synthetic datasets

```
Warning: package 'bartMachine' was built under R version 4.3.3

Warning: package 'randomForest' was built under R version 4.3.3

Warning: package 'missForest' was built under R version 4.3.3

Warning: package 'dbarts' was built under R version 4.3.3

Warning: package 'BART' was built under R version 4.3.3

Warning: package 'bench' was built under R version 4.3.3
```

create dataset

```r
linear_dgp_fun <- function(n_train, n_test, p, beta, noise_sd) {
  n <- n_train + n_test
  X <- matrix(rnorm(n * p), nrow = n, ncol = p)
  y <- X %*% beta + rnorm(n, sd = noise_sd)
  data_list <- list(
    X_train = X[1:n_train, , drop = FALSE],
    y_train = y[1:n_train],
    X_test = X[(n_train + 1):n, , drop = FALSE],
    y_test = y[(n_train + 1):n]
  )
  return(data_list)
}
linear_dgp <- create_dgp(
  .dgp_fun = linear_dgp_fun, .name = "Linear DGP",
  # additional named parameters to pass to .dgp_fun()
  n_train = 350, n_test = 120, p = 4, beta = c(1,2,1.5,3), noise_sd = 1
)
```

build BART model

```
BART_fun <- function(X_train, y_train, X_test, y_test, df,k,q) {
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- bench::mark(fit <- wbart(x.train = train_X,
                                y.train = y_train,
                                x.test = test_X,
                                k = k,
                                sigdf = df,
                                sigquant = q
                                ))
  time <- mean(t$time[[1]])
  predictions <- colMeans(fit$yhat.test)
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}

dbarts_fun <- function(X_train, y_train, X_test, y_test, df,k,q){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- bench::mark(bart_model <- bart(x.train = train_X,
                                       y.train = y_train,
                                       x.test = test_X,
                                       k = k,
                                       sigdf = df,
                                       sigquant = q))
  time <- mean(t$time[[1]])
  predictions <- colMeans(bart_model$yhat.test)
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}

bartMachine_fun <- function(X_train, y_train, X_test,y_test,df,k,q){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- bench::mark(bart_model <- bartMachine(
          X = train_X,
          y = y_train,
          k = k,
          nu = df,
          q=q))
        # The value of calculating the time required for modeling
  time <- mean(t$time[[1]])
  predictions <- predict(bart_model,test_X,type = "prob")
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}

SoftBart_fun<- function(X_train, y_train, X_test,y_test,num_trees,alpha,beta){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <-  bench::mark({bart_model <- softbart(X = train_X, Y = y_train, X_test = test_X, hypers
        #print(t)
  time <- mean(t$time[[1]])
  predictions <- bart_model$y_hat_test_mean
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}
```

create evaluation

```
posterior_mse <- function(fit_results,truth_col,estimate_col){
  y_test = fit_results$truth_col
  pred = fit_results$estimate_col
  return(mean((y_test - pred)^2))
}

pred_err <- create_evaluator(
  .eval_fun = posterior_mse, .name = 'Posterior MSE',
  # additional named parameters to pass to .eval_fun()
  truth_col = "y_test", estimate_col = "predictions"
)
```

```
BART <- create_method(
  .method_fun = BART_fun, .name = "BART",
  # additional named parameters to pass to .method_fun()
  k=2.5,q=0.95,df=4
)
dbarts <- create_method(.method_fun = dbarts_fun,.name = "dbarts",
                        k=2.5,q=0.95,df=4)
bartMachine <- create_method(.method_fun = bartMachine_fun,.name = "bartMachine",
                        k=2.5,q=0.95,df=4)
SoftBart <- create_method(.method_fun = SoftBart_fun,.name = "SoftBart",
                        num_trees=50,alpha=0.95,beta=2)
# Create experiment
experiment <- create_experiment(name = "Test Experiment") %>%
  add_dgp(linear_dgp) %>%
  add_method(dbarts) %>%
  add_method(BART) %>%
  add_method(bartMachine) %>%
  add_method(SoftBart) %>%
  add_evaluator(pred_err)



results <- run_experiment(experiment, n_reps = 4, save = TRUE)
```

```
Fitting Test Experiment...

Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.
Warning: Some expressions had a GC in every iteration; so filtering is
disabled.

Saving fit results...
```

```
Fit results saved | time taken: 0.028135 seconds
4 reps completed (totals: 4/4) | time taken: 2.784254 minutes
==============================
Evaluating Test Experiment...

Warning: Unknown or uninitialised column: `truth_col`.

Warning: Unknown or uninitialised column: `estimate_col`.

Evaluation completed | time taken: 0.000032 minutes
Saving eval results...
Eval results saved | time taken: 0.038831 seconds
==============================
No visualizers to visualize. Skipping visualization.
==============================
```

```
# Render automated documentation and view results
#render_docs(experiment)
```

```
results$fit_results
```

```
# A tibble: 16 x 7
   .rep  .dgp_name  .method_name time             mse y_test      predictions
   <chr> <chr>      <chr>        <list>         <dbl> <list>      <list>
 1 1     Linear DGP BART         <bench_tm [1]> 1.17  <dbl [120]> <dbl [120]>
 2 1     Linear DGP SoftBart     <bench_tm [1]> 1.17  <dbl [120]> <dbl [120]>
 3 1     Linear DGP bartMachine  <bench_tm [1]> 1.37  <dbl [120]> <dbl [120]>
 4 1     Linear DGP dbarts       <bench_tm [1]> 1.10  <dbl [120]> <dbl [120]>
 5 2     Linear DGP BART         <bench_tm [1]> 1.57  <dbl [120]> <dbl [120]>
 6 2     Linear DGP SoftBart     <bench_tm [1]> 1.35  <dbl [120]> <dbl [120]>
 7 2     Linear DGP bartMachine  <bench_tm [1]> 1.62  <dbl [120]> <dbl [120]>
 8 2     Linear DGP dbarts       <bench_tm [1]> 1.54  <dbl [120]> <dbl [120]>
 9 3     Linear DGP BART         <bench_tm [1]> 1.14  <dbl [120]> <dbl [120]>
10 3     Linear DGP SoftBart     <bench_tm [1]> 0.986 <dbl [120]> <dbl [120]>
11 3     Linear DGP bartMachine  <bench_tm [1]> 1.17  <dbl [120]> <dbl [120]>
12 3     Linear DGP dbarts       <bench_tm [1]> 1.05  <dbl [120]> <dbl [120]>
13 4     Linear DGP BART         <bench_tm [1]> 1.28  <dbl [120]> <dbl [120]>
14 4     Linear DGP SoftBart     <bench_tm [1]> 1.13  <dbl [120]> <dbl [120]>
15 4     Linear DGP bartMachine  <bench_tm [1]> 1.38  <dbl [120]> <dbl [120]>
16 4     Linear DGP dbarts       <bench_tm [1]> 1.25  <dbl [120]> <dbl [120]>
```

```
gain_summary <- function(repeat_time, results_table){

  result <- data.frame()
  time_BART <- c()
  mse_BART <- c()
  time_dbarts <- c()
  mse_dbarts <- c()
  time_BM <- c()
  mse_BM <- c()
  time_SB <- c()
  mse_SB <- c()

  for (i in 1:repeat_time) {
    if(results_table$.method_name[4*i-3]=="BART" &&
       as.numeric(results_table$.rep)[4*i-3]==i){
      time_BART[i] <- results_table$time[[4*i-3]]
      mse_BART[i] <- results_table$mse[4*i-3]
    }
    if(results_table$.method_name[4*i-2]=="SoftBart"&&
       as.numeric(results_table$.rep)[4*i-2]==i){
      time_SB[i] <- results_table$time[[4*i-2]]
      mse_SB[i] <- results_table$mse[4*i-2]
    }
    if(results_table$.method_name[4*i-1]=="bartMachine"&&
       as.numeric(results_table$.rep)[4*i-1]==i){
      time_BM[i] <- results_table$time[[4*i-1]]
      mse_BM[i] <- results_table$mse[4*i-1]
    }
    if(results_table$.method_name[4*i]=="dbarts"&&
       as.numeric(results_table$.rep)[4*i]==i){
      time_dbarts[i] <- results_table$time[[4*i]]
      mse_dbarts[i] <- results_table$mse[4*i]
    }
  }
  row_BART <- data.frame(MSE_mean = mean(mse_BART), MSE_se = sd(mse_BART),
                         running_time_mean = mean(time_BART),
                         running_time_sd = sd(time_BART),
                         package_name = "BART")
  row_SB <- data.frame(MSE_mean = mean(mse_SB), MSE_se = sd(mse_SB),
                         running_time_mean = mean(time_SB),
                         running_time_sd = sd(time_SB),
                         package_name = "SoftBart")
  row_BM <- data.frame(MSE_mean = mean(mse_BM), MSE_se = sd(mse_BM),
                       running_time_mean = mean(time_BM),
                       running_time_sd = sd(time_BM),
                       package_name = "bartMachine")
  row_dbarts <- data.frame(MSE_mean = mean(mse_dbarts), MSE_se = sd(mse_dbarts),
                       running_time_mean = mean(time_dbarts),
                       running_time_sd = sd(time_dbarts),
                       package_name = "dbarts")
  result <- rbind(result,row_BART)
  result <- rbind(result,row_SB)
  result <- rbind(result,row_BM)
  result <- rbind(result,row_dbarts)
  return(result)

}
```

```
summary <- gain_summary(repeat_time = 4,results_table = results$fit_results)
summary
```

```
  MSE_mean    MSE_se running_time_mean running_time_sd package_name
1 1.292466 0.1978935         2.5754682      0.05370103         BART
2 1.159817 0.1508278        15.0719108      0.21501656     SoftBart
3 1.383874 0.1846658         0.7956670      0.03956696  bartMachine
4 1.235827 0.2175892         0.6440335      0.05145436       dbarts
```

```
BART_fun <- function(X_train, y_train, X_test, y_test, num_trees,alpha,beta) {
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- system.time(fit <- wbart(x.train = train_X,
                                y.train = y_train,
                                x.test = test_X,
                                ntree = num_trees,
                                base = alpha,
                                power = beta))
  time <- as.numeric(t["elapsed"])
  predictions <- colMeans(fit$yhat.test)
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}
```

```r
dbarts_fun <- function(X_train, y_train, X_test, y_test, num_trees,alpha,beta){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- system.time(bart_model <- bart(x.train = train_X,
                                      y.train = y_train,
                                      x.test = test_X,
                                      ntree = num_trees,
                                      base = alpha,
                                      power = beta))
  time <- as.numeric(t["elapsed"])
  predictions <- colMeans(bart_model$yhat.test)
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}

bartMachine_fun <- function(X_train, y_train, X_test,y_test,num_trees,alpha,beta){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  bart_model <- bartMachine(
          X = train_X,
          y = y_train,
          num_trees = num_trees,
          beta = beta,
          alpha = alpha

        )
        # The value of calculating the time required for modeling
  time <- bart_model$time_to_build
  predictions <- predict(bart_model,test_X,type = "prob")
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}

SoftBart_fun<- function(X_train, y_train, X_test,y_test,num_trees,alpha,beta){
  train_X <- data.frame(X_train)
  test_X <- data.frame(X_test)
  t <- system.time({bart_model <- softbart(X = train_X, Y = y_train, X_test = test_X, hypers
        #print(t)
  time <- as.numeric(t["elapsed"])
  predictions <- bart_model$y_hat_test_mean
  mse_score <- mean((y_test - predictions)^2)

  return(list(time = time, mse=mse_score,y_test=y_test,predictions=predictions))
}
```

```r
BART <- create_method(
  .method_fun = BART_fun, .name = "BART",
  # additional named parameters to pass to .method_fun()
  num_trees=50,alpha=0.95,beta=2
)
dbarts <- create_method(.method_fun = dbarts_fun,.name = "dbarts",
                        num_trees=50,alpha=0.95,beta=2)
bartMachine <- create_method(.method_fun = bartMachine_fun,.name = "bartMachine",
                        num_trees=50,alpha=0.95,beta=2)
SoftBart <- create_method(.method_fun = SoftBart_fun,.name = "SoftBart",
                        num_trees=50,alpha=0.95,beta=2)
# Create experiment
experiment <- create_experiment(name = "Test Experiment") %>%
  add_dgp(linear_dgp) %>%
  add_method(dbarts) %>%
  add_method(BART) %>%
  add_method(bartMachine) %>%
  add_method(SoftBart) %>%
  add_evaluator(pred_err)



results <- run_experiment(experiment, n_reps = 4, save = TRUE)
```

```
Fitting Test Experiment...
Saving fit results...
Fit results saved | time taken: 0.028361 seconds
4 reps completed (totals: 4/4) | time taken: 1.225304 minutes
==============================
Evaluating Test Experiment...

Warning: Unknown or uninitialised column: `truth_col`.

Warning: Unknown or uninitialised column: `estimate_col`.
```

```
Evaluation completed | time taken: 0.000048 minutes
Saving eval results...
Eval results saved | time taken: 0.025764 seconds
==============================
No visualizers to visualize. Skipping visualization.
==============================
```

```
# Render automated documentation and view results
#render_docs(experiment)
```