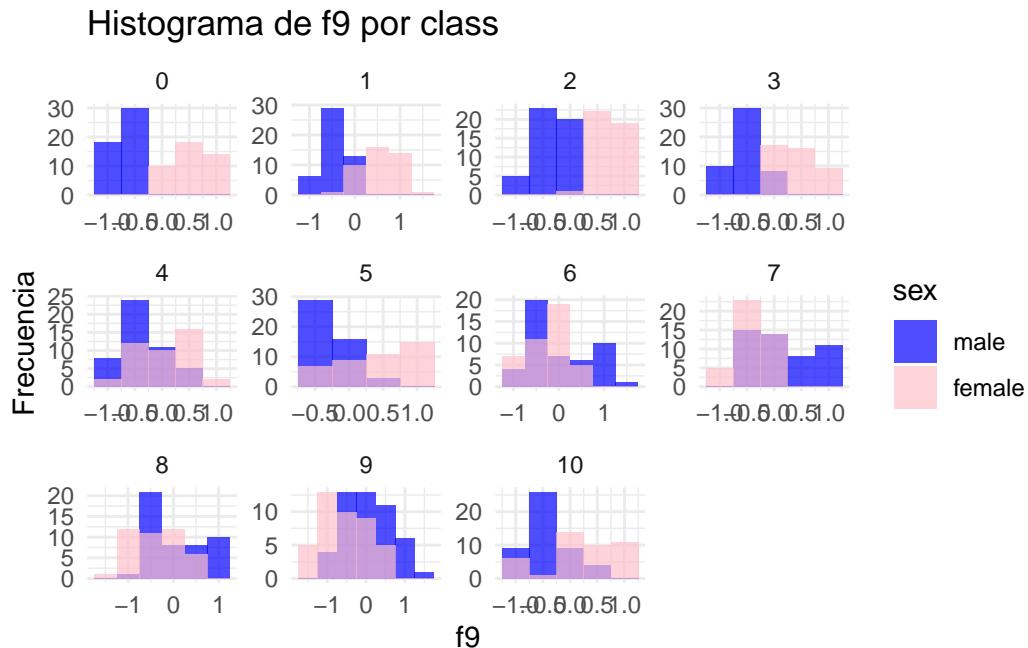


[[9]]

Histograma de f8 por class



[[10]]



De nuevo observamos que dependiendo del sexo, la distribuciones de las variables numéricas dependiendo de la variable objetivo difieren bastante.

5.5.2 Resumen del conjunto de datos

Tenemos un conjunto de datos en el que las variables siguen una distribución parecida a una normal, algo muy bueno de cara a la asunciones estadísticas que suponen distribuciones normales en las variables. En base al PCA, las características de las vocales más interesantes son f6, f9, f3, f2, y f7 pues son las que más explican la primera componente, y por parte de la segunda componente f1, f4, f5, f8, y f2 son las que más explican esta componente. También tenemos correlación entre las variables, algo que podrá influir negativamente en los modelos de clasificación.

6 Clasificación

Lo primero que vamos a hacer es leer las distintas particiones de entrenamiento y test que tenemos para nuestro conjunto de datos

```
library(MASS)
```

```
Attaching package: 'MASS'

The following object is masked from 'package:dplyr':
```

```
  select

library(kknn)

read_vowel_data <- function(index, data_type = "train") {
  file <- sprintf("vowel/vowel-10-%d%s.dat", index,
                  ifelse(data_type == "test", "tst", "tra"))
  data <- read.csv(file, comment.char="@", header=FALSE)

  data
}

train_list <- lapply(1:10, read_vowel_data, data_type = "train")
test_list <- lapply(1:10, read_vowel_data, data_type = "test")
```

A continuación vamos a aplicar el proceso de preparado de datos que hemos hecho durante el EDA:

```
apply_eda <- function(df) {
  colnames(df) <- c("tt", "speaker_number", "sex",
                    "f0", "f1", "f2", "f3",
                    "f4", "f5", "f6", "f7", "f8", "f9", "class")
  df$tt <- factor(df$tt, c(0,1), c("train", "test"))
  df$speaker_number <- factor(df$speaker_number, 0:14, as.character(0:14))
  df$sex <- factor(df$sex, c(0,1), c("male", "female"))
  df$class <- factor(df$class, 0:10, as.character(0:10))
  df <- df[,-(1:2)]

  df
}

train_list <- lapply(train_list, apply_eda)
test_list <- lapply(test_list, apply_eda)
```

También vamos a crear una lista con los datos normalizados para KNN:

```

apply_normalize <- function(df) {
  num_cols <- sapply(df, is.numeric)
  df[, num_cols] <- scale(df[, num_cols])

  df
}

train_list_normalized <- lapply(train_list, apply_normalize)
test_list_normalized <- lapply(test_list, apply_normalize)

```

Ahora vamos a crear una función que aplique knn usando a las distintas particiones de entrenamiento y test y nos calcule la precisión que obtenemos

```

cv_knn_accuracy <- function(formula, k, train_list, test_list, ...) {
  sapply(1:length(train_list), function(i) {
    fit <- kknn(formula, train_list[[i]], test_list[[i]], k=k, ...)
    preds <- fit$fitted.values

    sum(preds==test_list[[i]]$class)/length(preds)
  })
}

```

Vamos a probar con los siguientes valores de k: 1, 5, 10, 50 y 100

```

cv_knn_1_train <- cv_knn_accuracy(class~, 1,
                                      train_list_normalized,
                                      train_list_normalized)
cv_knn_1_test <- cv_knn_accuracy(class~, 1,
                                      train_list_normalized,
                                      test_list_normalized)
cv_knn_5_train <- cv_knn_accuracy(class~, 5,
                                      train_list_normalized,
                                      train_list_normalized)
cv_knn_5_test <- cv_knn_accuracy(class~, 5,
                                      train_list_normalized,
                                      test_list_normalized)
cv_knn_10_train <- cv_knn_accuracy(class~, 10,
                                       train_list_normalized,
                                       train_list_normalized)
cv_knn_10_test <- cv_knn_accuracy(class~, 10,
                                       train_list_normalized,

```

```

                test_list_normalized)
cv_knn_50_train <- cv_knn_accuracy(class~, 50,
                                      train_list_normalized,
                                      train_list_normalized)
cv_knn_50_test <- cv_knn_accuracy(class~, 50,
                                     train_list_normalized,
                                     test_list_normalized)
cv_knn_100_train <- cv_knn_accuracy(class~, 100,
                                       train_list_normalized,
                                       train_list_normalized)
cv_knn_100_test <- cv_knn_accuracy(class~, 100,
                                      train_list_normalized,
                                      test_list_normalized)

l_train <- rbind(cv_knn_1_train, cv_knn_5_train, cv_knn_10_train,
                  cv_knn_50_train, cv_knn_100_train)
l_test <- rbind(cv_knn_1_test, cv_knn_5_test, cv_knn_10_test,
                 cv_knn_50_test, cv_knn_100_test)

apply(l_train, 1, mean)

cv_knn_1_train   cv_knn_5_train   cv_knn_10_train   cv_knn_50_train
1.0000000      0.9983165      0.9936027      0.9135802
cv_knn_100_train
0.8320988

apply(l_test, 1, mean)

cv_knn_1_test   cv_knn_5_test   cv_knn_10_test   cv_knn_50_test cv_knn_100_test
0.9909091      0.9848485      0.9737374      0.8565657      0.7777778

```

Podemos observar que hasta $k = 5$ no observamos un empeoramiento muy significativo en la precisión obtenida. A partir de ahí empieza a empeorar gradualmente.

Comparando el resultado obtenido en test frente a entrenamiento para cada k , vemos que siempre la precisión obtenida en train es superior a la obtenida en test, algo lógico.

En vista de los resultados el mejor valor de k es 1, pues es con el que obtenemos mejores resultados, y a la vez es el que consume menos computacionalmente ya que tenemos que buscar sólo el vecino más cercano.

6.1 LDA

El primer paso con LDA es comprobar las hipótesis de partida:

- Que las observaciones siguen una distribución normal para cada predictor.

```
my_shapiro <- function(a) {  
  shapiro.test(a)$statistic  
}  
  
vowel[,-1] %>%  
  group_by(class) %>%  
  summarise(across(f0:f9, my_shapiro))  
  
# A tibble: 11 x 11  
#>   class     f0     f1     f2     f3     f4     f5     f6     f7     f8     f9  
#>   <dbl>  
#> 1 0       0.951 0.890 0.960 0.972 0.959 0.906 0.930 0.960 0.967 0.912  
#> 2 1       0.924 0.900 0.959 0.894 0.940 0.933 0.968 0.950 0.974 0.952  
#> 3 2       0.973 0.975 0.973 0.945 0.959 0.977 0.937 0.966 0.969 0.946  
#> 4 3       0.901 0.974 0.976 0.971 0.941 0.958 0.968 0.954 0.974 0.961  
#> 5 4       0.930 0.977 0.973 0.908 0.972 0.970 0.950 0.952 0.969 0.954  
#> 6 5       0.934 0.982 0.949 0.971 0.977 0.967 0.956 0.947 0.954 0.935  
#> 7 6       0.985 0.892 0.947 0.905 0.975 0.961 0.891 0.960 0.952 0.956  
#> 8 7       0.980 0.858 0.977 0.890 0.958 0.966 0.927 0.961 0.953 0.893  
#> 9 8       0.939 0.983 0.967 0.978 0.966 0.988 0.972 0.990 0.943 0.970  
#> 10 9      0.967 0.965 0.970 0.978 0.955 0.955 0.984 0.957 0.967 0.981  
#> 11 10     0.964 0.984 0.974 0.975 0.961 0.943 0.977 0.965 0.979 0.960
```

Como podemos observar, las mayoría de los resultados son bastante cercanos a 1, lo cual significa que los datos parecen que se ajustan bien a una distribución normal.

- Que las matrices de varianza-covarianza son homogéneas.

Para ello vamos a aplicar el test de Bartlett.

```
bartlett.test(f0 ~ class, vowel)  
  
Bartlett test of homogeneity of variances  
  
data: f0 by class  
Bartlett's K-squared = 390.14, df = 10, p-value < 2.2e-16
```

```
bartlett.test(f1 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f1 by class
Bartlett's K-squared = 173.97, df = 10, p-value < 2.2e-16

bartlett.test(f2 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f2 by class
Bartlett's K-squared = 153.91, df = 10, p-value < 2.2e-16

bartlett.test(f3 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f3 by class
Bartlett's K-squared = 119.96, df = 10, p-value < 2.2e-16

bartlett.test(f4 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f4 by class
Bartlett's K-squared = 40.711, df = 10, p-value = 1.269e-05

bartlett.test(f5 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f5 by class
Bartlett's K-squared = 152.15, df = 10, p-value < 2.2e-16
```

```

bartlett.test(f6 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f6 by class
Bartlett's K-squared = 173.87, df = 10, p-value < 2.2e-16

bartlett.test(f7 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f7 by class
Bartlett's K-squared = 53.454, df = 10, p-value = 6.131e-08

bartlett.test(f8 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f8 by class
Bartlett's K-squared = 151.77, df = 10, p-value < 2.2e-16

bartlett.test(f9 ~ class, vowel)

Bartlett test of homogeneity of variances

data: f9 by class
Bartlett's K-squared = 16.109, df = 10, p-value = 0.09654

```

Las pruebas de Bartlett sugieren que, en general, las varianzas no son homogéneas entre los grupos para la mayoría de las variables, excepto posiblemente para f9, donde la evidencia es más débil para rechazar la igualdad de varianzas.

No se da esta última condición pero igualmente vamos a aplicar LDA para ver qué resultado obtenemos:

```

cv_method_accuracy <- function(method, formula, train_list, test_list) {
  sapply(1:length(train_list), function(i) {

```

```

fit <- method(formula,data=train_list[[i]])
preds <- predict(fit, test_list[[i]])

sum(preds$class == test_list[[i]]$class) / length(preds$class)
})

lda_cv_test <- cv_method_accuracy(lda, class ~ ., train_list, test_list)
lda_cv_train <- cv_method_accuracy(lda, class ~ ., train_list, train_list)

mean(lda_cv_train)

[1] 0.6482604

mean(lda_cv_test)

[1] 0.6030303

```

Como podemos ver, obtenemos un bastante peor resultado que el obtenido en KNN.

6.2 QDA

Como hemos visto antes, las varianzas no son homogéneas entre los grupos para la mayoría de las variables. Esto significa que probablemente obtengamos un mejor resultado en QDA en en LDA.

Antes de ejecutar QDA debemos comprobar las siguientes asunciones:

- El número de predictores (variables independientes) debe ser menor que el número de casos dentro de cada clase: tenemos 11 predictores (lo hemos revisado en el EDA en el apartado de estructura del conjunto de datos) y por cada clase de la variable objetivo tenemos 90 instancias (lo hemos revisado en el EDA en el apartado de tablas de contingencia), por tanto esta asunción se cumple.
- Los predictores dentro de cada clase no tengan niveles patológicos de colinealidad: en este caso, a partir del EDA sabemos que entre las variables independientes existen una correlación a tener en cuenta (lo hemos revisado en el apartado grafico de gráfico de puntos variable independiente vs independiente), por lo que puede que esta asunción no se cumple en su totalidad.

Una vez revisadas las asunciones y viendo que una se cumple completamente y la otra no del todo, ejecutamos QDA para ver qué resultado obtenemos:

NOTA: se ha tenido que eliminar la variable sex del modelo pues debido al desequilibrio que presenta entre las clases de dicha variable no permite ejecutar QDA.

```
qda_cv_test <- cv_method_accuracy(qda, class ~ . - sex, train_list, test_list)
qda_cv_train <- cv_method_accuracy(qda, class ~ . - sex, train_list, train_list)

mean(qda_cv_train)
```

```
[1] 0.9338945
```

```
mean(qda_cv_test)
```

```
[1] 0.8828283
```

Tal y como pensabamos hemos obtenido unos mucho mejores resultados en QDA que en LDA, esto es debido a que QDA funcionará mejor cuando las varianzas sean muy diferentes entre clases.

6.3 Comparación de resultados:

Como hemos visto los modelos que han tenido un mejor desempeño son KNN, QDA y LDA, en ese orden. Las razones de que esto sea así son:

- Que KNN nos dé los mejores resultados es debido a que los límites de decisión son claramente no lineales en este dataset, tal y como vimos en el apartado de separabilidad de las variables por clases en el EDA. Cuanto más complejos sean los límites de decisión, mejor se comportará KNN frente a LDA.
- Que QDA nos dé unos resultados intermedios entre LDA Y KNN es debido a que este es un compromiso entre el método no paramétrico k-NN y el método lineal LDA.