Demo: bias & bootstrap bias correction

Guillaume A. Rousselet

Contents

——

```r
# dependencies
library(ggplot2)
library(tibble)
library(tidyr)
library(viridis)
library(retimes)
source("akerd.txt")
```

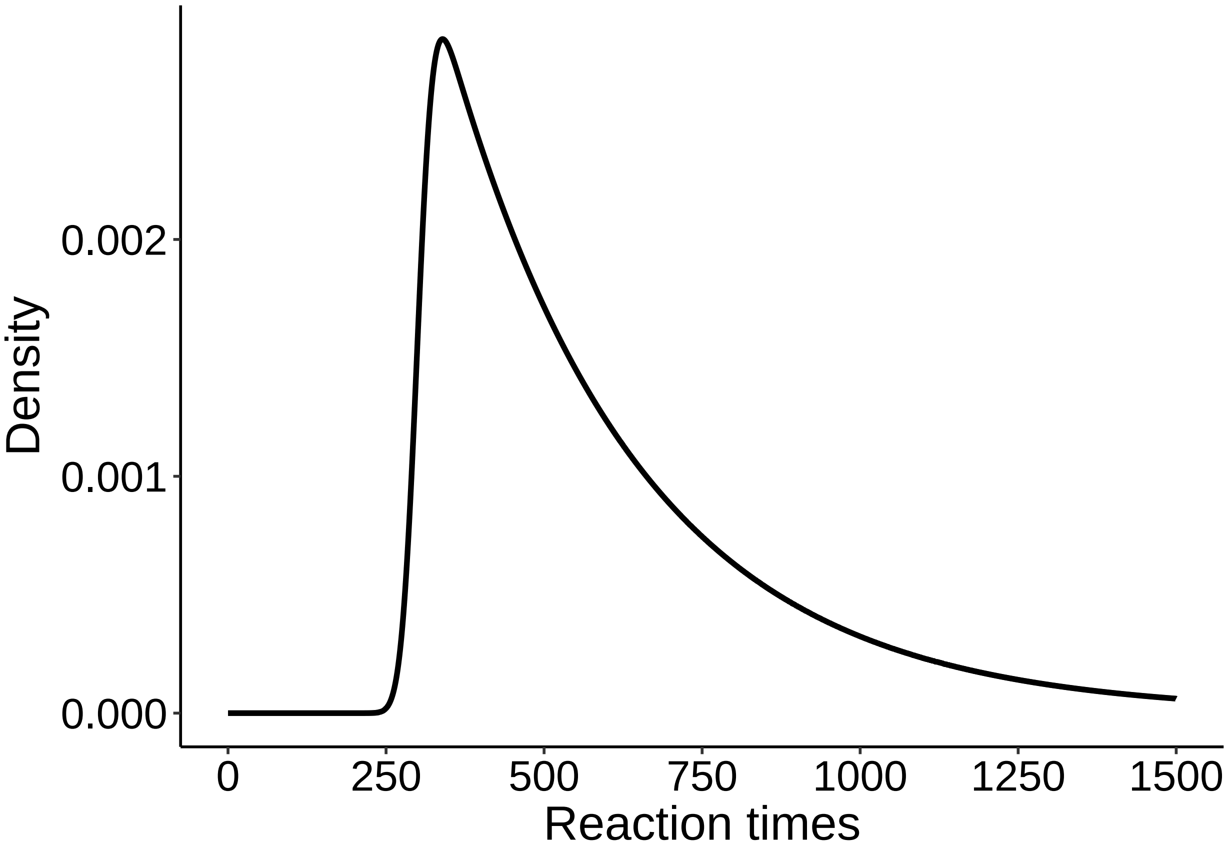
Describe population

We use an ex-Gaussian distribution because it looks a bit like a reaction time distribution, but you can call the x-axis whatever you want, there are plenty of skewed distributions in the nature.

Population density function

```r
x <- seq.int(0,1500,0.1)
mu <- 300
sigma <- 20
tau <- 300
mdist <- dexgauss(x, mu = mu, sigma = sigma, tau = tau)

# make data frame
df <- tibble(x=x, y=mdist)

# make plot
```

```
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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", y = "Density")
p
```



Population parameters

We create a population, then quantify several parameters, which we'll then try to estimate by sampling with replacement from the population. In other words, we'll perform experiments to see how close the sample estimates get to the true population values.

```
set.seed(4)
n <- 1000000
pop <- rexgauss(n, mu = mu, sigma = sigma, tau = tau)
pop.m <- mean(pop)
pop.md <- sort(pop)[round(length(pop)*0.5)] # median(pop)
pop.sd <- sqrt(sum((pop-pop.m)^2)/length(pop)) # population standard deviation
pop.es <- pop.m / pop.sd # population effect size
round(pop.m) # population mean
```

```
## [1] 600
```

```
round(pop.md) # population median
```

```
## [1] 509
```

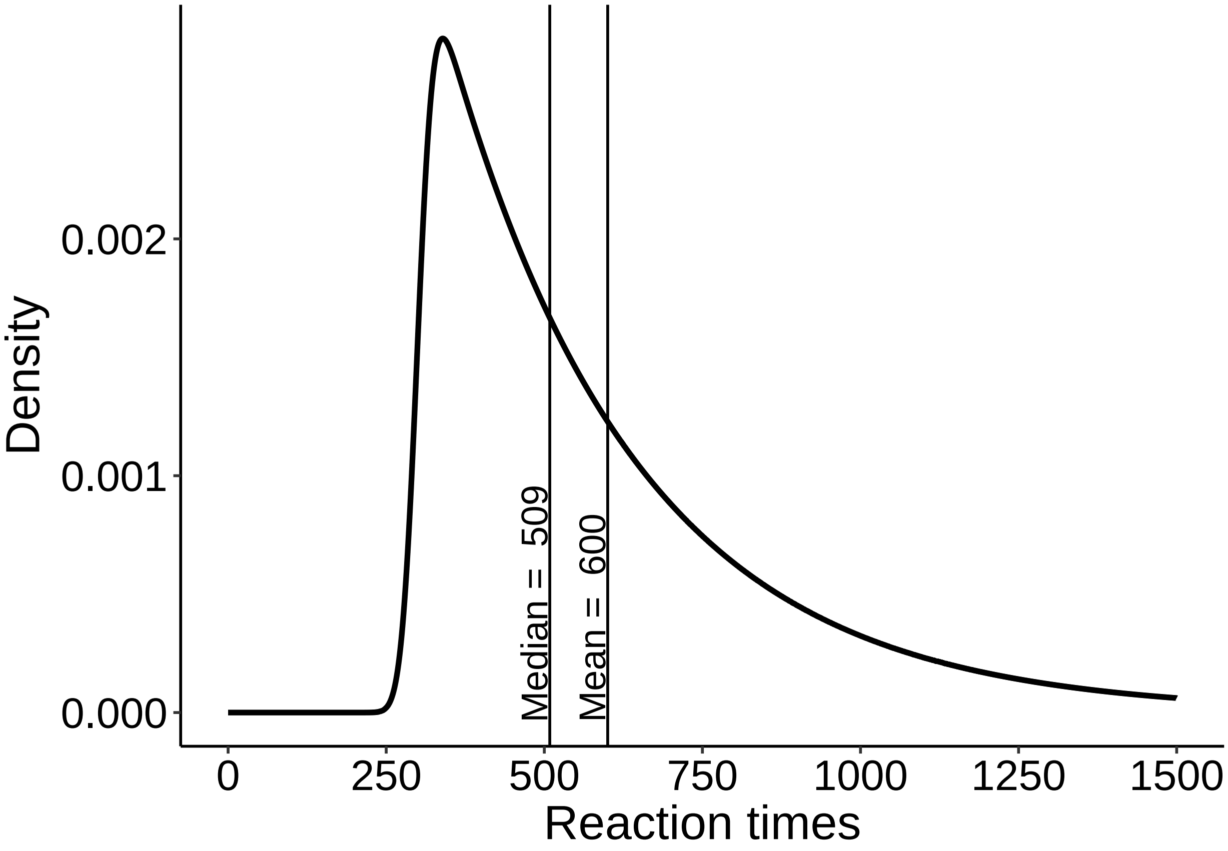
```
round(pop.m - pop.md) # measure of skewness
```

```
## [1] 92
```

The population mean is 600; the population median is 509. We can define a non-parametric measure of skewness as the difference between the mean and the median, which is 92. The population standard deviation (SD) is 301. The population standardised effect size (ES) is 2.

Density function + population mean + median

```
# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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", y = "Density") +
  geom_vline(xintercept=pop.m, linetype=1) +
  geom_vline(xintercept=pop.md) +
  annotate("text", x = pop.m-25, y = 0.0004, label = paste("Mean = ",round(pop.m)), angle=90, size=5) +
  annotate("text", x = pop.md-25, y = 0.00046, label = paste("Median = ",round(pop.md)), angle=90, size=
p
```

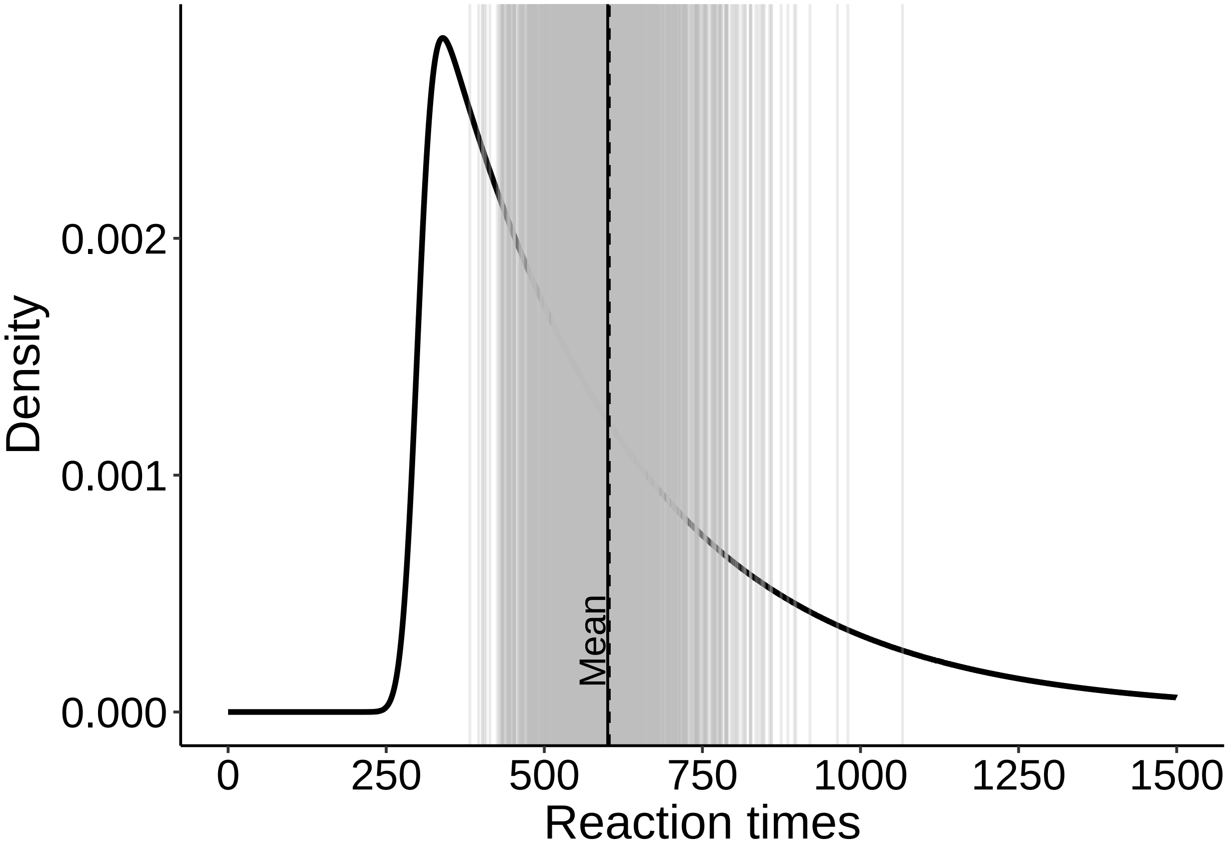


```
ggsave(filename=paste0('figure.kde.pop.m.md.jpg'),width=8,height=5) #path=pathname
```

Population + mean + nsim samples + sample mean

Simulation: 1000 samples (experiments) of 10 observations. The means the 1000 samples are shown by grey vertical lines: there is of course a lot variability, such that any experiment (sample), can be very far from the population mean. The population mean and the average sample mean are marked by vertical black lines, and are very similar. The population mean is shown using a continuous line; the average sample mean is shown using a dashed line. Increasing the number of samples or the sample size, or both, will lead to the sample mean to converge to the population mean.

```
nsim <- 1000 # number of samples
n <- 10 # sample size
set.seed(44)
sample.m <- apply(matrix(rexgauss(n*nsim, mu = mu, sigma = sigma, tau = tau), nrow=nsim), 1, mean)

# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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", y = "Density") +
  geom_vline(xintercept=sample.m, linetype=1, colour="grey", alpha=0.3) +
  geom_vline(xintercept=pop.m, linetype=1) +
  geom_vline(xintercept=mean(sample.m), linetype=2) +
  annotate("text", x = pop.m-25, y = 0.0003, label = "Mean", angle=90, size=5)
  # annotate("text", x = mean(sample.m)-23, y = 0.0003, label = "Sample mean", angle=90, size=5)
  # annotate("text", x = pop.md-23, y = 0.00035, label = "Median", angle=90, size=5)
p
```



```
print(paste("Bias = ", round(mean(sample.m) - pop.m, digits = 1)))
```

```
## [1] "Bias =  2.5"
```

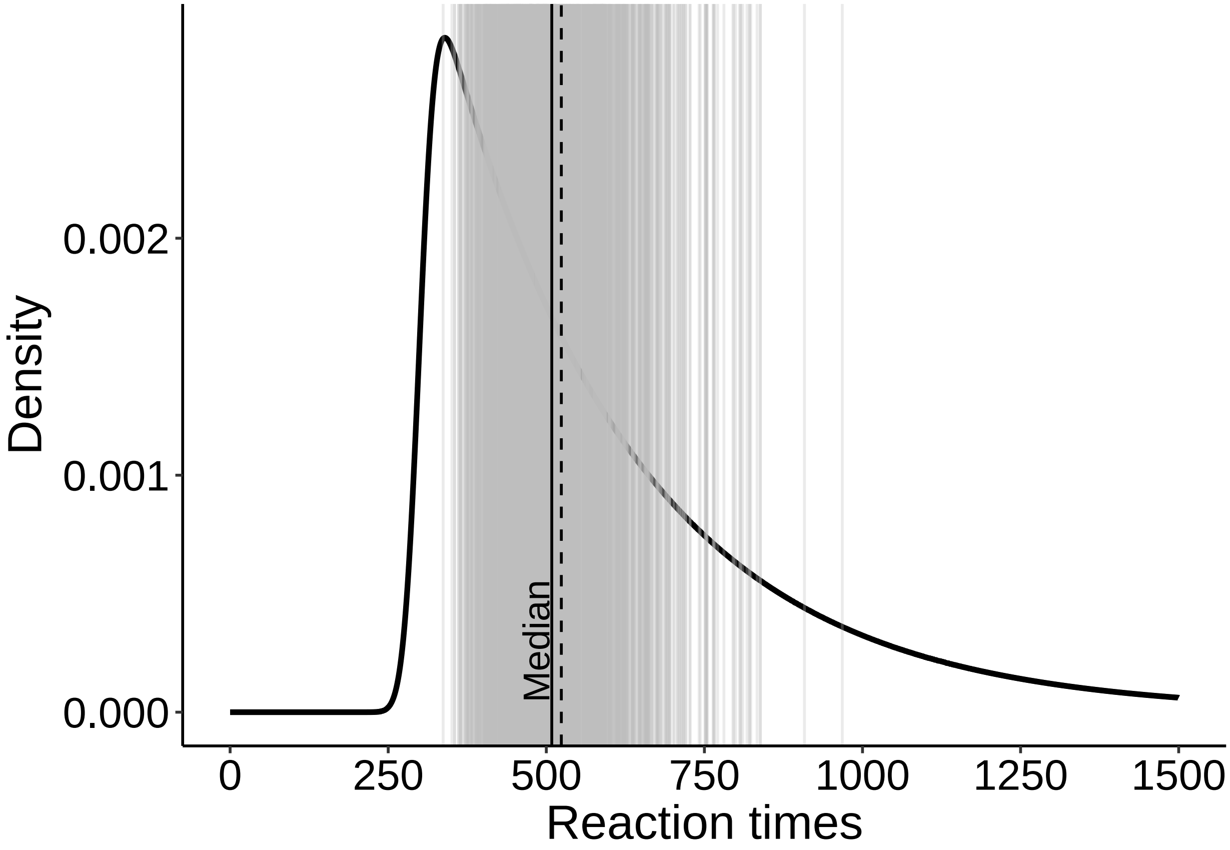
```
ggsave(filename=paste0('figure.kde.1000m.jpg'),width=8,height=5) #path=pathname
```

Population + median + nsim samples + sample median

With low sample sizes, the sample median tends to overestimate the population median: it provides a biased estimation of the true median.

```
nsim <- 1000 # number of samples
n <- 10 # sample size
set.seed(44)
sample.md <- apply(matrix(rexgauss(n*nsim, mu = mu, sigma = sigma, tau = tau), nrow=nsim), 1, median)
```

```
# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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", y = "Density") +
  geom_vline(xintercept=sample.md, linetype=1, colour="grey", alpha=0.3) +
  geom_vline(xintercept=pop.md, linetype=1) +
  geom_vline(xintercept=mean(sample.md), linetype=2) +
  annotate("text", x = pop.md-25, y = 0.0003, label = "Median", angle=90, size=5)
  # annotate("text", x = mean(sample.m)-23, y = 0.0003, label = "Sample mean", angle=90, size=5)
  # annotate("text", x = pop.md-23, y = 0.00035, label = "Median", angle=90, size=5)
p
```



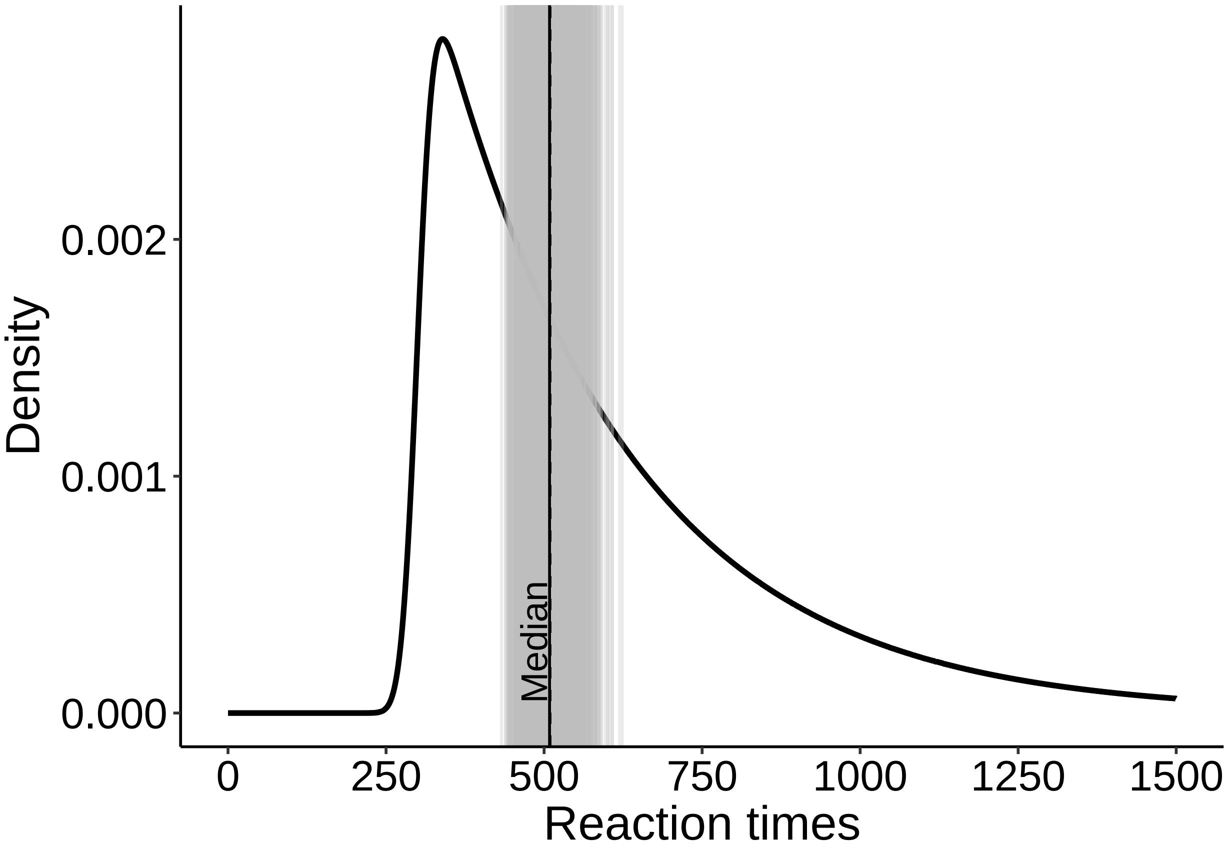
```
print(paste("Bias = ", round(mean(sample.md) - pop.md, digits = 1)))
```

```
## [1] "Bias =  15.1"
```

```
ggsave(filename=paste0('figure.kde.1000md.jpg'),width=8,height=5) #path=pathname
```

This bias disappears with larger sample sizes. Larger samples also increase the accuracy of our estimation, as illustrated by the reduced dispersion of sample medians.

```r
nsim <- 1000 # number of samples
n <- 100 # sample size
set.seed(44)
sample.md <- apply(matrix(rexgauss(n*nsim, mu = mu, sigma = sigma, tau = tau), nrow=nsim), 1, median)

# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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", y = "Density") +
  geom_vline(xintercept=sample.md, linetype=1, colour="grey", alpha=0.3) +
  geom_vline(xintercept=pop.md, linetype=1) +
  geom_vline(xintercept=mean(sample.md), linetype=2) +
  annotate("text", x = pop.md-25, y = 0.0003, label = "Median", angle=90, size=5)
  # annotate("text", x = mean(sample.m)-23, y = 0.0003, label = "Sample mean", angle=90, size=5)
  # annotate("text", x = pop.md-23, y = 0.00035, label = "Median", angle=90, size=5)
p
```



```
print(paste("Bias = ", round(mean(sample.md) - pop.md, digits = 1)))
```

```
## [1] "Bias =  0.7"
```

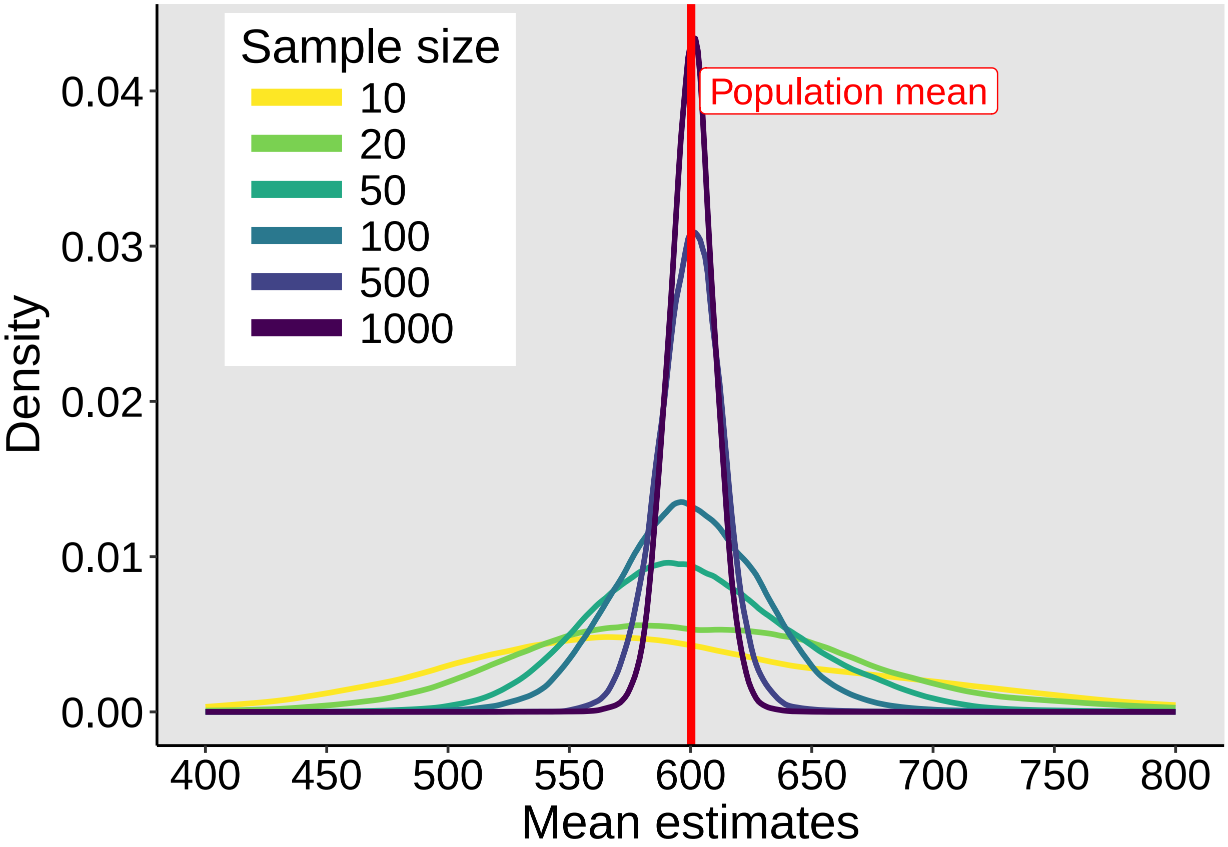
```
ggsave(filename=paste0('figure.kde.1000md_n100.jpg'),width=8,height=5) #path=pathname
```

The sample mean is not biased

Using our population, let's do 1000 experiments, in which we take different numbers of samples: * 1000 * 500 * 100 * 50 * 20 * 10 Then, we can illustrate how close all these experiments get to the true (population) value.

```
set.seed(21)
x <- seq(400, 800, 1)

nsim <- 1000 # 1000 simulations / experiments
nvec <- c(10, 20, 50, 100, 500, 1000) # different sample sizes
sim.res <- matrix(0, nrow=nsim, ncol=length(nvec)) # declare matrix of results
sim.kde <- matrix(0, nrow=length(x), ncol=length(nvec)) # declare matrix of results

# compute 1000 means from different sample sizes and save results
# We use `sample()` to sample with replacement from our population `pop`
for(S in 1:length(nvec)){
  sim.res[,S] <- apply(matrix(sample(pop, nsim*nvec[S], replace = TRUE), nrow = 1000), 1, mean)
}

# save kernel density estimates
for(S in 1:length(nvec)){
  sim.kde[,S] <- akerd(sim.res[,S], pts = x, pyhat = TRUE, plotit = FALSE)
}

df <- tibble(sd = rep(x, length(nvec)),
             kde = as.vector(sim.kde),
             `Sample size` = factor(rep(nvec, each = length(x))))

# make plot
p <- ggplot(df, aes(x=sd, y=kde)) + theme_classic() +
          geom_line(aes(colour = `Sample size`), size=1) +
       scale_color_viridis(discrete = TRUE, direction=-1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                legend.key.width = unit(1.5,"cm"),
                legend.position = c(0.2,0.75),
                legend.text=element_text(size=16),
                legend.title=element_text(size=18),
                 panel.background = element_rect(fill="grey90")) +
  scale_x_continuous(breaks = seq(400, 800, 50)) +
  labs(x = "Mean estimates", y = "Density") +
  guides(colour = guide_legend(override.aes = list(size=3))) +
  geom_vline(xintercept=pop.m, linetype=1, colour = "red", size = 1.5) +
  geom_label(data=tibble(sd=pop.m+65, kde=0.04),
             label = "Population mean", angle = 90, colour="red", size=5)
  # annotate("text", x = pop.sd-10, y = 0.006, label = "Population SD", angle=90, size=5, colour="red")
```

```
p
```



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

Our estimates of the population mean converge towards the true population value with increasing sample size. But for small samples, the sample mean (yellow, n=10) tends to underestimate the true population value. The median of the sampling distribution of the mean in the previous figure is 584.5, which is 15.7 under the population value. But despite the shift of the sampling distribution with small n, the average of the 1000 simulations/experiments is very close to the population value, for all sample sizes:

```
round(pop.m, digits = 1)
```

```
## [1] 600.2
```

```
round(apply(sim.res, 2, mean), digits = 1)
```

```
## [1] 593.7 603.1 601.9 599.6 600.1 600.3
```

The approximation gets closer to the true value with more experiments.

Let's try 10,000 experiments of n=10:

```
set.seed(21)
mean(apply(matrix(sample(pop, 10000*10, replace = TRUE), nrow = 10000), 1, mean))
```

```
## [1] 600.3001
```

The result is very close to the true value. That's why we say that the sample mean is an unbiased estimator of the population mean. It remains that with small n, the typical sample mean tends to underestimate the population mean, and the variance is large.

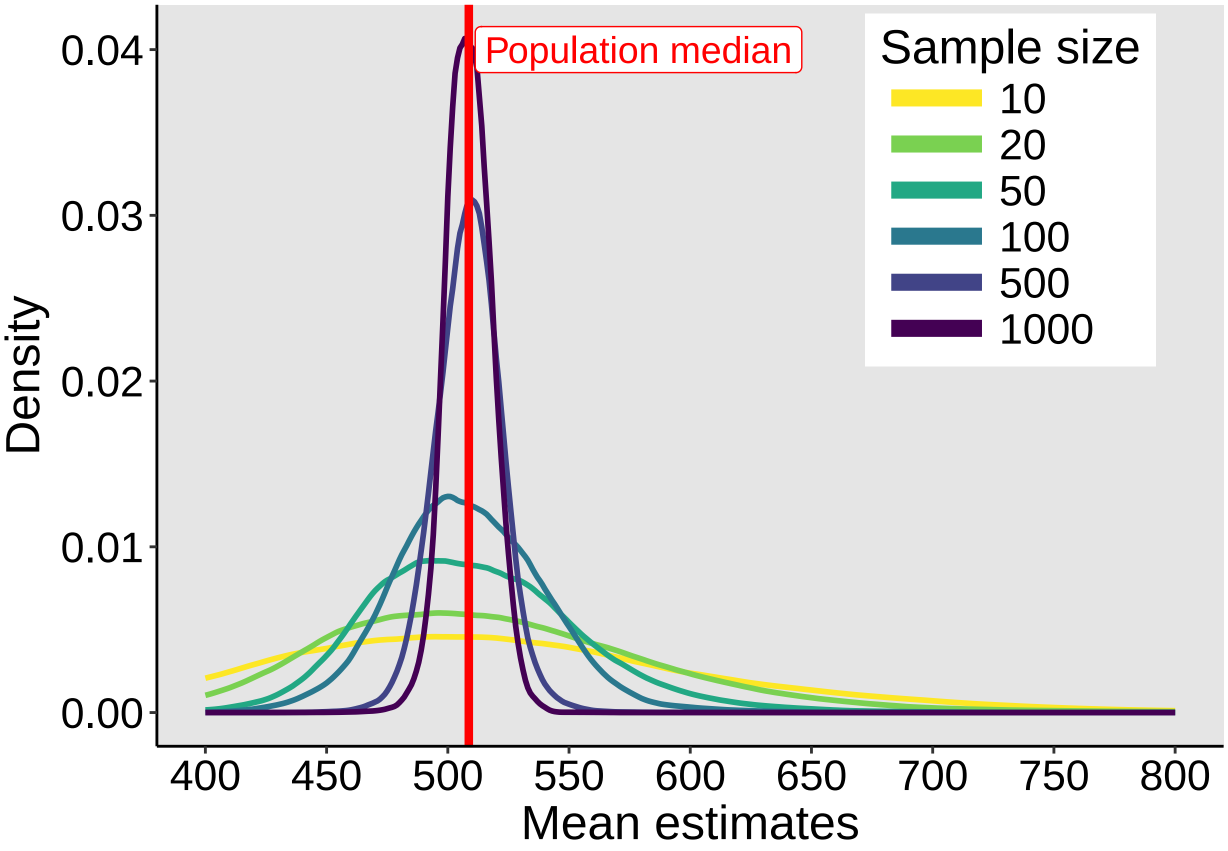
Bias of the median

The sample median is biased when n is small and we sample from skewed distributions:

Miller, J. (1988) A warning about median reaction time. J Exp Psychol Hum Percept Perform, 14, 539-543.

However, the bias can be corrected using the bootstrap. Let's first look at the sampling distribution of the median for different sample sizes.

```r
set.seed(21)
x <- seq(400, 800, 1)

nsim <- 1000 # 1000 simulations / experiments
nvec <- c(10, 20, 50, 100, 500, 1000) # different sample sizes
sim.res <- matrix(0, nrow=nsim, ncol=length(nvec)) # declare matrix of results
sim.kde <- matrix(0, nrow=length(x), ncol=length(nvec)) # declare matrix of results

# compute 1000 medians from different sample sizes and save results
# We use `sample()` to sample with replacement from our population `pop`
for(S in 1:length(nvec)){
  sim.res[,S] <- apply(matrix(sample(pop, nsim*nvec[S], replace = TRUE), nrow = 1000), 1, median)
}

# save kernel density estimates
for(S in 1:length(nvec)){
  sim.kde[,S] <- akerd(sim.res[,S], pts = x, pyhat = TRUE, plotit = FALSE)
}

df <- tibble(sd = rep(x, length(nvec)),
             kde = as.vector(sim.kde),
             `Sample size` = factor(rep(nvec, each = length(x))))

# make plot
p <- ggplot(df, aes(x=sd, y=kde)) + theme_classic() +
          geom_line(aes(colour = `Sample size`), size=1) +
       scale_color_viridis(discrete = TRUE, direction=-1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                legend.key.width = unit(1.5,"cm"),
                legend.position = c(0.8,0.75),
                legend.text=element_text(size=16),
                legend.title=element_text(size=18),
                 panel.background = element_rect(fill="grey90")) +
  scale_x_continuous(breaks = seq(400, 800, 50)) +
  labs(x = "Mean estimates", y = "Density") +
  guides(colour = guide_legend(override.aes = list(size=3))) +
  geom_vline(xintercept=pop.md, linetype=1, colour = "red", size = 1.5) +
  geom_label(data=tibble(sd=pop.md+70, kde=0.04),
             label = "Population median", angle = 90, colour="red", size=5)
  # annotate("text", x = pop.sd-10, y = 0.006, label = "Population SD", angle=90, size=5, colour="red")
p
```



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

Doesn't look too bad, but for small sample sizes, on average the sample median over-estimates the population median.

```
round(pop.md, digits = 1)
```

```
## [1] 508.7
```

```
round(apply(sim.res, 2, mean), digits = 1)
```

```
## [1] 522.1 518.4 511.5 508.9 508.7 508.7
```

Unlike what happened with the mean, the approximation does not get closer to the true value with more experiments. Let's try 10,000 experiments of n=10:

```
set.seed(21)
mean(apply(matrix(sample(pop, 10000*10, replace = TRUE), nrow = 10000), 1, median))
```

```
## [1] 523.8525
```

There is systematic shift between average sample estimates and the population value: thus the sample median is a biased estimate of the population median. Fortunately, this bias can be corrected using the bootstrap. The bootstrap bias correction is described in this book:

Efron, B. & Tibshirani Robert, J. (1993) An introduction to the bootstrap. Chapman & Hall, London

Bias estimation and correction

A simple technique to estimate and correct sampling bias is the percentile bootstrap. If we have a sample of n observations: - sample with replacement n observations from our original sample - compute the estimate - perform steps 1 and 2 nboot times - compute the mean of the nboot bootstrap estimates The difference between the estimate computed using the original sample and the mean of the bootstrap estimates is a bootstrap estimate of bias.

Let's consider one sample of 10 observations from our skewed distribution. It's median is:

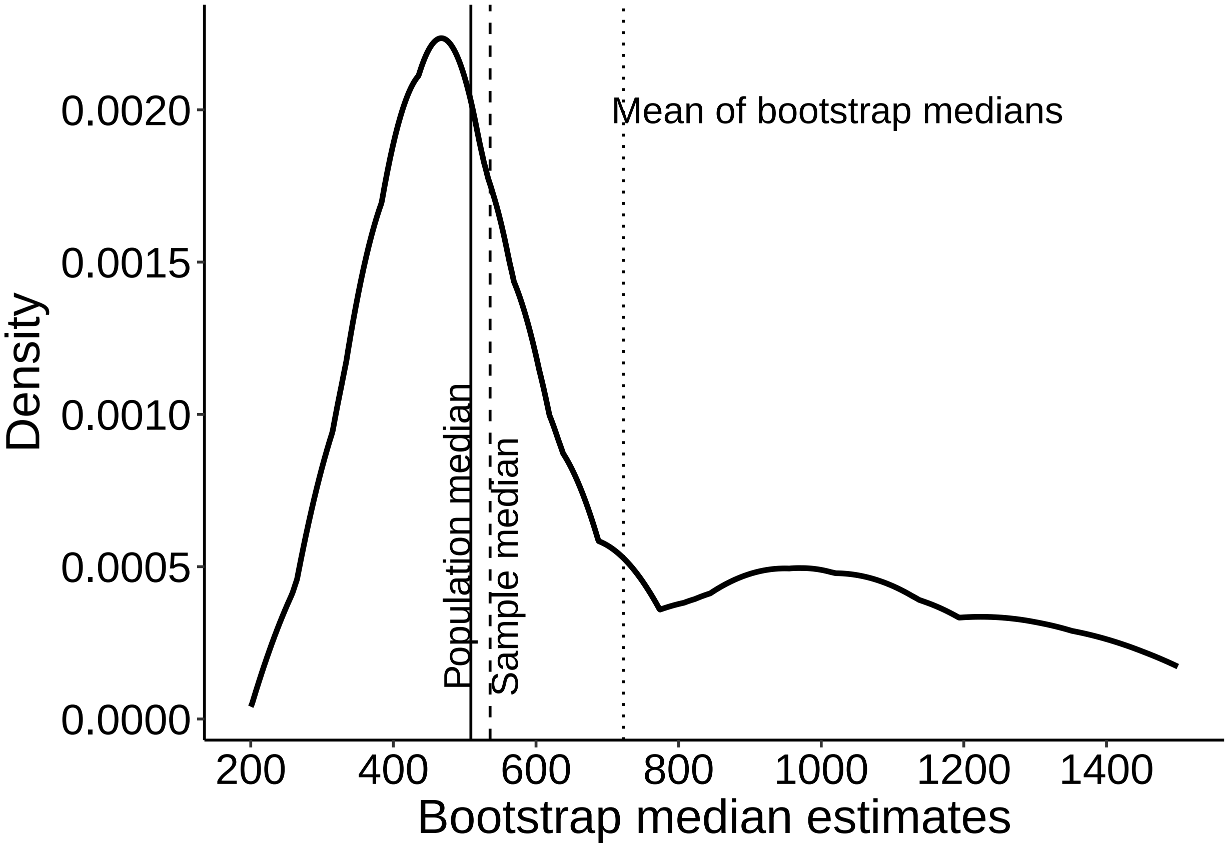
```r
set.seed(4)
samp <- sample(pop, 10, replace=TRUE)
median(samp)
```

```
## [1] 535.5856
```

The population median is 508.651647, so our sample considerably over-estimate the population value.

Next, we sample with replacement from our sample, and compute bootstrap estimates of the median. The distribution obtained is a bootstrap estimate of the sampling distribution of the median. The idea is this: if the bootstrap distribution approximates, on average, the shape of the sampling distribution of the median, then we can use the bootstrap distribution to estimate the bias and correct our sample estimate.

```r
set.seed(21)

# 1000 bootstrap median estimates
nboot <- 1000
boot.md <- apply(matrix(sample(samp, nboot*length(samp), replace = TRUE),nrow=nboot),1,median)

# make kernel density estimate
x <- seq.int(200, 1500, 1)
boot.md.kde <- akerd(boot.md, pts = x, pyhat = TRUE, plotit = FALSE)

# make data frame
df <- tibble(x=x, y=boot.md.kde)

# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
          geom_line(colour = "black", size = 1) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                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(breaks = seq(0, 1500, 200)) +
  labs(x = "Bootstrap median estimates", y = "Density") +
  geom_vline(xintercept=pop.md, linetype=1) +
  geom_vline(xintercept=median(samp), linetype=2) +
  geom_vline(xintercept=mean(boot.md), linetype=3) +
  annotate("text", x = pop.md-20, y = 0.0006, label = "Population median", angle=90, size=5) +
  annotate("text", x = median(samp)+20, y = 0.0005, label = "Sample median", angle=90, size=5) +
annotate("text", x = mean(boot.md)+300, y = 0.002, label = "Mean of bootstrap medians", angle=0, size=5
p
```



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

The mean of the bootstrap estimates is:

```
mean(boot.md)
```

```
## [1] 722.5708
```

Therefore, our estimate of bias is:

```
mean(boot.md) - median(samp)
```

```
## [1] 186.9852
```

Because we're running a simulation, we have access to the otherwise unknowable true bias:

```
median(samp) - pop.md
```

```
## [1] 26.93397
```

To correct for bias, we subtract the bootstrap bias estimate from the sample estimate:

```
median(samp) - (mean(boot.md) - median(samp))
```

```
## [1] 348.6005
```

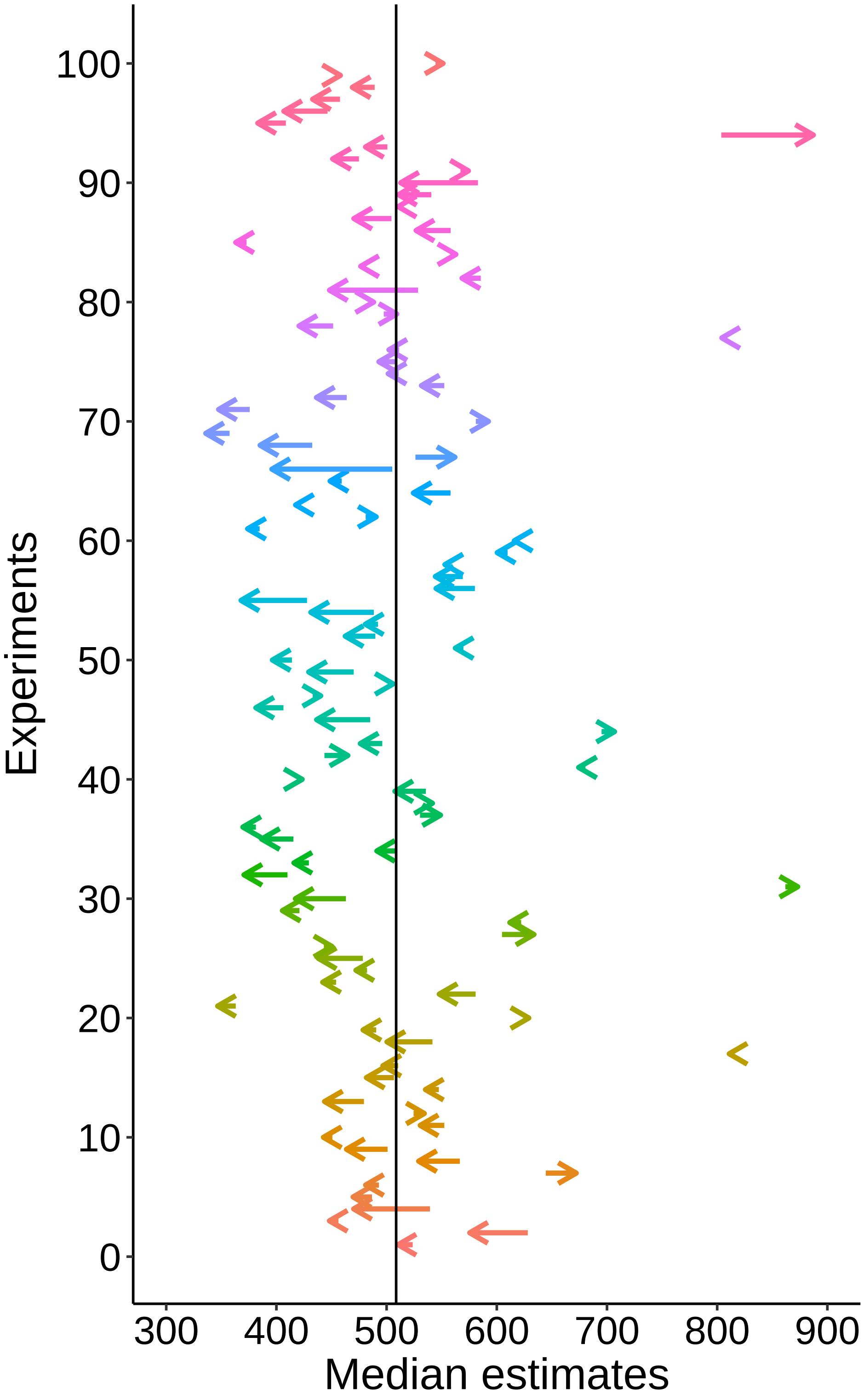
Which is the same as:

```
2*median(samp) - mean(boot.md)
```

```
## [1] 348.6005
```

So the sample bias has been reduced dramatically, clearly too much. But bias is a long run property of an estimator, so let's look at a few more examples. We take 100 samples of n = 10, and compute a bias correction for each of them. The arrows go from the sample median to the bias corrected sample median. The vertical black line shows the population median.

```r
set.seed(7)

nexp <- 100 # experiments
n <- 10 # sample size
nboot <- 200 # bootstrap samples

mat.res <- matrix(0, nrow=nexp, ncol=2)

for(E in 1:nexp){ # nexp samples
  samp <- sample(pop, n, replace=TRUE)
  mat.res[E,1] <- median(samp) # x = sample median
  # bootstrap
  boot.md <- apply(matrix(sample(samp, nboot*n, replace = TRUE),nrow=nboot),1,median)
  mat.res[E,2] <- 2*median(samp) - mean(boot.md) # xend = bias corrected sample median
}

# make data frames
df <- tibble(x=c(mat.res[,1], mat.res[,2]),
             y=rep(seq(1:nexp),2),
             exp=factor(rep(seq(1,nexp),2)))

df2 <- tibble(x=mat.res[,1],
             xend=mat.res[,2],
             y=seq(1,nexp),
             yend=seq(1,nexp),
             exp=factor(seq(1,nexp)))

# make plot
p <- ggplot(df, aes(x, y)) + theme_classic() +
     # geom_point(size = 2, aes(colour = exp)) +
          geom_segment(data=df2, aes(x=x, xend=xend, y=y, yend=yend, colour = exp),
            size = 1, arrow = arrow(length = unit(0.3, "cm"))) +
          theme(axis.title.x = element_text(size = 18),
                axis.text = element_text(size = 16, colour = "black"),
                axis.title.y = element_text(size = 18),
                legend.text = element_text(size = 16),
                legend.title = element_text(size = 18),
                legend.position = "none") +
                # panel.background = element_rect(fill="grey90")) +
  coord_cartesian(xlim = c(300, 900)) +
  scale_x_continuous(breaks = seq(300, 900, 100)) +
  scale_y_continuous(breaks = seq(0, 100, 10)) +
  labs(x = "Median estimates", y = "Experiments") +
  geom_vline(xintercept=pop.md, linetype=1)
p
```



```
# save figure
ggsave(filename='figure_mdbc_examples_n10.jpg',width=7,height=10) #path=pathname
```

```
print(paste("population median = ",round(pop.md, digits = 1)))
```

```
## [1] "population median =  508.7"
```

```r
print(paste("average sample median = ",round(mean(mat.res[,1]), digits = 1)))
```

```
## [1] "average sample median =  515.1"
```

```r
print(paste("average bias corrected sample median = ",round(mean(mat.res[,2]), digits = 1)))
```

```
## [1] "average bias corrected sample median =  498.8"
```

So across these 100 experiments, the bias correction was too strong.

What happens if we perform 1000 experiments, each with n=10, and compute a bias correction for each one? Now the average of the bias corrected median estimates is much closer to the true median.

```r
set.seed(21)

print(paste("population median = ",round(pop.md, digits = 1)))
```

```
## [1] "population median =  508.7"
```

```r
# collect data from nexp experiments
nexp <- 1000
n <- 10 # sample size
mc.data <-matrix(sample(pop, nexp*n, replace = TRUE),nrow=nexp)
# median for each experiment
mc.md <- apply(mc.data, 1, median)
# average median across experiments
print(paste("average sample median = ",round(mean(mc.md), digits = 1)))
```

```
## [1] "average sample median =  522.1"
```

```r
# Bootstrap bias correction
nboot <- 200
mc.md.bc <- vector(mode="numeric", length=nexp) # declare matrix of bias corrected estimates
for(E in 1:nexp){ # for each experiment
  # mean of nboot median bootstrap estimates
boot.md <- mean(apply(matrix(sample(mc.data[E,], nboot*n, replace = TRUE),nrow=nboot), 1, median))
mc.md.bc[E] <- 2*mc.md[E] - boot.md
}

# mean of bias corrested estimates
print(paste("average bias corrected sample median = ",round(mean(mc.md.bc), digits = 1)))
```

```
## [1] "average bias corrected sample median =  508.6"
```

It works very well!

But that’s not always the case: it depends on the estimator and on the amount of skewness.