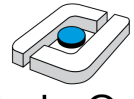
cutpointr

-

Improved and tidy estimation of optimal cutpoints

Christian Thiele & Gerrit Hirschfeld

06 July 2017

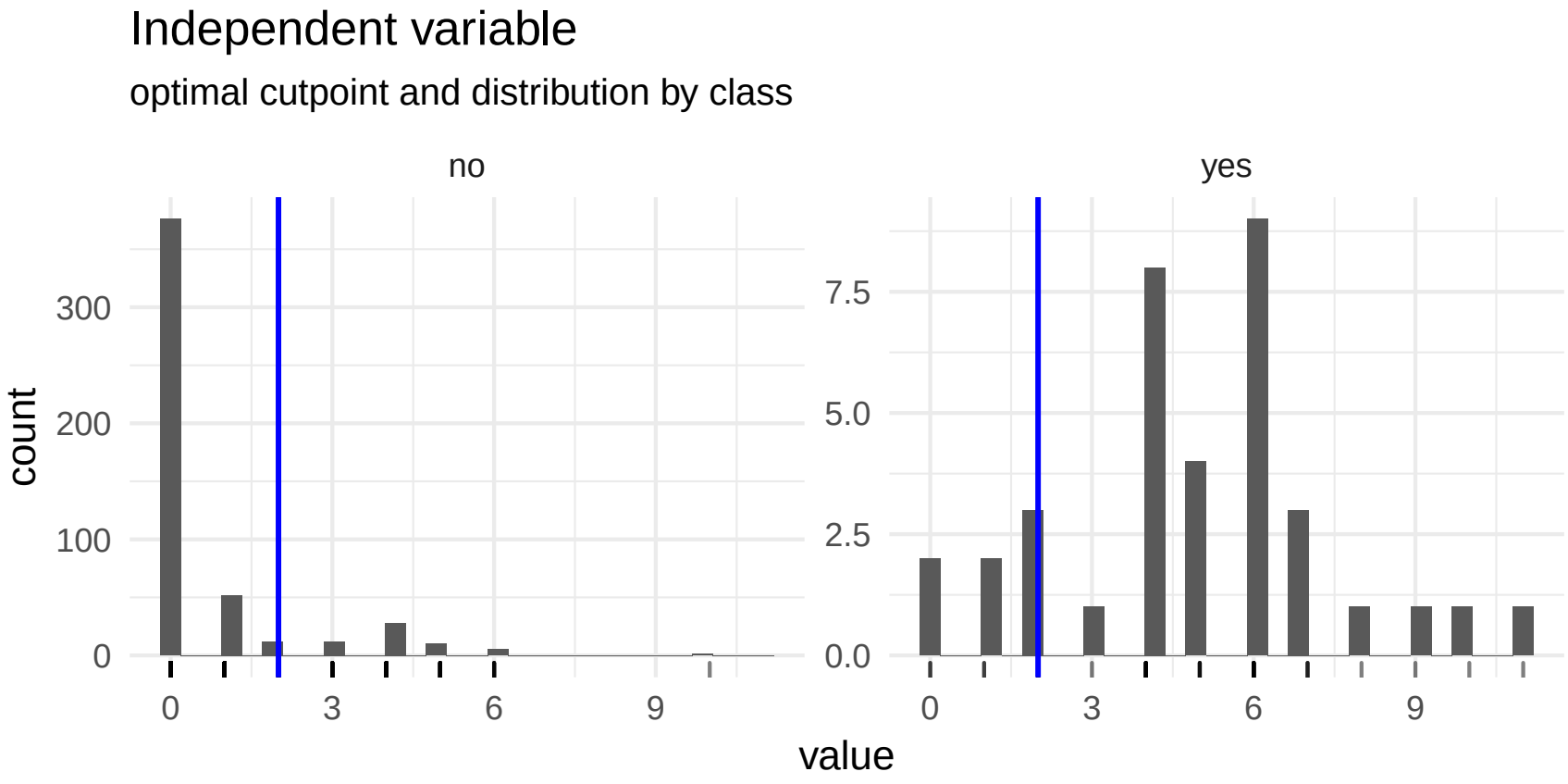


Hochschule Osnabrück
University of Applied Sciences

What do we want 'optimal' cutpoints for?

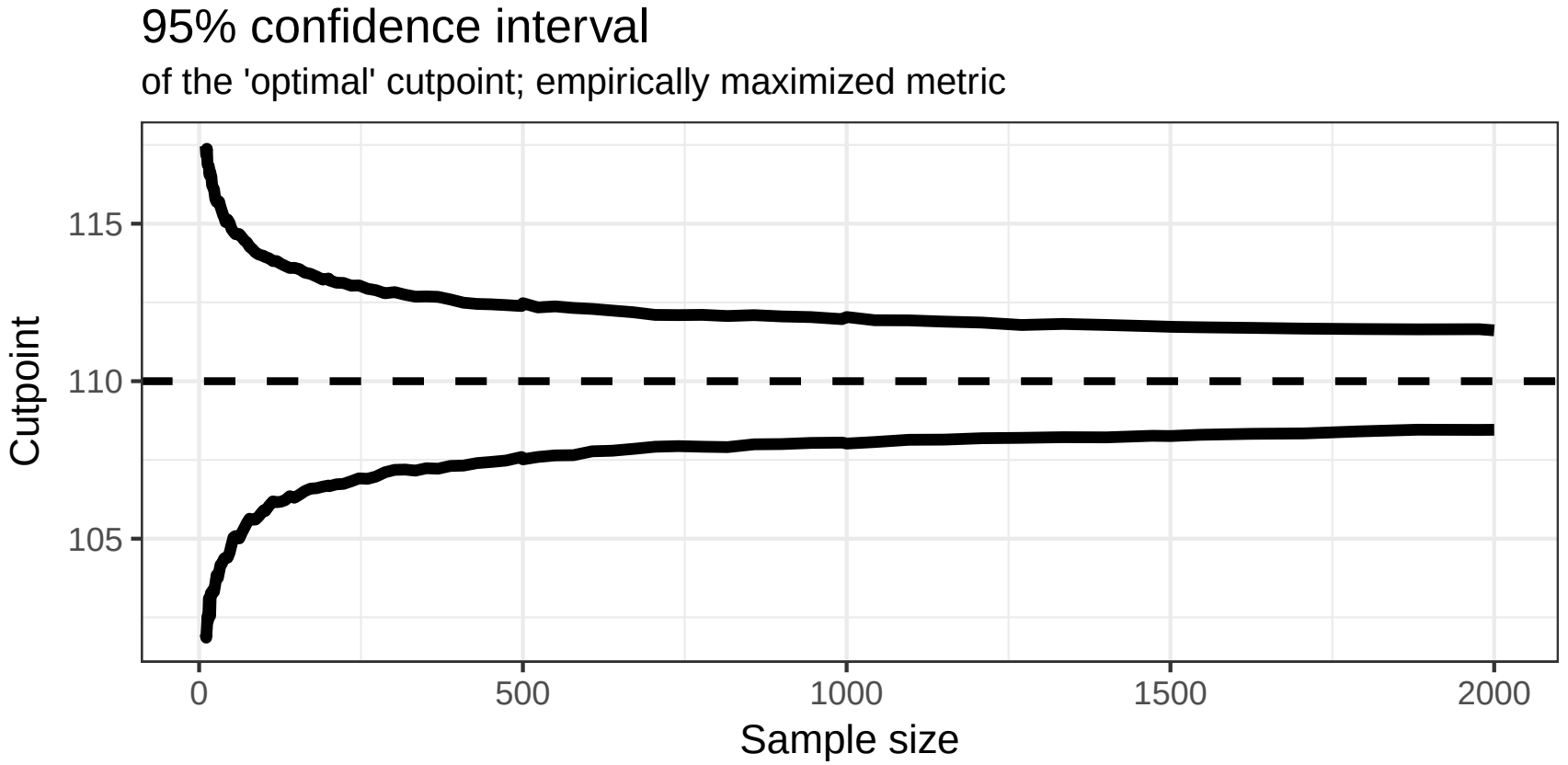
Binary classification via:

- Biological markers
- Psychological scores
- Model predictions



Problems with ‘optimal’ cutpoints

- Prone to overfitting
- Selecting the ‘optimal’ cutpoint by trying out all possible ones leads to
  - overestimation of accuracy
  - highly variable cutpoints

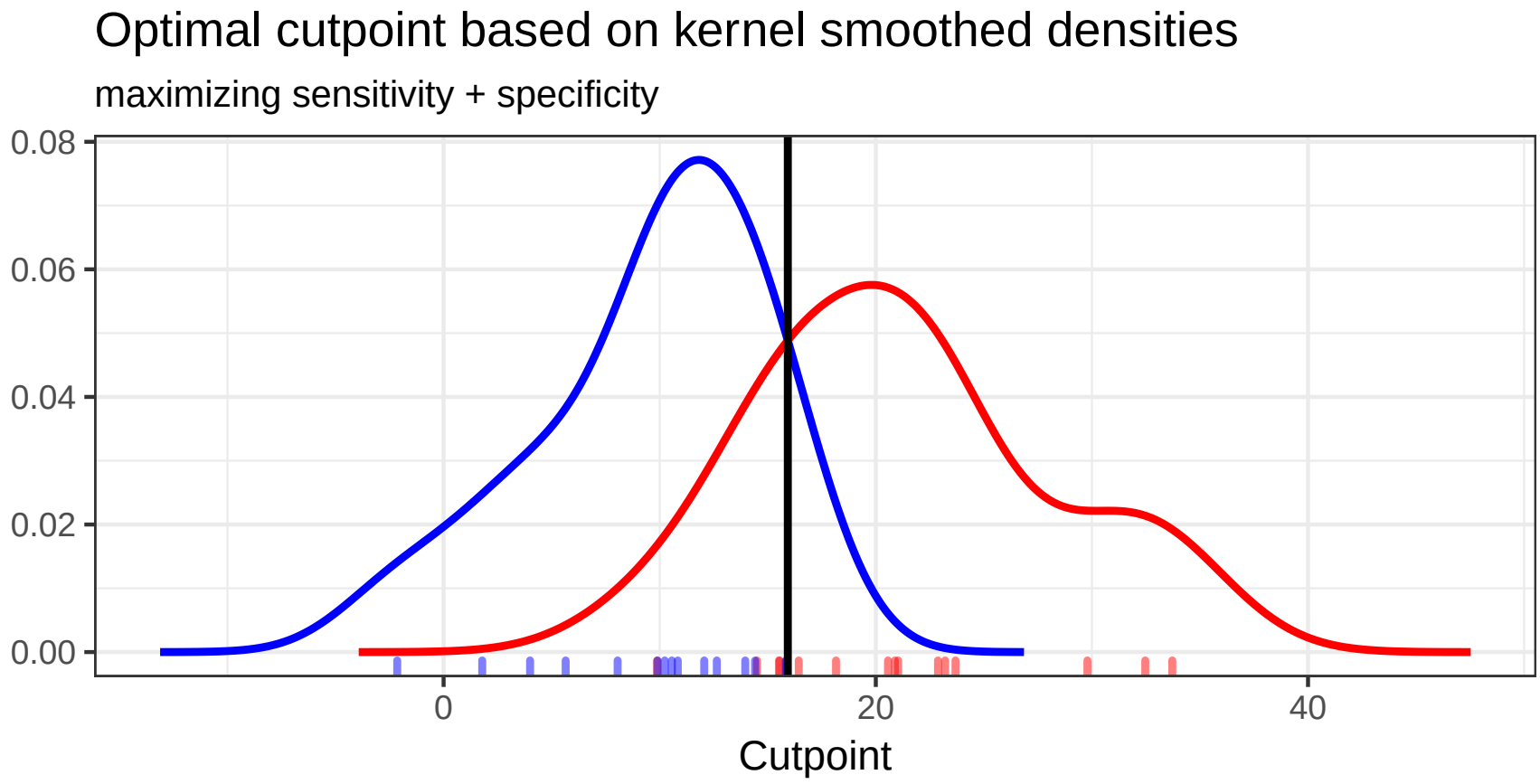


Some features of cutpointr

- More robust methods for lower variability of 'optimal' cutpoints
- Included bootstrapping (parallelizable)
- Extensibility by user-defined functions
- Tidy interface and output

Kernel method

- lower variability for maximizing sensitivity + specificity



Tidy interface and output

```
suicide %>%
    cutpointr(x = dsi, class = suicide,
              subgroup = gender,
              method = maximize_metric,
              metric = accuracy,
              direction = ">=",
              pos_class = "yes",  neg_class = "no",
              boot_runs = 200)
```

Tidy interface and output

```
suicide %>%
    cutpointr(x = dsi, class = suicide,
              subgroup = gender,
              method = maximize_metric,
              metric = accuracy,
              direction = ">=",
              pos_class = "yes",  neg_class = "no",
              boot_runs = 200)
```

Tidy interface and output

The returned object is also a normal tibble

Automatic 'guessing' of the positive / negative class and whether higher or lower predictor values imply the positive class

```
> suicide %>% cutpointr(dsi, suicide, gender, boot_runs = 200)
Assuming yes as the positive class
Assuming the positive class has higher x values
# A tibble: 2 × 18
  subgroup direction optimal_cutpoint          method Sum_Sens_Spec
     <chr>     <chr>            <dbl>           <chr>         <dbl>
1   female        >=                2 maximize_metric      1.808118
2     male        >=                3 maximize_metric      1.625106
   accuracy sensitivity specificity       AUC pos_class neg_class
      <dbl>       <dbl>       <dbl>     <dbl>    <fctr>    <fctr>
1 0.8852041   0.9259259   0.8821918 0.9446474       yes        no
2 0.8428571   0.7777778   0.8473282 0.8617472       yes        no
  prevalence outcome predictor grouping              data
       <dbl>   <chr>     <chr>    <chr>            <list>
1 0.06887755 suicide       dsi   gender <tibble [392 × 2]>
2 0.06428571 suicide       dsi   gender <tibble [140 × 2]>
           roc_curve                boot
              <list>              <list>
1 <tibble [11 × 10]> <tibble [200 × 18]>
2 <tibble [11 × 10]> <tibble [200 × 18]>
```

Data per group as nested tibbles

ROC curve and bootstrap results as nested tibbles

Tidy interface and output

The returned object is also a normal tibble

Automatic 'guessing' of the positive / negative class and whether higher or lower predictor values imply the positive class

```
> suicide %>% cutpointr(dsi, suicide, gender, boot_runs = 200)
Assuming yes as the positive class
Assuming the positive class has higher x values
# A tibble: 2 × 18
  subgroup direction optimal_cutpoint          method Sum_Sens_Spec
     <chr>     <chr>            <dbl>           <chr>         <dbl>
1   female        >=                2 maximize_metric      1.808118
2     male        >=                3 maximize_metric      1.625106
   accuracy sensitivity specificity       AUC pos_class neg_class
      <dbl>       <dbl>       <dbl>     <dbl>    <fctr>    <fctr>
1 0.8852041   0.9259259   0.8821918 0.9446474       yes        no
2 0.8428571   0.7777778   0.8473282 0.8617472       yes        no
  prevalence outcome predictor grouping              data
       <dbl>   <chr>     <chr>    <chr>            <list>
1 0.06887755 suicide       dsi   gender <tibble [392 × 2]>
2 0.06428571 suicide       dsi   gender <tibble [140 × 2]>
           roc_curve                boot
              <list>              <list>
1 <tibble [11 × 10]> <tibble [200 × 18]>
2 <tibble [11 × 10]> <tibble [200 × 18]>
```

Data per group as nested tibbles

ROC curve and bootstrap results as nested tibbles

Summary

```
summary(cp)
```

```
 optimal_cutpoint Sum_Sens_Spec accuracy sensitivity specificity    AUC n_pos n_neg
                2        1.7518   0.8647      0.8889      0.8629 0.9238    36   496

          observation
prediction yes  no
       yes  32  68
       no    4 428

Predictor summary:
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.         SD
 0.0000000  0.0000000  0.0000000  0.9210526  1.0000000 11.0000000  1.8527143

Predictor summary per class:
    Min. 1st Qu. Median      Mean 3rd Qu. Max       SD
no     0       0      0 0.6330645       0  10 1.412225
yes    0       4      5 4.8888889       6  11 2.549821

Bootstrap summary:
                   Min. 1st Qu. Median   Mean 3rd Qu.    Max     SD
optimal_cutpoint 1.0000  2.0000 2.0000 2.1950  2.0000 4.0000 0.8187
Sum_Sens_Spec    1.3939  1.6442 1.7240 1.7072  1.7729 1.8778 0.0941
Accuracy_b       0.7462  0.8534 0.8703 0.8623  0.8835 0.9267 0.0375
Accuracy_oob     0.7143  0.8462 0.8639 0.8538  0.8758 0.9196 0.0417
Sensitivity_b    0.7419  0.8680 0.8974 0.8985  0.9378 1.0000 0.0522
Sensitivity_oob  0.5000  0.7857 0.8750 0.8531  0.9286 1.0000 0.1091
Specificity_b    0.7321  0.8516 0.8663 0.8596  0.8824 0.9306 0.0415
Specificity_oob  0.6979  0.8438 0.8620 0.8541  0.8780 0.9412 0.0484
Kappa_b          0.2012  0.3679 0.4155 0.4127  0.4645 0.6042 0.0737
Kappa_oob        0.1775  0.3413 0.3950 0.3878  0.4440 0.5455 0.0818
```

User defined metric functions

The arguments to `method` and `metric` are actual functions

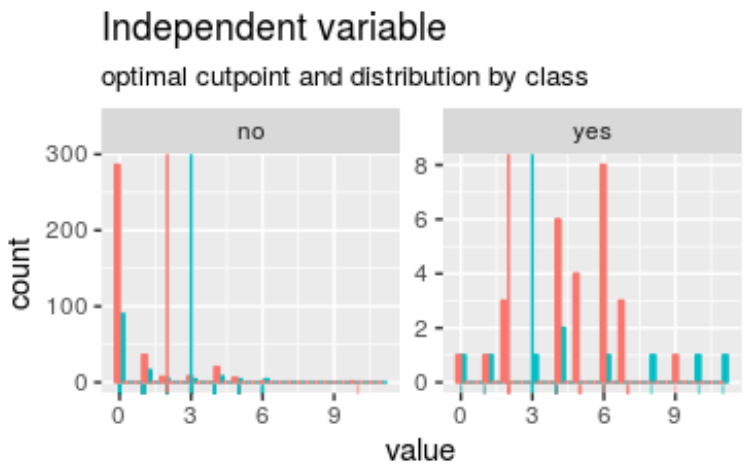
- `metric` is passed to `method`

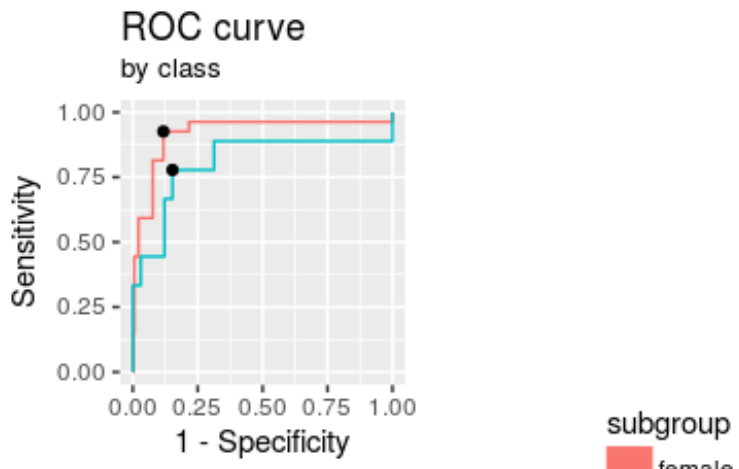
```
accuracy
```

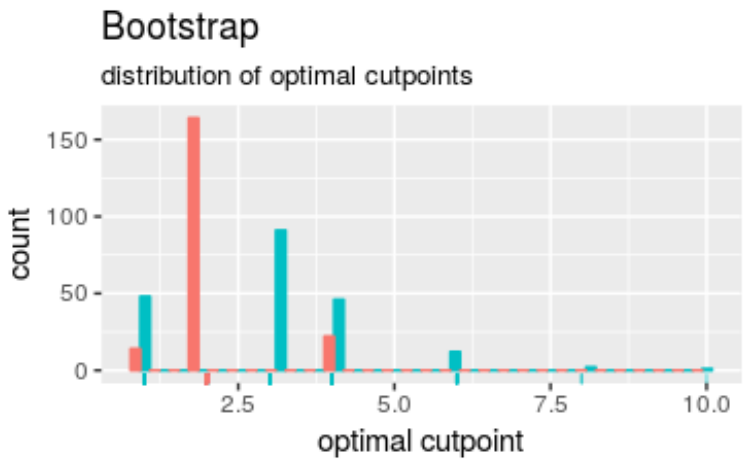
```
## function(tp, fp, tn, fn, ...) {
##     Accuracy <- cbind((tp + tn) / (tp + fp + tn + fn))
##     colnames(Accuracy) <- "Accuracy"
##     return(Accuracy)
## }
## <environment: namespace:cutpointr>
```

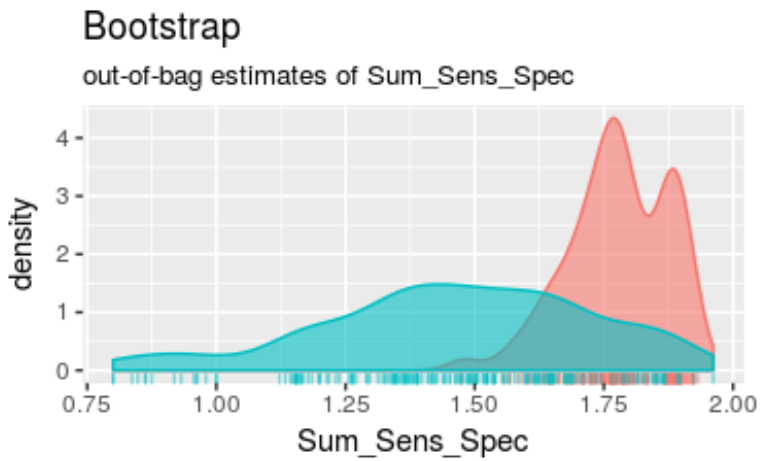
Plots

```
cp <- cutpointr(suicide, dsi, suicide, gender,
                boot_runs = 200,
                direction = ">=", pos_class = "yes")
plot(cp)
```



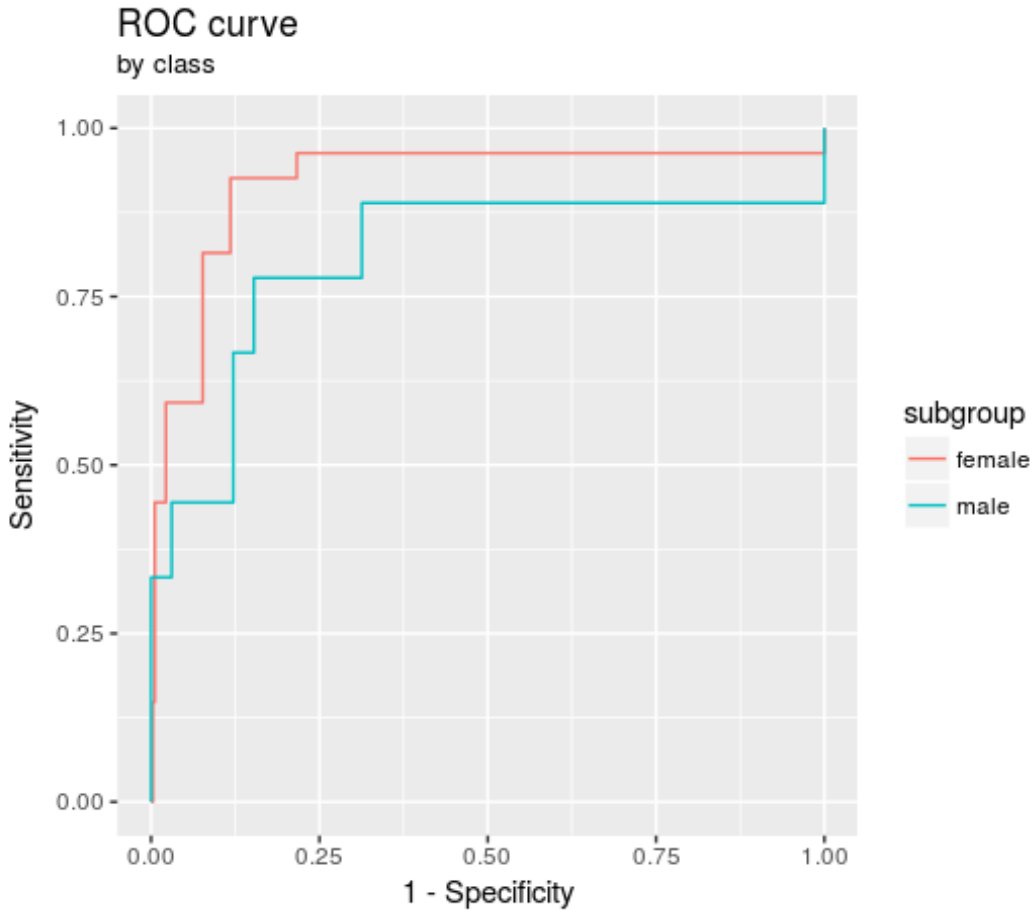




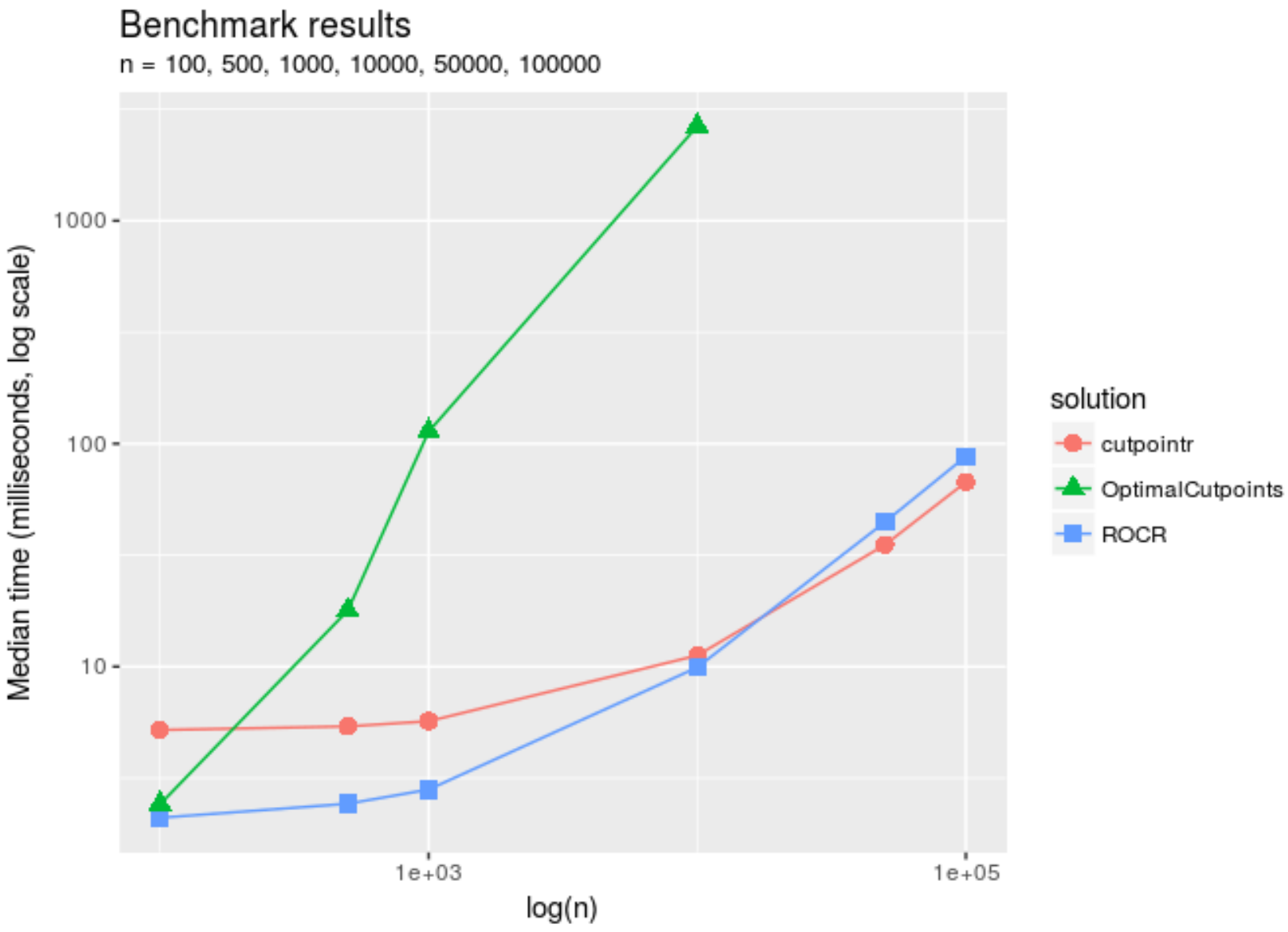


Single plots

```
suicide %>%
    cutpointr(dsi, suicide, gender) %>%
    plot_roc(display_cutpoint = FALSE)
```



Benchmarks


Benchmark results
n = 100, 500, 1000, 10000, 50000, 100000
Median time (milliseconds, log scale)
1000
100
10
1e+03
1e+05
log(n)
solution
cutpointr
OptimalCutpoints
ROCR

Thank you

Not yet on CRAN but on Github:
`https://github.com/Thie1e/cutpointr`

Funding: BMBF Indimed