Tarea 7 de Modelos Probabilistas Aplicados

Transformaciones mediante Escalera de poderes de Tukey

5271

20 de octubre de 2020

1. Introducción

En este trabajo se realiza un breve acercamiento al tema Transformaciones mediante Escalera de poderes de Tukey y regresión lineal. Además presenta un algoritmo desarrollado en lenguaje R para extraer la función que genera la variable dependiente así como sus coeficientes. La experimentación se realiza en el programa R versión 4.0.2 [1] en el entorno de desarrollo Rstudio [2]

2. Transformaciones mediante Escalera de poderes de Tukey

Para entender mejor como funciona las transformaciones mediante Escalera de poderes de Tukey creamos un *data.frame* con datos bivariados (x_1, y) de manera que x_1 se distribuye uniformemente e $y = ax_1 + b$. Lo anterior se realiza con el código 2 en R.

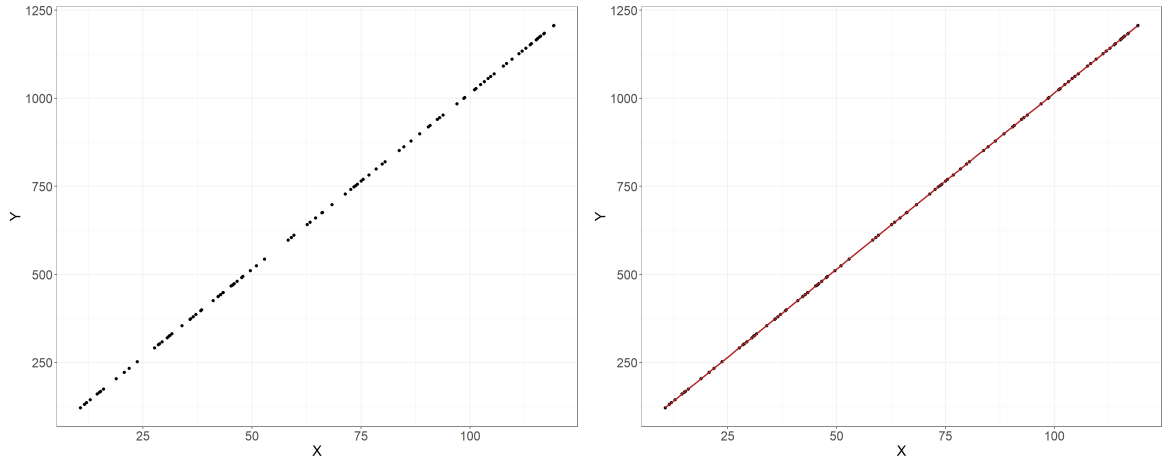
```r
x_1 = runif(100,10,120)
a = 8
b = 15
y = a*(x_1) + b
datas = as.data.frame(y)
datas = cbind(datas,x_1)
```

Tarea7n.R

Como primer paso se trazan los datos en un diagrama de dispersión. En la figura 1 de la página 2 se puede observar como se relaciona las variables x_1 e y. Como se aprecia en dicha figura la relación entre x_1 e y es una dependencia lineal y si aplicamos una regresión lineal obtendremos exactamente los coeficientes y formulación con la cual se crea la variable dependiente y. A continuación se metra el resultado de aplicar la regresión lineal. De donde se pueden extraer los coeficientes a y b de la ecuación, quedando $y = 15x_1 + 8$.

```
Call:
lm(formula = datas$y ~ datas$x_1)

Residuals:
```



(a) $y = ax_1 + b$ (b) regresión lineal

Figura 1: Diagramas de dispersión de los datos bivariados (x_1, y)

```
       Min         1Q     Median         3Q        Max
-8.102e-13 -3.110e-15  8.040e-15  1.811e-14  9.951e-14

Coefficients:
             Estimate Std. Error   t value Pr(>|t|)
(Intercept) 1.500e+01  2.016e-14 7.442e+14   <2e-16 ***
datas$x_1   8.000e+00  2.744e-16 2.915e+16   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 8.501e-14 on 98 degrees of freedom
Multiple R-squared:      1,Adjusted R-squared:      1
F-statistic: 8.498e+32 on 1 and 98 DF,  p-value: < 2.2e-16

Warning message:
In summary.lm(modelo_lineal) :
  essentially perfect fit: summary may be unreliable
```

El ejemplo anterior es sencillo pues con solo una regresión lineal se pueden obtener los coeficientes y la función que crea a y. En cambio si creamos representamos los datos bivariados creados con $y = a * log(x_1) + b$, se puede observar en la figura 2 de la página 3 que la relación de dependencia no es lineal, por lo que una simple regresión no arrogará los resultados deseados, por lo cual antes de aplicar la regresión se debe hacer una transformación a una de las variable, es decir tratar de linealizar la relación de dependencia.

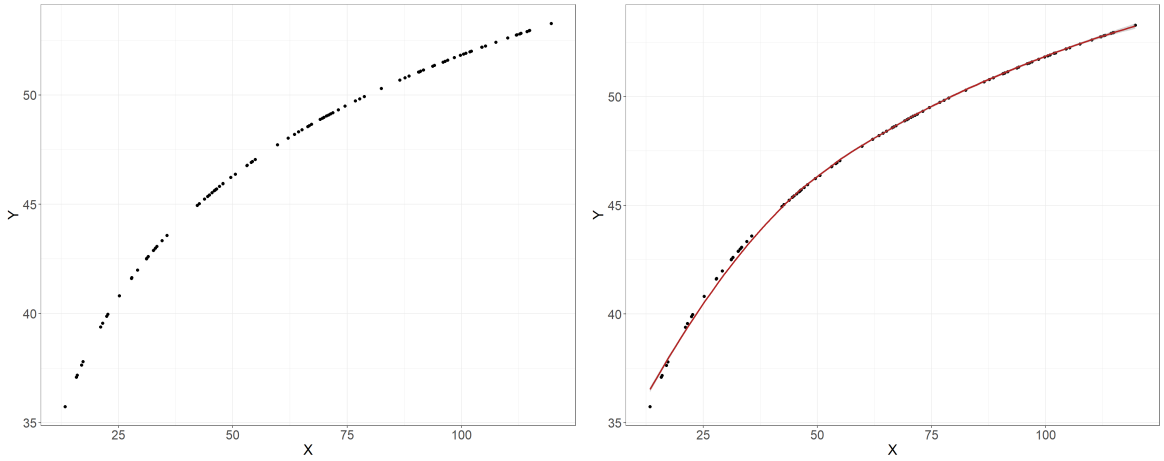
No hay ninguna restricción sobre los valores de λ que podamos considerar. Obviamente, elegir $\lambda = 1$ deja los datos sin cambios. Los valores negativos de λ también son razonables. Lo que da lugar a la ecuación 2. El cuadro 1 de la página 3 se muestran ejemplos de la escalera de transformaciones de Tukey.

Tukey sugiere explorar relaciones como:

$$y = ax_1^{\lambda} + b \tag{1}$$

Cuadro 1: Escalera de transformaciones de Tukey

λ	-2	-1	-1/2	0	1/2	1	2
y	$\frac{-1}{x^2}$	$\frac{-1}{x}$	$\frac{-1}{\sqrt{x}}$	$\log x$	$\sqrt{x}$	x	x^2



(a) $y = a * log(x_1) + b$ (b) regresión lineal

Figura 2: Diagramas de dispersión de los datos bivariados (x_1, y) con dependencia no lineal

donde λ es una parámetro que elegido con el fin de linealizar la relación.

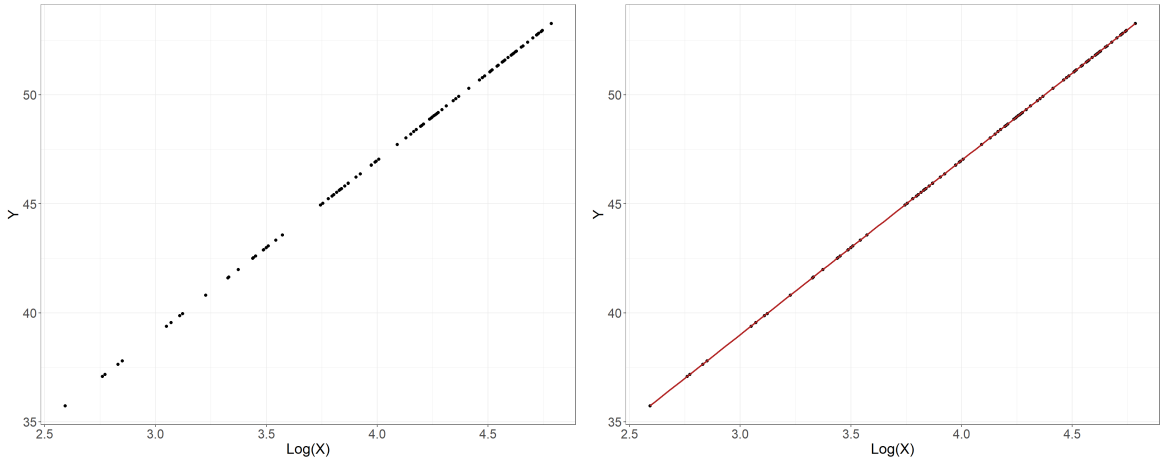
$$y = \begin{cases} x^{\lambda} & \text{si} \lambda > 0 \\ \log \text{ x} & \text{si} \lambda = 0 \\ -(x^{\lambda}) & \text{si} \lambda < 0 \end{cases} \tag{2}$$

Se Aplica transformaciones mediante la escalera de Tukey a los datos, en forma que se calcula el logaritmo de la variable x_1, y se vuelve a representar los datos y se realiza la regresión lineal. En la figura 3 de la página 4 se puede observar la transformación de los datos.

3. Algoritmo propuesto

Según lo visto en la sección anterior podemos mediante las transformaciones de la Escalera de Tukey, podemos acercar la relación entre los datos bivariados a una dependencia lineal. A partir de está conclusión se creo una función en R que la pasarle como parámetros un *data.frame* y la cantidad de variables de cuales depende y (máximo tres variables), devuelve un *data.frame* con los valores de λ encontrados para la linealización, los coeficientes de la función y el error estándar de la estimación delos coeficientes. En el código **??** se muestra dicha función.

```
ajuste1 = function(datos,v){
  rsq=numeric()
  landax1=numeric()
  landax2=numeric()
  landax3=numeric()
  coefi_a=numeric()
```



(a) $y = a * log(x_1) + b$ (b) regresión lineal

Figura 3: Diagramas de dispersión de los datos bivariados (x_1, y) con dependencia no lineal

```
coefi_b=numeric()
coefi_c=numeric()
error_a=numeric()
error_c=numeric()
error_b=numeric()
corr = numeric()
diferencia = numeric()
s=seq(-2,2,0.25)
if(v==1){
  for (i in s) {
    x1=numeric()
    if(i<0){
      x1=-1/((datos$x_1)**(i*-1))
    }else if(i>0){
      x1=(datos$x_1)**i
    }else{
      x1 =log(datos$x_1)}

    modelo_lineal = lm(datos$y ~ x1)
    rsq = c(rsq,summary(modelo_lineal)$r.squared)
    diferencia = c(diferencia,(1-summary(modelo_lineal)$r.squared))
    coefi_a=c(coefi_a,modelo_lineal$coefficient[1])
    error_a=c(error_a,round(summary(modelo_lineal)$coefficient[1,2],4))
    coefi_b=c(coefi_b,modelo_lineal$coefficient[2])
    error_b=c(error_b,round(summary(modelo_lineal)$coefficient[2,2],4))
    landax1=c(landax1,i)
  }
  corre= as.data.frame(rsq)
  corrl= cbind(corre,landax1,diferencia,coefi_a,error_a,coefi_b,error_b)
  resultado=filter(corrl,diferencia==min(corrl$diferencia))


}else if(v==2){
  for (i in s) {
    x1=numeric()
    if(i<0){
      x1=-1/((datos$x_1)**(i*-1))
    }else if(i>0){
      x1=(datos$x_1)**i
    }else{
```

```
        x1 =log(datos$x_1)}
      for (j in s) {
        x2=numeric()
        if(j<0){
          x2=-1/((datos$x_2)**(j*-1))
        }else if(j>0){
          x2=(datos$x_2)**j
        }else{
          x2 =log(datos$x_2)}

        modelo_lineal = lm(datos$y ~ x1+x2)
        rsq = c(rsq,summary(modelo_lineal)$r.squared)
        diferencia = c(diferencia,(1-summary(modelo_lineal)$r.squared))
        coefi_a=c(coefi_a,modelo_lineal$coefficient[2])
        error_a=c(error_a,round(summary(modelo_lineal)$coefficient[2,2],4))
        coefi_b=c(coefi_b,modelo_lineal$coefficient[3])
        error_b=c(error_b,round(summary(modelo_lineal)$coefficient[3,2],4))
        landax1=c(landax1,i)
        landax2=c(landax2,j)

      }
    }
    corre= as.data.frame(rsq)
    corrl= cbind(corre,landax1,landax2,diferencia,coefi_a,error_a,coefi_b,error_b)
    resultado=filter(corrl,diferencia==min(corrl$diferencia))
  }else if(v==3){

    for (i in s) {
      x1=numeric()
      if(i<0){
        x1=-1/((datos$x_1)**(i*-1))
      }else if(i>0){
        x1=(datos$x_1)**i
      }else{
        x1 =log(datos$x_1)}
      for (j in s) {
        x2=numeric()
        if(j<0){
          x2=-1/((datos$x_2)**(j*-1))
        }else if(j>0){
          x2=(datos$x_2)**j
        }else{
          x2 =log(datos$x_2)}
        for (k in s) {
          x3=numeric()
          if(k<0){
            x3=-1/((datos$x_3)**(k*-1))
          }else if(k>0){
            x3=(datos$x_3)**k
          }else{
            x3 =log(datos$x_3)}

          modelo_lineal = lm(datos$y ~ x1+x2+x3)
          rsq = c(rsq,summary(modelo_lineal)$r.squared)
          diferencia = c(diferencia,(1-summary(modelo_lineal)$r.squared))
          coefi_a=c(coefi_a,modelo_lineal$coefficient[2])
          error_a=c(error_a,round(summary(modelo_lineal)$coefficient[2,2],4))
          coefi_b=c(coefi_b,modelo_lineal$coefficient[3])
          error_b=c(error_b,round(summary(modelo_lineal)$coefficient[3,2],4))
          coefi_c=c(coefi_c,modelo_lineal$coefficient[4])
          error_c=c(error_c,round(summary(modelo_lineal)$coefficient[4,2],4))
          landax1=c(landax1,i)
          landax2=c(landax2,j)
          landax3=c(landax3,k)
        }
```

```
      }
    }
    corre= as.data.frame(rsq)
    corrl= cbind(corre,landax1,landax2,landax3,diferencia,coefi_a,error_a,coefi_b,
    error_b,coefi_c,error_c)
    resultado=filter(corrl,diferencia == min(corrl$diferencia))

  }

  return(resultado)
}
```

Tarea7n.R

Resultados El código general se encuentra disponible en el repositorio. https://github.com/Albertomnoa/Tareas

Referencias

[1] R Core Team. R: Un lenguaje y un entorno para la informática estadística, 2020.

[2] RStudio Team. Rstudio: Entorno de desarrollo integrado para R, 2020.