User's Guide for A Data Simulation Function For Any Multiple Linear model

Chenguang Pan

2023-04-21

0.1 Project Address

I synced the progress of this function on the Github. You can check it here if interested: https://github.com/cgpan/simdata

0.2 Data simulation function

This function is to generate data from any given **multiple linear regression model**. Before running this function, please make sure you installed the package `mvtnorm`. Please run the latest version `03_Data_simulation_V1.2.R`.

There are several arguments a user need to know:

- `size=500`, the number of generated observations is set to 500 by default. You can give any number you want;
- `beta_=`, you must give a coefficient array. It should be in the sequence of c(β_0,β_1,β_2,β_3...), which means the interception should be first numeric item;
- `pred_means=`, since this function mainly generate data from a multivariate norm distribution, you can give the variables means or it will use 0 by default;
- `pre_cov=`, you can give the variance-covariance matrix, or it will use the variance =1, covariance =0 matrix by default;
- `error_df=1`, the error will be generated from a normal distribution with mean=0 and sd=1, you can set the sd of error based on your theory;

The function will return a `dataframe` not a matrix, with the columns of `Y`, `X1`, `X2`,...

0.3 The source code of this function

```
> # before running the function , please load the package first
> library(mvtnorm)
> # write a function to generate data
> dat_gen <- function(size=500, # datasize sets to 500 by default,you can change to any size
+                     beta_ = NULL, # necessary, input the betas in the seugnce of c(b0, b1, b2...)
+                     pred_means = NULL, # not necessary,input the means of each variables
```

```
+                      pred_cov = NULL, # not necessary, input the cov matrix of variables
+                      error_sd = 1){
+   if (is.null(beta_) == TRUE){
+     # if user did not give a coefficients array, return warning.
+     return("Error: You need to give the coefficients in the sequence of beta0, beta1,...")
+   }else {
+     if (is.null(pred_means)== TRUE){
+       # if user did not give means of each variables,
+       # use 0 as means by default
+       predictor_nums <- length(beta_)
+       if (is.null(pred_cov)== TRUE){
+         print("Using the variance 1 and covariance 0 by default")
+         X <- rmvnorm(n=size,sigma=diag(predictor_nums-1))
+         Error <- as.matrix(rnorm(n=size, mean = 0, sd=error_sd))
+         X_aug <- cbind(rep(1,nrow(X)), X)
+         Y <- X_aug %*% as.matrix(beta_) + Error
+         out_data <- cbind(Y,X)
+       } else{ ### user gives the predictors covariance matrix
+         X <- rmvnorm(n=size,sigma=pred_cov)
+         Error <- as.matrix(rnorm(n=size, mean = 0, sd=error_sd))
+         X_aug <- cbind(rep(1,nrow(X)), X)
+         Y <- X_aug %*% as.matrix(beta_) + Error
+         out_data <- cbind(Y,X)
+       }
+     }else{ ## user gives the means of predictors
+       if (is.null(pred_cov)==TRUE){
+         print("Using the variance 1 and covariance 0 by default")
+         X <- rmvnorm(n=size,mean= pred_means,sigma=diag(predictor_nums-1))
+         Error <- as.matrix(rnorm(n=size, mean = 0, sd=error_sd))
+         X_aug <- cbind(rep(1,nrow(X)), X)
+         Y <- X_aug %*% as.matrix(beta_) + Error
+         out_data <- cbind(Y,X)
+       } else{ ### user gives the predictors covariance matrix
+         X <- rmvnorm(n=size,mean= pred_means,sigma=pred_cov)
+         Error <- as.matrix(rnorm(n=size, mean = 0, sd=error_sd))
+         X_aug <- cbind(rep(1,nrow(X)), X)
+         Y <- X_aug %*% as.matrix(beta_) + Error
+         out_data <- cbind(Y,X)
+       }
+     }
+     # give columns names in "Y","X1","X2",...
+     n_ = predictor_nums - 1
+     x_vars <- c("Y")
+     for (i in 1:n_) {
+       x_vars[i+1] <- paste0("X",i)
+     }
+     colnames(out_data) <- x_vars
+     return(as.data.frame(out_data))
+   }
+ }
```

0.4 Test of this function

0.4.1 Step 1. Run a linear model on real data

Let's use the R-built-in dataset `mtcars` to test this function. First, I run a linear model as follows:

```
> # test
> data(mtcars)
> # Fit a linear regression model to predict miles per gallon (mpg) based on horsepower (hp)
> model <- lm(mpg ~ hp+wt, data = mtcars)
>
> # View the summary of the model
> summary(model)

Call:
lm(formula = mpg ~ hp + wt, data = mtcars)

Residuals:
   Min     1Q Median     3Q    Max
-3.941 -1.600 -0.182  1.050  5.854

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 37.22727    1.59879  23.285  < 2e-16 ***
hp          -0.03177    0.00903  -3.519  0.00145 **
wt          -3.87783    0.63273  -6.129 1.12e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.593 on 29 degrees of freedom
Multiple R-squared:  0.8268,    Adjusted R-squared:  0.8148
F-statistic: 69.21 on 2 and 29 DF,  p-value: 9.109e-12
```

The coefficients are `37.227`, `-0.032`, and `-3.878`. The standard deviation of error term is 2.5934118. Let's plug this value in the function and generate a sample with size of 500.

0.4.2 Step 2. Data simulation using the function above

Note, in this step, I did not give the variance- covariance matrix. The function will use the var=1 and cov=0 by defalt.

```
> set.seed(666)
> sim_data <- dat_gen(size=500,
+                     beta_=c(37.23,-0.03177295,3.87783074),
+                     error_sd = summary(model)$sigma)
[1] "Using the variance 1 and covariance 0 by default"
> head(sim_data)
         Y         X1          X2
1 45.32666  0.7533110  2.01435467
2 49.16243 -0.3551345  2.02816784
3 37.86073 -2.2168745  0.75839618
4 31.20181 -1.3061853 -0.80251957
5 37.03116 -1.7922408 -0.04203245
6 31.30775  2.1500426 -1.77023084
```

The data looks good. Now, run linear model on this simulated dataset.

0.4.3 Step 3. Run a linear model on the simulated data.

```
> sim_model <- lm(Y ~ X1 +X2, data = sim_data)
> summary(sim_model)

Call:
lm(formula = Y ~ X1 + X2, data = sim_data)

Residuals:
   Min     1Q Median     3Q    Max 
-8.147 -1.821  0.048  1.695  7.388 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  37.2118     0.1132  328.75   <2e-16 ***
X1           -0.0315     0.1214   -0.26    0.795    
X2            3.8462     0.1099   35.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.529 on 497 degrees of freedom
Multiple R-squared:  0.7123,	Adjusted R-squared:  0.7111 
F-statistic: 615.3 on 2 and 497 DF,  p-value: < 2.2e-16
```

Note, the first predictor become non-significant. We might want to input the covariance matrix of the variables to make data simulation more accurate. Let's try it again.

```
> # get the var-cov matrix
> data(mtcars)
> cov_m <- cov(mtcars[,c("hp","wt")])
> # input the cov-matrix into the function and re-run the steps above
> set.seed(666)
> sim_data <- dat_gen(size=500,
+                     beta_=c(37.23,-0.03177295,3.87783074),
+                     pred_cov = cov_m,
+                     error_sd = summary(model)$sigma)
> sim_model <- lm(Y ~ X1 +X2, data = sim_data)
> summary(sim_model)

Call:
lm(formula = Y ~ X1 + X2, data = sim_data)

Residuals:
   Min     1Q Median     3Q    Max 
-8.147 -1.821  0.048  1.695  7.388 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) 37.211830   0.113191  328.75   <2e-16 ***
X1          -0.031369   0.002195  -14.29   <2e-16 ***
```

```
X2           3.834801   0.149183   25.70   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.529 on 497 degrees of freedom
Multiple R-squared:  0.5712,    Adjusted R-squared:  0.5695
F-statistic: 331.1 on 2 and 497 DF,  p-value: < 2.2e-16
```

Now comparing the result from the model on the real data. these simulated data looks good!