Strong Agile Metrics

Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams

Huijgens, Hennie; Lamping, Robert; Stevens, Dick; Rothengatter, Hartger; Gousios, Georgios; Romano, Daniele

DOI
10.1145/3106237.3117779

Publication date
2017

Document Version
Accepted author manuscript

Published in
ESEC/FSE 2017

Delft University of Technology
Software Engineering Research Group
Technical Report Series

Strong Agile Metrics

Hennie Huijgens, Robert Lamping, Dick Stevens, Hartger Rothengatter, Daniele Romano and Georgios Gousios

Report TUD-SERG-2017-010

TUD-SERG-2017-010

Published, produced and distributed by:

Software Engineering Research Group
Department of Software Technology
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4
2628 CD Delft
The Netherlands

ISSN 1872-5392

Software Engineering Research Group Technical Reports:
http://www.se.ewi.tudelft.nl/techreports/

For more information about the Software Engineering Research Group:
http://www.se.ewi.tudelft.nl/

Note: Accepted for publication in the Industry Track of the *ACM Proceedings of the 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, held in Paderborn, Germany, September 2017 (ESEC/FSE17). https://doi.org/10.1145/3106237.3117779

Strong Agile Metrics: Mining Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams

Hennie Huijgens
Delft University of Technology
Delft, The Netherlands
h.k.m.huijgens@tudelft.nl

Robert Lamping
ING Bank and CGI
Amsterdam, The Netherlands
robert.lamping@ing.com

Dick Stevens
ING Bank
Amsterdam, The Netherlands
dick.stevens@ing.com

Hartger Rothengatter
ING Bank
Amsterdam, The Netherlands
hartger.rothengatter@ing.com

Daniele Romano
ING Bank
Amsterdam, The Netherlands
daniele.romano@ing.com

Georgios Gousios
Delft University of Technology
Delft, The Netherlands
g.gousios@tudelft.nl

ABSTRACT

ING Bank, a large Netherlands-based internationally operating bank, implemented a fully automated continuous delivery pipeline for its software engineering activities in more than 300 teams, that perform more than 2500 deployments to production each month on more than 750 different applications. Our objective is to examine how strong metrics for agile (Scrum) DevOps teams can be set in an iterative fashion. We perform an exploratory case study that focuses on the classification based on predictive power of software metrics, in which we analyze log data derived from two initial sources within this pipeline. We analyzed a subset of 16 metrics from 59 squads. We identified two lagging metrics and assessed four leading metrics to be strong.

CCS CONCEPTS

• **General and reference** → Cross-computing tools and techniques → Metrics

KEYWORDS

Software Economics; Agile Metrics, Scrum, Continuous Delivery, Prediction Modelling, DevOps, Data Mining, Software Analytics.

ACM Reference format:

Hennie Huijgens, Robert Lamping, Dick Stevens, Hartger Rothengatter, Daniele Romano and Georgios Gousios. 2017. Strong Agile Metrics: Mining of Log Data to Determine Predictive Power of Software Metrics for Continuous Delivery Teams. In *ACM Proceedings of the 11th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 2017 (ESEC/FSE'17)*, 6 pages.
https://doi.org/10.1145/3106237.3117779

1 INTRODUCTION

In order to further speed up application deployments, reduce risks of failure, and deliver applications rapid, repeatable, and reliable, ING Bank, a large Netherlands-based internationally operating bank, introduced Continuous Delivery as a Service (CDaaS) and DevOps. ING's continuous delivery cycle includes code, build, deploy, test and release of all software engineering activities, and supports more than 300 software delivery teams - *squads* in ING terminology - that operate primarily on a Linux or a Windows platform. The mindset behind CDaaS is to go to production as fast as possible while maintaining or improving quality, so teams get fast feedback, and know they are on the right track. The continuous delivery pipeline is at the core of a transition that is ongoing within ING towards DevOps.

ING's continuous delivery cycle is now implemented. Two matured CDaaS squads support a huge variety of squads in different business domains and software technologies. And all teams work in an agile (Scrum) way. Yet, now a need is felt to develop a monitor and control capability that fits the different squads in the organization. In this process of setting up a software metrics approach, the company wants to prevent from looking at metrics in isolation, treating a metric (making alterations just to improve a metric), or one trick metrics on the one hand against metrics galore on the other [1]. In line with the fully automated building, testing, and deployment of software in the continuous delivery pipeline three important requirements are applicable. (1) The monitor and control capability is implemented as a fully automated and iterative process, (2) it supports the different squads where possible in improving their deliveries, and (3) the capability must have a high degree of predictive power.

In this paper we describe the initial process to determine strong software metrics - being metrics with high predictive power - in order to be able to support a highly effective monitor and control capability within ING Bank. Because the process to classify metrics will be iterative – adding new metrics to the procedure will lead to a fluid and ongoing redefinition of the concept of 'strong' – we use the terminology of strong *agile* metrics. We perform our

analysis as an exploratory case study, based on two initial data sources from ING's software engineering domain: the Backlog Management (BLM) discipline, and the Continuous Delivery (CDaaS) discipline. Although data is collected on a per squad level, strong metrics are analyzed and reported at an aggregated (average) level and not as such on a per squad level.
Our objectives are to explore whether data mining techniques can help to define such strong agile metrics. In a follow-up case study based at this exploratory study we aim to develop a 1-to-5 star-rating for software metrics, that can be used throughout ING's software organization as a support tool when preparing dashboards and other visualizations. The aspect of automation and building a performance dashboard itself is not within the scope of this paper. We address the following research question:

RQ *How can we set strong metrics for agile (Scrum) DevOps teams in an iterative fashion?*

As our key contribution we evaluate how to determine the predictive power of software metrics from log data of a toolset of more than 300 different development teams in a large software company, that perform more than 2500 deployments to production each month on more than 750 different applications.

1.1 Background

ING Bank implemented Continuous Delivery based on the model described by Humble and Farley [2], where the CD pipeline is proposed as a deployment pipeline for the whole value stream of software development. ING Bank set up two different pipelines based on the main technologies Linux and Windows. Its' main goal is to support squads in maximizing the benefits of shared use of tools.
In this paper we focus at the CDaaS Linux pipeline. It provides developers with a complete set of standard tools that are supported by a Linux CDaaS squad and available to all squads within the bank. The pipeline fully automates the software release process for Linux based applications. It contains several specialized tools which are connected to each other, such as GitLab (code), Jenkins, SonarQube, and QWasp and Artifactory (build), Nolio (deploy), iTested (test), and iValidate (release).
The final goal of our analysis (yet, out-of-scope for this exploratory study) is to examine 'good' deliveries (being better than average within the scope of a squad) and 'bad' deliveries (being worse than average), as described in previous work on success and failure in software engineering projects [3]. By doing so we expect to identify success factors that help squads to create better deliveries in future releases, and failure factors that help squads to prevent from 'bad' deliveries. However, due to the limited number of initial data sets this final goal is out-of-scope for this paper.

2 RESEARCH APPROACH

We conduct our research as an exploratory case study and we assess the predictive power of software delivery metrics, in which we use data derived from two initial sources. The first data source is log data from ServiceNow, the Backlog Management (BLM) tool that is used by most of the software development squads. The second data source that is used in this study is deployment log data from Nolio.
For our exploratory study we define a limited set of software metrics focused at a delivery scope (e.g. epics, user stories) as a minimum viable product (MVP): a product with just enough features to gather validated learning about the product and its continued development [4]. Based on the initial two data sets we designed four steps to define the MVP: (1) Scope definition, (2) Collect metrics, (3) Analyze for correlations, and (4) Determine prediction power of metrics.

2.1 Scope Definition

In this initial step we define the scope of the software metrics to be analyzed. Within the scope of this exploratory case study we limit the scope of data to log data of two data sources, derived from ServiceNow and Nolio. We combine a snapshot of ServiceNow data from 370 squads with CDaaS data of the deployment file from 101 squads. No time series are included in both datasets. We focus our analysis on squads; teams that deliver sets of user stories, combined in epics, to users within ING Bank or to ING's customers. Due to various missing data in both datasets, linking both datasets result in 59 squads for which all data are present. This is our analysis set.

2.2 Define and Collect Metrics

In a number of meetings with stakeholders definitions of metrics and hypothesis are classified. A draft metrics framework, an inventory of existing dashboards within the company, and an informal literature review are used as a baseline. Our intention is not to come up with a finalized inventory, but instead to set up and maintain a backlog of prioritized software metrics. In order to set up a repeatable future-proof procedure that is fully automated in-the-end, we do not analyze raw data itself. Instead we structure the log data upfront in a dedicated data warehouse with an automated feed from BLM and CDaaS in the background. Within the scope of this study a limited set of one daily download is used for further analysis. In this initial data set observations with missing values are beforehand removed.

2.3 Lagging and Leading Metrics

In this paper we distinguish two types of metrics: lagging and leading metrics. Lagging metrics are output oriented and cannot be directly influenced. A lagging indicator gives a signal *after* the trend or reversal occurs. These metrics are perceived as key indicators for high performing teams. Leading metrics are input oriented and easy to influence. A leading indicator gives a signal *before* the trend or reversal occurs [5].

Table 1. Metrics Descriptions and Descriptive Statistics.

Metric*	Source	n	Type	Skewness	Kurtosis	Min	1st Q	Median	Mean	3rd Q	Max
cdaas_cycletime_tst1_prd [Lagging]	CDaaS	59	Days	0.56	-0.19	0.41	11.21	17.22	17.96	25.18	46.52
cdaas_mtb_prd_lst90days	CDaaS	59	Days	0.73	0.96	2.98	23.02	34.16	36.77	49.22	104.90
cdaas_numberofdeploymentsprd	CDaaS	59	Number	4.12	18.77	1.00	3.00	7.00	16.73	17.00	194.00
sprint_averageleadtime	BLM	59	Days	5.19	45.31	-6.50	22.15	31.17	37.65	44.25	370.50
sprint_averagepointsperstory	BLM	59	Ratio	3.37	13.34	0.03	0.08	0.11	0.17	0.17	1.00
sprint_duration	BLM	59	Days	3.14	21.08	5.00	13.00	13.00	14.79	14.00	55.00
sprint_numberofchangemembers	BLM	59	Number	3.76	30.14	0.00	7.00	9.00	9.54	11.00	51.00
sprint_numberofepicslastsprint	BLM	59	Number	1.51	3.77	1.00	7.00	11.00	12.00	16.00	48.00
sprint_numberofsquadmembers	BLM	59	Number	3.56	26.93	2.00	7.00	10.00	10.01	12.00	51.00
sprint_plannedpointscompletionratio [Lagging]	BLM	59	Ratio	-0.62	-0.23	0.00	0.05	0.67	0.64	0.83	1.00
sprint_plannedstoriescompletionratio [Lagging]	BLM	59	Ratio	-0.53	-0.55	0.00	0.44	0.67	0.63	0.86	1.00
sprint_pointscompletionratio	BLM	59	Ratio	0.29	1.75	0.02	0.50	0.72	0.70	0.90	2.18
sprint_remainingtimeratio	BLM	59	Ratio	5.28	29.28	0.00	0.00	0.00	0.03	0.00	1.00
sprint_scopegrowth	BLM	59	Number	16.65	283.94	-112.00	0.00	0.00	2.82	0.00	919.00
sprint_unplannedexistingpointscompletionratio	BLM	59	Ratio	1.53	1.85	0.00	0.00	0.07	0.17	0.27	1.00
sprint_unplannednewpointscompletionratio	BLM	59	Ratio	3.95	18.55	0.00	0.00	0.00	0.07	0.07	1.00

Backlog Management (BLM) log data from ServiceNow and CDaaS log data from Nolio. *A more detailed description of each metric, including extended descriptive statistics is included in the Technical Report. In order to assess distribution we included Skewness and Kurtosis of each individual metric. Lagging metrics are indicated with the text [Lagging] behind their names in the first column of the table above.

2.4 Analysis

We define and collect metrics from two initial data sources, classify metrics, and specify whether metrics are lagging or leading. we examine descriptive statistics and we analyze the total set of metrics from the two initial data sources for statistical correlation. To understand any relations between individual metrics we perform linear regression. For visualization purposes we prepare a correlation matrix that plots positive and negative correlations between all individual metrics; this matrix is not included in this paper, it is to be found in a Technical Report [6].

2.5 Determining Predictive Power of Metrics

To classify software metrics based on their prediction strength with regard to the performance of releases delivered by agile (Scrum) DevOps teams, we use a search algorithm to find the best model, based on forward selection, backward elimination, and stepwise regression. We define strong metrics as leading metrics with strong correlation(s) to lagging metrics in the data set. Although in our exploratory case study we do this in a manual way, we plan for automated methods to identify predictor variables in a future solution. Automated methods are useful when the number of explanatory variables is large, as in our case, and when it is not feasible to fit all possible models. For this purpose we built a new model in R, based on the existing Corrplot package, in which we visualize the outcomes of the multiple linear regression in a Leading Lagging Matrix (see Figure 1).

3 RESULTS

In Table 1 we inventory descriptive statistics of the BLM metrics and CDaaS metrics in scope. To test whether the data in our datasets is normally distributed or not, we used a skewness and kurtosis test. As values for skewness and kurtosis between -2 and +2 are considered acceptable in order to prove normal univariate distribution [7], we assess the majority of metrics in both subsets to be not normally distributed.

3.1 Analysis of Predicting Variables

We perform pairwise correlation in order to find any relations between individual metrics. Because a small majority of the metrics are assessed to be not normally distributed we use the method Spearman. A visualization in the form of a correlation matrix is included in the Technical Report [6]. Variables that have no significant correlation in a 1-to-1 analysis, may act differently in multiple regression. Remaining time ratio is an example of such a variable.

We are modelling lagging indicators in terms of leading variables. In Figure 1 the impact of leading metrics on the set of lagging metrics is visualized in a *Leading Lagging Matrix*. The figure shows for each lagging variable (horizontal axis) what the impact is of each leading variable (vertical axis). The color of each circle indicates whether the impact is positive (blue) or negative (red). The size of a circle indicates whether impact is strong (large impact) or weak (small impact). Same size circles on the same row do not mean they are equal: for each lagging variable the leading variables are calculated using multiple linear regression, subsequently all multiple linear regression coefficients are rescaled to a

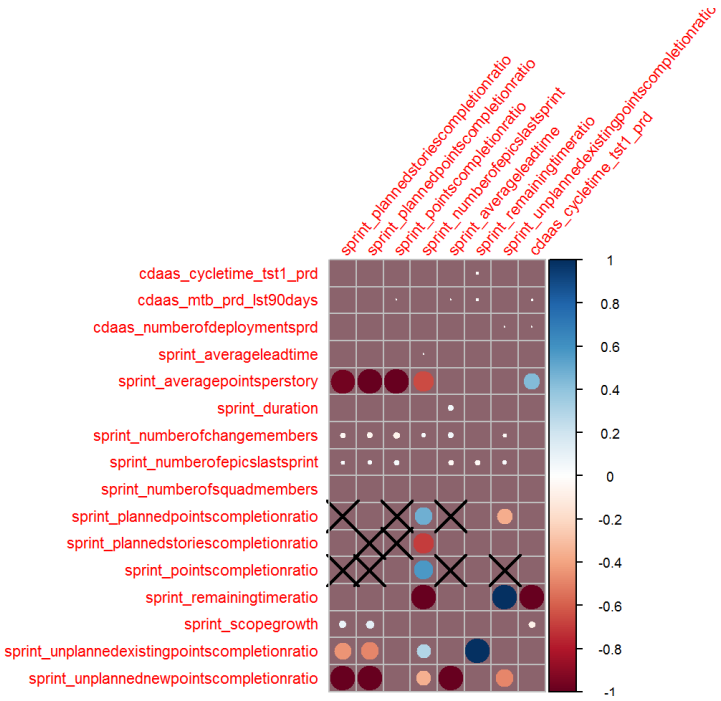


Figure 1: *Leading Lagging Matrix* showing the impact of leading metrics (vertical axis) on lagging metrics (horizontal axis).

scale of -1 to 1. Empty squares indicate a coefficient of zero. Crossed out variables are excluded from the lagging model in order to avoid collinearity (independent variables that are highly correlated).

In this first exploratory study we used a pragmatic approach to determine which variables are leading or lagging in our model. We argued that in this first analysis three metrics are assessed to be lagging. (1) *Planned stories completion ratio*; the number of planned stories that were completed in a sprint divided by the number of planned stories. (2) *Planned points completion ratio*; the number of completed planned story points divided by the number of planned story points. (3) *CDaaS cycle time*; the mean time from first test deployment after last production has been done until the next production deployment for all applications of a squad.

The choice for these three lagging metrics is mainly driven by the assumption that they are typically output related and cannot easily be planned upfront. For analysis purposes we included a reference set of five other metrics on the x-axis of the matrix, although these were not assumed to be lagging.

3.4 Key Findings

When examining the *Leading Lagging Matrix* as depicted in Figure 1, we observe the following:

Observation 1. Higher average story points (5th row) have a negative impact on the planned completion ratio (either points or stories) and on the total completion ratio. Besides that, higher average story points lead to a longer CDaaS cycle time (from test to production).

Observation 2. If the planned points completion ratio (10th row) goes up, also the number of epics increases. At the same time unplanned backlog work (unplanned existing) decreases.

Observation 3. The planned stories completion ratio (11th row) shows an opposite effect: if this variable goes up, the number of epics decreases. The different behavior of both metrics need to be examined further in follow-up research.

Observation 4. If the points completion ratio (12th row) increases, also the number of epics goes up, an effect that is similar to Observation 2.

Observation 5. If the remaining time ratio (13th row) goes up, the number of epics and the CDaaS cycle time decreases. Furthermore, if there is time left after all planned work is done, we see that squads pick up backlog work.

Observation 6. When unplanned existing work (15th row) (e.g. picked up from backlog) pops up this has a moderate negative impact on the planned work and the ability to pick up backlog work. In this case the number of epics increases moderately, while remaining time ratio increases strongly.

Observation 7. Average lead time is negatively impacted by unplanned new work (e.g. incidents) (bottom row), possibly because it delays planned work that was already started in the current sprint or earlier and that cannot finish in the planned time. Average lead time is not tightly linked to a sprint, but more to a user story.
Observation 8. No significant impact is caused in our models by the three CDaaS variables, and the remaining BLM variables.

4 DISCUSSION

As explained in paragraph 3.3 we assumed upfront three metrics to be lagging: planned stories completion ratio, planned points completion ratio, and CDaaS cycle time. The first two metrics are both about the completion of planned work, so to say the predictability of delivery of squads. We prefer planned stories completion ratio because it has the advantage that scope growth (measured in story points) has no influence on the ratio value. Leading variables for this metric are unplanned new points completion ratio, unplanned existing points completion ratio, and average points per story. With regard to this lagging metric we assess these three leading variables as 'strong metrics'.
The third lagging variable, CDaaS cycle time, can be influenced by the leading variables remaining time ratio and average points per story. With regard to this lagging metric we assess both leading variables as 'strong metrics'.
Besides these three variables we observe that also the number of epics (4th column from the left) can be influenced strongly by planned points completion ratio, points completion ratio, remaining time ratio, unplanned existing points completion ratio, and unplanned new points completion ratio. However, because we assume that this variable can be easily planned upfront we do not assess it as a lagging variable as such.
From an overall point of view we argue that average points per story is influencing both preferred lagging metrics, and due to that is to be assessed as the most powerful metric in the actual subset.

4.1 Implications

Our model, based on an initial subset of BLM and CDaaS data, indicates that squads can improve their planned stories completion ratio and reduce their CDaaS cycle time by slicing their deliverables in smaller user stories. Squads can reduce their CDaaS cycle time by keeping open space in their sprint planning (e.g. increasing their remaining time ratio). Finally, by reducing unexpected unplanned work squads can increase their predictability of delivery (e.g. planned stories completion ratio). These implications are also identified by Humble and Farley [2] as key measures for implementing Continuous Delivery and DevOps. Our research herewith substantiated these measures based on statistical analysis of ING's Continuous Delivery cycles.

4.2 Threats to Validity

With regard to construct validity we are aware that the use of story points for comparison purposes over squads might be spurious in a way. However, to prevent from differences in ranges used by different squads we calculated all measurements with story points involved to indexes. A second threat that we take into account is the way we picked metrics to be included in the study and the choice of lagging versus leading metrics. In our actual approach we inventoried metrics in related work and existing dashboards within ING bank, and mapped these on a metrics framework that we based on previous work [8]. We realize that some systemic bias might play a role here and are looking for ways to mitigate this in a more mature approach. A third threat with regard to construct validity is that we did not exclude outliers from our research dataset. Although we realize that this is important for a future approach we did not include this in this exploratory study. Finally we recognize challenges with data quality as a threat to validity. Especially to link BLM data on squads and applications with the CDaaS dataset was a blockade in some cases.
A threat to internal validity that we acknowledge is the fact that 'fishing for p -values' might hold a risk that some of the correlations we find are a coincidence. Although we acknowledge the fact that in a future approach corrections (e.g. Benjamini–Hochberg [9]) are to be implemented, we did not apply any of such corrections in this initial and exploratory study.
Concerning external validity we argue that results from our study are not to be generalized to other companies than ING Bank. We assume that different companies have their own specific metrics patterns. We expect that our approach to derive strong metrics by mining log data from software delivery pipeline tools can be successfully used in practice by other companies too. To encourage reuse and further improvement of our approach, we share a subset of the R-code developed by us in the Technical Report [6].

5 RELATED WORK

Where from the 80s onwards software companies used to follow a software process improvement (SPI) approach with varying success [10] [11], since the start of this millennium an industry-wide transformation towards agile development methods is obvious. Although several studies are performed on the success and failure of agile methods [12] [13] [3], a clear definition of success is difficult to find. A number of researchers and practitioners come up with terms such as hyperproductivity [14] [15] [16], usually with a limited focus on best practices in a Scrum environment, described as for example 'the most productive Java projects ever documented' [14]. To define definitions for hyperproductivity and accompanying software metrics we based our research besides the above on metrics for continuous delivery as mentioned by Humble and Farley [2] and on Puppet's State of DevOps Report [5].
The effect on strong positive correlations between *Project Size*, *Project Cost*, and *Number of Defects* is known from related work [3] [17] [18]. Also the effect of project size as a risk factor is described earlier. Smaller projects tend to have lower cancellation rates [19]. Smaller projects tend to perform better in terms of quality, being on budget, and being on schedule [19] [20]. Project size is found to be an important risk factor for success [21] [22] [23].

In previous work we found strong effects by comparing quantitative metrics such as cost, duration, number of defects, and size with qualitative metrics like stakeholder satisfaction and perceived value [8]. A recent guest editorial by Mäntylä et al. [24] mentions that, although "many papers investigate success and failure of software projects from diverse perspectives, leading to a myriad of antecedents, causes, correlates, factors and predictors of success and failure", a solid, empirically grounded body of evidence enabling actionable practices for increasing success and avoiding failure in software projects is not yet found.
Premrai et al. [25] investigated how software project productivity had changed over time, finding that an improving trend was measured, however less marked since 1990. The trend varied over companies and business sectors, a finding that matches the result of our previous research with regard to differentiation over business domains [3].

6 CONCLUSIONS AND FUTURE WORK

We analyzed a dataset built from BLM and CDaaS data from ING Bank in order to identify strong metrics; metrics with high predictive power towards a subset of lagging variables. We found two lagging metrics and four leading metrics that are assessed to be strong.
In future research we plan to extend the number of data sources - e.g. availability, squad decomposition, business process performance, customer experience, incidents - and due to that the number of variables in our model. We also plan to examine how lagging metrics and strong leading metrics can be identified in an automated procedure. Furthermore we intend to automatically set targets on the strong agile metrics, based on the performance of high performing teams within ING Bank. Our final objective is to use our findings to define relevant lagging metrics and related strong leading metrics to enable squads and management to steer on performance by delivering strong agile metric dashboards.

ACKNOWLEDGMENTS

Our sincere thanks to ING Bank for offering us the opportunity and the confidence to perform research in their challenging software development team environment.

REFERENCES

[1] E. Bouwers, J. Visser en A. van Deursen, „Getting What You Measure," *Communications of the ACM*, vol. 55, nr. 7, pp. 54-59, 2012.

[2] J. Humble en D. Farley, Continuous Delivery, reliable software releases through build, test and deployment automation, Addison-Wesley, 2010.

[3] H. Huijgens, R. van Solingen en A. van Deursen, „How to build a good practice software project portfolio?," in *ACM Companion Proceedings of the 36th International Conference on Software Engineering (ICSE)*, 2014.

[4] J. Münch, F. Fagerholm, P. Johnson, J. Pirttilahti, J. Torkkel en J. Jäarvinen, „Creating minimum viable products in industry-academia collaborations," in *Lean Enterprise Software and Systems*, Springer Berlin Heidelberg, 2013, pp. 137-151.

[5] „State of DevOps Report," Puppet, 2016.

[6] H. Huijgens, R. Lamping, D. Stevens, H. Rothengatter en G. Gousios, „Strong Agile Metrics - Technical Report TUD-SERG-2017-010," Delft University of Technology, 2017.

[7] D. George en M. Mallery, SPSS for Windows Step by Step: A Simple Guide and Reference, 17.0 update (10a ed.) red., Boston: Pearson., 2010.

[8] H. Huijgens, A. van Deursen en R. van Solingen, „The Effects of Perceived Value and Stakeholder Satisfaction on Software Project Impact," *Information and Software Technology*, 2017.

[9] W. Hopkins, A new view of statistics, Internet Society for Sport Science, 2000.

[10] T. Hall, A. Rainer en N. Baddoo, „Implementing Software Process Improvement: An Empirical Study," *Software Process Improvement and Practice*, vol. 7, pp. 3-15, 2002.

[11] T. Dyba, „An Empirical Investigation of the Key Factors for Success in Software Process Improvement," *IEEE Transactions on Software Engineering*, vol. 31, nr. 5, pp. 410-424, 2005.

[12] T. Chow en D.-B. Cao, „A survey study of critical success factors in agile software projects," *The Journal of Systems and Software*, vol. 81, pp. 961-971, 2008.

[13] S. C. Misra, V. Kumar en U. Kumar, „Identifying some important success factors in adopting agile software development practices," *The Journal of Systems and Software*, vol. 82, pp. 1869-1890, 2009.

[14] J. Sutherland, A. Viktorov, J. Blount en N. Puntikov, „Distributed Scrum: Agile project management with outsourced development teams," in *IEEE 40th Annual Hawaii International Conference on System Sciences (HICSS)*, 2007.

[15] M. Beedle , M. Devos, Y. Sharon, K. Schwaber en J. Sutherland, „SCRUM: An extension Pattern Language for Hyperproductive Software Development," in *Pattern Languages of Program Design*, Addison-Wesley, 2000, pp. 637-651.

[16] J. Sutherland, N. Harrison en J. Riddle, „Teams that Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams," in *47th Hawaii International Conference on System Science*, 2014.

[17] K. El Emam en A. Günes Koru, „A replicated survey of IT software project failures," *IEEE software*, vol. 25, nr. 5, pp. 84-90, 2008.

[18] M. Bhardwaj en A. Rana, „Key Software Metrics and its Impact on each other for Software Development Projects," *ACM SIGSOFT Software Engineering Notes*, vol. 41, nr. 1, pp. 1-4, 2016.

[19] D. Rubinstein, „Standish group report: There's less development chaos today," *Software Development Times*, vol. 1, 2007.

[20] R. Sonnekus en L. Labuschagne, „Establishing the Relationship between IT Project Management Maturity and IT Project Success in a South African Context," Proc. 2004," *PMSA Global Knowledge Conf., Project Management South Africa*, pp. 183-192, 2004.

[21] H. Barki, S. Rivard en J. Talbot, „Toward an assessment of software development risk," *Journal of Management Information Systems*, vol. 10, pp. 203-223, 1993.

[22] J. Jiang en G. Klein, „Software development risks to project effectiveness," *Journal of Systems and Software*, vol. 52, nr. 1, pp. 3-10, 2000.

[23] R. Schmidt, K. Lyytinen, P. Cule en M. Keil, „Identifying software project risks: An international Delphi study," *Journal of management information systems*, vol. 17, nr. 4, pp. 5-36, 2001.

[24] M. V. Mäntylä, M. Jørgensen, P. Ralph en H. Erdogmus, „Guest editorial for special section on success and failure in software engineering," *Empirical Software Engineering*, vol. April, pp. 1-17, 2017.

[25] R. Premrai, M. Shepperd, B. Kitchenham en P. Forselius, „An Empirical Analysis of Software Productivity Over Time," in *IEEE International Symposium Software Metrics*, Como, Italy, 2005.

Appendix A – Technical Report

Technical Report - Global Agile Metrics

1 Introduction and first inspection

1.1 Introduction

Goal of the first part of this project is to establish a way of working to analyze data and distinguish leading and lagging metrics.

Way of working:

- Get your data
- Inspect your data
- Correlate all data
- Define the output variable(s) (Lagging variable)
- Get an impression of the data and describe it
- Choose by intuition which variables could be lagging variables and test the assumption
- Calculate which independent variables are predictor for the output variable.(leading variable)
- Repeat above steps for all new data that is added to data set

1.2 Next Cycle: Addition of CDaaS Deployment Data to the Sprint Data

After obtaining the ServiceNow Backlog Management (BLM) data we added the CDaaS data. There are about 370 squads. For 316 we have BLM data for all the BLM variables. We excluded squads with missing data. For 101 squads be have CDaaS deployment data available. The union of both result in approximately 59 squad observations.

Regarding the CDaaS date we expect to see that:

- Teams with more deployments lead to shorter cycle.
- Shorter cycle times also indicate lower average story size?

Adagium of first cycle: Just add the data and see what happens.

Is the lagging variable still the lagging variable?

1.3 Data quality

CDaaS portal is not adequately connected to ServiceNow. It misses unique application names and application IDs and ServiceNow squad names.

Naming convention on the CD portal is not strict and CD portal teams and are named to the production name. Application ID is missing, but will be added in the future. CD Portal names are different from ServiceNow teams.

In preparation to go live of the IT ServiceManagement (ITSM) module in ServiceNow an excel list is prepared and maintained until go live of the link between application name and squad name. This list is not complete yet. An attempt to use the CI Long Name failed as it also contains environment post fixes, which were not expected.

Finally in a last attempt we tried to retrieve the squad name by finding out who deployed an application and whether this person is linked to only one squad name in ServiceNow. If so, that squad name was linked to the application name. Furthermore if that application was deployed more often and not yet linked to this squad name, then this squad name was added.

1.4 First Cycle

For the first cycle we tried some arbitrary metrics to find out how the R tooling works. And later on we added some more metrics. In this report you will see the combined data set.

1.5 Data source

- Snapshot Backlog management, (No time series)
- MS Access is used to gather the metrics.

1

Appendix A - Technical Report

1.6 Metrics Definitions

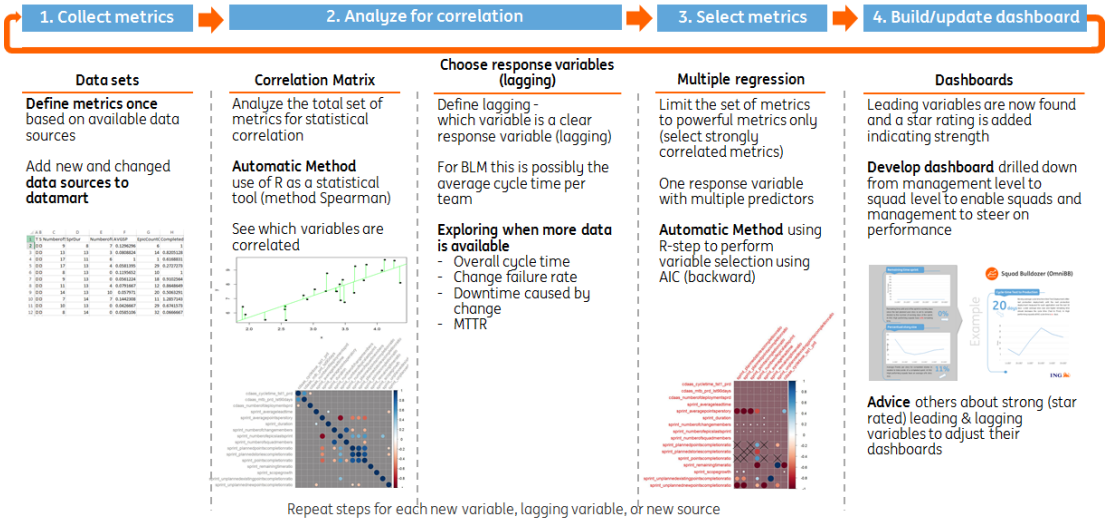
All sprint metrics are calculated based on the last completed sprint of a squad before a certain snapshot date

Metric Name	Metric Description
cdaas_cycletime_tst1_prd	Mean time from first test deployment after last production has been done until the next production deployment for all applications of a squad
cdaas_mtb_prd_lst90days	Mean time between production deployments in the last 90 days
cdaas_numberofdeploymentsprd	Number of deployments per squad
sprint_averageleadtime	The average time in days between creation of user story until completed, for the stories that got completed in the current sprint.
sprint_averagepointsperstory	Average size of a story compared to the completed points.
sprint_duration	Sprint End date – Sprint Start date
sprint_numberofchangemembers	Number of change members in sprint (opposed to business and operation members)
sprint_numberofepicslastsprint	Number of epics involved in a sprint
sprint_numberofsquadmembers	Number of squad members in a sprint
sprint_plannedpointscompletionratio	Sprint_PlannedPointsCompleted / Sprint_PlannedPoints
sprint_plannedstoriescompletionratio	Sprint_PlannedStoriesCompleted / Sprint_PlannedStories
sprint_pointscompletionratio	Sprint_CompletedPoints / Sprint_PlannedPoints
sprint_remainingtimeratio	Remaining time left until the end of the sprint measured from the moment that the last planned story's status is set to complete
sprint_scopegrowth	Change in point size during the sprint compare with the first day of the sprint. Can be positive, 0 or negative.
sprint_unplannedexistingpointscompletionratio	Unplanned Existing Completed Points / Sprint Completed Points Unplanned existing work is work that was already on the Product backlog and pulled into the current sprint after Day 1 of the sprint
sprint_unplannednewpointscompletionratio	Unplanned New Completed Points / Sprint Completed Points Unplanned New work is a story that is created after day 1 of the sprint.

2

Appendix A – Technical Report

1.7 Way of Working



1.8 Get your data

There is a daily download of the ServiceNow data. This is imported into Access. Using several queries the input for R is created and exported to csv format.

In the future this data will come directly from the datamart. The code for this link is already present in R here below.

First step: Load the data by reading the created csv.

```
if (FALSE) {
  dbhandle <- odbcDriverConnect("driver=SQL Server; server=xxxxxxxxxxx;trusted connection=true")
  dtinput <- sqlQuery(dbhandle, 'SELECT TOP 10 [Number] FROM [DataMart01].[dbo].[factsprint]')  # test
query
  odbcClose(dbhandle)
} else {
  dtinput <-  data.frame( read.csv("Data_BLM_CDAAS.csv", header = TRUE, sep=";" ) )
}
dtinput <- dtinput[ , order(names(dtinput))]
names(dtinput) <- tolower(names(dtinput))

#Leave out factor and character variables:
dt <- dtinput[,-grep ("factor|character", sapply (dtinput, class))]
#dt <- sort(dt,decreasing=TRUE)
#At some point in time you may wish to leave out columns by name
drops <- c("update_frequency", "bl_p_upnvsmax", "bl_p_upevsmax", "tribe", "squad_name")
dt <- dt[ , !(names(dt) %in% drops)]
```

3

Appendix A - Technical Report

1.9 Data inspection

A straightforward summary is always a good place to begin, because for one thing it will find any variables that have missing values.

Structure

```
str(dt)
```

```
## 'data.frame':    59 obs. of  16 variables:
##  $ cdaas_cycletime_tst1_prd                    : num  3.97 6.05 19.69 26.7 23.05 ...
##  $ cdaas_mtb_prd_lst90days                     : num  65.5 23 49 31.8 72.3 ...
##  $ cdaas_numberofdeploymentsprd                : int  65 1 14 7 1 4 18 39 4 13 ...
##  $ sprint_averageleadtime                      : num  34.7 27.2 25.5 33.3 40.7 ...
##  $ sprint_averagepointsperstory                : num  0.13 0.08 0.05 0.11 0.26 0.15 0.05 0.14 0.18 0.09
...
##  $ sprint_duration                             : int  14 13 13 14 13 13 20 20 13 13 ...
##  $ sprint_numberofchangemembers                : int  11 9 6 6 7 8 6 13 10 8 ...
##  $ sprint_numberofepicslastsprint              : int  10 15 16 11 5 7 24 5 7 11 ...
##  $ sprint_numberofsquadmembers                 : int  11 9 7 6 7 8 7 14 10 9 ...
##  $ sprint_plannedpointscompletionratio         : num  0.846 0.917 1 0.909 0.75 ...
##  $ sprint_plannedstoriescompletionratio        : num  0.84 0.941 1 0.893 0.88 ...
##  $ sprint_pointscompletionratio                : num  0.913 0.952 1.8 0.929 0.824 ...
##  $ sprint_remainingtimeratio                   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ sprint_scopegrowth                          : int  0 0 0 0 0 0 0 0 3 0 ...
##  $ sprint_unplannedexistingpointscompletionratio: num  0.0952 0.25 0.0926 0 0 ...
##  $ sprint_unplannednewpointscompletionratio    : num  0 0.05 0.2593 0.0385 0 ...
```

Summary

```
#t(summary(dt))

st <- t(do.call(cbind, lapply(dt, summary)))
st <- cbind(st, t(skewness(dt)), t(kurtosis(dt)) )
st
##                                                 Min. 1st Qu.   Median
## cdaas_cycletime_tst1_prd                     0.41370 11.2100 17.22000
## cdaas_mtb_prd_lst90days                      2.97900 23.0200 34.16000
## cdaas_numberofdeploymentsprd                 1.00000  3.0000  7.00000
## sprint_averageleadtime                       5.11200 25.5500 33.08000
## sprint_averagepointsperstory                 0.05000  0.0800  0.12000
## sprint_duration                             12.00000 13.0000 13.00000
## sprint_numberofchangemembers                 4.00000  7.0000  9.00000
## sprint_numberofepicslastsprint               1.00000  7.0000 11.00000
## sprint_numberofsquadmembers                  4.00000  8.0000 10.00000
## sprint_plannedpointscompletionratio          0.00000  0.4721  0.72410
## sprint_plannedstoriescompletionratio         0.00000  0.5074  0.67650
## sprint_pointscompletionratio                 0.04545  0.5601  0.68920
## sprint_remainingtimeratio                    0.00000  0.0000  0.00000
## sprint_scopegrowth                          -1.00000  0.0000  0.00000
## sprint_unplannedexistingpointscompletionratio  0.00000  0.0000  0.11110
## sprint_unplannednewpointscompletionratio       0.00000  0.0000  0.03061

##                                                 Mean 3rd Qu.     Max.
## cdaas_cycletime_tst1_prd                     17.96000 25.1800  46.5200
## cdaas_mtb_prd_lst90days                      36.77000 49.2200 104.9000
## cdaas_numberofdeploymentsprd                 16.73000 17.0000 194.0000
## sprint_averageleadtime                       35.52000 40.7200 136.7000
## sprint_averagepointsperstory                  0.15270  0.1800   1.0000
## sprint_duration                              14.42000 14.0000  34.0000
## sprint_numberofchangemembers                  9.10200 11.0000  15.0000
## sprint_numberofepicslastsprint               11.41000 15.0000  28.0000
## sprint_numberofsquadmembers                   9.76300 11.0000  15.0000
## sprint_plannedpointscompletionratio           0.64920  0.8258   1.0000
## sprint_plannedstoriescompletionratio          0.64020  0.8697   1.0000
## sprint_pointscompletionratio                  0.69730  0.8987   1.8000
## sprint_remainingtimeratio                     0.01425  0.0000   0.3077
## sprint_scopegrowth                            0.10170  0.0000   3.0000
## sprint_unplannedexistingpointscompletionratio  0.19170  0.2848   0.8889
## sprint_unplannednewpointscompletionratio       0.08522  0.1111   0.8000

##                                                Skewness Excess Kurtosis
## cdaas_cycletime_tst1_prd                     0.5583460      -0.1916332
## cdaas_mtb_prd_lst90days                      0.7303560       0.9596513
```

4

Appendix A – Technical Report

```
## cdaas_numberofdeploymentsprd                    4.1232022      18.7709372
## sprint_averageleadtime                          3.1495237      15.1360934
## sprint_averagepointsperstory                    4.6346163      26.7361704
## sprint_duration                                 3.7347634      16.7196697
## sprint_numberofchangemembers                    0.3598441      -0.4325227
## sprint_numberofepicslastsprint                  0.5736728      -0.2320281
## sprint_numberofsquadmembers                     0.3264224      -0.3740228
## sprint_plannedpointscompletionratio            -0.6125398      -0.4608336
## sprint_plannedstoriescompletionratio           -0.6127988      -0.4153330
## sprint pointscompletionratio                    0.5183665       1.5347557
## sprint_remainingtimeratio                       4.0891617      15.4764432
## sprint_scopegrowth                              3.5853894      13.9384310
## sprint_unplannedexistingpointscompletionratio   1.2242807       1.1303911
## sprint_unplannednewpointscompletionratio        2.7216999       8.9543944
```

```
##Display first and last rows of the table for inspection only
#head(dt)
#tail(dt)
```

1.10 Included observations and missing values

Currently observations with missing values are taken out of the dataset on forehand in Access.

To be included as an observation the variables must comply to the following rules:

Variable	Inclusion rule	Remark
sprint_numberofsquadmembers	> 0	Only Teams that have at least 1 CIO NL member are included
sprint_numberofchangemembers	not NULL	
sprint_duration	> 0	
sprint_averagepointsperstory	> 0	Maybe this rule should not be enforced
sprint_numberofepicslastsprint	not NULL	
sprint_pointscompletionratio	not NULL	Squad must have at least one Completed Sprint

The variable sprint_averagepointsperstory maybe has to be replaced with sprint_averageplannedpointsperstory. Currently it calculates only the user stories that are completed within the sprint.

A quick look at how variables in the current dataset are correlated can be established with a correlation matrix, combined with scatterplots.

5

Appendix A - Technical Report

1.11 The correlation matrix in color

For each pair of variables the correlation between the two is calculated. If the p-value is much less than 0.05, we reject the null hypothesis and conclude there is a significant relationship between the two variables in the linear regression model of the data.

The correlation value is always a number between -1 and 1.

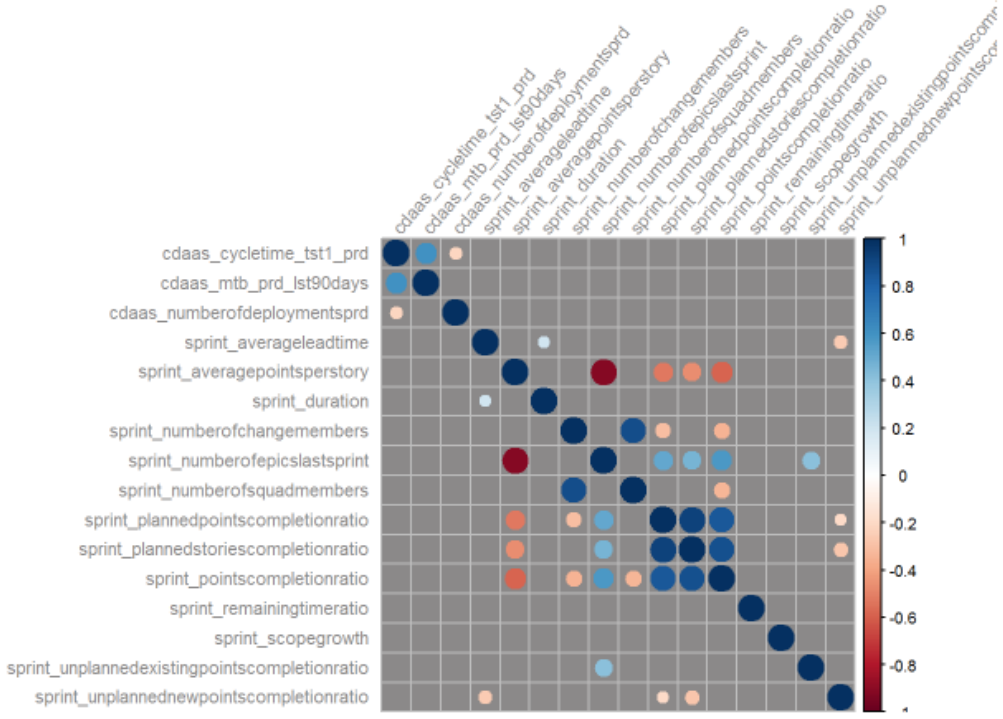
To blank out insignificant correlation values, the parameter p.mat must be provided as a matrix of p-values. If not provided, the arguments sig.level, insig, pch, pch.col, pch.cex are invalid and all correlation values will be shown. To see even the smallest circles a colored background is used.

Insignificant values are not shown (see parameter: insig="blank"). The significant level is set to 0.05.

Usually, a significance level (denoted as alpha) of 0.05 works well. A significance level of 0.05 indicates a 5% risk of concluding that a difference exists when there is no actual difference.

See also: http://support.minitab.com/en-us/minitab/17/topic-library/basic-statistics-and-graphs/introductory-concepts/p-value-and-significance-level/significance-level/

```
# corrplot the data ----------------------------------------------------------
# a Corplot matrix with coloured circles
dtm <- cor( dt,  method = "spearman")

par( mar=c(2,6,6,2))
res1 <- cor.mtest(dt,0.95)
corrplot(dtm, p.mat = res1[[1]], sig.level=0.05, insig = "blank" ,
              bg="snow4", tl.cex=0.7, cl.cex=0.6, tl.srt=50, tl.col="snow4"  )
```



Variables that seem to have no significant correlation when considered in a 1-to-1 situation, may act differently in the multiple regression. Sprint_remainingtimeratio is such a variable.

6

Appendix A – Technical Report

1.12 The correlation matrix in more detail for further analyses

The R package PerformanceAnalytics shows all the information in 1 plot.

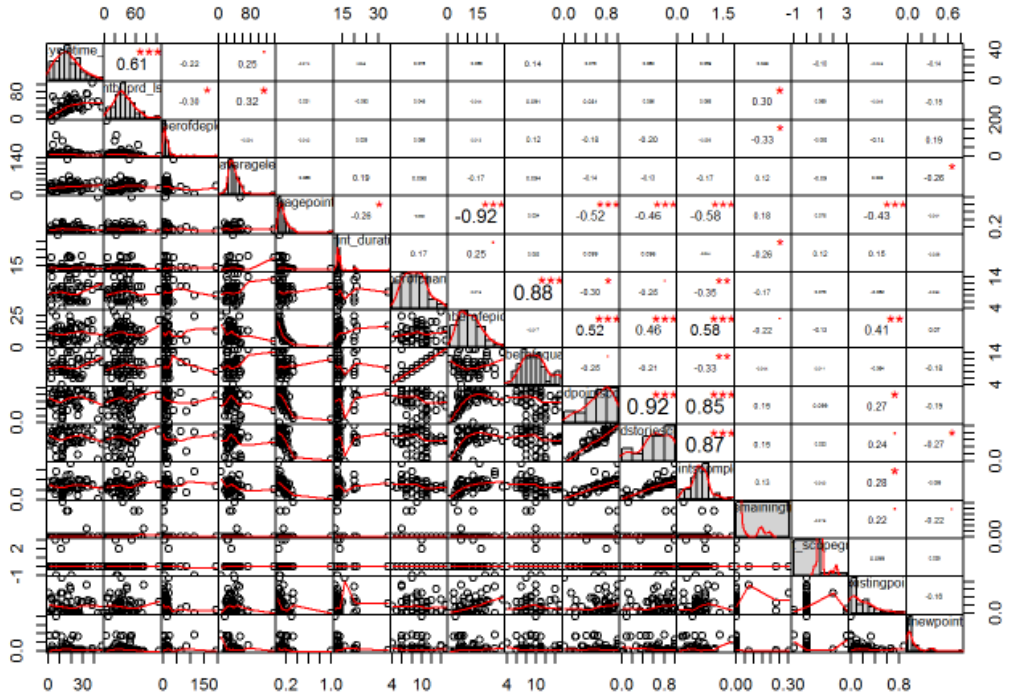
The chart.Correlation() function from the PerformanceAnalytics package produces a very nice scatterplot of the correlation matrix, with histograms, kernel density overlays, absolute correlations, and significance asterisks (0.05, 0.01, 0.001). Although the schema becomes unreadable when we get too much variables.

Visualization of a Correlation Matrix:

On top right the (absolute) value of the correlation plus the result of the cor.test as stars.

On bottom left, the bivariate scatterplots, with a fitted line.

```
# plot the data -------------------------------------------------------------------
library(PerformanceAnalytics)
suppressWarnings( chart.Correlation( dt,  histogram=TRUE, method="spearman") )
```




```
#' @param R          data for the x axis, can take matrix,vector, or timeseries
#' @param histogram  TRUE/FALSE whether or not to display a histogram
#' @param method     a character string indicating which correlation coefficient
#'                   (or covariance) is to be computed.  One of "pearson"
#'                   (default), "kendall", or "spearman", can be abbreviated.
#' @param /dots      any other passthru parameters into /code{/link{pairs}}
# within this Correlation function the folowwing significance cutpoints are used:
# Significance:     cutpoints = c(0    , 0.001, 0.01, 0.05, 0.1, 1),
#                   symbols   = c("***", "**" , "*" ,  "." , " "))
```

7

Appendix A - Technical Report

2 Choice of correlation method

Usually, in statistics, we measure four types of correlations: Pearson correlation, Kendall rank correlation, Spearman correlation, and the Point-Biserial correlation.

The histograms show that most of the data is not normalized and therefor use the Spearman correlation. In R we set this method in a variable.

```
cormethod = "spearman"
```

Spearman rank correlation

Spearman rank correlation is a non-parametric test that is used to measure the degree of association between two variables. It was developed by Spearman, thus it is called the Spearman rank correlation. Spearman rank correlation test does not assume any assumptions about the distribution of the data and is the appropriate correlation analysis when the variables are measured on a scale that is at least ordinal.

Further underpinning of the choice for Spearman can e.g. found at:
http://stats.stackexchange.com/questions/3730/pearsons-or-spearmans-correlation-with-non-normal-data/3744#3744

Spearman's correlation is a rank based correlation measure; it's non-parametric and does not rest upon an assumption of normality.
The sampling distribution for Pearson's correlation does assume normality; in particular this means that although you can compute it, conclusions based on significance testing may not be sound. …with large sample this is not an issue. With small samples though, where normality is violated, Spearman's correlation should be preferred.

8

Appendix A – Technical Report

3 Automatic methods to find the leading (predictor) variables

3.1 Multiple regression

See: http://www.statmethods.net/stats/regression.html

The step formula can be used to find the predictors for a chosen lagging (response) variable. Step computes the (generalized) Akaike An Information Criterion for a fitted parametric mode.

Automatic methods are useful when the number of explanatory variables is large (as in our case) and it is not feasible to fit all possible models. In this case, it is more efficient to use a search algorithm (e.g., Forward selection, Backward elimination and Stepwise regression) to find the best model.

What would be our lagging (response) variable for BLM? From our correlation matrix we learn that Lead-time BLM (LT_BLM) has the most significant relations with other variables. Also Cnt_Sqm is a candidate, but is would be more a leading (predictor) than a lagging (response) variable. (Lt_blm and ctn-sqm were work names during the first cycle for sprint_averageleadtime and sprint_numberofsquadmembers.)

At least this is what we thought in the beginning. During the analyses we found that sprint_plannedstoriescompletionratio was a better lagging indicator. It is more related to the predictiveness and stability of the squad.

For each lagging variable to test, we calculate by providing the model and let the step function do its work and show the result. A function has been created so that we can easily repeat the work:

```
stepmodel <- function( formulastring, data) {
    model <- lm( formula=as.formula(formulastring), data=data)
    model_step <- step(model, direction="backward", trace=FALSE)
    plot_coeffs(model_step)    #user defined function to plot coefficients
}
```

For each model we define the lagging variable and the independent variables that we want to ignore. The latter have a "-" sign before their name.

Example:

stepmodel(completion_ratio ~ . - sprint_plannedpointscompletionratio -sprint_plannedstoriescompletion_ratio, data=dt)

9

Appendix A - Technical Report

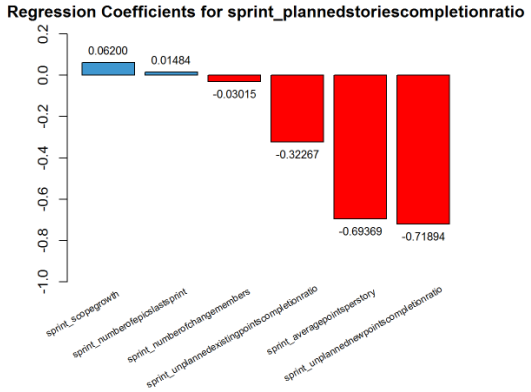
3.2 Sprint_PlannedStoriesCompletionRatio

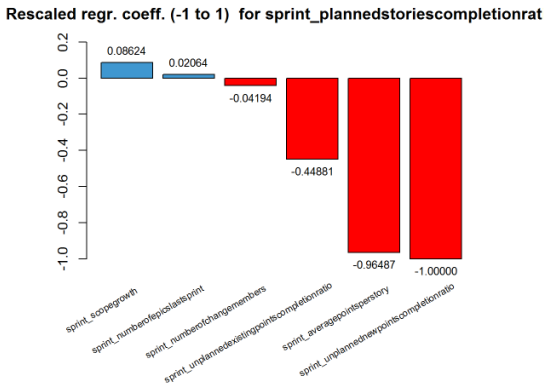
Sprint_plannedstoriescompletionratio ignores the scope growth of user stories and this is an advantage compared to Sprint_plannedpointscompletionratio.

Definition: Sprint_PlannedStoriesCompleted / Sprint_PlannedStories

Model: sprint_plannedstoriescompletionratio is strongly correlated with sprint_pointscompletionratio and sprint_plannedpointscompletionratio. Therefore we omit these in the regression model.

```
stepmodel( sprint_plannedstoriescompletionratio ~ . -sprint_pointscompletionratio
 -sprint_plannedpointscompletionratio,  data=dt )

## sprint_plannedstoriescompletionratio
```





10

Appendix A – Technical Report

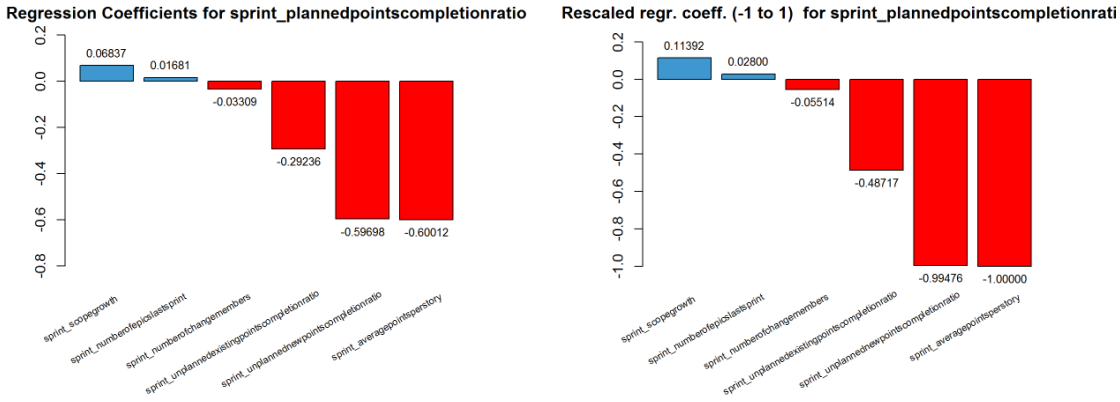
3.3 Planned Points Completion Ratio

More interesting it is to see whether you finish what you promised. In other words does a team complete its planned work. And if not, what is the cause?

Definition: Sprint_PlannedPointsCompleted / Sprint_PlannedPoints

Model: sprint_plannedpointscompletionratio is strongly correlated with sprint_storiescompletionratio and sprint_plannedpointscompletionratio. Therefore we omit these in the regression model.

```
stepmodel( sprint_plannedpointscompletionratio ~ . -sprint_pointscompletionratio
 -sprint_plannedstoriescompletionratio,  data=dt )
## sprint_plannedpointscompletionratio
```



Result:
We see that especially that "sprint_unplannednewpointscompletionratio" and sprint_averagepointsperstory has a negative impact on the completion of planned story points.

The variable "sprint_unplannedexistingpointscompletionratio" has also has a negative influence. The mechanism behind this is unclear. Could it mean that that priority gets changed and that fellow team members are disturbed in completing the planned work?

The other variables that are relevant do not show a high impact and can be neglected.

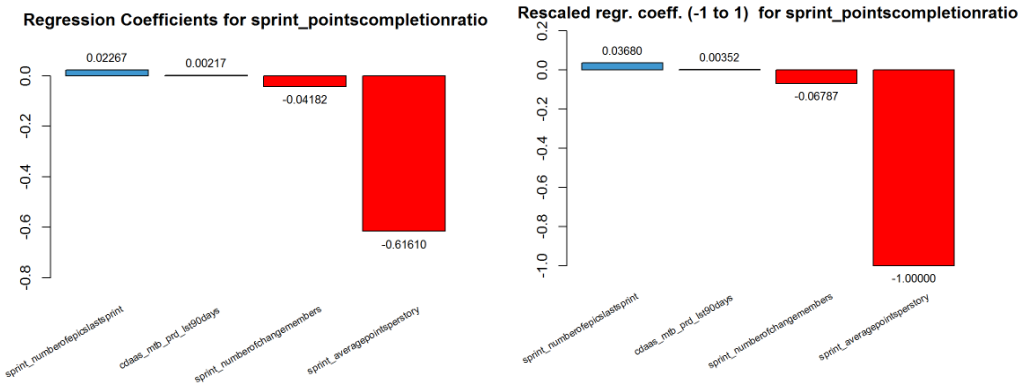
Appendix A - Technical Report

3.4 Sprint_pointscompletionratio

Definition: Sprint_CompletedPoints / Sprint_PlannedPoints

Model: This definition includes unplanned work in the CompletedPoints and possible scope growth (- or +).

```
summary(dt$sprint pointscompletionratio)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.04545 0.56010 0.68920 0.69730 0.89870 1.80000

stepmodel( sprint pointscompletionratio ~ .  -sprint plannedpointscompletionratio -
sprint_plannedstoriescompletionratio, data=dt )

## sprint_pointscompletionratio
```



Result

- An increase of average story points leads to a lower completion rate.

12

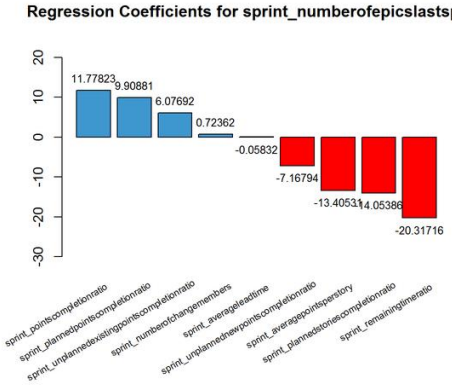
Appendix A – Technical Report

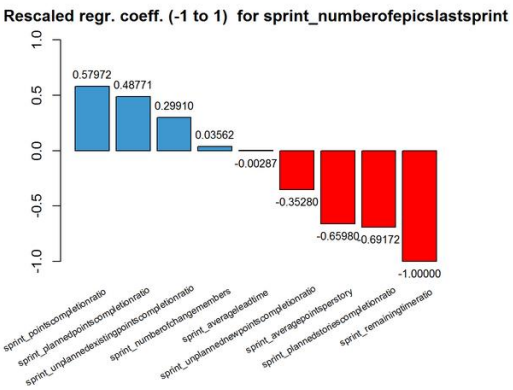
3.5 Focus

We expect that focus is high when the number of epics in one sprint is low. But is it true?

Model: Count_epics_last_sprint is tested against all other variables

```
stepmodel( sprint numberofepicslastsprint ~ .  , data=dt )
## sprint_numberofepicslastsprint
```





Results

Positive impact:

- Completing any kind of work (planned or unplanned existing work) increases the number of epics in the a completed sprint

Negative impact:

- Strongest negative impact is caused by the sprint_remainingtimeratio. So, if there is time left it seems that a squad continues with the other stories of the same epic?
- Unplanned new work (Surprise work) is decreasing the number of epics in a sprint just a bit.

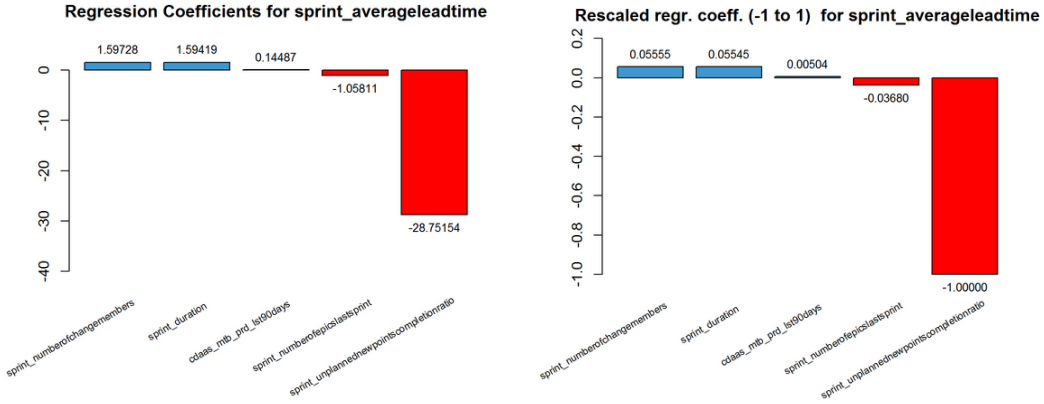
Appendix A - Technical Report

3.6 Lead Time

Lead Time is not regarded as a lagging indicator, but still it is interesting to see what is influencing lead time.

Definition: Here Lead time is the average time in days between creation of user story until completed, for the stories that got completed in the current sprint. The creation of the story happens outside the sprint. We might have to consider to leave it out of the dataset in the future and include it when observing a large timeline.

```
stepmodel( sprint_averageleadtime ~ .  -sprint_pointscompletionratio -sprint_plannedpointscompletionratio,
data=dt )
## sprint_averageleadtime
```



Results

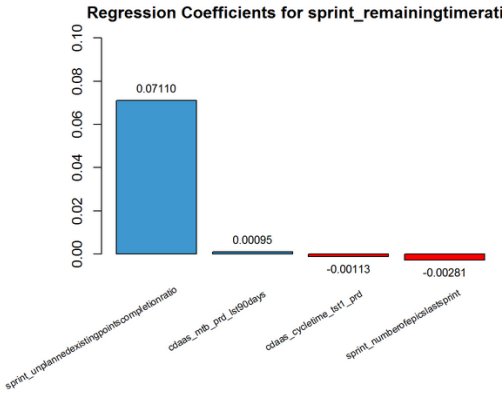
- Unplanned new decreasing effect on the sprint average lead-time. This ratio is based on the completed stories either planned or unplanned. Unplanned new work has an incident character and must be realized in shorter time, hence smaller lead times and it replaces planned work. The overall effect seems to lower the average lead time.

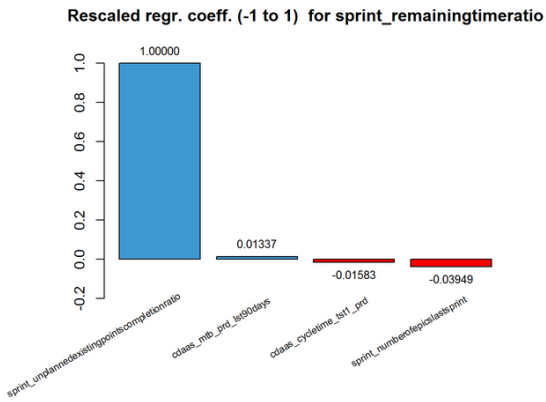
14

Appendix A – Technical Report

3.7 Remaining Time

```
stepmodel( sprint_remainingtimeratio ~ . , data=dt )
## sprint_remainingtimeratio
```





Result

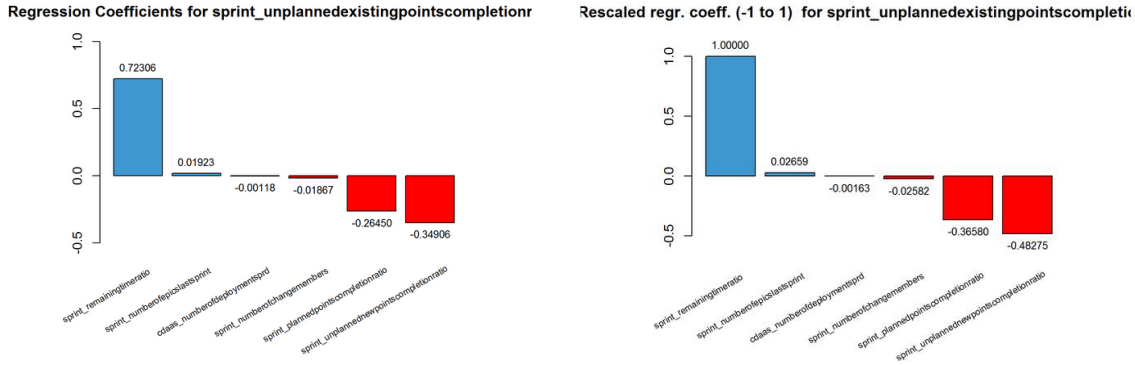
- When unplanned existing work is completed in a sprint, than there was obviously time to do so. We state it this way, as it is more the other way around. If you have completed your planned work in time and have some time left, then you can pick up some additional work of the backlog. See next model.

15

Appendix A - Technical Report

3.8 Unplanned Existing Work

```
stepmodel(sprint_unplannedexistingpointscompletionratio ~ . -sprint_pointscompletionratio , data=dt )
## sprint_unplannedexistingpointscompletionratio
```



Result:

- If there is time left after planned work is done, then unplanned is picked up. Arrival of unplanned new work on the other hand is really killing for unplanned existing work. Finishing planned work of new unplanned work has a negative impact on this ratio.

16

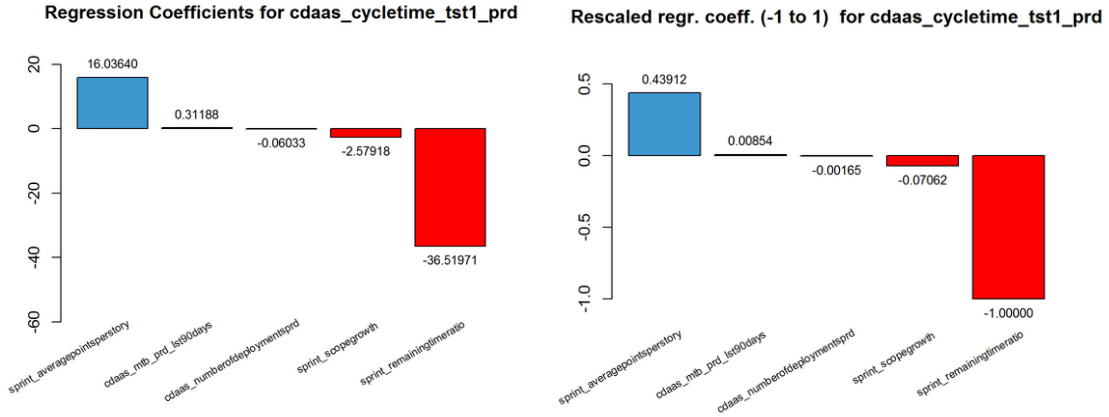
Appendix A – Technical Report

3.9 cdaas_cycletime_tst1_prd

Definition: Time from first test deployment after last production has been done until the next production deployment.

Model: This variable against all other variables.

```
summary(dt$cdaas cycletime tst1 prd )
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4137 11.2100 17.2200 17.9600 25.1800 46.5200
stepmodel( cdaas_cycletime_tst1_prd  ~ .  , data=dt )
## cdaas_cycletime_tst1_prd
```



Result

These are the first conclusion based on a limited data set of 59 observations:

- Although we don't see a direct linear correlation, the multiple regression with cdaas_Cycletime_tst1_prd reveals interesting leading metrics for this variable. The other two CDaaS variables are not strongly and significantly correlated with the response variable
- When average story points per story raises then also the cycle time increases (seems obvious)
- Squads that remaining time left after they finished the planned work in the sprint, show a decrease in cycle time.

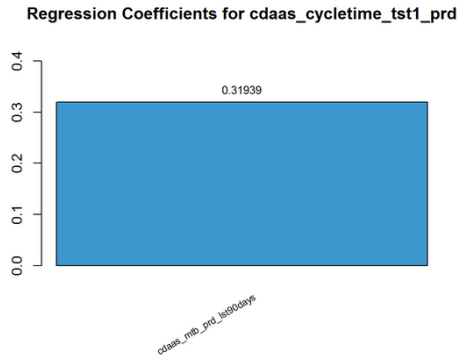
Appendix A - Technical Report

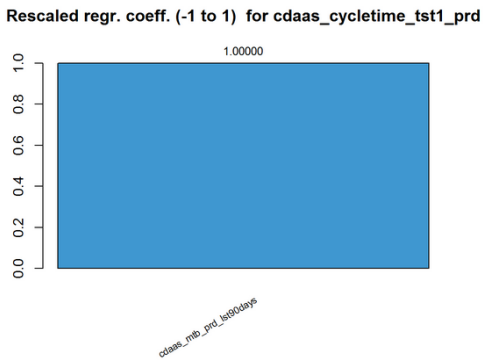
3.10 Focus on CDaaS variables only

Definition: Time from first test deployment after last production has been done until the next production deployment.

Model: This variable against only the two other CDaaS variables.

```
summary(dt$cdaas cycletime tst1 prd )
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.4137 11.2100 17.2200 17.9600 25.1800 46.5200
stepmodel( cdaas_cycletime_tst1_prd ~ cdaas_mtb_prd_lst90days + cdaas_numberofdeploymentsprd , data=dt )
## cdaas_cycletime_tst1_prd
```





Result:
Similar to the linear regression. cdaas_mtb_prd_lst90days is most strongly correlated to the response variable.
The set of CDaaS variables is not complete yet. Much more variables must be retrieved from the system.

Appendix A – Technical Report

4 Combining the models in one model corrplot

4.1 Impact of the leading variables on the lagging variables

And finally after analyzing all lagging variables we can put them all together in one diagram to see the impact of each leading variable on the lagging variable candidates.

The following two variables are lagging candidates within the current data set.

#	Response variable	Remark
1	sprint_plannedstoriescompletionratio	Obvious. We want to know what influences our planned work. If we base it on the number of stories than it excludes the influence of scope growth in value of this ratio, so that it stays within the range 0 - 100%.
2	sprint_plannedpointscompletionratio	Obvious. We want to know what influences our planned work
3	sprint_pointscompletionratio	this is a model with all the completed points including the unplanned work

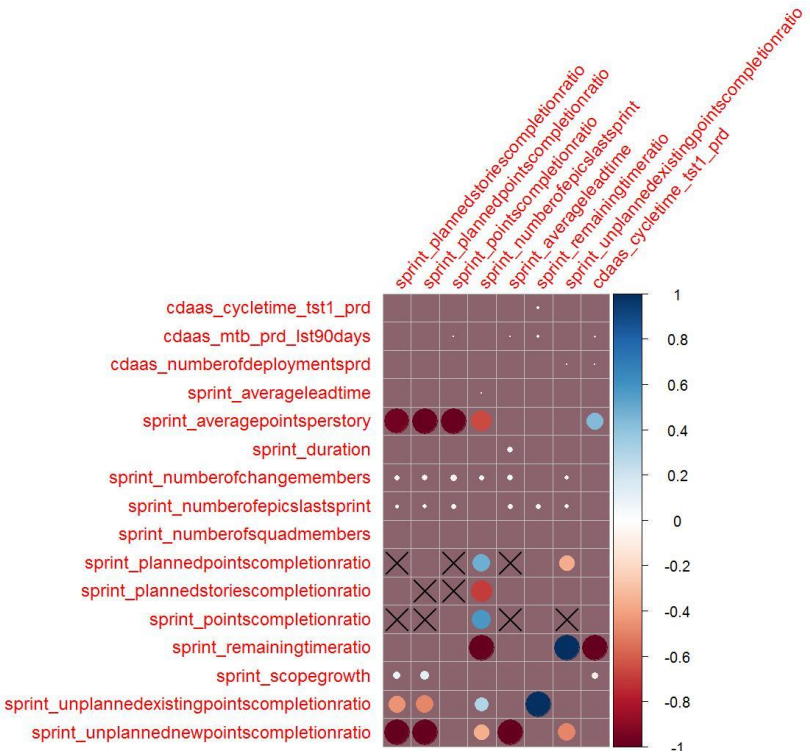
Although the following variables are not considered lagging, we wanted to know the influence of other variables.

#	Response variable	Remark
4	sprint_numberofepicslastsprint	We had the idea that working on multiple epics would distract the member focus and would negatively impact the completion rate. so let's test it
5	sprint_averageleadtime	Is there anything that influences the Leadtime?
6	sprint_remainingtimeratio	What is influencing the amount of remaining time after all planned stories are completed
7	sprint_unplannedexistingpointscompletionratio	Added to prove that this sprint_remaingtimeration is a predictor to this variable
8	cdaas_cycletime_tst1_prd	Test if we can already see some interaction between cdaas and backlog management data

```
## Impact of the leading variables on the lagging variable

c1 <- stepcoeffs( sprint_plannedstoriescompletionratio ~ . -sprint_pointscompletionratio
 -sprint_plannedpointscompletionratio,  data=dt)

c2 <- stepcoeffs( sprint_plannedpointscompletionratio ~ . -sprint_pointscompletionratio -
sprint_plannedstoriescompletionratio,  data=dt)

c3 <- stepcoeffs( sprint pointscompletionratio ~ .  -sprint plannedpointscompletionratio -
sprint_plannedstoriescompletionratio, data=dt)

c4 <- stepcoeffs( sprint_numberofepicslastsprint ~ ., data=dt)

c5 <- stepcoeffs( sprint_averageleadtime ~ .  -sprint_pointscompletionratio -
sprint_plannedpointscompletionratio, data=dt )

c6 <- stepcoeffs( sprint_remainingtimeratio ~ . , data=dt)

c7 <- stepcoeffs( sprint_unplannedexistingpointscompletionratio ~ . -sprint_pointscompletionratio ,
data=dt)

c8 <- stepcoeffs( cdaas_cycletime_tst1_prd  ~ ., data=dt)

mar.before <- par("mar")
  par( mar=c(2,7,7,2)+0.1 )
  ctest <- combinecoeffs( list(c1,c2,c3,c4,c5,c6,c7,c8),data=dt)

  corrplotll(ctest, leadinglagging=TRUE, tl.cex=0.7, cl.cex=0.6, tl.srt=50,  bg="pink4", cl.ratio=0.8,
cl.lim = c(-1,1), insig="blank", plotCI="n")

mar <- mar.before
```

Appendix A - Technical Report

A modification of the existing corrplot function (see package corrplot) has been created to combine the models in one diagram.



4.2 What do you see?

- For each lagging variable you see what the impact is of the leading variable.
- Blue means a positive impact and red means a negative impact on the lagging variable.
- The strongest leading variable (neg or pos) has the largest circle.
- Same size circles on the same row do not mean they are equal.
- Empty squares means the coefficient was 0.
- Crossed out variables are excluded from the lagging model. Usually this is done to avoid collinearity (independent variables that are highly correlated).
- For each lagging variable the leading variables are calculated using multiple linear regression.
- All multiple linear regression coefficients are then rescaled on a scale of -1 to 1.

4.3 What can we tell?

Using the leading/lagging diagram we can tell more about the interaction of the variables in the model.

1. Higher average story points have a negative impact on the planned completion ratio (either points or stories) and the total completion ratio. This is the story of the jar filled with rocks instead of pebbles. Smaller pieces of work can easier be finished within the sprint time frame. Large chunks of work can flow to the next sprint, even if only half a day is necessary to finish it.
2. Higher average story points decrease the number of epics in a sprint decrease. Pebbles and rocks again. The larger the average story points are, the lesser stories you probably have, since they also have to fit within the sprint. This statistically reduces the chance that they belong to multiple epics.
3. Planned Stories Completion Rate: If this ratio goes up, then also the number of epics increases. at the same time unplanned backlog work (unplanned existing) reduces.
4. Planned Stories Completion Rate: if this ratio goes up, the number of epics decreases.
   It was expected that it would have the same tendency as Planned Stories Completion Rate. This needs further study. Perhaps with a larger data set, we will see more details.

20

5. Remainingtimeratio: If there is time left after all planned work is done, than we see that squad pickup backlog work.
6. However, when Unplanned new work (forgotten to specify, or incidents) pops up it has a negative impact on the planned work and the ability to pick up backlog work. Also the number of epics decreases as well as the average lead time.
7. Average lead-time is negatively impacted by new unplanned work, possibly because it delays planned work that was already started in the current sprint or earlier and can't finish in the planned time. Average lead-time is not tightly linked to a sprint, but more to a user story.
8. No significant impact is caused in these models by the other variables.

4.4 Conclusions

There are two candidates for the lagging variable:

- sprint_plannedstoriescompletionratio
- sprint_plannedpointscompletionratio

Sprint_plannedstoriescompletionratio is chosen as the lagging variable.

Both ratios focus on the completion of planned work.

Sprint_plannedstoriescompletionratio has the advantage that scope growth (measured in points) has no influence on the ratio value.

The leading variables are:

- sprint_unplannednewpointscompletedratio
- sprint_unplannedexistingpointscompletedratio

Unplanned new work (and also unplanned existing work) has a negative influence on the planned work and leads to delay. A squad should dig into the nature of the unplanned work, and think out ways to prevent unplanned work. The average points should be small enough to reduce the impact on the sprint_plannedstoriescompletionratio.

Currently roughly 10% finish all the work that they plan at the start of the sprint.

Dashboarding:
Currently we don't have a variable to calculated the planned points completed versus the total completed points. This would come in handy when creating the dashboard.

1. Can we also use sprint_pointscompletionratio instead? No, it includes incidents and fill up work. Working on incidents or filling up buffers is not what squads should target for.
2. Remaining time It was suggested in an article written by Jeff Sutherland that improving teams always have a little spare time left after the planned work was done. [Teams that Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams, Jeff Sutherland e.a., 2014] In the current dataset the remainingtimeratio is a low 3% and only 11% of the squads finish all the planned work. The remaining is not predictor for sprint_plannedpointscompletionratio.

Recommendation for model improvement *

1. We currently used the sprint_averagepointsperstory for completed user stories in the sprint. It would be better to calculate the average point per story for the user stories as present at the sprint change (Day 1). That gives a better idea of how large the average stories are that a squad has.
2. Sprint_remainingtimeratio currently includes weekend days for remaining days and for the sprint duration. In the datamart model the working days will be excluded.

21

Appendix A - Technical Report

5 Outliers and Distribution

Outliers are not taken out of the dataset yet.

5.1 Theoretical Background

DRAFT:

https://www.r-bloggers.com/outlier-detection-and-treatment-with-r/

Outliers in data can distort predictions and affect the accuracy, if you don't detect and handle them appropriately especially in regression models.

https://explorable.com/statistical-outliers:
Statistical outliers are data points that are far removed and numerically distant from the rest of the points. Outliers occur frequently in many statistical analyses and it is important to understand them and their occurrence in the right context of the study to be able to deal with them.

An outlier can be a chance phenomenon, measurement error or due to an experimental error. It can also occur in special cases that have a heavy tail distribution, in which cases the assumption of a normal distribution may not hold.

Certain statistical estimators are able to deal with statistical outliers and are robust, while others cannot deal with them. A typical example is the case of a median, that can deal with outliers well, since it would not matter whether the extreme point is far away or near the other data points, as long as the central value is unchanged.

The mean, on the other hand, is affected by outliers as it increases or decreases in value depending on the position of the outlier.

One should be careful while dealing with outliers and not mistake them for experimental errors or exceptions at all times. outliers can indicate a different property and may indicate that they belong to a different population.

Many times, outliers should be given special attention till their cause is known, which is not always random or chance. Therefore a study needs to be made before an outlier is discarded.

Statistical outliers are common in distributions that do not follow the traditional normal distribution. [ING: And such is the case in our dataset] For example, in a distribution with a long tail, the presence of statistical outliers is more common than in the case of a normal distribution.

In case of a normal distribution, it is easy to see that at random, about 1 in 370 observations will deviate by more than three times the standard deviation from the mean. This ratio decreases drastically for more distant values. Therefore if there is a more than frequent case of data away from the mean, then the cause needs to be examined.

For example, if out of 1000 data points, 5 points are at a distance of four times the standard deviation or more, then these outliers need to be examined.

5.1.1 Detect Outliers

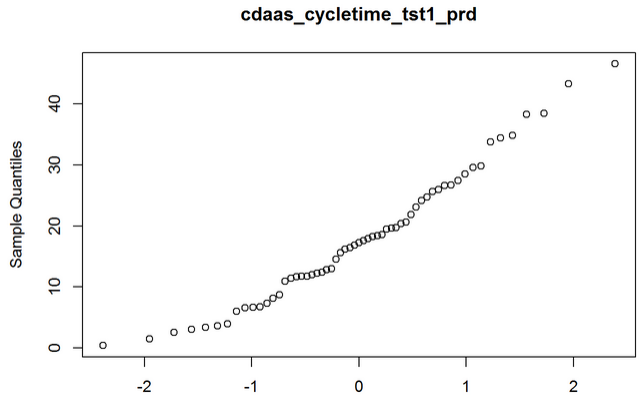
Univariate approach
For a given continuous variable, outliers are those observations that lie outside 1.5 * IQR, where IQR, the 'Inter Quartile Range' is the difference between 75th and 25th quartiles. Look at the points outside the whiskers in below box plot.

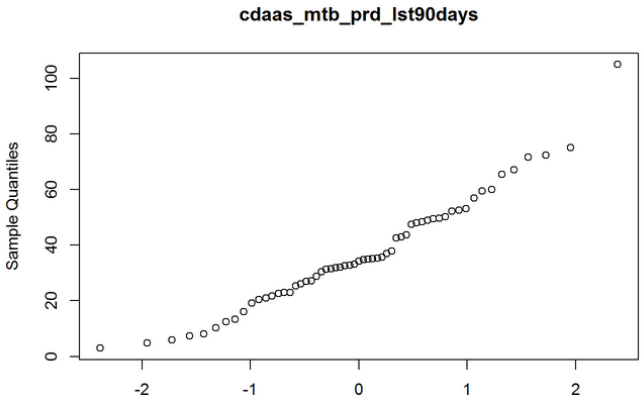
http://www.statistics4u.com/fundstat_eng/cc_outlier_tests.html :

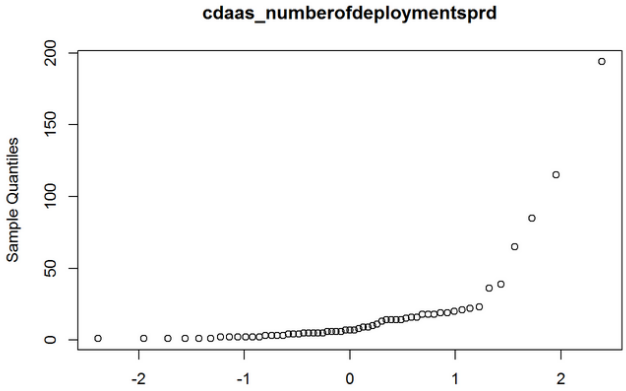
```
#par(mfrow=c(3,2))
for(i in seq(1,ncol(dt),1)) qqnorm(dt[,i], main=colnames(dt)[i])
```

22

Appendix A – Technical Report

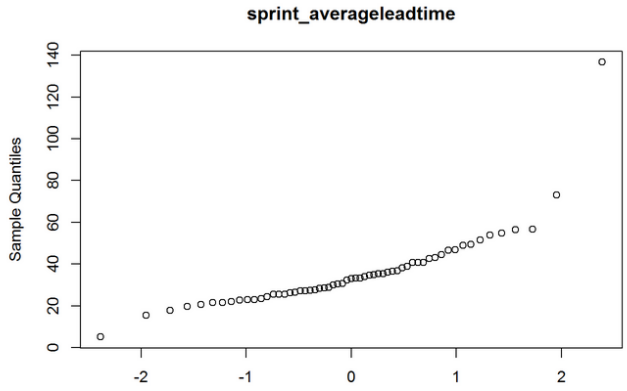


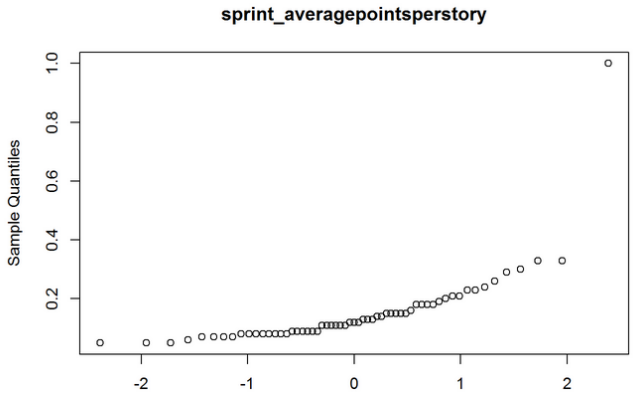


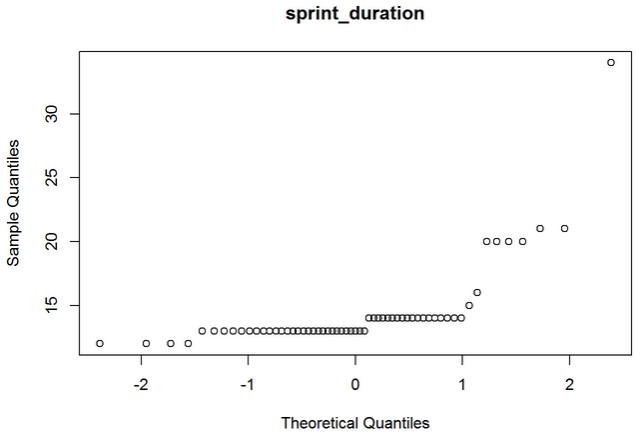


23

Appendix A - Technical Report

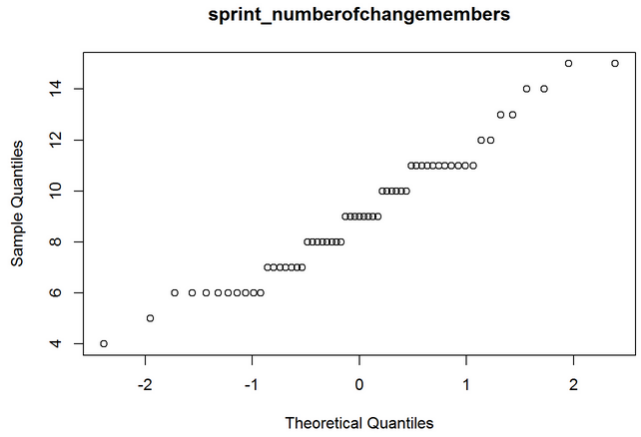


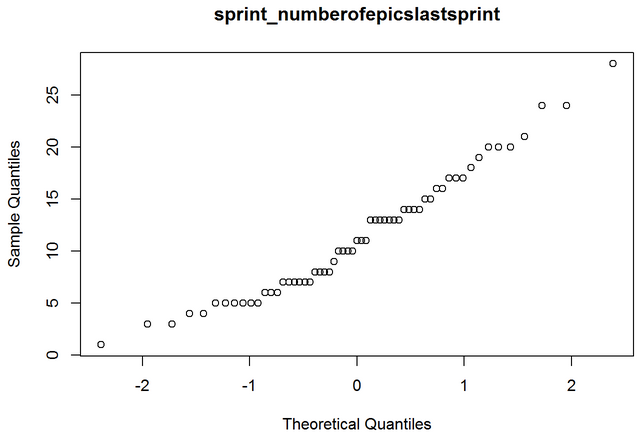


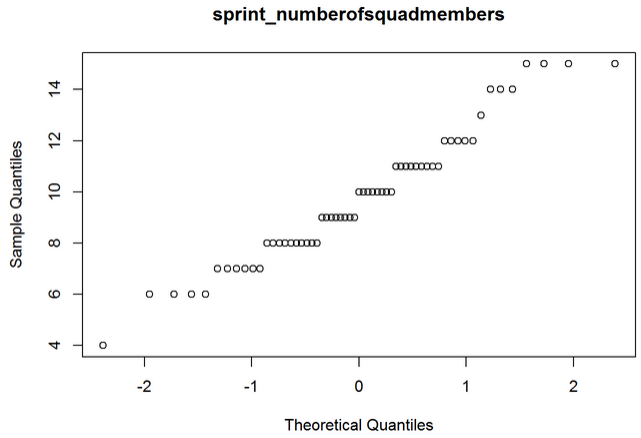


24

Appendix A – Technical Report

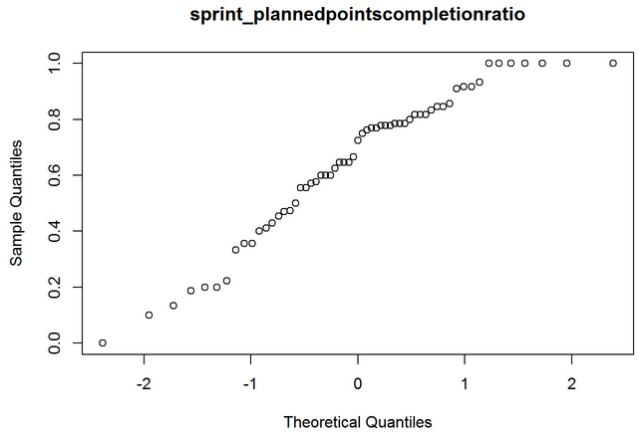


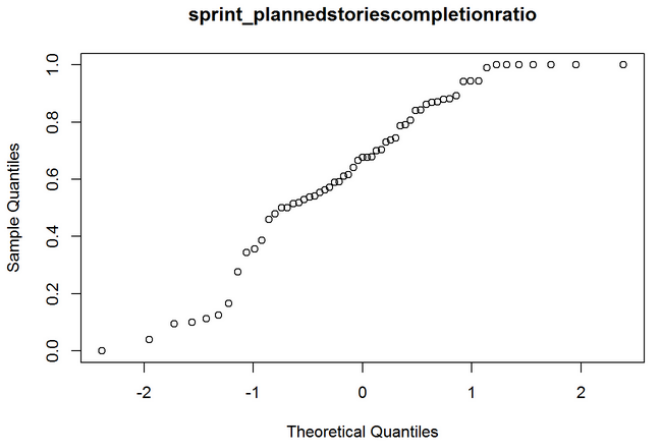


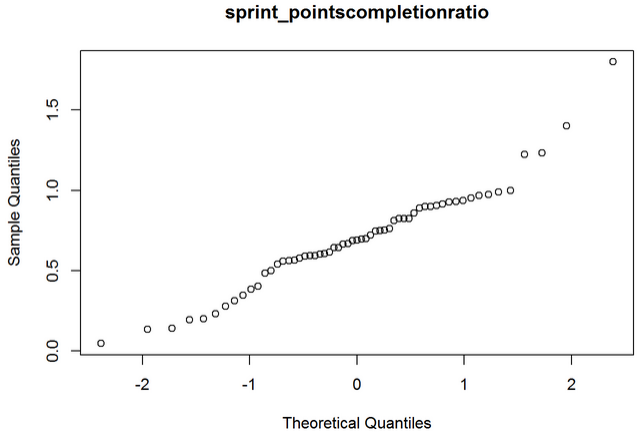


25

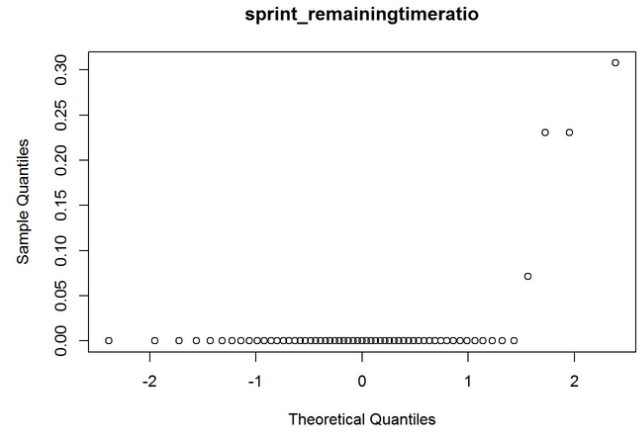
Appendix A - Technical Report

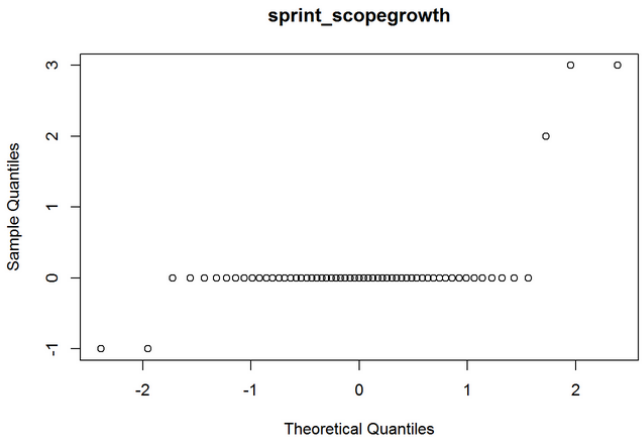


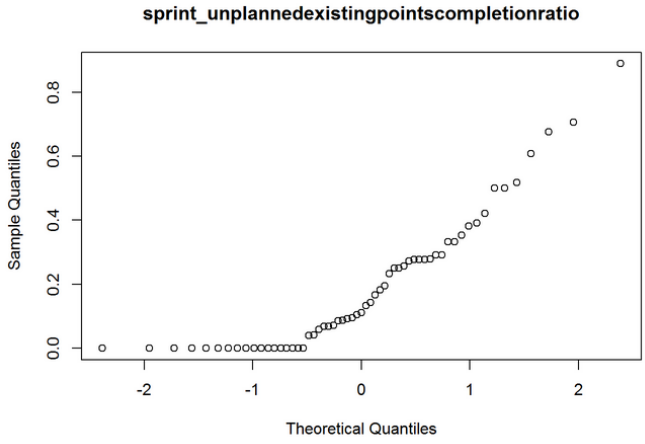




Appendix A – Technical Report

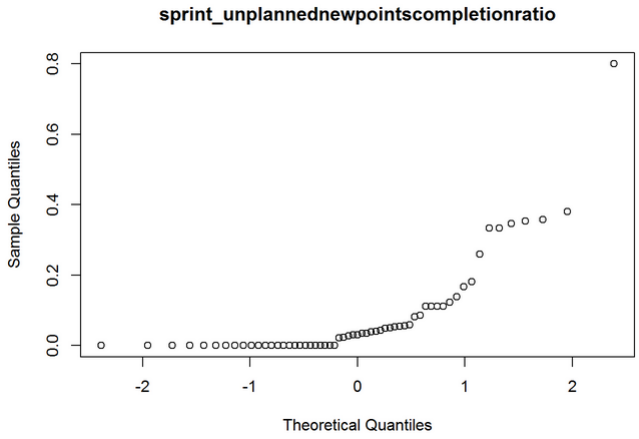






27

Appendix A - Technical Report



1 All histograms

- Echo is set to off. Code is displayed. JPG files are created for use in PowerPoint files.
-

```
cn <- colnames(dt)
mybreaks <- c( 100, 200, 50, 100, 50, NULL,100, 100, 100, 100,100,100,100,100,100,100,100)
z <- 1
par(mfrow = c((ncol(dt)+1)/2, 2))
repeat {
  jpeg( filename=paste("HIST " , cn[z], ".jpg", sep=""), width = 400, height=300)
    hist( x=dt[,z] , xlab= cn[z], col="lightblue1", main=cn[z], breaks = mybreaks[z] )
    abline(v=median(dt[,z]), col="magenta", lwd=2)
    abline(v=mean(dt[,z]), col="blue", lwd=2)
    legend("topright", legend=c("Median", "Mean"),
           fill = c("magenta", "blue"), lty=1:2, cex=0.8,
          box.lty=0)
  dev.off()
  z = z+1
  if (z > ncol(dt)){
    break
  }
}
graphics.off()
```

```
udf_singlehist( colname = "average_leadtime_sprint" , data=dt, breaks = 50)
## [1] "average leadtime sprint does not exists as column name in data"
udf_singlehist( colname= "average_story_points", data=dt, breaks = 25)
## [1] "average_story_points does not exists as column name in data"
udf_singlehist( colnum = 3, data=dt, breaks = 10)
udf_singlehist( colnum = 4, data=dt, breaks=60)
udf_singlehist( colnum = 5, data=dt, breaks=60)
udf_singlehist( colname="sprint_plannedpointscompletionratio", data=dt)
summary(dt$sprint plannedpointscompletionratio)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.4721  0.7241  0.6492  0.8258  1.0000
```

There are only 10 squads out of 306 that have 0 points completed.
35 squads, roughly 10% finish all the work that they plan.

```
udf singlehist( colname="sprint remainingtimeratio", data=dt)
summary(dt$sprint_remainingtimeratio)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.00000 0.00000 0.01425 0.00000 0.30770
```

28

Appendix A – Technical Report

The sprint_remainingtimeratio has a low 3% in this data set. This is as expected for a dataset where only 11% ends its planned work. Maybe it can't be measured well enough. It might well be that user stories are put to completed at the end of the sprint.

More histograms will be included in the nearby future.

Appendix A - Technical Report

6 External links

The initial setup of this statistical part of the project follows the receipt as described in:

https://ww2.coastal.edu/kingw/statistics/R-tutorials/multregr.html

and was later adapted to insights from other websites

7 System Environment

```
## R version 3.3.2 (2016-10-31)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC COLLATE=Dutch Netherlands.1252  LC CTYPE=Dutch Netherlands.1252
## [3] LC_MONETARY=Dutch_Netherlands.1252 LC_NUMERIC=C
## [5] LC TIME=Dutch Netherlands.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] fpc_2.1-10                   cluster_2.0.6
##  [3] kmeans.ddR_0.1.0             reshape2_1.4.2
##  [5] psych_1.7.5                  RODBC_1.3-15
##  [7] coefplot_1.2.4               ggplot2_2.2.1
##  [9] texreg_1.36.23               PerformanceAnalytics_1.4.3541
## [11] xts_0.9-7                    zoo_1.7-14
## [13] markdown 0.7.7               rmarkdown 1.3
## [15] xtable_1.8-2                 knitr_1.15.1
## [17] corrplot 0.77
##
## loaded via a namespace (and not attached):
##  [1] modeltools_0.2-21 kernlab_0.9-25    lattice_0.20-34
##  [4] colorspace_1.3-2  htmltools_0.3.5   stats4_3.3.2
##  [7] yaml_2.1.14       foreign_0.8-67    DBI_0.6
## [10] prabclus_2.2-6    plyr_1.8.4        robustbase_0.92-7
## [13] stringr_1.2.0     munsell_0.4.3     gtable_0.2.0
## [16] mvtnorm_1.0-6     evaluate_0.10     flexmix_2.3-13
## [19] parallel_3.3.2    class_7.3-14      DEoptimR_1.0-8
## [22] trimcluster_0.1-2 Rcpp_0.12.9       scales_0.4.1
## [25] backports 1.0.5   diptest 0.75-7    ddR 0.1.2
## [28] useful_1.2.1      mnormt_1.5-5      digest_0.6.12
## [31] stringi_1.1.2     dplyr_0.5.0       grid_3.3.2
## [34] rprojroot_1.2     tools_3.3.2       magrittr_1.5
## [37] lazyeval_0.2.0    tibble_1.2        MASS_7.3-45
## [40] assertthat_0.1    R6_2.2.0          mclust_5.2.3
## [43] nnet_7.3-12       nlme_3.1-131
```

TUD-SERG-2017-010
ISSN 1872-5392