GitHub repositories: Shiny app: <https://github.com/samrajsingh4/MT5763_Shiny.git>

SAS codes: <https://github.com/NavarroJulia/MT5763_Lanark.git>

**MT5763: Software for Data Analysis Group project**

Group Lanark - 2022

**Shiny App (Stock Analysis App)**

<https://samrajsingh4.shinyapps.io/mt5763_shiny/>

This app allows users to analyse and compare the stock prices of selected companies, as well as download and view the price data on the selected stocks. It also displays the biggest gainers and losers of the stock market on a given day and allows users to view and download data on the gainers and losers. The data on biggest gainers and losers, as well as the rest of the app is refreshed every hour.

The app retrieves real-time stock data from Yahoo Finance using the httr and rsconnect packages. It also uses the tidyquant and tidyverse packages to manipulate and clean the data.

The app, moreover, allows users to input a date range and select up to 4 stocks to compare. It then displays candlestick charts of the selected stocks. It also displays the top 3 biggest gainers and losers of the stock market on the given date, and gives the gainers and losers price data for the last 6 months, as well as in candlestick price chart form.

## How to Use

* To run the app, you require the files: server.R, UI.R, and global.R.
* Run the app in R.
* Input the date range and select the stocks you want to analyse or use the three pre-selected stocks to generate charts.
* View and download data on the selected stocks and the biggest gainers and losers of the stock market.
* Compare the stock prices of the selected stocks using candlestick charts.
* Press the refresh button to refresh session and update API data on biggest gainers and losers.

## Inputs

* Date Range: the date range for which you want to view the stock prices of the selected companies, going back up to 1 year.
* Stocks: The stocks you want to analyse. You can select up to 4 stocks from the dropdown menu.
* Generate Button: Generates candlestick price charts on selected stocks with respect to chosen date.
* Refresh Button: Refreshes entire app and data when pressed.

Chart

Description automatically generated

*Screenshot 1: Biggest daily stock gainers.*

Chart

Description automatically generated

*Screenshot 2: Biggest daily stock losers.*

Chart

Description automatically generated

*Screenshot 3: Selected stock price charts.*

## Outputs

* Candlestick Charts: Candlestick charts displaying the stock prices for the selected companies and the top 3 biggest gainers and losers of the stock market on the given date.
* Selected Stock Data: A table displaying data on the selected stocks.
* Biggest Gainers Data: A table displaying data on the top 3 biggest gainers of the stock market on the given date.
* Biggest Losers Data: A table displaying data on the top 3 biggest losers of the stock market on the given date.

Table

Description automatically generated with medium confidence

Table

Description automatically generated

Table

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

*Screenshot 4: Data corresponding to the previous screenshots (1-3).*

## Downloadable Files

* Selected Stock Data: A CSV file containing data on the selected stocks.
* Biggest Gainers Data: A CSV file containing data on the top 3 biggest gainers of the stock market on the given date.
* Biggest Losers Data: A CSV file containing data on the top 3 biggest losers of the stock market on the given date.

## Bootstrapping (Lengths of seals)

See the SAS code for the bootstrap task below. In the two histograms in this section 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 regBoot code provided which had a run time of around 0:00:17.17 only for 1000 samples. We can clearly see how inefficient the regBoot macro is for a fraction of the samples used for the regressionBoot code.

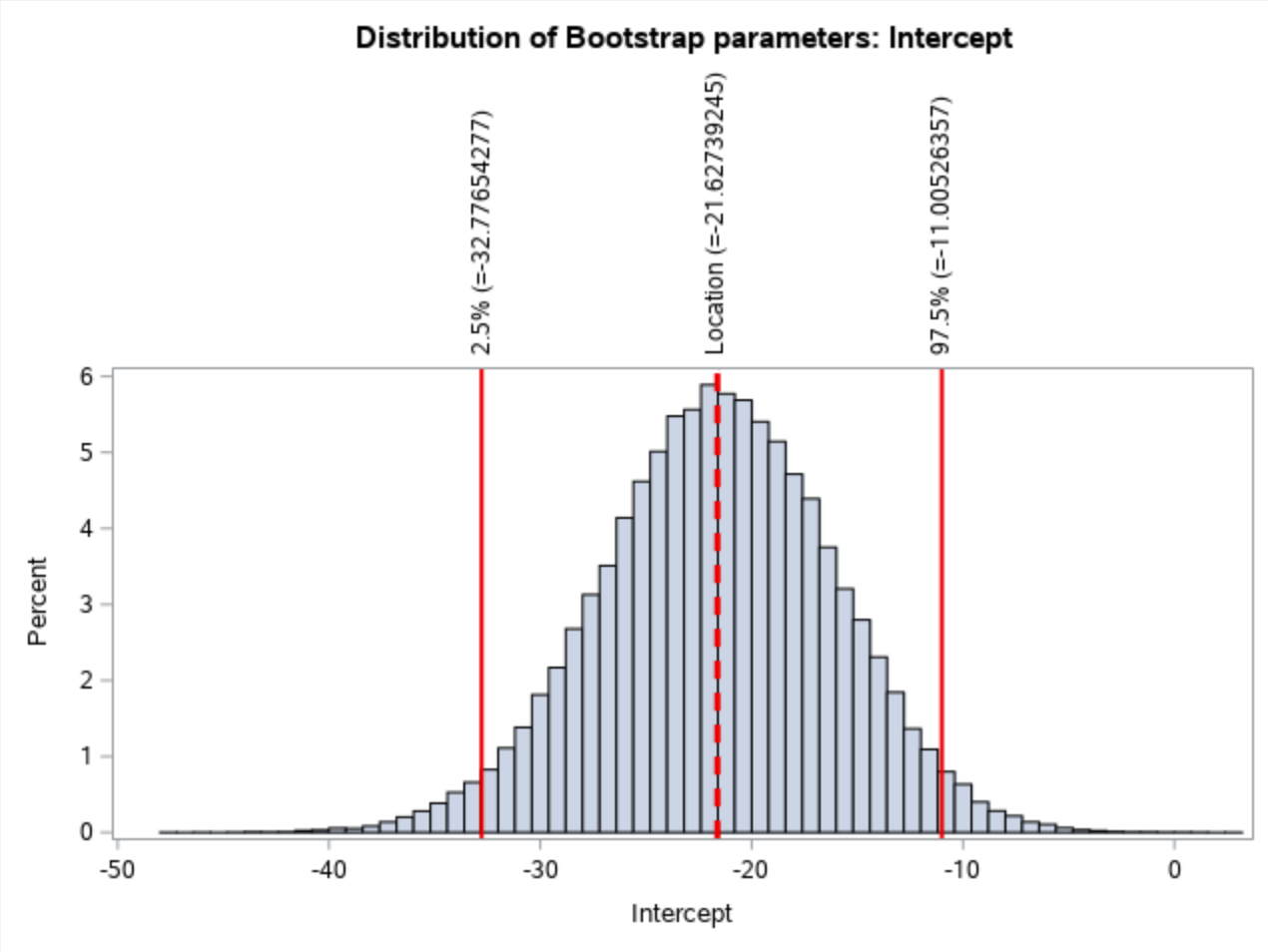
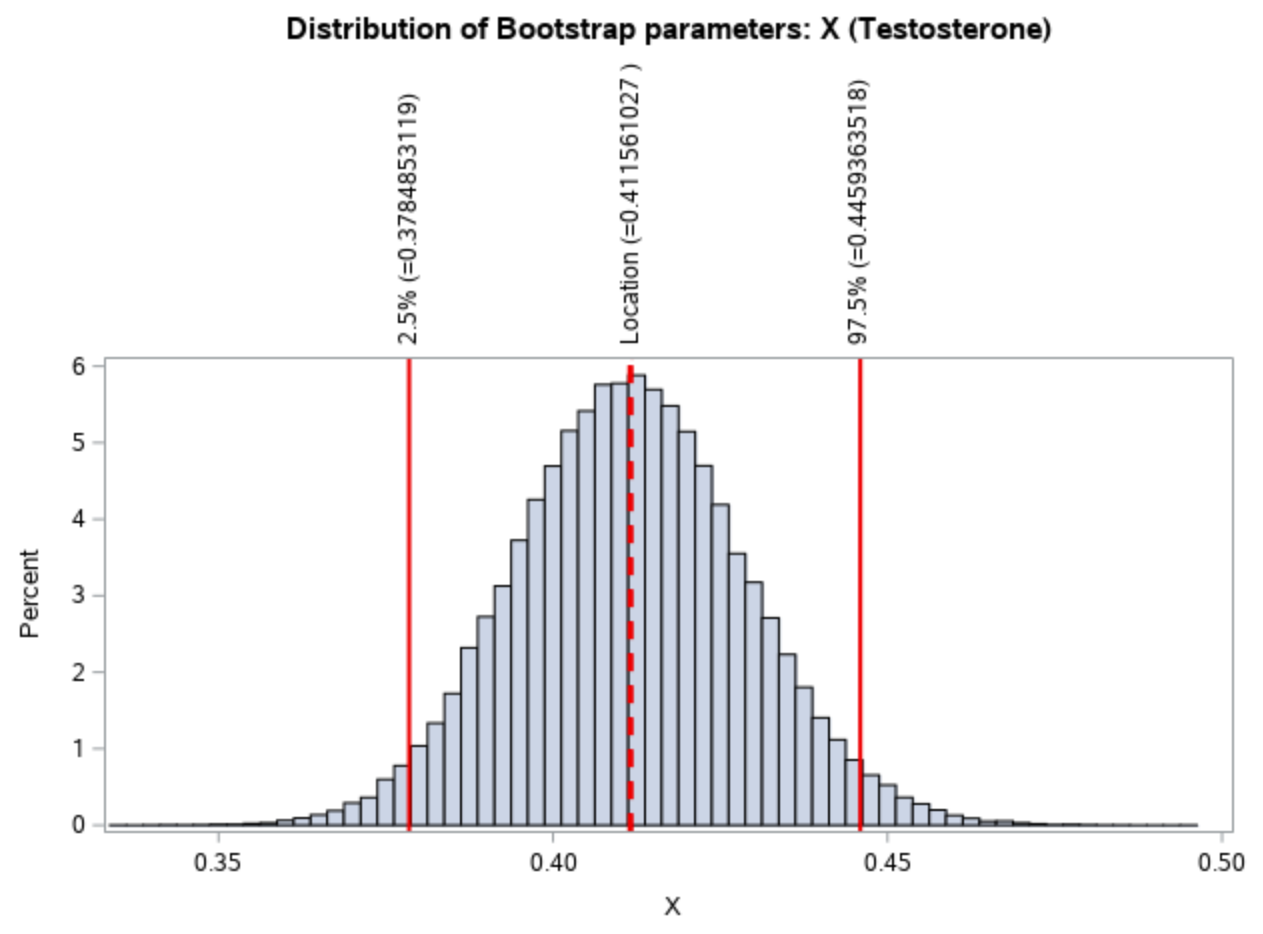
In the case for 1,000 samples the older macro took around 34 times longer to run compared to the newer macro. The speed-up is mainly achieved by noticing a huge flaw in the old macro: instead of resampling our data for each bootstrap sample, we resample our data all at once. We then calculate our statistic in question (regression) for all samples and then visualize observations all in a histogram. Moreover, we plot the 95% confidence intervals of each parameter. Resampling in each loop and calculating their associated statistic causes the macro regBoot to run very slowly. (Note that it would take a very long time running the older macro (regBoot) with significantly larger samples)

|  |  |  |
| --- | --- | --- |
| Number of samples | New code (time in secs) | Old code (time in secs) |
| 100 | 0.54 | 1.73 |
| 500 | 0.51 | 8.53 |
| 1000 | 0.50 | 17.17 |
| 2500 | 0.63 | 43.00 |
| 5000 | 0.96 | 87.05 |

Confidence intervals of the improved bootstrap code read:

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

The location and confidence intervals for 100,000 samples plotted for the two parameters are:

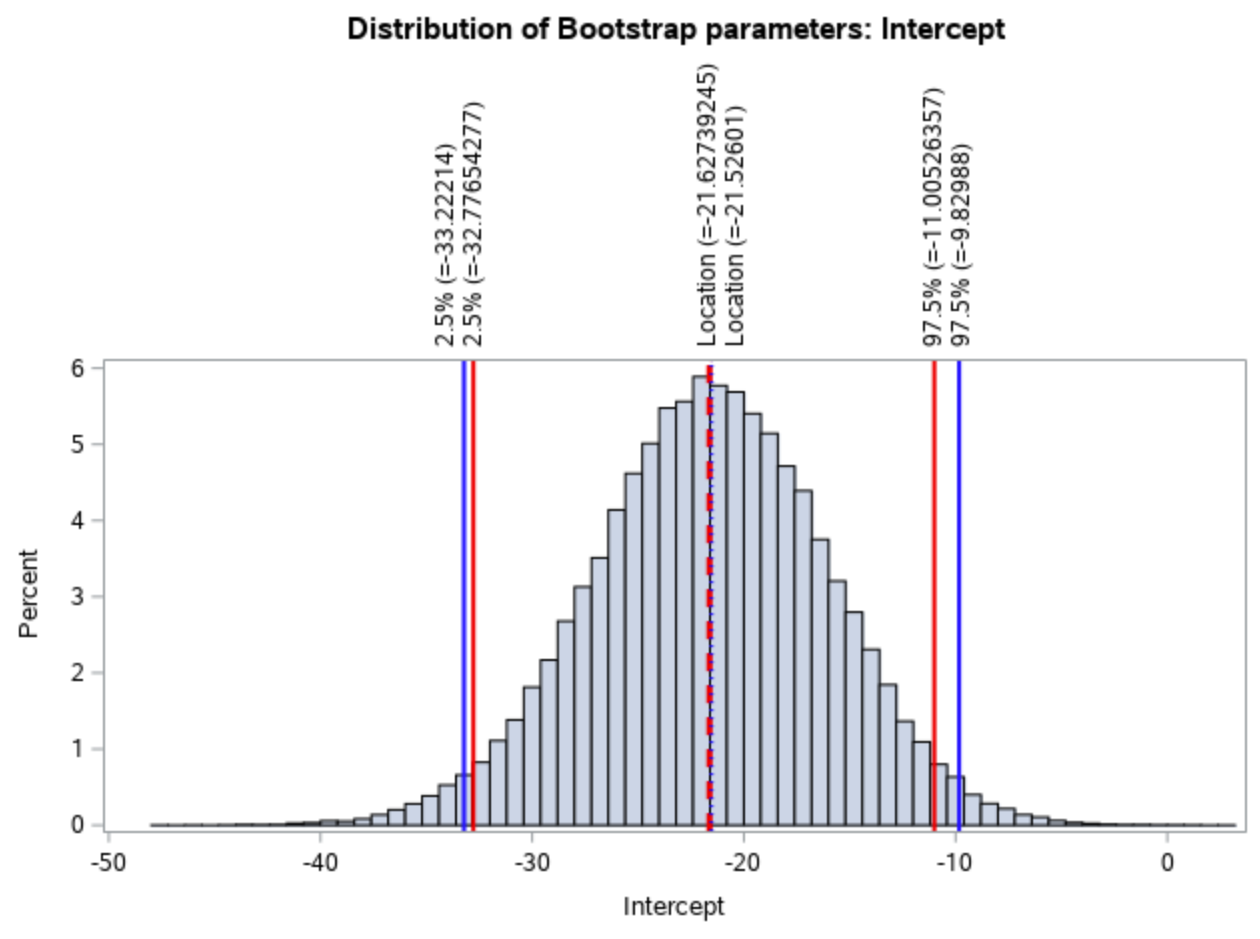
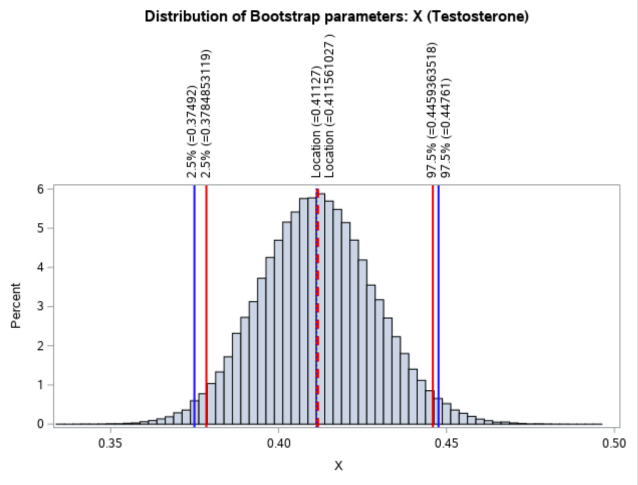
 

Let us now compare our bootstrapped results with the in-built 95% CIs. We overlay the CIs and the approximate parameter values in blue.

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

And we include the locations in blue too:

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



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

### Code used:

/\* \*/

/\* \*/

/\* Please create a library called SEALS2 and add the path to this folder (where code is).

And add csv file in the folderwhere code is located (under server files & folder, under

sasuser.v94)

\*/

FILENAME REFFILE '/home/u62665966/sasuser.v94/Bootstrapping Group/seals.csv';

PROC IMPORT DATAFILE=REFFILE

DBMS=CSV

OUT=SEALS2.IMPORT;

GETNAMES=YES;

RUN;

/\* Main flow of the code (run in order):

----PART 1:----

Investigate the SAS CI for the data set and define macro parameters (lines 56 - 64)

-set MACROS: (lines 84 - 90)

----PART 2:----

Create the macro (faster version)

-please run the macro (lines 107 - 248)

-then run line 256 (without timer)

-or line 267 - 275 (with timer)

--> This will result in two histogram outputs and a table of values

----PART 3:----

Code for the old macro (lines 296 - 341)

-run without timer (line 348)

-with timer (lines 358 - 366)

+

+

+

+

\*/

/\*-----------------------------------------------------------------------------------------\*/

/\*-------------------------------------PART 1:---------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

/\* 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

And 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;

/\* Will be used at the end for the last two histograms... \*/

/\*-----------------------------------------------------------------------------------------\*/

/\*---------------------------------------PART 2:-------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

/\*

Task 2: Bootstrap (SAS)

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=&DataSet 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;

/\* Here we add the in build CIs with the bootstrapped ones

Let us add these on top the histograms previously plotted:

NOTE:

- Bootstrapped CIs and parameters are in RED,

- SAS CIs and parameters are in BLUE.

\*/

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;

%mend regressionBoot;

options nonotes;

/\*-----------------------------------------------------------------------------------------\*/

/\* Run code without timer: \*/

%regressionBoot(100000, SEALS2.Import2);

/\*-----------------------------------------------------------------------------------------\*/

/\* Run code WITH timer:

Measure how long it takes to run this code: \*/

/\* Start timer \*/

%let \_timer\_start = %sysfunc(datetime());

%regressionBoot(100000, SEALS2.Import2);

/\* Stop timer \*/

data \_null\_;

dur = datetime() - &\_timer\_start;

put 30\*'-' / ' TOTAL DURATION:' dur time13.2 / 30\*'-';

run;

/\*-----------------------------------------------------------------------------------------\*/

/\* Times observed: \*/

/\* for 5000 samples: TOTAL DURATION: 0:00:00.66

for 100000 samples: TOTAL DURATION: 0:00:08.18 \*/

/\*-----------------------------------------------------------------------------------------\*/

/\*---------------------------------------PART 3:-------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

/\* 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 without timer: \*/

%regBoot(NumberOfLoops= 1000, DataSet=SEALS2.IMPORT, XVariable=testosterone, YVariable=lengths);

/\*-----------------------------------------------------------------------------------------\*/

/\* Run the macro WITH timer: /\*

/\* 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;

/\*-----------------------------------------------------------------------------------------\*/

/\* Note the times: \*/

/\* for 500 samples: TOTAL DURATION: 0:00:08.80

for 1000 samples: TOTAL DURATION: 0:00:17.24\*/

## Jackknifing (Lengths of seals)

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 because of “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:

/\* \*/

/\* Please create a library called SEALS and add the path to this folder (where code is).

And add csv file in the folderwhere code is located (under server files & folder, under

sasuser.v94)

\*/

FILENAME REFFILE '/home/u62665966/sasuser.v94/Jack Knifing/seals.csv';

PROC IMPORT DATAFILE=REFFILE

DBMS=CSV

OUT=SEALS.IMPORT;

GETNAMES=YES;

RUN;

/\* Main flow of the code (run in order):

----PART 1:----

Perform Jackknife method in SAS and obtain the SE for the mean

- run code from lines 42 - 131,

- SE for the mean is on line 135 - 137

----PART 2:----

Calculating the SE for the mean without the jackknife method

- run code from lines 149 - 175,

- SE is on lines 180 - 182

----PART 3:----

Compare the means: lines 197 - 198

-> show data is linear: lines 211 - 214

\*/

/\*-----------------------------------------------------------------------------------------\*/

/\*-------------------------------------PART 1:---------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

DATA seals.import\_lengths;

SET seals.import;

Keep Lengths; \*keep lengths column, drop the other one (not needed);

RUN;

DATA seals.import\_lengths;

SET seals.import\_lengths;

RENAME lengths=Jackknife\_0; \*rename original data as Jackknife\_0 (did this

because the loops below will be easier to

implement);

RUN;

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;

/\* = 11.0298 \*/

/\*-----------------------------------------------------------------------------------------\*/

/\*-------------------------------------PART 2:---------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

/\* 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)\*/

/\*-----------------------------------------------------------------------------------------\*/

/\*-------------------------------------PART 3:---------------------------------------------\*/

/\*-----------------------------------------------------------------------------------------\*/

/\* 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;

/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \*/

/\* Relationship appears linear: \*/

proc plot data=SEALS.IMPORT;

plot lengths\*testosterone;

title 'Lengths against testosterone';

run;

/\* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ \*/

## 

## 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].