MATH 6359, Statistical Computing, Homework 6

Andrey Skripnikov

November 14, 2017

SUBMISSION GUIDELINES:

- **Bring the hard-copy (NO ELECTRONIC VERSIONS OVER EMAIL WILL BE ACCEPTED) of typed up solutions to me in PGH 611 on Monday, December 4th, or at any point prior to that - e.g. in class on Nov. 28, 30.**
- **If you can't hand it over to me in person on the aforementioned dates, please make sure to setup a different date to meet with me. Please MAKE SURE TO DO ALL OF THAT BEFORE YOU GO OUT OF TOWN.**
- **Keep it under 5 pages total (all included - text, code, plots, tables). DON'T (!) repeat the problem formulation, go straight to solution.**
- **Roughly follow the example format.**
- **Point total is 65 (100%).**

PROBLEM #1 - 30 points (Just provide the code).

Write your own bootstrapping function that will take in as arguments:

- **single vector of continuous values**
- **number of bootstrap replicates to be generated**
- **the statistic you're interested in (mean, median, etc) in the form of a FUNCTION (much like the 'statistic' argument for 'boot' function from R package 'boot')**

and output the following elements:

- **Bootstrap estimate for bias of the statistic of interest**
- **Bootstrap estimate for standard error of the statistic of interest**

- **Confidence intervals for the statistic of interest:**
  - **Normal**
  - **Basic**
  - **Percentile**

Having defined that function, proceed to find the data set containing at least one continuous variable (you CAN'T use the ones mentioned in class; you CAN use the data sets from you previous homeworks, e.g. problem # 1 in HW # 4). For that continuous variable, apply your bootstrapping function to calculate bias, error, and all confidence intervals for the following statistics:

1. **Mean.**
2. **Median.**

Alongside applying your own function, for comparative purposes also apply functions 'boot' and 'boot.ci' from 'boot' package for that same input (see example).

Additionally, just to compare with the (non-parametric) bootstrapping approach, obtain confidence interval for the MEAN via PARAMETRIC approach(!) - use $t.test$ **with a** $H_0 : \mu = 0$ **for your vector. Question - could you have done it for the MEDIAN (put a comment in your code)?**

Example. I will be looking to make inferences about the mean and median blood pressure for patients from *bp.obese* data set.

```
> my.bootstrap <- function(x,B,f){
.. All you ...
}
>
> library(ISwR)
>
> m <- 10000  # Number of bootstrap replicates to run.
>
> # Statistic: MEAN
>
> ## Our function:
> my.bootstrap(bp,B=m,f=mean)
$bias
[1] -0.01301569
$SE
[1] 1.795613
$CI.norm
   Lower    Upper
```

```
123.5003 130.5389
$CI.basic
   Lower    Upper
123.4118 130.4711
$CI.percentile
   Lower    Upper
123.5681 130.6275
>
> ## R's boot function to compare
> R.boot.obj <- boot(data=bp,statistic=function(x,i) mean(x[i]),R=m)
> R.boot.obj
  ORDINARY NONPARAMETRIC BOOTSTRAP
...
Bootstrap Statistics :                  # Your bias and SE should be ROUGHLY close
    original      bias    std. error    # to these values.
                                        # E.g. my bias is -0.013 vs "boot"s bias of +0.009
t1* 127.0196 0.009156863    1.784195    # and it's OK. It varies due to random subsampling.
> ## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
> boot.ci(R.boot.obj)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 10000 bootstrap replicates
...
Intervals :
Level      Normal              Basic          # Your CI.Normal, CI.Basic, CI.percentile
95%   (123.5, 130.5 )   (123.4, 130.4 )       # should be ROUGHLY close to respecitve
                                              # outputs of 'boot.ci'.
Level     Percentile            BCa
95%   (123.7, 130.6 )   (123.9, 130.9 )       # Don't worry too much about
Calculations and Intervals on Original Scale  # slight discrepancies.
Warning message:                              # And definitely don't worry about that warni
In boot.ci(R.boot.obj) :
  bootstrap variances needed for studentized intervals
>
> ## The "distributional" assumption approach to compare the CIs - simply to show you
> ## how one can obtain pretty similar intervals by using two very different approaches:
> ## NON-PARAMETRIC BOOTSTRAP (computational method) vs PARAMETRIC methods (normal approx.)
> ## Doesn't allow for "BIAS" estimation, as it ASSUMES THETA^HAT TO BE UNBIASED
> t.test(bp,mu=0)
...
95 percent confidence interval:
 123.4479 130.5914
...
```

```
> # Statistic: MEDIAN
>
> ## Our function:
> my.bootstrap(bp,B=m,f=mean)
$bias
[1] 0.1302
$SE
[1] 2.127181
$CI.norm
   Lower    Upper
119.8308 128.1692
$CI.basic
Lower Upper
  120   129
$CI.percentile
Lower Upper
  119   128
>
> ## R's boot function to compare
> R.boot.obj <- boot(data=bp,statistic=function(x,i) median(x[i]),R=m)
> R.boot.obj
...
Bootstrap Statistics :
    original  bias    std. error
t1*      124  0.1083    2.076632
> ## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
> boot.ci(R.boot.obj)
Based on 10000 bootstrap replicates
...
Intervals :
Level      Normal              Basic
95%   (119.8, 128.0 )   (120.0, 129.0 )

Level     Percentile            BCa
95%   (119, 128 )   (118, 125 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
Warning message:
In boot.ci(R.boot.obj) :
  bootstrap variances needed for studentized intervals
```

PROBLEM #2 - 15 points. (Just provide the code)

Coming back to the multivariate *cystic fibrosis* (*cystfibr*) **data set from** *ISwR* **package on patient's lung function, let's consider the following three models to predict patient's maximum respiratory pressure** (*pemax* **variable):**

- $pemax = \beta_0 + \beta_1\ age + \beta_2\ sex + \beta_3\ weight + \beta_4\ bmp + \beta_5\ fev1 + \beta_6\ rv + \beta_7\ frc + \beta_8\ tlc$
- $pemax = \beta_0 + \beta_1\ age + \beta_2\ sex + \beta_3\ height + \beta_4\ weight + \beta_5\ bmp$
- $pemax = \beta_0 + \beta_1\ age + \beta_2\ sex + \beta_3\ fev1 + \beta_4\ rv + \beta_5\ frc + \beta_6\ tlc$

Apply n**-fold (also known as "leave-one-out") cross-validation and calculate the resulting mean squared prediction error for each of the three models. Identify the best model with respect to that error.**

```
> library(ISwR)
> ## Do n-fold (or also 'leave-one-out') cross-validation to compare the models
> n <- nrow(cystfibr)
> cv.err.1 <- cv.err.2 <- cv.err.3 <- numeric(n)
> for (j in 1:n){
...All you. ..
... Define training/testing subsets...
... Use 'lm()' function to fit each of three models...
... Calculate the prediction errors...
}
>
> print(c(mean(cv.err.1^2),mean(cv.err.2^2),mean(cv.err.3^2)))
...
```

PROBLEM #3 - 10 points (Provide the code + the inverse function derivation).

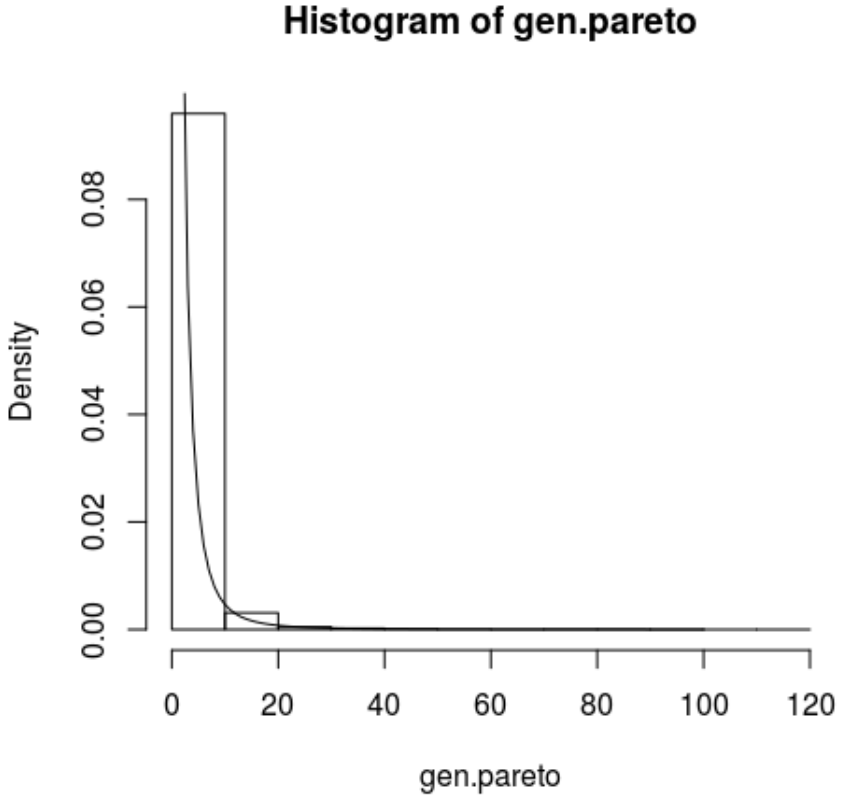
The $Pareto(a, b)$ distribution has cdf

$$F(x) = 1 - \frac{b}{x}^{a}, \ x \geq b > 0, \ a > 0.$$

For this distribution:

1. **Analytically derive the probability inverse transformation $F^{-1}(U)$.**
2. **Use the inverse transform method to simulate a random sample from the $Pareto(2, 2)$ distribution.**
3. **Graph the density histogram of the sample with the $Pareto(2, 2)$ density superimposed for comparison.**

Example. Here I won't show you any code, but will give an exemplary plot expected:



PROBLEM #4 - 10 points (Just provide the code).

The $Beta(a,b)$ distribution has the following density function (we look at the case of a and b being positive INTEGERS):

$$F(x) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)},\ B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)},\ \Gamma(n) = (n-1)!,\ 0 < x < 1$$

Write your own function to compute a Monte Carlo estimate of cumulative distribution function (cdf) for $Beta(a,b)$ at any user-defined point x for arbitrary positive integers a and b. Your function should take as arguments:

- **Number m of Monte Carlo simulations you'd like to run.**
- **Values a and b.**
- **The point x at which the user wants the $Beta(\alpha,\beta)$ cdf calculated outputted.**

So, if $F_{Beta(\alpha,\beta)}(X)$ is the cdf, then your function should output its value $F_{Beta(\alpha,\beta)}(x)$ at point x as a result of m Monte Carlo simulations (!).

Having defined the function, proceed to run it for the case of $\alpha = 2, \beta = 2$ and the following x values: $x = 0.1, 0.2, \ldots, 0.9$. Compare the estimates with the values returned by the *pbeta* function in R for the same x values.

Example. Some exemplary code:

```
> Beta.MC <- function(m,a,b,x){
  .. All you..
  .. You ARE ALLOWED to use 'rbeta' here to generate the beta random variables...
  .. This can literally be written in a couple of lines...
}
>
> a <- 2
> b <- 2
> m <- 10000
> Beta.MC(m,a,b,0.1)  # Evaluating cdf for Beta(a,b) at point x=0.1.
[1] 0.031
>
>
> # And that's how I expect your final output for x=0.1,0.2,...,0.9 to look like:
> # Your MC_Est_Beta numbers (resulting from your function) should be roughly close
> # to the Real_Beta numbers (resulting from $pbeta$ function)
...
              [,1]   [,2]   [,3]   [,4]   [,5]   [,6]   [,7]  [,8]   [,9]
x           0.1000 0.2000 0.3000 0.4000 0.5000 0.6000 0.7000 0.800 0.9000
Real_Beta   0.0280 0.1040 0.2160 0.3520 0.5000 0.6480 0.7840 0.896 0.9720
MC_Est_Beta 0.0296 0.0985 0.2141 0.3606 0.4951 0.6501 0.7828 0.895 0.9695
```