```
######################################################################
##                                                                  ##
##  Title   : R Replication script 2 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)
library(plyr)

# 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]))
}

# Determine test variants
variants <- c("withoutAB", "withoutG")

# Define true graphs
true.withoutG <- c("I|A", "I|B", "I|S", "A|B", "A|S", "B|S", "Z|A", "X|S")
true.withoutAB <- c("I|G", "I|S", "G|S", "Z|G", "Z|S", "X|S")

# Number of trials
n <- 1000

# Draw seeds for replication
suppressWarnings(RNGversion("3.6.0")) # sampling method from R 3.6
set.seed(4615)
seeds.param <- sample(.Machine$integer.max, n)

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

# Data Loop
# ----------
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 apart from G and S
     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 and S
     G <- sample(-5:5,1)*A + sample(-5:5,1)*B
     S <- sample(-5:5,1)*I + sample(-5:5,1)*A + sample(-5:5,1)*B + sample(-5:5,1)*X  + errorS

     data.withoutAB <- cbind(I,G,S,X,Z)
     data.withoutG <- cbind(I,A,B,S,X,Z)

     # Store data
     data.storage[[k]] <- list(data.withoutAB , data.withoutG)

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

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

       variant <- variants[m]

       if(variant=="withoutAB") data <- data.withoutAB
       if(variant=="withoutG") data <- data.withoutG

       # Run PC
       V <- colnames(data)
       if(variant=="withoutAB"){
       analysis.withoutAB <- pc(suffStat = list(C = cor(data), n = nrow(data)),
                        indepTest = gaussCItest , alpha = 0.01, labels = V)

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

```
        } else {
        analysis.withoutG <- pc(suffStat = list(C = cor(data), n = nrow(data)),
                        indepTest = gaussCItest , alpha = 0.01, labels = V)
        edgesGraph <- arrowsGraph <-  gsub(".weight",'',attr(unlist(analysis.withoutG@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
        if(variant=="withoutAB") {
        score.withoutAB <- allEdges %in% edgesGraph
        score.withoutAB <- as.data.frame(matrix(score.withoutAB ,nrow=1,ncol=length(allEdges),
                  byrow = T,dimnames = list(1,allEdges)))
        } else {
        score.withoutG <- allEdges %in% edgesGraph
        score.withoutG <- as.data.frame(matrix(score.withoutG ,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)
        if(variant=="withoutAB") {
            x <- unlist(sortString(true.withoutAB ))
            z <- length(setdiff(allEdges,unique(unlist(sortString(edgesGraph))))) # negative claims by Graph
            score.withoutAB$Adj.false.pos <- length(setdiff(unique(unlist(sortString(edgesGraph))), x))/length(unique(edgesGraph))
            score.withoutAB$AdjCompleteness <- length(intersect(unique(unlist(sortString(edgesGraph))),x))/length(x)

            } else {
                 x <- unique(unlist(sortString(true.withoutG)))
                 z <- length(setdiff(allEdges,unique(unlist(sortString(edgesGraph))))) # negative claims by Graph
                 score.withoutG$Adj.false.pos <- length(setdiff(unique(unlist(sortString(edgesGraph))), x))/length(unique(edgesGraph))
                 score.withoutG$AdjCompleteness <- length(intersect(unique(unlist(sortString(edgesGraph))),x))/length(x)
            }

        # False positives/negatives in arrows/ArrCompleteness (Arr.true.pos/directed arrows in true graph)/
        if(variant=="withoutAB") {
           x <- true.withoutAB
           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
           score.withoutAB$Arr.false.pos <- length(union(setdiff(dirArr,x) ,
                                            setdiff(unlist(sortString(z)), unlist(sortString(x))))/
                                length(unique(edgesGraph))
           score.withoutAB$ArrCompleteness <- length(intersect(dirArr,x))/length(x)
           score.withoutAB$chainIGS <- all(c("I|G", "G|S") %in% dirArr)
           } else {
                x <- true.withoutG
                t <- length(setdiff(allArrows , arrowsGraph)) # negative claims by Graph
                z  <- bidir(arrowsGraph) # bi- and undirectional edges in Graph
                dirArr <- setdiff(arrowsGraph,z) # directed edges in Graph
                score.withoutG$Arr.false.pos <- length(union(setdiff(dirArr,x) ,
                                                 setdiff(unlist(sortString(z)), unlist(sortString(x))))/
                                     length(unique(edgesGraph))
                score.withoutG$ArrCompleteness <- length(intersect(dirArr,x))/length(x)
                score.withoutG$pathA <- all(c("I|A", "A|S") %in% dirArr) || all(c("I|A", "A|B", "B|S") %in% dirArr)
                score.withoutG$pathB <- all(c("I|B", "B|S") %in% dirArr) || all(c("I|A", "A|B", "B|S") %in% dirArr)
           }
      } # End variant loop

      # Store analyses , scores
      analyses.storage[[k]] <- list(analysis.withoutAB ,analysis.withoutG)
      score.list.storage[[k]] <-  list(score.withoutAB ,score.withoutG)

} # End data loop


# Overall scoring
# ---------------
overall.score.withoutAB <- lapply(score.list.storage , "[[", 1)
overall.score.withoutAB <- do.call(rbind,overall.score.withoutAB)

overall.score.withoutG <- lapply(score.list.storage , "[[", 2)
overall.score.withoutG <- do.call(rbind,overall.score.withoutG)

overall.ratio.withoutAB <-  matrix(1, ncol = ncol(overall.score.withoutAB), byrow = T,
       dimnames = list(1,names(overall.score.withoutAB)))
overall.ratio.withoutG <-  matrix(1, ncol = ncol(overall.score.withoutG), byrow = T,
       dimnames = list(1,names(overall.score.withoutG)))

for(i in 1:10){
       overall.ratio.withoutAB[1,i] <- length(which(overall.score.withoutAB[,i]))/nrow(overall.score.withoutAB)
   }
for(i in 1:15){
       overall.ratio.withoutG[1,i] <- length(which(overall.score.withoutG[,i]))/nrow(overall.score.withoutG)
   }
overall.ratio.withoutAB[,c("Adj.false.pos")] <- mean(overall.score.withoutAB$Adj.false.pos)
overall.ratio.withoutG[,c("Adj.false.pos")] <- mean(overall.score.withoutG$Adj.false.pos)
overall.ratio.withoutAB[,c("Arr.false.pos")] <- mean(overall.score.withoutAB$Arr.false.pos)
overall.ratio.withoutG[,c("Arr.false.pos")] <- mean(overall.score.withoutG$Arr.false.pos)
overall.ratio.withoutAB[,c("AdjCompleteness")] <- mean(overall.score.withoutAB$AdjCompleteness)
overall.ratio.withoutG[,c("AdjCompleteness")] <- mean(overall.score.withoutG$AdjCompleteness)
overall.ratio.withoutAB[,c("ArrCompleteness")] <- mean(overall.score.withoutAB$ArrCompleteness)
overall.ratio.withoutG[,c("ArrCompleteness")] <- mean(overall.score.withoutG$ArrCompleteness)
overall.ratio.withoutAB[,c("chainIGS")] <- length(which(overall.score.withoutAB[,c("chainIGS")]))/nrow(overall.score.withoutAB)
overall.ratio.withoutG[,c("pathA")] <- length(which(overall.score.withoutG[,c("pathA")]))/nrow(overall.score.withoutG)
overall.ratio.withoutG[,c("pathB")] <- length(which(overall.score.withoutG[,c("pathB")]))/nrow(overall.score.withoutG)
```

```
# Plot 2 (Figure 7)
# -----------------
selection.withoutAB <- overall.ratio.withoutAB[,c("AdjCompleteness","ArrCompleteness",
                                                  "Adj.false.pos","Arr.false.pos", "chainIGS")]

selection.withoutG <- overall.ratio.withoutG[,c("AdjCompleteness","ArrCompleteness", "Adj.false.pos",
                                                "Arr.false.pos", "pathA", "pathB")]

selection <- rbind.fill(as.data.frame(t(selection.withoutAB)),as.data.frame(t(selection.withoutG)))

colnames(selection) <-  c("edge.complete","orient.complete","edge.false.pos","orient.false.pos",
                                             "path␣G", "path␣A", "path␣B")
final.score <-  selection
final.score$variants <- variants
k1 <- melt(final.score, id.vars = "variants")
k1[is.na(k1)] <- 0
colnames(k1) <- c("variants", "property", "value")
plot2 <- 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 = .95)+
  theme_bw()+
  theme(legend.title=element_blank())+
  theme(plot.title = element_text(size = 9))+
  scale_x_discrete(name ="")

options(tz="CA")
tikz(file = "plot2.tex", width = 6, height = 2.2)
print(plot2)
dev.off()
```