Quantifying Software Process Improvement

C. Verhoef

Free University of Amsterdam, Department of Mathematics and Computer Science,
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands

x@cs.vu.nl

Abstract

Many IT-metrics display large variation, time dependencies and noise, making it seemingly impossible to draw conclusions from them. Most of the software engineering literature proposed ways to stamp out this undesired behaviour, so that simple questions by management become simple to answer. In this paper we accepted that IT-metrics misbehave, in fact, we argued that large variation, time dependencies, and considerable noise are inherent to many IT-metrics. Many other fields know misbehaving metrics as well. These metrics range from the long-term temperature dynamics of beaver to the intra-tick graphs of the S&P500, their behaviour being sometimes even worse than our IT-metrics. We successfully applied the analysis methods common in other fields to software engineering questions. We illustrated our approach by solving a real-world problem. We answered the simple question by management whether a software process improvement program affecting 1500 IT-developers and business staff delivered its value. Moreover, we were able to predict the trends of important KPIs, like cost per function point, which enabled proactive steering and control. Our approach is not limited to this single question, but has a rich application potential to countless management and control issues concerning information technology.

Keywords and Phrases: Empirical software engineering, quantitative software engineering, software metrics, software process improvement, SPI, DSDM, CMM, Case study, IT-dashboard, balanced scorecard, peer review, Fagan inspection, function points, reliability of function point counters, distribution characteristics of software metrics, extreme value theory, heavy tail analysis, time series analysis, outlier detection, autoregression, moving averages, heteroscedasticity, smooth regression analysis, vector autoregression, *ARIMA* modelling, *GARCH* modelling, forecasting, IT-metrics, IT-audit, IT-assessment.

Die gerade Linie ist gottlos und unmoralisch.
– Friedensreich Hundertwasser [50].

1 INTRODUCTION

You can't control what you can't measure—everyone in IT nods in assent when hearing this maxim once coined by Tom DeMarco in his seminal book on controlling software projects [22]. I cannot measure, therefore I cannot control is equivalent to this maxim. And if we use simple logic to remove the two negations, we end up with: I can control, therefore I can measure, or in DeMarco's style: You can measure what you can control. All of a sudden this starts sounding a bit silly: namely, if you can

control software projects, you can obviously measure them, but what is the point then of measuring them, if you are already in control? All of a sudden this maxim starts sounding a bit silly, but let's put maxims aside for a moment, and try to understand what DeMarco is really aiming at. From his book we learn that if we measure IT with so-called convergent metrics, we can establish predictive models by which we gain control. And convergent means that IT-metrics can be related via standard regression methods. With these revealed relations between important software-specific variables, we create predictive power leading to control. Do these convergent IT-metrics exist? We think that they do not exist, at least we have never spotted them in practice. By no means, this implies that DeMarco's work is rendered useless, only the assumption that the IT-metrics should bear certain statistical appropriate behaviour is too strict. The next question is then, of course, can we still establish predictive power in the presence of non-convergent IT-metrics? DeMarco does not provide a formal definition of convergent or non-convergent IT-metrics, other than that the latter fail to subdue themselves to standard regression analysis methods. It is not our task to define the notions of others rigorously, but from the pictures illustrating his non-convergent IT-metrics, we suspect that non-convergent IT-metrics contain too many outliers, and exhibit considerable variation and noise.

Fighting variation The outliers and the noise, are they good or bad, or just a fact of life? If they are bad, we need to find a way to reduce the outliers, and temper the noise. Obviously, this has not happened since DeMarco published his book in 1982. At the time of writing this paper, decades later, we still observe IT-metrics with many outliers and noise. In his pamphlet *Why software costs vary*, Capers Jones testifies of this [53]:

> The software literature has typically relied on simplistic results that present only overall findings such as total project effort and total projects costs per function point or per line of code. This overall data is subject to wide variations for reasons that are difficult to understand. Serious economic analysis of software costs and productivity cannot occur unless the details of activity patterns are also defined, and unless the specific compensation levels and burden rate adjustments for projects are also included.

What is of particular interest here is Jones' observation that the wide variations are *difficult to understand*. We suspect that this is what DeMarco dubbed non-convergence. Many seem to strive to eradicate outliers and noise, in a quest for preciseness. For instance, Jones explains that software costs differ for a number of important reasons, in his words:

> A fundamental problem with software cost measures is the fact that salaries and compensation vary widely from job to job, worker to worker, company to company, region to region, industry to industry, and country to country.

Embracing variation When these and other issues would be better understood, completely deterministic results would ensue. We think that this is not going to happen. Namely, even in a situation where all differences that Jones notes are kept constant, still the costs of software vary widely. So we go one step further: keeping everything in the above quote invariant, software costs still vary widely from IT-project to IT-project. Maybe its just a fact of life that IT-metrics are containing a lot of outliers and display considerable noise. In this paper, we accept these facts instead of trying

to fight them, and therefore, we are not going to dive deeper into the activities on a per-project basis to eradicate this variation. From now on we assume that there will be many outliers and there is considerable noise for most IT-related metrics.

Simple questions, simple solutions? Starting from this assumption we are interested in answering the same question as DeMarco discussed in the case of convergent IT-metrics: can we gain control by inferring predictive models from the measured data? The answer is yes, but the road to that answer is by far not a straight line. In this paper we will illustrate how to deal with IT-metrics that contain many outliers and display noise that cannot be ignored. The relatively simple question that serves as an example to illustrate our ideas is as follows. In a large organization an expensive software process improvement program was initiated to produce IT more effectively and more efficiently. The simple question that management would like to see answered is whether the investment in this software process improvement program delivered the projected results. This was further quantified, namely that the cost per function point would be below a certain threshold upon successful implementation of the SPI-program. While answering this question, we explain how to deal with noisy IT-metrics with many outliers. Dealing with such questions consists of two parts: establishing the methods that fit the particular data and problem best, and carrying out a final audit. For the methods we need time, and therefore we usually do not have the final data available. Once we have the right methods in place, we can reiterate the analysis (often with some slight variation) and carry out an audit on all the information. So the first part of this paper consists of an extensive analysis of the initial data and the second part the final audit on the latest data. We call both sets the *research set* and the *audit set*.

1.1 What to expect?

Before we dive directly into technicalities, we first set the stage in Section 2, where we illustrate how to quantify aspects of software engineering once we accept the fundamental nature of noisy IT-metrics with many outliers. Moreover, we sketch the real-world problem on software process improvement in more detail. This organization measures the software process, which is an exception to the rule, so we can quantify at all.

Function points In Section 3 we address the function point data, compare individual function point analysts, and groups of analysts. Furthermore, we will discuss related work on the reliability of function point counters that has been done in the past [57, 56]. A first result will be that we explain a method to assess the accuracy and plausibility for function point totals. From these analyses we can infer how much recounting of function point totals is necessary for a qualitative check on the collected data.

Costs In Section 4, we turn our attention to estimated and actual costs. As it turns out in this particular case, we reveal so-called retrofitted data. So we cannot assess the plan accuracy of the cost estimates directly, since the estimates are overwritten with the actual costs after project closure. To that end, we use industry benchmarks to establish whether the actual costs are plausible.

Tails In Section 5, we want to assess the properties of the 3 basic IT-metrics (function point size, duration, and cost), and their derivative the costs per function point. We do

this by a so-called heavy tail analysis. Roughly, since the function point sizes are accurate and plausible, and for the costs this seems to be the case as well, then at least the properties of the function point data should somehow be present in the other metrics as well. One prominent property of the function point size is that there are many outliers, which should somehow be reflected in the other IT-metrics as well. We carry out such an analysis on the four IT-metrics, and quantify their tails. We simulate their behaviour via Monte Carlo simulation techniques, and conclude that the models are adequate. This implies that the distributional properties of the function point data are also present in the other indicators: duration, cost, and cost per function point.

Time In Section 6 we take time-dependencies into account. Namely, if we wish to say anything at all about the *progress* of a software control-variable, we have to investigate the IT-metrics over time. Methodologically, we should then start out with an analysis of correlations and co-movements of the entire system of function point sizes, and their corresponding durations and costs. Since this is a rather complex task, we will for explanatory reasons first (rightly) assume that this analysis is satisfactorily, and start out with a univariate time series analysis of the cost per function point over time. First we will show that the methods that are used within the organization to derive progress from the data are not adequate. Then we will investigate the microscopic behaviour of the cost per function points over time. We model this so-called time series, and diagnose the model by simulation and assessment methods. Next, we forecast this important KPI (Key Performance Indicator), which shows a rise in costs per function points.

Outliers To gain more certainty on the found rise, we construct a more robust model in Section 7, where time dependent outliers are identified and modelled so that a more precise model ensues. Also this model predicts a rise in the costs per function point. As the previous model, ex post forecasts show that the models possess credible predictive power.

Volatility Section 8 is devoted to the question whether the variance of the cost per function point varies itself over time. This is also called volatility or heteroscedasticity. We will show that the cost per function point is not "improving" in displaying less variance, when time progresses. This is a question that is often asked in SPI-programs, since it is assumed that IT-metrics become "more convergent" if the process improves. This need not be the case, and we explain why this is not a problem.

Smoothing After a detailed analysis of the microscopic behaviour of the IT-metrics, we address in Section 9 the question how to extract macro-behaviour from noisy IT-metrics to detect trends, if any. As pointed out briefly, the methods proposed to that end by the organization are not adequate, so we need to find alternatives. We propose these alternatives in the form of smoothing operations on the cost per function point metric which varies considerably over time. We assess the validity of the various smoothing operations by investigating their residual structure, which should be "as random as possible."

Vector After this so-called univariate time series analysis, we proceed in Section 10 with a trivariate vector time series analysis, to explore the properties of the system of function point sizes, and their supposedly corresponding durations and costs over

time. We model this time vector and its volatility. Furthermore we show that the found models adequately describe the system of IT-metrics. Our analysis shows that there is a clear correlation between the three IT-metrics, as should be the case with trustworthy data. We use the found model to predict the future behaviour of the three IT-metrics, and from that we derive a third prediction of the cost per function point (by dividing the cost prediction by the function point size prediction). Also this third prediction shows that the cost per function point will most likely rise. This concludes the first part of our paper, namely, the analysis of the research set.

Audit Next, in Section 11 we carry out a variant of the earlier analysis on the most recent data, to conduct the actual audit. Of course, we use the developed methods of the first part of this paper. An important result is that the thrice predicted rise in the cost per function points indeed materialized in the audit set. This confirms the predictive power of the inferred models from the data. To answer the simple question by management, we capture the macroscopic properties of the noisy IT-metric, with and without outliers. In Section 12, we combine all the knowledge gained, and come to a final audit: the target is met, and we provide evidence that this can hardly be a coincidence. In Section 13, we conclude the paper and provide references to the literature. But before we commence with all this, we discuss related work.

1.2 Related work

There is an abundance of work being done on improving the software process, there is even a separate journal devoted to the software process. Since this paper is not about software process improvement, but rather on an approach to quantify software issues like software process improvement, we refrain from giving a detailed account of the literature on this large topic in software engineering. Equally as well, we could have taken as example the quantification of the effects on important KPIs of the implementation of a CASE tool, introducing a software development method like RUP, configuration management, assessing maintenance, and so on. Some papers on SPI also address the question to quantify its value.

One such paper is [45]. In that paper the focus is on questions like: what effect does process improvement have on estimates of effort, product quality, and customer satisfaction? So this is more about the improvement of plan accuracy. The used statistical techniques are linear regression, rank correlation and χ^2 tests, all techniques that we do not use in this paper. Interestingly, the authors are not too satisfied with the statistical techniques and they warn the reader: "statistical techniques must be applied with care [..] process improvement must be very effective to alter statistical results significantly". Moreover, they were "altering data in different ways to reflect the impact of process improvement" which indicates that they simulate the effects of SPI by changing the data on the current process to see what the answers to their questions might be in reality, whereas we just measure the actual situation, also over time, and analyse the trends for real. And although the paper is almost ten years old, we still think that their observation holds today: "Although statistical techniques have been used to assess and predict product and process characteristics, little work has been done to statistically evaluate the effectiveness of process improvements."

There is an extensive report on the assessment of software process improvement techniques [25], again indicating that there are many mixed results. Overall, their findings are that management buy-in, involvement of technical staff in the SPI effort,

ensuring understanding current processes and alignment to business, clear SPI-goals, tailoring improvements, and respect are critical success factors. This is a more qualitative approach. Indeed, they indicate that further studies need to be performed to determine the rate of returns as an organization improves its processes according to current best practice models. In this paper we present such a study, containing a large sample of tens of thousands of function points in software over hundreds of IT-projects.

Naturally, the SEI has come up with reports on the benefits of CMM-based SPI-efforts [46]. This report claims (correctly): "Quantifying the results from investing in software process improvement is challenging [..] Are we improving? If so, by how much? These can be surprisingly difficult questions to answer." Apart from some initial findings that CMM-based SPI-programs can pay off, this report summarizes a number of the larger problems with choosing IT-metrics, and causality between changes in those IT-metrics and the SPI-program itself. We will address some of their issues and one of the reasons why this paper is a bit long, is perhaps their observation that the answers to their (and our) simple questions are indeed surprisingly difficult to find.

A more recent paper [69] extensively discusses a cost/benefit strategy for choosing the most promising types of SPI: ranging from the personal software process, clean-room development, reuse-driven software development, defect prevention, inspections, testing, implementing CMM, to certifying for ISO 9000. So, different aspects of SPI-efforts are compared, based on the costs and benefits of SPI as reported by 24 case studies describing various SPI-efforts and their results. The findings are mixed [69, p. 81]: "nevertheless, the survey and identification of SPI costs and benefits revealed a lack of uniform, industry standard metrics for measuring and reporting the costs and benefits of SPI." Again, severe problems with metrics are addressed, but no solutions are provided.

In a recent four page position paper [80], we can read that the problems are large and complex: "because of the incompleteness and inaccuracy of software measurements and metrics, the actual deployment of software measurements based quantitative software management and control could bring forth to software engineering practices have not yet appeared by large." It is then proposed that autonomic computing could solve the problems by a feedback control mechanism that ties measurement to SPI. Furthermore, they see the large variation as a main difficulty: "The large variability between software projects and that between software organizations prohibit the usability of measurements for software management and control." It is then proposed to divide software projects in phases, measure those phases, so that via regression the future phase can be predicted and compared to historical information of other projects in the same phase. This idea is somewhat similar to that of Jones [53], but it is neither worked out nor applied in their paper.

2 QUANTIFYING THE UNQUANTIFIABLE

As the related work already pointed out, it seems that the variation is so wide that issues become close to being unquantifiable. Indeed, the area of software development is fairly immature. According to Standish Group, the failure rates of software projects are high: about 30% of software projects fail, 50% are twice as expensive, take twice as much time, and deliver half the functionality, and only 20% of the software projects is on time, within budget, and with the desired functionality [51, 37, 38, 39]. In absolute figures, Standish group estimated in 1995 that this costed in the USA $ 81 billion on failed projects, and another $59 billion on serious cost overruns. These figures are sus-

tainable, given recent estimates of global IT-project failure costs of 290 billion dollar annually, of which 150 billion in the USA and 140 billion in Europe [21]. Furthermore, there are extended glossaries of runaway projects [35, 34, 36], confirming the overall image of immaturity. It turns out that starting an overall metrics program fails in 80% of the cases, which was shown in a longitudinal study of the US Department of Defense [66]. From this we might conclude that the field of software engineering is hopeless when it comes to quantifying costs, durations, projects size, scope, benefits, and other important indicators that are common in other engineering disciplines.

We think there is hope, but then we have to make a few fundamental assumptions about the important indicators that we wish to gain control over. The first one is to forget about the quest for trying to capture the magic formulas that would quantify software engineering. There is no grand software equation that is doing the trick. The second and most important fundamental insight is that we should incorporate in our thinking is that the important indicators are stochastic: they exhibit random behaviour and hopefully also some nonrandom behaviour. We will argue in this paper that these assumptions free ourselves from the roadblocks that hamper constructive thoughts in the direction of solutions.

For starters, let us reexamine the Standish results again, but now in mathematical/statistical terms. Projects are often twice as expensive, take twice as long and deliver half the functionality. How does this translate to statistical terminology?

- Cost and duration have an asymmetric leptokurtic possibly heterogeneous probability density function (PDF) with a heavy right tail.
- If we would count function point sizes of the Standish projects in retrospect, we should also find an asymmetric leptokurtic possibly heterogeneous PDF but now with a heavy tail on the left.

Let us explain this. The PDFs are asymmetric since there is a minimal cost, duration and size, so the PDFs will not vanish to minus infinity. The PDFs are possibly heterogeneous since without a measurement discipline we resort to fantasy numbers. Like a 6 month or a 12 month project, or a million dollar, 5 million dollar, or 10 million dollar cost. This clutter implies that a histogram or PDF of the distribution of such data will possibly display more than one local maximum. This is called heterogeneous data, since it is the superposition of more than one PDF. Leptokurtic stands for more peaky than a normal distribution. This is often present in combination with heavy tails. The heavy tails are present since we have so many outliers: many projects take longer than expected, and are more expensive than expected, and deliver less than expected. The word *expectation* is made precise by the heavy tail: the probability that we find some extremal value is large, and larger than if the indicators were normally distributed.

Now we describe a situation where we planned to mitigate the above risks, for instance, by implementing a software process improvement program. If this is done successfully, we would expect to find the following properties for the PDFs for size, duration and cost.

- The distribution for IT-project sizes is asymmetric leptokurtic possibly heterogeneous with a heavy *right* tail.
- Cost and duration, should display similar asymmetric leptokurtic possibly heterogeneous PDFs with a heavy right tail.
- Over time, IT-project size, duration, and cost should be correlated somehow.

- Over time, the cost per unit of production should show a downward trend, which is usually not linear.

Namely, when you start out with proper requirements analysis, and proper prioritization techniques, you will end up with a lot of so-called 80/20 decisions. This means that we plan to deliver 80% of the solution in 20% of the effort. Also for a given IT-budget you will prioritize projects by importance, and then we end up with say, 20% of the IT-projects taking 80% of the entire budget. Apart from the precise ratios this divide and conquer strategy is a sign of control on the one hand, and a sign of an asymmetric leptokurtic possibly heterogeneous probability density function (PDF) with a heavy right tail on the other hand. A well-known family of heavy tailed PDFs is known as Generalized Pareto Distributions, and this Pareto is the same as the above-mentioned 80/20 rule that is also known as the Pareto Principle. So, we expect to see this for IT-project size when risks are being mitigated successfully in an SPI-program. And if there is adequate control, we must find similar patterns in duration and/or cost. Namely, if we have relatively many (more than you would expect from a normal distribution!) large projects, we have relatively many long durations, and/or relatively many high costs. Of course, over time the important indicators should be correlated somehow: large projects should indeed correspond with larger costs and/or larger durations. Otherwise the reporting system is being gamed, since then the total sum of sizes, durations, and costs are booked to match, so that higher management who only sees the aggregates is satisfied. And of course, the production costs should trend downwards indicating that the SPI-effort is paying off.

So, by accepting the stochastic effects and the heterogeneity of the data, we suddenly become capable of quantifying the unquantifiable. Remains the question of how we would go about doing that.

One could say that in software engineering we suffer from extremal events: the large cost and time overruns, the failing projects, and so on. In real life we all suffer sometimes from extremal events, and we can insure ourselves against some of them. So someone out there possesses a method assuring that the premiums we pay are just about right to make a living as an insurer while being capable of paying for the damage, or do you see insurance companies go bankrupt as soon as a claim comes in? These types of mathematics are known as extreme value theory and heavy tail analysis. We will use them in this paper, and apply them to IT-project sizes, durations, costs, and costs per unit. We refer the interested reader to [19, 26] for introductions and overviews of modelling extremal events.

Next, it is hard to detect correlations between variables over time such as project size, duration, and cost, since there is such a large variation. But, this type of problem is known in many other fields as well. For instance, in finance we all know that stocks vary all the time, and if you want to create a risk-diversified stock portfolio that manages risks over time, you can deploy time series analysis for that. With such theory, it becomes possible to capture correlations, and to some extent predict future behaviour. Stocks also suffer from extremal events, but then often not to the up-side as in cost and time overruns, but to the down-side as in bankruptcy and stock crash. The field of time series analysis and forecasting is the branch of mathematics that deals with processes where noise and extremal events over time are part of the game. We will use this type of mathematics to model important IT-indicators over time. The interested reader is referred to [8] for a beautiful introduction to this type of mathematics.

Finally, we like to detect trends in strongly variating data. We are not the first who want to spot trends in nonconverging, drifting, and variably trending data. The field of

mathematics dealing with such questions is called smooth regression analysis [73], and we will deploy this work to extract overall trends from data with appreciable noise.

Statistical software used Throughout this paper we will meet the just mentioned mathematical modelling techniques, and more. We will illustrate how they can help us in gaining control over the production of software. Fortunately, there is a large amount of computer packages that implement the most recent mathematical and statistical methods, so it will be not too much of a problem to carry out complicated calculations and analyses. In this paper we mainly use Splus [13, 49, 11, 84, 65, 59], and the extension package S+FinMetrics [94, 24, 10] for all our analyses. Sporadically we use a system of the U.S. Census bureau [9], and we use some R functionality [3], an open source variant of Splus.

Case study We discuss a real-world case in this paper: quantifying the effects of a large software process improvement program that is implemented in a large organization building, maintaining, renovating, outsourcing, buying, and retirering lots of software. This SPI-program involved 1500 IT-developers and business people, and comprised of the implementation of the following technologies.

- Implementation of the DSDM method for the 1500 staff members [79, 78].
- All IT-development must be lifted to CMM level 2, and a few departments even to CMM level 3 [64].
- Implementation of peer reviews and Fagan inspections [28, 29, 33].
- Development and implementation of IT-dashboards, balanced scorecards [55], and its corresponding measurement practice.
- Integration of the abovementioned innovations and a cultural change project in order to assure the adoption of the integrated new approach [7, 71]. This comprised among others a professionalization effort for IT-developers in relation with the business inside and outside the organization.

This project costed about 4–6% of the total IT-development budget per year during the SPI-program that started in 2001, and ended in the middle of 2004 [42, 95]. Obviously a lot of money is involved here, and it is only natural to ask the question whether this delivered the promise of IT-productivity improvement. The answer is yes, and this paper shows how to infer such a conclusion.

3 COMPARING FUNCTION POINT COUNTERS

We wish to know how accurate and plausible the function point data is that we obtained in the first phase of this investigation, where we establish the methods and assess the quality of the data. This data is called the *research set* and consists of 193 IT-projects with a total size of 37315 function points. These counts are not estimates of future projects, but counts of the actually delivered functionality. Of course, when you are going to audit whether IT-productivity targets are met or not, you have to exclude the variant that more function points are reported than are delivered, since this would boost IT-productivity. Moreover, we wish to establish how much function points you have

to recount in order to obtain a clear idea of counting accuracy and the overall quality of the counting process. In this paper we will not address the more qualitative aspects such as recounting and comparing the recount reports to the original count reports, but we consider the quantitative aspects.

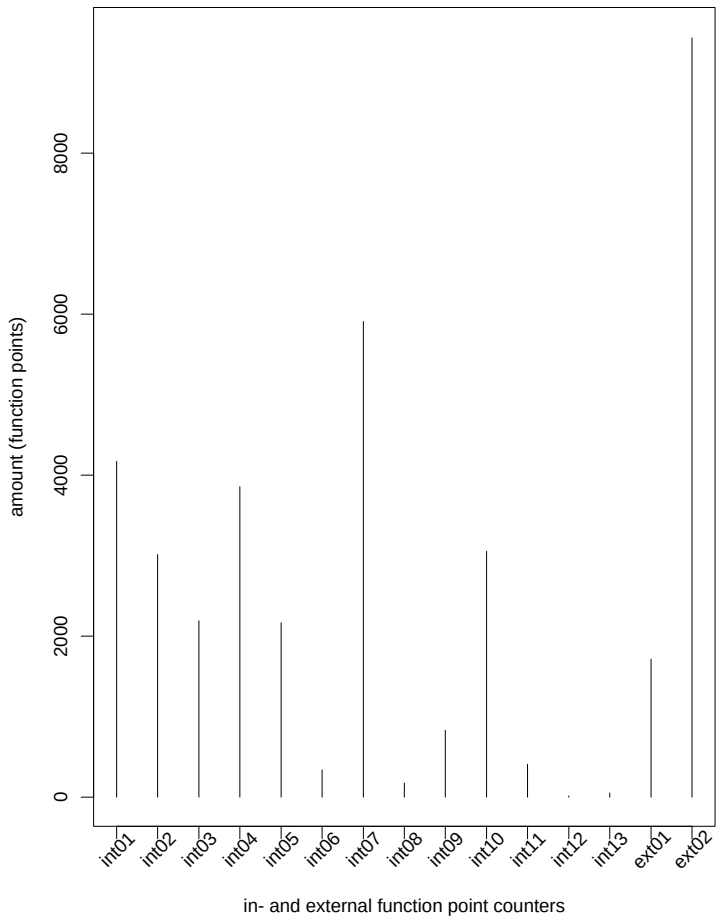

Figure 1: Visualizing the total amount of function points counted per function point counter.

This organization employs 15 function point counters, of which 13 are internal, and two are hired externally from a specialized company doing function point counting only. In Figure 1, we plot the total amount of function points measured by the various counters. The abbreviation int stands for an internal function point counter, and ext is short for an external one. We can spot right away from Figure 1 that some counters do a lot of counting and some do very little counting. For instance, external counter 2 is the star counter: more than 8000 function points. Internal counter 7 is second in row with about 6000 function points. Furthermore, there are a few counters who counted

almost no function points.

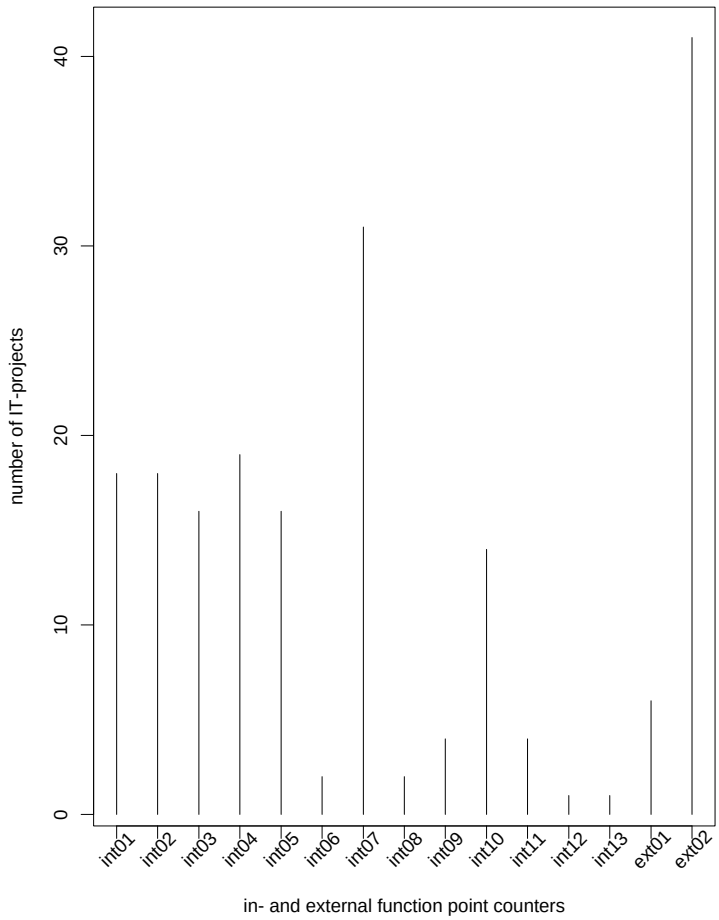

Figure 2: Visualizing the total number of IT-projects per function point counter.

Now that we have an initial idea of the totals per function point counter in the research set, we are also interested to know how many IT-projects each counting specialist took care of. To that end we give another view in Figure 2. This view resembles the one in Figure 1, indicating that there are no clear roles per counter for various sizes of IT-projects. So, it seems not the case that some counter always counts the large projects, and thus counts not too many projects, but a large amount of function points, or vice versa. One could say that these views combined give us some evidence that incoming IT-projects are just assigned to a counter who is available.

To obtain more evidence we construct another view in which we visualize the size-range per counter. We do this via a so-called box and whiskers plot, or boxplot for short [82, 62]. A boxplot is just a visual form to summarize the data with as few points as possible. One well-known point is the median, dividing the data in two equally sized

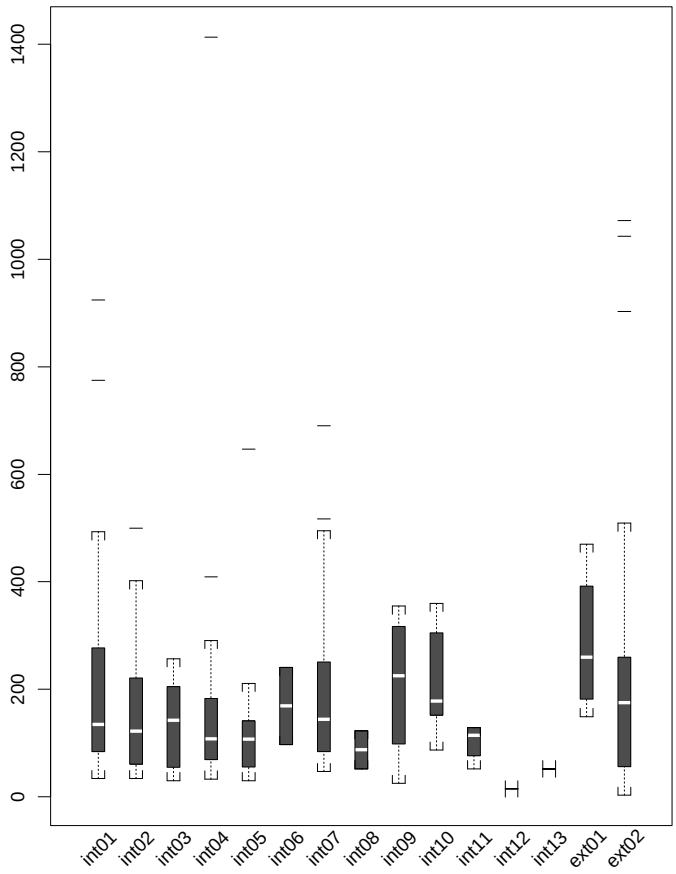

Figure 3: Boxplots of the different function point totals per function point counter.

groups. The other points we use are of the family of quantiles. In general, a *quantile* is any of several ways of dividing your observations into equally sized groups. An example of a quantile is the *percentile*: this divides your data into 100 equally sized groups. Likewise, *quintiles* divide into 5 equally sized groups, and *quartiles* divide data into 4 equally sized groups. You can obtain a fairly good idea of the distribution of your data by dividing it into quartiles [82, 62]. The shaded boxes in Figure 3 are limited by the first and third quartile, and the white line inside the shaded box is the median so that skewness of the data can be spotted right away. So the shaded box encloses the middle 50% of the observed values. Its length is also called the inter-quartile range, which is an important measure that is less influenced by extreme values. The whiskers are some standard span away from the quartiles, we used as standard span 1.5 times the inter-quartile range. Points that go beyond the whiskers are potential outliers and they are drawn individually. If we take the boxplot of internal counter 1, we can see

that the median is not symmetric, and that there is probably a large right-tail: there are many more function points larger than the median in the shaded box than smaller. Moreover, there are a few outliers on top of that. This indicates that we are dealing with a function point distribution that shows signs of control. Furthermore, we can see that the boxplots do not show very strange deviations, except that external counter 1 seems to deliver higher function point totals than others. Maybe this counter just counted somewhat larger projects, or there is a deviation that too high function point totals are reported. All in all, an external counter has no interest in counting too many function points, since it is their profession to count correctly. Another indication is that it seems that external counters count maybe more than internal counters, providing some initial evidence that there is no boosting of function point totals in place, by the internal metrics people.

cntr	#	min	1st qu	med	mean	3rd qu	max	sd	total
i_1	18	34	84	134	232	254	924	255	4169
i_2	18	34	63	122	167	210	500	142	3013
i_3	16	30	58	142	137	203	257	79	2190
i_4	19	33	70	108	203	178	1413	309	3855
i_5	16	30	63	108	135	138	647	146	2166
i_6	2	97	133	169	169	205	241	102	338
i_7	31	47	86	144	190	250	690	157	5906
i_8	2	52	70	88	88	105	123	50	175
i_9	4	25	135	226	208	298	355	143	831
i_{10}	14	87	156	178	218	297	360	89	3053
i_{11}	4	52	89	114	102	128	129	36	410
i_{12}	1	15	15	15	15	15	15	NA	15
i_{13}	1	52	52	52	52	52	52	NA	52
e_1	6	149	198	260	286	363	470	124	1713
e_2	41	3	56	175	230	260	1072	250	9429
all	193	3	73	144	193	244	1413	197	37315

Table 1: Nine point summary statistics on a per counter basis and the same for all the function points together.

After this initial visualization of the potential form of the PDFs we want to know these PDFs for real. Therefore, we further zoom in on the distribution of the function point totals per counter. As a first quantitative indication, we summarize for each counter a nine-point summary statistic in Table 1. The abbreviations in Table 1 are self-explanatory, except sd which is short for standard deviation. The in- and external counters are abbreviated with i and e with a subscript. From Table 1, we can see that function point counters i_6, i_8, i_{12}, and i_{13}, only rarely counted function points. We quantified now much more concrete how the data is skewed, for instance by the sometimes large differences between the median and the mean. But we gain the most insight in a plot of the various density functions. So we calculate the empirical probability density functions (PDFs) both on a per counter basis, and for the overall distribution. For the 4 counters spotted in Table 1, it is not a good idea to develop their probability density function per counter since there is not enough data for them. We keep those four in for an overall estimate of the probability density function, but leave them out in a per counter estimate. In Figure 4 we depicted 12 plots of the empirical probability

density functions of the 11 significant counters and the last one of all counters together (this one is labeled *all*). We used a technique where a non-parametric estimate of the probability density of the data is calculated. In fact this is a smoothing operation on a histogram. For more information on this topic we refer the interested reader to [91, 74].

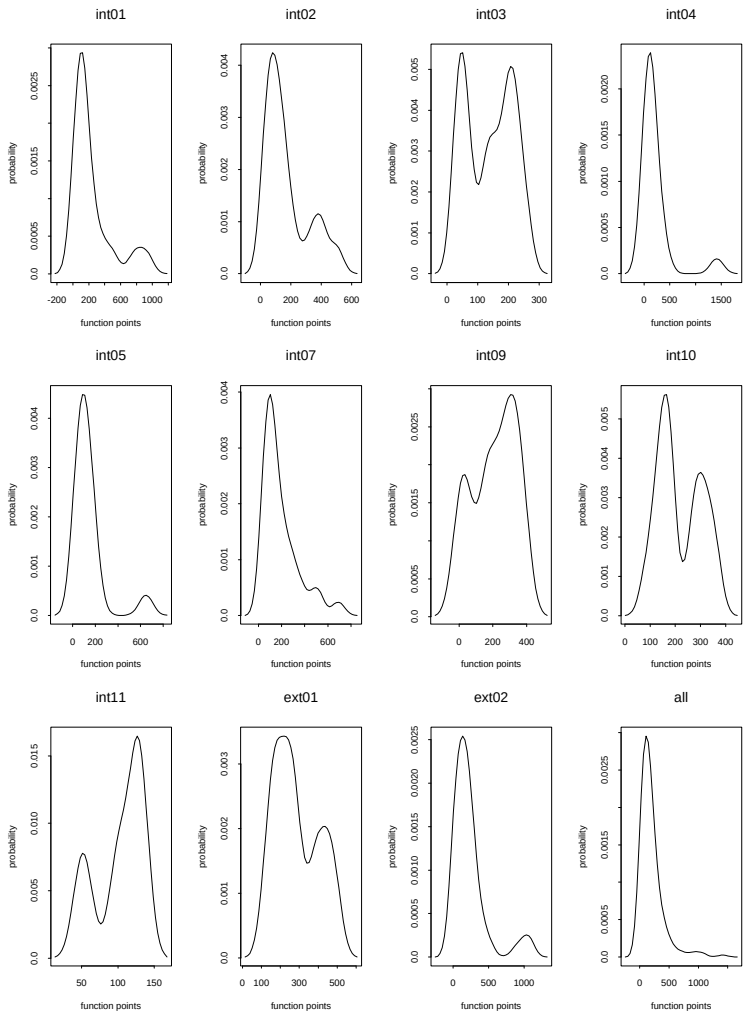

Figure 4: Probability density functions for 11 significant counters, plus an overall probability density function for all the counted function points.

From the 12 plots we can learn several things. First we see that there seem to be two tops in almost all plots, this is a sign of heterogeneous data. But for the plots of counters that count a lot, like internal counter 7 and external counter 2, we observe that the second top is much lower, indicating that the tops in the other density plots are embryonic heavy tails to the right. Indeed, if we inspect the overall density plot 12, we find that this is an asymmetric leptokurtic PDF with a heavy tail to the right. This is an important finding, since the outliers to the right indicate control rather than chaos. From chaos we would expect on the basis of the Standish data that there is a heavy

left-tail, since then there will be many outliers to the left, due to endemic solutions underdelivery [51, 37, 38, 39].

$p > 0.10$	*	No evidence against H_0: data seems consistent with H_0
$0.05 < p \leq 0.10$	**	Weak evidence against H_0 in favor of the alternative
$0.01 < p \leq 0.05$	***	Moderate evidence against H_0 in favor of the alternative
$0.001 < p \leq 0.01$	****	Strong evidence against H_0 in favor of the alternative
$p \leq 0.001$	*****	Very strong evidence against H_0 in favor of the alternative

Table 2: Symbolic notation for various ranges of p-values with qualitative explanations of their meaning.

Next we want to understand whether the density functions we depicted in Figure 4 are really different from each other. of course, when we look at the density functions we spot differences, but how different need these differences be before we would say that the function point counters are actually counting different? In order to find out we use a formal test.

Since in general we do not know anything about the distribution of the data, it would be best to use a test where this does not matter, a so-called distribution-free test. One such test is known as the Kolmogorov-Smirnov goodness of fit test, or KS-test for short [58, 76, 20]. The KS-test measures the maximal vertical distance between the cumulative distributions (CDF) of two data sets. So, when we would plot the CDF version of the plots in Figure 4, the test measures the maximal vertical distance between two such CDFs. This distance is known as the KS-test statistic. If this value is very small, this is an indication that both CDFs, and thus both PDFs are not fundamentally different. A related metric to indicate how strong the evidence supporting this hypothesis is the p-value. In Table 2, we introduce symbolic notation indicating in qualitative terms which evidence range is meant by which p-value. For instance, when a p-value is smaller than 0.001, there is very strong evidence that the hypothesis is not true, in favor of the alternative.

H_0	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	e_1	e_2
i_1	-	*	*	*	*	*	*	*	*	*	*	*	*	*	*
i_2	*	-	*	*	*	*	*	*	*	***	*	*	*	**	*
i_3	*	*	-	*	*	*	*	*	*	**	*	*	*	**	*
i_4	*	*	*	-	*	*	*	*	*	***	*	**	*	**	*
i_5	*	*	*	*	-	*	*	*	*	****	*	*	*	****	***
i_6	*	*	*	*	*	-	*	*	*	*	*	*	*	*	*
i_7	*	*	*	*	*	*	-	*	*	**	*	**	*	**	*
i_8	*	*	*	*	*	*	*	-	*	**	*	*	*	**	*
i_9	*	*	*	*	*	*	*	*	-	*	*	*	*	*	*
i_{10}	*	***	**	***	****	*	**	**	*	-	****	*	*	*	*
i_{11}	*	*	*	*	*	*	*	*	*	****	-	*	*	****	**
i_{12}	*	*	*	**	*	*	**	*	*	*	*	-	*	*	**
i_{13}	*	*	*	*	*	*	*	*	*	*	*	*	-	*	*
e_1	*	**	**	**	****	*	**	**	*	*	****	*	*	-	*
e_2	*	*	*	*	***	*	*	*	*	*	**	**	*	*	-

Table 3: Comparisons of individual function point counters with each other.

In Table 3, we summarize the results of carrying out 105 KS-tests, comparing the CDFs of all individual function point counters against each other, except themselves, hence 105 KS-tests. The null hypothesis, denoted H_0, is that both CDFs based on the

data coincide. For instance the first row of Table 3, shows that all the KS-tests are in the single-star category. This means according to Table 2 that there is no evidence to reject the null hypothesis. In other words, there is no evidence, based on this formal test, that the counting practice of counter i_1 differs from the counting practices of any of the other function point counters. Note that we made Table 3 symmetric, so that you can easily analyse the results (so the first column is the same as the first row). Internal counter 10 differs in 7 cases from other counters. Furthermore, external counter 1 differs in a number of cases. The more stars, the stronger the evidence that the differences are due to different counting practices. We are not too surprised that external counter 1 differs, since we already noticed from the boxplots in Figure 3 that this counter is at the high end in function point totals. It also seems that internal counter 10 did relatively many outlier counts, given the large top in plot 8 of Figure 4. All in all, these differences give us input for a more qualitative analysis to investigate why both counters seem to differ from other counters.

p-value category	notation	meaning	amount	% of total
single star	*	$p > 0.10$	86	81.9
double star	**	$0.05 < p \leq 0.10$	12	11.4
triple star	***	$0.01 < p \leq 0.05$	3	2.86
quadruple star	****	$0.001 < p \leq 0.01$	4	3.81
quintuple star	*****	$p \leq 0.001$	0	0
Total number of combinations			105	99.97

Table 4: Summary of the various ranges of p-values found in the individual comparisons between function point counters.

In Table 4, we summarized the various p-value categories for the 105 KS-tests. As can be seen, 86 of the 105 combinations do not show evidence of differences, which is about 82% of the combinations. Then we see that 11.4% of the combinations show weak evidence that there are differences in counting practice, and in 6.7% of the cases there is moderate to strong evidence that there are differences. Note that for the internal counters we have in 69 of the 78 cases no difference, which is 88.5%, and in 11.5% of the cases there are various ranges in evidence for differences. This is quite a good result, and if we take the total amount of counted function points into account for the combinations where no differences were found using the KS-test, the results are very favorable as well.

3.1 Comparing counter groups

In our first analysis we have seen, that there seem not too many differences between individual counters, with a few exceptions. In order to come up with an answer on the question whether a software process improvement project resorted the desired effect, we have to exclude the following possibility that we already alluded at. Suppose the internal counters have a vested interest in counting more than is produced. In this way, productivity can turn out to be higher, while in effect this is not the case. To that end, we analyse the data of the function point counters in two groups. The group of internal counters, who might have that interest, and the group of external counters, whose profession it is to count the correct amounts. If they deliver erroneous results they will be out of work, and litigation is then unavoidable, since erroneous contracts

would then be closed on the basis of their results.

In Figure 5, we plot the cumulative distribution functions of the group of internal counters and the group of external counters. The plot shows that the external counters are counting uniformly less (except at the smallest sizes) than the internal counters. Indeed if we carry out a formal test, using the Kolmogorov-Smirnov goodness of fit test, we find that the KS-test statistic is 0.2472, and the p-value is 0.0075, which is a strong indication that both distributions are not the same (see Table 2). This means that there are counting differences, when compared to the group level. The external counters are counting more strict than the internal counters, so there is an indication for a counting practice that might game the system.

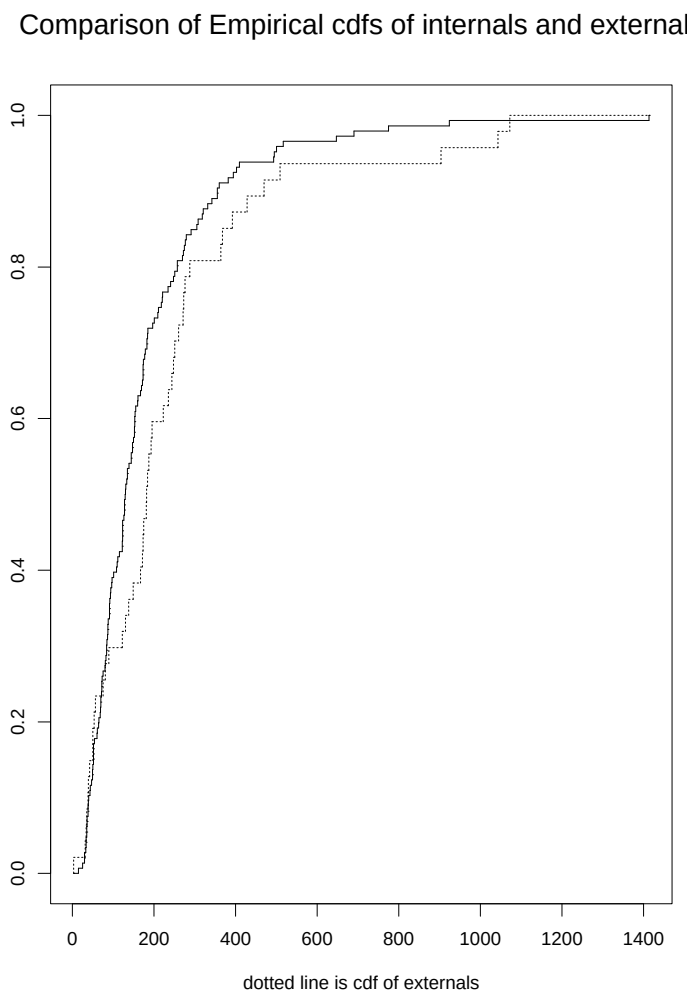

Figure 5: Comparison of CDFs of internal and external function point counters.

To be sure what is happening, we need to dive further into this issue. What could be the cause of this group difference, that was not visible at the individual level? To that end we carry out some more tests. Suppose we would throw out all the internal

counters that counted only one or two projects (see Table 1). In Figure 6 we depicted a comparison between the distributions of the rest of the internal counters versus the external ones. The plot shows as well, that there are differences. And a formal test using the Kolmogorov-Smirnov goodness of fit test, gives a slightly better result: 0.2384. So, the maximal vertical distance between both CDFs decreased a bit. Also the p-value is a bit larger: 0.0112, and enters the triple-star category. But still there is moderate evidence that both distributions are not the same. So we are tempted to reject the possibility that the group differences are caused by inexperienced counters, who counted only a few projects.

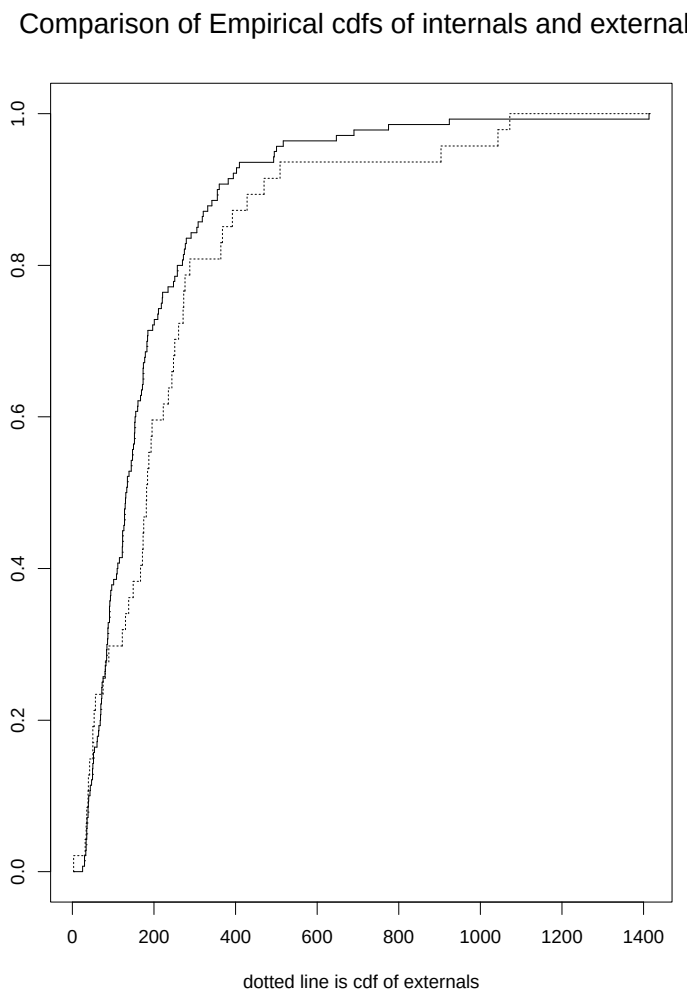

Figure 6: Comparison of CDFs of internal counters with more than 2 IT-project counts and external function point counters.

To be absolutely sure that size, and thus experience is not the cause, we throw out all counters under 500 function points in total. In Figure 7 we depict a comparison of both CDFs. Again, we can see from this plot that there are differences, and the KS-test

statistic is the largest of the previous cases: 0.2646, the p-value is: 0.0034, so definitely in the four-star category, and provides thus strong evidence that both distributions differ. In other words, it is unlikely that the amount of function points counted, or a low number of IT-projects counted is a clarification for the differences in counting practice.

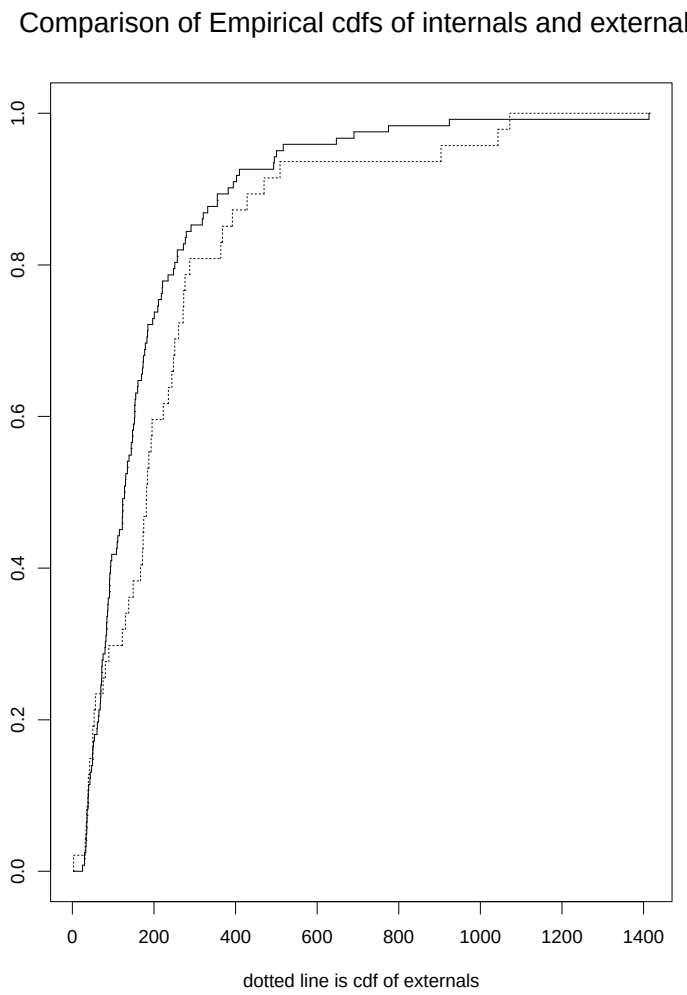

Figure 7: Comparison of CDFs of internal counters with more than 500 counted function points versus the external function point counters.

Now we wish to know whether the differences could be due to a single counter, internal, external, or a combination of both. To that end, we carried out a number of Kolmogorov-Smirnov goodness of fit tests, that are summarized in Table 5. From this table, we can learn that there is no single person that can clarify the difference in counting. If we look at pairs, there is only one pair that stands out: if we omit internal counter 5 and external counter 1, the distributions of the rest of the internal and external counters are no longer significantly different. Indeed in Figure 8 we depicted both cumulative distributions. Recall from Table 2 that a single star means that there is

no evidence that the null hypothesis does not hold, in other words, the assumption that the distributions of the internal counters minus the counts by number 5, are the same as counting by external counter 2. A Kolmogorov-Smirnov goodness of fit test gives these numerical results: The KS-test statistic is 0.1931, being the maximal vertical distance between both curves of Figure 8. The corresponding p-value is: 0.1062, indicating that there is no evidence to reject the null hypothesis in favor of the alternative.

H_0	p-value	H_0	p-value	H_0	p-value
$-i_1$	****	$-(i_1, e_1)$	**	$-(i_1, e_2)$	***
$-i_2$	****	$-(i_2, e_1)$	**	$-(i_2, e_2)$	***
$-i_3$	****	$-(i_3, e_1)$	***	$-(i_3, e_2)$	***
$-i_4$	***	$-(i_4, e_1)$	**	$-(i_4, e_2)$	***
$-i_5$	***	$-(i_5, e_1)$	*	$-(i_5, e_2)$	**
$-i_6$	****	$-(i_6, e_1)$	***	$-(i_6, e_2)$	***
$-i_7$	****	$-(i_7, e_1)$	***	$-(i_7, e_2)$	***
$-i_8$	****	$-(i_8, e_1)$	**	$-(i_8, e_2)$	***
$-i_9$	****	$-(i_9, e_1)$	***	$-(i_9, e_2)$	***
$-i_{10}$	****	$-(i_{10}, e_1)$	***	$-(i_{10}, e_2)$	**
$-i_{11}$	***	$-(i_{11}, e_1)$	**	$-(i_{11}, e_2)$	***
$-i_{12}$	****	$-(i_{12}, e_1)$	**	$-(i_{12}, e_2)$	***
$-i_{13}$	****	$-(i_{13}, e_1)$	**	$-(i_{13}, e_2)$	***
$-e_1$	**	-	-	-	-
$-e_2$	***	-	-	-	-

Table 5: Results of 41 KS-tests where single counters or pairs of counters are excluded to detect potential causes for differences in counting practice between in- and external function point analysts.

Is this plausible? When we analyze the boxplots of both counters further as depicted in Figure 3, we can see that internal counter 5 is entirely at the low side, and external counter 1 is at the high side compared to all counters. In addition, note that in Table 5 the omission of external counter 1 leads uniformly to a shift from each p-value: each p-value with this external counter shows more difference than without. So, the impact of both counters is to be interpreted as follows: internal counter 5 counts relatively sober, and external counter 1 counts relatively rich. Their combination in the groups of counters leads to a significant difference. Of course, this can be coincidentally the case if both counters got small respectively large counting jobs. Fact is that without the two, there is no significant difference between the internal and external groups, while still about 40% of the function points is counted by externals, and 60% is counted by internals.

So, if we wish to analyse the data without the effect that external counters count different from internal counters, we should filter both counters from the data set. On a total of 37315 function points counted, this amounts to 33436 function points left, so we could decide to remove 3879 function points, leaving almost 90% of the data for further analysis. In case of an IT-audit, it would then be adequate to recount 10% of the function points by a third independent party. This recounting was carried out and turned out not to reveal any systematic difference in counting practices.

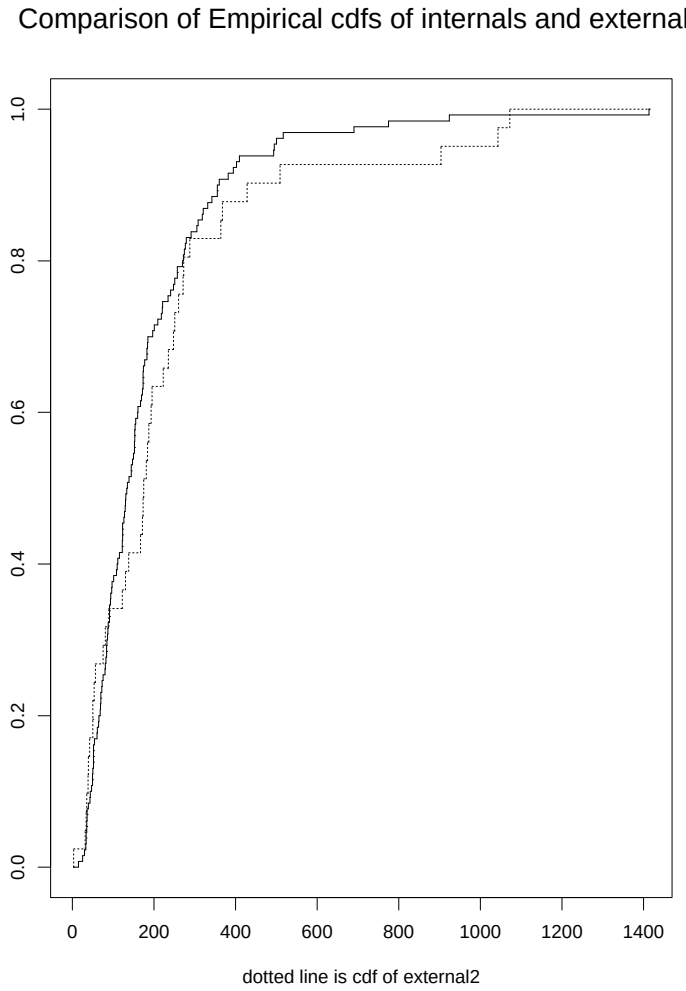

Figure 8: Comparison of the CDF of all internal counters except counter 5 with the CDF of external counter 2.

Summary So what we showed in this section is that there is no significant difference between almost any of the internal counters. To be precise out of the 78 combinations of internal counters, in 57 cases no difference could be spotted (73.1%), there were 5 combinations with weak evidence that there were differences, of which counter 12 counted only one system (two double star hits). Then there were 2 cases of moderate evidence, and 2 cases of strong evidence, all due to counter 10 who counted 3053 FPs in 14 systems. This implies in terms of function points counted that for more than 88.3% of the function point totals, there is no evidence that the internal function point counters are significantly different.

In summary, inside this organization the function point counting practice is state-of-the-art when it comes to interrater reliability: in about 75% of the cases no evidence is available showing any difference in counting practice, and the differences that are

found, concentrated around one counter. Furthermore, at least 90% of the counted function points are covered by counters where no evidence is found that their counting practice is distinct.

Of course, we found group differences, and we also found a way to deal with them, if it is needed to weed out the differences for further analysis. But for this analysis, it turned out that the external counter 1 counted relatively rich and internal counter 5 relatively strict, so that the effect of (un)willfully counting too much function points to boost productivity gains can savely be discarded. Moreover, a qualitative analysis by recounting about 10% of the function point totals confirmed our statistically inferred conclusions.

3.2 Intermezzo

Although the main focus of this paper is not a study on the reliability of function point counting, substantial research has been done on that in the past. In this paper we quantify the effects of software process improvements, and to be sure that there is an improvement, we need to know more about the reliability of the data on which these improvements are based. As an intermezzo, we discuss our reliability check with the research being done in the past.

In [57, 56], an extensive field experiment was conducted to address the questions of interrater reliability and intermethod reliability. All function points that were counted by our 15 counters used the same method: an approved adaptation of the IFPUG-standard. So in this case we did not address the intermethod reliability question. Our focus here concerns the interrater reliability. In [57, 56], the experiment set out was to have different counters analyse the same system, and test statistically whether the outcomes were the same. This turned out not to be the case, but an interrater reliability of about 12% was found in that experiment.

Our research design is different: we have not asked counters to carry out an FPA of the same system, but we just have about about 200 IT-projects with corresponding function point totals, delivered by the 15 counters. Some of the criticism on function points is that when two counters estimate the function point totals, the answers may be different. In fact, [57, 56] showed that this difference can be as high as 12%. As we see it, function point counting is a stochastic process, inevitably leading to differences. Rather than trying to ban the differences, it is in our opinion better to recognize the stochastic nature of function point analysis, and take that as the fundamental viewpoint on function points. In other engineering disciplines the stochastic nature of certain processes is not only recognized, but is used as a basic tool to construct systems. Let us explain. When van Doorne's transmissions invented the Variomatic (an automated kludge for automobiles), this consisted of a number of metal bands that fit very closely together. While in the lab situation the researchers could produce small numbers of the kludge, it turned out to be a problem to industrialize it. Apart from oven heating problems that disturbed production start-up, it was also close to impossible to deliver the various metal bands with ultra high precision. The solution was not to improve the reliability of the production process, but the other way around. You just make a large number of metal bands, and after production you collect them, measure them, categorize them, and construct a perfect fitting transmission system. So high tolerance, low cost, and high precision can go hand in hand in other engineering disciplines. Therefore, we should not abandon imprecise metrics as useless. We better recognize stochastic effects and if possible exploit their properties. We think that there are such opportunities in software engineering, of which this paper testifies.

Our research design to test whether the function point counters are reliable is done in a stochastic manner. We give another example to illustrate this. Suppose we would want to find out whether two dice are the same. Then by throwing the dice, we should not conclude that the dice are unreliable if the outcome of the dice is different. We would say that the dice are different if the *probability* of a certain outcome differs between the two dice. Exactly the same, we look at the function point counters. Some stochastic process produces a string of function point totals out of a universal of IT-projects that need to be counted. We are not interested in the question whether the exact outcome will be given if we provide one system to all counters, but we want to know whether the probability that they give the *same* outcome is not different. And by recognizing the stochastic nature of function point counting, this becomes in fact easy. We used the formal KS-test for that, giving us insight in how much different outcomes may be while still knowing with a certain accuracy that these outcomes are not caused by different "FPA-dice".

4 EXPLORING FINANCIAL DATA

We have an initial idea of the function point data by now, so we move on to exploring another important indicator: costs. The research set contains the planned and actual costs, so we could investigate whether the plan accuracy of the costs is somehow improving over time, which is a sign of the successful implementation of an SPI-program. As we will see in this section, the estimates and actuals largely coincide, which means that most probably, the estimates are retrofitted. A simple further qualitative analysis confirmed this. Often the reason for doing this is not malicious, but part of a financial tradition, namely to adapt the figures as soon as actuals come in. For financial reporting this is probably a good thing to do, but for assessing the accomplishments of an SPI-program it were better if the original estimates had not been destroyed.

So we cannot use the estimates of the costs for assessing the success of the SPI-effort. We will use the actual costs only for that. Since we still want to have an idea of the plausibility of the costs, we connect the costs with their sizes, and use industry benchmarks to find out whether the costs with their sizes are plausible. This is an indirect check for plausibility of the cost data. Of course, precise cost data is sensible information, and cannot be reported on directly. Throughout this paper we will report on costs, and related metrics such as costs per function point by means of an index. The index hides the precise costs, but exposes identical patterns as the costs and related metrics.

Both planned and actual costs were reported, we will visualize both. In Figures 9 and 10, we display a histogram, a boxplot, a density plot, and a Q-Q plot of the actual cost index and the planned cost index respectively. A histogram is the grouping of data into bins plotting the number of members in each bin versus the bin range. In plot 1 of Figure 9 we depict a histogram showing signs of an asymmetric heavy-tailed distribution. The boxplot was explained earlier. What can be seen from plot 2, is that there are many outliers, confirming the heavy tail to the right. Plot 3 displays the empirical probability density function, which indeed shows a leptokurtic asymmetric heavy-tailed shape. Finally, we plot the empirical quantiles against the quantiles of the standard normal distribution. If two data sets have the same distribution, there must be a linear relation between both their quantiles (see e.g., [48, p. 244]). You can visualize that with a so-called Q-Q plot, where Q-Q stands for quantile-quantile. Since plot 4 does not show a straight line, we have a strong indication that the data is not normally

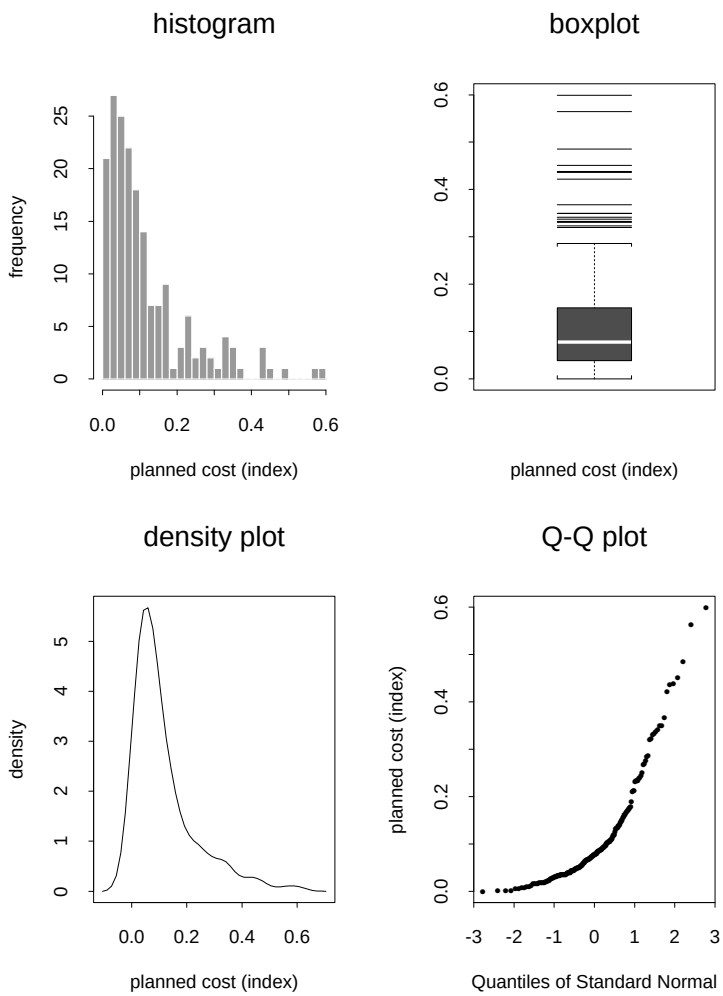

Figure 9: Visual insight into the empirical distribution of the planned cost index.

distributed. This four panel view of data gives us quickly insight into the nature of the data. If we compare the plots in both Figure 9 and 10, we see that there is a strong resemblance between all the plots.

To find out more, we analyse the potential relation between planned and actual costs further. To that end, we start out with a Q-Q plot of the actual cost index versus the planned cost index. In the planned costs there were a few data points missing (< 10). The Q-Q plot shows a clear linear connection between the quantiles of both indices, except for a few higher quantiles. This implies that we have a strong indication that their distributions are the same. And this means that the "cost-dice" behave the same. This is suspicious, since it is known that we can hardly plan costs that accurately. In fact, a measure for the quality of an estimate is the so-called EQF, short for estimating quality factor [22]. The relation between planned and actual costs implies an EQF approaching infinite, whereas an EQF above 10 (which means just 10% off) has never

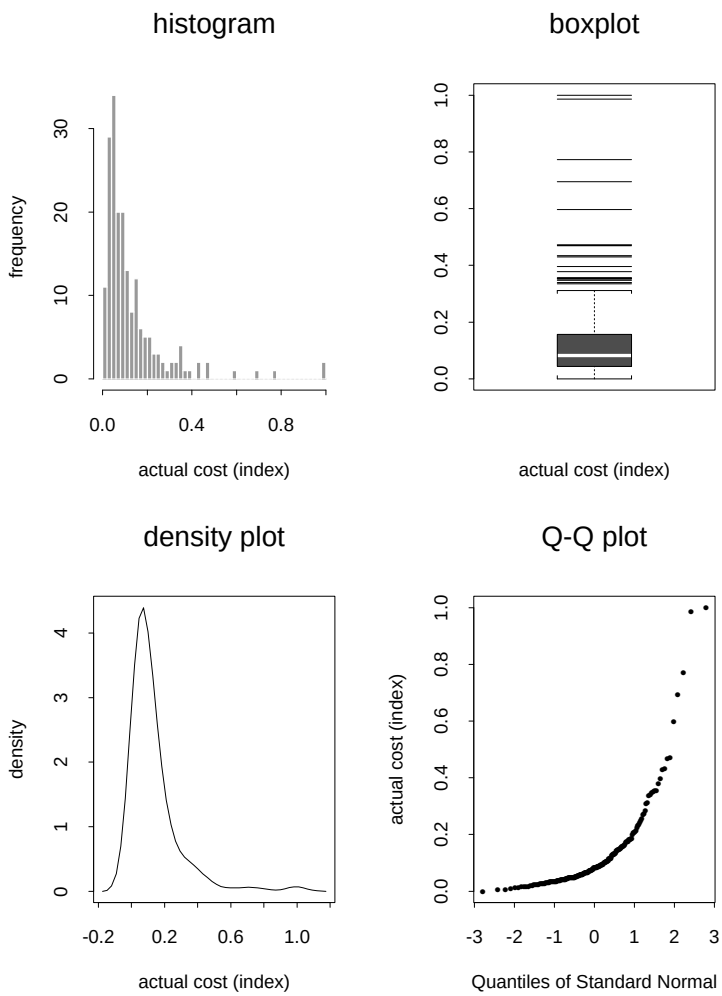

Figure 10: Visual insight into the empirical distribution of the actual cost index.

been observed by the proponents of the EQF-methodology [60].

For a further visual analysis, we compare the cumulative distribution functions of planned and actual costs in Figure 12. This shows a striking resemblance: both CDFs seem to overlap perfectly except for a few values, adding to our conjecture that the data is retrofitted. This resemblance makes us decide to carry out a formal statistical test: the Kolmogorov-Smirnov goodness of fit test [58, 76, 20]. With this test we can see whether the distributions of the actual cost index and the planned cost index differ. Indeed, the KS-test statistic gives: 0.0683 meaning that the maximal vertical distance between both CDFs is at most 0.0683. The chance that this is a coincidence is rather small and the p-value gives 0.7168, indicating that there is no evidence that both distributions differ (see Table 2). This implies that either the planned estimates are miraculously accurate, or that there is an error in the reporting process. After inquiring with the reporting department, it turned out that planned estimates were replaced

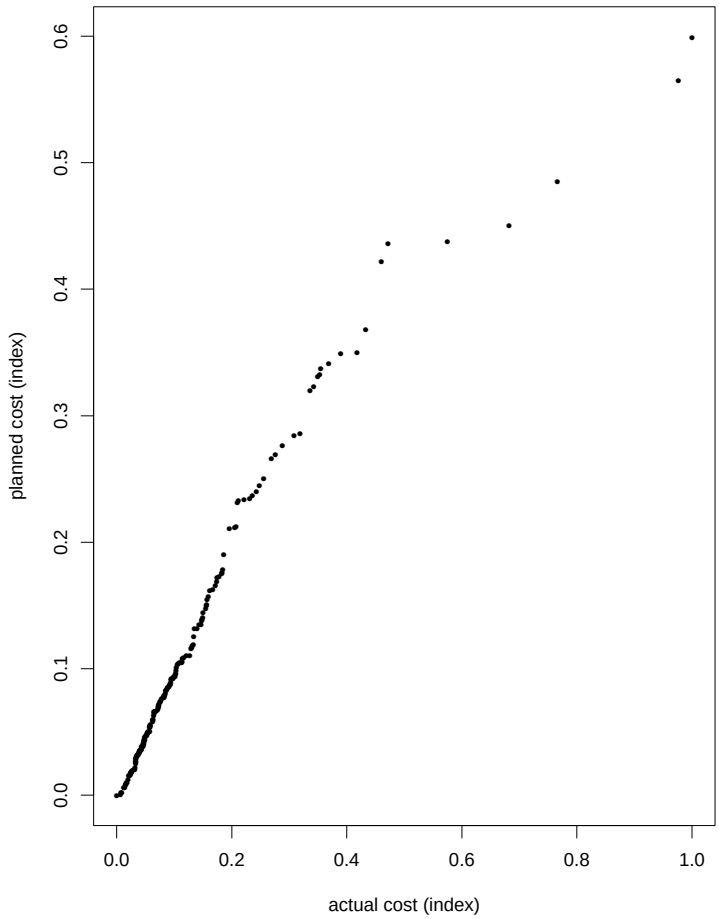

Figure 11: Q-Q plot of the actual cost index versus the planned cost index.

with the actual costs. This is not a good idea, since real estimates and actuals can now not be analyzed in order to assess improved plan accuracy for budgets—a sign of a successful SPI-implementation. The process needs a change here so that the estimated data is recorded for that purpose. Of course, then the estimators need to get feedback, otherwise this will not work. In [22], you can find ways to establish successful measurement programs, and for more information on characteristic data patterns that reveal retrofitted data and other effects we refer to [86].

Our conclusion is that it is useless to use the planned cost index any further. We will use the actual cost index in the sequel of this paper, and if we omit the word actual, it is tacitly implied.

Comparison with benchmarks Next we want to check if the cost and size are somehow related in a reasonable manner. But before we do this, we want to know whether

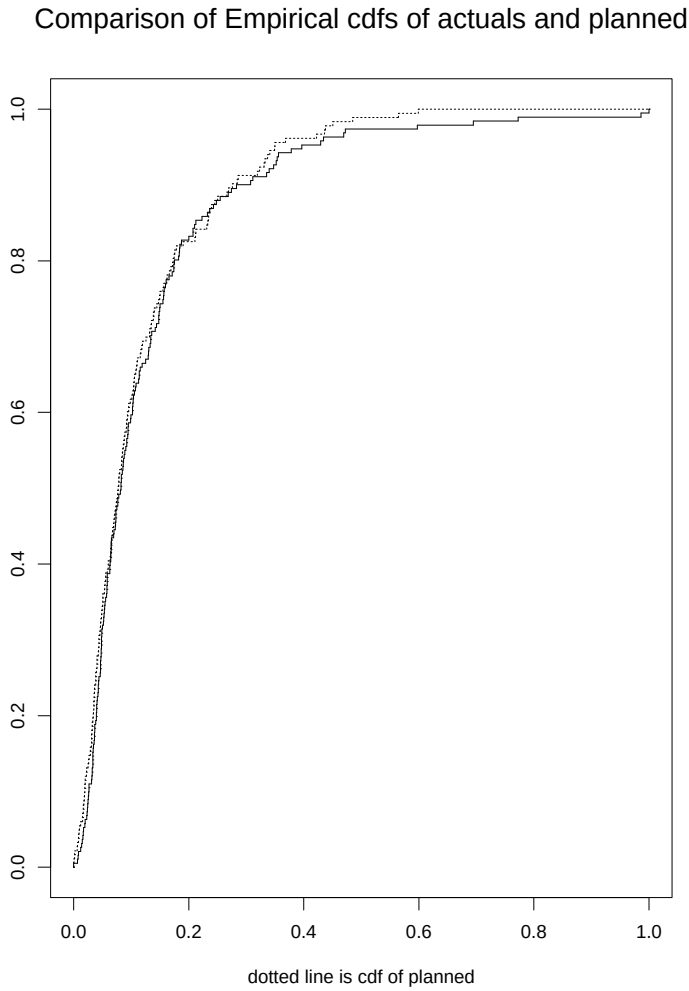

Figure 12: Comparison of the cumulative distribution functions of the actual cost index and the planned cost index.

the size displays similar distribution properties as the cost index. Recall that in Figure 4 we already showed a first small plot of the PDF for the function point sizes. In Figure 13 we give a more elaborate 4-plot overview of the distributional properties of the function points, and we see that the histogram alludes to a heavy right tail. The boxplot shows outliers, not as many as the costs, but enough to confirm a heavy right-tail. Indeed the empirical PDF of the function point sizes displays in plot 3 a heavy right-tail. The Q-Q plot also shows no straight line, so there is strong evidence that the distributional properties of the function point sizes and those of the costs are similar. This needs a further and deeper analysis, which we will do in Section 5. For now it suffices that there is no strong evidence that the distributional properties of the costs are structurally different from those for the function points (for instance, that there is a heavy left-tail).

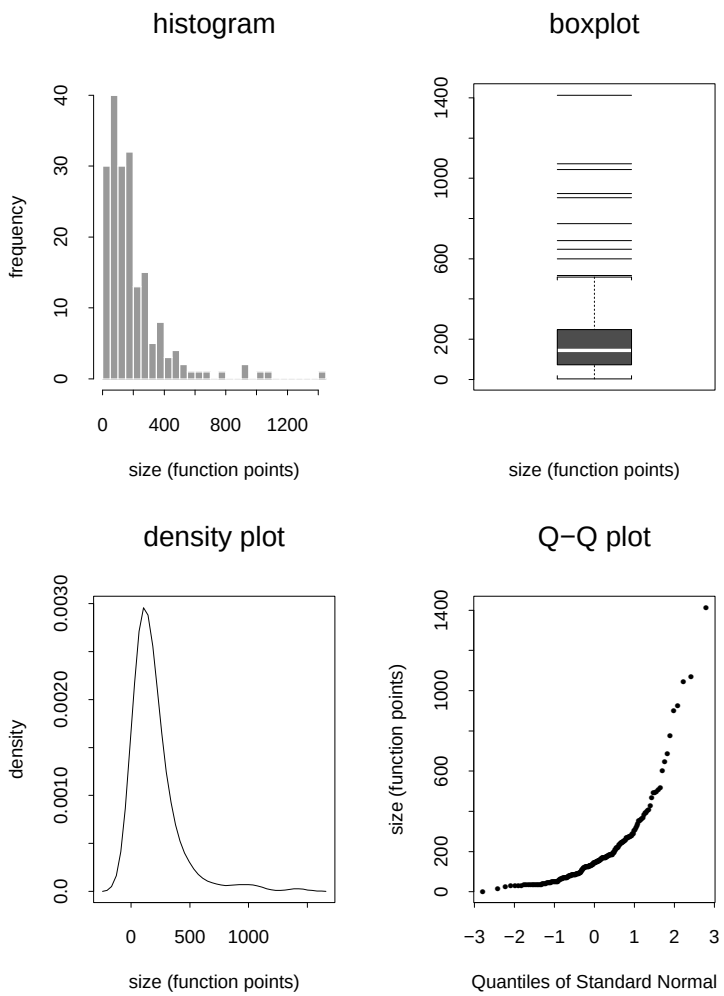

Figure 13: Visual insight into the empirical distribution of function point sizes of the IT-projects.

Now we commence with the cost-size relation. The cost index should be positively related to size, in the sense that larger sizes should lead to higher costs, and vice versa. In Figure 14, we depict a cost-size view. The dots are IT-projects: their cost index vertically, and the corresponding sizes horizontally. The solid line is an industry benchmark based on the assumption that these are all in-house development MIS projects (taken from [85]). The benchmark looks as follows:

$$tcd_i(f) = \frac{rw}{12} \cdot \frac{f}{p_i(f)} \quad (1)$$

$$p_i(f) = 1.627 + 38.373 \cdot e^{-0.06222733 f^{0.424459}} \quad (2)$$

where f is the size of an IT-project in function points, r the fully burdened daily

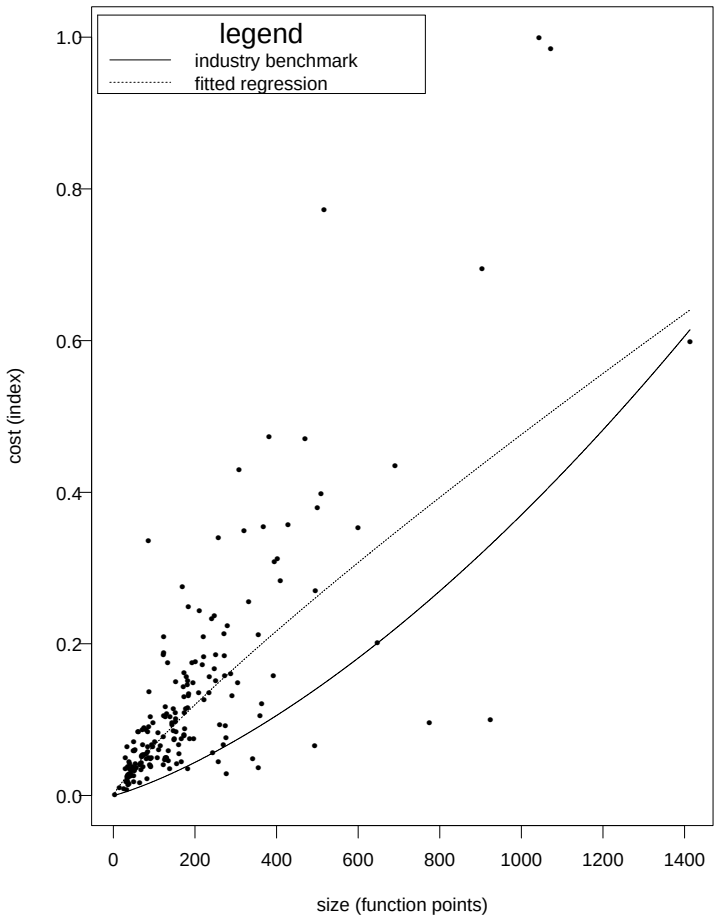

Figure 14: Cost-time view of the IT-projects, an industry benchmark, and a nonlinear regression forming an internal organization-specific benchmark.

rate, for instance in dollars or euros, and w is the number of working days per year. The abbreviation *tcd* stands for total cost of development, and formula 1 makes use of formula 2, where the p stands for productivity. Both subscripts i stand for in-house development. Formula 2 is statistically fitted to benchmarks for productivity of in-house MIS development [52], hence the strange constants. For more information on both formulas and their inference, we refer to [85].

We instantiated these formulas with the actual number of days per annum that IT-developers work, and the actual daily rates of the organization in question. We note that there are also maintenance, renovation, enhancement, outsourced, and retirement projects in the IT-portfolio, so the above benchmark formulas are likely to provide mixed results. We use this benchmark for the purpose of light-weight checking whether the actually reported costs are somehow correlated with the sizes. And this is the case:

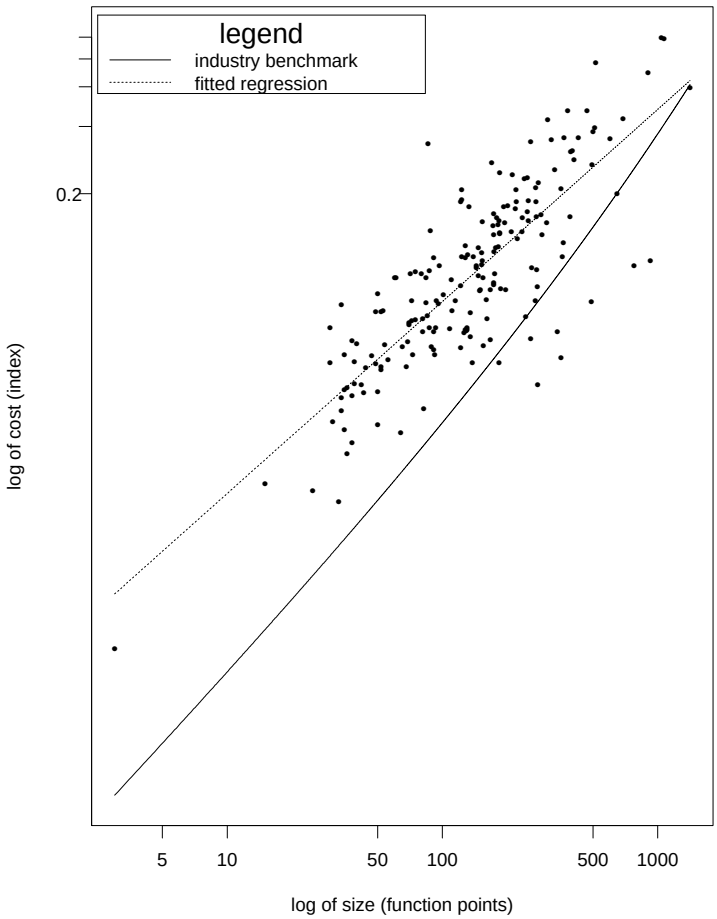

Figure 15: Log-log view of Figure 14.

apart from the fact that the benchmark-line is below most dots, a translation of this line fits the data somewhat better. This is best seen in Figure 15, which is just a log-log view of Figure 14. The property of a log-log view is that numbers with the same order of magnitude clutter together, which gives more insight into potential relations that are nonlinear. Indeed, if we look a bit closer to Figure 15, we see that the solid line, which is our industry benchmark, is systematically off the data, which is a good sign that the cost and size are reasonably well related. In order to gain a bit more precision we carried out a nonlinear regression through the cost-size data, which is the dotted line. This dotted line is in fact an internal organization-specific benchmark that can be used in future estimates. When the size is known, the costs can be predicted using the internal benchmark. We will not depict the formulas for confidentiality reasons, but it consists of formulas similar to the above formulas 1 and 2, but then with different coefficients. This internal benchmark has a correlation coefficient of 66.2%, so not

terribly accurate, and predictions need to be supplemented by different means as well (think of activity-based cost estimates). In any case both the industry and internal benchmarks are good enough for our purpose: to check whether there are reasonable relations between cost and size. The relations are present, and they provide no strong evidence that the cost data is not plausible.

5 HEAVY TAIL ANALYSIS

In the previous section we already found some spurious data (the planned costs), and we reassured ourselves via industry benchmarks that the actual costs are showing no signs of implausibility. In this section we want to dive deeper into the distributional properties of the important indicators size, duration, cost, and cost per function point, to further investigate the distributional properties, and thus plausibility of the data in the research set.

At this point we have concluded that the function point totals are as reliable as it can get: for about 90% of the counts there is no difference between counting practices to measure. Moreover the function point data shows heavy right-tails, which is a sign of control, as indicated earlier. So from now on we take the data for the function points as a basis to analyse relations between this and the other reported data (just as we did via benchmarks in the previous section). We analyse the distribution of the function points, and the properties of that distribution should somehow be reflected in the related variables: duration, actual cost indices, and the cost index per function point. This relation does not need to be direct in the sense of analytic correlations, such as highly correlated formulas. There are namely many unknown random effects in place. Therefore, we will explore whether the stochastic properties of the function point data are also found in supposedly related other IT-metrics residing in the research set.

variable	dispersion
function points	6.195733
duration	4.425686
actual cost indices	5.783603
cost index per function point	7.689344

Table 6: The important indicators and their dispersion measured in the number of standard deviations away from their mean value.

One important property of the function point data is easily illustrated informally. Suppose the function point data would be more or less normally distributed. In case of normality, 68% of the data are at most 1 standard deviation away from the mean, 95% of the data are at most 2 standard deviations away, and 99.7% of the data lie within 3 standard deviations of the mean. This well-known property of normal distributions can be used as a first check. We found that for the function points there are values more than 6 standard deviations away from the mean. So the function point data are most probably not normally distributed, since the tail of the distribution is too heavy. The related variables show similar patterns, which we summarized in Table 6, and are all not normally distributed. Therefore, we will carry out a so-called heavy tail analysis, usually applied in extreme value analysis [19, 26].

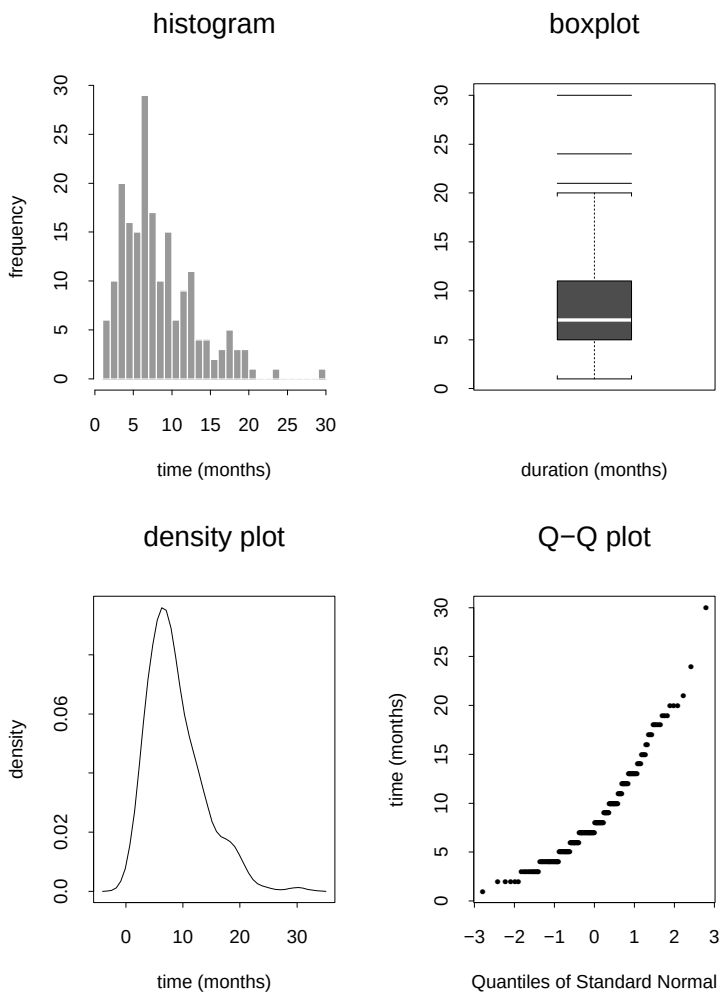

Figure 16: Visual insight into the empirical distribution of the reported durations of the IT-projects.

We have seen overviews of the function points (Figure 13), of the cost index (Figure 10), and to complete the picture, we also display such overviews for the duration and the cost index per function point in Figures 16 and 17 respectively. Figure 16 shows in plot 1 a histogram that is a bit more bell-shaped than the others, indeed in plot 2 we see outliers but not as many as for the other indicators, although the outliers are fairly large (cf. Table 6). Plot 3 shows the PDF which displays a heavy-right tail, and plot 4 is clearly not a straight line. We do see some clutter on several horizontal lines, which is due to many projects of the same length. So this plot gives also evidence that the durations are not normally distributed, and that a heavy tail to the right is present. Next, in Figure 17, we observe in plot 1 a histogram with a somewhat short, but heavy right tail. The many outliers, near the upper whisker of the boxplot plus a single large outlier also testify of a heavy tail. Then the empirical PDF shows a clean leptokurtic asym-

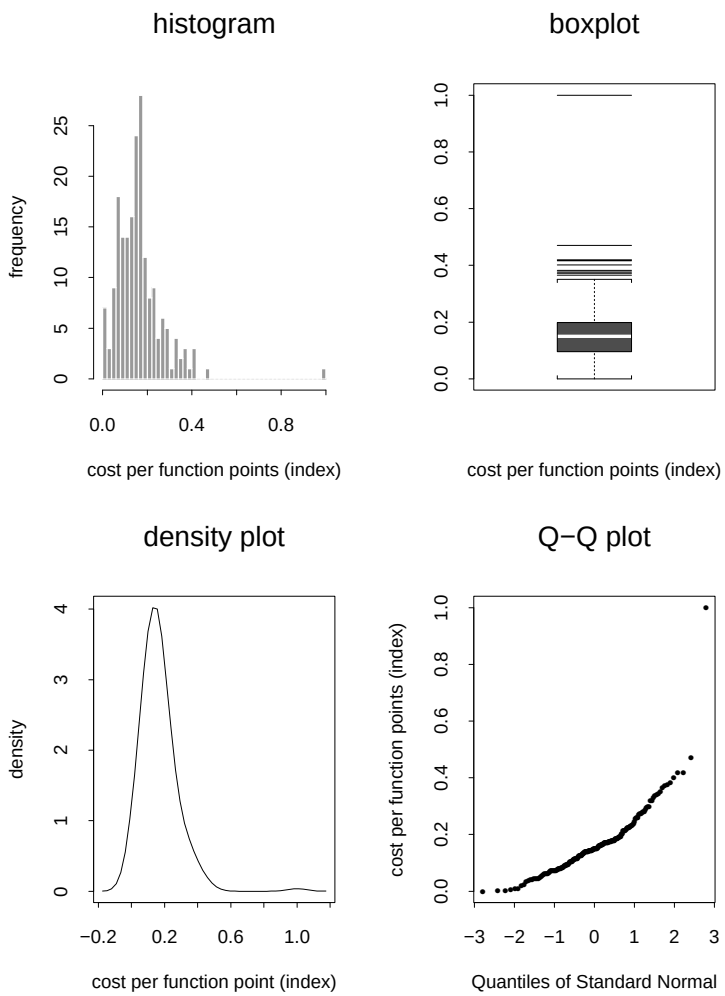

Figure 17: Visual insight into the empirical distribution of the reported cost index per function points of the IT-projects.

metric heavy right-tailed distribution, and the Q-Q plot shows no straight line. So we have now found visual, thus informal evidence that the four important indicators in the research set share important distributional characteristics. This is a sign of control as indicated before. But we want to know more about the heavy tails, we wish to model them so that we can also formally state that the distributional properties are shared.

A nice test for heavy tailed distributions is to provide a Q-Q plot of, say, the function point counts against a known simple heavy tail distribution: the exponential one. But before we do this, we first explain some elementary properties of this distribution. In the left-hand plot of Figure 18, we depict a density plot of a random sample taken from the exponential distribution. Note, that this is not the shape of the exponential distribution itself, which is of the form:

$$p(x) = \lambda e^{-\lambda x}, 0 < x < \infty$$

and thus is monotonically decreasing. Instead, we took a random sample using the exponential distribution, and plotted a density estimate based on this sample. The right-hand plot of Figure 18, is a density estimate of the function point data, which has a strong resemblance with the left-hand plot based on the random sample. Of course, the scales do not coincide, but the shapes do, which provides further evidence that the function point sizes are indeed heavy-tailed to the right. Recall that in Figure 4 we clearly saw that a number of the density plots are more or less similar to the generated plot. Of course, some of the data looked heterogeneous, but in the aggregate this turned out to be embryonic heavy tails, hence the trembles in the heavy-right tail.

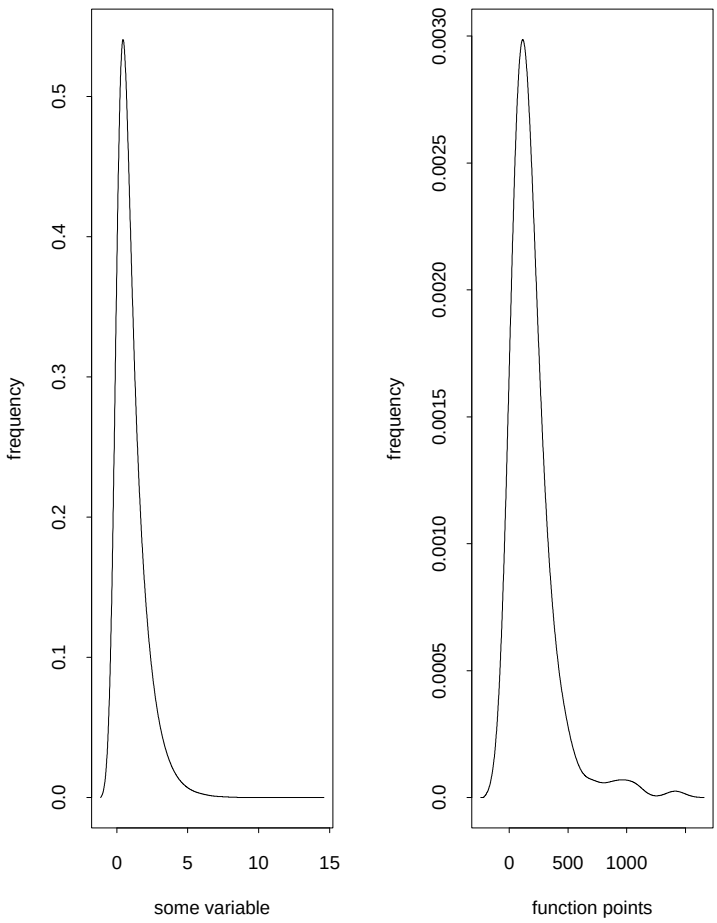

Figure 18: Left plot: density of a random sample taken from an exponential distribution with unit rate. Right plot: density of the function point data.

In Figure 19 we depict the Q-Q plot. As can be seen, the empirical quantiles of the

function point totals have a strong linear relation with the quantiles of an exponential distribution. This adds to our confidence that the function point totals are plausible values for an organization in control of software sizing.

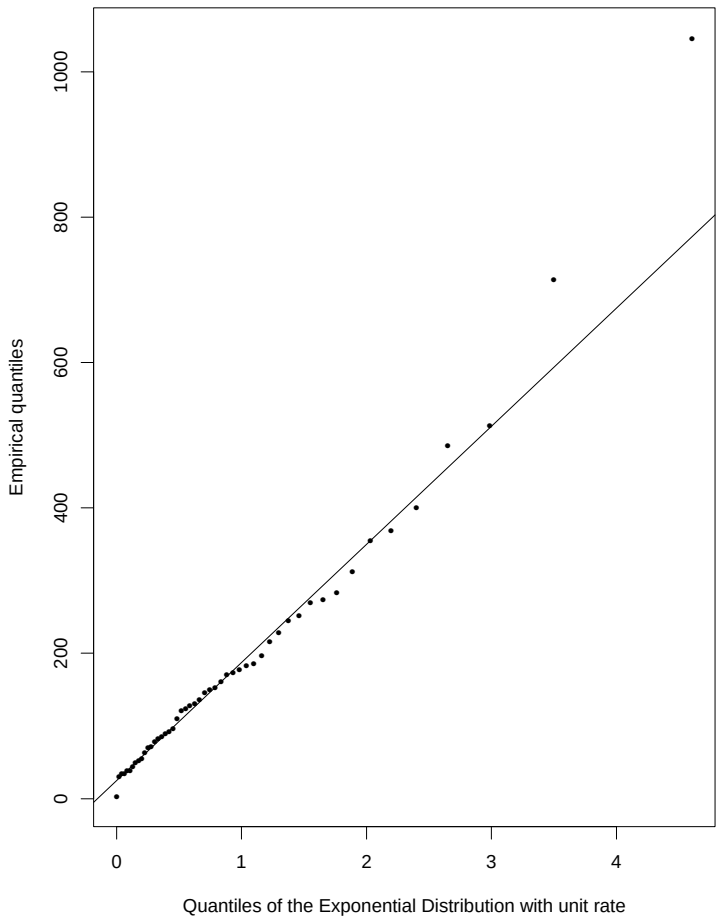

Figure 19: Q-Q plot of the function point totals against the exponential distribution.

We recall that the heavy tail is an indication that the function point data is not only accurate but also plausible. Namely, it is our experience that managed IT-portfolios are following the famous 20/80 patterns, roughly stating that 20% of the IT-projects take 80% of the effort. More precisely, often 10–25% of the largest projects consume the majority of the IT-budget (60–80%), while the rest of the projects is relatively small, and costing often not too much (with a few exceptions most of the times). This should also be the case for the related measurements. An outlier in size, should lead to an outlier in duration, an outlier in actual cost, and maybe an outlier in cost per function point. Of course, since there are many unknowns influencing these variables, there will be no one-to-one correspondence on a per-project basis, but the heavy-tail properties

should be reflected. In Figure 20 we depicted all Q-Q plots of the four important interrelated variables function points, duration, actual cost index, and cost index per function point.

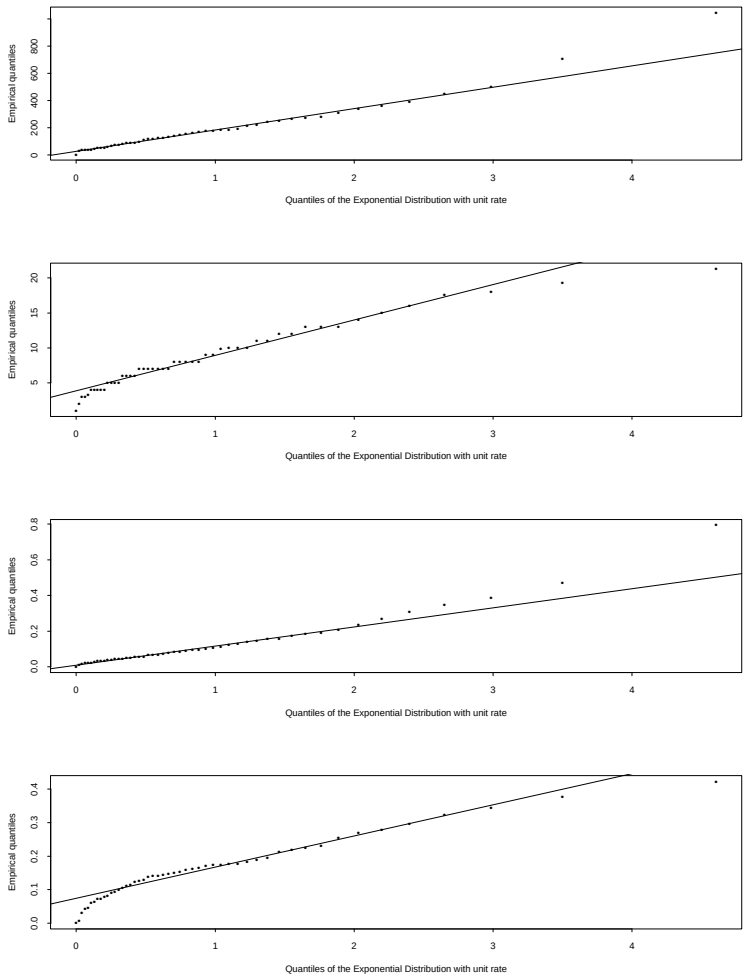

Figure 20: Q-Q plots of the function points, durations, cost index and cost index per function point against the exponential distribution.

As can be clearly seen, all variables seem to be heavy-tailed, since the empirical quantiles mainly follow a linear relation with the quantiles of the exponential distribution. Of course, there are deviations, and at the end-points sometimes even severe ones. This is due to the fact that the heavy tails are not exact exponential distributions, but the fit is good enough to conclude that heavy-tails are present. To get a better idea of the tail behaviour, we commence with a heavy-tail analysis, common in extreme value analysis. Such analyses are common for risk measures in financial markets. For instance, the daily returns on a stock exchange, insurance claim data, etc, are often heavy-tailed as well. Extreme value analysis is then used to calculate risks known as

VaR (Value-at-Risk), and ES (expected shortfall) [2, 10, 94]. We will use such analyses to fit heavy-tail distributions so that we can conclude that the data reported is plausible. This enables us to make simulations with the found models, which is useful for predictive modelling of future IT-investments. For instance if an entire business unit is in- or outsourced, you can simulate what-if scenarios given some IT-investment strategy. We will in this paper only touch upon that subject but then to assess the adequacy of the found models.

5.1 Pareto distributions

A well-known family of heavy-tail distributions is known as the *Generalized Pareto Distributions*, GPDs for short. This Pareto is the same one from the Pareto Principle or the 20/80 rule. A GPD is of the following form:

$$H(y) = \begin{cases} 1 - (1 + \xi y/\beta(u)), & \text{if } \xi \neq 0 \\ 1 - exp(-y/\beta(u)), & \text{if } \xi = 0 \end{cases}, \qquad \beta(u) > 0.$$

where ξ is a parameter, u is a (often) high threshold to indicate from whereon we wish to model the tail, and $\beta(u)$ is another parameter of the GPD that depends on the chosen threshold. The GPD H is defined for $y \geq 0$, if $\xi \geq 0$, and when $\xi < 0$, the range of y is $[0, -\beta(u)/\xi]$.

The problem of fitting GPDs starts out with the estimation of this threshold u. This can be quite tricky, since we now have to give an answer to the question when the tail of a distribution starts. The wrong choice can have large effects on the quality of the fit that models the tail behaviour. There are a few methods to get an impression of the proper choice, but there is no single best choice. One problem is that the more you are in the tail, the less data is left for a proper estimate. But if you have enough data for a good estimate of the parameters of the GPD, you may no longer be in the tail leading to erroneous results. We will discuss some methods that give an idea of the choice of a proper threshold, and we discuss one method of checking whether the found model is appropriate. They are:

- A mean excess plot. This is a plot that might reveal a linear relation between various threshold sizes and their mean excess.
- A shape plot showing how estimates of ξ vary with a range of different thresholds. If such a plot is reasonably stable for a certain range this gives an indication of an appropriate threshold.
- Hill's estimate of ξ: this is a method due to Hill that gives an estimate of ξ, given a threshold u.
- Random sampling of the model: if we found a tentative model, we take a random sample of it, and compare that sample to the original data set. Using the nonparametric Kolmogorov-Smirnov goodness of fit test [58, 76, 20], we can then see whether the random samples of the model reflect the actual empirical distribution of the data from which we inferred the model.

We start out with the mean excess plots in Figure 21. This calculates the sample mean excesses over increasing thresholds. The empirical mean excess function is defined as follows:

$$e_{n_u}(u) = \frac{1}{n_u} \sum_{i=1}^{n_u} (x_{(i)} - u)$$

where the $x_{(i)}$ is an ordered permutation of those values x_i, for which $x_i > u, i = 1 \ldots, n_u$. See [26] for elaborations on mean excess plots, and their theoretical justification. Based on these plots we can see that in plot 1 (Figure 21), somewhere after 300 the mean excess becomes somewhat linear. In plot 2, we see linearity after threshold 5, but not that clearly. In plot 3 we see after a dip at 0.3 linearity, but not so obvious as you would want to see it. Finally, plot 4 displays linearity, but only for smaller thresholds, except very large thresholds, but then the data set is too sparse. At least we have an idea of the order of magnitude for the various thresholds, so that we have a first impression of where the actual tail of the distributions is starting.

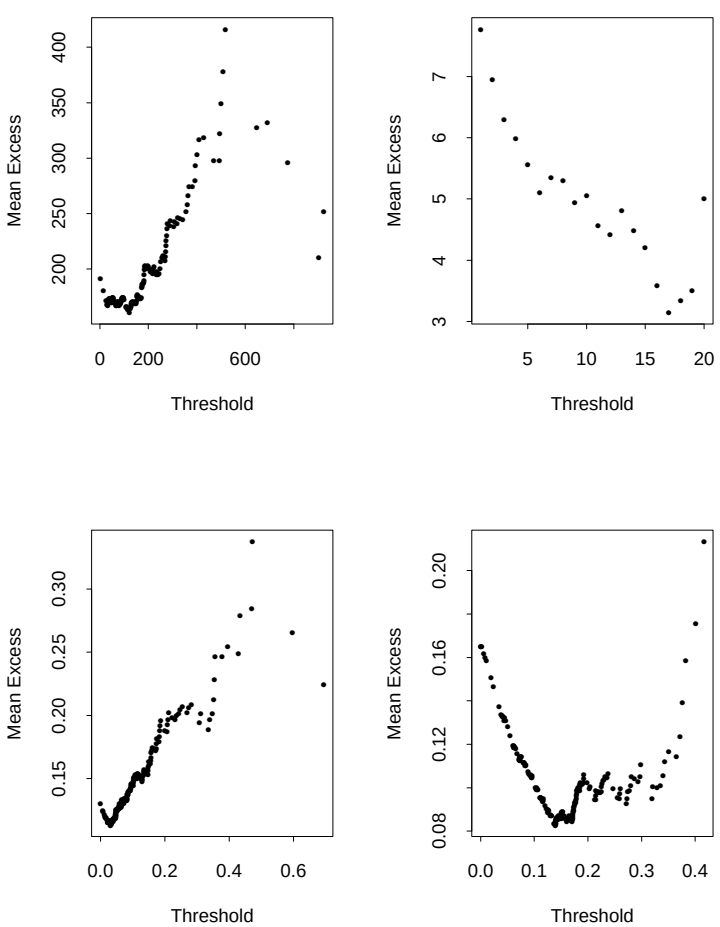

Figure 21: Sample mean excess plots for the function point data, durations, actual cost index, and cost index per function point.

Next we turn our attention to shape plots where we can inspect how various estimates for the shape parameter ξ vary with a number of different thresholds, or equivalently, the number of extremes. Plot 1 of Figure 22 shows fairly large stability for ξ between 30 and 88, and for larger values we see some instability, larger confidence intervals, and less data to base ourselves on. Since the function points seem to follow a GPD quite accurately given the resemblance shown in Figure 18, we are confident in taking a somewhat lower threshold of 88. Plot 2 is somewhat unstable, except for threshold 6. So this number could serve as a threshold, since lower numbers would really not be in the tail, and larger ones already lead to somewhat instable values for ξ. In plot 3 we can see a fairly stable range for ξ's over the entire range, so one would expect a threshold of 0.1 to be plausible. Plot 4 gives us an indication of thresholds around the 0.15 for a fairly stable ξ.

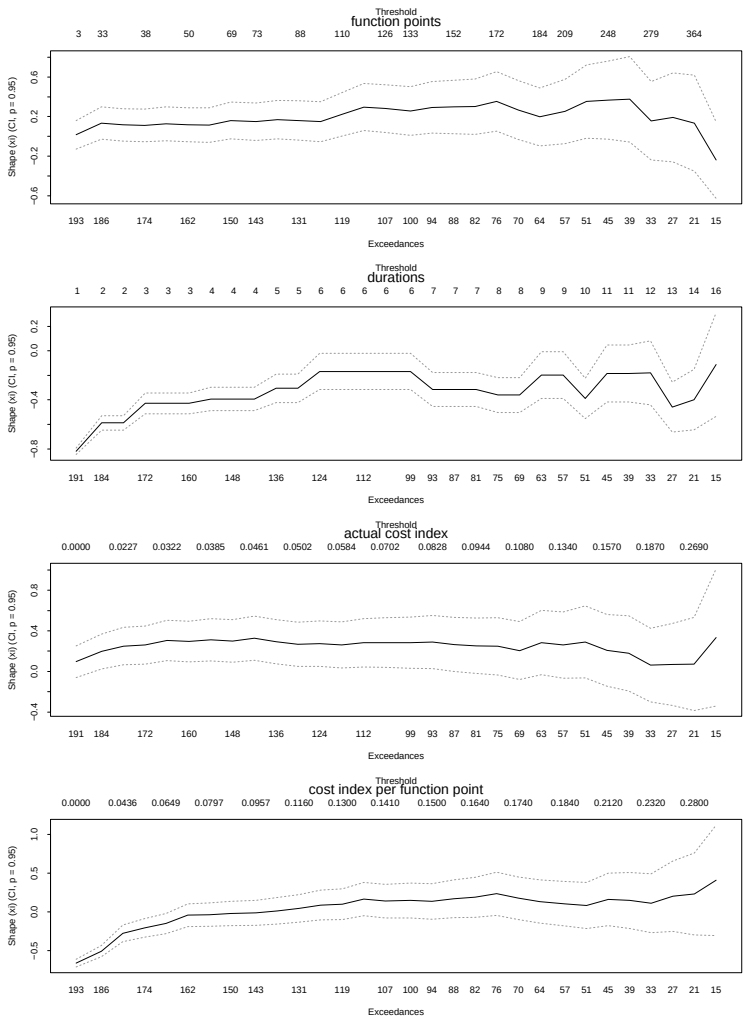

Figure 22: Shape plots plus confidence intervals for the function point data, durations, actual cost index, and cost index per function point.

The Hill estimator is defined as follows:

$$Hill(k) = \frac{1}{k}\sum_{j=1}^{k}(\log(y_{(j)} - \log(y_{(k)})$$

for an ordered sample $y_{(1)} \geq \cdots \geq y_{(n)}$. For details on Hill estimations we refer to [47]. In Figure 23 we plot the Hill estimates. In plot 1 we can clearly see that the value of ξ is stable starting very early already, so our threshold of 88 is not a bad idea also in the case of Hill estimation. For other thresholds we see a ξ of about 0.5, but these values do not need to be accurate. In [10, p. 37], we can read:

> This index ξ can be estimated parametrically by approximate maximum likelihood methods, such as Hill's estimator. But the resulting estimates are very unstable, and horror stories documenting erroneous results have been reported in the literature, and have attracted the attention of many practitioners.

We use the Hill's estimator here to obtain additional information on thresholds, not to estimate the parameters of the underlying GPDs. In plot 2 we see a similar choppy line, the longest stable piece is again around threshold 6 (also a 0.5 estimate for ξ). Plot 3 shows a slight increase for ξ for higher thresholds. It is not easy to give a first indication of a ξ, based on this plot, but thresholds up to 0.07 are fairly stable. Then in plot 4, we see again a stable line for many thresholds, somewhat below 0.5, and a threshold of about 0.15 is showing stable values for ξ. Sumarizing we opt for the following thresholds for the samples: 88 for the function points, 6 for the durations, 0.07 for the actual cost index, and 0.13 for the cost index per function point.

Now that we have an indication of the various thresholds, we fit the parameters of the GPD distributions on the sample data. The numerical results of those fits are summarized in Table 7. Before we go to the numbers, we discuss the diagnostics on tail behaviour that we depicted in Figures 24–27. For, those numbers only become important if the models are adequate.

In Figure 24, we see four plots. We discuss the ideas behind these plots. Plot 1 contains a solid-line which is the so-called excess distribution based on a GPD instantiated with the fitted parameters. The dots are the empirical values. What is important as a diagnostic for this plot is that the solid line should coincide as much as possible with the empirical data, which is the case. Plot 2 shows the tail estimate via the survival function. This is just the cumulative distribution function of the excess distribution only then inversed, hence the $1 - F(x)$ on the vertical axis. Again, what matters is that the solid line coincides as much as possible with the empirical survival rate, which is the case. So, the GPD estimate seems to fit the distribution of excesses pretty good. Not perfect, as can be seen in plot 3: for that the data are converted to residuals standardized by an exponential unit rate distribution. The solid line indicates the standard error. There are a few more outliers than you would expect from such a good fit, but this is due to the heavy tail of the data. If we order the data, and plot the residuals against the quantiles of a unit rate exponential distribution, we see a fairly straight line. These four plots suggest therefore that the fitted model for the function point data is adequate.

Figure 25 assesses the fit for the durations. Note that we did not use the log scale, as we did for Figure 24. The reason is that this data is a bit different than the other data. It has discrete properties rather than continuous values. Namely, the variable represents

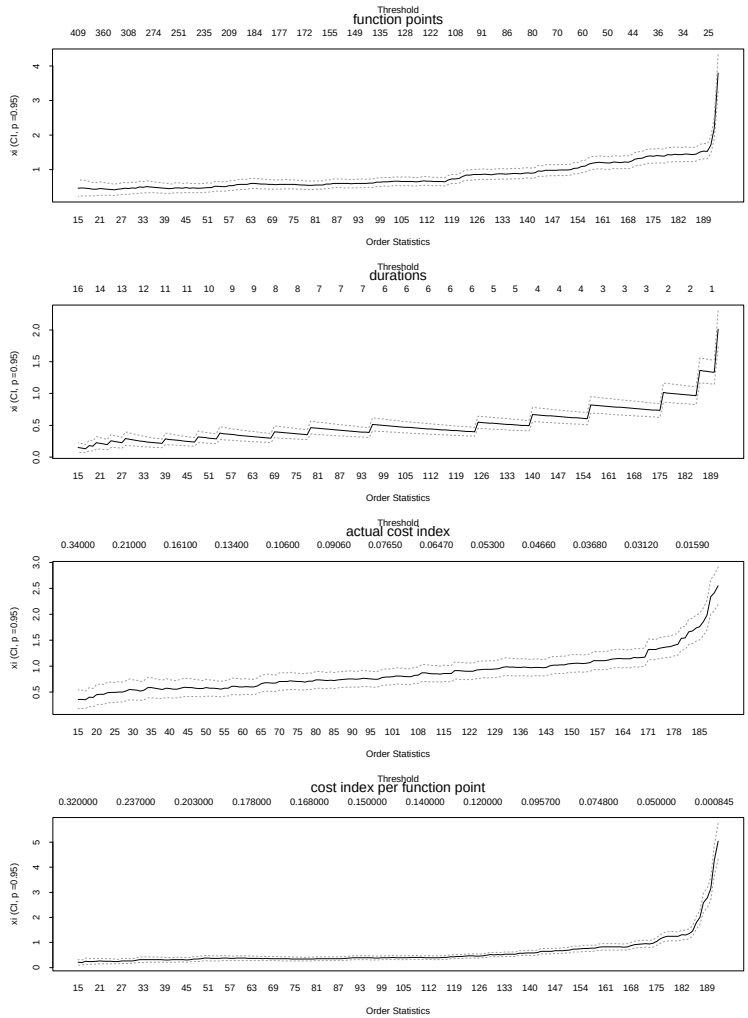

Figure 23: Hill's estimate for the function point data, durations, actual cost index, and cost index per function point.

calendar months, in particular durations of IT-projects. We can see in plots 1 and 2 that the data is stacked, and that for the duration of 6 months we have a high stack. This is maybe an effect that we also see in price policies: there are more items of $9.95 than just above ten dollar, since the former is a "better" price. Similarly, half a year for a project seems to be a better approval policy, than other time frames. We note that this can have potentially dangerous side-effects that could easily annihilate an entire investment in a software process improvement program. We refer the reader to [86] for elaborations on time compression and decompression risks in IT-portfolios. In any case, all four diagnostic plots seem to be in agreement that the GPD-fit for the tail of the durations is reasonable.

In Figure 26, we plot the diagnostics for the actual cost index. As we can see there are less data points than in the previous two figures. This has to do with the choice of

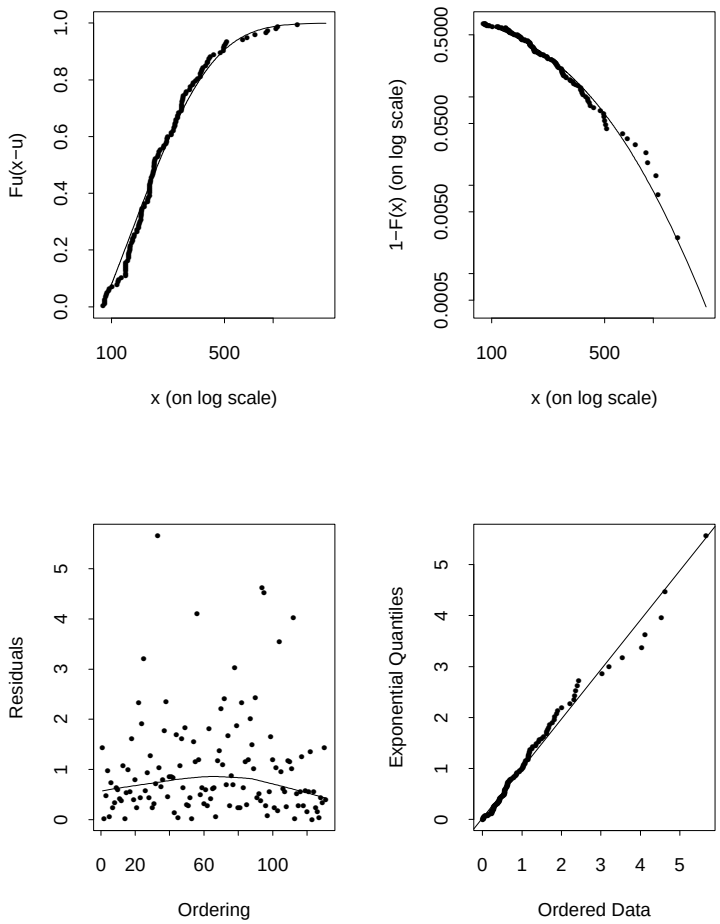

Figure 24: Four different plots for assessing the fitted GPD model on the function point data.

the threshold. Maybe we need another threshold in the end, but for now all four plots seem to be in accordance with the fitted GPD model for the cost index.

For the most important variable, namely the cost index per function point, we have more data points. In fact we tried to find the optimal amount of data points, such that we are in the tail, and simultaneously we have enough data for an accurate fit. Recall that the entire exercise that we are carrying out here is to gain confidence in the plausibility of the data, despite the many outliers. The cost index per function point is the key performance indicator to measure changes in IT-productivity, so departing from the reliability of the function point data, we want to establish a conclusion on plausibility of the cost per function point data. If this fits a GPD just as the function points are doing, this forms (indirect) evidence that the cost information is also reliable. As we can see in Figure 27 the excess distribution fits nicely, and so does the survival function.

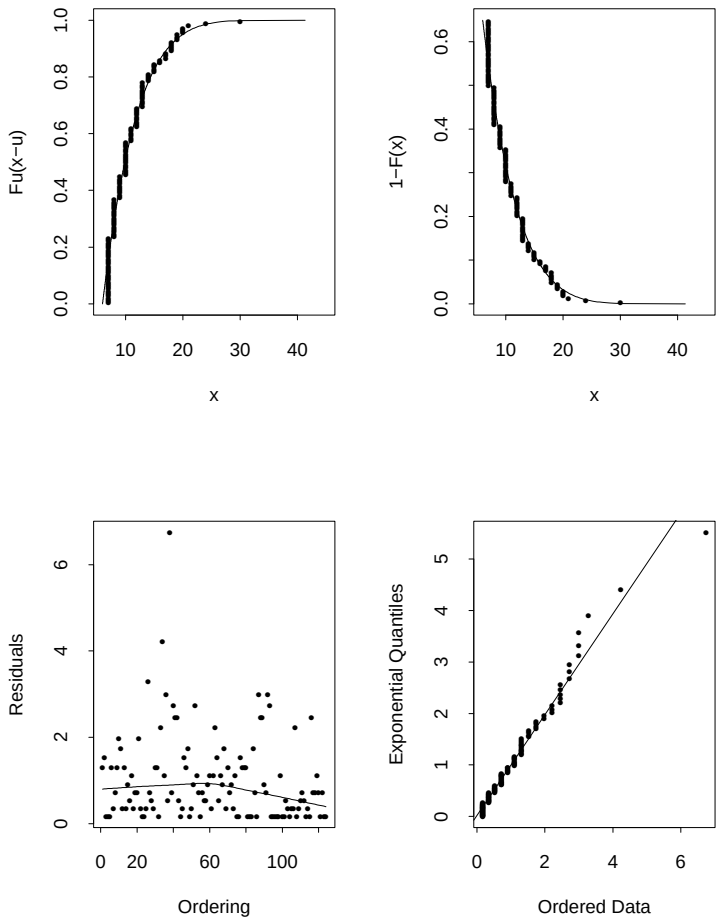

Figure 25: Four different plots for assessing the fitted GPD model on the duration data.

The residuals are not too many, and the Q-Q plot against the unit rate exponentials of the residuals is also very nice.

In conclusion we can state that all four indicators in the research set possess heavy tails to the right, and that the GPDs seem to fit reasonably to very well.

5.2 Model diagnostics

To establish a more formal idea of the adequacy of the found GPD models, we will use sampling techniques. The idea behind sampling is fairly straightforward. Suppose that the model is adequate, then this implies that if we would draw a random sample from the model, this sample should exhibit similar properties as the data itself. In statistical terminology, this means that the distribution of the random sample should resemble the distribution of the data. In Figure 18, we already alluded to this technique, but then

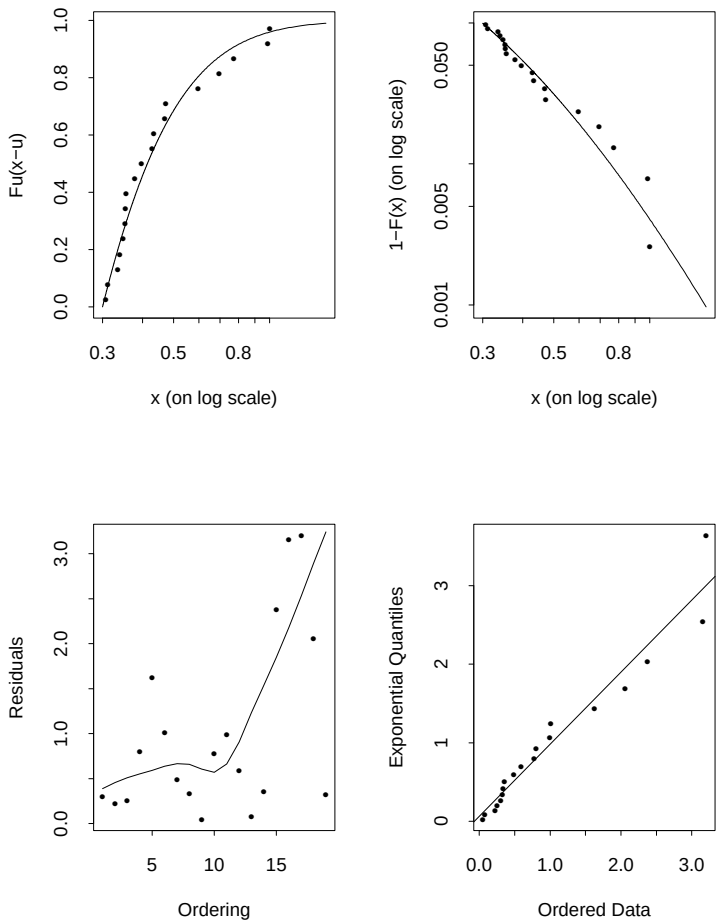

Figure 26: Four different plots for assessing the fitted GPD model on the actual cost index.

to close in on a family of distributions that resembled the shape of the distribution of the data. In that figure we took a random sample of a unit rate exponential distribution and estimated the density from that. We compared this visually to the density estimate based on the data. Now we take this idea a step further: we generate a random sample from the fitted models, and we test whether the original distribution and the random sample distribution coincide or not. If they coincide, the model is adequate.

In Figure 28, we depicted four plots. Plot 1 is a random sample of the same size as the actual data set, and the values are generated by the model that we fit for the function points. In plot 2 we depicted the actual data, in this case the function points. Plot 3 is a comparison of the CDFs of both the simulated and the actual data. We can see that these distributions seem to coincide. Plot 4 is a Q-Q plot of quantiles of the random sample against the actual data. The straight line indicates that the simulations

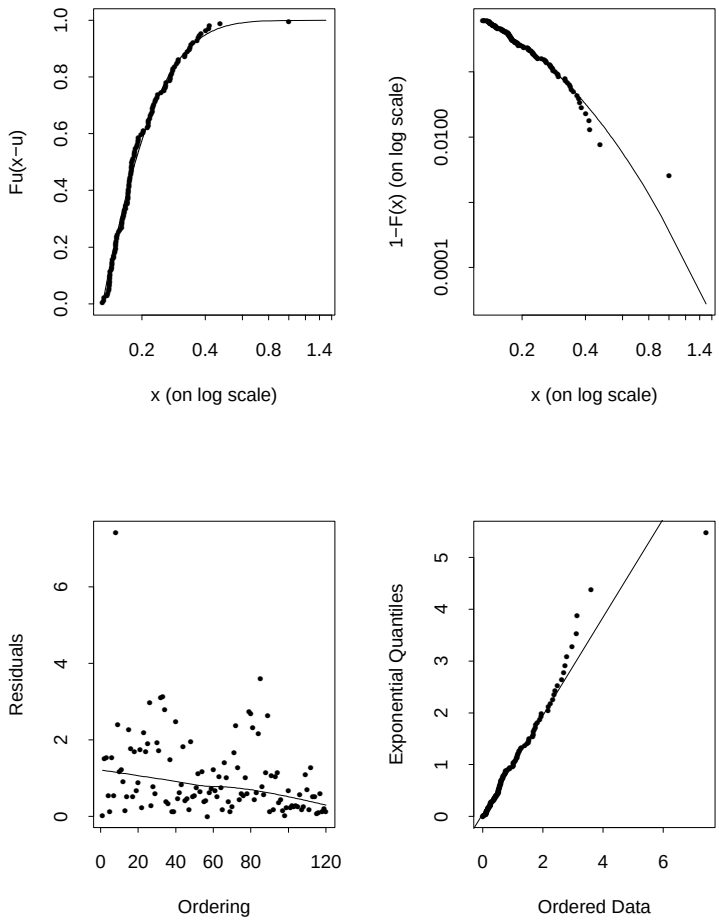

Figure 27: Four different plots for assessing the fitted GPD model on the cost index per function point.

and the actual data have the same distribution. Finally, a formal Kolmogorov-Smirnov goodness of fit test gives a KS-statistic of: 0.0466, with a p-value of 0.9676, indicating that there is no evidence whatsoever that both distributions differ. Hence the found GPD-model for the function point is adequate.

Figure 29 contains the four plots for the durations. Since some of the data is clogging together, this gives more choppy CDF comparisons and a more roughly looking Q-Q plot. The simulations in plot 1 are resembling the actual durations in pattern. Although a Kolmogorov-Smirnov goodness of fit test gives a favorable conclusion: the KS-statistic is: 0.089, the p-value is not as high as in the previous case: 0.3845, but still very high. So, also the found GPD-model for the durations is adequate.

Figure 30 looks just like Figure 28 very good, and the Kolmogorov-Smirnov goodness of fit test supports this with a KS-statistic of 0.0576, and a very high p-value of

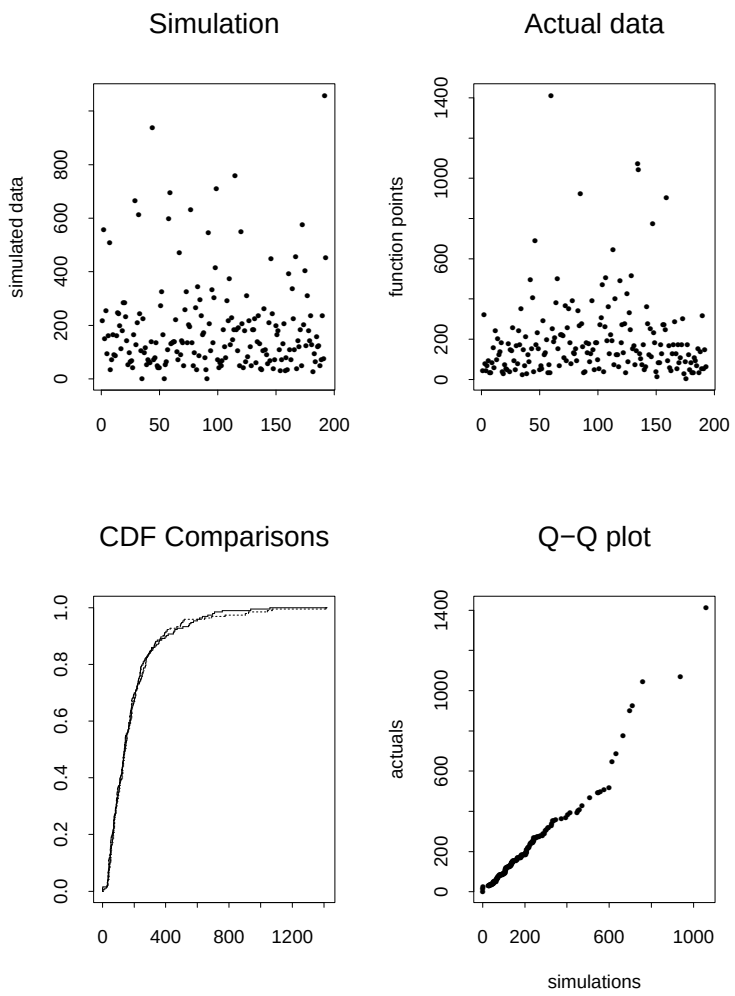

Figure 28: Four plots indicating the simulated data from the fitted GPD model, the function point data, a comparison of both cumulative distribution functions and a Q-Q plot.

0.8656. So also the GPD-model for the cost index is adequate. For Figure 31 the same holds; the plots look ideal. Moreover the KS-statistic is: 0.0466, and the p-value is 0.9676. Note that these values are the same as for the function points. We used the same random seed to initiate the pseudo-random generator, and apparently, this leads to identical numbers in this test. Of course, if we change the random seed for this one, these numbers will change a little. So also the cost index per function point is adequately modelled.

In Table 7, we summarized all the numerical results. The abbreviations in the first column stand for our four variables: function points, durations, actual cost index, and cost index per function point. In the second column we find the fitted ξs. As can be seen they differ from the values that the Hill estimator predicts. We note that if the shape

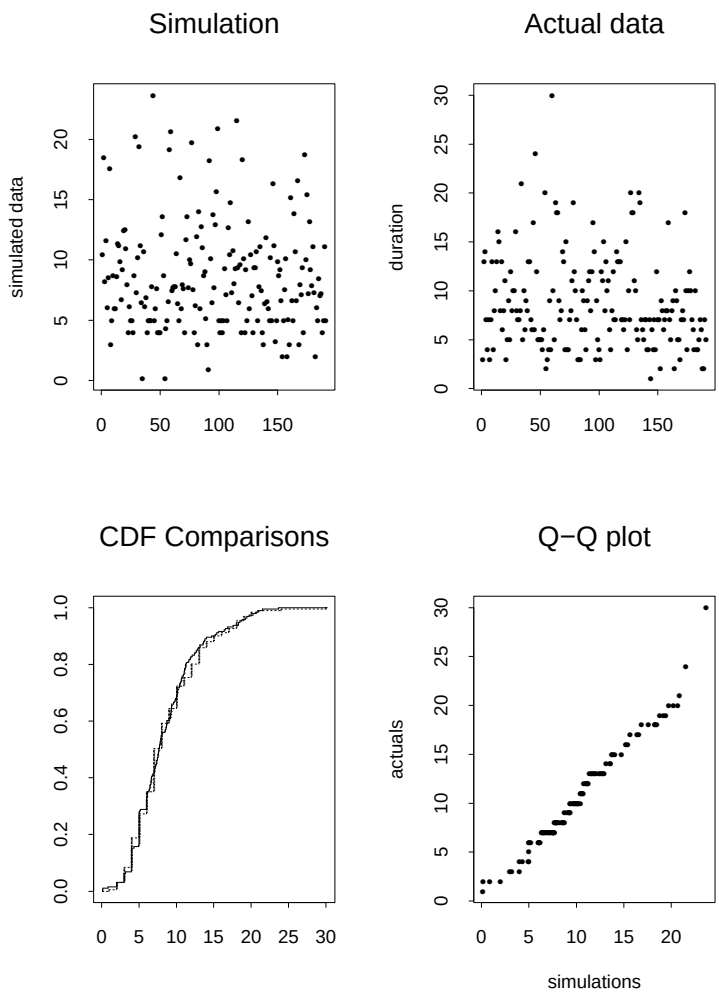

Figure 29: Four plots indicating the simulated data from the fitted GPD model, the durations, a comparison of both cumulative distribution functions and a Q-Q plot.

variable	ξ	u	$\beta(u)$	s.e. ξ	s.e. $\beta(u)$	KS	p-value
FPs	0.161	88	143.49	0.094	18.3	0.0466	0.9676
dur.	-0.168	6	5.95	0.068	0.665	0.089	0.3845
act.	0.295	0.07	0.0930	0.13	0.0150	0.0576	0.8656
cpf	0.103	0.13	0.0781	0.087	0.0098	0.0466	0.9676

Table 7: Numerical summary of heavy tails analysis, the fitted parameters, and the goodness of fit tests for the found models.

parameter is almost zero, that this implies that the tails are not too heavy. But since the ξ-values are all differing from zero, this is not the case. So all variables have a fat

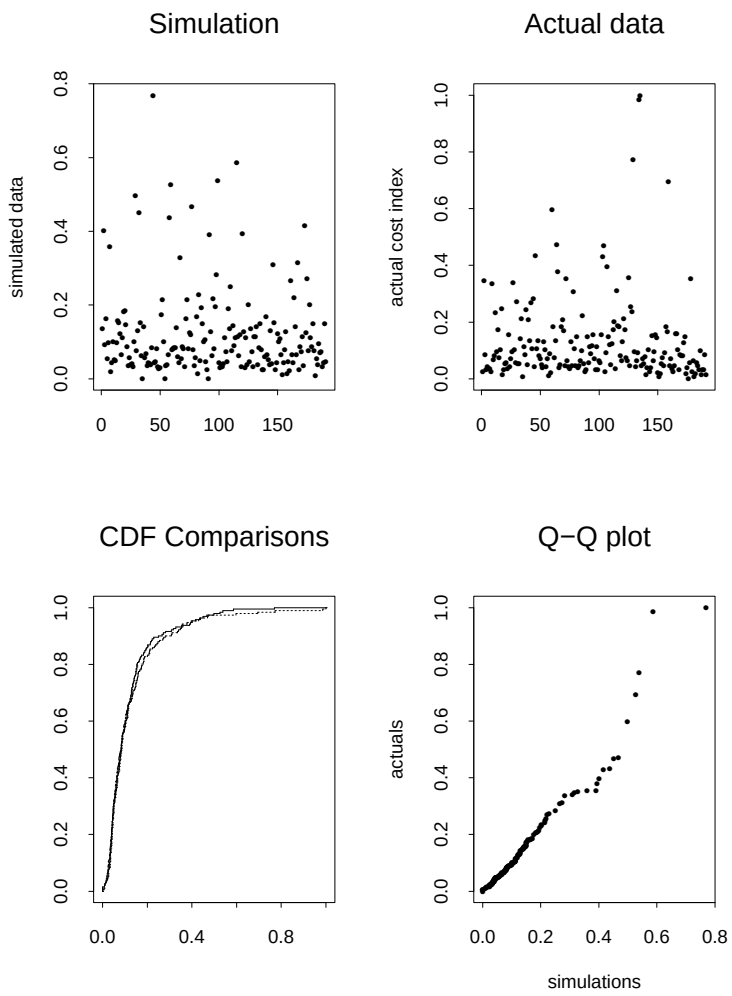

Figure 30: Four plots indicating the simulated data from the fitted GPD model, the actual cost index, a comparison of both cumulative distribution functions and a Q-Q plot.

tail as we hoped to find from the accurate function point data. The next column gives the used thresholds, and the subsequent column gives the fitted parameter that depends on the fitted threshold. It is important to know whether these values are significant. Therefore, we also provided the standard errors (s.e. for short) of both ξ, and $\beta(u)$. We can see right away that all fitted values are significant, but that some values do have a fairly large standard error, say almost half the fitted value. So it was wise to use a sampling test to get more confidence in the models, and we summarized the KS-statistic and its corresponding p-value in the last two columns.

At this point we have established the following: the function point data seems accurate and plausible, and based on the distribution type of the function point data we conclude that there is no evidence that the cost index per function point is not reliable

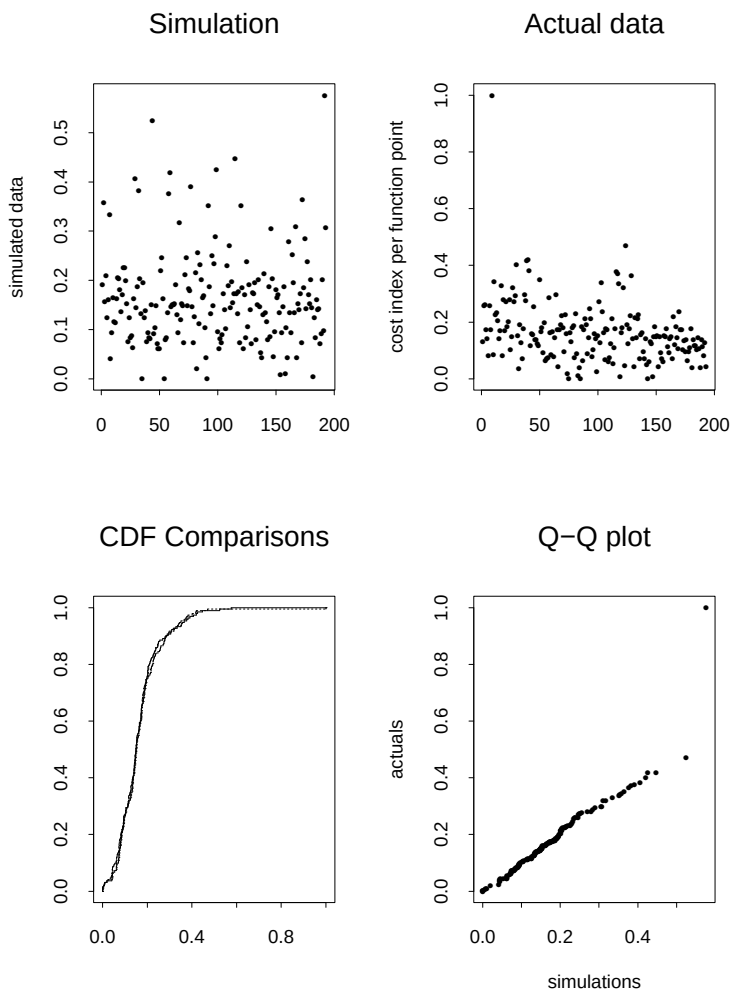

Figure 31: Four plots indicating the simulated data from the fitted GPD model, the cost index per function point, a comparison of both cumulative distribution functions and a Q-Q plot.

and not plausible, despite its large variation.

Next we should investigate the correlations between function point size, and their corresponding cost and duration over time in order to ascertain that the data is not only accurate and plausible distribution-wise but also satisfactorily correlated over time. This necessitates a multivariate time series analysis. We will carry out this somewhat complex analysis in Section 10, but for explanatory purposes, we start out with a univariate time series analysis, which is the simpler variant of a vector time series analysis. After we have done this, we will carry out the multivariate case to confirm that the function points, cost and durations are indeed satisfactorily correlated over time. For now we just assume (rightly as we will see later on) that the data is also reliable if time is taken into account, so that we can now analyse the cost per function point data.

6 TIME DEPENDENCIES

Until now we have not taken the time into account. But if we wish to know anything at all about how a software process improvement effort works out, we need to analyse the data through time. We will focus on the cost index per function point data as a function of time in this section. By the previous sections, we do not have doubts about the validity of the data, despite their large variation. We are not going to dive deeper into the activities on a per-project basis to eradicate this variation (as some researchers think should happen [53, 80]), but we accept this as a fundamental assumption. There are stochastic effects present for many IT-related metrics, in particular our cost index per function point, and we continue from this assumption. We want to understand whether a software process improvement project within a large organization indeed leads to tangible improvements. One of them being that the software production productivity increased when time passes, or equivalently that the cost index per function point decreased over time.

But if these costs vary so widely, how can we spot such improvements? Let us first depict in Figure 32 the cost index per function point as a function of time. We processed this data in such a manner that from a so-called irregular time series we extrapolated the data into a regular time series, of which Figure 32 is the result. The regular time series gives a measure every few days, whereas in reality in weekends this has not been done. This has no effect on the calculations, other than to ease them a little. Let's make the problems a bit more apparent. Although we indexed the cost we can state that the difference between the highest and lowest cost per function point is a factor of about 40. So, these costs vary enormously (4000%). In Figure 32 we put two horizontal lines, the solid one being the target line: this indexed IT-productivity should eventually result from the software process improvement program. The dotted line is the long-term average of the time series. As we can see, the so-called mean-reverting behaviour of the time series plot is fairly high, and a linear fit to come to grips with this wide variation would be immoral in order to gain insight. Moreover, the mean is above the target line, and it is not immediately obvious whether the time series data is on average more near to the target line than to the mean line. So we have a problem.

6.1 Naive Approach

The organization approached this as follows: they weed out the wide variation by taking an annual moving average. This works as follows: at a certain point in time, take all historical data of the last 12 months, average that and this is one new point of a graph. Do this for all points for which you can calculate the moving average. Now compare the moving average graph with the target line, and find out whether the new line is eventually diving under the target line. In fact, they found a way to get rid of the wide variation by a smoothing operation, in this case by taking a simple linear historical moving average. We will depict the results of such an operation for various time frames on the time series in Figure 33.

Indeed, we can see in Figure 33 that the wide variation decreases and a more stable pattern emerges. First we depicted the raw data, then a one-sided monthly moving average—one-sided means that only historical data is taken into account. The dashed line shows the long-term mean of the data, and the solid line the target line. We observe that the target line is sometimes crossed by the monthly moving average but not as often as the raw data. Also the difference in vertical range of the moving average is much smaller than for the raw data. This effect increases for longer time frames: less

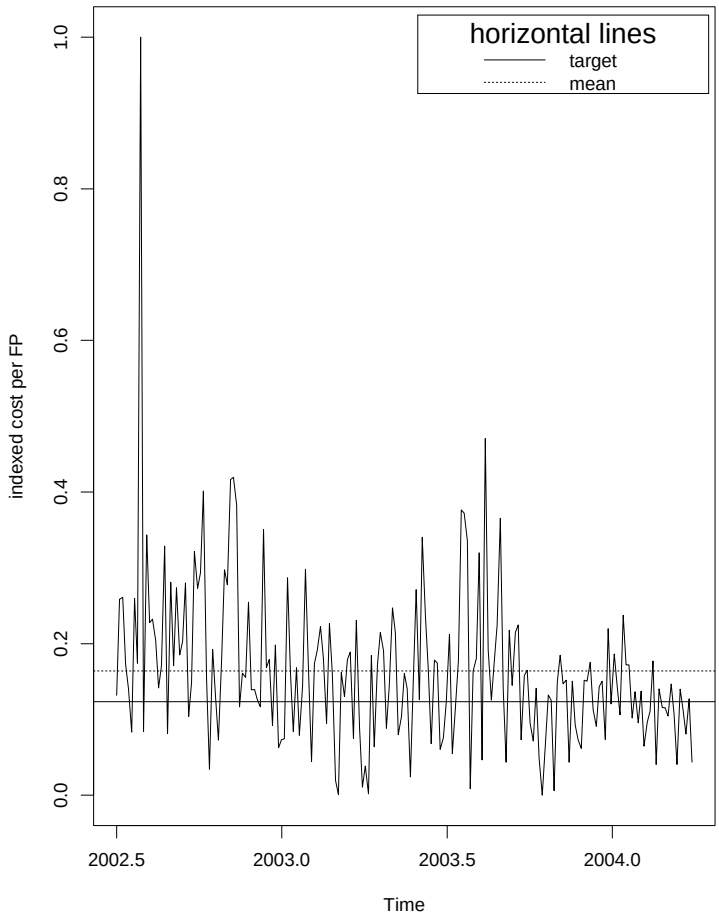

Figure 32: Indexed IT-productivity as function of elapsed time.

variation, less crossing the target line, and less difference in range. Finally, the annual moving average is almost a straight line, and not crossing the target line at all. The stability of the line is best viewed in the last plot of Figure 33. In fact, this long-term moving average is approaching the long-term mean of the time series, and trend effects are dampened to an unacceptably low degree.

To illustrate that taking a moving average is not a sound idea, we conducted an experiment. We took a random sample (without any trend) and we took longer and longer moving averages of that sample to see whether we could identify any trend. Although a trend is not present, we spot in Figure 34, trends up and down depending on the time-frame of the moving average. The longer the time-frame for the moving average, the more the line will approach the long-term average of the time series. So in effect, this smoothing operation cancels the factor time out. And this is not a good idea if we want to gain insight in a variable over time. Vice versa, if there is a trend in the

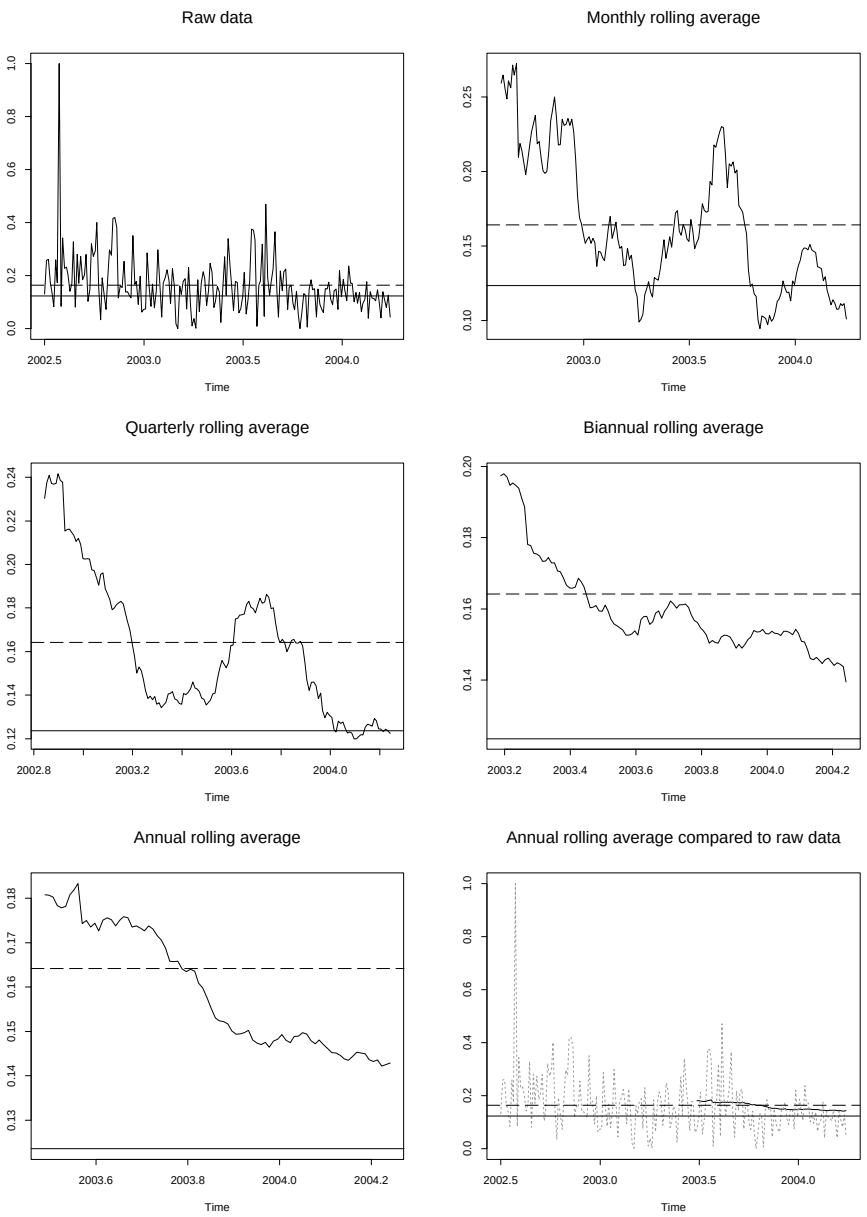

Figure 33: Different moving averages, and a comparison of the actual data with such an average.

data, taking a moving average is a good way of *hiding* it. We quote from [17, p. 17]:

> The simple moving average is not generally recommended by itself for measuring trend, although it can be useful for removing seasonal variation.

The moving average idea is a form of filtering but this works only if we know anything about the underlying data. For instance, if the data is cubic, we can use a Spencer

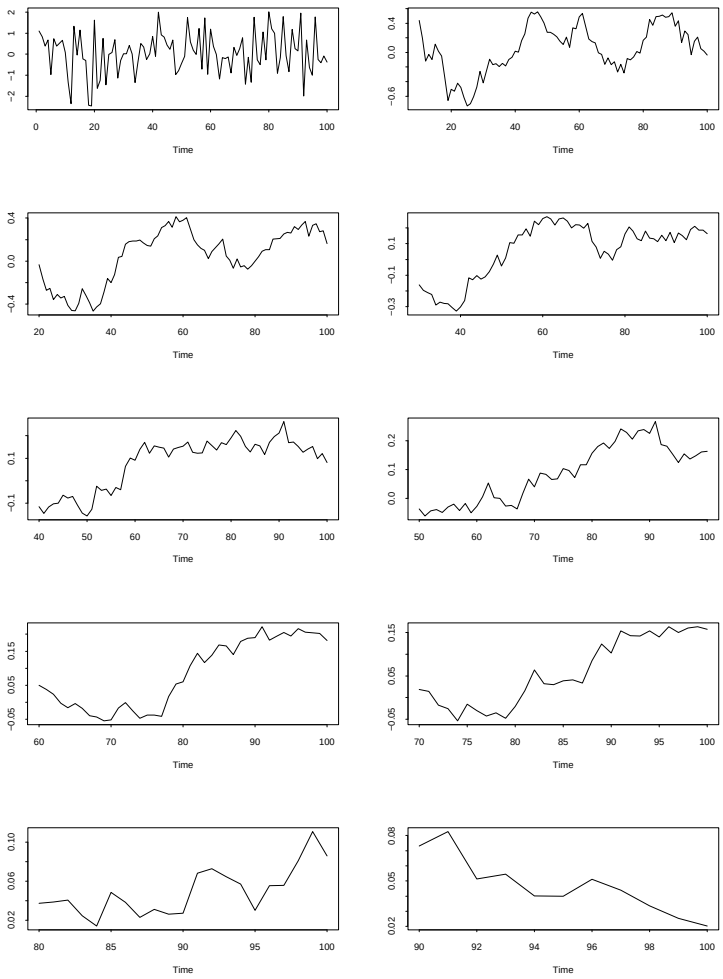

Figure 34: Different moving averages for a random sample.

filter, this is a 15-point filter with specific weights that will pass a cubic function without distortion [77]. In general for each polynomial trend we can define specific weights such that a filter using those weights will pass the polynomial without distortion [14, Prop. 1.1, p. 6]. In such a manner the high-frequency data is filtered out in favor of low-frequency data, therefore these filters are also called low-pass filters. Another important issue are outliers. They affect the quality of the model and introduce a bias in aggregates, like a moving average smooth, which can lead to misinterpretations of the actual situation. So we have to do something different. And this boils down to embracing the wide variation, outliers, warts and all, instead of trying to stamp them out all too quickly. And before we start thinking of smoothing the data, we first need insight in the data itself, which is the field of time series analysis. While in classical statistics we strive for independent observations, in a time series context there is often serial correlation, and in order to gain insight in such noisy data, we try to exploit this

correlation to construct appropriate models.

6.2 Scrutinizing noise

There is noise and noise, and one of the first things we need to do, is to investigate whether the noise in our time series is really random, or that there is some systematic effect in place. Namely, if you embark on a software process improvement program, there is no immediate effect. If there is an effect it is lagged and such effects can be measured by investigating the so-called autocorrelation of the data. Two plots are particularly important to that end: the *autocorrelation function*, ACF, for short, and the *partial* ACF, also PACF. For the time series depicted in Figure 32, we calculated both functions, and we exhibit them in Figure 35.

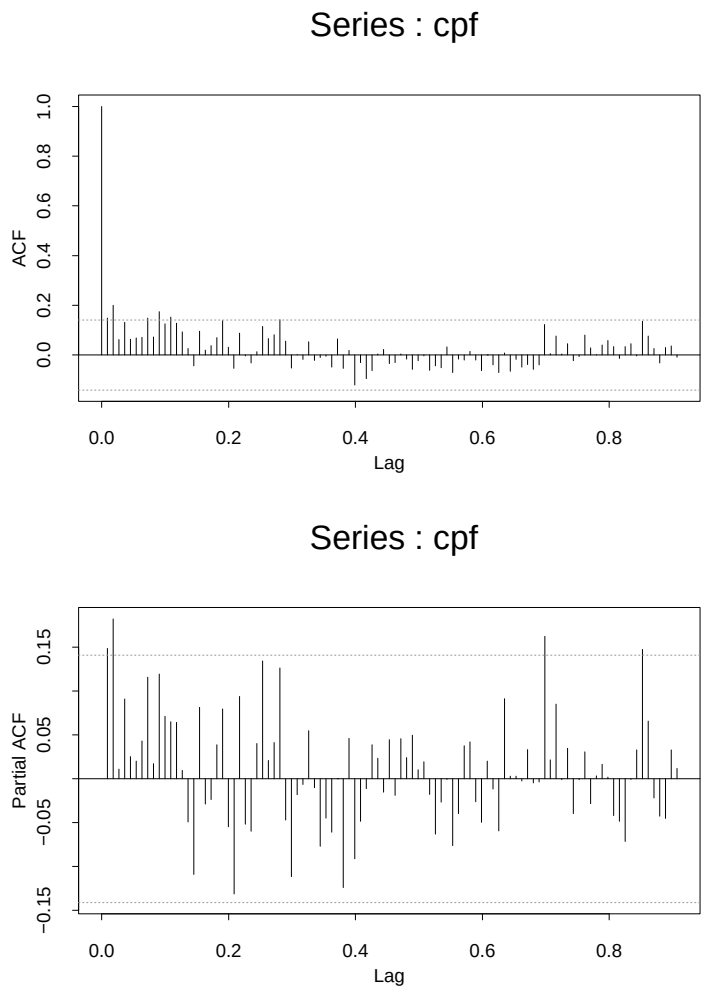

Figure 35: Autocorrelation and partial autocorrelation function for the cpf time series.

According to [8, Table3.3 p.84], we are dealing potentially with an autoregressive

process of order two, or abbreviated an $AR(2)$ process. The reason is that the ACF is infinite, and shows signs of a damped sine waves, and is slowly tailing off. The PACF is finite, and cuts off after the second lag. These two indicators combined give us some trust that the cpf time series is $AR(2)$. In fact, this implies that the productivity data is correlated by two previous values, but not more. So, by accepting its stochastic nature, we immediately found clues for systematic effects. One could say that the time series has a short memory, since the values depend on a few past values only. This is not strange if you think of a software process improvement program. First when you change things the effect will be zero, but after a while you will see effects, and each project things will become a bit better, and while you are doing things better, yet new measures come in, and they will have a lagged effect, and so on. Whatever the theoretical musings on the reasons why, fact is that our time series is not independent identically distributed noise.

For the sake of ease, let us abbreviate cost index per function point with cpf, short for cost per function point. As a first-order approximation, we will fit the cpf-series to an $AR(2)$ model. But before we do this, we like to know whether the visual inspection of the ACF and PACF plots, provided us with the correct order of the model. To that end, we calculated the so-called Akaike Information Criterion (*AIC*) [72], which is a trade-off between parsimoniousness of the parameters and accuracy of the model in terms of residual variance. In other words, if we would use more parameters to describe the model, would the model then be better or not? Akaike came up with a general criterion to provide an answer to this question for many statistical modelling techniques [1]. There are also other information criteria that one could use to test the optimal balance between number of parameters and accuracy of models, but we opted for the *AIC*. To give the reader an idea, we formulate the *AIC*:

$$AIC(m) = \log(\sigma_m^2) + 2m/n + constant$$

Where m is the number of parameters used, σ_m^2 is the maximum likelihood estimate of the residual variance, and n is the number of observations. This approach discourages overfitting (a model with too many parameters). Note, however that the *AIC* itself is known to select higher order models and will overestimate the true order of autoregressive processes, so a number of related criteria have been developed. For our purposes, *AIC* is sufficient. For further information on model selection criteria in a time series context we refer to [8, p. 200–201], or [32, p. 438–439], or other textbooks on time series analysis.

It turns out that the *AIC* is minimal for $m = 2$ for the cpf-series. We depicted the various *AIC*'s in Figure 36. In this plot, vertically we put values of the *AIC* relative to the lowest value resulting in an *AIC* of zero for the minimal value. This minimal value is reached for $m = 2$. Horizontally we set out the number of parameters that are used to fit the cpf-series. So also the Akaike information criterion results in an order 2 process. Therefore, we fit the cpf-time series to an $AR(2)$ model. This results in the following model:

$$x_t = 0.114 + 0.124(0.0711)x_{t-1} + 0.184(0.0711)x_{t-2} + \varepsilon_t \tag{3}$$

with $\sigma^2 = 0.0112$ and the real value for $AIC = -312.45$. Note that the ε_t is an error term that is supposed to be normally distributed, with mean zero, and variance $\sigma^2 = 0.0112$. The numbers in brackets in the above formula are the standard errors of the fitted coefficients; both values indicate that the fitted terms are significant, since the error is much smaller than the fitted value. All in all, we have found an initial

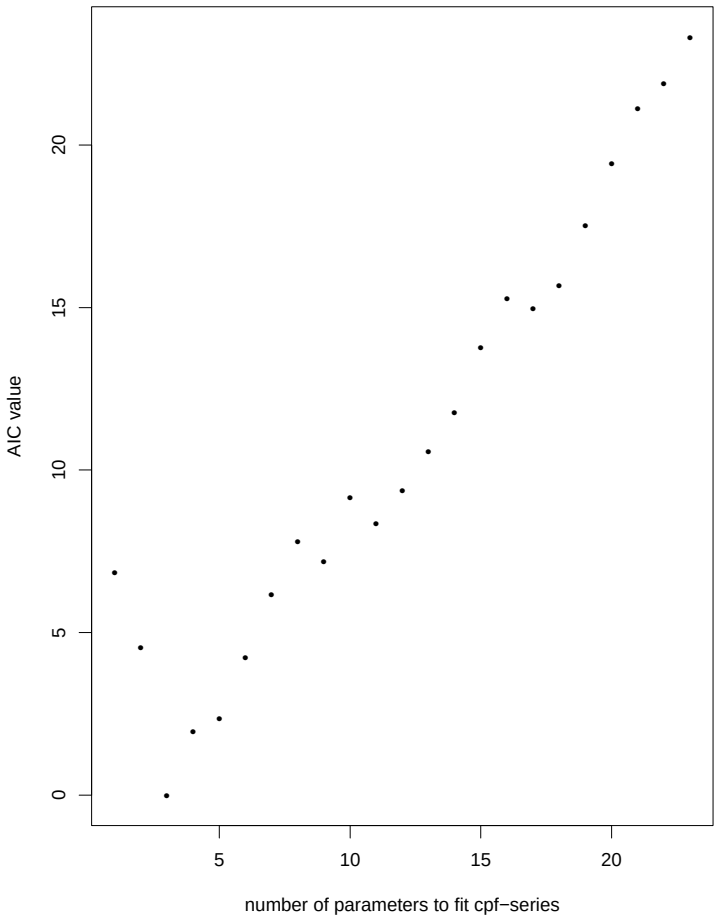

Figure 36: Akaike's Information Criterion for various numbers of parameters to fit the cpf-series.

model using two parameters and an error term. For the latter, we wish to know if there is still structure in it, or that this error term is random noise. One method to get an impression of this is to investigate the residuals of the cpf-time series. One effective means to detect nonrandomness is to plot the so-called cumulative periodogram of a time series. This notion is based on the assumption that a time series is in fact a superposition of sine and cosine waves. It can be shown that a truly random process has a cumulative periodogram that is a straight line [8, p. 321]. In Figure 37, we plot the cumulative periodograms of both the raw cpf-data and the residuals of the $AR(2)$-fit for the cpf-series. We note that the functionality for cumulative periodograms is not a standard part of Splus, the system we mainly use for all our analyses. We imported some code due to [84] that calculates the cumulative periodogram.

If we abstract from what a cumulative periodogram comprises, but for the moment

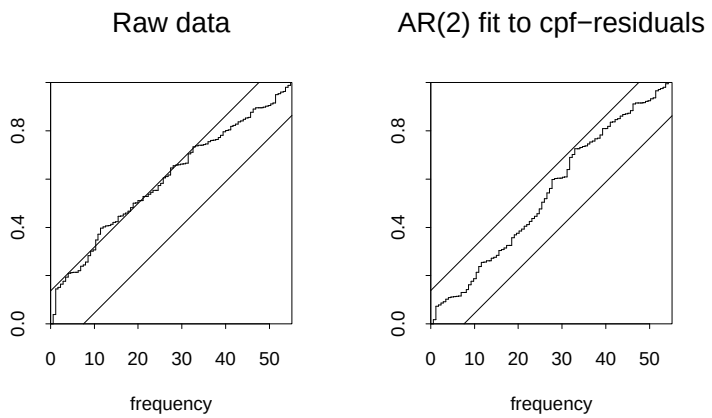

Figure 37: Cumulative periodograms of both the cpf-data, and the residuals of its fitted model.

assume that it is an effective nonrandomness checker, we can conclude that the cpf-data itself is not a random series, since the staircasing line is not a straight line. Note that the two truly straight lines form a confidence band, indicating that if the cumulative periodogram is within those lines, we consider the series indistinguishable from a truly random series. Indeed, for the residuals of the fitted model (that is, the ε_t-part of the fitted model), the cumulative periodogram is within the confidence band, but note that at one point the cumulative periodogram seems to touch the upper confidence band, indicating that there is still some structure left in the error term.

Before we conclude too fast that the residuals have no further structure, we carry out a second diagnostic test and depict this in Figure 38. This is a standard diagnostic giving us four plots. First of all the title of this diagnostic plot mentions *ARIMA*, this stands for *Autoregressive Integrated Moving Average* process. This is the more general

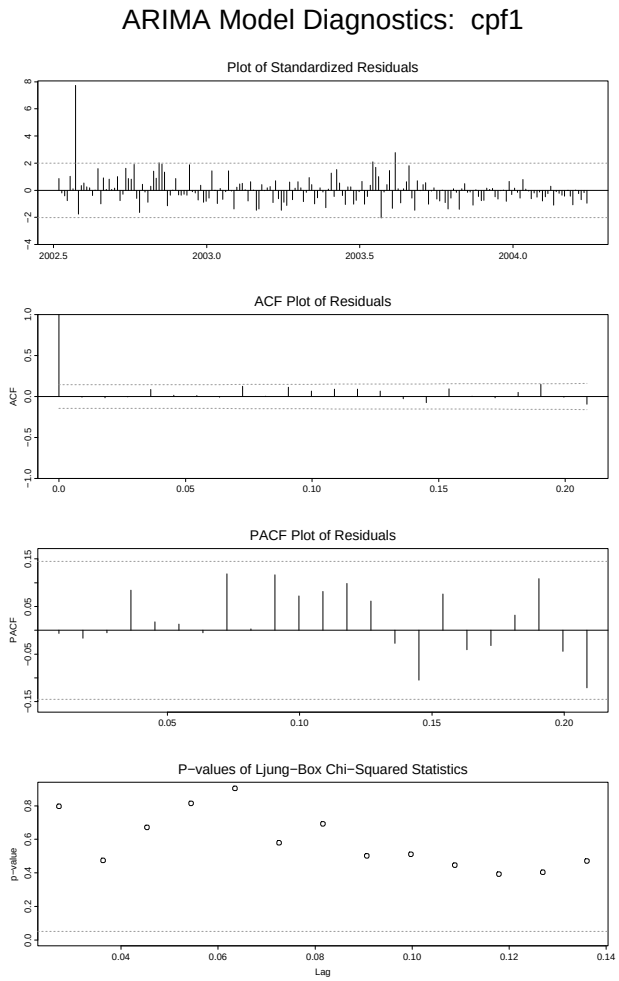

Figure 38: Diagnostics of the $AR(2)$ model for the cpf-time series, including the standardized residuals, the autocorrelation functions of the residuals, and the p-values for the portmanteau lack-of-fit test statistic.

family of process to which the $AR(2)$ process that we fitted belongs. More information on these more general processes can be found in any textbook on time series analysis, but we like the textbook of their advocators best [8]. Then there is mentioning of cpf1, this is a demeaned version of our cpf-series. This is done since the software we use for fitting *ARIMA* models assumes a time series with zero mean. Of course, in our first tentative $AR(2)$ model, we corrected for this, and the cpf-version is displayed, as will be the case for our second model.

Plot 1 of Figure 38, shows the standardized residuals of the error term ε_t. We can immediately see a giant outlier: in the early stage of the time series there seems to be an outlier that is almost 8 standard deviations away from the mean (which is zero in the demeaned version). This is an indication that we need to come up with a more

sophisticated model (this is the subject of Section 7). Plots 2 and 3 give the ACF and PACF of the error term ε_t, to get an idea if there is any *ARIMA*-behaviour left in the residuals. But since for all lags larger than zero, there are no values outside the confidence bands, there seems no autocorrelation left in the residuals. Plot 4 depicts the result of a so-called *portmanteau lack-of-fit test*. A portmanteau word is a new word formed by joining two others and combining their meanings (motel is a combination of motor and hotel, and their combined meaning is a hotel reachable by car). In the same vein, rather than considering the residuals individually, a portmanteau test blends a number of residuals into a whole to indicate adequacy or inadequacy of an *ARIMA* model. The precise formula for the used portmanteau statistic is not too important but this particular test statistic is approximately distributed as a χ^2-distribution, if the model is indeed of the *ARIMA* family (hence the mentioning of that distribution in Figure 38). Most important for our purpose is that the p-values are well above 0.10, implying that there is no evidence to reject the hypothesis that the fitted $AR(2)$ model is inadequate (cf. Table 2). For more information on portmanteau test statistics we refer to [8, p. 314] and for details on what portmanteau test was used exactly in our case, we refer the interested reader to [84, p. 416].

Next, we carry out a simulation exercise. Namely, based on the found model, and the historical data, we can simulate a time series that is supposed to approximate the cpf-time series. If the simulation is way off the actual data, we can doubt the adequacy of the model. You cannot just instantiate formula 3 by filling in some values. Of course there is an error term and we used a random generator to create these so-called innovations. This implies that other simulations are probably a little different from the one that we depict in Figure 39. In this figure we see two plots, plot 1 is the original cpf-time series, and plot 2 is a simulation based on formula 3 and the original data. From this figure we can already spot that the model imitates the flow of the original data.

In Figure 40 we see two other plots: the first is showing us the original data (solid line), and the simulation (dotted line). Although most of the peaky nature of the original data is more or less mimicked, the model is a little off here and there. In addition, it seems that the model is systematically a bit lower at the end of the time series. Nonetheless, for initial purposes this is not at all a bad approximation. Plot 2 shows these differences more accurately. We just subtracted the simulated model from the original data, and the resulting curve shows the deviations. We see that the maximum deviation is about 0.20 which is found in the beginning of the time series. Normally, a simulation needs some warm-up time, so initial values can be more off than further on. And second, the giant outlier that we noted in the residuals (cf, plot 1 of Figure 38), could here also be a problem in the first large peak in the beginning. The horizontal line divides positive and negative deviations. Indeed, we can see more clearly now that the last part of the model is slightly more positive than the actual data, but the deviation is modest: maximally 0.05 off the cost index that ranges from zero to one, so 5% at most.

6.3 A first forecast

Potentially, we can use the inferred model to forecast the cost index per function point. But the forecasting power of this model will not be very high, since the number of data points that was available for model inference was not that high. This means that forecasts will quickly approximate some fixed value, and thus loose forecasting power rapidly. To that end, we use in this series a 9-step forecast, where the number 9 is

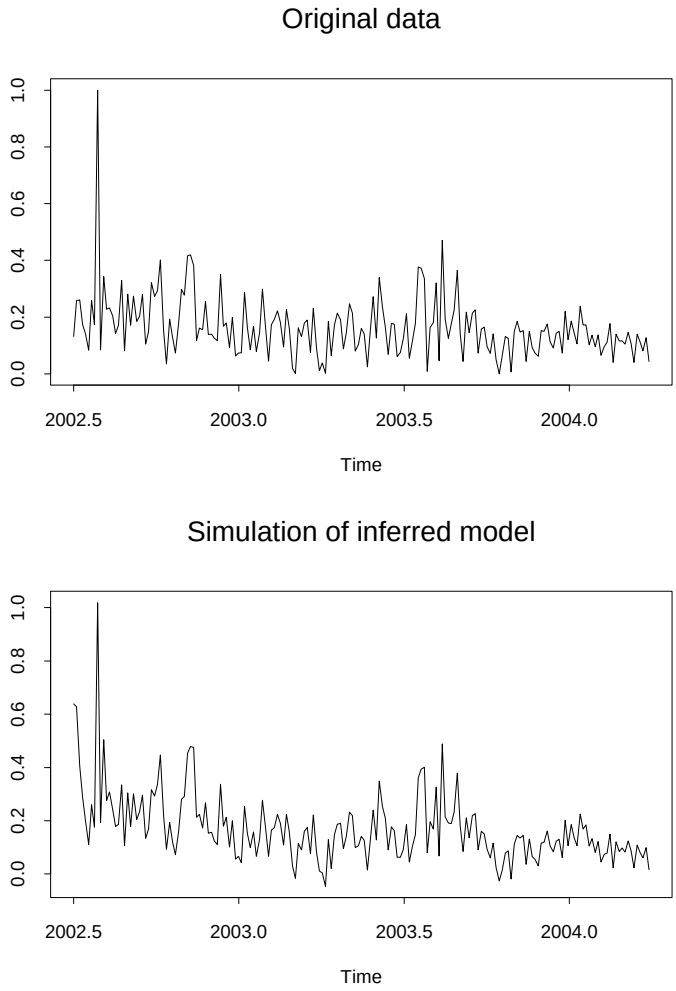

Figure 39: Two plots showing the original cpf-time series and a simulation based on the fitted $AR(2)$ model of the data.

found by inspecting at which number the fixed value is reached. One time slice in our regular time series is about 3.3 days, so a 9-step forecast represents about one calendar month. In order to assess the forecasting ability of the model we start out with an ex post forecast. We took a window of the cpf-time series and made a forecast that we could check with known values. In Figure 41, we depicted the result. The three lines around the beginning of 2004 are the 9-step forecast, plus a confidence band that totals two times the standard error away from the mean forecast. As we can see, there is almost no peaky behaviour, which is not surprising since we have not too much data. But we also see that the true values are trapped by the confidence band, which gives us some trust that a real forecast will at least give us the correct direction the curve will head to.

In Figure 42, we depicted an ex ante forecast. We again based this forecast on a

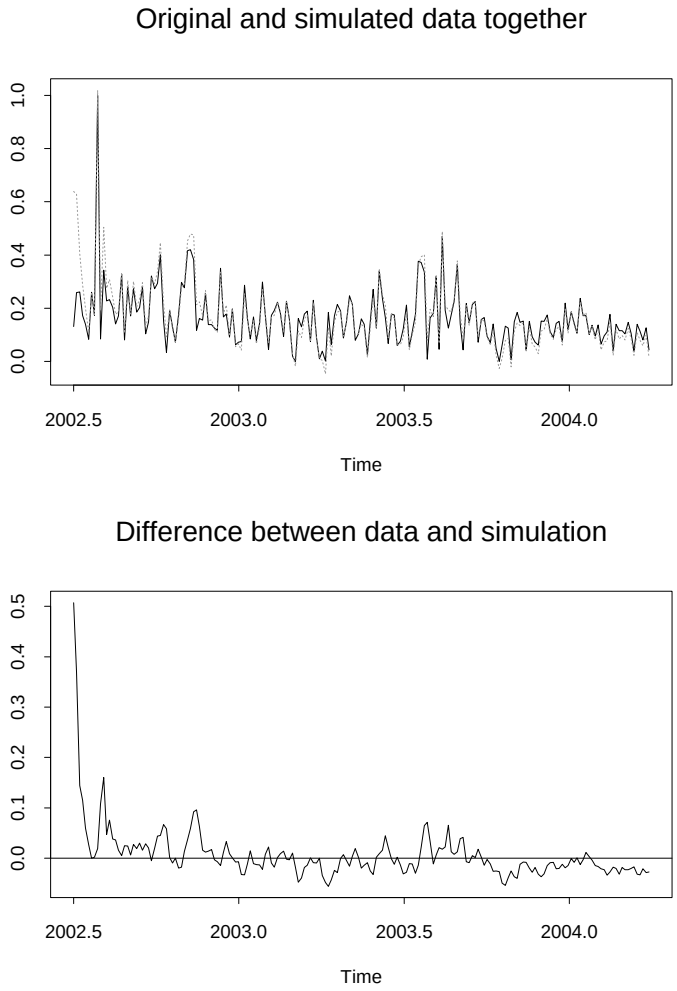

Figure 40: Two plots giving more insight in the adequacy of the simulated model: both an overlay, and the difference between actual and modelled data.

9-step lead time. We recall that the forecasting power of this model is not too high, but the forecast indicates that the cost index per function point will go up in the future. What is reassuring to note, is that indeed the cpf-time series moves up and down all the time, and that this is not too surprising. Moreover the confidence bands give us some indication that the productivity decrease is not going to be staggering, although we can never rule out large outliers, given the heavy tails of the distribution of the cpf-data. For more information on time series forecasting we refer to [43, 8, 16].

All in all, we fitted a reasonable model, but with predictive power that is most probably limited to directions instead of real values. Moreover, there were signs in the diagnostics that we should better not ignore outliers, and since the forecasts show a rise, we should be certain whether this is an effect of the model, or the outliers. This is the subject for the next section.

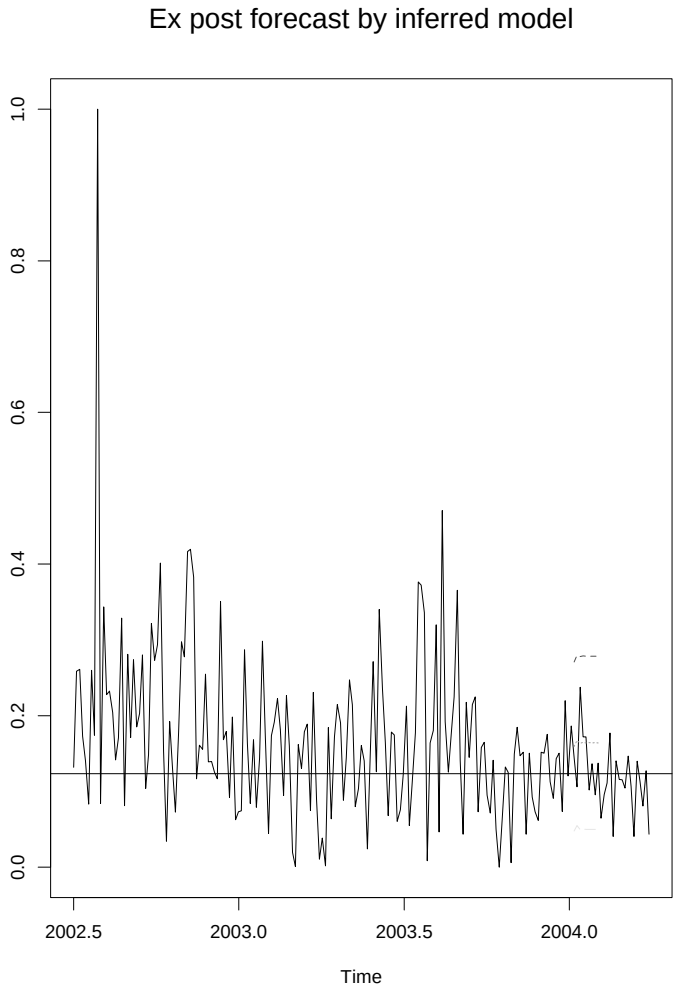

Figure 41: The cpf-time series, a 9-step ex post forecast, plus confidence bands, and the target line.

7 OUTLIERS

Although we made considerable progress in finding a suitable model for the cpf-time series, we also found in one of our diagnostic tests that there are potential outliers, and these outliers can influence the model considerably. In our case, we are interested in answering the question whether the cost index per function point shows a decrease over time, and erroneous outliers can skew such results to a point where you get the wrong answer. So in this case it is prudent to adjust the tentative $AR(2)$ model and take outlier detection into account.

An outlier is not per se a stray value, but can be due to external events such as sudden political or economic change, sudden changes in approach or even physical systems, and of course reporting errors. Often in advance the presence or absence of

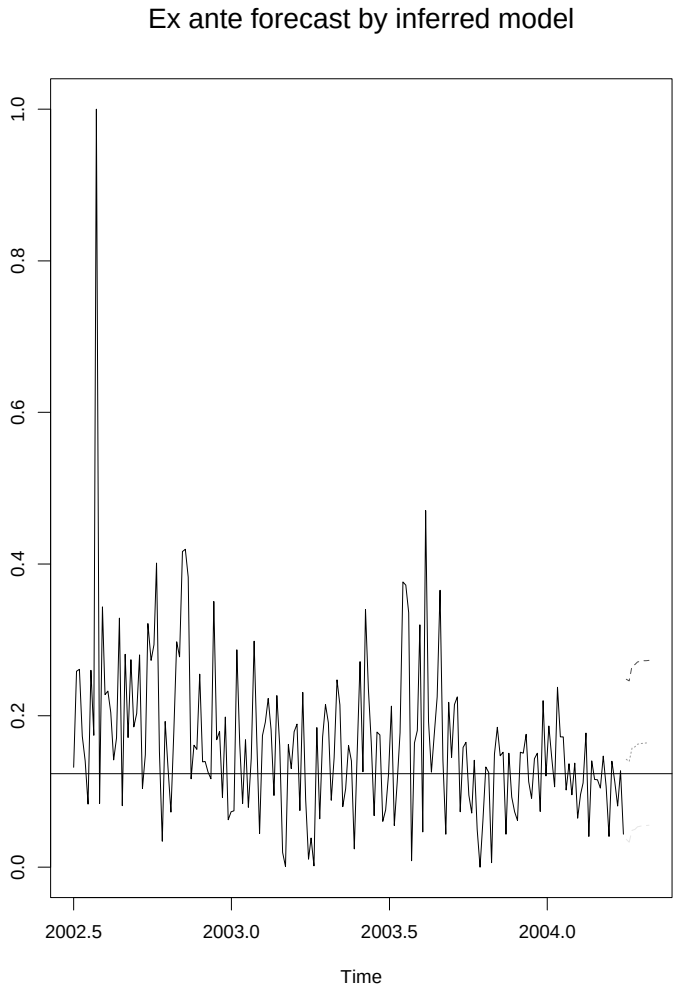

Figure 42: The cpf-time series, a 9-step ex ante forecast, plus confidence bands, and the target line.

such outliers is not known, but they can affect the models substantially, which is why we need to address them. First we explain what an outlier comprises. We follow the treatments of [30, 15, 81, 9]. We identify three types of outliers: additive outliers, innovational outliers, and level shifts. All these types of outliers can be described in a simple fashion. Suppose y_t is a time series void of outliers with error term ε_t, and y_t^* is the observed time series, with error term ε_t^*.

- Additive outliers (AO). If for some time T the observed time series at that time point is $y_T^* = y_T + \omega$, where ω is some constant, we call this an additive or sometimes an observational outlier.
- Innovational outliers (IO). If for some time T the error term is $\varepsilon_T^* = \varepsilon_T + \omega$, we call this an innovational outlier or innovational shock. An IO affects more than

one value for autoregressive time series, since it affected the error term, hence the word innovational.

- Level Shift (LS). If for some time T the observed time series from that point on is $y_t^* = y_t + \omega, t \geq T$, we call this a level shift.

We note that there are alternative formulations for outliers, such as temporary changes (with some exponential decay), and temporary ramps, including a host of special functions ranging from business holidays, trading days, leap years, and other special functions that can and sometimes must be modelled separately to ensure that the models fit best. See [9] for more information.

Recall that our initial effort to model the cpf-time series led to an autoregressive model of order 2 (see formula 3). If we however take outliers into account, this choice is no longer the best. But as a starting point, we carry out a robust autoregressive fit of order 2. This results in the detection of 14 outliers, an intercept, and two autoregressive coefficients. The second coefficient is now -0.0722, and its standard error is 0.0743, which means that this second-order coefficient is not significant. Therefore, we adapt the autoregressive order to 1, and proceed from there. This leads to the following model for the cpf-time series.

$$
\begin{aligned}
(4) \qquad x_t &= 0.1444(0.0070) + 0.8592P_t^9 + 0.2393P_t^{30} \\
&\quad + 0.2533P_t^{39} + 0.2452P_t^{40} + 0.2382P_t^{41} \\
&\quad + 0.3320P_t^{124} + 0.1178(0.0795)x_{t-1} + \varepsilon_t
\end{aligned}
$$

With $\sigma^2 = 0.07767469$, which is much worse than the residual error term of formula 3: 695.5%. So we will have to do something else. But before we do this, we explain the notation in formula 4, since it was useful in identifying the outliers. As before, we start with the intercept, with in brackets the standard error. We denoted the additive outliers as follows. Suppose that ω is an additive outlier at time T, then we can model its impact with a pulse function ωP_t^T, where $P_T^T = 1$, and zero otherwise. The six additive outliers found in the robust fit for the cpf-time series data are modelled this way. They are detected with reasonable so-called t-values. For now it is important to know that the critical value in order to give a value the outlier status is 3, and the outliers that we found have the following t-values: 11.13, 3.274, 3.409, 3.231, 3.292, and 4.389 respectively. Indeed, the two largest t-values also have the largest impact. If outliers have little impact, you can decide to drop them. Note that we see more outliers in the beginning of the series, and less when time passes. Then we see the autoregressive term, with a small standard error, indicating reasonable significance, followed by an error term, that is supposed to be unstructured white noise.

So, although formula 4 revealed the outliers, the robust fit did not lead to an improved residual error term. Since we wish to know whether the initial forecast is consistent if we do take outliers into account, it is necessary to work with a much smaller standard deviation for the residual error term than the found one. To solve this issue, we carry out a fit by hand, meaning that we must write some statistical code to explicitly regress the found outliers out of the data, and on that residue, carry out an *ARIMA* analysis, leading to an autoregression. This combination of a regression and an autoregression is called a *REGARIMA*-model. We use the found outliers in formula 4 as deterministic regressors, and we carry out an *ARIMA* analysis on that system. This leads to the following formula, which is a slight variation of formula 4, but with a much better residual error.

$$
\begin{aligned}
(5) \qquad x_t &= 0.15303 + 0.85619P_t^9 + 0.22119P_t^{30} \\
&\quad + 0.23900P_t^{39} + 0.26293P_t^{40} + 0.23617P_t^{41} \\
&\quad + 0.33046P_t^{124} + 0.2002(0.0707)x_{t-1} + \varepsilon_t
\end{aligned}
$$

With $\sigma^2 = 0.006105119$, which is an improvement of 182.9% over the residual error term of formula 3. We will use formula 5 instead of formula 4, to validate our initial forecast with this model.

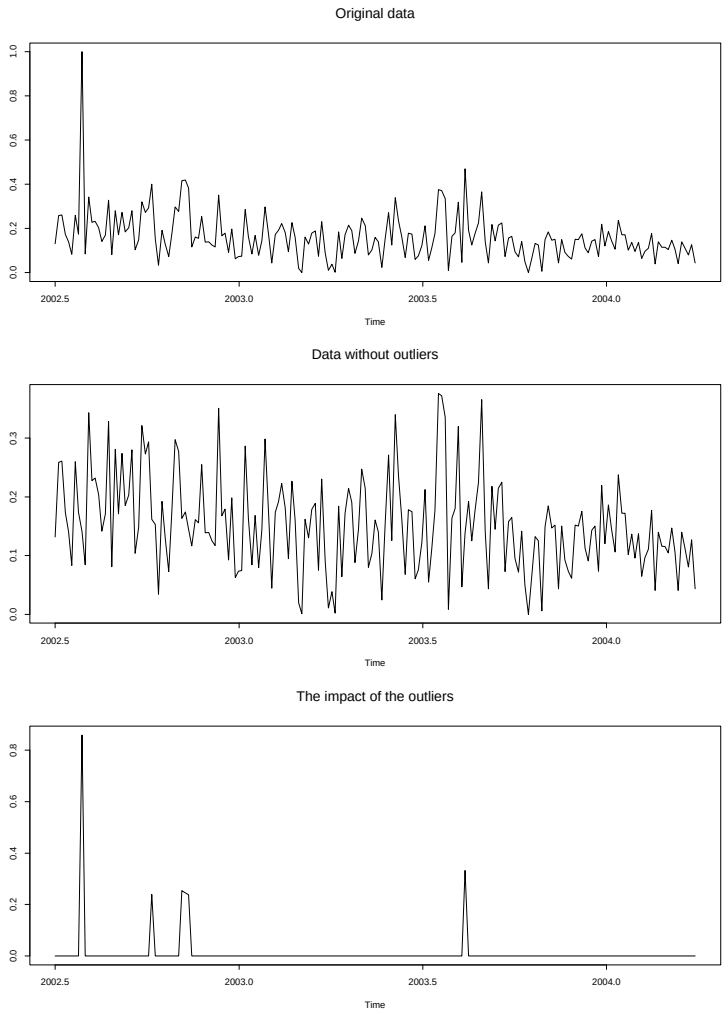

Figure 43: The original cpf-time series, the one without detected outliers, and the impact of the outliers.

Let us visualize the model represented by formula 5. First we display in Figure 43 three plots: in plot 1 we displayed the original data, and in plot 2 the same data but without the outliers. In plot 3 we depicted the following part of formula 5:

$$0.85619P_t^9 + 0.22119P_t^{30} + 0.23900P_t^{39} + 0.26293P_t^{40} + 0.23617P_t^{41} + 0.33046P_t^{124}$$

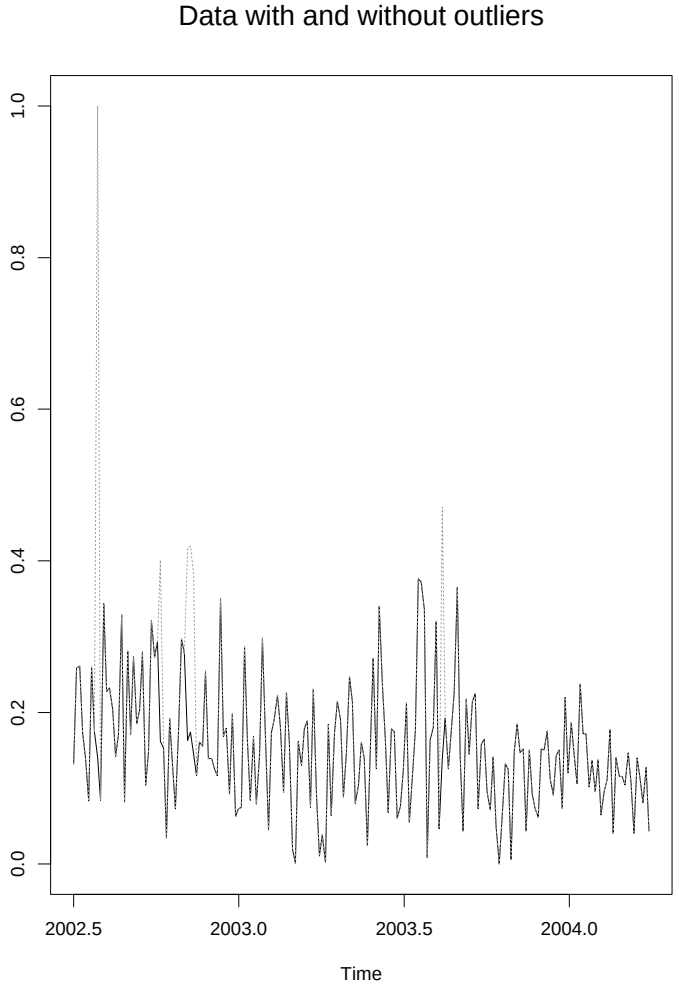

Figure 44: The cleaned cpf-time series and dotted the original time series.

In fact this is the overall pulse function that we fitted. Indeed if we subtract this pulse function from the original data we obtain plot 2. To illustrate this further, we plot both the cleaned data and the original data in Figure 44. The cleaned version is the solid line, and the original version is the dotted line. As we can see, only on a few locations the two curves are different. These locations are the outliers, of which there are only six.

As with the first approximation of the cpf-time series, we will once more look at the quality of the inferred robust outlier-aware model. For a start, we will provide a cumulative periodogram of the residuals of the $AR(1)$ fit of the cleaned cpf-series. Recall

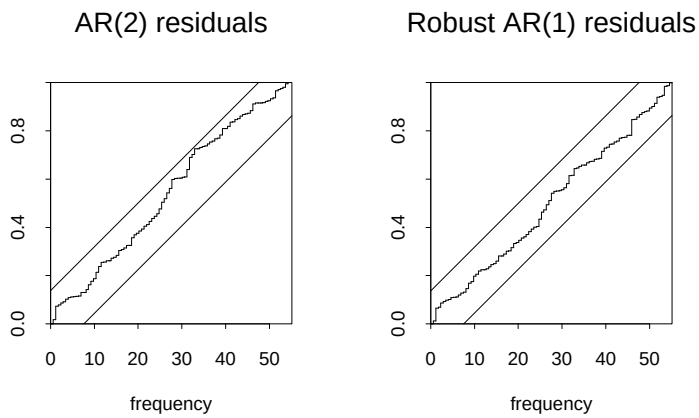

Figure 45: Cumulative periodograms of both the the residuals of the fitted model without and with outliers taken into account.

that if is this is a straight line, the residuals do not contain too much structure anymore. For comparisons, we plot in Figure 45, two cumulative periodograms: plot 1 depicts the cumulative periodogram of the residuals of the $AR(2)$ fit of the model where outliers are not taken into account. Recall that although the cumulative periodogram is between the confidence bands, the staircasing line touches one of them. In plot 2 we depict the cumulative periodogram of the residuals of the outlier-aware model. Indeed this cumulative periodogram is "more straight" than the left one, although formally, there is no difference, given the fact that they are both within the confidence bands. But still, this indicates that the residual structure that was suspected in our first model is reduced in the second model, which is an indication that the second model is better.

In Figure 46, we depicted the original cpf-time series again, and a simulation of the second model including the outliers as regressor variables. From plot 2 we can see that

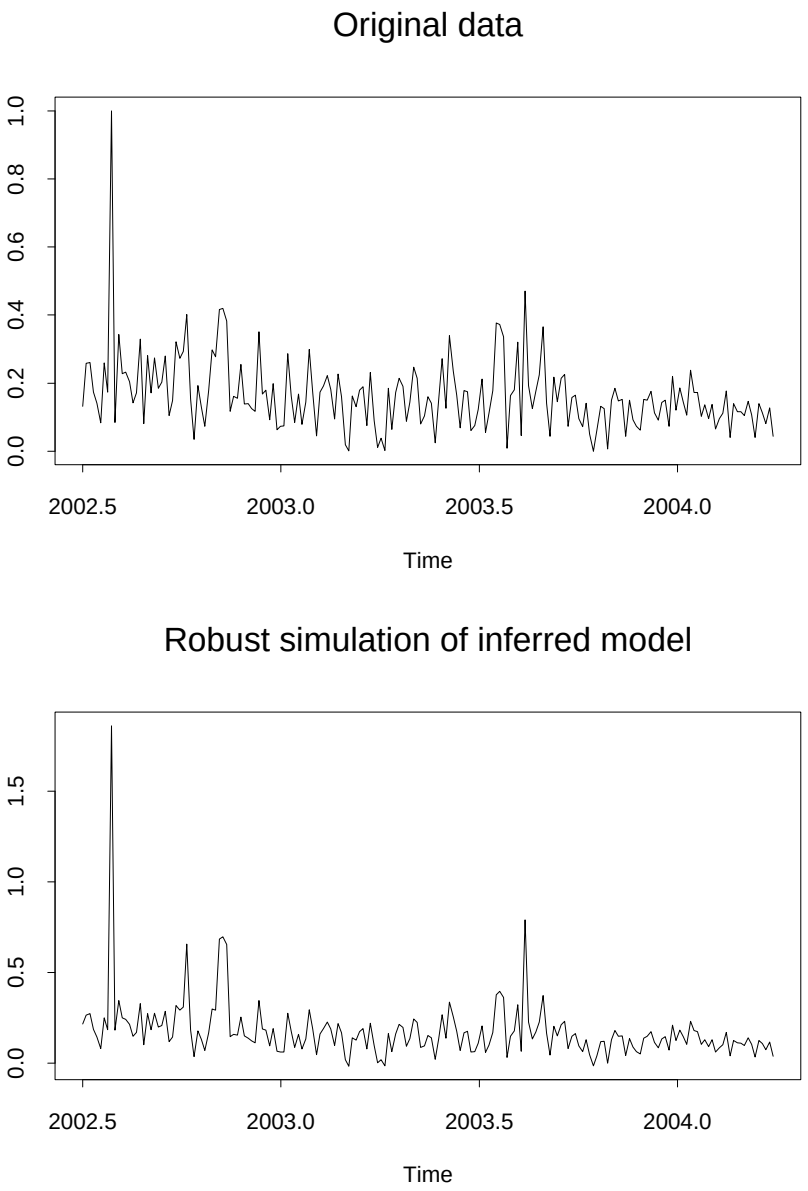

Figure 46: Two plots showing the original cpf-time series and a simulation based on the fitted outlier-aware *REGARIMA* model.

apart from the starting values which can be different since the simulation has to "burn in" a bit, this seems to follow the original data of plot 1 pretty well. We used a so-called *REGARIMA* model for that, which means that there are deterministic regressor variables, and a stochastic part that is supposed to behave like some *ARIMA* process. The deterministic part is formed by the found outliers, and the residual time series where the outliers are removed is then hopefully better fitting to an *ARIMA* model. Our

experience is that this is indeed the case, although the regressor variables will now also be subject to innovations of the stochastic residue of the *ARIMA* model. In other words, the peaks in plot 2 of Figure 46 are not necessarily precisely mimicking the peaks of the original data. This can be seen if you compare the scales of both plots. The large peak in the beginning of the time series is amplified in the simulation, and strongly deviates from the original. But the good news is that outside the 6 points in the time series where the regressor variables dominate the shape of the series, the fit is much better than the initial $AR(2)$ model. In Figure 47 this is reflected. In plot 1 we overlaid the original data with the simulation of the robust model, which indeed shows that on the outlier locations there are differences. But further it is hard to detect deviations between the model and the original. In order to gain insight in these differences, we depict in plot 2 the difference between the original and robust simulation. The solid straight line is the zero line, and ideally the difference should coincide with this line. Now we see that outside the outliers this difference is indeed very small, indicating that this model is very accurately describing the autoregressive part of the cpf-time series. If we compare this to the plots for the first fit, where the outliers were not explicitly taken into account, you will see in Figure 40 that the overall fit of the initial model is not as good as this fit.

Let us look a bit more at both models. In Figure 48, we compare them. In plot 1 we depicted the two models simultaneously, the solid curve is the first model, and the dotted line is the outlier-aware model (notice the larger peaks). In plot 2 we depicted the difference between the two models: indeed around the outliers the difference is substantial: the first model is better on those 6 points, and outside the outlier-regions the models differ within some reasonably small bandwidth. But still the worse fit for the outliers, and the better fit for the cleaned series make the second model the better of the two. This was also expressed in the residual error term that has a smaller standard deviation in the second model than in the first model. So we conclude that we have to model the audit series in the same way as the cpf-time series in the research set: fit an outlier-aware *ARIMA* process to it. We note that in this phase there is time to investigate whether the found outliers are erroneous values, or that they are real. In the final audit, there will be no time to assess the validity of outliers. So in the final phase we need to find trends in both the audit set and the cleaned audit set, and compare differences between the two.

7.1 A second forecast

Now that we have a better model, it is interesting to see whether the first forecast is confirmed or not. Recall that in Section 6.3 we forecast on the basis of the initial model that the cost per function point would rise. Since it was important to know for sure whether this was not an effect of outliers, we came up with this new model. And now we are in a position to do another forecast. In Figure 49, we depicted two forecasts: in plot 1 an ex post forecast. This one shows that the values of the cpf-time series will go down, which is true. This forecast is slightly better than the ex post forecast of the first model which kept the cpf-time series more or less constant (compare with Figure 42). The ex ante forecast, shows fairly similar to the earlier ex ante forecast that the cpf-time series will go up, but just as in Figure 41, the predictive power in the long run of the forecast is not too high.

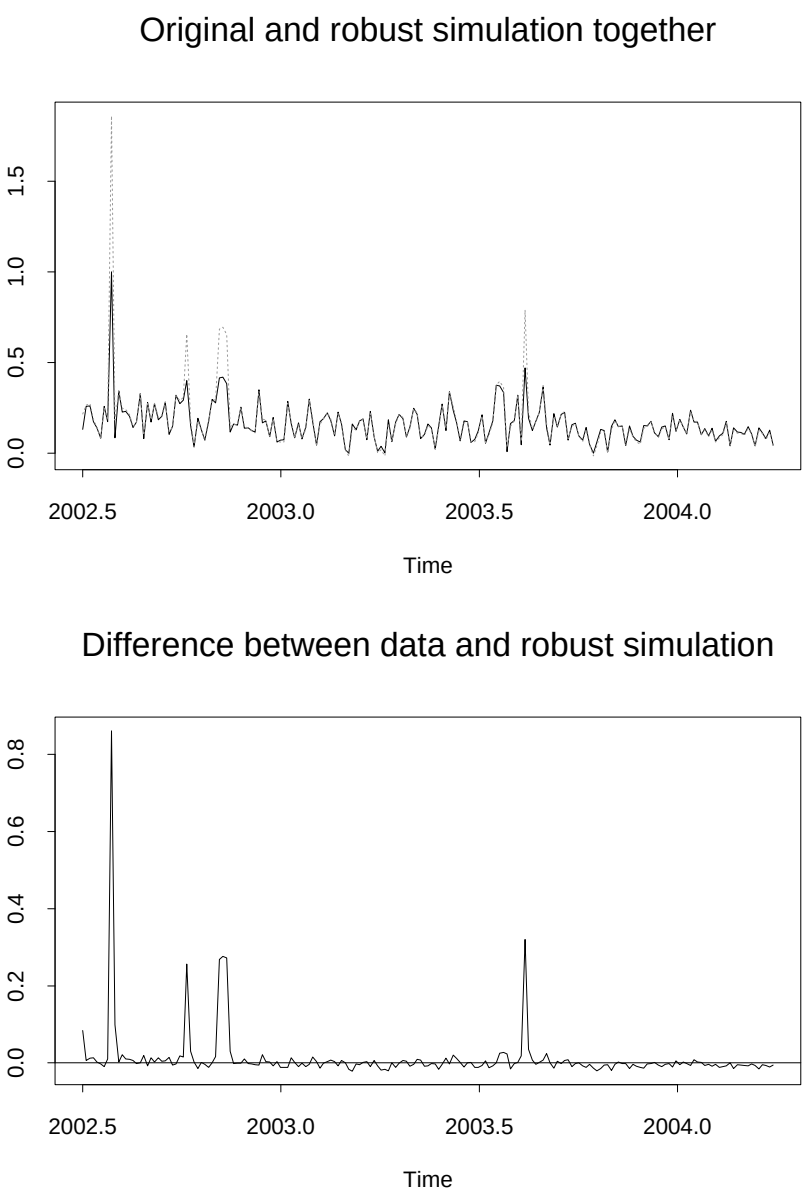

Figure 47: Two plots giving more insight in the adequacy of the simulated model: both an overlay, and the difference between actual and modelled data.

7.2 Outlier impact

At this point we like to illustrate the potential impact on the presence of outliers in the cost index per function point series. That is, the impact on trends that might be detected from the raw time series data. First of all, these outliers represent 1.26% of the total amount of function point counts. But for some reason that we do not know (yet), their

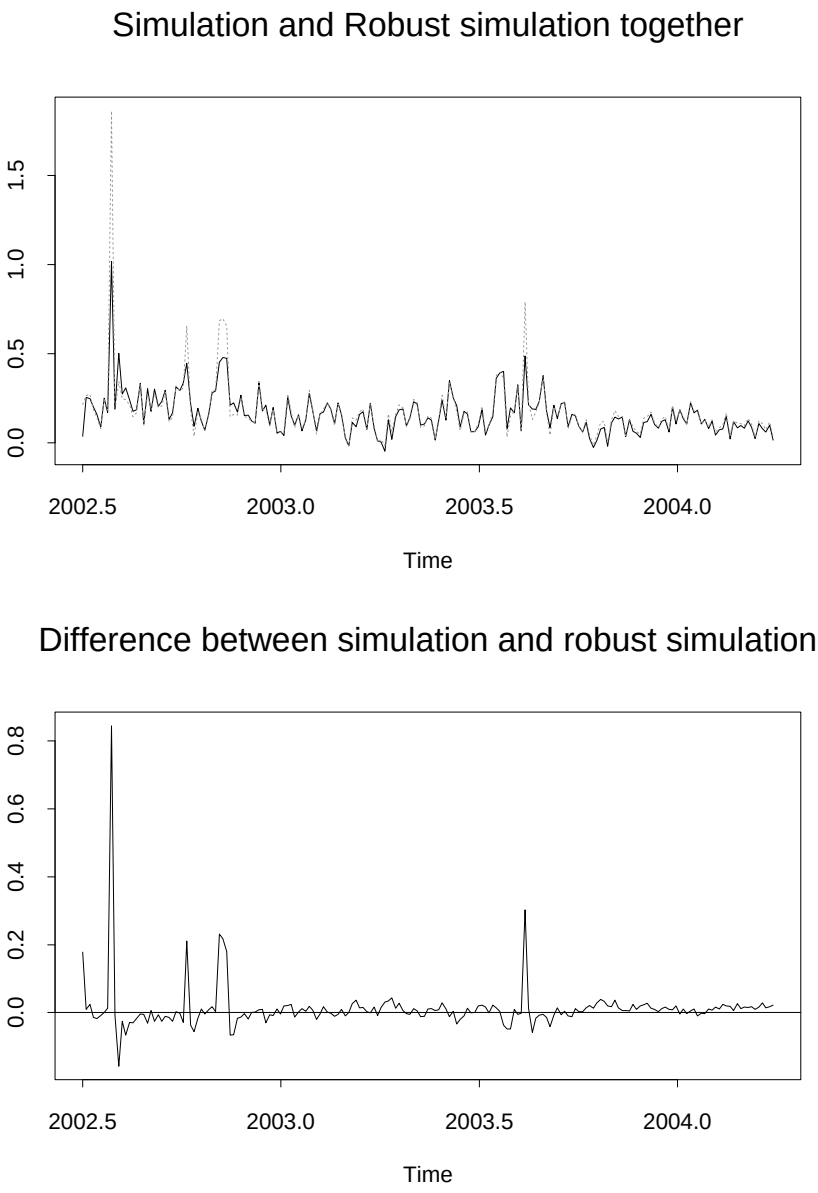

Figure 48: Comparison of both simulations of the first and second model for the cpf-time series.

cost per function point is too high, when time-dependency between IT-projects is taken into account. The latter means that the costs per function point are not necessarily too high in isolation, but are too high given the time when the numbers were reported. Of course, an IT-project in the beginning has a lower productivity than further on in the time series, so an outlier in the end is not necessarily an outlier in the beginning. Only a sophisticated time-dependent outlier analysis will reveal such effects, which is exactly

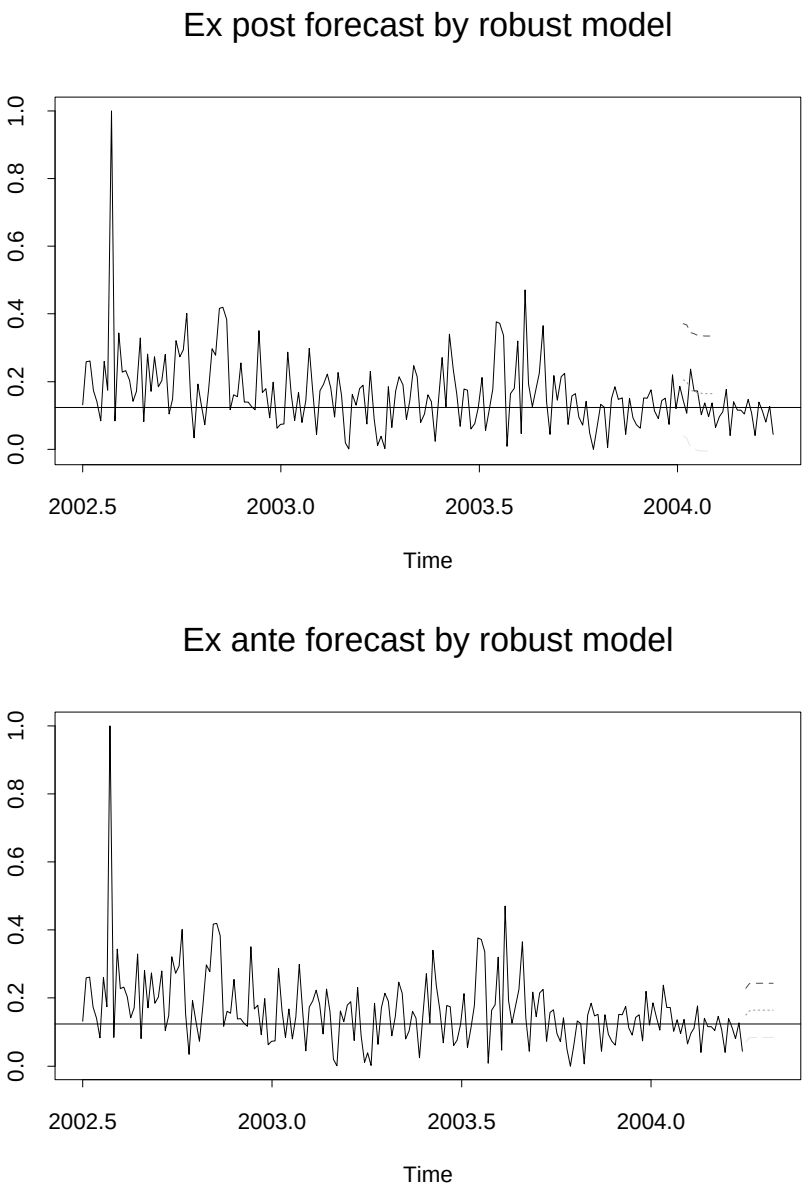

Figure 49: The cpf-time series, a 9-step ex post and ex ante forecast, plus confidence bands, and the target line.

what we did above.

Suppose the 6 outliers are erroneous, and must be removed from the data. Let us compare the moving averages that we depicted in Figure 33 with the same moving averages that would ensue for the cleaned data set, just to investigate the impact of outliers on the initial naive approach of the organization (cf. Section 6.1). Figure 50 summa-

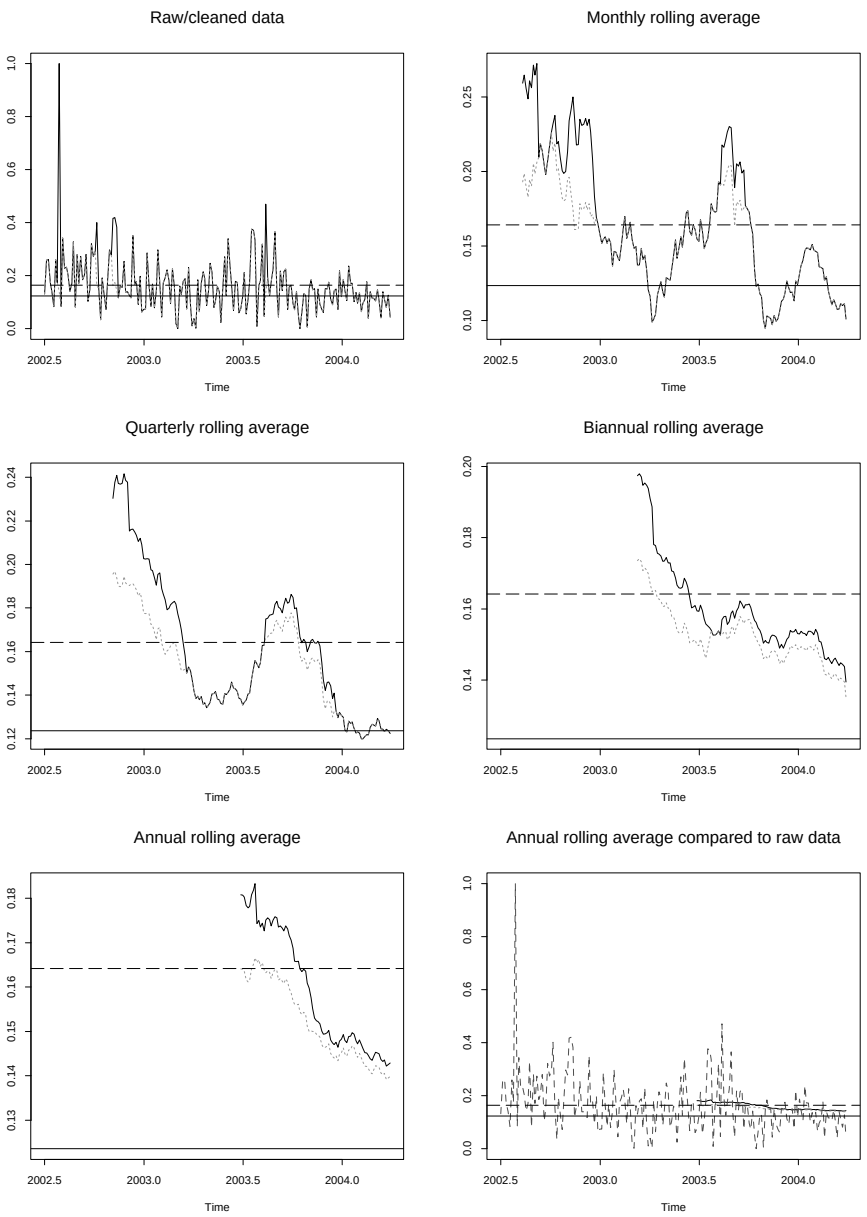

Figure 50: Different moving averages, for both the original data and the cleaned version without the detected additive outliers.

rizes visually what the impact is. Plot 1 shows the raw data with and without outliers. Since the outliers are mostly in the beginning of the cpf-time series, and are all positive in value (but negative in what we want to achieve: lower cost per function point), we can spot immediately in plot 2 that the monthly moving average of the cleaned version is at the beginning much lower than for the original data. When we calculate the maximum of the monthly rolling average of the original data we obtain 0.2722899, likewise

the maximum value for the cleaned monthly rolling average is 0.2264712. The difference is about 4.58%. Suppose a cost index per function point improvement of say 10% was the target, then these 1.26% function points influence the outcome by almost 50%. Despite the fact that you should not use such smoothing operations, it is clear that the 500 function points of IT-projects causing this need to be assessed, to find out whether the reported values are correct or not. In plot 3, the quarterly rolling averages show a similar pattern: the starting point of the cleaned version is lower, its maximum value being 0.1967611, while for the original data this amounts to 0.2416791, a difference of 4.49%. In plot 4, the effect diminished more: the maximal values are 0.1979626, and 0.1741276 respectively, leading to a difference of 2.38%, which sounds small, but still is about 25% of a 10% target. Finally in plot 5, the annual moving average shows a maximum value for the original data of 0.1833147, and for the cleaned data 0.1666302, a difference of 1.67%, which is still more than 15% potential beefing up of the target. Note that on the scale of the raw data this difference is barely visible: both annual rolling averages are shown in plot 6 of Figure 50 and they seem identical from a distance. So once again we see that the annual moving average approach is not what you want, and depending on the time window you use for the average, you will see more or less problems caused by outliers.

aggregate	$\max_o$	$\max_c$	Δ (%)	$\min_o$	$\min_c$	Δ (%)	$\%_o$	$\%_c$
none	1	0.376	62.4	0	0	0	100	37.6
monthly	0.272	0.226	4.58	0.0945	0.0945	0	17.8	13.2
quarterly	0.242	0.197	4.49	0.12	0.12	0	12.2	7.68
biannual	0.198	0.174	2.38	0.14	0.135	0.431	5.84	3.89
annual	0.183	0.167	1.67	0.142	0.139	0.302	4.11	2.75

Table 8: Numerical summary of differences in maximal values of the various rolling averages with and without outliers.

We could have expected also level shifts and innovational outliers from a software process improvement program. For, we would expect that certain software process improvements lead to different random behaviour, of which innovational outliers are an example. Just as a machine wears out leading to innovational outliers, the "machine to produce software" is altered which could also lead to such outliers. Another type we are missing is the level shift. You would expect that when a software improvement measure resorts an effect, this would lead to a lowering of the cost index per function point which is represented by level shifts. But we have to be careful with theoretical clarifications of presence or absence of certain outliers. For, the algorithms to model time series optimize towards parameter parsimony. To illustrate this, we can find innovational outliers, but the model contains just more parameters but not more accuracy. To give an idea, we depict the ACFs and PACFs of both the innovations of our model with 6 outliers, and a model with 13 outliers. As can be seen in Figure 51, there is no noticeable difference, which is an indication that the 13 outlier model is not adding anything.

We gain yet more insight with Figure 52, where we provide the two cleaned time series and their difference. In plot 1 we recall the original cpf-time series. Plot 2 shows the cpf-time series but then cleaned from the 13 found outliers. Note that the vertical range is about 0.4. Plot 3 shows the cleaned version when the 6 additive outliers are removed from the original data; note that the vertical range seems smaller than in plot 2.

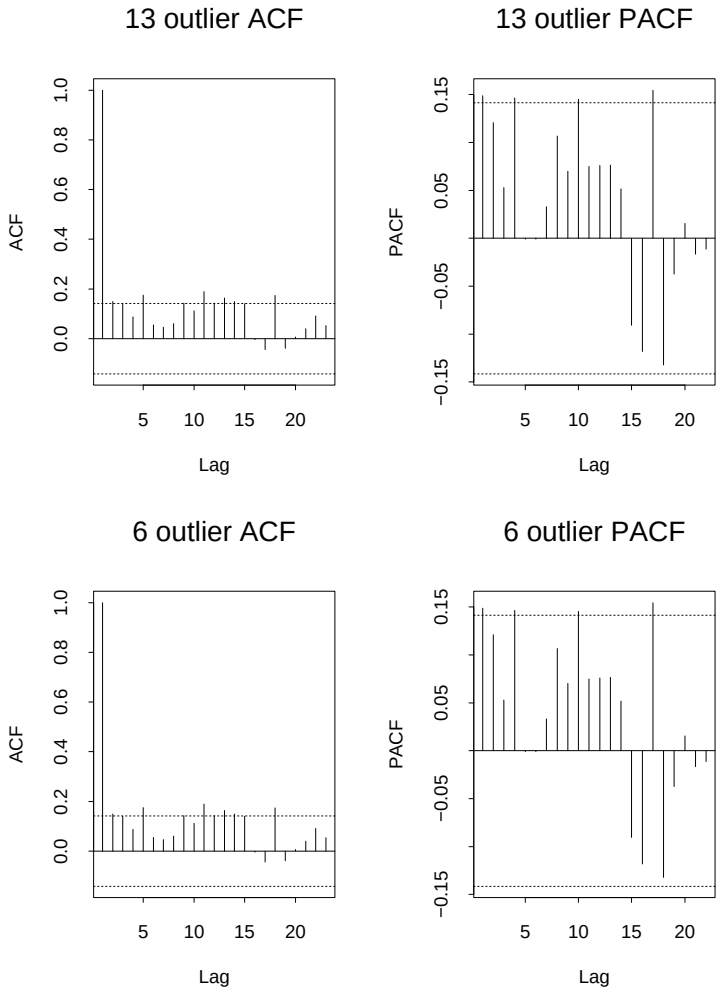

Figure 51: Autocorrelation and partial autocorrelation functions for two models of the cpf-time series: one with 13 identified outliers, and one with 6 outliers.

Finally in plot 4 we depict the difference between the two series. Note that there are very small differences on only a few locations. For most of the time, both models fully coincide.

In Figure 53, we graphically displayed the outliers identified in both models. Plot 1 shows the additive and innovational outliers, totaling to about 13 exceptions to be taken into account for the time series model of the cpf-data. Plot 2 shows the outliers with their impact for the additive outlier-only model. So, strictly speaking there are innovational outliers, what we would expect, but the autoregression will take care of those if the 6 additive outliers are regressed out from the time series, and more exceptions are not necessary to model the time series adequately. However, from an auditing stand point of view, the 13 outliers are interesting since they are candidates from a more thorough further qualitative analysis like recounting of function points, assessing the

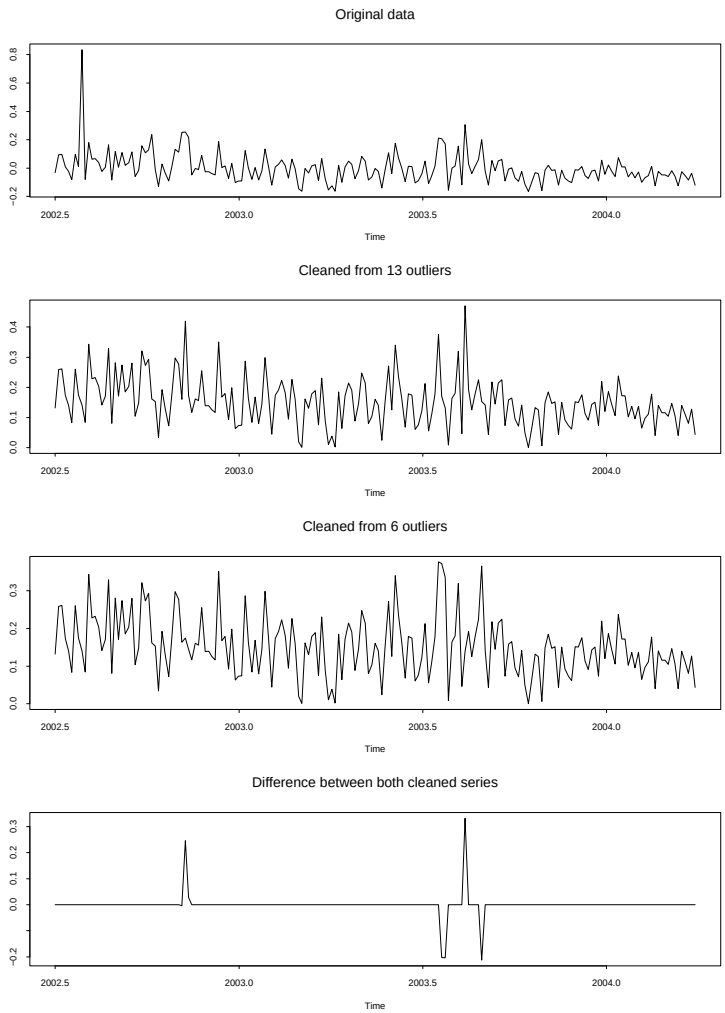

Figure 52: Four plots giving an idea of the differences between two cleaned versions of the cpf-time series.

booked hours, and so on. But from a modelling stand point of view, they are adding complexity without more accuracy.

We can also find level shifts. To illustrate this point, we used another system for outlier detection: the X12 system of the U.S. Bureau of Census [9]. When we modelled the data using X12, we found three additive outliers and five level shifts. Three out of five level shifts have a negative impact, meaning that the cost index per function point lowers from certain points in time on. Similar to our analysis of the 13 versus 6 outlier model, it can be shown that the X12 model is not more adequate than the 6 outlier one, but this is a lot of work, since X12 is not yet that flexible (the current version number is 0.2.10). Again, the level shifts can be interesting from an auditing viewpoint. From a mathematical viewpoint, we are confident that our 6 outlier model is optimal in terms of adequacy and parsimony of the parameters.

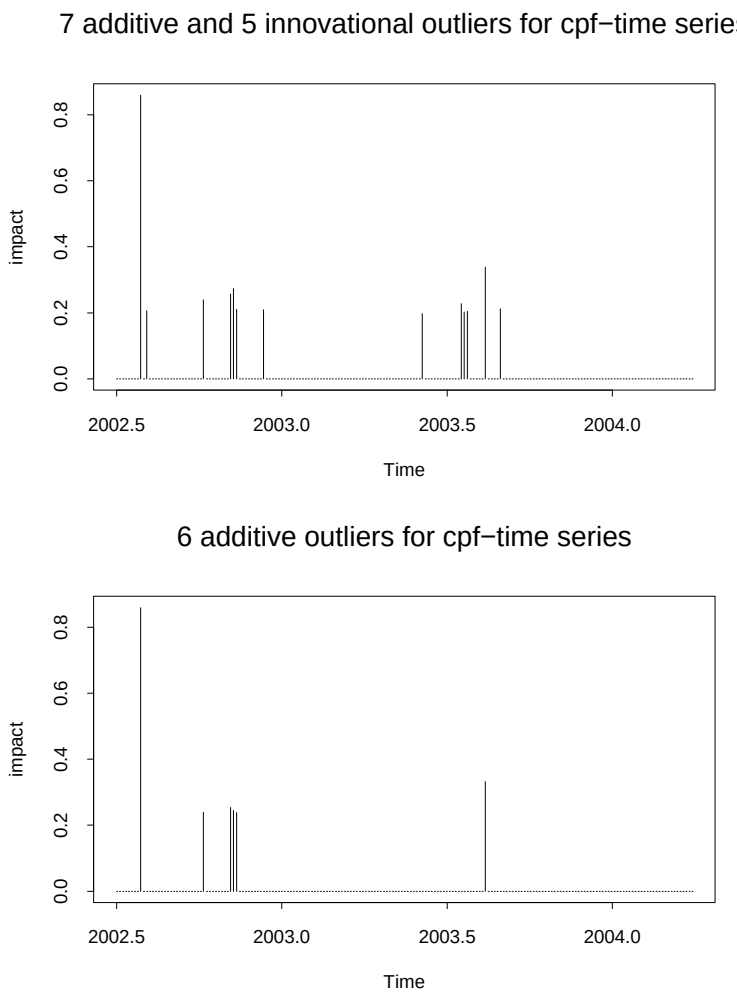

Figure 53: Comparison of the impacts of the 13 outlier model and the 6 outliers model of the cpf-time series; although there are differences, they are not significant, since the autoregression fit deals with the small impact outliers pretty well.

7.3 Single number strategy

Most managers are not fond of long analyses, like the one in this paper. If you concentrate a complex issue into a single number, you have abstracted so much away from the problem, that it becomes all too dangerous to draw conclusions based on this single number. We will now address the question how we might catch a software process improvement project in a single number using the proposed method by the organization. We will show that this leads to arbitrary results, and only with a lot of prudence we can find an adequate description of the actual situation.

To that end we summarized some numbers in Table 8 for all aggregates of Figure 50. They are: no aggregate (plot 1), monthly, quarterly, biannual, and annual

aggregates. In Table 8, we use o as a subscript for original data, and c for cleaned data. In column two ($\max_o$), the maximal value for all aggregates is listed. Note that for the raw data this is 1, since we use a cost index per function point ranging from zero to one. Column three ($\max_c$), does the same for the cleaned data, and in column 4, we give their difference as a percentage. Note that this difference decreases for longer aggregates, which is maybe not what you want. In column 5 ($\min_o$), we provide the minimal value of the original data for all aggregates, and obviously this is zero if no aggregate is taken. Column 6 ($\min_c$) gives the same for the cleaned values. The next column gives the delta's (Δ) as percentages. We see that the differences in the lower end are very small. In the last two columns we provide the maximal difference as a percentage between maximal and minimal value for both original and cleaned data for all aggregates. As can be seen, these percentages are smaller for longer aggregates, and if the target is to improve with say 10%, then an annual rolling average will not establish this, but a quarterly with the original data, and a monthly with cleaned data will. This shows once again that the rolling averages can give us answers depending on the time window of the average, which is an unwanted situation.

But if you must use this method, for instance since you negotiated this in a large outsourcing deal, what can you do? Given the short memory of the time series and the large dispersion of the data, we propose as a measure for software process improvement to look at a monthly rolling average. This is local enough for the amount of measurements within that time frame (about nine), and descriptive enough as a graph. If we look again at plot 2 of Figure 50, we can conclude a number of issues. For both the original data and the cleaned data we see that there are fluctuations. The first tops are the highest, then we see a short minimum under the target line, then the next tops are both lower than the first tops, and we see another longer minimum under the target line, and this pattern reiterates: the last tops are again lower than the second tops and again a long time both graphs are under the target line. Also note the short dip when both graphs go to the second and third top, of which the last dip is also under the target line. This indicates despite the wild behaviour of the raw data, that there is some consistent movement downwards to the target line. One way of capturing this in a single number is to give the range of the curves, either for the original data, or the cleaned, just to be sure if the outliers were erroneous after all. In this case, we think that given the extended analysis, the last two columns of Table 8, and especially the shorter aggregates (monthly, quarterly), might provide an adequate single digit summary of the actual situation. But since this approach is unsatisfactorily, given the dependencies for time windows, outliers, etc, we will propose an alternative manner in Section 9. But before we do this, we are going to investigate the cpf-time series once more, since the residuals of the found model are maybe not serially correlated anymore, but they do potentially contain more structure than independent identically random noise.

8 HETEROSCEDASTICITY

Heteroscedasticity is that the variance over time is not constant. During software process improvement programs, it is often assumed that the large variance of the measured productivity indicators will decrease over time when there is more experience with measurements, and when more accurate reporting structures become ingrained in the organization. In mathematical terms, there is time varying variance, which is also known as heteroscedasticity. Indeed, the large variation in the beginning of the cpf-time series and the smaller variation at the end of the series are an indication that the

variance over time is varying itself. In this section, we check for this, by investigating the residuals of the cpf-time series a bit more. Recall, that the residuals are assumed to have no serial correlation anymore, so that they are independently and identically distributed. If the residuals are normally distributed, this implies then that the residuals carry no more information than plain white noise. A simple visual test for normality of the residuals is to make a Q-Q plot as depicted in Figure 54. We can readily spot that the lower and higher quantiles are not on the straight line, indicating that there is evidence for heavy-tailed distributions in the residuals. And this is a sign that there might be more structure in the residuals left.

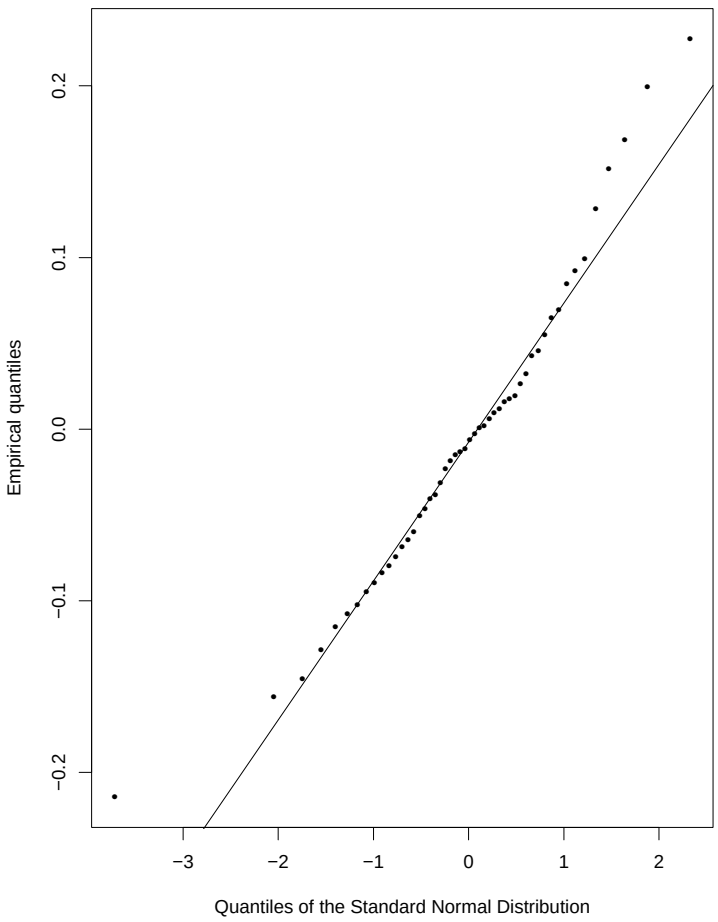

Figure 54: Q-Q plot of the residuals of the initial $AR(2)$ fit of the cpf-time series.

If the variance is varying, this implies in fact that the second moment of the time series is potentially autocorrelated. We can test this in a visual way by plotting the autocorrelation functions of the squared residuals. In Figure 55, we depicted 6 plots. Plot 1 contains the residuals of the nonrobust $AR(2)$ fit for the cpf-time series. Note

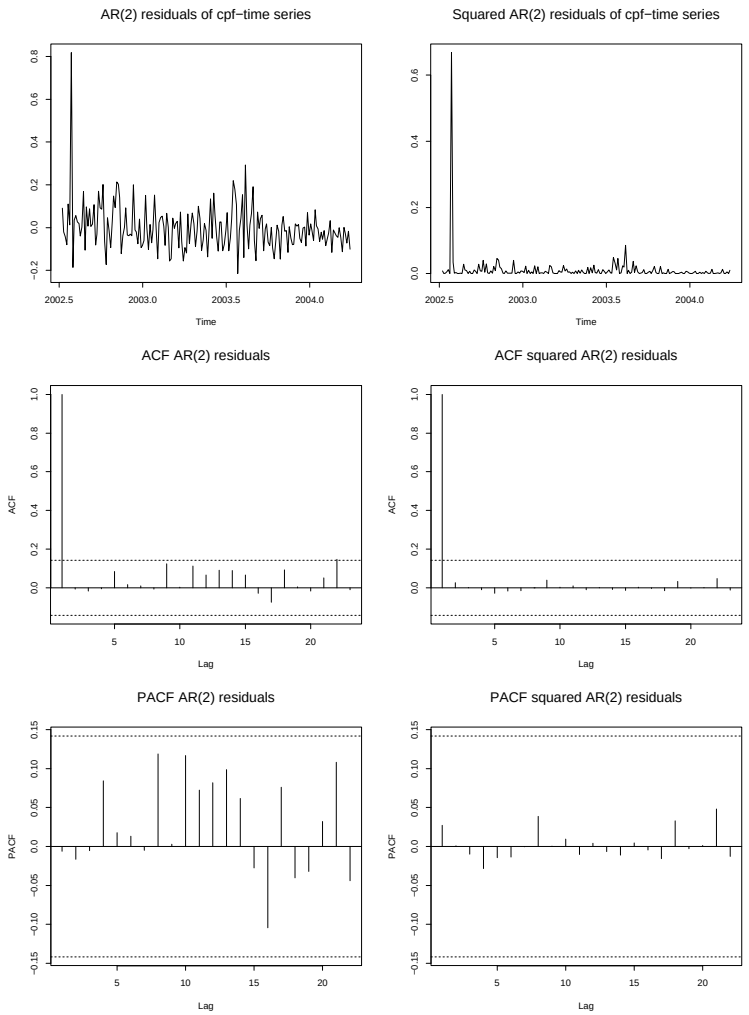

Figure 55: Six plots analysing the residuals and squared residuals of the initial $AR(2)$ fit of the cpf-time series.

that the shape of this curve resembles the shape of the original data, but it is not the time series itself. This is just an indication that the fit is not the best possible option. For reference we plot in the left-column of Figure 55 the autocorrelation functions of the residuals. As we can see, the autocorrelation is gone, since there are no longer values for lags significantly larger than zero. To get an idea of heteroscedasticity, we squared the residuals in plot 2. It seems that the variance is varying: at the beginning we see a range of about 0.7, which strongly decreases further in time. Still, the autocorrelation plots in the right-hand column do not indicate serial correlation. And a formal check testing the order of autocorrelation leads for both the residuals, and the squared residuals to a zero order. To be sure, we carry out a formal test to check for the presence of autoregressive conditional heteroscedastic (*ARCH*) effects [27]. We can use a Lagrange Multiplier test (LM-test) for that [27]. Carrying out such a test gives us a

test statistic of 40.0973, and a corresponding p-value of 0.0105. According to Table 2, there is moderate evidence against the null hypothesis that there are no *ARCH*-effects. If there is true time varying conditional heteroscedasticity, we have to fit a third model to the cpf-time series: one of the family of *GARCH* models. To that end, we also investigate whether the residuals of our robust fit display any evidence of *ARCH*-effects. For more information on the subject of *GARCH* modelling we refer the interested reader to [27, 5, 63, 40, 6].

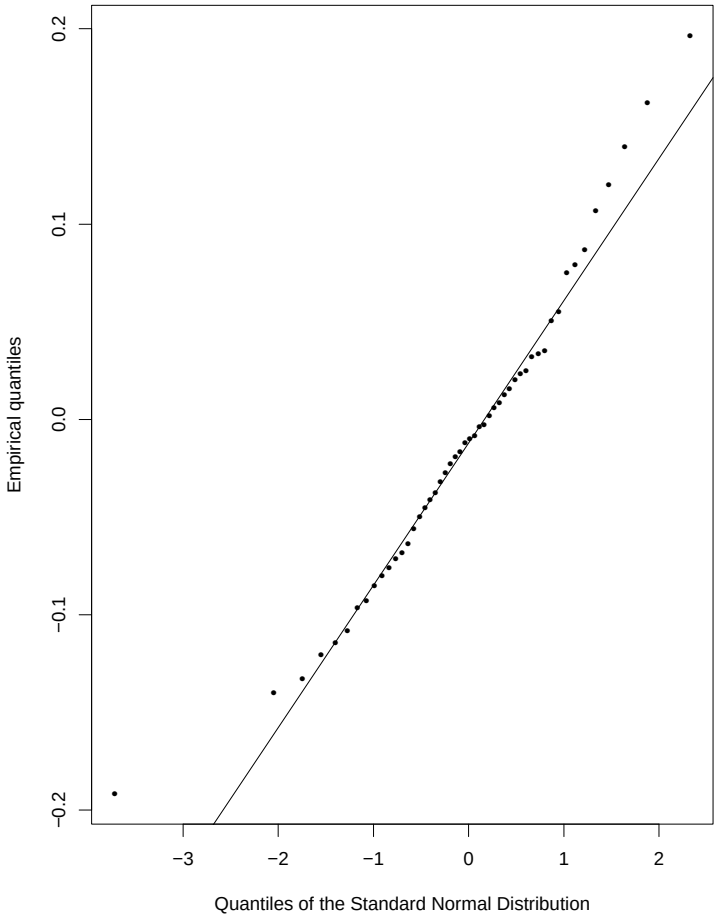

Figure 56: Q-Q plot of the residuals of the robust $AR(1)$ fit of the cpf-time series.

In Figure 56, we first check for the normality of the robust residuals of the outlier-aware $AR(1)$ fit for the cpf-time series. Recall that if the residuals are normally distributed, they carry no structure anymore. We can readily spot that also for these robust residuals there is evidence for heavy-tailed distributions, so the residuals are probably not normally distributed. Remains to see whether we can find *GARCH* structure in this time series.

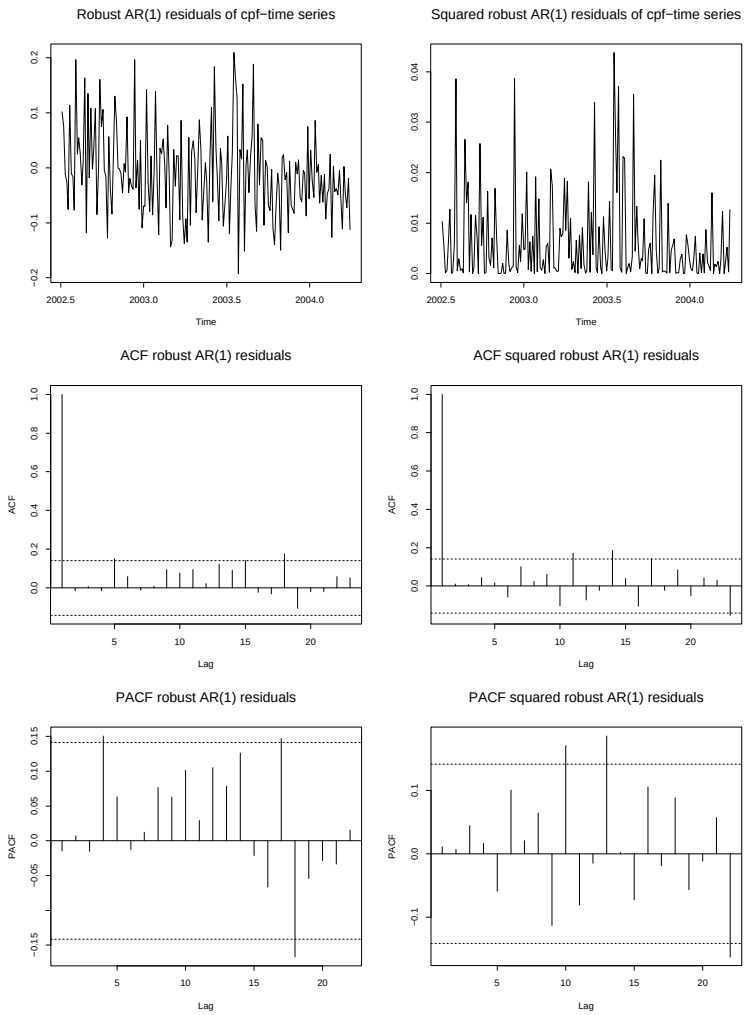

Figure 57: Six plots analysing the residuals and squared residuals of the robust $AR(1)$ fit of the cpf-time series.

So, we investigate the residuals and their squares in Figure 57, which has the same set-up as Figure 55. Plot 1 is much more different from the original cpf-time series, and its range is smaller than the original data. This is due to a better fit since the 6 outliers are regressed out. Indeed, the robust residuals have no serial correlation, as can be seen from the autocorrelation plots in the left-column. Then in the right-hand column, we see in plot 2 a wild curve, looking very volatile, but when we inspect the bandwidth, this is due to the scale. The range of the squared residuals for the robust fit is maximally 0.04, and there is not much varying variance left given the range of the original cpf-time series. Namely, the right-hand side autocorrelation plots do not reveal any serial correlation between the squared residuals, and a check on the autocorrelation order also gives zero. Finally, we carry out an *ARCH*-effect test on the robust residuals, and this provides us a different view than with the nonrobust residuals. The LM-test

statistic is 30.6138, and the corresponding p-value is 0.1043. Using Table 2, we find that there is no evidence to reject the null hypothesis that there are no *ARCH*-effects.

8.1 Windowing the residuals

In both cases, the p-values are near our chosen borders: 0.1043 and 0.0105. So it could be the case that for a small change in the data, the p-values are just under the chosen thresholds. And then we missed some *ARCH*-effects. To that end, we carried out a large number of *ARCH*-tests on so-called windows of the residuals. A window of a time series is just the same data but with different start and/or end times. Since the variance is large at the beginning, and really small at the end in the original data, we opted for two types of windows. One type keeps the starting point to the original, but the end point starts at data point 29, until the total time series. So, this gives us p-values of longer and longer time series where each increment a data point at the end is added to the window. The other type keeps the end point to the original, but the starting point gets earlier and earlier. We applied these windows on both the nonrobust residuals and the robust residuals of the cpf-time series.

In Figure 58 we plot the p-values of the nonrobust residuals. For the constant starting point windows in plot 1 we see that the p-values are all above the 0.05% significance level, only when the time series gets longer, the evidence for *ARCH*-effects becomes more apparent. As it turns out only adding the last few data points cause the *ARCH*-effects to become significant. For the constant end point windows in plot 2 we see the same pattern, except that for windows around the 80–100 there is some evidence for *ARCH*-effects, and again, at the end.

In Figure 59, we plot for the windows of Figure 58, their LM-test statistics. As is expected, the values for the LM-test statistic are high when the p-values are low. So, indeed we found evidence that the nonrobust residuals display time varying heteroscedasticity, but when we look at the various windows, this effect is gone for a bit less data, or is present for a window of the last 80–100 data points.

We already found that for the robust residuals, things are different. The p-value was just above the threshold. It could be the case that the p-value drops for somewhat smaller windows, indicating that we missed the nonconstant variance effects. In Figure 60, we plot for both window types the p-values. Plot 1 clearly shows that all windows are well above the 0.05% significance level, indicating that although the found p-value for the entire time series was on the edge, this was approximately the lowest value. So we added to our evidence that there are no heteroscedastic effects. Plot 2 shows again some slight heteroscedasticity for the 80–110 window lengths, but this vanishes for longer and shorter windows. In Figure 61, we plot for the windows of Figure 60, their LM-test statistics. As is expected, the values for the LM-test statistic are high when the p-values are low. It might be the case that if we would add the innovational outliers to the model and work with 13 exceptions instead of 6 outliers, that this effect is also gone. But since the p-values for the longer starting and end point constant series are not showing any sign of nonconstant variance, we are now pretty sure that we did not miss any *ARCH*-effects in the cpf-time series residuals.

So, we can conclude that there is no time varying variance for the residuals of the robust fit for the cpf-time series. This is not a sign of a bad software process improvement program, or due to careless reporting practices, but intrinsic to developing a large IT-portfolio for a large organization. One should not mix the variance that is often found in IT-productivity indicators with the improved accuracy of collecting the data that display large dispersions initially. The improved accuracy can be used to

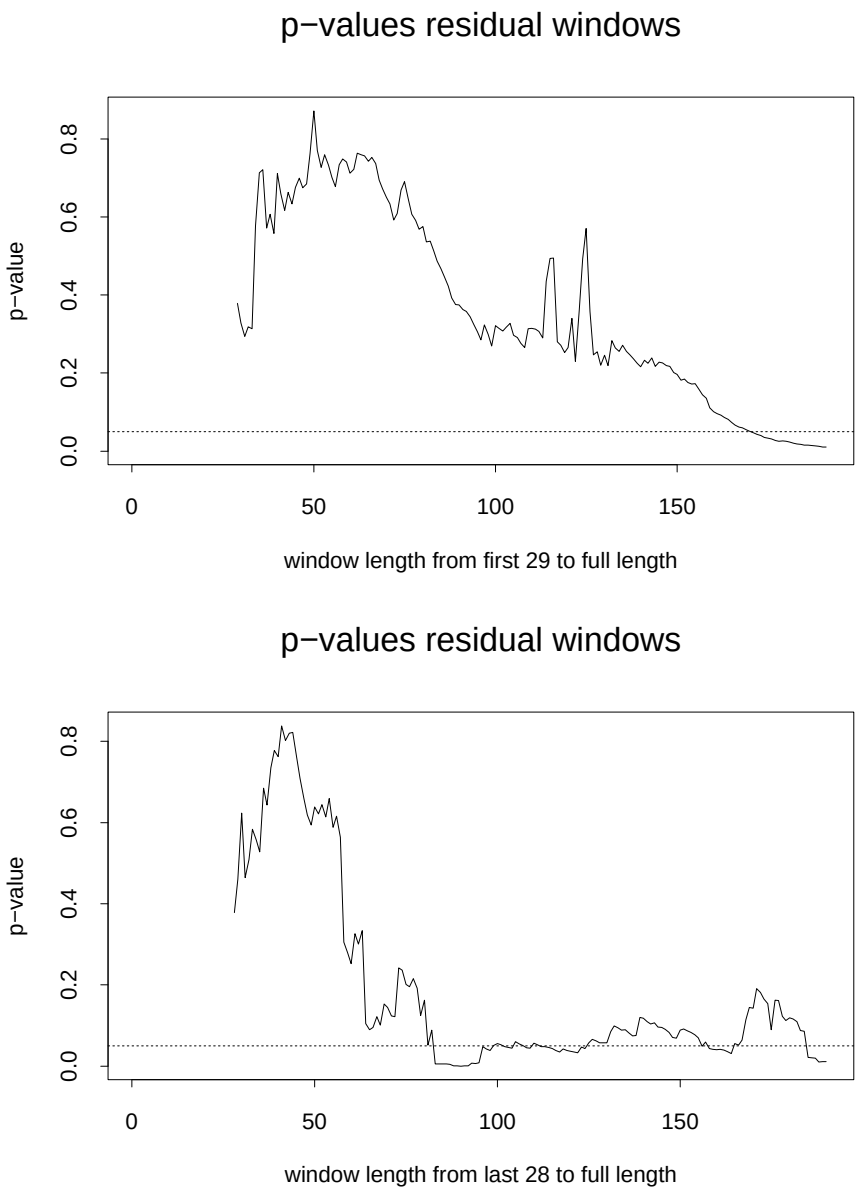

Figure 58: Several p-values of the Lagrange Multiplier test for windows of the residuals of the $AR(2)$ fit for the cpf-time series, to test whether nonconstant variance is present.

your advantage since models describing the history will then have more predictive power, but it will not automatically lead to smaller variance in the measured indicators themselves. Of course, when we assess time series that compare estimates with actual values over time, an effect of a successful software improvement program is that the variance decreases over time. So in those time series we often spot heteroscedasticity.

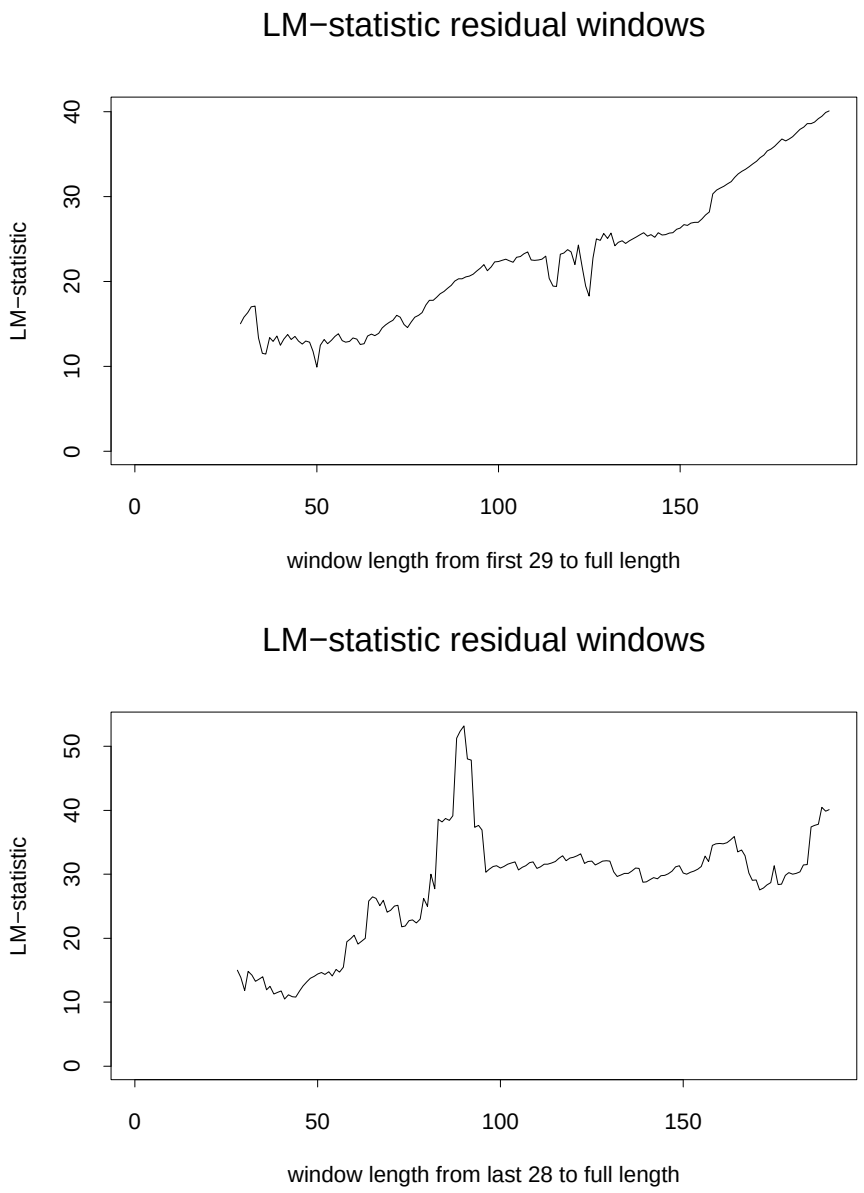

Figure 59: Several values of the Lagrange Multiplier test statistic for windows of the residuals of the $AR(2)$ fit for the cpf-time series, to test whether nonconstant variance is present.

But in the case we are discussing in this paper, we work with final data only, and we already learned that some estimates were retrofitted to the final data, namely in the case of cost estimations. But if the real estimates are kept, we can assess the quality of this part of the software process improvement program by calculating the so-called

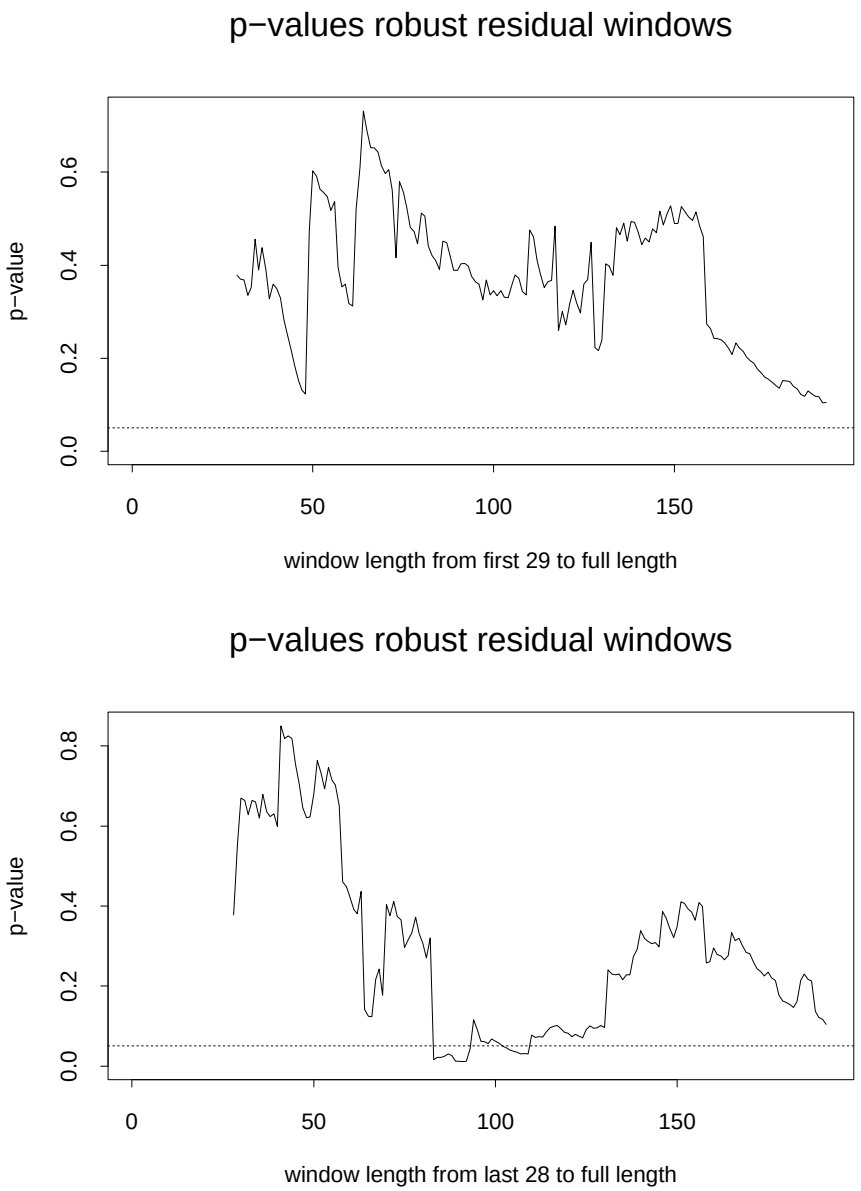

Figure 60: Several p-values of the Lagrange Multiplier test for windows of the residuals of the robust $AR(1)$ fit for the cpf-time series, to test whether nonconstant variance is present.

conditional standard deviation of the time series, and if this decreases, this is a sign of real improvements.

The reason that we are eager to find out whether the cost per function point is showing nonconstant variation is as follows. In Section 10 we will meet true heteroscedastic

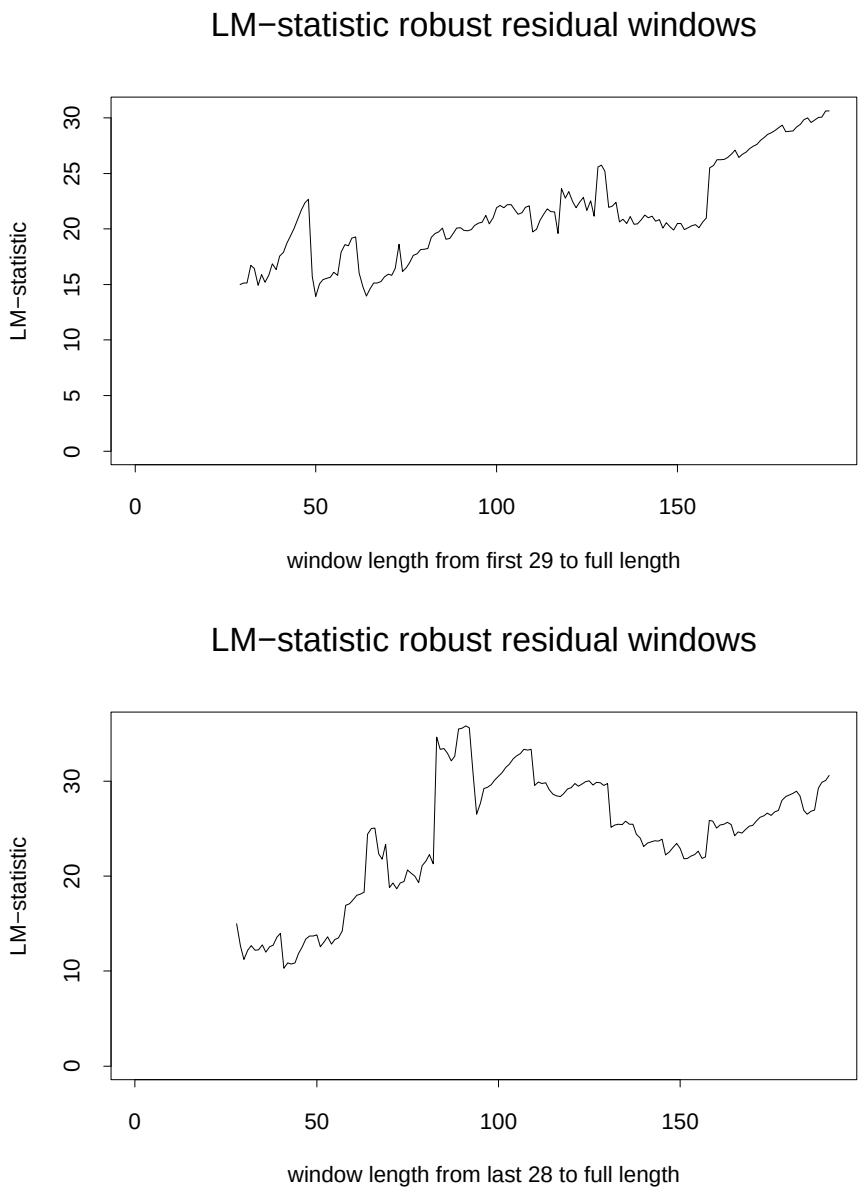

Figure 61: Several values of the Lagrange Multiplier test statistic for windows of the residuals of the robust $AR(1)$ fit for the cpf-time series, to test whether nonconstant variance is present.

data, namely the function points, durations and costs themselves show signs of heteroscedasticity. It is therefore interesting to know that the cost per function points does not show such behaviour. It may be the case that the varying variance is closely interrelated so that if you divide cost and function points this effect is strongly reduced. Such

co-movements could be a sign that the provided data is accurate over time, and that the SPI-program is resorting a positive effect. But before we discuss this, we propose an alternative to the annual moving average in the next section.

9 SMOOTHING AND SOOTHING

We have until now mainly concentrated ourselves on the micro-aspects of the data. From these elaborate analyses we concluded that the data is accurate and plausible. In this section we explore the macro-characteristics of the data. Such macro-aspects are necessary if only to sooth management: namely, corporate executives like numbers when there are only a few, with a preference for a single number. Fortunately, one of the merits of statistics is to summarize a lot of data with less—only think of the mean as the most well-known summary of lots of data. A visual approach to find the time-dependent "mean" is called smoothing, or smooth regression. Recall that the organization in this case study, also attempted to do this by taking a simple annual moving average of the data. We explained that their approach hides the effects that they wished to expose. We also illustrated that there is no linear trend in the data, so that simple summaries like a linear regression are not applicable.

The problem of summarizing high-frequency time series by trend lines is not new, and a number of fairly acceptable solutions have been proposed in the statistical literature. Sophisticated models such as a global polynomial regression for all the data could be used, but an alternative approach to analyse nonlinear observations is to fit a curve through the observed data using local means. That is, points of the fitted curve depend on observations in the neighborhood of the observations. Such curves estimate the macro-aspects of the data with less variability than the original data, so this can be seen as a summary of the data, and this summary is also called a smooth. Procedures to carry out smoothing operations are also called scatter plot smooths. For general information on smooth regression analysis we refer to [73].

There are various smoothing operations feasible that have a standard implementation in statistical programs. In this section we will apply 8 such smoothing operations to the cpf-time series and its cleaned version. The latter to illustrate the difference between working with and without outliers in the data. Note that for a final analysis there is no time to qualitatively assess the outliers so you want to see the smooths of both the data and the cleaned data. The ideal smoothing operation is robust, meaning that the outcome of the smoothing operation is not affected too heavily by outliers. We will illustrate the idea of smoothing with eight such operations (but others could be used as well). They are:

- Super smoothing. This scatter plot smooth is designed to be fast, hence the name. It uses cross validation to pick the so-called span: the size of the neighborhood around observations. It is less robust than the next smoothing operation, but it has a variable span for the neighborhood, which is not present in the one below. See [31] for more information.

- Lowess smoothing. This is a robust locally weighted scatter plot smooth (hence the name lowess, also known as loess). Lowess uses locally linear fits. Points in the neighborhood are weighted so that nearby points get the most weight. Lowess has a constant span for the neighborhood, but super smooth does not. So they each have their strong points. See [18, 12] for more information.

- Spline smoothing. This is a mixture of parametric and nonparametric smoothing. A spline [4, 41] is a local polynomial (a cubic one in our case) for points in a neighborhood, and these polynomials are stringed together, under certain conditions assuring smoothness. See [44] for more information.

- Box kernel smoothing. A kernel smooth is a local average smooth that calculates weighted averages based on a so-called kernel function. We used four kernel functions, of which the box function is the crudest. Kernel functions are symmetric probability density functions, the box being the uniform probability density function (looking like a box, hence the name). Important is the choice of the bandwith, i.e., the size of the neighborhood. In our example we used as bandwith 0.4. The interpretation of the bandwith is as follows: the larger the bandwidth the smoother the curve. We opted for a bandwidth that gives a meandrous curve for the box kernel. See [44, 90] for more information. Below we discuss three other kernel smooths.

- Triangle kernel smoothing. This is a kernel smooth where the symmetric probability density function is a triangle. This curve gives almost identical results as the other two below. This one has often a fast implementation so for large amounts of data it will give results the quickest.

- Parzen kernel smoothing. Here the density probability function is a combination of a box and a triangle, in such a manner that some sort of cosine wave ensues, this function is called a Parzen function.

- Normal kernel smoothing. The kernel function is the Normal probability density function.

- Robust median smoothing. This algorithm robustly smooths a time series by taking running medians. See [82, 61, 83] for more information.

It is not our intention to discuss smoothing in detail, we just want to give the reader a flavor of the main idea. For more information on the various smooths, their properties, their pitfalls, and their use we refer the interested reader to the above given references.

In Figures 62–69 we depict the above smooths. Each of these figures contains four plots, and we start to discuss these plots for Figure 62. Plot 1 consists of the original cpf-time series, the target line, and the robust median smooth. In plot 2 we depicted the cleaned cpf-time series, and a robust median smooth, which stays wiggly, too. These wiggly lines are still varying a little too much for summary purposes, so the robust median smooth will be discarded for the research set. Plot 3 contains the residuals of the smooth minus the original cpf-time series. As can be seen, the residuals have large dispersion, but do not show a descending trend anymore. Plot 4 contains the residuals of the smooth minus the cleaned cpf-time series. In our opinion, a smooth is adequate if the residuals do not show too much structure anymore; think of autocorrelation and heteroscedasticity. We tested both residuals for their tentative autoregression order by solving the Yule-Walker equations [92, 89, 8] for the residuals. These provide us with a tentative idea of autoregression. The robust median smooth scores worst in that sense: tentatively the residuals have an autoregression order of 1, and the cleaned residuals of 3. Also tentatively we checked for heteroscedasticity of both residuals, and the tentative autoregression order for the squared residuals is 0, and for the cleaned squared residuals it is 8. Apart from tentative checks, we also carried out more formal tests: an autocorrelation test for the residuals using the Ljung-Box modified Q-statistic, and

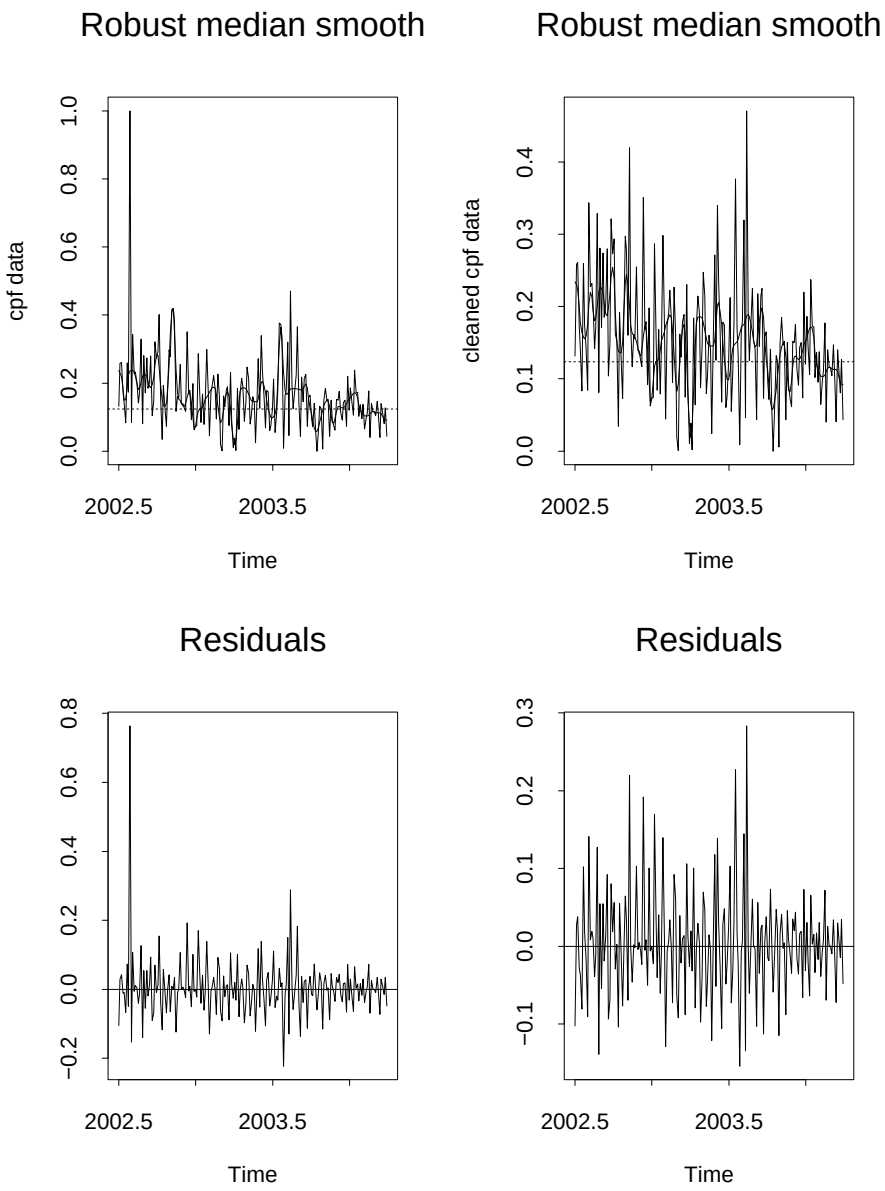

Figure 62: Robust median smooths of the cpf-time series, its cleaned version, and their residuals.

an *ARCH*-effect test using the Lagrange Multiplier test. These tests show that indeed, there is no evidence to reject the null hypothesis that the residuals of the robust median still contain autocorrelation, and that the residuals are heteroscedastic. For the cleaned versions there is no autocorrelation, and no *ARCH*-effects—apparently the autoregression of the cleaned squared residuals of 8 is spurious. Indeed, when we calculated a correlogram (containing the autocovariance function), we noted a significant lag at 8,

but the earlier lags were not statistically different from zero (not depicted). All in all, this smooth is not the best choice for summarizing the cpf-time series, cleaned or not. But what we can see, is that in both cases the wiggly line will trend under the target line.

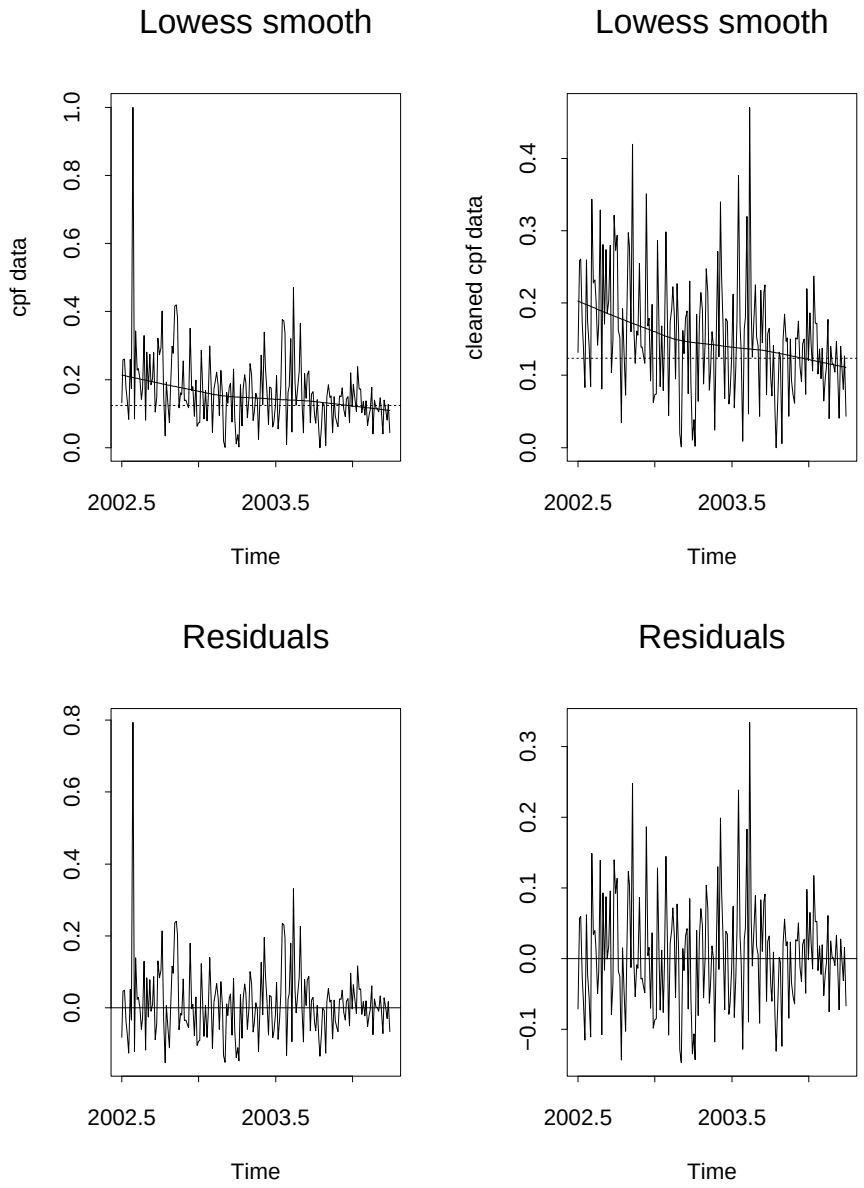

Figure 63: Lowess smooths of the cpf-time series, its cleaned version, and their residuals.

In Figure 63, we carried out a lowess smooth on both the original cpf-data and the cleaned version. The lowess smooth gives a much less wiggly line, and seems to show

several linear trends concatenated. As can be seen both smooths are fairly identical, so the impact on the outliers seems small. Of course, the residuals in both versions are large, and they do not show any sign of autoregression, or heteroscedasticity based on the tentative tests, except for the isolated order 8 autoregression of the squared residuals of the cleaned data. With formals tests there is indeed no evidence for autocorrelation on both residuals, but there is evidence for *ARCH*-effects in the residuals of plot 3, and not in the cleaned version.

Figure 64, contains a mixture of parametric and nonparametric smoothing: locally cubic splines are fitted through the scattered data, and these are connected in such a manner that a smooth curve ensues. The spline smooth has the same tentative characteristics for the residuals as the lowess smooth: no autocorrelation for the residuals, and only a tentative eight-order autoregression for the cleaned squared residuals. Formal tests confirm this, except that there is some evidence for autocorrelation for the cleaned residuals. Note that the spline smooths shows three local maxima, which is a bit like the inadequate smooths that we depicted earlier (cf. Figure33) to illustrate the negative side-effects of the smoothing operations initially chosen by the organization.

In Figure 65, we depicted the Triangle kernel smooth of the cpf-data and its cleaned version. Notice that there are almost no differences between this nonparametric smooth and the semi-parametric one that we calculated with the local splines. Not surprisingly, the tentative autoregressions in the residuals, and the squared residuals are of order zero, except the order 8 squared cleaned residuals. Indeed, no evidence for autoregression, but a bit for the cleaned version, and signs of *ARCH*-effects in the residuals, but not in the cleaned version.

Figure 66 contains the Parzen kernel smooth. As said, this smooth should not differ too much from the Triangle kernel smooth, and indeed basically all diagnostic tests on the residuals give almost the same answers. Also the smooths look almost the same.

Figure 67 is made with the Normal kernel smooth. Also this results in a similar view as the triangle kernel smooth. Also most tentative and formal tests lead to the same conclusion.

In Figure 68 we see a trend line like the Triangle kernel smooth, but much more wiggly. This is what is meant by the crudest kernel smoothing operation. An interesting point is that the spurious tentative order 8 autocorrelation for the cleaned squared residuals is gone. A Ljung-Box autocorrelation test gives a p-value of 0.6109, indicating the absence of autocorrelation. For the *ARCH* effects we see a p-value of 0.0066, meaning that there is evidence for heteroscedasticity in the residuals. For the cleaned version we notice that the residuals show some evidence for autocorrelation with a p-value of 0.0577, and the *ARCH*-effects are not obvious with the Lagrange multiplier test: we found a p-value of 0.4806. In our opinion, the Box kernel smooth shows the least residual structure until now, based on the tentative and formal tests, and is suited at least on the research set as a measure whether or not the software productivity targets are met.

Finally, Figure 69 contains the super smooth, which is not as smooth as the lowess smooth. What is interesting about this smooth is that as with the box kernel smooth, the residuals have the least structure: all tentative tests for autocorrelation result in a zero order autoregression. The Ljung-Box modified Q-statistic provides a p-value of 0.6947 indicating that there is no evidence for autocorrelation. The LM-test for *ARCH*-effects gives us evidence for heteroscedasticity: a p-value of 0.0059. The cleaned version results in residuals with no evidence for autocorrelation as well: a p-value of 0.1109. There are also no *ARCH*-effects, with a p-value of 0.4928. In Table 9, we summarize all the numerical data for the various smooths. We tabulated them in increasing order

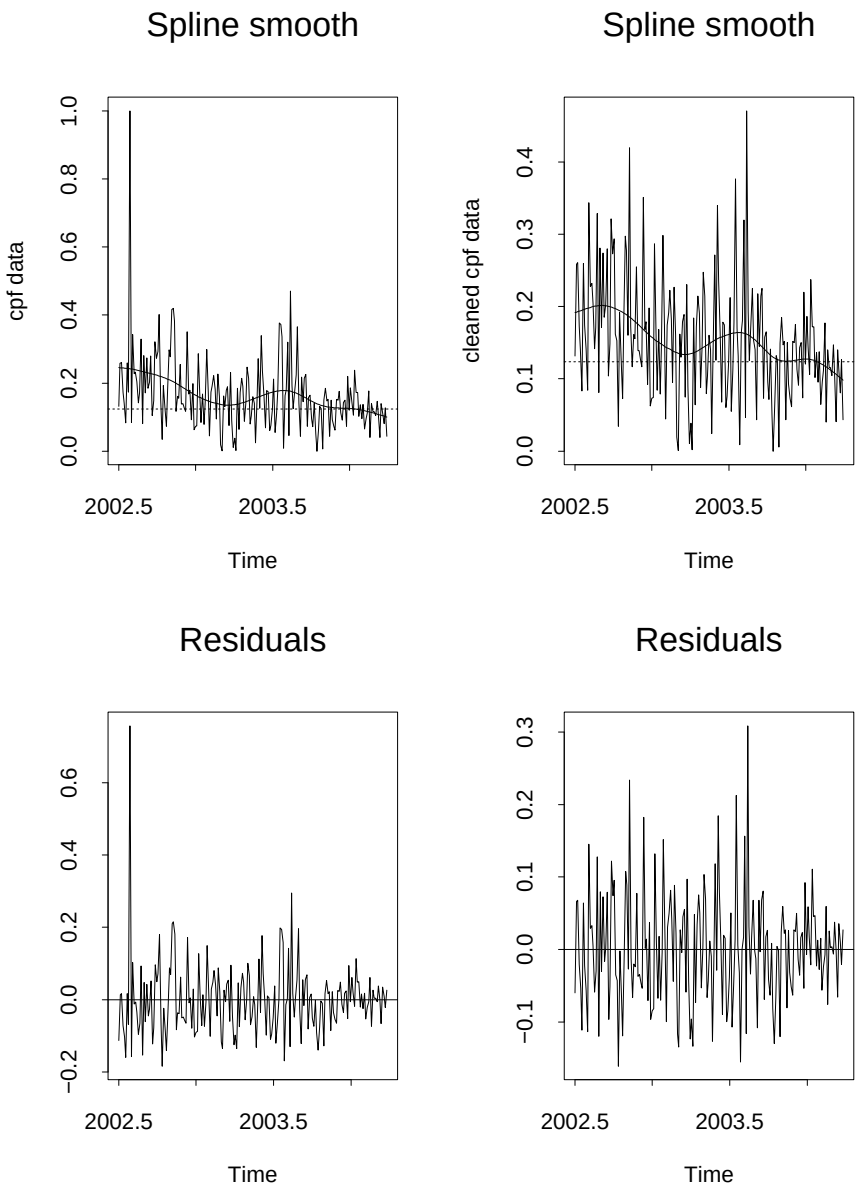

Figure 64: Spline smooths of the cpf-time series, its cleaned version, and their residuals.

of usefulness as a measure for software productivity improvement for the research set (note that for the audit set this can turn out to be a bit different). We recall that we investigated the *ARCH*-effects of the research set in great depth, so as to avoid missing such effects (see Section 8). The fact that we see such effects in the residuals are not a sign of probable heteroscedasticity, but the shape of the time series with its outliers gives rise to this outcome. Our extensive windowing of the time series and the many

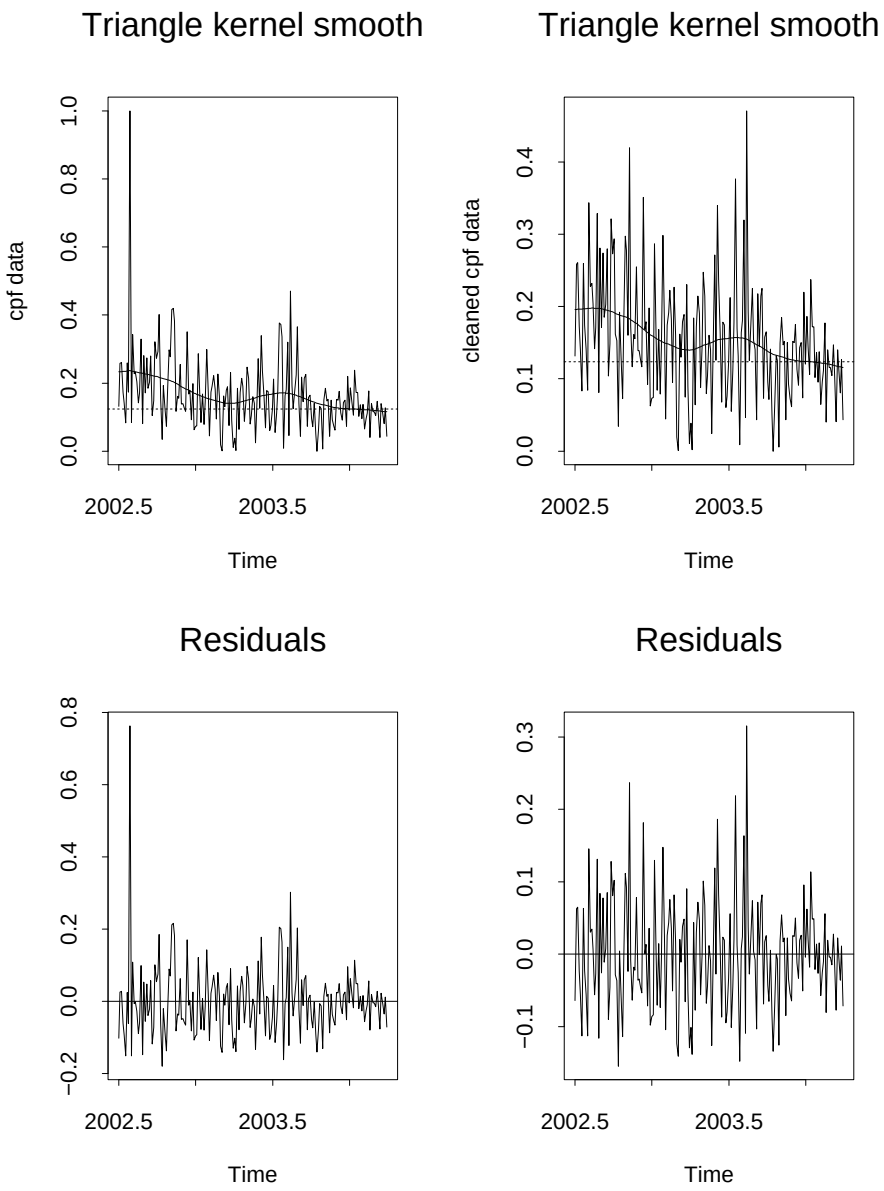

Figure 65: Triangle kernel smooths of the cpf-time series, its cleaned version, and their residuals.

checks for *ARCH*-effects showed clearly that the effects are not present. So, just as with the Box kernel smooth, this super smooth seems to capture almost all crucial trend information, leaving residuals with almost no structure, except for a heavy tail perhaps. So in the case of the research set we conclude that both the Box kernel smooth and the super smooth are suitable measures for deciding whether or not the targets for software productivity improvements are met or not.

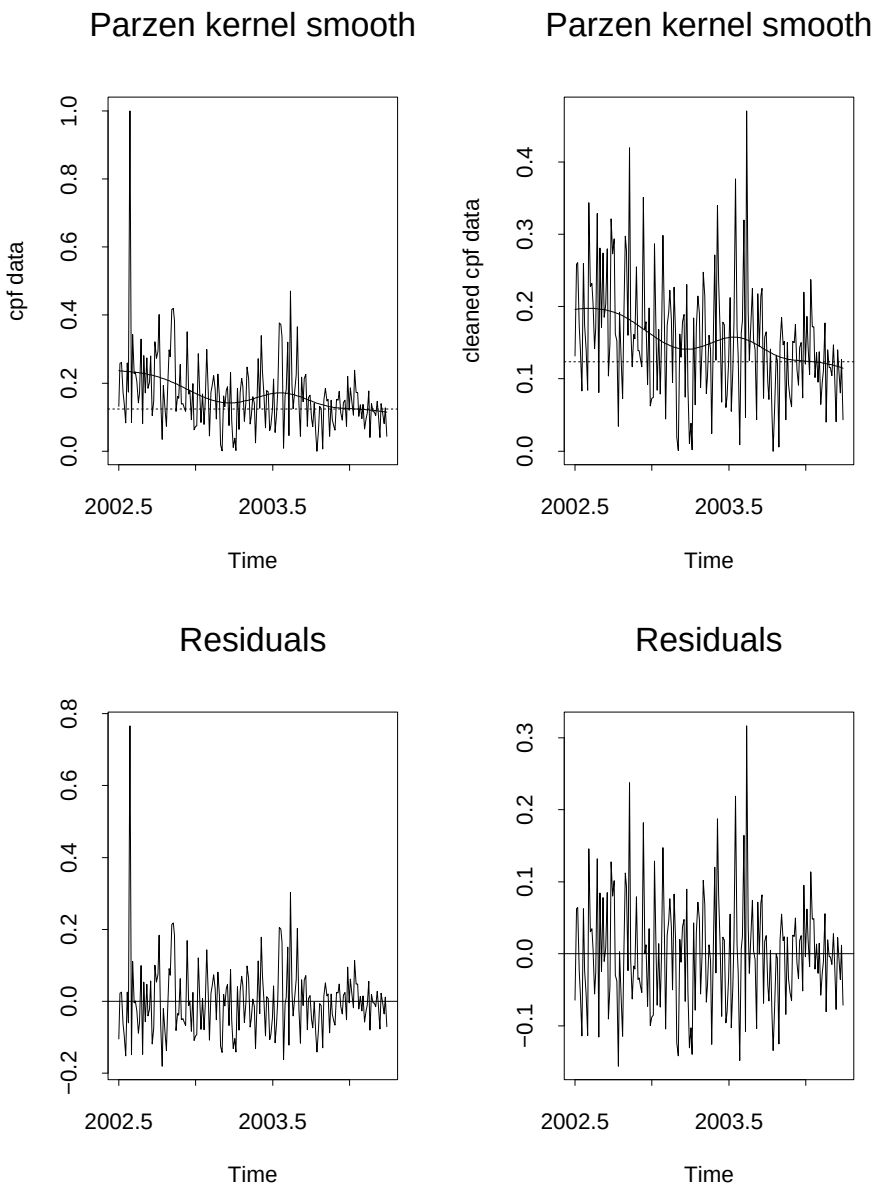

Figure 66: Parzen kernel smooths of the cpf-time series, its cleaned version, and their residuals.

Now that we have an idea of the residuals and the shape of the smoothing operations we compare them in one graph. In Figure 70, we depicted all smooths together. Plot 1 contains all smooths for the original cpf-data. We can see that the robust median smooth displays trend-reversal behaviour around the other smooths. Therefore, the wiggly smooth is not recommended (apart from the significant residual structure). It is insightful that the micro-patterns return in a damped way in the other smooths, who all

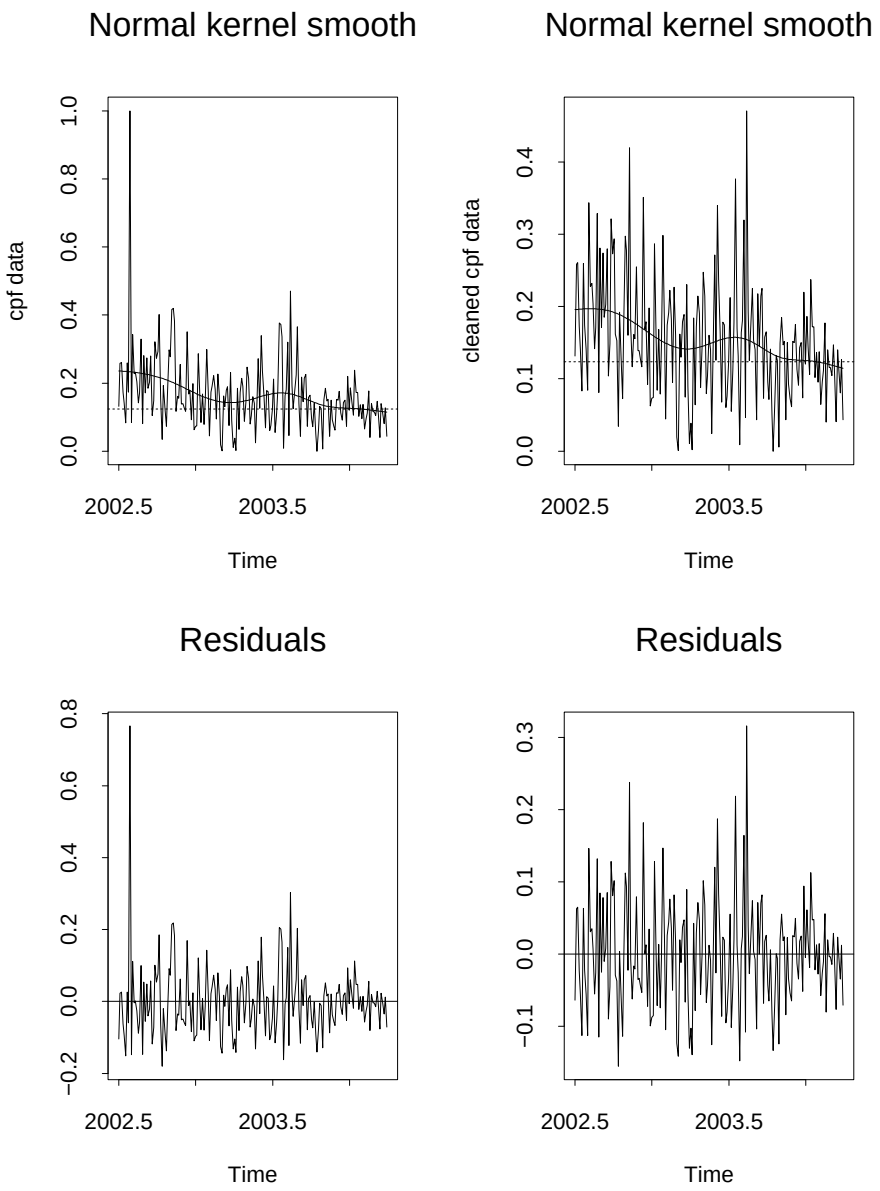

Figure 67: Normal kernel smooths of the cpf-time series, its cleaned version, and their residuals.

form a slowly descending wave. These trend waves are what we would expect from a successful software process improvement program, where consecutive improvements induce another wave of IT-productivity improvement. Only since there is so much variation, this is not visible in the raw data. Plot 2 contains the same eight smooths, but then for the cleaned cpf-time series. This to illustrate the effect that the outliers have on some of the trend lines. Note that in all cases the trends go under the target line when

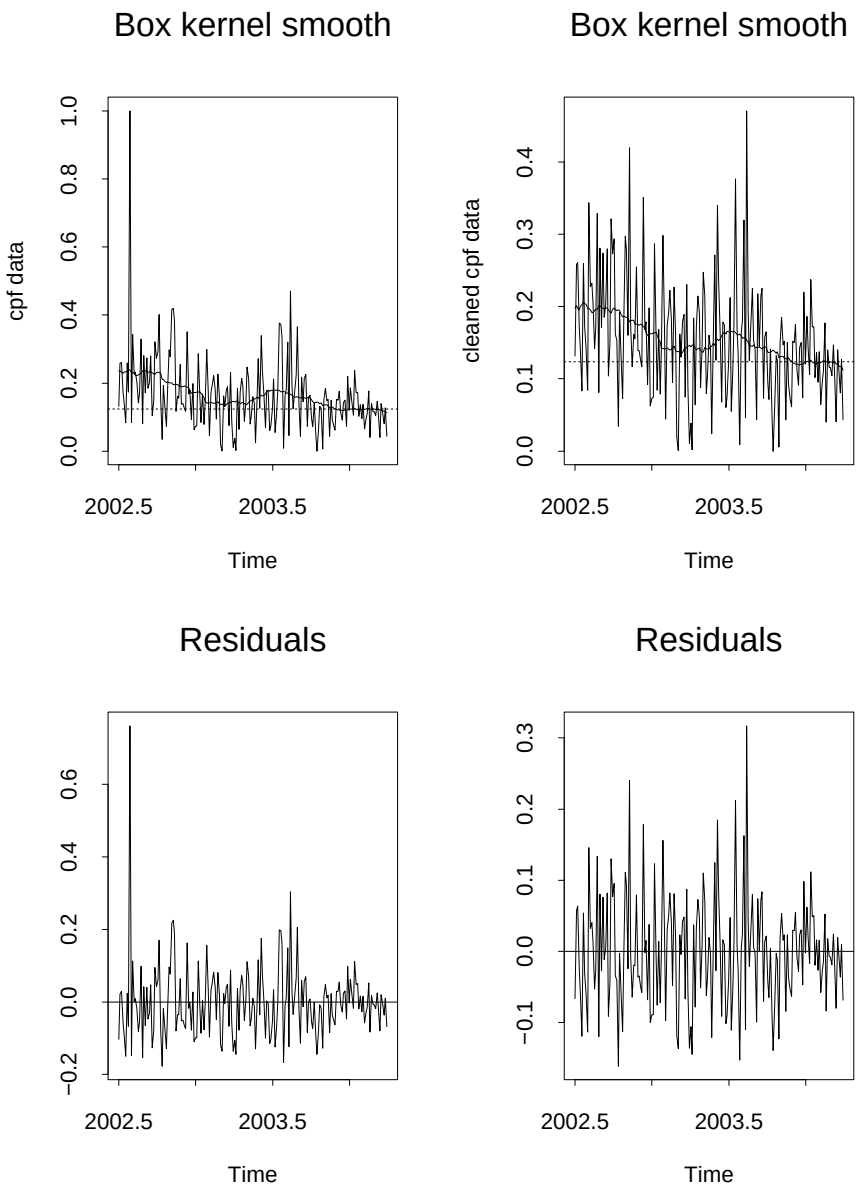

Figure 68: Box kernel smooths of the cpf-time series, its cleaned version, and their residuals.

time passes, but that the starting points differ significantly depending on the presence or absence of outliers in the time series. The target line is the dashed line in both plots.

Figure 71 is the same as Figure 70 except that we left out the wiggly robust median smooth in these plots. Now we can see a little better how close the other smooths are. For the original cpf-time series we observe in plot 1 that the starting values for the smooths do show some variation, and this is somewhat larger than the starting values in

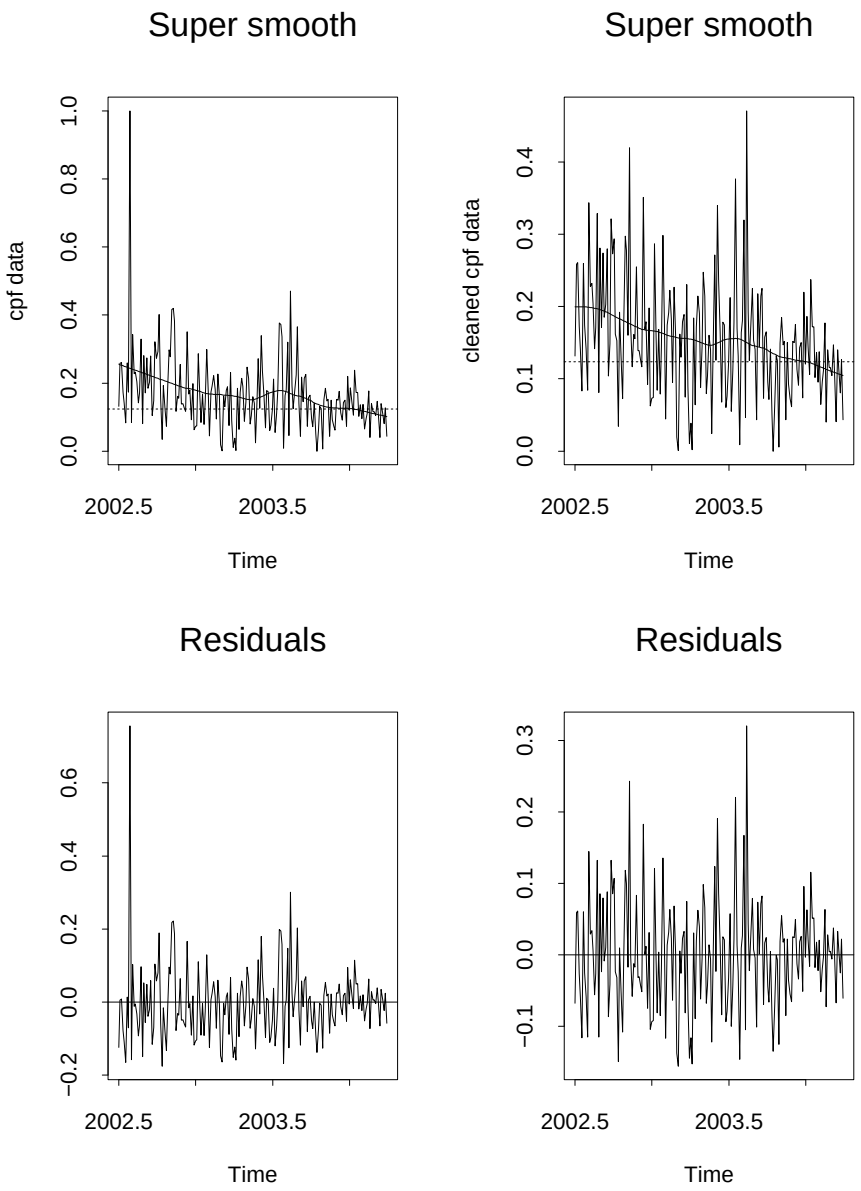

Figure 69: Super smooths of the cpf-time series, its cleaned version, and their residuals.

plot 2. The end values in both plots 1 and 2 are closer together, which is not a surprise, since further on in the time series there are less outliers. In Table 10, we provide numerical details on the smoothing operations. The top half of this table contains six-point summaries of all individual smooth values. We note that their minimum value is around 0.10 except for the robust median smooth, which shows a low (local) minimum. We can clearly see in the summaries of all smooths together that this robust median

smooth	res	res^2	cl. res	cl. res^2	LB	LM	LB cl.	LM cl.
Median	1	0	3	8	0.1166	0.0022	0.0002	0.2745
Lowess	0	0	0	8	0.7617	0.0026	0.1196	0.3665
Spline	0	0	0	8	0.5218	0.0058	0.0365	0.4714
Triangle	0	0	0	8	0.6305	0.0051	0.0647	0.4724
Parzen	0	0	0	8	0.6274	0.0051	0.0665	0.4690
Normal	0	0	0	8	0.6301	0.0049	0.0668	0.4669
Box	0	0	0	0	0.6109	0.0066	0.0577	0.4806
Super	0	0	0	0	0.6947	0.0059	0.1109	0.4928

Table 9: Analysis of residuals and the squared residuals plus their cleaned versions after smoothing for autoregression and *ARCH*-effects.

smooth	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
6-point summaries of smoothened cpf-data						
Super	0.1018	0.1352	0.1650	0.1673	0.1853	0.2555
Lowess	0.1093	0.1328	0.1458	0.1521	0.1716	0.2136
Spline	0.1001	0.1351	0.1551	0.1642	0.1818	0.2453
Box	0.1121	0.1365	0.1570	0.1644	0.1896	0.2410
Triangle	0.1154	0.1388	0.1571	0.1644	0.1830	0.2368
Parzen	0.1148	0.1382	0.1567	0.1644	0.1839	0.2363
Normal	0.1145	0.1376	0.1565	0.1644	0.1840	0.2357
Median	0.0187	0.1171	0.1518	0.1592	0.1849	0.4184
All 8	0.0187	0.1332	0.1559	0.1625	0.1833	0.4184
First 7	0.1001	0.1353	0.1562	0.1630	0.1815	0.2555
6-point summaries of smoothened cleaned cpf-data						
Super	0.1046	0.1330	0.1543	0.1537	0.1684	0.1996
Lowess	0.1108	0.1305	0.1425	0.1481	0.1654	0.2027
Spline	0.0984	0.1279	0.1495	0.1529	0.1703	0.2015
Box	0.1121	0.1352	0.1449	0.1535	0.1739	0.2059
Triangle	0.1154	0.1326	0.1496	0.1533	0.1705	0.1977
Parzen	0.1148	0.1324	0.1490	0.1533	0.1703	0.1975
Normal	0.1145	0.1320	0.1493	0.1532	0.1702	0.1970
Median	0.0187	0.1172	0.1509	0.1492	0.1776	0.2543
All 8	0.0187	0.1303	0.1486	0.1521	0.1717	0.2543
First 7	0.0984	0.1315	0.1483	0.1526	0.1691	0.2059

Table 10: Summaries of the values of the smoothing operations, all combinations of the smooths, and the first seven, excluding the wiggly robust median smooth.

smooth deviates a lot from the others, and the reason is that it's not that smooth. If we throw out this one, as in Figure 71, we see that the minimum is indeed 0.10. The same holds for the maximum values. The bottom half of Table 10 is the same as the top half only then we summaried the smooths of the cleaned cpf-time series. Indeed if we take the summary of the first 7 smooths in the last row, we see fairly similar minimum values with the smooths of the original data, and a difference of about 0.05 with the starting values, that are also approximately the maximum values. So compared to the target

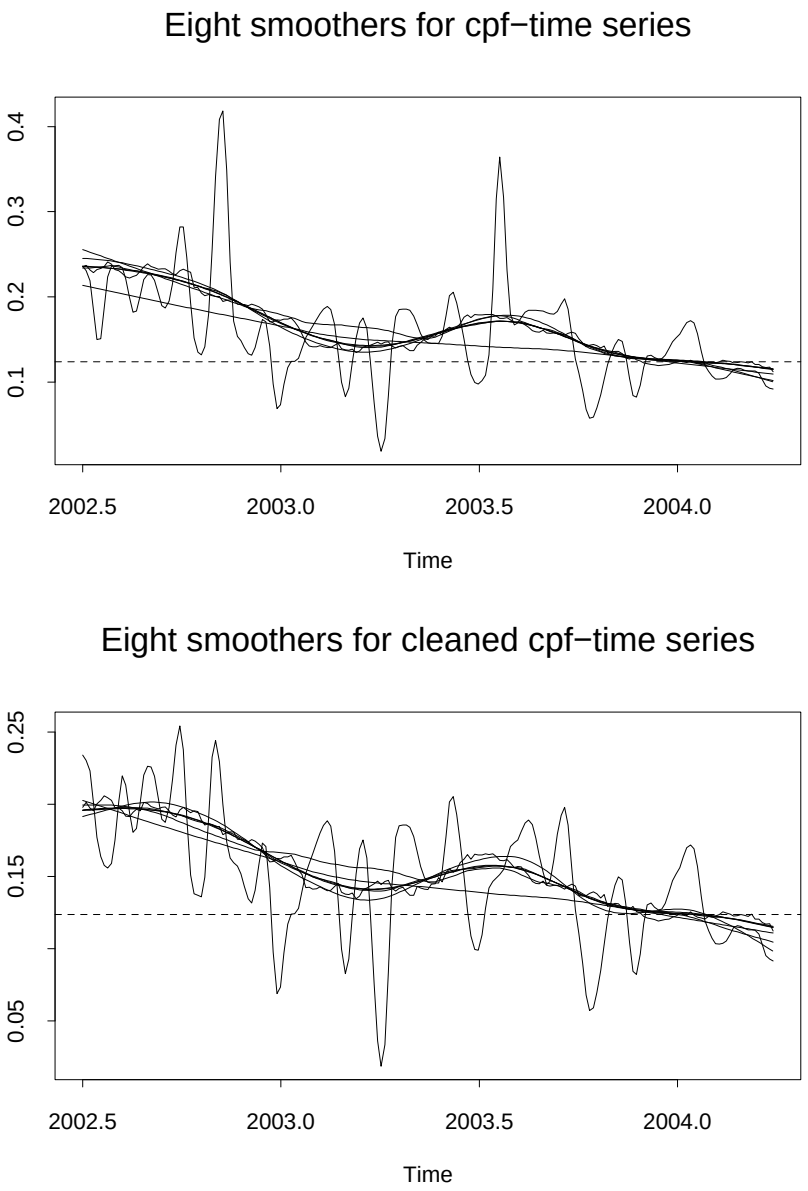

Figure 70: Eight different smoothing operations on the cpf-time series, and the cleaned cpf-time series.

line, we can now safely state that for the data in the research set the software process improvement program would have met its target. Note that our ex ante forecast predicts a rise in the cost index per function points: see Figure 42. So we have to be careful not to jump to conclusions. Moreover we should not take the start and end point of the smooths, but maxima and minima, provided that the maximum is at the beginning and

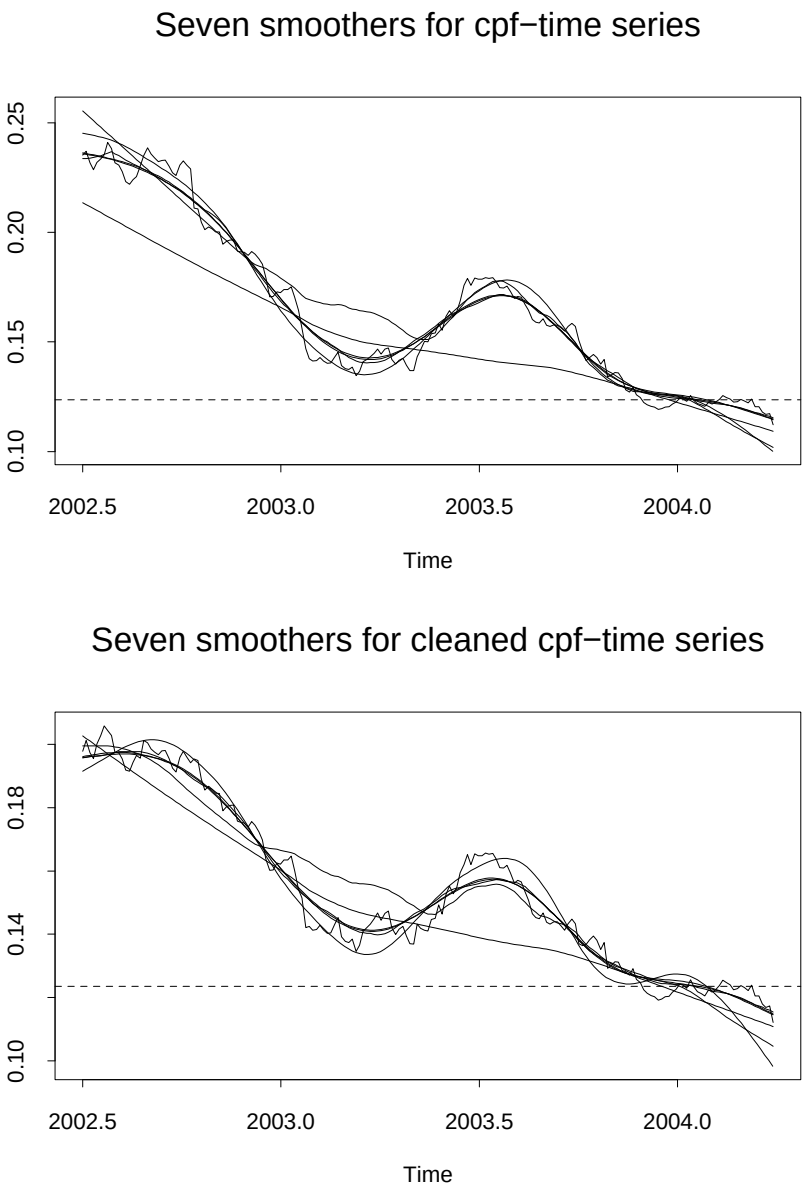

Figure 71: All but the wiggliest smooths on the cpf-time series, and the cleaned cpf-time series. Note that we excluded the robust median smooth.

the minimum is at the end (that is, there is a downward trend). Otherwise, the target is more dependent of the arbitrarily chosen start and end points, than of the overall trend indicated by the smoothing operations.

10 INTERRELATIONS AND CO-MOVEMENTS

In this section, we take time into account to investigate the major variables size, duration, and cost. Of course, we have seen in our heavy tail analysis that these indicators all show similar distributional properties. But we have not yet looked at their interrelations, such as time-dependencies, and their co-movements. One would expect from reliable data that IT-projects with more function points cost more money, and take more time. So, one would expect interrelations in the time series for size and their corresponding duration and cost index. Moreover, if function point data appears volatile at some point in time, you would expect co-movements of volatility for duration and/or cost as well. Again, such relations turn out not to be not deterministic, and there will be stochastic effects. Since our data is time-stamped, we can search for such interrelations and co-movements, which will add to our assessment of data reliability.

For a start, we plot the 3 time series for size in function points, duration in calendar months, and their cost index in Figure 72. We added a grid to the plots so that it becomes easier to spot interrelations. This is immediately clear, namely, function point spikes correspond with either peaked durations, high costs, or both. For instance, around the fourth quarter of 2002, we observe in plot 1, a large-sized IT-project of about 1400 function points. At the same time we see in plot 2 a large duration, of about 30 months. And although we see a spike in plot 3 or the cost index, it is not as high as the spike in September 2003 of the same plot 3. If we move up one plot to plot 2 again we see that the two spikes in plot 3 around September are also seen in the duration plot 2, and one of them is also seen in the size plot 1. When looking at the 3 time series, and their wiggles, it seems that they have something in common. This is what we expect from reliable data on IT-projects, since size, duration and cost should be somehow related, despite the large variances found in the literature. The 3 time series of Figure 72 should be seen as a system describing major indicators of IT-investments over time. We call such a system a multivariate time series. For the sake of ease, we will refer to this 3-variate times series as the fdc-time series, where fdc is short for the components of the system: the function points, durations, and cost index. Our purpose is to understand whether the data is a reliable source, also when the time dimension is taken into account. Of course, if the interrelations are too smooth to be true, there is most probably something wrong with the reporting system, or some other cause that invites to retrofit data.

When an IT-investment strategy is in place, where a series of IT-investments are planned and executed, we may determine various forms of correlation. For instance, deterministic or stochastic trends in building larger and larger systems, or the other way around. But also budget cuts, investment impulses, regulatory changes, and other aspects (snowstorms, etc) can create time dependencies. These dependencies can be quite intricate. Namely, while assessing the function point sizes in isolation, there is not much evidence for serial autocorrelation. But the trivariate time series shows evidence for serial correlation. We plot the autocorrelation functions and cross-correlations for the trivariate fdc-series in Figure 73. In this figure we use the following abbreviations: fps stands for function points, dur is duration, and cin means cost index. Looking at plot 1, we see that the autocorrelation of the function points do not show significant autocorrelations, except at lags 13 and 15. Plot 5 contains a significant autocorrelation at lag 1, indicating that the duration itself is autocorrelated. Furthermore, plot 6 shows cross-correlations between duration and cost index. Some borderline cases in plots 7 and 8, and plot 9 indicate that the cost index is also autocorrelated. All in all, Figure 73 shows enough clues for a further investigation in the direction of potential

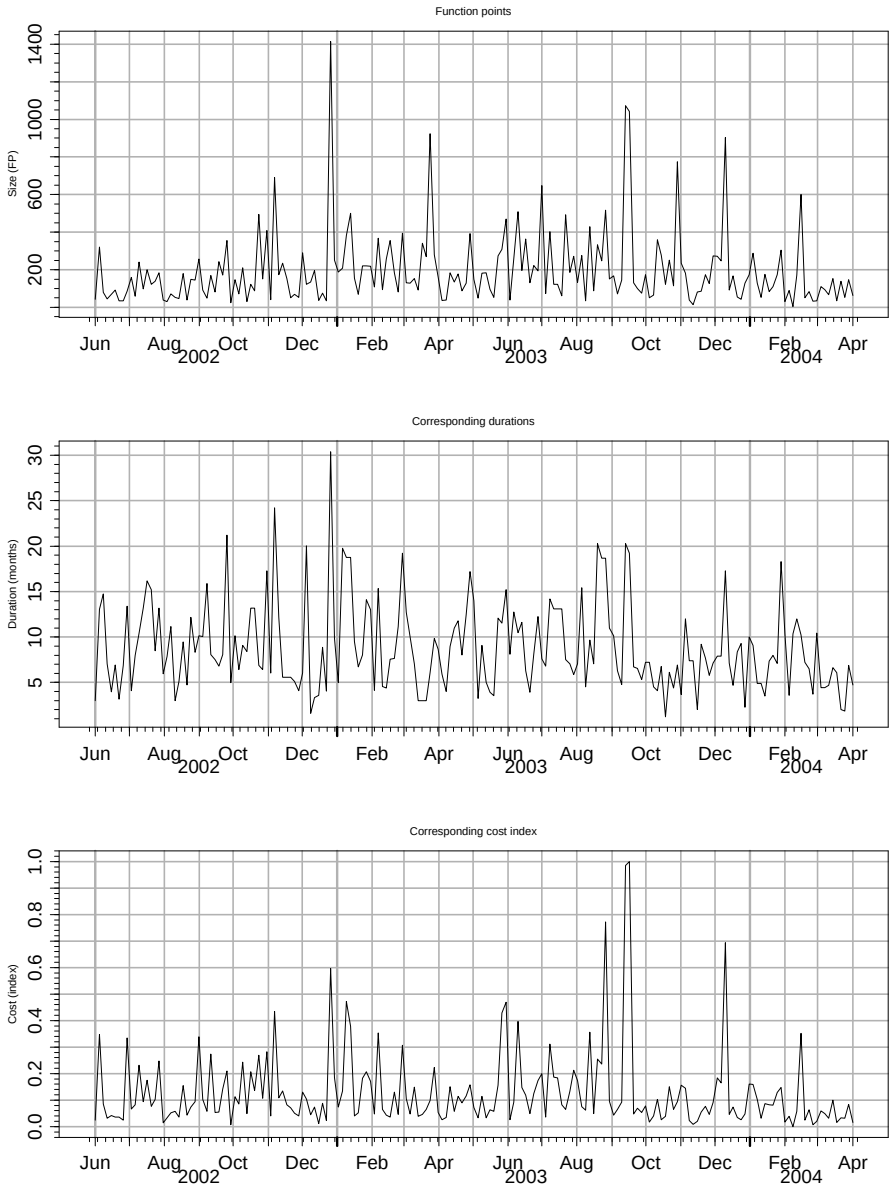

Figure 72: Three time series for the same IT-investments: their size, duration, cost.

vector autoregression present in the fdc-series. To that end, we conduct a simple vector autoregression (*VAR*). Using Akaike's Information Criterion, a parsimonious model for the fdc-time series is found. It is a vector autoregressive model of order 1, abbreviated a $VAR(1)$ model. Indeed, a formal autoregressive fit on the isolated components of the fdc-time series show order 0 for the function points, and order 1 for the other two. The fitted $VAR(1)$ model is as follows:

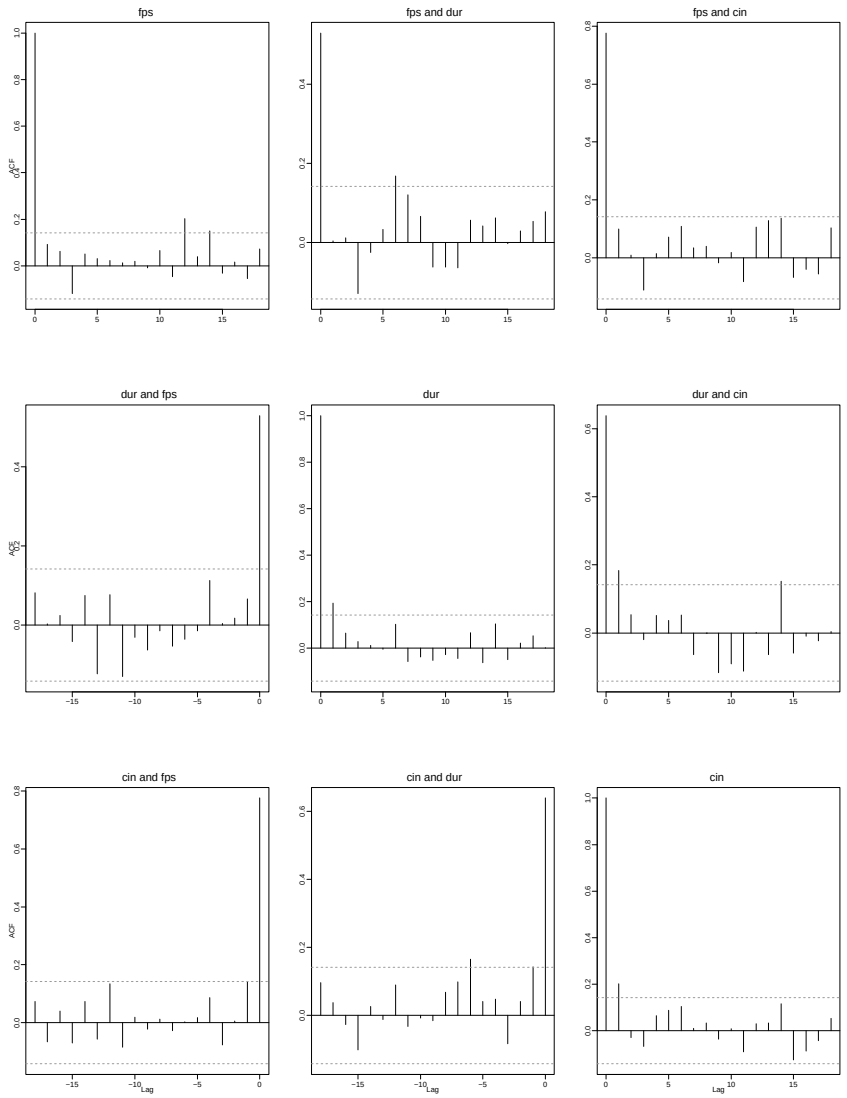

Figure 73: Autocorrelations and cross-correlations for the fdc time series.

$$
\begin{aligned}
f_t &= 202.9(31.1) - 0.0050(0.0028)d_{t-1} + e_{1t} \\
d_t &= 7.57(0.745) + 0.141(0.092)d_{t-1} + e_{2t} \\
c_t &= 0.10(0.02) + 8.20(4.03)d_{t-1} + 0.23(0.13)c_{t-1} + e_{3t}
\end{aligned}
$$

This $VAR(1)$ fit has a fairly large error covariance matrix (omitted here), so we

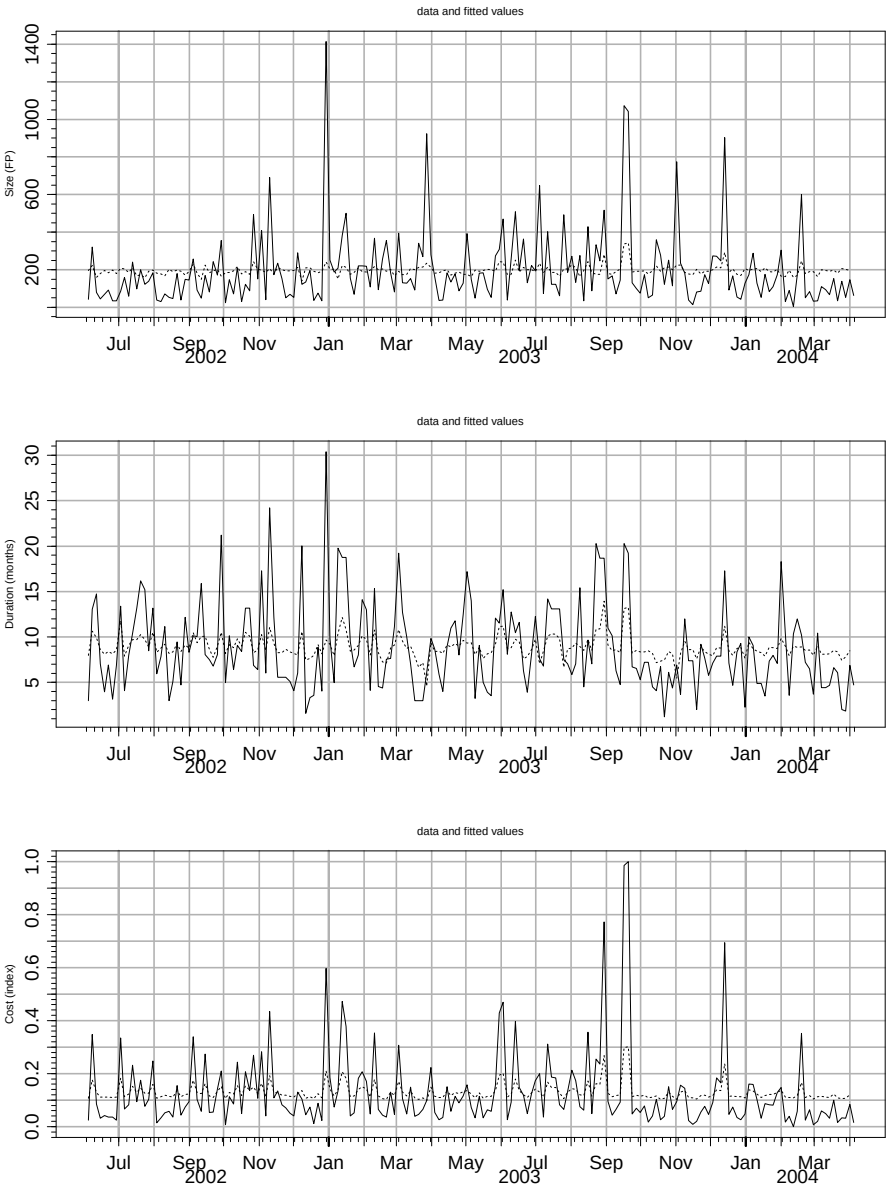

Figure 74: Both the fdc-time series and a first rough $VAR(1)$ model.

need a better model. But at this point we can already observe from the above equations, that the standard errors in brackets behind the fitted values are fairly large, and the values that are not present (e.g., the coefficient for f_{t-1} in the f_t formula) are not significantly different from zero. It seems that the function points are weakly negatively influenced by the first lag of the duration. The duration is weakly influenced by its own lag, and the cost index is weakly influenced by the lagged duration and the lagged cost

index. The residual noise is large, and still contains structure.

The formula does not give as much information about the quality of the fit as a picture would. Figure 74 contains both the original data (solid lines), and the fit (dotted lines). Indeed, the fit is not that good and we have seen better fits, e.g., for the cpf-time series. But as an indication that the fdc-series is clarified by a $VAR(1)$ model it will do. For a better fit, we carry out a Bayesian vector autoregression, where prior beliefs can be encoded via control variables. For our fit, we used the so-called default beliefs, and for more information on Bayesian vector autoregression we refer to [54, 75, 93, 70].

In Figure 75, we depict the Bayesian fit, which appears much better than the unrestricted classical vector autoregression of Figure 74. Again, the dotted lines depict the Bayesian $VAR(1)$ model, and the solid lines are the fdc-time series. In comparison with Figure 74, we can readily see that the Bayesian model follows the original data much closer than the initial model. Of course, the fit is also far from perfect, but it is much better than the first attempt. In a formula, this model looks as follows:

$$\begin{aligned} f_t &= 17.0(10.1) + 0.62(0.08)f_{t-1} + 4.64(2.37)d_{t-1} - 212.8(107)c_{t-1} + e_{1t} \\ d_t &= 0.51(0.25) + 0.84(0.06)d_{t-1} + e_{2t} \\ c_t &= 0.011(0.0072) + 0.0030(0.0017)d_{t-1} + 0.64(0.08)c_{t-1} + e_{3t} \end{aligned}$$

The residuals behave better than for the first attempt: their standard errors are 223.986 for the function point residuals, 5.659 for the duration residuals, and 0.161 for the cost index residuals. Also the regression is much better. The regression diagnostics such as the well-known R^2 regression coefficients are 0.305, 0.584, and 0.354 for the function points, durations and cost index. The adjusted R^2 regression coefficients are 0.294, 0.577, and 0.344. So the vector error term has a covariance matrix with much smaller values than the first attempt but still much residual noise is present. All in all, this fit seems not great, so we need additional diagnostics. Thus far, the Bayesian fit shows now significant autocorrelation in the first lag for all components. For supposedly strongly related data, autocorrelation should be significant in all components, if it is present in one of the components. And this is the case. Furthermore, there are weak cross-lag correlations, with fairly large standard errors. The lag-terms that are missing returned coefficients that were not significantly different from zero.

Let us assess the Bayesian $VAR(1)$ fit. We want to know whether this model is stable, that is, it does not contain autocorrelations of the form $y_t = y_{t-1} + u_t$, in other words, that there are lag coefficients with an absolute value of 1. To that end we have to calculate the Eigen values of the coefficient matrix, spanned by the lag coefficients. This matrix is:

$$\Pi = \begin{pmatrix} 6.2010^{-01} & 4.64025 & 212.833 \\ -1.0410^{-03} & 0.84274 & -1.691 \\ -9.7610^{-05} & 0.00299 & 0.635 \end{pmatrix}.$$

Indeed, the absolute values of the Eigen vectors of the coefficient matrix Π are 0.813, 0.772, and 0.513. As these values are smaller than one, the $VAR(1)$ model is stable in the abovementioned sense. We have seen the somewhat low regression diagnostics and the fairly large standard errors for some of the coefficients, so our next diagnostic test is to assess whether the found coefficients are actually helping in explaining the fdc-time series. In fact, we want to test the null hypothesis that the coefficient matrix Π equals the null vector. We do this via the so-called Wald statistic. To that end, we consider the coefficient matrix as a vector. This is not Π, but a matrix

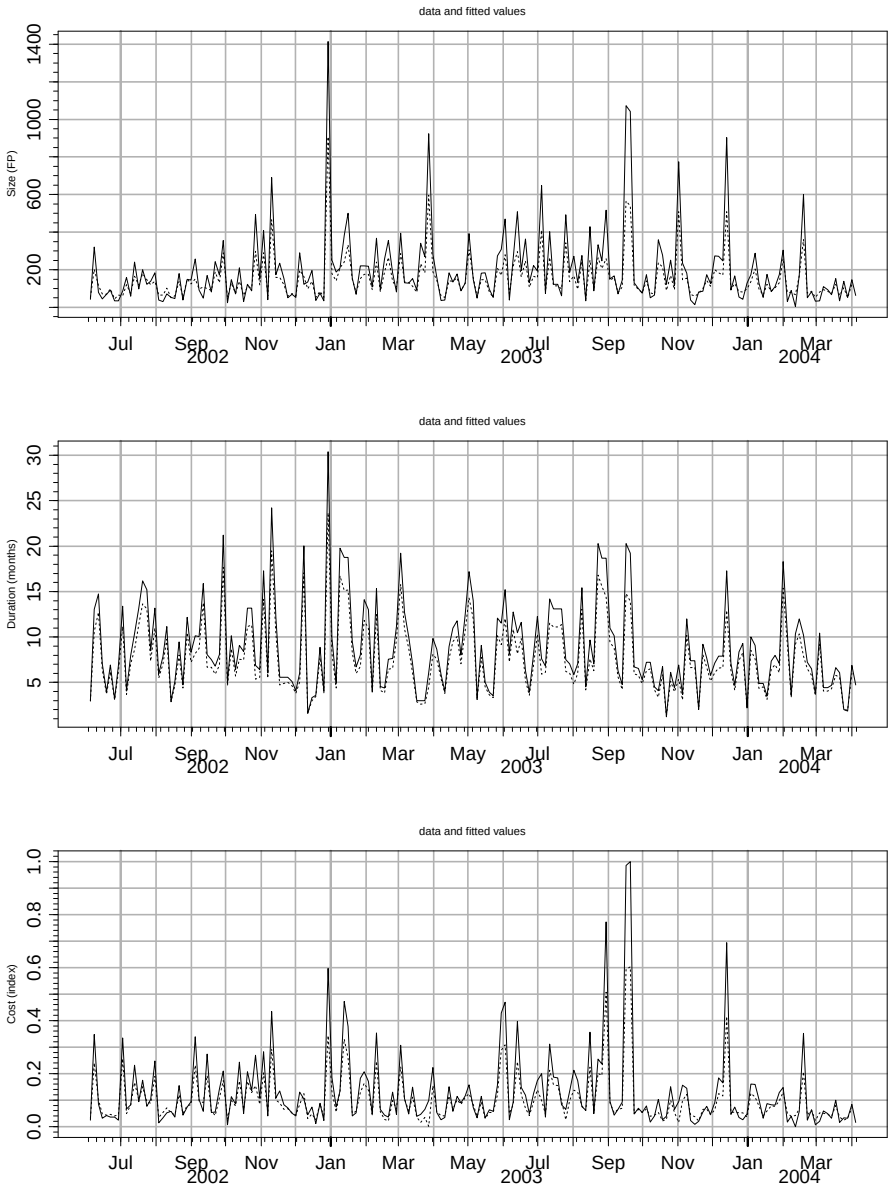

Figure 75: A Bayesian vector autoregression fit for the fdc-time series.

with all the fitted coefficients, including the found intercepts. If we see this matrix as a vector, we obtain a 12-dimensional vector β, whose first 4 values are the 4 fitted values for the f_t component of the fdc-series, the next four, are the fitted values for the d_t component, and the last 4 the fitted values of the c_t component. Then we formulate a linear hypothesis of the form:

$$H_0 : R\beta = r$$

where R is a fixed 9×12 restriction matrix, and r is the null vector. The matrix R is as follows:

$$R = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

The idea of this matrix R is that the restriction that Π is null, is expressed via the above matrix: $R(\beta)$ equals the coefficients of Π as a 9-dimensional vector. The null hypothesis states that this 9-dimensional vector equals r, which is the null vector. In our case the Wald statistic has the following formulation:

$$W = (R(\beta))^T (R(R\sigma^2 R^T)R^T)^{-1} R(\beta)$$

where σ^2 is the variance covariance matrix of the error term. We realize that the Wald statistic looks somewhat intimidating, and as an exception, we give the verbatim Splus code that we used to calculate the above value W.

```
R = matrix(c(
  0,1,0,0,0,0,0,0,0,0,0,0,
  0,0,1,0,0,0,0,0,0,0,0,0,
  0,0,0,1,0,0,0,0,0,0,0,0,
  0,0,0,0,0,1,0,0,0,0,0,0,
  0,0,0,0,0,0,1,0,0,0,0,0,
  0,0,0,0,0,0,0,1,0,0,0,0,
  0,0,0,0,0,0,0,0,0,1,0,0,
  0,0,0,0,0,0,0,0,0,0,1,0,
  0,0,0,0,0,0,0,0,0,0,0,1),
  9,12,byrow=T)
beta = as.vector(fdc.bvar$coef)
avar = R%*%vcov(fdc.bvar)%*%t(R)
wald = t(R%*%beta)%*%solve(avar)%*%(R%*%beta)
```

Let us explain this code. the letter `R` defines the restriction `9x12` matrix. The string `beta` takes the Bayesian $VAR(1)$ model's coefficients via `fdc.bvar$coef`. And `as.vector()` turns this into a vector. The string `avar` calculates an intermediate result: $R\sigma^2 R^T$, so `%*%` is matrix multiplication, and the transpose is invoked via the call `t()`. Then `wald` computes W. Note that `solve` is Splus syntax for taking the inverse of a matrix. The outcome of this exercise in matrix manipulation is $W = 712.3667$. Now under the null hypothesis, the Wald statistic behaves like a χ^2_q distribution with q the number of restrictions, which is nine, since we want to know whether all nine coefficients of Π are zero. It turns out that $1 - \chi^2_q(W)$ is zero. This

leads to the following important conclusion: the hypothesis that Π is the zero matrix can be rejected at any significance level. In other words, the Bayesian $VAR(1)$ model is stable, and the found coefficients really help in explaining the fdc-time series: the three values of fdc at time t are explained by the three values at $t-1$. This gives us further evidence that the movements of the trivariate fdc-time series are interrelated. So we can use the model to forecast the three variables. But before we do this, we want to know more about volatility.

10.1 Volatility

Another aspect that we like to investigate is whether the volatility of the fdc-time series is somehow related. That is, if the function point data shows large variance at some point in time, does this translate into large variance for the durations and/or the cost index? In the univariate case it turns out that the function points and the durations do not show signs of *ARCH*-effects, but the cost index shows weak evidence of *ARCH*-effects. We found with the Lagrange Multiplier test the following LM-test statistics: 9.3, 20.7, and 32.4 respectively, giving p-values: 0.99, 0.54, and 0.07, the latter showing a p-value indicating some *ARCH*-effects. We are going to fit a multivariate generalized *ARCH* model to the fdc-series, also called a *GARCH* model, or *MGARCH* model. We will not discuss *GARCH* processes in detail, since we only want to know whether there are co-movements in volatility. And we do not provide the actual fit in a set of (somewhat complex) formulas. Instead we will visualize the *GARCH* model, as we plotted other models as well in this paper. For more information on the subject of *GARCH* modelling we refer the interested reader to [27, 5, 63, 40, 6].

Let's see what happens if we look at the fdc-series as a whole. To that end, we fit a multivariate *GARCH*-model on the fdc-time series, to see if the volatility of the trivariate time series shows interrelations. We fit a $GARCH(1,1)$ model to the data, and we depict the volatility in Figure 76. Indeed, there are clear signs of co-movements. At times when volatility is low in one of the three plots, we see this mimicked in the other plots. But there are quite a few places where we do see large volatility, and then this is mimicked in at least one of the other plots, for instance, consider the spike in December 2001 of plot 1. We see a sharp increase in volatility of the function point data, which is also present in plot 2, and to a lesser extent in plot 3. The idea behind such co-movements is that a large variance displayed in function points, must also imply a large variance in cost and/or duration in principle. The fact that this is not present in both cases makes sense. Namely, if you keep schedules constant despite volatile IT-project sizes, this will induce large cost-volatility. This effect is called time compression [88, 87], and is clarified by an empirically found cost-time trade-off formula [68, 67]:

$$c \cdot d^{3.721} = constant$$

Furthermore, the same empirical relation clarifies that if we keep the cost constant for volatile IT-project sizes, this must be reflected in schedule volatility. And, if we relax schedules or cost a little, then we see some volatility in both other plots. This volatility effect is clearly visible in Figure 76.

Let us diagnose the $GARCH(1,1)$ fit. We plot the autocorrelations and cross correlations of the residuals and squared residuals in Figures 77 and 78. Both figures indicate that there is no significant serial correlation left in the residuals and the squared residuals. Indeed a multivariate portmanteau test of the Ljung-Box type with as null

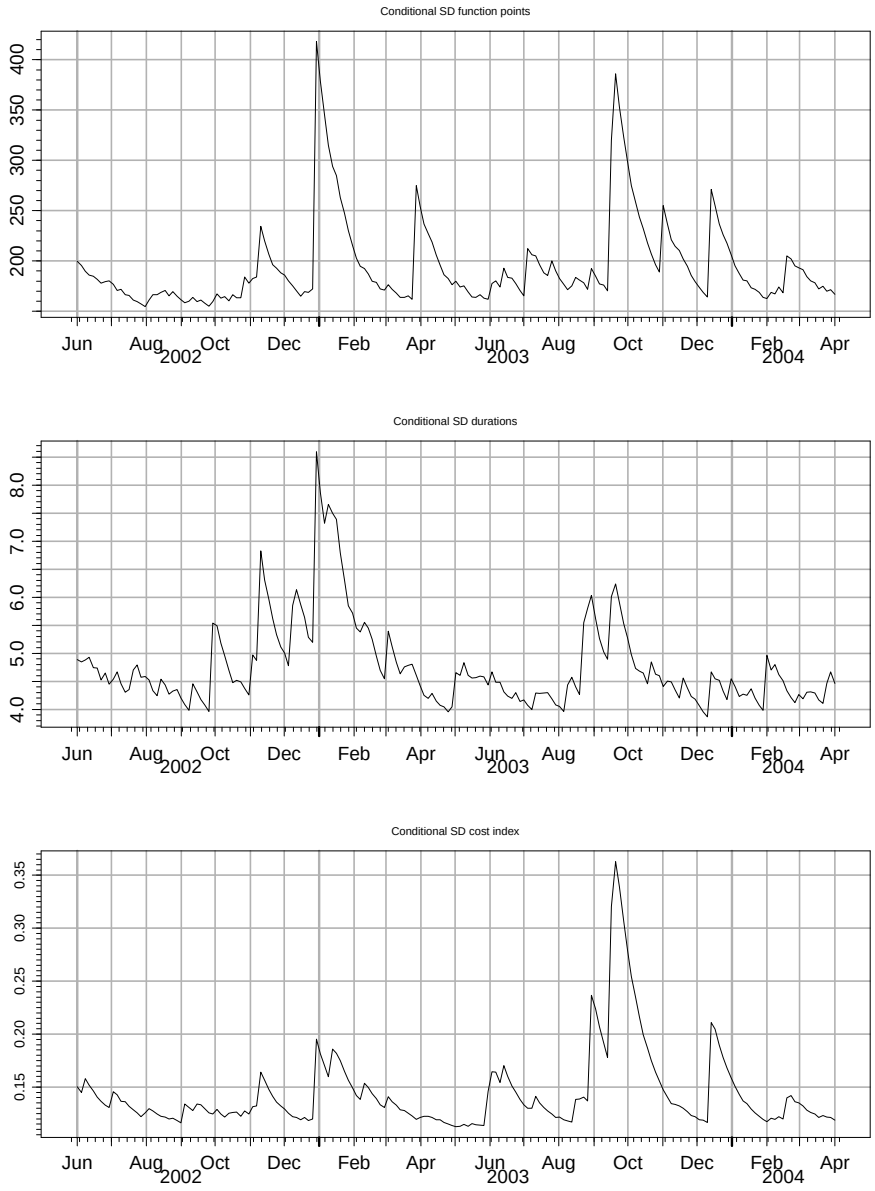

Figure 76: Trivariate conditional standard deviation as fitted from a $GARCH(1,1)$ model for the volatility of the fdc-time series.

hypothesis that there is no serial correlation, gives a test statistic of 93.1264, with a corresponding p-value of 0.8453, indicating that there is no evidence to reject this null hypothesis. In this test we also looked for cross-correlations. In a component by component test, we find for each of the three components of the fdc-time series that there is no evidence for residual autocorrelation. The test statistics for function points, du-

ration, and cost index are respectively: 12.0090, 7.1176, and 9.5735, with p-values: 0.4450, 0.8497, and 0.6533, indicating that there is no evidence to reject the null hypothesis that one of the components still displays serial correlation. Likewise we did the same for the squared residuals, to formally test that there is no autocorrelation in the residuals of the variance left as well. Indeed this is the case, both for the multivariate test with a Ljung-Box modified Q-statistic of 79.5804 plus a large p-value: 0.9816, and for the components we find 2.4668, 6.0713, and 11.3654 for the test statistics, with p-values: 0.9983, 0.9124s, and 0.4979, indicating no evidence for residual *ARCH*-effects. So the found *GARCH* model is adequate. This implies that we can use the model to predict future values and volatility of the trivariate time series.

10.2 A third forecast

With the found models we carry out a few predictions. In Figure 79, we predict the values of the function points sizes, durations, and cost index. These predictions show that the cost index slightly increases, the durations slightly decrease, and the function points remain fairly stable. If we look more closely, we notice that the lower error margins of the forecasts are much below zero, which is not in accordance with the physical reality of the variables. To that end we also carry out a conditional forecast: we put soft conditions on a variable that we understand best, or on which we know aspects in the near future. In this case, we impose soft conditions on the function point totals. We think that there will be no systems under 50 function points, and no systems above a thousand function points. Equally as well we could have conditioned the cost index via our prediction of the cost index per function point (as visualized in Figure 42), but we opted for the function point soft conditions, since often planned software projects do contain indications on size rather than cost per function point. From this conditional prediction, we find that the function point totals will increase, and that this translates in longer durations, but not much higher costs. Whether we look at the unconditional or the conditional predictions, the models do not misbehave in the sense that intrinsically impossible predictions are the result.

Furthermore, we are curious to a volatility prediction for the three variables function points, duration, and the cost index. In Figure 81, we plot a 12-step ahead forecast based on the $GARCH(1,1)$ fit. It turns out that the forecast predicts an increase in volatility, which means that the volatility of the cost index per function point will most likely increase. So we should look for such effects in the audit set as well.

Since we have a prediction of the function points and the cost index, we can now calculate the cost index per function point forecast based on them. Normally this is fully trivial, but since we indexed the costs for confidentiality reasons, we de-indexed the cost index, then divided these costs by the 12-step ahead forecasts of the function points, and indexed this again, resulting in a 12-step ahead forecast for the cost index per function point. The result of this indirect forecast is depicted in Figure 82. As can be seen, the prediction also indicates a rise in the cost index per function point, based on the fitted *GARCH* model for the fdc-time series. This finding is in accordance with our direct forecasts, that also predict a rise for the cpf-time series (cf. Figures 42 and 49).

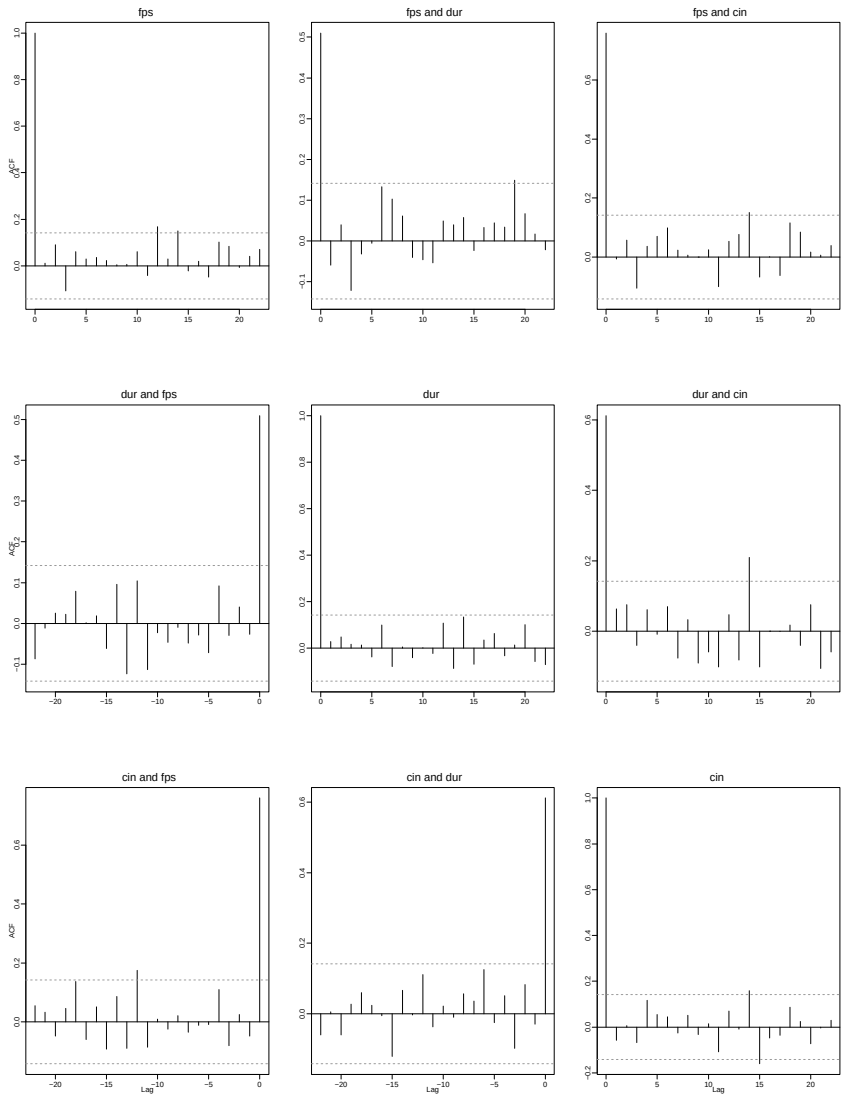

Figure 77: Autocorrelations and cross-correlations for the residuals of the $GARCH(1,1)$ fit for the fdc time series.

11 A FINAL ANALYSIS

Thus far, we carried out a number of analyses on what we called the *research set*. We recall these data were provided in an early stage, so that accuracy, reliability, plausibility, trends, predictions, methodology, and more can be worked out without too much

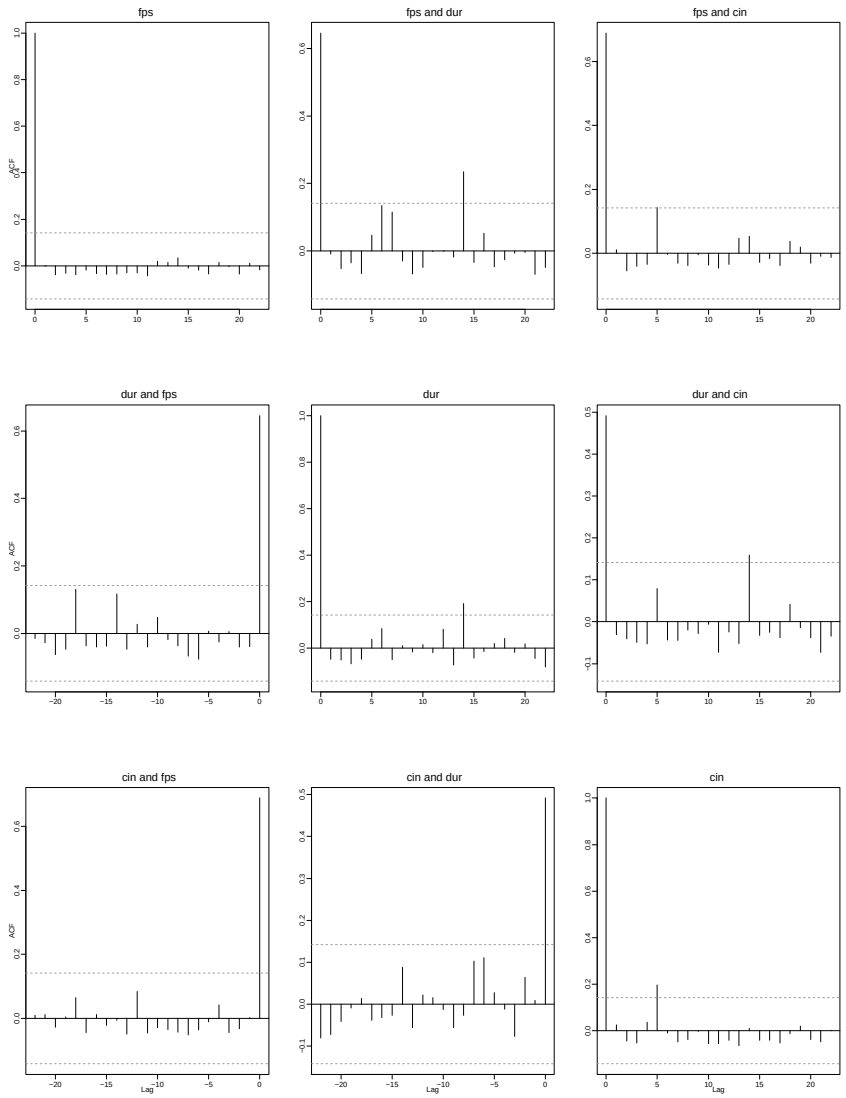

Figure 78: Autocorrelations and cross-correlations for the squared residuals of the $GARCH(1,1)$ fit for the fdc time series.

time pressure. Based on early findings, also qualitative assessments can be carried out, and outliers can be investigated. The next phase is a final audit, where the results of the extensive analyses on the research set are used as a basis for a final analysis of the latest available data. This analysis needs to answer whether the expected benefits of the software process improvement program are met or not. We recall that this is the

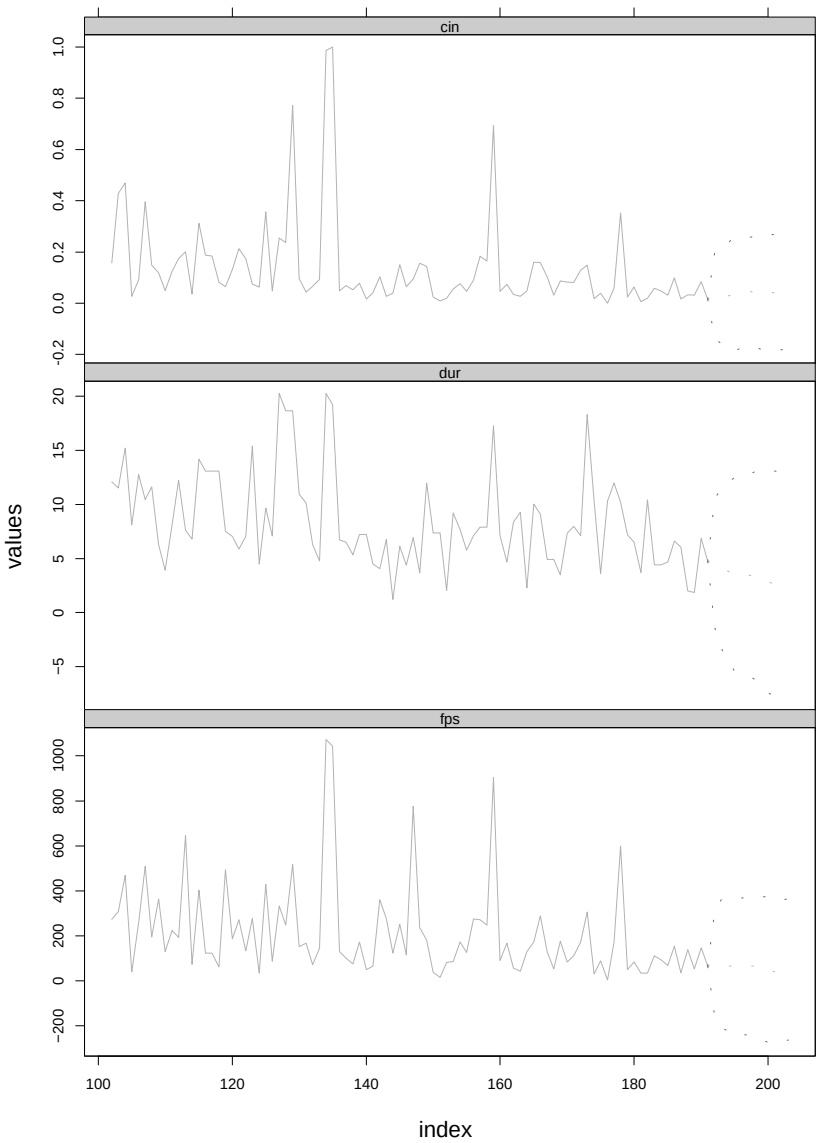

Figure 79: Predictions of the fdc-time series components based on the Bayesian $VAR(1)$ model for the fdc-time series.

audit set. The audit set contains 311 IT-projects, and a total of 58146 function points produced. A final analysis should not take too much time since as much as possible data must be taken into account for the audit. In this section we give an impression of the final analysis to support an audit.

For a start, we will depict the cost index per function point data of the audit set (see

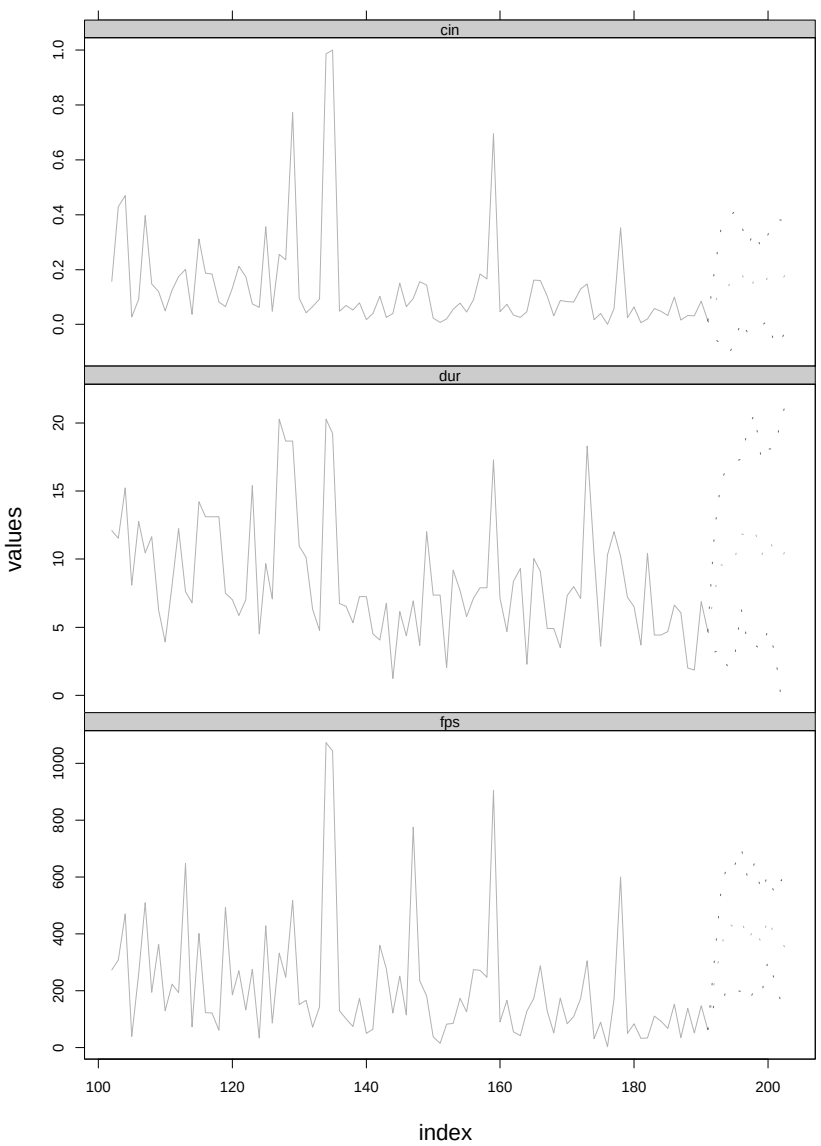

Figure 80: Conditional predictions of the fdc-time series components based on the Bayesian $VAR(1)$ model for the fdc-time series. The conditions are set on the function point sizes that they will be between 50 and 1000 function points.

Figure 83). The research set contains information starting from June 2002, and ending in March 2004. Note that the audit set contains data from January 2001 until November 2004. It turned out that more historical data was available than was provided for the research. An interesting feature of Figure 83 is that all our predictions that the cost

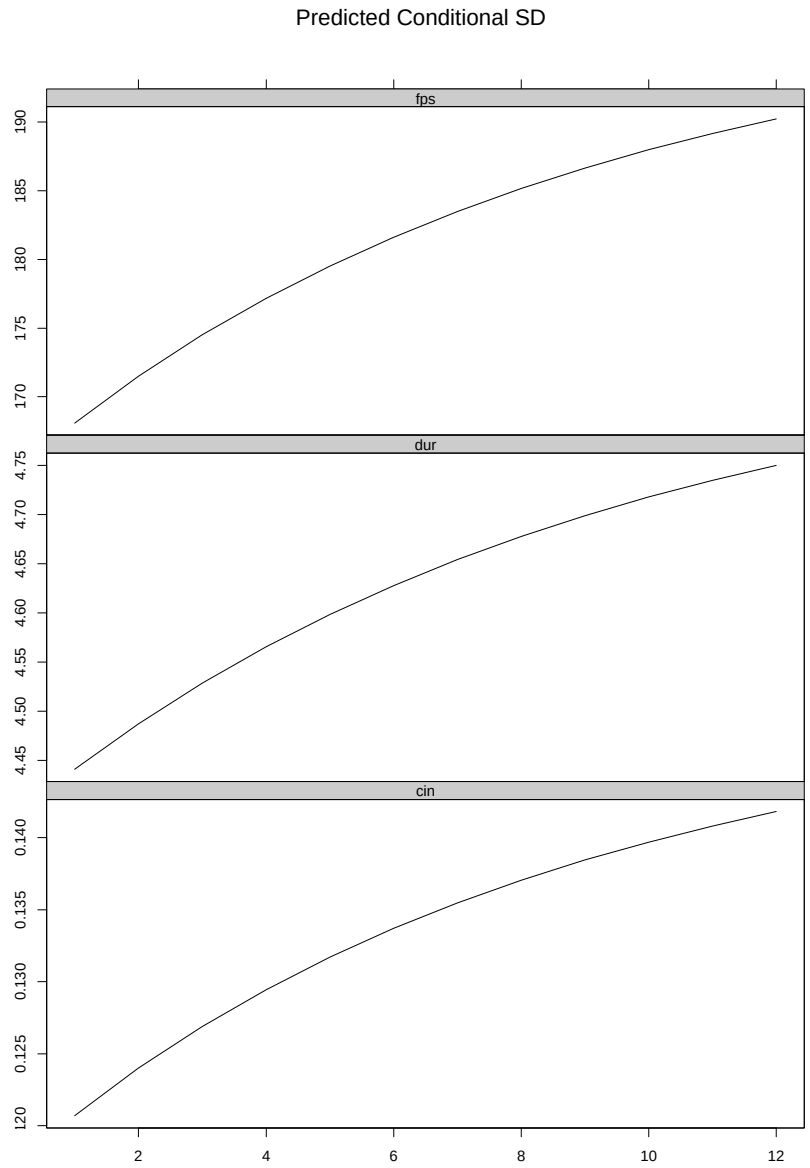

Figure 81: Predictions of the volatility of the fdc-time series components based on the $GARCH(1,1)$ model for the fdc-time series.

index per function point would rise, are in accordance with the audit set: shortly after March 2004 there is a rise, but also a fall in the last few data points. also the volatility increased a little. Apparently, the found models were all capable of predicting the short term future of this time series. Recall that our first prediction based on an $AR(2)$ model that does not take outliers into account gave this result, see Figure 42. To obtain

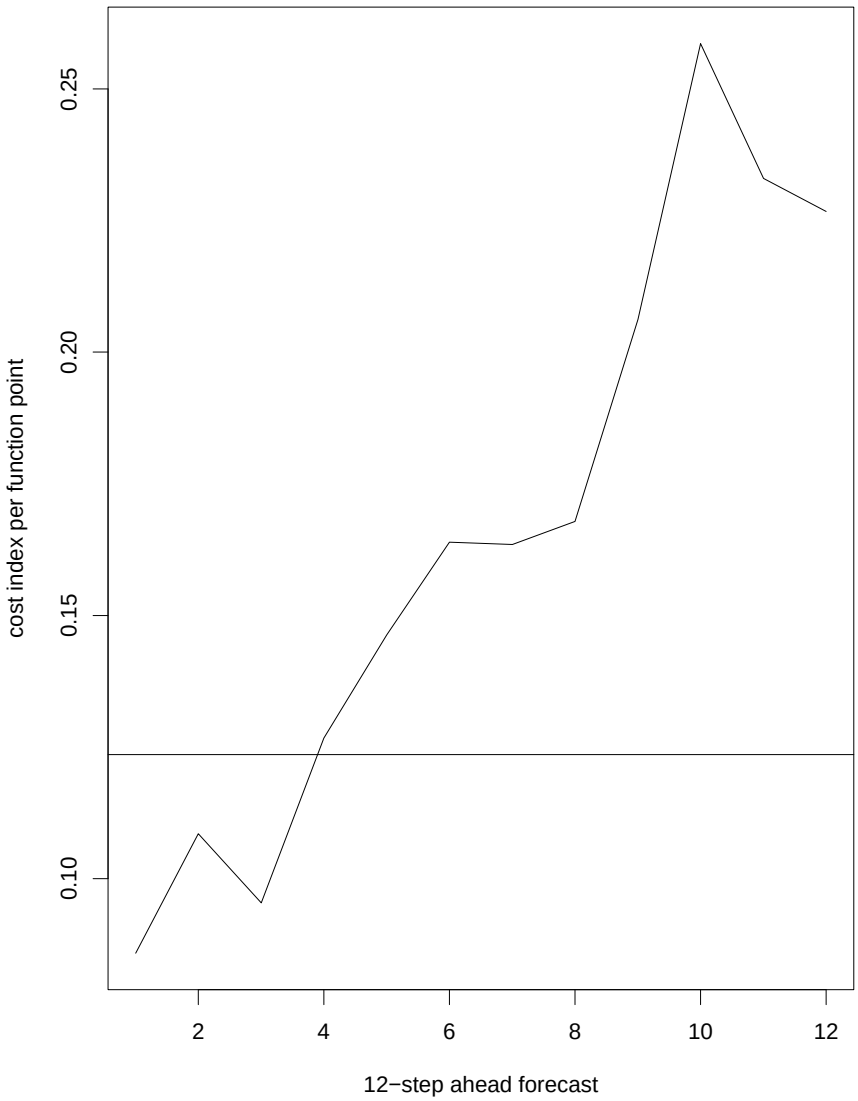

Figure 82: An indirect 12-step ahead forecast for the cost index per function point based on the Bayesian vector autoregression model for the fdc-time series.

more certainty about this trend, we investigated the research set in deeper detail, and took outliers into account. This led to a forecast as depicted in Figure 49. Also in that forecast we found a rise in the cost index per function point. Finally, in our multivariate time series analysis we carried out a prediction that also displayed a rise in the cost index per function point, see Figure 82. From the research set we learned that we

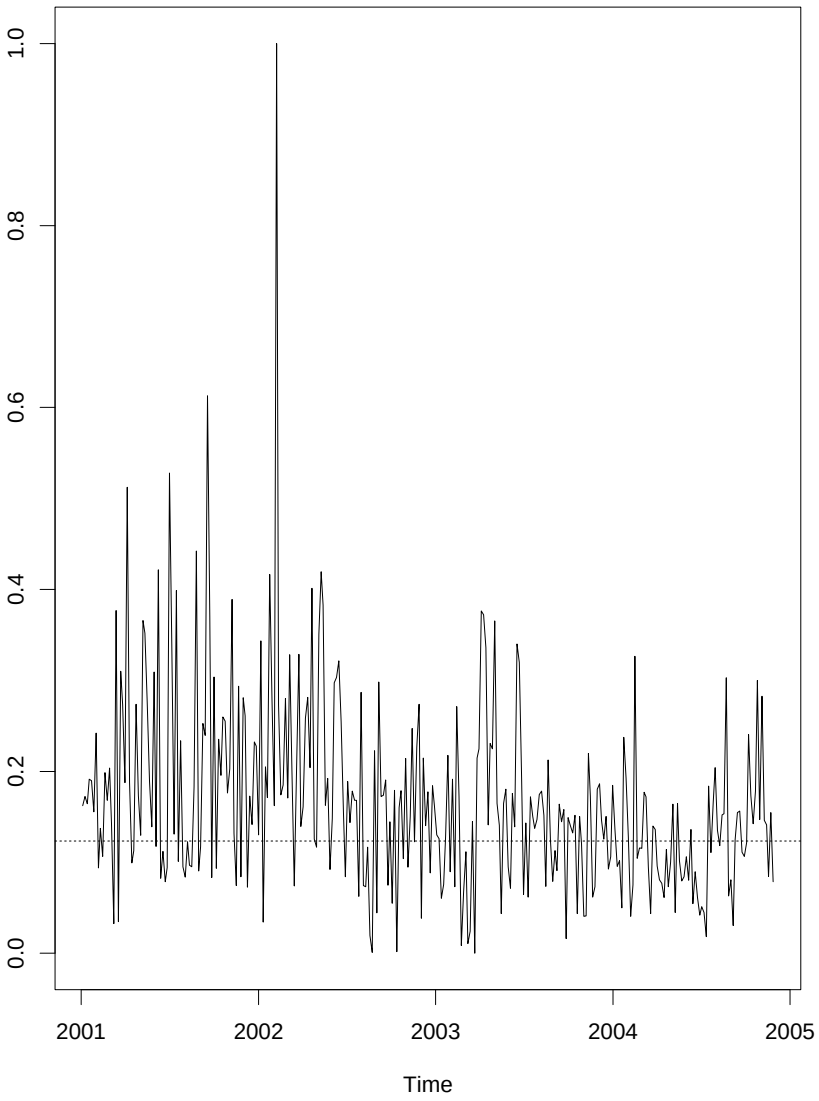

Figure 83: Cost index per function points for the full audit set, plus the target line.

probably have to fit a robust *ARIMA* model to the audit set, which we will do next.

11.1 Robust modelling

The analysis of the audit set turned out to be much more involved than the analysis of the research set, in the sense that the model is not as crisp. We give an impression of

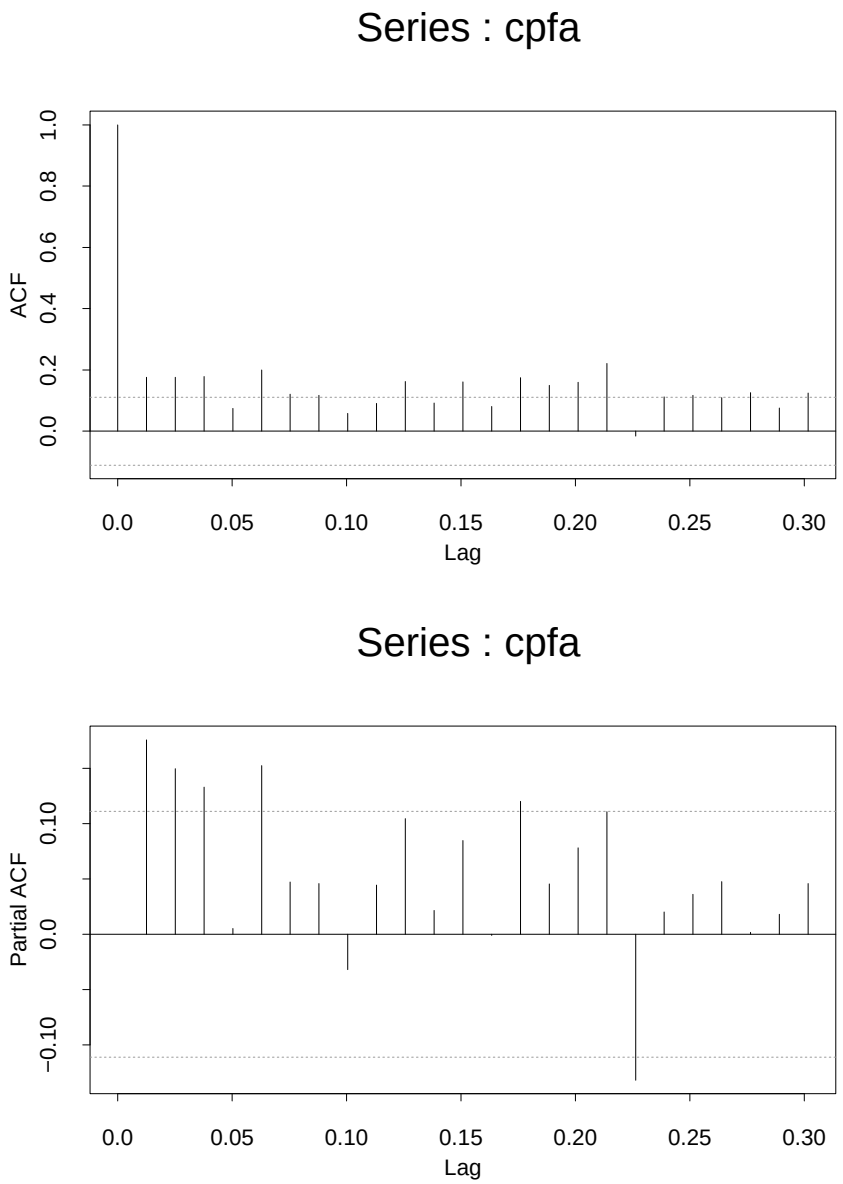

Figure 84: Autocorrelations and partial autocorrelations for the full audit set.

this analysis. In order to investigate the expected autocorrelation of the audit set, we calculated the autocorrelation and partial autocorrelation functions of the audit set. We depict the result of this in Figure 84. The notation cpfa that we readily see in this figure is short for cost index per function point for the audit set. As can be seen, the first 5 lags are significantly different from zero in plot 1, which directs to an autoregression order of 5. Moreover, the partial autocorrelation shows similar behaviour. Initially this might

result in an $ARMA(5,5)$ process, or an autoregressive moving average process of order $(5,5)$. Briefly, the autoregression means that lags of the data itself are correlated at most 5 steps back, and the moving average means that the error terms are also correlated with each other for at most 5 steps back. This seems not a parsimonious model, so we carried out a differencing analysis as well, which is that instead of analysing the audit set, we analyse their differences: $f_t - f_{t-1}$. But this did not lead to a more parsimonious model. Also, the fourth lag in both cases were not significant, which is indicating that selecting an $ARMA(5,5)$ model is not the best possible choice.

As a next step we carried out a robust *ARIMA* analysis. This led to a more parsimonious model. Moreover, the residuals of this model show little signs of autocorrelation. Let us start with the autocorrelations of the residuals of this robust fit in Figure 85. The abbreviation cpfa1.iorob is the object that we created by doing the robust fit: the 1 is for a demeaned version, the io refers to an innovational outlier included search, and the rob is short for robust. The residuals—also called innovations—as they are called in the output of the Splus macro we used for the robust fit, show no autocorrelation, except that at lag 5 and 6 the first possibly spurious significant autocorrelation starts. This is an indication that the found model seems fair.

A second test we carried out on the robust fit for the audit set is to calculate the partial autocorrelations, as shown in Figure 86. Also here the initial lags are not significantly different from zero. The lags starting at 5 show significant differences from zero. Also here we can assume that little serial autocorrelation is left, at least that the found model seems a fair approximation.

A third diagnostic is to test whether the residuals are plain white noise. This is easily seen using the Q-Q plot as depicted in Figure 87. As we can clearly see, the residuals are not normally distributed, since the Q-Q plot is not a straight line. This means that the model is not final yet. We suspect that we can model the squared residuals further and fit a *GARCH* model to the audit set. For the final analysis we do not need to do this, since we are not going to use the model for prediction purposes. In this phase we are interested to know whether there is autocorrelation at all, which is the case, since the found robust fit is an $AR(2)$ model. Moreover we are interested in the outliers, since they can impact the outcome of the audit significantly as we have seen in the analysis of the research set.

In order to otain an idea of the outliers and their impact we first plot in Figure 88 both the audit set and its cleaned version. As can be seen, the outliers are found in the beginning of the time series, and not at the end. Also we like to note that finding outliers is not a conservative operation. We found outliers for a window of a series that can be different than outliers for the entire series. This is also the case for the audit set. We found all three types of outliers: a level shift, 4 additional outliers and 1 innovational outlier. One additional outlier in the research set is now innovational. A level shift in the beginning of the series probably caused this additive outlier to become an innovational one.

In Figure 89 we depicted the impact of each type of outlier. Note that there are as in the research set 6 outliers, but they do not all coincide. The first outlier is a level shift, as discussed in Section 7. We recall that this means that from some time on, there is a constant impact, in our case this impact is 0.02523. This kind of outlier impacts the series in an early stage, which influences the residual outliers of the audit set minus the level shift. Hence, we obtain different outliers in general. In plot 2 of Figure 89, we find four additive outliers, with impacts 0.345, 0.3166, 0.4376, and 0.8117, respectively. And in between the first and second additive outliers, we find (as depicted in plot 3) an innovational outlier with impact 0.4054. The standard deviation of the resulting

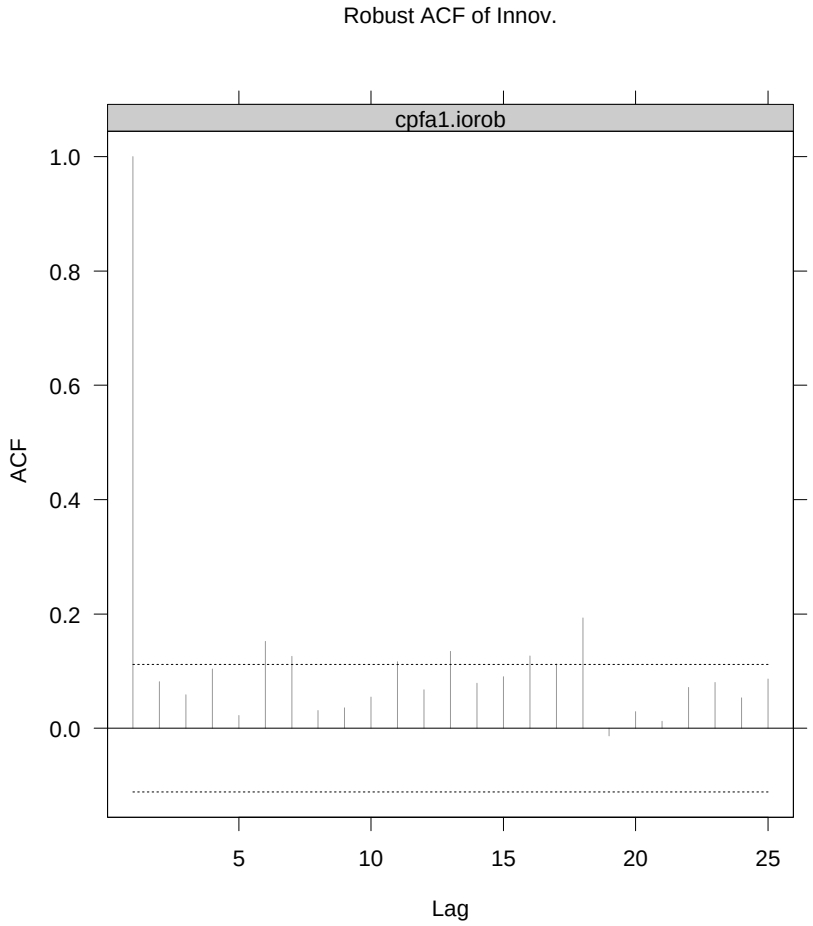

Figure 85: Autocorrelations for the residuals after fitting the audit set on a robust $AR(2)$ model with 3 types of outliers.

error term (the residuals, or innovations) is 0.07714. In terms of residual deviations, the standard deviation that we found for the robust fit for the research set was $\sigma^2 = 0.07767469$, which is slightly larger than the one we found now for the audit set. We know that this can be improved significantly by modelling this complex model by hand via a *REGARIMA* model (as done for formula 5). Then we have to model the three

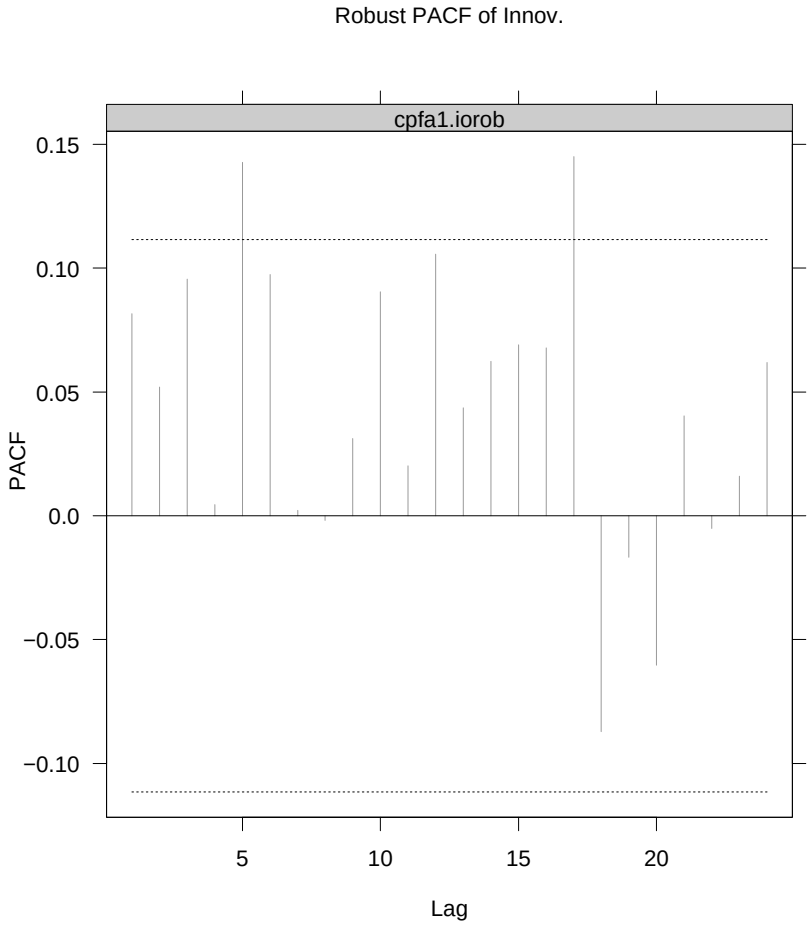

Figure 86: Partial autocorrelations for the residuals after fitting the audit set on a robust $AR(2)$ model with 3 types of outliers.

types of outliers explicitly, regress them out, and carry out an *ARIMA* analysis on the residues. Luckily, we do not need this handcrafted work, since we do not want to use the model for predictive purposes, but for audit purposes. For that we will apply smoothing both on the raw data and data from which we excluded the found outliers. The latter are accurately found, and the influence of the large standard deviation is not

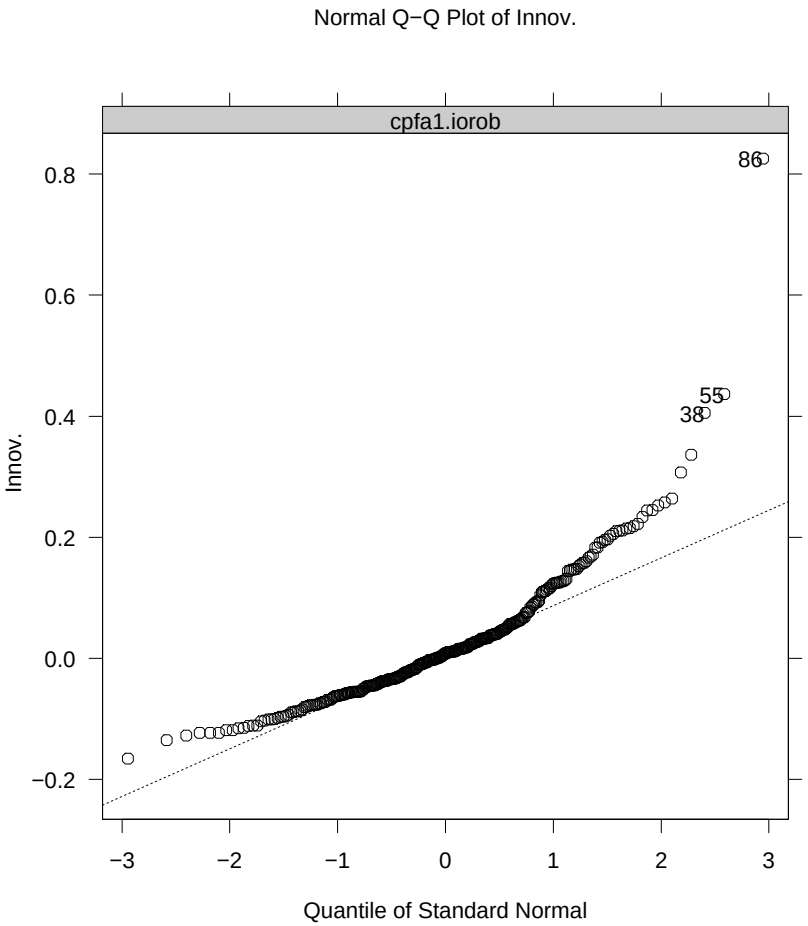

Figure 87: Q-Q plot of the residuals after fitting the audit set on a robust $AR(2)$ model with 3 types of outliers.

an issue then. So for this purpose we can work with this model.

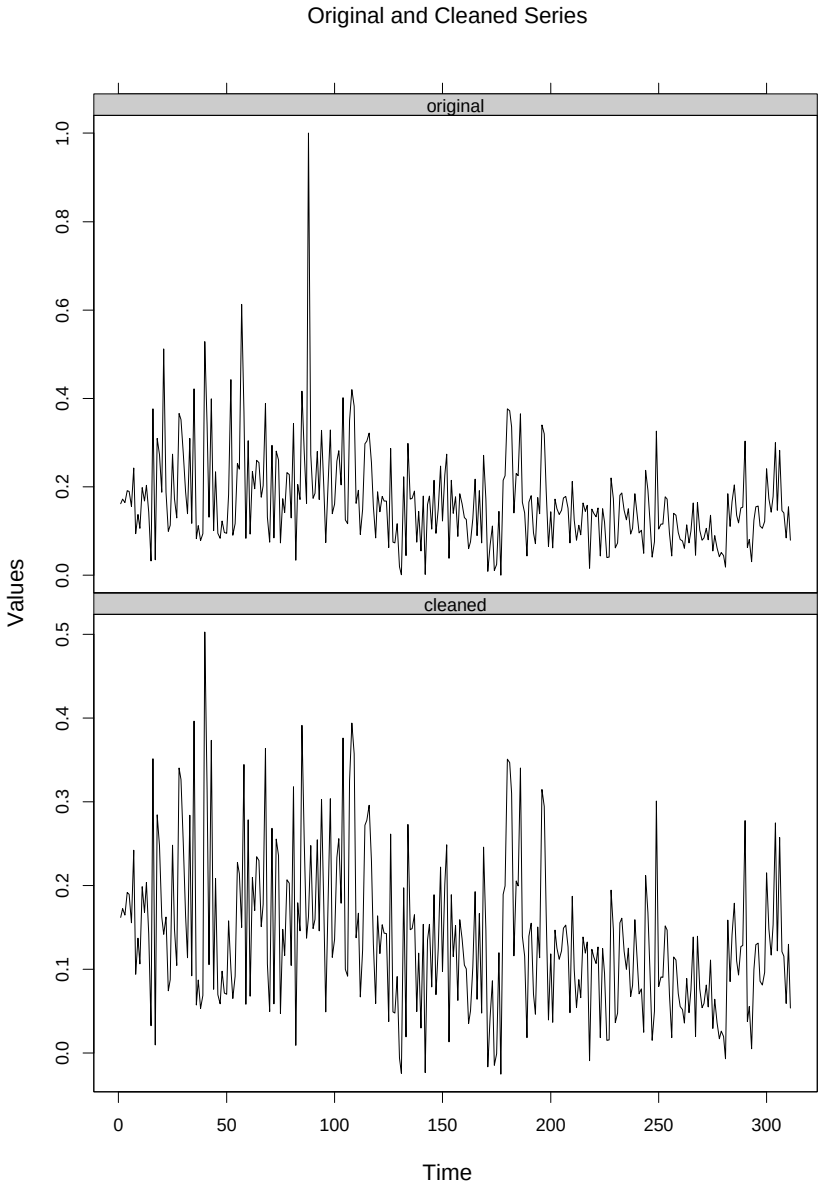

Figure 88: Plots of both the audit set and the cleaned version, where the outliers are removed.

11.2 Robust smoothing

This implies that we can now turn our attention to the smoothing operations that we carried out on the research set. Recall that the methods proposed by the organization did smooth the time series, but also concealed the trends, so we opted for a different approach, namely using smooth regression analysis as discussed in Section 9. We

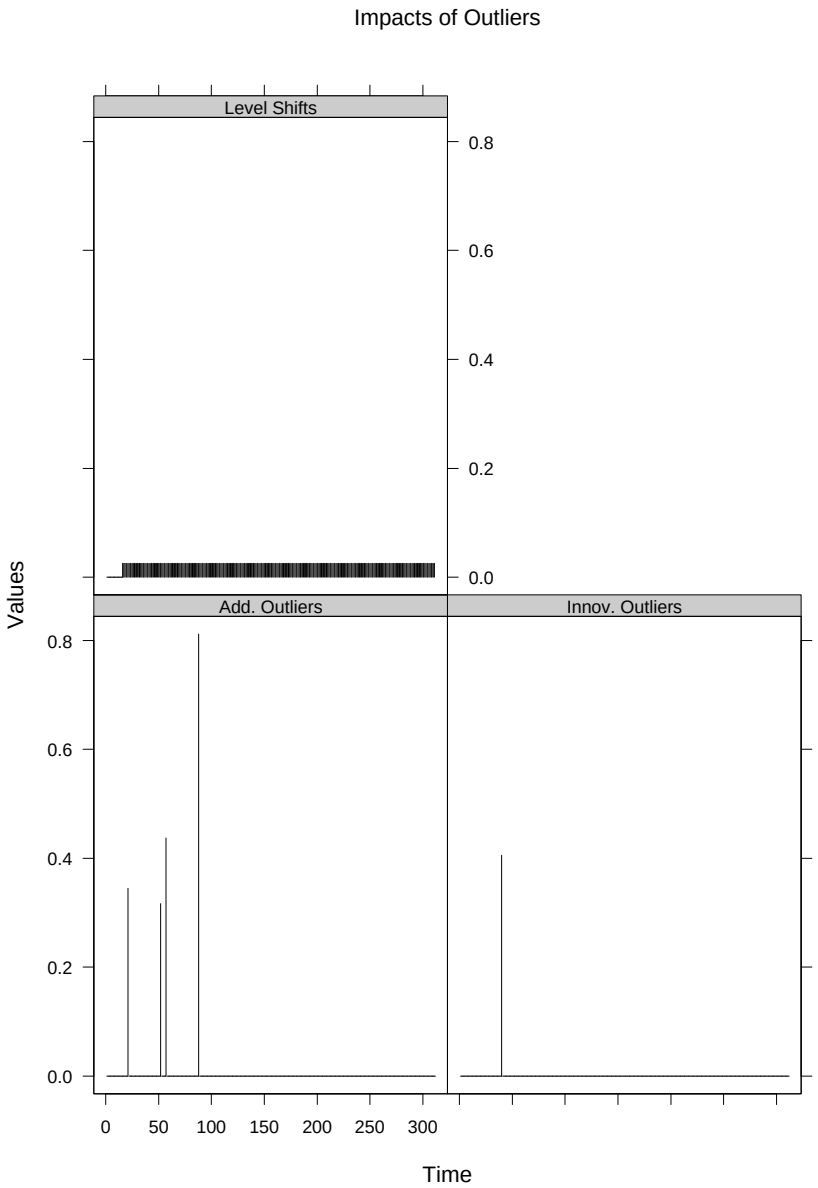

Figure 89: Impact of the various outliers: the level shifts, additive outliers, and innovational outliers.

depicted the result of the eight proposed smooths in Figures 90–97. Since it is always difficult to carry out a thorough qualitative analysis of the newly found outliers, we will as before smooth both the original and the cleaned version void of outliers. This to capture their influence on the smooths, and thus on the outcome of the audit. In Figure 90, we observe the following pattern: the super smooth of both the original and cleaned audit sets show a rise at the end. We also see that the original audit set

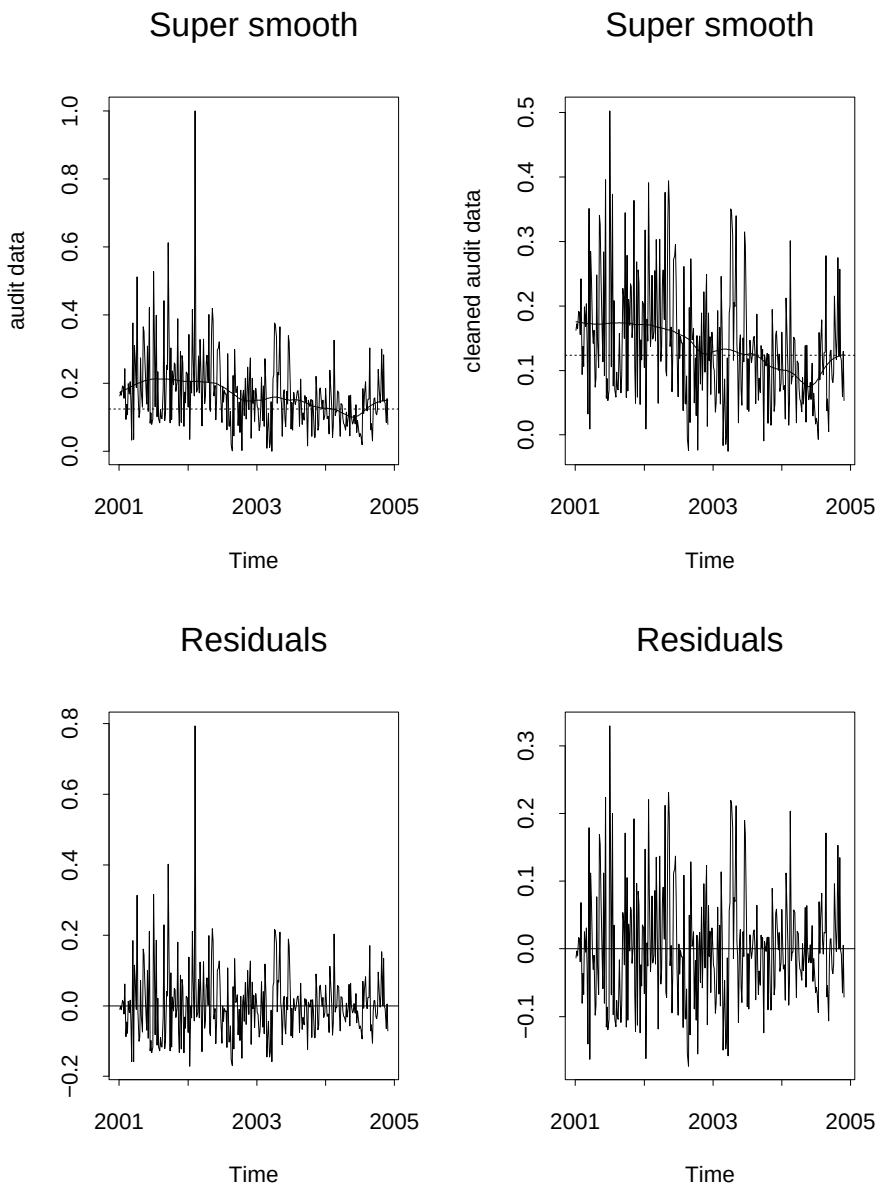

Figure 90: Super smooths of the cpf-time series, its cleaned version, and their residuals.

smooth rises above the target line, and the cleaned one does not. The residuals do not indicate autocorrelations, which is confirmed by a Ljung-Box test (p-value 0.2922). and although a first rough autoregression fit on the squared residuals does not lead to autocorrelation effects, we found with the Lagrange Multiplier test the following LM-test statistics for the original and cleaned audit set: 5.37 and 35.4 respectively, giving p-values: 1.00 and 0.06, the latter showing a p-value indicating weak evidence for

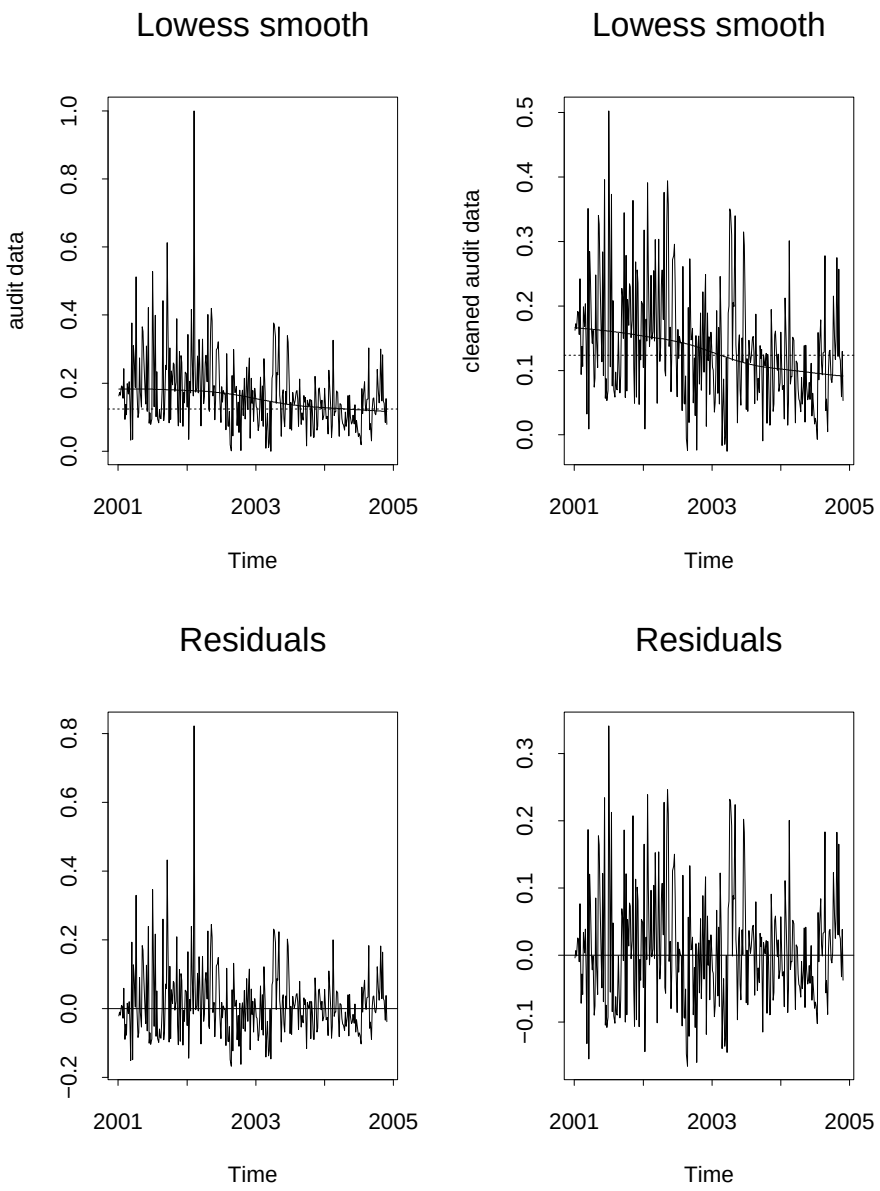

Figure 91: Lowess smooths of the audit series, its cleaned version, and their residuals.

ARCH-effects (see Table 2 for interpretations of p-values). Since in addition the shape of the super smooth is not too wiggly, it seems a fair estimator of the macroscopic properties of the audit set or its cleaned version.

In Figure 91, we depict the lowess smooth. As in the research set, the lowess smooth is the least wiggly smooth of the 8 possibilities that we used to extract the macroscopic behaviour of the cpf-time series. As can be seen in plots 1 and 2, both

smooths are monotonically decreasing, and both smooths go below the target line. Although this smooth looks really good, we have to be a bit careful. The residuals show weak evidence of serial correlation. As with the super smooth, there is no evidence for heteroscedasticity, and the presence of serial correlation can be formally rejected for the cleaned residuals of the lowess smooth. This smooth could also be used as an estimator, but it fails to alert the rise due to its strong dampening properties. The latter is not desirable.

The spline smooth depicted in Figure 92 shows similarities with the super smooth. Only the squared cleaned residuals show weak evidence of *ARCH*-effects. The pattern is more or less the same: we see a rise in the end, but now both smooths end above the target line. Again, given the shape of the smooth, this one will also do as an estimator of the macroscopic properties of both the original and cleaned audit data.

The box kernel smooth, is again the most rough kernel smooth, see Figure 93. It has the same pattern as the spline smooth, albeit less smooth. Both cases show a rise at the end, both above the target line. A tentative autoregression analysis reveals an order of 18 in the residuals, and a Ljung Box test shows indeed moderate evidence that there is serial correlation left. This is not found in the cleaned version. For this reason, and since the box kernel is a bit wiggly, we would not use the box kernel smooth as an estimator, although the final outcome will not be drastically different if you would use a box kernel smooth in this case.

The triangle kernel smooth of Figure 94 shows similar patterns as the box kernel smooth, a rise no matter the outliers, although the rise ends more near the target line for the cleaned data. There is weak evidence for residual serial correlation, and weak evidence for *ARCH*-effects in the residuals of the cleaned version. Given the shape, this is also a fair estimator for capturing the macroscopic effects of the audit set (or its cleaned version).

The Parzen kernel smooth also resembles the other kernel smooths (see Figure 95). There is weak evidence for residual serial correlation and *ARCH*-effects in the cleaned version. There is no evidence for such effects in the other cases: *ARCH*-effects in the residuals or serial correlation in the cleaned residuals. Also its wigglyness is fairly moderate, so this one could also serve as a fair means to measure overall software process improvement effects. Similarly, the normal kernel smooth shows similar results as the other kernel smooths (see Figure 96). And exactly the same properties as in the Parzen case hold for the residuals as well.

The robust median smooth displays as in the research set a fairly wiggly pattern, too wiggly for our purpose. It shows too many microscopic trends, and is therefore less suited for this analysis. Apart from that, there is a tentative autoregression order of 5 in the residuals, and a Ljung-Box test provides strong evidence that there is such serial correlation. There is no evidence for *ARCH*-effects in the residuals. The cleaned residuals also shows strong evidence of left-over serial correlation, and no *ARCH*-effects. For these reasons, it is not a good idea to use the latter smooth for assessing the macroscopic trends of the audit set. In Table 11, we summarized some numbers that assess the residuals of the smooths and their cleaned versions for the audit set. In this case the super smooth seems to have the best diagnostics to serve as the smooth to extract the trend from the audit set.

To compare all smooths, we plot them together in Figure 98. Now we can clearly see that the robust median smooth is not as smooth as we would like it to be, and that the others except the lowess smooth show fairly consistent patterns. In plot 2 we did the same for the cleaned variants. We can see a repeating pattern here: after a rise and a local maximum we see a decrease, and then another plateau, followed by

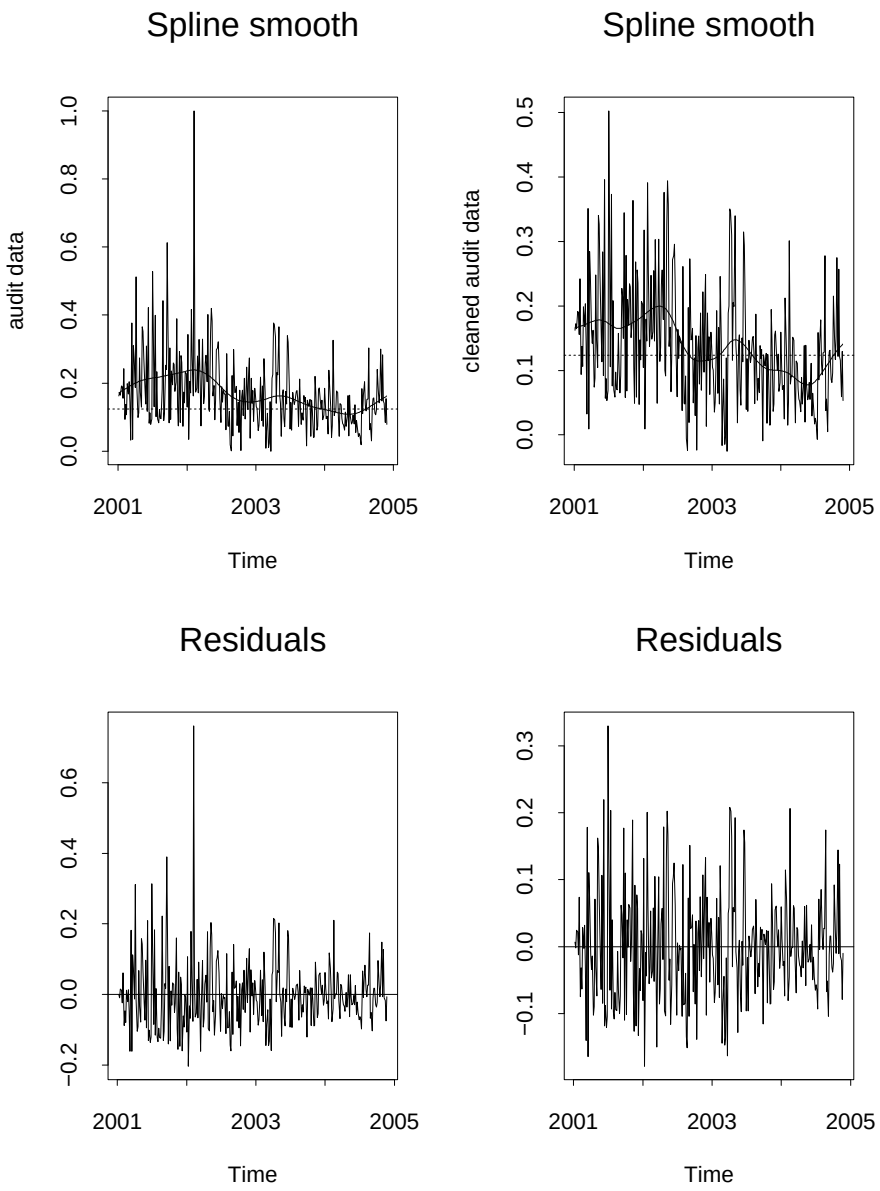

Figure 92: Spline smooths of the audit series, its cleaned version, and their residuals.

a rise to a local maximum, then again a decrease to a next (lower) plateau, followed by a dip and then another rise. In both smooths of the original and cleaned versions the same macroscopic pattern emerges. So the outliers do not affect the shape of the macroscopic trend, but they do affect the amplitude of the smooths. In plot 1 we see a larger variation than in plot 2. To get an even better view we remove the robust median smooth in Figure 99. Of course, we see the same pattern as before, only the scale is a

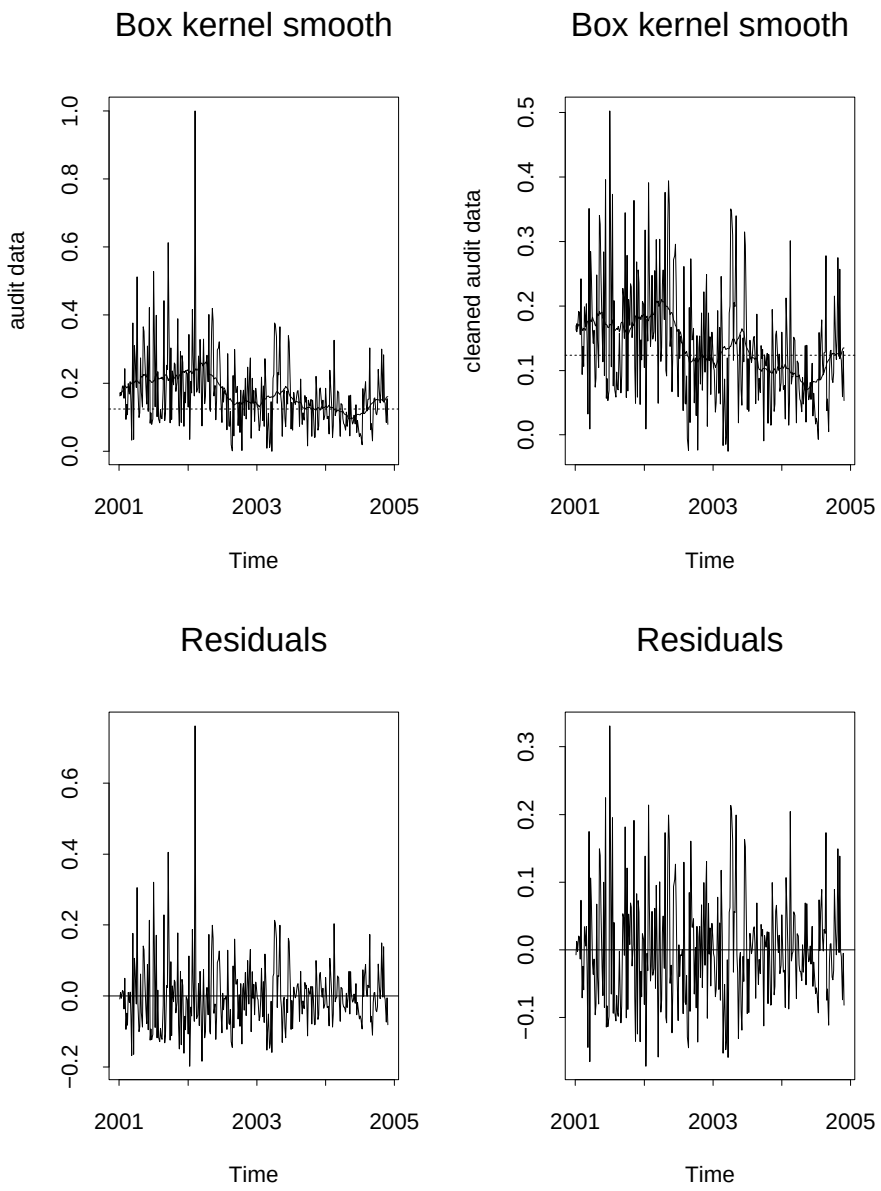

Figure 93: Box kernel smooths of the audit series, its cleaned version, and their residuals.

bit wider since the wiggly robust median smooth is gone.

What is important to realize is that the large dispersion of the cost index per function point is probably not caused by the process of constructing software alone. This effect is also caused by the business: more or less function points, more or less death-march projects, more or less outsourcing, and many more unknowns. These random effects plus the stochastic effects of constructing, maintaining and measuring software

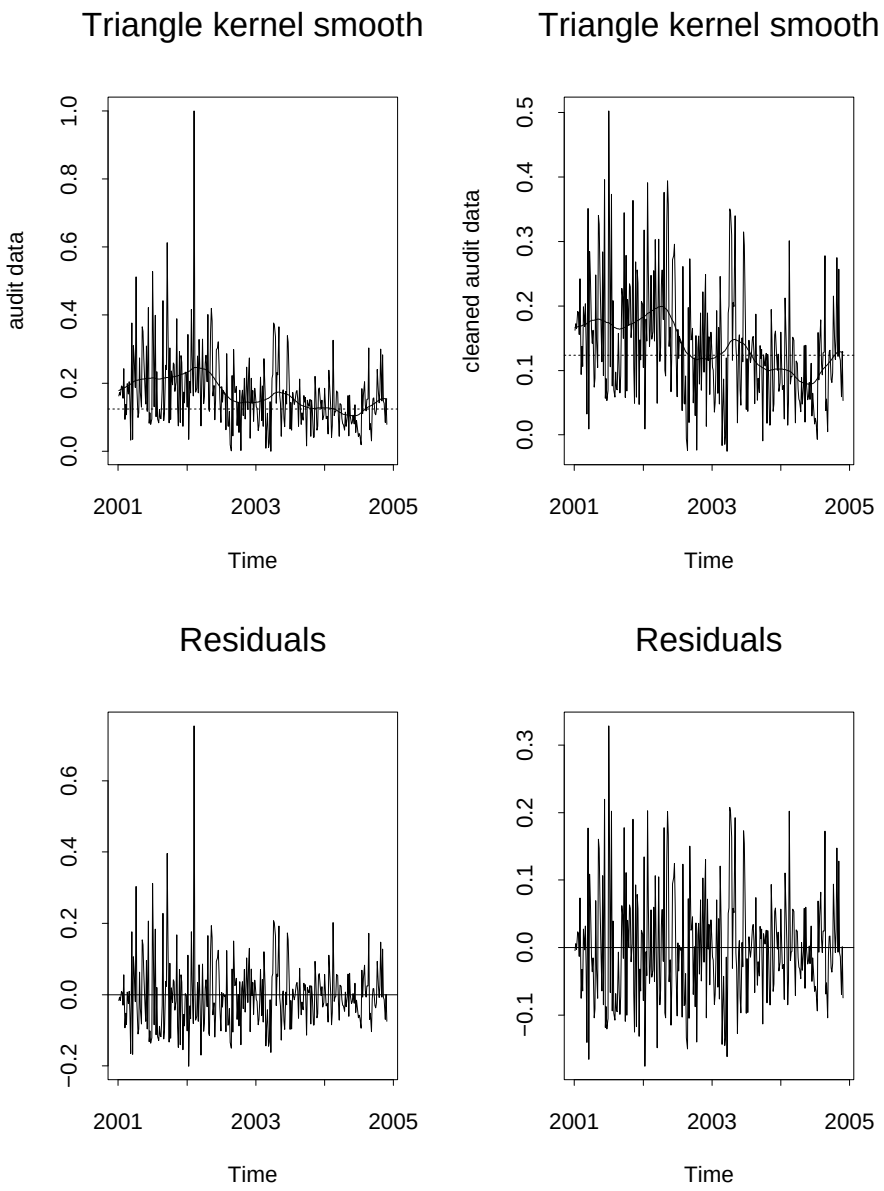

Figure 94: Triangle kernel smooths of the audit series, its cleaned version, and their residuals.

together are causing the large dispersion. It is therefore not a good idea to measure the cost index per function point at one point in time, say December 31, 2004. This can only be done if the smooth is monotonic (no wiggles at all), and if its residuals do not contain significant structure. The only one that might be a candidate for this kind of reasoning is the lowess smooth, but this one contains too much residual structure, so that we must conclude that the lowess smooth did not squeeze out all structure from

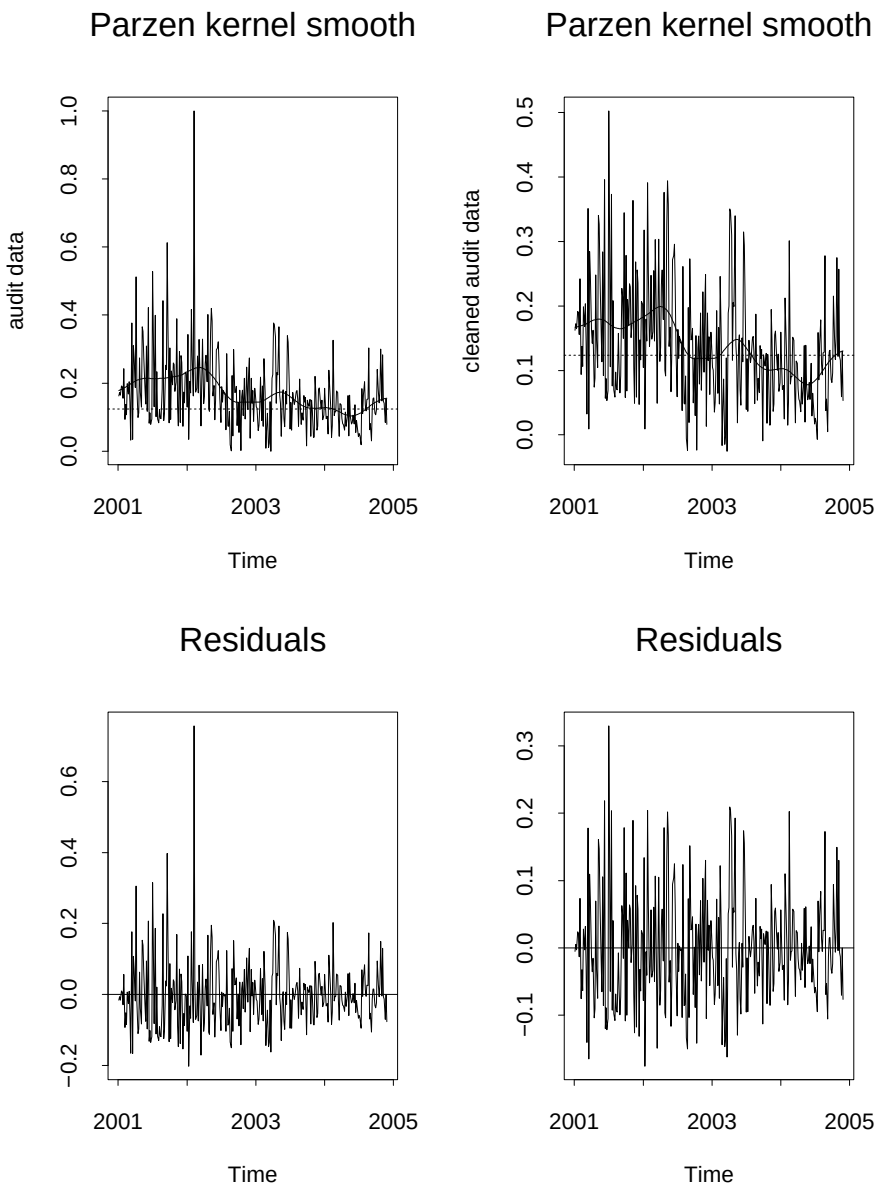

Figure 95: Parzen kernel smooths of the audit series, its cleaned version, and their residuals.

the audit set. Instead, it is wise to take the best behaving smooths, and if there is a clear downward trend, take the maximum of the smooth, and the minimum of the smooth. In effect this implies that the maximal value at the beginning is taken, which is severely damped by the smoothing operation, and the minimal value implies the same but then at the end of the time series. This method prevents that exogenous variation in the smooths at certain points in time determine the outcome of an audit, instead of a clear

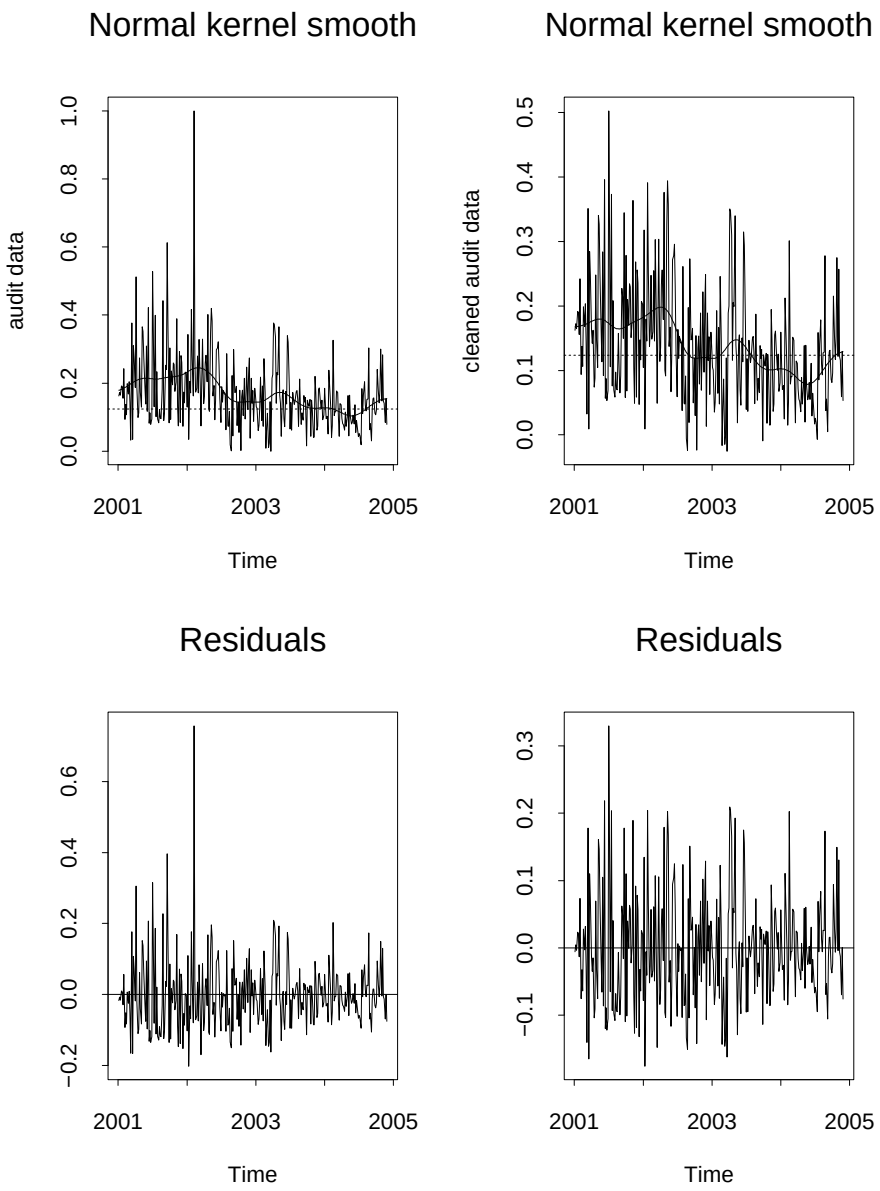

Figure 96: Normal kernel smooths of the audit series, its cleaned version, and their residuals.

trend.

To that end, we will provide in Table 12 some numbers. We note that the indexed target line has a value of 0.1235797. If we inspect the second column of Table 12, we see that in all cases the reported minimal values are under the target line, with and without outliers included. As soon as we look at the first quantile of the data, we observe that for the majority of the smooths of the original audit set the target is no

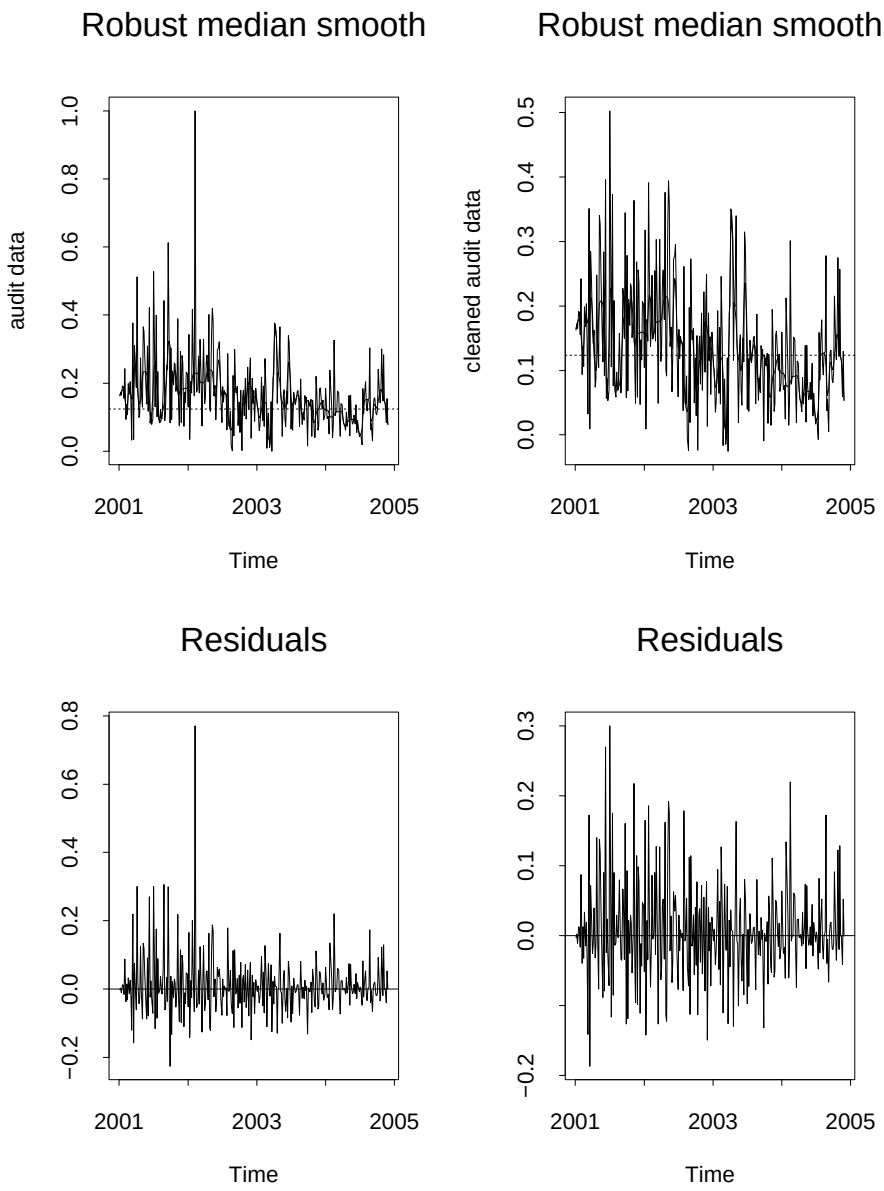

Figure 97: Robust median smooths of the audit series, its cleaned version, and their residuals.

longer met. For the cleaned version this is still the case. Starting from column 4 the target line is never met. This implies that it is an accomplishment to actually meet the target line, since it is not the case that the smoothing data on average is already below the target line. In order to make this more clear we depict the distributions of the four following aggregates in Figure 100. We took the data of the eight smooths, and made a nonparametric estimate of their probability density function. This is shown

smooth	res	res^2	cl. res	cl. res^2	LB	LM	LB cl.	LM cl.
Super	0	0	0	5	0.2922	1.0000	0.3609	0.0629
Lowess	5	0	2	1	0.0911	0.9998	0.2269	0.1849
Spline	0	0	0	5	0.1523	0.9999	0.2241	0.0506
Box	18	0	0	5	0.0388	0.9999	0.1932	0.1232
Triangle	0	0	0	5	0.0980	0.9999	0.2432	0.0526
Parzen	0	0	0	5	0.0855	0.9999	0.2135	0.0553
Normal	0	0	0	5	0.0916	0.9999	0.2317	0.0566
Median	5	0	7	5	0.0002	0.9999	0.0003	0.1242

Table 11: Analysis of residuals and the squared residuals plus their cleaned versions after smoothing for autoregression and *ARCH*-effects for the audit set.

smooth	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
6-point summaries of smoothened audit set						
Super	0.0999	0.1405	0.1549	0.1642	0.2021	0.2126
Lowess	0.1169	0.1288	0.1561	0.1536	0.1785	0.1828
Spline	0.1083	0.1369	0.1582	0.1688	0.2112	0.2386
Box	0.0943	0.1324	0.1580	0.1689	0.2090	0.2616
Triangle	0.1038	0.1330	0.1560	0.1690	0.2124	0.2459
Parzen	0.1031	0.1335	0.1566	0.1689	0.2130	0.2459
Normal	0.1038	0.1336	0.1561	0.1689	0.2125	0.2451
Median	0.0307	0.1164	0.1508	0.1570	0.1874	0.3642
All 8	0.0307	0.1307	0.1556	0.1649	0.2017	0.3642
First 7	0.0943	0.1325	0.1566	0.1660	0.2032	0.2616
6-point summaries of smoothened cleaned audit set						
Super	0.0747	0.1153	0.1315	0.1369	0.1712	0.1757
Lowess	0.0913	0.1031	0.1300	0.1290	0.1537	0.1663
Spline	0.0776	0.1096	0.1349	0.1386	0.1720	0.1997
Box	0.0690	0.1072	0.1328	0.1386	0.1707	0.2102
Triangle	0.0785	0.1077	0.1308	0.1386	0.1721	0.1995
Parzen	0.0779	0.1083	0.1314	0.1386	0.1716	0.1991
Normal	0.0786	0.1084	0.1309	0.1386	0.1719	0.1982
Median	0.0055	0.0909	0.1256	0.1304	0.1704	0.3390
All 8	0.0055	0.1050	0.1313	0.1362	0.1691	0.3390
First 7	0.0690	0.1069	0.1319	0.1370	0.1691	0.2102

Table 12: Summaries of the values of the smoothing operations, all combinations of the smooths, and the first seven, excluding the wiggly robust median for the audit set.

in plot 1 of Figure 100. We can see that the probability to be under the target line—independent of the time—is much lower than not meeting the target line. The same holds for plot 2 which is the density of the cleaned version of the eight smooths, albeit that it is somewhat easier to meet the target line if time is not taken into account. In plots 3 and 4, we provided the density functions for the 7 smooths excluding the robust median smooth. Plot 3 shows the density of the 7 smooths of the original data, and plot 4 its version without outliers. In all cases independent of time it is relatively hard

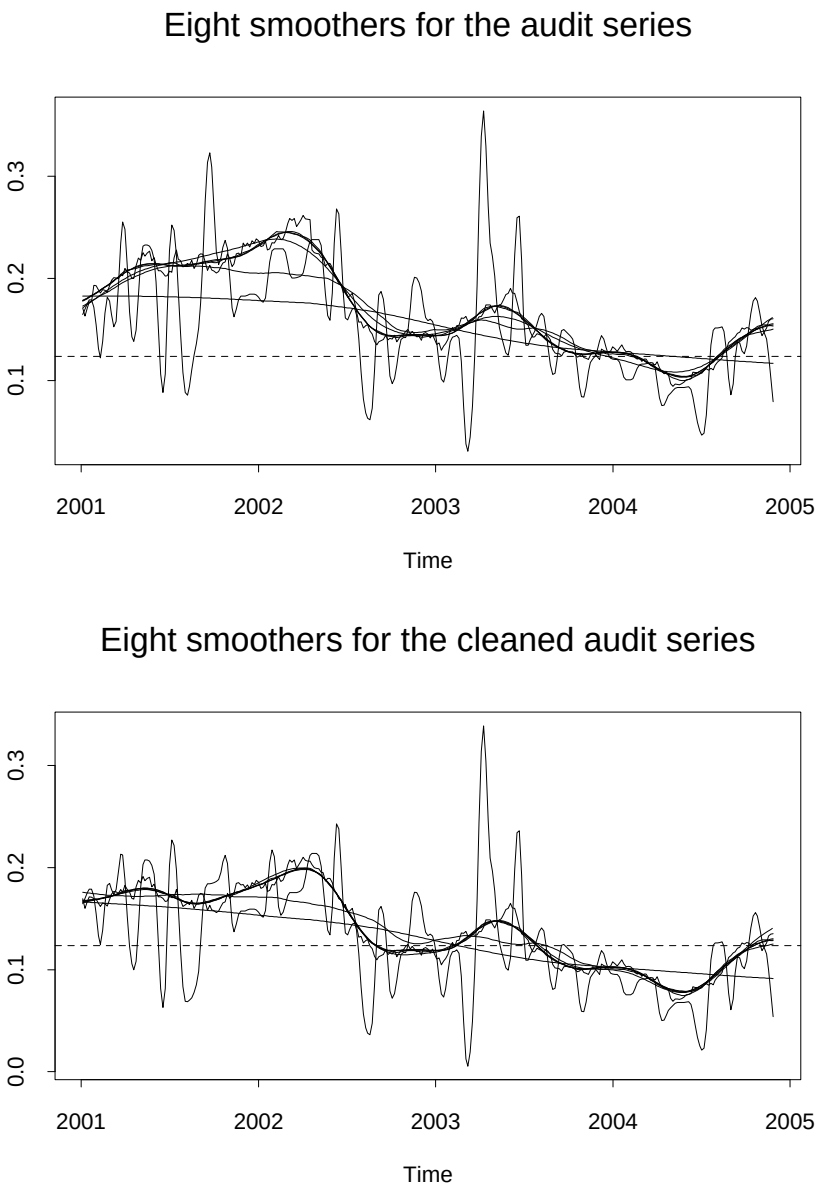

Figure 98: Eight different smoothing operations on the audit series, and the cleaned audit series.

to meet the target line. So, meeting the target depending on time at the end of the time series shows that there is a real improvement in place. Given the minimal values at the end, the maximal values at the beginning we must conclude on the basis of the data that the target is met.

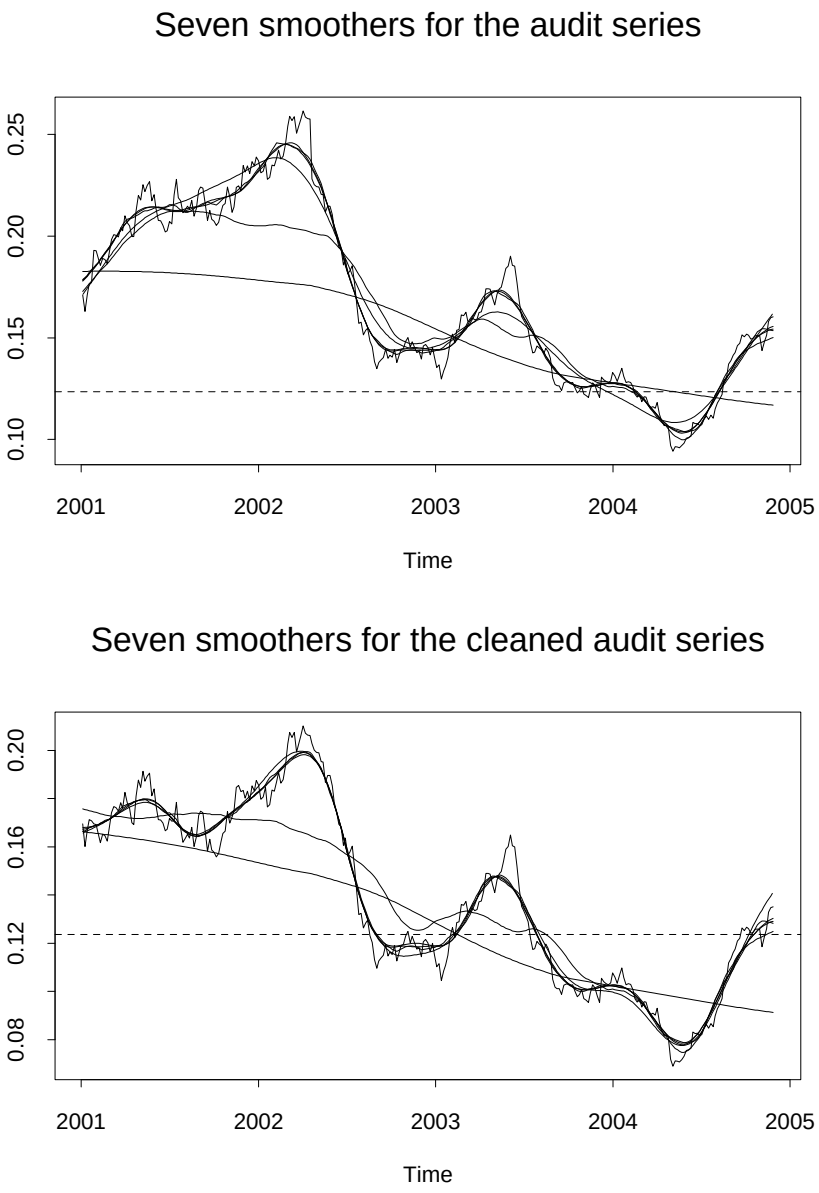

Figure 99: All but the robust median smooth on the audit series, and the cleaned audit series.

12 AUDIT

In this phase, we have all the analyses carried out to come to a final assessment of the SPI-program. The metrics group is an important spearhead to drive SPI-programs. We know that in 80% of the cases setting up such a department fails [66]. The mere fact that that this group is operational is in itself a success. Furthermore, from statistical

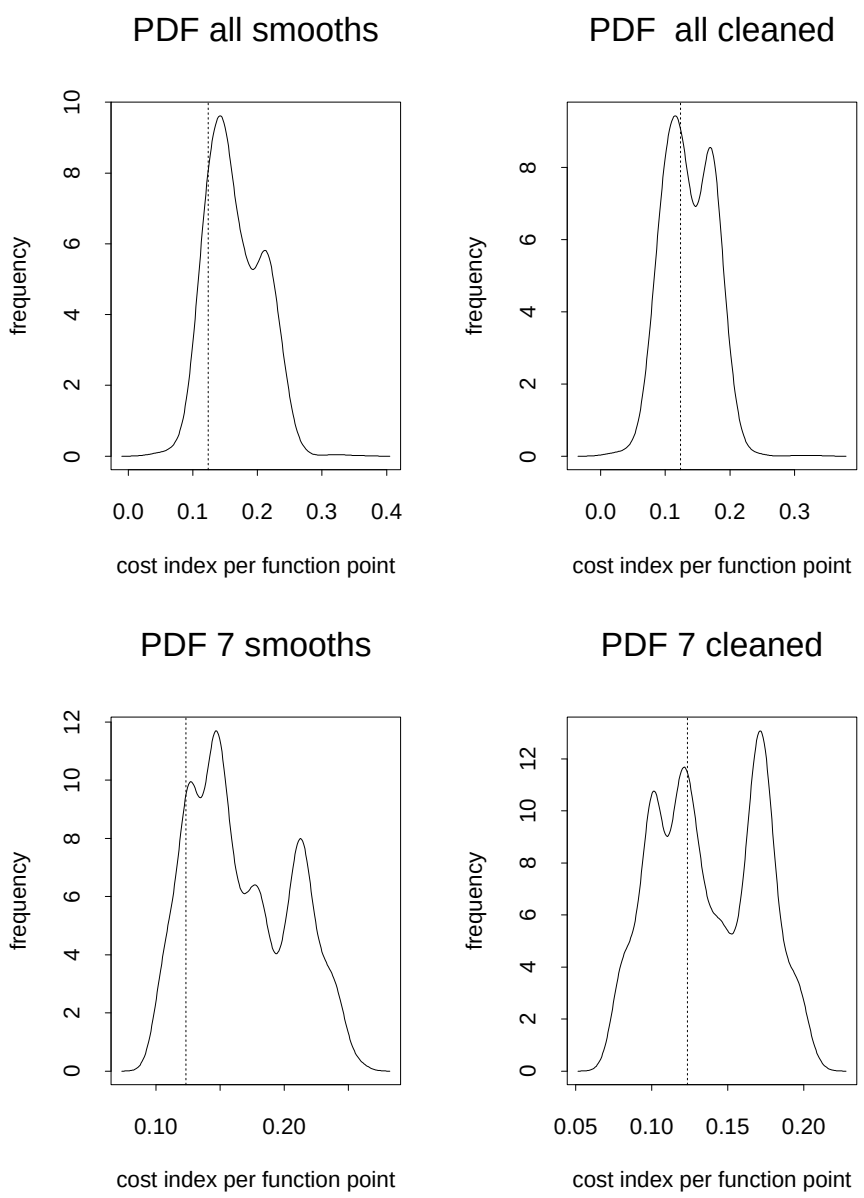

Figure 100: Probability density functions for the aggregates of all eight smooths, and their cleaned variants, plus the 7 smooths plus their cleaned version. The dashed line is the target line.

analyses we learned that for about 90% of the counted function points there is no evidence that the counting practices are different. In other words, not only is the metrics group operational, but the 15 FPA-analysts are delivering top notch quality as it comes to consistent counting. Also the counted function points are plausible: the function

point distribution is a heavy right-tailed one, which is a sign that solution underdelivery is not present. And the latter is not a common feature of IT-projects, of which we know that in 50% of the times only half the promised functionality is delivered (at twice the cost and duration) [37]. The heavy right tail in the function point size should also be present in derived indicators like duration, cost, and cost per function point. This is the case, as can be seen in Figure 101. Note that some of the left-tails dive below zero. Of course this cannot happen for function points, durations, and costs. The density estimates give us strong indications of the shape of the PDF, but not necessarily the exact PDF. These are found via a heavy tail analysis, which we have carried out in Section 5.

In addition, the function point totals and their corresponding durations and costs are correlated over time, so that not only the distributions show the same shape, but also the data over time clearly displays correlations. Since there is a trade-off between time and effort in IT-projects, and since all the variables are stochastic, this correspondence is not one-to-one. From statistical analyses we can reveal the correlations, including correlated varying variance (see Figures 72 and 76). So there is no sign of retrofitting budgets and durations from one IT-project to another just to make sure that the aggregates for higher management are met.

A very important finding is that with the research set we carried out 3 predictions that turned out to be valid. This is a sign that the data and the models are actually describing the real IT-situation. We predicted the trend in the cost index per function point, which would rise according to the statistical models. First we used an outlier-unaware model for the cost index per function points, which showed an increase of this variable, see Figure 42. Second, we carried out an outlier-aware analysis, and the found model also predicted a rise, as Figure 49 shows. Finally, based on the modelling of the trivariate system of function point sizes, durations and costs, we predicted those values, and calculated the prediction for the cost index per function points, which also indicated a rise, see Figure 82.

The final analysis did not only show the rise that was predicted three times, but it also showed the clear downward trend using smooth regression techniques. From the diagnostics summarized in Table 11, the super smooth is the one with the least structure in its residuals, so that one serves best as the trend indicator. In Figure 102, we depict this trend, with and without taking outliers into account.

It is clear that in both cases the trend is downwards: what is the minimal value in the beginning is the maximal value at the end. More numerical, from Table 12 we can see that the maximum value for the super smooth with outliers is 0.2126, without outliers 0.1757, and the minimal value is with outliers 0.0999, and without 0.0747. Given the fact that the target line index is 0.1235797, we conclude that the target is met. So according to the criterion imposed by management, the SPI-program delivered its value.

Apart from that, we learn a few things from Figure 102: first of all it is utterly important to understand whether the outliers are real, that is, expensive IT-projects, or that they are stray values, since these (in this case only 6) values strongly influence the vertical location of the trend. Namely, the trend taking the outliers into account lies much higher than the trend in the cleaned version. Another lesson learned is that we can now try to identify the possibly exogenous bumps and wiggles in the trend lines. They can be caused by reorganizations, mergers and acquisitions, (off-shore) outsourcing, moving from one place to another, and so on. This could provide us invaluable insights in a better understanding of exogenous influences on IT-productivity.

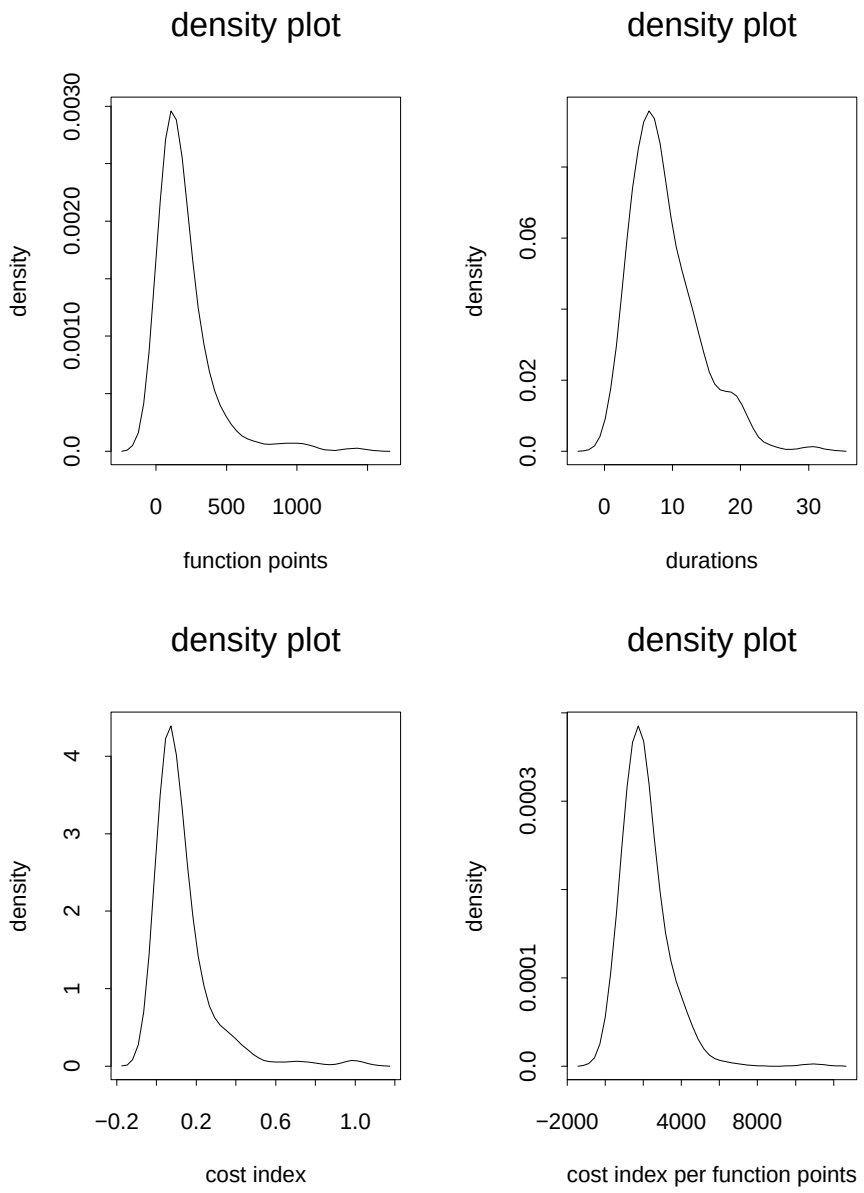

Figure 101: Probability density functions for function points, durations, cost, and cost per function points, all showing the same shape.

13 CONCLUSIONS

In this paper we accepted the fact that many important key performance indicators for IT-projects display stochastic effects, contain considerable noise, may vary over time, and that their variation over time may vary as well. In software engineering

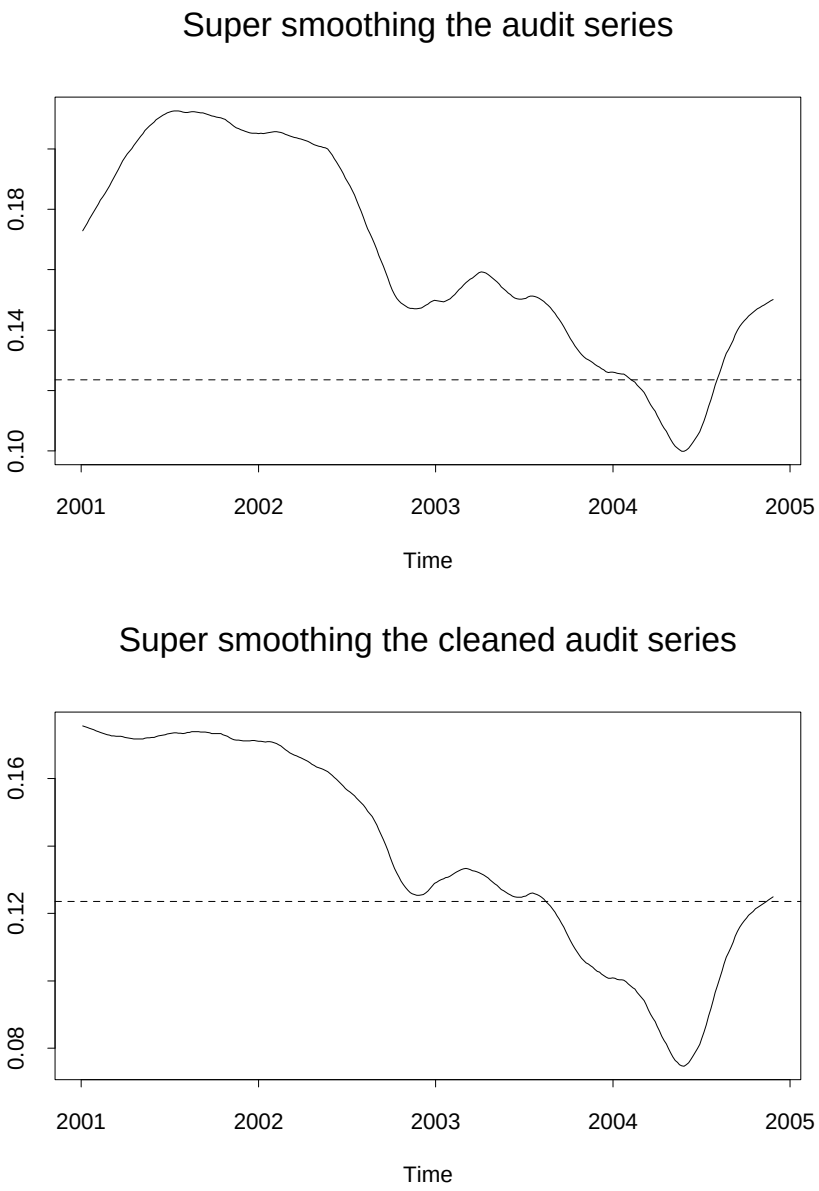

Figure 102: A clear downward trend for the key performance indicator cost index per function point.

terms this may be summarized as: hopeless. In mathematical terms, it turns out that many KPIs behave as a system of multivariate generalized autocorrelated conditional heteroscedastic processes (*MGARCH* processes), or less general time series structures. With rigorous mathematical and statistical techniques we showed how to come to grips with such seemingly hopeless KPIs and identified and quantified their properties. These

properties range from identification of their distribution family, to their co-movements in time with regard to their volatility. We fitted mathematical models to the KPIs, which turned out to be adequate for prediction purposes. We applied our approach to an extensive real-world problem: we quantified the effects of a large software process improvement program that was launched within a large organization involved in a large amount of business-critical software. We could successfully assess the effects of this improvement program, and revealed microscopic and macroscopic trends in the KPIs.

This approach is in no way limited to software process improvement. There is a rich application potential, as soon as you start collecting data. But we also have to realize that it will take considerable time before the software engineering discipline will be in a position to routinely benefit from our work. One reason is that the majority of the organizations is not measuring anything at all, another is that establishing a measurement discipline often fails. For organizations that are not capable of measuring important indicators, we refer to [85, 88, 87] where the author treats a way to start gaining control via benchmarked relations between important KPIs. For those organizations capable of collecting data, often the skills lack to apply the necessary tools and methods to analyse the data. The latter can be overcome: by hiring people skilled in insurance mathematics, actuarial science, biostatistics [23] and/or econometrics, in combination with keen software engineers with a feeling for numbers. We are confident that measurement, skilled analysts, and this paper brings us one step closer to exploring quantifiable information technology yields.

Acknowledgements This research has partially been sponsored by the Dutch Ministry of Economic Affairs via contract SENTER-TSIT3018 *CALCE: Computer-aided Life Cycle Enabling of Software Assets*. Furthermore, this research received partial support by the Dutch *Joint Academic and Commercial Quality Research & Development (Jacquard)* program on Software Engineering Research via contract 638.004.405 *EQUITY: Exploring Quantifiable Information Technology Yields*. We are grateful to our commercial EQUITY-partner and -sponsor ABN AMRO Bank N.V. for providing the data on which this paper is based. Especially, we like to thank the people from Group Audit and Business Unit Netherlands for stimulating discussions, insightful comments, challenging research questions, and deployment of the results of this research.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, AC-19(6):716–723, 1974.

[2] C. Alexander. *Market Models – A Guide to Financial Data Analysis*. Wiley, 2001.

[3] D. Bates et al. The R Project for Statistical Computing, 2004. `www.r-project.org`.

[4] E.K. Blum. *Numerical Analysis and Computation Theory and Practice*. Addison-Wesley, 1972.

[5] T. Bollerslev. Generalized Autoregressive Conditional Heteroskedasticity. *Journal of Econometrics*, 31:307–327, 1986.

[6] T. Bollerslev, R.F. Engle, and D.B. Nelson. ARCH Models. In R.F. Engle and D.L. McFadden, editors, *Handbook of Econometrics*, pages 2959–3038. Elsevier Science B.V., 1994. Available at: `www1.elsevier.com/hes/books/02/04/049/0204049.pdf`.

[7] B.M. Bouldin. *Agents of Change – Managing the Introduction of Automated Tools*. Yourdon-Press, 1989.

[8] G.E. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis – Forecasting and Control*. Prentice Hall, 3rd edition, 1994.

[9] U.S. Census Bureau. X-12-ARIMA Reference Manual, 2002. Available via: `www.census.gov/ts/`.

[10] R.A. Carmona. *Statistical Analysis of Financial Data in S-Plus*, volume XVI of *Springer Texts in Statistics*. Springer-Verlag, 2004.

[11] J.M. Chambers. *Programming with Data – A Guide to the S Language*. Springer Verlag, 1998.

[12] J.M. Chambers, W.S. Cleveland, B. Kleiner, and P.A. Tukey. *Graphical Methods for Data Analysis*. Wadsworth, 1983.

[13] J.M. Chambers and T.J. Hastie, editors. *Statistical Models in S*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1992.

[14] N.H. Chan. *Time Series – Applications to Finance*. John Wiley & sons, 2002.

[15] I. Chang, G.C. Tiao, and C. Chen. Estimation of time series parameters in the presence of outliers. *Technometrics*, 30:193–204, 1988.

[16] C. Chatfield. *Time-Series Forecasting*. Chapman & Hall, 2000.

[17] C. Chatfield. *The Analysis of Time Series – An Introduction*. Chapman & Hall, 6th edition, 2004.

[18] W.S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74:829–836, 1979.

[19] S. Coles. *An Introduction to Statistical Modeling of Extreme Values*. Springer Series in Statistics. Springer-Verlag, 2001.

[20] W.J. Conover. *Practical Nonparametric Statistics*. Probability and Mathematical Statistics. John Wiley & sons, 3rd edition, 1980.

[21] D. Dalcher and A. Genus. Introduction: Avoiding IS/IT Implementation Failure. *Technology Analysis and Strategic Management*, 15(4):403–407, December 2003.

[22] T. DeMarco. *Controlling Software Projects – Management Measurement & Estimation*. Yourdon Press Computing Series, 1982.

[23] P.J. Diggle. *Time Series – a Biostatistical Introduction*. Number 5 in Oxford Statistical Science Series. Oxford University Press, 1990.

[24] J. Durbin and S.J. Koopman. *Time Series Analysis by State Space Models*. Number 24 in Oxford Statistical Science Series. Oxford University Press, 2001.

[25] K. El-Emam and D.R. Goldenson. An Empirical Review of Software Process Assessments. In M.V. Zelkowitz, editor, *Advances in Computers – Emphasizing Distributed Systems*, volume 53, pages 319–423. Academic Press, 2000. Also available at `iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-43610.pdf`.

[26] P. Embrechts, C. Kluppelberg, and T. Mikosch. *Modelling Extremal Events – With a View Towards Insurance and Finance*, volume 33 of *Applications of Mathematics*. Springer-Verlag, 1997.

[27] R.F. Engle. Autoregressive conditional heteroskedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007, 1982.

[28] M.E. Fagan. Design and code inspections to reduce errors in programs. *IBM Systems Journal*, 15(3):182–211, 1976.

[29] M.E. Fagan. Advances in software inspections. *IEEE Transactions on Software Engineering*, SE-12(7):744–751, 1986.

[30] A.J. Fox. Outliers in time series. *Journal of the Royal Statistical Society*, 34:350–363, 1972.

[31] J.H. Friedman. A variable span smoother. Technical Report 5, Laboratory for Computational Statistics, Dept. of Statistics, Stanford University, Stanford, CA, USA, 1984.

[32] W.A Fuller. *Introduction to Statistical Time Series*. John Wiley & sons, 2nd edition, 1996.

[33] T. Gilb and D. Graham. *Software Inspection*. Addison-Wesley, 1993.

[34] R.L. Glass. *Computing Calamities – Lessons Learned From Products, Projects, and Companies that Failed*. Prentice Hall, 1998.

[35] R.L. Glass. *Software Runaways – Lessons Learned from Massive Software Project Failures*. Prentice Hall, 1998.

[36] R.L. Glass. *ComputingFailure.com – War Stories from the Electronic Revolution*. Prentice Hall, 2001.

[37] The Standish Group. CHAOS, 1995. Retrievable via: standishgroup.com/visitor/chaos.htm (Current February 2001).

[38] The Standish Group. CHAOS: A Recipe for Success, 1999. Retrievable via: www.pm2go.com/sample_research/chaos1998.pdf.

[39] The Standish Group. EXTREME CHAOS, 2001. Purchase via: https://secure.standishgroup.com/reports/reports.php.

[40] J.D. Hamilton. *Time Series Analysis*. Princeton University Press, 1994.

[41] R.W. Hamming. *Numerical Methods for Scientists and Engineers*. McGraw-Hill, 2nd edition, 1973.

[42] F. Harmsen and J. Kleijnen. Introduction of DSDM – ABN AMRO improves productivity, duration, and customer satisfaction. *Informatie*, 3:46–51, 2003. In Dutch.

[43] A.C. Harvey. *Forecasting, structural time series models and the Kalman filter*. Cambridge University Press, 1989.

[44] T.J. Hastie and R.J. Tibshirani. *Generalized Additive Models*. Chapman & Hall, 1990.

[45] J. Henry, A. Rossman, and J. Snyder. Quantitative Evaluation of Software Process Improvement. *Journal of Systems and Software*, 28(2):169–177, 1995.

[46] J. Herbsleb, A. Carleton, J. Rozum, J. Siegel, and D. Zubrow. Benefits of CMM-Based Software Process Improvement – Initial Results. Technical Report CMU/SEI-94-TR-013, Software Engineering Institute, 1994.

[47] B.M. Hill. A Simple General Approach to Inference about the Tail of a Distribution. *The Annals of Statistics*, 3:1163–1174, 1975.

[48] R.V. Hogg, J.W. McKean, and A.T. Graig. *Introduction to Mathematical Statistics*. Prentice Hall, 6th edition, 2005.

[49] S. Huet and M.-A. Gruet. *Statistical Tools for Nonlinear Regression – A Practical Guide with S-Plus Examples*. Springer Verlag, 1996.

[50] F. Hundertwasser. Verschimmelungsmanifest gegen den Rationalismus in der Architektur, 1958. Rede in der Benediktinerabtei Seckau am 4ten Juli 1958. Seckau, Österreich.

[51] J. Johnson. Chaos: The dollar drain of IT project failures. *Application Development Trends*, 2(1):41–47, 1995.

[52] C. Jones. *Software Assessments, Benchmarks, and Best Practices*. Information Technology Series. Addison-Wesley, 2000.

[53] C. Jones. Why software costs vary, 2003. Unpublished manuscript. Available via x@cs.vu.nl.

[54] K.R. Kadiyala and S. Karlsson. Forecasting with generalized Bayesian vector autoregressions. *Journal of Forecasting*, 12:365–378, 1993.

[55] R.S. Kaplan and D.P. Norton. *The Balanced Scorecard – Translating Strategy into Action*. Harvard Business School Press, 1996.

[56] C.F. Kemerer. Reliability of Function Points Measurement – A Field Experiment. *Communications of the ACM*, 36(2):85–97, 1993.

[57] C.F. Kemerer and B.S. Porter. Improving the Reliability of Function Point Measurement: An Empirical Study. *IEEE Transactions on Software Engineering*, SE-18(11):1011–1024, 1992.

[58] A.N. Kolmogorov. Sulla Determinazione Empirica di una Legge di Distribuzione. *Giornale dell' Istituto Italiano degli Attuari*, 4:83–91, 1933.

[59] A. Krause and M. Olson. *Basics of S and S-Plus*. Springer Verlag, 2nd edition, 2000.

[60] T. Lister. Becoming a Better Estimator – An Introduction to Using the EQF Metric, 2002. Available via www.stickyminds.com.

[61] D.R. McNeil. *Interactive Data Analysis – A Practical Primer*. Wiley and Sons, 1977.

[62] F. Mosteller and J.W. Tukey. *Data Reduction and Regression*. Addison-Wesley, 1977.

[63] D.B. Nelson. Conditional Heteroskedasticity in Asset Returns: a New Approach. *Econometrica*, 59(2):347–370, 1991.

[64] M.C. Paulk, C.V. Weber, B. Curtis, and M.B. Chrissis. *The Capability Maturity Model: Guidelines for Improving the Software Process*. Addison-Wesley Publishing Company, Reading, MA, 1995.

[65] J.C. Pinheiro and D.M. Bates. *Mixed-Effects Models in S and S-PLUS*. Springer Verlag, 2000.

[66] D.R. Pitts. Metrics: Problem Solved? *Crosstalk*, 1997. Available via: www.stsc.hill.af.mil/crosstalk/1997/dec/metrics.asp.

[67] L.H. Putnam and W. Myers. *Measures for Excellence – Reliable Software on Time, Within Budget*. Yourdon Press Computing Series, 1992.

[68] L.H. Putnam and D.T. Putnam. A Data verification of the Software Fourth Power Trade-Off Law. In *Proceedings of the International Society of Parametric Analysts – Sixth Annual Conference*, volume III(I), pages 443–471, 1984.

[69] D.F. Rico. Using Cost Benefit Analyses to Develop Software Process Improvement (SPI) Strategies. A DACS State-of-the-Art Report, Air Force Research Laboratory – Information Directorate, 2000.

[70] J.C. Robertson and E.W. Tallman. Vector Autoregressions: Forecasting and Reality. *Economic Review*, 0:4–18, 1999. Federal Reserve Bank of Atlanta.

[71] E.M. Rogers. *Diffusion of Innovations*. Free Press, 4th edition, 1995.

[72] Y. Sakamoto, M. Ishiguro, and Kitagawa G. *Akaike Information Criterion Statistics*. D. Reidel Publishing Company, 1986.

[73] M.G. Schimek. *Smoothing and Regression – Approaches, Computation, and Application*. Wiley, 1999.

[74] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall, 1986.

[75] C.A. Sims and T. Zha. Bayesian methods for dynamic multivariate models. *International Economic Review*, 39(4):949–968, 1998.

[76] N.V. Smirnov. Sur les écarts de la courbe de distribution empirique. *Matematicheskij Sbornik. (Novaya Seriya)*, 6:3–26, 1939. Russian/French summary.

[77] J. Spencer. On the graduation of the rates of sickness and mortality. *Journal of the Institute of Actuaries*, 38:334–347, 1904.

[78] J. Stapleton. *DSDM – Business Focused Development*. Addison-Wesley, 2nd edition, 2003.

[79] J. Stapleton and P. Constable. *DSDM – Dynamic System Development Method*. Addison-Wesley, 1997.

[80] H. Tianfield. An autonomic framework for quantitative software process improvement. In G. Fodor, M. Ulieru, and A. Weaver, editors, *Proceedings of the 1st IEEE International Conference on Industrial Informatics (INDIN 2003)*, pages 446–450. IEEE Xplore, 2003. Available at: `eeexplore.ieee.org:80/xpl/tocresult.jsp?isNumber=28887`.

[81] R.S. Tsay. Outliers, Level Shifts, and Variance Changes in Time Series. *Journal of Forecasting*, 7:1–20, 1988.

[82] J.W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[83] P.F. Velleman and D.C. Hoaglin. *Applications, Basics, and Computing of Exploratory Data Analysis*. Duxbury Press, 1981.

[84] W.N. Venables and B.D. Ripley. *Modern Applied Statistics with S-PLUS*. Springer Verlag, 3rd edition, 1999.

[85] C. Verhoef. Quantitative IT Portfolio Management. *Science of Computer Programming*, 45(1):1–96, 2002. Available via: `www.cs.vu.nl/~x/ipm/ipm.pdf`.

[86] C. Verhoef. Quantifying the Effects of IT-governance Rules, 2004. Available via `www.cs.vu.nl/~x/gov/gov.pdf`.

[87] C. Verhoef. Quantifying the Value of IT-investments. *Science of Computer Programming*, 2004. To Appear. Available via: `www.cs.vu.nl/~x/val/val.pdf`.

[88] C. Verhoef. Quantitative Aspects of Outsourcing Deals. *Science of Computer Programming*, 2004. To Appear. Available via: `www.cs.vu.nl/~x/out/out.pdf`.

[89] G. Walker. On Periodicity in Series of Related Terms. *Proceedings of the Royal Society of London*, A 131:518–532, 1931.

[90] G.S. Watson. Smooth regression analysis. *Sankhy$\overline{a}$ – The Indian Journal Of Statistics*, Series A 26 part 4:359–372, 1964.

[91] E.J. Wegman. Nonparametric probability density estimation. *Technometrics*, 14:533–546, 1972.

[92] G.U. Yule. On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers. *Philosophical Transactions of The Royal Society*, A 226:267–298, 1927.

[93] T. Zha. A Dynamic Multivariate Model for Use in Formulating Policy. *Economic Review*, 1:16–29, 1998. Federal Reserve Bank of Atlanta.

[94] E. Zivot and J. Wang. *Modeling Financial Time Series with S-Plus*. Springer-Verlag, 2002.

[95] C. de Zwart. Bank to a Higher Level – ABN AMRO implements DSDM to make software development more effective. *Computable*, 14:p. 16, 2003. In Dutch.