Assessing Programming Difficulty and Effort:

Statistical Correlations with the Index of Internal Effort

Alisson Antônio de Oliveira
Instituto Federal do Paraná (IFPR) - Campus Curitiba
R. João Negrão, 1285 - Rebouças, Curitiba - PR, 80230-150
alisson.oliveira@ifpr.edu.br

Abstract. *Code metrics are essential variables for evaluating complexity and effort in different development contexts. However, there is a gap in comprehensive measurement of complexity and team effort. This study uses a dataset of 46 codes from the Brazilian Informatics Olympiad to investigate correlations between metrics such as difficulty, number of lines of code, cyclomatic complexity, and the Index of Internal Effort (IIE), a generic framework for measuring Explicit Intellectual Activities (EIA). Statistical results revealed positive and significant correlations among the studied metrics, including the IIE, validating its applicability as a complexity indicator. This study contributes to the understanding of perceived complexity in programming competitions, suggesting practical applications in creating more balanced challenges and managing software projects, based on a new framework as a complexity metric.*

1. Introduction

Code metrics are diverse and depend on the context in which they are applied. In the literature reviews conducted by Ardito et al. (2020) and Vogel et al. (2020), more than a hundred code indicators are presented, each with a wide variety of applications. However, the most frequently cited indicators in the literature are Cyclomatic Complexity (CC) and the number of Lines of Code (LoC).

According to Elnaffar (2016), programming is a mental process that involves structuring code in some manner, regardless of the final quality produced. The complexity of this structure reflects the cognitive effort (difficulty) required to solve the problem. In his work, Elnaffar (2016), used five code metrics: (i) McCabe's Cyclomatic Complexity [McCabe 1976]; (ii) Average Depth of Nested Blocks; (iii) the total number of statements, which varies from one programming language to another; (iv) the total number of operators; (v) the number of unique operators. These five metrics were combined to form a Real Difficulty Index (RDI), proposed by the author.

According to the literature review by Usman et al. (2014), effort estimation is a fundamental part of software project management. This activity is essential for project planning and control as it involves forecasting the time and resources needed to complete various tasks. There is a vast amount of research dedicated to effort estimation, encompassing models, techniques, methods, and tools developed to improve the accuracy of these estimates. Like other software engineering activities, effort estimation is performed within the context of a structured software development

process, which provides guidelines and standards for the effective application of these various approaches. In this context, an indicator of coding complexity and work effort is a crucial tool in the software project management process.

Kasto and Whalley (2013) sought to explore the relationship between code metrics and student performance, using dynamic metrics along with cyclomatic complexity (CC) and average block depth. They found a significant correlation between these metrics and the difficulty of the questions.

The RSM Wizard (Resource Standard Metrics - RSM) software [M Squared Technologies LLC. 2024] is a graphical interface program that incorporates various classic code metrics and supports multiple programming languages, such as C/C++. In the work of Pighin and Marzona (2003), the RSM Wizard was used to measure cyclomatic complexity (CC) and examine its relationship with the number of failures in the first version of code during a software release. The conclusion from the correlation tests was that more indicators and different complexity measures are necessary before any robust correlation can be established between complexity and the persistence of defects in faulty files.

In the context of programming marathons or competitions, discussions about the difficulty of the questions often arise among participants. According to Elnaffar (2016), an intuitive approach to evaluating the difficulty of a programming question is to treat it as a classification problem. This involves categorizing the difficulty of a coding question based on the provided solution (answer key).

In seeking data for measuring code complexity, the SBC Programming Marathon [*Sociedade Brasileira de Computação* 2024] and the Brazilian Computing Olympiad [*Olimpíada Brasileira de Informática* – OBI 2024] were identified. The SBC Programming Marathon provides statistical data on the correct answers of each team on its website but does not provide answer keys for applying classic coding metrics. On the other hand, the OBI does not present statistical data on competitors but provides problem statements and reference codes (answer keys). Therefore, the OBI data was selected to form a dataset that allows statistical testing with the metrics found in the literature.

The Brazilian Computing Society (*Sociedade Brasileira de Computação* - SBC) promotes the Brazilian Computing Olympiad (OBI), a competition similar to other Brazilian scientific Olympiads such as Astronomy, Physics, and Mathematics. According to the official OBI website [*Olimpíada Brasileira de Informática* – OBI 2022], the main goal of the event is to spark students' interest in computer science, a crucial field in contemporary basic education, through challenges with a healthy dose of competition. The OBI has two categories, Programming and Introduction, each with different levels. In all categories and levels, the activities (exams) are conducted individually.

Based on our knowledge of code metrics, reinforced by the literature reviews of Ardito et al. (2020) and Vogel et al. (2020), there are no generic metrics for measuring code complexity, especially concerning the measurement of team work effort. In the research presented by Oliveira, Santos, and Pilatti (2024), a framework for measuring Explicit Intellectual Activities (EIA) was proposed. This framework starts with measuring complexity through an approximation (IIEa) that uses the Complexity Typology (TC) of Sheard and Mostashari (2010) and culminates in the average work

effort of the team that conducted the research (IIE0). However, the proof of concept (PoC) presented by Oliveira, Santos, and Pilatti, (2024) was applied to patents developed by public servants, requiring, among other items, the variables (i) development time of the research and (ii) the number of researchers involved in the work.

This work aims to analyze whether there is a significant correlation between the complexity measured using the Index of Internal Effort (IIE) framework [Mensures 2016] and the coding difficulty of the problems proposed by the organizers of the 2022 Brazilian Computing Olympiad (OBI). This analysis can contribute to academic programming events by investigating possible metrics that committees can use in their events to regulate the difficulty of the questions. Additionally, code metrics can be used in the workplace to assist in managing tasks, both in software development and in code maintenance.

This article is structured as follows: in the first section, the study topic is introduced. In the second section, the methodology used to create the dataset for the proof of concept (PoC) is detailed, along with the applied code metrics. The third section presents the obtained statistical results and their main characteristics. The fourth section discusses the achieved results, highlighting the strengths and weaknesses of the study, as well as the main conclusions reached. Finally, the fifth section provides final recommendations and general conclusions about the study conducted.

2. Method

After searching for the construction of a dataset containing codes suitable for applying classic metrics from the literature, freely available data from the OBI 2022 website [*Olimpíada Brasileira de Informática* – OBI 2022], were selected. According to the official site, the Programming category exams are conducted in three phases and cover students at four distinct levels: Junior Level (0), for students from any year of Elementary School; Level 1, for students from Elementary School up to the 1st year of High School; Level 2, for students from Elementary School up to the 3rd year of High School; and Senior Level (3), for students in the 4th year of Technical Education and the 1st year of a bachelor's degree program. The grouping of phases and levels creates a categorization that can be used as a reference for the difficulty proposed by the event organizers, making it possible to use this information as the main reference variable for the tests.

During the creation of the dataset, 46 codes were found in OBI 2022 website, with difficulty levels ranging from 1 to 6. The dataset for this article, including the statistical tests, is available via a link at the end of this article.

The metrics used in the correlation tests are: (i) code difficulty based on the selection of questions by the organizers of OBI 2022 (Diff), formed by the simple sum of the Olympiad phase (1, 2, 3) and student level (0, 1, 2, 3); (ii) metrics provided by the RSM Wizard software (2024), including: Lines of Code (LoC), Effective Lines of Code (eLoc), Logical Lines of Code (1LoC), Cyclomatic Complexity (CC) as per [McCabe 1976], Function Points for LoC (FP), and Total Function Complexity (TFC); (iii) IIEa, a generic metric for measuring explicit intellectual activities, according to the parameters proposed in the work of Oliveira and Pilatti (2021).

The Index of Internal Effort (IIE) is a generic tool, but when specifically applied to evaluate code complexity, it is crucial to identify the most relevant variables in the code, similar to the process of “System Identification” recurring in control engineering [Ljung 1999]. To achieve this, variable groups published in Oliveira and Pilatti (2021) work, which addresses code metrics on the Arduino platform, were selected. The four variable groups (Gn) were organized in reverse order compared to what is typically taught in programming classes [Oliveira and Pilatti 2021]. Normally, in classes, simpler concepts and functions are presented first, followed by more complex ones. The following describes the groups used in this study:

- G1: Iterations and their controls: For, While, Do While, Continue, Goto;
- G2: Conditional statements and their controls: IF-Else, Switch/Case, Break;
- G3: Relational and logical operators: &&, ||, !, <, <=, ==, >=, >, !=;
- G4: Code organizers and structures: braces {, brackets [, parentheses (;

The variables selected to form the Gn groups respect the Complexity Typology (TC) from Sheard and Mostashari (2010), making clear the difference between the items that form each group. With the selection of variable groups forming the study object, namely the C codes organized into groups G1, G2, G3, and G4, the next step is to count the occurrences of each element. These counted values are then processed according to the methodology of IIEa [Oliveira and Pilatti 2021] to obtain the complexity of each code. Equation 1 presents the complexity calculation method by IIE, known as IIEa because it is an approximation of complexity.

$$\text{IIEa} = 1 + \sum_{n>3} \sqrt{G_n} = 1 + \sqrt{G_1} + \sqrt{G_2} + \sqrt{G_3} + \sqrt{G_4} \quad (1)$$

To compare the code difficulty proposed by OBI 2022 with code metrics found in the literature, a Spearman correlation test (a non-parametric test) will be conducted using an "all against all" matrix structure. This technique allows investigating the possible relationships between all indicators, resulting in three main scenarios: (i) a significant and positive correlation, indicating that two variables or metrics are directly connected or associated by a third variable; (ii) a significant and negative correlation, where variables grow in an inversely proportional manner; and (iii) absence of significant correlation between the indicators [Triola 2013].

In total, 10 metrics will be compared in the correlation test, with the main metrics being difficulty (Diff) and IIEa complexity. These metrics are not classical in software literature, with the former originating from the group that manages OBI and the latter being a generic framework for measuring explicit intellectual activities (EIA).

3. Results

Table 1 presents the results of the correlation test among the 10 selected metrics for the research. The complete file with all collected data is made available by the author at the end of this article.

The values highlighted in Table 1 are those that showed a significant and positive correlation in the Spearman test. For this test, only cases with a confidence level of 99% were considered, where the correlation value must be equal to or greater than 0.384 for a sample size of 46 codes [Triola 2013].

The reference variable due to dataset characteristics is the difficulty of the questions (Diff). This variable showed a significant and positive correlation with the phase and level of the questions in OBI 2022. It is important to note that the phases did not show a significant correlation with the level of questions, which can be explained by the fact that some questions were duplicated across different levels or phases. However, each exam consisted of three or more questions, highlighting the importance of combining levels and phases to form the difficulty variable (Diff).

Furthermore, the difficulty of the questions proposed by the organizers of OBI 2022 (Diff) showed a significant and positive correlation with three metrics: lines of code (LoC), function points (FP), and the approximation of complexity (IIEa). The latter metric comes from a generic framework for measuring explicit intellectual activities (EIA), while the first two are specific metrics in the software field.

Table 1. Correlation Test Among Code Metrics

	Phase	Level	Diff	LoC	eLoC	1LoC	CC	FP	TFC	IIEa
Phase	**1,000**	-0,012	**0,575**	**0,428**	**0,412**	0,288	**0,392**	**0,503**	**0,423**	**0,457**
Level		**1,000**	**0,801**	0,171	0,138	0,202	0,068	0,206	0,043	0,154
Diff			**1,000**	**0,402**	0,361	0,343	0,295	**0,475**	0,298	**0,400**
LoC				**1,000**	**0,984**	**0,956**	**0,893**	**0,952**	**0,857**	**0,937**
eLoC					**1,000**	**0,956**	**0,884**	**0,943**	**0,840**	**0,936**
1LoC						**1,000**	**0,863**	**0,915**	**0,815**	**0,916**
CC							**1,000**	**0,884**	**0,974**	**0,930**
FP								**1,000**	**0,863**	**0,951**
TFC									**1,000**	**0,895**
IIEa										**1,000**

The research hypothesis investigates the possible significant positive correlation among three distinct elements: (i) the difficulty defined by the organizers of OBI 2022, (ii) classical code metrics, and (iii) the approximation of code complexity through the generic measurement framework for explicit intellectual activities (EIA), called the Index of Internal Effort (IIE) [Mensures 2016]. With this dataset, the results were positive, allowing us to infer that the difficulty attributed by the organizers of OBI 2022 aligns with classical code metrics, indicating that both LoC and FP effectively reflect the complexity perceived by specialists.

Additionally, the Index of Internal Effort (IIE) showed a consistent correlation with classical metrics and the perceived difficulty by the organizers of OBI 2022, thus not refuting the possibility of using a generic framework when assessing code complexity. This correlation is crucial as it does not invalidate the use of IIE as an indicator of complexity applicable generically to various Explicit Intellectual Activities (EIA), besides offering an additional metric for software project management and evaluation.

The application of these metrics together not only facilitates a more accurate analysis of code complexity in C but also provides a multifaceted approach that considers different aspects of difficulty and effort involved in programming. Therefore, the positive correlation found among these variables reinforces the importance of using multiple metrics to gain a more comprehensive and accurate understanding of software project complexity, contributing to better software development and management practices.

4. Discussion

In this study, codes in the C programming language from the Brazilian Olympiad in Informatics 2022 were used as a database to conduct a proof of concept (PoC) on the applicability of IIEa, allowing for comparison of this generic indicator with human data and classical coding metrics from the literature.

The difficulty of the codes (Diff), as defined by the organizers of OBI 2022, showed a significant and positive correlation with classical coding indicators such as lines of code (LoC) and function points (FP). Additionally, the complexity approximated by IIEa showed a significant correlation with the metric Diff.

Specifically, a significant internal correlation was found among the metrics of the RSM Wizard software (2024), including effective lines of code (eLoc), logic lines of code (1LoC), and total function complexity (TFC). The metrics from RSM Wizard also showed significant correlation with classical metrics LoC, CC, and FP, but did not show significant correlation with the difficulty defined by the organizers of OBI 2022 (Diff).

The Index of Internal Effort (IIE), in its approximation of complexity (IIEa), created to measure Explicit Intellectual Activities (EIAs) within the Brazilian public service, had its applicability not refuted in this proof of concept (PoC). This result is important to validate a generic and empirical methodology, aligned with the principles of modern science as advocated by the philosophy of modern science [Popper 2001].

In a previous study conducted on Arduino platform programming [Oliveira and Pilatti 2021], code complexity was measured using IIEa, finding a positive and significant correlation with metrics (i) CC (cyclomatic complexity), (ii) LoC (lines of code), and (iii) evaluation of students in a technical programming course. In the present study, using a non-similar dataset, IIEa demonstrated significant correlation not only with classical software metrics, but also with (iv) problems difficulty and its respective coding, as attributed by the event organizers. This finding addresses a gap identified in the tests of Oliveira and Pilatti (2021), which included student opinions but lacked input from programming teachers and experts.

By hypothesis, in academic exams, Olympiads, marathons, Hackathons, and similar events, it would be feasible to modify the current process with fixed questions, allowing organizers to provide multiple questions to participants, using IIEa as a reference for scoring each one. This would enable competitors to freely choose which and how many questions to tackle to try to win the competition. IIEa has emerged as the most suitable indicator for this context because it is less limited to a specific test condition. This is supported by the significant number of correlations of IIEa found in the tests and presented in Table 1.

The IIE was initially developed to measure EIAs carried out by public servants, with the aim of improving the efficiency and effectiveness of services provided to society [Mensures 2016]. Hypothetically, its application may have limitations or non-linearities in certain EIAs. For this reason, IIEa is being tested with various distinct datasets to explore these potential limitations. Examples include codes from the Arduino platform [Oliveira and Pilatti 2021], patents filed by public service [Oliveira *et al.* 2023a, Oliveira *et al.* 2023b], complexity of postgraduate program books evaluated by CAPES [Oliveira and Pinto 2024], among others.

In the research by Ferreira et al. (2016), the authors emphasize the importance of studies on "truck factor" in software development project management. This metric assesses the risk associated with team members' absence, which can interrupt or significantly compromise project progress due to the loss of knowledge concentrated in just a few developers. To mitigate this issue, a recommended practice is to regularly conduct code reviews, enabling more team members to become familiar with different parts of the project. In this context, IIEa can be used to measure the work delivered by each collaborator, thereby balancing the workload within the development team and minimizing the "truck factor."

The cyclomatic complexity (CC) indicator [McCabe 1976] did not show a significant correlation with the reference variable (Diff). Hypothetically, CC is an indicator limited to the quantity of conditional branches and loops in the code, whereas the questions selected by the organizers of OBI 2022 go beyond this limitation. In this context, a multi-criteria indicator like IIEa has advantages over other restrictive metrics. Within the application rules of IIEa indicated by Oliveira, Santos, and Pilatti (2024) are: the use of at least 4 distinct variables or groups of variables, and the use of the square root as an element to compress errors in collected data. These two conditions increase the robustness of IIEa compared to other metrics formed by a single group of variables.

A limitation of the dataset to be reported is the size of the codes. Since the dataset consists of problems from the Brazilian Olympiad in Informatics 2022, each code is produced by a single participant within a specific time limit, which restricts the size and complexity of the codes used. In future work, it is crucial to repeat the tests using considerably larger codes developed by teams of developers. In these future tests, variables such as (i) development time and (ii) number of developers can be incorporated to measure the average workload, as suggested in the research on patents deposited by the public service by Oliveira, Santos, and Pilatti (2024).

This study provides two significant contributions, one internal and one external. Internally, it contributes to the software field by introducing a new coding metric to the community. However, due to the lack of data on the number of participants and development time, the IIE was presented partially, calculating only the approximation of complexity (IIEa), while the average team effort [Oliveira, Santos and Pilatti 2024] was not addressed. Externally, this study contributes to the improvement of public service, as the IIE aims to enhance the performance evaluation of public servants through a multidisciplinary framework for measuring tasks performed. However, due to its generic nature, more tests in diverse areas and different datasets are needed to identify possible limitations in its use.

5. Conclusions

This study explored the complexity of codes from the Brazilian Olympiad in Informatics 2022 (OBI 2022) through the application of classical metrics and the Index of Internal Effort (IIE). The results highlight that the difficulty of the questions, as defined by the event organizers (Diff), correlates positively with metrics such as lines of code (LoC), function points (FP), and the complexity approximation by IIEa. These findings confirm that both traditional metrics and IIEa are effective in assessing perceived complexity and effort involved in programming for small codes.

This study contributes significantly to software research by presenting IIEa as a viable generic metric, while also validating the utility of classical metrics in programming competitions. Practical implications include applying these metrics to create more balanced questions in academic competitions, allowing participants to choose challenges aligned with their skill levels.

Limitations of the dataset include the limited size of the studied codes and the need to explore additional metrics in more complex software development contexts. Future research should expand the study of IIE to include codes in other programming languages and variables such as development time and number of developers. The potential influence of these results on the scientific community lies in the possibility of revising complexity assessment practices and developing new measurement models.

References

Ardito, L., Coppola, R., Barbato, L. and Verga, D. (2020) “A Tool-Based Perspective on Software Code Maintainability Metrics: A Systematic Literature Review”, *Scientific Programming*, 2020, 1–26. DOI: 10.1155/2020/8840389

Elnaffar, S. (2016) “Using software metrics to predict the difficulty of code writing questions”. *2016 IEEE Global Engineering Education Conference* (EDUCON). doi:10.1109/educon.2016.7474601

Ferreira, M., Avelino, G., Valente, M., & Ferreira, K. (2016) “A Comparative Study of Algorithms for Estimating Truck Factor”. In *Anais do X Simpósio Brasileiro de Componentes, Arquiteturas e Reutilização de Software*, (pp. 91-100). Porto Alegre: SBC.

Kasto, N., and Whalley, J. (2013) “Measuring the difficulty of code comprehension tasks using software metrics”, *Proc of the 15th Australasian Computer Education Conference* (ACE 2013), Adelaide, Australia, 57-6, 2013.

Ljung, L. (1999). System identification: theory for the user. 2nd ed. Upper Saddle River: *Prentice Hall PTR*, xxii, 609 p. (Prentice-Hall information and system sciences series). ISBN 0-13-656695-2.

M Squared Technologies LLC. (2024) “RSM Wizard”. Versão 7.75. Disponível em: < https://msquaredtechnologies.com/RSM-Wizard.html >. Acesso em 01 mai. 2024.

McCabe, T. (1976) “A Software Complexity Measure”. *IEEE Transactions on Software Engineering*, SE-2(4):308-320.

Mensures®. (2016) Software desenvolvido para mensurar atividades intelectuais: baseado na metodologia do Índice Interno de Esforço. INPI. Brasil. BR 51 2016 001521-7. RPI 2405.

Olimpíada Brasileira de Informática – OBI. (2022). OBI 2022 Provas e Gabaritos, Fase 1, Modalidade Programação. Disponível em: < https://olimpiada.ic.unicamp.br/passadas/OBI2022/fase1/programacao/ >. Acesso em: 01 mai. 2024.

Olimpíada Brasileira de Informática. (2024) Sobre a OBI 2024. Disponível em: < https://olimpiada.ic.unicamp.br/info/ >. Acesso em 01 abr. 2024.

Oliveira A. A., and Pilatti, L. A. (2021) “Mensuração da complexidade de códigos em C com o método do Índice Interno de Esforço”. In: Anais do XII Encontro Anual de Tecnologia da Informação – EATI. Ano 10, n. 2; Novembro/2021. Disponível em: < http://anais.eati.info:8080/index.php/2019/article/view/64/61 >.

Oliveira A. A., Santos, C. B., and Pilatti, A. P. (2024) “Bridging the gap in patent assessment: The Index of Internal Effort framework for pharma innovations”. | [Salvando las distancias en la evaluación de patentes: El marco del Índice de Esfuerzo Interno para las innovaciones farmacéuticas]. *J Pharm Pharmacogn Res* 12(5): 852-869. https://doi.org/10.56499/jppres23.1859_12.5.852. Retrieved from: https://jppres.com/jppres/pdf/vol12/jppres23.1859_12.5.852.pdf

Oliveira, A. A., and Pinto, L. R. (2024) “Modernização na avaliação de livros: possíveis melhorias na eficiência do serviço público”. In: VI Simpósio de Economia e Gestão da Lusofonia. No prelo.

Oliveira, A. A., Fung, C. W. H., Burkarter, E., Pilatti, L. A., and Santos, C.B. (2023) “Metrificação de patentes: Uma análise entre qualidade, complexidade e esforço”. In: *Encontro Nacional de Engenharia de Produção*, 2023, Ceará. XLIII ENEGEP. DOI: 10.14488/enegep2023_tn_st_404_1989_45333. Disponível em: < https://www.abepro.org.br/biblioteca/TN_ST_404_1989_45333.pdf >.

Oliveira, A. A., Guimarães, T. A., Ávila, C. A., Pilatti, L. A., and Santos, C. B. (2023) “Metrificação de patentes desenvolvidas no serviço público: Um estudo acerca do índice interno de esforço”. In: Encontro Nacional de Engenharia de Produção, 2023, Ceará. XLIII ENEGEP. DOI: 10.14488/enegep2023_tn_wpg_404_1986_45338. Disponível em: < https://www.abepro.org.br/biblioteca/TN_WPG_404_1986_45338.pdf >.

Pighin, M., ans Marzona, A. (2003) “An empirical analysis of fault persistence through software releases”. *2003 International Symposium on Empirical Software Engineering*, ISESE 2003. Proceedings. doi:10.1109/isese.2003.1237979

Popper, K. R. (2001) A lógica da pesquisa científica. 9. ed. São Paulo: *Cultrix*, 567 p. ISBN 85-316-0236-X.

Sheard S. A., and Mostashari, A. (2010) “A complexity typology for systems engineering”. *INCOSE Int Symp* 20(1): 933–945. https://doi.org/10.1002/j.2334-5837.2010.tb01115.x

Sociedade Brasileira de Computação. (2024) Maratona SBC de Programação. Disponível em: < https://maratona.sbc.org.br/ >. Acesso em 01 abr. 2024.

Triola, M. F. (2013) Introdução à estatística: atualização da tecnologia. Rio de Janeiro, *LTC*, xxviii, 707 p.

Usman, M., Mendes, E., Weidt, F., and Britto, R. (2014) “Effort estimation in agile software development”. *Proceedings of the 10th International Conference on Predictive Models in Software Engineering* - PROMISE ’14. doi:10.1145/2639490.2639503

Vogel, M., Knapik, P., Cohrs, M. Szyperrek, B., Pueschel, W., Etzel, H., and Kuhrmann, M. (2020) “Metrics in automotive software development: A systematic

literature review". *Journal of Software: Evolution and Process*, 33(2). DOI: 10.1002/smr.2296.

Appendices

Link to access the article dataset:

https://docs.google.com/spreadsheets/d/1BVZ3qGkAthi9E9L_ND5-MKLKVWI-aPTB/edit?usp=sharing&ouid=108894340029492345033&rtpof=true&sd=true