**Modern Statistics and Econometrics from a Computational Point of View, G63.2707: Assignment 4**

This assignment covers topics in robust statistics and data snooping, and will attempt to review and reinforce topics from each area. Note, there are a number of coding exercises within this assignment including suggestions on how to implement the codes. These are only suggestions. It is not necessary to build the codes in exactly the format I have suggested. If you have a different or easier way feel free to use it. Make sure, however, that the core calculations are done correctly.

The bootstrapping paper is not on the web, but a related paper may be downloaded from

http://wber.ucsd.edu/mbacci/white/pubs.html

**Problem 1:** Let $G$ be an arbitrary distribution and $H$ is an arbitrary distribution, and recall that we say $F$ should be regarded as having a distance $\leq \epsilon$ from $G$ if

$$F = (1 - \epsilon)G + \epsilon H$$

This is called the *gross error model*. Let $G = N(0, 1)$ and show:

1. The sample mean of $F$ has bias $\epsilon \cdot E[H]$.
2. What do we mean when we say that the "bias of $F$ is unbounded".
3. Write $\Phi(x)$ for the cumulative distribution function of $N(0, 1)$, and write $b$ for the bias of the sample median of $F$ relative to $G$.

   (a) Bias means measurement error. We are sampling from $F$ but trying to make estimates of the median of $G$ using these samples. Write down a mathematical expression for the bias $b$ of the sample median.

   (b) Show that at worst, the bias $b$ solves

   $$(1 - \epsilon)\Phi(b) = .5$$

   (c) Use the previous to show that when $\epsilon$ is small,

   $$b \approx 1.25 \frac{\epsilon}{1 - \epsilon}$$

   (Hint: expand $\Phi$ about zero). In terms of the basic notions of robust statistics, what does it mean for $\epsilon$ to be small?
4. Suppose $H$ above is symmetric about zero. Shose that the standard deviation of the sample mean is

$$\frac{1}{\sqrt{n}} \sqrt{(1 - \epsilon) + \epsilon V(H)},$$

where $V(H)$ is the variance of $H$.

**Problem 2** Recall that an $M$-estimator is an estimator from a sample of size $n$, $\hat{\theta}_n$, of a parameter $\theta$ of a family of distributions $f(x, \theta)$ satisfying

$$0 = \sum_{i=1}^{n} \phi(x_i, \hat{\theta}_n)$$

and further recall the influence function of the $M$-estimator $\phi$

$$\Omega(x) = \Lambda^{-1} \phi(x, \theta)$$

where

$$\Lambda = -\frac{\partial}{\partial \theta} E \phi(x, \theta)$$

Note that $\Omega(x)$ takes is actually a function of $\theta$, and should be read as the influence at a value of $\Omega$.

1. Explain based on the heuristic derivation of the influence function, why it is called "influence function".
2. Prove that the sample mean is an $M$-estimator. What is its influence function?
3. Prove that the sample median is an $M$-estimator.
4. Let $f \sim N(0, 1)$. Show that the influence function $\Omega(x)$ (at zero) of the median, when sampling from $f$ is given approximately by

$$\Omega(x) \approx 1.25 \text{sign}(x - \theta)$$

**Problem 3, Huber Estimators** Recall that a Huber estimator is an $M$-estimator $\hat{\theta}$ solving the equation

$$\sum_{i=1}^{n} \psi_{a}(x_i - \hat{\theta})$$

$$\psi_{a} = \max(-a, \min(x, a)), \quad a > 0$$

Answer the following:
1. Show that as $a \to \infty$ the Huber estimator converges to the mean.

2. Show that as $a \to 0$ the Huber estimator converges to the median.

3. Give a quick proof that the Huber estimator has influence function for a pdf $F$

$$
\Omega(x) = \frac{\psi_a(x)}{F(a) - F(-a)}
$$

**Problem 4, Stationary Timeseries** Recall that a timeseries $X_t$ is weakly stationary if the expected value of $X_t$ is constant for all $t$ and if the covariance

$$
Cov(X_t, X_{t+h}) = \gamma(h)
$$
is independent of $t$, i.e., only depends on the distance between points on the timeseries.

1. Let $e_t$ be iid normal with distribution $N(0, \sigma^2)$. Define

$$
X_t = \sum_{j=-M}^{M} \alpha_j e_{t-j}
$$

with $\alpha_j \neq 0$. Call this a **finite moving average process of order $M$**.

   - Compute $E[X_t]$
   - Compute $\gamma(h) = E[X_t \cdot X_{t+h}]$. (Hint: it is zero for $|h| > 0$).
   - Is the moving average process stationary?

2. Write a code $\text{ts} = \text{ma(len,M,alpha,sigma)}$ to generate a sample from the process

$$
X_t = \sum_{j=-M}^{M} \alpha_j e_{t-j}
$$

where

   - $\text{len}$ is the length of the timeseries $\text{ts}$ output by the code.
   - $M$ is the order of the process.
   - $\text{alpha}$ is a $2M + 1$ length vector of where $\alpha(i) = \alpha_{i-M}$
   - $\text{sigma}$ is the standard deviation of the $e_i$'s.
Problem 5, Stationary Bootstrap The following exercise will walk you through implementing the stationary bootstrap method of Romano and Politis from the paper *The Stationary Bootstrap*, Politis and Romano, Journal of the American Statistical Association, December 1994. The method was also given in class.

1. In this step you will implement the Politis and Romano bootstrap method in MATLAB. Write the following codes:

   - Write a code
     \[
     \text{reps} = \text{rreplicates}(\text{ts}, p, \text{num}, \text{autoseed})
     \]
     inputs a timeseries \(\text{ts} (1 \times N)\) vector, smoothing parameter \(p\) and the number \(\text{num}\) of bootstrap replicates to produce. The output should be a \(\text{num} \times N\) matrix of replicates where the \(i\)-th row is the \(i\)-th replicate from the Politis and Romano algorithm give in class and in the paper.
     About \text{autoseed}. For the datamining code that follows, it will be important to use the same random number seeds for generating each replicate. To ensure this, you can proceed by having a variable \text{autoseed} in your code. Then you can say that on the \(i\)-th replicate, you should use the \(i\)-th state of the random number generator, which, in MATLAB version 5.0 and higher, can be achieved by issuing the command
     \[
     \text{rand('seed',i)}
     \]

   - Write a code
     \[
     [\text{variance}, \text{means}] = \text{rbootstrapmean}(\text{ts}, p, \text{num})
     \]
     same inputs as above. This code computes the bootstrap variance of the sample mean of the timeseries \(\text{ts}\). Proceed as follows:
     - Compute replicates using \text{rreplicates}.
     - Compute mean of each replicated timeseries.
     - Find and return variance of the means (\text{variance}), and the vector of all means (\text{means}).

2. Use the code \text{ma} to generate two different time series:
• $ts_1$ a moving average timeseries with alpha = [1] (this is pure white noise) and standard deviation set to 1.
• $ts_2$ a moving average timeseries with order $M = 2$ and alpha equal to [.5.2.1].

Answer the following questions

• Does the stationary bootstrap accurately compute the variance of the sample mean for $ts_1$?
• Use the stationary bootstrap to compute the variance of the sample mean for $ts_2$. Is it greater than or less than that of $ts_1$? Do you think this is always true? Form a conjecture as to why.

**Problem 6. Data Snooping.** In this section we will practice implementing the Hal White data snooping algorithm. The set up is as follows. We imagine there is a dependent variable $y$ that we would like to study, and that we believe $y$ can be written as a linear model

$$y = X\beta + \epsilon$$

where $X$ is a vector of independent variables. That is, at any time $t$ we believe we might be able to forecast $y_t$ (value of $y$ at time $t$) as

$$y_t = X_t\hat{\beta}$$

We imagine that we get the $X_t$ values before we see what the value of $y$ will be, this simplifies notation, though in reality indexing like $y_{t+1} = X_t\hat{\beta}$ would be more appropriate.

The trouble is, we do not know exactly which variables to use to forecast $y$ so we are going to try a number of different variables in hopes of finding one that works very well. Fortunately, we are already using a two-variable model, so this model will serve as a benchmark. Only if one of the new models outperforms the benchmark will we use it. So we set out as follows. We run as many regressions as possible and see if its mean squared prediction error is less than that of the benchmarks over a *backtesting period*, and then we choose which model is the best one.

1. In your own words, and in one paragraph, explain what is wrong with the above stated approach.
2. Write a code

```python
errs = regresscheck(y,X,submodel,bmmodel,minfit):
```

This code is designed to compare the performance of one submodel of $X$ (submodel) with a different submodel of $X$ (bmmodel). The details follow.

The code accepts inputs of $y$, a dependent timeseries, $X$, a timeseries of independent variables. For example, if $y$ has 200 time periods of data in it, then $y$ will be a $200 \times 1$ matrix; if for the same input $X$ represents 7 independent variables, $X$ will be a $200 \times 7$ matrix. submodel is a list of integers, each less than the number of columns of $X$ (i.e., less than the number of independent variables) which describes which variables are included in the submodel.

We will refer to $X$ as a model for $y$, propose the linear model

$$y = X\beta + \epsilon$$

Now, write $X_i$ for the $i$-th column of $X$ and refer to this as the $i$-th independent variable. Now, a submodel of $X$ is just a subset of the variables in $X$, specified by a list of integers one through the number of columns of $X$. For example, if $X$ has 7 variables (columns) then a submodel could be [135] meaning the model with variables 1, 3 and 5. A submodel of size $n$ is a submodel with $n$ variables. The variable submodel represents which submodel we are going to check, e.g. if $X = [123]$ then the submodel contains only variables 1, 2 and 3.

The input variable (to regresscheck) bmmodel is a row vector representing the submodel of $X$ which will be used as the benchmark model. For example, suppose your trading group always uses two variables to forecast returns, i.e. $X_1, X_2$ and we want to try three new variables, $X_3, X_4, X_5$. Then we would want to look at various submodels of

$$X = [X_1, X_2, \ldots, X_5]$$

to see if any perform better. The code will do the following

- Write $n$ for the number of rows of $y$. Loop $t$ from minfit to $n$. 

• For

\[ y_t = X_{bm}^t \beta_{bm}^t + \epsilon_{bm} \]
\[ y_t = X_{sub}^t \beta_{sub}^t + \epsilon_{sub} \]

where \( y_t \) and \( X_{bm}^t \) etc. mean that \( y \) (resp. \( X_{bm}, X_{sub} \)) has been truncated to have only rows 1 through \( t \) so that the \( \beta^t \)'s are estimated only from data available from time 1 to \( t \).

• For each \( t \) compute the forecast of time \( t + 1 \) value of \( y \) based on the new data for \( X \) at time \( t + 1 \) (this is slightly different than what we did in class, but it amounts to the same thing conceptually).

\[ \hat{y}_{bm,t+1} = X_{bm t+1} \hat{\beta}_{bm}^t \]
\[ \hat{y}_{sub,t+1} = X_{sub t+1} \hat{\beta}_{sub}^t \]

the forecast values of \( y \) at time \( t + 1 \). The way to look at this is that at time \( t \) a regression is run using all data available up to time \( t \). Then data arrives at time \( t + 1 \) in \( X \). The regression run at time \( t \) is used to forecast the variable \( y \)'s value at time \( t + 1 \) using the \( X \) data at time \( t + 1 \). You can imagine that there is a delay between the time you get the \( X \) data at time \( t + 1 \) and the time the value of \( y \) is realized.

• For each \( t + 1 \) compute the outperformance

\[ f_t(sub,bm) = (y_{t+1} - \hat{y}_{sub,t+1})^2 - (y_{t+1} - \hat{y}_{bm,t+1})^2 \]

• Return the variable

\[ \text{errs} \]

which is the time series of \( f_t \)'s

3. Write a code

\[ \text{cutoff} = \text{maxpctl}(ts,pctl) \]

where \( ts \) is an \( n \times k \) matrix, whose \( k \)-th column is a timeseries (which we will refer to as the \( k \)-th timeseries). This code should find the \( pctl \)-th percentile of the timeseries formed by taking the maximum of each row of \( ts \). To test your code, do the following:

• Generate a single timeseries with 10000 elements with independent draws from \( N(0,1) \), and find the 95-th percentile (\( \approx 1.6493 \)
• Generate two timeseries with 10000 elements with independent draws from \( N(0,1) \), and find the 95-th percentile. (Answer \( \approx 1.9681 \))

• Do the same for 10 timeseries. (Answer \( \approx 2.5842 \))

4. Explain in your own words why the 95-th percentile increases with the number of timeseries.

5. Write a code

\[
[cutoff, models] = datamine(y, X, subsize, bmodel, minfit, pctl)
\]

such that

• The code loops through all submodels \( \text{sub} \) of \( X \) of a given size \( \text{subsize} \). This will assign a number to each model.

• At each iteration, the code calls \text{regresscheck} and obtains the timeseries of \( f_i \)'s for that model and then
  - creates the statistic
    \[
    V_{\text{sub}} = P^{1/2} \left( \tilde{f}(\text{sub}, \text{bm}) \right)
    \]
    which is the \textit{normalized performance} of the submodel \( \text{sub} \).
  - creates stationary bootstrap replicates of the \( f_i \)'s to a vector of values
    \[
    \bar{V}_{i, \text{sub}}(\text{sub}, \text{bm}) = P^{1/2} \left( \tilde{f}_{i*}(\text{sub}, \text{bm}) - \tilde{f}(\text{sub}, \text{bm}) \right)
    \]
    one for each replicate \( i = 1, \ldots, 1000 \), where \( \tilde{f}_{i*}(\text{sub}, \text{bm}) \) is the \( i \)-th replicated value for \( \tilde{f} \) formed as the mean of the replicated \( f_i \)'s. Make sure that \text{autoseed} is turned on at this point.
  - Create the vector
    \[
    v_{\text{model}}(i, \text{model num})
    \]
    where \( i \) is the \( i \)-th bootstrap replicate, and \( \text{model num} \) is the number of the model.

• Use the code \text{maxpctl} to find the stationary bootstrap approximation to the \( \text{pctl} \)-th percentile of the distribution

\[
\max_{\text{sub models}} \bar{V}(\text{sub}, \text{bm})
\]
where the maximum is taken for each bootstrap replicate across all submodels, as in

\[ \hat{V}_{i*\text{,max}} = \max_{\text{sub models}} \hat{V}_{i*\text{,sub}} \]

Note: it is possible to make this computation because the \( i \)-th bootstrap replicate has meaning because the \( i \)-th replicate is computed using the same random seed.

- Return the percentile value of this distribution.
- Provide a list of models that exceed the cut-off (the list of models could be a \( k \times \text{subsize} \) matrix, where \( k \) is the number of models exceeding the cutoff and \( \text{subsize} \) is the number of variables in the model.

**Problem 7: Bootstrap Challenge** On my web site there are two downloadable ascii files called

- dependent
- independent

The dependent file is a vector \((150 \times 1)\) and the indendent file is a vector \((150 \times 7)\). We will call the data \( y \) and \( X \) respectively. The variable \( y \) was created from a subset of 2 of the 7 variables as a linear model. The problem is, you do not know which 2. Here is your assignment:

1. Data mine the \( X \) variable and find the best fitting 2 variable model to the \( y \) variable. Use at least 100 data points in each backtesting regression. What is it?

2. Suppose a trading group has been using model \([23]\) historically and the group datamines over all possible models of size 2 and decides to switch to \([26]\). Should the group switch? Why or why not?

3. Suppose the has been historically using \([56]\) and the group datamines over all possible models of size 2 and decides to switch to \([26]\). Should the group switch? Why or why not?