AlwaysR, Módulo III: Estadística en R.

Lab 03

Giancarlo M. Correa

En esta sección se resume los conceptos más importantes vistos en la parte práctica de cada clase.

Cargamos librerías a utilizar

```
library(ggplot2)
library(Sleuth3)
library(faraway)
library(jtools)
library(ggstance)
library(broom.mixed)
```

Bases de datos a utilizar:

```
head(cheddar)
```

```
##   taste Acetic   H2S Lactic
## 1  12.3  4.543 3.135   0.86
## 2  20.9  5.159 5.043   1.53
## 3  39.0  5.366 5.438   1.57
## 4  47.9  5.759 7.496   1.81
## 5   5.6  4.663 3.807   0.99
## 6  25.9  5.697 7.601   1.09
```

Si queremos saber que información contiene cada base de datos, podemos ver la ayuda de ellas mediante `?cheddar`.

Modelos de regresión simple

Queremos evaluar cómo varía la media de la variable `taste` en función a la variable `H2S`. Implementamos el modelo lineal y observamos la tabla resumen:

```
mod1 = lm(taste ~ H2S, data = cheddar)
summary(mod1)
```

```
##
## Call:
## lm(formula = taste ~ H2S, data = cheddar)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -15.426  -7.611  -3.491   6.420  25.687
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.7868     5.9579  -1.643    0.112
## H2S           5.7761     0.9458   6.107 1.37e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.83 on 28 degrees of freedom
## Multiple R-squared:  0.5712, Adjusted R-squared:  0.5558
## F-statistic: 37.29 on 1 and 28 DF,  p-value: 1.374e-06
```

Interpretación de la pendiente β_1: el valor medio de `taste` aumenta en 5.7761 cuando el valor de `H2S` aumenta en una unidad.

De esta tabla, podemos extraer información importante que puede ser reportada en nuestro estudio:

```
mod1$coefficients # Coeficientes
```

```
## (Intercept)         H2S
##   -9.786837    5.776089
```

```
mod1$residuals # Residuales
```

```
##          1          2          3          4          5          6          7          8          9
##   3.978800   1.558023  17.376468  14.389277  -6.602732  -8.217212  -3.315312 -14.325484   5.648896
##         13         14         15         16         17         18         19         20         21
##  -6.818324   7.141408  25.686687  -4.694300   3.090779 -10.960779  -7.874734  -3.667630  -4.799025
##         25         26         27         28         29         30
##   5.423249   2.793608  -3.926291 -12.109221 -15.426315 -12.363299
```

```
mod1$fitted.values # Valores ajustados (sobre la linea ajustada) para las observaciones
```

```
##         1         2         3         4         5         6         7         8         9        10
##  8.321200 19.341977 21.623532 33.510723 12.202732 34.117212 40.615312 36.225484 12.451104 14.322556
##        14        15        16        17        18        19        20        21        22        23
## 18.758592 29.213313 45.594300 12.809221 17.360779 25.874734 42.567630 18.799025 20.364345 43.595773
##        27        28        29        30
## 30.426291 12.809221 28.826315 17.863299
```

También podemos extraer información de la tabla resumen:

```
mySummary = summary(mod1)
mySummary$coefficients # coeficientes
```

```
##              Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) -9.786837    5.95791 -1.642663 1.116377e-01
## H2S          5.776089    0.94585  6.106770 1.373783e-06
```

```
mySummary$sigma # desviacion estandar de los residuales
```

```
## [1] 10.83338
```

```
mySummary$adj.r.squared # R2-adj
```

```
## [1] 0.5558458
```

Si queremos encontrar los intervalos asociados a la línea ajustada:

```
predict(object = mod1, interval = 'confidence')
```

```
##          fit        lwr      upr
## 1   8.321200  1.5397877 15.10261
## 2  19.341977 14.9320752 23.75188
## 3  21.623532 17.4560870 25.79098
## 4  33.510723 28.4626634 38.55878
## 5  12.202732  6.4129042 17.99256
## 6  34.117212 28.9452197 39.28920
## 7  40.615312 33.8688584 47.36177
## 8  36.225484 30.5866522 41.86432
## 9  12.451104  6.7204950 18.18171
## 10 14.322556  9.0173018 19.62781
## 11 25.689899 21.6198312 29.75997
## 12 35.890471 30.3292115 41.45173
## 13  7.518324  0.5190994 14.51755
## 14 18.758592 14.2678191 23.24937
## 15 29.213313 24.8682851 33.55834
## 16 45.594300 37.4504396 53.73816
## 17 12.809221  7.1629273 18.45552
## 18 17.360779 12.6487385 22.07282
## 19 25.874734 21.7982892 29.95118
## 20 42.567630 35.2869177 49.84834
## 21 18.799025 14.3140850 23.28397
## 22 20.364345 16.0782626 24.65043
## 23 43.595773 36.0260861 51.16546
## 24 49.123490 39.9338287 58.31315
## 25 11.376751  5.3858618 17.36764
## 26  8.806392  2.1547780 15.45801
## 27 30.426291 25.9182720 34.93431
## 28 12.809221  7.1629273 18.45552
## 29 28.826315 24.5264827 33.12615
## 30 17.863299 13.2350535 22.49154
```

Si queremos encontrar los intervalos de predicción para los valores de X observados:

```
predict(object = mod1, interval = 'prediction')
```

```
## Warning in predict.lm(object = mod1, interval = "prediction"): predictions on current data refer to
```

```
##          fit         lwr      upr
## 1   8.321200 -14.8830218 31.52542
## 2  19.341977  -3.2831326 41.96709
## 3  21.623532  -0.9555722 44.20264
## 4  33.510723  10.7526194 56.26883
## 5  12.202732 -10.7313142 35.13678
## 6  34.117212  11.3312982 56.90313
## 7  40.615312  17.4212824 63.80934
## 8  36.225484  13.3290918 59.12188
## 9  12.451104 -10.4680639 35.37027
## 10 14.322556  -8.4939747 37.13909
## 11 25.689899   3.1285643 48.25123
## 12 35.890471  13.0130594 58.76788
## 13  7.518324 -15.7504858 30.78713
## 14 18.758592  -3.8824192 41.39960
## 15 29.213313   6.6007581 51.82587
## 16 45.594300  21.9559662 69.23263
## 17 12.809221 -10.0890102 35.70745
## 18 17.360779  -5.3251570 40.04672
## 19 25.874734   3.3122478 48.43722
## 20 42.567630  19.2126060 65.92265
## 21 18.799025  -3.8408303 41.43888
## 22 20.364345  -2.2369573 42.96565
## 23 43.595773  20.1490566 67.04249
## 24 49.123490  25.1047841 73.14220
## 25 11.376751 -11.6088774 34.36238
## 26  8.806392 -14.3602293 31.97301
## 27 30.426291   7.7818527 53.07073
## 28 12.809221 -10.0890102 35.70745
## 29 28.826315   6.2224010 51.43023
## 30 17.863299  -4.8053805 40.53198
```

Este último intervalo lo vamos a guardar para poder graficarlo:

```
newData_pred = predict(object = mod1, interval = 'prediction')
```

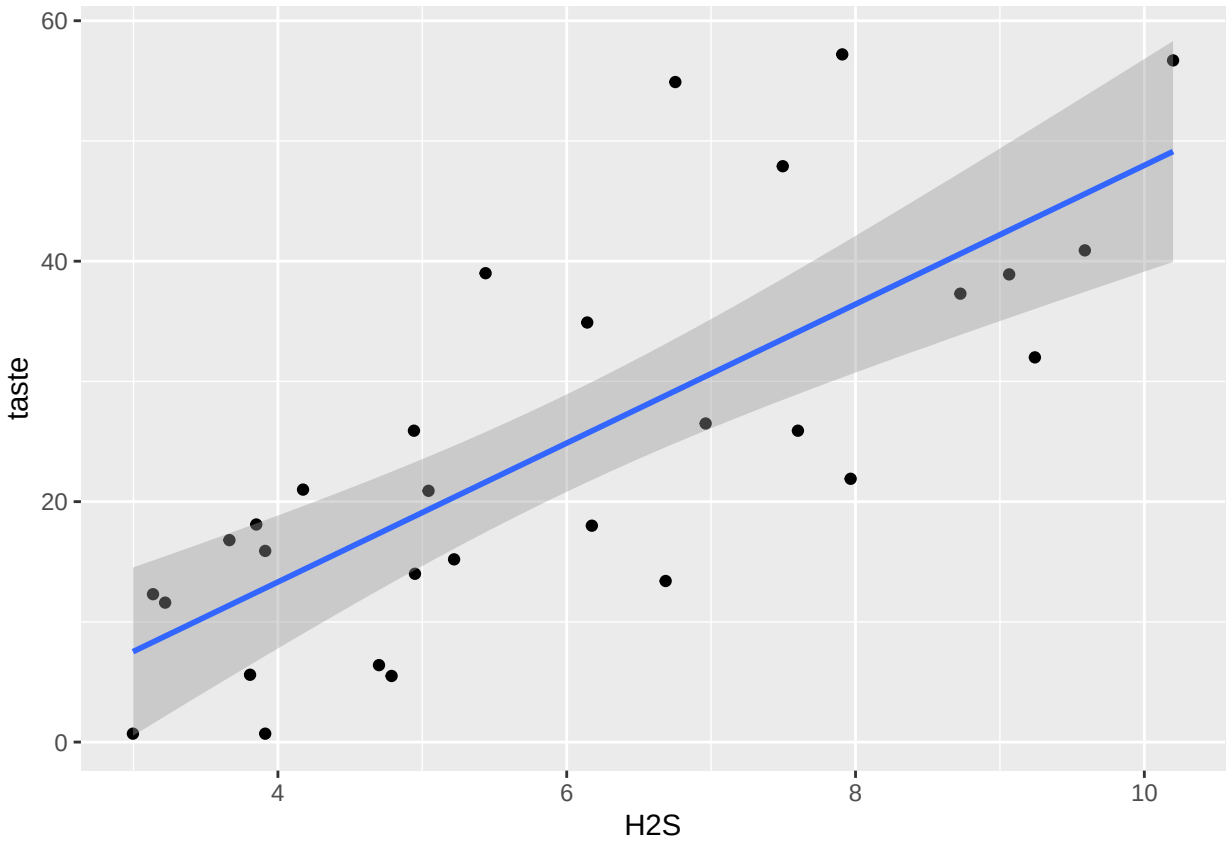
```
## Warning in predict.lm(object = mod1, interval = "prediction"): predictions on current data refer to
```

```
plotData = cheddar
plotData = cbind(plotData, newData_pred)
```

Hacemos una gráfica sin los intervalos de predicción:

```
ggplot(cheddar, aes(x = H2S, y = taste)) +
  geom_point() +
  geom_smooth(method='lm')
```

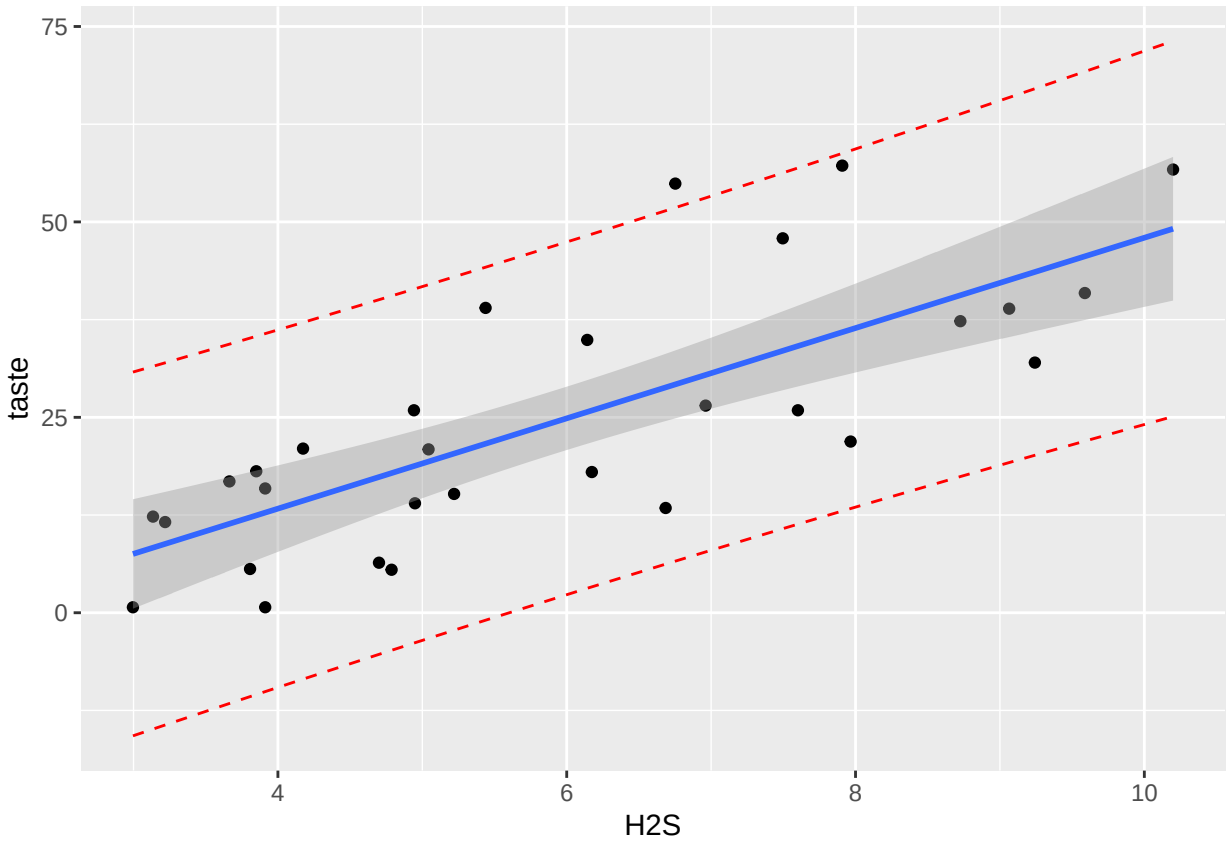
```
## `geom_smooth()` using formula = 'y ~ x'
```



Ahora incluimos la línea de predicción:

```
ggplot(plotData, aes(x = H2S, y = taste )) +
  geom_point() +
  geom_smooth(method="lm", se=TRUE) +
  geom_line(aes(y=lwr), color = "red", linetype = "dashed") +
  geom_line(aes(y=upr), color = "red", linetype = "dashed")
```

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



Podemos predecir también para valores nuevos de X:

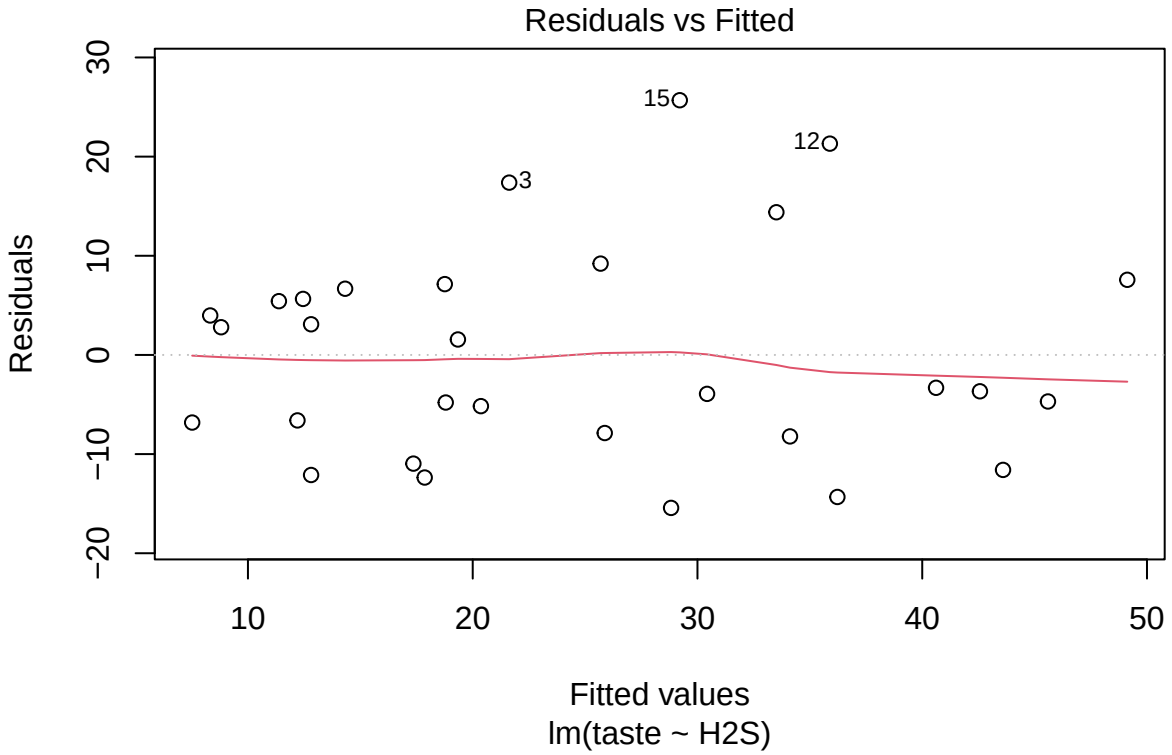
```
newData = data.frame(H2S = seq(from = 2.8, to = 10.2, by = 0.1))
predict(object = mod1, newdata = newData, interval = 'prediction')
```

```
##          fit           lwr      upr
## 1   6.386211 -16.978644207 29.75107
## 2   6.963820 -16.351309398 30.27895
## 3   7.541428 -15.725481749 30.80834
## 4   8.119037 -15.101170651 31.33925
## 5   8.696646 -14.478385271 31.87168
## 6   9.274255 -13.857134550 32.40564
## 7   9.851864 -13.237427188 32.94115
## 8  10.429473 -12.619271644 33.47822
## 9  11.007082 -12.002676120 34.01684
## 10 11.584690 -11.387648560 34.55703
## 11 12.162299 -10.774196636 35.09880
## 12 12.739908 -10.162327748 35.64214
## 13 13.317517  -9.552049009 36.18708
## 14 13.895126  -8.943367243 36.73362
## 15 14.472735  -8.336288978 37.28176
## 16 15.050344  -7.730820436 37.83151
## 17 15.627952  -7.126967529 38.38287
## 18 16.205561  -6.524735854 38.93586
## 19 16.783170  -5.924130684 39.49047
## 20 17.360779  -5.325156966 40.04672
## 21 17.938388  -4.727819314 40.60460
```

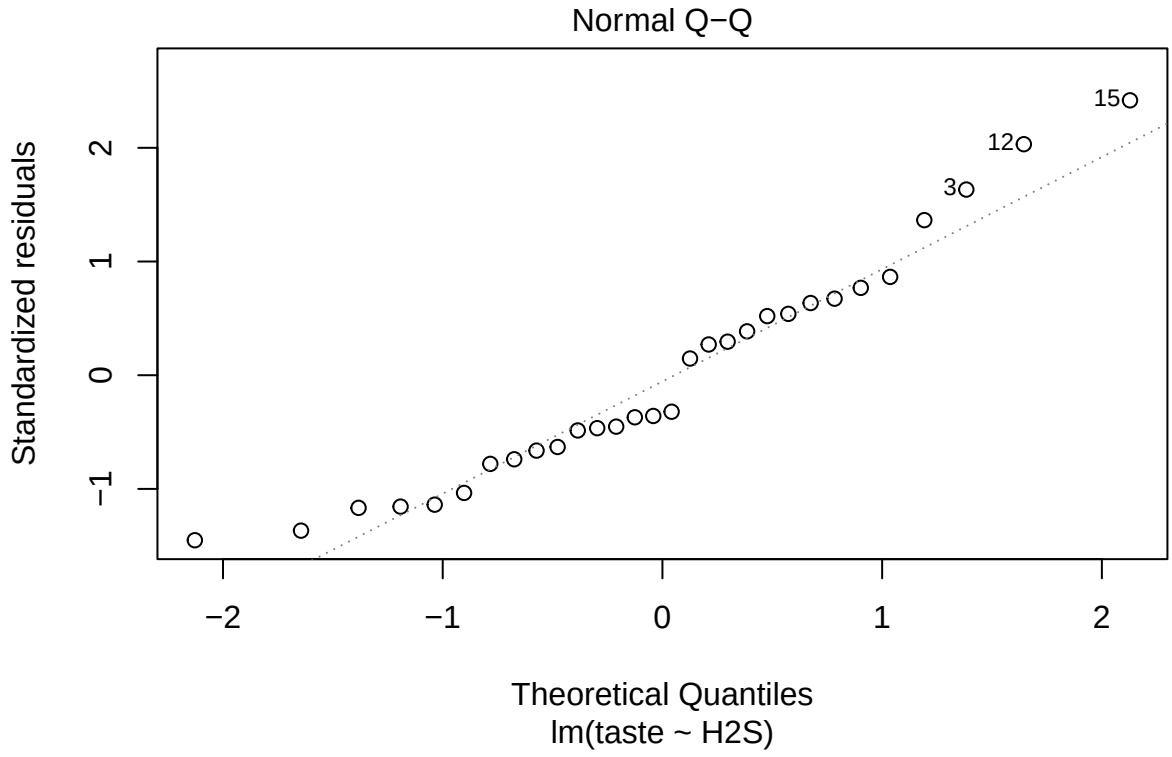
```
## 22 18.515997  -4.132122002 41.16412
## 23 19.093606  -3.538068965 41.72528
## 24 19.671214  -2.945663788 42.28809
## 25 20.248823  -2.354909707 42.85256
## 26 20.826432  -1.765809605 43.41867
## 27 21.404041  -1.178366007 43.98645
## 28 21.981650  -0.592581077 44.55588
## 29 22.559259  -0.008456618 45.12697
## 30 23.136868   0.574005931 45.69973
## 31 23.714477   1.154805497 46.27415
## 32 24.292085   1.733941376 46.85023
## 33 24.869694   2.311413228 47.42798
## 34 25.447303   2.887221085 48.00739
## 35 26.024912   3.461365344 48.58846
## 36 26.602521   4.033846772 49.17119
## 37 27.180130   4.604666503 49.75559
## 38 27.757739   5.173826034 50.34165
## 39 28.335347   5.741327228 50.92937
## 40 28.912956   6.307172309 51.51874
## 41 29.490565   6.871363862 52.10977
## 42 30.068174   7.433904827 52.70244
## 43 30.645783   7.994798499 53.29677
## 44 31.223392   8.554048520 53.89273
## 45 31.801001   9.111658881 54.49034
## 46 32.378609   9.667633913 55.08958
## 47 32.956218  10.221978285 55.69046
## 48 33.533827  10.774696998 56.29296
## 49 34.111436  11.325795378 56.89708
## 50 34.689045  11.875279075 57.50281
## 51 35.266654  12.423154052 58.11015
## 52 35.844263  12.969426584 58.71910
## 53 36.421871  13.514103245 59.32964
## 54 36.999480  14.057190908 59.94177
## 55 37.577089  14.598696736 60.55548
## 56 38.154698  15.138628171 61.17077
## 57 38.732307  15.676992932 61.78762
## 58 39.309916  16.213799006 62.40603
## 59 39.887525  16.749054639 63.02599
## 60 40.465133  17.282768327 63.64750
## 61 41.042742  17.814948813 64.27054
## 62 41.620351  18.345605075 64.89510
## 63 42.197960  18.874746318 65.52117
## 64 42.775569  19.402381966 66.14876
## 65 43.353178  19.928521657 66.77783
## 66 43.930787  20.453175227 67.40840
## 67 44.508395  20.976352710 68.04044
## 68 45.086004  21.498064326 68.67394
## 69 45.663613  22.018320468 69.30891
## 70 46.241222  22.537131704 69.94531
## 71 46.818831  23.054508756 70.58315
## 72 47.396440  23.570462502 71.22242
## 73 47.974049  24.085003963 71.86309
## 74 48.551658  24.598144293 72.50517
## 75 49.129266  25.109894775 73.14864
```

Ahora podemos revisar los supuestos del modelo gráficamente:

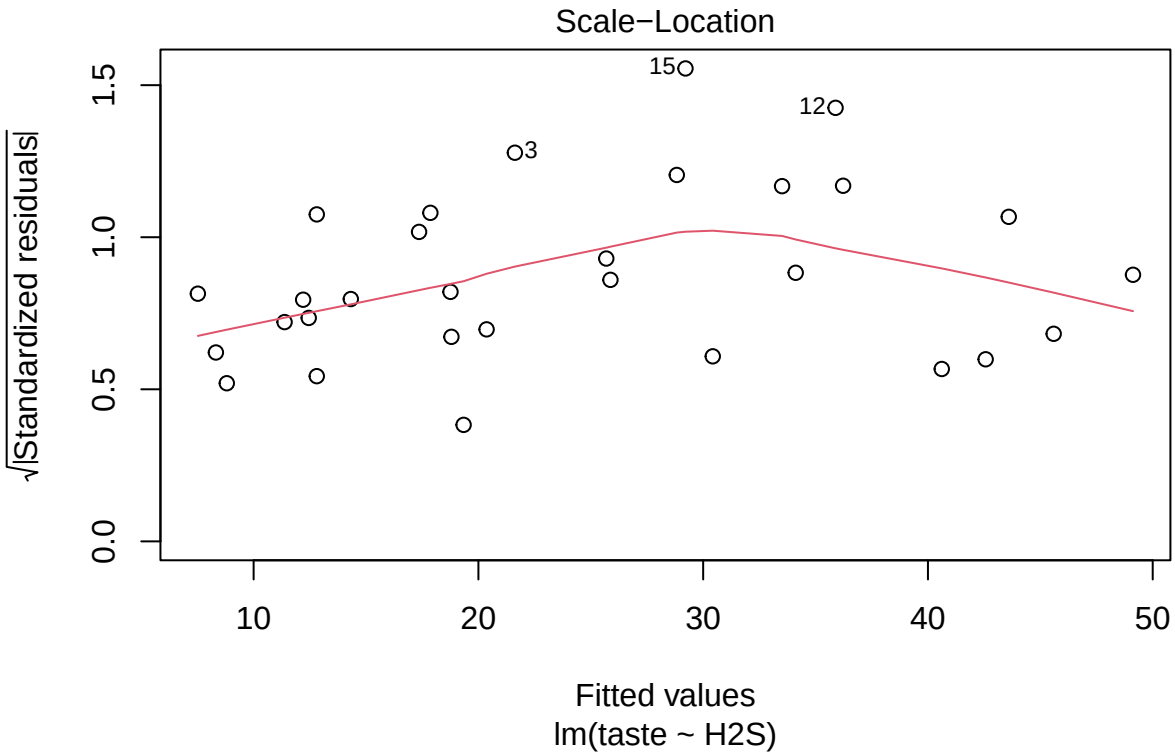
```
plot(mod1, which = 1) # varianza constante
```



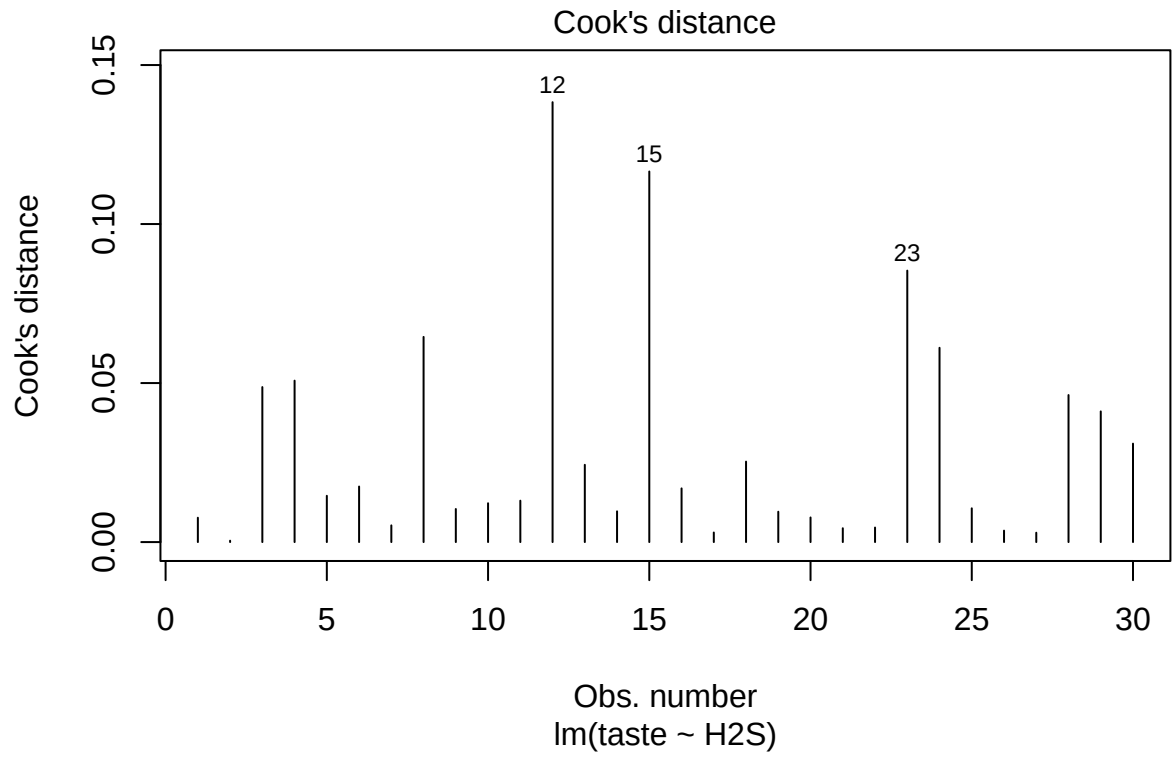

```
plot(mod1, which = 2) # distribucion normal de residuos
```



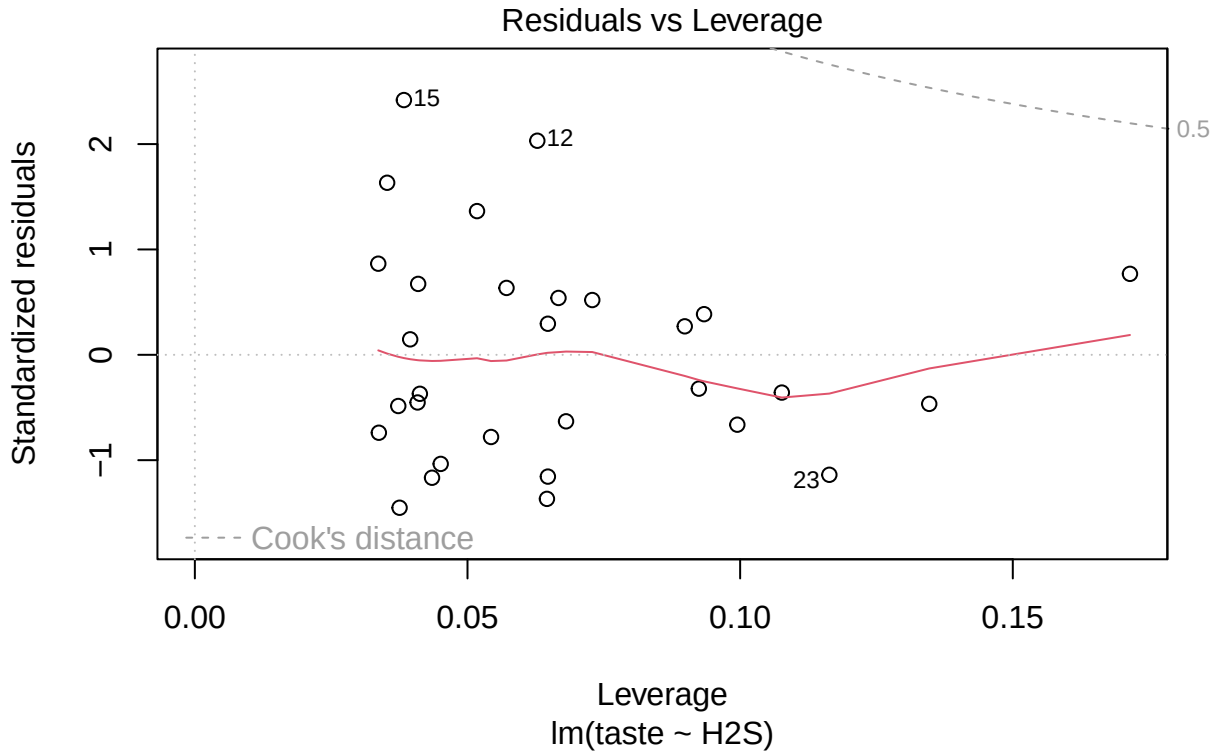
```r
plot(mod1, which = 3) # outliers
```




```r
plot(mod1, which = 4) # puntos influyentes
```



```r
plot(mod1, which = 5) # leverage
```



Modelos de regresión múltiple

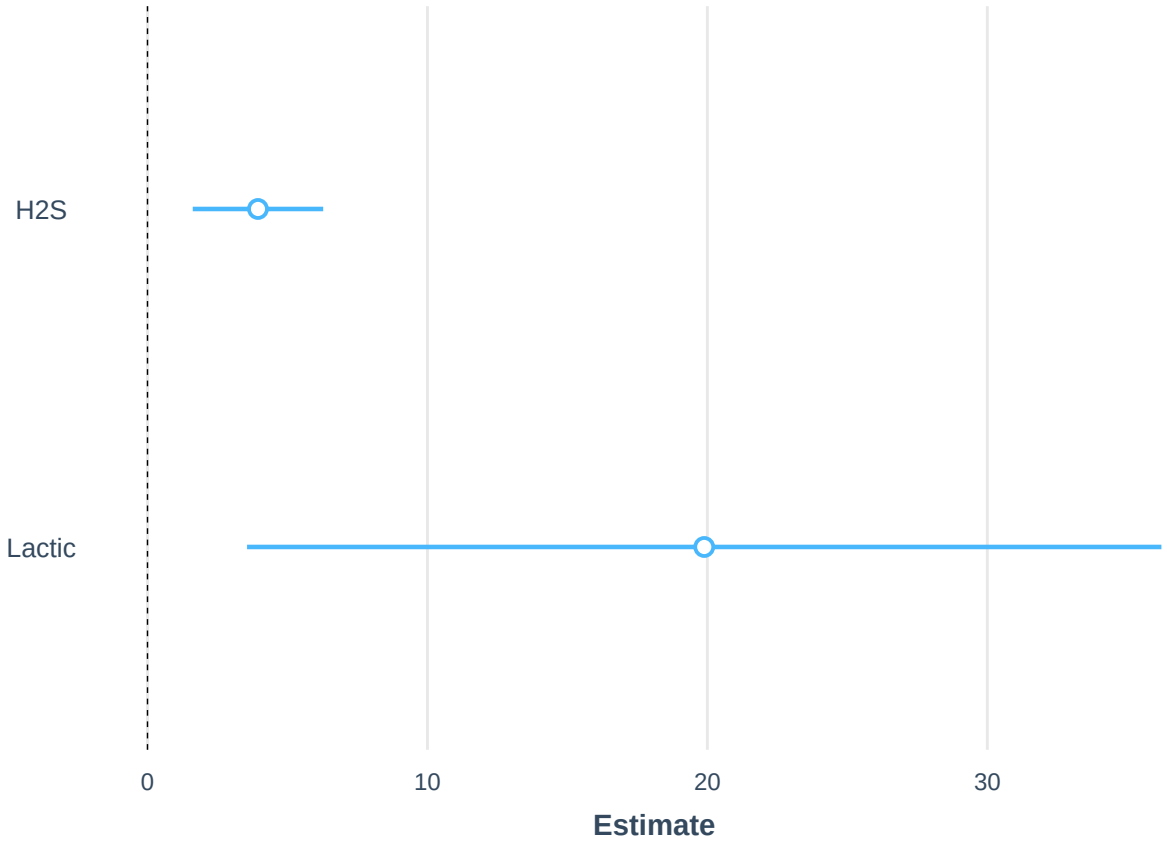
Podemos incluir una variable más al modelo anterior:

```r
mod2 = lm(taste ~ H2S + Lactic, data=cheddar)
summary(mod2)
```

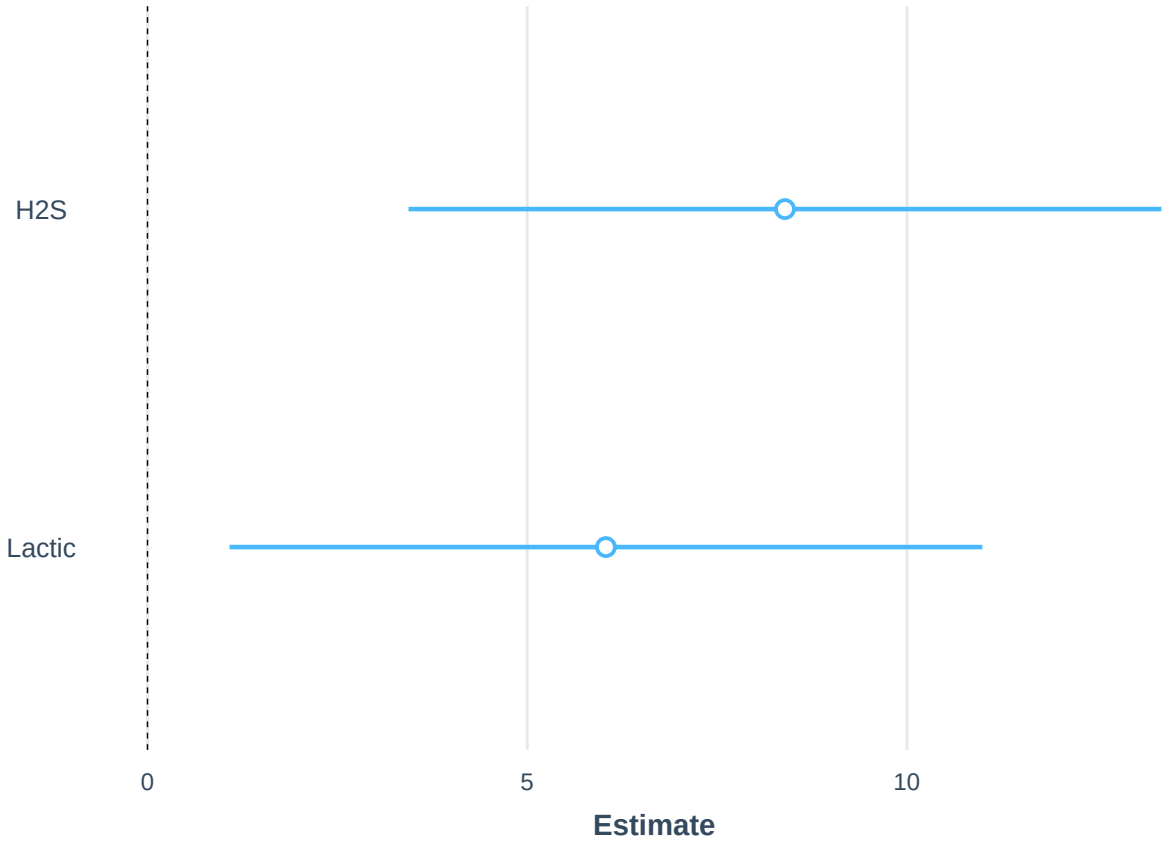
```
##
## Call:
## lm(formula = taste ~ H2S + Lactic, data = cheddar)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.343  -6.530  -1.164   4.844  25.618
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -27.592      8.982  -3.072  0.00481 **
## H2S            3.946      1.136   3.475  0.00174 **
## Lactic        19.887      7.959   2.499  0.01885 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.942 on 27 degrees of freedom
## Multiple R-squared:  0.6517, Adjusted R-squared:  0.6259
## F-statistic: 25.26 on 2 and 27 DF,  p-value: 6.551e-07
```

Podemos hacer gráficas para observar los parámetros estimados y su error asociado:

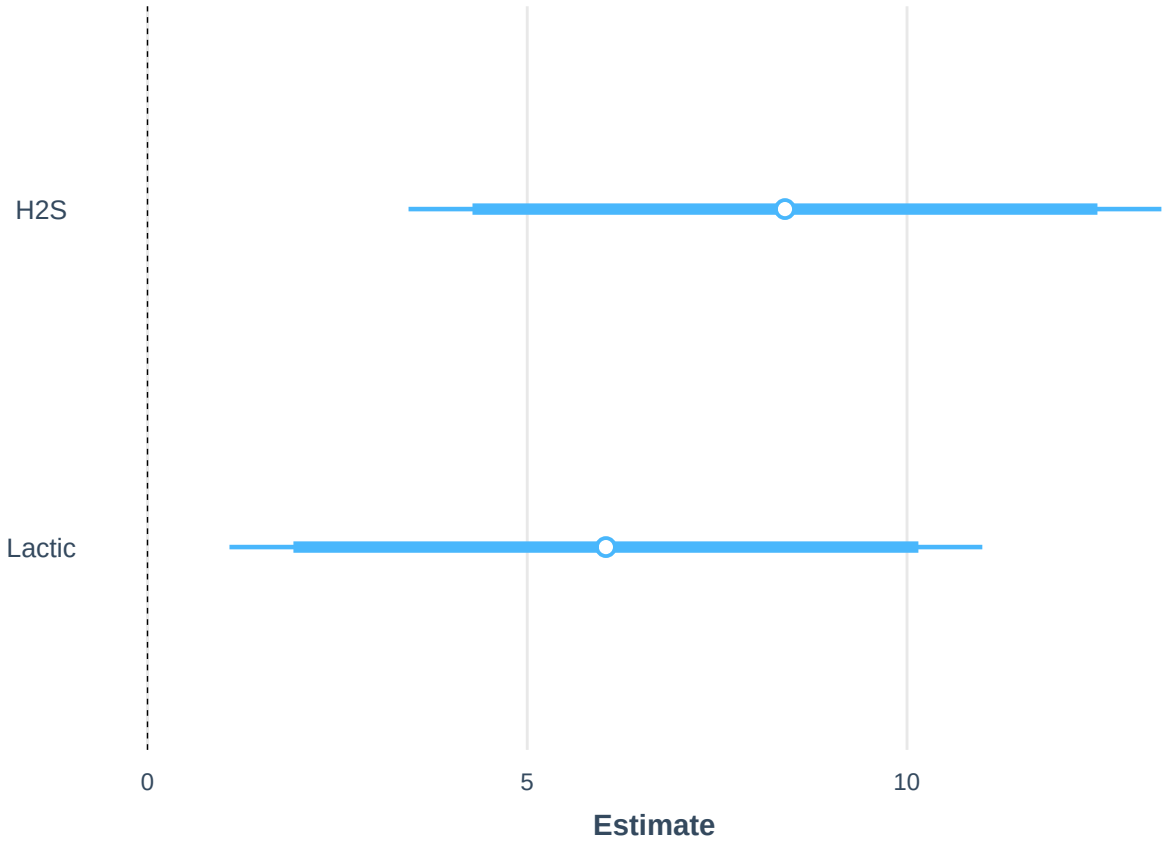
```
plot_summs(mod2)
```



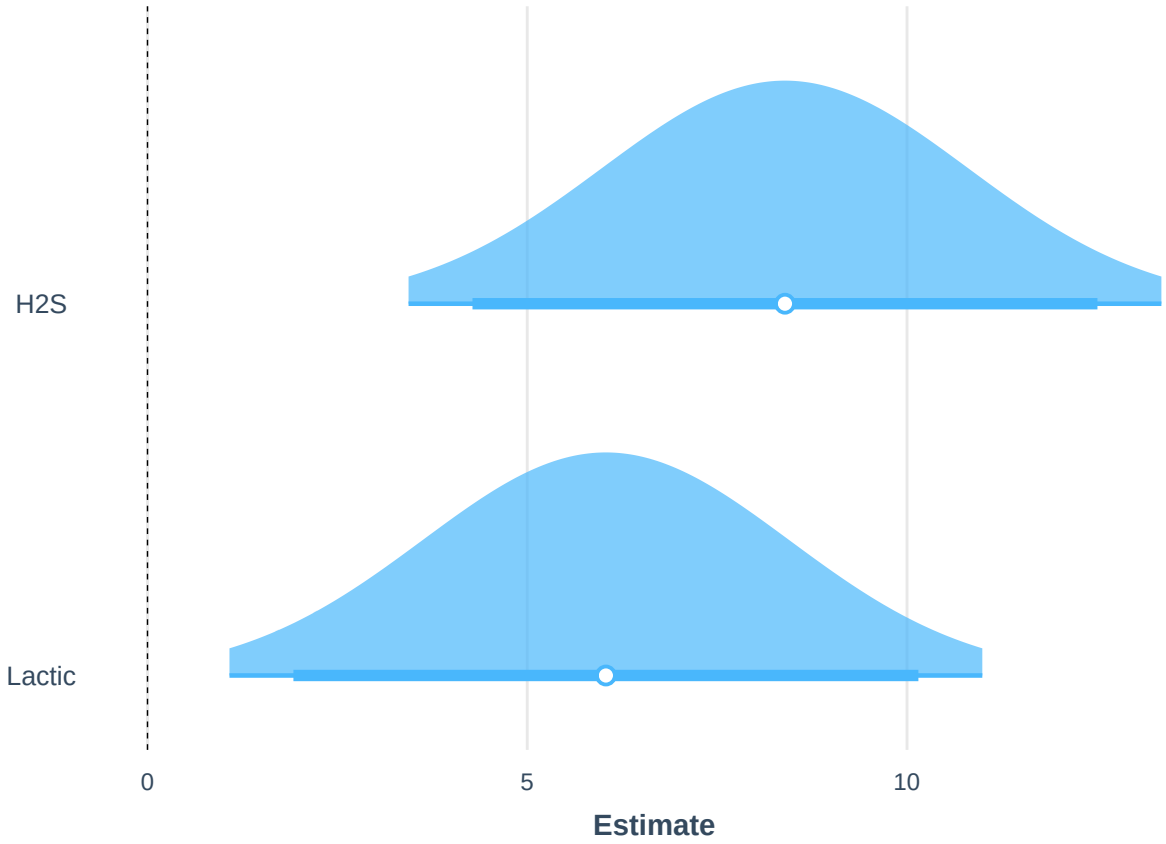

```
plot_summs(mod2, scale = TRUE)
```




```
plot_summs(mod2, scale = TRUE, inner_ci_level = .9)
```

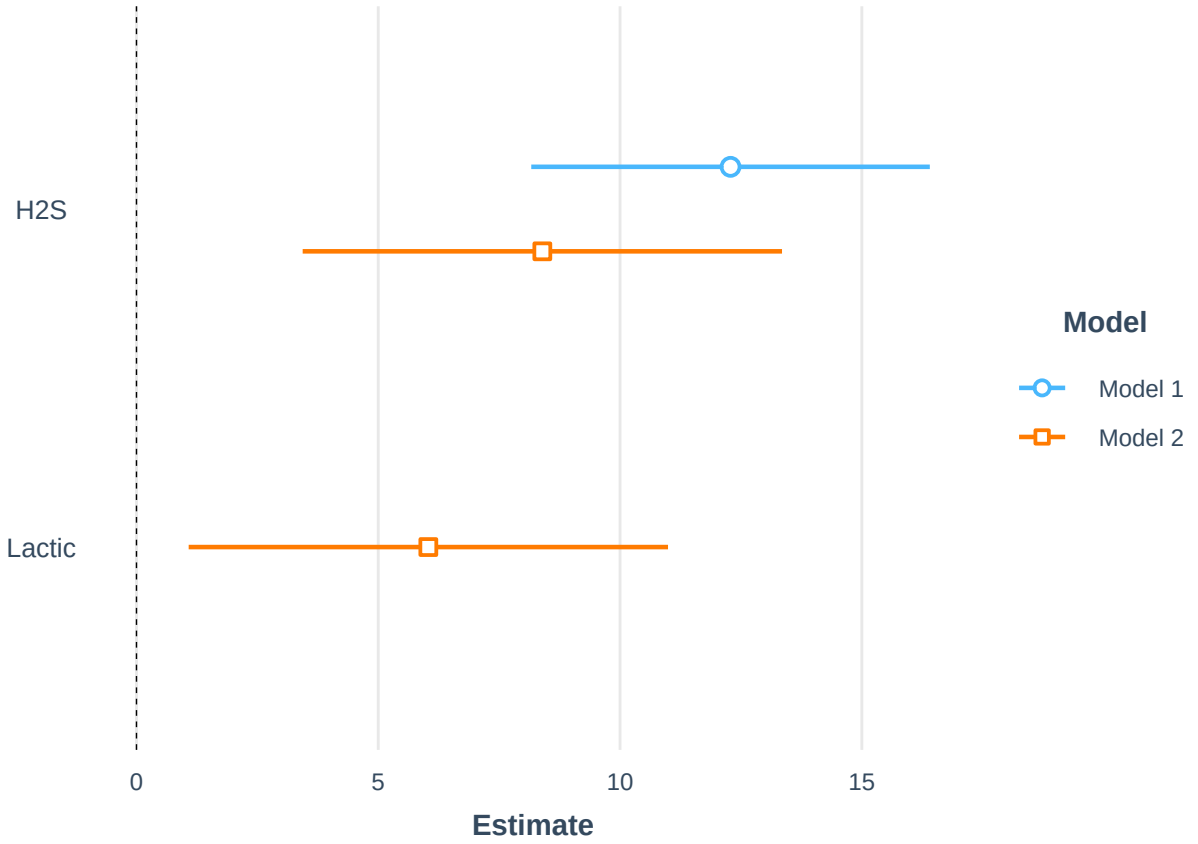



```
plot_summs(mod2, scale = TRUE, plot.distributions = TRUE, inner_ci_level = .9)
```

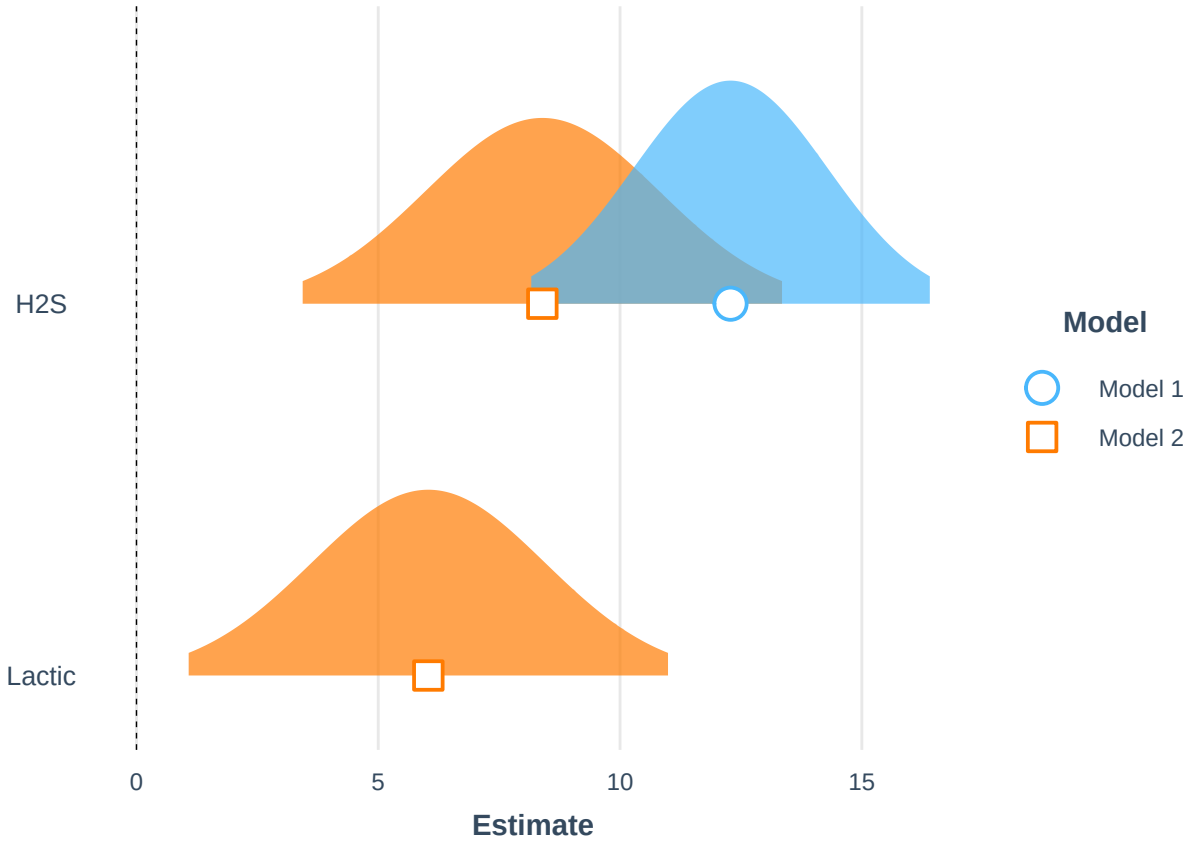


También podemos comparar los estimados de dos modelos diferentes:

```
plot_summs(mod1, mod2, scale = TRUE)
```




```
plot_summs(mod1, mod2, scale = TRUE, plot.distributions = TRUE)
```


H2S
Lactic
0
5
10
15
Estimate
Model
Model 1
Model 2