ROC curve-Receiver operating characteristic

Sameh MAGDELDIN M.V.Sc, 2Ph.D

February 8, 2025

In this tutorial, i will present the receiver operating characteristics

(ROC curve)

```
library(pROC) #calculating Roc coordinates
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(rockr)#calculating Roc coordinates
```

```
## Loading required package: httr
```

```
library(ggplot2) # drawing
library(plyr) # data handling
library(dplyr)# data handling
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(gplots)
```

```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(knitr)
library (caret)
```

```
## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:httr':
##
##     progress
```

```r
library(magrittr) # chaining
library(ROCR) #calculating Roc coordinates
```

Overview on ROC curve principal

The ROC curve is a tool to evaluate the performance of binary classification

models. It visualizes the trade-off between TPR (TP/TP+FN) and FPR (FP/TN+FP)

It is commonly used to characterize the sensitivity/specificity

trades for a binary classifier

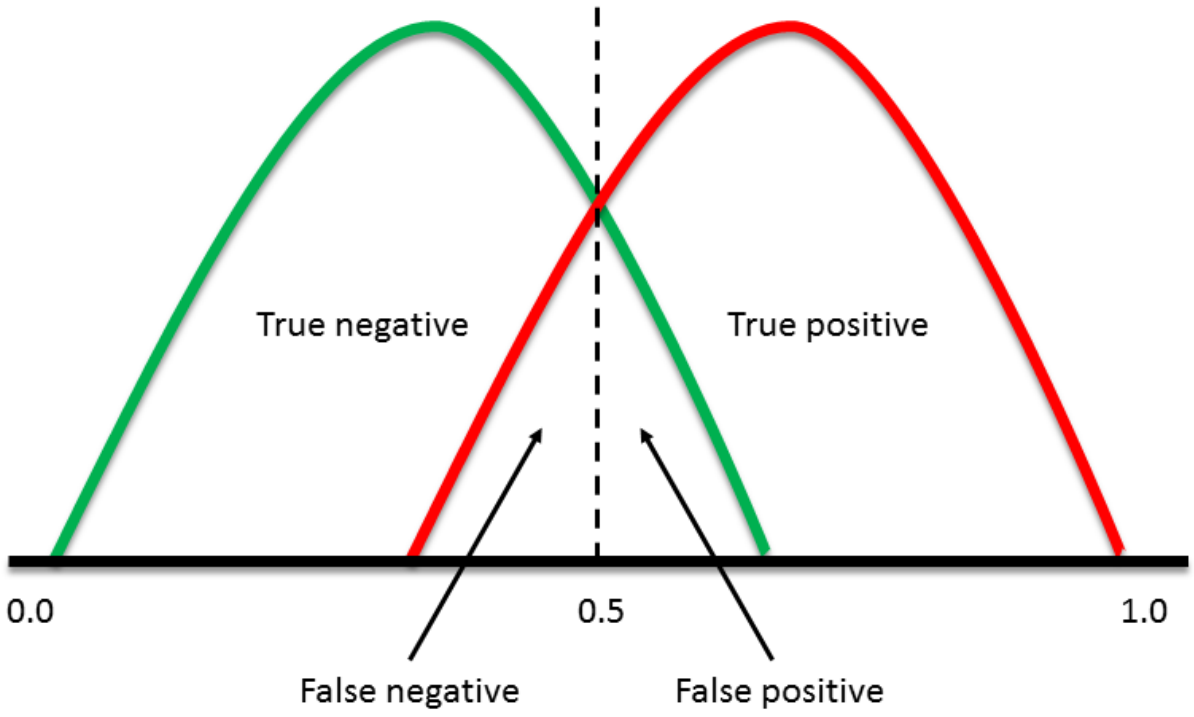
Further, it is used to see how well your classifier can separate positive and

negative identifier and to identify the best threshold for separating them.

To be able to use the ROC curve, your classifier has to be ranked

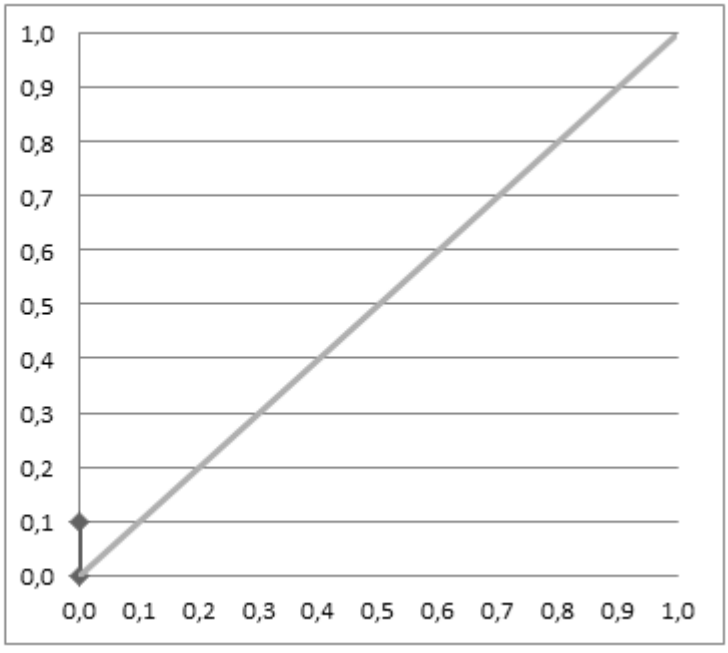
Quick overlook on the principle of the test

```
knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/ROC1.png")
```




```
knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/roc.png")
```

#	C	Score
1	P	0,9
2	P	0,8
3	N	0,7
4	P	0,6
5	P	0,55
6	P	0,54
7	N	0,53
8	N	0,52
9	P	0,51
10	N	0,505
11	P	0,4
12	N	0,39
13	P	0,38
14	N	0,37
15	N	0,36
16	N	0,35
17	P	0,34
18	N	0,33
19	P	0,3
20	N	0,1



```
#knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/rocx.GIF")
```

How it works

1. Variables are classified and turned into ranked value (score)
2. The ranked value is ordered by the score from highest to lowest
3. To draw the ROC curve, start in (0,0)(0,0) for each example xx in the sorted order

   If xx is positive, move 1/pos1/pos **Up** If xx is negative, move 1/neg1/neg **Right**

Where pos/pos and neg/neg are the fractions of positive and negative examples respectively.

On this graph, the yy-axis is true positive rate, and the xx-axis is false positive rate. Note that the diagonal line - this is the base line, that can be obtained with a random classifier. The far our ROC curve from this line,the better.

Lets try some data !!, load the rocdata.csv In this data, we need to determine the optimum cutoff value of albumin conc (conc) that differenciate between healthy and patient'sick patients.

```
# rocdata = read.csv((file.choose()))
#or
rocdata <- read.csv("C:/Users/magde/Desktop/R2025/8.ROC curve/rocdata.csv")
```

lets view the data

```
tail(rocdata)
```

```
##       X gos6 outcome gender age wfns conc  ndka
## 108 136    5    Good Female  68    4 0.47 10.33
## 109 137    4    Good   Male  53    4 0.17 13.87
## 110 138    1    Poor   Male  58    5 0.44 15.89
## 111 139    5    Good Female  32    1 0.15 22.43
## 112 140    5    Good Female  39    1 0.50  6.79
## 113 141    5    Good   Male  34    1 0.48 13.45
```

str function shows the class of each vairable

```
str(rocdata)
```

```
## 'data.frame':    113 obs. of  8 variables:
##  $ X      : int  29 30 31 32 33 34 35 36 37 38 ...
##  $ gos6   : int  5 5 5 5 1 1 4 1 5 4 ...
##  $ outcome: chr  "Good" "Good" "Good" "Good" ...
##  $ gender : chr  "Female" "Female" "Female" "Female" ...
##  $ age    : int  42 37 42 27 42 48 57 41 49 75 ...
##  $ wfns   : int  1 1 1 1 3 2 5 4 1 2 ...
##  $ conc   : num  0.13 0.14 0.1 0.04 0.13 0.1 0.47 0.16 0.18 0.1 ...
##  $ ndka   : num  3.01 8.54 8.09 10.42 17.4 ...
```

From these data, i would like to define an optimal cutoff point between good and poor observations (outcome) based on the albumin concentration data (conc).

lets first extract these 2 columns only (outcome and conc)

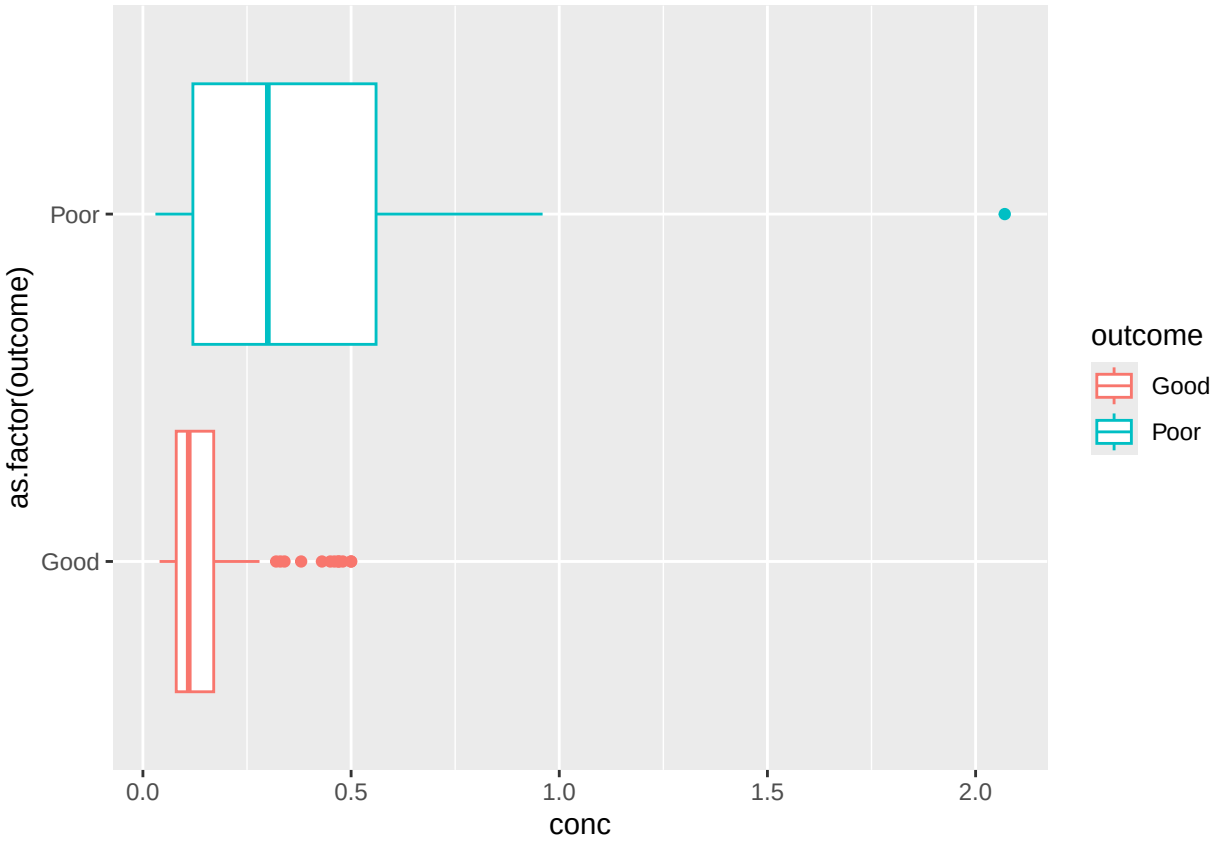
```
rocdata <- rocdata %>% select(outcome,conc) # %>% "piping or chaining mean
#literaly[then]"
```

%>% [Ctrl-Shift-M] <- [Alt + -]

lets plot the data

```
p <- ggplot(rocdata, aes(x=as.factor(outcome), y=conc, color=outcome)) +
  geom_boxplot()
```

```
p+ coord_flip() # just to flip the boxplots
```



as you can see, the poor value have higher albumin concentration than good

(normal) values.

Question: Can we define an optimal cutoff point that we can say above it

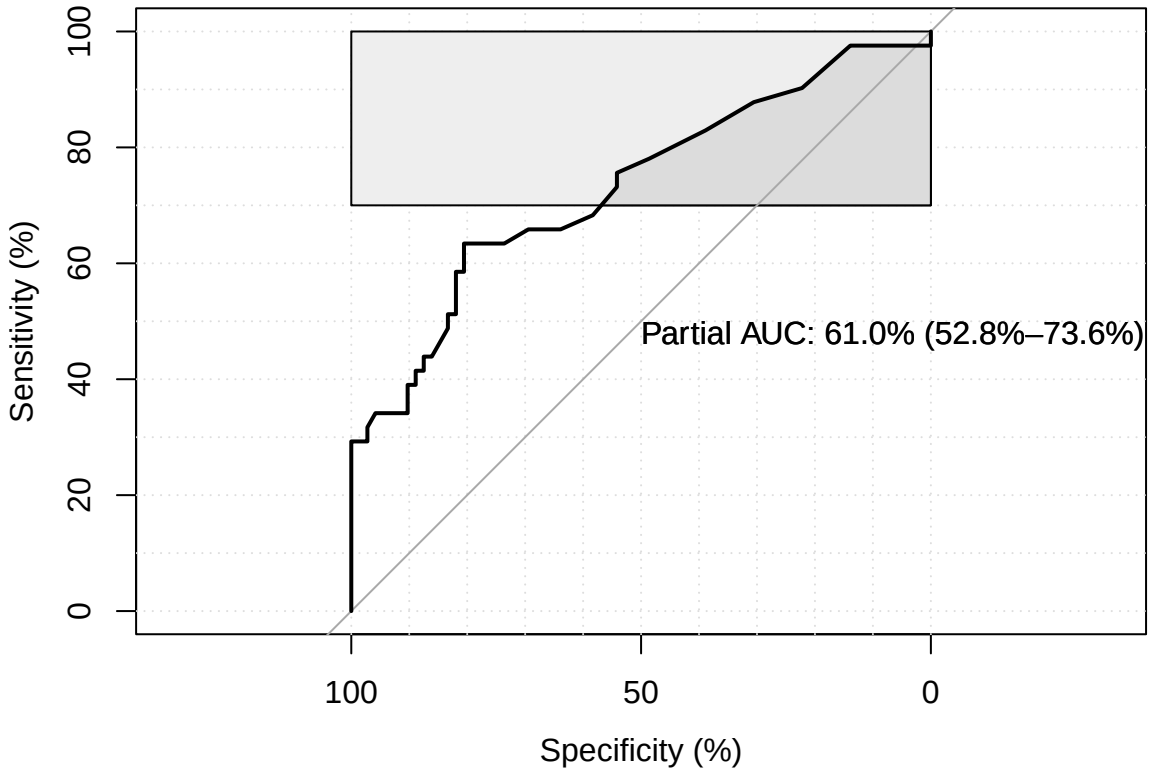
#or below it patients are normal or sick?

ROC curve answer this question, lets see

```
roc1 <- roc(
  rocdata$outcome,
  rocdata$conc,
  percent = TRUE,
  partial.auc = c(100, 70),
  partial.auc.correct = TRUE,
  partial.auc.focus = "sens", # try change to spec
  ci = TRUE,
  boot.n = 1000,
  ci.alpha = 0.05, #95% CI
  stratified = TRUE,
  plot = TRUE,
  auc.polygon = TRUE,
  max.auc.polygon = TRUE,
  grid = TRUE,
  print.auc = TRUE,
  show.thres = TRUE
)
```

```
## Setting levels: control = Good, case = Poor
```

```
## Setting direction: controls < cases
```



Tutors note

Lets understand the code

rocdata*outcome* ∗ ∗*Binaryresponsevariable*(*e.g.*, 0/1*or**TRUE*/*FALSE*). ∗ ∗*rocdata***conc** Continuous predictor variable (e.g., biomarker or model probabilities) **percent = TRUE** Report AUC and CI as percentages (0-100) instead of 0-1. **partial.auc = c(100, 70)** Focus on partial AUC between 70-100% sensitivity **partial.auc.correct = TRUE** adjust partial AUC **ci = TRUE** Calculate bootstrap confidence intervals for the AUC. **boot.n = 1000** number of bootstrap replicates **stratified = FALSE** Use non-stratified bootstrap resampling (may affect imbalanced data). **plot = TRUE** to draw the plot **auc.polygon = TRUE** Shade the area under the ROC curve (AUC) **print.auc = TRUE** Display the AUC value on the plot

##points to consider in the code settings

1. Increase to boot.n = 1000 or higher for stable confidence intervals. (increase the bootstrap iteration)
2. Partial AUC Interpretation: Ensure the range partial.auc = c(100, 70) aligns with your study goals (here, prioritizing high sensitivity).
3. Stratified Bootstrap: if your data is imbalanced (e.g., 90% negative outcomes) set stratified = TRUE for more reliable CI estimates.
4. Confidence Level (ci.alpha = 95%): Use ci.alpha = 0.05 for a standard 95% CI.

Always remember

Sensitivity = True positive rate (TPR) **specificity** = True negative rate (TNR)

>1-Specificity= False positive rate (FPR)

you always seek higher sensitivity and higher specificity (low FPR)

Sensitivity (also called the **true positive rate**, the **recall**, or probability of detection in some fields) measures the proportion of positives that are correctly identified as such (i.e. the percentage of sick people who are correctly identified as having the condition).

Specificity (also called the true negative rate) measures the proportion of negatives that are correctly identified as such (i.e., the percentage of healthy people who are correctly identified as not having the condition).

How to read ROC curve?

1. X-axis:False Positive Rate (1 - Specificity) → How often the model incorrectly predicts "positive." [FP/FP+TN] or sometimes the specificity it self (TNR)
2. Y-axis:True Positive Rate (Sensitivity) → How often the model correctly predicts "positive." [Tp/Tp+FN]
3. Bottom-left (0,0): Model predicts everything as negative
4. Top-right (1,1): Model predicts everything as positive.
5. Top-left (0,1): Perfect Model (100% sensitivity & 100% specificity).
6. Diagonal Line (Random Guessing): A useless model (AUC = 0.5). 7.When AUC is close to 1, the model is perfect. It shows excellent performance with a strong ability to distinguish between classes.
7. AUC around 0.5 is a concerning scenario, as it highlights that the model isn't doing any better than random guessing
8. True positive rate: measuring success
9. False positive rate: measuring the noise; how often our model falsely identifies negatives as positives

Tutor note We need to fine-tune our model's sensitivity (TPR) and specificity (1 - FPR). We can control how sensitive our model is to detecting positive class instances and reduce false positives by tuning the threshold. In most real-world scenarios, the cost of false negatives (missing a positive) versus false positives (false hits) can vary significantly.

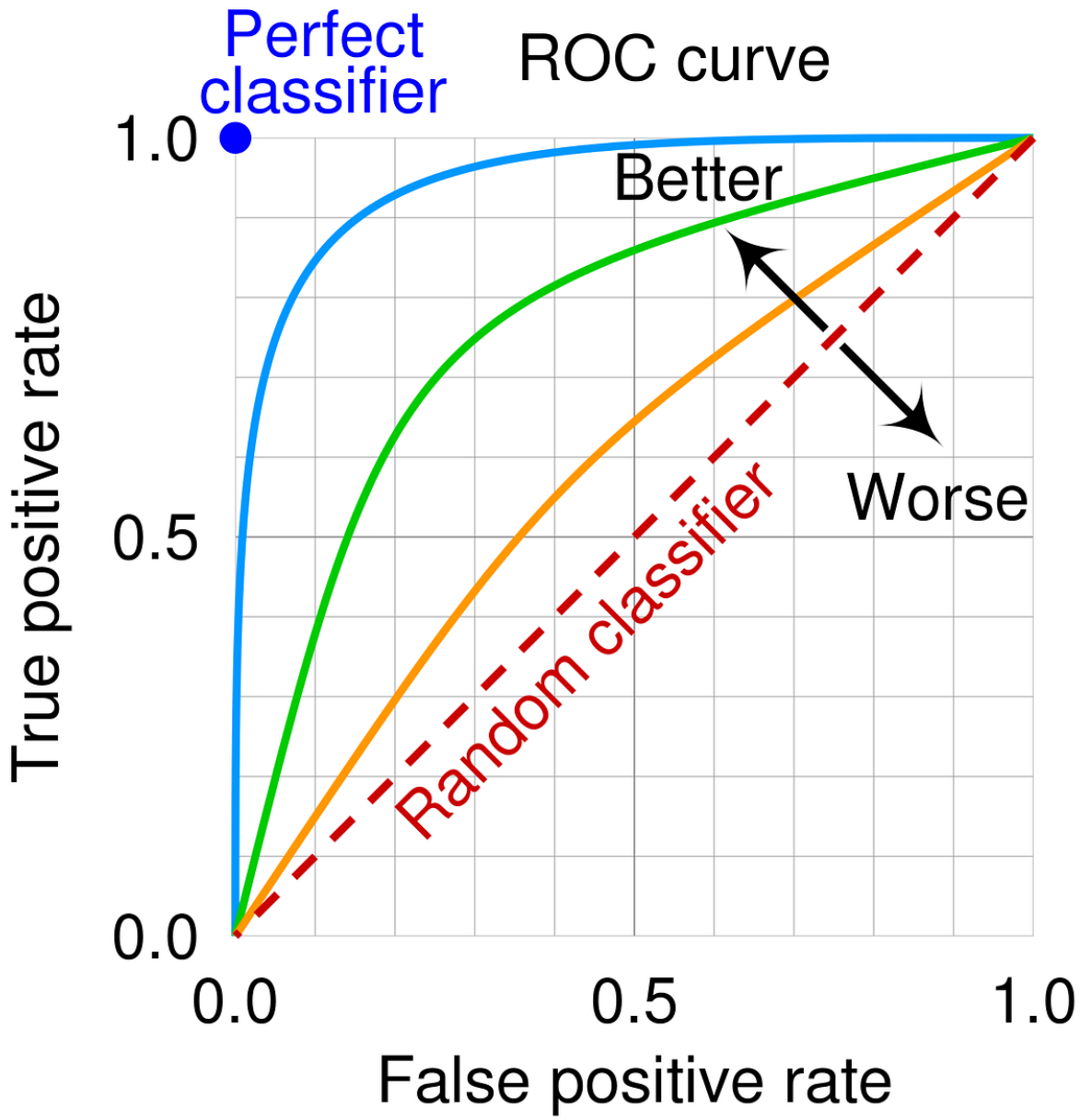
How to interprt the ROC curve? using AUC (area under the curve)- ill talk about it later.

AUC = 1.0 → Perfect classifier AUC > 0.9 → Excellent AUC 0.7 - 0.9 → Good AUC 0.5 - 0.7 → Fair AUC = 0.5 → No better than guessing- bad

generally speaking AUC less than 0.7 is BAD

Lets see examples

```
knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/goodbad.png")
```



choosing the best cut off value "Sensitivity-Specificity Tradeoff"

The **best cut-off** is actually depends on what you want to have. You can either maximize **sensitivity** at the expense of **specificity**.

You may consider choosing a cut point that is most sensitive, but on the expense of specificity (gets each positive cases) but at the same time with much FP.(produces a lot of false-positives).

Likewise, you can increase **specificity** (minimize the axis 1-specificity) and you will get a test that will capture all positive cases but on the expense of sensitivity (false hits will be introduced).

If you want to maximize both, sensitivity and specificity, you can apply the **Youden's index**. For this, you aim to maximize the Youden's index, which is

Maximum =Sensitivity + Specificity - 1

So you choose those value of the ROC-curve as a cut-off, where the term "Sensitivity + Specificity - 1" is met; which is the **maximal point**.

lets calculate the best Coordinates of the curve (based on Youden index)

```
coords(roc1, "best", ret=c("threshold", "sensitivity","specificity", "1-npv"))
```

```
##           threshold sensitivity specificity    1-npv
## threshold     0.115    75.60976    54.16667 20.40816
```

#Code explanation ret: which vector parameter should be retuned Threshold: your optimum cut point sensitivity and specificity: as said before 1-NPV: 1- Negative Predictive Value.

NPV is the probability that subjects with a negative test truly do not have the condition, and is calculated as: [TN/TN+FN]

used to assess the error rate among negative predictions

In fact, there is 2 method to calculate the best cutoff, the Youden and the

closest topleft method.

but what is the differences?

Youden index= sensitivity+specificity-1 Closest topleft method: select threshold that is closest to idea point in the top left corner using distance

Here is the code to define your method. "I recommend Youden"

```
coords(roc1, "best", ret=c("threshold", "sensitivity","specificity", "1-npv"),
       best.method="youden")
```

```
##           threshold sensitivity specificity    1-npv
## threshold     0.115    75.60976    54.16667 20.40816
```

```
coords(roc1, "best", ret=c("threshold", "sensitivity","specificity", "1-npv"),best.method="closest.topl
```

```
##           threshold sensitivity specificity    1-npv
## threshold     0.115    75.60976    54.16667 20.40816
```

"best" argument tells the function to choose an optimal threshold from the ROC curve.

and using coords function , you can retain everything

```
coords(roc1, "best", ret=c("threshold", "specificity", "sensitivity",
                           "accuracy","tn", "tp", "fn", "fp", "npv", "ppv", "1-specificity","1-sensitivi
```

```
##           threshold specificity sensitivity accuracy tn tp fn fp      npv
## threshold     0.115    54.16667    75.60976  61.9469 39 31 10 33 79.59184
##               ppv 1-specificity 1-sensitivity 1-accuracy    1-npv   1-ppv
## threshold 48.4375      45.83333      24.39024    38.0531 20.40816 51.5625
```

The coordinates return one or more of “threshold”, “specificity”, “sensitivity”, “accuracy”, “tn” (true negative count), “tp” (true positive count), “fn” (false negative count), “fp” (false positive count), “npv” (negative predictive value), “ppv” (positive predictive value). “1-specificity”, “1-sensitivity”, “1-accuracy”, “1-npv” and “1-ppv” are recognized as well. Values can be shortenend (for example to “thr”, “sens” and “spec”, or even to “se”, “sp” or “1-np”).

Accuracy is the overall proportion of correct predictions, both true positives and true negatives, over the total number of cases. The formula is (TP + TN) / (TP + TN + FP + FN).

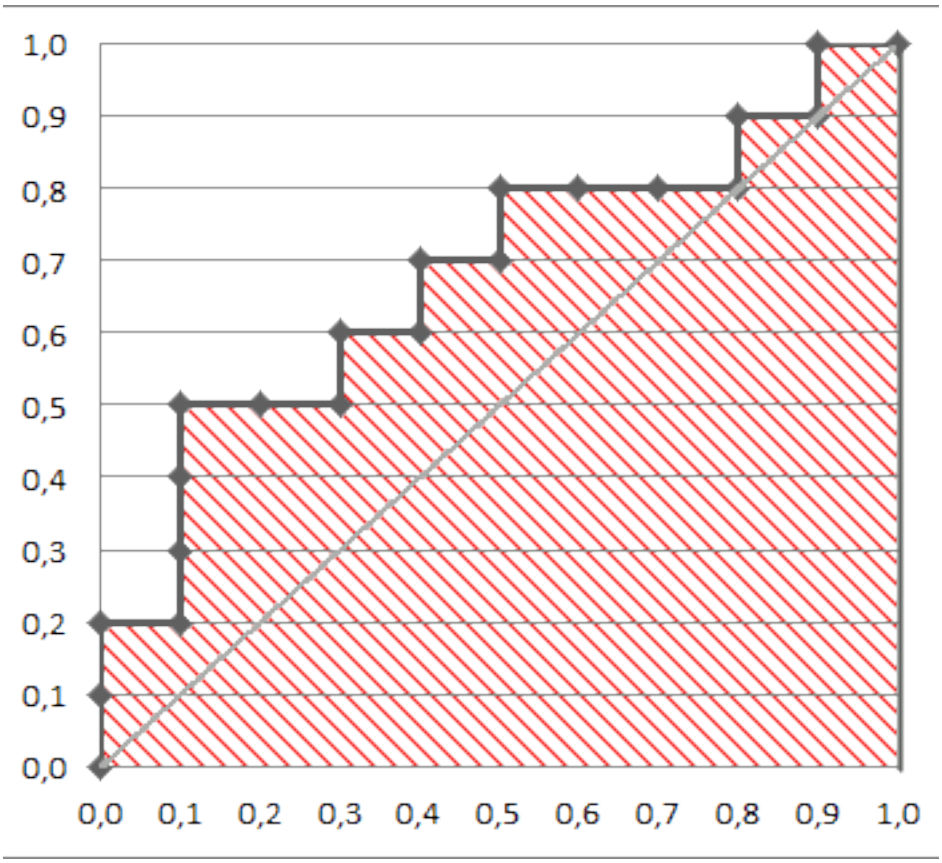
```
coords(roc1, "best", ret=c("threshold", "sens", "spec", "ppv", "npv"))
```

shows the best coordinates within specified sensitivity

```
##           threshold sensitivity specificity     ppv      npv
## threshold     0.115    75.60976    54.16667 48.4375 79.59184
```

###Understanding and calculating Area under the curve (AUC)

```
knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/auc.png")
```



###The area under the ROC Curve (shaded) naturally shows how far the curve from the base line. For the baseline it’s 0.5, and for the perfect classifier it’s 1.

Well to this point, plotting TPR and FPR will generate a curve called the

ROC curve and the area under is the **AUC** or more correctly called **AUROC** stands for "AUROC = Area Under the Receiver Operating Characteristic curve."

> The dashed line in the diagonal we present the ROC curve of a random predictor: it has an AUROC of 0.5. The random predictor is commonly used as a baseline to see whether the model is useful.

lets check the AUC in our model

```
auc(rocdata$outcome, rocdata$conc) # AUC 0.7314
```

```
## Setting levels: control = Good, case = Poor
```

```
## Setting direction: controls < cases
```

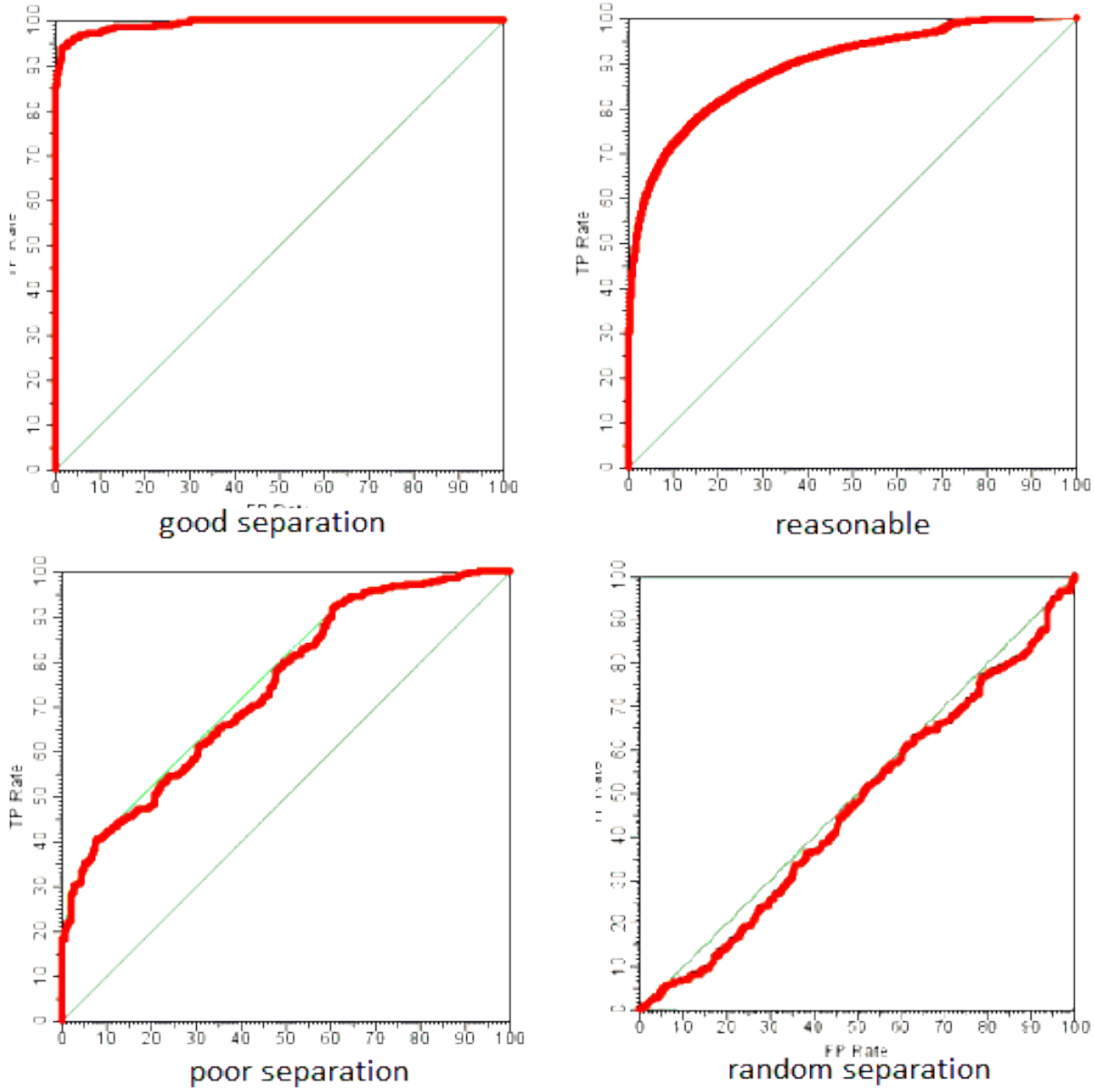
```
## Area under the curve: 0.7314
```

###AUC scale

AUC |	Scoring
0.9-1 |	excellent
0.8-0.9 |	good
0.7-0.8 |	fair
0.6-0.7 |	poor
0.5-0.6 |	fail

AUC explained

```
knitr::include_graphics("C:/Users/magde/Desktop/R2025/8.ROC curve/roc-curves.png")
```



calculating the confidience interval

A confidence interval for the AUC can be computed using methods like:

1. DeLong's test (asymptotic normal approximation).
2. Bootstrapping (resampling the data to estimate variability).

```
ci(roc1) # default is bootstrap method
```

```
## 95% CI: 52.51%-73.92% (2000 stratified bootstrap replicates)
```

```
# ci(roc1, method = "delong")
```

note the error message!

> A confidence interval provides a range of values within which the true AUC is expected to fall, with a given level of confidence (typically 95%).

you may change it

```
ci(roc1, conf.level = 0.99)
```

```
## 99% CI: 49.51%-77.5% (2000 stratified bootstrap replicates)
```

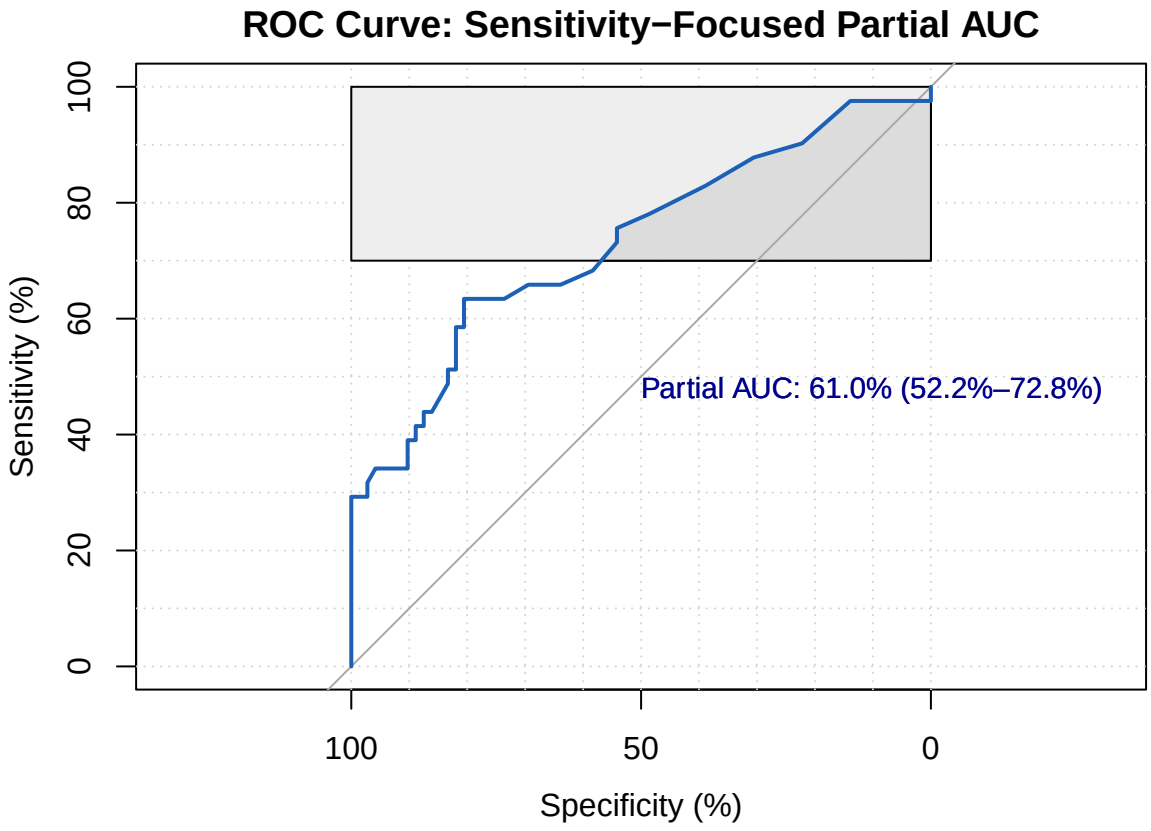
Tutor note 1. Note that DeLong's test can not be computed with partial AUC 2. Quantifying Uncertainty: The CI tells you how precise your AUC estimate is. A narrow interval suggests that the classifier's performance is estimated
nicely, while a wide interval indicates more uncertainty. 3. Model Comparison: When comparing multiple classifiers, overlapped CI means that their performances are not significantly different, and vice versa. 4. Reporting Results: AUC along side with CI gives full picture (professional)

Drawing the CI

```
roc1 <- roc(response = rocdata$
             outcome,predictor = rocdata$conc,
             partial.auc = c(100, 70),
             partial.auc.focus = "sens",
             partial.auc.correct = TRUE,
             percent = TRUE,
            ci = TRUE,
            boot.n = 2000,
            ci.alpha = 0.05,
            stratified = TRUE,
            plot = TRUE,
            auc.polygon = TRUE,
            max.auc.polygon = TRUE,
            grid = TRUE,
            grid.col = "lightgray",
            print.auc = TRUE,
            print.auc.cex = 0.9,
            print.auc.col = "darkblue",
            show.thres = TRUE,          # Show optimal threshold (Youden's index)
            main = "ROC Curve: Sensitivity-Focused Partial AUC",
            col = "#1c61b6"
)
```

```
## Setting levels: control = Good, case = Poor
```

```
## Setting direction: controls < cases
```




```
sens.ci <- ci.se(roc1, specificities=seq(0, 100, 5) )
#Generates a sequence of specificity values from 0% to 100% in 5%
```

sp:This is the specificity value (expressed as a percentage). Specificity refers to the ability of the test to correctly identify negatives

se.low The lower bound of the 95% CI for sensitivity at the corresponding specificity. **se.median** and **se.high** is similar.

This code calculates confidence intervals (CIs) for sensitivity (true positive rate, TPR) at predefined specificity (true negative rate, TNR) values for a previously fitted ROC curve (roc1). It answers the question:

"At specific specificity levels (e.g., 90% specificity), what is the sensitivity of the model, and how uncertain is that estimate?"

Tutors note on pAUC

Focuses on a specific region of the ROC curve & calculates the pAUC under certain condition for senetivity.

Use partial.auc.correct = TRUE (default) to normalize the partial AUC by the maximum possible area in the specified range. This ensures the value lies between 0.5 (random) and 1 (perfect)

Example Use Case

Imagine evaluating the earlier model where FPR < 0.1 is most relevant.

The full AUC might be 0.7314, but if the pAUC (0 to 0.1 FPR) is only 0.61, the model is unreliable at low FPR values.

Why this matters?

1. Clinical/Operational Thresholds: In many applications (e.g., medical testing), you might need to fix specificity at a required level (e.g., 90%) and report sensitivity at that threshold.
2. Model Tuning: Helps identify specificity levels where sensitivity is stable or unstable.
3. Uncertainty Quantification: Confidence intervals show the variability in sensitivity estimates (critical for decision-making).

Generating the curve using ROCR package in 3 separate commands

Lets read the data again

```
head(rocdata)
```

```
##   outcome conc
## 1    Good 0.13
## 2    Good 0.14
## 3    Good 0.10
## 4    Good 0.04
## 5    Poor 0.13
## 6    Poor 0.10
```

###Generating predictions

```
pred <- prediction(rocdata$conc, rocdata$outcome)# note the code arrangement
```

###Generating performance

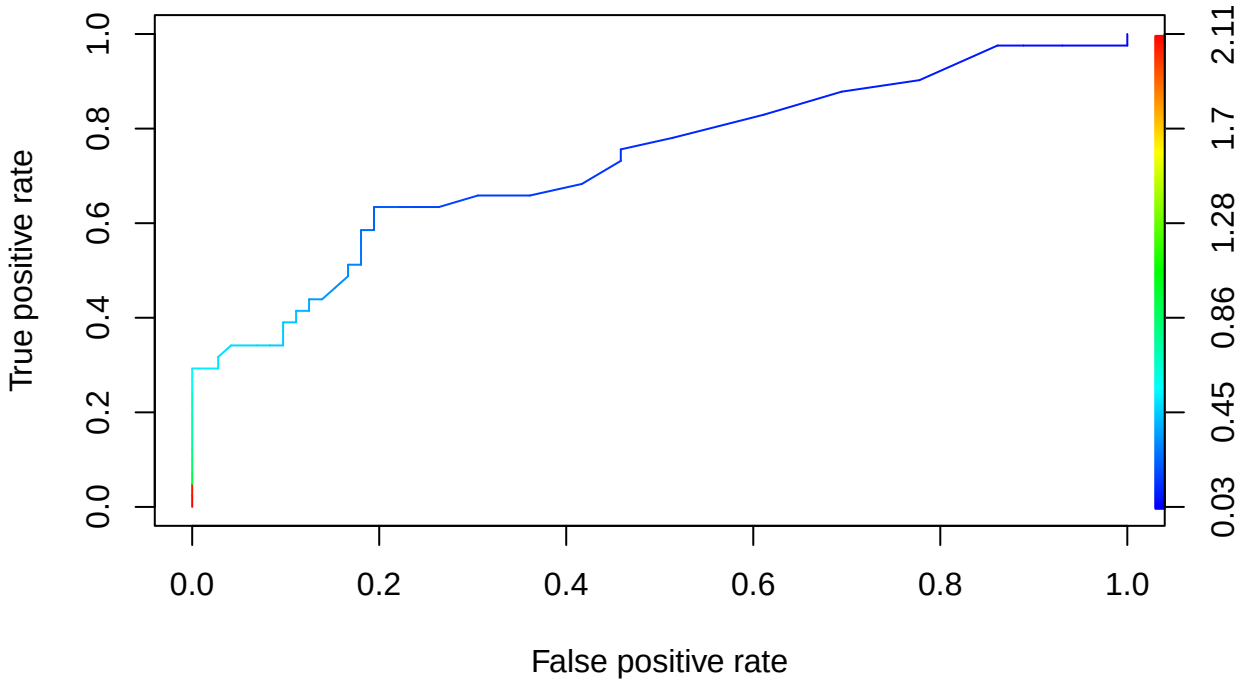
```
perf <- performance(pred,"tpr","fpr")
```

#Extracting the AUC

```
auc <- performance(pred, "auc")
auc_value <- auc@y.values[[1]] # extracting the AUC values
```

plotting the ROC curve

```
plot(perf,colorize=TRUE) # requires also gplots
```



Tutors note

Linking logistic regression with ROC curve (revise the logistic regression session; Example 3)

```
model <- glm(am ~ mpg + wt, data = mtcars, family = binomial)
set.seed(123)
train_index <- createDataPartition(mtcars$am, p = 0.8, list = FALSE)
train_data <- mtcars[train_index, ] # 80%
```

```
train_probs <- predict(model, newdata = train_data, type = "response")
```

Predict probabilities on the test set

```
roc_curve <- roc(train_data$am, train_probs)
```

```
## Setting levels: control = 0, case = 1
```

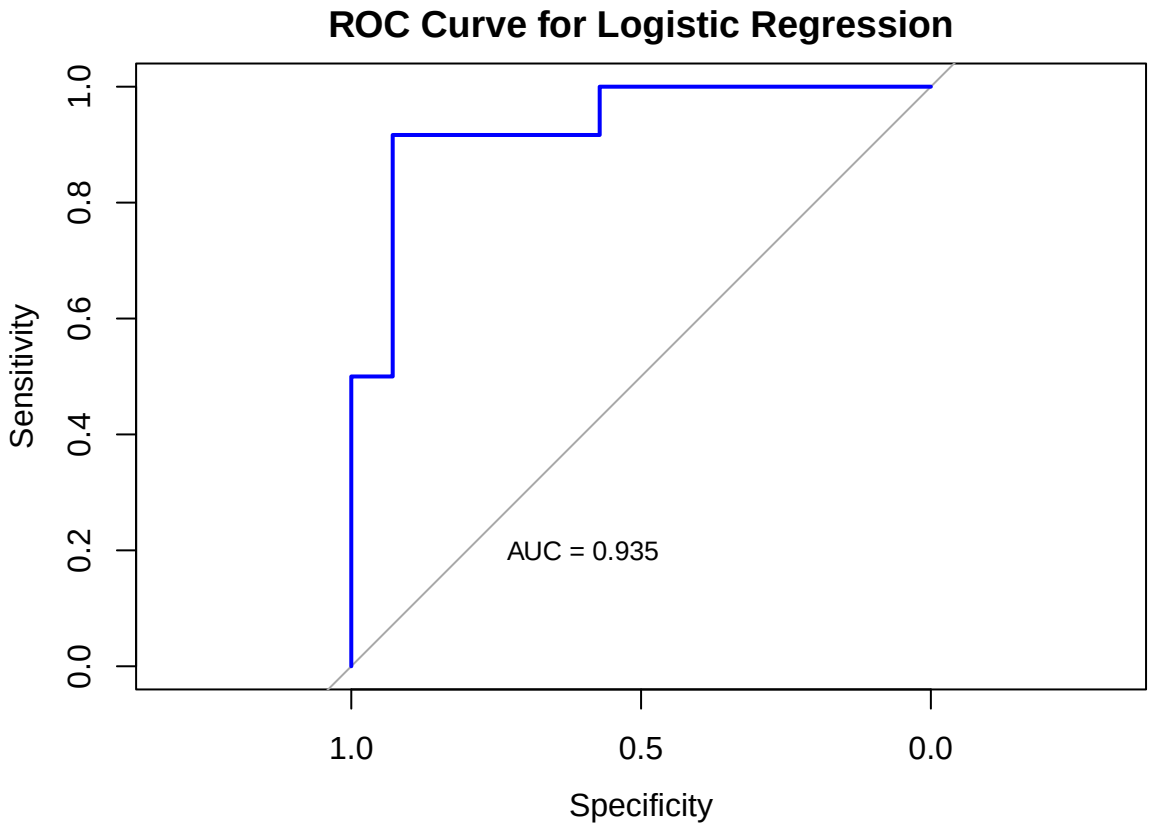
```
## Setting direction: controls < cases
```

extracting AUC

```
auc_value <- auc(roc_curve)
```

plot using basic package add the AUC value

```
plot(roc_curve, col = "blue", lwd = 2, main = "ROC Curve for Logistic Regression")
text(0.6, 0.2, paste("AUC =", round(auc_value, 3)), col = "black", cex = 0.8)
```



Additional hints

In default ROC, the test examin the optimal sensitivity and specificity taking in mind that increased observations towards abnormal. However, in some cases, your data is reversed (negative data is the normal). in that case we need to inform R that the direction is reversed.

In pROC code, please add [direction="<"] when direction is reversed. This will reversed all the positive and negative predictions, which is equivalent to reversing the ROC curve.

##Now you are ready to **ROC and ROLL** !!