#Clear variables in global environment

rm(list=ls())

* Simulate some data where u is not correlated to x

true.beta <-c(2,-3,4)

set.seed(12345)

# 100000 draws from X ~ N(1.8,2)

x1<-rnorm(100000, mean=1.8, sd=2)

# 100000 draws from X ~ N(5,4)

x2<-rnorm(100000, mean=5, sd=4)

# 100000 draws from U ~ N(0,4)

u<-rnorm(100000, mean=0, sd=4)

# Create y dependent or outcome variable from the population regression function

y <- true.beta[1] + true.beta[2]\*x1 + true.beta[3]\*x2+ u

data<-data.frame(y, x1, x2, u)

* **Use loop to repeatedly draw samples of 500 from the population**

#Using the for loop so you can store the betas

beta0s<-**c()**

beta1s<-c()

beta2s<-c()

**for (i in 1:5000)** {

# draw a sample of 1000 observations

sample<-data[sample(nrow(data), 500),]

reg<-lm(y ~ x1 + x2, sample)

beta0s<-c(beta0s,reg$coefficients["(Intercept)"])

beta1s<-c(beta1s,reg$coefficients["x1"])

beta2s<-c(beta2s,reg$coefficients["x2"])

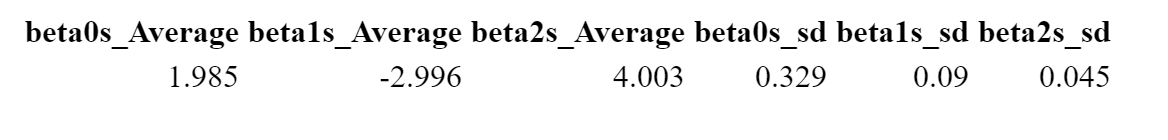
}

#Add to the existing vectors

betas<-as.data.frame(**cbind**(beta0s,beta1s, beta2s))

betas %>% summarise\_all(list(Average=mean, sd=sd)) %>% kable(digits=3)

-Output:



* Loops in R run very slow, the purpose of the apply family is to avoid loops, the purpose is to apply many functions with redundant application to a R’s object.
* `**lapply()**` performs an operation to a list object and returns a list object of the required length of the object

**lapply(x, fun)**

* sapply() function takes a list, vector or data frame as input and gives output in vector or matrix. It simplifies the object. sapply() function does the same job as lapply() function but returns a vector.

**sapply(x, fun)**

Example:

# we want to *run a regression 5000* *times*

list.lm<-**lapply**(1:5000, function(x)lm(y~x1+ x2, data=data[sample(nrow(data), 500), ]))

# We want to *extract the coefficients* from the regressions and put all of them into a vector.

betas\_apply<-as.data.frame(t(**sapply**(list.lm,coefficients)))

# Do *a table to show that it our betas are unbiased, plot the distribution of betas too*.

betas\_apply %>% summarise\_all(list(Beta=mean, Sd=sd)) %>% t() %>% kable(digits=4)

#*histograms*

p1<-ggplot(betas\_apply, aes(x=`(Intercept)`))+

geom\_histogram( fill=2)

p2<-ggplot(betas\_apply, aes(x=x1))+

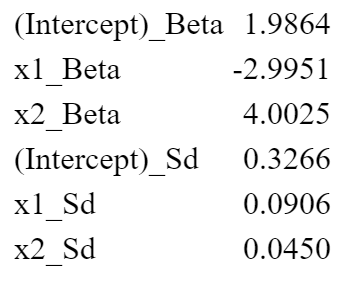
geom\_histogram(fill=4)

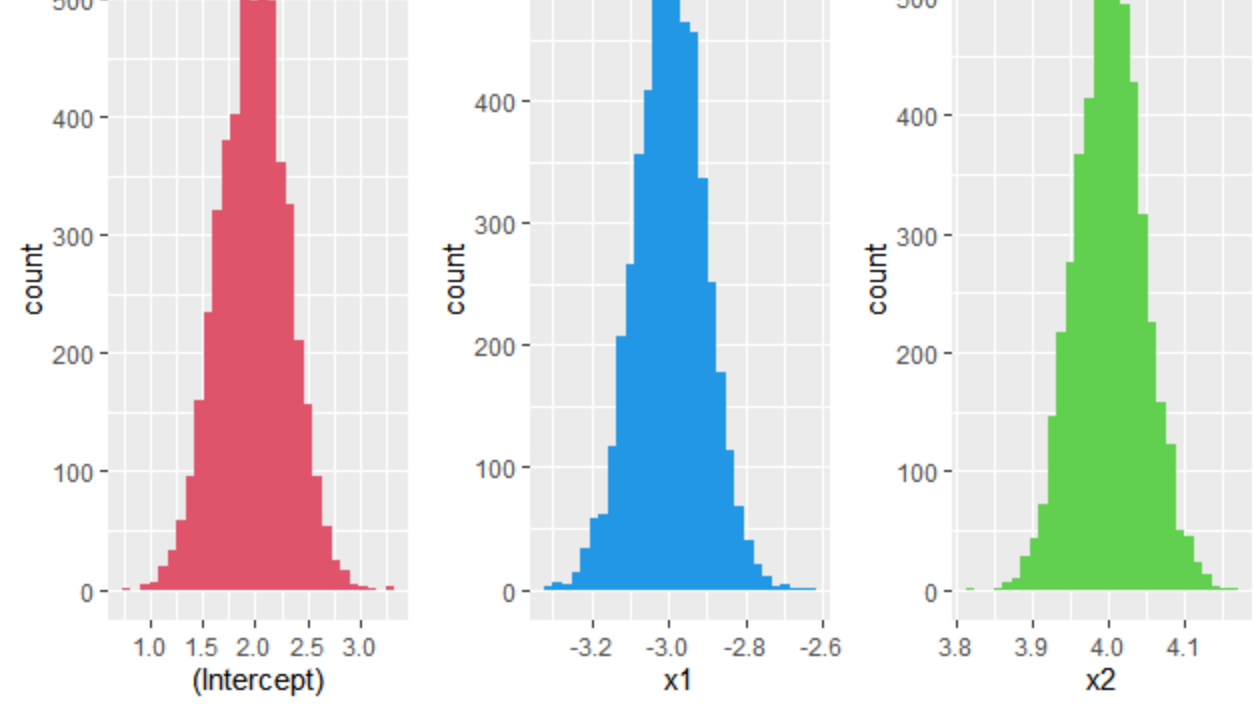
p3<-ggplot(betas\_apply, aes(x=x2))+

geom\_histogram(fill=3)

grid.arrange(p1, p2, p3, nrow = 1)

-Output:





* When x′s and u are correlated

# *start with the variance covariance matrix of x and u*

*set.seed(1234)*

varcovar<- matrix(c(1, 1.5, 1.7,

1.5, 9, 0.5,

1.7, 0.5, 9 ), nrow=3, ncol=3)

means<-c(1.5, 2.0, 0)

# *now let's create our dataframe with our independent variables x1 , x2, and the disturbance term u*

x<- data.frame(**rmvnorm**(1000000, means, varcovar))

# *y is generated same as before*

y <- true.beta[1] + true.beta[2]\*x[ , 1] + true.beta[3]\*x[ , 2]+ x[ , 3]

dt <- as.data.frame(cbind(y,x))

**colnames**(dt) <- c("y","x1","x2","u")

# Now the process to estimate the betas is the same as before but what happens with betas? (farther away from the true betas)

list.lmcovar<-lapply(1:5000, function(x)lm(y~x1+ x2, data=dt[sample(nrow(dt), 200), ]))

betas\_covar<-as.data.frame(t(sapply(list.lmcovar,coefficients)))

betas\_covar %>% summarise\_all(list(mean=mean, sd=sd))%>% t() %>% kable( digits=3)