In [6]:

```
#' Nonparametric multiple comparisons (Nemenyi test)
#'
#' Perform nonparametric multiple comparisons, across columns, using the Friedman an
#'
#' @param data an array that includes values to be compared for several treatments (
#' @param conf.level the confidence level used for the comparison. Default is 0.95.
#' @param sort if \code{TRUE}, then function sorts the outputted values of mean rank
#' @param plottype type of plot to produce:
#' \itemize{
#'  \item{\code{"none"}}{: no plot.}
#'  \item{\code{"mcb"}}{: \emph{Multiple Comparison with the Best} style plot.}
#'  \item{\code{"vmcb"}}{: vertical \emph{MCB} plot.}
#'  \item{\code{"line"}}{: summarised \emph{line} plot.}
#'  \item{\code{"vline"}}{: vertical \emph{line} plot.}
#'  \item{\code{"matrix"}}{: complete \emph{matrix} visualisation.}}
#' @param select highlight selected treatment (column). Number 1 to k. Use NULL for
#' @param labels optional labels for models. If NULL column names of \code{data} wil
#' @param ... additional arguments passed to the \code{plot} function.
#'
#' @return Return object of class \code{nemenyi} and contains:
#' \itemize{
#' \item{\code{means}}{: mean rank of each treatment.}
#' \item{\code{intervals}}{: intervals within there is no evidence of significance d
#' \item{\code{fpavl}}{: Friedman test p-value.}
#' \item{\code{fH}}{: Friedman test hypothesis outcome.}
#' \item{\code{cd}}{: Nemenyi critical distance. Output \code{intervals} is calculat
#' \item{\code{conf.level}}{: confidence level used for testing.}
#' \item{\code{k}}{: number of treatments (columns).}
#' \item{\code{n}}{: number of observations (rows).}
#' }
#'
#' @author Nikolaos Kourentzes, \email{nikolaos@kourentzes.com},
#' @author Ivan Svetunkov, \email{ivan@svetunkov.ru}.
#'
#' @references
#' \itemize{
#' \item{The tests are deailed by Hollander, M., Wolfe, D.A. and Chicken, E. (2014)
#' \item{The \emph{line} plot is introduced \href{http://kourentzes.com/forecasting/
#' \item{The \emph{matrix} plot is introduced by Kourentzes, N., & Athanasopoulos, G
#' \item{The \emph{MCB} plot is described by Koning, A. J., Franses, P. H., Hibon, M
#' }
#'
#' @keywords htest
#'
#' @examples
#' x <- matrix( rnorm(50*4,mean=0,sd=1), 50, 4)
#' x[,2] <- x[,2]+1
#' x[,3] <- x[,3]+0.7
#' x[,4] <- x[,4]+0.5
#' colnames(x) <- c("Method A","Method B","Method C - long name","Method D")
#' nemenyi(x,conf.level=0.95,plottype="vline")
#'
#' @export nemenyi

nemenyi <- function(data, conf.level=0.95, sort=c(TRUE,FALSE),
                    plottype=c("vline","none","mcb","vmcb","line","matrix"),
                    select=NULL, labels=NULL, ...){

    # Default
    sort <- sort[1]
    plottype <- match.arg(plottype,c("vline","none","mcb","vmcb","line","matrix"))
    # Check data
    if (length(dim(data)) != 2){
        stop("Data must be organised as methods in columns and observations in rows.
    }
    data <- na.exclude(data)
    rows.number <- nrow(data)
    cols.number <- ncol(data)

    # Check select argument
    if (!is.null(select) && (select > cols.number)){
        select <- NULL
    }

    # If plot is asked, always sort the results
    if (plottype != "none"){
        sort <- TRUE
    }

    # Checks for labels
    if (is.null(labels)){
        labels <- colnames(data)
        if (is.null(labels)){
            labels <- 1:cols.number
        }
    } else {
        labels <- labels[1:cols.number]
    }

    # First run Friedman test. If insignificant then ignore Nemenyi (Weaker)
    fried.pval <- stats::friedman.test(data)$p.value
    if (fried.pval <= 1-conf.level){
        fried.H <- "Ha: Different" # At least one method is different
    } else {
        fried.H <- "H0: Identical" # No evidence of differences between methods
    }

    # Nemenyi critical distance and bounds of intervals
    # Nikos: Use the formulas for mean ranks, not sums
    r.stat <- stats::qtukey(conf.level,cols.number,Inf)*sqrt((cols.number*(cols.numb
    # r.stat <- qtukey(conf.level,cols.number,Inf)*sqrt((rows.number*cols.number*(co
    # NSM3::cWNMT(0.95, cols.number, rows.number, method="Asymptotic") # This is the

    # Rank methods for each time series
    ranks.matrix <- t(apply(data,1,function(x){rank(x,na.last="keep",ties.method="av

    # Calculate mean rank values
    ranks.means <- colMeans(ranks.matrix)
    # ranks.sum <- colSums(ranks.matrix)

    # Calculate intervals for each of the methods
    # The comparison is abs(diff(ranks)) < r.stat, otherwise different groups
    ranks.intervals <- rbind(ranks.means - r.stat,ranks.means + r.stat)

    # Sort interval matrix and means
    if(sort==TRUE){
        order.idx <- order(ranks.means)
    } else {
        order.idx <- 1:cols.number
    }
    ranks.means <- ranks.means[order.idx]
    ranks.intervals <- ranks.intervals[,order.idx]
    labels <- labels[order.idx]
    if (!is.null(select)){
        select <- which(order.idx == select)
    }

    # Produce plots
    # For all plots
    if (plottype != "none"){
        args <- list(...)
        args.nms <- names(args)

        # Create title for plots
        if (!("main" %in% args.nms)){
            args$main <- paste0("Friedman: ", format(round(fried.pval,3),nsmall=3),
        }

        # Remaining defaults
        if (!("xaxs" %in% names(args))){
            args$xaxs <- "i"
        }
        if (!("yaxs" %in% names(args))){
            args$yaxs <- "i"
        }

        # Size of labels
        nc <- max(nchar(labels))
        nc <- nc/1.75 + 1
        nr <- nchar(sprintf("%1.2f",round(max(ranks.means),2)))/1.75

        # Get margins
        parmar.def <- parmar <- graphics::par()$mar

    }

    # MCB style plots
    if ((plottype == "mcb")|(plottype == "vmcb")){

        # Set colours
        cmp <- RColorBrewer::brewer.pal(3,"Set1")[1:2]
        if (fried.pval > 1-conf.level){pcol <- "gray"} else {pcol <- cmp[2]}

        # Find min max
        mnmx <- range(ranks.means) + c(-0.5,0.5)*r.stat
        mnmx <- mnmx + diff(mnmx)*0.04*c(-1,1)

        # Set plot - horizontal or vertical
        if (plottype == "mcb"){ # Horizontal
            if (!("xlab" %in% names(args))){
                args$xlab <- ""
            }
            if (!("ylab" %in% names(args))){
                args$ylab <- "Mean ranks"
            }
            if(is.null(args$xlim)){
                args$xlim <- c(0,cols.number+1)
            }
            if(is.null(args$ylim)){
                args$ylim <- mnmx
            }
        } else { # Vertical
            if (!("ylab" %in% names(args))){
                args$ylab <- ""
            }
            if (!("xlab" %in% names(args))){
                args$xlab <- "Mean ranks"
            }
            if(is.null(args$ylim)){
```

```
            args$ylim <- c(0,cols.number+1)
        }
        if(is.null(args$xlim)){
            args$xlim <- mnmx
        }
    }

    # Remaining defaults
    args$x <- args$y <- NA
    args$axes <- FALSE

    # Change plot size to fit labels
    if ((plottype == "mcb") && (parmar[1] < (nc+nr))){
        parmar[1] <- nc + nr
    }
    if ((plottype == "vmcb") && (parmar[2] < (nc+nr))){
        parmar[2] <- nc + nr
    }
    par(mar=parmar)

    if (is.null(select)){
        select <- 1
    }

    # Use do.call to use manipulated ellipsis (...)
    do.call(plot,args)
    # Plot rest
    if (plottype == "mcb"){
        # Intervals for best method
        polygon(c(0,rep(cols.number+1,2),0),rep(ranks.means[select],4)+r.stat/2*
        # Ranks
        points(1:cols.number,ranks.means,pch=20,lwd=3)
        axis(1,at=c(1:cols.number),labels=paste0(labels," - ",sprintf("%1.2f",ro
        axis(2)
        # Intervals for all methods
        for (i in 1:cols.number){
            lines(rep(i,times=2),ranks.means[i]+c(-1,1)*0.5*r.stat, type="o", lw
        }
        # Highlight identical
        idx <- abs(ranks.means[select] - ranks.means) < r.stat
        points((1:cols.number)[idx],ranks.means[idx],pch=20,lwd=3,col=cmp[1])
    } else { # vmcb
        # Intervals for best method
        polygon(rep(ranks.means[select],4)+r.stat/2*c(1,1,-1,-1),c(0,rep(cols.nu
        # Ranks
        points(ranks.means,1:cols.number,pch=20,lwd=3)
        axis(2,at=c(1:cols.number),labels=paste0(labels," - ",sprintf("%1.2f",ro
        axis(1)
        # Intervals for all methods
        for (i in 1:cols.number){
            lines(ranks.means[i]+c(-1,1)*0.5*r.stat, rep(i,times=2), type="o", l
        }
        # Highlight identical
        idx <- abs(ranks.means[select] - ranks.means) < r.stat
        points(ranks.means[idx],(1:cols.number)[idx],pch=20,lwd=3,col=cmp[1])
    }
    box(which="plot", col="black")

}

# New complete Nemenyi visualisation
if (plottype == "matrix"){

    # Construct group matrix
```

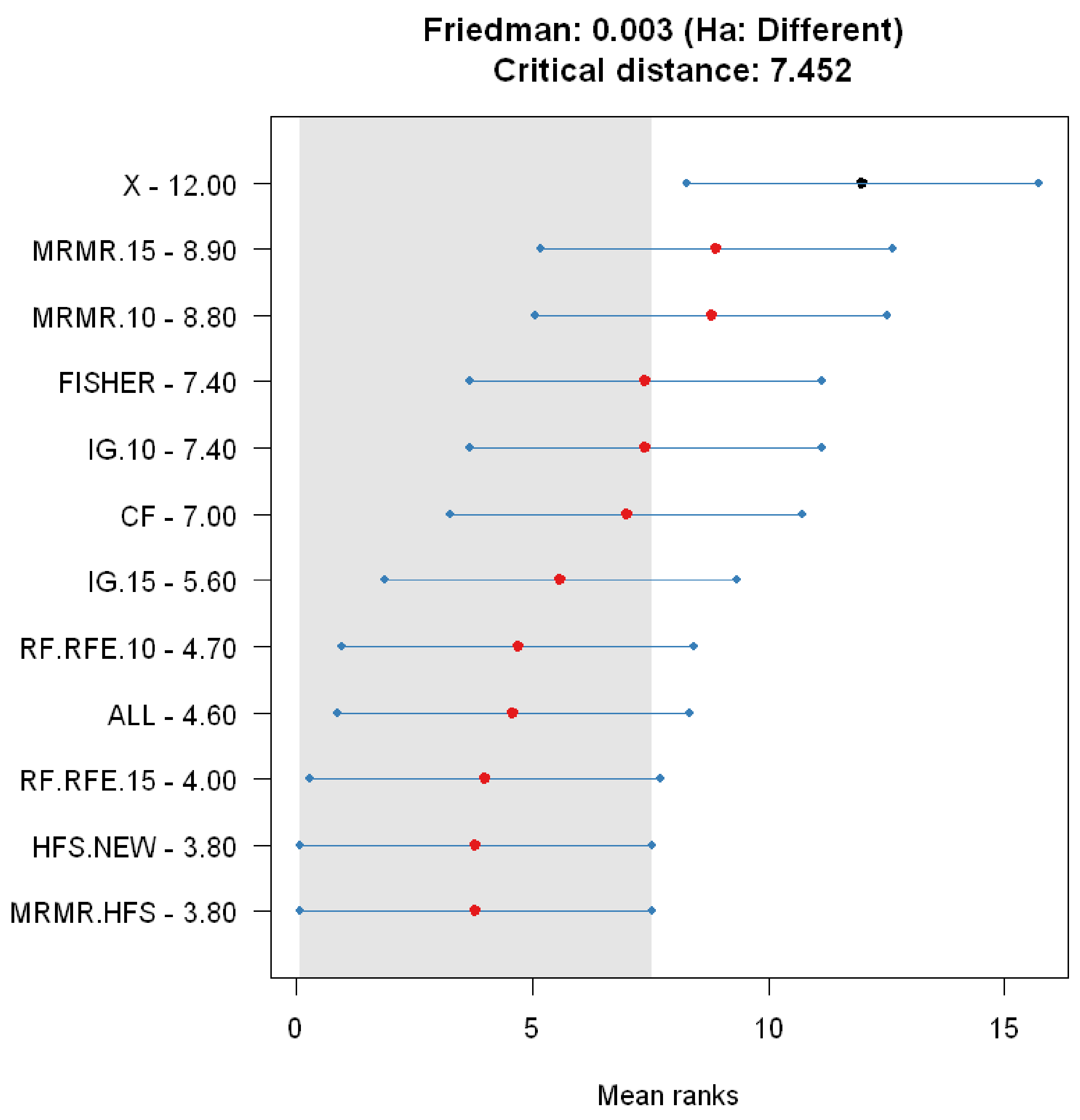
```
        rline <- array(NA, c(cols.number,2))
        nem.mat <- array(0,c(cols.number,cols.number))
        for (i in 1:cols.number){
            rline[i,] <- c(which(ranks.means > ranks.intervals[1,i])[1], # Start mod
                           tail(which(ranks.means < ranks.intervals[2,i]),1)) # End
            nem.mat[i,rline[i,1]:rline[i,2]] <- 1
        }
        diag(nem.mat) <- 2

        # Set margins to fit labels
        if (parmar[1] < nc){
            parmar[1] <- nc
        }
        nr <- nchar(sprintf("%1.2f",round(max(ranks.means),2)))/1.75
        if (parmar[2] < (nc + nr)){
            parmar[2] <- nc + nr
        }
        par(mar=parmar)

        # Start plotting
        # Get defaults
        if (!("xlab" %in% names(args))){
            args$xlab <- ""
        }
        if (!("ylab" %in% names(args))){
            args$ylab <- ""
        }
        args$x <- 1:cols.number
        args$y <- 1:cols.number
        args$z <- nem.mat[order(order.idx),cols.number:1]
        args$axes <- FALSE
        # Colours
        cmp <- c("white",RColorBrewer::brewer.pal(3,"Set1")[2],"black")
        if (fried.pval > 1-conf.level){
            cmp[3] <- "gray60"
            cmp[2] <- "gray70"
        }
        args$col <- cmp

        # Do plot
        do.call(image,args)
        for (i in 1:cols.number){
            abline(v=i+0.5)
            abline(h=i+0.5)
        }
        axis(1,at=1:cols.number,labels=labels[order(order.idx)],las=2)
        axis(2,at=1:cols.number,labels=paste0(rev(labels)," - ",sprintf("%1.2f",roun
        box()
        if (!is.null(select)){
            polygon(order.idx[select]+c(-.5,.5,.5,-.5),which((nem.mat[order(order.id
        }

    }

    # Line style plot (as in ISF reference)
    if ((plottype == "line")|(plottype == "vline")){

        # Find groups
        rline <- matrix(NA, nrow=cols.number, ncol=2)
        for (i in 1:cols.number){
            tloc <- which((abs(ranks.means-ranks.means[i])<r.stat) == TRUE)
            rline[i,] <- c(min(tloc),max(tloc))
        }
        # Remove duplicates and single member groups
        rline <- unique(rline)
        rline <- rline[apply(rline,1,min) != apply(rline,1,max),]
        # Reshape to matrix if necessary and find number of remaining groups
        if (length(rline)==2){
            rline <- as.matrix(rline)
            rline <- t(rline)
        }
        k <- nrow(rline)
        # Choose colour depending on Friedman test result
        cmp <- colorRampPalette(RColorBrewer::brewer.pal(12,"Paired"))(k)
        if (fried.pval > 1-conf.level){cmp <- rep("gray",times=k)}
        # Prepare method labels and add mean rank to them
        lbl <- paste0(labels," - ",sprintf("%1.2f",round(ranks.means,2)))

        # Produce plot
        if (!("ylab" %in% names(args))){
            args$ylab <- ""
        }
        if (!("xlab" %in% names(args))){
            args$xlab <- ""
        }
        args$x <- args$y <- NA
        args$axes <- FALSE

        if (plottype == "line"){
            if(is.null(args$xlim)){
                args$xlim <- c(1,cols.number)
            }
            if(is.null(args$ylim)){
                args$ylim <- c(0,k+1)
            }
        } else { # vline
            if(is.null(args$xlim)){
                args$xlim <- c(0,k+1)
            }
            if(is.null(args$ylim)){
                args$ylim <- c(1,cols.number)
            }
        }

        # Change plot size to fit labels
        if ((plottype == "line") && (parmar[1] < (nc+nr))){
            parmar[1] <- nc + nr
        }
        if ((plottype == "vline") && (parmar[2] < (nc+nr))){
            parmar[2] <- nc + nr
        }
        par(mar=parmar)

        do.call(plot,args)

        if (plottype == "line"){
            points(1:cols.number,rep(0,cols.number),pch=20,lwd=4)
            if (k>0){
                for (i in 1:k){
                    lines(rline[i,],c(i,i), col=cmp[i], lwd = 4)
                    lines(rep(rline[i,1],times=2),c(0,i), col="gray", lty = 2)
                    lines(rep(rline[i,2],times=2),c(0,i), col="gray", lty = 2)
                }
            }
            axis(1,at=c(1:cols.number),labels=lbl,las=2)
            if (!is.null(select)){
                points(select,0,pch=20,col=RColorBrewer::brewer.pal(3,"Set1")[1],cex
            }
```

```
            } else { # vline
                points(rep(0,cols.number),1:cols.number,pch=20,lwd=4)
                if (k>0){
                    for (i in 1:k){
                        lines(c(i,i), rline[i,], col=cmp[i], lwd = 4)
                        lines(c(0,i), rep(rline[i,1],times=2), col="gray", lty = 2)
                        lines(c(0,i), rep(rline[i,2],times=2), col="gray", lty = 2)
                    }
                }
                axis(2,at=c(1:cols.number),labels=lbl,las=2)
                if (!is.null(select)){
                    points(0,select,pch=20,col=RColorBrewer::brewer.pal(3,"Set1")[1],cex
                }
            }

        }

        if (plottype != "none"){
            par(mar=parmar.def)
        }

        return(structure(list("means"=ranks.means,"intervals"=ranks.intervals,"fpval"=fr

    }

    #' @export
    #' @method summary nemenyi
    summary.nemenyi <- function(object,...){
        print(object)
    }

    #' @export
    #' @method print nemenyi
    print.nemenyi <- function(x,...){

        writeLines("Friedman and Nemenyi Tests")
        writeLines(paste0("The confidence level is ", (1-x$conf.level)*100, "%"))
        writeLines(paste0("Number of observations is ", x$n, " and number of methods is
        writeLines(paste0("Friedman test p-value: ", format(round(x$fpval,4),nsmall=4) ,
        writeLines(paste0("Critical distance: ", format(round(x$cd,4),nsmall=4)))
    }
```

In [7]:

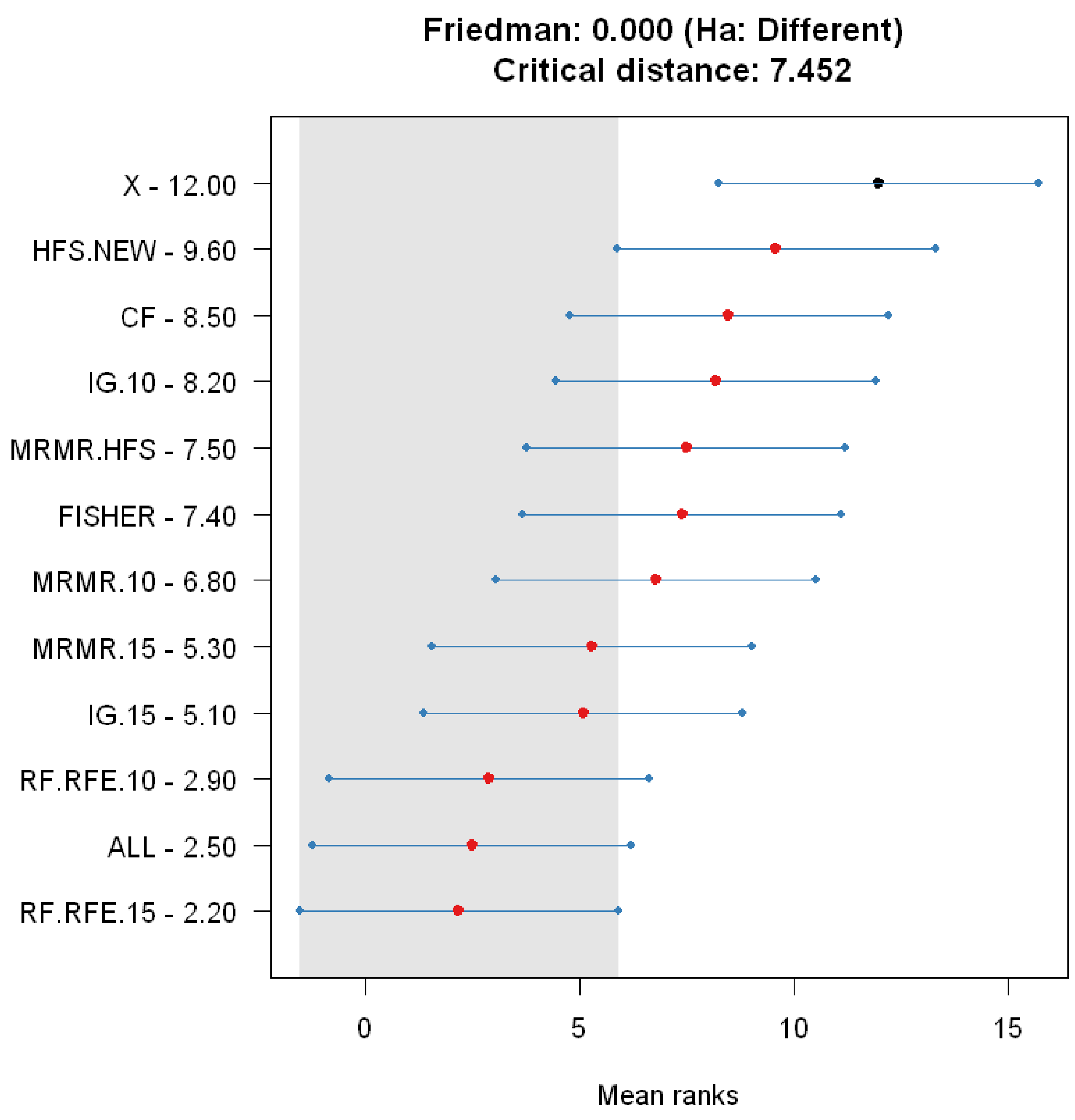
```
data <- read.csv("C:\\Users\\chetn\\Downloads\\AEEEM GNB.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 5 and number of methods is 12
Friedman test p-value: 0.0026 - Ha: Different
Critical distance: 7.4522
```



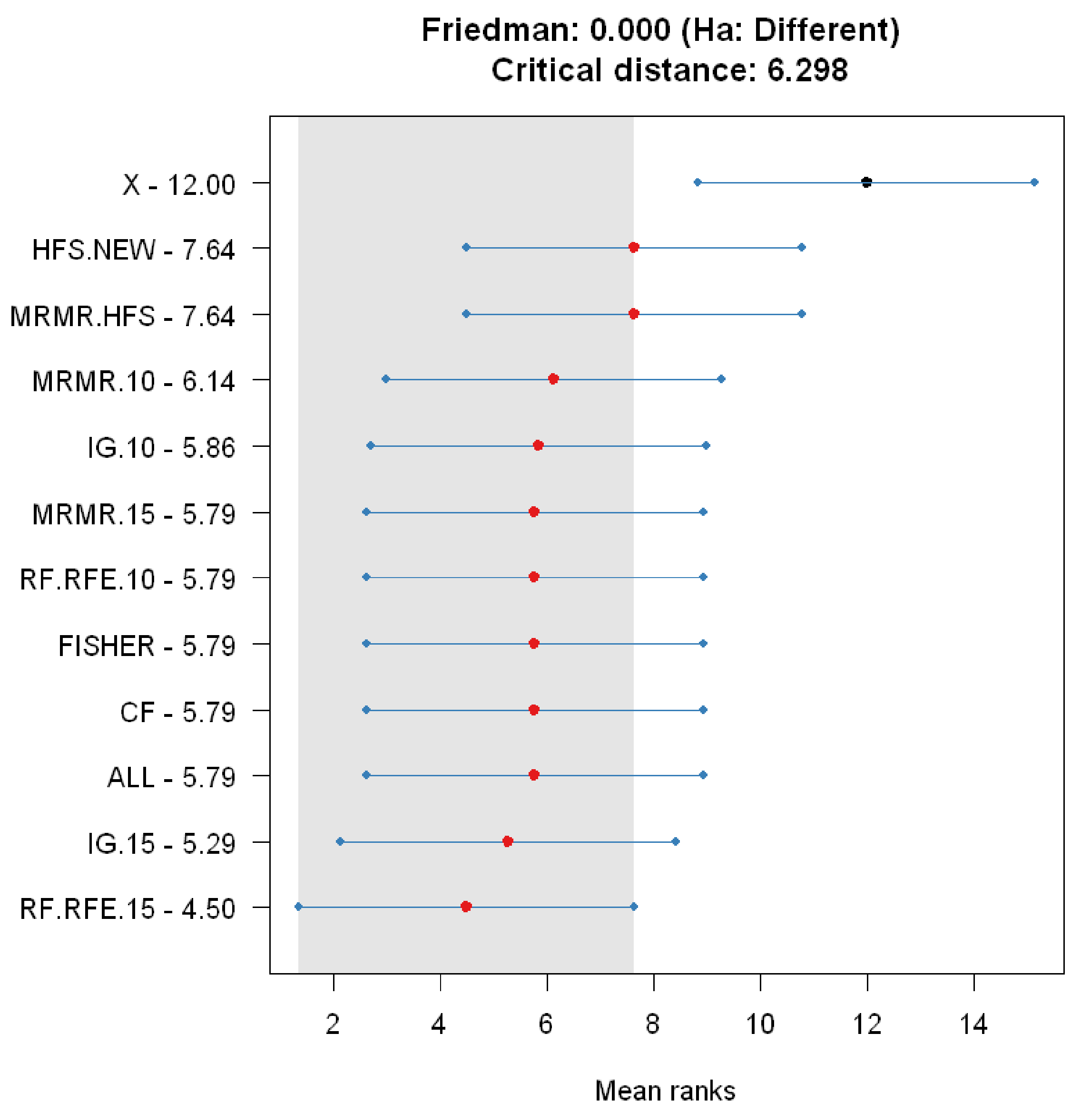
In [8]:
```
data <- read.csv("C:\\Users\\chetn\\Downloads\\AEEEM KNN.csv")
d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 5 and number of methods is 12
Friedman test p-value: 0.0000 - Ha: Different
Critical distance: 7.4522
```



In [9]:
```
data <- read.csv("C:\\Users\\chetn\\Downloads\\ARP GNB.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

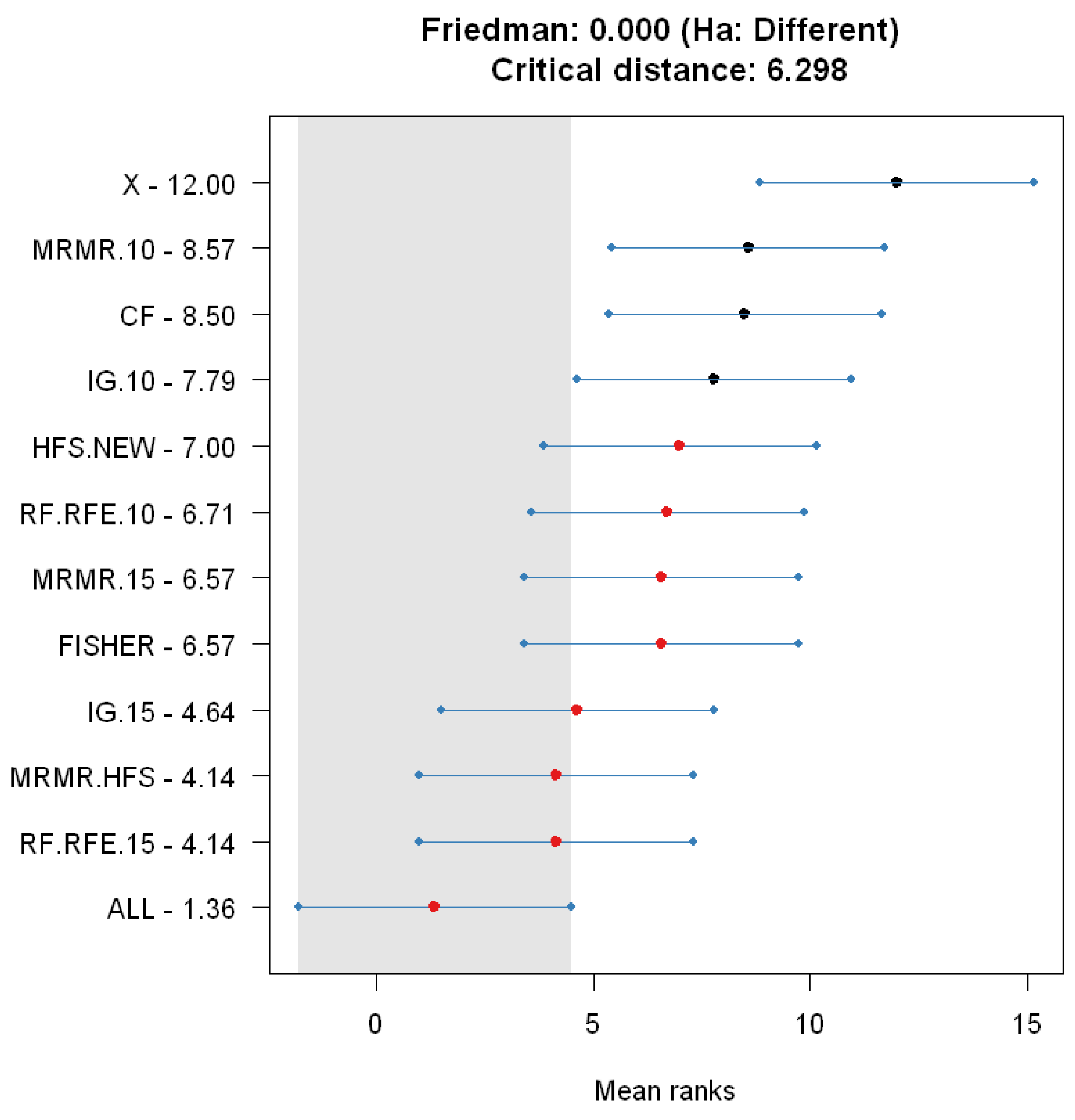
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 7 and number of methods is 12
Friedman test p-value: 5e-04 - Ha: Different
Critical distance: 6.2983
```



In [10]:

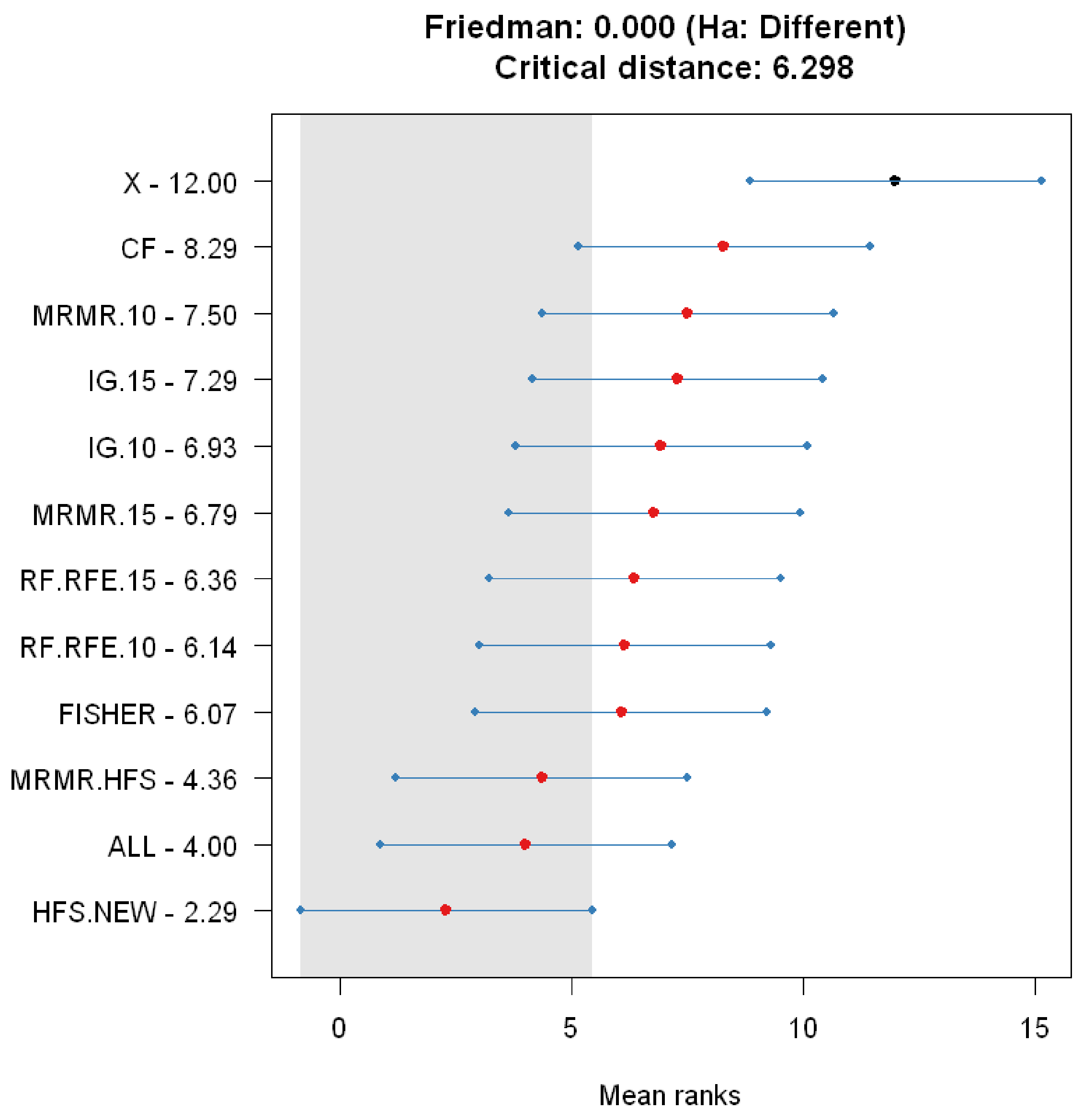
```
data <- read.csv("C:\\Users\\chetn\\Downloads\\JIRA GNB.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 7 and number of methods is 12
Friedman test p-value: 0.0000 - Ha: Different
Critical distance: 6.2983
```



```
In [11]:
data <- read.csv("C:\\Users\\chetn\\Downloads\\JIRA KNN.csv")
d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

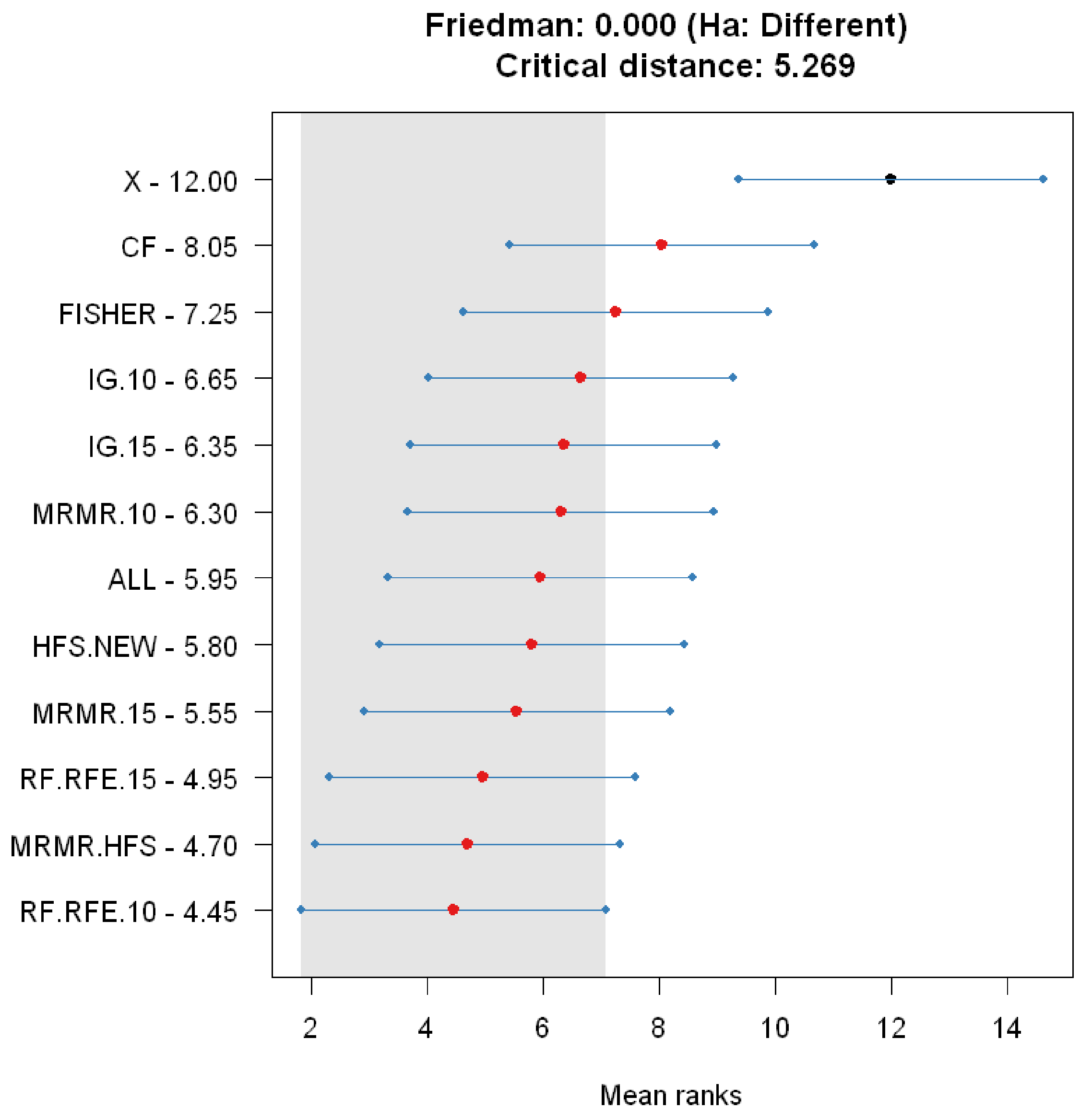
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 7 and number of methods is 12
Friedman test p-value: 2e-04 - Ha: Different
Critical distance: 6.2983
```



In [12]:

```
data <- read.csv("C:\\Users\\chetn\\Downloads\\PROMISE GNB.csv")
d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

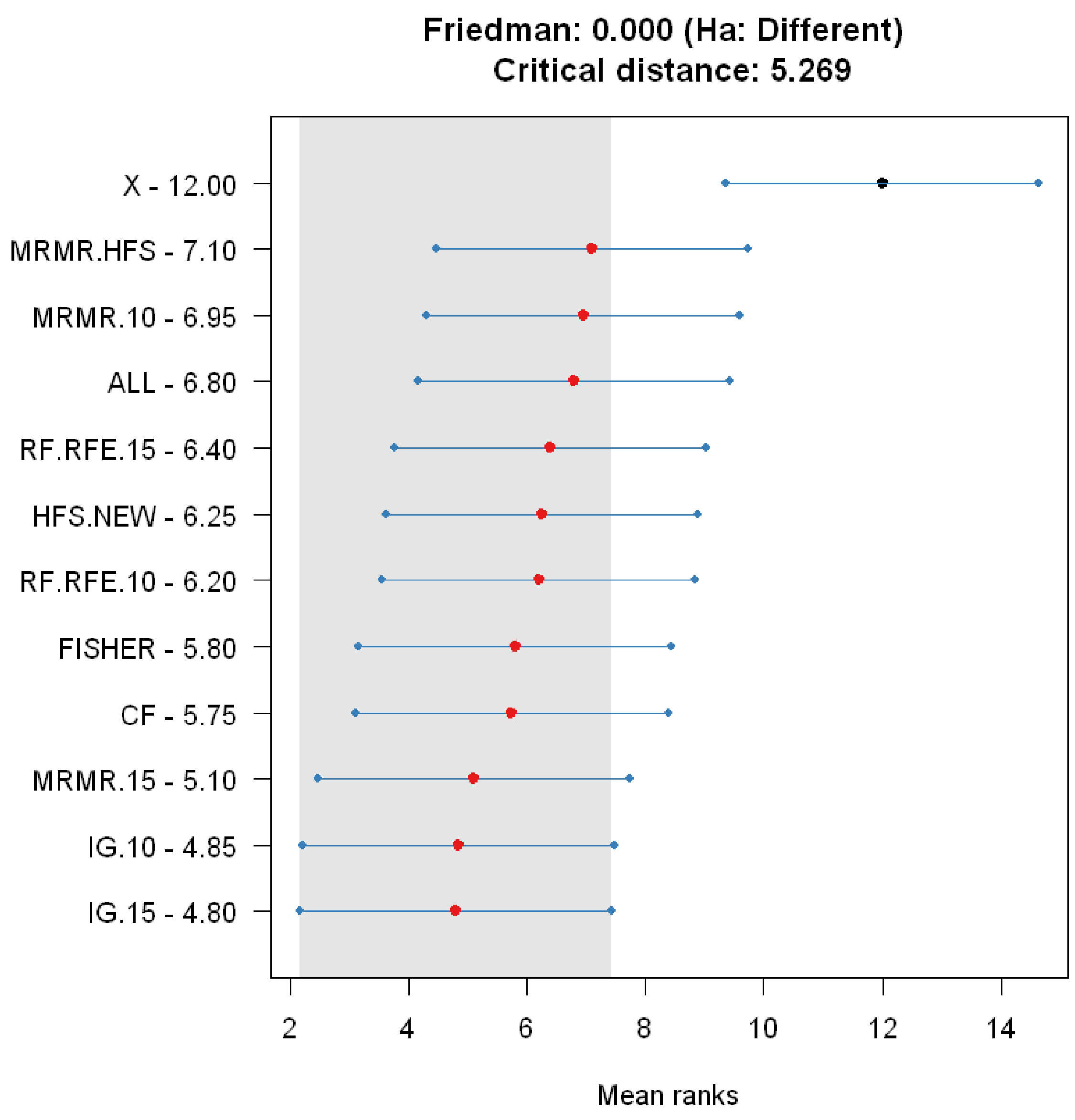
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 10 and number of methods is 12
Friedman test p-value: 1e-04 - Ha: Different
Critical distance: 5.2695
```



In [13]:

```
data <- read.csv("C:\\Users\\chetn\\Downloads\\PROMISE KNN.csv")
d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

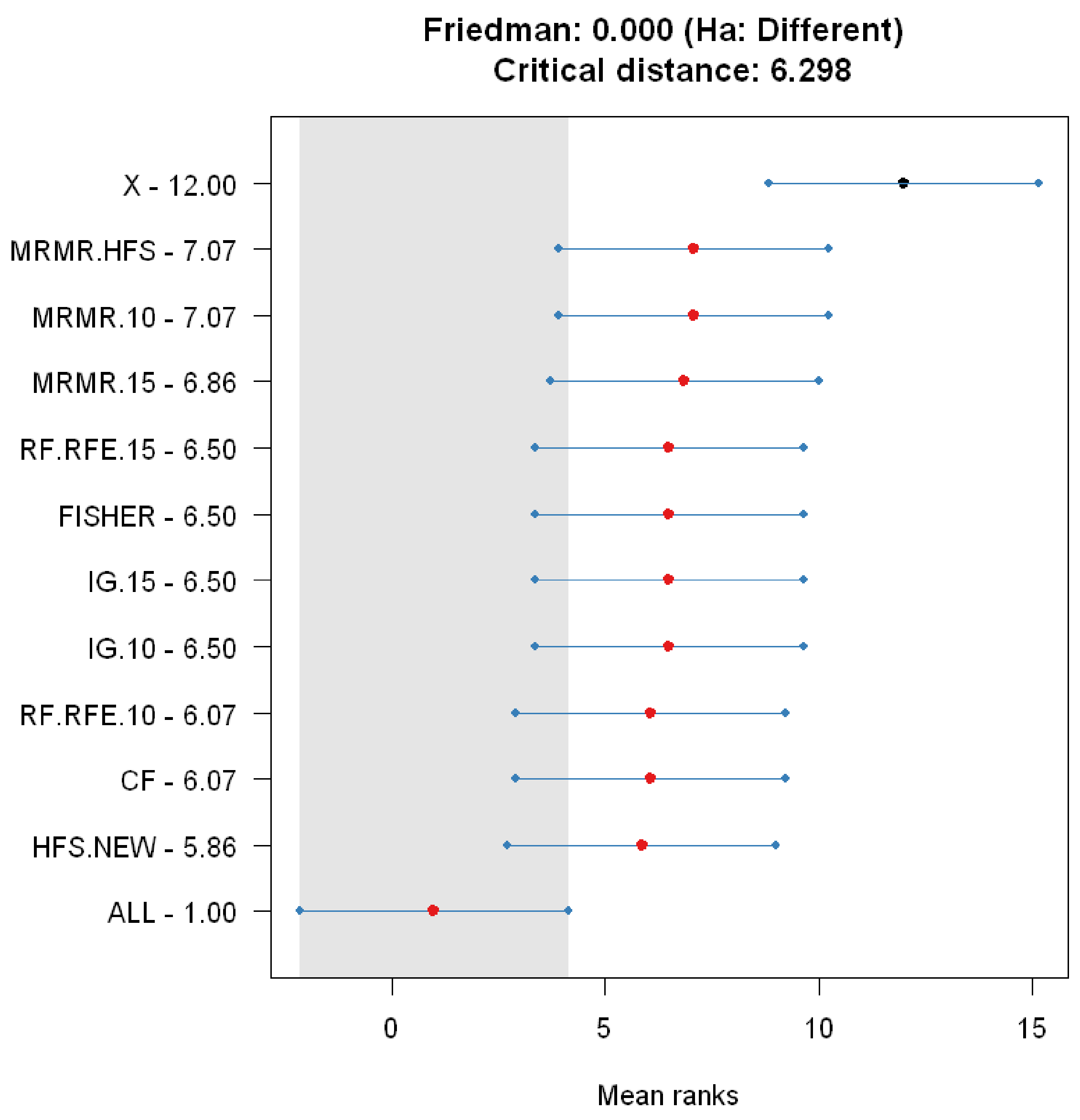
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 10 and number of methods is 12
Friedman test p-value: 3e-04 - Ha: Different
Critical distance: 5.2695
```



In [14]:

```
data <- read.csv("C:\\Users\\chetn\\Downloads\\ARP KNN.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

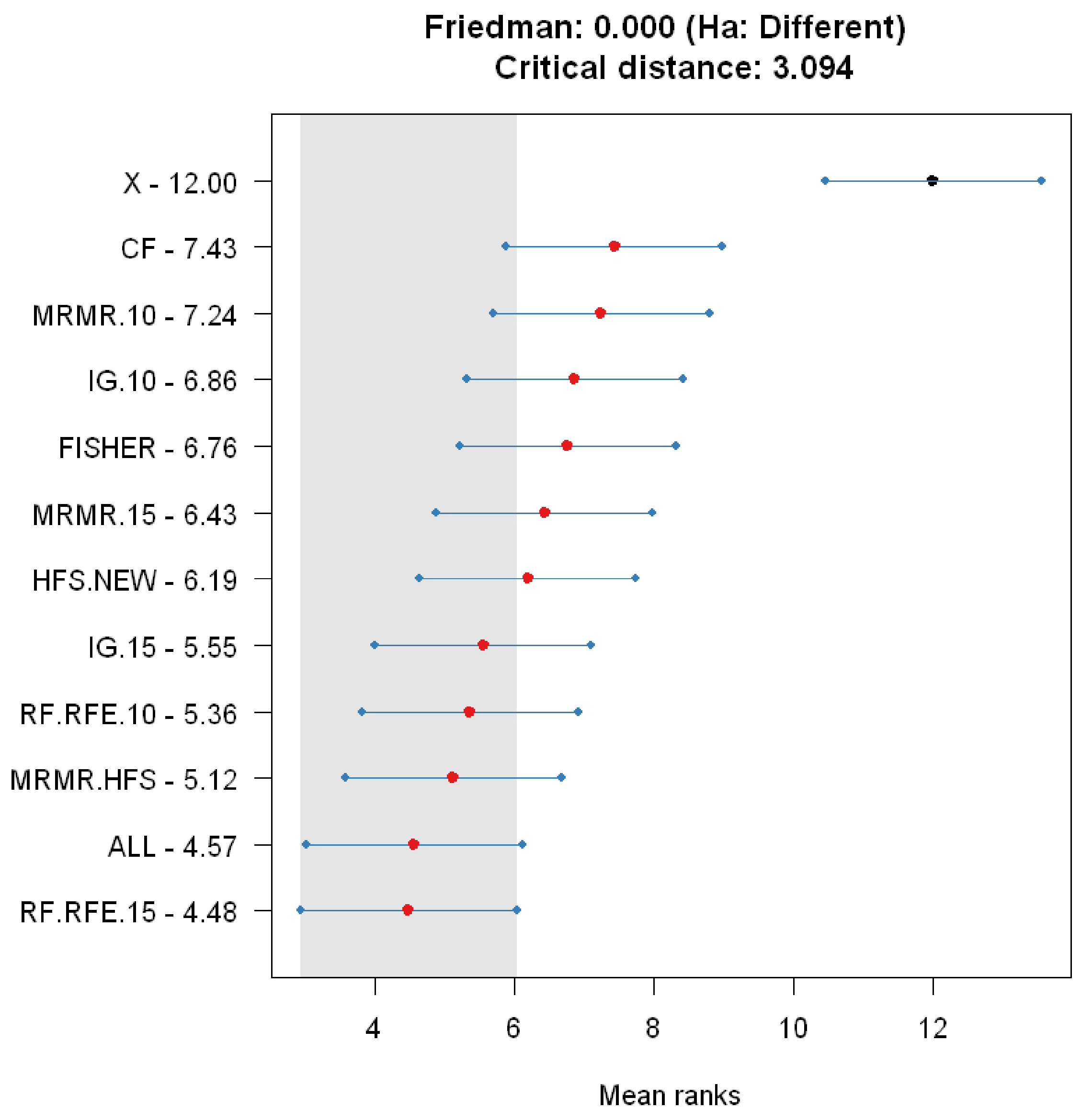
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 7 and number of methods is 12
Friedman test p-value: 0.0000 - Ha: Different
Critical distance: 6.2983
```



In [15]:

```
data <- read.csv("C:\\Users\\chetn\\Downloads\\GNB Data.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

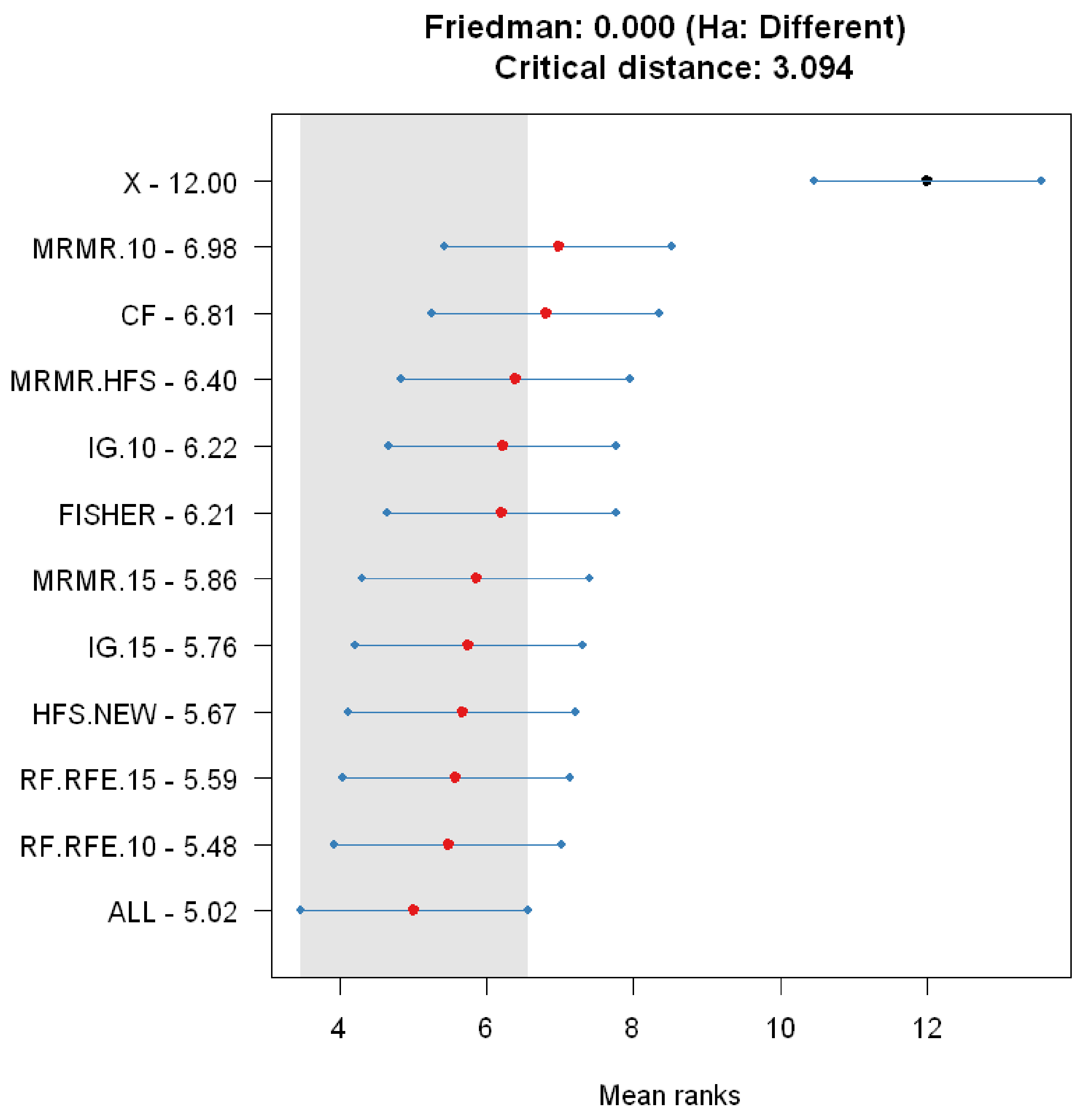
```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 29 and number of methods is 12
Friedman test p-value: 0.0000 - Ha: Different
Critical distance: 3.0944
```



In [16]:

```
data <- read.csv("C:\\Users\\chetn\\Downloads\\KNN Data.csv")

d=as.matrix(data)
nemenyi(d,conf.level=0.95,plottype="vmcb")
```

```
Friedman and Nemenyi Tests
The confidence level is 5%
Number of observations is 29 and number of methods is 12
Friedman test p-value: 0.0000 - Ha: Different
Critical distance: 3.0944
```


Friedman: 0.000 (Ha: Different)
Critical distance: 3.094
X - 12.00
MRMR.10 - 6.98
CF - 6.81
MRMR.HFS - 6.40
IG.10 - 6.22
FISHER - 6.21
MRMR.15 - 5.86
IG.15 - 5.76
HFS.NEW - 5.67
RF.RFE.15 - 5.59
RF.RFE.10 - 5.48
ALL - 5.02
4
6
8
10
12
Mean ranks

```
In [ ]:
```