mrstudyr

Colton J. McCurdy mccurdyc@allegheny.edu

Installing mrstudyr Package from GitHub

This code will install the mrstudyr package from GitHub using the `install_github` function.

```
devtools::install_github("mccurdyc/mrstudyr")
```

```
## Skipping install of 'mrstudyr' from a github remote, the SHA1 (6b376bee) has not changed since last i
##   Use `force = TRUE` to force installation
```

Initialize the System

First, load in the libraries that are used in addition to those with the mrstudyr package (i.e., load all of the packages not used by mrstudyr but still used in this RMarkdown file). Note that right now the mrstudyr package will automatically load all of the packages that it needs to performs its various analyses. Now, we are ready to call the functions from the mrstudyr package and produce the appropriate summary tables and graphs.

```
suppressPackageStartupMessages(library(mrstudyr))
suppressPackageStartupMessages(library(knitr))
```

Comparing Mutant Reduction Techniques

Show the schemas used in this study

```
sqlite <- read_sqlite_avmdefaults() %>% collect_normal_data()
```

```
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
```

```
schemas <- sqlite %>% select_all_schemas()
knitr::kable(schemas, format="latex")
```

schema
ArtistSimilarity
ArtistTerm
BankAccount
BookTown
BrowserCookies
Cloc
CoffeeOrders
CustomerOrder
DellStore
Employee
Examination
Flights
FrenchTowns
Inventory
Iso3166
IsoFlav_R2
JWhoisServer
MozillaExtensions
MozillaPermissions
NistDML181
NistDML182
NistDML183
NistWeather
NistXTS748
NistXTS749
Person
Products
RiskIt
StackOverflow
StudentResidence
UnixUsage
Usda
WordNet
iTrust

Visualize summary graphs of data before performing reduction

The summary graphs before performing reduction include

- fractional operator costs (per dbms)
- fractional operator costs (per dbms, per schema)
- fractional operator frequencies (per dbms) (e.g., this operator accounts for 20% of total mutants)
- fractional operator frequencies (per dbms, per schema)
- original mutation score (per dbms)
- original mutation score (per dbms, per schema)

Visualize the correlation and cost reduction between reduced sets from performing random sampling

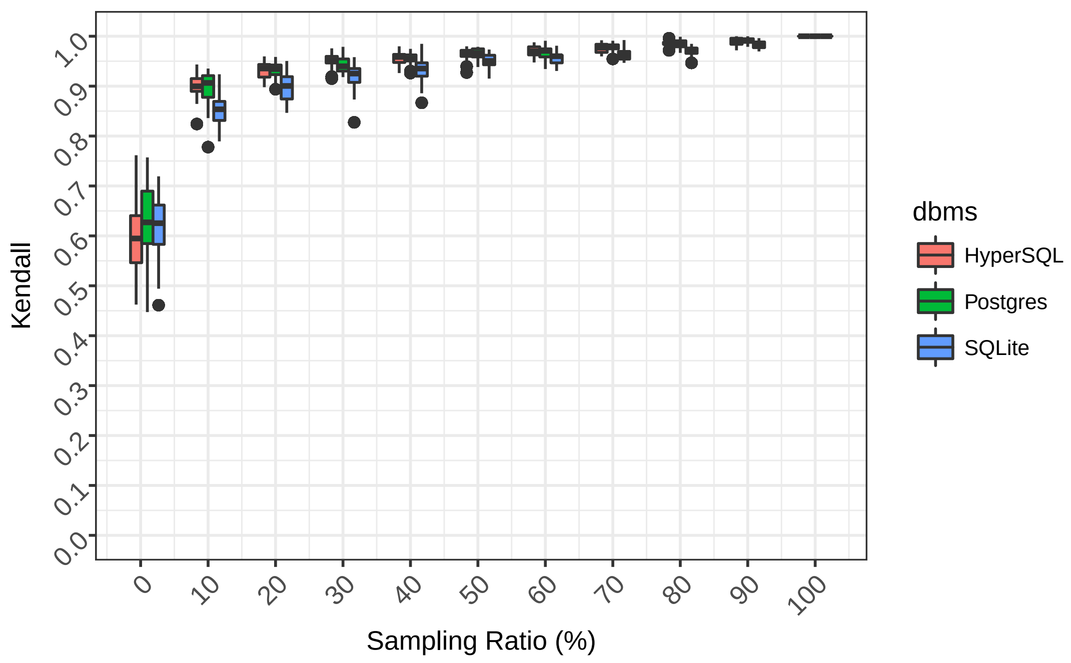
This will perform random sampling for all DBMSs where the technique outlined later will only display the data collected for SQLite.

```r
rs <- create_random_sampling_graphs()
```

```
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
```

```
## [1] "RANDOM SAMPLING: Currently analyzing x = 1 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 10 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 20 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 30 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 40 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 50 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 60 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 70 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 80 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 90 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 100 percent ..."
```

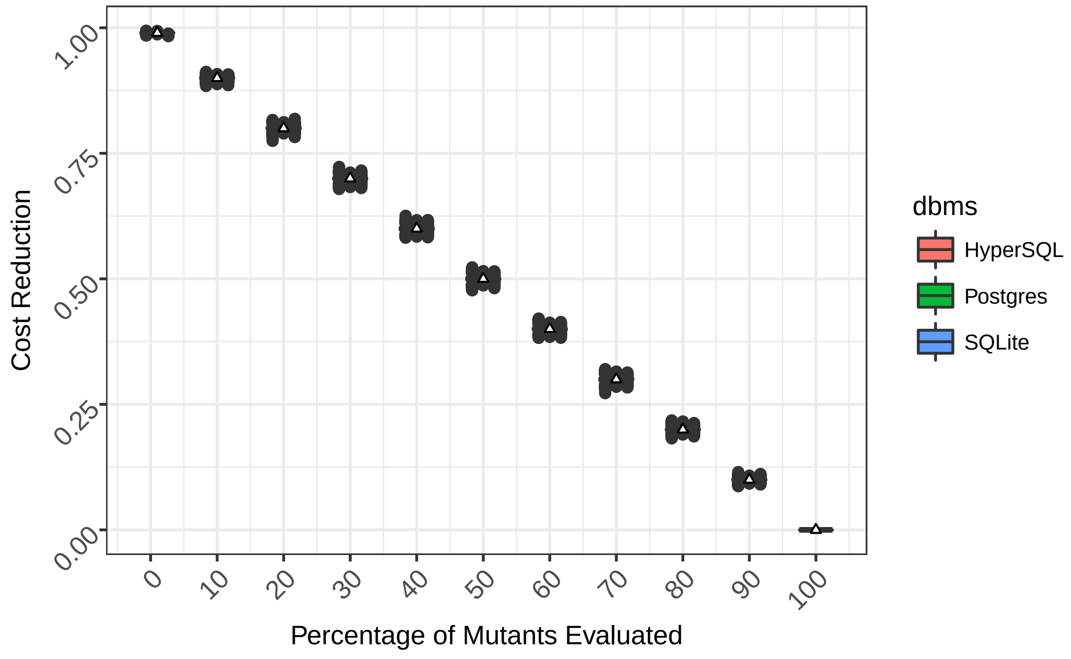
```r
visualize_plot_percentage_correlation(rs)
```



```r
rs <- create_random_sampling_graphs()
```

```
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
## [1] "RANDOM SAMPLING: Currently analyzing x = 1 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 10 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 20 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 30 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 40 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 50 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 60 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 70 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 80 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 90 percent ..."
## [1] "RANDOM SAMPLING: Currently analyzing x = 100 percent ..."
```

```
visualize_plot_percentage_cost_reduction(rs)
```



Visualize correlation between reduced and original mutation score generated by hill climbing

NOTE: Since the hill climbing technique takes long, I provide my data as a feather

To install and load the feather tool:

```
install.packages('feather')
```

```
## Installing package into '/home/mccurdyc/R/x86_64-pc-linux-gnu-library/3.3'
## (as 'lib' is unspecified)
```

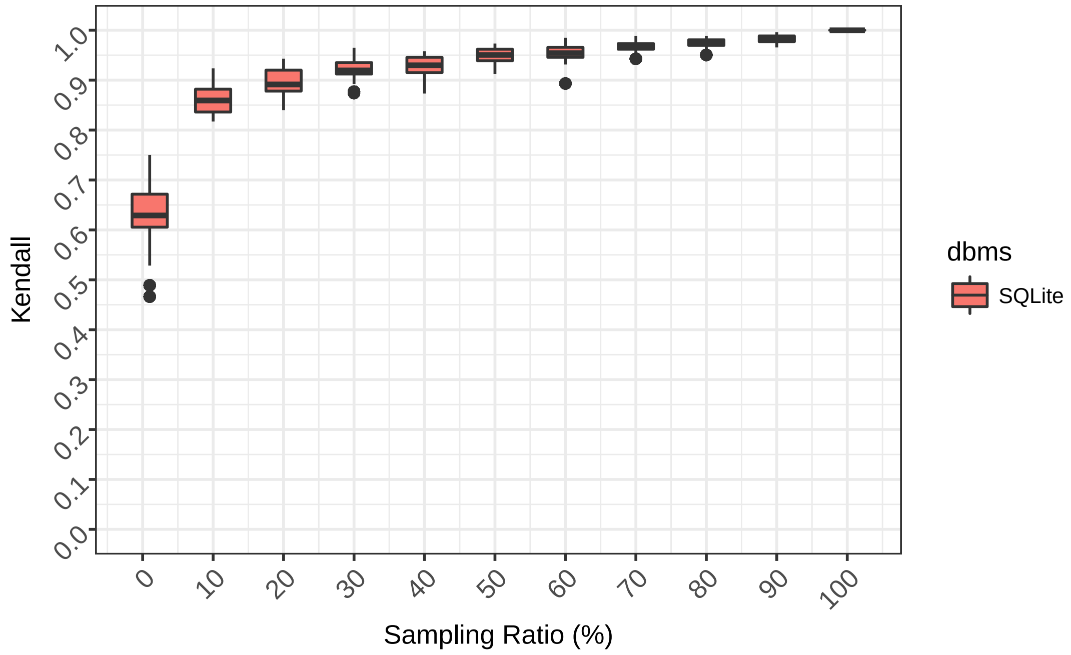
```
library(feather)
```

To read the data from performing all reduction techniques

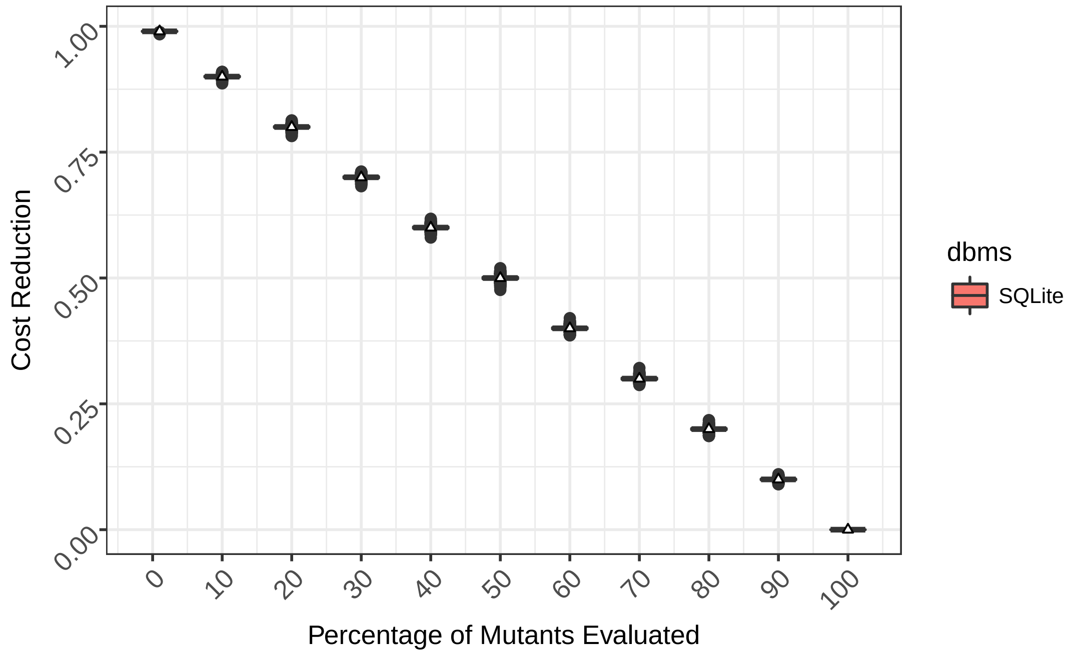
```
data <- read_feather("feathers/combined_technique_data.feather")
```

Now, you can create the random sampling, hill climbing, selective and selective random sampling plots using the data that I collected.

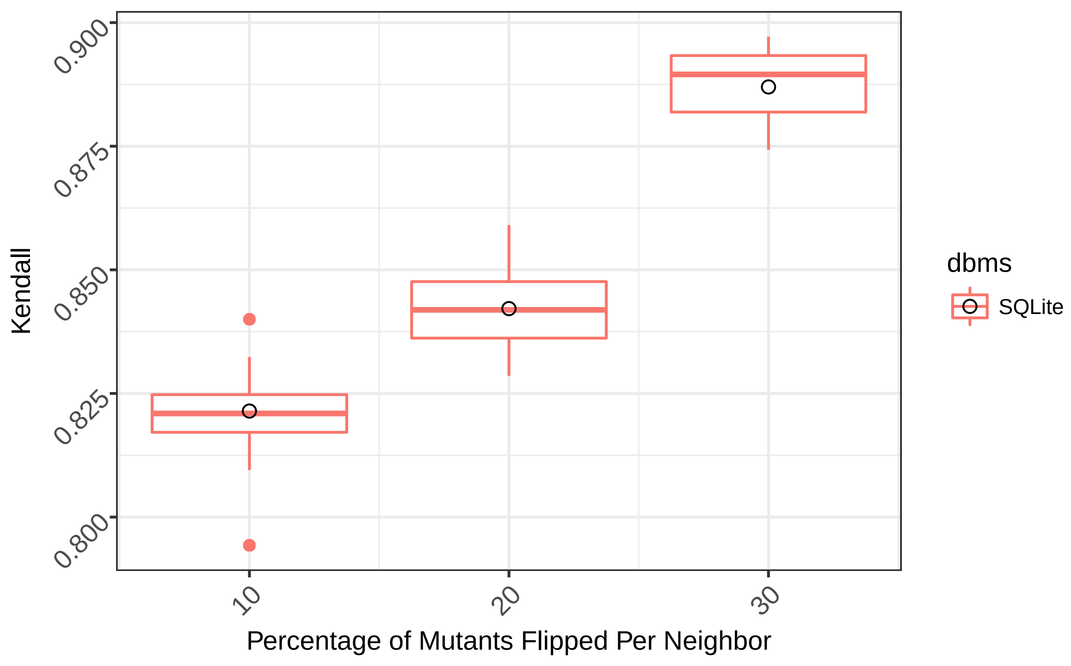
```r
data %>% dplyr::filter(technique_group == 'RS') %>% visualize_plot_percentage_correlation()
```



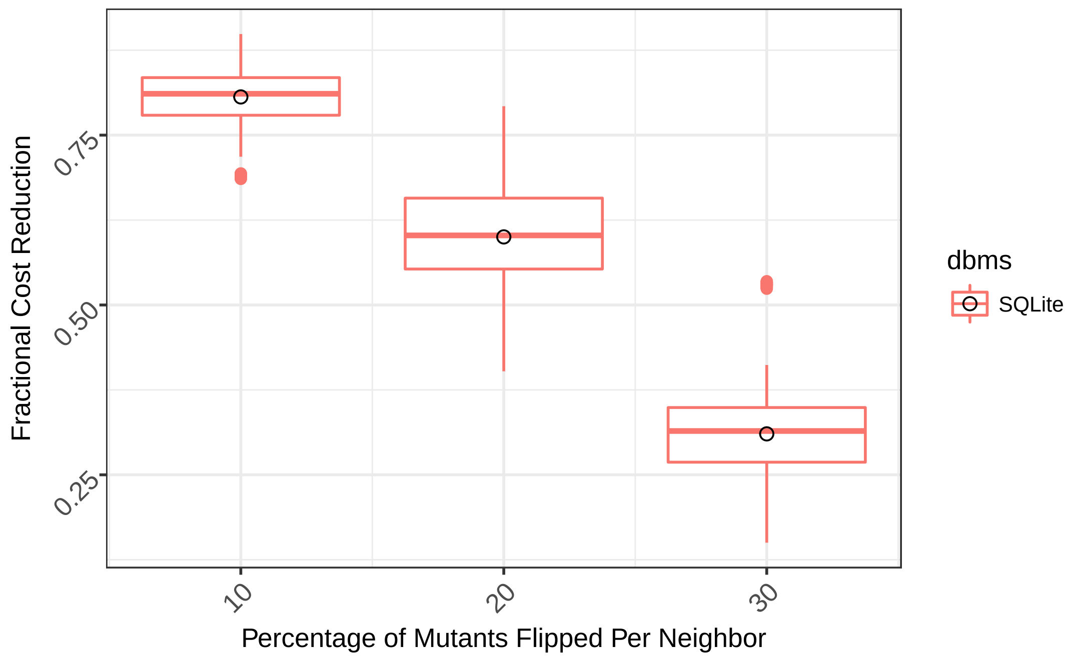
```r
data %>% dplyr::filter(technique_group == 'RS') %>% visualize_plot_percentage_cost_reduction()
```



```r
data %>% dplyr::filter(technique_group == 'HC') %>% visualize_plot_hill_climbing_correlation()
```



```
data %>% dplyr::filter(technique_group == 'HC') %>% visualize_plot_hill_climbing_cost_reduction()
```



You will notice if you do this, you will only have the data from SQLite, this is because this is the DBMSs I used to generate the hill climbing model. In the future, we would like to also generate models from HyperSQL and PostgreSQL.

Apply hill climbing model to other DBMSs

First, you need to read in the generated model using the feather package again. For this example, I will only read in the small (generated from a more granular step size) model. Then, you will need to make sure you have the data for a DBMS read in. In this example, I use the HyperSQL DBMS because it is one of the DBMSs not used to generate the model.

```r
small <- feather::read_feather("feathers/small_model.feather")
hypersql <- read_hypersql_avmdefaults() %>% collect_normal_data()
```

```
## Parsed with column specification:
## cols(
##   identifier = col_character(),
##   dbms = col_character(),
##   schema = col_character(),
##   operator = col_character(),
##   type = col_character(),
##   killed = col_character(),
##   time = col_integer()
## )
```

```r
apply_operator_model(hypersql, small)
```

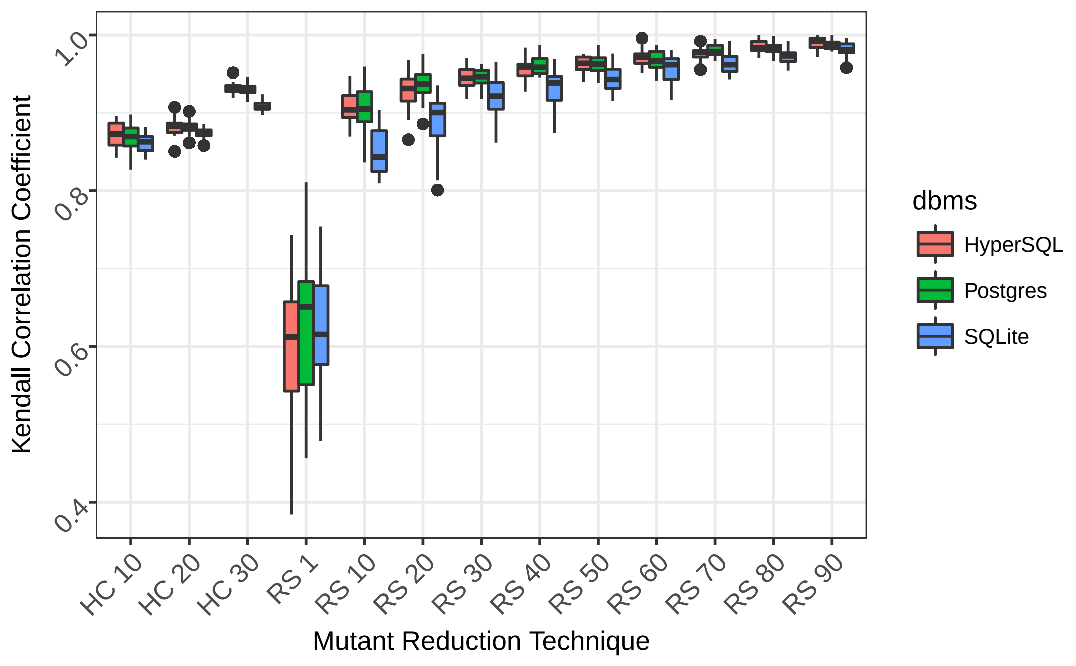
```
## Source: local data frame [34 x 12]
## Groups: dbms, schema [34]
##
##        dbms           schema reduced_numerator reduced_denominator
##       <chr>            <chr>             <int>               <int>
## 1  HyperSQL ArtistSimilarity                41                  41
## 2  HyperSQL       ArtistTerm               123                 123
## 3  HyperSQL      BankAccount               162                 195
## 4  HyperSQL         BookTown              1487                1525
## 5  HyperSQL   BrowserCookies               351                 382
## 6  HyperSQL             Cloc               223                 223
## 7  HyperSQL     CoffeeOrders               309                 309
## 8  HyperSQL    CustomerOrder               489                 534
## 9  HyperSQL        DellStore               855                 855
## 10 HyperSQL         Employee               222                 229
## # ... with 24 more rows, and 8 more variables: original_numerator <int>,
## #   original_denominator <int>, reduced_time <dbl>, original_time <dbl>,
## #   cost_reduction <dbl>, reduced_mutation_score <dbl>,
## #   original_mutation_score <dbl>, error <dbl>
```

Alternatively, you could use the data that I collected from applying the models to other DBMSs and step sizes by reading in the feather file.

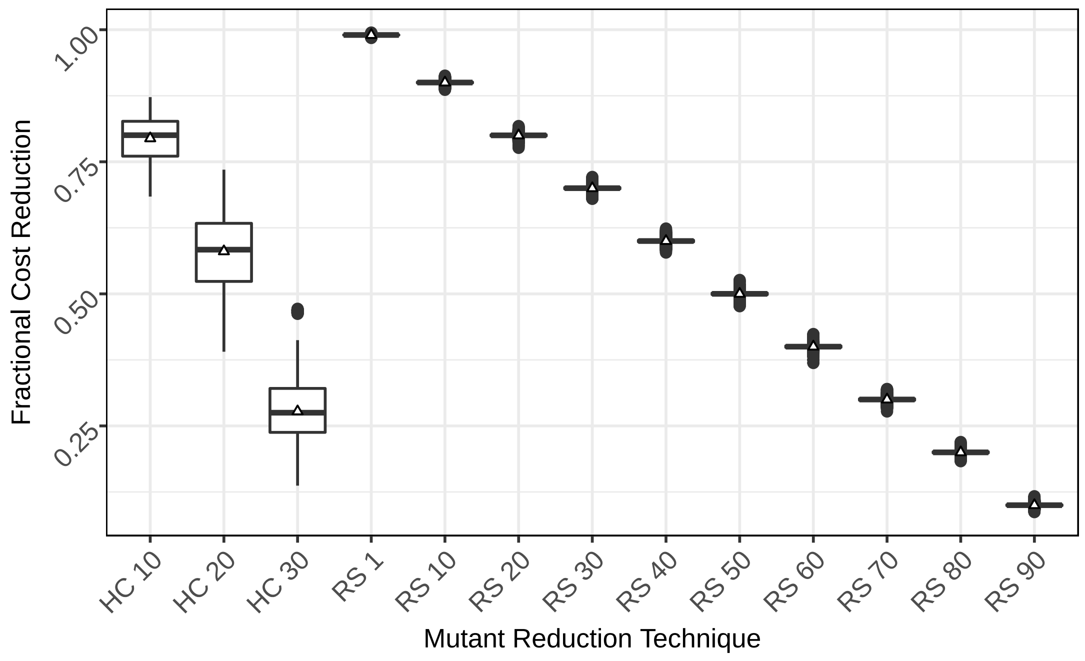
```r
all_dbms_joined_technique_data <- feather::read_feather("feathers/all_dbms_joined_technique_data.feathe
```

Now, you should be able to produce graphs comparing random sampling and hill climbing correlation and cost reduction using the following.

```r
all_dbms_joined_technique_data %>% visualize_plot_correlation_all_reduction_techniques_box()
```

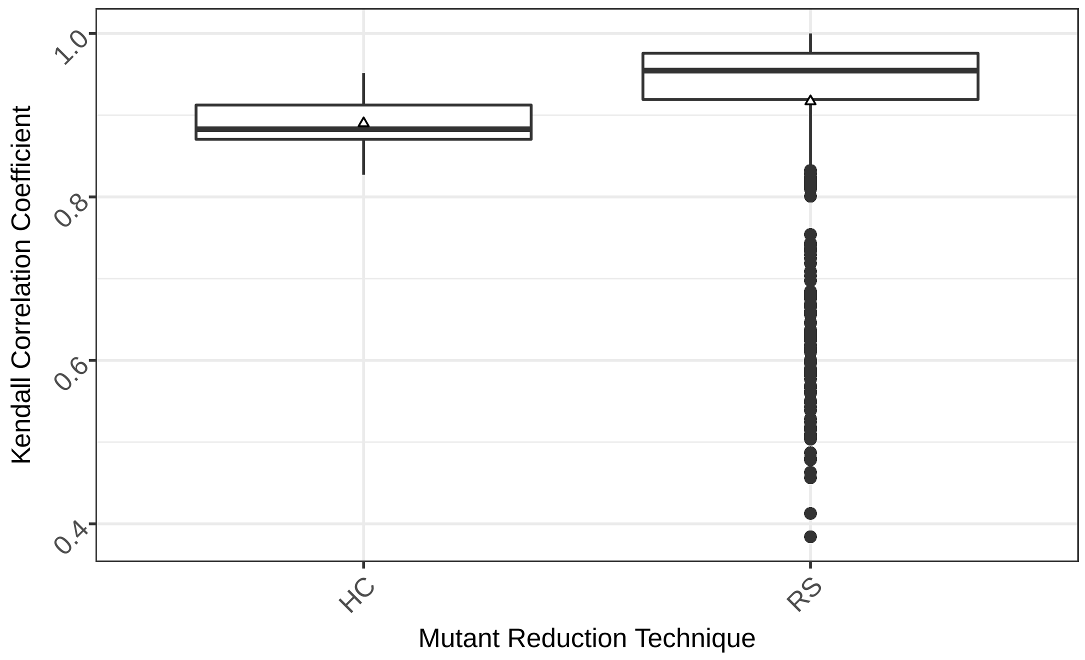


```
all_dbms_joined_technique_data %>% visualize_plot_cost_reduction_all_reduction_techniques_box()
```

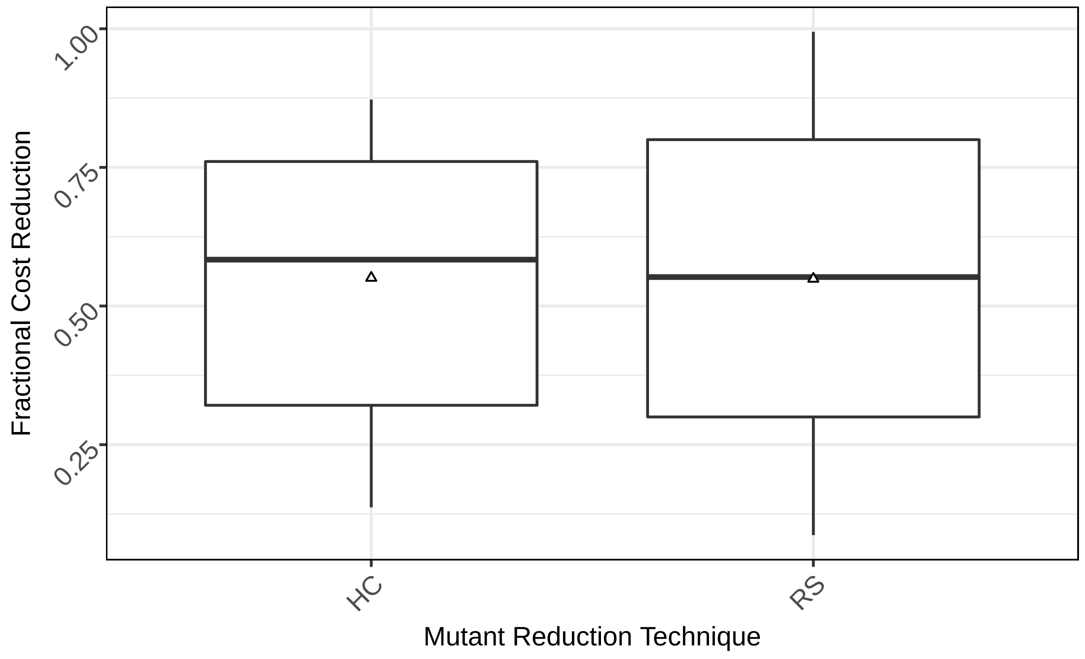


If you are interested to compare the techniques at a higher level, you can compare the techniques by reduction technique group (e.g., random sampling versus hill climbing instead of at the configuration level).

```
all_dbms_joined_technique_data %>% visualize_plot_correlation_all_groups()
```



```
all_dbms_joined_technique_data %>% visualize_plot_cost_reduction_all_groups()
```



Perform a statistical analysis on the correlation reduced and original mutation scores

```
perform_wilcoxon_accurate(all_dbms_joined_technique_data, "correlation")
```

Perform a statistical analysis on the cost reduction

```
perform_wilcoxon_accurate(all_dbms_joined_technique_data, "cost reduction")
```

Perform a head-to-head effect size calculation comparing the correlation of the reduced and original mutation scores

```
perform_effectsize_accurate(all_dbms_joined_technique_data, "correlation")
```

Perform a head-to-head effect size calculation comparing the cost reduction

```
perform_effectsize_accurate(all_dbms_joined_technique_data, "cost reduction")
```