One-sample tests: false positive & power analysis [2]

Guillaume A. Rousselet & Rand R. Wilcox

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*.

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

Power analysis assuming normality

We start with a standard power analysis. We're going to look at the power of the one-sample t-test using the mean, as a function of both sample size and effect size.

```r
# Define parameters
iter = 5000 # number of simulation iterations
nvec <- seq(10,500,10) # vector of sample sizes to test
esvec <- seq(0,1,0.1) # vector of effect sizes
```

You could run a faster simulation by changing `iter` to 1000 in the previous chunk for instance. Do not run with 5000 iterations unless you're planning a long coffee break.

```r
simres <- array(0, dim = c(length(esvec), length(nvec))) # define matrix holding the results
set.seed(44) # set number generator for reproducible results
for(S in 1:iter){ # simulation loop
  for(N in 1:length(nvec)){ # sample sizes
    x <- rnorm(nvec[N]) # draw one sample from normal distribution
    for(E in 1:length(esvec)){ # effect sizes
      # one-sample t-test on sample from normal distribution with a given shift in location:
      pm = trimci(x + esvec[E], pr=F, tr=0)$p.value
      if(pm<=.05) simres[E,N] <- simres[E,N] + 1 # number of type I errors
    }
  }
}
```

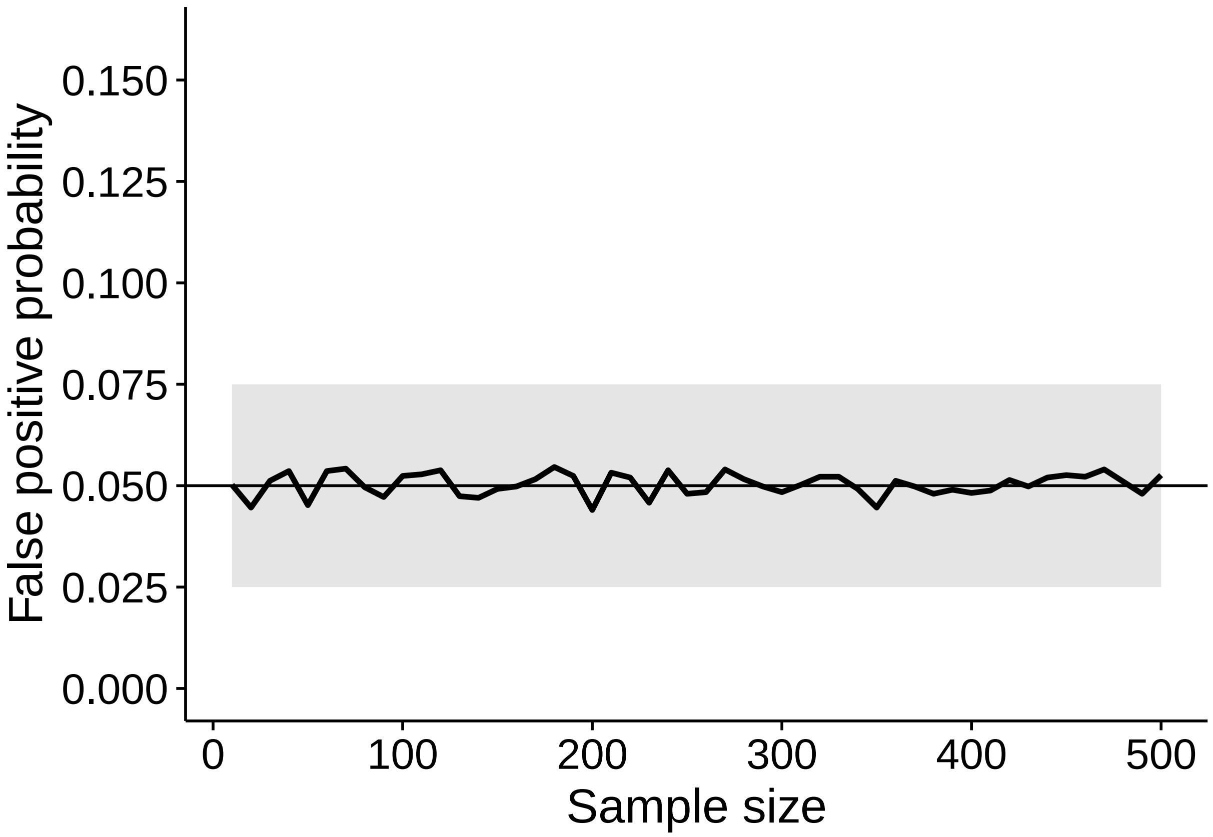
We get the type I error rate and power for each combination of effect sizes and sample sizes.

```r
# if you run the simulation in the previous chunk and want to visualise your own results,
# uncomment the next two lines, and comment out the last line.
# If you keep the same name `power_res_m.RData`, the results used in the post will be replaced by the n

# power.res <- simres / iter
# save(power.res, file = "power_res_m.RData")

load("power_res_m.RData")
```

Results

Type I error rate

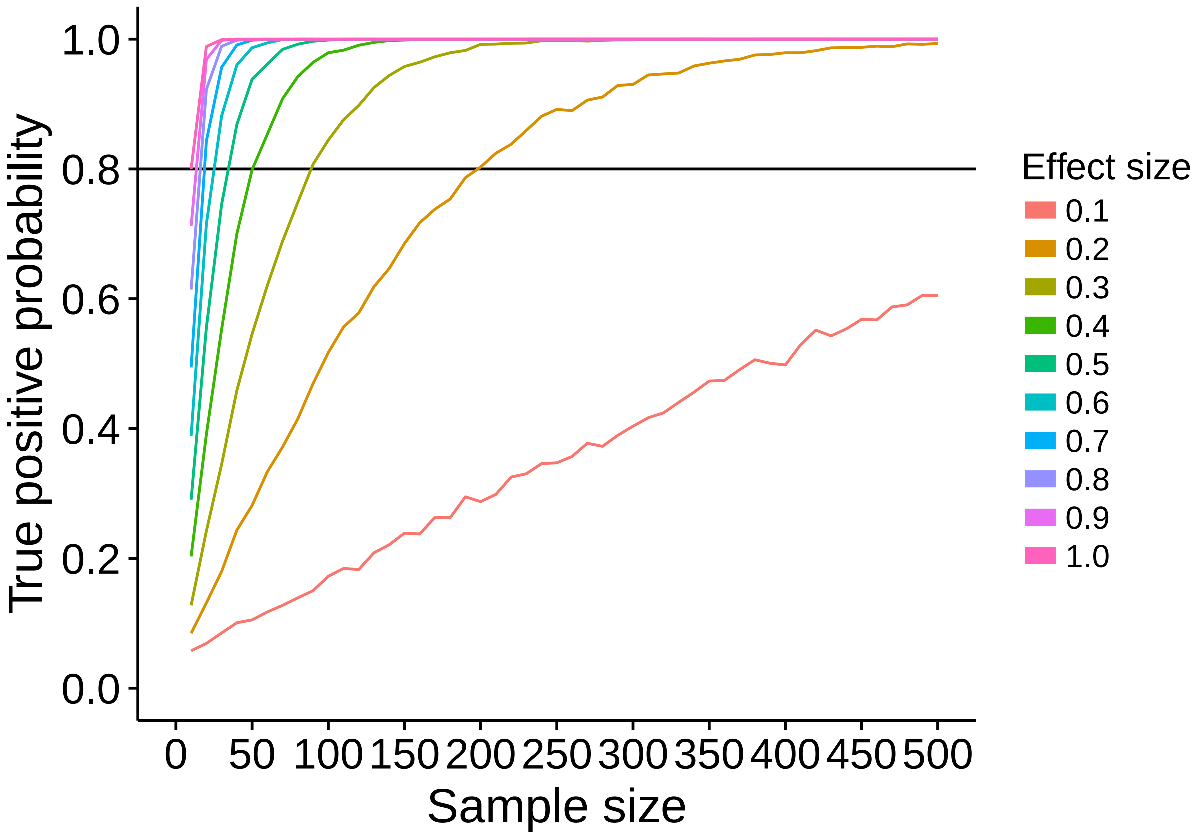
```
# create data frame
df <- tibble(x = nvec,
             y = power.res[1,])

# 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(size=1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18)) +
  # scale_x_continuous(limits = c(0, 500)) +
          scale_y_continuous(limits = c(0,0.16),
                             breaks = seq(0, 0.15, 0.025)) +
  labs(x = "Sample size", y = "False positive probability")
p
```



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

Power

```r
# create data frame
fm <- array(0, dim = c(length(nvec), length(esvec))) # make full matrix
fm[,1] <- nvec
fm[,2:length(esvec)] <- t(power.res[2:length(esvec),])
colnames(fm) <- c("SS", "0.1", "0.2", "0.3", "0.4", "0.5", "0.6", "0.7", "0.8", "0.9", "1.0")
df <- as_tibble(fm)
df <- tidyr::gather(df,`Effect size`,`TP`,2:length(esvec))
df[[2]] <- as.factor(df[[2]])

# make plot
p <- ggplot(df, aes(SS, TP, group = `Effect size`)) +
      geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour = `Effect size`), size=0.5) +
          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, 500),
                     breaks = seq(0, 500, 50)) +
  labs(x = "Sample size", y = "True positive probability") +
  guides(colour = guide_legend(override.aes = list(size=3)))
p
```



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

More realistic power simulations

The results show power of 15% for the mean and 65% for 20% trimmed mean. Here 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 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.

g & h distributions

The ghdist() function is used to generate random numbers from g & h distributions. All such distributions have a median of zero. The parameter g controls the asymmetry of the distribution, while the parameter h controls the thickness of the tails. The g & h distributions are described in this 1985 book: http://eu.wiley.com/WileyCDA/WileyTitle/productCd-047004005X.html There is also a description in Rand Wilcox's book Introduction to Robust Estimation.

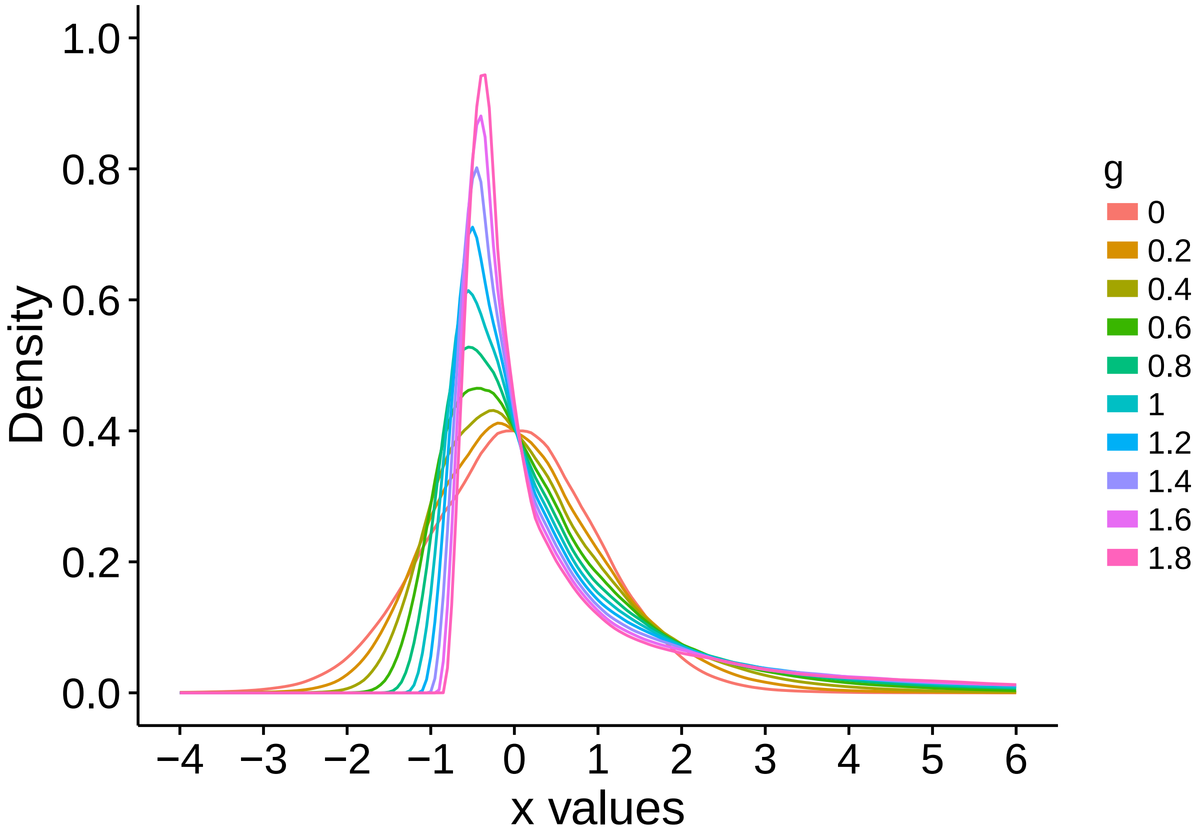
Examples in which we vary g from 0 to 1.8.

```
ng <- seq(0,1.8,0.2)
x <- seq(-4, 15, 0.05)
res.g <- array(0, dim = c(length(x), length(ng)))
for(G in 1:length(ng)){
  set.seed(7)
  res.g[,G] <- akerd(ghdist(10000, g = ng[G], h = 0), pts = x, pyhat = TRUE, plotit = FALSE)
}
```

Combine all kernel density functions into one data frame and make summary figure.

```
# make data frame
fm <- array(0, dim = c(length(x), length(ng)+1)) # make full matrix
fm[,1] <- x
fm[,2:(length(ng)+1)] <- res.g
# colnames(fm) <- c("x", "0", "0.1", "0.2", "0.3", "0.4", "0.5", "0.6", "0.7", "0.8", "0.9", "1")
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, Density,2:(length(ng)+1))
df[[2]] <- as.factor(df[[2]])

# make plot
p <- ggplot(df, aes(x, Density, group = g)) +
          geom_line(aes(colour = g), size=0.5) +
          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(-4, 6),
                     breaks = seq(-4, 15, 1)) +
  labs(x = "x values", y = "Density") +
  guides(colour = guide_legend(override.aes = list(size=3)))
p
```

```
## Warning: Removed 1800 rows containing missing values (geom_path).
```



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

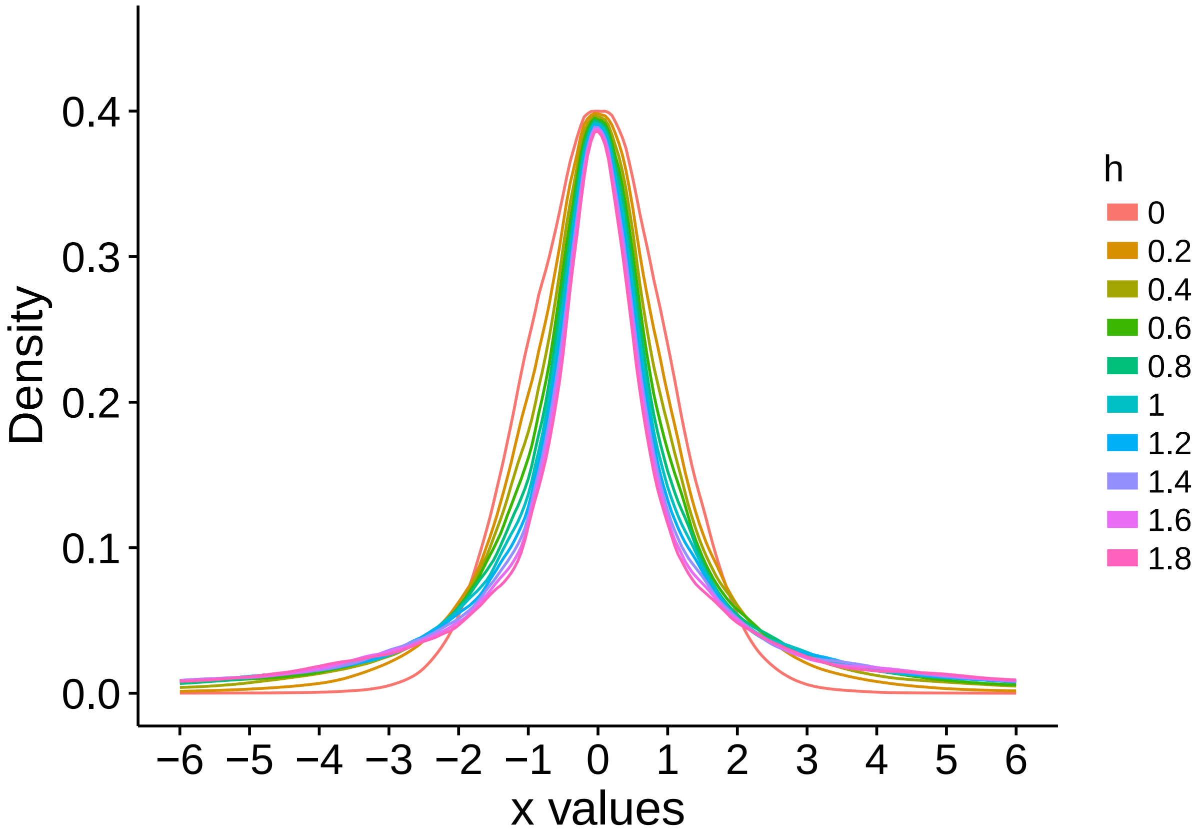
```
## Warning: Removed 1800 rows containing missing values (geom_path).
```

Examples in which we vary h from 0 to 1.8.

```r
nh <- seq(0,1.8,0.2)
x <- seq(-6, 6, 0.05)
res.h <- array(0, dim = c(length(x), length(nh)))
for(H in 1:length(nh)){
  set.seed(7)
  res.h[,H] <- akerd(ghdist(10000, g = 0, h = nh[H]), pts = x, pyhat = TRUE, plotit = FALSE)
}
```

Combine all kernel density functions into one data frame and make summary figure.

```r
# make data frame
fm <- array(0, dim = c(length(x), length(nh)+1)) # make full matrix
fm[,1] <- x
fm[,2:(length(nh)+1)] <- res.h
# colnames(fm) <- c("x", "0", "0.1", "0.2", "0.3", "0.4", "0.5", "0.6", "0.7", "0.8", "0.9", "1")
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, Density,2:(length(nh)+1))
df[[2]] <- as.factor(df[[2]])

# make plot
p <- ggplot(df, aes(x, Density, group = h)) +
          geom_line(aes(colour = h), size=0.5) +
          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, 0.45),
                             breaks = seq(0, 0.5, 0.1)) +
  scale_x_continuous(limits = c(-6, 6),
                     breaks = seq(-6, 6, 1)) +
  labs(x = "x values", y = "Density") +
  guides(colour = guide_legend(override.aes = list(size=3)))
p
```



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

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
iter <- 2000 # number of simulations
gvec <- seq(0, 1.8, 0.2) # g values
hvec <- seq(0, 1.8, 0.2) # h values
nvec <- seq(10,500,10) # vector of sample sizes to test
```

The next chunk will take a long time to run.You could run a faster simulation by changing `iter` to 500 in the previous chunk for instance.

```r
# save separately results for 3 estimators (mean, trimmed mean, median), two families of sampling param

ES <- 0.4 # effect size to estimate true positives

g.m0.res <- array(0, dim = c(length(nvec), length(gvec)))
g.md0.res <- array(0, dim = c(length(nvec), length(gvec)))
g.tm0.res <- array(0, dim = c(length(nvec), length(gvec)))
g.m1.res <- array(0, dim = c(length(nvec), length(gvec)))
g.md1.res <- array(0, dim = c(length(nvec), length(gvec)))
g.tm1.res <- array(0, dim = c(length(nvec), length(gvec)))

h.m0.res <- array(0, dim = c(length(nvec), length(hvec)))
h.md0.res <- array(0, dim = c(length(nvec), length(hvec)))
h.tm0.res <- array(0, dim = c(length(nvec), length(hvec)))
h.m1.res <- array(0, dim = c(length(nvec), length(hvec)))
h.md1.res <- array(0, dim = c(length(nvec), length(hvec)))
h.tm1.res <- array(0, dim = c(length(nvec), length(hvec)))

set.seed(45) # set random number generator for reproducibility

# save means and trimmed means of g and h distributions
mu.g <- numeric(length(gvec))
tmu.g <- numeric(length(gvec))
for(G in 2:length(gvec)){
  mu.g[G] <- ghmean(gvec[G], 0)$mean # mean of g and h distribution
  tmu.g[G] <- ghtrim(tr = .2, g = gvec[G], h = 0) # trimmed mean of g and h distribution
}

for(S in 1:iter){ # simulation iterations

  #save(".Random.seed",file="random_state_seed.RData") ## save current state
  #load("random_state_seed.RData")

  if(S %% 100 == 0){
    print(S)
  }

  for(N in 1:length(nvec)){ # sample sizes

    for(G in 1:length(gvec)){
      large.sample <- ghdist(max(nvec), g = gvec[G], h = 0)

      # type I errors ============================================
      # t-test on mean - subtract mu so the mean is zero on average
      pval <- trimci(large.sample[1:nvec[N]] - mu.g[G], tr=0, pr=FALSE)$p.value
       if(pval<=.05) g.m0.res[N,G] <- g.m0.res[N,G] + 1 # number of type I errors
      # t-test on 20% trimmed mean - subtract tmu so the trimmed mean is zero on average
      pval <- trimci(large.sample[1:nvec[N]] - tmu.g[G], tr=0.2, pr=FALSE)$p.value
      if(pval<=.05) g.tm0.res[N,G] <- g.tm0.res[N,G] + 1 # number of type I errors
      # median test
      # ghdist() generates data from distributions with median = 0, so no need to subtract the median
      pval <- sintv2(large.sample[1:nvec[N]], pr=FALSE)$p.value
      if(pval<=.05) g.md0.res[N,G] <- g.md0.res[N,G] + 1 # number of type I errors

      # true positives ============================================
      # t-test on mean - subtract mu so the mean is zero on average - add effect size
      pval <- trimci(large.sample[1:nvec[N]] - mu.g[G] + ES, tr=0, pr=FALSE)$p.value
       if(pval<=.05) g.m1.res[N,G] <- g.m1.res[N,G] + 1 # number of type I errors
      # t-test on 20% trimmed mean - subtract tmu so the trimmed mean is zero on average
      pval <- trimci(large.sample[1:nvec[N]] - tmu.g[G] + ES, tr=0.2, pr=FALSE)$p.value
      if(pval<=.05) g.tm1.res[N,G] <- g.tm1.res[N,G] + 1 # number of type I errors
      # median test
      # ghdist() generates data from distributions with median = 0, so no need to subtract the median
      pval <- sintv2(large.sample[1:nvec[N]] + ES, pr=FALSE)$p.value
      if(pval<=.05) g.md1.res[N,G] <- g.md1.res[N,G] + 1 # number of type I errors
    }

    for(H in 1:length(hvec)){
      large.sample <- ghdist(max(nvec), g = 0, h = hvec[H])

      # type I errors ============================================
      # t-test on mean
      pval <- trimci(large.sample[1:nvec[N]], tr=0, pr=FALSE)$p.value
       if(pval<=.05) h.m0.res[N,H] <- h.m0.res[N,H] + 1 # number of type I errors
      # t-test on 20% trimmed mean
      pval <- trimci(large.sample[1:nvec[N]], tr=0.2, pr=FALSE)$p.value
      if(pval<=.05) h.tm0.res[N,H] <- h.tm0.res[N,H] + 1 # number of type I errors
      # median test
      pval <- sintv2(large.sample[1:nvec[N]], pr=FALSE)$p.value
      if(pval<=.05) h.md0.res[N,H] <- h.md0.res[N,H] + 1 # number of type I errors

      # true positives ============================================
      pval <- trimci(large.sample[1:nvec[N]] + ES, tr=0, pr=FALSE)$p.value
       if(pval<=.05) h.m1.res[N,H] <- h.m1.res[N,H] + 1 # number of type I errors
      # t-test on 20% trimmed mean
      pval <- trimci(large.sample[1:nvec[N]] + ES, tr=0.2, pr=FALSE)$p.value
      if(pval<=.05) h.tm1.res[N,H] <- h.tm1.res[N,H] + 1 # number of type I errors
      # median test
      pval <- sintv2(large.sample[1:nvec[N]] + ES, pr=FALSE)$p.value
      if(pval<=.05) h.md1.res[N,H] <- h.md1.res[N,H] + 1 # number of type I errors

    }
  }
}
```

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

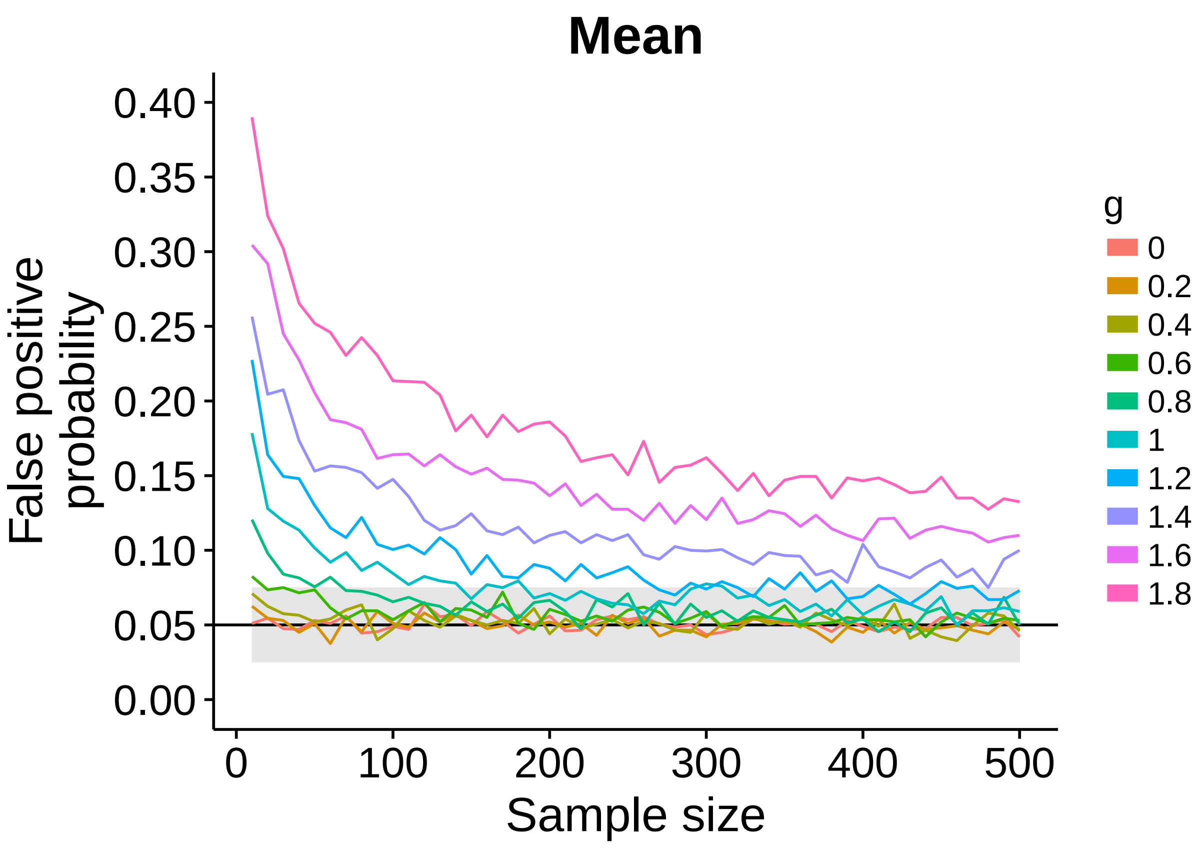
```
# if you run the simulation in the previous chunk and want to visualise your own results,
# uncomment the call to `save` and comment out the `load` line.
# If you keep the same name (power_onesample_res.RData), the results used in the post will be replaced

# save(g.m0.res, g.md0.res, g.tm0.res,
#      g.m1.res, g.md1.res, g.tm1.res,
#      h.m0.res, h.md0.res, h.tm0.res,
#      h.m1.res, h.md1.res, h.tm1.res,
#      file = "power_onesample_res.RData")

load("power_onesample_res.RData")

# number of positive results -> probabilities
g.m0.res <- g.m0.res / iter
g.md0.res <- g.md0.res / iter
g.tm0.res <- g.tm0.res / iter
g.m1.res <- g.m1.res / iter
g.md1.res <- g.md1.res / iter
g.tm1.res <- g.tm1.res / iter
h.m0.res <- h.m0.res / iter
h.md0.res <- h.md0.res / iter
h.tm0.res <- h.tm0.res / iter
h.m1.res <- h.m1.res / iter
h.md1.res <- h.md1.res / iter
h.tm1.res <- h.tm1.res / iter
```

Results

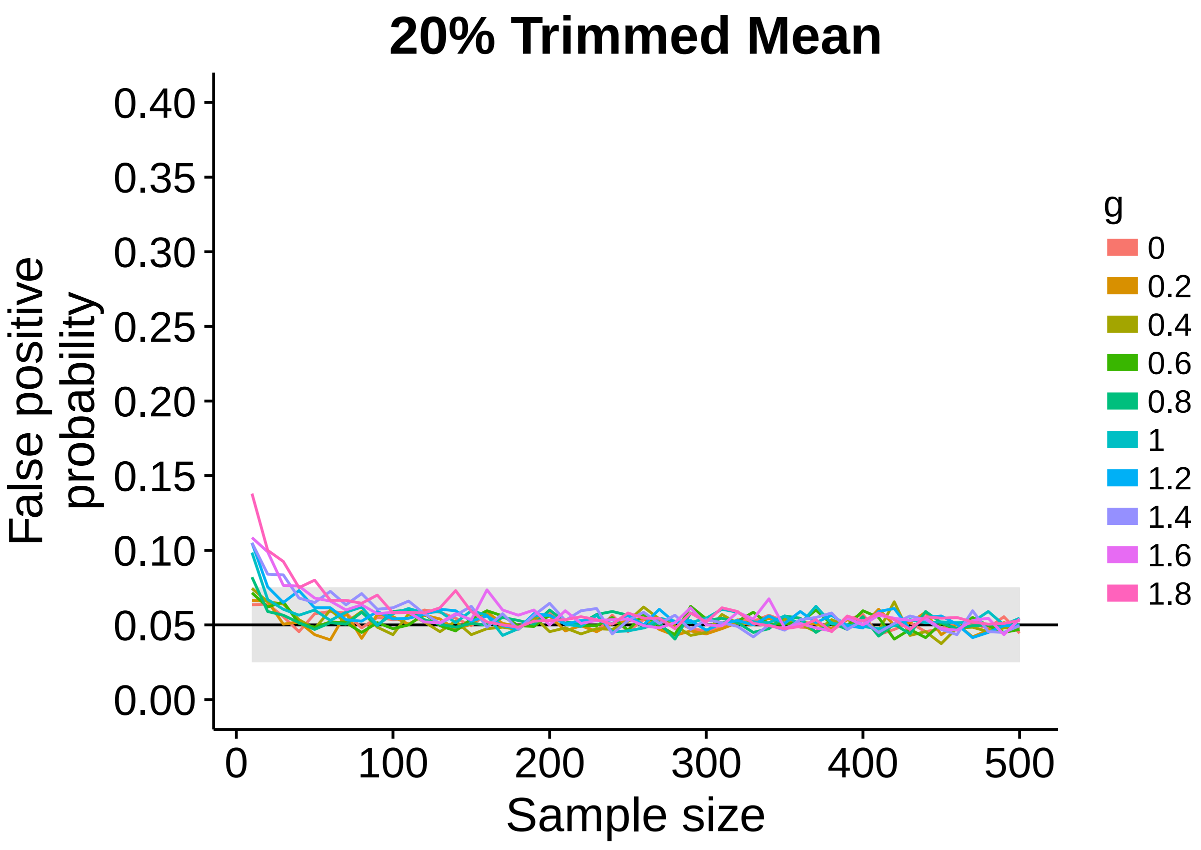
G distributions: Type I error rate

We make two figures, one for g, one for h, each showing 6 panels: a column of 3 panels for false positives, a column of 3 panels for true positives. Panels show results for mean, 20% trimmed mean and median.

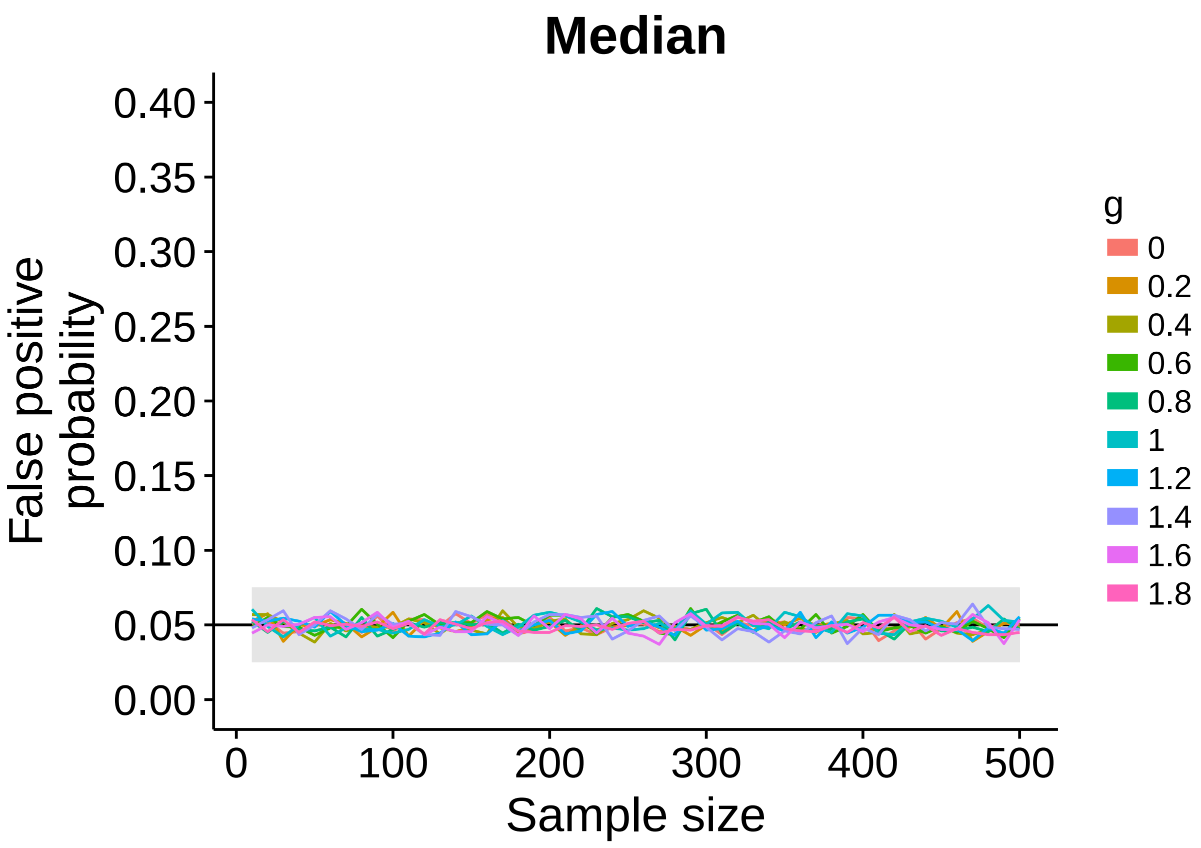
```
# g: column 1 = false positives -------------------------------------------

# MEAN
# create data frame
fm <- cbind(nvec, g.m0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.m0 <- ggplot(df, aes(x, y, group = g)) +
      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(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.m0
```



```r
# 20% TRIMMED MEAN
# create data frame
fm <- cbind(nvec, g.tm0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.tm0 <- ggplot(df, aes(x, y, group = g)) +
      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(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "20% Trimmed Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.tm0
```

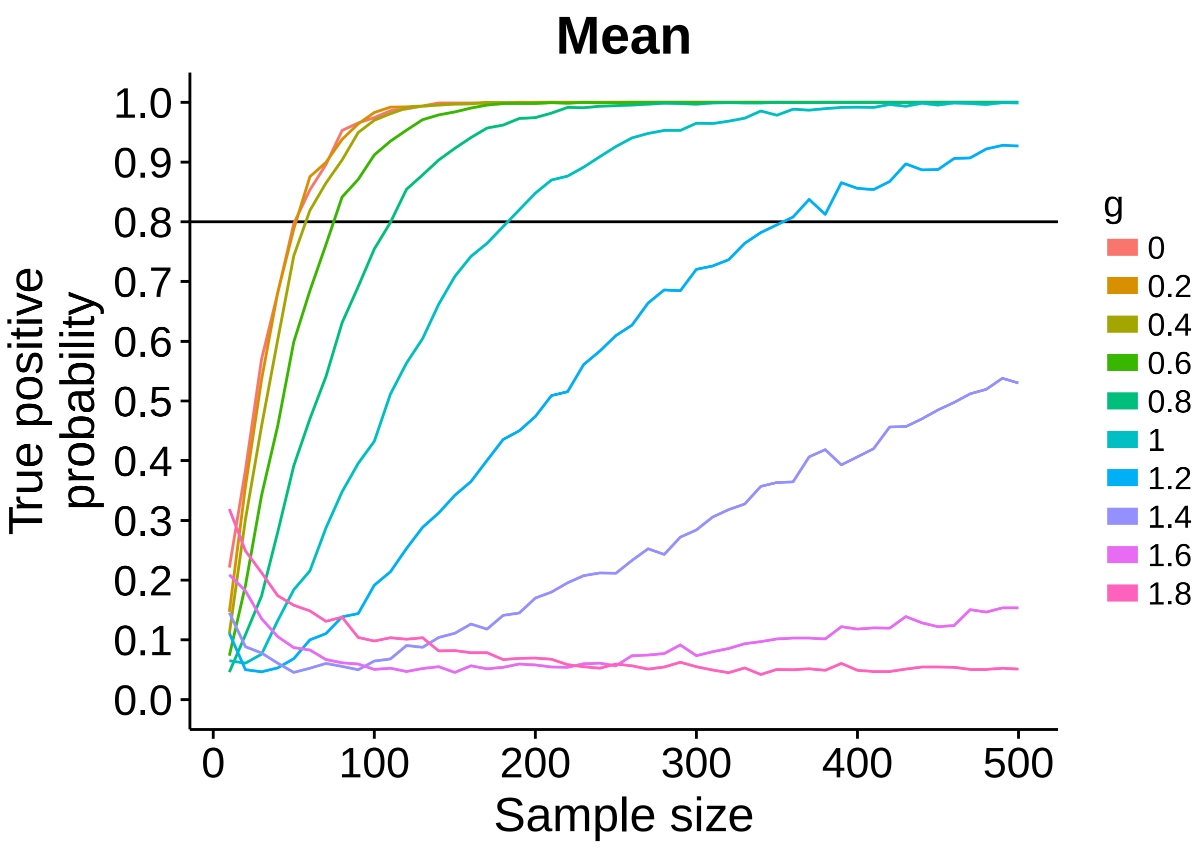


```
# MEDIAN
# create data frame
fm <- cbind(nvec, g.md0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.md0 <- ggplot(df, aes(x, y, group = g)) +
      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(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "Median") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.md0
```

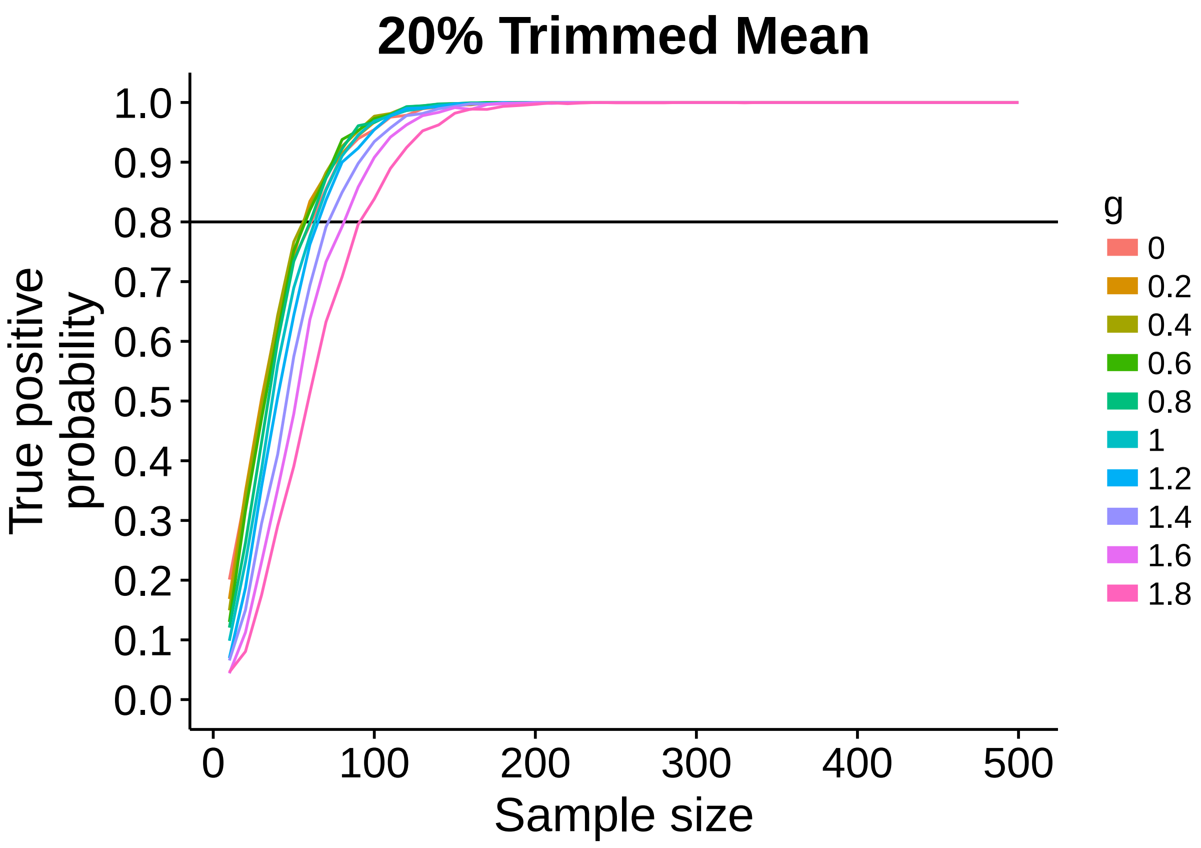


G distributions: Power

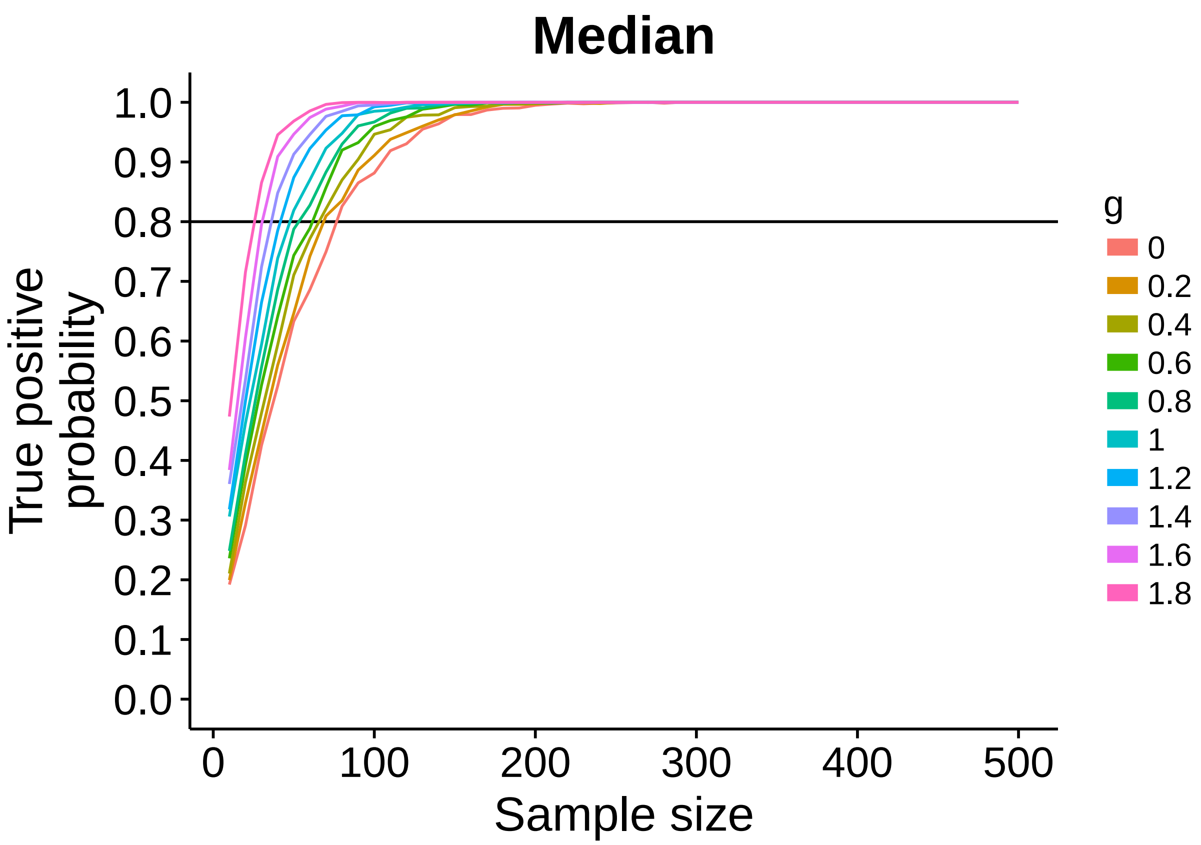
```
# g: column 2 = true positives ------------------------------------------

# MEAN
# create data frame
fm <- cbind(nvec, g.m1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.m1 <- ggplot(df, aes(x, y, group = g)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.m1
```



```
# 20% TRIMMED MEAN
# create data frame
fm <- cbind(nvec, g.tm1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.tm1 <- ggplot(df, aes(x, y, group = g)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "20% Trimmed Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.tm1
```

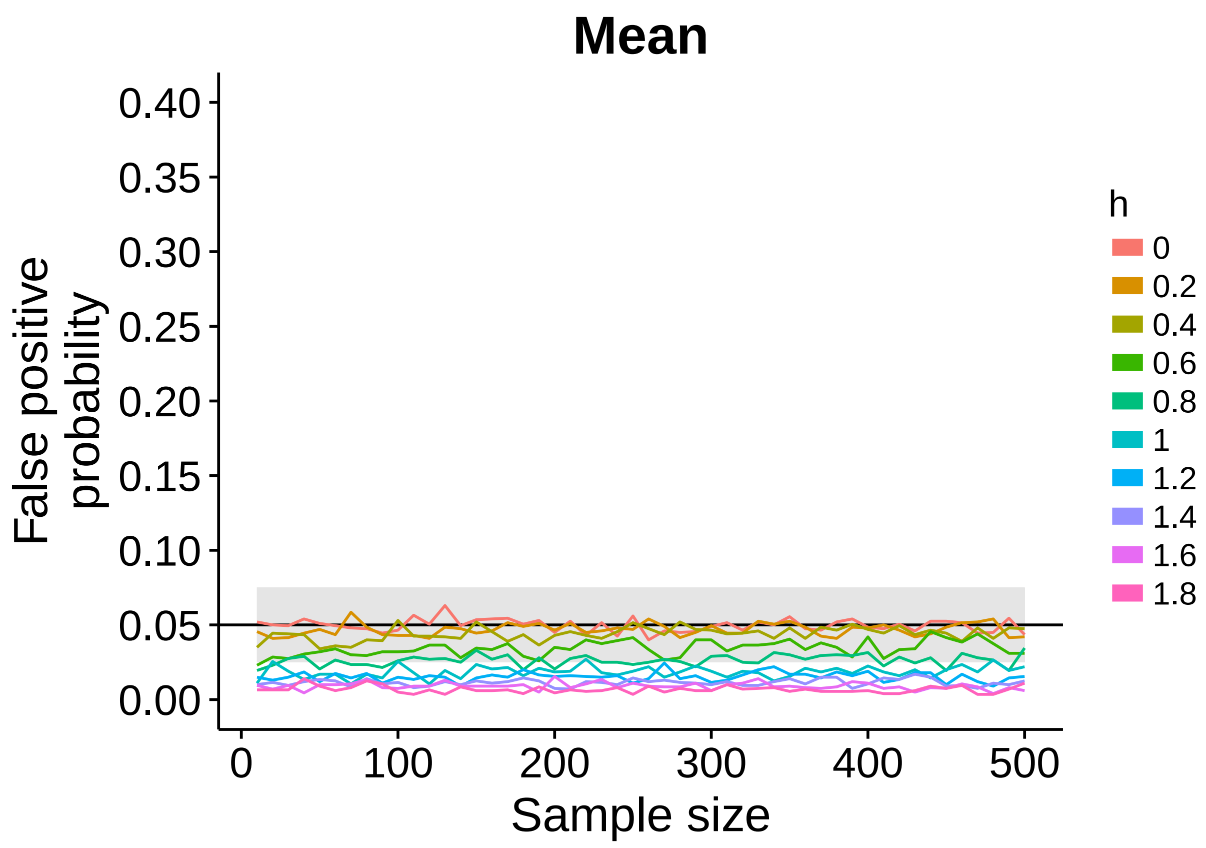


```
# MEDIAN
# create data frame
fm <- cbind(nvec, g.md1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, g, y, 2:(length(gvec)+1))
df$g <- as.factor(df$g)

# make plot
pg.md1 <- ggplot(df, aes(x, y, group = g)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=g), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "Median") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
pg.md1
```

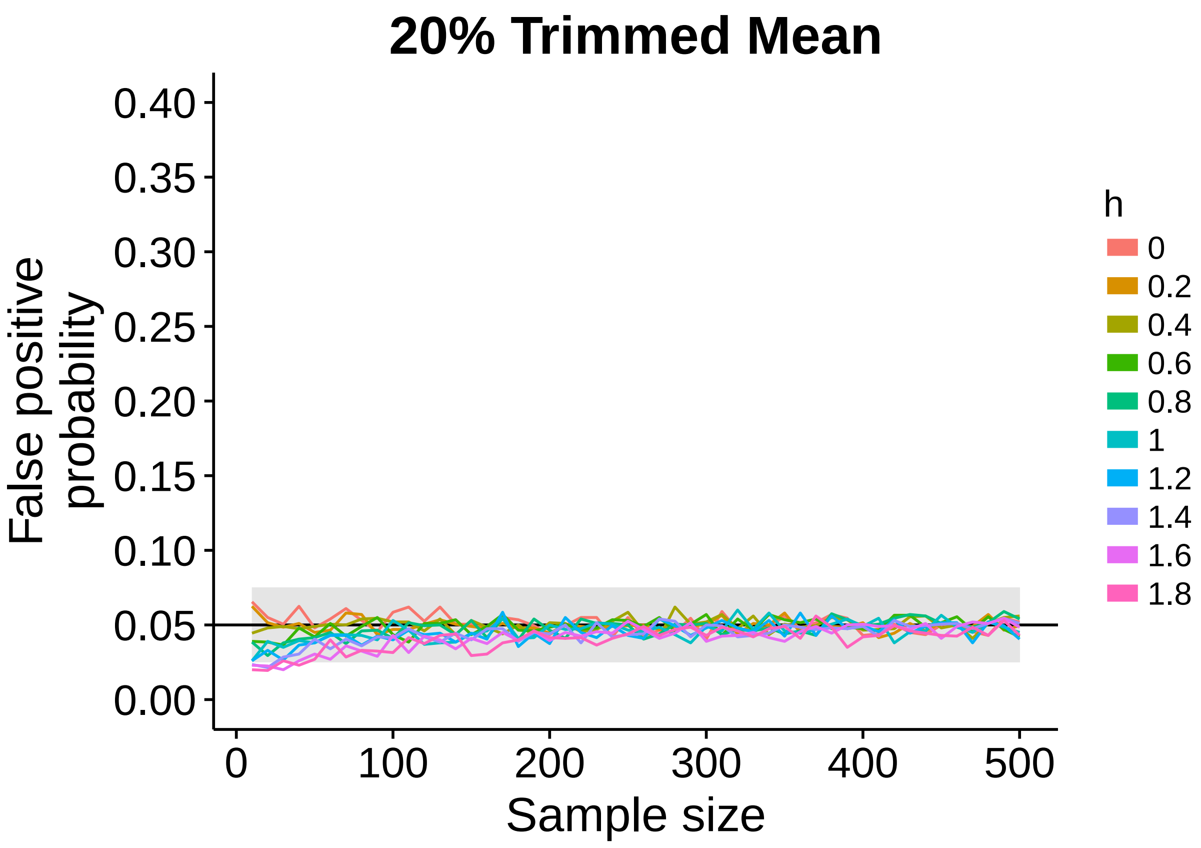


H distributions: Type I error rate

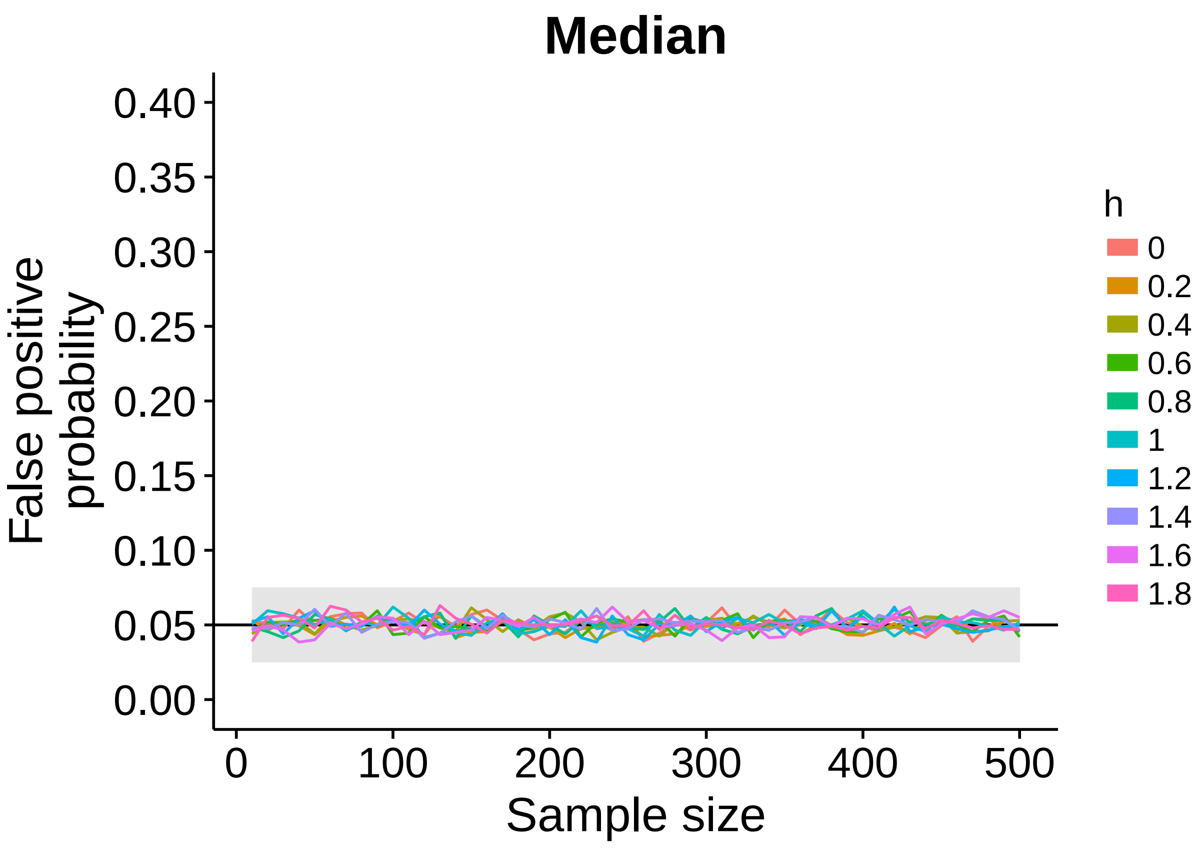
```
# h: column 1 = false positives -------------------------------------------

# MEAN
# create data frame
fm <- cbind(nvec, h.m0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(hvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.m0 <- ggplot(df, aes(x, y, group = h)) +
      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(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.m0
```



```r
# 20% TRIMMED MEAN
# create data frame
fm <- cbind(nvec, h.tm0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(hvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.tm0 <- ggplot(df, aes(x, y, group = h)) +
      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(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "20% Trimmed Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.tm0
```

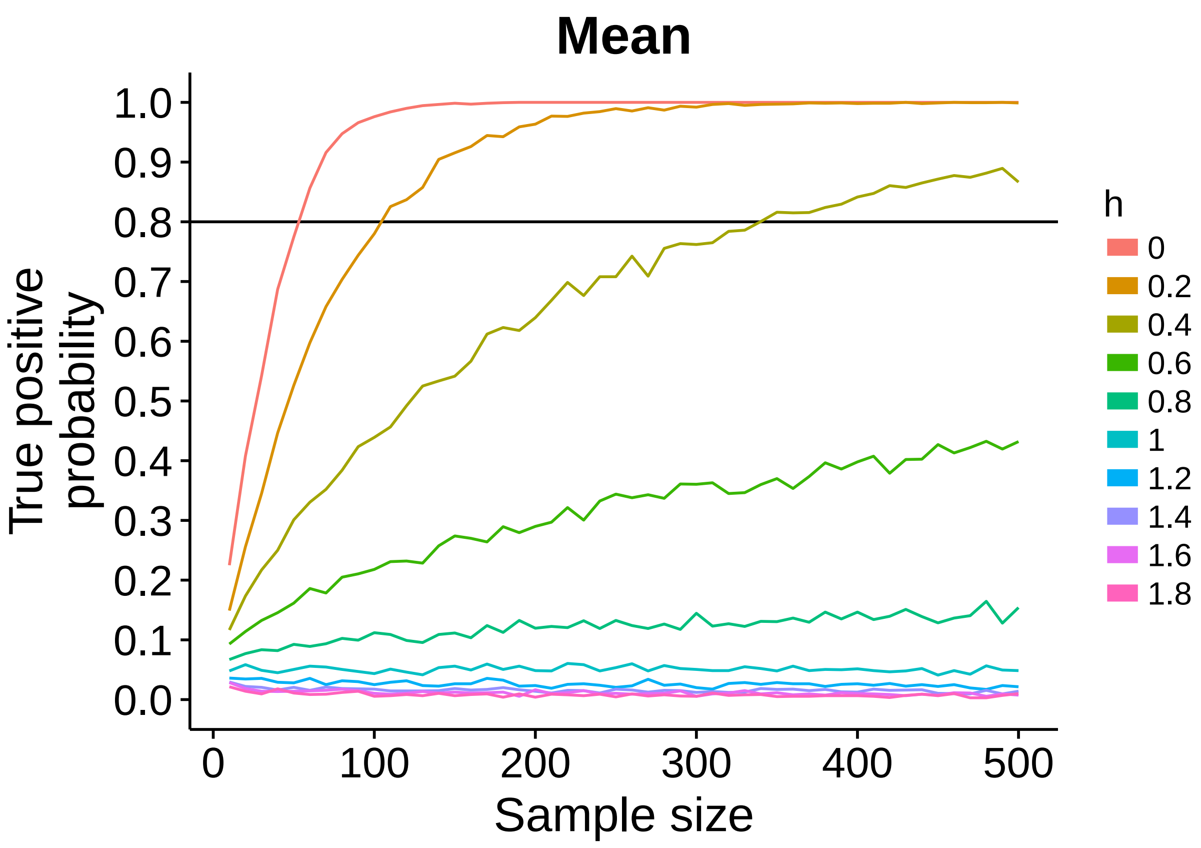


```
# MEDIAN
# create data frame
fm <- cbind(nvec, h.md0.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(gvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.md0 <- ggplot(df, aes(x, y, group = h)) +
      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(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,0.4),
                             breaks = seq(0, 0.4, 0.05)) +
  labs(x = "Sample size", y = "False positive\nprobability", title = "Median") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.md0
```

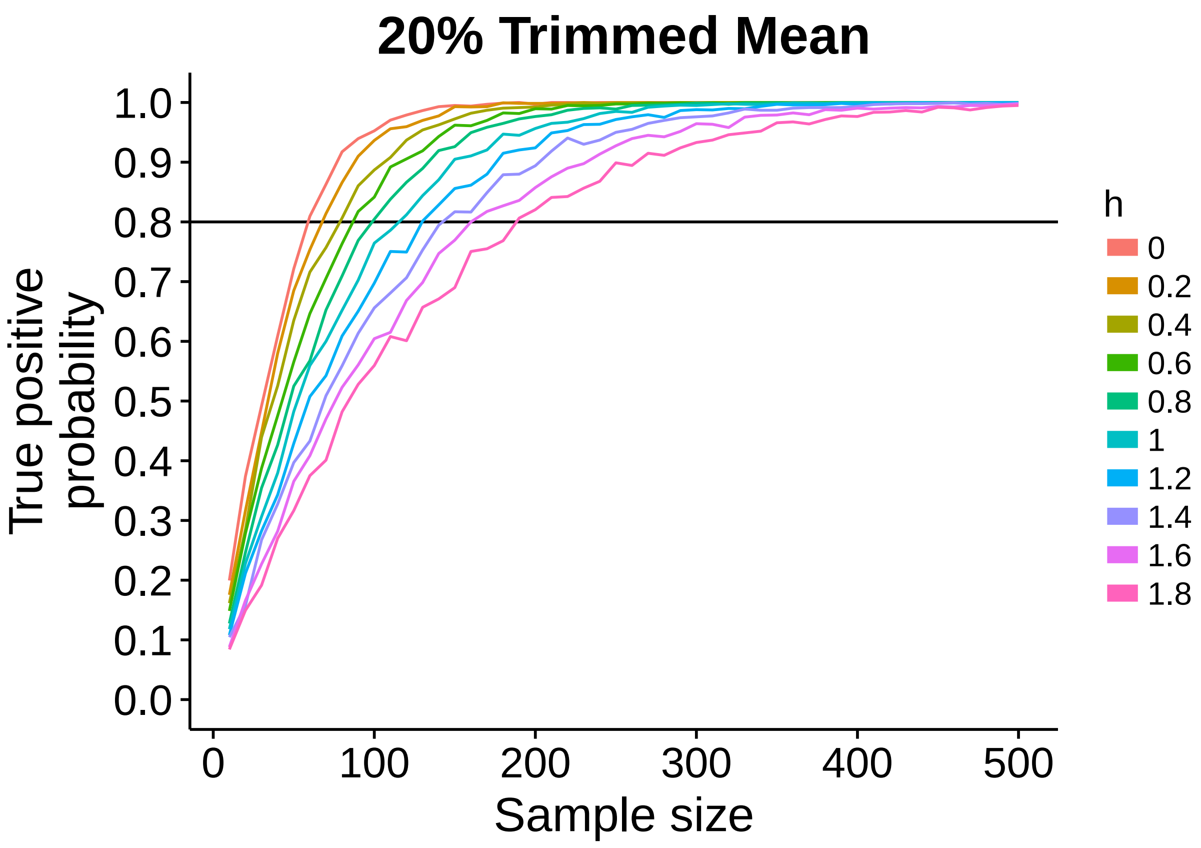


H distributions: Power

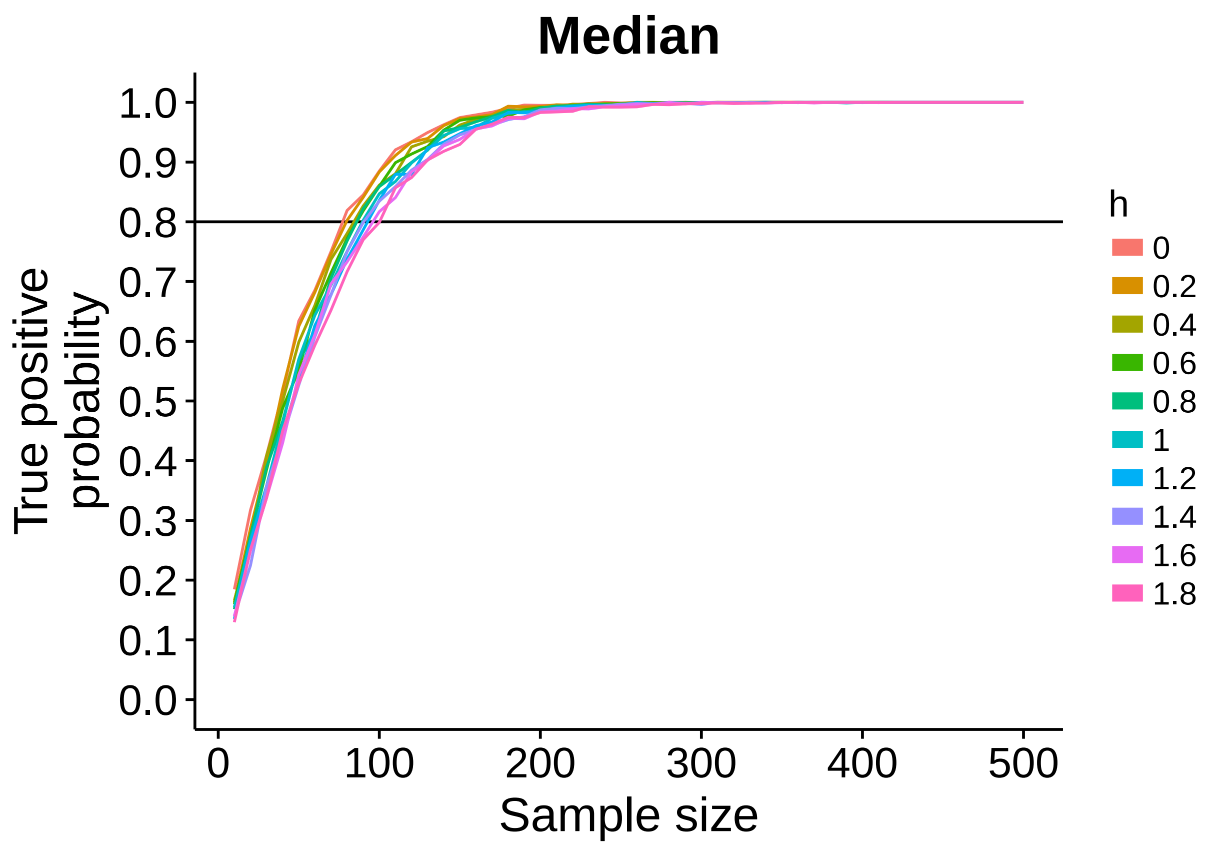
```
# h: column 2 = true positives -------------------------------------------

# MEAN
# create data frame
fm <- cbind(nvec, h.m1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(hvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.m1 <- ggplot(df, aes(x, y, group = h)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.m1
```



```
# 20% TRIMMED MEAN
# create data frame
fm <- cbind(nvec, h.tm1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(hvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.tm1 <- ggplot(df, aes(x, y, group = h)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "20% Trimmed Mean") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.tm1
```



```r
# MEDIAN
# create data frame
fm <- cbind(nvec, h.md1.res)
colnames(fm) <- c("x", "0", "0.2", "0.4", "0.6", "0.8", "1", "1.2", "1.4", "1.6", "1.8")
df <- as_tibble(fm)
df <- tidyr::gather(df, h, y, 2:(length(hvec)+1))
df$h <- as.factor(df$h)

# make plot
ph.md1 <- ggplot(df, aes(x, y, group = h)) +
          geom_abline(intercept = 0.80, slope = 0) + # 0.80 reference line
          geom_line(aes(colour=h), size = 0.5) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                plot.title = element_text(size = 20)) +
          scale_y_continuous(limits = c(0,1),
                             breaks = seq(0, 1, 0.1)) +
  labs(x = "Sample size", y = "True positive\nprobability", title = "Median") +
  guides(colour = guide_legend(override.aes = list(size = 3)))
ph.md1
```



Combine all panels together

You need to run each chunk manually if you want to reproduce the figures. `eval` was set to `FALSE` because the very large figures don't render well in the pdf and could do with a bit of tweaking in a vector image software.

Make G figure.

```r
cowplot::plot_grid(pg.m0, pg.m1, pg.tm0, pg.tm1, pg.md0, pg.md1,
                   labels=c("A", "B", "C", "D", "E", "F"),
                   ncol = 2,
                   nrow = 3,
                   rel_heights = c(1, 1, 1, 1, 1, 1),
                   label_size = 20,
                   hjust = -0.8,
                   scale=.95,
                   align = "v")
# save figure
ggsave(filename='figure_gdist_sim.jpg',width=14,height=14) #path=pathname
```

Make H figure.

```r
cowplot::plot_grid(ph.m0, ph.m1, ph.tm0, ph.tm1, ph.md0, ph.md1,
                   labels=c("A", "B", "C", "D", "E", "F"),
                   ncol = 2,
                   nrow = 3,
                   rel_heights = c(1, 1, 1, 1, 1, 1),
                   label_size = 20,
                   hjust = -0.8,
                   scale=.95,
                   align = "v")
# save figure
ggsave(filename='figure_hdist_sim.jpg',width=14,height=14) #path=pathname
```