fdrSAFE Vignette

Jenna Landy

2 August 2025

Contents

```
# remove.packages("fdrSAFE")
# devtools::document()
# devtools::build(vignettes = FALSE)
# devtools::install()
```

```
library(fdrSAFE)

library(dplyr)
library(tidyr)
library(ggplot2)
library(ggthemes)

set.seed(111)

color_list = list(
  "fdrSAFE" = "#E69F00",
  "locfdr" = "#D55E00",
  "fdrtool" = "#009E73",
  "qvalue" = "#0072B2",
  "true" = "black"
)
```

0.1 Overview

`fdrSAFE` computes local and tail-end false discovery rates using ensemble methods given test statistics that are centered at zero under the null with large magnitude providing evidence against the two-sided null hypotheses. This method is introduced in the paper "fdrSAFE: Selective Aggregation for Local False Discovery Rates".

0.2 Quick start guide

Given a vector of test statistics, the `fdrSAFE` function can be used to compute local and tail-end false discovery rates. Here is an example using `fdrSAFE` on a simulated set of 1000 test statistics, corresponding to 1000 two-sided hypotheses tests. Parallelization is utilized by default, but can be turned off with `parallel = FALSE`. Progress of the algorithm is displayed by default, but this can be turned off with `verbose = FALSE`.

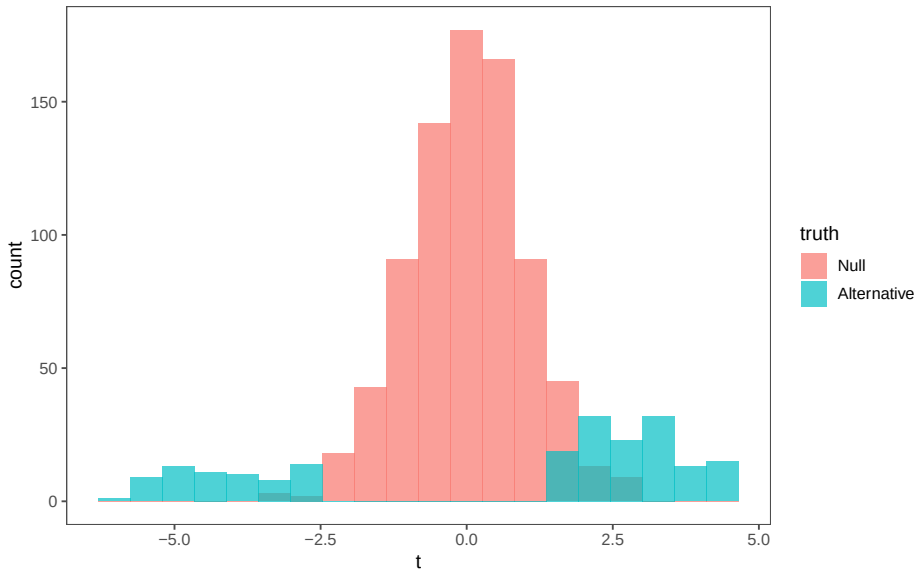
1. Simulate test statistics and true hypothesis labels

This dataset is from simulation scheme 2 defined in our paper. Null hypotheses have statistics coming from a $N(0,1)$, while alternative hypothese have most $(2/3)$ statistics coming from $U(1.5,4.5)$ and the rest $(1/3)$ from $U(-6,-2.5)$. 80% of hypotheses are null ($\pi_0 = 0.8$).

```
test_statistics = c(
  rnorm(800, 0, 1),   # 80% of values near 0
  runif(66, -6, -2.5), # add 6.6% negative extreme values
  runif(134, 1.5, 4.5)    # add 13.4% positivive extreme values
)
truth = c(
  rep(FALSE, 800),
  rep(TRUE, 200)
```

```
)
true_pi0 = 1-mean(truth)

data.frame(
  t = test_statistics,
  truth = factor(
    ifelse(truth, 'Alternative', 'Null'),
    levels = c('Null','Alternative')
  )
) %>%
  ggplot(aes(x = t, fill = truth)) +
  geom_histogram(bins = 20, alpha = 0.7, position = 'identity') +
  theme_few()
```



For evaluation purposes, we can also compute the true local and tail-end false discovery rates.

```
null = function(t) {dnorm(t, 0, 1)}
alt = function(t) {dunif(t, -6, -2.5) * 1/3 + dunif(t, 1.5, 4.5) * 2/3}
mix = function(t) {
  true_pi0 * null(t) + (1-true_pi0) * alt(t)
}
true_fdr = sapply(test_statistics, function(t) {
  null(t) * true_pi0 / mix(t)
})
true_Fdr = sapply(test_statistics, function(t) {
  # empirical Pr(hypotheses i null | t_i more extreme than t)
  mean(truth[abs(test_statistics) >= abs(t)] == 0)
})
```

2. Run fdrSAFE with default options Uncomment parallel = FALSE if multiple cores are not available or R session is crashing

```
fdrSAFE_res <- fdrSAFE(
  test_statistics#, parallel = FALSE
)
#> Building locfdr grid in parallel
#>  139/150 locfdr models included
#>      11 models dropped for pi0 > 1
#> Building fdrtool grid in parallel
#>  20/22 fdrtool models included
#>      2 models dropped for pi0 == 1
#> Building qvalue grid in parallel
#>  120/120 qvalue models included
#> Fitting synthetic generator
#> Running grid search in parallel
#> Ensembling
```

3. Access Results

`fdrSAFE` returns a list object containing estimates of local (fdr) and two-sided tail-end (Fdr) false discovery rates, as well as the proportion of test statistics from the null distribution, π_0.

```
fdrSAFE_fdr <- fdrSAFE_res$fdr
fdrSAFE_Fdr <- fdrSAFE_res$Fdr
fdrSAFE_pi0 <- fdrSAFE_res$pi0
```

`fdrSAFE` also returns the top grid of models included in the final ensemble. We can investigate how many models each package contributes to this ensemble.

```
table(fdrSAFE_res$top_grid$method)
#>
#> locfdr qvalue
#>      2      8
```

We can compare estimates between `fdrSAFE` and benchmark models, which are the three packages `fdrSAFE` is based on with their default parameters: `locfdr`, `fdrtool`, and `qvalue`. Each packages' default results are found as `fdrSAFE_res$default_<package>`.

0.3 Evaluation

The results of each package with default hyperparameters are stored in the fdrSAFE output as `default_[package]`. Each benchmark model has it's own notation for the π_0 estimate; see the respective package's documentation for details.

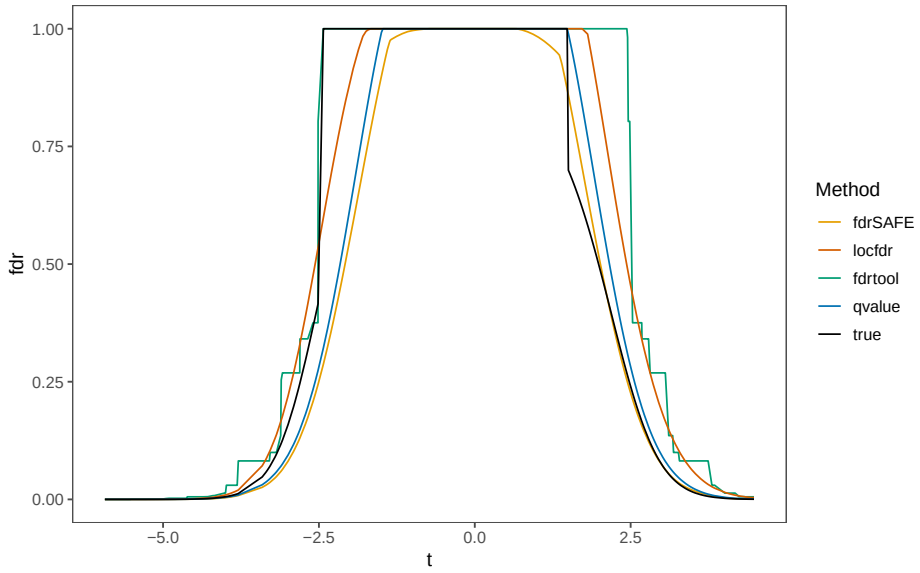
Below we compare methods just by estimated π_0, to start. We see that `fdrSAFE` estimates closest to the true value of 0.8.

```
data.frame(
  'true' = true_pi0,
  'fdrSAFE' = fdrSAFE_pi0,
  'locfdr' = unlist(fdrSAFE_res$default_locfdr$fp0['mlest','p0']),
  'fdrtool' = unname(fdrSAFE_res$default_fdrtool$param[,'eta0']),
  'qvalue' = fdrSAFE_res$default_qvalue$pi0
)
```

```
#>   true   fdrSAFE    locfdr   fdrtool    qvalue
#> 1  0.8 0.7969691 0.8504312 0.8589255 0.8351367
```

Here we compare methods by estimated fdr. We see that `fdrSAFE` tends to be closest to the truth without severely overestimating fdr. This is mirrored by `fdrSAFE` having the lowest fdr MSE. And importantly, it is close to the truth for small fdr values / extreme test statistics.

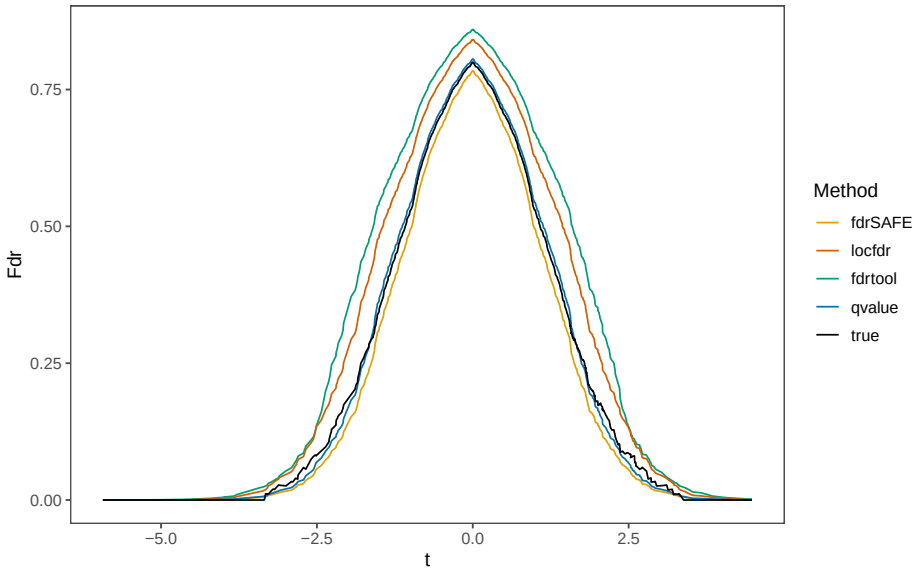
```
fdr_dat <- data.frame(
  t = test_statistics,
  true = true_fdr,
  fdrSAFE = fdrSAFE_fdr,
  locfdr = fdrSAFE_res$default_locfdr$fdr,
  fdrtool = fdrSAFE_res$default_fdrtool$lfdr,
  qvalue = fdrSAFE_res$default_qvalue$lfdr
)
```




```
#> fdrSAFE_fdr_MSE  locfdr_fdr_MSE fdrtool_fdr_MSE  qvalue_fdr_MSE
#>     0.009144359     0.013111226     0.024278721     0.009415681
```

Here we compare methods by estimated Fdr. For each t, true Fdr is the probability a test is null given it's statistic is as or more extreme than t, estimated empirically with the simulated data using the true hypothesis labels. We see that `fdrSAFE` tends to be closest to the truth. This is mirrored by `fdrSAFE` having the lowest Fdr MSE.

```
Fdr_dat <- data.frame(
  t = test_statistics,
  true = true_Fdr,
  fdrSAFE = fdrSAFE_Fdr,
  locfdr = Fdr_from_fdr(fdrSAFE_res$default_locfdr$fdr, test_statistics),
  fdrtool = Fdr_from_fdr(fdrSAFE_res$default_fdrtool$lfdr, test_statistics),
  qvalue = Fdr_from_fdr(fdrSAFE_res$default_qvalue$lfdr, test_statistics)
)
```




```
#> fdrSAFE_Fdr_MSE  locfdr_Fdr_MSE fdrtool_Fdr_MSE  qvalue_Fdr_MSE
#>    0.0007343194    0.0055541082    0.0123475373    0.0001233548
```

0.4 Options

0.4.1 Number of Simulations and Ensemble Size

By default, `fdrSAFE` will use `n_synthetic = 10` synthetic datasets to estimate model performances, and will choose the top `ensemble_size = 10` models to ensemble This can easily be changed by the user.

```
fdrSAFE_res <- fdrSAFE(
  test_statistics,
  n_synthetic = 5,
  ensemble_size = 20,
  verbose = FALSE
)

fdrSAFE_res$pi0
#> [1] 0.8212192
```

0.4.2 Grids of Model Options

The models considered are held in a data frame, or a grid, returned by the functions `build_locfdr_grid`, `build_fdrtool_grid`, and `build_qvalue_grid`.

If a user wants a more personalized set of models considered, they can build their own grids with these functions. Options for categorical variables are specified as a vector of unspecified length. Ranges for continuous variables are specified as a vector of two values: `c(min, max)`. The parameter `grid_depth` determines how many values are considered within each range of continuous variables.

```
my_locfdr_grid <- build_locfdr_grid(
  test_statistics,
  pct_range = c(0.001, 0.003),
  pct0_range = c(1/3,1/4),
  nulltype = c(1,2),
  type = c(0),
  grid_depth = 10
)

my_fdrtool_grid <- build_fdrtool_grid(
  test_statistics,
  cutoff.method = c('fndr','pct0'),
  pct0_range = c(1/3,1/4),
  grid_depth = 10
)

my_qvalue_grid <- build_qvalue_grid(
  test_statistics,
  transf = c('probit','logit'),
  adj_range = c(0.5, 1.5),
  pi0.method = c('bootstrap'),
  smooth.log.pi0 = c('FALSE'),
  grid_depth = 10
)

fdrSAFE_res <- fdrSAFE(
  test_statistics,
  locfdr_grid = my_locfdr_grid,
  fdrtool_grid = my_fdrtool_grid,
  qvalue_grid = my_qvalue_grid,
  verbose = FALSE
)

fdrSAFE_res$pi0
#> [1] 0.7480094
```

To exclude one of the three package from consideration entirely, set the grid value to `NULL`.

```
fdrSAFE_nofdrtool_res <- fdrSAFE(
  test_statistics,
  fdrtool_grid = NULL,
  verbose = FALSE
)

fdrSAFE_res$pi0
#> [1] 0.7480094
```

0.4.3 Size of Simulated Datasets

By default, the simulated datasets are the same size as the real data. If this is too large given a user's computational limitations, the size of simulated datasets can be specified with `synthetic_size`.

```
fdrSAFE_res <- fdrSAFE(
  test_statistics,
  synthetic_size = 100,
  verbose = FALSE
)

fdrSAFE_res$pi0
#> [1] 0.8043711
```

0.4.4 Provide custom test statistic to p-value conversion function

If a t-distributed (or standard normal, since we're not providing df) null is not appropriate for converting test statistics to p-values, a custom `to_pval_function` can be provided. Here we define a function to convert statistics to p-values based on a $t_{(3)}$ as an example.

```
my_to_pval_function = function(test_statistics) {
  one_sided <- unlist(lapply(test_statistics, function(z) {
    stats::pt(-1*abs(z), df = 3)
  }))
  2*one_sided
}

fdrSAFE_res <- fdrSAFE(
  test_statistics,
  to_pval_function = my_to_pval_function,
  verbose = TRUE
)
#> Building locfdr grid in parallel
#>  139/150 locfdr models included
#>      11 models dropped for pi0 > 1
#> Building fdrtool grid in parallel
#>  20/22 fdrtool models included
#>      2 models dropped for pi0 == 1
#> Building qvalue grid in parallel
#>  120/120 qvalue models included
#> Fitting synthetic generator
#> Running grid search in parallel
#> Ensembling

table(fdrSAFE_res$top_grid$method)
#>
#> locfdr
#>     10

fdrSAFE_res$pi0
#> [1] 0.7983285
```

0.5 Advanced tutorial: understanding how fdrSAFE works

0.5.1 Synthetic Generator

The synthetic generator is a mixture model fit to the observed test-statistics $\mathbf{u}$ such that both test statistics and hypothesis labels can be generated for model evaluation on synthetic data. Below is the form of the synthetic generator:

$$f_G(u|\phi) = \pi_0 f_{G,0}(u|\sigma_0) + (1-\pi_0) f_{G,1}(u|\pi_{1n}, \sigma_{1n}, \sigma_{1p})$$
$$f_{G,0}(u|\sigma_0) = N(u; 0, \sigma_0)$$
$$f_{G,1}(u|\pi_{1n}, \sigma_{1n}, \sigma_{1p}) = \pi_{1n} \cdot \frac{2u^2}{\sigma_{1n}^2} N(u; 0, \sigma_{1n}) I(u<0) + (1-\pi_{1n}) \cdot \frac{2u^2}{\sigma_{1p}^2} N(u; 0, \sigma_{1p}) I(u>0)$$

The first component, $f_{G0}(u|\sigma_0)$, corresponds to statistics under the null hypothesis, and is Normal and centered at zero with standard deviation σ_0. The second, $f_{G1}(u|\pi_{1n}, \sigma_{1n}, \sigma_{1p})$, corresponding to statistics under the alternative hypothesis, is a mixture of two Non-Local Half-Normals centered at 0 and with separate standard deviations for the negative and positive sides, σ_{1n} and σ_{1p}. Mixing parameter π_{1n} controls the proportion of alternative test statistics that are negative. This allows for asymmetry in not-null test statistics, both in terms of mass and distribution shape, while keeping density strictly away from zero. The full synthetic generator mixture is completed by the mixture weight π_0, the proportion of null tests. For conciseness, $\phi = (\pi_0, \sigma_0, \pi_{1n}, \sigma_{1n}, \sigma_{1p})$, and $f_G(u|\phi)$ is the implied marginal.

fdrSAFE results report the fit synthetic generator in `fdrSAFE_res$synthetic_generator$parameters`. In this case, the synthetic generator is able to produce synthetic datasets that are similar in shape to the original data–similar spread, more positive test statistics than negative with $\pi_{1n} < 0.5$, and a wider spread of negative test statistics than positive with $\sigma_{1n}^2 > \sigma_{1p}^2$.

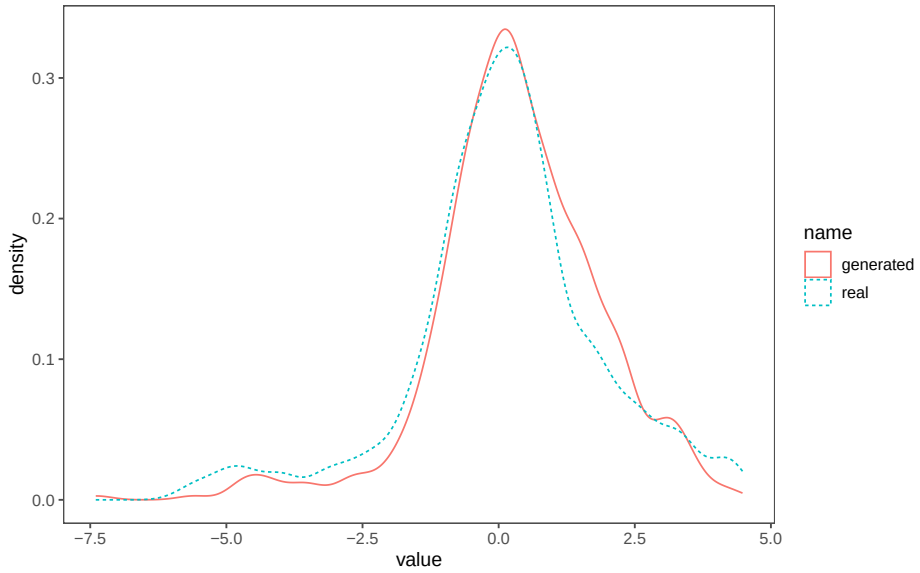
```
fdrSAFE_res <- fdrSAFE(
  test_statistics,
  verbose = FALSE
)
data.frame(fdrSAFE_res$synthetic_generator$parameters)
#>    sigmasq0       pi0 sigmasq1n sigmasq1p      pi1n
#> 1 0.7201775 0.7175283  4.442042  2.416294 0.3483208
```

Internally, fdrSAFE calls `simulate_from_synthetic_generator` to generate the synthetic datasets. We can call this now to visually compare the synthetic and real datasets, for both the standard and extreme test statistic distributions.

```
generated = simulate_from_synthetic_generator(
  n = length(test_statistics),
  synthetic_generator = fdrSAFE_res$synthetic_generator
)

data.frame(
  real = test_statistics, generated = generated$t
) %>%
  pivot_longer(1:2) %>%
  ggplot(aes(x = value, color = name, lty = name)) +
  geom_density() +
```

```
  theme_few()
```



If we add a more extreme range to the test-statistics and rerun fdrSAFE, the synthetic generator has adapted with wider alternative variances, and much larger for the negative statistics than the positive.

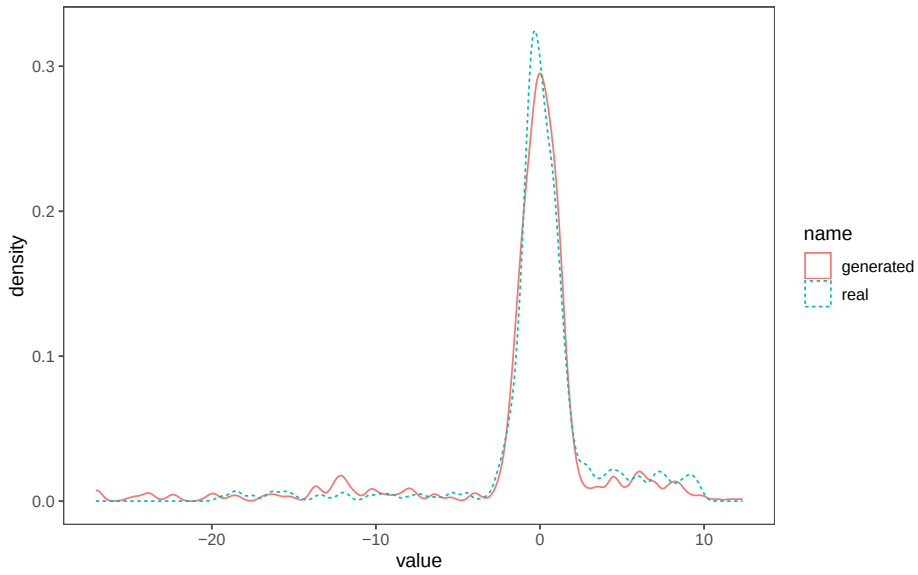
```
test_statistics_extreme = c(
  rnorm(800, 0, 1),   # 80% of values near 0
  runif(66, -20, -2.5), # add 6.6% negative extreme values
  runif(134, 1.5, 10)    # add 13.4% positivive extreme values
)
truth_extreme = c(
  rep(FALSE, 800),
  rep(TRUE, 200)
)
null = function(t) {dnorm(t, 0, 1)}
alt = function(t) {dunif(t, -20, -2.5) * 1/3 + dunif(t, 1.5, 10) * 2/3}
mix = function(t) {
  true_pi0 * null(t) + (1-true_pi0) * alt(t)
}
true_fdr_extreme = sapply(test_statistics, function(t) {
  null(t) * true_pi0 / mix(t)
})
true_Fdr_extreme = sapply(test_statistics, function(t) {
  # empirical Pr(hypotheses i null | t_i more extreme than t)
  mean(truth[abs(test_statistics) >= abs(t)] == 0)
})
```

```
fdrSAFE_extreme_res <- fdrSAFE(
  test_statistics_extreme,
  verbose = FALSE
)
#> Warning in locfdr::locfdr(test_statistics, plot = 0): f(z) misfit = 2.5.  Rerun
```

```
#> with increased df
data.frame(fdrSAFE_extreme_res$synthetic_generator$parameters)
#>    sigmasq0      pi0 sigmasq1n sigmasq1p      pi1n
#> 1 0.9840483 0.802034  53.85021  13.14825 0.3305044
```

```
generated_extreme = simulate_from_synthetic_generator(
  n = length(test_statistics_extreme),
  synthetic_generator = fdrSAFE_extreme_res$synthetic_generator
)

data.frame(
  real = test_statistics_extreme, generated = generated_extreme$t
) %>%
  pivot_longer(1:2) %>%
  ggplot(aes(x = value, color = name, lty = name)) +
  geom_density() +
  theme_few()
```



While there is variation between real and generated test statistics for both datasets, we would expect the models to perform more similarly between generated and real standard test statistics than across real standard and extreme test statistics. This is the premise of fdrSAFE's model selection procedure: the synthetic datasets are *close enough* to the real ones that we are able to get an idea of what models perform well in the given setting. This is validated in Section 0.5.3.

fdrSAFE has selected models from fdrtool and locfdr for the extreme dataset, and models frmm qvalue for the standard dataset.

```
table(fdrSAFE_extreme_res$top_grid$method)
#>
#> fdrtool  qvalue
#>       1       9
table(fdrSAFE_res$top_grid$method)
```

```
#>
#> locfdr qvalue
#>      2      8
```

0.5.2 Model subset selection

`fdrSAFE_res$all_grids` holds the estimated model performance for each model on each synthetic dataset, where a model is defined by a package ("method" column) and a set of parameters (mapped to by the "row" column), and synthetic datasets are indexed by the "sim" column. The fdrSAFE objective estimator is obtained by averaging fdrerror across synthetic datasets for each model. Importantly, these are metrics on synthetic data only, not on the observed data.

```
head(fdrSAFE_res$all_grids)
#>    fdrerror  Fdrerror      pi0 method row sim
#> 1 0.0947946 0.1004309 0.919209 locfdr   1   1
#> 2 0.1231669 0.2148697 0.919209 locfdr   2   1
#> 3 0.1474595 0.2943368 0.919209 locfdr   3   1
#> 4 0.2112936 0.4250289 0.919209 locfdr   4   1
#> 5 0.2112936 0.4250289 0.919209 locfdr   5   1
#> 6 0.0947946 0.1004309 0.919209 locfdr   6   1
```

Exact model specifications can be found in each package's respective grid. Note that these grids do not hold estimated metrics, only parameter values.

```
head(fdrSAFE_res$locfdr_grid)
#>    pct  pct0 nulltype type
#> 1 0.00 0.000        1    0
#> 2 0.05 0.000        1    0
#> 3 0.10 0.000        1    0
#> 4 0.15 0.000        1    0
#> 5 0.20 0.000        1    0
#> 6 0.00 0.075        1    0
```

`fdrSAFE_res$top_grid` reports the subset of models used in the final ensemble. This is a subset of `fdrSAFE_res$all_grids` after averaging across synthetic datasets and ordering by the fdrSAFE objective estimate. Importantly, these are still metrics on synthetic data only, not on the observed data.

```
fdrSAFE_res$top_grid
#> # A tibble: 10 x 6
#> # Groups:   method [2]
#>    method   row fdrerror Fdrerror   pi0 best
#>    <chr>  <int>    <dbl>    <dbl> <dbl> <lgl>
#>  1 locfdr    44   0.0136  0.00790 0.790 TRUE
#>  2 locfdr   136   0.0137  0.0121  0.838 TRUE
#>  3 qvalue    40   0.0158  0.00859 0.797 TRUE
#>  4 qvalue    38   0.0161  0.00881 0.797 TRUE
#>  5 qvalue    36   0.0164  0.00903 0.797 TRUE
#>  6 qvalue    80   0.0165  0.00877 0.808 TRUE
#>  7 qvalue   120   0.0167  0.00889 0.810 TRUE
#>  8 qvalue    34   0.0168  0.00927 0.797 TRUE
```

```
#>  9 qvalue    78   0.0168  0.00897 0.808 TRUE
#> 10 qvalue   118   0.0170  0.00909 0.810 TRUE
```

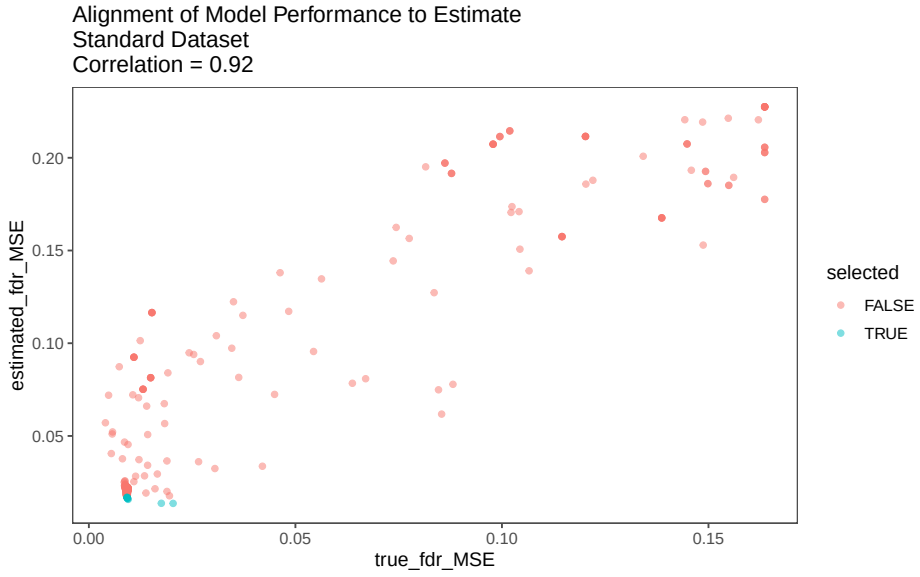
0.5.3 Oracle model performance, if ground truth is available

We provide the function `oracle_grid_search` to repeat the grid search procedure but with models evaluated on the real dataset. This is only an option if true local or tail-end false discovery rates are available, i.e., only when the true hypothesis labels or generating distributions are known.

```
oracle_grid = oracle_grid_search(
  test_statistics, true_fdr = true_fdr, true_Fdr = true_Fdr,
  locfdr_grid = fdrSAFE_res$locfdr_grid,
  fdrtool_grid = fdrSAFE_res$fdrtool_grid,
  qvalue_grid = fdrSAFE_res$qvalue_grid
)
#> Running grid search
head(oracle_grid)
#>      fdrerror     Fdrerror       pi0 method row
#> 1 0.004007450 2.842660e-03 0.9088626 locfdr  57
#> 2 0.004778255 2.210347e-03 0.9070326 locfdr  37
#> 3 0.005435148 4.122684e-05 0.8051745 locfdr 112
#> 4 0.005655681 5.613677e-04 0.8183173 locfdr 108
#> 5 0.005734819 6.536578e-04 0.8178576 locfdr  36
#> 6 0.007360936 2.881423e-03 0.9078349 locfdr 104
```

Here, we combine the oracle performance data with the estimated performances from fdrSAFE. We plot the true vs estimated fdr MSE, and color models by whether they were selected for the final ensemble.

There is a high correlation between estimated and true model performance, and importantly, the models selected by fdrSAFE for ensemble show excellent model performance on the real data. This plot also gives the scale for how bad an fdr model can get if chosen arbitrarily.

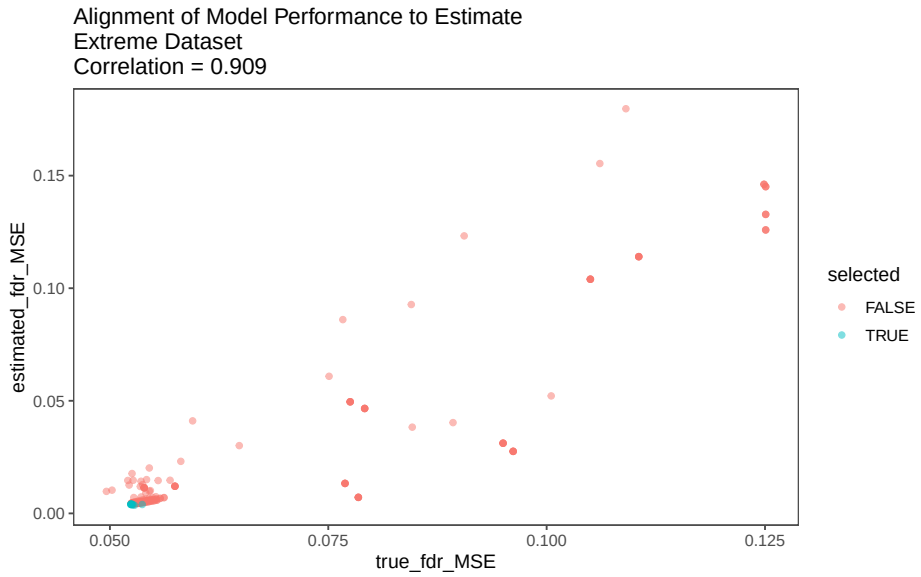


We can run the grid search on the extreme test statistics as well. . .

```
oracle_grid_extreme = oracle_grid_search(
  test_statistics_extreme, true_fdr = true_fdr_extreme, true_Fdr = true_Fdr_extreme,
  locfdr_grid = fdrSAFE_extreme_res$locfdr_grid,
  fdrtool_grid = fdrSAFE_extreme_res$fdrtool_grid,
  qvalue_grid = fdrSAFE_extreme_res$qvalue_grid
)
#> Running grid search
```

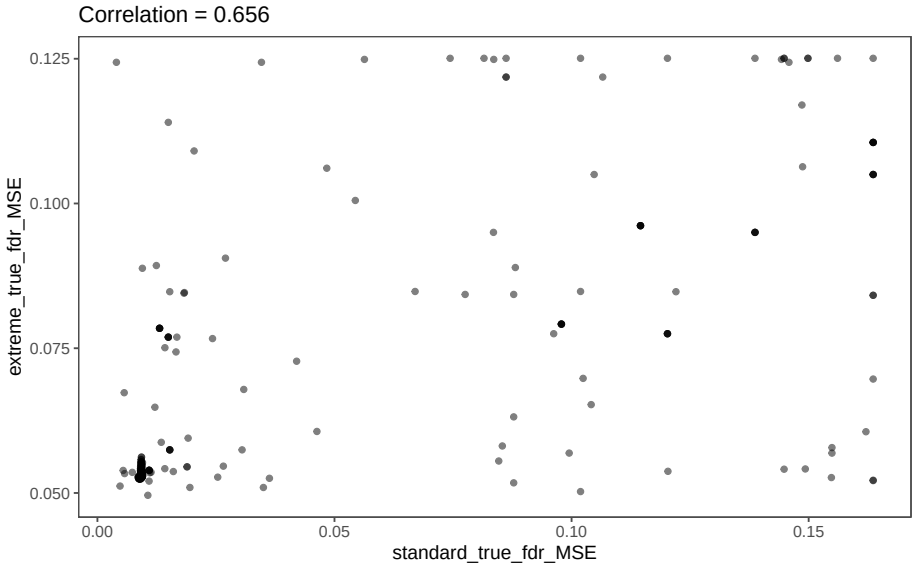
And again there is a high correlation between estimated and true model performance, and importantly, and the models selected by fdrSAFE for ensemble show excellent model performance on the real data.

```
#> `summarise()` has grouped output by 'method'. You can override using the
#> `.groups` argument.
```



Importantly, fdrSAFE was able to properly identify high-performing models in these two settings even though they are very different settings and models perform differently between them. Below is a plot of the true fdr MSE between the standard versus extreme test-statistic datasets.

Model performances are not well correlated between the two datasets – in fact some of the models that perform the best on the standard dataset perform the worst on the extreme dataset, and vise versa. This shows how single models on their own do not have robust near-optimality, as different models will perform best in different scenarios. This highlights the need for our synthetic data-based model selection procedure.


Correlation = 0.656
0.125
0.100
0.075
0.050
extreme_true_fdr_MSE
0.00
0.05
0.10
0.15
standard_true_fdr_MSE