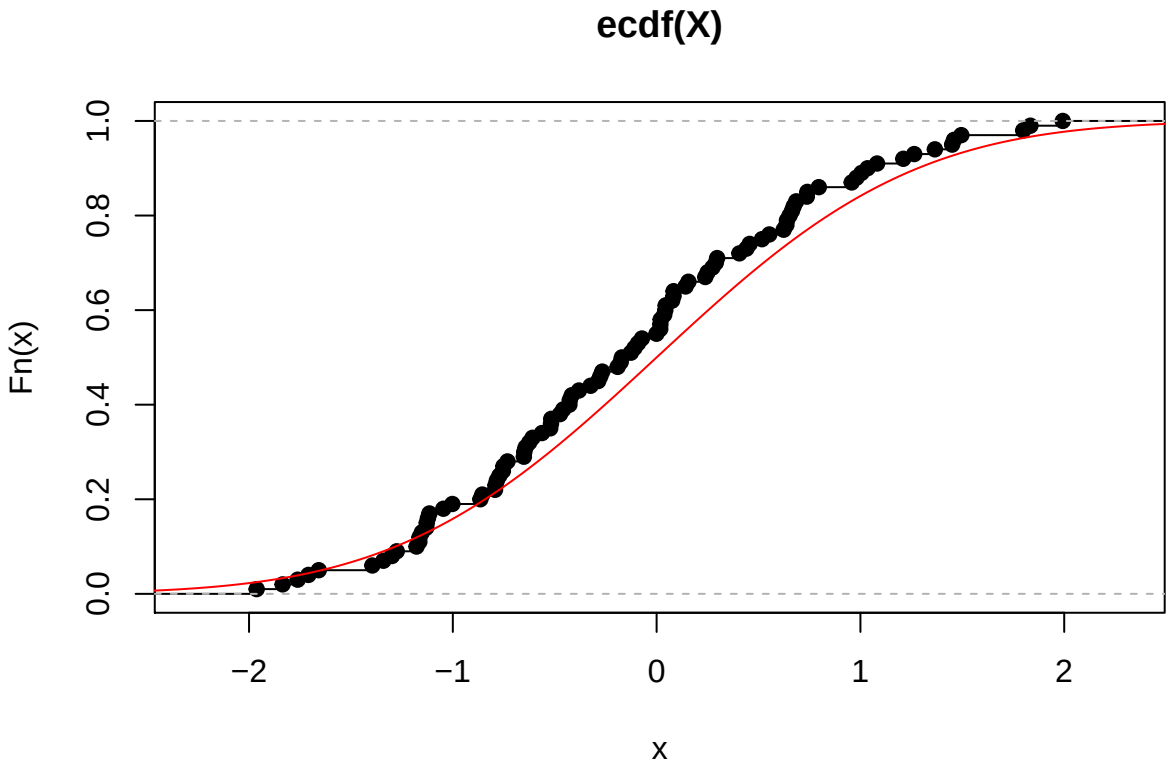
Computer class 3 solutions

Richard Wilkinson

15 March 2016

Solution 0

```
X <- rnorm(100)
plot(ecdf(X))
lines(seq(-4,4,0.01), pnorm(seq(-4,4,0.01)), col=2)
```



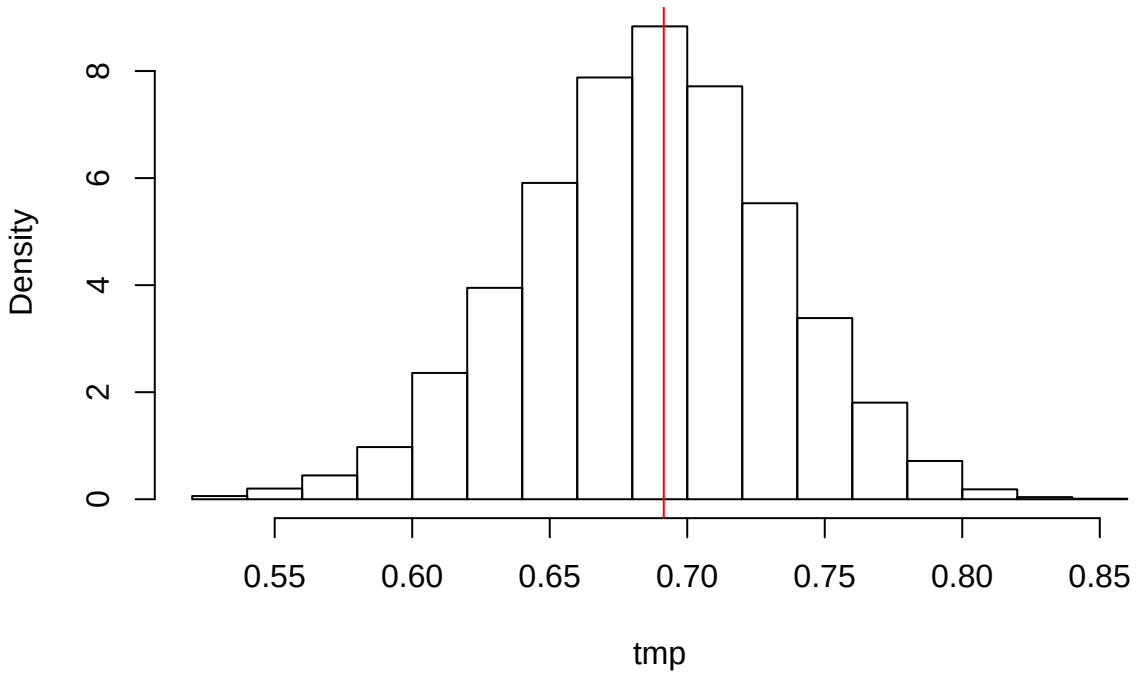

```
sum(X<0.5)/100
```

```
## [1] 0.74
```

```
pnorm(0.5)
```

```
## [1] 0.6914625
```

```
tmp <- replicate(10^4, {X <- rnorm(100)
sum(X<0.5)/100}
)
hist(tmp, probability=T)
abline(v=pnorm(0.5), col=2)
```



Check we are unbiased

```
mean(tmp)-pnorm(0.5)
```

```
## [1] -0.0001884613
```

which looks pretty small to me. By increasing the number of replicates we get a number that shrinks to zero.

Lets now examine the DKW inequality. Rather than considering all x, we'll restrict ourselves to a grid

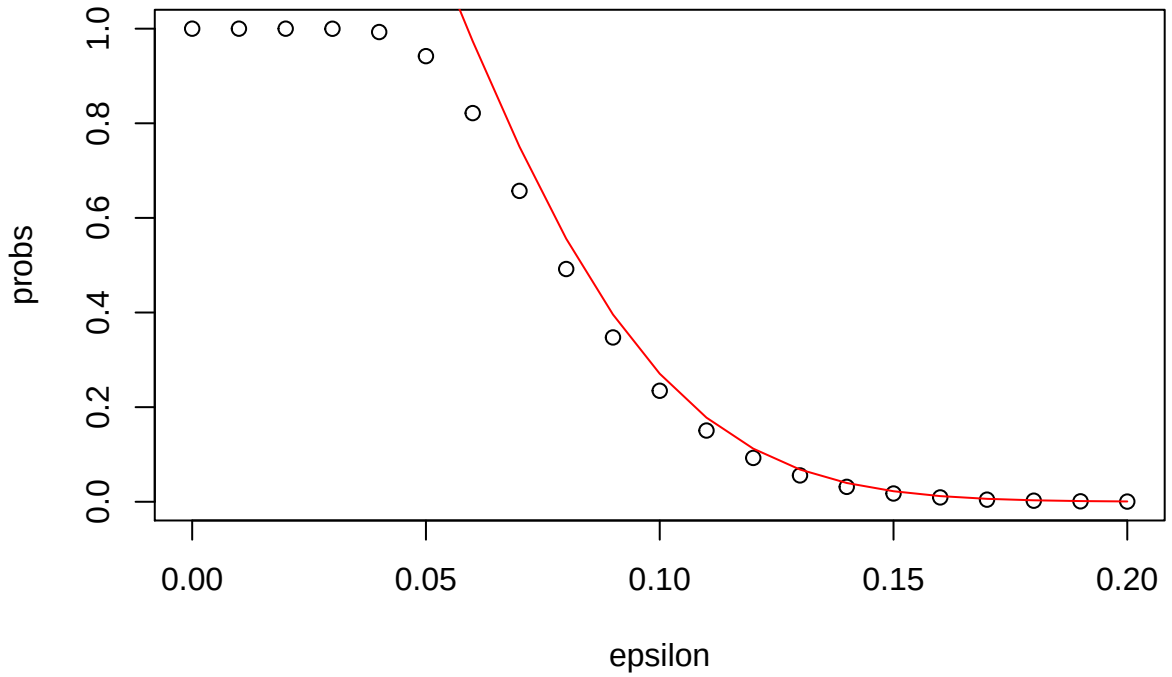
```
xtest <- seq(-4,4,0.01)
```

Lets now generate a large number of datasets, and for each, look at the maximum absolute difference between the ECDF and the true CDF at these test locations.

```
DKW <- replicate(10^5,{X <- rnorm(100)
max(abs(ecdf(X)(xtest)-pnorm(xtest)))})
```

Now lets plot the proportion of these maximums that are bigger than epsilon. On top we will plot the upper bound given by the DKW theorem.

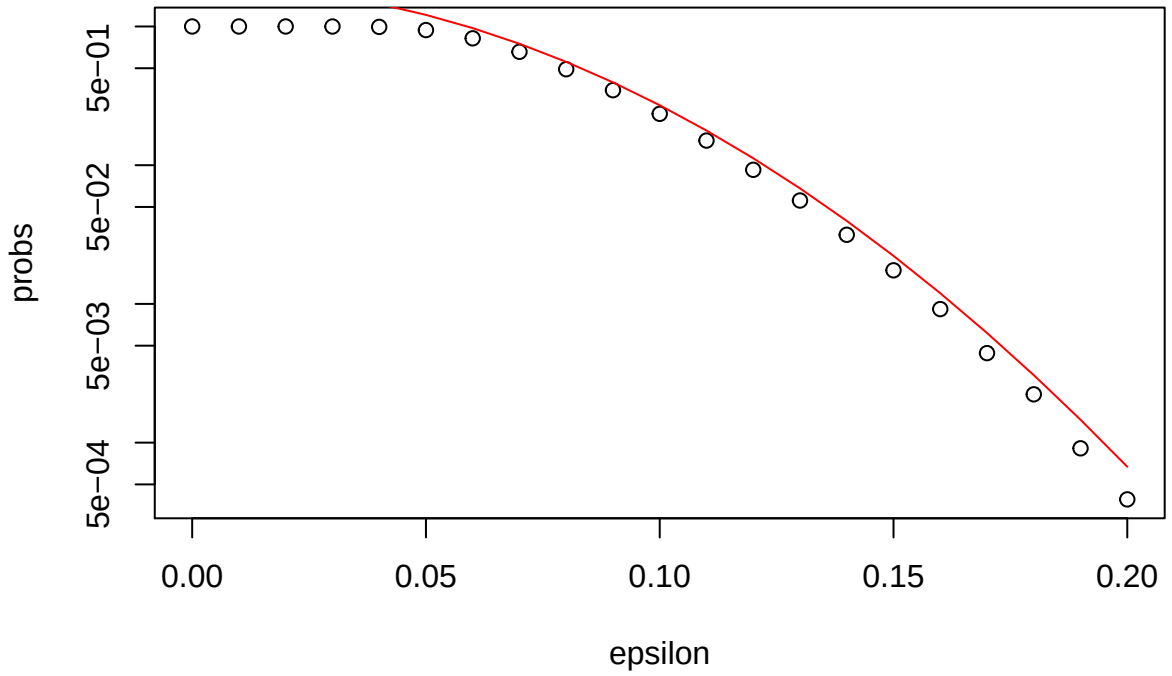
```
epsilon <- seq(0,0.2, 0.01)
probs <- c()
i<-1
for(eps in epsilon){
  probs[i]<-sum(DKW>eps)/length(DKW)
  i<-i+1
}
plot(epsilon, probs)
lines(epsilon, 2*exp(-2*100*epsilon^2), col=2)
```



To make this clearer, let's take the log of the y axis

```
plot(epsilon, probs, log="y")
lines(epsilon, 2*exp(-2*100*epsilon^2), col=2, log="y")
```

```
## Warning in plot.xy(xy.coords(x, y), type = type, ...): "log" is not a
## graphical parameter
```



We can see that the inequality does look to be true (the red curve bounds the black points). Note that as we have estimated the probability of the maximum exceeding epsilon by sampling, this is itself a random estimator, and so there will be some noise here (meaning that we may find the inequality appears not to be true if we don't use enough samples).

Question 1

Consider the `hills` dataset in the MASS package in R, which contains data on the record time for each of 35 Scottish hill races. We want to build a model to predict the record time on the basis of the race distance and the total amount of height gained during the route. Because we are worried about outliers in the data, we will use a robust regression approach using M estimators.

We can fit a robust linear model to this dataset using the command

```
library(MASS)
fit <- rlm(time~dist+climb, hills)
summary(fit)
```

```
##
## Call: rlm(formula = time ~ dist + climb, data = hills)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -10.75039  -3.28395  -0.03358   3.53791  65.70100
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -9.6067  1.7545    -5.4754
## dist         6.5507  0.2451    26.7237
## climb        0.0083  0.0008     9.9199
##
## Residual standard error: 5.209 on 32 degrees of freedom
```

The coefficient standard errors reported by `rlm` rely on asymptotic approximations, and may not be trustworthy in a sample of size 35. Thus, we will use bootstrapping to estimate confidence intervals.

1. Calculate a bootstrap 95% confidence interval for the coefficient of `dist` using model-based resampling.

```
modelBasedBoot <- function(n){
  fit <- rlm(time~dist+climb, data =hills, maxit=100)
  resids <- resid(fit)
  coefs <- coef(fit)
  fits.boot <- list()
  fitted.values <- predict(fit)

  for(i in 1:n){
    time.boot <- fitted.values + sample(resids, replace=TRUE)
    fits.boot[[i]] <- rlm(time.boot ~ dist+climb, data=hills, maxit=100)
   }
  return(fits.boot)
  }

bootstrap.fits <- modelBasedBoot(1000)
bootstrap.coefs <- sapply(bootstrap.fits, function(x) coef(x))

apply(bootstrap.coefs,1, sd)
```

```
## (Intercept)        dist       climb
## 2.062045846 0.394364089 0.001122638
```

```
summary(fit)
```

```
##
## Call: rlm(formula = time ~ dist + climb, data = hills)
```

```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -10.75039  -3.28395  -0.03358   3.53791  65.70100
##
## Coefficients:
##             Value   Std. Error t value
## (Intercept) -9.6067  1.7545    -5.4754
## dist         6.5507  0.2451    26.7237
## climb        0.0083  0.0008     9.9199
##
## Residual standard error: 5.209 on 32 degrees of freedom
```

```
quantile(bootstrap.coefs[2,], c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 6.111495 7.696865
```

2. Recalculate this confidence interval by now using case resampling - i.e. by bootstrap resampling (x_i, y_i), re-fitting the model to each bootstrap sample, and forming the bootstrap distribution of β. Contrast this with your answer from the previous part.

```
nrow = dim(hills)[1]
nboot <- 100
boot.coefs <-matrix(nrow=nboot, nc=3)
for(i in 1:nboot){
  index.boot <- sample(1:nrow, nrow, replace=TRUE)
  hills.boot <- hills[index.boot,]
  fit.boot <- rlm(time ~ dist+climb, data=hills.boot, maxit=100)
  boot.coefs[i,] <- coef(fit.boot)
}
quantile(boot.coefs[,2], c(0.025, 0.975))
```

```
##     2.5%    97.5%
## 5.038782 6.995763
```

The asymptotic 95% CI can be obtained using

```
6.5507 -2*0.2451
```

```
## [1] 6.0605
```

```
6.5507 +2*0.2451
```

```
## [1] 7.0409
```

3. An alternative model is proposed, which includes an interaction term between `dist` and `climb` and a quadratic `climb` term. Calculate the mean-square prediction error for both models using leave-one-out cross validation.

```
fit2 <- rlm(time~dist*climb+I(climb^2), hills)
```

```
nrow <- dim(hills)[1]
lm1.pred <- c()
lm2.pred <- c()

for(i in 1:nrow){
  hills.reduced<-hills[-i,]
  lm1<-rlm(time ~ dist+climb, data=hills.reduced, maxit=100)
  lm2 <- rlm(time~dist*climb+I(climb^2), hills.reduced, maxit=100)
  lm1.pred[i] <- predict(lm1, hills[i,])
```

```
  lm2.pred[i] <- predict(lm2, hills[i,])
}

rmse1 <- sqrt(mean((hills$time - lm1.pred)^2))
rmse2 <- sqrt(mean((hills$time - lm2.pred)^2))
print(c(rmse1, rmse2))
```

```
## [1] 15.98353 15.01099
```

So model 2 is slightly better than model 1.

Solution 2

Failure of the bootstrap: Suppose $X_1, \ldots, X_n \sim U[0, \theta]$.

1. Show that the maximum likelihood estimator of θ is

$$\hat{\theta} = \max_{i=1,\ldots,n} X_i$$

The likelihood function is

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\theta} \mathbb{I}_{0 \leq X_i \leq \theta} = \mathbb{I}_{0 \leq \min X_i \leq \max X_i \leq \theta} \frac{1}{\theta^n}$$

which is maximised at

$$\hat{\theta} = \max X_i$$

2. Calculate the CDF of $\hat{\theta}$

We can see that

$$\mathbb{P}(\hat{\theta} \leq t) = \prod \mathbb{P}(X_i \leq t) = \left(\frac{t}{\theta}\right)^n$$

3. Generate a toy dataset of size $n = 100$ (assuming that $\theta = 1$) using the code

```
set.seed(1)
n=100
X = runif(n,min=0, max=1)
```

4. The parametric bootstrap works by generating a bootstrap sample from the fitted parametric model, i.e., simulating

$$X_1^*, \ldots, X_n^* \sim U[0, \hat{\theta}],$$

and then fitting the parametric model, i.e., setting

$$\hat{\theta}^* = \max_{i=1,\ldots,n} X_i^*$$

Explain why for the parametric bootstrap

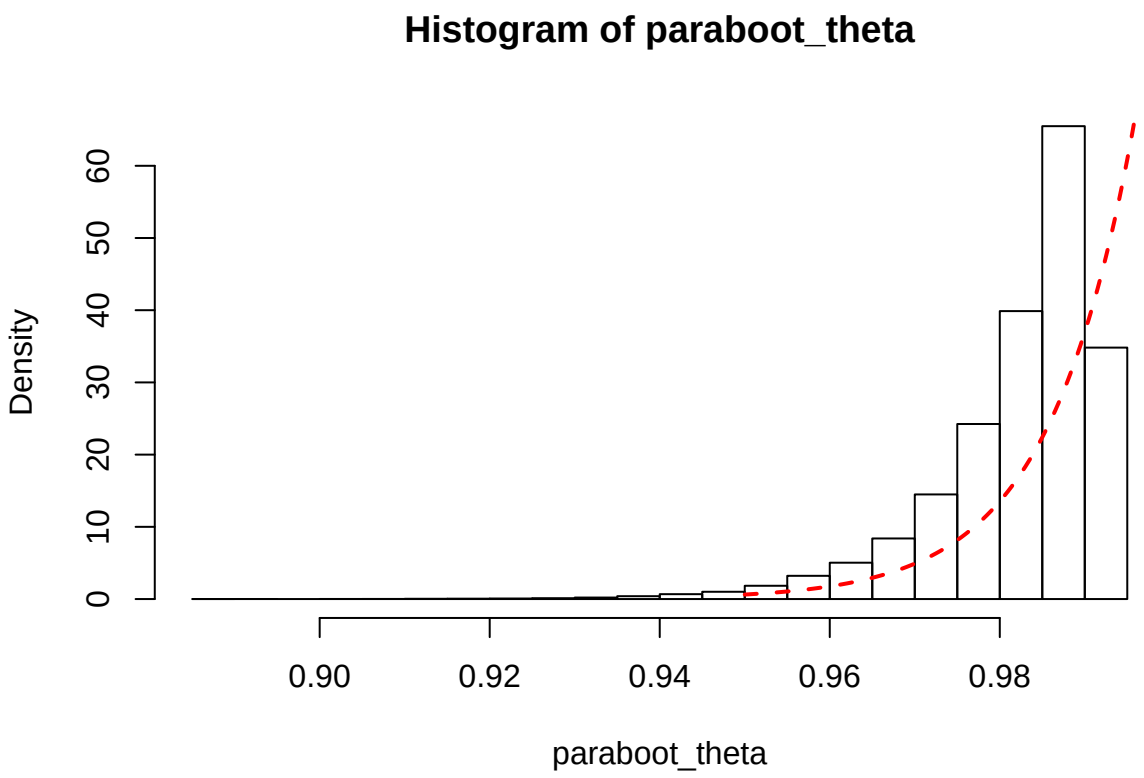
$$\mathbb{P}(\hat{\theta}^* = \hat{\theta}) = 0.$$

This is because $\hat{\theta}^* = \max(X_i)$, and the probability of any simulated value exactly hitting $\hat{\theta}$ is 0 (moreover, $P(X_i = c) = 0$ for any value of c for that matter).

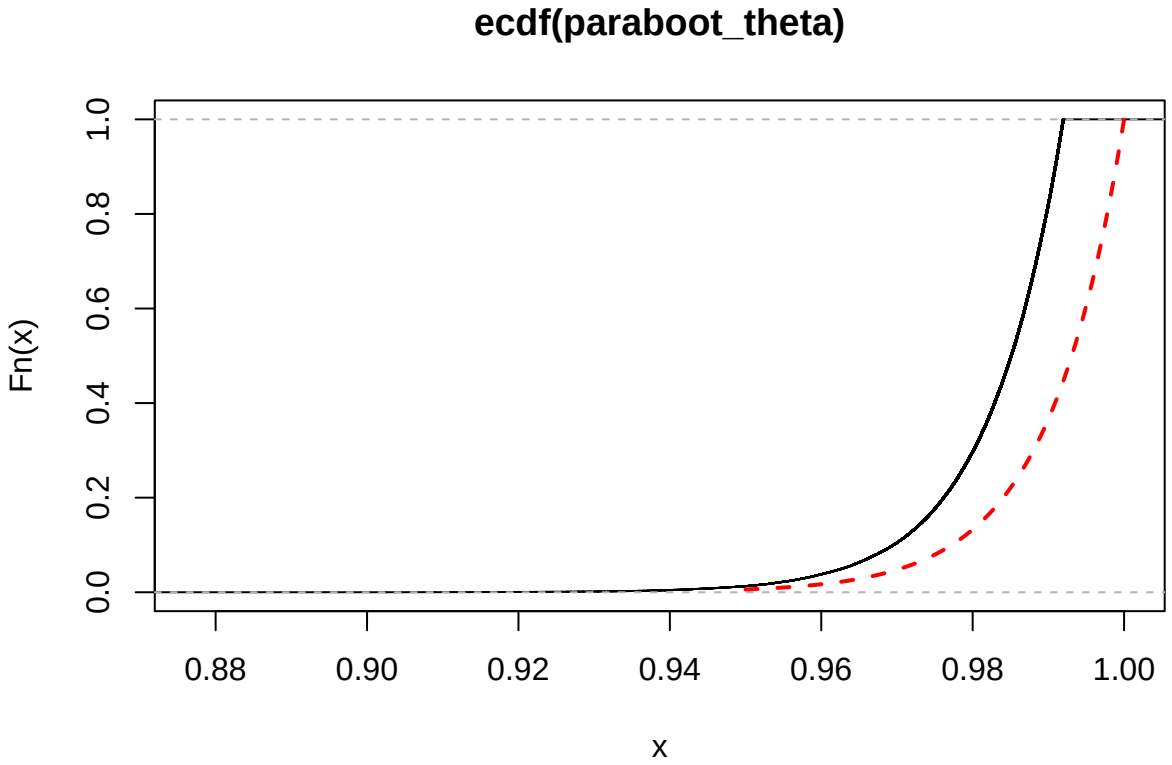
Simulate 10^5 (parametric) bootstrap replicates and compare the distribution of these with the non-parametric bootstrap estimate.

The parametric bootstrap works as follows

```r
paraboot_theta <- replicate(10^5,
                        {
                          Xrep = runif(n,0, max(X))
                          max(Xrep)
                        })
hist(paraboot_theta, probability=T)
curve(n*x^{n-1}, 0.95,1, lwd=2, lty=2, col=2, add=T)
```



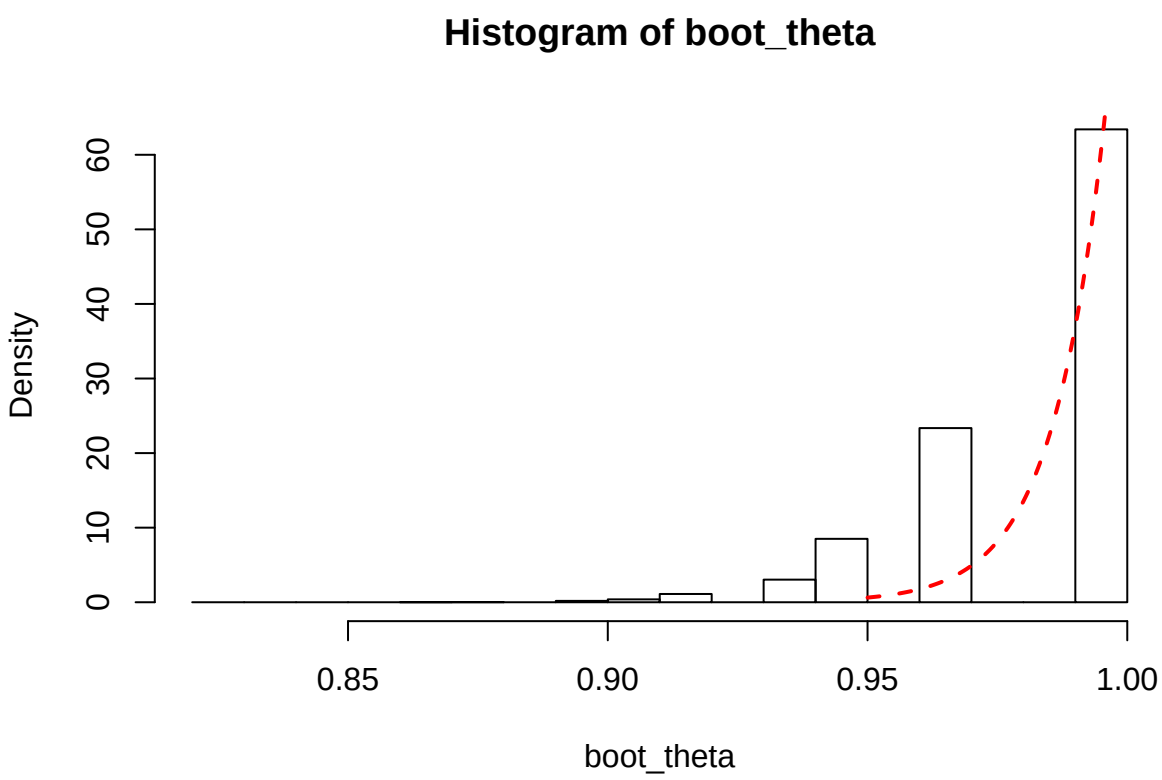

```r
plot(ecdf(paraboot_theta))
curve(x^n, 0.95,1, add=T, lwd=2, lty=2, col=2)
```



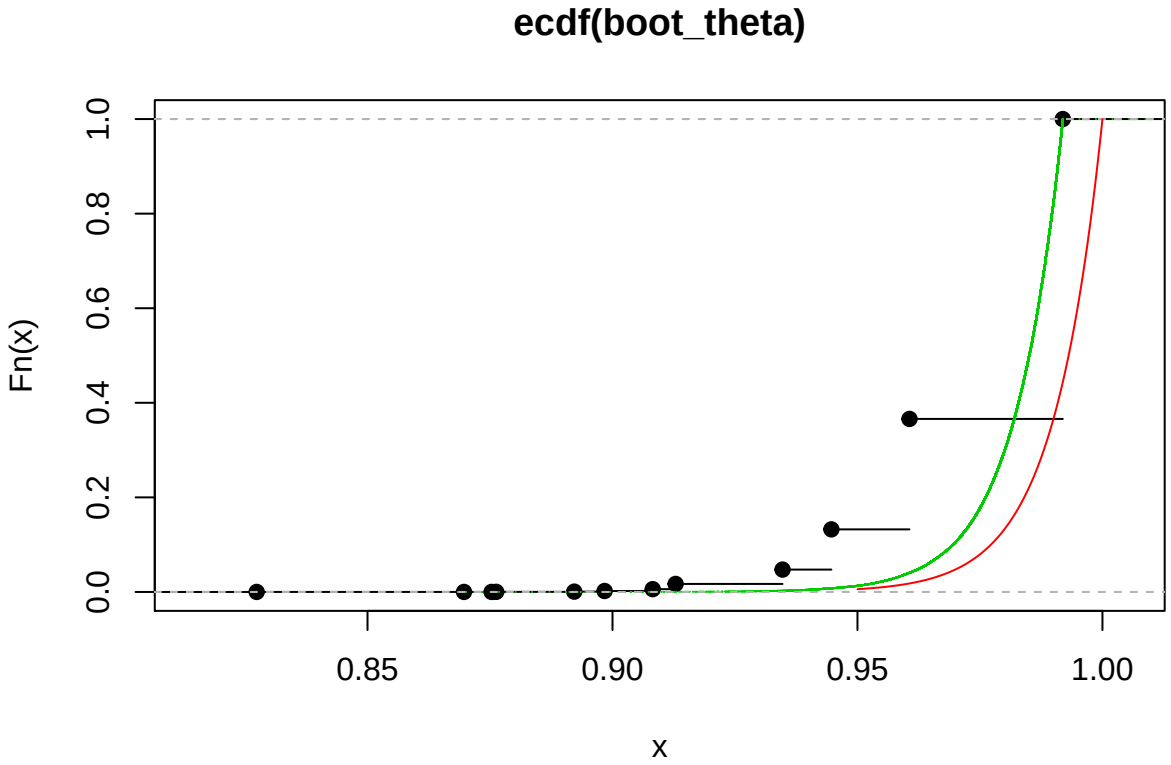
It is clear that the parametric bootstrap gives a poor approximation to the distribution of $\hat{\theta}$.

5. The non-parametric bootstrap creates bootstrap samples by sampling from the empirical CDF. Simulate 10^5 non-parametric bootstrap samples and compare the true distribution of $\hat{\theta}$ to the histogram from the non-parametric bootstrap and with the parametric bootstrap.

```
boot_theta <- replicate(10^5,
                        {
                          max(sample(X, n, replace=T))
                        })
hist(boot_theta, probability=T)
curve(n*x^{n-1}, 0.95,1, lwd=2, lty=2, col=2, add=T)
```




```
plot(ecdf(boot_theta))
plot(ecdf(paraboot_theta), add=T, col=3, lty=3)
curve(x^n, 0.95,1, add=T, col=2)
```



Again, we can see that the non-parametric bootstrap is also a poor estimator of the distribution.

6. If $\hat{\theta}^*$ is a bootstrap estimate of $\hat{\theta}$, show that

$$\mathbb{P}(\hat{\theta}^* = \hat{\theta}) = 1 - (1 - (1/n))^n$$

and hence that

$$\mathbb{P}(\hat{\theta}^* = \hat{\theta}) \to 1 - e^{-1} \approx 0.632$$

as $n \to \infty$.

Suppose X_M is the largest value of $X_1, \ldots, X_n$.

$$\begin{aligned}
\mathbb{P}(\hat{\theta}^* = \hat{\theta}) &= P(X_M \text{is resampled}) \\
&= 1 - P(X_M \text{is not resampled}) \\
&= 1 - \prod_{i=1}^{n} P(X^*{}_i \neq X_M) \\
&= 1 - (1 - 1/n)^n
\end{aligned}$$

where X_i^* is the ith resampled value.

Finally, recall that

$$(1 + x/n)^n \to e^x$$

which gives the required result.