

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 app retrieves real-time stock data from Yahoo Finance using the httr and rconnect 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.



Screenshot 1: Biggest daily stock gainers.



Screenshot 2: Biggest daily stock losers.



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.

symbol	date	open	high	low	close	volume	adjusted
VVNT	2022-06-08	5.67	5.67	5.25	5.36	529700	5.36
VVNT	2022-06-13	5.22	5.24	4.36	4.37	1598700	4.37
VVNT	2022-06-13	4.15	4.21	3.725	3.96	1725100	3.96
VVNT	2022-06-14	3.98	4.027	3.72	3.99	799100	3.99
VVNT	2022-06-15	4.03	4.4	3.74	4.11	1571300	4.11
VVNT	2022-06-16	4.02	4.02	3.64	3.62	1099800	3.62
VVNT	2022-06-17	3.78	4.08	3.78	3.97	905100	3.97
VVNT	2022-06-21	4.08	4.56	3.83	3.96	1003800	3.96
VVNT	2022-06-22	3.96	4.01	3.77	3.82	823400	3.82
VVNT	2022-06-23	3.81	3.86	3.6	3.62	1108500	3.62
VVNT	2022-06-24	3.89	4.01	3.78	3.80	882500	3.80
VVNT	2022-06-27	3.86	4.01	3.79	3.88	568800	3.88
VVNT	2022-06-28	3.80	3.91	3.64	3.65	693800	3.65
VVNT	2022-06-29	3.51	3.51	3.26	3.49	678800	3.49
VVNT	2022-06-30	3.58	3.49	3.305	3.48	497000	3.48
VVNT	2022-07-01	3.48	3.578	3.38	3.51	583800	3.51
VVNT	2022-07-05	3.47	3.69	3.29	3.69	431800	3.69
VVNT	2022-07-06	3.65	3.76	3.68	3.69	458800	3.69
VVNT	2022-07-07	3.71	3.84	3.83	3.81	483200	3.81
VVNT	2022-07-08	3.77	3.94	3.716	3.84	505500	3.84
VVNT	2022-07-11	3.8	3.83	3.65	3.69	369800	3.69
VVNT	2022-07-12	3.65	3.82	3.85	3.79	321500	3.79
VVNT	2022-07-13	3.65	3.85	3.68	3.77	379400	3.77
VVNT	2022-07-14	3.69	3.86	3.66	3.6	481800	3.6
VVNT	2022-07-15	3.9	4.5	3.84	4.06	663800	4.06
VVNT	2022-07-18	4.12	4.23	3.88	3.97	317800	3.97
VVNT	2022-07-19	4.06	4.3	4.08	4.21	569100	4.21
VVNT	2022-07-20	4.24	4.68	4.24	4.54	693300	4.54
VVNT	2022-07-21	4.51	4.665	4.48	4.63	313200	4.63
VVNT	2022-07-22	4.67	4.69	4.32	4.43	318800	4.43
VVNT	2022-07-26	4.45	4.68	4.384	4.55	593800	4.55
VVNT	2022-07-26	4.66	4.77	4.55	4.58	434800	4.58

Biggest Losers Stock Data									
symbol	date	open	high	low	close	volume	adjusted		
TLRY	2022-05-09	3.87	3.89	3.63	3.63	2887200	3.63		
TLRY	2022-05-10	3.59	3.57	3.37	3.38	2704700	3.38		
TLRY	2022-05-13	3.21	3.33	3.11	3.12	2728600	3.12		
TLRY	2022-05-14	3.15	3.37	3.1	3.18	3130600	3.18		
TLRY	2022-05-15	3.19	3.26	3.04	3.2	3084800	3.2		
TLRY	2022-05-16	3.11	3.14	3.02	3.11	2502400	3.11		
TLRY	2022-05-17	3.1	3.28	3.08	3.2	1955500	3.2		
TLRY	2022-05-21	3.3	3.37	3.25	3.27	1634700	3.27		
TLRY	2022-05-22	3.2	3.25	3.19	3.21	1551800	3.21		
TLRY	2022-05-23	3.23	3.28	3.14	3.28	2331000	3.28		
TLRY	2022-05-24	2.85	2.72	3.31	2.67	2181900	3.69		
TLRY	2022-05-27	3.63	3.69	3.51	3.64	1251400	3.64		
TLRY	2022-05-28	3.66	3.72	3.61	3.66	1777300	3.66		
TLRY	2022-05-29	3.42	3.43	3.27	3.31	1151300	3.31		
TLRY	2022-06-01	3.26	3.27	3.37	3.12	2388700	3.12		
TLRY	2022-07-01	3.12	3.22	3.06	3.18	1240100	3.18		
TLRY	2022-07-05	3.08	3.24	3	3.24	1857600	3.24		
TLRY	2022-07-06	3.22	3.54	3.22	3.5	2170200	3.5		
TLRY	2022-07-07	3.47	3.69	3.38	3.68	1584600	3.68		
TLRY	2022-07-08	3.49	3.63	3.41	3.43	1557300	3.43		
TLRY	2022-07-11	3.38	3.45	3.15	3.15	1288500	3.15		
TLRY	2022-07-12	3.16	3.3	3.13	3.22	1334200	3.22		
TLRY	2022-07-13	3.17	3.38	3.11	3.32	1185100	3.32		
TLRY	2022-07-14	3.17	2.87	3.08	3.09	6408800	3.09		
TLRY	2022-07-15	3.66	3.72	3.29	3.3	3902100	3.3		
TLRY	2022-07-18	3.37	3.68	3.31	3.4	3824000	3.4		
TLRY	2022-07-19	3.45	3.67	3.31	3.58	4315600	3.58		
TLRY	2022-07-20	3.55	4.09	3.53	3.54	6451400	3.54		
TLRY	2022-07-21	4.12	4.16	3.72	3.75	5284100	3.75		
TLRY	2022-07-22	3.73	3.78	3.49	3.5	2453400	3.5		
TLRY	2022-07-25	3.54	3.65	3.37	3.54	2788700	3.54		
TLRY	2022-07-26	3.49	3.58	3.36	3.38	1938800	3.38		

Selected Stock Data									
symbol	date	open	high	low	close	volume	adjusted		
GOOG	2022-12-13	146.444	148.9225	145.360001	146.704438	24164000	146.704438		
GOOG	2022-12-14	145.710004	145.542001	142.245493	144.510005	25178000	144.510005		
GOOG	2022-12-15	144.385987	147.572142	142.765505	147.3885	27280000	147.3885		
GOOG	2022-12-16	145.878996	145.951498	144.058499	144.838531	37400000	144.838531		
GOOG	2022-12-17	142.714493	142.283008	141.787994	142.802994	42840000	142.802994		
GOOG	2022-12-20	142.817698	142.810004	140.25	142.461338	20240000	142.461338		
GOOG	2022-12-21	143.186994	141.832147	141.738001	144.225538	18180000	144.225538		
GOOG	2022-12-22	144.188994	147.303994	143.087997	146.848938	16338000	146.848938		
GOOG	2022-12-23	147.089493	148.512601	146.558493	147.146232	13819000	147.146232		
GOOG	2022-12-27	147.863005	149.426496	147.25	148.002995	13260000	148.002995		
GOOG	2022-12-28	148.374495	145.371498	145.555501	146.447393	13824000	146.447393		
GOOG	2022-12-29	146.420504	147.183746	145.864701	146.004731	17222000	146.004731		
GOOG	2022-12-30	145.448997	147.382000	145.158499	146.002332	12578000	146.002332		
GOOG	2022-12-31	145.518499	149.385005	144.877505	146.679634	17260000	146.679634		
GOOG	2023-01-03	144.076494	145.880005	143.802502	145.076493	25714000	145.076493		
GOOG	2023-01-04	145.850507	146.810001	143.816147	144.416034	27938000	144.416034		
GOOG	2023-01-05	144.181	144.288004	137.523499	137.603333	48840000	137.603333		
GOOG	2023-01-06	137.867496	138.888005	138.763504	137.500995	28150000	137.500995		
GOOG	2023-01-07	137.364099	138.254745	138.786001	137.604331	18105000	137.604331		
GOOG	2023-01-10	138.038999	138.138999	133.146003	136.028435	36380000	136.028435		
GOOG	2023-01-11	138.180495	140.328496	138.815507	140.610332	23820000	140.610332		
GOOG	2023-01-12	141.554504	142.811535	141.12	141.647395	23642000	141.647395		
GOOG	2023-01-13	141.8495	143.185501	138.814001	138.150987	28660000	138.150987		
GOOG	2023-01-14	137.5	141.2005	137.5	138.786439	27850000	138.786439		
GOOG	2023-01-16	138.880005	137.281496	135.817004	136.280487	21782000	136.280487		
GOOG	2023-01-19	138.826507	138.338008	135.5	136.601993	20760000	136.601993		
GOOG	2023-01-20	138.814098	137.812003	133.144001	133.8068	31830000	133.8068		
GOOG	2023-01-21	133.091993	134.788496	132.001007	136.001995	41920000	136.001995		
GOOG	2023-01-24	132.027496	129.778005	128.641993	136.317994	35140000	136.317994		
GOOG	2023-01-25	128.055504	129.338001	138.377998	13810000	128.725637	128.725637		
GOOG	2023-01-26	133.802499	137.827495	137.103503	138.248005	38320000	138.248005		
GOOG	2023-01-27	137.363007	140.028899	128.546007	138.121032	30104000	138.121032		

VVNT	2022-08-11	5.91	6.28	6.01	6.97	486000	6.97		
VVNT	2022-08-12	6.07	6.48	6.95	6.4	533000	6.4		
VVNT	2022-08-15	6.37	6.57	6.29	6.54	482000	6.54		
VVNT	2022-08-16	6.54	6.98	6.45	6.8	438100	6.8		
VVNT	2022-08-17	6.94	6.17	6.39	6.44	404200	6.44		
VVNT	2022-08-18	6.35	6.17	6.75	6.36	383500	6.36		
VVNT	2022-08-19	6.3	6.26	6.08	6.1	288000	6.1		
Showing 1 to 6 of 772 entries						Previous			
						Download			

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)

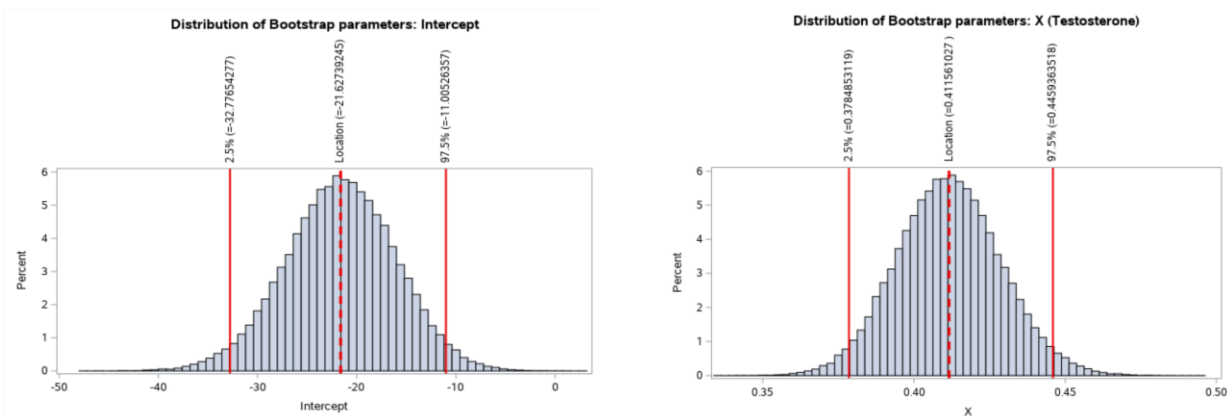
See the SAS code for the bootstrap task below. In the two histograms in this section we used 100,000 bootstrap samples to approximate our parameters (intercept and X, the explanatory variable). The code (regressionBoot macro) used for this task was more efficient with a run time of approximately 0:00:08.18 for 100,000 samples. Compare this to the 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.

In the case for 1,000 samples the older macro took around 31.81 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 BootRes to run very slowly. (Note that it would take a very long time running the older macro with anything more than 5,000 samples)

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:

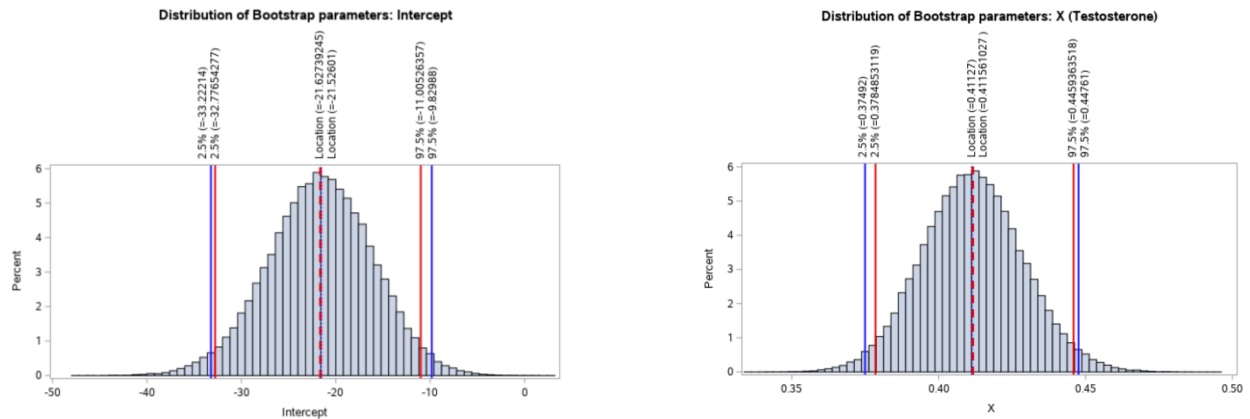


Let us now compare our bootstrapped results with the in-built 95% CIs. We overlay the CIs and the approximate parameter values in blue.

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 folder where 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=P
ctls;
    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 (us
ed 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=dashed) 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
    int sampsiz=&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 i
terations*/

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  
the code is).  
  
And add csv file in the folder where 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;


```

/* 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
-w-ise);
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 (ro
w-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., Jackk
nife_0);
Diff = Means-Means1;  *store the differences in new column, Diff;
Square = Diff**2;      *square the differences and store in new column, Squar
e;
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, s
tore 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 (no
te 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 obser
vation (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., Müller, U., Nielsen, M., Paoella, 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].