

Capítulo

2

Computação Paralela com a Linguagem R: técnicas, ferramentas e aplicações

Carlos Amaral Hölbig¹ - holbig@upf.br

Angela Mazzonetto² - angelamazzonettofw@gmail.com

Willingthon Pavan³ - pavan@upf.br

Resumo

O processamento, a análise e a visualização de grande quantidade de dados estão presentes em aplicações nas mais diversas áreas como biologia, química, física, estatística, geografia, meteorologia, agronomia, entre outras. A linguagem de programação R é uma ferramenta computacional eficiente para realizar estas tarefas, possibilitando, ainda, que elas sejam realizadas em ambientes computacionais de alto desempenho, objetivando sua paralelização e, conseqüentemente, obtendo um melhor desempenho computacional. Este minicurso apresenta uma visão geral da linguagem R, destacando suas principais funcionalidades e faz um levantamento do estado-da-arte do seu uso em ambientes computacionais paralelos, apresentando, principalmente, os seus principais pacotes (bibliotecas)

¹Graduado em Tecnologia em Processamento de Dados pela Universidade Católica de Pelotas (1993), mestre e doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (1996/2005) com doutorado sandwich na Bergische Universität Wuppertal (Alemanha). Realizou pós-doutorado no Instituto Nacional de Pesquisas Espaciais (INPE) em 2015. Atualmente é professor titular III da Universidade de Passo Fundo, vinculado ao Programa de Pós-Graduação em Computação Aplicada e aos cursos de Ciência da Computação e em Engenharia de Computação. Atua principalmente nos seguintes temas: computação científica, ambientes de alto desempenho e modelagem e simulação computacional.

²Graduada em Ciência da Computação pela Universidade Regional Integrada do Alto Uruguai e das Missões (2013), mestre em Computação Aplicada pela Universidade de Passo Fundo (2016). Atualmente é Bolsista DTI-B do CNPq vinculada à projeto sobre mudanças climáticas coordenado pelo INPE, em parceria com a UPF. Atua principalmente nos seguintes temas: ambientes de alto desempenho e modelagem e simulação computacional.

³Graduado em Ciência da Computação pela Universidade de Passo Fundo (1994), mestre em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (2000), doutor em Agronomia pela Universidade de Passo Fundo (2007). Realizou pós-doutorado na Universidade da Florida/USA (2008 e 2009). É professor titular do Programa de Pós-Graduação em Computação Aplicada da Universidade de Passo Fundo com experiência na área de Ciência da Computação e Agronomia (Fitopatologia), ênfase em Modelos Analíticos e de Simulação, atuando principalmente nos seguintes temas: modelos matemáticos e de simulação, computação móvel e linguagens de programação.

que possibilitam a sua execução em ambientes paralelos como, por exemplo, em grids, em clusters, em processadores multicore e em placas gráficas.

2.1. Introdução

Inúmeras organizações atualmente têm a necessidade de realizar o processamento e análise de uma grande quantidade de dados em tempo computacional hábil. Por este motivo, a utilização de ferramentas computacionais torna-se indispensável. Uma ferramenta que pode ser utilizada para suprir estas necessidades é a linguagem R [R Core Team 2016]. De acordo com Torgo [Torgo 2009], o R é uma linguagem de programação *open source* e um ambiente para computação estatística, modelagem e visualização de dados. Trata-se de uma linguagem de programação especializada em computação estatística⁴. Além disso, o R está disponível para sistemas operacionais Linux, Unix, Windows e Mac OS X.

Uma grande vantagem desta linguagem é a grande disponibilidade de pacotes voltados a solução dos mais diversos tipos de problemas matemáticos e estatísticos e de aplicações específicas. Grande parte destes pacotes estão disponibilizados em um repositório próprio do R, o The Comprehensive R Archive Network - CRAN⁵. Como neste repositório, atualmente (dados de fevereiro de 2017), existem mais de 10 mil pacotes, o CRAN criou uma área em seu site chamada de CRAN *Task Views*⁶ que objetiva agrupar parte destes pacotes em algumas áreas de aplicação como, por exemplo, computação de alto desempenho, *machine learning*, gráficos, matemática numérica, entre outras.

A interface do R é bastante simples, basicamente baseada em linha de comandos. Para a elaboração de programas em R recomenda-se o uso de algum editor de texto ou de alguma *Integrated Development Environment* (IDE). A IDE mais popular e mais utilizada para o desenvolvimento de programas em R é o RStudio⁷ disponível para sistemas operacionais Linux, Windows e Mac OS X. Um *cheat sheet* com os conceitos básicos para uso da IDE está disponível em goo.gl/XFbXWY.

A linguagem R foi criada por Ross Ihaka e Robert Gentleman na Universidade de Auckland, Nova Zelândia, em 1993. Ela é originada da linguagem S e sua funcionalidade é estendida por meio do uso de pacotes (bibliotecas). Algumas características importantes da linguagem R podem ser resumidas em:

- Sua implementação é em formato livre;
- Ela é *case-sensitive* - objeto \neq OBJETO \neq Objeto;
- O R é uma linguagem orientada à objetos, tudo no R é tratado como um objeto;
- Não é necessário declarar o tipo numérico das variáveis (exceto para os tipos de dados descritos abaixo);
- Possui tipos de dados especiais: arrays, vetores, matrizes, data-frames e listas;
- Os índices de estruturas uni ou multidimensionais iniciam em 1;

⁴Página oficial do projeto R: <https://www.r-project.org/>

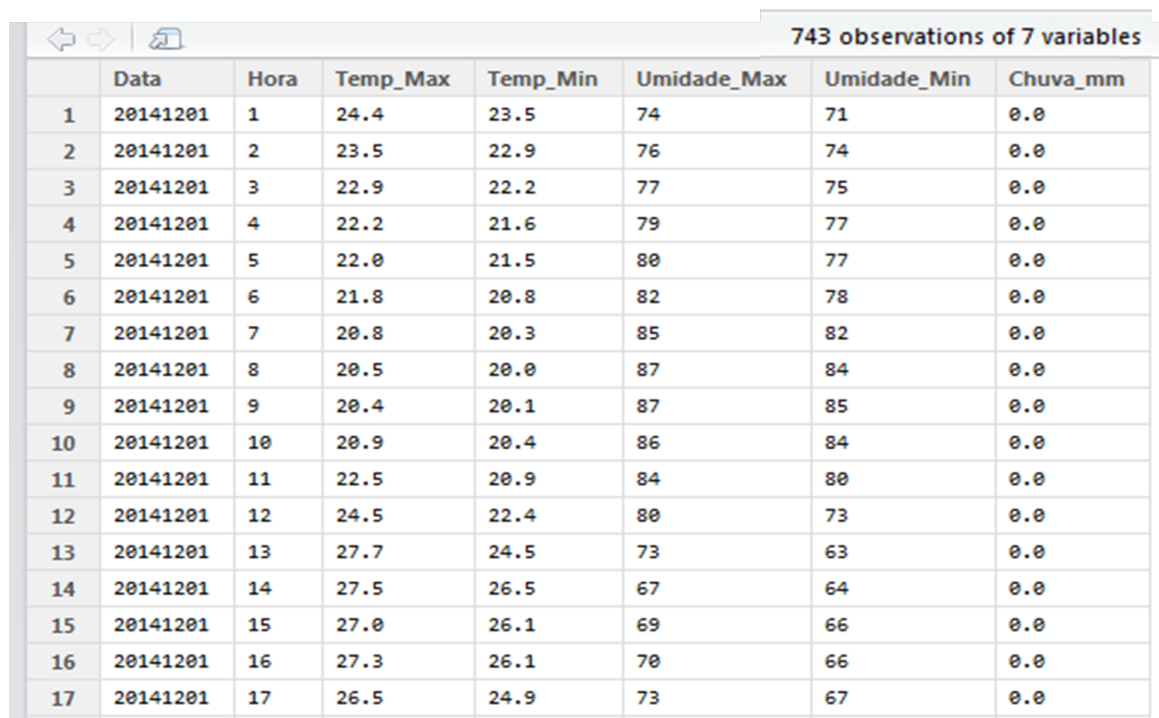
⁵Repositório CRAN: <https://cran.r-project.org/>

⁶CRAN *Task Views*: <https://cran.r-project.org/web/views/>

⁷RStudio: <https://www.rstudio.com/products/rstudio/>

- As dimensões de suas estruturas são definidas da seguinte forma: dimensão = 1 = linha; dimensão = 2 = coluna; dimensão = 3, ... = ...;
- Possui várias funções matemáticas e estatísticas pré-definidas;
- Possui funções que facilitam a geração de sequências e de repetições;
- Possibilita a fácil utilização e manipulação de vetores e matrizes.

Uma das estruturas mais utilizadas em R é o `data.frame`. Ele é uma estrutura semelhante à uma matriz porém com cada coluna sendo tratada separadamente. Desta forma pode-se ter colunas de valores numéricos e colunas de caracteres no mesmo objeto. Note que, entretanto, dentro de uma mesma coluna todos elementos ainda serão forçados a serem do mesmo tipo. Exemplo da visualização de um `data.frame` no RStudio é apresentado na Figura 2.1.



	Data	Hora	Temp_Max	Temp_Min	Umidade_Max	Umidade_Min	Chuva_mm
1	20141201	1	24.4	23.5	74	71	0.0
2	20141201	2	23.5	22.9	76	74	0.0
3	20141201	3	22.9	22.2	77	75	0.0
4	20141201	4	22.2	21.6	79	77	0.0
5	20141201	5	22.0	21.5	80	77	0.0
6	20141201	6	21.8	20.8	82	78	0.0
7	20141201	7	20.8	20.3	85	82	0.0
8	20141201	8	20.5	20.0	87	84	0.0
9	20141201	9	20.4	20.1	87	85	0.0
10	20141201	10	20.9	20.4	86	84	0.0
11	20141201	11	22.5	20.9	84	80	0.0
12	20141201	12	24.5	22.4	80	73	0.0
13	20141201	13	27.7	24.5	73	63	0.0
14	20141201	14	27.5	26.5	67	64	0.0
15	20141201	15	27.0	26.1	69	66	0.0
16	20141201	16	27.3	26.1	70	66	0.0
17	20141201	17	26.5	24.9	73	67	0.0

Figura 2.1. Exemplo de um `data.frame` em R

Os conceitos básicos da linguagem, tipos de dados e funções podem ser obtidos em dois *cheat sheets* disponibilizados pela RStudio. O primeiro, com os conceitos básicos da linguagem, chamado de “*Base R*”, está disponível em <https://goo.gl/ZvWlg8> e o segundo, chamado de “*Advanced R*”, está disponível em <https://goo.gl/GFCq70>. Além deste material, indica-se as apostilas “Introdução à Programação em R” de Luís Torgo [Torgo 2009] e “Introdução ao Ambiente Estatístico R” de Paulo Justiniano Ribeiro Junior, disponível em <http://www.leg.ufpr.br/~paulojus/embrapa/Rembrapa/Rembrapa.pdf>.

2.2. Análise e visualização de dados

A análise e a visualização de dados são, provavelmente, um dos maiores pontos fortes da linguagem R. Na verdade, em sua concepção inicial, ele foi projetado exatamente para

estas atividades. Claro que, como tudo no R, a criação de pacotes possibilitou uma extraordinária expansão de suas funcionalidades, principalmente quando se trata da geração de gráficos mais complexos e interativos e da integração com tecnologias web.

O R em sua distribuição já apresenta uma série de funcionalidades estatísticas contemplando a maioria das fórmulas básicas como, por exemplo, média, desvio padrão, mediana, quantis, variância, co-variância e correlação. Isto também ocorre quando se trata de gráficos. Na distribuição do R já vem instalados os pacotes **lattice** e **graphics** que disponibilizam gráficos tradicionais como de barra, histogramas e de linhas. Para visualizar estes gráficos básicos basta digitar na linha de comando do R o comando `demo(graphics)`.

No momento que necessita-se de funções estatísticas avançadas ou de gerar gráficos mais elaborados, o uso de pacotes é um grande aliado para os programadores em R. No caso de gráficos, pacotes como o **ggplot2**, o **Rgraphviz**, o **rgl**, o **dygraphs** e o **plot3D** oferecem opções de plotagem de gráficos mais elaborados, com interatividade, de grafos e de gráficos em três dimensões. Pacote como o **animation** permite a criação de animações e sua gravação em diversos formatos.

Outra vantagem do R é sua integração e interação com mapas que pode ser realizada por pacotes como o **RgoogleMaps**, **maps** e **leaflet**.

Aliando todas estas funcionalidades, a visualização de dados e interação online com os usuários na web por meio do R é outra característica muito utilizada pelos usuários do R, tanto simples usuários como empresas, universidades, centros de pesquisa e entidades governamentais.

Esta utilização deu-se, em parte, ao pacote **shiny** [Chang et al. 2017], também desenvolvido pela RStudio. O **shiny** é um pacote *open source* que disponibiliza um poderoso e elegante *framework* web para a construção de aplicações web usando o R. O **shiny** possibilita a integração do R com as tecnologias HTML, Javascript e CSS.

Um exemplo do uso do **shiny** e a integração com outras tecnologias pode ser visto na Figura 2.2. Nela está representada a interface de um projeto desenvolvido pelo PPGCA/UPF em parceria com o CPTEC/INPE que trata da disponibilização de dados de estações climáticas localizadas no Brasil e sua visualização em diversos formatos (mapas, gráficos e tabelas). Neste projeto, além do **shiny**, são utilizados os pacotes **shinydashboard** (para a criação de *dashboards*), **dygraphs** (para a geração de gráficos interativos), **googleVis** (para um interface entre o R e o Google Chart API), **ggplot2** (para a geração de gráficos avançados), **leaflet** (para a geração e manipulação de mapas) e **DT** (para visualização de data frames e tabelas).

Outro exemplo é um serviço que disponibiliza mapas de risco de doenças na cultura do trigo⁸ no site do projeto Sisalert (Sistemas de alerta de doenças em culturas agrícolas), desenvolvido pelo grupo de pesquisa Mosaico do PPGCA/UPF em parceria com a Embrapa Trigo de Passo Fundo (RS).

⁸Sisalert - Wheat Blast Map: (http://dev.sisalert.com.br:8080/webinar/wheat_blast_map/)

eficientes.

Uma das técnicas que pode ser utilizada para evitar os laços ao percorrer um vetor são as funções vetorizáveis do R. Se, por exemplo, fosse necessário somar todos os elementos de uma matriz, isso precisaria de dois laços de repetição para percorrer todas as linhas e colunas somando as mesmas. Porém, no R, isso se torna simples, pois somente é necessário escrever o comando `s <- sum(m)`, onde `s` é a variável que receberá o resultado, `sum` a função que realizará o somatório e `m` a matriz. Além da simplicidade, a versão com laços de repetição demoraria um tempo considerável em sua execução quando aplicada a matrizes muito grandes. “O R tem ainda outras funções que podem ser explicitamente usadas para vetorizar operações. Supõe-se que se pretende saber o valor médio de cada coluna de uma matriz. No R pode ser aplicada a função `mean` a cada uma das colunas da matriz” [Torgo 2009]. Porém, existe a função `apply` que facilita a realização desta operação. Ela permite aplicar qualquer função a uma das dimensões de uma matriz ou de um vetor.

Portanto, ainda conforme Torgo [Torgo 2009], estas funções representam um ganho de eficiência computacional significativo em relação à alternativa dos laços. Porém, pela sua generalidade, a função `apply` tem ela própria algumas ineficiências, e existem alternativas ainda mais rápidas para utilizações comuns, como é o caso da média. Por exemplo, uma forma mais rápida de obter as médias das colunas seria por meio da função `colMeans`.

Entre outras funções do R relacionadas à execução de operações estão: a `tapply` em subgrupos de dados, determinados por valores de fatores, `sapply`, que permite aplicar uma função a todos os elementos de um vetor, a função `lapply` e por fim a função `rapply` que permite aplicar uma função a certos elementos de um vetor ou substituí-los por outros.

Também é importante ressaltar que para as operações de álgebra linear a linguagem R faz uso implícito das rotinas da biblioteca BLAS. Esta inclui sub-rotinas para operações algébricas básicas, como produto escalar e multiplicação de matrizes. Para realizar a comunicação entre os processos é utilizada a biblioteca *Basic Linear Algebra Communication Subprogram* (BLACS), que é uma biblioteca de troca de mensagens indicada para álgebra linear. Esta funciona de modo que o modelo computacional consiste de uma grade de processos de uma ou duas dimensões, sendo que cada processo armazena uma parte da matriz ou vetor. As rotinas de comunicação são síncronas e operam sobre MPI ou PVM [Rodrigues 2014].

2.4. O R em ambientes paralelos

Esta seção apresenta uma visão geral e um levantamento do estado-da-arte do uso da linguagem R em ambientes computacionais paralelos, apresentando, principalmente, os seus principais pacotes (bibliotecas) que possibilitam a sua execução em ambientes como grids, clusters, processadores multicore e GPU's. Este levantamento foca em pacotes para uso geral, não abordando pacotes que foram desenvolvidos para uso específico em aplicações reais paralelizadas. Uma lista atual destes pacotes, tanto os de uso geral como os vinculados à aplicações, pode ser encontrada na página específica para Computação de

Alto Desempenho com o R¹⁰, disponibilizada pelo CRAN.

Alguns dos principais pacotes para uso geral do R em ambientes computacionais paralelos são descritos a seguir:

- **Clusters de Computadores:** a linguagem R possui conjunto de vários pacotes para seu uso clusters de computadores. Estes pacotes possuem características que vão desde o uso em operações paralelas simples, a paralelização de laços de repetição até a integração com o MPI. Entre os pacotes existentes pode-se citar **Rmpi**, **snow**, **snowFT**, **snowfall**, **foreach**, **doMC**, **doSNOW**, **doMPI** e **Rdsm**. Na Tabela 2.1 é apresentada uma breve descrição destes pacotes. Ressalta-se que, a partir da versão 2.14.0 do R, foi incorporada em sua distribuição o pacote **parallel** que incorpora cópias (ligeiramente revisadas) dos pacotes **multicore** e **snow**.

Tabela 2.1. Principais pacotes para uso do R em clusters de computadores

PACOTE	DEPENDÊNCIA	DESCRIÇÃO	PLATAFORMA
Rmpi	R ($\geq 2.15.1$)	Uma interface (<i>wrapper</i>) para MPI.	Linux, Windows
snow	R ($\geq 2.13.1$)	Suporte para simples computações paralelas em R.	Linux, Windows, OS X Mavericks
snowfall	R (≥ 2.10), snow	<i>Wrapper</i> de usabilidade do snow para facilitar o desenvolvimento de programas paralelos em R.	Linux, Windows, OS X Mavericks
foreach	R ($\geq 2.5.0$)	Suporte para o uso do construtor de looping foreach e para a exceção de laços em paralelo.	Linux, Windows, OS X Mavericks
doMC	R ($\geq 2.14.0$), foreach ($\geq 1.2.0$), iterators ($\geq 1.0.0$), parallel	Fornece um <i>backend</i> paralelo para a função <code>%dopar%</code> usando a funcionalidade multicore do pacote parallel .	Linux, OS X Mavericks
doSNOW	R ($\geq 2.5.0$), foreach ($\geq 1.2.0$), iterators ($\geq 1.0.0$), snow ($\geq 0.3.0$), utils	Fornece um <i>backend</i> paralelo para a função <code>%dopar%</code> usando o pacote snow .	Linux, Windows, OS X Mavericks
doMPI	R ($\geq 2.14.0$), foreach ($\geq 1.3.0$), iterators ($\geq 1.0.0$), Rmpi ($\geq 0.5-7$)	Fornece um <i>backend</i> paralelo para a função <code>%dopar%</code> usando o pacote Rmpi .	Linux, Windows
Rdsm	bigmemory ($\geq 4.0.0$), parallel	Fornece um ambiente de programação com suporte a <i>threads</i> em R. Possibilita trabalhar de maneira eficiente com matrizes de grande porte.	Linux, OS X Mavericks

¹⁰Task View: High-Performance and Parallel Computing with R:

<http://cran.r-project.org/web/views/HighPerformanceComputing.html>

- **Processadores Multicore:** para ambientes com processadores multicore pode-se destacar os pacotes **fork** e o **multicore**, descritos na Tabela 2.2.

Tabela 2.2. Principais pacotes para uso do R em processadores multicore

PACOTE	DEPENDÊNCIA	DESCRIÇÃO	PLATAFORMA
fork	nenhuma	Funções em R para manipulação de múltiplos processos	Unix, Windows, OS X Mavericks
multicore	R ($\geq 2.14.0$)	Código para processamento paralelo do R em máquinas com múltiplos cores ou CPU's	Linux, Windows, OS X Mavericks

- **Placas Gráficas ou *Graphics Processing Unit* (GPU):** o desenvolvimento de pacotes para uso do R em GPU's vem crescendo muito nos últimos anos, principalmente para uso em GPU's da NVIDIA. Como exemplos de pacotes desenvolvidos para uso em GPU's pode-se citar o **gputools**, **OpenCL**, **HiPLARM**, **gmatrix** e **gpuR**. Na Tabela 2.3 é apresentada uma breve descrição destes pacotes.

Tabela 2.3. Principais pacotes para uso do R em GPU's

PACOTE	DEPENDÊNCIA	DESCRIÇÃO	PLATAFORMA
gputools	R ($\geq 3.1.2$)	Disponibiliza uma interface em R para um completo conjunto de funções que são implementadas utilizando o NVIDIA CUDA <i>toolkit</i> .	Linux
OpenCL	R ($\geq 2.0.0$)	Disponibiliza uma interface em R para o OpenCL <i>toolkit</i> .	Linux
HiPLARM	R (≥ 2.14), Matrix , methods	Pacote para computação de alto desempenho que oferece suporte ao pacote Matrix do R.	Unix
gmatrix	methods , stats	Pacote para uso do R em GPU's da NVIDIA.	Unix
gpuR	R ($\geq 3.0.2$), methods , utils	Fornece acesso as funcionalidades das GPU's aos objetos do R.	Linux

- **Grid de Computadores:** Não há um desenvolvimento significativo de pacotes para este tipo de ambiente. Segundo o CRAN, atualmente, dois pacotes são disponibilizados: **multiR** e **biocep-distrib**. O pacote **multiR** [Grose 2008] foi apresentado na conferência useR! 2008 mas não foi disponibilizado ao público. Ele prometia oferecer um *framework* estilo o pacote **snow** em uma plataforma de grid de computadores. Já o pacote **biocep-distrib** é um projeto desenvolvido na China que oferece um *framework* baseado em Java para computadores locais, em grid ou para *cloud computing*. Este projeto ainda está em desenvolvimento.

2.5. Big data e large memory

Sabe-se que a linguagem R é utilizada para fazer cálculos estatísticos, análise de dados, entre outros. No entanto, algumas organizações precisam que uma grande quantidade de

dados seja processada, analisada e armazenada. Uma dificuldade encontrada no R é ligada à necessidade de armazenamento desta grande quantidade de dados sendo que todos os dados necessitam ser mantidos na memória. Mesmo para os computadores modernos com grande capacidade de armazenamento, isto pode se apresentar como um desafio significativo. O problema não está apenas na capacidade mas, também, em como organizar todos os dados analisados na memória. O desempenho e a escalabilidades também são pontos que devem ser considerados [Rickert 2014]. A capacidade de manipular esta grande quantidade de dados e gerenciar seu uso em memória é um dos pontos fortes da linguagem R. **RevoScaleR**, **parallel** e **RHadoop** são exemplos de pacotes que auxiliam neste tipo de tarefa.

O pacote chamado **RevoScaleR** foi desenvolvido visando resolver desafios de capacidade, desempenho e estabilidade. Ele oferece bons resultados no que tange ao desempenho e à capacidade para as análises estatísticas no ambiente R, possibilitando que os usuários de R possam processar, visualizar e modelar maiores conjuntos de dados em poucas frações de tempo, sem a necessidade de implantar um hardware mais caro ou especializado [Pukacki et al. 2006]. O **RevoScaleR** é um pacote que possibilita o uso das funções da ferramenta ScaleR da Microsoft¹¹ que é uma coleção de funções para aplicação em *Data Science* em escala.

Conforme descrito por Rickert [Rickert 2014], as funções de manipulação e de análise de dados no **RevoScaleR** são adequadas tanto para pequenos como para grandes conjuntos de dados, mas são particularmente úteis em três situações comuns:

- analisar conjuntos de dados que são grandes demais para armazenar na memória;
- realizar cálculos distribuídos ao longo de vários núcleos, processadores ou nós em um cluster;
- criar rotinas de análise de dados escaláveis que podem ser desenvolvidas localmente, com conjuntos de dados menores, então implantado em dados maiores em um cluster de computadores.

Estes são candidatos ideais para o pacote **RevoScaleR** porque ele baseia-se no conceito de operar em blocos de dados, usando algoritmos de atualização. Ainda de acordo com Rickert [Rickert 2014], o pacote **RevoScaleR** fornece um mecanismo para a ampliação da linguagem R em lidar com grandes conjuntos de dados. Existem três componentes principais para este pacote:

- um novo formato de arquivo projetado especialmente para grandes arquivos;
- implementações de memória externa dos algoritmos estatísticos mais comumente usados com grandes conjuntos de dados;
- um quadro de programação extensível que permite que programadores de R e C++ possam escrever seus próprios algoritmos de memória externa, podendo tirar vantagem de novas capacidades de *Big Data*.

¹¹Manual disponível em <https://msdn.microsoft.com/en-us/microsoft-r/scaler-user-guide-introduction>

Algumas ferramentas em R têm uma aplicabilidade mais flexível e podem ser utilizadas em clusters ou processadores multicore. Duas delas são abordadas a seguir.

- **RHadoop**: Hadoop foi desenvolvido em Java e esta é a principal linguagem de programação para o Hadoop. Apesar de Java ser a linguagem principal, pode ser utilizada qualquer outra linguagem como, por exemplo, Python, R ou Ruby. Ele é chamado de “*streaming API*”. Porém, nem todos os recursos disponíveis no Java estão disponíveis em R. Infelizmente, *streaming API* não é fácil de ser utilizado e é por isso que o **RHadoop** foi criado. De acordo com [Hadoop 2013], ele é um conjunto dos seguintes pacotes para a linguagem R:
 - **Rmr**: oferece interface MapReduce, mapeador e redutor podem ser descritos em código R e, em seguida, chamados a partir do R;
 - **Rhdfs**: fornece acesso a HDFS, usando funções R simples e permitindo a cópia de dados entre a memória R, o sistema de arquivos local e o HDFS;
 - **Rhbase**: necessário para realizar o acesso da HBase;
 - **Plyrmr**: permite operações de manipulação de dados comuns.
- **parallel**: de acordo com Samatova [Samatova et al. 2006], o pacote **parallel** pode fornecer uma capacidade computacional paralela transparente para o usuário. Por meio do **parallel** o usuário pode configurar o ambiente paralelo, distribuir dados e realizar a computação paralela necessária mas mantendo a mesma interface *look-and-feel* do sistema R. Dois grandes níveis de paralelismo são suportados dentro do **parallel**: paralelismo de dados e paralelismo de tarefas. Com uma abordagem paralela de dados (por exemplo, o agrupamento paralelo ou PCA), os dados são divididos entre vários processos que executam mais ou menos a mesma tarefa em seus blocos de dados para obter os resultados desejados. Com a tarefa de paralelismo, um determinado trabalho é dividido em várias tarefas que são executadas em paralelo para obter os resultados esperados. O **parallel** realiza o particionamento das tarefas, agendamento de tarefas e mapeamento de recursos computacionais que exigem nenhuma ou pouca modificação nos códigos R. No paralelismo de dados são fornecidos “ganchos” para incorporar uma rotina de análise de dados em paralelo em um ambiente com baixa sobrecarga no desempenho. Ainda como parte do pacote, foi desenvolvida a biblioteca **RScalAPACK** que substituiu as rotinas de análise de dados em R que são baseados na LAPACK, otimizando o uso das rotinas da ScaLAPACK. A nova interface simulou a interface R original por meio de uma única chamada de função. As pequenas alterações introduzidas na interface não são obrigatórias e são fornecidas com a finalidade de ter o controle em nível de usuário versus uma série de rotinas paralelas [Samatova et al. 2006]. A arquitetura da **RScalApack** é motivada pela possibilidade de realizar computações paralelas eficientes em grandes conjuntos de dados fora do processo R pai e gerenciar a execução dos processos externos. A entidade computacional em execução fora do processo de R pai (processos gerados) poderia ser qualquer sistema computacional paralelo com a distribuição de dados bem definida e com estratégias de coleta de resultado. A entidade de gestão Agente Paralelo (AP) coordena o fluxo de dados

entre o meio ambiente R e a unidade computacional paralela. O AP foi projetado para fornecer um único ponto de entrada para o ambiente R para a manipulação de dados [Samatova et al. 2006].

2.6. Aplicações

O uso da linguagem R tem aumentado muito nos últimos anos. Segundo o ranking da IEEE Spectrum's de 2016¹², a linguagem R é a quinta linguagem mais popular devido, principalmente, a sua utilização em aplicações de *big data*. Até por isso pode-se esperar que aplicações paralelas comecem a serem solucionadas por meio de sua utilização.

Vários dos trabalhos que abordam aplicações implementadas em R e que utilizam de alguns dos pacotes descritos na seção 2.4, entre outras ferramentas, tratam da paralelização de modelos geoestatísticos [Beleti 2013], da paralelização de códigos em R com OpenMPstyle [Jiang et al. 2012], de cálculos de bioestatística paralelas em um supercomputador [Samatova et al. 2006] e da utilização do paradigma MapReduce para R [Stokely et al. 2011]. Outros trabalhos tratam da utilização de ferramentas R para computação em Grid [Vera and Suppi 2010], do uso do pacote paralelo Parallel R [Samatova et al. 2006], do emprego do R para reduzir complexidade da computação de alto desempenho [Mitchell et al. 2011], de um sistema de simulação agrícola paralelizado para executar em Grid de computadores [Bryan et al. 2014], da avaliação de ferramentas para HPC [Bryan 2013], do desenvolvimento de plataformas escaláveis utilizando algoritmos desenvolvidos em R [Das et al. 2010], do uso do R em conjunto com a linguagem C++ [Eddelbuettel and Sanderson 2014], da abordagem do uso de pacotes e de ferramentas para paralelização em R [Eugster et al. 2011, Kumar et al. 2011], da execução de algoritmos de R na infraestrutura de nuvem [Patel et al. 2012] e da abordagem paralela utilizada no projeto Bioconductor [Petrone et al. 2011, Gentleman et al. 2004].

Um resumo destas aplicações pode ser encontrado em trabalhos como de Mazzonetto [Mazzonetto 2016]. Além dele, uma visão geral das técnicas de computação paralela em clusters de computadores, em sistemas multicore e em computação em grid com R são apresentadas no trabalho de Schmidberger et al [Schmidberger et al. 2009].

2.7. Testes utilizando o R em ambientes paralelos

Com o objeto de ilustrar o uso de pacotes e rotinas do R para realizar tarefas em ambientes paralelos, alguns testes foram realizados. Estes testes foram desenvolvidos no grupo de pesquisa ComPaDi, vinculado ao PPGCA/UPF. O computador utilizado possui um processador Intel Core i7 920, que opera à frequência de 2.66 Ghz, com 8 MB de cache L2, 8 GB de memória RAM, sistema operacional Ubuntu 14.04 64 bits e placa de vídeo GeForce GTS250 1GB DDR3 ECS. Os softwares utilizados foram a linguagem R (versão 3.3.2 de 64 bits), a IDE RStudio (versão 1.0.136) e os pacotes **foreach** (1.4.3), **snow** (0.4-2), **doSNOW** (1.0.14), **iterators** (1.0.8), **parallel** (3.3.2) e **gputools** (1.1).

A Figura 2.3 apresenta dois cálculos. O primeiro realiza o cálculo sequencial da soma entre duas matrizes de 2000 elementos cada. Na linha 12 a soma é realizada, por

¹²Disponível em: <http://spectrum.ieee.org/computing/software/the-2016-top-programming-languages>

```

1  ordem = 2000
2  A = array(0, dim = c(ordem, ordem))
3  B = array(0, dim = c(ordem, ordem))
4  S1 = array(0, dim = c(ordem, ordem))
5
6  for(i in 1:ordem)
7    for(j in 1:ordem)
8      {
9        A[i,j] = round(runif(1)*10);
10       B[i,j] = round(runif(1)*10);
11      }
12  system.time(S <- A+B)
13  #  usuario      sistema decorrido
14  #    0.02      0.00      0.01
15
16  system.time(
17    for(i in 1:ordem)
18      for(j in 1:ordem)
19        {
20          S1[i,j] <- A[i,j] + B[i,j];
21        })
22  #  usuario      sistema decorrido
23  #    9.39      0.03      9.43
24
25  system.time(X <- sum(A))
26  #  usuario      sistema decorrido
27  #    0.02      0.00      0.01
28
29  X <- 0
30  system.time(
31    for(i in 1:ordem)
32      for(j in 1:ordem)
33        {
34          X <- A[i,j] + X;
35        })
36  #  usuario      sistema decorrido
37  #    3.11      0.00      3.13

```

Figura 2.3. Programa em R com uso de funções otimizadas pela linguagem

meio do comando `S <- A+B` (a soma em dados do tipo `array` é vetorizada pelo R) e a operação resultou em um tempo computacional de 0,01s. Se esta operação fosse realizada na forma tradicional, pelo uso de laços de repetição - linhas 17 a 21, o tempo de execução seria de 9,43s. Já o segundo, realiza a soma dos elementos de uma matriz. Na linha 25, utilizando a função vetorizada `sum`, o tempo de execução ficou em 0,01s. A mesma operação, mas utilizando dois laços de repetição (linhas 31 a 35), o tempo de execução passou para 3,13s. Este fato demonstra a importância do uso das funções vetorizáveis

disponibilizadas pelo R (neste caso a função `sum` e o uso dos tipos de dados especiais do R - no exemplo o tipo de dado `array`).

A Figura 2.4 apresenta um programa com uso dos pacotes **foreach** e **doSNOW**. Este programa realizou a chamada da função do usuário `check` 100 vezes. Pode ser observado que o tempo decorrido foi de 1,84s com o laço **foreach** (linha 14) utilizando a opção `%dopar%` (execução em paralelo utilizando 4 processadores do computador, especificados na linha 3). Posteriormente, com a utilização de um laço `for` simples (linha 19), o tempo foi de 3,82s (execução sequencial).

```

1  library(foreach)
2  require(doSNOW)
3  cl<-makeCluster(4) # numero de cores
4  registerDoSNOW(cl)
5
6  # cria uma funcao para rodar em cada iteracao do loop
7  check <- function(n) {
8    for(i in 1:1000)
9      {
10       sme <- matrix(rnorm(100), 10,10)
11       solve(sme)
12     }
13 }
14 times <- 100      # numero de vezes para rodar o loop
15 system.time(x <- foreach(j=1:times ) %dopar% check(j))
16 # usuario sistema decorrido
17 #    0.08      0.02      1.84
18
19 system.time(for(j in 1:times ) x <- check(j))
20 # usuario sistema decorrido
21 #    3.80      0.00      3.82
22 stopCluster(cl)

```

Figura 2.4. Programa em R com uso dos pacotes `foreach` e `doSNOW`

A Figura 2.5 apresenta o uso do pacote **gputools** para a multiplicação entre duas matrizes A e B, de ordem 2000 cada. Pode ser observado que o tempo decorrido primeiramente sem uso de GPU, execução sequencial, conforme descrito na linha 14, foi de 1,665s. Posteriormente, utilizando a função `gpuMatMult` do pacote **gputools** (linha 18), utilizando a placa gráfica do ambiente computacional, o tempo foi de 0,025s.

Por fim, como último exemplo do uso do R em paralelo, apresenta-se uma (muito) simples comparação entre um programa em MPI (Figura 2.6) e um em R utilizando o pacote **Rmpi** (Figura 2.7). Ambos os programas implementam o “famoso” “*Hello world!!!*”.

```

1  library(gputools)
2  ordem = 2000
3  A = array(0, dim = c(ordem, ordem))
4  B = array(0, dim = c(ordem, ordem))
5  C = array(0, dim = c(ordem, ordem))
6
7  for(i in 1:ordem)
8    for(j in 1:ordem)
9      {
10         A[i,j] = round(runif(1)*10);
11         B[i,j] = round(runif(1)*10);
12      }
13
14  system.time(C<-A%*%B)
15  #  usuario      sistema decorrido
16  #    1.662      0.003      1.665
17
18  system.time(gpuMatMult(A, B))
19  #  usuario      sistema decorrido
20  #    0.023      0.002      0.025

```

Figura 2.5. Programa em R com uso do pacote gputools

```

1  #include <stdio.h>
2  #include "mpi.h"
3
4  int main (int argc, char** argv)
5  {
6      int rank, size, nameLen;
7      char processorName [MPI_MAX_PROCESSOR_NAME];
8
9      MPI_Init(&argc, &argv);
10     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
11     MPI_Comm_size(MPI_COMM_WORLD, &size);
12
13     MPI_Get_processor_name(processorName, &nameLen);
14
15     printf("Hello, _rank_%d, _size_%d, on _processor_%s\n",
16           rank, size, processorName);
17
18     MPI_Finalize();
19     return 0;
20 }

```

Figura 2.6. Programa exemplo em MPI

2.8. Conclusões

Observa-se que o processamento e análise dos dados atualmente é algo imprescindível para as organizações que manipulam grande quantidade de dados. Por este motivo a lin-

```
1 #!/usr/bin/env r
2
3 library(Rmpi) #calls MPI_Init
4
5 rk <- mpi.comm.rank(0)
6 sz <- mpi.comm.size(0)
7 name <- mpi.get.processor.name()
8 cat("Hello", rank, " size ", sz, " on ", name, "\n")
```

Figura 2.7. Programa exemplo em MPI

guagem R torna-se uma forte aliada para criação de modelos que realizem estas tarefas de forma eficiente. Porém, inúmeras vezes, com apenas as funções básicas desta linguagem não é o suficiente para que o processamento dos dados seja feito em tempo computacional hábil. Consequentemente, surge a necessidade de buscar novos meios de proporcionar este processamento ainda mais eficiente em termos de desempenho computacional. Observa-se que a paralelização é uma opção significativa. Com o estudo da linguagem R é possível encontrar diferentes opções para realizar a paralelização, desde funções da própria linguagem até pacotes desenvolvidos especificamente para determinadas aplicações.

Neste minicurso foram abordadas as diferentes ferramentas para auxiliar o paralelismo em R. Também se realizou alguns testes com a utilização da função `sum` própria do R e dos pacotes **foreach**, **doSNOW**, **gputools** e **Rmpi**. Com estes testes pode-se constatar a validade e facilidade do uso destas ferramentas em ambientes de alto desempenho.

Por fim, existe uma diversidade de ferramentas que objetivam a paralelização no R, de acordo com a necessidade da aplicação, com a estrutura de hardware e com o sistema operacional envolvido. Algumas destas ferramentas são mais recomendadas do que outras. Mais aspectos sobre o desempenho computacional do R em ambientes de alto desempenho podem ser conhecidos mediante a implementação ou as comparações abordadas nas aplicações descritas na literatura.

Referências

- [Beleti 2013] Beleti, C. R. (2013). Paralelização de aplicações na plataforma R. Master's thesis, Universidade Estadual de Maringá. Disponível em: <http://www.din.uem.br/~mestrado/diss/2013/beletijr.pdf>. Acesso em: 15 fev. 2017.
- [Bryan 2013] Bryan, B. A. (2013). High-performance computing tools for the integrated assessment and modelling of social–ecological systems. *Environmental Modelling & Software*, 39:295 – 303. Thematic Issue on the Future of Integrated Modeling Science and Technology.
- [Bryan et al. 2014] Bryan, B. A., King, D., and Zhao, G. (2014). Influence of management and environment on australian wheat: information for sustainable intensification and closing yield gaps. *Environmental Research Letters*, 9:1–12.

- [Chang et al. 2017] Chang, W., Cheng, J., Allaire, J., Xie, Y., and McPherson, J. (2017). *shiny: Web Application Framework for R. R package version 1.0.0*. Disponível em: <https://CRAN.R-project.org/package=shiny>. Acesso em: 15 fev. 2017.
- [Das et al. 2010] Das, S., Sismanis, Y., Beyer, K. S., Gemulla, R., Haas, P. J., and McPherson, J. (2010). Ricardo: Integrating R and hadoop. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 987–998, New York, NY, USA. ACM.
- [Eddelbuettel and Sanderson 2014] Eddelbuettel, D. and Sanderson, C. (2014). Rcpparmadillo: Accelerating R with high-performance C++ linear algebra. *Computational Statistics & Data Analysis*, 71:1054 – 1063.
- [Eugster et al. 2011] Eugster, M., Knaus, J., Porzelius, C., Schmidberger, M., and Vicedo, E. (2011). Hands-on tutorial for parallel computing with R. *Computational Statistics*, 26(2):219–239.
- [Gentleman and et al 2004] Gentleman, R. C. and et al (2004). Bioconductor: open software development for computational biology and bioinformatics. *Genome Biology*, 5(10):80. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/15461798>. Acesso em: 10 nov. 2016.
- [Grose 2008] Grose, D. J. (2008). Distributed computing using the multir package. In *Abstracts of The R User Conference 2008*, UseR! 2008, Dortmund, Germany. Disponível em: <https://www.statistik.uni-dortmund.de/useR-2008/abstracts/Grose.pdf>. Acesso em: 15 fev. 2017.
- [Hadoop 2013] Hadoop (2013). The apache hadoop. Disponível em: <http://hadoop.apache.org/>. Acesso em: 15 fev. 2017.
- [Jiang et al. 2012] Jiang, L., Patel, P. B., Ostrouchov, G., and Jamitzky, F. (2012). Openmp-style parallelism in data-centered multicore computing with R. In *Proceedings of the 17th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '12, pages 335–336, New York, NY, USA. ACM.
- [Kumar et al. 2011] Kumar, P., Ozisikyilmaz, B., Liao, W.-K., Memik, G., and Choudhary, A. (2011). High performance data mining using R on heterogeneous platforms. In *Proceedings of the 2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum*, IPDPSW '11, pages 1720–1729, Washington, DC, USA. IEEE Computer Society. Disponível em: <http://dx.doi.org/10.1109/IPDPS.2011.329>. Acesso em: 10 nov. 2016.
- [Mazzonetto 2016] Mazzonetto, A. (2016). Uma abordagem para execução paralela de modelos de simulação. Master's thesis, Programa de Pós-Graduação em Computação Aplicada, Universidade de Passo Fundo. Disponível em: <https://secure.upf.br/pdf/2016AngelaMazzonetto.pdf>. Acesso em: 15 fev. 2017.
- [Mitchell et al. 2011] Mitchell, L., Sloan, T. M., Mewissen, M., Ghazal, P., Forster, T., Piotrowski, M., and Trew, A. S. (2011). A parallel random forest classifier for R.

- In *Proceedings of the Second International Workshop on Emerging Computational Methods for the Life Sciences*, ECMLS '11, pages 1–6, New York, NY, USA. ACM.
- [Patel et al. 2012] Patel, I., Rau-Chaplin, A., and Varghese, B. (2012). A platform for parallel R-based analytics on cloud infrastructure. In *Proceedings of the 2012 41st International Conference on Parallel Processing Workshops*, ICPPW '12, pages 188–193, Washington, DC, USA. IEEE Computer Society.
- [Petrou et al. 2011] Petrou, S., Sloan, T. M., Mewissen, M., Forster, T., Piotrowski, M., Dobrzelecki, B., Ghazal, P., Trew, A., and Hill, J. (2011). Optimization of a parallel permutation testing function for the SPRINT R package. *Concurrency and computation : practice & experience*, 23(17):2258–2268.
- [Pukacki et al. 2006] Pukacki, J. et al. (2006). Programming grid applications with gridge. *Computational Methods in Science and Technology*, 12:47–68.
- [R Core Team 2016] R Core Team (2016). *R: a Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. Disponível em: <https://www.R-project.org/>. Acesso em: 15 fev. 2017.
- [Rickert 2014] Rickert, J. (2014). Big data analysis with revolution R enterprise. *Revolution White Paper*.
- [Rodrigues 2014] Rodrigues, F. A.; Traviesco, G. (2014). Avaliação do desempenho de rotinas de diagonalização do pacote “scalapack” em um cluster de computadores. *IFSC - Instituto de Física de São Carlos. USP*. Disponível em: https://www.icmc.usp.br/~francisco/works/scalapack/_analise.pdf. Acesso em: 15 fev. 2017.
- [Samatova et al. 2006] Samatova, N. F. et al. (2006). High performance statistical computing with parallel R: applications to biology and climate modelling. *Journal of Physics: Conference Series*, 46:505–509.
- [Schmidberger et al. 2009] Schmidberger, M. et al. (2009). State of the art in parallel computing with R. *Journal of Statistical Software*, 31:1–27.
- [Stokely et al. 2011] Stokely, M., Rohani, F., and Tassone, E. (2011). Large-scale parallel statistical forecasting computations in R. *JSM Proceedings*.
- [Torgo 2009] Torgo, L. (2009). Introdução à programação em R. Disponível em <https://cran.r-project.org/doc/contrib/Torgo-ProgrammingIntro.pdf>. Acesso em: 10 nov. 2016.
- [Vera and Suppi 2010] Vera, G. and Suppi, R. (2010). Integration of heterogeneous and non-dedicated environments for R. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pages 667–672.