MSG500/MVE190
Linear Models - Lecture 13

Rebecka Jörnsten
Mathematical Statistics
University of Gothenburg/Chalmers University of Technology

December 11, 2012

Bootstrap can be used to construct CI in nonlinear regression models, or when the error distribution is non-normal. Here's a few notes on bootstrap.

1 Generating data

1.1 Parametric bootstrap

Let's say you have a data set consisting of n data objects $(x_i, y_i)_{i=1}^n$ where x_i may be a vector of multiple explanatory variable observations.

- Fit your model to the data ($y_i = \beta_0 + \sum_{j=1}^{p-1} \beta_j x_{ij}$). Record the coefficient estimates $\hat{\beta}_j$, $j = 0, \cdots, p-1$
- Generate a new set of residuals $(e_i^b)_{i=1}^n$ from distribution F_e.
  F_e is for example $N(0, \hat{\sigma}^2)$ (with $\hat{\sigma}^2$ estimated from the model fit). In the lab you can also simulate errors from t_{df} where you choose the degrees of freedom: set e_i to $\hat{\sigma} * t_i$ where $t_i \sim t_{df}$.
- Create the bootstrap data set $(x_i, y_i^b)_{i=1}^n$ where $y_i^b = \hat{\beta}_0 + \sum_{j=1}^{p-1} \hat{\beta}_j x_{ij} + e_i^b$

1.2 Non-parametric bootstrap

Let's say you have a data set consisting of n data objects $(x_i, y_i)_{i=1}^n$ where x_i may be a vector of multiple explanatory variable observations.

- Fit your model to the data ($y_i = \beta_0 + \sum_{j=1}^{p-1} \beta_j x_{ij}$). Record the coefficient estimates $\hat{\beta}_j$, $j = 0, \cdots, p-1$
- Standardize the residuals $(\tilde{e}_i)_{i=1}^n = (e_i/\sqrt{1-h_{ii}})_{i=1}^n$.
- Draw new residuals e_i^b by resampling from $\tilde{e}_i$ (draw n residuals e_i^b from $(\tilde{e}_i)_{i=1}^n$ with replacement).
- Create the bootstrap data set $(x_i, y_i^b)_{i=1}^n$ where $y_i^b = \hat{\beta}_0 + \sum_{j=1}^{p-1} \hat{\beta}_j x_{ij} + e_i^b$

Note, a second variant of non-parametric bootstrap is to resample the data pairs (x_i, y_i) directly, without specifying a model.

1.3 The principle

The idea behind bootstrap is to create a data set for which we know the true model, and can thus investigate (or estimate) the sampling distribution of estimators of interest directly. In the above examples, for bootstrap data y^b the true model *is* $\hat{\beta}_j$, $j = 0, \cdots, p-1$. The principle underlying bootstrap is thus

$$\frac{\hat{\beta}_j^b - \hat{\beta}_j}{SE(\hat{\beta}_j^b)} =^d \frac{\hat{\beta}_j - \beta_j}{SE(\hat{\beta}_j)}.$$

By estimating, or observing, the sampling distribution of $\hat{\beta}_j^b$ (or the pivotal element $\frac{\hat{\beta}_j^b - \hat{\beta}_j}{SE(\hat{\beta}_j^b)}$) across multiple bootstrap data sets, we get an idea about the sampling distribution of $\hat{\beta}_j$ from the real data.

2 Bootstrap confidence intervals

There are several approaches to constructing bootstrap confidence intervals.

The normal-theory interval assumes that the statistic T (e.g. a regression coefficient estimate) is normally distributed, and uses the bootstrap estimate of sampling variance, and perhaps of bias, to construct a 100(1 - α)-percent confidence interval of the form

$$(T - B^*) \pm z_{1-\alpha/2}\widehat{SE(T)},$$

where z refers to the normal distribution quantiles (e.g., 1.96 for a 95-percent confidence interval, where $\alpha = .05$).

Here, B^* is the bias estimate obtained from the bootstrap: $\sum_{b=1}^{B}(T - T^b)/B$, and $\widehat{SE(T)} = \sqrt{\sum_{b=1}^{B}(T^b - T^*)/(B-1)}$ is the bootstrap estimated standard error of statistics T (where $T^* = \sum_{b=1}^{B} T^b/B$).

An alternative approach, called the bootstrap percentile interval, is to use the empirical quantiles of T^b to form a confidence interval: $[T^b(.025), T^b(.0975)]$, where the end-points of the interval correspond to the 2.5% and 97.5% percentiles of the bootstrap statistic values.

2.1 Pivotal method

The confidence intervals based on

$$(T - B^*) \pm z_{1-\alpha/2}\widehat{SE(T)}$$

assume that the bootstrap T^b have the same sampling distribution as T. However, this is not true if we use the non-pivotal elements. We should instead use the t-statistic $\theta^b = (T^b - T)/SE(T^b)$. We reformulate the confidence intervals as

$$(T - B^*) \pm q_{1-\alpha/2}\widehat{SE(T)},$$

where $q_{1-\alpha/2}$ percentile of θ^b.

If the SE of the estimates T is unknown (like in non-linear regression) we need a second level of bootstrap to estimate $SE(T^b)$. For each bootstrap sample b, we generate a second set of bootstrap samples $u, u = 1, \cdots, U$. Each bootstrap data u provides an estimate $T^{u(b)}$. We compute $\widehat{SE(T^b)} = \sqrt{\sum_{u=1}^{U}(T^{u(b)} - T^b)/(U-1)}$. Usually we can get away with a smaller value for U than B, e.g. $U = 25$ and $B = 1000$.

In class, we used the most simple pivotal method where we assume $SE(T)$ is known. In linear regression models $SE(\hat{\beta}_j)$ is $\hat{\sigma} * \sqrt{(X'X)_{jj}}$. Each pivotal element is thus $\frac{\hat{\beta}_j^b - \hat{\beta}_j}{SE(\hat{\beta}_j)}$.

To construct the confidence interval, from bootstrap we have the quantiles of the pivotal elements $q_{\alpha/2}$ and $q_{1-\alpha/2}$:

$$P(q_{\alpha/2} \leq \theta^b \leq q_{1-\alpha/2}) = 1 - \alpha.$$

We then assume this distribution mimics the sampling distribution of the original estimate and set up the CI as

$$[\hat{\beta}_j - q_{1-\alpha/2}\widehat{SE}, \hat{\beta}_j - q_{\alpha/2}\widehat{SE}]$$

2.2 If you're curious: More on the Percentile method

The percentile method is very easy to use. We simple generate bootstrap data and compute the corresponding estimates T^b. The confidence interval is obtained from the lower and upper percentiles of the bootstrap estimates.

However, when the distribution of T is skewed we can get poor coverage of the confidence intervals generated with the percentile methods. There's been work to alleviate this problem using a *bias-corrected and accelerated* bootstrap method proposed by Brad Efron.

- Start by computing the proportion of T^b that is below the original T: P.
- Compute the corresponding normal quantile $\phi^{-1}(P)$. For example, if 50% of the $T^b < T$, $\phi^{-1}(.5) = 0$. (In R, simple take P and use the function `qnorm(P)`). The value $z = \phi^{-1}(P)$ is called the *correction factor*.
- Compute $A = \frac{\sum_{i=1}^{n}(T_{-i}-\bar{T})^3}{6[\sum_{i=1}^{n}(T_{-i}-\bar{T})^2]^{3/2}}$
- Using A we compute

$$a_1 = \phi(z + \frac{z - z_{1-\alpha/2}}{1 - a(z - z_{1-\alpha/2})})$$

and

$$a_2 = \phi(z + \frac{z + z_{1-\alpha/2}}{1 - a(z + z_{1-\alpha/2})})$$

- Pick q_{lower} as the largest integer $< B * a_1$, and q_{upper} and the smallest integer $> B * a_2$. Note that if the correction factors $z = 0$ and $a = 0$, $q_{lower} = \alpha/2$ and $q_{upper} = 1 - \alpha/2$. That is, if the corrections are 0 we will use the regular percentiles of T^b to set up our confidence interval.
- Finally, the confidence interval is obtained as

$$[T^b(q_{lower}), T^b(q_{upper})]$$

3 Demo 13

We revisit the count data from the previous lecture.

```
> library(stats)
> time <- c(0, 1, 2, 3, 4, 7, 9, 11, 14, 16, 18, 21, 24, 29, 32,
+     35, 38, 46)
> count <- c(5149, 5435, 5358, 5462, 4755, 4457, 4538, 4378, 4189,
+     4195, 3743, 3855, 3569, 3045, 3091, 3064, 2729, 2477)
> dataf <- data.frame(cbind(count, time))
> names(dataf) <- c("count", "time")
```

We create the data set first.

```
> m1 <- nls(count ~ exp(b) * exp(-cc * time), data = dataf, start = list(b = log(5000),
+     cc = 0.02))
> m1s <- summary(m1)
> print(m1s)

Formula: count ~ exp(b) * exp(-cc * time)

Parameters:
    Estimate Std. Error t value Pr(>|t|)
b  8.5859321  0.0145649  589.50  < 2e-16 ***
cc 0.0173375  0.0008904   19.47 1.45e-12 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 181.7 on 16 degrees of freedom

Number of iterations to convergence: 3
Achieved convergence tolerance: 8.692e-06
```

We fit a nonlinear regression model to the data.

```
> pf <- profile(m1, alphamax = 0.05)
> par(mfrow = c(1, 1))
> plot(pf$b[, 2][, 1], pf$b[, 1], type = "l", main = "profile of b",
+     xlab = "b", ylab = "tau")
> abline(v = m1s$p[1, 1], lty = 2)

> plot(pf$cc[, 2][, 2], pf$cc[, 1], type = "l", main = "profile of c",
+     xlab = "c", ylab = "tau")
> abline(v = m1s$p[2, 1], lty = 2)
```

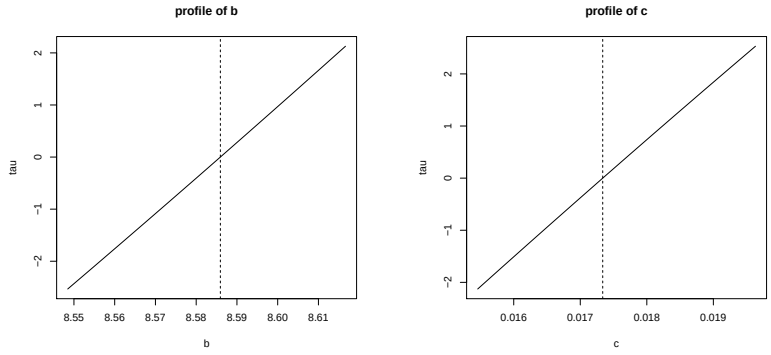

Figure 1: Profile plots.

In Figure 1 I show the profile plots for parameter b and c.

The t-based confidence intervals are given as:

```
> bint <- m1s$coef[1, 1] + m1s$coef[1, 2] * qt(0.975, m1s$df[2]) *
+     c(-1, 1)
> cint <- m1s$coef[2, 1] + m1s$coef[2, 2] * qt(0.975, m1s$df[2]) *
+     c(-1, 1)
> dd <- as.data.frame(rbind(bint, cint))
> row.names(dd) <- c("b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)

          2.5       97.5
b  8.55505592 8.61680834
cc 0.01544995 0.01922507
```

I also provide the centered CI (around the estimate) so you can see the symmetry.

```
> print(dd - m1s$p[, 1])

           2.5       97.5
b  -0.03087621 0.03087621
cc -0.00188756 0.00188756
```

Compare this to the profile based intervals

```
> print(confint(m1))

         2.5%      97.5%
b  8.55465368 8.61652749
cc 0.01546541 0.01925541

> print(confint(m1) - m1s$p[, 1])
```

```
           2.5%       97.5%
b  -0.031278450 0.030595364
cc -0.001872103 0.001917905
```

Let us now try bootstrap for this data set.

```
> B <- 1000
> betap <- matrix(0, B, 2)
> sebetap <- betap
> for (bb in (1:B)) {
+     datab <- dataf
+     ct2 <- exp(m1s$p[1, 1]) * exp(-m1s$p[2, 1] * dataf$time) +
+         rnorm(length(dataf$time), sd = m1s$sigma)
+     datab$count <- ct2
+     m2 <- nls(count ~ exp(b) * exp(-cc * time), data = datab,
+         start = list(b = m1s$p[1, 1], cc = m1s$p[2, 1]), nls.control(warnOnly = TRUE))
+     m2s <- summary(m2)
+     betap[bb, ] <- m2s$p[, 1]
+     sebetap[bb, ] <- m2s$p[, 2]
+ }

> hist(betap[, 1], n = 25, main = "bootstrap dist of b")

> hist(betap[, 2], n = 25, main = "bootstrap dist of c")
```

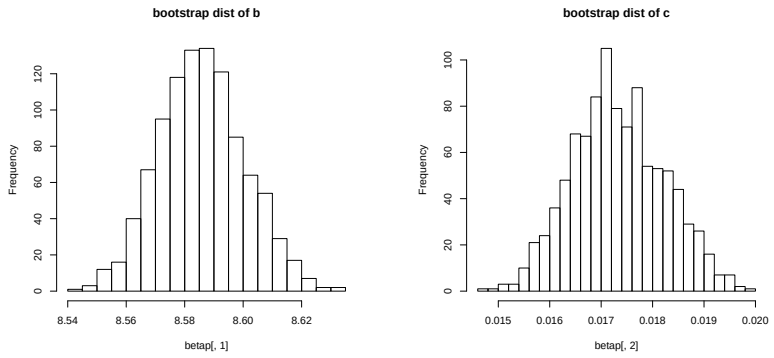

Figure 2: Histogram of bootstrap estimates

In Figure 2 I show the histogram of the bootstrap estimates of the parameters b and c. From the bootstrap data we can estimate both bias and variance for the estimates. First, the bias

```
> print(round(apply(betap, 2, mean) - m1s$p[, 1], 5))

     b     cc
-9e-05  0e+00
```

and the SE

```
> print(round(apply(betap, 2, sd), 5))

[1] 0.01478 0.00090
```

How do these compare to the output in the model summary?

I first construct with the percentile intervals.

```
> qb <- quantile(betap[, 1], c(0.025, 0.975))
> qc <- quantile(betap[, 2], c(0.025, 0.975))
> dd <- as.data.frame(rbind(qb, qc))
> row.names(dd) <- c("b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)

          2.5      97.5
b  8.55762742 8.6154061
cc 0.01565839 0.0191197

> print(dd - m1s$p[, 1])

            2.5        97.5
b  -0.028304708 0.029473923
cc -0.001679121 0.001782194
```

A more commonly used bootstrap confidence interval is the one based on the pivots (t-statistics):

```
> pivb <- (betap[, 1] - m1s$p[1, 1])/sebetap[, 1]
> pivc <- (betap[, 2] - m1s$p[2, 1])/sebetap[, 2]
> qb <- quantile(pivb, c(0.025, 0.975))
> qc <- quantile(pivc, c(0.025, 0.975))
> intb <- m1s$p[1, 1] - m1s$p[1, 2] * c(qb[2], qb[1])
> intc <- m1s$p[2, 1] - m1s$p[2, 2] * c(qc[2], qc[1])
> dd <- as.data.frame(rbind(intb, intc))
> row.names(dd) <- c("b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)

          2.5       97.5
b  8.55373904 8.61564960
cc 0.01546438 0.01915276

> print(dd - m1s$p[, 1])

            2.5        97.5
b  -0.032193088 0.029717477
cc -0.001873128 0.001815248
```

Let us try a non-parametric resampling.

```
> B <- 1000
> betap <- matrix(0, B, 2)
> sebetap <- betap
> for (bb in (1:B)) {
+     datab <- dataf
+     res <- sample(residuals(m1), dim(dataf)[1])
+     ct2 <- exp(m1s$p[1, 1]) * exp(-m1s$p[2, 1] * dataf$time) +
+         res
+     datab$count <- ct2
+     m2 <- nls(count ~ exp(b) * exp(-cc * time), data = datab,
+         start = list(b = m1s$p[1, 1], cc = m1s$p[2, 1]), nls.control(warnOnly = TRUE))
+     m2s <- summary(m2)
+     betap[bb, ] <- m2s$p[, 1]
+     sebetap[bb, ] <- m2s$p[, 2]
+ }

> pivb <- (betap[, 1] - m1s$p[1, 1])/sebetap[, 1]
> pivc <- (betap[, 2] - m1s$p[2, 1])/sebetap[, 2]
> qb <- quantile(pivb, c(0.025, 0.975))
```

```
> qc <- quantile(pivc, c(0.025, 0.975))
> intb <- m1s$p[1, 1] - m1s$p[1, 2] * c(qb[2], qb[1])
> intc <- m1s$p[2, 1] - m1s$p[2, 2] * c(qc[2], qc[1])
> dd <- as.data.frame(rbind(intb, intc))
> row.names(dd) <- c("b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)

         2.5       97.5
b  8.5590221 8.61111241
cc 0.0154879 0.01915022

> print(dd - m1s$p[, 1])

           2.5        97.5
b  -0.02691001 0.025180277
cc -0.00184961 0.001812716
```

You can also try resampling observation pairs instead of residuals above.

In the above example, the bootstrap didn't make that much of a difference. Try it on your own data, or on simulated data. I will here repeat the exercise on a simulated data set. Note, I don't use the original data set since it is extremely poorly behaved for the background radiation parameter.

```
> count <- 1000 + 4000 * exp(-0.05 * dataf$time) + rnorm(dim(dataf)[1],
+     sd = 100)
> datap <- as.data.frame(cbind(count, dataf$time))
> names(datap) <- names(dataf)
> m1 <- nls(count ~ exp(a) + exp(b) * exp(-cc * time), data = datap,
+     start = list(a = log(2000), b = log(2000), cc = 0.2))
> m1s <- summary(m1)
> print(m1s)

Formula: count ~ exp(a) + exp(b) * exp(-cc * time)

Parameters:
   Estimate Std. Error t value Pr(>|t|)
a  6.945837   0.101062   68.73  < 2e-16 ***
b  8.296935   0.023493  353.17  < 2e-16 ***
cc 0.049899   0.002985   16.72 4.17e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 73.03 on 15 degrees of freedom

Number of iterations to convergence: 6
Achieved convergence tolerance: 5.196e-07

> par(mfrow = c(1, 3))
> pf <- profile(m1)
> plot(pf)
```

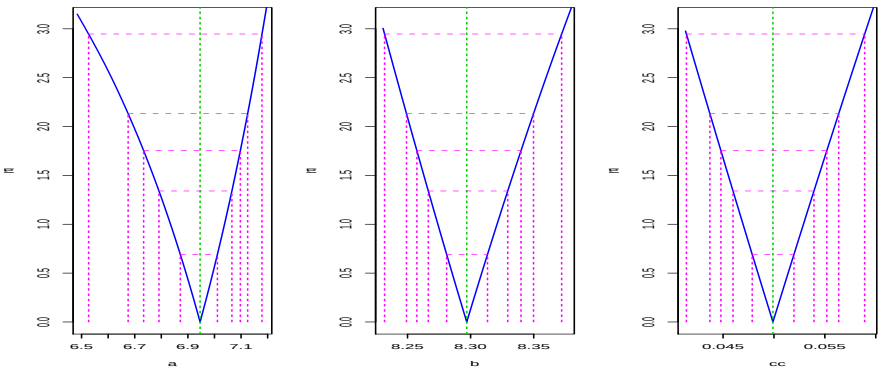

Figure 3: Profile plots - simulation.

In Figure 3 you can see the profiles for the three parameters.

```
> aint <- m1s$coef[1, 1] + m1s$coef[1, 2] * qt(0.975, m1s$df[2]) *
+     c(-1, 1)
> bint <- m1s$coef[2, 1] + m1s$coef[2, 2] * qt(0.975, m1s$df[2]) *
+     c(-1, 1)
> cint <- m1s$coef[3, 1] + m1s$coef[3, 2] * qt(0.975, m1s$df[2]) *
+     c(-1, 1)
> dd <- as.data.frame(rbind(aint, bint, cint))
> row.names(dd) <- c("a", "b", "cc")
```

```
> colnames(dd) <- c("2.5", "97.5")
> print(dd)

          2.5       97.5
a  6.73042869 7.16124505
b  8.24686116 8.34700896
cc 0.04353674 0.05626173

> print(dd - m1s$p[, 1])

            2.5        97.5
a  -0.215408178 0.215408178
b  -0.050073899 0.050073899
cc -0.006362497 0.006362497
```

Compare this to the profile based intervals

```
> print(confint(m1))

         2.5%      97.5%
a  6.67514812 7.12471519
b  8.24920037 8.34985964
cc 0.04370528 0.05636027

> print(confint(m1) - m1s$p[, 1])

           2.5%       97.5%
a  -0.270688753 0.178878317
b  -0.047734697 0.052924581
cc -0.006193958 0.006461036

> B <- 1000
> betap <- matrix(0, B, 3)
> sebetap <- betap
> for (bb in (1:B)) {
+     datab <- dataf
+     ct2 <- exp(m1s$p[1, 1]) + exp(m1s$p[2, 1]) * exp(-m1s$p[3,
+         1] * dataf$time) + rnorm(length(dataf$time), sd = m1s$sigma)
+     datab$count <- ct2
+     m2 <- nls(count ~ exp(a) + exp(b) * exp(-cc * time), data = datab,
+         start = list(a = log(2000), b = log(2000), cc = 0.02),
+         nls.control(warnOnly = TRUE))
+     m2s <- summary(m2)
+     betap[bb, ] <- m2s$p[, 1]
+     sebetap[bb, ] <- m2s$p[, 2]
+ }

> hist(betap[, 1], n = 25, main = "bootstrap dist of a")

> hist(betap[, 2], n = 25, main = "bootstrap dist of b")

> hist(betap[, 3], n = 25, main = "bootstrap dist of c")
```

In Figure 4 I show the histogram of the bootstrap estimates of the parameters a, b and c.

The percentile CIs are:

```
> qa <- quantile(betap[, 1], c(0.025, 0.975))
> qb <- quantile(betap[, 2], c(0.025, 0.975))
> qc <- quantile(betap[, 3], c(0.025, 0.975))
> dd <- as.data.frame(rbind(qa, qb, qc))
> row.names(dd) <- c("a", "b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)
```

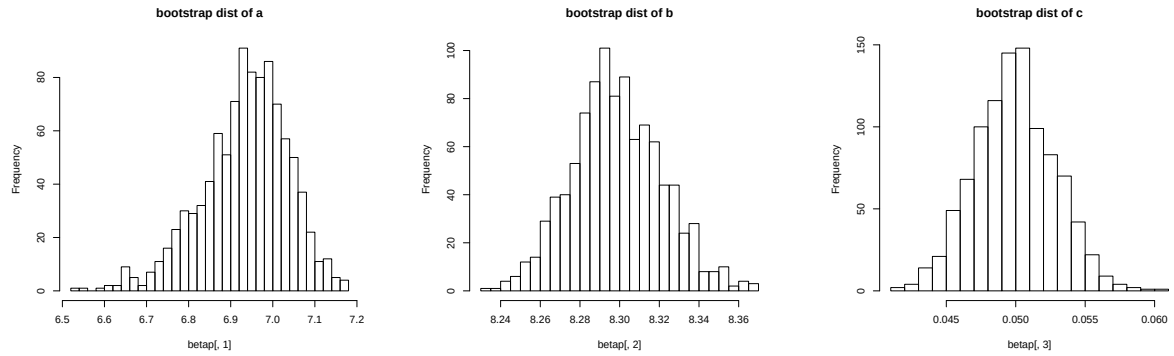

Figure 4: Histogram of bootstrap estimates - simulation

```
          2.5       97.5
a  6.70427884 7.11131037
b  8.25529253 8.34524156
cc 0.04452258 0.05557833
```

```
> print(dd - m1s$p[, 1])
```

```
            2.5        97.5
a  -0.241558030 0.165473497
b  -0.041642530 0.048306494
cc -0.005376656 0.005679097
```

The CIs based on the t-statistics (pivotal method):

```
> piva <- (betap[, 1] - m1s$p[1, 1])/sebetap[, 1]
> pivb <- (betap[, 2] - m1s$p[2, 1])/sebetap[, 2]
> pivc <- (betap[, 3] - m1s$p[3, 1])/sebetap[, 3]
> qa <- quantile(piva, c(0.025, 0.975))
> qb <- quantile(pivb, c(0.025, 0.975))
> qc <- quantile(pivc, c(0.025, 0.975))
> inta <- m1s$p[1, 1] - m1s$p[1, 2] * c(qa[2], qa[1])
> intb <- m1s$p[2, 1] - m1s$p[2, 2] * c(qb[2], qb[1])
> intc <- m1s$p[3, 1] - m1s$p[3, 2] * c(qc[2], qc[1])
> dd <- as.data.frame(rbind(inta, intb, intc))
> row.names(dd) <- c("a", "b", "cc")
> colnames(dd) <- c("2.5", "97.5")
> print(dd)
```

```
          2.5       97.5
a  6.69710320 7.12450114
b  8.24960023 8.34410859
cc 0.04376529 0.05635367
```

```
> print(dd - m1s$p[, 1])
```

```
            2.5        97.5
a  -0.248733668 0.178664274
b  -0.047334831 0.047173523
cc -0.006133945 0.006454432
```

References

Good book: B. Efron and R. Tibshirani. "An Introduction to the Bootstrap" (Chapman & Hall, 1994)
Also, check online for Bootstrap tutorials.