Essential Regression Vignette - pipelineER1 and pipelineER2

Contents

Please make sure you have read the vignette for `plainER` and `parseRun` before continuing with this tutorial as many of the section depend upon information included in that introductory file. This vignette uses the same data set:

X, available for download here.

Y, available for download here.

Housekeeping

Before beginning, please ensure that the following packages are installed and libraries are loaded. The Essential Regression package is available for downloading using the following code:

```
devtools::install_github("Hanxi-002/EssReg", auth_token = "TOKEN_HERE")
```

The other packages can be installed from CRAN.

```
library(matrixcalc)
library(readr)
library(ROCR)
library(e1071)
library(dplyr)
library(doParallel)
library(foreach)
library(scales)
library(doRNG)
library(matlib)
library(BAS)
library(gmp)
library(EssReg)
```

We also recommend running some of these function in parallel. We prepare the environment for this using the following code:

```
cores <- detectCores() ## detect available number of cores, can be set manually as well
registerDoParallel(cores)
```

Pipeline The main contribution of this package is a pipeline for the estimation of the hyperparameters of Essential Regression, δ and λ. The pipeline consists of four steps, which are split into two functions: `pipelineER1()` (steps 1 and 2) and `pipelineER2()` (steps 3 and 4). There is also an optional `pipelineER3()` (step 5) for evaluation of Essential Regression performance in comparison to other popular regression methods. For information on the algorithm, please review the main EssRegVignette.

`pipelineER1()` and `pipelineER2()` support the specification of which steps to run using the parameter `steps`. This parameter can be set to the specific step number to perform (i.e. `1` for `pipelineER1()`) or `all` to perform all steps. For the sake of this example, we run all steps.

Running `pipelineER1()` Both pipeline functions require `steps` and a path to a .yaml file (`yaml_path`) that contains all of the other parameter values/arguments. We organize the functions in this way to facilitate scripting.

An example `.yaml` file for `pipelineER1()` that uses the `x` and `y` downloaded above is available here Its format is as follows:

```
---
x_path: x.csv         # path to .csv file for data matrix
y_path: y.csv         # path to .csv file for response vector
out_path: pipeline/   # path to directory for saving results
k: 5                  # number of folds for cross-validation
y_factor: FALSE       # is y categorical?
y_levels: NULL         # if y is ordinal, provide the levels
eval_type: "corr"     # evaluation metric for method comparison
permute: TRUE         # permute y during comparison?
rep_cv: 50            # number of replicates for cross-validation for delta
nreps: 10             # number of replicates for cross-validation in step 2
alpha_level: 0.05     # alpha level for confidence intervals
thresh_fdr: 0.2       # false discovery rate thresholding p-value cutoff
```

We refer the user to the introductory vignette for information on most of these arguments and cover the ones specific to the pipeline here.

- `eval_type`: A string indicating the type of evaluation metric to use for the method comparison. Can be "corr" or "auc". REQUIRED
- `nreps`: An integer for the number of replicates for supervised cross-validation. This cross-validation is based upon spearman correlation (if indicated in `sel_corr`), AUC (if `y` is a factor), or MSE (if `y` is continuous). REQUIRED
- `permute`: A boolean flag indicating whether to permute the response as a baseline comparison. Only LASSO and plainER will be performed with the permuted responses. REQUIRED

Note that it is important that `x_path` and `y_path` must be correctly specified. For now, we leave them as `x.csv` and `y.csv` and assume the user has changed their working directory to the directory containing these files. Paths are very finicky so please be careful when specifying them!!!

We run steps 1 and 2 of the pipeline with:

```
pipelineER1(yaml_path = "path/to/yaml/file", ## change path
            steps = "all")
```

Output `pipelineER1()` saves the output of each step as an `.rds` file in the directory specified by `out_path`; however, the main output of interest are boxpots comparing various values of δ and models according to an evaluation metric (correlation, AUC, MSE). From our run above, we get a δ boxplot comparing different magnitudes of δ. This is saved to the output directory as `delta_selection_boxplot.pdf`. We also get a series of boxplots comparing a variety of methods for the best values of δ chosen by Step 1. For our run, we compared Essential Regression with LASSO.

The complete output found in `out_path` is below. Depending upon which steps you run, you may only have a subset of the following:

- 4 subdirectories for each of the 4 different magnitudes of `delta` tested: `delta_0.1/`, `delta_0.01/`, `delta_0.001/`, and `delta_1e-04`. These each contain two heatmaps: one of the sample correlation matrix and one of the thresholded sample correlation matrix.
- 4 boxplots of the method comparison results for each of the 4 different `delta` magnitudes. The names of these files will vary, but will follow the form `delta_[VALUE]_boxplot.pdf` where `[VALUE]` is the value of `delta` selected within each of the coarse searches. These are shown below. Fully opaque boxes show the method results using the data, and transparent boxes show the results using permuted data. Color indicates the method used.
- 4 `.rds` files with the results of the method comparisons used for the creation of the 4 boxplots. These will be named `essregCV_delta_[VALUE].rds` where `[VALUE]` is the value of `delta` selected within each of the coarse searches. These `.rds` files each contain 1 data frame with two columns: the first corresponding to the method used and the second corresponding to the evaluation metric (AUC, MSE, or Spearman correlation).
- 2 `.rds` files with the pipeline results: `pipeline_step1.rds` and `pipeline_step2.rds`. `pipeline_step1.rds` is a list of four lists where each sublist contains the results of `plainER()` using a specific value of `delta` (all 4 will be different and of different magnitudes). `pipeline_step2.rds` is a list of cross-validation results where each item in the list is, itself, a list: the first item in the sublist is the value of `delta` used and the second item is a data frame with 2 columns (the method and the evaluation metric).
- 1 final boxplot: `delta_selection_boxplot.pdf`, which shows a comparison between different magnitudes of `delta`. The user is expected to select one value from the four shown to use in `pipelineER2()`. Color indicates the permutation of the data. We display this in the next section.

For categorical `y`, the output directory would also contain the mapping from categorical to continuous values.

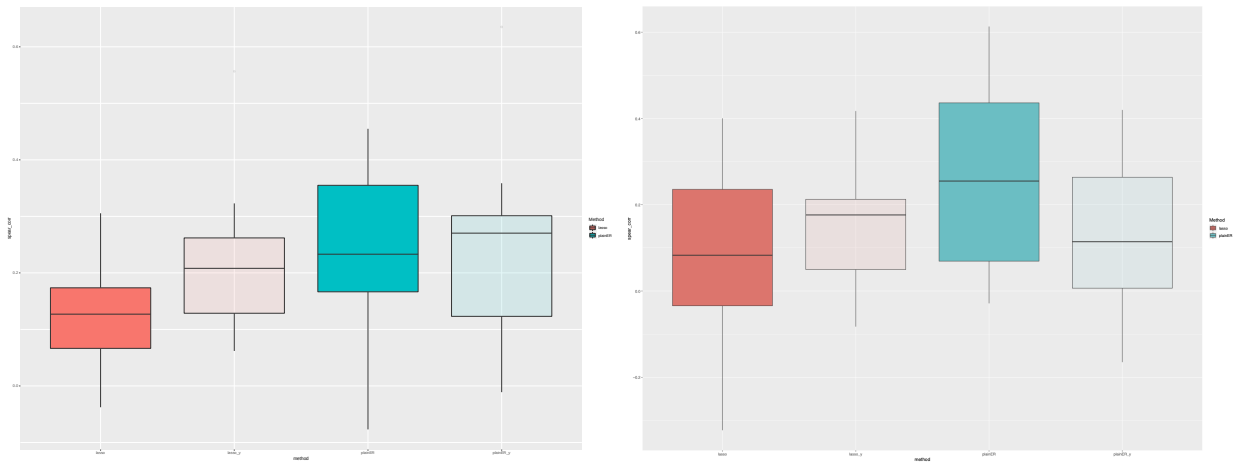

Figure 1: Delta = 1e-04, 1e-03 Boxplots

Reading The Results We use this boxplot to select the final value for `delta` to use in `pipelineER2()`. Assuming that permutation testing was conducted, we are looking for a value that has good separation between the original and permuted data boxes (or at least their medians). We also want to see (if we can), a smaller interquartile range (squished boxes). Here, we choose `delta = 0.07` as the best value. Sometimes the results are inconclusive. In this case, we recommend selecting any of the best values and continuing with the pipeline.

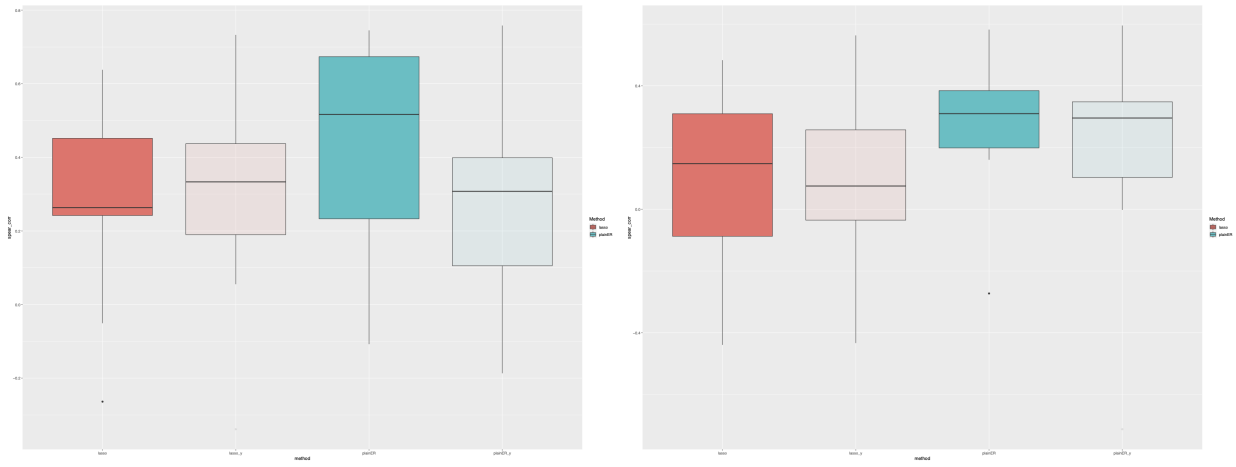

Figure 2: Delta = 0.07, 0.1 Boxplots

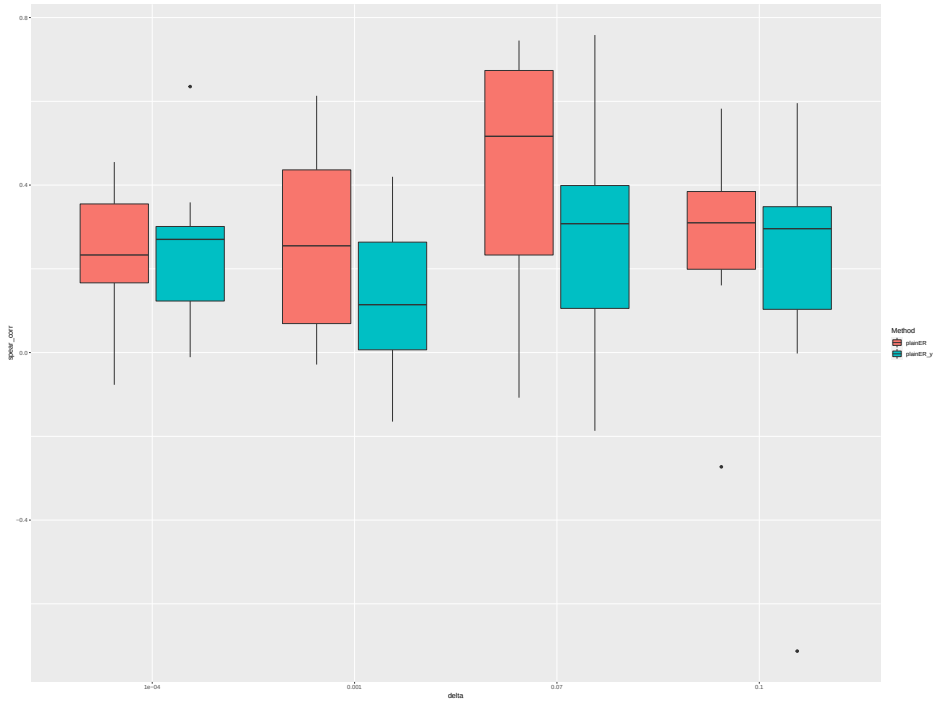

Figure 3: Delta Selection Boxplot

Running `pipelineER2()` The `.yaml` file used in `pipelineER2()` is very similar to the one used in `pipelineER1()`. A copy of the one used for this tutorial can be downloaded here

```
---
x_path: x.csv                # path to .csv file for data matrix
y_path: y.csv                # path to .csv file for response vector
out_path: pipeline2/         # path to directory for saving results
k: 4                         # number of folds for cross-validation
y_factor: FALSE              # is y categorical?
y_levels: NULL                # if y is ordinal, provide the levels
rep_cv: 5                    # number of replicates for non-supervised cross-validation
nreps: 3                     # number of replicates for supervised cross-validation
alpha_level: 0.05            # alpha level for confidence intervals
thresh_fdr: 0.2              # false discovery rate thresholding p-value cutoff
eval_type: "corr"            # type of evaluation metric for model comparisons
permute: TRUE                # permute y as baseline comparison?
lambda: [0.1, 0.5, 1, 1.5] # lambda, should be multiple values for step 4
delta: 0.07                  # the best delta chosen from step 1 and step 2.
```

All of these arguments are the same as in `pipelineER1()`; we just have two additional ones.

- `lambda`: A vector containing multiple values for comparison. We recommend supplying three or four different values between 0.01 and 2. The vector should be specified with square brackets, as shown above.
- `delta`: A single value of `delta`, preferably chosen by the user according to the results of `pipelineER1()`, or a vector of values. If a single value is supplied, then a fine grid centered at the provided value will be constructed. This grid is used to select the overall optimal value of `delta`. If a vector is supplied, no grid is constructed, and the vector is used for the final search instead.

We run steps 3 and 4 of the pipeline with:

```
pipelineER2(yaml_path = "path/to/yaml/file", ## change path
            steps = "all")
```

Output `pipelineER2()` has very similar output to `pipelineER1()`:

- 1 subdirectory for the overall optimal value of `delta`, containing two heatmaps: one of the sample correlation matrix and one of the thresholded sample correlation matrix. Note that this may not match the `.yaml` file because of the final search for the optimal `delta` value.
- Boxplots of the method comparison results for each of values of `lambda` supplied by the `.yaml` file (number will vary). The names of these files will also vary, but will follow the form `lambda_[VALUE]_boxplot.pdf` where `[VALUE]` is the `lambda` used by the methods in the cross-validation. These are shown below. Fully opaque boxes show the method results using the data, and transparent boxes show the results using permuted data. Color indicates the method used.
- `.rds` files with the results of the method comparisons used for the creation of the boxplots. These will be named `essregCV_lambda_[VALUE].rds` where `[VALUE]` is the value of `lambda` used. These `.rds` files each contain 1 data frame with two columns: the first corresponding to the method used and the second corresponding to the evaluation metric (AUC, MSE, or Spearman correlation).
- 2 `.rds` files with the pipeline results: `pipeline_step3.rds` and `pipeline_step4.rds`. `pipeline_step3.rds` are the results of `plainER()` using the final optimal value of `delta`. `pipeline_step4.rds` is a list of cross-validation results where each item in the list is, itself, a list: the first item in the sublist is the value of `lambda` used and the second item is a data frame with 2 columns (the method and the evaluation metric).
- 1 final boxplot: `lambda_selection_boxplot.pdf`, which shows a comparison between different magnitudes of `lambda`. Color indicates the permutation type. We display this in the next section.

For categorical `y`, the output directory would also contain the mapping from categorical to continuous values.

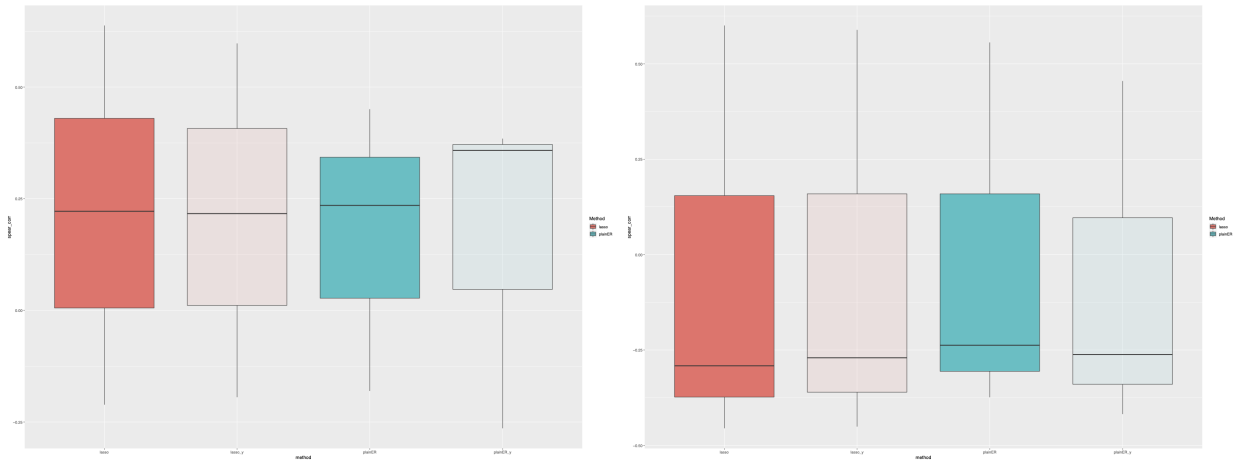

Figure 4: Lambda = 0.1, 0.5 Boxplots

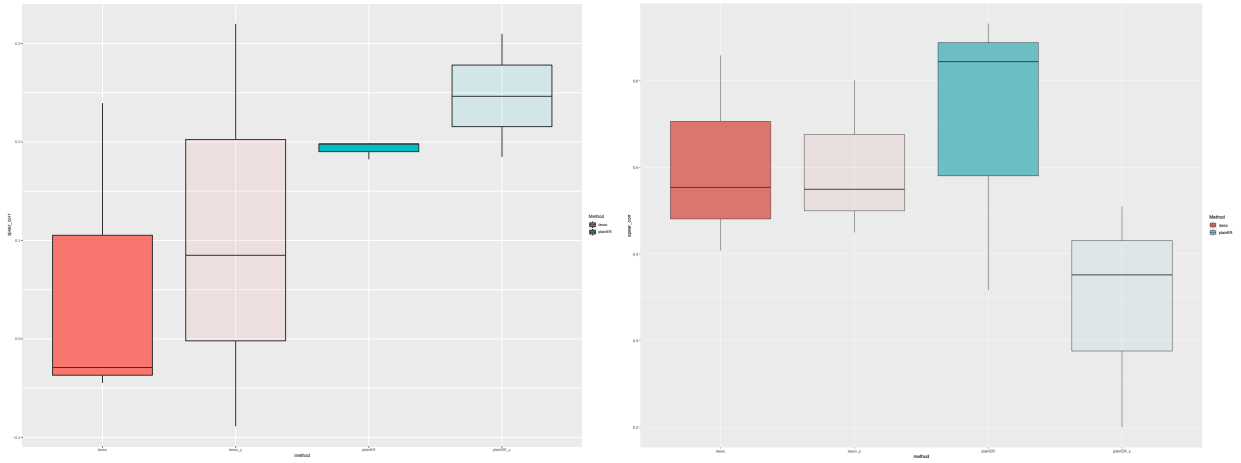

Figure 5: Lambda = 1, 1.5 Boxplots

Reading The Results Assuming that permutation testing was conducted, we are looking for a value that has good separation between the original and permuted data boxes (or at least their medians). We also want to see (if we can), a smaller interquartile range (squished boxes). Here, we choose `lambda = 1.5` as the best value. Sometimes the results are inconclusive. In this case, we recommend selecting any of the best values and continuing with the pipeline.

To get the value of `delta`, we need to read `pipeline_step3.rds`:

```
step3 <- readRDS("path/to/pipeline_step3.rds") ## change path
```

We see that the final optimal value of `delta` is 0.0609.

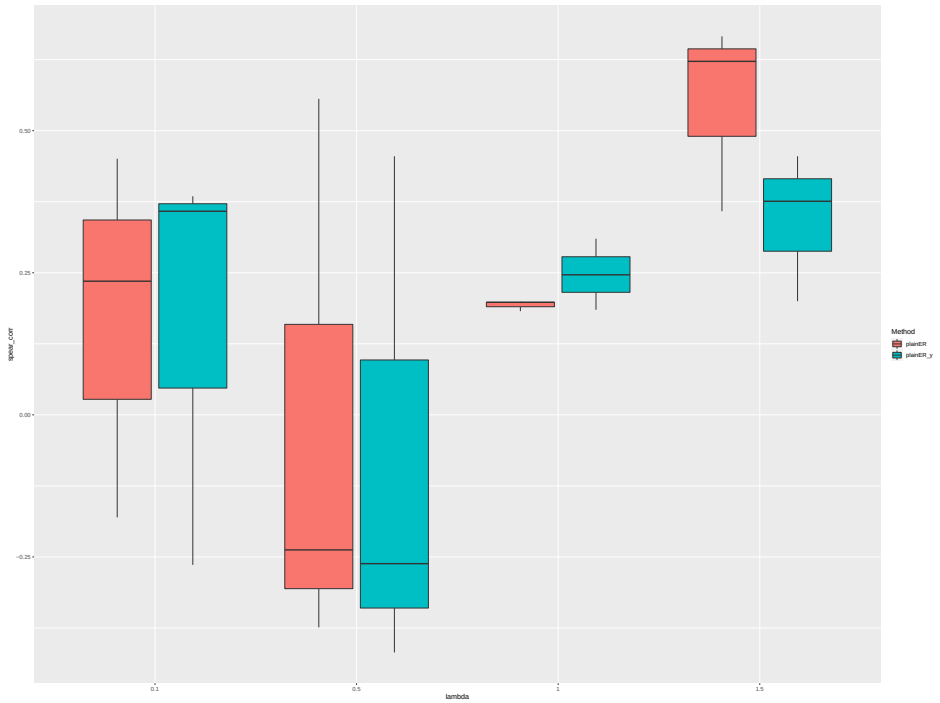

Figure 6: Lambda Selection Boxplot

Running `pipelineER3()` (optional) This step is optional and is run to get a better idea of the performance of Essential Regression compared to other methods. Alternatively, you can use the results of `pipelineER2()` to inform a call to `parseRun()`.

The `.yaml` file used in `pipelineER3()` is very similar to the one used in `pipelineER2()`. This `.yaml` can be downloaded here.

```
---
x_path: x.csv            # path to .csv file for data matrix
y_path: y.csv            # path to .csv file for response vector
out_path: pipeline3/     # path to directory for saving results
k: 4                     # number of folds for cross-validation
y_factor: FALSE          # is y categorical?
y_levels: NULL            # if y is ordinal, provide the levels
rep_cv: 5                # number of replicates for non-supervised cross-validation
nreps: 3                 # number of replicates for supervised cross-validation
alpha_level: 0.05        # alpha level for confidence intervals
thresh_fdr: 0.2          # false discovery rate thresholding p-value cutoff
eval_type: "corr"           # type of evaluation metric for model comparisons
permute: TRUE               # permute y as baseline comparison?
lambda: 1.5              # lambda, should be multiple values for step 4
delta: 0.0609            # the best delta chosen from step 1 and step 2.
```

All of these arguments are the same as in `pipelineER2()` except for `delta` and `lambda`. For these two arguments, we supply one, **single** value. We also elect to compare multiple different methods rather than just Essential Regression and LASSO.

We run step 5 of the pipeline with:

```
pipelineER3(yaml_path = "path/to/yaml/file") ## change path
```

Output `pipelineER3()` has less output than the other two pipeline functions:

- 1 subdirectory for the value of `delta` supplied by the `.yaml`, containing two heatmaps: one of the sample correlation matrix and one of the thresholded sample correlation matrix.
- 2 `.rds` files: `pipeline_step5.rds`, which is a data frame with 2 columns (the method and the evaluation metric), and `final_delta_[VALUE]_lambda_[VALUE].rds`, which contains the results from a final run of `plainER()` using the optimal values for `lambda` and `delta` provided in the `.yaml` file.
- 1 boxplot: `opt_delta_lambda_boxplot.pdf`, which shows the cross-validation performance of the comparison methods specified in the `.yaml` against Essential Regression. Color indicates the method, and opacity indicates permutation type.

For categorical `y`, the output directory would also contain the mapping from categorical to continuous values.

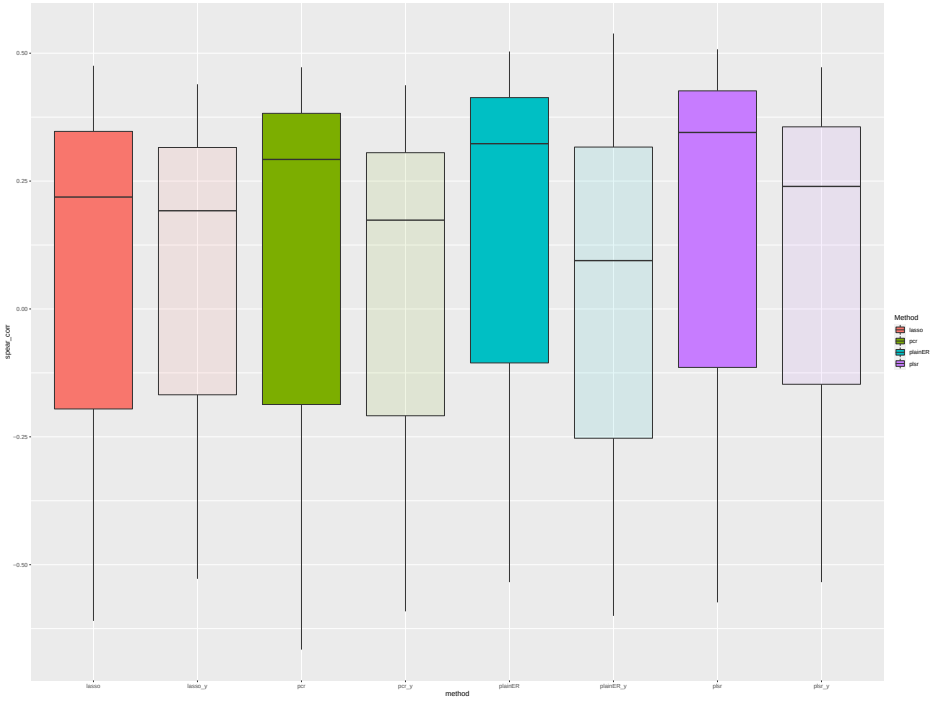

Figure 7: Optimal Delta/Lambda Boxplot

Final Notes All of the results from our run of `pipelineER1()`, `pipelineER2()`, and `pipelineER3()` can be accessed here. We recommend trying the pipeline yourself and then comparing your results to ours.