Examples

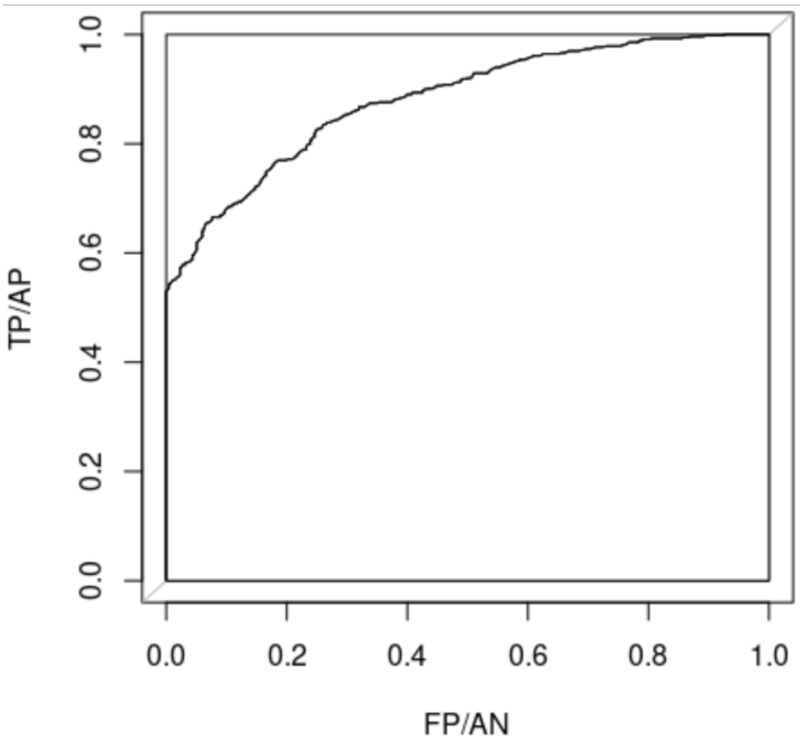
Given a data set, the following `rocc` function generates the coordinates of the points of a ROC curve. The `rocc` function parameters are the dataset, the column containing the estimates and the column containing the actuals.

```
rocc <- function(dataframe, ind1, ind2) {
  dataframe <- dataframe[order(dataframe[ind1], decreasing = TRUE),]
  predictions <- dataframe[[ind1]]
  labels <- dataframe[[ind2]]
  x<- c(0)
  y<-c(0)
  for(i in 1:length(predictions)){
    TP <- 0
    TN <- 0
    FP <- 0
    FN <- 0
    if(i!=1){
      if(predictions[i] == predictions[i-1]){
        next
      }
    }
    for(j in 1:length(predictions)){
      if(predictions[j] < predictions[i]){
        if(labels[j] == 0){
          TN <- (TN+1)
        }else{
          FN <- (FN+1)
        }
      }else{
        if(labels[j] == 0){
          FP <- (FP+1)
        }else{
          TP <- (TP+1)
        }
      }
    }
    TPR <- TP/(TP+FN)
    FPR <- 1 - TN/(TN+FP)
    x <- c(x,FPR)
    y <- c(y,TPR)
  }
  x<-c(x,1)
  y<-c(y,1)
  roc <- list(x,y)
  names(roc) <- c("x", "y")
  invisible(roc)
}
```

The following code

```
data1=read.csv("data_for_ROC1.csv")
n=nrow((data1))
AP=sum(data1$labels)
AN=n-AP
firstROCcurve <- rocc(data1, 1, 2)
rra_result1 <- rra(firstROCcurve$x, firstROCcurve$y, AP, AN, plot = FALSE)
rra.plot(rra_result1, cROI = "white", cUnder = "white", main="")
```

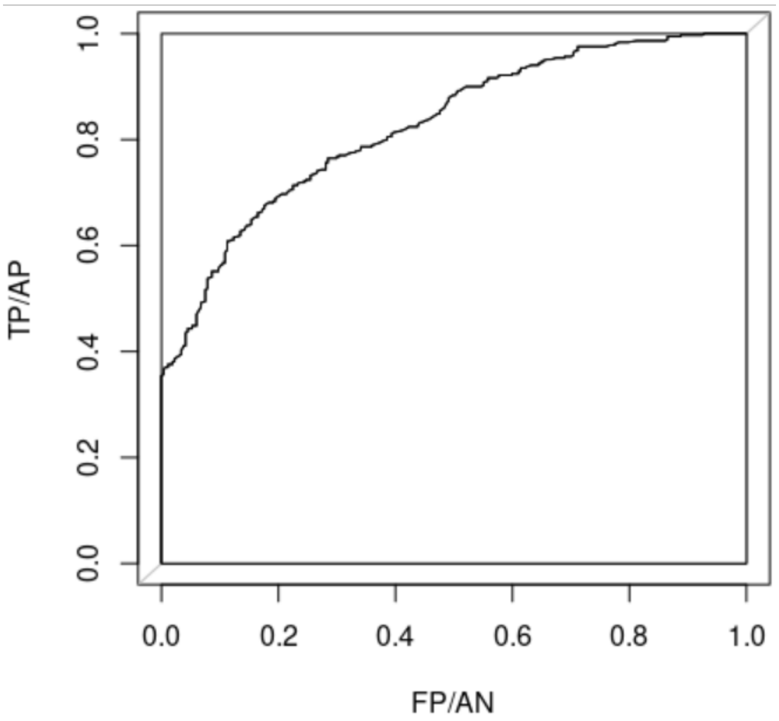
reads data from file "data_for_ROC1.csv" (which contains 1000 data points, of which 565 positives and 435 negatives), computes the ROC curve and plots it, as shown below



The following code

```
data2=read.csv("data_for_ROC2.csv")
n=nrow((data2))
AP=sum(data2$labels)
AN=n-AP
secondROCcurve <- rocc(data2, 1, 2)
rra_result2 <- rra(secondROCcurve$x, secondROCcurve$y, AP, AN, plot = FALSE)
rra.plot(rra_result2, cROI = "white", cUnder = "white", main="")
```

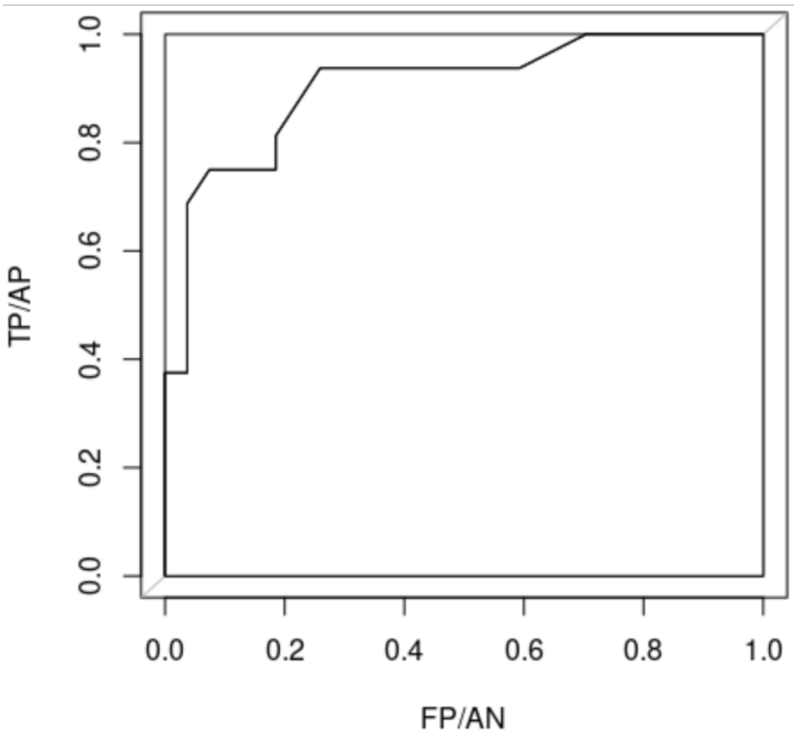
Reads data from file "data_for_ROC2.csv" (which contains 637 data points, of which 370 positives and 267 negatives), computes the ROC curve and plots it, as shown below



The following code

```
dataBerek=read.csv("berek.csv")
dataBerek$bug=ifelse(dataBerek$bug>0,1,0)
n=nrow((dataBerek))
AP=sum(dataBerek$bug)
AN=n-AP
berekWMCROCcurve <- rocc(dataBerek, 4, 24)
rocWMC <- rra(berekWMCROCcurve$x, berekWMCROCcurve$y, AP, AN, plot = FALSE)
rra.plot(rocWMC, cROI = "white", cUnder = "white", main="")
```

Reads data from file "berek.csv", which contains data for the berek project (from the PROMISE repository). The dataset contains 43 data points concerning classes: of these 16 are buggy (positive) and 27 are bug-free (negative). The code computes the ROC curve (using as predictor the WMC measure in column 4) and plots it, as shown below.



Note that the **`rra`** function also outputs the value of the AUC of the ROC curve.

Similar code can be used to obtain the ROC curves when LOC and CA measures are used as predictors.

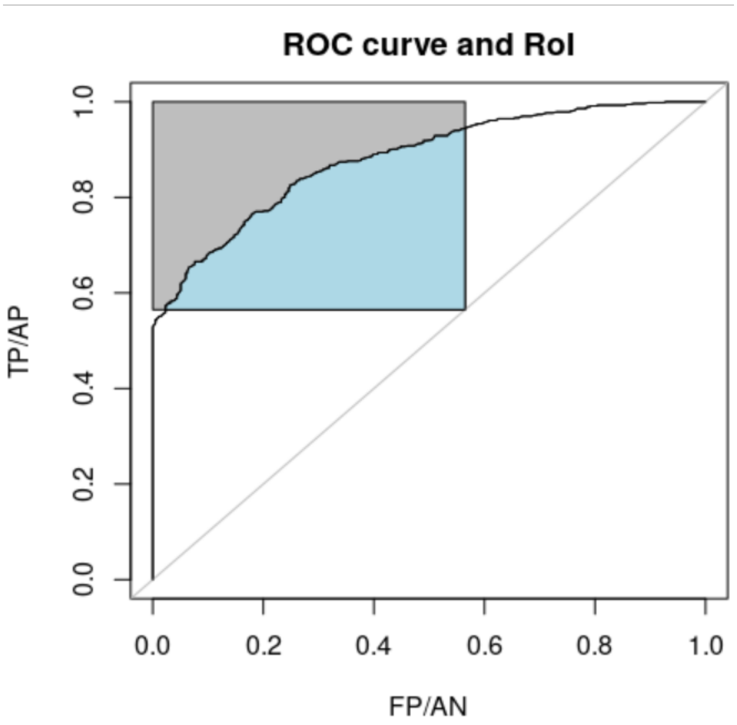
To compute RRA(TPR, FPR) the following code can be used

```
n=nrow((data1))
AP=sum(data1$labels)
AN=n-AP
temp=rra(firstROCcurve$x, firstROCcurve$y, AP, AN, recall=TRUE, fallout=TRUE)
```

This code prints the RRA:

```
The Ratio of Relevant Areas (RRA) is equal to 0.548289
```

and shows the RoI:



Having saved the result of the **`rra`** function in the **`temp`** variable, the RRA value is available also as **`temp$rra`**.

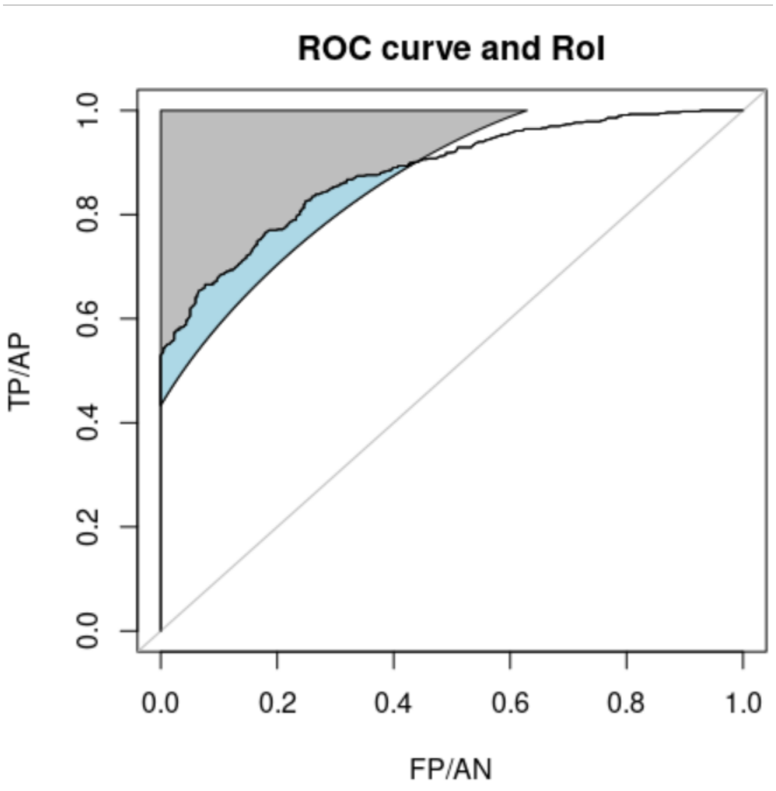
To compute RRA([illegible]=0.5) the following code can be used

```
rra(firstROCcurve$x, firstROCcurve$y, AP, AN, phi=TRUE, c_phi=0.5)
```

This code prints the RRA:

```
The Ratio of Relevant Areas (RRA) is equal to 0.1990632
```

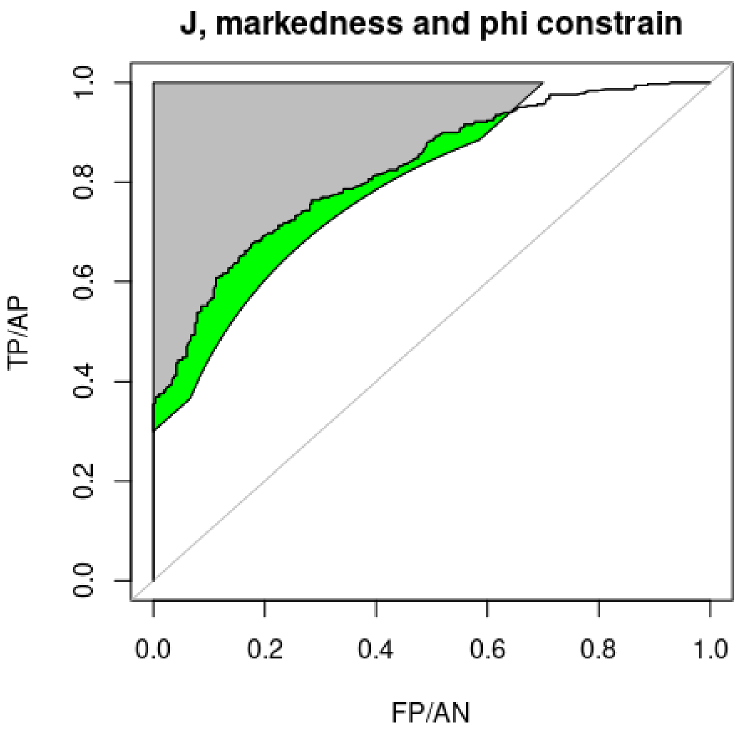
and shows the RoI:



The following code computes RRA when the RoI is constrained by both f>0.4 markedness>0.4 and Youden's J>0.3. In addition, the area under the curve belonging to the RoI is colored in green.

```
n=nrow((data2))
AP=sum(data2$labels)
AN=n-AP
rra(secondROCcurve$x, secondROCcurve$y, AP, AN, j=TRUE, c_j=0.3,
    markedness=TRUE, c_markedness=0.4, cUnder="green", main = "J and phi constrain")
```

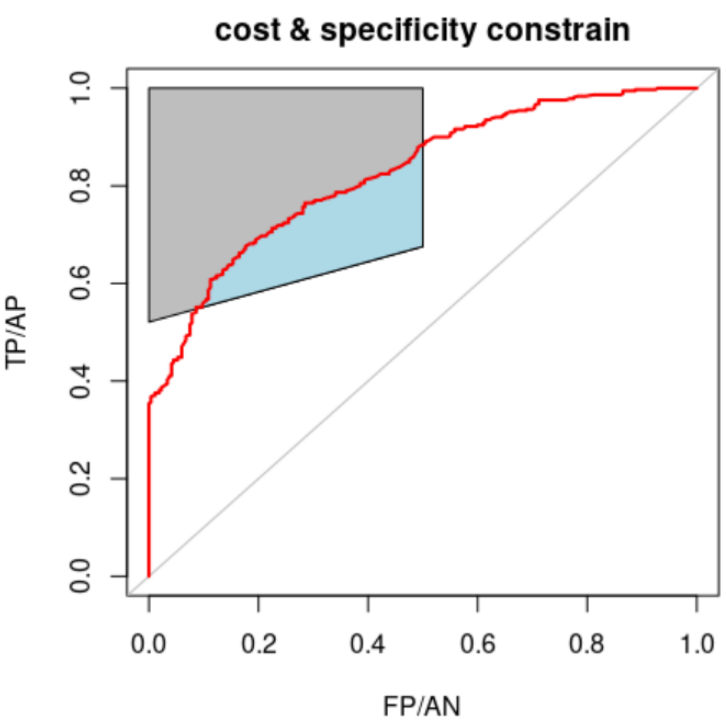
The code computes RRA (`The Ratio of Relevant Areas (RRA) is equal to 0.1801198`) and shows the RoI:



The following code computes RRA when the RoI is constrained by specificity and normalized cost. Namely, specificity is better than the average specificity of a uniform random policy ("uni") with p(m) = 0.5, and the normalized cost is less than 0.8 NC_{rnd}(lambda=0.7), where NC_{rnd}(lambda=0.8) is the averge normalized cost of random estimation, when lambda=0.7.

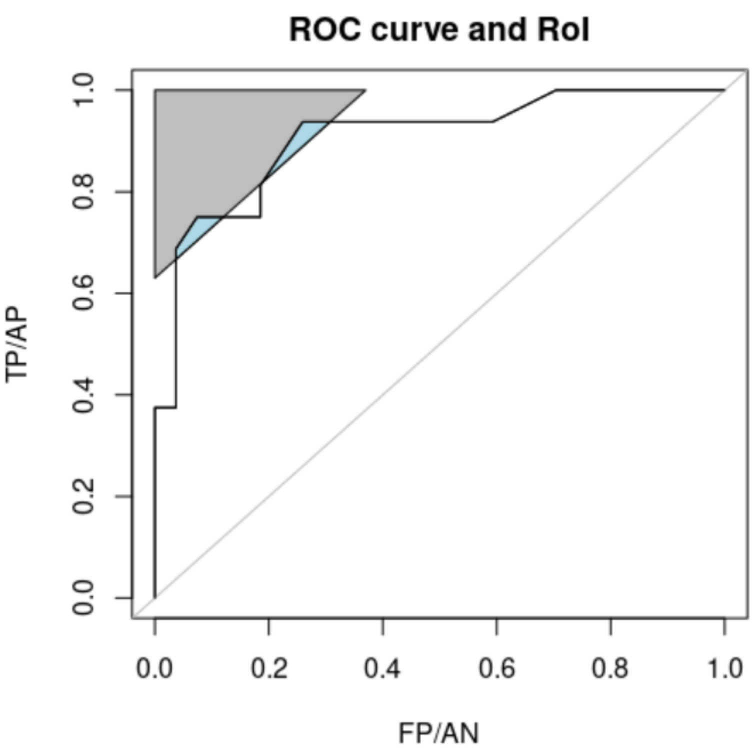
```
rra(secondROCcurve$x, secondROCcurve$y, AP, AN, ncost=TRUE, lambda=0.7, mu=0.8,
    specificity=TRUE, c_specificity="uni",  p_specificity=0.5, col="red",
    lwd=2, main="cost & specificity constrain")
```

As usual, the code computes RRA `(The Ratio of Relevant Areas (RRA) is equal to 0.2669933)` and shows the RoI:



The area under the curve belonging to the RoI may be formed by disjoint pieces, as in the following example.

```
n=nrow((dataBerek))
AP=sum(dataBerek$bug)
AN=n-AP
rra(berekWMCROCcurve$x, berekWMCROCcurve$y, AP, AN, j = TRUE, c_j = 0.63)
```



The following examples illustrate the usage of rra.test, a function that can be used to compare two ROC curves.

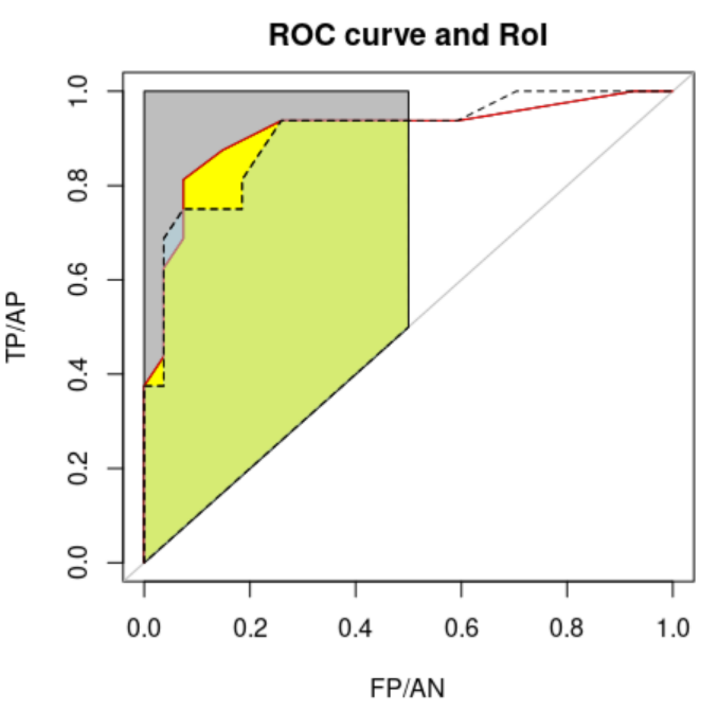
Suppose we want to compare the diagnostic tests based on WMC and AC for the berek dataset.
The following code performs the comparison, using as the RoI the region where precision is better than average random classification with p=AP/n and fallout is better than the average fallout of a uniform random policy ("uni") with p(m) = 0.5.

```
rra.test(berekWMCROCcurve$x, berekWMCROCcurve$y, berekACROCcurve$x, berekACROCcurve$y,
         AP, AN, precision=TRUE, fallout=TRUE, c_fallout="uni", p_fallout=0.5)
```

The code computes that:

```
The second ROC curve (red) has a better RRA value than the first one (black)
```

In the plot, the first curve (**`berekWMCROCcurve`**) is shown as a dashed black line, while the second curve (**`berekACROCcurve`**) is showd as a red solid line.



Rra.test does not provide details. If you want to know the RRA value for the two curve you can use the following code:

```
resWMC=rra(berekWMCROCcurve$x, berekWMCROCcurve$y, plot=FALSE,
         AP, AN, precision=TRUE, fallout=TRUE, c_fallout="uni", p_fallout=0.5)
resAC=rra(berekACROCcurve$x, berekACROCcurve$y, plot=FALSE,
         AP, AN, precision=TRUE, fallout=TRUE, c_fallout="uni", p_fallout=0.5)
cat("RRA for WMC is ", resWMC$rra, ", RRA for AC is ", resAC$rra,
    "\n", sep="")
```

which shows that

```
RRA for WMC is 0.7716049, RRA for AC is 0.808642
```

You can also feed `rra.test` with the results provided by `rra`. For instance this code:

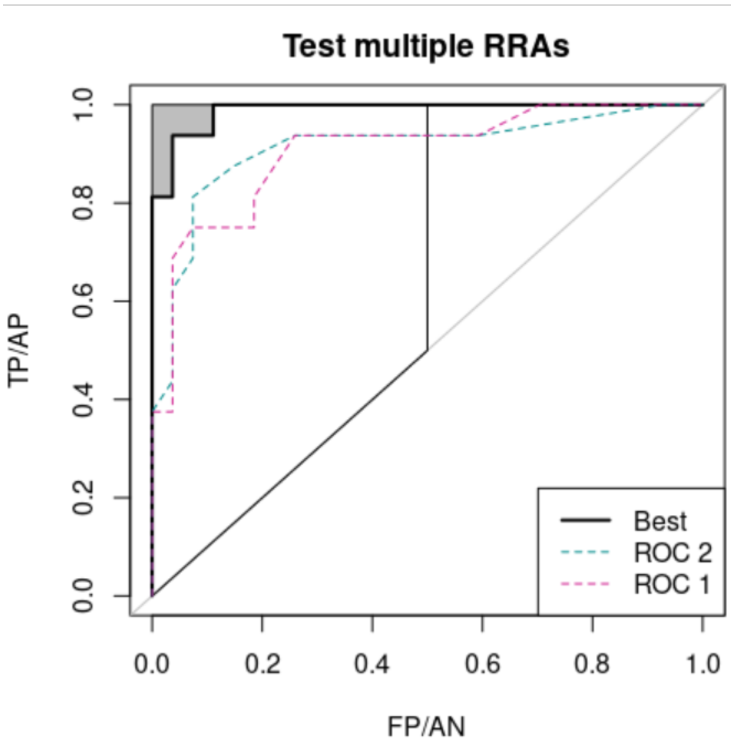
```
rra.test(resWMC, resAC)
```

provides the same picture shown above.

It is possible to compare more than two ROC curves, by supplying a list to rra.test, as in the following example:

```
resLOC=rra(berekLOCROCcurve$x, berekLOCROCcurve$y, plot=FALSE,
          AP, AN, precision=TRUE, fallout=TRUE, c_fallout="uni", p_fallout=0.5)

rra.test(list(resWMC, resAC, resLOC),legend = TRUE, cUnderBest="white",
         main = "Test multiple RRAs", colOther="midnight blue")
```


Test multiple RRAs
TP/AP
FP/AN
Best
ROC 2
ROC 1