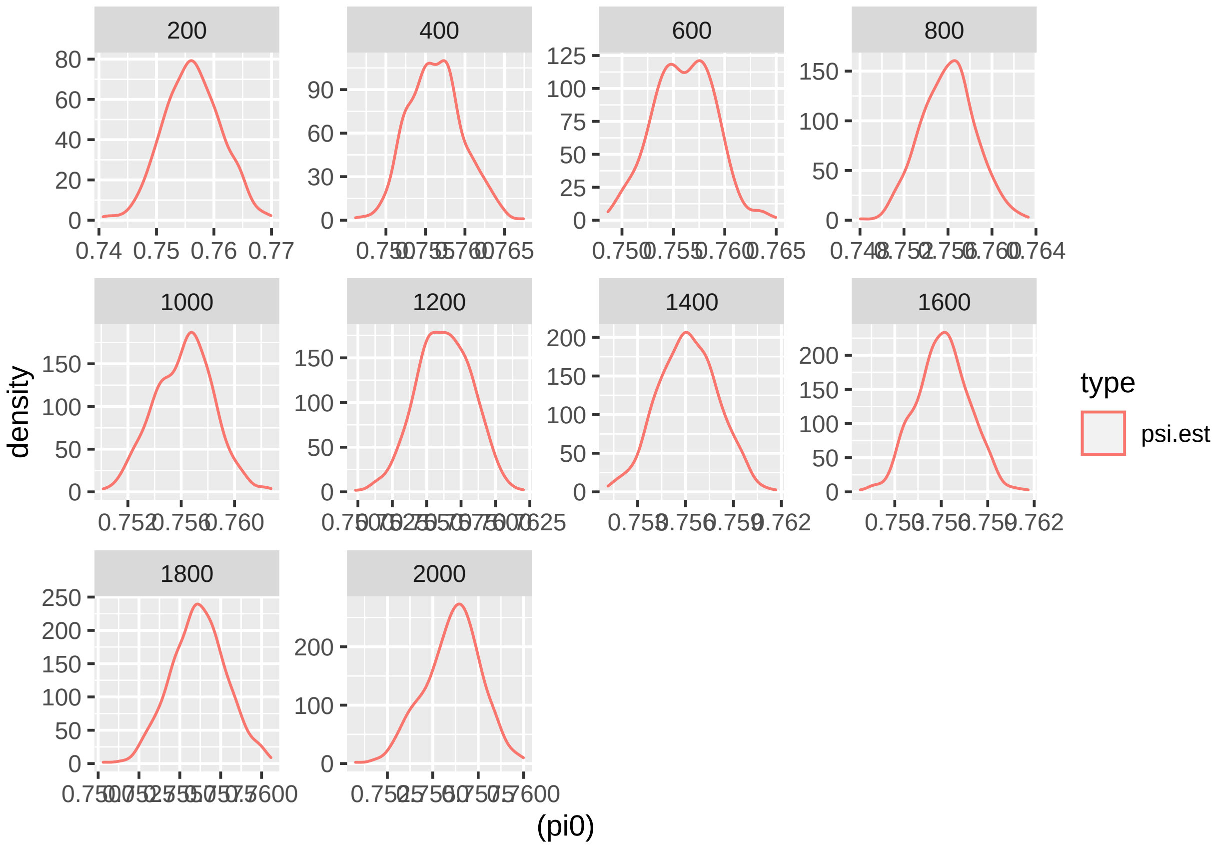
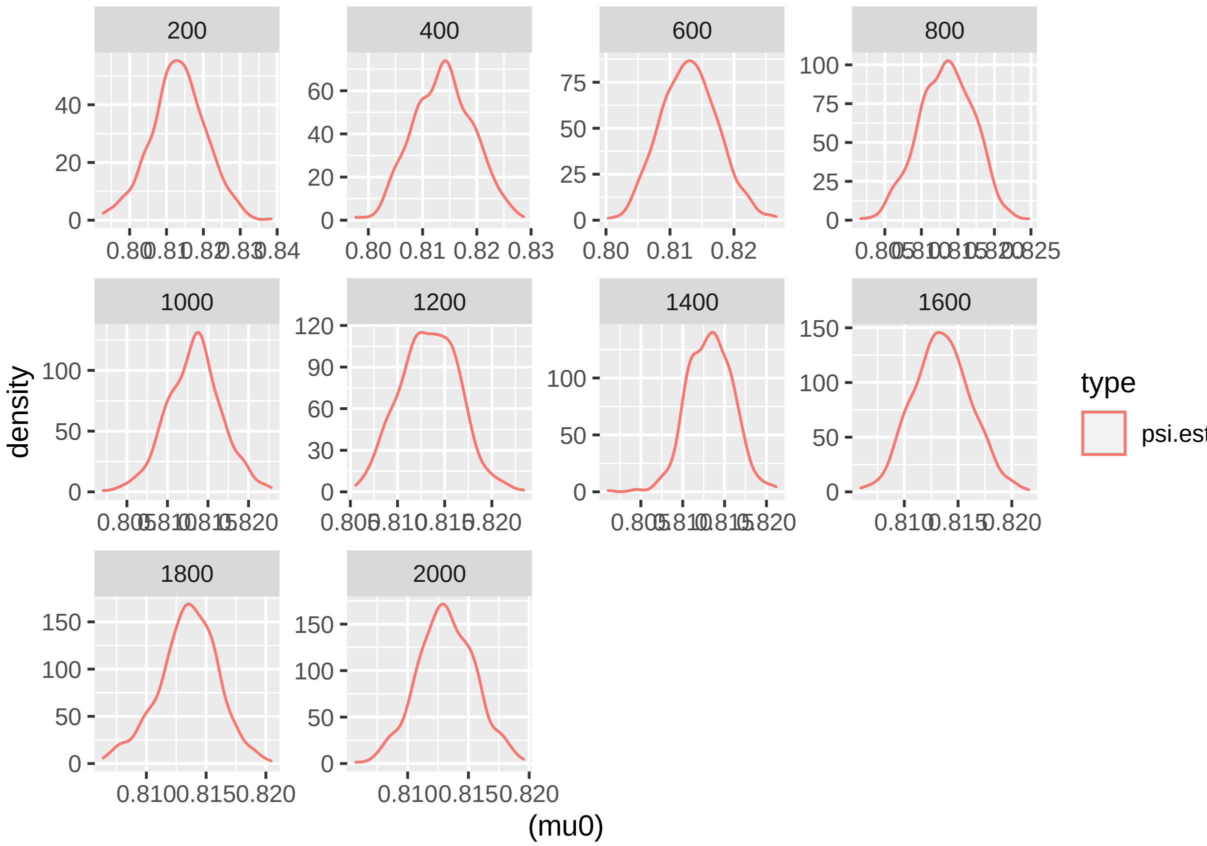
# Simulation result

## 1. case 1

```
## true data background
# W <- matrix(runif(n*3, 0, 1), ncol=3)
# A <- rbinom(n, size = 1, prob = pi0(W))
# if(null.sims) {
#   Y <- rbinom(n, size = 1, prob = mu0.null(A, W))
#   psi0 <- mean((mu0.null(1,W) - mu0.null(0,W))^2)
#   theta0 <- var((mu0.null(1,W) - mu0.null(0,W)))
# } else {
#   Y <- rbinom(n, size = 1, prob = mu0(A, W))
#   psi0 <- mean((mu0(1,W) - mu0(0,W))^2)
#   theta0 <- var((mu0(1,W) - mu0(0,W)))
# }

# ests.sim.1 <- ests.sim(200*c(1:10), 1:500, control = list(conf.int = TRUE), null.sims=FALSE, out.glm=

load("ests.sim.1.Rdata")
ggplot(subset(ests.sim.1, (type %in% 'psi.est'))) +
  geom_density(aes((pi0), color=type)) +
  facet_wrap(~n, scales='free')
```




```r
ggplot(subset(ests.sim.1, (type %in% 'psi.est'))) +
  geom_density(aes((mu0), color=type)) +
  facet_wrap(~n, scales='free')
```



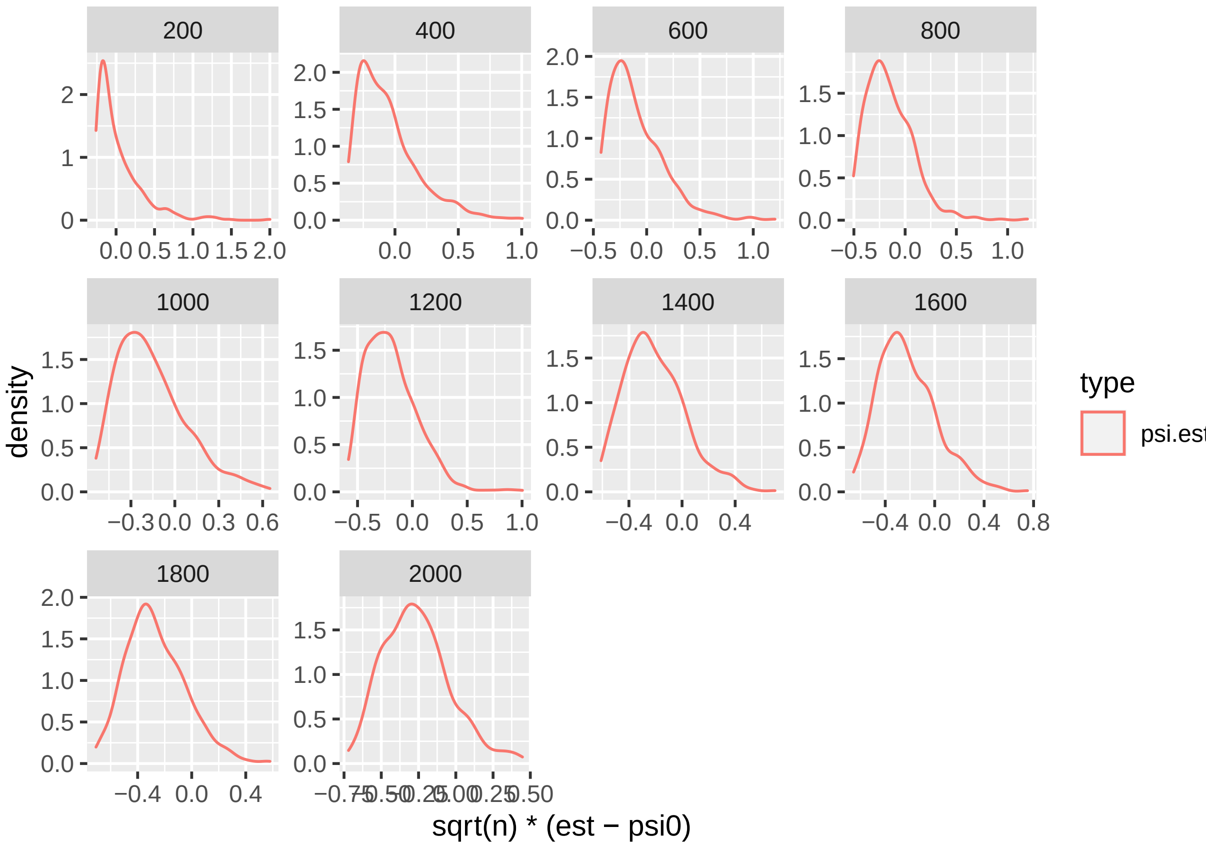

- null.sims=FALSE, which means that `mu1.hat` should be different from `mu0.hat`.
- The mean of `pi0` is around 0.75 and the mean of `mu0` is around 0.8.
- use `glm` to estimate both `pi.hat` and `mu.hat` in simple unit tests.

### 1.1 rates of convergence

$$n^{\frac{1}{2}}(\psi_n - \psi_0) \xrightarrow{d} N(0, \sigma^2_{0,\psi})$$

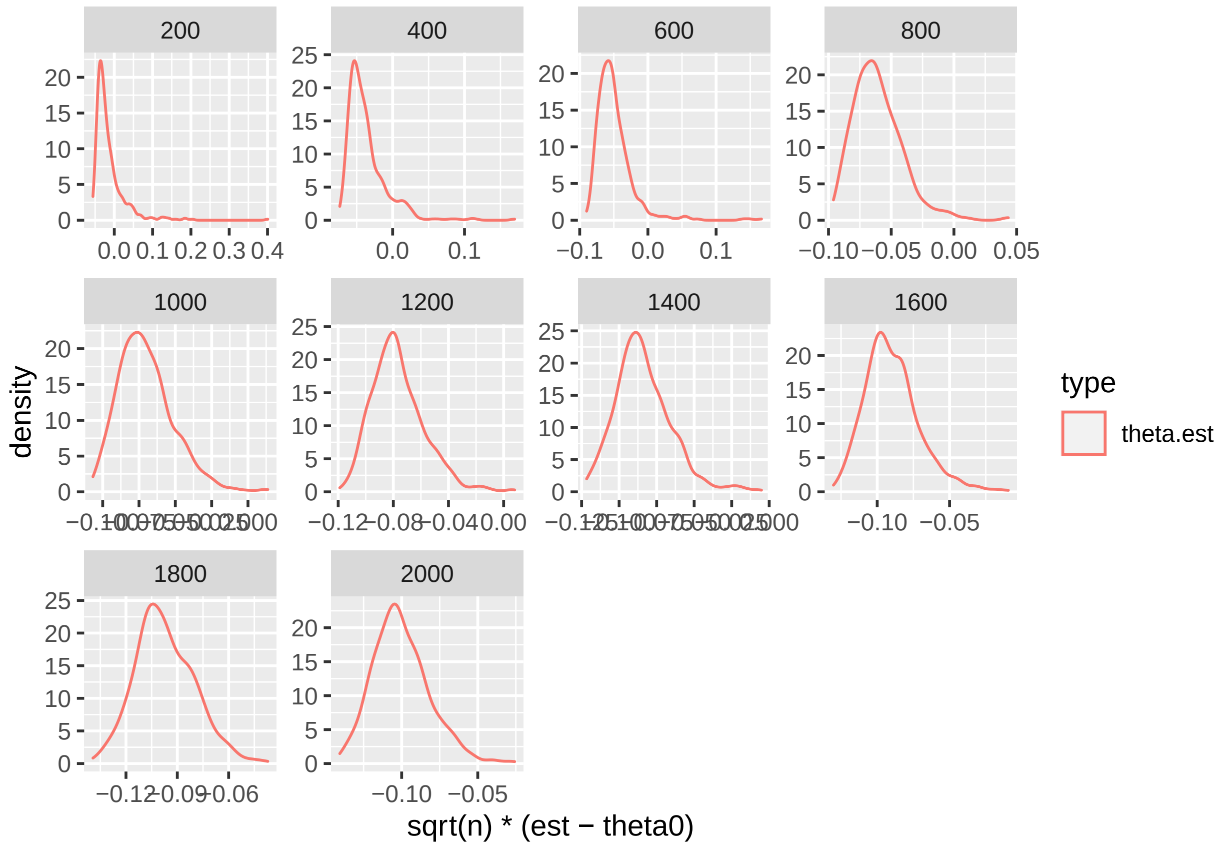
$$n^{\frac{1}{2}}(\theta_n - \theta_0) \xrightarrow{d} N(0, \sigma^2_{0,\theta})$$

```
ggplot(subset(ests.sim.1, (type %in% 'psi.est'))) +
  geom_density(aes(sqrt(n) * (est - psi0), color=type)) +
  facet_wrap(~n, scales='free')
```




#### 1.1.1 psi

```
ggplot(subset(ests.sim.1, (type %in% 'theta.est'))) +
  geom_density(aes(sqrt(n) * (est - theta0), color=type)) +
  facet_wrap(~n, scales='free')
```

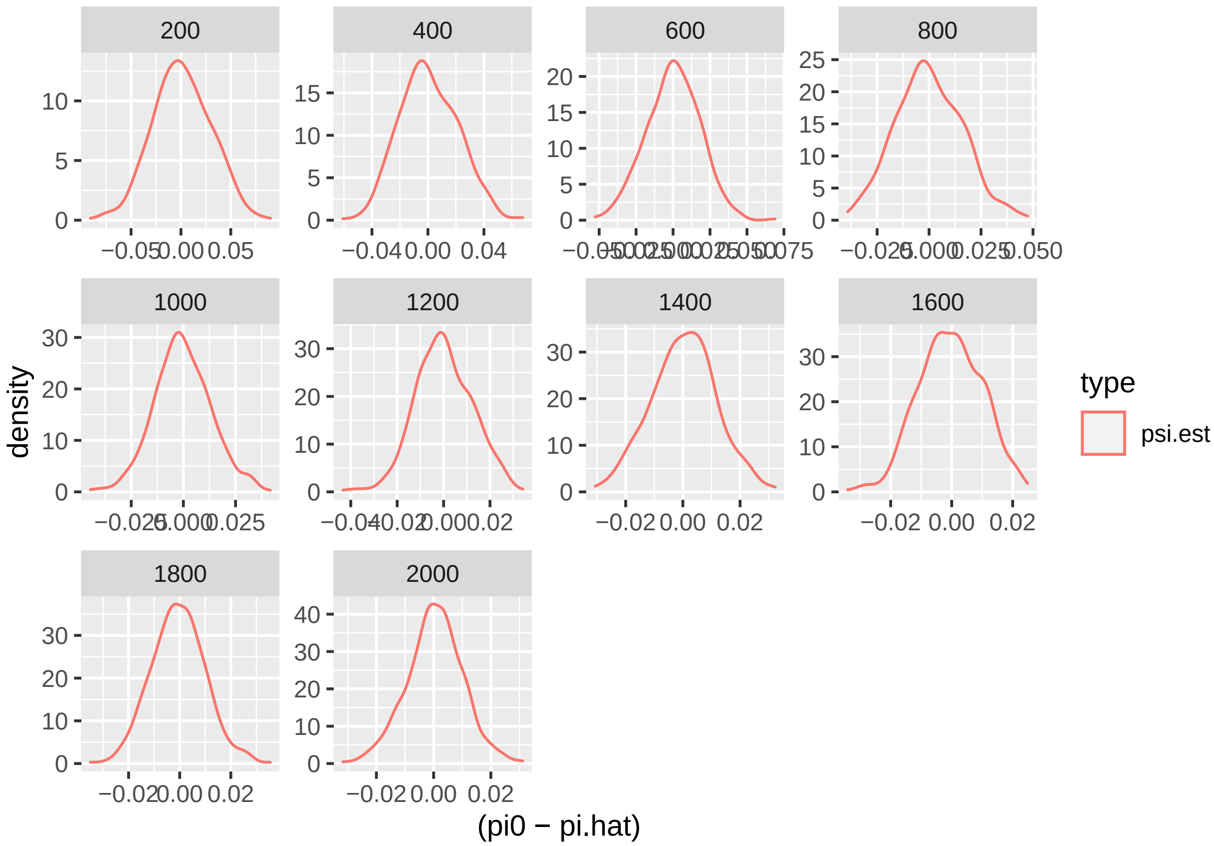



### 1.1.2 theta

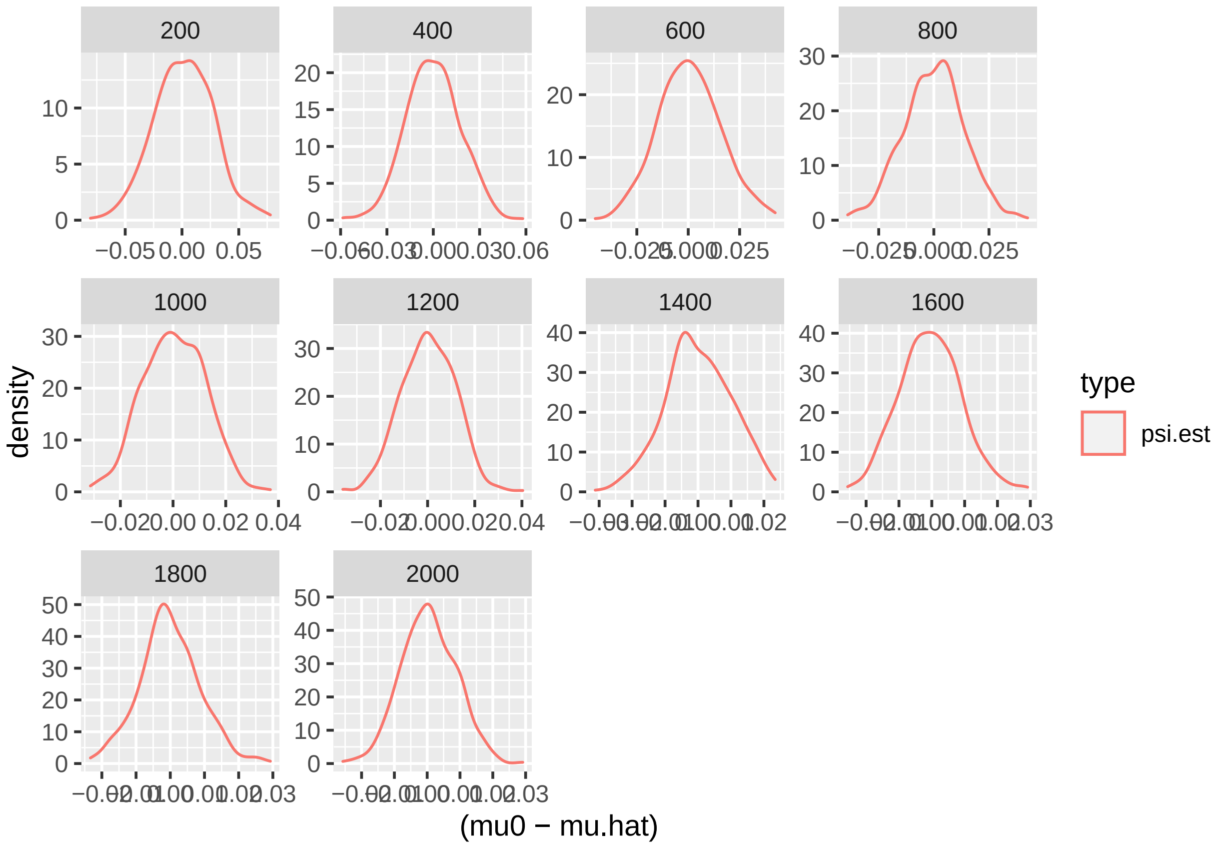
- Both $\psi_n$ and $\theta_n$ didn't seem to converge to Normal distribution as n goes to infinity.

**How about the performance of nuisance parameters?**

```
ggplot(subset(ests.sim.1, (type %in% 'psi.est'))) +
  geom_density(aes((pi0 - pi.hat), color=type)) +
  facet_wrap(~n, scales='free')
```




```
ggplot(subset(ests.sim.1, (type %in% 'psi.est'))) +
  geom_density(aes((mu0 - mu.hat), color=type)) +
  facet_wrap(~n, scales='free')
```



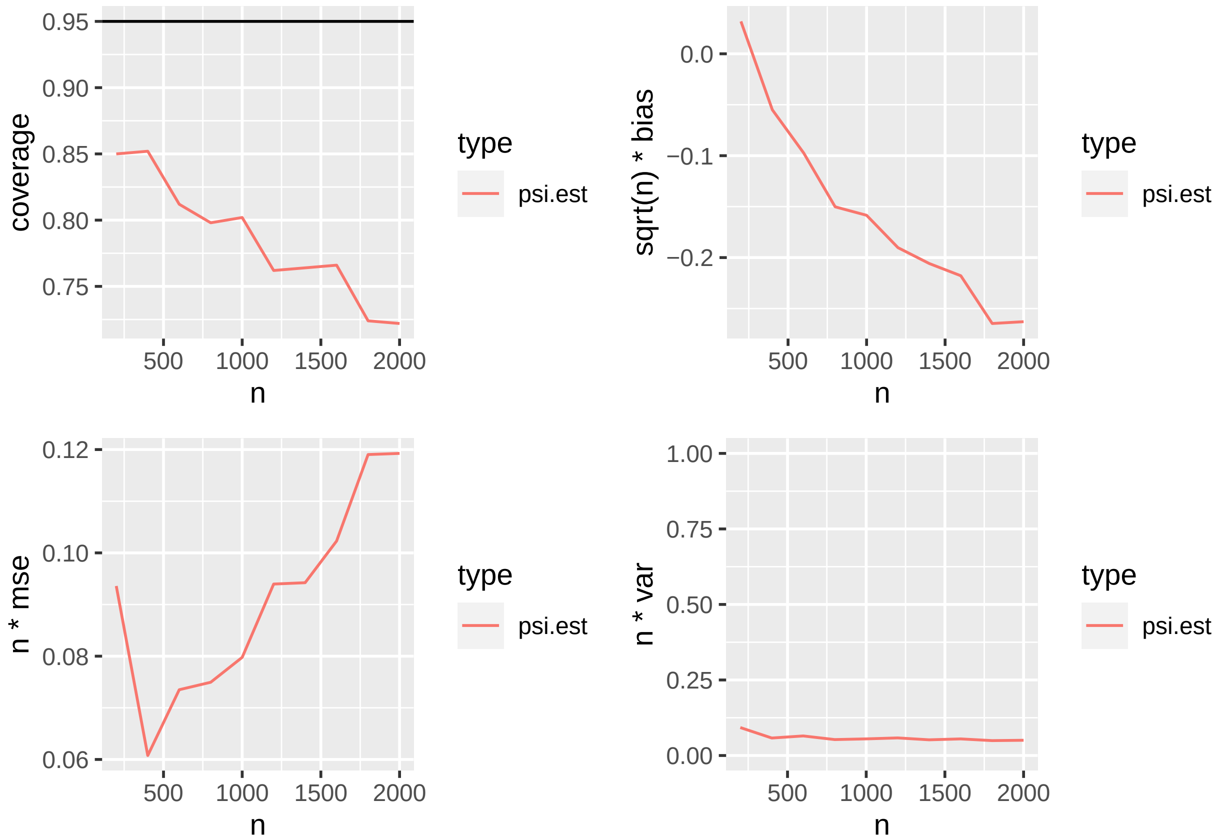

### 1.2 estimate and confidence interval

```
psi.summaries.1 <- ddply(subset(ests.sim.1, (type %in% 'psi.est')), .(n, type), summarize, na = sum(is.
                         coverage = mean(ll <= psi0 & psi0 <= ul, na.rm=TRUE),
                         bias = mean(est - psi0, na.rm=TRUE),
                         var = var(est, na.rm=TRUE),
                         mse = mean((est - psi0)^2, na.rm=TRUE))
```

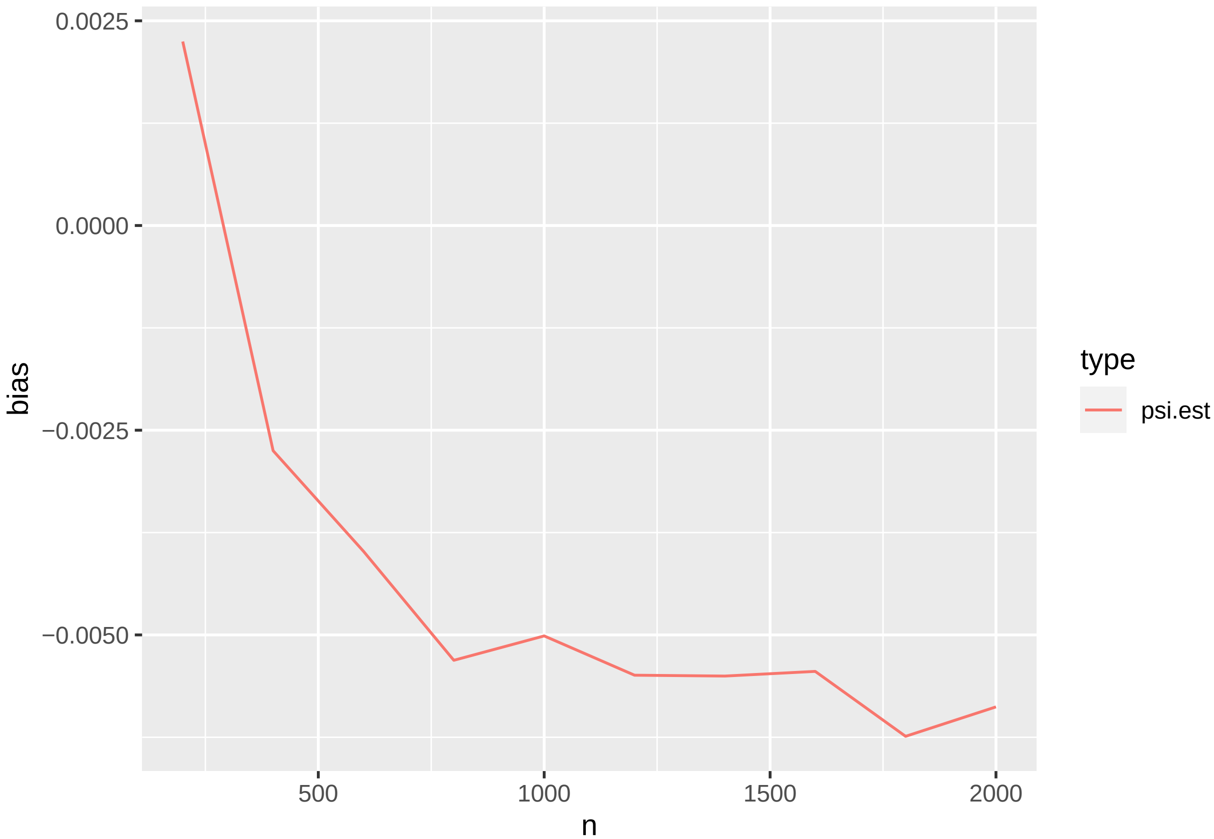
#### 1.2.1 psi Wald-type

```
library(ggplot2)
p1 <- ggplot(psi.summaries.1) +
  geom_line(aes(n, coverage, color=type)) +
  geom_hline(yintercept=.95)

p2 <- ggplot(psi.summaries.1) +
  geom_line(aes(n, sqrt(n) * bias, color=type))

p3 <- ggplot(psi.summaries.1) +
  geom_line(aes(n, n * mse, color=type))

p4 <- ggplot(psi.summaries.1) +
  geom_line(aes(n, n * var, color=type)) +
  coord_cartesian(ylim=c(0,1))

library(gridExtra)
```

```r
grid.arrange(p1, p2, p3, p4, ncol=2)
```

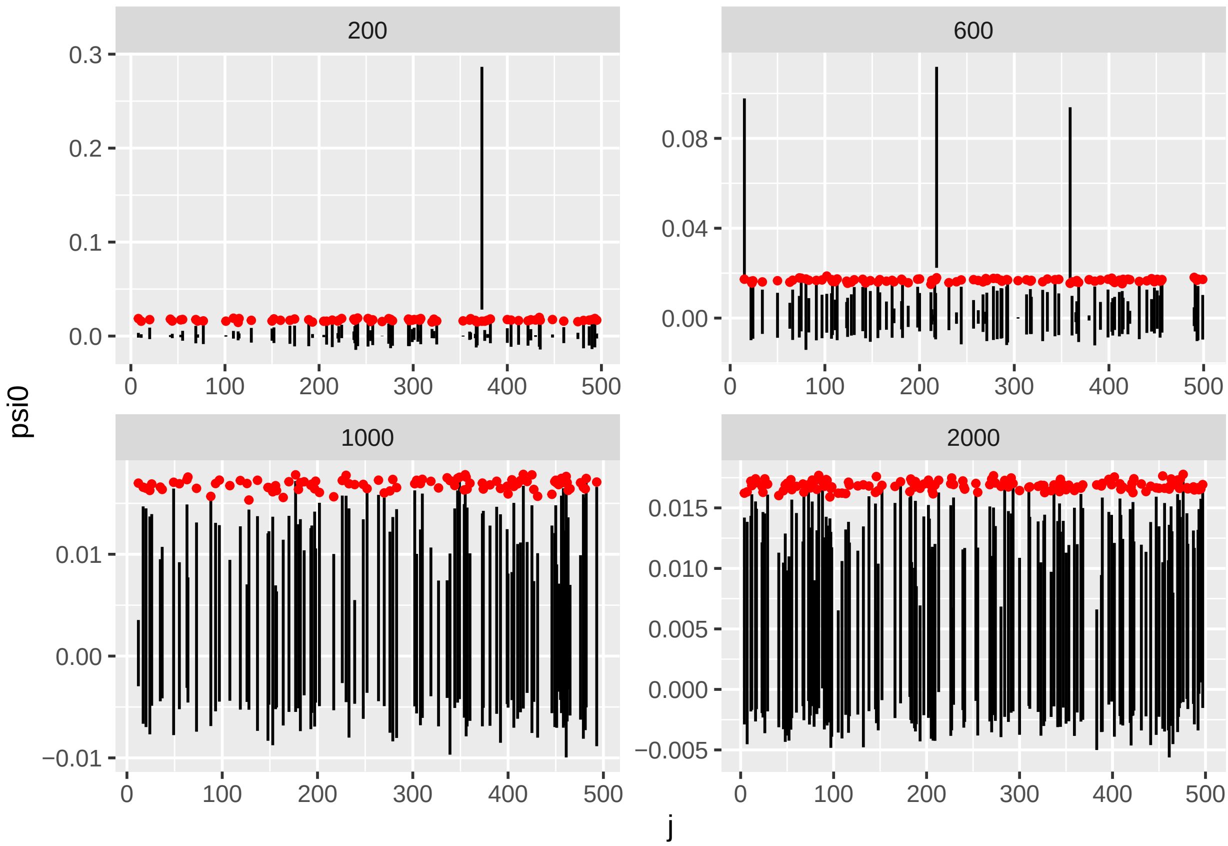



```r
ggplot(psi.summaries.1) +
  geom_line(aes(n, bias, color=type))
```




- The Wald-type CI coverages are below 0.95.
- As n increases, the coverage is going down.
- (have already check the formula and the formula should be correct).
- The absolute value of bias is getting larger even if plotting with bias instead of sqrt(n)*bias.

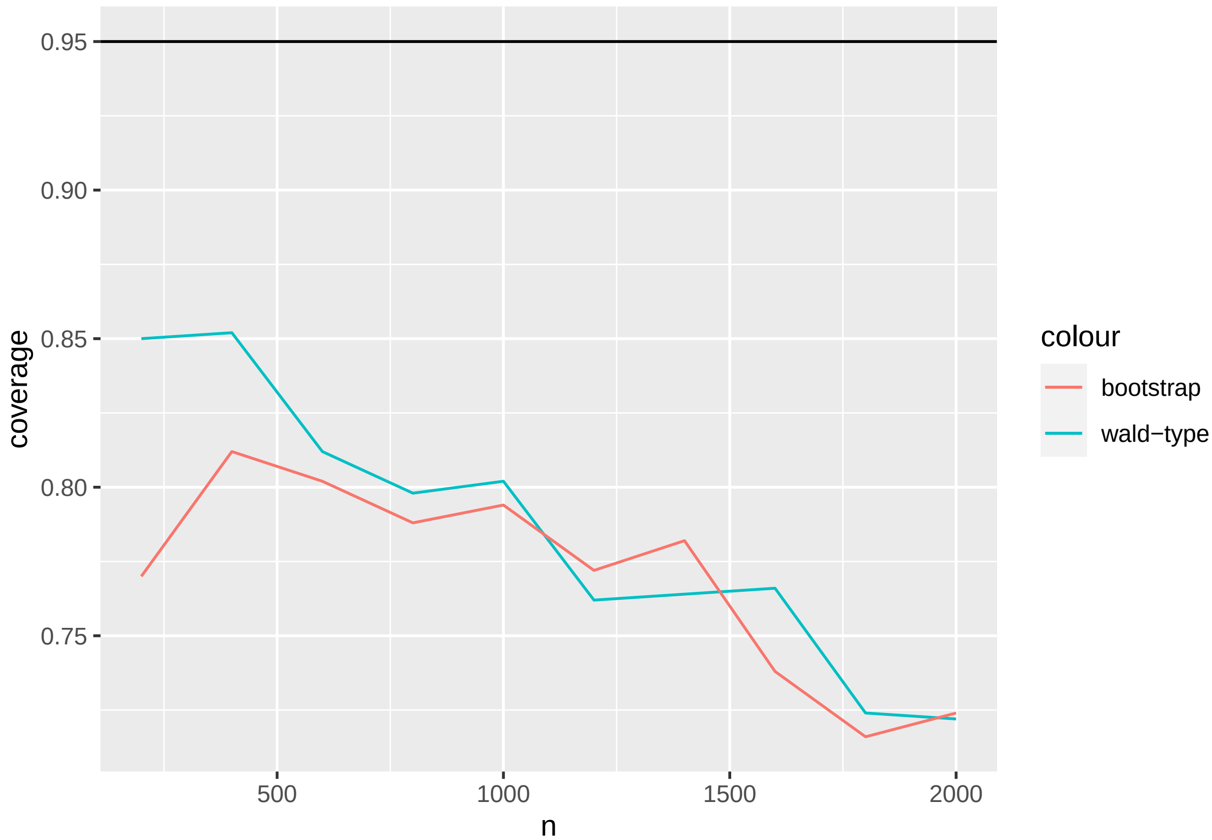
```
ests.sim.1$psi.coverage <- (ests.sim.1$ll <= ests.sim.1$psi0) & (ests.sim.1$psi0 <= ests.sim.1$ul)
ests.sim.1$theta.coverage <- (ests.sim.1$ll <= ests.sim.1$theta0) & (ests.sim.1$theta0 <= ests.sim.1$ul
ggplot(subset(ests.sim.1, (type %in% 'psi.est') & (n %in% c(200, 600, 1000, 2000)) &
                (psi.coverage ==FALSE))) +
  geom_linerange(mapping = aes(x=j, ymin=ll, ymax=ul)) +
  geom_point(mapping=aes(x=j, y=psi0), size=1, col="red") +
  facet_wrap(~n, scales='free')
```




- So I plot out all the Confidence Intervals that fails to cover the $\psi_0$.
- The estimator $\psi_n$ underestimate the real $\psi_0$.

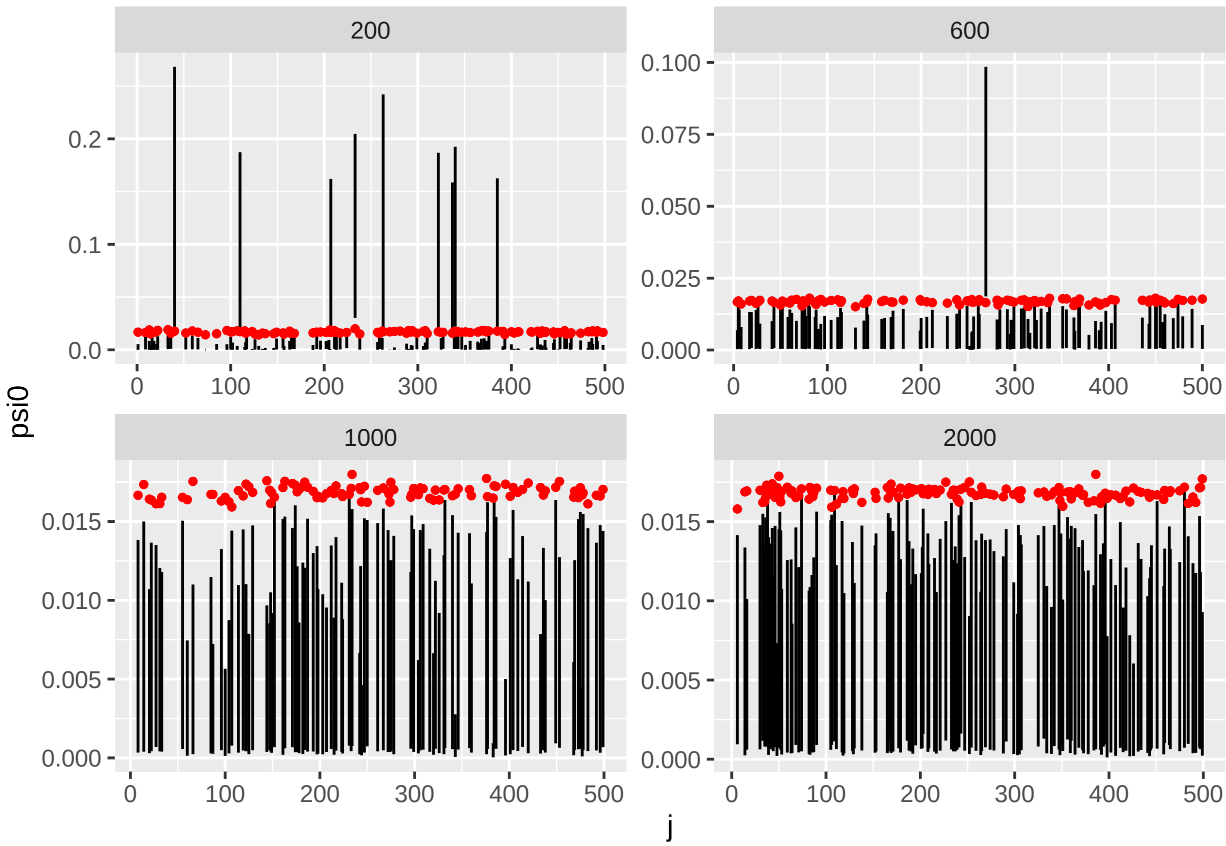
**Bootstrap**

```
# ests.sim.2 <- ests.sim(200*c(1:10), 1:500, control = list(conf.int = TRUE, conf.int.type = 'boot'), n
load("ests.sim.2.Rdata")
# analysis
psi.summaries.2 <- ddply(subset(ests.sim.2, (type %in% 'psi.est')), .(n, type), summarize, na = sum(is.n
                    coverage = mean(ll <= psi0 & psi0 <= ul, na.rm=TRUE),
                    bias = mean(est - psi0, na.rm=TRUE),
                    var = var(est, na.rm=TRUE),
                    mse = mean((est - psi0)^2, na.rm=TRUE))

ggplot() +
  geom_line(data = psi.summaries.1, aes(n, coverage, colour='wald-type')) +
  geom_line(data = psi.summaries.2, aes(n, coverage, colour='bootstrap')) +
  geom_hline(yintercept=.95)
```




- The Bootstrap CI coverages also below 0.95.

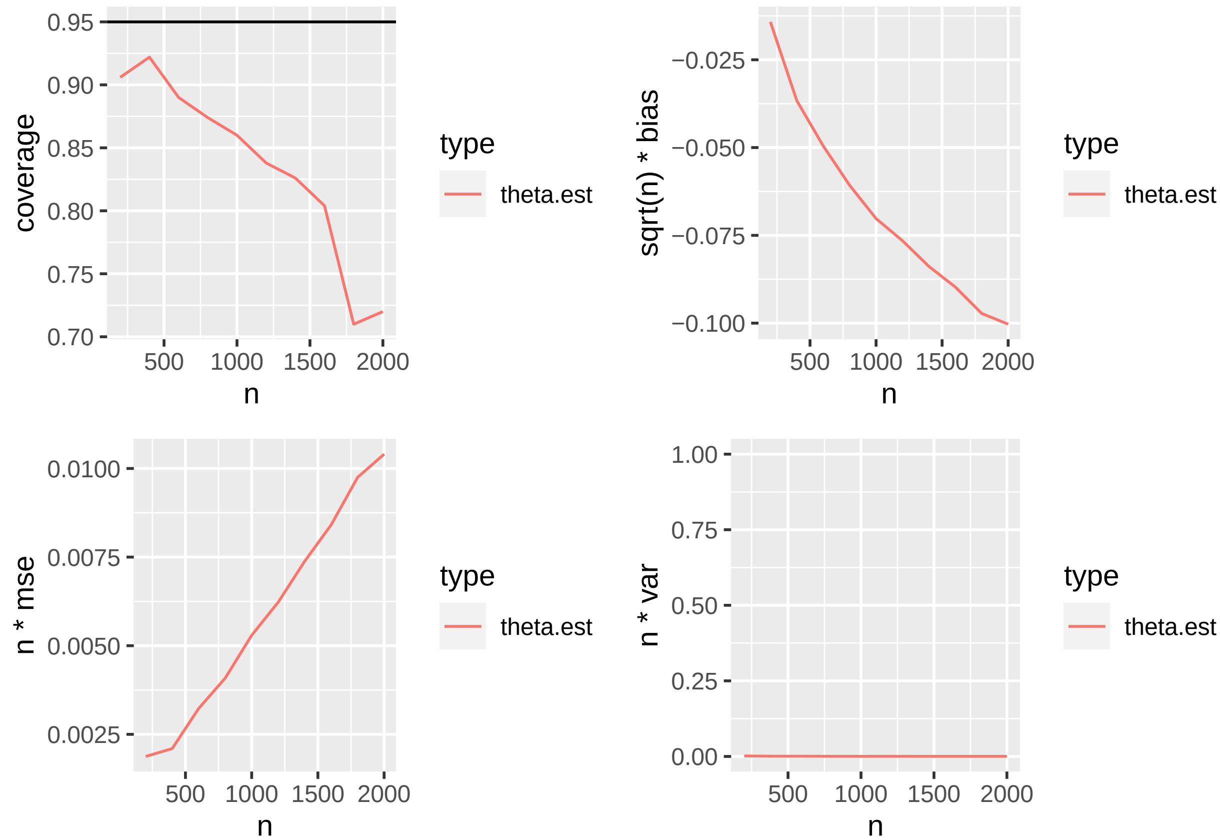
```
ests.sim.2$psi.coverage <- (ests.sim.2$ll <= ests.sim.2$psi0) & (ests.sim.2$psi0 <= ests.sim.2$ul)
ests.sim.2$theta.coverage <- (ests.sim.2$ll <= ests.sim.2$theta0) & (ests.sim.2$theta0 <= ests.sim.2$ul)
ggplot(subset(ests.sim.2, (type %in% 'psi.est') & (n %in% c(200, 600, 1000, 2000)) &
                (psi.coverage ==FALSE))) +
  geom_linerange(mapping = aes(x=j, ymin=ll, ymax=ul)) +
  geom_point(mapping=aes(x=j, y=psi0), size=1, col="red") +
  facet_wrap(~n, scales='free')
```



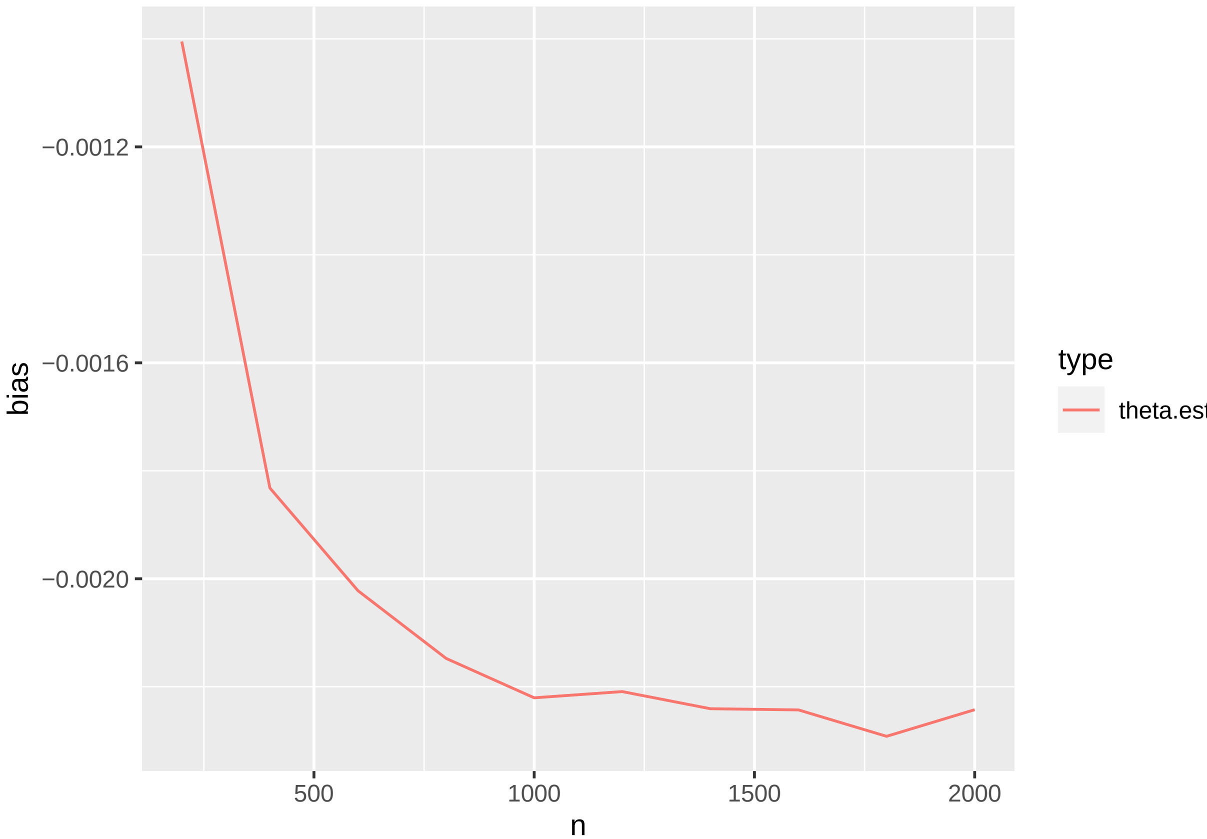

- There are more Bootstrap confidence intervals overestimating the $\psi_0$ when n is small.

### 1.2.2 theta Wald-type

```
theta.summaries.1 <- ddply(subset(ests.sim.1, (type %in% 'theta.est')), .(n, type), summarize,
                           na = sum(is.na(est)),
                           coverage = mean(ll <= theta0 & theta0 <= ul, na.rm=TRUE),
                           bias = mean(est - theta0, na.rm=TRUE),
                           var = var(est, na.rm=TRUE),
                           mse = mean((est - theta0)^2, na.rm=TRUE))
p1 <- ggplot(theta.summaries.1) +
  geom_line(aes(n, coverage, color=type)) +
  geom_hline(yintercept=.95)

p2 <- ggplot(theta.summaries.1) +
  geom_line(aes(n, sqrt(n) * bias, color=type))

p3 <- ggplot(theta.summaries.1) +
  geom_line(aes(n, n * mse, color=type))

p4 <- ggplot(theta.summaries.1) +
  geom_line(aes(n, n * var, color=type)) +
  coord_cartesian(ylim=c(0,1))

library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol=2)
```

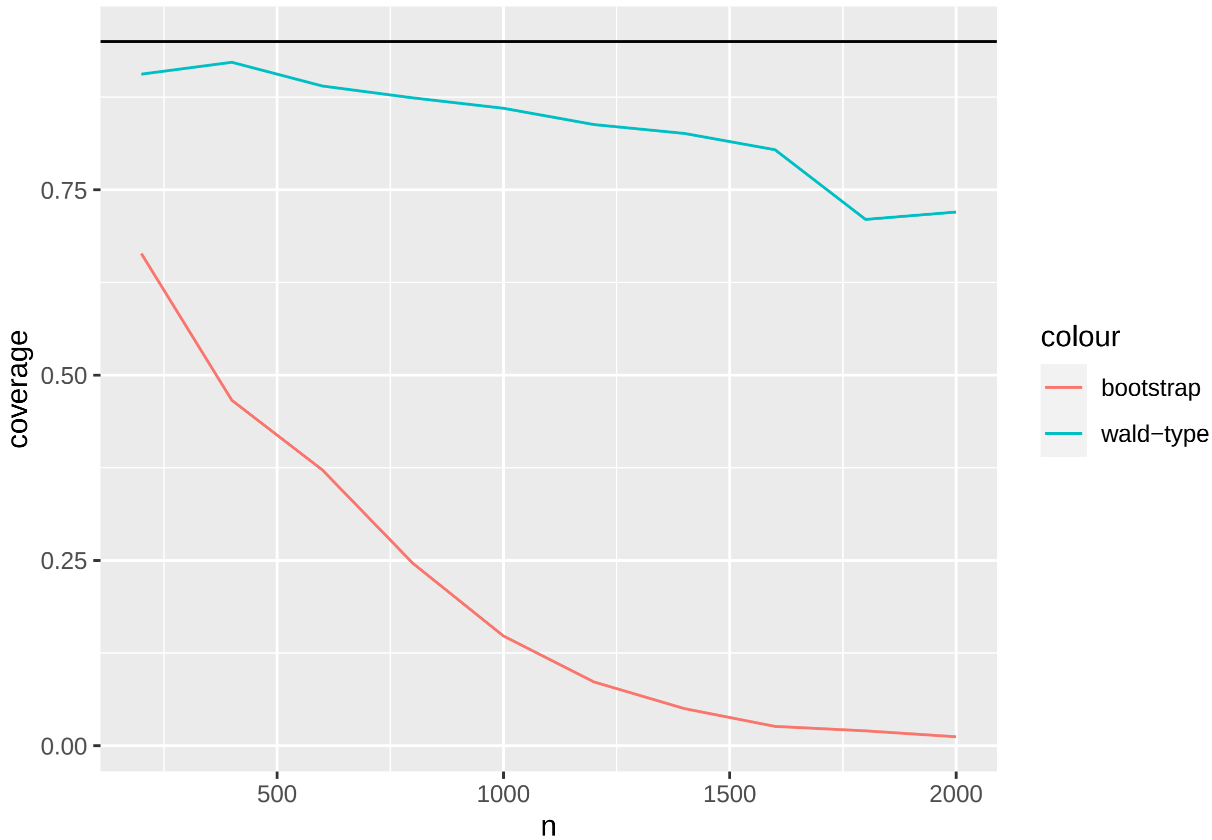



```
ggplot(theta.summaries.1) +
  geom_line(aes(n, bias, color=type))
```



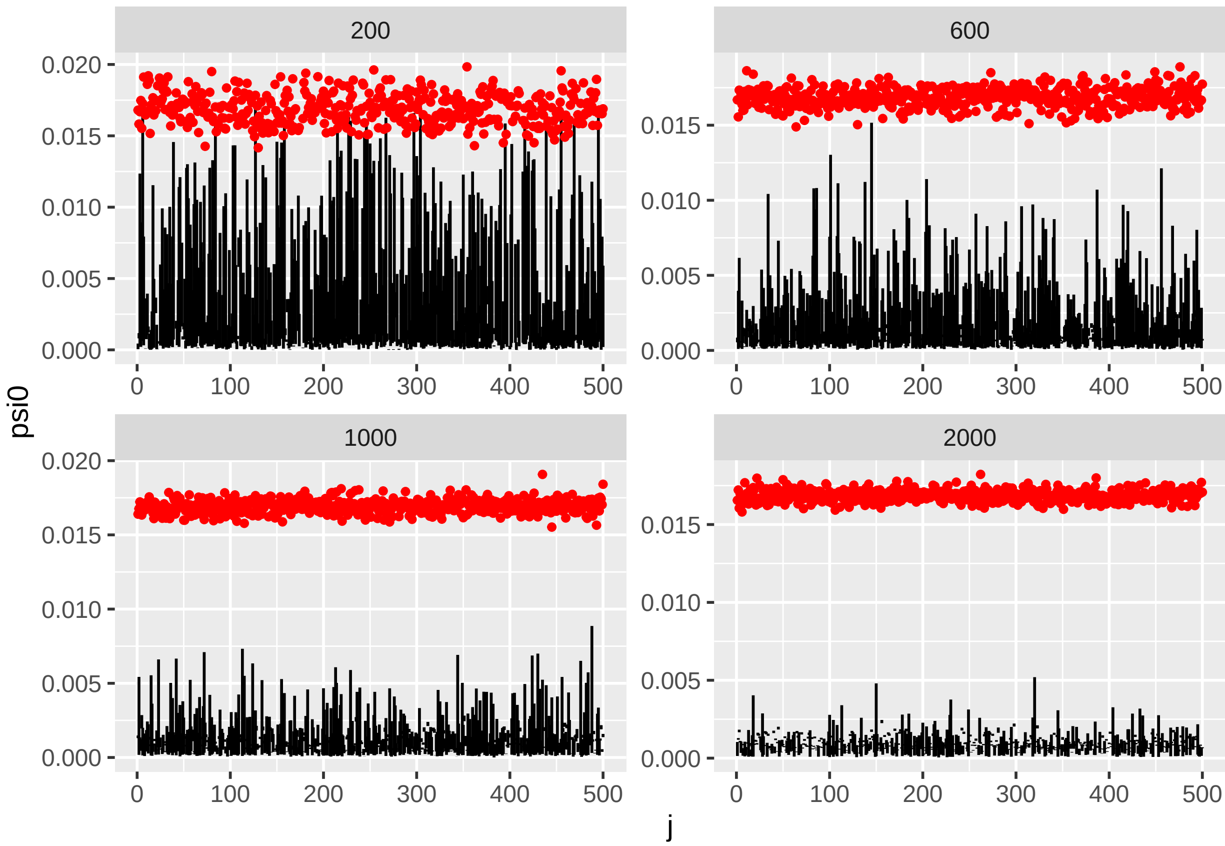

**Bootstrap**

```
theta.summaries.2 <- ddply(subset(ests.sim.2, (type %in% 'theta.est')), .(n, type), summarize,
                          na = sum(is.na(est)),
                          coverage = mean(ll <= theta0 & theta0 <= ul, na.rm=TRUE),
                          bias = mean(est - theta0, na.rm=TRUE),
                          var = var(est, na.rm=TRUE),
                          mse = mean((est - theta0)^2, na.rm=TRUE))

ggplot() +
  geom_line(data = theta.summaries.1, aes(n, coverage, colour='wald-type')) +
  geom_line(data = theta.summaries.2, aes(n, coverage, colour='bootstrap')) +
  geom_hline(yintercept=.95)
```




- Bootstrap confidence levels seem to be weird.

```
ggplot(subset(ests.sim.2, (type %in% 'theta.est') & (n %in% c(200, 600, 1000, 2000)) &
                (psi.coverage ==FALSE))) +
  geom_linerange(mapping = aes(x=j, ymin=ll, ymax=ul)) +
  geom_point(mapping=aes(x=j, y=psi0), size=1, col="red") +
  facet_wrap(~n, scales='free')
```




**Why Bootstrap confidence levels seem to be so weird?**

```
# unit test
null.sims <- FALSE

n <- 1000
set.seed(39338064)
W <- matrix(runif(n*3, 0, 1), ncol=3)
A <- rbinom(n, size = 1, prob = pi0(W))
if(null.sims) {
  Y <- rbinom(n, size = 1, prob = mu0.null(A, W))
  psi0 <- mean((mu0.null(1,W) - mu0.null(0,W))^2)
  theta0 <- var((mu0.null(1,W) - mu0.null(0,W)))
} else {
  Y <- rbinom(n, size = 1, prob = mu0(A, W))
  psi0 <- mean((mu0(1,W) - mu0(0,W))^2)
  theta0 <- var((mu0(1,W) - mu0(0,W)))
}

# use `glm` for both pi.hat and mu.hat in simple unit test
prop.reg <- gam(A ~ s(W[,1]) + s(W[,2]) + s(W[,3]), family = 'binomial')
pi.hat <- prop.reg$fitted.values
AW <- cbind(A, data.frame(W))
mu.reg <- glm(Y ~ ., data=AW, family='binomial')
mu.hat <- mu.reg$fitted.values
mu1.hat <- predict(mu.reg, newdata = cbind(data.frame(A = 1),
                                           data.frame(W)), type = 'response')
```

```
mu0.hat <- predict(mu.reg, newdata = cbind(data.frame(A = 0),
                                          data.frame(W)), type = 'response')
mu.hats <- data.frame(mu1=mu1.hat, mu0=mu0.hat)
# wald
ret1 <- htem.estimator(A, W, Y, control = list(conf.int = TRUE,
                                       pi.hat = pi.hat, mu.hats = mu.hats))
# bootstrap
ret2 <- htem.estimator(A, W, Y, control = list(conf.int = TRUE, conf.int.type='boot',
                                               pi.hat = pi.hat, mu.hats = mu.hats))

cat('theta0:', theta0)
```

```
## theta0: 0.003143182
```

```
# why bootstrap CI didn't work?
theta.test.vector <- rep(0, 500)
for (i in 1:500){
  boot.inds <- sample(1:n, n, replace=TRUE)

  boot.pi.hat <- pi.hat[boot.inds]
  boot.mu.hats <- mu.hats[boot.inds,]
  boot.ret <- htem.estimator(A[boot.inds], W[boot.inds], Y[boot.inds],
                             control = list(pi.hat = boot.pi.hat,
                                            mu.hats = boot.mu.hats,
                                            conf.int = FALSE))
  theta.test.vector[i] <-boot.ret$est[2]
}
density(theta.test.vector)
```

```
##
## Call:
##  density.default(x = theta.test.vector)
##
## Data: theta.test.vector (500 obs.);  Bandwidth 'bw' = 1.315e-05
##
##        x                    y
##  Min.   :0.0001208   Min.   :   0.000
##  1st Qu.:0.0008961   1st Qu.:   0.000
##  Median :0.0016715   Median :   0.000
##  Mean   :0.0016715   Mean   : 322.277
##  3rd Qu.:0.0024468   3rd Qu.:   3.512
##  Max.   :0.0032221   Max.   :7908.890
```

```
quantile(theta.test.vector, c(0.025,0.975))
```

```
##        2.5%       97.5%
## 0.001382813 0.001559773
```

- Since we do bootstrap based on pi.hat and mu.hats, so the estimator won't vary too much.
- The estimator itself also always underestimate the true value.
- Hence it is difficult to cover the true value.

## 2. case 2

- null.sims=TRUE it means that mu0 has nothing to do with A