Bootstrapping and Jackknifing

Group Lanark

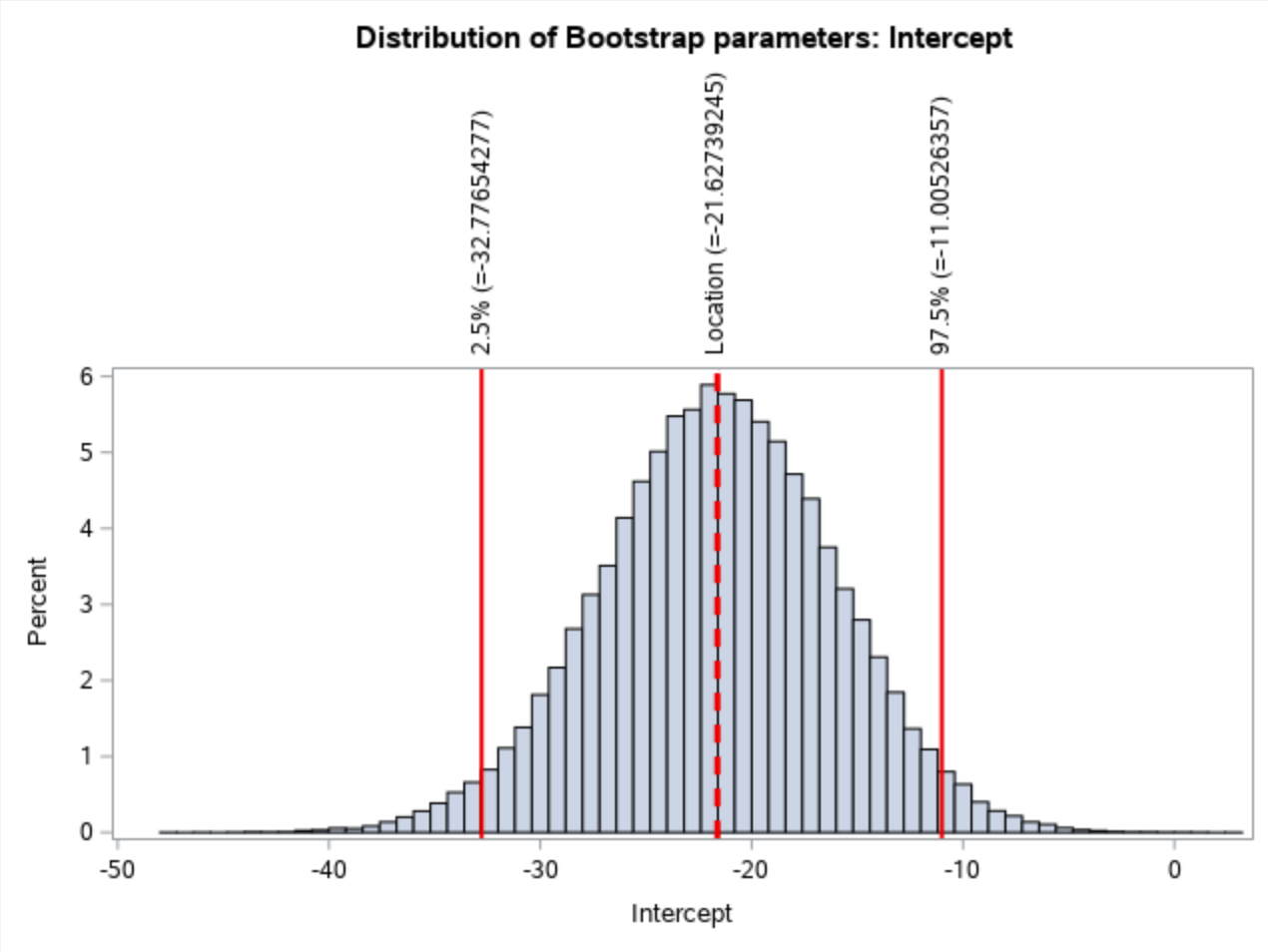
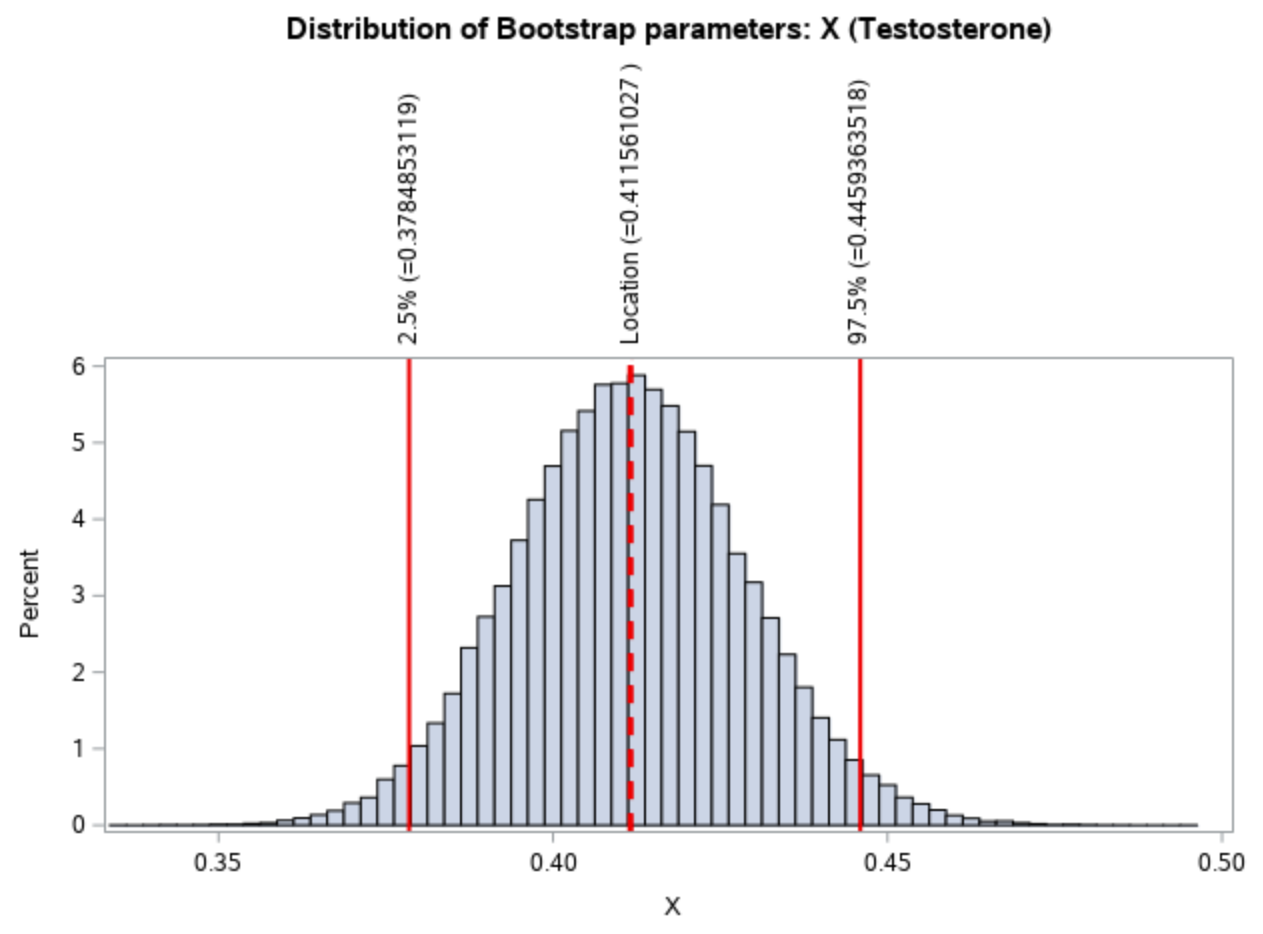
November 2022

## Bootstrapping

See the SAS code for the task below. In the two histograms below we used 100,000 bootstrap samples to approximate our parameters (Intercept and X, the explanatory variable).

The code (regressionBoot macro) used for this task was more efficient with a run time of approximately 0:00:08.18 for 100,000 samples. Compare this to the BootRes code provided which had a run time of around 0:00:17.24 only for 1000 samples. We can clearly see how inefficient the BootRes macro is for a fraction of the samples used for the regressionBoot code. Instead of resampling our data for each bootstrap sample, we resample our data all at once. We then calculate our statistic in question (regression) for all samples and then analysise all in a histogram. Moreover, we visualize the 95% confidence intervals of each parameter. Resampling in each loop and calculating their associated statistic causes the macro BootRes to run very slowly.

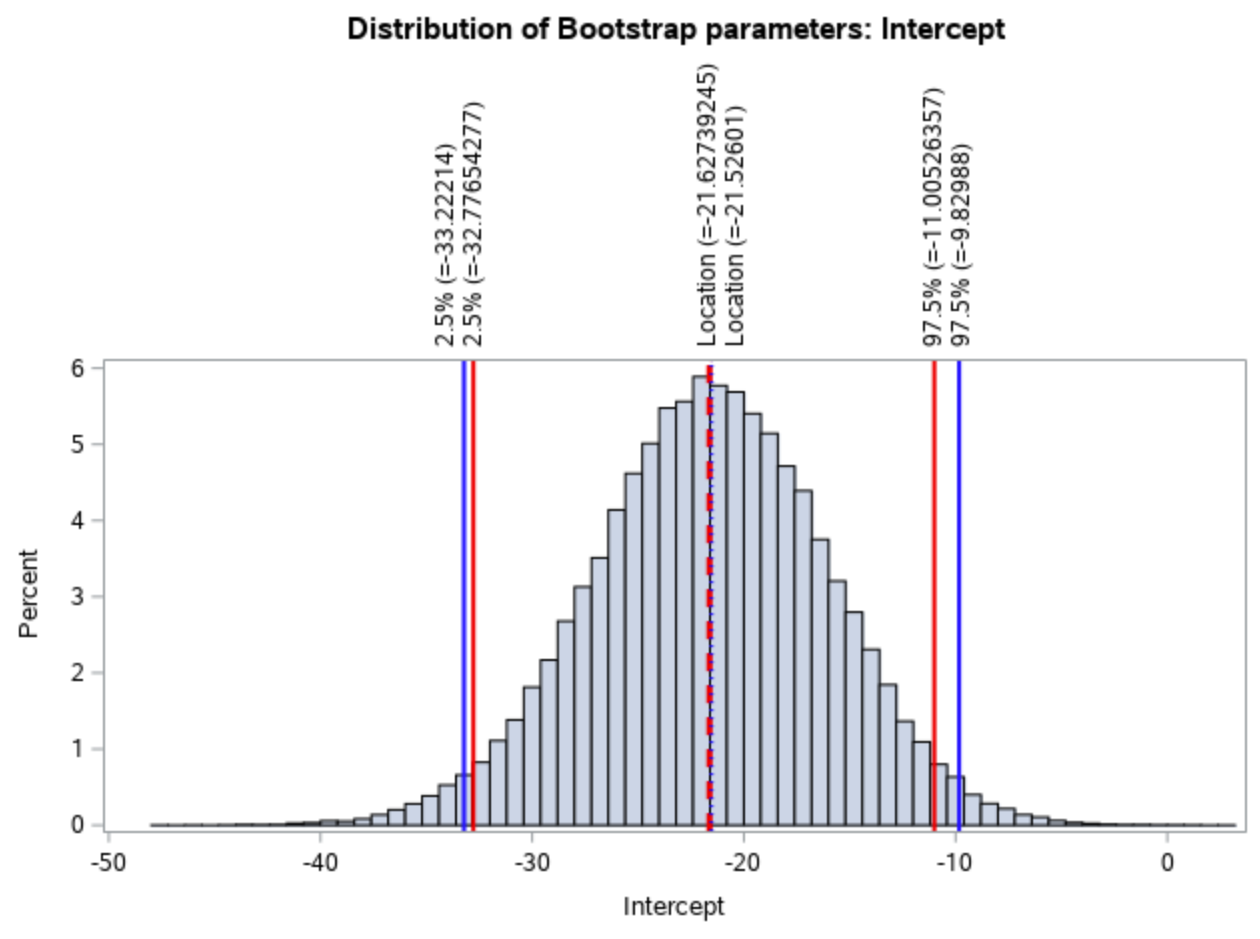
| Confidence Limit | Intercept parameter | X parameter |
| --- | --- | --- |
| Lower 2.5% | -32.77654 | 0.3784853 |
| Upper 97.5% | 11.00526 | - 0.4459364 |

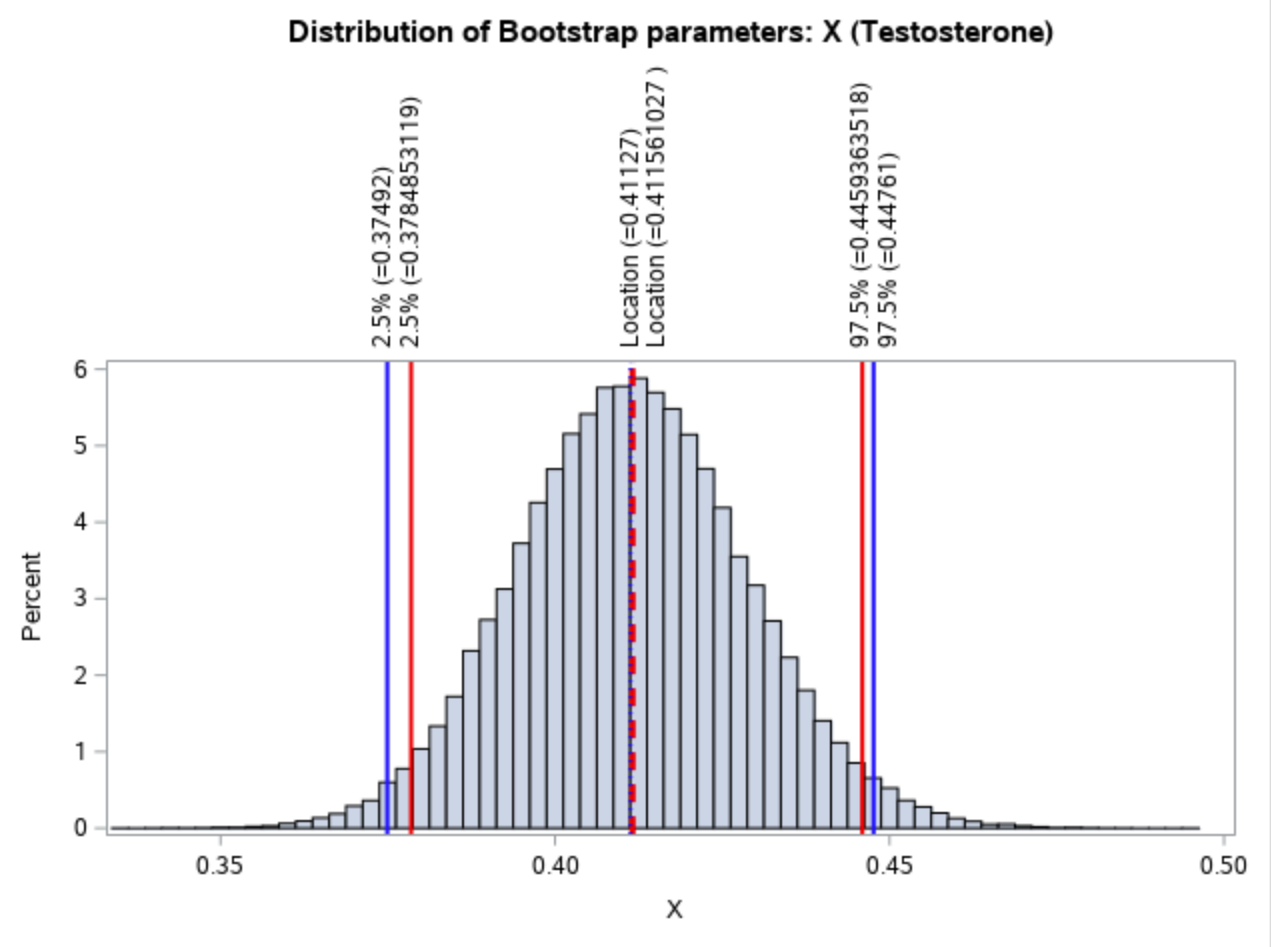
Let us now compare our bootstrapped results with the in-built 95% CIs. We overlay the CIs and the approximate parameter values in blue.

| Confidence Limit | Intercept parameter | X parameter |
| --- | --- | --- |
| Lower 2.5% | -33.22214 | 0.37492 |
| Upper 97.5% | -9.82988 | 0.44761 |

|  | Location |
| --- | --- |
| Intercept | -21.52601 |
| X | 0.41127 |



Histgram comparing the results for the intercept we obtained (red) against in-built SAS CI and location (blue).



Histgram comparing the results for the X parameter we obtained (red) against in-built SAS CI and location (blue).

### Code used

/\* Import file: \*/  
FILENAME REFFILE '/home/u62665966/sasuser.v94/Bootstrapping Group/seals.csv';  
  
PROC IMPORT DATAFILE=REFFILE  
 DBMS=CSV  
 OUT=SEALS2.IMPORT;  
 GETNAMES=YES;  
RUN;  
  
/\*   
  
 Task 2: Bootstrap (SAS)  
  
• Create a faster equivalent of the bootstrapping macro regBoot.sas. It only needs to work for one  
covariate. (done)  
• Use the seals data (seals.csv) to perform a regression of testosterone level (in µg/l) on length (in  
cm). This is a fictional dataset of male hormone levels in seals of different lengths.  
• State and visualise the 95% confidence intervals for the estimates of each parameter (intercept and  
slope). Provide a histogram for the distribution of each bootstrapped parameter. (done)  
  
• Compare regBoot.sas to your modified version to determine the speed-up.   
• Compare the boostrapped parameter estimates and their 95% confidence intervals to those obtained  
using the built-in SAS procedure.  
  
 Deliverables  
   
• Show your code and visualised results.  
• Discuss and interpret your comparative analysis.  
  
 Notes:  
  
 X - covariate : testosterone level  
 Y - response : lengths   
   
 From notes:  
  
 To produce an efficient bootstrap in SAS, we want to create all our bootstrap samples  
 first, and then run the analyses over these (as opposed to resample, analyse, resample,   
 analyse…).   
\*/  
   
  
   
 /\* Macro for bootstrapping of parameters: \*/  
   
%macro regressionBoot(NumSamples, DataSet);  
  
  
title "Bootstrap Distribution of Regression Estimates";  
title2 "Case Resampling";  
%let IntEst = -21.52601 ; \* exact estimates of the intercept;  
%let XEst = 0.41127; \* exact estimates of X - testosterone;  
   
/\* Generate our samples: (reps = number of samples wanted) \*/  
proc surveyselect data=SEALS2.IMPORT2 NOPRINT seed=314  
 out=BootCases(rename=(Replicate=SampleID))  
 method=urs /\* resample with replacement \*/  
 samprate=1 /\* each bootstrap sample has N observations \*/  
 reps=&NumSamples; /\* generate NumSamples bootstrap resamples \*/  
run;  
  
  
/\* Compute the statistic for EACH bootstrap sample \*/  
/\* eg we have size(Num\_samples) parameter estimations (PE):\*/  
proc reg data=BootCases outest=PEBoot NOPRINT; \*noprint so it does not show up in output;  
 by SampleID;  
 freq NumberHits;  
 model Y = X;  
run;quit;  
  
/\* Gives location and confidence intervals etc \*/  
proc stdize data=PEBoot vardef=N pctlpts=2.5 97.5 PctlMtd=ORD\_STAT outstat=Pctls;  
 var Intercept X;  
run;  
  
/\* Create changing macro variables - location of parameters and their CIs. \*/  
/\* Use CALL SYMPUT in a DATA step to assign the values to macro variables (used code from \*/  
/\* stackoverflow with minor edits) \*/  
data \_null\_;  
 set Pctls;  
 call symput('variable\_a\_'||left(\_n\_), left(Intercept));  
 call symput('variable\_b\_'||left(\_n\_), left(X));  
run;  
  
/\* The macro variables we will be using are below (note we do not use all): \*/  
  
/\* location of intercept: \*/  
%put &=variable\_a\_1;  
  
/\* location of X: \*/  
%put &=variable\_b\_1;  
  
/\* lower CI of Intercept: \*/  
%put &=variable\_a\_9;  
  
/\* upper CI of Intercept: \*/  
%put &=variable\_a\_10;  
  
/\* lower CI of X: \*/  
%put &=variable\_b\_9;  
  
/\* upper CI of X: \*/  
%put &=variable\_b\_10;  
  
  
/\* Visualize bootstrap distribution :   
 Histograms for each of the parameters  
   
Note that here we use the macro variables to indicate location of parameter estimate and  
the CIs of the parameters!!! \*/  
  
title 'Distribution of Bootstrap parameters: Intercept';  
 proc sgplot data=PEboot;  
 histogram intercept;  
 refline &variable\_a\_1 / axis=x lineattrs=(thickness=3 color=red pattern=dash) label = ("Location (=&variable\_a\_1)");  
/\* plot the confidence interval for intercept: \*/  
 refline &variable\_a\_9 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("2.5% (=&variable\_a\_9)");  
 refline &variable\_a\_10 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("97.5% (=&variable\_a\_10)");  
 run;  
  
title 'Distribution of Bootstrap parameters: X (Testosterone)';  
 proc sgplot data=PEboot;  
 histogram X;  
 refline &variable\_b\_1 / axis=x lineattrs=(thickness=3 color=red pattern=dash) label = ("Location (=&variable\_b\_1)");  
/\* plot the confidence interval for X: \*/   
 refline &variable\_b\_9 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("2.5% (=&variable\_b\_9)");  
 refline &variable\_b\_10 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("97.5% (=&variable\_b\_10)");  
run;  
  
  
/\* select the CI (gives a table of the CI for parameters) need this in macro output \*/  
title 'Distribution of Bootstrap parameters: Intercept and X';  
proc report data=Pctls nowd;  
 where \_type\_ =: 'P';  
 label \_type\_ = 'Confidence Limit';  
 columns ('Bootstrap Confidence Intervals' \_ALL\_);  
run;   
  
%mend regressionBoot;  
  
options nonotes;  
   
   
/\* Measure how long it takes to run this code: \*/  
   
 /\* Start timer \*/  
 %let \_timer\_start = %sysfunc(datetime());   
   
%regressionBoot(100000, SEALS2.Import);  
   
 /\* Stop timer \*/  
 data \_null\_;  
 dur = datetime() - &\_timer\_start;  
 put 30\*'-' / ' TOTAL DURATION:' dur time13.2 / 30\*'-';  
 run;   
   
/\* for 5000 samples: TOTAL DURATION: 0:00:00.66  
 for 100000 samples: TOTAL DURATION: 0:00:08.18 \*/  
   
   
/\* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ \*/  
   
 /\* Compare with code previously given: \*/  
   
  
%macro regBoot(NumberOfLoops, DataSet, XVariable, YVariable);  
  
/\*Number of rows in my dataset\*/  
 data \_null\_;  
 set &DataSet NOBS=size;  
 call symput("NROW",size);  
 stop;  
 run;  
  
/\*loop over the number of randomisations required\*/  
%do i=1 %to &NumberOfLoops;  
  
/\*Sample my data with replacement\*/  
 proc surveyselect data=&DataSet out=bootData seed=-3014 method=urs noprint sampsize=&NROW;  
 run;  
  
/\*Conduct a regression on this randomised dataset and get parameter estimates\*/  
 proc reg data=bootData outest=ParameterEstimates noprint;  
 Model &YVariable=&XVariable;  
 run;  
 quit;  
  
/\*Extract just the columns for slope and intercept for storage\*/  
 data Temp;  
 set ParameterEstimates;  
 keep Intercept &XVariable;  
 run;  
  
/\*Create a new results dataset if the first iteration, append for following iterations\*/  
 data ResultHolder;  
 %if &i=1 %then %do;  
 set Temp;  
 %end;  
 %else %do;  
 set ResultHolder Temp;  
 %end;  
 run;  
 %end;  
/\*Rename the results something nice\*/  
data ResultHolder;  
set ResultHolder;  
rename Intercept=RandomIntercept &XVariable=RandomSlope;  
run;  
%mend regBoot;  
  
options nonotes;  
  
  
/\*Run the macro, note this take comparatively longer than the the newer code!!!\*/  
  
 /\* Start timer \*/  
 %let \_timer\_start = %sysfunc(datetime());   
   
%regBoot(NumberOfLoops= 1000, DataSet=SEALS2.IMPORT, XVariable=testosterone, YVariable=lengths);  
  
 /\* Stop timer \*/  
 data \_null\_;  
 dur = datetime() - &\_timer\_start;  
 put 30\*'-' / ' TOTAL DURATION:' dur time13.2 / 30\*'-';  
 run;  
  
/\* for 500 samples: TOTAL DURATION: 0:00:08.80  
 for 1000 samples: TOTAL DURATION: 0:00:17.24\*/  
   
   
/\* $$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$ \*/  
  
 /\* Investigate using built-in SAS procedure: \*/  
  
data SEALS2.IMPORT2 (keep = X Y);  
 set SEALS2.IMPORT(rename=(lengths=Y testosterone=X));   
 \*rename lengths and testosterone as y and x (x is explanatory var and y is predicted var);  
run;  
   
/\* without bootstrapping the parameter values are: \*/  
proc reg data=SEALS2.IMPORT2;  
 model Y = X / CLB; \*gives the 95% confidence limits for parameters;  
run;quit;   
   
/\*   
See the 95% CI of the parameters:  
Confidence Limit Intercept | X  
lower 2.5 -33.22214 | 0.37492  
upper 97.5 -9.82988 | 0.44761   
  
Locations:  
Intercept ~ -21.52601   
 X ~ 0.41127  
\*/   
  
/\* Set new macro variables: \*/  
%let Intlwr = -33.22214 ; \* lwr CI of Intercept;  
%let Intupr = -9.82988; \* upr CI of Intercept;  
%let IntLoc = -21.52601; \* location of estimate;  
  
%let Xlwr = 0.37492 ; \* lwr CI of X;  
%let Xupr = 0.44761; \* upr CI of X;  
%let XLoc = 0.41127; \* location of estimate;  
  
  
/\* Let us add these on top the histograms previously plotted: \*/  
  
title 'Distribution of Bootstrap parameters: Intercept';  
 proc sgplot data=PEboot;  
 histogram intercept;  
 refline &variable\_a\_1 / axis=x lineattrs=(thickness=3 color=red pattern=dash) label = ("Location (=&variable\_a\_1)");  
 refline &IntLoc / axis=x lineattrs=(thickness=2.5 color=blue pattern=dot) label = ("Location (=&IntLoc)");  
  
/\* plot the confidence interval for intercept: \*/  
 refline &variable\_a\_9 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("2.5% (=&variable\_a\_9)");  
 refline &variable\_a\_10 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("97.5% (=&variable\_a\_10)");  
   
 refline &Intlwr / axis=x lineattrs=(thickness=2.5 color=blue pattern=solid) label = ("2.5% (=&Intlwr)");  
 refline &Intupr / axis=x lineattrs=(thickness=2.5 color=blue pattern=solid) label = ("97.5% (=&Intupr)");  
run;  
  
title 'Distribution of Bootstrap parameters: X (Testosterone)';  
 proc sgplot data=PEboot;  
 histogram X;  
 refline &variable\_b\_1 / axis=x lineattrs=(thickness=3 color=red pattern=dash) label = ("Location (=&variable\_b\_1)");  
 refline &XLoc / axis=x lineattrs=(thickness=2.5 color=blue pattern=dot) label = ("Location (=&XLoc)");  
  
/\* plot the confidence interval for X: \*/   
 refline &variable\_b\_9 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("2.5% (=&variable\_b\_9)");  
 refline &variable\_b\_10 / axis=x lineattrs=(thickness=2 color=red pattern=solid) label = ("97.5% (=&variable\_b\_10)");  
  
 refline &Xlwr / axis=x lineattrs=(thickness=2.5 color=blue pattern=solid) label = ("2.5% (=&Xlwr)");  
 refline &Xupr / axis=x lineattrs=(thickness=2.5 color=blue pattern=solid) label = ("97.5% (=&Xupr)");   
run;

## Jackknifing

See annotated code below for the implementation of the Jackknifing method in SAS.

We found that the average lengths using the jackknife estimate was 109.62 cm with a standard error (SE) of 11.03 cm. Calculating these statistics analytically we obtained a mean length of 110.72 cm with SE = 5.5 cm. Although the means are relatively the same, the SEs of the analytic and jackknifing methods differ (jackknife mean approximately twice as large sample mean). This is generally the because “the conservative property of the jackknife estimator” [1], and hence it will produce larger SEs. Inaccurate results can also arise if the data which is being estimated is not linear. However, in our case, the data is linear.

### Code used

/\*Import data/ read in file:\*/  
FILENAME REFFILE '/home/u62665966/sasuser.v94/Jack Knifing/seals.csv';  
  
PROC IMPORT DATAFILE=REFFILE  
 DBMS=CSV  
 OUT=SEALS.IMPORT;  
 GETNAMES=YES;  
RUN;  
  
/\*View the whole data set:\*/  
proc print data=seals.import (obs=100);  
run;  
  
/\* ------------------------------------------------------------------------------------- \*/  
  
/\* Task:  
  
Write and implement code (modifying code already given to you in the lecture notes e.g. the  
two sample randomisation test), to obtain a jackknife estimate for the standard error of the   
mean for seal body length, using the seals data set (seals.csv).  
  
I.e., sampling without replacement.  
  
\*/  
  
/\* ------------------------------------------------------------------------------------- \*/  
  
DATA seals.import\_lengths;   
SET seals.import;  
Keep Lengths; \*keep lengths column, drop the other one (not needed);  
RUN;  
  
/\* View: \*/  
proc print data=seals.import\_lengths (obs=100);  
run;  
  
DATA seals.import\_lengths;   
SET seals.import\_lengths;  
RENAME lengths=Original\_Lengths;  
RENAME original\_lengths=Jackknife\_0; \*rename original data as Jackknife\_0 (did this  
 because the loops below will be easier to   
 implement);  
RUN;  
  
DATA TestSettingCombining;  
SET seals.import\_lengths seals.import\_lengths seals.import\_lengths;   
RUN; \*Combine columns from the two data sets;  
  
  
  
data seals.import\_Jack\_100copies (drop=j);  
set seals.import\_lengths;  
array Jackknife\_[100]; \*define array;  
 do j = 1 to 100; \*create columns using loop;  
 Jackknife\_[j] = Jackknife\_0; \*100 data columns with same (original) data;  
 end; \*we have 101 total columns (Jackknife\_0, Jackknife\_1 -   
 Jackknife\_100);  
run;  
  
data seals.import\_Jack\_Diag (drop=i);  
set seals.import\_Jack\_100copies;  
array Jackknife Jackknife\_1 -- Jackknife\_100; \*apply loop over all columns;  
 do i=1 to dim(jackknife);  
 if \_n\_ = i then jackknife[i] = 0; \*if row = column replace entry by 0;  
 end; \*creates 0s across diagonals;  
run;  
  
  
/\* take transpose \*/  
PROC TRANSPOSE DATA=seals.import\_Jack\_Diag OUT=seals.import\_Jack\_Transpose;  
VAR Jackknife\_0-Jackknife\_100; \*transpose the data to take mean (row-wise);  
RUN; \*columns name go from COL1 to COL100;  
  
/\* calculate row wise mean: \*/  
data seals.import\_Jack\_Mean ;  
 set seals.import\_Jack\_Transpose;  
 Rename \_NAME\_ = Sample; \*rename column as sample (nicer name);  
 Means = mean(of Col1 - Col100); \*calculate the mean over all columns (row-wise);  
run;  
  
/\* Calculate standard error using this: \*/  
DATA seals.import\_Jack\_OnlyMean;   
SET seals.import\_Jack\_Mean;  
KEEP Means; \*only use the means column - need this for SE;  
RUN;  
  
  
  
 /\* Calculate Standard Error for Mean: \*/  
   
data seals.import\_Jack\_Square;  
set seals.import\_Jack\_OnlyMean;  
Means1 = 110.71628445; \*manually take the first observation's mean  
 (where we did not remove any observations, i.e., Jackknife\_0);  
Diff = Means-Means1; \*store the differences in new column, Diff;  
Square = Diff\*\*2; \*square the differences and store in new column, Square;  
run;  
  
proc means data=seals.import\_Jack\_Square sum;  
 variable Square; \*calculate the sums of the column, Square;  
run;  
/\* Sum(Square) = 122.8845513 \*/  
  
data seals.import\_Jack\_SE;  
set seals.import\_Jack\_Square;  
Sum = 122.8845513; \*we manually take the sum;  
SE = sqrt((99/100)\*Sum); \*calculate the rest of the formula, where n=100, store in SE;  
run;  
/\* SE ~ 11.029764539 \*/  
  
  
DATA seals.import\_Jack\_SE;   
SET seals.import\_Jack\_SE;  
KEEP SE; \*keep only the SE column;  
rename SE = Standard\_Error; \*rename appropriately;  
RUN;  
  
/\*Look at the SE:\*/  
proc print data=seals.import\_Jack\_SE (obs=1); \*keep the first observation (note that all   
 are the same in the column);  
run;  
  
  
/\* ------------------------------------------------------------------------------------- \*/  
  
 /\* Calculate analytical standard error \*/  
  
  
data seals.import\_Jack\_AnalyticalSE;  
set seals.import\_Jack\_Diag;  
keep Jackknife\_0; \*keep the original lengths;   
run;  
  
data seals.import\_Jack\_AnalyticalSE;  
set seals.import\_Jack\_AnalyticalSE;  
MeanJack\_0 = 110.71628445; \*manually write the mean;  
Diff = Jackknife\_0 - MeanJack\_0; \*find the difference, store in Diff;  
Square = Diff\*\*2; \*square it and store in Square;  
run;  
  
/\* find the sum manually: \*/  
proc means data=seals.import\_Jack\_AnalyticalSE sum;  
 variable Square; \*calculate the sums of the column, Square;  
run;  
/\* Sum(Square) = 3035.96 \*/  
  
data seals.import\_Jack\_AnalyticalSE;  
set seals.import\_Jack\_AnalyticalSE;  
Sum = 3035.96;   
Standard\_error = sqrt((1/100)\*Sum);   
keep Standard\_Error;   
run;  
  
/\*Look at the SE:\*/  
proc print data=seals.import\_Jack\_AnalyticalSE (obs=1); \*keep the first observation (note that all   
 are the same in the column);  
run;  
  
  
  
/\* Standard Error is 5.50995 which is smaller than for the Jackknife sample (11.029764539)\*/  
  
  
/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \*/  
  
/\* Compare the mean of the original data to the average using Jackknifing \*/  
  
  
/\* Mean from sample = 110.71628445  
 Mean using Jackknifing = 109.6201 (see code below for calculation)  
\*/  
  
proc sql;  
 select avg(Means) as Mean\_Jackknife  
 from seals.import\_Jack\_OnlyMean;  
quit;  
  
/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \*/  
  
  
  
/\* Note: from SAS website: (https://support.sas.com/kb/24/982.html)  
   
 The Jackknife:  
  
The jackknife works only for statistics that are smooth functions of the data. Statistics  
that are not smooth functions of the data, such as quantiles, may yield inconsistent   
jackknife estimates. The best results are obtained with statistics that are linear   
functions of the data. For highly nonlinear statistics, the jackknife can be inaccurate.  
  
\*/  
  
  
/\* relationship appears linear: \*/  
  
proc plot data=SEALS.IMPORT;  
 plot lengths\*testosterone;  
 title 'Lengths against testosterone';  
run;  
  
ods graphics on;  
proc reg data =SEALS.IMPORT alpha = 0.05 plots(only)=(diagnostics residuals fitplot observedbypredicted);  
 model lengths = testosterone;  
run;  
ods graphics off;  
/\* -appears linera  
 -appears normally distributed  
 -residuals exhibit no patterns  
\*/

## References

[1]

Hansen, B., Chesher, A., Chiang, H., Hillier, G., Ibragimov, R., Mackinnon, J., Müeller, U., Nielsen, M., Paolella, M., Phillips, P. and Welz, T. (2022). Jackknife Standard Errors for Clustered Regression. [online] Available at: <https://ssc.wisc.edu/~bhansen/papers/tcauchy.pdf> [Accessed 25 Nov. 2022].