

A guide to robust statistical methods in neuroscience: Figure 5

Rand R. Wilcox & Guillaume A. Rousselet

2023-03-02

Contents

Dependencies	1
Counter function	2
Simulation: false positives	2
Define parameters	2
Run simulation	2
Plot simulation results	3
Get results	3
Figure using ggplot2	3
Simulation: true positives	5
Define parameters	5
Run simulation	5
Plot simulation results	6
Get results	6
ggplot2 figure	6
Get values	8
Combine panels into one figure	9

We perform a simulation of false positives and true positives when sampling from normal and mixed normal distributions. The false positive results are illustrated in panel A; the true positive (power) results are illustrated in panel B.

Dependencies

```
library(ggplot2)
library(cowplot)
library(tidyr)
library(tibble)
# library(beepr) # for simulation only
source("../code/Rallfun-v40.txt")
source("../code/theme_gar.txt")
source("../code/xtrafun.R")
```

Counter function

Function used to print console updates during the simulation.

```
sim.counter <- function(S, nsim, inc){  
  if(S == 1){  
    # print(paste(nsim,"iterations:",S))  
    cat(nsim,"iterations:",S)  
    beep(2)  
  }  
  if(S %% inc == 0){  
    # print(paste("iteration",S,"/",nsim))  
    cat(" /",S)  
    beep(2)  
  }  
}
```

Simulation: false positives

Before you commit to 20,000 simulations, try with 1,000, because it might take a long time to compute. The code will output an iteration update every 500 simulations. The simulation includes:

- multiple sample sizes;
- sampling from normal and mixed normal distributions;
- t-tests using means and 20% trimmed means;
- a parametric test of medians.

Define parameters

```
nsim <- 20000 # number of simulation iterations  
nvec <- c(seq(10,150,10), seq(200,400,50), seq(500,800,100)) # vector of sample sizes to test  
max.size <- max(nvec)  
ES <- 0 # effect size
```

Run simulation

No need to run this chunk; the results are loaded in the next chunk for convenience. If you run your own simulation, change the name of the file at the end of the chunk, otherwise the results used in the article will be overwritten.

```
set.seed(45) # set random number generator for reproducibility  
  
# 6 conditions: normal / cnormal x mean / trimmed mean / median  
simres <- array(0, dim = c(6, length(nvec), nsim))  
  
simsteps <- 500 # get a console update every simsteps iterations  
  
for(S in 1:nsim){ # simulation iterations  
  
  sim.counter(S, nsim, simsteps)  
  
  large.norm.sample <- rnorm(max.size) # normal sample
```

```

large.lnorm.sample <- cnorm(max.size) # contaminated normal sample

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

  # sub-sample + shift by effect size
  norm.sample <- large.norm.sample[1:nvec[N]] + ES
  lnorm.sample <- large.lnorm.sample[1:nvec[N]] + ES

  # normal population
  # t-test on mean
  simres[1,N,S] <- trimci(norm.sample, tr=0, pr=FALSE)$p.value
  # t-test on 20% trimmed mean
  simres[2,N,S] <- trimci(norm.sample, tr=0.2, pr=FALSE)$p.value
  # median test
  simres[3,N,S] <- sintv2(norm.sample, pr=FALSE)$p.value

  # contaminated normal population
  # t-test on mean
  simres[4,N,S] <- trimci(lnorm.sample, tr=0, pr=FALSE)$p.value
  # t-test on 20% trimmed mean
  simres[5,N,S] <- trimci(lnorm.sample, tr=0.2, pr=FALSE)$p.value
  # median test
  simres[6,N,S] <- sintv2(lnorm.sample, pr=FALSE)$p.value

}
}
beep(sound = "mario")
save(simres, file = "./data/simres_cnorm_fp.RData")

```

Plot simulation results

Get results

We get the type I error rate for each combination of test and sample size.

```

load("./data/simres_cnorm_fp.RData")
res <- apply(simres <= 0.05, c(1,2), mean)

```

Figure using ggplot2

```

nvec.labs <- c("10", "", "", "", "50", "", "", "", "", "100", "", "", "", "", "150", "200", "250", "300", "350", "400", "500")

E <- 1 # effect size 1 = false positives

# create data frame
y <- c(res[1,], res[2,], res[3,],
       res[4,], res[5,], res[6,])
Distribution <- c(rep('Normal', length(nvec)*3), rep('Contaminated normal', 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)))

```

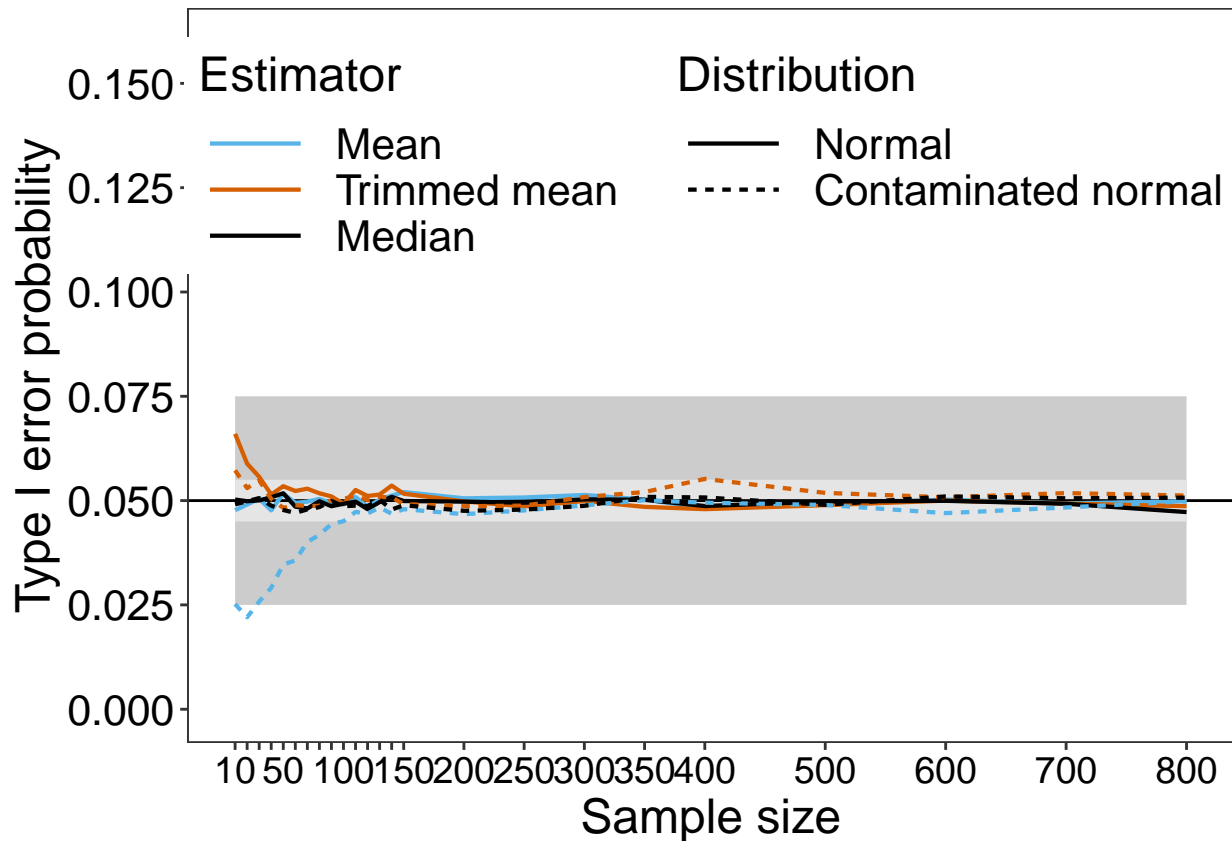
```

df <- tibble(x = rep(nvec, 6), # sample size
             y = y, # `Type I error probability`
             Distribution = as.factor(Distribution),
             Estimator = as.factor(Estimator))

df$Estimator <- keeporder(df$Estimator)
df$Distribution <- keeporder(df$Distribution)

# make plot
pA <- ggplot(df, aes(x, y)) + theme_gar +
  # Bradley's (1978) satisfactory range
  geom_ribbon(aes(ymin = 0.025, ymax = 0.075), fill = "grey80") +
  # Bradley's (1978) ideal range
  geom_ribbon(aes(ymin = 0.045, ymax = 0.055), fill = "grey90") +
  geom_abline(intercept = 0.05, slope = 0) + # 0.05 reference line
  geom_line(aes(linetype=Distribution, colour=Estimator), linewidth=0.75) +
  scale_colour_manual(values = c("#56B4E9", "#D55E00", "black")) + #009E73
  theme(axis.title.x = element_text(size = 18),
        axis.text.x = element_text(size = 14),
        axis.text.y = element_text(size = 16),
        axis.title.y = element_text(size = 18),
        panel.grid.minor = element_blank(),
        panel.grid.major = element_blank(),
        legend.position = c(0.5, 0.8),
        legend.box = "horizontal") +
  scale_x_continuous(breaks = nvec, labels = nvec.labs) +
  scale_y_continuous(limits = c(0, 0.16),
                    breaks = seq(0, 0.15, 0.025)) +
  labs(x = "Sample size", y = "Type I error probability")
pA

```



Simulation: true positives

Define parameters

```
nsim <- 20000 # number of simulation iterations
nvec <- c(seq(10,100,10), seq(125,250,25), 300) # vector of sample sizes to test
max.size <- max(nvec)
ES <- 0.5 # effect size
```

Run simulation

No need to run this chunk; the results are loaded in the next chunk for convenience. If you run your own simulation, change the name of the file at the end of the chunk, otherwise the results used in the article will be overwritten.

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

# 6 conditions: normal / skewed x mean / trimmed mean / median
simres <- array(0, dim = c(6, length(nvec), nsim))

simsteps <- 500 # get a console update every simsteps iterations

for(S in 1:nsim){ # simulation iterations
```

```

sim.counter(S, nsim, simsteps)

large.norm.sample <- rnorm(max.size) # normal sample
large.lnorm.sample <- cnorm(max.size) # lognormal sample

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

  # sub-sample + shift by effect size
  norm.sample <- large.norm.sample[1:nvec[N]] + ES
  lnorm.sample <- large.lnorm.sample[1:nvec[N]] + ES

  # normal population
  # t-test on mean
  simres[1,N,S] <- trimci(norm.sample, tr=0, pr=FALSE)$p.value
  # t-test on 20% trimmed mean
  simres[2,N,S] <- trimci(norm.sample, tr=0.2, pr=FALSE)$p.value
  # median test
  simres[3,N,S] <- sintv2(norm.sample, pr=FALSE)$p.value

  # contaminated normal population
  # t-test on mean
  simres[4,N,S] <- trimci(lnorm.sample, tr=0, pr=FALSE)$p.value
  # t-test on 20% trimmed mean
  simres[5,N,S] <- trimci(lnorm.sample, tr=0.2, pr=FALSE)$p.value
  # median test
  simres[6,N,S] <- sintv2(lnorm.sample, pr=FALSE)$p.value

}
}
beep(sound = "mario")
save(simres, file = "./data/simres_cnorm_tp.RData")

```

Plot simulation results

Get results

We get the power for each combination of test and sample size.

```

load("./data/simres_cnorm_tp.RData")
res <- apply(simres <= 0.05, c(1,2), mean)

```

ggplot2 figure

```

nvec.labs <- c("10", "", "", "", "50", "", "", "", "", "100", "", "150", "", "200", "", "250", "300")

# create data frame
y <- c(res[1,], res[2,], res[3,],
       res[4,], res[5,], res[6,])
Distribution <- c(rep('Normal', length(nvec)*3), rep('Contaminated normal', length(nvec)*3))
Estimator <- c(rep('Mean', length(nvec)), rep('Trimmed Mean', length(nvec)), rep('Median', length(nvec)),

```

```

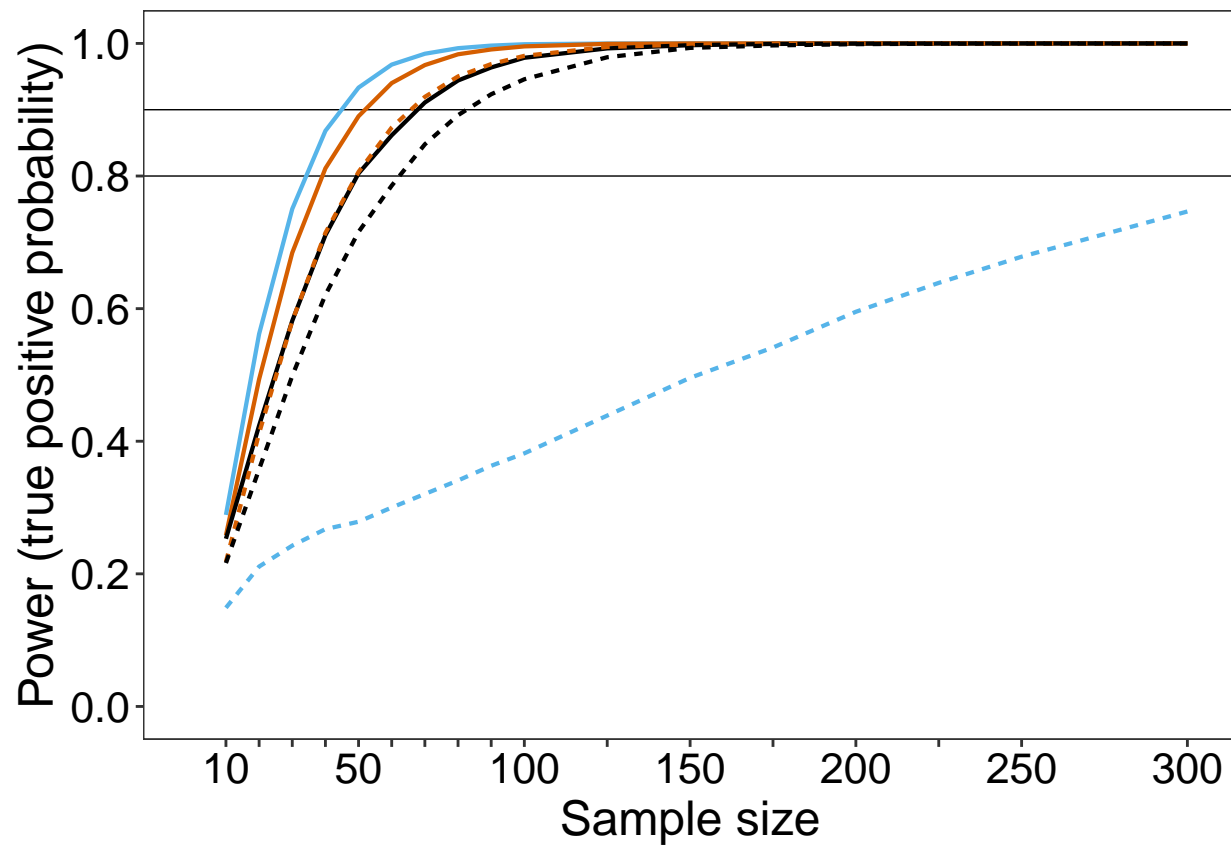
      rep('Mean',length(nvec)), rep('Trimmed Mean',length(nvec)), rep('Median',length(nvec)))

df <- tibble(x = rep(nvec, 6), # sample size
             y = y, # `Type I error probability`
             Distribution = as.factor(Distribution),
             Estimator = as.factor(Estimator))

df$Estimator <- keeporder(df$Estimator)
df$Distribution <- keeporder(df$Distribution)

# make plot
pB <- ggplot(df, aes(x, y)) + theme_gar +
  geom_abline(intercept = 0.80, slope = 0, linewidth=0.25) + # 0.80 reference line
  geom_abline(intercept = 0.90, slope = 0, linewidth=0.25) + # 0.90 reference line
  geom_line(aes(linetype=Distribution, colour=Estimator), linewidth=0.75, show.legend = FALSE) +
  scale_colour_manual(values = c("#56B4E9", "#D55E00", "black")) + #009E73
  theme(axis.title.x = element_text(size = 18),
        axis.text = element_text(size = 16),
        axis.title.y = element_text(size = 18),
        panel.grid.minor = element_blank(),
        panel.grid.major = element_blank(),
        legend.position = c(0.65, 0.2),
        legend.box = "horizontal") +
  scale_y_continuous(limits = c(0,1),
                    breaks = seq(0, 1, 0.2),
                    minor_breaks = seq(0.1,0.9,0.2)) +
  scale_x_continuous(limits = c(0, 300),
                    breaks = nvec, labels = nvec.labs) +
  labs(x = "Sample size", y = "Power (true positive probability)")
pB

```



Get values

Power when sampling from a normal population:

n=20

- mean: 0.562;
- 20% trimmed mean: 0.494;
- median: 0.424.

n=40

- mean: 0.868;
- 20% trimmed mean: 0.812;
- median: 0.711.

Power when sampling from a contaminated normal population:

n=20

- mean: 0.211;

- 20% trimmed mean: 0.412;
- median: 0.358.

n=40

- mean: 0.267;
- 20% trimmed mean: 0.714;
- median: 0.621.

Combine panels into one figure

```
cowplot::plot_grid(pA, pB,  
  labels=c("A", "B"),  
  ncol = 1,  
  nrow = 2,  
  rel_widths = c(1, 1),  
  label_size = 20,  
  hjust = -0.5,  
  scale=.95,  
  align = "h")  
  
# save figure  
ggsave(filename='./figures/figure5.pdf',width=7,height=10)
```