Reducing Biases in Individual Software Effort Estimations: A Combining Approach

Qi Li[1,2], Qing Wang[1], Ye Yang[1], Mingshu Li[1]
[1]Institute of Software, Chinese Academy of Sciences, China
[2]Graduate University of Chinese Academy of Sciences, China
{liqi,wq,ye,mingshu}@itechs.iscas.ac.cn

ABSTRACT
Software effort estimation techniques abound, each with its own set of advantages and disadvantages, and no one proves to be the single best answer. Combining estimating is an appealing approach. Avoiding the difficult problem of choosing the single "best" technique, it solves the problem by asking which techniques would help to improve accuracy, assuming that each has something to contribute. In this paper, we firstly introduce the systematic "external" combining idea into the field of software effort estimation, and estimate software effort using Optimal Linear Combining (OLC) method with an experimental study based on a real-life data set. The result indicates that combining different techniques can significantly improve the accuracy and consistency of software effort estimation by making full use of information provided by all components, even the much "worse" one.

Categories and Subject Descriptors
D.2.9 [**Software Engineering**]: Management–*Cost estimation*

General Terms: Algorithms, Management, Measurement, Economics, Experimentation

Keywords: OLC, Software Effort Estimation, Combining

1. INTRODUCTION
Accurate and consistent estimates of software effort are crucial for better project planning, monitoring, and control[1]. In pactice, effort estimation is one of the fields containing the most experiences. A lot of research has aimed to build, evaluate and recommend estimation techniques to help managers make more accurate and consistent effort estimates, e.g.[2, 3]. They employ one or more methods: parametric modeling, knowledge-based modeling, rule induction, fuzzy logic, dynamic modeling, neural networks, or case-based reasoning [4-6]. Recent research has attempted to determine which technique or tool might be the "best" by comparing them on the sense of one or more performance measures on some specific data sets and conditions[7, 8]. However, the outcomes of these studies do not always correspond, due to different data sets and experiment conditions[9, 10]. Common effort estimation practice in particular organizations displays that the single technique they adopts performs well on some projects, but badly on others. For these reasons, project managers are often confused about how to choose the "best" software effort estimation technique or tool. In fact, the always "best" technique or tool for all situations does not exist in real life[11]. Meanwhile, experiences and MacDonell and Martin J. Shepperd's initial research[9]have indicated that when one technique predicts poorly, one or more other techniques tend to perform significantly better, which implies that some techniques or tools are complementary and may have different contributions for accurate estimation. Best practices recommends that project managers should use at least two approaches[1, 12] since many factors affect the estimation and these might be captured by using alternative approaches. It has come to a consensus that combining estimating may help integrate estimating knowledge acquired by component methods, reduce errors deriving from faulty assumptions, bias, or mistakes in data and improve the estimation accuracy.

In this paper, we examine the literature on combining forecasts especially the linear combining methods in statistics science in section 2.1. This work helps to introduce the systematic "external" combining idea into the field of software effort estimation, and assist researchers that are interested in combining estimates and likely to do more extensive and thorough research in this field. We then investigate the application of current combining methods in software effort estimation to ensure our research's originality in section 2.2. In section 3, we give an overview of the most typical linear method: Optimal Linear Combination (OLC) method [13]for software effort estimation. To evaluate our method's effectiveness we introduce some performance measures for OLC method in section 4. Then we discuss the OLC method in details and apply it step by step to software effort estimation with an experimental study based on a real-life data set in section 5.1-5.4. Finally, in section 6, 7, 8, we discuss some lessons learned from the experiment, threats of validity, our future work and eagerness to call for more attention to explore the software effort combining estimation field.

2. RELATED WORK

2.1 Literature on Combining Forecasts
As early as 1818, Laplace initially proposed that "In combining the results of two methods, one can obtain a result whose probability law of error will be more rapidly decreasing". J.Scott Armstrong defines Combining Forecasts as "Combining forecasts, sometimes referred to as composite forecasts, refer to the averaging of independent forecasts. These forecasts can be based on different data or different methods or both. The averaging is done using a rule that can be replicated, such as to take a simple average of the forecasts" [14]. Since the influential

ESEM'08, October 9-10, 2008, Kaiserslautern, Germany.

work of Bates and Granger's [15],combining forecasting techniques have been rapidly developed and widely used in many practical field such as whether forecasting, money market, macro-economics analysis etc. with considerable success. The methods now available to the forecaster range from the robust simple average to the far more theoretically complex ones. Clemen [16]traces the literature back to Laplace, cites more than 200 studies in his review of the literature related to combining forecasts, including contributions from forecasting, psychology, statistics, and management science literatures. Some researchers [14, 17]provide guidelines for combining forecasts. In [18] a brief review on the linear combining methods is presented: in addition to simple combining methods based on averaging[14, 19], there are other combining methods based on ordinary least square[20], weighted least squares[21], discount MSFE[22], Bayesian shrinkage techniques[23], clusters[24], model selection[25], principal components[26], approximate Bayesian model averaging[27] and exponential reweighing[28].

In general, linear methods are often recommended by forecasters [14, 29, 30]: "First , all the component forecasting methods are trained to approximate the same actual value. Thus, forming a weighted sum of the corresponding outputs of components is directly comprehended. Besides, linear methods are often simpler to analyze and easier to implement than non-linear methods". The implementations of the four OLCs, discussed in Part 5, require modest computational effort that mainly involves a matrix inverse[30]. In the literature on combining forecasts, linear formulation is almost always adopted.

2.2 Literature on Composite Estimation Methods for Software Effort

In this section, we first differentiate the "composite estimation method" term frequently used in the software effort estimation field from our mentioned "combining estimation method" in this paper, and then we examine current combining methods or practices employed in software effort estimation, discuss their drawbacks, and propose the OLC method at last.

Based on reviews of software effort estimation approaches[4-6], three current typical "composite estimation methods" are frequently referred in the software effort estimation field, they are Cost Estimation Benchmarking and Risk Analysis (COBRA)[3], Incorporating Bayesian analysis to improve the accuracy of COCOMO II[2, 31] and Analogy-Based tools[32, 33]. However, the three methods are different from J.Scott Armstrong's definition of combining forecasts. Technically, they all combine experiential approaches with data-driven modeling, but expert knowledge used in the three methods is just a part of the modeling process: COBRA uses expert knowledge to construct the causal model; Analogy-Based tools use it to define similarity functions, and COCOMO II combine it with historical data to calibrate the value of cost drivers by Bayesian Analysis. This kind of composite technique is "internal" combining and in some sense, they all belong to individual techniques. In this paper, we refer "combining estimation method" as to combine the outputs (final effort estimates) of individual methods by simple average or a sophisticated weighting algorithm, as Armstrong's definition mentioned in section 2.1. For differentiating from "internal" combining, we call it "external" combining.

Although the "external" combining has been widely used in many other fields with considerable success, we find few related work [9, 34, 35] in the field of software effort estimation. MacDonell and Martin J. Shepperd's analysis [9]of effort data from a medical records information system reveals that there is little, or even negative, covariance between the accuracy of their three chosen estimating techniques (expert judgment, least squares regression and case-based reasoning). That indicates that when one technique predicts poorly, one or both of the others tends to perform significantly better. This comes to the conclusion that it might be valuable to combine estimation techniques for a given environment when there is no dominant technique. They also try to learn a decision tree to decide which technique to use in which circumstances by means of rule induction. However, they aren't able to identify a means of determining the prior and also don't give any suggestion on how to combine not optimal techniques to generate better estimation.

Magne Jorgensen considers combining estimation methods as one of practical guidelines for expert-judgment-based software effort estimation and suggest some useful combinations of estimation methods[34], such as expert judgments and formal methods, top-down and bottom-up methods, analogy and linear regression methods, expert judgments made by software professionals with different project experiences, expert judgments made by software professionals with different roles. However, he doesn't provide any practical guidelines on how to combine these estimation methods with some empirical studies.

One of the few studies to directly explore software effort combining estimation is Kitchenham's work based on estimation data from Computer Sciences Corporation[35]. "They had multiple estimates and used one of two strategies: either to take a simple average of the estimates or to select one that they decided they would use (almost always an expert opinion estimate)"[9]. Simple average might be easy to implement, but it gives each component the same weight without considering different contributions of individual techniques, so it can't always ensure that the combining accuracy is at least as high as that of the best component. Besides, selecting the "best" one by expert opinion might be influenced by subjective bias and even not so, ignore the fact that other components might provide useful information for accuracy. The OLC method gives components different weights according to their performances, can make full use of information provided by each component to maximize the accuracy in prediction, and often produces superior model accuracy, at least, not worse than the best component.

3. OVERVIEW OF OLC METHOD FOR SOFTWARE EFFORT ESTIMATION

Figure 1 illustrates the steps of software effort estimation with OLC. This method includes four main steps as shown in the left-side flow chart, and each main step's sub steps are numbered in the process rectangles of the right-side flow chart.

1).Preparing data and component methods. For better estimation, data used for modeling OLC is required to be from the same organization, that is to say, the company-specific data and preferably of the same project type, in addition component methods should be "different" enough to benefit combining. This step includes *Processing Rectangle 1*, *2* in the right flow chart (details in section 5.1).

2).OLC modeling. OLC modeling in essence is the multiple regression analysis (estimates of components as independent variables and actual effort as the attributive variable) using Ordinary Least Square[36], and we should select a proper case from four optional cases of OLCs according to their estimating performance. This step corresponds with *Processing Rectangle 3* in the right flow chart (details in section 5.2).

3).Further improving OLC's predictive power. Since the outputs of components are trained to approximate the same actual value, we would expect them to be highly collinear and in statistics it is named as collinearity. Considering collinearity might ruin the predictive power of OLC, we improve the predictive power by dropping a component or the constant term to weaken collinearity. This step includes *Processing Rectangle 4, 5* in the right flow chart (details in section 5.3).

4).Returning the final estimating model. If there is no room to further improve the predictive power of OLC, we should compare estimating performances of the current OLC, the best component method and the simple average to return the one with the best performance as our final estimating model. This step corresponds with *Processing Rectangle 6* in the right flow chart (details in section 5.4).

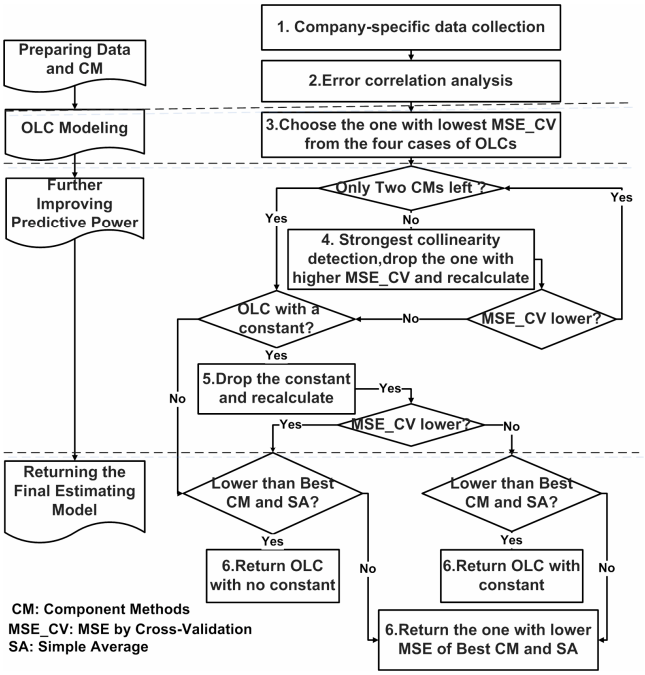

Figure 1. Software Effort Estimation Using OLC Method

4. PERFORMANCE MEASURES FOR OLC

To evaluate the performance of estimating methods, a series of estimating performance measures are commonly used. Assume $\hat{Y}$ denotes the estimating value, Y denotes the actual value. Table 1 presents the commonly used performance measures. As for a group of observations, the last asterisked four are used.

Table 1. Performance Measures

Performance Measures	Calculating Formula
RE (Relative Error)	$RE_t = \frac{\hat{Y}_t - Y_t}{Y_t}$
MRE (Magnitude of Relative Error)	$MRE_t = \left\lvert \frac{\hat{Y}_t - Y_t}{Y_t} \right\rvert$
****MMRE*** (Mean MRE)	$MMRE = \frac{1}{N}\sum_{t=1}^{N} MRE_t$
****MSE*** (Mean Squared Error)	$MSE = \frac{1}{N} * \sum_{t=1}^{N} \left[\hat{Y}_t - Y_t\right]^2$
****MSRE*** (Mean Squared Relative Error)	$MSRE = \frac{1}{N} * \sum_{t=1}^{N} \left[MRE_t\right]^2$
****SD*** *(*Standard Deviation of RE*)*	$SD = \sqrt{\frac{N * \sum_{t=1}^{N} RE_t^2 - \left(\sum_{t=1}^{N} RE_t\right)^2}{N*(N-1)}}$

MMRE, MSRE and MSE are strongly relevant measures for evaluating accuracy and they usually have the same tendency to show the accuracy. Generally, the lower MMRE, MSRE and MSE are, the more accurate the estimating method is. However, MSE, unlike MMRE, is highly sensitive to outliers. It penalizes a forecasting method much more for large errors than for small errors. Forecasting practices recommend using multiple error measures[37]. So in our research, we adopt them all for our accuracy measures. As for estimating consistency, we employ Standard Deviation of RE to reflect it. The lower SD is, the more consistent the estimating method is.

Besides, one of the most important performance measures is the performance measured on a separate data set from training data. This performance measure is referred to as the predictive power (including both predictive accuracy and consistency), the out-of-sample performance, the generalization ability, or the robustness. Cross-validation is one of several approaches to estimate how well the model just learned from some training data is going to perform on future as-yet-unseen data. There are two kinds of cross-validation: k-fold cross-validation and leave-one-out cross-validation (LOOCV). Generally, 10-fold cross-validation is the standard way of measuring the error rate of a learning scheme on a particular data set. For reliable results, 10 times 10-fold cross validation is suitable. LOOCV is useful because it does not waste data, this is especially true when the sample size small[38].

As for combining estimation methods' predictive power, intuitively, it should be judged by whether combination can improve the predictive power from the best component method, and if so, how significant is the improvement. We also adopt the simple average as one of the comparative baseline methods for it is very simple to implement and a lot of researches recommend it as one of the most effective combining methods, e.g.[14, 17]. The smaller the MSE, MSRE, MMRE and SD of the combining method than those of the best component method and the simple average, the more effective the combining method is. In this paper, because OLC aims at minimize the MSE (details in section 5.2), we consider MSE as the main performance measure. The MSE of the OLC after cross-validation may be compared to that of the best component method and the simple average in order to determine the predictive power's improvement. That is to say: If the MSE of OLC by cross-

validation is smaller than that of the best component method and that of the simple average, the OLC is robust or has sound predictive power.

5. OLC METHOD AND AN EXPERIMENTAL STUDY

In this section, we extend the OLC method in details step by step. In each part of 5.1-5.4, corresponding to the four steps of the OLC method respectively, we first discuss each step and then apply them to our experiment.

5.1 Preparing Data and Component Methods

This step is prerequisite for OLC modeling. The data we need for OLC modeling is the estimates of local projects from different component methods. The data required should be from the same organization, that is to say, the company-specific data and preferably of the same project type, and in such situation, we can find proper OLC for specific local environment. This can be shown in *Processing Rectangle 1* in Figure 1.

Many factors affect the estimation and these might be captured by using alternative approaches. If all the components carry the same information, no benefits may be expected from combining them. To increase the benefits of combining, the components may be constructed using different topologies, structure, different learning algorithms, core techniques, different training data etc.[14, 39, 40]. Error correlation analysis helps to reflect the "difference" between individual methods quantificationally and identify whether or at what extent it will benefit from combining. Assume the correlation coefficient is denoted by $r(-1 \le r \le 1)$. According to [41], $r < 0$ means negative correlation, $r > 0$ means positive correlation, $0 < r < 0.5$ means low positive correlation, $r \ge 0.8$ means high positive correlation. Generally, negative, even low positive correlation error indicates that when one technique estimates poorly, the other tends to perform significantly better. The lower error correlation among individual methods, the larger potential for combining, and the more benefits would be gained from combining. This can be shown in *Processing Rectangle 2* in Figure 1.

In this experimental study, we acquire the company-specific data from F.Kemerer's empirical work[42] , his work evaluates four algorithmic models (SLIM, COCOMO, Function Points, and ESTIMACS) based on their respective estimates on 15 large completed business data-processing projects of the same company. It provides 15 projects' estimates of COCOMO, SLIM and Function Points but only 9 projects' estimates of ESTIMACS. In our experimental study, we combine COCOMO, SLIM and Function Points based on 15 projects and exclude ESTIMACS because it lacks other 6 projects' estimates. For short we use their name's initial C, S, and F to denote each. As we know, the three tools have different modeling algorithms, e.g. C and S use SLOC for the project's general size while F uses function points. S uses Rayleigh curve model to produce its effort estimates, C uses easy-to-understand exponential model, while F estimates by a regression of actual effort and its estimating function points. So they have different core techniques and model structures, further more, they all come from different organizations and trained on different data set, thus theoretically they are complementary, low correlated tools and they may have different information to contribute.

In Appendix **1**, RE_C, RE_S, RE_F, RE_Average denote respectively relative errors of the three tools' estimates and simple average on the 15 projects. Table 2 compares the consistency (SD) and accuracy (MSE, MSRE, MMRE) measures' values of the three tools and simple average. Figure 2 presents box plots of the MRE values for each of the techniques applied.

Table 2. Accuracy and Consistency Comparison of C, S, F and Simple Average

	SD	MSE	MSRE	MMRE
C	9.329	3.20E+06	118.188	6.079
S	6.613	1.05E+07	100.398	7.719
F	1.533	2.91E+04	2.312	1.027
Average	4.509	2.53E+06	41.204	4.716

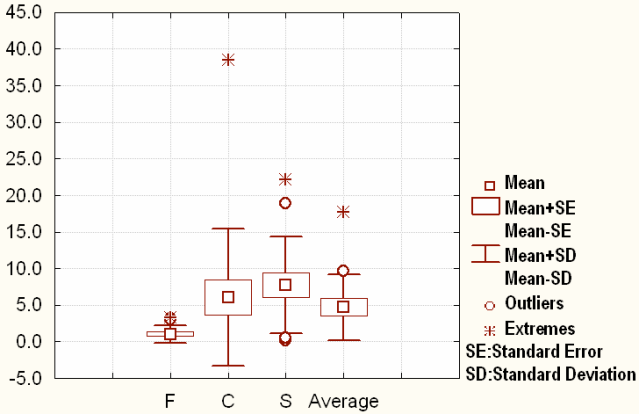

Figure 2. Box Plots of the MRE Values for C, S, F and Simple Average

It is noticeable that F is apparently the best component for this company's environment. Its SD, MSRE, MSE, and MMRE are all much smaller than those of C and S. However, it should also be noted in Appendix **1** that C performs the best on Project No.15 and S is the best on Project No.14 even though F is the best estimator on other 13 projects. This implies that even the much worse components might provide information that the apparently best one doesn't have. Combining them might improve the estimating accuracy. As for simple average, although it performs much better than C and S, it is also inferior to F, for it doesn't consider different contributions from each component.

We work out the three components' zero-order correlations of relative error in Table 3. The correlation matrix shows that the three methods have low correlation. This indicates that when one method predicts poorly, one or both of the others tends to perform significant better, so they have the potential to benefit from combining.

Table 3. RE Correlation Coefficients

	RE_C	RE_S	RE_F
RE_C	1.000	-	-
RE_S	0.258	1.000	-
RE_F	0.393	0.329	1.000

5.2 OLC Modeling

Granger and Ramanathan[20] first introduce the OLC method, Hashem and Schmeiser[13] extend the idea of OLCs and discuss related issues about how to improve the generalization ability

(predictive power) of the combined model by reducing collinearity. The essence of OLC is the multiple regression analysis using Ordinary Least Square (OLS) [36] estimates of components as independent variables, and actual effort as attributive variable.

The OLC model is $Y = \alpha_0 + \sum_{j=1}^{p} \alpha_j y_j$ (1)

Assume that there are *p* component methods. OLC's estimating result *Y* is the weighted average of *p* component methods' estimates. Regression coefficients $\alpha_1,\ldots\alpha_p$ denote optimal combination weights of the *p* component methods. α_0 is the constant term which allows for correction of any bias in $y_j s, j = 1,\ldots p.$ The constant term α_0 and combination weights $\alpha_1,\ldots\alpha_p$ are determined by OLS method to minimize the mean squared error (MSE)[36].

There are four extended cases of OLC models based on eqn(1), the variations among the four cases are in the inclusion (or exclusion) of the constant term α_0 , and/or constraining the combination weights $\alpha_1,\ldots\alpha_p$ to sum to one as shown in Table 4[13]. For short, we use Ⅰ ,Ⅱ ,Ⅲ, Ⅳ to denote each case of OLCs respectively in the paper. The contents in brackets denote constraints for each case.

Table 4. Four Cases of OLCs

Ⅰ	unconstrained OLC with a constant term (no constraints)
Ⅱ	constrained OLC with a constant term ($\alpha_1 + \alpha_2 + \ldots + \alpha_p = 1$)
Ⅲ	unconstrained OLC without a constant term ($\alpha_0 = 0$)
Ⅳ	constrained OLC without a constant term ($\alpha_1 + \alpha_2 + \ldots + \alpha_p = 1$、$\alpha_0 = 0$)

A question might be directly asked is that which one should be adopted and by what measures. On the one side, the unconstrained OLC with a constant term, theoretically yields the smallest MSE[13] (by mathematic formula deduction or in-sample performance evaluation) compared to the best component method, the simple average, and to the other three OLCs. Granger and Ramanathan also recommend combining forecasts using unconstrained OLC[20]. However, the more important performance measure is the accuracy measured on a separate data set, and we evaluate performance measures (MSE, MSRE, MMRE, and SD) by cross-validation to reflect the predictive power. On the other side, the inclusion of the constant term helps to correct for possible biases in the component methods, however, the constant may sometimes be involved in collinearity (see details in section 5.3) and do harm to the predictive power of OLC, in this case, we should drop the constant. Besides, constraining the combination-weights to sum to one may sometimes be used in improving the predictive power of OLC[13]. These indicate that any one of the four cases of OLCs might be more robust than others due to different data sets. A practical view to choose the one with the highest predictive power is modeling all the four cases of OLCs respectively, performing cross-validation to compare performance measures, and choosing the one that has the minimal MSE. This can be shown in *Processing Rectangle 3* in Figure 1.

In our experimental study, we evaluate measures on both the in-sample and out-of-sample data set. For evaluation on in-sample data set, we want to experimentally prove the theoretical conclusion that the unconstrained OLC with a constant term, theoretically yields the smallest MSE[13]. For evaluation on out-of-sample data set, we want to choose the one with highest predictive power for further estimating.

The in-sample accuracy of the four cases' OLCs is shown in Table 5. We can see that all the four cases of OLCs have lower in-sample MSE than the best component method F. The unconstrained OLC with a constant term (the Ⅰ OLC) has superior accuracy, in the sense of MSE, on the in-sample performance. These results are in accordance with the theory that the Ⅰ OLC yields the smallest MSE.

Table 5. In-Sample MSE Comparison

	α_0	α_1	α_2	α_3	**MSE**
F	0.000	0.000	0.000	1.000	2.91E+04
Ⅰ	6.906	-0.013	0.073	0.293	5.99E+03
Ⅱ	-92.31	-0.062	0.060	1.001	1.30E+04
Ⅲ	0.000	-0.014	0.073	0.316	6.01E+03
Ⅳ	0.000	-0.082	0.054	1.0282	1.86E+04

To identify the one with the highest predictive power, we perform cross-validation to compare MSE. As we have a small size data set of only 15 projects, in order to make full use of the data, we use LOOCV: that is training OLCs on 14 projects and testing the accuracy on the remaining one, repeating this process 15 turns to let each project to be a test sample. The MSE, MSRE and MMRE of the cross-validation are shown in Table 6, the best component F's value is also included to compare.

Table 6. Accuracy Comparison after LOOCV

	MSE	**MSRE**	**MMRE**
F	2.91E+04	2.312	1.027
Ⅰ	3.14E+04	0.684	0.695
Ⅱ	6.87E+04	7.364	1.687
Ⅲ	4.30E+04	2.086	0.926
Ⅳ	8.33E+04	3.935	1.377

As we may see in Table 6, Ⅰ OLC still has the lowest MSE of the four OLCs after cross-validation and this indicates that it is the one with the highest predictive power of the four cases, so we should choose it. We also can see that except MSE, the performance measures MSRE and MMRE of Ⅰ OLC are the smallest. These have shown that OLC has already improved predictive accuracy. However, Ⅰ OLC's MSE is still larger than the best component F. According to the definition of combining estimation methods' predictive power in the end of Part 4, the current OLC doesn't have sound predictive power. A direct

question might be what factors might influence the predictive power of OLC and how to improve it.

5.3 Further Improving Predictive Power

The problem that sometimes affects the estimation of the optimal combination weights, as well as the predictive power of the OLC, is the collinearity among the predictors variables $y_i, j=1,\ldots p$,in the regression model eqn(1). Since the y_i s are the outputs of components that are trained to approximate the same actual value Y, we would expect them to be highly collinear. Sometimes even the constant term is involved in collinearity[13]. Ill effects of collinearity are blamed for undermining the predictive power of OLCs[23, 29, 43]. A common and simple way to deal with collineariy is to drop a component involved in the strongest collinearity.

We adopt two rules of thumb [44] to identify the variables involved in the strongest collinearity. The first rule of thumb is "High R^2 (the multiple coefficient of determination [45])but few significant *t* ratios". As noted, this is the classic symptom of collinearity. If R^2 is high (generally higher than 0.8) the *F* test is significant, but few individual *t* tests of coefficients are significant. The variables whose coefficients are not significant are involved in collinearity. This rule of thumb helps us to detect collinearity and identify all the variables involved in collinearity. The other rule of thumb is "High pair-wise correlations among regressors". If the pair-wise or zero-order correlation coefficient between two regressors is high (generally higher than 0.8) then collinearity is a serious problem. We should also be noted that high zero-order correlations are a sufficient condition for the existence of collinearity. For this reason, high correlation means strong collinearity, largest correlation coefficient of a pair of variables means they are involved in the strongest collineairy. This rule of thumb helps us to find the pair involved in the strongest collinearity. According to these two rules of thumb, we can find the variables involved in the strongest collinearity and drop the one with larger MSE to weaken the bad influence of collinearity. The *Processing Rectangle 4* in Figure 1 denotes this process.

The inclusion of the constant term helps to correct for biases in the component methods but sometimes it is involved in collinearity which might ruin the predictive power of OLC, the simplest way to see whether the constant term does good or not to the predictive power of OLC is to compare the MSE before and after excluding the constant. The *Processing Rectangle 5* denotes this process. Because the inclusion of the constant term benefits the correction of biases, we don't consider the ill influence of the constant term until there is no room to improve accuracy by dropping component method.

It should be noted that Hashem and Schmeiser's OLC algorithms employ BKW diagnostics to determine the existence of collinearity and the variables involved in it[13]. Two rules of thumb that we adopts serve as the same effect and much easier to understand and implement. Furthermore, their algorithm cannot ensure to improve the predictive power of the combining model to the largest extent, as it returns as long as the MSE by cross-validation of OLC is smaller than the best component and the simple average. However, our algorithm aims at maximizing the OLC's predictive power. It doesn't return until there is no room for further improvement as shown in Figure 1.

In our experimental study, current OLC is still not more accurate than F on the sense of MSE. That means that collinearity has ill effects on the predictive power of the OLC, and we need to improve the predictive power by dropping one of the components or the constant term. By regression on 15 projects' estimating data, OLC's model is

Y=	6.906-	0.013*C+	0.073*S+	0.293*F	(2)
t	0.173	-0.475	5.661	1.483	
P-value	0.866	0.644	0.000146	0.166	
R^2=0.907					

We can see that R^2 is as high as 0.907, however, the coefficients of constant, C, F are all not statistically significant (*P-value*<0.05 denotes statistically significance). According to the first rule of thumb, this is the classic symptom of collinearity, and the constant, C, F are maybe involved in collinearity.

In order to find the pair involved in the strongest collinearity, we worked out the zero-order correlation matrix of the three components' estimates as shown in Table 7 .

Table 7.Estimates Correlation Coefficients

	C	S	F
C	1.000	-	-
S	0.789	1.000	-
F	0.809	0.708	1.000

It can be seen that the pair C and F's zero-order correlation 0.809 is the highest. According to the second rule of thumb, they are involved in the strongest collinearity. Besides, C performs much worse (has larger MSE) than F, so we drop C from the combining model, remodel using the OLC and gain MSE by LOOCV to see whether predictive power improved or not. The MSE, MSRE and MMRE of the new model OLC with only S and F (_S+F for short), compared with F and OLC with all three components (_C+S+F for short) are shown in Table 8. We can see that OLC with only S and F's MSE is smaller than that of the apparent best component F, this means that by dropping C, the combining model's predictive power has been improved. Since MSRE and MMRE are strongly related to MSE, the accuracy in terms of MSRE and MMRE are also improved by 18.34% (0.684/0.578-1) and 6.60% (0.695/0.652-1) respectively, compared with before dropping C. These results show the accuracy improvement by dropping C to weaken the ill influence of collinearity.

Table 8. Accuracy Comparison after Dropping C

	MSE	MSRE	MMRE
F	2.91E+04	2.312	1.027
_C+S+F	3.14E+04	0.684	0.695
_S+F	2.48E+04	0.578	0.652

After dropping C, the OLC trained by 15 projects' data is as follows:

Y=	10.173+	0.070*S+	0.241*F	(3)
t	0.267	6.376	1.518	

P-value	0.794	3.53E-05	0.155

R^2=0.905

Considering that the *t-test* of constant term and F's coefficient are still not significant which indicates that they are still involved in collinearity, so this might also ruin the predictive power of OLC. Besides, there are only two methods (S and F) left, according to the OLC algorithm in Figure 1, we try to drop the constant to see if predictive power could be further improved also using LOOCV. The result is shown in Table 9.

Table 9. Accuracy Comparison after Dropping Constant

	MSE	MSRE	MMRE
F	2.91E+04	2.312	1.027
I _C+S+F	3.14E+04	0.684	0.695
I _S+F	2.48E+04	0.578	0.652
III_S+F	1.75E+04	0.565	0.636

It is noticeable that after dropping the constant (III_S+F for short), the predictive power is further improved: the MSE has been reduced from 2.48E+04 to 1.75E+04. The accuracy on the sense of MSRE and MMRE are also improved by 2.30% (0.578/0.565-1) and 2.52% (0.652/0.636-1) respectively for they are strongly related to MSE. Considering that there is no room to improve the predictive power of OLC, we return the following model (trained on the whole 15 company-specific projects' data) as our final OLC. It can be seen that the coefficients of S and F are both significant now which indicates that the collinearity has been greatly weakened after dropping the constant term.

Y=	0.0697*S+	0.2700*F	(4)
t	6.690	2.432	
P-value	1.493E-05	0.030191	

R^2=0.945

In the process of improving OLC's predictive power, we start with the initial OLC (I _C+S+F), and then weaken collinearity by dropping C (I _S+F) and the constant (III_S+F) in succession to maximize OLC's predictive power. Table 9's last three rows show the decreasing trend of both MSE and strongly relevant MSRE and MMRE during the further improvement process.

5.4 Returning the Final Estimating Model

At last, if there is no room to improve the predictive power of OLC, we should identify whether or not the predictive power of the final OLC is higher than that of the best component method and the simple average, which is measured on MSE by cross-validation. If so, we should return the OLC as the effective combining model for the company-specific environment, otherwise it means that we cannot find an OLC estimating more accurate than that of the best component or the simple average, and in this case, we should return the best component or the simple average, depending on which has lower MSE. This can be shown in *Processing Rectangle 6* in Figure 1.

In our experimental study, compared with the apparently best component F (already superior to simple average), the final OLC's accuracy in terms of MSE, and strongly related MSRE, MMRE are improved by 66.29% (2.91E+04/1.75E+04-1), 3.09 (2.312/0.565-1) times and 61.48% (1.027/0.636-1) respectively. That indicates the OLC's (III_S+F) effectiveness and we should return it as our final estimating model. At the same time, we resolve the SD of OLC 0.778, so its consistency on the sense of SD is improved by 96.91% (1.533/0.778-1). Figure 3 can more vividly show OLC's the accuracy and consistency improvement from the best component F.

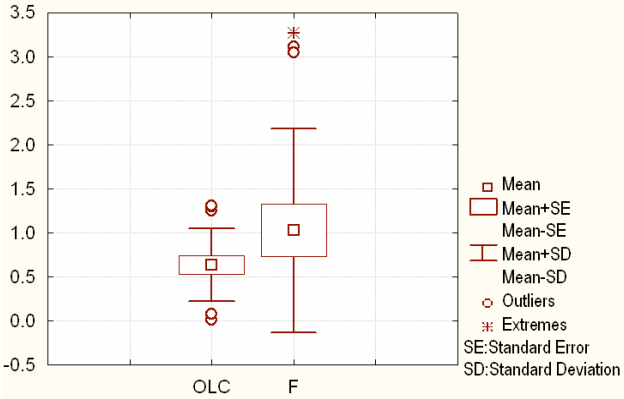

Figure 3. Box Plots of the MRE Values for OLC and F

6. LESSONS LEARNED FROM THE EXPERIMENT

From this experimental study based on 15 company-specific projects' data, it indicates that OLC greatly improves the accuracy and consistency of software effort estimation by integrating the information provided by components even the component (e.g. S,C) performs much worse than the apparent best component (e.g. F). As Sherif Hashem discussed in [13], the improvement in combining accuracy depends on the following factors:

1). "Degree of redundancy in the information obtained from the components." Combining components with different topologies, different learning algorithms, and at least different training data can increase the benefits of combining. In our experiment, we use error correlation to quantificationally measure "disagreement" between components. The experiment shows low correlation (0.258, 0.393, 0.329. see Table 3) between the three components, we would expect more benefit from combining if the error correlation coefficients were lower or negative[9].

2). "Superiority of the best component method." If one method performs much superior to the rest, while the other methods have no additional knowledge to contribute, the OLC will tend to favor using the best component by itself. In our experiment, F performs much better than C and S on most of the 15 projects, so F contributes most to the combining accuracy, and its weight is much heavier than S. This shows that the OLC tends to favor using F and its predictive power is the closest to F.

3). "Adequacy of the combination data." Small quantity of data might cause severe ill effects of collinearity[44]. In that case, the OLC algorithms can only save the best component method by dropping other components. However, dropping one of the collinear variables might at the same time drop some useful information for estimation or commit a "specification bias"[44] and this might reduce the accuracy of OLC. This is true in our experiment, 15 projects' data is so small that it causes severe collinearity. Although C has potential contributions to combining (e.g. C performs best on Project 15), it is offset by the harm of collinearity, so C is not

included in the final OLC, and the final OLC might lose the information provided by C. We believe that as the size increase, ill effects of collinearity can be reduced and some components might be saved to provide more useful information for combining. Moreover, in addition to increasing the size, other ways to reduce collinearity might be using polynomial regressions or factor analysis, ridge regression etc[44], we could use these techniques to improve the OLC algorithm in the future.

4). "Outliers at different noise levels". At the level of the OLC algorithm, OLC adopts OLS to determine combining weights and the constant term (if have) by minimizing MSE. Directly, the larger the MSE of one method, the smaller the weight it is assigned. But MSE are often blamed for its high sensitiveness to outliers as referred in section 4. Data with a large number of outliers at high noise levels, as we may say, poor data quality, can directly lead to bad weights allocation, which in turn ruins the predictive power of OLC, and small quantity of sample could aggravate this problem. However, dropping the worse one (with higher MSE) of the pair involved in the strongest collinearity can at the same time dropping outliers and in turn reduce outliers' bad effects, so by means of this, OLC's predictive power to tolerate with outliers at high noise levels is improved and OLC is less sensitive to outliers. This could explain why MSE values by cross validation of four cases of OLC with three components C, S, F are all larger than that of the best F (see Table 6). But after dropping C, at the same time, dropping the worst outlier (its RE on Project 9 is as high as 38.462, see Appendix 1), the predictive power of OLC is much stronger than that of F. Because of MSE's high sensitiveness to outliers, another way to reduce OLC sensitiveness to outliers might be employing other algorithms instead of OLS by minimizing not sensitive criterion e.g. MMRE.

In sum, in our experimental study, despite of the small size (only 15 projects' data) and not very good data quality (see Figure 2), OLC has shown significant improvement on predictive accuracy and consistency. This might show, at least in our experimental study, OLC's ability to tolerate with small size data at high noise levels. In addition, the OLC has exhibited improved predictive power both for small and large sample sizes and at different noise levels using other fields' data or by simulation in some other researches[13, 30]. As the quantity and quality of data available for combining increase, we believe that OLC can be clearly superior to component methods, which can also be expected based on the theoretical results of [13].

7. DISCUSSION OF POSSIBLE THREATS OF VALIDITY

It is noticeable that F.Kemerer's data were collected before 1987 and the programs were written in Cobol, Bliss and Natural. Meanwhile, this data set shows that C and S are very bad, MMRE of them are 6.079 and 7.719 respectively, and cannot be compared to the results of up-to-date methods, e.g. Bayesian-calibrated COCOMOII .2000 model has produces estimates that are within 30 percent of the actuals 66 percent of the time[2]. However, significant accuracy improvement in component method will result in further accuracy improvement in combining methods. This insight was quantified in [40]by a relation stating that "the generalization error of a weighted combination of predictors in combining is equal to the average error of the individual predictors minus the ''disagreement'' among them". That means the better performance individual predictors have, the more "disagreement" among them, the better the combining performance will be. At the same time, our focus in this paper is not to evaluate component methods, but to experimentally prove that combining methods can improve predictive power.

Another threat of validity might be that only 15 projects' data might be statistically so small a sample to show OLC method's effectiveness. Besides, small quantity of data might limit our choice of proper validation methods. In our experiment, due to the small data set, we employ LOOCV to measure OLC's predictive power. However, there is a disadvantage to LOOCV: "By its very nature, it cannot be stratified-worse than that, it guarantees a nonstratified sample. Stratification involves getting the correct proportion of examples in each class into the test set, and this is impossible when the test set contains only a single example."[38]. This feature of LOOCV might cause the bias of performance evaluation. As the data set increases, we could use k-fold cross-validation even other evaluating methods to avoid this problem[37, 38].

Meanwhile, we tried to find more recent real life software effort estimating data to validate OLC effectiveness. However, there are no other proper public data (the estimation results from more than one techniques) for us to do so. To better refine our method and validate its effectiveness, we plan to apply this method to broader communities.

As for the significance of the experimental result, we can see that OLC greatly improve its predictive power on the sense of MSE, MSRE, MMRE and SD. To determine statistically significant differences between OLC and F, commonly used statistic tests are parametric test (paired t test) or nonparametric test (Wilcoxon matched pair test), however, they only make sense when the pairs are independent[46]. But the OLC's predictive performance highly relies on components, especially the best, and we could expect that the performances of OLC and F in our experiment are highly correlated. So the two statistic tests might not be proper for us to adopt. Meanwhile, more intuitively, OLC's predictive power is based on components. At the most optimistic situation, if each component contributes the same to combining, as the number of components increases, the predictive power of the combining might be significantly better than each component. However, at the most pessimistic situation, if the best component contains all information while others contribute no additional knowledge, it returns the best as the final estimating method (see section 5.4). This makes no sense for combining, let alone its significant improvement from the best component. So for combining estimation methods, it cannot always ensure significant improvement from the best component and therefore it might be not proper to require their results should be intuitively or statistically significantly better than the best component due to their accuracy is highly dependent on components.

Furthermore, for the usability of the OLC method, there are a number of other factors that should be considered in the selection of an estimation technique besides the accuracy factor, and it is likely that trade-offs will need to be made in the process[9]. For example, OLC employs more than one techniques or tools which will increase the estimating cost, including the cost of purchasing business effort estimation tools, the cost of training estimators to use these tools, the cost of collecting company-specific estimating data using each tool and the cost of mastering OLC estimation. If the organization has a leading estimating technique and its estimating performance has already satisfy its organizational goal, improving accuracy using OLC with such amount of cost seems not cost-effective. However,

for the moment, our main focus in this paper is on optimizing the accuracy of our estimation. In other words, we wish to produce estimates that are as close as possible to the actual values, irrespective of the other factors that may be important in the wider organizational setting. In the long run, we plan to develop an automatic OLC tool that might integrate the most popular, mature and effective software effort estimating techniques, and this might be more cost-effective for estimators to use.

8. CONCLUSION & FUTURE WORK

The OLC method is one of the most typical and effective linear combining method and proves to be able to significantly improve the predictive power in our experiment and some other field's research. Combining estimates derived from different techniques or tools and draw from different sources of information should become part of the mainstream of estimating practice in software effort to improve estimating accuracy. Combining estimates is especially useful when you are uncertain about the situation, uncertain about which method is most accurate, and when you want to avoid large errors[14].

Recently there has been increased attention given to providing measures of uncertainty associated with forecasts. Measures of uncertainty surrounding a "central tendency" (the point forecast) can enhance the usefulness of the forecast[27]. Our next research work will include providing an OLC estimate of the probability distribution of its possible values.

Moreover, besides the OLC Approach, there are many other combining strategies including linear combination and nonlinear combination as mentioned in the part of related work, however there are still no unique criteria on the use of combination methods[16]. The choice mainly relies on the features of the particular application conditions and environment. Using the same combination approach for an estimating task may generate good results, but it may be bad for others. In addition, empirical experiments have not yet been able to find an optimal method for selection of the combinatorial strategies. More theoretical development and experiments are needed to explore this field. So it is might be a more meaningful and challenging work for us to explore software effort combining estimation in the future. In the USC-ISCAS joint lab, we plan to apply combining estimation methods through USC-CSSE affiliate programs to collect more proper data and implement more empirical studies.

9. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China under grant Nos. 60573082 , 90718042; the National Hi-Tech Research and Development Plan of China under Grant No. 2006AA01Z182 , 2007AA010303; the National Key Technologies R&D Program under Grant No. 2007CB310802. The author wishes to thank Stephen G.MacDonell and Martin J.Shepperd's initial work for it inspires the idea of combining estimating. Also great thanks to the reviewers of ICSE for their precious suggestion and comments. Additional thanks to Fengdi Shu, Juan Li and Da yang for their assistance with this paper.

10. REFERENCES

[1]. Boehm, B., Software Engineering Economics. 1981: Prentice-Hall.

[2]. Boehm, B., et al., Software Cost Estimation with COCOMO II Har/Cdr ed. 2000: Prentice Hall PTR.

[3]. C.Briand, L., K.E. Emam, and F. Bomarius, COBRA: A Hybrid Method for Software Cost Estimation, Benchmarking, and Risk Assessment, in International Conference on Software Engineering. 1998. p. 390-399.

[4]. Boehm, B. and C. Abts, Software Development Cost Estimation Approaches-A Survey. Annals of Software Engineering, 2000. **10**(1-4): p. 177-205.

[5]. C.Briand, L., Resource Estimation in Software Engineering, in International Software Engineering Research Network. 2000.

[6]. Jorgensen, M. and M. Shepperd, A System Review of Software Development Cost Estimation Studies. IEEE Transaction on Software Engineering, 2007. **33**(1): p. 33-53.

[7]. Briand, L., T. Langley, and I. Wieczorek, Using the European space agency data set: a replicated assessment and comparison of common software cost modeling techniques, in 22nd IEEE International Conference on Software Engineering. 2000.

[8]. Gray, A.R. and S.G. MacDonell, Software metrics data analysis-exploring the relative performance of some commonly used modeling techniques. Empirical Software Engineering, 1999(4): p. 297-316.

[9]. G.MacDonell, S. and M. J.Shepperd, Combining techniques to optimize effort predictions in software project management. The Journal of System and Software, 2003. **66**: p. 91-98.

[10].Jeffery, R., M. Ruhe, and I. Wieczorek, Using public domain metrics to estimate software development effort, in 7th IEEE International Metrics Symposium. 2001.

[11].Idri, A., A. Abran, and T.M. Khoshgoftaar, Computational Intelligence in Empirical Software Engineering http://www.lrgl.uqam.ca/publications/pdf/807.pdf, in ENSIAS. 2003.

[12].Kitchenham, B., Software Metrics: Measurement for Software Process Improvement 1996: NCC Blackwell

[13].Hashem, S., Optimal Linear Combinations of Neural Networks. Neural Network, 1997. **10**(4): p. 599-614.

[14].Armstrong, J.S., ed. Principles of Forecasting: A Handbook for Researchers and Practitioners. Chapter 13 Combining Forecasts. 1 ed. 2001, Springer.

[15].J.M.Bates and C.W.J.Granger, The Combination of Forecasts. Operational Research Quarterly, 1969. **20**(4): p. 451-468.

[16].Clemen, R.T., Combining forecasts: A review and annotated bibliography. International Journal of Forecasting, 1989. **5**: p. 559-583.

[17].Menezes, L.M.d., Review of Guidelines for the Use of Combined Forecasts. European Journal of Operation Research, 2000. **120**(1): p. 190-204

[18].E.Rapach, D. and J. K.Strauss, Forecasting U.S.Employment Growth Using Forecasting Combining Methods. Journal of Forecasting 2008.27(1):p. 75-93

[19].Stock, J.H. and M.W.Watson, Forecasting Inflation. Journal of Monetary Economics, 1999. **44**: p. 293-335.

[20].Granger, C.W.J. and R. Ramanathan, Improved Methods of Combining Forecasts. Journal of Forecasting, 1984. **3**(2): p. 197-204.

[21].Diebold, F.X. and P.Pauly, Structural Change and the Combination of Forecasts. Journal of Forecasting, 1987. **6**: p. 21-40.

[22].Stock, J.H. and M.W.Watson, Combination Forecasts of Output Growth in a Seven-Country Data Set. Journal of Forecasting, 2004. **23**: p. 405-430.

[23].Clemen, R.T. and R.L. Winkler, Combining Economic Forecasts. Journal of Business and Economic Statistics, 1986. **4**: p. 39-46.
[24].Aiolfi, M. and A. Timmermann, Persistence in Forecasting Performance and Conditional Combination Strategies. Journal of Econometrics, 2004.135(1-2): p.31-53
[25].Swanson, N.R. and T.Zeng, Choosing Among Competing Econometric Forecasts: Regression-Based Forecast Combination Using Model Selection. Journal of Forecasting, 2001. **20**: p. 425-440.
[26].Chan, Y.L., J.H.Stock, and M.W.Watson, A Dynamic Factor Model Framework for Forecast Combination. Spanish Economic Review, 1999. **1**: p. 21-121.
[27].Garratt, A., et al., Forecast Uncertainties in Macroeconomic Modeling: An Application to the U.K. Economy. Journal of the American Statistical Association, 2003. **98**: p. 829-838.
[28].Yang, Y., Combining Forecasting Procedures: Some Theoretical Results. Econometric Theory, 2004. **20**: p. 176-222.
[29].D.W.Bunn, Forecasting with more than one model. Journal of Forecasting, 1989. **8**: p. 161-166.
[30].Hashem, S., Optimal Linear Combinations of Neural Networks. In Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy, in School of Industrial Engineering. 1993, Purdue.
[31].Chulani, S. and B. Boehm, Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Transaction on Software Engineering, 1999. **25**(4): p. 573-583.
[32].Shepperd, M., Bournemouth University Automated Project Cost Estimation: Using Analogies The ANGEL Project.http://dec.bournemouth.ac.uk/ESERG/ANGEL. 1996.
[33].Mittas, N., M. Athanasiades, and L. Angelis, Improving analogy-based software cost estimation by a resampling method. Information and Software Technology, 2008. **50**(3): p. 221-230.
[34].Jorgensen, M., Practical Guidelines for Expert-Judgment-Based Software Effort Estimation. IEEE Software 2005. **22**(3): p. 57-63.
[35].Kitchenham, B., An empirical study of maintenance and development estimation accuracy. Journal of Systems and Software, 2002. **64**(1): p. 57-77.
[36].A.Rice, J., Mathematical Statistics and Data Analysis, Second Edition. Chapter 14 Linear Least Squares. 2003: Duxbury Press.
[37].Armstrong, J.S., ed. Principles of Forecasting: A Handbook for Researchers and Practitioners. Chapter 14 Evaluation Methods. 1 ed. 2001, Springer.
[38].H.Witten, I. and E. Frank, DATA MINING: Practical Machine Learning Tools and Techniques. Chapter 5 Credibility: Evaluating What's Been Learned. Second ed. 2005: Morgan Kaufmann.
[39].AJC, S., On Combining Artificial Neural Nets. Connection Science, 1996. **8**: p. 299-314.
[40].Krogh, A. and J. Vedelsby, Statistical mechanics of ensemble learning. Physical Review E, 1997. **55**(1): p. 811-825.
[41].A.Rice, J., Mathematical Statistics and Data Analysis, Second Edition. Chapter 4.3 Covariance and Correlation. Second ed. 2003: Duxbury Press.
[42].F.Kemerer, C., An Empirical Validation of Software Cost Estimation Models. Communications of ACM, 1987. **30**: p. 416-429.
[43].Guerard, J.B.J. and R.T. Clemen, Colinearity and the use of latent root regression for combining GDP forecasts. Journal of Forecasting, 1989. **8**: p. 231-238.
[44].Gujarati, Basic Econometrics. Chapter 10. . 4 ed. 2004: The McGraw-Hill Companies.
[45].Gujarati, Basic Econometrics. Chapter 7. 4 ed. 2004: The McGraw-Hill Companies.
[46].Motulsky, H., Analyzing Data with GraphPad Prism: A companion to GraphPad Prism version 3 http://www.graphpad.com. 1999: GraphPad Software, Inc.

Appendix 1.Estimates and Relative Errors of C, S, F, Simple Average and OLC

NO.	Actual MM	C	RE_C	S	RE_S	F	RE_F	Average	RE_Average	OLC	RE_OLC
1	287.00	932.96	2.251	3857.80	12.442	344.30	0.200	1711.69	4.964	368.62	0.284
2	82.50	151.19	0.833	100.10	0.213	92.13	0.117	114.47	0.388	31.21	-0.622
3	1107.31	5818.75	4.255	11982.00	9.821	731.43	-0.339	6177.39	4.579	717.71	-0.352
4	86.90	566.50	5.519	2017.20	22.213	192.03	1.210	925.24	9.647	195.06	1.245
5	336.30	1316.04	2.913	3382.00	9.056	387.11	0.151	1695.05	4.040	340.68	0.013
6	84.00	312.24	2.717	262.50	2.125	61.58	-0.267	212.11	1.525	34.76	-0.586
7	23.20	234.51	9.108	106.30	3.582	-52.60	-3.267	96.07	3.141	-6.99	-1.301
8	130.30	1206.17	8.257	1224.60	8.398	264.68	1.031	898.48	5.895	158.49	0.216
9	116.00	4577.62	38.462	1454.10	11.535	477.81	3.119	2169.84	17.706	267.49	1.306
10	72.00	181.36	1.519	235.70	2.274	-2.83	-1.039	138.08	0.918	15.61	-0.783
11	258.70	1575.68	5.091	1623.00	5.274	484.24	0.872	1227.64	3.745	239.22	-0.075
12	230.70	584.37	1.533	513.30	1.225	192.21	-0.167	429.96	0.864	81.40	-0.647
13	157.00	1124.36	6.162	3119.80	18.871	157.36	0.002	1467.17	8.345	266.80	0.699
14	246.90	663.84	1.689	380.30	0.540	390.63	0.582	478.26	0.937	98.08	-0.603
15	69.90	130.72	0.870	643.80	8.210	282.91	3.047	352.48	4.043	126.79	0.814