Replicate simulations from Miller 1988

Guillaume A. Rousselet and Rand R. Wilcox

2019-04-18

Contents

Define Miller's ex-Gaussian parameters **2**

Estimate population parameters from large samples **3**
Load ex-Gaussian parameters . . . 3

Illustrate 12 distributions **3**

Population parameters **5**

Simulation **6**

Compute bias **7**

Illustrate bias results **8**
Make plot function . . . 8
Mean . . . 9
Median . . . 10
Illustrate bias corrected median results . . . 10
Illustrate median bias of the median . . . 11

Illustrate median bias of the mean **11**
Summary figure . . . 12

Bias/SE $<$ 0.25? **12**
Compute bias/SE . . . 12
Illustrate P(bias/SE $>$ 0.25) . . . 13

Check bias correction **14**
Illustrate results: neg over wrong . . . 21
Illustrate results: pos under right . . . 23

```
# dependencies
library(ggplot2)
library(tibble)
library(tidyr)
library(cowplot)
library(retimes)
library(knitr)
library(HDInterval)
source("./functions/skew.txt")
source("./functions/akerd.txt")
library(beepr)
```

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
```

```
## Running under: macOS Mojave 10.14.4
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.5/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.5/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        HDInterval_0.2.0 knitr_1.21       retimes_0.1-2
## [5] cowplot_0.9.4    tidyr_0.8.2      tibble_2.0.1     ggplot2_3.1.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.0       pillar_1.3.1     compiler_3.5.2   plyr_1.8.4
##  [5] tools_3.5.2      digest_0.6.18    evaluate_0.13    gtable_0.2.0
##  [9] pkgconfig_2.0.2  rlang_0.3.1      yaml_2.2.0       xfun_0.4
## [13] withr_2.1.2      dplyr_0.8.0.1    stringr_1.4.0    grid_3.5.2
## [17] tidyselect_0.2.5 glue_1.3.0       R6_2.4.0         rmarkdown_1.11
## [21] purrr_0.3.0      magrittr_1.5     scales_1.0.0     htmltools_0.3.6
## [25] assertthat_0.2.0 colorspace_1.4-0 stringi_1.3.1    lazyeval_0.2.1
## [29] munsell_0.5.0    crayon_1.3.4     audio_0.1-5.1
```

We replicate and expand the simulations from:

> Miller, J. (1988). A warning about median reaction time. Journal of Experimental Psychology: Human Perception and Performance, 14(3), 539.

Define Miller's ex-Gaussian parameters

From Miller's Table 1. Save matrix with 3 parameters: - mean of the normal distribution - standard deviation of the normal distribution - mean of the exponential distribution

```
miller.param <- matrix(0, ncol=3, nrow=12)
miller.param[1,] <- c(300, 20, 300)
miller.param[2,] <- c(300, 50, 300)
miller.param[3,] <- c(350, 20, 250)
miller.param[4,] <- c(350, 50, 250)
miller.param[5,] <- c(400, 20, 200)
miller.param[6,] <- c(400, 50, 200)
miller.param[7,] <- c(450, 20, 150)
miller.param[8,] <- c(450, 50, 150)
miller.param[9,] <- c(500, 20, 100)
miller.param[10,] <- c(500, 50, 100)
miller.param[11,] <- c(550, 20, 50)
miller.param[12,] <- c(550, 50, 50)
```

Estimate population parameters from large samples

Miller used 10,000 samples; we use 1,000,000.

```
set.seed(4)
pop.m <- vector(mode="numeric", length=12)
pop.md <- vector(mode="numeric", length=12)
pop.sk <- matrix(NA, nrow=12, ncol=2)
n <- 1000000
nP <- length(miller.param[,1])
for(P in 1:nP){
  mu <- miller.param[P,1]
  sigma <- miller.param[P,2]
  tau <- miller.param[P,3]
  pop <- rexgauss(n, mu = mu, sigma = sigma, tau = tau)
  pop.m[P] <- mean(pop)
  pop.md[P] <- sort(pop)[round(length(pop)*0.5)] # median(pop)
  pop.sk[P,1] <- skew(pop)
  pop.sk[P,2] <- kurt(pop)
}
save(
  miller.param,
  pop.m,
  pop.md,
  pop.sk,
  nP,
  file = './data/miller_exg_param.RData'
)
```

Load ex-Gaussian parameters

```
load('./data/miller_exg_param.RData')
```

Illustrate 12 distributions

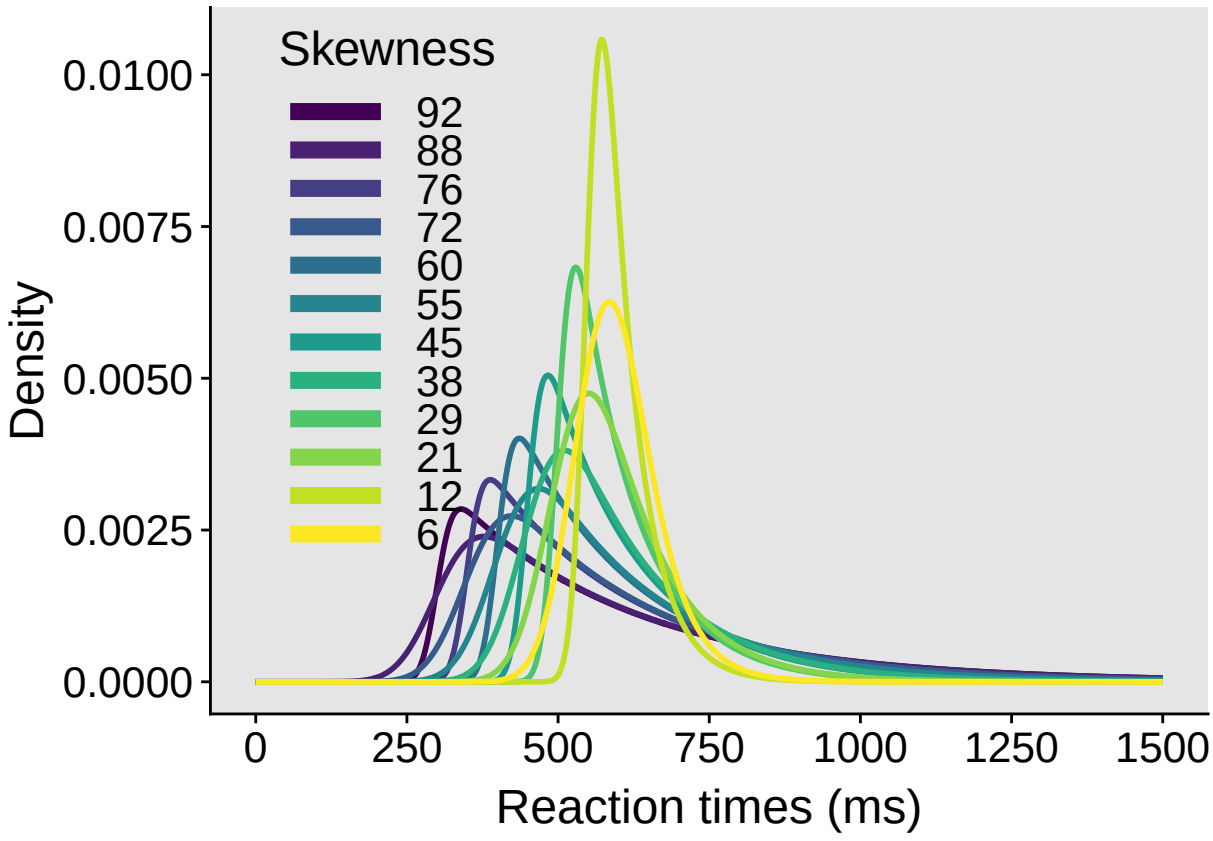
Plug Miller's parameters into ex-Gaussian density function.

```
x <- seq.int(0,1500,0.1)
mdist <- matrix(0, nrow=length(x), ncol=nP)
for(P in 1:nP){
  mu <- miller.param[P,1]
  sigma <- miller.param[P,2]
  tau <- miller.param[P,3]
  mdist[,P] <- dexgauss(x, mu = mu, sigma = sigma, tau = tau)
}
```

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

```
# make data frame
fm <- array(0, dim = c(length(x), nP+1)) # make full matrix
fm[,1] <- x
fm[,2:(nP+1)] <- mdist
```

```r
colnames(fm) <- c("x",round(pop.m - pop.md))
df <- as_tibble(fm)
df <- tidyr::gather(df, Skewness, Density,2:(nP+1))
df[[2]] <- as.character(df[[2]])
df[[2]] <- factor(df[[2]], levels=unique(df[[2]]))

# make plot
p <- ggplot(df, aes(x, Density)) +
          geom_line(aes(colour = Skewness), size = 1) +
          scale_colour_viridis_d() +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16),
                axis.title.y = element_text(size = 18),
                legend.key.width = unit(1.5,"cm"),
                legend.text = element_text(size = 16),
                legend.title = element_text(size = 18),
                legend.position = c(0.05,0.6),
                panel.background = element_rect(fill="grey90")) +
          scale_x_continuous(limits = c(0, 1500),
                             breaks = seq(0, 1500, 250)) +
  labs(x = "Reaction times (ms)", y = "Density") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
p
```




```r
# save figure
ggsave(filename='./figures/figure_miller_distributions.pdf',width=7,height=5)
```

Population parameters

Summary table of population mean, median and non-parametric skewness for each of the 12 distributions. Non-parametric skewness is defined as mean - median.

```
tmp <- cbind(miller.param, round(pop.m), round(pop.md), round(pop.m - pop.md))
colnames(tmp) <- list("mu", "sigma", "tau", "mean", "median", "skewness")
kable(tmp, caption="Population parameters")
```

Table 1: Population parameters

mu	sigma	tau	mean	median	skewness
300	20	300	600	509	92
300	50	300	600	512	88
350	20	250	600	524	76
350	50	250	600	528	72
400	20	200	600	540	60
400	50	200	600	544	55
450	20	150	600	555	45
450	50	150	600	562	38
500	20	100	600	572	29
500	50	100	600	579	21
550	20	50	600	588	12
550	50	50	600	594	6

Table of parametric skewness & kurtosis. The relative differences between levels of parametric skewness and non-parametric skewness differ substantially. Kurtosis also varies non-linearly. Obviously, the shapes of Miller 1988's distributions differ in several ways not captured by non-parametric skewness. But that's not important for the problem at hand, because the goal is to estimate bias from distributions with increasing differences between mean and median. No assumption is made about other aspects of the distributions.

```
tmp <- round(pop.sk,digits = 1)
colnames(tmp) <- list("Skewness", "Kurtosis")
kable(tmp, caption="Parametric skewness & kurtosis")
```

Table 2: Parametric skewness & kurtosis

Skewness	Kurtosis
2.0	8.9
1.9	8.6
2.0	8.9
1.9	8.6
2.0	8.9
1.8	8.3
1.9	8.7
1.7	7.9
1.9	8.5
1.4	6.9
1.6	7.5
0.7	4.5

Simulation

10,000 random samples of sizes 4, 6, 8, 10, 15, 20, 25, 35, 50 and 100, for each of the 12 distributions. We save all the sample means and medians to calculate bias (see below) and illustrate the distributions (in file `samp_dist`).

```
nvec <- c(4, 6, 8, 10, 15, 20, 25, 35, 50, 100)
maxn <- max(nvec)
nsim <- 10000 # simulation samples
nboot <- 200 # bootstrap bias correction

# declare matrices of results - save all iterations
sim.m <- array(NA, dim = c(nsim, nP, length(nvec))) # mean
sim.md <- array(NA, dim = c(nsim, nP, length(nvec))) # median
sim.md.bias <- array(NA, dim = c(nsim, nP, length(nvec))) # bias
sim.md.se <- array(NA, dim = c(nsim, nP, length(nvec))) # standard error
sim.md.bc <- array(NA, dim = c(nsim, nP, length(nvec))) # bias-correction median

set.seed(21)

  for(P in 1:nP){ # ex-Gaussian parameters
    print(paste0("parameters: ",P," out of ",nP,"..."))
    beep(2)

    # generate max sample size data first, then downsample
    mu <- miller.param[P,1]
    sigma <- miller.param[P,2]
    tau <- miller.param[P,3]
    max.data <- matrix(rexgauss(maxn*nsim, mu = mu, sigma = sigma, tau = tau), nrow=nsim)

    for(iter.n in 1:length(nvec)){ # sample sizes
  print(paste0("sample size: ",nvec[iter.n],"..."))

      mc.data <- max.data[,1:nvec[iter.n]]

    # compute estimates
    sim.m[,P,iter.n] <- apply(mc.data, 1, mean) # mean RTs
    sim.md[,P,iter.n] <- apply(mc.data, 1, median) # median RTs

    # compute bias corrected estimates using the same nboot bootstrap samples for all estimators
    bc.md <- vector(mode="numeric", length=nsim)
    for(iter in 1:nsim){
      boot.md <- apply(matrix(sample(mc.data[iter,], nvec[iter.n]*nboot, replace = TRUE), nrow=nboot),
     sim.md.bc[iter,P,iter.n] <- 2 * sim.md[iter,P,iter.n] - mean(boot.md)
     sim.md.se[iter,P,iter.n] <- sd(boot.md) # bootstrap SE
     sim.md.bias[iter,P,iter.n] <- mean(boot.md) - sim.md[iter,P,iter.n] # estimated bias
    }
  }
}
save(
  sim.m,
  sim.md,
  sim.md.bc,
  sim.md.bias,
```

```
  sim.md.se,
  nvec,
  nsim,
  nboot,
  file=paste0('./data/sim_miller1988.RData'))
beep(8)
```

Compute bias

Bias = average of sample estimates minus population value.

```
load('./data/sim_miller1988.RData')
bias.m <- apply(sim.m, c(2,3), mean) - matrix(rep(pop.m, length(nvec)),nrow=nP)
bias.m.md <- apply(sim.m, c(2,3), median) - matrix(rep(pop.m, length(nvec)),nrow=nP)
bias.md <- apply(sim.md, c(2,3), mean) - matrix(rep(pop.md, length(nvec)),nrow=nP)
bias.md.md <- apply(sim.md, c(2,3), median) - matrix(rep(pop.md, length(nvec)),nrow=nP)
bias.md.bc <- apply(sim.md.bc, c(2,3), mean) - matrix(rep(pop.md, length(nvec)),nrow=nP)

# 50% HDIs of mean and median biases
hdi.m <- array(NA, dim = c(2, nP, length(nvec)))
hdi.md <- array(NA, dim = c(2, nP, length(nvec)))
hdi.md.bc <- array(NA, dim = c(2, nP, length(nvec)))
for(iter.n in 1:length(nvec)){
  for(P in 1:nP){
      hdi.m[, P, iter.n] <- hdi(sim.m[, P, iter.n]-pop.m[P], credMass=0.50)
      hdi.md[, P, iter.n] <- hdi(sim.md[, P, iter.n]-pop.md[P], credMass=0.50)
      hdi.md.bc[, P, iter.n] <- hdi(sim.md.bc[, P, iter.n]-pop.md[P], credMass=0.50)
  }
}
```

Table of bias results

```
tmp <- round(bias.md)
colnames(tmp) <- nvec
rownames(tmp) <- round(pop.m-pop.md)
kable(tmp, caption = "Table of bias simulation results", booktabs = T)
```

Table 3: Table of bias simulation results

	4	6	8	10	15	20	25	35	50	100
92	41	27	19	15	9	7	6	4	3	1
88	41	27	19	14	9	8	6	4	3	1
76	35	23	16	13	8	6	4	3	2	1
72	33	22	16	13	8	7	5	4	3	1
60	28	17	12	10	6	5	3	3	2	1
55	26	17	13	10	6	5	4	3	2	1
45	21	14	10	8	5	4	3	2	2	1
38	17	11	7	6	3	3	2	2	1	1
29	13	9	6	5	3	2	2	1	1	0
21	10	6	4	4	2	2	2	1	1	1
12	5	3	3	2	1	1	1	1	0	0
6	3	2	1	1	1	1	1	0	0	0

Columns = sample sizes

Rows = skewness

Our results are very close to Miller's:

Table 1

Amount of Overestimation (in Milliseconds) of Population Median By Sample Median (OA) and Standard Error of Sample Median (SE), as a Function of Sample Size and Population Distribution: Reaction Time Distributions Modeled as Normal Plus Exponential

Distribution and statistic	$\mu - Md$	Sample size									
		4	6	8	10	15	20	25	35	50	100
N(300, 20) + *E*(300)	92										
OA		42	28	18	17	10	7	8	5	4	2
SE		144	120	103	92	79	66	61	51	42	30
N(300, 50) + *E*(300)	88										
OA		48	30	21	18	13	11	9	7	7	5
SE		149	121	103	92	80	67	61	51	42	30
N(350, 20) + *E*(250)	70										
OA		32	18	13	8	5	3	0	−1	−2	−3
SE		121	100	87	77	66	55	51	42	35	25
N(350, 50) + *E*(250)	71										
OA		34	22	16	12	8	5	4	3	2	1
SE		125	102	87	79	67	55	51	42	35	25
N(400, 20) + *E*(200)	61										
OA		27	17	14	10	7	5	4	3	2	1
SE		99	79	70	62	54	44	40	34	28	20
N(400, 50) + *E*(200)	54										
OA		27	17	12	11	6	6	4	4	2	1
SE		101	81	71	63	53	45	40	34	28	20
N(450, 20) + *E*(150)	36										
OA		11	6	1	−1	−4	−4	−6	−7	−7	−8
SE		69	58	49	44	38	32	29	24	20	14
N(450, 50) + *E*(150)	37										
OA		15	11	6	6	3	2	1	0	0	0
SE		74	59	52	46	39	33	30	25	21	15
N(500, 20) + *E*(100)	27										
OA		13	9	6	4	2	2	1	0	0	0
SE		49	40	35	31	26	22	20	17	14	10
N(500, 50) + *E*(100)	21										
OA		8	5	3	2	1	1	1	0	0	−1
SE		55	45	40	35	30	25	23	19	16	11
N(550, 20) + *E*(50)	12										
OA		5	3	2	2	1	1	1	0	0	0
SE		26	21	18	17	14	12	11	9	8	5
N(550, 50) + *E*(50)	5										
OA		2	1	−1	−1	−1	−1	−1	−1	−1	−1
SE		37	30	27	24	21	18	16	14	11	8

Note. Each distribution is the sum of independent normal and exponential random variables. The parameters of the normal (*N*) distribution are its mean and standard deviation, and the parameter of the exponential (*E*) distribution is its mean. The column labeled "$\mu - Md$" gives the difference between the mean and the median for each distribution.

Illustrate bias results

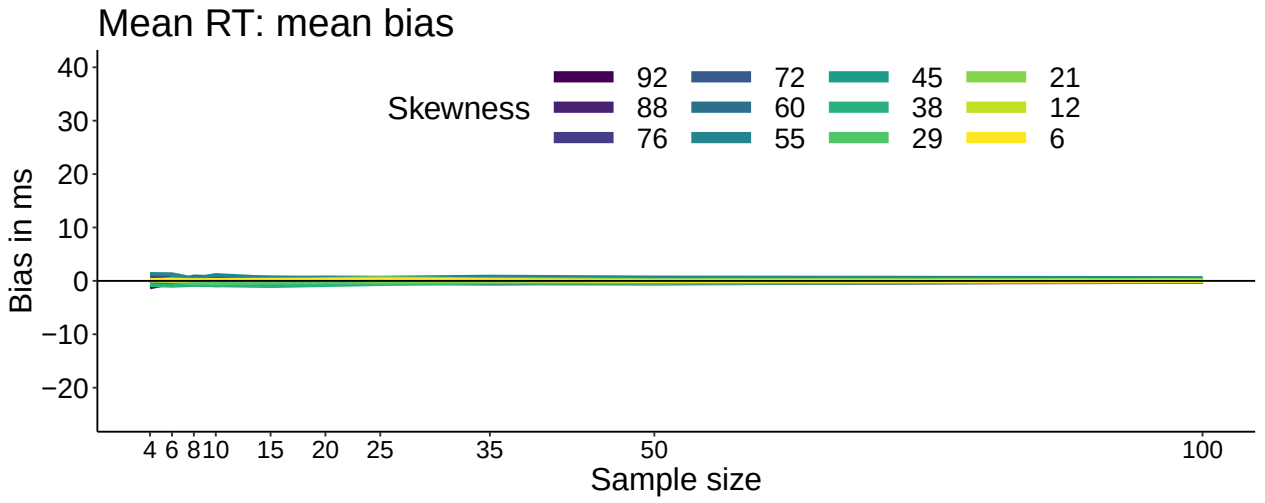
Make plot function

```
plot_bias <- function(data, nvec, nP, pop.m, pop.md){
  df <- tibble(`Bias`=as.vector(data),
               `Size`=rep(nvec,each=nP),
               `Skewness`=rep(round(pop.m - pop.md),length(nvec)))

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

```
# make plot
p <- ggplot(df) + theme_classic() +
  geom_line(aes(x=Size, y=Bias, colour = Skewness), size = 1) +
  geom_abline(intercept=0, slope=0, colour="black") +
  scale_colour_viridis_d() +
  scale_x_continuous(breaks=nvec) +
  scale_y_continuous(breaks=seq(-30,40,10)) +
  coord_cartesian(ylim=c(-25,40)) +
  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 = c(0.85,0.70),
        legend.position = c(0.55,0.85),
        legend.direction = "horizontal",
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Sample size", y = "Bias in ms") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
p
}
```

Mean

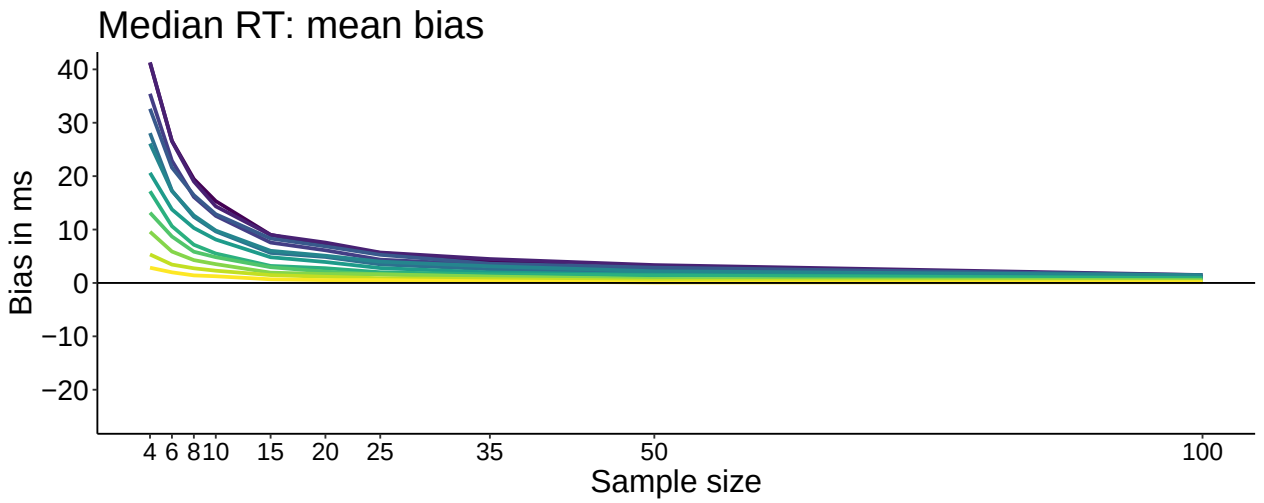
```
p <- plot_bias(bias.m, nvec, nP, pop.m, pop.md) +
      ggtitle("Mean RT: mean bias")
p
```




```
pm <- p
```

Median

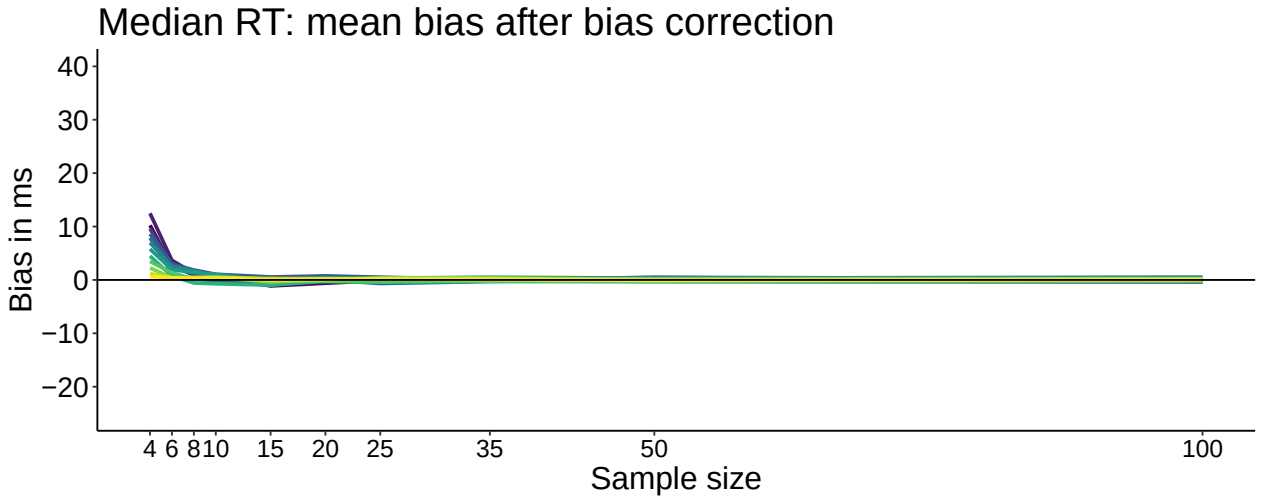
```
p <- plot_bias(bias.md, nvec, nP, pop.m, pop.md) +
    theme(legend.position = "blank") +
    ggtitle("Median RT: mean bias")
p
```




```
pmd <- p
```

Illustrate bias corrected median results

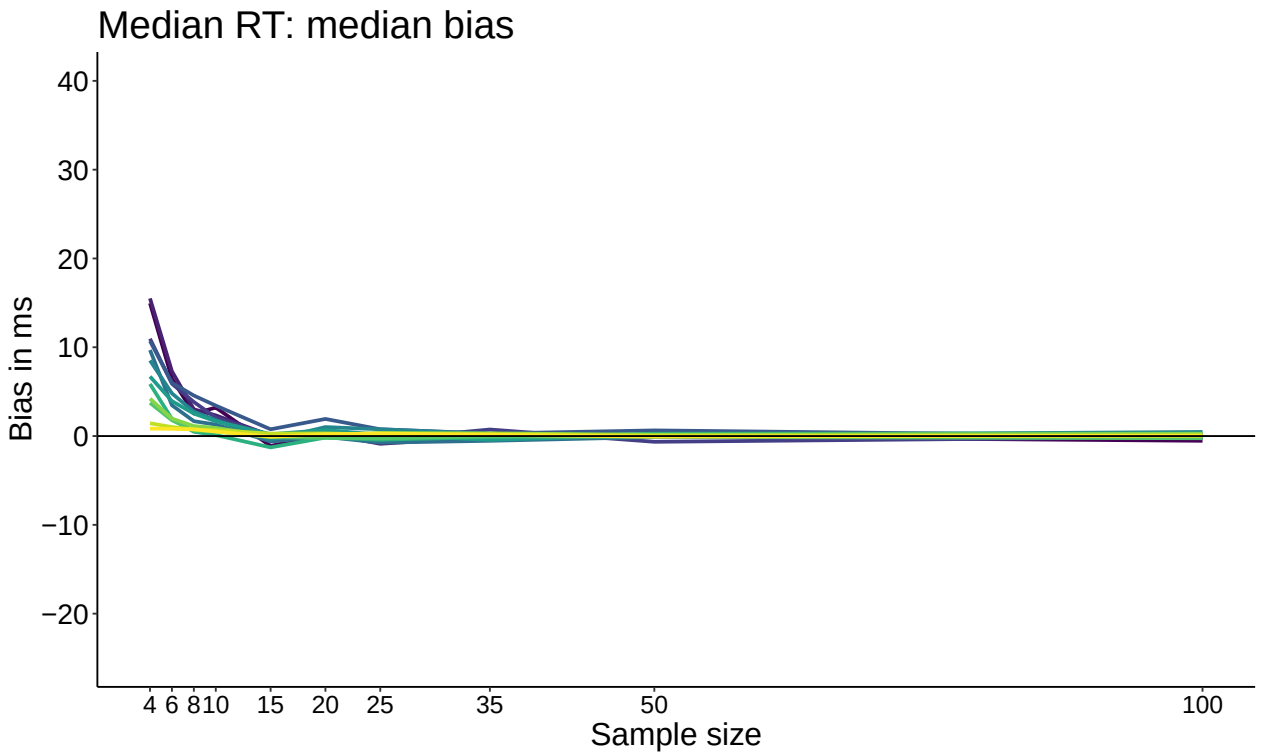
```
p <- plot_bias(bias.md.bc, nvec, nP, pop.m, pop.md) +
    theme(legend.position = "blank") +
    ggtitle("Median RT: mean bias after bias correction")
p
```




```
pmd.bc <- p
```

Illustrate median bias of the median

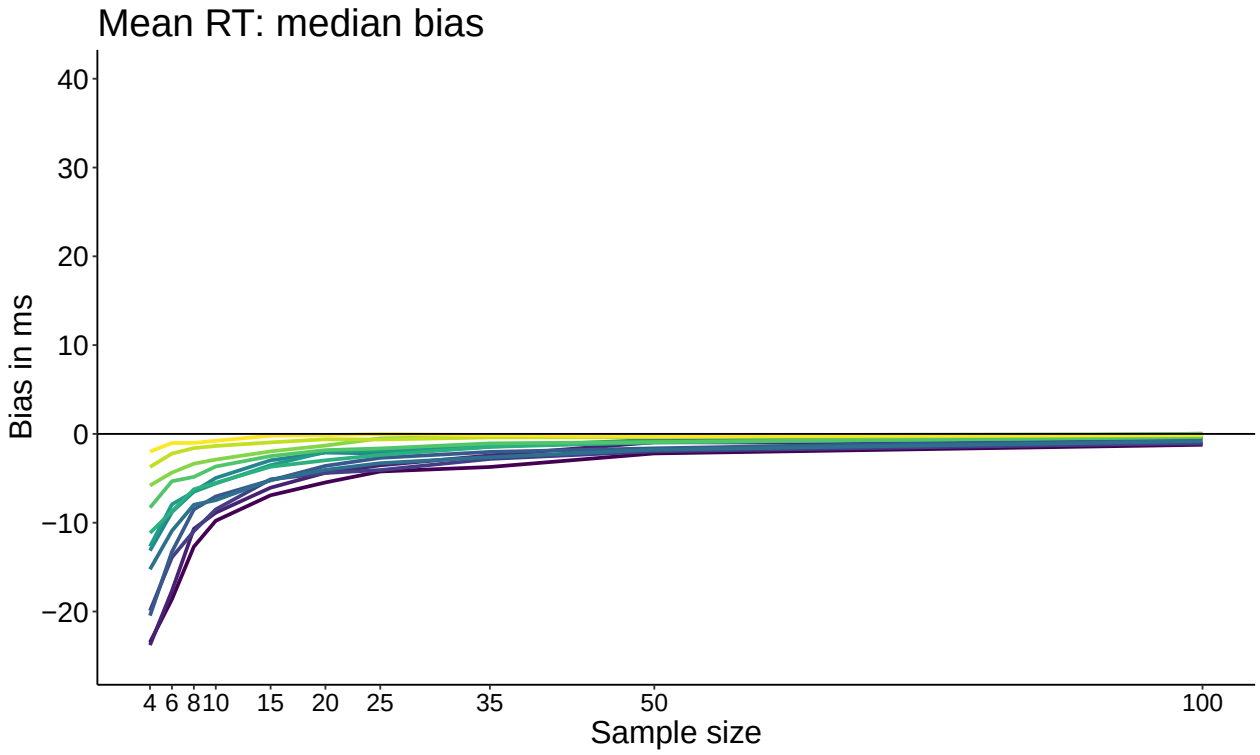
```
p <- plot_bias(bias.md.md, nvec, nP, pop.m, pop.md) +
    theme(legend.position = "blank") +
    ggtitle("Median RT: median bias")
p
```




```
pmd.md <- p
```

Illustrate median bias of the mean

```
p <- plot_bias(bias.m.md, nvec, nP, pop.m, pop.md) +
    theme(legend.position = "blank") +
    ggtitle("Mean RT: median bias")
p
```




```
pm.md <- p
```

Summary figure

```
# combine panels into one figure
cowplot::plot_grid(pm, pmd, NULL, pmd.bc, pm.md, pmd.md,
                           labels = c("A", "B", "", "C", "D", "E"),
                           ncol = 2,
                           nrow = 3,
                           label_size = 20,
                           hjust = -0.5,
                           scale=.95,
                           align = "h")
# save figure
ggsave(filename=('./figures/figure_miller_bias_summary.pdf'),width=14,height=15)
```

Bias/SE < 0.25?

Efron & Tibshirani (1993) suggest that as a rule of thumb, bias/SE < 0.25 can be ignored.

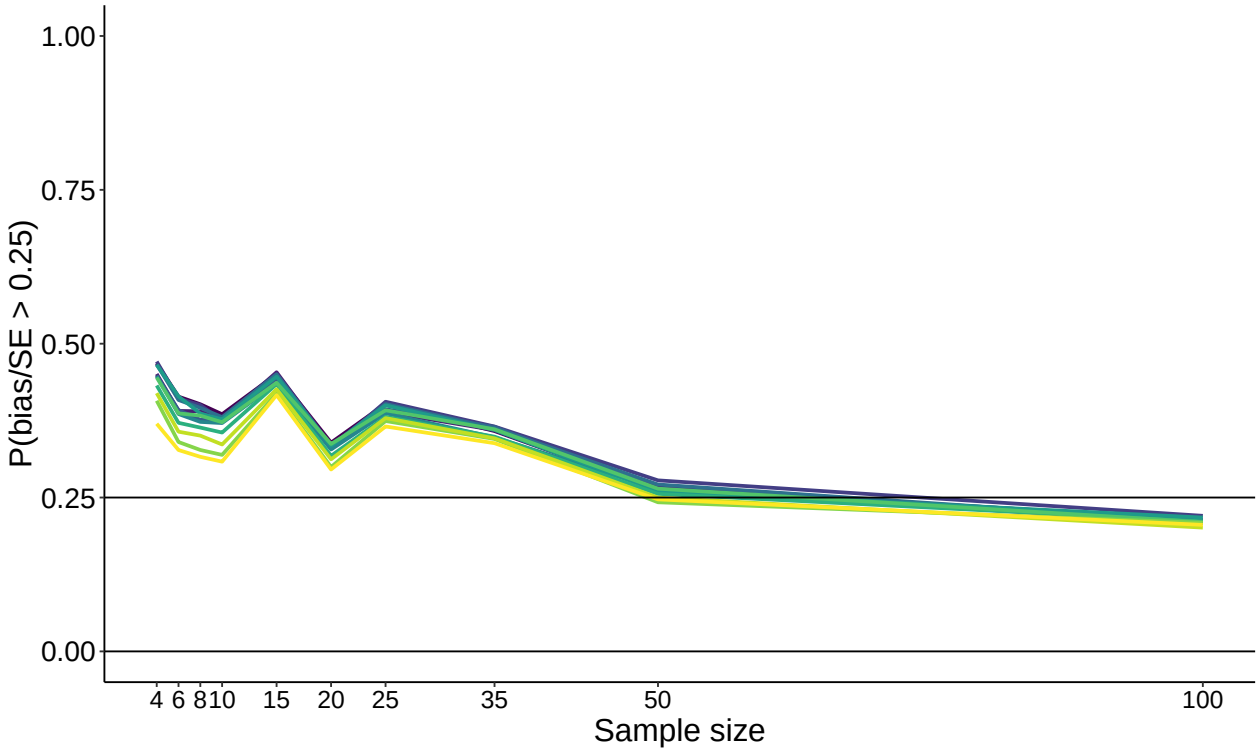
Compute bias/SE

Compute proportion of simulations with bias/SE > 0.25

```
load('./data/sim_miller1988.RData')
pbse <- apply(abs((sim.md.bias / sim.md.se)) > 0.25, c(2,3), mean)
```

Illustrate P(bias/SE > 0.25)

```
df <- tibble(`Bias`=as.vector(pbse),
             `Size`=rep(nvec,each=nP),
             `Skew`=rep(round(pop.m - pop.md),length(nvec)))

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

# make plot
p <- ggplot(df, aes(x=Size, y=Bias), group=Skew) + theme_classic() +
  geom_line(aes(colour = Skew), size = 1) +
  geom_abline(intercept=0.25, slope=0, colour="black") +
  geom_abline(intercept=0, slope=0, colour="black") +
  scale_colour_viridis_d() +
  scale_x_continuous(breaks=nvec) +
  scale_y_continuous(limits=c(0,1), breaks=seq(0,1,0.25)) +
  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 = "blank",#c(0.85,0.65),
    legend.text=element_text(size=16),
    legend.title=element_text(size=18)) +
  labs(x = "Sample size", y = "P(bias/SE > 0.25)") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
p
```

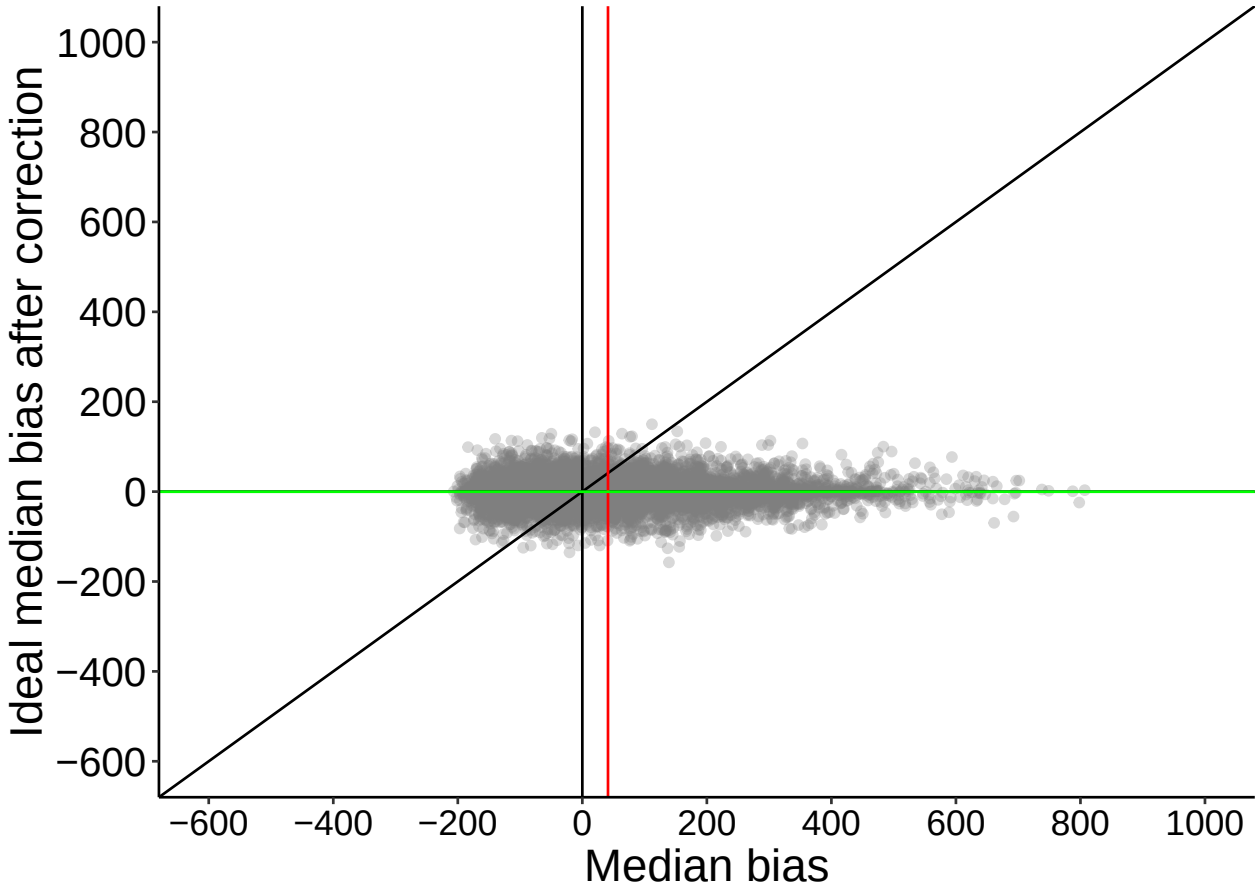


The proportion of experiments / simulations with bias/SE ratio larger than 0.25 is high overall, and decreases with sample size. So, following Efron & Tibshirani (1993)'s rule of thumb, median bias will tend to impact the accuracy of confidence intervals, especially with small sample sizes.

Check bias correction

Here we investigate how well bias correction works as a function of sample size and skewness. The smaller the sample size, the less the bootstrap can estimate the sampling distribution and its bias. Ideally, after bias correction, we would expect the sample medians to be centred around zero, with limited variability. This ideal situation could look something like that, considering the most skewed distribution with the smallest sample size.

```
# load('./data/sim_miller1988.RData')
P <- 1 # 1=most skewed distribution; 12=least
N <- 1 # 1=smallest sample size; 10=largest

set.seed(21)
df <- tibble(`ORI`=sim.md[,P,N]-pop.md[P],
             `BC`=runif(nsim, min=-50, max=50)*rnorm(nsim))

# for(N in 1:length(nvec)){
# make plot
p <- ggplot(df, aes(x=ORI, y=BC)) + theme_classic() +
  # geom_line(aes(colour = Skewness), size = 1) +
  geom_point(colour="grey50", alpha=0.3) +
  geom_abline(intercept=0, slope=1, colour="black") +
  geom_vline(xintercept = 0, colour="black") +
  geom_hline(yintercept = 0, colour="black") +
  geom_vline(xintercept = mean(df$ORI), colour="red") + # bias
  geom_hline(yintercept = mean(df$BC), colour="green") + # bias after BBC
  # scale_colour_viridis_d() +
  scale_x_continuous(breaks=seq(-600,1000,200)) +
  scale_y_continuous(breaks=seq(-600,1000,200)) +
  coord_cartesian(xlim=c(-600,1000), ylim=c(-600,1000)) +
  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 = "blank",#c(0.85,0.65),
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Median bias", y = "Ideal median bias after correction") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
  # ggtitle("Bias correction: direction")
p
```



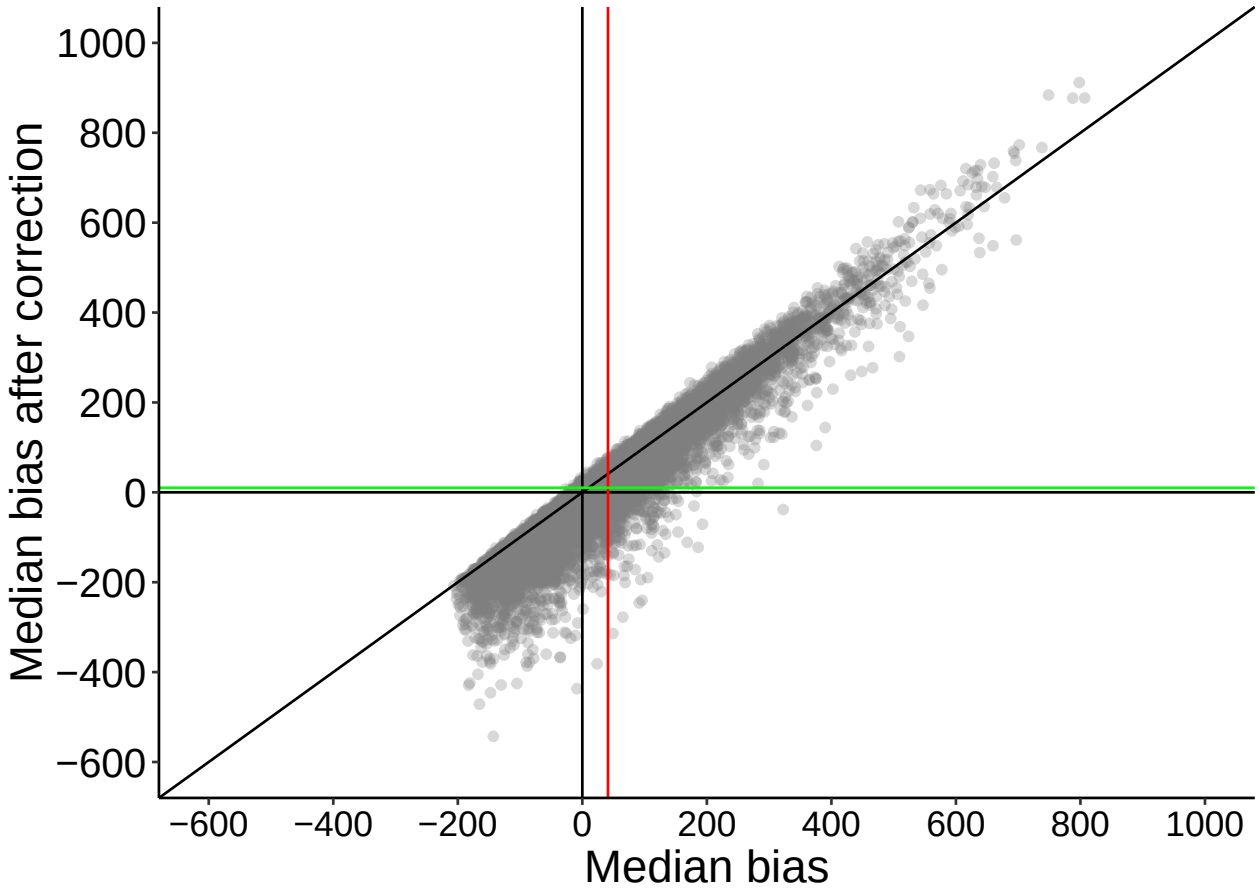

```
# p.bc.dir <- p
# save figure
# ggsave(filename=paste0('figure_miller_bc_ideal_P',P,'_N',nvec[N],'.pdf'),width=10,height=7)
# }
```

The reality is very different.

Figure without markup:

```
# load('./data/sim_miller1988.RData')
P <- 1 # 1=most skewed distribution; 12=least
N <- 1 # 1=smallest sample size; 10=largest

# for(N in 1:length(nvec)){
  # for(P in 1:nP){

    df <- tibble(`ORI`=sim.md[,P,N]-pop.md[P],
                 `BC`=sim.md.bc[,P,N]-pop.md[P])

    # make plot
    p <- ggplot(df, aes(x=ORI, y=BC)) + theme_classic() +
      geom_point(colour="grey50", alpha=0.3) +
      geom_abline(intercept=0, slope=1, colour="black") +
      geom_vline(xintercept = 0, colour="black") +
      geom_hline(yintercept = 0, colour="black") +
      geom_vline(xintercept = mean(df$ORI), colour="red") + # bias
      geom_hline(yintercept = mean(df$BC), colour="green") + # bias after BBC
```

```
    scale_x_continuous(breaks=seq(-600,1000,200)) +
    scale_y_continuous(breaks=seq(-600,1000,200)) +
    coord_cartesian(xlim=c(-600,1000), ylim=c(-600,1000)) +
    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 = c(0.15,0.65),
      legend.text=element_text(size=18),
      legend.title=element_blank()) + #element_text(size=18)) +
    labs(x = "Median bias", y = "Median bias after correction") +
    guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
  # ggtitle("Bias correction: direction")
  p
```

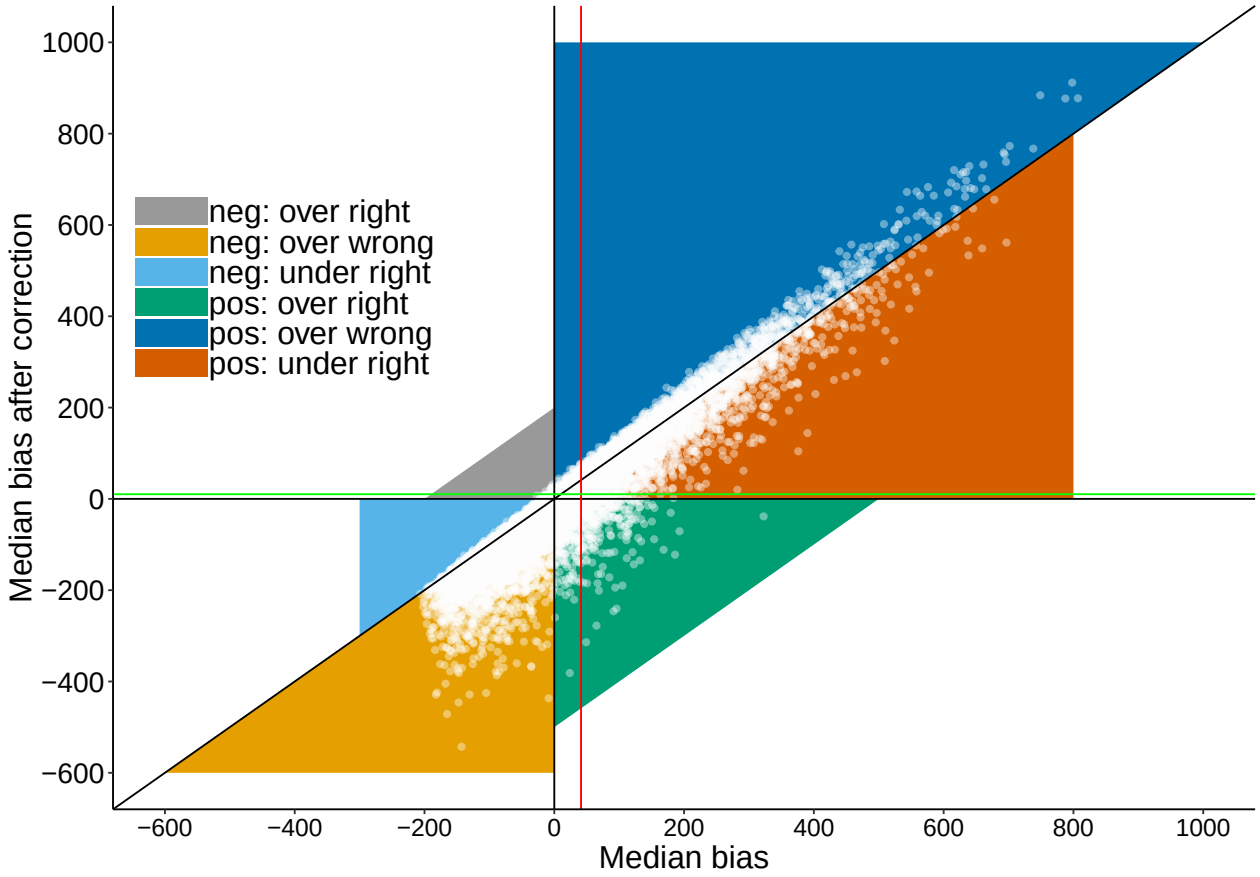



```
  # }
# }
```

Figure with markup:

```
# load('./data/sim_miller1988.RData')
P <- 1 # 1=most skewed distribution; 12=least
N <- 1 # 1=smallest sample size; 10=largest
```

```r
# define triangles
df.triangle <- tibble(x=c(-600,0,0, 0,0,500, 0,800,800, 0,0,1000, -200,0,0, -300,0,-300),
                      y=c(-600,-600,0, -500,0,0, 0,0,800, 0,1000,1000, 0,0,200, 0,0,-300),
                      g=factor(c(rep("neg: over wrong",3),
                                 rep("pos: over right",3),
                                 rep("pos: under right",3),
                                 rep("pos: over wrong",3),
                                 rep("neg: over right",3),
                                 rep("neg: under right",3)))
                      )

# define colour palette
cbPalette <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#0072B2", "#D55E00")

# for(N in 1:length(nvec)){
  # for(P in 1:nP){

    df <- tibble(`ORI`=sim.md[,P,N]-pop.md[P],
                 `BC`=sim.md.bc[,P,N]-pop.md[P])

    # make plot
    # p <- ggplot(df, aes(x=ORI, y=BC)) + theme_classic() +
      p <- ggplot(df.triangle, aes(x=x, y=y)) + theme_classic() +
        geom_polygon(aes(group=g, fill=g), alpha=1) +
        scale_fill_manual(values=cbPalette) +
      geom_point(data=df, aes(x=ORI, y=BC), colour="white", alpha=0.4) +
      geom_abline(intercept=0, slope=1, colour="black") +
      geom_vline(xintercept = 0, colour="black") +
      geom_hline(yintercept = 0, colour="black") +
      geom_vline(xintercept = mean(df$ORI), colour="red") + # bias
      geom_hline(yintercept = mean(df$BC), colour="green") + # bias after BBC
      scale_x_continuous(breaks=seq(-600,1000,200)) +
      scale_y_continuous(breaks=seq(-600,1000,200)) +
      coord_cartesian(xlim=c(-600,1000), ylim=c(-600,1000)) +
      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 = c(0.15,0.65),
        legend.text=element_text(size=18),
        legend.title=element_blank()) + #element_text(size=18)) +
      labs(x = "Median bias", y = "Median bias after correction") +
      guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
    # ggtitle("Bias correction: direction")
    p
```




```
  # }
# }
```

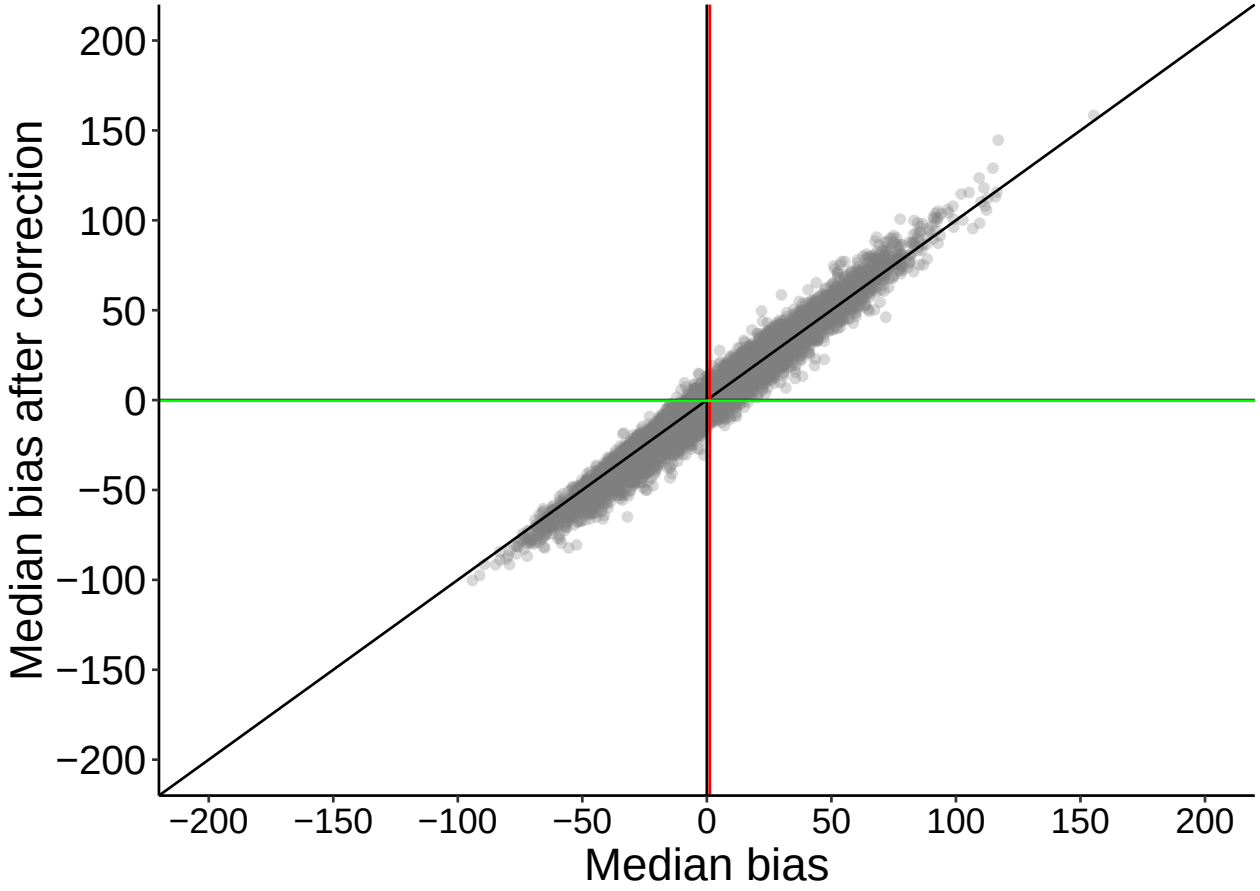
If the original bias is negative, after correction, the median tends to be even more negative, so over corrected in the wrong direction (lower left triangle).

If the original bias is positive, after correction, the median is either:
- over corrected in the right direction (lower right triangle)
- under corrected in the right direction (middle right triangle)
- over corrected in the wrong direction (upper right triangle)

This pattern remains, although attenuated, if we consider the largest sample size:

```
# load('./data/sim_miller1988.RData')
P <- 1 # 1=most skewed distribution; 12=least
N <- 10 # 1=smallest sample size; 10=largest

df <- tibble(`ORI`=sim.md[,P,N]-pop.md[P],
             `BC`=sim.md.bc[,P,N]-pop.md[P])

# for(N in 1:length(nvec)){
# make plot
p <- ggplot(df, aes(x=ORI, y=BC)) + theme_classic() +
  # geom_line(aes(colour = Skewness), size = 1) +
  geom_point(colour="grey50", alpha=0.3) +
  geom_abline(intercept=0, slope=1, colour="black") +
  geom_vline(xintercept = 0, colour="black") +
  geom_hline(yintercept = 0, colour="black") +
```

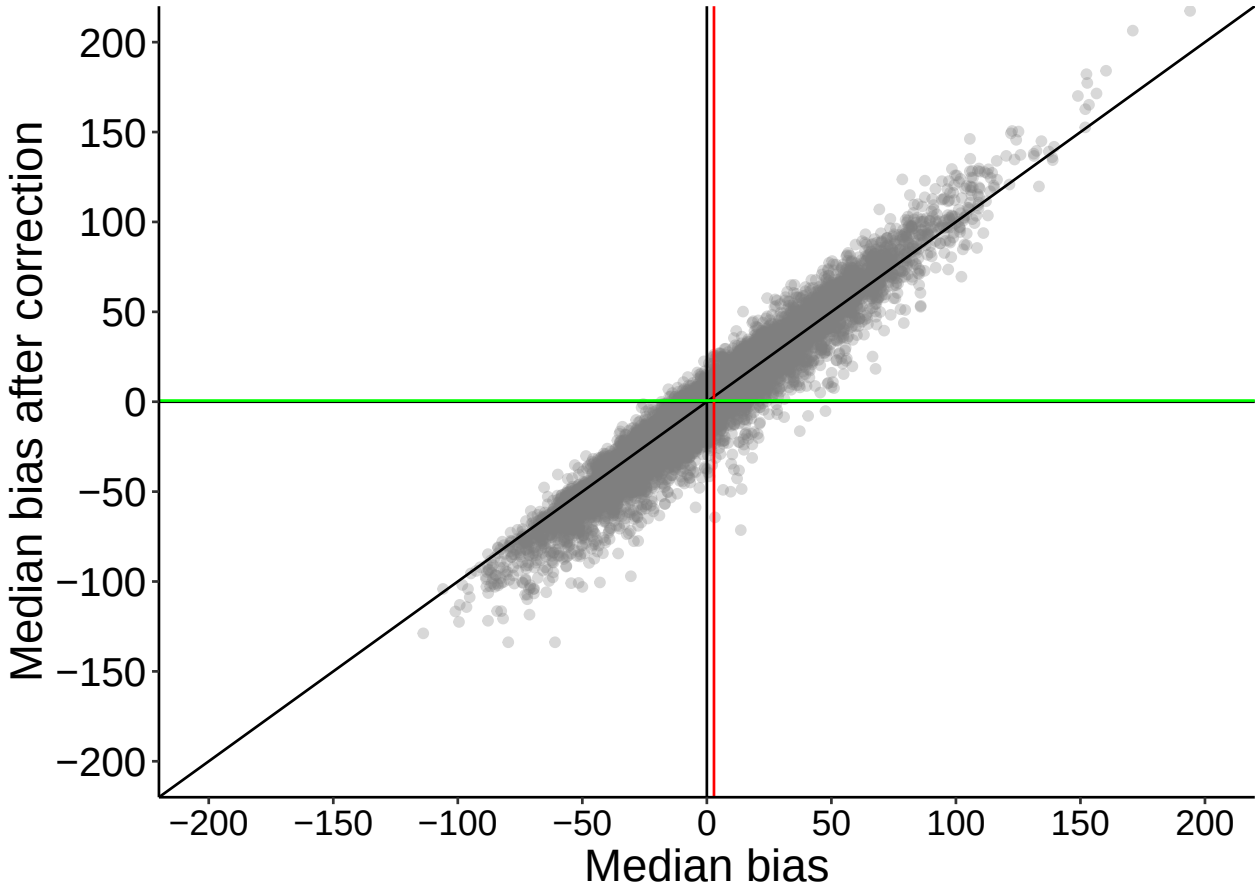
```
  geom_vline(xintercept = mean(df$ORI), colour="red") + # bias
  geom_hline(yintercept = mean(df$BC), colour="green") + # bias after BBC
  # scale_colour_viridis_d() +
  scale_x_continuous(breaks=seq(-200,200,50)) +
  scale_y_continuous(breaks=seq(-200,200,50)) +
  coord_cartesian(xlim=c(-200,200), ylim=c(-200,200)) +
  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 = "blank",#c(0.85,0.65),
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Median bias", y = "Median bias after correction") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
  # ggtitle("Bias correction: direction")
p
```




```
# p.bc.dir <- p
# save figure
# ggsave(filename=paste0('figure_miller_bc_check_P',P,'_N',nvec[N],'.pdf'),width=10,height=7)
# }
```

Or if we consider the least skewed distribution.

```r
# load('./data/sim_miller1988.RData')
P <- 12 # 1=most skewed distribution; 12=least
N <- 1 # 1=smallest sample size; 10=largest

df <- tibble(`ORI`=sim.md[,P,N]-pop.md[P],
             `BC`=sim.md.bc[,P,N]-pop.md[P])

# for(N in 1:length(nvec)){
# make plot
p <- ggplot(df, aes(x=ORI, y=BC)) + theme_classic() +
  # geom_line(aes(colour = Skewness), size = 1) +
  geom_point(colour="grey50", alpha=0.3) +
  geom_abline(intercept=0, slope=1, colour="black") +
  geom_vline(xintercept = 0, colour="black") +
  geom_hline(yintercept = 0, colour="black") +
  geom_vline(xintercept = mean(df$ORI), colour="red") + # bias
  geom_hline(yintercept = mean(df$BC), colour="green") + # bias after BBC
  # scale_colour_viridis_d() +
 scale_x_continuous(breaks=seq(-200,200,50)) +
  scale_y_continuous(breaks=seq(-200,200,50)) +
  coord_cartesian(xlim=c(-200,200), ylim=c(-200,200)) +
  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 = "blank",#c(0.85,0.65),
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Median bias", y = "Median bias after correction") +
  guides(colour = guide_legend(override.aes = list(size=3))) # make thicker legend lines
p
```



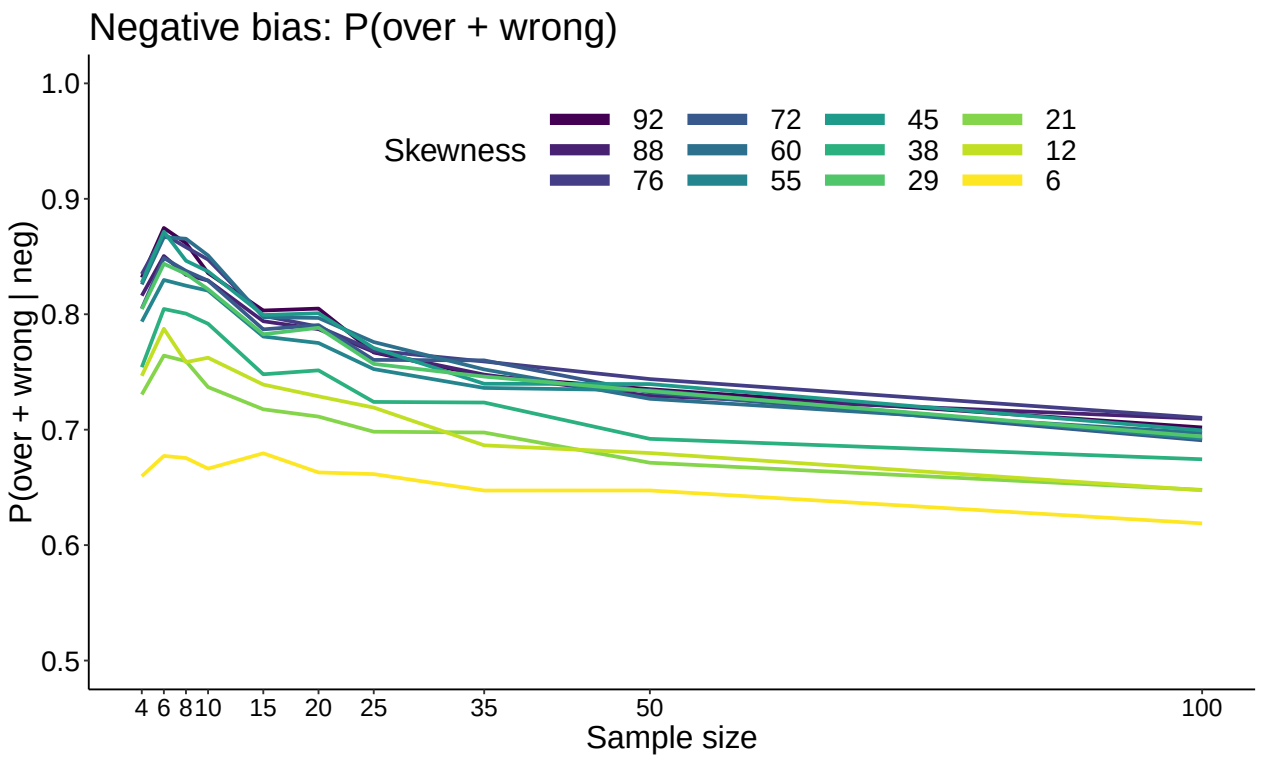

```
# }
```

We can look at the different patterns as a function of sample size and skewness.

Illustrate results: neg over wrong

```
m <- matrix(rep(pop.md, length(nvec)), nrow=nP)
POP <- aperm(replicate(nsim, m, simplify = "array"), c(3,1,2)) # 3D array
BIAS <- sim.md - POP
bc.res <- apply((sign(BIAS) == -1) * (sim.md.bc < sim.md ), c(2,3), sum) /
            apply((sign(BIAS) == -1), c(2,3), sum)

df <- tibble(`Bias`=as.vector(bc.res),
             `Size`=rep(nvec,each=nP),
             `Skewness`=rep(round(pop.m - pop.md),length(nvec)))

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

# compute expected proportion in ideal situation
set.seed(21)
BC=runif(nsim, min=-50, max=50)*rnorm(nsim)
N <- 1 # smallest n
P <- 1 # most skewed
BIAS=sim.md[,P,N]-pop.md[P]
```

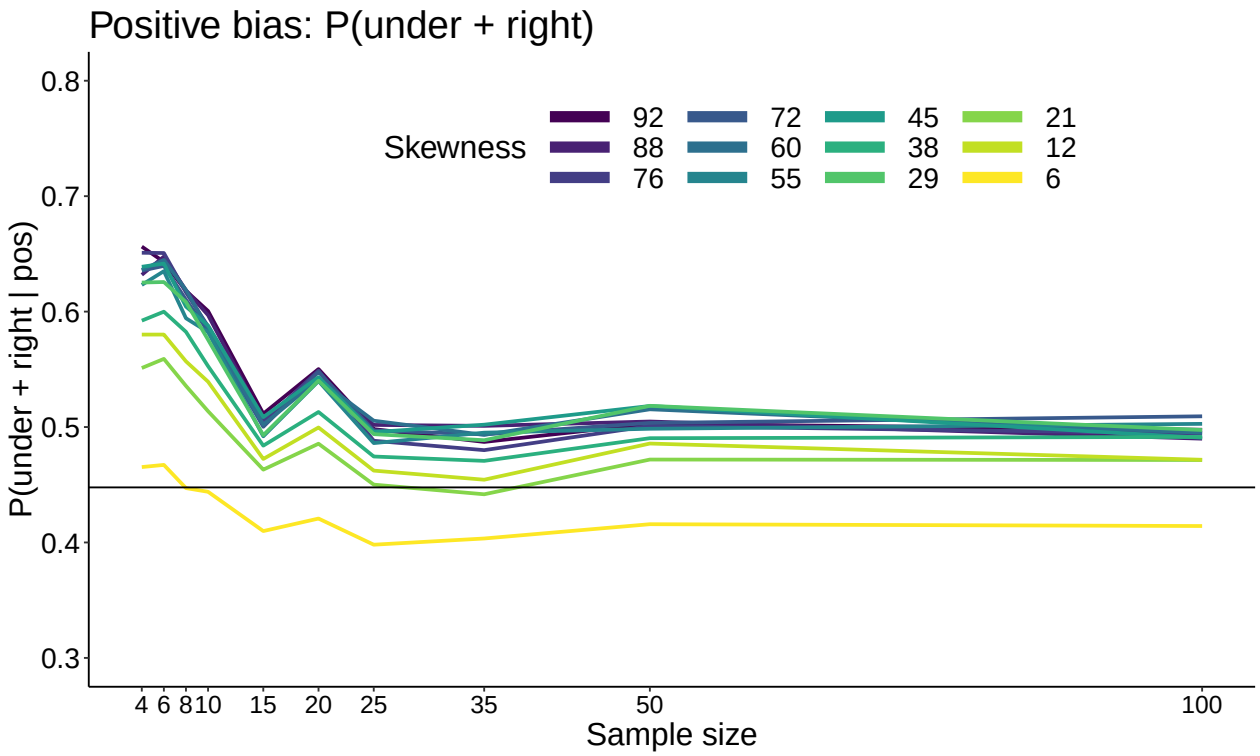
```
REF= sum((sign(BIAS) == -1) * (BC < BIAS)) /
            sum(sign(BIAS) == -1)

# make plot
p <- ggplot(df, aes(x=Size, y=Bias)) + theme_classic() +
  geom_line(aes(colour = Skewness), size = 1) +
  geom_hline(yintercept=REF, colour="black") +
  scale_colour_viridis_d() +
  scale_x_continuous(breaks=nvec) +
  scale_y_continuous(breaks=seq(0,1,.1)) +
  coord_cartesian(ylim=c(0.5,1)) +
  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 = c(0.55,0.85),
        legend.direction = "horizontal",
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Sample size", y = "P(over + wrong | neg)") +
  guides(colour = guide_legend(override.aes = list(size=3))) + # make thicker legend lines
  ggtitle("Negative bias: P(over + wrong)")
p
```



In the ideal situation illustrated previously, the expected proportion of over correction in the wrong direction, given an original negative bias, is 7.2%. So here, we clearly have an overrepresentation of these cases. When the original bias is negative, in most cases the bootstrap is unable to correct in the right direction. The situation gets worse with increasing skewness and smaller sample sizes.

Illustrate results: pos under right

```
m <- matrix(rep(pop.md, length(nvec)), nrow=nP)
POP <- aperm(replicate(nsim, m, simplify = "array"), c(3,1,2)) # 3D array
BIAS <- sim.md - POP
bc.res <- apply((sign(BIAS) == 1) * (sim.md.bc < sim.md) * (sign(sim.md.bc) == 1), c(2,3), sum) /
            apply((sign(BIAS) == 1), c(2,3), sum)

df <- tibble(`Bias`=as.vector(bc.res),
             `Size`=rep(nvec,each=nP),
             `Skewness`=rep(round(pop.m - pop.md),length(nvec)))

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

# compute expected proportion in ideal situation
set.seed(21)
BC=runif(nsim, min=-50, max=50)*rnorm(nsim)
N <- 1 # smallest n
P <- 1 # most skewed
BIAS=sim.md[,P,N]-pop.md[P]
REF= sum((sign(BIAS) == 1) * (BC < BIAS) * (sign(BC)==1)) /
            sum(sign(BIAS) == 1)

# make plot
p <- ggplot(df, aes(x=Size, y=Bias)) + theme_classic() +
  geom_line(aes(colour = Skewness), size = 1) +
  geom_hline(yintercept=REF, colour="black") +
  scale_colour_viridis_d() +
  scale_x_continuous(breaks=nvec) +
  scale_y_continuous(breaks=seq(0,1,.1)) +
  coord_cartesian(ylim=c(0.3,0.8)) +
  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 = c(0.55,0.85),
        legend.direction = "horizontal",
        legend.text=element_text(size=16),
        legend.title=element_text(size=18)) +
  labs(x = "Sample size", y = "P(under + right | pos)") +
  guides(colour = guide_legend(override.aes = list(size=3))) + # make thicker legend lines
  ggtitle("Positive bias: P(under + right)")
p
```



In the ideal situation illustrated previously, the expected proportion of under correction in the right direction, given an original positive bias, is 44.8%. So here, we have an overrepresentation of these cases. When the original bias is positive, in too many cases the bootstrap corrects in the right direction, but it undercorrects. The situation gets worse with increasing skewness and smaller sample sizes.