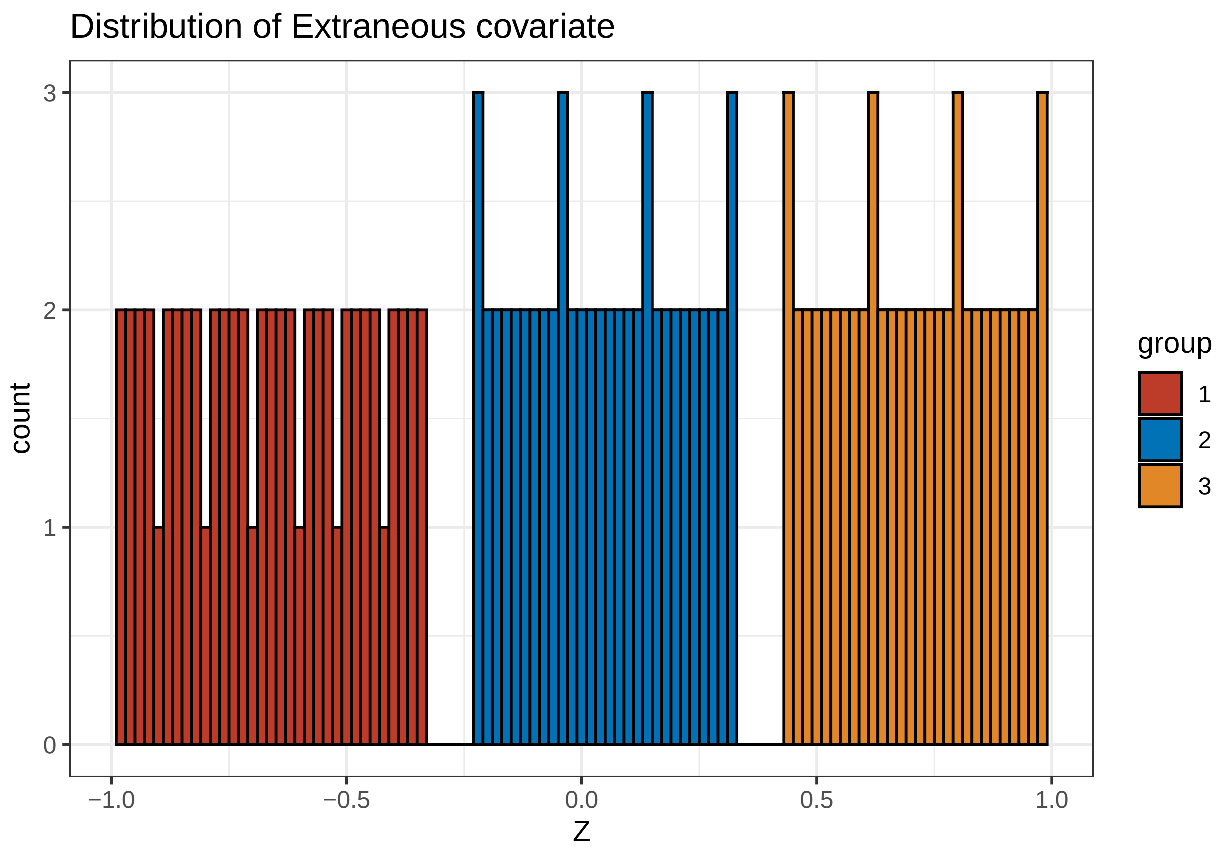
data-generation

```
#---------------------------------------------------------------------------------
#-------------------FUNCTION TO GENERATE CONTINUOUS DATA--------------------------
#---------------------------------------------------------------------------------

library(MASS)

# function to create the covariance matrix for a p-dimensional gaussian given
# the value of an extraneous covariate
# takes the scalar value of the covariate, the number of predictors, and the
# covariate bounds for each of the three clusters
# returns the covariance matrix
Var_cont <- function(z, p, limits1, limits2, limits3) {

  STR <- 1

  # determine the cluster of the given individual
  cl1 <- (limits1[1] <= z & z <= limits1[2]) * 1
  cl2 <- (limits2[1] <= z & z <= limits2[2]) * 1
  cl3 <- (limits3[1] <= z & z <= limits3[2]) * 1

  # create the precision matrix for the individual given their covariate
  pr <- matrix(0, p + 1, p + 1)

  # put 1 in the 2, 3 position
  pr[2, 3] <- STR

  # if the individual belongs to cluster 1 or 2, add a non-zero entry to the 1, 2 position
  pr[1, 2] <- STR * cl1 + (STR - STR * ((z + .23) / .56)) * cl2

  # if the individual belongs to cluster 2 or 3, add a non-zero entry to the 1, 3 position
  pr[1, 3] <- (STR * ((z + .23) / .56)) * cl2 + (STR) * cl3

  # symmetrize the matrix
  pr <- pr + t(pr)

  # add a 2 to the diagonal
  diag(pr) <- 2

  # find the covariance matrix from the precision matrix
  Var <- solve(pr)

  return(Var)
}

# function to generate the continuous data and covariates
# takes a RNG seed, sample size, number of predictors, and bounds of the
```

```r
# extraneous covariate for each of the three clusters
# returns the continuous data, covariates, and true covariance matrix
generate_continuous <- function(seed = 1, n = 180, p = 4,
                                limits1 = c(-.990, -.331),
                                limits2 = c(-.229, 0.329),
                                limits3 = c(0.431, 0.990)){

  set.seed(seed)

  # create covariate for individuals in each of the three clusters
  z1 <- seq(limits1[1], limits1[2], length = n %/% 3)
  z2 <- seq(limits2[1], limits2[2], length = n %/% 3)
  z3 <- seq(limits3[1], limits3[2], length = n %/% 3)
  Z <- matrix(c(z1, z2, z3), n, 1)

  # create the data matrix; each individual is generated from a MVN with 0 mean
  # and covariance matrix depending on their extraneous covariate
  data_mat <- matrix(0, n, p + 1)
  sigma_mats <- vector("list", n)
  for (l in 1:n) {

    # generate the covariance matrix depending on the covariates
    sigma_mats[[l]] <- Var_cont(Z[l], p, limits1, limits2, limits3)

    # draw from the multivariate normal
    data_mat[l, ] <- MASS::mvrnorm(1, rep(0, p + 1), sigma_mats[[l]])
  }

  return(list(data = data_mat, covts = Z, true_covariance = sigma_mats))

}

dat <- generate_continuous()
df <- cbind.data.frame(group = as.factor(rep(1:3, each = 60)), Z = dat$covts)
(ggplot2::ggplot(df, ggplot2::aes(Z, fill = group)) +
    ggplot2::geom_histogram(bins = 100, color = "black") +
    ggsci::scale_fill_nejm() +
    ggplot2::theme_bw() +
    ggplot2::ggtitle("Distribution of Extraneous covariate"))
```



```r
#---------------------------------------------------------------------------------
#-------------------FUNCTION TO GENERATE DISCRETE DATA---------------------------
#---------------------------------------------------------------------------------

# function to generate the discrete data and covariates
# takes a RNG seed, sample size, number of predictors, lambda (root of the
# non-zero elements in the precision matrix), values to populate the
# extraneous covariate vector with, and boolean for whether the two groups
# should have the same covariance
# returns the discrete data, covariates, and true covariance matrix
generate_discrete <- function(seed = 1, n = 100, p = 10, lambda = 15,
                              cov1 = -0.1, cov2 = 0.1, same = T){

  set.seed(seed)

  # generating the precision matrix: Assume two discrete covariate levels, one
  # for each group
  Lam1 <- c(rep(lambda, 4), rep(0, p - 3))
  Lam2 <- c(rep(0, 4), rep(lambda, p - 3))

  # if same is true, the individuals in both groups will have the same
  # covariance matrix
  if (same) Lam2 <- Lam1

  # create covariance matrix for both groups
  Var1 <- solve(Lam1 %*% t(Lam1) + diag(rep(10, p + 1)))
  Var2 <- solve(Lam2 %*% t(Lam2) + diag(rep(10, p + 1)))

  # create the extraneous covariate; individuals in group j have a covariate
  # vector of length p with cov_j as the only entry, j\in {1,2}
  Z <- matrix(c(rep(cov1, n %/% 2), rep(cov2, n %/% 2)), n, p)

  # create the data matrix; individuals in group j are generated from a MVN with
  # 0 mean vector and covariance matrix Var_j, j\in {1,2}
  X1 <- mvrnorm(n %/% 2, rep(0, p + 1), Var1)
  X2 <- mvrnorm(n %/% 2, rep(0, p + 1), Var2)
  data_mat <- rbind(X1, X2)

  return(list(data = data_mat, covts = Z, true_covariance = list(Var1, Var2)))

}

dat <- generate_discrete()
unique(dat$covts)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
## [1,] -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1  -0.1
## [2,]  0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.1  0.1   0.1
```