Applying moving windows to software effort estimation

Conference Paper · October 2009

DOI: 10.1145/1671248.1671260 · Source: DBLP

2 authors:

Chris Lokan
Australian Defence Force Academy
99 PUBLICATIONS **1,702** CITATIONS

SEE PROFILE

Emilia Mendes
Blekinge Institute of Technology
248 PUBLICATIONS **6,895** CITATIONS

SEE PROFILE

Applying Moving Windows to Software Effort Estimation

Chris Lokan
School of Engineering and IT
UNSW@ADFA
Canberra, Australia
c.lokan@adfa.edu.au

Emilia Mendes
Department of Computer Science
University of Auckland
Auckland, New Zealand
emilia@cs.auckland.ac.nz

Abstract

Models for estimating software development effort are commonly built and evaluated using a set of historical projects. An important question is which projects to use as training data to build the model: should it be all of them, or a subset that seems particularly relevant? One factor to consider is project age: is it best to use the entire history of past projects, or is it more appropriate in a rapidly changing world to use a window of recent projects?

We investigate the effect on estimation accuracy of using a moving window, using projects from the ISBSG data set. We find that using a moving window can improve accuracy, and we make some observations about factors that influence the range of possible window sizes and the best window size.

1. Introduction

Models for estimating software development effort are commonly built and evaluated using a set of historical projects. The usual approach involves separating the data into a training set (from which a model is built) and a testing set (with which the model's accuracy is assessed). An important question is which projects to use as training data to build the model: should it be all of them, or a subset that seems particularly relevant?

In this paper we propose that an important consideration when selecting a set of training projects is the sequence in which projects are completed.

We take a "project-by-project" approach to estimation, in which the chronological sequence of projects is important. A separate estimation model is built for each project in chronological order: the test set is the single project for which effort is to be estimated, and the training set is some or all of the projects completed before this single project was initiated. This is the situation faced in reality by an estimator.

Early in the process the set of completed projects is small, and all data is precious. Eventually, the data set becomes large enough that one might consider discarding training projects that seem less relevant. In particular, if one assumes that recent projects better reflect current development tasks and practice, while past projects become less relevant over time, it might make sense to discard older projects. In effect this is a "moving window" — as time passes, new projects enter the data set, and old projects cease to be considered as the window moves past them.

This idea seems intuitively realistic — but does it have any effect on the accuracy of the estimates? Using a window has a potential benefit and a potential disadvantage. It might help to produce a more accurate estimate for a new project, if more recent training data is more representative of the new project. It might hurt if the resulting smaller training set does not permit one to consider as many independent variables, or if the relationships between variables are harder to identify, so the resulting estimation model might be coarser and perhaps less accurate.

In this paper we investigate this trade-off. We address the following research questions:

1. Assuming a project-by-project approach to effort estimation, is there a difference between the accuracy of estimates using prediction models that are built using all available data in a training set, and the accuracy of estimates using prediction models that are built using only the N most recent projects in the training set? The null hypothesis is that there is no difference, for all values of N.

2. If there is a difference, can insights be gained by observing trends in estimation accuracy as N varies?

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

The contribution of this paper is that to our knowledge it is the first detailed study of the application of moving windows in software effort estimation, and the first to consider different window sizes.

The remainder of the paper is organized as follows. Section 2 briefly summarizes related work. Section 3 describes the research method employed in this study. Results are presented in Section 4, and discussed in Section 5. Section 6 discusses threats to validity, and finally our conclusions and directions for future work are presented in Section 7.

2. Related work

Chronological splitting has been applied as a data-splitting approach in several other domains, but is rare so far in software engineering. The few studies in software engineering that considered chronological splitting are presented next. Lefley and Shepperd [10], and Sentas et. al. [18], used chronological splitting as the basis for splitting their data into training and validation sets, when comparing effort and productivity models. Lokan and Mendes [12] compared a chronological split against a random split of data into training and testing sets, finding no significant impact on estimation accuracy.

Two studies addressed a slightly different problem, using information from previous tasks within a project to estimate effort in later tasks in the same project. MacDonell and Shepperd [13] studied effort distribution in major waterfall phases in 16 projects. They found the patterns of effort distribution between phases varied too much for this approach to work by itself, though it was helpful in conjunction with expert estimation. Abrahamsson et al. [1] were more successful in two projects from an extreme programming environment, finding that data from early iterations produced increasingly more accurate estimates for later iterations.

The most relevant work to the present paper is that in which the training data was viewed as a portfolio that grew over time as projects finished. Two papers [13, 16] noted that this is a sensible approach, but did not analyze their data in that way; one described it as future work [13].

Auer and Biffl [2] and Auer et al. [3] considered the effect of a growing portfolio in their research into estimation by analogy. They tracked changes in accuracy as the portfolio of completed projects grew. However, they did not consider the use of a window of projects.

In two recent studies we compared estimates based on a growing portfolio with estimates based on leave-one-out cross-validation, using two different data sets [11, 15]. In both cases, cross-validation estimates showed significantly superior accuracy.

We are aware of one study, by Kitchenham et al. [8], which considered a growing data set and whether a moving window should be used. They found that when they divided their data into four subsets by start date, the regression models changed between the subsets (there was no pattern to the changes, except a decline in adjusted R^2 over time). As a result they argued that old projects should be removed from the data set as new ones were added, so that the size of the data set remained constant. They recommended that the estimate for project n should be based on projects $n-30$ to $n-1$: a moving window of 30 projects. They found that doing so appeared to lead to slightly worse predictive accuracy, compared to retaining all training data, in those projects whose estimate could be affected by the use of such a window (this cannot be judged directly, however, since the accuracy statistics they tabulated were not based on the same sets of projects).

Kitchenham et. al. broke new ground by recommending the use of a moving window if regression equations change over time, and showing that using a moving window (of one particular size) had an effect on predictive accuracy. However, this was not the primary point of their paper, and they did not investigate moving windows more deeply. To our knowledge, the present paper is the first to study moving windows in detail.

The chronology of projects was also important in [17], which investigated changes in software development productivity over time. They found changes from year to year, and an overall pattern of improving productivity can be seen. Their focus was on characterizing productivity in different years, rather than using past data to estimate future projects. However, their results suggest that regression models are likely to change over time, which reinforces that a moving window might be appropriate.

3. Research Method

3.1. Data set

The data set used in this paper is the same one we analyzed in [11]. This data set is sourced from Release 10 of the ISBSG Repository.

Release 10 contains data for 4106 projects; however, not all projects provided the chronological data we needed (i.e. known duration and completion date, from which we could calculate start date), and those that did varied in data quality and definitions. To form a data set in which all projects provided the necessary

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

data for size, effort and chronology, defined size and effort similarly, and had high quality data, we removed projects according to the following criteria:

- Remove projects if they were not assigned a high data quality rating (A or B) by ISBSG.
- Remove projects for which the implementation date and/or overall project elapsed time is unknown.
- Remove projects if their size is measured in lines of code, or in a version of function points other than IFPUG, or in an outdated version of function points (size measured with an older version is not directly comparable with size measured with IFPUG version 4.0 or later). Also remove projects that measured size with an unspecified version of function points, and whose completion pre-dated IFPUG version 4.0.
- Remove projects for which the size in unadjusted function points is unknown.
- Remove projects with for which the development team effort (resource level 1) is unknown. Projects that also reported the effort for other participants were retained, but only if they identified the development team effort separately, and only the development team's effort was used in our analysis.
- Remove projects whose normalized effort differs from recorded effort. This should mean that the reported effort is the actual effort across the whole life cycle.
- Remove Web projects.

In the remaining set of 909 projects, 231 were all from the same organization and 678 were from other organizations. We only selected the 231 projects from the single organization as the use of single-company data was more suitable to answer our research questions than using cross-company data. Preliminary analysis showed that three projects were extremely influential and invariably removed from model building, so they were removed from the set. The final set contained 228 projects.

Release 10 of the ISBSG database provides data on numerous variables; however, this number was reduced to a small set that we believed could potentially have an impact on effort, and which did not suffer from a large number of missing data values. The remaining variables were size (measured in unadjusted function points), effort (hours), and three categorical variables: development type (new development, re-development, enhancement), primary language type (3GL, 4GL), and platform (mainframe, midrange, PC, multi-platform).

Table 1. Summary statistics for ratio-scaled variables

Variable	Mean	Median	StDev	Min	Max
Size	496	266	699	10	6294
Effort	4553	2408	6212	62	57749
PDR	16.47	8.75	31.42	0.53	387.10

Summary statistics for size, effort, and project delivery rate ("PDR", calculated as effort divided by size; high PDR values indicate low productivity) are presented in Table 1.

We do not know the identity of the organization that developed these projects; however we are aware that the projects were developed for a variety of industry sectors where insurance, banking and manufacturing were the most common. Start dates range from 1994 to 2002, but only 9 started before 1998. 3GLs are used by 86% of projects; mainframes account for 40%, and multi-platform for 55%; these percentages for language and platform vary little from year to year. There is a trend over time towards more enhancement projects and fewer new developments. Enhancement projects tend to be smaller than new developments, so there is a corresponding trend towards lower size and effort.

PDR also changes with time; in particular, it is notably worse in projects completed before 2001. The change in PDR is illustrated in Figure 1, which relates size to effort for the first 75 projects and the last 75 projects in chronological sequence, and also shows the regression lines relating log(size) to log(effort) for each group. The intercept is higher and the slope steeper for the early projects, both implying more effort for a given size (i.e. poorer productivity) in the early projects. This suggests that if the early projects are included as training data for the later projects, the resulting estimation models may be biased towards over-estimating the effort for the later projects. A moving window, which eventually ignores the early projects, might be helpful in this situation.

3.2. Estimating effort for a single project

A project-by-project approach was used to estimate the effort for each project. When a window was not being used, the approach was as follows:

- A project p was selected as the target project, for which effort was to be estimated.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

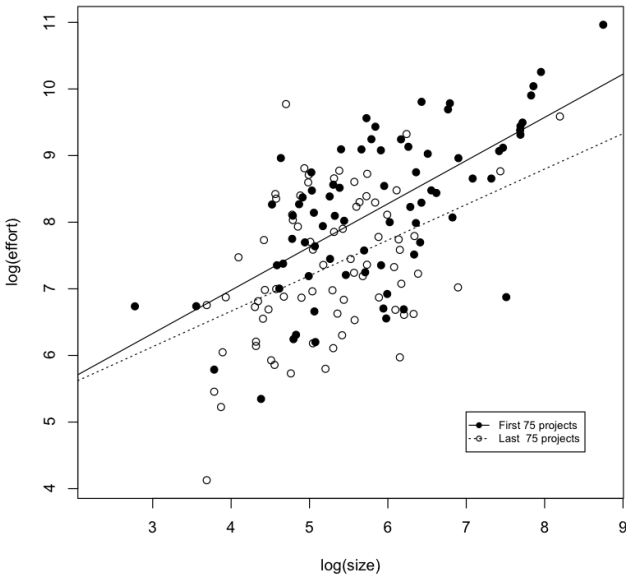

Figure 1. Size vs. effort for the first 75 and last 75 projects

- The starting date (*sd*) for p was used to split the remaining projects into two groups: *completed* projects that had finished prior to *sd*, and *active/future* projects that were active or had not yet started at *sd*.
- The set of *completed* projects was used as the training set to build a regression model R.
- Cook's distance [6] was used to determine whether any highly influential *completed* projects should be removed; if any were removed, R was then refitted using the reduced data set.
- R was applied to p's data in order to obtain an effort estimate for p.

These steps were repeated for each project in turn, until effort estimates were obtained for all projects.

When a window was being used, the approach was the same except for one additional step: before using the set of *completed* projects to build the regression model R, any that did not fall within the window of most "recent" projects was removed from the set.

3.3. Building an estimation model

All of the models used in this investigation were built using an automated process, programmed in the statistical programming language R.

The procedure followed in the automated process was as follows:

- The first step in building every regression model was to ensure numerical variables were normally distributed. We used the Shapiro-Wilk test on the training set to check if Effort and Size were normally distributed. Statistical significance was set at $\alpha = 0.05$. In every case, Size and Effort were not normally distributed, and were therefore transformed to a natural logarithmic scale.
- Independent variables whose value was missing for the project to be estimated were not considered for inclusion in the estimation model.
- Every model included Size as an independent variable. Beyond that, given a training set of N projects, no model was investigated if it involved more than $N/10$ independent variables (rounded to the nearest integer), assuming that at least 10 projects per independent variable is desirable [19].
- Models were built using multivariate backward stepwise regression (using the "`fastbw()`" function from Harrell's `Design` package for R).
- To verify the stability of an effort model, the following approach was used [14]: Calculate Cook's distance values for all projects to identify influential data points. Any projects with distances higher than $(3 \times 4/N)$, where N represents the total number of projects, were immediately removed from the analysis. Those with distances higher than $4/N$ but smaller than $(3 \times 4/N)$, were removed temporarily in order to test the model stability, by observing the effect of their removal on the model. If the model coefficients remained stable and the goodness of fit improved, the highly influential projects were retained in the analysis.
- If there was more than one possible model in which all independent variables were significant, the model with highest adjusted R^2 was preferred.

3.4. Prediction accuracy measures

The most common measures used in software engineering to compare different effort estimation techniques are the mean (MMRE) and median (MdMRE) magnitude of relative error; and prediction at level l (Pred(l)): the fraction of estimates that are within l % of the actual values. It is suggested [5] that l should be set at 25% and that a good prediction system should offer this accuracy level 75% of the time.

Although MMRE, MdMRE and Pred(l) are often used as evaluation criteria to compare different effort estimation techniques, Kitchenham et al. [9] showed

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

that MMRE and Pred(l) are respectively measures of the spread and kurtosis of z, where (z = estimated value / actual value). They suggested the use of box plots of z and box plots of residuals as useful alternatives to simple summary measures, since they can give a good indication of the distribution of residuals and z and can help explain summary statistics such as MMRE and Pred(.25). Average z values that exceed 1.0 indicate a bias towards over-estimates, while average values below 1.0 indicate a bias towards under-estimates.

z itself has a bias towards minimizing over-estimates. For this reason we prefer to focus on absolute residuals, which have no bias towards over- or under-estimates, and show the scale of errors.

In this study we used MRE and absolute residuals to compare the accuracy of effort models. In addition, some of the accuracy comparisons are based on the shape of graphs. In particular, we use Regression Error Characteristic Curves ("REC curves") [4] to visualize the distributions of accuracy measures. Space limitations prevent us from presenting boxplots as well.

To test for statistically significant differences between accuracy measures, we used the paired-samples t-test and set statistical significance at 5%. All calculations were done using the statistical language R.

3.5. Moving windows

3.5.1. Windowing approach

There are two ways in which a moving window might be defined: by number of projects, or by duration. By number of projects, only a fixed number of the most recent projects are retained. At times when the rate of completing projects is high, the time span of the window might be quite small, but when the rate of projects completed per month is low the time span of the window might be quite large. By duration, the size of the window is a time span. Projects completed before the start of the time span are excluded, and conversely, only projects within the defined time span are included. The number of such projects might therefore vary.

Each has potential benefits. A window based on duration can be scaled to include only those projects that reflect recent development projects and practices. One must hope that there are enough of them to support sound statistical analysis, or else one must trade off the degree of recency against the number of projects. On the other hand, a window based on a fixed number of projects can be scaled to provide enough data for sound analysis (e.g. Kitchenham et. al.'s 30 projects [8]), but might stretch across a wide time span so that earlier projects are perhaps less relevant to current projects.

In this paper we investigate the second approach, in which a window contains the N most recently completed projects.

3.5.2. Evaluation data sets

We have described the use of a window in forming the set of training data. The choice of window size also affected the testing data set, and number of projects that could sensibly be used to evaluate the impact of using the window. When evaluating the difference between two approaches (using a window or not), it makes no sense to include projects where there could be no difference in the prediction models. This was the case during the initial stages, while the window filled up.

If a window was not used, no training data was ever rejected due to its age. If a window of N projects was used, the model building process was identical until the window filled up, because no projects were rejected due to their age until at least N+1 were complete.

For example, when a window of 20 projects was used with our data set, it was only at the 28^{th} project in the sequence that at least 21 projects had finished (at that start date, 21 projects had finished and 6 were active) and the use of the window could make a difference because at least one project was excluded from the training set. Hence we evaluated the impact of using a window of 20 projects by comparing the two estimates (with and without window) for projects 28 to 228 in the sequence. With a window of 30 projects, the window could only make a difference from the 51^{st} project in the sequence (when 33 had finished and 17 were active), so the impact of the window was evaluated on projects 51 to 228 in the sequence.

Thus, as the window size increased, the set of evaluation projects decreased. This placed an upper bound on the window sizes that could sensibly be investigated. The absolute limit was 208, leaving 3 projects for evaluation.

3.5.3. Range of window sizes

In studies where a data set is divided into training and testing sets, it is common to use a split of about 2:1 respectively. The maximum window size that allowed us to retain at least one third of our data for evaluation purposes was a window of 114 projects. As this seems a somewhat arbitrary number, we set 120 projects as the largest window size to be investigated.

For the smallest window size, we needed to establish the smallest amount of data that seemed useable. Kitchenham et. al. [8] suggest at least 30 training

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

projects (partly to reflect typical data set sizes in the literature, and partly because 30 is about the size at which sample properties begin to converge to population properties) and no fewer than 20. On the other hand, Humphrey [7] suggests that 3 may be useable if they show a reasonable correlation ($R^2 \geq 0.5$) between size and effort.

To determine the smallest useful data set size, we experimented with window sizes ranging from 5 projects to 30. For each project and each window size, we formed the relevant training set (the last N projects to finish) and used that data set to build a regression model. Many regression models, usually based on few data points, were not significant at the 5% level. The question was, what was the minimum window size at which the regression model was almost always significant at the 5% level? In this data set, the threshold turned out to be 20 projects. (This reinforces Kitchenham et. al.'s [8] suggested lower limit).

4. Results

4.1. Summary statistics

Table 2 shows the effect of window sizes on mean absolute residuals.The second column shows the number of projects for which the use of a window could make a difference; the third column shows the mean absolute residual across all of those projects, when each is estimated using only the N most recently completed projects; the fourth column shows the mean absolute residual for the same set of projects, when the training set for each contains all projects completed so far; this varies from 21 projects at the start of the sequence to 214 projects at the end. The final column shows the p-value when the paired-samples t-test was used to compare the absolute residuals with and without a window; values below 0.05 indicate a statistically significant difference. Note that we investigated all window sizes from 20 to 120, but due to space limitations only some of them are tabulated in Table 2. This is still sufficient to show the important trends.

Figure 2 plots the difference in mean absolute residuals against window size. The dotted vertical line at window size 52 is a boundary: for window sizes up to 52, the difference is statistically significant (favouring the retention of all training data rather than using a window), but for larger window sizes the difference is not statistically significant (thus favouring neither approach).

Table 3 and Figure 3 present similar information for MMRE. This time there are two boundaries. For window sizes up to 23 the difference in MMRE is statistically significant, favouring the retention of all training data; for window sizes of 85 or more the difference is statistically significant, favouring the use of the window; in between, the differences are not statistically significant, thus favouring neither approach.

Table 2. Mean absolute residuals with different window sizes

Window size (N)	Testing projects	With window	Without window	p-value
20	201	3137	2712	0.030
30	178	2969	2662	0.035
40	165	3075	2631	0.019
50	153	2947	2624	0.050
60	136	2840	2567	0.104
70	126	2518	2405	0.487
80	126	2468	2405	0.513
90	111	2009	2189	0.154
100	88	2169	2234	0.191
110	75	1762	1821	0.468
120	71	1755	1840	0.113

Table 3. Mean MRE with different window sizes

Window size (N)	Testing projects	With window	Without window	p-value
20	201	1.618	1.330	0.029
30	178	1.436	1.402	0.706
40	165	1.483	1.413	0.505
50	153	1.442	1.449	0.942
60	136	1.480	1.493	0.890
70	126	1.491	1.537	0.566
80	126	1.473	1.537	0.416
90	111	1.219	1.392	0.002
100	88	1.232	1.405	0.001
110	75	1.155	1.403	0.011
120	71	1.273	1.417	0.008

We also calculated Pred(.25) for each window size. For window sizes up to 56, Pred(.25) was always better if all data was retained; for larger window sizes it was always better to use a window. The differences are trivial, though: at best, Pred(.25) could be raised from 18% to 24%, but these are far from the 75% target [5].

4.2. Error characteristic curves

Tables 2 and 3 and Figures 2 and 3 present mean values for different accuracy measures. It is also useful

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

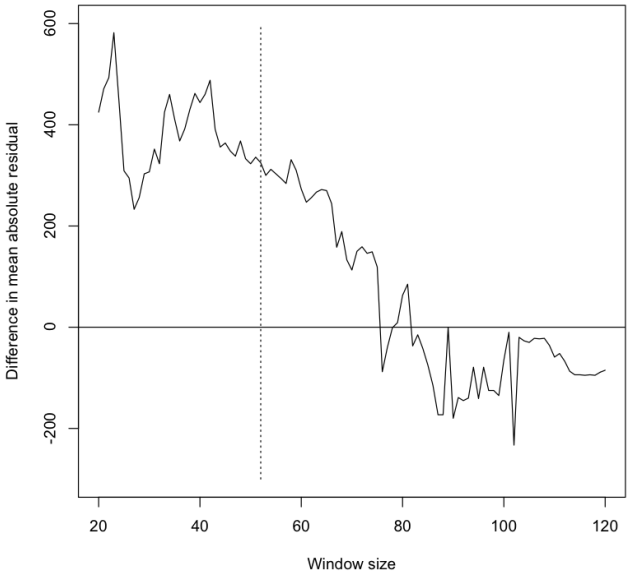

Figure 2. Difference in asolute residuals, by window size

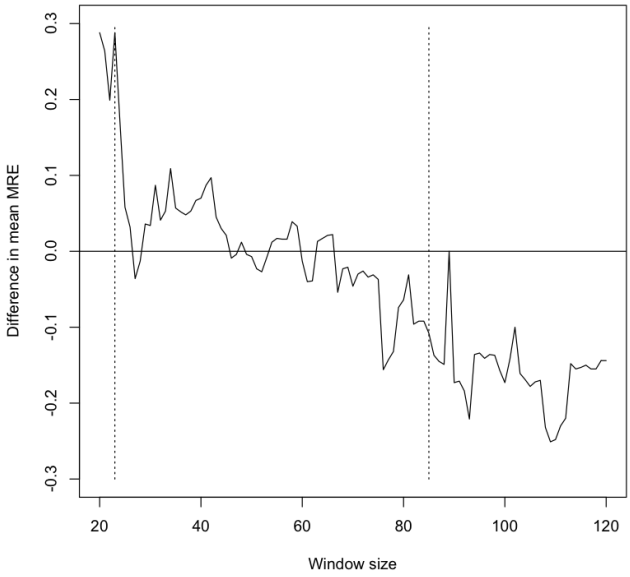

Figure 3. Difference in MMRE, by window size

to compare the distributions of the accuracy measures. For this purpose, we use regression error characteristic ("REC") curves [4].

In an REC curve, the x-value represents the value of an accuracy statistic. The y-value represents the fraction of error values that are less than or equal to the given x-value.

The ideal shape of a REC curve for an accuracy statistic whose value should be minimized (such as absolute residuals, or MRE) has a steep slope at the start, a "knee" near the upper left corner, and then a gradual slope towards the asymptotic y-value of 1.0. When comparing two curves, the one that is nearer the upper left corner is better.

For example, Figure 4 shows the REC curves for absolute residuals, for projects where a window of 20 projects could make a difference, with and without that window. The dotted line is closer to the upper left corner than the solid line, across the entire range of error values. Using terminology from the field of optimization, we can say that one approach (retaining all data) "dominates" the other (using a window of 20 projects): it is always better. This shows that retaining all training data is not just better on average than using a window of 20 projects: it is better across the board.

In contrast, Figure 5 shows the REC curves for absolute residuals at the other end of our range, where the window size is 120 projects. Table 2 shows a small advantage for using a window at this size, but the difference is not statistically significant. The absence of any significant difference is supported in Figure 5, where the curves are very similar. However, the REC curve is always at least as good when using the window: using a window dominates retaining all training data. Thus, even though the difference is not statistically significant, it still seems worthwhile to use a window.

Figures 6 and 7 show the corresponding REC curves for MRE, with window sizes of 20 and 120 projects, respectively. Again we see that it is better to retain all training data rather than to use a window of 20 projects (and Table 3 shows that the paired-samples difference in MRE is statistically significant); with a window size of 120 projects, Figure 7 shows that it is better to use a window (again, the difference is statistically significant).

Clearly, in this data set one does better to retain all possible training data than to use only the most recent 20 projects. But if the choice is to use all projects or only the most recent 120 completed projects, it is better to use a window rather than to use all projects. Somewhere in between there must be a cross-over point, at which it becomes better — or at least no worse — to use a moving window:

- With a window of 24 to 52 projects, MMRE is always better when using all training data, but the difference is not statistically significant. Mean absolute residuals are also better when using all training data, and this difference is statistically significant.

- With a window of 53 to 66 projects, average MRE

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

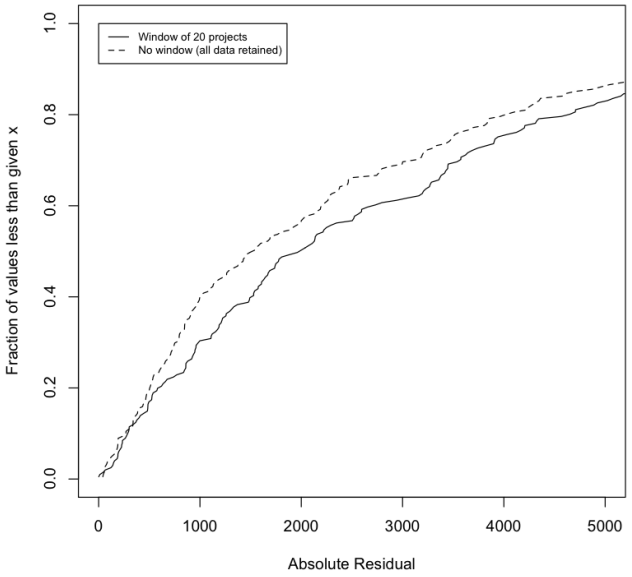

Figure 4. Absolute residuals, window of 20 projects

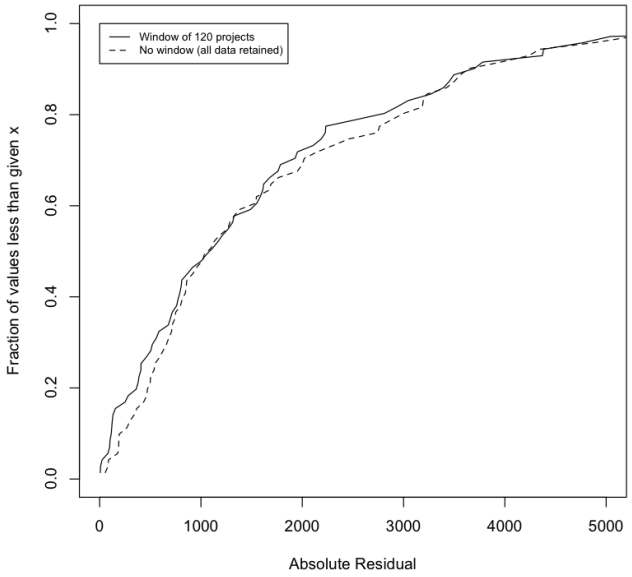

Figure 5. Absolute residuals, window of 120 projects

and absolute residuals are still both better when using all training data, but neither difference is statistically significant.

- With a window of 67 to 81 projects, MMRE is always lower using a window than without it. At this threshold the difference is not statistically significant, but the REC curve (Figure 8) shows that using a window dominates using all training data for all but a small fraction of the range.

- With a window of 82 projects or more, both MMRE and the mean absolute residual are lower using a window than using all training data. For a window of 82 to 84 projects, the differences are not statistically significant in either MRE or absolute residuals, but the REC curves show that using a window is better than using all training data. With a window of 85 projects or more the difference in MMRE is statistically significant (the difference in mean absolute residuals never achieves statistical significance).

Hence some obvious candidates for the "optimum" window size are windows of 53 projects (at which point neither accuracy statistic shows a statistically significant disadvantage for the window), 67 projects (when MMRE begins to favour the window), and 85 projects (when average statistics favour the window and the advantage in MRE is statistically significant).

The absolute residuals and MRE values resulting from our estimation models did not normally have Gaussian distributions. It may therefore be better to use a non-parametric test for significant differences in accuracy measures. It turned out that using the paired-samples Wilcoxon test instead of the paired-samples t-test made little difference. The only difference was that for windows of 24 to 52 projects, the Wilcoxon test did not find the difference in absolute residuals to be statistically significant. This still leaves 67 and 85 projects as candidates for the optimum window size.

4.3. Considering the entire data set

The results presented above show that using a window can be advantageous. However, the advantage only becomes clear as the window size increases, which means that more projects are needed to fill the window (so using a window does not affect their estimate) and fewer projects are affected by using the window. This may be all very well from a research point of view, but a practitioner would likely also care about the accuracy of estimates across the whole portfolio of projects.

It seems from the results above that a small window size has a negative effect. A large window size makes a significant positive difference, but to fewer projects. Overall, how does the accuracy of effort estimates vary with different window sizes, when the entire portfolio of projects is considered?

Figures 9 and 10 summarize the absolute residuals and MRE respectively, evaluated across the entire data

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

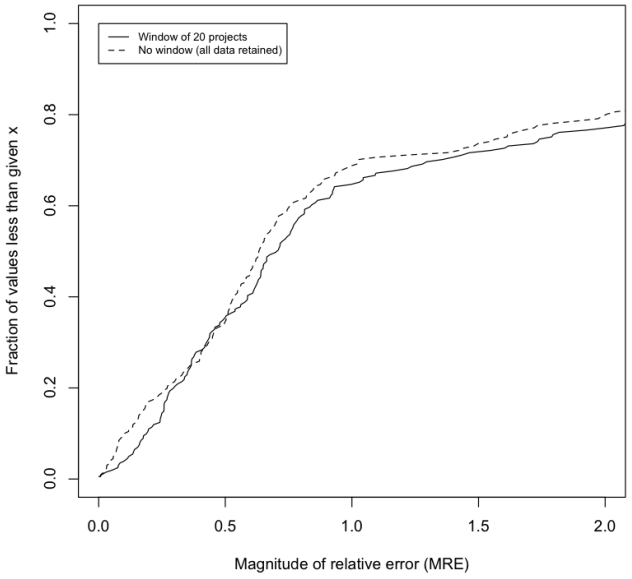

Figure 6. MMRE, window of 20 projects

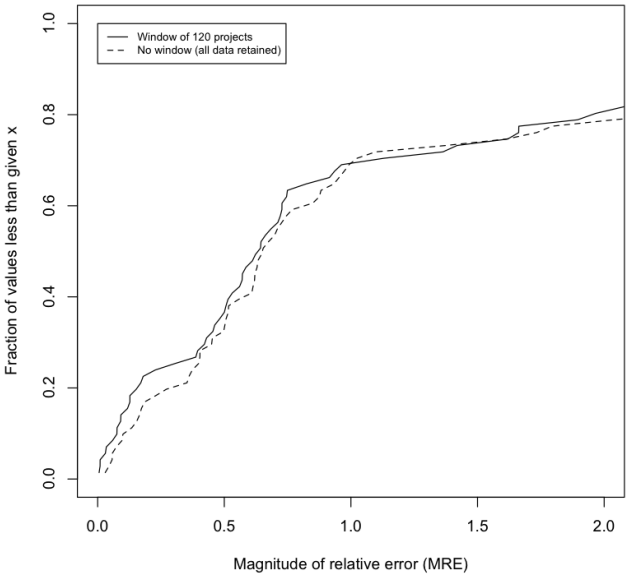

Figure 7. MMRE, window of 120 projects

set, for different window sizes. Both show that the mean error value varies with the window size, and is effectively minimized at a window size of around 75 projects.

5. Discussion

Our first research question was whether any use of a window of recent projects made a difference to accuracy. The null hypothesis is rejected under the following circumstances: for window sizes of 20 to 23, based on both MRE and absolute residuals; for window sizes of up to 52, based on absolute residuals; and for window sizes of 85 or more, based on MRE. Clearly the use of a window can affect accuracy.

The second question is whether insights can be gained by observing trends in accuracy as the window size varies. The general trend is that accuracy is increased when using larger windows, compared to not using windows. This may be misleading, however, as several inter-related factors could be involved, discussed next.

One factor is the number of independent variables that can be considered. This depends on the size of the data set. It favours larger windows (larger training sets), to the extent that being able to consider more independent variables leads to more accurate models (at least for the data set employed in this study). We can expect that there is one threshold (number of projects) below which estimation models are not statistically significant, and another above which there is so much variation in the data that it does not help to add more projects they add as much noise as they add extra information. Homogeneity in the data set would be relevant here; we might expect these thresholds to be higher with more heterogeneous data.

The rate of change in an organization's software development tasks and practices will affect window sizes. If change is rapid ("churn time" is short) the window should be small, so that only very recent projects are retained; if change is slow, a larger window could be used.

The number of completed projects per year will also be relevant. If this is large there is plenty of data to work with, and a range of window sizes might be considered. If it is small, and the time needed to accumulate useful information exceeds the churn time, using a window may add no value at all.

To summarize, the aim is to choose a window size that enables one to include recent projects that are representative of current tasks and practices, while excluding projects that seem out of date; while also having enough data to provide useful information for estimation. The factors discussed in the last three paragraphs may all influence the trade-off, and they are inter-related. Homogeneity in the data may affect how little is needed to be useful; this and the rate of organizational change would affect how long it takes to accumulate useful information; the rate of change also affects how much data it is valuable to keep for estimation purposes. This study provides some initial insights, but understanding these factors and their in-

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

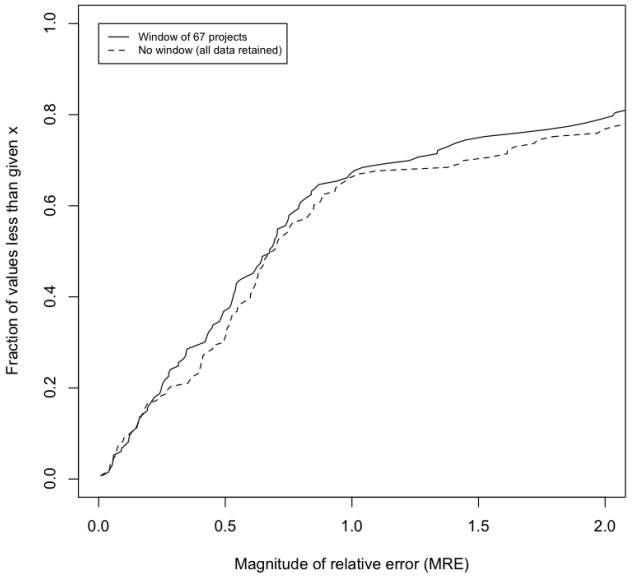

Figure 8. MMRE, window of 67 projects

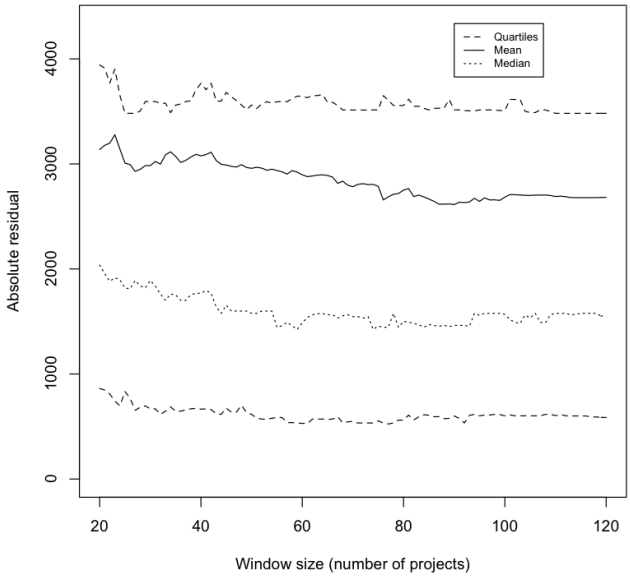

Figure 9. Absolute residuals, measured on entire data set

teractions will require further research.

Note that "old" data need not be simply discarded. It can still be useful, for example in longitudinal studies to investigate long-term trends. This is a different goal to shorter-term needs such as calibration of estimation methods for use in the current environment.

We have found that with this data set, using a small window is not advantageous. With fewer than 25 training projects the data can really only support two independent variables. The first is always Size; the second is usually LangType4GL, or occasionally DevTypeNew. These variables explain less than 35% of the variation in effort showing that the models simply do not capture enough information (they may be statistically significant, compared to no regression at all, but they are not accurate). By the time the window contains about 60 projects, there is enough data to identify the effect of each independent variable: most models involve size, language type and platform; few include the development type. Note that the larger models are still not accurate, with mean absolute residuals being about 70% of mean actual values, and MMRE being well over 1. The nature of the models remains basically the same for windows of 60 or more projects.

The present data set is very heterogeneous, and only being able to work with size and one other independent variable is restrictive.

In this data set, a window of only 20 projects is unrealistic as well: for most of the data set, 20 projects meant that training data comes from a time window of about 4 months. It does not seem realistic that projects completed only four months ago are "out of date".

At the other extreme, a window of 120 projects represents a time span averaging a little less than two years. This sort of scale makes more sense: tasks and practices from two years ago could well be less relevant now. However, the good performance with window sizes in the low 100s (Figures 2, 3 and 10) is more an artifact of the data than an inherent consequence of large windows. With these window sizes, only the last 75 to 80 projects in the data set are affected by the use of the window, and only about the first 45 are excluded from the window. It happens that the difference in productivity between the first 50 or so and the last 80 or so projects to finish is as big as it gets in this data set.

We saw that accuracy statistics are minimized across the entire data set if a window of 75 projects is used. Other useful boundaries (based on statistically significant differences, REC curves, and overall differences between means) occur at window sizes of 53, 67 and 85 projects. Therefore, for this particular data set, a window size of 75 (that is, the estimate for project n is based on data from projects $n-75$ to $n-1$) seems appropriate. Across most of the data set, this corresponds to a time window of about 12 months.

We emphasize that a window of 75 projects appears to strike the best balance for just this particular data set, which comes from one organization with its specific environment and rate of project completion. Other organizations, with different project completion rates

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

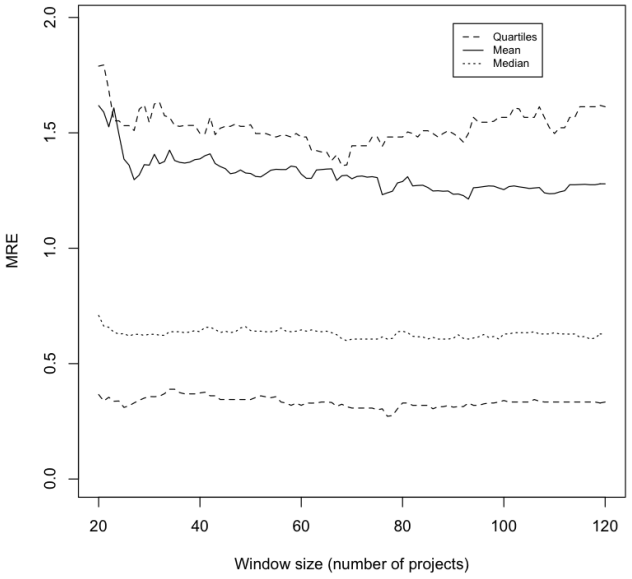

Figure 10. MRE, measured on entire data set

and different rates of change in their environment, may strike a different "sweet spot".

6. Threats to Validity

This study has some limitations and threats to validity.

First, the ISBSG repository is a convenience sample, and does not represent a random sample of projects. It is not clear that the projects studied here are representative of the software industry, or even of the organization from which they come. It remains possible that our results cannot generalize beyond this data set (all other studies based on convenience samples share the same threat). We trust that by studying data from only one organization, numerous potential sources of variation are removed; and that since the data set is large and covers a long time span, it is a fair representation of this organization's projects.

Second, all the models employed in this study were built automatically. Automating the process necessarily involved making some assumptions, and the validity of our results depends on those assumptions being reasonable. For example, logarithmic transformation is assumed to be adequate to transform numeric data to an approximately normal distribution; residuals are assumed to be random and normally distributed without that being actually checked; when choosing between two models in which all independent variables were significant, the one with higher adjusted R^2 is assumed to be preferred; multi-collinearity between independent variables is assumed to be handled automatically by the nature of the stepwise procedure. Based on our past experience building models manually with ISBSG data, we believe that these assumptions are acceptable. One would not want to base important decisions on a single model built automatically, without at least doing some serious manual checking, but for calculations such as project-by-project chronological estimation across a substantial data set we believe that the process here is reasonable.

7. Conclusions

We have analyzed a large data set, to see whether using a moving window of only the most recently completed projects, instead of using all available projects as training data, has a significant effect on the accuracy of effort estimation models. We have shown that it does indeed have a statistically significant effect on accuracy, as measured by absolute residuals and MMRE.

In this data set, we found that a minimum of 20 projects is needed as training data, so this is the minimum possible window size. For small windows of 20 to 25 projects, using a window has a negative effect (this supports Kitchenham et al.'s observation that accuracy appeared slightly worse with a window of 30 projects [8]). For large windows of 85 projects or more, the window has a positive effect, but fewer projects are affected. The best window size seems to be about 75 projects, which represents about one to two years of data for this organization.

There is no reason to expect that these specific window sizes or durations would be found to be important in other data sets. However, we believe it seems reasonable to expect that using a window would make a difference to accuracy in other data sets, as long as there is some variation over time in productivity or in the impact of productivity drivers — which is almost a given in the changing world of software development.

If this is true in general, it has implications for practitioners and researchers. For practitioners, it means that it is better to use a window of recent projects than to retain all possible training data. The value of having more data from which to learn is less than the value of having more recent data. For researchers, if the research is to seem relevant to practitioners it is important to consider the sequence of projects, and the use of time windows as one consideration when selecting relevant projects as training data for building estimation models.

We have discussed several factors that could influence whether a window is useful, and what window sizes might be appropriate. In this single study we have

978-1-4244-4841-8/09/$25.00 ©2009 IEEE

shown that windows can help, but we cannot separate the effect of the different factors. That requires further work. In the immediate future we plan to investigate windows based on duration instead of fixed numbers of projects; and to replicate the work with different data sets. Although the data set we have analyzed here is itself a "real world" data set, it is larger amd more heterogeneous than most in this research field; we plan to study data sets that are smaller, more homogeneous, or involve fewer completed projects per year.

References

[1] P. Abrahamsson, R. Moser, W. Pedrycz, A. Sillitti, and G. Succi. Effort prediction in iterative software development processes – incremental versus global prediction models. In *ESEM*, pages 344–353. IEEE Computer Society, 2007.

[2] M. Auer and S. Biffl. Increasing the accuracy and reliability of analogy-based cost estimation with extensive project feature dimension weighting. In *ISESE*, pages 147–155. IEEE Computer Society, 2004.

[3] M. Auer, A. Trendowicz, B. Graser, E. J. Haunschmid, and S. Biffl. Optimal project feature weights in analogy-based cost estimation: Improvement and limitations. *IEEE Trans. Software Eng.*, 32(2):83–92, 2006.

[4] J. Bi and K. P. Bennett. Regression error characteristic curves. In T. Fawcett and N. Mishra, editors, *ICML*, pages 43–50. AAAI Press, 2003.

[5] S. D. Conte, H. E. Dunsmore, and Y. E. Shen. *Software engineering metrics and models*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1986.

[6] R. Cook. Detection of influential observations in linear regression. *Technometrics*, 19:15–18, 1977.

[7] W. S. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[8] B. Kitchenham, S. L. Pfleeger, B. McColl, and S. Eagan. An empirical study of maintenance and development estimation accuracy. *Journal of Systems and Software*, 64(1):57–77, 2002.

[9] B. Kitchenham, L. Pickard, S. G. MacDonell, and M. J. Shepperd. What accuracy statistics really measure. *IEE Proceedings - Software*, 148(3):81–85, 2001.

[10] M. Lefley and M. J. Shepperd. Using genetic programming to improve software effort estimation based on general data sets. In *GECCO*, volume 2724 of *Lecture Notes in Computer Science*, pages 2477–2487. Springer, 2003.

[11] C. Lokan and E. Mendes. Investigating the use of chronological splitting to compare software cross- company and single-company effort predictions. In *Proceedings of the 12th Conference on Evaluation & Assessment in Software Engineering (EASE 2008)*, pages 151–160. BCS, June 2008.

[12] C. Lokan and E. Mendes. Using chronological splitting to compare cross- and single-company effort models: Further investigation. In *Proceedings of the 32nd Australasian Computer Science Conference (ACSC 2009)*. Australian Computer Society, Jan. 2009.

[13] S. G. MacDonell and M. J. Shepperd. Using prior-phase effort records for re-estimation during software projects. In *IEEE METRICS*, pages 73–. IEEE Computer Society, 2003.

[14] K. Maxwell. *Applied Statistics for Software Managers*. Software Quality Institute Series. Prentice Hall PTR, 2002.

[15] E. Mendes and C. Lokan. Investigating the use of chronological splitting to compare software cross- company and single-company effort predictions: a replicated study. In *Proceedings of the 13th Conference on Evaluation & Assessment in Software Engineering (EASE 2009)*. BCS, Apr. 2009.

[16] I. Myrtveit and E. Stensrud. Sw cost estimation: Measuring model performance of arbitrary function approximators. In M. H. Hamza, editor, *IASTED Conf. on Software Engineering and Applications*, pages 449–455. IASTED/ACTA Press, 2004.

[17] R. Premraj, M. J. Shepperd, B. A. Kitchenham, and P. Forselius. An empirical analysis of software productivity over time. In *IEEE METRICS*, page 37. IEEE Computer Society, 2005.

[18] P. Sentas, L. Angelis, I. Stamelos, and G. L. Bleris. Software productivity and effort prediction with ordinal regression. *Information & Software Technology*, 47(1):17–29, 2005.

[19] B. G. Tabachnick and L. S. Fidell. *Using Multivariate Statistics*. Harper-Collins, 1996.

978-1-4244-4841-8/09/$25.00 ©2009 IEEE