One-sample tests: false positive & power analysis [1]

Rand R. Wilcox & Guillaume A. Rousselet

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code. Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
# dependencies
library(ggplot2)
library(cowplot)
library(tibble)
library(tidyr)
source("Rallfun-v34.txt")
```

Simple example of power analysis

We draw samples of n = 30 observations from a lognormal distribution with a mean of 0.5 or a 20% trimmed mean of 0.5. If you want to run your own power analysis, you need to consider all these choices: the shape of the distribution to sample from, the sample size, the effect size, the statistical tests. With the current selection of paramaters, we simply wanted to illustrate that when sampling from a skewed distribution with a relatively large sample size in psychology and neuroscience, the choice of statistical test can have large effects on power. In particular, a t-test on the mean can have very low power, whereas a t-test on a trimmed mean, or a test on the median can provide much larger power.

```
iter = 5000 # number of simulation iterations
g = 1 # paramaters to specify a lognormal distribution in ghdist()
h = 0
es = 0.5 # specify effect size
npm = 0 # number of positive results for the mean
nptm = 0 # number of positive results for the trimmed mean
npmd = 0 # number of positive results for the median
set.seed(44) # set number generator for reproducible results
mu = ghmean(1,0)$mean # mean of the lognormal distribution
tmu = ghtrim(g=1) # trimmed mean of the lognormal distribution

for(i in 1:iter){ # simulation loop
  x = ghdist(30, g=g, h=h) # sample 30 observations from a lognormal distribution
  # one-sample t-tests on samples from lognormal distributions with:
  pm = trimci(x - mu + es, pr=F, tr=0)$p.value # t-test on mean
  ptm = trimci(x - tmu + es, pr=F)$p.value # t-test on 20% trimmed mean
  # trimci estimates the 20% trimmed mean by default
  pmd = sintv2(x + es, pr=FALSE)$p.value # test on median
  if(pm<=.05) npm = npm + 1 # mean
  if(ptm<=.05) nptm = nptm + 1 # 20% trimmed mean
  if(pmd<=.05) npmd = npmd + 1 # median
}
print(c(npm/i,nptm/i,npmd/i))
```

```
## [1] 0.1526 0.6534 0.8538
```

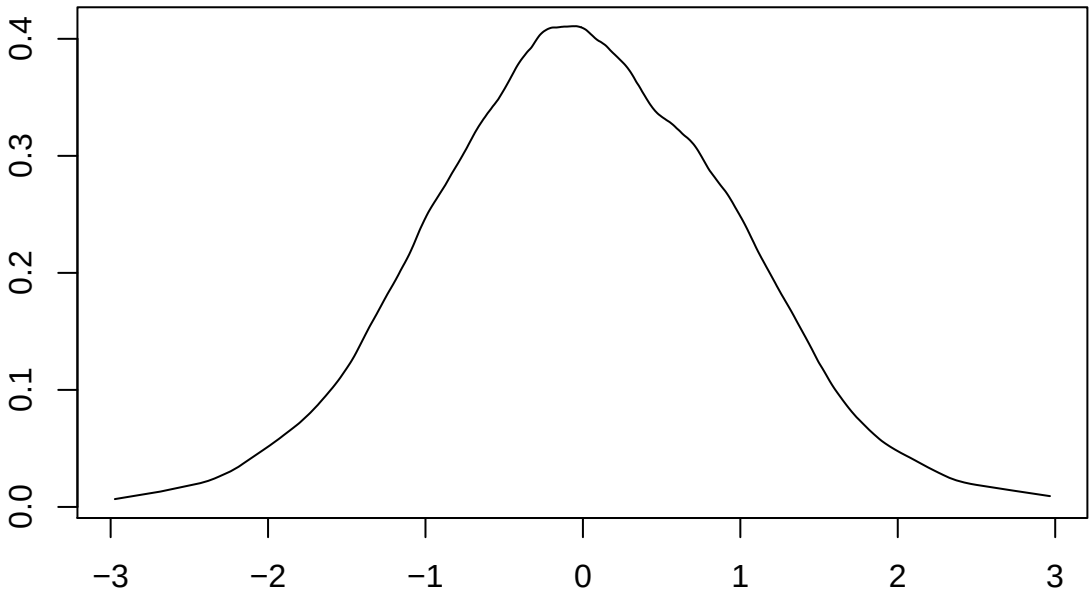
The results show power of 15% for the mean, 65% for 20% trimmed mean, and 85% for the median.

The `ghdist()` function is used to generate random numbers from g & h distributions. These distributions

have a median of zero. The parameter `g` controls the asymmetry of the distribution, while the parameter `h` controls the tails. Try out the following the examples:

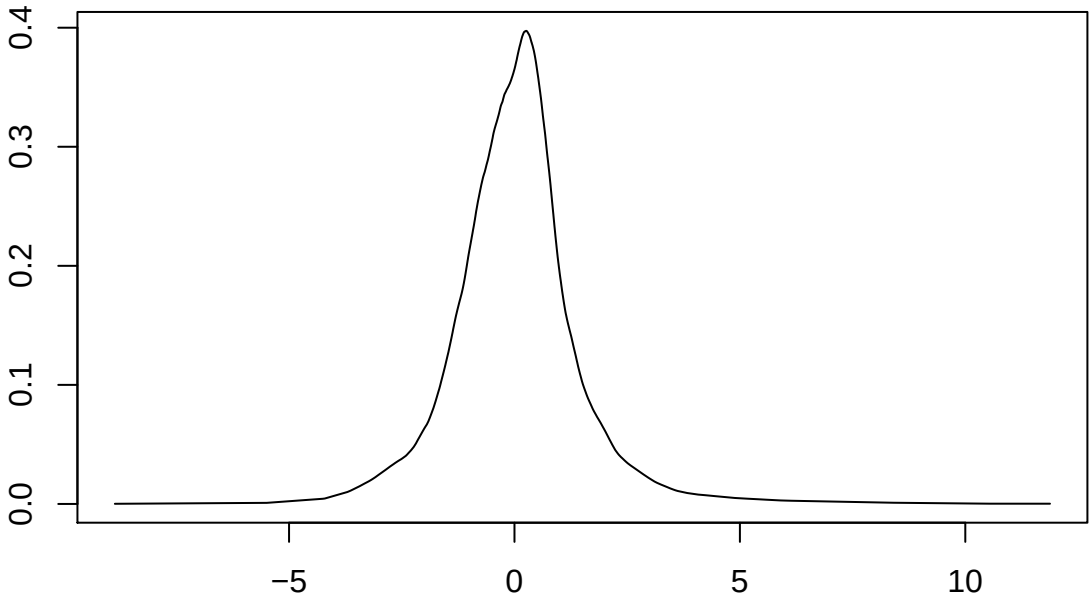
- the standard normal (g = h = 0);
- a symmetric heavy-tailed distribution (h = 0.2, g = 0.0);
- an asymmetric distribution with relatively light tails (h = 0.0, g = 0.2);
- an asymmetric distribution with heavy tails (g = h = 0.2).

```
set.seed(7)
# Sample from a normal distribution
akerd(ghdist(1000, g = 0, h = 0))
```



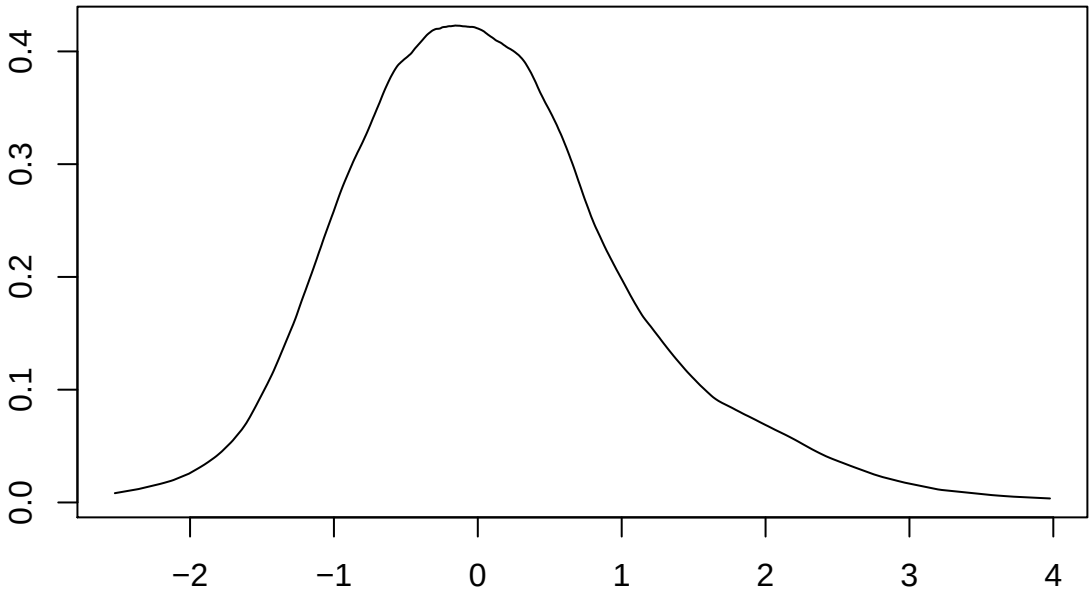

```
## [1] "Done"
```

```
# Sample from a symmetric heavy-tailed distribution
akerd(ghdist(1000, g = 0, h = 0.2))
```



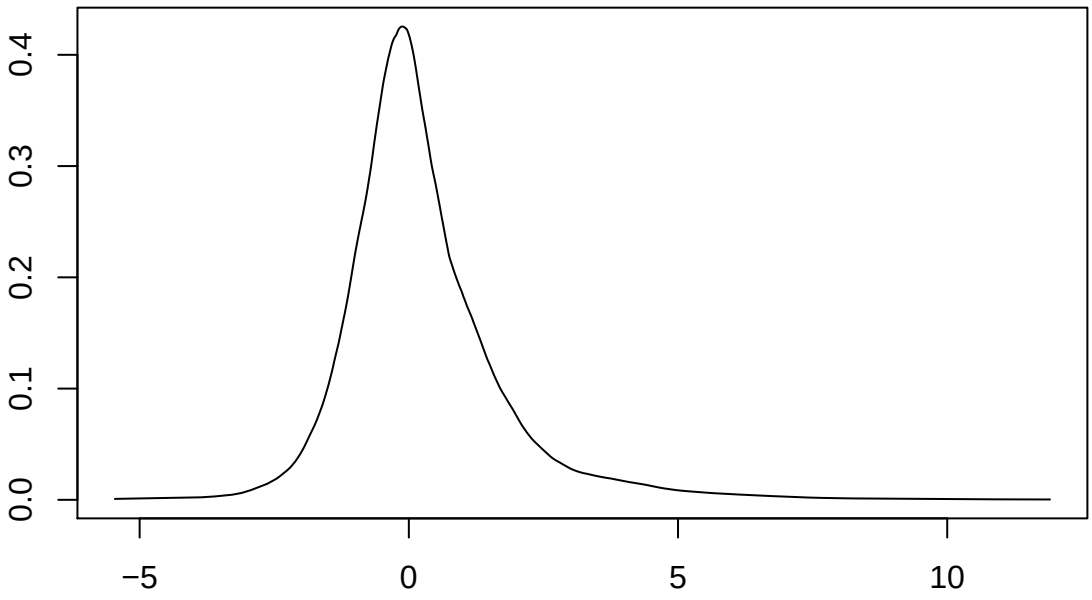

```
## [1] "Done"
```

```r
# Sample from an asymmetric distribution with relatively light tails
akerd(ghdist(1000, g = 0.2, h = 0))
```




```
## [1] "Done"
```

```r
# Sample from an asymmetric distribution with heavy tails
akerd(ghdist(1000, g = 0.4, h = 0.2))
```




```
## [1] "Done"
```

Full simulation

Here we perform a large simulation. Before you commit to 10,000 simulations, try with 1,000, because it might take a long time to compute. The code will output an iteration update every 100 simulations. The simulation is an extension of the previous one and includes: - multiple sample sizes - sampling from normal

and lognormal distributions - t-tests using means and 20% trimmed means - different effect sizes. We include an effect size of zero, to assess the type I error rate, or false positives, of the tests.

```
# Define parameters
npower <- 10000 # number of simulations
nvec <- seq(10,500,10) # vector of sample sizes to test
max.size <- max(nvec)
esvec <- c(0, 0.3, 0.5) # vector of effect sizes
```

To run the chunk below when knitting the notebook, you need to change `{r eval = FALSE}` to `{r eval = TRUE}`. You could run a faster simulation by changing `npower` to 1000 in the previous chunk for instance.

```
# 6 conditions: normal / skewed x mean / trimmed mean / median
simres <- array(0, dim = c(6, length(esvec), length(nvec), npower))
mu <- ghmean(1,0)$mean # mean of lognormal distribution
tmu <- ghtrim(g=1) # trimmed mean of lognormal distribution
set.seed(45) # set random number generator for reproducibility
for(S in 1:npower){ # simulation iterations
  if(S %% 100 == 0){
  print(S)
  }
  large.norm.sample <- ghdist(max.size, g = 0, h = 0)
  large.lnorm.sample <- ghdist(max.size, g = 1, h = 0)
  for(N in 1:length(nvec)){ # sample sizes
    for(E in 1:length(esvec)){ # effect sizes
      # sub-sample + shift by effect size
      norm.sample <- large.norm.sample[1:nvec[N]] + esvec[E]
      lnorm.sample <- large.lnorm.sample[1:nvec[N]] + esvec[E]
      # normal: t-test on mean
      simres[1,E,N,S] <- trimci(norm.sample, tr=0, pr=FALSE)$p.value
      # normal: t-test on 20% trimmed mean
      simres[2,E,N,S] <- trimci(norm.sample, tr=0.2, pr=FALSE)$p.value
      # normal: median test
      simres[3,E,N,S] <- sintv2(norm.sample, pr=FALSE)$p.value
      # lognormal: t-test on mean - subtract mu so the mean is zero on average
      simres[4,E,N,S] <- trimci(lnorm.sample - mu, tr=0, pr=FALSE)$p.value
      # lognormal: t-test on 20% trimmed mean - subtract tmu so the trimmed mean is zero on average
      simres[5,E,N,S] <- trimci(lnorm.sample - tmu, tr=0.2, pr=FALSE)$p.value
      # lognormal: median test
      # ghdist() generates data from distributions with median = 0, so no need to subtract the median
      simres[6,E,N,S] <- sintv2(lnorm.sample, pr=FALSE)$p.value
    }
  }
}
```

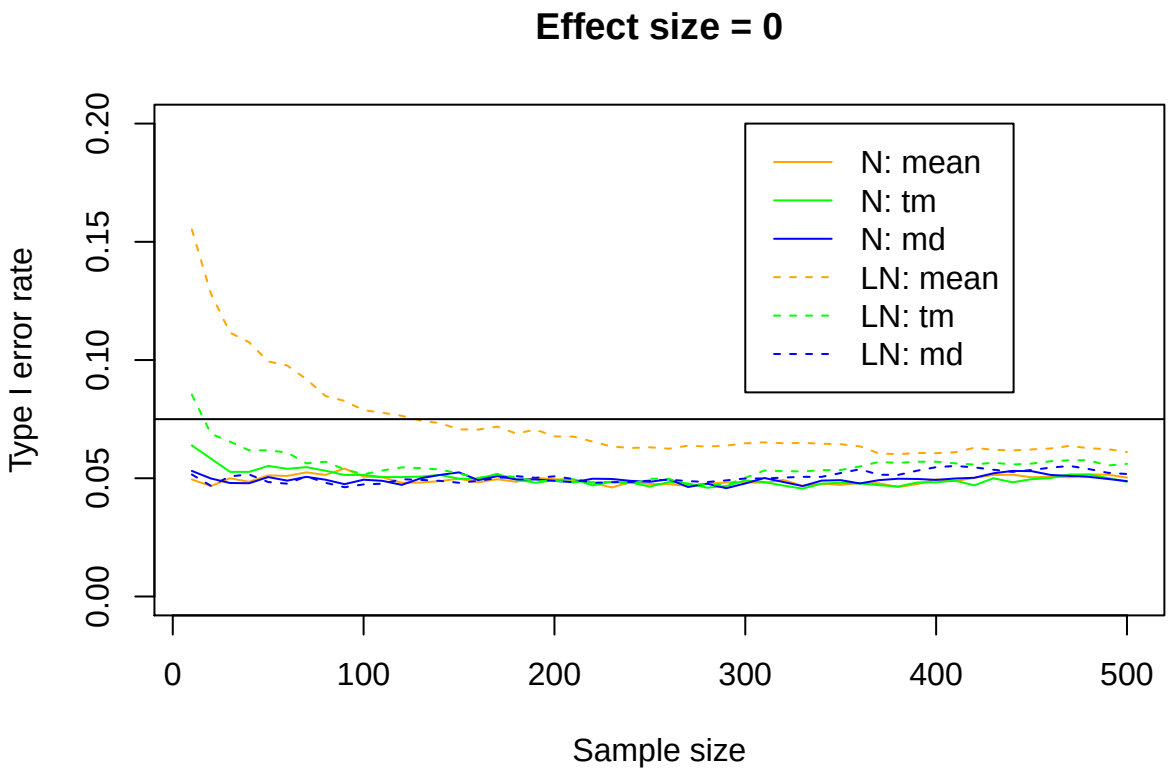
We get the type I error rate and power for each combination of tests, sample sizes and effect sizes.

```
# if you run a proper simulation, with say 10,000 iterations, uncomment the next two lines, and comment

# power.res <- apply(simres <= 0.05, c(1,2,3), mean)
# save(power.res, file = "power_res_tm.RData")

load("power_res_tm.RData")
```

Figures using base R

Type I error rate

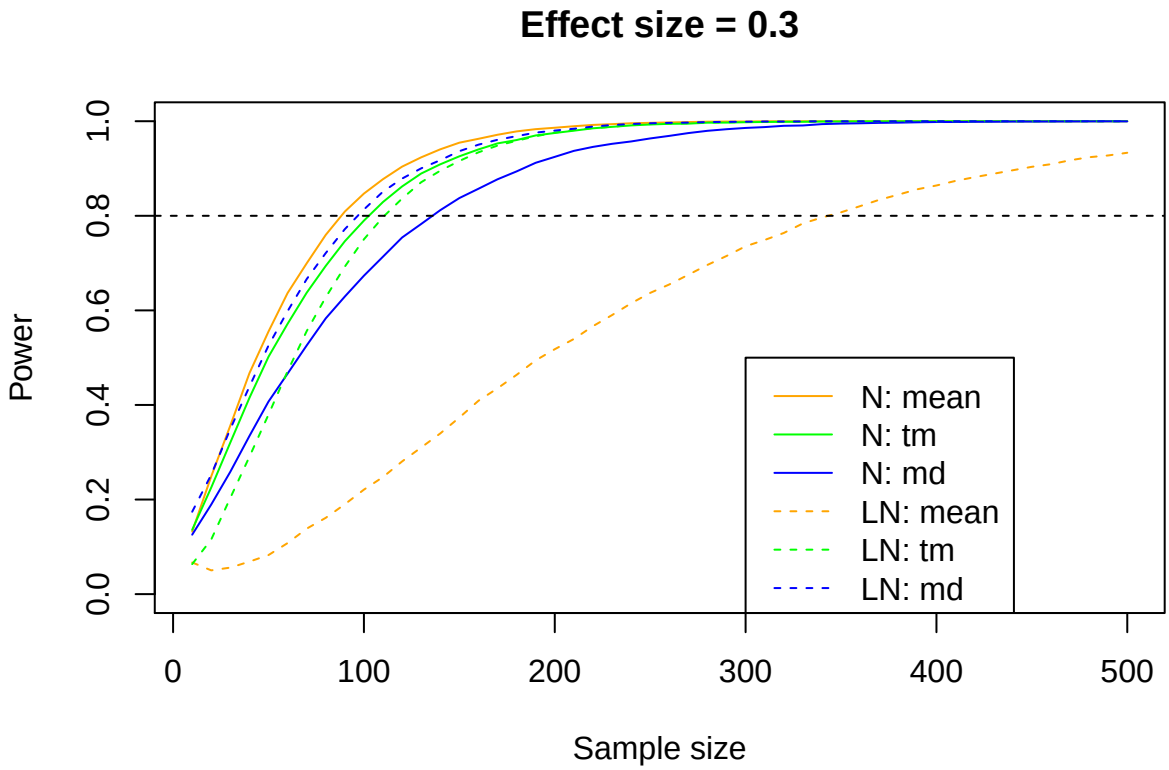
```
E <- 1
plot(nvec, seq(0, 0.2, length.out=length(nvec)), xlab='Sample size', ylab='Type I error rate', type='n')
lines(nvec, power.res[1,E,], lty=1, col="orange") # normal: t-test on mean
lines(nvec, power.res[2,E,], lty=1, col="green") # normal: t-test on 20% trimmed mean
lines(nvec, power.res[3,E,], lty=1, col="blue") # normal: median test
lines(nvec, power.res[4,E,], lty=2, col="orange") # lognormal: t-test on mean
lines(nvec, power.res[5,E,], lty=2, col="green") # lognormal: t-test on 20% trimmed mean
lines(nvec, power.res[6,E,], lty=2, col="blue") # lognormal: median test
abline(0.075, 0, lty=1)
legend(300, 0.2,
       c("N: mean", "N: tm", "N: md", "LN: mean", "LN: tm", "LN: md"),
       lty = c(1,1,1,2,2,2),
       col = c("orange","green","blue","orange","green","blue"))
title(paste0("Effect size = ",esvec[E]))
```

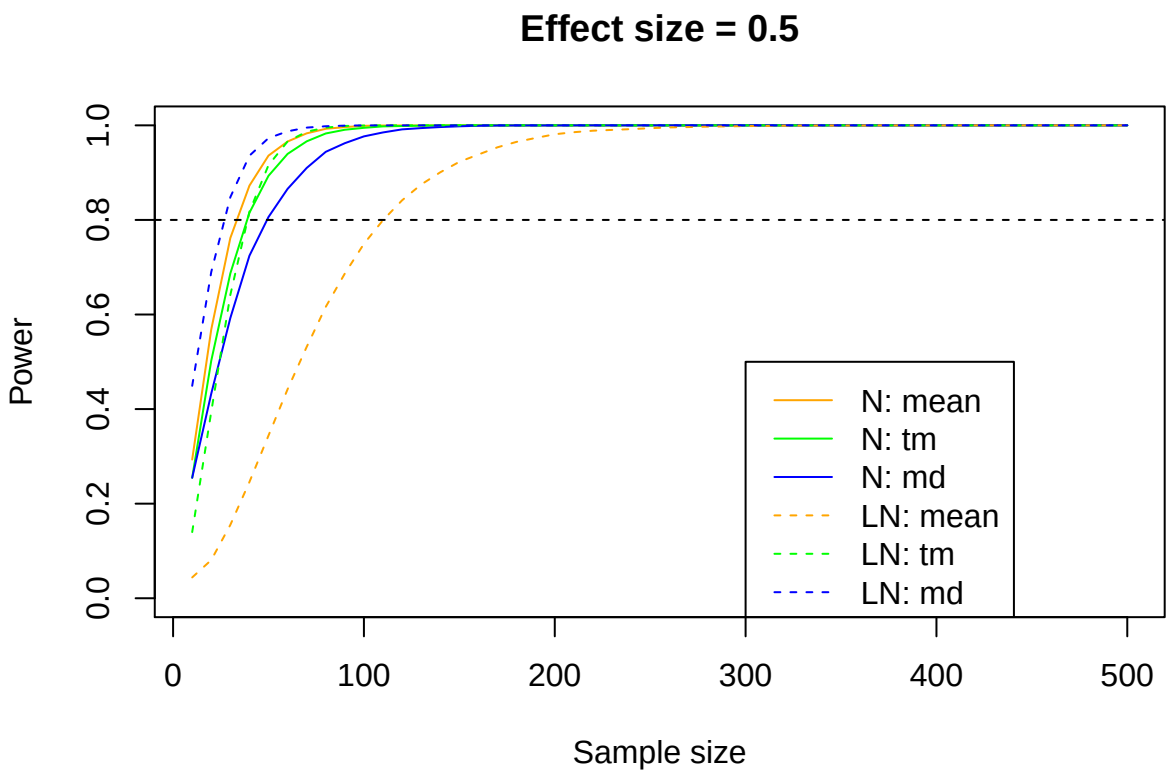


Power

```
for(E in 2:3){
plot(nvec, seq(0, 1, length.out=length(nvec)), xlab='Sample size', ylab='Power', type='n')
lines(nvec, power.res[1,E,], lty=1, col="orange") # normal: t-test on mean
lines(nvec, power.res[2,E,], lty=1, col="green") # normal: t-test on 20% trimmed mean
lines(nvec, power.res[3,E,], lty=1, col="blue") # normal: median test
lines(nvec, power.res[4,E,], lty=2, col="orange") # lognormal: t-test on mean
```

```
lines(nvec, power.res[5,E,], lty=2, col="green") # lognormal: t-test on 20% trimmed mean
lines(nvec, power.res[6,E,], lty=2, col="blue") # lognormal: median test
abline(0.8, 0, lty=2)
legend(300, 0.5,
       c("N: mean", "N: tm", "N: md", "LN: mean", "LN: tm", "LN: md"),
       lty = c(1,1,1,2,2,2),
       col = c("orange","green","blue","orange","green","blue"))
title(paste0("Effect size = ",esvec[E]))
}
```



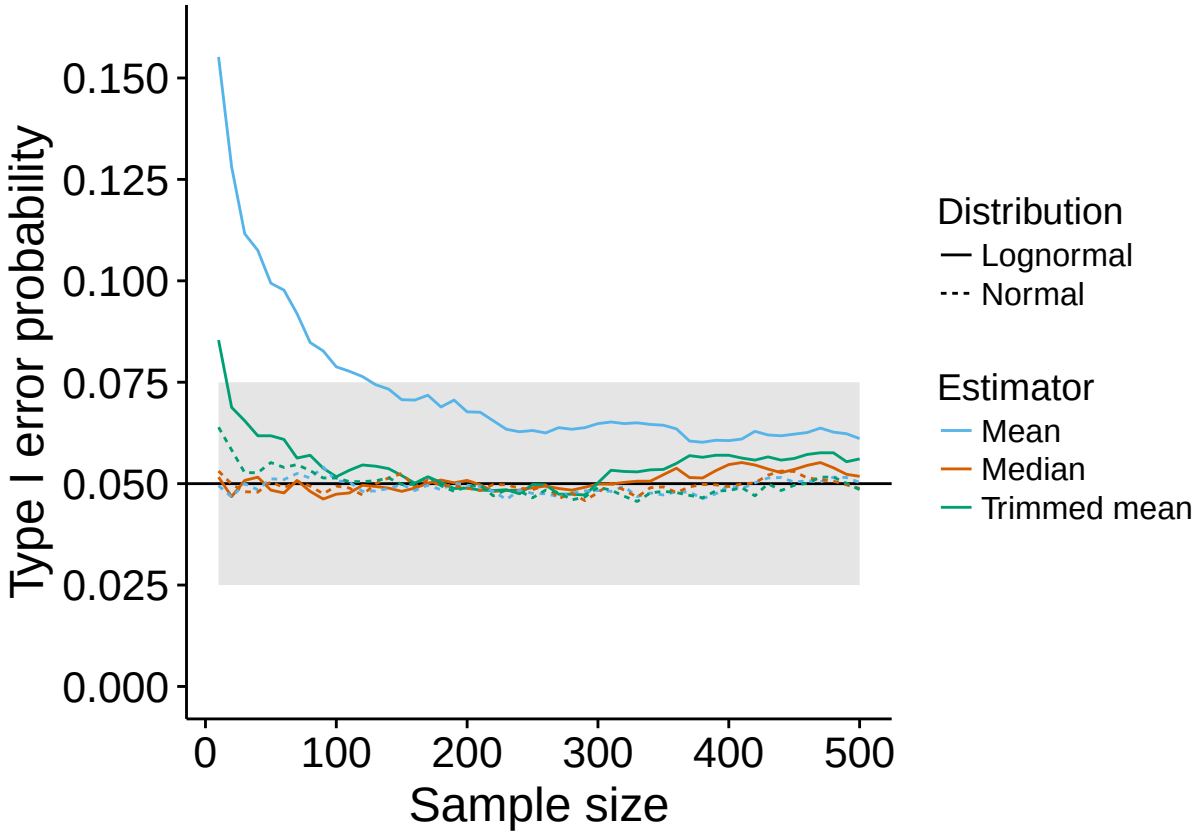


Figures using ggplot2

Type I error rate

```
# effect size
E <- 1
# create data frame
y <- c(power.res[1,E,1:50],power.res[2,E,1:50],power.res[3,E,1:50],
       power.res[4,E,1:50],power.res[5,E,1:50],power.res[6,E,1:50])
Distribution <- c(rep('Normal',length(nvec)*3), rep('Lognormal',length(nvec)*3))
Estimator <- c(rep('Mean',length(nvec)), rep('Trimmed mean',length(nvec)), rep('Median',length(nvec)),
               rep('Mean',length(nvec)), rep('Trimmed mean',length(nvec)), rep('Median',length(nvec)))
df <- tibble(x = rep(nvec, 6), # sample size
             y = y, # `Type I error probability
             as.factor(Distribution),
             as.factor(Estimator))

# make plot
p <- ggplot(df, aes(x, y)) +
  geom_ribbon(ymin = 0.025, ymax = 0.075, fill = "grey90") + # Bradley's (1978) satisfactory range
      geom_abline(intercept = 0.05, slope = 0) + # 0.05 reference line
          geom_line(aes(linetype=Distribution, colour=Estimator), size=0.5) +
  scale_colour_manual(values = c("#56B4E9", "#D55E00", "#009E73")) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18)) +
          scale_y_continuous(limits = c(0,0.16),
                             breaks = seq(0, 0.15, 0.025)) +
```

```
  labs(x = "Sample size", y = "Type I error probability")
p
```



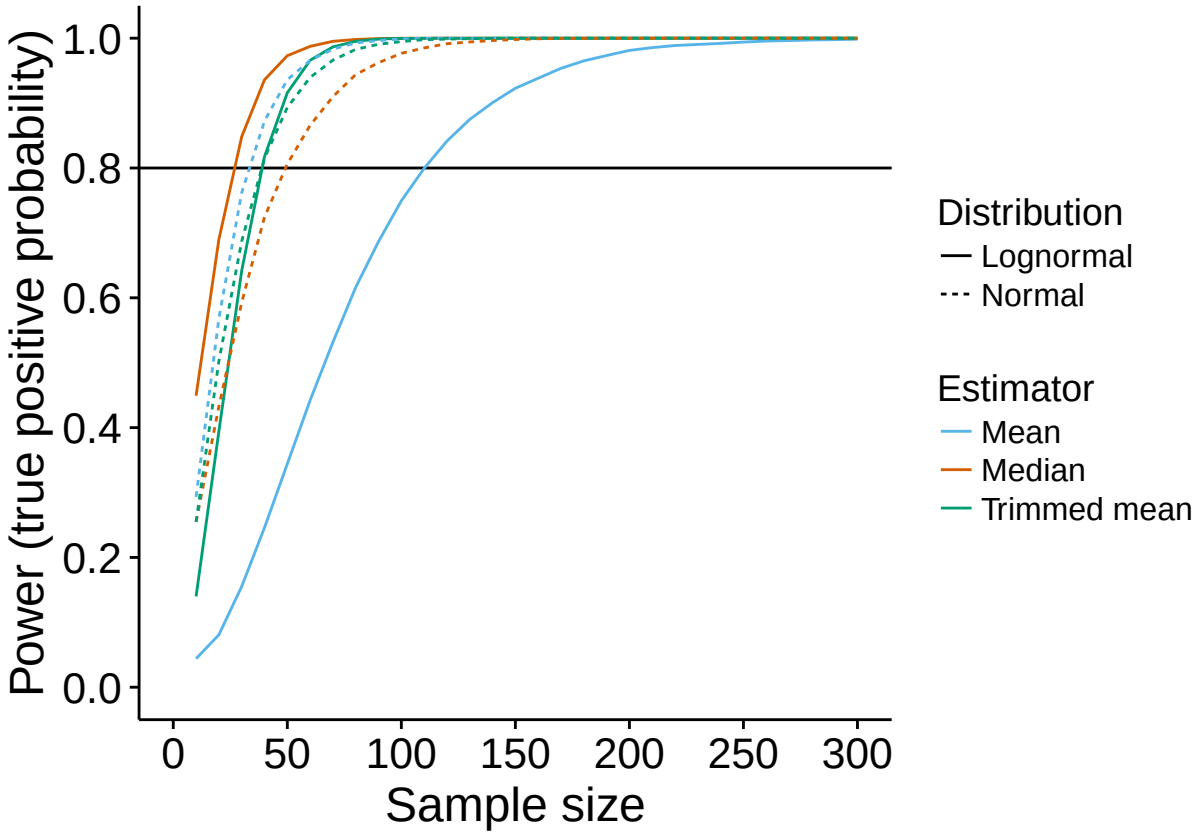

```
# save figure
# ggsave(filename='alpha.pdf',width=7,height=5) #path=pathname
ggsave(filename='alpha.jpg',width=7,height=5) #path=pathname
```

Power

```
# effect size
E <- 3 # 0.5 shift in location
# create data frame
y <- c(power.res[1,E,1:50],power.res[2,E,1:50],power.res[3,E,1:50],
       power.res[4,E,1:50],power.res[5,E,1:50],power.res[6,E,1:50])
Distribution <- c(rep('Normal',length(nvec)*3), rep('Lognormal',length(nvec)*3))
Estimator <- c(rep('Mean',length(nvec)), rep('Trimmed mean',length(nvec)), rep('Median',length(nvec)),
               rep('Mean',length(nvec)), rep('Trimmed mean',length(nvec)), rep('Median',length(nvec)))
df <- tibble(x = rep(nvec, 6), # sample size
             y = y, # `Type I error probability
             as.factor(Distribution),
             as.factor(Estimator))

# make plot
p <- ggplot(df, aes(x, y)) +
      geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(linetype=Distribution, colour=Estimator), size=0.5) +
```

```
  scale_colour_manual(values = c("#56B4E9", "#D55E00", "#009E73")) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.2)) +
  scale_x_continuous(limits = c(0, 300),
                     breaks = seq(0, 300, 50)) +
  labs(x = "Sample size", y = "Power (true positive probability)")
p
```

```
## Warning: Removed 120 rows containing missing values (geom_path).
```




```
# save figure
# ggsave(filename='power.pdf',width=7,height=5) #path=pathname
ggsave(filename='power.jpg',width=7,height=5) #path=pathname
```

```
## Warning: Removed 120 rows containing missing values (geom_path).
```