```
######################################################################
##                                                                  ##
##  Title   : R Replication script 1 for the BJPS manuscript        ##
##            "Constitution and Causal Roles"                       ##
##  Authors: [anonymized]                                           ##
##  Version: 17/09/2019                                             ##
##                                                                  ##
######################################################################



# Required R packages
library(pcalg)
library(ggplot2)
library(reshape)
library(tikzDevice)


# Auxiliary Functions
sortString <- function(x){
  if(length(x)>0) {
  r <- strsplit(x, "\\|")
  r <- lapply(r, sort)
  lapply(r, paste0, collapse = "|")} else {
  x
  }
}

bidir <- function(x)  {
  r <- expand.grid(x,x)
  r <- r[apply(r, 1, function(x) all(!duplicated(x))),]
  resul <- vector("logical",nrow(r))
  for(i in 1:nrow(r)){
    resul[i] <-  identical(sortString(as.character(r[i,1])),sortString(as.character(r[i,2])))
  }
  r <- cbind(r,resul)
  r <- as.matrix(r[which(r[,3]==TRUE),])
  unique(as.vector(r[,1:2]))
}

# Define test variants
variants <- c("series1", "series2", "series3")

# Number of trials
n <- 1000

# Draw seeds for replication
suppressWarnings(RNGversion("3.6.0")) # enforcing sampling method from R 3.6
set.seed(46)
seeds.param <- sample(.Machine$integer.max, n)

# Initialize score lists
compare.variants <- score.list.storage <- data.storage <- analyses.storage <- vector("list", length(variants))

# Variant Loop
# ------------

for(m in 1:length(variants)){
  cat(m,  "variant","\n")

  variant <- variants[m]

  # Define true graph
  if(variant=="series1"||variant=="series2"){
  true.graph <- c("I|A", "I|B", "I|S", "A|B", "A|S", "A|G", "B|S", "B|G", "Z|A", "X|S")
  } else {
  true.graph <- c("I|A", "I|B", "I|S", "A|B", "A|S", "A|G", "B|S", "B|G", "Z|A", "X|S", "I|G", "G|S")
  }

  # Repetition Loop
  # -------------
  score.list.rep <- edges.list.rep <- data.list.rep <- analyses.rep <- vector("list", n)

  for(k in 1:n){
  cat(k, "\n")
  set.seed(seeds.param[k])

     # Sample error terms
     errorA <- rnorm(10000,0,sample(1:10,1))
     errorB <- rnorm(10000,0,sample(1:10,1))
     errorS <- rnorm(10000,0,sample(1:10,1))
     errorG <- rnorm(10000,0,sample(1:10,1))

     # Sample exogenous variables
     I <- sample(-5:5,1) + rnorm(10000,0,sample(1:10,1))
     X <- sample(-5:5,1) + rnorm(10000,0,sample(1:10,1))
     Z <- sample(-5:5,1) + rnorm(10000,0,sample(1:10,1))

     # Sample endogenous variables A and B
     A <- sample(-5:5,1)*I + sample(-5:5,1)*Z  + errorA
     B <- sample(-5:5,1)*I + sample(-5:5,1)*A  + errorB

     # Sample G
     if(variant=="series1"||variant=="series3"){ G <- sample(-5:5,1)*A + sample(-5:5,1)*B
     } else {
      G <- sample(-5:5,1)*A + sample(-5:5,1)*B  + errorG
     }

     # Sample S
     S <- sample(-5:5,1)*I + sample(-5:5,1)*A + sample(-5:5,1)*B + sample(-5:5,1)*X + errorS

     # Generate data list with all combinations of variable endogenous factors.
     data <- cbind(I,A,B,S,G,X,Z)

     # Store data
     data.list.rep[[k]] <- data
```

```
      # Run PC
      V <- colnames(data)
      analysis <- pc(suffStat = list(C = cor(data), n = nrow(data)),
                     indepTest = gaussCItest , alpha = 0.01, labels = V)
      analyses.rep[[k]] <- analysis

      # Build all edges in graph
      edgesGraph <- arrowsGraph <-  gsub(".weight",'',attr(unlist(analysis@graph@edgeData@data),"names"))
      edgesGraph <- unlist(sortString(edgesGraph))

      # Build all possible (un)directed edges
      allEdges <- combn(colnames(data),2)
      allEdges <- apply(allEdges , 2, function(x) paste0(x[1],"|",x[2]))
      allEdges <- unlist(sortString(allEdges))

      allArrows <- expand.grid(colnames(data),colnames(data))
      allArrows <- allArrows[apply(allArrows , 1, function(x) all(!duplicated(x))),]
      allArrows <- as.vector(apply(as.matrix(allArrows), 1, function(x) paste0(x[1],"|",x[2])))

      # Score undirected edges recovered
      score <- allEdges %in% edgesGraph
      score <- as.data.frame(matrix(score,nrow=1,ncol=length(allEdges),
             byrow = T,dimnames = list(1,allEdges)))

      # False positives/negatives in undirected edges/AdjCompletness (true edges in Graph/edges in true graph)/
      x <- unique(unlist(sortString(true.graph)))
      z <- length(setdiff(allEdges,unique(unlist(sortString(edgesGraph))))) # negative claims by Graph
      Adj.false.pos <- length(setdiff(unique(unlist(sortString(edgesGraph))), x))/length(unique(edgesGraph))
      AdjCompleteness <- length(intersect(unique(unlist(sortString(edgesGraph))),x))/length(x)
      score$Adj.false.pos <- Adj.false.pos
      score$AdjCompleteness <- AdjCompleteness

      # False positives/negatives in arrows/ArrCompleteness (Arr.true.pos/directed arrows in true graph)/
      x <- true.graph
      t <- length(setdiff(allArrows , arrowsGraph)) # negative claims by Graph
      z <- bidir(arrowsGraph) # bi- and undirectional edges in Graph
      dirArr <- setdiff(arrowsGraph,z) # directed edges in Graph
      Arr.false.pos <- length(union(setdiff(dirArr,x) ,
                                    setdiff(unlist(sortString(z)), unlist(sortString(x))))/
                              length(unique(edgesGraph))
      ArrCompleteness <- length(intersect(dirArr,x))/length(x)
      pathAG <- "A|G" %in% dirArr
      pathBG <- "B|G" %in% dirArr

      score$Arr.false.pos <- Arr.false.pos
      score$ArrCompleteness <- ArrCompleteness
      score$pathAG <- pathAG
      score$pathBG <- pathBG
      score.list.rep[[k]] <-  score

      } # End repetition loop

    # Overall scoring
    overall.score <- do.call(rbind,score.list.rep)
    # overall.score.withoutList <- overall.score[,-which(names(overall.score) %in% names(func.comb))]
    overall.ratio <-  matrix(1, ncol = ncol(overall.score), byrow = T,
            dimnames = list(1,names(overall.score)))


    for(i in 1:length(allEdges)){
        overall.ratio[1,i] <- length(which(overall.score[,i]))/nrow(overall.score)
    }
        overall.ratio[,c("Adj.false.pos")] <- mean(overall.score$Adj.false.pos)
        overall.ratio[,c("Arr.false.pos")] <- mean(overall.score$Arr.false.pos)
        overall.ratio[,c("AdjCompleteness")] <- mean(overall.score$AdjCompleteness)
        overall.ratio[,c("ArrCompleteness")] <- mean(overall.score$ArrCompleteness)
        overall.ratio[,c("pathAG")] <- length(which(overall.score[,c("pathAG")]))/nrow(overall.score)
        overall.ratio[,c("pathBG")] <- length(which(overall.score[,c("pathBG")]))/nrow(overall.score)

     # Store analyses, data, scores
     analyses.storage[[m]] <- analyses.rep
     data.storage[[m]] <-  data.list.rep
     score.list.storage[[m]] <- score.list.rep
     compare.variants[[m]] <- overall.ratio

}  # END variants loop

# Plot 1 (Figure 5)
# -------------------
selection <- lapply(compare.variants , function(x) x[,c("AdjCompleteness","ArrCompleteness",
                                                        "Adj.false.pos","Arr.false.pos", "pathAG", "pathBG")])
selection <- lapply(selection , setNames , c("edge.complete","orient.complete","edge.false.pos","orient.false.pos",
                                             "A->G", "B->G"))
final.score <-  as.data.frame(do.call(rbind,selection))
final.score$variants <- factor(variants , levels =variants)
k1 <- melt(final.score, id.vars = "variants")
colnames(k1) <- c("variants", "property", "value")
plot1 <- ggplot(k1, aes(x = variants , y = value,group = 1, fill = property)) +
  geom_bar(stat = "identity", position = position_dodge2(), width=.7) +
   scale_fill_grey(start = 0, end = .92)+
  theme_bw()+
  theme(legend.title=element_blank())+
  theme(plot.title = element_text(size = 9))+
  scale_x_discrete(name ="")

options(tz="CA")
tikz(file = "plot1.tex", width = 6, height = 2.2)
print(plot1)
dev.off()
```