Attachment "A"

Project Title: One Parameter

Table of Contents

A. INTRODUCTION/BACKGROUND:

The South Florida Water Management District (SFWMD) is a regional governmental agency that oversees the water resources in the southern half of the state, covering 16 counties from Orlando to the Florida Keys. The agency is responsible for managing and protecting water resources of South Florida by balancing and improving water quality, flood control, natural systems and water supply.

Data gathered and managed by the SCADA system must be published to several downstream systems. One of the downstream systems is DBHydro. Once data is loaded into staging tables, Quality Assurance/Quality Control needs to be performed on the data by applying rules that can detect anomalies in the data and attempt to correct the data if possible. The solution will leverage Java, IBM's ODM (Operational Decision Management), Mule and statistical computing using the "R" statistical programming language.

B. OBJECTIVES:

The project started in April 2018 and was interrupted. The sections below provide the completed and pending tasks. The District is seeking a consultant to complete the remaining tasks as itemized in Section D. The consultant has the option to work on-site or remotely. This is a deliverables-based Statement of Work and the cost is a "Not-to-Exceed" amount determined by the total for all deliverables.

C. SCOPE OF WORK:

C.1 SKILLS/ COMPETENCIES Requirements:

1. Three years of experience in performing data analysis using data mining techniques and various approaches such as Data processing for statistical analysis, Regression analysis, optimization techniques, Time series analysis, and Linear & Logistic Regression analysis.
2. Five years of experience with Statistical Capabilities in a business context using R statistical language. Previous experience with the setup of R infrastructure and server.
3. Ability to provide meaningful insights to data for extraction and cleansing.
   a) Work into different and large data sets; structured and unstructured data
   b) Reviewing and controlling quality of client's data.
4. Experience working in fast-paced Agile environment.
5. Excellent written and oral communication skills.
6. Knowledge of any hydrologic real-time data model e.g. DBHYDRO
7. Bachelor's degree in Computer Science, Computer Engineering, or related field.

C.2 Completed Tasks:

Tasks completed:

1) Proof of concept and customer validation of the concept.

2) Design document

3) Core logic of historical statistics has been completed. The "R" logic/script has to be productionized, deployed and setup on the "R" server.

4) Modelling scripts to correlate timeseries within water body for strong learners is complete.

D. SUMMARY SCHEDULE OF TASKS AND DELIVERABLES:

This is a deliverables-based Statement of Work and the cost is a "Not-to-Exceed" amount determined by the total for all deliverables. The Consultant shall submit a weekly status report to the project manager by 9:00 AM on the first working day following the reporting period. The report must include the percentage of completion for each deliverable and the plans for the next reporting period. The vendor may invoice at the completion of each deliverable. Payment terms are net 30 days. Payment is subject to District approval that the deliverable is complete.

Task#	Task Description	Deliverables
1	Make following changes to strong learner script which is attached in Exhibits 2, 3, 4 for reference. 1. Write model output results into a relational database table. 2. Make model parameters configurable. These model parameters are stored in relational database tables. 3. Refactor existing script to make it more flexible, supportable and maintainable such as enhance logging, exception handling, notification etc. 4. Make model execution dynamic and support for batch runs.	1. Finished code which is reviewed and approved by District staff. 2. Finished code deployed and running on dev, test and prod server. 3. Test plan and test results which is reviewed and approved by District staff
2	Design and implement weak learner script 1. Develop design strategy 2. Implement weak learner script that will write model output to database tables and store configuration parameters in database. Make model execution dynamic 3. Make model execution dynamic and support for batch runs.	1. Design document for weak learner script which is reviewed and approved by District staff. 2. Finished code which is reviewed and approved by District staff. 3. Finished code deployed and running on dev, test and prod server. 4. Test plan and test results which is reviewed and approved by District staff
3	Design and implement model validation/sampling approach such as using PSI methods or other proven techniques. At predefined times of year models need to be sampled to verify their continuing validity.	1. Design document for model validation/sample strategy which is reviewed and approved by District staff.

	Population Stability Index needs to be leveraged to identify shift in models. If the shift is more than configured limits, models need to be regenerated.	2. Finished code which is reviewed and approved by District staff. 3. Finished code deployed and running on dev, test and prod server. 4. Test plan and test results which is reviewed and approved by District staff
4	Set up the "R" infrastructure for the development, test and production environments. The "R" servers are Linux OS based. The order of setting up "R" infrastructure is DEV first followed by Test and then Production.	1. DEV, TEST and Production server infrastructure reviewed, validated and approved by District staff.
5	Make following changes to historical statistic script which is attached in Exhibits 5, 6, 7 for reference. 1. Write output results into a relational database table. 2. Refactor existing script to make it more flexible, supportable and maintainable such as enhance logging, exception handling, notification etc. 3. Make script execution dynamic and support for batch runs.	1. Finished code which is reviewed and approved by District staff. 2. Finished code deployed and running on dev, test and prod server. 3. Test plan and test results which is reviewed and approved by District staff

E. EXHIBITS

1. Design document
2. Data_prep_model.R.txt
3. Substitution_Ranking.R.txt
4. Estimation_Ranking.R.txt
5. Hist_summary_stat.R.txt
6. Hist_summary_stat_write.R.txt
7. Summary_stat.R.txt

EXHIBIT 1 – DESIGN DOCUMENT

ONE PARAMETER

Ranking

08-27-2018

This document describes the Design for one parameter Estimation and Substitution Ranking

Bandaru, Satish

Document History

Author

Name	Title	Contact Info
Satish Bandaru	Consultant	

Document Version Control

#	Date	Changed By	Section Changed	Description of Change
1				Initial document creation

Stakeholder Approval to Begin Development:

Indicates requirements have been approved in order to begin development. With this sign-off, requirements are base-lined and any changes to the requirements will require a change request.

Name	Title	Link to Email Sign-off	Date of Approval
Lokendra Matoli	Chief Enterprise Architect		

Stakeholder Final Approval & Acceptance of Solution:

Name	Title	Link to Email Sign-off	Date of Approval
Lokendra Matoli	Chief Enterprise Architect		
John Raymond	Section Leader		
Ketankumar Shah	Principal Enterprise Developer		
Brian Smith	Principal Enterprise Developer		

Related Documents (For Reference Documents, please see Appendix B)

Name	Description	Location
One parameter Design	The Business Requirements Document details the business and functional requirements for the project.	

Abbreviations, Acronyms, and Terminology

Acronym / Abbreviation / Term	Expansion	Definition
PSI	Population stability index	Industry approved metric to measure the stability

Acronym / Abbreviation / Term	Expansion	Definition
Model/Algorithm		A Statistical /mathematical (Algebra) equation would be fit to the given data
Dependent Variable		A dependent variable is something whose value is estimated from other variable
Independent Variable		An independent variable is something whose value is used to estimate other variable

Section 1 Version Control

#	Date	Changed By	Description of Change
1	08-26-2018	Satish Bandaru	Initial Document Creation
2	08-27-2018	Satish Bandaru	Review comments incorporated

Table of Contents

1 INTRODUCTION

1.1. DOCUMENT PURPOSE

The Design document documents and tracks the necessary information required to effectively define architecture and system design in order to give the development, business team guidance on architecture of the system to be developed. The Specifications and Design document is created during the Planning Phase of the project. Its intended audience is the business, project manager, project team, and development team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

1.2. AUDIENCE

- Project Sponsors
- DBHYDRO One Parameter: Project Manager, Director, Development Team

1.3. SYSTEM OVERVIEW

The solution will be a data table(s) with Estimation Ranking, model coefficients, and Substitution ranking capability by configurable thresholds powered by statistical algorithms. It will include scientific periodic model monitoring structure, capability of storing the Ranking & model coefficients for historical data periods. The platform will be highly scalable in all dimension with minimal (or no) human intervention and has a goal of less than 10% error rates.

1.4. SCOPE

1.4.1. FRONT-END SCOPE

ID	Area	Capability
FE.1	Portal UI	Web page to set the thresholds by User

1.4.2. BACK-END SCOPE

ID	Process	Description
BE.1	Data Preparation	The preparation of the data from SITE & GIS source.
BE.2	Model Development & Testing	The development and testing of models and store the model outcome
BE.3	Ranking algorithm & Results capture	The execution and storing of ranking results.
B4.4	Historical Ranking	Rank(measure) the TIMESERIES for historical data

1.5. ASSUMPTIONS AND CONSTRAINTS

1.5.1. ASSUMPTIONS

Initial assumptions made before starting the system design process. Assumptions are future situations beyond the control of the project, whose outcomes influence the success of the project.

Date Identified	Assumption	Comment	Validated Y/N	Date Valid	Validated By:
	GIS will provide the Stage and other parametric TIME SERIES data				
	GIS Data environment and data sets are “ready for use” from Day-0				
	May need intermediate table creation to accommodate GIS				
	The infrastructure can be scaled up to handle the required volumes	Reference Software Requirements			
	Security policies do not prohibit 3rd party libraries chosen for this implementation Eg: • Magrittr • Dplyr • Readr				
	Ongoing Sensor catalogue and their current status to be maintained and notified				
	All the Parameters should be suitable for statistical Parametric(supervised) modeling	Any set of Parameters are not met this criteria, considering the priority and no.of such list new model will be built.			

1.5.2. CONSTRAINTS

Constraints are boundary conditions on how the system must be designed and constructed.

Date Identified	Constraint	Comment	Validated Y/N	Date Valid	Validated By:
	Scheduled processes will be run yearly once or earlier if changes in sensors are observed				
	The current R software and packages had the limited capability to perform write operations w.r.to PK and FK constraints in Oracle DB	This can be mitigated by creating a Trigger/stored procedure once the results are written in temporary table			
	Usage of Existing IVG (interval value generator) currently not verified for interpolation for model generation.	R’s Interpolation statistical methods are used			
	Sensors latest status to be notified or Maintained in a catalogue to see the recent changes	Manually run R ranking job will for missing sensors in waterbody			
	Performance monitoring script runs periodically (6months)				

1.5.3. DEPENDENCIES

Item	Notes	
DCVP_ARCHIVE		
GIS Data		

1.5.4. R PACKAGES

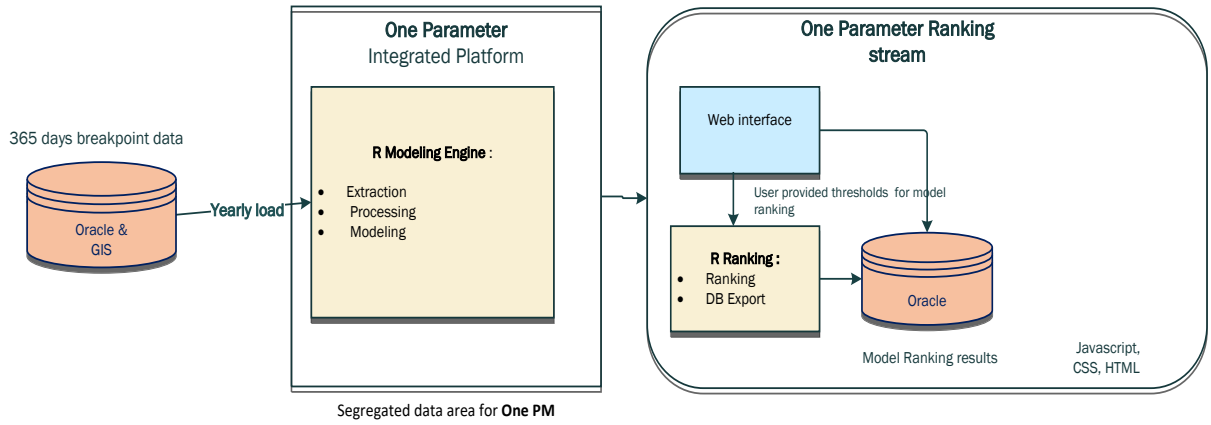
Below are some of the list of R packages

Package Name	Description
DPLYR	Essential shortcuts for sub setting, summarizing, rearranging, and joining together data sets. dplyr is our go to package for fast data manipulation.
R JDBC	Data import and export from DB
ZOO	Provides the most popular format for saving time series objects in R.
TIDYR	Tools for changing the layout of your data sets. Use the gather and spread functions to convert your data into the tidy format, the layout R likes best.
Data.Table	Tabular way of handling data
Ggplot2	Plotting
Impute TS	Interpolation & Missing data treatment
Metrics	Model validation
SQLDF	Data import and export from DB
Reshape	Data shaping
Corrplot	Correlations
Lubridate	Tools that make working with dates and times easier.
Quantmod	Tools for downloading financial data, plotting common charts, and doing technical analysis.
Parallel	Use parallel processing in R to speed up your code or to crunch large data sets.

2. ARCHITECTURE DESIGN

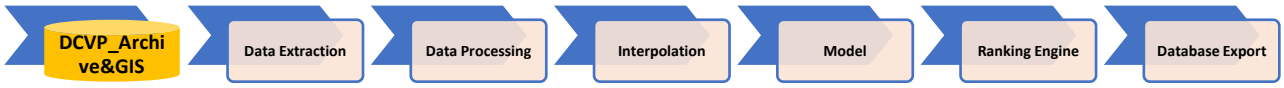
2.1. LOGICAL VIEW

2.1.1. SYSTEM DIAGRAM



3. SYSTEM DESIGN

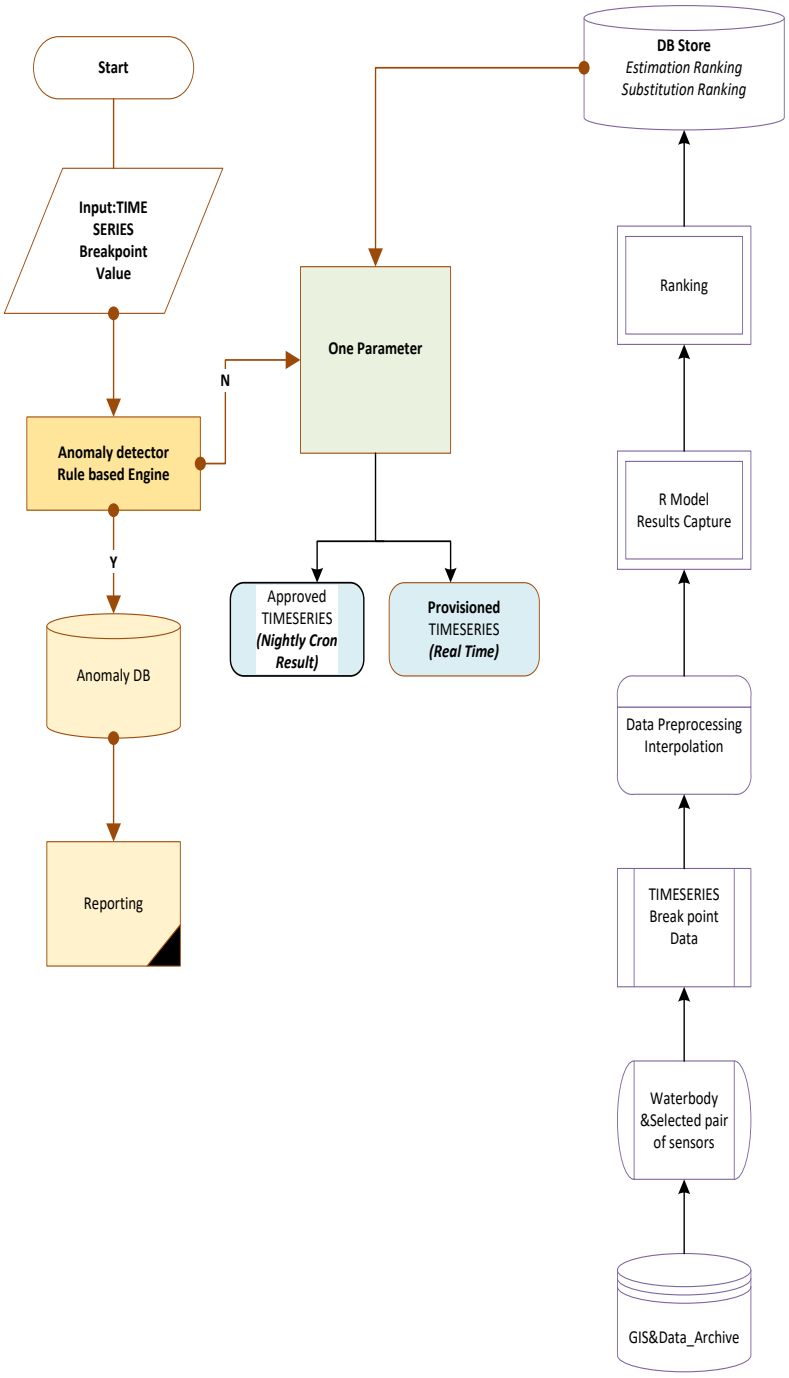
3.1. DATA FLOW DIAGRAM



3.2. DATA FLOW DETAILS

ID	PROCESS	STEPS
1	Data Extraction	• Fetch the TIMESERIES(sensors) in each waterbody ID from GIS • Capture the breakpoint data for different sensors in a given waterbody • Length of data starts from 1year (breakpoint interval data)
2	Data Transformation & Processing	• Time series data filtered by different business given conditions (eg: DCVP_TAG is NULL) • Data reshaping for candidate sensors (Eg: Dependent ~ Independent)
3	Interpolation	• Understand the missing data and replace it with 'Interpolation'. This makes the both dependent and independent TIMESERIES with equal time stamps
4	Ranking Engine-Part1 Modeling	Pre-requisite: Successful data transformation& processing, Interpolation. • Model the Sensors(TIMESERIES) with each other sensors belong to water control unit (one-one relation) • Capture the model outcome such as coefficients, error rates, R2 • Group the data by TIMESERIES
5	Ranking Engine-Part2 Estimation Ranking	For each TIMESERIES in waterbody: • Arrange data by High R^2 • Arrange the data by low error rate Provide the Ranking based on above priority
6	Ranking Engine-Part3 Substitution Ranking	The following multi step process • Based on highest R^2 flag the substitution indicator Eg:1-Eligible,0-Not eligible • Default R^2 value or user provided R^2 value used • On the eligible TIMESERIES arrange the data by lowest error rate, SENSOR QUALITY (TECHNOLOGY) and rank it.
7	Database Export	• Export the substitution ranks to database (newly created substitution _rank table) • Export the Estimation ranks to database (newly created Estimation _rank table)

3.3. OVERALL FLOW DIAGRAM



4. PARAMETER ONBOARDING:

First Phase: In the first phase all the TIMESERIES belongs to 'Stage' activities such as (Head water, tail water) are in scope

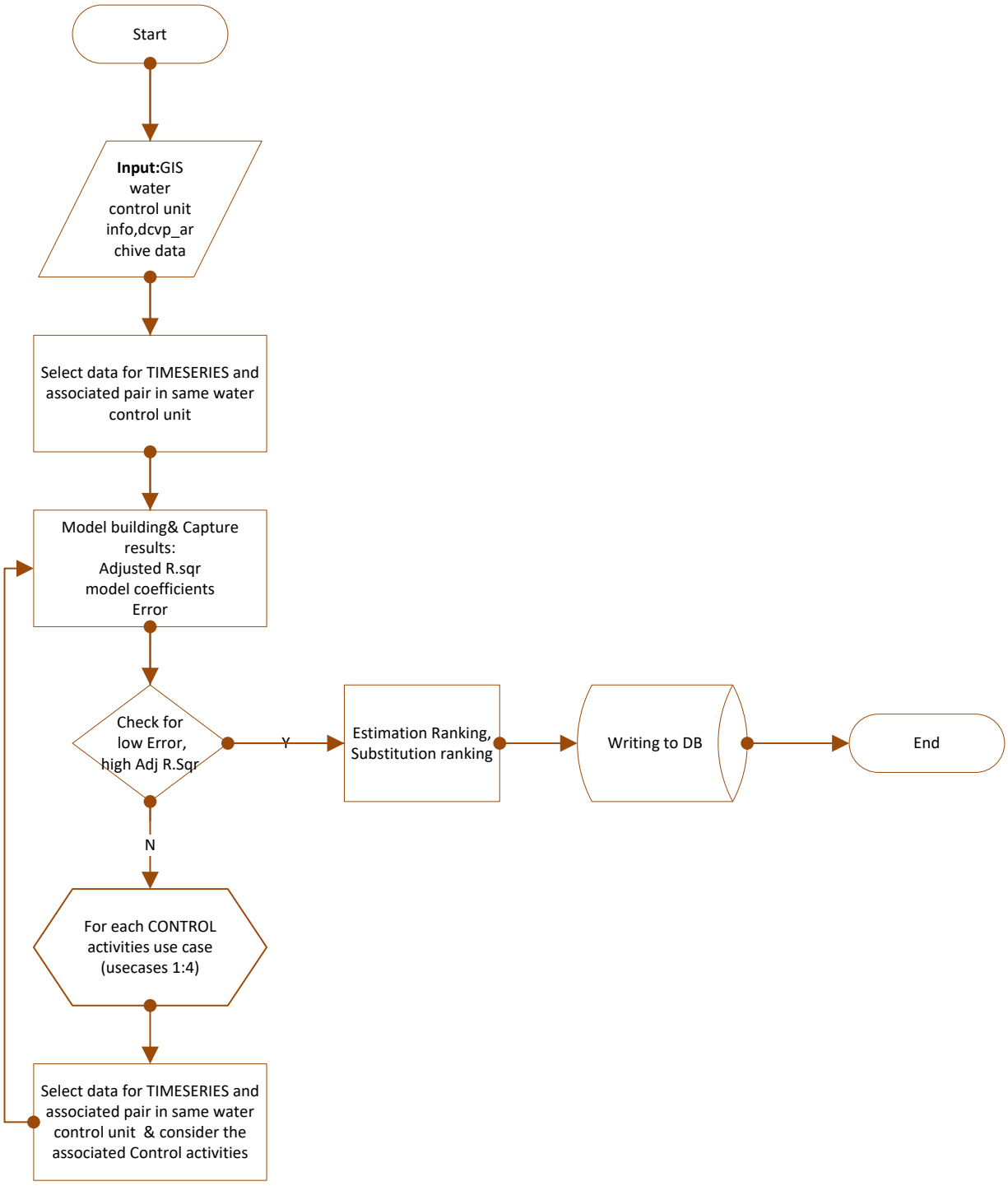
Approach:

1. GIS will provide the sensors in each waterbody and respective distance
2. Using GIS, DCVP_ARCHIVE fetch the legitimate data (as per business guidelines) as model input
3. Built the models for each pair of sensors and capture the model results
4. Estimation ranking
5. Substitution ranking by considering user given thresholds else default thresholds will be used.

4.1. PROCESS FLOW DIAGRAM

4.1.1. FIRST PHASE:

STAGE ACTIVITIES ALGORITHM FLOW



4.1.2. GENERAL MODEL APPROACH

- Select the pair of legitimate TIME SERIES breakpoint data within the same water control unit.
- Length of TIMESERIES is 1 year
- Perform the data shaping, interpolation, wrangling activities as part of pre processing
- Apply the parametric model such as Linear Regression
  Capture the model information such as Coefficients, R^2, Error etc.
- Derive the estimation ranking based on High R^2, low error
- Derive the substitution eligibility and rank based on High R^2, lower error thresholds either given by user or default value.
- Write the results to Data base.

4.1.3. MODEL IMPROVEMENT APPROACH:

This section elaborates the model improvement strategies.

Select the candidate list of sensors which is having Low Adjusted R^2 (which is equaling to high Error).

** Threshold values of R^2 and Error TBD based on overall analysis.

Consider the above list of sensors as Weak learners (Weak predictors) and check the below strategies to reduce the error

- Increase length of data (model input) and run the model and revalidate the results
- Apply statistical model tuning methods.
- Consider the other influential (business) factors of the Sensors such as
  Control Activities:
    - Analyze when the Control activities are not performed on both Dependent and Independent variables
    - Analyze when the Control activities are performed only on independent Variables
    - Analyze when the Control activities are performed on both dependent and independent variables
    - Analyze when the Control activities are performed only on dependent variable
- Validate the model performance by above strategies and group the sensors based on their improved model performance criteria
- Run that Batch separately
- If the model is not improved any treatment those sensors will go as per general approach

Future Phases:

In the next phases remaining metrics(parameters) are on boarded.

- The idea of onboarding different parameters within the same Parametric Statistical model.
- To achieve this, list of parameters should follow the underlines assumption of Parametric strategy.

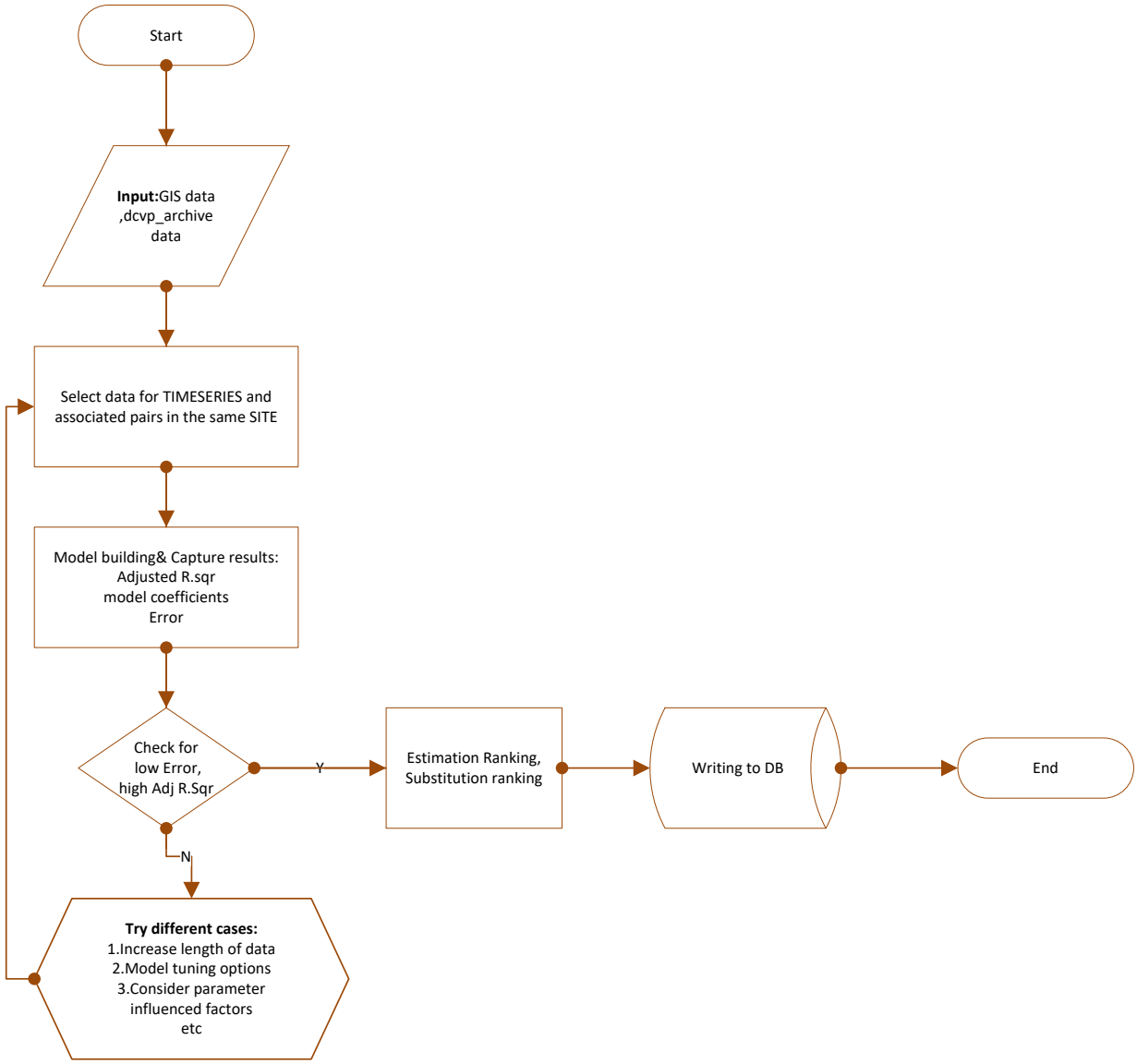
Once the parameters satisfy these criteria below are steps to model, rank the Parameters

1. GIS will provide the sensor list for each group based on business guidelines. (*For Eg: To gather the sensors related to rainfall, GIS will provide combination of sensors based on distance or similarity of site. i.e. RF ID(rainfall ID),TIMESERIES 1, TIMESERIES 2, TIMESERIES 3 etc*)
2. Using GIS, DCVP_ARCHIVE fetch the legitimate data (as per business guidelines) as model input
3. Built the models for each pair of sensors and capture the model results
4. Estimation ranking
5. Substitution ranking by considering user given thresholds else default thresholds will be used.

4.2. PROCESS FLOW DIAGRAM

4.2.1. FUTURE PHASES:

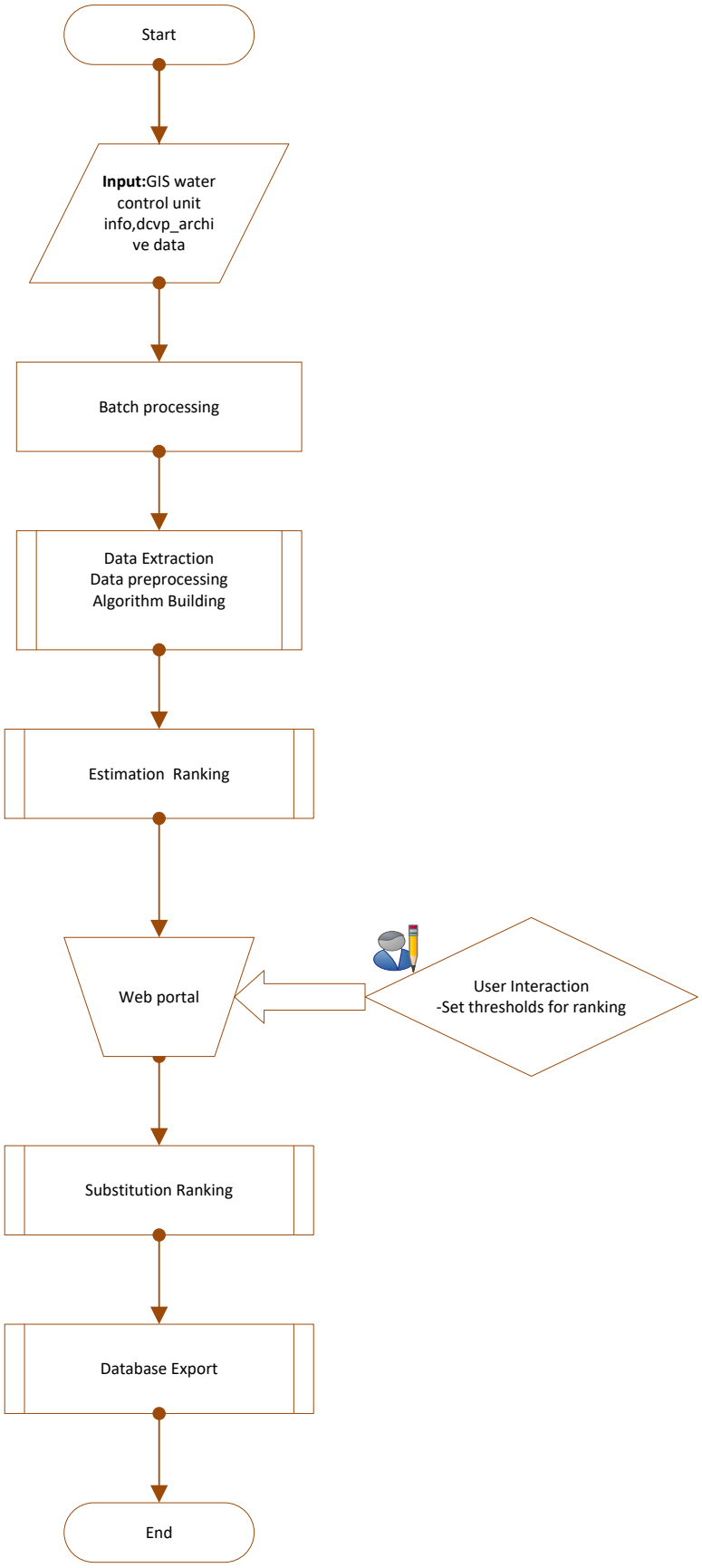
ALGORITHM FLOW



5. IMPLEMENTATION FLOW

5.1. ONGOING RANKING IMPLEMENTATION FLOW:

The below flow chart explains the ongoing process of Estimation and Substitution Rankings


Start
Input:GIS water control unit info,dcvp_archive data
Batch processing
Data Extraction
Data preprocessing
Algorithm Building
Estimation Ranking
Web portal
User Interaction
-Set thresholds for ranking
Substitution Ranking
Database Export
End

5.1.1. BATCH PROCESSING:

Ranking and storing model coefficient information are done phase

wise or per batch. Below list out the high level criteria of a batch

- Batch contain 20-25 water control units. Scalability determines the size of a batch
- Batch can be a user provided priority list
- Batch can be prepared by an area
- Batch can be a weak learner's use case group
- Batch can be a list of GIS candidates
- Batch can be size of water control unit (Large, medium, small) etc.

5.1.2. YEARLY IMPLEMENTATION

- Yearly once all the batches will be run.
- In each run compare the model details with previous model details and only changed values are updated with latest EFFECTIVE_DATES
- Latest filtered values are updated to temporary table in Oracle
- From Oracle Temporary tables data will be transferred to respective designated tables

5.1.3. R SCRIPTS

As part of the Ranking Process the below R scripts are run sequentially once the success of previous one.

- Data Extraction. R
- Data Preprocessing & Model. R
- Estimation Ranking. R
- Substitution Ranking. R
- Data export. R

5.1.4. EXCEPTION HANDLING

- Scripts are scheduled sequentially and success of script triggers next script
- Logs are maintained provide the details of error outcome
- Any script failure notifies the technology team (business etc)
- The roll back strategy is to delete failed entries and rerun the scripts from scratch after fixing.

5.2. HISTORICAL RANKING IMPLEMENTATION FLOW:

This exercise is to facilitate the idea of PUBLISH TIMESERIES for all time periods (once active). To accomplish this, store the ranking information year wise based on availability of sensors at that time frame

- GIS will provide or confirm the list of TIMESERIES with in a waterbody (WBID, TS1, TS2, TS3 so on...) for all the required time periods.
- Gather the previous year/years of data (based on the availability) for each historical year (test year is 2000, capture 1999's information)
- R Model engine collects and process the information and provide the model results
- R Ranking engine arrange and rank based on model information along with external factors (such as distance, user threshold inputs)
- R results are exported to DB
- From DB results are set into respective tables (such as Estimations are stored in Estimation_ranking, Substitutions are stored in substitution_rank tables etc)
- Results are updated with "EFFECTIVE_DATE" information to understand the validity of data

5.3. CAPABILITIES

ID	TITLE	DESCRIPTION
CP.ID1	Error Handling	Each script is equipped with exception handling mechanism by running independently and write the logs. Sequence will be halt and no further execution happened till each step is successful Notification Capability
CP.ID2	Adopt to different parameters (metrics)	Existing process is capable enough hold different parameters with minimal human intervention and provide the ranking, model results
CP.ID3	Notify	Notify the users (technology team) about the job status

5.4. USER CONFIGURATION

The below web page enables the users to control, configure the model thresholds and also model performance threshold values.

Note: *The below is for illustrative purpose Actual page might not be same.*

MODEL CONFIGURATION PAGE

Substitution Thresholds

R.Square	0.95
Error	0.1
EFFECTIVE DATE	05-SEP-2019

SAVE RESET

Model Performance Criteria

No Impact	0.0	5%
Low Impact	5	10%
Medium Impact	10	15%
High Impact	>15%	
EFFECTIVE DATE	05-SEP-2019	

SAVE RESET

View Model Details

Water Control Unit	
TIMESERIES	
R Square	0.95
Error	0.05
Slope Value	.92
Substitution Eligible	No
Top Ranked TIME SERIES	TIMESERIES NAME
EFFECTIVE DATE	05-SEP-2019

APPROVE NOTE

- Above inputs are saved into database.
- R “model” call look for Substitution thresholds and use this information to Rank.
- R “model monitoring” call look for model performance criteria and derive the stability of the model

6. DEPLOYMENT PROCEDURE

6.1. TEST ENVIRONMENT:

ID	PROCESS	PROCESS STEPS
1	Data Extraction	Data extraction done by taking production environment
2	R Model Engine	R model engine provides the model outcomes
3	R Ranking Engine	R Ranking results are write to test environment (test oracle DB)

6.2. PRODUCTION ENVIRONMENT:

ID	PROCESS	PROCESS STEPS
1	Data Extraction	Data extraction done by taking production environment
2	R Model Engine	R model engine provides the model outcomes
3	R Ranking Engine	R Ranking results are write to production environment (Oracle DB)

6.3. VALIDATION PROCESS:

ID	PROCESS	DESCRIPTION
1	Unit Test	Verify the script's outcome w.r.to functionality in development environment
2	Integration Test	Test the inter-module interactions from R to DB (Test DB)
3	User Acceptance Test	In this Users will be involved and check the model outcome and rankings
4	Post deployment monitoring	A model performance analysis script will be provided to run periodically to see the overall model performance Accordingly run the Ranking jobs to store the new rankings and model values

6.4. POST DEPLOYMENT (PRODUCTION) MONITORING:

A script will be provided to run the models on sample number of sensors and cross verify the error rate. The shift of error rate (average error rate) determines the model re run or not with new set of data elements.

1. Proposed to run the model for every year so that model will be stores with latest TIMESERIES information
2. Pick a random sample of pair of sensors and respective R^2
3. For the same list of sensors capture latest data (one year from test day)
4. Run the R Model, R Ranking Jobs
5. Compare Historical R^2 vs Current R^2 by using a statistical technique called PSI
6. The range of PSI outcome determines the model stability

7. The higher PSI (this value will be provided at final stages) tells the more fluctuation and requires an action
8. Re run the model.

   OR
9. Select the top rated (Rank1,2) pairs from random waterbodies (5% of waterbodies)
10. Calculate average R^2 in each water control unit
11. For the selected waterbodies re run the model with latest data (Previous one year from test day)
12. Calculate average R^2 in each water control unit
13. Compare the difference of R^2 and check the result against below table

Criteria	Impact
<5%	No
>5% & <10%	low
>10 % <15%	Medium
>15%	High

14. The more the impact the sooner the action.

Based on the merit of the models one of the above or similar strategy will be applied.

7. RECOMMENDATIONS

7.1. RECOMMENDED HARDWARE ARCHITECTURE

ID	NAME	CONFIGURATION
1	HARD DISK	>2 TB
2	CPU	>2.25 GHZ
3	MEMORY(INTERNAL)	>16GB
4	PROCESSORS	>8
5	Bits	64

7.2. RECOMMENDED SOFTWARE ARCHITECTURE

ID	NAME	VERSION
1	MS WINDOWS	Latest
2	R Studio	3.5.0 or latest
3	R Packages	Separate list will be given
4	ORACLE TOAD	Latest version
5	R Server	TBD

7.3. INITIAL MODEL AND DATA RECOMMENDATIONS

ID	AREA	DETAILS
1	Model criteria	Parametric learnings of each pair in a waterbody/site/area
2	Data	Any time consider immediate previous years data as model input Scalability vs Data inversely correlate
3	Estimation ranking criteria	High adjusted R^2 and low error rate gets high priority Next priority is lowest inter distance between sensors
4	Substitution eligibility	High model R^2 and low errored TIMESERIES eligible for Substitution. Initial R^2 recommendation is 0.97% (Business team can configure this number) Next priority is lowest inter distance between sensors
5	Interpolation	Linear aggregation techniques are scalable
6	Bucket the TIMESERIES	Group TIMESERIES with the below As High R^2 correlated with low error rate

		75-100% are grouped to Good learners 50-75% are grouped to average learners <50% are grouped to weak learners Based on this model refinement, implementation strategies may differ slightly
7	Ranking yearly	Yearly once run the Ranking scripts with latest data to get better model prospects
8	Model performance monitoring	Every 6 months run the model performance script
9	Sensor status Catalogue	Maintain ongoing sensor catalogue and re run the model on the impacted water control unit

7.4. SCHEDULING RECOMMENDATIONS

SCRIPT NAME	FREQUENCY	CONSTRAINTS/NOTES
Data Extraction	Yearly once	
Data processing & model	Yearly once	
Ranking	Yearly once	
Export	Yearly once	
Model Performance Monitoring script	Periodically (6 months)	

8. SYSTEM ACCEPTANCE CRITERIA

Specify the general system acceptance criteria specified and agreed upon by the project sponsor and key stakeholders that will be used to accept the final end product.

ID	Acceptance Criteria
AC.1	For each TIMESERIES the ability to rank the best possible neighborhood TIMESERIES in same waterbody(complex/area)
AC.2	The ability to provide the coefficient (slope/beta) value with lowest error rate
AC.3	The ability to rank and give coefficients for historical data (backlog. Eg: last 10 years)
AC.4	Ability to have monitoring criteria to check the model performance periodically

9. SUMMARY STATISTICS

The summary statistics are generated as part of Anomaly Detection rule based models. The below statistics are captured periodic (yearly twice) basis such as Wet and Dry periods

ID	STATISTICS (BY TIMESERIES)	PERIOD
1	MINIMUM Breakpoint value	Wet & Dry
2	MAXIMUM Breakpoint value	Wet & Dry
3	AVERAGE Breakpoint value	Wet & Dry
4	COUNT Breakpoint value	Wet & Dry
5	STANDARD DEVIATION Breakpoint value	Wet & Dry
6	RATE OF CHANGE Breakpoint value	Wet & Dry
7	MAXIMUM RATE OF CHANGE Breakpoint value	Wet & Dry
8	MAXIMUM RATE OF CHANGE Breakpoint value	Wet & Dry

Anomaly Detection algorithm (High level):

1. Anomaly detection algorithm takes the TIMESERIES breakpoint as input
2. Compare the breakpoint value with above summary statistics
3. If the value is not falling within acceptable criteria of summary stat raise the flag as anomaly

10. APPENDIX

10.1. RANKING:

	Dependent	Independent	Slope	Intercept	adj.r.squared	RMS Error	Water	Estimation	Substitution	substitution	Recorder_class
1	G300-H	S5AS-T	1.005195914	-0.030179731	0.999371332	0.027136555	L40		1	Y	MOSCAD
2	G300-H	G302-H	1.004695386	-0.04849948	0.997432865	0.054836335	L40		2	Y	MOSCAD
3	G300-H	S5A-T	1.001548147	0.039566696	0.994513606	0.080165514	L40		3	Y	MOSCAD
4	G300-H	G301-H	0.99003546	0.168460644	0.95540825	0.228544801	L40	1		N	MOSCAD
5	G300-H	G311-H	0.869005502	1.971485428	0.739309027	0.552595173	L40	2		N	MOSCAD
6	G301-H	S5A-T	0.969704003	0.542560484	0.956440085	0.223013625	L40	1		N	MOSCAD
7	G301-H	G302-H	0.97091	0.486908039	0.955619769	0.225103715	L40	2		N	MOSCAD
8	G301-H	G300-H	0.965024496	0.55549461	0.95540825	0.225639506	L40	3		N	MOSCAD
9	G301-H	S5AS-T	0.969857904	0.529273318	0.954451771	0.228046616	L40	4		N	MOSCAD
10	G301-H	G311-H	0.858300232	2.137671252	0.739897962	0.544953896	L40	5		N	MOSCAD
11	G302-H	G300-H	0.992771433	0.089438771	0.997432865	0.054509958	L40		1	Y	MOSCAD
12	G302-H	S5AS-T	0.997763937	0.062140417	0.996474508	0.063879482	L40		2	Y	MOSCAD
13	G302-H	S5A-T	0.99502019	0.117297384	0.993381312	0.087526083	L40		3	Y	MOSCAD
14	G302-H	G301-H	0.984251876	0.234572846	0.955619769	0.226645083	L40	1		N	MOSCAD
15	G302-H	G311-H	0.86279189	2.045565911	0.737527457	0.551180014	L40	2		N	MOSCAD
16	G311-H	G301-H	0.862051787	2.389333358	0.739897962	0.546143571	L40	1		N	MOSCAD
17	G311-H	G300-H	0.850754472	2.56444918	0.739309027	0.546761522	L40	2		N	MOSCAD
18	G311-H	S5A-T	0.854154606	2.564684734	0.738852213	0.547240363	L40	3		N	MOSCAD
19	G311-H	S5AS-T	0.854738058	2.545788688	0.738089567	0.54803885	L40	4		N	MOSCAD
20	G311-H	G302-H	0.854816515	2.522103598	0.737527457	0.548626633	L40	5		N	MOSCAD
21	S5A-T	G300-H	0.99297636	0.04874941	0.994513606	0.079821727	L40		1	Y	MOSCAD
22	S5A-T	S5AS-T	0.998185182	0.017988369	0.993986773	0.083566351	L40		2	Y	MOSCAD
23	S5A-T	G302-H	0.998352953	-0.010896458	0.993381312	0.087672542	L40		3	Y	MOSCAD
24	S5A-T	G301-H	0.986321906	0.163851124	0.956440085	0.22491641	L40	1		N	MOSCAD
25	S5A-T	G311-H	0.865011439	1.972063723	0.738852213	0.550707263	L40	2		N	MOSCAD
26	S5AS-T	G300-H	0.994205529	0.040099876	0.999371332	0.026987798	L40		1	Y	MOSCAD
27	S5AS-T	G302-H	0.998707699	-0.00544848	0.996474508	0.063909686	L40		2	Y	MOSCAD
28	S5AS-T	S5A-T	0.995793988	0.078646472	0.993986773	0.083466197	L40		3	Y	MOSCAD
29	S5AS-T	G301-H	0.984115294	0.210538241	0.954451771	0.229716699	L40	1		N	MOSCAD
30	S5AS-T	G311-H	0.863528722	2.007342877	0.738089567	0.550849829	L40	2		N	MOSCAD

Exhibit 2 - data_prep_model.R.txt

```
model_output<-function(dat){
  dat1<-dat %>%filter(!BKPT_TAG %in% c('?','M','m'))
  dat1<- dat1%>% dplyr::select (BKPT_DATE,STATION_ID,BKPT_VALUE) #select required
variables

  #Variables to capture the required format of results############

  out_start=1
  out_end= length(unique(dat$STATION_ID))
  out_nvar=out_end-out_start+1

  out_variable=rep(NA, out_nvar)
  out_beta=rep(NA, out_nvar)
  out_intercept=rep(NA, out_nvar)
  out_se = rep(NA, out_nvar)
  out_pvalue=rep(NA, out_nvar)
  out_adj.r.squared=rep(NA,out_nvar)
  out_rms=rep(NA,out_nvar)
  out_se_int=rep(NA,out_nvar)
  out_se_slope=rep(NA,out_nvar)
  out_rms_p = rep(NA,out_nvar)
  out_map_p = rep(NA,out_nvar)
  # exposure
  exp_start=1
  exp_end= length(unique(dat$STATION_ID))
  exp_nvar=exp_end-exp_start+1

  exp_variable=rep(NA, exp_nvar)
  exp_beta=rep(NA, exp_nvar)
  exp_intercept=rep(NA, out_nvar)
  exp_se = rep(NA, out_nvar)
  exp_pvalue=rep(NA, exp_nvar)
  exp_adj.r.squared=rep(NA,out_nvar)
  exp_rms=rep(NA,out_nvar)
  exp_se_int=rep(NA,out_nvar)
  exp_se_slope=rep(NA,out_nvar)
  exp_rms_p = rep(NA,out_nvar)
  exp_map_p = rep(NA,out_nvar)
  number=1

############All the pairs capturing#################
  tictoc::tic("pairs capture")
  for (i in out_start:out_end){
    outcome = unique(dat1$STATION_ID)[i]
```

```
    for (j in exp_start:exp_end){
      exposure = unique(dat1$STATION_ID)[j]
      if(outcome==exposure) next
      tempdf <- dat1%>%filter(STATION_ID %in% c(outcome,exposure))
      # tempdf<-tempdf %>%
      #   group_by_at(vars(-BKPT_VALUE)) %>%  # group by everything other than the
value column.
      #   mutate(row_id=1:n()) %>% ungroup() %>%  # build group index
      #   spread(key=STATION_ID, value=BKPT_VALUE) %>%    # Data reshaping
      #   select(-row_id)
      tempdf<- tempdf %>% spread(STATION_ID,BKPT_VALUE) %>% select(-BKPT_DATE)
      tempdf<-na.interpolation(tempdf,option="linear")       #Dynamic interpolation
      model <- lm(base::get(outcome) ~ base::get(exposure),data=tempdf)
      beta <- coef(model)
      intercept<-coef(model)[[1]]
      adj.r.squared<-summary(model)$adj.r.squared
      rms<-summary(model)$sigma
      rms[is.null(rms)] <- 0
      se_int<-coef(summary(model))[[3]]
      se_slope<-coef(summary(model))[[4]]
      out_beta[number] = as.numeric(beta[2])
      out_intercept[number] = as.numeric(intercept)
      out_adj.r.squared[number] = as.numeric(adj.r.squared)
      out_rms[number] = as.numeric(rms)
      out_se_int[number]=as.numeric(se_int)
      out_se_slope[number]=as.numeric(se_slope)
      out_variable[number] = outcome
      number = number + 1
      exp_beta[number] = as.numeric(beta[2])
      exp_intercept[number] = as.numeric(intercept)
      exp_adj.r.squared[number] = as.numeric(adj.r.squared)
      exp_rms[number] = as.numeric(rms)
      exp_se_int[number]=as.numeric(se_int)
      exp_se_slope[number]=as.numeric(se_slope)
      exp_variable[number] = exposure
      number = number + 1
    }
  }
  tictoc::toc()


  tictoc::tic("data output consolidation")
  outcome = data.frame(out_variable, out_beta,out_intercept, out_adj.r.squared,
out_rms,out_se_int,out_se_slope)
  exposure = data.frame(exp_variable, exp_beta,exp_intercept, exp_adj.r.squared,
exp_rms,exp_se_int,exp_se_slope)
  tictoc::toc()
  tictoc::tic("data output1 consolidation")
```

```
  #library(tidyverse)
  outcome = outcome %>%
    rename(
      variable = out_variable,
      beta = out_beta,
      intercept=out_intercept,
      adj.r.squared=out_adj.r.squared,
      rms=out_rms,
      se_int=out_se_int,
      se_slope=out_se_slope
    )
  exposure = exposure %>%
    rename(
      variable = exp_variable,
      beta = exp_beta,
      intercept=exp_intercept,
      adj.r.squared=exp_adj.r.squared,
      rms=exp_rms,
      se_int=exp_se_int,
      se_slope=exp_se_slope
    )
  tictoc::toc()

  exposure1<-exposure%>%rename(variable1=variable)
  tictoc::tic("data output2 consolidation")
  all2<-outcome %>% dplyr::inner_join(exposure1, by =
c("beta","intercept","adj.r.squared","rms","se_int","se_slope"))
  all2 = na.omit(all2)

all3<-all2%>%select(variable,variable1,beta,intercept,adj.r.squared,rms,se_int,se_sl
ope)%>%filter(variable!=variable1)
  tictoc::toc()
  return(all3)

}
```

Exhibit 3 - Substitution_Ranking.R.txt

```
sub_rank <-function(res,r.sqr){
  fin <- res %>% rowwise() %>% mutate(substition=if(adj.r.squared>r.sqr){1}else 0)
  fin<-fin %>% filter(substition==1)
  #fin <- fin%>% dplyr::select(-c(rnk,beta,intercept,se_int,se_slope,`Pair ID`))
  fin <- fin%>% dplyr::select(-c(rnk,beta,intercept,se_int,se_slope))
  fin<-fin %>% dplyr::group_by(variable) %>% mutate(sub_rnk=row_number())
  return(fin)
}


# res<-est_rank(m)
# n<-sub_rank(res,0.97)
```

Exhibit 4 - Estimation_Ranking.R.txt

```
est_rank<-function(res){

  #dis<-readxl::read_xlsx("//whqhome01p/home$/sbandaru/Desktop/New
folder/week11/dist.xlsx")
  #dis<-readxl::read_xlsx("//whqhome01p/home$/sbandaru/Desktop/New
folder/week12/dist1.xlsx")
  #all4<-all3
  #res<-res %>% inner_join(dis,by=c("variable"="Station ID1","variable1"="Station
ID2"))
  #res<-res %>% group_by(variable) %>% arrange(rms,`Distance
(miles)`,desc(adj.r.squared))
  res<-res %>% group_by(variable) %>% arrange(rms,desc(adj.r.squared))
  res <- res %>% arrange(variable)%>% mutate(rnk=row_number())
  return(res)
}

#r.sqr<-0.95
```

Exhibit 5 - hist_summary_stat.R.txt

```
myfunc<-function(begin.date,end.date){

  ######Data Extraction################
  SQLstring <- "select to_char(BKPT_DATE, 'mm/dd/yyyy hh24:mi:ss') as
BKPT_DATE,STATION_ID,BKPT_VALUE,BKPT_TAG from dcvp_archive WHERE BKPT_DATE BETWEEN ?
AND ? and STATION_ID in ('S5A-T', 'S5AS-T', 'G300-H', 'G301-H', 'G302-H','G311-H')"
  dat_s <- dbGetQuery(conn, SQLstring, begin.date,end.date)
  #########Data preprocessing###########
  dat_s<-as.data.table(dat_s)
  dat_s$BKPT_DATE<-parse_date_time(dat_s$BKPT_DATE, orders="mdY hms")
  dat_s$BKPT_VALUE<-as.numeric(dat_s$BKPT_VALUE)
  dat_s$BKPT_TAG[is.na(dat_s$BKPT_TAG)] <- "TE"
  ##########Data processing#############
  dcvp_arc_3m_roc<-dat_s %>%
    dplyr::select (BKPT_DATE,STATION_ID,BKPT_VALUE,BKPT_TAG) %>%
    dplyr::group_by(STATION_ID) %>%
    arrange(BKPT_DATE) %>%
    mutate(lag1 = lag(BKPT_VALUE, n = 1),lag2=lag(BKPT_DATE,n=1)) %>%
    mutate(ROC_TIME=as.numeric(difftime(BKPT_DATE,lag2),units="mins")) %>%
    mutate(ROC_VALUE=BKPT_VALUE-lag1)%>%
    mutate(ROC_abs=abs(ROC_VALUE/ROC_TIME))%>%
    #mutate(ROC=ROC_VALUE/ROC_TIME)%>%
    dplyr::filter(BKPT_TAG ==lag(BKPT_TAG) & BKPT_TAG == "TE")%>%
    group_by(STATION_ID) %>%
    mutate(MINIMUM=min(BKPT_VALUE),
           MAXIMUM=max(BKPT_VALUE),
           MEAN=mean(BKPT_VALUE,na.rm=FALSE),
           STANDARDDEVIATION=sd(BKPT_VALUE),
           #max_roc=max(ROC),
           #min_roc=min(ROC),
           #avg_roc=mean(ROC),
           MAXIMUM_RATE_OF_CHANGE=max(ROC_abs),
           MINIMIM_RATE_OF_CHANGE=min(ROC_abs),
           AVERAGE_RATE_OF_CHANGE=mean(ROC_abs),
           BEGIN_DATE=as.Date(min(dat_s$BKPT_DATE)),
           END_DATE=as.Date(max(dat_s$BKPT_DATE)),
           COUNT=n()) %>%
    #arrange(BKPT_DATE) %>%

select(BEGIN_DATE,END_DATE,STATION_ID,BKPT_TAG,MINIMUM,MAXIMUM,MEAN,STANDARDDEVIATIO
N,MAXIMUM_RATE_OF_CHANGE,MINIMIM_RATE_OF_CHANGE,AVERAGE_RATE_OF_CHANGE,COUNT)%>%
    distinct()
  return(dcvp_arc_3m_roc)
  #evals('myfunc()')
```

```
}
res<-myfunc('01-Jan-2017','01-Feb-2017')
```

Exhibit 6- Hist_summary_stat_write.R.txt

```
#!/usr/bin/Rscript
options(java.parameters = "-Xmx8g")
#library(RJDBC)
library(RJDBC)
library(dplyr)
library(quantmod)
library(tidyr)
library(sqldf)
library(tibbletime)
library(data.table)
library(lubridate)
library(ggplot2)
library(imputeTS)
library(Metrics)
library(lattice)
library(tibble)
library(readxl)
#install.packages("tibbletime")
jdbcDriver=JDBC("oracle.jdbc.OracleDriver",classPath = "C:/test/ojdbc7.jar")
conn<- dbConnect(jdbcDriver,"","","")

myfunc<-function(begin.date,end.date){
  #args <- commandArgs(trailingOnly = TRUE)
  #begin.date<-args[1]
  #end.date<-args[2]
  ######Data Extraction###############
  SQLstring <- "select to_char(BKPT_DATE, 'mm/dd/yyyy hh24:mi:ss') as
BKPT_DATE,s.TIME_SERIES_ID,d.BKPT_VALUE,d.BKPT_TAG from station_reference s,
dcvp_archive d WHERE BKPT_DATE BETWEEN ? AND ? and d.STATION_ID in ('S5A-T',
'S5AS-T', 'G300-H', 'G301-H', 'G302-H','G311-H') and d.station_id=s.station_id"
  dat_s <- dbGetQuery(conn, SQLstring, begin.date,end.date)
  #begin.date<-'01-Jan-2016'
  #end.date<-'01-Jun-2016'
  #########Data preprocessing###########
  dat_s<-as.data.table(dat_s)
  dat_s$BKPT_DATE<-parse_date_time(dat_s$BKPT_DATE, orders="mdY hms")
  dat_s$BKPT_VALUE<-as.numeric(dat_s$BKPT_VALUE)
  dat_s$BKPT_TAG[is.na(dat_s$BKPT_TAG)] <- "TE"
  ##########Data processing############
  dcvp_arc_3m_roc<-dat_s %>%
    dplyr::select (BKPT_DATE,TIME_SERIES_ID,BKPT_VALUE,BKPT_TAG) %>%
    dplyr::group_by(TIME_SERIES_ID) %>%
    arrange(BKPT_DATE) %>%
    mutate(lag1 = lag(BKPT_VALUE, n = 1),lag2=lag(BKPT_DATE,n=1)) %>%
    mutate(ROC_TIME=as.numeric(difftime(BKPT_DATE,lag2),units="mins")) %>%
```

```
    mutate(ROC_VALUE=BKPT_VALUE-lag1)%>%
    mutate(ROC_abs=abs(ROC_VALUE/ROC_TIME))%>%
    dplyr::filter(BKPT_TAG ==lag(BKPT_TAG) & BKPT_TAG == "TE")%>%
    group_by(TIME_SERIES_ID) %>%
    mutate(MINIMUM=min(BKPT_VALUE),
           MAXIMUM=max(BKPT_VALUE),
           MEAN=mean(BKPT_VALUE,na.rm=FALSE),
           STANDARDDEVIATION=sd(BKPT_VALUE),
           MAXIMUM_RATE_OF_CHANGE=max(ROC_abs),
           MINIMIM_RATE_OF_CHANGE=min(ROC_abs),
           AVERAGE_RATE_OF_CHANGE=mean(ROC_abs),
           FROM_DATE=as.Date(min(dat_s$BKPT_DATE)),
           TO_DATE=as.Date(max(dat_s$BKPT_DATE)),
           COUNT=n()) %>%

select(FROM_DATE,TO_DATE,TIME_SERIES_ID,BKPT_TAG,MINIMUM,MAXIMUM,MEAN,STANDARDDEVIAT
ION,MAXIMUM_RATE_OF_CHANGE,MINIMIM_RATE_OF_CHANGE,AVERAGE_RATE_OF_CHANGE,COUNT)%>%
    distinct()
  #return(dcvp_arc_3m_roc)
  #evals('myfunc()')
  ################### Data Gathering  ######################

  stat_by_date<-dbGetQuery(conn,"select max(ID) as ID from
TIMESERIES_HIST_STAT_BY_DATE" )
  stat_by_date_rge<-dbGetQuery(conn,"select max(ID) as ID from
TIMESERIES_HIST_STAT_DATE_RGE" )
  tl_ts_stat<-dbGetQuery(conn,"select ID,STATISTIC_NAME from
TL_TIMESERIES_STATISTIC" )

  ###############Get the station id#############################
  library(sqldf)
  k1_s<-dcvp_arc_3m_roc %>% gather(measure,stat,MINIMUM:COUNT) #Get the measure
row_wise
  k2_s<-sqldf("select * from k1_s k join tl_ts_stat s where
k.measure==s.STATISTIC_NAME")
  k2_s<-k2_s%>%select(-measure)
  k2_s<-k2_s%>%dplyr::group_by(TIME_SERIES_ID,FROM_DATE,TO_DATE) %>% {
    mutate(ungroup(.),gr_id=group_indices(.))}

  ##################Date_rge#################################

  date_rge<-k2_s%>%mutate(ID=gr_id)%>%select(ID,TIME_SERIES_ID,FROM_DATE,TO_DATE)
  date_rge$CREATED_BY<-"TEST"
  date_rge$DATE_CREATED<-Sys.Date()
  date_rge$LAST_MODIFIED_BY<-Sys.Date()
  date_rge$LAST_MODIFIED_DATE<-Sys.Date()
  date_rge<-date_rge%>%dplyr::distinct()%>%arrange(ID)
  latest<-stat_by_date_rge$ID
```

```
  s1<-seq_along(date_rge$ID)
  s2<-s1+latest
  s3<-rep.int(s2,8)
  date_rge$ID<-s2

  ###################stat_date#####################################


stat_date<-k2_s%>%mutate(ID1=row_number(),DATE_RGE_ID=gr_id,STATISTIC_NAME_REF_ID=ID
,VALUE=stat)%>%
    select(-ID)%>%
    mutate(ID=ID1,THRESHOLD=0)%>%
    select(ID,DATE_RGE_ID,STATISTIC_NAME_REF_ID,VALUE,THRESHOLD)

  stat_date$CREATED_BY<-"TEST"
  stat_date$DATE_CREATED<-Sys.Date()
  stat_date$LAST_MODIFIED_BY<-"TEST"
  stat_date$LAST_MODIFIED_DATE<-Sys.Date()

  stat_date<-stat_date%>%mutate(DATE_RGE_ID=s3)
  latest_stat<-stat_by_date$ID #1
  s4<-seq_along(stat_date$ID) #2
  s5<-s4+latest_stat
  stat_date$ID<-s5
  #############Conversions for db writing#########################

  date_rge$FROM_DATE<-format.Date(date_rge$FROM_DATE,"%d-%b-%y")
  date_rge$TO_DATE<-format.Date(date_rge$TO_DATE,"%d-%b-%y")
  date_rge$DATE_CREATED<-format.Date(date_rge$DATE_CREATED,"%d-%b-%y")
  date_rge$LAST_MODIFIED_BY<-format.Date(date_rge$LAST_MODIFIED_BY,"%d-%b-%y")
  date_rge$LAST_MODIFIED_DATE<-format.Date(date_rge$LAST_MODIFIED_DATE,"%d-%b-%y")
  stat_date$DATE_CREATED<-format.Date(stat_date$DATE_CREATED,"%d-%b-%y")
  stat_date$LAST_MODIFIED_DATE<-format.Date(stat_date$LAST_MODIFIED_DATE,"%d-%b-%y")

  ######################################################################

  dbWriteTable(conn,
"TIMESERIES_HIST_STAT_DATE_RGE",date_rge,overwrite=FALSE,row.names = FALSE,append =
TRUE)
  return(dbWriteTable(conn,
"TIMESERIES_HIST_STAT_BY_DATE",stat_date,overwrite=FALSE,row.names = FALSE,append =
TRUE))
  #dbWriteTable(conn, "TEST5",date_rge,overwrite=FALSE,row.names = FALSE,append =
TRUE)
  #dbWriteTable(conn, "TEST6",stat_date,overwrite=FALSE,row.names = FALSE,append =
TRUE)
}
```

EXHIBIT 7 - summary_stat.R.txt

```
rm(list=ls(all=TRUE))
options(java.parameters = "-Xmx10g")
#library(RJDBC)
library(RJDBC)
library(dplyr)
library(quantmod)
library(tidyr)
library(sqldf)
library(tibbletime)
library(data.table)
library(lubridate)
library(ggplot2)
library(imputeTS)
library(Metrics)
library(lattice)
library(tibble)
library(readxl)
#install.packages("tibbletime")
jdbcDriver=JDBC("oracle.jdbc.OracleDriver",classPath = "C:/test/ojdbc7.jar")
conn<- dbConnect(jdbcDriver,"","","")


summary_stat<-function(){
  end.date<-Sys.Date()
  begin.date<-end.date-180
  begin.date<-format.Date(begin.date,"%d-%b-%y")
  end.date<-format.Date(end.date,"%d-%b-%y")
  ######Data Extraction################

  #SQLstring <- "select to_char(BKPT_DATE, 'mm/dd/yyyy hh24:mi:ss') as
BKPT_DATE,s.TIME_SERIES_ID,d.BKPT_VALUE,d.BKPT_TAG,s.CURRENT_STATUS from
station_reference s, dcvp_archive d WHERE BKPT_DATE BETWEEN ? AND ?  and d.BKPT_TAG
IS NULL and d.station_id=s.station_id"
  SQLstring <- "select to_char(BKPT_DATE, 'mm/dd/yyyy hh24:mi:ss') as
BKPT_DATE,STATION_ID,BKPT_VALUE,BKPT_TAG from dcvp_archive WHERE BKPT_DATE BETWEEN ?
AND ? "
  # begin.date<-'01-Dec-2016'
  # end.date<-'01-Jun-2017'
  #tictoc::tic("data pull")
  dat_s <- dbGetQuery(conn, SQLstring, begin.date,end.date)
  #tictoc::toc()
  ##########Data preprocessing###########
  #dat_s<-dat_s%>%filter(CURRENT_STATUS=='A')
  dat_s<-as.data.table(dat_s)
  dat_s$BKPT_DATE<-parse_date_time(dat_s$BKPT_DATE, orders="mdY hms")
```

```
  dat_s$BKPT_VALUE<-as.numeric(dat_s$BKPT_VALUE)
  dat_s$BKPT_TAG[is.na(dat_s$BKPT_TAG)] <- "TE"
  ##########Data processing#############
  dcvp_arc_3m_roc<-dat_s %>%
    dplyr::select (BKPT_DATE,STATION_ID,BKPT_VALUE,BKPT_TAG) %>%
    dplyr::group_by(STATION_ID) %>%
    arrange(BKPT_DATE) %>%
    mutate(lag1 = lag(BKPT_VALUE, n = 1),lag2=lag(BKPT_DATE,n=1)) %>%
    mutate(ROC_TIME=as.numeric(difftime(BKPT_DATE,lag2),units="mins")) %>%
    mutate(ROC_VALUE=BKPT_VALUE-lag1)%>%
    mutate(ROC_abs=abs(ROC_VALUE/ROC_TIME))%>%
    #mutate(ROC=ROC_VALUE/ROC_TIME)%>%
    dplyr::filter(BKPT_TAG ==lag(BKPT_TAG) & BKPT_TAG == "TE")%>%
    group_by(STATION_ID) %>%
    mutate(MINIMUM=min(BKPT_VALUE),
           MAXIMUM=max(BKPT_VALUE),
           MEAN=mean(BKPT_VALUE,na.rm=FALSE),
           STANDARDDEVIATION=sd(BKPT_VALUE),
           #max_roc=max(ROC),
           #min_roc=min(ROC),
           #avg_roc=mean(ROC),
           MAXIMUM_RATE_OF_CHANGE=max(ROC_abs),
           MINIMIM_RATE_OF_CHANGE=min(ROC_abs),
           AVERAGE_RATE_OF_CHANGE=mean(ROC_abs),
           FROM_DATE=begin.date, #as.Date(min(dat_s$BKPT_DATE)),
           TO_DATE=end.date, #as.Date(max(dat_s$BKPT_DATE)),
           COUNT=n()) %>%
    #arrange(BKPT_DATE) %>%

select(FROM_DATE,TO_DATE,STATION_ID,BKPT_TAG,MINIMUM,MAXIMUM,MEAN,STANDARDDEVIATION,
MAXIMUM_RATE_OF_CHANGE,MINIMIM_RATE_OF_CHANGE,AVERAGE_RATE_OF_CHANGE,COUNT)%>%
    distinct()
  #return(dcvp_arc_3m_roc)

  ################### Data Gathering  ########################

  stat_by_date<-dbGetQuery(conn,"select max(ID) as ID from
TIMESERIES_HIST_STAT_BY_DATE" )
  stat_by_date_rge<-dbGetQuery(conn,"select max(ID) as ID from
TIMESERIES_HIST_STAT_DATE_RGE" )
  tl_ts_stat<-dbGetQuery(conn,"select ID,STATISTIC_NAME from
TL_TIMESERIES_STATISTIC" )
  station<-dbGetQuery(conn,"select STATION_ID,TIME_SERIES_ID from station_reference
where current_status='A'" )
  #station<-station%>%select(STATION_ID,TIME_SERIES_ID)
  ################Get the station id##############################
  library(sqldf)
  k<-sqldf("select * from dcvp_arc_3m_roc d join station s where
```

```
d.STATION_ID==s.STATION_ID")
  k1<-k %>% select(-STATION_ID..13,-STATION_ID)
  k1_s<-k1 %>% gather(measure,stat,MINIMUM:COUNT) #Get the measure row_wise
  #k1_s<-k1 %>% gather(measure,stat,MINIMUM:COUNT)
  k2_s<-sqldf("select * from k1_s k join tl_ts_stat s where
k.measure==s.STATISTIC_NAME")
  k2_s<-k2_s%>%select(-measure)
  k2_s<-k2_s%>%dplyr::group_by(TIME_SERIES_ID,FROM_DATE,TO_DATE) %>% {
    mutate(ungroup(.),gr_id=group_indices(.))}

  ###################Date_rge#################################

  date_rge<-k2_s%>%mutate(ID=gr_id)%>%select(ID,TIME_SERIES_ID,FROM_DATE,TO_DATE)
  date_rge$CREATED_BY<-"TEST"
  date_rge$DATE_CREATED<-Sys.Date()
  date_rge$LAST_MODIFIED_BY<-Sys.Date()
  date_rge$LAST_MODIFIED_DATE<-Sys.Date()
  date_rge<-date_rge%>%dplyr::distinct()%>%arrange(ID)
  latest<-stat_by_date_rge$ID
  s1<-seq_along(date_rge$ID)
  s2<-s1+latest
  s3<-rep.int(s2,8)
  date_rge$ID<-s2

  ####################stat_date#####################################


stat_date<-k2_s%>%mutate(ID1=row_number(),DATE_RGE_ID=gr_id,STATISTIC_NAME_REF_ID=ID
,VALUE=stat)%>%
    select(-ID)%>%
    mutate(ID=ID1,THRESHOLD=0)%>%
    select(ID,DATE_RGE_ID,STATISTIC_NAME_REF_ID,VALUE,THRESHOLD)

  stat_date$CREATED_BY<-"TEST"
  stat_date$DATE_CREATED<-Sys.Date()
  stat_date$LAST_MODIFIED_BY<-"TEST"
  stat_date$LAST_MODIFIED_DATE<-Sys.Date()
  stat_date<-stat_date%>%arrange(DATE_RGE_ID)  #use latest date_rge_id
  s3<-sort(s3)
  stat_date<-stat_date%>%mutate(DATE_RGE_ID=s3)
  latest_stat<-stat_by_date$ID #1
  s4<-seq_along(stat_date$ID) #2
  s5<-s4+latest_stat
  stat_date$ID<-s5
  #############Conversions for db writing#########################

  #date_rge$FROM_DATE<-format.Date(date_rge$FROM_DATE,"%d-%b-%y")
  #date_rge$TO_DATE<-format.Date(date_rge$TO_DATE,"%d-%b-%y")
```