Group level true positives using exGaussian distributions with FLP parameters

Guillaume A. Rousselet

2019-01-16

Contents


```
# dependencies
library(ggplot2)
library(tibble)
library(tidyr)
library(cowplot)
library(retimes)
source("./functions/tests.txt")
library(beepr)
```

```
sessionInfo()
```

```
## R version 3.4.0 (2017-04-21)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS  10.14.2
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
```

```
## [1] en_GB.UTF-8/en_GB.UTF-8/en_GB.UTF-8/C/en_GB.UTF-8/en_GB.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] beepr_1.3     retimes_0.1-2 cowplot_0.9.1 tidyr_0.7.2   tibble_1.4.2
## [6] ggplot2_3.0.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.19     pillar_1.3.0     compiler_3.4.0   plyr_1.8.4
##  [5] bindr_0.1.1      tools_3.4.0      digest_0.6.12    evaluate_0.10.1
##  [9] gtable_0.2.0     pkgconfig_2.0.2  rlang_0.2.2      rstudioapi_0.8
## [13] yaml_2.1.15      bindrcpp_0.2.2   withr_2.1.0      dplyr_0.7.6
## [17] stringr_1.2.0    knitr_1.17       rprojroot_1.2    grid_3.4.0
## [21] tidyselect_0.2.4 glue_1.3.0       R6_2.3.0         rmarkdown_1.8
## [25] purrr_0.2.5      magrittr_1.5     backports_1.1.1  scales_0.5.0
## [29] htmltools_0.3.6  assertthat_0.2.0 colorspace_1.3-2 stringi_1.1.6
## [33] lazyeval_0.2.1   munsell_0.4.3    crayon_1.3.4     audio_0.1-5.1
```

Original effect sizes

Get data

```
# get data - tibble = `flp`
load("./data/french_lexicon_project_rt_data.RData")
# columns =
#1 = participant
#2 = rt
#3 = acc = accuracy 0/1
#4 = condition = word/non-word
```

Mean and median RT for every participant

```
# get data:
medres <- tapply(flp$rt, list(flp$participant, flp$condition), median)
meanres <- tapply(flp$rt, list(flp$participant, flp$condition), mean)
```

Mean difference of means

```
out <- t.test(meanres[,2]-meanres[,1])
```

Non-Word - Word mean difference = 84.9882641 [80.7339035 ,89.2426248].

Median difference of medians

```
diff <- medres[,2]-medres[,1]
out <- sint(diff)
```

Non-Word - Word median difference = 78.5 [74 ,82.5].

Get ex-Gaussian parameters from FLP data

This is done in `flp_exg_parameters`.

```
load('./data/flp_exg_param.RData')
Np.total <- nrow(exg_param_w)
```

Interpolate parameters between Word and Non-Word conditions

Also simulate data, then check group effect sizes against original ones.

```
set.seed(1)

nt <- 100
sim.data.m <- matrix(NA, nrow = Np.total, ncol = 2) # W/NW
sim.data.md <- matrix(NA, nrow = Np.total, ncol = 2) # W/NW
exg_param_interp <- array(NA, dim = c(Np.total, 3, 10))

for(P in 1:Np.total){
  for(param in 1:3){
    # interpolate parameters between conditions: 10 steps
    exg_param_interp[P, param, ] <- seq(exg_param_w[P,param], exg_param_nw[P,param], length.out = 10)
  }
  # simulate data for the two conditions using nt trials
  w <- rexgauss(nt,
                mu = exg_param_w[P,1],
                sigma = exg_param_w[P,2],
                tau = exg_param_w[P,3])
  nw <- rexgauss(nt,
                 mu = exg_param_nw[P,1],
                 sigma = exg_param_nw[P,2],
                 tau = exg_param_nw[P,3])
  sim.data.m[P,1] <- mean(w)
  sim.data.m[P,2] <- mean(nw)
  sim.data.md[P,1] <- median(w)
  sim.data.md[P,2] <- median(nw)
}
save(
  sim.data.m,
  sim.data.md,
  exg_param_interp,
  file='./data/flp_exg_param_interp.RData'
)
```

Check difference results

```
load('./data/flp_exg_param_interp.RData')
diff <- sim.data.m[,2] - sim.data.m[,1]
t.test(diff)
```

```
##
##  One Sample t-test
##
## data:  diff
## t = 34.353, df = 958, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  80.19289 89.91017
## sample estimates:
## mean of x
##  85.05153
```

```
diff <- sim.data.md[,2] - sim.data.md[,1]
median(diff)
```

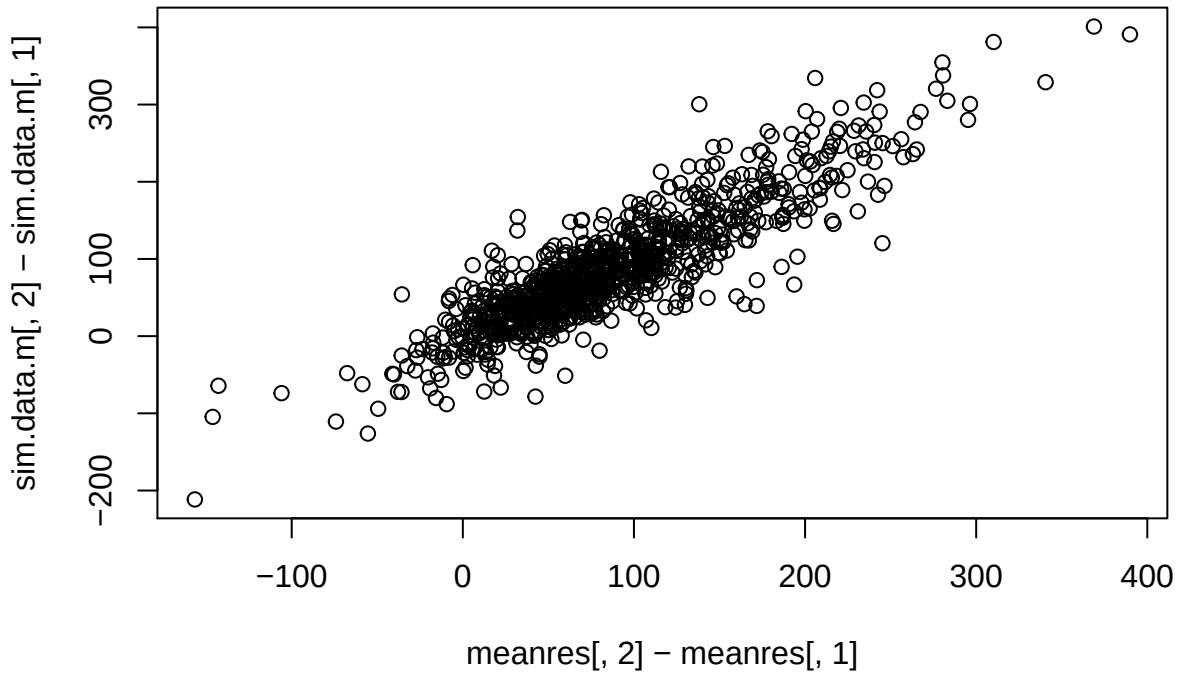
```
## [1] 77.12523
```

```
sint(diff)
```

```
## [1] 71.64771 81.12961
```

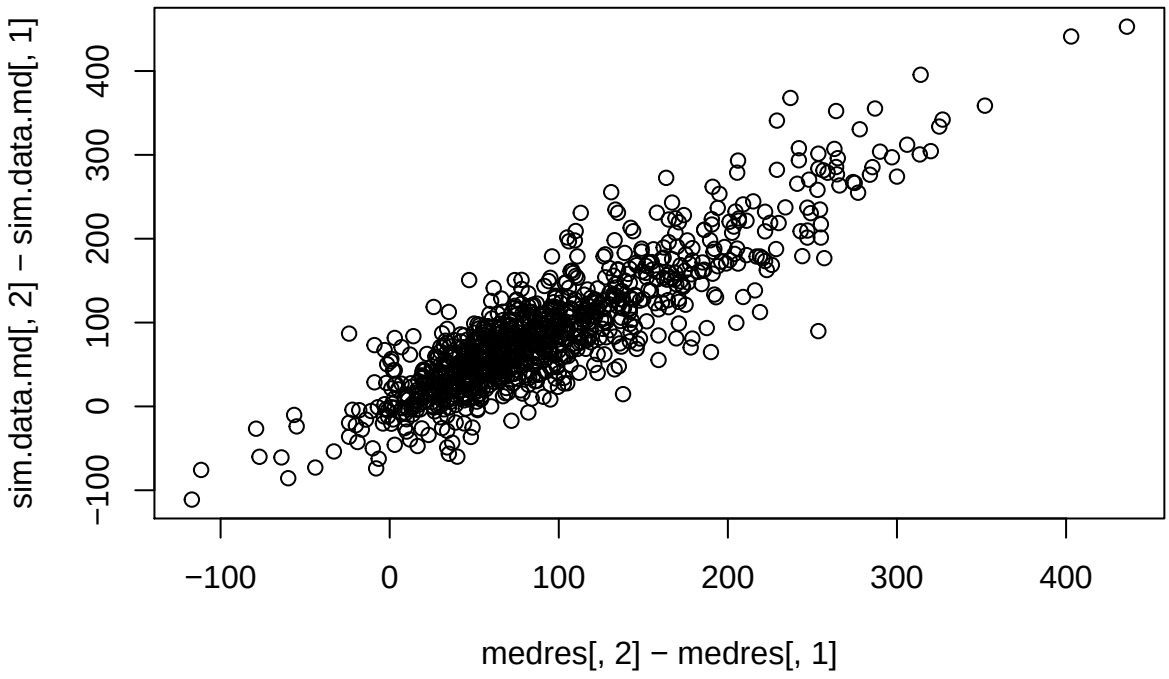
Scatterplot: means

```
plot(meanres[,2] - meanres[,1], sim.data.m[,2] - sim.data.m[,1])
```



Scatterplot: medians

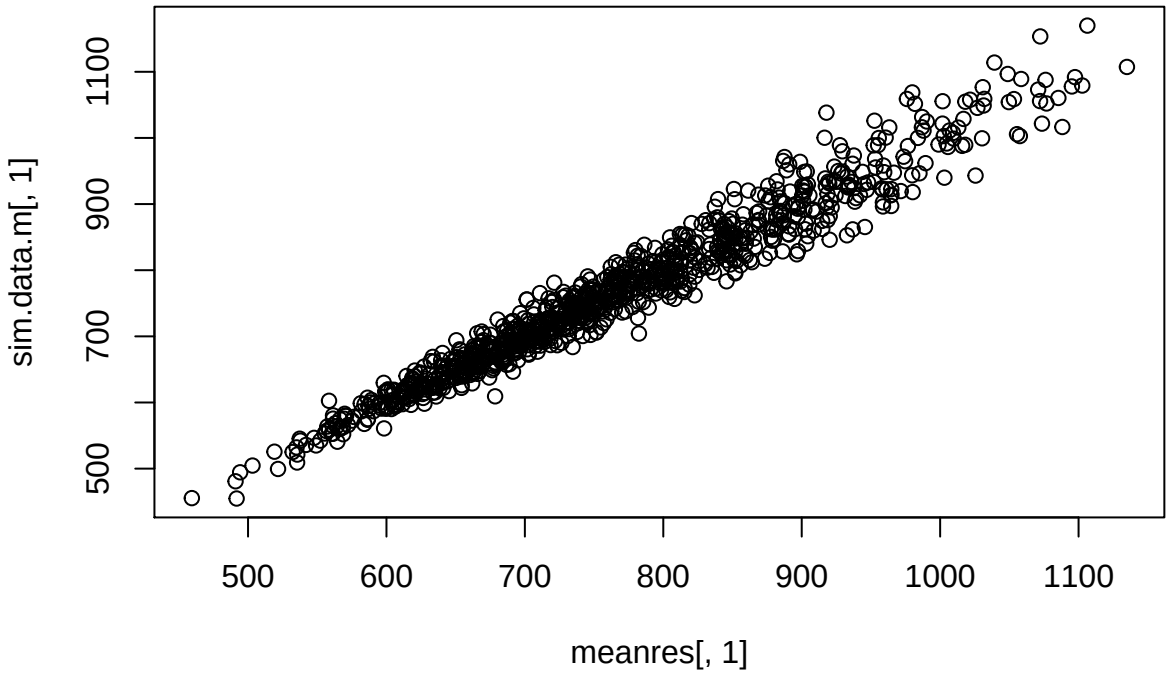
```
plot(medres[,2] - medres[,1], sim.data.md[,2] - sim.data.md[,1])
```



Compare original to simulated data

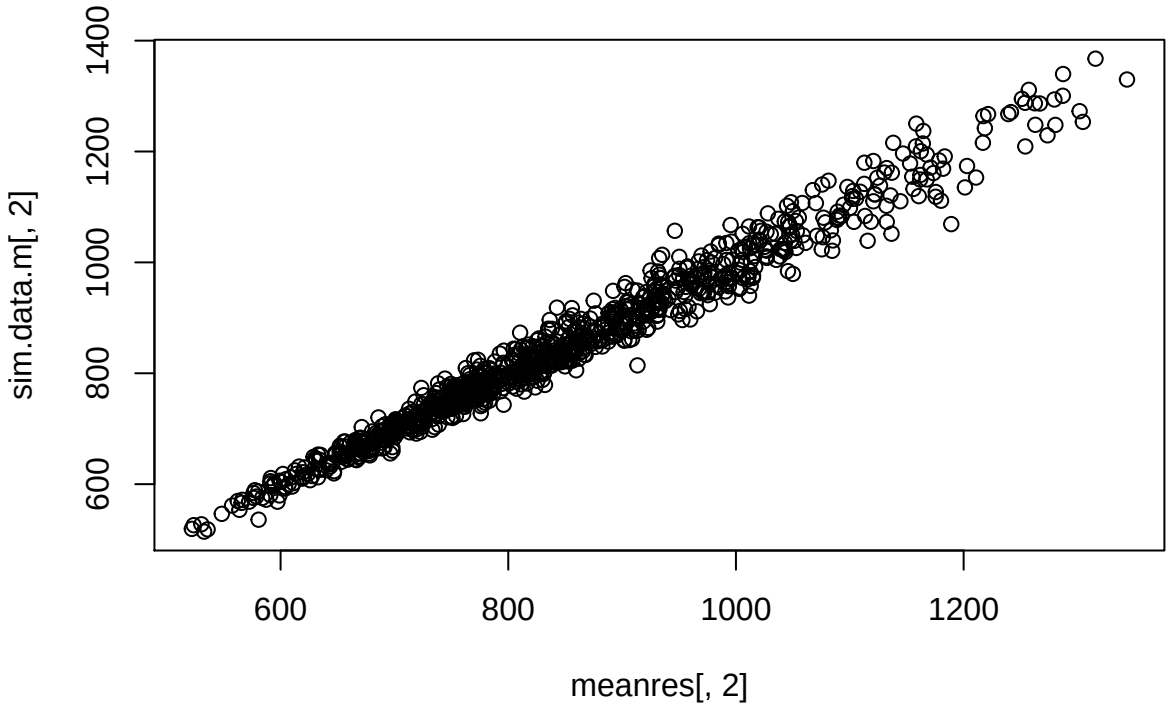
Word: means

```
plot(meanres[,1],sim.data.m[,1])
```



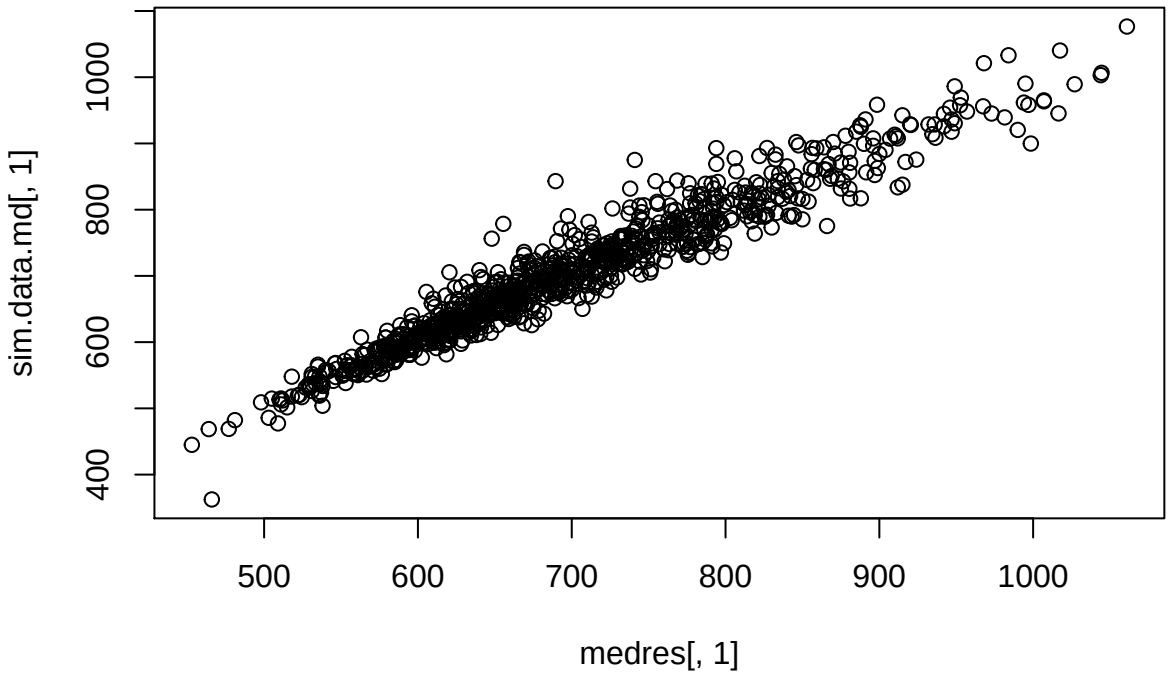
Non-Word: means

```
plot(meanres[,2],sim.data.m[,2])
```



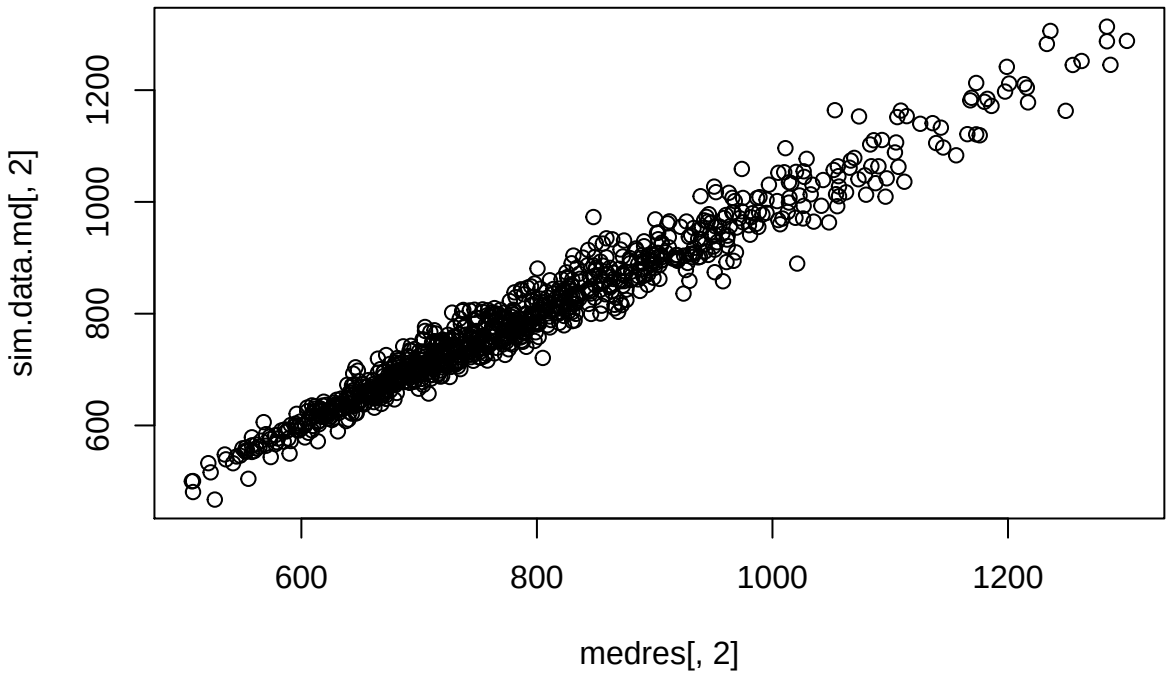
Word: medians

```
plot(medres[,1],sim.data.md[,1])
```



Non-Word: medians

```
plot(medres[,2],sim.data.md[,2])
```



Group simulation 1: same number of trials in each group

```
npseq <- c(25, 50, 100)
ntseq <- c(seq(50, 100, 10), 150, 200)
maxnp <- max(npseq) # max number of participants
maxnt <- max(ntseq) # max number of trials
nsim <- 10000 # simulation samples
qseq <- seq(0.1,0.9,0.1) # define deciles for quantile function

alpha <- 0.05
ES <- 1 # step in interpolated continuum between conditions

# declare matrices of results - save all iterations
res.m.m.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.md.md.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.md.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.tm10.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.tm20.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.m.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))

pvals0 <- vector(mode = "numeric", length = 9)
pvals10 <- vector(mode = "numeric", length = 9)
pvals20 <- vector(mode = "numeric", length = 9)

set.seed(21)

# Generate data for the entire simulation
mc.data1 <- array(NA, dim = c(maxnt, maxnp, nsim))
mc.data2 <- array(NA, dim = c(maxnt, maxnp, nsim))
```

```r
# Resample participants' triads of parameters
bootsample <- matrix(sample(Np.total, maxnp*nsim, replace = TRUE), nrow = nsim)

for(S in 1:nsim){
  for(P in 1:maxnp){
    mc.data1[,P,S] <- rexgauss(maxnt,
      mu = exg_param_interp[bootsample[S,P],1,1],
      sigma = exg_param_interp[bootsample[S,P],2,1],
      tau = exg_param_interp[bootsample[S,P],3,1])

    mc.data2[,P,S] <- rexgauss(maxnt,
      mu = exg_param_interp[bootsample[S,P],1,1+ES],
      sigma = exg_param_interp[bootsample[S,P],2,1+ES],
      tau = exg_param_interp[bootsample[S,P],3,1+ES])
  }
}

for(TR in 1:length(ntseq)){ # number of trials
  print(paste0("number of trials: ",ntseq[TR],"..."))
  beep(2)

  for(PT in 1:length(npseq)){ # number of participants

    # compute estimates
    todo1 <- mc.data1[1:ntseq[TR], 1:npseq[PT],]
    todo2 <- mc.data2[1:ntseq[TR], 1:npseq[PT],]
    mc.m <- apply(todo1, c(2,3), mean) - apply(todo2, c(2,3), mean) # diff in mean RTs
    mc.md <- apply(todo1, c(2,3), median) - apply(todo2, c(2,3), median) # diff in median RTs

    # tests for each simulation ====================
    # one-sample t-test
    res.m.m.sig[, TR, PT] <- ttest.sig(mc.m)
    # parametric median test
    res.md.md.sig[, TR, PT] <- apply(mc.md, 2, sint.sig)

    # shift function
    # array of differences: 9 deciles x participants x simulations
      # mc.hd <- apply(todo1, c(2,3), hdseq) - apply(todo2, c(2,3), hdseq)
      mc.q <- apply(todo1, c(2,3), quantile, probs = qseq, type = 8) -
                apply(todo2, c(2,3), quantile, probs = qseq, type = 8)
      # median test for each decile distribution
      pval <- apply(mc.q, c(1,3), sintv2.pval)
       # trimmed mean tests for each decile distribution
      tvals0 <- apply(mc.q, 1, ctval.yuen, tr = 0)
      tvals10 <- apply(mc.q, 1, ctval.yuen, tr = 0.1)
      tvals20 <- apply(mc.q, 1, ctval.yuen, tr = 0.2)
      df0 <- df.yuen(mc.q[1,,1], tr = 0)
      df10 <- df.yuen(mc.q[1,,1], tr = 0.1)
      df20 <- df.yuen(mc.q[1,,1], tr = 0.2)
      for(S in 1:nsim){
        res.sf.md.sig[S, TR, PT] <- sum(p.adjust(pval[,S], method = "hochberg") <= alpha)
        for(Q in 1:9){
        pvals0[Q] <- cpval(tvals0[S,Q], df0)
```

```
        pvals10[Q] <- cpval(tvals10[S,Q], df10)
        pvals20[Q] <- cpval(tvals20[S,Q], df20)
        }
        res.sf.m.sig[S, TR, PT] <- sum(p.adjust(pvals0, method = "hochberg") <= alpha)
        res.sf.tm10.sig[S, TR, PT] <- sum(p.adjust(pvals10, method = "hochberg") <= alpha)
        res.sf.tm20.sig[S, TR, PT] <- sum(p.adjust(pvals20, method = "hochberg") <= alpha)
      }
  }
}

save(
  res.m.m.sig,
  res.md.md.sig,
  res.sf.m.sig,
  res.sf.md.sig,
  res.sf.tm10.sig,
  res.sf.tm20.sig,
  ntseq,
  npseq,
  nsim,
  file=paste0('./data/sim_gp_tp1_flp.RData'))
beep(8)
```

Compute true positive probability

```
load('./data/sim_gp_tp1_flp.RData')
fp.m.m <- apply(res.m.m.sig, c(2,3), mean)
fp.md.md <- apply(res.md.md.sig, c(2,3), mean)
fp.sf.m <- apply(res.sf.m.sig>0, c(2,3), mean)
fp.sf.md <- apply(res.sf.md.sig>0, c(2,3), mean)
fp.sf.tm10 <- apply(res.sf.tm10.sig>0, c(2,3), mean)
fp.sf.tm20 <- apply(res.sf.tm20.sig>0, c(2,3), mean)
```

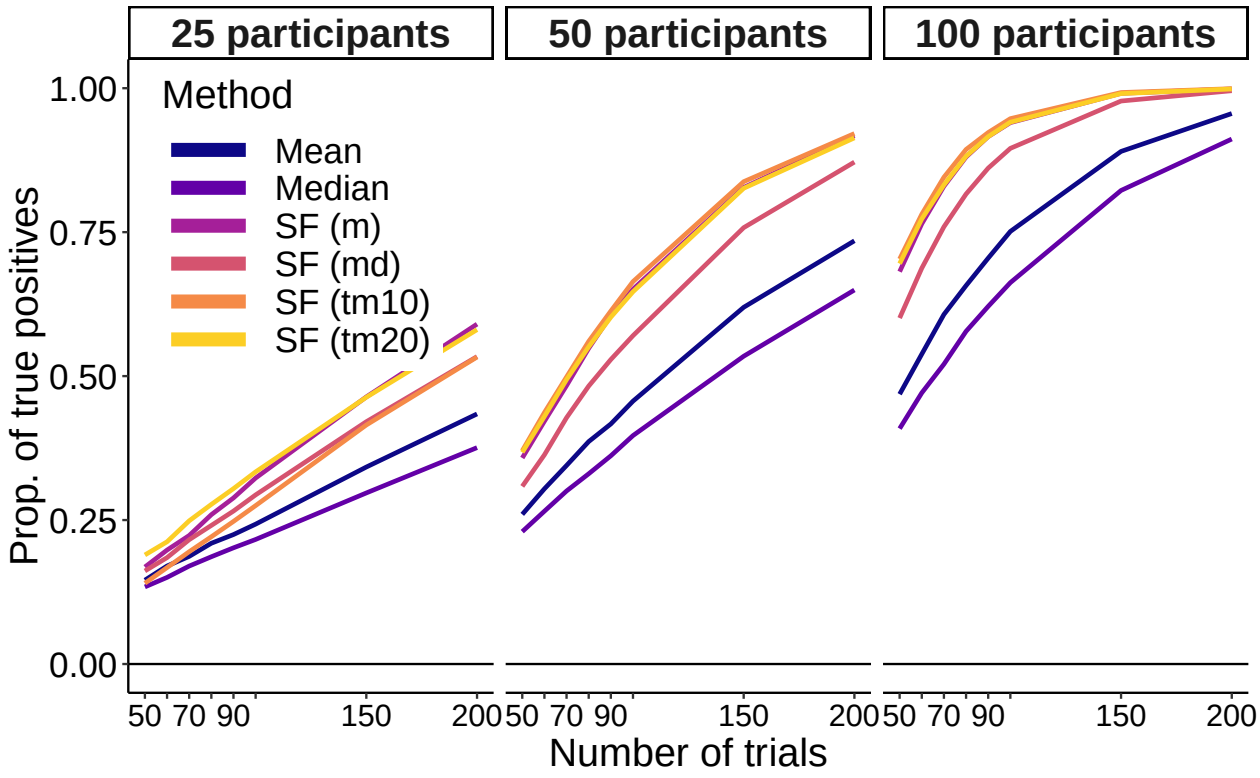
Illustrate results

Make function

```
plot_tp <- function(data, lab.in, ntseq, npseq){
  df <- tibble(`FP`=data,
               `Trials`=rep(ntseq, length(npseq)*length(lab.in)),
               `Participants`=rep(rep(npseq,each=length(ntseq)),length(lab.in)),
               `Method`=rep(lab.in,
                            each = length(ntseq)*length(npseq))
               )

ntseq.labels <- c("50", "", "70", "", "90", "", "150", "200")
labels <- c("25" = "25 participants", "50" = "50 participants", "100" = "100 participants")

df$Method <- as.character(df$Method)
df$Method <- factor(df$Method, levels=unique(df$Method))
```

```
df$Participants <- as.character(df$Participants)
df$Participants <- factor(df$Participants, levels=unique(df$Participants))

# make plot
p <- ggplot(df, aes(x=Trials, y=FP, colour = Method)) + theme_classic() +
  geom_abline(intercept=0, slope=0, colour="black") +
  geom_line(size = 1) +
  scale_colour_viridis_d(option = "plasma", end=0.9) +
  scale_x_continuous(breaks=ntseq,
                     labels = ntseq.labels) +
  coord_cartesian(ylim=c(0,1)) +
  theme(plot.title = element_text(size=22),
        axis.title.x = element_text(size = 18),
        axis.text.x = element_text(size = 14, colour="black"),
        axis.text.y = element_text(size = 16, colour="black"),
        axis.title.y = element_text(size = 18),
        legend.key.width = unit(1.5,"cm"),
        # legend.position = "right",
        legend.position = c(0.15,0.75),
        legend.direction = "vertical",
        legend.text=element_text(size=16),
        legend.title=element_text(size=18),
        strip.text = element_text(size=18, face="bold"),
        strip.background = element_rect(colour="black", fill="white")) +
  labs(x = "Number of trials", y = "Prop. of true positives") +
  guides(colour = guide_legend(override.aes = list(size=3))) + # make thicker legend lines
  facet_grid(cols = vars(Participants), labeller=labeller(Participants = labels))
p
}
```

```
FP.in <- c(as.vector(fp.m.m),as.vector(fp.md.md),
                    as.vector(fp.sf.m),as.vector(fp.sf.md),
                    as.vector(fp.sf.tm10),as.vector(fp.sf.tm20))
lab.in <- c("Mean", "Median", "SF (m)", "SF (md)", "SF (tm10)", "SF (tm20)")
p <- plot_tp(FP.in, lab.in, ntseq, npseq)
p
```



Power comparison

```
nt <- 6 # 100 trials ntseq1[nt]
np <- 2 # 50 participants
mean(res.m.m.sig[,nt,np])
```

```
## [1] 0.4565
```

```
mean(res.md.md.sig[,nt,np])
```

```
## [1] 0.3967
```

```
mean(res.sf.m.sig[,nt,np]>0)
```

```
## [1] 0.6509
```

```
mean(res.sf.md.sig[,nt,np]>0)
```

```
## [1] 0.5702
```

```
mean(res.sf.tm10.sig[,nt,np]>0)
```

```
## [1] 0.6641
```

```
mean(res.sf.tm20.sig[,nt,np]>0)
```

```
## [1] 0.6465
```

Group simulation 2: one group = always 200 trials, other group n trials

```
npseq <- c(25, 50, 100)
maxnp <- max(npseq) # max number of participants
ntseq1 <- c(seq(50, 100, 10), 150, 200)
maxnt1 <- max(ntseq1) # max number of trials
ntseq2 <- rep(200, length(ntseq1)) # number of trials in group 2
maxnt2 <- max(ntseq2) # max number of trials
nsim <- 10000 # simulation samples
qseq <- seq(0.1,0.9,0.1) # define deciles for quantile function

alpha <- 0.05
ES <- 1 # step in interpolated continuum between conditions

# declare matrices of results - save all iterations
res.m.m.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.md.md.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.md.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.tm10.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.tm20.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))
res.sf.m.sig <- array(NA, dim = c(nsim, length(ntseq), length(npseq)))

pvals0 <- vector(mode = "numeric", length = 9)
pvals10 <- vector(mode = "numeric", length = 9)
pvals20 <- vector(mode = "numeric", length = 9)

set.seed(21)

# Generate data for the entire simulation
mc.data1 <- array(NA, dim = c(maxnt1, maxnp, nsim))
mc.data2 <- array(NA, dim = c(maxnt2, maxnp, nsim))

# Resample participants' triads of parameters
bootsample <- matrix(sample(Np.total, maxnp*nsim, replace = TRUE), nrow = nsim)

for(S in 1:nsim){
  for(P in 1:maxnp){
    mc.data1[,P,S] <- rexgauss(maxnt1,
      mu = exg_param_interp[bootsample[S,P],1,1],
      sigma = exg_param_interp[bootsample[S,P],2,1],
      tau = exg_param_interp[bootsample[S,P],3,1])

    mc.data2[,P,S] <- rexgauss(maxnt2,
      mu = exg_param_interp[bootsample[S,P],1,1+ES],
      sigma = exg_param_interp[bootsample[S,P],2,1+ES],
      tau = exg_param_interp[bootsample[S,P],3,1+ES])
  }
}

for(TR in 1:length(ntseq1)){ # number of trials
  print(paste0("number of trials: ",ntseq1[TR],"..."))
```

```
  beep(2)

  for(PT in 1:length(npseq)){ # number of participants

    # compute estimates
    todo1 <- mc.data1[1:ntseq1[TR], 1:npseq[PT],]
    todo2 <- mc.data2[1:ntseq2[TR], 1:npseq[PT],]
    mc.m <- apply(todo1, c(2,3), mean) - apply(todo2, c(2,3), mean) # diff in mean RTs
    mc.md <- apply(todo1, c(2,3), median) - apply(todo2, c(2,3), median) # diff in median RTs

    # tests for each simulation ======================
    # one-sample t-test
    res.m.m.sig[, TR, PT] <- ttest.sig(mc.m)
    # parametric median test
    res.md.md.sig[, TR, PT] <- apply(mc.md, 2, sint.sig)

    # shift function
    # array of differences: 9 deciles x participants x simulations
      # mc.hd <- apply(todo1, c(2,3), hdseq) - apply(todo2, c(2,3), hdseq)
      mc.q <- apply(todo1, c(2,3), quantile, probs = qseq, type = 8) -
                apply(todo2, c(2,3), quantile, probs = qseq, type = 8)
      # median test for each decile distribution
      pval <- apply(mc.q, c(1,3), sintv2.pval)
       # trimmed mean tests for each decile distribution
      tvals0 <- apply(mc.q, 1, ctval.yuen, tr = 0)
      tvals10 <- apply(mc.q, 1, ctval.yuen, tr = 0.1)
      tvals20 <- apply(mc.q, 1, ctval.yuen, tr = 0.2)
      df0 <- df.yuen(mc.q[1,,1], tr = 0)
      df10 <- df.yuen(mc.q[1,,1], tr = 0.1)
      df20 <- df.yuen(mc.q[1,,1], tr = 0.2)
      for(S in 1:nsim){
        res.sf.md.sig[S, TR, PT] <- sum(p.adjust(pval[,S], method = "hochberg") <= alpha)
        for(Q in 1:9){
        pvals0[Q] <- cpval(tvals0[S,Q], df0)
        pvals10[Q] <- cpval(tvals10[S,Q], df10)
        pvals20[Q] <- cpval(tvals20[S,Q], df20)
        }
        res.sf.m.sig[S, TR, PT] <- sum(p.adjust(pvals0, method = "hochberg") <= alpha)
        res.sf.tm10.sig[S, TR, PT] <- sum(p.adjust(pvals10, method = "hochberg") <= alpha)
        res.sf.tm20.sig[S, TR, PT] <- sum(p.adjust(pvals20, method = "hochberg") <= alpha)
      }
  }
}

save(
  res.m.m.sig,
  res.md.md.sig,
  res.sf.m.sig,
  res.sf.md.sig,
  res.sf.tm10.sig,
  res.sf.tm20.sig,
  ntseq1,
  ntseq2,
```

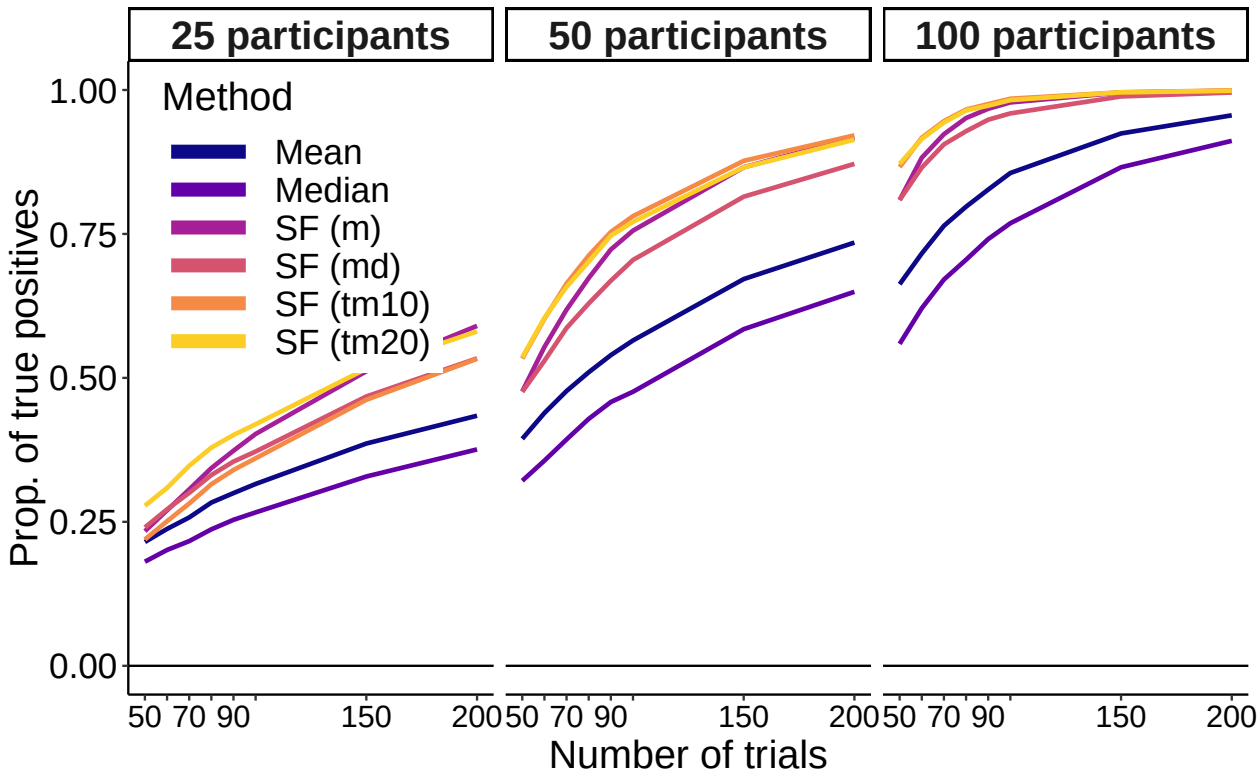
```
  npseq,
  nsim,
  file='./data/sim_gp_tp2_flp.RData')
beep(8)
```

Compute true positive probability

```
load('./data/sim_gp_tp2_flp.RData')
fp.m.m <- apply(res.m.m.sig, c(2,3), mean)
fp.md.md <- apply(res.md.md.sig, c(2,3), mean)
fp.sf.m <- apply(res.sf.m.sig>0, c(2,3), mean)
fp.sf.md <- apply(res.sf.md.sig>0, c(2,3), mean)
fp.sf.tm10 <- apply(res.sf.tm10.sig>0, c(2,3), mean)
fp.sf.tm20 <- apply(res.sf.tm20.sig>0, c(2,3), mean)
```

Illustrate results

```
FP.in <- c(as.vector(fp.m.m),as.vector(fp.md.md),
                    as.vector(fp.sf.m),as.vector(fp.sf.md),
                    as.vector(fp.sf.tm10),as.vector(fp.sf.tm20))
lab.in <- c("Mean", "Median", "SF (m)", "SF (md)", "SF (tm10)", "SF (tm20)")
p <- plot_tp(FP.in, lab.in, ntseq, npseq)
p
```



Power comparison

```
nt <- 6 # 100 trials ntseq1[nt]
np <- 2 # 50 participants
mean(res.m.m.sig[,nt,np])
```

```
## [1] 0.565
```

```
mean(res.md.md.sig[,nt,np])
```

```
## [1] 0.4759
```

```
mean(res.sf.m.sig[,nt,np]>0)
```

```
## [1] 0.7557
```

```
mean(res.sf.md.sig[,nt,np]>0)
```

```
## [1] 0.7048
```

```
mean(res.sf.tm10.sig[,nt,np]>0)
```

```
## [1] 0.7809
```

```
mean(res.sf.tm20.sig[,nt,np]>0)
```

```
## [1] 0.771
```

Summary figure

Show mean, median, and SF TM20 only.

```
# Same numbers of trials
load('./data/sim_gp_tp1_flp.RData')
fp.m.m <- apply(res.m.m.sig, c(2,3), mean)
fp.md.md <- apply(res.md.md.sig, c(2,3), mean)
fp.sf.tm20 <- apply(res.sf.tm20.sig>0, c(2,3), mean)

FP.in <- c(as.vector(fp.m.m),as.vector(fp.md.md),as.vector(fp.sf.tm20))
lab.in <- c("Mean", "Median", "SF")
p.same <- plot_tp(FP.in, lab.in, ntseq, npseq) + ggtitle("n1 = n2")

# Different numbers of trials
load('./data/sim_gp_tp2_flp.RData')
fp.m.m <- apply(res.m.m.sig, c(2,3), mean)
fp.md.md <- apply(res.md.md.sig, c(2,3), mean)
fp.sf.tm20 <- apply(res.sf.tm20.sig>0, c(2,3), mean)

FP.in <- c(as.vector(fp.m.m),as.vector(fp.md.md),as.vector(fp.sf.tm20))
lab.in <- c("Mean", "Median", "SF")
p.diff <- plot_tp(FP.in, lab.in, ntseq, npseq) +
          theme(legend.position = "none") + ggtitle("n2 = 200")

# combine panels into one figure
cowplot::plot_grid(p.same, p.diff,
                           labels = c("A", "B"),
                           ncol = 1,
```

```
                          nrow = 2,
                          # rel_widths = c(1, 1, 1),
                          label_size = 20,
                          hjust = -0.5,
                          scale=.95,
                          align = "h")
# save figure
ggsave(filename='./figures/figure_gp_tp_flp_summary.pdf',width=10,height=8)
```