Results and plots

Marjolein Fokkema

20-7-2022

Regression problems

For the regression problems, I fitted:

- Basic approaches:
  - GLMM tree
  - BART
  - Simplistic born again: GLMM tree fitted to original X, and $\hat{y}$ from BART instead of y.
- Posterior sampling approaches:
  - $N_{gen} = 1$
  - $N_{gen} = 5$
  - $N_{gen} = 10$
- Born-again approaches:
  - Several combinations of p_{alt} (0, .25, .5, 1) and N_{gen} (1, 5, 10).

As in the Shang & Breiman paper, I used 10 repeats of 10-fold CV for performance assessment.

Summary of findings

- Best-performing approach is Breiman's born-again.
- Higher values of N_{gen} yield both lower MSE and tree size.
- Simplistic born-again approach, which keeps original X intact seems to do pretty well already.
- High values of p_{alt} (0.5, 1.0) **only** work well on Friedman problems, which has zero correlation between predictors.
- Often, $p_{alt} = 0$ works well. Thus, simply resampling from X a lot, with no permuting of the values may often work better!
- Posterior sampling approach does not seem to outperform simplistic or Breiman's born-again approaches.

Further, I often omitted $N_{gen} = 5$ from the plots for the born-again approaches, because it always seemed to perform i nthe middle between $N_{gen} = 1$ and $N_{gen} = 10$.

Posterior sampling might work better for uncertainty quantification, but this would be future research.

Boston Housing

```
library("mlbench")
library("ggplot2")

## Compute intercorrelations
data("BostonHousing")
p <- ncol(BostonHousing)-1
sum(cor(sapply(BostonHousing[ , -14L], function(x)
  if (!is.numeric(x)) as.numeric(x) else x))) / (p*(p-1))
```

```
## [1] 0.1266869
```

```
load(file = "BostonHousing MSE.Rda")
load(file = "BostonHousing tree_size.Rda")
colMeans(MSE)
```

```
##              GLMM_tree                 Bart                   Ba
##              22.457318             9.478921            21.566889
##             BaBayes_N=1          BaBayes_N=5         BaBayes_N=10
##              23.530386            21.786926            21.623075
##      BaSmear_N=1_palt=0     BaSmear_N=5_palt=0  BaSmear_N=10_palt=0
##              22.484683            21.574473            21.364545
##   BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##              23.266902            21.820402            21.446998
##    BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##              26.156004            26.711783            26.284915
##      BaSmear_N=1_palt=1     BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##              39.813390            42.392009            42.771864
```

```
which.min(colMeans(MSE[ , -2]))
```

```
## BaSmear_N=10_palt=0
##                   8
```

```
sapply(MSE, sd)
```

```
##              GLMM_tree                 Bart                   Ba
##              10.501581             4.938728            11.125934
##             BaBayes_N=1          BaBayes_N=5         BaBayes_N=10
##              10.669522            11.371690            10.048628
##      BaSmear_N=1_palt=0     BaSmear_N=5_palt=0  BaSmear_N=10_palt=0
##               9.738342             9.966675            10.392816
##   BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##               9.832149             9.026467             8.443231
##    BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##               9.042027             9.050548             9.187882
##      BaSmear_N=1_palt=1     BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##              13.099729            12.940042            11.924300
```

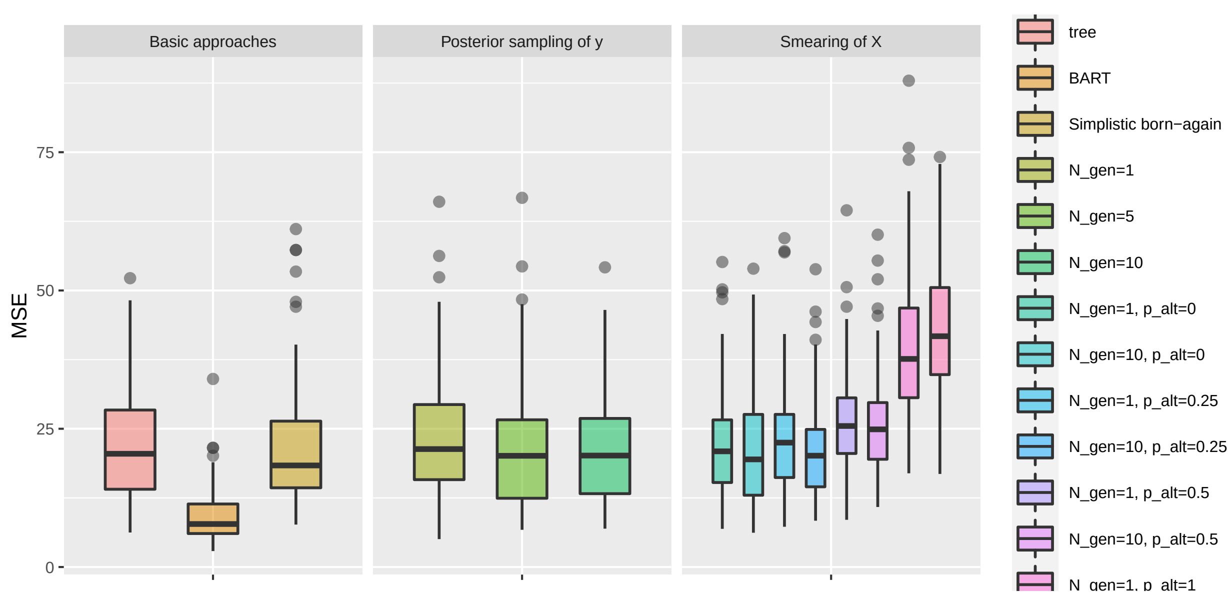
```r
colMeans(tree_size)
```

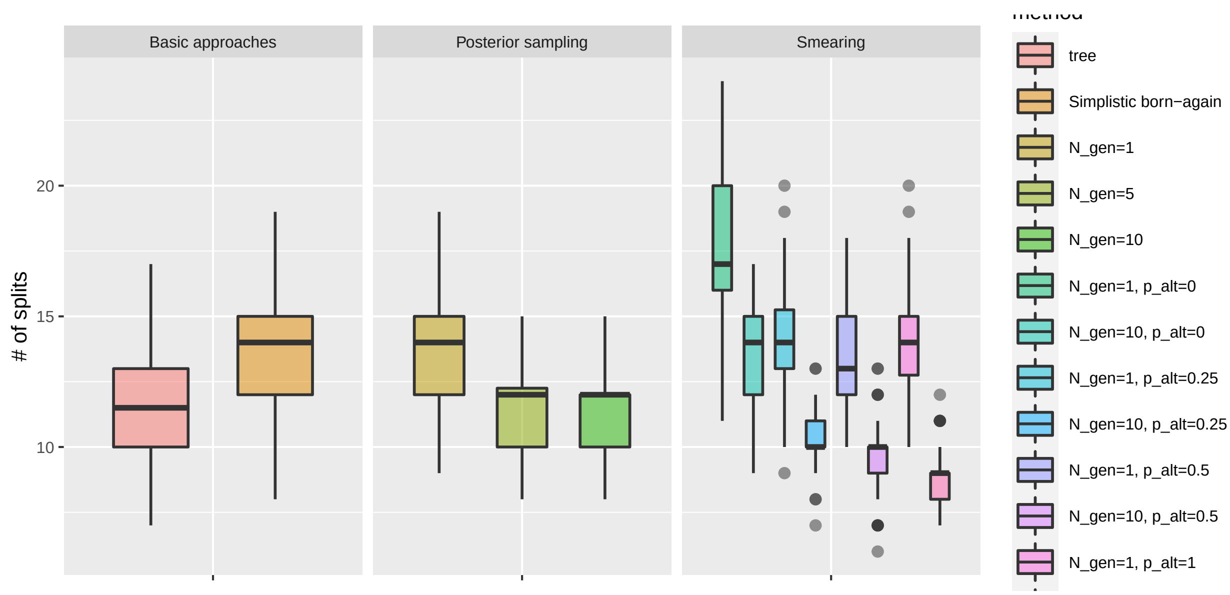
```
##                GLMM_tree                 Bart                   Ba
##                    11.54                   NA                13.62
##              BaBayes_N=1          BaBayes_N=5         BaBayes_N=10
##                    13.86                11.57                11.44
##        BaSmear_N=1_palt=0   BaSmear_N=5_palt=0  BaSmear_N=10_palt=0
##                    17.22                13.28                13.45
##   BaSmear_N=1_palt=0.25 BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##                    14.06                10.23                10.16
##     BaSmear_N=1_palt=0.5  BaSmear_N=5_palt=0.5 BaSmear_N=10_palt=0.5
##                    13.79                 9.86                 9.58
##       BaSmear_N=1_palt=1    BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##                    13.91                 9.18                 8.79
```

```r
sapply(tree_size, sd)
```

```
##                GLMM_tree                 Bart                   Ba
##                1.8280641                   NA            2.1451378
##              BaBayes_N=1          BaBayes_N=5         BaBayes_N=10
##                2.2919270            1.6221634            1.7192904
##        BaSmear_N=1_palt=0   BaSmear_N=5_palt=0  BaSmear_N=10_palt=0
##                2.8161870            2.0893187            1.6537377
##   BaSmear_N=1_palt=0.25 BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##                2.0588317            1.2701308            0.9818556
##     BaSmear_N=1_palt=0.5  BaSmear_N=5_palt=0.5 BaSmear_N=10_palt=0.5
##                1.9451831            1.3028330            1.3040729
##       BaSmear_N=1_palt=1    BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##                1.9389417            1.3210036            0.9670845
```

```r
theme_set(theme_gray(base_size = 8))
```





Ozone

```
## Compute intercorrelations
data("Ozone")
p <- ncol(Ozone)-2
sum(cor(sapply(Ozone[ , -c(9L, 13L)], function(x)
  if (!is.numeric(x)) as.numeric(x) else x), use = "pairwise.complete")) / (p*(p-1))
```

```
## [1] 0.1872888
```

```
load(file = "Ozone MSE.Rda")
load(file = "Ozone tree_size.Rda")

sapply(MSE[ , 1:5], function(x) table(is.na(x))) ## Not all computations completed
```

```
##       GLMM_tree Bart Ba BaBayes_N=1 BaBayes_N=5
## FALSE        36   36 36          36          36
## TRUE         64   64 64          64          64
```

```
colMeans(MSE, na.rm=TRUE)
```

```
##              GLMM_tree                   Bart                     Ba
##               22.39551               16.51937               21.09979
##            BaBayes_N=1            BaBayes_N=5           BaBayes_N=10
##               23.55882               21.96742               21.90282
##      BaSmear_N=1_palt=0      BaSmear_N=5_palt=0     BaSmear_N=10_palt=0
##               22.16787               21.15082               20.38417
##   BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##               23.92898               22.95203               22.92381
##    BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##               25.95640               25.99136               26.18073
##      BaSmear_N=1_palt=1      BaSmear_N=5_palt=1     BaSmear_N=10_palt=1
##               30.02278               30.43922               30.97491
```

```r
which.min(colMeans(MSE[ , -2], na.rm=TRUE))
```

```
## BaSmear_N=10_palt=0
##                   8
```

```r
sapply(MSE, sd, na.rm=TRUE)
```

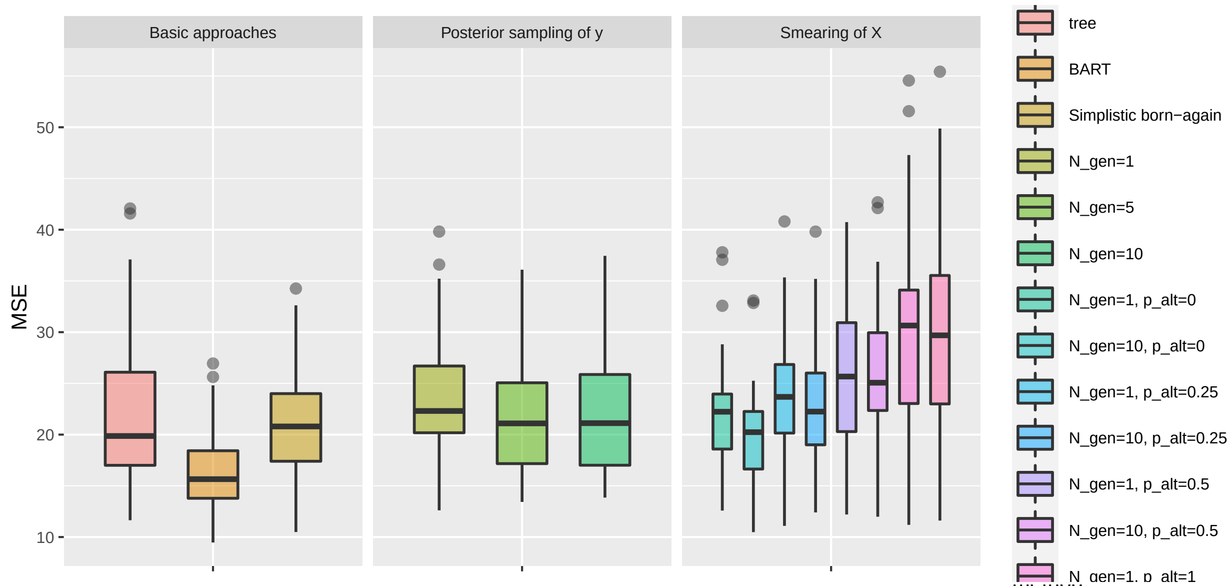
```
##             GLMM_tree                  Bart                    Ba
##              7.824551              4.448027              5.320992
##           BaBayes_N=1           BaBayes_N=5          BaBayes_N=10
##              5.933254              5.699108              6.004954
##     BaSmear_N=1_palt=0     BaSmear_N=5_palt=0   BaSmear_N=10_palt=0
##              5.618673              5.400080              4.643377
##  BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##              6.159990              6.237090              6.014912
##   BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##              7.489999              6.992449              6.573391
##     BaSmear_N=1_palt=1     BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##              9.910711              9.226773             10.367994
```

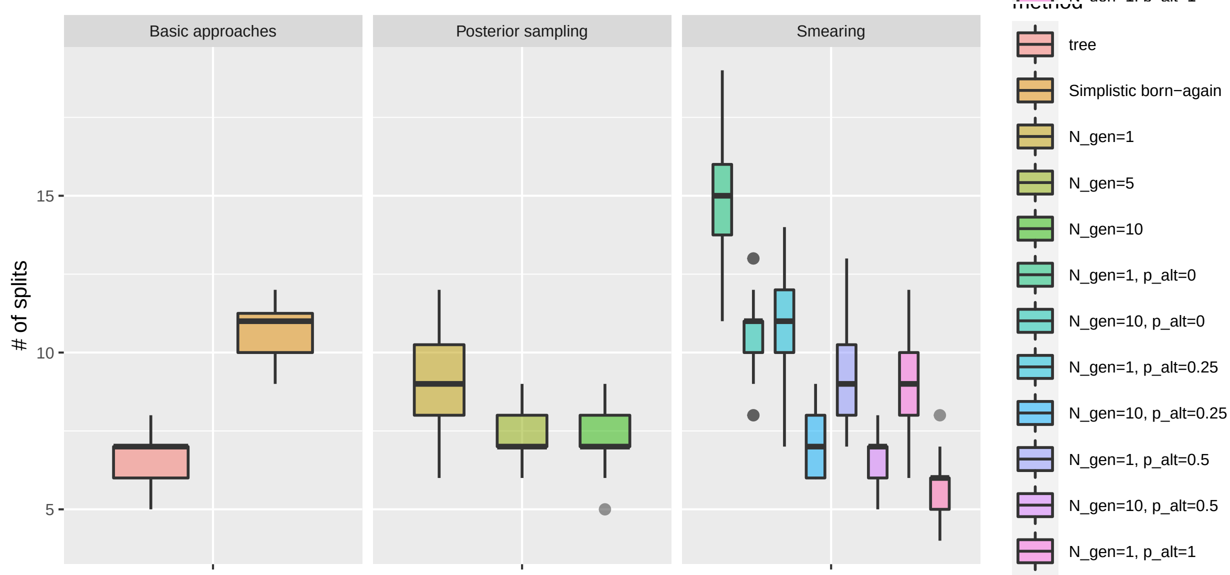
```r
colMeans(tree_size, na.rm=TRUE)
```

```
##             GLMM_tree                  Bart                    Ba
##              6.694444                   NaN             10.833333
##           BaBayes_N=1           BaBayes_N=5          BaBayes_N=10
##              9.250000              7.277778              7.388889
##     BaSmear_N=1_palt=0     BaSmear_N=5_palt=0   BaSmear_N=10_palt=0
##             14.722222             10.944444             10.555556
##  BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##             10.638889              7.527778              7.083333
##   BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##              9.416667              6.805556              6.500000
##     BaSmear_N=1_palt=1     BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##              9.027778              5.694444              5.888889
```

```r
sapply(tree_size, sd, na.rm=TRUE)
```

```
##             GLMM_tree                  Bart                    Ba
##             0.7862913                    NA             0.8783101
##           BaBayes_N=1           BaBayes_N=5          BaBayes_N=10
##             1.5560940             0.8145502             0.9343532
##     BaSmear_N=1_palt=0     BaSmear_N=5_palt=0   BaSmear_N=10_palt=0
##             1.7338827             1.3927249             1.1574466
##  BaSmear_N=1_palt=0.25  BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##             1.6062873             0.9705996             0.8409179
##   BaSmear_N=1_palt=0.5   BaSmear_N=5_palt=0.5  BaSmear_N=10_palt=0.5
##             1.5189282             0.7099072             0.7745967
##     BaSmear_N=1_palt=1     BaSmear_N=5_palt=1   BaSmear_N=10_palt=1
##             1.5581328             0.8886408             0.8544933
```





Friedman

```
library("mlbench")
load(file = "Friedman MSE.Rda")
load(file = "Friedman tree_size.Rda")
set.seed(42)
vars <- c(var(mlbench.friedman1(10000)$y),
          var(mlbench.friedman2(210000)$y),
          var(mlbench.friedman3(10000)$y))
sapply(MSE, colMeans)
```

```
##                              [,1]      [,2]        [,3]
## GLMM_tree               12.076007 33187.34 0.05033159
## Bart                     2.304523 21054.85 0.01880145
```

```
## Ba                     11.899889 30487.48 0.04839876
## BaBayes_N=1            13.166062 35566.36 0.05146380
## BaBayes_N=5            12.247483 32711.49 0.05013863
## BaBayes_N=10           12.207703 32719.61 0.05031103
## BaSmear_N=1_palt=0     13.042185 33394.36 0.05041427
## BaSmear_N=5_palt=0     12.073354 31498.65 0.04928297
## BaSmear_N=10_palt=0    11.976251 31029.08 0.04846355
## BaSmear_N=1_palt=0.25  12.075405 31172.79 0.04935003
## BaSmear_N=5_palt=0.25  11.507193 30023.96 0.04778332
## BaSmear_N=10_palt=0.25 11.339046 29879.85 0.04781966
## BaSmear_N=1_palt=0.5   12.034039 31381.05 0.04956370
## BaSmear_N=5_palt=0.5   11.288992 29343.29 0.04745073
## BaSmear_N=10_palt=0.5  11.329361 29413.42 0.04769477
## BaSmear_N=1_palt=1     11.819585 31570.06 0.04928029
## BaSmear_N=5_palt=1     11.198944 29506.09 0.04784267
## BaSmear_N=10_palt=1    11.247026 29137.52 0.04811013
```

```r
round((1/vars)*(t(sapply(MSE, colMeans))), digits = 3) ## R2s
```

```
##      GLMM_tree  Bart    Ba BaBayes_N=1 BaBayes_N=5 BaBayes_N=10
## [1,]     0.486 0.093 0.479       0.530       0.493        0.491
## [2,]     0.208 0.132 0.191       0.223       0.205        0.205
## [3,]     0.442 0.165 0.425       0.451       0.440        0.441
##      BaSmear_N=1_palt=0 BaSmear_N=5_palt=0 BaSmear_N=10_palt=0
## [1,]              0.525              0.486               0.482
## [2,]              0.210              0.198               0.195
## [3,]              0.442              0.432               0.425
##      BaSmear_N=1_palt=0.25 BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
## [1,]                 0.486                 0.463                  0.456
## [2,]                 0.196                 0.188                  0.187
## [3,]                 0.433                 0.419                  0.420
##      BaSmear_N=1_palt=0.5 BaSmear_N=5_palt=0.5 BaSmear_N=10_palt=0.5
## [1,]                0.484                0.454                 0.456
## [2,]                0.197                0.184                 0.185
## [3,]                0.435                0.416                 0.418
##      BaSmear_N=1_palt=1 BaSmear_N=5_palt=1 BaSmear_N=10_palt=1
## [1,]              0.476              0.451               0.453
## [2,]              0.198              0.185               0.183
## [3,]              0.432              0.420               0.422
```

```r
sapply(MSE, function(x) which.min(colMeans(x[ , -2])))
```

```
##   BaSmear_N=5_palt=1  BaSmear_N=10_palt=1 BaSmear_N=5_palt=0.5
##                   16                   17                   13
```

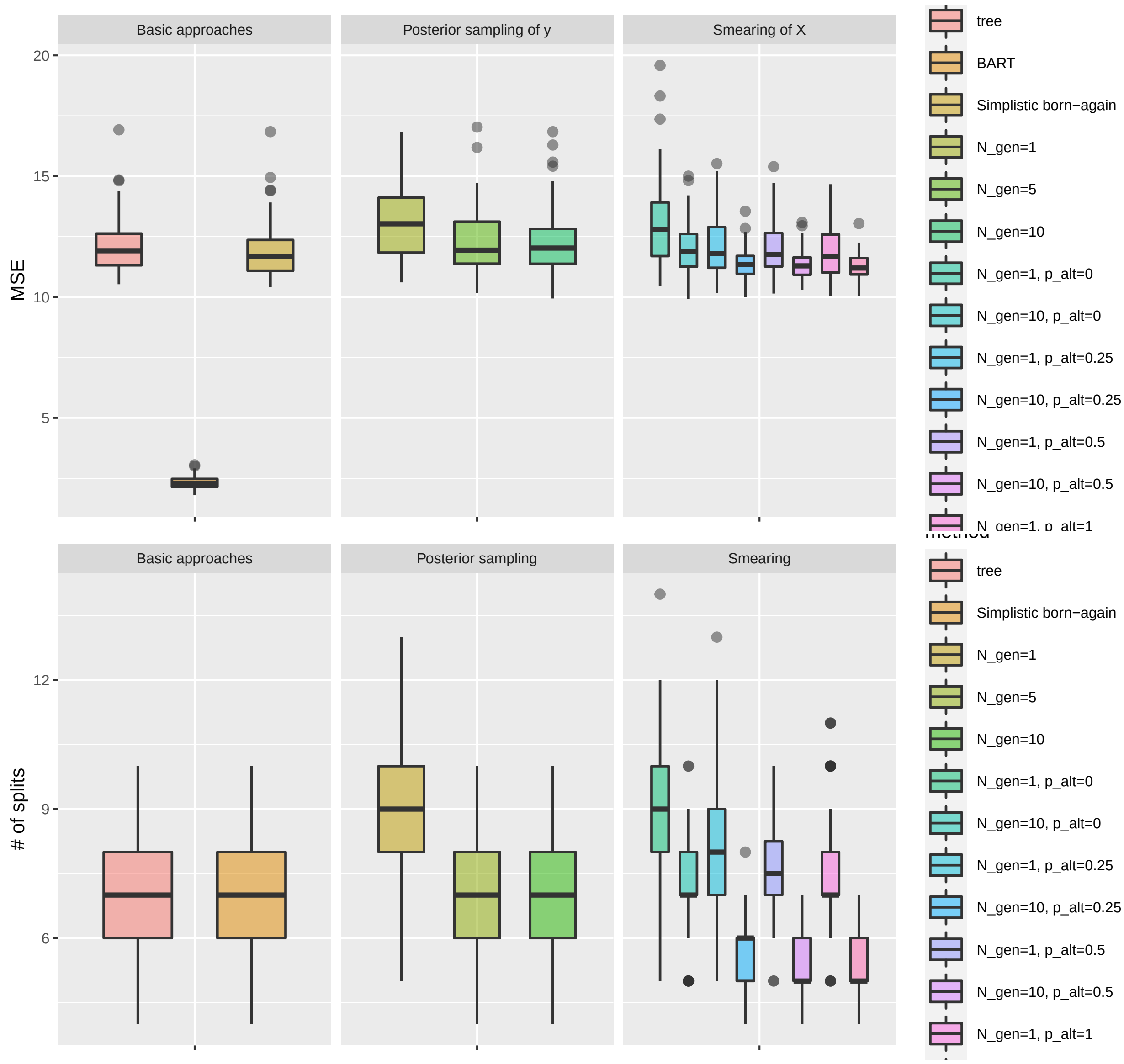
```r
sapply(MSE, function(x) sapply(x, sd))
```

```
##                             [,1]     [,2]        [,3]
## GLMM_tree              1.0830838 2599.800 0.006148888
## Bart                   0.2321861 1206.724 0.001907856
## Ba                     1.1238855 2565.068 0.005855916
## BaBayes_N=1            1.5623625 5189.825 0.006404210
```

```
## BaBayes_N=5            1.2824767 2559.969 0.005485072
## BaBayes_N=10           1.3374984 3283.079 0.005854500
## BaSmear_N=1_palt=0     1.6714595 3839.703 0.006336858
## BaSmear_N=5_palt=0     1.2009723 2653.883 0.005337941
## BaSmear_N=10_palt=0    1.0352730 2696.107 0.005994306
## BaSmear_N=1_palt=0.25  1.1761710 2496.272 0.005894570
## BaSmear_N=5_palt=0.25  0.8533598 2113.327 0.005512098
## BaSmear_N=10_palt=0.25 0.5798938 2351.263 0.005618629
## BaSmear_N=1_palt=0.5   1.1020613 2541.695 0.005726027
## BaSmear_N=5_palt=0.5   0.7202745 2222.965 0.005854793
## BaSmear_N=10_palt=0.5  0.5822913 2295.298 0.005720670
## BaSmear_N=1_palt=1     1.0387107 2930.827 0.006163060
## BaSmear_N=5_palt=1     0.6842474 2190.938 0.005722531
## BaSmear_N=10_palt=1    0.5267453 1979.481 0.005788078
```

```r
#sapply(tree_size, colMeans)
#sapply(tree_size, function(x) sapply(x, sd))
```

Plot 1st Friedman problem



Classification problems

I took the same experimental design as for the regression problems. Because $p_{alt} = 1$ did not perform well on any of the real datasets, I omitted that option.

To generate surrogate data, I took an approach reminiscent of gradient boosting: I fitted the BART model for a binomial response, and generated predicted probabilities. I fed the predicted probabilities into born-again *linear* model trees. I evaluated accuracy as MSE, which does not make it easy to compare to the results of Breiman, but it's a more informative measure of accuracy than the misclassification rate.

Breast cancer

```
## Compute intercorrelation
data("BreastCancer")
p <- ncol(BreastCancer)-2
sum(cor(sapply(BreastCancer[ , -c(1, 11)], function(x)
  if (!is.numeric(x)) as.numeric(x) else x), use = "pairwise.complete")) / (p*(p-1))
```

```
## [1] 0.7277931
```

```
load(file = "BreastCancer MSE.Rda")
load(file = "BreastCancer acc.Rda")
load(file = "BreastCancer tree_size.Rda")
colMeans(MSE)
```

```
##             tree            BART Born-again tree             N=1             N=5
##       0.04735438      0.03167728      0.04427750              NA      0.04680040
##             N=10      N=1, palt=0      N=5, palt=0    N=10, palt=0  N=1, palt=0.25
##       0.04493157              NA      0.04426277      0.04396718              NA
##   N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##       0.04169558      0.04177982              NA      0.04895715      0.04811543
```

```
which.min(colMeans(MSE[ , -2]))
```

```
## N=5, palt=0.25
##             10
```

```
which.min(colMeans(acc[ , -2]))
```

```
## N=10, palt=0.25
##              11
```

```
sapply(MSE, sd)
```

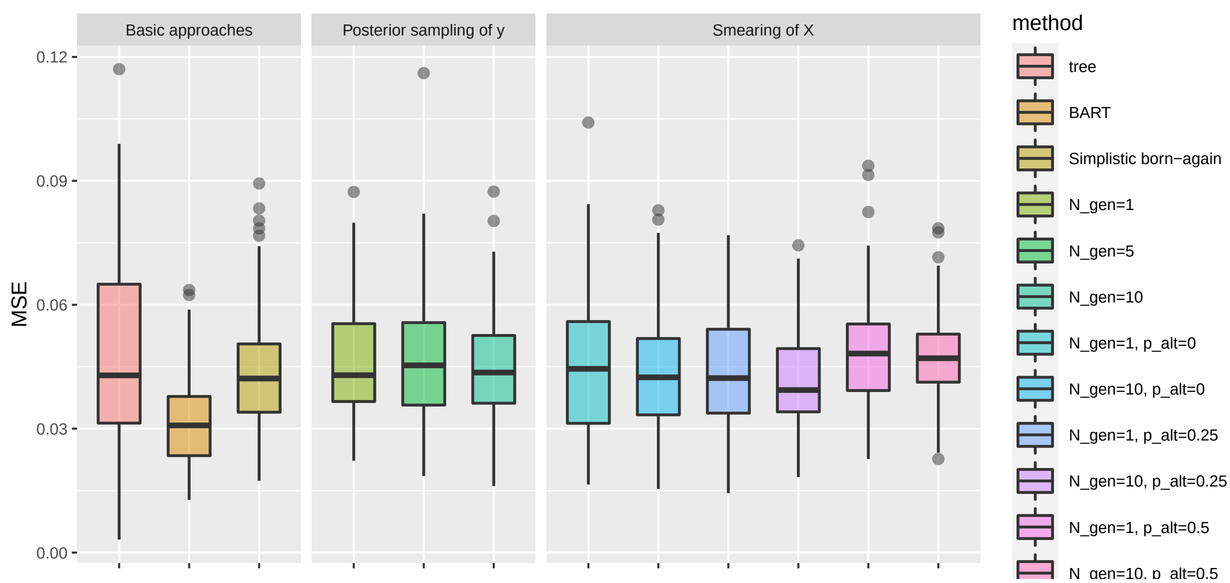
```
##             tree            BART Born-again tree             N=1             N=5
##       0.02246363      0.01097219      0.01523935              NA      0.01561629
##             N=10      N=1, palt=0      N=5, palt=0    N=10, palt=0  N=1, palt=0.25
##       0.01400681              NA      0.01591395      0.01425620              NA
##   N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##       0.01121713      0.01195805              NA      0.01220546      0.01095069
```

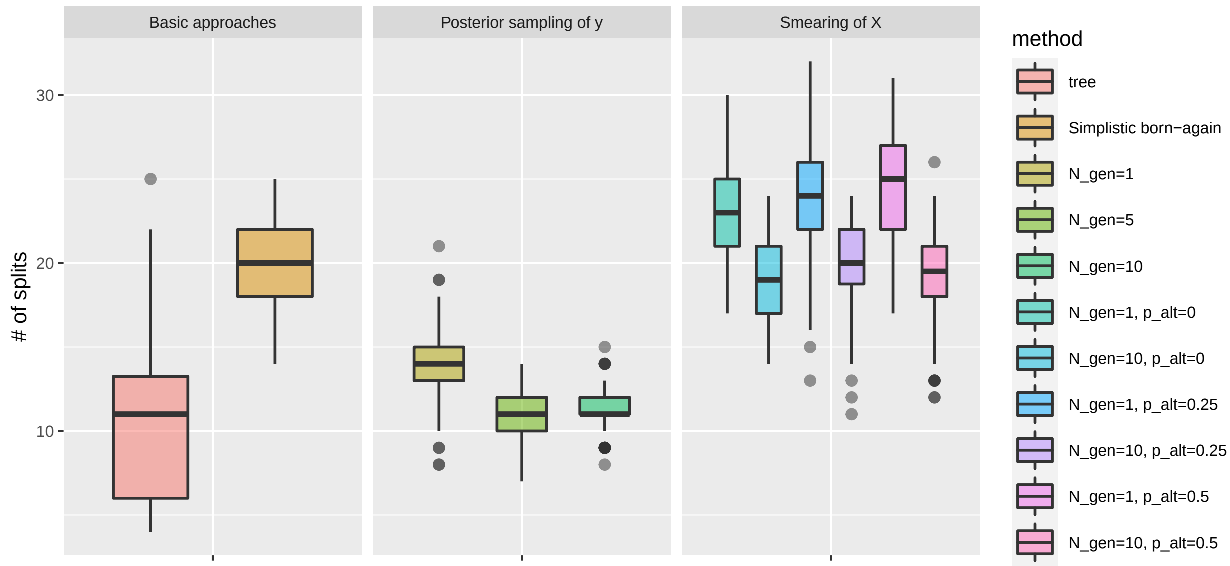
```
colMeans(tree_size)
```

```
##              tree              BART Born-again tree              N=1              N=5
##             11.06                NA           19.94               NA            11.19
##              N=10         N=1, palt=0     N=5, palt=0     N=10, palt=0   N=1, palt=0.25
##             11.42                NA           19.68            18.95               NA
##  N=5, palt=0.25 N=10, palt=0.25   N=1, palt=0.5   N=5, palt=0.5  N=10, palt=0.5
##             20.58             19.80              NA            20.08            19.01
```

```
sapply(tree_size, sd)
```

```
##              tree              BART Born-again tree              N=1              N=5
##          4.581077                NA        2.411337               NA         1.454321
##              N=10         N=1, palt=0     N=5, palt=0     N=10, palt=0   N=1, palt=0.25
##          1.342243                NA        2.428285         2.302283               NA
##  N=5, palt=0.25 N=10, palt=0.25   N=1, palt=0.5   N=5, palt=0.5  N=10, palt=0.5
##          2.433686          2.605356              NA         2.805766         2.900697
```





Ionosphere

```
library("mlbench")

## Compute intercorrelation
data("Ionosphere")
p <- ncol(Ionosphere)-2
sum(cor(sapply(Ionosphere[ , -c(2L, 35L)], function(x)
  if (!is.numeric(x)) as.numeric(x) else x), use = "pairwise.complete")) / (p*(p-1))
```

```
## [1] 0.1491748
```

```
load(file = "Ionosphere MSE.Rda")
load(file = "Ionosphere acc.Rda")
load(file = "Ionosphere tree_size.Rda")
colMeans(MSE)
```

```
##            tree            BART Born-again tree             N=1             N=5
##      0.08687282      0.07422132      0.09323717      0.10437273      0.10375217
##            N=10      N=1, palt=0      N=5, palt=0     N=10, palt=0 N=1, palt=0.25
##      0.09948669      0.09944799      0.09605123      0.09387876      0.11116266
##  N=5, palt=0.25 N=10, palt=0.25   N=1, palt=0.5   N=5, palt=0.5  N=10, palt=0.5
##      0.10825918      0.11029849      0.12843295      0.12609946      0.12708126
```

```
which.min(colMeans(MSE[ , -2]))
```

```
## tree
##    1
```

```
which.min(colMeans(acc[ , -2]))
```

```
## N=10, palt=0.5
##             14
```

```
sapply(MSE, sd)
```

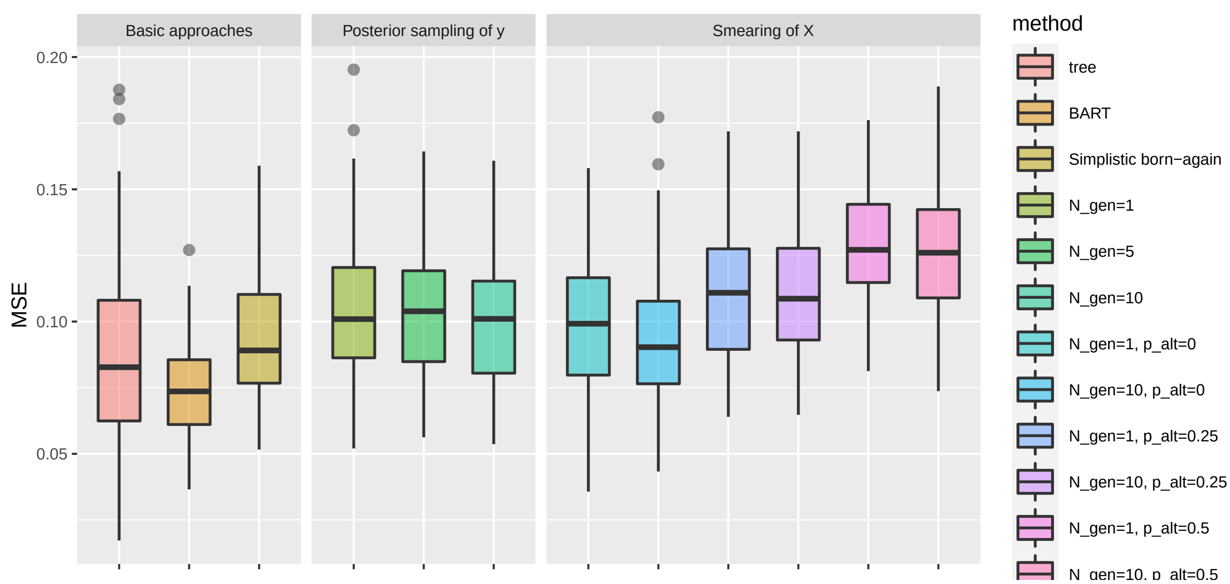
```
##            tree            BART Born-again tree             N=1             N=5
##      0.03745643      0.01740475      0.02455141      0.02710168      0.02448745
##            N=10      N=1, palt=0      N=5, palt=0     N=10, palt=0  N=1, palt=0.25
##      0.02477888      0.02718709      0.02339188      0.02548563      0.02496218
##  N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##      0.02156075      0.02260041      0.02265609      0.02256287      0.02334652
```

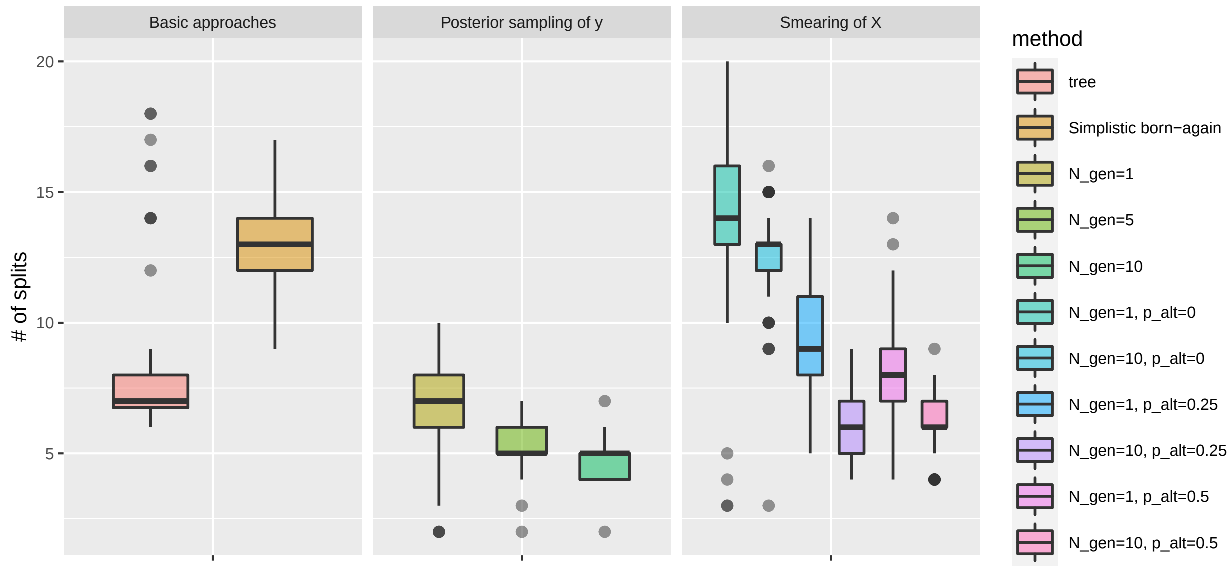
```
colMeans(tree_size, na.rm=TRUE)
```

```
##            tree            BART Born-again tree             N=1             N=5
##            7.81             NaN           12.83            6.52            5.17
##            N=10      N=1, palt=0      N=5, palt=0     N=10, palt=0  N=1, palt=0.25
##            4.92           14.33           12.71           12.49            9.16
##  N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##            6.36            6.12            8.28            6.38            6.31
```

```
sapply(tree_size, sd)
```

```
##            tree            BART Born-again tree             N=1             N=5
##       2.6041730              NA       1.4978436       1.6906158       0.8415354
##            N=10      N=1, palt=0      N=5, palt=0     N=10, palt=0  N=1, palt=0.25
##       0.7872725       2.7672508       1.8764624       1.6907054       2.0583410
##  N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##       1.2187094       1.1658439       1.7528044       1.0519823       1.1164066
```





Sonar

```
## Compute intercorrelation
data("Sonar")
p <- ncol(Sonar)-1
sum(cor(sapply(Sonar[ , -61L], function(x)
  if (!is.numeric(x)) as.numeric(x) else x))) / (p*(p-1))
```

```
## [1] 0.1166146
```

```
load(file = "Sonar MSE.Rda")
load(file = "Sonar acc.Rda")
load(file = "Sonar tree_size.Rda")
colMeans(MSE)
```

```
##            tree           BART Born-again tree            N=1            N=5
##       0.2141351      0.1468447       0.1847915      0.1998829      0.2053185
##            N=10     N=1, palt=0     N=5, palt=0    N=10, palt=0  N=1, palt=0.25
##       0.2060179      0.1911827       0.1832554      0.1832785      0.2019839
##  N=5, palt=0.25 N=10, palt=0.25    N=1, palt=0.5    N=5, palt=0.5  N=10, palt=0.5
##       0.2142539      0.2139775       0.2144150      0.2215831      0.2230961
```

```
which.min(colMeans(MSE[ , -2]))
```

```
## N=5, palt=0
##           7
```

```
which.min(colMeans(acc[ , -2]))
```

```
## N=10, palt=0
##            8
```

```r
sapply(MSE, sd)
```

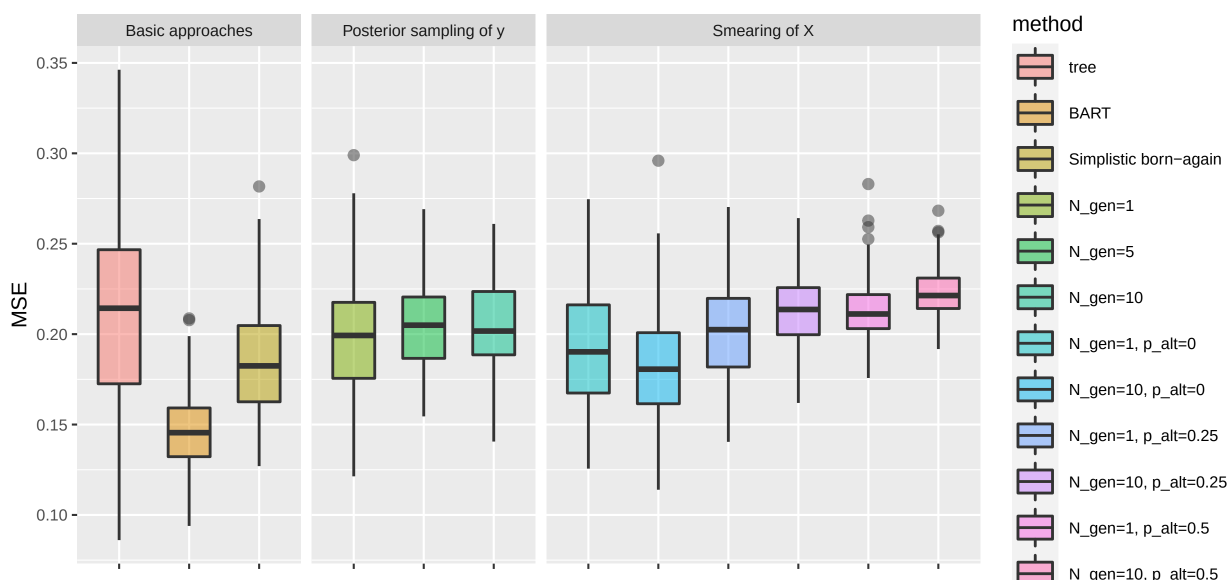
```
##             tree              BART Born-again tree               N=1               N=5
##       0.05597754        0.02335226        0.03209856        0.03057963        0.02687019
##             N=10      N=1, palt=0       N=5, palt=0      N=10, palt=0   N=1, palt=0.25
##       0.02553597        0.03340737        0.03102662        0.03316581        0.02696557
##  N=5, palt=0.25  N=10, palt=0.25     N=1, palt=0.5     N=5, palt=0.5   N=10, palt=0.5
##       0.02292906        0.02112953        0.01913660        0.01589785        0.01430024
```

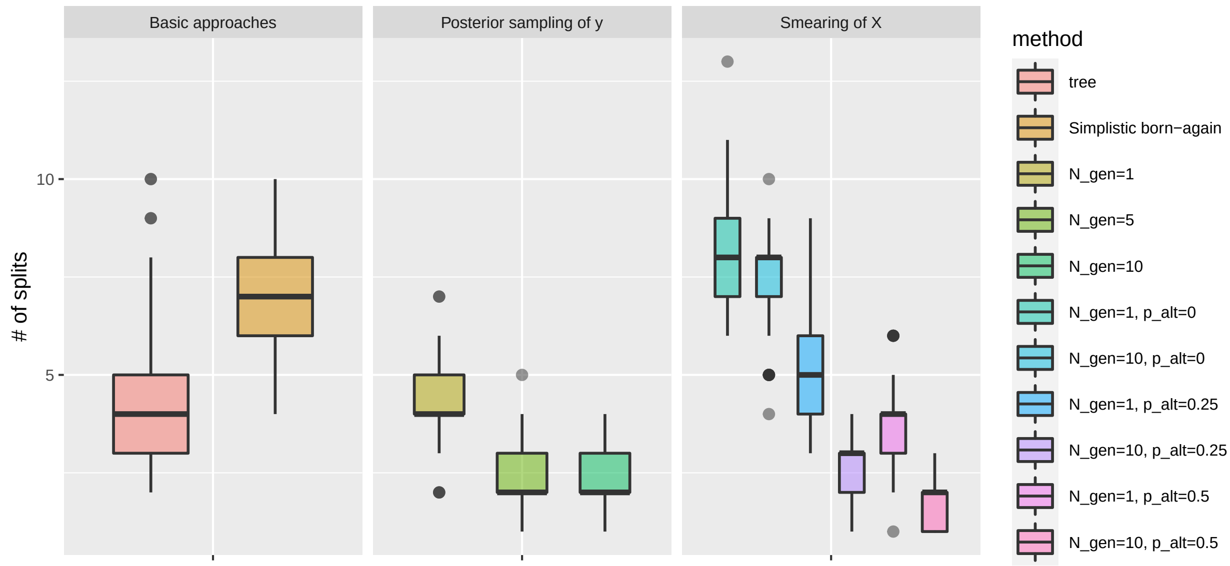
```r
colMeans(tree_size, na.rm=TRUE)
```

```
##             tree              BART Born-again tree               N=1               N=5
##             4.32               NaN              7.17              4.29              2.48
##             N=10      N=1, palt=0       N=5, palt=0      N=10, palt=0   N=1, palt=0.25
##             2.34              8.30              7.06              7.44              5.31
##  N=5, palt=0.25  N=10, palt=0.25     N=1, palt=0.5     N=5, palt=0.5   N=10, palt=0.5
##             2.82              2.54              3.64              1.91              1.69
```

```r
sapply(tree_size, sd)
```

```
##             tree              BART Born-again tree               N=1               N=5
##        1.7516515                NA         1.3857500         0.9877533         0.8584694
##             N=10      N=1, palt=0       N=5, palt=0      N=10, palt=0   N=1, palt=0.25
##        0.8787043         1.3521401         1.2698525         1.1833867         1.1780398
##  N=5, palt=0.25  N=10, palt=0.25     N=1, palt=0.5     N=5, palt=0.5   N=10, palt=0.5
##        0.6416519         0.6878454         1.0873004         0.6210939         0.6145541
```




Basic approaches
Posterior sampling of y
Smearing of X
# of splits
10
5
method
tree
Simplistic born−again
N_gen=1
N_gen=5
N_gen=10
N_gen=1, p_alt=0
N_gen=10, p_alt=0
N_gen=1, p_alt=0.25
N_gen=10, p_alt=0.25
N_gen=1, p_alt=0.5
N_gen=10, p_alt=0.5

Clustered / multilevel problems

I used a similar design as for the experiments above. However, with the very large sample sizes for the ECLSK Math and the Marriage dataset, this was computationally infeasible, so I selected small samples of observations and used the rest of the data as a test set.

For longitudinal data, I used 'new' observations for assessing predictive accuracy; i.e., training and test observations are from different clusters. For the `safety` data, training and test observations were forced to be from the same clusters. Of note, for all GLMM trees (both born-again and default), predictions were generated *excluding* the random effects. For BART, this does not seem possible. For generating BART predictions, both for the born-again approaches and accuracy assessment, the random effects were always included.

For permuting X in Breiman's born-again approach, I only permuted predictor variable values. The cluster indicator was *never* permuted.

For the ECLSK Math data, I additionally specified `months` as a linear predictor in the GLMM tree, to see if adding this information improves performance.

Of note, if `months` was completely omitted from all models, predictive accuracy was low, but BART performed better than GLMM tree and the born-again approach was effective.

Of note:

- I have no idea how BART generates random effects predictions for test observations from new folds.
- Observations weights < 1 are probably not used correctly by `lmer` / `glmer` in `lmertree` / `glmertree`. This will affect the size of the random effects (which will likely deviate more strongly from 0 than they should). This does not affect predictive accuracy too much, because random effects are not included in accuracy evaluation of the trees. But it will at least slightly affect the resulting tree.

Summary of findings

- Born-again approach seems to outperform posterior sampling and simplistic born-again approach.
- The simplistic born-again approach does already quite well, but not on ECLSK Math data.
- Non-zero values of p_{alt} perform well.
- It is very interesting to see that posterior sampling with $N_{gen} > 1$ worsens performance, unlike with the non-multilevel regression and classification problems above. A possible explanation is that the PPD of BART contains too much variation due to the random effects.

Safety data

```
load("Safety MSE.Rda")
load("Safety tree_size.Rda")
colMeans(MSE)
```

```
##             GLMM_tree                  Bart                    Ba
##             0.4973520             0.4454251             0.4619109
##           BaBayes_N=1           BaBayes_N=5          BaBayes_N=10
##             0.4618641             0.4662029             0.4669751
##     BaSmear_N=1_palt=0    BaSmear_N=5_palt=0   BaSmear_N=10_palt=0
```

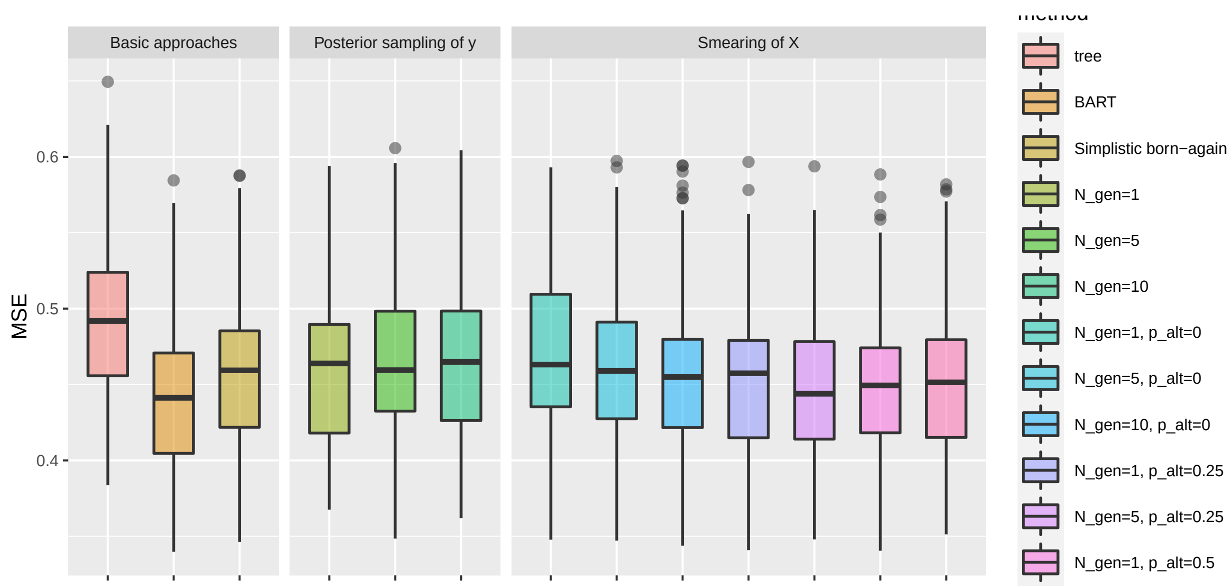
```
##                 0.4709109                0.4647347                0.4633411
##   BaSmear_N=1_palt=0.25   BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##                 0.4552308                0.4517064                0.4534223
##    BaSmear_N=1_palt=0.5    BaSmear_N=5_palt=0.5   BaSmear_N=10_palt=0.5
##                 0.4560568                0.4541583                0.4549779
```

```
which.min(colMeans(MSE[ , -2]))
```

```
## BaSmear_N=5_palt=0.25
##                    10
```

```
sapply(MSE, sd)
```

```
##                 GLMM_tree                     Bart                       Ba
##                0.06353793               0.05958080               0.05838785
##                BaBayes_N=1              BaBayes_N=5              BaBayes_N=10
##                0.05338970               0.05833726               0.05722090
##       BaSmear_N=1_palt=0      BaSmear_N=5_palt=0     BaSmear_N=10_palt=0
##                0.06105648               0.05853945               0.05806369
##   BaSmear_N=1_palt=0.25   BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##                0.05899406               0.05603396               0.05855780
##    BaSmear_N=1_palt=0.5    BaSmear_N=5_palt=0.5   BaSmear_N=10_palt=0.5
##                0.05760549               0.05930226               0.05806632
```



ECLSK Math

```
load("ECLSK Math MSE.Rda")
load("ECLSK Math tree_size.Rda")
colMeans(MSE, na.rm=TRUE)
```

```
##                GLMM_tree                   Bart                     Ba
##                0.1484964              0.1181759              0.1455195
##              BaBayes_N=1            BaBayes_N=5           BaBayes_N=10
##                0.1387246              0.1531658              0.1542694
##       BasSmear_N=1_palt=0     BasSmear_N=5_palt=0    BasSmear_N=10_palt=0
##                0.1575133              0.1611703              0.1626692
##   BasSmear_N=1_palt=0.25  BasSmear_N=5_palt=0.25 BasSmear_N=10_palt=0.25
##                0.1317319             48.3380399              0.1443590
##    BasSmear_N=1_palt=0.5   BasSmear_N=5_palt=0.5  BasSmear_N=10_palt=0.5
##                0.1313846              0.1340787              0.1357035
##          GLMM_tree(time)
##                0.1425245
```

```
## Omit crazy outlier; don't know how such extreme value is even possible
MSE$`BasSmear_N=5_palt=0.25`[MSE$`BasSmear_N=5_palt=0.25` > 100] <- NA
which.min(colMeans(MSE[ , -c(2, 16)], na.rm = TRUE))
```

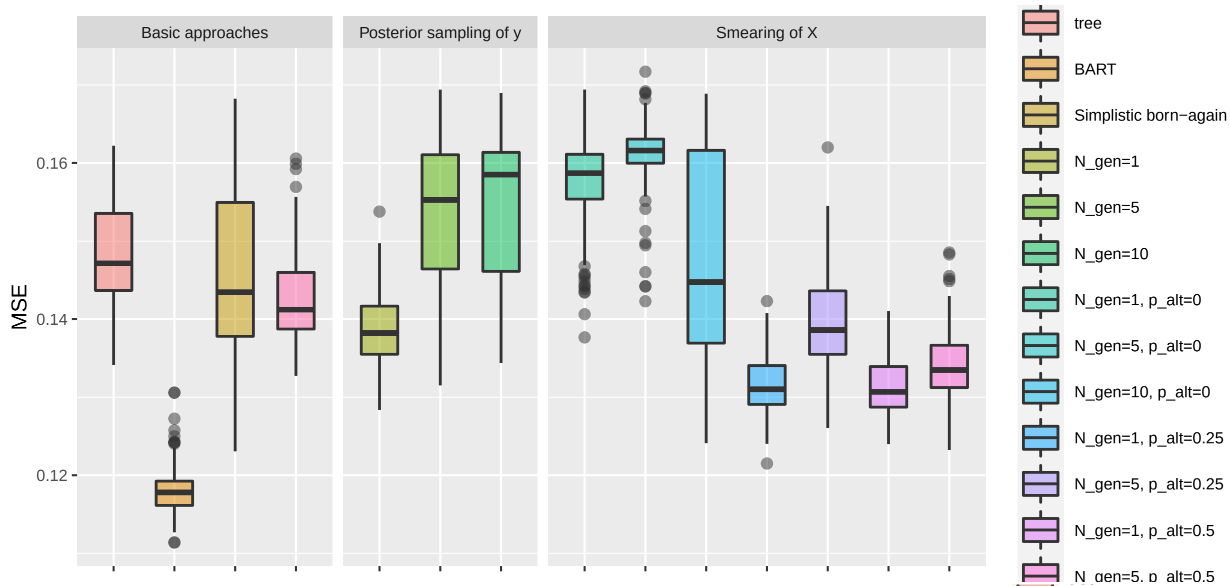
```
## BasSmear_N=1_palt=0.5
##                    12
```

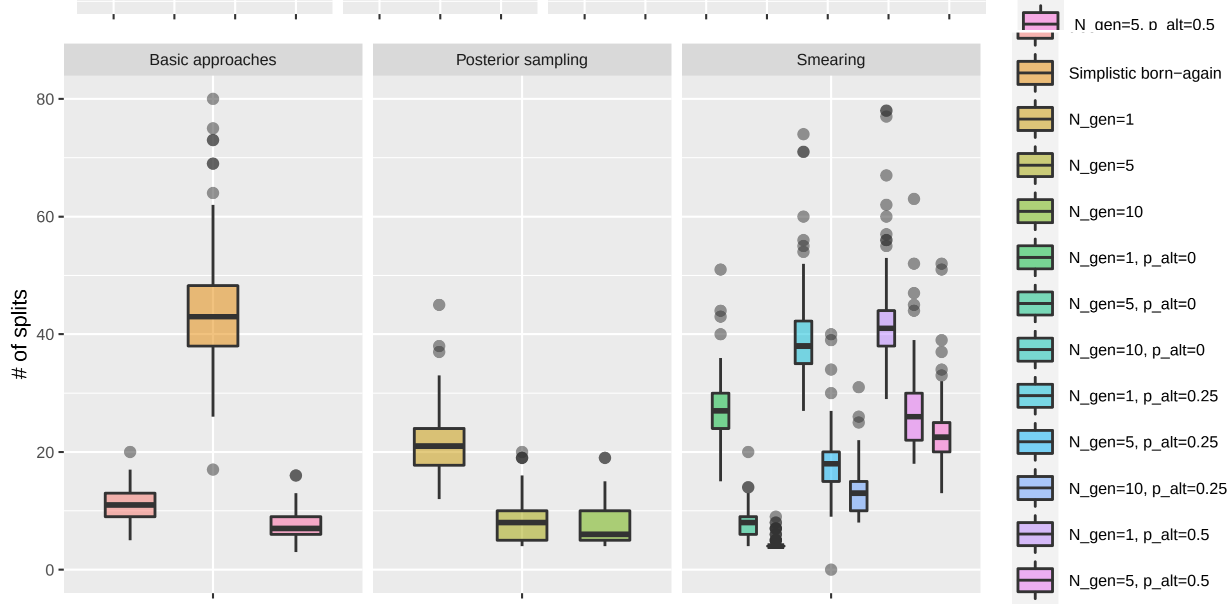
```
round(sapply(MSE, sd, na.rm=TRUE), digits = 4)
```

```
##                GLMM_tree                   Bart                     Ba
##                   0.0065                 0.0033                 0.0102
##              BaBayes_N=1            BaBayes_N=5           BaBayes_N=10
##                   0.0047                 0.0089                 0.0087
##       BasSmear_N=1_palt=0     BasSmear_N=5_palt=0    BasSmear_N=10_palt=0
##                   0.0060                 0.0046                 0.0018
##   BasSmear_N=1_palt=0.25  BasSmear_N=5_palt=0.25 BasSmear_N=10_palt=0.25
##                   0.0038                 0.0056                 0.0058
##    BasSmear_N=1_palt=0.5   BasSmear_N=5_palt=0.5  BasSmear_N=10_palt=0.5
##                   0.0037                 0.0045                 0.0046
##          GLMM_tree(time)
##                   0.0057
```

```
sapply(MSE[ , 1:5], function(x) table(is.na(x)))
```

```
##    GLMM_tree.FALSE        Bart.FALSE          Ba.FALSE BaBayes_N=1.FALSE
##                128               128               128               128
## BaBayes_N=5.FALSE
##                128
```





Marriage

```r
load("Marriage MSE.Rda")
load("Marriage tree_size.Rda")
colMeans(MSE, na.rm=TRUE)
```

```
##             GLMM_tree                 Bart                   Ba
##             0.2162583            0.2095577            0.2127169
##           BaBayes_N=1          BaBayes_N=5         BaBayes_N=10
##             0.2129928            0.2145004            0.2144389
##      BaSmear_N=1_palt=0   BaSmear_N=5_palt=0  BaSmear_N=10_palt=0
##             0.2128367            0.2131265            0.2128848
##   BaSmear_N=1_palt=0.25 BaSmear_N=5_palt=0.25 BaSmear_N=10_palt=0.25
##             0.2114904            0.2116715            0.2117098
```

```
##    BaSmear_N=1_palt=0.5    BaSmear_N=5_palt=0.5   BaSmear_N=10_palt=0.5
##               0.2114497               0.2114502               0.2114618
```

```
which.min(colMeans(MSE[ , -2], na.rm = TRUE))
```

```
## BaSmear_N=1_palt=0.5
##                   12
```

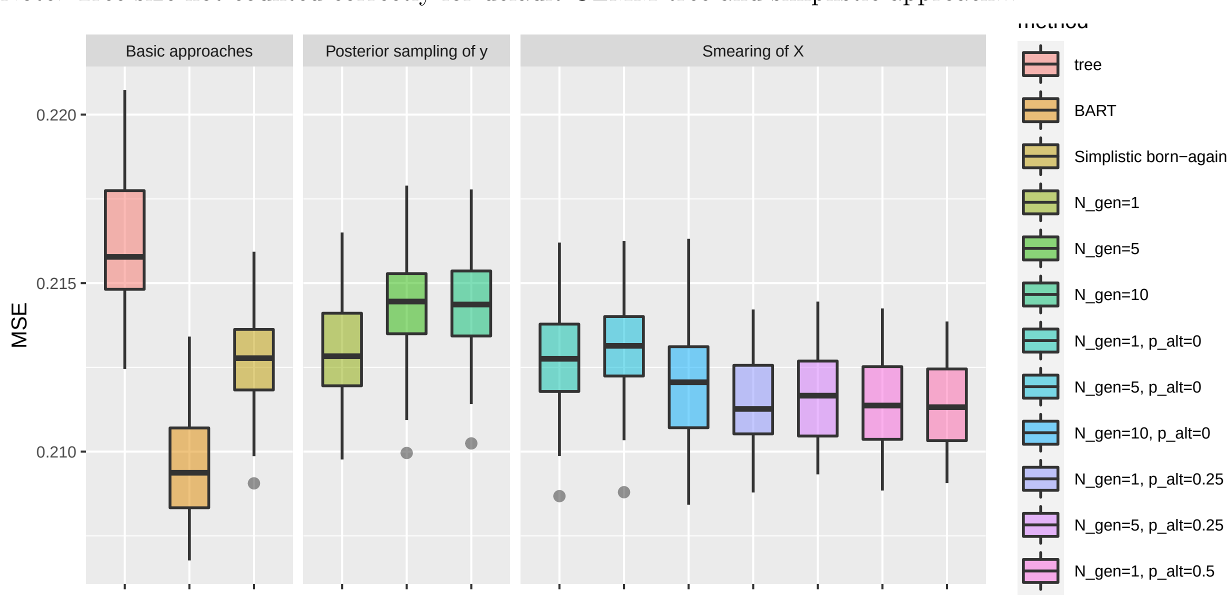
```
sapply(MSE, sd, na.rm=TRUE)
```

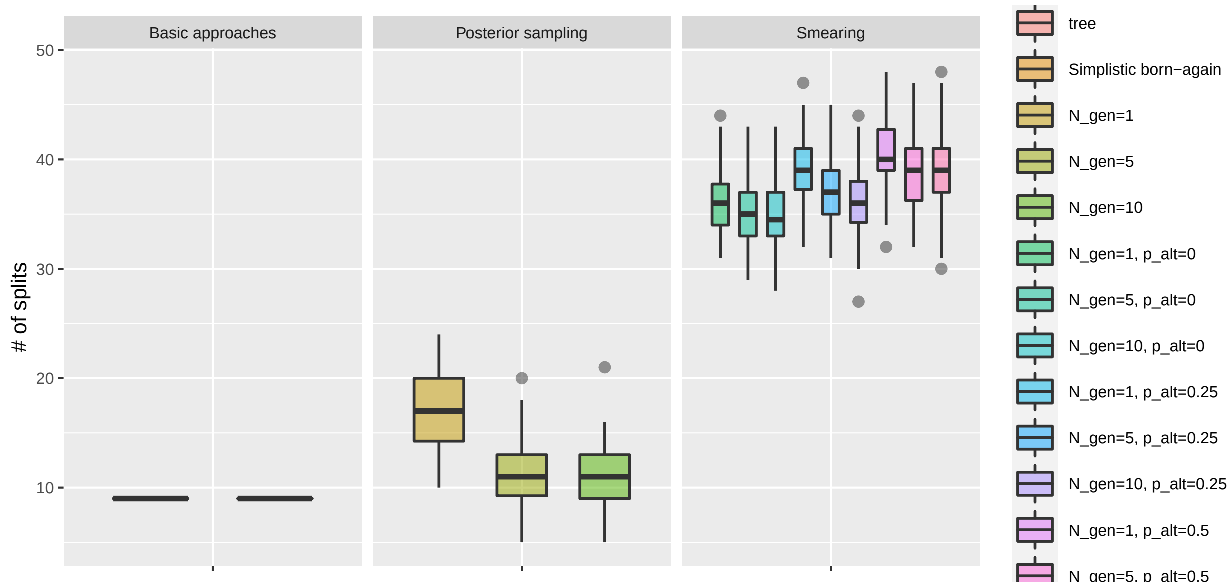
```
##               GLMM_tree                   Bart                     Ba
##             0.002280889             0.001465275             0.001576410
##             BaBayes_N=1             BaBayes_N=5            BaBayes_N=10
##             0.001547609             0.001685830             0.001622935
##      BaSmear_N=1_palt=0      BaSmear_N=5_palt=0     BaSmear_N=10_palt=0
##             0.001756477             0.001534353             0.001640853
##   BaSmear_N=1_palt=0.25   BaSmear_N=5_palt=0.25  BaSmear_N=10_palt=0.25
##             0.001403057             0.001380516             0.001339079
##    BaSmear_N=1_palt=0.5    BaSmear_N=5_palt=0.5   BaSmear_N=10_palt=0.5
##             0.001397754             0.001316614             0.001395660
```

```
sapply(MSE[ , 1:5], function(x) table(is.na(x)))
```

```
##    GLMM_tree.FALSE         Bart.FALSE           Ba.FALSE BaBayes_N=1.FALSE
##                 50                 50                 50                50
## BaBayes_N=5.FALSE
##                 50
```

Note: Tree size not counted correctly for default GLMM tree and simplistic approach!!!




Basic approaches
Posterior sampling
Smearing
# of splits
50
40
30
20
10
tree
Simplistic born−again
N_gen=1
N_gen=5
N_gen=10
N_gen=1, p_alt=0
N_gen=5, p_alt=0
N_gen=10, p_alt=0
N_gen=1, p_alt=0.25
N_gen=5, p_alt=0.25
N_gen=10, p_alt=0.25
N_gen=1, p_alt=0.5
N_gen=5, p_alt=0.5