Błażej Kochański
Gdańsk University of Technology
blazejkochanski.pl

Computing ROC AUC Efficiently with R

Many AUC implementations in R

Table 1: Functions computing AUC.

Package	Function	Usage	Method	Language
bigstatsr	AUC	`AUC(pred, target)`	Pairwise comparison (optimized)	C++, R
caTools	colAUC	`colAUC(pred, target)[1, 1]`	Rank sum	R
cvAUC	AUC	`AUC(pred, target)`	Trapezoidal rule	R
DescTools	Cstat	`Cstat(pred, target)`	Pairwise comparison (optimized)	C++, R
effsize	VD.A	`VD.A(-pred ~ factor(target))$estimate`	Rank sum	R
fbroc	boot.roc	`boot.roc(pred, as.logical(target))$auc`	Trapezoidal rule	C++, R
Hmisc	somers2	`somers2(pred, target)["C"]`	Rank sum	R
Metrics	auc	`auc(target, pred)`	Rank sum	R
MLmetrics	AUC	`AUC(pred, target)`	Rank sum	R
mltools	auc_roc	`auc_roc(pred, target)`	Trapezoidal rule	R
ModelMetrics	auc	`auc(target, pred)`	Rank sum	C++, R
precrec	evalmod	`evalmod(mmdata(scores = pred, labels = target), mode="aucroc")$uaucs$aucs`	Rank sum	C++, R
pROC	auc	`auc(target, pred, lev=c('0', '1'), dir="<")`	Trapezoidal rule	R
PRROC	roc.curve	`roc.curve(pred[target == 1], pred[target == 0])$auc`	Trapezoidal rule	R
rcompanion	vda	`vda(x = pred[target == 1], y = pred[target == 0], digits=100)`	Rank sum	R
ROCit	rocit	`rocit(pred, target)$AUC`	Trapezoidal rule	R
ROCR	performance	`performance(prediction(pred, target), "auc")@y.values[[1]]`	Trapezoidal rule	R
scikit-learn	roc_auc_score	`import("sklearn.metrics")$roc_auc_score(target, pred)`	Trapezoidal rule	Python, C
scorecard	perf_eva	`perf_eva(pred, target, binomial_metric = "auc", show_plot=FALSE)$binomial_metric$dat$AUC`	Trapezoidal rule	R
yardstick	roc_auc_vec	`roc_auc_vec(as.factor(target), -pred)`	Trapezoidal rule	R
DescTools	SomersDelta	`SomersDelta(pred, target)/2 + 1/2`	Pairwise comparison (unoptimized)	C++, R

How to compute AUC in R?

There are several common meanings of **AUC (Area Under the Curve)** in R. The most common is the **area under the ROC curve** used to evaluate binary classifiers. Another is the area under a numerical curve (e.g., concentration–time profiles). Here are both.

1. AUC for an ROC curve (binary classification)

Using the `pROC` package

R

```r
install.packages("pROC")
library(pROC)

# True class (0/1 or factor)
y <- c(0, 0, 1, 1, 1, 0, 1, 0)

# Predicted probabilities
pred <- c(0.1, 0.4, 0.35, 0.8, 0.9, 0.2, 0.7, 0.3)

roc_obj <- roc(y, pred)
auc(roc_obj)
```

AUC implementations differ dramatically in speed

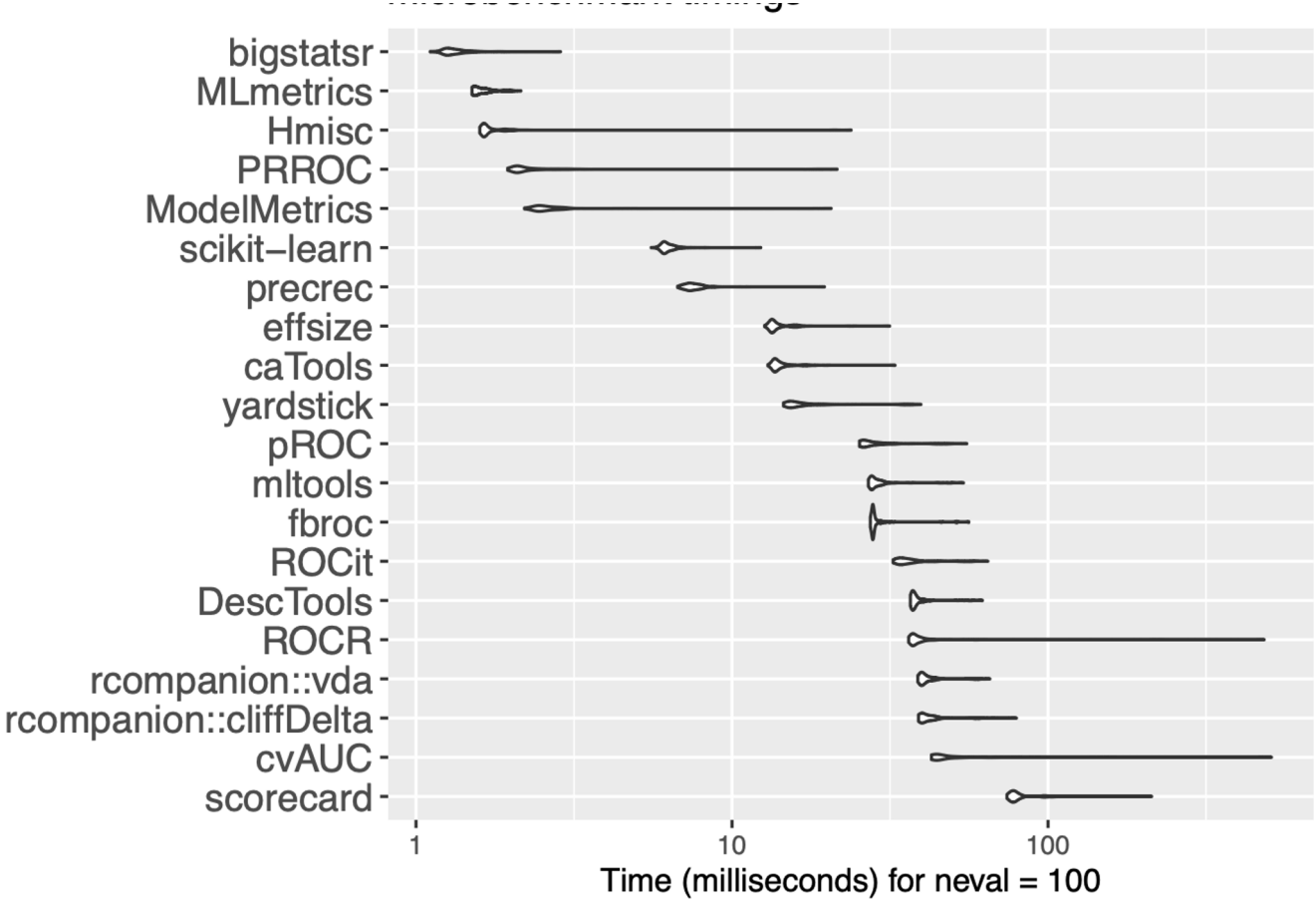


Figure 3: Benchmark of the computational efficiency of R packages for AUC computation using 10,000 observations.

If you need to compute AUC repeatedly,

choose efficient algorithms or packages that implement them.

[ROC] AUC is a performance measure of a **binary classifier**

[ROC] AUC is a performance measure of a **binary classifier**

- Binary target

  (yes/no; good/bad; spam/ham; infected/healthy)

[ROC] AUC is a performance measure of a **binary classifier**

- Binary target

  (yes/no; good/bad; spam/ham; infected/healthy)

- Ordinal or numeric **„score”**

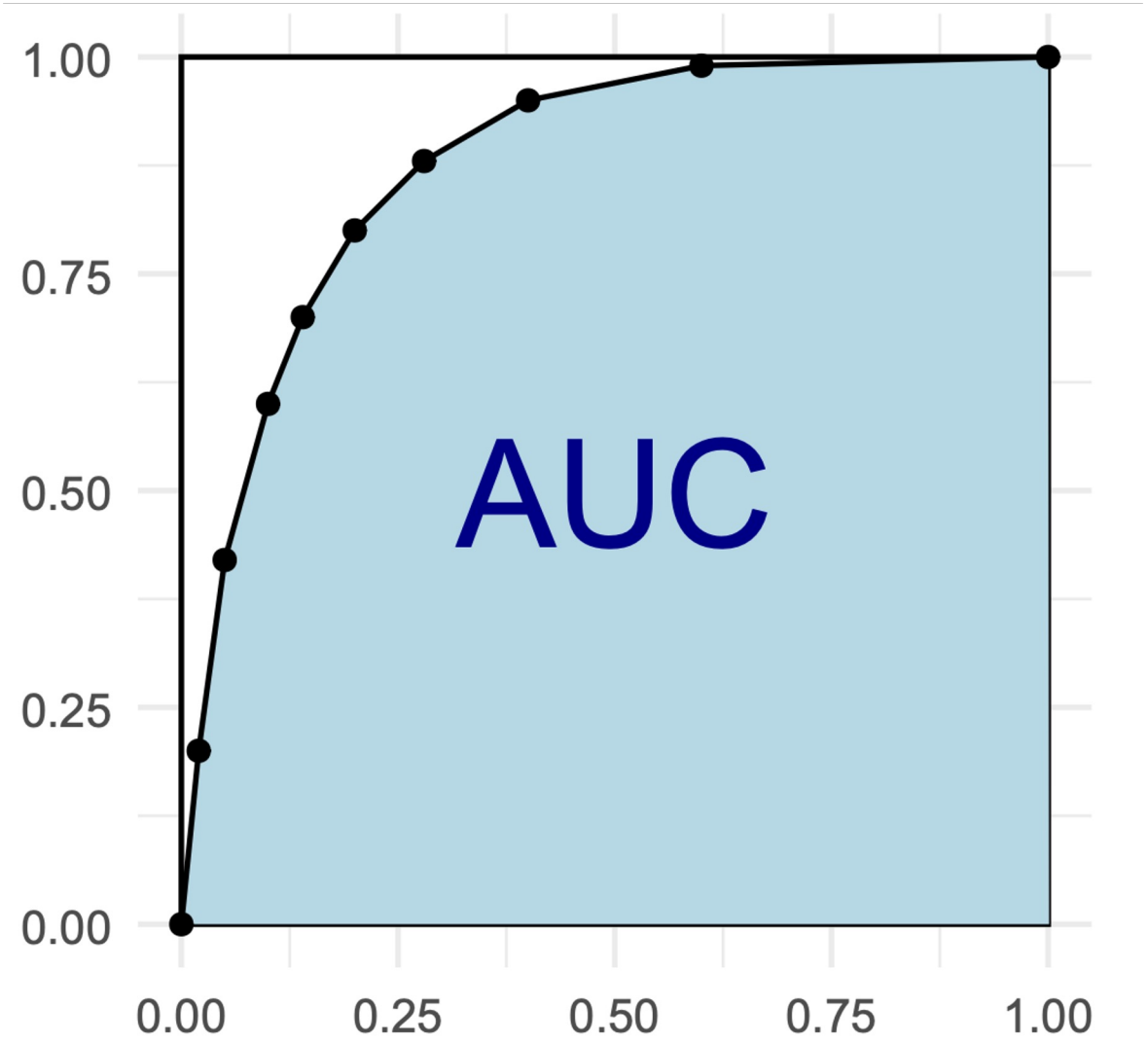
  (logit from regression model; credit scoring/rating; biomarker…)

```
> head(df)
  id probability_score actual
1 71          0.5876006      0
2 27          0.6025248      1
3 94          0.4051917      0
4  5          0.7409907      1
5 86          0.2150401      0
6 20          0.8574268      1
```

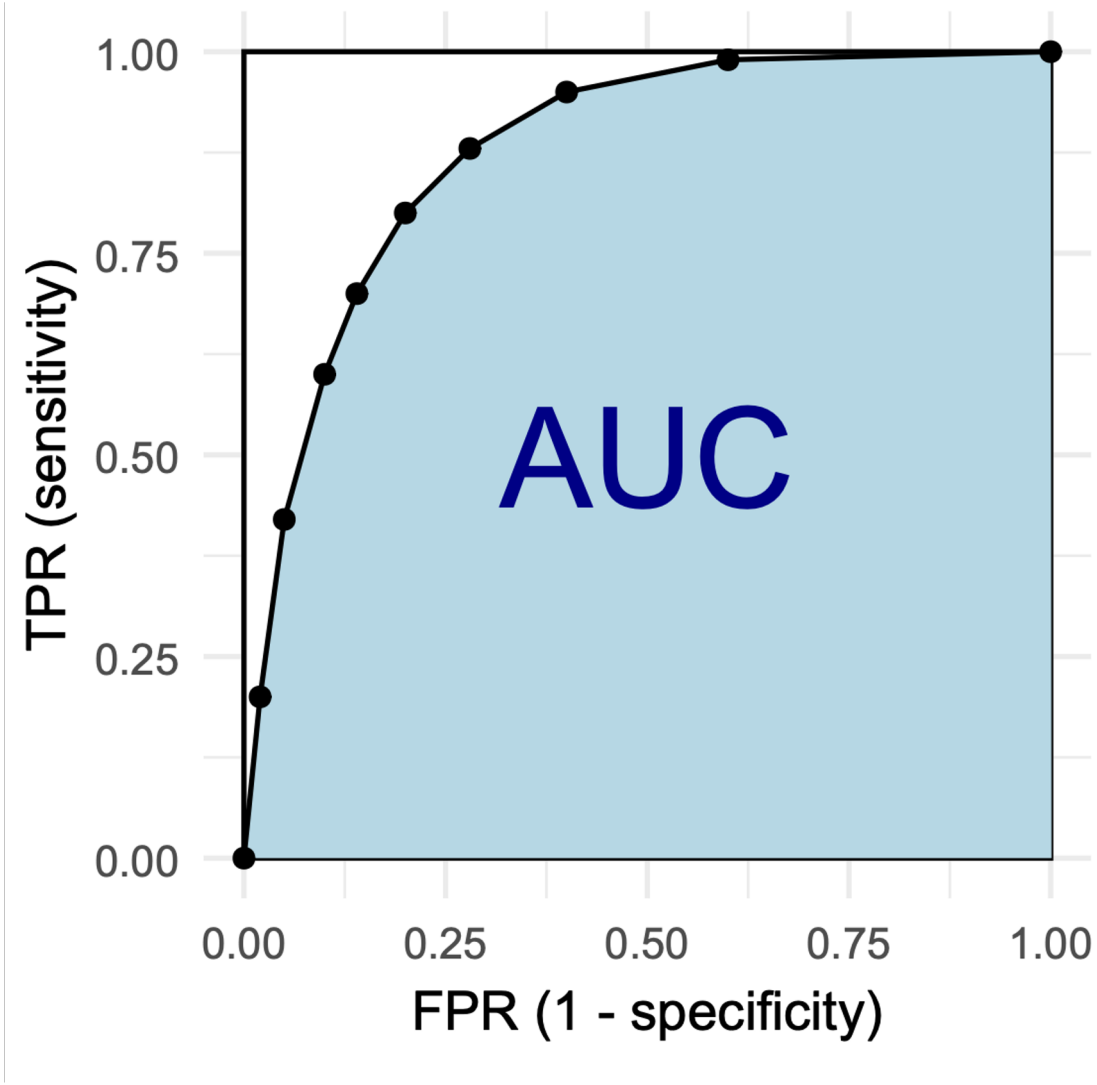
```
> head(df)
  id credit_rating credit_score default_flag
1 69           BBB          663            0
2 31            AA          790            0
3 38           CCC          500            1
4 49           BBB          650            0
5 89           BBB          684            0
6 11            AA          761            0
```

```
> head(df)
  id HbA1C diabetes_flag
1 44   5.6             0
2 17   5.3             0
3 22   6.1             1
4 77   5.1             0
5 18   6.8             1
6 62   4.7             0
```

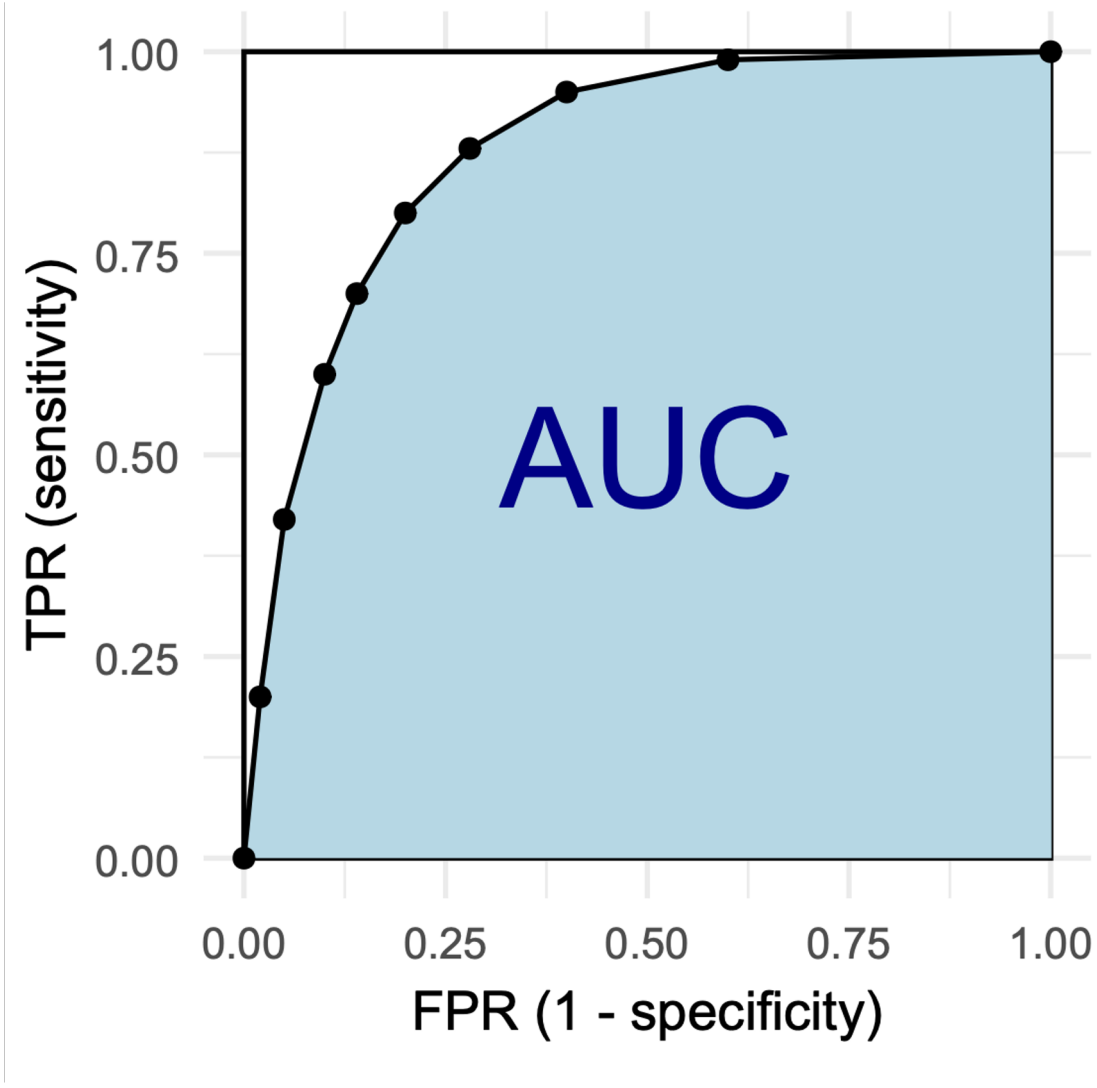
AUC = Area under the Curve



AUC = Area under the **ROC** Curve



AUC = Area under the **ROC** Curve

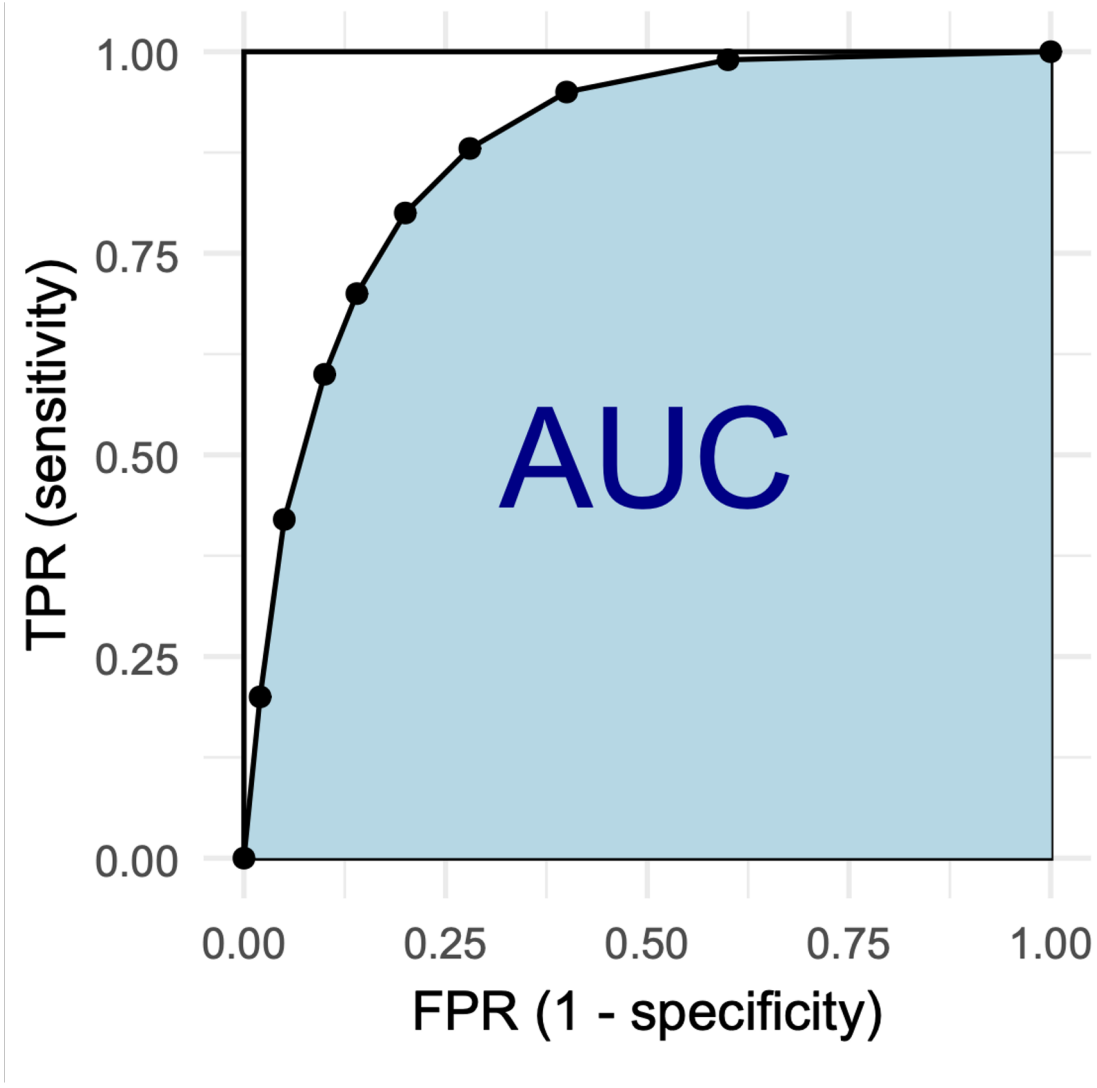


ROC = receiver operating characteristic Curve

plots True Positive Rate (TPR) against False Positive Rate (FPR)

for many (or all) score cut-off levels

AUC = Area under the **ROC** Curve

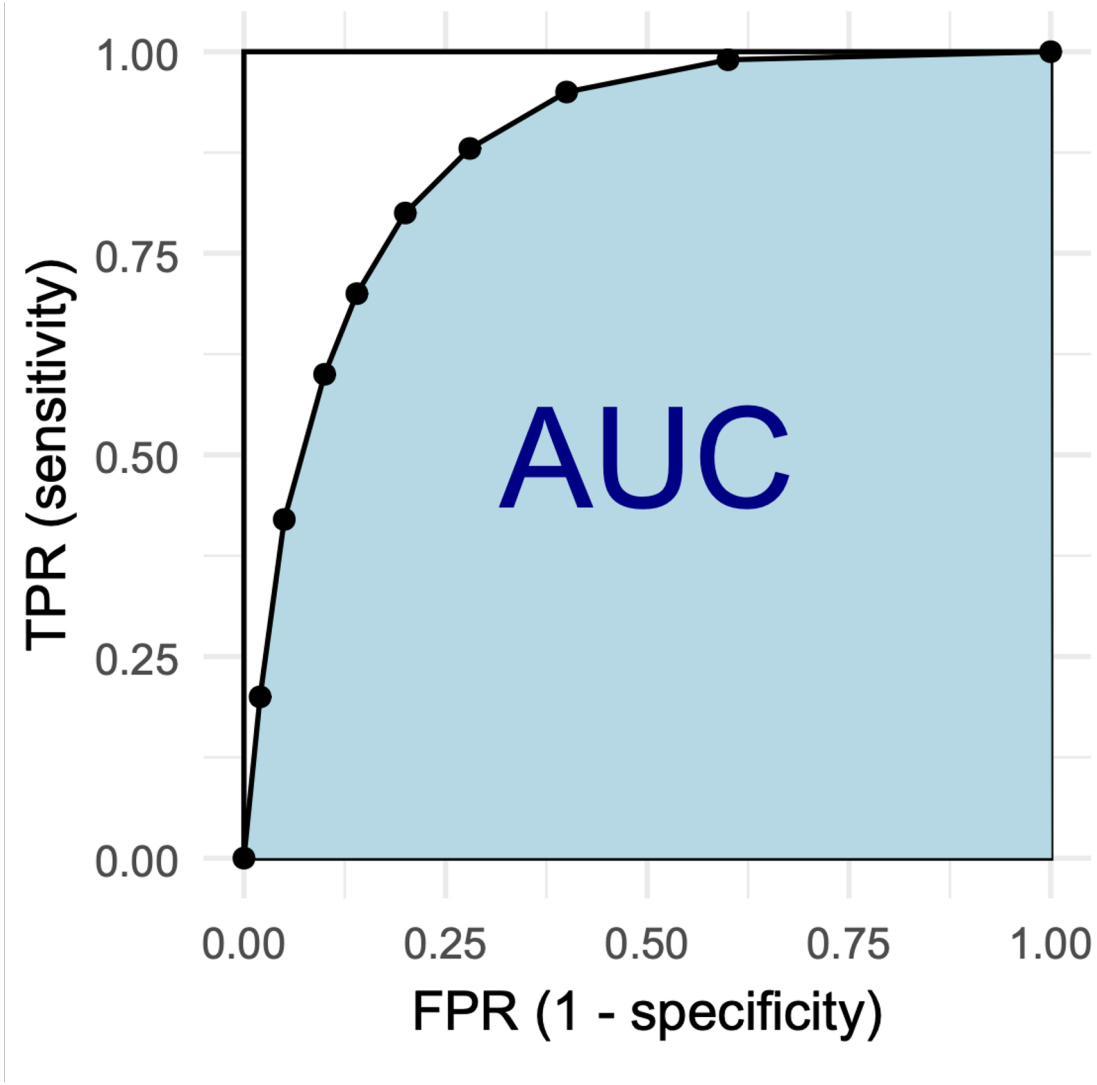


ROC = receiver operating characteristic Curve

plots sensitivity (or: recall) against 1 minus specificity

for many (or all) score cut-off levels

AUC = Area under the **ROC** Curve

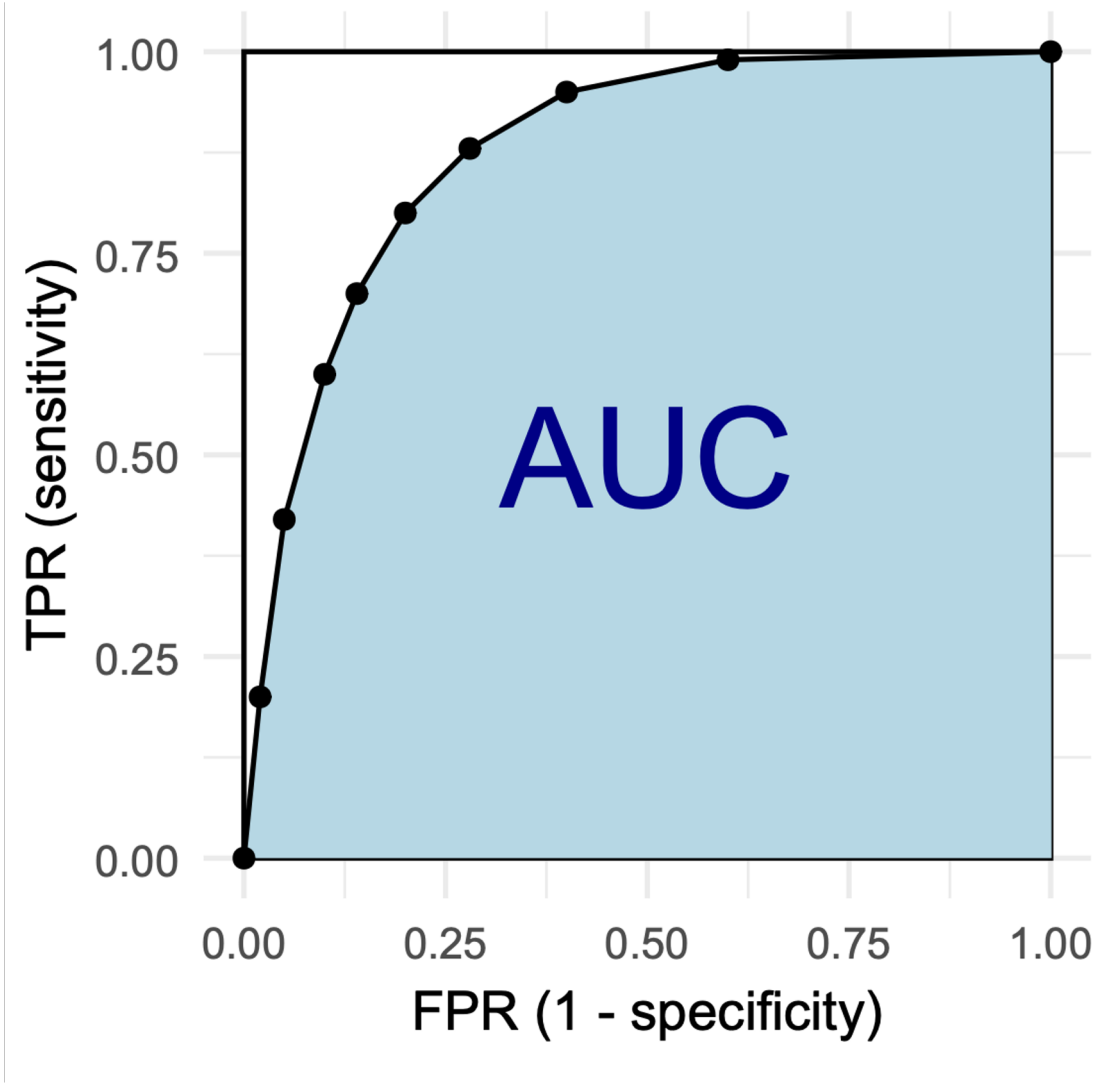


ROC = receiver operating characteristic Curve

plots probability of detection against probability of false alarm

for many (or all) score cut-off levels

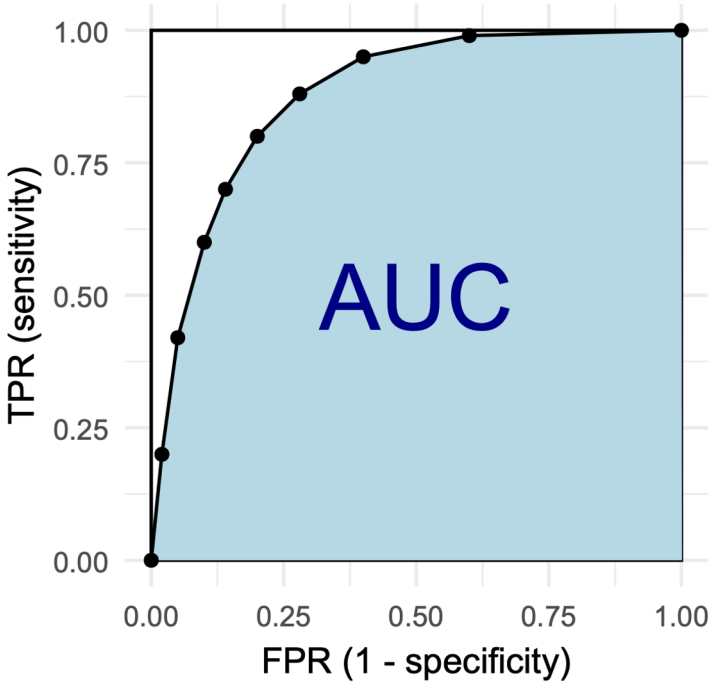
AUC = Area under the **ROC** Curve



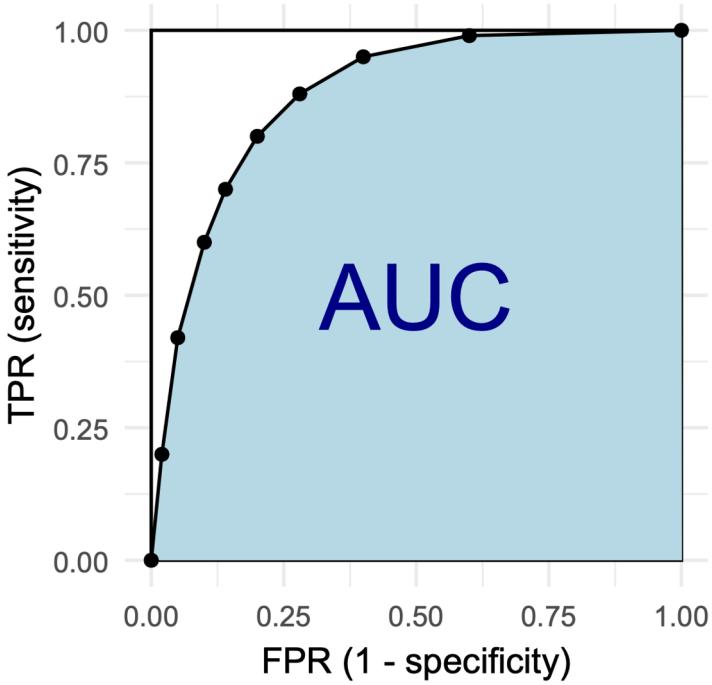
ROC = receiver operating characteristic Curve

plots hit rate
against fall out

for many (or all) score cut-off levels

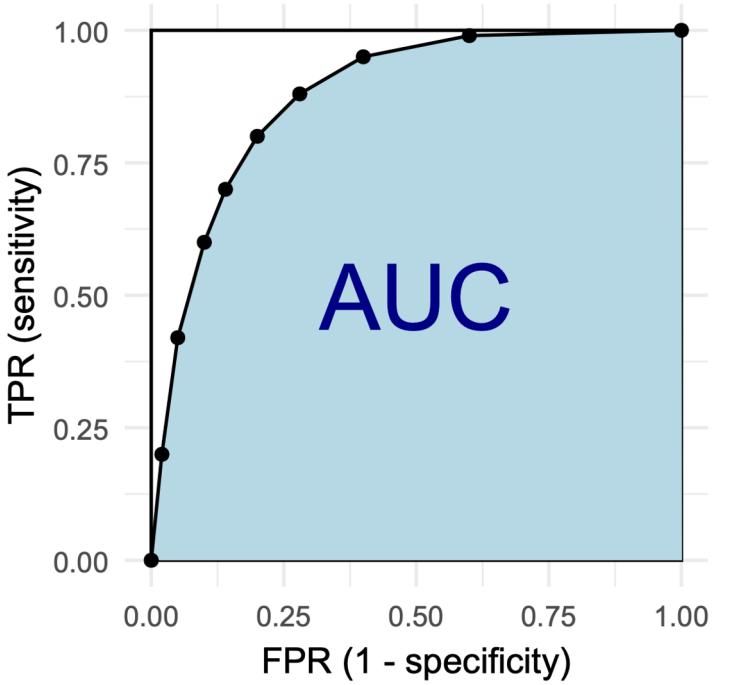


AUC is also known by several other names



AUC is also known by several other names

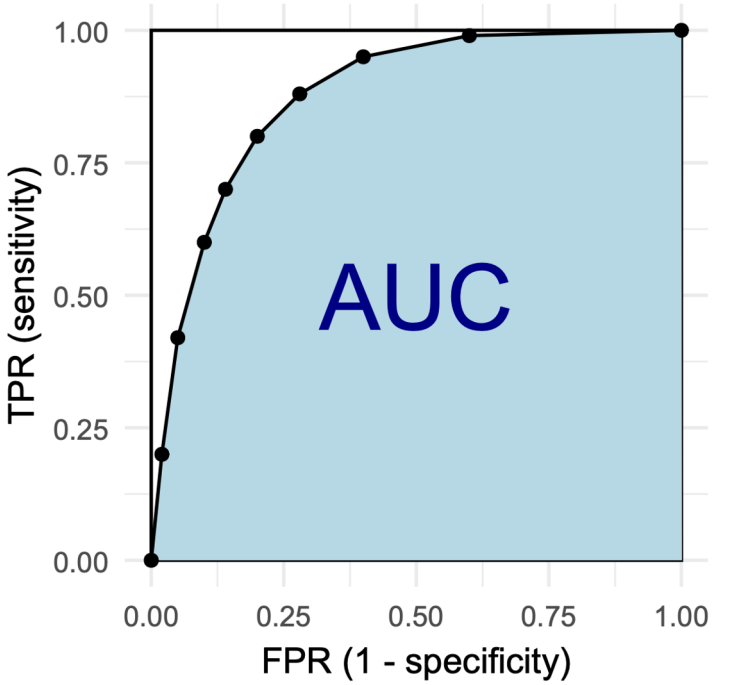
- AUROC
- ROC score
- concordance statistic (C-statistic)
- Vargha-Delaney A statistic
- Brunner-Munzel test statistic
- relative treatment effect
- probability of superiority
- measure of stochastic superiority (α)
- [a version of] Common Language Effect Size



AUC is also known by several other names

- **probability of superiority**

$$\text{AUC} = \mathbb{P}(\text{score}_1 > \text{score}_0) + \tfrac{1}{2}\,\mathbb{P}(\text{score}_1 = \text{score}_0)$$

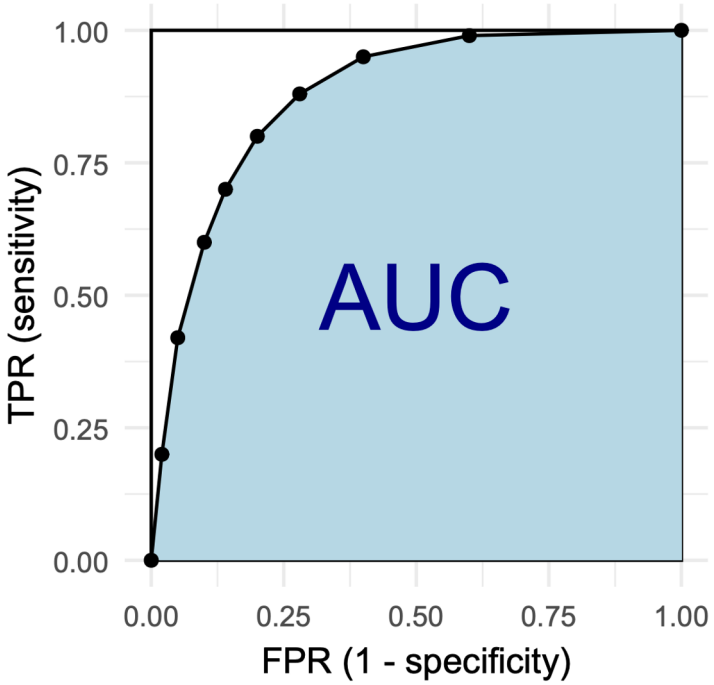


AUC is also known by several other names

- **probability of superiority**

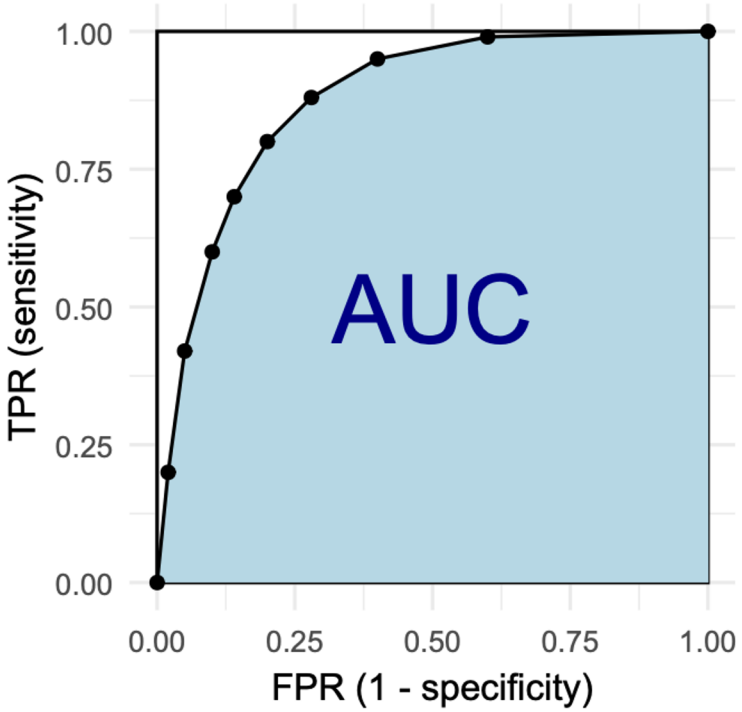
$$\text{AUC} = \mathbb{P}(\text{score}_1 > \text{score}_0) + \tfrac{1}{2}\,\mathbb{P}(\text{score}_1 = \text{score}_0)$$

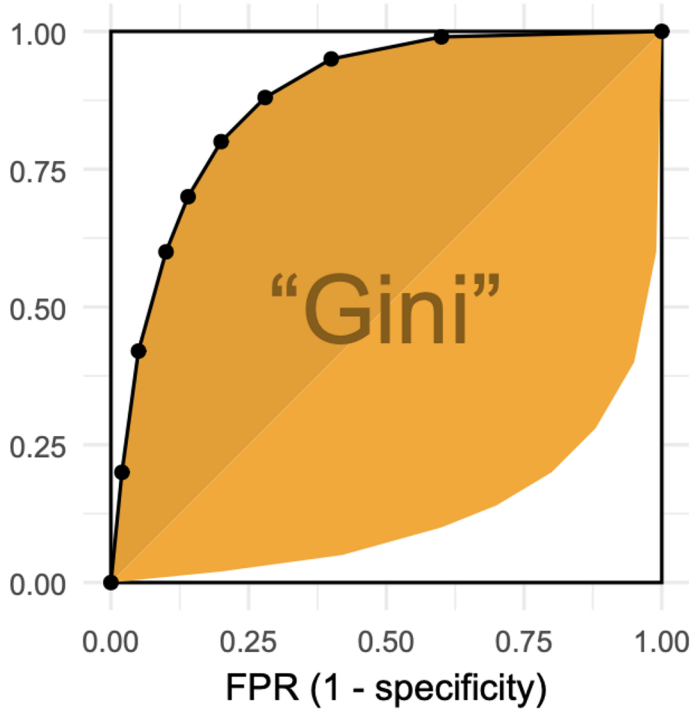
...if we randomly sample a pair of “0” and “1”.



AUC is related to Mann-Whitney U statistic:

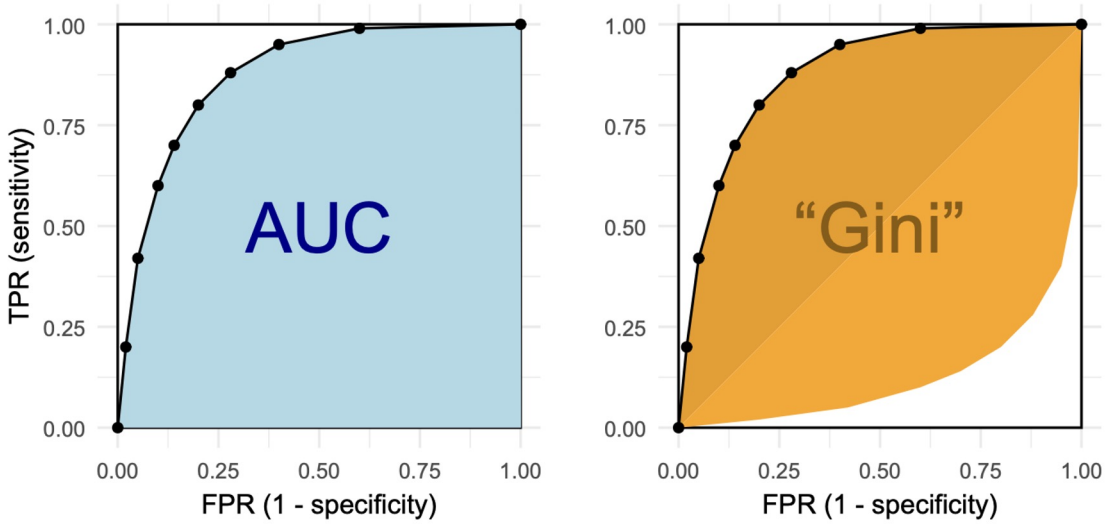
$$\mathrm{AUC} = \frac{U}{n_1 n_0}$$





$$\text{Gini} = 2 \times \text{AUC} - 1$$

$$\text{Gini} = 2 \times \text{AUC} - 1$$



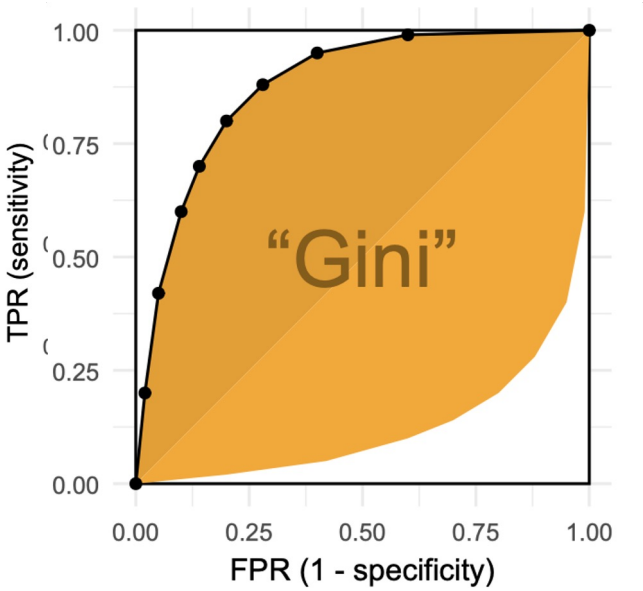
Perfect model AUC =1.00 Gini = 1.00

Not a bad model AUC = 0.80 Gini = 0.60

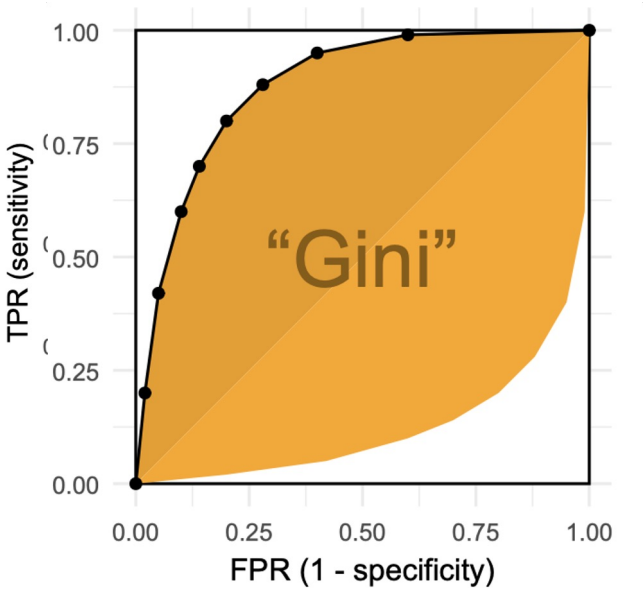
?

✓

Useless model AUC = 0.50 Gini =



„Gini” is also known by several other names



„Gini" is also known by several other names

- "pseudo-Gini"
- ROC skill score (ROCSS)
- Cliff's delta
- Accuracy Ratio (AR) based on Cumulative Accuracy Profile (CAP) curve
- Glass rank-biserial correlation coefficient
- [A special case of] Somers' D statistic (where Y is binary)
- Δ measure of stochastic superiority

Computing

AUC, AUROC, ROC score, concordance statistic, C-statistic, Vargha-Delaney A statistic, Brunner-Munzel test statistic, relative treatment effect, probability of superiority, measure of stochastic superiority (α), Common Language Effect Size, “pseudo-Gini”, ROC skill score (ROCSS), Cliff’s delta, Accuracy Ratio (AR) based on Cumulative Accuracy Profile (CAP) curve, Glass rank-biserial correlation coefficient, Somers’ D statistic and Δ measure of stochastic superiority

efficiently with R

AUC calculation – three main approaches:

- Trapezoidal integration

$$\mathrm{AUC} = \sum_{k=1}^{m-1} (\mathrm{FPR}_{k+1} - \mathrm{FPR}_k) \cdot \frac{\mathrm{TPR}_{k+1} + \mathrm{TPR}_k}{2}$$

- Pairwise comparison

$$\mathrm{AUC} = \frac{1}{n_1 n_0} \sum_{i:y_i=0} \sum_{j:y_j=1} \left[\mathbb{I}(s_i < s_j) + \frac{1}{2} \mathbb{I}(s_i = s_j) \right]$$

- Rank-based

$$\mathrm{AUC} = \frac{\bar{R}_1 - n_1(n_1+1)/2}{n_0} \qquad \bar{R}_1 = \frac{1}{n_1} \sum_{i:y_i=1} \mathrm{Rank}(s_i)$$

Table 1: Functions computing AUC.

Package	Function	Usage	Method	Language
bigstatsr	AUC	`AUC(pred, target)`	Pairwise comparison (optimized)	C++, R
caTools	colAUC	`colAUC(pred, target)[1, 1]`	Rank sum	R
cvAUC	AUC	`AUC(pred, target)`	Trapezoidal rule	R
DescTools	Cstat	`Cstat(pred, target)`	Pairwise comparison (optimized)	C++, R
effsize	VD.A	`VD.A(-pred ~ factor(target))$estimate`	Rank sum	R
fbroc	boot.roc	`boot.roc(pred, as.logical(target))$auc`	Trapezoidal rule	C++, R
Hmisc	somers2	`somers2(pred, target)["C"]`	Rank sum	R
Metrics	auc	`auc(target, pred)`	Rank sum	R
MLmetrics	AUC	`AUC(pred, target)`	Rank sum	R
mltools	auc_roc	`auc_roc(pred, target)`	Trapezoidal rule	R
ModelMetrics	auc	`auc(target, pred)`	Rank sum	C++, R
precrec	evalmod	`evalmod(mmdata(scores = pred, labels = target), mode="aucroc")$uaucs$aucs`	Rank sum	C++, R
pROC	auc	`auc(target, pred, lev=c('0', '1'), dir="<")`	Trapezoidal rule	R
PRROC	roc.curve	`roc.curve(pred[target == 1], pred[target == 0])$auc`	Trapezoidal rule	R
rcompanion	vda	`vda(x = pred[target == 1], y = pred[target == 0], digits=100)`	Rank sum	R
ROCit	rocit	`rocit(pred, target)$AUC`	Trapezoidal rule	R
ROCR	performance	`performance(prediction(pred, target), "auc")@y.values[[1]]`	Trapezoidal rule	R
scikit-learn	roc_auc_score	`import("sklearn.metrics")$roc_auc_score(target, pred)`	Trapezoidal rule	Python, C
scorecard	perf_eva	`perf_eva(pred, target, binomial_metric = "auc", show_plot=FALSE)$binomial_metric$dat$AUC`	Trapezoidal rule	R
yardstick	roc_auc_vec	`roc_auc_vec(as.factor(target), -pred)`	Trapezoidal rule	R
DescTools	SomersDelta	`SomersDelta(pred, target)/2 + 1/2`	Pairwise comparison (unoptimized)	C++, R

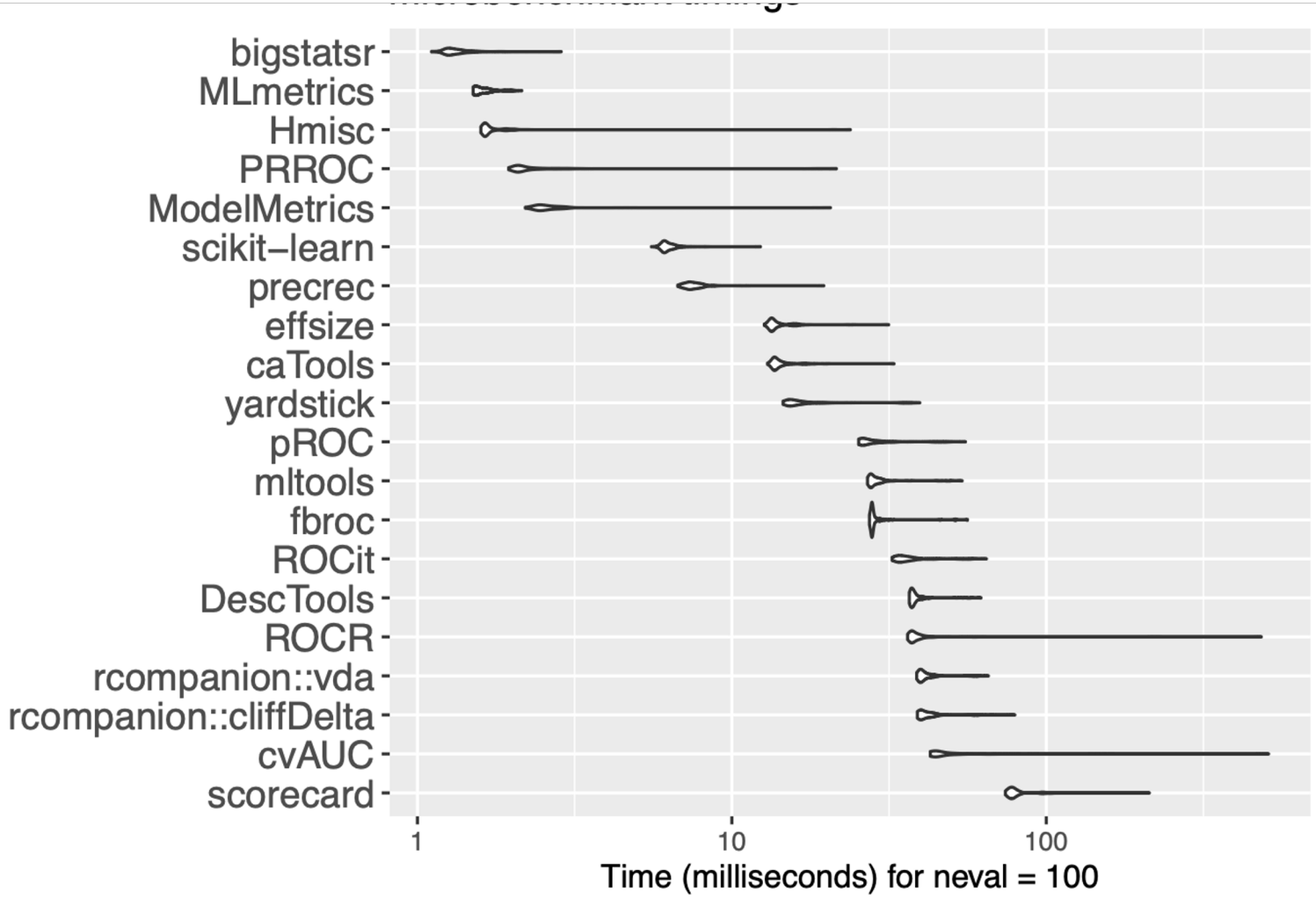


Figure 3: Benchmark of the computational efficiency of R packages for AUC computation using 10,000 observations.

Why do AUC implementations differ in efficiency?

- extra computations

  (other metrics,
  confidence intervals, plots)

- algorithm used

- algorithm optimization

- R vs C++

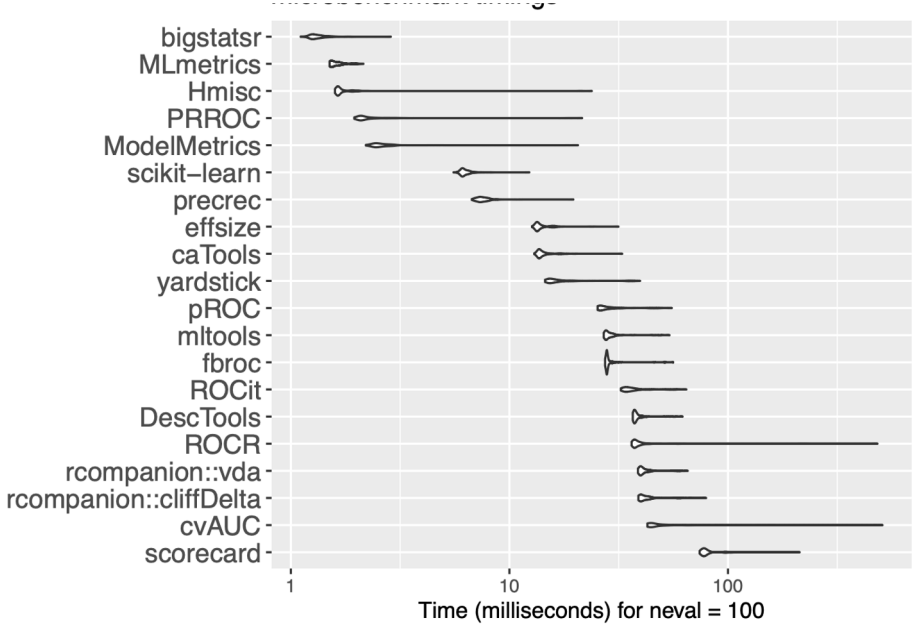


Figure 3: Benchmark of the computational efficiency of R packages for AUC computation using 10,000 observations.

AUC algorithms – empirical speed assessment

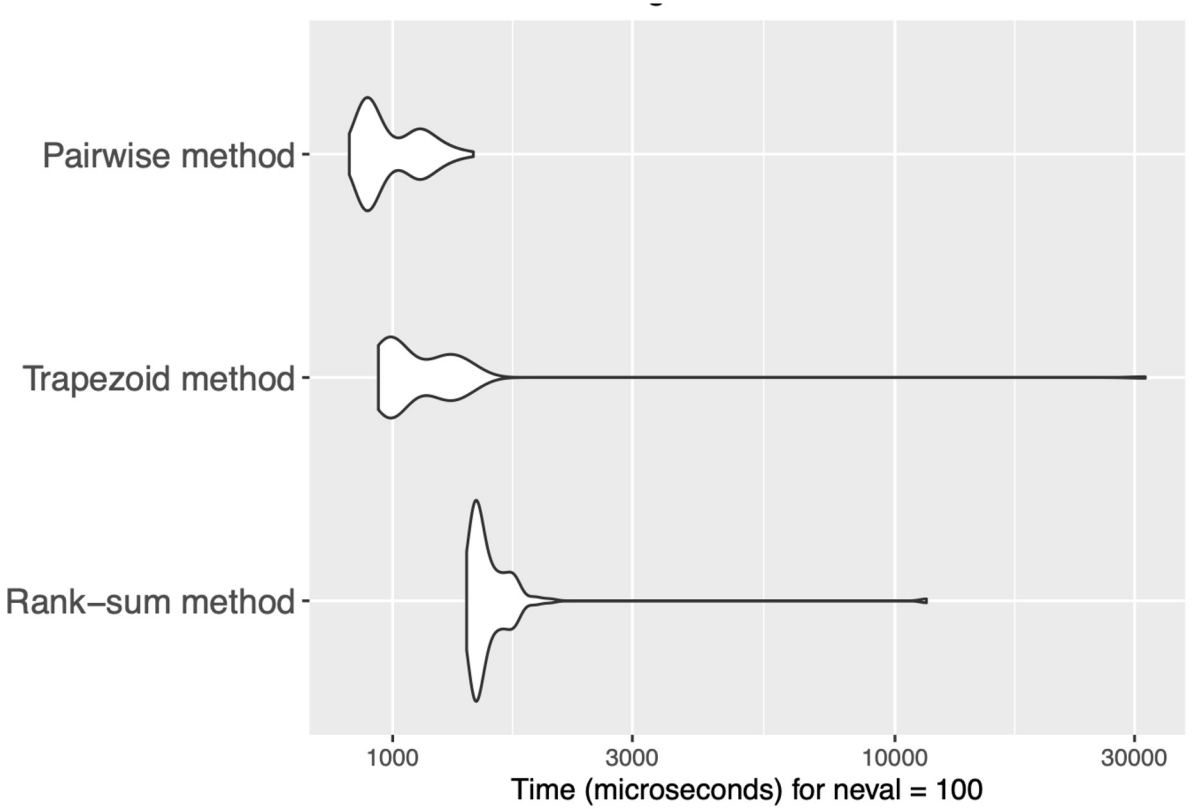


Figure 6: Benchmark of the computational efficiency of custom implementations for AUC computation using 10,000 observations.

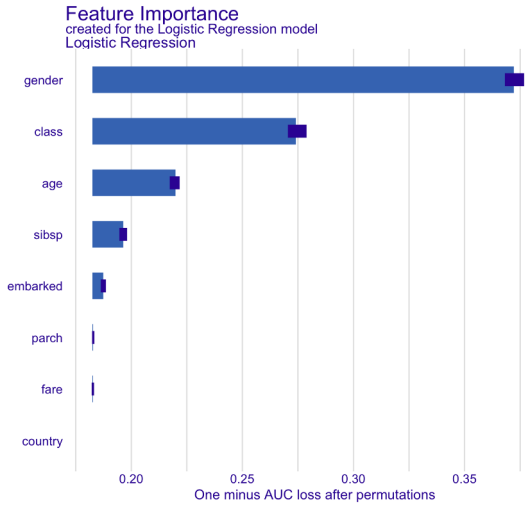
Why does AUC computation speed matter?

Important where AUC is computed many times:

- Bootstrap **confidence intervals** and **permutation tests**
- **Direct AUC optimization** in machine learning
- AUC-based **feature selection and feature importance**
- Large-scale banking market **simulations**

Case study

Optimized AUC computation in the DALEX explainability framework.



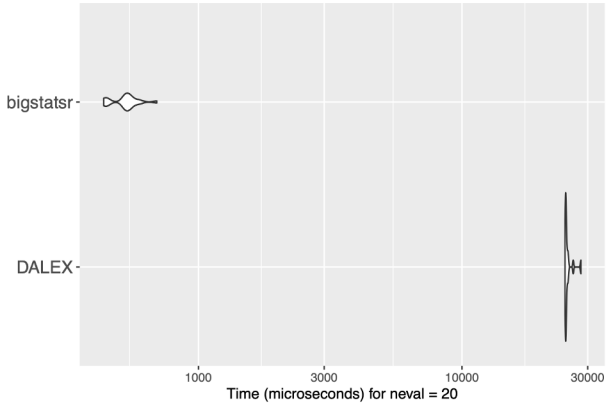


Figure 8: Benchmark showcasing the 1 - AUC loss function in DALEX and a wrapper function implementing bigstatsr.

- **Setup:** Logistic regression on the Titanic dataset (7 predictors); **Permutation Variable Importance** with 50 permutations per variable (350 AUC calculations).
- **Optimization:** Replaced **the default 1 − AUC** loss function with a **bigstatsr-based implementation**.
- **Performance:** ~0.026 s faster per AUC calculation, saving about 9–11 s per importance plot.
- **Result:** 75% reduction in median computation time, demonstrating the value of fast AUC implementations in repetitive evaluation tasks.

github.com/przem-pep/AUCwR

README

przypadku konieczności wielokrotnego obliczania AUC.

EN

Repository for project: **AUC Measure – review and efficient computation in R**

Project description:

The AUC (Area Under the Curve) measure is widely used in statistical classification and machine learning, including credit scoring, where it is employed to assess the quality of predictive models. The goal of this project is to review methods for calculating the AUC measure, followed by an analysis of the efficiency of computing this measure in R.

Project outcomes:

- Summary of AUC calculation methods and their applications;
- Benchmarking different AUC implementations in R;
- Recommendations for efficient AUC computation in practice.

Hypothesis:

Some methods of calculating AUC in R are significantly more efficient in terms of computation time and resource usage than others, while maintaining the same result precision. This may be of particular importance in cases where AUC needs to be computed multiple times.

https://github.com/przem-pep/AUCwR

Computing

AUC, AUROC, ROC score, concordance statistic, C-statistic, Vargha-Delaney A statistic, Brunner-Munzel test statistic, relative treatment effect, probability of superiority, measure of stochastic superiority (α), Common Language Effect Size, “pseudo-Gini”, ROC skill score (ROCSS), Cliff’s delta, Accuracy Ratio (AR) based on Cumulative Accuracy Profile (CAP) curve, Glass rank-biserial correlation coefficient, Somers’ D statistic and Δ measure of stochastic superiority

efficiently with R

Authors

Przemysław Peplinski
Piotr Geremek
Błażej Kochański (Politechnika Gdańska)
Wiktor Galewski
Miriam Nieslona