Universidade Federal da Bahia
Instituto de Matemática

Programa de Pós-graduação em Ciência da Computação

CARACTERIZAÇÃO DA COMPLEXIDADE ESTRUTURAL EM FERRAMENTAS DE ANÁLISE ESTÁTICA DE CÓDIGO-FONTE

Joenio Marques da Costa
joenio@joenio.me

QUALIFICAÇÃO DE MESTRADO

Salvador
08 de Julho de 2016

Universidade Federal da Bahia
Instituto de Matemática

Joenio Marques da Costa
joenio@joenio.me

CARACTERIZAÇÃO DA COMPLEXIDADE ESTRUTURAL EM FERRAMENTAS DE ANÁLISE ESTÁTICA DE CÓDIGO-FONTE

Trabalho apresentado ao Programa de Pós-graduação em Ciência da Computação do Instituto de Matemática da Universidade Federal da Bahia como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Christina von Flach G. Chavez
Co-orientador: Prof. Dr. Paulo Roberto Miranda Meirelles

Salvador
08 de Julho de 2016

AGRADECIMENTOS

(pendente)

RESUMO

(pendente)

Palavras-chave:
(pendente)

ABSTRACT

(pendente)

Keywords:
(pendente)

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

CAPÍTULO 1

INTRODUÇÃO

(pendente)

1.1 OBJETIVOS

O objetivo principal deste trabalho é compreender as ferramentas de software para análise estática de código-fonte, do ponto de vista de sua manutenabilidade, a partir da análise de sua complexidade estrutural, discutindo quais características arquiteturais explicam seus atributos de qualidade interna.

Objetivos específicos:

- Realizar uma revisão da literatura para caraterizar a arquitetura das ferramentas de análise estática de código-fonte.
- Realizar uma revisão estruturada, com base em artigos publicados em conferências relacionadas, para selecionar e obter código-fonte de ferramentas de análise estática (explicadas na Seção 3.3.2), para coletar suas métricas de código-fonte.
- Selecionar e obter o código-fonte de ferramentas de análise estática desenvolvidas pelo indústria (explicadas na Seção 3.4.2), para coletar suas métricas de código-fonte.
- Propor intervalos de referência para a observação parametrizada da qualidade interna das ferramentas de análise estática, a partir de suas métricas de código-fonte.
- Cálculo da distância Bayesiana entre valores de referência e os valores das ferramentas estudadas.
- Evoluir uma ferramenta de análise de código-fonte para coleta de métricas de código-fonte (detalhado na Seção 4.4).
- Um conjunto de estratégias para a manutenção e evolução de ferramentas de análise estática de código-fonte.

1.2 CONTRIBUIÇÕES ESPERADAS

CC1: Um conjunto de intervalos de referência da frequência dos valores de métricas de código-fonte para o domínio de ferramentas de análise estática

CC2: Explicação da alta complexidade estrutural em ferramentas de análise de código-fonte

CT1: Evolução de uma ferramenta de análise de código-fonte

CAPÍTULO 2

ANÁLISE ESTÁTICA DE CÓDIGO-FONTE

Análise estática de código-fonte é um ramo da engenharia de software voltado a obter informações acerca de um programa a partir do seu código-fonte sem necessidade de execução, e sem requerer qualquer outro artefato do programa além do próprio código. É considerada uma atividade meio com objetivo de suportar uma variedade de tarefas comuns da engenharia de software, muitas dessas tarefas são substancialmente úteis em atividades de manutenção, Binkley (2007) define uma extensa lista dessas atividades, segue abaixo um sub-conjunto delas.

- recuperação arquiterural
- detecção de clone
- compreensão
- pesquisa em engenharia de software empírica
- localizaçao de falhas
- desenvolvimento baseado em modelos
- análise de performance
- evolução de software
- garantia de qualidade
- engenharia reversa
- manutenção de software
- versionamento de software
- testes
- ferramentas e ambientes
- desenvolvimento de aplicações web

Seja em qual atividade for, a análise estática possui uma importância significativa, pois ao ser capaz de extrair informações diretamente do código-fonte de um programa pode auxiliar a responder com precisão a seguinte pergunta: "O que é que este programa significa?"

2.1 ANATOMIA DA ANÁLISE DE CÓDIGO-FONTE

Cruz, Henriques e Pinto (2009), Binkley (2007) realizaram um estudo onde definiram a estrutura comum da análise de código-fonte em três componentes: a) extração de dados, b) representação interna, e c) análise da representação interna. Esta mesma estrutura é também apresentada na documentação da ferramenta de análise de código-fonte CodeSonar[1], reproduzida aqui na Figura 2.1.

As seções à seguir detalham cada um destes componentes.

[1] http://www.grammatech.com/products/source-code-analysis

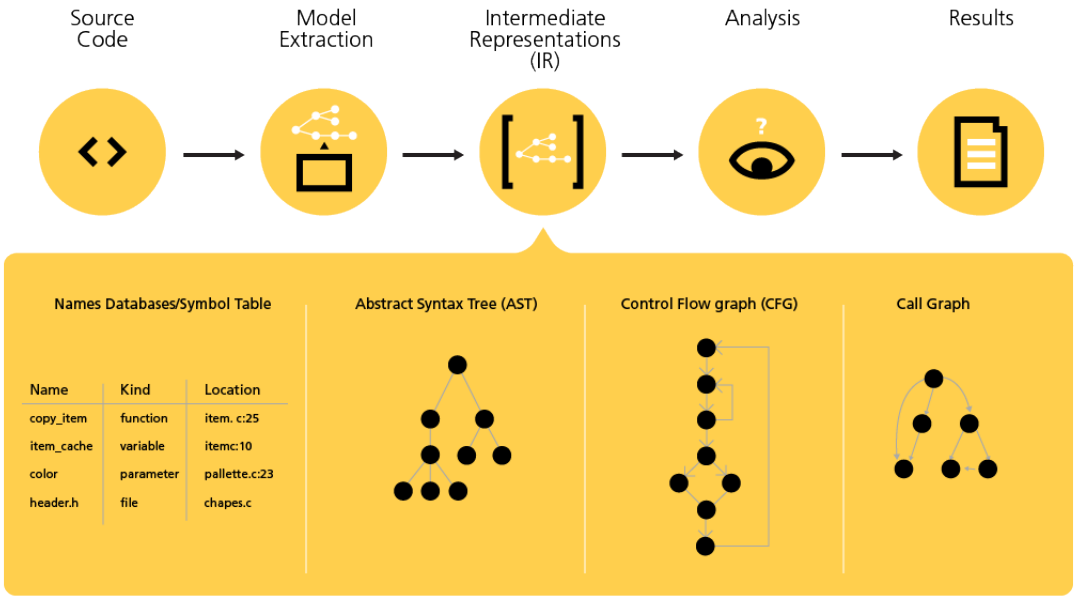

Figura 2.1 CodeSonar: Static Analysis Representation

2.1.1 Extração de dados

O processo de recuperar dados para futuro processamento ou armazenamento é chamado de extração de dados, exportar estes dados em uma representação intermadiária é uma estratégia comum para facilitar análise e transformação de dados e possivelmente adição de metadados.

O primeiro componente da análise de código-fonte é a extração de dados, responsável por processar o código-fonte e transformar em uma ou mais representações internas. Em essência este componente converte a sintaxe de um programa em uma outra sintaxe abstrata e mais adequada para análise posterior.

Este componente é usualmente chamado de analisador sintático (ou *parser*) e a tarefa realizada por ele é considerada como um mal necessário da análise de código-fonte. Apesar de teoricamente não ser uma tarefa difícil, a complexidade das linguagens de programação modernas podem restringir significativamente a análise de código-fonte.

2.1.2 Representação interna

Os dados obtidos na extração precisam ser representados em um formato mais abstrato, esta é a responsabilidade do segundo componente da análise de código-fonte: armazenar os dados coletados usando uma representação interna num formato mais adequado para análise automática, o principal papel deste componente é abstrair aspectos particulares do programa.

Muitas das representações internas existentes tem seu surgimento na área de compiladores e algumas delas são produzidas diretamente pelo *parser* enquanto outras requerem uma análise específica. Os formatos mais comuns são baseados em grafos, dentre os

quais o mais amplamente utilizado é Control Flow Graph (CFG), ou Call Graph. Outros formatos conhecidos são:

- Identifiers Table
- Abstract Syntax Tree (AST)
- Decorated Abstract Syntax Tree (AST)
- Control Flow Graph (CFG)
- Value Dependence Graph (VDG)
- Call Graph
- Module Dependence Graph (MDG)
- Trace Flow Graph (TFG)
- Static Single Assignment (SSA)

Estas representações podem ser utilizadas tanto na análise estática quanto na análise dinâmica, o uso de um ou outro formato depende do tipo de análise e seu propósito. Numa aplicação real é comum combinar diferentes tipos no sentido de enriquecer e estruturar a informação extraída.

2.1.3 Análise da representação interna

Este terceiro componente da análise de código é responsável por realizar inferências a partir das informações representadas internamente, este processo requer que as informações armazenadas estejam interconectadas e também interrelacionadas com conhecimento anterior. Esta análise pode gerar conhecimento quantitativo ou qualitativo, como por exemplo métricas de software ou mineração de dados, respectivamente. Técnicas de visualização são cruciais para a efetividade deste processo.

É neste componente que acontece a análise propriamente dita, e ele pode ser classificado em seis dimensões básicas: estática versus dinâmica, sound versus unsound, segura vesus insegura, sensível ao fluxo versus insensível ao fluxo, sensível ao contexto versus insensível ao contexto, e, em relação à sua complexidade.

Estas características da análise vão afetar o resultado gerado quanto à confiabiliade, completude, garantia de precisão, eficiência, performance, complexidade computacional, precisão da informaçao, entre outras.

2.2 MÉTRICAS DE CÓDIGO-FONTE

Uma métrica, segundo a definição da ISO/IEC 25010 (ISO, 2011), é a composição de procedimentos para a definição de escalas e métodos para medidas, em engenharia de software estas métricas podem ser classificadas em três categorias: métricas de produto, métricas de processo e métricas de projeto.

Métricas de produto são aquelas que descrevem as características de artefatos do desenvolvimento, como documentos, diagramas, código-fonte e arquivos binários. Métricas de processo medem atributos relacionados ao ciclo de desenvolvimento do software. Métricas de projeto são aquelas que descrevem as características dos recursos disponíveis ao desenvolvimento.

As métricas de produto podem ser classificadas entre internas ou externas, ou seja,

aquelas que medem propriedades visíveis apenas aos desenvolvedores ou que medem propriedades visíveis aos usuários, respectivamente.

As propriedades visíveis aos desenvolvedores podem ser medidas através de métricas de código-fonte e a sua observação pode indicar aspectos relevantes à manutenibilidade de um programa. Dentre as inúmeras métricas de código-fonte iremos destacar aquelas utilizadas no estudo de Meirelles (2013) onde associou-se características de qualidade de produto de software à características de qualidade de código-fonte através da observação de suas métricas.

- **ACC** *Aferent Connections per Class (Conexões aferentes de uma classe)*: Mede a conectividade de uma classe.
- **ACCM** *Average Cyclomatic Complexity per Method (Média de complexidade ciclomática por método)*: mede a complexidade do programa (MCCABE, 1976).
- **AMLOC** *Average Method LOC (Média do número de linhas de código por método)*: indica se o código está bem distribuído entre os métodos, quanto maior mais "pesados" são os métodos (??)
- **ANPM** *Average Number of Parameters per Method (Média do Número de Parâmetros por Método)*: calcula a média de parâmetros dos métodos da classe (Bansiya e Davi, 1997).
- **CBO** *Coupling Between Objects (Acoplamento entre objetos)*: mede o acoplamento entre objetos do software (CHIDAMBER; KEMERER, 1994).
- **DIT** *Depth of Inheritance Tree (Profundidade da árvore de herança)*: mede o número de ancestrais de uma classe (SHIH et al., 1997).
- **LCOM4** *Lack of Cohesion in Methods (Ausência de coesão em métodos)*: mede o grau de falta de coesão em métodos (HITZ; MONTAZERI, 1995).
- **LOC** *Lines of Code (Número de linhas de código)*: mede o número de linhas excluindo linhas em branco e comentários.
- **NOA** *Number of Attributes (Número de atributos)*: calcula o número de atributos de uma classe
- **NOC** *Number Of Children (Número de filhos)*: número total de filhos de uma classe (ROSENBERG; HYATT, 1997).
- **NOM** *Number of Methods (Número de métodos)*: mede o tamanho das classes em termos das suas operações implementadas.
- **NPA** *Number of Public Attributes (Número de atributos públicos)*: mede o encapsulamento.

- **NPM** *Number of Public Methods (Número de métodos públicos)*: representa o tamanho da "interface" da classe.
- **RFC** *Response For a Class (Resposta para uma classe)*: número de métodos dentre todos os métodos que podem ser invocados em resposta a uma mensagem enviada por um objeto de uma classe (SHARBLE; COHEN, 1993).
- **SC** *Structural Complexity (Complexidade estrutural)*: mede a complexidade do software (DARCY et al., 2005).

2.2.1 Complexidade estrutural

Complexidade estrutural (SC) é uma medida calculada através da combinação das métricas de acoplamento (CBO) e coesão (LCOM4), esta medida é, possivelmente, um indicativo de problemas na manutenibilidade de sistemas de software, em especial sobre o esforço necessário para atividades de manutenção (TERCEIRO, 2012). Ela está relacionada à como os módulos de um programa estão organizados entre sí bem como à estrutura interna de cada módulo.

Esta métrica pode dar indícios importantes sobre características arquiteturais de um programa de software, e podem explicar seus atributos de qualidade interna.

2.3 ANALIZO

Analizo[2] é um *toolkit* livre, multi-linguagem e extensível para análise de código-fonte, calcula uma grande quantidade de métricas, como CBO, LCOM4, RFC, LOC, e suporta análise das linguagens de programação C, C++ e Java.

É uma ferramenta mantida constantemente, com desenvolvedores ativos, e atualizações frequentes, sua última versão 1.19.0 foi lançada em 18 de Fevereiro de 2016 e será esta a versão utilizada neste estudo.

[2]http://analizo.org

CAPÍTULO 3

METODOLOGIA

Visando então caracterizar as ferramentas de análise estática de código-fonte será feito um levantamento de artigos com publicação de ferramentas deste domínio e uma busca por ferramentas de análise estática na indústria.

Este levantamento resultará num conjunto de ferramentas de análise estática de código-fonte, estas ferramentas serão caracterizadas através da análise de suas métricas de qualidade interna e darão valores de referência de métricas para ferramentas deste domínio.

A avaliação destes valores juntamente com as características arquiteturais encontradas darão origem a recomendações de refatoração para ferramentas de análise estática de código-fonte.

3.1 TRABALHOS RELACIONADOS

(pendente)

3.2 HIPÓTESES

Para guiar os estudos, conforme os objetivos acima, definimos as seguintes hipóteses:

H1: *É possível calcular valores de referência de métricas de código-fonte para ferramentas de análise estática a partir de um conjunto de softwares da academia e da indústria*

H2: *Ferramentas de análise estática tendem a ter uma maior complexidade estrutural do que ferramentas de outros domínios de aplicação*

H3: *Dentre as ferramentas de análise estática de código-fonte, aquelas desenvolvidas na indústria apresentam uma menor complexidade estrutural*

A hipótese **H1** será validada a partir da análise das métricas calculadas para cada uma das ferramentas estudadas, esta análise levará em consideração a caracterização das ferramentas (explicados na Seção 3.5.3), em especial um subconjunto das ferramentas com melhores valores de métricas.

A hipótese **H2** será validada a partir da comparação com os trabalhos relacionados (Seção 3.1), onde realizou-se estudos similares com cálculo e distribuição de métricas utilizando conjuntos de softwares e ferramentas distintos dos estudados aqui.

A hipótese **H3** será validada a partir do cálculo da distância das métricas de cada ferramenta com os valores de referências encontrados neste estudo (explicados na Seção 3.5.1).

3.3 PLANEJAMENTO DO ESTUDO

3.3.1 Seleção de ferramentas de análise estática

Iremos realizar uma busca por ferramentas de análise estática desenvolvidas no contexto da academia e da indústria, para isso, será feito um planejamento detalhado para realizar a seleção de ferramentas em cada um destes contextos.

No contexto acadêmico a busca por ferramentas será feita através de artigos publicados em conferências que tenham histórico de publicação sobre ferramentas de análise estática de código-fonte. Estes artigos serão analisados e aqueles com publicação de ferramenta serão selecionados.

Na indústria, a busca por ferramentas será feita a partir da base mantida pelo projeto SAMATE, um projeto do NIST dedicado ao desenvolvimento de métodos que permitam avaliar e medir a eficiência de ferramentas e técnicas sobre garantia de qualidade em software. O site do projeto mantém uma lista de ferramentas de análise estática.

Uma vez que as ferramentas tenham sido selecionadas iniciaremos a extração de seus atributos de qualidade interna a partir do cálculo de suas métricas de código-fonte.

3.3.2 Revisão estruturada

Chamamos de revisão estruturada um processo disciplinado para seleção de artigos a partir de critérios bem definidos de forma que seja possível a reprodução do estudo por parte de pesquisadores interessados.

A primeira etapa da revisão é definir onde os artigos serão encontrados, as fontes serão conferências com o tema de interesse ao estudo realizado, uma vez definido a origem é feito cópia local de todos os artigos em formato PDF.

A caracterização dos artigos se inicia por uma busca textual realizada através de um script[1] escrito especialmente para este estudo, esta busca seleciona os artigos através dos seguintes termos:

```
"tool" OU "framework"; E
"download" OU "available"; E
"http" OU "ftp"; E
"static analysis" OU "parser".
```

O segundo passo é realizado a partir de uma leitura do artigo a fim de identificar se realmente trata-se de um artigo com publicação de ferramenta de análise estática, uma vez que se confirme que o artigo publica uma ferramenta este artigo será selecionado, ferramentas que sejam mais abrangentes do que apenas análise estática mas que contenham esta função em seu conjunto também serão considerados.

Uma vez identificado os artigos que publicam ferramentas de análise estática, procuramos no próprio artigo referências de onde encontrar o software, neste momento algumas ações serão tomadas a partir da situação encontrada.

[1] http://github.com/joenio/dissertacao-ufba-2016/blob/master/revisao-estruturada/filter

- Aqueles autores que afirmam que a ferramenta está disponível mas o artigo não contém referências de onde encontrar o software serão contactados por email solicitando informações de onde obter o código-fonte.
- Os artigos que indicam onde obter o cófigo-fonte mas o acesso ao local indicado não está disponível, ou está disponível mas o software não se encontra lá, os autores também serão contactados solicitando informações atualizadas de onde obter uma cópia do código-fonte da ferramenta
- Os demais artigos que indicam onde obter o código-fonte e a referência está correta, iremos fazer download da última versão disponível do software

Uma vez que os autores contactados por email respondam com informações de onde obter o software iremos adicionar estes softwares no conjunto de softwares a serem analisados.

3.4 COLETA DE DADOS

Serão realizadas duas etapas para identificar e mapear as ferramentas de análise estática com código-fonte disponível, uma atividade relacionada ao levantamento de ferramentas da academia, descrita na Seção 3.4.1, outra atividade relacionada ao levantamento de ferramentas da indústria, descrita na Seção 3.4.2.

Uma vez definido o conjunto de ferramentas a serem estudados iremos identificar em qual linguagem de programação cada ferramenta é escrita, isto é necessário pois apenas as linguagens suportadas pela ferramenta Analizo serão selecionadas. Esta análise será feita pela ferramenta *sloccount*[2], uma ferramenta livre para contar linhas de código fonte e que identifica em qual linguagem de programação um projeto é escrito.

As ferramentas selecionadas serão analisadas com o Analizo para extração de suas métricas de código-fonte, esta análise utilizará o comando *metrics* da súite Analizo, ele calcula métricas globais de projeto e métricas por módulos, este estudo levará em consideração a distribuição das métricas por módulos.

3.4.1 Ferramentas da academia

A seleção de ferramentas da academia será relizada através da revisão estruturada dos artigos da conferência SCAM - Source Code Analysis and Manipulation Working Conference[3] e mais uma dentre as seguintes conferências:

- ASE - Automated Software Engineering[4]
- CSMR[5] - Conference on Software Maintenance and Reengineering[6]

[2]http://www.dwheeler.com/sloccount

[3]http://www.ieee-scam.org

[4]http://ase-conferences.org

[5]A conferência CSMR tornou-se SANER - Software Analysis, Evolution, and Reengineering a partir da edição 2015.

[6]http://ansymore.uantwerpen.be/csmr-wcre

- ICSME - International Conference on Software Maintenance and Evolution[7]

3.4.2 Ferramentas da indústria

A seleção de ferramentas da indústria será feita de forma não estruturada a partir de uma busca livre e manual no site do projeto SAMATE[8] - *Software Assurance Metrics and Tool Evaluation* disponível em NIST (2016). As ferramentas com código-fonte disponível, escritas nas linguagens de programaçção suportadas pelo Analizo serão selecionadas.

3.5 ANÁLISE DE DADOS

Após download do código-fonte de cada ferramenta em sua versão mais recente, foi utilizada a ferramenta Analizo para a coleta das métricas. Com todas as métricas perfeitamente calculadas utilizamos a linguagem R para manipulação dos dados, tabelas e plotagem de gráficos.

Foram calculados os percentis de cada métrica para cada ferramenta analisada, cada percentil é a centésima parte dos dados ordenados de forma crescente, os percentis calculados foram: 1, 5, 10, 25, 50, 75, 90, 95 e 99. Dentre estes iremos discutir os resultados em função dos percentis 75, 90 e 95, correspondendo a valores muito frequentes, frequentes e pouco frequentes, respectivamente.

3.5.1 Distribuição dos valores das métricas

(pendente) “como será calculado a distribuição dos valores”.

3.5.2 Cálculo de distância e modelo de aproximação

(pendente) “como será feito este cálculo”.

3.5.3 Caracterização das ferramentas

(pendente) “como as ferramentas serão caracterizadas”.

[7] http://www.icsme.org
[8] http://samate.nist.gov

CAPÍTULO 4

RESULTADOS PRELIMINARES

4.1 ANÁLISE EXPLORATÓRIA DA REVISÃO ESTRUTURADA

A Tabela 4.1 apresenta um resumo do número de artigos em cada edição do SCAM e quantos artigos trazem publicação de ferramenta de análise estática com código fonte disponível.

Tabela 4.1 Total de artigos analisados por edições do SCAM

Edição	Total de artigos	Script filter	Artigos com ferramenta
SCAM 2001	23	6	-
SCAM 2002	18	6	-
SCAM 2003	21	7	-
SCAM 2004	17	3	-
SCAM 2005	19	7	-
SCAM 2006	22	9	2
SCAM 2007	23	7	1
SCAM 2008	29	14	-
SCAM 2009	20	10	-
SCAM 2010	21	15	1
SCAM 2011	21	9	1
SCAM 2012	22	12	4
SCAM 2013	24	12	-
SCAM 2014	35	16	1
SCAM 2015	30	18	0
Total	315	133	10

4.2 FERRAMENTAS COM CÓDIGO-FONTE DISPONÍVEL

A Tabela 4.2 apresenta as ferramenta da academia e da indústria após avaliação final sobre disponibilidade do código-fonte e possibilidade de ser analizada. Das 54 ferramentas do NIST apenas 19 tinham código fonte disponível, destas apenas 15 eram suportadas pelo Analizo. Dos 315 artigos avaliados na revisão estruturada apenas 10 tinham ferramentas disponíveis capaz de serem analizadas pelo Analizo.

Assim, temos um total de 25 ferramentas, 15 da indústria e 10 da academia.

Tabela 4.2 Lista com total de ferramentas a serem analisadas

n	Ferramentas da indústria	Linguagem	n	Ferramentas da academia	Linguagem
1	BOON	ansic	16	Indus	java
2	CQual	ansic	17	TACLE	java
3	RATS	ansic	18	JastAdd	java
4	Smatch	ansic	19	WALA	java
5	Splint	ansic	20	error-prone	java
6	UNO	ansic	21	AccessAnalysis	java
7	Clang Static Analyzer	cpp	22	Bakar Alir	java
8	Cppcheck	cpp	23	InputTracer	ansic
9	Jlint	cpp	24	srcML	cpp
10	WAP	java	25	Source Meter	java
11	Closure Compiler	java			
12	FindBugs	java			
13	FindSecurityBugs	java			
14	Pixy	java			
15	PMD	java			

4.3 ANÁLISE EXPLORATÓRIA DOS VALORES DAS MÉTRICAS

(pendente) “métricas das ferramentas, cálculo dos percentis, score de aproximação, valores de referências, discussão sobre cada métrica, discussão sobre as ferramentas com melhor qualidade, discutir detalhes da arquitetura das melhores ferramentas”.

4.3.1 Métricas para as ferramentas da academia

4.3.1.1 Conexões aferentes de uma classe (ACC) A Tabela 4.3 apresenta a métrica ACC para cade ferramenta acadêmica.

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	0.0	0.0	4.0	5.5	14.7
bakarali	611	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.9
errorprone	1703	0.0	0.0	0.0	0.0	0.0	0.0	2.0	3.0	20.0
indus	214	0.0	0.0	0.0	0.0	0.0	2.0	5.0	9.0	14.6
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	2.0	13.0	29.0	215.9
jastadd	59	0.0	0.0	0.0	0.0	0.0	4.0	21.8	32.1	36.5
sourcemeter	143	0.0	0.0	0.0	0.0	0.0	1.5	7.6	15.9	37.9
srcml	871	0.0	0.0	0.0	0.0	0.0	1.0	6.0	16.5	92.9
tacle	34	0.0	0.0	0.0	1.0	1.5	6.5	9.4	11.7	35.1
wala	2626	0.0	0.0	0.0	0.0	0.0	3.0	14.0	30.0	113.0

Tabela 4.3 percentis da métrica acc

4.3.1.2 Média de complexidade ciclomática por método (ACCM) A Tabela 4.4 ...

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	1.0	1.0	1.7	2.7	3.2	5.0
bakarali	611	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	2.0
errorprone	1703	0.0	0.0	0.0	0.0	1.0	1.0	2.2	3.5	6.5
indus	214	0.0	1.0	1.0	1.0	1.0	2.3	3.6	4.2	8.6
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	1.1	3.0	5.0	12.1
jastadd	59	0.0	1.0	1.0	1.0	1.1	1.7	2.5	3.1	4.8
sourcemeter	143	0.4	1.0	1.0	1.0	1.0	2.0	3.4	4.0	4.6
srcml	871	0.0	0.0	0.0	0.0	1.0	1.0	2.0	2.9	9.9
tacle	34	0.3	1.0	1.0	1.0	1.8	2.1	3.1	4.1	5.2
wala	2626	0.0	0.0	1.0	1.0	1.0	2.0	3.2	4.5	7.4

Tabela 4.4 percentis da métrica accm

4.3.1.3 Média do número de linhas de código por método (AMLOC) A Tabela 4.5...

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	4.0	7.0	10.7	16.0	17.8	22.9
bakarali	611	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.1	10.0
errorprone	1703	0.0	0.0	0.0	0.0	3.0	7.1	14.3	19.9	35.0
indus	214	0.0	1.0	1.0	3.0	7.5	14.6	23.7	27.8	36.9
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	14.5	31.5	49.0	156.4
jastadd	59	0.0	1.0	2.2	3.0	4.6	7.7	10.7	15.4	22.4
sourcemeter	143	0.8	3.0	3.0	3.0	5.7	12.4	23.3	29.8	1105.1
srcml	871	0.0	0.0	0.0	0.0	1.0	5.4	20.5	51.4	544.6
tacle	34	0.3	1.0	1.0	3.5	7.2	11.3	26.0	27.9	235.9
wala	2626	0.0	0.0	1.0	3.0	4.6	9.0	16.4	24.0	43.8

Tabela 4.5 percentis da métrica amloc

4.3.1.4 Média do Número de Parâmetros por Método (ANPM) A Tabela 4.6...

4.3.1.5 Acoplamento entre objetos (CBO) A Tabela 4.7...

4.3.1.6 Profundidade da árvore de herança (DIT) A Tabela 4.8...

4.3.1.7 Ausência de coesão em métodos (LCOM4) A Tabela 4.9...

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	0.8	1.0	1.2	1.5	1.8
bakarali	611	0.0	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.7
errorprone	1703	0.0	0.0	0.0	0.0	0.0	1.3	2.0	2.0	3.0
indus	214	0.0	0.0	0.1	0.5	1.0	1.4	2.0	2.5	3.0
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.4	4.0
jastadd	59	0.0	0.0	0.0	0.3	1.0	1.4	1.9	2.0	2.0
sourcemeter	143	0.0	0.0	0.0	0.3	1.0	2.0	2.7	3.0	3.5
srcml	871	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0	4.0
tacle	34	0.0	0.0	0.0	0.5	0.9	1.2	1.7	1.9	2.5
wala	2626	0.0	0.0	0.0	0.4	0.9	1.3	2.0	2.7	4.2

Tabela 4.6 percentis da métrica anpm

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	1.0	1.0	3.5	6.0	6.0	7.0	9.0	10.1
bakarali	611	4.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0	8.0
errorprone	1703	5.0	78.0	86.0	130.0	162.0	168.0	196.0	197.0	198.0
indus	214	3.1	4.0	10.0	20.2	22.0	58.5	61.7	62.0	63.0
inputtracer	2045	54.0	123.0	203.2	376.0	503.0	584.0	643.0	660.0	672.0
jastadd	59	3.5	7.0	7.0	8.0	13.0	18.5	24.0	24.0	24.0
sourcemeter	143	1.3	5.2	8.0	15.5	32.0	38.0	39.0	39.0	39.0
srcml	871	0.0	5.0	5.0	9.0	94.0	105.0	148.0	166.0	176.0
tacle	34	0.0	0.0	0.3	5.8	14.5	17.0	23.0	23.0	23.0
wala	2626	40.0	52.0	128.0	369.2	615.0	873.0	986.0	998.0	1011.0

Tabela 4.7 percentis da métrica cbo

4.3.1.8 Número de linhas de código (LOC)

A Tabela 4.10...

4.3.1.9 Número de atributos (NOA)

A Tabela 4.11...

4.3.1.10 Número de filhos (NOC)

A Tabela 4.12...

4.3.1.11 Número de métodos (NOM)

A Tabela 4.13...

4.3.1.12 Número de atributos públicos (NPA)

A Tabela 4.14...

4.3.1.13 Número de métodos públicos (NPM)

A Tabela 4.15...

4.3.1.14 Resposta para uma classe (RFC)

A Tabela 4.16...

4.3.1.15 Complexidade estrutural (SC)

A Tabela 4.17...

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	1.0	1.0	1.0	2.0	2.0
bakarali	611	0.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
errorprone	1703	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0	3.0
indus	214	0.0	0.0	0.0	1.0	1.0	2.0	3.0	3.0	4.0
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jastadd	59	0.0	0.0	0.0	0.0	1.0	1.0	2.0	3.0	3.0
sourcemeter	143	0.0	0.0	1.0	1.0	2.0	2.0	3.0	3.0	3.0
srcml	871	0.0	0.0	0.0	0.0	0.0	0.0	2.0	4.0	6.3
tacle	34	0.0	0.0	0.0	0.0	1.0	1.0	2.0	2.0	2.0
wala	2626	0.0	0.0	0.0	0.0	1.0	2.0	3.0	4.0	5.0

Tabela 4.8 percentis da métrica dit

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	1.0	2.0	4.0	9.0	19.0	27.3
bakarali	611	1.0	2.0	2.0	3.0	8.0	10.0	12.0	14.0	19.8
errorprone	1703	0.0	0.0	0.0	0.0	1.0	3.0	6.0	9.0	24.0
indus	214	0.0	1.0	1.0	1.0	2.0	4.0	7.0	13.0	25.6
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	1.0	3.0	8.0	36.6
jastadd	59	0.0	1.0	1.0	1.0	2.0	6.0	20.2	25.2	96.9
sourcemeter	143	0.4	1.0	1.0	1.0	2.0	3.0	4.0	5.0	7.6
srcml	871	0.0	0.0	0.0	0.0	1.0	3.0	8.0	13.0	30.3
tacle	34	0.3	1.0	1.0	1.0	2.0	3.0	4.7	7.0	10.4
wala	2626	0.0	0.0	1.0	1.0	2.0	4.0	8.0	12.8	27.0

Tabela 4.9 percentis da métrica lcom4

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	6.5	24.0	72.5	135.0	160.0	365.9
bakarali	611	6.0	6.0	6.0	12.0	24.0	30.0	36.0	42.0	60.0
errorprone	1703	0.0	0.0	0.0	0.0	6.0	20.0	64.0	119.0	333.8
indus	214	0.0	1.0	2.0	6.0	34.0	107.8	199.6	280.3	584.6
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	37.0	241.8	823.4	3712.7
jastadd	59	0.0	1.9	4.6	11.0	27.0	80.0	296.0	482.2	746.6
sourcemeter	143	1.3	3.0	3.0	9.0	17.0	49.0	138.6	232.1	1105.1
srcml	871	0.0	0.0	0.0	0.0	2.0	26.0	107.0	245.5	908.6
tacle	34	0.7	2.0	3.0	13.8	78.5	139.8	263.7	683.4	966.5
wala	2626	0.0	0.0	1.0	6.0	20.5	55.0	133.0	232.0	586.5

Tabela 4.10 percentis da métrica loc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	1.0	3.0	7.0	9.5	13.0
bakarali	611	0.0	1.0	1.0	2.0	4.0	5.0	6.0	7.0	10.0
errorprone	1703	0.0	0.0	0.0	0.0	0.0	1.0	2.0	4.0	9.0
indus	214	0.0	0.0	0.0	0.0	1.0	4.0	7.0	11.3	17.7
inputtracer	2045	0.0	0.0	0.0	0.0	2.0	4.0	8.0	13.8	50.0
jastadd	59	0.0	0.0	0.0	0.0	1.0	3.0	5.0	6.2	10.5
sourcemeter	143	0.0	0.0	0.0	0.0	1.0	4.0	9.0	26.6	131.7
srcml	871	0.0	0.0	0.0	0.0	0.0	1.0	3.0	6.5	22.2
tacle	34	0.0	0.0	0.0	0.0	2.0	7.5	10.0	10.3	15.0
wala	2626	0.0	0.0	0.0	0.0	1.0	2.0	5.0	7.0	23.2

Tabela 4.11 percentis da métrica noa

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.5	5.0
bakarali	611	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
errorprone	1703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	7.0
indus	214	0.0	0.0	0.0	0.0	0.0	1.0	1.0	2.0	3.0
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
jastadd	59	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.1	4.8
sourcemeter	143	0.0	0.0	0.0	0.0	0.0	0.0	4.0	4.0	4.6
srcml	871	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	4.0
tacle	34	0.0	0.0	0.0	0.0	0.0	0.8	1.7	2.0	3.3
wala	2626	0.0	0.0	0.0	0.0	0.0	0.0	2.0	3.0	9.8

Tabela 4.12 percentis da métrica noc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	1.5	3.0	7.0	17.0	21.5	27.3
bakarali	611	1.0	2.0	2.0	3.5	8.0	10.0	12.0	14.0	20.0
errorprone	1703	0.0	0.0	0.0	0.0	1.0	3.0	7.0	11.0	29.0
indus	214	0.0	1.0	1.0	2.0	4.0	8.8	13.7	18.0	31.2
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	2.0	11.0	27.8	135.6
jastadd	59	0.0	1.0	1.0	2.0	6.0	16.0	32.6	83.6	157.1
sourcemeter	143	0.4	1.0	1.0	2.0	3.0	5.0	6.0	8.9	13.6
srcml	871	0.0	0.0	0.0	0.0	1.0	3.0	10.0	19.0	42.2
tacle	34	0.3	1.0	1.3	4.0	8.0	15.2	25.4	28.4	37.0
wala	2626	0.0	0.0	1.0	2.0	4.0	8.0	15.0	23.0	45.5

Tabela 4.13 percentis da métrica nom

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.5	5.1
bakarali	611	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
errorprone	1703	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0
indus	214	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	6.7
inputtracer	2045	0.0	0.0	0.0	0.0	2.0	4.0	8.0	13.8	50.0
jastadd	59	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.2	6.3
sourcemeter	143	0.0	0.0	0.0	0.0	0.0	1.0	4.6	26.6	130.7
srcml	871	0.0	0.0	0.0	0.0	0.0	1.0	2.0	4.0	14.6
tacle	34	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.7
wala	2626	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.8	10.0

Tabela 4.14 percentis da métrica npa

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	1.0	1.0	4.0	10.0	17.5	24.9
bakarali	611	1.0	2.0	2.0	3.0	8.0	10.0	12.0	14.0	20.0
errorprone	1703	0.0	0.0	0.0	0.0	1.0	3.0	6.0	9.0	28.0
indus	214	0.0	0.0	1.0	1.0	3.0	6.0	11.0	13.3	23.7
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	2.0	11.0	27.8	135.6
jastadd	59	0.0	0.9	1.0	2.0	5.0	11.0	29.4	79.1	157.1
sourcemeter	143	0.0	1.0	1.0	2.0	2.0	3.0	5.0	6.9	11.7
srcml	871	0.0	0.0	0.0	0.0	1.0	3.0	9.0	13.0	34.3
tacle	34	0.3	1.0	1.0	3.0	7.0	12.8	20.7	23.3	30.7
wala	2626	0.0	0.0	1.0	1.0	3.0	6.0	12.0	18.0	36.0

Tabela 4.15 percentis da métrica npm

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	2.0	5.0	11.5	27.0	46.5	72.6
bakarali	611	1.0	2.0	2.0	4.0	8.0	10.0	12.0	14.0	20.0
errorprone	1703	0.0	0.0	0.0	0.0	2.0	6.0	13.0	25.0	61.0
indus	214	0.0	1.0	1.0	2.0	7.0	20.0	38.7	48.0	106.0
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	8.0	47.6	145.0	647.6
jastadd	59	0.0	1.0	1.0	3.0	12.0	34.0	73.4	206.1	344.6
sourcemeter	143	0.4	1.0	2.0	3.0	7.0	17.5	35.6	51.3	200.3
srcml	871	0.0	0.0	0.0	0.0	1.0	9.0	27.0	41.5	177.4
tacle	34	0.3	1.7	2.0	5.5	21.5	56.8	82.1	112.4	169.9
wala	2626	0.0	0.0	1.0	2.0	8.0	20.0	46.0	79.8	198.8

Tabela 4.16 percentis da métrica rfc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
accessanalysis	91	0.0	0.0	0.0	4.0	8.0	20.0	54.0	114.0	163.8
bakarali	611	8.0	16.0	16.0	17.0	64.0	80.0	96.0	112.0	144.0
errorprone	1703	0.0	0.0	0.0	0.0	165.0	417.5	844.8	1348.2	4113.8
indus	214	0.0	4.7	20.0	36.0	62.0	120.0	298.2	370.2	668.7
inputtracer	2045	0.0	0.0	0.0	0.0	0.0	539.0	1348.0	3413.6	18666.0
jastadd	59	0.0	7.0	7.0	16.0	34.0	102.0	211.2	492.0	1014.2
sourcemeter	143	0.0	7.2	16.4	34.5	56.0	96.0	117.0	147.2	195.0
srcml	871	0.0	0.0	0.0	0.0	17.0	297.5	1008.0	1800.0	4654.1
tacle	34	0.0	0.0	0.0	5.2	18.0	45.8	68.1	87.4	100.7
wala	2626	0.0	0.0	53.0	391.0	998.0	2477.5	5059.0	8087.5	18411.5

Tabela 4.17 percentis da métrica sc

4.3.2 Métricas para as ferramentas da indústria

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.0	0.5	3.2	4.9	5.9	6.8
clang	579	0.0	0.0	0.0	0.0	0.0	1.0	3.0	9.0	44.2
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.9	4.0
cppcheck	338	0.0	0.0	0.0	0.0	0.0	2.0	18.0	51.7	264.5
cqual	78	0.0	0.0	0.0	0.0	3.0	24.0	85.7	110.0	187.6
findbugs	1486	0.0	0.0	0.0	0.0	0.0	3.0	11.0	21.8	129.5
findsecuritybugs	91	0.0	0.0	0.0	0.0	0.0	0.0	5.0	15.5	27.2
jlint	3	0.0	0.2	0.4	1.0	2.0	4.0	5.2	5.6	5.9
pixy	229	0.0	0.0	0.0	0.0	1.0	6.0	18.0	38.8	80.9
pmd	1340	0.0	0.0	0.0	0.0	0.0	2.0	9.0	19.0	82.6
rats	19	0.0	0.0	0.0	0.0	0.0	8.0	13.4	15.1	15.8
smatch	483	0.0	0.0	0.0	0.0	0.0	1.0	19.0	93.9	326.7
splint	681	0.0	0.0	0.0	0.0	0.0	7.0	34.0	96.0	544.6
uno	19	0.0	0.0	0.0	1.0	4.0	34.0	95.4	129.4	132.3
wap	338	0.0	0.0	0.0	0.0	0.0	0.0	4.0	17.1	45.8

Tabela 4.18 percentis da métrica acc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.6	1.0	1.4	4.1	5.4	7.3	8.0	8.5
clang	579	0.0	0.0	0.0	1.0	1.0	1.4	3.0	4.1	7.7
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	1.6	2.3	5.0
cppcheck	338	0.0	0.0	0.0	1.0	1.2	3.0	6.8	9.4	17.3
cqual	78	0.0	0.0	1.0	1.0	1.2	2.6	6.3	8.7	38.4
findbugs	1486	0.0	0.0	1.0	1.0	1.3	2.5	4.8	7.0	12.4
findsecuritybugs	91	1.0	1.0	1.0	1.3	2.0	3.0	4.2	5.8	7.9
jlint	3	15.2	15.2	15.3	15.5	15.9	160.5	247.2	276.1	299.2
pixy	229	0.0	0.0	1.0	1.0	1.5	2.0	4.1	5.5	9.1
pmd	1340	0.0	1.0	1.0	1.0	1.0	1.9	3.0	4.0	7.0
rats	19	0.0	0.0	0.0	0.0	4.9	5.5	8.8	9.4	10.2
smatch	483	0.0	0.0	0.2	1.0	1.8	3.2	4.5	6.1	10.7
splint	681	0.0	0.0	1.0	1.0	1.0	2.0	4.0	6.0	15.0
uno	19	0.0	0.0	0.8	1.0	4.7	6.9	8.2	11.5	18.1
wap	338	1.0	1.0	1.0	1.0	1.0	1.0	2.0	5.6	23.9

Tabela 4.19 percentis da métrica accm

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.4	2.2	4.0	5.6	18.3	31.1	38.8	43.2	46.9
clang	579	0.0	0.0	0.0	1.0	1.7	9.3	21.4	29.1	56.0
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	4.0	7.8	17.3
cppcheck	338	0.0	0.0	0.0	4.0	9.6	20.9	30.9	45.5	83.0
cqual	78	0.0	0.0	1.0	4.2	9.2	18.6	48.6	86.2	234.9
findbugs	1486	0.0	0.0	1.0	3.0	6.0	12.3	21.7	30.0	59.9
findsecuritybugs	91	2.8	3.0	3.2	5.3	7.5	12.8	21.4	26.8	53.0
jlint	3	11.8	15.2	19.4	31.9	52.8	69.5	79.5	82.8	85.5
pixy	229	0.0	0.0	1.3	3.7	7.0	15.0	28.7	39.5	64.7
pmd	1340	0.0	1.0	3.0	3.0	4.6	9.6	16.0	22.0	37.9
rats	19	0.0	0.0	0.0	0.0	23.2	33.5	50.2	57.8	60.1
smatch	483	0.0	0.0	0.2	4.6	9.0	14.9	22.4	26.0	51.2
splint	681	0.0	0.0	1.0	4.5	9.0	15.0	28.0	42.7	129.6
uno	19	0.0	0.0	0.8	12.0	27.5	36.9	62.1	87.4	214.3
wap	338	1.0	1.0	1.0	1.0	1.0	7.0	27.9	119.7	642.3

Tabela 4.20 percentis da métrica amloc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.4	0.8	1.0	1.4	2.0	2.0	2.1	2.3
clang	579	0.0	0.0	0.0	0.0	0.6	1.3	2.0	2.4	3.1
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	2.0	2.0	2.0
cppcheck	338	0.0	0.0	0.0	0.0	0.6	1.3	2.0	2.4	3.0
cqual	78	0.0	0.0	0.9	1.0	1.6	2.0	2.4	2.8	4.8
findbugs	1486	0.0	0.0	0.0	0.5	1.0	1.3	2.0	2.3	4.0
findsecuritybugs	91	0.4	0.5	0.6	0.7	1.0	1.0	1.8	2.1	3.0
jlint	3	1.2	1.2	1.3	1.4	1.7	2.1	2.3	2.4	2.5
pixy	229	0.0	0.0	0.1	0.5	1.0	1.5	2.0	3.0	3.6
pmd	1340	0.0	0.0	0.0	0.2	1.0	1.5	2.0	2.0	3.1
rats	19	0.0	0.0	0.0	0.0	1.2	1.5	1.7	1.8	1.8
smatch	483	0.0	0.0	0.1	1.0	1.0	1.7	2.0	2.2	3.0
splint	681	0.0	0.0	0.5	1.0	1.0	1.4	2.0	2.2	3.9
uno	19	0.0	0.0	0.8	1.2	1.5	1.7	2.0	2.0	2.1
wap	338	0.0	0.0	0.0	0.0	0.0	0.7	1.0	2.8	7.6

Tabela 4.21 percentis da métrica anpm

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.0	2.5	5.2	6.0	6.0	6.0
clang	579	10.0	21.8	35.8	60.0	94.0	125.0	139.0	140.0	145.0
closure	35	0.0	0.0	0.0	1.0	2.0	2.0	2.0	2.0	2.0
cppcheck	338	17.4	25.0	31.0	62.8	93.0	114.0	140.0	145.0	150.0
cqual	78	0.0	0.8	1.0	8.0	40.0	46.0	51.0	51.3	53.0
findbugs	1486	8.8	66.0	114.0	253.0	393.0	506.0	554.0	562.8	570.0
findsecuritybugs	91	0.0	0.0	0.0	3.0	5.0	10.0	15.0	15.0	16.0
jlint	3	0.0	0.1	0.2	0.5	1.0	1.5	1.8	1.9	2.0
pixy	229	17.0	36.4	42.8	68.0	81.0	89.0	108.0	115.0	117.0
pmd	1340	24.8	61.0	95.0	162.0	266.0	338.0	378.0	389.0	400.0
rats	19	0.7	3.6	4.0	5.5	6.0	8.0	8.0	8.0	8.0
smatch	483	16.0	44.1	49.0	68.0	109.0	116.0	122.0	123.0	123.0
splint	681	0.0	192.0	220.0	277.0	300.0	317.0	320.0	321.0	322.0
uno	19	3.0	3.0	3.0	3.5	7.0	12.5	14.0	14.0	14.0
wap	338	29.5	35.0	35.0	36.0	36.0	37.0	43.0	44.1	50.6

Tabela 4.22 percentis da métrica cbo

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
clang	579	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	0.6	1.0	1.0
cppcheck	338	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0	2.0
cqual	78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
findbugs	1486	0.0	0.0	0.0	0.0	1.0	2.0	4.0	6.0	7.0
findsecuritybugs	91	0.0	0.0	0.0	1.0	1.0	1.0	5.0	5.0	5.0
jlint	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pixy	229	0.0	0.0	0.0	0.0	1.0	1.0	1.0	2.0	2.0
pmd	1340	0.0	0.0	0.0	0.0	1.0	4.0	4.0	5.0	6.0
rats	19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
smatch	483	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
splint	681	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
uno	19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wap	338	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	3.0

Tabela 4.23 percentis da métrica dit

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.6	1.0	1.0	1.0	1.0	2.8	3.4	3.9
clang	579	0.0	0.0	0.0	1.0	1.0	3.0	5.0	8.0	16.4
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.3	2.0
cppcheck	338	0.0	0.0	0.0	1.0	2.0	5.0	9.0	14.3	50.4
cqual	78	0.0	0.0	1.0	1.0	1.0	3.8	12.3	17.1	208.3
findbugs	1486	0.0	0.0	1.0	1.0	2.0	4.0	7.0	11.0	27.5
findsecuritybugs	91	1.0	1.0	1.0	1.0	2.0	2.0	2.0	2.5	6.5
jlint	3	1.0	1.0	1.0	1.0	1.0	2.0	2.6	2.8	3.0
pixy	229	0.0	0.0	1.0	1.0	3.0	4.0	7.2	11.0	21.2
pmd	1340	0.0	1.0	1.0	1.0	2.0	4.0	6.0	8.0	22.6
rats	19	0.0	0.0	0.0	0.0	1.0	26.0	49.4	51.1	51.8
smatch	483	0.0	0.0	0.2	1.0	1.0	2.0	4.0	7.0	31.2
splint	681	0.0	0.0	1.0	1.0	2.0	4.0	8.0	13.0	65.2
uno	19	0.0	0.0	0.8	1.0	2.0	3.0	6.0	6.0	6.0
wap	338	1.0	1.0	1.0	1.0	1.0	3.0	5.0	11.6	86.0

Tabela 4.24 percentis da métrica lcom4

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.4	2.2	4.2	8.2	59.0	139.0	403.0	1008.8	1573.8
clang	579	0.0	0.0	0.0	1.0	4.0	27.0	84.6	144.3	289.7
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	4.4	7.8	17.3
cppcheck	338	0.0	0.0	0.0	8.0	54.5	302.8	1102.6	2527.9	5240.0
cqual	78	0.0	0.0	4.5	9.0	49.0	327.5	1469.2	2425.2	3372.7
findbugs	1486	0.0	0.0	2.0	8.0	28.0	77.0	170.5	307.5	705.3
findsecuritybugs	91	2.8	3.5	6.0	12.5	20.0	52.5	90.0	139.0	294.7
jlint	3	40.5	114.6	207.2	485.0	948.0	1028.0	1076.0	1092.0	1104.8
pixy	229	0.0	0.0	4.0	14.0	32.0	86.0	275.2	551.4	1443.0
pmd	1340	0.0	1.0	3.0	9.0	17.0	50.0	108.0	170.2	383.3
rats	19	0.0	0.0	0.0	0.0	470.0	1437.0	1578.6	1597.8	1603.6
smatch	483	0.0	0.0	0.2	8.0	22.0	90.0	326.8	798.9	2029.1
splint	681	0.0	0.0	2.0	9.0	22.0	85.0	343.0	1047.0	4479.2
uno	19	0.0	0.0	3.2	173.5	639.0	1196.5	1703.0	1960.0	3752.8
wap	338	1.0	1.0	1.0	1.0	1.0	32.5	180.5	641.7	9333.0

Tabela 4.25 percentis da métrica loc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.8	2.5	5.8	8.9	22.9	36.6
clang	579	0.0	0.0	0.0	0.0	0.0	2.0	4.0	5.0	12.2
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
cppcheck	338	0.0	0.0	0.0	0.0	1.0	3.0	6.3	9.1	20.5
cqual	78	0.0	0.0	0.0	0.0	2.5	15.8	70.3	90.1	268.4
findbugs	1486	0.0	0.0	0.0	0.0	1.0	3.0	7.0	11.0	25.2
findsecuritybugs	91	0.0	0.0	0.0	1.0	2.0	3.0	6.0	8.0	10.2
jlint	3	0.2	1.2	2.4	6.0	12.0	13.5	14.4	14.7	14.9
pixy	229	0.0	0.0	0.0	1.0	2.0	3.0	6.0	8.0	26.2
pmd	1340	0.0	0.0	0.0	0.0	1.0	2.0	4.1	7.0	17.0
rats	19	0.0	0.0	0.0	0.0	24.0	41.5	43.6	46.4	49.3
smatch	483	0.0	0.0	0.0	0.0	1.0	4.0	9.0	13.9	92.0
splint	681	0.0	0.0	0.0	0.0	1.0	4.0	9.0	20.0	98.6
uno	19	0.0	0.0	0.8	4.5	16.0	45.0	64.0	77.6	117.9
wap	338	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	331.7

Tabela 4.26 percentis da métrica noa

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
clang	579	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.4
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.6
cppcheck	338	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	5.3
cqual	78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
findbugs	1486	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	7.0
findsecuritybugs	91	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0
jlint	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
pixy	229	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	5.7
pmd	1340	0.0	0.0	0.0	0.0	0.0	0.0	1.0	3.0	11.0
rats	19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
smatch	483	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
splint	681	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
uno	19	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
wap	338	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.0

Tabela 4.27 percentis da métrica noc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.6	1.0	1.8	2.0	5.8	8.9	44.1	78.4
clang	579	0.0	0.0	0.0	1.0	2.0	3.0	6.0	10.0	20.4
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.3	2.0
cppcheck	338	0.0	0.0	0.0	1.0	6.0	17.0	36.3	92.9	177.5
cqual	78	0.0	0.0	1.0	1.0	4.5	20.8	77.3	105.8	256.0
findbugs	1486	0.0	0.0	1.0	2.0	4.0	7.0	15.0	23.0	59.5
findsecuritybugs	91	1.0	1.0	1.0	2.0	2.0	4.0	7.0	13.0	29.8
jlint	3	2.2	2.9	3.8	6.5	11.0	16.0	19.0	20.0	20.8
pixy	229	0.0	0.0	1.0	2.0	4.0	8.0	18.2	29.0	59.2
pmd	1340	0.0	1.0	1.0	2.0	3.0	6.0	12.0	20.0	46.6
rats	19	0.0	0.0	0.0	0.0	11.0	40.5	49.4	51.1	51.8
smatch	483	0.0	0.0	0.2	1.0	2.0	7.0	21.8	43.0	99.5
splint	681	0.0	0.0	1.0	1.0	3.0	8.0	20.0	43.0	178.4
uno	19	0.0	0.0	0.8	6.5	17.0	40.5	53.8	70.5	138.9
wap	338	1.0	1.0	1.0	1.0	1.0	3.0	5.0	17.1	86.0

Tabela 4.28 percentis da métrica nom

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.8	2.5	5.8	8.9	22.9	36.6
clang	579	0.0	0.0	0.0	0.0	0.0	1.0	2.0	4.0	7.4
closure	35	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0
cppcheck	338	0.0	0.0	0.0	0.0	0.0	1.0	4.0	7.0	15.3
cqual	78	0.0	0.0	0.0	0.0	2.5	15.8	70.3	90.1	268.4
findbugs	1486	0.0	0.0	0.0	0.0	0.0	0.0	2.0	4.0	12.0
findsecuritybugs	91	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	4.1
jlint	3	0.2	1.2	2.4	6.0	12.0	13.5	14.4	14.7	14.9
pixy	229	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.6	8.3
pmd	1340	0.0	0.0	0.0	0.0	0.0	0.0	1.0	2.0	7.0
rats	19	0.0	0.0	0.0	0.0	24.0	41.5	43.6	46.4	49.3
smatch	483	0.0	0.0	0.0	0.0	1.0	4.0	9.0	13.9	92.0
splint	681	0.0	0.0	0.0	0.0	1.0	4.0	9.0	20.0	98.6
uno	19	0.0	0.0	0.8	4.5	16.0	45.0	64.0	77.6	117.9
wap	338	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	323.7

Tabela 4.29 percentis da métrica npa

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.6	1.0	1.8	2.0	5.8	8.9	44.1	78.4
clang	579	0.0	0.0	0.0	0.5	2.0	3.0	6.0	9.0	19.2
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.3	2.0
cppcheck	338	0.0	0.0	0.0	1.0	2.0	8.0	15.0	22.1	50.6
cqual	78	0.0	0.0	1.0	1.0	4.5	20.8	77.3	105.8	256.0
findbugs	1486	0.0	0.0	1.0	2.0	3.0	6.0	13.0	18.0	54.0
findsecuritybugs	91	1.0	1.0	1.0	2.0	2.0	3.0	5.0	6.0	21.2
jlint	3	2.2	2.9	3.8	6.5	11.0	16.0	19.0	20.0	20.8
pixy	229	0.0	0.0	1.0	2.0	3.0	6.0	13.2	28.6	46.4
pmd	1340	0.0	1.0	1.0	1.0	3.0	5.0	10.0	16.0	45.6
rats	19	0.0	0.0	0.0	0.0	11.0	40.5	49.4	51.1	51.8
smatch	483	0.0	0.0	0.2	1.0	2.0	7.0	21.8	43.0	99.5
splint	681	0.0	0.0	1.0	1.0	3.0	8.0	20.0	43.0	178.4
uno	19	0.0	0.0	0.8	6.5	17.0	40.5	53.8	70.5	138.9
wap	338	1.0	1.0	1.0	1.0	1.0	3.0	5.0	17.0	84.0

Tabela 4.30 percentis da métrica npm

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.1	0.6	1.0	2.5	8.0	21.8	42.0	355.7	661.6
clang	579	0.0	0.0	0.0	1.0	3.0	9.0	16.0	26.1	66.7
closure	35	0.0	0.0	0.0	0.0	0.0	1.0	2.0	2.0	3.3
cppcheck	338	0.0	0.0	0.0	2.0	15.0	67.0	180.5	375.5	707.8
cqual	78	0.0	0.0	1.0	4.2	14.0	122.5	629.3	745.4	1043.4
findbugs	1486	0.0	0.0	1.0	3.0	8.0	24.0	52.0	88.8	202.8
findsecuritybugs	91	1.0	1.5	3.0	5.0	6.0	10.0	17.0	38.0	105.3
jlint	3	14.7	17.7	21.4	32.5	51.0	83.5	103.0	109.5	114.7
pixy	229	0.0	0.0	2.0	5.0	9.0	22.0	71.4	100.0	284.9
pmd	1340	0.0	1.0	2.0	3.0	7.0	18.0	39.0	68.0	139.0
rats	19	0.0	0.0	0.0	0.0	65.0	399.0	542.6	551.0	565.4
smatch	483	0.0	0.0	0.2	3.0	7.0	41.5	121.6	281.9	763.5
splint	681	0.0	0.0	1.0	3.0	7.0	28.0	104.0	267.0	1151.6
uno	19	0.0	0.0	0.8	27.5	93.0	304.0	522.2	742.5	1401.3
wap	338	1.0	1.0	1.0	1.0	1.0	3.0	16.3	100.0	185.1

Tabela 4.31 percentis da métrica rfc

	classes	1%	5%	10%	25%	50%	75%	90%	95%	99%
boon	12	0.0	0.0	0.0	0.0	1.5	5.2	14.1	19.0	23.0
clang	579	0.0	0.0	0.0	24.0	115.0	226.0	460.8	723.0	1678.3
closure	35	0.0	0.0	0.0	0.0	0.0	2.0	2.0	2.0	3.3
cppcheck	338	0.0	0.0	0.0	90.2	237.0	409.0	704.0	950.8	5293.0
cqual	78	0.0	0.0	0.0	1.0	46.0	140.8	558.9	816.0	7501.3
findbugs	1486	0.0	0.0	90.5	322.5	605.0	1243.5	2507.5	3923.5	10572.4
findsecuritybugs	91	0.0	0.0	0.0	3.0	9.0	15.0	22.0	30.0	70.5
jlint	3	0.0	0.2	0.4	1.0	2.0	2.5	2.8	2.9	3.0
pixy	229	0.0	0.0	40.0	82.0	185.0	351.0	533.6	788.4	1471.6
pmd	1340	0.0	51.0	124.0	266.0	411.0	836.0	1514.4	1976.2	4876.0
rats	19	0.0	0.0	0.0	0.0	8.0	155.0	296.4	306.6	310.9
smatch	483	0.0	0.0	2.6	75.5	113.0	123.0	320.4	541.6	2032.9
splint	681	0.0	0.0	0.0	311.0	322.0	939.0	1935.0	2900.0	8942.4
uno	19	0.0	0.0	2.4	5.5	13.0	23.0	32.4	46.2	76.4
wap	338	33.1	35.0	35.0	36.0	37.0	108.0	211.5	503.2	3064.2

Tabela 4.32 percentis da métrica sc

4.3.3 Gráficos das métricas para as ferramentas academicas

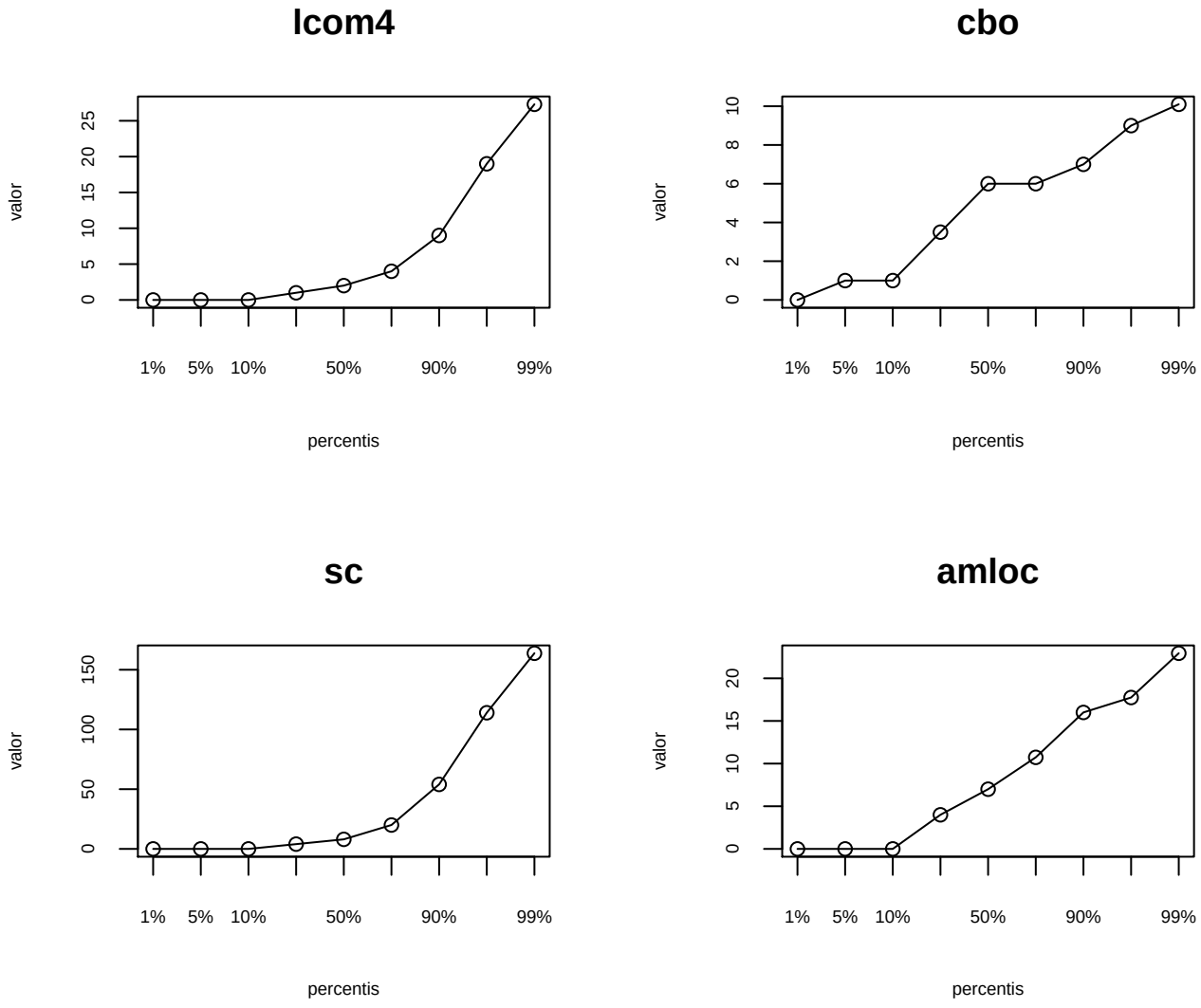

Figura 4.1 distribuição das métricas para a ferramenta accessanalysis

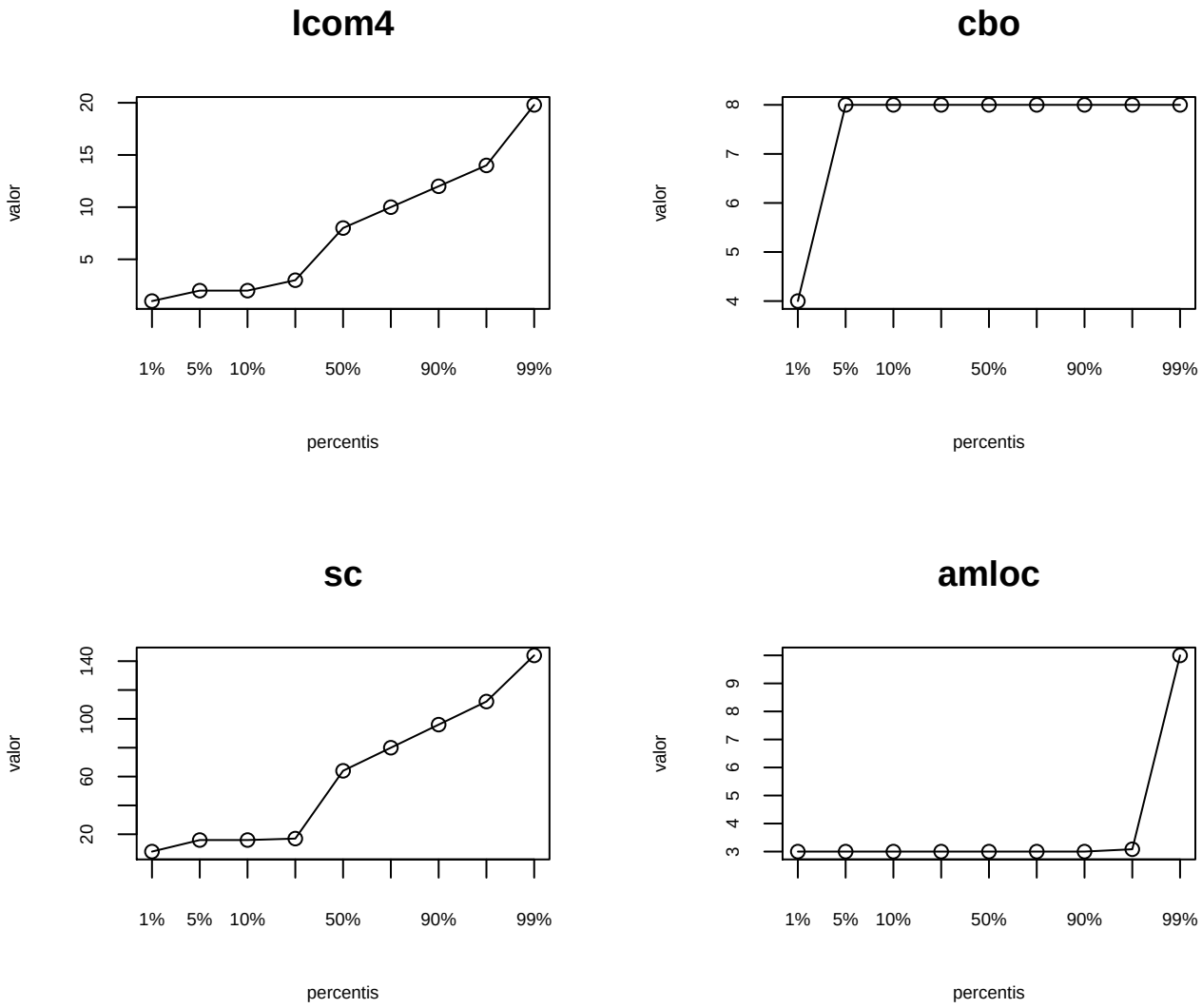

Figura 4.2 distribuição das métricas para a ferramenta bakar-ali

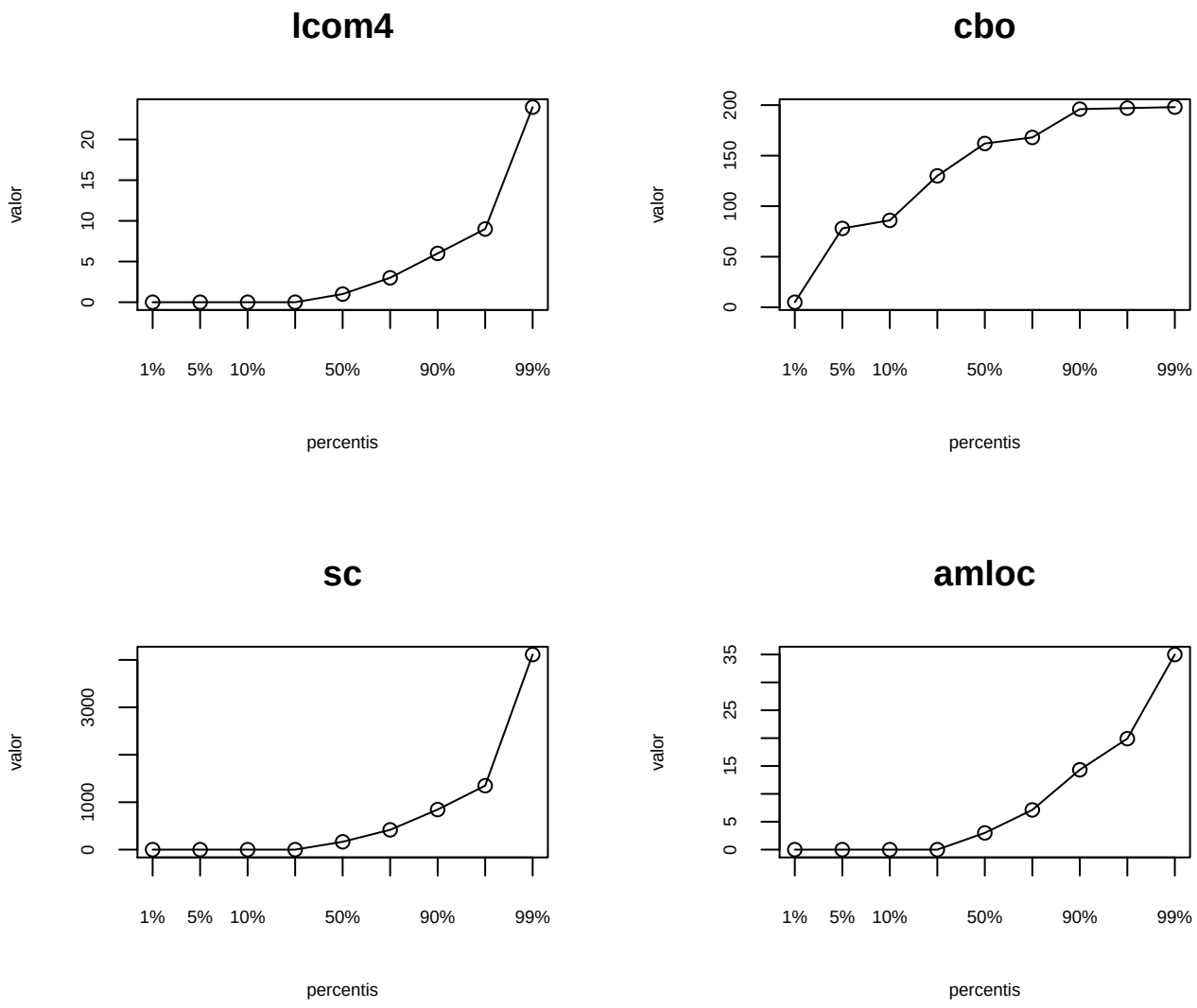

Figura 4.3 distribuição das métricas para a ferramenta error-prone

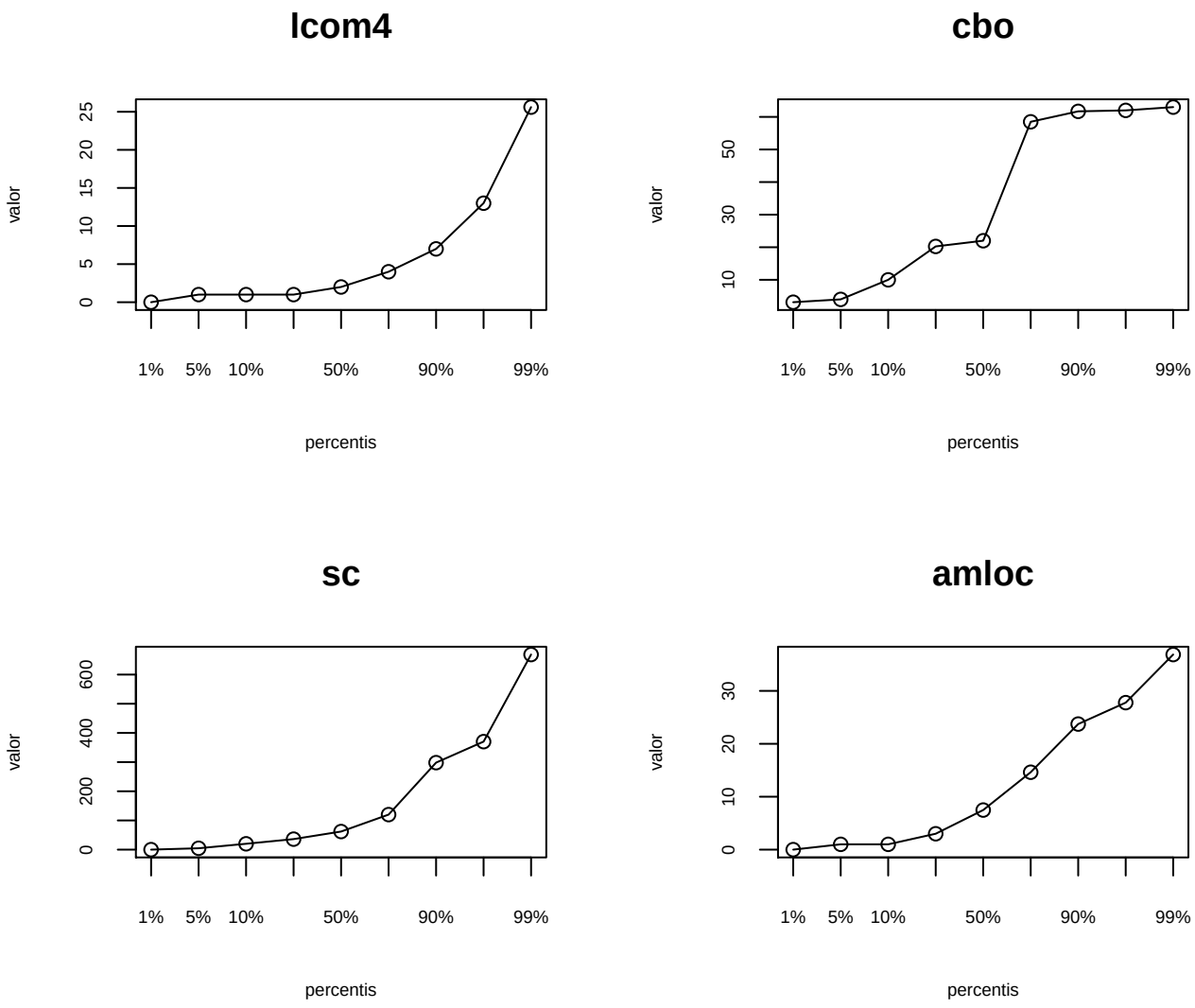

Figura 4.4 distribuição das métricas para a ferramenta indus

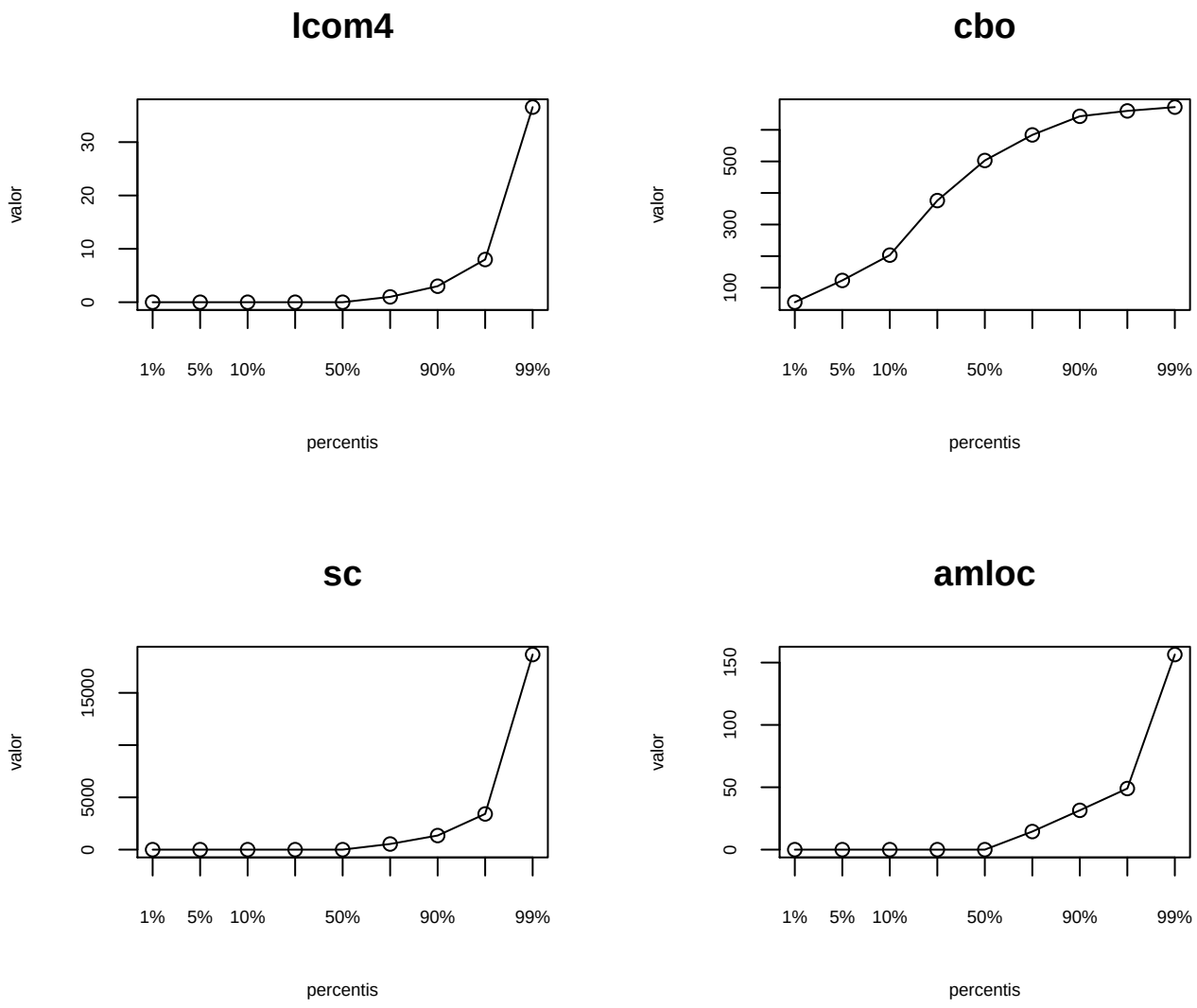

Figura 4.5 distribuição das métricas para a ferramenta inputtracer

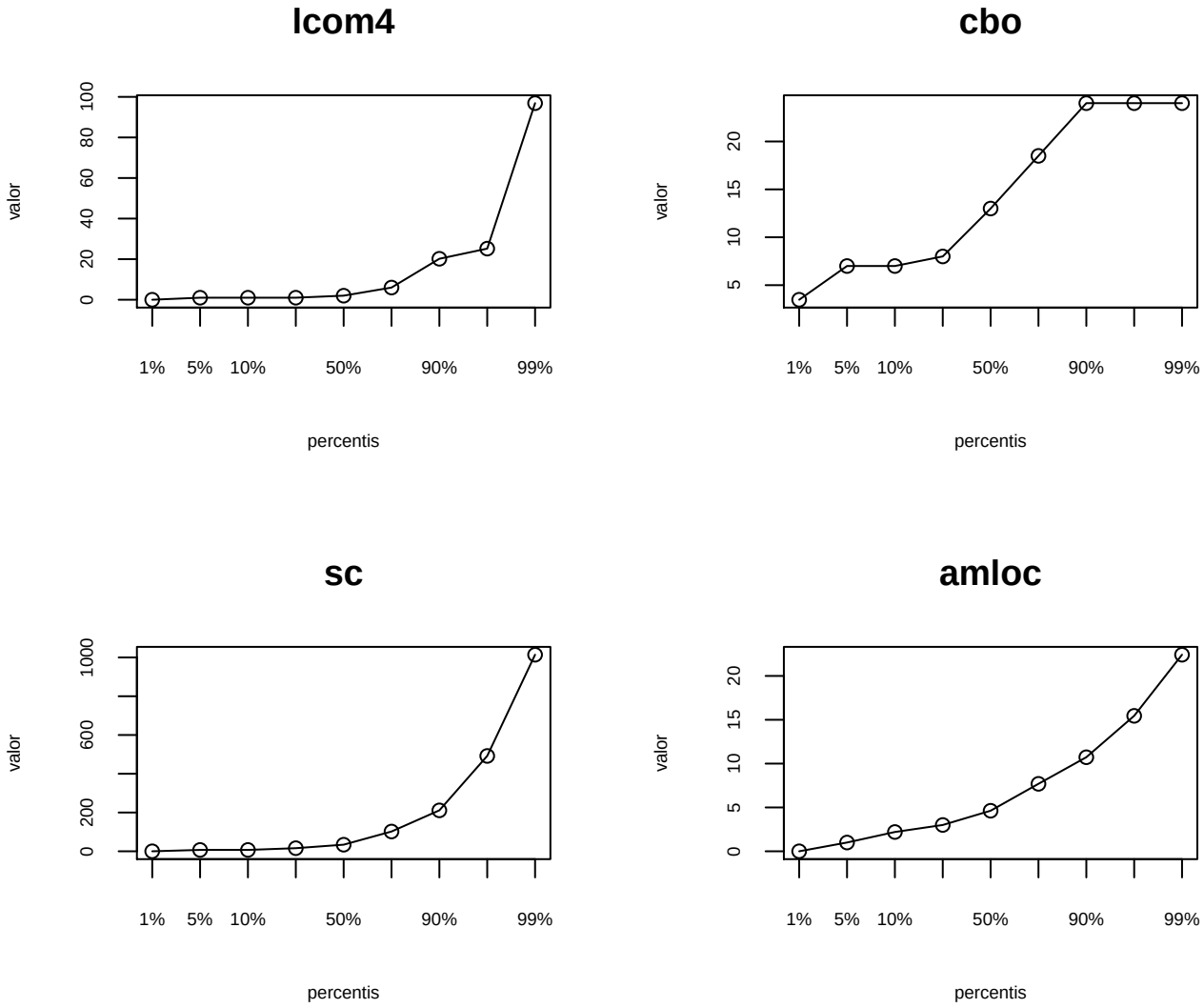

Figura 4.6 distribuição das métricas para a ferramenta jastadd

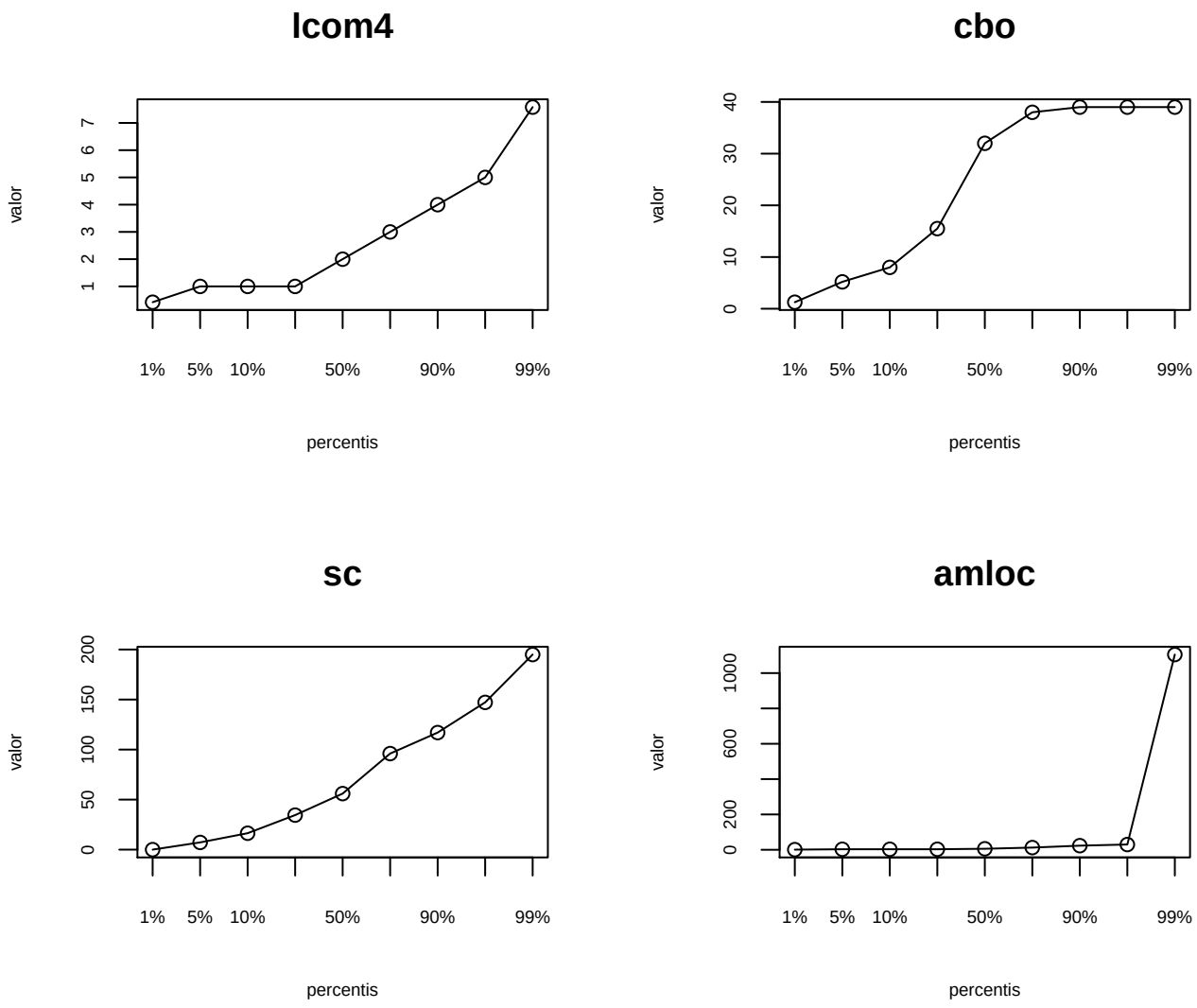

Figura 4.7 distribuição das métricas para a ferramenta source-meter

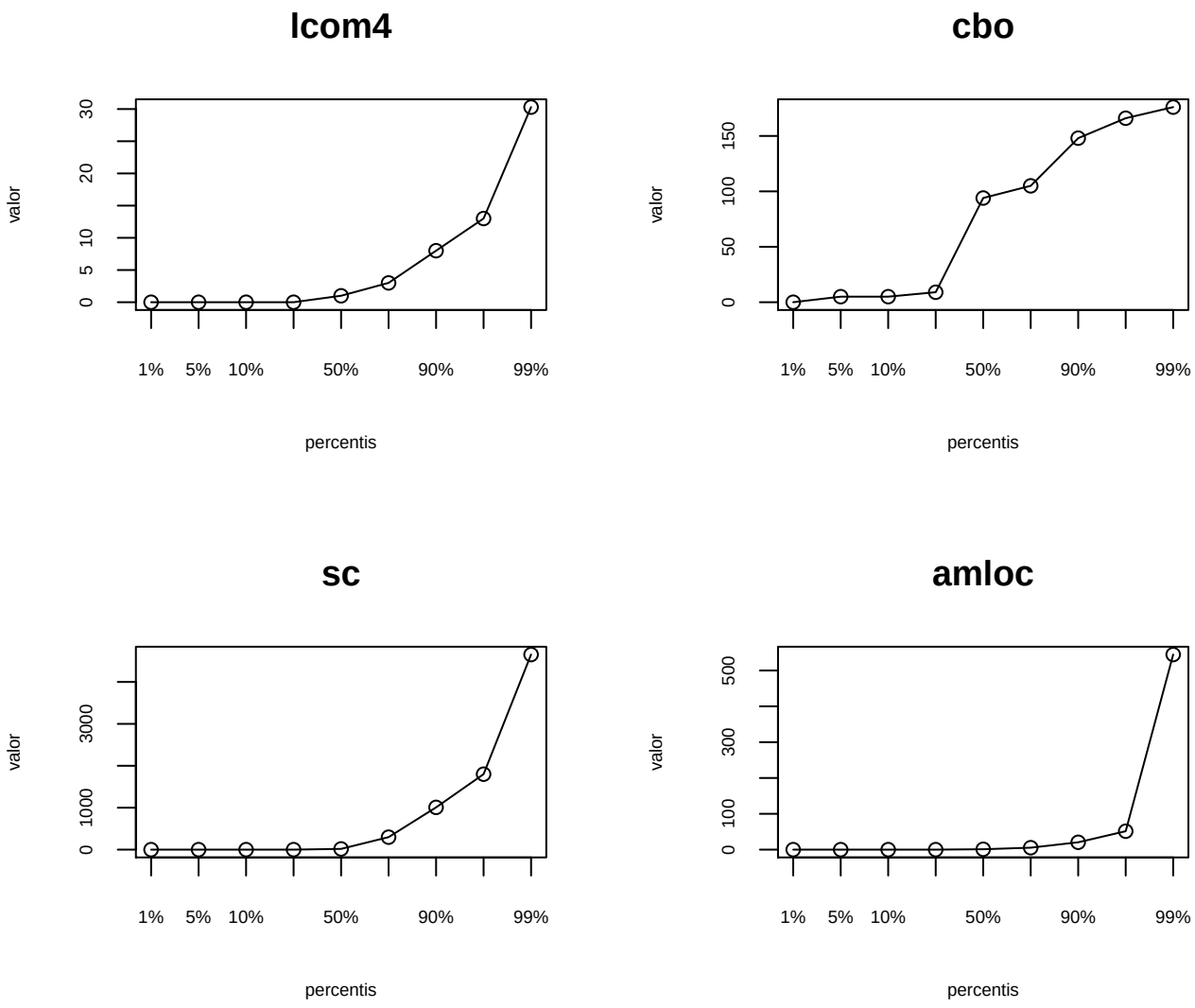

Figura 4.8 distribuição das métricas para a ferramenta srcml

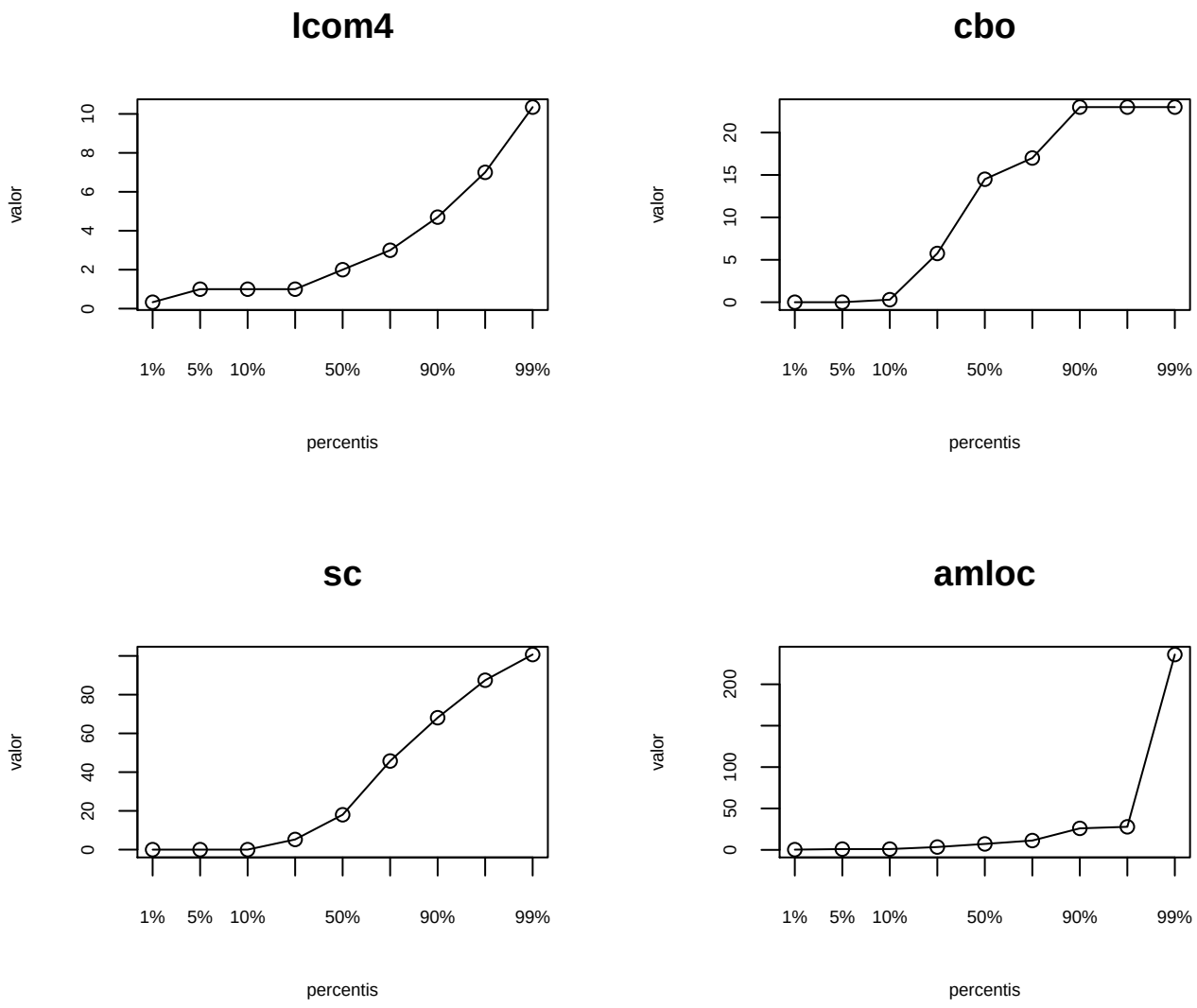

Figura 4.9 distribuição das métricas para a ferramenta tacle

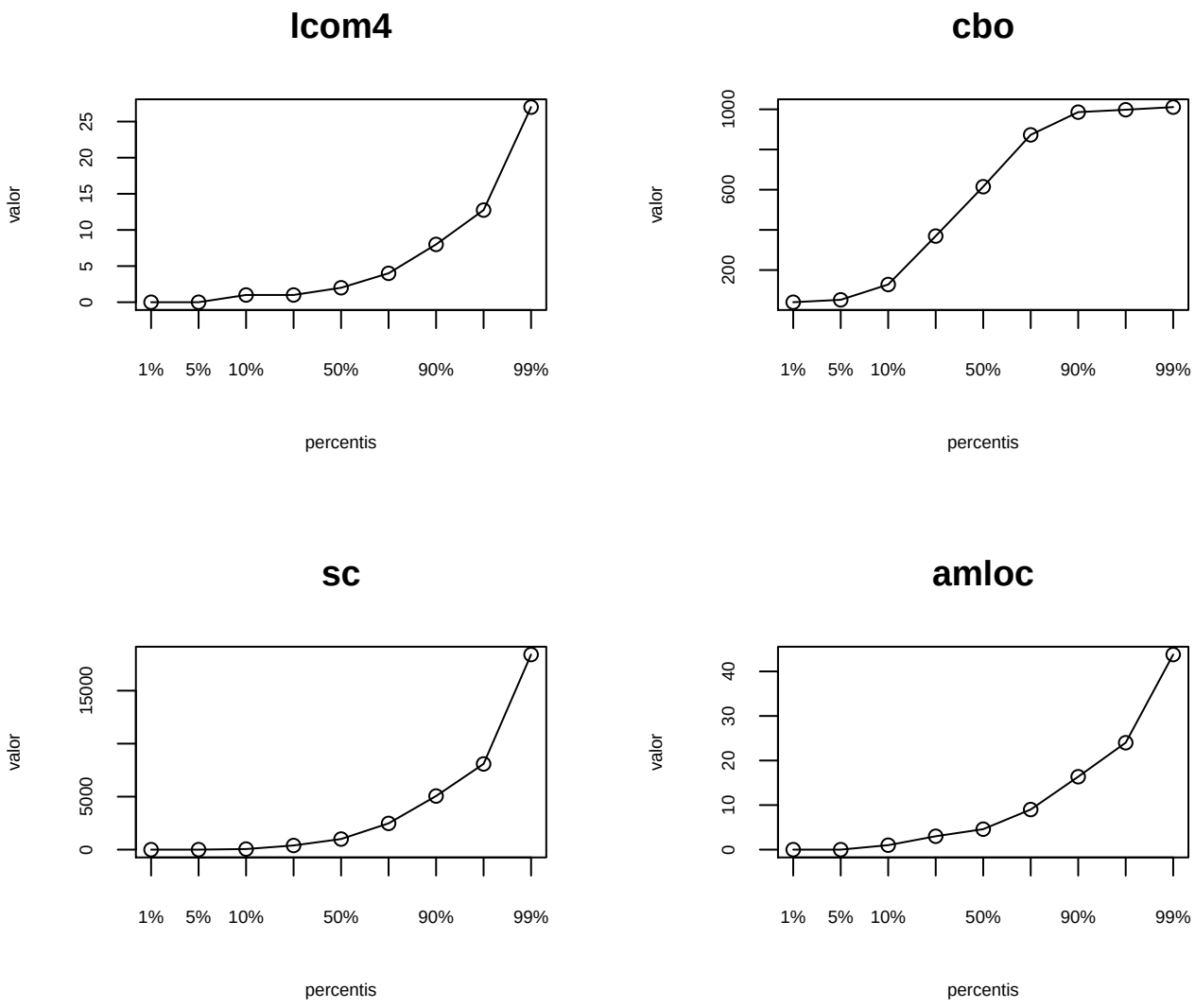

Figura 4.10 distribuição das métricas para a ferramenta wala

4.3.4 Gráficos das métricas para as ferramentas da indústria

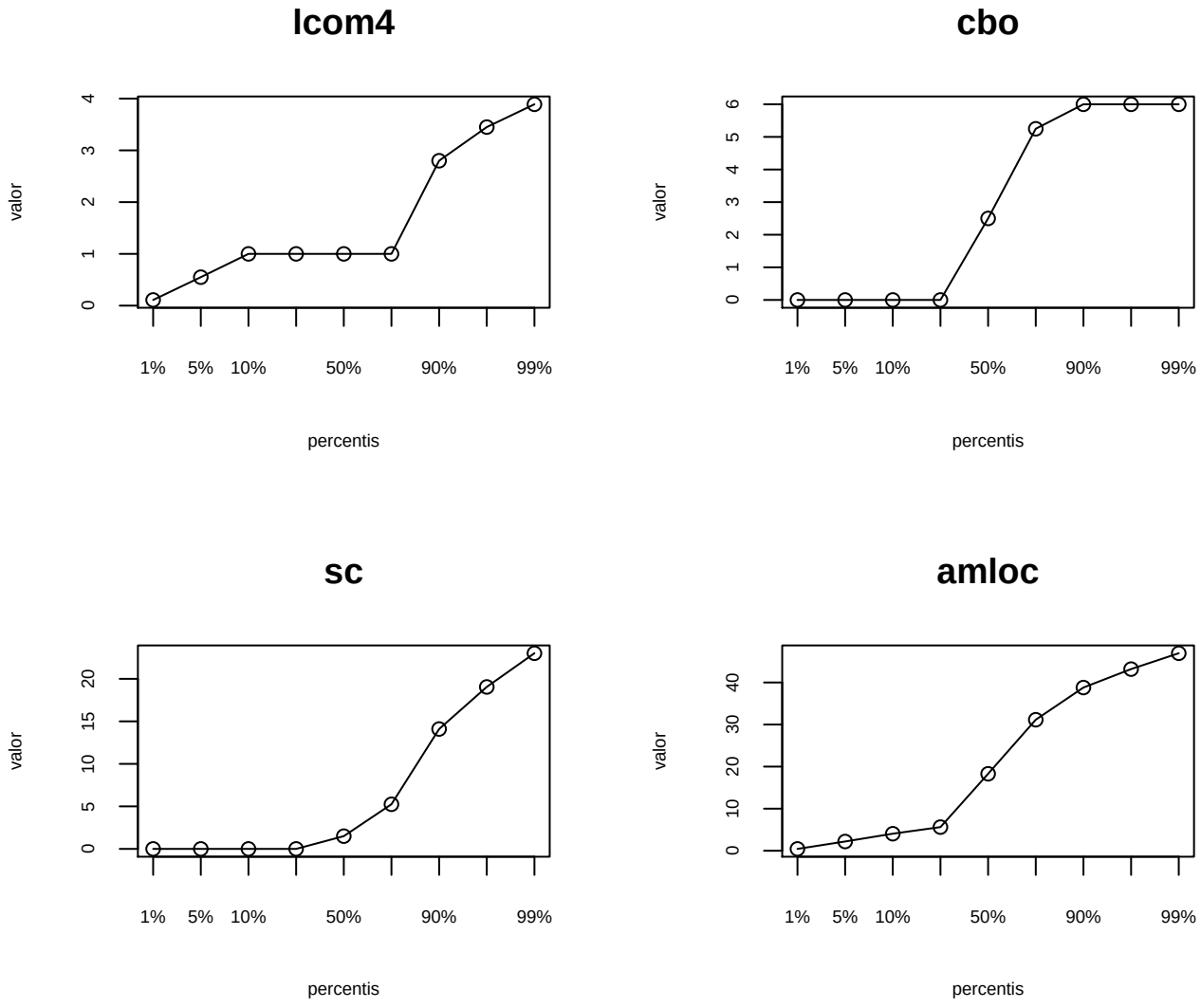

Figura 4.11 distribuição das métricas para a ferramenta boon

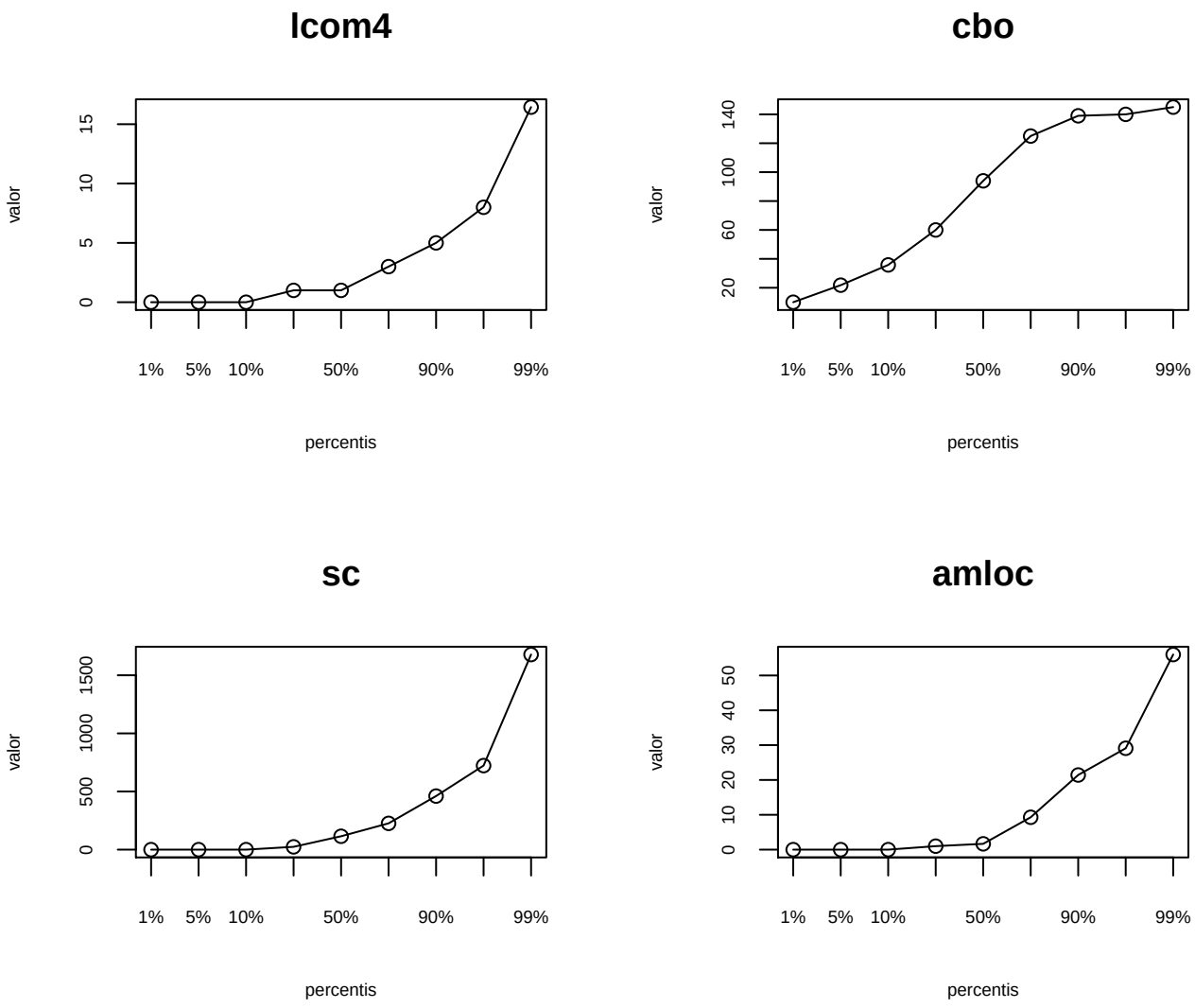

Figura 4.12 distribuição das métricas para a ferramenta clang

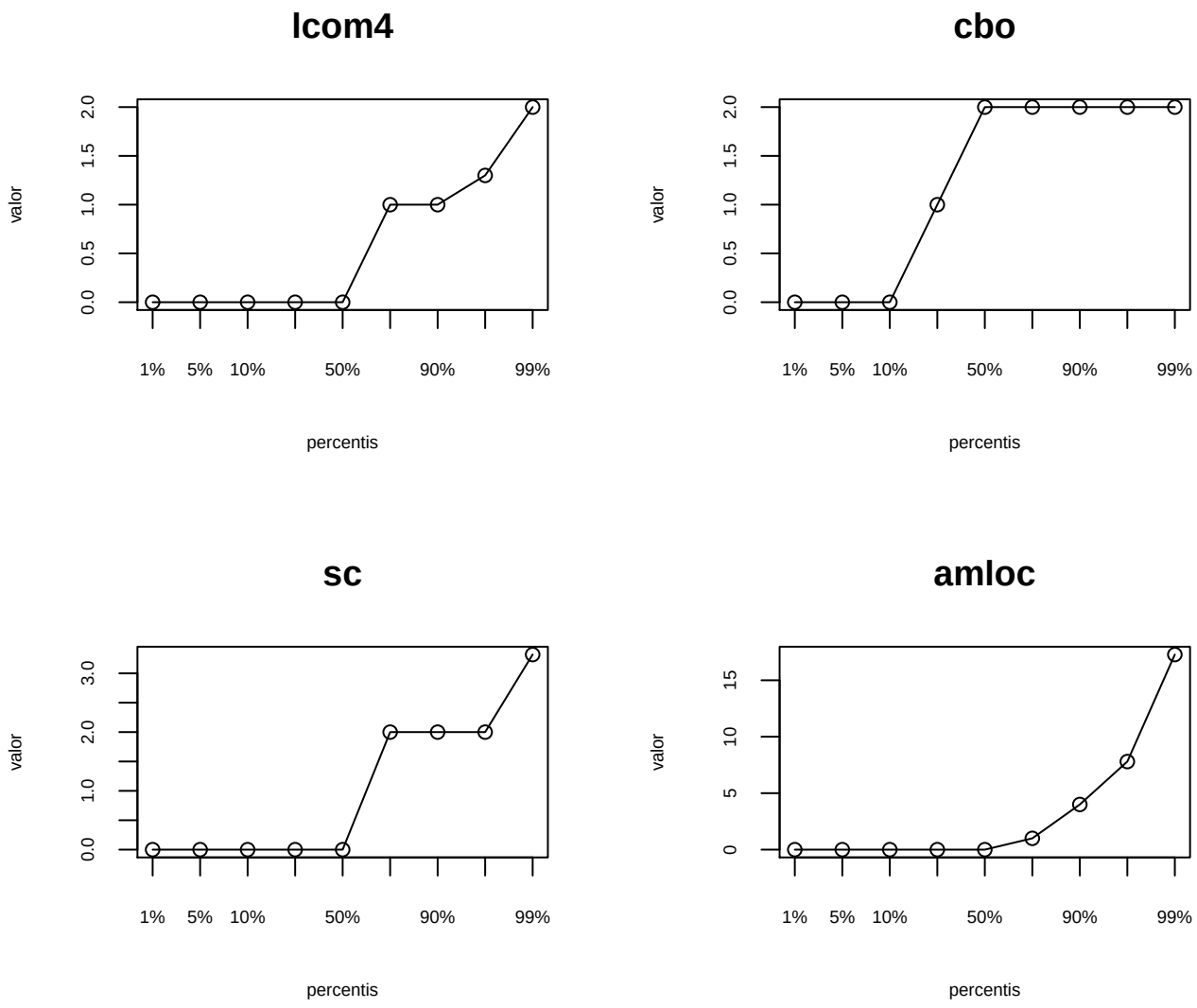

Figura 4.13 distribuição das métricas para a ferramenta closure-compiler

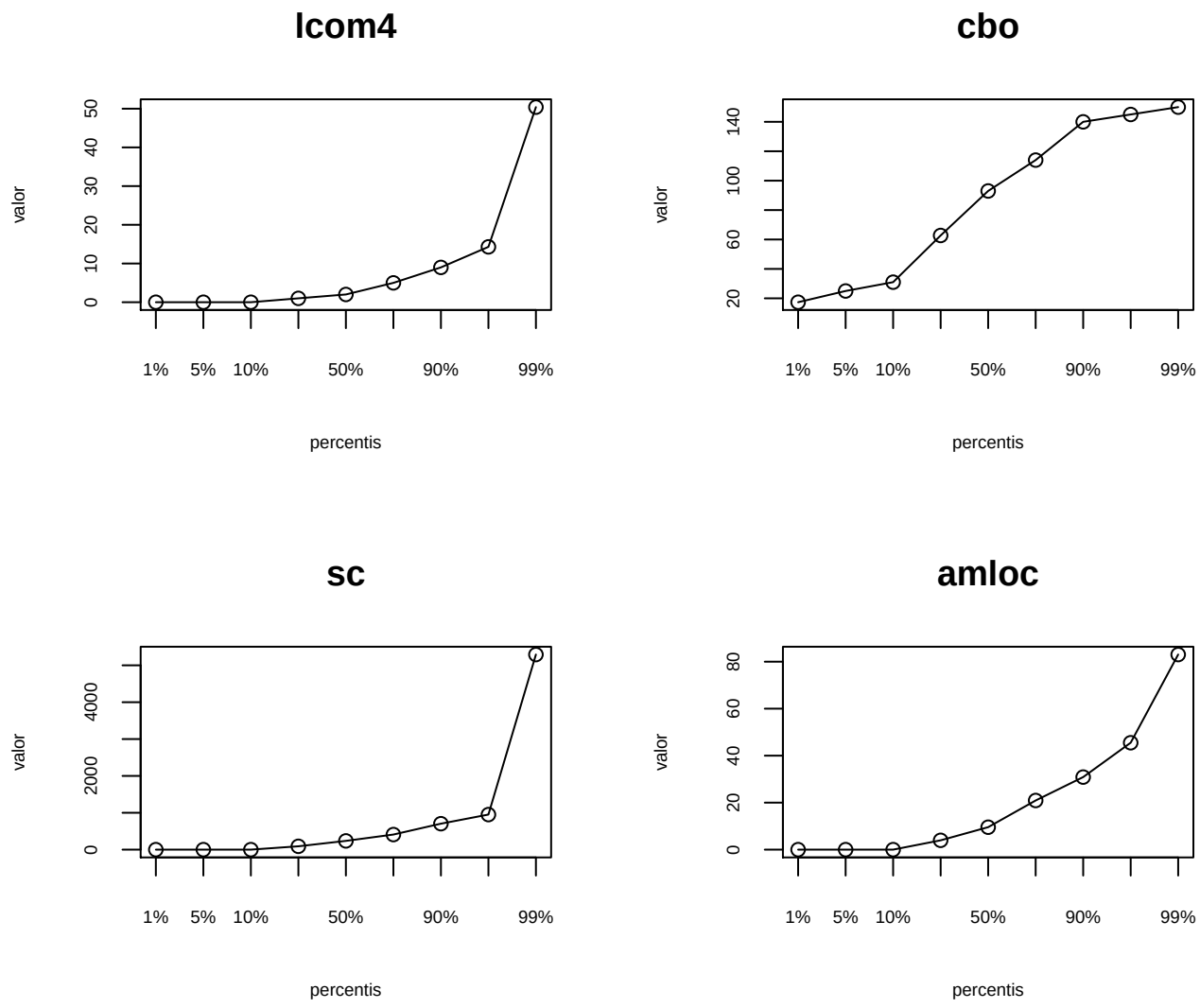

Figura 4.14 distribuição das métricas para a ferramenta cppcheck

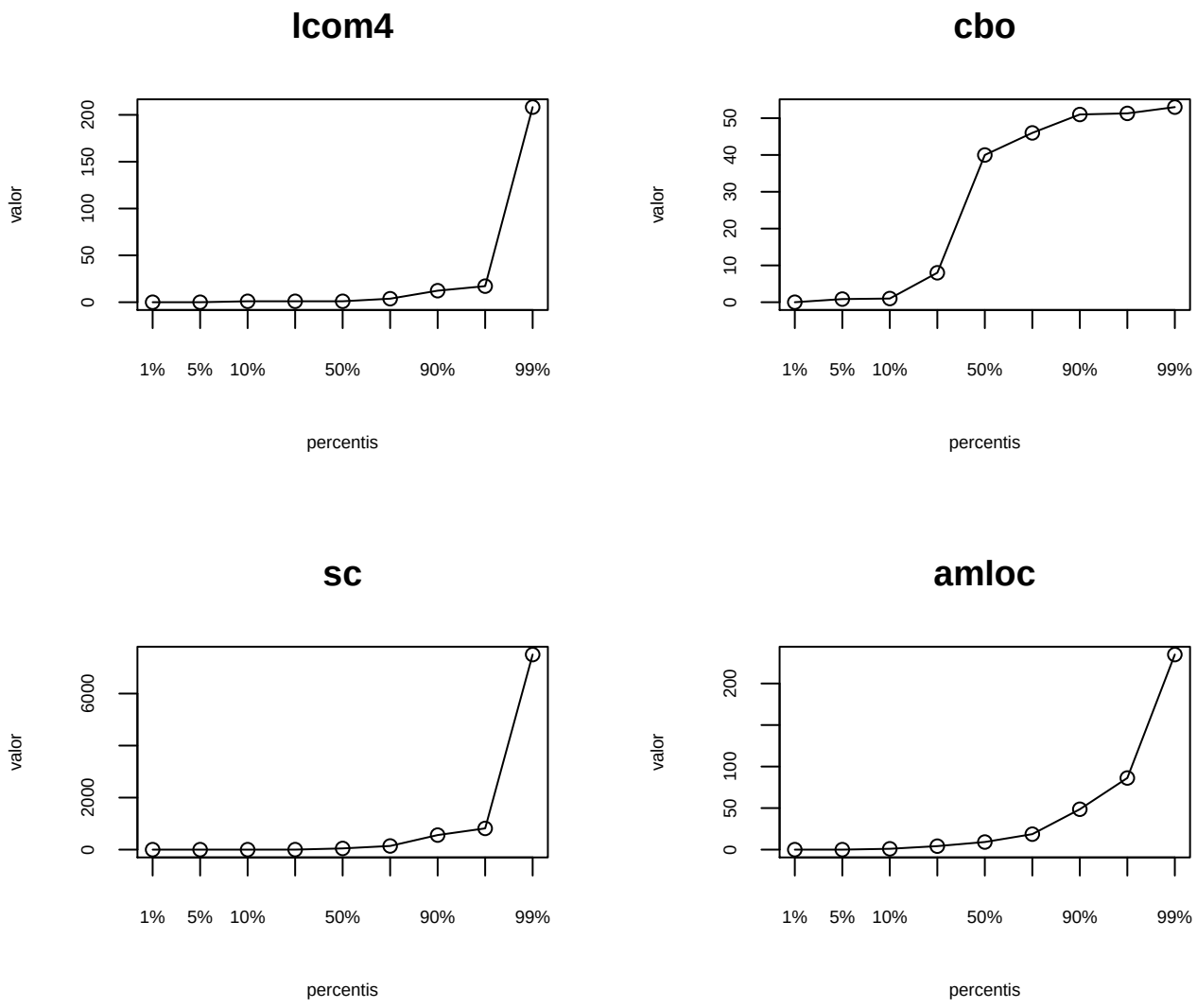

Figura 4.15 distribuição das métricas para a ferramenta cqual

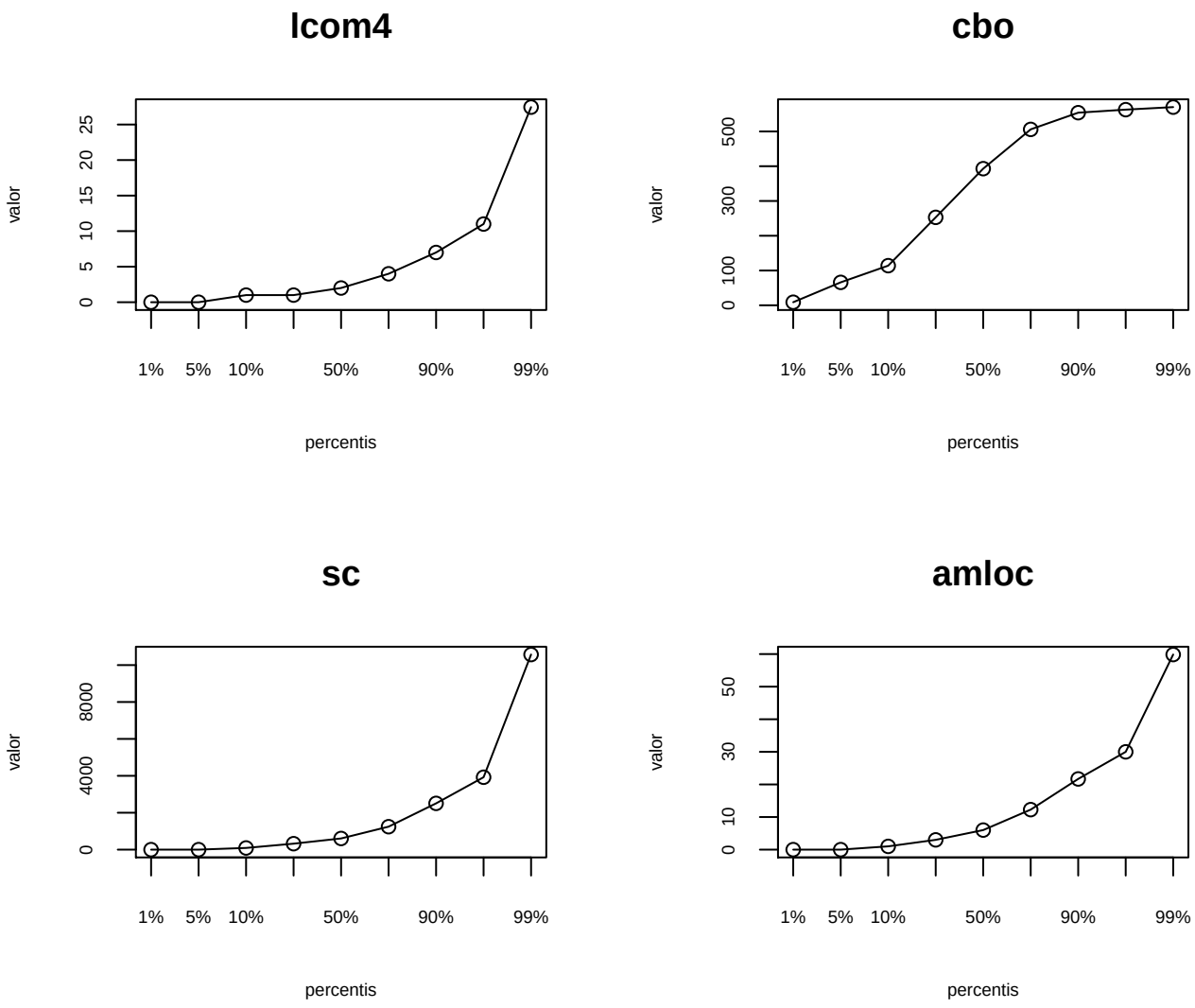

Figura 4.16 distribuição das métricas para a ferramenta findbugs

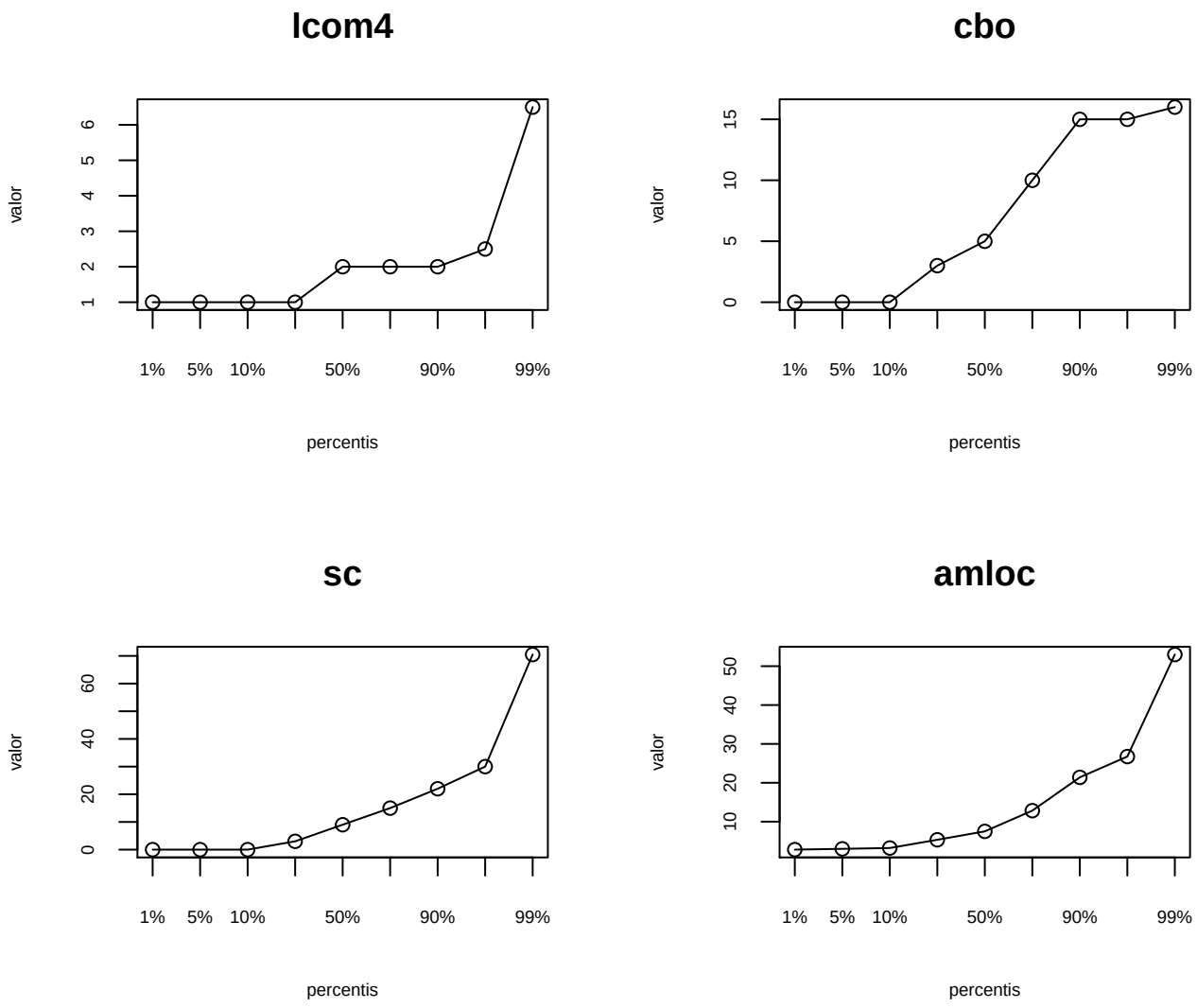

Figura 4.17 distribuição das métricas para a ferramenta findsecuritybugs

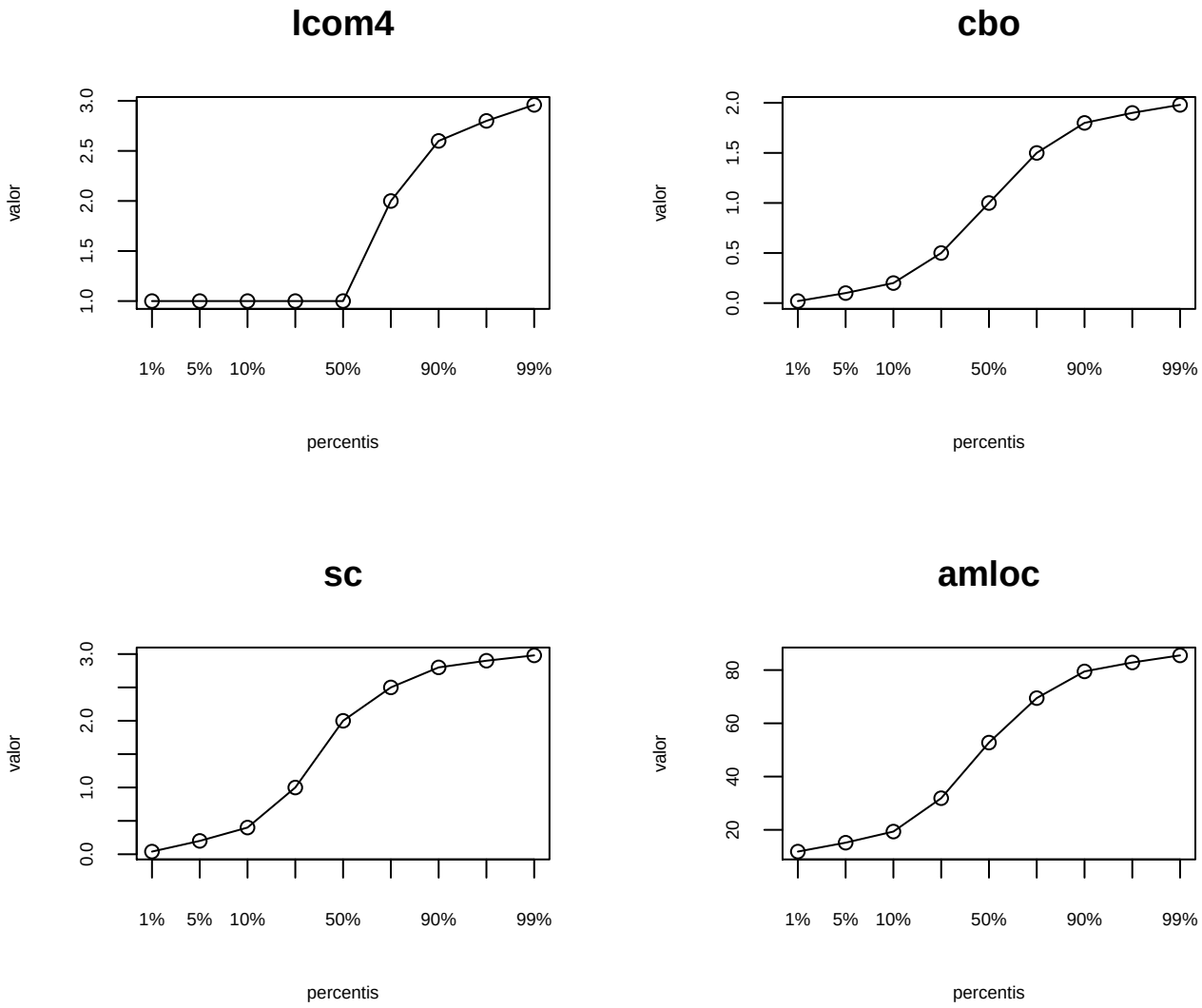

Figura 4.18 distribuição das métricas para a ferramenta jlint

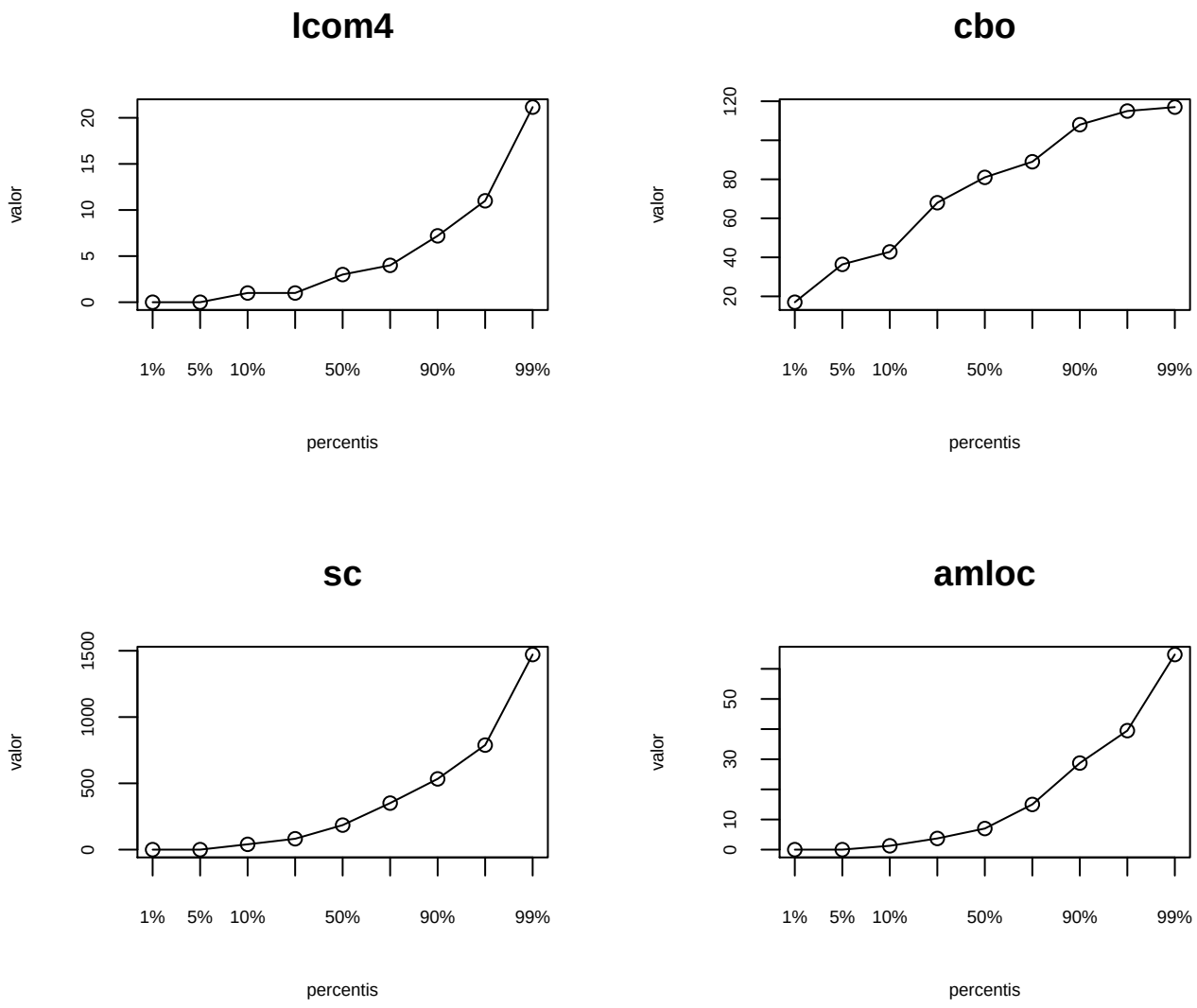

Figura 4.19 distribuição das métricas para a ferramenta pixy

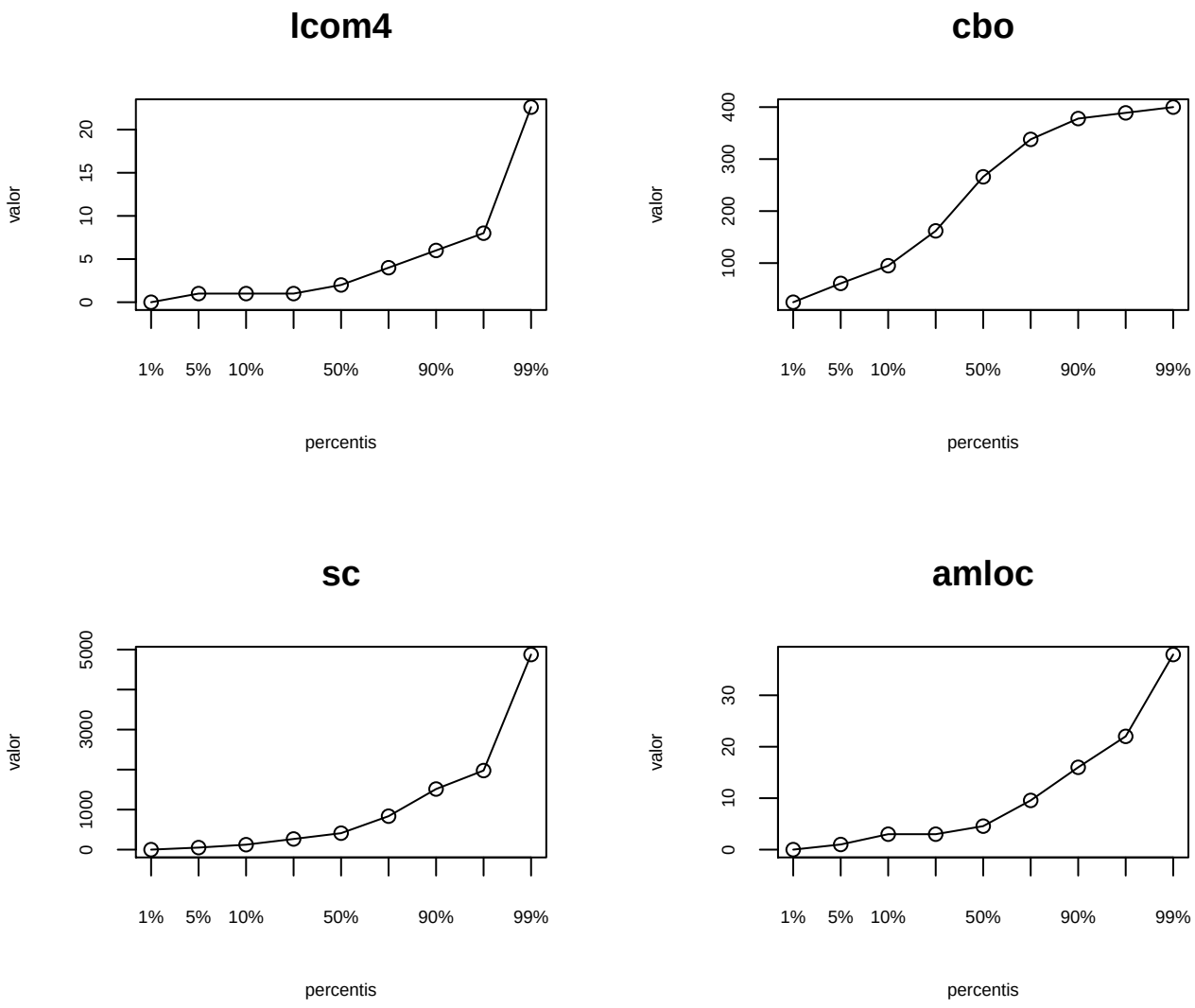

Figura 4.20 distribuição das métricas para a ferramenta pmd

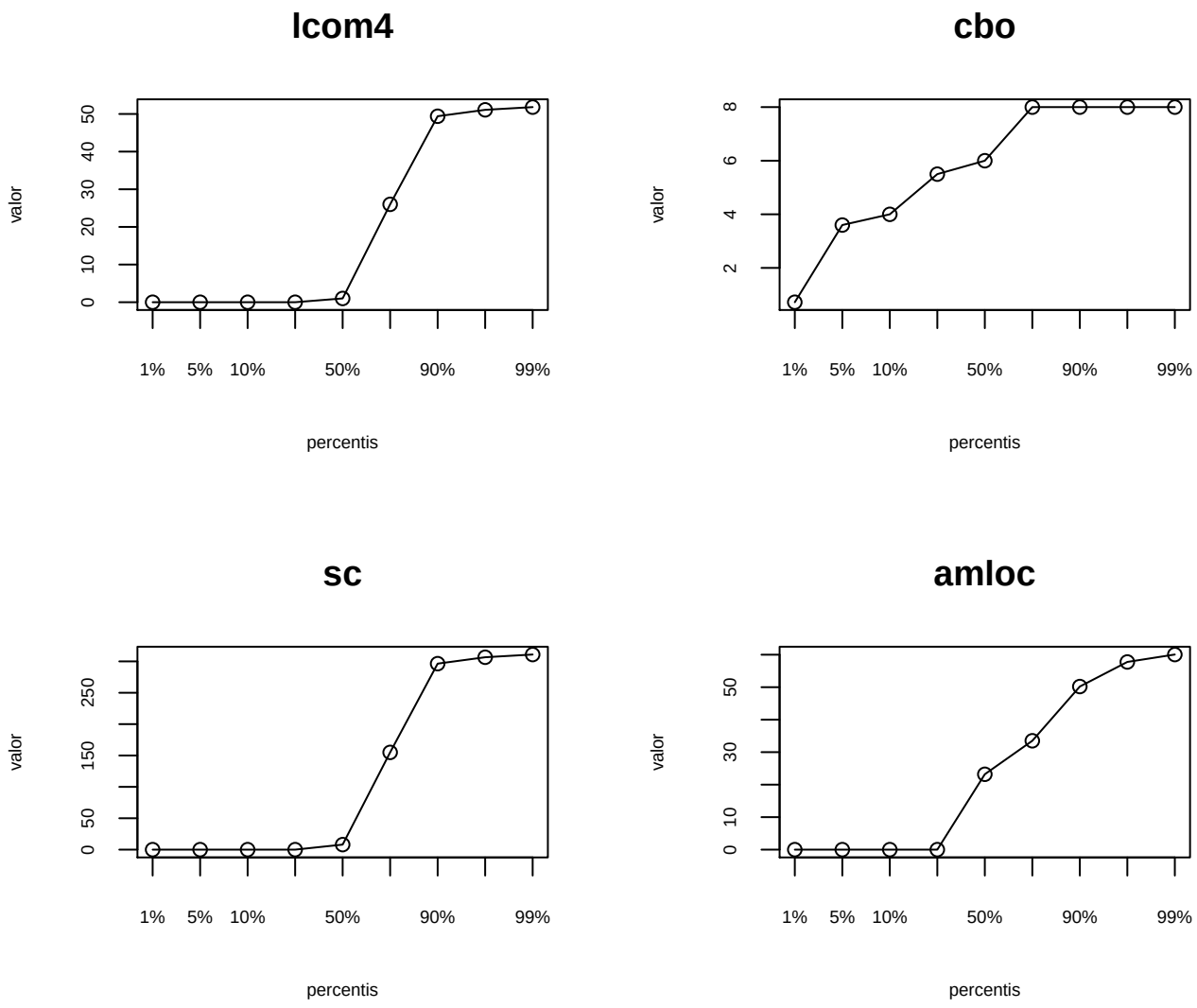

Figura 4.21 distribuição das métricas para a ferramenta rats

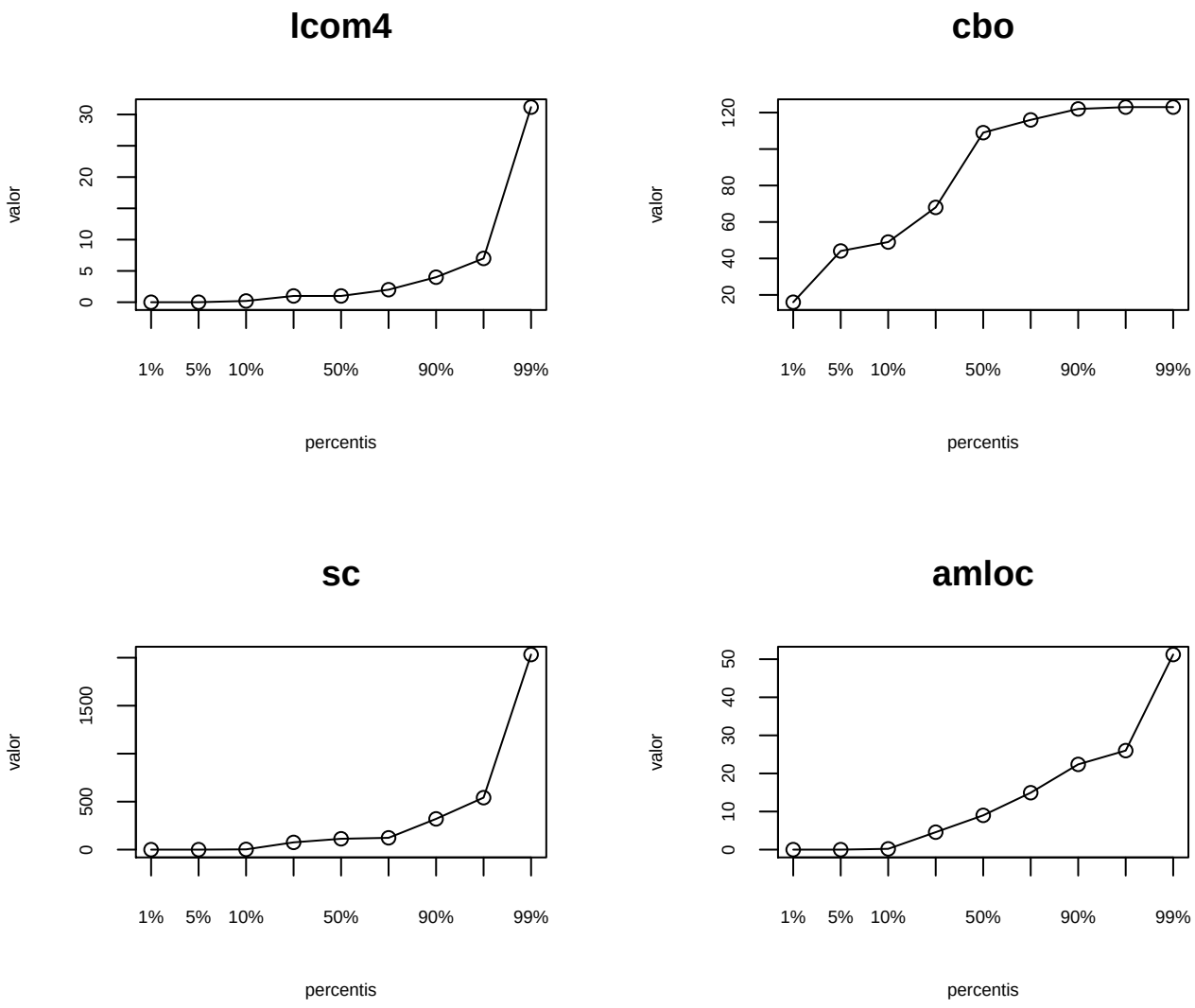

Figura 4.22 distribuição das métricas para a ferramenta smatch

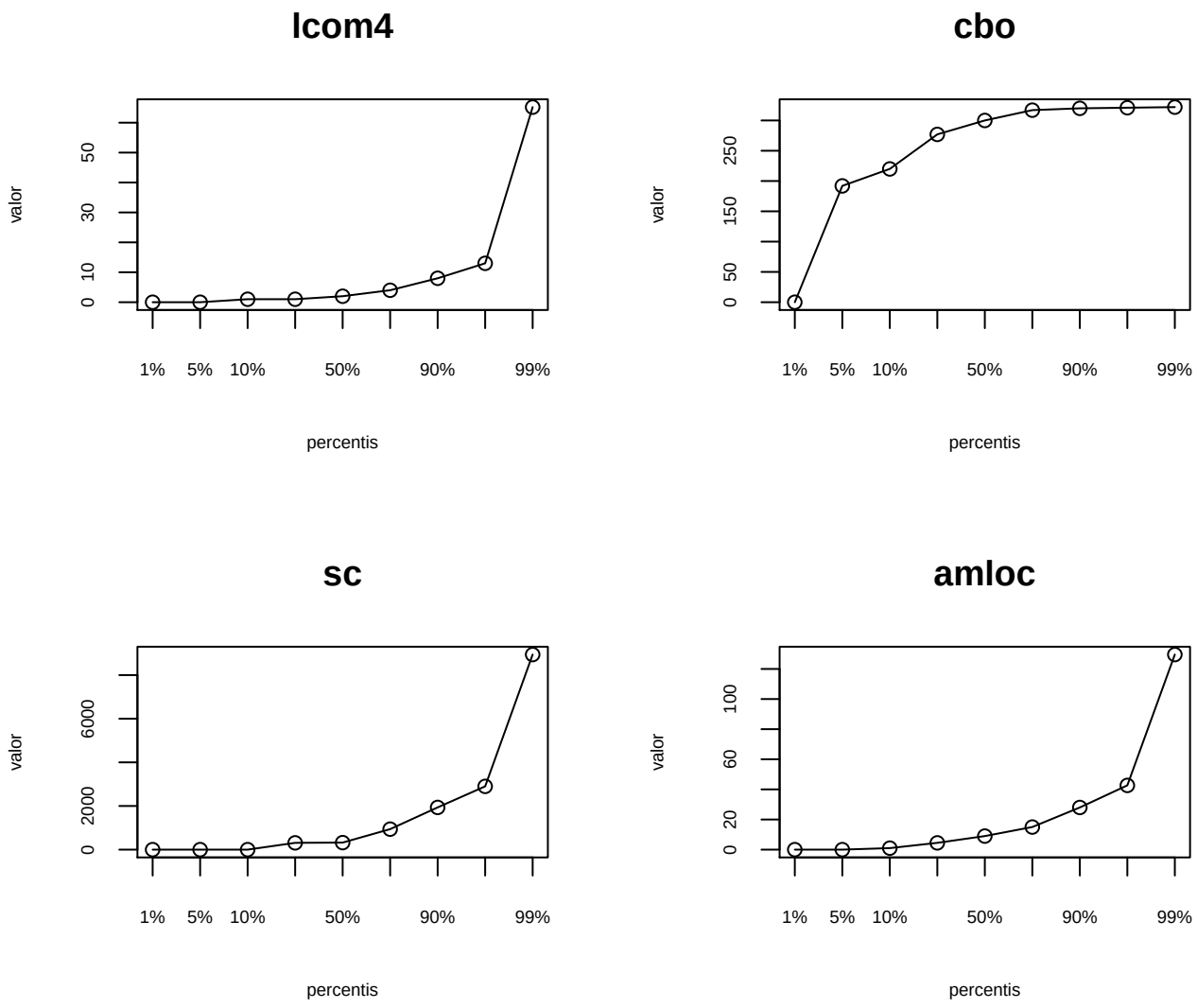

Figura 4.23 distribuição das métricas para a ferramenta splint

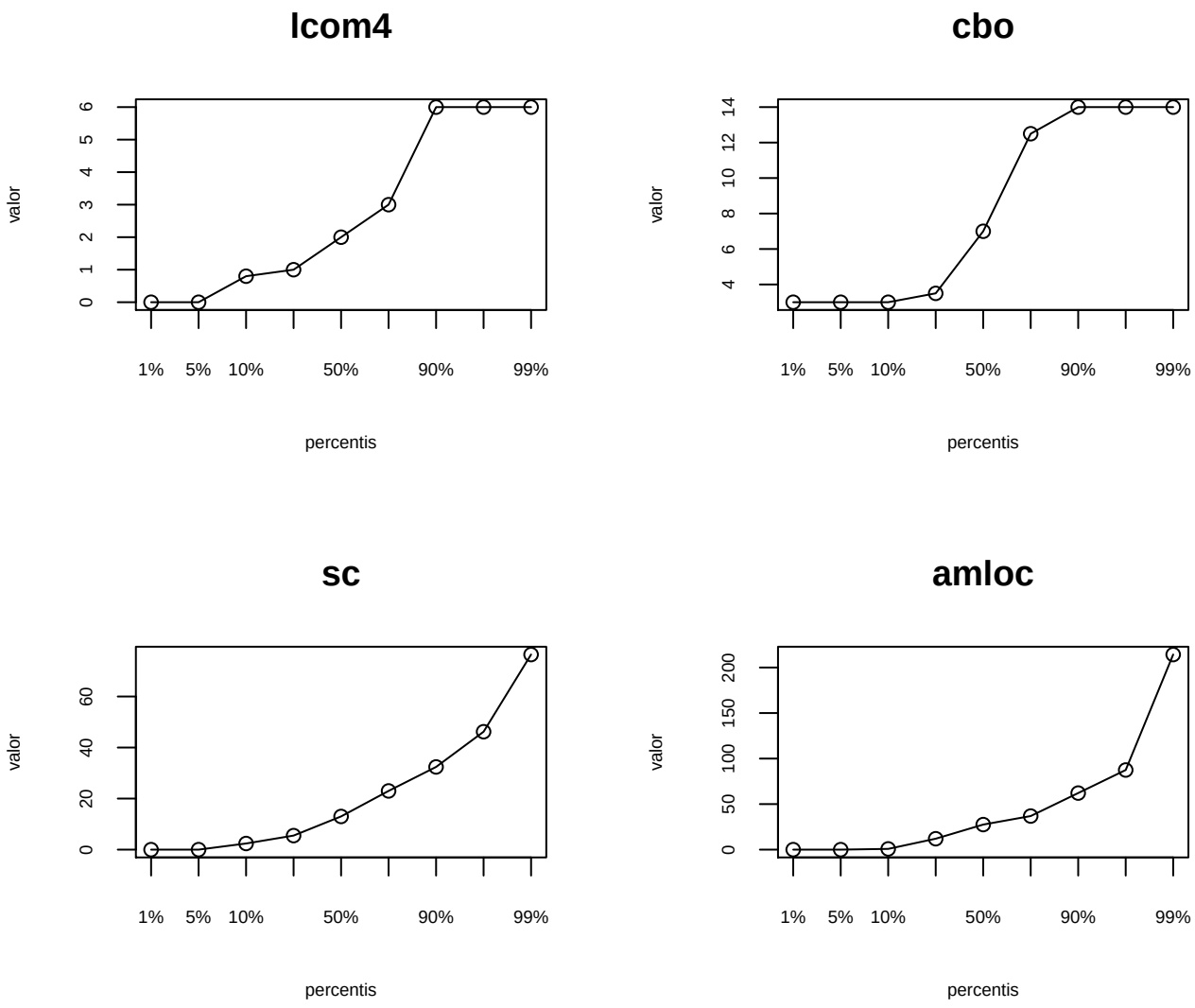

Figura 4.24 distribuição das métricas para a ferramenta uno

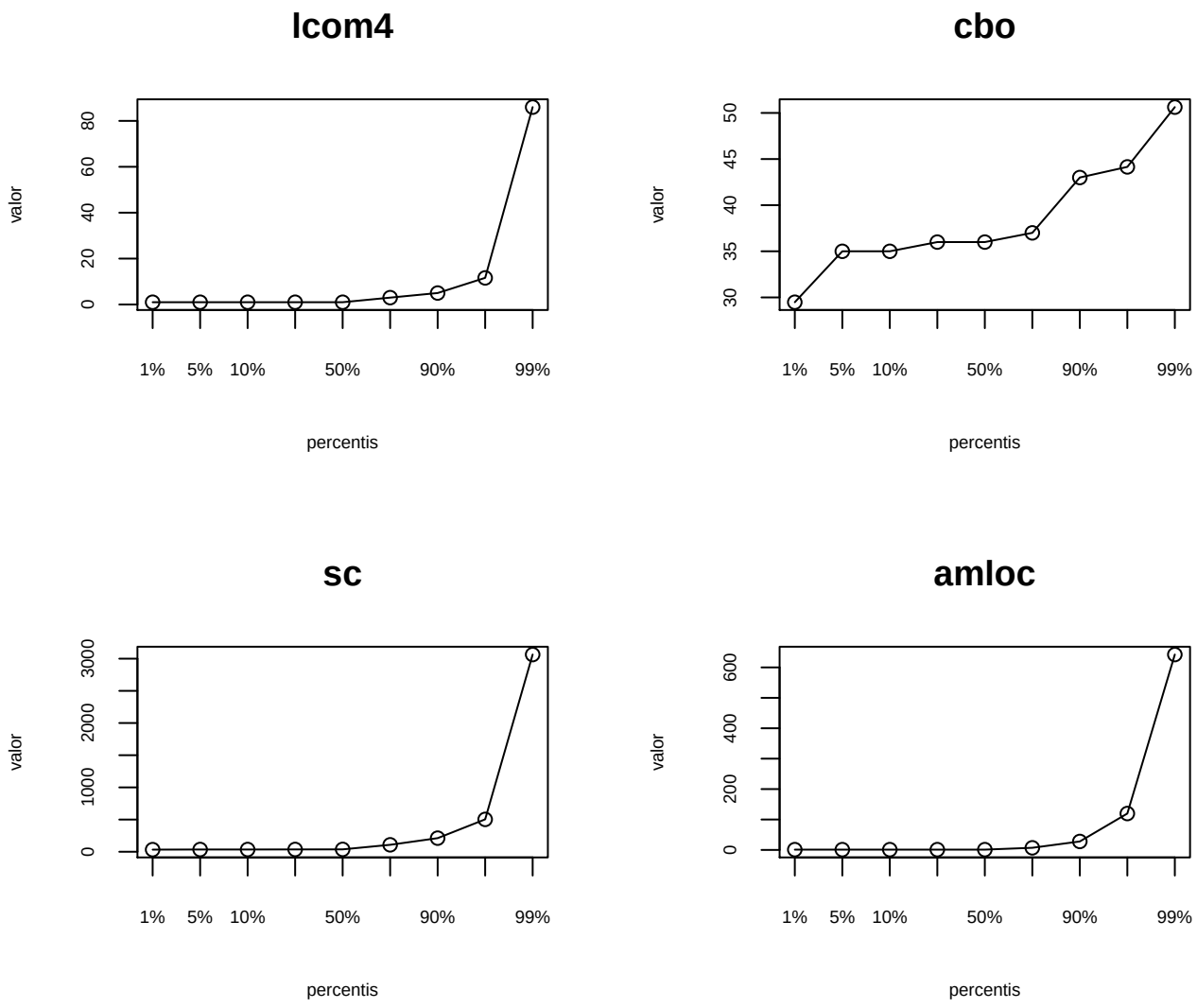

Figura 4.25 distribuição das métricas para a ferramenta wap

4.3.5 Gráfico comparativo para as ferramentas academicas

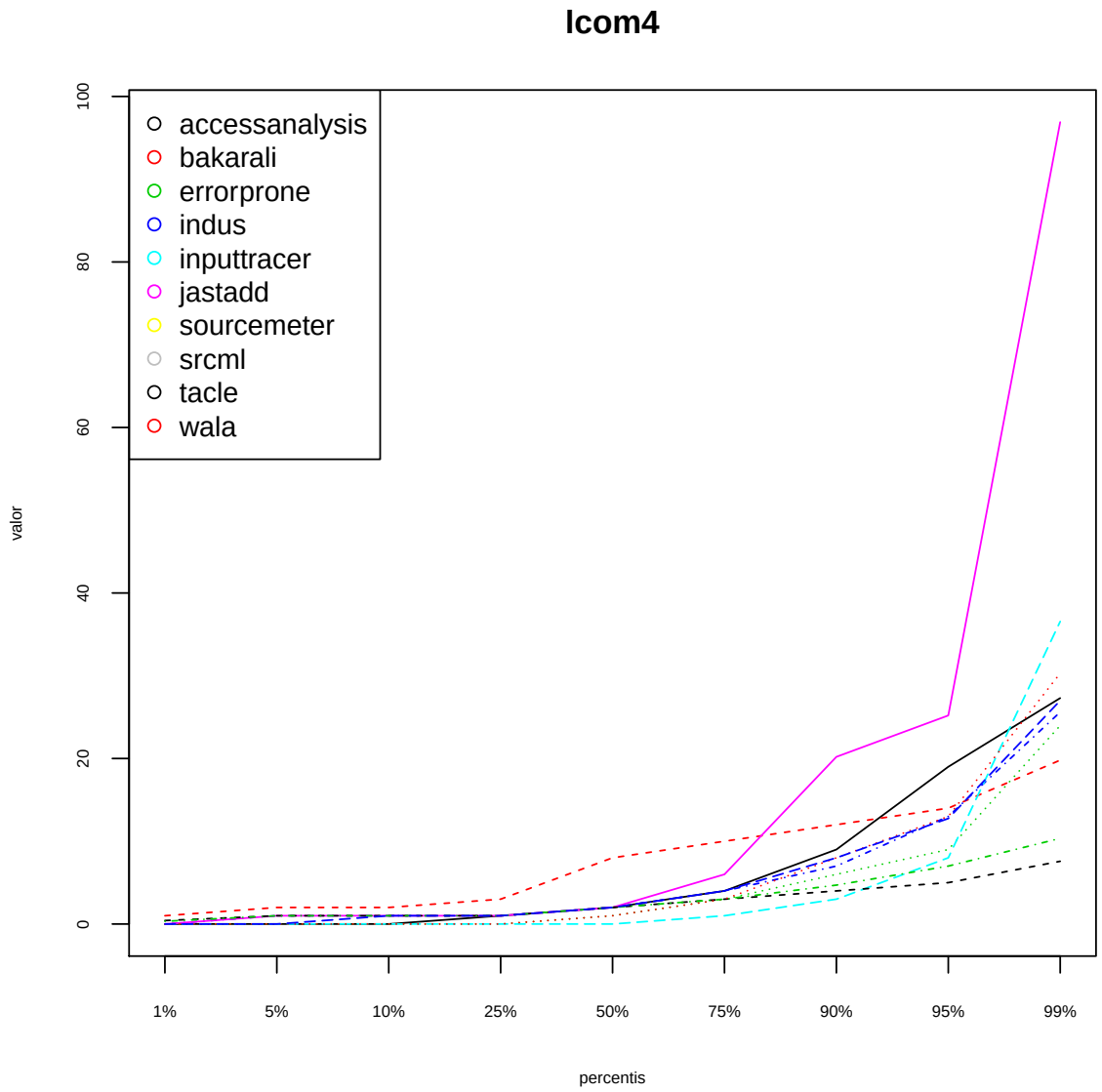

Figura 4.26 distribuição para as ferramentas da academia

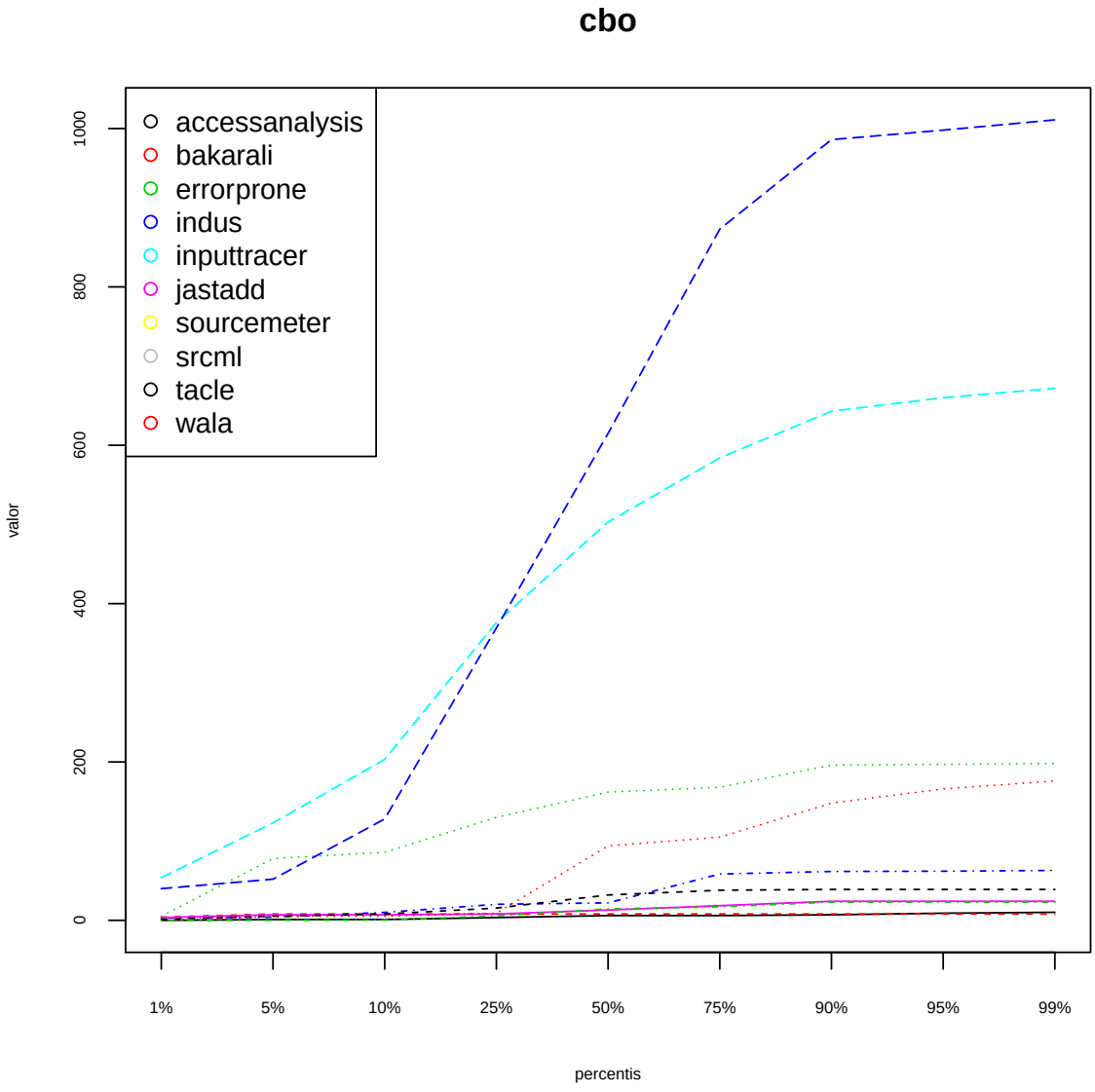

Figura 4.27 distribuição para as ferramentas da academia

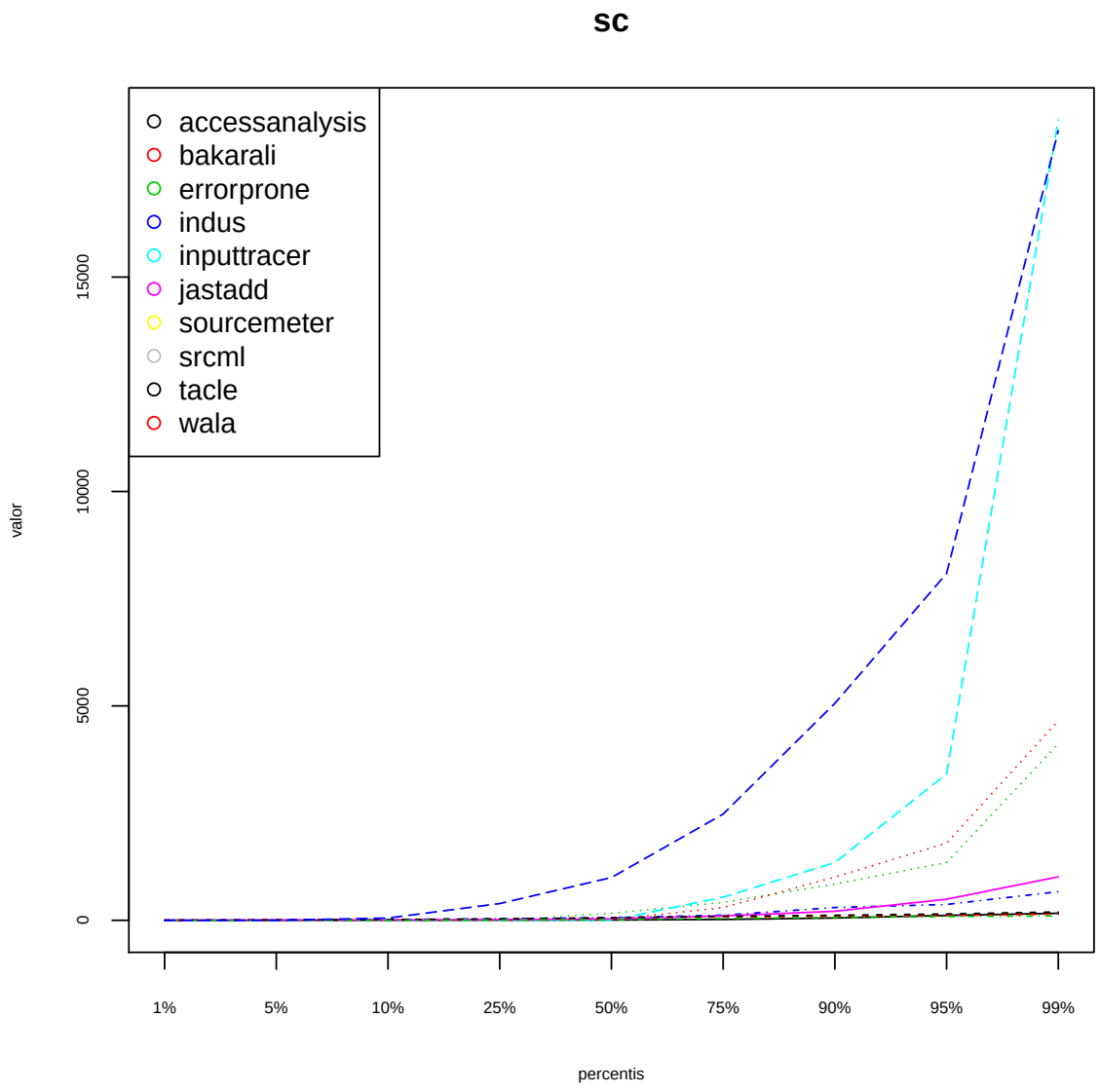

Figura 4.28 distribuição para as ferramentas da academia

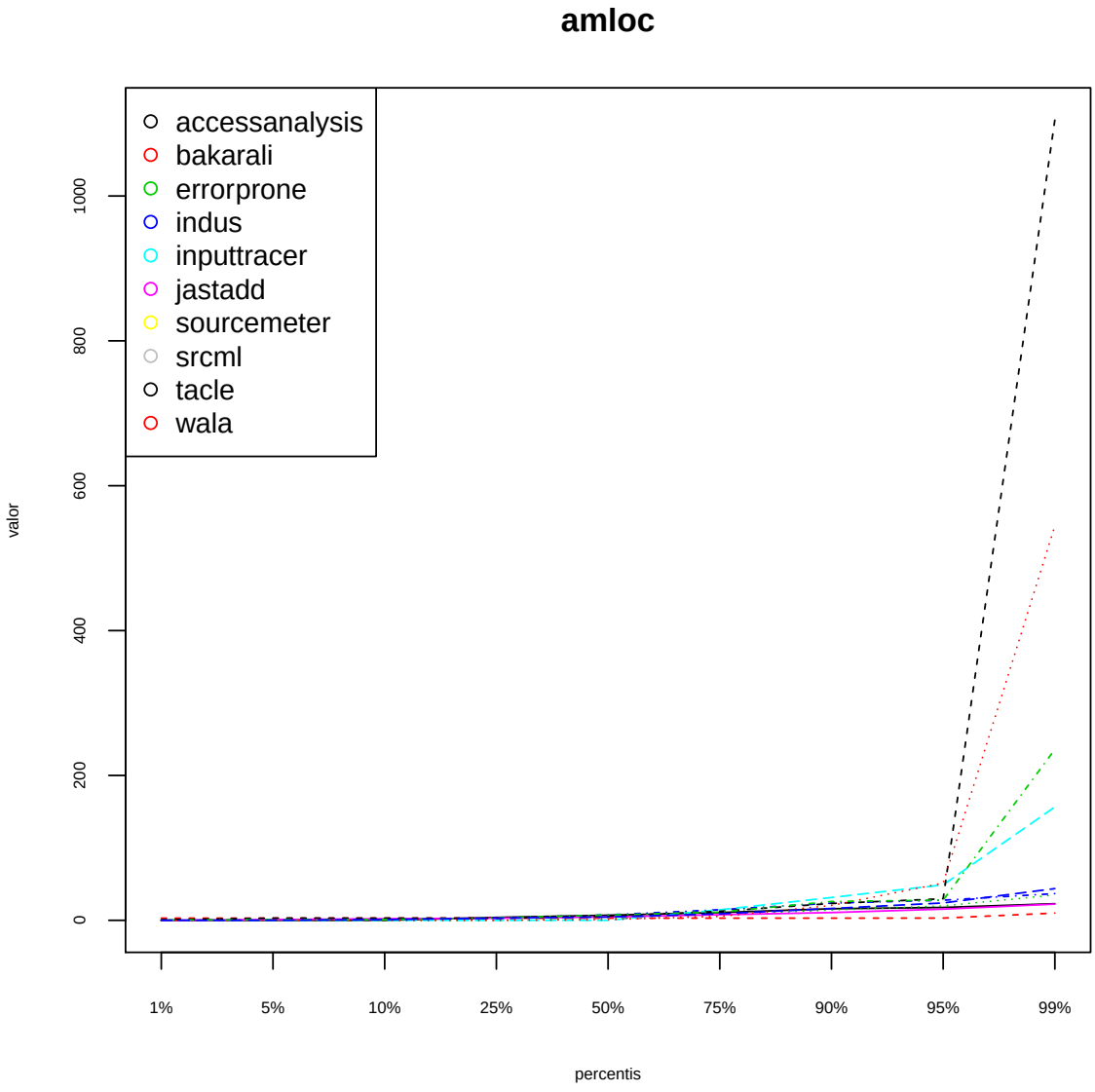

Figura 4.29 distribuição para as ferramentas da academia

4.3.6 Gráfico comparativo para as ferramentas da indústria

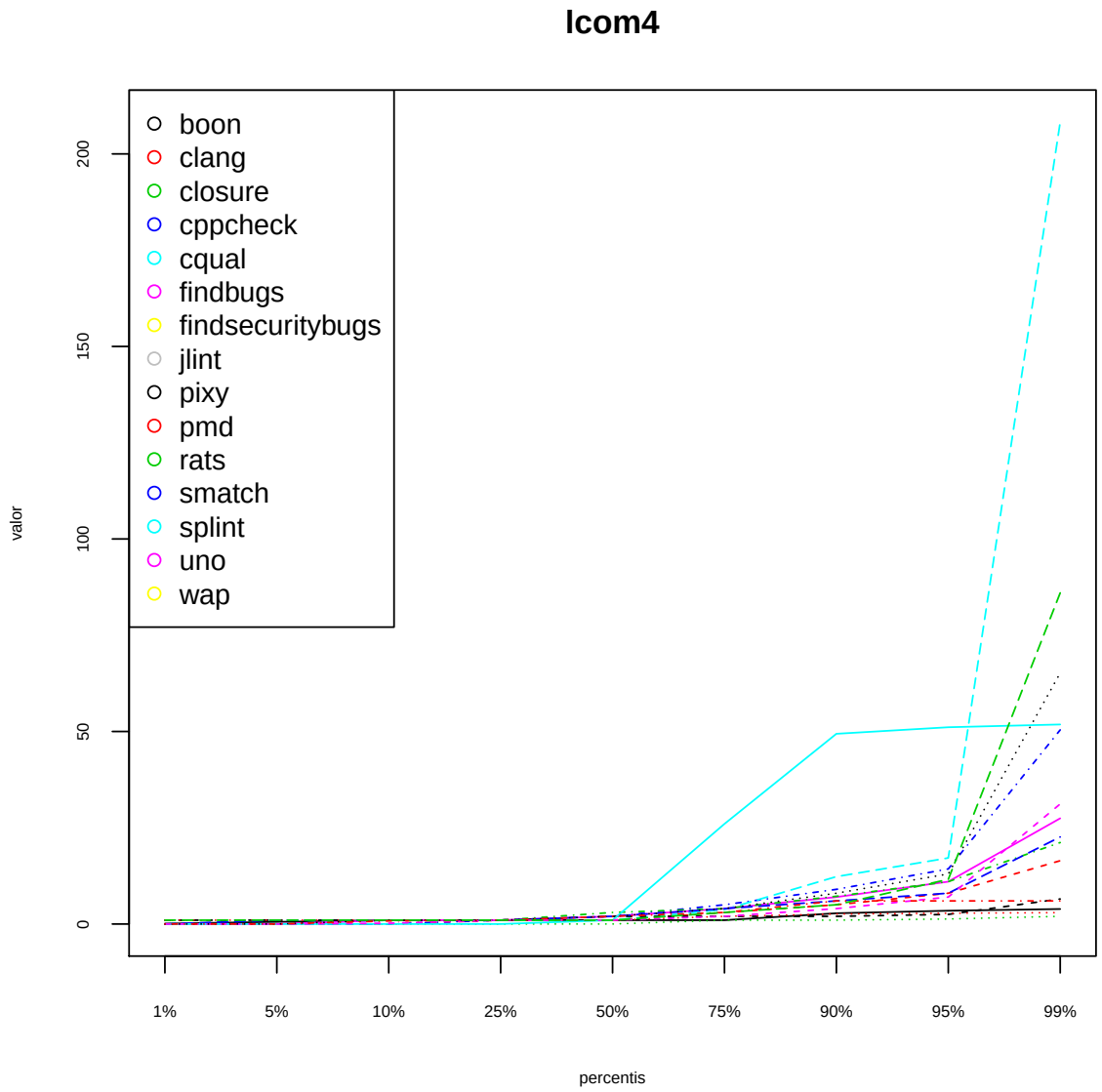

Figura 4.30 distribuição para as ferramentas da indústria

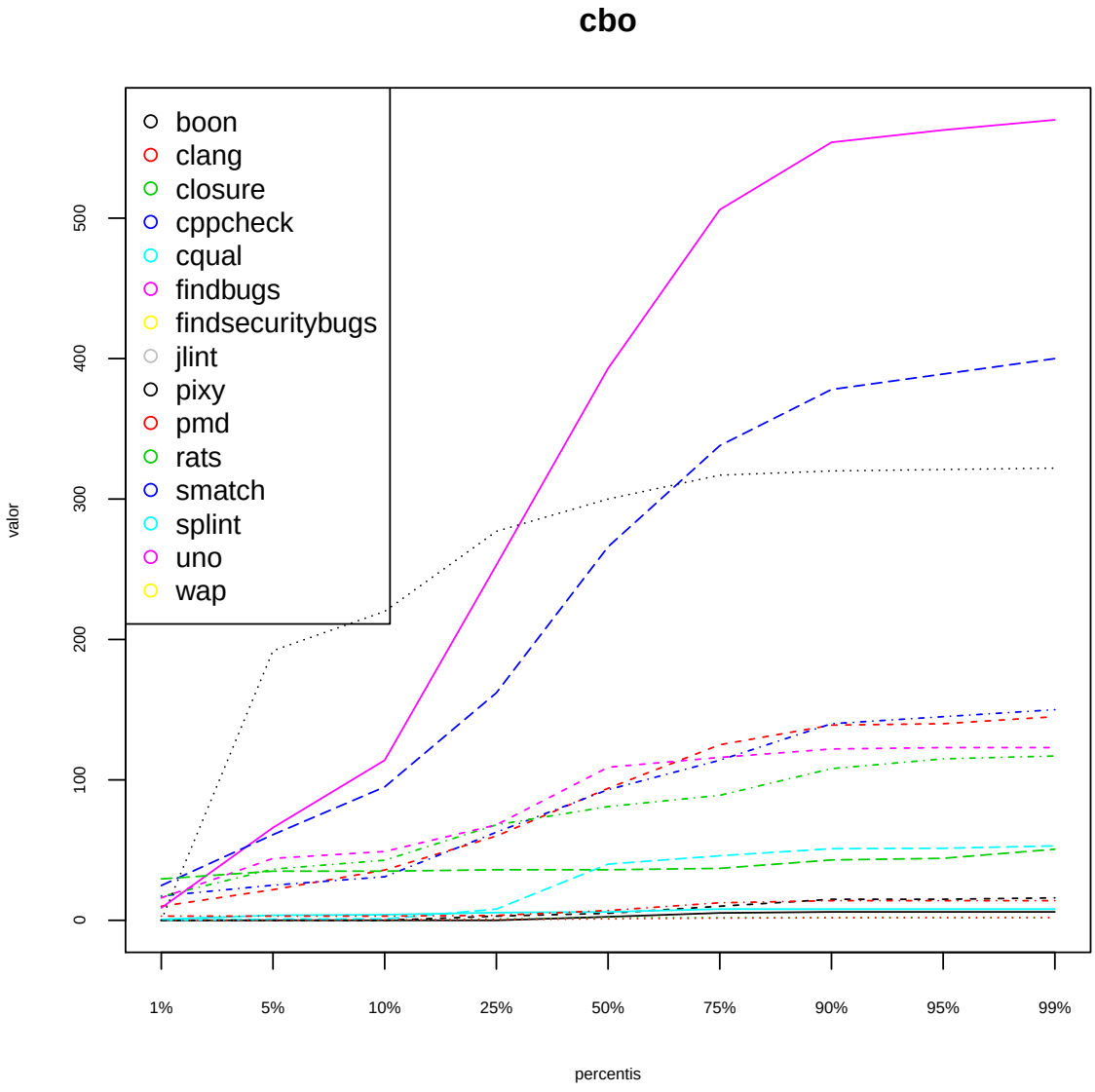

Figura 4.31 distribuição para as ferramentas da indústria

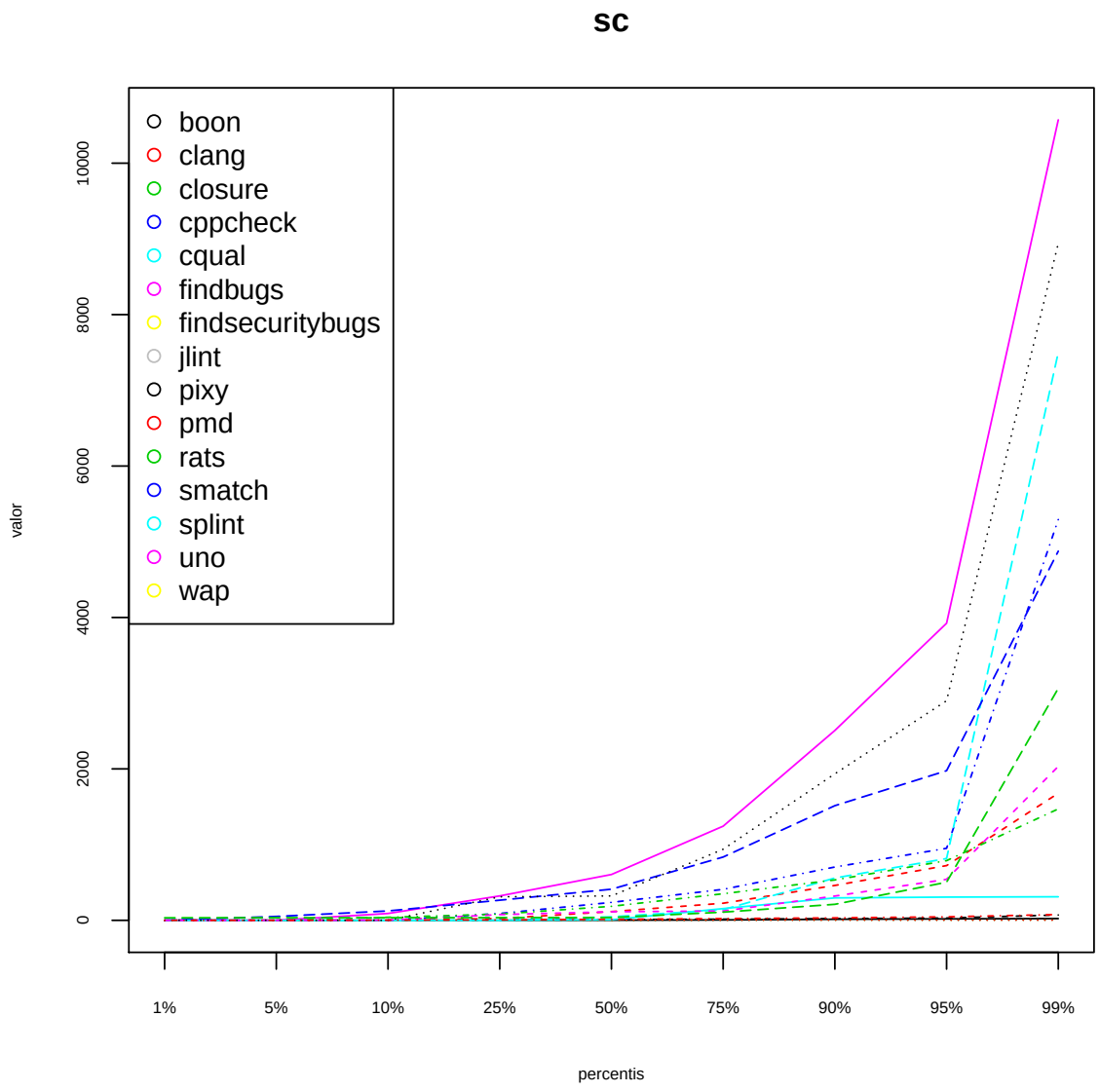

Figura 4.32 distribuição para as ferramentas da indústria

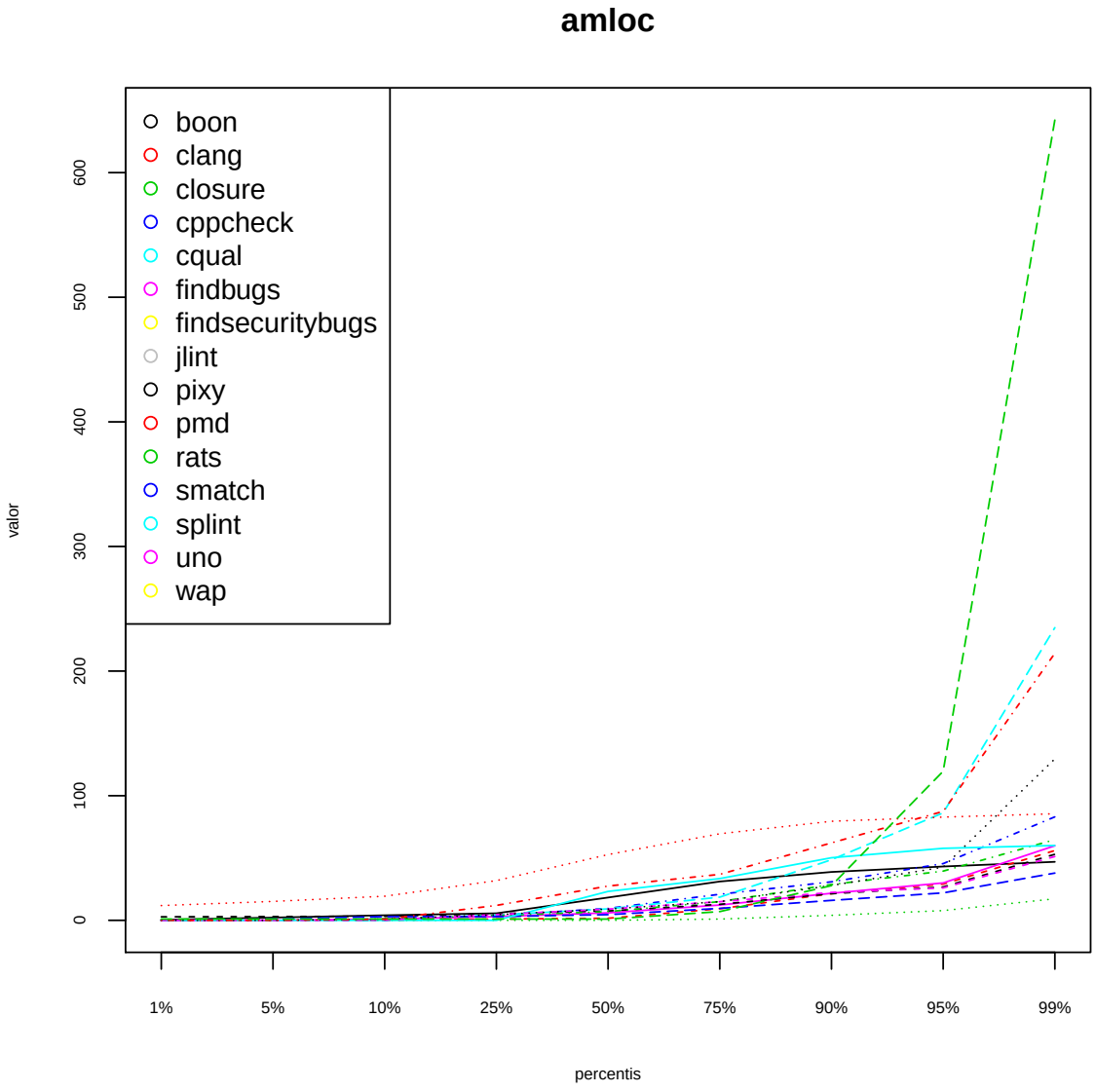

Figura 4.33 distribuição para as ferramentas da indústria

4.4 EVOLUÇÃO INICIAL DA FERRAMENTA ANALIZO

(pendente) “apenas se der tempo”.

CAPÍTULO 5

CONCLUSÃO

(pentende) “discutir as contribuições dando resposta ao que foi colocado na introdução, fizemos isto, conseguimos estes resultados, e iremos fazer aquilo nos proximos passos.”

5.1 LIMITAÇÕES DO TRABALHO

(pendente)

5.2 TRABALHOS FUTUROS

(pendente)

REFERÊNCIAS BIBLIOGRÁFICAS

BINKLEY, D. Source code analysis: A road map. In: IEEE COMPUTER SOCIETY. *2007 Future of Software Engineering*. [S.l.], 2007. p. 104–119.

CHIDAMBER, S. R.; KEMERER, C. F. A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, IEEE, v. 20, n. 6, p. 476–493, 1994.

CRUZ, D. d.; HENRIQUES, P. R.; PINTO, J. S. Code analysis: Past and present. 2009.

DARCY, D. P. et al. The structural complexity of software an experimental test. *Software Engineering, IEEE Transactions on*, IEEE, v. 31, n. 11, p. 982–995, 2005.

HITZ, M.; MONTAZERI, B. *Measuring Coupling and Cohesion In Object-Oriented Systems*. [s.n.], 1995. Disponível em: ⟨http://www.isys.uni-klu.ac.at/PDF/1995-0043-MHBM.pdf⟩.

ISO, I. Iec25010: 2011 systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models. *International Organization for Standardization*, p. 34, 2011.

MCCABE, T. J. A complexity measure. *Software Engineering, IEEE Transactions on*, IEEE, n. 4, p. 308–320, 1976.

MEIRELLES, P. R. M. *Monitoramento de métricas de código-fonte em projetos de software livre*. Tese (Doutorado) — Universidade de São Paulo, São Paulo, Brazil, 2013.

NIST. *SAMATE - Source Code Security Analyzers*. 2016. [Online; acessado 20 Abril de 2016]. Disponível em: ⟨http://samate.nist.gov/index.php/Source_Code_Security_Analyzers.html⟩.

ROSENBERG, L. H.; HYATT, L. E. Software quality metrics for object-oriented environments. *Crosstalk journal*, v. 10, n. 4, 1997.

SHARBLE, R. C.; COHEN, S. S. The object-oriented brewery: a comparison of two object-oriented development methods. *ACM SIGSOFT Software Engineering Notes*, ACM, v. 18, n. 2, p. 60–73, 1993.

SHIH, T. K. et al. Decomposition of inheritance hierarchy dags for object-oriented software metrics. In: *Engineering of Computer-Based Systems, 1997. Proceedings., International Conference and Workshop on*. [S.l.: s.n.], 1997. p. 238–245.

TERCEIRO, A. S. de A. *Caracterização da Complexidade Estrutural em Sistemas de Software*. Tese (Doutorado) — Universidade Federal da Bahia, Salvador, 2012.