Sesion 2 Analisis anomalías

Jose Fernando Zea y Fernando López-Torrijos

8 de abril de 2021

Prueba de Grubbs

Una primera prueba para detección de datos extremos anterior a los cinco de Tukey fue el test de Grubbs, denominada así por Frank E. Grubbs, quien la publicó en 1950. Se conoce también como prueba residual máxima normalizada o prueba de desviación extrema studentizada. Se utiliza para detectar valores atípicos en un conjunto de datos univariados que se supone que proviene de una población distribuida normalmente.

Obsérvese el supuesto. Es importante.

La prueba de Grubbs detecta un valor atípico a la vez. Este valor atípico se elimina del conjunto de datos y la prueba se repite hasta que no se detectan valores atípicos adicionales. Sin embargo, las probabilidades de detección cambian en cada iteración. La prueba no debe usarse para tamaños de muestra de seis datos o menos, ya que con frecuencia etiqueta la mayoría de los puntos como valores atípicos.

La hipótesis nula H_0 es que no hay datos atípicos. Y la estadística de prueba es:

$$G = \frac{max_{1,\ldots,n} \mid y_i - \bar{y} \mid}{s}$$

siendo $\bar{y}$ la media y s la desviación estándar.

Mide la máxima desviación y la divide entre la desviación estándar. El resultado lo compara con respecto a un valor de referencia sacado a partir de la distribución t de student. De ahí el origen de su nombre alternativo.

La hipótesis se rechaza si $G > \frac{n-1}{n}\sqrt{\frac{t^2_{\alpha/(2n),n-2}}{n-2+t^2_{\alpha/(2n),n-2}}}$

donde $t^2_{\alpha/(2n),n-2}$ denota el valor crítico después del cual se debe considerar un valor extremo.

Se puede definir el test con la estadística sólo hacia un lado (test de una cola): $G = \frac{\bar{y}-y_{min}}{s}$ ó $G = \frac{y_{max}-\bar{y}}{s}$

Python no tiene una función para su cálculo. Se escribe la función *grubbs()* correspondiente:

```
# importar modulos
import os
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

```
## C:\Users\Fernando\Documents\R\win-library\4.0\reticulate\python\rpytools\loader.py:24: VisibleDeprec
##     Install tornado itself to use zmq with the tornado IOLoop.
##
##   level=level
```

```
from sklearn import datasets
from scipy.stats import t, zscore
import random
```

```
def grubbs(X, test='two-tailed', alpha=0.05):

    '''
    Ejecuta el test de Grubbs recursivamente hasta que la hipótesis nula sea cierta.

    Parametros
    ----------
    X : ndarray
        El arreglo de numeros (numpy) sobre el cual se desea hallar outliers.
    test : str
        Describe los tipos de outliers que se están buscando. Puede ser 'min'
        (si se buscan los outliers muy pequeños), 'max' (si se buscan los grandes),
        o 'two-tailed' (si se buscan ambos).
    alpha : float
        El nivel de significancia.

    Retorna
    -------
    X : ndarray
        El arreglo original con los outliers removidos.
    outliers : ndarray
        Un arreglo de outliers.
    '''

    Z = zscore(X, ddof=1)  # Z-score
    N = len(X)  # Número de muestras

    # Calcula valores extremos y en valor crítico de la t de student
    if test == 'two-tailed':
        extreme_ix = lambda Z: np.abs(Z).argmax()
        t_crit = lambda N: t.isf(alpha / (2.*N), N-2)
    elif test == 'max':
        extreme_ix = lambda Z: Z.argmax()
        t_crit = lambda N: t.isf(alpha / N, N-2)
    elif test == 'min':
        extreme_ix = lambda Z: Z.argmin()
        t_crit = lambda N: t.isf(alpha / N, N-2)
    else:
        raise ValueError("Test must be 'min', 'max', or 'two-tailed'")

    # Calcula el umbral
    thresh = lambda N: (N - 1.) / np.sqrt(N) * \
        np.sqrt(t_crit(N)**2 / (N - 2 + t_crit(N)**2))

    # Crea un arreglo donde almacena los outliers
    outliers = np.array([])

    # Bucle sobre el arreglo de datos y remueve los outliers
    while abs(Z[extreme_ix(Z)]) > thresh(N):
```

```
        # actualiza los outliers
        outliers = np.r_[outliers, X[extreme_ix(Z)]]
        # remueve los outlier del arreglo
        X = np.delete(X, extreme_ix(Z))
        # recalcula el Z score
        Z = zscore(X, ddof=1)
        N = len(X)

    return X, outliers
```

Sean simulados 20 datos mediante una distribución normal, y se añaden dos datos outlier:

```
np.random.seed(1548)
x1 = np.random.normal(30, 10, 20)
x2 = np.random.randint(50, 100, 2)
x = np.append(x1, x2)
print(x)
```

```
## [25.70407946 41.84081878 40.90397337 34.28815348 24.79200971 25.41526208
##  32.18174438 31.80755495 26.64518827 39.67046269 40.20563416 33.29813472
##  28.31732909 18.26280697 32.78518143 11.63954665 32.69773757 35.02533554
##  31.4466714  13.31562267 52.         68.        ]
```

Llámese a la función.

```
grubbs(x)
```

```
## (array([25.70407946, 41.84081878, 40.90397337, 34.28815348, 24.79200971,
##         25.41526208, 32.18174438, 31.80755495, 26.64518827, 39.67046269,
##         40.20563416, 33.29813472, 28.31732909, 18.26280697, 32.78518143,
##         11.63954665, 32.69773757, 35.02533554, 31.4466714 , 13.31562267,
##         52.        ]), array([68.]))
```

Consideró al número 52 como *normal*, y al 68 como *outlier*.

Se menciona como antecedente histórico de cómo en análisis unidimensional *datos extremos* y *datos anómalos* son equivalentes. Pero también es un ejemplo de una de las formas de determinar anomalías. Si el valor de la estadística G es mayor al umbral especificado, es anómalo. Si no, es *normal*. Se trata de una asignación binaria. Sin ambigüedades. Se verán métodos que asignan una probabilidad, y es deber del analista determinar de una manera razonable el umbral de probabilidad a partir del cual lo clasificará como dato anómalo.

Regresión lineal

El modelo de regresión lineal para muchas variables toma la siguiente forma:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

, donde:

- y es la respuesta
- β_0 es el intercepto
- β_1 es el coeficiente asociado a x_1
- ...
- β_p es el coeficiente asociado a x_p

Los valores de β se conocen como los coeficientes de regresión. Estos valores se estiman en un proceso de ajuste usando la función de perdida cuadrático:

$$\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

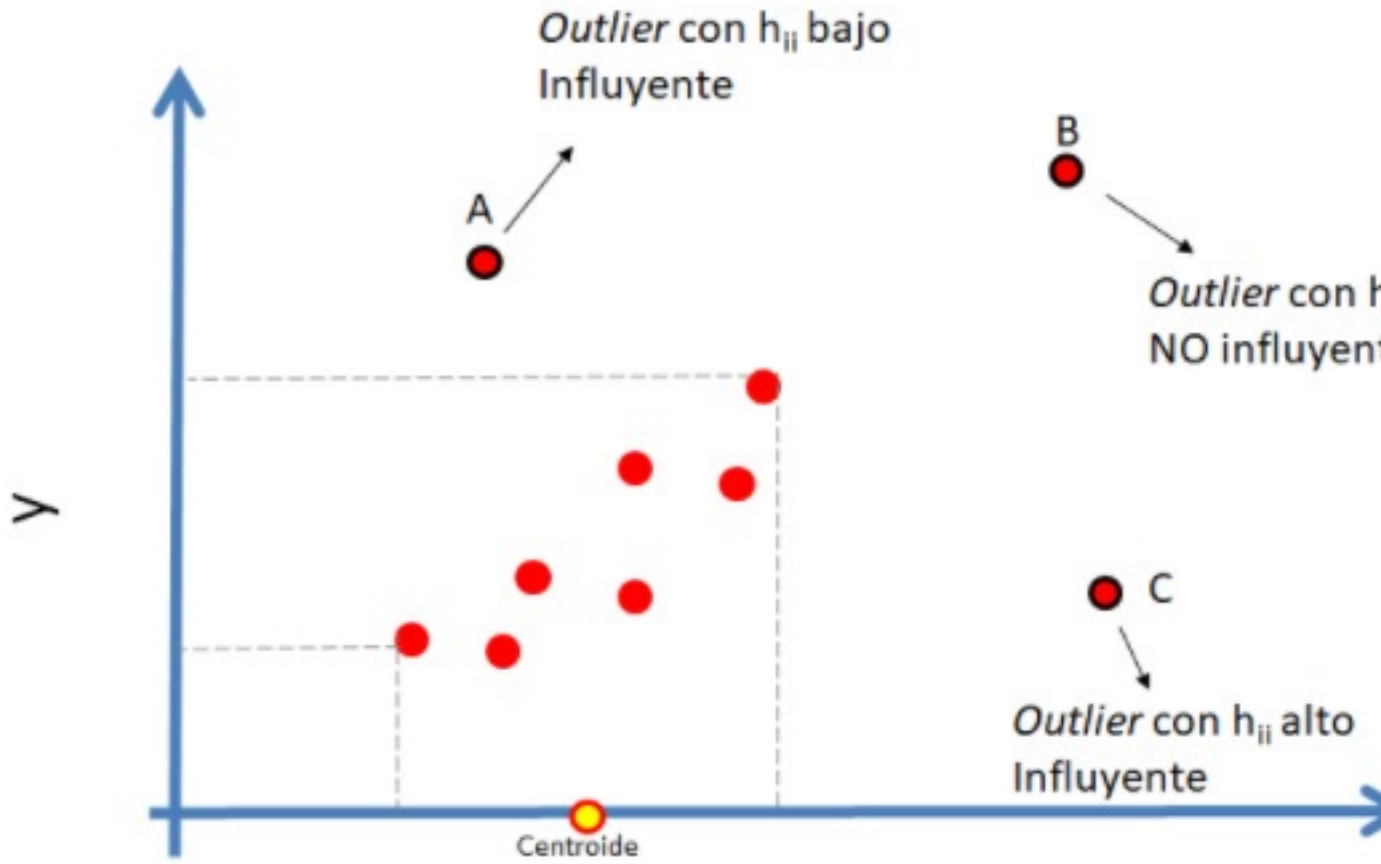
Esta función de perdida se conoce como suma de cuadrados del error.

Se van a exponer varias maneras de utilizar los resultados de una regresión para identificar datos anómalos.

Para exponer las ideas que se van a trabajar, se utiliza una base de datos con medidas corporales.

Fuente: https://fhernanb.github.io/libro_regresion/diag2.html#distancia-de-cook

Medidas de influencia




```
import numpy as np
import pandas as pd
import plotnine
import statsmodels.formula.api as smf
from plotnine import ggplot, geom_point, aes, geom_abline
```

```
datos = pd.read_csv("medidas_cuerpo2.csv")
```

La formulación del modelo y los coeficientes de regresión son:

```
lm=smf.ols(formula = "Peso ~ Estatura", data = datos).fit()
lm.params
```

```
## Intercept   -66.737734
## Estatura     78.540296
## dtype: float64
```

El coeficiente de determinación del modelo es:

```
## 0.43827250709468546
```

Los primeros seis registros:

```
datos.head()
```

```
##    ID  Peso Sexo  Estatura  circun_cuello  circun_muneca
## 0   1  47.6    F      1.57           29.5           13.9
## 1   2  68.1    M      1.66           38.4           16.0
## 2   3  68.0    M      1.90           36.5           16.6
## 3   4  80.0    M      1.76           38.0           17.1
## 4   5  68.1    M      1.83           38.0           17.1
```

Número de datos total:

```
n=datos.shape[0]
print(n)
```

```
## 26
```

Las medidas influyentes son:

```
p=2
influence = lm.get_influence()
print(influence)
```

```
## <statsmodels.stats.outliers_influence.OLSInfluence object at 0x000000006A20A4E0>
```

Residuales estudentizados

Otra manera de identificar outliers es el uso de los residuales estandarizados:

$$t_i = \frac{\hat{\epsilon}_i}{\hat{\sigma}\sqrt{\frac{n-1}{n}}}$$

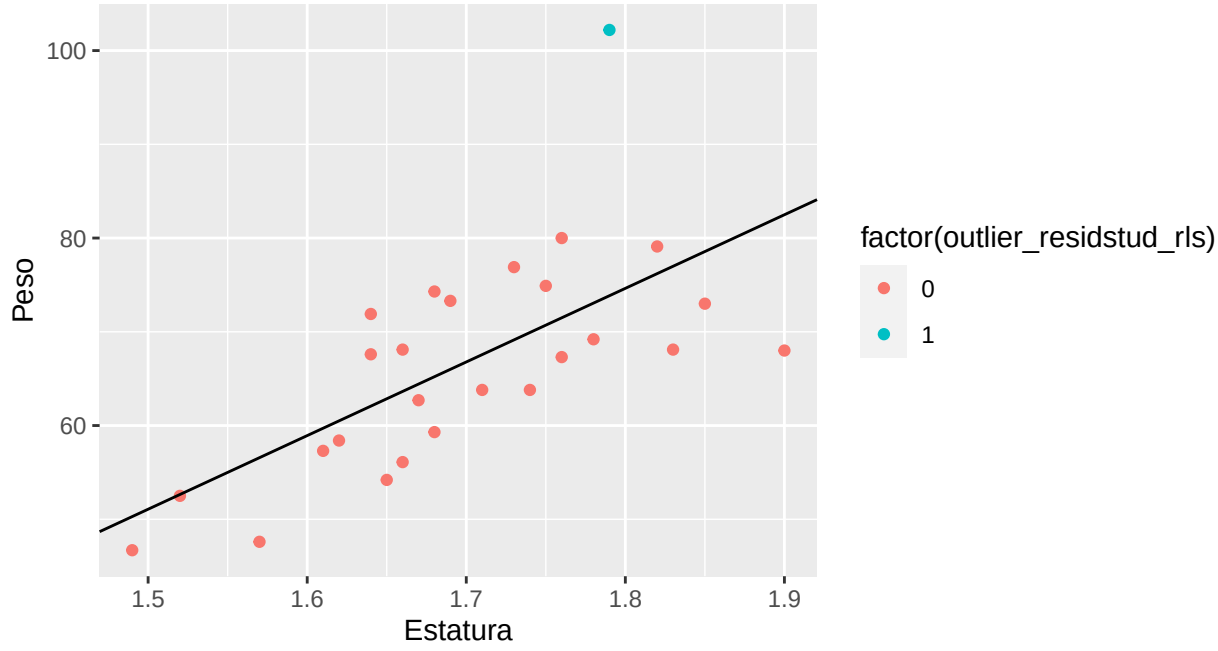
Un umbral razonable para identificar valores extremos según el modelo es considerar los puntos con valor absoluto de los residuales estudentizados mayores a dos:

$$|t_i| > 2$$

```
resid_student = influence.resid_studentized_external
datos[['outlier_residstud_rls']] = np.where(abs(resid_student) >= 2, 1, 0)
```

```
(ggplot(datos, aes('Estatura', 'Peso', color='factor(outlier_residstud_rls)'))
 + geom_point()
 + geom_abline(intercept = lm.params[0], slope = lm.params[1], color = "black"))
```

Se puede visualizar el punto atípico:



Distancia de Cook

Se define como

$$D_i = \frac{1}{p\sigma^2} \sum_{j=1}^{n} \left(\hat{y}_j - \hat{y}_{j(i)}\right)^2$$

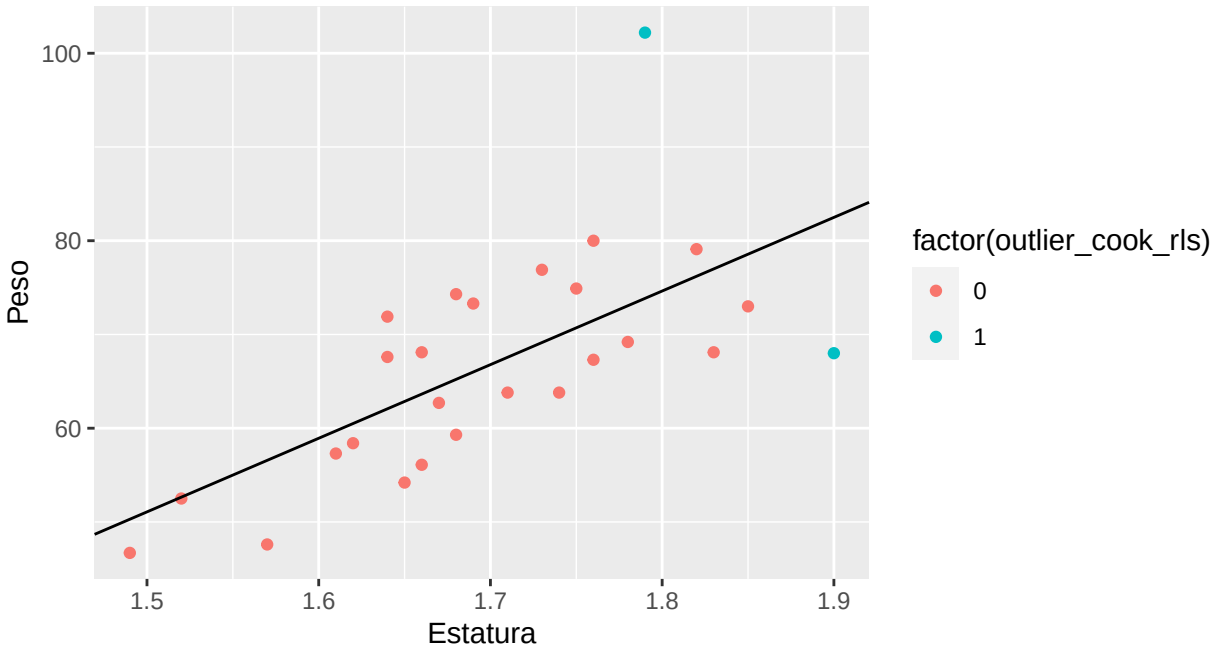
Donde σ corresponde a la varianza de los residuales, $\hat{y}_j$ es el valor pronosticado para la observación j-ésima usando todos los datos en el modelo de regresión y $\hat{y}_{j(i)}$ es el pronóstico para la observación j eliminando la observación i-ésima. p es el número de variables consideradas + 1 (el intercepto también se tiene en cuenta), por ejemplo si se consideran para el modelo de regresión dos variables independientes para pronosticar la variable de interés Y el valor de p = 3.

Un punto se considera influyente si $D_i > \frac{4}{n-k-2}$

```
(cooks, p) = influence.cooks_distance
datos[['outlier_cook_rls']] = np.where(cooks >= 4 /(n-2-2), 1, 0)
```

```
(ggplot(datos, aes('Estatura', 'Peso', color='factor(outlier_cook_rls)'))
 + geom_point()
 + geom_abline(intercept = lm.params[0], slope = lm.params[1], color = "black"))
```

Se puede visualizar el punto atípico que ya quedó también identificado con el método anterior, y un punto adicional:



Estadística dffit

La diferencia "estandarizada" entre el pronóstico de la obsevación i-esima con el modelo haciendo uso de todos los datos y el modelo quitando la observación i-ésima

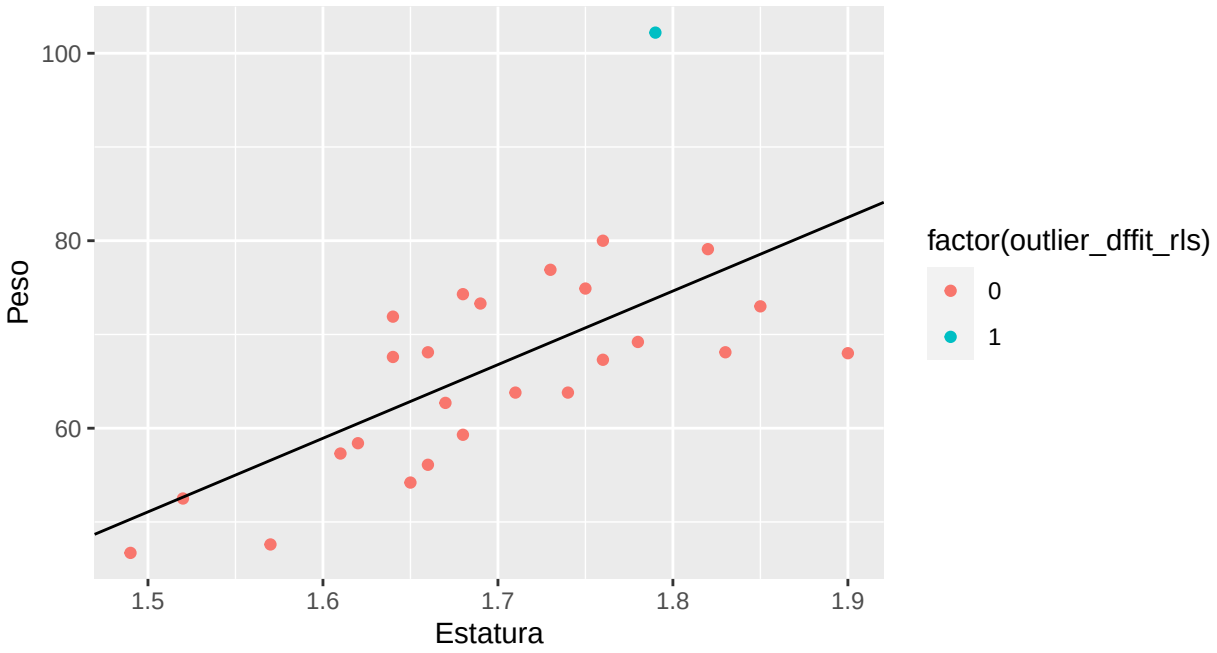
$$DFFIT_i = \frac{\hat{y}_i - \hat{y}_{i(i)}}{\sqrt{s^2_{(i)} h_{ii}}}$$

Si el valor de dffit de la i-ésima es mayor a $2\sqrt{p/n}$ la observación se considera un outlier.

```
(dffits, p) = influence.dffits
datos[['outlier_dffit_rls']] = np.where(dffits > 2 / np.sqrt(p/n), 1, 0)
```

```
(ggplot(datos, aes('Estatura', 'Peso', color='factor(outlier_dffit_rls)'))
 + geom_point()
 + geom_abline(intercept = lm.params[0], slope = lm.params[1], color = "black"))
```

Se pueden visualizar los puntos atípico detectados por la prueba dffit:



Leverege point

Es relevante identificar los valores atípicos en los predictores, para esto se utiliza la matriz de proyección (matriz "hat"):

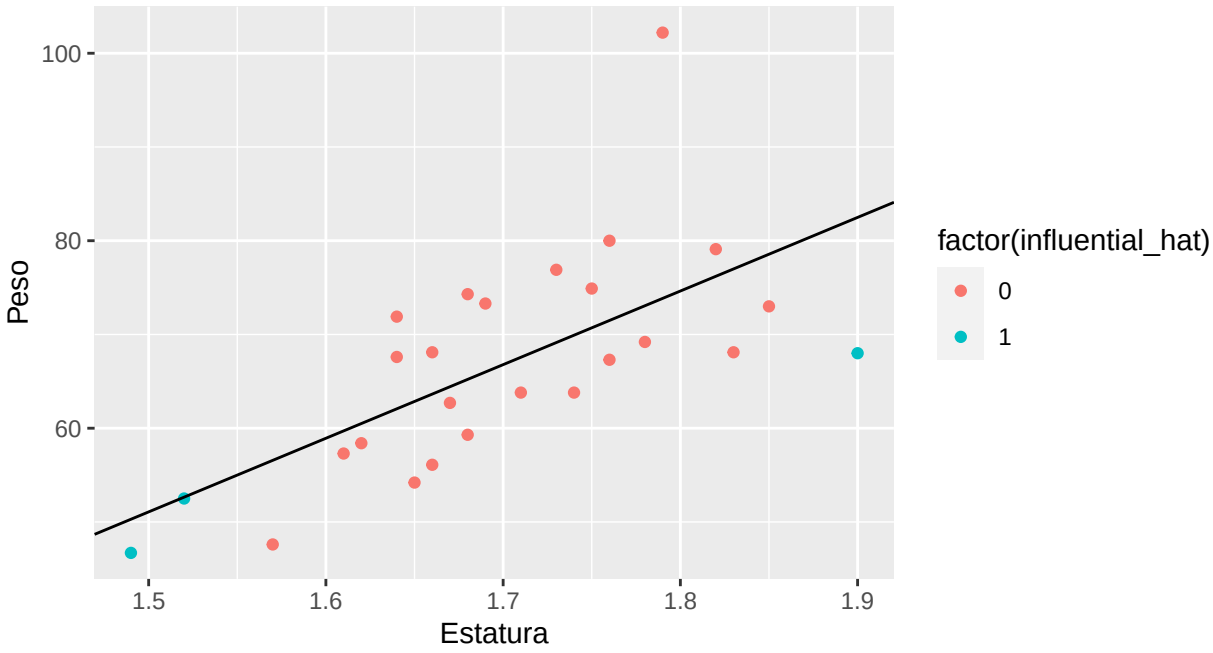
$$H = X(X^T X)^{-1} X^T$$

Identificaremos los valores que son extremos en el espacio de los predictores:

```
leverage = influence.hat_matrix_diag
datos[['influential_hat']] = np.where(leverage >= 2*p/n, 1, 0)
```

```
(ggplot(datos, aes('Estatura', 'Peso', color='factor(influential_hat)'))
 + geom_point()
 + geom_abline(intercept = lm.params[0], slope = lm.params[1], color = "black"))
```

Se puede visualizar los punto atípico detectados mediante la medida de apalancamiento (leverege):




```
datos.head()
```

```
##    ID  Peso Sexo  ...  outlier_cook_rls  outlier_dffit_rls  influential_hat
## 0   1  47.6    F  ...                 0                  0                1
## 1   2  68.1    M  ...                 0                  0                1
## 2   3  68.0    M  ...                 1                  0                1
## 3   4  80.0    M  ...                 0                  0                1
## 4   5  68.1    M  ...                 0                  0                1
##
## [5 rows x 10 columns]
```

Caso multivariado

Una buena estrategia es identificar valores atípicos por varios criteros. En el caso multivariado no es posible visualizar los outliers, sin embargo si se pueden marcar los puntos y analizarlos.

```
datos = pd.read_csv("medidas_cuerpo2.csv")
```

```
lm2 = smf.ols(formula = "Peso~Estatura+circun_cuello+circun_muneca", data = datos).fit()
n=datos.shape[0]
p=4
lm2.params
```

```
## Intercept       -62.018297
## Estatura          9.440395
## circun_cuello     2.258735
## circun_muneca     1.989082
## dtype: float64
```

```
lm2.rsquared
```

```
## 0.7582013458665
```

Calculamos las diferentes medidas de influencia bajo el modelo multivariado:

```
influence_rlm = lm2.get_influence()
resid_student_rlm = influence_rlm .resid_studentized_external
(cooks_rlm, p_cooks_rlm) = influence_rlm.cooks_distance
(dffits_rlm, p_dffits_rlm) = influence_rlm.dffits
leverage_rlm = influence_rlm.hat_matrix_diag
```

```
datos[['outlier_residstud_rlm']] = np.where(abs(resid_student_rlm) >= 2, 1, 0)
datos[['outlier_cook_rlm']] = np.where(cooks_rlm >= 4 /(n-2-2), 1, 0)
datos[['outlier_dffit_rlm']] = np.where(dffits_rlm > 2 * np.sqrt(p/n), 1, 0)
datos[['influential_hat_rlm']] = np.where(leverage_rlm >= 2*p/n, 1, 0)
```

```
df_outliers = datos.query('outlier_residstud_rlm == 1 | outlier_cook_rlm == 1 | outlier_dffit_rlm == 1'
df_outliers
```

```
##     ID   Peso Sexo  ...  outlier_cook_rlm  outlier_dffit_rlm  influential_hat_rlm
## 7    8   69.2    M  ...                 1                  0                    0
## 10  11  102.2    M  ...                 1                  1                    0
##
## [2 rows x 10 columns]
```

```
df_outliers = df_outliers.drop(['outlier_residstud_rlm','outlier_cook_rlm', 'outlier_dffit_rlm', 'influ
df_outliers
```

```
##     ID   Peso Sexo  Estatura  circun_cuello  circun_muneca
## 7    8   69.2    M      1.78           40.5           16.5
## 10  11  102.2    M      1.79           41.5           17.1
```

Ejercicio

Considerar la estatura de las mujeres e identificar outliers