HW3

Dongnan Liu

2024-02-25

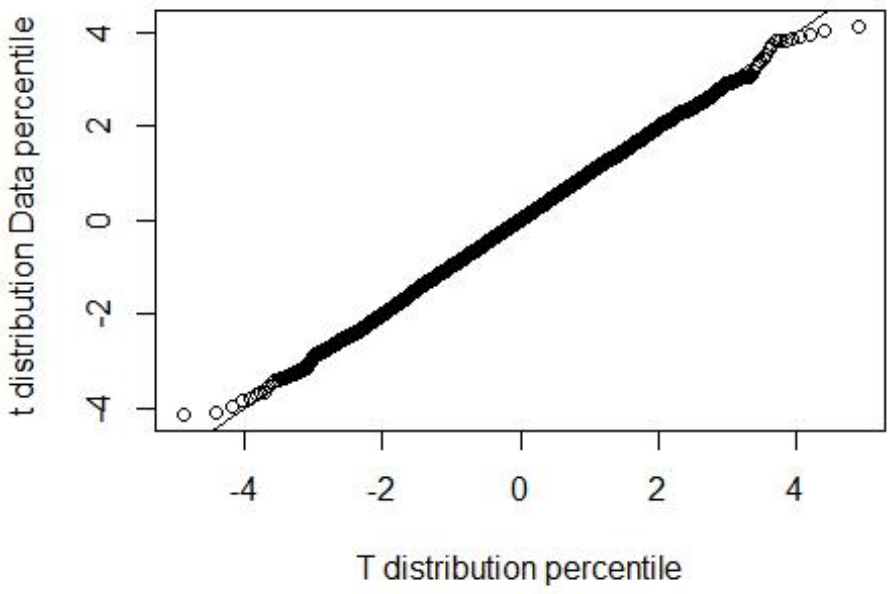
```
#1(a)
set.seed(3701)
mu=68
sd=3
theta=2
reps=9*1e4
x.list=rnorm(n=reps,mean=mu,sd=sd)+runif(n=reps,min=-theta,max=theta)
simu_mean=mean(x.list)
simu_var=var(x.list)
ob_mean=mu
ob_var=sd^2+theta^2/3
c(simu_mean,ob_mean)
```

```
## [1] 67.99953 68.00000
```

```
c(simu_var,ob_var)
```

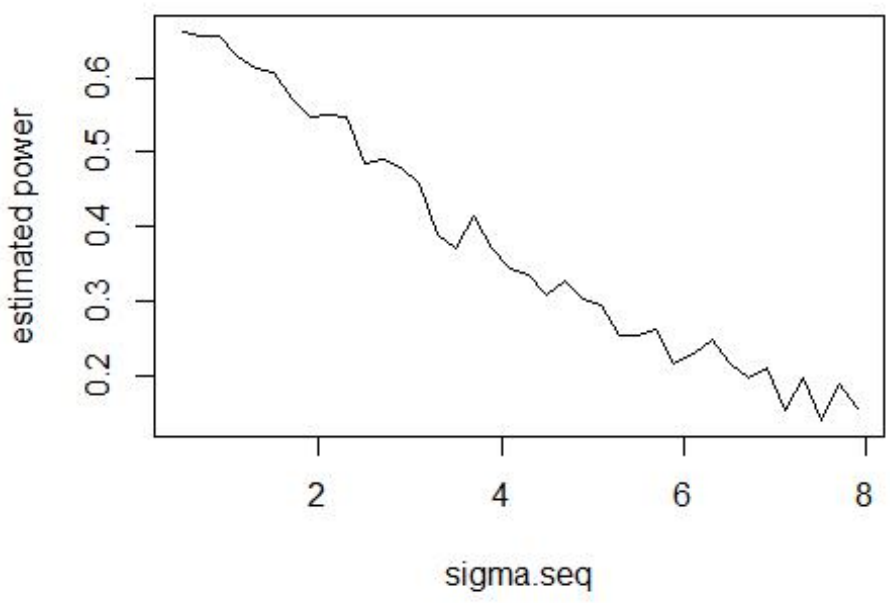
```
## [1] 10.34190 10.33333
```

```
#These simulation-based estimates are close to actual values
#1(b)
set.seed(3701)
n = 20;mean=65; sd = 3; theta = 1.5;mu0=65
reps=1e4
t.list=numeric(reps)
for(r in 1:1e4){
x.list=rnorm(n,mean,sd)+runif(n,min=-theta,max=theta)
xbar=mean(x.list)
t.list[r]=(xbar- mu0)/(sd(x.list)/sqrt(n))
}
probs=ppoints(reps)
plot(qt(probs, df=n-1), quantile(t.list, probs),xlab="T distribution pe
rcentile", ylab=" t distribution Data percentile")
abline(0,1)
```

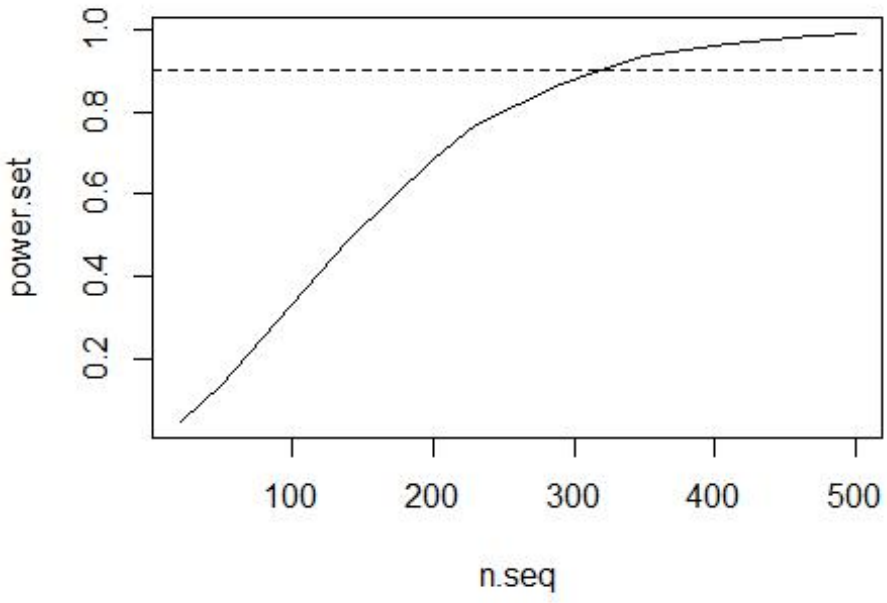



```
#The QQ plot is normal so the distribution of T is well approximated by
the t-distribution with
#n – 1 degrees of freedom.
#1(c)
#return two-sided alternative observed p-value
est.pval.dist=function(n, mu0, mu, sigma, theta, reps=1e4) {
  pvalue.list=numeric(reps)
  for(r in 1:reps)
  {
    x.list=rnorm(n=n, mean=mu, sd=sigma) + runif(n=n, min=-theta, max=th
eta)
    xbar=mean(x.list)
    s=sd(x.list)
    t=(xbar - mu0)/(s/sqrt(n))
    pvalue.list[r]=2*pt(-abs(t), n-1) }
  return(pvalue.list)
}
#simulation-based estimated power curve
set.seed(3701)
mu=66; n=20; mu0=68; alpha=0.05
sigma.seq=seq(from=0.5, to=8, by=0.2)
est.power=numeric(length(sigma.seq))
for(r in 1:length(sigma.seq))
{
  pvals=est.pval.dist(n=n, mu=mu, mu0=mu0, sigma=sigma.seq[r], theta=6,
reps=1e3)
  est.power[r]=mean(pvals < alpha)
```

```
}
plot(sigma.seq, est.power, t="l",
     xlab="sigma.seq", ylab="estimated power")
```




```
#The sequence of values for σ I choose is sigma.seq=seq(from=0.5, to=8,
#by=0.2) so that the simulated estimate of the power decreases from #rou
ghly 70% to roughly 20%.
#1(d)
set.seed(3701)
mu0 = 68; mu = 67; sigma = 3; theta = 6
n.seq=seq(from=20,to=500,by=30)
m=length(n.seq)
power.set=numeric(m)
for(r in 1:m){
  pvals=est.pval.dist(n=n.seq[r], mu=mu, mu0=mu0,sigma=sigma, theta=6,
reps=1e4)
  power.set[r]=mean(pvals < 0.01)
}
plot(n.seq,power.set,type="l")
abline(h=0.90, lty=2)
```




```r
#When n is approximately 300 the simulated estimate of the power is roug
hly
#90%
#2(a)
set.seed(3701)
n=30; mu=68; sigma=3; reps=1e4
var.list=numeric(reps)
for(r in 1:reps){
  x.list=rnorm(n=n, mean=mu, sd=sigma)
  var.list[r]=var(x.list)
}
t.test(var.list, conf.level=0.99)$conf.int
```

```
## [1] 8.952654 9.074745
## attr(,"conf.level")
## [1] 0.99
```

```r
#σ^2 = 9 is in this interval.
#2(b)
set.seed(3701);n=30; mu=68; sigma=3; reps=1e4
est.list=numeric(reps)
for(r in 1:reps){
  x.list=rnorm(n=n, mean=mu, sd=sigma)
  est.list[r]=var(x.list)*((n-1)/(n+1))
}
t.test(est.list, conf.level=0.99)$conf.int
```

```
## [1] 8.375063 8.489277
## attr(,"conf.level")
## [1] 0.99

#σ^2 = 9,so σ^2 is not in this interval. It confirms that  S^2 is biased
 toward zero
#2(c)
set.seed(3701)
mu=68; sigma=3; reps=1e4
n.list=c(5, 10, 50, 200)
for(i in 1:length(n.list)){
  n=n.list[i]
  y.list = numeric(reps)
  z.list = numeric(reps)
  for(r in 1:reps){
    x.list = rnorm(n=n, mean=mu, sd=sigma)
    y.list[r]=abs(var(x.list)-sigma^2)
    z.list[r]=abs(var(x.list)*((n-1)/(n+1))-sigma^2)
  }
    CI=t.test(y.list, conf.level=0.99)$conf.int[1:2]
    cat("n=", n, "99% approx CI for E|S^2 - sigma^2| is\n\t", CI, "\n")
    CI=t.test(z.list, conf.level=0.99)$conf.int[1:2]
    cat("n=", n, "99% approx CI for E|tildeS^2 - sigma^2| is\n\t", CI, "
\n")
    CI=t.test(y.list-z.list, conf.level=0.99)$conf.int[1:2]
    cat("n=", n,"99% approx CI for E(|S^2 - sigma^2| - |tildeS^2 - sigma
^2|) is\n\t",
        CI, "\n")
}

## n= 5 99% approx CI for E|S^2 - sigma^2| is
##   4.812271 5.023056
## n= 5 99% approx CI for E|tildeS^2 - sigma^2| is
##   4.437604 4.570231
## n= 5 99% approx CI for E(|S^2 - sigma^2| - |tildeS^2 - sigma^2|) is
##   0.32559 0.5019025
## n= 10 99% approx CI for E|S^2 - sigma^2| is
##   3.255253 3.390032
## n= 10 99% approx CI for E|tildeS^2 - sigma^2| is
##   3.148151 3.256587
## n= 10 99% approx CI for E(|S^2 - sigma^2| - |tildeS^2 - sigma^2|) is
##   0.07644155 0.1641063
## n= 50 99% approx CI for E|S^2 - sigma^2| is
##   1.436208 1.493052
## n= 50 99% approx CI for E|tildeS^2 - sigma^2| is
##   1.426256 1.480758
## n= 50 99% approx CI for E(|S^2 - sigma^2| - |tildeS^2 - sigma^2|) is
##   0.002076203 0.0201689
## n= 200 99% approx CI for E|S^2 - sigma^2| is
##   0.7123627 0.7406023
```

```
## n= 200 99% approx CI for E|tildeS^2 - sigma^2| is
##   0.7099527 0.7378914
## n= 200 99% approx CI for E(|S^2 - sigma^2| - |tildeS^2 - sigma^2|) is
##   0.0002707623 0.004850182

#S2 is not s better estimator than tildeS^2 is for these data-generating
#models since we see that the E|S2 – σ2| – |S˜2 – σ2| are all greater th
an zero so
#we have tildeS^2 is a better estimator than S2. The difference in estim
ation performance for these two estimators appears to
#decrease as n increases.
```