Compare_methods.R

gjo15

Fri Jul 06 16:31:11 2018

```
# Summarize Parameter Effect on Model Accuracy
# Gabriel Odom
# 2018-07-05

# Given the DMR method comparison results in the DMRcompare package, assess the
#   strengths and directions between the model parameters and performance
#   metrics for each of the four methods.
library(DMRcompare)
```

```
## Loading required package: ChAMPdata
## Loading required package: GenomicRanges
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
## Loading required package: S4Vectors
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##     expand.grid
## Loading required package: IRanges
```

```
##
## Attaching package: 'IRanges'

## The following object is masked from 'package:grDevices':
##
##     windows

## Loading required package: GenomeInfoDb

## Loading required package: DMRcatedata

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)

## Warning: replacing previous import 'plyr::summarise' by 'plotly::summarise'
## when loading 'ChAMP'

## Warning: replacing previous import 'plyr::rename' by 'plotly::rename' when
## loading 'ChAMP'

## Warning: replacing previous import 'plyr::arrange' by 'plotly::arrange'
## when loading 'ChAMP'

## Warning: replacing previous import 'plyr::mutate' by 'plotly::mutate' when
## loading 'ChAMP'

##

##

## Warning: replacing previous import 'igraph::edges' by 'graph::edges' when
## loading 'FEM'

## Warning: replacing previous import 'igraph::intersection' by
## 'graph::intersection' when loading 'FEM'

## Warning: replacing previous import 'igraph::degree' by 'graph::degree' when
## loading 'FEM'

## Warning: replacing previous import 'igraph::union' by 'graph::union' when
## loading 'FEM'

## Warning: replacing previous import 'limma::plotMA' by
## 'BiocGenerics::plotMA' when loading 'FEM'

## Warning: replacing previous import 'igraph::path' by 'BiocGenerics::path'
## when loading 'FEM'

## Warning: replacing previous import 'Matrix::colSums' by
## 'BiocGenerics::colSums' when loading 'FEM'

## Warning: replacing previous import 'Matrix::colMeans' by
## 'BiocGenerics::colMeans' when loading 'FEM'

## Warning: replacing previous import 'Matrix::rowMeans' by
## 'BiocGenerics::rowMeans' when loading 'FEM'

## Warning: replacing previous import 'Matrix::rowSums' by
## 'BiocGenerics::rowSums' when loading 'FEM'

## Warning: replacing previous import 'Matrix::which' by 'BiocGenerics::which'
## when loading 'FEM'
```

```
## Warning: replacing previous import 'igraph::normalize' by
## 'BiocGenerics::normalize' when loading 'FEM'

## Warning: replacing previous import 'plyr::is.discrete' by
## 'Hmisc::is.discrete' when loading 'ChAMP'

## Warning: replacing previous import 'plyr::summarize' by 'Hmisc::summarize'
## when loading 'ChAMP'

## Warning: replacing previous import 'plotly::subplot' by 'Hmisc::subplot'
## when loading 'ChAMP'

## Warning: replacing previous import 'GenomicRanges::sort' by
## 'globaltest::sort' when loading 'ChAMP'

## Warning: replacing previous import 'globaltest::model.matrix' by
## 'stats::model.matrix' when loading 'ChAMP'

## Warning: replacing previous import 'globaltest::p.adjust' by
## 'stats::p.adjust' when loading 'ChAMP'
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------

## v ggplot2 2.2.1     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.5
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0

## -- Conflicts ------------------------------------------------------------------------------------
## x dplyr::collapse()   masks IRanges::collapse()
## x dplyr::combine()    masks BiocGenerics::combine()
## x dplyr::desc()       masks IRanges::desc()
## x tidyr::expand()     masks S4Vectors::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks S4Vectors::first()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()     masks S4Vectors::rename()
## x dplyr::slice()      masks IRanges::slice()
```

```
######  DMRcate  ##################################################################
data("dmrcateRes_df")
colnames(dmrcateRes_df)
```

```
##  [1] "method"    "delta"     "seed"      "lambda"    "C"
##  [6] "time"      "FN"        "FP"        "TN"        "TP"
## [11] "power"     "nPower"    "AuPR"      "FPprecis"  "TPprecis"
## [16] "precision" "nPrecis"   "mcc"       "F1"        "nCPG_q1"
## [21] "nCPG_med"  "nCPG_q3"
```

```
# Model components:
#   Predictors: lambda and C
#   Responses : power, AuPR, precision, mcc, and F1
#   Covariates: delta, nCPG_med, nCPG_q3
resultsDMRcate_df <-
  dmrcateRes_df %>%
  select(power, AuPR, precision, mcc, F1,
```

```
        lambda, C,
        delta, seed, nCPG_med, nCPG_q3) %>%
  mutate(nCPG_med = as.numeric(nCPG_med)) %>%
  mutate(nCPG_q3 = as.numeric(nCPG_q3))
```

```
## Warning in eval(substitute(expr), envir, enclos): NAs introduced by
## coercion
```

```
###  Replace Missing Values with 0  ###
# For each of the responses, 0 is as bad as it gets. Therefore, we can replace
#   all NAs and NaNs with 0. However, precision start at 1 and never reach 0,
#   so we replace all of the 0s for precision with 1.
resultsDMRcate_df <- resultsDMRcate_df[complete.cases(resultsDMRcate_df), ]


###  Correlation Matrix  ###
# First we build a correlation matrix to inspect possible multicollinearity.
round(cor(resultsDMRcate_df), 2)
```

```
##           power  AuPR precision   mcc    F1 lambda     C delta  seed
## power      1.00  1.00     -0.56  0.99  0.99  -0.14  0.18  0.79 -0.02
## AuPR       1.00  1.00     -0.57  1.00  1.00  -0.15  0.17  0.77 -0.03
## precision -0.56 -0.57      1.00 -0.57 -0.59  -0.34  0.20 -0.50 -0.18
## mcc        0.99  1.00     -0.57  1.00  1.00  -0.15  0.17  0.75 -0.03
## F1         0.99  1.00     -0.59  1.00  1.00  -0.13  0.15  0.75 -0.02
## lambda    -0.14 -0.15     -0.34 -0.15 -0.13   1.00  0.00  0.00  0.00
## C          0.18  0.17      0.20  0.17  0.15   0.00  1.00  0.00  0.00
## delta      0.79  0.77     -0.50  0.75  0.75   0.00  0.00  1.00  0.00
## seed      -0.02 -0.03     -0.18 -0.03 -0.02   0.00  0.00  0.00  1.00
## nCPG_med   0.08  0.10     -0.57  0.12  0.13   0.67 -0.51  0.18 -0.01
## nCPG_q3    0.15  0.16     -0.61  0.17  0.19   0.66 -0.49  0.25 -0.04
##           nCPG_med nCPG_q3
## power         0.08    0.15
## AuPR          0.10    0.16
## precision    -0.57   -0.61
## mcc           0.12    0.17
## F1            0.13    0.19
## lambda        0.67    0.66
## C            -0.51   -0.49
## delta         0.18    0.25
## seed         -0.01   -0.04
## nCPG_med      1.00    0.91
## nCPG_q3       0.91    1.00
```

```
# We see that all of the responses (except for precision) are nearly perfectly
#   correlated with each other. Precision is rather strongly negatively
#   correlated with the other output variables, but not perfectly (-0.6). This
#   means we can simply model the AuPR instead. Further, it appears that
#   neither C nor lambda are correlated with the outcome at all. These two nCPG
#   values are also highly correlated. Thankfully, the seed has no effect.
resultsDMRcate_df <-
  resultsDMRcate_df %>%
  select(AuPR, lambda, C, delta, nCPG_q3)
cor(resultsDMRcate_df)
```

```
##                AuPR     lambda          C     delta    nCPG_q3
## AuPR     1.0000000 -0.1450766  0.1701860 0.7703374  0.1611232
## lambda  -0.1450766  1.0000000  0.0000000 0.0000000  0.6570122
## C        0.1701860  0.0000000  1.0000000 0.0000000 -0.4889496
## delta    0.7703374  0.0000000  0.0000000 1.0000000  0.2537752
## nCPG_q3  0.1611232  0.6570122 -0.4889496 0.2537752  1.0000000
```

```
# We've removed any multicollinearity issues.
cor(resultsDMRcate_df, method = "spearman")
```

```
##                AuPR     lambda          C     delta     nCPG_q3
## AuPR     1.00000000 -0.1889265  0.2586212 0.8895525  0.03372137
## lambda  -0.18892654  1.0000000  0.0000000 0.0000000  0.67941950
## C        0.25862122  0.0000000  1.0000000 0.0000000 -0.46456863
## delta    0.88955250  0.0000000  0.0000000 1.0000000  0.31342727
## nCPG_q3  0.03372137  0.6794195 -0.4645686 0.3134273  1.00000000
```

```
# Now, the correlation test:
cor.test(resultsDMRcate_df$lambda,
         resultsDMRcate_df$AuPR,
         method = "spearman")$p.value
```

```
## Warning in cor.test.default(resultsDMRcate_df$lambda, resultsDMRcate_df
## $AuPR, : Cannot compute exact p-value with ties
```

```
## [1] 1.788338e-08
```

```
cor.test(resultsDMRcate_df$C,
         resultsDMRcate_df$AuPR,
         method = "spearman")$p.value
```

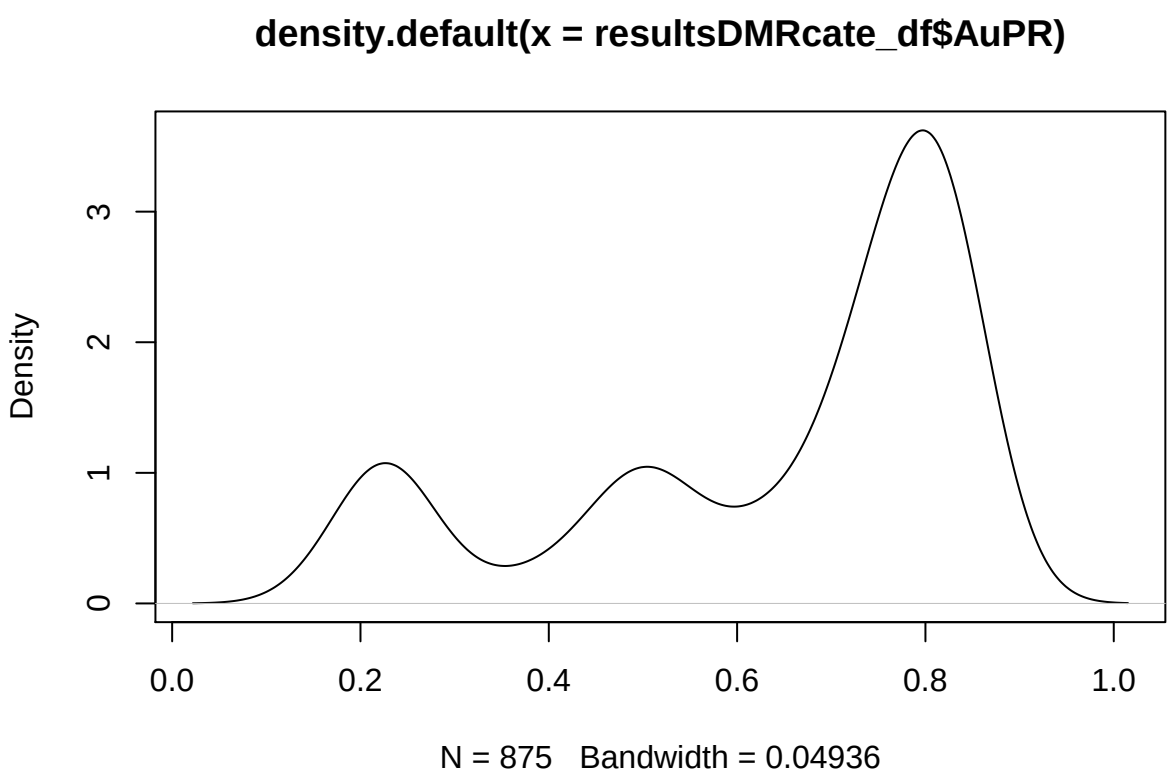
```
## Warning in cor.test.default(resultsDMRcate_df$C, resultsDMRcate_df$AuPR, :
## Cannot compute exact p-value with ties
```

```
## [1] 7.739108e-15
```

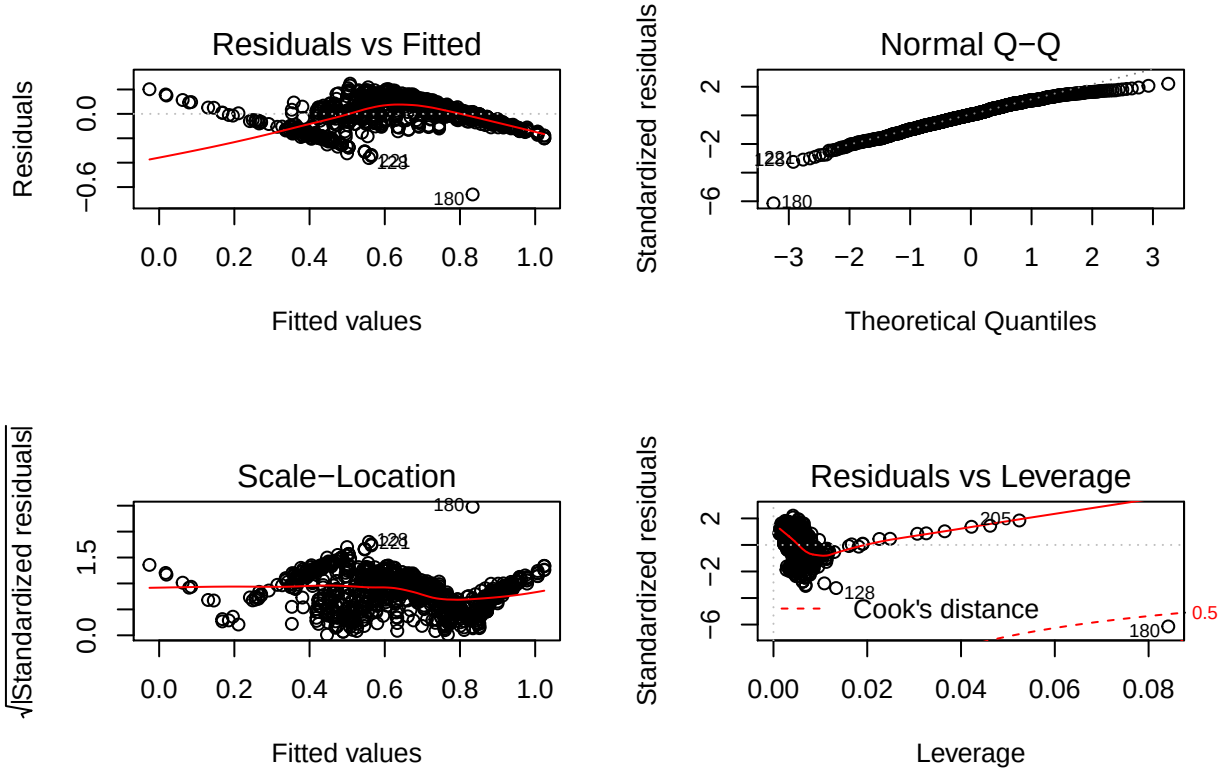
```
# Both are related.


###  Statistical Prediction for AuPR  ###
summary(resultsDMRcate_df$AuPR)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1700  0.5033  0.7406  0.6394  0.8077  0.8670
```

```
plot(density(resultsDMRcate_df$AuPR))
```




```
# The AuPR values can only range from 0-1, so I think we should be using
#   beta regression if the residuals of OLS do not appear normal.
dmrcateLin_mod <- lm(AuPR ~ ., data = resultsDMRcate_df)
par(mfrow = c(2, 2))
plot(dmrcateLin_mod)
```




```
# Well that's a resounding "NO". However, we see that observation 180 is a
#   severe outlier. Let's check it.
dmrcateRes_df[178:180, ]
```

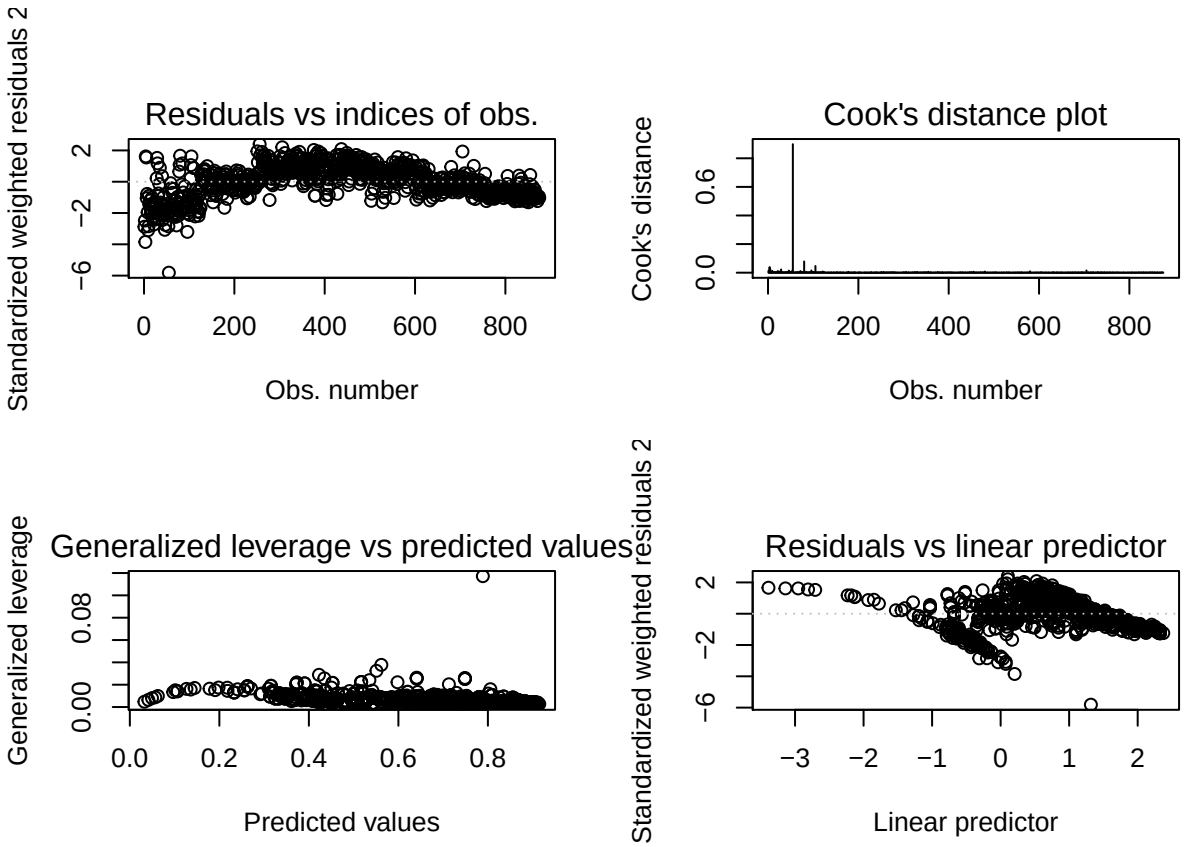
```
##      method delta seed lambda C   time  FN FP   TN TP power nPower
## 178 DMRcate 0.025  330    500 1 137.66 492  0 2563  8 0.016    500
## 179 DMRcate 0.025  330    750 1 139.80 496  0 2563  4 0.008    500
## 180 DMRcate 0.025  330   1000 1 139.25 497  0 2563  3 0.006    500
##           AuPR FPprecis TPprecis precision nPrecis        mcc         F1
## 178 0.1873283        0        8         1       8 0.11585871 0.03149606
## 179 0.1760648        0        4         1       4 0.08187090 0.01587302
## 180 0.1730991        0        3         1       3 0.07089069 0.01192843
##     nCPG_q1 nCPG_med nCPG_q3
## 178    6.75      8.5      11
## 179    6.75      7.5    14.5
## 180       7        8    21.5
```

```
# I don't see anything wrong with it, but  it's apparently very influential.

# Beta regression it is. The default link is logit.
library(betareg)
```

```
## Warning: package 'betareg' was built under R version 3.5.1
```

```
resultsDMRcate_df <-
  resultsDMRcate_df %>%
  mutate(sigma = lambda / C)
dmrcateBeta_mod <- betareg(AuPR ~ ., data = resultsDMRcate_df)
```

```r
plot(dmrcateBeta_mod)
```

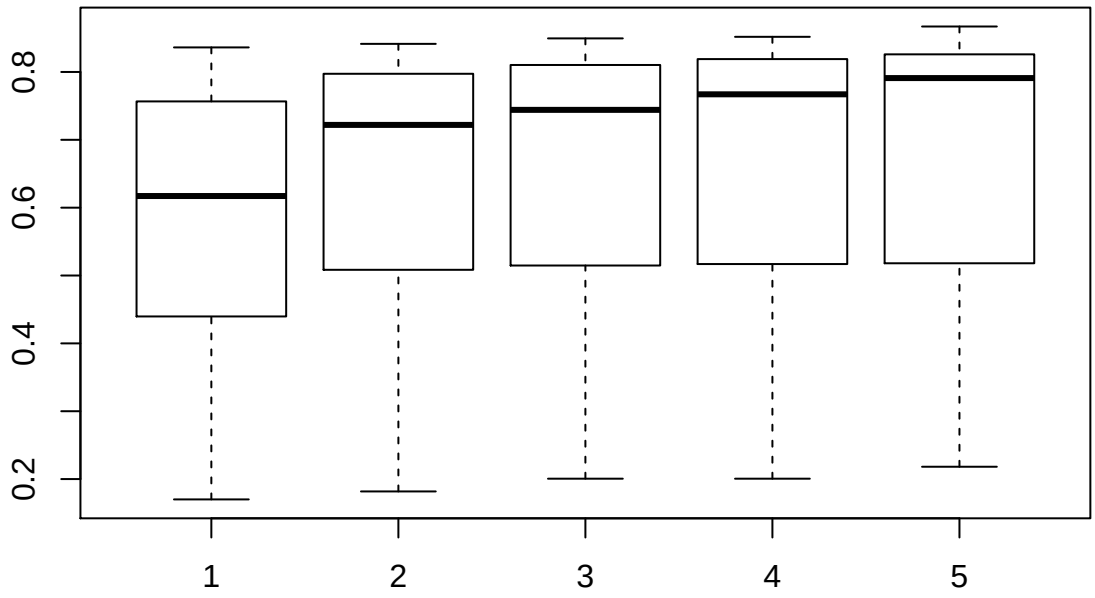



```r
summary(dmrcateBeta_mod)
```

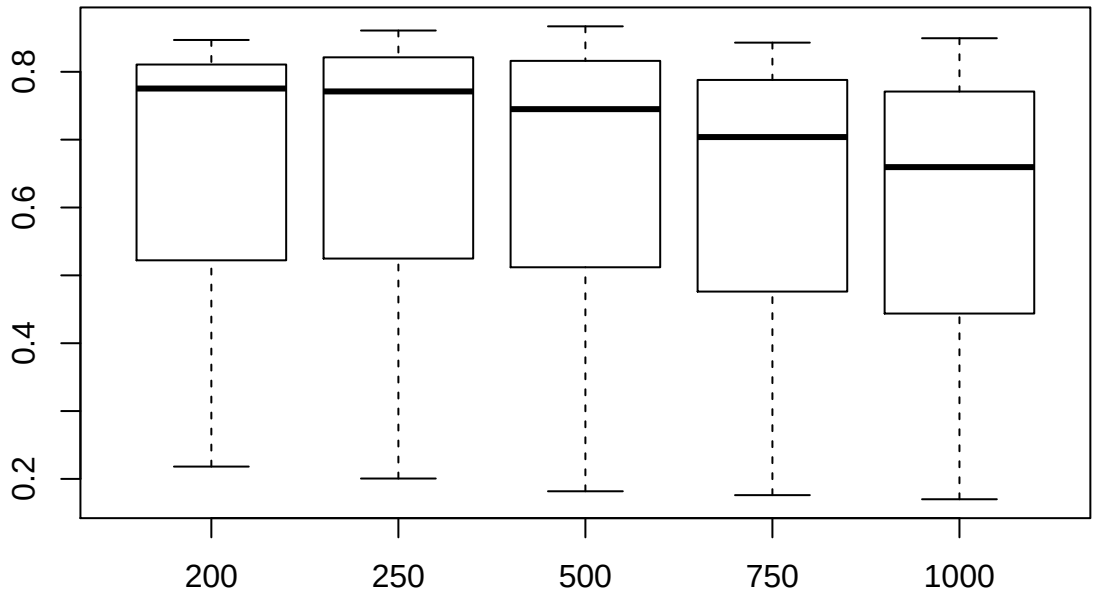
```
##
## Call:
## betareg(formula = AuPR ~ ., data = resultsDMRcate_df)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -5.8122 -0.6919  0.0018  0.7068  2.3983
##
## Coefficients (mean model with logit link):
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -3.379e+00  1.764e-01  -19.15   <2e-16 ***
## lambda      -1.201e-03  9.878e-05  -12.16   <2e-16 ***
## C            1.972e-01  1.937e-02   10.18   <2e-16 ***
## delta        5.121e+00  1.515e-01   33.79   <2e-16 ***
## nCPG_q3      3.302e-01  1.788e-02   18.46   <2e-16 ***
## sigma       -1.527e-03  1.448e-04  -10.55   <2e-16 ***
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)   23.194      1.093   21.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 870.9 on 7 Df
## Pseudo R-squared: 0.7659
## Number of iterations: 12 (BFGS) + 2 (Fisher scoring)
```

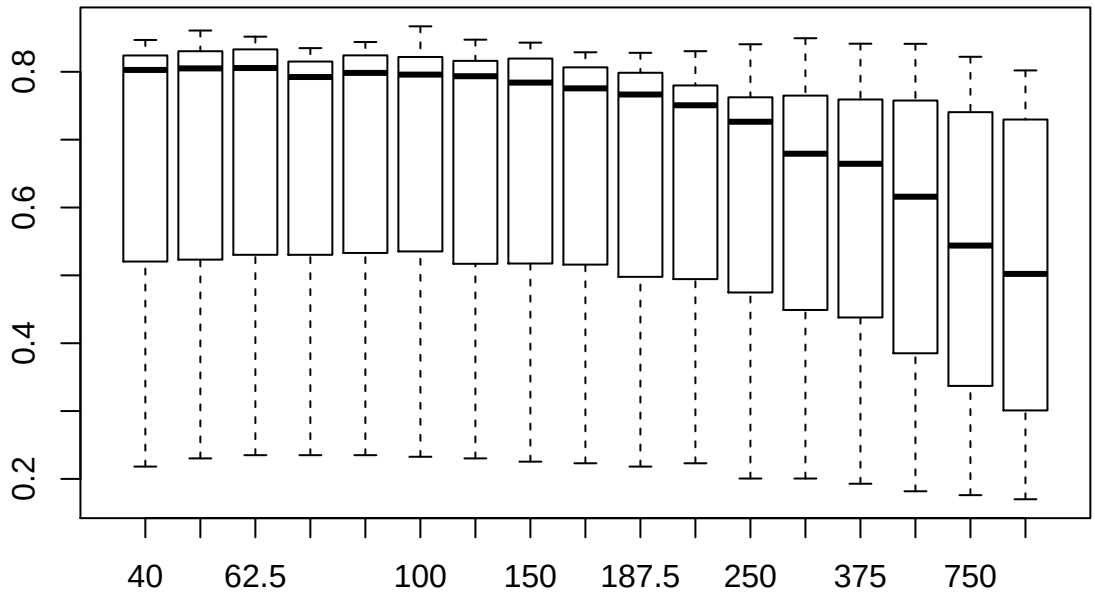
```r
# The directions of the relationships jive with the Spearman correlations. What
#   we can then say is this: given delta and the number of CPGs, for each unit
#   increase in lambda, logit(AuPR) decreases by 0.18892654; for each unit
#   increase of C, logit(AuPR) increases by 0.25862122. Also, the interaction
#   between C and lambda (sigma) is significant: as lambda / C increases, AuPR
#   decreases.

# Let's have a sanity check:
par(mfrow = c(1,1))
boxplot(resultsDMRcate_df$AuPR ~ resultsDMRcate_df$C)
```




```r
# I'll buy the results that C is positively related with AuPC, but not that the
#   relationship is logit-linear
boxplot(resultsDMRcate_df$AuPR ~ resultsDMRcate_df$lambda)
```




```
# Same as before, but with the negative relationship.
boxplot(resultsDMRcate_df$AuPR ~ resultsDMRcate_df$sigma)
```




```
# Smaller lambda / C is better


######  ProbeLasso  ##########################################################
data("probeLassoRes_df")
colnames(probeLassoRes_df)
```

```
##  [1] "method"    "delta"     "seed"      "adjPval"   "mLassoRad"
##  [6] "minDmrSep" "time"      "FN"        "FP"        "TN"
## [11] "TP"        "power"     "nPower"    "AuPR"      "FPprecis"
## [16] "TPprecis"  "precision" "nPrecis"   "mcc"       "F1"
## [21] "nCPG_q1"   "nCPG_med"  "nCPG_q3"
```

```
# Model components:
#   Predictors: adjPval, mLassoRad, and minDmrSep
#   Responses : power, AuPR, precision, mcc, and F1
#   Covariates: delta, nCPG_med, nCPG_q3
resultsPL_df <-
  probeLassoRes_df %>%
  select(power, AuPR, precision, mcc, F1,
         adjPval, mLassoRad, minDmrSep,
         delta, seed, nCPG_med, nCPG_q3) %>%
  mutate(nCPG_med = as.numeric(nCPG_med)) %>%
  mutate(nCPG_q3 = as.numeric(nCPG_q3))
```

```
## Warning in eval(substitute(expr), envir, enclos): NAs introduced by
## coercion
```

```
###  Replace Missing Values with 0  ###
# For each of the responses, 0 is as bad as it gets. Therefore, we can replace
#   all NAs and NaNs with 0. However, precision start at 1 and never reach 0,
#   so we replace all of the 0s for precision with 1.
resultsPL_df <- resultsPL_df[complete.cases(resultsPL_df), ]

###  Correlation Matrix  ###
# First we build a correlation matrix to inspect possible multicollinearity.
round(cor(resultsPL_df), 2)
```

```
##           power  AuPR precision   mcc    F1 adjPval mLassoRad minDmrSep
## power      1.00  0.99     -0.53  0.98  0.98    0.02      0.56      0.00
## AuPR       0.99  1.00     -0.51  0.99  0.99    0.01      0.50      0.00
## precision -0.53 -0.51      1.00 -0.51 -0.50    0.17     -0.65     -0.02
## mcc        0.98  0.99     -0.51  1.00  1.00   -0.02      0.44      0.00
## F1         0.98  0.99     -0.50  1.00  1.00    0.00      0.45      0.00
## adjPval    0.02  0.01      0.17 -0.02  0.00    1.00      0.00      0.00
## mLassoRad  0.56  0.50     -0.65  0.44  0.45    0.00      1.00      0.00
## minDmrSep  0.00  0.00     -0.02  0.00  0.00    0.00      0.00      1.00
## delta      0.58  0.59     -0.06  0.58  0.58   -0.07      0.00      0.00
## seed       0.02  0.03      0.11  0.04  0.03   -0.02      0.00      0.00
## nCPG_med  -0.11 -0.18     -0.05 -0.28 -0.25    0.14      0.53      0.00
## nCPG_q3    0.22  0.17     -0.61  0.12  0.13   -0.05      0.77      0.06
##            delta  seed nCPG_med nCPG_q3
## power       0.58  0.02    -0.11    0.22
## AuPR        0.59  0.03    -0.18    0.17
## precision -0.06  0.11    -0.05   -0.61
## mcc         0.58  0.04    -0.28    0.12
## F1          0.58  0.03    -0.25    0.13
## adjPval    -0.07 -0.02     0.14   -0.05
## mLassoRad   0.00  0.00     0.53    0.77
## minDmrSep   0.00  0.00     0.00    0.06
## delta       1.00  0.02    -0.26   -0.20
## seed        0.02  1.00    -0.04   -0.03
## nCPG_med   -0.26 -0.04     1.00    0.61
## nCPG_q3    -0.20 -0.03     0.61    1.00
```

```
# Basically the same story as DMRcate. Because we have the mean Lasso radius as
#   one of our predictors, we will take the median number of CPGs instead of
#   upper quartile as our block effect.
resultsPL_df <-
  resultsPL_df %>%
  select(AuPR, adjPval, mLassoRad, minDmrSep, delta, nCPG_q3)
# We've removed any multicollinearity issues, except for the nCPG with mLasRad
cor(resultsPL_df, method = "spearman")
```

```
##                  AuPR     adjPval mLassoRad   minDmrSep       delta
## AuPR       1.000000000  0.06948835 0.6148738 0.006859731  0.64223538
## adjPval    0.069488348  1.00000000 0.0000000 0.000000000 -0.07349834
## mLassoRad  0.614873794  0.00000000 1.0000000 0.000000000  0.00000000
## minDmrSep  0.006859731  0.00000000 0.0000000 1.000000000  0.00000000
## delta      0.642235382 -0.07349834 0.0000000 0.000000000  1.00000000
## nCPG_q3    0.345081448 -0.05957444 0.8292860 0.060095038 -0.18952114
##               nCPG_q3
## AuPR       0.34508145
```

```
## adjPval   -0.05957444
## mLassoRad  0.82928604
## minDmrSep  0.06009504
## delta     -0.18952114
## nCPG_q3    1.00000000
```

```
# Now, the correlation test:
cor.test(resultsPL_df$adjPval,
         resultsPL_df$AuPR,
         method = "spearman")$p.value # related
```

```
## Warning in cor.test.default(resultsPL_df$adjPval, resultsPL_df$AuPR, method
## = "spearman"): Cannot compute exact p-value with ties
```

```
## [1] 0.002713263
```

```
cor.test(resultsPL_df$mLassoRad,
         resultsPL_df$AuPR,
         method = "spearman")$p.value # related
```

```
## Warning in cor.test.default(resultsPL_df$mLassoRad, resultsPL_df$AuPR,
## method = "spearman"): Cannot compute exact p-value with ties
```

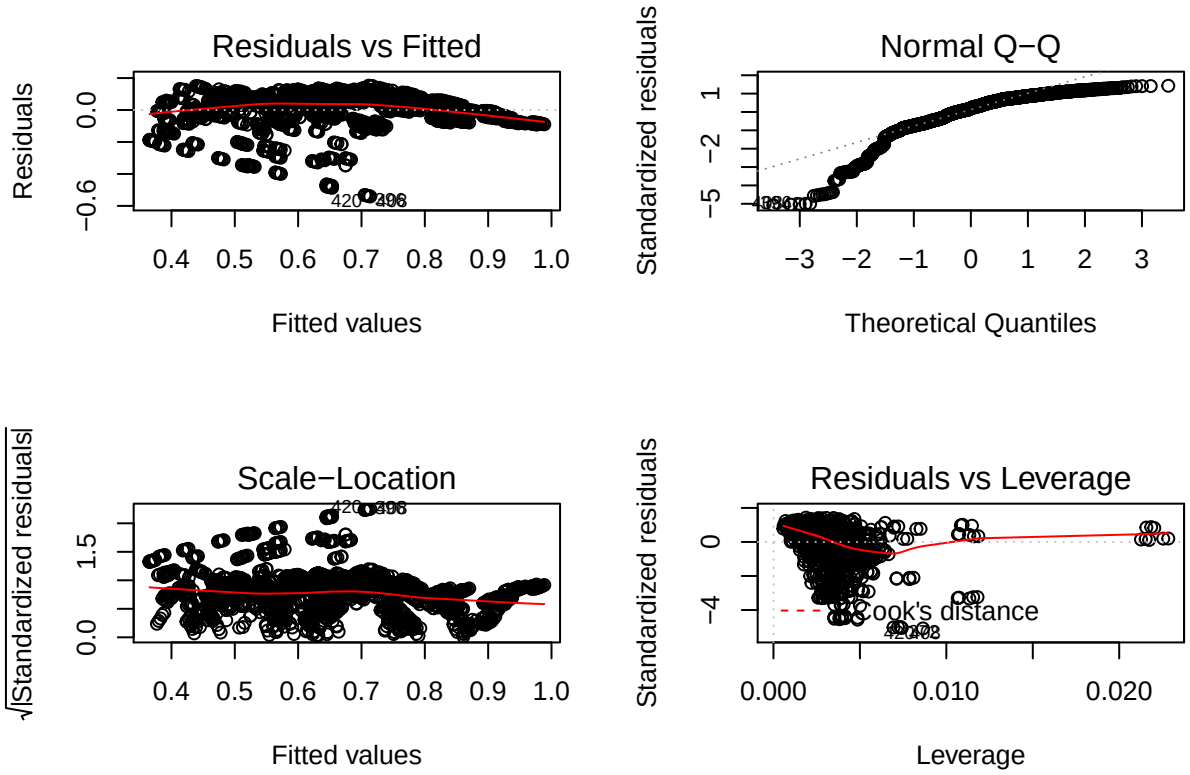
```
## [1] 7.314597e-194
```

```
cor.test(resultsPL_df$minDmrSep,
         resultsPL_df$AuPR,
         method = "spearman")$p.value # not related
```

```
## Warning in cor.test.default(resultsPL_df$minDmrSep, resultsPL_df$AuPR,
## method = "spearman"): Cannot compute exact p-value with ties
```

```
## [1] 0.767498
```

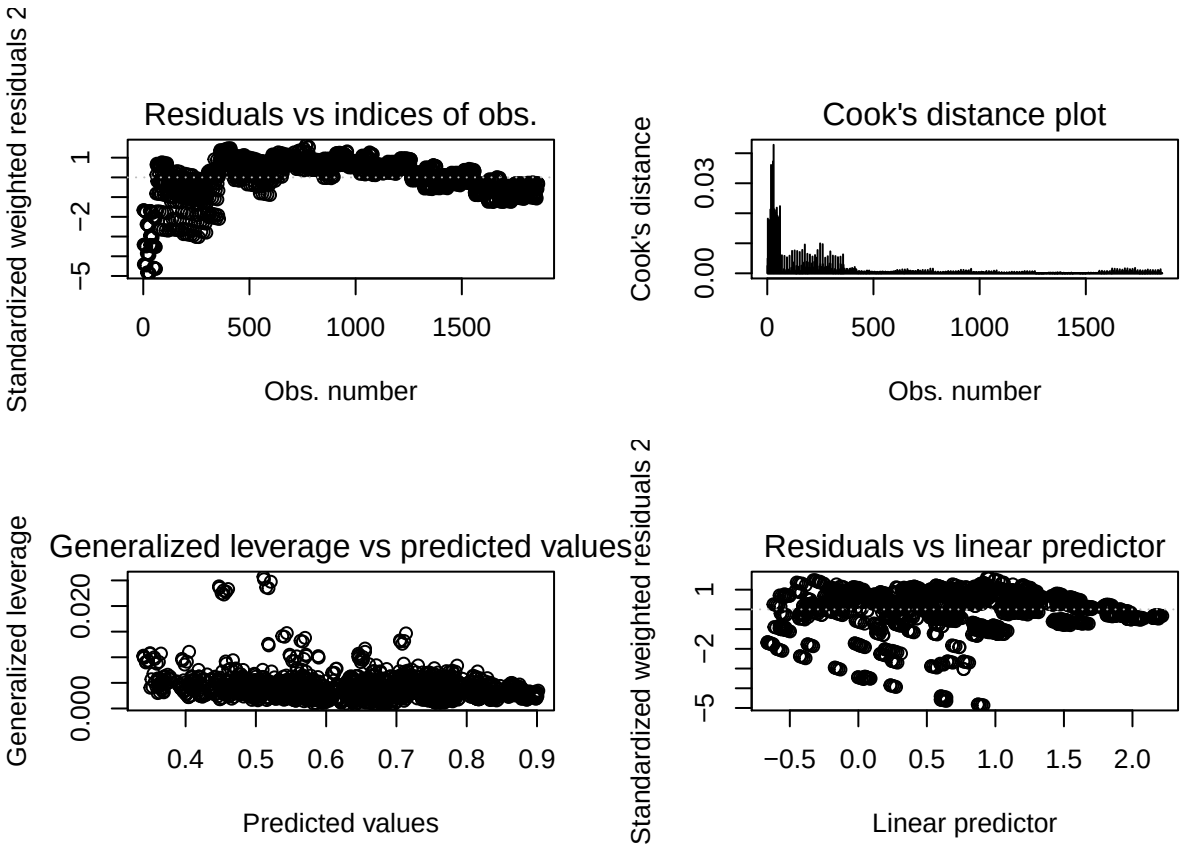
```
###  Statistical Prediction for AuPR  ###
plLin_mod <- lm(AuPR ~ ., data = resultsPL_df)
par(mfrow = c(2, 2))
plot(plLin_mod)
```




```
# Well that's a resounding "NO". However, we see that observation 420 is a
#   potential outlier.
dmrcateRes_df[418:420, ]
```

```
##      method delta seed lambda C   time  FN FP   TN  TP power nPower
## 418 DMRcate   0.1  210    500 4 132.68 186  7 2556 314 0.628    500
## 419 DMRcate   0.1  210    750 4  84.04 206  7 2556 294 0.588    500
## 420 DMRcate   0.1  210   1000 4  83.75 233  9 2554 267 0.534    500
##          AuPR FPprecis TPprecis precision nPrecis       mcc        F1
## 418 0.7280077        7      312 0.9780564     319 0.7535078 0.7637699
## 419 0.6924499        7      289 0.9763514     296 0.7238948 0.7307206
## 420 0.6533225        9      262 0.9667897     271 0.6814422 0.6840731
##     nCPG_q1 nCPG_med nCPG_q3
## 418       6        8      11
## 419       7        9      11
## 420       8        9      12
```

```
# I don't see any problems.

resultsPL_df <-
  resultsPL_df %>%
  mutate(pVal_X_mLasRd = adjPval * mLassoRad)
plBeta_mod <- betareg(AuPR ~ ., data = resultsPL_df)
plot(plBeta_mod)
```




```
summary(plBeta_mod)
```

```
##
## Call:
## betareg(formula = AuPR ~ ., data = resultsPL_df)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -4.8638 -0.4332  0.1979  0.6923  1.6519
##
## Coefficients (mean model with logit link):
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.193e-01  7.438e-02  -8.326   <2e-16 ***
## adjPval        9.645e-01  7.872e-01   1.225    0.220
## mLassoRad      2.308e-03  8.761e-05  26.349   <2e-16 ***
## minDmrSep      5.194e-05  3.710e-05   1.400    0.162
## delta          3.659e+00  1.026e-01  35.655   <2e-16 ***
## nCPG_q3       -6.761e-02  5.616e-03 -12.037   <2e-16 ***
## pVal_X_mLasRd -6.934e-04  1.104e-03  -0.628    0.530
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)  19.2674     0.6196    31.1   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Type of estimator: ML (maximum likelihood)
## Log-likelihood:  1679 on 8 Df
## Pseudo R-squared: 0.6625
## Number of iterations: 13 (BFGS) + 2 (Fisher scoring)
```

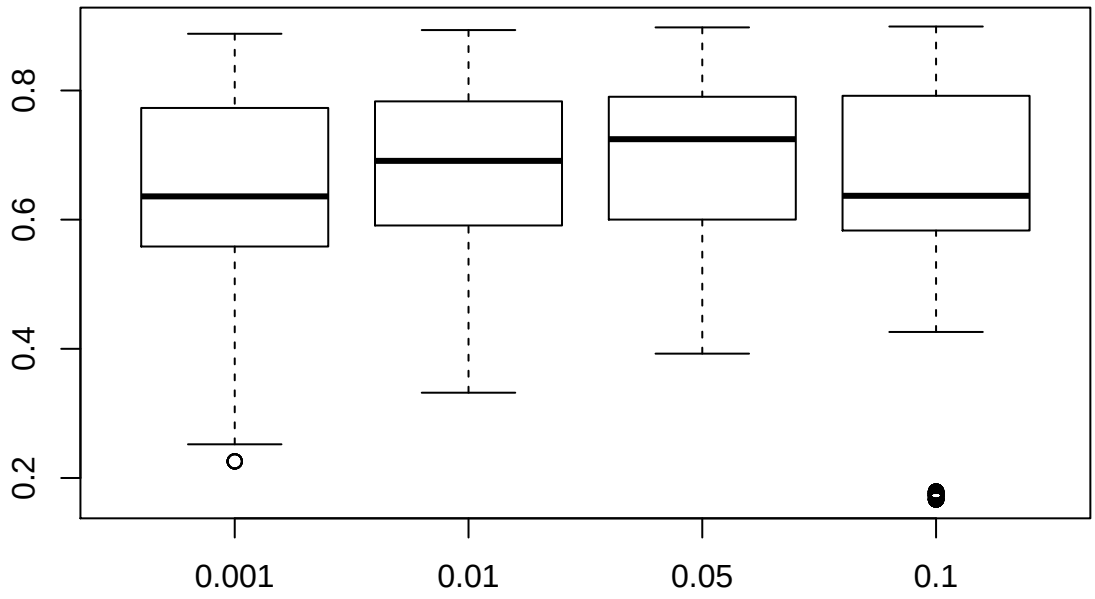
```
coef(plBeta_mod)
```

```
##   (Intercept)       adjPval     mLassoRad     minDmrSep         delta
## -6.193428e-01  9.645190e-01  2.308287e-03  5.193814e-05  3.659159e+00
##       nCPG_q3 pVal_X_mLasRd          (phi)
## -6.760701e-02 -6.934512e-04  1.926744e+01
```

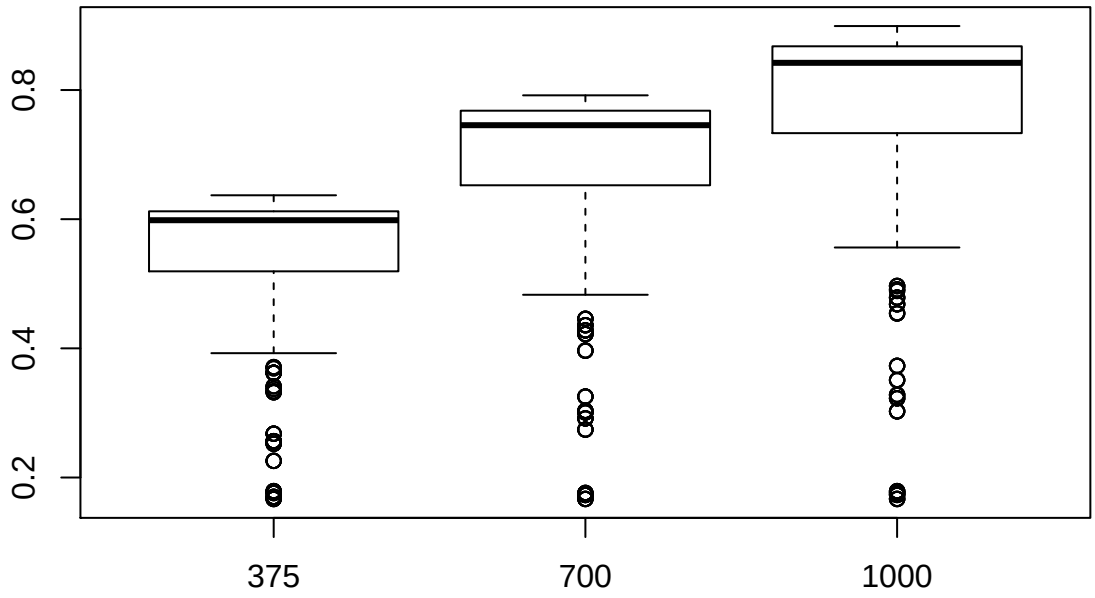
```
# The direction of the adjPval and mLassoRd relationships jive with their
#   Spearman correlations, as does the "not a relationship" result for
#   minDmrSep. What we can then say is this: given delta and the number of
#   CPGs, for each unit increase in adjPval, logit(AuPR) increases by 0.96451899;
#   for each unit increase of mLassoRd, logit(AuPR) increases by 0.00230829
#   (but remember that this effect may be occluded by the nCPG block).

# Let's have a sanity check:
par(mfrow = c(1,1))
unique(resultsPL_df$adjPval)
```

```
## [1] 0.100 0.001 0.010 0.050
```

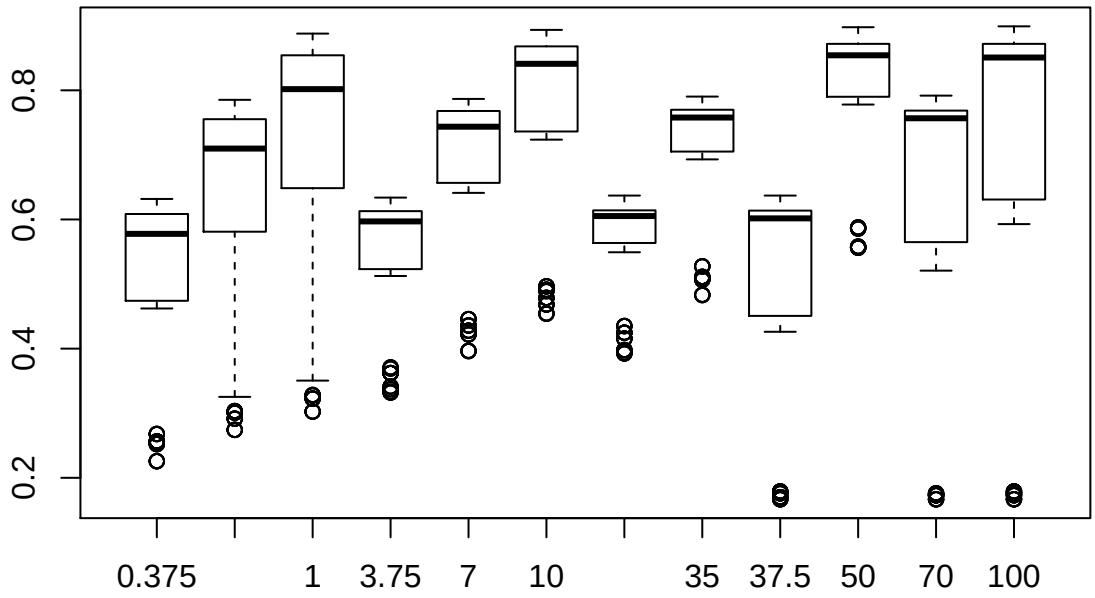
```
boxplot(resultsPL_df$AuPR ~ resultsPL_df$adjPval)
```



```r
# This shows no effect.
boxplot(resultsPL_df$AuPR ~ resultsPL_df$mLassoRad)
```




```r
# This effect is clear: increasing the mean lasso radius increases the AuPR. We
#   take this result with a grain of salt, however: obviously if we test a
#   larger area, we will find more stuff. Interpeting the mean lasso radius
#   fairly (conditional on the number of CPGs), may prove to be quite difficult
boxplot(resultsPL_df$AuPR ~ resultsPL_df$pVal_X_mLasRd)
```




```
# What I see here: the interaction is simply showing how the p-value make the
#   lasso radius effect more pronounced the smaller it is



######  Bumphunter  ##################################################################
data("bumphunterRes_df")
colnames(bumphunterRes_df)
```

```
##  [1] "method"    "delta"     "seed"      "cutoffQ"   "maxGap"
##  [6] "time"      "FN"        "FP"        "TN"        "TP"
## [11] "power"     "nPower"    "AuPR"      "FPprecis"  "TPprecis"
## [16] "precision" "nPrecis"   "mcc"       "F1"        "nCPG_q1"
## [21] "nCPG_med"  "nCPG_q3"
```

```
# Model components:
#   Predictors: cutoffQ and maxGap
#   Responses : power, AuPR, precision, mcc, and F1
#   Covariates: delta, nCPG_med, nCPG_q3
resultsBump_df <-
  bumphunterRes_df %>%
  select(power, AuPR, precision, mcc, F1,
         cutoffQ, maxGap,
         delta, seed, nCPG_med, nCPG_q3) %>%
  mutate(nCPG_med = as.numeric(nCPG_med)) %>%
  mutate(nCPG_q3 = as.numeric(nCPG_q3))
```

```
###  Replace Missing Values with 0  ###
# For each of the responses, 0 is as bad as it gets. Therefore, we can replace
#   all NAs and NaNs with 0. However, precision start at 1 and never reach 0,
#   so we replace all of the 0s for precision with 1.
resultsBump_df    <- resultsBump_df[complete.cases(resultsBump_df), ]

###  Correlation Matrix  ###
# First we build a correlation matrix to inspect possible multicollinearity.
round(cor(resultsBump_df), 2)
```

```
##           power  AuPR precision   mcc    F1 cutoffQ maxGap delta  seed
## power      1.00  0.94      0.09  0.76  0.90   -0.45  -0.04  0.53  0.05
## AuPR       0.94  1.00      0.37  0.90  0.96   -0.21  -0.10  0.69  0.03
## precision  0.09  0.37      1.00  0.70  0.47    0.78  -0.23  0.43  0.00
## mcc        0.76  0.90      0.70  1.00  0.96    0.19  -0.17  0.65  0.04
## F1         0.90  0.96      0.47  0.96  1.00   -0.07  -0.11  0.63  0.04
## cutoffQ   -0.45 -0.21      0.78  0.19 -0.07    1.00   0.00  0.00  0.00
## maxGap    -0.04 -0.10     -0.23 -0.17 -0.11    0.00   1.00  0.00  0.00
## delta      0.53  0.69      0.43  0.65  0.63    0.00   0.00  1.00  0.00
## seed       0.05  0.03      0.00  0.04  0.04    0.00   0.00  0.00  1.00
## nCPG_med   0.04 -0.10     -0.55 -0.32 -0.18   -0.42   0.63 -0.06 -0.03
## nCPG_q3    0.20  0.05     -0.55 -0.22 -0.06   -0.48   0.69  0.06  0.01
##           nCPG_med nCPG_q3
## power         0.04    0.20
## AuPR         -0.10    0.05
## precision    -0.55   -0.55
## mcc          -0.32   -0.22
## F1           -0.18   -0.06
## cutoffQ      -0.42   -0.48
## maxGap        0.63    0.69
## delta        -0.06    0.06
## seed         -0.03    0.01
## nCPG_med      1.00    0.78
## nCPG_q3       0.78    1.00
```

```
resultsBump_df <-
  resultsBump_df %>%
  select(AuPR, cutoffQ, maxGap, delta, nCPG_med)
# We've removed any multicollinearity issues.
cor(resultsBump_df, method = "spearman")
```

```
##                AuPR    cutoffQ     maxGap       delta    nCPG_med
## AuPR      1.0000000 -0.2458081 -0.1758126  0.82989028 -0.12365722
## cutoffQ  -0.2458081  1.0000000  0.0000000  0.00000000 -0.40883238
## maxGap   -0.1758126  0.0000000  1.0000000  0.00000000  0.64852098
## delta     0.8298903  0.0000000  0.0000000  1.00000000 -0.07872703
## nCPG_med -0.1236572 -0.4088324  0.6485210 -0.07872703  1.00000000
```

```
# Now, the correlation test:
cor.test(resultsBump_df$cutoffQ,
         resultsBump_df$AuPR,
         method = "spearman")$p.value
```

```
## Warning in cor.test.default(resultsBump_df$cutoffQ, resultsBump_df$AuPR, :
## Cannot compute exact p-value with ties
```

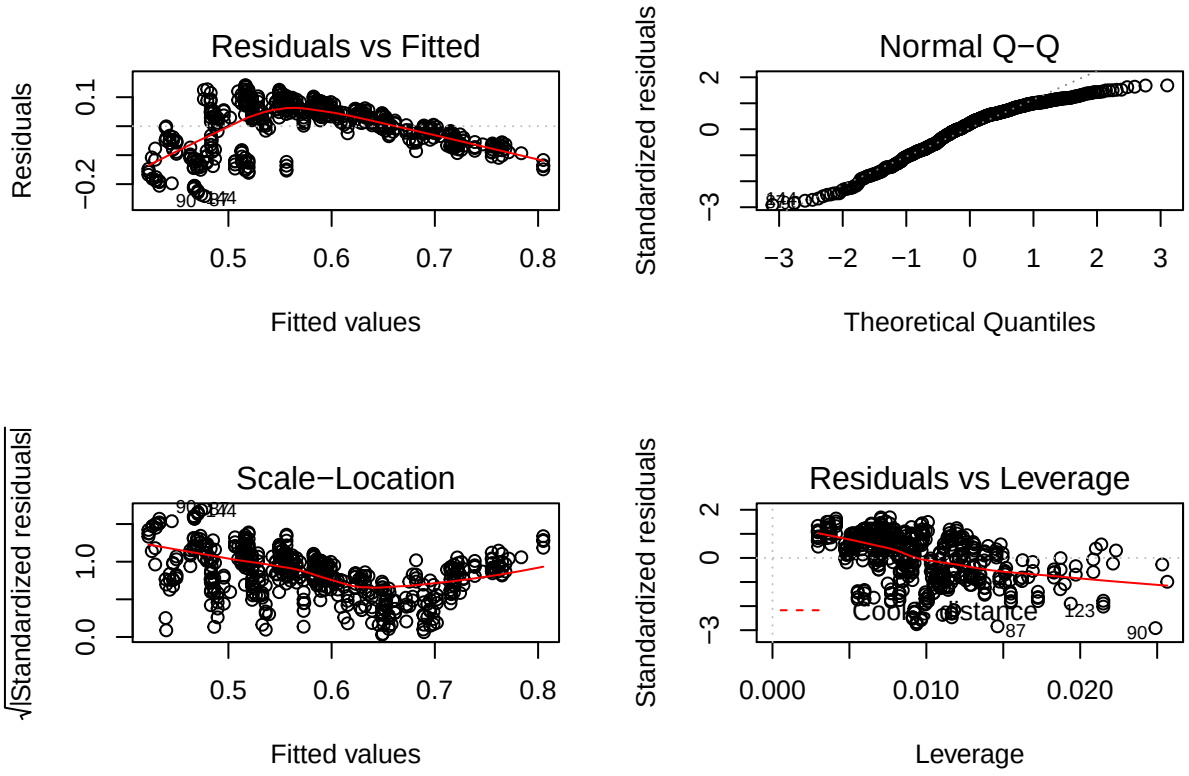
```
## [1] 1.153028e-08
```

```
cor.test(resultsBump_df$maxGap,
         resultsBump_df$AuPR,
         method = "spearman")$p.value
```

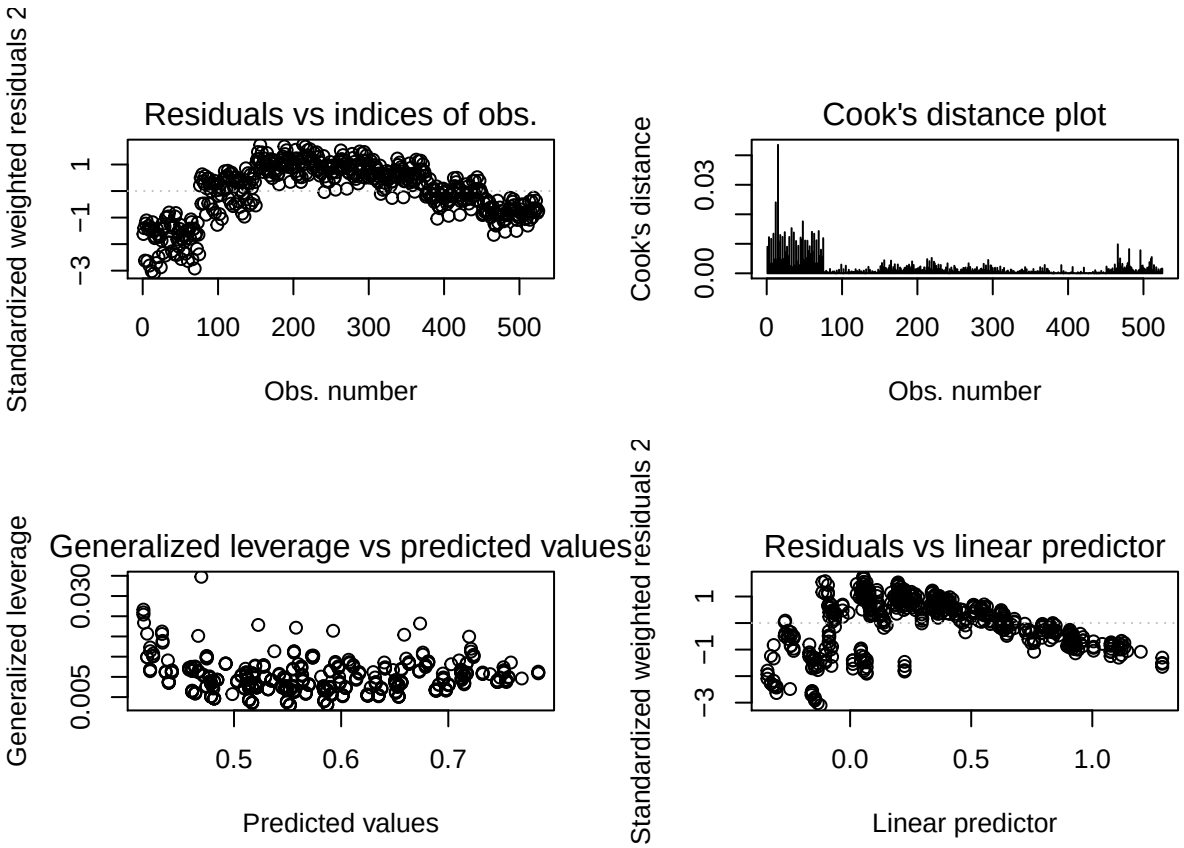
```
## Warning in cor.test.default(resultsBump_df$maxGap, resultsBump_df$AuPR, :
## Cannot compute exact p-value with ties
```

```
## [1] 5.113853e-05
```

```
# Both are related


###  Statistical Prediction for AuPR  ###
bumpLin_mod <- lm(AuPR ~ ., data = resultsBump_df)
par(mfrow = c(2, 2))
plot(bumpLin_mod)
```




```
# Well that's a resounding "NO".

bumpBeta_mod <- betareg(AuPR ~ ., data = resultsBump_df)
plot(bumpBeta_mod)
```




```
summary(bumpBeta_mod)
```

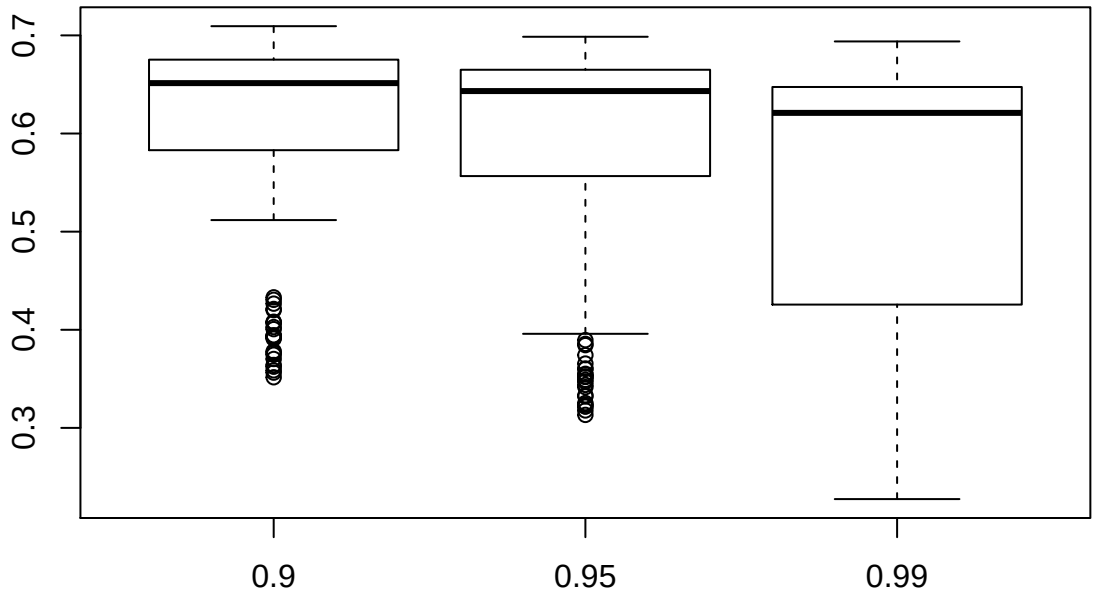
```
##
## Call:
## betareg(formula = AuPR ~ ., data = resultsBump_df)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -3.0998 -0.7260  0.1912  0.7663  1.7431
##
## Coefficients (mean model with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.079e+00  6.225e-01   8.158 3.39e-16 ***
## cutoffQ     -4.269e+00  4.858e-01  -8.787  < 2e-16 ***
## maxGap       4.637e-05  6.844e-05   0.678    0.498
## delta        2.837e+00  1.230e-01  23.072  < 2e-16 ***
## nCPG_med    -1.820e-01  3.878e-02  -4.693 2.69e-06 ***
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)   35.024      2.133   16.42   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 576.9 on 6 Df
```

```
## Pseudo R-squared: 0.5469
## Number of iterations: 12 (BFGS) + 1 (Fisher scoring)
```

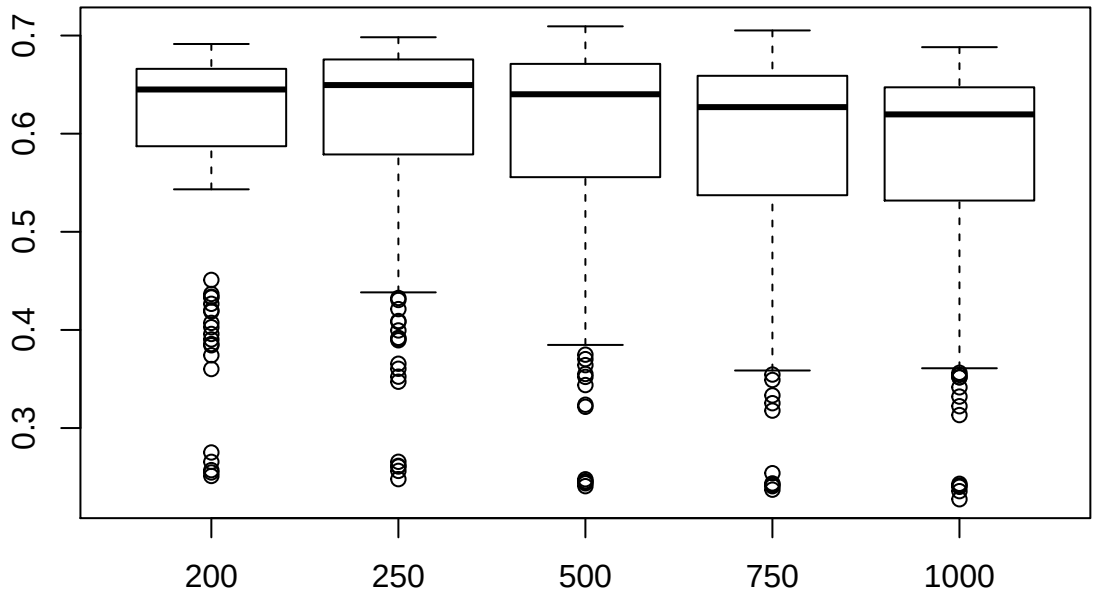
```
coef(bumpBeta_mod)
```

```
##   (Intercept)       cutoffQ        maxGap         delta      nCPG_med
##  5.078971e+00 -4.268951e+00  4.637477e-05  2.837170e+00 -1.820260e-01
##         (phi)
##  3.502381e+01
```

```
# The cutoffQ parameter results agree with the Spearman correlation. The maxGap
#   parameter is not significant in the model, but the sign changes.

# Let's have a sanity check:
par(mfrow = c(1,1))
boxplot(resultsBump_df$AuPR ~ resultsBump_df$cutoffQ)
```




```
# The smaller quantiles yield better performance
boxplot(resultsBump_df$AuPR ~ resultsBump_df$maxGap)
```




```
# There is no effect.
```

```
######  Comb-p  ###########################################################
data("combpRes_df")
colnames(combpRes_df)
```

```
##  [1] "method"    "delta"     "seed"      "combSeed"  "combDist"
##  [6] "time"      "FN"        "FP"        "TN"        "TP"
## [11] "power"     "nPower"    "AuPR"      "FPprecis"  "TPprecis"
## [16] "precision" "nPrecis"   "mcc"       "F1"        "nCPG_q1"
## [21] "nCPG_med"  "nCPG_q3"
```

```
# Model components:
#   Predictors: combSeed and combDist
#   Responses : power, AuPR, precision, mcc, and F1
#   Covariates: delta, nCPG_med, nCPG_q3
resultsComb_df <-
  combpRes_df %>%
  select(power, AuPR, precision, mcc, F1,
         combSeed, combDist,
         delta, seed, nCPG_med, nCPG_q3) %>%
  mutate(nCPG_med = as.numeric(nCPG_med)) %>%
  mutate(nCPG_q3 = as.numeric(nCPG_q3))
```

```
## Warning in eval(substitute(expr), envir, enclos): NAs introduced by
```

```
## coercion
```

```
###  Replace Missing Values with 0  ###
# For each of the responses, 0 is as bad as it gets. Therefore, we can replace
#   all NAs and NaNs with 0. However, precision start at 1 and never reach 0,
#   so we replace all of the 0s for precision with 1.
resultsComb_df    <- resultsComb_df[complete.cases(resultsComb_df), ]


###  Correlation Matrix  ###
# First we build a correlation matrix to inspect possible multicollinearity. We
#   will keep nCPG_q3 because nCPG_med has near-constant variance. Half of the
#   values are 7, and another third of the values are 8.
round(cor(resultsComb_df), 2)
```

```
##           power  AuPR precision   mcc    F1 combSeed combDist delta  seed
## power      1.00  1.00     -0.13  0.99  0.99     0.06     0.30  0.68 -0.02
## AuPR       1.00  1.00     -0.07  1.00  1.00     0.06     0.24  0.68 -0.03
## precision -0.13 -0.07      1.00 -0.05 -0.05    -0.14    -0.86  0.05 -0.18
## mcc        0.99  1.00     -0.05  1.00  1.00     0.06     0.22  0.68 -0.04
## F1         0.99  1.00     -0.05  1.00  1.00     0.06     0.21  0.66 -0.03
## combSeed   0.06  0.06     -0.14  0.06  0.06     1.00     0.00  0.00  0.00
## combDist   0.30  0.24     -0.86  0.22  0.21     0.00     1.00  0.00  0.00
## delta      0.68  0.68      0.05  0.68  0.66     0.00     0.00  1.00  0.00
## seed      -0.02 -0.03     -0.18 -0.04 -0.03     0.00     0.00  0.00  1.00
## nCPG_med   0.09  0.04     -0.77  0.02  0.02    -0.07     0.85 -0.06  0.01
## nCPG_q3    0.21  0.15     -0.82  0.13  0.12    -0.06     0.93  0.02 -0.05
##           nCPG_med nCPG_q3
## power         0.09    0.21
## AuPR          0.04    0.15
## precision    -0.77   -0.82
## mcc           0.02    0.13
## F1            0.02    0.12
## combSeed     -0.07   -0.06
## combDist      0.85    0.93
## delta        -0.06    0.02
## seed          0.01   -0.05
## nCPG_med      1.00    0.86
## nCPG_q3       0.86    1.00
```

```
resultsComb_df <-
  resultsComb_df %>%
  select(AuPR, combSeed, combDist, delta, nCPG_med)
# We've removed any multicollinearity issues, except for with nCPGs
cor(resultsComb_df, method = "spearman")
```

```
##                AuPR      combSeed      combDist       delta    nCPG_med
## AuPR     1.00000000  0.0450990234  0.4510015263  0.79234282  0.29286572
## combSeed 0.04509902  1.0000000000 -0.0009060959  0.00000000 -0.05343249
## combDist 0.45100153 -0.0009060959  1.0000000000  0.00000000  0.85764193
## delta    0.79234282  0.0000000000  0.0000000000  1.00000000 -0.05883045
## nCPG_med 0.29286572 -0.0534324857  0.8576419315 -0.05883045  1.00000000
```

```
# Now, the correlation test:
cor.test(resultsComb_df$combSeed,
         resultsComb_df$AuPR,
```

```
        method = "spearman")$p.value # not related
```

```
## Warning in cor.test.default(resultsComb_df$combSeed, resultsComb_df$AuPR, :
## Cannot compute exact p-value with ties
```

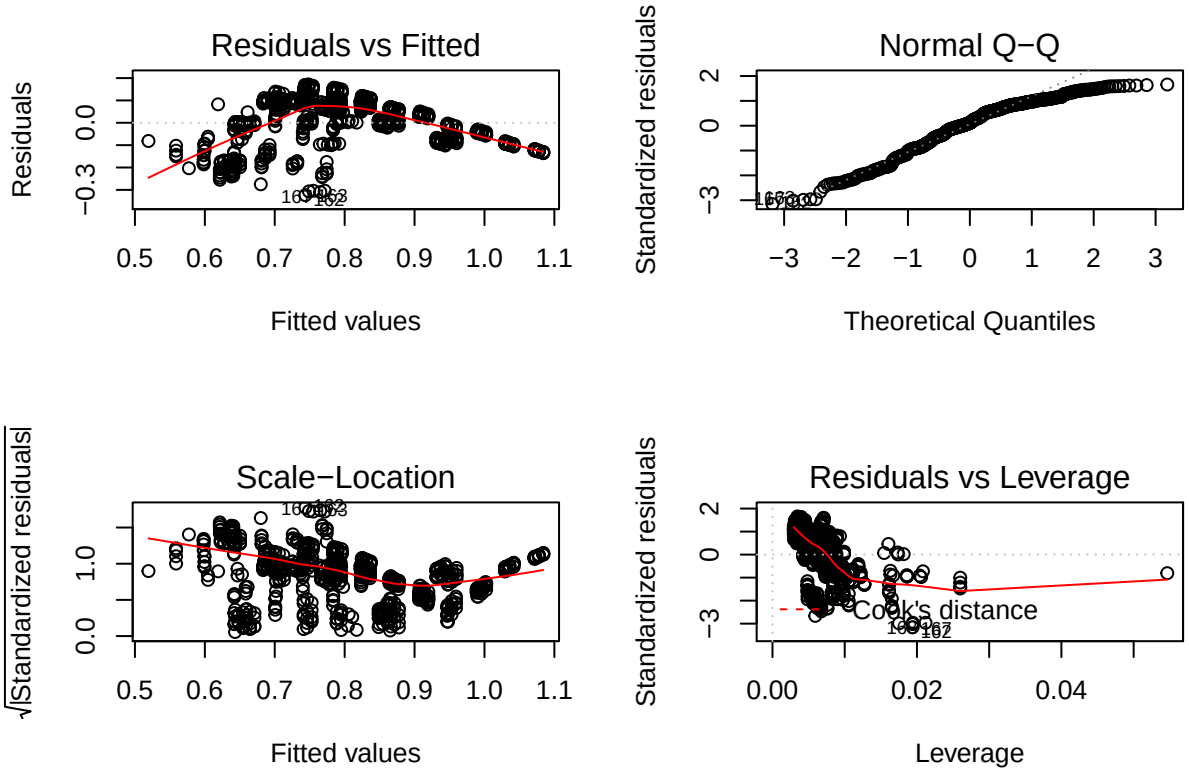
```
## [1] 0.2337214
```

```
cor.test(resultsComb_df$combDist,
         resultsComb_df$AuPR,
         method = "spearman")$p.value # related
```

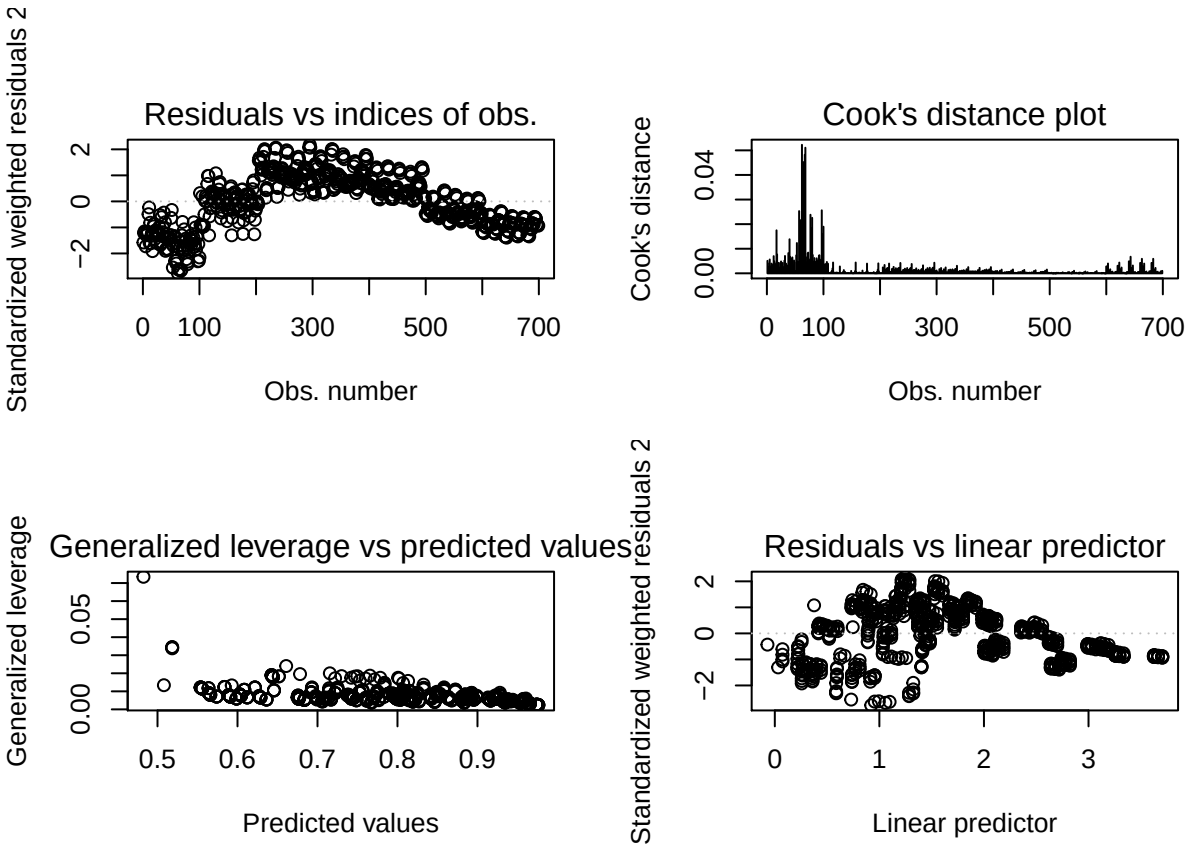
```
## Warning in cor.test.default(resultsComb_df$combDist, resultsComb_df$AuPR, :
## Cannot compute exact p-value with ties
```

```
## [1] 2.543217e-36
```

```
###  Statistical Prediction for AuPR  ###
combLin_mod <- lm(AuPR ~ ., data = resultsComb_df)
par(mfrow = c(2, 2))
plot(combLin_mod)
```




```
# Well that's a resounding "NO".

combBeta_mod <- betareg(AuPR ~ ., data = resultsComb_df)
plot(combBeta_mod)
```




```
summary(combBeta_mod)
```

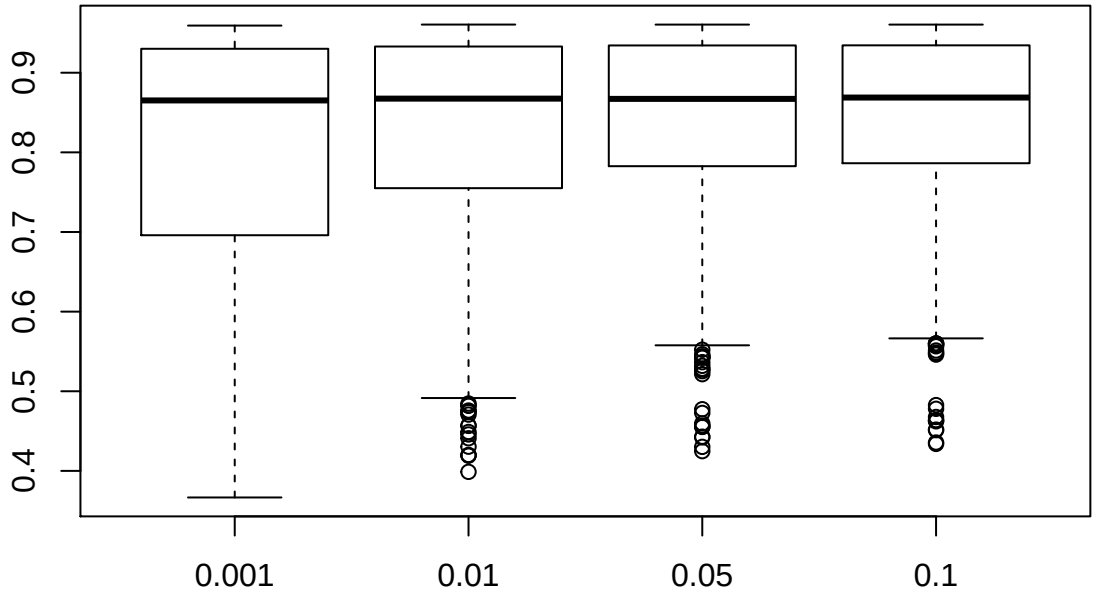
```
##
## Call:
## betareg(formula = AuPR ~ ., data = resultsComb_df)
##
## Standardized weighted residuals 2:
##     Min      1Q  Median      3Q     Max
## -2.7634 -0.8209  0.0974  0.7462  2.1742
##
## Coefficients (mean model with logit link):
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.2860768  0.4024680  10.649   <2e-16 ***
## combSeed     0.8120693  0.5122967   1.585    0.113
## combDist     0.0020535  0.0001263  16.265   <2e-16 ***
## delta        6.3503683  0.1905001  33.335   <2e-16 ***
## nCPG_med    -0.6569414  0.0603140 -10.892   <2e-16 ***
##
## Phi coefficients (precision model with identity link):
##       Estimate Std. Error z value Pr(>|z|)
## (phi)   23.377      1.258   18.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Type of estimator: ML (maximum likelihood)
## Log-likelihood: 870.5 on 6 Df
```

```
## Pseudo R-squared: 0.6746
## Number of iterations: 14 (BFGS) + 5 (Fisher scoring)
```

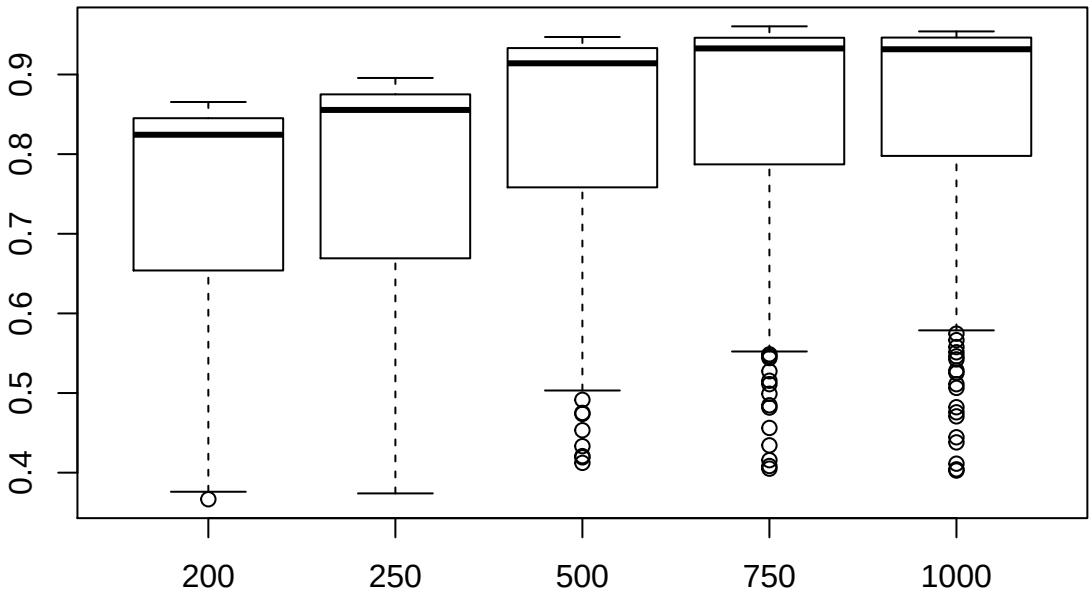
```
coef(combBeta_mod)
```

```
##  (Intercept)     combSeed     combDist        delta     nCPG_med
##  4.286076803  0.812069280  0.002053485  6.350368261 -0.656941392
##         (phi)
## 23.377096607
```

```
# The directions and significances of the parameters agree with the Spearman
#   correlations.

# Let's have a sanity check:
par(mfrow = c(1,1))
boxplot(resultsComb_df$AuPR ~ resultsComb_df$combSeed)
```




```
# There is no effect.
boxplot(resultsComb_df$AuPR ~ resultsComb_df$combDist)
```




```
# Increasing combDist yields better performance.
```