stat\_public\_func.R

Harper

Thu Sep 19 13:14:28 2019

###################################  
# Title : "Statistical Functions"  
# Author : Harper  
# Release : 20190915  
###################################  
  
# FUNCTION: Sample Statistics  
stat\_Sample <- function(sample\_data) {  
 stat\_ <- list(  
 "size" = length(sample\_data),   
 "mean" = mean(sample\_data),   
 "var" = var(sample\_data)  
 )  
 # return the statistics of the Sample  
 return (stat\_)  
}  
  
# FUNCTION: create the Sampling Distribution of the Mean  
dis\_Mean <- function(sample) {  
 samp\_stat <- stat\_Sample(sample)  
 # return the parameters of the sampling distribution  
 u <- samp\_stat$mean  
 df <- samp\_stat$size - 1  
 se <- sqrt(samp\_stat$var / samp\_stat$size)  
 dis\_mean <- list(  
 "u" = u,   
 "df" = df,   
 "se" = se  
 )  
 return (dis\_mean)  
}  
  
# FUNCTION: create the Sampling Distribution of the Difference between Means  
dis\_Diff\_means <- function(sample\_1, sample\_2) {  
 samp1\_stat <- stat\_Sample(sample\_1)  
 samp2\_stat <- stat\_Sample(sample\_2)  
   
 df <- samp1\_stat$size-1 + samp2\_stat$size-1  
 if (samp1\_stat$size != samp2\_stat$size) {  
 sse <- sum((sample\_1 - samp1\_stat$mean)^2) + sum((sample\_2 - samp2\_stat$mean)^2)  
 mse <- sse / df  
 n <- 2 / (1/samp1\_stat$size + 1/samp2\_stat$size)  
 } else {  
 mse <- (samp1\_stat$var + samp2\_stat$var) / 2  
 n <- samp1\_stat$size  
 }  
 se <- sqrt(2\*mse/n)  
 u <- samp1\_stat$mean - samp2\_stat$mean  
 # return parameters of the Sampling Distribution  
 dis\_diff <- list(  
 "u" = u,   
 "df" = df,   
 "mse" = mse,   
 "n" = n,   
 "se" = se  
 )  
 return (dis\_diff)  
}  
  
  
# FUNCTION: t-test on Sampling Distribution of Mean  
# Input :  
# sample (vector),   
# Null\_hypothesis value, one-/two-tailed, direction of one-tailed  
# Output:  
# t\_ {  
# t\_test { # t-test result }  
# dis\_mean { # parameters of the sampling distribution }  
# }  
t\_Mean\_run <- function(sample, h0\_u=0, tailed=2, direction=1) {  
 if ((tailed!=1)&&(tailed!=2) ||   
 ((direction!=1)&&(direction!=-1)&&(direction!=0)))   
 { return (NULL) }  
   
 dis\_mean <- dis\_Mean(sample)  
 t\_t <- (dis\_mean$u - h0\_u) / dis\_mean$se  
 if (tailed==2) { # Two-tailed  
 t\_p <- 2 \* pt(abs(t\_t), dis\_mean$df, lower.tail=FALSE)  
 } else { # One-tailed  
 if (direction==1) { # positive direction  
 t\_p <- pt(t\_t, dis\_mean$df, lower.tail=FALSE)  
 } else if (direction==-1) { # negative direction  
 t\_p <- pt(t\_t, dis\_mean$df, lower.tail=TRUE)  
 } else { # either positive or negative direction  
 t\_p <- pt(abs(t\_t), dis\_mean$df, lower.tail=FALSE)  
 }  
 }  
 # return t-test result and the sampling distribution parameters  
 t\_test <- list(  
 "t" = t\_t,   
 "p" = t\_p  
 )  
 t\_ <- list(  
 "t\_test" = t\_test,   
 "dis\_mean" = dis\_mean  
 )  
 return (t\_)  
}  
  
  
# FUNCTION: t-test on Sampling Distribution of the Difference between Means  
# Input :  
# sample1 (vector), sampe2 (vector),   
# Null\_hypothesis value, one-/two-tailed, direction of one-tailed  
# Output:  
# t\_ {  
# t\_test { # t-test result }  
# dis\_diff { # parameters of the sampling distribution }  
# }  
t\_Diff\_means\_run <- function(sample\_1, sample\_2, h0\_diff\_u=0, tailed=2, direction=1) {  
 if ((tailed!=1)&&(tailed!=2) ||   
 ((direction!=1)&&(direction!=-1)&&(direction!=0)))   
 { return (NULL) }  
   
 dis\_diff <- dis\_Diff\_means(sample\_1, sample\_2)  
 t\_t <- (dis\_diff$u - h0\_diff\_u) / dis\_diff$se  
 if (tailed==2) { # Two-tailed  
 t\_p <- 2 \* pt(abs(t\_t), dis\_diff$df, lower.tail=FALSE)  
 } else { # One-tailed  
 if (direction==1) { # positive direction  
 t\_p <- pt(t\_t, dis\_diff$df, lower.tail=FALSE)  
 } else if (direction==-1) { # negative direction  
 t\_p <- pt(t\_t, dis\_diff$df, lower.tail=TRUE)  
 } else { # either positive or negative direction  
 t\_p <- pt(abs(t\_t), dis\_diff$df, lower.tail=FALSE)  
 }  
 }  
 # return t-test result and the sampling distribution parameters  
 t\_test <- list(  
 "t" = t\_t,   
 "p" = t\_p  
 )  
 t\_ <- list(  
 "t\_test" = t\_test,   
 "dis\_diff" = dis\_diff  
 )  
 return (t\_)  
}  
  
  
# FUNCTION: Confidence Interval for Mean (or Means)  
# Assuming “homogeneity of variance”; both gender means are normal distributed; and each score is independent  
# Input :  
# mean, standard error, degree of freedom, Confidence Interval level  
# Output:  
# CI\_ { # the Lower and Upper Critical Values of mean }  
CI\_Mean <- function(u, se, df, ci=0.95, tailed=2, direction=1) {  
 if (((ci<0)||(ci>1)) ||   
 (tailed!=1)&&(tailed!=2) ||   
 ((direction!=1)&&(direction!=-1)&&(direction!=0)))   
 { return (NULL) }  
   
 if (tailed==2) { # Two-tailed  
 t\_crit <- qt((1-ci)/2, df, lower.tail=FALSE)  
 ci\_L <- u - se \* t\_crit  
 ci\_U <- u + se \* t\_crit  
 } else { # One-tailed  
 if (direction==1) { # positive direction  
 t\_crit <- qt(1-ci, df, lower.tail=FALSE)  
 ci\_L <- u + se \* t\_crit  
 ci\_U <- NULL  
 } else if (direction==-1) { # negative direction  
 t\_crit <- qt(ci-1, df, lower.tail=TRUE)  
 ci\_L <- NULL  
 ci\_U <- u - se \* t\_crit  
 } else { # either positive or negative direction  
 t\_crit <- qt(1-ci, df, lower.tail=FALSE)  
 ci\_L <- u - se \* t\_crit  
 ci\_U <- u + se \* t\_crit  
 }  
 }  
 # return the Confidence Interval  
 CI\_ <- list(  
 "t\_crit" = t\_crit,   
 "ci\_L" = ci\_L,   
 "ci\_U" = ci\_U   
 )  
 return (CI\_)  
}  
  
  
# FUNCTION: Confidence Interval calculation for Pearson Correaltion  
# Input :  
# sample1 (vector), sampe2 (vector), Confidence Interval level  
# Output:  
# CI\_r { # the Lower and Upper limits of r }  
CI\_Correlation <- function(sample\_1, sample\_2, ci=0.95) {  
 n <- length(sample\_1)  
 if ((length(sample\_2)!=n) || ((ci<0)||(ci>1))) { return (NULL) }  
   
 # Correlation r  
 r <- cor(sample\_1, sample\_2)  
 # Fisher’s Transformation to z'  
 z\_ <- log((1.0+r)/(1.0-r)) / 2.0  
 z\_se <- 1.0 / sqrt(n-3.0)  
 # z' Critical Value  
 z\_crit <- qnorm((1.0-ci)/2, mean=0, sd=1, lower.tail=FALSE)  
 # CI on z'  
 z\_L <- z\_ - z\_crit \* z\_se  
 z\_U <- z\_ + z\_crit \* z\_se  
 # Fisher's Transformation back to r  
 r\_L <- exp(2\*z\_L)  
 r\_L <- (r\_L-1)/(r\_L+1)  
 r\_U <- exp(2\*z\_U)  
 r\_U <- (r\_U-1)/(r\_U+1)  
   
 CI\_r <- list(  
 "r\_L" = r\_L,   
 "r\_U" = r\_U  
 )  
 return (CI\_r)  
}