Metodologias Experimentais em Informática

Meta 1 - Exploratory Data Analisis

Joana Brás - 2021179983
Leandro Pais - 2017251509
Renato Ferreira - 2015228102

Novembro 2021

1 Introdução

No interesse da disciplina de Metodologias e Experimentação em Informática esta primeira meta do projeto, a análise exploratória de dados, é uma abordagem às cegas que consiste na aplicação de metodologias matemáticas que visa encontrar estruturas nos dados, extrair variáveis relevantes bem como detetar outliers e por consequência, testar pressupostos que se possam ter em relação aos dados. Este tipo de análise é principalmente baseada em técnicas gráficas, é também orientada aos dados e procura modelos que são sugeridos pelos próprios dados.

No entanto, para além da análise dos dados, esta meta engloba ainda a geração dos dados que serão depois alvo de análise. Assim sendo, ao longo deste documento vai ser retratado todo este processo desde a geração dos dados, passando pela EDA (Exploratory Data Analysis) dos mesmos até às conclusões alcançadas.

2 Análise das Variáveis

No contexto do projeto, e como foi sugerido nas aulas prática-laboratoriais, primeiramente foi realizada uma análise teórica, na qual, se analisou o enunciado e concluiu que existe uma clara distinção entre as variáveis independentes e as dependentes.

Assim sendo, como variáveis independentes foram identificadas, o **número de exames** e a **probabilidade de colisão**.

Já no que toca às variáveis dependentes, identifica-se, obviamente, o **tempo** mas também o **número de casos resolvidos**.

3 Testes preliminares

Numa segunda fase, fez-se uma experimentação inicial com os programas fornecidos. Deste primeiro contacto, concluiu-se que iria ser necessário a definição de intervalos para ambas as variáveis independentes.

3.1 Número de Exames

No que toca ao número de exames, foram realizados testes com várias probabilidades e verificou-se que abaixo de 10 exames não se consegue retirar dados experimentais relevantes. Isto acontece porque os programas correm de forma tão rápida que a máquina não tem sensibilidade para avaliar quanto tempo a execução demorou e, como tal, devolve sempre zero segundos.

Para mais do que cinquenta exames o oposto acontece. O problema torna-se tão complexo de resolver que o tempo de execução iria ser muito superior aquele disponível.

Por fim, importa definir um passo. Como em muitos casos não existe uma grande diferença com um incremento de apenas um exame, optou-se por um passo de 5 exames.

Assim, o intervalo final para o número de exames vai dos **15 aos 45 com um passo de 5.**

3.2 Probabilidade de Colisão

Seguindo a mesma lógica dos testes efetuados para o número de exames e tendo em conta as mesmas preocupações concluiu-se que o intervalo de estudo que melhor se adequa ao nosso caso para a probabilidade de colisão é entre **40% a 70% com um passo de 5%.**

4 Implementação

Depois de ter os valores para as variáveis independentes bem estabelecidos passou-se à implementação. Aqui, de forma a agilizar o trabalho, foram desenvolvidos dois scripts.

O primeiro script utiliza o intervalo definido para a probabilidade de colisão e para cada valor desta cria 5 testes para cada combinação probabilidade-exames, ou seja, para, por exemplo, uma probabilidade de 40% são gerados 5 testes distintos com 15 exames, de seguida outros 5 testes com 20 exames, ... até 45 exames. Posteriormente este processo é repetido para 45%, 50%, ..., até 70% de probabilidade de colisão. Isto é feito para garantir que nos testes há cobertura suficiente do problema. Importa ainda mencionar que o nome de cada ficheiro inclui um *id* que representa o grupo de testes em que está inserido, inclui também o número de exames que está naquele teste, a probabilidade de colisão e por fim a seed com que foi gerado.

O segundo script, tem como propósito testar os programas fornecidos contra os inputs gerados anteriormente. Para tal, leva como argumentos o nome do ficheiro com o código fonte (para compilar) e o max runtime. De seguida vai à pasta *data* onde o script anterior gerou todos os dados de teste e vai correr o executável uma vez por cada ficheiro que se encontre naquela diretoria. Por fim, o script vai gerar um ficheiro *out* cujo nome inclui a seed da execução, o max runtime e o programa que está a ser alvo de teste (*code1.c* ou *code2.c*) e que contém a seguinte informação: um header para o *read_table* que diz o que cada coluna representa. Depois para cada teste tem o número de exames, a probabilidade de colisão, a seed do teste, o número de slots da solução (este valor é -1 se não for encontrada) e o tempo de execução (que no limite é o max runtime).

5 Exploratory Data Analysis

5.1 Método Experimental

Feita a implementação passou-se à fase de geração de dados. Nesta fase, foi tomada a decisão de fazer todos os testes necessários numa só máquina para garantir alguma fiabilidade e coerência nos dados gerados. Assim sendo, utilizou-se o script de teste de código já mencionado (que pode ser visto no repositório do trabalho com link nas referências) que ficou a correr numa máquina unicamente dedicada ao efeito para que se pudesse evitar interferências no tempo de execução por parte de outras aplicações a correr no OS.

5.2 Condições Experimentais

De modo a complementar a secção anterior listam-se aqui as caraterísticas da máquina em questão.

5.2.1 Máquina

- *Hardware*
  - Processador: i7-7700HQ @ 2.80Ghz (8 CPUs)
  - Disco: SSD Micron_1100_MTFDDAV256TBN 256GB
  - RAM: 16GB DDR4
- *Benchmark*
  - UserBenchmark: Single Core 120 pontos; Multi Core: 490 pontos

5.3 Análise dos Resultados Experimentais

Obtidos todos os dados necessários chegou-se então à quarta fase. Esta consiste em nada mais nada menos do que a análise exploratória de dados em si. Assim, vai-se proceder à análise dos dados obtidos para as variáveis dependentes anteriormente identificadas.

5.3.1 Número de Casos Resolvidos

Foi desenvolvido um script R para contar o número de casos resolvidos em cada ficheiro de resultados utilizando estes resultados para fazer o barplot com os diferentes max runtimes.

O resultado é o gráfico que se pode ver abaixo, o número de casos resolvidos aumenta com o aumento do max runtime. Isto está de acordo com o esperado pois com mais tempo disponível para resolver cada caso de teste a probabilidade do mesmo ser resolvido é também maior.

Para além disto, apenas pelo análise visual deste barplot é fácil identificar que pelo menos no que toca ao número de casos resolvidos não há um código que se destaque em relação ao outro tendo resolvido exatamente o mesmo número de casos em cada iteração.

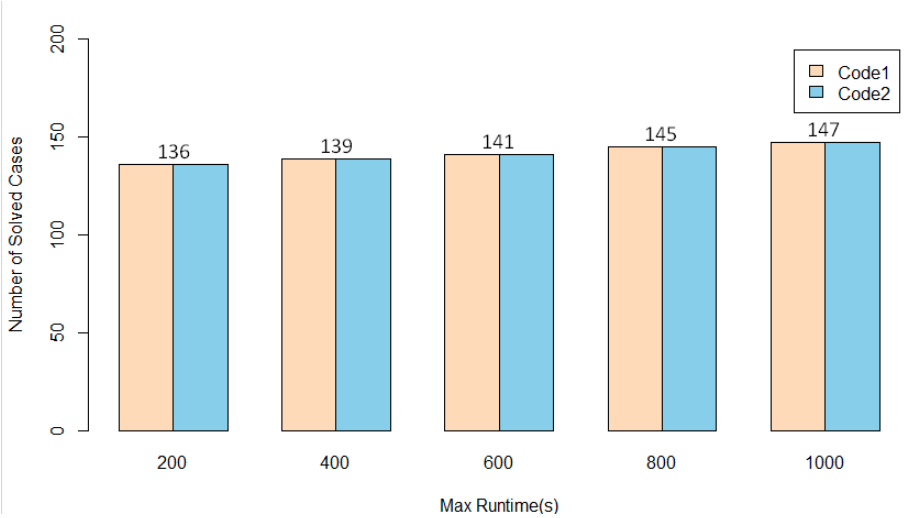


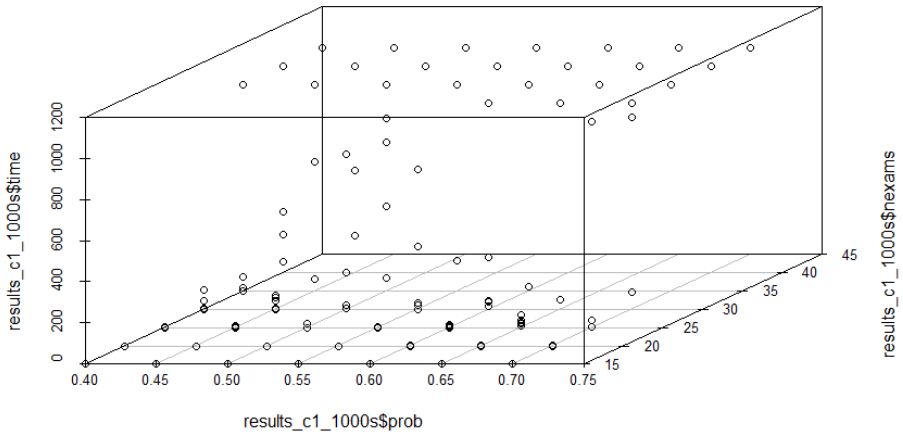
Figure 1: Barplot - Número de Casos Resolvidos: Programa 1 vs Programa 2

5.3.2 Tempo

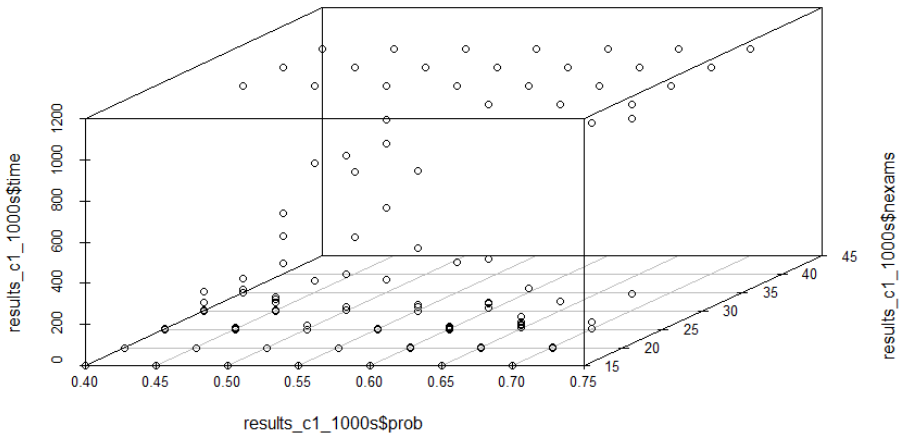
No que toca à análise do tempo, primeiramente importa esclarecer que para esta fase decidiu-se olhar apenas para os resultados com 1000 segundos de max runtime pois como é o que contém mais casos resolvidos sendo também o mais representativo.

Para a análise desenvolveu-se um script R que carrega os dados de todos os resultados para memória e faz o plot3d do tempo em função do número de exames e da probabilidade colisão.

O resultado são os gráficos que se podem ver abaixo. Da análise visual dos mesmos, pode-se especular que em ambos os códigos o fator tempo segue um crescimento que aparenta ser exponencial.



(a) Código 1



(b) Código 2

Figure 2: Plot 3D - Tempo em função do Número de Exames e da Probabilidade de Overlap

Para comprovar este possível modelo exponencial decidiu-se aplicar a regressão exponencial para ver que valores de R são obtidos. Para tal, acrescentou-se ao script o seguinte excerto.

```
#Adjust 0 values that will generate NaNs when logarithm is applied
results_c1_1000s_frame$time[results_c1_1000s_frame$time==0] <- 0.0001
results_c2_1000s_frame$time[results_c2_1000s_frame$time==0] <- 0.0001

#Exponential model
#Code1
lr_c1_3d_exp.out <- lm(log(results_c1_1000s_frame$time)
~ log(results_c1_1000s_frame$nexams) * log(results_c1_1000s_frame$prob))
summary(lr_c1_3d_exp.out)

#Code2
lr_c2_3d_exp.out <- lm(log(results_c2_1000s_frame$time)
~ log(results_c2_1000s_frame$nexams) * log(results_c2_1000s_frame$prob))
summary(lr_c2_3d_exp.out)
```

Primeiramente, ajustaram-se os valores de tempo que eram 0 para um valor mínimo superior a 0 para evitar que se gerassem NaNs na regrssão.

De seguida, aplicou-se a regressão exponencial com as três variáveis em questão para os dois códigos.

Assim, obtiveram-se os resultados que se podem observar nas seguintes tabelas. Para o **código 1** regista-se um r quadrado ajustado de aproximadamente **87%** o que nos permite afirmar com alguma confiança que de facto os dados seguem o modelo em estudo. Já para o **código 2** é observável um r quadrado ajustado de aproximadamente **85%** e que novamente permite afirmar com segurança que de facto os dados seguem este modelo.

Coeficientes: Código 1				
	Estimte Std.	Error	t value	Pr
Intercept	-36.019	4.453	-8.089	2.95e-14
log(nexams)	12.300	1.326	9.278	2e-16
log(prob)	22.113	6.932	3.190	0.00161
log(nexams):log(prob)	-4.907	2.064	-2.378	0.01821

Coeficientes: Código 2				
	Estimte Std.	Error	t value	Pr
Intercept	-37.579	4.626	-8.123	2.37e-14
log(nexams)	12.747	1.377	9.255	2e-16
log(prob)	18-152	7.202	2.521	0.0124
log(nexams):log(prob)	-3.776	2.144	-1.761	0.0795

6 Conclusão

No que toca às conclusões relevantes para o trabalho podemos dizer que após realizar testes com outros modelos de regressão (que não incluímos por falta de espaço mas que estão disponíveis aqui 1) e comparando os respetivos valores de R podemos concluir que o modelo que melhor explica os dados obtidos é de facto o exponencial (daí ter sido escolhido para constar no relatório).

Resta ainda mencionar que este projeto permitiu ao grupo aplicar conceitos teóricos sobre *Exploratory Data Analysis* apreendidos nas aulas teóricas e práticas. Para além disto, a oportunidade de trabalhar de perto com variáveis dependentes e independentes aprofundando melhor o que são e como podem ser identificadas. Ainda para mais, esta primeira meta foi também o primeiro contacto com a linguagem R o que também nos permitiu adquirir competências com a mesma.

References

[1] *Link para o repositório com scripts, testes e dados de input*
`https://github.com/lbpaisDev/MEI_Meta1_v2`