Nonparametric Bootstrapping

Quentin F. Gronau

University of Amsterdam

January, 12, 2016

Overview

Origin of the Term "Bootstrap"

- derives from "to pull oneself up by one's bootstrap"
- widely thought to be based on 18th century adventures of Baron Munchausen by R. E. Raspe
- Baron had fallen to bottom of deep lake and rescued himself by picking up himself by his own bootstraps
- however, it appears that this is a misattribution: in the original story, he picks up himself by his hair

Bootstrapping in Statistics

Efron & Tibshirani (1994), *An Introduction to the Bootstrap*:

> *The bootstrap is a computer-based method for assigning measures of accuracy to statistical estimates.*

Bootstrapping in Statistics

Efron & Tibshirani (1994), *An Introduction to the Bootstrap*:

> *The bootstrap is a computer-based method for assigning measures of accuracy to statistical estimates.*

- in principle available no matter how mathematically complicated the estimator is
- when it is very hard or impossible to obtain formulas for standard error of estimator analytically, we can still use bootstrapping to assess the accuracy of the estimator

Bootstrapping in Statistics

Efron & Tibshirani (1994), *An Introduction to the Bootstrap*:

> *The bootstrap is a computer-based method for assigning measures of accuracy to statistical estimates.*

- in principle available no matter how mathematically complicated the estimator is
- when it is very hard or impossible to obtain formulas for standard error of estimator analytically, we can still use bootstrapping to assess the accuracy of the estimator
- to motivate the bootstrapping algorithm, we will briefly review the "plug-in" principle

The "Plug-In" Principle

- in statistics, we are interested in population parameters which are functions of the population distribution F (F denotes the population cumulative distribution function)

The "Plug-In" Principle

- in statistics, we are interested in population parameters which are functions of the population distribution F (F denotes the population cumulative distribution function)
- For instance, we are interested in the population mean

$$\mu = \mathbb{E}_F(X) = \int x \, \mathrm{d}F(x). \tag{1}$$

The "Plug-In" Principle

- in statistics, we are interested in population parameters which are functions of the population distribution F (F denotes the population cumulative distribution function)
- For instance, we are interested in the population mean

$$\mu = \mathbb{E}_F(X) = \int x \, \mathrm{d}F(x). \tag{1}$$

- this is the Riemann-Stieltjes integral notation
- if cdf F differentiable everywhere, i.e., a pdf f exists, equivalent to $\int x \, f(x) \mathrm{d}x$
- advantage of the Riemann-Stieltjes integral notation: also holds when only valid cdf, but no pdf which will be handy for the following

The "Plug-In" Principle: Continued 1

More generally, most quantities we are interested in can be written in the form:

$$\mathbb{E}_F(g(x)) = \int g(x)\,\mathrm{d}F(x). \tag{2}$$

The "Plug-In" Principle: Continued 2

- we do not know F, the population distribution, in real applications

The "Plug-In" Principle: Continued 2

- we do not know F, the population distribution, in real applications
- "plug-in" principle simply replaces the unknown distribution F with the empirical distribution function $\hat{F}_n$ which assigns probability $\frac{1}{n}$ to each observed data point x_i

$$\hat{F}_n(x) = \frac{\text{number of } x_i \leq x}{n} = \frac{1}{n}\sum_{i=1}^{n} I(x_i \leq x). \tag{3}$$

The "Plug-In" Principle: Continued 3

Then the estimator for the population mean is:

$$\bar{x} = \mathbb{E}_{\hat{F}_n}(x) = \int x \, \mathrm{d}\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{4}$$

The "Plug-In" Principle: Continued 3

Then the estimator for the population mean is:

$$\bar{x} = \mathbb{E}_{\hat{F}_n}(x) = \int x \, \mathrm{d}\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{4}$$

More generally:

$$\mathbb{E}_{\hat{F}_n}(g(x)) = \int g(x) \, \mathrm{d}\hat{F}_n(x) = \frac{1}{n}\sum_{i=1}^{n} g(x_i). \tag{5}$$

Bootstrapping: Using the "Plug-In" Principle to Approximate the Sampling Distribution

- to assess accuracy of estimator, we need to know its sampling distribution

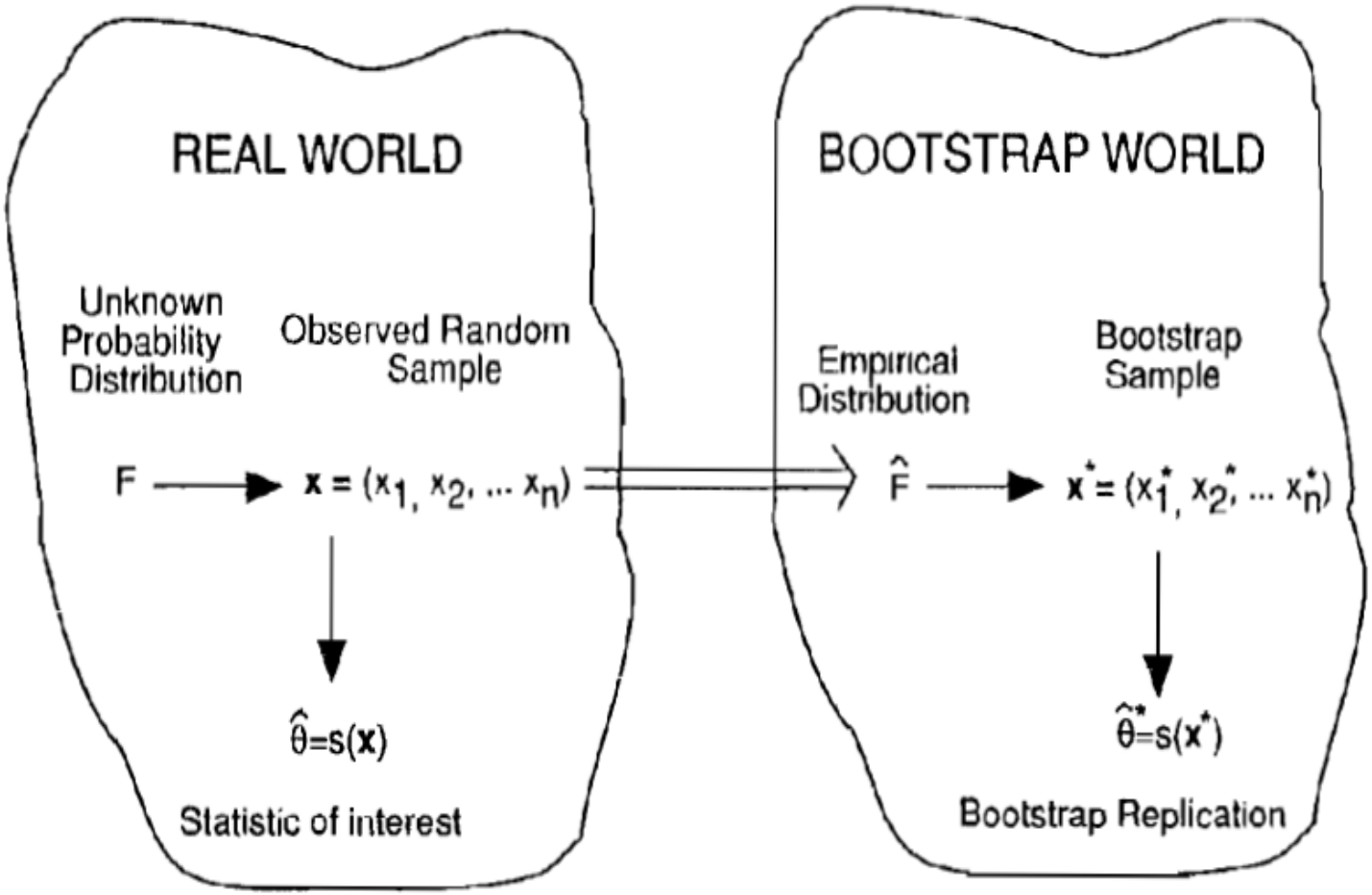
Bootstrapping: Using the "Plug-In" Principle to Approximate the Sampling Distribution

- to assess accuracy of estimator, we need to know its sampling distribution
- suppose we knew population distribution F
- then, we could construct sampling distribution by repeatedly sampling from the population and calculating the statistic of interest for each sample

Bootstrapping: Using the "Plug-In" Principle to Approximate the Sampling Distribution

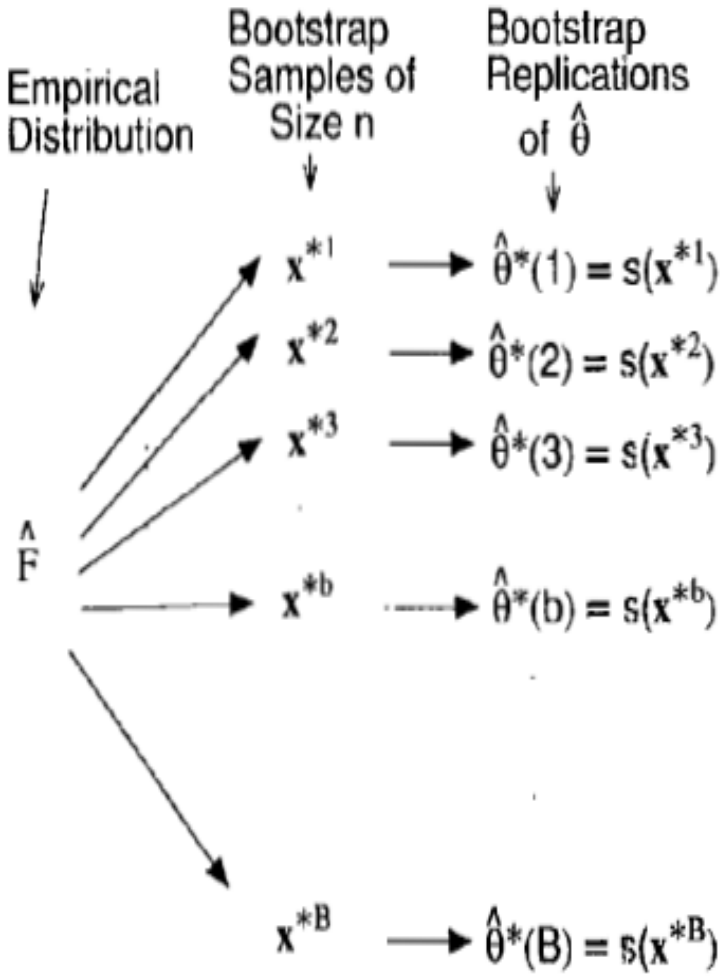
- to assess accuracy of estimator, we need to know its sampling distribution
- suppose we knew population distribution F
- then, we could construct sampling distribution by repeatedly sampling from the population and calculating the statistic of interest for each sample
- bootstrapping simply replaces the population distribution F by the empirical distribution $\hat{F}_n$ ("plug-in" principle)

Bootstrapping: Using the "Plug-In" Principle to Approximate the Sampling Distribution Continued



The Bootstrap Algorithm

1. Obtain B independent bootstrap samples each of size n by sampling from $\hat{F}_n$. This simply means, sampling *with replacement* from the observed data values.
2. Calculate the statistic of interest for each of the B bootstrap samples.



Why does it work? – A Little Intuition

- why possible to replace unknown population distribution F with empirical distribution $\hat{F}_n$?

Why does it work? – A Little Intuition

- why possible to replace unknown population distribution F with empirical distribution $\hat{F}_n$?
- one reason: has been proven that empirical distribution function $\hat{F}_n$ converges to population distribution F

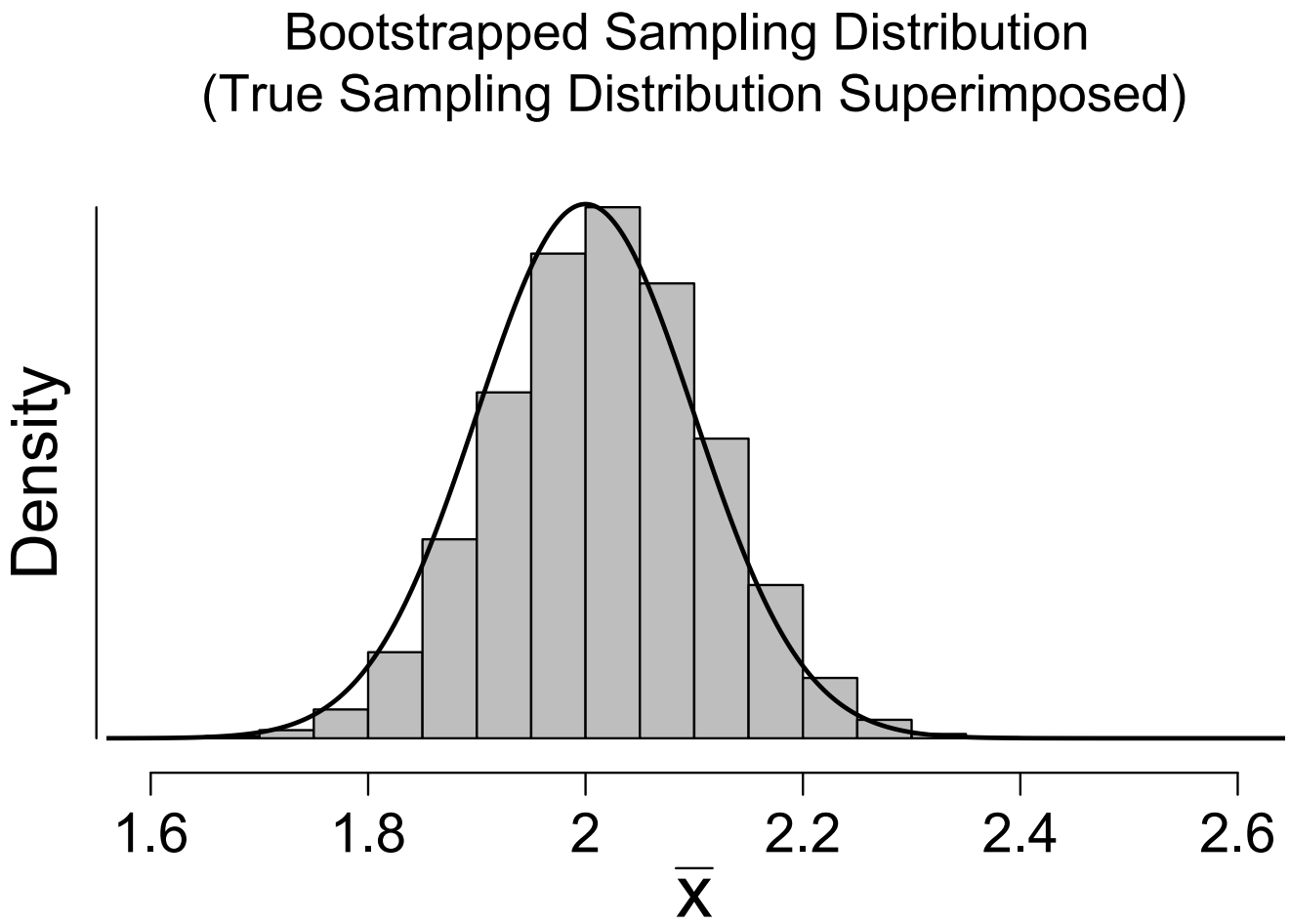
Theorem (Glivenko-Cantelli theorem)

$\sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)| \longrightarrow 0$ *almost surely as* $n \longrightarrow \infty$.

A Toy Example: Bootstrapping the Mean

```r
### generate data ###

set.seed(123456)

n <- 100 # number of observations
data <- rnorm(n, mean = 2)

### bootstrapping ###

B <- 100000 # number of bootstrap replicates

boot.means <- numeric(B)

for (i in seq_len(B)) {

        indices <- sample(seq_len(n), n, replace = TRUE)
        boot.data <- data[indices]
        boot.means[i] <- mean(boot.data)

}
```

A Toy Example: Bootstrapping the Mean Continued



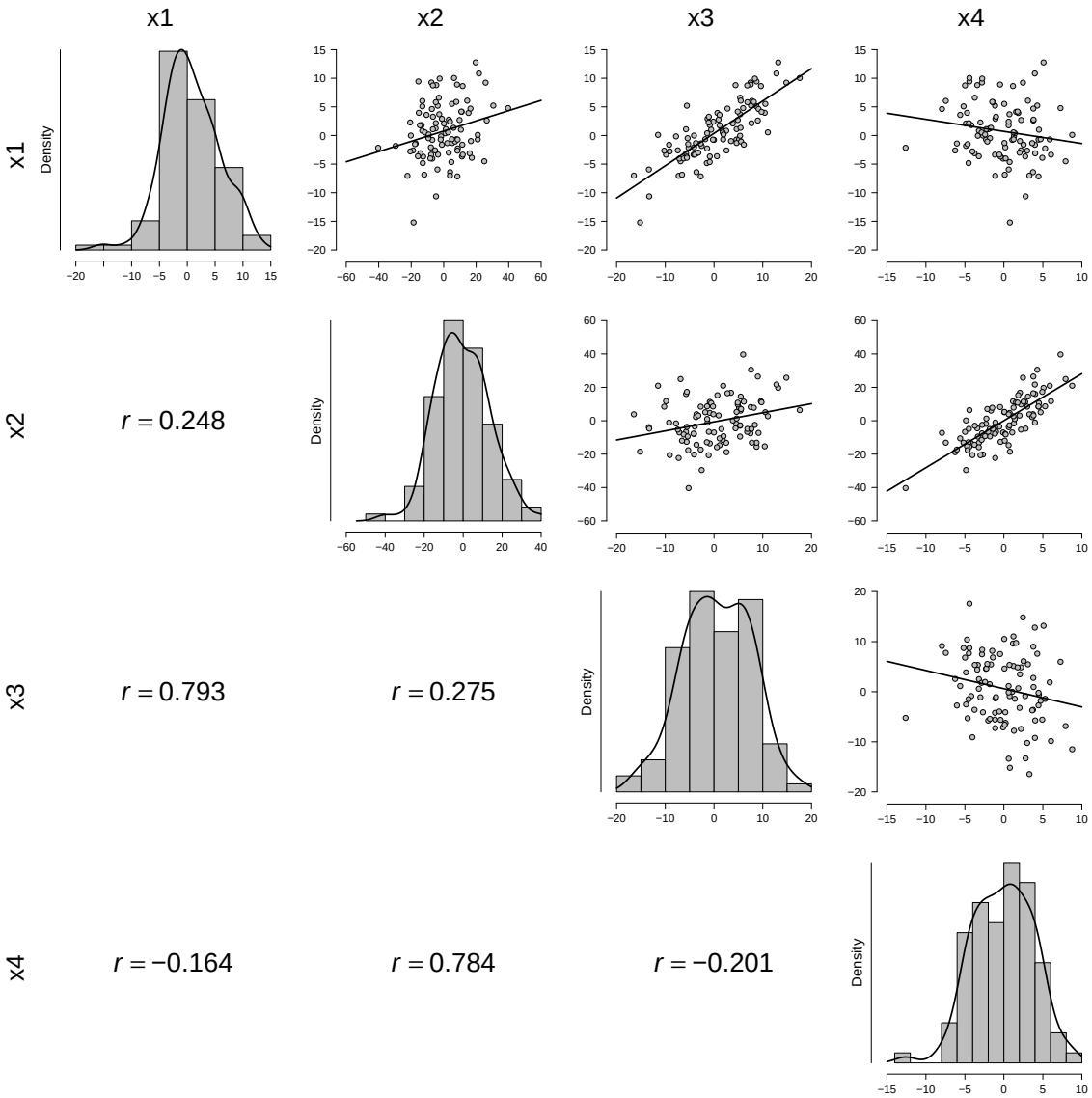
Bootstrapping Confidence Intervals in Principle Component Analysis

- assume we are interested in assessing accurracy of proportion of variance explained by principle components via confidence intervals
- to the best of my knowledge, no easy formula for doing this exists
- we can use bootstrapping to obtain approximate confidence intervals

Data Generation

```r
### generate data ###

library(mvtnorm)

# standard deviations

sds <- c(5, 15, 7, 4)

# correlation matrix

R <- matrix(c(

          1, .2,  .8, -.2,
         .2,  1,  .2,  .8,
         .8, .2,   1, -.2,
        -.2, .8, -.2,   1

        ), 4, 4, byrow = TRUE)

# covariance matrix

Sigma <- diag(sds) %*% R %*% diag(sds)

# data

set.seed(2343457)

n <- 100
data <- rmvnorm(n, mean = rep(0, 4), sigma = Sigma)
```

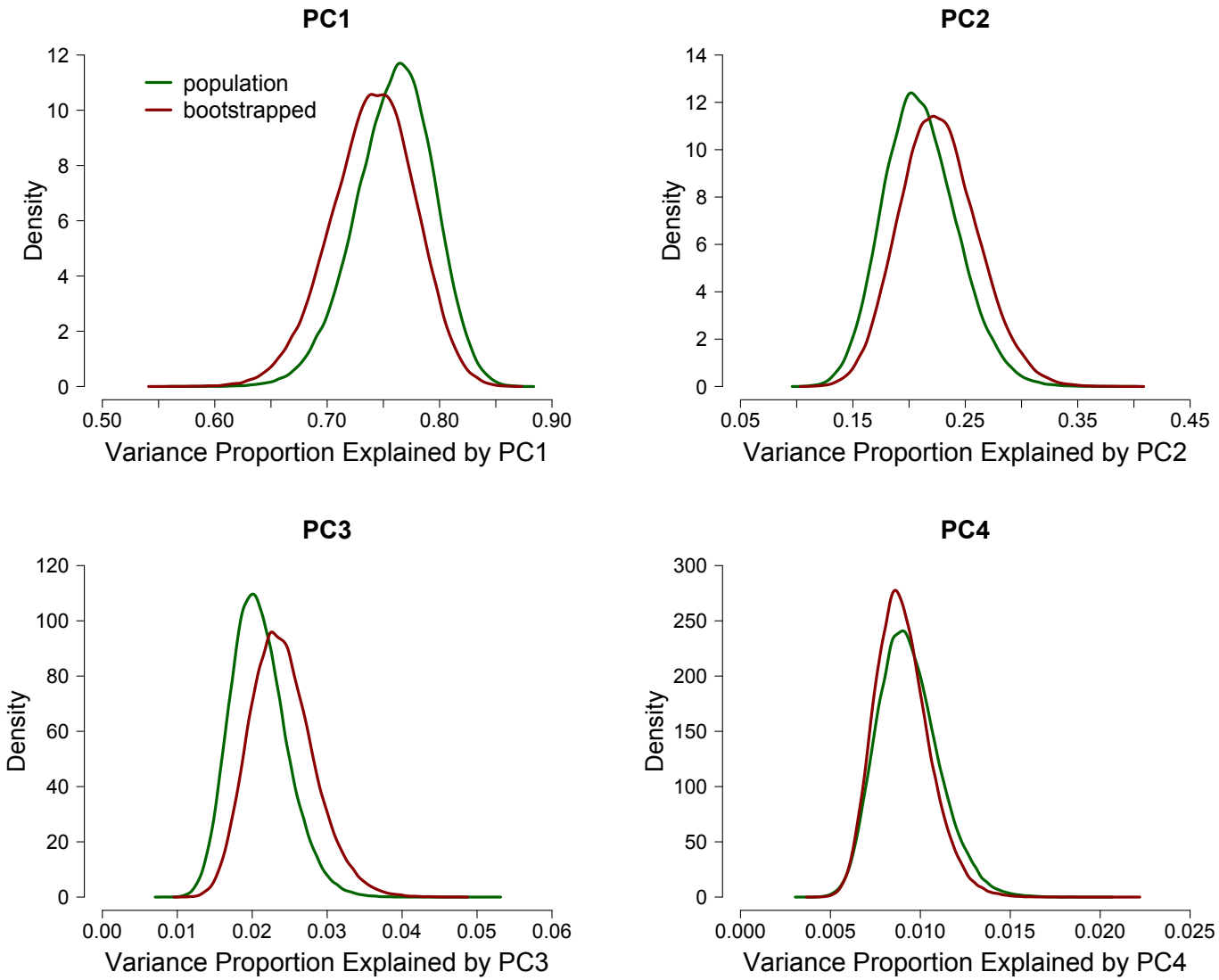
Data Visualization



Bootstrapping: R Code

```
### bootstrapping ###

library(boot)

B <- 100000 # number of bootstrap replicates

boot.func <- function(data, indices) {

        pca <- prcomp(data[indices, ])

        summary(pca)$importance["Proportion of Variance", ]

}

boot.result <- boot(data, boot.func, B)
boot.samples <- boot.result$t
```

Population and Bootstrapped Sampling Distribution



Eigendecomposition of Covariance Matrix to Obtain True Variance Proportions

$$\mathbf{\Sigma} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{\mathsf{T}} = \sum_i \lambda_i \mathbf{v}_i \mathbf{v}_i^{\mathsf{T}} \tag{6}$$

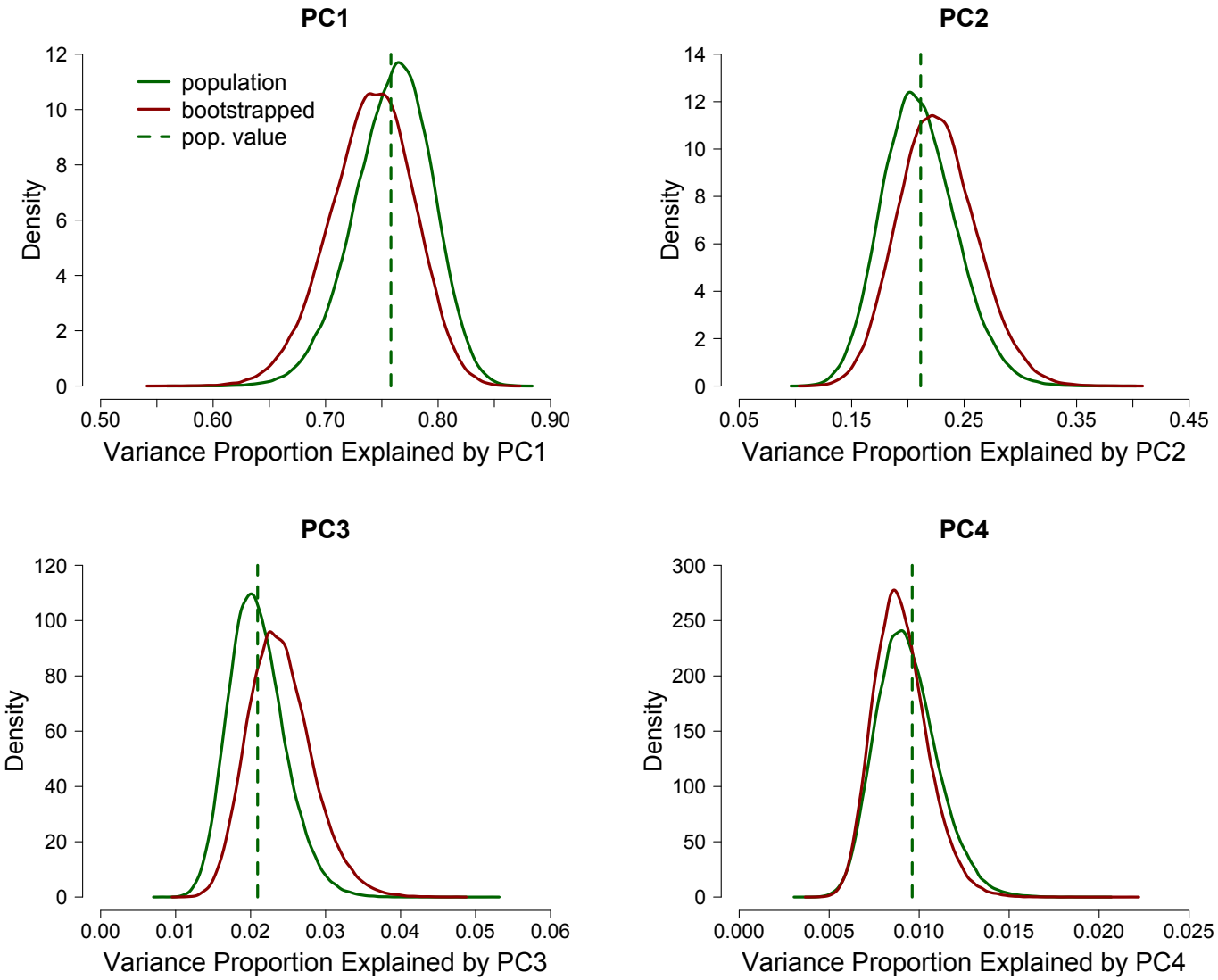
Eigenvalues λ_i correspond to variances explained by different components. Hence, proportion of variance explained by i-th component is given by

$$\frac{\lambda_i}{\sum_i \lambda_i}. \tag{7}$$

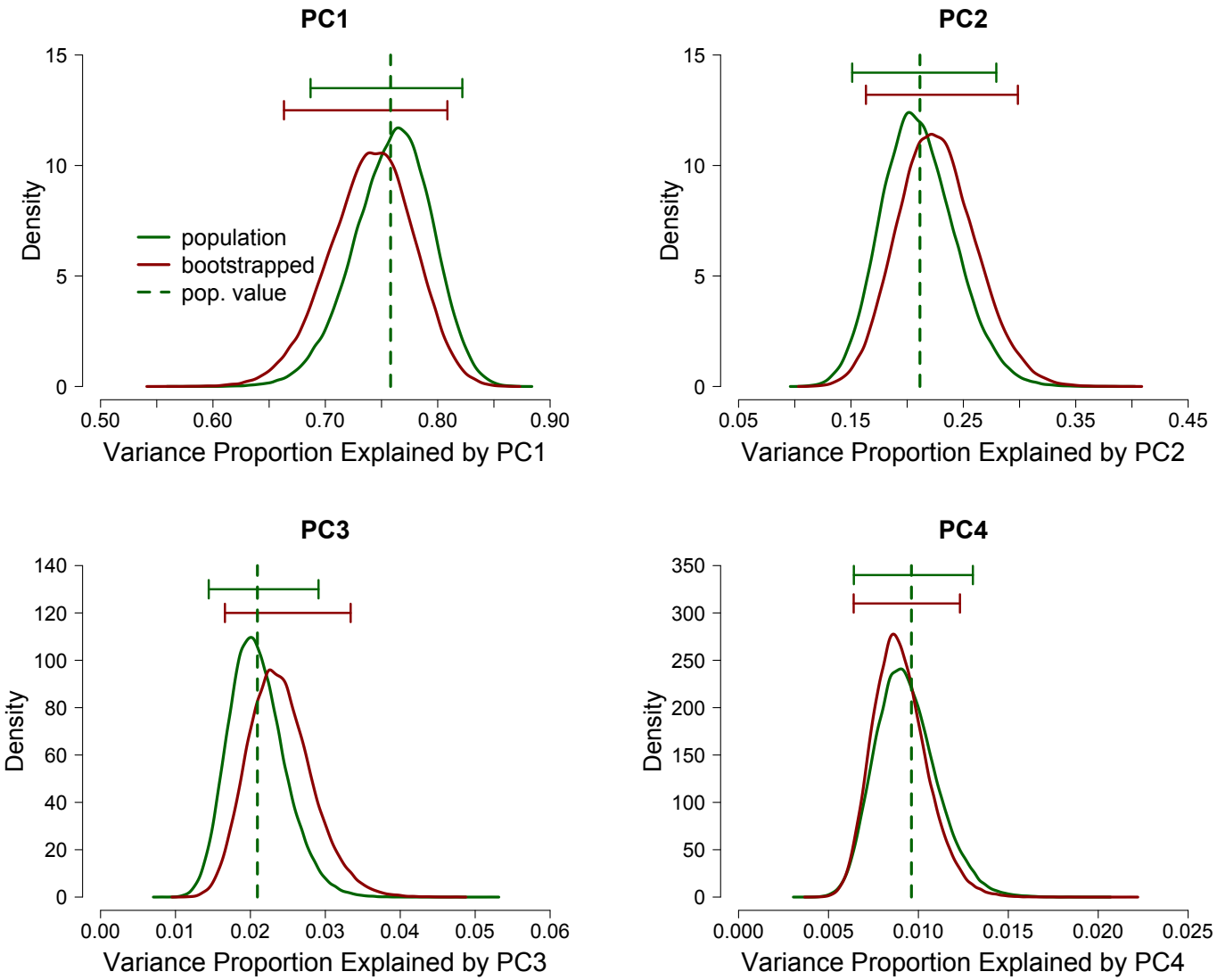
```
eigen(Sigma)$values / sum(eigen(Sigma)$values)
```

```
## [1] 0.758087539 0.211362100 0.020937298 0.009613063
```

Population and Bootstrapped Sampling Distribution with "True" Value



Population CI and Percentile Bootstrapped CI



Bootstrapping Confidence Intervals in Principle Component Analysis: BC_a Intervals

Also use percentiles of bootstrap distribution, but they are *acceleration* and *bias*-corrected.

$$p_1 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{\alpha/2}}{1 - \hat{\alpha}(\hat{z}_0 + z^{\alpha/2})}\right) \tag{8}$$

$$p_2 = \Phi\left(\hat{z}_0 + \frac{\hat{z}_0 + z^{1-\alpha/2}}{1 - \hat{\alpha}(\hat{z}_0 + z^{1-\alpha/2})}\right) \tag{9}$$

where Φ is the standard normal cdf and z^{α} is the 100αth percentile of a standard normal distribution. $\hat{\alpha}$ is the acceleration and $\hat{z}_0$ is the bias correction.

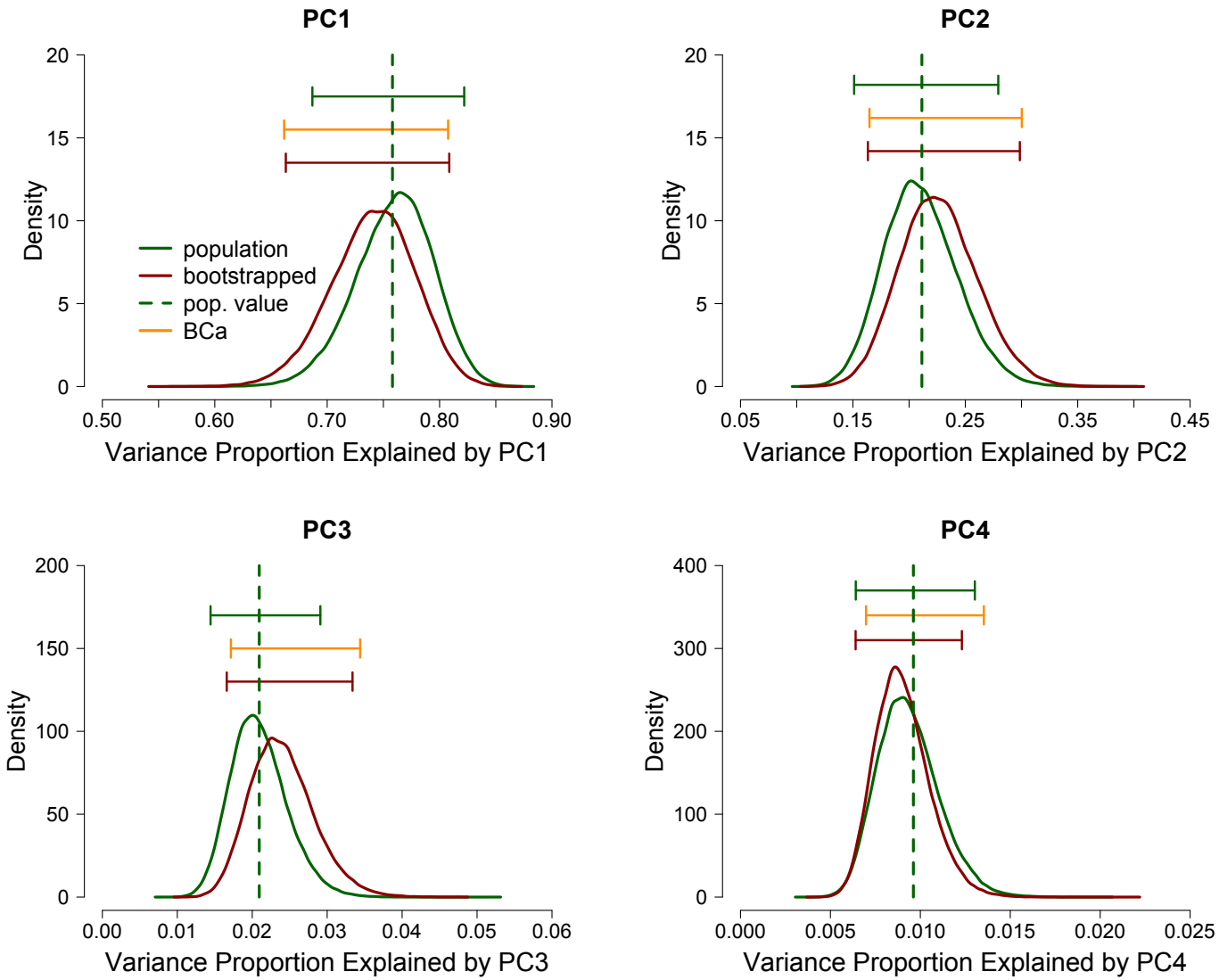
Bootstrapping Confidence Intervals in Principle Component Analysis: BC_a Intervals Continued

- *acceleration* $\hat{\alpha}$ corrects for fact that standard error might be different for different population values
- $\hat{z}_0$ corrects for *bias* by considering proportion of bootstrap samples smaller than original estimate
- if 50% of bootstrapped values smaller than original estimate, no bias correction

In R:

```
# bias-corrected and accelerated bootstrapped confidence intervals

boot.ci.bca <- matrix(ncol = 4, nrow = 2)

boot.ci.bca[ ,1] <- boot.ci(boot.out = boot.result, type = "bca", index = 1)$bca[4:5]
boot.ci.bca[ ,2] <- boot.ci(boot.out = boot.result, type = "bca", index = 2)$bca[4:5]
boot.ci.bca[ ,3] <- boot.ci(boot.out = boot.result, type = "bca", index = 3)$bca[4:5]
boot.ci.bca[ ,4] <- boot.ci(boot.out = boot.result, type = "bca", index = 4)$bca[4:5]
```

Population CI, Percentile Bootstrapped CI, and BC_a CI




Thanks!