LPCC02 - Lab 2 - Checkpoint 1

**Tiago Lucas Pereira Clementino**

**15 de abril de 2018**

### Bibliotecas utilizadas

library(tidyverse)  
library(here)  
library(knitr)  
library(gridExtra)  
library(ggExtra)  
library(gapminder)  
library(ggalt)  
theme\_set(theme\_bw())  
options(warn=-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.

# Nossos Dados

Os dados são informações diversas referentes a projetos disponíveis no Github e que, no momento da coleta, utilizaram o Travistorent nos ultimos 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.

### Carregando os dados

projetos = read\_csv(here::here("dados/projetos.csv"))

## Parsed with column specification:  
## cols(  
## gh\_project\_name = col\_character(),  
## team = col\_double(),  
## lang = col\_character(),  
## sloc\_end = col\_integer(),  
## sloc\_med = col\_double(),  
## activity\_period = col\_integer(),  
## num\_commits = col\_integer(),  
## commits\_per\_month = col\_double(),  
## tests\_per\_kloc = col\_double(),  
## total\_builds = col\_integer(),  
## build\_success\_prop = col\_double(),  
## builds\_per\_month = col\_double(),  
## tests\_added\_per\_build = col\_double(),  
## tests\_successful = col\_double(),  
## test\_density = col\_double(),  
## test\_size\_avg = col\_double()  
## )

### Filtrando os dados (eliminando linguagens diferentes de Java e Ruby identificadas)

projetos = projetos %>%   
 filter(lang != "javascript")

# Variáveis

Nossa base de dados conta com variáveis bem intuitivas.

projetos %>% names()

## [1] "gh\_project\_name" "team"   
## [3] "lang" "sloc\_end"   
## [5] "sloc\_med" "activity\_period"   
## [7] "num\_commits" "commits\_per\_month"   
## [9] "tests\_per\_kloc" "total\_builds"   
## [11] "build\_success\_prop" "builds\_per\_month"   
## [13] "tests\_added\_per\_build" "tests\_successful"   
## [15] "test\_density" "test\_size\_avg"

Cada uma delas descreve alguma característica dos projetos.

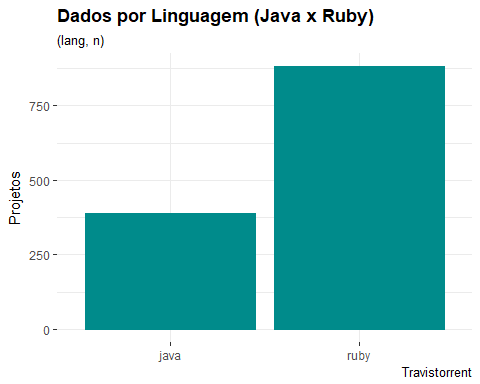
|  |  |  |
| --- | --- | --- |
| Nome | Descrição | Tipo |
| **gh\_project\_name** | nome do projeto | Categórica |
| **team** | total de desenvolvedores que participaram do projeto até sua última medição | Numérica |
| **lang** | linguagem de programação predominante | Categórica |
| **sloc\_end** | 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 |

Na nossa análise buscaremos uma ou mais características da relação entre as variáveis **sloc\_end** e **sloc\_med**, e **tests\_per\_kloc** e **test\_size\_avg** explorando marcas e canais gráficos. Além de incluirmos **activity\_period**, **team** e **lang** eventualmente para posicionar nossas análises no nosso conjunto de dados.

# 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.

projetos %>%   
 group\_by(lang) %>%   
 summarise(projetos\_ = n()) %>%   
 ggplot(aes(x = lang, y = projetos\_)) +   
 geom\_col(fill = "darkcyan", color = "darkcyan") +  
 labs(x=NULL,   
 y='Projetos',   
 title="Dados por Linguagem (Java x Ruby)",   
 subtitle="(lang, n)",   
 caption="Travistorrent") +  
 theme(plot.title = element\_text(face="bold"),panel.border=element\_blank())



# Objetivos

Precisamos avaliar um conjunto de variáveis em um exercício de visualização de correlações em 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 questões alvo ligadas às relações entre variáveis que julguemos interessantes, para em seguida proceder a análise.

### Regras

De acordo com o enunciado do Laboratório 2 da disciplina LPCC2, devemos elaborar pelo menos seis gráficos envolvendo quatro variáveis diferentes. Cada um com ao menos duas marcas distintas.

# Variáveis de estudo.

Selecionamos para este relatório quatro variáveis que serão alvo de comparações, medições e visualizações, são elas: **sloc\_end**, **sloc\_med**, **tests\_per\_kloc** e **test\_size\_avg**, outras variáveis também podem ser usadas na comparação, como **lang**, **activity\_period** ou **team**. Nosso objetivo é observar seu comportamento, confirmando ou refutando certas prespectivas de correlação que chamaremos de questões e são descritas abaixo.

# 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** em relação à outras variáveis. Em seguida tentaremos entender a relação entre **tests\_per\_kloc** e **test\_size\_avg**, sempre fazendo um paralelo com variáveis significativas como **lang** e/ou **team**, priorizando canais mais eficazes para variáveis alvo de análise.

### **sloc\_end** e **sloc\_med**

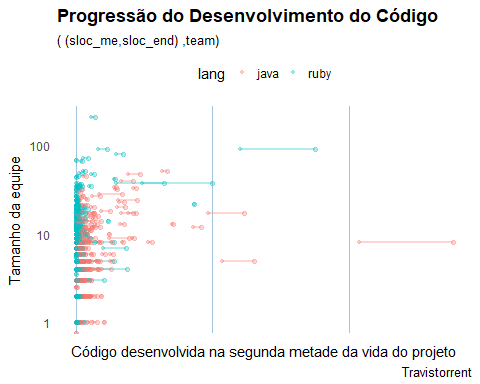
A principal relação entre estas duas variáveis é o tempo. **sloc\_med**, como já foi mensionado, mede o volume de código de um projeto na metade de seu tempo de vida (até o fim da medição dos dados), já **sloc\_end** mede o volume total de código no fim do período de medição. É de se esperar que **sloc\_med**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

#### Gráfico de dispersão

Para começar a responder esta pergunta poderiamos pensar em usar um gráfico de disperção (scattler plot). O gráfico de disperção 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 disperção. Nele o canal “posicionamento bidimensional”" está representado na forma de pontos em um plano tal como o gráfico de dispersão. Porém, o exito 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 e a distância entre ela e **sloc\_end**. Incluiremos também as variáveis **lang** e **team** nos canais cor dos pontos e dimenssão vertical para posicionar melhor nosso gráfico nos nossos dados.

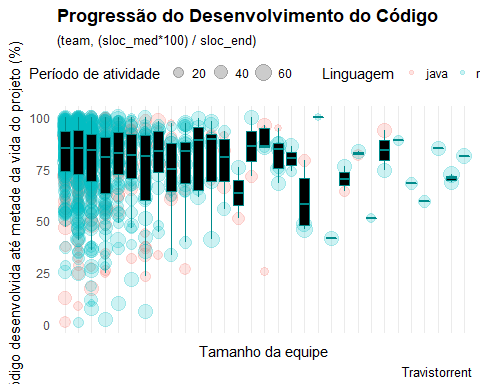
gg <- projetos %>%  
 ggplot(aes(x=sloc\_med, xend=sloc\_end, y=team, group=team, color=lang)) +   
 geom\_dumbbell(  
 #color="#a3c4dc",   
 size=1.0,   
 alpha=.3,  
 #colour\_xend = "#0e668b"  
 ) +   
 #scale\_x\_continuous(label=percent) +   
   
 scale\_y\_log10()+   
 labs(x='Código desenvolvida na segunda metade da vida do projeto',   
 y='Tamanho da equipe',   
 title="Progressão do Desenvolvimento do Código",   
 subtitle="( (sloc\_me,sloc\_end) ,team)",   
 caption="Travistorrent") +  
 theme(plot.title = element\_text(#hjust=0.5,   
 face="bold"),  
 plot.background=element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_line(color="#a3c4dc"),  
 axis.ticks=element\_blank(),  
 legend.position="top",  
 axis.text.x = element\_blank(),  
 panel.border=element\_blank())  
plot(gg)



Observando os pontos e traços neste gráfico podemos ter à idéia de que talvez a maior parte do desenvolvimento ocorra entre o início e a primeira metade do projeto. Isto com base na diferença entre os 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 “(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 dimenssão horizontal, área do ponto e cor, respectivamente. Um *boxplot* para cada um de cem *breaks* (quando diferentes de vazio) ajuda a visualizar a relação.

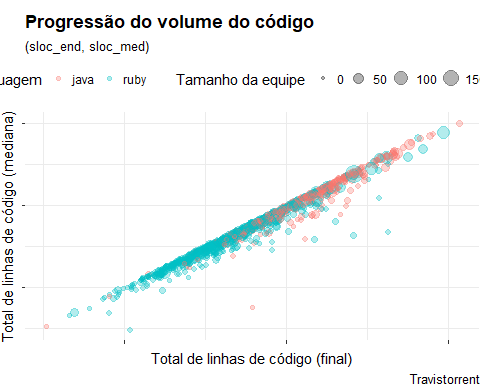
projetos %>%  
 ggplot(aes(x=cut(team, breaks=100), y=(sloc\_med\*100)/sloc\_end)) +  
 #geom\_violin()   
 geom\_count(aes(color=lang,size=activity\_period),alpha = .2)+   
 geom\_boxplot(outlier.alpha = .0, fill='#000000', color = "darkcyan") +  
 labs(x='Tamanho da equipe',   
 y='Código desenvolvida até metade da vida do projeto (%)',   
 color='Linguagem',   
 size='Período de atividade',   
 title="Progressão do Desenvolvimento do Código",   
 subtitle="(team, (sloc\_med\*100) / sloc\_end)",   
 caption="Travistorrent") +  
 theme(plot.title = element\_text(#hjust=0.5,   
 face="bold"),  
 plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_line(),  
 axis.ticks=element\_blank(),  
 axis.text.x = element\_blank(),  
 legend.position="top",  
 panel.border=element\_blank())



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 frequência, 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 talves possa vir com um gráfico de disperção (scattler 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.

projetos %>%   
 ggplot(aes(x= sloc\_end, y= sloc\_med , color = lang, size=team),na.rm=TRUE) +   
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 geom\_point(alpha = .3) +   
 labs(x='Total de linhas de código (final)',   
 y='Total de linhas de código (mediana)',   
 title='Progressão do volume do código',   
 color='Linguagem',   
 size='Tamanho da equipe',   
 subtitle='(sloc\_end, sloc\_med)',   
 caption="Travistorrent") +  
 theme\_bw() +  
 theme(plot.title = element\_text(face="bold"),  
 panel.border=element\_blank(),  
 legend.position="top",  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 plot.margin=unit(c(10,0,0,0),"points"))



Veja a eficácia do canal “posicionalmento 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 tudos os casos. Perceba que não há nenhum ponto acima da diagonal principal do noso gráfico. Neste momento, não podemos grarantir que esta afirmação é verdade para todos os casos, pois os canais de visualização geralmente proveêm apenas uma direção para a análise. Em buscar de uma resposta categórica, precisamos de um modelo. O que não é difícil de se conseguir. Veja abaixo.

projetos %>%   
 filter(sloc\_med > sloc\_end) %>%  
 summarise(`Total de projetos com mais código na metade de seu tempo de vida que o final`=n())

## # A tibble: 1 x 1  
## `Total de projetos com mais código na metade de seu tempo de vida que o~  
## <int>  
## 1 0

Agora podemos afirmar categoricamente que nenhum projeto em nossa base encolheu 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 encima 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 descrescente, 0 a ausência de correlação e 1 representa uma correlação crescente forte. Aqui calculamos a correção entre estas variáveis de três métodos diferentes.

projetos %>%  
 summarise(pearson = cor(sloc\_end, sloc\_med, method = "pearson"),  
 spearman = cor(sloc\_end, sloc\_med, method = "spearman"),  
 kendall = cor(sloc\_end, sloc\_med, method = "kendall"))

## # 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.

#### 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 projeto 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.

### **tests\_per\_kloc** e **test\_size\_avg**

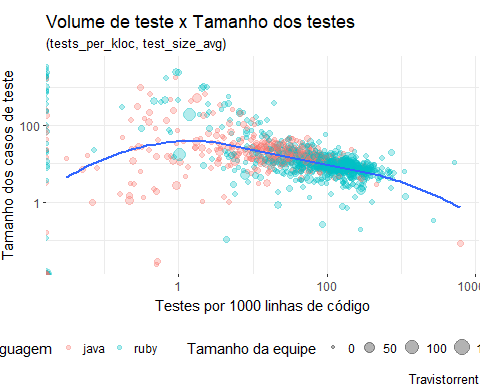
Um relação presente entre estas duas variáveis é que ambas aumentam o volume total do código destinado a testes. **tests\_per\_kloc** mede o volume de testes por mil linhas de código, já **test\_size\_avg** mede o tamanho médio de cada caso de teste em um projeto. Casos de teste mais complexos demandam mais código (geralmente), mas a maioria dos casos são descritos em poucas linhas. Por isto é de se esperar que o tamanho médio destes casos não seja muito grande. Porém, como veremos adiante, não é o que acontece na base. Muitos projetos tem, em média, casos de teste demasiadamente grande. Um caso de teste grande pode representar um problema complexo ou muitos casos de teste em um só. Com isto temos a pergunta: Casos de teste muito grandes levam a menos casos de testes?

#### Gráfico de dispersão

O posicionamento no espaço bidimensional é o mais poderoso canal de magnitude em gráficos estatísticos. Por isso o gráfico de dispersão, quando aplicável, será sempre uma boa opção para iniciar uma análise. Este possibilita não apenas a visualização dos valores numéricos em escala, mas também a marcação da identidade de variáveis categóricas.

Em busca da resposta para a pergunta proposta partiremos de um gráfico de dispersão comparativo, onde x é **tests\_per\_kloc** e y é **test\_size\_avg**. Mais uma vez, incluiremos também as variáveis **lang** e **team** nos canais cor e área dos pontos para posicionar melhor nosso gráfico nos nossos dados.

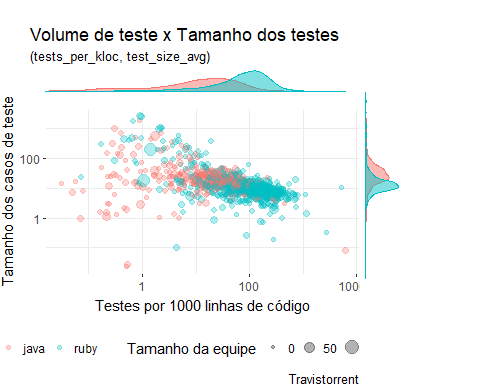
projetos %>%   
 ggplot(aes(x= tests\_per\_kloc, y= test\_size\_avg)) +   
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 geom\_point(aes(x= tests\_per\_kloc, y= test\_size\_avg, size=team, color=lang), alpha = .3) +  
 stat\_smooth(method='loess', se=F)+  
 labs(x='Testes por 1000 linhas de código',   
 y='Tamanho dos casos de teste',   
 caption="Travistorrent",  
 size="Tamanho da equipe",  
 color="Linguagem",  
 title="Volume de teste x Tamanho dos testes",  
 subtitle="(tests\_per\_kloc, test\_size\_avg)") +  
 theme(panel.border=element\_blank(),  
 legend.position="bottom",  
 plot.margin=unit(c(10,0,0,0),"points"))



É possível observar uma certa correlação linear entre as duas variáveis, o que sugere uma conexão (ainda que pequena). A curva que descreve a distribuição dos pontos torna isto mais nítido. Novamente, por mais forte que canais de posicionamento sejam, somente o calculo desta correlação pode responder à esta pergunta definitivamente. Por enquanto nosso interesse é apenas na observação dos dados.

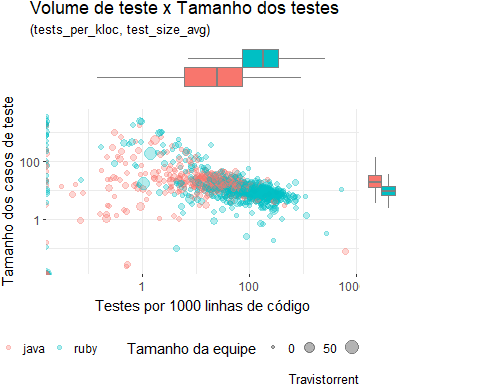
Os canais de posicionamento não encontram-se restritos apenas ao uso convencional nos eixos x e y. Como mostramos no *Dumbbell Plot*, é possível ir além. Partiremos para esplorar a visualização de gráficos justapostos, posicionados em escala de modo a possibilitar a comparação da distribuição dos valures nos dados. Veja o mesmo gráfico exibido acima, mas agora justaposto a gráficos de densidade acumulada alinhados de modo a explorarmos o fator posicionamento.

p1 <- projetos %>%   
 filter(test\_size\_avg > 0.0, tests\_per\_kloc > 0.0) %>%   
 ggplot(aes(x= tests\_per\_kloc, y= test\_size\_avg, size=team, color=lang)) +   
 geom\_point(alpha = .3) +  
 #scale\_x\_continuous(expand=c(0.02,0)) +  
 #scale\_y\_continuous(expand=c(0.02,0)) +  
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 labs(x='Testes por 1000 linhas de código',   
 y='Tamanho dos casos de teste',   
 title=NULL,   
 subtitle=NULL,   
 size="Tamanho da equipe",  
 color="Linguagem",  
 caption="Travistorrent") +  
 theme\_bw() +  
 theme(panel.border=element\_blank(),  
 legend.position="bottom",  
 plot.margin=unit(c(10,0,0,0),"points"))  
  
  
  
p2 <- projetos %>% ggplot(aes(x=tests\_per\_kloc,colour=lang,fill=lang)) +   
 geom\_density(alpha=0.5) +   
 #scale\_x\_continuous(breaks=NULL,expand=c(0.02,0)) +  
 #scale\_y\_continuous(breaks=NULL,expand=c(0.02,0)) +  
 scale\_x\_log10()+  
 labs(x=NULL,   
 y=NULL,  
 title="Volume de teste x Tamanho dos testes",  
 subtitle="(tests\_per\_kloc, test\_size\_avg)") +  
 theme\_bw() +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(20,0,0,20),"points"))  
  
p3 <- projetos %>% ggplot(aes(x=test\_size\_avg,colour=lang,fill=lang)) +   
 geom\_density(alpha=0.5) +   
 #scale\_x\_continuous(labels = NULL,breaks=NULL,expand=c(0.02,0)) +  
 #scale\_y\_continuous(labels = NULL,breaks=NULL,expand=c(0.02,0)) +  
 scale\_x\_log10()+  
 coord\_flip() +   
 labs(x=NULL,   
 y=NULL) +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(-10,60,70,0),"points"))  
  
grid.arrange(arrangeGrob(p2,ncol=2,widths=c(3,1)),  
 arrangeGrob(p1,p3,ncol=2,widths=c(3,1)),  
 heights=c(1,3))



Observe que ainda não adicionamos nenhuma nova variável ao gráfico, apeanas a densidade acumulada de cada cada eixo. Isto nos deu mais evidências de que ambas acumulam-se em valores médios, o sugere um baixo desvio padrão. Uma forma de percebermos isto com mais segurança por meio de visualizações é utilizando *bloxplots* ao invés de distribuição acumulada.

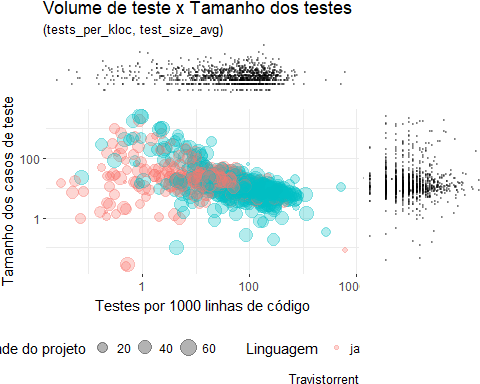
p7 <- projetos %>% ggplot(aes(x= tests\_per\_kloc, y= test\_size\_avg, size=team, color=lang)) +   
 geom\_point(alpha = .3) +  
 #scale\_x\_continuous(expand=c(0.02,0)) +  
 #scale\_y\_continuous(expand=c(0.02,0)) +  
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 labs(x='Testes por 1000 linhas de código',   
 y='Tamanho dos casos de teste',   
 title=NULL,   
 subtitle=NULL,   
 size="Tamanho da equipe",  
 color="Linguagem",  
 caption="Travistorrent") +  
 theme\_bw() +  
 theme(panel.border=element\_blank(),  
 legend.position="bottom",  
 plot.margin=unit(c(10,0,0,0),"points"))  
  
  
p8 <- projetos %>%  
 ggplot(aes(y = tests\_per\_kloc,x='hjhj',fill=lang)) +  
 scale\_y\_log10()+  
 coord\_flip() +   
 geom\_boxplot(outlier.alpha = .0, color = "#808080") +  
 #geom\_histogram(bins = 40,colour='#ffffff') +  
 labs(x=NULL,   
 y=NULL,  
 title="Volume de teste x Tamanho dos testes",  
 subtitle="(tests\_per\_kloc, test\_size\_avg)") +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.text.y = element\_blank(),  
 axis.text.x = element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(0,-20,0,20),"points"))  
  
p9 <- projetos %>%  
 ggplot(aes(x = 'um label',y = test\_size\_avg,fill=lang)) +  
 scale\_y\_log10()+  
 #coord\_flip() +   
 geom\_boxplot(outlier.alpha = .0, color = "#808080") +  
 #geom\_histogram(bins = 40,colour='#ffffff') +  
 labs(x=NULL,   
 y=NULL) +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(0,60,70,0),"points"))  
  
grid.arrange(arrangeGrob(p8,ncol=2,widths=c(3,1)),  
 arrangeGrob(p7,p9,ncol=2,widths=c(3,1)),  
 heights=c(1,3))



Perceba que os *boxs* são curtos em relação a amplitude de valores das variáveis, o que aponta para um acumulo de valores em torno da mediana e, por sua vés, um provável desvio padrão baixo para ambas as variáveis. Mas ainda não sabemos se testes grandes levam a poucos testes.

Podemos tentar agrupar o gráfico de dispersão destas duas variáveis como suas disperções em relação a outras variáveis. Usaremos agora o canal de posicionamento espacial para comparar estas variáveis com a variável **team** em um canal mais eficazes que a o tamanho das marcas (na área das marcas exibiremos **activity\_period**, apenas para ampliar a visualização dos dados no nosso gráfico). Observe abaixo.

p7 <- projetos %>%   
 filter(test\_size\_avg > 0.0, tests\_per\_kloc > 0.0) %>%   
 ggplot(aes(x= tests\_per\_kloc, y= test\_size\_avg, size=activity\_period, color=lang)) +   
 geom\_point(alpha = .3) +  
 #scale\_x\_continuous(expand=c(0.02,0)) +  
 #scale\_y\_continuous(expand=c(0.02,0)) +  
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 labs(x='Testes por 1000 linhas de código',   
 y='Tamanho dos casos de teste',   
 title=NULL,   
 subtitle=NULL,   
 size="Idade do projeto",  
 color="Linguagem",  
 caption="Travistorrent") +  
 theme\_bw() +  
 theme(panel.border=element\_blank(),  
 legend.position="bottom",  
 plot.margin=unit(c(10,0,0,0),"points"))  
  
  
p8 <- projetos %>%  
 filter(test\_size\_avg > 0.0, tests\_per\_kloc > 0.0) %>%   
 ggplot(aes(x=tests\_per\_kloc,y = team )) +  
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 geom\_point(alpha = .3, size=.0) +  
 #geom\_histogram(bins = 40,colour='#ffffff') +  
 labs(x=NULL,   
 y=NULL,  
 title="Volume de teste x Tamanho dos testes",  
 subtitle="(tests\_per\_kloc, test\_size\_avg)") +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.ticks=element\_blank(),  
 axis.text.y = element\_blank(),  
 axis.text.x = element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(0,0,0,30),"points"))  
  
p9 <- projetos %>%  
 filter(test\_size\_avg > 0.0) %>%  
 ggplot(aes(x=test\_size\_avg, y = team)) +  
 scale\_x\_log10()+   
 scale\_y\_log10()+   
 geom\_point(alpha = .3, size=.0) +  
 coord\_flip() +   
 #geom\_histogram(bins = 40,colour='#ffffff') +  
 labs(x=NULL,   
 y=NULL) +  
 theme(plot.background=element\_rect(fill="#ffffff"),  
 legend.background = element\_rect(fill="#ffffff"),  
 panel.background=element\_rect(fill="#ffffff"),  
 panel.grid.minor=element\_blank(),  
 panel.grid.major.y=element\_blank(),  
 panel.grid.major.x=element\_blank(),  
 axis.text.x = element\_blank(),  
 axis.text.y = element\_blank(),  
 axis.ticks=element\_blank(),  
 panel.border=element\_blank(),  
 legend.position="none",  
 plot.margin=unit(c(10,-20,85,0),"points"))  
  
grid.arrange(arrangeGrob(p8,ncol=2,widths=c(3,1)),  
 arrangeGrob(p7,p9,ncol=2,widths=c(3,1)),  
 heights=c(1,3))



Veja que, mesmo através de um canal eficáz, não conseguimos ver correlação entre estas variáveis.

Para responder nosso questionamento nos restou calcular o coeficiente de correlação linear para as duas variáveis do nosso estudo. Novamente usaremos os mesmos três métodos distintos (Pearson, Spearman e Kendall).

projetos %>%  
 summarise(pearson = cor(tests\_per\_kloc, test\_size\_avg, method = "pearson"),  
 spearman = cor(tests\_per\_kloc, test\_size\_avg, method = "spearman"),  
 kendall = cor(tests\_per\_kloc, test\_size\_avg, method = "kendall"))

## # A tibble: 1 x 3  
## pearson spearman kendall  
## <dbl> <dbl> <dbl>  
## 1 -0.0770 -0.475 -0.378

#### Conclusão

Com base nos gráficos, sobretudo a dispersão dos pontos no plano, e no resultado dos coeficientes, podemos concluir que estas duas variáveis possuem um certa correlação linear inversa, porém fraca. O coeficiente de Pearson aponta para a ausência de correlação, porém Spearman e Kendall aprensentam um correlação muito fraca.

# Conclusão final

Podemos perceber a eficácia de diversos canais de visualização e extrair informação dos nossos dados com base neles. Verificamos que nossas observações foram mais frutíferas quando utilizamos gráficos de dispersão, o que argumenta a favor da grande importância deste instrumento.

A fraca, porém nítida, correlação linear entre **tests\_per\_kloc** e **test\_size\_avg** visível nos gráficos, mas não detectada pelos coeficientes de dispersão nos mostra que modelos matemáticos não são o bastante em certos casos. Já a indiscutível correlação entre **sloc\_end** e **sloc\_med** aponta que quando os gráficos cumprem o seu papel de apontar o caminho, é o modelo matemático que confirmar o resultado.