Helper function for the muscle testing project

Joachim Vandekerckhove, Beth Baribault, and Jennifer Wilson

April 11, 2018

A collection of helper functions for the muscle testing paper.

Function: kendalls.tau()

Computes an unnormalized Kendall's tau-a coefficient

```
kendalls.tau
```

```
## function (list1, list2)
## {
##     if (any(is.na(list1 + list2))) {
##         return(NA)
##     }
##     n <- 1:length(list2)
##     rank1 <- sort(x = list1, index.return = TRUE)$ix
##     rank2 <- sort(x = list2, index.return = TRUE)$ix
##     ii <- outer(n, n, function(x, y) {
##         x
##     })
##     jj <- outer(n, n, function(x, y) {
##         y
##     })
##     sign1 <- rank1[jj[jj > ii]] > rank1[ii[jj > ii]]
##     sign2 <- rank2[jj[jj > ii]] > rank2[ii[jj > ii]]
##     return(sum(sign1 != sign2))
## }
```

Function: expected.tau()

Computes tau for every possible outcome and tabulates the frequencies

```
expected.tau
```

```
## function (concordance, length)
## {
##     domain <- combinat::permn(length - concordance)
##     N <- length(domain)
##     arr <- sapply(domain, kendalls.tau, domain[[1]])
##     freqs <- table(arr)
##     labels <- as.numeric(labels(freqs)[[1]])
##     values <- as.numeric(freqs)
##     out <- array(0, length * (length - 1)/2 + 1)
##     out[labels + 1] <- values
##     return(out/sum(out))
## }
```

Function: tau.matrix()

Compute probability function for each possible level of concordance

```
tau.matrix
```

```
## function (length)
## {
##     return(t(sapply(0:(length - 1), expected.tau, length)))
## }
```

Function: norm()

A normalization function

```
norm
```

```
## function (x)
## {
##     return(x/sum(x))
## }
```

Function: read.mt.data()

This function will read the .csv data file from OSF and format into a list

```
read.mt.data
```

```
## function (data.url = "https://osf.io/4rp7c/download")
## {
##     a <- read.csv(file = url(data.url), header = FALSE, col.names = c("session",
##         "tester", "participant", "suppl.1", "suppl.2", "suppl.3",
##         "suppl.4", "suppl.5", "label.1", "label.2", "label.3",
##         "label.4", "label.5", "confidence", "belief"))
##     a$labels <- cbind(a$label.1, a$label.2, a$label.3, a$label.4,
##         a$label.5)
##     a$supplements <- cbind(a$suppl.1, a$suppl.2, a$suppl.3, a$suppl.4,
##         a$suppl.5)
##     return(a[, -grep("\\.", colnames(a))])
## }
```

Function: process.data()

Extract the relevant subset of the muscle testing data

```
process.data
```

```
## function (muscle.data, tester.subset = c(1:5, 7, 9), data.type = "supplements",
##     confident.only = FALSE, believers.only = FALSE)
## {
##     use <- switch(confident.only + 2 * believers.only + 1, !logical(length(muscle.data$belief)),
##         muscle.data$confidence == 1, muscle.data$belief == 1,
##         muscle.data$confidence & muscle.data$belief, stop("process.data# Invalid input(s) check 'con
##     rankings <- switch(data.type, supplements = muscle.data$supplements[use,
##         ], labels = muscle.data$labels[use, ], stop("process.data# Invalid input. Check 'data.type' 
##     tester <- muscle.data$tester[use]
```

```
##     participant <- muscle.data$participant[use]
##     participantList <- unique(muscle.data$participant[use])
##     taus <- integer()
##     for (p in 1:length(participantList)) {
##         foundranks <- rankings[muscle.data$participant[use] ==
##             participantList[p] & is.element(muscle.data$tester[use],
##             tester.subset), ]
##         M <- dim(foundranks)[1]
##         taus.tmp <- array(NA, c(M, M))
##         for (m1 in 1:M) {
##             for (m2 in 1:M) {
##                 if (m2 > m1) {
##                   taus.tmp[m1, m2] <- kendalls.tau(foundranks[m1,
##                     ], foundranks[m2, ])
##                 }
##             }
##         }
##         taus <- c(taus, taus.tmp[!is.na(taus.tmp)])
##     }
##     return(taus)
## }
```

Function: run.mt()

This function runs the analysis

```
run.mt
```

```
## function (muscle.data, tester.subset = c(1:5, 7, 9), data.type = "supplements",
##     confident.only = TRUE, believers.only = TRUE, sigma.prior = function(x) {
##         seq(1, 1, along.with = x)
##     }, lambda.prior = function(x) {
##         seq(1, 1, along.with = x)
##     })
## {
##     data <- process.data(muscle.data = muscle.data, tester.subset = tester.subset,
##         data.type = data.type, confident.only = confident.only,
##         believers.only = believers.only)
##     items <- 5
##     sigma <- 0:(items - 1)
##     lambda <- seq(0, 1/2, by = 1/length(data))
##     ln <- length(lambda)
##     sigma.table <- tau.matrix(length = items)
##     lapse.table <- t(matrix(rep(expected.tau(concordance = 0,
##         length = items), each = items), ncol = items, byrow = TRUE))
##     likelihood <- array(NA, c(items, items * (items - 1)/2 +
##         1, ln))
##     for (ctr in 1:ln) {
##         likelihood[, , ctr] <- lambda[ctr] * lapse.table + (1 -
##             lambda[ctr]) * sigma.table
##     }
##     log.likelihood <- log(likelihood)
##     prior.sigma <- norm(sigma.prior(sigma))
##     prior.lambda <- norm(lambda.prior(lambda))
```

```
##     prior.joint <- outer(prior.sigma, prior.lambda)
##     sum.log.likelihood <- apply(log.likelihood[, data + 1, ],
##         c(1, 3), sum)
##     log.posterior <- sum.log.likelihood + log(prior.joint)
##     posterior.joint <- norm(exp(log.posterior - median(log.posterior)))
##     posterior.lambda <- apply(posterior.joint, 2, sum)
##     posterior.sigma <- apply(posterior.joint, 1, sum)
##     posterior.ratio <- posterior.sigma[1]/sum(posterior.sigma[-1])
##     prior.ratio <- prior.sigma[1]/sum(prior.sigma[-1])
##     bayes.factor <- posterior.ratio/prior.ratio
##     return(list(joint.posterior = posterior.joint, lambda.posterior = posterior.lambda,
##         sigma.posterior = posterior.sigma, bayes.factor = bayes.factor))
## }
```