贝叶斯期末作业

PS：第一问和第二问完整代码已经包含在Final_homework.R文件中，分析结果保存在SummaryInfo.csv文件中。由于代码包含绘图部分，故整体代码长度偏长，前半部分为题解代码。

> 假定现在有 3 位篮球运动员的投篮数据，包括每个人的投篮次数，以及投中的次数，而我们想要了解的是，这三位运动员投篮命中的概率是否存在差异。

第一问

> 用 R 编写代码，随机产生包含 3 位运动员数据的模拟数据集。其中第一位运动员的投篮次数在 40 到 50 次之间，且投篮命中的概率为0.1；第二位运动员的投篮次数在 50 到 60 次之间，且投篮命中的概率为 0.4；第三位运动员的投篮次数在 60 到 70 次之间，且投篮命中的概率为 0.7。（20%）

代码如下：

```r
# 1. 生成模拟数据
generate_data <- function() {
  # 第一位运动员
  n1 <- sample(40:50, 1)
  p1 <- 0.1

  # 第二位运动员
  n2 <- sample(50:60, 1)
  p2 <- 0.4

  # 第三位运动员
  n3 <- sample(60:70, 1)
  p3 <- 0.7

  # 创建数据框
  data <- data.frame(
    s = factor(rep(1:3, times = c(n1, n2, n3))),  # 运动员标识
    y = c(rbinom(n1, 1, p1), rbinom(n2, 1, p2), rbinom(n3, 1, p3))  # 投篮结果
  )
  return(data)
}

# 生成数据
set.seed(123)  # 设置随机种子以便结果可复现
sim_data <- generate_data()
print("模拟数据: ")
print(sim_data)
```

第二问

> 用 R 和 JAGS 编写分析投篮命中率差异的代码，所有代码需要包含在一个 r 程序文件里面，即无需援引其他 r 文件即可独立完成所需的数据分析的 r 程序文件。所需包含的内容参考 Jags-YdichXnomSsubj-MbernBeta.r 和 Jags-Ydich-XnomSsubj-MbernBetaExample.r。（60%）

代码如下：

```r
# 2. 贝叶斯分析代码
genMCMC <- function(data, numSavedSteps = 50000, saveName = NULL) {
  require(rjags)

  # 数据准备
  y <- data$y
  s <- as.numeric(data$s)
  if (any(y != 0 & y != 1)) {
    stop("All y values must be 0 or 1.")
  }
  Ntotal <- length(y)
```

```
  Nsubj <- length(unique(s))

  dataList <- list(
    y = y,
    s = s,
    Ntotal = Ntotal,
    Nsubj = Nsubj
  )

  # 模型定义
  modelString <- "
  model {
    for (i in 1:Ntotal) {
      y[i] ~ dbern(theta[s[i]])
    }
    for (sIdx in 1:Nsubj) {
      theta[sIdx] ~ dbeta(2, 2)  # 先验分布
    }
  }
  "
  writeLines(modelString, con = "TEMPmodel.txt")

  # 初始化MCMC链
  # Initial values of MCMC chains based on data:
  # Option 1: Use single initial value for all chains:
  #  thetaInit = rep(0,Nsubj)
  #  for ( sIdx in 1:Nsubj ) { # for each subject
  #    includeRows = ( s == sIdx ) # identify rows of this subject
  #    yThisSubj = y[includeRows]  # extract data of this subject
  #    thetaInit[sIdx] = sum(yThisSubj)/length(yThisSubj) # proportion
  #  }
  #  initsList = list( theta=thetaInit )
  # Option 2: Use function that generates random values near MLE:
  initsList = function() {
    thetaInit = rep(0,Nsubj)
    for ( sIdx in 1:Nsubj ) { # for each subject
      includeRows = ( s == sIdx ) # identify rows of this subject
      yThisSubj = y[includeRows]  # extract data of this subject
      resampledY = sample( yThisSubj , replace=TRUE ) # resample
      thetaInit[sIdx] = sum(resampledY)/length(resampledY)
    }
    thetaInit = 0.001+0.998*thetaInit # keep away from 0,1
    return( list( theta=thetaInit ) )
  }


  # 运行 MCMC
  parameters <- c("theta")
  adaptSteps <- 500
  burnInSteps <- 500
  nChains <- 4
  thinSteps <- 1
  nIter <- ceiling((numSavedSteps * thinSteps) / nChains)

  jagsModel <- jags.model("TEMPmodel.txt", data = dataList, inits = initsList,
                          n.chains = nChains, n.adapt = adaptSteps)
  update(jagsModel, n.iter = burnInSteps)
  codaSamples <- coda.samples(jagsModel, variable.names = parameters,
                              n.iter = nIter, thin = thinSteps)

  if (!is.null(saveName)) {
    save(codaSamples, file = paste(saveName, "Mcmc.Rdata", sep = ""))
  }
  return(codaSamples)
}


# 3. 分析模拟数据
codaSamples <- genMCMC(sim_data, numSavedSteps = 10000)  # 减少迭代次数以节省时间

# 4. 报告结果
summarizePost = function( paramSampleVec ,
                          compVal=NULL , ROPE=NULL , credMass=0.95 ) {
  meanParam = mean( paramSampleVec )
  medianParam = median( paramSampleVec )
  dres = density( paramSampleVec )
  modeParam = dres$x[which.max(dres$y)]
```

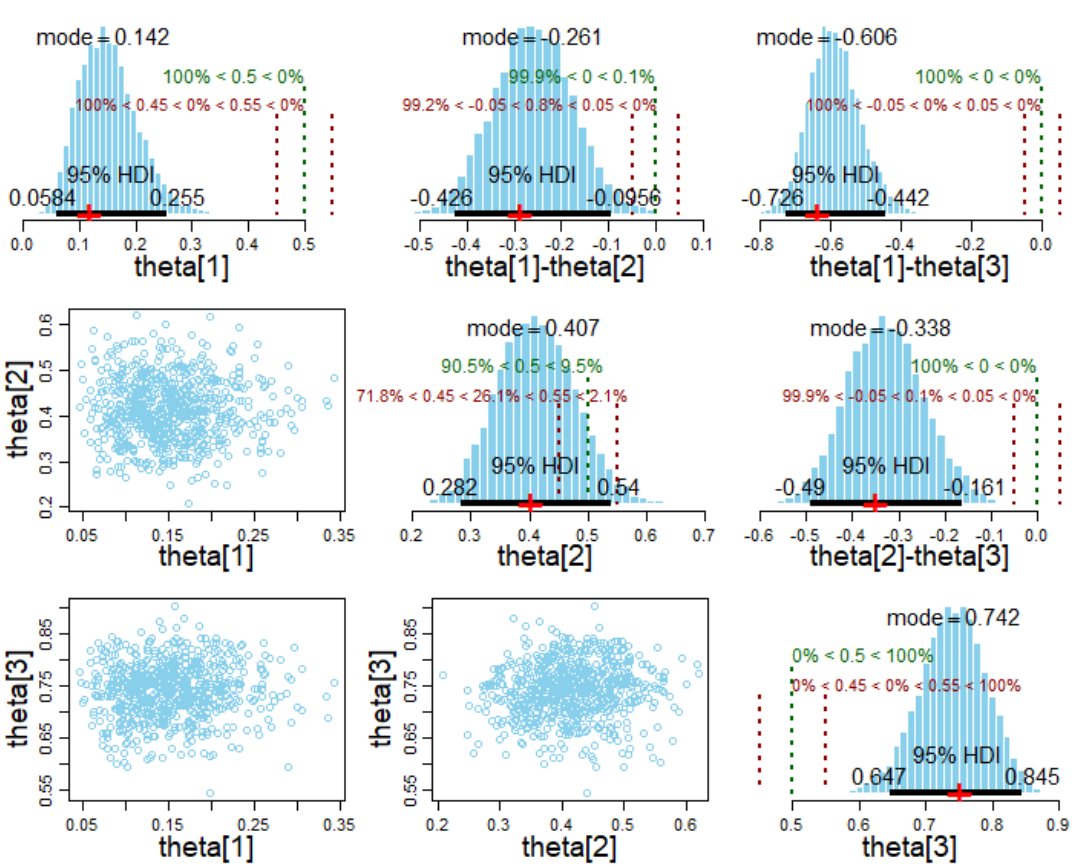
```
  mcmcEffSz = round( effectiveSize( paramSampleVec ) , 1 )
  names(mcmcEffSz) = NULL
  hdiLim = HDIofMCMC( paramSampleVec , credMass=credMass )
  if ( !is.null(compVal) ) {
    pcgtCompVal = ( 100 * sum( paramSampleVec > compVal )
                    / length( paramSampleVec ) )
  } else {
    compVal=NA
    pcgtCompVal=NA
  }
  if ( !is.null(ROPE) ) {
    pcltRope = ( 100 * sum( paramSampleVec < ROPE[1] )
                 / length( paramSampleVec ) )
    pcgtRope = ( 100 * sum( paramSampleVec > ROPE[2] )
                 / length( paramSampleVec ) )
    pcinRope = 100-(pcltRope+pcgtRope)
  } else {
    ROPE = c(NA,NA)
    pcltRope=NA
    pcgtRope=NA
    pcinRope=NA
  }
  return( c( Mean=meanParam , Median=medianParam , Mode=modeParam ,
             ESS=mcmcEffSz ,
             HDImass=credMass , HDIlow=hdiLim[1] , HDIhigh=hdiLim[2] ,
             CompVal=compVal , PcntGtCompVal=pcgtCompVal ,
             ROPElow=ROPE[1] , ROPEhigh=ROPE[2] ,
             PcntLtROPE=pcltRope , PcntInROPE=pcinRope , PcntGtROPE=pcgtRope ) )
}

#------------------------------------------------------------------------------
# Functions for computing limits of HDI's:

HDIofMCMC = function( sampleVec , credMass=0.95 ) {
  # Computes highest density interval from a sample of representative values,
  #   estimated as shortest credible interval.
  # Arguments:
  #   sampleVec
  #     is a vector of representative values from a probability distribution.
  #   credMass
  #     is a scalar between 0 and 1, indicating the mass within the credible
  #     interval that is to be estimated.
  # Value:
  #   HDIlim is a vector containing the limits of the HDI
  sortedPts = sort( sampleVec )
  ciIdxInc = ceiling( credMass * length( sortedPts ) )
  nCIs = length( sortedPts ) - ciIdxInc
  ciWidth = rep( 0 , nCIs )
  for ( i in 1:nCIs ) {
    ciWidth[ i ] = sortedPts[ i + ciIdxInc ] - sortedPts[ i ]
  }
  HDImin = sortedPts[ which.min( ciWidth ) ]
  HDImax = sortedPts[ which.min( ciWidth ) + ciIdxInc ]
  HDIlim = c( HDImin , HDImax )
  return( HDIlim )
}

HDIofICDF = function( ICDFname , credMass=0.95 , tol=1e-8 , ... ) {
  # Arguments:
  #   ICDFname is R's name for the inverse cumulative density function
  #     of the distribution.
  #   credMass is the desired mass of the HDI region.
  #   tol is passed to R's optimize function.
  # Return value:
  #   Highest density iterval (HDI) limits in a vector.
  # Example of use: For determining HDI of a beta(30,12) distribution, type
  #   HDIofICDF( qbeta , shape1 = 30 , shape2 = 12 )
  #   Notice that the parameters of the ICDFname must be explicitly named;
  #   e.g., HDIofICDF( qbeta , 30 , 12 ) does not work.
  # Adapted and corrected from Greg Snow's TeachingDemos package.
  incredMass =  1.0 - credMass
  intervalWidth = function( lowTailPr , ICDFname , credMass , ... ) {
    ICDFname( credMass + lowTailPr , ... ) - ICDFname( lowTailPr , ... )
  }
  optInfo = optimize( intervalWidth , c( 0 , incredMass ) , ICDFname=ICDFname ,
                      credMass=credMass , tol=tol , ... )
  HDIlowTailPr = optInfo$minimum
```

```
  return( c( ICDFname( HDIlowTailPr , ... ) ,
             ICDFname( credMass + HDIlowTailPr , ... ) ) )
}

smryMCMC <- function(codaSamples, compVal = 0.5, rope = NULL,
                     compValDiff = 0.0, ropeDiff = NULL, saveName = NULL) {
  mcmcMat <- as.matrix(codaSamples, chains = TRUE)
  Ntheta <- length(grep("theta", colnames(mcmcMat)))
  summaryInfo <- NULL
  rowIdx <- 0
  for (tIdx in 1:Ntheta) {
    parName <- paste0("theta[", tIdx, "]")
    summaryInfo <- rbind(summaryInfo,
                         summarizePost(mcmcMat[, parName], compVal = compVal, ROPE = rope))
    rowIdx <- rowIdx + 1
    rownames(summaryInfo)[rowIdx] <- parName
  }
  for (t1Idx in 1:(Ntheta - 1)) {
    for (t2Idx in (t1Idx + 1):Ntheta) {
      parName1 <- paste0("theta[", t1Idx, "]")
      parName2 <- paste0("theta[", t2Idx, "]")
      summaryInfo <- rbind(summaryInfo,
                           summarizePost(mcmcMat[, parName1] - mcmcMat[, parName2],
                                         compVal = compValDiff, ROPE = ropeDiff))
      rowIdx <- rowIdx + 1
      rownames(summaryInfo)[rowIdx] <- paste0(parName1, "-", parName2)
    }
  }
  if (!is.null(saveName)) {
    write.csv(summaryInfo, file = paste(saveName, "SummaryInfo.csv", sep = ""))
  }
  print(summaryInfo)
  return(summaryInfo)
}

summaryInfo = smryMCMC( codaSamples , compVal=0.5 , rope=c(0.45,0.55), saveName="")
```

第三问

> 使用以上程序对模拟数据进行分析，报告相应的贝叶斯数据分析结果。（20%）

	Mean	Median	Mode	ESS	HDImass	HDIlow	HDIhigh	CompVal	PcntGtCompVal	ROPElow	ROPEhigh	PcntLtROPE	PcntInROPE	PcntGtROPE
theta[1]	0.151360544443257	0.146810967070989	0.142396460096604	9455.9	0.95	0.0584131756505334	0.25450625544628	0.5	0	0.45	0.55	100	0	0
theta[2]	0.411201264567479	0.409785232761778	0.406912176098958	10000	0.95	0.282362330513333	0.539991943579457	0.5	9.47	0.45	0.55	71.83	26.09	2.08
theta[3]	0.739756823883261	0.741167080807465	0.74194588554659	10000	0.95	0.646588403472274	0.845344568098402	0.5	100	0.45	0.55	0	0.0499999999999972	99.95
theta[1]-theta[2]	-0.25984072	-0.260420648	-0.260692543	10000	0.95	-0.426381719	-0.09558354	0	0.12	NA	NA	NA	NA	NA
theta[1]-theta[3]	-0.588396279	-0.592120929	-0.606410981	10000	0.95	-0.725707438	-0.442074014	0	0	NA	NA	NA	NA	NA
theta[2]-theta[3]	-0.328555559	-0.330949677	-0.338269796	10000	0.95	-0.489609742	-0.161119207	0	0	NA	NA	NA	NA	NA



以上为使用MCMC方法进行分析的结果及相关统计图的绘制结果。其中，theta[1]、theta[2]、theta[3]分别代表三位运动员的投篮命中率。

- theta[1]-theta[2]：由表格结果可知，后验均值为 -0.2598，95% 可信区间为 [-0.4264, -0.0956]。由图可见数值0落在95%可信区间右侧。故可得参数 $\theta[1]$ 显著小于 $\theta[2]$。
- theta[1]-theta[3]：由表格结果可知，后验均值为 -0.5884，95% 可信区间为 [-0.7257, -0.4421]。由图可见数值0落在95%可信区间右侧。故可得参数 $\theta[1]$ 显著小于 $\theta[3]$。
- theta[2]-theta[3]：由表格结果可知，后验均值为 -0.3286，95% 可信区间为 [-0.4896, -0.1611]。由图可见数值0落在95%可信区间右侧。故可得参数 $\theta[2]$ 显著小于 $\theta[3]$。

因此，这三位运动员投篮命中的概率是否存在差异，有充分证据表明 $\theta[1] < \theta[2] < \theta[3]$。