Usaremos la librería `mvtnorm` para generar los datos; específicamente la función `rmvt`.

```
library(mvtnorm)
n <- 120
medias <- c(1,0)
df <- 3
Sigma <- matrix(c(1.64, -0.8, -0.8, 1), ncol = 2)*((df-2)/df)
Z <- rep(medias, each = n) + rmvt(n, sigma = Sigma, df = 3)
Z <- data.frame(Z)
names(Z) <- c("Y", "X")
```

Comprobamos las medias de los datos muestreados:

```
round(colMeans(Z), 2)
```

```
##    Y    X
## 0.89 0.07
```

Y la varianza:

```
round(var(Z), 2)
```

```
##       Y     X
## Y  1.58 -0.86
## X -0.86  0.89
```

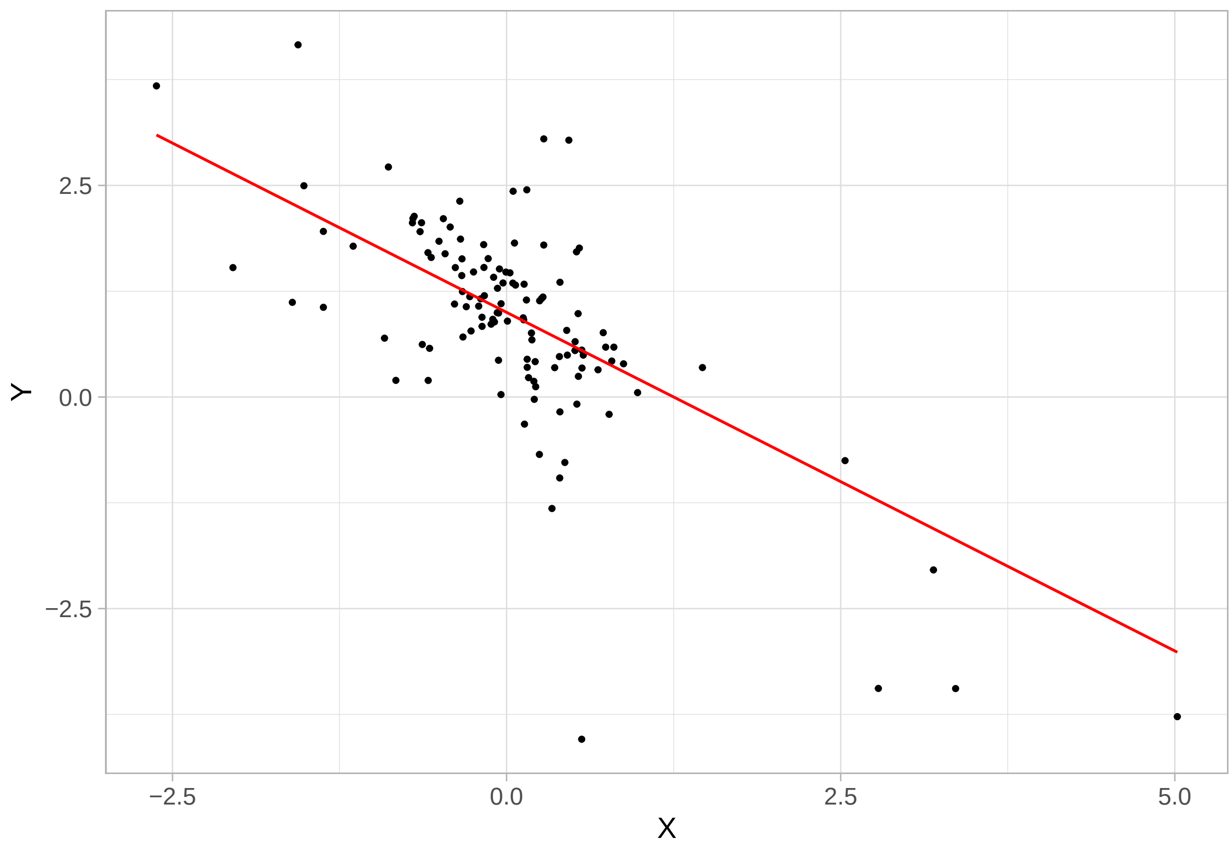
Dada la reducida muestra, los parámetros calculados pueden no ser iguales a los especificados, pero se deben asemejar, como puede verse.

La curva de regresión es: $g(X) = 1 - \frac{4}{5}X$

```
g <- 1-(4/5)*Z$X
```

Graficando los datos simulados y la curva de regresión:

```
library(ggplot2)
datos <- data.frame(Z, g)
names(datos)[3] <- c("g")
ggplot(data = datos) +
  geom_point(aes(x = X, y = Y),
             size = 0.6) +
  geom_line(aes(x = X, y = g),
            col = "red") +
  theme_light()
```


2.5
0.0
−2.5
Y
−2.5
0.0
2.5
5.0
X

Generemos los datos de la distribución utilizando la siguiente función, basada en la idea de que X sigue una distribución exponencial y que $Y|X$ sigue una distribución gamma. Se sigue la siguiente estrategia para muestrear[1]:

1. Muestreamos X de una distribución exponencial.
2. Muestreamos Y condicionada en X

Entonces, para el caso donde $\lambda = 0.5$

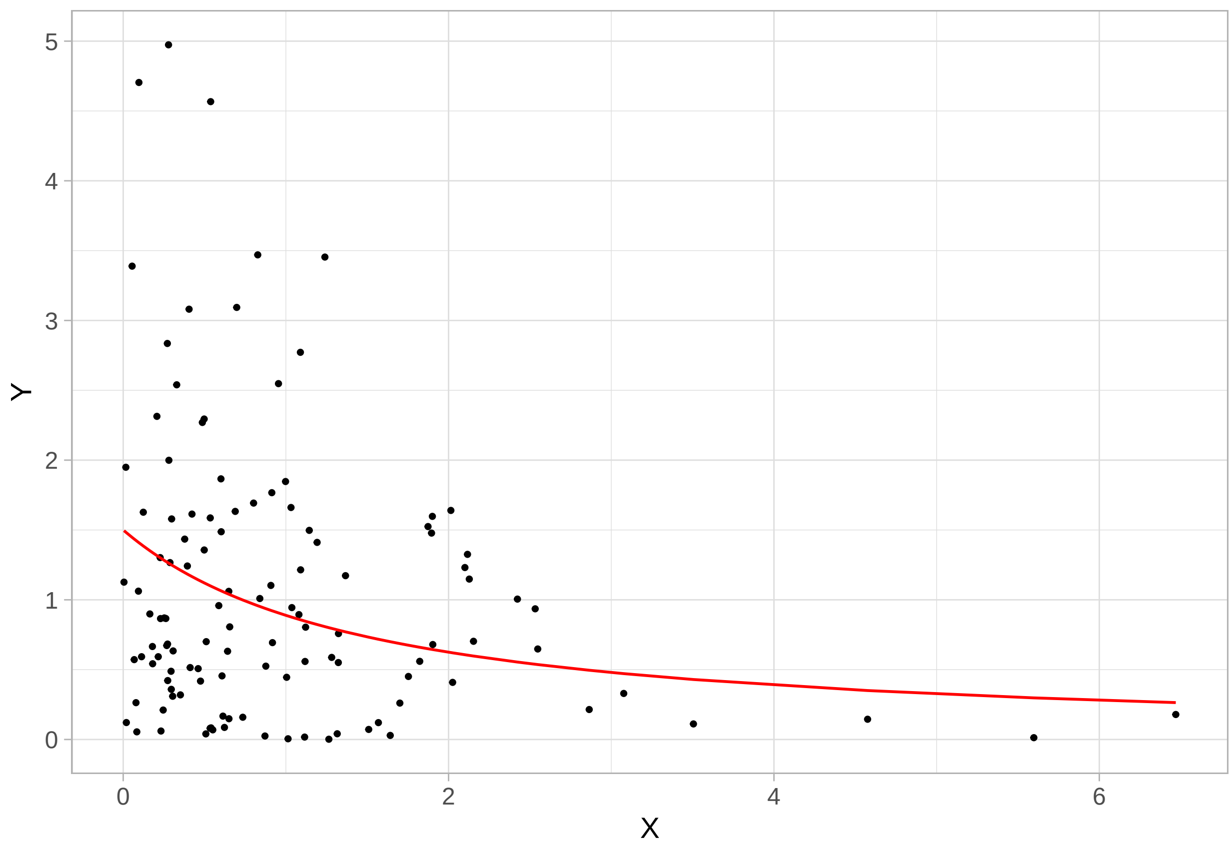
```
rGBVE <- function(n, alpha1, alpha2, lambda) {
  x1 <- rexp(n, alpha1)
  alpha12 <- alpha1*alpha2
  pprod <- alpha12*lambda
  C <- exp(alpha1*x1)
  A <- (alpha12 - pprod + pprod*alpha1*x1)/C
  B <- (pprod*alpha2 + pprod^2*x1)/C
  D <- alpha2 + pprod*x1
  wExp <- A/D
  wGamma <- B/D^2
  data.frame(x1, x2 = rgamma(n, (runif(n) > wExp/(wExp + wGamma)) + 1, D))
}

n <- 120
lmda <- 0.5
Z <- rGBVE(n = n, alpha1 = 1, alpha2 = 1, lambda = lmda)
names(Z) <- c("X", "Y")
```

La gráfica de los datos contra su regresión es:

```
g <- function(x, lambda) {
  (1+lambda+lambda*x)/(1+lambda*x)^2
}
datos <- data.frame(Z, g(Z$X, lambda = lmda))
names(datos) <- c("X", "Y", "g")
ggplot(data = datos) +
  geom_point(aes(x = X, y = Y),
             size = 0.6) +
  geom_line(aes(x = X, y = g),
            col = "red") +
  theme_light()
```

[1] Basado en https://stackoverflow.com/questions/71176416/how-to-simulate-data-from-gumbel-bivariate-exponential-distribution


Y
X
0
1
2
3
4
5
0
2
4
6

Generar 30 observaciones i.i.d. $\left\{\begin{pmatrix} X_{i1} \\ X_{i2} \\ V_i \end{pmatrix}\right\}_{i=1}^{30}$ donde $\begin{pmatrix} X_{i1} \\ X_{i2} \\ V_i \end{pmatrix} \sim N_3\left(\begin{pmatrix} 1 \\ 2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0.04 & -0.03 & 0 \\ -0.03 & 0.09 & 0.01 \\ 0 & 0.01 & 0.01 \end{pmatrix}\right)$

```
n <- 30
medias <- c(1,2,0)
Sigma <- matrix(c(0.04, -0.03, 0, -0.03, 0.09, 0.01, 0, 0.01, 0.01), ncol = 3)
Z <- rmvnorm(n, mean = medias, sigma = Sigma)
Z <- data.frame(Z)
names(Z) <- c("X1", "X2", "V")
```

Podemos ver que la media de los datos muestreados se asemeja a la especificada:

```
round(colMeans(Z), 2)
```

```
##   X1   X2    V
## 0.99 2.07 0.01
```

Y su matriz de varianza-covarianza:

```
round(var(Z), 3)
```

```
##        X1     X2      V
## X1  0.039 -0.032 -0.003
## X2 -0.032  0.075  0.005
## V  -0.003  0.005  0.007
```

Generar

$$Y_i = e^{X_{i1}} - sen(X_{2i}) + V_i \; i = 1, 2, \ldots, 30$$

```
Y <- exp(Z$X1) - sin(Z$X2) + Z$V
```

Para explicar la variable Y en función de las variables explicativas X_1 y X_2 se propone el "modelo":

$$Y_i = 10 + \pi X_{i1} + eX_{i2} + U_i, \text{ con } i = 1, 2, \ldots, 30$$

```
g1 <- 10 + pi*Z$X1 + exp(1)*Z$X2
U <- Y - g1
Ymodel1 <- g1+U
```

¿Se cumple este modelo? (i.e. ¿Se satisfacen las 30 ecuaciones con los datos)

La respuesta a esta pregunta es **sí**, debido a que, a pesar de que quizás no es el mejor modelo, existen $U_1, U_2, U_3, \ldots, ..., U_{30}$ tales que hacen que se cumplan las 30 ecuaciones del vector `Ymodel1`, es decir, ocurre que Y_i=`Ymodel1_i` $\forall i \in \{1, 2, \ldots, 30\}$

```r
data.frame(Y,Ymodel1,g1,U)
```

```
##            Y   Ymodel1       g1         U
## 1  1.7884208 1.7884208 18.26531 -16.47688
## 2  1.2878967 1.2878967 18.71571 -17.42781
## 3  1.5039825 1.5039825 19.04973 -17.54575
## 4  1.5469401 1.5469401 18.80553 -17.25859
## 5  1.4030178 1.4030178 19.08179 -17.67877
## 6  1.8192765 1.8192765 17.72652 -15.90724
## 7  1.9581314 1.9581314 18.23897 -16.28084
## 8  2.5842788 2.5842788 19.51555 -16.93127
## 9  2.1152657 2.1152657 19.54978 -17.43452
## 10 2.5914916 2.5914916 19.56146 -16.96996
## 11 2.1551966 2.1551966 19.59259 -17.43739
## 12 2.6205560 2.6205560 18.81469 -16.19413
## 13 2.9480458 2.9480458 19.30512 -16.35707
## 14 0.8715154 0.8715154 17.72907 -16.85756
## 15 2.0676019 2.0676019 18.92516 -16.85756
## 16 1.9940734 1.9940734 18.24967 -16.25559
## 17 1.6363613 1.6363613 18.92274 -17.28637
## 18 1.7214821 1.7214821 17.04938 -15.32790
## 19 2.0713315 2.0713315 18.87721 -16.80588
## 20 2.0202826 2.0202826 18.55541 -16.53512
## 21 2.1093777 2.1093777 19.45466 -17.34528
## 22 1.6713971 1.6713971 17.92193 -16.25053
## 23 2.0605541 2.0605541 18.96785 -16.90729
## 24 1.6809899 1.6809899 18.43920 -16.75821
## 25 2.1923240 2.1923240 18.54016 -16.34783
## 26 1.3165268 1.3165268 18.37938 -17.06285
## 27 2.0775279 2.0775279 19.31715 -17.23963
## 28 1.9234088 1.9234088 19.42126 -17.49786
## 29 1.7089578 1.7089578 18.64186 -16.93290
## 30 1.8767981 1.8767981 18.77788 -16.90108
```

Se propone un segundo modelo:

$$Y_i = 13 + 3X_{i1} + 2024X_{i2} + \epsilon_i, \text{ con } i = 1, 2, \ldots, 30$$

```r
g2 <- 13 + 3*Z$X1 + 2024*Z$X2
epsilon <- Y - g2
Ymodel2 <- g2+epsilon
```

¿Se cumple este modelo? (i.e. ¿Se satisfacen las 30 ecuaciones con los datos)

La respuesta es nuevamente **sí**, por la misma razón que el modelo anterior: existen $\epsilon_1, \epsilon_2, \ldots, \epsilon_{30}$ que hacen que las 30 ecuaciones se cumplan; es decir, ocurre que Y_i=`Ymodel2_i` $\forall i \in \{1, 2, \ldots, 30\}$:

```
data.frame(Y, Ymodel2, g2, epsilon)
```

```
##            Y   Ymodel2       g2   epsilon
## 1  1.7884208 1.7884208 3885.691 -3883.902
## 2  1.2878967 1.2878967 4976.302 -4975.014
## 3  1.5039825 1.5039825 5031.823 -5030.319
## 4  1.5469401 1.5469401 4591.496 -4589.949
## 5  1.4030178 1.4030178 5348.670 -5347.267
## 6  1.8192765 1.8192765 3233.600 -3231.781
## 7  1.9581314 1.9581314 3703.053 -3701.095
## 8  2.5842788 2.5842788 4341.297 -4338.713
## 9  2.1152657 2.1152657 4841.780 -4839.665
## 10 2.5914916 2.5914916 4161.874 -4159.282
## 11 2.1551966 2.1551966 4649.196 -4647.040
## 12 2.6205560 2.6205560 3624.469 -3621.848
## 13 2.9480458 2.9480458 3689.284 -3686.336
## 14 0.8715154 0.8715154 4646.504 -4645.633
## 15 2.0676019 2.0676019 4132.454 -4130.386
## 16 1.9940734 1.9940734 3720.147 -3718.153
## 17 1.6363613 1.6363613 4585.128 -4583.492
## 18 1.7214821 1.7214821 2923.309 -2921.587
## 19 2.0713315 2.0713315 4195.861 -4193.789
## 20 2.0202826 2.0202826 3788.968 -3786.947
## 21 2.1093777 2.1093777 4704.887 -4702.777
## 22 1.6713971 1.6713971 3575.022 -3573.351
## 23 2.0605541 2.0605541 4268.111 -4266.050
## 24 1.6809899 1.6809899 4177.740 -4176.059
## 25 2.1923240 2.1923240 3646.717 -3644.524
## 26 1.3165268 1.3165268 4519.882 -4518.566
## 27 2.0775279 2.0775279 4437.800 -4435.722
## 28 1.9234088 1.9234088 4558.127 -4556.204
## 29 1.7089578 1.7089578 4216.259 -4214.550
## 30 1.8767981 1.8767981 4117.488 -4115.611
```