Simulation

Data Generation

```
# Example 2 of the thesis
# Generate data for simulation study
generate = function(n, # number of observations
                    p, # number of predictors
                    ps = 3, # number of strong signals
                    pwbc = 30, # number of WBC signals
                    pwai = 30, # number of WAI signals
                    corr = 0.7, # correlation coefficient (see example 2)
                    c = 20 # the c in the definition of strong and weak
                    ) {

  # Exception stop rules
  if (ps + pwbc + pwai >= n) {
    stop("number of true predictors should be less than number of observations")
  }
  if (abs(corr) > 1) {
    stop("correlation coefficient must be between -1 and 1")
  }
  # We hope that beta_strong = 20 and beta_weak = 0.5
  if ((c*sqrt(log(p)/n)) <= 0.5 & (c*sqrt(log(p)/n)) >= 20) {
    stop("please select other n and p")
  }

  # Generate correlation matrix based on example 2
  corr_matrix = matrix(rep(0, len = p^2), nrow = p)
  corr_num = pwbc %/% ps
  # Correlations between strong & wbc signals
  for (i in 1:(ps - 1)) {
    for (j in (ps + 1 + (i - 1)*corr_num):(ps + i*corr_num)) {
      corr_matrix[i, j] = corr
      corr_matrix[j, i] = corr
      for (k in j:(ps + i*corr_num)) {
        corr_matrix[j, k] = corr
        corr_matrix[k, j] = corr
      }
    }
  }
  for (j in (ps + 1 + (ps - 1)*corr_num):(ps + pwbc)) {
    corr_matrix[i, j] = corr
    corr_matrix[j, i] = corr
    for (k in j:(ps + pwbc)) {
      corr_matrix[j, k] = corr
      corr_matrix[k, j] = corr
    }
  }
  # Correlations within wai signals
  for (j in (ps + pwbc + 1):(ps + pwbc + pwai)) {
    for (k in j:(ps + pwbc + pwai)) {
      corr_matrix[j, k] = corr
      corr_matrix[k, j] = corr
    }
  }
  diag(corr_matrix) = 1

  # Generate simulation data
  X = mvrnorm(n, mu = rep(0, p), Sigma = corr_matrix, tol = 1)
  beta = c(rep(20, ps), rep(0.5, pwbc + pwai), rep(0, p - ps - pwbc - pwai))
  Y = X %*% beta + rnorm(n)
  df = as.data.frame(cbind(Y, X))
  colnames(df)[1] = "y"

  # Rename signals
  for (i in 2:(1 + ps)) {
    colnames(df)[i] = paste("strong", i - 1, sep = "_")
  }
  for (i in (2 + ps):(1 + ps + pwbc)) {
    colnames(df)[i] = paste("wbc", i - 1 - ps, sep = "_")
  }
  for (i in (2 + ps + pwbc):(1 + ps + pwbc + pwai)) {
    colnames(df)[i] = paste("wai", i - 1 - ps - pwbc, sep = "_")
  }
  for (i in (2 + ps + pwbc + pwai):(1 + p)) {
    colnames(df)[i] = paste("null", i - 1 - ps - pwbc - pwai, sep = "_")
  }

  return(df)
}
```

Simulation

```
# Simulate 100 times for n and p
# Simulation process
simulation = function(n, p, param_table) {
  param_estimate = param_table
  set.seed(1)
  for (i in 1:100) {
    # Data manipulation
    df = generate(n, p)
    X = as.matrix(df[1:p + 1])
    Y = as.matrix(df[1])

    # Forward Selection
    fit_forward = step(object = lm(y ~ 1, data = df),
                       scope = formula(lm(y ~ ., data = df)),
                       direction = "forward",
                       k = 2,
                       trace = 0)
    param_forward = fit_forward$coefficients[-1]

    # LASSO
    fit_lasso = cv.glmnet(X, Y,
                          nfolds = 10,
                          type.measure = "mse")
    param_lasso = fit_lasso$glmnet.fit$beta[, fit_lasso$lambda == fit_lasso$lambda.1se]

    param_estimate_i =
      rbind(param_forward, param_lasso) %>%
      data.frame %>%
      mutate(
        sim_time = i,
        n = n,
        p = p,
        method = c("forward", "lasso")
      ) %>%
      pivot_longer(
        cols = -c("sim_time", "n", "p", "method"),
        names_to = c("type", "num"),
          names_sep = "_",
        values_to = "estimate"
      )
    param_estimate = rbind(param_estimate, param_estimate_i)
  }
  return(param_estimate)
}

param_estimate = data.frame(matrix(ncol = 0, nrow = 0))
param_estimate = simulation(n = 100, p = 100, param_table = param_estimate)
param_estimate = simulation(n = 500, p = 100, param_table = param_estimate)
param_estimate = select(param_estimate, sim_time, n, p, method, everything())
write_csv(param_estimate, "parameter_estimate.csv")
```

Fitting Analysis

Settings	Case 1	Case 2
Number of Predictors	100	500
Strong Signals	3	3
Weak-Correlated Signals	30	30
Weak-Independent Signals	30	30
Null Signals	67	67

```
param_estimate = read_csv("parameter_estimate.csv")
```

```
## Rows: 40000 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): method, type
## dbl (5): sim_time, n, p, num, estimate
##
```

```
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```