Prediction Interval

Project

2023-03-30

Method

We introduce four methods to generate the prediction interval.

Let $X_1, ..., X_n$ be a simple random sample from a distribution with continuous cumulative distribution function F. For a given nominal error rate α, a $100((1-\alpha)\%$ prediction interval for an independent future observation X_{n+1} from the same distribution is a random interval $(L(X), U(X))$ such that

$$\inf_{F\in\mathcal{F}} P(L(X) < X_{n+1} < U(X)) \geq 1-\alpha$$

where $\mathcal{F}$ is the set of distributions under consideration, $L(X)$ is the lower bound of the interval and $U(X)$ is the upper bound of the interval.

We would like to use different methods to find $(L(X), U(X))$.

Prediction interval with the choosen position

If we choose a random β, we can find the left tail x_1 from the random sample such that $P(X < x_1) \leq \beta$ and right tail is x_2 from the random sample such that $P(X < x_2) \geq \beta + (1-\alpha)$.

Shortest prediction interval

Find the shortest intervals (x_1, x_2) such that such that $P(X < x_1) \leq \beta$ and right tail is x_2 from the random sample such that $P(X < x_2) \geq \beta + (1-\alpha)$, where $\beta \in (0, \alpha)$.

Prediction interval using cross validation

In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. Of the k subsamples, a single subsample is retained as the validation data for testing the model, and the remaining $k-1$ subsamples are used as training data. Find the shortest prediction interval with a coverage probability at least $100(1-\alpha)\%$ using training data set and evaluate its performance on the testing data set. The chosen prediction interval is the one having a more accurate coverage probability on the testing data set.

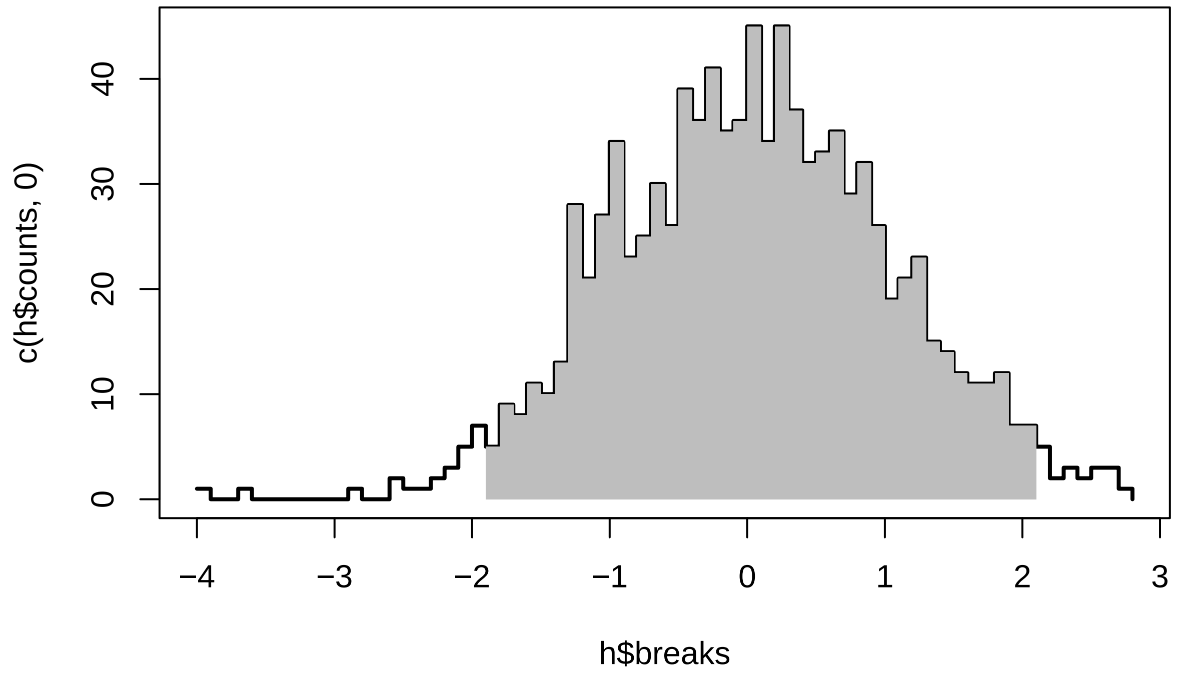
Data Driven Prediction interval

The data driven prediction interval is obtained by choosing the shortest of the intervals $(X_{(r)}, X_{(s)})$ for $r = 1, ..., n-k$, where $k = n(1-\alpha) + 1.12\sqrt{n\alpha}$. For $n\alpha$ large, this interval gives exact confidence coefficient $1-\alpha$.

Performance evaluated using simulated data

Getting the prediction interval for one simulated data set

```r
# more details in ?get.interval
# Example: generate the samples from standard normal distribution
# use cross validation method to get the prediction interval using the generated sample
res = get.interval(METHOD = 3, DIST = "Normal", n = 1000)

# things needed for the plot below
lower_q = res$interval[1]
upper_q = res$interval[2]
data = res$data
# A histogram of the sample with a shaded area representing the range of the prediction interval
# inputs: sample data, lower bound of the prediction interval, upper bound of the prediction interval
# output: a plot
h = hist(data, breaks=50, plot=FALSE)
cuts = cut(h$breaks, c(lower_q, upper_q))
plot(h$breaks, c(h$counts,0) ,type="s",col="black", lwd=2)
plot(h, col="gray"[cuts], lty="blank", add=T)
```

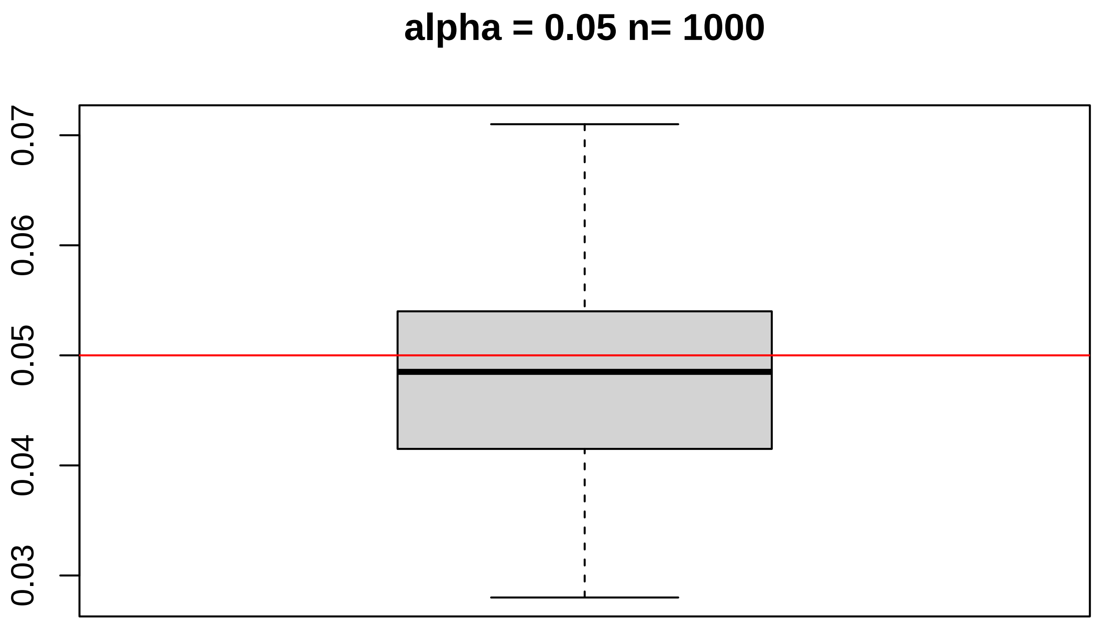


Comparison of the coverage

We conduct 100 experiments. For each experiment, we get the prediction interval using simulated data and get the coverage probability of the the prediction interval on the test data generated from the same distribution as the simulated data. The boxplot of 100 coverage probabilities is shown.

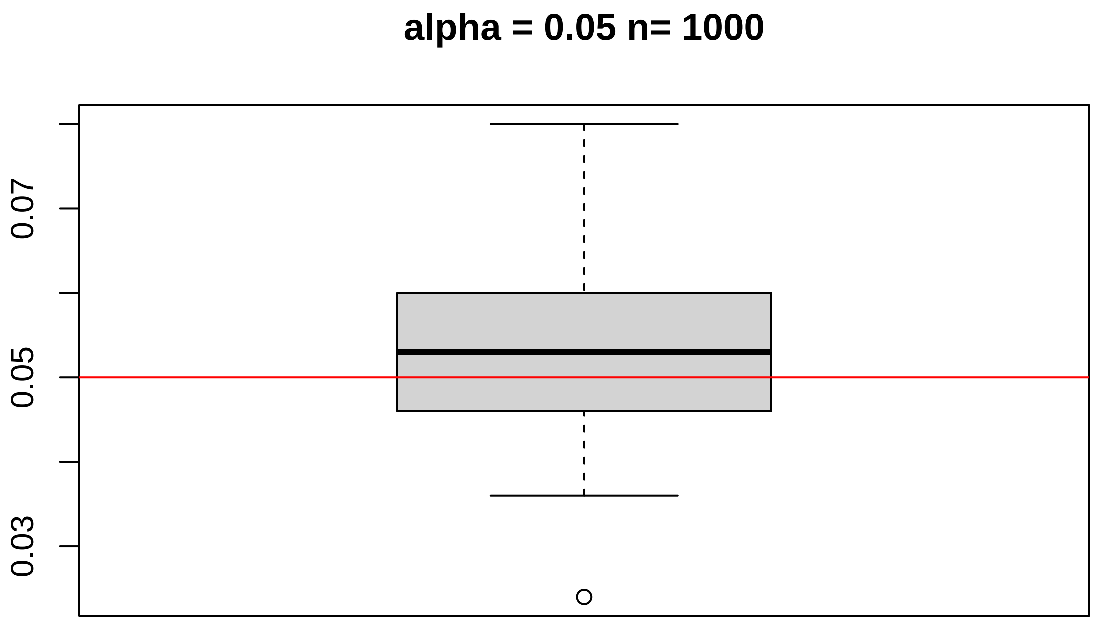
Prediction interval with random positions In each experiment, we use a random chosen position. It's not going to be the shortest, but it's valid in coverage.

```r
n = 1000 ## training sample size
n0 = 1000 ## test sample size
alpha = 0.05 ## nonimal error rate
n_exp = 100 # number of experiments
cov.box = vector(mode="double", length=n_exp)
for(i in 1:n_exp){
  beta = runif(1, min = 0, max = alpha)

  res = get.interval(METHOD = 1, DIST = "Normal", n = n, test_n = n0, beta = beta, test_data = T)
  x = res$data
  x0 = res$test_dataset
  lower_q = res$interval[1]
  upper_q = res$interval[2]

  coverage.x0 = 1-mean(lower_q<= x0 & upper_q >=x0)
  cov.box[i] = coverage.x0
}

boxplot(cov.box, main = paste(expression(alpha), '=', alpha, 'n=', n))
abline(h = alpha, col = 'red')
```



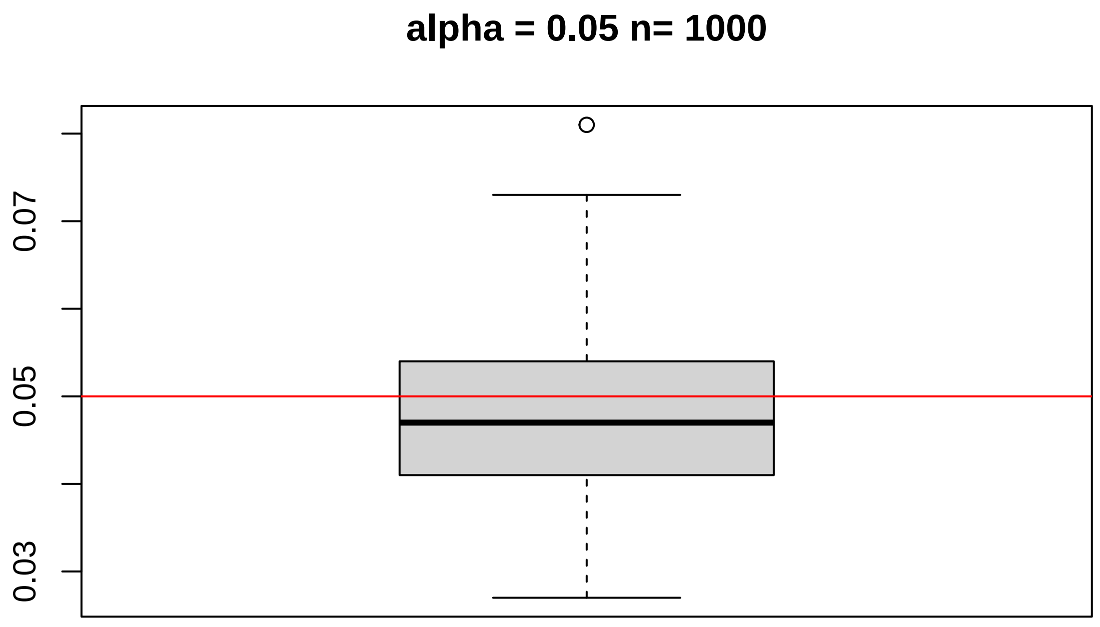
Shortest prediction interval With no correction on the nominal error rate, the prediction interval is going to undercover:

```r
n = 1000 ## training sample size
n0 = 1000 ## test sample size
alpha = 0.05 ## nonimal error rate
n_exp = 100 # number of experiments
cov.box = vector(mode="double", length=n_exp)
for(i in 1:n_exp){
  res = get.interval(METHOD = 2, DIST = "Normal", n = n, test_n = n0, test_data = T)
  x = res$data
  x0 = res$test_dataset
  shortest_lower_q = res$interval[1]
  shortest_upper_q = res$interval[2]
  coverage.x0 = 1-mean(shortest_lower_q<= x0 & shortest_upper_q >=x0)
  cov.box[i] = coverage.x0
}

boxplot(cov.box, main = paste(expression(alpha), '=', alpha, 'n=', n))
abline(h = alpha, col = 'red')
```



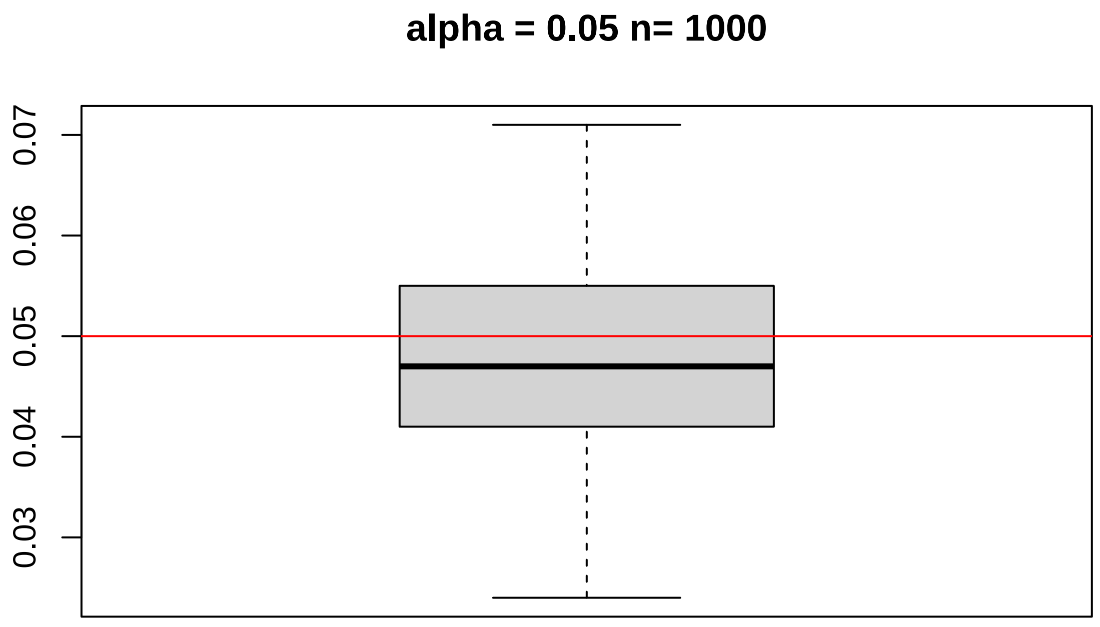
Prediction interval using cross validation

```
K = 2
n = 1000 ## training sample size
n0 = 1000 ## test sample size
alpha = 0.05 ## nonimal error rate
n_exp = 100 # number of experiments
cov.box = vector(mode="double", length=n_exp)
for(i in 1:n_exp){
  res = get.interval(METHOD = 3, DIST = "Normal", n = n, test_n = n0, K=K, test_data = T)
  x = res$data
  x0 = res$test_dataset
  cv_lower_q = res$interval[1]
  cv_upper_q = res$interval[2]

  coverage.x0 = 1-mean(cv_lower_q<= x0 & cv_upper_q >=x0)
  cov.box[i] = coverage.x0
}

boxplot(cov.box, main = paste(expression(alpha), '=', alpha, 'n=', n))
abline(h = alpha, col = 'red')
```



Data Driven Prediction interval

```
n = 1000 ## training sample size
n0 = 1000 ## test sample size
alpha = 0.05 ## nonimal error rate
n_exp = 100 # number of experiments
cov.box = vector(mode="double", length=n_exp)
for(i in 1:n_exp){
  res = get.interval(METHOD = 4, DIST = "Normal", n = n, test_n = n0, test_data = T)
  x = res$data
  x0 = res$test_dataset
  dv_lower_q = res$interval[1]
  dv_upper_q = res$interval[2]

  coverage.x0 = 1-mean(dv_lower_q<= x0 & dv_upper_q >=x0)
  cov.box[i] = coverage.x0
}

boxplot(cov.box, main = paste(expression(alpha), '=', alpha, 'n=', n))
abline(h = alpha, col = 'red')
```


alpha = 0.05 n= 1000
0.03
0.04
0.05
0.06
0.07