Supplementary material 1 to Chapter 11: Code for undertaking meta-analysis

Yemisi Takwoingi, Nandini Dendukuri, Ian Schiller, Gerta Rücker, Hayley E Jones, Christopher Partlett, Petra Macaskill

To cite this chapter, please use:

Takwoingi Y, Dendukuri N, Schiller I, Rücker G, Jones HE, Partlett C, Macaskill P. Supplement 1 to Chapter 11: Supplementary material 1 to Chapter 11: Code for undertaking meta-analysis. Draft version (13 May 2022) for inclusion in: Deeks JJ, Bossuyt PM, Leeflang MM, Takwoingi Y, editor(s). *Cochrane Handbook for Systematic Reviews of Diagnostic Test Accuracy* Version 2. London: Cochrane. Available from www.training.cochrane.org/handbook-diagnostic-test-accuracy

Appendix 1: Meta-analysis of a single index test using Proc NLMIXED and MetaDAS in SAS to fit a bivariate model

```
/*********** Using Proc NLMIXED for the bivariate model *****************/

/* Import data. Note you should change the file path to where you saved the
anti-CCP.csv file */

proc import out=nishimura_accp
      datafile='U:\Handbook 2020\anti-CCP.csv'
      dbms=csv replace;
      getnames=yes;

run;

proc print;

run;

/* Create two separate records for the true results in each study,

the first for the diseased group, and the second for the non-diseased
group. The variable sens is an indicator which takes the value 1 if
true=true positives and 0 otherwise, the variable spec is also an indicator
that takes the value 1 if true =true negatives and 0 otherwise */

data nishimura_accp;
      set nishimura_accp;
      sens=1; spec=0; true=tp; n=tp+fn; output;
      sens=0; spec=1; true=tn; n=tn+fp; output;

run;

/* Ensure that both records for a study are clustered together */

proc sort data=nishimura_accp;
      by study_id;

run;

/* Run the bivariate model with no covariates.

The "COV" option requests that a covariance matrix is printed for all model
parameter estimates. The "ECOV" option requests a covariance matrix for all
additional estimates that are computed. Using the PARMS statement, specify
starting values for all parameters to be estimated. Ensure that the
variances of the random effects cannot be negative by using the BOUNDS
statement. usens and uspec represent the random effects. They are both
assumed to be normally distributed with mean zero. Their variances
estimates are s2usens and s2uspec, and their covariance estimate is covesp.
Additional estimates, e.g. positive and negative likelihood ratios, that
are functions of the model parameters can be estimated using the ESTIMATE
statement. Take the exponent of the log LR values in the additional
estimates table to get the LRs. */

proc nlmixed data=nishimura_accp cov ecov;

parms msens=1 to 2 by 0.5 mspec=2 to 4 by 0.5 s2usens=0.2 s2uspec=0.6
covsesp=0;
      bounds s2usens>=0;
      bounds s2uspec>=0;
      logitp = (msens + usens)*sens + (mspec + uspec)*spec;
```

```
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0 , 0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
      estimate 'logLR+' log((exp(msens)/(1+exp(msens)))/(1-
      (exp(mspec)/(1+exp(mspec)))));
      estimate 'logLR-' log((1-
      (exp(msens)/(1+exp(msens))))/(exp(mspec)/(1+exp(mspec))));
run;
/* Check assumption of normality for the random effects */
proc univariate data=randeffs plot normal;
      class effect;
      var estimate;
run;


/*********** Using MetaDAS for the bivariate model ********************/
/* The INCLUDE statement specifies the path and name of the SAS file
containing the macro.*/
%include 'U:\1 Projects-Ongoing\METADAS macro\METADAS v1.3\Metadas
v1.3.sas';
/* The code below uses only 4 of the input parameters available in MetaDAS
to perform the meta-analysis of anti-CCP using the bivariate model
(method=b). The input parameter logfile is very useful. If the content of
the log window is saved to a file, the tables _metadas_errors,
_metadas_warnings and _metadas_modfail are produced and can be used for
identifying problems with the model or macro instead of going through the
entire log. */
%metadas(dtfile= 'U:\Handbook 2020\anti-CCP.csv', logfile='U:\Handbook
2020\anti-CCP log.log', method=b, rfile ='U:\Handbook 2020\anti-CCP
results.rtf');
run;
```

Appendix 2: Meta-analysis of a single index test using metandi in Stata to fit the bivariate model

```
*** You only need to run the following 2 lines if gllamm and metandi are
not installed on your computer or not up to date ***

ssc install gllamm, replace

ssc install metandi, replace

*** Read in the data from the .csv file. Note you should change the file
path to where you saved the anti-CCP.csv file ***

insheet using "U:\Handbook 2020\anti-CCP.csv", comma clear

*** Produce a summary of the dataset to check data import was ok ***

describe

*** Use metandi for meta-analysis of anti-CCP ***

metandi tp fp fn tn

*** To obtain a SROC plot as well as parameter estimates, add the plot
option to the metandi statement as follows ***

metandi tp fp fn tn, plot

*** There is no option with metandi to modify the plot but this can be done
using metandiplot. ***

metandiplot

/* If the optional variables tp fp fn tn are included in the command line,
the plot also includes estimates of sensitivity and specificity from each
study. */

metandiplot tp fp fn tn

/* The default is to scale the plot symbol by the sample size of each
study. To make the symbols all the same size, specify constant weights,
e.g. [aw=1]. Try other options too. */

metandiplot tp fp fn tn [aw=1], curve(off) pred(off)

*** Here's another example including some twoway graph options. ***

metandiplot tp fp fn tn [aw=1], curve(off) legend(off) title(SROC plot for
anti-CCP) scheme(s1mono)
```

Appendix 3: Meta-analysis of a single index test using meqrlogit in Stata to fit the bivariate model

```
*************** 1. IMPORT DATA *********************

*** Read in the data from the .csv file. Note you should change the file
path to where you saved the anti-CCP.csv file ***

insheet using "U:\Handbook 2020\anti-CCP.csv", comma clear

*** Produce a summary of the dataset to check data import was ok ***

describe


/*** 2. SET UP THE DATA BEFORE THE META-ANALYSIS

Generate 5 new variables of type long. We need these before we can reshape
the data.

• n1 is number diseased

• n0 is number without disease

• true1 is number of true positives

• true0 is the number of true negatives

• study is the unique identifier for each study. _n will generate a
sequence of numbers.*/

IMPORTANT NOTE: after modifying the data, if you want to run metandi,
remember to use the original data. ***/

gen long n1=tp+fn

gen long n0=fp+tn

gen long true1=tp

gen long true0=tn

gen long study= _n

*** Convert data from wide to long form ***

reshape long n true, i(study) j(sens)

*** Generate a new binary variable spec of type byte that takes the value 0
when sens=1 and vice versa ***

gen byte spec=1-sens

*** Sort data to ensure studies are clustered together first by study   ***

sort studyid


/*** 3. PERFORM META-ANALYSIS: replace meqrlogit with xtmelogit if your
version of Stata is earlier than Stata 13

Note that xtmelogit was introduced in Stata 10 and meqrlogit in Stata 13 so
you cannot run meqrlogit or xtmelogit if your version is earlier than Stata
10

Explanation of model specification:

• The variable true specifies the response while sens and spec are the
fixed portions of the model similar to if we were using regress or some
other Stata estimation command. Our fixed effects are coefficients on sens
and spec without a constant term (nocons)
```

```
• With || studyid: the random effects were specified at the level
identified by the group variable studyid.

• intpoints(5): the number of integration points for adaptive Gaussian
quadrature

• cov(): covariance specifies the structure of the covariance matrix for
the random effects. cov(un)specifies unstructured covariance allowing all
variances and covariances to be distinct.

• nocons: suppresses the constant (intercept) term and is specified here
for both the fixed effects and random-effects equations.

• refineopts(iterate(3)): controls the maximization process during the
refinement of starting values. Two iterations is the default. Should the
maximization fail because of instability in the Hessian calculations, one
possible solution may be to increase the number of iterations here.

• binomial(n): specifies the data are in binomial form and n as the
binomial variable.

• variance: displays the random-effects parameter estimates as variances
and covariances. To display standard deviations and correlation replace
option variance with stddeviations. ***/

meqrlogit true sens spec, nocons|| studyid: sens spec, ///

nocons cov(un) binomial(n) refineopts(iterate(5)) intpoints(5)  variance

/* To find the covariance between the expected (mean) logit sensitivity and
expected logit specificity, display contents of the variance-covariance
matrix: */

matrix list e(V)


*** DISPLAY SUMMARY ESTIMATES ***

*** Drop the program in case it is already in Stata's memory. ***

capture program drop renamematrix

/* Write a little program renaming elements of the coefficient and variance
matrices */

program define renamematrix, eclass

matrix mb = e(b)

matrix mv = e(V)

matrix colnames mb = logitse:_cons logitsp:_cons

matrix colnames mv = logitse:_cons logitsp:_cons

matrix rownames mv = logitse:_cons logitsp:_cons

ereturn post mb mv

end

*** Run the program ***

renamematrix

*** Display summary points ***

_diparm logitse, label(Sensitivity) invlogit

_diparm logitsp, label(Specificity) invlogit
```

```
_diparm logitse logitsp, label(LR+) ci(log) f(invlogit(@1)/(1-
invlogit(@2))) d( exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))

_diparm logitse logitsp, label(LR-) ci(log) f((1-
invlogit(@1))/invlogit(@2)) d( exp(-@1)*invlogit(@1)^2/invlogit(@2)  exp(-
@1-@2)*invlogit(@1))

_diparm logitse logitsp, label(DOR) ci(log) f(exp(@1+@2)) d(exp(@1+@2)
exp(@1+@2))
```

Appendix 4: Meta-analysis of a single index test using gllamm in Stata to fit a bivariate model

The structure of the model with `gllamm` is similar to `xtmelogit` in some respects. The main difference in the execution of `gllamm` is that the user must define equations for the linear predictors, multiplying the latent variables before running the command to fit the model the first time. `eqs(eq1 eq 0)` below specifies the equation names defined before running `gllamm`.

To run `gllamm`, use the code below. See the help file for a description of the available options for `gllamm` and the syntax.

```
************************ 1. IMPORT DATA ****************************

*** Read in the data from the .csv file. Note you should change the file
path to where you saved the anti-CCP.csv file ***

insheet using "U:\Handbook 2020\anti-CCP.csv", comma clear

*** Produce a summary of the dataset to check data import was ok ***

describe

************************ 2. SET UP THE DATA ****************************

/*** Generate 5 new variables of type long. We need these before we can
reshape the data.

• n1 is number diseased

• n0 is number without disease

• true1 is number of true positives

• true0 is the number of true negatives

• study is the unique identifier for each study. _n will generate a
sequence of numbers.

IMPORTANT NOTE: after modifying the data, if you want to run metandi,
remember to use the original data. ***/

gen long n1=tp+fn

gen long n0=fp+tn

gen long true1=tp

gen long true0=tn

gen long recordid= _n

*** Convert data from wide to long form ***

reshape long n true, i(recordid) j(sens)

*** Generate a new binary variable spec of type byte that takes the value 0
when sens=1 and vice versa ***

gen byte spec=1-sens

*** Sort data to ensure studies are clustered together first by study   ***

sort studyid


*************** 3. PERFORM META-ANALYSIS USING GLLAMM *************
```

```
/*** Numeric variable expected for option i() so encode studyid and
generate a new numeric variable named id. Since this analysis is not a test
comparison and there are no comparative studies, recordid and id are
essentially the same. However, to avoid confusion if this code is later
modified for a test comparison, id was created and used instead of
recordid. ***/

encode studyid, gen(id)

*** create the gllamm equations ***

eq eq1: sens

eq eq0: spec

*** See the help file for a description of the available options for gllamm
and the syntax. ***

gllamm true sens spec, nocons i(id) nrf(2) eqs(eq1 eq0) ///

family(binomial) link(logit) denom(n) ip(g) nip(5) adapt

/* To find the covariance between the expected (mean) logit sensitivity and
expected logit specificity, display contents of the variance-covariance
matrix: */

matrix list e(V)


*** DISPLAY SUMMARY ESTIMATES ***

*** Drop the program in case it is already in Stata's memory. ***

capture program drop renamematrix

/* Write a little program renaming elements of the coefficient and variance
matrices */

program define renamematrix, eclass

matrix mb = e(b)

matrix mv = e(V)

matrix colnames mb = logitse:_cons logitsp:_cons

matrix colnames mv = logitse:_cons logitsp:_cons

matrix rownames mv = logitse:_cons logitsp:_cons

ereturn post mb mv

end

*** Run the program ***

renamematrix

*** Display summary points ***

_diparm logitse, label(Sensitivity) invlogit

_diparm logitsp, label(Specificity) invlogit

_diparm logitse logitsp, label(LR+) ci(log) f(invlogit(@1)/(1-
invlogit(@2))) d( exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))

_diparm logitse logitsp, label(LR-) ci(log) f((1-
invlogit(@1))/invlogit(@2)) d( exp(-@1)*invlogit(@1)^2/invlogit(@2)  exp(-
@1-@2)*invlogit(@1))

_diparm logitse logitsp, label(DOR) ci(log) f(exp(@1+@2)) d(exp(@1+@2)
exp(@1+@2))
```

The results of the meta-analysis using `gllamm` are shown below. The parameters of the bivariate model are shown in the red (logit estimates) and the blue (variance, covariance and correlation estimates) boxes. For input into RevMan either the covariance or the correlation of the logits is needed. The covariance between the summary estimates of logit(sensitivity) and logit(specificity) is shown in the green box. There may be slight discrepancies in the parameter estimates obtained using `gllamm` compared to those obtained using `meqrlogit` directly or `metandi` due to different starting values and options such as number of integration points.

```
. gllamm true sens spec , nocons i(id) nrf(2) eqs(eq1 eq0) ///
> family(binomial) link(logit) denom(n)  ip(g) nip(5) adapt

Running adaptive quadrature
Iteration 0:    log likelihood = -284.43005
Iteration 1:    log likelihood = -273.40422
Iteration 2:    log likelihood =  -272.6518
Iteration 3:    log likelihood = -272.64672
Iteration 4:    log likelihood = -272.64672


Adaptive quadrature has converged, running Newton-Raphson
Iteration 0:   log likelihood = -272.64672
Iteration 1:   log likelihood = -272.64672  (backed up)
Iteration 2:   log likelihood = -272.64645
Iteration 3:   log likelihood = -272.64645

number of level 1 units = 74
number of level 2 units = 37

Condition Number = 1.7448082

gllamm model

log likelihood = -272.64645
```

true	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
sens	.6535312	.1276418	5.12	0.000	.4033578	.9037045
spec	3.109758	.1465388	21.22	0.000	2.822547	3.396969

```
Variances and covariances of random effects
------------------------------------------------------------------------------


***level 2 (id)

    var(1): .54390522 (.14654722)
    cov(2,1): -.27021072 (.12013621) cor(2,1): -.4822471

    var(2): .57722173 (.18850562)
------------------------------------------------------------------------------
```

var(1) is the variance of the random effects for logit sensitivity

var(2) is the variance of the random effects for logit specificity

Use either the covariance of the logits, cov(2,1), or the correlation of the logits, cor(2,1) in RevMan.

```
. matrix list e(V)

symmetric e(V)[5,5]
                    true:       true:      id1_1:      id1_2:    id1_2_1:
                    sens        spec        sens        spec       _cons
   true:sens    .01629243
   true:spec   -.00740711    .02147362
  id1_1:sens    .00060126   -.0000359    .00987125
  id1_2:spec    .00010537    .00250905    .00029921    .01404057
id1_2_1:_cons   .00025264   -.00155173   -.00310775    .00032123    .0210112
```

Appendix 5: Meta-analysis of a single index test using glmer in R to fit a bivariate model

```
## Set your working directory to the appropriate drive where you saved the
file anti-ccp.csv. Replace "U:/Handbook 2020" with your path.

setwd("U:/Handbook 2020")

################## 1. DATA IMPORT ###################################

### Read in the data from the .csv file. ##

(X=read.csv("anti-ccp.csv"))

############## 2. META-ANALYSIS OF DTA - USING GLMER ##################

### Install lme4 package if required (to run remove the # and select an
appropriate CRAN mirror).

# install.packages("lme4")

### Load the package lme4.

library(lme4)

### In order to specify the generalized linear model, first, we need to set
up the data.

### Generate 5 new variables of type long. We need these before we can
reshape the data.

# n1 is number diseased

# n0 is number without disease

# true1 is number of true positives

# true0 is the number of true negatives

# recordid is the unique identifier for each observation in the dataset

X$n1 <- X$TP+X$FN

X$n0 <- X$FP+X$TN

X$true1 <- X$TP

X$true0 <- X$TN

X$recordid <- 1:37

### Reshape the data from wide to long format ###

Y = reshape(X, direction="long", varying=list(c("n1", "n0"), c("true1",
      "true0")), timevar="sens", times=c(1,0), v.names=c("n","true"))

### Sort data by study to cluster the 2 records per study together. ###

Y = Y[order(Y$id),]

Y$spec<- 1-Y$sens

############################

### Perform meta-analysis ###

############################

## Now, let's examine the model specification and output in more detail.

# The variable true specifies the response while sens and spec are dummy
variables.

# The fixed effect for logit sensitivity and logit specificity are the
coefficients of sens and spec.
```

```
# The constant term is supressed by adding (0 + ...) to the model formula.

# Adding (0 + sens + spec | study) to the model includes study-level random
effects.

# family = binomial specifies the data are in binomial form.

# Specified in the form above, the between study covariance matrix is
unstructured.

# nAGQ controls number of points per axis for evaluating the adaptive
Gauss-Hermite approximation to the log-likelihood. Defaults to 1,
corresponding to the Laplace approximation.

(MA_Y = glmer(formula=cbind(true, n - true) ~ 0 + sens + spec + (0+sens +
      spec|Study.ID), data=Y, family=binomial, nAGQ=1, verbose=2))

### More detail can be obtained by using the summary command.

(ma_Y = summary(MA_Y))

### To obtain the between study covariance between logit sensitivity and
specificity for each test use

(summary(MA_Y))$vcov

### For the full list of outputs

labels( ma_Y)

### Therefore, to extract the coefficients

ma_Y$coeff

(lsens = ma_Y$coeff[1,1])

(lspec = ma_Y$coeff[2,1])

se.lsens = ma_Y$coeff[1,2]

se.lspec = ma_Y$coeff[2,2]

### Then we can manually create 95% confidence intervals for logit sens and
spec

Sens = c(lsens, lsens-qnorm(0.975)*se.lsens, lsens+qnorm(0.975)*se.lsens)

Spec = c(lspec, lspec-qnorm(0.975)*se.lspec, lspec+qnorm(0.975)*se.lspec)

### Or as a dataframe

logit_sesp = data.frame(estimate = c(lsens, lspec),

      lci = c(lsens-qnorm(0.975)*se.lsens, lspec-qnorm(0.975)*se.lspec),

      uci = c(lsens+qnorm(0.975)*se.lsens, lspec+qnorm(0.975)*se.lspec),

      row.names = c("lSens", "lSpec"))

### R has a built in logit and inv.logit function (use qlogis and plogis).

plogis(Sens)

plogis(Spec)

### Obtaining diagnostic odds rato (DOR), positive likelihood ratio (LRp)
and negative likelihood ratio (LRn)

(DOR = exp(lsens+lspec))

(LRp = plogis(lsens)/(1-plogis(lspec)))

(LRn = ((1-plogis(lsens))/plogis(lspec)))

### Standard errors and confidence intervals of DOR and LRs can be
calculated using delta method. This requires the package msm.
```

```
# install.packages("msm")

library(msm)

se.logDOR = deltamethod (~ (x1+x2), mean=c(lsens,lspec), cov=ma_Y$vcov)

se.logLRp = deltamethod (~ log((exp(x1)/(1+exp(x1)))/(1-
      (exp(x2)/(1+exp(x2))))), mean=c(lsens,lspec), cov=ma_Y$vcov)

se.logLRn = deltamethod (~ log((1-
      (exp(x1)/(1+exp(x1))))/(exp(x2)/(1+exp(x2)))), mean=c(lsens,lspec),
      cov=ma_Y$vcov)

data.frame(estimate = c(DOR, LRp, LRn),

      lci = c(exp(log(DOR)-qnorm(0.975)*se.logDOR), exp(log(LRp)-
      qnorm(0.975)*se.logLRp), exp(log(LRn)-qnorm(0.975)*se.logLRn)),

      uci = c(exp(log(DOR)+qnorm(0.975)*se.logDOR), exp(log(LRp)+
      qnorm(0.975)*se.logLRp), exp(log(LRn)+qnorm(0.975)*se.logLRn)),

      row.names = c("DOR", "LR+", "LR-"))
```

Appendix 6: Bayesian estimation of diagnostic test accuracy meta-analysis models using rjags in R

In order to use the rjags package within the R software environment please install JAGS first (from https://sourceforge.net/projects/mcmc-jags/files/).

Besides rjags, the packages DTAplots and mcmcplots are useful for producing graphs. All three packages must be loaded prior to the start of the session using the following commands.

```
library(rjags)
library(DTAplots)
library(mcmcplots)
```

The rjags model must be saved in a text file in the current working directory. This file is called *model.txt* in the illustrations below. In each case the R script is given first followed by the contents of the *model.txt* file.

A6.1: Bayesian bivariate model in 11.2.4

A6.1 Rscript

```
# Data

TP=c(115, 110, 40, 23, 236, 74, 89, 90, 31, 69, 25, 43, 70, 167, 26, 110, 26, 64,
71, 68, 38, 42, 149, 147, 47, 24, 40, 171, 72, 7, 481, 190, 82, 865, 139, 69, 90)

FP=c(17, 24, 5, 0, 20, 11, 4, 2, 0, 8, 2, 1, 5, 8, 8, 3, 1, 14, 2, 14, 3, 2, 7,
10, 7, 3, 11, 26, 14, 2, 23, 12, 13, 79, 7, 5, 7)

FN=c(16, 86, 58, 7, 88, 8, 29, 50, 22, 18, 10, 63, 17, 98, 15, 148, 20, 15, 58,
35, 0, 60, 109, 35, 20, 18, 46, 60, 77, 9, 68, 105, 71, 252, 101, 107, 101)

TN=c(73, 215, 227, 39, 231, 130, 142, 129, 75, 38, 40, 120, 228, 88, 15, 118, 56,
293, 66, 132, 73, 96, 114, 106, 375, 79, 146, 443, 298, 51, 185, 408, 301, 2218,
464, 133, 313)

pos=TP + FN

neg=TN + FP

n=length(TP) #  Number of studies

dataList = list(TP=TP,TN=TN,n=n,pos=pos,neg=neg)


# Compile the model

jagsModel = jags.model("model.txt",data=dataList,n.chains=3)

# Burn-in iterations

update(jagsModel,n.iter=5000)

# Parameters to be monitored

parameters = c( "Summary_Se" , "Summary_Sp", "prec", "mu", "tau", "tau.sq",
"rho",  "se","sp", "Predicted_Se", "Predicted_Sp")

# Posterior samples

output = coda.samples(jagsModel,variable.names=parameters,n.iter=10000)

# Plots for evaluating convergence
```

```
tiff("Figure_11_3.tiff",width = 23, height = 23, units = "cm", res=600)

par(oma=c(0,0,3,0))

layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))

denplot(output, parms=c("Summary_Se"), auto.layout=FALSE, main="(a)",
xlab="Summary_Se", ylab="Density")

rmeanplot(output, parms=c("Summary_Se"), auto.layout=FALSE, main="(b)")

title(xlab="Iteration", ylab="Running mean")

traplot(output, parms=c("Summary_Se"), auto.layout=FALSE, main="(c)")

title(xlab="Iteration", ylab="Summary_Se")

mtext("Diagnostics for Summary_Se", side=3, line=1, outer=TRUE, cex=2)

dev.off()

gelman.diag(output)

# Summary statistics

summary(output)

# Covariance between the posterior samples of the mean logit-transformed
sensitivity and mean logit-transformed specificity. This term is needed in order
to create the SROC plot in RevMan

cov(as.matrix(output[, "mu[1]"]), as.matrix(output[, "mu[2]"]))

# Posterior density plots

denplot(output, parms=c("Summary_Se","Summary_Sp"))

# SROC plot

SROC_rjags(X=output[[2]], model="Bivariate",n=n, study_col1="blue",
study_col2=rgb(0, 0, 1, 0.15), dataset=cbind(TP,FP,FN,TN),
ref_std=FALSE,SROC_curve = F)
```

A6.1 model.txt

```
model {

#=== LIKELIHOOD ===#

for(i in 1:n) {

TP[i] ~ dbin(se[i],pos[i])
TN[i] ~ dbin(sp[i],neg[i])

# === PRIOR FOR INDIVIDUAL LOGIT SENSITIVITY AND SPECIFICITY === #

logit(se[i]) <- l[i,1]
logit(sp[i]) <- l[i,2]

l[i,1:2] ~ dmnorm(mu[], T[,])

}

#=== HYPER PRIOR DISTRIBUTIONS POOLED LOGIT SENSITIVITY AND SPECIFICITY === #

mu[1] ~ dnorm(0,0.01)
mu[2] ~ dnorm(0,0.01)

# Between-study variance-covariance matrix


T[1:2,1:2]<-inverse(TAU[1:2,1:2])
TAU[1,1] <- tau[1]*tau[1]
TAU[2,2] <- tau[2]*tau[2]
TAU[1,2] <- rho*tau[1]*tau[2]
TAU[2,1] <- rho*tau[1]*tau[2]

#=== HYPER PRIOR DISTRIBUTIONS FOR PRECISION OF LOGIT SENSITIVITY ===#
#=== AND LOGIT SPECIFICITY, AND CORRELATION BETWEEN THEM === #
```

```
prec[1] ~ dgamma(2,0.5)
prec[2] ~ dgamma(2,0.5)
rho ~ dunif(-1,1)

# ===  PARAMETERS OF INTEREST === #

# BETWEEN-STUDY STANDARD DEVIATION (tau) AND VARIANCE (tau.sq) OF LOGIT
SENSITIVITY AND SPECIFICITY

tau[1]<-pow(prec[1],-0.5)

tau[2]<-pow(prec[2],-0.5)

tau.sq[1]<-pow(prec[1],-1)

tau.sq[2]<-pow(prec[2],-1)

# SUMMARY SENSITIVITY AND SPECIFICITY
Summary_Se <- 1/(1+exp(-mu[1]))
Summary_Sp <- 1/(1+exp(-mu[2]))

# PREDICTED SENSITIVITY AND SPECIFICITY IN A NEW STUDY
l.predicted[1:2] ~ dmnorm(mu[],T[,])
Predicted_Se <- 1/(1+exp(-l.predicted[1]))
Predicted_Sp <- 1/(1+exp(-l.predicted[2]))

}
```

A6.2: Bayesian HSROC model in 11.3.2

A6.2 RScript

```
# Data
TP=c(64, 482, 36, 143, 80, 61, 27, 60, 261, 20, 42, 18, 8, 93, 143,
     84, 30, 32, 70, 48, 75, 64, 32, 130, 50, 20, 77, 73, 36, 56,
     115, 49, 22, 8, 35, 161, 80, 5, 57, 383, 89, 57, 196, 163, 75,
     157, 26, 62, 113, 25)
FP=c(16, 2, 6, 43, 50, 36, 6, 8, 54, 2, 46, 3, 8, 28, 39, 41, 2,
     29, 39, 1, 42, 18, 10, 8, 14, 32, 16, 22, 3, 11, 53, 23, 2, 8,
     8, 89, 28, 4, 9, 38, 3, 25, 75, 10, 21, 287, 1, 11, 19, 1)
FN=c(27, 82, 41, 53, 39, 37, 3, 20, 63, 29, 14, 31, 0, 25, 63, 56,
     23, 3, 36, 40, 12, 29, 9, 128, 20, 26, 52, 29, 5, 46, 67, 28,
     20, 31, 51, 7, 69, 11, 33, 166, 9, 6, 99, 28, 21, 78, 32, 114,
     29, 14)
TN=c(153, 153, 313, 196, 45, 196, 33, 119, 197, 18, 127, 25, 31,
     118, 130, 90, 73, 13, 93, 99, 191, 73, 13, 113, 191, 25, 52,
     90, 70, 87, 63, 359, 80, 91, 149, 360, 284, 49, 93, 170, 39,
     111, 345, 140, 106, 1466, 29, 127, 481, 20)
pos= TP + FN
neg= TN + FP
n <- length(TP) #  Number of studies
dataList = list(TP=TP,FP=FP,n=n,pos=pos,neg=neg)
# Compile the model
jagsModel = jags.model("model.txt",data=dataList,n.chains=3)
# Burn-in iterations
```

```
update(jagsModel,n.iter=25000)

# Parameters to be monitored

parameters = c( "THETA", "LAMBDA", "beta", "prec", "tau.sq", "tau")

# Posterior samples

output = coda.samples(jagsModel,variable.names=parameters,n.iter=20000)

# Plots for evaluating convergence

tiff("Figure 11.6.tiff",width = 23, height = 23, units = "cm", res=600)

par(oma=c(0,0,3,0))

layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))

denplot(output, parms=c("LAMBDA"), auto.layout=FALSE, main="(a)", xlab="LAMBDA",
ylab="Density")

rmeanplot(output, parms=c("LAMBDA"), auto.layout=FALSE, main="(b)")

title(xlab="Iteration", ylab="Running mean")

traplot(output, parms=c("LAMBDA"), auto.layout=FALSE, main="(c)")

title(xlab="Iteration", ylab="LAMBDA")

mtext("Diagnostics for LAMBDA", side=3, line=1, outer=TRUE, cex=2)

dev.off()

gelman.diag(output)

# summary statistics

summary(output)

# Posterior density plots

denplot(output)

# SROC plot

SROC_rjags(X=output, model="HSROC",n=n, study_col1="blue", study_col2=rgb(0, 0,
1, 0.15), dataset=cbind(TP,FP,FN,TN), ref_std=FALSE,SROC_curve = T,
cex.summary.point = 0, cred_region=F, predict_region = F)
```

A6.2 model.txt

```
model {

# === LIKELIHOOD === #


for(i in 1:n) {

TP[i] ~ dbin(TPR[i],pos[i])

FP[i] ~ dbin(FPR[i],neg[i])

se[i] <- TPR[i]

sp[i] <- 1-FPR[i]

# === HIERARCHICAL PRIOR FOR TPR AND FPR === #


logit(TPR[i]) <- (theta[i] + 0.5*alpha[i])/exp(beta/2)

logit(FPR[i]) <- (theta[i] - 0.5*alpha[i])*exp(beta/2)

theta[i] ~ dnorm(THETA,prec[2])

alpha[i] ~ dnorm(LAMBDA,prec[1])

}

### === HYPER PRIOR DISTRIBUTIONS === ###

THETA ~ dunif(-10,10)
```

```
LAMBDA ~ dunif(-2,20)

beta ~ dunif(-5,5)

for(i in 1:2) {

prec[i] ~ dgamma(2.1,2)

tau.sq[i] <- 1/prec[i]

tau[i] <- pow(tau.sq[i],0.5)

}
```

A6.3: Bayesian bivariate meta-regression model in 11.4.4

A6.3 R script

```
# Data

TP=c(115, 110, 40, 23, 236, 74, 89, 90, 31, 69, 25, 43, 70, 167, 26, 110, 26, 64,
71, 68, 38, 42, 149, 147, 47, 24, 40, 171, 72, 7, 481, 190, 82, 865, 139, 69, 90)



FP=c(17, 24, 5, 0, 20, 11, 4, 2, 0, 8, 2, 1, 5, 8, 8, 3, 1, 14, 2, 14, 3, 2, 7,
10, 7, 3, 11, 26, 14, 2, 23, 12, 13, 79, 7, 5, 7)

FN=c(16, 86, 58, 7, 88, 8, 29, 50, 22, 18, 10, 63, 17, 98, 15, 148, 20, 15, 58,
35, 0, 60, 109, 35, 20, 18, 46, 60, 77, 9, 68, 105, 71, 252, 101, 107, 101)

TN=c(73, 215, 227, 39, 231, 130, 142, 129, 75, 38, 40, 120, 228, 88, 15, 118, 56,
293, 66, 132, 73, 96, 114, 106, 375, 79, 146, 443, 298, 51, 185, 408, 301, 2218,
464, 133, 313)

Z=c(1,0,0,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,0,1,0)

pos= TP + FN

neg= TN + FP

n <- length(TP) #  Number of studies

dataList = list(TP=TP,TN=TN,n=n,pos=pos,neg=neg,Z=Z)

# Compile the model

jagsModel = jags.model("model.txt",data=dataList,n.chains=3)

# Burn-in iterations

update(jagsModel,n.iter=25000)

# Parameters to be monitored

parameters = c( "mu", "nu", "mu_reg", "tau", "tau.sq", "rho",  "se","sp",
"Summary_Se_CCP1", "Summary_Se_CCP2", "Summary_Sp_CCP1", "Summary_Sp_CCP2",
"Difference_Se", "Difference_Sp", "prob_Se", "prob_Sp")

# Posterior samples

output = coda.samples(jagsModel,variable.names=parameters,n.iter=20000)

tiff("Figure 11.11.tiff",width = 23, height = 23, units = "cm", res=600)

par(oma=c(0,0,3,0))

layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))

denplot(output, parms=c("Summary_Se_CCP2"), auto.layout=FALSE, main="(a)",
xlab="Summary_Se_CCP2", ylab="Density")

rmeanplot(output, parms=c("Summary_Se_CCP2"), auto.layout=FALSE, main="(b)")

title(xlab="Iteration", ylab="Rsnning mean")

traplot(output, parms=c("Summary_Se_CCP2"), auto.layout=FALSE, main="(c)")
```

```
title(xlab="Iteration", ylab="Summary_Se_CCP2")
mtext("Diagnostics for Summary_Se_CCP2", side=3, line=1, outer=TRUE, cex=2)
dev.off()
gelman.diag(output)
# summary statistics
summary(output)
# Posterior density plots
denplot(output, parms=c("Summary_Se_CCP2"," Summary_Sp_CCP2"))
```

A6.3 model.txt

```
 model {
#=== LIKELIHOOD ===#
for(i in 1:n) {
TP[i] ~ dbin(se[i],pos[i])
TN[i] ~ dbin(sp[i],neg[i])
# === PRIOR FOR INDIVIDUAL LOGIT SENSITIVITY AND SPECIFICITY === #
logit(se[i]) <- l[i,1]
logit(sp[i]) <- l[i,2]
mu_reg[i,1] <- mu[1] + nu[1]*Z[i]
mu_reg[i,2] <- mu[2] + nu[2]*Z[i]
l[i,1:2] ~ dmnorm(mu_reg[i,], T[,])
}
#=== HYPER PRIOR DISTRIBUTIONS MEAN LOGIT SENSITIVITY AND SPECIFICITY === #
mu[1] ~ dnorm(0,0.01)
mu[2] ~ dnorm(0,0.01)
nu[1] ~ dnorm(0, 0.01)
nu[2] ~ dnorm(0, 0.01)
# Between-study variance-covariance matrix
T[1:2,1:2]<-inverse(TAU[1:2,1:2])
TAU[1,1] <- tau[1]*tau[1]
TAU[2,2] <- tau[2]*tau[2]
TAU[1,2] <- rho*tau[1]*tau[2]
TAU[2,1] <- rho*tau[1]*tau[2]
#=== HYPER PRIOR DISTRIBUTIONS FOR PRECISION OF LOGIT SENSITIVITY ===#
#=== AND LOGIT SPECIFICITY, AND CORRELATION BETWEEN THEM === #
prec[1] ~ dgamma(2,0.5)
prec[2] ~ dgamma(2,0.5)
rho ~ dunif(-1,1)
# ===  PARAMETERS OF INTEREST === #
# BETWEEN-STUDY STANDARD DEVIATION OF LOGIT SENSITIVITY AND SPECIFICITY
tau[1]<-pow(prec[1],-0.5)
tau[2]<-pow(prec[2],-0.5)
```

```
# BETWEEN-STUDY VARIANCE OF LOGIT SENSITIVITY AND SPECIFICITY
tau.sq[1]<-pow(tau[1],2)
tau.sq[2]<-pow(tau[2],2)


# SUMMARY SENSITIVITY AND SPECIFICITY OF CCP1
Summary_Se_CCP1<-1/(1+exp(-mu[1]))
Summary_Sp_CCP1<-1/(1+exp(-mu[2]))
# SUMMARY SENSITIVITY AND SPECIFICITY OF CCP2
Summary_Se_CCP2<-1/(1+exp(-mu[1] - nu[1]))
Summary_Sp_CCP2<-1/(1+exp(-mu[2] - nu[2]))
# PREDICTED SENSITIVITY AND SPECIFICITY IN A NEW STUDY EVALUATING CCP1
l_CCP1_new[1:2] ~ dmnorm(mu[],T[,])
Predicted_se_CCP1 <- 1/(1+exp(-l_CCP1_new[1]))
Predicted_sp_CCP1 <- 1/(1+exp(l_CCP1_new[2]))
# PREDICTED SENSITIVITY AND SPECIFICITY IN A NEW STUDY EVALUATING CCP2
mu_new[1] <- mu[1] + nu[1]
mu_new[2] <- mu[2] + nu[2]
l_CCP2_new[1:2] ~ dmnorm(mu_new[],T[,])
Predicted_se_CCP2 <- 1/(1+exp(-l_CCP2_new[1]))
Predicted_sp_CCP2 <- 1/(1+exp(l_CCP2_new[2]))
# POSTERIOR DIFFERENCES\ PROBABILITIES
Difference_Se <- Summary_Se_CCP2 - Summary_Se_CCP1
Difference_Sp <- Summary_Sp_CCP2 - Summary_Sp_CCP1


prob_Se <- step(Difference_Se)
prob_Sp <- step(Difference_Sp)
}
```

A6.4: Bayesian HSROC meta-regression model in 11.5.2

A6.4 RScript

```
# Data
TP=c(64, 482, 36, 143, 80, 61, 27, 60, 261, 20, 42, 18, 8, 93, 143,
     84, 30, 32, 70, 48, 75, 64, 32, 130, 50, 20, 77, 73, 36, 49,
     22, 8, 35, 161, 80, 5, 57, 383, 89, 57, 196, 163, 75, 157, 26,
     62, 113)
FP=c(16, 2, 6, 43, 50, 36, 6, 8, 54, 2, 46, 3, 8, 28, 39, 41, 2,
     29, 39, 1, 42, 18, 10, 8, 14, 32, 16, 22, 3, 23, 2, 8, 8, 89,
     28, 4, 9, 38, 3, 25, 75, 10, 21, 287, 1, 11, 19)
FN=c(27, 82, 41, 53, 39, 37, 3, 20, 63, 29, 14, 31, 0, 25, 63, 56,
     23, 3, 36, 40, 12, 29, 9, 128, 20, 26, 52, 29, 5, 28, 20, 31,
     51, 7, 69, 11, 33, 166, 9, 6, 99, 28, 21, 78, 32, 114, 29)
```

```
TN=c(153, 153, 313, 196, 45, 196, 33, 119, 197, 18, 127, 25, 31,
     118, 130, 90, 73, 13, 93, 99, 191, 73, 13, 113, 191, 25, 52,
     90, 70, 359, 80, 91, 149, 360, 284, 49, 93, 170, 39, 111, 345,
     140, 106, 1466, 29, 127, 481)
Z1=c(0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0,
     1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0,
     0, 0, 0, 0, 0, 0)
Z2=c(0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1,
     0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
     1, 1, 1, 0, 1, 0)
pos= TP + FN
neg= TN + FP
n <- length(TP) #  Number of studies
dataList = list(TP=TP,FP=FP,n=n,pos=pos,neg=neg,Z1=Z1,Z2=Z2)


# Compile the model
jagsModel = jags.model("model.txt",data=dataList,n.chains=3)
# Burn-in iterations
update(jagsModel,n.iter=25000)
# Parameters to monitor
parameters = c( "THETA", "LAMBDA", "beta", "prec", "tau.sq", "gamma", "epsilon",
"delta", "THETA_ELISA", "LAMBDA_ELISA", "beta_ELISA", "THETA_Nephelometry",
"LAMBDA_Nephelometry", "beta_Nephelometry")
# Posterior samples
output = coda.samples(jagsModel,variable.names=parameters,n.iter=20000)


# Plots to check convergence
for(i in c(1:3)) {
   par(oma=c(0,0,3,0))
   layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))
   denplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
   main="(a)", xlab=paste(parameters[i],sep=""), ylab="Density")
   rmeanplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
   main="(b)")
   title(xlab="Iteration", ylab="Running mean")
   traplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
   main="(c)")
   title(xlab="Iteration", ylab=paste(parameters[i],sep=""))
   mtext(paste("Diagnostics for ", parameters[i],"",sep=""), side=3, line=1,
   outer=TRUE, cex=2)
   #  dev.off()
}
gelman.diag(output)
# summary statistics
summary(output)
```

```
# Posterior density plots
denplot(output, parms=c("THETA","LAMBDA"))
```

A6.4 model.txt

```
model {
# === LIKELIHOOD === #
for(i in 1:n) {
TP[i] ~ dbin(TPR[i],pos[i])
FP[i] ~ dbin(FPR[i],neg[i])
se[i] <- TPR[i]
sp[i] <- 1-FPR[i]
# === HIERARCHICAL PRIOR FOR TPR AND FPR === #
logit(TPR[i]) <- (theta[i] + 0.5*alpha[i])/b[i]
logit(FPR[i]) <- (theta[i] - 0.5*alpha[i])*b[i]
t[i] <- THETA + Z1[i]*gamma[1] + Z2[i]*gamma[2]
l[i] <- LAMBDA + Z1[i]*epsilon[1] + Z2[i]*epsilon[2]
b[i] <- exp((beta + Z1[i]*delta[1] + Z2[i]*delta[2])/2)
theta[i] ~ dnorm(t[i],prec[1])
alpha[i] ~ dnorm(l[i],prec[2])
}
### === HYPER PRIOR DISTRIBUTIONS === ###
THETA ~ dunif(-10,10)
LAMBDA ~ dunif(-2,20)
beta ~ dunif(-5,5)
for(j in 1:2) {
gamma[j] ~ dnorm(0, 0.01)#dunif(-10,10)
epsilon[j] ~ dnorm(0, 0.01)#dunif(-2,20)
delta[j] ~ dnorm(0, 0.01)#dunif(-5,5)
prec[j] ~ dgamma(2.1,2)
tau.sq[j] <- 1/prec[j]
tau[j] <- pow(tau.sq[j],0.5)
}
### === PARAMETERS OF INTEREST === ###
THETA_Nephelometry <- THETA + gamma[1]
LAMBDA_Nephelometry <- LAMBDA + epsilon[1]
beta_Nephelometry <- beta + delta[1]
THETA_ELISA <- THETA + gamma[2]
LAMBDA_ELISA <- LAMBDA + epsilon[2]
beta_ELISA <- beta + delta[2]
}
```

A6.5: Bayesian latent class meta-analysis model using rjags

A6.5 R script

```
# Data
n11 <- c(7, 6, 2, 5, 0, 1, 4, 3, 3, 1, 7, 35, 103, 1, 2, 22, 6, 25,
         27, 15, 5, 2, 2, 11, 2, 0, 8, 3, 1)
n10 <- c(5, 4, 0, 0, 0, 0, 6, 0, 3, 0, 0, 5, 6, 1, 0, 1, 1, 3, 11, 3,
         9, 3, 0, 2, 4, 0, 2, 0, 2)
n01 <- c(5, 4, 0, 1, 2, 0, 8, 1, 1, 0, 0, 1, 18, 2, 0, 5, 0, 20, 25,
        7, 10, 1, 1, 2, 2, 3, 5, 0, 0)
n00 <- c(63, 115, 2, 44, 132, 4, 83, 250, 67, 14, 150, 119, 252, 107,
         8, 31, 148, 687, 204, 205, 115, 53, 4, 118, 22, 16, 186, 28,
         43)
cell<-cbind(n11, n10, n01, n00)
N <- length(cell[,1]) #  Number of studies
n <- rowSums(cell) #  Study sample sizes
dataList = list(cell=cell,N=N, n=n)
#Initial values
initsList = list(
  list( mu = c(0,0), mu2 = c(0,0)),
  list( mu = c(rnorm(1,0,2),rnorm(1,0,2)), mu2 = c(rnorm(1,0,2),rnorm(1,0,2))),
  list( mu = c(rnorm(1,0,2),rnorm(1,0,2)), mu2 = c(rnorm(1,0,2),rnorm(1,0,2))),
  list( mu = c(rnorm(1,0,2),rnorm(1,0,2)), mu2 = c(rnorm(1,0,2),rnorm(1,0,2))),
  list( mu = c(rnorm(1,0,2),rnorm(1,0,2)), mu2 = c(rnorm(1,0,2),rnorm(1,0,2))) )
# Compile the model
jagsModel = jags.model("model.txt",data=dataList,n.chains=5,inits=initsList,
n.adapt=0)
# Burn-in iterations
update(jagsModel,n.iter=30000)
# Parameters to be monitored
parameters = c( "prev", "se","sp", "Summary_Se", "Summary_Se2", "Summary_Sp",
"Summary_Sp2")
# Posterior samples
output = coda.samples(jagsModel,variable.names=parameters,n.iter=300000, thin=10)
# Plots to check convergence
for(i in c(1,2,10,11)) {
    tiff(paste(parameters[i],".tiff",sep=""),width = 23, height = 23, units =
    "cm", res=200)
    par(oma=c(0,0,3,0))
    layout(matrix(c(1,2,3,3), 2, 2, byrow = TRUE))
    denplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
    main="(a)", xlab=paste(parameters[i],sep=""), ylab="Density")
    rmeanplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
    main="(b)")
    title(xlab="Iteration", ylab="Running mean")
```

```
    traplot(output, parms=c(paste(parameters[i],sep="")), auto.layout=FALSE,
    main="(c)")

    title(xlab="Iteration", ylab=paste(parameters[i],sep=""))

    mtext(paste("Diagnostics for ", parameters[i],"",sep=""), side=3, line=1,
    outer=TRUE, cex=2)

    dev.off()

 }

gelman.diag(output)


# summary statistics

summary(output[[1]])

# Posterior density plots

denplot(output, parms=c("Summary_Se","Summary_Sp"))

# SROC plot

par(mfrow=c(1,2))

SROC_rjags(X=output_LC, model="Bivariate",n=N, study_col1="blue",
study_col2=rgb(0, 0, 1, 0.15), dataset=cell, ref_std=FALSE,SROC_curve = F,
title="Latent Class meta-analysis", Sp.range=c(0.8,1))

mtext("Latent Class meta-analysis", side=3, line=1, outer=FALSE, cex=1.25)

SROC_rjags(X=output_Bivariate, model="Bivariate",n=N, study_col1="blue",
study_col2=rgb(0, 0, 1, 0.15), dataset=cbind(TP,FP,FN,TN),
ref_std=TRUE,SROC_curve = F, title="Standard bivariate meta-
analysis",  Sp.range=c(0.8,1))

mtext("Standard Bivariate meta-analysis", side=3, line=1, outer=FALSE, cex=1.25)
```

A6.5 model.txt

```
model {

#=================================================================

# LIKELIHOOD

#=================================================================

for(i in 1:N) {

# The 4 cells in the two-by-two table of the new test vs. the imperfect reference
standard in each study in the meta-analysis

cell[i,1:4] ~ dmulti(prob[i,1:4],n[i])

# Multinomial probabilities of the 4 cells of the two-by-two table expressed in
terms of the disease prevalence (prev), the sensitivity (se1, se2) and
specificity (sp1, sp2) of the two tests and the covariance between them among
disease positive (covs12) and disease negative (covc12) groups

se[i] <- p[1,i]

sp[i] <- 1-p[2,i]

prob[i,1] <- prev[i]*( p[1,i]     * se2[i]     + covp[i] ) + (1-prev[i])*( p[2,i]
*(1-sp2[i]) + covn[i] )

prob[i,2] <- prev[i]*( p[1,i]     * (1-se2[i]) - covp[i] ) + (1-prev[i])*( p[2,i]
*sp2[i]     - covn[i] )

prob[i,3] <- prev[i]*( (1-p[1,i]) * se2[i]     - covp[i] ) + (1-prev[i])*( (1-
p[2,i]) *(1-sp2[i]) - covn[i] )

prob[i,4] <- prev[i]*( (1-p[1,i]) * (1-se2[i]) + covp[i] ) + (1-prev[i])*( (1-
p[2,i]) *sp2[i]     + covn[i] )
```

```
#================================================================
# CONDITIONAL DEPENDENCE
#================================================================
#=======================================
# upper limits of covariance parameters
#=======================================
us[i]<-min(se[i],se2[i])-(se[i]*se2[i]);
uc[i]<-min(sp[i],sp2[i])-(sp[i]*sp2[i]);
ls[i]<- -(1-se[i])*(1-se2[i])
lc[i]<- -(1-sp[i])*(1-sp2[i])
#================================================================
# prior distribution of transformed covariances on (0,1) range
#================================================================
covp[i]~dunif(ls[i],us[i]);
covn[i]~dunif(lc[i],uc[i]);
 #================================================================
# HIERARCHICAL PRIORS
#================================================================
#=======================================
# Hierarchichal prior for xpert
#=======================================
logit(p[1, i]) <- l[i,1]
logit(p[2, i]) <- -l[i,2]
l[i,1:2] ~ dmnorm(mu[], T[,])
#=======================================
# Prior distribution on prevalence
#=======================================
prev[i] ~ dbeta(1,1)
}
# ==================================
# Hierarchical prior for CULTURE
# ==================================
for(j in 1:N) {
logit(se2[j]) <- l2[j,1]
logit(sp2[j]) <- l2[j,2]
l2[j,1:2] ~ dmnorm(mu2[1:2], T2[1:2,1:2])
}
#================================================================
############  HYPER PRIOR DISTRIBUTIONS    #################
#================================================================
###                 XPERT TEST
#================================================================
### prior for the logit transformed sensitivity (mu[1]) and specificity (mu[2])
```

```
mu[1] ~ dnorm(0,0.25)

mu[2] ~ dnorm(0,0.25)

T[1:2,1:2]<-inverse(TAU[1:2,1:2])

#### BETWEEN-STUDY VARIANCE-COVARIANCE MATRIX

TAU[1,1] <- tau[1]*tau[1]

TAU[2,2] <- tau[2]*tau[2]

TAU[1,2] <- rho*tau[1]*tau[2]

TAU[2,1] <- rho*tau[1]*tau[2]

#### prec = between-study precision in the logit(sensitivity) and
logit(specificity)

prec[1] ~ dgamma(2,0.5)

prec[2] ~ dgamma(2,0.5)

rho ~ dunif(-1,1)

######################## OTHER PARAMETERS OF INTEREST

#### SUMMARY SENSITIVITY AND SPECIFICITY OF XPERT

Summary_Se<-1/(1+exp(-mu[1]))

Summary_Sp<-1/(1+exp(-mu[2]))

#### BETWEEN_STUDY VARIANCE IN THE LOGIT(SENSITIVITY) AND LOGIT(SPECIFICITY)

tau.sq[1] <- pow(tau[1], 2)

tau.sq[2] <- pow(tau[2], 2)

#### BETWEEN_STUDY STANDARD DEVIATION IN THE LOGIT(SENSITIVITY) AND
LOGIT(SPECIFICITY)

tau[1]<-pow(prec[1],-0.5)

tau[2]<-pow(prec[2],-0.5)

#### PREDICTED SENSITIVITY AND SPECIFICITY OF XPERT IN A FUTURE STUDY

Predicted.l[1:2] ~ dmnorm(mu[],T[,])

Predicted_Se <- 1/(1+exp(-Predicted.l[1]))

Predicted_Sp <- 1/(1+exp(-Predicted.l[2]))

#=================================================================

###                 CULTURE TEST

#=================================================================

### prior for the logit transformed sensitivity (mu2[1]) and specificity (mu2[2])

mu2[1] ~ dnorm(0,0.25)

mu2[2] ~ dnorm(0,0.25)

T2[1:2,1:2]<-inverse(TAU2[1:2,1:2])

#### BETWEEN-STUDY VARIANCE-COVARIANCE MATRIX

TAU2[1,1] <- tau2[1]*tau2[1]

TAU2[2,2] <- tau2[2]*tau2[2]

TAU2[1,2] <- rho2*tau2[1]*tau2[2]

TAU2[2,1] <- rho2*tau2[1]*tau2[2]

#### prec = between-study precision in the logit(sensitivity) and
logit(specificity)

prec2[1] ~ dgamma(2,0.5)

prec2[2] ~ dgamma(2,0.5)
```

```
rho2 ~ dunif(-1,1)
#=================================================================
# OTHER PARAMETERS OF INTEREST
#=================================================================
#### SUMMARY SENSITIVITY AND SPECIFICITY OF CULTURE
Summary_Se2<-1/(1+exp(-mu2[1]))
Summary_Sp2<-1/(1+exp(-mu2[2]))
#### BETWEEN_STUDY STANDARD DEVIATION IN THE LOGIT(SENSITIVITY) AND
LOGIT(SPECIFICITY)
tau2[1] <-pow(prec2[1],-0.5)
tau2[2] <-pow(prec2[2],-0.5)
#### BETWEEN_STUDY VARIANCE IN THE LOGIT(SENSITIVITY) AND LOGIT(SPECIFICITY)
tau.sq2[1] <- pow(tau2[1], 2)
tau.sq2[2] <- pow(tau2[2], 2)
#### PREDICTED SENSITIVITY AND SPECIFICITY OF CULTURE IN A FUTURE STUDY
Predicted_Se2 <- 1/(1+exp(-Predicted.ls2))
Predicted_Sp2 <- 1/(1+exp(-Predicted.lc2))
Predicted.ls2 ~ dnorm(mu2[1],prec2[1])
Predicted.lc2 ~ dnorm(mu2[2],prec2[2])
}
```

Appendix 7: Meta-analysis of a single index test using Proc NLMIXED and MetaDAS in SAS to fit an HSROC model

```
/*********** Using Proc NLMIXED for the HSROC model *****************/

/* Import data. Note you should change the file path to where you saved the
RF.csv file. */

proc import out=nishimura_RF
      datafile='U:\Handbook 2020\RF.csv'
      dbms=csv
      replace;
      getnames=yes;
run;

/* Display the data */

proc print;

run;

/* Create separate records for the diseased and non-diseased groups in each
study. The variable dis is the disease indicator which takes the value 0.5
if diseased and -0.5 if not diseased. */

data nishimura_RF;
      set nishimura_RF;
      dis=0.5; pos=tp; n=tp+fn; output;
      dis=-0.5; pos=fp; n=tn+fp; output;
run;

/* Ensure that both records for a study are clustered together. */

proc sort data=nishimura_RF;
      by study_id dis;
run;

/* Run the Rutter and Gatsonis HSROC model with no covariates. Request
covariance matrices for model parameters ("COV"). Using the PARMS
statement, specify starting values for all model parameters to be
estimated. Ensure that the variances of the random effects cannot be
negative by using the BOUNDS statement. The random effects for accuracy
(ua) and threshold (ut) are assumed to be approximately normally
distributed, both with mean zero and with variances s2ua and s2ut
respectively. The covariance of the random effects is set to 0 in the
RANDOM statement. */

proc nlmixed data=nishimura_RF cov;
      parms alpha=2 theta=-1 beta=0 s2ua=0 s2ut=0;
      bounds s2ua>=0;
      bounds s2ut>=0;
      logitp = (theta + ut + (alpha + ua)*dis) * exp(-(beta)*dis);
      p = exp(logitp)/(1+exp(logitp));
      model pos ~ binomial(n,p);
```

```
      random ut ua ~ normal([0,0],[s2ut,0,s2ua]) subject=study_id
      out=randeffs;
run;
/* Using the ESTIMATE statement to estimate sensitivity at a fixed value of
specificity. The ECOV option requests covariance matrices for additional
estimates. Estimate sensitivity at fixed specificities of 80%, 87% and
93%.*/
proc nlmixed data=nishimura_RF ecov cov;
      /* set starting values for all model parameters to be estimated */
      parms alpha=2 theta=-1 beta=0 s2ua=0 s2ut=0;
      bounds s2ua>=0;
      bounds s2ut>=0;
      logitp = (theta + ut + (alpha + ua)*dis) * exp(-(beta)*dis);
      p = exp(logitp)/(1+exp(logitp));
      model pos ~ binomial(n,p);
      random ut ua ~ normal([0,0],[s2ut,0,s2ua]) subject=study_id
      out=randeffs;
      estimate 'E(logitSe_sp80)' alpha*exp(-0.5*beta)+log(0.20/0.80)*exp(-
      beta);
      estimate 'E(logitSe_sp87)' alpha*exp(-0.5*beta)+log(0.13/0.87)*exp(-
      beta);
      estimate 'E(logitSe_sp93)' alpha*exp(-0.5*beta)+log(0.07/0.93)*exp(-
      beta);
run;
/*********** Using MetaDAS for the HSROC model *********************/
/* The INCLUDE statement specifies the path and name of the SAS file
containing the macro. */
%include 'U:\1 Projects-Ongoing\METADAS macro\METADAS v1.3\Metadas
v1.3.sas';
/* The code below uses only 4 of the input parameters available in MetaDAS
to perform the meta-analysis of RF using the HSROC model (method=h). Note
the default method in MetaDAS is the HSROC model so it is not necessary to
state this option when fitting the HSROC model. The input parameter logfile
is very useful. If the content of the log window is saved to a file, the
tables _metadas_errors, _metadas_warnings and _metadas_modfail are produced
and can be used for identifying problems with the model or macro instead of
going through the entire log. */
%metadas(dtfile= 'U:\Handbook 2020\RF.csv', logfile='U:\Handbook 2020\RF
log.log', method=h, rfile ='U:\Handbook 2020\RF results.rtf');
run;
```

Appendix 8: Bivariate meta-regression using Proc NLMIXED and MetaDAS in SAS – investigation of CCP generation

```
/* Import data. Note you should change the file path to where you saved the
anti-CCP.csv file. */
proc import out=nishimura_accp
      datafile='U:\Handbook 2020\anti-CCP.csv'
      dbms=csv replace;
      getnames=yes;
run;
proc print;
run;
/* Create two separate records for the true results in each study, the
first for the diseased group, and the second for the non-diseased group.
The variable sens is an indicator which takes the value 1 if true=true
positives and 0 otherwise, the variable spec is also an indicator that
takes the value 1 if true =true negatives and 0 otherwise. */
data nishimura_accp;
      set nishimura_accp;
      sens=1; spec=0; true=tp; n=tp+fn; output;
      sens=0; spec=1; true=tn; n=tn+fp; output;
run;
/* Ensure that both records for a study are clustered together. */
proc sort data=nishimura_accp;
      by study_id;
run;
/* Run the bivariate model with no covariates. */
proc nlmixed data=nishimura_accp cov ecov;
      parms msens=1 to 2 by 0.5 mspec=2 to 4 by 0.5 s2usens=0.2 s2uspec=0.6
      covsesp=0;
      bounds s2usens>=0;
      bounds s2uspec>=0;
      logitp = (msens + usens)*sens + (mspec + uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
      estimate 'logLR+' log((exp(msens)/(1+exp(msens)))/(1-(
      exp(mspec)/(1+exp(mspec)))));
      estimate 'logLR-' log((1-
      (exp(msens)/(1+exp(msens))))/(exp(mspec)/(1+exp(mspec))));
run;
```

```
/* Check assumption of normality for the random effects. */
proc univariate data=randeffs plot normal;
      class effect;
      var estimate;
run;


/*** META-REGRESSION FOR INVESTIGATIONS OF HETEROGENEITY AND TEST
COMPARISONS ***/
/* Create a dummy variable for CCP generation, coded as 0 for 'CCP1' (the
referent generation) and coded as 1 for 'CCP2'. This new variable is added
to the dataset set created above. */
data nishimura_accp;
      set nishimura_accp;
      ccpg=0;
      if generation ="CCP2" then ccpg=1;
run;


/***************** PART I: Model with equal variances ******************/
/* Add the covariate CCPG to the model to allow both sensitivity and
specificity to be associated with generation of the test. Estimate
logit(sensitivity) and logit(specificity) for CCP2 (their correlation will
be output because of the "ecov" option for nlmixed), and also log
likelihood ratios for CCP1 and CCP2. */
proc nlmixed data=nishimura_accp cov ecov;
      parms msens=1 mspec=2 s2usens=0.2 s2uspec=0.6 covsesp=0 se2=0 sp2=0;
      bounds s2usens>=0;
      bounds s2uspec>=0;
      logitp=(msens+usens+se2*ccpg)*sens+(mspec+uspec+sp2*ccpg)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0], [s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
      estimate 'logitsens CCP2'  msens + se2;
      estimate 'logitspec CCP2'  mspec + sp2;
      estimate 'logLR+ CCP1' log((exp(msens)/(1+exp(msens)))/(1-
      (exp(mspec)/(1+exp(mspec)))));
      estimate 'logLR- CCP1' log((1-
      (exp(msens)/(1+exp(msens))))/(exp(mspec)/(1+exp(mspec))));
      estimate 'logLR+ CCP2' log((exp(msens+se2)/(1+exp(msens+se2)))/(1-
      (exp(mspec+sp2)/(1+exp(mspec+sp2)))));
      estimate 'logLR- CCP2' log((1-
```

```
      (exp(msens+se2)/(1+exp(msens+se2))))/(exp(mspec+sp2)/(1+exp(mspec+sp2
      ))));
run;
/* Check assumption of normality for the random effects. */
proc univariate data=randeffs plot normal;
  class effect;
  var estimate;
run;


/*** Using MetaDAS for bivariate meta-regression ***/
/* The INCLUDE statement specifies the path and name of the SAS file
containing the macro. Do not need to run this again if included previously
by running the code above. */
%include 'U:\1 Projects-Ongoing\METADAS macro\METADAS v1.3\Metadas
v1.3.sas';
/* The additional options to add to the code are the ones for the covariate
(covariate=generation) and type of effect (cveffect), i.e. covariate terms
for both sensitivity and specificity (cveffect=sesp), for only sensitivity
(cveffect=se) or for only specificity (cveffect=sp). */
%metadas(dtfile= 'U:\Handbook 2020\anti-CCP.csv', logfile='U:\Handbook
2020\anti-CCP regression log.log', method=b, cveffect=sesp, covariate =
generation, rfile ='U:\Handbook 2020\anti-CCP regression results.rtf');
run;


/*************** PART II: Model with unequal variances ****************/
/*Meta-analysis of CCP1 */
proc nlmixed data=nishimura_accp cov ecov;
      parms msens=1 mspec=2 s2usens=0.5 s2uspec=0.5 covsesp=0.1;
      where ccpg=0;
      bounds s2usens>=0;
      bounds s2uspec>=0;
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0], [s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;
/*Meta-analysis of CCP2 */
proc nlmixed data=nishimura_accp cov ecov;
      parms msens=1 mspec=2 s2usens=0.5 s2uspec=0.5 covsesp=0.1;
      where ccpg=1;
      bounds s2usens>=0;
      bounds s2uspec>=0;
```

```
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0], [s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;


/* Easier for obtaining the required parameter estimates and covariances,
and also for extending beyond binary variables if dummy variables are
created for every subgroup of the covariate. */
data nishimura_accp;
      set nishimura_accp;
      ccpg1=0;
      ccpg2=0;
      if generation ="CCP1" then ccpg1=1;
      if generation ="CCP2" then ccpg2=1;
run;
/* Fitting model with unequal variances.
The analysis did not converge with the default optimization technique (a
quasi-Newton technique) and so the technique was changed to the Newton-
Raphson technique specified as "tech=newrap". Due to the change in the
format of the regression equation compared to how it was specified for
model 1, the ESTIMATE statements can be used to obtain the differences in
the mean logits for sensitivity and specificity of CCP1 and CCP2 along with
P values based on Wald statistics. */
proc nlmixed data=nishimura_accp cov ecov tech=newrap;
      parms msens1=0.1 mspec1=2 s2usens1=0.5 s2uspec1=0.5 covsesp1=0.1
            msens2=1 mspec2=-0.2 s2usens2=0.5 s2uspec2=0.5 covsesp2=0.1;
      bounds s2usens1>=0;
      bounds s2uspec1>=0;
      bounds s2usens2>=0;
      bounds s2uspec2>=0;
      logitp=((msens1+usens1)*ccpg1+(msens2+usens2)*ccpg2)*sens+
      ((mspec1+uspec1)*ccpg1+(mspec2+uspec2)*ccpg2)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens1 uspec1 usens2 uspec2 ~ normal([0,0,0,0],
      [s2usens1,covsesp1,s2uspec1,0,0,s2usens2,0,0,covsesp2,s2uspec2])
      subject=study_id out=randeffs;
      estimate 'Diff logitsens CCP2-CCP1' msens2-msens1;
      estimate 'Diff logitspec CCP2-CCP1' mspec2-mspec1;
      estimate 'logLR+ CCP1' log((exp(msens1)/(1+exp(msens1)))/(1-(
      exp(mspec1)/(1+exp(mspec1)))));
```

```
      estimate 'logLR- CCP1' log((1-
      (exp(msens1)/(1+exp(msens1))))/(exp(mspec1)/(1+exp(mspec1))));
      estimate 'logLR+ CCP2' log((exp(msens2)/(1+exp(msens2)))/(1-
      (exp(mspec2)/(1+exp(mspec2)))));
      estimate 'logLR- CCP2' log((1-
      (exp(msens2)/(1+exp(msens2))))/(exp(mspec2)/(1+exp(mspec2))));
run;
/* Check assumption of normality for the random effects. */
proc univariate data=randeffs plot normal;
  class effect;
  var estimate;
run;
```

Appendix 9: Bivariate meta-regression using Proc NLMIXED in SAS – comparison of CT and MRI for CAD

```
/*************** META-REGRESSION FOR TEST COMPARISONS *************/

/* Import data. Note you should change the file path to where you saved the
schuetz.csv file. */

proc import out=schuetz

      datafile='U:\Handbook 2020\schuetz.csv'

      dbms=csv replace;

      getnames=yes;

run;


/* Create a two separate records for the true results in each study, the
first for the diseased group, and the second for the non-diseased group.
The variable sens is an indicator which takes the value 1 if true=true
positives and 0 otherwise, the variable spec is also an indicator that
takes the value 1 if true =true negatives and 0 otherwise. */

data schuetz;

      set schuetz;

      testtype=0;

      if test ="CT" then testtype=1;

      sens=1; spec=0; true=tp; n=tp+fn; output;

      sens=0; spec=1; true=tn; n=tn+fp; output;

run;


/* Ensure that both records for a study are clustered together. */

proc sort data=schuetz;

      by study_id test;

run;


/* Run the bivariate model with no covariates

The "cov" option requests that a covariance matrix is printed for all model
parameter estimates.

The "ecov" option requests a covariance matrix for all additional estimates
that are computed. */

proc nlmixed data=schuetz cov ecov;

      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0 ;

      logitp=(msens+usens)*sens+(mspec+uspec)*spec;

      p = exp(logitp)/(1+exp(logitp));

      model true ~ binomial(n,p);

      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
```

```
run;


/* Bivariate model with test as a covariate, variances of the random
effects are assumed not to vary by test type. */
proc nlmixed data=schuetz cov ecov;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0 se_CT=1 sp_CT=0;
      logitp=(msens+usens+se_CT*testtype)*sens+(mspec+uspec+sp_CT*testtype)
      *spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
      estimate 'logitsens CT'  msens + se_CT;
      estimate 'logitspec CT'  mspec + sp_CT;
run;


/* Check assumption of normality for the random effects. */
proc univariate data=randeffs plot normal;
  class effect;
  var estimate;
run;


/* Bivariate model with effect of test type on only sensitivity. */
proc nlmixed data=schuetz cov ecov;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0 se_CT=1 ;
      logitp=(msens+usens+se_CT*testtype)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;


/* Bivariate model with effect of test type on only specificity. */
proc nlmixed data=schuetz cov ecov;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0 sp_CT=0;
      logitp=(msens+usens)*sens+(mspec+uspec+sp_CT*testtype)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;
```

```
/***************** DIRECT COMPARISONS ************************/
/* Create new dataset of studies with within-study comparison of CT and
MRI. "indirect" is a binary variable in the dataset coded 1 if the study
evaluated only one test (CT or MRI) and 0 if both tests were evaluated in a
study. */
data schuetz_direct;
      set schuetz;
      where indirect=0;
run;


/* Fit bivariate model without covariate. */
proc nlmixed data=schuetz_direct cov ecov;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0;
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;


/* Fit bivariate model without covariate - Attempt 1: increasing the number
of quadrature points to 10 using option qpoints. */
proc nlmixed data=schuetz_direct cov ecov qpoints=10;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0;
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;
run;


/* Fit bivariate model without covariate - Attempt 2: changing the
optimization technique from the default quasi-Newton technique to the
Newton-Raphson technique. */
proc nlmixed data=schuetz_direct cov ecov tech=newrap;
      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0;
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
subject=study_id out=randeffs;
```

```
run;

/* Fit bivariate model without covariate - Attempt 3: grid search for
starting values for parameters. */

proc nlmixed data=schuetz_direct cov ecov;

      parms msens=2 to 3 by 0.2 mspec=1 to 2 by 0.2  s2usens=0 to 0.2 by
      0.05 s2uspec=0 to 1 by 0.1 covsesp=-0.1 to 0.1 by 0.05;

      logitp=(msens+usens)*sens+(mspec+uspec)*spec;

      p = exp(logitp)/(1+exp(logitp));

      model true ~ binomial(n,p);

      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;

run;


/* Fit bivariate model without covariate - Attempt 4: set boundary
constraints for the variance parameters. */

proc nlmixed data=schuetz_direct cov ecov;

      parms msens=2 mspec=1 s2usens=0 s2uspec=0 covsesp=0;

      bounds s2usens>=0;

      bounds s2uspec>=0;

      logitp=(msens+usens)*sens+(mspec+uspec)*spec;

      p = exp(logitp)/(1+exp(logitp));

      model true ~ binomial(n,p);

      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;

run;


/* Fit bivariate model without covariate - Attempt 5: set boundary
constraints for the variance parameters, and change starting values for the
variance parameters. */

proc nlmixed data=schuetz_direct cov ecov;

      parms msens=2 mspec=1 s2usens=0.1 s2uspec=0.1 covsesp=0;

      bounds s2usens>=0;

      bounds s2uspec>=0;

      logitp=(msens+usens)*sens+(mspec+uspec)*spec;

      p = exp(logitp)/(1+exp(logitp));

      model true ~ binomial(n,p);

      random usens uspec ~ normal([0,0],[s2usens,covsesp,s2uspec])
      subject=study_id out=randeffs;

      estimate 'Corr(logits)' covsesp/(SQRT(s2usens)*SQRT(s2uspec)) ;

run;


/******** SIMPLIFYING THE BIVARIATE MODEL TO UNIVARIATE MODELS *********/
```

```
/* Simplify the bivariate model to 2 univariate random-effects logistic
regression models by removing the covariance parameter. */
proc nlmixed data=schuetz_direct cov ecov;
      parms msens=2 mspec=1 s2usens=0.1 s2uspec=0.1;
      bounds s2usens>=0;
      bounds s2uspec>=0;
      logitp=(msens+usens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random usens uspec ~ normal([0,0],[s2usens,0,s2uspec])
      subject=study_id out=randeffs;
run;


/* Simplify the bivariate model further by removing the variance parameter
for logit sensitivity, i.e. fixed effect for sensitivity. */
proc nlmixed data=schuetz_direct cov ecov;
      parms msens=2 mspec=1 s2uspec=0.1;
      bounds s2uspec>=0;
      logitp=(msens)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random uspec ~ normal([0],[s2uspec]) subject=study_id out=randeffs;
run;
/* Fit the simplified model with covariate terms for both sensitivity and
specificity. Random effects only for specificity. */
proc nlmixed data=schuetz_direct cov ecov qpoints=10;
      parms msens=2 mspec=1 s2uspec=0.1 se_CT=0 sp_CT=0;
      bounds s2uspec>=0;
      logitp=(msens+se_CT*testtype)*sens+(mspec+uspec+sp_CT*testtype)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random uspec ~ normal([0],[s2uspec]) subject=study_id out=randeffs;
      estimate 'logitsens CT'  msens + se_CT;
      estimate 'logitspec CT'  mspec + sp_CT;
run;


/* Fit the model with covariate terms only for specificity. Random effects
only for specificity. */
proc nlmixed data=schuetz_direct cov ecov;
      parms msens=2 mspec=1 s2uspec=0.1 sp_CT=0;
      bounds s2uspec>=0;
```

```
      logitp=(msens)*sens+(mspec+uspec+sp_CT*testtype)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random uspec ~ normal([0],[s2uspec]) subject=study_id out=randeffs;
run;


/* Fit bivariate model with covariate terms only for sensitivity. Random
effects only for specificity. */
proc nlmixed data=schuetz_direct cov ecov;
      parms msens=2 mspec=1  s2uspec=0.1 se_CT=0;
      bounds s2uspec>=0;
      logitp=(msens+se_CT*testtype)*sens+(mspec+uspec)*spec;
      p = exp(logitp)/(1+exp(logitp));
      model true ~ binomial(n,p);
      random uspec ~ normal([0],[s2uspec]) subject=study_id out=randeffs;
run;
```

Appendix 10: Bivariate meta-regression using meqrlogit (or xtmelogit) in Stata – investigation of CCP generation

```
/* Set your working directory to the appropriate drive where you saved the
file "anti-ccp.csv". Replace "U:\Handbook 2020" with your path. */

cd "U:\Handbook 2020"

*************** 1. IMPORT DATA *********

*** Read in the data from the .csv file. Note you should change the file
path to where you saved the anti-CCP.csv file. ***

insheet using "U:\Handbook 2020\anti-CCP.csv", comma clear

*** Produce a summary of the dataset to check data import was ok. ***

describe


/************* 2. SET UP THE DATA BEFORE THE META-ANALYSIS

Generate 5 new variables of type long. We need these before we can reshape
the data.

• n1 is number diseased

• n0 is number without disease

• true1 is number of true positives

• true0 is the number of true negatives

• study is the unique identifier for each study. _n will generate a
sequence of numbers. */

gen long n1=tp+fn

gen long n0=fp+tn

gen long true1=tp

gen long true0=tn

gen long recordid= _n

*** Convert data from wide to long form. ***

reshape long n true, i(recordid) j(sens)

*** Generate a new binary variable spec of type byte that takes the value 0
when sens=1 and vice versa. ***

gen byte spec=1-sens

*** Sort data to ensure studies are clustered together first by study.  ***

sort studyid

***** 3. META-REGRESSION - USING meqrlogit WITH A COVARIATE *********

*** Replace meqrlogit with xtmelogit if that is the command available in
your version of Stata. ***

*** Create dummy variables for the covariate anti-CCP generation. ***

gen se1=0

gen sp1=0

gen se2=0

gen sp2=0
```

```
replace se1=1 if generation=="CCP1" & sens==1

replace sp1=1 if generation=="CCP1" & spec==1

replace se2=1 if generation=="CCP2" & sens==1

replace sp2=1 if generation=="CCP2" & spec==1

/*** Perform meta-analysis for each generation of anti-CCP and examine the
variances of the random effects. Are they similar so that common/equal
variances can be assumed in the meta-regression model or are they very
different? ***/

*** Perform meta-analysis for CCP1 ***

meqrlogit true sens spec if generation=="CCP1", nocons|| studyid: sens
spec, nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5)
variance

*** Perform meta-analysis for CCP2 ***

meqrlogit true sens spec if generation=="CCP2", nocons|| studyid: sens
spec, nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5)
variance


/*** PART I: ANALYSES WITH COVARIATE GENERATION ASSUMING EQUAL VARIANCES
***/

*** Fit the model without the covariate. ***

meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

*** Store the estimates of the log likelihood from the model above for
doing the likeilhood ratio test later. ***

estimates store A

/*** Add covariate terms to the model for both logit sensitivity and logit
specificity. This model assumes equal variances of the random effects for
the logit sensitivities (and similarly for that of the logit specificities)
for both generations of anti-CCP. ***/

meqrlogit true se1 se2 sp1 sp2, nocons || studyid: sens spec, nocons
cov(un) binomial(n)  refineopts(iterate(3)) intpoints(5) variance nolr

estimates store B

/* Perform a likelihood ratio test comparing model (A) without covariate
with model (B) that includes the covariate generation and assumes equal
variances. Use the stored values in A and B. */

lrtest A B

*** Assume sensitivity is the same for both generations but allow
specificity to vary with anti-CCP generation. ***

meqrlogit true sens sp1 sp2, nocons || studyid: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store C

/* Perform a likelihood ratio test comparing model (C) that assumes
sensitivity is the same for both generations but allows specificity to vary
with generation with model (B) that allows both sensitivity and specificity
to vary with anti-CCP generation. Use the stored values in B and C. */

lrtest B C

*** Assume specificity is the same for both generations but allow
sensitivity to vary with anti-CCP generation. ***
```

```
meqrlogit true se1 se2 spec, nocons || studyid: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store D

/* Perform a likelihood ratio test comparing model (D) that assumes
specificity is the same but allows sensitivity to vary with generation with
model (B) that allows both sensitivity and specificity to vary with anti-
CCP generation. Use the stored values in B and D. */

lrtest B D

*** Run the model with covariate terms for both sensitivity and
specificity) again. ***

meqrlogit true se1 se2 sp1 sp2, nocons || studyid: sens spec, nocons
cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

/* To find the covariance between the expected (mean) logit sensitivity and
expected logit specificity, display contents of the variance-covariance
matrix. */

matrix list e(V)

*** Delete the program from Stata's memory if it exists already. ***

capture program drop renamematrix

*** Rename the elements of the coefficient and variance matrices. ***

program define renamematrix, eclass

      matrix mb = e(b)

      matrix mv = e(V)

      matrix colnames mb = logitse1:_cons logitse2:_cons logitsp1:_cons
      logitsp2:_cons

      matrix colnames mv = logitse1:_cons logitse2:_cons logitsp1:_cons
      logitsp2:_cons

      matrix rownames mv = logitse1:_cons logitse2:_cons logitsp1:_cons
      logitsp2:_cons

      ereturn post mb mv

end

*** Run the program. ***

renamematrix


*** Display summary estimates by taking the inverse logits of the mean
logit sensitivity and mean logit specificity for each test. ***

_diparm logitse1, label(Sensitivity CCP1) invlogit

_diparm logitse2, label(Sensitivity CCP2) invlogit

_diparm logitsp1, label(Specificity CCP1) invlogit

_diparm logitsp2, label(Specificity CCP2) invlogit

*** Display other summary estimates derived using functions of the mean
logit sensitivities and mean logit specificities. ***

_diparm logitse1 logitsp1, label(LR+ CCP1) ci(log)
function(invlogit(@1)/(1-invlogit(@2))) derivative(exp(@2-
1)*invlogit(@1)^2/invlogit(@2) exp(@2)*invlogit(@1))
```

```
_diparm logitse2 logitsp2, label(LR+ CCP2) ci(log)
function(invlogit(@1)/(1-invlogit(@2))) derivative(exp(@2-
1)*invlogit(@1)^2/invlogit(@2) exp(@2)*invlogit(@1))

_diparm logitse1 logitsp1, label(LR- CCP1) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))

_diparm logitse2 logitsp2, label(LR- CCP2) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))


/*** PART II: ANALYSES WITH COVARIATE GENERATION ALLOWING FOR UNEQUAL
VARIANCES ***/

/* As noted in the "Analysing and presenting results" chapter, the
variances of the random effects were assumed to be the same for
logit(sensitivity) and logit(specificity) for both generations of the test
in the above analyses. This assumption can be investigated by fitting
additional models as follows. A likelihood ratio test can be used to assess
if separate variances are needed. */

*** Model with equal variances ***

meqrlogit true se1 se2 sp1 sp2, nocons || studyid: sens spec, nocons
cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store A

*** Assumption of equal variances for logit sensitivities and logit
specificities may not be tenable so allow separate variances. ***

meqrlogit true se1 se2 sp1 sp2, nocons || studyid: se1 sp1, nocons cov(un)
|| studyid: se2 sp2, nocons cov(un) binomial(n) refineopts(iterate(3))
intpoints(5) variance nolr

estimates store B

lrtest A B
```

Stata output for model with equal variances

```
. meqrlogit true se1 se2 sp1 sp2, nocons || studyid: sens spec, nocons cov(un) ///
> binomial(n)  refineopts(iterate(3)) intpoints(5) variance nolr

Refining starting values:

Iteration 0:   log likelihood =  -276.0889
Iteration 1:   log likelihood = -271.85015  (not concave)
Iteration 2:   log likelihood = -267.86572
Iteration 3:   log likelihood = -267.11828

Performing gradient-based optimization:

Iteration 0:   log likelihood = -267.11828
Iteration 1:   log likelihood = -267.01518
Iteration 2:   log likelihood = -267.01396
Iteration 3:   log likelihood = -267.01396

Mixed-effects logistic regression               Number of obs     =         74
Binomial variable: n
Group variable: studyid                         Number of groups  =         37

                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2

Integration points =   5                        Wald chi2(4)      =     699.17
Log likelihood = -267.01396                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        true |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         se1 |  -.0965806   .2205958    -0.44   0.662    -.5289404    .3357792
         se2 |    .866114   .1210208     7.16   0.000     .6289175     1.10331
         sp1 |   3.447135   .2993556    11.52   0.000     2.860409    4.033861
         sp2 |   3.016265    .162807    18.53   0.000     2.697169     3.33536
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters |   Estimate   Std. Err.     [95% Conf. Interval]
----------------------------+-------------------------------------------------
studyid: Unstructured       |
                  var(sens) |   .3607972    .102521       .206725    .6296993
                  var(spec) |   .5449808   .1824097      .2827983    1.050233
             cov(sens,spec) |  -.1963092   .0987386     -.3898334    -.002785
------------------------------------------------------------------------------
```

Stata output for model with unequal variances

```
. meqrlogit true se1 se2 sp1 sp2, nocons || studyid: se1 sp1, nocons cov(un) ///
> || studyid: se2 sp2, nocons cov(un) binomial(n)  refineopts(iterate(3)) intpoints(5) variance nolr

Refining starting values:

Iteration 0:   log likelihood = -276.25889  (not concave)
Iteration 1:   log likelihood = -269.07163
Iteration 2:   log likelihood =  -264.0156
Iteration 3:   log likelihood = -263.00155

Performing gradient-based optimization:

Iteration 0:   log likelihood = -263.00155
Iteration 1:   log likelihood = -262.91643
Iteration 2:   log likelihood = -262.91611
Iteration 3:   log likelihood = -262.91611

Mixed-effects logistic regression               Number of obs     =         74
Binomial variable: n
Group variable: studyid                         Number of groups  =         37

                                                Obs per group:
                                                              min =          2
                                                              avg =        2.0
                                                              max =          2

Integration points =   5                        Wald chi2(4)      =     669.25
Log likelihood = -262.91611                     Prob > chi2       =     0.0000

------------------------------------------------------------------------------
        true |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         se1 |  -.0777962   .0937993    -0.83   0.407    -.2616395    .1060472
         se2 |   .8725146   .1377378     6.33   0.000     .6025534    1.142476
         sp1 |   3.458744   .2918493    11.85   0.000      2.88673    4.030758
         sp2 |   3.014642   .1655438    18.21   0.000     2.690182    3.339102
------------------------------------------------------------------------------

------------------------------------------------------------------------------
  Random-effects Parameters |   Estimate   Std. Err.     [95% Conf. Interval]
----------------------------+-------------------------------------------------
studyid: Unstructured       |
                  var(se1) |   .0428519   .0331124      .0094239    .1948532
                  var(sp1) |    .483393   .3237828      .1300633    1.796578
               cov(se1,sp1) |  -.0625768   .0793047     -.2180112    .0928576
----------------------------+-------------------------------------------------
studyid: Unstructured       |
                  var(se2) |   .4829672   .1529674      .2596101    .8984911
                  var(sp2) |   .5660857   .2185455      .2656217    1.206426
               cov(se2,sp2) |   -.232604   .1302635     -.4879159    .0227078
------------------------------------------------------------------------------
```

Likelihood ratio test comparing the two models

```
. lrtest A B

Likelihood-ratio test                                 LR chi2(3)  =      8.20
(Assumption: A nested in B)                           Prob > chi2 =    0.0421
```

Appendix 11: Bivariate meta-regression using meqrlogit (or xtmelogit) in Stata – comparison of CT and MRI for CAD

```
/* Set your working directory to the appropriate drive where you saved the
file schuetz.csv. Replace "U:\Handbook 2020" with the appropriate path for
you. */
cd "U:\Handbook 2020"
**************** 1. IMPORT DATA ***********************************
*** Read in the data from the .csv file. ***
insheet using "schuetz.csv", comma clear
*** Produce a summary of the dataset to check data import was ok. ***
describe


*************** 2. SET UP THE DATA BEFORE THE META-ANALYSIS **************
/* Generate 5 new variables of type long. We need these before we can
reshape the data.
•     n1 is number diseased
•     n0 is number without disease
•     true1 is number of true positives
•     true0 is the number of true negatives
•     study is the unique identifier for each study (and test if a study
evaluated more than one test). _n will generate a sequence of numbers. */
gen long n1=tp+fn
gen long n0=fp+tn
gen long true1=tp
gen long true0=tn
gen long recordid= _n
*** Reshape the data from wide to long format. ***
reshape long n true, i(recordid) j(sens)
*** Sort data by study and test variables to cluster the records per study
together. ***
gen byte spec=1-sens
sort study_id test


********** 3. TEST COMPARISON - META-REGRESSION APPROACH ****************
*** Create dummy variables for the covariate test. ***
gen seCT=0
gen spCT=0
gen seMRI=0
gen spMRI=0
replace seCT=1 if test=="CT" & sens==1
```

```
replace spCT=1 if test=="CT" & spec==1

replace seMRI=1 if test=="MRI" & sens==1

replace spMRI=1 if test=="MRI" & spec==1


*** Meta-analysis of CT ***

meqrlogit true sens spec if test=="CT", nocons ||study_id: sens spec,
nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance

*** Meta-analysis of MRI ***

meqrlogit true sens spec if test=="MRI", nocons ||study_id: sens spec,
nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance


/*** PART I: ANALYSES WITH COVARIATE TEST ASSUMING EQUAL VARIANCES ***/

*** Fit the model without the covariate. ***

meqrlogit true sens spec, nocons ||study_id: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

*** Store the estimates of the log likelihood from the model above for
doing the likelihood ratio test later. ***

estimates store A

*** Add covariate terms to the model for both logit sensitivity and logit
specificity. This model assumes equal variances for both tests. ***

meqrlogit true seCT seMRI spCT spMRI, nocons ||study_id: sens spec, nocons
cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store B

/* Perform a likelihood ratio test comparing the model (A) without
covariate with the model (B) that includes the covariate test and assumes
equal variances for each test. Use the stored values in A and B. */

lrtest A B

*** Assume sensitivity is the same for CT and MRI but allow specificity to
vary with test type. ***

meqrlogit true sens spCT spMRI, nocons ||study_id: sens spec, nocons
cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store C

/* Perform a likelihood ratio test comparing the model (C) that assumes
sensitivity is the same for CT and MRI but allows specificity to vary with
test type with the model (B) that allows both sensitivity and specificity
to vary with test. Use the stored values in B and C. */

lrtest B C

*** Assume specificity is the same for CT and MRI but allow sensitivity to
vary with test type. ***

meqrlogit true seCT seMRI spec, nocons ||study_id: sens spec, nocons
cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr

estimates store D

/* Perform a likelihood ratio test comparing the model (D) that assumes
specificity is the same for CT and MRI but allows sensitivity to vary with
test type with the model (B) that allows both sensitivity and specificity
to vary with test. Use the stored values in B and D. */
```

```
lrtest B D


/*** PART II: ANALYSES WITH COVARIATE TEST AND UNEQUAL VARIANCES ***/

/* Having looked at the variances assumption of equal variances may not be
appropriate. There was a marked difference in the variances for the logit
specificities in the 2 meta-analyses so let's fit a model with separate
variances for the logits for each test. */

*** Fit model (E) with covariate test and separate variances for the tests.
***

meqrlogit true seCT seMRI spCT spMRI, nocons ||study_id: seCT spCT, nocons
cov(un) ||study_id: seMRI spMRI, nocons cov(un) binomial(n)
refineopts(iterate(3)) intpoints(5) variance nolr

estimates store E

/* Perform a likelihood ratio test comparing the model (B) that assumes
equal variances with the model (E) that allows for separate variances for
each test. Use the stored values in B and E. */

lrtest B E

/* Perform a likelihood ratio test comparing the model (A) without
covariate with the model (E) that includes the covariate test and allows
for separate variances for each test. Use the stored values in A and E. */

lrtest A E

/* To find the covariance between the estimated mean logit sensitivity and
mean logit specificity for each test, display contents of the variance-
covariance matrix. */

matrix list e(V)

/*** The nlcom command in Stata post-estimation of the regression model.
The nlcom command uses nonlinear combinations of the parameter estimates to
compute point estimates and their standard errors are computed using the
delta method. ***/

*** To compute absolute differences ***

nlcom diff_sensitivity: invlogit(_b[seCT])-invlogit(_b[seMRI])

nlcom diff_specificity: invlogit(_b[spCT])-invlogit(_b[spMRI])

*** To compute ratios ***

nlcom log_relative_sensitivity: log(invlogit(_b[seCT]))-
log(invlogit(_b[seMRI]))

nlcom log_relative_specificity: log(invlogit(_b[spCT]))-
log(invlogit(_b[spMRI]))

*** Delete the program from Stata's memory if it exists already. ***

capture program drop renamematrix

*** Rename the elements of the coefficient and variance matrices. ***

program define renamematrix, eclass

      matrix mb = e(b)

      matrix mv = e(V)

      matrix colnames mb = logitseCT:_cons logitseMRI:_cons logitspCT:_cons
      logitspMRI:_cons
```

```
        matrix colnames mv = logitseCT:_cons logitseMRI:_cons logitspCT:_cons
        logitspMRI:_cons

        matrix rownames mv = logitseCT:_cons logitseMRI:_cons logitspCT:_cons
        logitspMRI:_cons

        ereturn post mb mv

end

*** Run the program. ***

renamematrix


*** Display summary estimates by taking the inverse logits of the mean
logit sensitivity and mean logit specificity for each test. ***

_diparm logitseCT, label(Sensitivity CT) invlogit

_diparm logitseMRI, label(Sensitivity MRI) invlogit

_diparm logitspCT, label(Specificity CT) invlogit

_diparm logitspMRI, label(Specificity MRI) invlogit

*** Display other summary estimates derived using functions of the mean
logit sensitivities and mean logit specificities. ***

_diparm logitseCT logitspCT, label(LR+ CT) ci(log)
function(invlogit(@1)/(1-invlogit(@2))) derivative(exp(@2-
@1)*invlogit(@1)^2/invlogit(@2) exp(@2)*invlogit(@1))

_diparm logitseMRI logitspMRI, label(LR+ MRI) ci(log)
function(invlogit(@1)/(1-invlogit(@2))) derivative(exp(@2-
@1)*invlogit(@1)^2/invlogit(@2) exp(@2)*invlogit(@1))

_diparm logitseCT logitspCT, label(LR- CT) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))

_diparm logitseMRI logitspMRI, label(LR- MRI) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))


**********Additional analyses **************************

*** Fit model (F) assuming same sensitivity but separate variances for the
tests. ***

meqrlogit true sens spCT spMRI, nocons || study: seCT spCT, nocons cov(un)
|| study: seMRI spMRI, nocons cov(un) binomial(n) refineopts(iterate(3))
intpoints(5) variance nolr

estimates store F

/* Perform a likelihood ratio test comparing the model (F) that assumes the
same sensitivity with the model (E) that allows for the effect of testtype.
Use the stored values in F and E. */

lrtest F E

*** Fit model (G) assuming same specificity but separate variances for the
tests. ***

meqrlogit true seCT seMRI spec, nocons || study: seCT spCT, nocons cov(un)

|| study: seMRI spMRI, nocons cov(un) binomial(n) refineopts(iterate(3))
intpoints(5) variance nolr
```

```
estimates store G

/* Perform a likelihood ratio test comparing the model (G) that assumes the
same specificity with the model (E) that allows for the effect of testtype.
Use the stored values in G and E. */

lrtest G E


********************* DIRECT COMPARISON ****************************

*** Fit the model for each test. ***

meqrlogit true sens spec if indirect==0 & test =="CT", nocons ||study_id:
sens spec, nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5)
stddev nolr gradient

meqrlogit true sens spec if indirect==0 & test=="MRI", nocons ||study_id:
sens spec, nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5)
stddev nolr gradient

*** Since the correlation parameter was poorly estimated for CT, fit a
model without this parameter by specifying an independent variance-
covariance structure (using option cov(ind)). ***

meqrlogit true sens spec if indirect==0 & test=="CT", nocons ||study_id:
sens spec, nocons cov(ind) binomial(n) refineopts(iterate(3)) intpoints(5)
stddev nolr gradient

*** Fit the model without the covariate. ***

*** Add gradient option to display gradient values. ***

meqrlogit true sens spec if indirect==0, nocons ||study_id: sens spec,
nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) stddev nolr
gradient

meqrlogit true sens spec if indirect==0, nocons ||study_id: sens spec,
nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5) variance
nolr gradient

*** Store the estimates of the log likelihood from the model above for
doing the likelihood ratio test later ***

estimates store X

*** Add covariate terms to the model for both logit sensitivity and logit
specificity. This model assumes equal variances for both tests. ***

meqrlogit true seCT seMRI spCT spMRI if indirect==0, nocons ||study_id:
sens spec, nocons cov(un) binomial(n) refineopts(iterate(3)) intpoints(5)
variance nolr gradient

estimates store Y

/* Perform a likelihood ratio test comparing the model (A) without
covariate with the model (B) that includes the covariate test and assumes
equal variances for each test. Use the stored values in A and B. */

lrtest X Y

*** Fit model (Z) with covariate test and separate variances for the tests.
***

meqrlogit true seCT seMRI spCT spMRI if indirect==0, nocons ||study_id:
seCT spCT, nocons cov(un) ||study_id: seMRI spMRI, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr gradient

estimates store Z

lrtest X Y
```

```
lrtest Y Z
```

Model with covariate and equal variances (B)

```
Mixed-effects logistic regression               Number of obs     =        216
Binomial variable: n
Group variable: study_id                        Number of groups  =        103

                                                Obs per group:
                                                              min =          2
                                                              avg =        2.1
                                                              max =          4

Integration points =   5                        Wald chi2(4)      =     710.49
Log likelihood = -476.06442                     Prob > chi2       =     0.0000
```

true	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
seCT	3.479028	.1552639	22.41	0.000	3.174716	3.783339
seMRI	2.176863	.2465182	8.83	0.000	1.693696	2.66003
spCT	1.917175	.1180104	16.25	0.000	1.685879	2.148471
spMRI	.8764553	.2121019	4.13	0.000	.4607433	1.292167

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
study_id: Unstructured				
var(sens)	.8874542	.2354463	.5276156	1.492706
var(spec)	.8629172	.1744708	.5805867	1.282541
cov(sens,spec)	.1911995	.1408996	-.0849587	.4673577

Model with covariate and unequal variances (E)

```
Mixed-effects logistic regression               Number of obs     =        216
Binomial variable: n
Group variable: study_id                        Number of groups  =        103

                                                Obs per group:
                                                              min =          2
                                                              avg =        2.1
                                                              max =          4

Integration points =   5                        Wald chi2(4)      =     752.83
Log likelihood = -471.22359                     Prob > chi2       =     0.0000
```

true	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
seCT	3.556827	.1742296	20.41	0.000	3.215343	3.898311
seMRI	1.966556	.1621878	12.13	0.000	1.648673	2.284438
spCT	1.932284	.1234624	15.65	0.000	1.690302	2.174266
spMRI	.8405046	.2423616	3.47	0.001	.3654847	1.315525

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
study_id: Unstructured				
var(seCT)	1.125342	.3199719	.6445545	1.964758
var(spCT)	.9009871	.1964317	.5876802	1.381326
cov(seCT,spCT)	.3205208	.178406	-.0291485	.6701901
study_id: Unstructured				
var(seMRI)	.111677	.1616459	.0065449	1.905582
var(spMRI)	.7268403	.3515486	.2816702	1.875586
cov(seMRI,spMRI)	-.1462591	.1381686	-.4170645	.1245463

Estimates of the summary sensitivities and specificities from the meta-analysis above

```
. _diparm logitseCT, label(Sensitivity CT) invlogit
Sensitivit~T │   .9722621   .0046987                      .9614076    .9801268

. _diparm logitseMRI, label(Sensitivity MRI) invlogit
Sensitivit~I │   .8772407   .0174659                      .8387117     .90758

. _diparm logitspCT, label(Specificity CT) invlogit
Specificit~T │    .873502   .0136421                      .8442639    .8979147

. _diparm logitspMRI, label(Specificity MRI) invlogit
Specificit~I │   .6985715   .0510339                      .5903675    .7884362
```

Appendix 12: Bivariate meta-regression using glmer in R – indirect comparison of CT and MRI for CAD

```
### Set your working directory to the appropriate drive where you saved the
file schuetz.csv. Replace "U:/Handbook 2020" with your path.

setwd("U:/Handbook 2020")

#################### 1. DATA IMPORT ############################

### Read in the data from the .csv file. ##

(X=read.csv("schuetz.csv"))

################## 2. PREPARE THE DATASETS ####################

### Install lme4 package if required (to run remove the # and select an
appropriate CRAN mirror).

# install.packages("lme4")

### Load the package lme4.

library(lme4)

### In order to specify the generalized linear model, first, we need to set
up the data.

### Generate 5 new variables of type long. We need these before we can
reshape the data.

# n1 is number diseased

# n0 is number without disease

# true1 is number of true positives

# true0 is the number of true negatives

# recordid is the unique identifier for each observation in the dataset

X$n1 <- X$TP+X$FN

X$n0 <- X$FP+X$TN

X$true1 <- X$TP

X$true0 <- X$TN

X$recordid <- 1:108

### Reshape the data from wide to long format. ###

Y = reshape(X, direction="long", varying=list(c("n1", "n0"),
      c("true1","true0")), timevar="sens", times=c(1,0), v.names=
      c("n","true"))

### Sort data by study to cluster the 2 records per study together. ###

Y = Y[order(Y$id),]

Y$spec<- 1-Y$sens

### Generate a seperate data frame for each test type.

Y.CT = Y[Y$Test=="CT",]

Y.MRI = Y[Y$Test=="MRI",]

########### 3. META-REGRESSION - USING GLMER WITH A COVARIATE ###########

### Meta-analysis of CT ###

(ma_CT = glmer(formula=cbind(true, n - true ) ~ 0 + sens + spec + (0+sens +
      spec|Study_ID), data=Y.CT, family=binomial))
```

```
### More detail about the parameter estimates can be obtained by using the
summary command.

summary(ma_CT)

### Meta-analysis of MRI ###

(ma_MRI = glmer(formula=cbind(true, n - true ) ~ 0 + sens + spec + (0+sens
      + spec|Study_ID), data=Y.MRI, family=binomial))

### More detail about the parameter estimates can be obtained by using the
summary command.

summary(ma_MRI)


### Fit the model without the covariate. ###

(A = glmer(formula=cbind(true, n - true ) ~ 0 + sens + spec + (0+sens +
      spec|Study_ID), data=Y, family=binomial))

### Add covariate terms to the model for both logit sensitivity and logit
specificity. This model assumes equal variances for both tests.

Y$CT <- as.numeric((Y$Test == "CT"))

Y$MRI <- as.numeric((Y$Test == "MRI"))

Y$seCT <- (Y$CT)*(Y$sens)

Y$seMRI <- (Y$MRI)*(Y$sens)

Y$spCT <- (Y$CT)*(Y$spec)

Y$spMRI <- (Y$MRI)*(Y$spec)

(B = glmer(formula=cbind(true, n - true) ~ 0 + seCT + seMRI + spCT + spMRI
      + (0+sens + spec|Study_ID), data=Y, family=binomial))

### The models can be formally compared using a LR test.

### Install lmtest package if required (to run remove the #).

# install.packages("lmtest")

### Load the package lmtest.

library(lmtest)

lrtest(A,B)

### Is there a statistically significant difference in sensitivity between
CT and MRI?

(C = glmer(formula=cbind(true, n - true) ~ 0 + sens + spCT + spMRI +

      (0+sens + spec|Study_ID), data=Y, family=binomial))

lrtest(B,C)

### Is there a statistically significant difference in specificity between
CT and MRI?

(D = glmer(formula=cbind(true, n - true) ~ 0 + seCT + seMRI + spec +

      (0+sens + spec|Study_ID), data=Y, family=binomial))

lrtest(B,D)

### Different variances for each test

(E = glmer(formula=cbind(true, n - true ) ~ 0 + seCT + seMRI + spCT + spMRI
      +(0 +seMRI + spMRI |Study_ID) +(0 +seCT + spCT |Study_ID), data=Y,
      family=binomial))

### More detail can be obtained by using the summary command.
```

```
summary(E)

lrtest(B,E)

lrtest(A,E)

### More detail about the parameter estimates can be obtained by using the
summary command.

summary(E)

### To obtain the between study covariance between logit sensitivity and
specificity for each test use

(vcovE = (summary(E))$vcov)

### The overall logit-sensitivity and -specificity are given in

cB = summary(B)$coefficients

cE = summary(E)$coefficients

### Therefore confidence intervals can be extracted with the following
function

sespci <-function(X) {
  cX = summary(X)$coefficients
  rows = nrow(cX)
  rnames = rownames(summary(X)$coefficients)
  lsesp = matrix(data=NA, nrow=rows, ncol=3)
  for(i in 1:rows) {
    lsesp[i,1] = cX[i,1]
    lsesp[i,2] = cX[i,1] - qnorm(0.975)*cX[i,2]
    lsesp[i,3] = cX[i,1] + qnorm(0.975)*cX[i,2]
  }
  rnames
  sesp = round(100 * plogis(lsesp), digits=1)
  row.names(sesp) <- c(rnames)
  sesp
}

sespci(ma_CT)

sespci(ma_MRI)

sespci(A)

sespci(B)

sespci(C)

sespci(D)

sespci(E)


### Standard errors and confidence intervals for absolute and relative
differences can be calculated using the delta method.

# This requires the package msm.

# install.packages("msm")
```

```
### Load the package msm.
library(msm)
seCT = cE[1,1]
seMRI = cE[2,1]
spCT = cE[3,1]
spMRI = cE[4,1]
se_cov = vcovE[1:2,1:2]
sp_cov = vcovE[3:4,3:4]


### Absolute difference
diff_Se=(exp(seCT)/(1+exp(seCT)))-(exp(seMRI)/(1+exp(seMRI)))
diff_Sp=(exp(spCT)/(1+exp(spCT)))-(exp(spMRI)/(1+exp(spMRI)))
se.diff_Se=deltamethod (~ (exp(x1)/(1+exp(x1)))-(exp(x2)/(1+exp(x2))),
            mean=c(seCT,seMRI), cov=se_cov)
se.diff_Sp=deltamethod (~ (exp(x1)/(1+exp(x1)))-(exp(x2)/(1+exp(x2))),
            mean=c(spCT,spMRI), cov=sp_cov)
data.frame(estimate=c(diff_Se, diff_Sp),
            lci=c(diff_Se-qnorm(0.975)*se.diff_Se, diff_Sp-
                  qnorm(0.975)*se.diff_Sp),
            uci=c(diff_Se+qnorm(0.975)*se.diff_Se, diff_Sp+
                  qnorm(0.975)*se.diff_Sp),
            row.names=c("Absolute difference Sens", "Absolute difference
                  Spec"))


### Relative difference
rel_Se = (exp(seCT)/(1+exp(seCT)))/(exp(seMRI)/(1+exp(seMRI)))
rel_Sp = (exp(spCT)/(1+exp(spCT)))/(exp(spMRI)/(1+exp(spMRI)))
se.rel_Se = deltamethod (~ log((exp(x1)/(1+exp(x1)))/
      (exp(x2)/(1+exp(x2)))), mean=c(seCT,seMRI), cov=se_cov)
se.rel_Sp = deltamethod (~ log((exp(x1)/(1+exp(x1)))/
      (exp(x2)/(1+exp(x2)))), mean=c(spCT,spMRI), cov=sp_cov)
data.frame(estimate=c(rel_Se, rel_Sp),
            lci=c(exp(log(rel_Se)-qnorm(0.975)*se.rel_Se), exp(
                  log(rel_Sp)-qnorm(0.975)*se.rel_Sp)),
            uci=c(exp(log(rel_Se)+qnorm(0.975)*se.rel_Se),
                  exp(log(rel_Sp)+qnorm(0.975)*se.rel_Sp)),
            row.names=c("Relative difference Sens", "Relative difference
                  Spec"))
```

Appendix 13: HSROC meta-regression using Proc NLMIXED in SAS

```
/* Import data. Note you should change the file path to where you saved the
RF.csv file. */
proc import out=nishimura_RF
      datafile='U:\Handbook 2020\RF.csv'
      dbms=csv
      replace;
      getnames=yes;
run;


proc print;
run;


/* Create separate records for the diseased and non-diseased groups in each
study. The variable dis is the disease indicator which takes the value 0.5
if diseased and -0.5 if not diseased. */
data nishimura_RF;
      set nishimura_RF;
      dis=0.5; pos=tp; n=tp+fn; output;
      dis=-0.5; pos=fp; n=tn+fp; output;
run;


/* Ensure that both records for a study are clustered together. */
proc sort data=nishimura_RF;
      by study_id dis;
run;


/******** META-REGRESSION FOR INVESTIGATIONS OF HETEROGENEITY ********/
/* Create two dummy variables to allow for the three RF measurement
methods. LA is the referent method. Delete the 2 studies that did not
report the method, and the study that used a different method. */
data nishimura_RF;
      set nishimura_RF;
      if method ne "ELISA" and method ne "Nephelometry" and
         method ne "LA" then delete;
      rfm1=0; rfm2=0;
      if method ="ELISA" then rfm1=1;
      if method ="Nephelometry" then rfm2=1;
run;
```

```
/* Include covariates to allow accuracy, threshold and shape to vary by
method. */
proc nlmixed data=nishimura_RF ecov cov;
      parms alpha=2 theta=0 beta=0 s2ua=1 s2ut=1 a1=0 a2=0 t1=0 t2=0 b1=0
      b2=0;
      logitp = (theta + ut + t1*rfm1 + t2*rfm2 + (alpha + ua + a1*rfm1 +
      a2*rfm2)*dis)* exp(-(beta + b1*rfm1 + b2*rfm2)*dis);
      p = exp(logitp)/(1+exp(logitp));
      model pos ~ binomial(n,p);
      random ut ua ~ normal([0,0],[s2ut,0,s2ua]) subject=study_id
      out=randeffs;
      /* parameter estimates for the methods of RF measurement */
      estimate 'alpha ELISA' alpha + a1;
      estimate 'theta ELISA' theta + t1;
      estimate 'beta ELISA' beta + b1;
      estimate 'alpha Nephelometry' alpha + a2;
      estimate 'theta Nephelometry' theta + t2;
      estimate 'beta Nephelometry' beta + b2;
run;


/* Simplify the model to assume that all three curves have the same
shape.*/
proc nlmixed data=nishimura_RF ecov cov;
      parms alpha=2 theta=0 beta=0 s2ua=1 s2ut=1 a1=0 a2=0 t1=0 t2=0 ;
      logitp = (theta + ut + t1*rfm1 + t2*rfm2 + (alpha + ua + a1*rfm1 +
      a2*rfm2)*dis)* exp(-(beta)*dis);
      p = exp(logitp)/(1+exp(logitp));
      model pos ~ binomial(n,p);
      random ut ua ~ normal([0,0],[s2ut,0,s2ua]) subject=study_id
      out=randeffs;
      estimate 'alpha ELISA' alpha + a1;
      estimate 'theta ELISA' theta + t1;
      estimate 'alpha Nephelometry' alpha + a2;
      estimate 'theta Nephelometry' theta + t2;
run;


/* Check assumption of normality for random effects. */
proc univariate data=randeffs plot normal;
  class effect;
  var estimate;
run;
```

```
/* This model assumes that all three curves have the same shape and
position. The position is the same because there are no covariates included
for accuracy. Comparison with the previous model allows us to test whether
accuracy varies by method. */
proc nlmixed data=nishimura_RF ecov cov;
      parms alpha=2 theta=0 beta=0 s2ua=1 s2ut=1 t1=0 t2=0;
      logitp = (theta + ut + t1*rfm1 + t2*rfm2 + (alpha + ua)*dis)*
      exp(-(beta)*dis);
      p = exp(logitp)/(1+exp(logitp));
      model pos ~ binomial(n,p);
      random ut ua ~ normal([0,0],[s2ut,0,s2ua]) subject=study_id
      out=randeffs;
run;
/* Check assumption of normality for random effects. */
proc univariate data=randeffs plot normal;
  class effect;
  var estimate;
run;
```

Appendix 14: Direct comparison of CT and MRI using glmer in R

```
### Set your working directory to the appropriate drive where you saved the
file schuetz.csv. Replace "U:/Handbook 2020" with your path.

setwd("U:/Handbook 2020")

################ 1. DATA IMPORT ###################################

### Read in the data from the .csv file. ###

(X=read.csv("schuetz direct only.csv"))

################# 2. PREPARE THE DATASETS ##############################

### Install lme4 package if required (to run remove the # and select an
appropriate CRAN mirror).

# install.packages("lme4")

### Load the package lme4.

library(lme4)

### In order to specify the generalized linear model, first, we need to set
up the data.

### Generate 5 new variables of type long. We need these before we can
reshape the data.

# n1 is number diseased

# n0 is number without disease

# true1 is number of true positives

# true0 is the number of true negatives

# recordid is the unique identifier for each observation in the dataset

X$n1 <- X$TP+X$FN

X$n0 <- X$FP+X$TN

X$true1 <- X$TP

X$true0 <- X$TN

X$recordid <- 1:10

### Reshape the data from wide to long format. ###

Y = reshape(X, direction="long", varying=list(c("n1", "n0"),
      c("true1","true0")), timevar="sens", times=c(1,0),
      v.names=c("n","true"))

### Sort data by study to cluster the 2 records per study together. ###

Y = Y[order(Y$id),]

Y$spec<- 1-Y$sens

### Generate a seperate data frame for each test type.

Y.CT = Y[Y$Test=="CT",]

Y.MRI = Y[Y$Test=="MRI",]

############## 3. META-REGRESSION - USING GLMER WITH A COVARIATE ##########

### Meta-analysis of CT ###

(ma_CT = glmer(formula=cbind(true, n - true ) ~ 0 + sens + spec + (0+sens +
      spec|Study_ID), data=Y.CT, family=binomial))

### Meta-analysis of MRI ###
```

```
(ma_MRI = glmer(formula=cbind(true, n - true) ~ 0 + sens + spec + (0+sens +
spec|Study_ID), data= Y.MRI, family=binomial))

### Fit the model without the covariate. ###

(A = glmer(formula=cbind(true, n - true) ~ 0 + sens + spec + (0+sens +
      spec|Study_ID), data=Y, family=binomial))


### Estimation of the model above may be unreliable - try different
optimization techniques to see if one performs better than the others and
does not give error or warning messages. You need the optimx and dfoptim
packages.

### Install optimx and dfoptim packages if required (to run remove the #
and select an appropriate CRAN mirror).

# install.packages("optimx")

# install.packages("dfoptim")

### Load the package optimx.

library(optimx)

### Load the package dfoptim.

library(dfoptim)

### Show available methods.

allFit(show.meth.tab=TRUE)

A.all <- allFit(A)

ss <- summary(A.all)

ss$which.OK            ## logical vector: which optimizers worked?

ss$llik                ## vector of log-likelihoods

ss$fixef               ## table of fixed effects

ss$sdcor               ## table of random effect SDs and correlations

### More detail can be obtained by using the summary command.

(A.all = summary(A.all))

(A = summary(A))


### Fit the model without the covariate again for performing the likelihood
ratio test to compare models with and without the covariate. ###

(A = glmer(formula=cbind(true, n - true) ~ 0 + sens + spec + (0+sens +
      spec|Study_ID), data=Y, family=binomial))

### Add covariate terms to the model for both logit sensitivity and logit
specificity. This model assumes equal variances for both tests.

Y$CT <- as.numeric((Y$Test == "CT"))

Y$MRI <- as.numeric((Y$Test == "MRI"))

Y$seCT <- (Y$CT)*(Y$sens)

Y$seMRI <- (Y$MRI)*(Y$sens)

Y$spCT <- (Y$CT)*(Y$spec)

Y$spMRI <- (Y$MRI)*(Y$spec)

(B = glmer(formula=cbind(true, n - true) ~ 0 + seCT + seMRI + spCT + spMRI
      + (0+sens + spec|Study_ID), data=Y, family=binomial))
```

```
### The models can be formally compared using the LR test.
# install.packages("lmtest")
library(lmtest)
lrtest(A,B)
### Is there a statistically significant difference in sensitivity between
CT and MRI?
(C = glmer(formula=cbind(true, n - true) ~ 0 + sens + spCT + spMRI +
      (0+sens + spec|Study_ID), data=Y, family=binomial))
lrtest(B,C)
### Is there a statistically significant difference in specificity between
CT and MRI?
(D = glmer(formula=cbind(true, n - true ) ~ 0 + seCT + seMRI + spec +
      (0+sens + spec|Study_ID), data=Y, family=binomial))
lrtest(B,D)
### Using different variances for each test
(E = glmer(formula=cbind(true, n - true) ~ 0 + seCT + seMRI + spCT + spMRI
      +(0 +seMRI + spMRI |Study_ID) +(0 +seCT + spCT |Study_ID), data=Y,
      family = binomial))
lrtest(B,E)
lrtest(A,E)
### More detail about the parameter estimates can be obtained by using the
summary command.
summary(E)
### To obtain the between study covariance between logit sensitivity and
specificity for each test use
(summary(E))$vcov
### The overall logit-sensitivity and -specificity are given in
cB = summary(B)$coefficients
cE = summary(E)$coefficients
### Therefore confidence intervals can be extracted with the following
function
sespci <-function(X) {
  cX = summary(X)$coefficients
  rows = nrow(cX)
  rnames = rownames(summary(X)$coefficients)
  lsesp = matrix(data=NA, nrow=rows, ncol=3)
  for(i in 1:rows) {
    lsesp[i,1] = cX[i,1]
    lsesp[i,2] = cX[i,1] - qnorm(0.975)*cX[i,2]
    lsesp[i,3] = cX[i,1] + qnorm(0.975)*cX[i,2]
  }
  rnames
  sesp = round(100 * plogis(lsesp), digits=1)
```

```
  row.names(sesp) <- c(rnames)
  sesp
}
sespci(ma_CT)
sespci(ma_MRI)
sespci(A)
sespci(B)
sespci(C)
sespci(D)
sespci(E)
```

Appendix 15: Fixed effect logistic regression using blogit command in Stata

```
/* Set your working directory to the appropriate drive where you saved the
file Xpert Ultra_sputum_composite reference standard.csv. Replace
"U:\Handbook 2020" with the appropriate path for you. */

cd "U:\Handbook 2020"

**************** 1. IMPORT DATA ***********************************

*** Read in the data from the .csv file. ***

insheet using "Xpert Ultra_sputum_composite reference standard.csv", comma
clear

*** Produce a summary of the dataset to check data import was ok. ***

describe

gen long n1=tp+fn

gen long n0=fp+tn

gen long true1=tp

gen long true0=tn

gen long recordid= _n

*** Reshape the data from wide to long format. ***

reshape long n true, i(recordid) j(sens)

*** Generate a new binary variable spec of type byte that takes the value 0
when sens=1 and vice versa. ***

gen byte spec=1-sens

*** Sort data to ensure studies are clustered together first by study. ***

sort studyid sens

* Fixed effect logistic regression

blogit true n sens spec, nocons

************** DISPLAY SUMMARY ESTIMATES ************************

*** Drop the program in case it is already in Stata's memory. ***

capture program drop renamematrix

/* Write a program renaming elements of the coefficient and variance
matrices. */

program define renamematrix, eclass

matrix mb = e(b)

matrix mv=e(V)

matrix colnames mb = logitse:_cons logitsp:_cons

matrix colnames mv = logitse:_cons logitsp:_cons

matrix rownames mv = logitse:_cons logitsp:_cons

ereturn post mb mv

end

*** Run the program. ***

renamematrix
```

```
*** Display the summary estimates for sensitivity and specificity. ***

_diparm logitse, label(Sensitivity) invlogit

_diparm logitsp, label(Specificity) invlogit

_diparm logitse logitsp, label(LR+ ) ci(log) function(invlogit(@1)/(1-
invlogit(@2))) derivative(exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))

_diparm logitse logitsp, label(LR- ) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))
```

Appendix 16: Simplifying the bivariate model and fitting the models using meqrlogit in Stata

```
/* Set your working directory to the appropriate drive where you saved the
file "anti-ccp.csv". Replace "U:\Handbook 2020" with your path. */

cd "U:\Handbook 2020"

*************** 1. IMPORT AND SET UP DATA *********

*** Read in the data from the .csv file. ***

insheet using "IOC for CBD stones.csv", comma clear

*** Produce a summary of the dataset to check data import was ok. ***

describe

/*Set up the data

Generate 5 new variables of type long. This is needed before reshaping the
data.

•     n1 is number diseased

•     n0 is number without disease

•     true1 is number of true positives

•     true0 is the number of true negatives

•     study is the unique identifier for each study. _n will generate a
sequence of numbers. */

gen long n1=tp+fn

gen long n0=fp+tn

gen long true1=tp

gen long true0=tn

gen long recordid= _n

*** Convert data from wide to long form. ***

reshape long n true, i(recordid) j(sens)

*** Generate a new binary variable spec of type byte that takes the value 0
when sens=1 and vice versa. ***

gen byte spec=1-sens

*** Sort data to ensure studies are clustered together first by study.  ***

sort studyid sens


****************** 2. META-ANALYSES OF IOC ***************************

*** Model A: unstructured variance-covariance matrix ***

meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) variance

*** Running it again simply to get the correlation instead of covariance of
the logits. ***

meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(un)
binomial(n) refineopts(iterate(3)) intpoints(5) stddev

*** Drop the program in case it is already in Stata's memory. ***
```

```
capture program drop renamematrix
/* Write a little program renaming elements of the coefficient and variance
matrices.   */
program define renamematrix, eclass
matrix mb = e(b)
matrix mv=e(V)
matrix colnames mb = logitse:_cons logitsp:_cons
matrix colnames mv = logitse:_cons logitsp:_cons
matrix rownames mv = logitse:_cons logitsp:_cons
ereturn post mb mv
end
*** Run the program. ***
renamematrix
*** Display the summary estimates for sensitivity and specificity. ***
_diparm logitse, label(Sensitivity) invlogit
_diparm logitsp, label(Specificity) invlogit
_diparm logitse logitsp, label(LR+ ) ci(log) function(invlogit(@1)/(1-
invlogit(@2))) derivative(exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))
_diparm logitse logitsp, label(LR- ) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))


*** Model B: exchangeable variance-covariance matrix ***
meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(exc)
binomial(n) refineopts(iterate(3)) intpoints(5) variance
meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(exc)
binomial(n) refineopts(iterate(3)) intpoints(5) stddev
*** Drop the program in case it is already in Stata's memory. ***
capture program drop renamematrix
/* Write a little program renaming elements of the coefficient and variance
matrices.   */
program define renamematrix, eclass
matrix mb = e(b)
matrix mv=e(V)
matrix colnames mb = logitse:_cons logitsp:_cons
matrix colnames mv = logitse:_cons logitsp:_cons
matrix rownames mv = logitse:_cons logitsp:_cons
ereturn post mb mv
end
*** Run the program. ***
renamematrix
*** Display the summary estimates for sensitivity and specificity. ***
```

```
_diparm logitse, label(Sensitivity) invlogit

_diparm logitsp, label(Specificity) invlogit

_diparm logitse logitsp, label(LR+ ) ci(log) function(invlogit(@1)/(1-
invlogit(@2))) derivative(exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))

_diparm logitse logitsp, label(LR- ) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))


*** Model C: independent variance-covariance matrix ***

meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(ind)
binomial(n) refineopts(iterate(3)) intpoints(5) variance

meqrlogit true sens spec, nocons || studyid: sens spec, nocons cov(ind)
binomial(n)  refineopts(iterate(3)) intpoints(5) stddev

*** Drop the program in case it is already in Stata's memory. ***

capture program drop renamematrix

/* Write a little program renaming elements of the coefficient and variance
matrices.   */

program define renamematrix, eclass

matrix mb = e(b)

matrix mv=e(V)

matrix colnames mb = logitse:_cons logitsp:_cons

matrix colnames mv = logitse:_cons logitsp:_cons

matrix rownames mv = logitse:_cons logitsp:_cons

ereturn post mb mv

end

*** Run the program. ***

renamematrix

*** Display the summary estimates for sensitivity and specificity. ***

_diparm logitse, label(Sensitivity) invlogit

_diparm logitsp, label(Specificity) invlogit

_diparm logitse logitsp, label(LR+ ) ci(log) function(invlogit(@1)/(1-
invlogit(@2))) derivative(exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))

_diparm logitse logitsp, label(LR- ) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))


*** Model D: fixed sensitivity but random effects included for specificity
***

meqrlogit true sens spec, nocons || studyid: spec, nocons cov(ind)
binomial(n) refineopts(iterate(3)) intpoints(5) variance

meqrlogit true sens spec, nocons || studyid: spec, nocons cov(ind)
binomial(n) refineopts(iterate(3)) intpoints(5) stddev

*** Drop the program in case it is already in Stata's memory. ***
```

```
capture program drop renamematrix
/* Write a little program renaming elements of the coefficient and variance
matrices.   */
program define renamematrix, eclass
matrix mb = e(b)
matrix mv=e(V)
matrix colnames mb = logitse:_cons logitsp:_cons
matrix colnames mv = logitse:_cons logitsp:_cons
matrix rownames mv = logitse:_cons logitsp:_cons
ereturn post mb mv
end
*** Run the program. ***
renamematrix
*** Display the summary estimates for sensitivity and specificity. ***
_diparm logitse, label(Sensitivity) invlogit
_diparm logitsp, label(Specificity) invlogit
_diparm logitse logitsp, label(LR+ ) ci(log) function(invlogit(@1)/(1-
invlogit(@2))) derivative(exp(@2-1)*invlogit(@1)^2/invlogit(@2)
exp(@2)*invlogit(@1))
_diparm logitse logitsp, label(LR- ) ci(log) function((1-
invlogit(@1))/invlogit(@2)) derivative(exp(-@1)*invlogit(@1)^2/invlogit(@2)
exp(-@1-@2)*invlogit(@1))
```

Appendix 17: Univariate meta-regression in Stata comparing nine first trimester serum test strategies for Down syndrome screening at a fixed 5% false positive rate

```
/* Set your working directory to the appropriate drive where you saved the
file "Nine T1 serum test strategies.csv". Replace "U:\Handbook 2020" with
your path. */
cd "U:\Handbook 2020"
*************** 1. IMPORT DATA ********************
*** Read in the data from the .csv file. ***
insheet using "Nine T1 serum test strategies.csv", comma clear
*** Produce a summary of the dataset to check data import was ok. ***
describe
/*** test is currently a string variable so for ease of selecting a test,
create a new numeric variable t and then list the contents of t. ***/
encode test, gen(t)
label list t
*** Tabulate the number of studies for each test. ***
table t

************** 2. SET UP THE DATA   ********************
*** Create new variables and reshape the data. ***
gen long n1=tp+fn
gen long n0=fp+tn
gen long true1=tp
gen long true0=tn
gen record_id = _n
reshape long n true, i(record_id) j(sens)
sort record_id sens
gen byte spec=1-sens
egen study= group(studyid)
*** Set up dummy variables for the covariate test. ***
gen se1=0
gen se2=0
gen se3=0
gen se4=0
gen se5=0
gen se6=0
gen se7=0
gen se8=0
gen se9=0
```

```
replace se1=1 if t==1 & sens==1
replace se2=1 if t==2 & sens==1
replace se3=1 if t==3 & sens==1
replace se4=1 if t==4 & sens==1
replace se5=1 if t==5 & sens==1
replace se6=1 if t==6 & sens==1
replace se7=1 if t==7 & sens==1
replace se8=1 if t==8 & sens==1
replace se9=1 if t==9 & sens==1


************** 3. UNIVARIATE META-REGRESSION    ******************
*** Model without covariate ***
meqrlogit true sens, nocons || study: sens, nocons cov(un) binomial(n)
refineopts(iterate(3)) intpoints(5) variance nolr
estimates store A
*** Model with covariate terms and equal variances ***
meqrlogit true se1 se2 se3 se4 se5 se6 se7 se8 se9, nocons || study: ///
sens, nocons binomial(n) refineopts(iterate(3)) intpoints(5) variance nolr
estimates store B
*** Model with covariate terms and separate variances ***
meqrlogit true  se1 se2 se3 se4 se5 se6 se7 se8 se9, nocons ///
|| study: se1, nocons || study: se2, nocons || study: se3, nocons ///
|| study: se4, nocons || study: se5, nocons || study: se6, nocons ///
|| study: se7, nocons || study: se8, nocons || study: se9, nocons ///
binomial(n) refineopts(iterate(6)) intpoints(1) variance nolr
estimates store C
*** Perform likelihood ratio tests to compare models. ***
lrtest A B
lrtest B C
lrtest A C
*** Display contents of the covariance matrix. ***
matrix list e(V)
/*** Write a little program renaming elements of the coefficient and
variance matrices. Drop the program first though in case it is already in
Stata's memory. ***/
capture program drop renamematrix
program define renamematrix, eclass
      matrix mb = e(b)
      matrix mv = e(V)
      matrix colnames mb = logitse1:_cons logitse2:_cons logitse3:_cons
      logitse4:_cons logitse5:_cons logitse6:_cons logitse7:_cons
      logitse8:_cons logitse9:_cons
```

```
        matrix colnames mv = logitse1:_cons logitse2:_cons logitse3:_cons
        logitse4:_cons logitse5:_cons logitse6:_cons logitse7:_cons
        logitse8:_cons logitse9:_cons

        matrix rownames mv = logitse1:_cons logitse2:_cons logitse3:_cons
        logitse4:_cons logitse5:_cons logitse6:_cons logitse7:_cons
        logitse8:_cons logitse9:_cons

        ereturn post mb mv

end

*** Run the program. ***

renamematrix

*** Display summary points by taking the inverse logits of the mean logit
sensitivity for each type. ***

_diparm logitse1, label(Sensitivity 1) invlogit

_diparm logitse2, label(Sensitivity 2) invlogit

_diparm logitse3, label(Sensitivity 3) invlogit

_diparm logitse4, label(Sensitivity 4) invlogit

_diparm logitse5, label(Sensitivity 5) invlogit

_diparm logitse6, label(Sensitivity 6) invlogit

_diparm logitse7, label(Sensitivity 7) invlogit

_diparm logitse8, label(Sensitivity 8) invlogit

_diparm logitse9, label(Sensitivity 9) invlogit
```

Output of the model with a covariate and separate variances (C) is shown below.

```
. *** Model with covariate and separate variances ***
. meqrlogit true  se1 se2 se3 se4 se5 se6 se7 se8 se9, nocons ///
> || study: se1, nocons || study: se2, nocons || study: se3, nocons ///
> || study: se4, nocons || study: se5, nocons || study: se6, nocons ///
> || study: se7, nocons || study: se8, nocons || study: se9, nocons ///
> binomial(n)  refineopts(iterate(6)) intpoints(1) variance nolr

Refining starting values:

Iteration 0:   log likelihood = -35571.667
Iteration 1:   log likelihood = -35564.426  (not concave)
Iteration 2:   log likelihood = -35553.451  (not concave)
Iteration 3:   log likelihood = -35547.852
Iteration 4:   log likelihood = -35540.199
Iteration 5:   log likelihood = -35540.168
Iteration 6:   log likelihood = -35538.595

Performing gradient-based optimization:

Iteration 0:   log likelihood = -35538.595
Iteration 1:   log likelihood = -35537.134
Iteration 2:   log likelihood = -35536.135
Iteration 3:   log likelihood = -35536.069
Iteration 4:   log likelihood = -35536.069

Mixed-effects logistic regression               Number of obs     =         92
Binomial variable: n
Group variable: study                           Number of groups  =         22

                                                Obs per group:
                                                              min =          2
                                                              avg =        4.2
                                                              max =         14

Integration points =   1                        Wald chi2(9)      =     244.38
Log likelihood = -35536.069                     Prob > chi2       =     0.0000
```

true	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
se1	1.0629	.2661084	3.99	0.000	.5413372	1.584463
se2	.1917927	.1828995	1.05	0.294	-.1666838	.5502692
se3	.7663903	.066723	11.49	0.000	.6356157	.8971649
se4	1.053165	.2120418	4.97	0.000	.637571	1.46876
se5	1.166439	.1857759	6.28	0.000	.8023249	1.530553
se6	-.3199136	.1224223	-2.61	0.009	-.559857	-.0799702
se7	-.0224274	.2094067	-0.11	0.915	-.4328569	.3880021
se8	-1.082267	.219215	-4.94	0.000	-1.51192	-.6526129
se9	.0804179	.2682041	0.30	0.764	-.4452524	.6060882

Random-effects Parameters	Estimate	Std. Err.	[95% Conf.	Interval]
study: Identity var(se1)	1.88e-13	2.31e-07	0	.
study: Identity var(se2)	.072911	.0866067	.0071072	.7479721
study: Identity var(se3)	1.21e-17	8.43e-10	0	.
study: Identity var(se4)	7.03e-14	1.13e-07	0	.
study: Identity var(se5)	6.06e-13	6.24e-07	0	.
study: Identity var(se6)	.001514	.0291662	6.06e-20	3.78e+13
study: Identity var(se7)	.0316561	.0869512	.0001454	6.894324
study: Identity var(se8)	.079508	.1154448	.0046182	1.368821
study: Identity var(se9)	.1493525	.1591024	.0185116	1.204984

Appendix 18: Fitting the Jones multiple thresholds meta-analysis model using rjags

A18.1: JAGS model code for the extended (Box-Cox) version of the model

The JAGS model code in the main text (**Error! Reference source not found.**) is for the version of the Jones (2019) model with $g()$ set to $log()$. As noted in 11.7.2, a more flexible approach is to assume only that $g()$ is in the set of Box-Cox transformations and simultaneously estimate a transformation parameter λ. The JAGS model code to fit this extended version of the model is provided below. Note that it is identical to that in the main text (**Error! Reference source not found.**) except for the text highlighted with bold font.

Warning: Mixing of chains may be poor for this extended version of the model, requiring longer simulation runs for robust parameter estimation. You must check convergence and other diagnostics carefully.

```
model{
  #=== LIKELIHOOD ===#
  for(i in 1:I){
    for(j in 1:2){
     for(t in 1:T[i]){
        x[i,j,t] ~ dbin(p[i,j,t], n[i,j,t])
                  }
      # DEFINE CONDITIONAL BINOMIAL ORDER PARAMETERS
            n[i,j,1] <- N[i,j]
            for(t in 2:T[i]){
            n[i,j,t] <- x[i,j,t-1]
            }
      # DEFINE CONDITIONAL BINOMIAL PROBABILITIES
            # IN TERMS OF FPR AND TPR
            p[i,j,1] <- pr[i,j,1]
            for(t in 2:T[i]){
            p[i,j,t] <-  pr[i,j,t] / pr[i,j,t-1]
            }
      # === MODEL FOR STUDY-LEVEL LOGIT(FPR) AND LOGIT(TPR) === #
      for(t in 1:T[i]){
            d[i,j,t] <- (mu[i,j] - q[i,t]) / s[i,j]
            pr[i,j,t] <- ilogit(d[i,j,t])
            }
      }
      # === BOX-COX TRANSFORMATION OF THRESHOLD === #
      for(t in 1:T[i]){
```

```
            q[i, t] <- (( pow(C[i,t], lambda) - 1 ) / lambda )*(1 -
            equals(lambda, 0)) +  log(C[i,t])*equals(lambda, 0)
      }

# === 4 SETS OF RANDOM EFFECTS ACROSS STUDIES === #
# COVARIANCE STRUCTURE AS DESCRIBED IN JONES 2019
mu[i,1] ~ dnorm(m_mu[1], prec_mu[1])

mu[i,2] ~ dnorm(cond_mean_mu[i], cond_prec_mu)
cond_mean_mu[i] <- m_mu[2] + (rho_mu*tau_mu[2]/tau_mu[1])*(mu[i,1] -
m_mu[1])

for(j in 1:2){
      cond_mean_s[i,j] <- m_sigma[j] +
(rho_mu_sigma*tau_sigma[j]/tau_mu[j])*(mu[i,j] - m_mu[j])
      logs[i,j] ~ dnorm(cond_mean_s[i,j], cond_prec_s[j])
      s[i,j] <- exp(logs[i,j])
    }
 }
# DEFINE PRECISION PARAMETERS FOR CONDITIONAL NORMAL DISTRIBUTIONS
  cond_var_mu <-  (1- pow(rho_mu,2))*pow(tau_mu[2], 2)
  cond_prec_mu <- 1/cond_var_mu
  for(j in 1:2){
    cond_var_s[j]<-  (1- pow(rho_mu_sigma,2))*pow(tau_sigma[j], 2)
    cond_prec_s[j] <- 1/cond_var_s[j]
  }

#=== HYPER PRIOR DISTRIBUTIONS ===#
# BOX-COX TRANSFORMATION PARAMETER:
lambda ~ dunif(-3, 3)
for(j in 1:2){
    # MEAN LOCATION PARAMETERS OF UNDERLYING LOGISTIC DISTRIBUTIONS
    m_mu[j]  ~ dnorm(0, 0.001)
    # MEAN LOG(SCALE) PARAMETERS OF UNDERLYING LOGISTIC DISTRIBUTIONS
    m_sigma[j] ~ dnorm(0, 0.001)
    # BETWEEN-STUDY STANDARD DEVIATION OF LOCATION PARAMETERS
    tau_mu[j] ~ dunif(0,5)
    # BETWEEN-STUDY STANDARD DEVIATION OF LOG(SCALE) PARAMETERS
    tau_sigma[j] ~ dunif(0,5)
    prec_mu[j] <- pow(tau_mu[j], -2)
    prec_sigma[j] <- pow(tau_sigma[j], -2)
```

```
  }
  # BETWEEN-STUDY CORRELATIONS
  rho_mu ~ dunif(-1,1)
  rho_mu_sigma ~ dunif(-1,1)
}
```

A18.2: rjags code to fit the model to the FeNo data

The "basic" (with $g()$ set to $log()$) version of the Jones (2019) model was demonstrated using the FeNO dataset in Section **Error! Reference source not found.**. Here we present the R code used, which is an adapted version of the more general code provided in Appendix 6. Please refer to Appendix 6 for general details of model fitting using rjags.

The data set in "wide" format (i.e. with one row per study, *i*) is provided in file `feno_wide.csv`. Here:

- The column headed `T` denotes the total number of thresholds reported at in each study. The first study (Arora 2006) reports at the most thresholds (19).
- Columns headed `N1` and `N2` denote the number of individuals without and with the target condition respectively.
- Columns headed `C1-C19` denote the numerical threshold values reported at in each study.
- Columns headed `tp10-tp19` denote the number of individuals with the target condition who tested positive at each threshold, i.e. numbers of true positives.
- Columns headed `fp1-fp19` denote the number of individuals without the target condition who tested positive at each threshold. i.e. numbers of false positives.

Note that values of `C`, `tp` and `fp` are set to missing ("`NA`") beyond the maximum number of thresholds reported at in each study. For example, the second study (El Halawini 2003) reports at 4 thresholds. Therefore, `C5-C19`, `tp5-tp19` and `fp5-fp19` are all set to `NA` for this study.

We load the data into R and extract the data items required for the JAGS model. As described in Section **Error! Reference source not found.**, we format all true positives and false positives as a single data array, `x`.

```
fenodata <- read.csv("feno_wide.csv", header = T)
I <- length(fenodata$study_id)
T <- fenodata$T
N <- fenodata[, grepl("N", names(fenodata))]
C <- fenodata[, grepl("C", names(fenodata))]
tp <- fenodata[, grepl("tp", names(fenodata))]
fp <- fenodata[, grepl("fp", names(fenodata))]
# Format numbers of false and true positives as an array, x,
# to pass to JAGS:
x <- array(NA, c(I, 2, max(T)))
```

```
for(i in 1:I){
  for(t in 1:max(T)){
    x[i, 1, t] <- fp[i,t]
    x[i, 2, t] <- tp[i,t]
  }
}
```

We then place all data in a list.

```
dataList = list(I = I, T = T, N = N, C = C, x = x)
```

We set initial values for the means and standard deviations of each of the four sets of random effects. It may be helpful to also set initial values for other parameters, for example any between-study correlation parameters (e.g. `rho_mu`).

Note: If running the extended version of the model code, with unknown Box-Cox transformation parameter λ (A18.1), you will need to also set reasonable initial values for the parameter `lambda` (which was assigned a vague Uniform(-3, 3) prior distribution in the model code, A18.1). In determining what might be reasonable initial values, note that $\lambda = 1$ corresponds to underlying continuous test results having a symmetrical (logistic) distribution, while $\lambda = 0$ corresponds to the degree of right-skew of a log-logistic distribution.

```
# Initial values for 3 chains:
myinits1 = list(
  m_mu = c(3, 5), m_sigma = c(log(1), log(1)),
  tau_mu = c(1, 1), tau_sigma = c(1,1)
)
myinits2 = list(
  m_mu = c(0, 6), m_sigma = c(log(2), log(2)),
  tau_mu = c(0.5, 0.5), tau_sigma = c(0.5,0.5)
)
myinits3 = list(
  m_mu = c(2, 4), m_sigma = c(log(5), log(5)),
  tau_mu = c(0.1, 0.1), tau_sigma = c(0.1,0.1)
)
initsList = list(myinits1, myinits2, myinits3)
```

After loading the relevant packages, and setting the number of chains, we compile the model and run the burn-in.

As in Appendix 6, the rjags model should be saved in a text file in the current R working directory. The code below assumes that the model (i.e. the text in **Error! Reference source not found.**) has been saved as “jones_model_log.txt”, where “log” refers to the choice of $g()$. If running the extended version of the model code, with unknown Box-Cox

transformation parameter λ, we would recommend saving this (i.e. the text in A18.1) with a different name for clarity. Remember to then edit the "`jags.model`" line in the following R code, in order to call the correct model.

```
library(rjags)
library(mcmcplots)
n.chains <- 3
jagsModel = jags.model("jones_model_log.txt",
                       data=dataList, inits=initsList,
      n.chains=n.chains)
update(jagsModel,n.iter=10000)
```

We next specify the parameters to be monitored, the number of iterations and any thinning of chains, and run the model.

Note: if running the extended version of the model (A18.1), it is important to also monitor the transformation parameter, `lambda`. This extended version of the model tends to have worse mixing of MCMC chains, so the model may need to be run for considerably longer and with the 'thinning' option set to a higher value.

```
parameters = c("m_mu", "m_sigma", "tau_mu", "tau_sigma")
n.iter <- 100000
thin <- 20
# Run the model:
output =    coda.samples(jagsModel, variable.names=parameters, n.iter=
n.iter, thin = thin)
```

As always with Bayesian modelling, it is crucial to assess convergence and mixing. Refer to Appendix 6 for more detail.

```
# Check the MCMC process has converged:
mcmcplot(output)
gelman.diag(output)
# summary statistics
summary(output)
# Posterior density plots
denplot(output)
```

A18.3: R code for post processing of the coda

Summary estimates of the TPR and FPR (or equivalently the sensitivity and specificity) across a range of thresholds of interest are obtained by evaluating equation 11.9 at each iteration of the MCMC simulation. This can be done within the JAGS model code or by post processing of the 'coda' in R as follows.

First extract the simulated chains from the posterior distribution of the relevant parameters, i.e. $m_{\mu j}, m_{\sigma j}$.

Note: If running the extended version of the model (A18.1) you will also need to extract the simulated values of λ (`lambda`).

```
nsims = n.chains*n.iter/thin     # total number of simulations
```

```
m_mu <- m_sigma <- matrix(NA, nsims, 2)
m_mu[,1] <- unlist(output[, "m_mu[1]"])
m_mu[,2] <- unlist(output[, "m_mu[2]"])
m_sigma[,1] <- unlist(output[, "m_sigma[1]"])
m_sigma[,2] <- unlist(output[, "m_sigma[2]"])
```

Specify all thresholds for which you would like to calculate the summary TPR and FPR. For example, here we specify thresholds = 5, 6, 7, ..., 100

```
thres <- seq(5, 100)
nthres <- length(thres)
```

For each threshold of interest, and at each iteration $k = 1, \ldots nsims$ of the MCMC chains, we then calculate the FPR and TPR, by evaluating equation 11.9:

```
logit_pr <- pr <- array(NA, c(nsims, 2, nthres ))
for(k in 1:nsims){
  for(j in 1:2){
    for(t in 1:nthres){
      # Evaluate logit(TPR) and logit(FPR) at each iteration, k
      logit_pr[k,j,t]<-(m_mu[k,j]-log(thres[t]))/exp(m_sigma[k,j])
      # Undo the logit transformation:
      pr[k,j,t] <- plogis(logit_pr[k,j,t])
    }
  }
}
```

Note: In the extended version of the model, equation 11.9 also depends on the transformation parameter, λ (`lambda`). For this model, replace the code in the box above with the following.

```
g <- matrix(NA, nsims, nthres)
logit_pr <- pr <- array(NA, c(nsims, 2, nthres ))
for(k in 1:nsims){
  for(t in 1:nthres){
    if(lambda[k] == 1){g[k,t] <- log(thres[t])}
    if(lambda[k] != 1){g[k,t] <- ((thres[t])^lambda[k] - 1)/lambda[k]    }
for(j in 1:2){
      logit_pr[k,j,t]<-(m_mu[k,j]-g[k,t])/exp(m_sigma[k,j])
      # Undo the logit transformation:
      pr[k,j,t] <- plogis(logit_pr[k,j,t])
    }
  }
}
```

We then summarize the chains using the posterior medians and 95% credible intervals.

```
summ_fpr <- summ_tpr <- matrix(NA, nthres, 3)
for(t in 1:nthres){
  summ_fpr[t,] <- quantile(pr[,1,t], c(0.5, 0.025, 0.975))
  summ_tpr[t,] <- quantile(pr[,2,t], c(0.5, 0.025, 0.975))
}
```

To view the summary TPR and FPR at each threshold, you might use `data.frame`, e.g.:

```
round(data.frame(thres, summ_tpr, summ_fpr), 2)
```

We see, for example, that at a threshold of 25, the summary TPR (= sensitivity) = 0.69 (95% credible interval 0.58, 0.78) and the summary FPR = 0.22 (95% credible interval = 0.16, 0.29). It is informative to plot the summary TPR and FPR, with 95% credible intervals, against threshold or log(threshold). For example, the following R code produces **Error! Reference source not found.**.

```
par(mar = c(5,5,1,15))
par(xpd = NA)
# Set limits of x and y-axis:
xlims <- c(log(min(thres)), log(max(thres)))
ylims <- c(0, 1)
cex <- 1.5
cex.data <- 1.2
coldis <- "red"
colnotdis <- "blue"
coldata <- "azure4"
# Plot summary TPR:
plot(log(thres), summ_tpr[,1], type = "l", xlim = xlims, ylim = ylims, col
= coldis, axes = F, xlab = "Threshold (ppm)",  ylab = "Probability
positive", cex.lab = cex, lwd = 3)
# Plot summary FPR:
lines(log(thres), summ_fpr[,1], type = "l", ylim = ylims, col = colnotdis,
lwd = 3)
# Label axes :
axis(2, cex.axis = cex)
labpoints = c(5, 7, 10, 14, 20, 30, 40, 60, 100)
axis(1, labels = labpoints, at = log(labpoints), cex.axis = cex)
# Plot observed data points and join points
# from the same study with a line:
obs.fpr <- fp/N[,1]
obs.tpr <- tp/N[,2]
for(i in 1:I){
```

```
  points(log(C[i,]), obs.tpr[i,], pch = 20, col = coldis,cex =
      cex.data)
  points(log(C[i,]), obs.fpr[i,], pch = 20, col = colnotdis,cex =
      cex.data)
  lines(log(C[i,]), obs.fpr[i,], col = coldata)
  lines(log(C[i,]), obs.tpr[i,], col = coldata)
}
# Plot shaded 95% credible intervals around summary TPR and FPR:
polygon( c(log(thres[1:nthres]), rev(log(thres[1:nthres]))),
         c(summ_tpr[,2], rev(summ_tpr[,3])),
         col=adjustcolor(coldis, alpha.f = 0.2),
         border = NA, xlab = "", ylab = "", main = "",ylim = ylims)
polygon( c(log(thres[1:nthres]), rev(log(thres[1:nthres]))),
         c(summ_fpr[,2], rev(summ_fpr[,3])),
         col=adjustcolor(colnotdis, alpha.f = 0.2),
         border = NA, xlab = "", ylab = "", main = "",ylim = ylims)
# Add figure legend:
xlegend <- log(70)
ylegend <- 1
legend(xlegend, ylegend,
       c("Observed diseased (TPR)", "Observed disease-free (FPR)",
         "Summary TPR", "Summary FPR" ),
       col = c(coldis, colnotdis, coldis, colnotdis),
       pch = c(20, 20, NA, NA), bty = "n", cex = cex,
       lty = c(NA, NA, 1, 1),
       pt.cex = c(cex, cex, NA, NA),
       lwd = c(NA, NA, 3,3)
)
```