**Online Appendix Chapter 22: Evaluating standalone CAD vs. Radiologists**

# Online Appendix 22.A: Random-reader fixed-case analysis

This relates to book section 22.2.1. Following is a listing of file mainAnalysisFixed.R, which implements the analysis reported in the Hupse-Karssemeijer publication1. The file mainAnalysisFixed1.R implements the corresponding RJafroc version.

## Online Appendix 22.A.1: Code listing

# mainAnalysisFixed.R

# case regarded as a fixed effect

rm(list = ls())

library("RJafroc")

library(ggplot2)

require("caTools") #needed for trapezoidal area

source("Wilcoxon.R")

alpha <- 0.05

cat("Hupse-Karssemeijer analysis\n: random readers fixed cases\n")

FOM <- "PCL" # allowed values are "PCL" "ALroc", "AUC"

FPFArr <- c(0.05, 0.2, 0.5, 1) # at which to evaluate PCL

retNico <- DfReadLrocDataFile()

zjk1 <- retNico$NL[1,,,1]

zjk2Cl <- retNico$LLCl[1,,,1]

zjk2Il <- retNico$LLIl[1,,,1]

zjk2 <- pmax(zjk2Cl,zjk2Il)

J <- dim(zjk1)[1] - 1

for (i in 1:length(FPFArr)) {

FPF <- FPFArr[i]

cat("FOM = ", FOM, "\n")

if (FOM == "PCL") cat("FPF = ", FPF, "\n")

thetajc <- array (dim = (J+1))

if (FOM == "AUC") {

for (j in 1:(J+1))

thetajc[j] <- Wilcoxon(zjk1[j,],zjk2[j,])

} else if (FOM == "PCL") {

for (j in 1:(J+1))

thetajc[j] <- (LrocFoms(zjk1[j,], zjk2Cl[j,], FPF))$PCL

} else if (FOM == "ALroc") {

for (j in 1:(J+1))

thetajc[j] <- (LrocFoms(zjk1[j,], zjk2Cl[j,], FPF))$ALroc

} else stop("wrong FOM value")

fom\_diff <- thetajc[-1] - thetajc[1]

ret <- t.test(fom\_diff)

cat("FomCad = ", thetajc[1],"\n")

cat("AVG radiologist performance = ", mean(thetajc[-1]),

"\n95%CI = ", as.numeric(ret$conf.int)+thetajc[1],"\n")

cat("AVG diff. performance = ", mean(thetajc[-1]-thetajc[1]),

"\n95%CI = ", as.numeric(ret$conf.int),"\n")

cat("t-statistic = ", as.numeric(ret$statistic),"\n")

cat("df = ", as.numeric(ret$parameter),"\n")

cat("p-value = ", as.numeric(ret$p.value),"\n\n")

if ((i == 1) && (FOM == "PCL")) {

# last argument is the readers to display, all 9 readers in this case

plots1 <- LrocPlots (zjk1, zjk2Cl, seq(1,9))$lrocPlot

print(plots1)

}

if (FOM == "AUC" ||FOM == "ALroc" ) break

}

Click on Source to obtain the following output and Figure 1.

Browse[2]> source(…)

Hupse-Karssemeijer analysis

: random readers fixed cases

FOM = PCL

FPF = 0.05

FomCad = 0.4625

AVG radiologist performance = 0.5063657

95%CI = 0.4368152 0.5759163

AVG diff. performance = 0.04386574

95%CI = -0.02568478 0.1134163

t-statistic = 1.454404

df = 8

p-value = 0.1839158

FOM = PCL

FPF = 0.2

FomCad = 0.5916667

AVG radiologist performance = 0.7110987

95%CI = 0.6702476 0.7519499

AVG diff. performance = 0.119432

95%CI = 0.07858089 0.1602832

t-statistic = 6.741811

df = 8

p-value = 0.0001462774

FOM = PCL

FPF = 0.5

FomCad = 0.675

AVG radiologist performance = 0.775119

95%CI = 0.7344911 0.815747

AVG diff. performance = 0.100119

95%CI = 0.05949107 0.140747

t-statistic = 5.682659

df = 8

p-value = 0.0004635653

FOM = PCL

FPF = 1

FomCad = 0.675

AVG radiologist performance = 0.7833333

95%CI = 0.7398297 0.826837

AVG diff. performance = 0.1083333

95%CI = 0.06482968 0.151837

t-statistic = 5.742439

df = 8

p-value = 0.0004327327

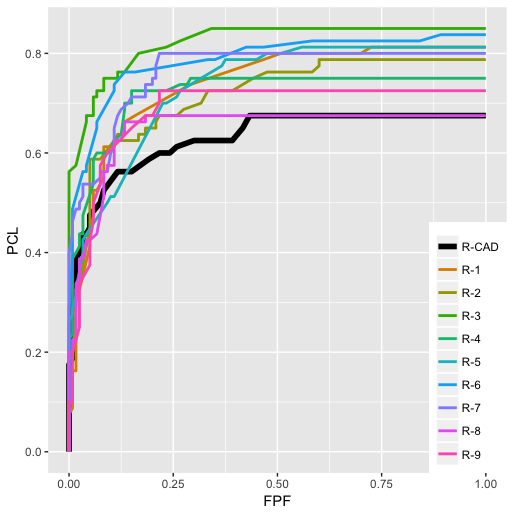


Figure : Plot produced by the code.

Line 11 sets FOM to "PCL", the interpolated  at the value of *FPF* specified at line 21, i.e., FPF = 0.05 on the first iteration of the for-loop starting at line 20, 0.2 on the second iteration, etc. Other allowed choices are "ALroc" and "AUC", for empirical area under the entire LROC and entire area under the ROC. For these choices the for-loop is executed only once (see break statement at line 53).

Line 14 reads the radiologist LROC data using the RJafroc function DfReadLrocDataFile(). By default this function reads the radiologist data only[[1]](#footnote-1). The details of the file read code are there for those who want to learn how to read data from text files. The result is saved to retNico, a list variable, Figure 2. The ratings were obtained on a 0 – 101 scale (the 0-rating was used to designate a "no-mark" image[[2]](#footnote-2)).

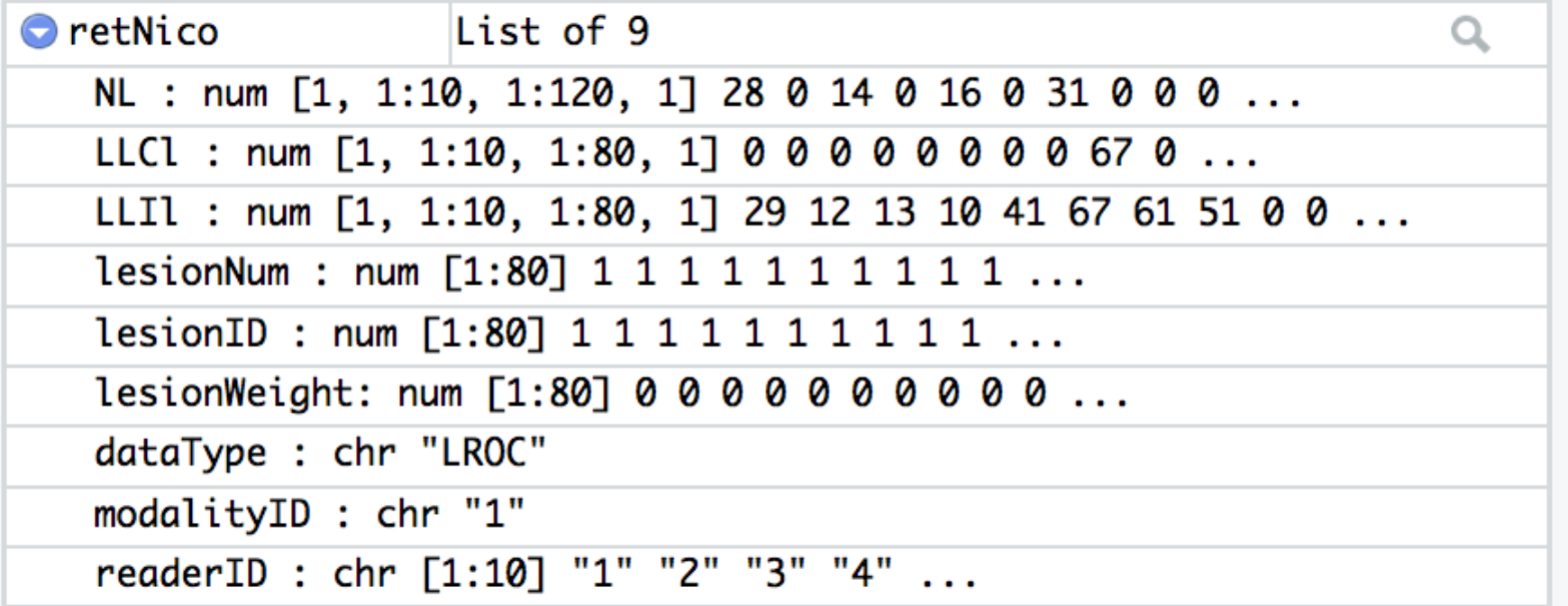


Figure 2: Contents of retNico, a one modality 10 reader LROC dataset, with CAD being the first reader. Because LL events can be of two types - correct localization and incorrect localization – an LROC dataset object has one extra element than other dataset objects. Also, dataType = "LROC".

The dataset actually contains 12-radiologist data, but three of them were residents. Since the residents were ignored that leaves J = 9 radiologists plus the CAD algorithmic "reader", which explains the 1:10 indexing on the ratings arrays retNico$NL, retNico$LLCl and retNico$LLIl. The 1st reader is standalone CAD and the subsequent 9 readers correspond to the radiologists (the reader is encouraged to redo the analysis with the residents included).

Line 15 – 17 extract the ratings arrays. The array named zjk1 stands for , the ROC ratings array for the non-diseased cases. The length of the second dimension of the zjk1 array (K1 = 120) corresponds to the number of non-diseased cases, and the corresponding lengths of the zjk2CL and zjk2IL arrays (K2 = 80) correspond to the number of diseased cases (the presence of two arrays for diseased cases is explained below). To view the ratings for specific radiologists, use the following examples.

## Online Appendix 22.A.2: Code Snippet

> retNico$zjk1[1,1:10]

[1] 28 22 45 48 19 21 27 30 39 21

> retNico$zjk1[2,1:10]

[1] 0 10 12 26 17 26 15 13 25 19

The 2nd line shows the CAD ratings for the 1st 10 non-diseased cases, while the 4th line shows the corresponding ratings for the 1st radiologist. The rating scale was 0-101[[3]](#footnote-3) (use the max() and min() functions to confirm this). Notice that generally the radiologist rated the non-diseased cases lower than did CAD. In fact, on the first non-diseased case the first radiologist found nothing to mark (the 0 rating was assigned to such cases; since only the ordering of the ratings is important, any number below 1 - the lowest explicit rating - would have sufficed). *Lower values on non-diseased and higher values on diseased imply a larger figure of merit, but don’t jump to conclusions based on ratings alone – the only way to really tell is to calculate a figure of merit, such as the Wilcoxon statistic, as shown below*.

The 2nd member of retNico is named zjk2Cl, which is the array containing the ratings of the K2 = 80 diseased cases *on each of which the lesion was correctly localized*, and the 3rd member of retNico is named zjk2Il, which is the array containing the ratings of diseased cases *on each of which the lesion was incorrectly localized*. Line 18 uses the pmax() function, for *parallel* maximum, to extract, for each diseased case, the higher of two ratings, one in the correct localization array and the other in the incorrect localization array corresponding to each diseased case.

In the LROC paradigm, each diseased case receives a single rating, but depending on the proximity of the mark to the true lesion it would be classified either as a correct or as an incorrect localization. By taking the *parallel maximum* we are effectively ignoring how each mark was classified, just retaining the highest ROC rating. In other words, this statement converts the two LROC arrays zjk2Cl and zjk2Il for diseased cases to a single array, zjk2, which contains the ROC ratings of the diseased cases (irrespective of how the mark was classified).

These steps are illustrated below for the 1st ten diseased cases for CAD and the 1st radiologist:

## Online Appendix 22.A.3: Code Snippet

> zjk2Cl[1,1:10]

[1] 0 70 83 38 67 65 59 97 30 60

> zjk2Il[1,1:10]

[1] 29 0 0 0 0 0 0 0 0 0

> zjk2[1,1:10]

[1] 29 70 83 38 67 65 59 97 30 60

> zjk2Cl[2,1:10]

[1] 0 78 93 36 51 61 31 31 21 61

> zjk2Il[2,1:10]

[1] 12 0 0 0 0 0 0 0 0 0

> zjk2[2,1:10]

[1] 12 78 93 36 51 61 31 31 21 61

With FOM (line 11) set to PCL, line 31 calls, for each of 10 readers, the function LrocFoms(),[[4]](#footnote-4) which returns a list variable with members PCL and ALroc, containing the interpolated *PCL* @ the specified *FPF*, and the empirical area under the entire LROC plot, respectively; the $PCL at the end of this line means that only the list member *PCL* @ specified FPF is extracted and saved to thetajc. Since there are 1 + 9 readers, the length of the thetajc array is 10. Highlight thetajc and click Run. The 10 listed values are the empirical FOMs for the different radiologists. The first value refers to CAD and the remaining nine refer to the radiologists. If one seeks to compare CAD to the average of the radiologists, one uses mean(thetajc[-1])- thetajc[1] (the minus one array index denotes the array with the first element, i.e., CAD, deleted, a convenient way of getting the reader array).

Line 37 computes the difference figure of merit, between each of the radiologists and CAD (i.e., each radiologist minus CAD). Line 38 runs the t-test on this array and saves the result to the list variable ret. Line 48 - 52 plots the LROC plots for CAD and each of the 9 radiologists, Figure 1, suitably color-coded. The analysis is repeated for all values of *FPF* in FPFArr, line 12.

Note the subtle difference in the second argument supplied (if FOM = AUC) at lines 28 and 31 to Wilcoxon() and LrocFoms(): the first function needs zjk2, the ROC ratings on diseased cases, while the second function needs zjk2Cl, the correct localization ratings. Line 34 is almost identical to line 31 except that an area measure is being extracted from the returned list.

# Online Appendix 22.B: Random-reader random-case analysis

Following is a listing of file mainAnalysisRandom.R, which implements random-reader random-case analysis[[5]](#footnote-5) and extends mainAnalysisRandomBrief.R to other FOMs[[6]](#footnote-6). The brief version of the code was explained in book Section 22.3.1. The longer version extends it to other FOMs. DiffFomAnal2007Hillis53(), an RJafroc function[[7]](#footnote-7), implements the analysis described in Equations 22.3 through 22.14 and gives due credit to Section 5.3 of a Hillis publication2.

## Online Appendix 22.B.1: Code listing mainAnalysisRandom.R

# MainAnalysisRandom.R # local version of functions; see also CheckInterpolation.xlsx

rm(list = ls())

library(RJafroc);library(ggplot2);library("caTools") #needed for trapezoidal area

source("DiffFomAnal2007Hillis53.R");source("Wilcoxon.R")

source("LrocFoms1.R");source("LrocOperatingPointsFromRatings.R")

cat("Random-reader random-case analysis")

cat("\nof Hupse Karssemeijer radiologist data:\n")

FOM <- "ALROC" # allowed values are "PCL" "ALROC", "Wilcoxon"

FPFArr <- c(0.05, 0.2, 0.5, 1)

retNico <- DfReadLrocDataFile()

zjk1 <- retNico$NL[1,,,1]

zjk2Cl <- retNico$LLCl[1,,,1]

zjk2Il <- retNico$LLIl[1,,,1]

zjk2 <- pmax(zjk2Cl,zjk2Il)

for (i in 1:length(FPFArr)) {

FPF <- FPFArr[i]

cat("FOM = ", FOM, "\n")

if (FOM != "Wilcoxon") cat("FPF = ", FPF, "\n")

if (FOM == "Wilcoxon") {

ret\_nh2 <- DiffFomAnal2007Hillis53 (

zjk1,

zjk2,

FOM)

} else if (FOM == "PCL") {

ret\_nh2 <- DiffFomAnal2007Hillis53 (

zjk1,

zjk2Cl,

FOM,

FPF)

} else if (FOM == "ALROC") {

ret\_nh2 <- DiffFomAnal2007Hillis53 (

zjk1,

zjk2Cl,

FOM,

FPF)

} else stop("wrong FOM value")

print(ret\_nh2)

thetajc <- ret\_nh2$thetajc

psijc <- thetajc[-1] - thetajc[1]

avgRad <- mean(ret\_nh2$thetajc[-1])

sdRad <- sd(psijc)

CIRad <- avgRad + ret\_nh2$CI - ret\_nh2$PsiMean

cat("CIRad = ", CIRad, "\n")

if (FOM == "Wilcoxon") break

}

Line 9 selects the figure of merit, FOM, currently set to "ALROC", , the empirical area under the LROC curve from FPF = 0 to a specified value less than or equal to one. Other allowed choices are "PCL" and "Wilcoxon". Depending on the value of FOM, the if() functions in lines 16 – 31, call the appropriate random-reader random-case analyses. The reader should examine these lines carefully to be convinced that, depending on the choice of FOM, the correct ratings arrays are being passed (insert a break and source the code and click Next, etc.)

## Online Appendix 22.B.2: Code output for mainAnalysisRandom1.R

> source(...)

Random-reader random-case analysis

of Hupse Karssemeijer radiologist data:

FOM = PCL

FPF = 0.05

$Var

[1] 0.01757411

$Cov2

[1] 0.004419852

$PsiMean

[1] 0.04386574

$CI

[1] -0.09985196 0.18758344

$reject

[1] 0

$ddfH

[1] 274.6022

$Tstat

[1] 0.6008715

$p\_val

[1] 0.5484212

$thetajc

[1] 0.4625000 0.4250000 0.5156250 0.6750000 0.5291667 0.4479167 0.5979167 0.5375000 0.4250000 0.4041667

FOM Cad = 0.4625

FOM Rad = 0.5063657

CIRad = 0.362648 0.6500834

FOM = PCL

FPF = 0.2

$Var

[1] 0.005617004

$Cov2

[1] 0.003126171

$PsiMean

[1] 0.119432

$CI

[1] 0.004332322 0.234531766

$reject

[1] 1

$ddfH

[1] 961.2334

$Tstat

[1] 2.0363

$p\_val

[1] 0.04199388

$thetajc

[1] 0.5916667 0.6945313 0.6520833 0.8062500 0.7250000 0.6598214 0.7684524 0.7437500 0.6750000 0.6750000

FOM Cad = 0.5916667

FOM Rad = 0.7110987

CIRad = 0.595999 0.8261984

FOM = PCL

FPF = 0.5

$Var

[1] 0.003787245

$Cov2

[1] 0.002480112

$PsiMean

[1] 0.100119

$CI

[1] -0.00361093 0.20384903

$reject

[1] 0

$ddfH

[1] 646.546

$Tstat

[1] 1.895284

$p\_val

[1] 0.05850058

$thetajc

[1] 0.6750000 0.7975000 0.7625000 0.8500000 0.7500000 0.8000000 0.8160714 0.8000000 0.6750000 0.7250000

FOM Cad = 0.675

FOM Rad = 0.775119

CIRad = 0.6713891 0.878849

FOM = PCL

FPF = 1

$Var

[1] 0.003640309

$Cov2

[1] 0.002436726

$PsiMean

[1] 0.1083333

$CI

[1] 0.004503257 0.212163410

$reject

[1] 1

$ddfH

[1] 492.5538

$Tstat

[1] 2.050008

$p\_val

[1] 0.04089256

$thetajc

[1] 0.6750 0.8125 0.7875 0.8500 0.7500 0.8125 0.8375 0.8000 0.6750 0.7250

FOM Cad = 0.675

FOM Rad = 0.7833333

CIRad = 0.6795033 0.8871634

As FPF increases, the p-value decreases. As noted in the book, there is misunderstanding within the CAD community that the clinically relevant part of the operating characteristic is that near the origin. Besides being incorrect, measuring performance at small FPF (point measure or area measure) sacrifices statistical power.

## Online Appendix 22.B.3: Using Area under the LROC as the figure of merit

The previous example used  as the FOM. A similar conclusion, namely loss of statistical power, is reached using  as the FOM. Change line 9 so that FOM is initialized to "ALROC". Source the code. The output is summarized in Table 1 (the rows labeled *fixed* were obtained by sourcing mainAnalysisFixed.R with a similar change in line 11 of that file).

Table 1: Comparison of fixed case and random case analyses. The figure of merit = ALROC @ specified FPF. Rows labeled "fixed" were obtained by sourcing mainAnalysisFixed.R, while rows labeled "random" were obtained by sourcing mainAnalysisRandom.R. Because it is accounting for an additional source of variability, each of the lines labeled *random* yields a larger p-value and a wider confidence interval than the corresponding lines labeled *fixed*.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| FPF | Cases | CAD | <RAD> |  |  |  | p-value |
| 0.05 | fixed | 0.018 | 0.020 | (0.016, 0.024) | 0.00165 | (-0.0028, 0.0061) | 0.41 |
| random | (0.012, 0.028) | (-0.0065, 0.0098) | 0.69 |
| 0.2 | fixed | 0.101 | 0.116 | (0.105, 0.128) | 0.016 | (0.0044, 0.027) | 0.0128 |
| random | (0.093, 0.14) | (-0.0077, 0.04) | 0.186 |
| 0.5 | fixed | 0.291 | 0.343 | (0.321, 0.365) | 0.0524 | (0.031, 0.074) | 0.00054 |
| random | (0.292, 0.394) | (0.0016, 0.103) | 0.043 |
| 1 | fixed | 0.628 | 0.734 | (0.693, 0.774) | 0.105 | (0.065, 0.146) | 0.00031 |
| random | (0.636, 0.831) | (0.0075, 0.203) | 0.0349 |

## Online Appendix 22.B.4: Other instructive functions

The following functions were used in the previous code demonstrations. It is useful to inspect them.

LrocOperatingPointsFromRatings.R

This calculates LROC operating points, i.e., (FPF, PCL) from the raw ratings data.

# NOTE: two versions yield identical results

# older version

LrocOperatingPointsFromRatings1 <- function( zk1, zk2Cl )

{

FPF <- 1

PCL <- NULL

zk11 <- zk1;zk2Cl1 <- zk2Cl

while(1) {

cutoff <- min( c( zk11, zk2Cl1 ) )

zk11 <- zk1[ zk1 > cutoff ]

zk2Cl1 <- zk2Cl[ zk2Cl > cutoff ]

FPF1 <- length( zk11 ) / length( zk1 )

PCL1 <- length( zk2Cl1 ) / length( zk2Cl )

FPF <- c( FPF, FPF1 )

if (length(PCL) == 0) PCL <- c(PCL1, PCL1) else PCL <- c( PCL, PCL1 )

if( FPF1 == 0 && PCL1 == 0 ) {

break

}

}

return( list(

FPF = FPF[length(FPF):1],

PCL = PCL[length(PCL):1]

) )

}

# cleaner version

LrocOperatingPointsFromRatings <- function( zk1, zk2Cl )

{

bins <- sort(unique(c(zk1,zk2Cl)))

nBins <- length(bins)

fpCounts <- array(0, dim = nBins)

clCounts <- array(0, dim = nBins)

for (b in 1:nBins){

fpCounts[b] <- sum(zk1 == bins[b])

clCounts[b] <- sum(zk2Cl == bins[b])

}

FPF <- cumsum(rev(fpCounts)) / length(zk1)

PCL <- cumsum(rev(clCounts)) / length(zk2Cl)

FPF <- FPF[-length(FPF)]

PCL <- PCL[-length(PCL)]

FPF <- c(0, FPF, 1) # add origin and largest value

PCL <- c(0, PCL, PCL[length(PCL)]) # add origin and max PCL value

return( list(

FPF = FPF,

PCL = PCL

) )

}

The two functions in the above code are equivalent but the second form is more transparent. To debug/understand the function ensure that line 11 in mainAnalysisFixed.R initializes FOM to "ALROC". Insert a break point at line 33 and click Source, Figure 3.

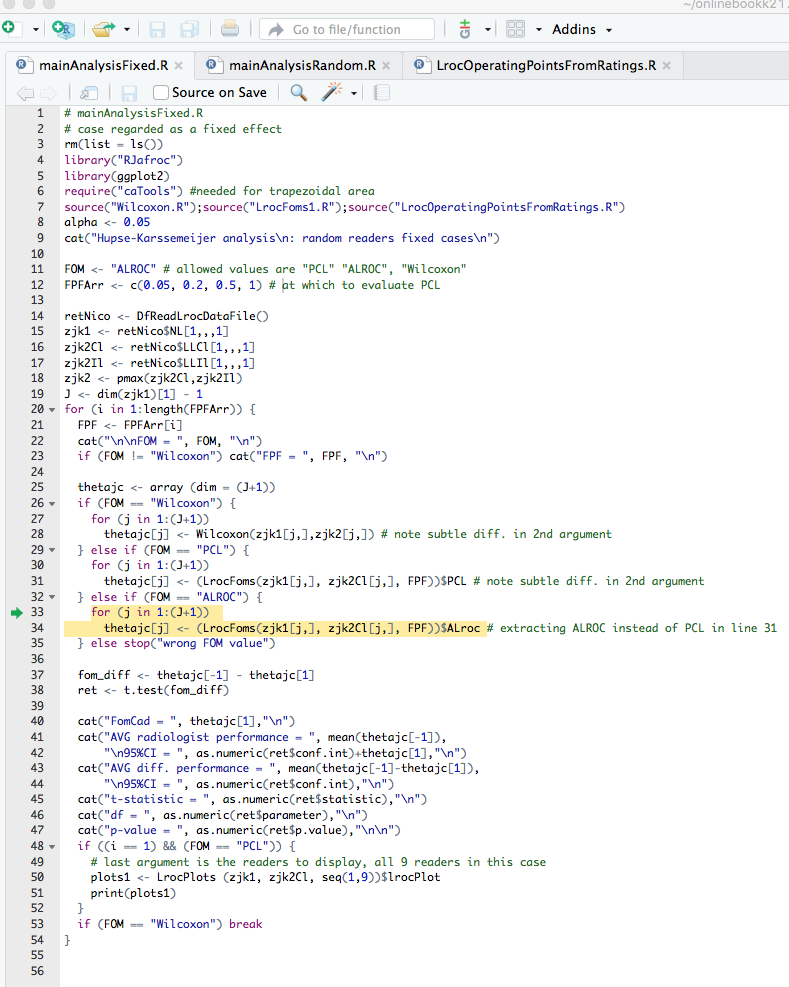


Figure : Screen shot. Note that code pointer is at breakpoint previously inserted at line 33.

Exit debug mode (click Stop). Insert a break point at line 30 of LrocOperatingPointsFromRatings.R and click Source. This compiles the file in debug mode; see code output below and Figure 4:

> debugSource('~/onlinebookk21778/Ch22/software/LrocOperatingPointsFromRatings.R')

>

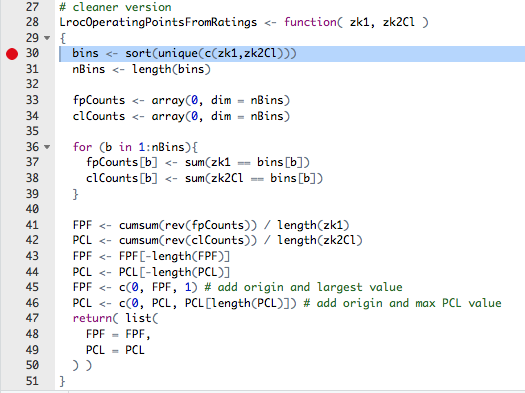


Figure : Break point inserted at beginning of function.

Position the cursor at line 33 in mainAnalysisFixed.R and click Run twice. The code pointer should be at line 3 of LrocOperatingPointsFromRatings.R, Figure 5.

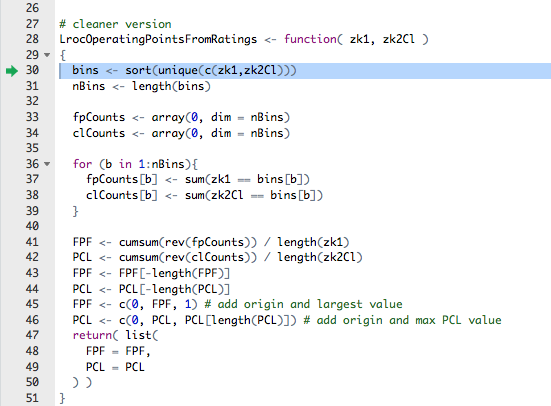


Figure : Code pointer is at break point in function being debugged (or understood).

Use standard debug methods from here on to see what is going on in the function[[8]](#footnote-8).

LrocFoms1.R

This calculates  from the raw ratings data.

LrocFoms <- function (zk1, zk2Cl, FPFValue) {

lroc <- LrocOperatingPointsFromRatings( zk1, zk2Cl )

PCL <- (approx(lroc$FPF, lroc$PCL, xout = FPFValue))$y

tempFpf <-c(lroc$FPF[lroc$FPF < FPFValue],FPFValue)

tempPcl <-c(lroc$PCL[lroc$FPF < FPFValue],PCL)

ALroc <- trapz(tempFpf, tempPcl)

return (list (

PCL = PCL,

ALroc = ALroc

))

}

A numerical check of the R interpolation function approx() for FPF = 0.05 is in CheckInterpolation.xlsx.

# References

1. Hupse R, Samulski M, Lobbes M, et al. Standalone computer-aided detection compared to radiologists’ performance for the detection of mammographic masses. *Eur Radiol.* 2013;23(1):93-100.

2. Hillis SL. A comparison of denominator degrees of freedom methods for multiple observer ROC studies. *Statistics in Medicine.* 2007;26:596-619.

1. By default, the resident data in the raw data files are ignored; the raw data files, jaf\_truth.txt and findings.txt, were provided by Prof. Nico Karssemeijer, the senior author on the publication describing the study. [↑](#footnote-ref-1)
2. This is an issue with the LROC paradigm, which requires a forced localization even when the radiologist sees nothing to report; in a later version of the LROC MLE fitting program Prof. Swensson allowed the reader to *not report a location when the confidence was below a specified value*, e.g., 15 on a 0 to 100 scale (the "15" is a source of arbitrariness) and the program internally drew a random number to determine if the rating was to be classified as a correct or incorrect localization. The FROC paradigm does not require a forced localization, so this type of arbitrariness is not present. [↑](#footnote-ref-2)
3. The zero rating comes from unmarked cases; to avoid clashes with actual (i.e., assigned) zero ratings, all actual ratings were incremented by unity. This is why the ratings extend from 0 to 101. [↑](#footnote-ref-3)
4. This function is implemented in file gpfMyFOM.R, which is in the RJafroc directory. [↑](#footnote-ref-4)
5. File mainAnalysisRandom1.R uses the RJafroc functions, hiding details of the analysis from the user. [↑](#footnote-ref-5)
6. Note that this code is inconsequentially different from that in Section 22.3.1 in the book. In the online appendix we are using the local version of the function in DiffFomAnal2007Hillis53.R, which overrides the RJafroc version; the latter continues to "evolve" unfettered by what is in the book. As an example, a significant change is that the RJafroc version takes ROC, FROC or LROC dataset objects, and not ratings, as arguments. An example using the RJafroc version is in mainAnalysisRandomBrief2.R. [↑](#footnote-ref-6)
7. Notice that the local version is *not* sourced, so it is invisible to the code. [↑](#footnote-ref-7)
8. Sourcing mainAnalysisFixed.R compiles all related files in *standard* mode, not *debug* mode; hence the need to avoid sourcing it *after* a breakpoint has been set inside a function and the function has been sourced (the presence of the break point ensures that it is compiled in debug mode). Try sourcing mainAnalysisFixed.R after a function has been compiled in debug mode: the code pointer will not stop at the desired location. [↑](#footnote-ref-8)