LPCC02 - Lab 2 - Checkpoint 1

Tiago Lucas Pereira Clementino

20 de Março de 2018

Contents

1 Contexto

Nosso objetivo é analisar e extrair informação de dados coletados por uma plataforma aberta de integração contínua chamada Travistorrent. O Travisttorrent é um serviço de integração contínua de projetos disponíveis no Github, um repositório de projetos colaborativos também aberto.

A integração contínua é um processo onde o desenvolvedor integra o código alterado e/ou criado ao projeto principal na mesma frequência com que as funcionalidades são introduzidas.

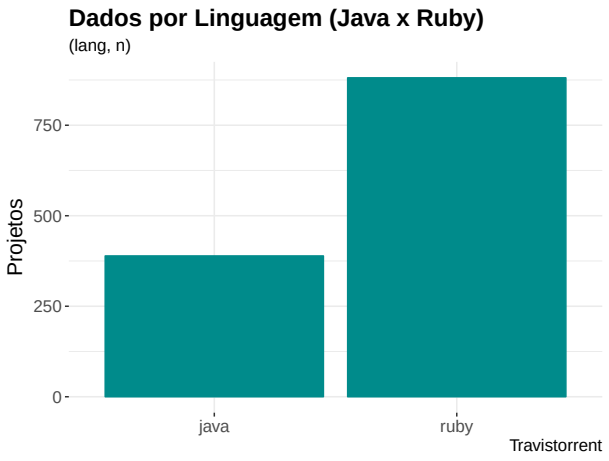
Nossa intenção é analisar os dados fazendo um paralelo entre as linguagens de programação Java e Ruby, levantando questões inerentes às características dos projetos de software que as utilizam. Diante disto, descartamos quaisquer dados referentes a outras linguagens.

2 Nossos Dados

Os dados são informações diversas, referentes a projetos disponíveis no Github e que, no momento da coleta, utilizaram o Travistorent nos últimos três meses, além de corresponder a certas especificações de filtragem. Estas informações descrevem operações inerentes ao andamento de projetos no Github e a procedimentos de integração contínua (testes, *commits*, PR, *builds* de integração, etc) ao longo de um certo período de tempo.

3 Distribuições dos dados entre Java e Ruby

O gráfico abaixo apresenta a distribuição dos dados entre Java e Ruby. É fácil perceber que a vantagem numérica de Ruby em relação à Java é grande, mas isto não deve interferir em nossas observações. Nossas análises farão um paralelo entre estas duas linguagens.



4 Objetivos

Precisamos avaliar um conjunto de variáveis em um exercício de visualização de correlações de dados através de marcas e canais. O objetivo de nossa investigação é entender os dados e as relações entre suas variáveis. Tendo isto em vista, é importante ser capaz de escolher as marcas e canais adequados para nossa análise.

A partir disto, devemos propor possíveis correlações alvo ligadas às relações entre variáveis que julguemos interessantes, para em seguida proceder a análise.

4.1 Regras

De acordo com o enunciado do Laboratório 2 checkpoint 02 da disciplina LPCC2, este documento deve conter gráficos plenamente legíveis (tanto o texto quanto os elementos) para um zoo de 100% em um documento pdf de duas colunas (os gráficos devem ter a largura de uma coluna) e seguir as boas práticas de visualização estudadas. Além de texto descrevendo a tarefa de visualização.

5 Variáveis

Nossa base de dados conta com variáveis bem intuitivas, cada uma delas descreve alguma característica dos projetos. Vide tabela no final deste documento

Na nossa análise buscaremos uma ou mais características da relação entre as variáveis sloc_end e sloc_med, explorando marcas e canais gráficos. Além de incluirmos eventualmente activity_period, team e lang para posicionar nossas análises no nosso conjunto de dados.

5.1 Variáveis de estudo

Selecionamos para este relatório duas variáveis que serão alvo de comparações, medições e visualizações, são elas: sloc_end e sloc_med, outras variáveis também podem ser usadas na comparação como lang ou team. Nosso objetivo é observar seu comportamento, confirmando ou refutando certas perspectivas de correlação que chamaremos de questões e são descritas abaixo.

6 Análise

A partir daqui analisaremos nossos dados com base em questões levantadas a partir de correlações entre variáveis. Neste estudo, o principal foco é a análise visual. Utilizaremos diversas variedades de gráficos estatísticos, canais de visualização e marcas.

Um canal é uma dimensão usada para converter dados de uma tabela em uma visualização gráfica. Um canal pode ser de alta ou baixa magnitude, representando uma eficácia na visualização de informação maior ou menor.

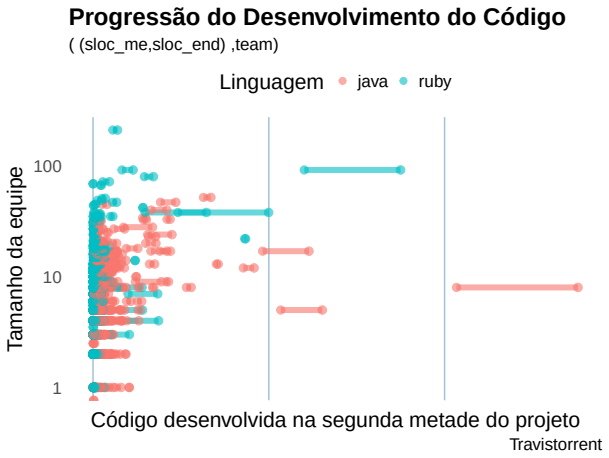
Nosso primeiro objetivo é entender como o volume de código se comporta ao longo do tempo, e faremos isto comparando as variáveis sloc_end e sloc_med, sempre fazendo um paralelo com variáveis significativas como lang e/ou team, priorizando canais mais eficazes para variáveis alvo de análise.

6.1 sloc_end e sloc_med

A principal relação entre estas duas variáveis é o tempo. sloc_med, como já foi mencionado, mede o volume de código de um projeto na metade de seu tempo de vida (até o fim da medição dos dados), já sloc_end mede o volume total de código no fim do período de medição. É de se esperar que sloc_med sejá menor que sloc_end, mas qual a proporção desta diferença?

Para começar a responder esta pergunta poderíamos pensar em usar um gráfico de dispersão (*scatter plot*). O gráfico de dispersão posiciona todos os elementos observados como pontos em um plano cartesiano, onde x corresponde a uma variável e y à outra.

Um bom tipo de gráfico para relacionar sloc_med e sloc_end é o *Dumbbell Plot*, que pode ser visto como uma variação do gráfico de dispersão. Nele o canal "posicionamento bidimensional" está representado na forma de pontos em um plano tal como o gráfico de dispersão. Porém, o eixo x apresenta duas variáveis e a progressão de uma até a outra, formando uma linha (o comprimento da linha é mais um canal). Abaixo está um *Dumbbell Plot* onde x descreve a variável sloc_med e a distância entre ela e sloc_end. Incluiremos também as variáveis lang e team nos canais cor dos pontos e dimensão vertical para posicionar melhor nosso gráfico nos nossos dados.

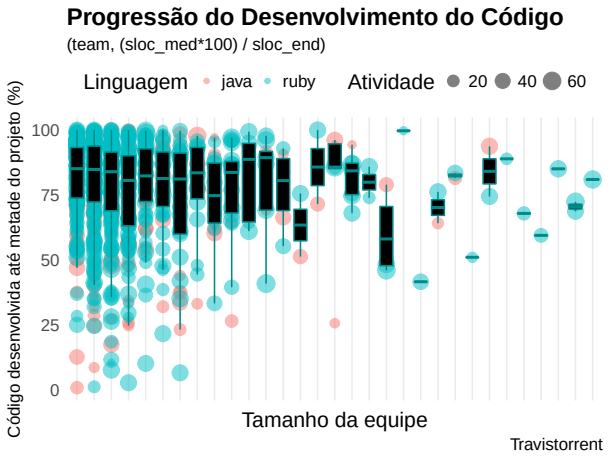


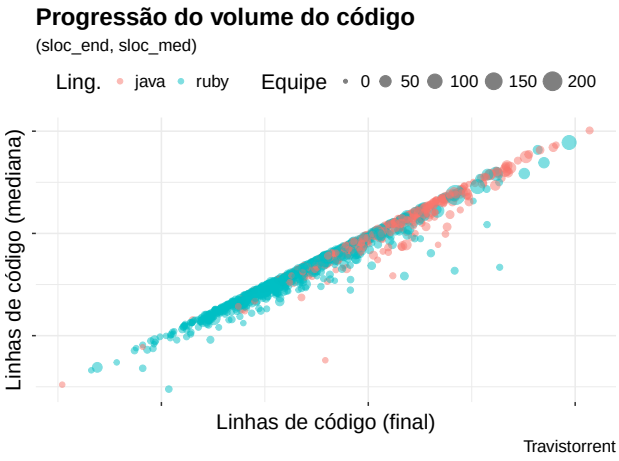
Observando os pontos e traços neste gráfico podemos ter a ideia de que talvez a maior parte do desenvolvimento ocorra entre o início e a primeira metade do projeto. Isto com base na diferença entre o comprimento dos traços que descrevem a distância entre sloc_med e sloc_end, e a distância entre sloc_med e a origem de x. Porém, é uma conclusão muito inicial, pois há muitos pontos aglomerados de difícil visualização, além de alguns pontos com grandes distâncias entre sloc_med e sloc_end.

Uma nova evidência para esta afirmação poderia ser obtida observando não as duas variáveis separadas, mas uma nova variável composta pela proporção entre elas. A fórmula

$$\frac{sloc\_med * 100}{sloc\_end}$$

nos traz este valor. Novamente, para posicionar melhor nossa visualização nos nossos dados, incluiremos as seguintes três variáveis no gráfico; team, activity_period e lang nos canais dimensão horizontal, área do ponto e cor, respectivamente. Um *boxplot* para cada um de cem *breaks* (quando diferentes de vazio) ajuda a visualizar a relação.





No gráfico acima é possível perceber que a maior parte dos pontos está acima de 60%, o que nos leva a concluir que geralmente mais da metade do código é escrito no início do projeto. Além disto, é frequênte, de acordo com o gráfico, projetos com 80% ou 90% do desenvolvimento ocorrido na primeira metade de sua vida. Observando os *bosplots* pode-se perceber apenas um *box* abaixo de 50%, e com apenas um projeto.

Uma visualização mais clara talvez possa vir com um gráfico de dispersão (*scatter plot*) simples, que descreve a posição de cada projeto em um plano cartesiano onde x é o sloc_end e y é sloc_med. Novamente, incluiremos também as variáveis lang e team nos canais cor e área dos pontos para posicionar melhor nosso gráfico nos nossos dados.

Veja a eficácia do canal "posicionamento no plano", apenas com base no gráfico acima já podemos perceber claramente dois atributos desta relação. O primeiro aponta que de fato, como já poderíamos supor, sloc_end é maior que sloc_med aparentemente em todos os casos. Perceba que não há nenhum ponto acima da diagonal principal do nosso gráfico. Neste momento, não podemos garantir que esta afirmação é verdade para todos os casos, pois os canais de visualização geralmente proveem apenas uma direção para a análise. Em buscar de uma resposta categórica, precisamos de um modelo. Veja abaixo.

```
## # A tibble: 1 x 1
##   `sloc_med > sloc_end`
##                   <int>
## 1                     0
```

Agora podemos afirmar categoricamente que nenhum projeto em nossa base encolheu (refatoramento) entre a metade e o fim de sua vida. O segundo atributo da relação entre estas duas variáveis diz respeito à linearidade. É possível perceber que a figura composta pelos pontos no gráfico forma uma linha muito nítida à quase que

exatos 45° de inclinação, bem em cima da diagonal principal do gráfico. Isto sugere que os valores de sloc_med e sloc_end são muito parecidos, o que nos leva a crer que a maior parte do desenvolvimento ocorra na primeira metade da vida dos projetos na nossa base de dados, como os dois gráficos anteriores já demonstravam.

Novamente os canais de visualização nos apontam a direção da análise, mas para responder se a maior parte do desenvolvimento realmente ocorre no início do projeto devemos recorrer aos modelos. Através de um coeficiente de correlação linear entre sloc_med e sloc_end podemos ter a confirmação que buscamos. Nesta correlação sempre retornará um valor entre -1 e 1, onde -1 representa uma perfeita correlação decrescente, 0 a ausência de correlação e 1 representa uma correlação crescente forte. Aqui calculamos a correção entre estas variáveis de três métodos diferentes.

```
## # A tibble: 1 x 3
##   pearson spearman kendall
##     <dbl>    <dbl>   <dbl>
## 1   0.980    0.981   0.906
```

A tabela apresenta os três valores referentes as três formas de calcular a correlação entre as variáveis. Perceba que todos apresentam uma correlação superior a 0.9, o que indica uma forte correlação. Esta evidência responde nossa atual dúvida. De fato, como os gráficos mostraram, a maior parte do desenvolvimento ocorre na primeira metade do projeto.

7 Conclusão

Podemos concluir com base nos gráficos e resultados matemáticos, que nenhum projeto teve seu código refatorado a ponto de encolher da metade do tempo de vida do projeto até o fim e que a maior parte do desenvolvimento ocorre, em média, na primeira metade do tempo de vida do projeto.

Como ameaça à validade desta conclusão, podemos mencionar que a variável sloc_end não representa, de fato, o fim da vida do projeto, mas o fim da medição. Alguns projetos podem ter entrado em produção (quando passar a sofrer muito menos alterações) antes mesmo do momento em que sloc_med foi registrado, outros não terem atingido a maturidade mesmo ao final do tempo total.

Anexos

Nome	Descrição	Tipo
gh_project_name	nome do projeto	Categórica
team	total de desenvolvedores que participaram do projeto até sua última medição	Numérica
lang	linguagem de programação predominante	Categórica
sloc_end	total de linhas de código na última medição do projeto	Numérica
sloc_med	total de linhas de código no meio do tempo de atividade estimado do projeto	Numérica
activity_period	tempo de atividade estimado do projeto	Numérica
num_commits	total de submissões de alteração durante todo o tempo de atividade do projeto	Numérica
commits_per_month	total de submissões por mês	Numérica
tests_per_kloc	casos de teste por total de linhas de código	Numérica
total_builds	total de integrações	Numérica
build_success_prop	proporção de integrações bem-sucedidas	Numérica
builds_per_month	total médio de integrações por mês	Numérica
tests_added_per_build	total médio de testes adicionados por integração	Numérica
tests_successful	total de testes bem-sucedidos	Numérica
test_density	densidade de testes	Numérica
test_size_avg	tamanho médio dos casos de testes	Numérica