Bootstrap method

CMED6040 – Session 2

Tim Tsang (matklab@hku.hk)

School of Public Health
The University of Hong Kong

16 May 2023

Session 2 learning objectives

After this session, students should be able to

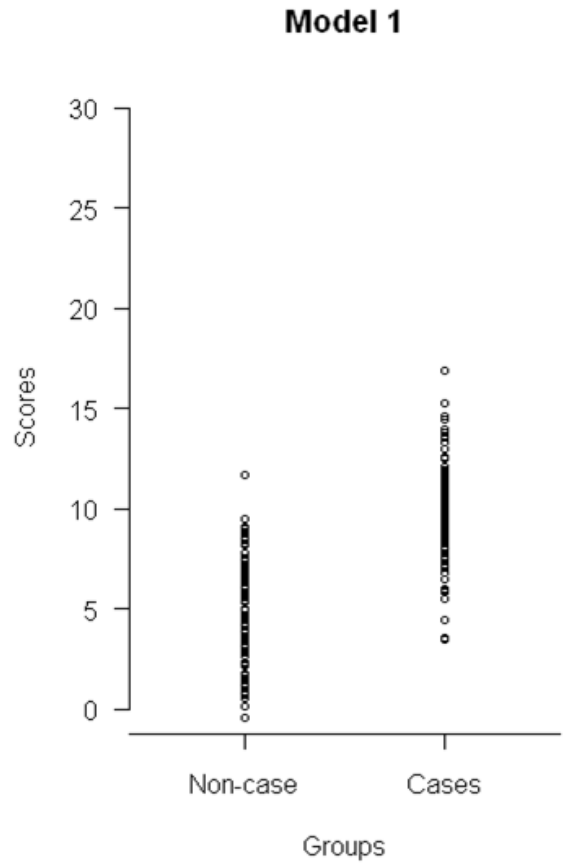
- Understand the mechanism of bootstrap method
- Generate various statistical estimates using the bootstrap method
- Recognize different types of bootstrap methods

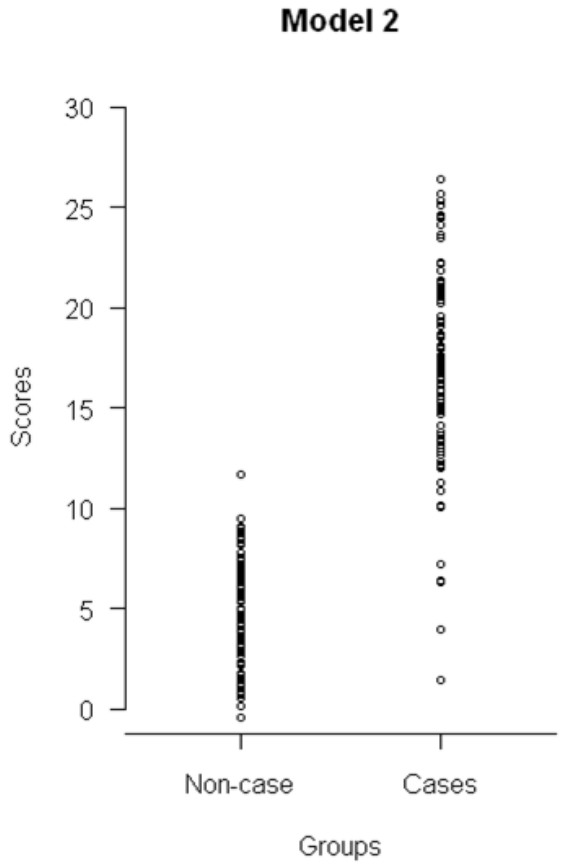
Motivating example – area under ROC curve

- Receiver operating characteristics (ROC) curve
- Measure of discriminative performance of a model
- How accurately does the model distinguish the cases from the controls / the deaths from the survivors / the positives from the negatives?

Example

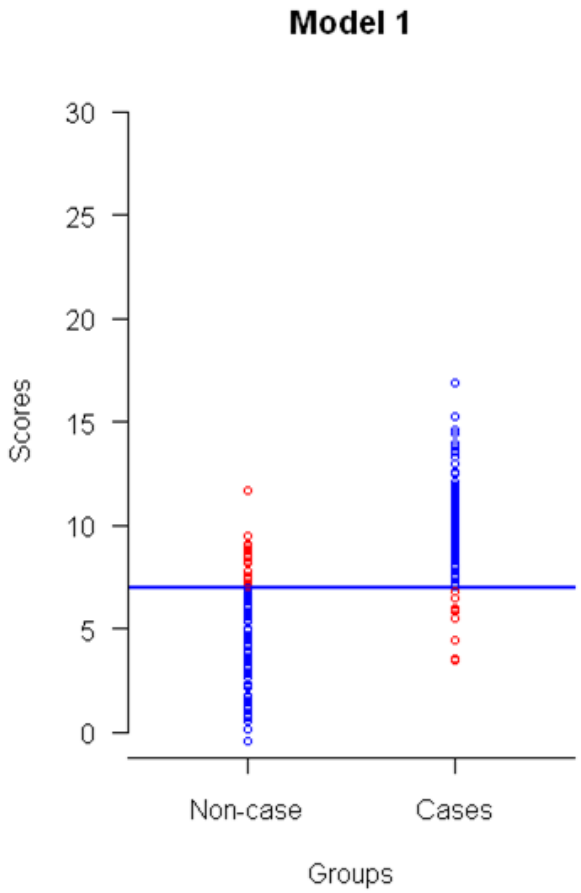
Say there are two models which give “scores” of some sort, where cases generally have higher scores than non-cases.

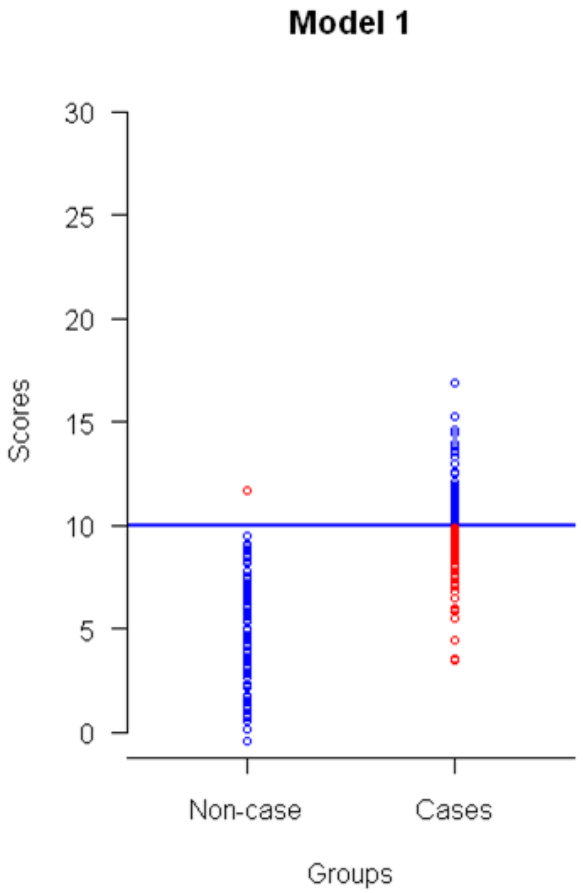




Example

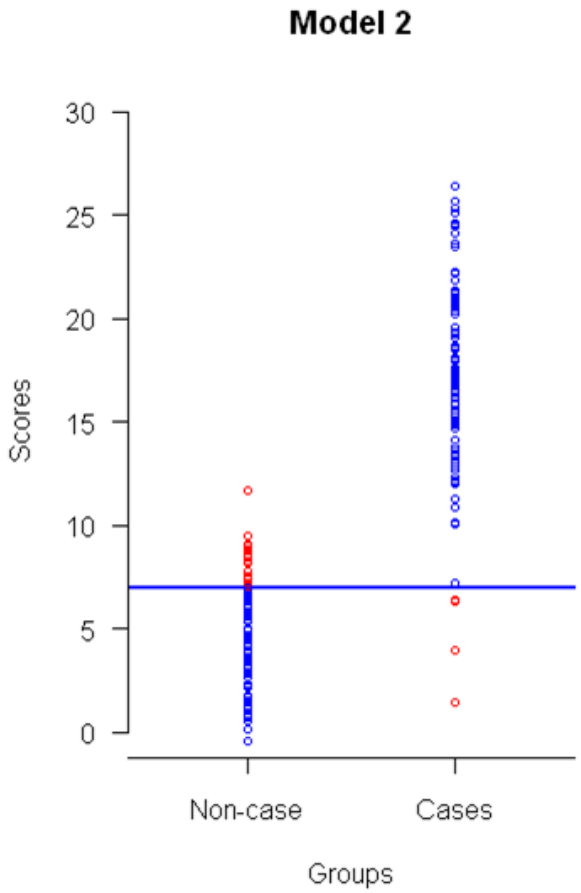
If we are only given a subject's score, how can we use a model to determine whether the subject is a case or not? With a threshold? Which threshold?

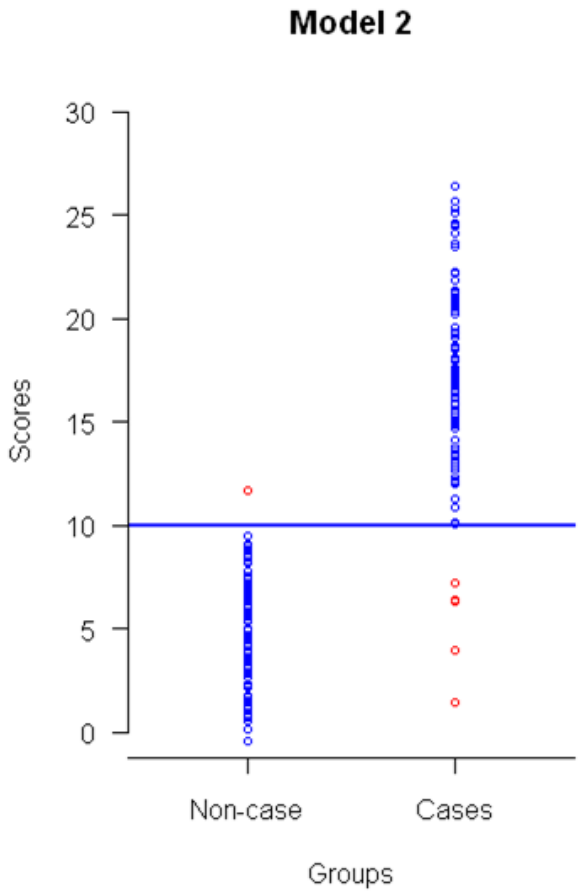




Example

If we are only given a subject's score, how can we use a model to determine whether the subject is a case or not? With a threshold? Which threshold?





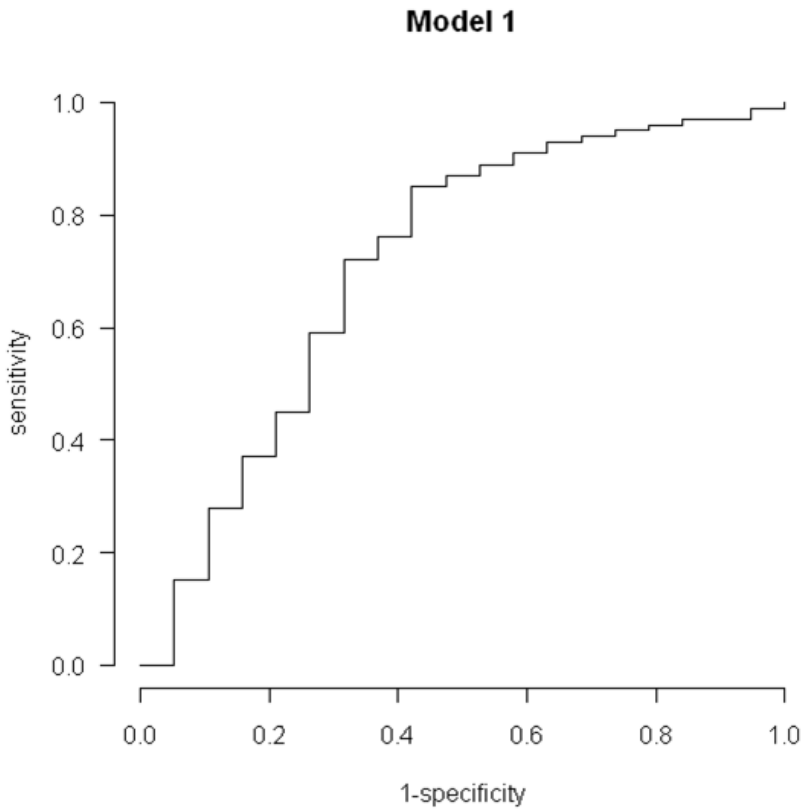
Sensitivity and specificity

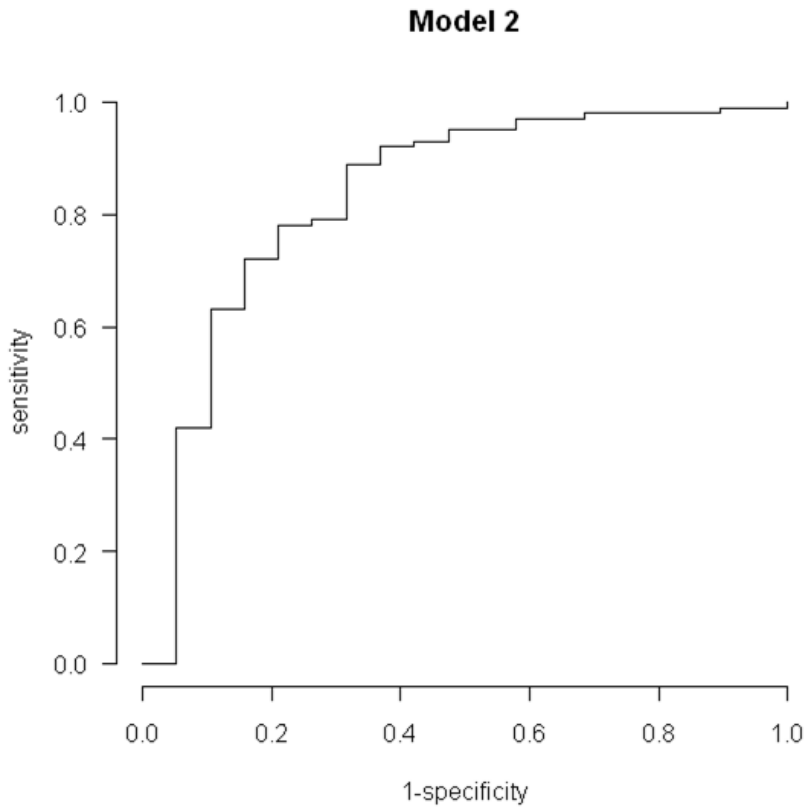
We are interested in reducing the number of errors of classification

- **False positive**: Incorrectly saying a subject is a case
- **False negative**: Incorrectly saying a subject is not a case
- **Sensitivity**: Proportion of cases correctly identified as cases
- **Specificity**: Proportion of non-cases correctly not identified as cases

ROC curves

Can calculate sensitivity and specificity for each possible threshold value:





Area under the curve can be interpreted as probability that a randomly chosen case will have a higher score than a randomly chosen non-case.

Area under ROC curve (AUROC) in R

- Can use package ROCR

- `library(ROCR)`
- `prediction(scores, true values)` where true values should be vector of 0s and 1s
- `performance(prediction.object, measure="auc")`
- `performance.object@y.values[[1]]`

Area under ROC curve (AUROC) in R

```
library(ROCR)
sars <- read.csv(”YOUR PATH/SARS08.CSV")
sars.glm <- glm(case ~ age + male + fever + sorethroat + cough,
data=sars, family=binomial)
sars$pred <- predict(sars.glm, type="response")
pred.obj <- prediction(sars$pred, sars$case)
perf.obj <- performance(pred.obj, measure="auc")
auroc <- perf.obj@y.values[[1]]
auroc
[1] 0.7500875
```

- How to estimate the se for AUROC?

Introduce bootstrapping

- As an alternative to asymptotic (large sample) approximation
  - Central limit theorem: sample mean approaches a normal distribution $\bar{X} \rightarrow Normal(\mu,\ \sigma^2/n)$
  - Maximum likelihood estimator (MLE) approaches a normal distribution
  - When we have a small sample size, such approximation may not be that accurate
- To estimate the variance without using a formula
- The analytic form of the distribution of an estimator is not available
- Deviation from the assumptions of the parametric model

Plug-in principle

- For a parameter $\theta = t(F)$, the plug-in estimate is given by $\hat{\theta} = t(\hat{F})$.
  - For z-test, if σ is unknown, we 'plug-in' the sample standard deviation *s*
  - For bootstrap, the true distribution F is unknown, we 'plug-in' $\hat{F}$
- Assume we have observed data $X_1, X_2, \ldots, X_n$, $\hat{F}$ gives a probability mass of $1/n$ to each X
- Let $S(\boldsymbol{X}) = S(X_1, X_2, \ldots, X_n)$ be any statistic, then

$$\hat{S}^* = \frac{1}{B}\sum_{b=1}^{B} S^{*b}$$

$$\mathrm{se}(\hat{S}^*) = \frac{1}{B-1}\sqrt{\sum_{b=1}^{B}(S^{*b} - \hat{S}^*)^2}$$

Bootstrapping (non-parametric)

- In order to construct confidence intervals for our parameter θ and calculate p-values, we need to know the sampling distribution of its estimate $\hat{\theta}$.
- If we don't know the sampling distribution we can instead **bootstrap** the sample.
  - i.e., to approximate the true distribution F by the empirical distribution $\hat{F}$:
- We sample **with replacement** from the original sample
- New sample size is same as original sample size
- Calculate $\hat{\theta}$ for the new sample, repeat the process by *B* times
- Distribution of $\hat{\theta}$ will approximate the sampling distribution

Bootstrapping – first example

- A hypothetical small dataset: {1, 5, 7, 2, 4, 6, 8, 9}
- 1st bootstrap dataset: {7, 2, 1, 4, 7, 8, 5, 9} → 1st estimate
- 2st bootstrap dataset: {1, 4, 9, 4, 7, 6, 8, 5} → 2nd estimate
- 3st bootstrap dataset: {9, 6, 8, 2, 4, 1, 8, 7} → 3rd estimate
- ...
- mth bootstrap dataset → mth estimate
- Distribution of the m estimates can be used

Bootstrapping – second example

```
mvc <- read.csv(”YOUR PATH/mvc.csv")
lm.mvc <- lm(height ~ age, mvc)
summary(lm.mvc)
confint(lm.mvc)[2, ]
```

Leads to $\hat{\beta}$ = –0.195, s.e. = $\hat{\sigma}_{\beta}$ = 0.087, 95% CI = (-0.371, -0.019)

Bootstrapping the standard error

We can calculate the standard error of $\hat{\beta}$ using bootstrapping.

```
bootlm <- function (m, original.data){
  beta <- rep (NA, m)
  for (i in 1:m){
    newdata <- original.data[sample(1:41, 41, replace=TRUE), ]
    b.lm <- lm(height ~ age, data=newdata)
    beta[i] <- coef(b.lm)[2]
  }
return(beta)
}
b.beta <- bootlm(1000, original.data=mvc)
mean(b.beta) # bootstrapped mean of beta
sd(b.beta) # bootstrapped standard error of the estimate of the mean
quantile(b.beta, c(0.025, 0.975)) # 95% CI of mean
```

Loop in R – FOR statement

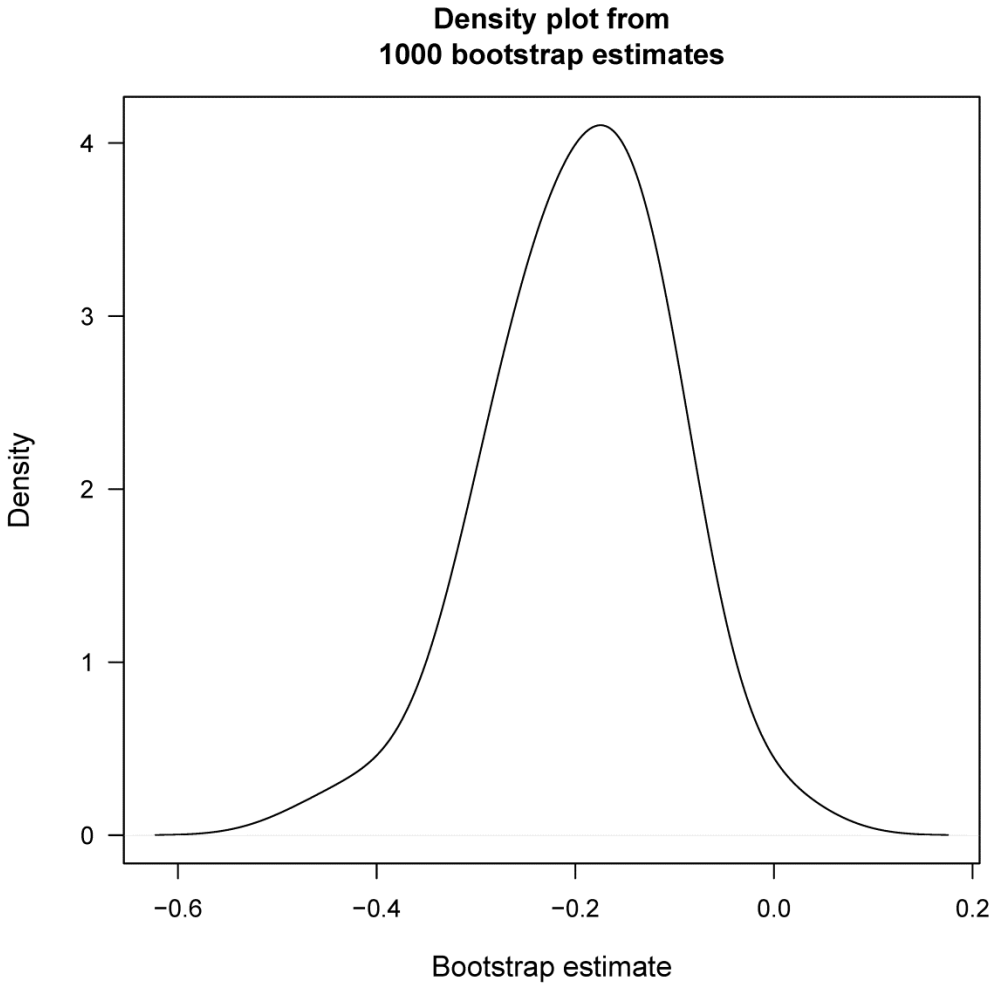
for (*variable in sequence*) *statement*

- *variable* is the loop variable
- *sequence* is the vector of values for variable within the loop
- *statement* is the body of the function, usually enclosed within braces {} for a grouped statement
- example:

```
x <- 0
for (i in 1:10){
x <- x + i}
x
```

Density plot

```
plot(density(b.beta,
bw=0.05), main = "Density
plot from 1000 bootstrap
estimates", xlab =
"Bootstrap estimate",
cex.lab=1.2, las=1)
```



Types of bootstrap

- Non-parametric bootstrap
  - No distributional assumption for the data
  - Resampling (with replacement) from the observations
- Residual bootstrap (semi-parametric)
  - Fit a (parametric) model and calculate the residuals $\hat{\varepsilon}_i = y_i - \hat{y}_i$
  - Resampling the residuals ε_i^* (non-parametric) and recreate the response $y_i^* = \hat{y}_i + \varepsilon_i^*$
  - Fit the model using the response y_i^* to obtain estimates for the parameter of interest
- Parametric bootstrap
  - First, estimate the distribution of the parameters
  - Resampling (with replacement) the observations from the model $y_i^* \sim M_\theta$
  - Estimate the parameters from each bootstrap dataset

Residual bootstrap for the linear regression example

Estimate β and its 95% CI using residual bootstrap (slide 15).

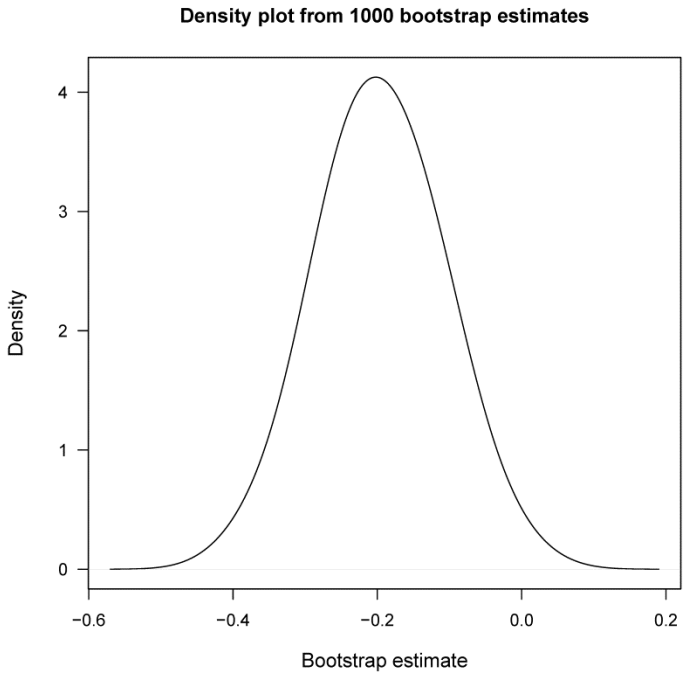
Residual bootstrap for the linear regression example

```
lm.mvc <- lm(height ~ age, mvc)
mvc$residual <- resid(lm.mvc)
mvc$height.pred <- predict(lm.mvc)
resbootlm <- function (m, original.data){
  beta <- rep (NA, m)
  for (i in 1:m){
    b.resid <- original.data$residual[sample(1:41, 41, replace=TRUE)]
    newdata <- original.data
    newdata$height.b <- newdata$height.pred + b.resid
    b.lm <- lm(height.b ~ age, data=newdata)
    beta[i] <- coef(b.lm)[2]
  }
return(beta)
}
```

Residual bootstrap for the linear regression example

```
resb.beta <- resbootlm(1000, original.data=mvc)
mean(resb.beta) # residual bootstrapped mean of beta
sd(resb.beta) # residual bootstrapped standard error of the
estimate of the mean
quantile(resb.beta, c(0.025, 0.975)) # 95% CI of mean
```

```
plot(density(resb.beta,
bw=0.05), main = "Density plot
from 1000 bootstrap estimates",
xlab = "Bootstrap estimate",
cex.lab=1.2, las=1)
```



Parametric bootstrap for the linear regression example

Estimate β and its 95% CI using parametric bootstrap (slide 15).

Parametric bootstrap for the linear regression example

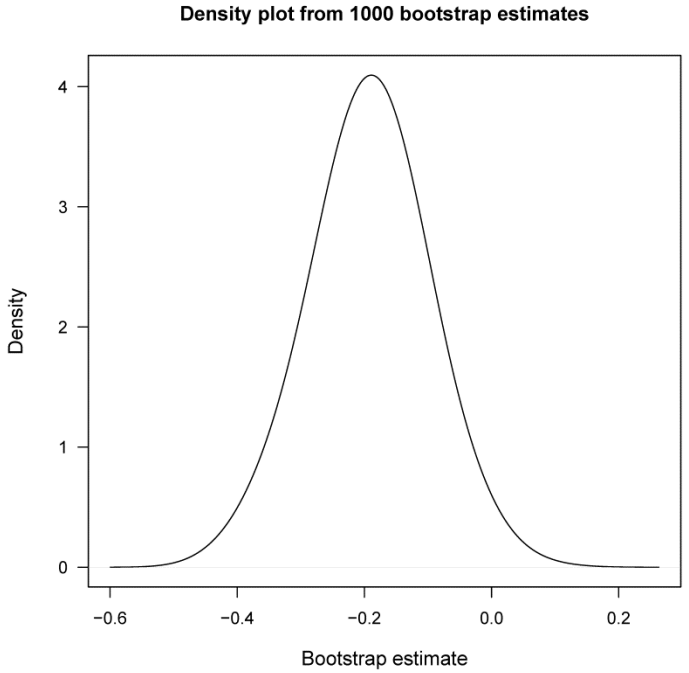
```
lm.mvc <- lm(height ~ age, mvc)


parabootlm <- function (m, original.data){
  beta <- rep (NA, m)
  for (i in 1:m){
    newdata <- original.data
    newdata$height.b <- simulate(lm.mvc)$sim_1
    b.lm <- lm(height.b ~ age, data=newdata)
    beta[i] <- coef(b.lm)[2]
  }
return(beta)
}
```

Parametric bootstrap for the linear regression example

```
parab.beta <- parabootlm(1000, original.data=mvc)
mean(parab.beta) # bootstrapped mean of beta
sd(parab.beta) # bootstrapped standard error of the estimate of the mean
quantile(parab.beta, c(0.025, 0.975)) # 95% CI of mean
```

```
plot(density(parab.beta,
bw=0.05), main = "Density plot
from 1000 bootstrap estimates",
xlab = "Bootstrap estimate",
cex.lab=1.2, las=1)
```



Bootstrap confidence intervals

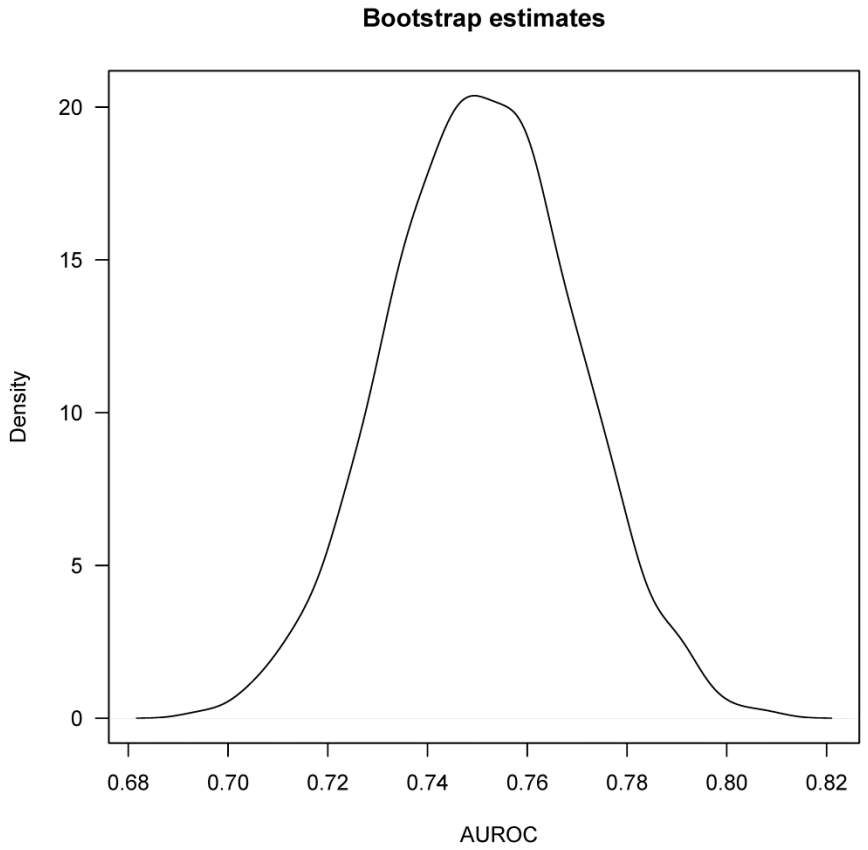
- Normal bootstrap confidence interval
  - $\hat{\theta} \pm t_{1-\alpha/2}\, se^*(\hat{\theta})$
  - Need large sample size or normal assumption
- Studentized bootstrap confidence interval
  - Extension of the normal bootstrap confidence interval by resampling *T** to replace $t_{1-\alpha/2}$
- Percentile bootstrap confidence interval
  - Given by $(\hat{\theta}^*_{\alpha/2},\ \hat{\theta}^*_{1-\alpha/2})$
- Bias-corrected and accelerated (BCa) bootstrap confidence interval
  - Shift and rescale the estimate to correct for bias

Confidence interval for AUROC

```
b.auroc <- rep(NA,1000)
for (i in 1:1000){
  sars.new <- sars[sample(1:800, 800, replace=TRUE),]
  b.pred.obj <- prediction(sars.new$pred, sars.new$case)
  b.perf.obj <- performance(b.pred.obj, measure="auc")
  b.auroc[i] <- b.perf.obj@y.values[[1]]
}
mean(b.auroc)
[1] 0.7504905
quantile(b.auroc, c(0.025,0.975))
     2.5%     97.5%
0.7128376 0.7845543
```

Example: density for area under ROC curve

```
plot(density(b.auroc), las=1, xlab="AUROC",
main="Bootstrap estimates")
```



Bootstrap package ('boot') in R

boot(*data*, *statistic*, *R*) [generate bootstrap samples]

- *statistic* is a function returning the statistic of interest
  - there should be at least two arguments for the function, first: original data; second: indices which specify bootstrap samples
- *R* is the number of bootstrap replicate

boot.ci(*boot.out*, *conf = 0.95*, *type = "all"*) [construct bootstrap CI]

- *boot.out* is the object of class "boot" with bootstrap calculation
- *conf* is the confidence level for the confidence interval
- *type* selects the type of bootstrap confidence intervals, such as "*norm*", *"stud", "perc", "bca"*

boot function – MVC example

```
mvclm.out <- function (data, indices){
  newdata <- data[indices, ]
  b.lm <- lm(height ~ age, data=newdata)
  return(coef(b.lm)["age"])
}

mvc.b.out <- boot(data=mvc, statistic=mvclm.out,
R=1000)

boot.ci(mvc.b.out)
```

boot function – AUROC

```
auroc.b.out <- function (data, indices){
  sars.new <- sars[indices,]
  sars.glm <- glm(case ~ age + male + fever + sorethroat + cough,
data=sars.new, family=binomial)
  sars.new$pred <- predict(sars.glm, type="response")
  b.pred.obj <- prediction(sars.new$pred, sars.new$case)
  b.perf.obj <- performance(b.pred.obj, measure="auc")
  return(b.perf.obj@y.values[[1]])
}

sars.b.out <- boot(data=sars, statistic=auroc.b.out, R=1000)

boot.ci(sars.b.out)
```

Review

What bootstrap cannot do…

- Generate new data
- Obtain better estimates
- Correct for model misspecification

"pull oneself up by one's bootstraps"

What bootstrap can do…

- Provide an alternative estimate when the analytic form of an estimator is complicated or unavailable
- Provide an alternative estimate when deviated from the assumptions of the parametric model