A Large Scale Empirical Evaluation of the Accuracy of Function Points Estimation Methods

Luigi Lavazza
Dipartimento di Scienze teoriche e Applicate
Università degli Studi dell'Insubria
21100 Varese, Italy
Email: luigi.lavazza@uninsubria.it

Geng Liu
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
Email: liugeng@hdu.edu.cn

Abstract—**Functional size measures of software—especially Function Points—are widely used, because they provide an objective quantification of software size in the early stages of development, as soon as functional requirements have been analyzed and documented. Unfortunately, in some conditions, performing the standard Function Point Analysis process may be too long and expensive. Moreover, functional measures could be needed before functional requirements have been elicited completely and at the required detail level. To solve this problem, many methods have been invented and are being used to estimate the functional size based on incomplete or not fully detailed requirements. Using these methods involves a trade-off between ease and timeliness on one side and accuracy on the other side. In fact, estimates are always affected by some error; knowing the magnitude of estimation errors that characterize the estimates provided by a given method is of great importance to practitioners. This paper reports the results of an empirical study devoted to evaluate the accuracy of estimates provided by Function Points estimation methods. The results of the study show that some of the evaluated methods—including the Early & Quick Function Points, the ISBSG average and the NESMA estimated method)—provide estimates that are accurate enough for practical usage, while some other methods appear quite inaccurate.**

Keywords–Function Points; IFPUG; Function point Analysis; Functional Size Measurement; Functional Size Estimation; NESMA Estimated; NESMA Indicative; Early Size Estimation.

I. INTRODUCTION

This paper extends the results provided in a previous paper [1], in which we started estimating the accuracy methods for estimating functional size measures.

The availability of accurate functional size measures can help software companies plan, monitor, estimate development costs, and control software development processes. So, the availability of accurate functional size measures may provide software companies with competitive advantages over other companies.

Among the functional size measurement methods that have been proposed, Function Point Analysis (FPA) [2] is by far the most popular. The International Function Points User Group (IFPUG) took charge of maintaining FPA and publishes the official Function Point counting manual [3].

In some conditions, performing the standard FPA process may be too long and expensive. Moreover, standard FPA can be applied only after the completion of the software requirements elicitation stage, while functional measures could be needed earlier, i.e., before functional requirements have been elicited completely and at the required detail level.

Therefore, many methods were invented and used to provide *estimates* of functional size measures based on less or coarser grained information than required by standard FPA.

Among the most widely known and used methods are the NESMA methods [4], Early&Quick Function Points (EQFP) [5], [6], the Tichenor ILF Model [7], the simplified FP (sFP) approach [8], the ISBSG distribution model and the ISBSG average weights model, the latter two methods being based on the analysis of the ISBSG dataset [9]. Actually, the NESMA methods were adopted by IFPUG as the preferred methods for early estimation of Function Point size [10] (IFPUG renamed the NESMA Estimated method 'High Level FPA Method' and the NESMA Indicative method as 'Indicative FPA Method'; nonetheless, in this paper we use the original NESMA names).

Inevitably, all the early functional size estimation methods involve some estimation error. Hence, project managers need to know—at least approximately—the magnitude of the potential error that affects size estimates.

Not surprisingly, several researchers and practitioners evaluated the accuracy of the proposed functional size estimation methods (as described in Section VII). However, most evaluations were based on academic software projects or small datasets, hence most evaluations cannot be considered very reliable, and they are hardly generalizable. In order to assess the actual value of the FP estimation methods for industry, it is necessary to perform an experimental evaluation based on a large dataset collecting measures from industrial settings.

In this paper, we present the experimental evaluation of the accuracy of several estimation methods, based on a dataset that includes data from 479 software development projects in the finance and banking domain. The size of the dataset and the real-life nature of the data make the analysis presented here the most exhaustive and reliable published evaluation of Function Point estimation methods.

This paper extends the results provided in a previous paper [1], in which we estimated only the accuracy of NESMA methods. This paper also replicates a previous empirical study [11], in which we evaluated the accuracy of the same estimation methods studied in this paper. In [11] we were able to study only 18 project data, while here we analyze a dataset collecting measures from 479 software projects. The results provided here are relevant in the following respects:

- We were able to confirm that the 'NESMA Estimated' method is among the most accurate proposed methods.
- Some methods appear definitely not accurate; more precisely, their accuracy appears too low, even for usage in the earliest phases of software development.
- Some methods appear even more accurate than the NESMA method, hence project managers can consider using these methods, instead of the NESMA Estimated, even though the latter was chosen as the 'official' estimation methods by IFPUG.

The study described in this paper is of practical relevance because of the following reasons:

- One of the main hindrances perceived by software developers when applying FPA is that it is relatively difficult and expensive, and requires specifically trained and certified measurers. To this end, estimation methods were proposed not only to make FPA applicable before functional requirement specifications are complete and detailed, but also as an easier—hence quicker and cheaper—alternative to FPA, to be applied even when requirements have been fully documented.
- The application of standard functional size measurement does not fit in agile development processes, which are being adopted by software development organizations to a continuously increasing degree. On the contrary, estimation methods are simple enough to fit into agile development processes. Consider for instance that the data required to estimate Function Points using methods like the Early & Quick Function Points or the NESMA methods can be easily derived from the 'stories' that are often used to describe requirements in agile contexts.
- Anticipating the moment when (approximated) functional measures become available means that development effort estimations can be anticipated as well. This is often of great importance, e.g., in bidding processes.

The rest of the paper is organized as follows. Section II briefly describes FPA. Section III gives a concise introduction to the FP estimation measurement methods studied in this paper. Section IV describes the empirical study. Section V proposes some considerations on the outcomes of the empirical study. Section VI discusses the threats to the validity of the study. Section VII illustrates the related work. Finally, Section VIII draws some conclusions and outlines future work.

II. Function Point Analysis

This section provides a concise introduction to FPA. Readers are referred to the official documentation [3] for further details.

Function Point Analysis was originally introduced by Albrecht to measure the size of data-processing systems from the end-user's point of view, with the goal of estimating the development effort [2].

The initial interest sparked by FPA, along with the recognition of the need for maintaining FPA counting practices, led to founding the IFPUG (International Function Points User Group). The IFPUG (http://www.ifpug.org/) maintains the counting practices manual [3], provides guidelines and examples, and oversees the standardization of the measurement method.

The IFPUG method is an ISO standard [12] in its "unadjusted" version. The adjustment factor originally proposed by Albrecht and endorsed by IFPUG is meant to obtain measures more apt for effort estimation, by accounting for factors not dealing with functional requirements, namely with product and process features that do not belong to the notion of functional size. As such, the adjustment was not accepted by ISO; so, throughout the paper we refer exclusively to unadjusted FP (UFP), even when we omit adjective "unadjusted."

Albrecht's basic idea—which is still at the basis of the IFPUG method—is that the "amount of functionality" released to the user can be evaluated by taking into account 1) the data used by the application to provide the required functions, and 2) the transactions (i.e., operations that involve data crossing the boundaries of the application) through which the functionality is delivered to the user. Both data and transactions are evaluated at the conceptual level, i.e., they represent data and operations that are relevant to the user. Therefore, IFPUG Function Points are counted on the basis of the user requirements specification. The boundary indicates the border between the application being measured and the external applications and user domain.

FURs are modeled as a set of base functional components (BFCs), which are the measurable elements of FURs: each of the identified BFCs is measured, and the size of the whole application is obtained as the sum of the sizes of BFCs.

The IFPUG model of a software application to be measured is shown in Figure 1. IFPUG BFCs are data functions (also known as logical files), which are classified into internal logical files (ILF) and external interface files (EIF), and elementary processes (EP)—also known as transaction functions—which are classified into external inputs (EI), external outputs (EO), and external inquiries (EQ), according to the activities carried out within the process and its main intent.

So, the functional size of a given application, expressed in unadjusted Function Points, $Size_{UFP}$, is given by the sum

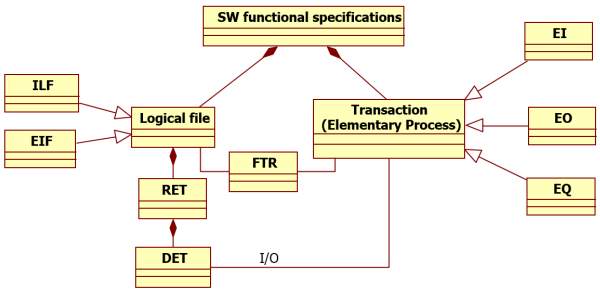

Figure 1. The IFPUG model of software.

of the sizes of the different types of functions, as shown in equation (1).

$$Size_{UFP} = \sum_{f \in ILFs} Size_{ILF}(f) + \sum_{f \in EIFs} Size_{EIF}(f) + \quad (1)$$
$$\sum_{f \in EIs} Size_{EI}(f) + \sum_{f \in EOs} Size_{EO}(f) + \sum_{f \in EQs} Size_{EQ}(f)$$

In equation (1), *ILFs* is the set of all data functions of type *ILF*, *EIs* is the set of all transactional functions of type *EI*, etc. Also, $Size_X(f)$ is the weight of function f, which depends on its complexity, and its type ($X \in \{ILF, EIF, EI, EO, EQ\}$, as described in Table I.

TABLE I. FPA WEIGHT TABLE.

	Complexity		
Function type	Low	Average	High
ILF	7	10	15
EIF	5	7	10
EI	3	4	6
EO	4	5	7
EQ	3	4	6

The complexity of a data function (ILF or EIF) depends on the RETs (Record Element Types), which indicate how many types of information (e.g., sub-classes, in object-oriented terms) can be contained in the given logical data file, and DETs (Data Element Types), which indicate how many types of elementary information (e.g., attributes, in object-oriented terms) can be contained in the given logical data file.

The complexity of a transaction depends on the number of file types references (FTRs)—i.e., the number of types of logical data files used while performing the required operation—and the number of DETs—i.e., the number of types of elementary data—that the considered transaction sends and receives across the boundaries of the application.

Details concerning the determination of complexity can be found in the official documentation [3].

The core of FPA involves three main activities:

1) Identifying data and transaction functions.
2) Classifying data functions as ILF or EIF, and transactions as EI, EO or EQ.
3) Determining the complexity of each data or transaction function.

The first two of these activities can be carried out even if the FUR have not yet been fully detailed. On the contrary, the last activity requires that all details are available, so that such details can be analyzed, to get the number of RET or FTR and DET involved in every function. In the next section, several FP estimation methods are presented. All of these method do not require activity 3 above, thus allowing for size estimation when FUR are not fully detailed. In addition, since activity 3 is the most effort- and time-consuming, the presented estimation methods are relatively fast and cheap, with respect to full-fledged FPA.

III. FUNCTION POINTS ESTIMATION METHODS

In this section we briefly present the FP estimation methods mentioned in the introduction.

A. The NESMA methods

The NESMA methods were proposed to get an estimate of the functional size of a given application without analyzing data and transactions in detail [4].

There are two NESMA estimation methods: the NESMA Indicative method and the NESMA Estimated method.

The former estimates size (*EstSize*) based on the number of ILF (*#ILF*) and the number of EIF (*#EIF*), as follows:

$$EstSize = \#ILF \times W_{ILF} + \#EIF \times W_{EIF}$$

where W_{ILF} is 25 or 35 and W_{EIF} is 10 or 15, depending on ILF and EIF being identified based on a data model in third normal form or not, respectively.

The process of applying the NESMA indicative method involves only identifying logic data and classifying them as ILF or EIF. Accordingly, it requires less time and effort than standard FPA. However, the NESMA Indicative method is quite rough in its computation: the official NESMA counting manual specifies that errors in functional size with this approach can be up to 50%.

The NESMA Estimated method requires the identification and classification of all data and transaction functions, but does not require the assessment of the complexity of each function: Data Functions (ILF and EIF) are assumed to be of low complexity, transactions (EI, EQ and EO) are assumed to be of average complexity. Hence, estimated size is computed as follows:

EstSize = 7 #ILF + 5 #EIF + 4 #EI + 5 #EO + 4 #EQ

IFPUG adopted the NESMA methods as the official early function point analysis methods [10].

B. The Early & Quick Function Points method

One of the most well-known approach for simplifying the process of FP counting is the Early & Quick Function Points (EQFP) method [5], [6]. EQFP descends from the consideration that estimates are sometimes needed before requirements analysis is completed, when the information on the software to be measured is incomplete or not sufficiently detailed. Since several details for performing a correct measurement following the rules of the FP manual [3] are not used in EQFP, the result is a less accurate measure. The trade-off between reduced measurement time and costs is also a reason for adopting the EQFP method even when full specifications are available, but there is the need for completing the measurement in a short time, or at a lower cost. An advantage of the method is that different parts of the system can be measured at different detail levels: for instance, a part of the system can be measured following the IFPUG manual rules [3], [12], while other parts can be measured on the basis of coarser-grained information, possibly considering analogy with previously measured software parts. The EQFP method is based on the classification of the processes and data of an application according to a hierarchy (see Figure 2 [6]).

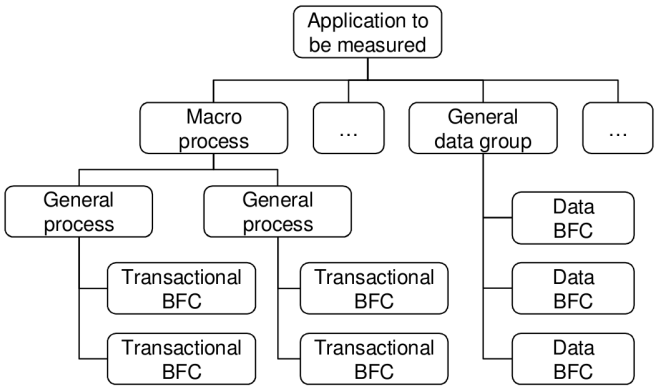

Figure 2. Functional hierarchy in the Early & Quick FP technique.

Transactional BFC and Data BFC correspond to IFPUG's elementary processes and LogicData, while the other elements are aggregations of processes or data groups. The idea is that if you have enough information at the most detailed level you count FP according to IFPUG rules; otherwise, you can estimate the size of larger elements (e.g., General or Macro processes) either on the basis of analogy (e.g., a given General process is "similar" to a known one) or according to the structured aggregation (e.g., a General process is composed of 3 Transactional BFC). By considering elements that are coarser-grained than the FPA BFC, the EQFP measurement process leads to an approximate measure of size in IFPUG FP.

Tables taking into account the previous experiences with the usage of EQFP are provided to facilitate the task of assigning a minimum, maximum and most likely quantitative size to each component. For instance, Table II provides minimum, maximum and most likely weight values for generic (i.e., not weighted) functions as given in [6]. The time and effort required by the weighting phases are thus saved. Such saving can be relevant, since weighting a data or transaction function requires analyzing it in detail.

TABLE II. EQFP: FUNCTION TYPE WEIGHTS FOR GENERIC FUNCTIONS.

	Weight		
Function type	Low	Likely	High
Generic ILF	7.4	7.7	8.1
Generic EIF	5.2	5.4	5.7
Generic EI	4	4.2	4.4
Generic EO	4.9	5.2	5.4
Generic EQ	3.7	3.9	4.1

C. Tichenor method

The Tichenor ILF Model [7] bases the estimation of the size on the number of ILF via the following formula for transactional system (for batch systems, Tichenor proposes a smaller multiplier): *EstSize = 14.93 #ILF*

This model assumes a distribution of BFC with respect to ILF as follows: EI/ILF = 0.33, EO/ILF = 0.39, EQ/ILF = 0.01, EIF/ILF = 0.1. If the considered application features a different distribution, the estimation can be inaccurate.

The fact that a method based only on ILF requires a given distribution for the other BFC is not surprising. In fact, the size of the application depends on how many transactions are needed to elaborate those data, and the number of transaction cannot be guessed only on the basis of the number of ILF, as it depend on the number of ILF just very loosely. Instead of allowing the user to specify the number of transactions that are needed, the Tichenor method practically imposes that the number of transactions complies with the distribution given above.

D. Simplified FP

The simplified FP (sFP) approach assumes that all BFC are of average complexity [8], thus:

EstSize = 4 #EI + 5 #EO + 4 #EQ + 10 #ILF + 7 #EIF

E. ISBSG distribution model

The analysis of the ISBSG dataset yielded the following distribution of BFC contributions to the size in FP: ILF 22.3%, EIF 3.8%, EI 37.2%, EO 23.5%, EQ 13.2%

The analysis of the ISBSG dataset also shows that the average size of ILF is 7.4 UFP. It is thus possible to compute the estimated size on the basis of the number of ILF as follows:

$EstSize = 7.4\ \#ILF\frac{100}{22.3}$

The same considerations reported above for the Tichenor model apply. If the application to be measured does not fit the distribution assumed by the ISBSG distribution model, it is likely that the estimation will be inaccurate.

F. ISBSG average weights

This model is based on the average weights for each BFC, as resulting from the analysis of the ISBSG dataset [9], which contains data from a few thousand projects. Accordingly, the ISBSG average weights model suggests that that the average function complexity is used for each BFC, thus

$$EstSize = 4.3\ \#EI + 5.4\ \#EO + 3.8\ \#EQ + 7.4\ \#ILF + 5.5\ \#EIF$$

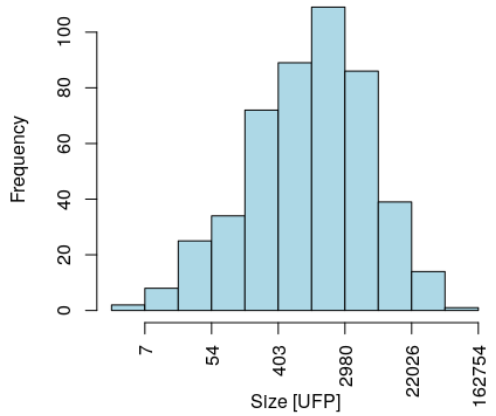

Figure 3. Distribution of projects' sizes in the studied dataset (note the logarithmic scale on the x axis).

IV. The Empirical Study

In this section, we first provide a brief description of the dataset used for the evaluation of functional size estimation methods; then, we describe the analysis and the resulting accuracy evaluations.

A. The Dataset

We use a dataset that collects data from 479 software applications developed and used by a Chinese financial enterprise.

Descriptive statistics of the dataset are given in Table III.

TABLE III. Descriptive statistics of dataset.

	Standard UFP
Mean	3554
St. Dev.	6673
Median	1155
Min	4
Max	80880

The distribution of size is given in Figure 3. Specifically, it is worth noting that the studied dataset contains relatively few data from small projects (32 projects not greater than 50 UFP) and relatively few data concerning very large projects (17 projects bigger than 20000 UFP).

B. The Analysis

For each of the 479 project in the dataset, we computed the estimated size, using each of the estimation method described in Section III.

To assess the estimates, in Figures 4 and 5 we plot the values of the estimates with respect to the actual size measured according to the standard IFPUG counting manual [3]. Specifically, Figure 4 illustrates the estimates yielded by the most accurate methods, while Figure 5 shows the estimates yielded by the less accurate methods.

In Figures 4 and 5, we also draw the estimated size = actual size line: if the estimates were perfect, all the points would lie on the line.

Looking at Figure 4, it is easy to see that the estimates provided by the NESMA Estimated, EQFP, Simplified FP, and ISBSG average methods are close to the x=y line, thus indicating fairly accurate estimates. On the contrary, in Figure 5 it can be observed that the estimates provided by the NESMA indicative (normalized and not normalized), Tichenor, and ISBSG distribution methods are widely spread, with many points definitely far from the x=y line, thus indicating that many estimates are quite inaccurate.

To better appreciate the accuracy of estimates, in the Appendix, Figures 13 and 14 illustrate estimates for projects having size in the [50, 20000] UFP range.

Figures 5 and 14 seem to indicate that the NESMA indicative (normalized and not normalized), Tichenor, and ISBSG distribution methods are rather inaccurate. To better investigate this indication, in Figure 6 we give the boxplots that illustrate the distributions of errors (errors are defined as the difference actual size minus estimated size). To keep the figure readable, outliers (i.e., the biggest errors) are not shown.

It is easy to see that NESMA Estimated, EQFP, Simplified FP, and ISBSG average weight methods have narrow errors distributions, close to the zero-error line. On the contrary, the other methods feature widespread distributions, in some cases centered on positive errors. Specifically, the Tichenor method provides the worst accuracy and tends to underestimate (for over 75% of the projects, estimates are smaller than the actual size). Similar considerations—although to a lesser extent—apply to the NESMA indicative (normalized) and ISBSG distribution methods. The NESMA indicative (not normalized) method does not tend to underestimate nor overestimate, but its estimation errors are often quite large.

To better study the errors of the best methods, in Figure 7 we give the boxplots of the estimation errors of the NESMA Estimated, EQFP, Simplified FP, and ISBSG average weight methods.

Figure 7 shows that the NESMA Estimated method tends to underestimates, while the Simplified FP method tends to overestimate. On the contrary, the EQFP and the ISBSG average weights methods appear symmetrically distributed with respect to the zero-error line.

It is worth noticing that size underestimation is quite dangerous, since it may lead to underestimating development cost

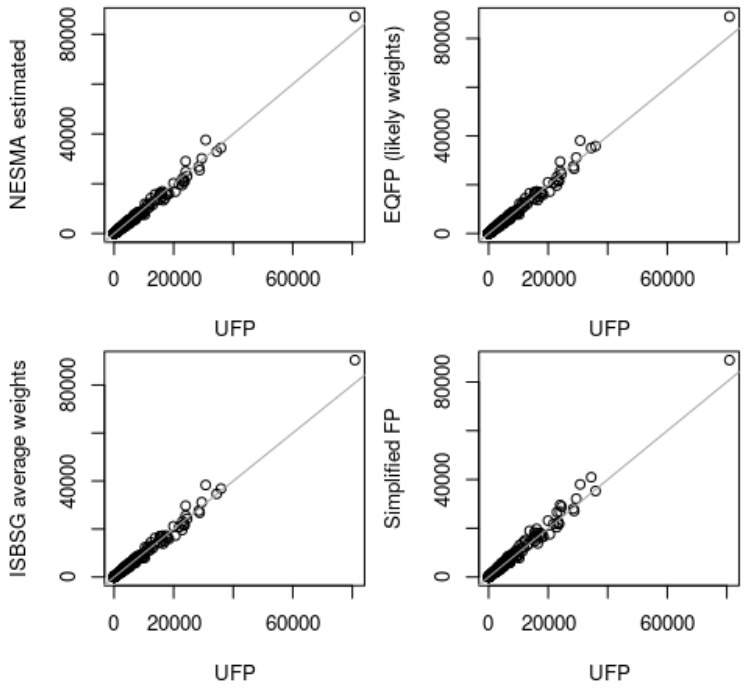

Figure 4. Standard IFPUG UFP measures vs. estimates provided by most accurate methods.

and duration, hence to defining not realistic plans, that usually lead to bad project organization, understaffing, overspending, etc.. Accordingly, using the NESMA Estimated method could be risky.

This observation concerning the NESMA Estimated method is particularly interesting because the NESMA Estimated method is also the one that most frequently provides the best accuracy in the lot of estimation methods, as shown in Figure 8 (the most accurate estimate here is the one characterized by the least absolute error). So, practitioners must consider that the NESMA Estimated method is likely to provide the smallest estimation errors, but is also most frequently underestimating.

C. Accuracy Evaluation

It is now necessary to evaluate quantitatively the accuracy of estimates. First of all—as suggested by Shepperd and MacDonell [13]—we checked whether estimates perform better than "baseline" models. Shepperd and MacDonell [13] proposed that the accuracy of a given estimation method be measured via the Mean Absolute Residual (MAR): $MAR = \frac{1}{n}\sum_{i=1..n}|y_i - \hat{y}_i|$, where y_i is the actual value of the i^{th} observation, and $\hat{y}_i$ is its estimate. Shepperd and MacDonell suggest to use random estimation, based solely on the known (actual) values of previously measured applications, as a baseline model. Shepperd and MacDonell observed also that the value of the 5% quantile of the random estimate MARs can be interpreted like α for conventional statistical inference, that is, any accuracy value that is better than this threshold has a less than one in twenty chance of being a random occurrence. Accordingly, the MAR of a proposed model should be compared with the 5% quantile of the random estimate MARs, to make us reasonably sure that the model is actually more accurate than random estimation.

Lavazza and Morasca [14] proposed to use a "constant model," where the estimate of the size of an application is given by the mean size of the other applications.

With our dataset, the MAR of the constant model is 3864 UFP, while the 5% quantile of absolute residuals for random estimates is 4566 UFP. The MARs of estimates are given in Table IV, together with MdAR (the median of absolute errors) and MMRE (the mean of relative absolute errors).

TABLE IV. ACCURACY INDICATORS FOR ESTIMATION METHODS.

Method	MAR	MdAR	MMRE
NESMA Estimated	315	73	0.102
NESMA Indic. (not norm.)	2135	435	0.629
NESMA Indic. (norm.)	1931	425	0.523
EQFP (likely weights)	299	79	0.103
Tichenor	2363	638	0.648
ISBSG distribution est.	2045	451	0.589
ISBSG average weights	298	82	0.103
Simplified FP	411	112	0.141

According to Table IV, all the evaluated methods satisfy the necessary conditions for being considered acceptable estimation methods.

Concerning the accuracy of estimates, in Figure 9 the distribution of absolute errors (excluding the biggest ones,

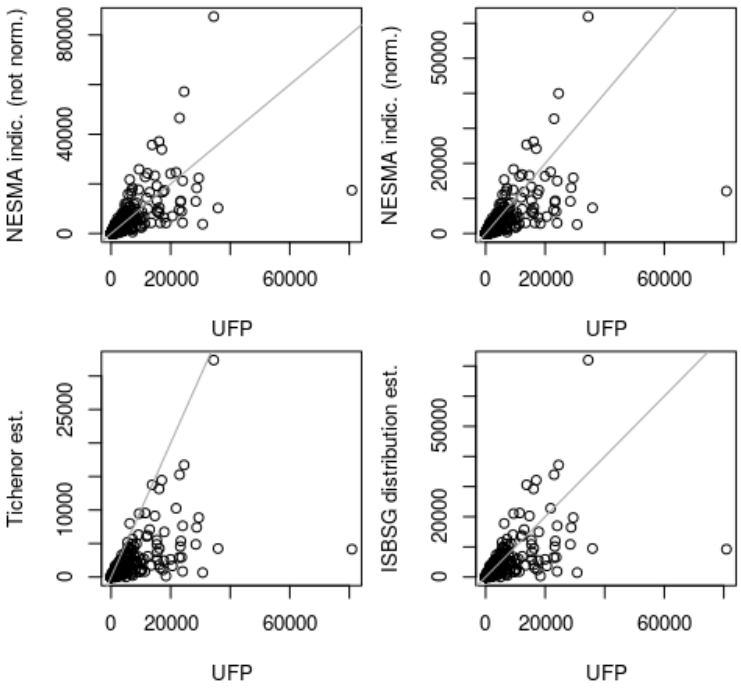

Figure 5. Standard IFPUG UFP measures vs. estimates provided by less accurate methods.

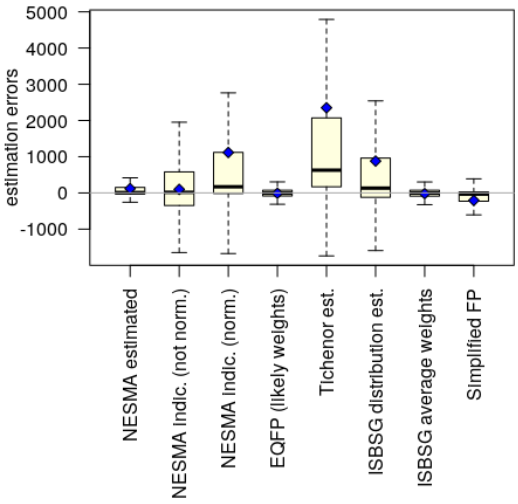

Figure 6. Boxplots of estimation errors (no outliers shown).

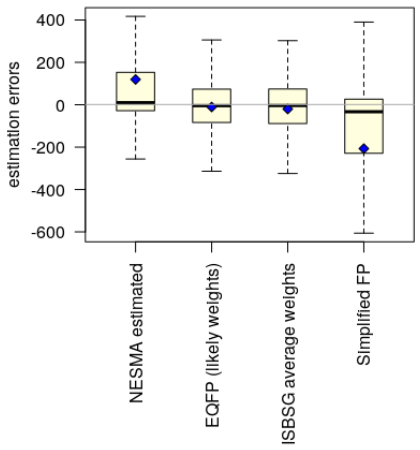

Figure 7. Boxplots of selected methods' estimation errors (no outliers shown).

to keep the figure readable) is given. The blue diamond is the mean, i.e., the MAR of the estimates. Figure 9 confirms the observation already reported in Section IV-B, i.e., even not considering the biggest errors, it appears that NESMA Indicative, Tichenor and ISBSG distribution methods feature many fairly inaccurate estimates.

As a result of the previously described findings, it seems that NESMA Estimated, EQFP, ISBSG average weights and Simplified FP methods qualify as the best candidates for size estimation, therefore these methods deserve most attention. Specifically, we should evaluate whether these methods are acceptable for functional size estimation. To this end, in Figure 10 the boxplot of the absolute errors of the estimates obtained via these methods are illustrated. Note that in Figure 10—unlike in Figure 9—all the absolute errors are

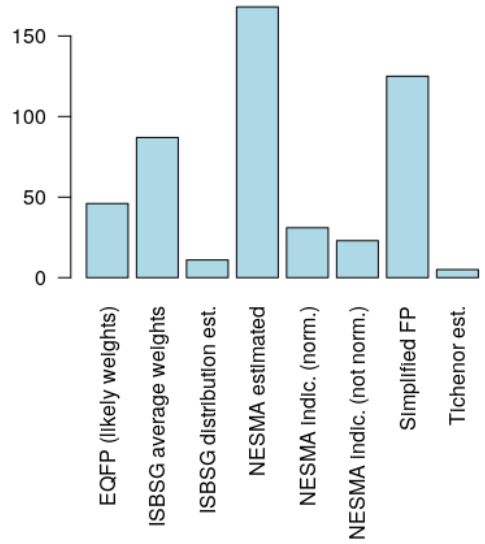

Figure 8. How many times every method yielded the best estimate.

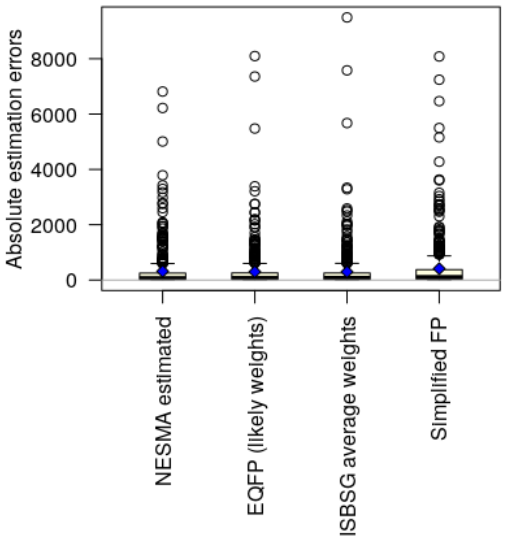

Figure 10. Boxplots of all absolute errors for selected methods.

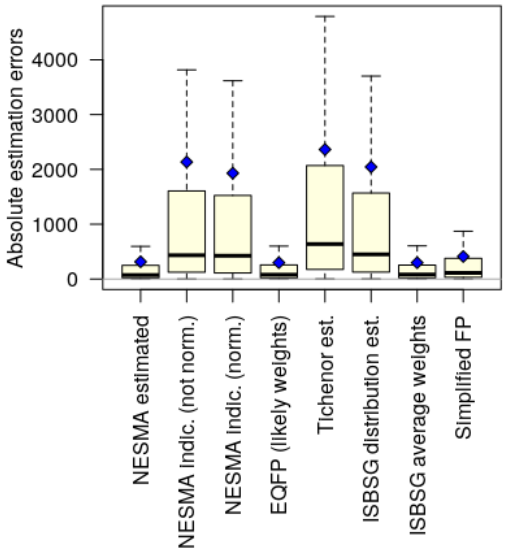

Figure 9. Boxplots of absolute errors (outliers not shown).

represented.

Figure 10 shows quite clearly that while the considered methods feature a fairly small absolute error, all methods are characterized by some quite big errors. To evaluate whether these errors are acceptable, we can consider that in general, relatively large estimation errors are deemed acceptable in very large projects.

To help practitioners appreciate the "importance" of errors with respect to the size of the estimated project, in Figure 11 we give the boxplots representing the distributions of relative errors (the relative error of an estimate is the estimation error divided by the actual size). To keep Figure 11 readable, the biggest errors are not shown.

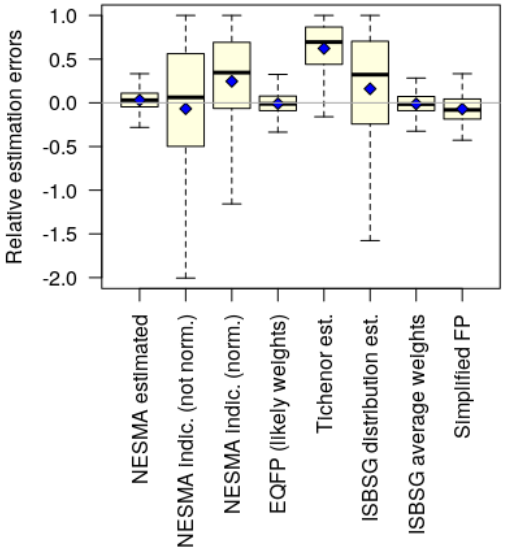

Figure 11. Boxplots of relative errors (no outliers).

Figure 11 shows that the estimation errors caused by the NESMA Indicative, Tichenor and ISBSG distribution methods are quite large also in relative terms. Specifically, these methods feature several relatively large underestimations: all the three mentioned methods underestimate no less that 25% of the projects by 50% or more. Even though the acceptability of estimates is largely subjective, we have strong doubts that any

practitioner would find these levels of accuracy acceptable.

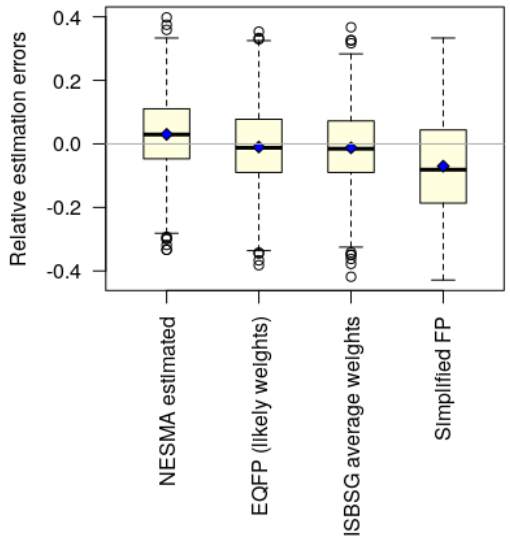

Figure 12. Boxplots of all relative errors for selected methods.

Considering the other methods, Figure 12 shows the boxplots of all the relative estimation errors. Figure 12 indicates that all the selected methods perform well. For all methods, the great majority of errors is in the [-20%, +20%] range, which is generally considered acceptable, especially when the estimate is performed in the earliest phases of the software lifecycle, when user requirements have not yet been fully analyzed.

V. Considerations on the Outcomes of the Empirical Study

The results obtained via the empirical study support a few considerations, which we illustrate in the following sections.

A. Methods based on the knowledge of data functions

The NESMA Indicative, Tichenor and ISBSG distribution estimation methods are all based on the knowledge of the number of data functions only. The NESMA Indicative method also requires that logical data files are classified as ILF or EIF, which is usually quite easy, since one has just to check whether the application being measured maintains (i.e., creates, modifies or deletes) the considered data files or not. On the contrary, the other methods considered in this paper base their estimates on the knowledge or transactional functions as well.

Since the NESMA Indicative, Tichenor and ISBSG distribution estimation methods use the knowledge of the number of data functions only, it is not surprising that they provide a level of accuracy that is definitely lower with respect to methods that account also for transactions.

So, the question is now whether it is reasonable to use estimates that are based exclusively on the knowledge of logical data files. First of all, we can observe that the question makes sense, since in many cases requirements analysis starts with domain analysis, which provides reliable indications concerning the data the software application will have to deal with. In such conditions, no method appears to yield acceptably accurate estimates.

B. Methods based on the knowledge of data and transaction functions

In order to estimate size using the NESMA Estimated, EQFP, ISBSG average weights or Simplified FP methods, you have to know both the number of data functions (and how they are classified into ILF and EIF) and the number of transactions (and how they are classified in EI, EO and EQ). If you have this knowledge of functional requirements, you can get size estimates whose error is in the [-20%, +20%] range, except in a minority of cases.

Given this, our results show that the Early & Quick FP and the ISBSG average methods are probably preferable, since they avoid both underestimation (which affects the NESMA Estimated method) and overestimation (which affects the Simplified FP method). In addition, the Early & Quick FP and the ISBSG average methods a slightly smaller MAR than the NESMA estimated method (see Table IV).

Actually, both the NESMA Estimated method and the Simplified FP method are based on assumptions that have never been verified. The former method assumes that data functions are all of low complexity and all transactions are of average complexity; the latter method assumes that all functions are of average complexity. Our study shows that none of these assumptions appear totally correct: in fact, the NESMA Estimated method tends to underestimate and the Simplified FP method tends to overestimate (note that, by definition, for a given application, the Simplified FP estimate will always be greater than the NESMA Estimated estimate).

The remaining methods, i.e., the EQFP and the ISBSG average weights methods are based on statistical analysis of large datasets. They propose for each function type the average weight observed. As a consequence, the definition of the two methods are very similar, and the estimates they provide are usually very close. This makes quite hard to choose one of the two methods. To this end, an interesting observation is that the EQFP method provides not only ‘likely’ weights for function types, but also low and high values (see Table II). By sizing all functions with the low weights and with the high weights gives a sort of confidence interval that could cope with the inherent uncertainty of the estimation due to the lack of details concerning functional requirements.

As a final consideration, the fact that both the EQFP and the ISBSG average weights methods performs well shows that organizations that have large enough collections of historical data could build their own estimation method following the same process as the EQFP and the ISBSG average weights methods: compute the average weight of each function type from the historical dataset and use such weights to estimate the size of new developments.

VI. Threats to Validity

Given the type of study we presented, there are two main threats to validity that need attention.

First, we should consider the correctness of the given data. In fact, the data in the analyzed dataset were derived from the analysis and measurement of functional requirements: both analysis and measurement could be affected by error, which would end up in the dataset. Concerning this threat, we are reasonably sure that the used data are of good quality, since they were collected by professionals in industrial contexts where functional size measures are quite important, hence great attention is posed in the measurement activities.

Second, we need to consider external validity, i.e., whether we can generalize the results of our study outside the scope and context that characterize the considered software projects. On the one hand, our dataset is much larger than the datasets usually involved in software engineering empirical studies; besides, our dataset includes data from fairly large projects (e.g., over 20000 FP). In this sense, our dataset represents a large and varied enough sample. On the other hand, all the considered projects are from the economic, financial and banking domain, hence we cannot be sure that the results of our study apply equally well in other domains. In this respect, readers are reminded that previous studies show some difference in accuracy when estimates concern other types of software applications, e.g., real-time applications [11].

VII. Related work

Meli and Santillo were among the first to recognize the need for comparing the various functional size methods proposed in the literature [15]. To this end, they also provided a benchmarking model.

Popović and Bojić compared different functional size measures—including NESMA Indicative and Estimated—by evaluating their accuracy in effort estimation in various phases of the development lifecycle [16]. Not surprisingly, they found that the NESMA Indicative method provided the best accuracy at the beginning of the project. With respect to Popović and Bojić, we made two quite different choices: the accuracy of the method is evaluated against the actual size of the software product and—consistently—all the information needed to perform measurement is available to all processes.

Santillo suggested probabilistic approaches, where the measurer can indicate the minimum, medium and maximum weight of each BFC, together with the expected probability that the weight is actually minimum, medium or maximum [17]. This leads to estimate not only the size, but also the probability that the actual size is equal to the estimate.

NESMA defined the application of FPA in the early phases of the application life cycle, and recognizes three function point analysis methods: Detailed function point analysis (currently corresponding to IFPUG measurement), Estimated function point analysis, and Indicative function point analysis. Using a database of over 100 developed and implemented applications, NESMA empirically evaluated the accuracy of the Estimated and Indicative FPA approximation methods [18]. The results showed that size measures of the high-level function point analysis and the detailed function point analysis are very close. Moreover, Indicative function point analysis gives a surprisingly good estimate of the size of several applications.

van Heeringen described the size accuracy—as well as the difference in measurement effort—of the NESMA Estimated and NESMA Indicative methods, by measuring 42 projects [4]. The results show that the estimation error of NESMA Estimated was in the [-6%, +15%] range, with average 1.5%; the estimation error of NESMA Indicative was in the [-15%, +50%] range with average 16.3%. In both cases the estimation error was evaluated with respect to detailed measurement.

Wilkie et al. [19] used five commercial projects to evaluate the cost-benefit trade-off of size measurement with respect to size estimation; they concluded that whilst the NESMA Indicative method was insufficiently accurate for the involved commercial organization, the NESMA Estimated approach was definitely viable.

IFPUG adopted NESMA methods for early "high-level" size estimation [10]. IFPUG suggested that 1) The High Level FPA method can be used to size an application early in the software development life cycle; 2) The High Level FPA method can also be applied as an alternative to standard FPA estimate (the outcome is not significantly different, while the estimation time is considerably shorter); 3) The indicative FPA method may be used to get a very fast, rough indication of the size of an application, but it is not suited for contractual commitments.

Lavazza et Liu [11] used 7 real-time applications and 6 non real-time applications to evaluate the accuracy of the EQFP [5] and NESMA methods with respect to full-fledged Function Point Analysis. The results showed that the NESMA Indicative method yields the greatest errors. On the contrary, the NESMA Estimated method yields size estimates that are close to the actual size. The NESMA Indicative method is generally outperformed by the other methods. The NESMA Estimated method proved fairly good in estimating both Real-Time and non Real-Time applications.

Morrow et al. used a dataset of 11 projects to evaluate the quality of sizing estimates provided by NESMA methods [20]. They also adapted NESMA methods' general principles to enhance their accuracy and extent of relevance, and empirically validated such an adapted approach using commercial software projects.

The main limitations of the mentioned research are that most studies used small datasets containing data concerning little projects of not industrial nature. In our paper, we evaluate measurement accuracy of the NESMA method with respect to FPA method over a dataset containing data from 479 industrial projects, of which several are above 10000 FP.

Ochodek proposed a method to approximate IFPUG FPA functional size in an automatic way, based on given UML use-case diagrams or a list of use-case names [21].

Meli proposed Simple Function Points (SiFP), as an alternative to standard IFPUG FP [22]. SiFP have been evaluated

via empirical studies [23], [24] that showed that they appear suitable replacements of standard FP.

VIII. Conclusions

In this paper we addressed the evaluation of the accuracy of functional size estimates that can be achieved via several Function Points estimation methods. To this end, we compared functional size measures obtained via the standard IFPUG Function Point Analysis process with estimates obtained via the considered estimated methods. Both measures and estimates were computed for a dataset containing data from 479 software projects. Based on the results of the analysis, we can draw a few relevant conclusions:

- The methods that use knowledge of both data and transaction functions provide estimates that are much more accurate than those provided by methods that use only knowledge of data functions.
- The methods that use knowledge of both data and transaction functions, namely the NESMA Estimated, Early & Quick FP, ISBSG average weights and the Simplified FP methods, provide estimates that are mostly in the [-20%, -20%] error range, hence their estimates are likely accurate enough in most cases.
- The NESMA Estimated method tends to underestimate. This can be dangerous, since at the initial stages of development one could be induced to believe that the development process will be shorter and cheaper than actually required.
- The Simplified FP method tends to overestimate.
- With the Early & Quick FP and the ISBSG average weights methods, the probabilities of getting underestimates and overestimates appear approximately equal.

Future work includes experimenting with new estimation methods that can be derived from the available dataset, and investigating whether and how estimation accuracy can be improved.

Acknowledgment

Parts of this work have been supported by the "Fondo di ricerca d'Ateneo" funded by the Università degli Studi dell'Insubria, by Zhejiang Provincial Science Foundation of China under grant no. LY19F020046, and by the Chinese Scholarship Council under grant no. 201708330399.

References

[1] L. Lavazza and G. Liu, "An Empirical Evaluation of the Accuracy of NESMA Function Points Estimates," in International Conference on Software Engineering Advances (ICSEA 2019), 2019.

[2] A. J. Albrecht, "Measuring application development productivity," in Proceedings of the joint SHARE/GUIDE/IBM application development symposium, vol. 10, 1979, pp. 83–92.

[3] International Function Point Users Group (IFPUG), "Function point counting practices manual, release 4.3.1," 2010.

[4] H. van Heeringen, E. van Gorp, and T. Prins, "Functional size measurement-accuracy versus costs-is it really worth it?" in Software Measurement European Forum (SMEF 2009), 2009.

[5] L. Santillo, M. Conte, and R. Meli, "Early & Quick function point: sizing more with less," in 11th IEEE International Software Metrics Symposium (METRICS'05). IEEE, 2005, pp. 41–41.

[6] DPO, "Early & Quick Function Points for IFPUG methods v.3.1 Reference Manual 1.1," DPO srl, 2012.

[7] C. Tichenor, "The IRS Development and Application of the Internal Logical File Model to Estimate Function Point Counts," in IFPUG Fall Conference of Use (ESCOM-ENCRESS 1998). IFPUG, 1998.

[8] L. Bernstein and C. M. Yuhas, Trustworthy Systems Through Quantitative Software Engineering. John Wiley & Sons, 2005.

[9] International Software Benchmarking Standards Group, "Worldwide Software Development: The Benchmark, release 11," ISBSG, 2009.

[10] A. Timp, "uTip – Early Function Point Analysis and Consistent Cost Estimating," 2015, uTip # 03 (version # 1.0 2015/07/01).

[11] L. Lavazza and G. Liu, "An empirical evaluation of simplified function point measurement processes," International Journal on Advances in Software, vol. 6, no. 1& 2, 2013.

[12] International Standardization Organization (ISO), "ISO/IEC 20926: 2003, Software engineering IFPUG 4.1 Unadjusted functional size measurement method Counting Practices Manual," 2003.

[13] M. Shepperd and S. MacDonell, "Evaluating prediction systems in software project estimation," Information and Software Technology, vol. 54, no. 8, 2012, pp. 820–827.

[14] L. Lavazza and S. Morasca, "On the evaluation of effort estimation models," in Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering. ACM, 2017, pp. 41–50.

[15] R. Meli and L. Santillo, "Function point estimation methods: a comparative overview," in Software Measurement European Forum (FESMA 1999), 1999.

[16] J. Popović and D. Bojić, "A comparative evaluation of effort estimation methods in the software life cycle," Computer Science and Information Systems, vol. 9, 2012.

[17] L. Santillo, "Easy Function Points – 'Smart' Approximation Technique for the IFPUG and COSMIC Methods," in Joint Conf. of the 22nd Int. Workshop on Software Measurement and the 7th Int. Conf. on Software Process and Product Measurement, 2012.

[18] nesma, "Early Function Point Analysis." [Online]. Available: https://nesma.org/themes/sizing/function-point-analysis/early-function-point-counting/ accessed on November 8, 2020

[19] F. G. Wilkie, I. R. McChesney, P. Morrow, C. Tuxworth, and N. Lester, "The value of software sizing," Information and Software Technology, vol. 53, no. 11, 2011, pp. 1236–1249.

[20] P. Morrow, F. G. Wilkie, and I. McChesney, "Function point analysis using nesma: simplifying the sizing without simplifying the size," Software Quality Journal, vol. 22, no. 4, 2014, pp. 611–660.

[21] M. Ochodek, "Functional size approximation based on use-case names," Information and Software Technology, vol. 80, 2016, pp. 73–88.

[22] R. Meli, "Simple Function Point: a new functional size measurement method fully compliant with ifpug 4. x," in Software Measurement European Forum, 2011.

[23] L. Lavazza and R. Meli, "An evaluation of Simple Function Point as a replacement of IFPUG function point," in 2014 joint conference of the international workshop on software measurement and the international conference on software process and product measurement. IEEE, 2014, pp. 196–206.

[24] F. Ferrucci, C. Gravino, and L. Lavazza, "Simple function points for effort estimation: a further assessment," in Proceedings of the 31st Annual ACM Symposium on Applied Computing, 2016, pp. 1428–1433.

APPENDIX

Figures 13 and 14 illustrate estimates vs. actual values, concerning projects in the [50, 20000] UFP range.

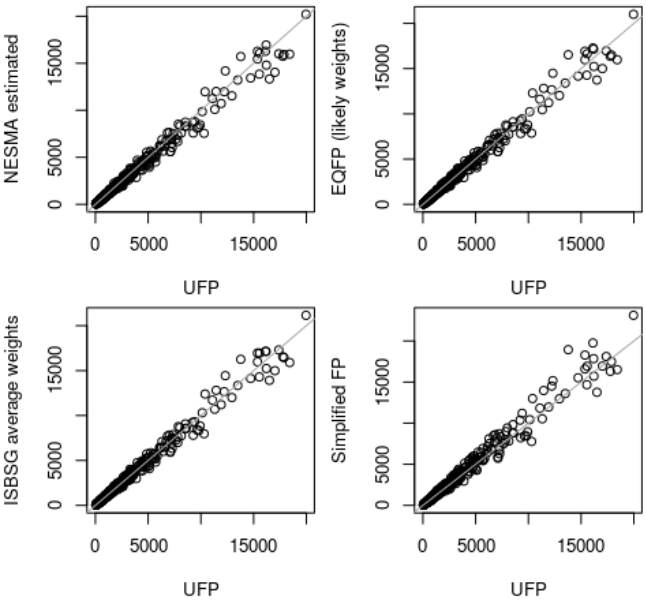

Figure 13. Standard IFPUG UFP measures vs. estimates (no smallest and biggest projects).

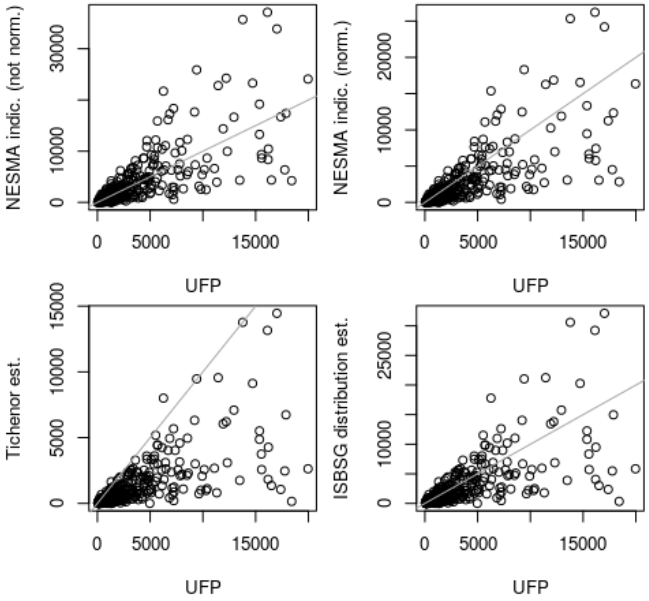

Figure 14. Standard IFPUG UFP measures vs. estimates (no smallest and biggest projects).