Random sample consensus (RANSAC)

Fernando López-Torrijos

abril de 2021

Introducción

Random sample consensus (RANSAC) es una heurística (algoritmo iterativo) para calcular los parámetros de un modelo de regresión lineal sobre un conjunto de datos observados que contiene valores atípicos. Es no determinista en el sentido de que produce un resultado razonable solo con una cierta probabilidad, mayor a medida que se permita que el algoritmo explore un mayor número de iteraciones. El algoritmo fue publicado por primera vez por Fischler y Bolles SRI International en 1981.

El supuesto es que los datos consisten en "inliers", es decir, datos cuya distribución se explica por un conjunto de parámetros del modelo, aunque, como es usual en detección de anomalías, pueden estar sujetos a ruido, y "valores atípicos".

RANSAC asume que, dado un conjunto de *inliers* (generalmente pequeño), existe un procedimiento que permite estimar los parámetros del modelo.

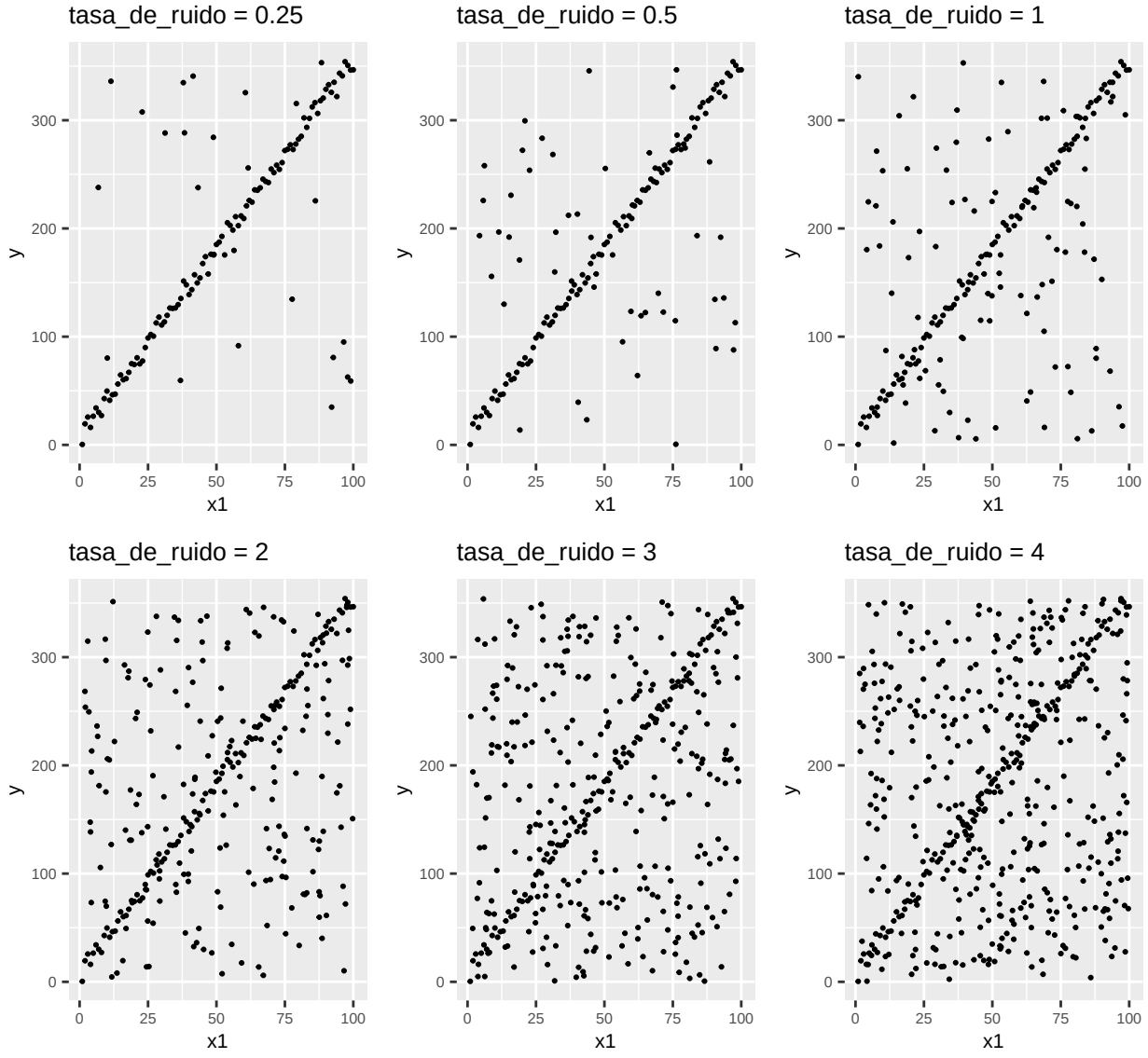
Una ventaja de RANSAC es su capacidad para hacer una estimación robusta de los parámetros del modelo, es decir, estima el modelo a pesar de haber un número significativo de valores atípicos. Una desventaja de RANSAC es que no hay tiempo máximo para calcular los parámetros del modelo, excepto el agotamiento del número de iteraciones. El número de iteraciones puede limitar la solución a una no óptima, y puede encontrar una solución que no se ajuste a los datos.

Ejemplo[1]

Se mostrará un ejemplo extremo, con diverso número de datos atípicos, para ejemplificar las ventajas y desventajas del método. Se hará mediante la incorporación de diversos *niveles de ruido* a los datos.

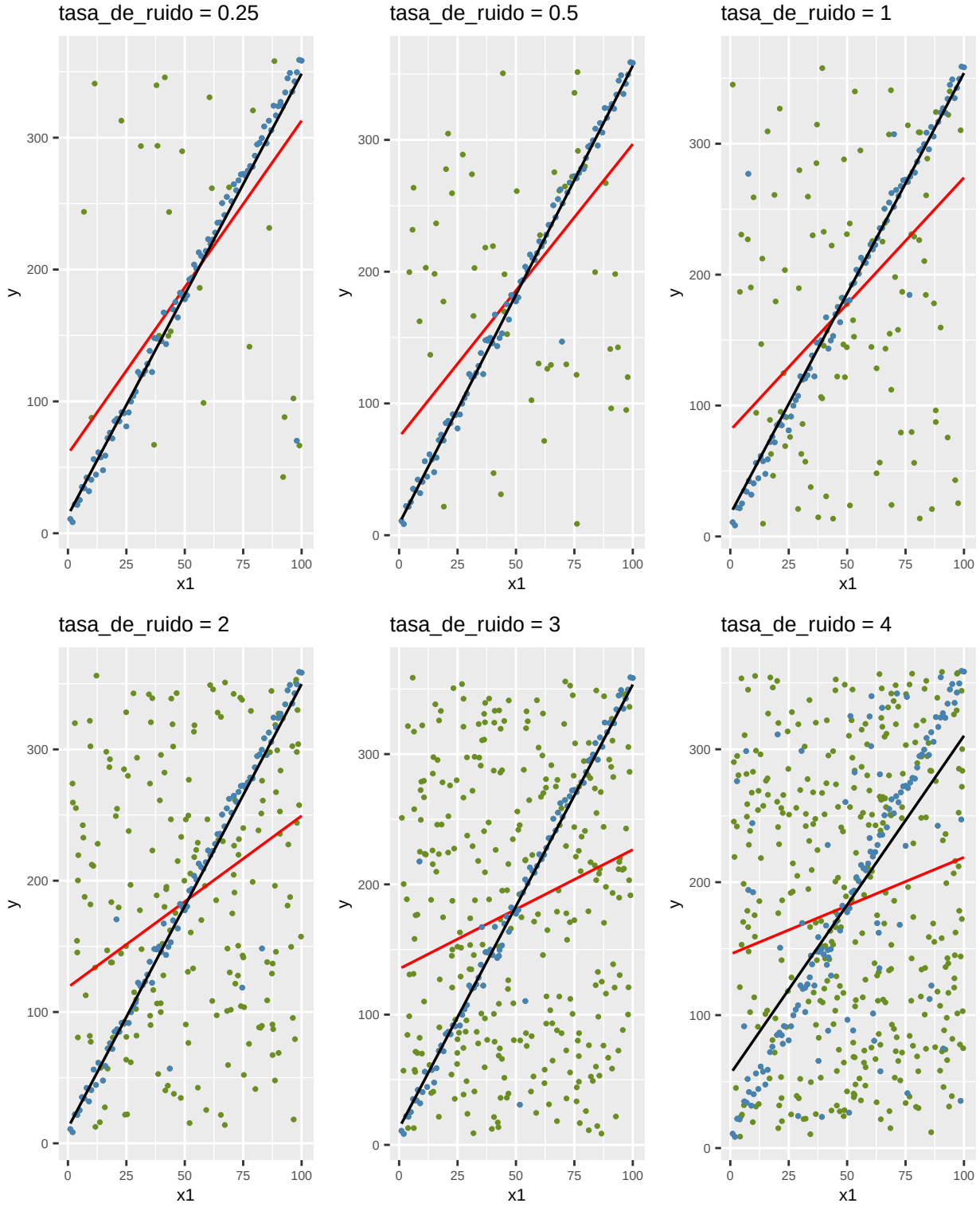
Para el modelamiento, supondremos una relación entre la variable de respuesta y un par de variables explicativas: $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$

El siguiente gráfico presenta los datos de la primera variable, junto con el ruido incorporado. Una tasa de ruido = 0.50 implica que hay un 50% de puntos *outliers* adicionales a los *inliers*. Una tasa de ruido = 4 implica que hay cuatro veces más *outliers* que *inliers*. Este ejemplo tan ruidoso no es el usual en aplicaciones donde el interés es encontrar los *outliers*, pero es demostrativo de lo robusto que es el algoritmo.

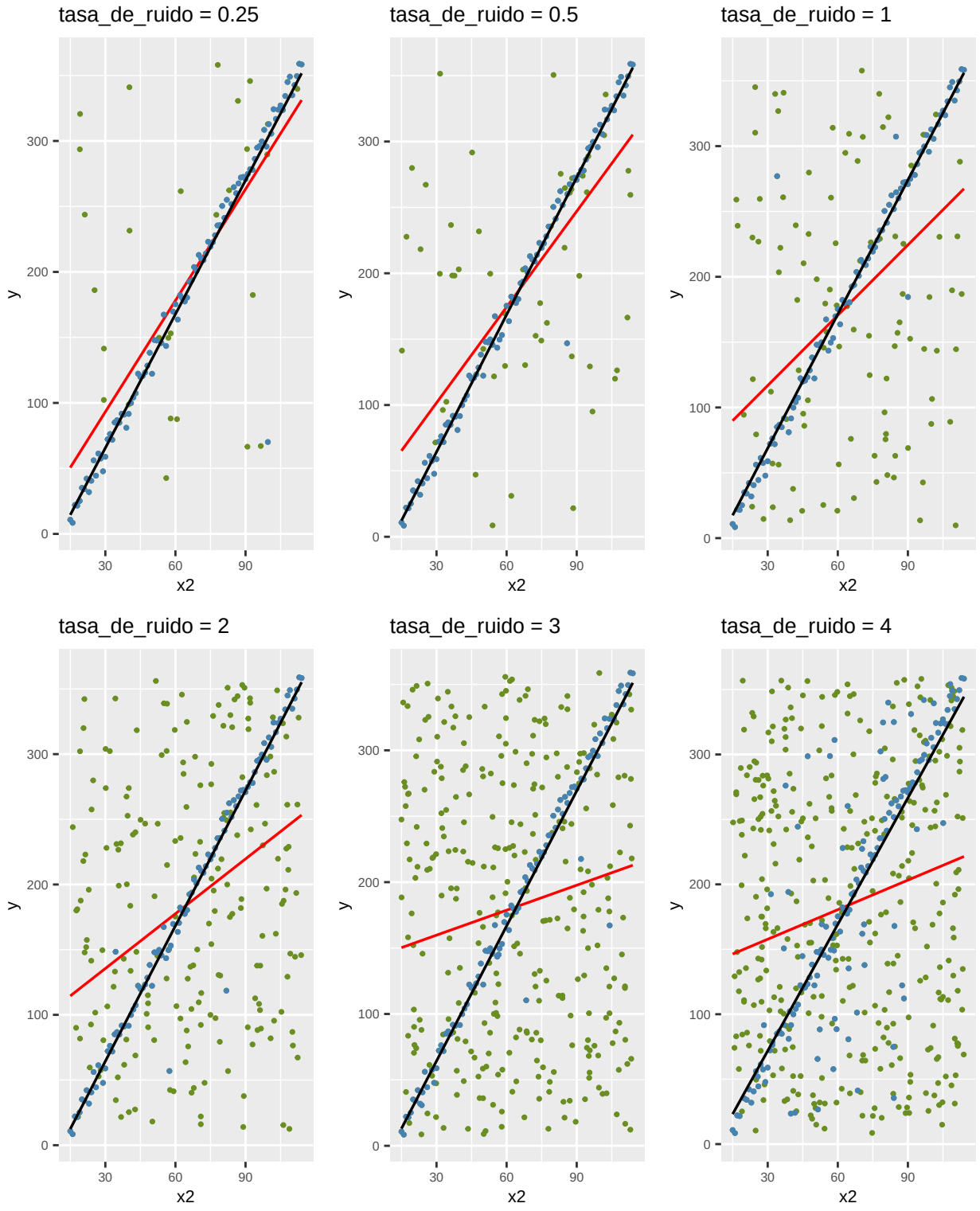


[1] Código tomado de http://www.pjthepooh.com/ransac_in_r.html y ajustado a dos variables independientes.

Este es el resultado sobre la primera variable. Obsérvese que los *outliers* están resaltados en color verde oliva. Tuvo equivocaciones sobre la tasa de ruido más alta.



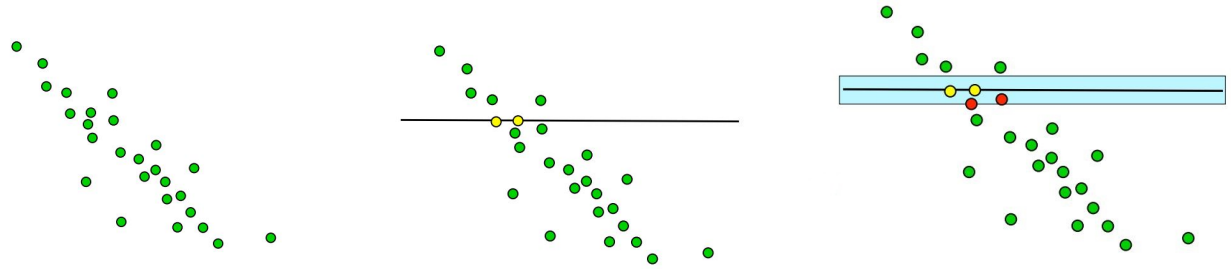
Sobre la segunda variable funcionó bastante bien.



Las equivocaciones pueden tener que ver con la parametrización del modelo. Hay que entender cómo funciona.

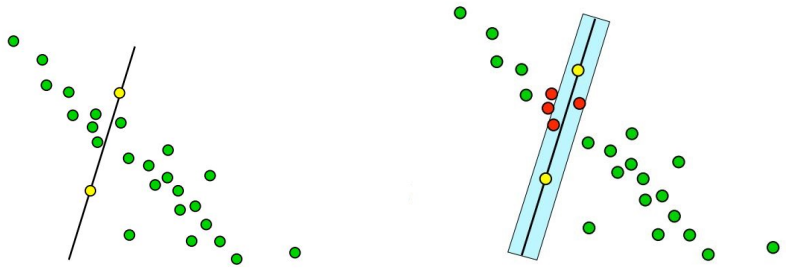
El algoritmo[2]

Una vez se tiene la información selecciona n puntos al azar y establece la regresión lineal. En este esquema de ejemplo n = 2. Cuenta cuántos puntos están dentro del umbral cercano a la regresión lineal.



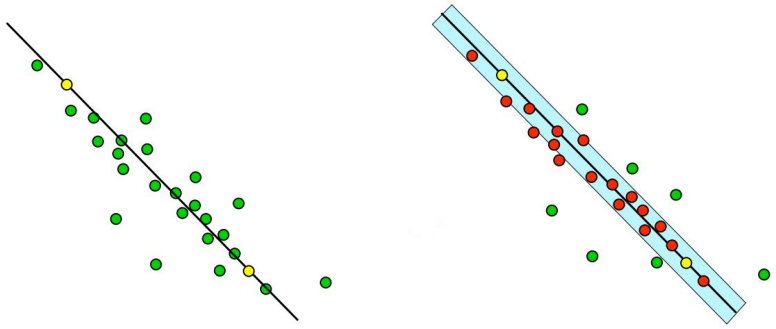
Hay 4 puntos.

Procede a seleccionar otro subconjunto de datos de tamaño n y a establecer otra regresión lineal. Y cuenta cuántos puntos están cercanos al umbral en la nueva regresión.



Hay 6 puntos.

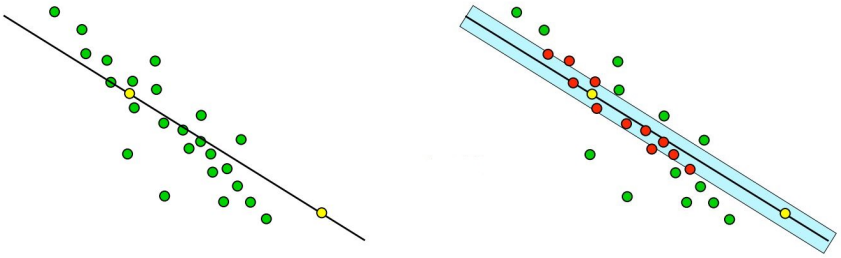
Lo realiza iterativamente.



Hay 19 puntos.

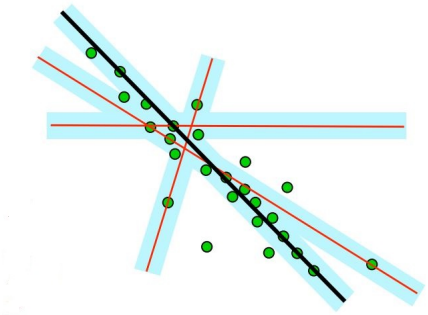
El número de veces que se le haya especificado.

[2]Gráficas y explicación del algoritmo tomadas del curso sobre procesamiento de imágenes de Roberts Collins, Penn State University



Hay 13 puntos.

Evalúa las diferentes opciones y selecciona aquellas que hayan llegado al número de puntos *inliers* esperado, y de éstas selecciona la mejor.



El mejor modelo es el de 19 puntos.

Dados el conjunto de datos y el modelo de regresión lineal, los parámetros del algoritmo son:

```
*n* - Un número mínimo de puntos con los cuales estimar la regresión.
*k* - El número máximo de iteraciones.
*t* - Un umbral que permita determinar que el modelo ajusta bien.
*d* - El número de puntos cercanos requeridos para asegurar que el modelo ajusta bien.
```

El algoritmo es el siguiente:

```
iteracion ← 0;
Mejor_ajuste ← inicia vacío;
Mejor_error ← inicia muy grande;
while iteracion < k do
    pueden_ser_inliers ← selección aleatoria de datos, de tamaño n;
    modelo_posible ← ajusta modelo y selecciona datos que pueden ser inliers;
    inliers_adicionales ← inicia vacío;
    for cada punto que no está en el conjunto pueden_ser_inliers do
        if el punto < umbral t then
            añada el punto al conjunto inliers_adicionales;
        end
    end
    if el número de elementos en inliers_adicionales > d then
        Ha encontrado el modelo;
        Mida qué tan bueno es;
        mejor_modelo_actual ←
          modelo ajustado con los puntos de los conjuntos pueden_ser_inliers e inliers_adicionales;
        este_error ← una medida de cuan bueno es el mejor_modelo_actual;
        if este_error < al Mejor_error then
            Mejor_ajuste ← mejor_modelo_actual;
            Mejor_error ← este_error;
        end
    end
    incremente iteracion;
end
```

Algorithm 1: Algoritmo de Ransac en pseudo código

¿Cómo seleccionar el tamaño del n y del k?

Se deben conocer (o suponer) los siguientes parámetros:

```
*n* - Un número mínimo de puntos con los cuales estimar la regresión.
*k* - El número máximo de iteraciones.
*e* - Probabilidad de que un punto sea un outlier. (Porcentaje esperado de outliers,
        entonces $d = N*(1 - e)$, si N es el número total de puntos).
*p* - Probabilidad deseada de haber llegado a una buena muestra.
```

$1 - e$ probabilidad de obtener un *inlier*.

$(1 - e)^n$ probabilidad de obtener **n** *inliers*.

$1 - (1 - e)^n$ probabilidad de que el conjunto esté contaminado con al menos un *outlier*.

$(1 - (1 - e)^n)^k$ probabilidad de que k muestras estén contaminadas con al menos un *outlier*.

$1 - (1 - (1 - e)^n)^k$ probabilidad de que al menos una muestra de k no esté contaminada con ningún *outlier*.

$1 - (1 - (1 - e)^n)^k = p$

Fijamos p en, por ejemplo, 99% y despejamos k.

$$k = \frac{log(1-p)}{log(1-(1-e)^n)} = \frac{-4.60517}{log(1-(1-e)^n)}$$

La tabla presenta el número de iteraciones que se deben especificar. Las filas de la tabla corresponden al número de puntos con respecto a los cuales se ajusta la regresión lineal y las columnas la proporción esperada de *outliers*.

	0.02	0.04	0.06	0.08	0.1	0.12	0.14	0.16	0.18	0.2	0.3	0.5
2	2	2	3	3	3	4	4	4	5	5	7	17
4	2	3	4	4	5	6	6	7	8	9	17	72
6	3	4	4	5	7	8	9	11	13	16	37	293
8	3	4	5	7	9	11	13	17	21	26	78	1177
10	3	5	6	9	11	15	19	24	32	41	161	4714
12	3	5	8	11	14	19	26	35	48	65	331	18861
14	4	6	9	13	18	26	36	51	72	103	677	75449
16	4	7	10	16	23	34	50	73	108	162	1384	301803
18	4	8	12	19	29	44	68	104	162	254	2826	1207216
20	5	8	14	23	36	58	92	149	242	398	5770	4828869
22	5	9	16	27	45	75	125	212	361	622	11777	19315482
24	5	10	18	32	56	97	170	301	537	973	24036	77261933
26	6	11	21	38	69	126	231	427	800	1522	49055	309047738
28	6	12	24	46	86	163	312	606	1191	2379	100114	1236190957
30	6	14	28	54	107	211	423	859	1772	3718	204315	4944763834

La tabla presenta que usar un n pequeño es mejor que uno grande. Y que el algoritmo es más rápido, por necesitar menor número de iteraciones, en presencia de una proporción pequeña de *outliers*.

Usos

Ransac se ideó para determinar la tendencia real a pesar del ruido. Actualmente es muy popular su uso en el área de procesamiento de imágenes. No es tradicional su uso para identificar outliers, pero en procesos en que sea lícito buscar puntos que no corresponden a un proceso de tendencia lineal (combinación lineal de variables), es una alternativa a explorar.

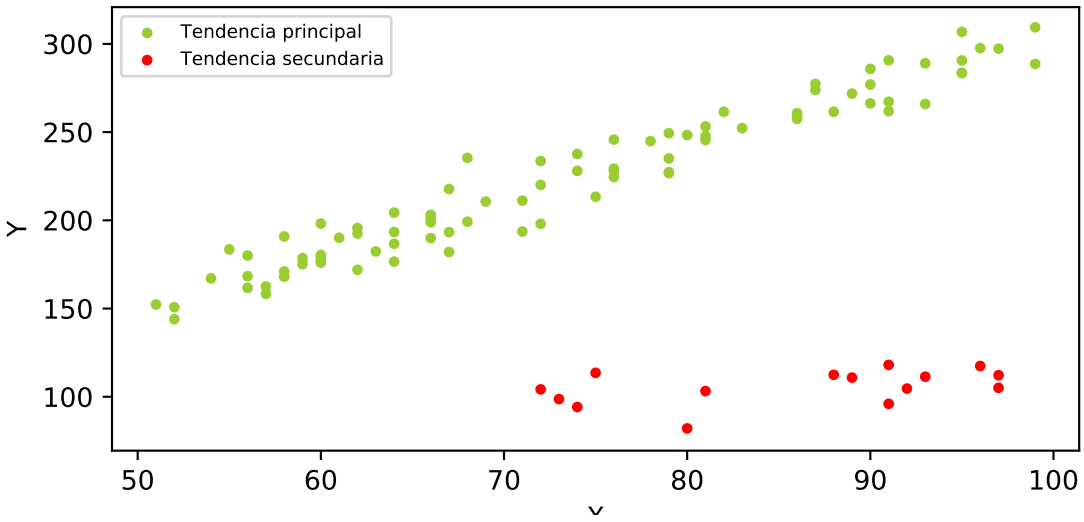
Práctica en python

```
# importar modulos
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import RANSACRegressor
# import random
from random import seed
```

Sea simulado un conjunto de 100 datos, en los cuales hay dos tendencias.

```
np.random.seed(158)
prin = 85
sec = 100 - prin
x1 = np.random.randint(50, 100, prin)
x2 = np.random.randint(70, 100, sec)
noise = np.random.normal(0, 10, prin)
y1 = 2 + 3 * x1 + noise
noise2 = np.random.normal(0, 10, sec)
y2 = 20 + x2 + noise2
x = np.append(x1, x2)
y = np.append(y1, y2)
```

```
lw = 2
plt.scatter(x1, y1, color='yellowgreen', marker='.',
            label='Tendencia principal')
plt.scatter(x2, y2, color='red', marker='.',
            label='Tendencia secundaria')
plt.legend(loc='upper left', fontsize='x-small')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



A continuación se aplica el algoritmo Ransac para separar la tendencia principal y la tendencia secundaria.

```
# Se requiere que el arreglo este en forma de columna
x1 = x.reshape(-1,1)
y1 = y.reshape(-1,1)
lr = LinearRegression()
lr.fit(x1, y1)
# Muestre el intercepto y el coeficiente
```

```
## LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
##          normalize=False)
```

```
lr.intercept_, lr.coef_
```

```
## (array([80.57262198]), array([[1.65194051]]))
```

```
ransac = RANSACRegressor(LinearRegression())
ransac.fit(x1, y1)
# yhat_ransac = ransac.predict(y1)
# El objeto guarda los inliers
```

```
## RANSACRegressor(base_estimator=LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
##          normalize=False),
##         is_data_valid=None, is_model_valid=None, loss='absolute_loss',
##         max_skips=inf, max_trials=100, min_samples=None, random_state=None,
##         residual_threshold=None, stop_n_inliers=inf, stop_probability=0.99,
##         stop_score=inf)
```

```
inlier_mask = ransac.inlier_mask_
outlier_mask = np.logical_not(inlier_mask)
# Variables que se pueden usar luego si se desea
outliers_x = x[outlier_mask]
outliers_y = y[outlier_mask]
```

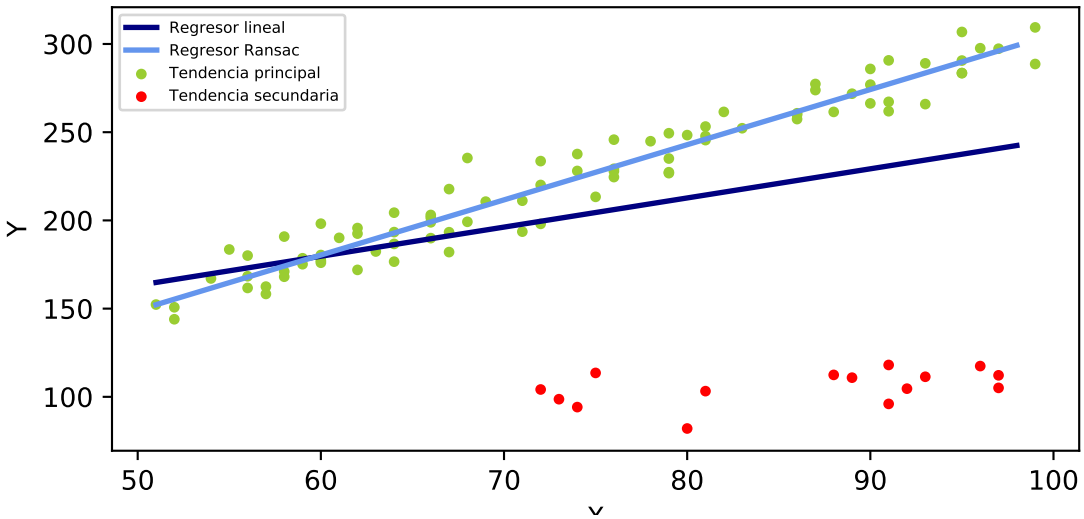
Obsérvese que el algoritmo elige internamente los parámetros con los cuales ejecutarse.

```
line_x = np.arange(x1.min(), x1.max())[:, np.newaxis]
line_y = lr.predict(line_x)
line_y_ransac = ransac.predict(line_x)
# Muestre el intercepto y el coeficiente
ransac.estimator_.intercept_, ransac.estimator_.coef_
```

```
## (array([-7.29538489]), array([[3.12758957]]))
```

Mediante la librería matplotlib se presenta el resultado.

```
lw = 2
plt.scatter(x[inlier_mask], y[inlier_mask], color='yellowgreen', marker='.',
            label='Tendencia principal')
plt.scatter(x[outlier_mask], y[outlier_mask], color='red', marker='.',
            label='Tendencia secundaria')
plt.plot(line_x, line_y, color='navy', linewidth=lw, label='Regresor lineal')
plt.plot(line_x, line_y_ransac, color='cornflowerblue', linewidth=lw,
         label='Regresor Ransac')
plt.legend(loc='upper left', fontsize='xx-small')
plt.xlabel("X")
plt.ylabel("Y")
plt.show()
```



Ejercicio