Chapter 10: ORH Analysis

Online Supplementary Material

A. Online Appendix 10.A: The DeLong method for estimating the covariance matrix

B. Online Appendix 10.B: Estimation of covariance matrix: single-reader multiple-treatment

C. Online Appendix 10.C: Comparing DBMH and ORH methods for single-reader multiple-treatment

D. Online Appendix 10.D: Minimal implementation of ORH method

E. Online Appendix 10.E: Proof of Eqn. 10.64.

F. Online Appendix 10.F: Single-treatment multiple-reader analysis

Online Appendix 10.A: The DeLong method for estimating the covariance matrix

In 1988 DeLong et al described a non-parametric method[1] for estimating the covariance between two empirical AUCs that are based on the same cases (earlier, in Section 7.4, a simpler result from that paper was used, namely estimating the variance of an empirical AUC). Define *structural components*:

$$V_{01}^{ij}\left(z_{ijk_1 1}\right)=\frac{1}{K_2}\sum_{k_2=1}^{K_2}\psi\left(z_{ijk_1 1},z_{ijk_2 2}\right) \qquad k_1=1,2,...,K_1$$
$$V_{10}^{ij}\left(z_{ijk_2 2}\right)=\frac{1}{K_1}\sum_{k_1=1}^{K_1}\psi\left(z_{ijk_1 1},z_{ijk_2 2}\right) \qquad k_2=1,2,...,K_2 \quad . \quad \textbf{(10.91.1)}$$

The kernel function $\psi\left(x,y\right)$ is defined by:

$$\psi\left(x,y\right)=1 \qquad if \qquad x<y$$
$$\psi\left(x,y\right)=1/2 \qquad if \qquad x=y \quad . \quad \textbf{(10.91.2)}$$
$$\psi\left(x,y\right)=0 \qquad if \qquad x>y$$

Define the *IJ* x *IJ* matrix S_{01} such that its $\left(\ ij,i'j'\right)$ element is given by

$$s_{01}^{ij,i'j'}=\frac{1}{K_1-1}\sum_{k_1=1}^{K_1}\left[V_{01}^{ij}\left(z_{ijk_1 1}\right)-\widehat{\theta_{ij}}\right]\left[V_{01}^{i'j'}\left(z_{ijk_1 1}\right)-\widehat{\theta_{i'j'}}\right] \quad . \quad \textbf{(10.91.3)}$$

Similarly the *IJ* x *IJ* matrix S_{10} is defined such that its $\left(\ ij,i'j'\right)$ element is given by

$$s_{10}^{ij,i'j'}=\frac{1}{K_2-1}\sum_{k_2=1}^{K_2}\left[V_{10}^{ij}\left(z_{ijk_2 2}\right)-\widehat{\theta_{ij}}\right]\left[V_{10}^{i'j'}\left(z_{ijk_2 2}\right)-\widehat{\theta_{i'j'}}\right] \quad . \quad \textbf{(10.91.4)}$$

In matrix notation, the covariance matrix S between θ_{ij} and $\theta_{i'j'}$ can be estimated using:

$$S = \frac{1}{K_1} S_{01} + \frac{1}{K_2} S_{10} \quad . \quad \textbf{(10.91.5)}$$

Specifically,

$$Cov\left(\theta_{ij}, \theta_{i'j'}\right) = \frac{1}{K_1} s_{01}^{ij,i'j'} + \frac{1}{K_2} s_{10}^{ij,i'j'} \quad . \quad \textbf{(10.91.6)}$$

The **R** implementation of these formulae is in file **`VarCovMtrxDLStr.R`**.

For completeness, and for possible research use, the implementation of Eqn. 4 ibid, *under the assumption that one can replace expected values with observed values*, is included in file **`VarCovMtrxDLEqn4.R`**. For large numbers of cases this gives almost identical results to those given by the other "standard" methods. According to Prof. Elizabeth DeLong (private communication, ca. 2010) one should use the structural components method, which is expected to be almost equivalent to the jackknife, as observed in the above code output.

Online Appendix 10.B: Estimation of covariance matrix: single-reader multiple-treatment

The following code computes the correlations for a single reader interpreting a common case-set in multiple treatments. The jackknife and bootstrap methods of computing Var and Cov_1 are implemented in **`mainVarCov1.R`**, a listing of which follows. The DeLong method is implemented in function **`VarCovMtrxDLStr()`**, which is included in a **`.R`** file with the same name.

Online Appendix 10.B.1: Code Listing

```
rm(list = ls()) #mainVarCov1.R
library(RJafroc)
source("VarCov1Bs.R")
source("Wilcoxon.R")
source("VarCov1Bs.R")
source("VarCov1Jk.R"); source("VarCovs.R")
source("VarCovMtrxDLStr.R")
seed <- 1;set.seed(seed)
ROC <- FALSE
if (ROC) { # this block has ROC datasets
  #fileName <- "Franken1.lrc"
  fileName <- "VanDyke.lrc"
  rocData <- DfReadDataFile(fileName, format = "MRMC", renumber = "TRUE")
} else {  # this block has FROC datasets
  fileName <- "CXRinvisible3-20mm.xlsx"
  frocData <- DfReadDataFile(fileName, format = "JAFROC", renumber = "TRUE")
  rocData <- DfFRoc2HrRoc(frocData)
}
jSelect <- 1  # selects the reader to be analyzed
rocData1R <- DfExtractDataset(rocData, rdrs = jSelect)

zik1 <- rocData1R$NL[,1,,1];K <- dim(zik1)[2];I <- dim(zik1)[1]
zik2 <- rocData1R$LL[,1,,1];K2 <- dim(zik2)[2];K1 <- K-K2;zik1 <- zik1[,1:K1]

FOM <- array(dim=c(I))
for (i in 1:I) {
  FOM[i] <- Wilcoxon(zik1[i,],zik2[i,])
}

FOMik <- array(dim = c(I,  K))
for (i in 1:I) {
  for (k in 1:K) {
    if (k <= K1) {
      FOMik[i,k] <- Wilcoxon(zik1[i,-k], zik2[i,])
    } else {
      FOMik[i,k] <- Wilcoxon(zik1[i,], zik2[i,-(k-K1)])
    }
  }
}
```

```
ret1 <- VarCov1Jk(FOMik)

cat("data file = ", fileName, "\n")
cat("number of treatments = ", I, ", number of non-diseased cases = ", K1,
    ", number of diseased cases = ", K2, "\n")
cat("reader = ", jSelect, "\n")
cat("OR variance components using jackknife resampling", "\n")
cat("Variance = ",  ret1$Var, ", Cov1 = ",  ret1$Cov1, ", rho = ",  ret1$Cov1/ret1$Var, "\n")

B <- 2000;aucBs <- array(dim = c(I,B))#to save the bs Auc values
for (b in 1 : B){
  k1b <- ceiling( runif(K1) * K1 ) # bs indices for non-diseased
  k2b <- ceiling( runif(K2) * K2 ) # bs indices for diseased
  for ( i in 1 : I) aucBs[i,b] <- Wilcoxon(zik1[i,k1b], zik2[i,k2b])
}

ret2 <- VarCov1Bs(aucBs)
cat("OR variance components using bootstrap resampling", "\n")
cat("Variance = ",  ret2$Var, ", Cov1 = ",  ret2$Cov1, ", rho = ",  ret2$Cov1/ret2$Var, "\n")

mtrxDLStr <- VarCovMtrxDLStr(rocData1R)
VarCovDLStr <- VarCovs(mtrxDLStr)

cat("OR variance components using DeLong method", "\n")
cat("Variance = ",  VarCovDLStr$var, ", Cov1 = ",  VarCovDLStr$cov1, ", rho = ",
VarCovDLStr$cov1/VarCovDLStr$var, "\n")
```

Line 9 sets the **ROC** flag variable to **FALSE**, causing the FROC data file **CXRinvisible3-20mm.xlsx** to be loaded at line 16 and line 17 converts it to an ROC dataset object using the highest rating on each case as its inferred ROC rating. Line 19 selects the reader to be analyzed, **jSelect**, currently set to reader 1, and the next line extracts the ratings for this reader in all treatments and saves it to a dataset named **rocData1R**. Insert a breakpoint (red dot) to the left of line number 25 (in the grey area of the window) and click **Source**. Look at the **Environment** window to confirm that **zik1**, representing $z_{ik_1 1}$, is an [1:4,1:52] array and **zik2**, representing $z_{ik_2 2}$, is an [1:4,1:106] array. Also confirm from the **Environment** panel that **rocData** contains 5 readers (see length of 2nd index of **NL** or **LL** lists) while **rocData1R** contains a single reader. Each z-sample is indexed by $ik_t t$ where $i = 1,2,...,I$ is the treatment index with $I = 4$, and $t = 1,2$ is the truth index and $k_t = 1,2,...,K_t$ indexes cases in truth state *t*. The next 4 lines initialize the **FOM[1:4]** array using the **Wilcoxon()** function, i.e., empirical ROC- AUCs. Click on **Next** twice to enter the for-loop; to get out of the for-loop one could keep clicking on **Next** or click the green "*step out of function*" arrow; hovering over any icon reveals what it will do if clicked, in this case the hover message is "*Execute the remainder of the current function or loop*" and also shows a keyboard shortcut. Either way, the cursor advances to line 30.

Highlight **FOM** and click on **Run**. This yields the following values: 0.5696662, 0.5296626, 0.6358853 and 0.6249093. The first two values are close to chance-level performance as the nodules were invisible on 2-view chest x-rays (CXR) and were only included because they were visible on CT, which served as the gold standard. The treatments are, in order, 1:CXR, 2:CXR+DE, 3:TOMO and 4:TOMO+DE, where DE stands for dual energy and TOMO stands for chest-tomosynthesis[2].

Lines 30-40 calculate the *jackknife FOM* values $\theta_{i(k)}$, not to be confused with *jackknife pseudovalues*, see paragraph before Eqn. **Error! Reference source not found.**. Line 34 does this for non-diseased cases and line 36 for diseased cases. Notice that by allowing a negative array index **R** makes it particularly easy to remove a case. Click on **Next** and step out of the **for**-loops to get to line 40. Examine the structure of **FOMik**; it is a numeric array [1:4, 1:158]. Line 40 calculates Var and Cov_1 using the jackknife and save the results to **ret1**. The explanation of this code, contained in file **VarCov1Jk.R**, is in Appendix 10.B.5. For now simply execute it by clicking on **Next**. Examine the structure of **ret1** and the ratio shown below.

Online Appendix 10.B.2: Code snippet

```
Browse[2]> str(ret1)
List of 2
```

```
 $ Var : num 0.00161
 $ Cov1: num 0.000497
Browse[2]> ret1$Cov1/ret1$Var
[1] 0.3078498
```

The jackknife estimates of Var and Cov_1 are 0.00161 and 0.000497, respectively. As expected, covariance is less than variance and their ratio is the correlation $\rho_1 = 0.308$. Lines 42 – 47 print out these values in a relatively "user-friendly" way. Keep clicking on **Next** until the cursor has advanced to line 49.

Line 49 sets the number of bootstraps **B** to 2000 and allocates memory for the bootstrap figure-of-merit values. The bootstrap method was explained in Section 7.5. Click on **Next** and step out of the for-loop to get to line 56. Execute this line (click **Next**), which calculates the bootstrap estimates of Var and Cov_1, using the code in file **VarCov1Bs.R** in Appendix 10.B.6, to get to line 57. Examine the structure of **ret2**.

Online Appendix 10.B.3: Code snippet

```
Browse[2]> str(ret2)
List of 2
 $ Var : num 0.00158
 $ Cov1: num 0.000527
Browse[2]> ret2$Cov1/ret2$Var
[1] 0.3346733
```

The bootstrap estimates of Var and Cov_1 are 0.00158 and 0.000527, respectively. As expected, the covariance is smaller than the variance and their ratio is the correlation $\rho_1 = 0.335$. Lines 57 – 58 print out these values. The last few lines (60 - 64) calculate and print the corresponding quantities using the DeLong method of structural components described in Online Appendix 10.A. Click **Continue** to execute the rest of the code.

Online Appendix 10.B.4: Calculation of Var and Cov1

Listed below is the code for **VarCov1Jk.R** for the function **VarCov1Jk()** which takes one argument, the jackknife FOM matrix **JackFoMMatrix**, with dimension **[1:I,1:K]** and returns **Var** and **Cov1** as a list.

Online Appendix 10.B.5: Code Listing VarCov1Jk

```
VarCov1Jk <- function (JackFoMMatrix)
{
  I <- dim(JackFoMMatrix)[1]
  K <- dim(JackFoMMatrix)[12
  Cov <- array(dim = c(I, I))

  for (i in 1:I){
    for (ip in 1:I){
      Cov[i, ip] <- cov(JackFoMMatrix[i,], JackFoMMatrix[ip,])
    }
  }

  Var <- 0
  count <- 0
  for (i in 1:I){
    Var <- Var + Cov[i, i]
    count <- count + 1
  }
  Var <- Var / count
  Var <- Var * (K-1)^2/K

  Cov1 <- 0
  count <- 0
  for (i in 1:I){
    for (ip in 1:I){
      if (ip != i){
        Cov1 <- Cov1 + Cov[i, ip]
        count <- count + 1
      }
```

```
    }
  }
  Cov1 <- Cov1 / count
  Cov1 <- Cov1 * (K-1)^2/K
  return (list (
    Var = Var,
    Cov1 = Cov1
  ))
}
```

Lines 3 and 4 extract the dimensions of the jackknife FOM matrix. The action takes place in lines 7-11 where the variable **ip** denotes i'. The **cov()** function (supplied by **R**) takes as arguments **JackFoMMatrix [i,]**, which stands for the K-length array $\theta_{i(k)}$ (k = 1, 2, ..., K) and **JackFoMMatrix [ip,]**, which stands for the K-length array $\theta_{i'(k)}$, and calculates the covariance, i.e., it implements the term inside the square brackets on the right hand side of Eqn. **Error! Reference source not found.**. Confirm that the values are exactly those predicted by the term inside the square brackets on the right hand side of Eqn. **Error! Reference source not found.**; this will exercise one's ability to write a simple function implementing the formula (for the faint hearted, there is a function in file **CovarianceFirstPrinciples.R** implementing this equation from scratch). Line 13-19 averages the variances over the different treatments (i.e., it averages over the diagonal terms of the matrix shown in Eqn. **Error! Reference source not found.**), and similarly, line 23-32 averages the co-variances (i.e., it averages over all the off-diagonal terms); after application the variance inflation factor, two numbers **Var** and **Cov1** are returned as a **list** variable.

Listed next is the code for **VarCov1Bs.R** for the function **VarCov1Bs()** which takes one argument, the bootstrap FOM matrix **FomBs**, with dimension **[1:I,1:B]** and returns **Var** and **Cov1** as a list.

Online Appendix 10.B.6: Code Listing VarCov1Bs

```
VarCov1Bs <- function (FomBs)
{
  I <- dim(FomBs)[1]
  Covariance <- array(dim = c(I, I))

  for (i in 1:I){
    for (ip in 1:I){
      Covariance[i, ip] <- cov(FomBs[i, ], FomBs[ip, ])
    }
  }
  #CovarianceFirstPrinciples(FomBs)
  Var <- 0
  count <- 0
  for (i in 1:I){
    Var <- Var + Covariance[i, i]
    count <- count + 1
  }
  Var <- Var / count

  Cov1 <- 0
  count <- 0
  for (i in 1:I){
    for (ip in 1:I){
      if (ip != i){
        Cov1 <- Cov1 + Covariance[i, ip]
        count <- count + 1
      }
    }
  }
  Cov1 <- Cov1 / count

  return (list (
    Var = Var,
    Cov1 = Cov1
  ))
}
```

The action takes place in lines 6-10. The **cov()** function takes as arguments the *B*-length array **FomBs[i,]**, which stands for $\theta_{i\{b\}}$, and the *B*-length array **FomBs[ip,]**, which stands for $\theta_{i'\{b\}}$, and calculates the covariance, i.e., it directly implements Eqn. **Error! Reference source not found.**. The reader should confirm that the values are exactly those predicted by Eqn. **Error! Reference source not found.**. Line 12-18 averages the variances over the different treatments (i.e., it averages over all diagonal terms), and similarly, line 20-30 averages the co-variances (i.e., it averages over all off-diagonal terms); this yields the two numbers **Var** and **Cov1**, which are returned as a **list** variable to the calling code. Note the absence of the variance inflation term in the bootstrap method.

Online Appendix 10.C: Comparing DBMH and ORH methods for single-reader multiple-treatment

A listing of the file **mainOrDbmh1R** follows:

Online Appendix 10.C.1: Code listing

```
rm(list = ls()) #mainOrDbmh1R.R
library(RJafroc)
source("Wilcoxon.R")
source("VarCov1Jk.R")
ROC <- FALSE
if (ROC) {
  #fileName <- "Franken1.lrc"
  fileName <- "VanDyke.lrc"
  rocData <- ReadDataFile(fileName, format = "MRMC", renumber = "TRUE")
} else {
  fileName <- "CXRinvisible3-20mm.xlsx"
  frocData <- ReadDataFile(fileName, format = "JAFROC", renumber = "TRUE")
  rocData <- FROC2HrROC(frocData)
}
jSelect <- 1
rocData <- ExtractDataset(rocData, rdrs = jSelect)

ret1 <- DBMHAnalysis(rocData,fom = "Wilcoxon")
cat("data file = ", fileName, "\n")
cat("selected reader = ", jSelect, "\n")
cat("DBMH: F-stat = ", ret1$fFRRC, ", ddf = ", ret1$ddfFRRC, ", P-val = ", ret1$pFRRC,"\n")

ret2 <- ORHAnalysis(rocData,fom = "Wilcoxon")
cat("ORH:  F-stat = ", ret2$fFRRC, ", ddf = ", ret2$ddfFRRC, ", P-val = ", ret2$pFRRC,"\n")
```

Lines 1 – 16 are almost identical to the code described in Appendix 10.B.1. Line 20 implements DBMH analysis and line 21 prints out the relevant results. Lines 23 and 24 repeat the same for ORH analysis.

Online Appendix 10.D: Minimal Implementation of ORH method

A minimal version of ORH analysis is implemented in file **mainORH.R** listed below (the **RJafroc** package has the full implementation with more detailed output):

Online Appendix 10.D.1: Code Listing

```
rm(list = ls()) #mainORH.R
library(RJafroc)
source("Wilcoxon.R")
source("VarCovMtrxJK.R")
source("VarCovs.R")
source("Wilcoxon.R")
alpha <- 0.05
ROC <- FALSE
if (ROC) {
  #fileName <- "Franken1.lrc"
  fileName <- "VanDyke.lrc"
  rocData <- ReadDataFile(fileName, format = "MRMC", renumber = "TRUE")
} else {
  fileName <- "CXRinvisible3-20mm.xlsx"
  frocData <- ReadDataFile(fileName, format = "JAFROC", renumber = "TRUE")
  rocData <- FROC2HrROC(frocData)
}
```

```
zijk1 <- rocData$NL[,,,1];K <- dim(zijk1)[3];I <- dim(zijk1)[1];J <- dim(zijk1)[2]
zijk2 <- rocData$LL[,,,1];K2 <- dim(zijk2)[3];K1 <- K-K2;zijk1 <- zijk1[,,1:K1]

cat("data file = ", fileName, "\n")
cat("number of treatments = ", I, ", number of readers = ", J, ", number of non-diseased cases =
", K1,
    ", number of diseased cases =  = ", K2, "\n")

FOM <- array(dim=c(I,J))
for (i in 1:I) {
  for (j in 1:J) {
    FOM[i,j] <- Wilcoxon(zijk1[i,j,],zijk2[i,j,])
  }
}

mtrxJK <- VarCovMtrxJK(rocData)
VarCovJK <- VarCovs(mtrxJK)
Var <- VarCovJK$var;Cov1 <- VarCovJK$cov1;Cov2 <- VarCovJK$cov2;Cov3 <- VarCovJK$cov3

msT <- 0
for (i in 1:I){
  msT <- msT + (mean(FOM[i,]) - mean(FOM))^2
}
msT <- msT * J / (I - 1)

msTR <- 0
for (i in 1:I){
  for (j in 1:J) {
    msTR <- msTR + (FOM[i,j] - mean(FOM[i,]) - mean(FOM[,j]) + mean(FOM))^2
  }
}
msTR <- msTR / ((I - 1)*(J - 1))

cat("\nRandom reader random case analysis\n")
MS_DEN_DIFF_FOM_RRRC <- (msTR+max(J*(Cov2-Cov3),0))
F_ORH <- msT / MS_DEN_DIFF_FOM_RRRC
ndf <- (I-1)
ddfH <- MS_DEN_DIFF_FOM_RRRC^2/(msTR^2/((I-1)*(J-1)))
cat("Hillis ddfH = ", ddfH, "\n")
FCrit <- qf(1 - alpha, ndf, ddfH);cat("F statistic is ", F_ORH, "and critical value of F is ",
FCrit, "\n")
pValue <- 1 - pf(F_ORH, ndf, ddfH);cat("pvalue = ", pValue, "\n")

trtMeans <- array(dim = I)
for (i in 1:I) trtMeans[i] <- mean(FOM[i,])
trtDiff <- array(dim = c(I,I))
trtStr <- array(dim = c(I,I))
for (i1 in 1:(I-1)) {
  for (i2 in (i1+1):I) {
    trtDiff[i1,i2] <- trtMeans[i1]- trtMeans[i2]
    trtStr[i1,i2] <- gsub(", ", "", toString(c(i1,-i2)))
  }
}
trtDiff <- trtDiff[!is.na(trtDiff)];strDiff <- trtStr[!is.na(trtStr)]

std_DIFF_FOM_RRRC <- sqrt(2*MS_DEN_DIFF_FOM_RRRC/J)
nDiffs <- I*(I-1)/2
CI_DIFF_FOM_RRRC <- array(dim = c(nDiffs, 3))
for (i in 1 : nDiffs) {
  CI_DIFF_FOM_RRRC[i,1] <- trtDiff[i]
  CI_DIFF_FOM_RRRC[i,2] <- qt(alpha/2,df = ddfH)*std_DIFF_FOM_RRRC + trtDiff[i]
  CI_DIFF_FOM_RRRC[i,3] <- qt(1-alpha/2,df = ddfH)*std_DIFF_FOM_RRRC + trtDiff[i]
  cat("For pairing", strDiff[i], ", mean diff is ", CI_DIFF_FOM_RRRC[i,1], " and 95% CI is ",
CI_DIFF_FOM_RRRC[i,2], CI_DIFF_FOM_RRRC[i,3], "\n")
}

cat("\nFixed reader random case analysis\n")
MS_DEN_DIFF_FOM_FRRC <- Var-Cov1+(J-1)*max((Cov2-Cov3),0)
FDbmFR <- msT / MS_DEN_DIFF_FOM_FRRC
ndf <- (I-1)
ddf <- Inf
cat("ddf = ", ddf, "\n")
FCrit <- qf(1 - alpha, ndf, ddf);cat("F statistic is ", FDbmFR, "and critical value of F is ",
FCrit, "\n")
pValue <- 1 - pf(FDbmFR, ndf, ddf);cat("p-value is ", pValue, "\n")

std_DIFF_FOM_FRRC <- sqrt(2*MS_DEN_DIFF_FOM_FRRC/J)
nDiffs <- I*(I-1)/2
```

```
CI_DIFF_FOM_FRRC <- array(dim = c(nDiffs, 3))
for (i in 1 : nDiffs) {
  CI_DIFF_FOM_FRRC[i,1] <- trtDiff[i]
  CI_DIFF_FOM_FRRC[i,2] <- qt(alpha/2,df = ddf)*std_DIFF_FOM_FRRC + trtDiff[i]
  CI_DIFF_FOM_FRRC[i,3] <- qt(1-alpha/2,df = ddf)*std_DIFF_FOM_FRRC + trtDiff[i]
  cat("For pairing", strDiff[i], ", mean diff is ", CI_DIFF_FOM_FRRC[i,1], " and 95% CI is ",
CI_DIFF_FOM_FRRC[i,2], CI_DIFF_FOM_FRRC[i,3], "\n")
}

cat("\nRandom reader fixed case analysis\n")
MS_DEN_DIFF_FOM_RRFC <- msTR
FDbmFC <- msT / MS_DEN_DIFF_FOM_RRFC
ndf <- (I-1)
ddf <- (I-1)*(J-1)
cat("ddf = ", ddf, "\n")
FCrit <- qf(1 - alpha, ndf, ddf);cat("F statistic is ", FDbmFC, "and critical value of F is ",
FCrit, "\n")
pValue <- 1 - pf(FDbmFC, ndf, ddf);cat("p-value is ", pValue, "\n")

std_DIFF_FOM_RRFC <- sqrt(2*MS_DEN_DIFF_FOM_RRFC/J)
nDiffs <- I*(I-1)/2
CI_DIFF_FOM_RRFC <- array(dim = c(nDiffs, 3))
for (i in 1 : nDiffs) {
  CI_DIFF_FOM_RRFC[i,1] <- trtDiff[i]
  CI_DIFF_FOM_RRFC[i,2] <- qt(alpha/2,df = ddf)*std_DIFF_FOM_RRFC + trtDiff[i]
  CI_DIFF_FOM_RRFC[i,3] <- qt(1-alpha/2,df = ddf)*std_DIFF_FOM_RRFC + trtDiff[i]
  cat("For pairing", strDiff[i], ", mean diff is ", CI_DIFF_FOM_RRFC[i,1], " and 95% CI is ",
CI_DIFF_FOM_RRFC[i,2], CI_DIFF_FOM_RRFC[i,3], "\n")
}

plotM <- list(1, 2, 3, 4)
plotR <- list(c(1:5), c(1:5), c(1:5), c(1:5))
plot <- EmpiricalOpCharac(dataset = rocData, trts = plotM, rdrs = plotR,
                  lgdPos = "bottom", opChType = "ROC")
print(plot)
```

Lines 1-31 are similar to the previous examples excepting this time interest is in all readers, not just one reader. The file **CXRinvisible3-20mm.xlsx** is read and converted to an ROC dataset object **rocData**. Lines 19 – 20 extract the ratings and the dimensions of **rocData**. Lines 26 – 31 populate the array **FOM[1:I,1:J]** containing the empirical AUCs for all treatment-reader combinations. Insert a break point at line 33 and click **Source**. Highlight **FOM** and click **Run**.

Online Appendix 10.D.2: Code snippet

```
Browse[2]> FOM
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.5696662 0.4783200 0.5463534 0.5448113 0.5522496
[2,] 0.5296626 0.4960994 0.5599601 0.5244013 0.5714804
[3,] 0.6358853 0.6369739 0.6703556 0.6422351 0.6413280
[4,] 0.6249093 0.6269049 0.6673621 0.5862663 0.6262700
```

The printed array has four rows, corresponding to the four treatments, and five columns, corresponding to the five readers. Line 33 uses the included function **VarCovMtrxJK()** to estimate the covariance matrix, Eqn. **Error! Reference source not found.**, using the jackknife. [The reader is encouraged to step into the function to see how it works – there is a descriptive icon in the debug window that needs to be clicked.] The result is saved to **mtrxJK**. The function **VarCovs()** performs the requisite averaging, returning the 4 parameters characterizing the covariance matrix, Eqn. **Error! Reference source not found.**. The result is saved to **VarCovJK**. Click **Next** and examine the structure of **mtrxJK**.

Online Appendix 10.D.3: Code snippet

```
Browse[2]> str(mtrxJK)
 num [1:4, 1:4, 1:5, 1:5] 0.001442 0.000976 0.000297 0.00047 0.000976 ...
```

As expected, the covariance matrix is *IJ* x *IJ*, excepting that the indexing has been organized as *II* x *JJ* (there are 4 treatments and 5 readers). Click **Next** and examine the structure of **VarCovJK**.

Online Appendix 10.D.4: Code snippet

```
Browse[2]> str(VarCovJK)
List of 4
```

```
 $ var : num 0.00142
 $ cov1: num 0.000434
 $ cov2: num 0.000285
 $ cov3: num 0.000139
```

The 4 distinct elements of the covariance matrix, after the averaging, are shown above. The ordering shown in Eqn. **Error! Reference source not found.** is obeyed by the above listed values. Lines 37-41 calculate MST and lines 43 – 47 calculate MSTR, thereby implementing Eqn. **Error! Reference source not found.**. Lines 51 – 58 implements random-reader random-case analysis and prints out the value of the F-statistic, the critical value and the p-value. They are a straightforward application of the relevant formulae. Lines 60 – 70 is a little complicated, but the basic idea is to identify all different treatment pairings that are possible and to assign string names to them for clarity of printed output. Exit debug mode, clear any breakpoint and insert a new break point at line 72 and click **Source**. Highlight **trtDiff** (for reader-averaged inter-treatment FOM difference) and click **Run** and repeat for **strDiff** (for "helpful string identifying reader-averaged inter-treatment FOM difference"):

Online Appendix 10.D.5: Code snippet

```
Browse[2]> trtDiff
[1]  0.001959361 -0.107075472 -0.109034833 -0.088062409 -0.090021771  0.019013062
Browse[2]> strDiff
[1] "1-2" "1-3" "2-3" "1-4" "2-4" "3-4"
```

Lines 72 – 80 calculates and prints the reader-averaged differences in FOMs between different treatment pairings, with helpful strings indicating the specific pairing each difference applies to, and also calculates and prints the confidence interval. This completes random-reader random-case analyses. Lines 82 – 99 repeats the process for fixed reader analysis. Lines 101 – 118 repeats the process for fixed case analysis. The remaining 4 lines displays the reader-averaged empirical ROC plots for the four treatments, **Error! Reference source not found.**. Click **Continue** to execute the rest of the code.

Online Appendix 10.E: Proof of Eqn. 10.63

The OR model is:

$$\theta_{ij} = \mu + \tau_i + R_j + (\tau R)_{ij} + \varepsilon_{ij} \quad . \quad \textbf{(10.92.1)}$$

The sample mean for treatment i is:

$$\theta_{i.} = \frac{1}{J}\sum_{j=1}^{J}\left[\mu + \tau_i + R_j + (\tau R)_{ij} + \varepsilon_{ij}\right] \quad . \quad \textbf{(10.92.2)}$$

The variance of the sample mean for treatment i is:

$$\mathrm{var}(\theta_{i.}) = \frac{1}{J^2}\mathrm{var}\left(\sum_{j=1}^{J}\left[\mu + \tau_i + R_j + (\tau R)_{ij} + \varepsilon_{ij}\right]\right) \quad . \quad \textbf{(10.92.3)}$$

The terms are uncorrelated. Thus, the covariance between pairs of terms is 0, and above equation can be written as:

$$\mathrm{var}(\theta_{i.}) = \frac{1}{J^2}\left[\mathrm{var}\left(\sum_{j=1}^{J}\mu\right) + \mathrm{var}\left(\sum_{j=1}^{J}\tau_i\right) + \mathrm{var}\left(\sum_{j=1}^{J}R_j\right) + \mathrm{var}\left(\sum_{j=1}^{J}(\tau R)_{ij}\right) + \mathrm{var}\left(\sum_{j=1}^{J}\varepsilon_{ij}\right)\right] \quad . \quad \textbf{(10.92.4)}$$

Since μ is a fixed number and τ_i is fixed effect, their variances are 0. R_j and $R_{j'}$ are also uncorrelated for $j \neq j'$ and the covariance between them is 0. Likewise, the covariance between $(\tau R)_{ij}$ and $(\tau R)_{ij'}$ is also 0. The covariance between the error terms for different readers is included as follows:

$$\text{var}(\theta_{i.}) = \frac{1}{J^2}\left[\sum_{j=1}^{J}\sigma_R^2 + \sum_{j=1}^{J}\sigma_{\tau R}^2 + \sum_{j=1}^{J}\sigma_\varepsilon^2 + \sum_{j=1}^{J}\sum_{j'\neq j}\text{cov}\left(\varepsilon_{ij},\varepsilon_{ij'}\right)\right]$$

$$= \frac{1}{J^2}\left(J\sigma_R^2 + J\sigma_{\tau R}^2 + J\sigma_\varepsilon^2 + J(J-1)Cov2\right) \quad . \quad \textbf{(10.92.5)}$$

$$= \frac{1}{J}\left(\sigma_R^2 + \sigma_{\tau R}^2 + \sigma_\varepsilon^2 + (J-1)Cov2\right)$$

For a single treatment, the term $\sigma_{\tau R}^2$ can be removed, yielding Eqn. 10.67, reproduced below.

$$\sigma_{\theta.}^2 = \frac{1}{J}\left(\sigma_R^2 + Var + (J-1)Cov_2\right) . \quad . \quad \textbf{(10.92.6)}$$

■

Online Appendix 10.F: Single-treatment multiple-reader analysis

The file **mainSingleTreatment.R**, listed below, demonstrates single treatment analysis.

Online Appendix 10.F.1: Code Listing

```
rm(list = ls()) # mainSingleTreatment.R
library(RJafroc)
source("Wilcoxon.R")
source("CovJk.R")
source("CovDL.R")
source("SingleTreatmentAnalysis.R")
alpha <- 0.05
ROC <- FALSE
if (ROC) {
  #fileName <- "Franken1.lrc"
  fileName <- "VanDyke.lrc"
  rocData <- ReadDataFile(fileName, format = "MRMC", renumber = "TRUE")
} else {
  fileName <- "CXRinvisible3-20mm.xlsx"
  frocData <- ReadDataFile(fileName, format = "JAFROC", renumber = "TRUE")
  rocData <- FROC2HrROC(frocData)
}
cat("data file = ", fileName, "\n")

i <- 1 # select the treatment to be analyzed
singleData <- ExtractDataset(rocData, trts = i) # extract the first treatment
fomArray <- FigureOfMerit(singleData, "Wilcoxon")
thetaDot <- mean(fomArray[i, ])
mu0 <- 0.583422#mu0 <- 0.6
ret <- SingleTreatmentAnalysis(singleData, mu0, covEstMthd = "JK", alpha = alpha)
cat("The NH is that thetaDot = mu0, where thetaDot= ", thetaDot, "and mu0 = ", mu0,"\n")
cat("The mean FOM for the anal2zed treatment is:", thetaDot,"\n")
cat("The", 100 * (1 - alpha), "% CI for the preceding value is:", "(", ret$ci[1], ",", ret$ci[2],
")\n")
cat("The t-statistic and p-value to test H0: (analyzed treatment = standard) are:", ret$tStat, ",
and ", ret$pVal, "\n")
cat("The difference in reader averaged analyzed treatment minus standard = ",thetaDot - mu0, "\n")
cat("The", 100 * (1 - alpha), "% CI of the preceding value is", "(", ret$ciDiff[1], ",",
ret$ciDiff[2], ")\n")
```

Lines 1 – 18 should be familiar from the previous examples. Line 20 selects the 1st treatment for analysis, i.e., *i* = 1. Line 21 extracts data for selected treatment from the 4-treatment dataset. Line 22 calculates the FOMs of

the individual readers for the selected treatment. Line 23 calculates the average of the 5 FOM values; this is **thetaDot**, which stands for $\theta_\bullet = \mu$. Line 24 specifies the comparison value **mu0**, which stands for μ_0.

Line 25 applies single treatment analysis to this dataset and saves the results to **ret**. The remaining lines prints out the results contained in **ret**. The reader may wish to use debugging methods (insert a break point at line 24, **source** and click on the enter function button) to step into the function **SingleTreatmentAnalysis()** and be satisfied that it implements the formulae in the main text of this chapter.

Online Appendix 10.F.2: Code Listing

```
SingleTreatmentAnalysis <- function(singleData, mu0, covEstMthd = "JK", alpha = 0.05){
  fomArray <- FigureOfMerit(singleData, "Wilcoxon")
  J <- length(fomArray)
  NL <- singleData$NL
  LL <- singleData$LL
  K <- dim(NL)[3]
  K2 <- dim(LL)[3]
  K1 <- K - K2
  msR <- 0
  thetaDot <- mean(fomArray)
  for (j in 1:J){
    msR <- msR + (fomArray[j] - thetaDot)^2
  }
  msR <- msR / (J - 1)
  zijk1 <- NL[ , , 1:K1, ]
  dim(zijk1) <- c(1, J, K1)
  zijk2 <- LL
  dim(zijk2) <- c(1, J, K2)
  if (covEstMthd == "JK"){
    cov2 <- CovJk(zijk1, zijk2)$Cov2
  }else if (covEstMthd == "DL"){
    cov2 <- CovDL(zijk1, zijk2)$Cov2
  }

  msSingle <- msR + max(J * cov2, 0)
  dfSingle <- msSingle^2 / (msR^2/(J - 1))
  sigmaSingle <- sqrt(msSingle / J)
  tStat <- (thetaDot - mu0)/sigmaSingle
  pVal <- 2 * pt(abs(tStat), dfSingle, lower.tail = FALSE)
  halfCIWidth <- qt(alpha/2, dfSingle, lower.tail = FALSE) * sigmaSingle
  ci <- c(thetaDot - halfCIWidth, thetaDot + halfCIWidth)
  ciDiff <- ci - mu0
  return(list(
    ci = ci,
    tStat = tStat,
    pVal = pVal,
    ciDiff = ciDiff
  ))
}
```

G: References

1. DeLong ER, DeLong DM, Clarke-Pearson DL. Comparing the Areas Under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics.* 1988;44:837-845.
2. Dobbins JT, McAdams HP, Sabol JM, et al. Multi-Institutional Evaluation of Digital Tomosynthesis, Dual-Energy Radiography, and Conventional Chest Radiography for the Detection and Management of Pulmonary Nodules. Radiology. *Radiology.* 2016;000(000):(in press).