UNIVERSIDADE ESTADUAL DE MARINGÁ
DEPARTAMENTO DE FÍSICA

Leonardo Ribeiro da Cunha

Estudo de Agrupamento de Séries Temporais Aplicado à Popularidade de Jogos *Online*

Maringá, Março de 2020.

UNIVERSIDADE ESTADUAL DE MARINGÁ
DEPARTAMENTO DE FÍSICA

Leonardo Ribeiro da Cunha

Estudo de Agrupamento de Séries Temporais Aplicado à Popularidade de Jogos *Online*

Trabalho apresentado ao Programa de Pós-Graduação em Física da Universidade Estadual de Maringá como dissertação de mestrado.

Orientador: Prof. Dr. Renio dos Santos Mendes

Maringá, Março de 2020.

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá - PR, Brasil)

C972e

Cunha, Leonardo Ribeiro da

Estudo de agrupamento de séries temporais aplicado à popularidade de jogos *online* / Leonardo Ribeiro da Cunha. -- Maringá, PR, 2020.

65 f.: il. color., figs.

Orientador: Prof. Dr. Renio dos Santos Mandes.

Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Ciências Exatas, Departamento de Física, Programa de Pós-Graduação em Física, 2020.

1. Sistemas complexos. 2. Ciência de dados. 3. Aprendizagem de máquina. 4. Jogos eletrônicos. 5. Séries temporais. I. Mandes, Renio dos Santos, orient. II. Universidade Estadual de Maringá. Centro de Ciências Exatas. Departamento de Física. Programa de Pós-Graduação em Física. III. Título.

CDD 23.ed. 530.13

Marinalva Aparecida Spolon Almeida - 9/1094

LEONARDO RIBEIRO DA CUNHA

ESTUDO DE AGRUPAMENTO DE SÉRIES TEMPORAIS APLICADO A POPULARIDADE DE JOGO *ONLINE*

Dissertação apresentada à Universidade Estadual de Maringá, como requisito parcial para a obtenção do título de mestre.

Aprovado em: Maringá, 27 de março de 2020.

BANCA EXAMINADORA

Prof. Dr. Renio dos Santos Mendes
Universidade Estadual de Maringá - UEM

Dr. Luiz Gustavo de Andrade Alves
Northwestern University

Prof. Dr. Haroldo Valentin Ribeiro
Universidade Estadual de Maringá - UEM

Resumo

Neste trabalho aplicamos técnicas de aprendizado de máquina não supervisionado a dados de popularidade de jogos *online*. Em particular, analisamos as séries temporais da popularidade dos jogos da plataforma *Steam*. A extração dos dados, o tratamento, as análises e os gráficos foram feitos em ambientes de programação em *Python*. Extraímos informações de 18211 jogos, referindo a: nome, data de lançamento, gêneros, desenvolvedora, publicadora e série temporal da popularidade mensal. No primeiro capítulo, apresentamos o dado e fornecemos informações demográficas acerca dele, mostrando tendências gerais das principais variáveis e algumas correlações. Após mostrar as principais informações demográficas, passamos os dados por um processo de triagem e aproveitamos os dados de 2972 jogos, o restante foi descartado. No segundo capítulo, abordamos a análise principal do trabalho, o estudo de agrupamento de séries temporais. Empregando o algoritmo de redução de dimensionalidade t-SNE, combinado com o método de detecção de grupos *Gaussian Mixture Model*, distinguimos 16 grupos de séries temporais. Porém, argumentando qualitativamente, reduzimos os 16 grupos a apenas três grandes padrões de dinâmica de popularidade. O primeiro padrão apresenta a forma de decaimento, o segundo padrão apresenta a forma de sino, e o terceiro apresenta a forma de morro de inclinações suave. De forma geral, acreditamos que nosso estudo representa uma contribuição para estudos de dinâmica de popularidade, mesmo que nosso estudo tenha abordado o dados de jogos, também cremos que este trabalho pode ser útil ou até mesmo aplicável em outros contextos.

Palavras-chave: Sistemas complexos. Ciência de Dados. Aprendizagem de máquina. t-SNE. Gaussian Mixture Model. Jogos Eletrônicos. *Steam*.

Abstract

In this work we employ unsupervised machine learning techniques to game popularity data. We focus our investigation on the popularity of Steam games. We extract information about more than 18000 games, concerning: name, release date, genre, developer, publisher, and time series. In the first chapter, we show some demographic informations about them. We also filter the most important data for our analysis, in this processes we discard most data and remain with 2972 time series. In the second chapter, we employ the t-SNE algorithm and Gaussian Mixture Model to detect the optimal number of clusters. Our findings show 16 significant clusters, but these clusters can be reduced to three major. We believe that, our investigation represents a little contribution for studies about popularity times series, and can even be useful in the other contexts.

Keywords: Complex systems. Data Science. Machine learning. Clustering analysis. t-SNE. Gaussian Mixture Model. Eletronic Games. Steam.

Sumário

Introdução

A conceituação de sistemas complexos é bem abrangente no contexto de ciência. Comumente, sistemas constituídos por uma grande quantidade de componentes e com padrões matemáticos pouco determinísticos podem ser considerados sistemas complexos [1]. Em particular, a subjetividade em quão grande deve ser o número de componentes do sistema dificulta uma definição precisa para sistemas complexos. Além disso, não existe uma forma certa de determinar o grau em que um sistema é determinístico ou não [2]. Contudo, geralmente os sistemas complexos são compostos por componentes que podem interagir entre elas das maneiras mais heterogêneas possíveis, que é resultado da individualidade de cada componente. Dessa forma, localmente, existe a possibilidade de surgir inúmeros padrões [3]. Devido a alta variabilidade de tais sistemas, a análise de sistemas complexos envolve uma grande quantidade de ferramentas matemáticas, estatísticas e físicas para investigar padrões globais [4]. De uma maneira geral, sistemas complexos podem parecer muito confusos observando localmente, porém ao sumarizar as principais características, identifica-se a emergência de padrões gerais e bem definidos [5].

Como já deve estar claro, muitos sistemas nos mais variados contextos podem ser abordados como sistemas complexos. Em tais sistemas podem ocorrer uma grande variedade de fenômenos, que incluem desde fenômenos naturais, biológicos, econômicos a sociais [6–22]. Em fenômenos naturais, exemplos clássicos de análise de sistemas complexos são os estudos das ocorrências dos terremotos [23]. Segundo resultados bem documentados na literatura, uma vertente de correlações temporais de curto alcance entre os terremotos é descrita pela Lei de Omori, sugerindo que após um terremoto principal há uma sequência de choques posteriores cuja a frequência diminui com o tempo: $T^{-\alpha}$, em que $\alpha \approx 1$ [24, 25]. Além disso, entre outros resultados, já foi mostrado que a distribuição espacial dos epicentros seguem uma estrutura de fractal [26].

O sucesso das análises de sistemas complexos fez surgir aplicações clássicas na literatura

científica dos últimos anos. Em biologia, espécies podem ser interpretadas como as componentes de um sistema complexo, já as interações entre as componentes se dão pelas relações ecológicas entre as espécies [27–29]. Em uma vertente de aplicação em fenômenos econômicos, pode-se relacionar a valorização e a desvalorização de produtos e empresas no mercado financeiro [30, 31]. Um exemplo recorrente nos últimos anos se deu pelo crescente interesse nas criptomoedas, trabalhos foram feitos testando a hipótese de eficiência de mercado [32,33]. Em fenômenos sociais, um dos muitos sistemas complexos possíveis pode ser uma dada empresa, dessa forma os trabalhadores assumiriam o papel das componentes e, por exemplo, as interações entre as componentes são as relações sociais entre os funcionários [34–36]. Outra aplicação em fenômenos sociais está relacionada aos índices sociais de uma cidade e sua população, por meio de leis alométricas [37, 38]. Enfim, em meio a tantos exemplos, já foi possível evidenciar a relevância e o sucesso dos estudos de sistemas complexos.

A interdisciplinaridade presente em trabalhos de sistemas complexos tem sido um grande trunfo para seu sucesso, porém um elemento chave para esse tipo de produção científica é o dado empregado. Grandes trabalhos científicos estão cada vez mais relacionados à grande quantidade e a alta qualidade dos dados empregados; não há dúvidas que os dados têm uma importância central na ciência. Paralelamente ao que já foi dito, deve-se salientar os avanços tecnológicos no setor de processamento e armazenamento de dados, que possibilitam a circulação livre e aberta de alguns bancos de dados na internet [39]. Em meio a esse contexto de acúmulo de dados, foi cunhado o termo *big data*, que ficou popular nos últimos anos para denotar esses grandes bancos de dados [40]. A popularização do *big data* revelou a necessidade de ferramentas para manipular grandes quantidades de dados de forma rápida e eficiente; essas ferramentas computacionais são baseadas em linguagens de programação. Uma linguagem de programação que cada dia tem se tornado mais popular na ciência é o *Python* [41]. Nela, por meio de bibliotecas disponíveis abertamente, é possível extrair dados de sites na internet, organiza-los em *data frames*, manipula-los matematicamente e visualizar em gráficos de alta qualidade [42, 43].

Após os processo de obtenção dos dados, começa o processo de analisar os dados e consequentemente entender aspectos intrínsecos da natureza do dado. O primeiro passo da análise de dados está ligado ao entendimento do tipo de dado, sendo que a forma mais completa seria a disposição espacial ao longo do tempo da magnitude de uma grandeza. Porém, nem sempre o dado obtido contém a informação espacial e temporal da grandeza, por exemplo muitas vezes trabalhos científicos apresentam apenas a disposição espacial da magnitude de uma grandeza [44, 45]. Outras vezes, o dado obtido representa apenas a evolução temporal da magnitude da grandeza e nesse caso é comum se referir a tal dado como série temporal. De forma simples, uma série temporal nada mais é que uma coleção de observações feitas sequencialmente ao longo do tempo [46]. Anteriormente já foram citados exemplos de trabalhos de séries temporais, além dos já citados existem inúmeros

outros na literatura [47–51]. Os trabalhos de análises de séries temporais são, em especial, importantes para determinar as tendências de crescimento/decrescimento dos dados, para análises relacionadas a auto correlações das séries temporais ou similaridade entre elas [52–55].

Nesse contexto técnico de análise de dados, um grande foco do atual trabalho está direcionado a agrupamentos de séries que compartilham características similares. Trabalhos nesse sentido têm o objetivo de caracterizar grupos de séries temporais, determinar critérios de classificação e se possível trazer modelos preditivos para eventos futuros [56]. Estudos de agrupamento de dados são comumente chamados de análises de *clusters*. Antes do procedimento de agrupamento propriamente dito, as séries são comparadas utilizando uma medida de distância, também chamada de medida de dissimilaridade, a partir dessa medida pode ser construído uma matriz de distância entre as séries. Algumas das medidas de distância conhecidas são as de Euclides, Minkowski, Chebyshev, correlação de Pearson, entropia, DTW, LCSS etc [57, 58]. Não é o objetivo desse trabalho explorar as diferentes propriedades de cada medida de distância, porém o que pode ser deixado claro é que cada distância tem sua forma matemática, ou seja, cada distância agrupará ponderando diferentemente características específicas das séries.

Nesse ponto, baseado na matriz de dissimilaridade, a análise de *cluster* separa os grupos de séries temporais de tal forma que a similaridade seja máxima dentro do grupo e mínima fora [58, 59]. O procedimento de agrupamento é comumente chamado de aprendizado de máquina não supervisionado, essa nomenclatura se deve ao fato de não haver uma variável específica a ser respondida, portanto ao fim da análise não haverá o valor de uma variável mas sim grupos de séries semelhantes [60]. A forma de separar os grupos depende do tipo de algoritmo usado, os quatro principais tipos de algoritmos são *hierarchical clustering*, *partitioning clustering*, *model based clustering* e *density based clustering* [58]. O *hierarchical clustering* tem como principal característica fazer uma hierarquia da similaridade das séries, conhecido como dendrograma [61]. A construção dessa hierarquia pode ser de duas formas: aglomerativa e divisiva. A primeira forma considera que cada série é um grupo e a cada passo do algorítimo os dois grupos mais próximos se juntam, esse processo se repete até todas as séries constituírem um único grupo; a segunda forma representa o processo inverso, inicialmente todas as séries estão em um mesmo grupo e elas se separam até cada uma ficar sozinha. Visualmente o dendrograma é muito interessante para entender toda a estrutura de grupos contida no dado, porém o *hierarchical clustering* não determina diretamente quais grupos maximizam as semelhanças internas e minimizam as externas [62]. O *partitioning clustering* requer um conhecimento prévio dos dados, pois esse procedimento depende do número k de grupos como *input*. De forma resumida, o *partitioning clustering* elege k grupos (aleatoriamente como padrão) e então, a medida que o algoritmo avança ele troca as séries de grupo com objetivo de minimizar a distância total em relação ao centro do *cluster* [63, 64].

O *model based clustering* assume um modelo para cada grupo, a partir disso tenta fazer o melhor ajuste de dados para esse modelo [65, 66]. Esse tipo de análise pode ser usado em dados com padrões de modelos matemáticos bem conhecidos, o ponto positivo dessa análise é ter um resultado analítico que representa cada grupo. O *density based clustering* identifica grupos em regiões do espaço com concentração de dados semelhantes. Dessa forma, para cada dado dentro de um grupo, a vizinhança em um dado raio tem que conter pelo menos um número mínimo de pontos [67]. Diferentemente das análises anteriores que tenta encaixar os dados em algum grupo, os dados no *density based clustering* que não se encaixam em nenhum desses grupos geralmente são considerados ruídos. Depois da exposição das ideias gerais por trás de cada algorítimo de agrupamento, deve ser dito também que ainda há alguns trabalhos que apresentam algorítimos híbridos, isto é, combina dois ou mais procedimentos de *clusterização* [68, 69]. O entendimento de aprendizado de máquina não supervisionado tem um importante papel como guia no desenvolvimento de intuições a respeito do dado, apesar desse tipo de análise não trazer uma variável resposta, a análise de *cluster* pode servir como base para modelos preditivos.

Do ponto de vista cientifico, as ferramentas de análises de dados aplicados a sistemas sociais podem, por exemplo, ser usadas para explicar padrões comportamentais da sociedade contemporânea. Nos últimos anos, houve um forte impacto no comportamento humano devido ao desenvolvimento de tecnologias, muitas formas de entretenimento ficaram populares e são consumidas por muitas pessoas das diversas camadas socioeconômicas [70]. Em particular, o impacto gerado pela tecnologia motivou algumas discussões sociológicas, além de um certo interesse por parte da ciência de modo geral [71]. Como já foi dito anteriormente, o *big data* muitas vezes é gerado automaticamente pelo consumo da própria tecnologia e os dados por vezes estão disponíveis livremente, a partir disso segundo o autor Matthew Salganik [72], o *big data* já faz parte da nossa sociedade a tempo suficiente para permitir medições de mudanças de nosso comportamento.

Os estudos comportamentais focam uma variedade de aspectos da sociedade, desde o consumo de produtos culturais, dinâmica de atenção em assuntos contemporâneos, até a distribuição de citações no meio científico. Muitos trabalhos podem ser citados, por exemplo, no artigo *Robust dynamic classes revealed by measuring the response function of a social system*, os autores estudaram a popularidade de 5 milhões de vídeos do *YouTube*, os resultados mostram que a popularidade pode ser descrita por processos de Poisson e leis de potência [73]. O padrão lei de potência é verificado em muitos outros contextos, tendo como exemplo o reportado por [74], a distribuição de citações de artigos científicos é lei de potência. Além disso, nesse trabalho um segundo achado é a dinâmica de *burst* nos primeiros anos da publicação. Outro artigo mostra a dinâmica de *burst* não apenas no contexto de citações cientificas, mas na propagação de *hashtags* no *Twitter*, a ocorrência anual de livros por autores e vendas de ingresso para filmes, o trabalho mostrou que os picos de popularidade

estão ficando mais rápidos, abruptos e frequentes [75]. Nessa direção de explicar a dinâmica de atenção, pesquisadores [76] utilizaram o dado de um milhão de pessoas que visitaram a plataforma de noticias digg.com, seus resultados mostraram um decaimento exponencial da novidade dentro dos grupos de pessoas. Estudos comportamentais também levam a respostas sobre a dinâmica de esquecimento, por exemplo as músicas não permanecem muito em nossa memória (aproximadamente $5,6$ anos), enquanto que as biografias tendem a permanecer mais tempo (de 20 a 30 anos) [77]. Após essa discussão, deve estar claro o recente interesse da comunidade científica na dinâmica de substituição da atenção por parte da sociedade.

O presente trabalho é dedicado ao estudo da dinâmica de popularidade de jogos *online*, em que técnicas de aprendizado de máquina são empregadas. Uma das formas de entretenimento proporcionada pelas tecnologias é os jogos digitais, com destaque para os de interação *online*. Atualmente, os jogos eletrônicos são uma forma difundida de entretenimento contemporâneo entre crianças, jovens e adultos [78]. A globalização contribui com a facilidade de acessar aos jogos, além disso o aperfeiçoamento gráfico e as melhorias de jogabilidade são responsáveis por conquistar cada vez mais jogadores ao redor do mundo [79]. Um conjunto considerável dos jogadores de jogos eletrônicos faz parte da comunidade da *Steam*, a plataforma de distribuição de mídia digital desenvolvida pela Valve Corporation [80]. De forma mais detalhada, a *Steam* é uma loja *online* de jogos que gerencia os direitos autorais e administrativos dos produtos de sua base. No presente, ela conta com mais de 65 milhões de usuários, atingindo milhões de usuários *online* simultaneamente. Paralelamente a isso, o *site* [81] divulga abertamente o número médio de jogadores mensais de cada jogo da *Steam*, ou seja, torna público as séries temporais da popularidade mensal média de cada jogo. Assim usando as técnicas de *clusterização*, faz parte da proposta deste trabalho agrupar as séries temporais cuja forma da dinâmica temporal sejam parecidas, dessa maneira resultando na caracterização do padrão de consumo de cada grupo de jogos. Para isso, o trabalho foi dividido em três capítulos: 1) apresentação dos dados, 2) resultados da análise de *cluster* e 3) conclusão. A apresentação dos dados mostrará detalhes da extração e da manipulação, algumas informações demográficas, e a filtragem dos dados mais interessantes para o estudo. Vinculado aos resultados da análise de *cluster*, haverá um detalhamento dos métodos de agrupamentos, do algorítimo empregado e dos aspectos intrínsecos de cada grupo detectado. Sob o enfoque das ferramentas empregadas para a obtenção e análise do dado, deve ser dito que são de acesso livre e são baseados em *Python*. Finalmente, a conclusão conterá a sumarização dos principais resultados do trabalho e perspectivas para pesquisas futuras.

CAPÍTULO 1

Apresentação dos dados

Neste capítulo, apresentamos informações fundamentais a respeito do dado empregado na presente pesquisa. Mostramos como o dado está disponibilizado na *internet*, descrevemos com detalhes a técnica utilizada para a obtenção dos dados e até mesmo disponibilizamos o código em *Python*. Com os dados em mãos, apresentamos as informações básicas da demografia dos jogos da *Steam*, isto é, expomos alguns primeiros resultados tirados das variáveis do sistema. Fechamos este capítulo propondo filtros que selecionam os dados mais interessantes para a análise de *cluster* (abordado no capítulo 2).

1.1 Técnica de extração dos dados

Os dados utilizados neste trabalho dizem respeito unicamente a registros disponíveis da plataforma *Steam* e *Steam Charts*. A *Steam* é uma loja *online* de jogos que gerencia os direitos autorais e administrativos dos produtos de sua base, atualmente ela conta com mais de 65 milhões de usuários, atingindo milhões de usuários *online* simultaneamente [82]. Esses números representam uma amostra significativa da população mundial de jogadores, dado que a plataforma é acessada na maior parte dos países do globo. A figura 1.1 ilustra o *layout* do site da *Steam*, ao abrir a página de um jogo algumas informações técnicas são oferecidas, tais como gênero, desenvolvedora, publicadora e data de lançamento do jogo. Na figura 1.1, apresentamos o exemplo do jogo *Rocket League* dos gêneros Ação, *Indie*, Corrida e Esportes, desenvolvido e publicado por *Psyonix LCC*, lançado em 07 de Julho de 2015. Em várias revistas especializadas em jogos, é comum dizer que a *Steam* é a maior, em número de usuários e produtos disponíveis, plataforma que oferece esse serviço. Origin [83], Uplay [84], GOG [85], entre outras [86–88], também oferecem serviços parecidos.

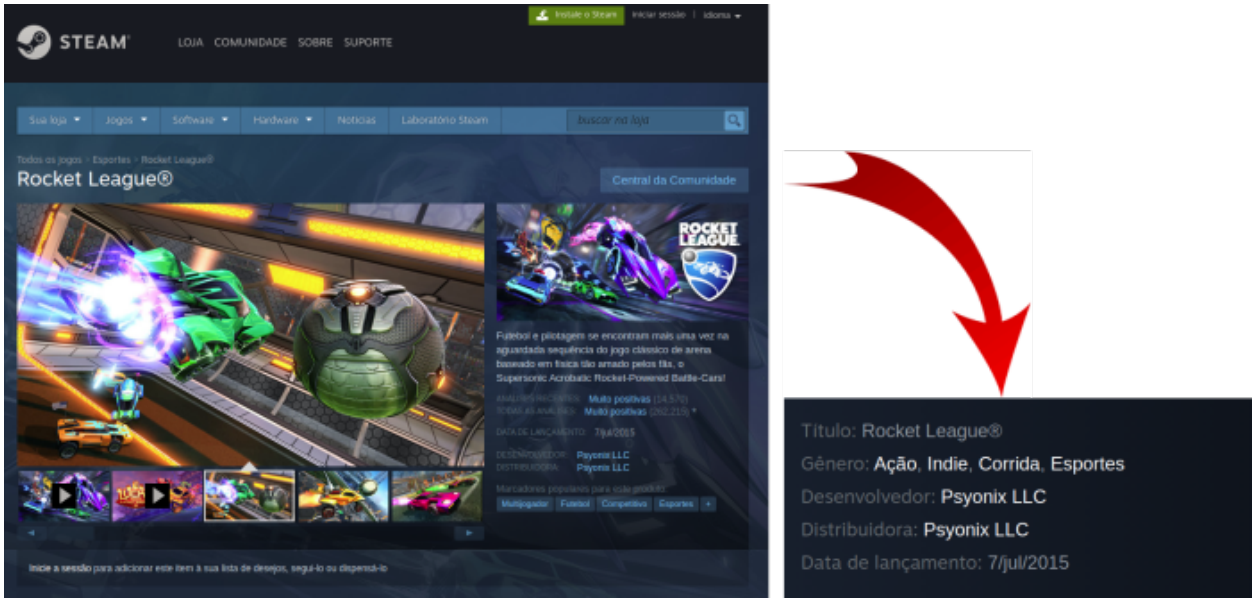

Figura 1.1: *Layout* da página do jogo *Rocket League* na *Steam*. A página apresenta o nome do jogo (*Rocket League*) em seu topo, algumas imagens e vídeos de demonstração logo abaixo, e no canto direito são disponibilizados algumas informações gerais do jogo e um avaliação votada pelos usuários da *Steam*. Mais abaixo nessa mesma página tem disponível um quadro com as informações técnicas do jogos, como mostrado pela seta vermelha.

A escolha pelos dados da *Steam* se deu pelo fácil acesso, pois um site chamado *Steam Charts* [81] disponibiliza abertamente o número mensal médio de jogadores por jogo. Assim como a *Steam*, a *Steam Charts* tem uma página dedicada para cada jogo, nessa página é apresentado um gráfico interativo do número de pessoas jogando e logo abaixo o número de jogadores mensais médio são organizados em uma coluna, veja a figura 1.2. Deve-se deixar claro como é contado o número mensal médio de jogadores para um determinado jogo, o algorítimo da *Steam Charts* soma o número de jogadores distintos diário e divide pelo número de dias do mês, isto é, se uma pessoa joga um dia no mês isso conta como 1/30 (supondo um mês de 30 dias) para a popularidade média mensal daquele mês. Apesar da plataforma *Steam* ter sido lançada em Setembro de 2003, o site *Steam Charts* começou em Julho de 2012, portanto os dados da popularidade mensal médias dos jogos anteriores a data de Julho de 2012 não são possíveis de ser obtidas livremente. Também vale dizer que alguns jogos oferecidos pela *Steam* também são oferecidos por outras plataformas citadas acima, ou seja, os usuários da *Steam* podem jogar um determinado jogo online com outros jogadores que não são usuários da *Steam*. No entanto, os dados disponíveis na *Steam Charts* são referentes apenas aos jogadores que são usuários da plataforma *Steam*.

O nosso banco de dados foi construído combinando as informações disponíveis da *Steam* e da *Steam Charts*, por meio da biblioteca *Request* da linguagem de programação *Python*. A biblioteca *Request* permite acesso ao código html de uma dada página. Basicamente,

Figura 1.2: *Layout* da página do jogo *Rocket League* na *Steam Charts*. A página apresenta o nome do jogo (*Rocket League*) em seu topo, logo abaixo o número de jogadores da última hora, o pico de jogadores das últimas 24 horas e o pico total de jogadores desde o lançamento. As principais informações da página encontram-se no gráfico interativo do número de jogadores ao longo do tempo, além da tabela contendo o número de jogadores médios por mês. Apesar de não estar exposta a tabela completa, a página da *Steam Charts* disponibiliza a série temporal completa desde o lançamento do jogo na plataforma *Steam*.

o conjunto de funções dessa biblioteca é utilizado para transformar o código html de uma página em uma *string* manipulável em um ambiente *Python*. Como o resultado é uma *string*, normalmente a biblioteca *Request* é utilizada conjuntamente com a biblioteca de expressões

regulares, pois esta última permite manipular de forma bem eficiente as *strings*. Resumindo o processo, a biblioteca *Request* baixa o html de uma página como uma *string* e dessa *string* retiramos os dados utilizando expressões regulares.

Exceto pelas duas bibliotecas citadas no parágrafo anterior, o código para extrair os dados da *Steam* e *Steam Charts* foi construindo apenas com funções básicas do *Python*. No decorrer deste parágrafo, vamos explicar passo a passo o nosso código. A primeira tarefa foi obter o conjunto de todos os *links* que levam às páginas dos jogos, para isso varremos a lista de todos os jogos. O *site Steam Charts* tem uma seção que organiza os jogos em uma lista do mais popular até o menos popular. A URL das páginas com essa lista tem uma estrutura muito similar, aproveitamos esse fato para fazer a lista em *Python* com a URL de todas as páginas que compõe a lista de jogos:

```
link_1= 'https://steamcharts.com/top'
lista_links= [link_1]
for i in range(2,1400):
    lista_links.append(link_1 + '/p.' + str(i))
```

Ao carregar o html de uma das páginas da lista com os nomes dos jogos, aparece uma *tag* para cada jogo da página, essas *tags* fazem parte da URL da página dedicada ao dado do jogo especificado. Dessa forma, utilizamos a função de recorrência para abrirmos todas as páginas da lista e pegarmos essas *tags* da página de cada jogo:

```
#faz a lista de links de cada jogo
lista_apps= []
for i in lista_links:
    r= requests.get(i)
    sub_lista= re.findall('href="/app/[0-9]+', r.text, flags=0)
    lista_apps.append(sub_lista)

#Pega somente 'app' e o número.
lista_temporaria= []
for i in lista_apps:
    for j in i:
        if 'app' in j:
            lista_temporaria.append(j[6:])
lista_apps= lista_temporaria
```

As *tags* obtidas foram úteis tanto para acessar o site da *Steam Charts* quanto o site da *Steam*, por exemplo, o jogo *Rocket League* tem sua página na *Steam* com a URL https://store.steampowered.com/app/252950/Rocket_League/ e na *Steam Charts* com

https://steamcharts.com/app/252950; note que "app/252950"aparece nos dois *links*. Alguns jogos da *Steam* apresentam conteúdo explícito de violência e nudez, na página desses jogos o site faz uma requisição da idade do usuário. Abaixo mostramos a configuração de *cookies* utilizada para conseguirmos acessar todas as páginas da *Steam*:

```
cookies = {
    'wants_mature_content': '1',
    'browserid': '1195052588184909523',
    'timezoneOffset': '-10800,0',
    '_ga': 'GA1.2.1167280872.1556023312',
    'sessionid': '245d1e8666669fa5ed1267ef',
    '_gid': 'GA1.2.2050464602.1556122848',
    'steamCountry': 'BR%7C099c202446b5b2b1c2f5652a882dd788',
    'birthtime': '-728510399',
    'lastagecheckage': '1-0-1947',
}

headers = {
    'Connection': 'keep-alive',
    'Upgrade-Insecure-Requests': '1',
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64)' + \
    'AppleWebKit/537.36 (KHTML, like Gecko) ' + \
    'Chrome/73.0.3683.103 Safari/537.36',
    'Accept': 'text/html,application/xhtml+xml,' + \
    'application/xml;q=0.9,image/webp,image/apng,' + \
    '*/*;q=0.8,application/signed-exchange;v=b3',
    'Accept-Encoding': 'gzip, deflate, br',
    'Accept-Language': 'en-US,en;q=0.9',
}
```

Com todas as *tags* dos jogos e as configurações de *cookies* prontas, bastou acessar a página de cada jogo na *Steam* e *Steam Charts*, e procurar no html as informações relevantes para o nosso trabalho. O código ficou bem técnico daqui para frente, mas o resultado final é um código que extrai as seguintes informações de cada jogo: nome, data de lançamento, gênero, desenvolvedora, publicadora, *link* do *site* da *Steam* e a série temporal da popularidade mensal média. Segue o restante do código:

```
#faz as listas que serão necessárias para capturar os dados
lista_erro_apps_2= []
```

```
lista_erro_nome= []
lista_erro_lancamento= []
lista_erro_publisher= []
lista_erro_developer= []
lista_erro_genero= []
lista_erro_request= []
dado= []

#Abre o site da \textit{Steam} de cada jogo e captura o nome,
#data de lançamento, lista de gênero do jogo,
#lista de desenvolvedores e publicadores.
#Aproveitando o mesmo 'app', abre o site
#da \textit{Steam Charts} e pega as séries.
for i in lista_apps:
    link_completo_charts= 'https://steamcharts.com' + i
    r= requests.get(link_completo_charts)
    lista_temp= re.findall('class="right num-f">[0-9]+.[0-9]' + \
    , r.text, flags=0)
    lista_time_series= []
    for j in range(len(lista_temp)-1, -1, -1):
        lista_time_series.append(lista_temp[j][20:])

    link_completo_steam= 'https://store.steampowered.com'+i
    try:
        r= requests.get(link_completo_steam, \
        headers=headers, cookies=cookies)
    except:
        lista_erro_request.append(link_completo_steam)
        lista_info_jogos= []
        dado.append([lista_info_jogos,lista_time_series])
        time.sleep(5)
        continue
    texto= r.text
    try:
        nome= re.findall('<title>.*on Steam</title>', texto, \
        flags=0)[0][7:-17]
        if nome[:5] == 'Save ' and nome[7:12]== '% on ':
            nome= nome[12:]
```

```
    except:
        lista_erro_nome.append(link_completo_steam)
        lista_info_jogos= []
        dado.append([lista_info_jogos,lista_time_series])
        continue
    try:
        data_lancamento= re.findall('<div class="date">.*' + \
        </div>', texto, flags=0)[0][18:-6]
    except:
        data_lancamento= ''
        lista_erro_lancamento.append(link_completo_steam)
    lista_generos= []
    try:
        auxilio_1= re.findall('Genre:.*<br>', texto, flags=0)[0]
        auxilio_1= re.split('<a href="https://store.' + \
        'steampowered.com/genre', auxilio_1, flags=0)
        for j in auxilio_1:
            try:
                auxilio_2= re.findall('>.*</a>', j, flags=0)[0]
                lista_generos.append(auxilio_2[1:-4])
            except:
                pass
    except:
        lista_erro_genero.append(link_completo_steam)
    lista_desenvolvedor=[]
    try:
        auxilio_3= re.findall('Developer:</b>' + \
        '\r\n\r\n\t\t\t\t<a.*</a>', texto, flags=0)[0]
        auxilio_3=re.split('/a>', auxilio_3, flags=0)
        for j in auxilio_3:
            try:
                auxilio_4= re.findall('>.*<', j, flags=0)[0]
                lista_desenvolvedor.append(auxilio_4[1:-1])
            except:
                pass
    except:
        lista_erro_developer.append(link_completo_steam)
    lista_publicadora= []
```

```
    try:
        auxilio_5=re.findall('Publisher:</b>' + \
        '\r\n\r\n\t\t\t\t<a.*</a>', texto, flags=0)[0]
        auxilio_5=re.split('/a>', auxilio_5, flags=0)
        for j in auxilio_5:
            try:
                auxilio_6= re.findall('>.*<', j, flags=0)[0]
                lista_publicadora.append(auxilio_6[1:-1])
            except:
                pass
    except:
        lista_erro_publisher.append(link_completo_steam)
    lista_info_jogos= [nome,data_lancamento,lista_generos, \
    lista_desenvolvedor,lista_publicadora,link_completo_steam]
    dado.append([lista_info_jogos,lista_time_series])
```

Alguns comentários devem ser feitos a respeito do código. Os dados foram extraídos no dia 27 de Abril de 2019, por esse motivo as maiores séries presentes no dado tem 81 pontos referentes aos 81 meses entre Abril de 2019 e Julho de 2012. O código captura todos os aplicativos comercializados na *Steam*, inclusive editores de imagem e áudio, mídias como filmes e curtas metragens animadas, entre outros aplicativos em geral. A respeito disso, foram descartados todos os aplicativos que não fossem jogos. Alguns produtos da *Steam* estão disponíveis a pouco tempo para serem comercializados, por conta disso alguns jogos não possuem série temporal e o código preenche com uma lista vazia. Uma lista pequena de jogos vieram sem data de lançamento, isso requisitou acrescentar algumas datas de lançamento manualmente. Por último, alguns dados vieram duplicados e precisaram ser descartados.

1.2 Informações demográficas: primeiros resultados

Esta seção é dedicada às informações demográficas do dado, objetivando mostrar uma visão geral do comportamento do sistema. Inicialmente, a nossa análise se concentrou em determinar os aspectos mais macroscópicos do sistema. Dessa forma, na figura 1.3, mostramos a evolução temporal do número médio de usuários da *Steam*, de jogos e a média de jogadores por jogo. Os valores mostrados na figura 1.3A são a soma do número médio de jogadores por jogo em cada mês. Isso indica uma superestimação, pois um usuário pode jogar mais de um jogo e ser contados múltiplas vezes para o cálculo da média. Independente disso, olhando para a figura 1.3A, podemos notar um crescimento aproximadamente linear do número de

consumidores da plataforma *Steam* (bolinhas azuis). O ajuste linear (linha tracejada preta) rendeu um coeficiente angular de 40518, sugerindo um aumento na taxa de consumo dos jogos da *Steam* de 40518 ao mês. Além de tendência de crescimento, ainda é possível notar um comportamento de oscilação em torno do ajuste linear, muito provavelmente essa oscilação está ligada aos períodos do ano e aos novos jogos lançados. Observando o gráfico, podemos perceber um aumento da popularidade nos meses de férias escolares (janeiro, junho, julho e dezembro); no gráfico representamos os meses de janeiro e julho pelas linhas verticais tracejadas. O pico de usuários mais expressivo (bolinha vermelha) representa janeiro de 2018, nesse mês a *Steam Charts* registrou um número médio de 4209893 jogadores. O grande recorde de janeiro de 2018 se deve também ao sucesso do jogo *Playerunknown's Battlegrounds* [89], que alcançou um número médio de 1584887 de jogadores no mês, a maior média de jogadores no mês entre todos os jogos da *Steam*. Fica claro que janeiro de 2018 foi um *outlier* em meio ao demais meses, de certa forma isso empurrou para cima a tendência de crescimento da plataforma. Assim, em termos de predição, fica um pouco difícil definir como a plataforma continuará crescendo. Nesse sentido, trabalhos futuros podem ser propostos para verificar até quando essa taxa média de crescimento vai se manter, isto é, pode ser estudado se haverá uma saturação em breve ou se o mercado dos jogos continuará crescendo.

Outra importante variável do sistema é o número de jogos disponíveis na base, a esse respeito medidas quantitativas podem ajudar a compreender o sucesso comercial da industria dos jogos. Na figura 1.3B, mostramos o número de jogos disponíveis da *Steam* mês a mês (bolinhas azuis), o resultado é um padrão de crescimento aproximadamente exponencial. Principalmente nos primeiros meses o crescimento exponencial (linha tracejada preta) ajusta-se bem aos dados, porém nos últimos meses a taxa de crescimento dos jogos não parece acompanhar tão bem a exponencial, pois entre os meses 50 e 80 o crescimento é diferente deste padrão. O ajuste foi feito na base 10 e tem sua taxa de crescimento exponencial dado por 0.0145. Em termos numéricos, o parâmetro ajustado parece ser pequeno, porém ao longo do tempo representa um crescimento substancial, dado que em um período de 80 meses foi suficiente para mudar o fator de escala, sendo mais específico, a base tinha 1382 jogos em julho de 2012, já em abril de 2019 contava com 18211. Comparativamente ao número de jogadores, o número bruto de jogos é bem menor mas a taxa de crescimento é muito mais acentuada. Esse crescimento acelerado é um desdobramento direto do sucesso econômico da industria dos jogos eletrônicos nos últimos anos [90]. Não é possível determinar com certeza se essa taxa de crescimento se manterá por muito tempo. Assim como o dado do número de usuários da *Steam* pode ser verificado no futuro, estudos podem ser propostos abordando especificamente as taxas de crescimento. Apenas como uma discussão para aguçar o interesse científico, o estudo do crescimento do número de jogos e jogadores pode ser relacionado com dados de ecologia de população. Nessa situação, os jogos assumem o papel de uma população de indivíduos competindo por recursos, isto é, pela atenção das pessoas.

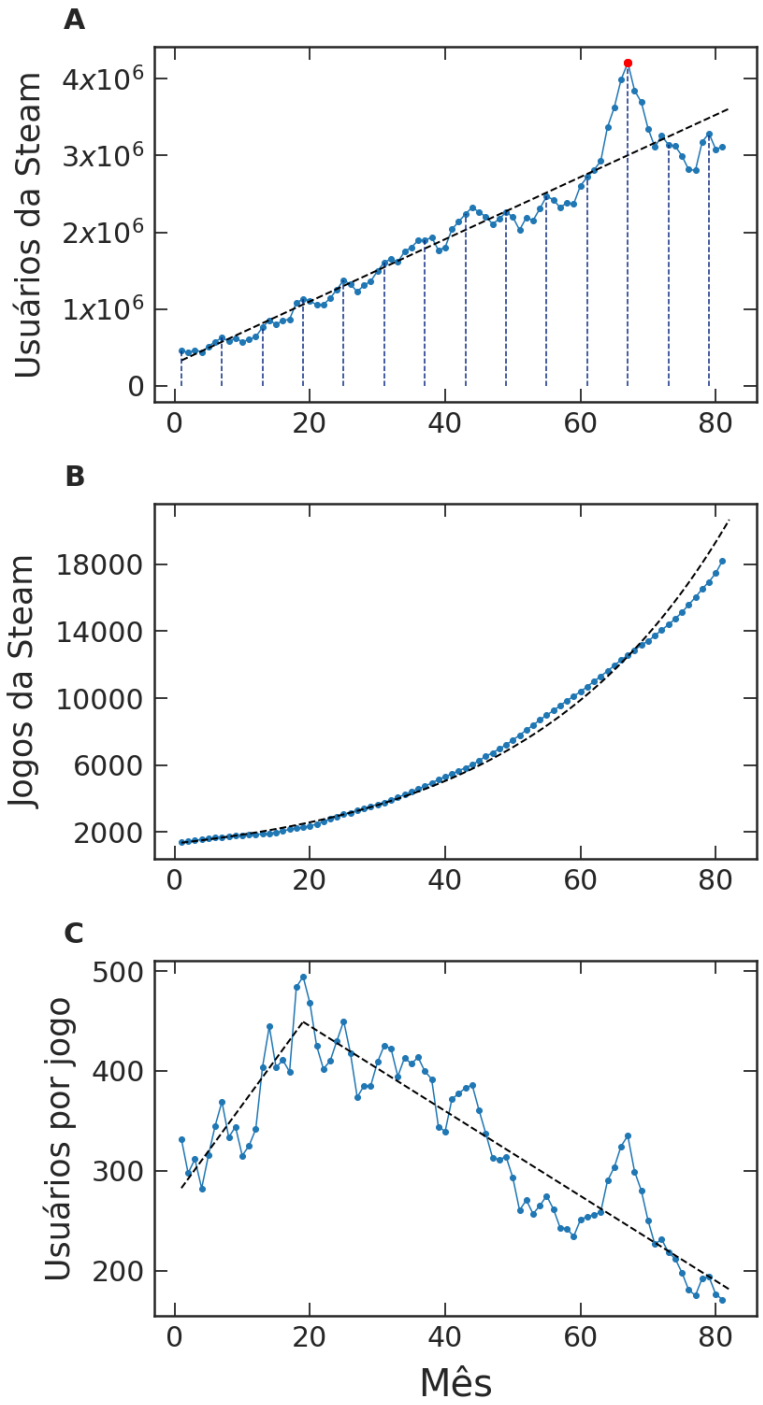

Figura 1.3: Dinâmica temporal do número de usuários e de jogos da *Steam*. (**A**) O crescimento da soma do número médio de usuários de cada jogo (bolinhas azuis) comparado com a reta de ajuste $y = 40518x + 288562$. A bolinha vermelha representa o maior pico de jogadores registrado pela *Steam Charts*, 4209893 de jogadores em janeiro de 2018. (**B**) O número de jogos disponíveis (bolinhas azuis) ajustado por uma função exponencial: $y = 10^{0.014x+3.121}$ (linha tracejada preta). (**C**) Comportamento do número médio de jogadores por jogo (bolinhas azuis) comparado com os modelos lineares (linhas tracejadas pretas): $y = 9.2x + 273$ e $y = -4.2x + 530$.

Seguindo a analogia de ecologia de população, esperamos o surgimento de um regime que possa envolver algum tipo de saturação na evolução da média mensal de jogadores por jogo. Do ponto de vista científico, muitas vezes o crescimento sem limite de uma espécie pode ser interpretado como exponencial, porém ao surgir alguma limitação por parte dos recursos a população apresenta um valor de saturação [91]. A desaceleração do crescimento do número de jogos dos últimos meses pode estar relacionado com esse regime de saturação. Uma maneira de visualizar o efeito limitante da atenção dos jogadores em relação ao número de usuários foi exposto na figura 1.3C. Nela, mostramos o quociente dos valores apresentados nas figuras 1.3A e 1.3B, ou seja, o número médio de usuários por jogo ao longo do tempo. Inicialmente, a proporção de jogadores em relação ao número de jogos estava crescendo até alcançar 494 jogadores por jogo, a partir do mês 19 a média de usuários por jogo entrou em decrescimento. Até o mês 19, o crescimento do número de jogadores (figura 1.3A) se sobressai, levando ao ajuste linear de coeficiente $9,2$ usuários por jogo ao mês. Do mês 19 em diante, a reta de coeficiente angular de $-4,2$ usuários por jogo ao mês foi ajustada, mostrando que o crescimento exponencial do número de jogos (figura 1.3B) superou o crescimento dos jogadores. Esse resultado leva a acreditar que em breve o número de jogos deve promover algum tipo de saturação, pois a atenção dos usuários ficará cada vez mais concorrida.

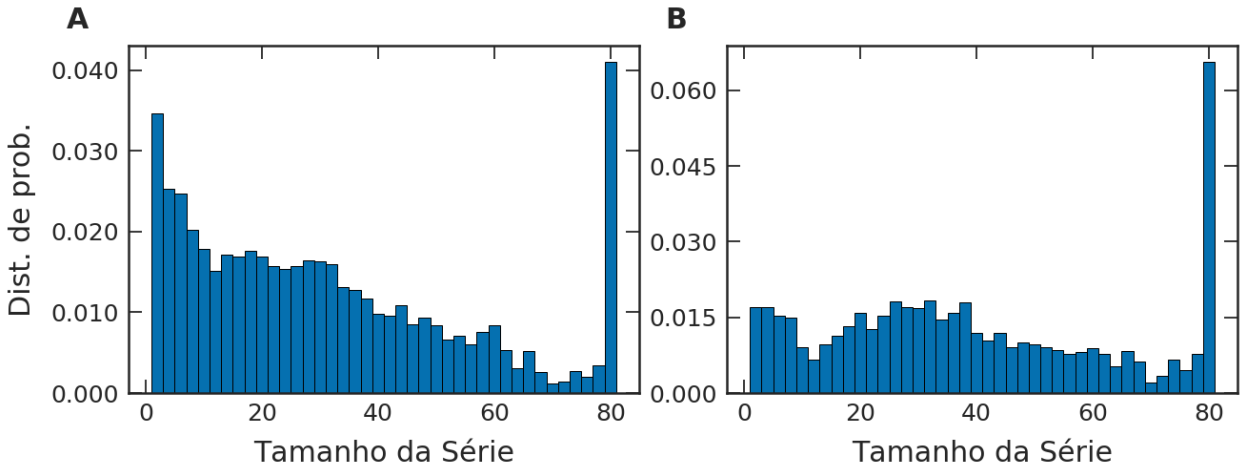

Figura 1.4: Distribuição de probabilidade do tamanho das séries (**A**) Distribuição de probabilidade levando em consideração todos os jogos da *Steam*. (**B**) Distribuição de probabilidade considerando apenas as séries com a média maior que 10 jogadores mensais.

Os dados brutos do número de jogadores e jogos são importantes para a visão geral, porém utilizando uma abordagem de distribuição de probabilidade podemos obter informações mais específicas. Nessa direção, fizemos a distribuição de probabilidade do tamanho das séries temporais, veja figura 1.4. No parágrafo anterior, discutimos sobre a concorrência pela atenção dos usuários, na figura 1.4 mostramos quantitativamente a proporção de jogos lançados que tiveram algum sucesso. A figura 1.4A mostra a distribuição de probabilidade do tamanho das séries temporais para todos os jogos da *Steam*, já a figura 1.4B mostra apenas

os jogos com a média da série temporal maior que 10 jogadores, ambos os histogramas foram construídos usando 40 *bins*. Comparando os dois histogramas para os jogos mais antigos (série maior que 40 meses), não há quase nenhuma diferença. Isso mostra que os jogos mais antigos não foram esquecidos pelo público. Vale dizer que a última barra dos dois gráficos são grandes pois elas acumulam os jogos lançados antes de Julho de 2012. No entanto, o contexto muda ao compararmos as séries menores, claramente notamos que uma proporção de jogos lançados recentemente não alcançam sequer algum sucesso. Por exemplo, observando a primeira barra que representa os meses um e dois na figura 1.4A, a probabilidade chega a aproximadamente $3,4\%$ enquanto que na figura 1.4B aproxima-se de $1,7\%$. A falta de sucesso pode desacelerar o crescimento do número de jogos lançados, pois a popularidade está de alguma forma ligada ao sucesso econômico e uma vez que os jogos têm lucro deve haver um desinteresse por parte de investidores e desenvolvedores.

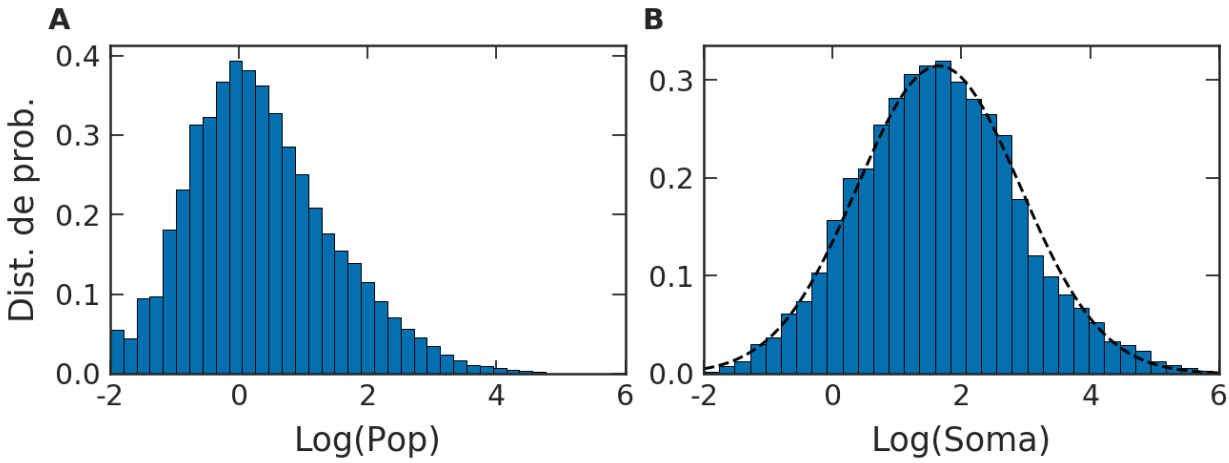

Figura 1.5: Distribuição de probabilidade da popularidade dos jogos. (**A**) Distribuição de probabilidade do número médio de jogadores em cada mês de todos jogos juntos. (**B**) Distribuição de probabilidade da soma de cada série temporal até o último mês da nossa base de dados (em azul) e um ajuste normal de média 1.65 e desvio padrão 1.27 (linha tracejada preta).

Seguindo na investigação das probabilidades, o próximo passo foi mostrar quantitativamente como a popularidade se distribui nos jogos. A distribuição de popularidade é muito desigual, pois existem alguns poucos jogos que têm muitos jogadores, enquanto a grande maioria tem poucos jogadores. Devido a essa grande discrepância tomamos o logaritmo na base 10 das séries temporais, e como o logaritmo diverge em zero acabamos eliminando os valores zeros presentes no dado. A transformação logarítmica encurta a diferença de escala, dessa forma facilita comparar desde os jogos mais populares até os jogos menos populares. A figura 1.5A mostra o histograma da distribuição de probabilidade da popularidade de todos os jogos em todos os meses, a distribuição unimodal mostra que a grande maioria dos meses de cada jogo tem entre $0,1$ e 10 jogadores mensais médio. Os menores valores de

popularidade não têm um início mais próximo da probabilidade zero por um limite inferior da própria variável. Para entender esse fato, basta lembrar que se ninguém acessa o jogo ao longo do mês a media mensal fica em zero, porém se uma pessoa acessa o jogo em um mês de 31 dias a média mensal será 1/31, portanto o menor valor acima do zero que a variável pode assumir é 1/31. Uma forma de minimizar o efeito de acúmulo nos pequenos valores seria adotar uma variável que se distribui em um espectro mais amplo, para isso adotamos a soma da popularidade de cada jogo ao longo do tempo, veja figura 1.5B. Na construção dessa figura, utilizamos a soma da série temporal original para cada jogo, após a soma é que foi tomado o logaritmo. A figura 1.5B mostra uma distribuição mais simétrica para o logaritmo da soma, aproximadamente uma distribuição normal de média 1.65 e desvio 1.27. Como estamos trabalhando com o logaritmo da soma, a distribuição da soma é bem aproximada por uma distribuição lognormal. Por exemplo, distribuições do tipo lognormal já foram empregadas em estudos de abundância de uma espécia em ecologia de população. Nesse contexto, o modelo estocástico de crescimento exponencial proposto por MacArthur prevê que a abundância populacional tem distribuição lognormal ao longo do tempo [92]. Isso é outra evidência indicando que de fato os jogos podem ser interpretados como "indivíduos" competindo por recursos.

As distribuições de probabilidade foram importantes para entender aspectos da natureza de cada variável separadamente, mas outras informações pertinentes podem ser obtidas da correlação entre as variáveis. Em especial, a figura 1.6 mostra como se comportam a média e o desvio padrão das séries temporais em relação ao tamanho delas. Ambos os gráficos foram construídos na escala logarítmica, por conta disso foram descartados os jogos com média igual à zero, isto é, descartamos 781 jogos. Já foi exibido na figura 1.4 que muitos jogos lançados recentemente não alcançam uma popularidade mínima de 10 jogadores por mês. Nessa mesma direção, o gráfico 1.6A mostra outro aspecto quantitativo sobre o efeito da idade do jogo em sua popularidade. Cada par ordenado representa o tamanho da série temporal no eixo x e sua média no eixo y. A linha azul escura foi obtida juntando todas as séries de mesmo tamanho e fazendo a média, depois ainda foi feito a média em uma janela móvel de tamanho 3 para suavizar. Em média, os jogos mais antigos parecem ter uma popularidade maior, porém um *outlier* foi detectado próximo do mês 25, representado no gráfico pela bolinha vermelha, trata-se do jogo *Playerunknown's Battlegrounds* com a popularidade média de 683655 jogadores mensais. Comparativamente, a segunda maior popularidade média é de 457356 jogadores mensais do jogo *Dota 2*, representado pela bolinha verde. A linha tracejada preta representa um ajuste exponencial com taxa de crescimento igual à 0.015. Esse ajuste deixa bem claro a diferença de uma ordem de grandeza na popularidade média de jogos lançados ao longo desses 81 meses.

O gráfico 1.6B dá uma ideia da dispersão em relação a média de cada série. No eixo x, mantemos o tamanho da série e, no eixo y, fizemos o desvio padrão da série. Vale dizer que as

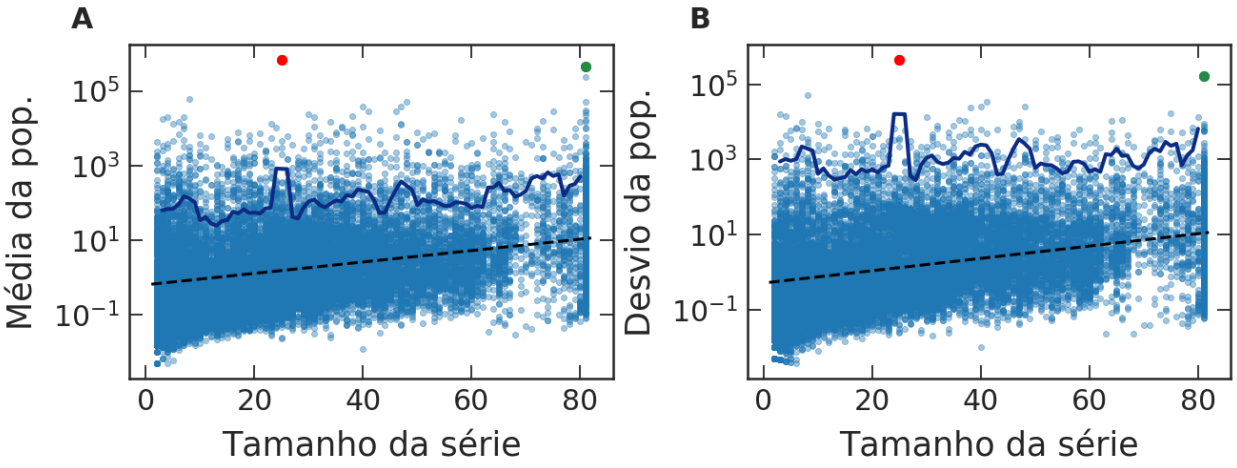

Figura 1.6: Relação da média e do desvio padrão comparados ao tamanho da série. (**A**) O par ordenado de cada ponto representa o tamanho da série e sua média de popularidade, a linha em azul escuro é a média das séries de mesmo tamanho juntas, a linha tracejada é o ajuste exponencial dado por $y = 10^{0.015x-0.2}$, a bolinha vermelha e a verde são os jogos *Playerunknown's Battlegrounds* e *Dota 2*, respectivamente. (**B**) Cada ponto contém a informação do tamanho da série e seu desvio padrão, a linha escura é o desvio padrão das séries de mesmo tamanho juntas, a linha preta é o ajuste $y = 10^{0.016x-0.3}$, as bolinhas em destaque são os mesmos dois jogos ressaltados em (**A**).

duas bolinhas em destaque, a vermelha e a verde, são os mesmos jogos citados anteriormente. Analogamente ao gráfico 1.6A, a linha azul escuro em 1.6B representa o desvio padrão de todas as séries de mesmo tamanho juntas e, também, foi tomada a média em uma janela móvel de tamanho 3. Diferentemente do caso ao lado, dessa vez a linha escura se manteve aproximadamente constante, apesar de algumas flutuações. Obviamente, o pico em torno do *outlier* ficou mais acentuado justamente pela forma matemática do desvio padrão favorecer a diferença entre os pontos. A constância do desvio padrão das séries juntas contrasta com o ajuste exponencial dos dados com taxa de crescimento de 0.016, representado pela linha tracejada. Nossa explicação para o contraste é a diferença do número de pontos nas diferentes regiões do gráfico, ou melhor dizendo, a dispersão em relação a média das séries de mesmo tamanho é constante, porém tem muito mais séries curtas do que longas. O contraste surge quando o cálculo do desvio é ponderado pelo número de pontos, ao passo que no cálculo do ajuste cada ponto contribui igualmente. Juntando as informações dos gráficos 1.6A e 1.6B, notamos que está mais difícil um jogo lançado atualmente concorrer a altura com jogos já consolidados, pois existe uma relação crescente entre quanto tempo um jogo é comercializado e sua popularidade média. Por outro lado, a concorrência entre os jogos só está maior por causa da maior quantidade de jogos lançados e não pela desigualdade de popularidade, pois a proporção de jogos bem sucedidos e mal sucedidos manteve-se a mesma ao longo do tempo.

Após mostrar algumas relações entre as características gerais das séries, os próximos passos são focados em expandir a investigação para as variáveis categóricas. O primeiro aspecto a

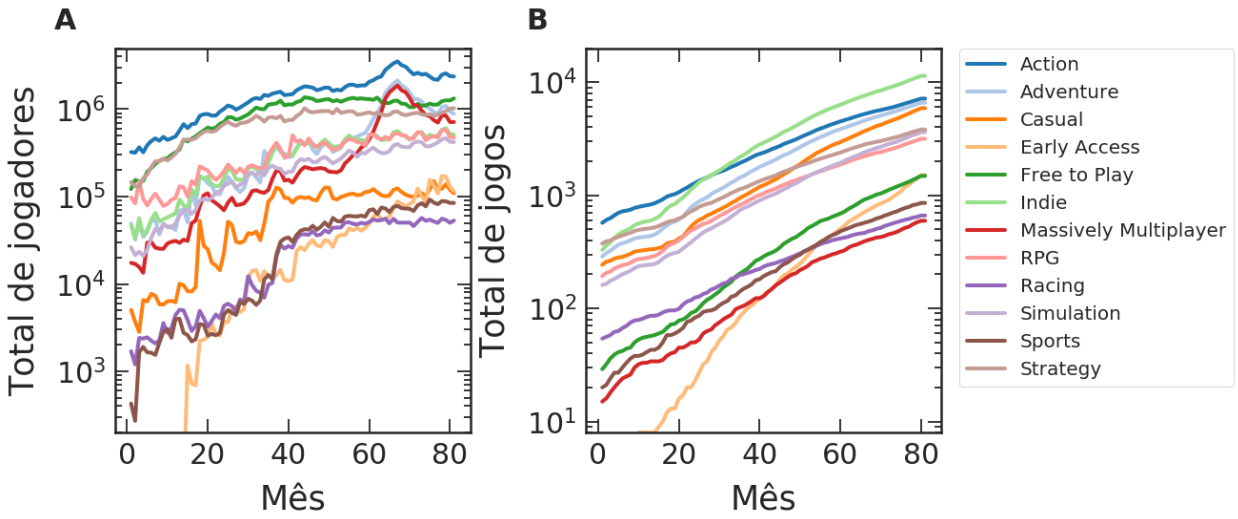

Figura 1.7: Número de jogos e jogadores separados por gênero. (**A**) Crescimento do número de jogadores por gênero ao longos dos 81 meses. (**B**) Crescimento do número de jogos de cada gênero com o passar dos 81 meses, cada gênero foi representado por uma cor correspondente a legenda ao lado.

ser mostrado está relacionado aos gêneros dos jogos. Nesse sentido, o gráfico 1.7A representa o número de jogadores ao longo do tempo de cada gênero, levando em consideração todos os 12 gêneros oficiais da *Steam*. Quando nos referimos à jogadores de um gênero, estamos dizendo o número total de pessoas que jogaram os jogos de um determinado gênero. No gráfico 1.7A, percebemos dois grupos de popularidade: os gêneros que começaram com mais de 10000 jogadores e ao fim dos 81 meses tem mais de 300000 jogadores (*Action*, *Adventure*, *Free to Play*, *Indie*, *Massively Multiplayer*, RPG, *Simulation*, *Strategy*), e outro grupo que começa com menos de 10000 jogadores e termina com valores próximos de 100000 jogadores (*Casual*, *Early Access*, *Racing*, *Sports*). Observando o gráfico, notamos que todos os gêneros tiveram um grande crescimento até o mês 40, após esse mês todos tiveram uma desaceleração na taxa de crescimento, exceto os gêneros *Action*, *Adventure* e *Massively Multiplayer* que tiveram um pico devido ao *outlier* mostrado na figura 1.6. Diferentemente do número de jogadores por gênero, o número de jogos teve uma taxa de crescimento exponencial aproximadamente constante ao longo dos meses, gráfico 1.6B. Em termos de classificação, os gêneros dos jogos apresentam uma divisão clara de dois grupos, os que lançaram menos de 100 jogos no início da base (*Early Access*, *Free to Play*, *Massively Multiplayer*, *Racing*, *Sports*); e o grupo com os gêneros que haviam lançados mais de 100 jogos em julho de 2012 (*Action*, *Adventure*, *Casual*, *Indie*, RPG, *Simulation*, *Strategy*). Apesar de serem identificados dois grupos nos dois gráficos, não há uma relação direta entre o número de jogos e o número de jogadores. Por exemplo: o gênero *Action* está entre os mais populares e com o segundo maior número de jogos lançados, por outro lado, o gênero *Massively Multiplayer* está entre os mais populares e é o gênero com menos jogos lançados. A respeito das taxas de crescimento, apesar

do crescimento do número de jogadores ser diferente dos jogos, comparando internamente, tanto os jogadores quanto os jogos crescem mais ou menos na mesma proporção entre si.

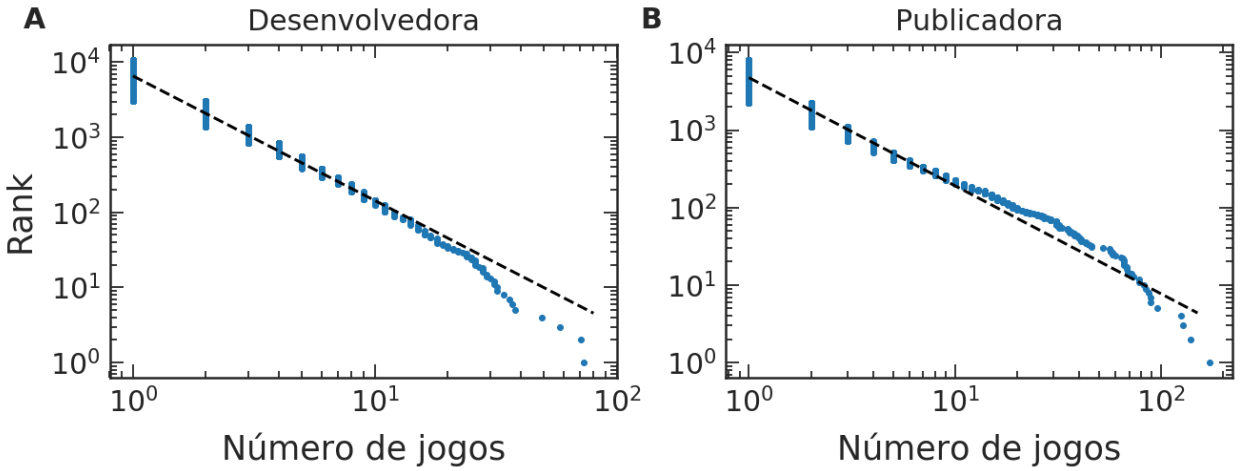

Figura 1.8: *ranking* do número de jogos lançados pelas desenvolvedoras e publicadoras. (**A**) Os pontos azuis representam os dados do *ranking* de cada desenvolvedora considerando o número de jogos lançados, enquanto que a linha tracejada preta é uma lei de potência, $y = 6563x^{-1.7}$, colocada como guia para os olhos. (**B**) O mesmo tipo de gráfico de ranking da (**A**), porém separado por publicadora e a linha tracejada é dada por $y = 4745x^{-1.4}$.

Seguindo a análise das variáveis categóricas, os dois últimos aspectos a serem levados em consideração são as desenvolvedoras e publicadoras. A figura 1.8 expõe o ranking de cada desenvolvedora e publicadora em relação ao número de jogos lançados. Explicando de forma detalhada, nós contamos o número de jogos que cada desenvolvedora/publicadora lançou até Abril de 2019, a partir disso classificamos de forma decrescente. O gráfico 1.8A mostra o *ranking* das desenvolvedoras e o gráfico 1.8B o das publicadoras. Ambos os gráficos foram construídos em escala logarítmica nos dois eixos, isso se deve a boa aproximação por Lei de Potência dos dados. A linha tracejada foi colocada como guia para os olhos, ela mostra a boa aproximação para as desenvolvedoras/publicadoras de poucos jogos lançados, no entanto, a medida que consideramos os melhores classificados a aproximação vai ficando ruim. A respeito do acúmulo de pontos na parte de poucos jogos lançados, isso se deve a natureza discreta do dado que não permite uma transição contínua. Deve ser dito que comportamentos Leis de potência são muito comuns na natureza e já foram relatados em diversos sistemas, basicamente elas dizem que as menores ocorrências ocorram com muito mais frequência. Muitas vezes na literatura científica, uma Lei de potência em gráficos de *ranking* tem o expoente de -1 e nesse caso é comumente chamada de Lei de Zipf [93, 94]. Por exemplo, o *ranking* da frequência das palavras mais comuns de língua inglesa segue aproximadamente uma Lei de Zipf [95]. No entanto, nossos dados mostram um expoente um pouco menor, -1.7 no *ranking* das desenvolvedoras (figura 1.8A) e -1.4 no ajuste das publicadoras (figura 1.8B). A magnitude dos expoentes dá uma noção da desigualdade, pois

a inclinação da reta expressa a taxa que o ranking varia de acordo com o número de jogos lançados pelas desenvolvedoras/publicadoras. Portanto, nesse sentido as desenvolvedoras e publicadoras são mais desiguais quando comparadas com outros sistemas naturais já citados.

1.3 Filtros

Nesta seção, fazemos algumas considerações preliminares acerca das séries utilizadas na análise do capítulo seguinte. Baseado nos resultados das demografias, propomos dois critérios para filtrar os dados, o primeiro está relacionado a popularidade média dos jogos e o outro com o tamanho da série temporal. De acordo com a figura 1.4, grande parte dos jogos lançados recentemente não apresentam uma popularidade média mínima de 10 jogadores mensais. Partindo do pressuposto que queremos analisar a popularidade dos jogos com certa relevância, acabamos fixando um limite inferior de 10 jogadores mensais médio. Devemos dizer que não utilizamos nenhum método analítico para decidir o valor desse limite, porém tentamos escolher um valor razoável de jogadores mensais e que não descartasse um número muito grande de séries. Além do número mensal médio de jogadores, ainda filtramos as séries que tenham no mínimo um ano, ou seja, no mínimo 12 observações. Esse número também foi fixado sem um método analítico. Todavia, a escolha de um limite para os tamanhos das séries temporais tem uma razão, argumentamos que inicialmente a popularidade dos jogos deve variar muito rapidamente, dessa forma, como o atual trabalho se propõe a determinar os padrões da dinâmica temporal da popularidade dos jogos, isso deixa implícito que precisamos levar em consideração as séries estacionárias. Ao aplicar os dois filtros, obtemos um conjunto de 2972 séries temporais.

CAPÍTULO 2

Resultados

Neste capítulo, apresentamos os resultados da análise de *clustering* aplicado as séries temporais dos jogos da *Steam*. O capítulo anterior apresentou resultados demográficos a respeito do dado, este capítulo avança na investigação das séries temporais. Inicialmente, mostramos alguns exemplos das séries dos jogos, baseado nelas discutimos transformações matemáticas que poderiam ajudar a agrupar as séries mais similares. Após preparar as séries, definimos a medida de dissimilaridade utilizada e o critério de agrupamento, ambos foram escolhidos pensando em agrupar as séries com padrões de dinâmicas parecidas. Visando uma melhor compreensão e um exemplo ilustrativo, dedicamos uma parte do trabalho à uma aplicação do método de agrupamento em um dado sintético. Em seguida, aplicamos a análise de *clustering* nos dados reais obtidos da *Steam*. Desse procedimento, tiramos os principais resultados do trabalho, isto é, os padrões da dinâmica de popularidade dos jogos. Na parte final do capítulo, expomos algumas informações demográficas de cada grupo e a sumarização das características de cada grupo.

2.1 Apresentação das séries temporais

Inicialmente, para fins de contextualização, vamos apresentar alguns exemplos de séries temporais presentes no dado. Até aqui ainda não tínhamos abordado as séries propriamente ditas, apenas algumas características demográficas relacionadas a elas. Um pouco dessa procrastinação se deve a dificuldade em fazer uma representação gráfica global do dado. Assim, fica impossível representar todas as séries em um único gráfico de modo inteligível, pois a grande quantidade de séries, a diferença de magnitude e a flutuação de cada série deixaria o gráfico muito poluído. Porém, precisávamos pelos menos ilustrar um conjunto de séries

temporais, apenas para ver as formas da dinâmica de algumas delas. Nesse contexto, elegemos os 10 jogos mais populares da *Steam* e ilustramos suas séries temporais na figura 2.1. Apenas levando em consideração a ordem de grandeza da popularidade, mesmo entre os 10 jogos mais populares notamos dois grupos distintos. O grupo dos mais populares apresentam o número de jogadores próximo de 500000 por mês (*Dota 2*, *Counter-Strike: Global Offensive*, *Playerunknown's battlegrounds*), já o segundo grupo varia nas dezenas de milhares de jogadores mensais médio (*Team Fortress 2*, *Grand Theft Auto V*, *Sid Meier's Civillization V*, *Warframe*, *Garry's Mod*, *ARK: Survival Evolved*, *The Elder Scrolls V: Skyrim*). Pensando apenas no número bruto de jogadores, entre os 10 maiores jogos já existe bastante diferença, pois precisamos construir o gráfico em escala logarítmica para todos ficarem enquadrados. Avaliando a magnitude seria uma forma de agrupar as series, porém não parece a melhor de agrupamento. Nesse sentido, pode ser mais interessante detectar os padrões de dinâmica de cada série, dessa forma entendendo como as séries se comportam sem levar em consideração a magnitude em si. Observando o exemplo da figura 2.1, os grupos de padrões de dinâmica não são tão claros, pois os comportamentos das séries dos 10 maiores jogos não são tão parecidos. Aparentemente, os jogos lançados têm sua popularidade crescente nos primeiros meses e depois de um tempo eles parecem estabilizar em torno de um certo valor. Entretanto algumas apresentam um certo padrão de crescimento, como o caso do *Warframe*, enquanto o *The Elder Scrolls V: Skyrim* está em decaimento nos últimos 40 meses, ou seja, levando em consideração as 10 maiores não é possível elaborar um critério de agrupamento. Além delas apresentarem diferentes tendências, outro fator que atrapalha bastante a tarefa de agrupar são as flutuações. Como construímos o gráfico em escala logarítmica, as flutuações acabaram sendo suavizadas nos jogos mais populares, o que torna a comparação entre as séries ainda mais trabalhosa. Observando as séries dos 10 jogos mais populares, foi possível entender as dificuldades de comparar tais séries, nesse sentido algumas estratégias podem ser adotadas para ajudar a agrupas as séries.

Após deixar bem claro as dificuldades de comparar as séries, é conveniente propor transformações matemáticas que devem nos ajudar a lidar com a séries. Nesse contexto, propomos uma segunda tentativa de ilustrar um conjunto de séries temporais, veja figura 2.2. Nessa figura, representamos três pares de séries temporais, cada par aparenta representar um tipo de padrão. O primeiro par de séries, figuras 2.2A e 2.2B, mostra duas séries que parecem fazer sucesso em sua data de lançamento, porém imediatamente após o lançamento há um decaimento brusco até valores pequenos, cerca de 20% da popularidade inicial, após a queda vertiginosa ao longo dos cinco primeiros meses, as séries entram em um regime de flutuação em torno de um certo valor. O segundo par, figuras 2.2C e 2.2D, mostra um comportamento totalmente diferente, exibe o crescimento aproximadamente linear em todo o intervalo registrado. O terceiro par mostra outro tipo de comportamento, o comportamento de morro, ou seja, a popularidade deles cresceram por um período, mais ou menos por 40 meses, e

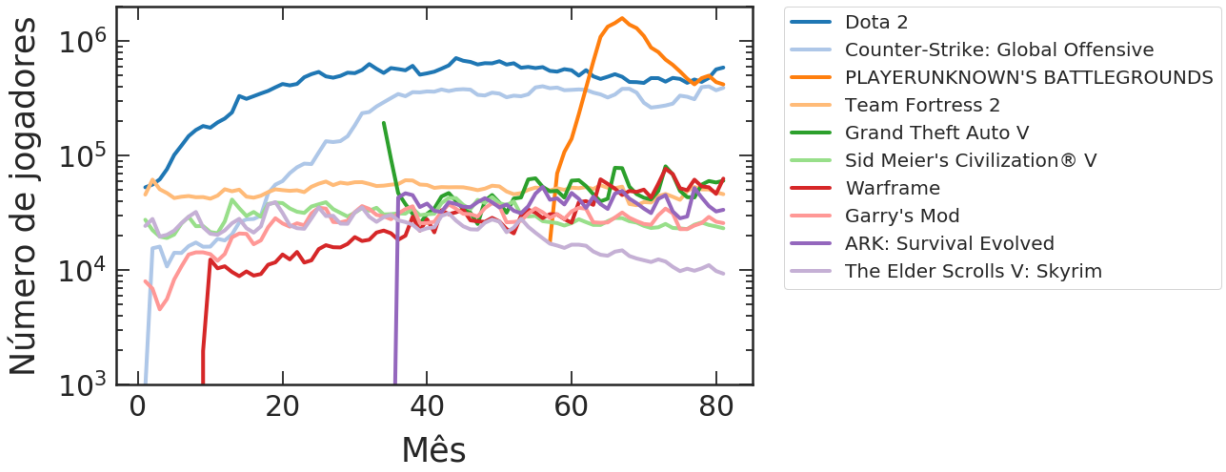

Figura 2.1: Dinâmica temporal do número de jogadores dos 10 jogos mais populares da *Steam*. Cada cor representa um jogo entre os 10 maiores jogos da *Steam*, o gráfico mostra a dinâmica temporal da popularidade de tais jogos ao longo de todo o período de 81 meses, logo ao lado encontra-se a legenda com a cor correspondente ao nome de cada jogo.

depois passaram por um decrescimento. Após mostrarmos três exemplos de dinâmicas da popularidade dos jogos, vamos utilizar essas mesmas séries para exemplificar as transformações matemáticas empregadas. Inicialmente, vamos definir o conjunto de todas as séries temporais como S, matematicamente:

$$S = \begin{Bmatrix} {s_0}^0,\ {s_1}^0,\ {s_2}^0 \cdots \\ {s_0}^1,\ {s_1}^1,\ {s_2}^1 \cdots \\ {s_0}^2,\ {s_1}^2,\ {s_2}^3 \cdots \\ \vdots \\ {s_0}^l,\ {s_1}^l,\ {s_2}^l \cdots \end{Bmatrix}, \tag{2.1}$$

em que ${s_i}^j$ é a i-ésima observação da j-ésima série. O número total de séries é $l = 2972$ séries, já o número de observações de cada série pode variar de 11 à 80, pois cada série tem no máximo 81 observações e já foi dito que retiramos as séries com menos de 12 observações.

O primeiro procedimento adotado foi acumular as séries, o principal objetivo foi diminuir o ruído do dado. Matematicamente, definimos as séries acumuladas como C e da mesma forma adotamos:

$$C = \begin{Bmatrix} {c_0}^0,\ {c_1}^0,\ {c_2}^0 \cdots \\ {c_0}^1,\ {c_1}^1,\ {c_2}^1 \cdots \\ {c_0}^2,\ {c_1}^2,\ {c_2}^3 \cdots \\ \vdots \\ {c_0}^l,\ {c_1}^l,\ {c_2}^l \cdots \end{Bmatrix}, \tag{2.2}$$

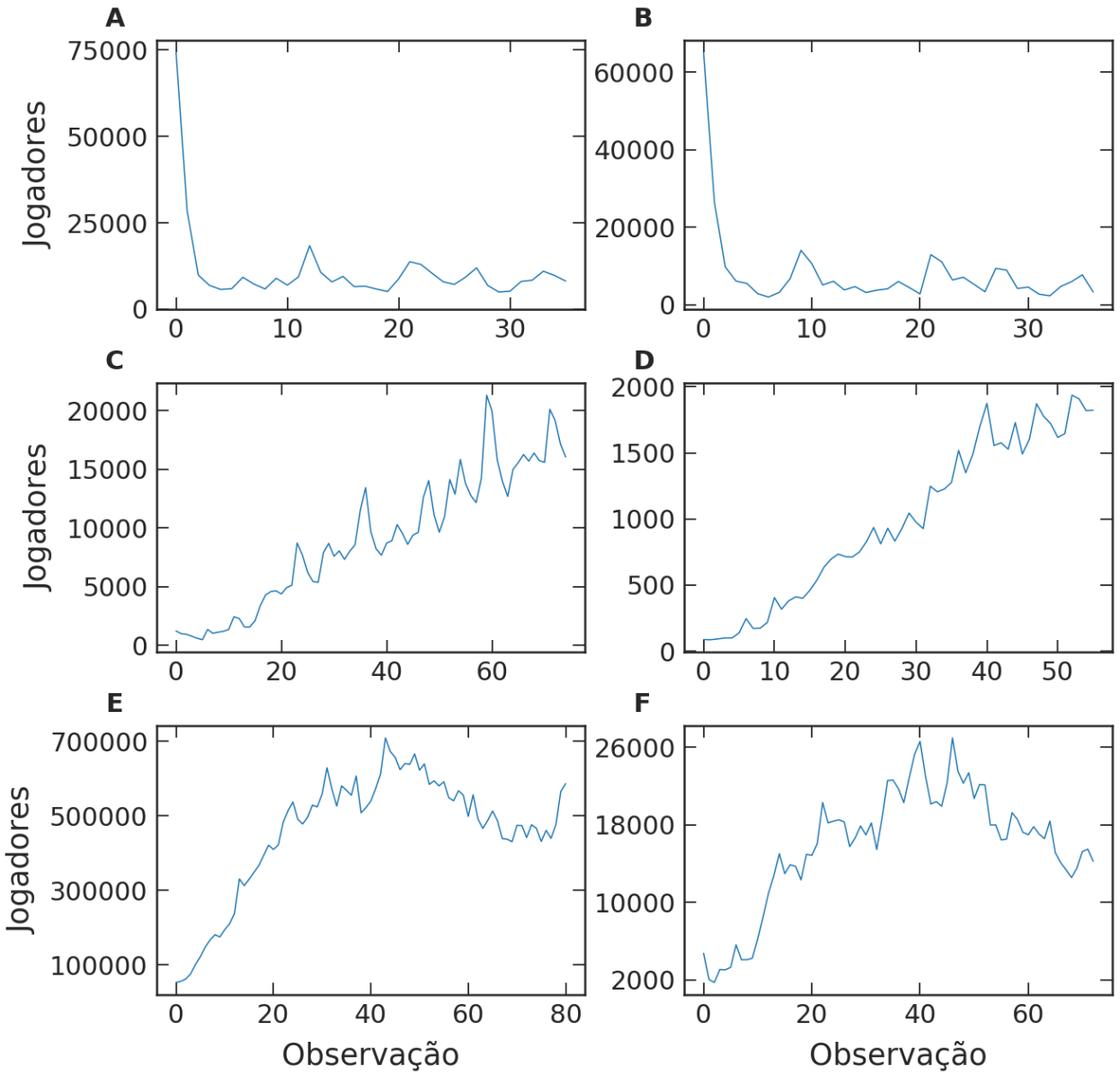

Figura 2.2: Exemplos de séries temporais do dado obtido da *Steam*. (**A**) e (**B**) representam duas séries que têm um comportamento de decrescimento abrupto nos primeiro meses e depois estabiliza em torno de um dado valor. (**C**) e (**D**) mostram duas séries com tendência de crescimento em todo o intervalo. (**E**) e (**F**) são um par de séries com uma tendência de crescimento até aproximadamente metade do intervalo e depois apresentam uma pequena queda ao longo da segunda metade do intervalo.

em que cada ${c_i}^j$ pode ser definido como,

$$ {c_i}^j = \sum_{m=0}^{i} {s_m}^j. \tag{2.3} $$

As séries acumuladas são muito úteis para diminuir os ruídos, podemos conferir a mudança na figura 2.3. Nessa figura, representamos os mesmos pares de séries descritos na figura

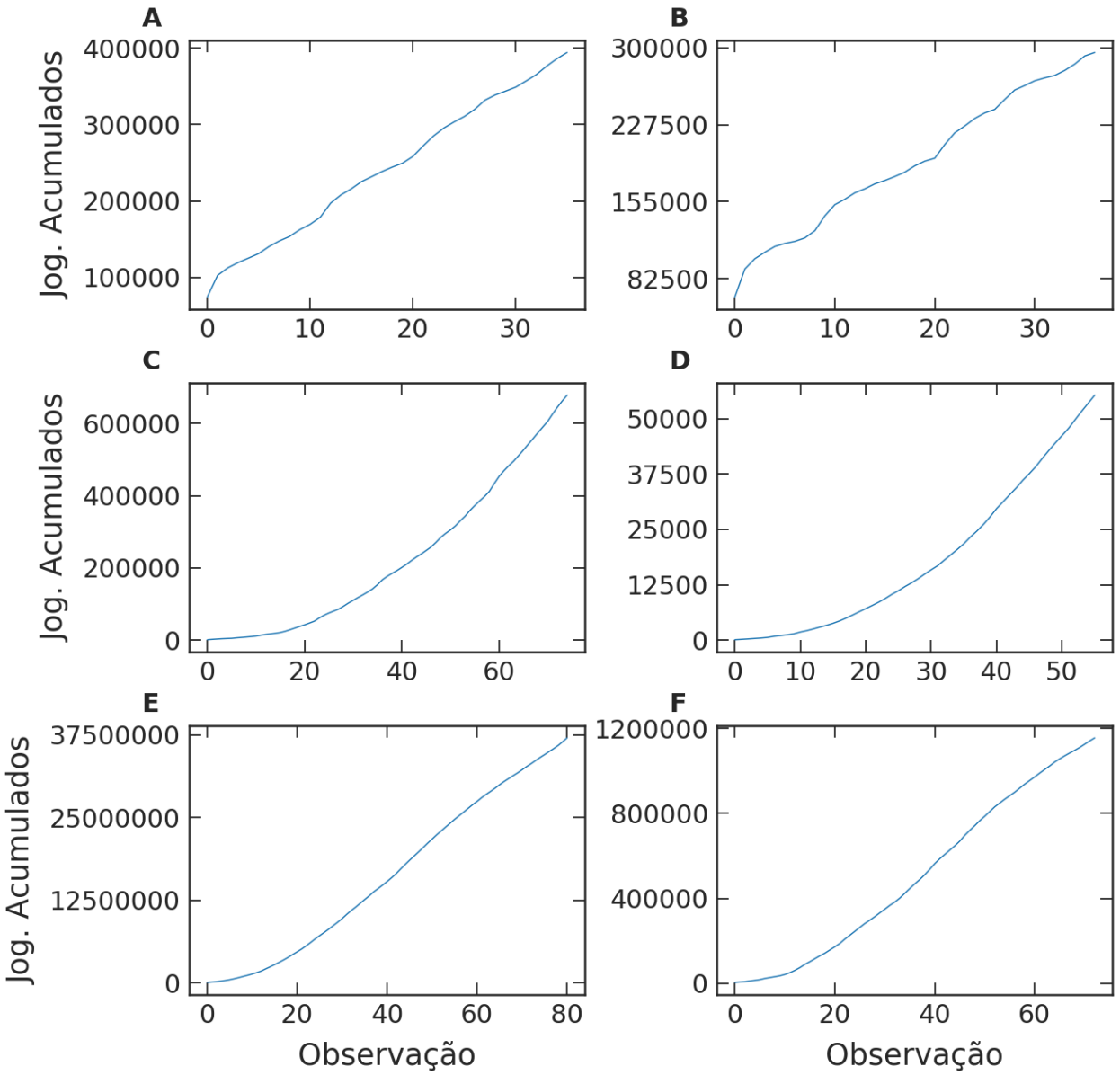

Figura 2.3: Exemplos de séries temporais acumuladas. As séries acumuladas nesta figura são os mesmos exemplos da figura 2.2. (**A**) e (**B**) apresentam um crescimento com pequenos degraus ao longo do tempo. (**C**) e (**D**) é o par de séries com um crescimento aproximadamente parabólico. (**E**) e (**F**) mostra um comportamento de crescimento apresentando um ponto de inflexão no meio do intervalo.

2.2, porém agora utilizamos a séries acumuladas. O efeito negativo é que visualmente todas as séries ficaram mais parecidas, pois todas ficaram com o comportamento crescente. Mesmo assim, com um olhar mais atento, podemos perceber que a forma de crescimento é diferente para os diferentes pares. O primeiro par, figuras 2.3A e 2.3B, mostra uma espécie de degraus, em certos pontos da curva há um crescimento abrupto curtos. Já as figuras 2.3C e 2.3D, o crescimento é aproximadamente parabólico, apesar de algumas flutuações suaves. O terceiro par já mostra uma tendência diferente, aparentemente existe um ponto

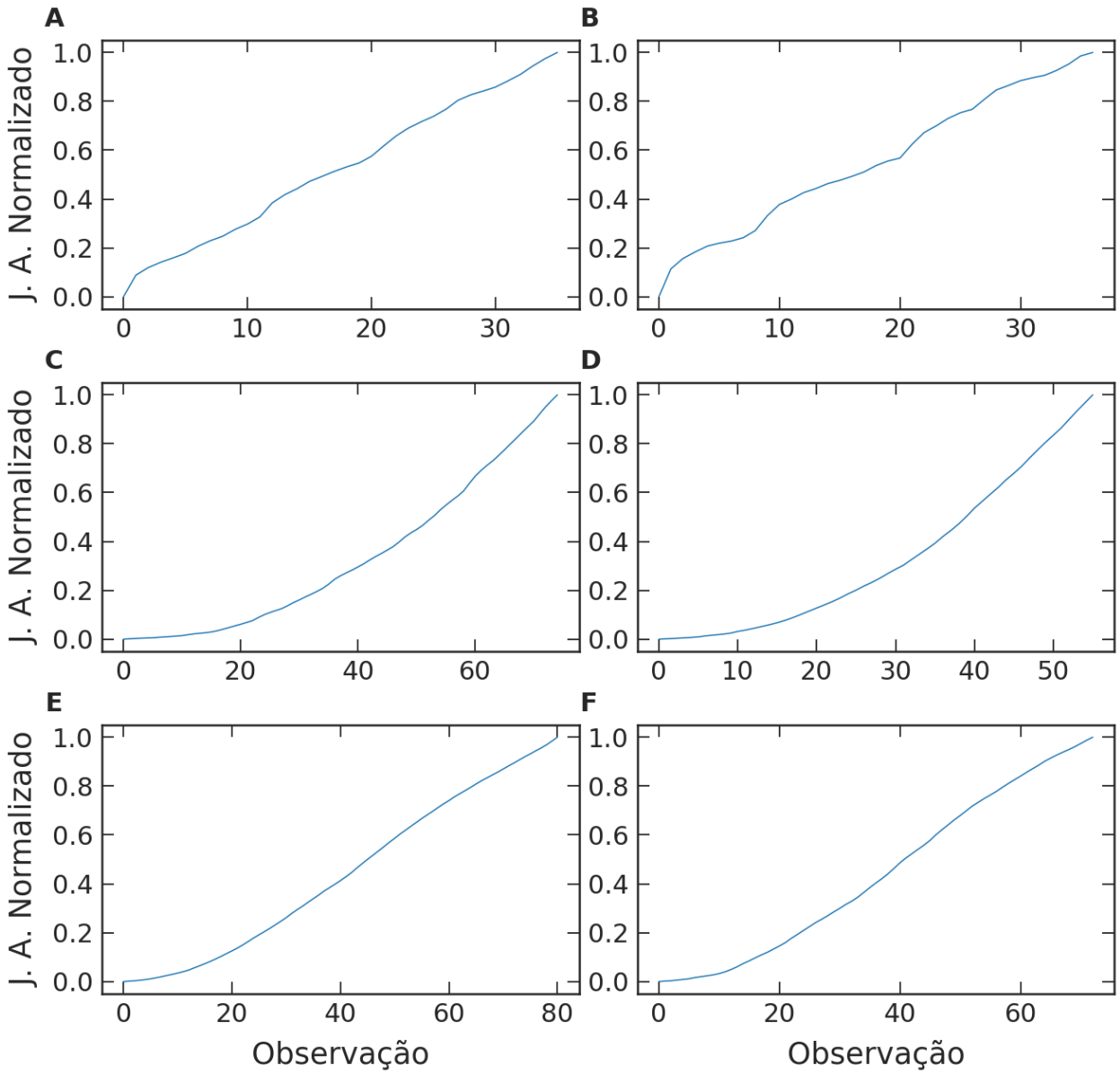

Figura 2.4: Exemplos de séries temporais acumuladas e normalizadas. As séries acumuladas e normalizadas nesta figura são os mesmos exemplos da figura 2.2 e 2.3, além disso as séries foram normalizadas utilizando a relação 2.5.

de inflexão aproximadamente no meio do intervalo dos meses, pois inicialmente ela tem um crescimento acentuado com concavidade para cima, enquanto que na segunda metade dos gráficos as curvas tendem a ter um crescimento mais contido com a concavidade para baixo. Observando a diferença da figura 2.2 para a figura 2.3, ficou muito nítido que a flutuação foi bem suavizada, no entanto ainda existe um fator atrapalhando, a diferença de magnitude das popularidades.

Em meio a isso, consideramos uma segunda transformação matemática, normalizamos todas as séries. Consideramos o conjunto de todas as séries acumuladas e normalizadas como

N, dado por:

$$N = \begin{Bmatrix} {n_0}^0,\ {n_1}^0,\ {n_2}^0 \cdots \\ {n_0}^1,\ {n_1}^1,\ {n_2}^1 \cdots \\ {n_0}^2,\ {n_1}^2,\ {n_2}^3 \cdots \\ \vdots \\ {n_0}^l,\ {n_1}^l,\ {n_2}^l \cdots \end{Bmatrix}, \tag{2.4}$$

em que cada elemento da série é calculado da seguinte forma,

$${n_i}^j = \frac{{c_i}^j - \min_m\{{c_m}^j\}}{\max_m\{{c_m}^j\} - \min_m\{{c_m}^j\}}. \tag{2.5}$$

Deve estar claro que o termo $\min_m\{{c_m}^j\}$ representa o menor valor da série c^j que será o primeiro valor, analogamente $\max_m\{{c_m}^j\}$ representa o máximo da série que é o último termo. Esse tipo de normalização força que todas as séries comecem em zero e terminem em um, dessa forma resolvemos o problema de comparar séries de diferentes escalas de popularidade. A figura 2.4 mostra os pares das séries acumuladas e normalizadas. Comparativamente com a figura 2.3, o comportamento geral não mudou, apenas houve uma mudança de escala obrigando todo o dado a variar no mesmo intervalo de popularidade. Considerando as séries acumuladas e normalizadas, acreditamos estarmos prontos para investigar os padrões de dinâmica de popularidade dos jogos.

2.2 Distância de Euclides sem viés

O seguinte passo envolve a escolha de um critério para comparar as séries, ou seja, nesta seção discutimos a medida de dissimilaridade empregada neste trabalho. Pesquisando na literatura, encontramos inúmeras medidas de distâncias para comparar séries temporais, cada uma apresenta uma relação matemática que tende a maximizar algum tipo de diferença entre as séries. O principal objetivo do atual trabalho é detectar os padrões de dinâmica temporal da popularidade dos jogos, isto é, estamos preocupados em determinar as formas das séries temporais mais parecidas. Em nosso caso, o fato das séries estarem acumuladas garante que todas têm padrões apenas de crescimento, dessa forma escolher uma distância que compare apenas a forma das séries não é uma boa opção. Esse fato exclui uma grande quantidade de medidas de dissimilaridades conhecidas, mas ainda restam muitas opções. Aproveitando o fato das séries utilizadas estarem normalizadas, podemos pensar em uma medida de distância que compara os valores para cada observação. Dito de outra forma, como todas as séries acumuladas e normalizadas são crescentes, comparando os valores de cada observação ao longo de toda a série, podemos determinar a velocidade com que cada série cresce. Levando isso em consideração, escolhemos a distância mais simples e popular, a

distância de Euclides. Considerando duas séries temporais, n^u e n^v, a distância de Euclides entre essas duas séries é definida como

$$D(n^u, n^v) = \sqrt{\sum_k ({n_k}^u - {n_k}^v)^2}. \tag{2.6}$$

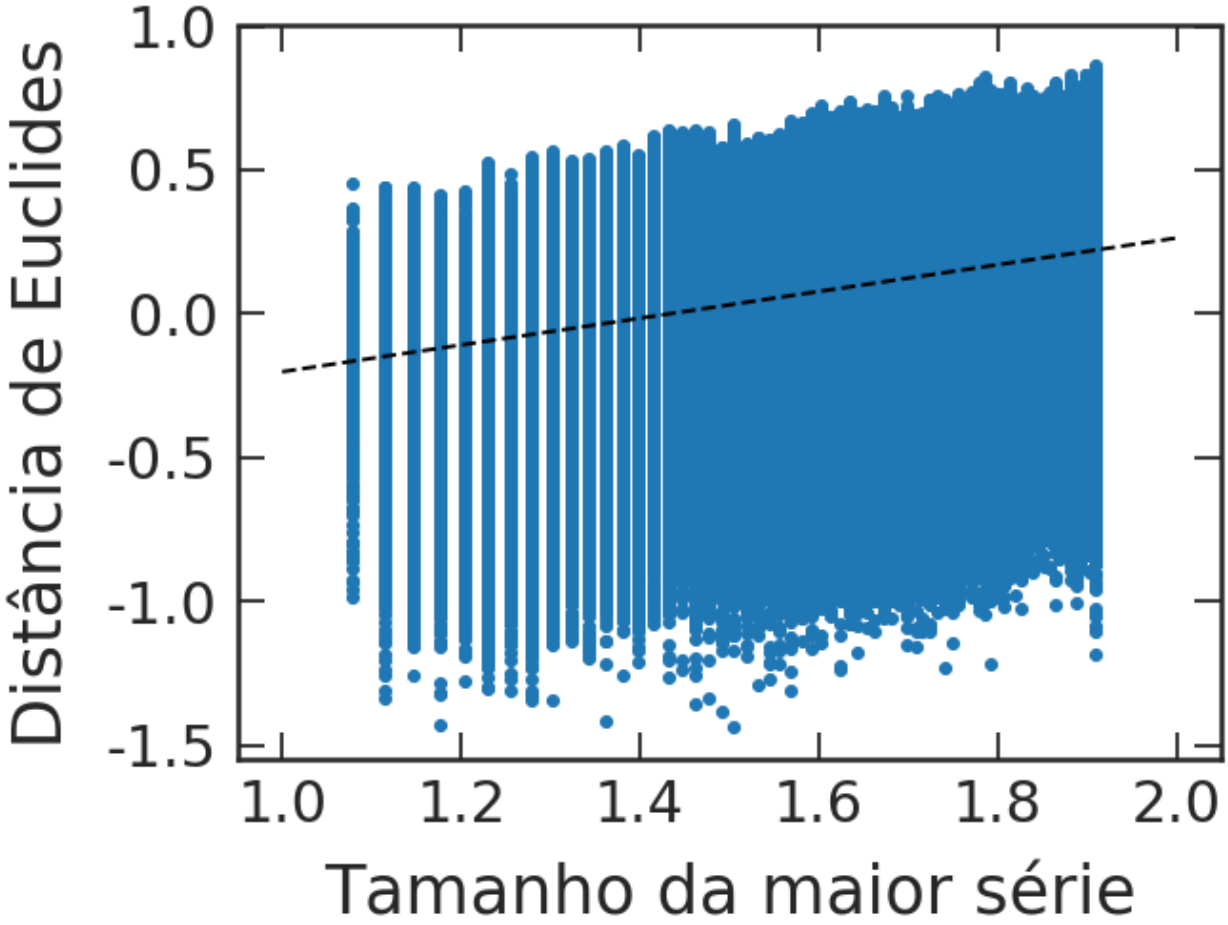

Figura 2.5: Relação entre a distância de duas séries e o tamanho da maior série. Primeiro, calculamos a distância entre todos os pares de séries temporais acumuladas e normalizadas, nesse cálculo utilizamos a definição de distância de Euclides padrão 2.6. A seguir, separamos o tamanho da maior série temporal de cada par de séries. O gráfico foi construído utilizando esses dois valores, no eixo vertical as distâncias e o no eixo horizontal o tamanho das maiores séries de cada par. A linha tracejada representa o ajuste linear $y = 0.46x - 0.67$.

Um comentário deve ser feito, o índice k está relacionado com as observações de cada série, portanto para calcular a distância entre duas séries é necessário que ambas tenham o mesmo tamanho. Porém, já foi dito anteriormente que as séries não têm o mesmo tamanho, elas variam entre 12 à 81 observações. Uma forma de contornar esse problemas é travar a soma até a última observação da menor série, dessa maneira o índice k da somatória varia

até o número de observações da menor série. Limitar o valor de k faz surgir outro problema, parte da maior série não importará no cálculo e isso não parece a melhor forma de comparar duas séries de tamanhos bem diferentes. Fica claro que devemos propor uma correção para a distância definida pela equação 2.6, e essa correção deve estar intimamente relacionada ao tamanho da maior série temporal. Para resolver esse impasse, construímos a figura 2.5. Nela, o eixo das ordenadas representa a distância entre cada par de séries temporais do nosso dado, enquanto que no eixo das abscissas mostramos o tamanho da maior série do par. No gráfico, tomamos o logaritmo na base 10 dos valores nos dois eixos, dessa forma, apesar de o gráfico estar em escala linear, devemos deixar claro que é equivalente a um gráfico dilog. As bolinhas azuis representam os dados empíricos e a linha tracejada representa o ajuste linear de coeficiente angular 0.46, este ajuste sugere a existência de uma lei de potência. Para compreender melhor a lei de potência, vamos definir duas séries temporais, n^u de tamanho t_u e n^u de tamanho t_v. Segundo o gráfico 2.5, temos

$$D(n^u, n^v) \propto (\max\{t_u, t_v\})^{0.46}, \tag{2.7}$$

em que $\max\{t_u, t_v\}$ é o tamanho da maior série. A primeira vista, o valor da potência, 0.46, não parece fazer muito sentido. Porém, pensando na forma matemática da distância de Euclides, a distância crescer como uma raiz quadrada do tamanho da maior série parece razoável. Assim, consideramos $0.46 \approx 0.5$ e, então,

$$D(n^u, n^v) \propto \sqrt{\max\{t_u, t_v\}}, \tag{2.8}$$

ou seja, a distância proposta na equação 2.6 cresce com a raiz quadrada do tamanho da maior série temporal. Desse modo, para tirarmos o viés da distância com o tamanho da série devemos dividir a equação 2.6 pela raiz quadrada do tamanho da maior série temporal. A definição matemática da distância de Euclides sem viés ficou:

$$D(n^u, n^v) = \sqrt{\frac{\sum_k ({n_k}^u - {n_k}^v)^2}{\max\{t_u, t_v\}}}, \tag{2.9}$$

em que o termo $\max\{t_u, t_v\}$ é o tamanho da maior série do par $\{n^u\}$ e $\{n^v\}$. Utilizando a definição de distância definida em 2.9, calculamos a distâncias entre todos os pares de séries temporais acumuladas e normalizadas, e isso rendeu a matriz de distâncias.

2.3 *Hierarchical clustering*

Ao definirmos a matriz de distância, nos dispomos a começar nossa análise de *clustering* por um algoritmo ilustrativo, o chamado *Hierarchical clustering*. Optamos por essa primeira

abordagem por ser um bom método introdutivo, pois ele garante um resultado visual muito interessante e, além disso, ele não é caro computacionalmente falando. O único *imput* necessário para o algoritmo do *Hierarchical clustering* é a matriz de distâncias, baseada nela que será construído toda a estrutura de grupos interna do dado. Inicialmente, o método do *Hierarchical clustering* define cada série como um grupo, a cada passo do algoritmo ele detecta os dois grupos mais similares e os une. Após unir os dois grupos, a matriz de distância é atualizada calculando a distância desse novo grupo em relação a todos os outros grupos. No caso que existam l séries temporais, o algoritmo repetirá os dois passos $l-1$ vezes até formar um único grupo contendo todas as séries. Para mais detalhes técnicos do algorítimo, confira os artigos [96–98].

Em relação a atualização da matriz de distância após juntar dois grupos, devemos fazer um comentário sobre o método utilizado. A literatura oferece muitas opções de métodos para calcular essa distância, porém alguns trabalhos apontam que o método chamado *ward* apresenta os melhores resultados [99]. Para uma contextualização matemática, suponha dois grupos u e v que juntam-se para formar o grupo x. Segundo o método *ward*, a distância do novo grupo x em relação a outro grupo presente no dado, y, será

$$D(x,y) = \sqrt{\frac{|y|+|u|}{T}D(y,u)^2 + \frac{|y|+|v|}{T}D(y,v)^2 - \frac{|y|}{T}D(u,v)^2}, \tag{2.10}$$

em que $T = |y| + |v| + |u|$, e $|*|$ é a cardinalidade do argumento. Devemos deixar claro que $D(y,u)$, $D(y,v)$ e $D(u,v)$ são tirados diretamente da matriz de distâncias. Aplicando o algoritmo *Hierarchical clustering* na matriz de distâncias obtidas na seção anterior, obtemos o dendrograma da figura 2.6. O dendrogama mostra toda a estrutura interna de agrupamento, ele revela com riqueza de detalhes o quão parecidas são as séries. Porém, justamente por ser uma representação detalhada e pelo fato das séries serem muito parecidas, acaba sendo muito difícil detectar visualmente os grupos de séries temporais. Na literatura, já foi discutido que o *Hierarchical clustering* não delimita os grupos, apenas produz uma ilustração da hierarquia [62]. Dessa forma, a literatura sugere alguns critérios estatísticos para determinar uma distância de corte e portanto definir o número de *clusters* [100].

No atual trabalho, escolhemos duas medidas para nos auxiliar a determinar o número de grupos, o primeiro chamado *silhouette coefficient* [101] e o segundo chamado *Calinski Harabasz score* [102]. O *silhouette coefficient* leva em consideração duas medidas, a distância interna média dos elementos de um dado *cluster* e a distância média entre dois *clusters* próximos. De forma simples, o *silhouette coefficient* é calculado fazendo a diferença dessas duas distâncias. Em termos matemáticos, o coeficiente é calculado como,

$$s^j = \frac{b^j - a^j}{\max\{a^j, b^j\}}, \tag{2.11}$$

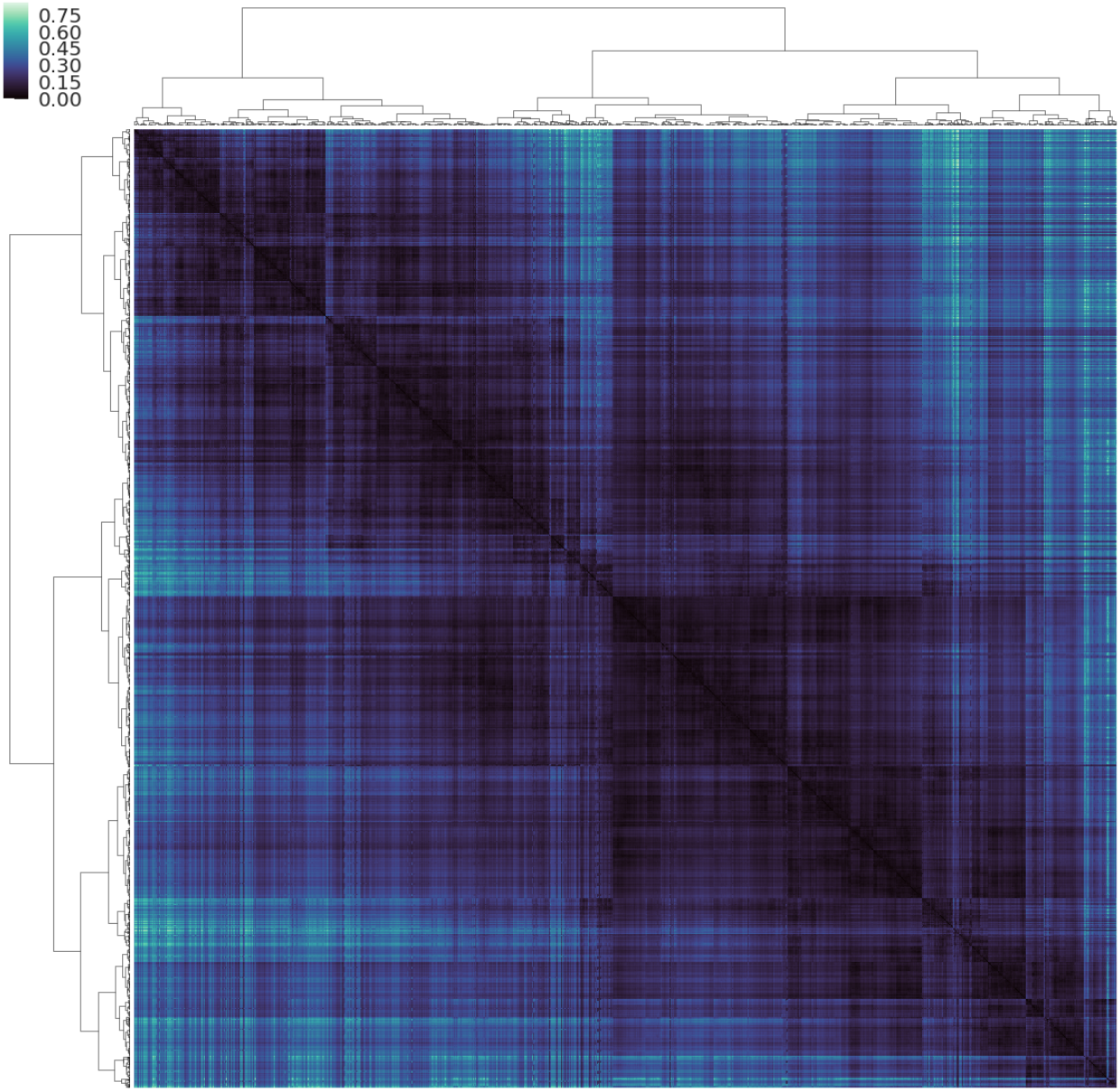

Figura 2.6: Mapa de calor da matriz de distâncias das séries da *Steam*. A borda superior e a borda esquerda mostram o dendrograma obtido via *Hierarchical clustering*, nele está contido toda a estrutura de grupos do dado. No canto superior esquerdo encontra-se a barra de cores correspondente as distâncias.

em que a^j representa a média das distâncias internas entre as séries de um *cluster* e b^j a média das distâncias com o *cluster* mais próximo. É importante deixar claro que para cada série haverá um valor de s^j, após calcular todos os s^j o *silhouette coefficient*, S, será a média, ou seja,

$$S = \langle s^j \rangle. \tag{2.12}$$

A grandeza S pode variar de -1 à 1, quanto mais próximo de 1 melhor é a configuração de *clusters* escolhida, no entanto valores negativos de S indicam que a configuração pode

ter mais ou menos grupos. Explicamos conceitualmente o que a^j e b^j significam, porém não dizemos especificamente como calculamos cada um deles. Tanto a^j como b^j são calculados utilizando a matriz de distâncias atualizada. Suponha uma série, n^j, calculamos a^j tomando a média das distâncias da série n^j em relação as demais séries de seu grupo, já o b^j é a média das distâncias de n^j em relação as séries do grupo mais próximo de n^j. A ideia por trás do *silhouette coefficient* é maximizar a distância fora do grupo e minimizar a distância dentro do grupo, isso pode ser alcançado variando a distância de corte e procurando o máximo de S.

Nessa mesma direção, o *Calinski Harabasz score* também delimita os grupos baseando-se nas distâncias intra e inter *clusters*. Apesar do conceito ser o mesmo, minimizar a distância intra *cluster* e maximizar a inter *cluster*, a formulação matemática é um pouco diferente. O *Calinski Harabasz score* é a taxa de dispersão entre os *clusters* e a dispersão dentro do *cluster*. Se l é o número total de séries temporais do dado e k o suposto número de *clusters*, o *Calinski Harabasz score* é dado por

$$S = \frac{B}{A}\left(\frac{l-k}{k-1}\right). \tag{2.13}$$

As grandezas A e B estão relacionadas às distâncias internas nos grupos e às distâncias externas aos grupos. Assim como no caso anterior, A e B também são obtidas da matriz de distâncias. O fator A é a soma total da soma das distâncias médias entre os elementos de cada grupo, ou melhor,

$$A = \sum_{q=1}^{k} \sum_{x \in C_q} \frac{\sum_{y \in C_q} D(x,y)^2}{N_q - 1}, \tag{2.14}$$

em que C_q é o conjunto de séries do *cluster* q e N_q é o número de séries do conjunto q. Já O fator B diz respeito a soma das distâncias entre os grupos, sua forma matemática é

$$B = \sum_{x \in C} N_x \frac{\sum_{y \in C} D(x,y)^2}{N - 1}, \tag{2.15}$$

em que C é conjunto de todos os *clusters*, N é o número de *clusters* e N_x é o número de séries no *cluster* x. Similar ao *silhouette coefficient*, a ideia do *Calinski Harabasz score* está em variar a distância de corte, limitando o número k de grupos e a partir disso determinar o máximo de S.

Como já dito, empregamos o *silhouette coefficient* e o *Calinski Harabasz score* na hierarquia da figura 2.6. Calculamos os dois coeficientes para diversas distâncias de corte, mais especificamente, tomamos um intervalo de zero à oito. O resultado do *silhouette coefficient* e do *Calinski Harabasz score* para esse intervalo pode ser conferido na figura 2.7. Olhando separadamente para cada gráfico, notamos algumas semelhanças. O gráfico 2.7A, tem um decrescimento inicial, seguido por um crescimento até um máximo local e em seguida ele é

crescente até o fim. Levando em consideração a teoria, o valor máximo do *silhouette coefficient* ocorre para uma distância de corte aproximadamente em sete. Porém essa distância de corte dividiria o dado em dois grupos, veja o dendrograma na escala da distância de corte da figura 2.8 ou 2.9. Isso não seria consistente com aquilo que construímos até aqui neste trabalho, pois no início deste capítulo mostramos alguns pares de séries e lá conseguimos distinguir três grupos de séries temporais (veja figura 2.4). Dessa maneira, levar em consideração o máximo global não seria uma boa escolha. Nesse contexto, algumas medidas são sugeridas na literatura, muitas vezes uma forma de determinar a distância de corte é procurando o primeiro máximo local mais pronunciado. Usando esse critério, marcamos em vermelho na figura 2.7A o primeiro máximo perceptível no dado, sugerindo que a distância ideal de corte é de 1.37. Utilizando essa distância de corte, fizemos uma representação da hierarquia separando cada grupo por uma cor, veja figura 2.8. No dendrograma 2.8, detectamos 11 grupos de séries temporais.

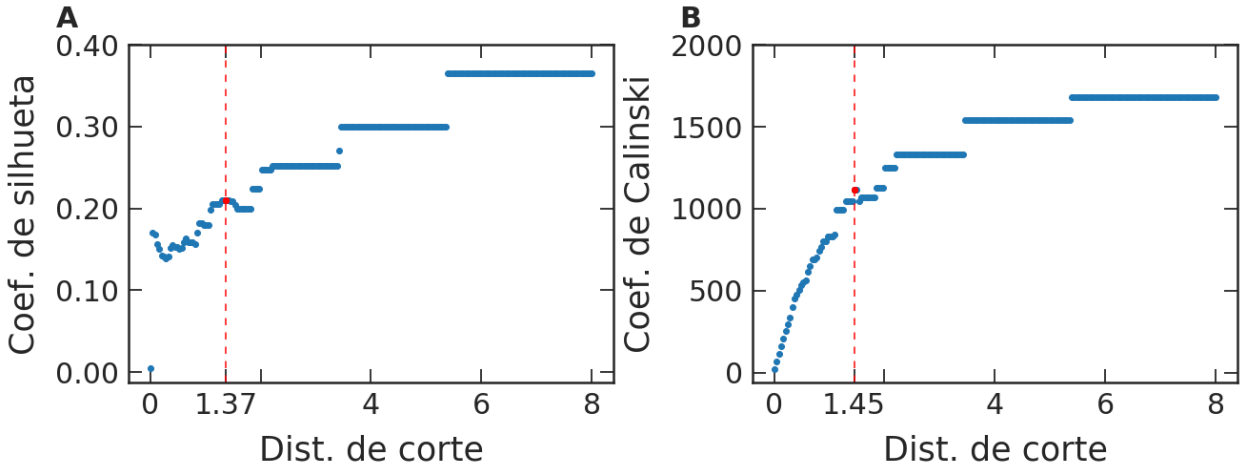

Figura 2.7: Dois coeficientes para detecção do número ótimo de *clusters*. (**A**) Mostra os valores do coeficiente de silhueta em função da distância de corte, marcamos em vermelho o primeiro máximo local pronunciado, na distância de corte de aproximadamente 1.37. (**B**) São os valores do coeficiente de Calinski Harabasz em função da distância de corte, também marcamos em vermelho o primeiro máximo local pronunciado, na distância de corte de aproximadamente 1.45.

Trocando o coeficiente, levando em consideração o *Calinski Harabasz score* representado no gráfico 2.7B, também verificamos o comportamento crescente. O *Calinski Harabasz score* apresenta seu máximo para a distância de corte igual à sete e, além disso, apresenta um máximo local próximo do máximo local da figura 2.7A. A diferença é pequena, mas o máximo local do *Calinski Harabasz score* ocorre na distância de corte igual à 1.45, representado em vermelho na figura 2.7B. Analogamente o que foi feito para o outro coeficiente, a figura 2.9 mostra o dendrograma com o a distância de corte ideal segundo o *Calinski Harabasz score*. Diferentemente da figura 2.8, a 2.9 apresenta 10 grupos de séries temporais. A diferença entre as duas figuras está no grupo representado em laranja na figura 2.9. Esse mesmo

grupo é divido em dois na figura 2.8, representado pelas cores laranja e verde escuro.

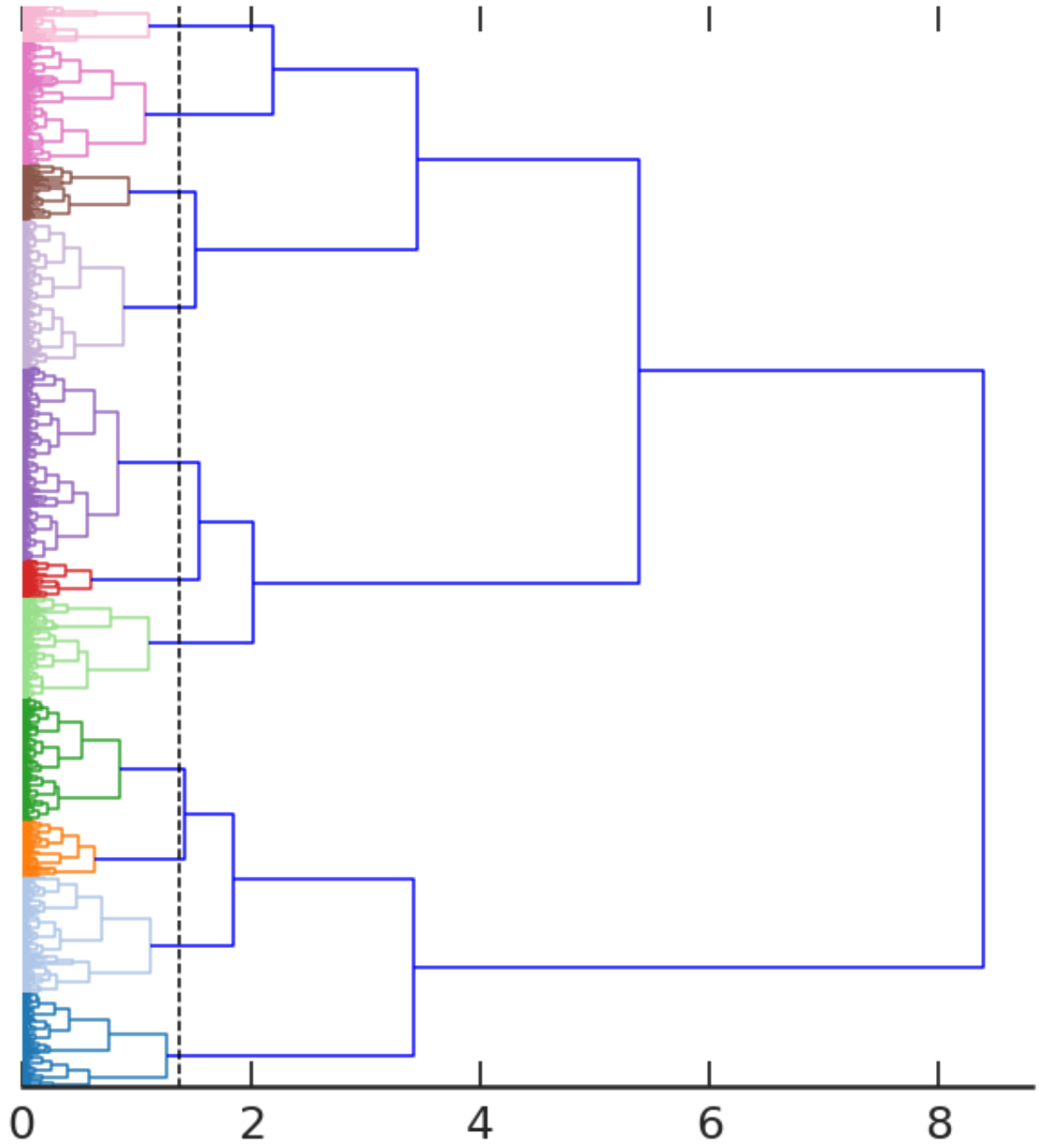

Figura 2.8: Dendrograma dividido pela distância de corte ideal do *silhouette coefficient*. Outra representação do mesmo dendrograma da figura 2.6, porém aqui separamos por cores os 11 grupos sugeridos pelo *silhouette coefficient*.

Ao longo desta seção, algumas dificuldades ficaram claras a respeito da aplicação do *Hierarchical clustering*. Inicialmente, percebemos que as séries são muito similares, apenas construindo o dendrograma não foi possível determinar os grupos de séries temporais. Ao usarmos *silhouette coefficient* e o *Calinski Harabasz score* algumas dificuldades surgiram.

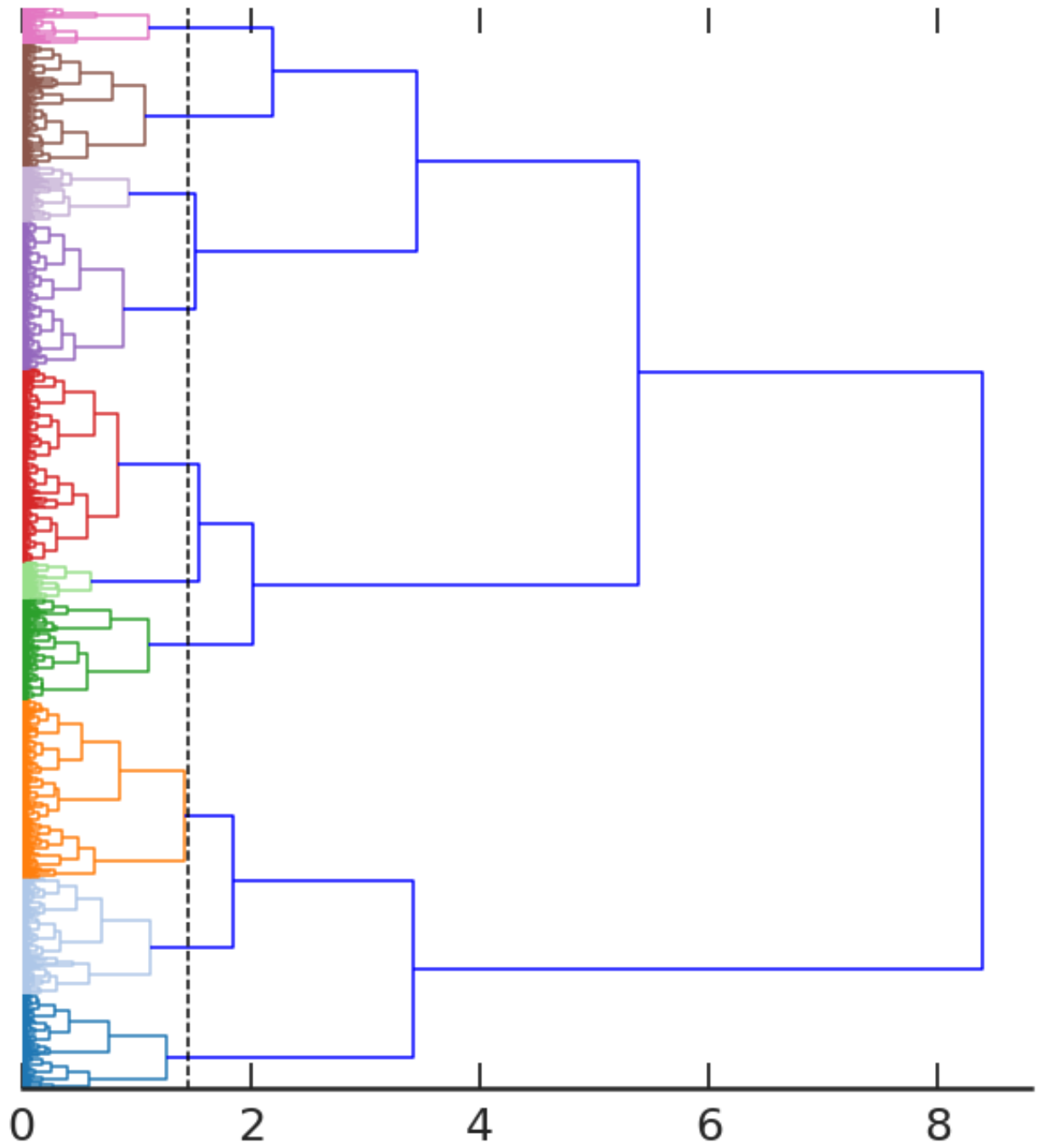

Figura 2.9: Dendrograma dividido pela distância de corte ideal do *Calinski Harabasz score*. A figura mostra o dendrograma cortado pela distância sugerida pelo *Calinski Harabasz score*, aproximadamene 1.45. Nesse corte, aparecem 10 grupos, cada um pintado de uma cor.

Primeiro, o máximo global dos dois coeficientes não são consistentes com achados já mostrados neste trabalho, pois não faz sentido haver apenas dois grupos de séries. Segundo, ao procurarmos por máximos locais, há uma diferença no número de *clusters* detectado para cada coeficiente. Em adição, podemos destacar a subjetividade no critério, primeiro máximo

local pronunciado não é um critério concreto. A falta de significância estatística desses números de *clusters* estimados deixou o problema em aberto. Em meio a esse contexto, outra estratégia foi procurar por um algorítimo mais sensível a detecção de grupos.

2.4 Fundamentação teórica do t-SNE

Outro algoritmo para visualizar grupos e detecta-los é o t-SNE, um procedimento de redução de dimensionalidade. O *t-distributed stochastic neighbor embelding* (t-SNE) foi proposto por Laurens Van der Maaten e Geoffrey Hinton, no artigo *Visualizing High-Dimensional Data Using t-SNE* [103]. Abordagens tradicionais para análises de dados e visualização frequentemente falham em dados com alta dimensão. Assim, procedimentos para fazer a redução de dimensionalidade têm sido cada vez mais implementados, o t-SNE é um algorítimo com essa proposta e tem ganhado popularidade nos últimos anos. Como um método de aprendizado de máquina não supervisionado, o t-SNE é comumente empregado para visualizar dados de alta dimensionalidade e, a partir disso, prover uma intuição da estrutura de grupo presente no dado.

Partindo para um descrição mais matemática, vamos definir como ocorre a redução de dimensionalidade não linear do t-SNE. Seja, $X = x_1, x_2, ..., x_l \subset \mathbb{R}^d$, o conjunto dos dados analisados, em que no nosso caso x_1 representa a primeira série temporal, x_2 representa a segunda série e assim por diante, já d é a dimensão do dado que no nosso caso é igual à 81. A ideia do algorítimo é projetar um conjunto $Y = y_1, y_2, ..., y_l \subset \mathbb{R}^s$ em que $s << d$, a partir desses dois conjuntos, minimizar a distância da probabilidade associada a X, p_{ij}, da probabilidade associada a Y, q_{ij}. Essa é a ideia principal do método, os detalhes matemático serão expostos nos próximos parágrafos.

Inicialmente, o algoritmo projeta o conjunto Y aleatoriamente em um espaço s-dimensional. A dimensão s é escolhida pelo usuário, comumente se escolhe $s = 2$ ou $s = 3$, pois a representação com essas dimensões são muito mais fáceis de serem visualizadas. Definido um espaço s-dimensional, o algorítimo distribui l pontos aleatoriamente nesse espaço. Devemos salientar que escolhendo adequadamente os parâmetros do algoritmo, o resultado final é independente da posição inicial do conjunto Y, dessa maneira o fato dos pontos serem distribuídos aleatoriamente no início não é um problema. Falando nos parâmetros, os parâmetros de *input* do algoritmo são a dimensão do espaço de projeção s, o *perplexity parameter* e o número de passos do algoritmo. O primeiro parâmetro já foi falado. O *perplexity parameter* é a distância de sensibilidade para detecção de *cluster*, valores baixos desse parâmetro resultam na detecção de mais grupos do que realmente existem, enquanto valores altos resultam na detecção de poucos grupos. O terceiro parâmetro é número de passos, esse parâmetro determina a quantidade de vezes que o algoritmo vai iterar. Os dois últimos parâmetros ainda serão mais explorados adiante.

Após definir o espaço de baixa dimensão e posicionar o conjunto Y, duas distribuições de probabilidades são definidas, uma associada a X e outra associada a Y. Vamos denotar a distribuição de probabilidade associada a X por p_{ij}, essa probabilidade está relacionada a distância de x_i e x_j. A forma matemática de p_{ij} é

$$p_{ij} = \frac{\exp\left(-D(x_i, x_j)^2/2\sigma^2\right)}{\sum_{k\neq i} \exp -D(x_i, x_k)^2/2\sigma^2}, \tag{2.16}$$

em que σ é *perplexity parameter* fornecido como *input* do algoritmo. A probabilidade p_{ij} serve como modelo para a probabilidade q_{ij}, pois a ideia do algoritmo é fazer o pontos do espaço s-dimensional ser o mais similar possível com ao pontos do espaço d-dimensional. Nesse contexto, a distribuição de probabilidade relacionada a distância dos pontos de Y é definida como,

$$q_{ij} = \frac{(1 + D(y_i, y_j)^2)^{-1}}{\sum_{k\neq i}(1 + D(y_i, y_j)^2)^{-1}}. \tag{2.17}$$

A distância $D(y_i, y_j)$ entre os dois pontos do espaço s-dimensional, y_i e y_j, é calculada como uma distância de Euclides, veja equação 2.6. Como já foi dito, a probabilidade q_{ij} é sempre comparada à p_{ij}, com intuito de maximizar a similaridade entre o conjunto X e o conjunto Y. Diante disso, o algoritmo do t-SNE calcula a distância entre as distribuições por meio da divergência de Kullback-Leibler, que também pode ser chamado de função custo. Matematicamente, teremos a distância entre p_{ij} e q_{ij} dada por

$$KL(P//Q) = \sum_{i\neq j} p_{ij} \log_{10}\left(\frac{p_{ij}}{q_{ij}}\right). \tag{2.18}$$

A última etapa do algoritmo é a minimização da função custo, essa minimização com respeito ao ponto y_i é feita usando *gradient descent*. Não entraremos em detalhes matemáticos a respeito do método, mas uma descrição teórica completa pode ser conferida nos artigos [104, 105]. A parte importante dessa minimização é entender que função custo acaba funcionando como mola, pois ela empurra e puxa os pontos na representação de baixa dimensionalidade. O resultado é um mapa que reflete a similaridade entre os *inputs* de alta dimensionalidade.

Após entender os passos mais fundamentais do algoritmo t-SNE, devemos fazer alguns comentários a respeito. O t-SNE é capaz de capturar muito da estrutura local do dado de alta dimensão, enquanto revela também a estrutura global, tal como a presença de *clusters* em muitas escalas. Porém, a distância entre os *clusters* não são conservadas, ou seja, no espaço de baixa dimensionalidade os grupos mais próximos não necessariamente são os mais similares. Nessa mesma questão, nem as densidades são conservadas. O comentário final é acerca dos ruídos, o algoritmo é sensível a ruídos, pois nem sempre os ruídos são representados como aleatórios. A armadilha clássica é achar que existe um padrão, mas no fundo é aleatório, uma tática para detectar ruídos é variar o *perplexity parameter*.

2.4.1 Exemplo sintético

Antes de abordar os dados da *Steam*, aplicamos o algoritmo t-SNE em um dado sintético. Geramos o dado utilizando a função *make_ blobs* do *Python*, o dado sintético corresponde a 3000 pontos distribuídos em cinco *clusters* de mesmas densidades. Para facilitar a apresentação o dado foi gerado com duas dimensões, veja a figura 2.10. Os cinco grupos são muitos perceptíveis visualmente, dessa maneira, deixamos claro de que se trata de uma aplicação simples para entender o funcionamento do algoritmo.

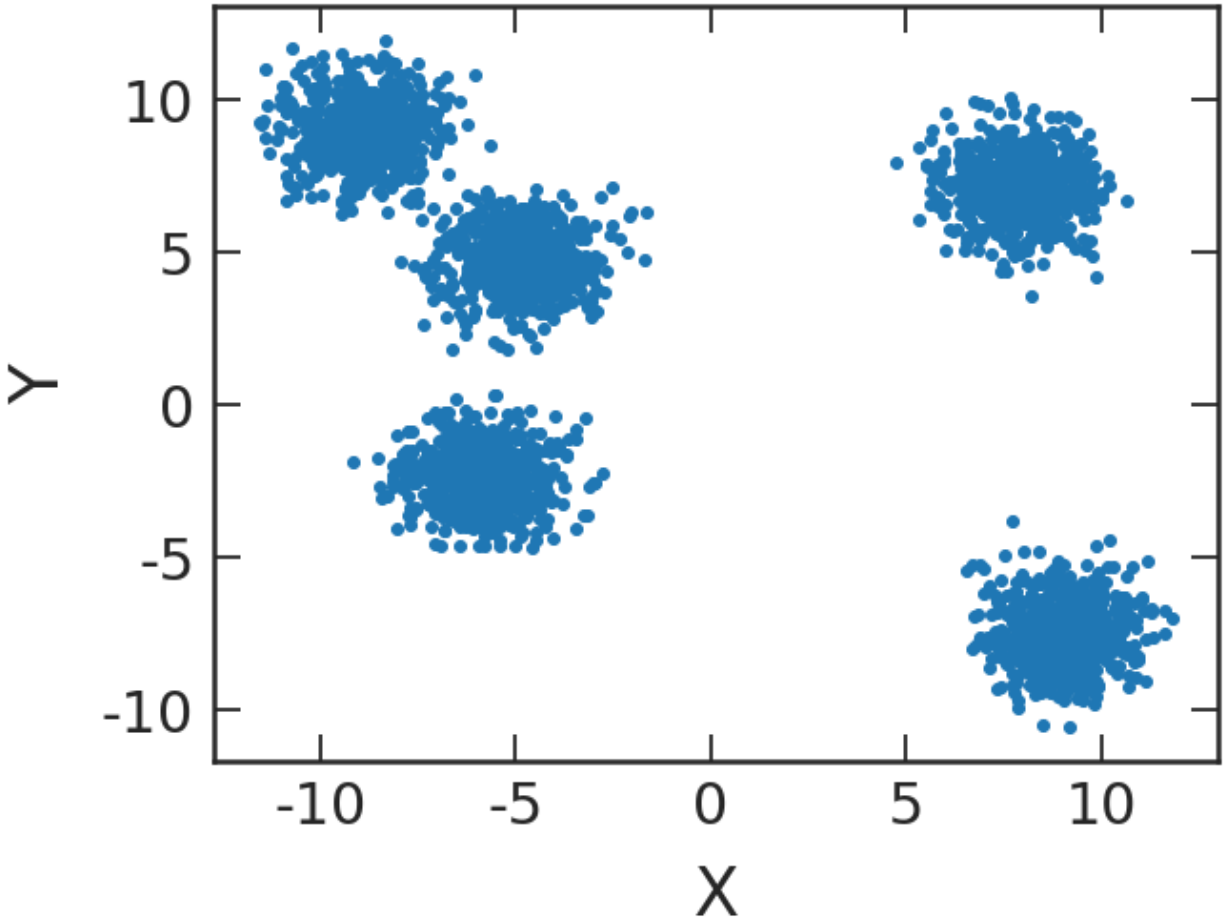

Figura 2.10: Dados gerados sinteticamente no plano. A imagem mostra o conjunto de três mil pontos, divididos em cinco grupos de mesma densidade. O dado foi gerado pela função *make_ blobs* do *Python*.

O primeiro passo foi determinar a matriz de distâncias do dado, a distância empregada foi a de Euclides, definida na equação 2.6. Aproveitando a matriz de distâncias e lembrando que o *hierarchical clustering* depende apenas dela, representamos o dendrograma do dado sintético na figura 2.11. O mapa de calor da hierarquia deixa os cinco grupos bem pronunciados, apenas visualmente já é possível detecta-los. Nesse caso, nem se faz necessário um critério de corte.

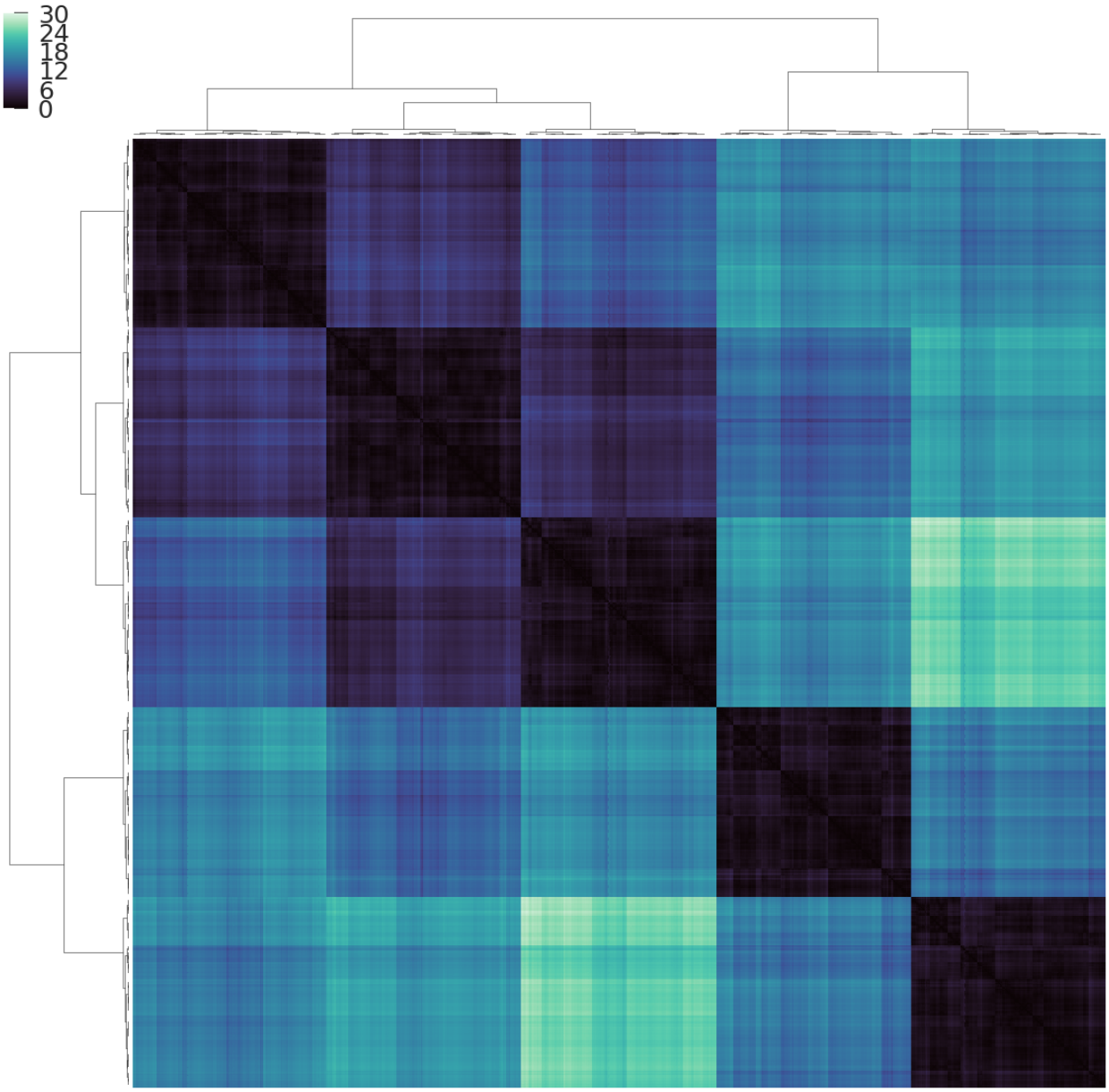

Figura 2.11: Mapa de calor da matriz de distâncias do dado sintético. No canto superior esquerdo mostramos a barra de cor para cada distância, a parte de cima e da esquerda mostram o dendrograma feito a partir dos dados sintéticos. O mapa de calor mostra visivelmente cinco grupos bem definidos.

Retomando à aplicação do t-SNE, a matriz de distância é necessária para calcular as distâncias da equação 2.16. Junto com a matriz de distâncias definimos os parâmetros de *input*, adotamos a dimensionalidade de $s = 2$, *perplexity parameter* igual à 400 e 3000 passos de iteração do algoritmo. Geramos um dado bidimensional e escolhemos $s = 2$ para facilitar a representação, assim todos podem visualizar o *input* e o *output* do algoritmo. O resultado da projeção no plano t-SNE está exposto na figura 2.12A, nele podemos perceber que o algoritmo detectou os grupos e deixou a distância entre grupos ainda mais espaçada, compare o espaçamento entre os grupos da figura 2.10 e 2.12A.

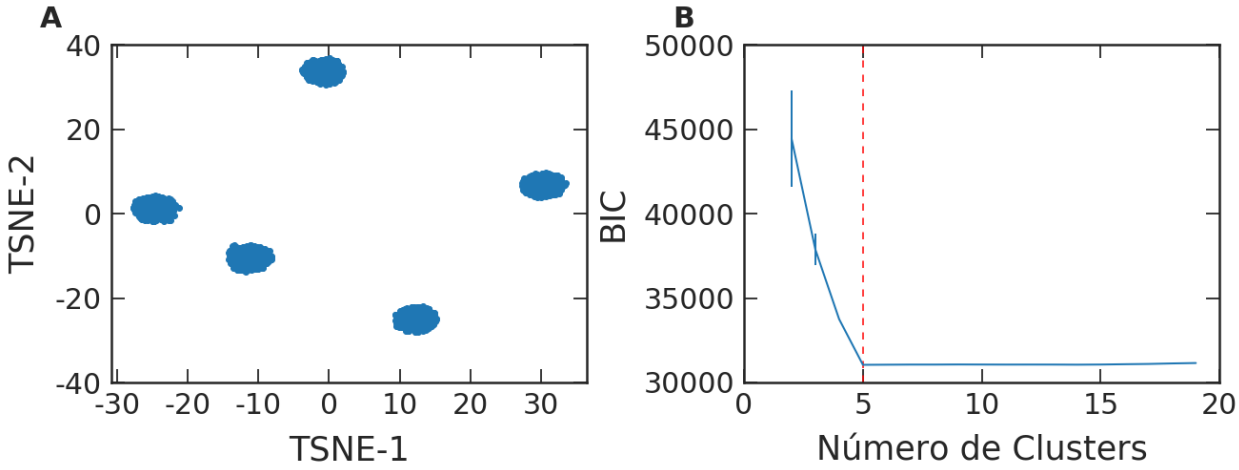

Figura 2.12: Resultados da analise t-SNE para os dados sintéticos. (**A**) Projeção da matriz de distâncias do dado sintético no plano t-SNE. Na projeção, usamos os seguintes parâmetros: $s = 2$, *perplexity parameter* igual à 400 e 3000 passos. O algoritmo conseguiu definir ainda melhor os 5 grupos, os deixando ainda mais espaçados e densos. (**B**) Valores do BIC para um conjunto de número de *clusters*. Em vermelho marcamos o valor mínimo do BIC que confirma os cinco *clusters*.

Apesar da projeção ter sido visivelmente bem sucedida, remontando os cinco grupos, ainda escolhemos um critério para determinar estatisticamente o número de grupos. O critério escolhido foi o *Bayesian information criterion* (BIC), este critério dá uma estimativa do quão bom é o *Gaussian Mixture Model* (GMM) em termos dos dados que temos. Qualitativamente, o GMM ajusta um número de distribuições gaussianas para o dado, o número de gaussianas é determinado como *input* para o algoritmo. A partir das distribuições de probabilidades ajustadas o BIC estima a qualidade geral dos ajustes, porém devemos deixar claro que quanto mais baixo for o valor do BIC melhor será o modelo para explicação dos dados. Os detalhes acerca do GMM e do BIC podem ser conferidos nas seguintes referências [106–108]. Retomando a discussão da projeção t-SNE dos dados sintéticos, calculamos o BIC para o conjunto de três à 20 *clusters*, o resultado está exposto na figura 2.12B. Apenas confirmando o que já era esperado, o BIC apresenta seu mínimo em cinco grupos onde está marcado em vermelho. Portanto, a aplicação do algoritmo t-SNE foi bem sucedida para o dado sintético, ele detectou os cinco grupos gerados.

2.5 Aplicação do t-SNE nos dados da *Steam*

Nesta seção nos dedicamos em apresentar os detalhes da aplicação dos t-SNE nos dados da *Steam*. Como *input* para o algoritmo usamos a matriz de distância calculada na seção 2.2, a dimensão de projeção $s = 3$, *perplexity parameter* igual à 50 e 5000 passos. Optamos por projetar o espaço t-SNE em um espaço tridimensional, pois projeções em espaços de muitas

dimensões fogem da intuição humana básica, além disso, acreditamos que três dimensões é uma representação que possibilita uma boa visualização. A respeitos do *perplexity parameter*, testamos um grande conjunto de valores, porém o valor de 50 alcançou os melhores resultados em termos visuais. Comparativamente ao exemplo sintético, aqui usamos um *perplexity parameter* bem inferior, isso acaba sendo um pouco óbvio, pois a séries da *Steam* são muitos mais similares entre si do que os dados sintéticos, compare as figuras 2.6 e 2.11. Apenas lembrando que *perplexity parameter* é interpretado como a distância de sensibilidade para detecção dos *clusters*. Ainda comparando com o exemplo sintético, usamos um número maior de passos para o algoritmo porque ele depende de como é o dado de entrada. Lá no exemplo sintético, o dado era muito mais simples, então um número pequeno de passos já seria suficiente. No entanto, os dados da *Steam* são muito mais complexos, e por isso, escolhemos 5000 passos por ser uma quantidade segura para o algoritmo convergir.

Empregando os parâmetros discutidos, a figura 2.13 mostra uma representação da projeção dos dados da *Steam* no espaço t-SNE. Dessa vez, os grupos não ficaram bem pronunciados, muito pelo contrário apenas apareceram algumas manchas. Já era esperado que visualmente o resultado não ficasse ótimo, isso se deve a maior complexidade dos dados. Todavia, é inegável que algumas regiões dos gráficos são mais densas do que outras, isso ainda garante que existem grupos de séries que são mais similares que outras. O objetivo agora é determinar esses grupos.

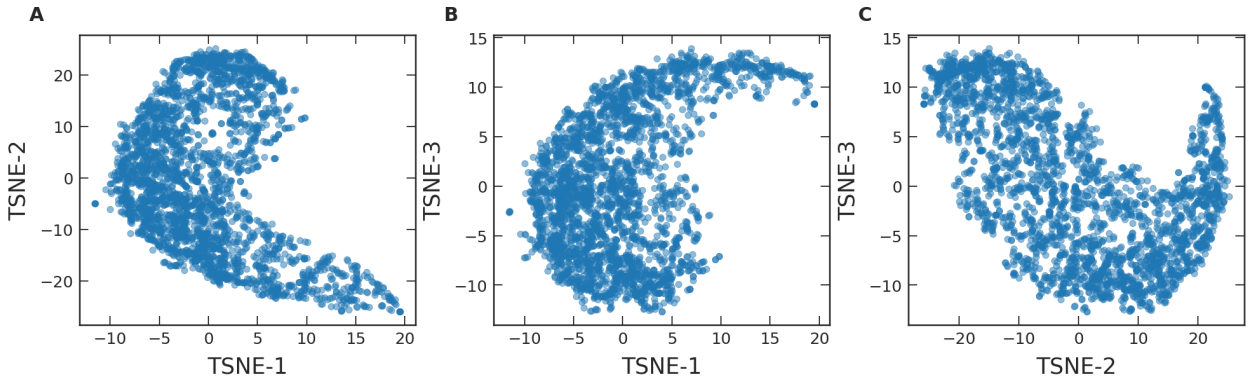

Figura 2.13: Projeção da matriz de distâncias do dado da *Steam* no espaço t-SNE. Em cada gráfico projetamos um par dos eixos do espaço t-SNE. (**A**) Projeção dos eixos um e dois. (**B**) Projeção dos eixos um e três. (**C**) E projeção dos eixos dois e três. Esse resultado foi obtido usando $s = 3$, *perplexity parameter* igual à 30 e 5000 passos de iteração.

Seguiremos o mesmo protocolo da análise dos dados sintéticos, utilizamos o *Bayesian Information Criterion* (BIC) como critério para determinar o número de *clusters*. Testamos o BIC para um conjunto de três a 40 grupos, o resultado está exposto no gráfico 2.14A. Observando o gráfico, percebemos que o valor do BIC cai muito rapidamente até oito *clusters*. De oito até 16 *clusters*, ele tem uma queda menos inclinada e depois disso o valor do BIC

sobre bem vagarosamente. A linha azul do gráfico 2.14A representa a estimativa do BIC, já as barras em torno da linha dão a dimensão da incerteza da estimativa. Levando em consideração a incerteza, podemos afirmar que os valores do BIC entre 16 e aproximadamente 25 *clusters* são estatisticamente iguais. Em meio a esse impasse, optamos por fazer um teste de significância estatística usando o p-valor. O teste com o p-valor para o mesmo conjunto de *clusters* testado para o BIC pode ser conferido no gráfico 2.14B. O pico de significância do p-valor ocorreu em 16 grupos, marcamos em vermelho nos dois gráficos da figura 2.14. O fato do mínimo do BIC ter ocorrido no mesmo lugar do pico de significância do p-valor trouxe uma certa robusteza no resultado da estimativa de 16 grupos para o dado da *Steam*.

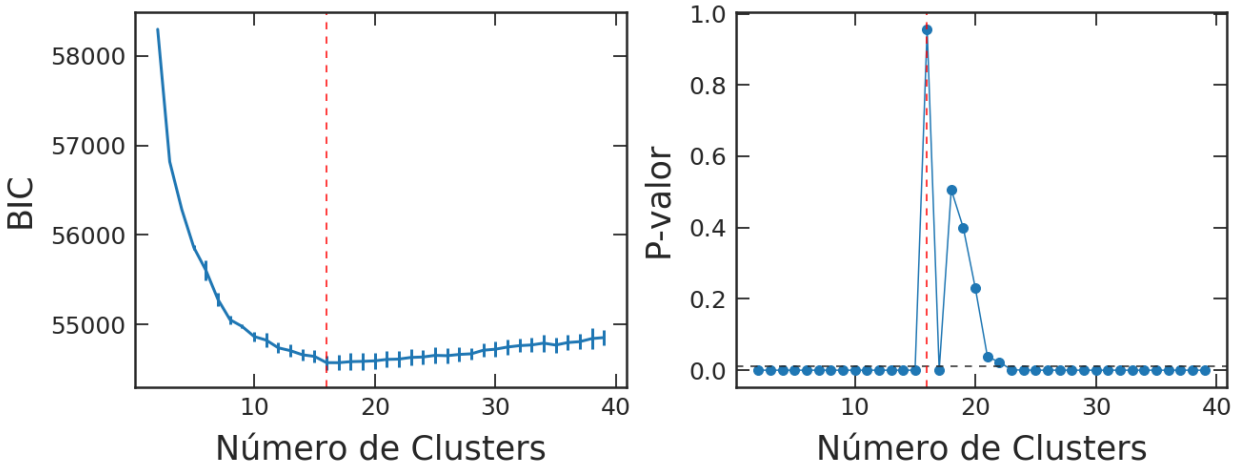

Figura 2.14: Critério para determinar o número de *clusters* ótimo. (**A**) Valores do BIC para um conjunto de número de *clusters*, o valor mínimo do coeficiente foi marcado em vermelho. (**B**) Teste de significância do p-valor para o número de *clusters*, o primeiro pico significante foi marcado em vermelho. Ambos os testes detectaram 16 *clusters*.

Após os dois critérios estatísticos definirem os 16 grupos, o passo seguinte é visualizar os grupos no espaço t-SNE. Nesse contexto, colorimos a projeção no espaço t-SNE de tal forma que cada grupo fosse correspondido por uma cor, veja figura 2.15. Apesar de muitos grupos terem sido detectados, percebemos que as cores parecem bem agrupadas, isto é, quase não observamos pontos de uma cor rodeados por outra cor. Além disso, a projeção colorida nos dá a ideia de quais grupos são mais parecidos. Por exemplo, baseado no gráfico 2.15A, o grupo roxo é bem mais parecido com o grupo alaranjado do que com o grupo rosa. Isso desperta nossa intuição de que existem grupos com a dinâmica de popularidade bem característica e outros grupos de transição, ou seja, alguns grupos podem ter um comportamento intermediário entre dois ou mais grupos.

Terminamos o parágrafo anterior falando sobre possíveis grupos intermediários, e isso levanta uma questão: Será que o algoritmo superestimou o número de grupos? Como já reportado na literatura, se os grupos são de tamanhos diferentes, o modelo tende a superestimar o ajuste para explicar os pequenos grupos, o que resulta em um grande número de

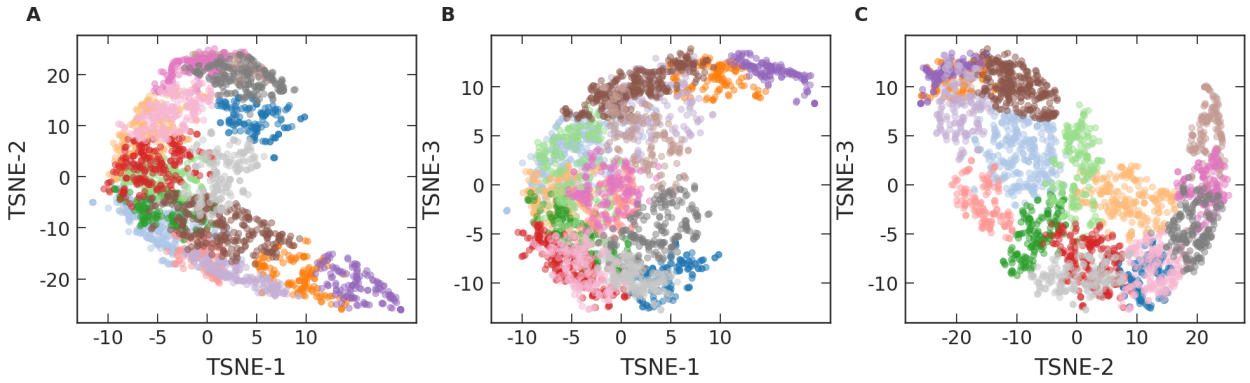

Figura 2.15: Projeção das séries no espaço t-SNE coloridos por grupos. Assim como na figura 2.13, cada gráfico mostra um par dos eixos do espaço t-SNE.(**A**) Projeção dos eixos um e dois. (**B**) Projeção dos eixos um e três. (**C**) E projeção dos eixos dois e três.

clusters espúrios [109]. Diante dessa questão, aplicamos um procedimento para detecção de *clusters* espúrios. O procedimento é baseado na comparação de duas densidades de pontos, a primeira referente aos pontos em torno do centro dos grupos e a segunda referente aos pontos de um modelo nulo. De forma mais detalhada, seja $Y = \{y_{(1)}, y_{(2)}, y_{(3)}, ..., y_{(l)}\}$ o conjunto dos pontos projetado no espaço t-SNE, em que $y_{(1)} = \{y_{1,1}, y_{1,2}, y_{1,3}\}$, isto é, cada elemento do conjunto $y_{(i)}$ representa uma coordenada do i-ésimo ponto no espaço t-SNE. Considerando o conjunto de pontos Y, calculamos a densidade (ρ) no centro de cada grupo, g_j para $j = 1, 2, ..., 16$, como sendo

$$\rho(g_j) = \frac{1}{l} \sum_{i=1}^{l} K_h(g_j, y_i), \tag{2.19}$$

dado que o fator $K_h(g_j, y_i)$ denota um *kernel* com largura de banda h. No nosso caso, empregamos o *kernel* gaussiano, dado por:

$$K_h(g_j, y_i) = \frac{1}{(2\pi h^2)^{3/2}} \exp\left(-\frac{\sum_{d=1}^{3} (y_{i,d} - g_{i,d})^2}{2h^2} \right), \tag{2.20}$$

em que a largura de banda h é calculada da média da distância de Euclides (equação 2.6) dos vizinhos mais próximos de g_j.

Comparamos a densidade ρ obtida dos dados com uma densidade estimada do modelo nulo $\tilde{\rho}$. O modelo nulo foi construído embaralhando os pontos por componentes, ou seja, fixamos uma componente do espaço t-SNE e embaralhamos os pontos, isso feito para todas as três componentes. A partir dos pontos embaralhados, calculamos a densidade embaralhada $\tilde{\rho}$ usando as equações 2.19 e 2.20. Repetimos o processo de embaralhamento inúmeras vezes, isso nos possibilitou estimar uma distribuição de probabilidade $p(\tilde{\rho})$. Da distribuição

de probabilidade, calculamos duas grandezas, o p-valor para $p(\rho < \tilde{\rho})$ e o enriquecimento dado por $\rho/\langle\tilde{\rho}\rangle$. A figura 2.16, mostra a relação do logaritmo do p-valor pelo logaritmo do enriquecimento, ambos os logaritmos foram calculados na base 10. As linhas tracejadas da figura marcam os limites de p-valor e de enriquecimento aceitáveis, ambos os limites foram tomados baseados no trabalho [110]. Segundo o método de detecção de *clusters* espúrios, são descartados os *clusters* correspondentes aos pontos que estão a esquerda da linha vertical e/ou acima da linha horizontal. Levando isso em consideração, descartamos a possibilidade da detecção de *clusters* espúrios, logo, julgamos todos os 16 grupos como legítimos.

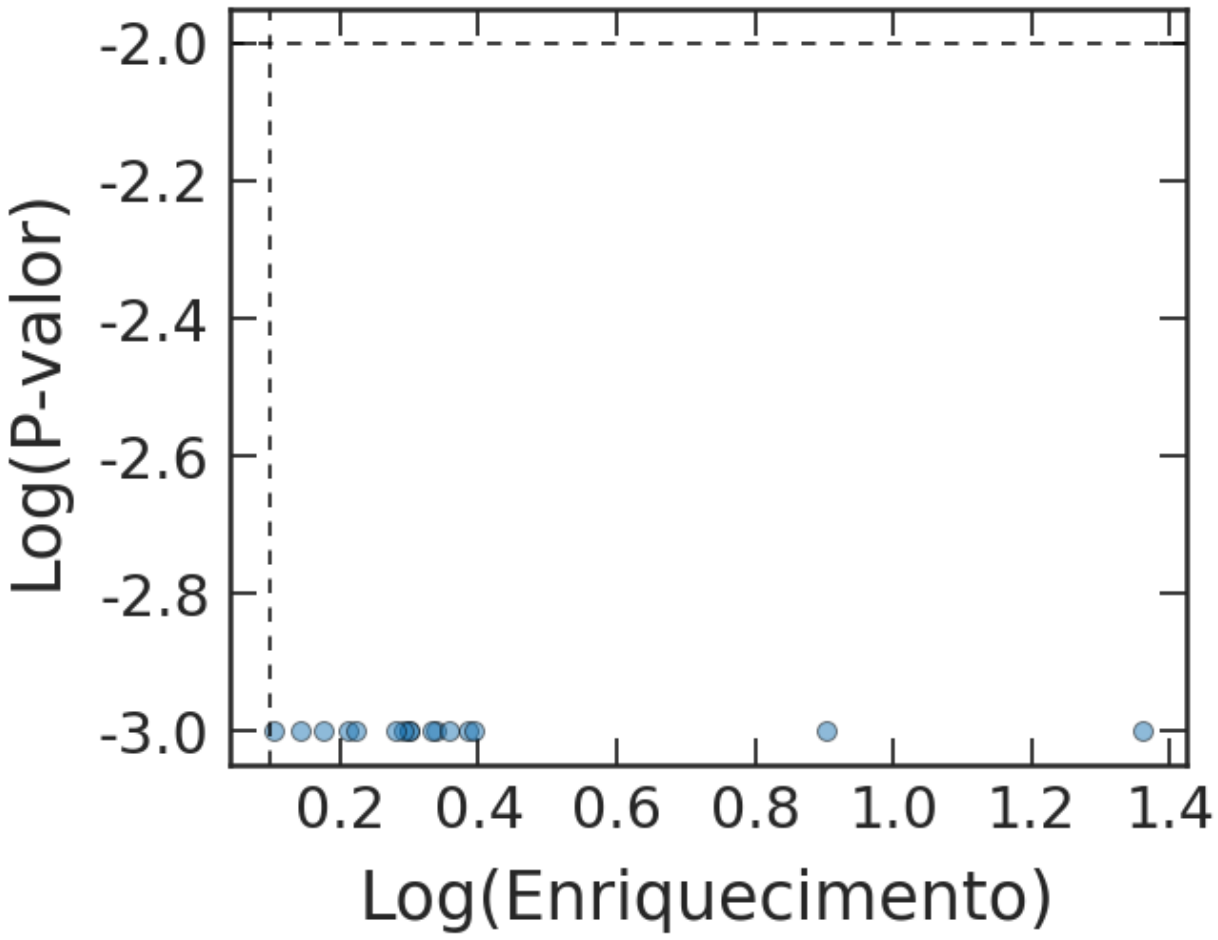

Figura 2.16: Análise de detecção de *clusters* espúrios. As bolinhas em azuis representam os dados empíricos, as linhas tracejadas demarcam os limites do enriquecimento e do p-valor para detecção dos *clusters* espúrios. O gráfico mostra que não foi detectado nenhum *cluster* espúrio, isto é, os 16 grupos detectados são legítimos.

A etapa final da análise de *clustering* é a visualização dos grupos de séries temporais. A figura 2.17 mostra os 16 grupos de séries temporais acumuladas e normalizadas, o esquema de cores de cada grupo foi mantido o mesmo da figura 2.15. A primeira vista, o que podemos perceber é que alguns grupos flutuam mais do que outros em relação a média, porém a grande maioria dos grupos apresentam séries bem alinhadas com as séries do grupo. Por

exemplo, o grupo laranja claro (primeira linha, quarta coluna), o vermelho (segunda linha, terceira coluna), marrom escuro (terceira linha, terceira coluna) e rosa escuro (quarta linha, primeira coluna) parecem estar bem uniformes. Diferente delas, alguns grupos parecem ter uma flutuação maior em relação a média, é o caso dos grupos de cor alaranjada (primeira linha, terceira coluna) e arroxeada (terceira linha, primeiro coluna).

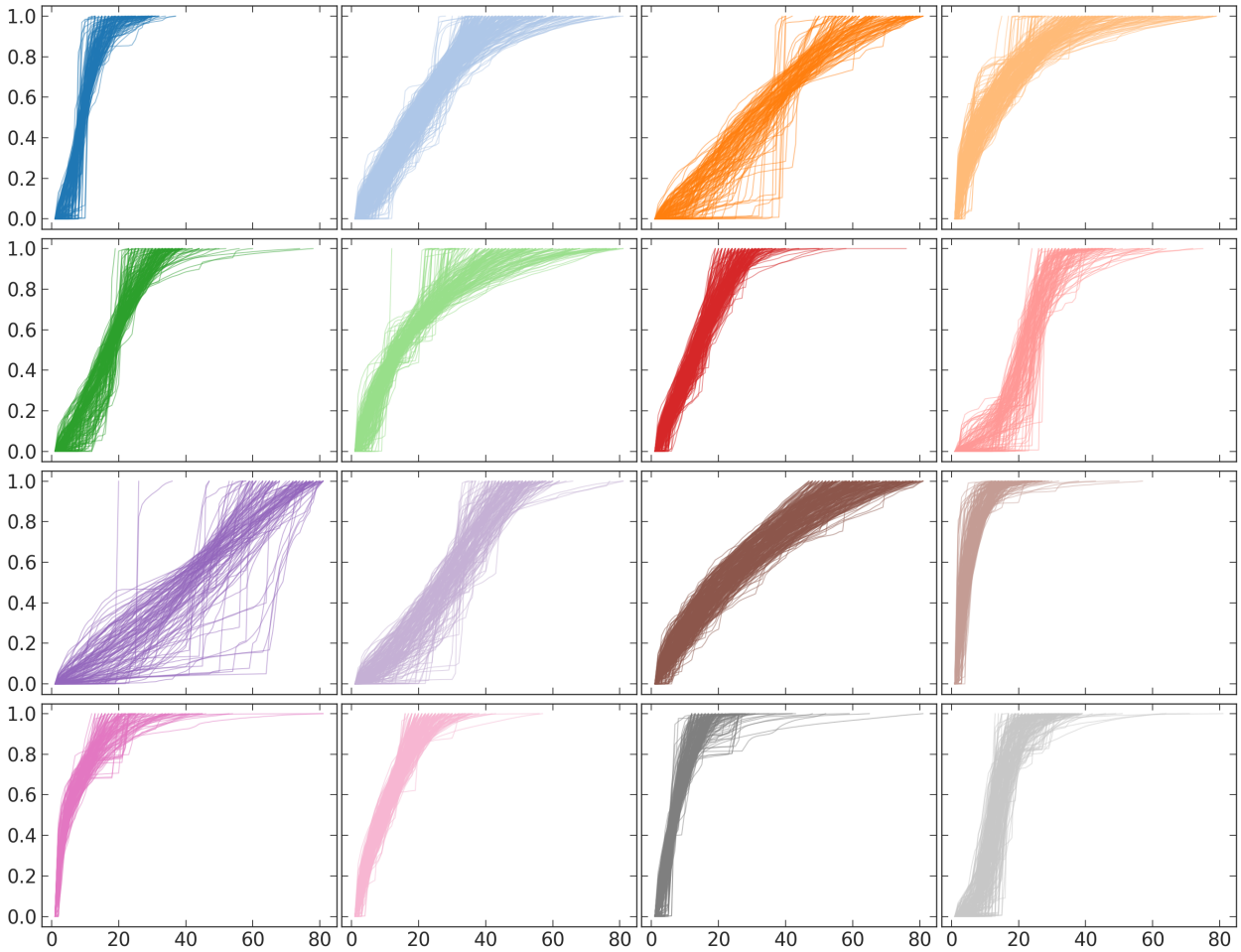

Figura 2.17: Apresentação dos 16 grupos de séries temporais acumuladas e normalizadas. Os gráficos apresentam o crescimento de popularidade de cada grupo de séries temporais acumuladas e normalizadas, as cores de cada grupo correspondem as cores do gráfico 2.15.

A visualização das séries acumulas dá uma boa noção sobre o crescimento da popularidade do grupos, mas o grande objetivo deste trabalho é determinar os padrões de dinâmica das séries. Já foi levantado a questão da flutuação no parágrafo anterior, mesmo utilizando as séries acumuladas alguns grupos apresentaram bastante flutuação em relação a média. Dessa maneira, construímos a figura 2.18 tomando a média mês a mês das séries do grupo, a linha escura de cada grupo representa a média, já a barra clara em torno da linha é um desvio padrão para cima da média e um para baixo. Observando em detalhes cada grupo de séries, podemos perceber que alguns se parecem bastante, na verdade alguns grupos se comportam

como uma espécie de filhos de outros grupos. Por exemplo, veja o grupo rosa (quarta linha, primeira coluna) ele decai muito rapidamente em seu começo e depois se estabiliza em valores bem baixos, mas além dele, o grupo alaranjado claro (primeira linha, quarta coluna) tem o mesmo padrão só que com séries mais longas, alcançando até 60 meses. Adicionalmente, o grupo marrom escuro (terceira linha, terceira coluna) segue o mesmo decaimento, porém suas séries chegam a aproximadamente 80 meses. Outra particularidade interessante, é que alguns grupos parecem versões transladadas de outros grupos. Veja o grupo azul (primeira linha, primeira coluna), ele tem um formato de sino, e observando o grupo cinza claro (quarta linha, quarta coluna), notamos o mesmo comportamento, porém o pico está ligeiramente deslocado para a direita. Nessa mesma linha, o grupo de cor salmão (segunda linha, quarta coluna) apresenta o mesmo padrão de sino com pico ainda mais deslocado para a direita.

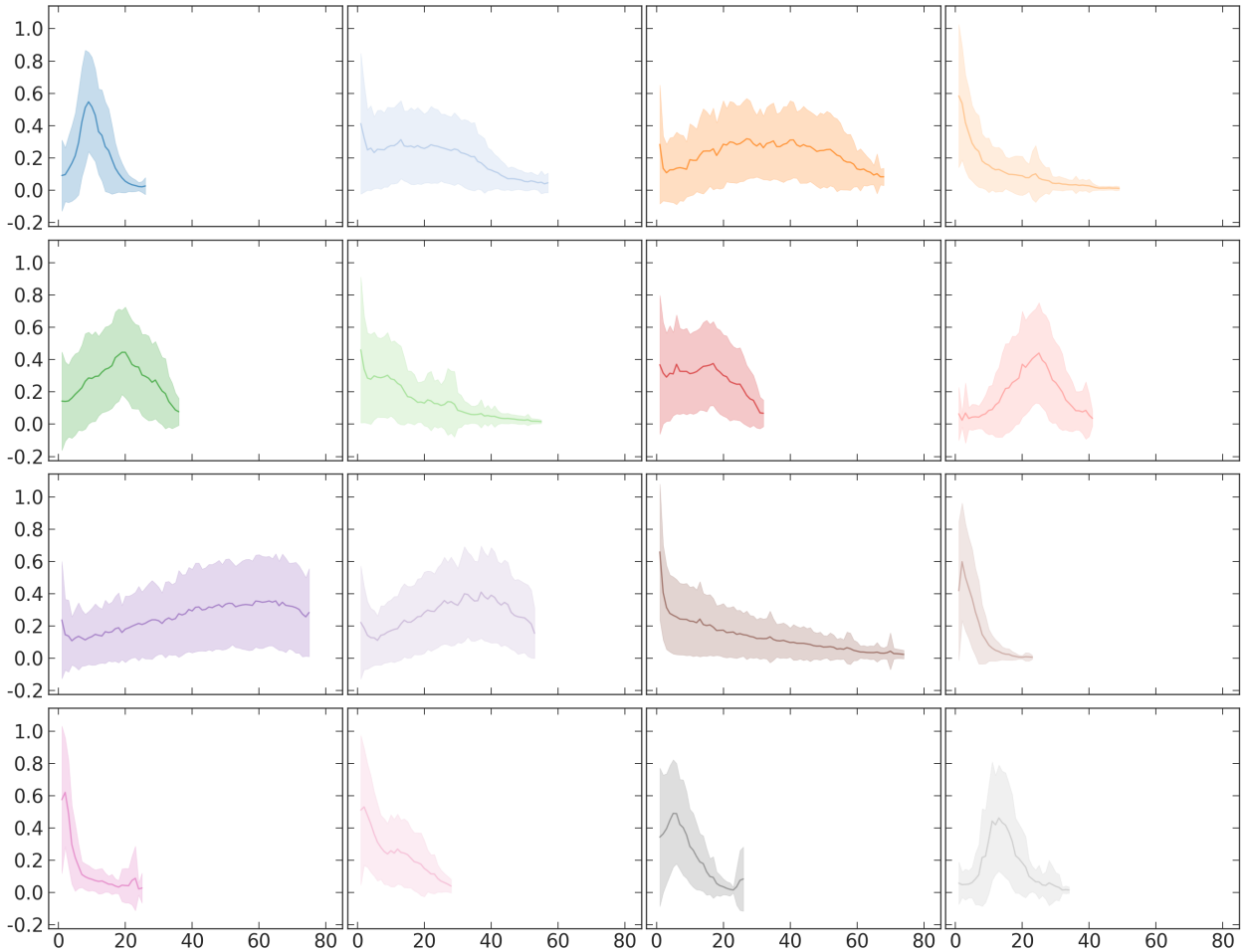

Figura 2.18: Apresentação dos 16 grupos de séries temporais normalizadas. A linha escura de cada gráfico é a média de todas as séries normalizadas do grupo, já a barra mais clara representa um desvio padrão em torno da média.

O fato de alguns grupos se parecerem filhos ou versões transladadas de outros grupos, nos encoraja a agrupar qualitativamente grupos de séries temporais. Alguns trabalhos já separam alguns grupos de séries temporais em grupos que apresentam o padrão de *burst* e os

que não apresentam. Como definido na referência [111] a dinâmica de *burst* é quando ocorre um pico bem abrupto nas séries, ou pode ser apenas um decaimento bem acelerado. Nesse trabalho citado, os autores analisaram a popularidade de repositórios do GitHub e de *hashtags* do *Twitter*. Diante dos resultados encontrados, os autores decidiram dividir os grupos de séries temporais em dois grandes padrões, os que apresentam *burst* e os que não apresentam. Entretanto, prestando atenção na figura 2.18, os tipos de comportamentos encontrados nos dados da *Steam* apresentam uma diversidade maior. Portanto, consideramos existir três grandes tipos de comportamentos distintos na dinâmica de popularidade mensal dos jogos da *Steam*. O primeiro, é o comportamento de decaimento, apresentado pelos oito grupos seguintes: azul claro (primeira linha, segunda coluna), alaranjado claro (primeira linha, quarta coluna), verde claro (segunda linha, segunda coluna), vermelho (segunda linha, terceira coluna), marrom (terceira linha, terceira coluna), marrom claro (terceira linha, quarta coluna), rosa (quarta linha, primeira coluna) e rosa claro (quarta linha, segunda coluna). O segundo, é o comportamento de sino, dos próximos cinco grupos: azul (primeira linha, primeira coluna), verde (segunda linha, primeira coluna), salmão (segunda linha, quarta coluna), cinza escuro (quarta linha, terceira coluna) e cinza claro (quarta linha, quarta coluna). E terceiro, os três grupos que apresenta um comportamento de crescimento e decrescimento suave: alaranjado (primeira linha, terceira coluna), roxo (terceira linha, primeira coluna) e roxo claro (terceira linha, segunda coluna).

Acreditamos que nesse trabalho alcançamos nosso objetivo estudar alguns aspectos de jogos, com ênfase em detectar os tipos de dinâmicas temporais da popularidade dos jogos. Nossa análise principal detectou 16 grupos de séries temporais no dado da *Steam* e, posteriormente, argumentamos que estes 16 grupos podem ser reduzidos a três grandes padrões de comportamento. Não sabemos se esses padrões permanecerão ao longo do tempo, talvez propostas de trabalhos futuros possam verificar esse aspecto. Além disso, não sabemos se esses padrões se repetem em outras plataformas de jogos, nesse caso trabalhos poderão ser propostos para analisar outras bases similares a *Steam*. Por último, devemos deixar claro que nosso trabalho teve seu grande foco na investigação dos tipos de dinâmica de popularidade de jogos *online*. Apesar de usarmos a temática dos jogos, nossa pesquisa pode ser vista como mais abrangente do que isso e, portanto, resultados da nossa pesquisa podem ser úteis ou até mesmo aplicáveis em outros contextos.

CAPÍTULO 3

Considerações finais

Nesse trabalho, analisamos dados de popularidade dos jogos *online* com ênfase no uso aprendizado de maquina não supervisionado. Mais especificamente, investigamos as séries temporais sob a perspectiva de análise de *clustering*. O dado empregado nessa investigação foi obtido de duas bases de dados, o site oficial da *Steam* e o site que disponibiliza dados oficiais da *Steam*, chamado *Steam Charts*. Desses dois bancos de dados, tiramos informações de 18211 jogos. Essas informações correspondem a: nome, data de lançamento, gêneros, desenvolvedora, publicadora e série temporal da popularidade mensal. Extraímos e tratamos todo o dado utilizando pacotes e programas escritos em *Python*, as análises e as figuras também foram feitas nesse ambiente de programação.

Em um primeiro momento, trouxemos as informações demográficas mais relvantes do dado. Sem abarcar as séries temporais em si, extraímos umas primeiras informações dos dados. Mostramos que o número de jogos lançados cresce exponencialmente com o tempo, enquanto que o número de jogadores cresce linearmente. A respeito da grande quantidade dos jogos lançados nos últimos anos, mostramos que grande parte não alcançam um grande público. Quando somamos a popularidade de cada jogo ao longo de todo o tempo, a distribuição de probabilidade dessa soma é bem ajustada por uma lognormal. Comparando os jogos mais recentes com os jogos velhos, em média os jogos mais velhos alcançam uma popularidade maior. Separando os jogos por gênero, mostramos que cada gênero lança os jogos a uma taxa exponencial ao longo do tempo, ou seja, além de global o padrão exponencial é verificado por gênero também. Levando em consideração as desenvolvedoras e publicadoras, obtivemos aproximadamente um padrão lei de potência no gráfico de *ranking* do número de jogos lançados por elas.

O segundo capítulo abordou o tema central do trabalho, a análise de agrupamento das

séries temporais. Desde o começo, deixamos claro o nosso objetivo em determinar os padrões de dinâmica da popularidade dos jogos, visando atingir o objetivo fizemos um procedimento preparatório. Primeiro, dos dados dos 18211 jogos obtidos utilizamos apenas 2972, pois descartamos os dados que não satisfaziam alguns critérios. Dos dados úteis para nossa análise, acumulamos as séries temporais para diminuir as flutuações, depois normalizamos as séries para todas serem comparadas dentro do mesmo intervalo de magnitude. Após prepararmos as séries, determinamos uma medida de distância baseada na de Euclides que pudesse ser aplicada a séries de tamanhos diferentes. Com as séries preparadas e a distância definida, partimos para uma primeira tentativa de análise usando *hierarchical clustering*. A primeira tentativa não mostrou resultados consistentes estatisticamente. Partimos para a segunda tentativa, utilizamos o algoritmo de redução de dimensionalidade chamado t-SNE, combinado com ele aplicamos o algoritmo de detecção de grupos *Gaussian Mixture Model*. A segunda tentativa passou por teste estatísticos e seu resultado foi consistente, detectamos 16 grupos de séries temporais. Examinando os grupos, argumentamos qualitativamente que existem três grandes padrões de dinâmica de popularidade nos jogos da *Steam*.

Finalmente, pensando em perspectivas futuras, podemos tentar verificar esses padrões de dinâmica de popularidade em outros contextos. Como citado em várias partes do trabalho, muitos artigos científicos estão sendo propostos nessa linha de estudos de dinâmica de popularidade. Em uma tentativa mais ambiciosa, trabalhos futuros podem propor formas de unificar todos esses padrões de dinâmica. Em uma vertente de aplicação, trabalhos focados em investigar a popularidade podem ser úteis para compreender a dinâmica da mudança de atenção das pessoas. Esses estudos têm uma imensa aplicabilidade na industria do *marketing* e na disseminação de ideias de forma geral.

Referências Bibliográficas

[1] Kwapień, J. & Drożdż, S. Physical approach to complex systems. *Physics Reports* **515**, 115–226 (2012).

[2] Mitchell, M. Complex systems: Network thinking. *Artificial Intelligence* **170**, 1194–1212 (2006).

[3] Cameron, L. & Larsen-Freeman, D. Complex systems and applied linguistics. *International Journal of Applied Linguistics* **17**, 226–239 (2007).

[4] Boccara, N. *Modeling complex systems* (Springer Science & Business Media, 2010).

[5] Kobayashi, N., Kuninaka, H., Wakita, J.-i. & Matsushita, M. Statistical features of complex systems–toward establishing sociological physics–. *Journal of the Physical Society of Japan* **80**, 072001 (2011).

[6] Arthur, W. B. Complexity and the economy. *Science* **284**, 107–109 (1999).

[7] Lillo, F. & Mantegna, R. N. Power-law relaxation in a complex system: Omori law after a financial market crash. *Physical Review E* **68**, 016119 (2003).

[8] Grimm, V., Revilla, E., Berger, U., Jeltsch, F., Mooij, W. M., Railsback, S. F., Thulke, H.-H., Weiner, J., Wiegand, T. & DeAngelis, D. L. Pattern-oriented modeling of agent-based complex systems: lessons from ecology. *Science* **310**, 987–991 (2005).

[9] Guimera, R. & Amaral, L. A. N. Functional cartography of complex metabolic networks. *Nature* **433**, 895 (2005).

[10] Hamilton, M. J., Milne, B. T., Walker, R. S., Burger, O. & Brown, J. H. The complex structure of hunter–gatherer social networks. *Proceedings of the Royal Society B: Biological Sciences* **274**, 2195–2203 (2007).

[11] Mason, O. & Verwoerd, M. Graph theory and networks in biology. *IET Systems Biology* **1**, 89–119 (2007).

[12] May, R. M., Levin, S. A. & Sugihara, G. Complex systems: Ecology for bankers. *Nature* **451**, 893 (2008).

[13] Picoli Jr, S. & Mendes, R. Universal features in the growth dynamics of religious activities. *Physical Review E* **77**, 036105 (2008).

[14] Donges, J. F., Zou, Y., Marwan, N. & Kurths, J. Complex networks in climate dynamics. *The European Physical Journal Special Topics* **174**, 157–179 (2009).

[15] Galea, S., Hall, C. & Kaplan, G. A. Social epidemiology and complex system dynamic modelling as applied to health behaviour and drug use research. *International Journal of Drug Policy* **20**, 209–216 (2009).

[16] Tabak, B., Serra, T. & Cajueiro, D. Topological properties of commodities networks. *The European Physical Journal B* **74**, 243–249 (2010).

[17] Garrett, K. A. *et al.* Complexity in climate-change impacts: An analytical framework for effects mediated by plant disease. *Plant Pathology* **60**, 15–30 (2011).

[18] Xiong, F., Liu, Y., Zhang, Z.-j., Zhu, J. & Zhang, Y. An information diffusion model based on retweeting mechanism for online social media. *Physics Letters A* **376**, 2103–2108 (2012).

[19] Cervantes-De la Torre, F., González-Trejo, J. I., Real-Ramirez, C. A. & Hoyos-Reyes, L. F. Fractal dimension algorithms and their application to time series associated with natural phenomena. In *Journal of Physics: Conference Series*, vol. 475, 012002 (IOP Publishing, 2013).

[20] Bittencourt, N., Meeuwisse, W., Mendonça, L., Nettel-Aguirre, A., Ocarino, J. & Fonseca, S. Complex systems approach for sports injuries: Moving from risk factor identification to injury pattern recognition—narrative review and new concept. *British Journal of Sports Medicine* **50**, 1309–1314 (2016).

[21] Brailsford, S., Lattimer, V., Tarnaras, P. & Turnbull, J. Emergency and on-demand healthcare: Modeling a large complex system. In *Operational Research for Emergency Planning in Healthcare: Volume 2*, 13–30 (Springer, 2016).

[22] Fry, J. & Cheah, E.-T. Negative bubbles and shocks in cryptocurrency markets. *International Review of Financial Analysis* **47**, 343–352 (2016).

[23] Carlson, J. M., Langer, J. S. & Shaw, B. E. Dynamics of earthquake faults. *Reviews of Modern Physics* **66**, 657 (1994).

[24] Shaw, B. E. Generalized omori law for aftershocks and foreshocks from a simple dynamics. *Geophysical Research Letters* **20**, 907–910 (1993).

[25] Bak, P., Christensen, K., Danon, L. & Scanlon, T. Unified scaling law for earthquakes. *Physical Review Letters* **88**, 178501 (2002).

[26] Marsan, D., Bean, C. J., Steacy, S. & McCloskey, J. Observation of diffusion processes in earthquake populations and implications for the predictability of seismicity systems. *Journal of Geophysical Research: Solid Earth* **105**, 28081–28094 (2000).

[27] Christian, R. R. & Luczkovich, J. J. Organizing and understanding a winter's seagrass foodweb network through effective trophic levels. *Ecological Modelling* **117**, 99–124 (1999).

[28] Bascompte, J. & Melián, C. J. Simple trophic modules for complex food webs. *Ecology* **86**, 2868–2873 (2005).

[29] Woodward, G., Papantoniou, G., Edwards, F. & Lauridsen, R. B. Trophic trickles and cascades in a complex food web: Impacts of a keystone predator on stream community structure and ecosystem processes. *Oikos* **117**, 683–692 (2008).

[30] Matia, K., Ashkenazy, Y. & Stanley, H. E. Multifractal properties of price fluctuations of stocks and commodities. *EPL (Europhysics Letters)* **61**, 422 (2003).

[31] Farmer, J. D., Gallegati, M., Hommes, C., Kirman, A., Ormerod, P., Cincotti, S., Sanchez, A. & Helbing, D. A complex systems approach to constructing better models for managing financial markets and the economy. *The European Physical Journal Special Topics* **214**, 295–324 (2012).

[32] Bariviera, A. F., Zunino, L. & Rosso, O. A. An analysis of high-frequency cryptocurrencies prices dynamics using permutation-information-theory quantifiers. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28**, 075511 (2018).

[33] Sigaki, H. Y., Perc, M. & Ribeiro, H. V. Clustering patterns in efficiency and the coming-of-age of the cryptocurrency market. *Scientific Reports* **9**, 1440 (2019).

[34] Guimera, R., Danon, L., Diaz-Guilera, A., Giralt, F. & Arenas, A. Self-similar community structure in a network of human interactions. *Physical Review E* **68**, 065103 (2003).

[35] Ding, K., Jiang, P., Leng, J. & Cao, W. Modeling and analyzing of an enterprise relationship network in the context of social manufacturing. *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* **230**, 752–769 (2016).

[36] Cai, M., Wang, W., Cui, Y. & Stanley, H. E. Multiplex network analysis of employee performance and employee social relationships. *Physica A: Statistical Mechanics and its Applications* **490**, 1–12 (2018).

[37] Alves, L. G., Ribeiro, H. V., Lenzi, E. K. & Mendes, R. S. Distance to the scaling law: A useful approach for unveiling relationships between crime and urban metrics. *Plos one* **8**, e69580 (2013).

[38] Alves, L. G., Ribeiro, H. V. & Rodrigues, F. A. Crime prediction through urban metrics and statistical learning. *Physica A: Statistical Mechanics and its Applications* **505**, 435–443 (2018).

[39] Bakker, E. M. Open and free datasets for multimedia retrieval. *International Journal of Multimedia Information Retrieval* **5**, 135–136 (2016).

[40] Gandomi, A. & Haider, M. Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management* **35**, 137–144 (2015).

[41] Rossum, G. Python reference manual (1995).

[42] Pedregosa, F. *et al.* Scikit-learn: Machine learning in python. *Journal of Machine Learning Research* **12**, 2825–2830 (2011).

[43] McKinney, W. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython* ("O'Reilly Media, Inc.", 2012).

[44] Gastner, M. T. & Newman, M. E. Optimal design of spatial distribution networks. *Physical Review E* **74**, 016117 (2006).

[45] Gastner, M. T. & Newman, M. E. Shape and efficiency in spatial distribution networks. *Journal of Statistical Mechanics: Theory and Experiment* **2006**, P01015 (2006).

[46] Brockwell, P. J. & Davis, R. A. *Introduction to time series and forecasting* (springer, 2016).

[47] Mendes, R., Malacarne, L. & Anteneodo, C. Statistics of football dynamics. *The European Physical Journal B* **57**, 357–363 (2007).

[48] Lacasa, L., Luque, B., Ballesteros, F., Luque, J. & Nuno, J. C. From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences* **105**, 4972–4975 (2008).

[49] Marwan, N., Donges, J. F., Zou, Y., Donner, R. V. & Kurths, J. Complex network approach for recurrence analysis of time series. *Physics Letters A* **373**, 4246–4254 (2009).

[50] Gao, Z.-K., Small, M. & Kurths, J. Complex network analysis of time series. *EPL (Europhysics Letters)* **116**, 50001 (2017).

[51] Pessa, A. A. & Ribeiro, H. V. Characterizing stochastic time series with ordinal networks. *Physical Review E* **100**, 042304 (2019).

[52] Theiler, J. Spurious dimension from correlation algorithms applied to limited time-series data. *Physical Review A* **34**, 2427 (1986).

[53] Bence, J. R. Analysis of short time series: Correcting for autocorrelation. *Ecology* **76**, 628–639 (1995).

[54] Ramoni, M. F., Sebastiani, P. & Kohane, I. S. Cluster analysis of gene expression dynamics. *Proceedings of the National Academy of Sciences* **99**, 9121–9126 (2002).

[55] Siqueira Jr, E. L., Stošić, T., Bejan, L. & Stošić, B. Correlations and cross-correlations in the brazilian agrarian commodities and stocks. *Physica A: Statistical Mechanics and its Applications* **389**, 2739–2743 (2010).

[56] Rasmussen, C. E. Gaussian processes in machine learning. In *Summer School on Machine Learning*, 63–71 (Springer, 2003).

[57] Huang, A. Similarity measures for text document clustering. In *Proceedings of the sixth New Zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, vol. 4, 9–56 (2008).

[58] Aghabozorgi, S., Shirkhorshidi, A. S. & Wah, T. Y. Time-series clustering–a decade review. *Information Systems* **53**, 16–38 (2015).

[59] Fu, T.-c. A review on time series data mining. *Engineering Applications of Artificial Intelligence* **24**, 164–181 (2011).

[60] Love, B. C. Comparing supervised and unsupervised category learning. *Psychonomic bulletin & review* **9**, 829–835 (2002).

[61] Schonlau, M. The clustergram: a graph for visualizing hierarchical and nonhierarchical cluster analyses. *The Stata Journal* **2**, 391–402 (2002).

[62] Milligan, G. W. & Cooper, M. C. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate Behavioral Research* **21**, 441–458 (1986).

[63] Ahmad, A. & Dey, L. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering* **63**, 503–527 (2007).

[64] Martínez-Álvarez, F., Troncoso, A., Riquelme, J. & Riquelme, J. Partitioning-clustering techniques applied to the electricity price time series. In *International Conference on Intelligent Data Engineering and Automated Learning*, 990–999 (Springer, 2007).

[65] Yeung, K. Y., Fraley, C., Murua, A., Raftery, A. E. & Ruzzo, W. L. Model-based clustering and data transformations for gene expression data. *Bioinformatics* **17**, 977–987 (2001).

[66] Fraley, C. & Raftery, A. E. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association* **97**, 611–631 (2002).

[67] Kriegel, H.-P., Kröger, P., Sander, J. & Zimek, A. Density-based clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **1**, 231–240 (2011).

[68] Lai, C.-P., Chung, P.-C. & Tseng, V. S. A novel two-level clustering method for time series data analysis. *Expert Systems with Applications* **37**, 6319–6326 (2010).

[69] Zhang, X., Liu, J., Du, Y. & Lv, T. A novel clustering method on time series data. *Expert Systems with Applications* **38**, 11891–11900 (2011).

[70] Molteni, L. & Ordanini, A. Consumption patterns, digital technology and music downloading. *Long Range Planning* **36**, 389–406 (2003).

[71] Rosa, H. *Social acceleration: A new theory of modernity* (Columbia University Press, 2013).

[72] Salganik, M. *Bit by bit: Social research in the digital age* (Princeton University Press, 2019).

[73] Crane, R. & Sornette, D. Robust dynamic classes revealed by measuring the response function of a social system. *Proceedings of the National Academy of Sciences* **105**, 15649–15653 (2008).

[74] Eom, Y.-H. & Fortunato, S. Characterizing and modeling citation dynamics. *PloS ONE* **6**, e24926 (2011).

[75] Lorenz-Spreen, P., Mønsted, B. M., Hövel, P. & Lehmann, S. Accelerating dynamics of collective attention. *Nature Communications* **10**, 1759 (2019).

[76] Wu, F. & Huberman, B. A. Novelty and collective attention. *Proceedings of the National Academy of Sciences* **104**, 17599–17601 (2007).

[77] Candia, C., Jara-Figueroa, C., Rodriguez-Sickert, C., Barabási, A.-L. & Hidalgo, C. A. The universal decay of collective memory and attention. *Nature human behaviour* **3**, 82 (2019).

[78] Connolly, T. M., Boyle, E. A., MacArthur, E., Hainey, T. & Boyle, J. M. A systematic literature review of empirical evidence on computer games and serious games. *Computers & education* **59**, 661–686 (2012).

[79] Eglesz*, D., Fekete, I., Kiss, O. E. & Izso, L. Computer games are fun? on professional games and players' motivations. *Educational Media International* **42**, 117–124 (2005).

[80] Corporation, V. Steam (2020). URL https://store.steampowered.com/. Acessado em 22 de fevereiro de 2020.

[81] Gray, J. Steam charts (2019). URL https://steamcharts.com. Acessado em 22 de fevereiro de 2020.

[82] Corporation, V. Steam (2019). URL https://store.steampowered.com/stats. Acessado em 22 de fevereiro de 2020.

[83] Arts, E. Origin (2020). URL https://www.origin.com/bra/pt-br/store. Acessado em 22 de fevereiro de 2020.

[84] Ubisoft. Uplay (2020). URL https://uplay.ubisoft.com/. Acessado em 22 de fevereiro de 2020.

[85] Projekt, C. Gog.com (2020). URL https://www.gog.com/. Acessado em 22 de fevereiro de 2020.

[86] Entertainment, B. Battle.net (2020). URL https://us.shop.battle.net/en-us. Acessado em 22 de fevereiro de 2020.

[87] Corcoran, L. itch.io (2020). URL https://itch.io/. Acessado em 22 de fevereiro de 2020.

[88] Corporation, M. Microsoft store (2020). URL https://www.microsoft.com/pt-br/store/b/home. Acessado em 22 de fevereiro de 2020.

[89] Corporation, P. Playerunknown's battlegrounds (2020). URL https://www.pubg.com/. Acessado em 22 de fevereiro de 2020.

[90] Anderton, K. The business of video games: A multi billion dollar industry [infographic] (2017). URL https://www.forbes.com/sites/kevinanderton/2017/04/29/the-business-of-video-games-a-multi-billion-dollar-industry-infograph#4ac634386d27. Acessado em 10 de fevereiro de 2019.

[91] Gotelli, N. J. *Ecologia* (Ed. Planta, 2007).

[92] Crow, E. L. & Shimizu, K. *Lognormal distributions* (Marcel Dekker New York, 1987).

[93] Adamic, L. A. Zipf, power-laws, and pareto-a ranking tutorial. *Xerox Palo Alto Research Center, Palo Alto, CA, http://ginger. hpl. hp. com/shl/papers/ranking/ranking. html* (2000).

[94] Furusawa, C. & Kaneko, K. Zipf's law in gene expression. *Physical Review Letters* **90**, 088102 (2003).

[95] Ferrer i Cancho, R. & Solé, R. V. Two regimes in the frequency of words and the origins of complex lexicons: Zipf's law revisited. *Journal of Quantitative Linguistics* **8**, 165–173 (2001).

[96] Eisen, M. B., Spellman, P. T., Brown, P. O. & Botstein, D. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences* **95**, 14863–14868 (1998).

[97] Bar-Joseph, Z., Gifford, D. K. & Jaakkola, T. S. Fast optimal leaf ordering for hierarchical clustering. *Bioinformatics* **17**, S22–S29 (2001).

[98] Müllner, D. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378* (2011).

[99] Milligan, G. W. & Cooper, M. C. Methodology review: Clustering methods. *Applied Psychological Measurement* **11**, 329–354 (1987).

[100] Kingrani, S. K., Levene, M. & Zhang, D. Estimating the number of clusters using diversity. *Artificial Intelligence Research* **7**, 15–22 (2018).

[101] Rousseeuw, P. J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics* **20**, 53–65 (1987).

[102] Caliński, T. & Harabasz, J. A dendrite method for cluster analysis. *Communications in Statistics-theory and Methods* **3**, 1–27 (1974).

[103] Maaten, L. v. d. & Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research* **9**, 2579–2605 (2008).

[104] Mason, L., Baxter, J., Bartlett, P. L. & Frean, M. R. Boosting algorithms as gradient descent. In *Advances in Neural Information Processing Systems*, 512–518 (2000).

[105] Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747* (2016).

[106] Constantinopoulos, C., Titsias, M. K. & Likas, A. Bayesian feature and model selection for gaussian mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**, 1013–1018 (2006).

[107] Steele, R. J. & Raftery, A. E. Performance of bayesian model selection criteria for gaussian mixture models. *Frontiers of Statistical Decision Making and Bayesian Analysis* **2**, 113–130 (2010).

[108] Huang, T., Peng, H. & Zhang, K. Model selection for gaussian mixture models. *Statistica Sinica* 147–169 (2017).

[109] Lancichinetti, A., Sirer, M. I., Wang, J. X., Acuna, D., Körding, K. & Amaral, L. A. N. High-reproducibility and high-accuracy method for automated topic classification. *Physical Review X* **5**, 011007 (2015).

[110] Gerlach, M., Farb, B., Revelle, W. & Amaral, L. A. N. A robust data-driven approach identifies four personality types across four large data sets. *Nature Human Behaviour* **2**, 735–742 (2018).

[111] Ozer, M., Sapienza, A., Abeliuk, A., Muric, G. & Ferrara, E. Discovering patterns of online popularity from time series. *Expert Systems with Applications* 113337 (2020).