UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO DE BIOTECNOLOGIA DA UFRGS
PROGRAMA DE PÓS-GRADUAÇÃO EM BIOLOGIA CELULAR E MOLECULAR

Gene Expression Variation Analysis (GEVA): Um novo pacote do R para avaliar variações de expressão diferencial em múltiplas comparações

Itamar José Guimarães Nunes

Porto Alegre, Dezembro de 2020

Itamar José Guimarães Nunes

Gene Expression Variation Analysis (GEVA): Um novo pacote do R para avaliar variações de expressão diferencial em múltiplas comparações

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Biologia Celular e Molecular (PPGBCM) da UFRGS, como parte dos requisitos necessários para a obtenção do título de Mestre.

Orientador: Prof. Dr. Marcio Dorn
Coorientador: Dr. Bruno César Feltes

Porto Alegre
Dezembro de 2020

Itamar José Guimarães Nunes

Gene Expression Variation Analysis (GEVA): Um novo pacote do R para avaliar variações de expressão diferencial em múltiplas comparações / Itamar José Guimarães Nunes. – Porto Alegre, Dezembro de 2020

130 p.

Orientador: Prof. Dr. Marcio Dorn
Coorientador: Dr. Bruno César Feltes

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul, Centro de Biotecnologia da UFRGS, Programa de Pós-Graduação em Biologia Celular e Molecular, Porto Alegre (RS), Brasil, Dezembro de 2020.

1. Expressão gênica. 2. Transcritomas. 3. R. 4. Pacote. 5. Estatística.
I. Dorn, Márcio, orient. II. Feltes, Bruno César, coorient. III. Título

Itamar José Guimarães Nunes

Gene Expression Variation Analysis (GEVA): Um novo pacote do R para avaliar variações de expressão diferencial em múltiplas comparações

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Biologia Celular e Molecular (PPGBCM) da UFRGS, como parte dos requisitos necessários para a obtenção do título de Mestre.

Prof. Dr. Marcio Dorn
Orientador

Dr. Bruno César Feltes
Co-Orientador

Joice de Faria Poloni
Membro da Banca 1

Charley Christian Staats
Membro da Banca 2

Daniel Pens Gelain
Membro da Banca 3

Porto Alegre
Dezembro de 2020

Agradecimentos

Se há algo que realmente aprendi nesta jornada pelo mundo acadêmico é que não estamos sozinhos. E sim, essa frase é super-clichê e todo mundo já deve estar cheio dela, porém a razão de ainda ser válida é um pouco mais pragmática: nós simplesmente não podemos fazer nada sozinhos. O ser humano é um animal social, e é justamente essa maravilhosa habilidade de se unificar que o fez chegar tão longe, seja construindo uma pirâmide em cima de um deserto, seja competindo entre quem pinta o quadro mais surreal, seja fazendo a descoberta que mudará a vida de todos – claro, só para depois alguém contestá-lo com uma ideia ainda melhor. Para toda conquista existe um apoio, alguém que no mínimo torce para que ela se realize, e é neste momento que dedico meus agradecimentos a aqueles que ofereceram seu apoio para que eu pudesse caminhar sobre essa estrada pavimentada de desafios.

Primeiramente, agradeço aos meus pais, Giovana e Paulo, que desde sempre me deram muito amor e carinho, me aturaram quando vinha com "ideias de *girino*", e o mais importante: me incentivaram desde cedo a buscar conhecimento e a sempre seguir os meus sonhos. Essas palavras são poucas para agradecê-los por tudo que fizeram por mim até hoje, e acredito que uma versão completa renderia mais outra dissertação, então espero ter transmitido a mensagem. Da mesma forma, tenho muita gratidão por minha família, que sempre me deu suporte para que eu chegasse até aqui.

Também tenho muito a agradecer aos meus amigos (Davi Padilha Mesquita, Denise Pazetto Hauenstein, Murilo Zanini David, entre outros), que por vezes foram quase uma segunda família para mim e estiveram ao meu lado nos piores e melhores momentos. Jamais esquecerei de seu apoio por mim. Também devo um agradecimento especial ao Murilo, que contribuiu ativamente com seu conhecimento avançado de estatística durante a produção deste trabalho.

Da mesma forma, agradeço ao meus orientadores, Bruno e Marcio, por todo apoio acadêmico que me ofereceram, além de paciência para aturar todas as bobagens que vinham de mim. O Bruno ainda merece um agradecimento à parte: ele tem me aguentado por mais de sete anos como "orientador *de facto*", e já me conhece mais do que qualquer outro da faculdade. Tem me acompanhado desde o início – quando insistia para eu parar de usar imagens de caveiras pegando fogo para representar inflamação crônica – até o presente momento, me ajudando a crescer como pesquisador, me incentivando como programador e me inspirando como pessoa.

Por fim, também agradeço ao apoio financeiro fornecido pela FAPERGS durante toda a realização deste trabalho, e ao PPGBCM por oferecer essa oportunidade, o que permitiu que eu trabalhasse com um assunto que aprecio.

"Só porque você está certo, não significa que você está correto."
(Shirou Emiya)

Resumo

Transcritomas descrevem o perfil de expressão gênica de um organismo e oferecem uma quantidade considerável de informações sobre uma condição biológica, o que permite avaliar, por exemplo, todos os genes diferencialmente expressos no contexto de uma doença complexa. Além disso, dezenas de milhares de conjuntos de dados produzidos experimentalmente estão disponíveis publicamente, podendo ser analisados *in silico* com *softwares* especializados ou pela linguagem de programação R. No entanto, análises transcritômicas são efetuadas individualmente para cada condição experimental, onde cada comparação gera um resultado independente dos demais, e não há uma metodologia enquadrada a milhares de genes para comparar múltiplas condições biológicas entre si. Neste sentido, considerando a aplicabilidade da linguagem R para transcritomas, foi desenvolvido um pacote de métodos em R, denominado Gene Expression Variation Analysis (GEVA), para avaliar quais genes poderiam responder a diferentes condições experimentais. O pacote recebe múltiplos resultados de expressão diferencial como entrada e efetua uma sequência de operações estatísticas intermediárias, como sumarização ponderada, separação por quantis e clusterização, a fim de encontrar genes cuja expressão altera-se similarmente entre os experimentos. Essas operações levam todos os genes em conta para que se possa distinguir os transcritos relevantes daqueles que não demonstram diferença de expressão gênica. Além disso, se as condições experimentais forem divididas em grupos (denominados *fatores*), o módulo também utilizará testes ANOVA (Fisher e Levene) para identificar genes diferencialmente expressos em resposta a todos os fatores ou a um fator específico. Deste modo, três diferentes classificações para genes relevantes são definidas nos resultados finais: *similar*, *fator-dependente*, e *fator-específico*. Para validar esses resultados, após o desenvolvimento do GEVA, foram testados 28 conjuntos de dados transcritômicos utilizando 11 diferentes combinações de parâmetros disponíveis pelo pacote, incluindo uma variedade de métodos de clusterização, de quantis e de sumarização. As classificações finais foram avaliadas especialmente para condições de nocaute, onde a mudança na expressão dos genes modificados já é esperada. Neste sentido, embora as classificações finais fossem afetadas pela escolha de certos parâmetros, os resultados dos testes demonstraram coerência com os estudos experimentais referentes aos conjuntos testados com os parâmetros-padrão. Deste modo, conclui-se que o GEVA pode ser uma alternativa robusta para análises com múltiplas comparações, podendo ser utilizado para encontrar similaridades entre grupos de condições biológicas.

Palavras-chaves: Expressão gênica; Transcritomas; R; Pacote; Estatística.

Abstract

Transcriptomes represent an organism's gene expression profile and abound substantial information for a biological condition, by which it is possible to evaluate, for instance, an entire set of genes in the context of a complex disease. Furthermore, there are dozen of thousands of publicly available datasets from previous experiments which can be analyzed *in silico* using specialized software or the R programming language. However, transcriptomic analyses consider experimental conditions individually, giving one independent result per comparison, and there is no suitable methodology to compare within multiple biological conditions containing thousands of genes. In this sense, considering that R has been adopted for transcriptomic analyses, we developed a new R package named Gene Expression Variation Analysis (GEVA) to evaluate which genes would react in response to different experiments. This package gets multiple differential expression analysis results as input and performs an array of statistical steps such as weighted summarization, quantiles partition, and clustering, to find genes whose differential expression is similar among the experiments. These operations take all genes into account so that relevant transcripts are distinguished from those without differential expression. In addition, if the experimental conditions are divided into groups (*i.e.*, *factors*), this module will also perform ANOVA (Fisher's and Levene's) tests to identify differentially expressed genes in response to every factor or to a single factor. This way, the final results present three possible classifications for relevant genes: *similar*, *factor-dependent*, and *factor-specific*. To validate these results subsequently to the GEVA's development, 28 transcriptomic datasets were tested using 11 different combinations of the available parameters in this package, including several methods for clustering, quantiles and summarization. The validation regarding the final classifications was particularly performed using knockout studies, as these cases include modified genes whose differential expression is expected. In this sense, although some of the final classifications differed depending on the parameters' choice, the test results from the default parameters corroborated with the published experimental studies regarding the selected datasets. Thus, we conclude that GEVA can effectively find similarities between groups of biological conditions, and therefore could be a robust alternative for multiple comparison analyses.

Keywords: Gene Expression; Transcriptomics; R; Package; Statistics.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ANOVA	Análise de variância
DNA	Ácido desoxirribonucleico
FDR	*False-discovery rate*
GDE	Gene diferencialmente expresso
GEO	*Gene Expression Omnibus*
GEVA	*Gene Expression Variation Analysis*
GSE	*GEO Series* (identificador de série do GEO)
MAD	*Median Absolute Deviation* (desvio absoluto mediano)
SV	Relação sumarização-variação
UML	Unified Modeling Language
logFC	Logaritmo na base 2 de *fold-change* (expressão diferencial)
mRNA	RNA-mensageiro
ncRNA	RNA não-codificante

Lista de símbolos

s	Desvio-padrão (amostra)
σ	Desvio-padrão (população)
$\overline{x}$	Média de um conjunto
$\overline{x}_w$	Média ponderada de um conjunto
$\tilde{x}$	Mediana de um conjunto
$\tilde{x}_w$	Mediana ponderada de um conjunto
N	Número total de elementos de um conjunto
$\sum_{i=1}^{N}$	Somatório de um conjunto de N elementos partindo da posição 1
x_i	Valor na i-ésima posição de um conjunto
s^2	Variância (amostra)
σ^2	Variância (população)
$\lceil x \rceil$	Teto, menor número inteiro maior ou igual a x
$\lfloor x \rfloor$	Piso, maior número inteiro menor ou igual a x
p_i	Valor-p associado a um $logFC$ na coluna i
w_i	Valor de peso do elemento na posição i
w_T	Soma total dos valores de peso
$\in$	Pertence um conjunto
$1, ..., N$	Conjunto com valores de 1 a N

Sumário

Parte I

Embasamento

1 Introdução

1.1 Transcritomas e expressão diferencial

A chave para a sustentação de um organismo está nas constantes alterações em nível molecular que permitem sua adaptação ao ambiente, o que abrange uma série de vias moleculares de interações bioquímicas, transferências intercelulares de metabólitos, e regulações na expressão gênica. Destes, a expressão gênica se destaca por indicar uma forte relação com a atual condição biológica de uma célula, além de ser suficientemente estável para ser mensurada em comparação às permutações dinâmicas entre biomoléculas. Assim, ainda que um organismo compartilhe o mesmo genoma entre todas as suas células, dependendo da ativação e inativação de certos genes, diferentes tipos de células e tecidos podem ser formados no intuito de exercer seu papel específico na manutenção fisiológica. Por outro lado, desbalanços drásticos na maquinaria molecular que desestabilizam a expressão gênica também podem ocorrer em um indivíduo, causando desde uma mera alteração fenotípica sem impacto na sobrevivência ou na capacidade reprodutiva, até uma condição patológica potencialmente severa. Nestes casos mais nocivos, onde geralmente há a persistência da desregulação dos genes afetados, torna-se relevante entender qual ou quais genes mais contribuem para tal situação patológica, o que implicaria uma análise com todos os genes possíveis de um organismo afetado para avaliar o que houve de diferente em relação a um indivíduo saudável.

Neste sentido, uma das formas de identificar alterações na expressão gênica é através de sua quantificação em larga-escala por meio de **transcritomas**, que consistem em perfis de detecção de transcritos de RNA, sejam estes RNA menssageiro (mRNA) ou RNAs não codificantes (ncRNA), contidos em amostras sob determinadas condições, como tempo, tipo de tecido, estado fisiológico, entre outros exemplos. Transcritomas podem ser obtidos por meio de sondas moleculares que emitem luz ao se parearem a um transcrito, uma técnica conhecida como **microarranjo** (BUMGARNER, 2013); ou também por meio do sequenciamento dos próprios transcritos, uma técnica conhecida como **RNA-seq** (WANG; GERSTEIN; SNYDER, 2009). Ambas as técnicas levam em conta a totalidade de genes de um organismo, sendo o RNA-seq também capaz de quantificar isoformas e detectar mutações pontuais entre os genes analisados (Figura 1). No final, é gerada uma amostra transcritômica contendo um conjunto de valores de detecção para as moléculas utilizadas como referência aos genes (*e.g.*, sondas de um microarranjo ou *reads* de um RNA-seq).

Contudo, para extrair informações de transcritomas, amostras de diferentes condições devem ser quantificadas e comparadas entre si. Neste caso, precisamente, dois conjuntos de replicatas amostrais são considerados: um grupo de amostras sob uma con-

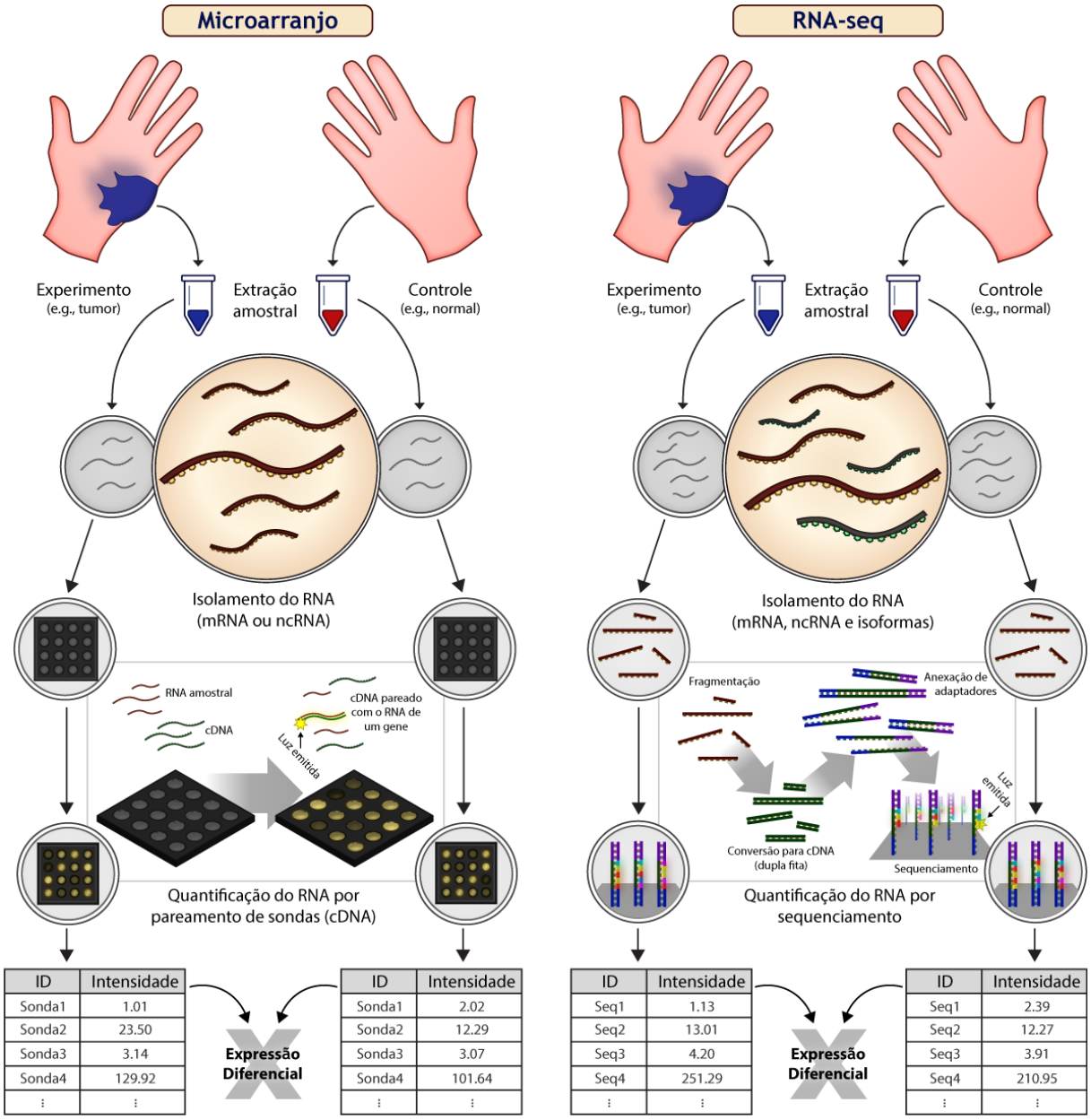

Figura 1 – Representação simplificada de experimentos com transcritomas utilizando microarranjos (coluna esquerda) e RNA-seq (coluna direita). Ambas as técnicas permitem a avaliação de todos os genes de um organismo, porém RNA-seq também abrange isoformas e mutações. Em microarranjo, poços microscópicos sobre um *Biochip* são preenchidos com sondas, moléculas de DNA complementar (cDNA) que emitem luz detectável ao se parearem com seu transcrito correspondente. Em RNA-seq, os transcritos são convertidos para cDNA de dupla-fita, amplificados e conectados a adaptadores específicos, seguido pela adição de bases que emitem luz detectável. Ambas as técnicas possuem diversas variantes entre protocolos experimentais, porém seguem a mesma lógica de quantificação de transcritos por intensidade de luz emitida. A iluminação detectada é, por fim, convertida em valores computáveis e comparada *in silico* com outras amostras, obtendo-se uma expressão diferencial.

dição observada (denominada *experimental*) e outro grupo na condição de referência (denominada *controle*). Comparando-se estes grupos amostrais entre si – grupo experimental contra grupo controle –, um valor de **expressão diferencial** é obtido para cada gene, o qual é numericamente representado por **logaritmo de *fold-change* (*logFC*)**. Este valor é obtido a partir da razão entre os valores de detecção sumarizados do grupo experimental (S_{exp}) e do grupo controle (S_{ctrl}), aplicando-se um logaritmo na base 2 para normalização de sua escala, como indicado na equação abaixo:

$$logFC = log_2 \left(\frac{S_{exp}}{S_{ctrl}} \right) \tag{1.1}$$

Para cada gene, o valor de $logFC$ determinará o quanto que sua expressão gênica foi alterada durante a condição experimental: um valor positivo indicará super-expressão, um valor negativo indicará sub-expressão, e um valor próximo de zero indicará que não houve diferença significante. Além disso, com a inclusão de replicatas amostrais, cada $logFC$ é acompanhado de um valor-p indicando sua significância estatística, o qual é calculado a partir dos desvios entre os valores de detecção e representa a probabilidade de o $logFC$ ser considerado incoerente à comparação. Um ajuste deste valor-p também costuma ser aplicado na finalidade de diminuir a taxa de erros tipo I ("falso-positivos"), utilizando-se métodos como o *false-discovery rate* (FDR).

Sendo assim, genes que apresentam um $logFC$ positivo ou negativo, bem como significância estatística (*i.e.* valor-p ajustado próximo de zero), serão atribuídos como **genes diferencialmente expressos (GDEs)** para a condição observada. O propósito das análises de expressão diferencial por transcritomas consiste, portanto, em identificar os GDEs que respondem ao conjunto de experimentos sendo estudados.

1.2 Múltiplas comparações

O termo "múltiplas comparações" pode ser utilizado no ramo da estatística para referir-se ao problema das múltiplas inferências, o qual sugere que há um aumento de erros de estimativa conforme mais testes estatísticos são efetuados simultaneamente (ABDI, 2007). Este problema também é referenciado na descrição do FDR para ajustar valores-p (BENJAMINI; HOCHBERG, 1995). Contudo, especificamente neste trabalho, o termo "múltiplas comparações" será utilizado no sentido literal para se referir ao ato de efetuar **duas ou mais comparações de expressão diferencial**, não havendo qualquer relação ao jargão estatístico mencionado.

Os procedimentos de uma análise de expressão diferencial para uma única comparação já encontram-se bem estabelecidos, existindo alternativas tanto por meio de programação quanto através de *softwares* especializados (mais detalhes na seção 2.1 –

Métodos computacionais para analisar transcritomas). As soluções por programação, especificamente, são majoritariamente efetuadas através da linguagem de programação R (IHAKA; GENTLEMAN, 1996) em virtude da vasta disponibilidade de módulos desenvolvidos nesta linguagem especialmente para aplicações de Bioinformática, em sua maior parte hospedados no repositório Bioconductor (GENTLEMAN et al., 2004). Ao mesmo tempo, há uma carência de opções quando deseja-se estender as análises de transcritomas a múltiplas comparações, particularmente na intenção de se relacionar a expressão diferencial entre duas ou mais condições biológicas. A demanda por esse tipo de informação pode surgir quando os processos biológicos analisados compartilham características em comum, principalmente no caso de doenças de uma mesma categoria – como seria o caso, por exemplo, de um estudo com diferentes tipos de tumor maligno que afetam o mesmo tecido, ou então de múltiplas doenças crônicas agravadas pelo mesmo processo inflamatório.

Considerando a imensa quantidade de transcritomas disponíveis publicamente (BARRETT et al., 2012), torna-se viável realizar um grande número de comparações com diferentes conjuntos de dados, porém ainda sendo necessário a aplicação de um método que lidasse com variações nos valores de expressão diferencial entre condições experimentais distintas, mas relacionáveis. Esse tipo de método permitiria uma melhor aproximação a problemáticas mais complexas, como doenças multifatoriais. Deste modo, o presente trabalho consistiu no desenvolvimento de um método de análise para múltiplas comparações, bem como sua aplicação computacional utilizando programação em R, por esta ser uma linguagem já devidamente estabelecida para expressão gênica.

2 Fundamentos

Para que se compreenda a problemática deste trabalho, há conceitos básicos que necessitam ser considerados primordialmente. Como será detalhado adiante, os métodos desenvolvidos neste trabalho envolvem a utilização de informações de **expressão gênica**, no contexto de uma **linguagem de programação específica**, para encontrar inferências de variabilidade por meio de **aplicações estatísticas** e **algoritmos de agrupamento**. Estes conceitos destacados serão apresentados, de forma resumida, ao longo das seções deste capítulo em prol de uma melhor familiarização com a metodologia aplicada e com a interpretação dos resultados obtidos.

2.1 Métodos computacionais para analisar transcritomas

Com o uso cada vez mais decorrente de transcritomas, editoras científicas como Elsevier e Nature passaram a exigir que todos os estudos utilizando essa técnica disponibilizassem os seus dados publicamente (ELSEVIER, 2020; DATA, 2020), de modo que estes pudessem ser reanalisados por outros autores. Atualmente, o maior banco de dados para este tipo de experimento é o **Gene Expression Omnibus (GEO)**, que contém milhares de estudos passíveis de reanálise (BARRETT et al., 2012).

Os dados de transcritoma disponíveis pelo GEO podem ser pesquisados no próprio *site*, onde serão identificados pelo título do estudo e um código de série com o prefixo GSE (sigla para "GEO Series"). Ao encontrar um GSE contendo um perfil de expressão para condições biológicas, é possível fazer o *download* dos dados para dar início a uma análise. Dados de microarranjo, encontrados no GEO pela categoria "Expression profiling by array", são o tipo mais abundante e com maior variedade de *softwares* de análise. Além do GEO2R (BARRETT et al., 2012), oferecido pelo próprio servidor do GEO para dados previamente curados, outros programas de microarranjo incluem Babelomics (ALONSO et al., 2015), Chipster (KALLIO et al., 2011), eUTOPIA (MARWAH et al., 2019), GEAP (NUNES, 2018), GenomeStudio (ILLUMINA, 2010), ShinyGEO (DUMAS; GARGANO; DANCIK, 2016) e Transcriptome Analysis Console (AFFYMETRIX, 2017). Por outro lado, dados de RNA-seq, encontrados no GEO pela categoria "Expression profiling by high throughput sequencing", possuem uma variedade mais restrita de programas, como Chipster (KALLIO et al., 2011) e Genomics Workbench (LIU; DI, 2020), possivelmente devido à sobrecarga de memória característica dos dados produzidos por esse tipo de análise. A Tabela 1 inclui uma lista de exemplos de programas com interface gráfica de usuário para análises transcritômicas *in silico*. Além disso, é possível analisar programaticamente ambas as tecnologias utilizando a linguagem de programação R, que conta com pacotes

especializados como: *affy* e *oligo* para plataformas Affymetrix (GAUTIER et al., 2004; CARVALHO; IRIZARRY, 2010); *beadarray* e *lumi* para plataformas Illumina (DUNNING et al., 2007; DU; KIBBE; LIN, 2008); *limma* para plataformas Agilent e GenePix e para análises de expressão diferencial com microarranjo (RITCHIE et al., 2015); e *DESeq2* ou *edgeR* para expressão diferencial com RNA-seq (LOVE; HUBER; ANDERS, 2014; ROBINSON; MCCARTHY; SMYTH, 2010).

Tabela 1 – Exemplos de *softwares* utilizados para análises de transcritomas incluindo expressão diferencial. A primeira coluna indica o nome do programa. A segunda coluna indica os tipos de transcritoma suportados pelo *software*. A terceira coluna indica as fabricantes cujos dados brutos podem ser processados pelo programa, onde N/A (não-aplicável) indica suporte restrito aos formatos pré-processados do GEO. A quarta coluna indica o ambiente ou sistema operacional em que o programa é executado, sendo os tipos "Máquina virtual", "R" e "Web" não específicos a um sistema operacional.

Software	**Tipo de transcritoma**	**Fabricantes suportadas**	**Ambiente ou Sistema operacional**
Babelomics	Microarranjo	Agilent; Affymetrix; GenePix	Web
Chipster	Microarranjo; RNA-seq	Affymetrix; Agilent; Illumina	Máquina Virtual
eUTOPIA	Microarranjo	Affymetrix; Agilent; Illumina	R
GEAP	Microarranjo	Agilent; Affymetrix; GenePix; Illumina; NanoString; NimbleGen	Windows; Linux Ubuntu
GenomeStudio	Microarranjo	Illumina	Windows
Genomic Workbench	Microarranjo; RNA-seq	Agilent	Windows; Linux RHEL/SUSE; Mac OSX
GEO2R	Microarranjo	N/A	Web
ShinyGEO	Microarranjo	N/A	Web
Transcriptome Analysis Console	Microarranjo	Affymetrix	Windows

Em todos os casos de análise, haverá um conjunto de etapas para processamento dos dados, resultando em uma tabela que será utilizada para avaliação da expressão diferencial. Nesta tabela, cada coluna é uma amostra biológica, e suas linhas apresentam números que indicam o quanto que um determinado gene foi detectado para esta amostra. Como sugerido na introdução, replicatas das amostras são usadas para cada condição biológica a ser estudada por motivos de maior significância estatística. A partir disso, o usuário irá comparar um conjunto de amostras em condição experimental contra um conjunto de amostras controle, obtendo para cada gene seu valor de expressão diferencial ($logFC$).

Considerando que este trabalho envolverá o processamento destes valores de $logFC$ programaticamente, a sessão seguinte introduzirá a linguagem R como base para os métodos

explicados nos demais capítulos.

2.2 Programação em R

2.2.1 A linguagem

R é uma linguagem de programação cuja aplicação é voltada especialmente para análises estatísticas e geração de gráficos (IHAKA; GENTLEMAN, 1996). Por ser uma linguagem interpretada, seu funcionamento se dá por comandos de entrada a partir do usuário, retornando um resultado para cada chamada em linha de comando – diferentemente de linguagens como C e Fortran, onde todas as instruções devem ser compiladas antes da execução do programa.

O R pode ser inicializado diretamente como programa executável ou através de um ambiente de desenvolvimento integrado que facilita o gerenciamento de dados e a construção de comandos na linguagem, como o programa RStudio (ALLAIRE, 2012). Após a inicialização, o usuário já pode dar entrada suas próprias instruções, como por exemplo, um simples cálculo de soma:

```
> 2 + 2
# Retorna:
# 4
```

No *script* acima, linhas começando com `>` representam comandos de entrada do usuário, enquanto que `#` denota um texto de comentário para fins de esclarecimento. Manteremos esse formato nos exemplos posteriores.

Além de comandos com operadores matemáticos, o R também possui funções mais complexas definidas por padrão e que podem ser chamadas através de parâmetros. Por exemplo, pode-se chamar a função `log` para calcular logaritmos de base 4:

```
> log(16, 4)
# Retorna:
# 2
> log(29, 4)
# Retorna:
# 2.42899
```

Como mencionado, cada chamada em linha de comando no R retorna um valor resultante, o qual é impresso na tela para o usuário. Além de chamadas comuns, o usuário pode armazenar esses valores retornados em um objeto usando o operador `<-` (também é possível usar o operador `=`), possibilitando acessar esse dado em outra ocasião.

```
> resultado <- log(16, 4) # armazena resultado do logaritmo
> resultado               # chama o valor do resultado
# Retorna:
# 2
```

A operação acima irá instanciar um **objeto**, o qual pode ser acessado pelo seu nome (neste exemplo, `resultado`) e também utilizado como parâmetro para outras funções. Diversos tipos de objeto podem ser instanciados no R, sendo os mais básicos:

- `integer`: números inteiros;
- `numeric`: números decimais;
- `logical`: valores TRUE ou FALSE (similar a `Boolean` em outras linguagens);
- `character`: texto (similar a `String` em outras linguagens);
- `list`: lista de objetos.

Objetos representados por valores inteiros, decimais, booleanos e textuais são considerados **vetores** e podem ser compostos por um ou mais elementos do mesmo tipo. Por exemplo, o comando `c("lagoa", "azul")` retorna um vetor `character` composto pelos elementos "lagoa" e "azul", enquanto que o comando `1:5` gera um vetor numérico de 1 a 5 (equivalente ao comando `c(1, 2, 3, 4, 5)`). Esses tipos também são a base de dados para objetos mais complexos – por exemplo, um objeto `matrix` é um vetor com implementações adicionais para representar uma matriz bidimensional.

Listas, por outro lado, podem conter elementos de qualquer tipo de objeto. Por exemplo, o comando `list(c("lagoa", "azul"), 1:5)` cria uma lista contendo um vetor `character` e outro numérico. Alguns tipos pré-definidos no R também usam listas como base para aplicações mais específicas, como é o caso do `data.frame`, uma tabela bidimensional onde as colunas podem ser vetores de diferentes tipos. Outros tipos também são possíveis no R, sendo geralmente uma derivação de um destes quatro tipos mais simples, o que permite gerenciar seus dados de forma organizada para uma aplicação específica.

Além de instanciar objetos, o usuário também pode definir novas funções usando a sintaxe da linguagem. Abaixo, temos um exemplo de função que calcula a hipotenusa a partir de dois catetos de um triângulo através da fórmula de Pitágoras:

```
> pitagoras <- function(catA, catB) { sqrt(sum(catA^2, catB^2)) }
> pitagoras(3, 4)
# Retorna:
# 5
```

Esta nova função utiliza outras funções predefinidas do R, sendo que `sqrt` calcula a raiz quadrada, `sum` soma todos os parâmetros e o circunflexo (`^`) calcula a potência de um valor a partir do formato `valor^potencia`. As chaves (`{` e `}`) definem o corpo da função, sendo necessárias quando a função é composta por múltiplas linhas de comando, mas opcionais para linhas únicas como no exemplo acima (inseridas apenas para fins didáticos). Deste modo, operações mais complexas podem ser construídas de acordo com a lógica formada pelo programador.

A despeito das definições básicas acima, um dos maiores benefícios que o R oferece para o contexto da Bioinformática é o vasto acervo de pacotes publicados e continuamente atualizados. A seguir, será explicado o propósito destes pacotes nessa linguagem.

2.2.2 Pacotes (*R Packages*)

Como mencionado na seção anterior, o usuário pode construir suas próprias funções, instanciar objetos e criar novos tipos de dados, mantendo essas definições durante toda a execução do R. Um pacote, por sua vez, tem o propósito de reunir as definições em um único módulo para que sejam reutilizadas em outra ocasião, muitas vezes oportunizando sua publicação e aplicação por outros usuários. Ele também facilita a descrição das funções desenvolvidas, mantém um escopo para funções internas, congrega definições de classes e sinaliza quando há pacotes exigidos a fim de facilitar a resolução de dependências. Um pacote também pode conter várias versões e ser atualizado sempre que uma nova versão for lançada, a fim de se obter melhorias para as funções e classes desenvolvidas sem prejudicar sua compatibilidade. Em contraste, outras formas de armazenamento, como a concatenação dos *scripts* em um único arquivo `.R` ou a serialização dos dados para um arquivo `.RData`, podem não funcionar caso ocorram alterações nas definições de origem.

Usando um pacote

Para utilizar um pacote, é necessário que este tenha sido previamente instalado. Neste sentido, existem diferentes funções que permitem sua obtenção via *download* dependendo do repositório que o disponibiliza, sendo algumas das principais:

```
# Para pacotes disponíveis no repositório CRAN:
> install.packages("nomedopacote")
# Para pacotes do Bioconductor (requer pacote "BiocManager"):
> BiocManager::install("nomedopacote")
# Para pacotes hospedados no GitHub (requer pacote "devtools"):
> devtools::install_github("nomedesenvolvedor/nomedopacote")
```

Os pacotes instalados são armazenados em uma biblioteca local, geralmente um subdiretório da pasta de usuário do sistema operacional. Para utilizar um pacote instalado,

é necessário carregá-lo primeiramente, o que pode ser feito com o comando abaixo:

```
# Carrega um pacote. Não utilizar aspas neste caso
> library(nomedopacote)
```

Com o pacote carregado, todas as declarações inclusas passarão a ser reconhecidas pelos comandos do usuário. Evidentemente, este procedimento também será necessário para carregar o pacote final desenvolvido neste trabalho.

Este capítulo apresentou, brevemente, os fundamentos básicos do R, assim como a obtenção e o carregamento de seus módulos importáveis. Em virtude da portabilidade característica de um pacote do R, procedimentos cada vez mais complexos podem ser desenvolvidos e distribuídos para o usuário final. Além disso, novos pacotes podem utilizar métodos de outros pacotes em etapas intermediárias, deste modo usufruindo da robustez de procedimentos previamente estabelecidos.

2.3 Conceitos estatísticos

Estatística possui uma participação fundamental em todos os estágios de uma análise biológica, além de ser uma das principais finalidades da programação em R. Do mesmo modo, um dos alicerces deste trabalho é a aplicação estatística dentro de um contexto biológico a fim de se obter padrões de variação. Esta seção apresenta brevemente alguns conceitos que foram empregados durante o desenvolvimento deste trabalho.

2.3.1 Cálculos de Variação

No ponto de vista estatístico, variação é uma forma de expressar o quanto que um conjunto de valores numéricos diferem entre si. Usa-se também o termo *desvio* quando essas diferenças são calculadas a partir de um valor representativo, como a média ou a mediana dos valores do conjunto. Os principais métodos que utilizamos para calcular variação foram **desvio-padrão**, **variância** e **desvio absoluto mediano**.

O **desvio-padrão** (representado por σ) representa a distância esperada de um número qualquer em relação à média de um conjunto de dados, caso fosse inserido neste conjunto. Seu valor pode ser obtido a partir da seguinte fórmula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N}} \tag{2.1}$$

Onde N é a quantidade de valores no conjunto e $\overline{x}$ é a média desses valores. Essa fórmula também possui uma variante que usa $N - 1$ como divisor quando os valores são uma amostra dentro de uma população, a qual serve para encontrar o **desvio-padrão amostral**

(representado por s). Sua fórmula é expressa da seguinte forma:

$$s = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1}} \tag{2.2}$$

A **variância** (representada por σ^2) é definida como o desvio padrão elevado ao quadrado, sendo calculada pela seguinte fórmula:

$$\sigma^2 = \frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N} \tag{2.3}$$

Da mesma forma, para subconjuntos, a **variância amostral** (representada por μ) pode ser expressa por:

$$\mu = \frac{\sum_{i=1}^{N}(x_i - \overline{x})^2}{N-1} \tag{2.4}$$

O desvio absoluto mediano (representado pela sigla **MAD**, do inglês *median absolute deviation*), por outro lado, utiliza a mediana como valor central de referência para o cálculo do desvio, sendo uma alternativa mais robusta quando há valores discrepantes no conjunto. Para um conjunto de valores ordenados, sua fórmula é definida por:

$$MAD = mediana(|x_i - \tilde{x}|) \tag{2.5}$$

Onde $\tilde{x}$ é a mediana dos valores iniciais do conjunto, e $|x_i - \tilde{x}|$ (equivalente a $\sqrt{(x_i - \tilde{x})^2}$) é o desvio absoluto calculado para cada item x_i no conjunto. A mediana mais externa é aplicada sobre todos os desvios absolutos resultantes, obtendo-se um desvio mediano.

É possível também aplicar essas fórmulas no R sobre um vetor numérico a partir das funções `sd` (para desvio-padrão), `var` (para variância) e `mad` (para desvio absoluto mediano). Por exemplo, supondo que quiséssemos aplicar para os valores 1, 4, 5, 6 e 9, teríamos os seguintes comandos:

```
# Define o vetor:
> v <- c(1, 4, 5, 6, 9)
# Desvio-padrão:
> sd(v)   # Retorna 2.915476
# Variância:
> var(v)  # Retorna 8.5
# Desvio absoluto mediano:
> mad(v)  # Retorna 1.4826
```

A diferença dos valores retornados acima é evidente, principalmente porque os valores 1 e 9 do vetor estão bem distantes dos outros três (4, 5 e 6), o que pode ser considerado um caso de discrepância. Se, no mesmo exemplo, o número 36 fosse colocado

no lugar do valor 9, os valores resultantes de variância e desvio-padrão aumentariam drasticamente (`208.3` e `14.4326`, respectivamente), enquanto que o de MAD permaneceria o mesmo (`1.4826`). Há também o caso trivial em que todos os valores são iguais, e portanto qualquer um dos métodos retornará zero – ou seja, nenhuma variação.

Sendo assim qualquer um dos três métodos apresentados pode ser aplicado para um conjunto de valores numéricos a fim de mensurar sua variação.

2.3.2 Análises de Variância (ANOVA)

Uma **análise de variância** (**ANOVA**) é um tipo de procedimento estatístico que busca avaliar variações entre múltiplos grupos de observações. Embora existam diferentes metodologias para ANOVA, a ideia básica consiste em testar a variância para subconjuntos de valores, separados de acordo com um atributo ou variável-resposta, a fim de verificar se esses valores poderiam estar ou não variando de acordo com seu agrupamento estabelecido.

Uma ANOVA é uma medida estimativa, e portanto, possui uma probabilidade de a associação entre os grupos ser errônea (*i.e.*, hipótese nula). Essa probabilidade é expressa pelo *valor-p* obtido, onde valores menores refletem maiores chances do agrupamento ser estatisticamente significante. Há um leque de opções para calcular o valor-p, feito principalmente por algoritmos, porém a escolha do algoritmo irá depender de como os dados estão distribuídos e que tipo de resposta se está buscando. Para este trabalho, foram utilizados os métodos de **Fisher** e de **Levene**.

Método de Fisher

O **método de Fisher** (também conhecido como *F-test*) estima a variação dos grupos por meio da razão de variâncias entre todos os grupos ($\mu_{(intergrupos)}$) e de dentro de cada grupo ($\mu_{(intragrupos)}$). A fórmula para calcular o *F-test* pode representada por:

$$F = \frac{\mu_{(intergrupos)}}{\mu_{(intragrupos)}} \tag{2.6}$$

$$\mu_{(intergrupos)} = \sum_{g=1}^{K} N_g \frac{(\overline{x}_g - \overline{x})^2}{K-1} \tag{2.7}$$

$$\mu_{(intragrupos)} = \sum_{g=1}^{K} \sum_{i=1}^{N_g} \frac{(x_{gi} - \overline{x}_g)^2}{N-K} \tag{2.8}$$

Onde K é o número de grupos; N_g é o número de valores dentro de um grupo; $\overline{x}_g$ é a média dos valores de um grupo; $\overline{x}$ é a média total de todos os elementos; x_{gi} é o i-ésimo elemento do grupo; e N é o número total de elementos. O valor de F será alto se a variância entre grupos for maior do que dentro de cada grupo, e menor se todos os grupos possuírem a mesma variação.

Em outras palavras, se os valores de dentro de um grupo estão próximos entre si, mas distantes em relação aos valores de outros grupos, é mais provável que os valores estejam sendo moldados de acordo com esse agrupamento. Se esses grupos forem diferentes atributos observados experimentalmente, isso também poderia indicar uma associação de causa ou efeito dos valores dependendo do atributo experimental. Da mesma forma, seria pouco confiável inferir essa associação se os valores estiverem variando de forma aparentemente aleatória independentemente do grupo experimental que se encontra, visto que não haveria como distinguir sua variação entre grupos.

Apesar de sua ampla aplicação, o teste de Fisher possui algumas limitações, dentre elas a exigência de que os valores amostrais encontrem-se normalmente e similarmente distribuídos. Por esta razão, sua estimativa não funciona adequadamente para comparar variâncias entre grupos de observações (HOSKEN; BUSS; HODGSON, 2018).

O teste de Fisher é disponível no módulo básico do R, sendo o teste padrão para a função `aov` para o cálculo de ANOVA. O código abaixo demonstra um exemplo de caso de uso:

```
# Define o vetor de valores:
> v <- c(0.84, 1.22, 1.32, -1.37, -2.17, -1.51, 4.23, 4.93, 3.92)
# Define os grupos (g1, g2 e g3):
> g <- c("g1", "g1", "g1", "g2", "g2", "g2", "g3", "g3", "g3")
# Teste de ANOVA padrão (Fisher):
> result <- anova(aov(v ~ g))
> result
# Resultado impresso na tela:
# Analysis of Variance Table
# Response: v
#           Df Sum Sq Mean Sq F value    Pr(>F)
# g          2 54.872 27.4362  160.02 6.232e-06 ***
# Residuals  6  1.029  0.1715
# Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
#
> result[["Pr(>F)"]][1]   # para obter apenas o valor-p
# Resultado:
# 6.232e-06
```

Como observado acima, ao chamar a variável `result` contendo o resultado da ANOVA, o R imprime uma tabela com os resultados do teste. O valor-p é determinado pela coluna `Pr(>F)` (linha `g`), que apresenta o valor `6.232e-06` neste exemplo. Também é possível obter este valor numericamente pelo comando `result[["Pr(>F)"]][1]`, também retornando `6.232e-06` (ou $0,000006232$). No ponto de vista estatístico, onde costuma-se usar pontos de corte de $p < 0,05$ ou $p < 0,001$ para significância do valor-p, esse resultado

seria considerado bastante significante, tendo em vista que os valores internos dos grupos "g1", "g2" e "g3" estão bem próximos entre si. Ser significante, neste contexto, representa ter alta probabilidade dos valores estarem consistentemente distinguidos através destes grupos atribuídos.

Em uma situação oposta, o valor-p seria bastante alto se os grupos não tivessem sido bem estabelecidos:

```
# Define o vetor de valores:
> v <- c(0.84, 1.22, 1.32, -1.37, -2.17, -1.51, 4.23, 4.93, 3.92)
# Define os grupos (g1, g2 e g3):
> g <- c("g1", "g2", "g3", "g1", "g2", "g3", "g1", "g2", "g3")
# Teste de ANOVA padrão (Fisher):
> result <- anova(aov(v ~ g))
> result[["Pr(>F)"]][1]   # chama apenas o valor-p
# Resultado:
# 0.9991547
```

No caso acima, cada grupo possui uma alta variação de seus valores, de forma que seu agrupamento atribuído tem pouca significância estatística, indicando pouca viabilidade de estabelecer alguma relação entre os grupos e os valores.

Testes de Levene e Brown–Forsythe

O teste de Levene tem como propósito comparar variâncias entre diferentes grupos, avaliando se dois ou mais grupos amostrais estão igualmente distribuídos (LEVENE, 1960; CARROLL; SCHNEIDER, 1985). Deste modo, em contraste com o método de Fisher, o método de Levene não possui a limitação de distribuição normal e similar, sendo mais adequado para detectar flutuações entre valores amostrais (HOSKEN; BUSS; HODGSON, 2018). O método de Brown-Forsythe, por sua vez, é uma variante de Levene onde utiliza-se mediana em vez de média para o cálculo do valor central de cada grupo (BROWN; FORSYTHE, 1974).

Os testes estatísticos de Levene e Brown–Forsythe, denotados por W, podem ser representados pelas seguintes equações:

$$W = \frac{\mu_{(intergrupos)}}{\mu_{(intragrupos)}} \tag{2.9}$$

$$\mu_{(intergrupos)} = \sum_{g=1}^{K} N_g \frac{(\overline{x}_g - \overline{x})^2}{K-1} \tag{2.10}$$

$$\mu_{(intragrupos)} = \sum_{g=1}^{K} \sum_{i=1}^{N_g} \frac{(Z_{gi} - \overline{x}_g)^2}{N-K} \tag{2.11}$$

Onde:

$$Z_{gi} = \begin{cases} |x_{gi} - \overline{x}_g| & \text{(Levene)} \\ |x_{gi} - \tilde{x}_g| & \text{(Brown-Forsythe)} \end{cases}$$

Embora o R não possua uma implementação do método de Levene em seus módulos-base, seu teste pode ser acessado por pacotes externos como *car* (FOX et al., 2012). Para referência, um exemplo de código em R encontra-se disponível na seção B.1 do Apêndice B.

2.3.3 Clusterização

Clusterização (do inglês, *clustering*) é um tipo de análise que possui a finalidade de separar e classificar dados de acordo com uma ou mais de suas propriedades, atribuindo-os a grupos (ou *clusters*). Dentre suas principais aplicações, está a separação de pontos de um espaço cartesiano para grupos de pontos distinguíveis, de forma que seja possível usar estes conjuntos para alguma finalidade.

Neste trabalho, quatro principais algoritmos de clusterização foram levados em conta: ***K-means***, **máxima verossimilhança**, **hierarquia** e **densidade** (Figura 2). Como será visto, nem todos apresentaram aplicabilidade para o atual contexto, sendo mantidos apenas os algoritmos hierárquicos e de densidade durante a implementação final. A seguir, apresentaremos alguns detalhes sobre cada método e aplicação.

Clusterização K-means

O ***K-means*** é um dos métodos mais populares de clusterização, sendo sua versão mais básica relativamente rápida e simples de implementar em comparação com a de outros algoritmos (RAYKOV et al., 2016). Seu procedimento consiste em particionar um conjunto de pontos em um número K de grupos, o que torna necessário estabelecer um número de *clusters* previamente (Figura 2A).

Cada um destes K grupos possui um centroide inicial, que é inicializado com coordenadas aleatórias. Neste sentido, testa-se para todos os pontos qual o centroide mais próximo (*i.e.* menor distância euclidiana), associando cada ponto ao centroide vencedor. A seguir, cada grupo tem seu centroide redefinido como o ponto médio entre os pontos, e o teste inicial é repetido até que haja pouca alteração nos grupos de pontos, assim estabelecendo os *clusters*.

O principal benefício deste algoritmo, como mencionado, é sua facilidade de implementação, além de ser consideravelmente rápido devido a sua baixa complexidade. Uma de suas limitações, contudo, é a necessidade do conhecimento prévio de número de *clusters*, tornando necessário a interferência do usuário ou alguma programação mais complexa que permita de estimar o valor de K. Outra limitação é devido à natureza de sua

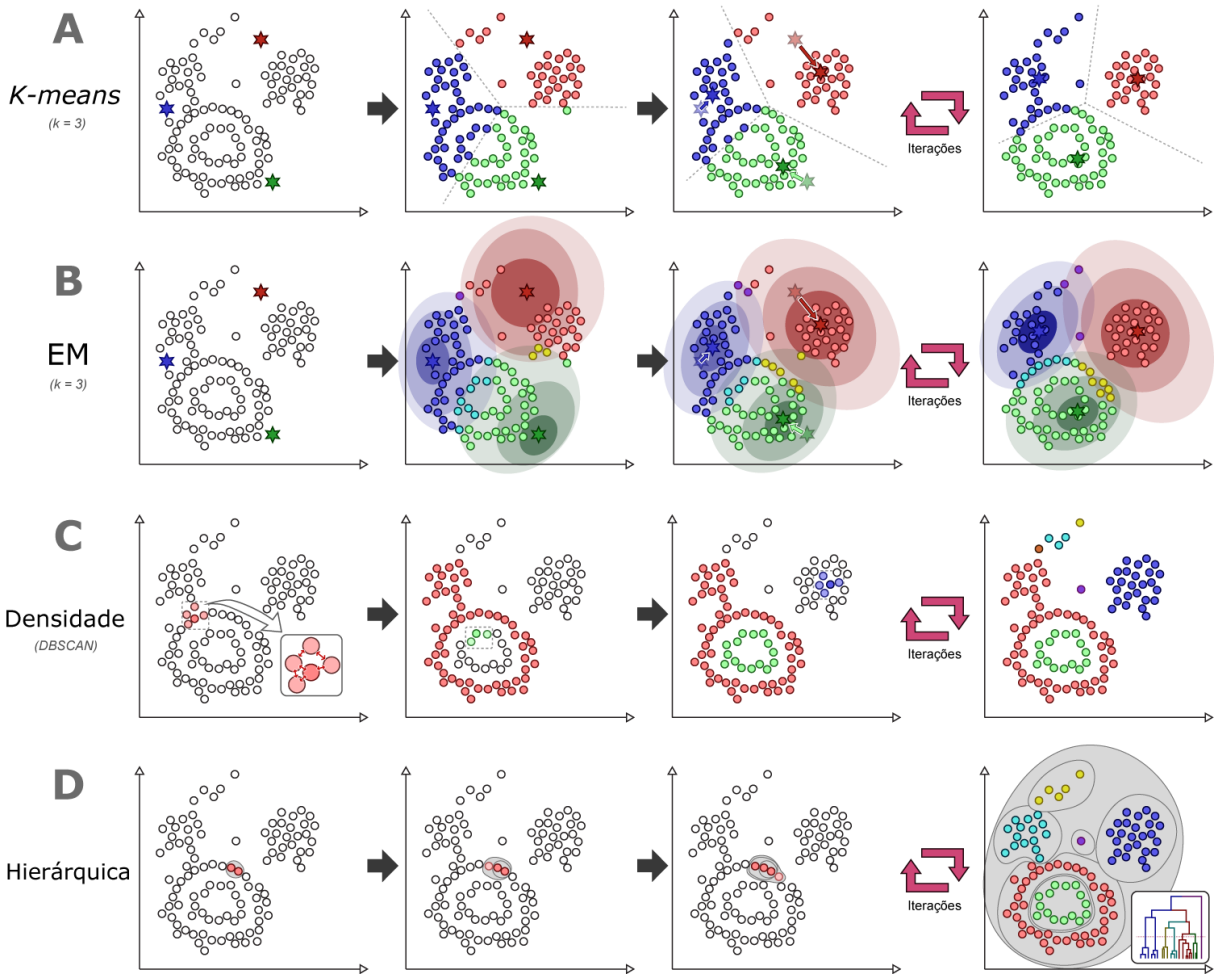

Figura 2 – Representação esquemática do algoritmo de cada um dos tipos de clusterização avaliados durante o projeto desenvolvido. As cores indicam o *cluster* atualmente atribuído a cada ponto. As setas de repetição em rosa-salmão indicam uma quantidade significa de repetições dos passos até que não haja mudanças significativas (*k-means* e EM) ou pontos restantes para agrupar (densidade e hierárquica). Os símbolos de estrela (★) apontam o centroide atual de cada *cluster* nos algoritmos *k-means* e EM, sendo inicializados aleatoriamente para então convergirem a um novo centro em cada iteração. As áreas coloridas em EM representam nuvens de probabilidade, onde pontos que possuem a chance de inclusão a mais de um *cluster* foram marcados com cores mistas (*e.g.*, vermelho + verde = amarelo). Na clusterização por densidade, a iteração ocorre entre os pontos mais próximos de seus vizinhos até que não exista mais pontos suficientemente dentro do alcance. Na clusterização hierárquica, uma hierarquia é definida para todos os pontos, gerando-se um dendrograma final que é seccionado a partir de um limiar de distância entre camadas hierárquicas, cada uma contendo um ou mais *clusters* como sub-camadas.

classificação por "fatiamento" do espaço euclidiano, o que impede a distinção de *clusters* mais complexos, como nuvens côncavas ou formatos encaixados.

Clusterização de máxima verossimilhança

O algoritmo de clusterização por **máxima verossimilhança** (**EM** do inglês, *expectation-maximization*) segue procedimentos similares ao *K-means* básico, com a principal diferença de que os pontos classificados possuem uma probabilidade associada a diferentes *clusters* (Figura 2B). Desta forma, para cada iteração, uma probabilidade é calculada de cada ponto pertencer a cada *cluster*. Embora este método produza mais informações e seja mais robusto que o *K-means*, ainda permanecem as limitações em termos de número prévio de *clusters* e pouca precisão para certas distribuições de pontos.

Clusterização por densidade

Em contraste aos outros dois métodos apresentados acima, que dividem o espaço euclidiano em secções convexas, a **clusterização por densidade** realiza seu agrupamento por meio da densidade dos pontos, a qual é calculada para cada ponto levando em conta sua proximidade com pontos próximos (Figura 2C). Neste caso, a entrada do usuário é um valor que representa a distância de agrupamento de pontos. O algoritmo usa esse valor para comparar com a distância de um ponto para todos os pontos próximos, concatenando em um mesmo *cluster* dois pontos que estiverem a uma distância menor que esse valor. Deste modo, aglomerados de pontos podem ser agrupados independente de sua forma, contanto que estejam similarmente densos. Além disso, o número resultante de *clusters* depende apenas da distribuição dos dados e do valor de distância, sendo assim mais adaptável a grandes anormalidades dos dados em comparação ao *K-means*. Todavia, pode apresentar limitações na distinção de *clusters* com grande variação de densidade, ou na presença de estreitos densos entre dois conjuntos de pontos que deveriam estar separados, visto que basta um ou poucos pontos próximo para ocorrer a concatenação em um único *cluster*.

Clusterização hierárquica

A **clusterização hierárquica**, como o próprio nome sugere, classifica os pontos por meio de uma hierarquia, a qual é composta por múltiplas camadas de agrupamento (Figura 2D). Nesta hierarquia, cada camada é composta por um ponto e um grupo, e a posição hierárquica é definida pela distância de um ponto para a sua camada mais baixa, de forma que camadas mais baixas representam pontos mais próximos entre si. A partição espacial é feita calculando-se, a partir de um parâmetro estabelecido, a menor distância entre dois pontos, seja ele um ponto definido ou o centroide de uma camada hierárquica. Com a menor distância calculada, uma nova camada é formada contendo um centroide entre estes pontos, que será comparado com pontos de outras camadas. Deste modo, um conjunto com N pontos pode gerar de 1 a $N-1$ *clusters*, onde um ponto de corte de distância entre camadas hierárquicas definirá o número final de agrupamentos. Além disso, em contraste com a clusterização por densidade, o método hierárquico consegue diferir agrupamentos

contendo densidades distintas, desde que possuam uma distribuição similar. Contudo, sua limitação é principalmente relacionada a eficiência, pois cada iteração envolve o cálculo da distância do centroide de uma camada hierárquica com todos os pontos restantes, sendo inviável para análises com milhares de pontos em sua implementação original. Para contornar este problema, são oferecidos pacotes com implementações heurísticas deste algoritmo, como o *fastcluster* (MÜLLNER et al., 2013), que resultam em clusterizações virtualmente idênticas ao algoritmo-base.

Por fim, considerando que a clusterização pode ser utilizada para agrupar valores, esse tipo de método também poderia ser aplicado para auxiliar a classificação de múltiplas observações de expressão diferencial. Seus algoritmos podem, portanto, ser combinados com os demais métodos estatísticos apresentados neste capítulo em uma única *pipeline* metodológica para avaliar diversas condições experimentais em um mesmo contexto.

3 Objetivos

3.1 Proposta geral

O objetivo deste trabalho foi **desenvolver um pacote do R para analisar variações na expressão diferencial em múltiplas condições biológicas**.

3.2 Objetivos específicos

- Desenvolver um sistema de métodos para avaliar a variação de expressão diferencial em múltiplas condições experimentais;
- Desenvolver funções e classes (tipos de objetos) em R que possibilitem a aplicação da metodologia desenvolvida;
- Concatenar todas as classes em funções em um único pacote do R, de modo que possam ser importadas em qualquer outro computador com a versão atual do R instalada;
- Testar o pacote com dados reais em um número significante de casos de uso, avaliando precisão e confiabilidade da obtenção dos resultados;
- Disponibilizar o pacote publicamente, possibilitando sua aplicação por qualquer usuário.

Parte II

Desenvolvimento

4 Sistema Proposto

O pacote desenvolvido neste trabalho tem o propósito de não apenas **automatizar** procedimentos existentes, mas também de **criar uma metodologia** que reúne múltiplas aplicações estatísticas e computacionais para identificar genes relevantes entre duas ou mais condições experimentais. Além disso, a lógica de seu desenvolvimento foi intencionalmente planejada em torno da aplicação sobre **resultados de expressão gênica diferencial**, não havendo pretensão de extrapolar este tipo de análise para outros casos de uso. Dito isso, seu desenvolvimento também passou por uma série de etapas de testes de tolerância de erros e *benchmark* com o propósito de produzir resultados flexíveis. Neste capítulo, será apresentado o sistema de aplicações metodológicas que serviu como alicerce na produção do pacote final.

4.1 Racional do método desenvolvido

Como qualquer outro método computacional, este pacote tem o intuito de **automatizar** um conjunto de procedimentos, o qual deve ser desconstruído e avaliado em etapas para posteriormente ser replicado em forma de código. O que será "automatizado", neste caso, é a observação empírica do que se está variando ao levar em conta expressões diferenciais em diferentes condições.

O racional da proposta de encontrar variações a partir da expressão diferencial ($logFC$), e não diretamente dos transcritomas, se deve à representação de seus valores. Amostras transcritômicas apresentam a quantidade de transcritos detectados durante um determinado experimento, sendo que a escala de seus valores é sempre relativa às demais amostras do mesmo estudo. Mesmo quando diferentes experimentos utilizam uma plataforma transcritômica em comum, os níveis detectados estarão em escalas distintas de normalização dependendo do protocolo utilizado, seja por mudanças sutis nos reagentes ou durante a amplificação do mRNA, além de outras diversas variáveis envolvidas. Por esta razão, dados brutos de expressão gênica não são comumente adequados de se comparar entre diferentes GSE num único contexto (*e.g.*, em uma mesma matriz do R), pois não há garantia que seus procedimentos experimentais tenham sido replicados exatamente da mesma maneira e tenham permanecido na mesma escala.

Em contrapartida, o $logFC$ permanece sempre na mesma escala independentemente dos níveis normais que flutuam entre conjuntos de dados. Isso ocorre pois *fold-change* é uma medida de razão – por exemplo, se um transcrito variou do nível $1,0$ para $2,0$ ou de $2,0$ para $4,0$, ele continuou variando em dobro. A escala log também favorece a separação entre genes super- e sub-expressos e diminui discrepâncias: por exemplo, um gene que teve

o dobro de expressão gênica detectada em um experimento e apenas metade em outro terá, respectivamente, um *fold-change* direto de $2,0$ e $0,5$, enquanto respectivamente um $logFC$ de $1,0$ e $-1,0$, mais representativo. Por esta razão, torna-se mais viável assumir alterações biológicas usando os próprios GDEs e adotando o $logFC$ como medida comparativa.

4.2 Analisando variações manualmente

4.2.1 Passo 1: Análises de expressão diferencial

Considerando que os dados de entrada são resultados expressos em $logFC$, o primeiro passo consiste em realizar a própria expressão diferencial (Figura 3). Isso pode ser feito com uma série de programas, como o GEO2R para dados do GEO previamente curados, o Transcriptome Analysis Console para dados da Affymetrix, ou o GEAP[1], que utiliza o R em plano de fundo para análises de microarranjo, dentre outras ferramentas citadas na seção 2.1. Qualquer que tenha sido o método escolhido, todos eles resultarão em tabelas de expressão diferencial, cada uma contendo uma coluna de valores de $logFC$, e outra de valores-p associados à confiabilidade de cada $logFC$.

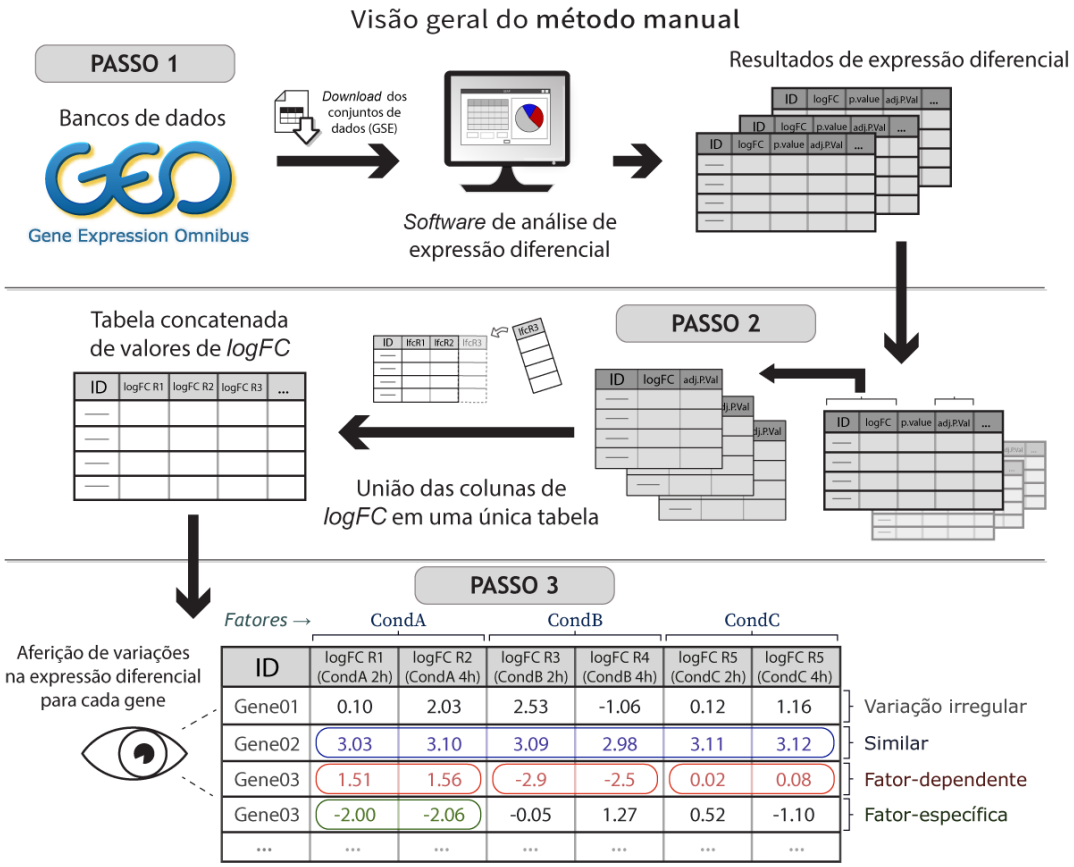

Figura 3 – Resumo esquemático do método manual apresentado nesta seção.

[1] Programa que desenvolvi durante o bacharelado e mestrado para facilitar análises de microarranjo. Pode ser acessado pelo link: <https://inf.ufrgs.br/geap>

4.2.2 Passo 2: União dos resultados

O segundo passo consiste em concatenar os resultados de $logFC$ em uma única tabela, a fim de comparar os valores de expressão diferencial para cada gene (ou sonda, para microarranjos). Essa concatenação pode ser feita não apenas pelo R, mas também por um editor de planilhas como o *Microsoft Excel* ou o *OpenOffice Calc*, embora o uso de uma linguagem de programação ofereça maior controle sobre os dados processados. Os *softwares* de planilhas também costumam inicializar com uma configuração padrão que introduz erros de nomeação de genes (ZIEMANN; EREN; EL-OSTA, 2016), o que requer cuidado extra durante a manipulação de tabelas com dados de larga escala. Em quaisquer casos, a criação de uma tabela única permitirá a observação direta de quais genes podem ter um $logFC$ similar ou variante entre os resultados observados.

4.2.3 Passo 3: Classificação das variações

O terceiro passo consiste em classificar os genes de acordo com o comportamento de sua expressão diferencial – isto é, a maneira como seu $logFC$ varia entre os resultados. Teoricamente, poderia-se pensar em inúmeras maneiras de classificar desta forma, porém este trabalho será focado em apenas algumas destas possibilidades. Em um primeiro momento, ao observar cuidadosamente cada linha de $logFC$, nota-se duas situações evidentes: em algumas linhas, existe uma **variação irregular** dos valores de $logFC$, a qual não apresenta nenhuma resposta evidente, enquanto em outras, o $logFC$ apresenta com um valor **similar** entre todas as colunas, indicando que a expressão diferencial é similar para todas as análises. Contudo, dentre os valores similares, apenas uma pequena parte consistirá em $logFC$ significantes (muito positivos ou muito negativos), sendo estes os que ajudarão a responder a problemática em questão. Essa distinção pode ser feita a partir do valor central de $logFC$ para uma linha da tabela, que pode ser calculado por qualquer método de sumarização (*e.g.*, média ou mediana), separando valores distantes de zero. O valor central deve ser usado em conjunto com a medida de variação de seu valor (*e.g.*, desvio-padrão, variância ou MAD, ver Seção 2.3.1), a fim de selecionar baixos valores de variação que, por conseguinte, apresentam maior consistência de $logFC$ entre os resultados.

Por outro lado, essa situação é melhor aplicada quando todos os resultados observados seguem um mesmo tipo de condição experimental. Em muitas ocasiões, deseja-se também buscar respostas para grupos de experimentos – por exemplo, ao comparar diferentes tipos de doenças que podem ser dividas em categorias. Neste caso, seriam esperadas variações exorbitantes entre os valores de $logFC$ em razão das condições analisadas serem distintas por si só, embora nem todas estas variações sejam efetivamente irregulares. De fato, comparar a variação de $logFC$ no contexto de cada grupo de experimento, nesta ocasião, poderá trazer mais respostas do que avaliar todos valores globalmente. Neste

trabalho, grupos de resultados de expressão diferencial são denominados **fatores** e poderão ser levados em conta para análises de variação. Quando um $logFC$ está similar dentro de cada fator, mas variante entre diferentes fatores, determina-se que este gene possui uma **expressão diferencial fator-dependente**. Isso pode ser estimado aplicando uma ANOVA pelo método de Fisher (ver Seção 2.3.2), que funciona coerentemente para este tipo de variação. Em outros casos, porém, o $logFC$ também pode estar muito positivo ou negativo apenas dentro de um fator, mas irregular no contexto de outros fatores, significando que há uma **expressão diferencial fator-específica**. Nesta situação, como variações estão sendo comparadas, pode-se considerar pertinente a aplicação de uma ANOVA pelo método de Levene (ver Seção 2.3.2) para essa estimativa.

Para resumir, portanto, utilizamos as quatro seguintes classificações para variação de expressão diferencial:

- Variação irregular;
- Expressão diferencial similar;
- Expressão diferencial fator-dependente;
- Expressão diferencial fator-específica.

Para o contexto desta análise, os genes de maior relevância serão aqueles com expressão diferencial similar, fator-dependente ou fator-específica dentre os resultados observados. Além disso, por esta classificação ser de origem estatística e estimativa, mais de uma classificação pode ser atribuída para o mesmo gene, uma vez que satisfeitos os critérios de relevância estabelecidos pelo próprio pesquisador.

4.3 Procedimento manual revisitado

4.3.1 Tamanho dos dados de entrada

Ao examinar o método apresentado na seção anterior, é possível notar algumas limitações. A mais evidente seria a quantidade de genes sendo levados em conta – dezenas de milhares sondas para um microarranjo, por exemplo – onde, para cada gene, seria preciso efetuar uma série de etapas de cálculo e classificação. A avaliação manual de cada linha da tabela concatenada não apenas se demonstraria trabalhosa, como também abster-se-ia da certeza de não ocorrerem erros humanos para tal quantidade. Filtrar os dados de entrada por meio de um ponto de corte de $logFC$ também não seria uma opção segura, pois adicionaria viés nos parâmetros de variação observados, além de dificultar a identificação de variações quando um ou mais fatores apresentam um $logFC$ central próximo de zero. Por outro lado, pode-se filtrar esses dados pelo critério de significância

estatística – por exemplo, removendo as linhas onde todos os valores-p estão abaixo de um ponto de corte (*e.g.*, $0,05$ ou $0,01$), o que eliminaria valores de $logFC$ evidentemente não-confiáveis. Ainda assim, para manter a precisão da metodologia, é necessário que todos os $logFC$ confiáveis sejam mantidos independentemente do tamanho inicial da tabela. Portanto, torna-se indispensável que todos os cálculos e classificações sobre a tabela concatenada sejam efetuados programaticamente, dependendo do usuário apenas para ajustar os parâmetros em etapas intermediárias. Nos tópicos seguintes, serão apresentadas algumas limitações aparentes ao longo destas etapas, bem como melhorias propostas para cada ocasião.

4.3.2 Ponderação dos valores de *logFC*

Toda análise de expressão diferencial que inclui replicatas terá um valor-p, geralmente ajustado, associado ao $logFC$ resultante. Isso significa que os valores de $logFC$ não são iguais em termos de significância, também indicando a possibilidade de viés em cálculos de sumarização e variação que levam em conta apenas o $logFC$.

Uma maneira de contornar esse problema consiste em introduzir **pesos** nos cálculos de sumarização, aplicando-se as versões ponderadas das fórmulas de média e mediana. A Figura 4 ilustra a diferença entre os métodos de sumarização quando há a inclusão de pesos. Para um comparativo, as fórmulas de média ($\overline{x}$) e média ponderada ($\overline{x}_w$) seguem listadas abaixo:

$$\overline{x} = \frac{\sum_{i=1}^{N} x_i}{N} \qquad \overline{x}_w = \frac{\sum_{i=1}^{N} x_i w_i}{\sum_{i=1}^{N} w_i} \tag{4.1}$$

Onde cada w_i representa um valor de peso maior ou igual a zero. Nota-se que pelo menos um dos valores deve ser diferente de zero, a fim de evitar uma divisão por zero. Se todos os pesos forem iguais, $\overline{x}_w$ é equivalente a $\overline{x}$.

Da mesma forma, para um conjunto de valores ordenados, as fórmulas de mediana ($\tilde{x}$) e mediana ponderada ($\tilde{x}_w$) seguem listadas abaixo:

$$\tilde{x} = \frac{x_{\lceil (N+1)/2 \rceil} + x_{\lfloor (N+1)/2 \rfloor}}{2} \tag{4.2}$$

$$\tilde{x}_w = x_k \Rightarrow \{k \in 1, ..., N \mid \sum_{i=1}^{k-1} w_i \leqslant \frac{w_T}{2} \geqslant \sum_{i=k+1}^{N} w_i\} \tag{4.3}$$

Onde k é a posição do valor mediano; $w_T = \sum_{i=1}^{N} w_i$ indica a soma total dos pesos; e os símbolos $\lceil\ \rceil$ e $\lfloor\ \rfloor$ representam as funções de arredondamento *teto* e *piso*, respectivamente:

$$\lceil x \rceil = min\,\{m \in \mathbb{Z} \mid m \geqslant x\} \tag{4.4}$$

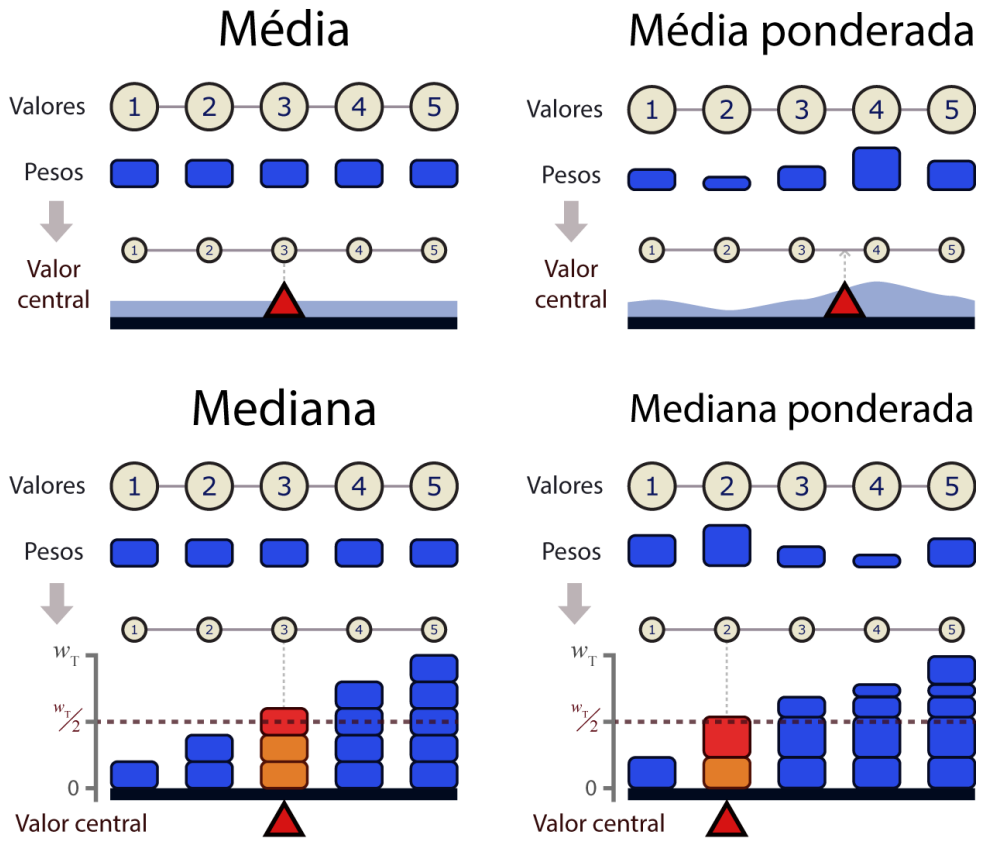

Figura 4 – Representação ilustrativa de média e mediana em comparação com suas versões ponderadas caso fossem aplicadas, por exemplo, sobre um conjunto de valores de 1 a 5. Para cada cálculo de sumarização, há uma seta vermelha indicando seu valor central resultante. Blocos azuis representam exemplos arbitrários de valores de peso, sendo idênticos entre si quando a média ou mediana não é ponderada. Na média ponderada, o valor central é tendenciado para os elementos com pesos maiores. A curva em azul-claro enfatiza a proporção dessa tendência, onde a média é deslocada para próximo do relevo mais alto. Na mediana ponderada, o valor central é encontrado quando a soma acumulada dos pesos está o mais próximo possível da metade do peso total ($\frac{w_T}{2}$), como indicado pelos blocos em vermelho e laranja.

$$\lfloor x \rfloor = max\left\{n \in \mathbb{Z} \mid n \leqslant x\right\} \tag{4.5}$$

A mediana ponderada, em outras palavras, encontra a posição central somando cumulativamente os valores dos pesos até resultar na metade da soma total dos pesos, assim definindo o valor central nesta posição (EDGEWORTH, 1888). Novamente, se todos os valores forem iguais, $\tilde{x}_w$ é equivalente $\tilde{x}$.

Assim como as fórmulas de média e mediana possuem variantes ponderadas, o mesmo vale para os cálculos de variação e ANOVA, onde em cada uma de suas fórmulas ocorre a substituição da média pela média ponderada, e da mediana pela mediana ponderada. Neste sentido, as fórmulas de desvio-padrão ponderado (σ_w), variância ponderada

(σ_w^2) e MAD ponderado (MAD_w) podem ser formalizadas respectivamente por:

$$\sigma_w = \sqrt{\frac{\sum_{i=1}^{N} w_i(x_i - \overline{x}_w)^2}{\sum_{i=1}^{N} w_i}} \tag{4.6}$$

$$\sigma_w^2 = \frac{\sum_{i=1}^{N} w_i(x_i - \overline{x}_w)^2}{\sum_{i=1}^{N} w_i} \tag{4.7}$$

$$MAD_w = mediana_w(|x_i - \tilde{x}_w|) \tag{4.8}$$

Onde $mediana_w$ é uma mediana ponderada sobre os desvios absolutos calculados como resultado de cada $|x_i - \tilde{x}_w|$.

Considerando a presente problemática dos valores $logFC$, uma sugestão seria aplicar essas versões ponderadas das fórmulas de sumarização e variação utilizando o valor-p como base para os valores de pesos. Nesta proposta, cada valor de peso w_i seria definido como:

$$w_i = 1 - p_i \tag{4.9}$$

Onde p_i é o valor-p associado a um $logFC$ do resultado i. Tendo em vista que os valores-p estão em escala logarítmica, os $logFC$ com menor relevância estatística teriam um peso exponencialmente menor, o que permitiria priorizar os valores de $logFC$ com maior significância estatística. Deste modo, mesmo não sendo uma solução convencional, inserir pesos utilizando observações estatísticas conhecidas pode ser visto como uma forma de aumentar a precisão dos cálculos durante a etapa de sumarização.

4.3.3 Definindo significância – SV e Quantis

Como já mencionado, o propósito do pacote desenvolvido é encontrar genes significantes. Em uma única análise de expressão diferencial, esses genes seriam estabelecidos por meio de um ponto de corte, como por exemplo, $logFC > 1,00$ para genes super-expressos, $logFC < -1,00$ para genes sub-expressos e $p < 0.05$ para valor-p significativo. Contudo, o mesmo ponto de corte não é aplicável no presente contexto, que envolve variação de $logFC$ entre diferentes comparações. Em vez disso, a significância é mensurada levando-se em conta dois parâmetros: **sumarização** (valor central) de $logFC$ e **variação** de $logFC$. Ao caracterizar-se com base na relação entre estas duas variáveis, serão principalmente destacados genes com um valor de sumarização muito alta ou muito baixa (genes super- e sub-expressos, respectivamente), bem pouca variação (consistência) entre comparações.

Representação SV

A **relação sumarização-variação** (SV) pode ser representada em um espaço cartesiano, como ilustrado nos gráficos da Figura 5. Esse tipo de gráfico foi nomeado

como *SV-plot*, pois os pontos de dados representam resultados dos cálculos de sumarização (eixo S) e variação (eixo V) de cada linha de uma tabela concatenada. Desta forma, é possível observar a distribuição da expressão diferencial levando em conta a relação de ambas as variáveis, o que também permite parametrizar a significância a partir de pontos destacados.

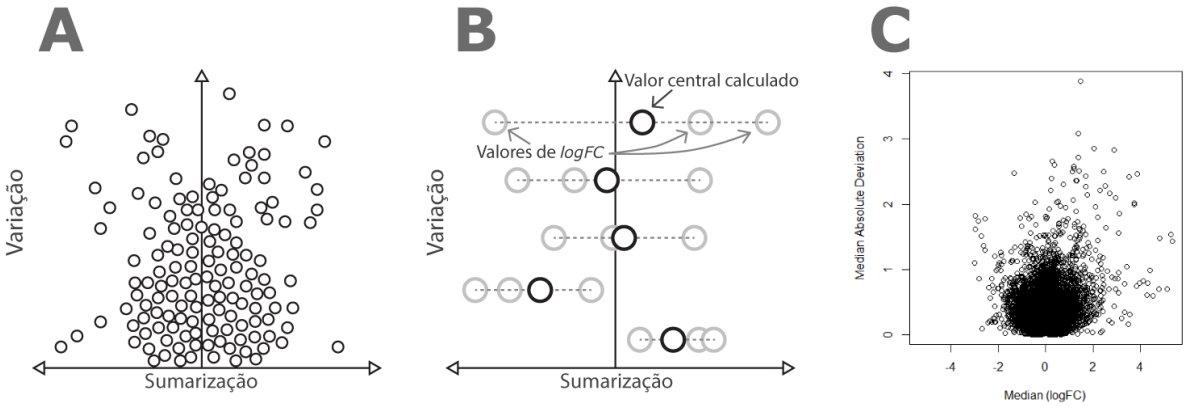

Figura 5 – Gráficos tipo *SV-plot*. (A) Exemplo ilustrado de um *SV-plot*, simulando uma distribuição de pontos ocorrente em dados reais; (B) Interpretação dos valores sumarizados (pontos contrastados) de um *SV-plot* caso os valores que os compõem (pontos transparentes) também estivessem desenhados. Como demonstrado, o eixo horizontal indica a posição do valor de *logFC* (sub-expressos à esquerda, sem expressão diferencial ao centro e super-expressos à direita), enquanto o eixo vertical determina o quão próximos (mais abaixo) ou dispersos (mais acima) estão os valores de seu ponto central; (C) Gráfico *SV-plot* gerado no R para um exemplo arbitrário de análise com dezenas de milhares de genes – quantidade usual em dados de microarranjo – utilizando mediana e MAD como métodos de sumarização e variação, respectivamente. A elevada densidade de pontos na área central inferior foi uma particularidade observada nesse tipo de gráfico, tendo sido nomeada neste trabalho como "aglomerado indiferencial".

Quantis

A partir dos pontos SV estabelecidos em um espaço cartesiano, é possível dividi-lo entre sub-espaços que encobrem pontos SV com valores similares de S ou V, a fim de separá-los qualitativamente de acordo com sua sumarização. Neste sentido, definiu-se como **quantis** cada uma das divisões do espaço SV de acordo com delimitações estabelecidas. Por exemplo, ao dividir o espaço S em três secções, e o espaço V em duas secções, serão gerados 6 quantis no total (Figura 6), onde cada ponto terá uma classificação de acordo com seu presente quantil.

Embora essa definição seja análoga a definir um ponto de corte para cada eixo, os quantis possuem um papel diferente no contexto deste trabalho, onde representam mais uma área aproximada e ajustável do que um limiar fixo. Além disso, cada ponto definido para um quantil possui uma pontuação, denominada ***score* de quantil**, o qual é definido

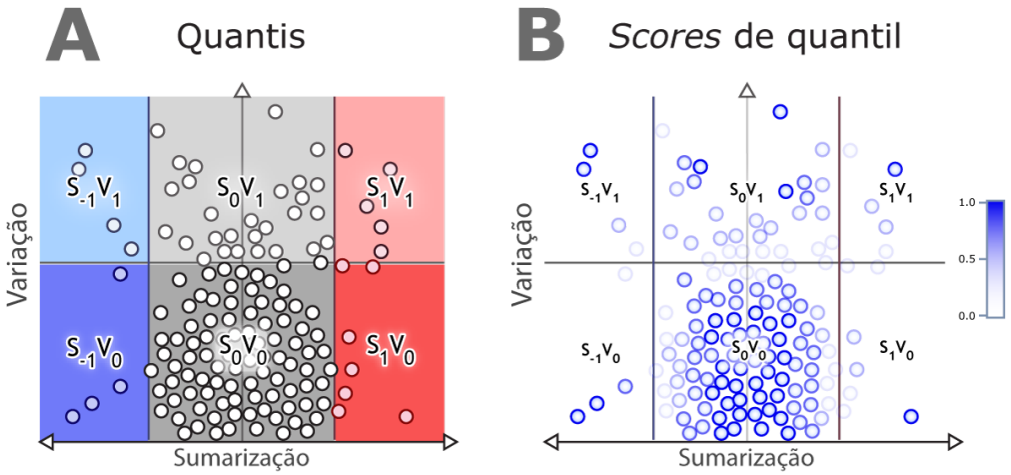

Figura 6 – Separação de quantis em um *SV-plot*. (A) Destaque das áreas dos quantis mediante a cor (azul para sub-expresso, cinza para pouca expressão diferencial e vermelho para super-expresso) e opacidade (opaco para pouco variado e transparente para muito variado). As legendas no centro de cada quantil apontam seu índice em forma de coordenadas (por exemplo, S_0V_1 para quantil central superior); (B) *Scores* de quantil pontuados de 0 a 1, calculados baseando-se na probabilidade de inclusão dos pontos em seus atuais quantis. Cada ponto foi renderizado com uma opacidade proporcional ao seu *score* calculado, contrastando os de maior pontuação e desvanecendo os de menor pontuação.

pela proximidade deste ponto ao centro do quantil. Esta pontuação é minimizada quando o ponto se encontra próximo de um ou mais quantis, e maximizada quando mais próxima do centro do quantil ou na extremidade, quando não há fronteira com outros quantis. Deste modo, é possível classificar os pontos SV utilizando os quantis como parâmetro qualitativo e o *score* de quantil como grau de inclusão ao quantil atribuído.

4.3.4 Alternativa ao ponto de corte – *Clusters*

Ao examinar o *SV-plot* da Figura 5C, torna-se perceptível o conglomerado de pontos sobre a região central inferior do gráfico, predominantemente sobre o quantil indicando valores de $logFC$ centrais próximos de zero e com pouca variação. Análises preliminares apresentaram essa mesma forma de concentração de pontos nos *SV-plots* de todas as observações testadas, sendo referenciado neste trabalho como **aglomerado indiferencial**. Embora a causa deste amontoado ocorrer ainda não tenha sido determinada, poderia-se supor que a necessidade do organismo de manter a maioria dos genes em um nível consistente, em prol da manutenção homeostática, possa estar relacionada a isso. O que se pode inferir, porém, é que os pontos deste aglomerado indiferencial são similares em termos de SV, em contraste com outros pontos destacados, os quais representariam genes que respondem mais intensamente ou variavelmente às condições experimentais.

Neste sentido, a distinção entre pontos que pertencem ou não ao aglomerado

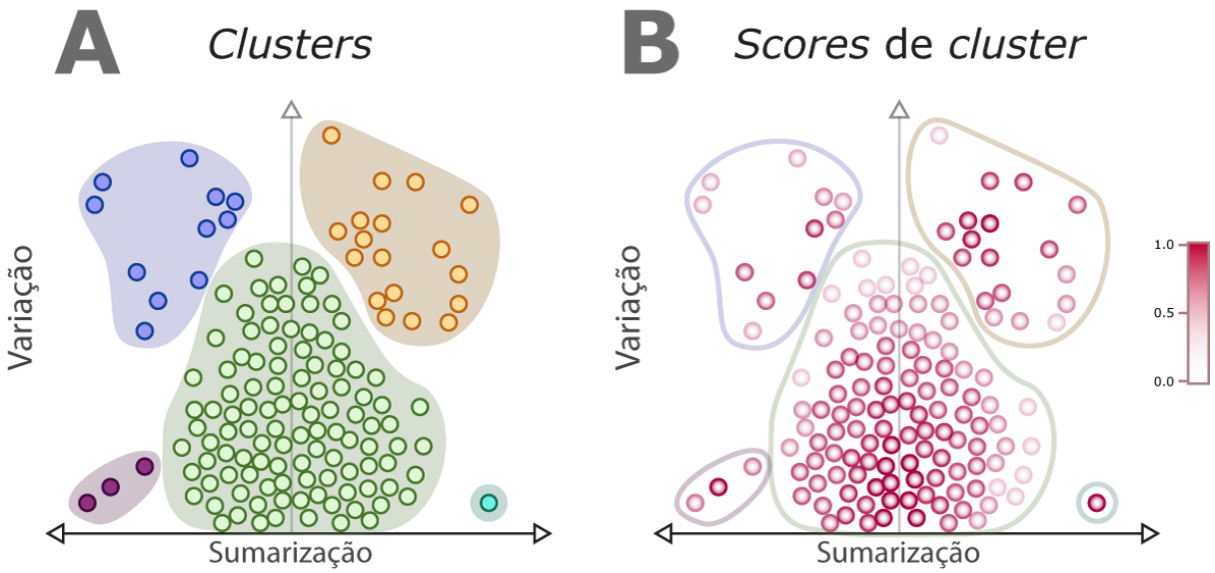

Figura 7 – Representação ilustrativa da detecção de *clusters* em um *SV-plot*. (A) Classificação dos pontos SV a seus respectivos *clusters*, estes distinguidos por cores e áreas hachuradas; e (B) *Scores* de *cluster* pontuados de 0 a 1, calculados baseando-se na probabilidade de inclusão de um ponto a seu *clusters* atribuído. Cada ponto foi renderizado com uma opacidade proporcional ao seu *score* calculado, contrastando os de maior pontuação e desvanecendo os de menor pontuação.

indiferencial pode ajudar a remover pontos que não estão respondendo à problemática, o que representaria uma separação mais flexível e adaptável ao espaço SV do que um ponto de corte fixo em cada eixo. Como discutido na Seção 2.3.3, uma análise de clusterização pode ser aplicada com o propósito de separar pontos em um espaço cartesiano, o que torna incluí-la neste contexto uma ideia atraente para auxiliar a classificação final.

4.3.5 Relação entre *clusters* e quantis

Uma clusterização seria aplicada para seccionar os pontos no espaço de uma forma diferente em relação aos quantis. Ao mesmo tempo, similarmente ao *score* para quantis, eles possuiriam uma pontuação associada ao pertencimento de cada ponto ao seu *cluster*, determinado como ***score* de *cluster***. Além disso, todo *cluster* estaria predominantemente engolfado por algum dos quantis definidos, baseado no quantil onde seu centroide está posicionado, que pode ser definido como seu **quantil de *cluster***.

Considerando essa lógica, este trabalho também propõe um método de **redefinição do quantil** atribuído a um ponto ao comparar seu *score* de quantil com seu *score* de *cluster*: se o *score* do quantil for maior, o ponto continua a pertencer ao quantil atribuído; enquanto que se o *score* de *cluster* for maior, o ponto passa a pertencer ao quantil atribuído ao *cluster*. Isso resultará em **quantis ajustados** (Figura 8), onde a delimitação dos quantis será acomodada às medidas de clusterização e, portanto, às relações SV entre os $logFC$

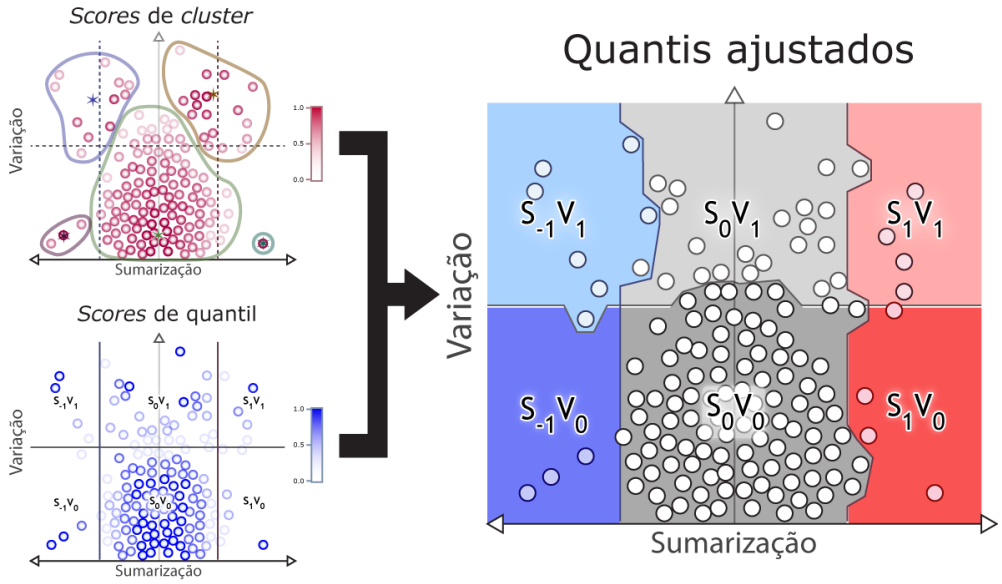

Figura 8 – Ajuste de quantis pela comparação entre *scores* de *clusters* e de quantis. Símbolos de estrela (★) representam o centroide de um *cluster*, sendo que o quantil em que está posicionado é definido como o quantil de *cluster*. Pontos SV contendo uma pontuação maior para seu *cluster* passam pertencer ao quantil do *cluster* em que foi classificado. Áreas hachuradas distorcidas indicam a mudança desses pontos para os novos quantis, figurativamente alterando o aspecto do quantil original.

sendo analisados. Visto que um quantil é utilizado para atribuir, qualitativamente, a relevância de ponto SV à problemática atual, o quantil ajustado também poderá redefinir as classificações finais de relevância, desta vez levando em conta os aglomerados de pontos SV fortemente associados entre si. Similarmente à solução das sumarizações ponderadas, essa proposta apresentada, apesar de não convencional, tem como finalidade aumentar a capacidade do algoritmo de melhor distinguir e classificar os resultados mais relevantes.

Por fim, esta seção apresentou soluções adicionais ao método manual visto na seção anterior que, ao serem levadas em conta durante a implementação do pacote, possivelmente enriqueceriam as análises para gerar resultados mais precisos. Todos os procedimentos apresentados neste capítulo foram adaptados e concatenados em um único pacote com o intuito de reproduzir o mesmo método para qualquer conjunto de resultados de expressão diferencial.

5 Materiais e Métodos

5.1 Módulos essenciais de desenvolvimento

O pacote foi desenvolvido na linguagem R (IHAKA; GENTLEMAN, 1996) com o auxílio do ambiente de programação RStudio (ALLAIRE, 2012), este o qual disponibiliza ferramentas visuais para melhor produtividade da construção de um pacote. Para compilar o pacote, fez-se necessário utilizar o complemento *RTools* para Windows, disponível pelo site oficial do R[1]. A compilação local também foi promovida pelo pacote *devtools* (WICKHAM; CHANG, 2016), que permitiu testar o pacote localmente sem a necessidade de instalação. Para geração de documentação descritiva, utilizou-se o pacote *roxygen2* (WICKHAM; DANENBERG; EUGSTER, 2017), o qual possui um sistema de marcação própria que agiliza a produção de documentos em PDF e HTML durante a compilação das classes e funções para distribuição final. Para a otimização pontual de certas funções, uma breve parcela do código foi desenvolvida na linguagem C++ usando o pacote *Rcpp* (EDDELBUETTEL et al., 2011), o qual realiza a compilação de funções feitas em C++ e as adapta para execução em R. Vale ressaltar que, com exceção do próprio R, nenhum dos módulos indicados neste parágrafo são requisitados para o usuário final, tendo sido adotados essencialmente para o desenvolvimento do GEVA.

5.2 Pacotes dependentes e auxiliares

Como mencionado na seção de fundamentos, um dos pontos fortes da linguagem R é a imensa quantidade de pacotes disponíveis para aplicação imediata, o que dispensa, em muitas ocasiões, a recriação de soluções que já foram desenvolvidas e otimizadas anteriormente. Da mesma forma, o desenvolvimento do GEVA fez uso dessa vantagem importando funções de pacotes externos e incorporando-as entre as próprias funções, o que também torna esses módulos parte das dependências do pacote. Em certos casos, também utilizou-se pacotes que auxiliam o acesso e experiência do usuário à análise, apesar de não fazerem parte das dependências para a funcionalidade básica do GEVA. Nas seções a seguir, serão listados os pacotes **dependentes** – *i.e.*, aqueles que serão instalados precedentemente ao pacote GEVA – e **auxiliares** – *i.e.*, não-essenciais para seu funcionamento, mas recomendados em certas etapas –, juntamente com a justificativa de cada aplicação.

[1] Acessado em: <https://cran.r-project.org/bin/windows/>

5.2.1 Pacotes dependentes

- *matrixStats*
- *fastcluster*
- *dbscan*
- *sp*
- *SearchTrees*

As funções básicas de sumarização (média e mediana) e variação (desvio-padrão, variância e MAD), bem como suas variantes ponderadas, foram importadas do pacote *matrixStats* (BENGTSSON et al., 2018). Embora os pacotes nativos do R já ofereçam as funções mais básicas para o mesmo fim, esse pacote foi necessário para incluir pesos nos cálculos de sumarização e variação com o intuito de aprimorar a robustez estatística. As únicas exceções foram as funções `weightedVar` (variância ponderada) e `weightedSd` (desvio-padrão ponderado), que não foram importadas de *matrixStats* devido a sua incapacidade de lidar com certos valores de peso até a versão mais recente do pacote. No lugar destas, foram implementadas as funções internas `weighted.var` e `weighted.sd`.

Para clusterização hierárquica, foi utilizado o pacote *fastcluster* (MÜLLNER et al., 2013), o qual disponibiliza uma implementação heurística para este método e torna possível sua aplicação em grandes conjuntos de dados. Embora os pacotes nativos do R também disponibilizem algoritmos de clusterização hierárquica, testes preliminares demonstraram que torna-se computacionalmente inviável utilizar tais funções para matrizes com dezenas de milhares de linhas (sondas, sequências ou genes), assim justificando a aplicação heurística do *fastcluster*.

Para análises de clusterização por densidade, foi utilizado o pacote *dbscan*, o qual disponibiliza os algoritmos DBSCAN e OPTICS para este tipo de agrupamento (HAHSLER; PIEKENBROCK; DORAN, 2019). Em adição, foram desenvolvidas funções para adaptar a clusterização por densidade para os tipos de dados do GEVA e a problemática atual, onde se fez necessário o uso dos pacotes auxiliares *sp* (PEBESMA; BIVAND, 2005) e *SearchTrees* (BECKER, 2016). Estes dois pacotes permitem a criação de uma estrutura de árvore para distribuir os pontos, com a finalidade de reduzir a quantidade de testes de distância entre pontos e tornar possível a análise com grandes volumes de dados.

5.2.2 Pacotes auxiliares

- *topGO*
- *limma*

Para ontologias gênicas, foram utilizados os testes estatísticos do pacote *topGO* (ALEXA; RAHNENFÜHRER, 2009). Estes testes podem ser aplicados sobre os resultados mais relevantes encontrados pelo GEVA, a fim de se prospectar associações significantes com processos biológicos descritos para esses alvos.

Considerando que os dados de *input* serão resultados de comparação por expressão diferencial, algumas funções foram adaptadas para receber tipos de dados provenientes do pacote *limma* (RITCHIE et al., 2015), que até o momento, permanece sendo um dos mais utilizados para análises comparativas com microarranjos e perfis de metilação. Esse pacote também inclui suporte para resultados com *RNA-seq*.

5.3 Definições de classes

Em todas as etapas de análise com o GEVA, serão retornados objetos com tipos definidos pelo próprio pacote. Estes objetos pertencem à família de objetos genéricos chamados **classes S4**, que tem como finalidade armazenar dados de uma forma organizada e pré-estabelecida. Essas classes foram adotadas com o propósito de manter a consistência dos dados e a assertividade dos tipos de argumentos das funções por meio de regras, e ao mesmo tempo manter um nível razoável de compatibilidade. Como comparativo, existem também as *classes S3*, que são classes formadas por listas com pouca capacidade de definição, e *classes de referência*, que organizam os dados por referência sem copiá-los por inteiro, mas que ainda possuem um suporte menor por terem sido implementadas mais recentemente.

Em virtude da adoção de classes S4, a maior parcela das funções nas etapas pós-*input* receberá e produzirá objetos de tipos definidos, que poderão ser acessados por meio de funções como, por exemplo, `plot` para geração de gráficos, `print` para imprimir a descrição de seu conteúdo, ou `head` para visualizar apenas as primeiras linhas da tabela principal, caso isso seja aplicável ao objeto. A lista completa de funções disponíveis para cada classe S4 encontra-se no capítulo seguinte, Seção 6.1.2 – *Classes e funções*.

Parte III

Resultados e Discussão

6 Pacote desenvolvido

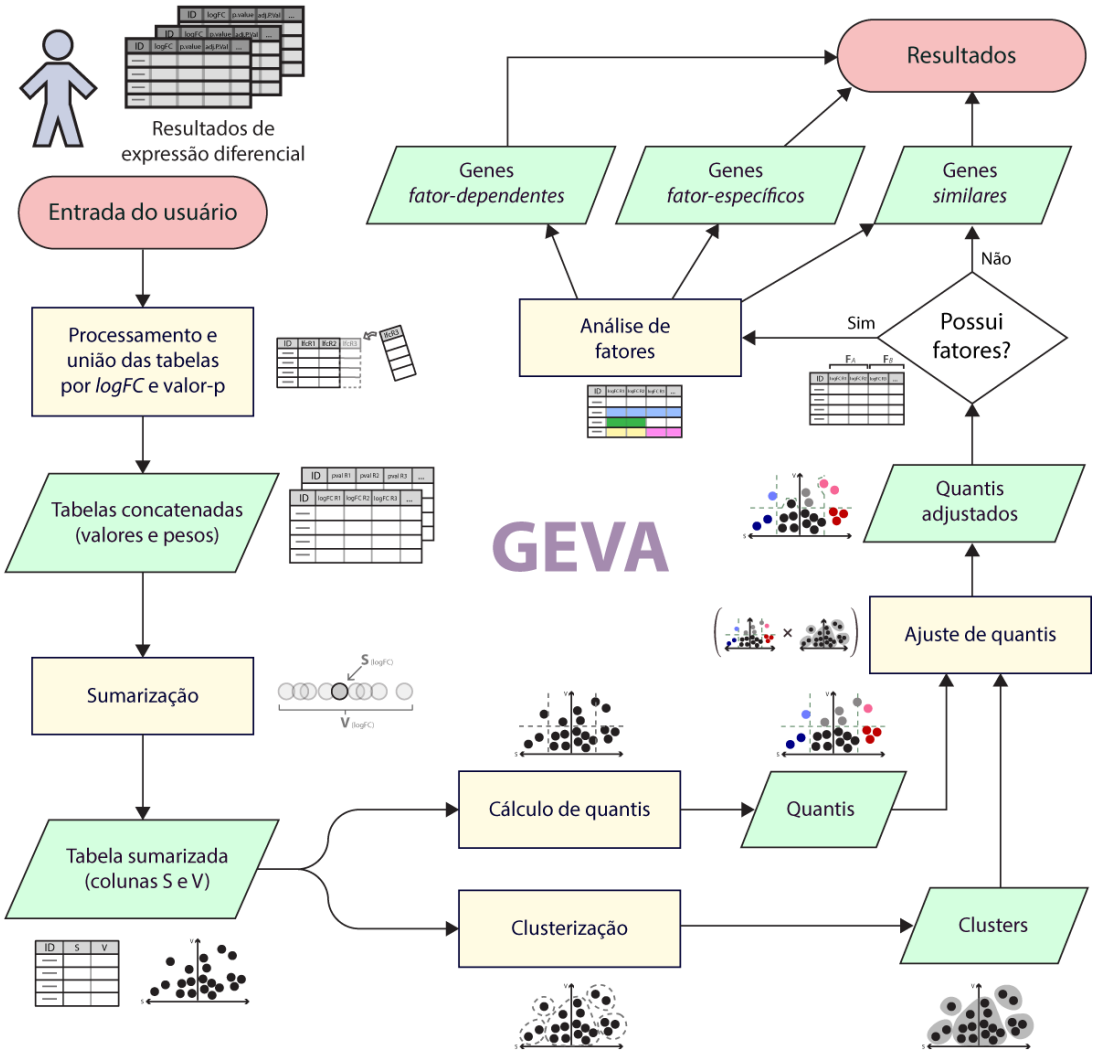

Figura 9 – Visão geral dos métodos aplicados com o pacote desenvolvido. Retângulos em amarelo indicam procedimentos, e trapézios em verde representam os dados obtidos após cada procedimento.

6.1 Especificações técnicas

6.1.1 Descrição geral

Nome do projeto: GEVA

Nome do pacote do R: `geva`

Código-fonte: <https://github.com/nunesijg/geva>

Sistema operacional: Multiplataforma

Licença: GNU LGPL

6.1.2 Classes e funções

O uso do pacote desenvolvido, seguido de sua instalação e importação, consistirá na efetivação dos passos apresentados no Capítulo 4 – Sistema Proposto por meio do uso de funções, as quais retornarão dados armazenados em objetos de classes definidas. De modo geral, as funções podem ser aplicadas linearmente seguindo o sistema proposto.

Abaixo, estão listadas as principais funções aplicadas na utilização deste pacote:

- ▷ `geva.merge.input`: recebe um conjunto de objetos de tabela (*e.g.*, matriz ou `data.frame`), extraindo e unificando suas colunas de $logFC$ e valor-p. Retorna um objeto `GEVAInput`, que armazena uma tabela concatenada para $logFC$ e outra para valor-p (se aplicável);
- ▷ `geva.read.tables`: lê um conjunto de arquivos de tabelas de entrada e concatena colunas de $logFC$ e valor-p. Retorna um objeto `GEVAInput`, que armazena uma tabela concatenada para $logFC$ e outra para valor-p (se aplicável);
- ▷ `geva.input.correct`: remove colunas e linhas inválidas das tabelas concatenadas;
- ▷ `geva.input.rename.rows`: (opcional) renomeia os títulos das linhas das tabelas, removendo duplicatas se necessário. Esta função é particularmente útil para substituir sondas (microarranjos) ou códigos de transcritos (RNA-seq) pelos nomes de genes;
- ▷ `geva.summarize`: realiza a sumarização dos valores das tabelas concatenadas, retornado um objeto `GEVASummary`. Este objeto representa uma tabela de duas colunas, S e V, com os resultados de sumarização e variação. A função `plot` no objeto retornado produz um gráfico *SV-plot*;
- ▷ `geva.quantiles`: define os quantis iniciais para uma tabela SV. Os limiares iniciais podem ser especificados, sendo o padrão a divisão proporcional do espaço em cada eixo. Retorna os resultados armazenados em um objeto `GEVAQuantiles`;
- ▷ `geva.cluster`: aplica uma análise de clusterização sobre uma tabela SV. Deve-se especificar o nome do método e seus parâmetros. Retorna os resultados armazenados em um objeto `GEVACluster`;
- ▷ `geva.hcluster`: função de atalho para clusterização hierárquica, equivalente a invocar a função `geva.cluster` usando o método `"hierarquical"` como parâmetro;
- ▷ `geva.finalize`: reúne informações de sumarização, quantis e clusterização para atribuir as classificações finais. Se fatores forem especificados, inclui uma etapa de análise para encontrar variações fator-dependentes e fator-específicas. Retorna um objeto `GEVAResults` representando a tabela com as classificações finais.

A lista completa de classes e funções, a qual inclui mais detalhes técnicos de uso, encontra-se no Apêndice A (classes em A.1 e funções em A.2). O diagrama UML (do inglês *Unified Modeling Language*) na Figura 10 oferece um panorama das classes, incluindo seus campos, métodos genéricos e relações de herança implementados.

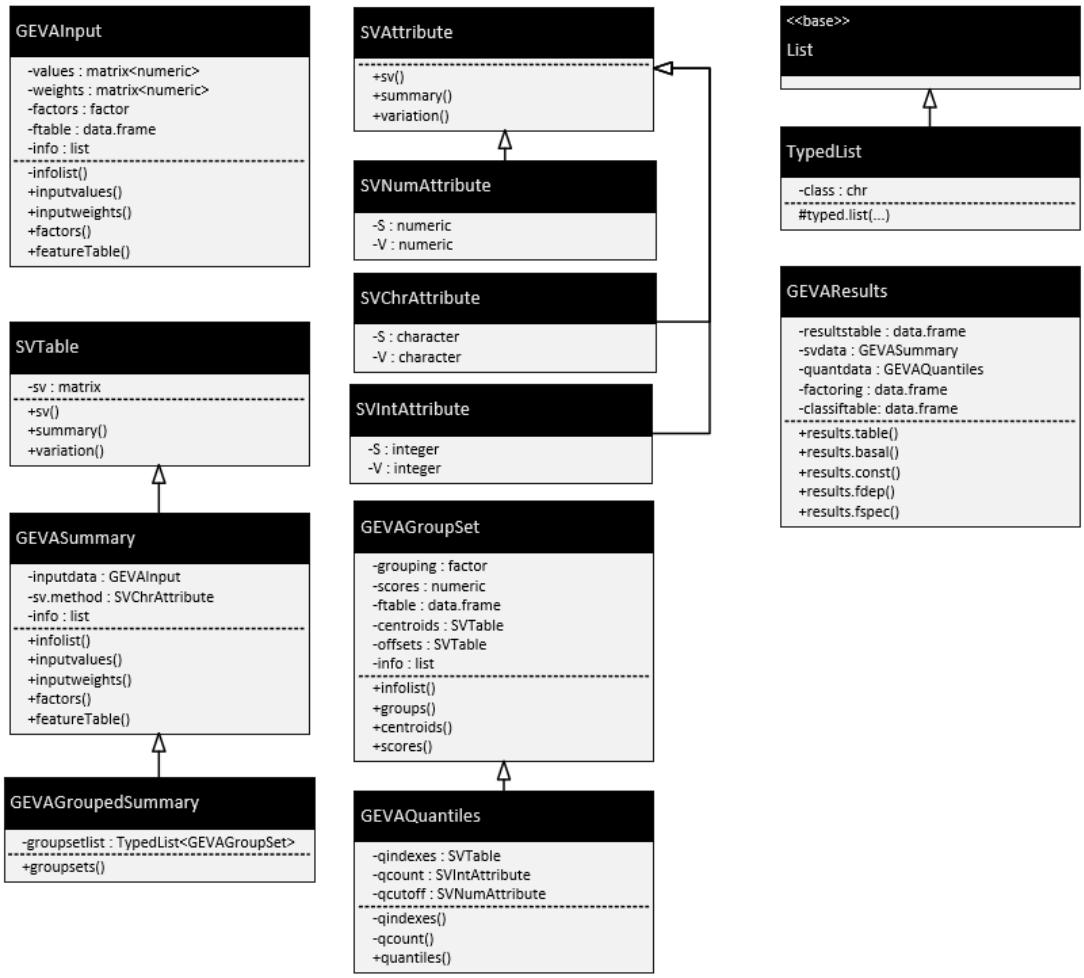

Figura 10 – Diagrama UML representando os tipos de classe S4 definidos no GEVA. Setas brancas indicam **herança**, onde uma classe deriva de uma classe base, obtendo seus campos e funções. A classe `List` é a única nativa do R neste diagrama, sendo incluída na figura apenas para indicar derivação da classe `TypedList`.

7 Aplicação prática do GEVA

Neste capítulo, serão abordadas as etapas de aplicação prática do pacote *geva*, desde sua importação até a obtenção dos resultados finais. Algumas destas etapas podem ser efetuadas por mais de uma maneira ou podem diferir conforme os parâmetros oferecidos pelo usuário, cabendo a este selecionar os critérios que melhor acomodam-se a sua problemática.

7.1 Pré-análise

Antes de efetuar qualquer tipo de cálculo ou aplicação analítica mais complexa com o GEVA, é necessário que o pacote esteja prontamente carregado e que os dados de entrada já tenham sido processados. A pré-análise consiste nestas etapas iniciais, tendo como finalidade a obtenção das tabelas concatenadas (valores de $logFC$ e pesos) que serão utilizadas como base para análises posteriores.

7.1.1 Instalação do pacote

O pacote está hospedado em um repositório do GitHub[1], e pode ser instalado pela entrada do seguinte comando:

```
# Instala o GEVA a partir do repositório GitHub
> install_github('nunesijg/geva')
```

O R irá fazer o *download* do código-fonte do repositório e instalar o pacote na biblioteca local do sistema. Com a instalação do pacote concluída, este pode finalmente ser carregado por meio do seguinte comando:

```
> library(geva)
```

7.1.2 Entrada de dados

Os dados de entrada, como mencionado no Capítulo 4, são múltiplos resultados de expressão diferencial constituídos por valores de $logFC$, preferencialmente acompanhados por valores-p. As análises de expressão diferencial possuem uma miríade de variáveis que diferem para cada tipo de experimento ou plataforma, o que não se encontra dentro do escopo deste trabalho. Ainda assim, para uma breve referência, disponibilizou-se um exemplo de análise para dados de microarranjo com R na seção B.2 do Apêndice B.

[1] Acessado em: <https://github.com/nunesijg/geva>

O GEVA oferece mais de uma alternativa para entrada dos dados, seja pela leitura de arquivos externos ou por parâmetros dentro do próprio R. A seguir, são apresentadas cada uma dessas alternativas de entrada de dados, onde o usuário precisará escolher apenas uma para seguir para os passos posteriores.

Alternativa 1 – Arquivos de texto

Muitos dos programas convencionais de análise de dados de larga escala, como de microarranjos, produzem uma tabela de resultados de expressão diferencial que pode ser exportada para um arquivo de texto. O arquivo salvo consistirá, neste caso, em uma tabela onde colunas são separadas por caracteres de tabulação (`Tab`) e linhas pelo próprio delimitador de nova linha. Para carregar as tabelas no R a partir dos arquivos exportados, pode-se usar a função `geva.read.tables`, como demonstrado abaixo:

```
# Substituir os valores abaixo por nomes de arquivos
# no diretório atual
> nomesarqs <- c("brca1_2h.txt", "brca1_4h.txt", "brca2_2h.txt",
                 "brca2_4h.txt", "brca3_2h.txt", "brca3_4h.txt")
> ginput <- geva.read.tables(filenames=nomesarqs)
```

Essa função possui diversos parâmetros opcionais. Se o parâmetro `filenames` não for especificado, todos os arquivos com extensão `.txt` (ou como estiver indicado no argumento `files.pattern`) serão buscados no diretório indicado em `dirname` (por padrão, o diretório de `getwd()`), fazendo a leitura de cada arquivo como tabela. Todas as tabelas devem estar corretamente formatadas – caso contrário, um erro de leitura será emitido. Além disso, cada tabela terá que incluir uma coluna com o nome indicado no argumento `col.values` (por padrão, `"logFC"`) para valores de $logFC$ e, opcionalmente, uma coluna com o nome indicado em `col.pvals` (por padrão, `"adj.P.Val"`) para valores-p, que serão convertidos em pesos durante a leitura.

Alternativa 2 – Múltiplos objetos de tabela

Para objetos previamente processados, como matrizes e `data.frame`, pode-se utilizar suas colunas correspondentes (e.g., `logFC` e `adj.P.Val`) para formar a tabela de entrada. A função `geva.merge.input` facilita a extração e concatenação entre múltiplas tabelas fornecidas pelo usuário, como demonstrado abaixo:

```
# dt1 e dt2 são data.frame de entrada, cada um contendo uma
# coluna logFC e adj.P.Val
> ginput <- geva.merge.input(dt1, dt2)
```

Alternativa 3 – Resultados do *limma*

Caso o usuário esteja utilizando um pacote para análise de microarranjo, como o *limma*, é possível fornecer a tabela de resultados diretamente para a função geva.merge.input. Para isso, partindo-se do pressuposto que o usuário possui o pacote *limma* instalado, utiliza-se sua função `topTable` para extração do `data.frame` de um objeto `MArrayLM`, oferecendo seu resultado como argumento de entrada para o GEVA, como indicado abaixo:

```
# de1 e de2 são objetos do tipo MArrayLM,
# produzidos pelo limma (e.g., por eBayes)
> dt1 <- topTable(de1, number=1000000, sort.by="none")
> dt2 <- topTable(de2, number=1000000, sort.by="none")
> ginput <- geva.merge.input(dt1, dt2)
```

Alternativa 4 – Dados ideais (somente para testes)

Seja pela ausência de dados experimentais ou por razões didáticas, pode haver situações onde necessita-se testar o pacote de imediato sem o provimento de dados reais, visto que isto significaria o encargo de efetuar uma série de análises de expressão diferencial em precedência. Para isso, pode-se utilizar a função geva.ideal.example, como indicado abaixo:

```
# Gera um GEVAInput aleatório com 10000 sondas, 6 colunas
> ginput <- geva.ideal.example()
```

As tabelas contidas no objeto terão valores essencialmente aleatórios, porém simulando uma distribuição que seria observada em dados reais, incluindo variações fator-dependente e fator-específicas. Os parâmetros opcionais desta função estão listados no Apêndice A (ver geva.ideal.example).

7.1.3 Correção dos dados de entrada (Opcional)

É importante levar em conta que todas as etapas posteriores dependerão dos valores das tabelas concatenadas nessa etapa inicial. Valores irregulares como `NA` (valor não encontrado) ou `Inf` (número infinito) podem enviesar os cálculos numéricos ou até mesmo resultar em erros de processamento, sendo importante removê-los. Além disso, se os valores de peso foram obtidos através de valores-p, também é aconselhável remover linhas onde os todos os valores-p seriam considerados estatisticamente insignificantes. Ajustes como esses podem ser efetuados por meio das funções do GEVA que serão apresentadas nessa seção.

Removendo valores inválidos

O comando abaixo utiliza a função `geva.input.correct` para remover todas as linhas contendo valores `NA` ou infinitos:

```
# Remove linhas contendo valores NA e infinitos
> ginput <- geva.input.correct(ginput)
```

A validação é feita tanto para tabela de valores quanto para de pesos, e o filtro é aplicado para ambas as tabelas caso valores inválidos tenham sido encontrados em qualquer uma delas, mantendo sempre o mesmo número de linhas em todos os campos `matrix` ou `data.frame` do objeto `GEVAInput`. Essa função também leva em conta a possibilidade de uma coluna inteira ser composta por valores inválidos, o que faria com que todas as linhas fossem deletadas pelos parâmetros do filtro. Para prevenir isso, são deletadas todas as colunas onde há somente valores `NA` e infinitos antes de aplicar o filtro às linhas. O resultado final é o mesmo objeto, porém sem linhas ou colunas inválidas.

Definindo nome das linhas

Em muitas técnicas experimentais de larga escala, como microarranjos, a expressão diferencial não é aplicada no contexto de genes, mas de sondas associadas a um ou mais genes. Embora seja viável utilizar sondas como referência para as tabelas concatenadas, o GEVA também oferece a opção de redefinir os títulos das linhas para outro nome representativo, como por exemplo, os nomes dos genes. Isso pode ser efetuado através da função `geva.input.rename.rows`, que recebe um vetor `character` com os nomes substituintes. Caso o `GEVAInput` de entrada também contenha colunas adicionais, pode-se também usar um `character` único como o argumento indicando o nome da coluna que substituirá os nomes de linha. O comando abaixo exemplifica a aplicação dessa função:

```
# Define os nomes das linhas como "Gene_Symbol",
# ignorando nomes vazios e duplicatas
> ginput <- geva.input.rename.rows(ginput,
                                   attr.column="Gene_Symbol",
                                   dupl.rm.method="order")
```

Essa função também pode ser conveniente para lidar com duplicatas, pois oferece parâmetros de como e quais valores repetidos serão deletados. Por exemplo, utilizando o argumento `dupl.rm.method="least.p.vals"`, a função irá remover valores duplicados de forma que o valor mantido será aquele com menor valor-p, deste modo priorizando o $logFC$ mais significante. O parâmetro `dupl.rm.method="order"`, por outro lado, remove os valores duplicados de acordo com sua ordem na tabela.

7.2 Análises SV

Com a pré-análise concluída, obteve-se um objeto `GEVAInput` contendo as tabelas de valores de $logFC$ e de pesos. O próximo passo é calcular a sumarização de variação a partir destes dados e efetuar análises para extrair informações dos pontos SV, as quais servirão de base para a classificação final.

7.2.1 Sumarização

A função abaixo irá aplicar uma sumarização com as tabelas de um `GEVAInput` usando os parâmetros `"median"` (mediana) e `"mad"` (MAD):

```
# Efetua a sumarização de ginput (objeto GEVAInput)
> gsummary <- geva.summarize(ginput, 'median', 'mad')
```

Se os valores de peso foram definidos na tabela de pesos do `GEVAInput` de entrada, os cálculos de sumarização e variação serão ponderados (neste exemplo, mediana e MAD ponderados). Essa função retornará um `GEVASummary`, que contém uma tabela composta pelas colunas `S` e `V`.

Para visualizar os pontos SV de um `GEVASummary`, pode-se aplicar a função `plot` sobre este objeto:

```
# Desenha um SV-plot dos dados sumarizados
> plot(gsummary)
```

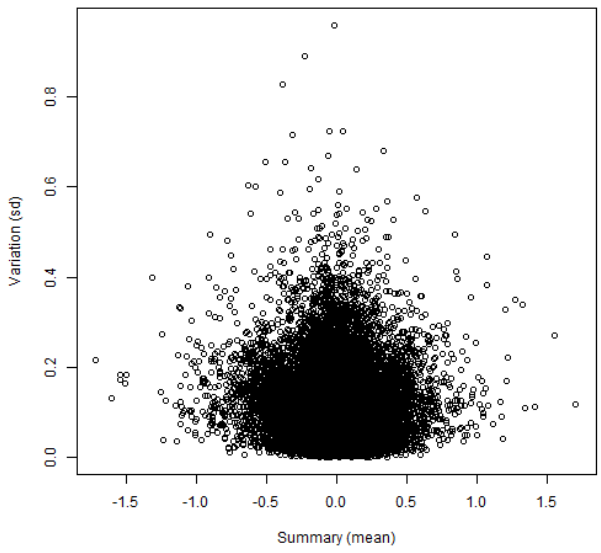

Figura 11 – Gráfico SV representando um objeto `GEVASummary` produzido pela função geva.summarize.

O comando acima produzirá um gráfico tipo *SV-plot*, como ilustrado na Figura 11.

7.2.2 Cálculo dos quantis

Com a obtenção do `GEVASummary`, o próximo passo consistirá no cálculo dos quantis para todos os pontos SV. A função responsável por esse cálculo é a `geva.quantiles`, podendo ser aplicada como no exemplo abaixo:

```
# Calcula os quantis de um objeto GEVASummary (gsummary)
> gquant <- geva.quantiles(gsummary)
```

O comando acima irá produzir um objeto `GEVAQuantiles` contendo os resultados dos quantis calculados. Neste exemplo, como os limiares iniciais não foram especificados, a separação dos quantis será feita pelo método padrão (`"range.slice"`), baseado na distribuição dos valores de cada eixo. Seus limiares calculados podem ser visualizados ao gerar um gráfico pela função `plot` tendo o objeto (Figura 12) como argumento, como demonstrado abaixo:

```
# Desenha um SV-plot incluindo os quantis
> plot(gquant)
```

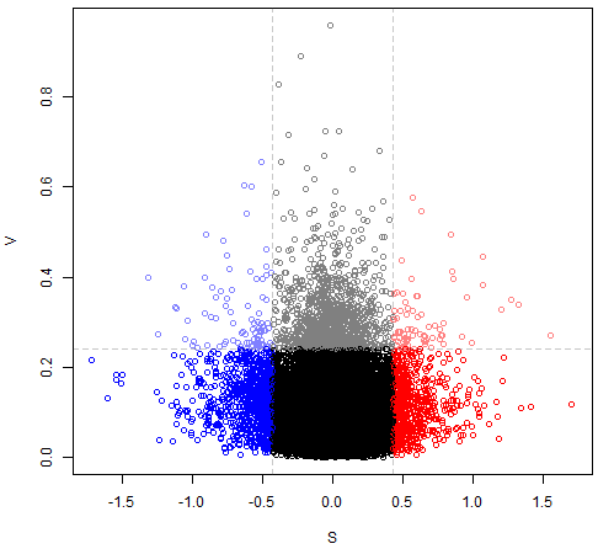

Figura 12 – Gráfico SV representando um objeto `GEVAQuantiles` produzido pela função `geva.quantiles`.

Por outro lado, é possível também determinar os limiares dos eixos `S` e `V`, que serão o ponto de separação de quantil mais próximo do valor zero para estes eixos. O exemplo abaixo demonstra um caso onde é feita essa especificação (valor `1` para sumarização e `0.5` para variação):

```
# Calcula os quantis de um objeto GEVASummary (gsummary)
> gquant <- geva.quantiles(gsummary,
                           initial.thresholds=c(S=1, V=0.5))
```

A Figura 13 ilustra os limiares dos quantis quando essa opção é indicada pelo usuário. Definir os limiares, principalmente o de `S`, pode vir a ser útil quando os parâmetros padrões incluem muito mais ou muito menos pontos pertencentes ao aglomerado indiferencial. Novamente, é importante ressaltar que o propósito dos quantis não é de um ponto de corte, mas de uma área aproximada de definição qualitativa. O quantil só se comportará como ponto de corte caso nenhuma medida para correção de quantis for incluída na análise, como a clusterização.

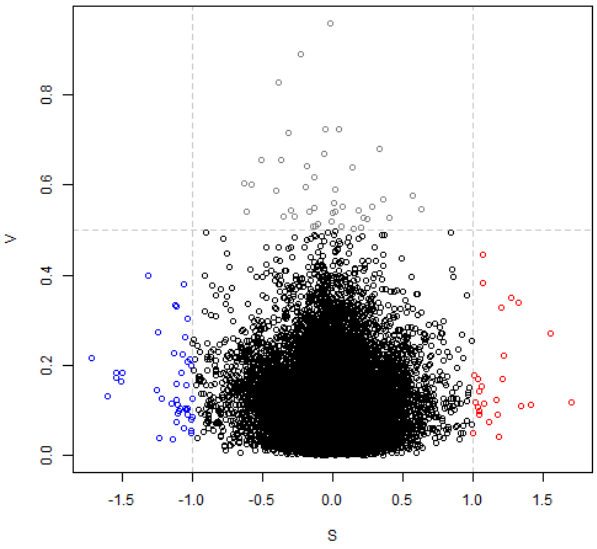

Figura 13 – Gráfico SV representando um objeto `GEVAQuantiles` produzido pela função `geva.quantiles` com parâmetros de corte especificados para separação de quantis (`initial.thresholds=c(S=1, V=0.5)`).

7.2.3 Clusterização

Nesta etapa, será feita a clusterização dos pontos SV, a qual permitirá a distinção entre os pontos relevantes e os pontos do aglomerado indiferencial. A função `geva.cluster` realiza essa etapa dependendo do método de clusterização especificado no argumento `cluster.method`, que pode ser `"hierarchical"` (hierárquica) ou `"density"` (por densidade). Ambos os métodos recebem um argumento `resolution` (resolução), que indica a proporção relativa de *clusters* a serem produzidos – se o valor for `0`, retorna um único *cluster*, enquanto que se o valor for `1.00`, retorna o máximo de *clusters* possíveis. O exemplo abaixo apresenta um caso onde a opção de clusterização hierárquica com resolução `0.3` é feita sobre um objeto `GEVASummary`:

```
> gcluster <- geva.cluster(gsummary,
                           cluster.method='hierarchical',
                           resolution=0.3)
```

Um objeto `GEVACluster` é retornado, contendo os resultados da clusterização efetuada. Alternativamente, para ter-se um maior controle sobre o método, é possível também utilizar as funções específicas de cada método de clusterização, as quais permitem inserir parâmetros mais específicos para o método a ser considerado. Para isso, os tópicos a seguir apresentam casos de uso para os métodos de clusterização possíveis:

Alternativa 1 – Clusterização hierárquica

Por meio da função `geva.hcluster`, uma análise de clusterização é efetuada a partir da hierarquia de distância entre pontos, como explicado na subseção 2.3.3, tópico *Clusterização Hierárquica*. O método de pontuação dos *clusters* também é medido pela hierarquia relativa dos pontos de cada *cluster*. Contudo, há também dois parâmetros que podem ser indicados para o algoritmo de clusterização: `hc.method` para seleção de hierarquia e `hc.metric` para métrica de distância entre hierarquias. No exemplo abaixo, o algoritmo selecionará a hierarquia baseado no centroide de cada nível utilizando distância euclidiana:

```
# Efetua uma clusterização hierárquica sobre os dados SV
> gcluster <- geva.hcluster(gsummary, resolution=0.3,
                            hc.method='centroid',
                            hc.metric='euclidean')
```

A lista completa de opções pode ser encontrada na seção `geva.hcluster` do Apêndice A. Com os resultados de clusterização obtidos, pode-se visualizar graficamente os *clusters* pelo comando `plot` utilizando o objeto `GEVACluster` retornado. Esta função também pode ser chamada em conjunto com um `GEVAQuantiles` do passo anterior como segundo argumento, a fim de incluir a demarcação dos quantis no gráfico (Figura 14). Abaixo, há exemplos de chamada de `plot` para ambos os casos:

```
# Desenha um SV-plot incluindo a clusterização
> plot(gcluster)
# Desenha um SV-plot incluindo a clusterização com
# demarcação dos quantis
> plot(gcluster, gquant)
```

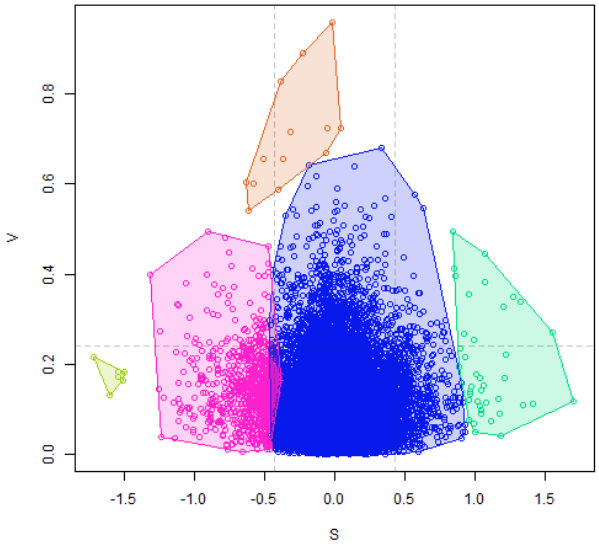

Figura 14 – Gráfico SV representando um objeto `GEVACluster` produzido pela função `geva.hcluster` (clusterização hierárquica) com resolução de 30% (`resolution=0.3`). O objeto `GEVAQuantiles` da etapa anterior foi utilizado como segundo argumento da função `plot` para destacar a separação dos quantis pelas linhas tracejadas. Os *clusters* são indicados especificamente pelas cores dos pontos (escolhidas aleatoriamente), sendo os polígonos inclusos apenas para auxílio visual.

Alternativa 2 – Clusterização por densidade

A função `geva.dcluster` pode ser utilizada para efetuar uma análise de clusterização por densidade, onde os pontos com distância similar entre vizinhos são agrupados entre si, como explicado na subseção 2.3.3, tópico *Clusterização por Densidade*. Diferentemente da clusterização hierárquica, o método por densidade não necessariamente agrupa todos os pontos, e uma quantidade mínima de pontos por *cluster* pode ser especificada pelo argumento `minpts`. Pontos não-agrupados terão um *score* nulo e não serão levados em conta nas etapas posteriores, durante a correção de quantis. O exemplo abaixo demonstra o uso básico dessa função, onde pelo menos dois pontos são requisitados para formar um *cluster*:

```
# Efetua uma clusterização por densidade sobre os dados SV
> gcluster <- geva.dcluster(gsummary, resolution=0.3, minpts=2)
```

A representação gráfica dos *clusters* pode ser obtida da mesma maneira que a primeira alternativa de clusterização: chamando-se a função `plot` com o objeto `GEVACluster` sozinho ou acompanhado de um `GEVAQuantiles` para destacar os quantis (Figura 15). Portanto, pode-se utilizar qualquer um dos seguintes comandos:

```
# Desenha um SV-plot incluindo a clusterização
> plot(gcluster)
# Desenha um SV-plot incluindo a clusterização com
# demarcação dos quantis
> plot(gcluster, gquant)
```

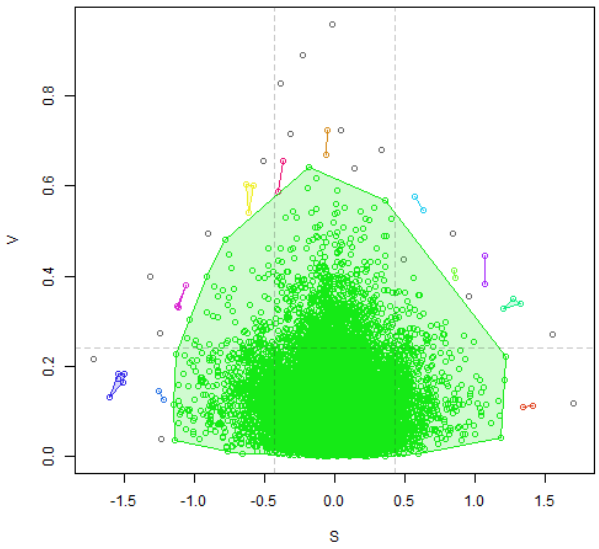

Figura 15 – Gráfico SV representando um objeto `GEVACluster` produzido pela função geva.dcluster (clusterização por densidade) com resolução de 30% (`resolution=0.3`). O objeto `GEVAQuantiles` da etapa anterior foi utilizado como segundo argumento da função `plot` para destacar a separação dos quantis pelas linhas tracejadas. Os *clusters* são indicados especificamente pelas cores dos pontos (escolhidas aleatoriamente), sendo os polígonos inclusos apenas para auxílio visual. Pontos em cinza não pertencem a nenhum *cluster*.

7.3 Obtenção dos resultados

Com as etapas de demarcação de quantis e clusterização concluídas para os dados SV, o próximo e último passo consistirá em unificar todos esses dados em um mesmo contexto, a fim de se estimar as classificações finais de cada gene.

7.3.1 Concatenação das informações obtidas e finalização

Para esta etapa final, é chamada a função geva.finalize, que recebe o objeto `GEVASummary` como argumento primário, além de objetos retornados em outras etapas de análise como argumentos secundários. Neste caso, os objetos `GEVAQuantiles` e `GEVACluster` serão adicionados como argumentos para representar definição e correção de quantis, respectivamente. Se os quantis não forem fornecidos, eles serão calculados

automaticamente durante essa etapa utilizando os parâmetros-padrão; enquanto que se os *clusters* não forem informados, não haverá correção dos quantis, o que limitará sua definição a apenas pontos de corte. Além disso, caso a análise esteja incluindo fatores (*i.e.*, grupos de condições experimentais) a serem comparados, tais fatores devem ser definidos antes da chamada da função `geva.finalize`. Os tópicos a seguir assinalam ambos os casos:

Alternativa 1 - Sem fatores envolvidos

Na ausência de fatores, a chamada da função pode ser feita diretamente. Contudo, somente a classificação *similar* será atribuída para resultados relevantes, não incluindo variações relacionadas a grupos de experimentos. O código abaixo demonstra o uso básico da função `geva.finalize` para este caso, apresentando os objetos `GEVASummary`, `GEVAQuantiles` e `GEVACluster` obtidos nas etapas anteriores como argumento, e retornando um objeto `GEVAResults` como resultado:

```
# Obtém as classificações finais a partir dos valores
# retornados pelas análises anteriores
> gresults <- geva.finalize(gsummary, gquant, gcluster)
```

Alternativa 2 - Com fatores envolvidos

Na presença de fatores a serem analisados, deve-se definir seus valores sobre o objeto `GEVAInput` na etapa inicial através da função `factors` junto ao operador de definição `<-` ou `=`. O valor a ser atribuído deve ser um objeto do tipo `factor` ou um vetor `character`, e deve ser do tamanho equivalente ao número de colunas do `GEVAInput` inicial. O código abaixo demonstra como isso seria feito sobre o `GEVAInput` antes da sumarização, supondo que a tabela inicial fosse composta por seis experimentos classificados entre "g1", "g2" ou "g3":

```
# Supondo que a tabela inicial (ginput) tenha 6 colunas:
# Exemplo com vetor caracteres
> factors(ginput) <- c('g1', 'g1', 'g2', 'g2', 'g3', 'g3')
# Exemplo com objeto tipo factor
> factors(ginput) <- factor(c('g1', 'g1', 'g2', 'g2', 'g3', 'g3'))
```

Contudo, caso os fatores não tenham sido informados antes da sumarização, também pode-se aplicar a função `factors` de instanciamento sobre o objeto `GEVASummary` principal, que irá efetuar essa atribuição sobre o `GEVAInput` que armazenara internamente. Como demonstrado no exemplo abaixo, utiliza-se a função da mesma forma para definir os fatores:

```
# Supondo que a tabela inicial (ginput) tenha 6 colunas e
# gsummary seja o resultado de sua sumarização:
# Exemplo com vetor caracteres
> factors(gsummary) <- c('g1', 'g1', 'g2', 'g2', 'g3', 'g3')
# Exemplo com objeto tipo factor
> factors(gsummary) <- factor(c('g1', 'g1', 'g2',
                                'g2', 'g3', 'g3'))
```

Deve-se levar em conta, porém, que os fatores serão utilizados particularmente nas análises de variação do tipo ANOVA, incluindo os métodos de Fisher e Levene discutidos na subseção 2.3.2. Embora dois fatores (*i.e.*, dois valores únicos) sejam o requisito mínimo para este tipo de análise, recomenda-se pelo menos duas ou mais ocorrências de cada grupo – como no exemplo acima – para atingir uma significância estatística confiável, pois a ANOVA utilizará as variâncias parciais de cada grupo para determinar a confiabilidade da classificação final. Além disso, um ponto de corte para o valor-p estimado em cada ANOVA pode ser especificado através do argumento `p.val.adjust`, sendo o padrão `0.05` ($\alpha < 0.05$). O código abaixo apresenta a utilização deste argumento na função `geva.finalize` após os fatores já terem sido definidos:

```
# Obtém as classificações finais a partir dos valores
# retornados pelas análises anteriores
> gresults <- geva.finalize(gsummary, gquant, gcluster,
                            p.val.adjust=0.05)
```

Assim como em outras etapas, é possível visualizar o resultado através de um gráfico aplicando-se a função `plot` sobre o objeto `GEVAResults`. O gráfico terá a mesma representação de quando aplicado aos quantis, porém com certos pontos redefinidos para quantis vizinhos em resposta ao ajuste de quantis (Figura 16). O código abaixo exemplifica este caso:

```
# Desenha um SV-plot com os quantis ajustados
> plot(gresults)
```

7.3.2 Consulta e extração dos resultados

O objeto `GEVAResults` representa a concatenação das informações utilizadas ao longo da análise com GEVA, além de resultados finais e parciais que podem ser consultados por meio de funções. As principais funções são: `results.table` para a tabela com a classificação final de todos os genes; `inputdata` para uma cópia do `GEVAInput` utilizado; `sv.data` para o objeto `GEVASummary` (ou `GEVAGroupedSummary`, que contém a lista de *clusters* utilizados); `quantiles` para o `GEVAQuantiles` (ou `GEVAQuantilesAdjusted` se

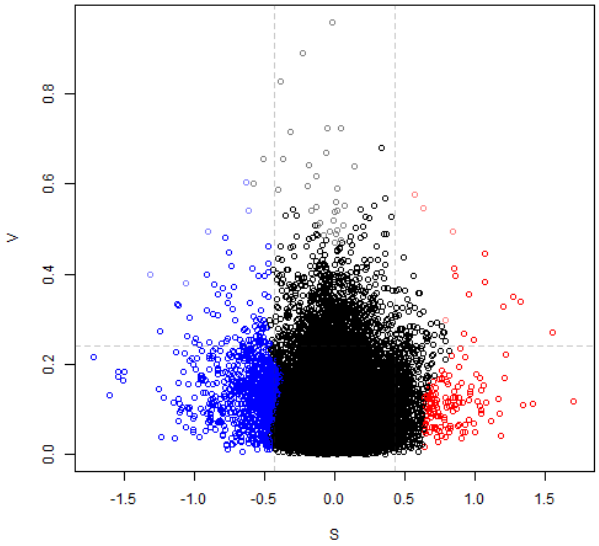

Figura 16 – Gráfico SV representando um objeto `GEVAResults` produzido pela função `geva.finalize`. A representação é a mesma para um objeto `GEVAQuantiles`, porém nota-se certos pontos redefinidos para o quantil correspondente ao seu *cluster*.

forem quantis ajustados); e, finalmente, `top.genes` para consultar os genes classificados como relevantes.

A função `top.genes` é a principal forma de consultar os resultados obtidos. Ela recebe como argumento um `GEVAResults` e retorna uma tabela contendo a classificação de cada gene na primeira coluna (Figura 17A), indicando uma dentre as seguintes possibilidades: `"similar"`, `"factor-dependent"`, `"factor-specific"`, `"basal"` (indiferencial) e `"sparse"` (irregular), nomeadas de acordo com as classificações discutidas na subseção 4.2.3. A segunda coluna indica um fator específico para genes que possuem esta classificação (Figura 17A). Outras colunas podem ser anexadas indicando seu nome no argumento opcional `add.cols` (*e.g.*, símbolos dos genes), desde que essas colunas estejam presentes no objeto `GEVAInput` inicial (Figura 17B).

O código abaixo demonstra o uso básico do `top.genes`:

```
# Extrai a tabela com os resultados mais relevantes
> dt.genes <- top.genes(gresults)

# Exemplo para inclusão de uma coluna denominada
# "gene_assignment", caso esta se encontre na tabela
# inicial (GEVAInput)
> dt.genes <- top.genes(gresults, add.cols="gene_assignment")
```

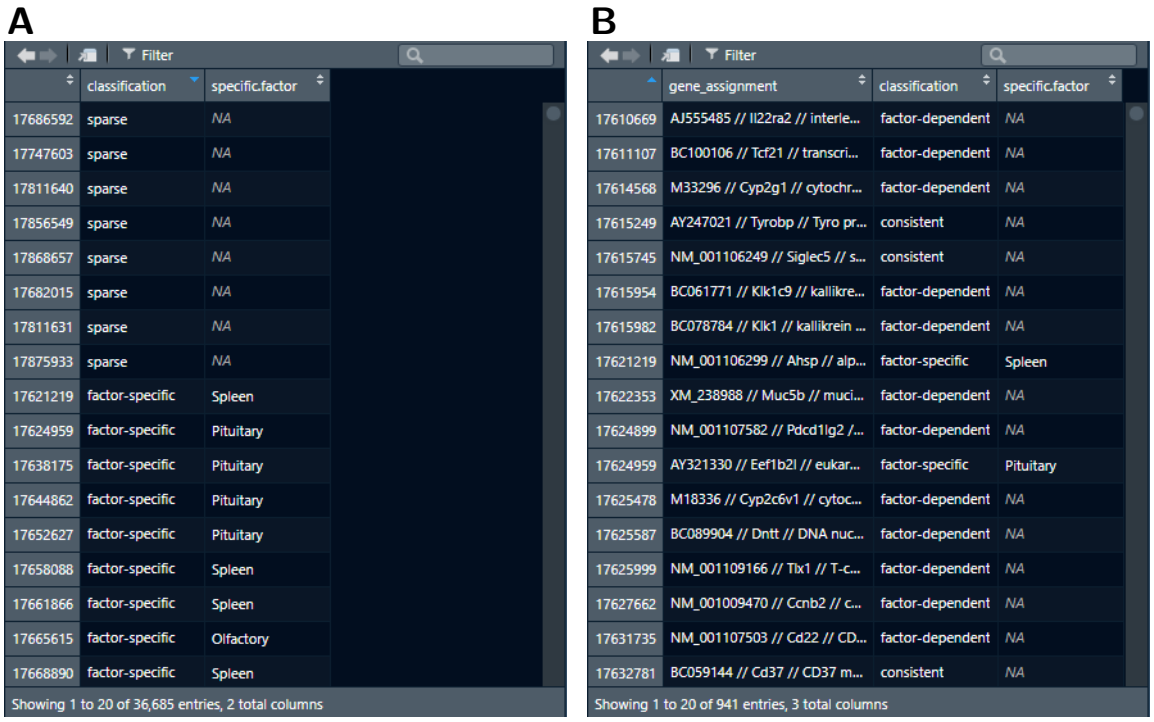

Figura 17 – Exemplo de visualização da tabela final de classificação utilizando o RStudio. Esse modo de navegação é exclusivo do programa, e pode ser acessado através da função `View`. (A) Visualização da tabela total pelo comando `View(results.table(gresults))`; e (B) visualização da tabela de resultados relevantes incluindo nomes dos genes pelo comando `View(top.genes(gresults, add.cols="gene_assignment"))`.

Deste modo, a tabela com os genes relevantes pode ser salva em um arquivo externo para outras finalidades de utilização. O comando abaixo exemplifica este caso de uso:

```
# Salva a tabela em um arquivo "resultados.txt"
> write.table(dt.genes, "resultados.txt", quote=F, sep="\n",
              row.names=T, col.names=T)
```

7.4 Resumo dos passos de análise

O *script* abaixo agrega todos os passos descritos neste capítulo:

```
# Define os nomes dos arquivos de tabela de entrada
> nomesarqs <- c("brca1_2h.txt", "brca1_4h.txt", "brca2_2h.txt",
                 "brca2_4h.txt", "brca3_2h.txt", "brca3_4h.txt")
# Lê os arquivos do disco local e processa as tabelas
> ginput <- geva.read.tables(filenames=nomesarqs,
                             attrcols='gene_assignment')
# Define os grupos de experimentos (fatores)
> facts <- c('brca1', 'brca2', 'brca3')[c(1, 1, 2, 2, 3, 3)]
# Atribui os fatores ao objeto de tabela concatenada
> factors(ginput) <- facts
# Sumariza a tabela concatenada, obtendo os pontos SV
> gsummary <- geva.summarize(ginput)
# Realiza uma análise de clusterização (hierárquica como padrão)
> gclust <- geva.cluster(gsummary)
# Calcula os quantis que separam o espaço SV qualitativamente
> gquant <- geva.quantiles(ginput)
# Unifica os resultados e obtém a classificação final
> gresults <- geva.finalize(ginput, gclust, gquant)
# Extrai a tabela final com os genes relevantes
> dt.genes <- top.table(gresults, attrcols='gene_assignment')
# Salva a tabela extraída no diretório local
> write.table(dt.genes, "resultados.txt", quote=F, sep="\n",
              row.names=T, col.names=T)
```

Como alternativa, a função `geva.quick` pode ser utilizada como uma forma concisa de efetuar sumarização, clusterização, detecção de quantis e finalização a partir de um objeto `GEVAInput`, como indicado no código abaixo:

```
# Realiza sumarização, clusterização, detecção de quantis
# e classificação final, obtendo um GEVAResults
> gresults <- geva.quick(ginput)
```

A função `geva.quick` recebe um `GEVAInput` como primeiro argumento, além de todos os argumentos opcionais das funções `geva.summarize`, `geva.quantiles`, `geva.cluster` e `geva.finalize`, retornado um `GEVAResults`. Ela também aceita um `GEVAResults` como primeiro argumento para refazer toda a análise com os argumentos utilizados para sua obtenção, sobrepondo-os com parâmetros diferentes indicados pelo usuário como argumentos opcionais. O exemplo abaixo demonstra este caso de uso:

```
# Refaz a análise com os mesmos parâmetros, exceto pelo método
# de clusterização, que será por densidade em vez de hierárquico
> gresults2 <- geva.quick(gresults, cluster.method="density")
```

A reaplicação de análises torna-se importante uma vez que diferentes escolhas de métodos podem alterar os resultados finais, cabendo ao usuário decidir o melhor parâmetro dependendo da distribuição dos dados. Levando em conta a variedade de métodos disponíveis pelo GEVA, a próxima seção apresentará os resultados dos testes com o objetivo de verificar como cada parâmetro responde a diversos conjuntos de dados reais.

8 Testes e validação

Com o intuito de validar a capacidade de obtenção de resultados utilizando as funções do pacote, foi estabelecido um protocolo de testes para múltiplos conjuntos de dados de transcritoma. Neste protocolo, foram testados conjuntos de comparações contendo uma combinação de: (i) poucas ou muitas comparações; (ii) nenhum, poucos ou muitos fatores; (iii) média ou mediana; e (iv) clusterização hierárquica ou de densidade. Diferentes métodos de separação de quantis também foram testados, considerando que esta etapa influencia na atribuição qualitativa inicial dos pontos SV. A performance, em termos de velocidade, não foi precisamente mensurada, porém demostrou variações em torno de 20 segundos para 10 mil sondas e aproximadamente um minuto para cerca de 40 mil sondas. Diferenças na quantidade de comparações foram priorizadas para avaliar a precisão estatística dos resultados.

8.1 Transcritomas utilizados

Considerando uma possível heterogeneidade dos dados, optou-se pela utilização de transcritomas que propiciariam um maior controle sobre as variações de expressão gênica. Isso inclui, por exemplo, estudos com mutações genéticas onde a expressão diferencial foi evidente no gene modificado, deste modo podendo-se utilizar este gene como parte dos fatores analisados.

Neste sentido, foram selecionados 28 conjuntos de dados transcritômicos por microarranjo para três diferentes organismos (*Caenorhabditis elegans*, *Mus musculus* e *Rattus norvegicus*) listados na Tabela 2. Dentre os experimentos utilizados, 21 envolvem a regulação direta e negativa da expressão de um ou mais genes através de mutação; três envolvem a supressão gênica por RNA de interferência (GSE9665 e GSE13537) ou anti-microRNA (GSE140190); dois são exclusivamente tratamentos bioquímicos (GSE75157 e GSE75158); um compara diferenças de dieta (GSE77943); e um compara a diferença de idade entre os animais (GSE84511). Os resultados de expressão diferencial destes conjuntos de dados foram obtidos por meio do programa GEAP (NUNES, 2018) utilizando os parâmetros-padrão disponíveis para tratamento estatístico e excluindo amostras que possivelmente enviesariam os valores finais de acordo com as análises de qualidade. Notoriamente, as comparações envolvendo supressão gênica por *knockout* apresentaram um *logFC* negativo para os genes deletados. As tabelas completas de resultados foram exportadas para arquivos de texto e utilizadas como *input* para o GEVA, carregando-as através da função `geva.read.tables`. Cada conjunto GSE foi testado individualmente com exceção dos conjuntos GSE75157 e GSE75158 que, por pertencerem ao mesmo estudo (ANDERSEN et al.,

Tabela 2 – Conjuntos de dados inclusos na etapa de testes. A primeira coluna indica o código identificador do experimento no banco de dados GEO. A segunda coluna indica os tipos de conjuntos amostrais comparados durante as análises de expressão diferencial. A terceira coluna aponta os nomes das condições experimentais rotuladas pelo próprio autor do experimento. A quarta coluna aponta as variáveis experimentais utilizadas como fatores para o GEVA, também utilizando os rótulos dos autores. A quinta coluna indica o organismo envolvido em cada experimento. A sexta coluna informa o número de sondas do conjunto de dados, o que é equivalente ao número de linhas da tabela de entrada para o GEVA. KO: *Knockout* (gene nocaute); WT: *Wild-type* (linhagem selvagem).

Código GEO	Tipo de comparação	Condições (nomes de origem)	Fatores (nomes de origem)	Organismo	Nº de sondas
GSE9665	*Knockdown* x Controle	143; 186; 230	txb-8/9; elt-1; vab-7; hlh-1; nhr-25; elt-3; nob-1; unc-120; hnd-1; lin-26; scrt-1; pal-1	*Caenorhabditis elegans*	22625
GSE27677	Mutante x WT	Fed; Fast	kgb-1; jun-1; daf-16	*Caenorhabditis elegans*	22625
GSE30505	Mutante x WT	L8; Day 8	EV; ash-2	*Caenorhabditis elegans*	22625
GSE51162	Mutante x WT	daf-2; daf-16; daf-2/daf-16	Untreated; UV; Starvation	*Caenorhabditis elegans*	22625
GSE8396	KO x WT	WY14643; Fenofibrate; C18:1; C10; C18:2; C18:3; C20:5; C22:6	Untreated; Treated	*Mus musculus*	45101
GSE14829	KO x WT	H-ras; N-ras; HN-ras	0h; 1h; 8h	*Mus musculus*	12488
GSE17160	KO x WT	TNFa; LPS; TSG6; HA; TSG6+HA	-	*Mus musculus*	45101
GSE25250	KO x WT	KO; Q192; R192	0.35; 0.5; DMSO	*Mus musculus*	35557
GSE27245	KO x WT; Tratado x Não-tratado	ES; NP; TNd2; TNd6; ICRF193 Neuron; ICRF193 Cortical	KO; ICRF193	*Mus musculus*	35557
GSE35061	KO x WT	JMJD1a; G9a	Normoxia; Hypoxia	*Mus musculus*	25697
GSE43419	TG x WT	Tumor; Tumor Mdk KO; Ganglion; Neurosphere	Homozygous; Hemizygous	*Mus musculus*	45101
GSE44387	KO x WT	Stroma; Epithelium	Untreated; BPA; EE2	*Mus musculus*	45101
GSE55143	Mutante x WT	Heterozygote; KO	Saline; VPA	*Mus musculus*	35557
GSE58103	Mutante x WT	KO; Q-tg; R-tg	0.00; 0.50; 0.75; 0.85	*Mus musculus*	35557
GSE58244	KO x WT	Notch3; Notch4	Air; O3 6h; O3 24h; O3 48h	*Mus musculus*	45101
GSE58271	KO x WT	LFnC; HFnC; HFC	Male; Female	*Mus musculus*	35557
GSE63027	KO x WT	3mo; 8mo	GNMT; MAT1A	*Mus musculus*	45101
GSE68365	KO x WT	CARko/PXRko; hCAR/hPXR; PXRko	CO;CPA;DCB	*Mus musculus*	45101
GSE75156	Mutante x WT	1w 0ppm; 1w 40ppm; 1w 120ppm; 4w 0ppm; 4w 40ppm; 4w 120ppm	TG (hCYP2F21); KO (CYP2F2)	*Mus musculus*	45141
GSE75157	Tratado x Não-tratado	1ppm; 5ppm; 10ppm; 20ppm; 40ppm; 120ppm	Lung	*Mus musculus*	45141
GSE75158	Tratado x Não-tratado	1ppm; 5ppm; 10ppm; 20ppm; 40ppm; 120ppm	Liver	*Mus musculus*	45141
GSE77757	KO x WT	6h; 24h; 72h	WT; KO (SIRT1)	*Mus musculus*	45220
GSE77943	Com Dieta x Regular	1w; 9w; 18w	Adipose; Liver; Pancreas	*Mus musculus*	45101
GSE84511	Envelhecido x Jovem	T0; T4; T8; T12; T16; T20	-	*Mus musculus*	45141
GSE120465	KO x WT	AA E17.5; DA E17.5; AA P1; DA P1	Eln; Efemp2; Lox	*Mus musculus*	41801
GSE13537	Supresso x Controle	0h; 1h; 6h	-	*Rattus norvegicus*	31099
GSE100696	Mutante x WT	Heterozygote; KO	Spleen; Hippocampus; Striatum; Olfactory Bulbs; Pituitary Gland	*Rattus norvegicus*	36685
GSE140190	Supresso x Controle	ZD KO; ZS KO; Anti-miR31 5w; Anti-miR31 20w	KO; Anti-miR31	*Rattus norvegicus*	31099

2017), foram concatenados em um único *input* para o GEVA, totalizando 27 testes. Além disso, três conjuntos de dados (GSE17160, GSE84511 e GSE13537) não incluíram fatores na análise (Tabela 2), restringindo seus resultados apenas para a classificação *similar*, enquanto todos os restantes puderam apresentar classificações para genes *fator-específico* ou *fator-dependente*.

Cada conjunto de resultados de expressão diferencial gerou um objeto de entrada do tipo `GEVAInput`. Utilizando os parâmetros-padrão do GEVA indicados na Tabela 3, foi aplicada a função `geva.quick` em cada um destes objetos de entrada. Os parâmetros incluem média como método de sumarização, desvio-padrão como cálculo de variação, resolução de 30% (`0.3`), separação de quantis por distribuição parcial de S e V (método `range.slice`) e clusterização pelo método hierárquico (Tabela 3). Para análises que incluem fatores, foi mantido o grau de significância estatística $\alpha < 0.05$, indicado por pelo parâmetro `p.value = 0.05`. Como resultado, todas as análises demonstraram genes com algum grau de relevância (Figura 18), podendo flutuar de poucas dezenas a até mesmo milhares de observações significativas. Contudo, os parâmetros-padrão demonstraram aparentes flutuações em certas distribuições dos pontos SV, especialmente no caso dos conjuntos GSE27677, GSE35061, GSE43419 e GSE13537, onde há a difusão discrepante de pontos atribuídos como relevantes (Figura 18). Deste modo, tornou-se necessário averiguar se outras opções oferecidas pela ferramenta apresentariam o mesmo viés ou se, de outro modo, seriam capazes de lidar com estes casos de uso.

Tabela 3 – Parâmetros-padrão das funções em cada uma das etapas de análise do GEVA. A primeira coluna indica a função específica onde o parâmetro é fornecido pelo usuário. A segunda coluna indica o tipo de parâmetro e sua sigla. A terceira coluna indica o nome do argumento na função do R. A quarta coluna indica o valor padrão do argumento indicado, caso não seja especificado na função. Todos os argumentos da tabela são igualmente aceitos pela função `geva.quick`.

Função	Parâmetro	Argumento	Valor Padrão
`geva.summarize`	Método de sumarização (S_{met})	`summary.method`	`"mean"`
`geva.summarize`	Método de variação (V_{met})	`variation.method`	`"sd"`
`geva.cluster`	Resolução da clusterização (C_{res})	`resolution`	`0.3`
`geva.cluster`	Método de clusterização (C_{met})	`cluster.method`	`"hierarchical"`
`geva.quantiles`	Método separação de quantis (Q_{met})	`quantile.method`	`"range.slice"`

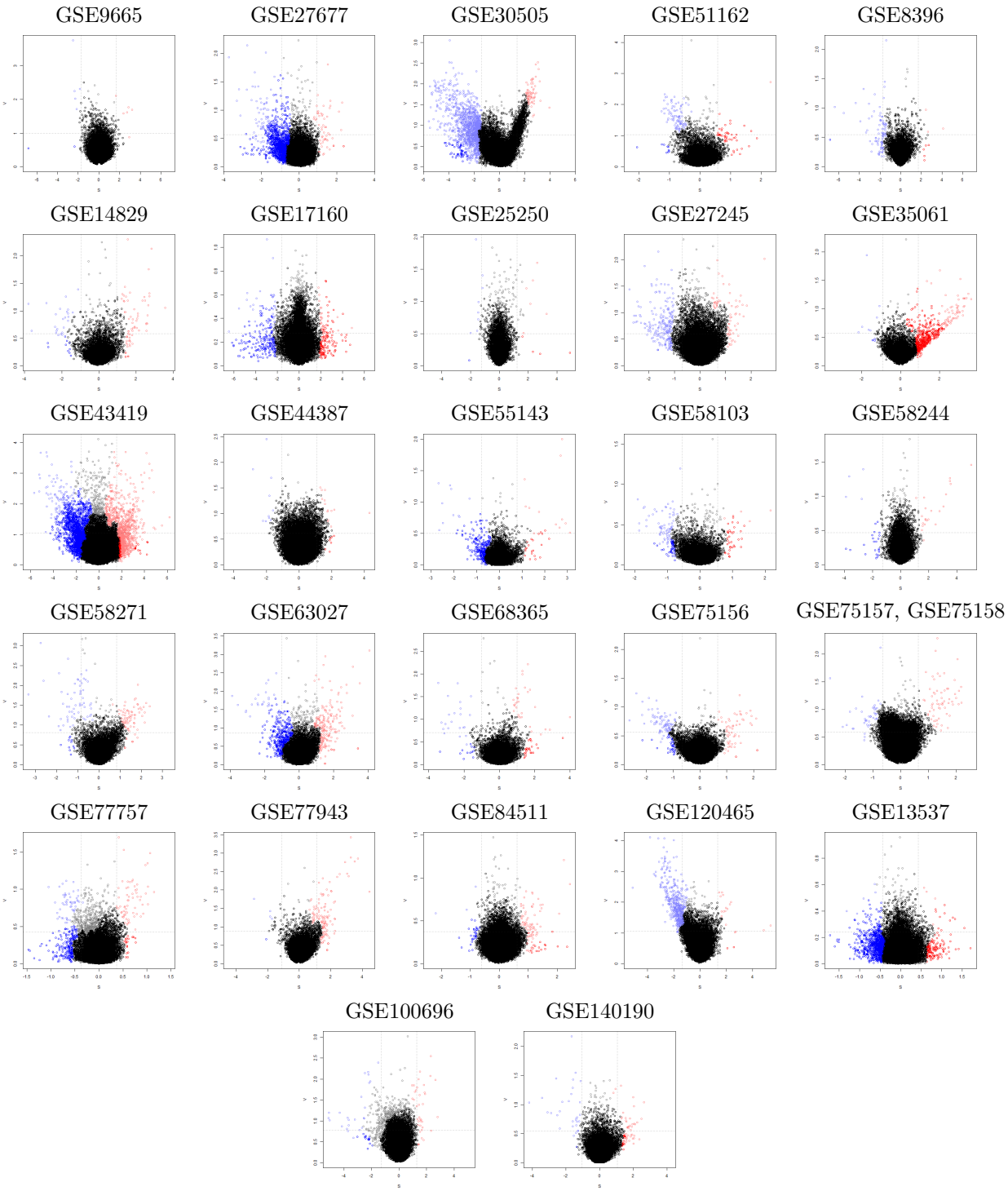

Figura 18 – Gráficos SV representando os resultados finais testados para cada conjunto de dados utilizando os parâmetros-padrão das funções do pacote. Os conjuntos de dados GSE75157 e GSE75158 foram concatenados em um único teste. As cores dos pontos indicam as classificações atribuídas pelo pacote a estes, sendo: preto para nenhuma variação relevante, vermelho e azul opacos para pontos de expressão diferencial *similar* e tons claros de cinza, vermelho e azul para outros tipos de variações (*não-relevante*, *fator-dependente* ou *fator-específica*). Linhas tracejadas indicam separação entre quantis.

8.2 Comparação entre os parâmetros de análise

Esta seção contém os resultados avaliados com diferentes argumentos para as funções do GEVA em comparação aos parâmetros-padrão (Tabela 3). Os números observados podem ser consultados na Tabela 4, e a lista de gráficos SV encontra-se no Apêndice C.

Tabela 4 – Resultados dos testes utilizando diferentes parâmetros entre as funções do pacote. Os valores indicam o número de genes encontrados para uma determinada classificação, onde a ausência de um campo implica que não houve genes com tal atribuição. SM: *similares*; FD: *fator-dependentes*; FS: *fator-específicos*. As siglas para os parâmetros estão indicadas na Tabela 3.

Código GEO	Padrão	C_{res}=0.2 C_{met}="density"	C_{met}="density"	C_{res}=0.4 C_{met}="density"	C_{res}=0.2	C_{res}=0.4	Q_{met}="density"	Q_{met}="k.max.sd"	Q_{met}="proportional"	V_{met}="var"	S_{met}="median" V_{met}="mad"
GSE9665	SM 2 FD 11 FS 2	SM 4 FD 53 FS 6	SM 5 FD 69 FS 6	SM 5 FD 73 FS 6	SM 1 FD 3 FS 2	SM 6 FD 49 FS 10	SM 6 FD 7 FS 2	FD 14 FS 3	SM 6 FD 4 FS 1	SM 1 FD 3 FS 1	SM 2 FD 66 FS 7
GSE27677	SM 1324 FD 110 FS 7	SM 108 FD 118 FS 11	SM 123 FD 129 FS 12	SM 159 FD 189 FS 14	SM 41 FD 41 FS 5	SM 353 FD 363 FS 15	SM 75 FD 32 FS 2	SM 4 FD 1100 FS 67	SM 124 FD 31 FS 2	SM 273 FD 31 FS 1	SM 69 FD 30 FS 2
GSE30505	SM 54	SM 152	SM 174	SM 215	SM 29 FD 1	SM 116	SM 257	SM 28 FD 573 FS 57	SM 1251	SM 914	SM 128 FD 1
GSE51162	SM 52 FD 32 FS 20	SM 56 FD 18 FS 13	SM 66 FD 19 FS 14	SM 77 FD 25 FS 15	SM 10 FD 13 FS 5	SM 72 FD 65 FS 31	SM 46 FD 25 FS 18	SM 1 FD 42 FS 28	SM 50 FD 2 FS 1	SM 8 FD 3 FS 1	SM 6 FD 11 FS 3
GSE8396	SM 6 FD 20 FS 2	SM 9 FD 23 FS 1	SM 13 FD 29 FS 1	SM 21 FD 41 FS 2	SM 2 FD 3 FS 1	SM 84 FD 18 FS 2	SM 94 FD 4 FS 1	SM 2 FD 21 FS 2	SM 8 FD 2 FS 1	SM 31 FD 5 FS 2	SM 29 FD 3 FS 1
GSE14829	SM 13 FD 96 FS 3	SM 39 FD 91 FS 4	SM 50 FD 100 FS 4	SM 87 FD 152 FS 5	SM 11 FD 88 FS 3	SM 252 FD 211 FS 8	SM 84 FD 10	SM 7 FD 113 FS 2	SM 88 FD 24	SM 73 FD 14	SM 71 FD 22 FS 1
GSE17160	SM 325	SM 171	SM 210	SM 280	SM 332	SM 411	SM 280	SM 15	SM 298	SM 446	SM 329
GSE25250	SM 5 FD 82 FS 2	SM 4 FD 199 FS 2	SM 5 FD 204 FS 2	SM 6 FD 243 FS 3	SM 4 FD 36	SM 6 FD 430 FS 7	SM 11 FD 48 FS 2	SM 1 FD 32250 FS 117	SM 7 FD 49 FS 1	SM 11 FD 44	SM 30 FD 30 FS 1
GSE27245	SM 1 FD 203 FS 15	SM 74 FD 268 FS 22	SM 89 FD 309 FS 25	SM 126 FD 374 FS 28	SM 7 FD 46 FS 5	SM 46 FD 699 FS 46	SM 215 FD 18 FS 4	FD 15354 FS 710	SM 280 FD 16 FS 4	SM 164 FD 40 FS 7	SM 60 FD 156 FS 9
GSE35061	SM 446 FD 1	SM 5 FD 1	SM 18 FD 4	SM 40 FD 4	FD 1	SM 256 FD 5	SM 65 FD 1	FD 6	SM 502 FD 1	SM 378 FD 1	SM 319
GSE43419	SM 2001 FD 78 FS 12	SM 256 FD 37 FS 2	SM 304 FD 40 FS 2	SM 430 FD 57 FS 7	SM 63 FD 11	SM 2100 FD 145 FS 12	SM 80 FD 15 FS 1	SM 7 FD 7376 FS 312	SM 1015 FD 12	SM 181 FD 16 FS 1	SM 1407 FD 26
GSE44387	SM 7 FD 29 FS 1	SM 19 FD 234 FS 7	SM 21 FD 256 FS 8	SM 36 FD 381 FS 10	SM 7 FD 29 FS 1	SM 1 FD 743 FS 12	SM 11 FD 6	SM 2 FD 109 FS 8	SM 19 FD 6	SM 58 FD 15 FS 3	SM 230 FD 32 FS 2
GSE55143	SM 279 FD 16	SM 53 FD 26	SM 67 FD 34 FS 1	SM 98 FD 50 FS 2	SM 12 FD 15	SM 97 FD 134 FS 3	SM 26 FD 14	SM 268 FD 27	SM 37 FD 8	SM 44 FD 6	SM 117 FD 16
GSE58103	SM 85 FD 113 FS 3	SM 97 FD 75 FS 3	SM 120 FD 91 FS 3	SM 142 FD 120 FS 3	SM 10 FD 19 FS 1	SM 421 FD 126 FS 4	SM 110 FD 39 FS 1	SM 33 FD 191 FS 1	SM 138 FD 6	SM 151 FD 6 FS 1	SM 406 FD 129 FS 2
GSE58244	SM 22 FD 5 FS 3	SM 11 FD 85 FS 5	SM 15 FD 108 FS 8	SM 22 FD 132 FS 9	SM 4 FD 1	SM 22 FD 12 FS 3	SM 39	SM 2 FD 14 FS 1	SM 37	SM 54	SM 16 FD 49
GSE58271	SM 14 FD 60	SM 19 FD 43	SM 26 FD 50	SM 36 FD 69	SM 1 FD 16	SM 433 FD 132	SM 73 FD 49	SM 1 FD 81 FS 3	SM 87 FD 9	SM 78 FD 41	SM 192 FD 144 FS 2
GSE63027	SM 397 FD 66 FS 17	SM 38 FD 31 FS 10	SM 50 FD 33 FS 12	SM 72 FD 50 FS 12	SM 8 FD 12 FS 9	SM 110 FD 68 FS 14	SM 18 FD 5 FS 6	SM 6 FD 156 FS 30	SM 193 FD 5 FS 6	SM 38 FD 5 FS 3	SM 13 FD 51 FS 10
GSE68365	SM 36 FD 58 FS 5	SM 53 FD 40 FS 5	SM 66 FD 51 FS 7	SM 81 FD 78 FS 7	SM 15 FD 35 FS 3	SM 171 FD 160 FS 11	SM 65 FD 35 FS 2	SM 11 FD 76 FS 13	SM 60 FD 33 FS 2	SM 135 FD 18	SM 303 FD 22
GSE75156	SM 36 FD 129 FS 7	SM 60 FD 70 FS 3	SM 72 FD 83 FS 3	SM 99 FD 99 FS 5	SM 35 FD 125 FS 7	SM 489 FD 157 FS 13	SM 131 FD 12 FS 1	SM 17 FD 194 FS 7	SM 178 FD 4 FS 1	SM 114 FD 4 FS 1	SM 366 FD 14 FS 2
GSE75157 GSE75158	FD 94 FS 26	SM 9 FD 150 FS 56	SM 16 FD 168 FS 76	SM 21 FD 197 FS 102	FD 56 FS 10	SM 14 FD 353 FS 192	SM 56 FD 42 FS 8	FD 29633 FS 8363	SM 67 FD 44 FS 7	SM 102 FD 70 FS 13	SM 226 FD 208 FS 9
GSE77757	SM 232 FD 118 FS 4	SM 246 FD 48 FS 2	SM 273 FD 54 FS 2	SM 329 FD 70 FS 3	SM 58 FD 27 FS 1	SM 782 FD 178 FS 5	SM 256 FD 6 FS 2	SM 108 FD 146 FS 10	SM 284 FD 7 FS 1	SM 242 FD 15 FS 1	SM 310 FD 111 FS 1
GSE77943	SM 3 FD 86 FS 35	SM 5 FD 51 FS 13	SM 10 FD 74 FS 18	SM 15 FD 92 FS 31	SM 3 FD 25 FS 10	SM 6 FD 190 FS 51	SM 121 FD 17 FS 1	FD 89 FS 48	SM 8 FD 15 FS 1	SM 77 FD 17 FS 3	SM 18 FD 51 FS 7
GSE84511	SM 47	SM 38	SM 56	SM 68	SM 24	SM 113	SM 98		SM 128	SM 223	SM 119
GSE120465	SM 2 FD 310 FS 387	SM 6 FD 817 FS 354	SM 12 FD 890 FS 379	SM 15 FD 928 FS 390	SM 1 FD 59 FS 102	SM 10 FD 453 FS 446	SM 2 FD 18 FS 2	FD 38955 FS 1835	SM 646 FD 24 FS 23	SM 28 FD 44 FS 65	SM 12 FD 135 FS 207
GSE13537	SM 1171	SM 437	SM 472	SM 563	SM 198	SM 2497	SM 70	SM 99	SM 1148	SM 607	SM 1617
GSE100696	SM 15 FD 281 FS 60	SM 38 FD 771 FS 132	SM 40 FD 829 FS 140	SM 54 FD 910 FS 149	SM 7 FD 29 FS 5	SM 106 FD 891 FS 154	SM 23 FD 14 FS 2	FD 32693 FS 1310	SM 30 FD 15 FS 2	SM 50 FD 36 FS 11	SM 263 FD 43 FS 6
GSE140190	SM 19 FD 25 FS 10	SM 12 FD 21 FS 2	SM 20 FD 22 FS 2	SM 25 FD 22 FS 4	SM 2 FD 4 FS 1	SM 5 FD 34 FS 9	SM 40 FD 7 FS 1	SM 1 FD 40 FS 17	SM 78	SM 84 FD 5	SM 54 FD 13 FS 1

8.2.1 Métodos de sumarização

Em relação aos parâmetros-padrão de sumarização (média e desvio-padrão), a variância é caracterizada por um achatamento relativo do aglomerado indiferencial e pela introdução de valores discrepantes no eixo V (Figuras 19A e 19B) em razão de sua diferença exponencial de escala. Em contrapartida, ter mediana e MAD como parâmetros resulta em separações locais dos pontos no centro do eixo S, ao mesmo tempo que uma distribuição mais estendida no eixo V (Figura 19C) em relação aos outros métodos.

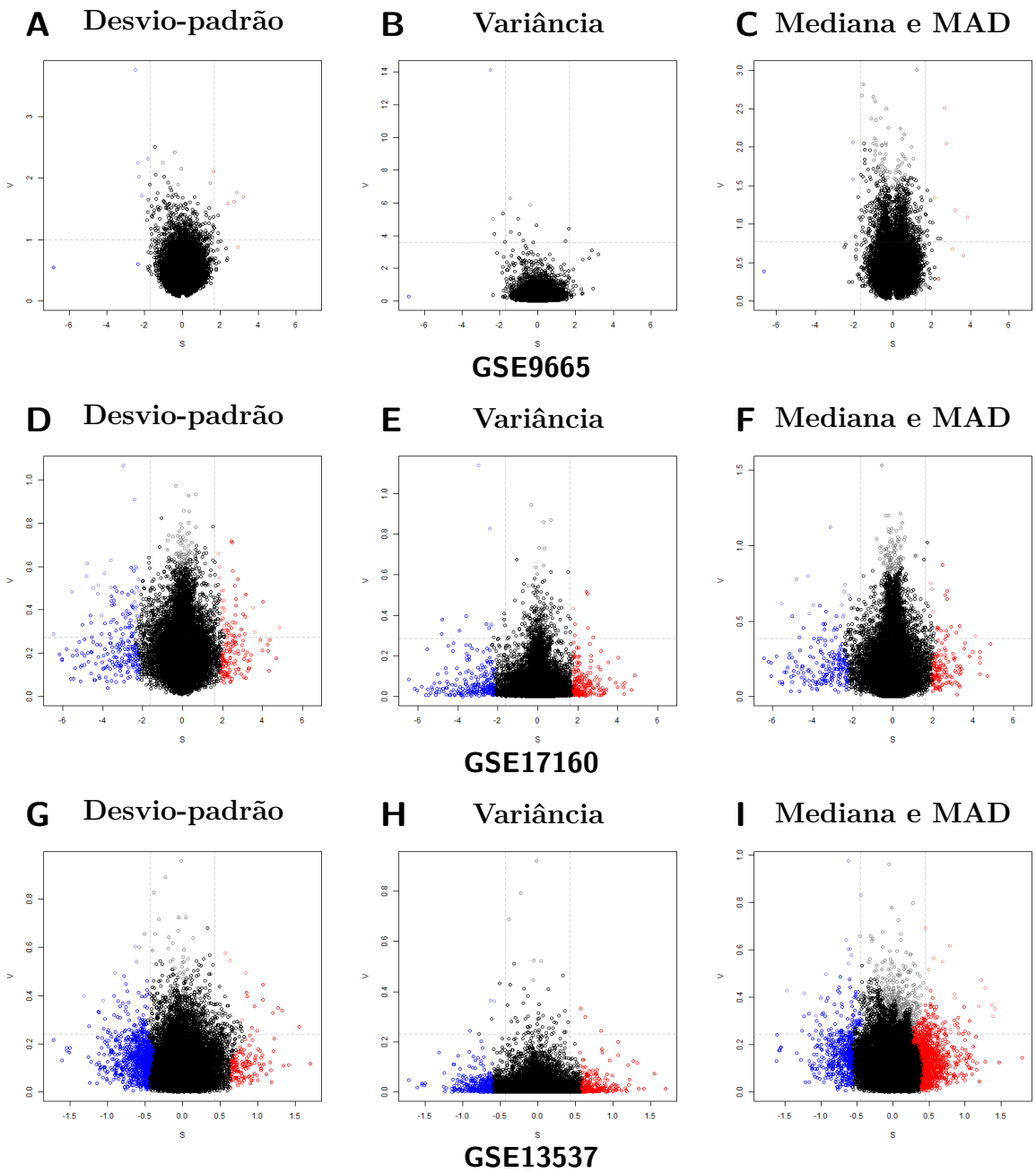

Figura 19 – Gráficos SV com os resultados em resposta à utilização de diferentes parâmetros de sumarização, descritos no topo de cada figura. As descrições se referem os conjuntos GSE9665 (A–C), GSE17160 (D–F) e GSE13537 (G–I). As demais figuras podem ser encontradas no Apêndice C.

As mesmas características destes parâmetros são observadas entre diferentes conjuntos de dados, como GSE9665, GSE17160 e GSE13537 (Figura 19). Contudo, para o GSE9665 utilizando o método MAD, a distribuição dos pontos no eixo V apresentaram menor discrepância e mais resultados fator-dependentes e fator-específicos (Tabela 4), enquanto para o GSE17160 houve uma menor diferença entre média com desvio-padrão e mediana com MAD (Figuras 19D, 19E e 19F). Ainda em relação ao GSE9665, a variância apresentou poucos resultados em comparação aos testes com outros métodos, sendo 4 genes relevantes por variância contra 15 por desvio-padrão e 75 por mediana e MAD (Tabela 4). Para o GSE13537, porém, houve uma estimativa exacerbada de genes *similares* utilizando-se desvio-padrão (1171) ou MAD (1617), enquanto que a variância apresentou um número mais razoável de resultados (607) conforme a distribuição observada no gráfico (Tabela 4; Figuras 19G, 19H e 19I). Possivelmente, o aspecto achatado do aglomerado indiferencial pela variância teria direcionado o algoritmo de clusterização hierárquico, causando menos divisões dentro do aglomerado indiferencial e prevenindo a formação de grandes *clusters* próximos dos limiares dos quantis, os quais tendenciariam a correção dos quantis nestes casos.

Estas observações indicam que média e desvio-padrão, assim como mediana com MAD, poderiam ser utilizados para detecção de um maior número de genes, especialmente em quantis de maior variação, enquanto que a variância poderia ser aplicada para reduzir o número de genes encontrados quando há uma aparente exorbitância de observações.

8.2.2 Métodos de detecção de quantis

O GEVA disponibiliza quatro métodos capazes calcular partições significativas dentro dos eixos S e V (`range.slice`, `density`, `k.max.sd` e `proportional`), separando a extensão de cada eixo em quantis. Os cálculos dos métodos testados podem ser descritos, resumidamente, da seguinte forma:

- `range.slice` (padrão): separação ocorre no ponto que mais se aproxima de uma fração da extensão total (por padrão, 25% da extensão no eixo);
- `density`: separação ocorre no ponto cuja densidade entre k pontos mais se aproxima de uma fração da extensão de todas as densidades (por padrão, 25% da extensão das densidades e $k = 16$);
- `k.max.sd`: separação ocorre no ponto com maior desvio-padrão da distância entre seus k vizinhos (por padrão, $k = 16$);
- `proportional`: separação é definida pela divisão exata do eixo (*i.e.*, espaço entre o menor e o maior valor) pelo número de quantis. Neste caso, todos os quantis são do mesmo tamanho.

Em relação ao método padrão, `range.slice`, o método `density` apresentou um número menor de genes relevantes em diversos testes, como GSE27667 (109 contra 1441), GSE35061 (66 contra 447), GSE63027 (29 contra 480) e GSE13537 (70 contra 1171), porém houve casos com maior número, como GSE8396 (95 contra 28) e GSE30505 (257 contra 54), embora os resultados de ambos permaneçam similares na maioria dos testes (Tabela 4). Como observado na Figura 20, as delimitações por `density` resultaram em uma área maior do quantil central inferior em relação aos demais métodos para o GSE43419 e o GSE35061, prevenindo uma estimativa exorbitante de observações similar às Figuras 20A, 20D, 20I e 20L.

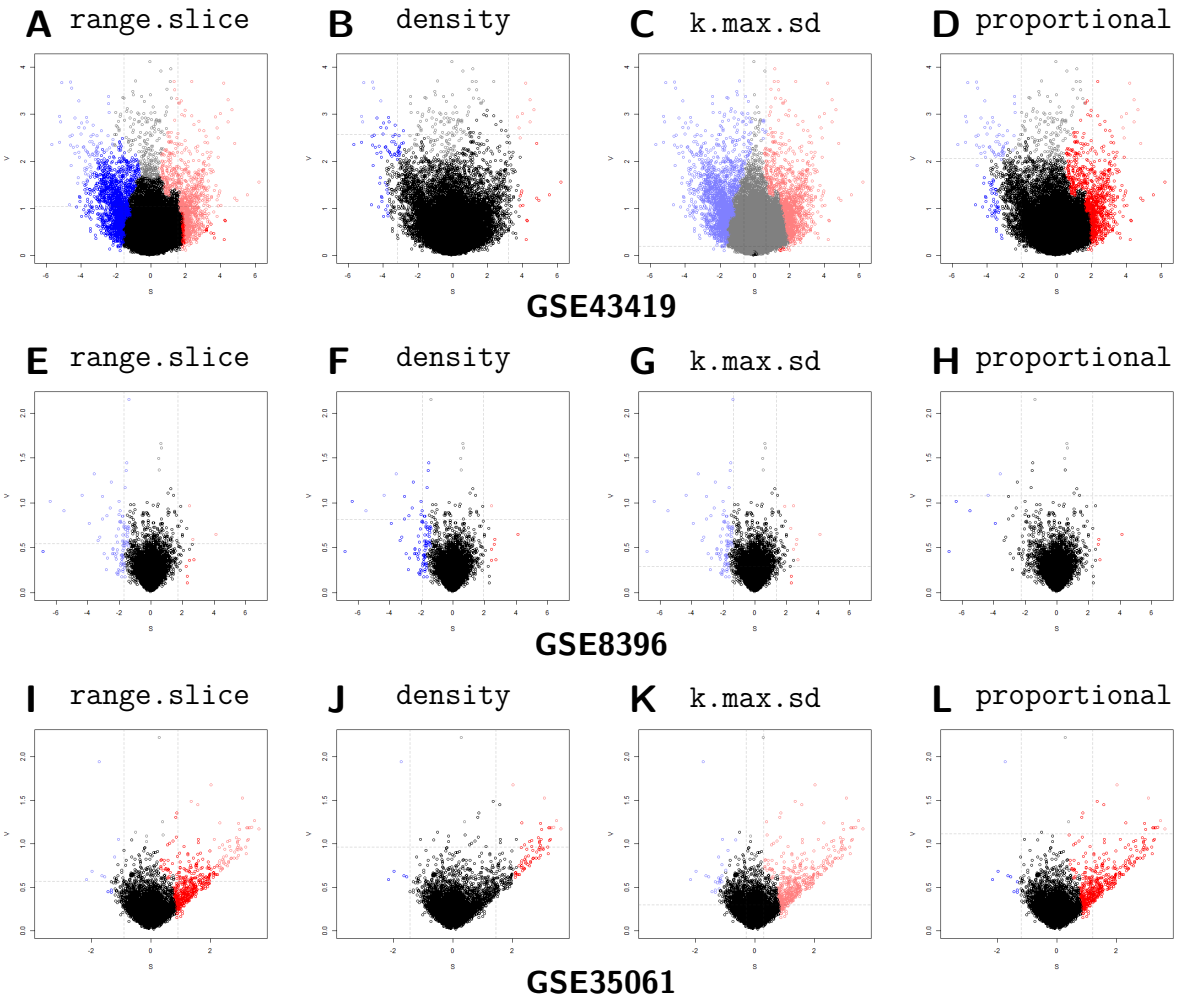

Figura 20 – Gráficos SV com os resultados em resposta à utilização de diferentes parâmetros de detecção de quantis, indicados no topo de cada figura. As descrições se referem os conjuntos GSE43419 (A–D), GSE8396 (E–H) e GSE35061 (I–L). As demais figuras podem ser encontradas no Apêndice C.

O método `k.max.sd`, por outro lado, foi o que gerou os resultados mais imprevisíveis entre todos os métodos, apresentando números razoáveis em certos casos como GSE9665 (17) e GSE58244 (17), porém valores exorbitantes em outros como GSE27677 (1171), GSE25250 (32368) e GSE27245 (16064) (Tabela 4). Estes extremos são originados devido ao

fato da divisão calculada por `k.max.sd` localizar-se no interior do aglomerado indiferencial, o que pode ser observado em nove dentre os 27 testes (seção C.8 do Apêndice C).

Finalmente, o método `proportional` apresentou resultados semelhantes ao parâmetro padrão (`range.slice`) em GSE35061 (Figuras 20I e 20L), e ao mesmo tempo menos resultados em GSE8396 (11) que os parâmetros padrão (28) e `density` (43) (Tabela 4). Neste caso, a presença de pontos discrepantes significou uma maior extensão dos eixos S e V, resultando em uma divisão dos quantis mais distante do aglomerado indiferencial.

Em súmula, essas observações sugerem que os métodos `range.slice` (padrão) e `density` apresentam uma divisão mais adaptativa a diferentes distribuições de pontos SV, bem como uma quantidade razoável de resultados relevantes, sendo o `density` ligeiramente mais rigoroso em certos casos por ocasionar em um número ainda menor de resultados finais. Em contrapartida, os métodos `k.max.sd` e `proportional` apresentaram limitações de imprevisibilidade e de sensibilidade a valores discrepantes, respectivamente.

8.2.3 Métodos de clusterização

Para todos os conjuntos de dados, foram testados ambos os métodos de clusterização (hierárquica e de densidade) em três diferentes resoluções (20%, 30% e 40%), sendo 30% a resolução padrão. Dentre os 27 conjuntos de dados testados, 22 apresentaram uma correlação majoritária entre número de genes classificados como relevantes e a resolução, enquanto que os conjuntos GSE27677, GSE35061, GSE44387, GSE55143, GSE63027 apresentaram essa correlação para clusterização por densidade, mas não pela hierárquica (Tabela 4). Além disso, em todas as resoluções, foi observada uma diferença majoritária entre os métodos em relação a sua capacidade de detectar agrupamentos. Para o conjunto GSE68365, que apresentou um aglomerado indiferencial compacto e pontos SV esparsos (Figura 21), a clusterização hierárquica foi capaz de identificar padrões de distribuição similares entre pontos aparentemente dispersos, enquanto que a clusterização de densidade limitou-se ao *cluster* relativo ao aglomerado indiferencial (Figuras 21A e 21B). A diferença entre os resultados finais, embora sutil, acarretou em um aumento na detecção de genes *similares* na clusterização por densidade (66) em comparação à hierárquica (36) (Figuras 21C e 21D; Tabela 4).

Por outro lado, para o conjunto GSE30505, caracterizado por um aglomerado indiferencial anômalo com grandes concentrações de pontos transvazando aos quantis vizinhos (Figura 22), a clusterização de densidade foi capaz de separar os pontos SV com maior fidelidade ao quantil mais indiferencial (Figura 22B), enquanto que a clusterização hierárquica seccionou uma grande parcela do aglomerado indiferencial que se encontrava predominante em quantis com expressão diferencial variante Figura 22A. Como resultado, a clusterização hierárquica reduziu o número de genes *similares* em mais da metade em relação à clusterização por densidade (54 para 174, respectivamente) na resolução padrão

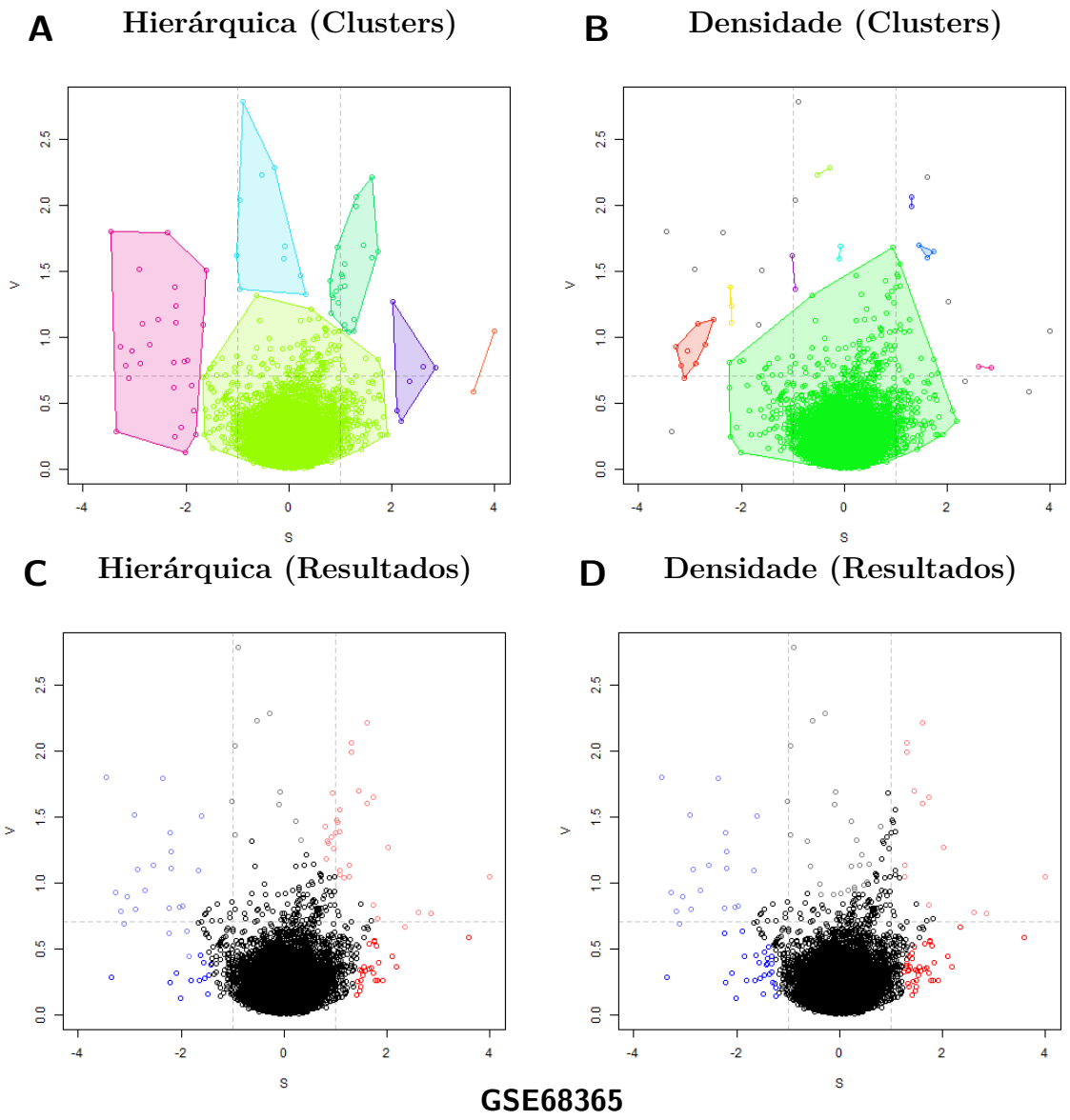

Figura 21 – Gráficos SV comparando os resultados em resposta a clusterização hierárquica e de densidade, como indicado no topo de cada figura. As descrições se referem ao conjunto GSE68365. (A–B) Destaque aos *clusters* formados, distinguidos pelas cores dos pontos (escolhidas arbitrariamente) e contornados por um polígono para melhor visualização; (C–D) Resultados finais, em quantis ajustados, para cada método de clusterização.

(Tabela 4), mesmo quando os genes exclusos estavam claramente associados a um quantil *similar* (Figura 22D).

Essas observações indicam que a clusterização hierárquica poderia ser mais adequada para aglomerados basais concentrados e pontos dispersos, enquanto que a clusterização por densidade lidaria melhor com aglomerados basais irregulares ou quando há grandes concentrações de pontos excedendo quantis vizinhos. A resolução pode ser usada nesses casos para aumentar ou diminuir o número de observações finais proporcionalmente, especialmente no caso de clusterização por densidade.

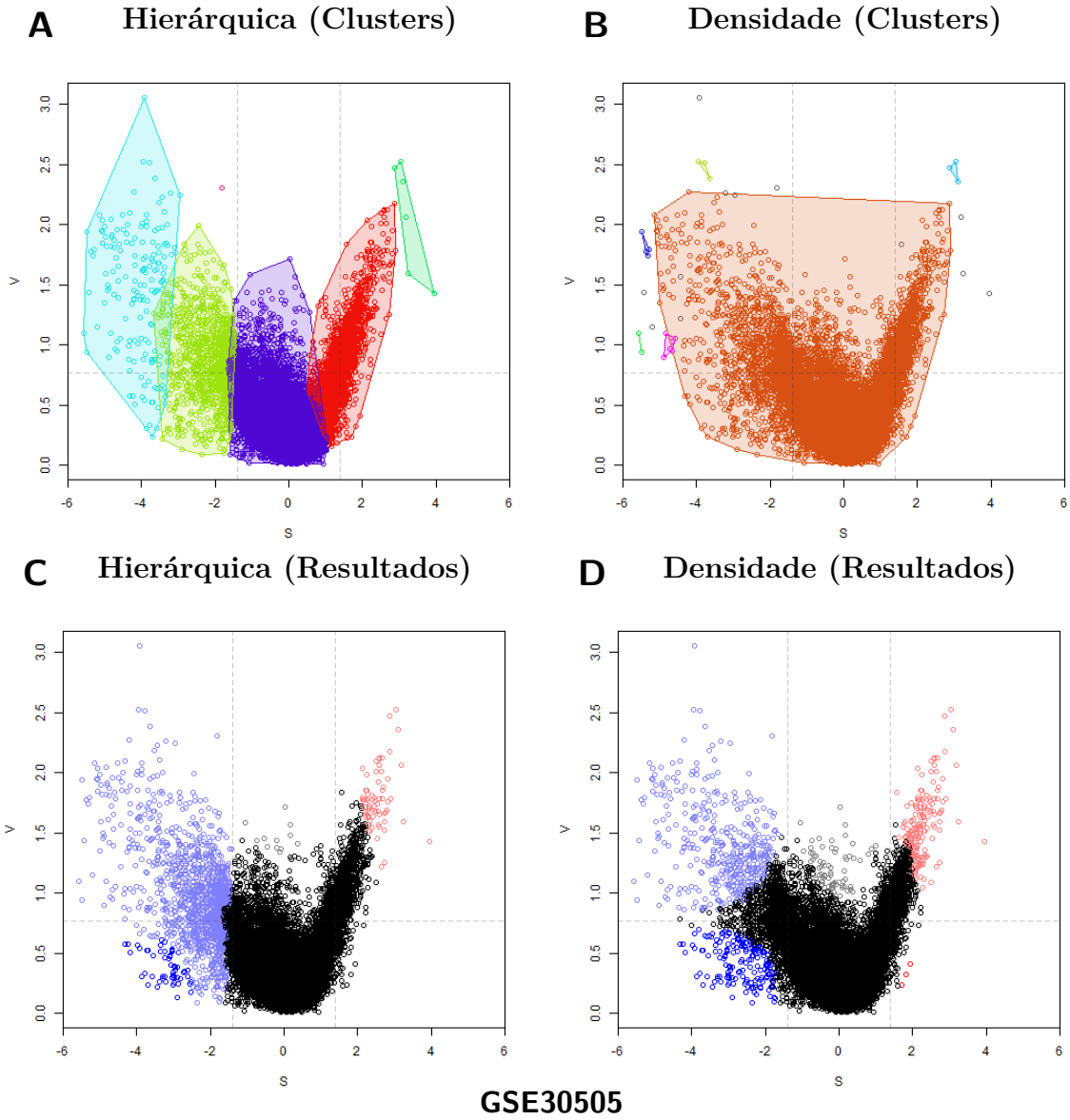

Figura 22 – Gráficos SV comparando os resultados em resposta a clusterização hierárquica e de densidade, como indicado no topo de cada figura. As descrições se referem ao conjunto GSE30505, onde pode-se notar uma distribuição irregular do aglomerado indiferencial. (A–B) Destaque aos *clusters* formados, distinguidos pelas cores dos pontos (escolhidas arbitrariamente) e contornados por um polígono para melhor visualização; (C–D) Resultados finais, em quantis ajustados, para cada método de clusterização.

8.3 Concordância com as observações experimentais

Para finalizar a validação, os genes classificados como relevantes pelos parâmetros-padrão foram avaliados e comparados com as evidências descritas nas publicações de origem.

Para o conjunto GSE9665, o gene *tbx-8* foi classificado como *similar* (*logFC* médio de -6,82), corroborando com sua supressão intencional por RNAi durante experimento (YANAI et al., 2008). Para o conjunto GSE27677, o gene *daf-16* foi *similar* (*logFC* médio de -0.72), coincidindo com a sub-expressão observada pelo autor causada pelo nocaute deste gene ou de seus ativadores, *kgb-1* e *jun-1* (UNO et al., 2013). Nestes mesmos resultados, o gene *jun-1* foi confirmado como *fator-específico* para o fator "jun-1", representando experimentos com seu nocaute (UNO et al., 2013). Para o conjunto GSE30505, contudo, os genes *ash-1* e *glp-1* não foram classificados como relevantes dentro dos parâmetros-padrão mesmo indicando sub-expressão, em contraste ao que foi predito pelo experimento (GREER et al., 2010). Esta incoerência da classificação pode ter ocorrido devido à predominância de um número exacerbado de genes com expressão diferencial, indicando um possível ruído originado nos dados experimentais. Para o conjunto GSE51162, os genes nocauteados (*daf-2* e *daf-16*) também não encontraram-se em qualquer classificação relevante. Porém, este resultado foi esperado considerando que estes genes não foram destacados pelo próprio autor (MUELLER et al., 2014), além do fato de que suas condições foram misturadas em cada fator neste protocolo de testes, o que contrapõe a intenção de distingui-los entre os experimentos. Para o conjunto GSE14829, o gene NRas apresentou-se como *similar* (*logFC* médio de -2,76), corroborando com seu nocaute e sub-expressão majoritária entre as condições (CASTELLANO et al., 2009). Para o conjunto GSE27245, o gene Top2b foi classificado como *fator-dependente* com sub-expressão dentro do fator representando seu nocaute, em concordância com seu estudo de origem (TIWARI et al., 2012).

Além disso, para o conjunto GSE43419, o gene MYCN foi classificado como *similar* (*logFC* médio de 1,89), em concordância com sua amplificação no experimento realizado (MURAKAMI-TONAMI et al., 2014). Para o conjunto GSE58103, o gene PON-1 foi apontado como fator-específico para o fator "KO" (condições de nocaute para PON-1), estando sub-expresso nesta condição de acordo com o experimento pelos autores do estudo (COLE et al., 2014). Para o conjunto GSE58271, o gene Cyp51 foi classificado como *similar* (*logFC* médio de -1,38), estando de acordo com sua sub-expressão por nocaute (LORBEK et al., 2015). Para o conjunto GSE63027, os genes Gnmt e Mat1a foram classificados como *fator-específico* (fator "GNMT") e *fator-dependente*, respectivamente. Como apenas dois fatores foram levados em conta neste caso, ambas as classificações tornam-se válidas para assinalar a resposta ao nocaute de ambos os genes dentro de seus fatores correspondentes, o que corrobora com o estudo de origem (FRADES et al., 2015). Para o conjunto GSE75156, o gene Cyp2F2 foi marcado como *similar* (*logFC* médio de -1,20), estando de acordo com

seu experimento de supressão por nocaute e transgenia (ANDERSEN et al., 2017). Para o conjunto GSE120465, os genes Lox e Eln foram classificados como *fator-específicos* para seus fatores correspondentes, enquanto que Efemp2 foi classificado como *fator-dependente*. Além destes três genes sub-expressos por nocaute, dois genes marcados como *similares*, Col8a1 ($logFC$ médio de 2,59) e Igfbp2 ($logFC$ médio de 2,07), foram apontados no estudo de origem como super-expressos em todos os experimentos (STAICULESCU et al., 2018). Para o conjunto GSE13537, dois genes supressos por RNAi foram classificados como *similares*, sendo estes o Mef2d ($logFC$ médio de -0,89) e o gene Mef2a em quatro sondas diferentes ($logFC$ médios de -1,54, -1,61, -0,98 e -0,87), conforme o experimento realizado pelos autores (FLAVELL et al., 2008). Finalmente, para o conjunto GSE100696, o gene Csf1r foi marcado como *fator-dependente*, o que condiz com o contraste entre os fatores "Heterozygote" (heterozigoto) e "KO" (nocaute), visto que o genótipo do organismo testado altera seus níveis basais de expressão (PRIDANS et al., 2018).

Com base nas observações indicadas nesta seção, pode-se concluir que o GEVA possui a capacidade de distinguir e classificar genes que demonstram consistência ou variação de expressão diferencial, de forma que seus resultados condizem com as implicações experimentais. Embora ainda seja necessário que os dados de entrada apresentem uma qualidade razoável em razão de potenciais ruídos, os métodos permanecem válidos para aplicação sobre dados reais contanto que tenham sido devidamente processados, como espera-se de qualquer tipo de análise de larga escala.

9 Conclusão e Perspectivas

Por fim, o pacote GEVA foi desenvolvido a partir da síntese de uma série de métodos estatísticos em uma única e nova análise, a qual pode ser aplicada sobre genes diferencialmente expressos para avaliar sua resposta entre diferentes condições biológicas. Por meio de algoritmos avançados de sumarização, clusterização, detecção de quantis e ANOVA, desenvolvidos em R, a metodologia criada permite encontrar não apenas expressões diferenciais similares entre múltiplos experimentos, mas também variações de expressão gênica relevantes entre condições experimentais agrupadas em fatores, deste modo oportunizando a exploração de mais camadas de complexidade no contexto biológico. Tal profundidade de prospecção torna-se ainda mais necessária perante a problemas multifatoriais, como doenças crônicas ainda pouco entendidas. Neste caso, considerando que o GEVA demonstrou capacidade de lidar com dados reais envolvendo múltiplos desenhos experimentais, além de corroborar com os estudos de origem avaliados, pode-se afirmar que a ferramenta desenvolvida tem potencial de aplicação prática na finalidade de encontrar novos alvos terapêuticos para doenças mais complexas.

Passos seguintes

Melhores argumentos-padrão

A atual implementação do GEVA possui um nível razoável de flexibilidade e provê uma série de opções, incluindo certos procedimentos ainda em fase de testes, como a combinação entre múltiplas análises de clusterização e de detecção de scores dos pontos SV. Considerando a infinidade de possíveis combinações de análise com os argumentos disponíveis no momento, o próximo passo seria otimizar as combinações de testes para encontrar parâmetros que gerassem resultados mais condizentes com a realidade biológica, a fim de se estabelecer melhores parâmetros-padrão nas próximas versões do pacote. Essa aplicação seria auxiliada por abordagens de meta-heurística (*e.g.*, algoritmos genéticos), que associariam os parâmetros com a exatidão dos dados experimentais. Neste caso, não apenas a disposição destes algoritmos seria necessária, mas também um conjunto ainda maior e mais controlado de dados experimentais teria de ser providenciado a fim de se garantir a precisão estatística dos resultados.

Inclusão de aprendizado de máquina

Com o fluxo de procedimentos bem estabelecido, será possível incluir novos sistemas de pontuação que agregarão e influenciarão a classificação final dos genes, bem como sua

relevância para a análise. No momento, os resultados são obtidos utilizando essencialmente a estatística, com exceção dos fatores definidos pelo usuário que, teoricamente, oferecem um viés mais biológico aos valores finais. Por outro lado, com a grande disponibilidade de conjuntos de dados públicos, certos padrões de $logFC$ poderiam ser detectados e utilizados como informações adicionais com o auxílio de algoritmos de aprendizado de máquina para monitorar a classificação final. Deste modo, as análises proporcionariam a exploração de uma outra camada de complexidade, a qual envolve contexto biológico como forma de pontuação para classificações finais. Essa última observação seria aplicada em seguida ao estabelecimento dos argumentos-padrão otimizados.

Referências

ABDI, H. Bonferroni and šidák corrections for multiple comparisons. *Encyclopedia of measurement and statistics*, Sage Thousand Oaks, CA, v. 3, p. 103–107, 2007. Citado na página 4.

AFFYMETRIX. *Home Page*. 2017. Accessed 03 Sep 2019. Disponível em: <http://www.affymetrix.com/site/mainPage.affx>. Citado na página 6.

ALEXA, A.; RAHNENFÜHRER, J. *Gene set enrichment analysis with topGO*. 2009. Citado na página 35.

ALLAIRE, J. Rstudio: integrated development environment for R. *Boston, MA*, Citeseer, v. 770, p. 394, 2012. Citado 2 vezes nas páginas 8 e 33.

ALONSO, R. et al. Babelomics 5.0: functional interpretation for new generations of genomic data. *Nucleic Acids Res*, Oxford University Press, v. 43, n. W1, p. W117–W121, 2015. Citado na página 6.

ANDERSEN, M. E. et al. Assessing molecular initiating events (mies), key events (kes) and modulating factors (mfs) for styrene responses in mouse lungs using whole genome gene expression profiling following 1-day and multi-week exposures. *Toxicology and applied pharmacology*, Elsevier, v. 335, p. 28–40, 2017. Citado 2 vezes nas páginas 58 e 68.

BARRETT, T. et al. Ncbi geo: archive for functional genomics data sets—update. *Nucleic acids research*, Oxford University Press, v. 41, n. D1, p. D991–D995, 2012. Citado 2 vezes nas páginas 5 e 6.

BECKER, G. *SearchTrees: spatial search trees*. 2016. 1–2 p. Disponível em: <https://cran.r-project.org/package=SearchTrees>. Citado na página 34.

BENGTSSON, H. et al. *matrixStats: Functions that apply to rows and columns of matrices (and to vectors)*. 2018. Disponível em: <https://cran.r-project.org/package=matrixStats>. Citado na página 34.

BENJAMINI, Y.; HOCHBERG, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *J R Stat Soc B*, JSTOR, p. 289–300, 1995. Citado na página 4.

BROWN, M. B.; FORSYTHE, A. B. Robust tests for the equality of variances. *Journal of the American Statistical Association*, Taylor & Francis, v. 69, n. 346, p. 364–367, 1974. Citado na página 15.

BUMGARNER, R. Overview of dna microarrays: types, applications, and their future. *Current protocols in molecular biology*, Wiley Online Library, v. 101, n. 1, p. 22–1, 2013. Citado na página 2.

CARROLL, R. J.; SCHNEIDER, H. A note on levene's tests for equality of variances. *Statistics & probability letters*, Elsevier, v. 3, n. 4, p. 191–194, 1985. Citado na página 15.

CARVALHO, B. S.; IRIZARRY, R. A. A framework for oligonucleotide microarray preprocessing. *Bioinformatics*, Oxford University Press, v. 26, n. 19, p. 2363–2367, 2010. Citado na página 7.

CASTELLANO, E. et al. Serum-dependent transcriptional networks identify distinct functional roles for h-ras and n-ras during initial stages of the cell cycle. *Genome biology*, Springer, v. 10, n. 11, p. R123, 2009. Citado na página 67.

COLE, T. B. et al. Repeated gestational exposure of mice to chlorpyrifos oxon is associated with paraoxonase 1 (pon1) modulated effects in maternal and fetal tissues. *Toxicological Sciences*, Oxford University Press, v. 141, n. 2, p. 409–422, 2014. Citado na página 67.

DATA, N. S. *Data Policies*. 2020. Accessed 15 Sep 2020. Disponível em: <https://www.nature.com/sdata/policies/data-policies>. Citado na página 6.

DU, P.; KIBBE, W. A.; LIN, S. M. lumi: a pipeline for processing Illumina microarray. *Bioinformatics*, Oxford University Press, v. 24, n. 13, p. 1547–1548, 2008. Citado na página 7.

DUMAS, J.; GARGANO, M. A.; DANCIK, G. M. shinygeo: a web-based application for analyzing gene expression omnibus datasets. *Bioinformatics*, Oxford University Press, v. 32, n. 23, p. 3679–3681, 2016. Citado na página 6.

DUNNING, M. J. et al. beadarray: R classes and methods for Illumina bead-based data. *Bioinformatics*, Oxford University Press, v. 23, n. 16, p. 2183–2184, 2007. Citado na página 7.

EDDELBUETTEL, D. et al. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, v. 40, n. 8, p. 1–18, 2011. Disponível em: <https://cran.r-project.org/package=Rcpp>. Citado na página 33.

EDGEWORTH, F. Y. Xxii. on a new method of reducing observations relating to several quantities. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Taylor & Francis, v. 25, n. 154, p. 184–191, 1888. Citado na página 27.

ELSEVIER. *Research Data Policies*. 2020. Accessed 15 Sep 2020. Disponível em: <https://www.elsevier.com/about/policies/research-data>. Citado na página 6.

FLAVELL, S. W. et al. Genome-wide analysis of mef2 transcriptional program reveals synaptic target genes and neuronal activity-dependent polyadenylation site selection. *Neuron*, Elsevier, v. 60, n. 6, p. 1022–1038, 2008. Citado na página 68.

FOX, J. et al. Package 'car'. *Vienna: R Foundation for Statistical Computing*, 2012. Citado na página 16.

FRADES, I. et al. Integrative genomic signatures of hepatocellular carcinoma derived from nonalcoholic fatty liver disease. *PLoS One*, Public Library of Science, v. 10, n. 5, p. e0124544, 2015. Citado na página 67.

GAUTIER, L. et al. affy–analysis of affymetrix genechip data at the probe level. *Bioinformatics*, Oxford University Press, v. 20, n. 3, p. 307–315, 2004. Citado na página 7.

GENTLEMAN, R. C. et al. Bioconductor: open software development for computational biology and bioinformatics. *Genome biology*, Springer, v. 5, n. 10, p. R80, 2004. Citado na página 5.

GREER, E. L. et al. Members of the h3k4 trimethylation complex regulate lifespan in a germline-dependent manner in c. elegans. *Nature*, Nature Publishing Group, v. 466, n. 7304, p. 383–387, 2010. Citado na página 67.

HAHSLER, M.; PIEKENBROCK, M.; DORAN, D. dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, Springer-Verlag, v. 91, n. 1, p. 1–30, 2019. Disponível em: <https://cran.r-project.org/package=dbscan>. Citado na página 34.

HOSKEN, D.; BUSS, D.; HODGSON, D. Beware the F test (or, how to compare variances). *Animal behaviour*, Elsevier, v. 136, p. 119–126, 2018. Citado 2 vezes nas páginas 14 e 15.

IHAKA, R.; GENTLEMAN, R. R: a language for data analysis and graphics. *Journal of computational and graphical statistics*, Taylor & Francis Group, v. 5, n. 3, p. 299–314, 1996. Citado 3 vezes nas páginas 5, 8 e 33.

ILLUMINA. Genomestudio genotyping. Illumina, Inc. San Diego, CA, 2010. Accessed 10 Oct 2020. Disponível em: <https://www.illumina.com/techniques/microarrays/array-data-analysis-experimental-design/genomestudio.html>. Citado na página 6.

KALLIO, M. A. et al. Chipster: user-friendly analysis software for microarray and other high-throughput data. *BMC Genomics*, BioMed Central, v. 12, n. 1, p. 507, 2011. Citado na página 6.

LEVENE, H. *Contributions to Probability and Statistics. Essays in Honor of Harold Hotelling*. [S.l.]: Stanford University Press, 1960. 278–292 p. Citado na página 15.

LIU, C.-H.; DI, Y. P. Analysis of rna sequencing data using clc genomics workbench. In: *Molecular Toxicology Protocols*. [S.l.]: Springer, 2020. p. 61–113. Citado na página 6.

LORBEK, G. et al. Lessons from hepatocyte-specific cyp51 knockout mice: impaired cholesterol synthesis leads to oval cell-driven liver injury. *Scientific reports*, Nature Publishing Group, v. 5, p. 8777, 2015. Citado na página 67.

LOVE, M. I.; HUBER, W.; ANDERS, S. Moderated estimation of fold change and dispersion for rna-seq data with deseq2. *Genome biology*, Springer, v. 15, n. 12, p. 550, 2014. Citado na página 7.

MARWAH, V. S. et al. eutopia: solution for omics data preprocessing and analysis. *Source Code Biol Med*, BioMed Central, v. 14, n. 1, p. 1, 2019. Citado na página 6.

MUELLER, M. M. et al. Daf-16/foxo and egl-27/gata promote developmental growth in response to persistent somatic dna damage. *Nature cell biology*, Nature Publishing Group, v. 16, n. 12, p. 1168–1179, 2014. Citado na página 67.

MÜLLNER, D. et al. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software*, Foundation for Open Access Statistics, v. 53, n. 9, p. 1–18, 2013. Disponível em: <https://cran.r-project.org/package=fastcluster>. Citado 2 vezes nas páginas 19 e 34.

MURAKAMI-TONAMI, Y. et al. Inactivation of smc2 shows a synergistic lethal response in mycn-amplified neuroblastoma cells. *Cell cycle*, Taylor & Francis, v. 13, n. 7, p. 1115–1131, 2014. Citado na página 67.

NUNES, I. J. G. Gene expression analysis platform (geap): uma plataforma flexível e intuitiva para análise de transcriptoma. 2018. Citado 2 vezes nas páginas 6 e 56.

PEBESMA, E. J.; BIVAND, R. S. *Classes and methods for spatial data in R*. 2005. 9–13 p. Disponível em: <https://CRAN.R-project.org/doc/Rnews/>. Citado na página 34.

PRIDANS, C. et al. Pleiotropic impacts of macrophage and microglial deficiency on development in rats with targeted mutation of the csf1r locus. *The Journal of Immunology*, Am Assoc Immnol, v. 201, n. 9, p. 2683–2699, 2018. Citado na página 68.

RAYKOV, Y. P. et al. What to do when k-means clustering fails: a simple yet principled alternative algorithm. *PloS one*, Public Library of Science San Francisco, CA USA, v. 11, n. 9, p. e0162259, 2016. Citado na página 16.

RITCHIE, M. E. et al. limma powers differential expression analyses for RNA-sequencing and microarray studies. *Nucleic acids research*, Oxford University Press, v. 43, n. 7, p. e47–e47, 2015. Citado 2 vezes nas páginas 7 e 35.

ROBINSON, M. D.; MCCARTHY, D. J.; SMYTH, G. K. edger: a bioconductor package for differential expression analysis of digital gene expression data. *Bioinformatics*, Oxford University Press, v. 26, n. 1, p. 139–140, 2010. Citado na página 7.

STAICULESCU, M. C. et al. Comparative gene array analyses of severe elastic fiber defects in late embryonic and newborn mouse aorta. *Physiological genomics*, American Physiological Society Bethesda, MD, v. 50, n. 11, p. 988–1001, 2018. Citado na página 68.

TIWARI, V. K. et al. Target genes of topoisomerase iiβ regulate neuronal survival and are defined by their chromatin state. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 109, n. 16, p. E934–E943, 2012. Citado na página 67.

UNO, M. et al. A fasting-responsive signaling pathway that extends life span in c. elegans. *Cell reports*, Elsevier, v. 3, n. 1, p. 79–91, 2013. Citado na página 67.

WANG, Z.; GERSTEIN, M.; SNYDER, M. Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, Nature Publishing Group, v. 10, n. 1, p. 57–63, 2009. Citado na página 2.

WICKHAM, H.; CHANG, W. *Devtools: Tools to make developing R packages easier*. 2016. 9000 p. Disponível em: <https://cran.r-project.org/package=devtools>. Citado na página 33.

WICKHAM, H.; DANENBERG, P.; EUGSTER, M. *roxygen2: in-line documentation for R*. 2017. Disponível em: <https://cran.r-project.org/package=roxygen2>. Citado na página 33.

YANAI, I. et al. Pairing of competitive and topologically distinct regulatory modules enhances patterned gene expression. *Molecular systems biology*, John Wiley & Sons, Ltd Chichester, UK, v. 4, n. 1, p. 163, 2008. Citado na página 67.

ZIEMANN, M.; EREN, Y.; EL-OSTA, A. Gene name errors are widespread in the scientific literature. *Genome biology*, BioMed Central, v. 17, n. 1, p. 1–3, 2016. Citado na página 24.

Apêndices

APÊNDICE A – Documentação técnica

A.1 Classes

`GEVAInput`

Armazena os valores de entrada de entrada do usuário. Possui duas matrizes principais de mesma dimensão ($m \times n$), sendo uma para valores de expressão diferencial e outra para pesos.

`values` matriz numérica ($m \times n$) com valores de $logFC$

`weights` matriz numérica ($m \times n$) com valores de pesos, geralmente valor-p

`factors` `factor` (n elementos, zero a n níveis) indicando os fatores (grupos de resultados)

`ftable` `data.frame` (m linhas) com atributos adicionais de sondas ou genes

`info` lista de informações adicionais de *input*

`SVTable`

Armazena uma matriz ($m \times 2$) com as colunas `S` (para *sumário*) e `V` (para *variação*).

`sv` matriz ($m \times 2$) de colunas `S` e `V`. Pode ser numérica ou textual

`GEVASummary`

Contém os resultados de sumarização e variação calculados a partir de um `GEVAInput`. Essa classe é derivada de `SVTable`.

`sv` matriz numérica ($m \times 2$) de colunas `S` e `V` (herdado de `SVTable`)

`inputdata` `GEVAInput` com os dados de *input* usados para o cálculo

`sv.method` Atributos de texto (`SVChrAttribute`) contendo os nomes das funções usadas para obter o sumário (`S`) e variação (`V`)

`info` Lista com informações adicionais de sumarização

`SVAttribute`

Estoca dois valores do mesmo tipo, um no campo `S` e outro no campo `V`. Essa é uma classe abstrata e serve de base para as classes `SVChrAttribute` (onde `S` e `V` são `character`), `SVNumAttribute` (onde `S` e `V` são `numeric`) e `SVIntAttribute` (onde `S` e `V` são `integer`).

`S` Valor de `S`

`V` Valor de `V`

`GEVAGroupSet`

Representa os resultados de classificação de valores sumarizados em uma `SVTable`, onde cada gene/sonda foi incluso em um dentre g grupos definidos. Essa é uma classe abstrata e serve de base para as classes `GEVAQuantiles` e `GEVACluster`.

`grouping` `factor` (m elementos e g níveis) indicando o grupo de cada sonda/gene

`scores` vetor numérico (tamanho m) indicando o *score* de grupo para cada sonda ou gene

`ftable` `data.frame` (m linhas) com resultados adicionais de agrupamento

`centroids` `SVTable` numérica (g linhas) com as posições S e V dos centroides de cada grupo

`offsets` `SVTable` numérica (m linhas) com as distâncias S e V aos dados de origem

`info` Lista com informações adicionais de agrupamento

`GEVAQuantiles`

Representa uma classificação determinada por quantis. Para cada sonda/gene, é atribuído um quantil dentre g quantis definidos. Essa classe é derivada da classe `GEVAGroupSet`, e também serve de base para a classe `GEVAQuantilesAdjusted`, a qual inclui valores ajustados.

`...` (campos herdados de `GEVAGroupSet`)

`svscores` `SVTable` numérica (m linhas) com *scores* parciais para os quantis

`qareasizes` `SVTable` numérica (g linhas), contendo os comprimentos `S` e `V` de cada quantil

`qindexes` `SVTable` de inteiros (g linhas), contendo o índice de posicionamento de cada quantil nos eixos `S` e `V`

`qcount` atributos inteiros (`SVIntAttribute`) indicando a quantidade de partições `S` e `V` feitas para gerar $S \times V$ quantis

`qcutoff` atributos numéricos (`SVNumAttribute`) indicando o ponto de corte inicial para geração dos quantis nos eixos `S` e `V`

`GEVACluster`

Representa uma classificação determinada por clusterização. Para cada sonda/gene, é atribuído um *cluster* dentre g *clusters* definidos. Essa classe é derivada da classe `GEVAGroupSet`.

`...` (campos herdados de `GEVAGroupSet`)

`cluster.method` `character` único indicando o método (algoritmo) de clusterização aplicado

`TypedList`

Lista de itens de um mesmo tipo. Pode conter objetos derivados de uma classe-base em comum. Essa classe é derivada da classe `List`, nativa do R.

`elem.class` `character` indicando o nome da classe que representa os elementos da lista

`GEVAResults`

Contém os principais resultados das análises com o GEVA, concatenando dados de sumarização, clusterização, quantis e fatores de variância.

`resultstable` `data.frame` (m linhas) com a classificação final para os genes/sondas

`svdata` `GEVASummary` usado como *input*

`quantdata` `GEVAQuantiles` ou `GEVAQuantilesAdjusted` contendo a definição final de quantis para os dados sumarizados

`factoring` `data.frame` (m linhas) contendo resultados parciais das análises de fatores, caso tenha-se incluído esse tipo de análise

`classiftable` `data.frame` usado para relacionar diferentes classificações

A.2 Funções

`geva.read.tables`

```
> geva.read.tables(filenames=NULL, dirname=".",
                   col.values="logFC", col.pvals="adj.P.Val",
                   col.other=NULL, files.pattern = "\\.txt$")
```

Lê um conjunto de arquivos de tabela de resultados de expressão diferencial, extrai as colunas de *logFC* de cada tabela e as concatena em uma única tabela. Se uma coluna de valor-p ou valor-p adjustado estiver presente, será processada e concatenada para a tabela de pesos.

`filenames` (vetor `character`) nomes dos arquivos de tabela, podendo ser endereços absolutos ou relativo ao diretório `dirname`. Se vazio ou não especificado, busca todos os arquivos em `dirname` que satisfaçam o padrão de `files.pattern`

`dirname` (`character` único) diretório dos arquivos de entrada. Se não especificado, usa o diretório atual (`getwd()`)

`col.values` (vetor `character`) nomes de colunas representando valores de *logFC*. Se uma tabela possuir mais de um nome correspondente, a primeira coluna encontrada será utilizada

`col.pvals` (vetor `character`) nomes de colunas representando valores de valor-p. Se uma tabela possuir mais de um nome correspondente, a primeira coluna encontrada será utilizada

`col.other` (vetor `character`) nomes de colunas adicionais a serem armazenadas

`files.pattern` (`character` único) padrão de expressão regular para filtrar quais arquivos em um diretório serão levados em conta, caso `filenames` não tenha sido indicado. Este argumento deve ser indicado explicitamente.

Retorna: objeto `GEVAInput` com os dados processados e concatenados

`geva.merge.input`

```
> geva.merge.input(..., col.values="logFC",
                   col.pvals="adj.P.Val", col.other=NULL)
```

Unifica múltiplas tabelas em um único objeto `GEVAInput`, extraindo as colunas de *logFC* e de pesos quando presentes. Nomes de linha correspondentes são combinados e reorganizados caso necessário.

`...` dois ou mais objetos `matrix`, `data.frame` ou `GEVAInput`. Vetores numéricos são convertidos para colunas de *logFC* sem pesos e anexados à tabela. Listas são exploradas recursivamente. Outros tipos de objetos são ignorados;

`col.values` (vetor `character`) nomes de colunas representando valores de $logFC$. Se uma tabela possuir mais de um nome correspondente, a primeira coluna encontrada será utilizada

`col.pvals` (vetor `character`) nomes de colunas representando valores de valor-p. Se uma tabela possuir mais de um nome correspondente, a primeira coluna encontrada será utilizada

`col.other` (vetor `character`) nomes de colunas adicionais a serem armazenadas.

Retorna: objeto `GEVAInput` com os dados processados e concatenados

`geva.ideal.example`

```
> geva.ideal.example(probecount=10000, condcount=3, respercond=3, seed=NA)
```

Cria um `GEVAInput` com valores aleatórios de $logFC$ e pesos. Os títulos das linhas serão nomes genéricos para sondas (como `probe_001`, `probe_002`, etc), e grupos de colunas terão fatores atribuídos. Apesar dos valores em si serem gerados aleatoriamente, as distribuições dos valores simularão um dado experimental contendo todos os tipos de classificações. Recomenda-se utilizar esta função apenas para testes.

`probecount` (`numeric` único) quantidade total de sondas (linhas)

`condcount` (`integer` único) número de fatores criados. O mínimo é 1

`respercond` (`integer` único) número de colunas para cada fator

`seed` (`integer` único) se especificado, usa este número como ponto de partida para geração dos valores aleatórios. Se não especificado, o `seed` é obtido automaticamente pelo relógio do sistema

Retorna: objeto `GEVAInput` com dados idealmente gerados. O tamanho final de cada tabela será `probecount` linhas por `condcount * respercond` colunas

`geva.input.correct`

```
> geva.input.correct(ginput, na.rm=TRUE, inf.rm=TRUE, invalid.col.rm=TRUE)
```

Remove linhas contendo valores inválidos dentro de um *GEVAInput*. Se especificado, também remove linhas onde todos os valores de valor-p estão abaixo de um certo ponto de corte.

`ginput` objeto `GEVAInput`

`na.rm` (`TRUE/FALSE`) se `TRUE`, elimina linhas com valores `NA`

`inf.rm` (`TRUE/FALSE`) se `TRUE`, remove linhas com valores infinitos

`invalid.col.rm` (`TRUE/FALSE`) se `TRUE`, remove colunas onde todos seus valores foram marcados para remoção (`NA` por `na.rm` e infinitos por `inf.rm`)

Retorna: `GEVAInput` corrigido

geva.input.rename.rows

```
> geva.input.rename.rows(ginput, attr.column,
                         dupl.rm.method = c("least.p.vals", "order"))
```

Renomeia os nomes das linhas de acordo com uma coluna de atributos especificada. Valores vazios são ignorados, e valores duplicados são removidos de acordo com o parâmetro `dupl.rm.method`.

`ginput` objeto `GEVAInput`

`attr.column` (vetor `character`) se indicado um valor único, busca uma coluna da tabela adicional (campo `@ftable` de `GEVAInput`) com o nome correspondente e utiliza seus valores. Se especificado um vetor de tamanho igual ao mesmo número de linhas da tabela, usa esses valores como nomes de linhas

`dupl.rm.method` (`character` único) método de remoção de valores duplicados, podendo ser `"least.p.vals"` para manter a duplicada com menor valor-p, ou `"order"` para remover duplicatas de acordo com seu ordenamento no vetor

Retorna: `GEVAInput` contendo os novos nomes de linha

geva.summarize

```
> geva.summarize(gevainput, summary.method, variation.method)
```

Realiza a sumarização dos valores de um objeto `GEVAInput` de acordo com o parâmetro de sumário e variação. Se os pesos estiverem definidos na matriz `weights` do objeto, a versão ponderada dos métodos especificados será aplicada.

`ginput` objeto `GEVAInput`

`summary.method` (`character` único) método de cálculo da sumarização, podendo ser `"mean"` (média) ou `"median"` (mediana)

`variation.method` (`character` único) método de cálculo da variação, podendo ser `"sd"` (desvio-padrão), `"var"` (variância) ou `"mad"` (desvio mediano absoluto)

Retorna: objeto `GEVASummary` com os valores de sumarização e variação

geva.quantiles

```
> geva.quantiles(sv, nq.s = 3L, nq.v = 2L,
                 initial.thresholds=c(S=NA_real_, V=NA_real_),
```

```
                comb.score.fn = prod)
```

Calcula os quantis de um objeto `SVTable` numérico, geralmente um `GEVASummary`.

`sv` objeto `SVTable`

`nq.s` (`integer` único), número de quantis no eixo sumário (`S`)

`nq.v` (`integer` único), número de quantis no eixo variação (`V`)

`initial.thresholds` (vetor `numeric` de tamanho 2), indica os pontos de corte iniciais de cada quantil, a partir do quantil mais indiferencial. Se não especificado, o ponto de corte de um quantil será seu maior valor absoluto dividido pelo número de quantis definidos, tendo como resultado quantis de mesmo tamanho

`comb.score.fn` função usada para combinar a pontuação dos diferentes quantis em uma única pontuação. As funções `prod` (produtório) e `mean` (média) são alguns exemplos

Retorna: objeto `GEVAQuantiles` com o mapeamento de quantis

`geva.cluster`

```
> geva.cluster(sv, cluster.method, cl.score.method, resolution=0.3, ...)
```

Realiza uma análise de clusterização a partir dos valores de uma `SVTable` numérica, geralmente um objeto `GEVASummary`.

`sv` objeto `SVTable`

`cluster.method` (`character` único) método de clusterização a ser aplicado. São aceitas as opções `"hierarchical"` para hierárquica e `"density"` para densidade;

`cl.score.method` (`character` único) método de obtenção dos *scores* da clusterização resultante. O cálculo da pontuação pode ser feito de acordo com a posição hierárquica de um ponto (`"hclust.height"`), pela densidade de cada ponto (`"density"`) ou pela posição relativa de um ponto ao centroide de seu *cluster* (`"centroid"`). A opção `"auto"` escolhe hierarquia ou densidade dependendo da análise de clusterização realizada;

`resolution` (`numeric` único), valor de 0 a 1 indicando a resolução da clusterização. Quanto maior a resolução, maior o número de *clusters* resultantes. Um valor zero resulta em um único *cluster*, enquanto o valor 1 resulta no número máximo de *clusters* detectáveis.

`...` parâmetros adicionais para as funções de clusterização subjacentes.

geva.hcluster

```
> geva.hcluster(sv, resolution=0.3, hc.method, hc.metric)
```

Realiza uma clusterização hierárquica a partir dos valores de uma `SVTable` numérica, geralmente um objeto `GEVASummary`.

`sv` objeto `SVTable`

`resolution` (`numeric` único), valor de 0 a 1 indicando a resolução da clusterização. Quanto maior a resolução, maior o número de *clusters* resultantes. Um valor zero resulta em um único *cluster*, enquanto o valor 1 resulta em um *cluster* para cada ponto.

`cl.score.method` (`character` único) ver geva.cluster;

`hc.method` (`character` único), método de definição de hierarquias para formação um *cluster*, podendo ser `"centroid"` (centróide), `"median"` (ponto mediano), `"ward"` (método *Ward*), ou `"single"` (distância arbitrária)

`hc.metric` (`character` único), métrica para calcular distância entre pontos, podendo ser `"euclidean"` (distância euclidiana), `"manhattan"` (distância de Manhattan), `"maximum"` (distância máxima), `"canberra"` (fórmula de Canberra), `"binary"` (distância binária), ou `"minkowski"` (fórmula de Minkowski)

geva.dcluster

```
> geva.dcluster(sv, resolution=0.3, hc.method, hc.metric)
```

Realiza uma clusterização hierárquica a partir dos valores de uma `SVTable` numérica, geralmente um objeto `GEVASummary`.

`sv` objeto `SVTable`

`resolution` (`numeric` único), valor de 0 a 1 indicando a resolução da clusterização. Quanto maior a resolução, maior o número de *clusters* resultantes. Um valor zero resulta em um único *cluster*, enquanto o valor 1 resulta em um *cluster* para cada ponto.

`dcluster.method` (`character` único), método de detecção da densidade de *clusters*, podendo ser `"dbscan"` ou `"optics"`. Na implementação atual, os resultados das opções são virtualmente similares, com raras diferenças em pontos discrepantes;

`cl.score.method` (`character` único) ver geva.cluster

geva.finalize

```
> geva.finalize(gsummary, ..., p.value=0.05)
```

Concatena todos os resultados e, se especificado os fatores, realiza análises de variância. O *score* final dos quantis será ajustado baseado em cada resultado de clusterização indicado em `...`. Se não estiver especificado nenhum resultado para quantis, eles serão gerados pela função `geva.quantiles` utilizando os argumentos-padrão.

`gsummary` objeto `GEVASummary`

`...` zero ou mais objetos derivados de `GEVAGroupSet` a serem levados em conta, podendo ser resultados de clusterização (`GEVACluster`) ou quantis (`GEVAQuantiles`)

`p.value` ponto de corte para valor-p usado em análises com fatores. Este argumento deve ser indicado explicitamente

Retorna: objeto `GEVAResults`, contendo as classificações finais das sondas ou genes sendo analisados

geva.quick

```
> geva.quick(ginput, ...)
```

Realiza todas as etapas de análise do GEVA (sumarização, detecção de quantis, clusterização e concatenação dos resultados) em uma única função.

`ginput` objeto `GEVAInput`. Alternativamente, pode ser um `GEVASummary` ou um `GEVAResults`, o que resultará na re-análise com os parâmetros de origem, porém sobrepondo estes com os parâmetros indicados pelo usuário;

`...` parâmetros para as funções geva.summarize, geva.quantiles, geva.cluster e geva.finalize.

Retorna: objeto `GEVAResults`, contendo as classificações finais das sondas ou genes sendo analisados

top.genes

```
> top.genes(gevaresults, classif=c("similar",
                                   "factor-dependent", "factor-specific"),
            which.spec=levels(gevaresults), add.cols=NULL, names.only=FALSE)
```

Extrai os resultados relevantes obtidos no final de uma análise com o GEVA.

`gevaresults` objeto `GEVAResults`;

`classif` (vetor `character`) classificações correspondentes aos genes a serem extraídos. São aceitas uma ou mais opções entre `"similar"`, `"factor-dependent"`, `"factor-specific"`, `"basal"` (indiferencial) e `"sparse"` (irregular);

`which.spec` (vetor `character`) caso `"factor-specific"` tenha sido indicado em `classif`, filtra os genes que correspondem a este fator específico;

`field` add.cols (vetor `character`) inclui colunas adicionais na tabela retornada (*e.g.*, nome dos genes). Opção conveniente quando nomes de sondas são utilizados no lugar dos nomes de genes;

`field` names.only (`TRUE/FALSE`) se `TRUE`, retorna apenas os nomes de linha em vez da tabela.

Retorna: uma tabela tipo `data.frame` com uma coluna de classificação e outra indicando o nome do fator específico para genes com esta classificação; ou um vetor `character` caso a opção `names.only` tenha sido marcada como `TRUE`

APÊNDICE B – Protocolos auxiliares em R

B.1 Aplicação do teste de Levene

```
# Se o pacote 'car' não estiver instalado:
> install.packages("car")
# Carrega o pacote 'car'
> library(car)
# Define um vetor de valores:
> v <- c(0.84, 1.22, 1.32, -1.37, -2.17, -1.51, 4.23, 4.93, 3.92)
# Define os grupos (g1, g2 e g3):
> g <- c("g1", "g1", "g1", "g2", "g2", "g2", "g3", "g3", "g3")
# Converte grupos para factor (opcional)
> g <- as.factor(g)
# Teste de Levene:
> result <- leveneTest(v ~ g)
> result
# Resultado impresso na tela:
# Levene's Test for Homogeneity of Variance (center = median)
#       Df  F value  Pr(>F)
# group  2  0.2519   0.7851
#        6
#
> result[["Pr(>F)"]][1]   # para obter apenas o valor-p
# Resultado:
# 0.7851379
```

B.2 Exemplo de expressão diferencial por microarranjo

Instalação dos pacotes necessários:

```
# BiocManager
> install.packages("BiocManager")
# limma
> BiocManager::install("limma")
# GEOquery (para este exemplo)
> BiocManager::install("GEOquery")
```

Obtenção da matriz de expressão gênica (exemplo com o conjunto de dados GSE39009):

```
# Carrega o pacote necessário
> library(GEOquery)
# Baixa e processa o conjunto de dados GSE39009
> eset <- getGEO('GSE39009')[[1]]
# Imprime o nome das colunas (opcional)
> colnames(eset)
# Resultado impresso:
# [1] "GSM953748" "GSM953749" "GSM953750" "GSM953751"
#     "GSM953752" "GSM953753" "GSM953754"
```

Análise de expressão diferencial com o *limma*:

```
# Carrega o pacote necessário
> library(limma)
# Extrai a matriz principal do conjunto de dados
> eset.mat <- exprs(eset)
# Aplica log2 caso os dados não estejam em escala log
> eset.mat <- log2(eset.mat)
# Define o grupo experimental
# Neste caso: GSM953748, GSM953749 e GSM53750
> gexp <- colnames(eset.mat)[1:3]
# Define o grupo controle
# Neste caso: GSM953751, GSM953752, GSM53753 e GSM53754
> gctrl <- colnames(eset.mat)[4:7]
# Define matriz de design
> mdesign <- ifelse(colnames(eset.mat) %in% gexp, 1, 2)
# Define design dos grupos
> design.groups <- model.matrix(~ 0+factor(mdesign))
> colnames(design.groups) <- c("g1","g2")
> rownames(design.groups) <- colnames(eset.mat)
# Cria um modelo linear a partir do design dos grupos
> agrupa.repls <- lmFit(eset.mat, design.groups)
# Cria uma matriz de contraste usando o modelo linear
> contrast.mat <- makeContrasts(g2-g1, levels=design.groups)
# Computa os coeficientes e erros a partir dos contrastes
> group.contrast <- contrasts.fit(agrupa.repls, contrast.mat)
```

```
# Obtém os genes diferencialmente expressos (eBayes)
> difexp <- eBayes(group.contrast)
# Extrai tabela dos genes diferencialmente expressos
> de1 <- topTable(difexp, coef=1, number=1000000, p.value=1,
                  sort.by="none", adjust.method="BH")
# Anexa símbolo dos genes (opcional)
> de1[, "Gene␣Symbol"] <- fData(eset)[,"Gene␣Symbol"]
# Imprime as colunas logFC, valor-p ajustado e nomes dos genes
> head(de1[,c("logFC", "adj.P.Val", "Gene␣Symbol")])
# Resultado impresso:
#                    logFC  adj.P.Val Gene Symbol
# 1415670_at     0.32438559 0.01007287       Copg1
# 1415671_at     0.11361803 0.15755852    Atp6v0d1
# 1415672_at     0.05806381 0.43649404      Golga7
# 1415673_at     0.10359302 0.26718123        Psph
# 1415674_a_at -0.22794373 0.05015130     Trappc4
# 1415675_at   -0.02194565 0.83032142        Dpm2
```

APÊNDICE C – Grade completa dos parâmetros de análise testados

C.1 C_{met}="density"; C_{res}=0.2

C.1.1 Clusterização

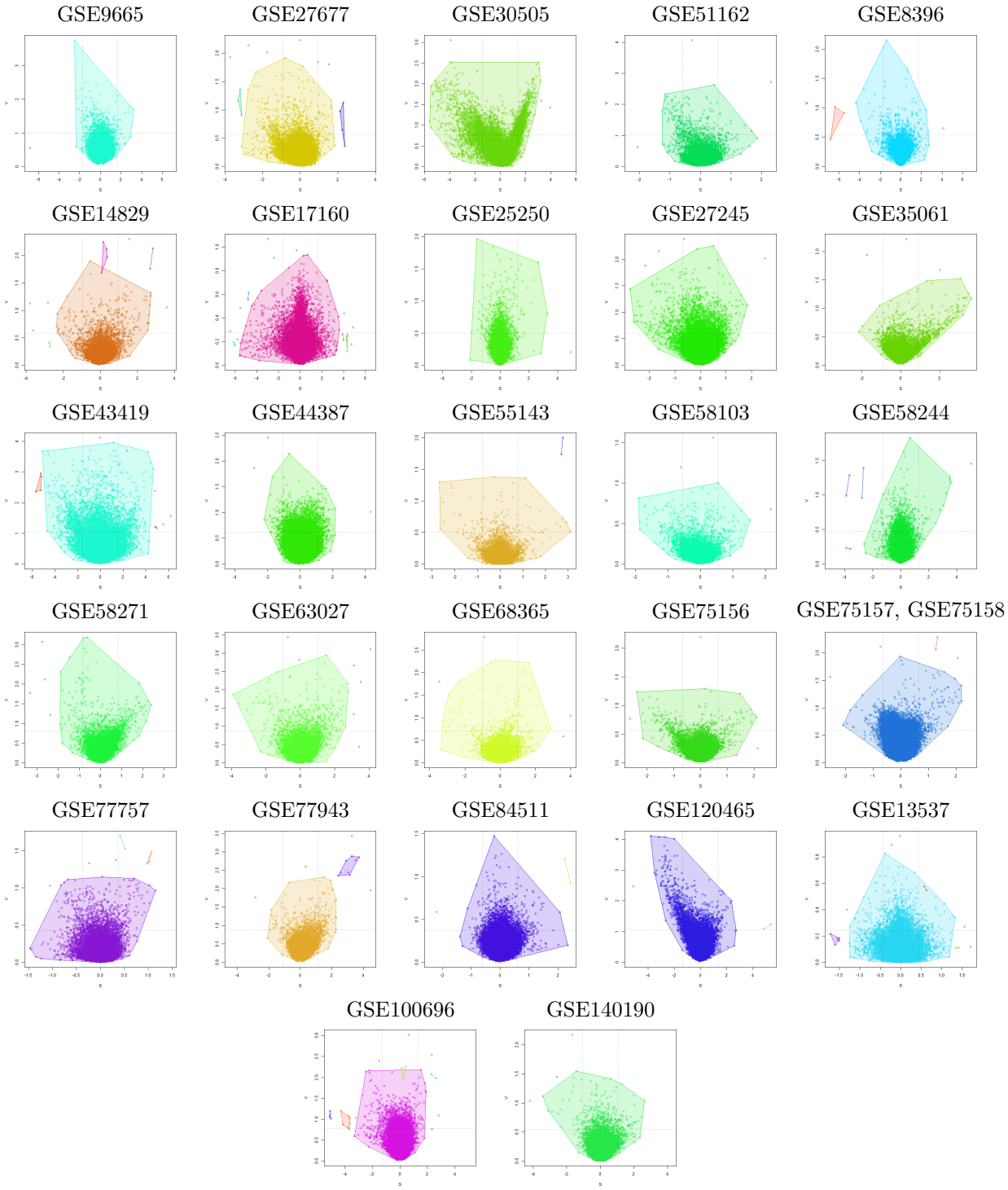

Figura C1 – Clusterização por densidade com 20% de resolução para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.1.2 Resultados

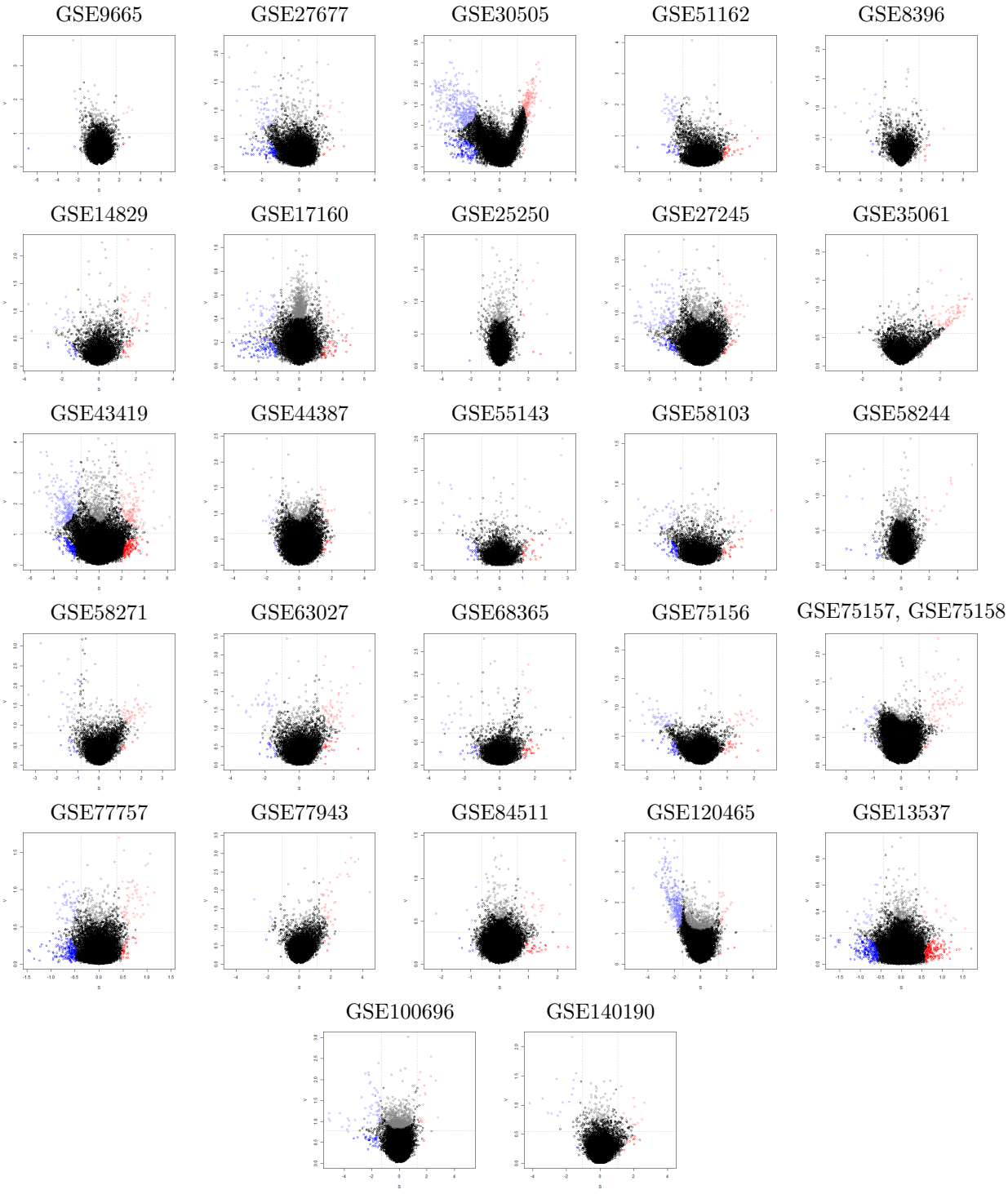

Figura C2 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização por densidade com 20% de resolução. Sua representação é equivalente à da Figura 18.

C.2 C_{met}="density"; C_{res}=0.3

C.2.1 Clusterização

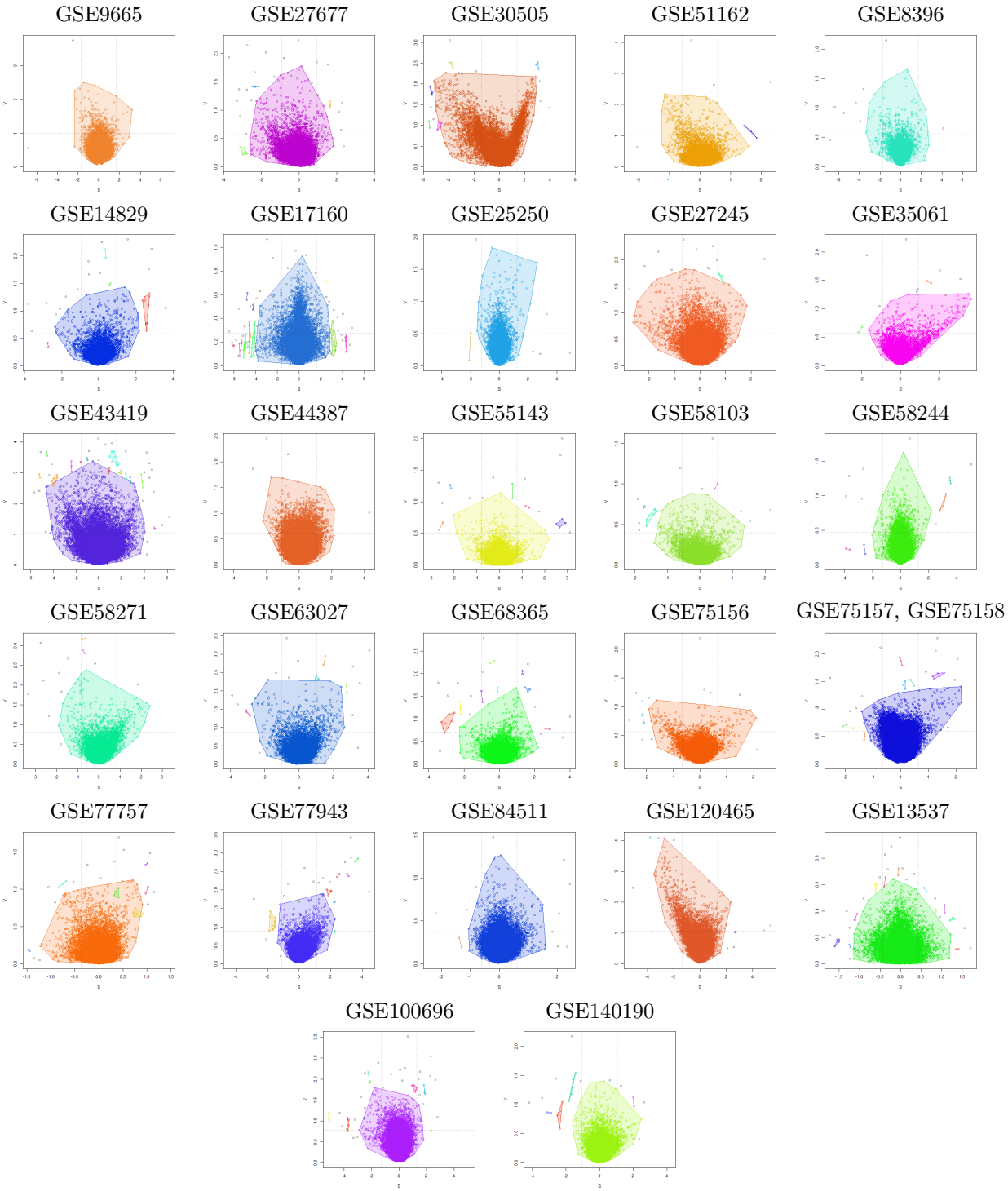

Figura C3 – Clusterização por densidade com 30% de resolução para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.2.2 Resultados

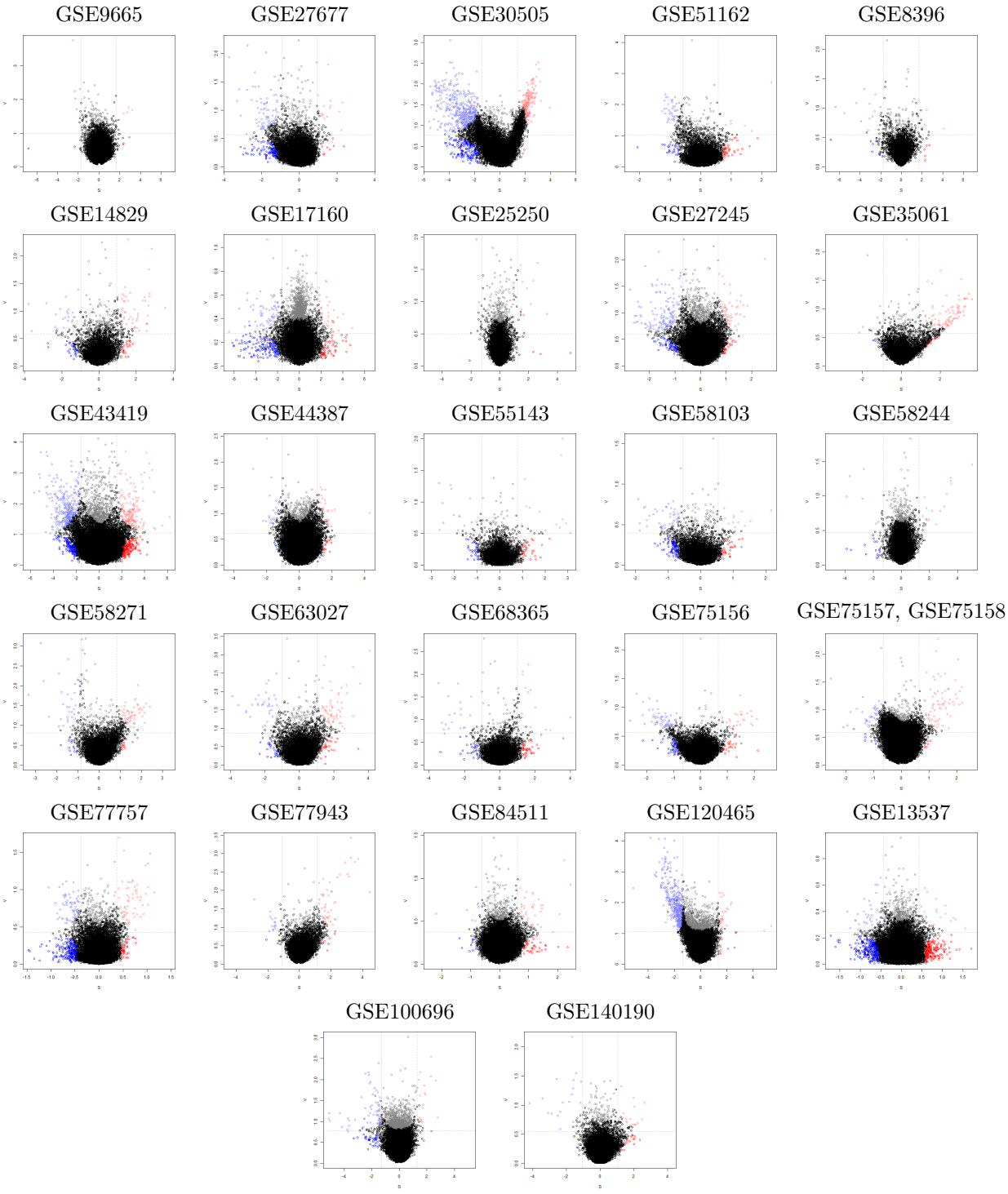

Figura C4 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização por densidade com 30% de resolução. Sua representação é equivalente à da Figura 18.

C.3 C_{met}="density"; C_{res}=0.4

C.3.1 Clusterização

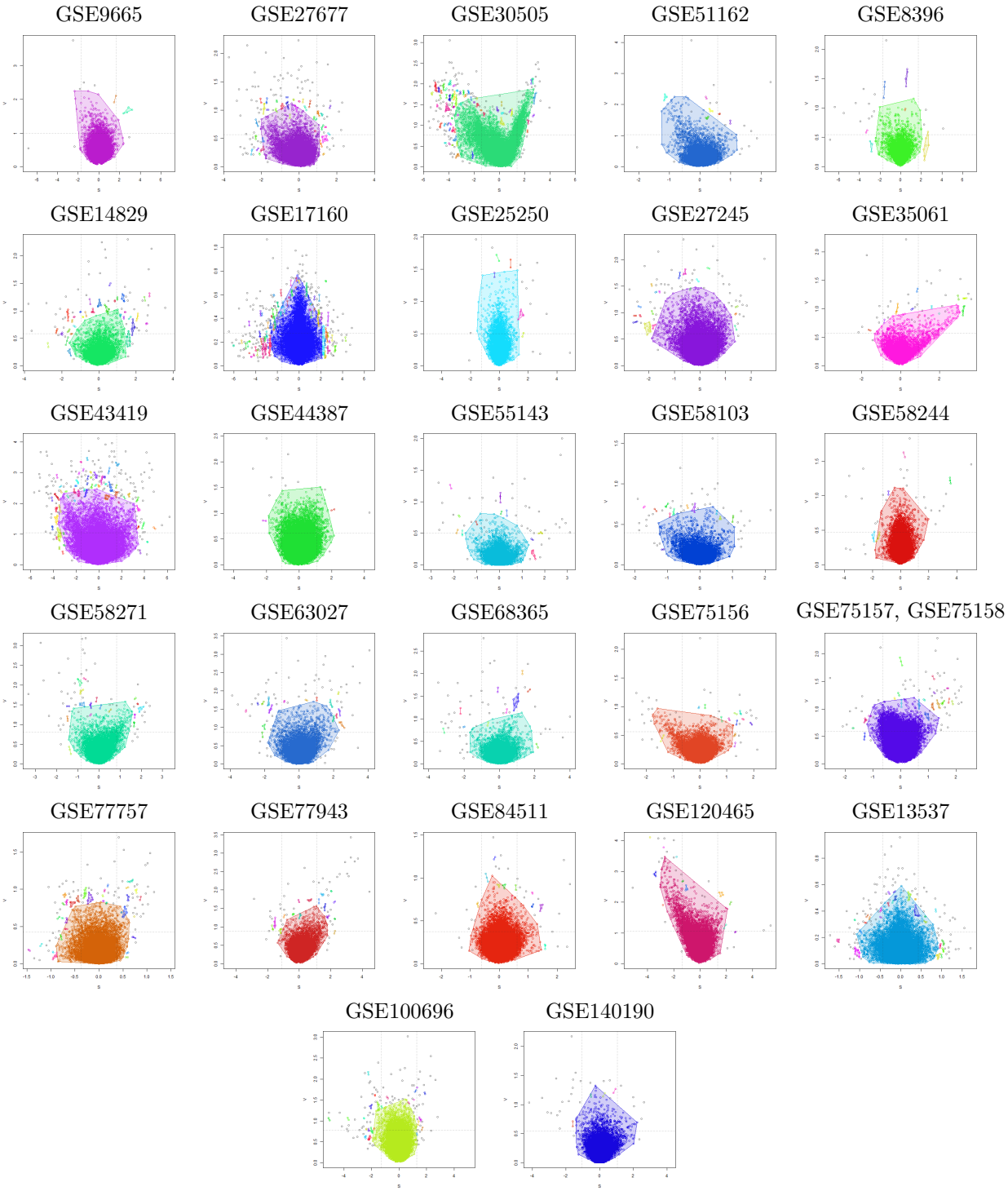

Figura C5 – Clusterização por densidade com 40% de resolução para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.3.2 Resultados

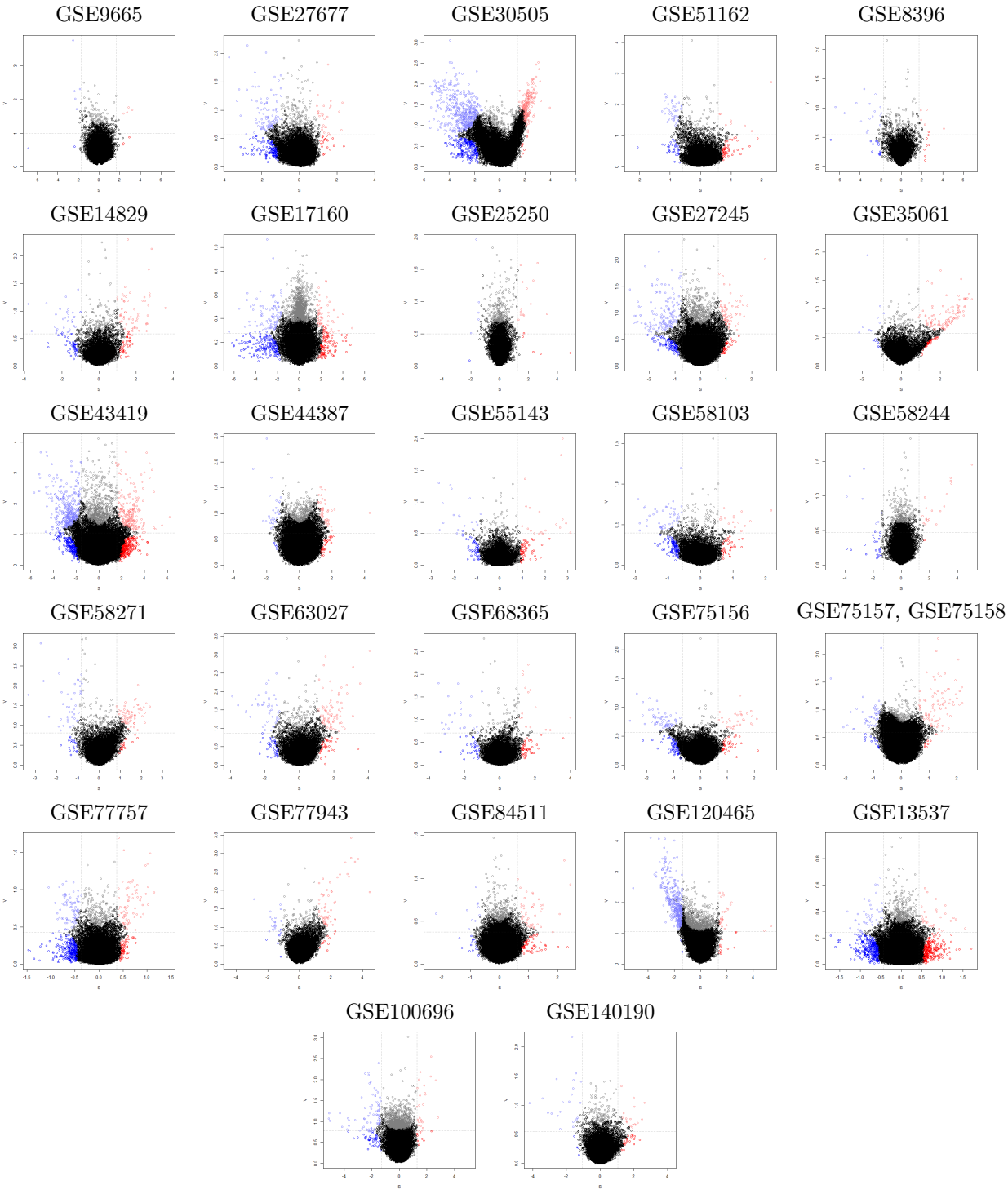

Figura C6 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização por densidade com 40% de resolução. Sua representação é equivalente à da Figura 18.

C.4 C_{met}="hierarchical"; C_{res}=0.2

C.4.1 Clusterização

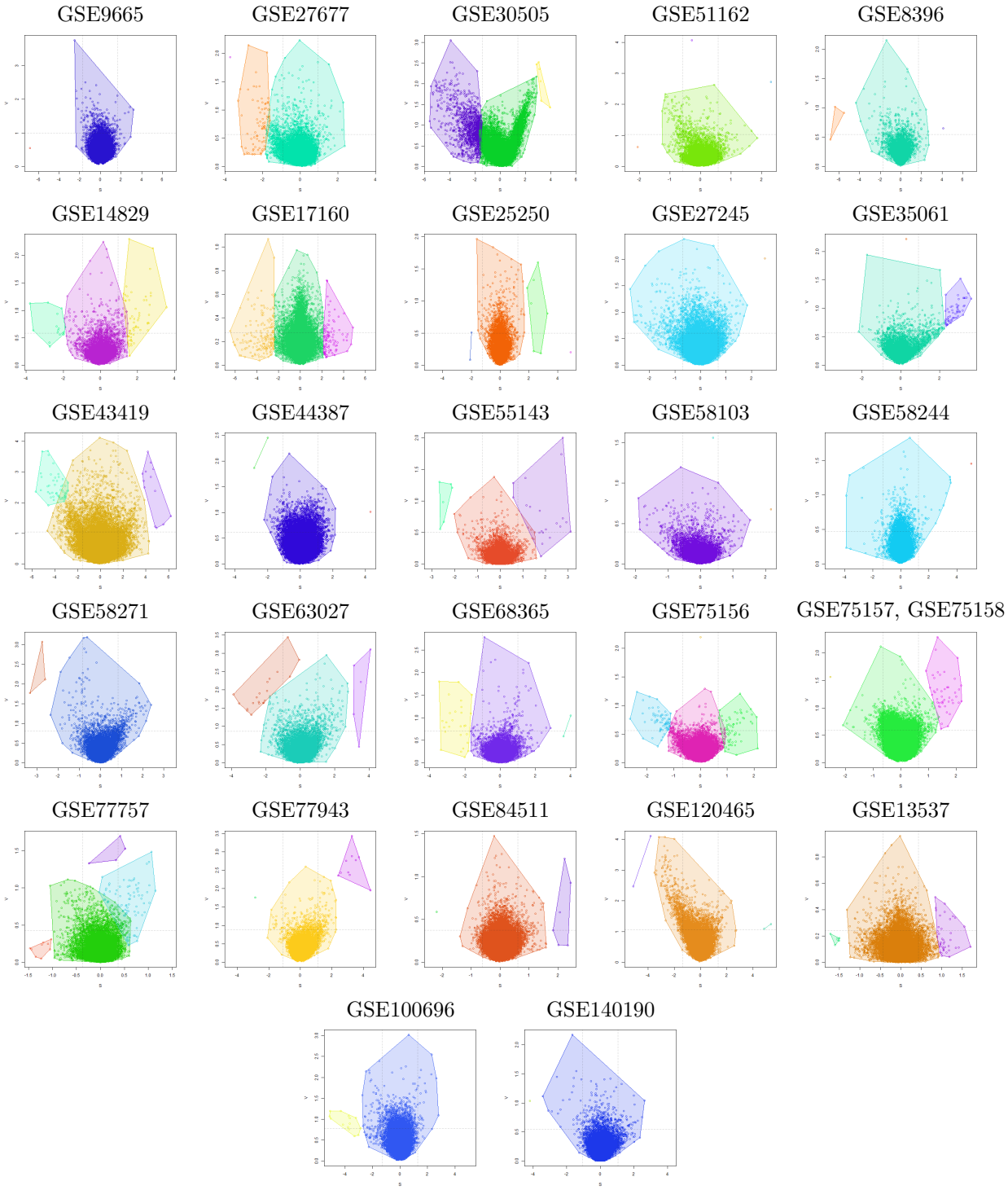

Figura C7 – Clusterização por hierárquica com 20% de resolução para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.4.2 Resultados

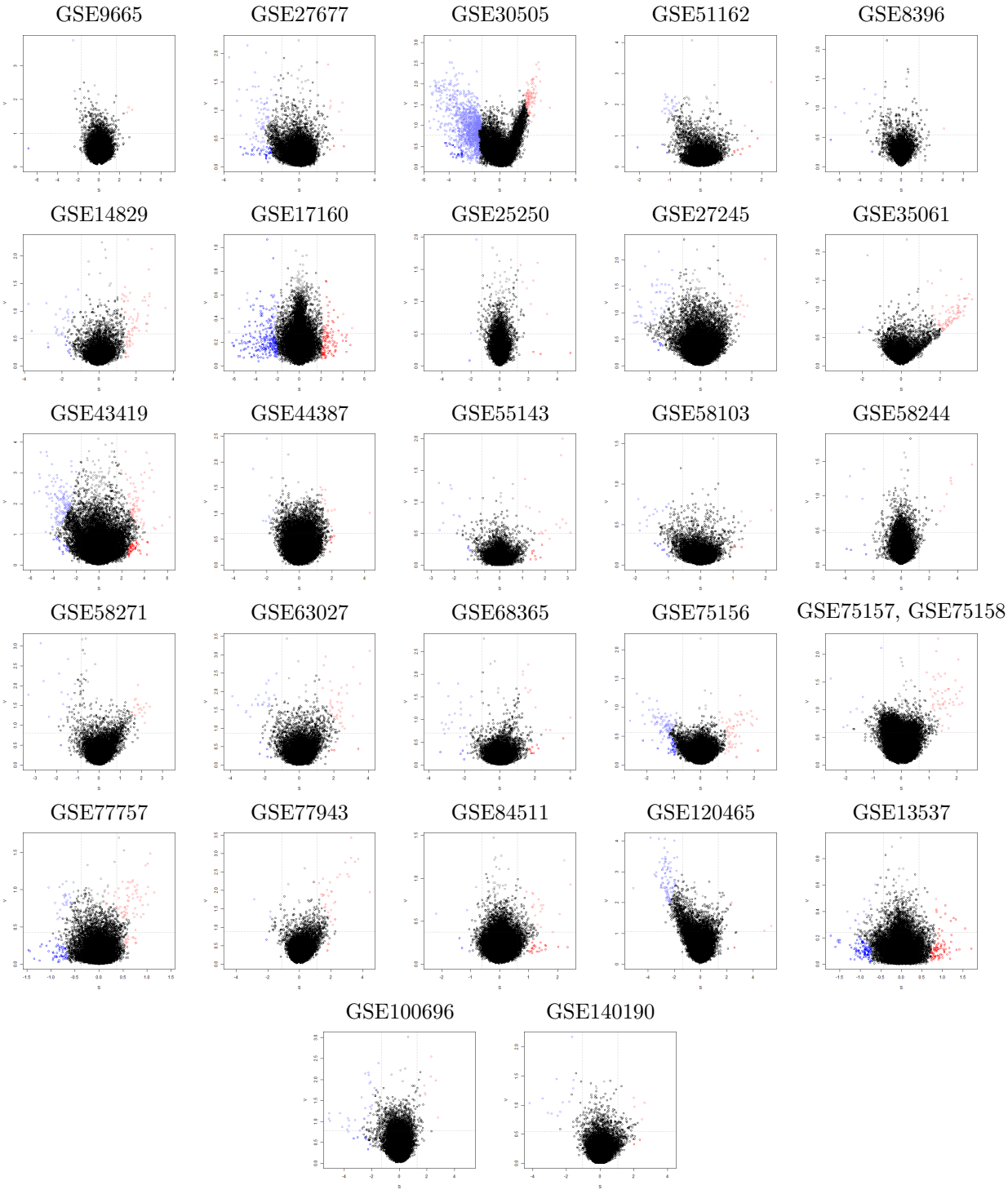

Figura C8 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização hierárquica com 20% de resolução. Sua representação é equivalente à da Figura 18.

C.5 C_{met}="hierarchical"; C_{res}=0.3 (Padrão)

C.5.1 Clusterização

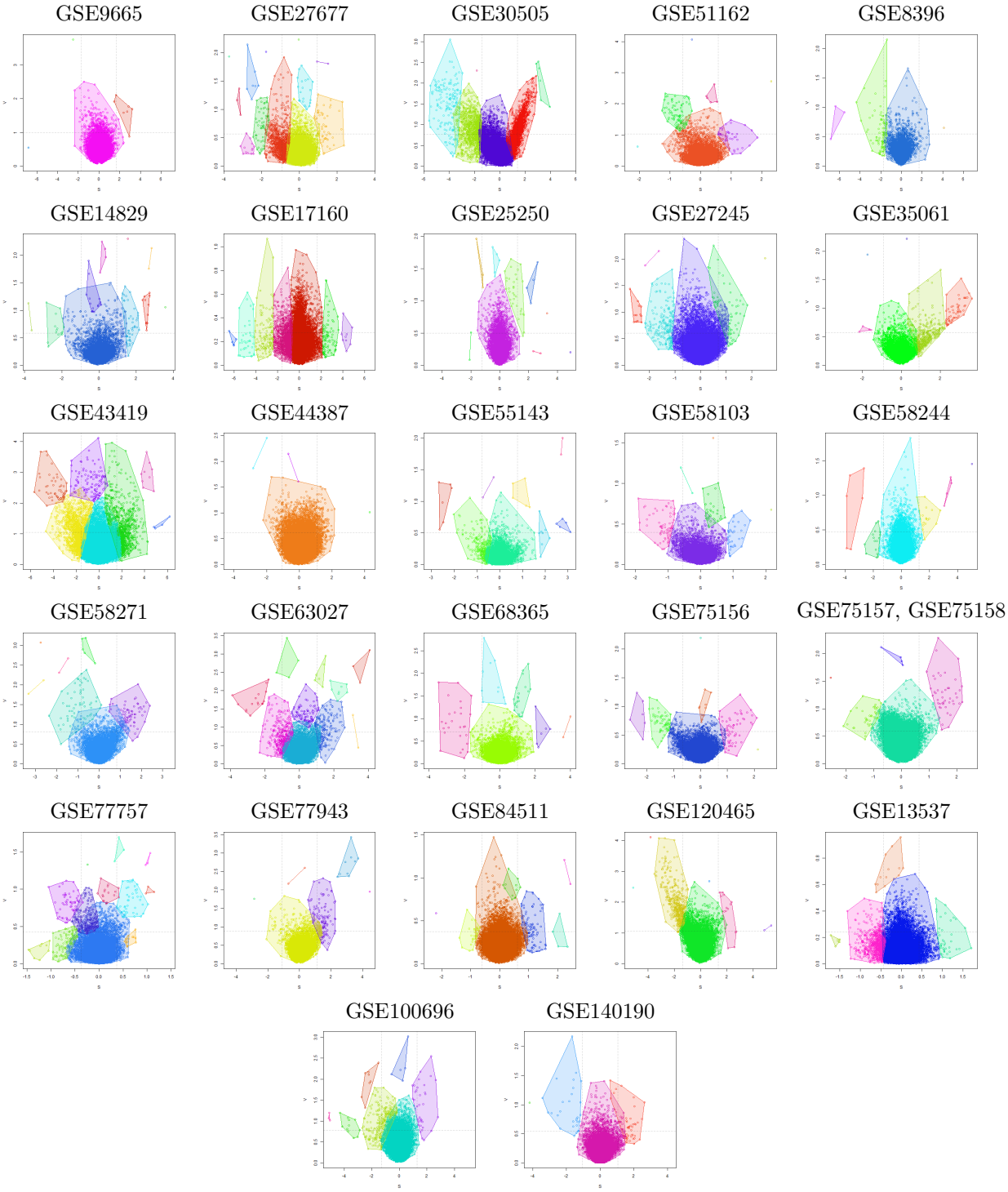

Figura C9 – Clusterização por hierárquica com 30% de resolução (valor padrão) para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.5.2 Resultados

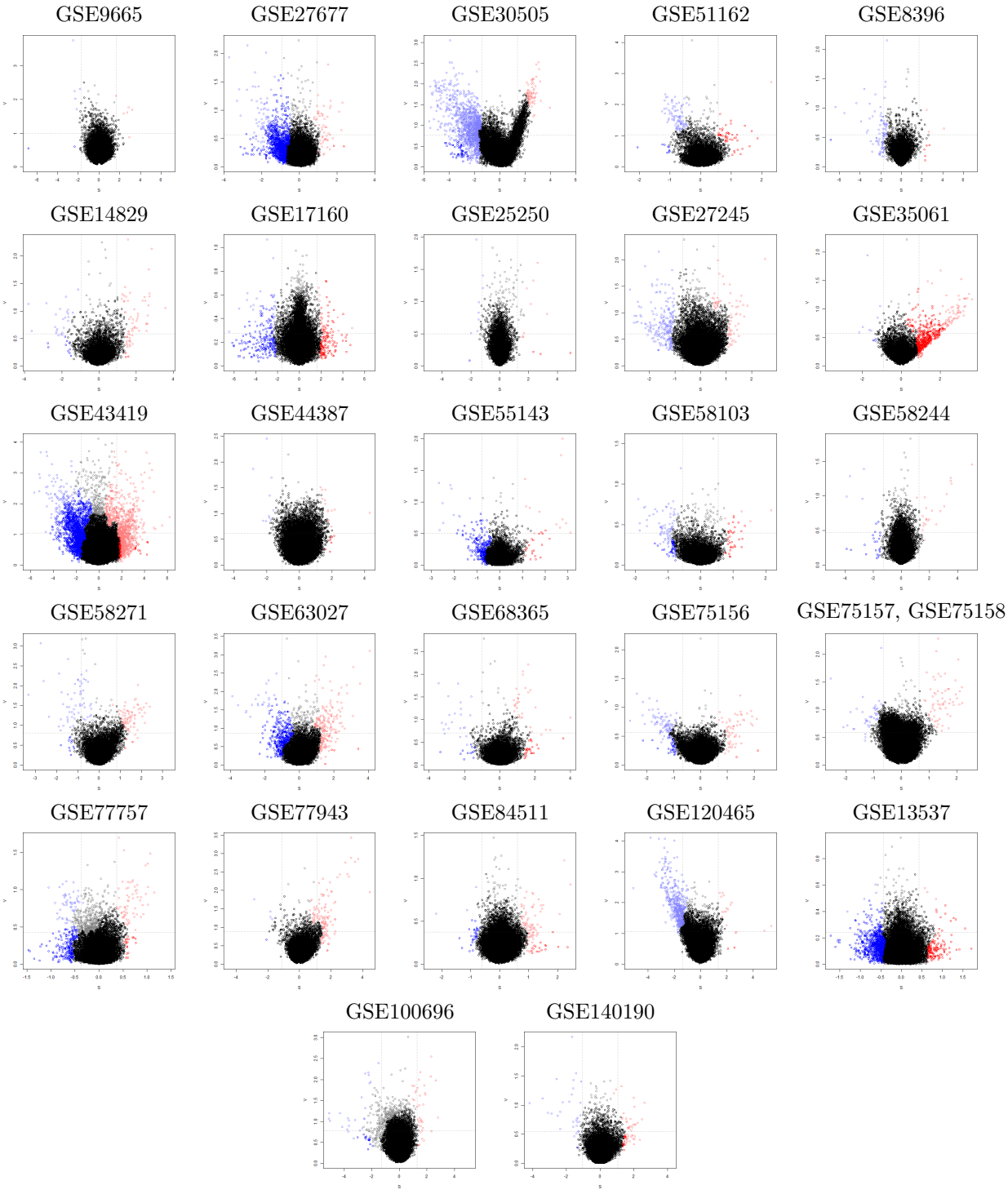

Figura C10 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização hierárquica com 30% de resolução. Sua representação é equivalente à da Figura 18.

C.6 C_{met}="hierarchical"; C_{res}=0.4

C.6.1 Clusterização

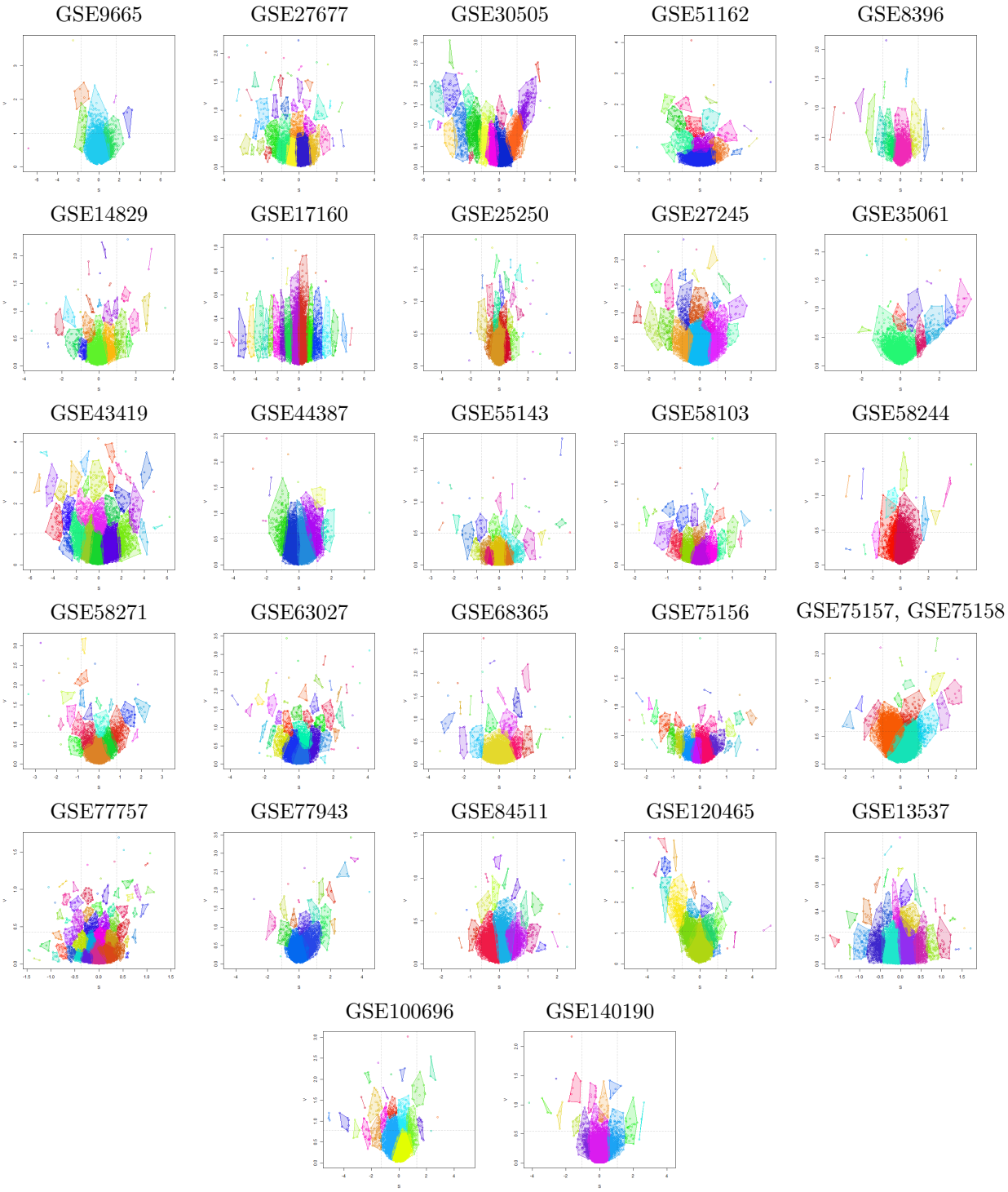

Figura C11 – Clusterização por hierárquica com 40% de resolução para os conjuntos de dados analisados, onde cada agrupamento de pontos é destacado por um polígono convexo de cor arbitrária.

C.6.2 Resultados

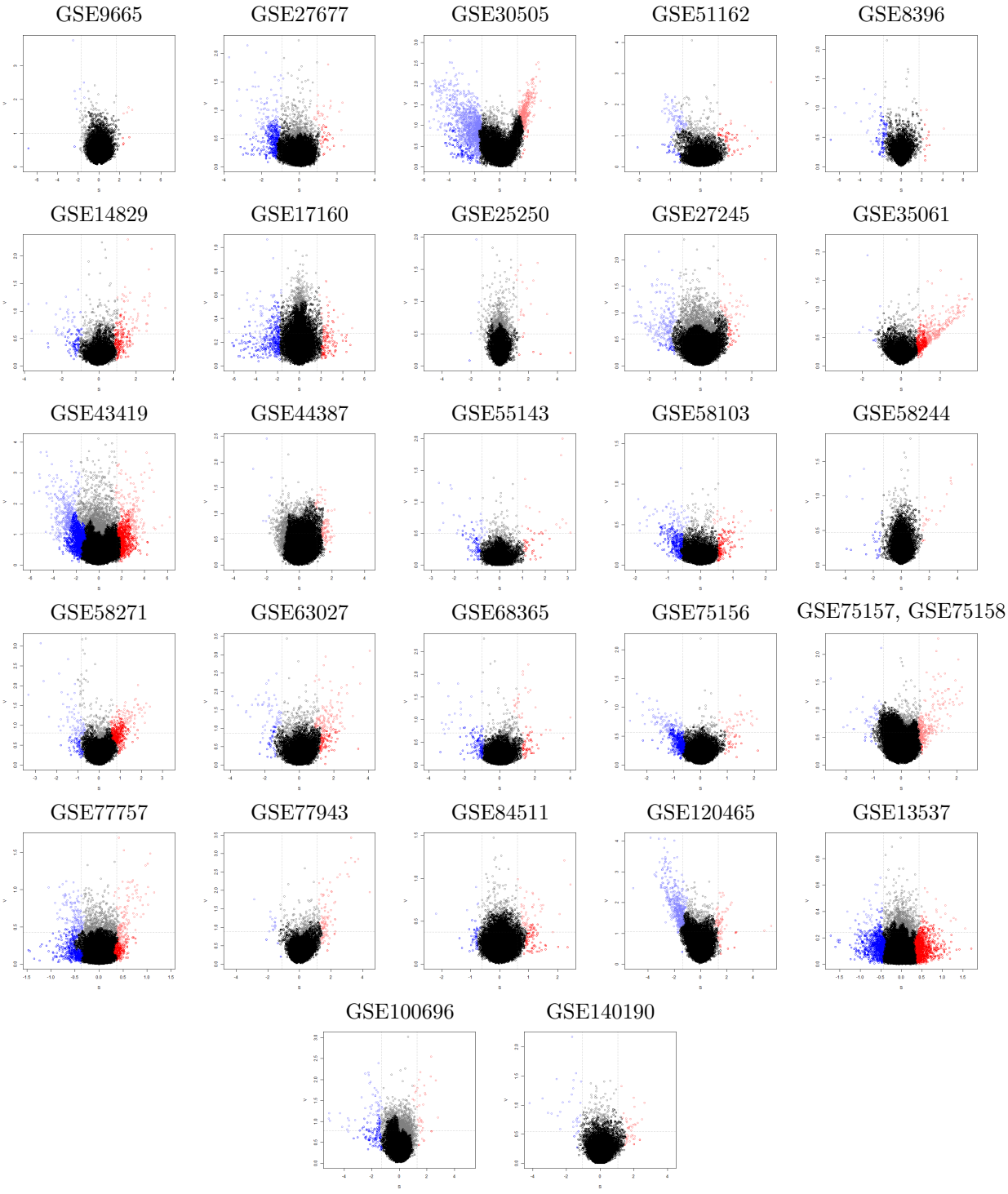

Figura C12 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando clusterização hierárquica com 40% de resolução. Sua representação é equivalente à da Figura 18.

C.7 Q_{met}="density"

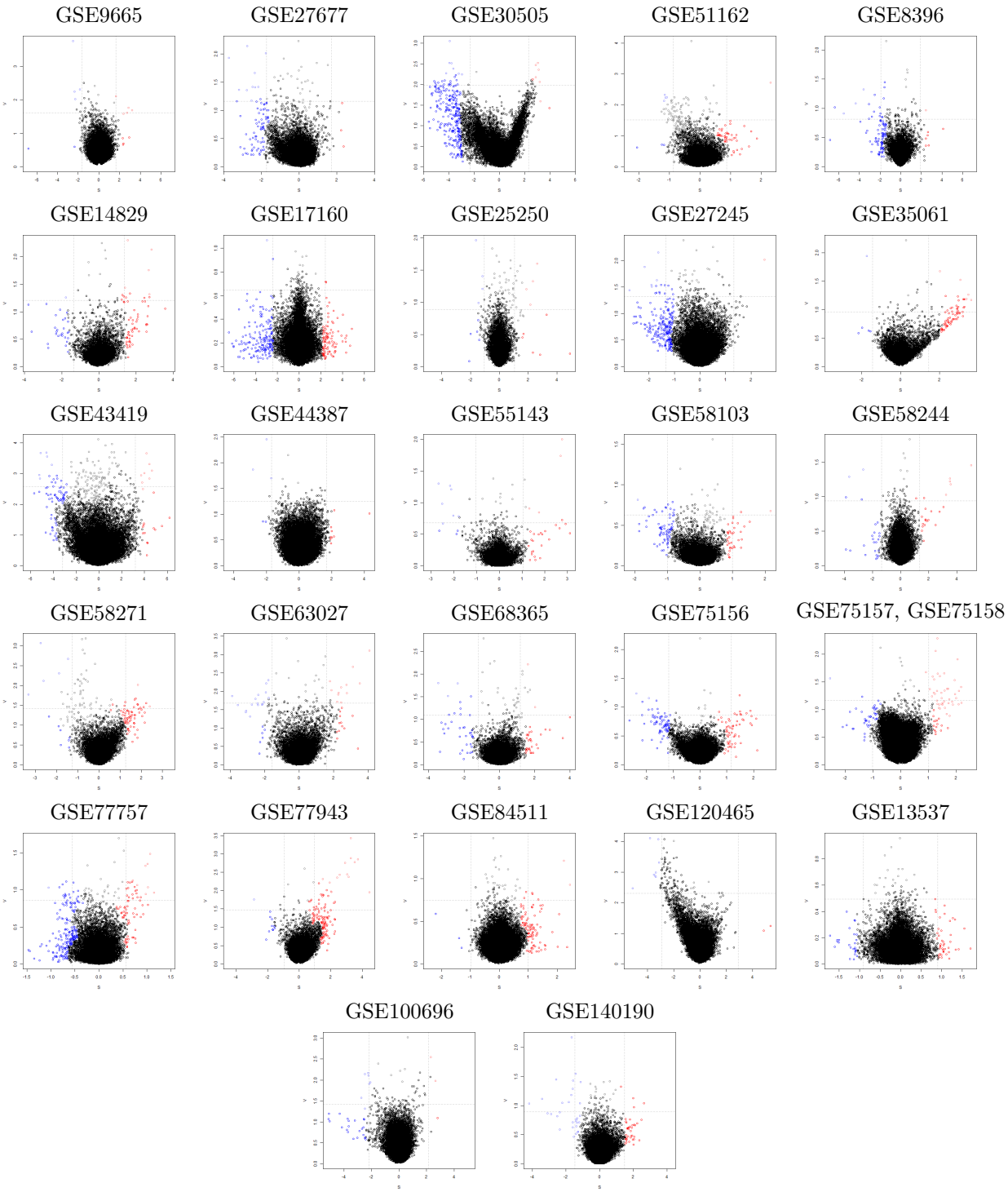

Figura C13 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando o método de separação de quantis por diferença de densidades (`quantile.method="density"`). Sua representação é equivalente à da Figura 18.

C.8 Q_{met}="k.max.sd"

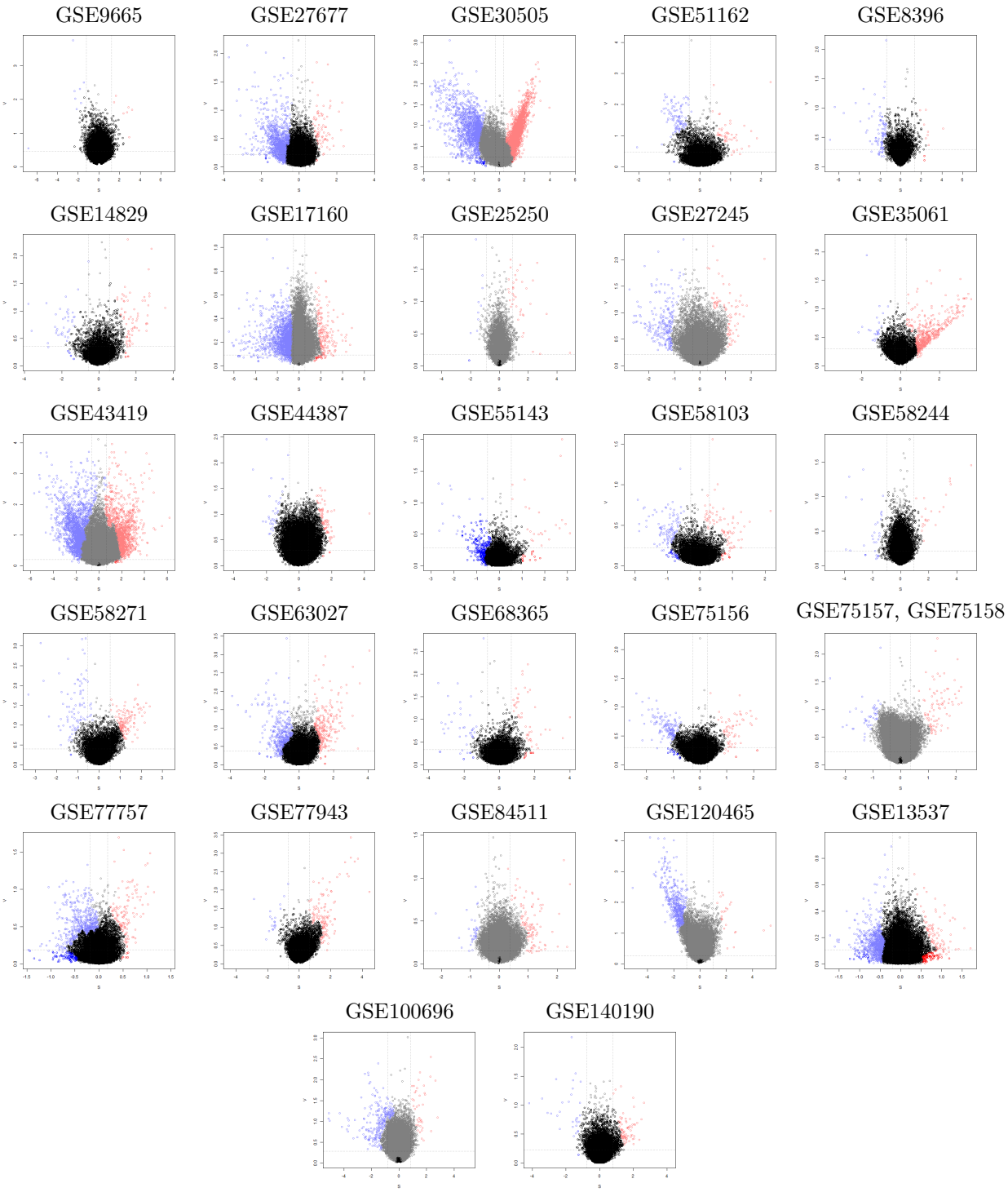

Figura C14 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando o método de separação de quantis pelo valor máximo de desvio-padrão entre k pontos vizinhos (`quantile.method="k.max.sd"`). Sua representação é equivalente à da Figura 18.

C.9 Q_{met}="proportional"

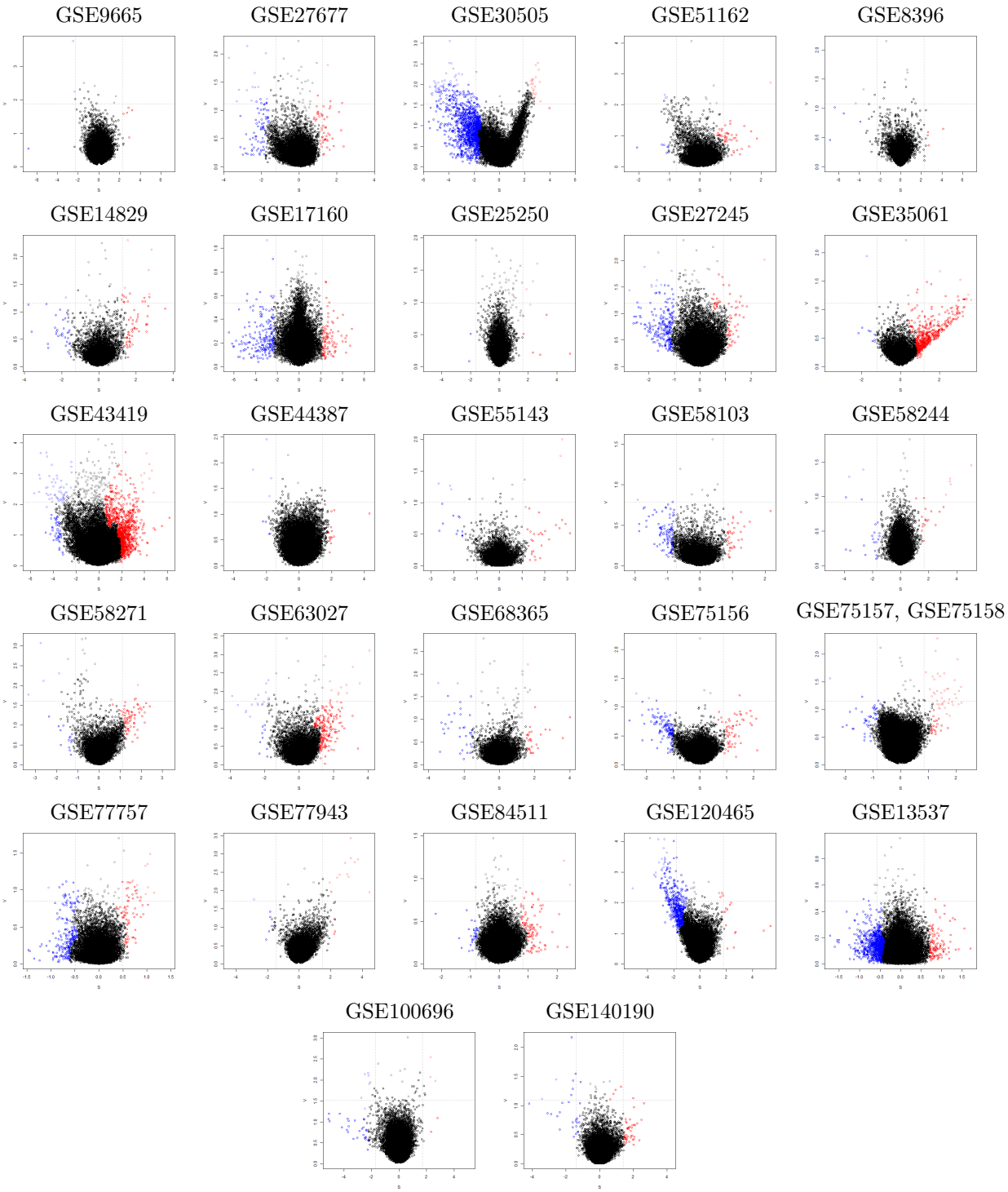

Figura C15 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando o método de separação de quantis por áreas proporcionais (`quantile.method="proportional"`). Sua representação é equivalente à da Figura 18.

C.10 S_{met}="mean"; V_{met}="var"

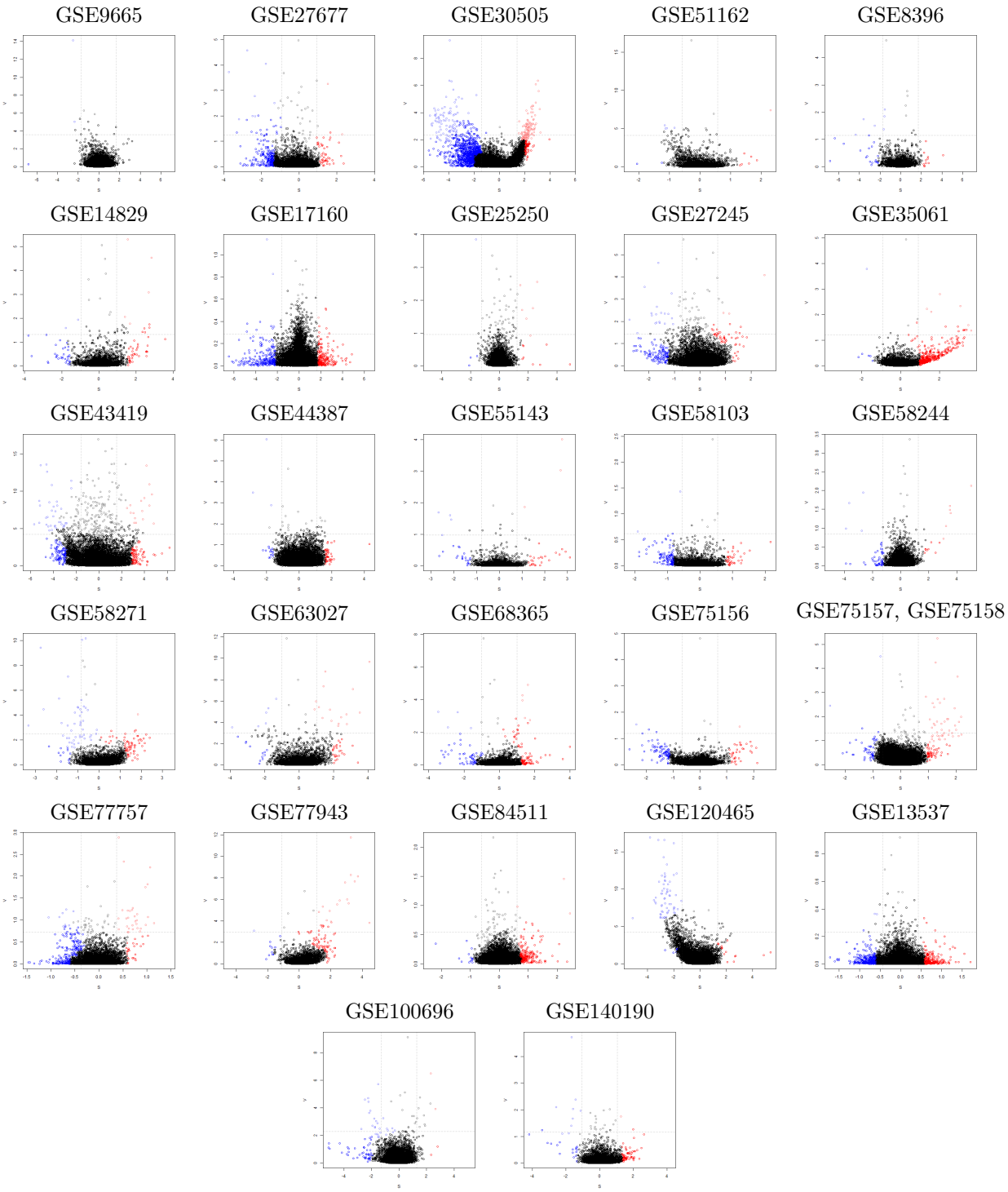

Figura C16 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando variância como método de estimativa de variação (`variation.method="var"`). Sua representação é equivalente à da Figura 18.

C.11 S_{met}="median"; V_{met}="mad"

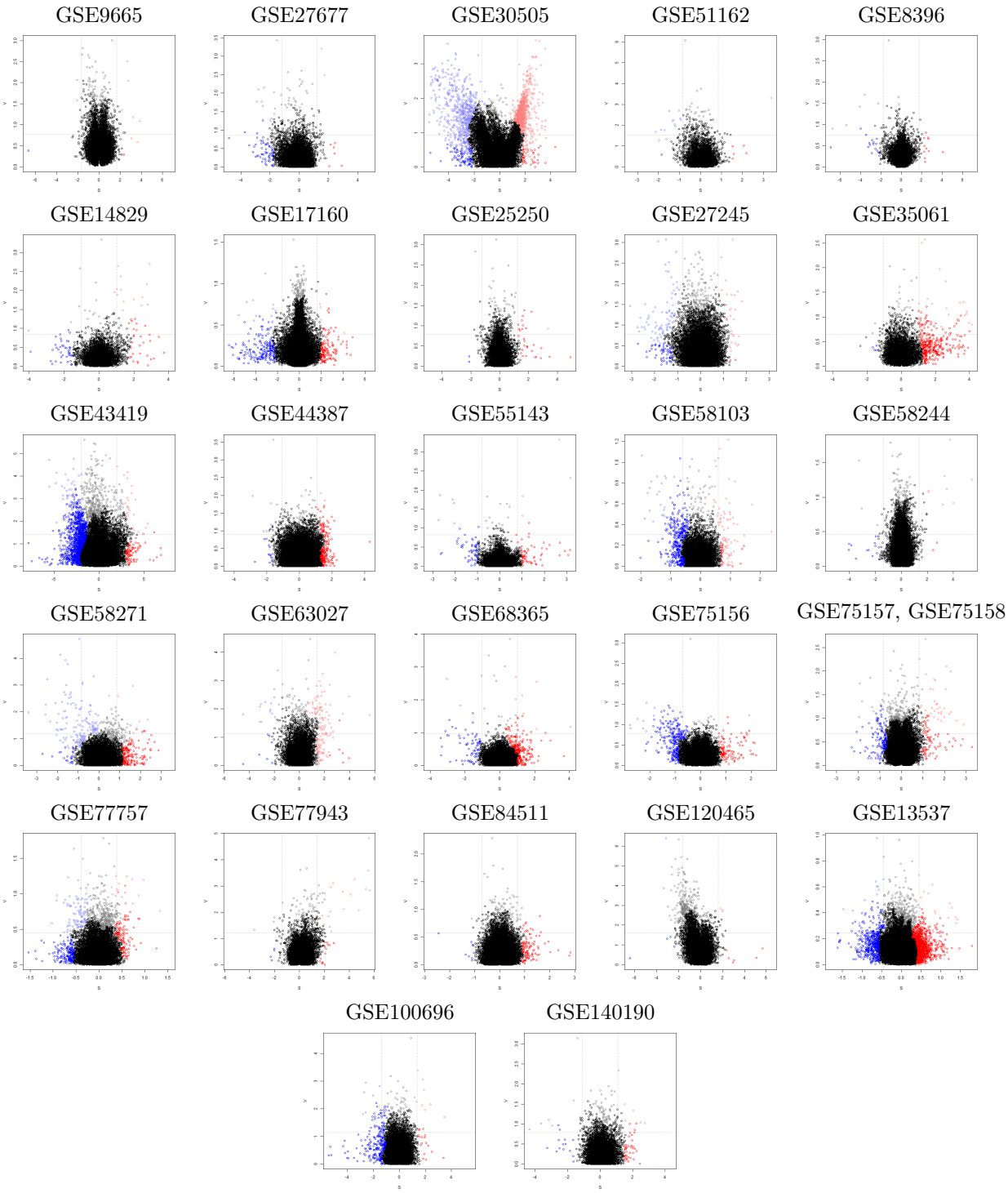

Figura C17 – Gráficos SV representando os resultados finais para os conjuntos de dados analisados com o GEVA utilizando mediana como método de sumarização (`summary.method="median"`) e MAD como método de estimativa de variação (`variation.method="mad"`). Sua representação é equivalente à da Figura 18.

Anexos

ANEXO A – Curriculum Vitæ resumido

NUNES, I. J. G.

1. **Dados pessoais**

Nome: Itamar José Guimarães Nunes

Data e local de nascimento: 26 de junho de 1994 – Porto Alegre (RS), Brasil

Endereço profissional: Av. Bento Gonçalves, 9500, Instituto de Informática, Prédio 67, Lab. 203 – Porto Alegre (RS), Brasil

E-mail: nunesijg@gmail.com

2. **Formação**

`2012-2018` Bacharelado em Biotecnologia – Bioinformática pela UFRGS

`2018-2020` Mestrado pelo Programa de Pós-Graduação em Biologia Celular e Molecular (PPGBCM) da UFRGS

3. **Estágios**

`2012-2013` Iniciação Científica Voluntária – UFRGS

Título: Participação do processo inflamatório e metalotioneínas durante o envelhecimento

Orientador: Diego Bonatto
Co-orientador: Bruno César Feltes

`2013-2014` Bolsa de Iniciação Científica – UFRGS

Programa: BIC UFRGS
Título: O papel da neuroinflamação durante o envelhecimento cerebral através de análises interatômicas

Orientador: Diego Bonatto
Co-orientador: Bruno César Feltes

`2014-2015` Bolsa de Iniciação Científica – UFRGS

Programa: PROBIC FAPERGS-UFRGS
Título: Estudo de mutações no gene *xpa-1* de *C. elegans* como uma forma de explorar o processo de envelhecimento e sua relação com o reparo de DNA

Orientador: Diego Bonatto
Co-orientador: Bruno César Feltes

`2016-2017` Iniciação Científica Voluntária – UFRGS

Título: Relação entre a deficiência do gene *xpa-1* de *C. elegans* e o processo de fertilidade em *Caenorhabditis elegans*

Orientador: Diego Bonatto
Co-orientador: Bruno César Feltes

`2017-2018` Bolsa de Iniciação Tecnológica – UFRGS

Programa: PIBITI CNPq-UFRGS
Título: Desenvolvimento de uma plataforma visual, flexível e intuitiva para análises de expressão gênica por microarranjo

Orientadora: Mariana Recamonde-Mendoza
Co-orientador: Bruno César Feltes

`2018` Estágio Supervisionado em Biotecnologia – UFRGS

Título: Desenvolvimento de uma ferramenta on-line para geração e inserção de estruturas glicanas

Orientador: Hugo Verli

4. **Prêmios e distinções**

`2014` Destaque em Apresentação Oral – XXVI Salão de Iniciação Científica da UFRGS

`2018` Menção Honrosa em Apresentação Oral e Vídeo – VIII Feira de Inovação Tecnológica da UFRGS

5. **Artigos completos publicados em periódicos**

`2014` FELTES, B. C.; POLONI, J. F.; NUNES, I. J. G. & BONATTO, D. Fetal Alcohol Syndrome, Chemo-Biology and OMICS: Ethanol Effects on Vitamin Metabolism During Neurodevelopment as Measured by Systems Biology Analysis. *OMICS: Journal of Integrative Biology*, v. 18, p. 344-363, 2014.

`2020` FELTES, B. C.; POLONI, J. F.; NUNES, I. J. G.; FARIA, S. S. & DORN, M. Multi-Approach Bioinformatics Analysis of Curated Omics Data Provides a Gene Expression Panorama for Multiple Cancer Types. *Frontiers in Genetics*, v. 11, p. 1354, 2020.

6. **Capítulos de livro publicados**

`2015` NUNES, I. J. G. & BONATTO, D. Envelhecimento molecular: uma série de teorias para um único mecanismo. In: FILHO, W. J. *Envelhecimento: Uma Visão Interdisciplinar*. Rio de Janeiro: Atheneu, Cap.32, p. 307-318, 2015.

7. **Programas de computador registrados**

2018 NUNES, I. J. G.; FELTES, B. C. & MENDOZA, M. R. Gene Expression Analysis Platform. *Instituto Nacional da Propriedade Industrial – INPI*, registro nº BR512018052262-9, 2018.

2019 NUNES, I. J. G.; & HAUENSTEIN, N. D. P. Livro Digital. *Instituto Nacional da Propriedade Industrial – INPI*, registro nº BR512019000261-0, 2019.

8. **Resumos e trabalhos apresentados em congressos**

2013 NUNES, I. J. G.; FELTES, B. C. & BONATTO, D. Participação do processo inflamatório e metalotioneínas durante o envelhecimento. In: XXV Salão de Iniciação Científica, Porto Alegre, 2013.

2014 NUNES, I. J. G.; FELTES, B. C. & BONATTO, D. O papel da neuroinflamação durante o envelhecimento cerebral através de análises interatômicas. In: XXVI Salão de Iniciação Científica, Porto Alegre, 2014.

2014 NUNES, I. J. G. The role of neuroinflammation during brain aging through an interatomic analysis. In: XVII Meeting of the Brazilian Society for Cell Biology, Foz do Iguaçu, 2014.

2015 NUNES, I. J. G.; FELTES, B. C. & BONATTO, D. Estudo de mutações no gene *xpa-1* de *C. elegans* como uma forma de explorar o processo de envelhecimento e sua relação com o reparo de DNA. In: XXVII Salão de Iniciação Científica, Porto Alegre, 2015.

2016 NUNES, I. J. G.; FELTES, B. C. & BONATTO, D. Relação entre a deficiência do gene *xpa-1* de *C. elegans* e o processo de fertilidade em *Caenorhabditis elegans*. In: XXVIII Salão de Iniciação Científica, Porto Alegre, 2016.

2018 NUNES, I. J. G.; FELTES, B. C. & MENDOZA, M. R. Desenvolvimento de uma plataforma visual, flexível e intuitiva para análises de expressão gênica por microarranjo. In: VIII Feira de Inovação Tecnológica (FINOVA), Porto Alegre, 2018.

ANEXO B – Artigos publicados

ORIGINAL RESEARCH ARTICLE

Front. Genet. | doi: 10.3389/fgene.2020.586602

Multi-approach bioinformatic analysis of curated omic data provide a gene expression panorama for multiple cancer types

Accepted article

 Bruno César Feltes[1*], Joice De Faria Poloni[1],  Itamar J. Guimarães Nunes[2],  Sara S. Faria[3] and  Marcio Dorn[1*]

[1]Institute of Informatics, Federal University of Rio Grande do Sul, Brazil
[2]Federal University of Rio Grande do Sul, Brazil
[3]University of Brasilia, Brazil

Studies describing the expression patterns and biomarkers for the tumoral process grow every year. This availability of new datasets, although essential, also creates a confusing landscape, where common or critical mechanisms are clouded amidst the divergent and heterogeneous nature of such results. In this work, we manually curated the Gene Expression Omnibus using rigorous filtering criteria, to select the most homogeneous and quality microarray and RNA-seq datasets from multiple types of cancer. By applying systems biology approaches, combined with a machine learning analysis, we investigated possible frequently deregulated molecular mechanisms underlying the tumoral process. Our multi-approach analysis of 99 curated datasets, composed of 5.406 samples, revealed 47 differentially expressed genes in all analyzed cancer types, which were all in agreement with the validation using TCGA data. Results suggest that the tumoral process is more related to the overexpression of a core deregulated machinery than the underexpression of a given gene set. Additionally, we identified gene expression similarities between different cancer types not described before and performed an overall survival analysis using 20 cancer types. Finally, we were able to suggest a core regulatory mechanism that could be frequently deregulated.

Keywords: regulatory networks, overall survival, machine learning, omics, bioinformatics, Systems Biology, Cancer

Received: 23 Jul 2020; **Accepted:** 09 Oct 2020.

*** Correspondence:**
Dr. Bruno César Feltes, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, 90040-060, Rio Grande do Sul, Brazil, bcfeltes@gmail.com
Prof. Marcio Dorn, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, 90040-060, Rio Grande do Sul, Brazil, mdorn@inf.ufrgs.br