Error Rate Validation

```r
library(knitr)
opts_chunk$set(tidy.opts=list(width.cutoff=60),tidy=TRUE)
```

Validation error rate dependent t-test

We can simulate the Type 1 error rates for equivalence tests using the TOST procedure based on raw scores, when we set the true effect size to one of the equivalence bounds. Note that error rates at 5% max, but it is possible for error rates to be much lower, for example when the sample size is so small, that confidence intervals are always wider than the equivalence bounds (e.g., n = 10, low_eqbound = -0.05, high_eqbound = 0.05). Furthermore, error rates will be lower then the true effect size is larger than the upper equivalence bound, or smaller than the lower equivalence bound.

```r
require(TOSTER)
```

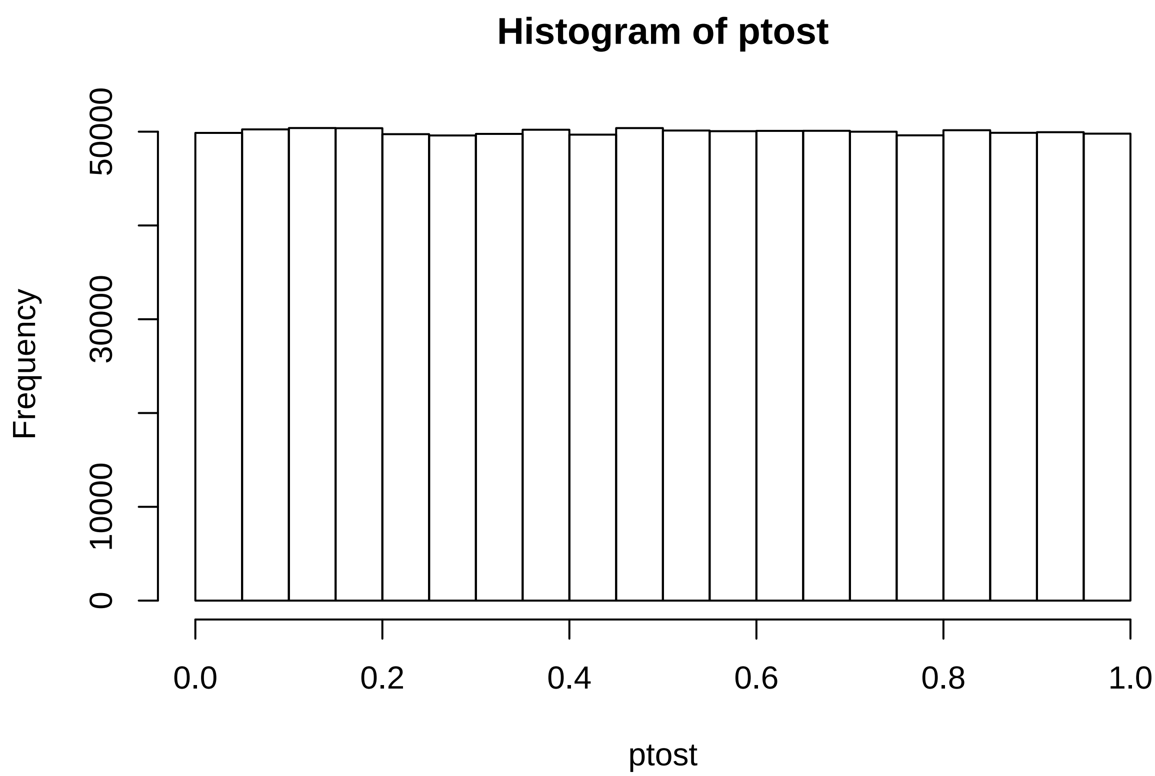
```
## Loading required package: TOSTER
```

```r
library(mvtnorm)  #to simulate correlated data
n <- 100  #set sample size
r <- 0.5  #set correlation
low_eqbound = -0.5
high_eqbound = 0.5
sd <- 1
m <- 0.5
alpha <- 0.05
nSims <- 1e+06  #number of simulated experiments
pttest <- numeric(nSims)  #set up empty container for all simulated p-values
ptost <- numeric(nSims)  #set up empty container for all simulated p-values
tlist <- numeric(nSims)  #set up empty container for all simulated p-values
p1list <- numeric(nSims)  #set up empty container for all simulated p-values
p2list <- numeric(nSims)  #set up empty container for all simulated p-values
dz <- numeric(nSims)  #set up empty container for all simulated p-values
rlist <- numeric(nSims)
LLlist <- numeric(nSims)
ULlist <- numeric(nSims)

for (i in 1:nSims) {
    # for each simulated experiment
    a <- rmvnorm(n = n, mean = c(m, 0), sigma = matrix(c(sd,
        r, r, sd), 2, 2))
    x <- a[, 1]
    y <- a[, 2]
    m1 <- mean(x)
    m2 <- mean(y)
    sd1 <- sd(x)
    sd2 <- sd(y)
    r12 <- cor(x, y)
    rlist[i] <- r12
    sdif <- sqrt(sd1^2 + sd2^2 - 2 * r12 * sd1 * sd2)
    dz[i] <- (m1 - m2)/sdif
    se <- sdif/sqrt(n)
```

```
    t <- (m1 - m2)/se
    tlist[i] <- t
    degree_f <- n - 1
    pttest <- 2 * pt(abs(t), degree_f, lower = FALSE)
    t1 <- ((m1 - m2) + (low_eqbound))/se
    p1 <- 1 - pt(t1, degree_f, lower = FALSE)
    p1list[i] <- p1
    t2 <- ((m1 - m2) + (high_eqbound))/se
    p2 <- pt(t2, degree_f, lower = FALSE)
    p2list[i] <- p2
    ttost <- ifelse(abs(t1) < abs(t2), t1, t2)
    LL90 <- ((m1 - m2) - qt(1 - alpha, degree_f) * se)
    LLlist[i] <- LL90
    UL90 <- ((m1 - m2) + qt(1 - alpha, degree_f) * se)
    ULlist[i] <- UL90
    ptost[i] <- max(p1, p2)
}

# P-value distribution TOST
hist(ptost, breaks = 20, xlim = c(0, 1))
```




```
sum(ptost < 0.05)/nSims  #Power for TOST
```

```
## [1] 0.049869
```

```
sum(LLlist >= low_eqbound & ULlist <= high_eqbound)/nSims  #Same power or Type 1 error, now based on CI
```

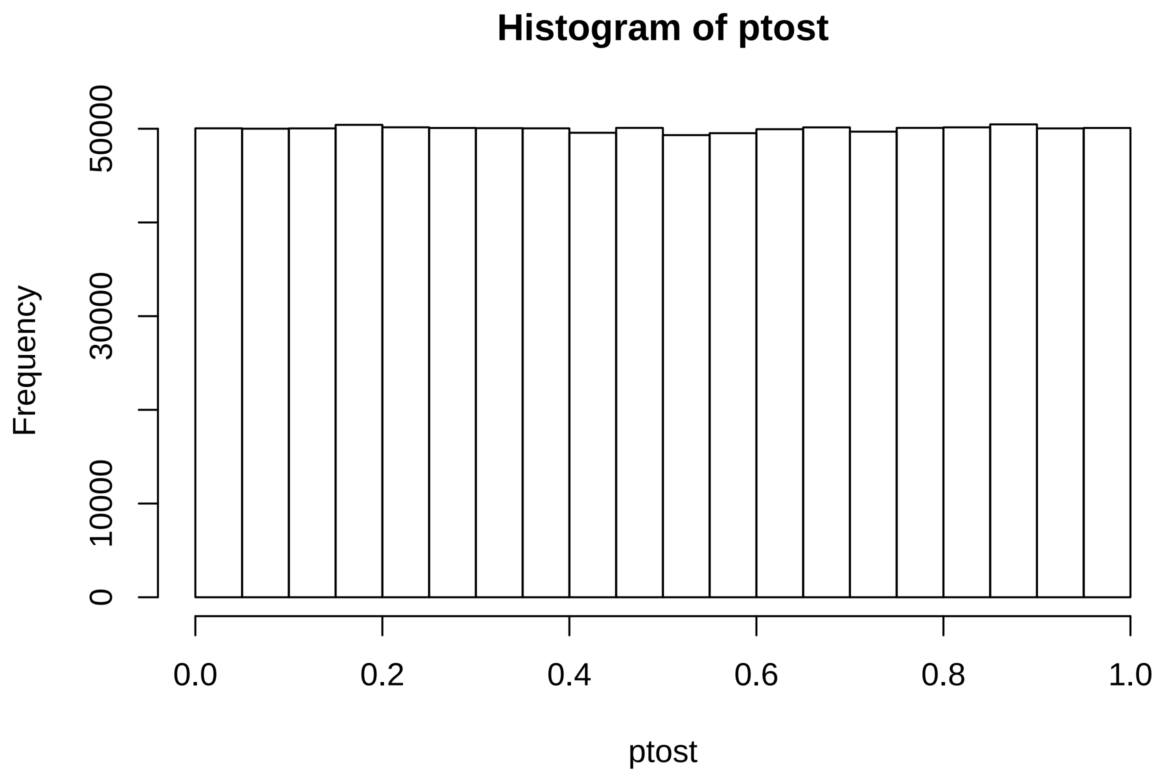
```
## [1] 0.049869
```

Validation error rate one-sample t-test

```r
require(TOSTER)
n <- 100  #set sample size
low_eqbound = -0.5
high_eqbound = 0.5
sd_true <- 1
m_true <- 0.5
mu <- 0
alpha <- 0.05
nSims <- 1e+06  #number of simulated experiments
pttest <- numeric(nSims)  #set up empty container for all simulated p-values
ptost <- numeric(nSims)  #set up empty container for all simulated p-values
tlist <- numeric(nSims)  #set up empty container for all simulated p-values
p1list <- numeric(nSims)  #set up empty container for all simulated p-values
p2list <- numeric(nSims)  #set up empty container for all simulated p-values
LLlist <- numeric(nSims)
ULlist <- numeric(nSims)

for (i in 1:nSims) {
    # for each simulated experiment
    x <- rnorm(n = n, mean = m_true, sd = sd_true)  #Simulate data with specified mean, standard deviat
    m <- mean(x)
    sd <- sd(x)
    degree_f <- n - 1
    t1 <- (m - mu - low_eqbound)/(sd/sqrt(n))  # t-test
    p1 <- pt(t1, degree_f, lower = FALSE)
    p1list[i] <- p1
    t2 <- (m - mu - high_eqbound)/(sd/sqrt(n))  #t-test
    p2 <- pt(t2, degree_f, lower = TRUE)
    p2list[i] <- p2
    t <- (m - mu)/(sd/sqrt(n))
    tlist[i] <- t
    pttest <- 2 * pt(-abs(t), df = degree_f)
    LL90 <- (m - mu) - qt(1 - alpha, degree_f) * (sd/sqrt(n))
    LLlist[i] <- LL90
    UL90 <- (m - mu) + qt(1 - alpha, degree_f) * (sd/sqrt(n))
    ULlist[i] <- UL90
    ptost[i] <- max(p1, p2)
    ttost <- ifelse(abs(t1) < abs(t2), t1, t2)  #Get lowest t-value for summary TOST result
}

# P-value distribution TOST
hist(ptost, breaks = 20, xlim = c(0, 1))
```




```
sum(ptost < 0.05)/nSims  #Power for TOST
```

```
## [1] 0.050047
```

```
sum(LLlist >= low_eqbound & ULlist <= high_eqbound)/nSims  #Same power or Type 1 error, now based on CI
```

```
## [1] 0.050047
```

Validation error rate independent t-test

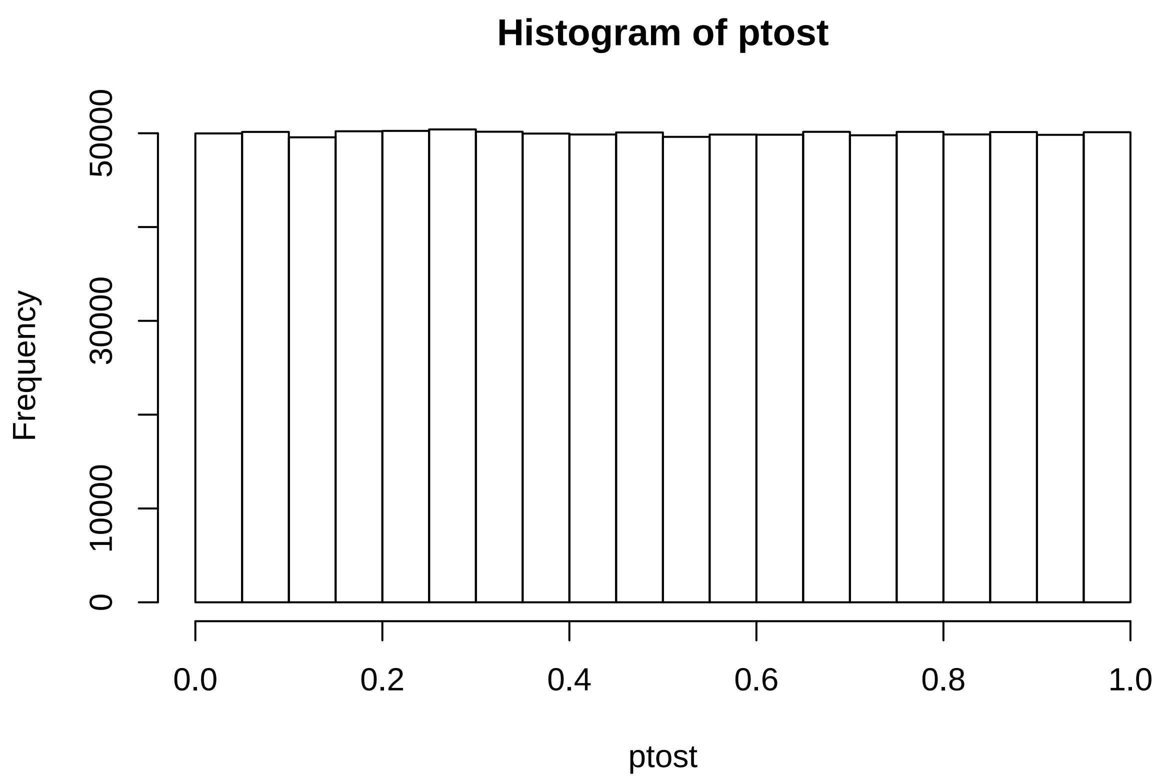
```
require(TOSTER)
n1 <- 100  #set sample size
n2 <- 100  #set sample size
low_eqbound = -0.5
high_eqbound = 0.5
m1_true <- 0.5
m2_true <- 0
sd1_true <- 1
sd2_true <- 1
alpha <- 0.05
var.equal <- TRUE
nSims <- 1e+06  #number of simulated experiments
pttest <- numeric(nSims)  #set up empty container for all simulated p-values
ptost <- numeric(nSims)  #set up empty container for all simulated p-values
tlist <- numeric(nSims)  #set up empty container for all simulated p-values
```

```r
p1list <- numeric(nSims)  #set up empty container for all simulated p-values
p2list <- numeric(nSims)  #set up empty container for all simulated p-values
LLlist <- numeric(nSims)
ULlist <- numeric(nSims)

for (i in 1:nSims) {
    # for each simulated experiment
    x <- rnorm(n = n1, mean = m1_true, sd = sd1_true)  #Simulate data with specified mean, standard dev
    y <- rnorm(n = n2, mean = m2_true, sd = sd2_true)  #Simulate data with specified mean, standard   m
    m1 <- mean(x)
    m2 <- mean(y)
    sd1 <- sd(x)
    sd2 <- sd(y)
    if (var.equal == TRUE) {
        sdpooled <- sqrt((((n1 - 1) * (sd1^2)) + (n2 - 1) * (sd2^2))/((n1 +
            n2) - 2))  #calculate sd pooled
        t1 <- (abs(m1 - m2) - low_eqbound)/(sdpooled * sqrt(1/n1 +
            1/n2))
        degree_f <- n1 + n2 - 2
        p1 <- pt(t1, degree_f, lower = FALSE)
        t2 <- (abs(m1 - m2) - high_eqbound)/(sdpooled * sqrt(1/n1 +
            1/n2))
        p2 <- pt(t2, degree_f, lower = TRUE)
        LL90 <- (m1 - m2) - qt(1 - alpha, degree_f) * (sdpooled *
            sqrt(1/n1 + 1/n2))
        UL90 <- (m1 - m2) + qt(1 - alpha, degree_f) * (sdpooled *
            sqrt(1/n1 + 1/n2))
        LL95 <- (m1 - m2) - qt(1 - (alpha/2), degree_f) * (sdpooled *
            sqrt(1/n1 + 1/n2))
        UL95 <- (m1 - m2) + qt(1 - (alpha/2), degree_f) * (sdpooled *
            sqrt(1/n1 + 1/n2))
        t <- (m1 - m2)/(sdpooled * sqrt(1/n1 + 1/n2))
        pttest <- 2 * pt(-abs(t), df = degree_f)
    } else {
        sdpooled <- sqrt((sd1^2 + sd2^2)/2)  #calculate sd root mean squared for Welch's t-test
        t1 <- (abs(m1 - m2) - low_eqbound)/sqrt(sd1^2/n1 + sd2^2/n2)  #welch's t-test lower bound
        degree_f <- (sd1^2/n1 + sd2^2/n2)^2/(((sd1^2/n1)^2/(n1 -
            1)) + ((sd2^2/n2)^2/(n2 - 1)))  #degrees of reedom for Welch's t-test
        p1 <- pt(t1, degree_f, lower = FALSE)  #p-value for Welch's TOST t-test
        t2 <- (abs(m1 - m2) - high_eqbound)/sqrt(sd1^2/n1 + sd2^2/n2)  #welch's t-test upper bound
        p2 <- pt(t2, degree_f, lower = TRUE)  #p-value for Welch's TOST t-test
        t <- (m1 - m2)/sqrt(sd1^2/n1 + sd2^2/n2)  #welch's t-test NHST
        pttest <- 2 * pt(-abs(t), df = degree_f)  #p-value for Welch's t-test
        LL90 <- (m1 - m2) - qt(1 - alpha, degree_f) * sqrt(sd1^2/n1 +
            sd2^2/n2)  #Lower limit for CI Welch's t-test
        UL90 <- (m1 - m2) + qt(1 - alpha, degree_f) * sqrt(sd1^2/n1 +
            sd2^2/n2)  #Upper limit for CI Welch's t-test
    }
    p1list[i] <- p1
    p2list[i] <- p2
    tlist[i] <- t
    LLlist[i] <- LL90
    ULlist[i] <- UL90
```

```r
    ptost[i] <- max(p1, p2)
}

# P-value distribution TOST
hist(ptost, breaks = 20, xlim = c(0, 1))
```




```r
sum(ptost <= 0.05)/nSims  #Power for TOST
```

```
## [1] 0.049983
```

```r
sum(LLlist >= low_eqbound & ULlist <= high_eqbound)/nSims  #Same power or Type 1 error, now based on CI
```

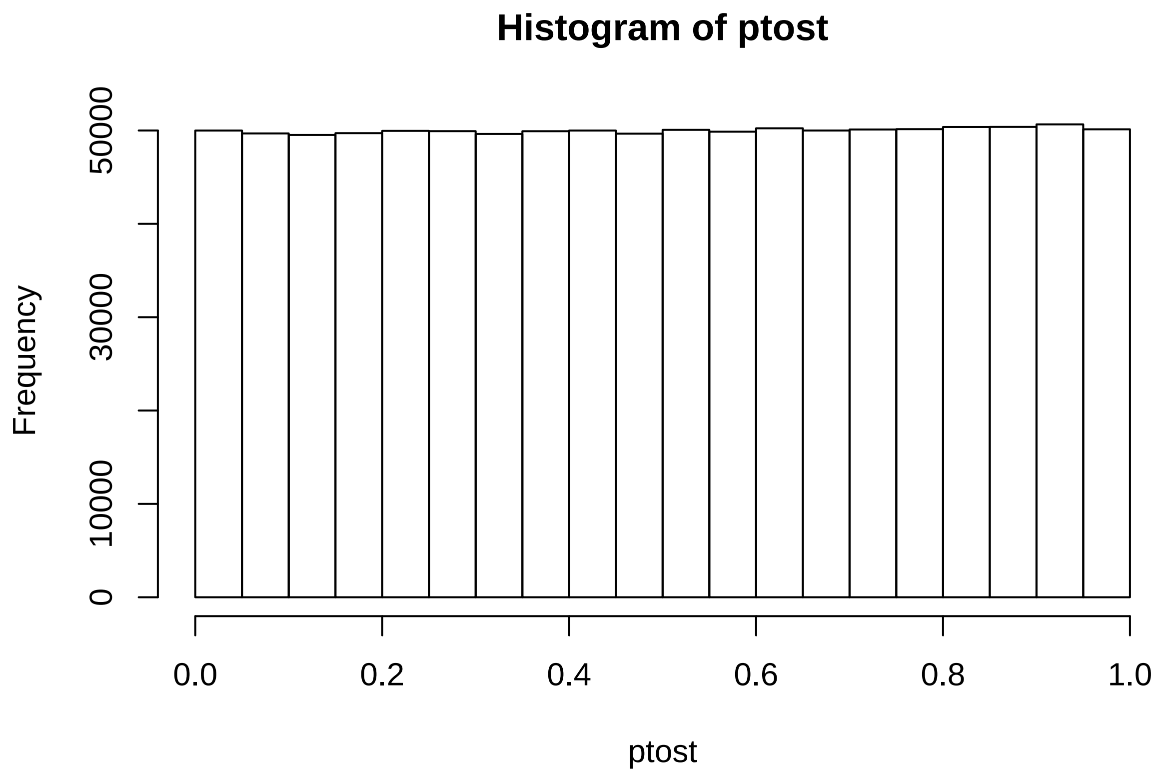
```
## [1] 0.049983
```

Validation error rate correlations

```r
library(mvtnorm)  #to simulate correlated data
n <- 1000  #set sample size
m <- 0
sd <- 1
alpha <- 0.05
r_true <- -0.25  #set correlation
low_eqbound_r <- -0.25
high_eqbound_r <- 0.25
nSims <- 1e+06
```

```r
ptost <- numeric(nSims)  #set up empty container for all simulated p-values
p1list <- numeric(nSims)  #set up empty container for all simulated p-values
p2list <- numeric(nSims)  #set up empty container for all simulated p-values
LLlist <- numeric(nSims)
ULlist <- numeric(nSims)
rlist <- numeric(nSims)
for (i in 1:nSims) {
    # for each simulated experiment
    a <- rmvnorm(n = n, mean = c(m, 0), sigma = matrix(c(sd,
        r_true, r_true, sd), 2, 2))
    x <- a[, 1]
    y <- a[, 2]
    r <- cor(x, y)
    rlist[i] <- r
    z1 <- ((log((1 + abs(r))/(1 - abs(r)))/2) + (log((1 + low_eqbound_r)/(1 -
        low_eqbound_r))/2))/(sqrt(1/(n - 3)))
    z2 <- ((log((1 + abs(r))/(1 - abs(r)))/2) + (log((1 + high_eqbound_r)/(1 -
        high_eqbound_r))/2))/(sqrt(1/(n - 3)))
    p1 <- ifelse(low_eqbound_r < r, pnorm(-abs(z1)), 1 - pnorm(-abs(z1)))
    p2 <- ifelse(high_eqbound_r > r, pnorm(-abs(z2)), 1 - pnorm(-abs(z2)))
    ptost[i] <- max(p1, p2)
    pttest <- 2 * (1 - pt(abs(r) * sqrt(n - 2)/sqrt(1 - abs(r)^2),
        n - 2))
    zLL90 <- (log((1 + r)/(1 - r))/2) - qnorm(1 - alpha) * sqrt(1/(n -
        3))
    zUL90 <- (log((1 + r)/(1 - r))/2) + qnorm(1 - alpha) * sqrt(1/(n -
        3))
    LL90 <- (exp(1)^(2 * zLL90) - 1)/(exp(1)^(2 * zLL90) + 1)
    UL90 <- (exp(1)^(2 * zUL90) - 1)/(exp(1)^(2 * zUL90) + 1)
    p1list[i] <- p1
    p2list[i] <- p2
    LLlist[i] <- LL90
    ULlist[i] <- UL90
}

# P-value distribution TOST
hist(ptost, breaks = 20, xlim = c(0, 1))
```




```r
sum(ptost < 0.05)/nSims  #Power for TOST
```

```
## [1] 0.049995
```

```r
sum(LLlist >= low_eqbound_r & ULlist <= high_eqbound_r)/nSims  #Same power or Type 1 error, now based o
```

```
## [1] 0.049995
```

```r
mean(rlist)
```

```
## [1] -0.24988
```