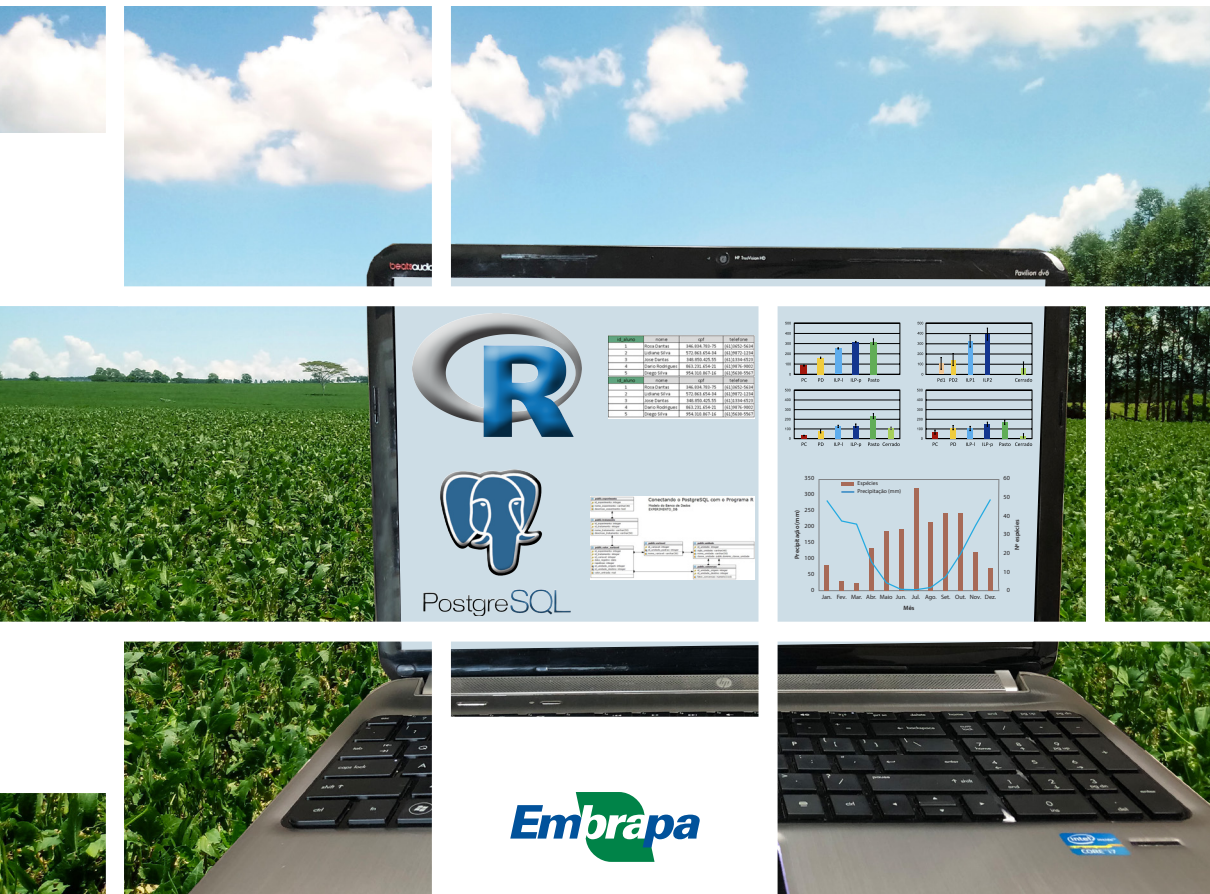
DOCUMENTOS
370

ISSN 1517-5111
ISSN on-line 2176-5081
Fevereiro/2021

Organização de dados de pesquisa no PostgreSQL e realização de análise estatística em ambiente R

Abordagem prática



ISSN 1517-5111
ISSN on-line 2176-5081
Fevereiro/2021

Empresa Brasileira de Pesquisa Agropecuária
Centro de Pesquisa Agropecuária dos Cerrados
Ministério da Agricultura, Pecuária e Abastecimento

DOCUMENTOS 370

Organização de dados de pesquisa no PostgreSQL e realização de análise estatística em ambiente R

Abordagem prática

Ozanival Dario Dantas da Silva
Juaci Vitória Malaquias

Embrapa Cerrados
Planaltina, DF
2021

Exemplar desta publicação disponível gratuitamente no link: https://www.bdpa.cnptia.embrapa.br

Embrapa Cerrados
BR 020, Km 18, Rod. Brasília / Fortaleza
Caixa Postal 08223
CEP 73310-970, Planaltina, DF
Fone: (61) 3388-9898
embrapa.br/cerrados
embrapa.br/fale-conosco/sac

Comitê Local de Publicações da Unidade

Presidente
Lineu Neiva Rodrigues

Secretária-executiva
Alexsandra Duarte de Oliveira

Secretária
Alessandra Silva Gelape Faleiro

Membros
Alessandra Silva Gelape Faleiro;
Alexandre Specht; Edson Eyji Sano;
Fábio Gelape Faleiro; Gustavo José Braga;
Jussara Flores de Oliveira Arbues;
Kleberson Worslley Souza;
Maria Madalena Rinaldi;
Shirley da Luz Soares Araujo

Supervisão editorial e revisão de texto
Jussara Flores de Oliveira Arbues

Normalização bibliográfica
Shirley da Luz Soares Araújo

Projeto gráfico da coleção
Carlos Eduardo Felice Barbeiro

Editoração eletrônica e
tratamento das ilustrações
Wellington Cavalcanti

Foto da capa
Luiz Adriano Maia Cordeiro

Impressão e acabamento
Alexandre Moreira Veloso

1ª edição
1ª impressão (2021): tiragem (30 exemplares)

Todos os direitos reservados.
A reprodução não autorizada desta publicação, no todo ou em parte,
constitui violação dos direitos autorais (Lei nº 9.610).

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Cerrados

S586o Silva, Ozanival Dario Dantas da.

Organização de dados de pesquisa no PostgreSQL e realização de análise estatística em ambiente R: abordagem prática. / Ozanival Dario Dantas da Silva e Juaci Vitória Malaquias. – Planaltina, DF : Embrapa Cerrados, 2021.

81 p. (Documentos / Embrapa Cerrados, ISSN 1517-5111, ISSN online 2176-5081, 370).

1. Banco de dados. 2. Modelagem. 3. Software livre. I. Malaquias, Juaci Vitória. II. Título. III. Série.

CDD (21 ed.) 001.6

Shirley da Luz Soares Araújo (CRB 1/1948) © Embrapa, 2021

Autores

Ozanival Dario Dantas da Silva

Cientista da Computação, mestre em Engenharia Elétrica e de Computação, analista da Embrapa Cerrados, Planaltina, DF

Juaci Vitória Malaquias

Estatístico, mestre em Ciência de Materiais em Modelagem e Simulação Computacional, analista da Embrapa Cerrados, Planaltina, DF

Apresentação

Com grande satisfação, apresentamos o documento intitulado *Organização de Dados de Pesquisa no PostgreSQL e Realização de Análise Estatística em Ambiente R: abordagem prática*, que trata da preparação, da modelagem e da organização de um banco de dados e da conexão do PostgreSQL ao software livre R, a fim de facilitar o processo da aplicação de análises estatísticas, sem a necessidade de uso frequente de planilhas eletrônicas.

Normalmente, durante o decorrer do processo de uma pesquisa de campo – após o planejamento, o delineamento e sua aplicação, propriamente dita –, os dados provenientes do estudo são registrados num formulário e, em seguida, digitados em um meio digital, como por exemplo, uma planilha eletrônica, a fim de serem, posteriormente, analisados estatisticamente e interpretados, respondendo as hipóteses estabelecidas no seu planejamento. Esse procedimento é mais comum do que se imagina, principalmente quando se trata de pesquisas no meio agropecuário. O que é pouco relatado são os percalços enfrentados para manter os dados guardados e organizados, num formato padrão que possa, de tempos em tempos, serem recuperados e devidamente atualizados, espacialmente ou temporalmente.

Talvez pela falta de conhecimento técnico ou pela familiaridade com ferramentas já disponíveis, adequadas para o melhor aproveitamento em termos de tempo, agilidade, segurança e eficiência no processo de organização, registro e consulta de dados, a maioria dos pesquisadores ainda se utilizem da construção de tabelas em planilhas eletrônicas para guardar dados gerados.

A Embrapa Cerrados é um centro de pesquisa ecorregional, que atua em diferentes campos da pesquisa agropecuária, dentro dos temas que envolvem

a conservação dos recursos naturais, estudos nas áreas vegetal e animal e tem potencial de gerar uma gama infinita de dados diariamente. Por esse motivo, o uso de banco de dados para o procedimento de registro e consulta se torna indispensável.

Nesse contexto, o presente documento pretende oferecer uma alternativa com instruções práticas que podem auxiliar o pesquisador na sua tarefa diária de registro e consulta de dados, de maneira segura e mais eficiente, associada à oportunidade de realizar futuramente análises estatísticas diretamente do banco de dados PostgreSQL, via o programa livre R.

Espera-se que as contribuições propostas neste manuscrito possam ser utilizadas para agilizar o processo de geração de informações, que orientarão a tomada de decisões, subsidiando a implementação de políticas públicas e promovendo o avanço do conhecimento.

Sebastião Pedro da Silva Neto
Chefe-Geral da Embrapa Cerrados

Sumário

Introdução

É comum pesquisadores utilizarem planilhas eletrônicas para armazenar dados de experimentos e, quando necessitam utilizar o ambiente R para realizar estatísticas, depararem-se com uma tarefa operacional bastante trabalhosa, que é obter e preparar os dados necessários para a importação no programa R para começar a gerar as estatísticas necessárias. O problema aumenta quando os critérios, que levaram à obtenção e à preparação dos dados para uso no programa R, mudam. Neste momento, é preciso voltar ao início do processo, às planilhas, montar uma outra planilha com as colunas necessárias, preparar os dados, salvar no formato que o programa R compreende e recomeçar a fazer as estatísticas. Seria mais prático, seguro e eficiente poder gravar seus dados em um banco de dados, conectá-lo ao programa R e com o uso de uma função de consulta ao banco de dados, informar os critérios para filtrar os dados necessários e disponibilizá-los dentro do programa R para a realização de estatísticas.

Nesta publicação, mostra-se como criar um banco de dados de pesquisa para atender a experimentos com delineamento em blocos casualizados (DBC), inserir os dados, conectar o programa R ao banco de dados, consultar os dados desejados, disponibilizá-los no programa R e, finalmente, realizar a análise estatística com maior flexibilidade e segurança. É utilizado um caso de uso hipotético para exemplificar todo o processo. Com isso, são apresentadas as vantagens de armazenar e gerenciar os dados e a facilidade de sua utilização no programa R.

Caso de uso – experimento cana-de-açúcar

Com o objetivo de facilitar o entendimento e mostrar o passo a passo da organização dos dados e sua análise, foi criada uma descrição sobre um experimento hipotético de pesquisa de cana-de-açúcar.

Em um experimento sobre a aplicação de amadurecedores em cana-de-açúcar, conduzido pela Embrapa Cerrados, foi realizado um estudo com o delineamento em blocos casualizados (DBC), estruturados em 4 blocos com

os tratamentos: T1-Testemunha; T2-Polaris (dose 2 mL/ha); T3-Polaris (dose 4 mL/ha); T4-Polaris (dose 6m L/ha); T5-Ethrel (dosagem 3 mL/ha); T6-Ethrel (dosagem 5 mL/ha); T7-Ethrel (dosagem 7 mL/ha). Para cada parcela, foram coletadas amostras de cana após a aplicação dos amadurecedores.

Os dados de produtividade, medidos em quilos por hectare, são apresentados na Tabela 1. Na Figura 1, é apresentado o croqui que representa o desenho do delineamento desse experimento no campo.

Tabela 1. Dados de produtividade do experimento.

Maturador	Bloco			
	1	2	3	4
T1-Testemunha	1.770	1.695	1.860	1.705
T2-Polaris (dose 2 mL/ha)	1.895	1.865	1.970	2.020
T3-Polaris (dose 4mL/ha)	1.683	1.680	1.725	1.940
T4-Polaris (dose 6 mL/ha)	1.663	1.770	1.752	1.726
T5-Ethrel (dosagem 3 mL/ha	1.416	1.356	1.488	1.364
T6-Ethrel (dosagem 5 mL/ha	1.516	1.492	1.576	1.616
T7-Ethrel (dosagem 7 mL/ha	1.346	1.344	1.380	1.552

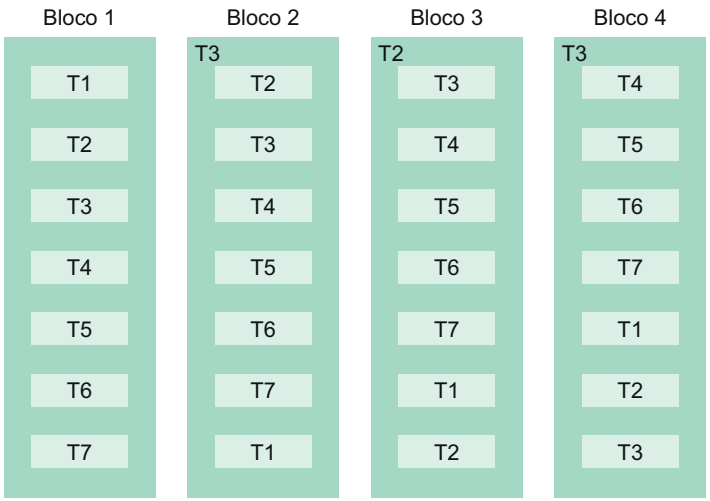


Figura 1. Delineamento em DBC.

Modelo de banco de dados para armazenar dados de experimentos de pesquisa

Neste tópico, será mostrado um Modelo de Banco de Dados Relacional (MBDR), que representa as tabelas do banco de dados ***Experimento_DB***, criado no Sistema Gerenciador de Banco de Dados (SGBD) *PostgreSQL* para armazenar dados de experimentos de pesquisa. Para possibilitar a agilidade e sem afetar o entendimento, foi abstraído apenas o essencial para a confecção do MBDR: a identificação do experimento, os tratamentos e as variáveis de análise.

Entendendo o modelo de banco de dados relacional

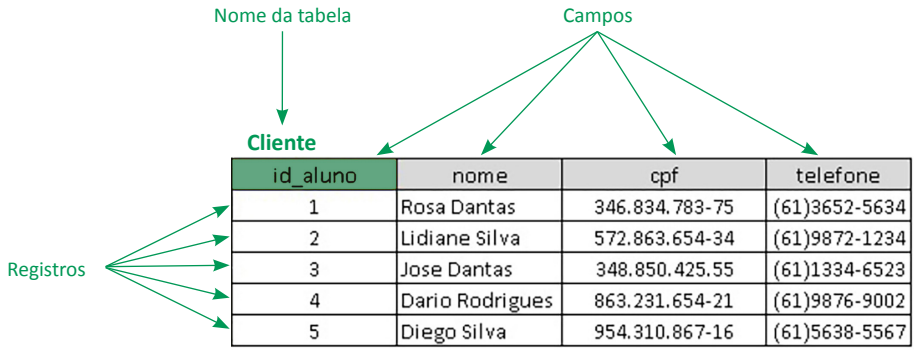
Neste documento, a intenção não é ensinar a modelar um banco de dados com todas as regras de normalização de dados e lógica relacional, o que pode ser consultado em (Silberschatz et al., 1999). Será mostrado o conhecimento necessário para entender o MBDR utilizado e, para tanto, alguns conceitos básicos serão apresentados cujos exemplos são ilustrados na Figura 2:

- **Campo**: atributo (característica) de um objeto da vida real. Por exemplo, o número do CPF 346.834.783-75 é um atributo da cliente Rosa Dantas.
- **Registro**: nome dado ao conjunto de campos de uma tabela e equivale a uma linha da tabela.
- **Tabela**: conjunto de registros e cada registro representa uma instância de uma entidade ou objeto. Por exemplo, “Rosa Dantas” é uma instância da entidade Cliente, enquanto, CPF e telefone são seus atributos ou campos.

Ao analisar um MBDR (Figura 3), deve-se observar, em cada tabela, a chave primária (PK)[1]; a chave estrangeira (FK)[2]; seus campos de dados; e seus relacionamentos com outras tabelas do modelo.

[1] Do inglês Primary Key. É um campo ou conjunto de campos em que seus valores formam uma sequência única de dados que identifica uma única linha (registro) na tabela.

[2] Do inglês Foreign Key. É um campo ou conjunto de campos que formam uma PK em sua tabela de origem. Uma FK pode identificar mais de uma linha na tabela de destino.



id_aluno	nome	cpf	telefone
1	Rosa Dantas	346.834.783-75	(61)3652-5634
2	Lidiane Silva	572.863.654-34	(61)9872-1234
3	Jose Dantas	348.850.425.55	(61)1334-6523
4	Dario Rodrigues	863.231.654-21	(61)9876-9002
5	Diego Silva	954.310.867-16	(61)5638-5567

Figura 2. Esquema de uma tabela.

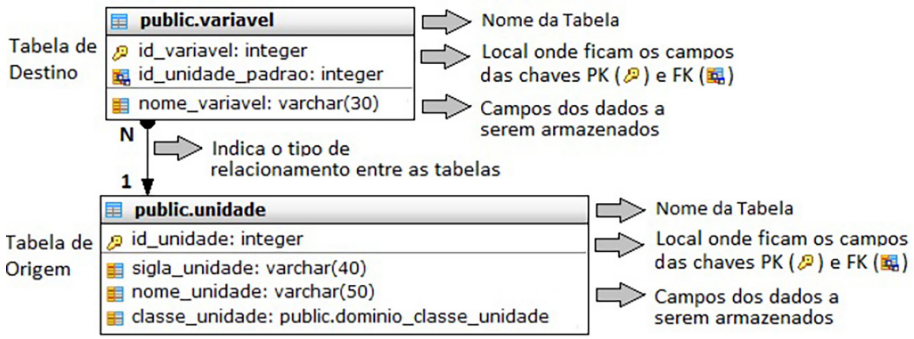


Figura 3. Entendendo o MBD.

Na Figura 3, observa-se que cada tabela é representada graficamente por um retângulo dividido em três partes: o nome da tabela, os campos chaves e os campos de dados. Além disso, foi utilizada uma nomenclatura que mostra a linha que liga os retângulos com uma seta em uma das pontas e um meio círculo na outra ponta para representar o relacionamento (1xN, leia-se 1 para N) entre as tabelas. A tabela que é conectada ao meio círculo (lado N) é a tabela de destino, assim, ela recebe a chave primária da outra tabela, a de origem, que está conectada à seta (lado do número 1). Dessa forma, pode-se perceber que o campo *Id_unidade* é a chave primária (PK) da tabela **Unidade** e que este campo foi incluído na tabela **Variável**, como chave estrangeira

(FK), mudando seu nome de *id_unidade* para *Id_unidade_padrão*. Isso possibilita identificar, nas realizações de cálculos e resultados, qual a unidade de medida uma determinada variável utiliza como sua unidade padrão. Para saber qual é a descrição da unidade padrão utilizada, é só consultar, na tabela **Unidade**, qual é o *Id_unidade* que possui o valor encontrado no campo *Id_unidade_padrao*, na tabela **Variável** (Tabelas 2 e 3).

Tabela 2. Lista de Variáveis.

Variável		
id_variavel	nome_variavel	**id_unidade_padrao**
1	Produtividade	106
2	Brix	2

Tabela 3. Unidades de medida.

Unidade			
id_unidade	sigla_unidade	nome_unidade	classe_unidade
2	%	Percentual	6
23	kg/ha	Quilos por hectare	9
106	sc/ha	Sacas por hectare	9

Por definição, em um modelo relacional, cada tabela possui uma chave primária (PK). Na Figura 3, deve-se observar o desenho de uma chave amarela na frente do(s) campo(s) que a compõe(m). Uma PK pode ser definida com apenas um campo, nesse caso, é chamada de *PK simples* ou com vários campos, chamada de *PK composta*. A chave primária tem como função encontrar apenas um registro dentro de sua tabela. (Silberschatz et al., 1999). Por exemplo, no MBDR do **Experimento_DB** (Figura 4), o valor 1 da PK simples, definida com apenas o campo *id_experimento* na tabela **Experimento**, só pode estar relacionada a uma única linha na tabela. Ou seja, o valor 1 não pode se repetir e o SGBD cuida de evitar que isso aconteça, gerando um aviso de erro, caso o usuário tente gravar mais de um registro com o mesmo valor para o campo da chave primária simples. Isso também ocorre quando

a PK é do tipo composta, que é o caso existente na tabela **Tratamento**. Note que os campos *id_experimento* e *id_tratamento* foram utilizados para definir a PK desta tabela, formando uma PK composta. Nesse caso, o conjunto dos valores formados por todos os campos da PK não pode se repetir. Por exemplo, pode-se ter o valor 1 para identificar o experimento e o valor 1 para identificar o tratamento. Este par de valores (1,1) não pode mais se repetir, pois já está relacionado a uma linha na tabela. Porém, pode-se ter outros valores como (1, 2); (1, 3); (1,N) ou (2, 1); (2, 2); (2,N); e assim por diante, sem nunca repetir o conjunto dos valores da chave primária composta, já cadastrada no banco de dados.

Por fim, cada tabela pode ter campos de dados a serem armazenados, esses campos só aparecem uma única vez no MDB e estão intimamente ligados à tabela à qual pertencem. Ou seja, o nome da unidade só vai existir na tabela de **Unidade**. Isso evita inconsistência nos dados, pois a entrada para o nome de unidade só ocorre em um único lugar, na tabela de **Unidade**, no campo *nome_unidade*. Note também que cada campo na tabela possui um tipo de dado (Tabela 4).

Tabela 4. Tipos de dados usados no banco de dados *Experimento_DB*.

Tipo de dado	Descrição
interger	Usado para armazenar dados inteiros com valores no intervalo de -2147483648 a +2147483647
varchar(x)	Armazena cadeia de caracteres de comprimento fixo, definido no valor indicado em x

O *SGBD PostgreSQL* possui um leque bastante completo de tipos de dados que podem ser estudados em seu site oficial (EDB - EnterpriseDB - Tipos de Dados, 2020), mas permite a criação de outros tipos de dados. Na tabela Unidade (Figura 3) o tipo **Domínio** é utilizado para definir o campo *classe_unidade*, que consiste em criar um conjunto de dados possíveis para seu armazenamento. Assim, o domínio classe unidade foi definido conforme os valores apresentados na Tabela 5. Note que, por exemplo, quando é gravado o valor 9, no campo *classe_unidade*, interpreta-se que a unidade em questão pertence à classe de unidades de *produção* e assim por diante.

Tabela 5. Domínio de dados para classe unidade.

Classe unidade	Descricao_unidade
1	Distância
2	Área
3	Volume
4	Massa
5	Concentração
6	Taxa
7	Velocidade
8	Tempo
9	Produção
10	Força
11	Temperatura

Modelo do banco de dados: *experimento_DB*

Na Figura 4, mostra-se um modelo para armazenar dados de pesquisa, que servirá para o registro dos dados necessários para o estudo de caso sobre um experimento de cana-de-açúcar apresentado na página 9.

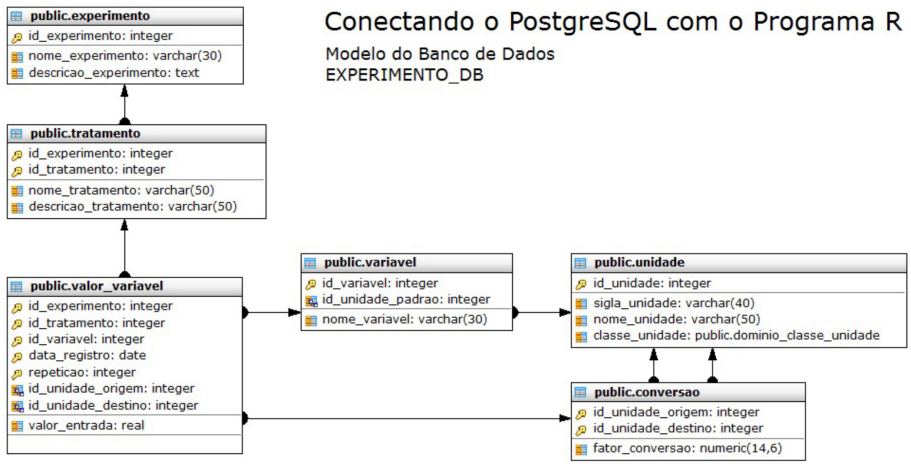


Figura 4. MBD Experimento_DB.

Note que existem 6 tabelas (Figura 4):

1) Tabela **Experimento**: responsável por armazenar os dados sobre os experimentos. O campo *id_experimento* é a chave primária (PK simples) e *nome_experimento* e *descrição_experimento* são os campos que irão armazenar os dados do experimento. Nessa tabela, poder-se-ia armazenar mais dados sobre o experimento, por exemplo, data de início e data fim. Mas, neste estudo, será armazenado apenas os dados da Tabela 6.

Tabela 6. Lista de experimentos.

Experimento		
id_experimento	**nome_experimento**	**descricao_experimento**
1	Cana 375-BR	Experimento da Usina PlanCana
2	Trigo BRS-27	Experimento de Trigo BRS-27

2) Tabela **Tratamento**: cada experimento, normalmente, tem vários tratamentos, surgindo um relacionamento entre a tabela de **Experimento** e a de **Tratamento** do tipo (1xN). Assim, a tabela **Tratamento** recebe a PK de **Experimento**, ou seja, *id_experimento* é uma chave estrangeira (FK) e também faz parte da chave primária (PK) na tabela **Tratamento**. Dessa forma, tem-se *id_experimento e id_tratamento* como PK composta da tabela **Tratamento**. Note que apenas o conjunto dos valores da chave primária é capaz de identificar uma única linha na tabela. Além disso, a hierarquia dos campos da PK permite agrupar os dados por experimento. Os campos de dados são: *nome_tratamento e descrição_tratamento* (Tabela 7).

3) Tabela **Unidade**: cadastro de todas as unidades de medidas que serão utilizadas nos experimentos, em que: *id_unidade* é o campo PK simples; *sigla_unidade*, *nome_unidade* e *classe_unidade* são os campos que irão armazenar os dados de cada unidade de medida. Embora os valores da PK não sejam uma sequência unitária não há problema, pois o que importa é que cada valor seja positivo e único. O campo

classe_unidade armazena os valores 6 e 9 que são, respectivamente, as classes de unidades *taxa* e *produção*, conforme Tabela 5.

Tabela 7. Dados de tratamentos.

Tratamento			
id_experimento	**id_tratamento**	**nome_tratamento**	**descricao_ tratamento**
1	1	Testemunha (dose 1 mL/ha)	Testemunha
1	2	Polaris (dose 2 mL/ha)	Tratamento 1
1	3	Polaris (dose 4 mL/ha)	Tratamento 2
1	4	Polaris (dose 6 mL/ha)	Tratamento 3
1	5	Ethrel (dosagem 3 mL/ha	Tratamento 4
1	6	Ethrel (dosagem 5 mL/ha	Tratamento 5
1	7	Ethrel (dosagem 7 mL/ha	Tratamento 6

4) Tabela **Conversão**: possui os fatores necessários para converter um valor que está em uma determinada unidade de medida para outra unidade. A chave primária PK é do tipo composta, contendo os campos *id_unidade_origem* e *id_unidade_destino*. Por exemplo, para converter um valor de entrada da unidade 23 (kg/ha) para a unidade 106 (sc/ha), deve-se multiplicar o valor que está na unidade 23 pelo fator de conversão igual a 0.016666667, conforme Tabela 8.

Tabela 8. Fatores de conversão de unidades.

Conversao		
id_unidade_origem	**id_unidade_destino**	**fator_conversao**
23	106	0.016666667
23	23	1
106	106	1
2	2	1

5) Tabela **Variavel**: nesta tabela, é armazenada a lista de variáveis necessárias para acompanhar os experimentos, em que *id_variavel* é a chave primária da tabela; *id_unidade_padrao* é uma chave estrangeira FK vinda da tabela **Unidade**, que informa a escolha de unidade de medida para uma determinada variável. Por exemplo, para a variável *Produtividade* a unidade padrão é 106, note que a descrição da variável está na tabela **Unidade** e é acessada por meio de sua PK com valor igual a 106 (Tabela 3). Na Tabela 2, encontram-se as variáveis utilizadas neste experimento.

6) Tabela **Valor_Variavel**: armazenam-se os valores das variáveis de cada experimento, tratamento e repetição, nas datas em que ocorrem suas leituras, por isso a chave primária é composta por vários campos para organizar os dados de forma a otimizar o acesso (Tabela 9).

Tabela 9. Exemplo de Dados do experimento.

Valor_Variável							
id_ experimento	id_ tratamento	id_variavel	data_registro	repeticao	valor_ entrada	id_unidade_ origem	id_unidade_ destino
1	1	2	2020/04/18	1	1770.00	23	106
1	1	2	2020/04/19	2	1695.00	23	106
1	1	2	2020/04/20	3	1860.00	23	106
1	1	2	2020/04/21	4	1705.00	23	106
1	2	2	2020/04/22	1	1895.00	23	106
1	2	2	2020/04/23	2	1865.00	23	106
1	2	2	2020/04/24	3	1970.00	23	106
1	2	2	2020/04/25	4	2020.00	23	106
1	3	2	2020/04/26	1	1683.00	23	106
1	3	2	2020/04/27	2	1680.00	23	106
1	3	2	2020/04/28	3	1725.00	23	106
1	3	2	2020/04/29	4	1940.00	23	106
1	4	2	2020/04/22	1	1663.00	23	106
1	4	2	2020/04/23	2	1770.00	23	106
1	4	2	2020/04/24	3	1752.00	23	106
1	4	2	2020/04/25	4	1726.00	23	106
1	5	2	2020/04/26	1	1416.00	23	106
1	5	2	2020/04/27	2	1356.00	23	106
1	5	2	2020/04/28	3	1488.00	23	106

Continua...

Tabela 9. Continuação.

Valor_Variável							
id_ experimento	id_ tratamento	id_variavel	data_registro	repeticao	valor_ entrada	id_unidade_ origem	id_unidade_ destino
1	5	2	2020/04/29	4	1364.00	23	106
1	6	2	2020/04/22	1	1516.00	23	106
1	6	2	2020/04/23	2	1492.00	23	106
1	6	2	2020/04/24	3	1576.00	23	106
1	6	2	2020/04/25	4	1616.00	23	106
1	7	2	2020/04/26	1	1346.00	23	106
1	7	2	2020/04/27	2	1344.00	23	106
1	7	2	2020/04/28	3	1380.00	23	106
1	7	2	2020/04/29	4	1552.00	23	106
1	1	1	2020/04/18	1	17.70	2	2
1	1	1	2020/04/19	2	16.95	2	2
1	1	1	2020/04/20	3	18.60	2	2
1	1	1	2020/04/21	4	17.05	2	2
1	2	1	2020/04/22	1	18.95	2	2
1	2	1	2020/04/23	2	18.65	2	2
1	2	1	2020/04/24	3	19.70	2	2
1	2	1	2020/04/25	4	20.20	2	2
1	3	1	2020/04/26	1	16.83	2	2
1	3	1	2020/04/27	2	16.80	2	2

Continua...

Tabela 9. Continuação.

Valor_Variável							
id_ experimento	id_ tratamento	id_variavel	data_registro	repeticao	valor_ entrada	id_unidade_ origem	id_unidade_ destino
1	3	1	2020/04/28	3	17.25	2	2
1	3	1	2020/04/29	4	19.40	2	2
1	4	1	2020/04/22	1	16.63	2	2
1	4	1	2020/04/23	2	17.70	2	2
1	4	1	2020/04/24	3	17.52	2	2
1	4	1	2020/04/25	4	17.26	2	2
1	5	1	2020/04/26	1	14.16	2	2
1	5	1	2020/04/27	2	13.56	2	2
1	5	1	2020/04/28	3	14.88	2	2
1	5	1	2020/04/29	4	13.64	2	2
1	6	1	2020/04/22	1	15.16	2	2
1	6	1	2020/04/23	2	14.92	2	2
1	6	1	2020/04/24	3	15.76	2	2
1	6	1	2020/04/25	4	16.16	2	2
1	7	1	2020/04/26	1	13.46	2	2
1	7	1	2020/04/27	2	13.44	2	2
1	7	1	2020/04/28	3	13.80	2	2
1	7	1	2020/04/29	4	15.52	2	2

Preparando o ambiente de banco de dados

Neste tópico, será mostrado como instalar o *PostgreSQL*; criar o branco de dados com suas tabelas, com uma função para obter os dados armazenados; como popular os dados nas tabelas e; finalmente, utilizando o *Data Browser*, como consultar o banco de dados.

Instalando o PostgreSQL

Para instalar uma das versões do *PostgreSQL*, siga as instruções no Anexo 1. Outra opção é baixar o instalador acessando o sítio oficial Windows Installers e fazer a instalação utilizando o tutorial encontrado em (EDB, 2020c).

Criando o banco de dados: experimento_db

A primeira ação a ser feita após a instalação do *PostgreSQL* é criar o banco de dados **experimento_db**. Para isso, abra o *pgAdmin 4*, que foi instalado durante a instalação do *PostgreSQL*. Ao abri-lo será solicitada a senha de acesso do superusuário **postgres** (Figura 5). Digite a mesma senha criada na instalação do *PostgreSQL* (Anexo 1, Figura 7).

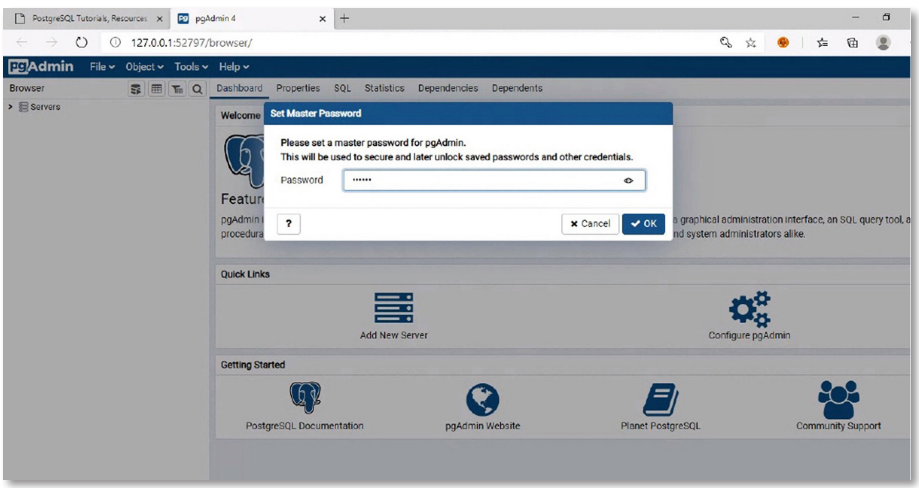


Figura 5. Tela de conexão do *pgAdmin 4*.

Na Figura 6, observa-se a tela inicial do *pgAdmin*, em que, à esquerda, há uma árvore de objetos do banco de dados *PostgreSQL*.

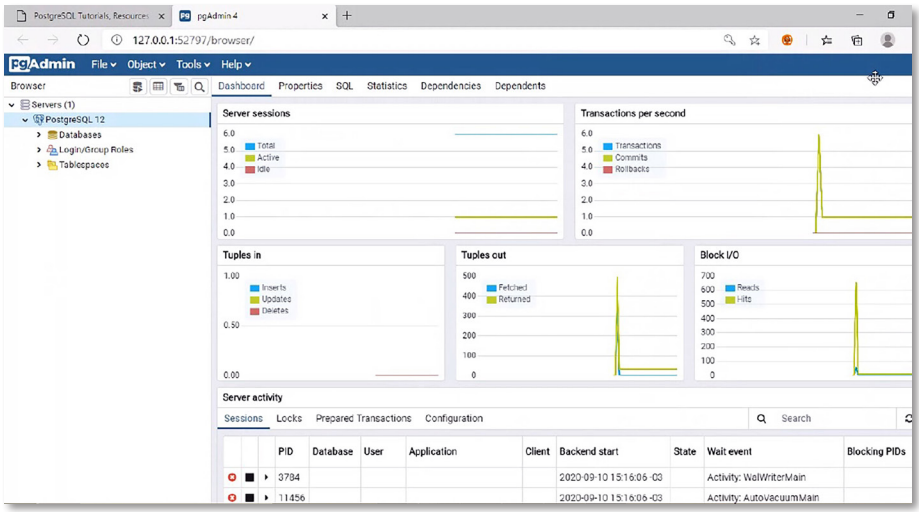


Figura 6. Tela inicial do *pgAdmin 4*.

Abra os itens da árvore, clicando na sequência **Servers**, **PostgreSQL**, **Databases**. Agora, clique com o botão direito do mouse sobre o item da árvore **Databases** (Figura 7) para abrir o menu de criação de um novo banco de dados.

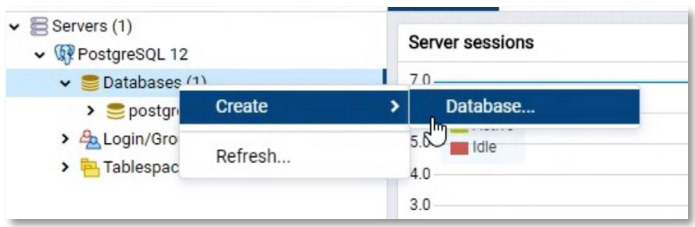


Figura 7. Menu de criação de um novo banco de dados.

Na tela inicial de criação do banco de dados (Figura 8), digite os dados solicitados conforme são apresentados: em **Database**, informe o nome do novo banco de dados; em **Owner**, entre com o usuário dono do novo banco de

dados (neste caso o usuário **postgres**); e em **Comment**, entre com o comentário sobre o novo banco de dados.

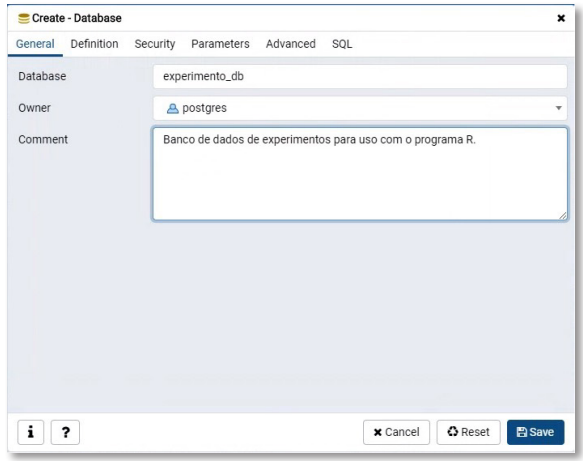


Figura 8. Criação do banco de dados – Aba *General*.

A seguir, entre com as seguintes configurações (Figura 9) para definir como será a acentuação (**Encoding**, **Collation**, **Character type**) e qual será o número máximo de conexões ao banco de dados (**Connection limit**). Se informado o valor **-1** para **Connection Limit**, significa que o banco de dados não terá limites de acessos simultâneos.

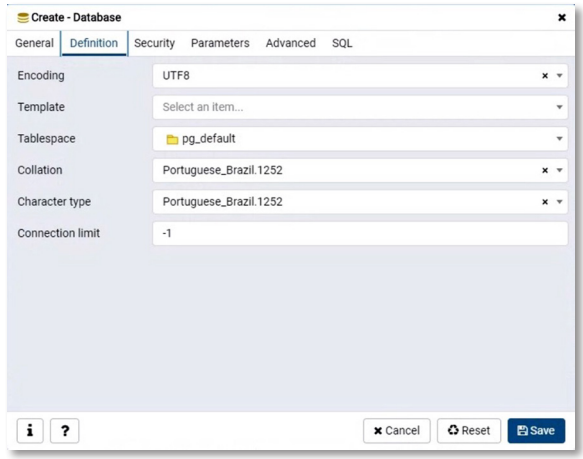


Figura 9. Criação do banco de dados – Aba *Definition*.

Após entrar com as configurações, acesse a aba SQL (Figura 10) para ver o comando em SQL[3] de criação do novo banco de dados **experimento_db**. Para finalizar e criar fisicamente o banco de dados no *SGBD PostgreSQL*, clique em **Save**.

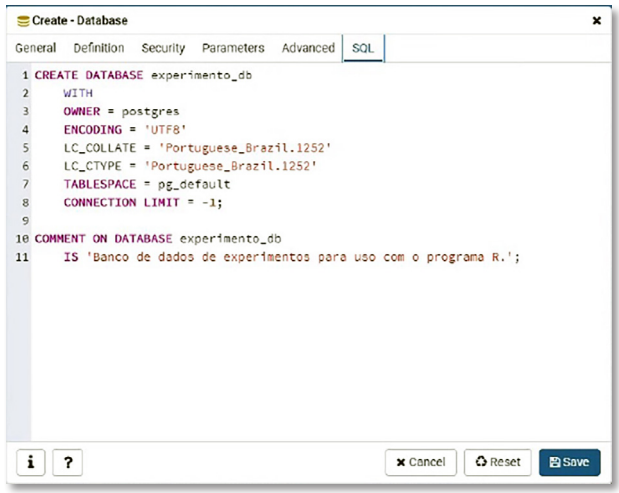


Figura 10. Criação do banco de dados – Aba *SQL*.

Criando as tabelas do banco de dados

Com o objetivo de criar as tabelas do banco de dados, inicialmente, abra o editor de comandos do *pgAdmin 4*, conforme Figura 11.

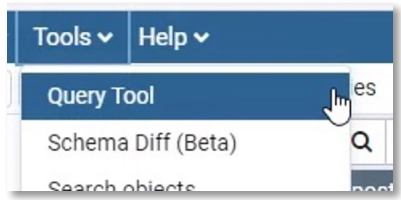


Figura 11. Menu de acesso a *Query Tool*.

[3] Do inglês: *Structured Query Language*, linguagem de consulta estruturada utilizada para "conversar" com o banco de dados.

No Anexo II , estão todas as instruções para a criação das tabelas do banco de dados **experimento_db**. Copie e cole, individualmente, cada conjunto de instruções na área de comandos da *Query Tool*, como exemplificado na Figura 12. Depois, execute as instruções clicando no ícone play. A mensagem "*Query returned sucessfully in 999 msec*" deve ser apresentada para cada conjunto de instruções SQL executado. Em *999*, será mostrado o tempo gasto para executar o conjunto de instruções.

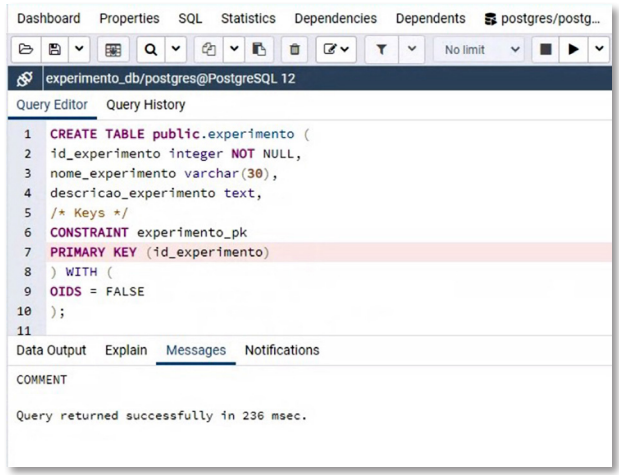


Figura 12. Query editor.

Ao final, todas as 6 tabelas do banco de dados *experimento_db* deverão constar na árvore de objetos do banco de dados, conforme Figura 13.

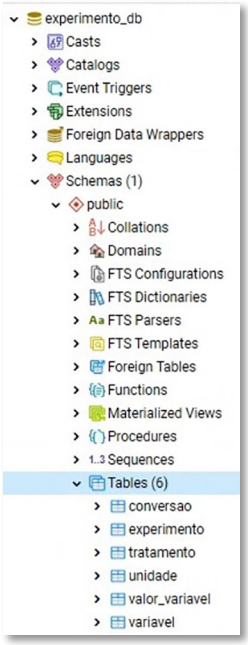


Figura 13. Árvore de objetos do banco de dados *experimento_db*.

Criando uma função para obter dados do experimento

Siga o mesmo procedimento utilizado na página 25 para executar as instruções SQL de criação de tabelas agora, com as instruções em SQL (Anexo II) para criar a função de obtenção de dados do banco de dados *experimento_db*, fn_obter_dados_experimento (Figura 14).

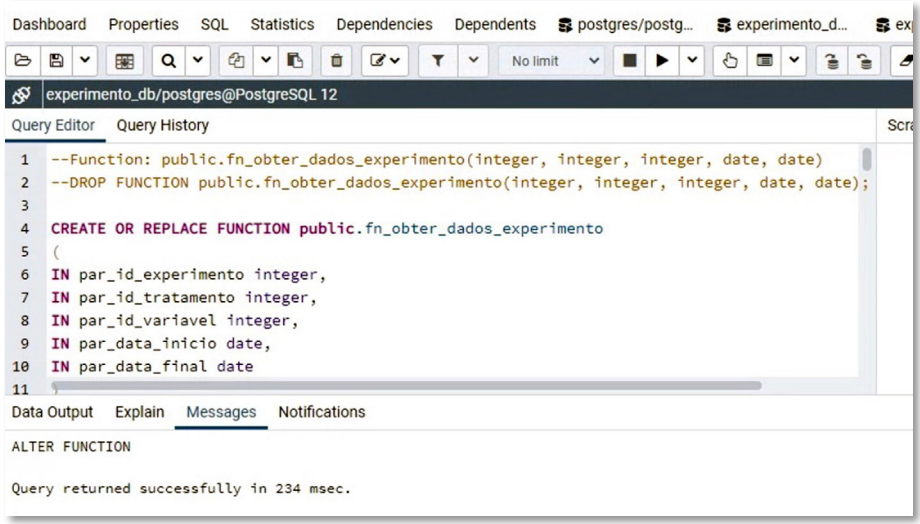


Figura 14. Executando script para criar a função.

Popular os dados nas tabelas do banco de dados

É possível popular[4] as tabelas do banco de dados **Experimento_DB com** os dados apresentados nesta publicação no caso de uso, página 9, seguindo o mesmo procedimento utilizado para executar as instruções de criação de tabelas; agora, as instruções em SQL serão as de inserção de dados que estão no Anexo III. Para agilizar o procedimento, copie e execute todo o script de uma só vez. Não mude a sequência das tabelas, pois isso poderá incorrer em erro. Ao final, será possível executar os comandos para verificar o conteúdo de cada tabela do banco de dados: *experimento_db*, como ilustrado na Figura 15. Execute um comando de cada vez, selecionando o comando e mandando executá-lo clicando no ícone play.

[4] Ato de inserir dados em uma tabela no banco de dados.

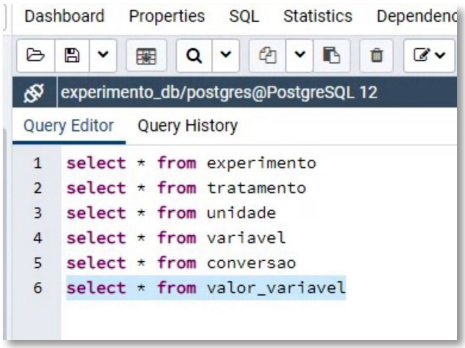


Figura 15. Consultando os dados das tabelas.

Consultando o banco de dados

Neste tópico, será apresentado um outro programa para consultar os dados no banco de dados e será mostrado como utilizar a função, criada para obter os dados para realizar a análise estatística. Os comandos em SQL mostrados aqui, podem ser executados também na linha de comando do programa R.

Utilizando o Database Browser para consultar as tabelas

Para facilitar o entendimento de como executar os comandos SQL e verificar suas ações no banco de dados, entre elas a consulta, será utilizado um outro software, o *Database Browser*. Assim, siga as instruções do Anexo IV para proceder sua instalação.

Ao abrir o programa pela primeira vez, é preciso criar uma conexão de banco com o *SGBD PostgreSQL* no menu Connection selecione *Add* (Figura 16).

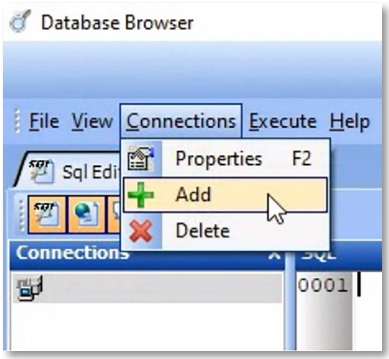


Figura 16. Criação de uma nova conexão de banco.

Na tela que se abre (Figura 17), devem ser preenchidos os campos conforme apresentados. A senha e a porta a serem utilizadas na conexão são as mesmas que foram criadas no momento da instalação do *PostgreSQL* (Figura 7, Anexo I).

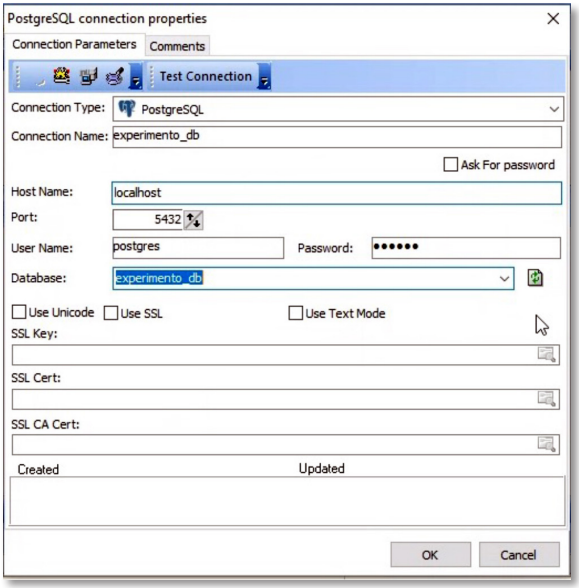


Figura 17. Configuração da conexão com o banco *Experimento_DB*.

É possível testar a conexão clicando em *Test Connection*, depois clique em **OK** para gravar a nova conexão. A conexão salva aparece como uma conexão criada (Figura 18); nela, pode ser observada também a lista de tabelas da conexão aberta, *Experimento_DB*, e, caso seja selecionada uma tabela com o mouse, será possível verificar seus dados e o comando executado para trazê-los.

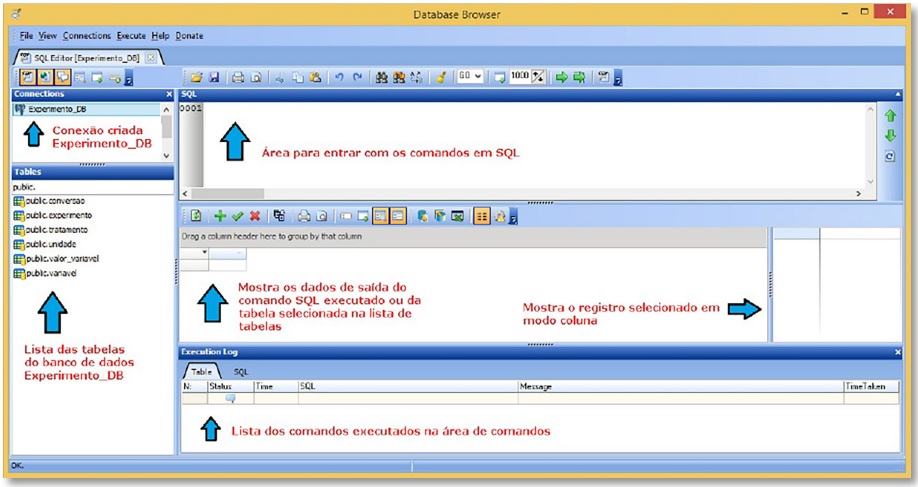


Figura 18. Esquema da tela do *Database Browser*.

Os comandos, que são digitados na área para entrada de comando (Figura 18), são executados conforme o botão escolhido (Tabela 10).

Tabela 10. Botões de execução SQL.

Botão	Descrição de uso
	Executa todos os comandos que estão na área de comandos, na sequência que aparecem
	Executa apenas o comando SQL selecionado com o mouse

Na Figura 19, observa-se uma lista de comandos que são executados individualmente, no exemplo, é possível observar um comando selecionado. Para executá-lo, foi pressionado o botão que executa um comando por vez (Tabela 10). O comando solicita para que o banco de dados mostre os dados da tabela *tratamento*. Para verificar os dados das outras tabelas selecione outro comando e execute um de cada vez.

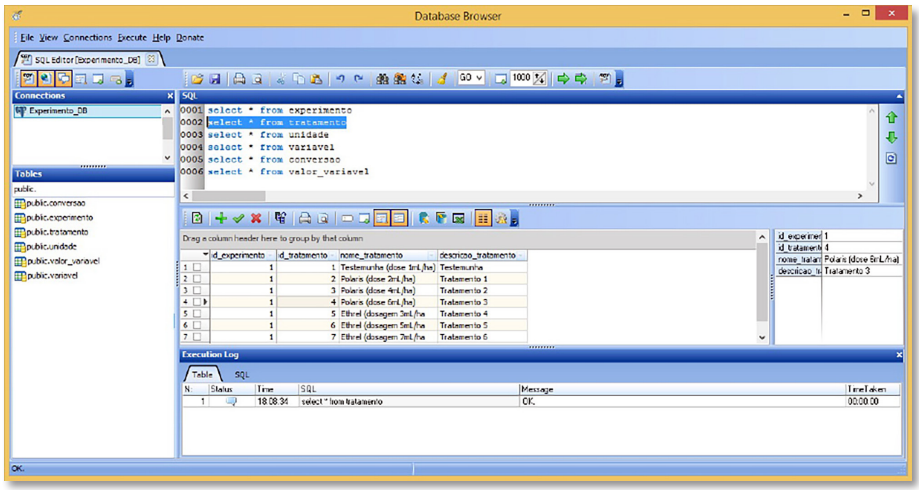


Figura 19. Executando comandos individuais.

Utilizando a função *fn_obter_dados_experimento* para obter os dados para análise

Esta função serve para recuperar os dados de um determinado experimento e será utilizada dentro do programa R. Veja sua sintaxe:

Sintaxe:

```
SELECT <campos>
  FROM fn_obter_dados_experimento(integer, integer, integer, date, date);
```

Em que:

<campos> é a lista dos campos relacionados para o retorno da função, separados por vírgula. São todos eles:

id_experimento, nome_experimento, id_tratamento.
nome_tratamento, repeticao ,
data_registro, id_variavel, nome_variavel, valor_entrada.
id_unidade_origem, unidade_origem, valor_destino_calc.
id_unidade_destino, unidade_destino, valor_padrao_calc,
id_unidade_padrao,unidade_padrao.

Ou apenas alguns desejados:

nome_experimento, nome_tratamento, repeticao ,
valor_padrao_calc, unidade_padrao.

(Integer, integer, integer, date, date) são os parâmetros de consulta (filtros) da função. Os 5 parâmetros são apresentados na sequência de uso:

1) par_id_experimento: informar um número inteiro que indique o identificador do experimento que deseja obter seus dados.
2) par_id_tratamento: informar um número inteiro que indique o identificador do tratamento do experimento indicado, se informar **null**, todos os tratamentos serão retornados na consulta.
3) par_id_variavel: informar um número inteiro que indique o identificador da variável desejada, se informar **null**, todas as variáveis serão retornados na consulta.
4) par_data_inicio: informar a data inicial do período de registro do dado que deseja consultar, a data deve estar no formato **aaaa/mm/dd** (ano/mês/dia).
5) par_data_final: informar a data final do período de registro do dado que deseja consultar, a data deve estar no formato **aaaa/mm/dd** (ano/mês/dia).

A seguir, observe como responder a questões de consulta ao banco de dados:

Como chamar a função para obter todos os campos?

Na área de entrada de comandos SQL (Figura 18), digite, selecione e execute o seguinte comando:

```
Select  id_experimento, nome_experimento, id_tratamento,
        nome_tratamento, repeticao,
        data_registro, id_variavel, nome_variavel,
        valor_entrada::NUMERIC(15, 2),
        id_unidade_origem, unidade_origem,
        valor_destino_calc::NUMERIC(15, 2),
        id_unidade_destino, unidade_destino, valor_padrao_calc,
        id_unidade_padrao, unidade_padrao
  from  fn_obter_dados_experimento(1, 1, 2, '2020/04/18'::date,
        '2020/04/28'::date)
```

Resultado

Na Figura 20, note que foram apresentados todos os campos relacionados no comando. Outra forma seria colocar um asterisco (*) no lugar da relação de todos os campos que a função retorna. Na Figura 20, o **Select** retornaria o mesmo resultado, porém o uso do asterisco no lugar da lista dos campos só é aconselhável quando número de usuários for pequeno, pois não definir a lista dos campos faz com que o *SGBD PostgreSQL* tenha que buscar a relação dos campos antes de executar o comando. Para uma quantidade alta de usuários, demandando consultas ao banco e não selecionando os campos de interesse, pode onerar o tempo necessário para realizar a consulta. Por isso é aconselhável, sempre, informar a lista dos campos que se deseja no comando de consulta.

```
Select   *

  from   fn_obter_dados_experimento(1, 1, 2, '2020/04/18'::date,
         '2020/04/28'::date)
```

id_experimento	nome_experimento	id_tratamento	nome_tratamento	repeticao	data_registro	id_variavel	nome_variavel	valor_entrada
1	Cana 375-BR	1	Testemunha (dose 1mL/ha)	4	21/04/2020	2	Produtividade	1705
1	Cana 375-BR	1	Testemunha (dose 1mL/ha)	3	20/04/2020	2	Produtividade	1860
1	Cana 375-BR	1	Testemunha (dose 1mL/ha)	2	19/04/2020	2	Produtividade	1695
1	Cana 375-BR	1	Testemunha (dose 1mL/ha)	1	18/04/2020	2	Produtividade	1770

id_unidade_origem	unidade_origem	id_unidade_destino	unidade_destino	valor_padrao_calc	id_unidade_padrao	unidade_padrao
23	Quilos por Hectáre	28.42	Sacas por Hectáre	1705	23	Quilos por Hectáre
23	Quilos por Hectáre	31	Sacas por Hectáre	1860	23	Quilos por Hectáre
23	Quilos por Hectáre	28.25	Sacas por Hectáre	1695	23	Quilos por Hectáre
23	Quilos por Hectáre	29.5	Sacas por Hectáre	1770	23	Quilos por Hectáre

Figura 20. Consulta com todos os campos.

Como chamar a função "selecionado" os campos que deseja obter?

Na área de entrada de comandos SQL (Figura 18), digite, selecione e execute o seguinte comando:

```
Select   nome_experimento, nome_tratamento, repeticao, valor_padrao_calc,
         unidade_padrao
  from   fn_obter_dados_experimento(1, 1, 2, '2020/04/18'::date,
         '2020/04/28'::date);
```

Resultado

Na Figura 21, apresenta-se apenas os campos informados no comando de consulta.

nome_experimento	nome_tratamento	repeticao	valor_padrao_calc	unidade_padrao
Cana 375-BR	Testemunha (dose 1mL/ha)	4	1705	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	3	1860	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	2	1695	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	1	1770	Quilos por Hectáre

Figura 21. Consulta com apenas alguns campos.

Como chamar a função passando “null” no parâmetro de escolha do tratamento?

Na área de entrada de comandos SQL (Figura 18), digite, selecione e execute o seguinte comando:

```
Select   nome_experimento, nome_tratamento, repeticao, valor_padrao_calc,
         unidade_padrao
  from   fn_obter_dados_experimento(1, null, 2, ‘2020/04/18’::date,
         ‘2020/04/28’::date);
```

Resultado

Como foi informado o “null“ para o parâmetro que indica qual tratamento se deseja no resultado da consulta, a função ignora este parâmetro e traz todos os tratamentos envolvidos (Figura 22).

nome_experimento	nome_tratamento	repeticao	valor_padrao_calc	unidade_padrao
Cana 375-BR	Testemunha (dose 1mL/ha)	1	1770	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	2	1695	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	3	1860	Quilos por Hectáre
Cana 375-BR	Testemunha (dose 1mL/ha)	4	1705	Quilos por Hectáre
Cana 375-BR	Polaris (dose 2mL/ha)	1	1895	Quilos por Hectáre
Cana 375-BR	Polaris (dose 2mL/ha)	2	1865	Quilos por Hectáre
Cana 375-BR	Polaris (dose 2mL/ha)	3	1970	Quilos por Hectáre
Cana 375-BR	Polaris (dose 2mL/ha)	4	2020	Quilos por Hectáre
Cana 375-BR	Polaris (dose 4mL/ha)	1	1683	Quilos por Hectáre
Cana 375-BR	Polaris (dose 4mL/ha)	2	1680	Quilos por Hectáre
Cana 375-BR	Polaris (dose 4mL/ha)	3	1725	Quilos por Hectáre
Cana 375-BR	Polaris (dose 6mL/ha)	1	1663	Quilos por Hectáre
Cana 375-BR	Polaris (dose 6mL/ha)	2	1770	Quilos por Hectáre
Cana 375-BR	Polaris (dose 6mL/ha)	3	1752	Quilos por Hectáre
Cana 375-BR	Polaris (dose 6mL/ha)	4	1726	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 3mL/ha	1	1416	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 3mL/ha	2	1356	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 3mL/ha	3	1488	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 5mL/ha	1	1516	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 5mL/ha	2	1492	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 5mL/ha	3	1576	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 5mL/ha	4	1616	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 7mL/ha	1	1346	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 7mL/ha	2	1344	Quilos por Hectáre
Cana 375-BR	Ethrel (dosagem 7mL/ha	3	1380	Quilos por Hectáre

Figura 22. Consulta com todos os tratamentos.

Conexão do PostgreSQL com o programa R

Com um banco de dados estruturado e com uma função de consulta aos dados, é possível informar os critérios para filtrar os dados necessários e disponibilizá-los dentro do programa R para a realização de estatísticas. Sendo assim, neste tópico, será mostrado como instalar o programa R e os pacotes necessários à conexão com o *PostgreSQL*; e criar a conexão do programa R com o banco **Experimento_db**.

Instalando o programa R

Para instalar o programa R, siga as instruções encontradas no Anexo V.

Instalando os pacotes

Para preparar o ambiente do programa R, a fim de estabelecer uma conexão de banco de dados com o *SGBD PostgreSQL*, é necessário instalar dois pacotes: "**RPostgreSQL**" e "**sqldf**". Deve-se ater à forma como é escrito o nome do pacote, letras maiúsculas e minúsculas fazem diferença, se alguma letra

for trocada o pacote não é encontrado para instalação. A seguir, apresenta-se o procedimento para instalar e disponibilizar para uso o primeiro pacote, esse mesmo procedimento deve ser realizado para instalar o outro pacote, assim como quaisquer outros pacotes de interesse específico.

A seguir é apresentada a sintaxe do comando no programa R para fazer a instalação de pacotes:

Sintaxe:

install.packages(“<pacote>”)

Em que:

<pacote> é o nome do pacote que se deseja instalar no programa R.

Segue o comando, que deve ser digitado na linha de comandos do programa R, para instalar o primeiro pacote.

>install.packages(“RPostgreSQL”)

Ao executar o comando de instalação de pacotes, o programa R solicita que o servidor deve ser utilizado para fazer a instalação. Então selecione um servidor e clique em **OK** (Figura 23).

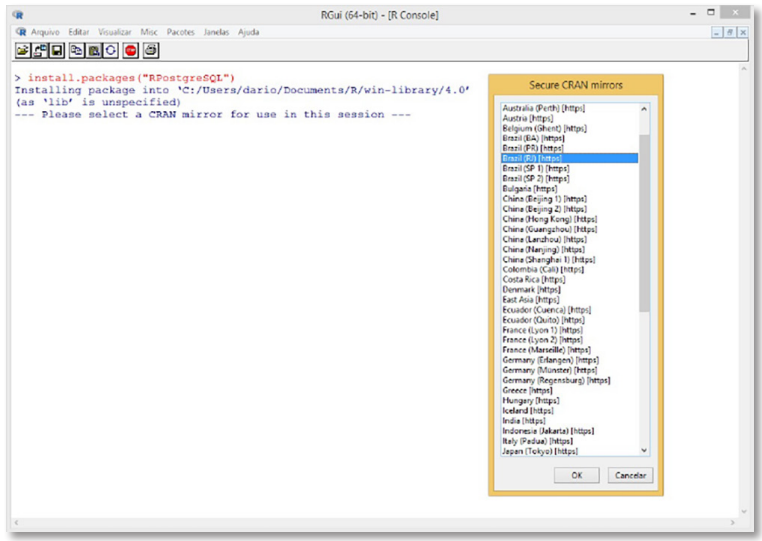


Figura 23. Instalação de pacotes no programa R.

Ao concluir a instalação do pacote, o programa R apresenta as informações referentes ao processo de instalação realizado (Figura 24).

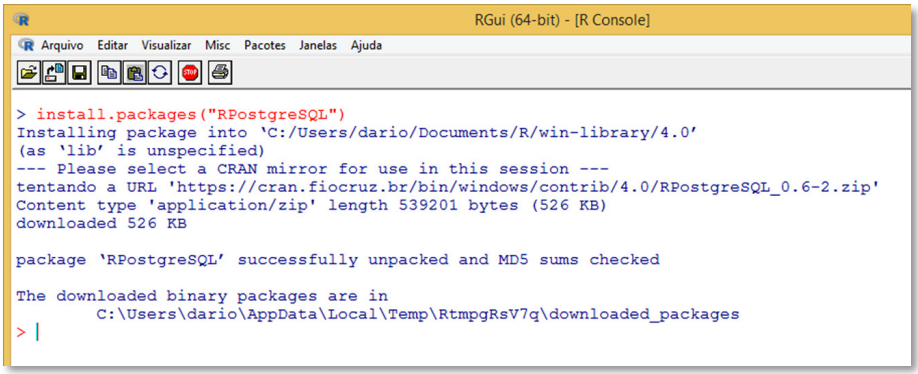


Figura 24. Informes sobre a instalação do pacote.

Neste momento, repita o procedimento realizado para instalar o segundo pacote, o "**sqldf**", mudando apenas o nome do pacote no comando de instalação:

>install.packages("sqldf")

Para disponibilizar o uso dos pacotes instalados, é necessário carregá-los para memória do computador. A seguir, é apresentada a sintaxe do comando em **R** para fazer isso:

Sintaxe:

require("<pacote>")

Em que:

<pacote> é o nome do pacote já instalado e que se deseja tornar disponível para uso.

Após a devida instalação dos dois pacotes, "**RPostgreSQL**" e "**sqldf**", siga o procedimento a seguir para carregá-los em memória e disponibilizá-los para uso no programa **R**.

>require("RPostgreSQL")

Ao executar um comando “require” para o pacote “**RPostgreSQL**” pela primeira vez na sessão do programa R, ele ainda não estará carregado em memória, assim, o programa R carrega todos os pacotes relacionados ao pacote requerido para a memória do computador. Caso o pacote já tenha sido carregado, nenhuma ação é tomada pelo programa R e o cursor retorna para a linha de comando (Figura 25).

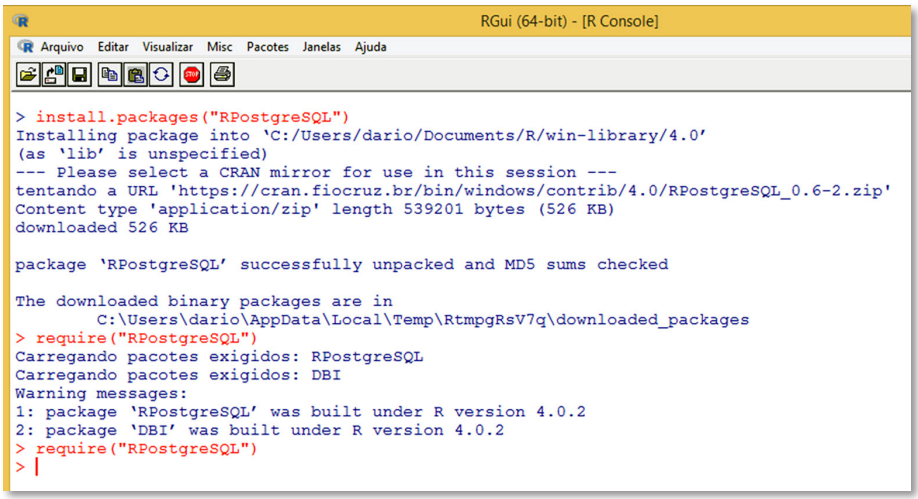


Figura 25. Carregando pacotes em memória.

Agora, repita o comando para carregar em memória e disponibilizar para uso o segundo pacote, o “**sqldf**”, mudando apenas o nome do pacote no comando executado:

>require(“sqldf”)

Criando a conexão de banco de dados com o PostgreSQL

Para estabelecer uma conexão com o PostgreSQL, siga o passo a passo do *script* descrito, a seguir, no ambiente R:

############# INÍCIO DO SCRIPT R #############
Estabelecendo uma conexão com o PostgreSQL
A password vem da instalação do PostgreSQL (Anexo I)

```
>options(sqldf.RPostgreSQL.user =”postgres”,
sqldf.RPostgreSQL.password =”******”,
sqldf.RPostgreSQL.dbname =”experimento_db”,
sqldf.RPostgreSQL.host =”localhost”,
sqldf.RPostgreSQL.port =5432)

#############
# Ler a tabela experimento no banco de dados
>tab_experimento <- sqldf(“select * from experimento”)

#############
# Mostrar os dados da tabela experimento
>tab_experimento

#############
# Executar a função fn_obter_dados_experimento()
# Primeira forma:
>dados_exp <- sqldf(“select * from public.fn_obter_dados_experimento(1,
null, 2, ‘2020/04/18’::date, ‘2020/04/28’::date)”)

# Observar os resultados do sql neste primeiro formato da variável dados_exp
>dados_exp

# Segunda forma:
# Escolher os campos a serem mostrados no momento da análise
>dados_exp<- sqldf(
“select nome_experimento, nome_tratamento, repeticao,
data_registro, nome_variavel, valor_entrada, unidade_origem,
valor_padrao_calc, unidade_padrao
from public.fn_obter_dados_experimento(1, null, 2, ‘2020/04/18’::date,
‘2020/04/28’::date)”)

# Observar os resultados do sql neste segundo formato da variável dados_exp
>dados_exp
############# FIM DO SCRIPT R #############
```

Análise dos dados do estudo de caso no programa R

A versatilidade é uma das qualidades do programa R, pois, além dos pacotes disponibilizados em sua base, recebe contribuições de pesquisadores de todo o mundo na forma de novos pacotes e possibilita a criação de novas rotinas e funções para o desenvolvimento das mais variadas análises estatísticas. E, com o auxílio do *PostgreSQL*, essa tarefa de organização, preparação e análise dos dados ficou muito mais rápida, segura e eficiente. Para exemplificar como pode ser feito uma determinada análise estatística, a partir do procedimento de integração do PostgreSQL com o programa R, a seguir é apresentado o script em R com um estudo sobre os efeitos da aplicação de maturadores em cana-de-açúcar, conduzido pela usina, hipotética, PlanaSucar. Neste estudo, foi realizado um delineamento em Blocos Casualizados (DBC), estruturados em 4 blocos com os tratamentos: T1-Testemunha; T2-Polaris (dose 2 mL/ha); T3-Polaris (dose 4 mL/ha); T4-Polaris (dose 6 mL/ha); T5-Ethrel (dosagem 3 mL/ha); T6-Ethrel (dosagem 5 mL/ha); T7-Ethrel (dosagem 7 mL/ha). A característica de interesse analisada é a produtividade de cana-de-açúcar (quilos/hectare).

Para a criação de um script, primeiramente clique em "**Arquivo → Novo script**" (Figura 26). Automaticamente, abrirá uma janela; em seguida, clique em "**Arquivo → Salvar como**..." (Figura 27) para salvar no local desejado o script. Para executar as ações das linhas de comandos digitadas no script, basta posicionar o cursor na respectiva linha ou selecionar as linhas de comandos que deseja executar e pressionar as teclas Ctrl + R.

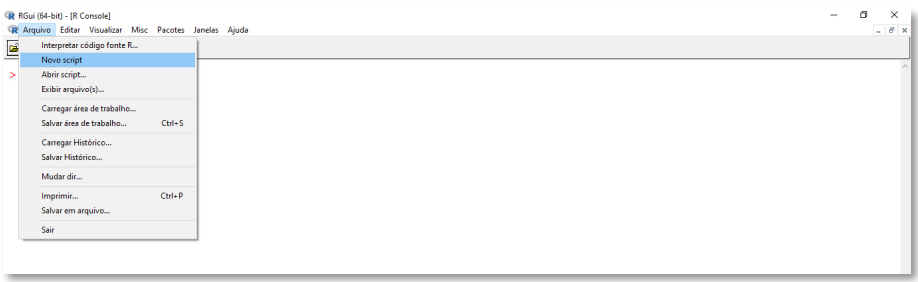


Figura 26. Criação de um script no programa R.

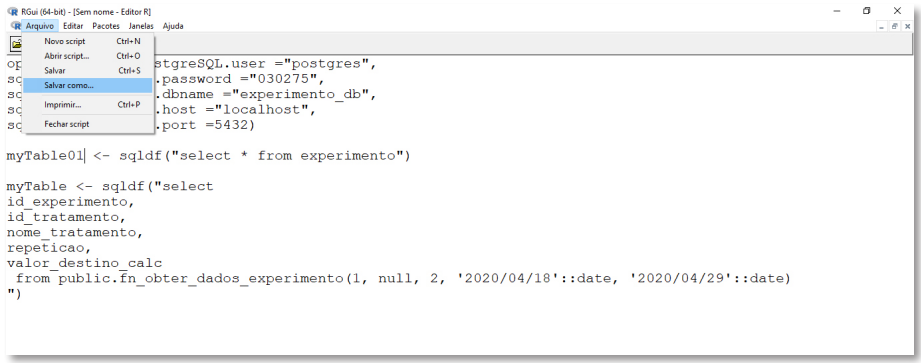


Figura 27. Salvando um script criado no programa R.

Após a instalação dos pacotes "**RPostgreSQL**" e "**sqldf**", conforme já orientado, para se aplicar a análise estatística no programa R, deve-se proceder os seguintes passos indicados no *script*, a seguir, para a importação dos dados.

Sintaxe:

```
>require("RPostgreSQL")
>require("sqldf")
>myTable <- sqldf("select
        id_experimento,
        id_tratamento,
        nome_tratamento,
        repeticao,
        valor_destino_calc
from public.fn_obter_dados_experimento(1, null, 2, '2020/04/18'::date,
'2020/04/29'::date)")

>myTable
```

Vale ressaltar que a função criada no PostgreSQL "obter_dados_experimento()" é de fundamental importância nesta tarefa em que o usurário pode realizar a seleção dos dados a serem analisados. Com a referida função, é possível solicitar os dados a serem analisados somente informando: experimento, tratamento, variável de interesse, data de início do registro do dado e data final do registro do dado que se deseja analisar.

Combinando a função do PostgreSQL “obter_dados_experimento()” com a função sqldf() do programa R, tem-se a possibilidade de acessar o banco de dados diretamente pelo console do programa R.

A partir do *script* acima foram importadas as seguintes colunas do banco de dados “myTable”: id_experimento, id_tratamento, nome_tratamento, repetição e valor_destino_calc. As variáveis nome_tratamento e valor_destino_calc, correspondem respectivamente aos amadurecedores e produtividade de cana-de-açúcar.

Vale ressaltar que, com auxílio da “**tabela conversão**” criada no PostgreSQL, foi possível realizar a conversão dos dados de produtividade originalmente em quilos/hectare para a unidade de sacas/hectare de maneira totalmente automática. Esse pode ser um recurso de grande valia para o pesquisador no momento da análise dos dados.

Veja, na Figura 28, a aparência da tabela de dados para análise no console do programa R.

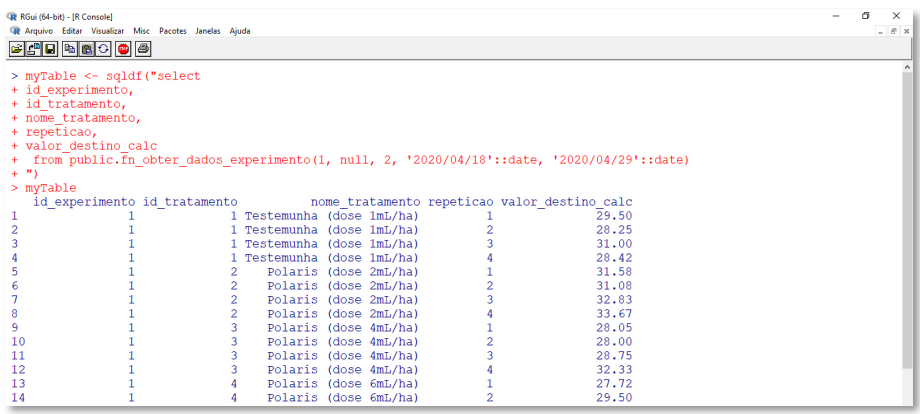


Figura 28. Aparência da tabela de dados para análise no console do programa R.

Após a importação dos dados, é possível verificar se a tabela de dados a ser analisada foi devidamente importada. Uma forma é simplesmente chamar o objeto que representa a tabela de dados, neste caso, “myTable” (Figura 28). A segunda forma é utilizar as funções “head()” e “tail()” para visualizar as cinco primeiras e as cinco últimas linhas da tabela de dados (Figura 29).

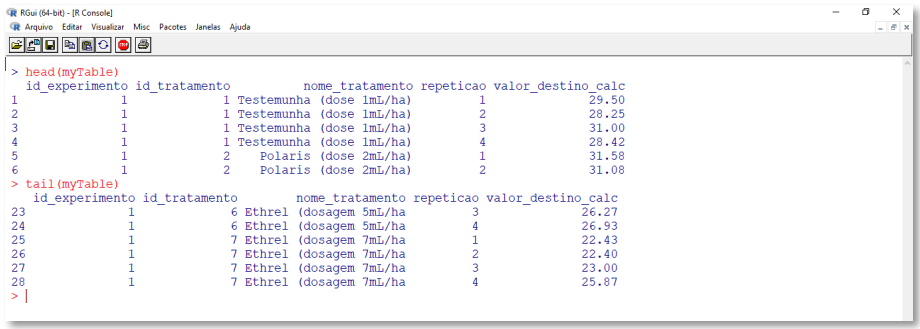


Figura 29. Formas de verificação se a tabela de dados a ser analisada foi devidamente importada.

No processo de início da análise exploratória dos dados, para se obter as principais medidas de posição: Mínimo, 1º Quartil, Mediana, Média, 3º Quartil e Máximo, usa-se a função summary() (Figura 30).

Sintaxe:

>summary(myTable$valor_destino_calc)

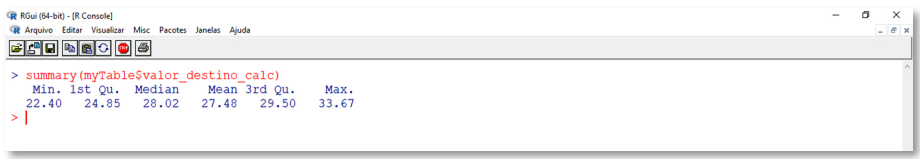


Figura 30. Principais medidas de posição na análise exploratória dos dados.

Para se realizar a estatística descritiva de uma determinada variável de interesse, neste estudo de caso, a produtividade representada pela “valor_destino_calc”, em função de um fator, que são os amadurecedores representados por “nome_tratamento”, pode-se utilizar a função tapply() (Figura 31).

Sintaxe:

Indicado para o cálculo das médias:

>with(myTable,tapply(valor_destino_calc,nome_tratamento,mean))

```
### Indicado para o cálculo do desvio padrão:
>with(myTable,tapply(valor_destino_calc,nome_tratamento,sd))
```

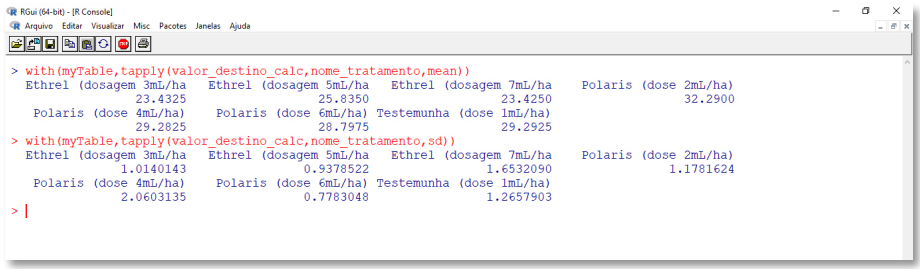


Figura 31. Utilização da função tapply.

Na análise exploratória dos dados, orienta-se realizar uma análise gráfica a fim de verificar o comportamento dos dados e a identificação de possíveis outliers. Para tanto, pode-se utilizar, nesta tarefa, a função boxplot() para gerar o gráfico de boxplot, no qual, são representadas de forma gráfica as medidas de posição : Mínimo, 1º Quartil, Mediana, Média, 3º Quartil e Máximo. A função points() auxilia na identificação da média e na sua inserção no gráfico (Figura 32).

Sintaxe:

```
>boxplot(myTable$valor_destino_calc~myTable$nome_tratamento)
>points(with(myTable,tapply(valor_destino_calc,nome_tratamento,-
mean)))
```

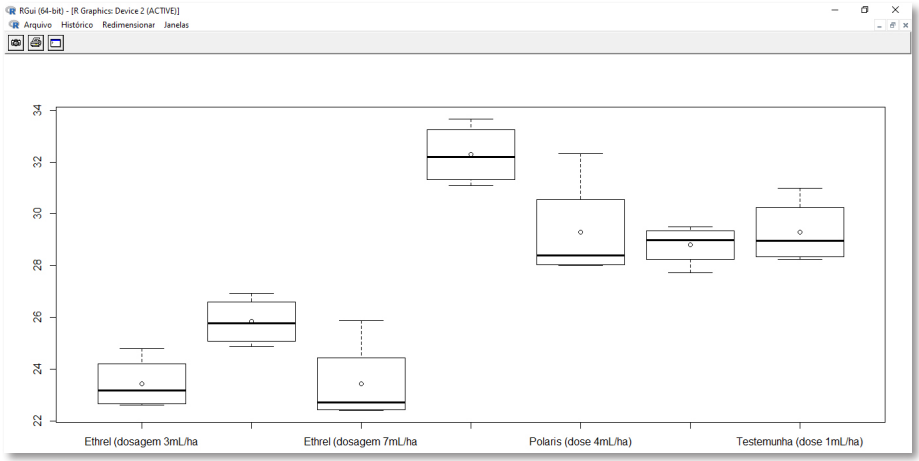


Figura 32. Gráfico Boxplot.

No processo de geração da análise inferencial dos dados, para se obter a Análise de Variância (Anova), utiliza-se a função aov() (Figura 33). Neste caso, em que o delineamento adotado é o DBC, a seguinte linha de comando foi construída:

Sintaxe:

```
>model.anova <- aov(valor_destino_calc ~ factor(repeticao) +
nome_tratamento, data=myTable)
>summary(model.anova)
```

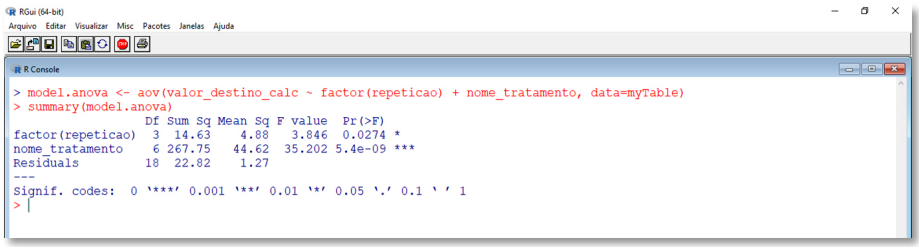


Figura 33. Análise de Variância (Anova)

Conforme demonstrado na Figura 33, os resultados indicam que existe diferença significativa entre os tratamentos. Mas ainda é necessário verificar se os pressupostos de normalidade dos resíduos e a homogeneidade da variân-

cia foram devidamente atendidos. Tais pressupostos são de suma importância, pois eles validam o resultado da Anova. Sendo assim, para se averiguar tais pressupostos utilizam-se as seguintes funções: “shapiro.test()” e “bartlett.test()”, necessárias para se verificar normalidade dos resíduos e homogeneidade da variância, respectivamente (Figura 34).

Sintaxe:

```
>shapiro.test(model.anova$res)
>bartlett.test(valor_destino_calc ~ nome_tratamento, data = myTable)
```

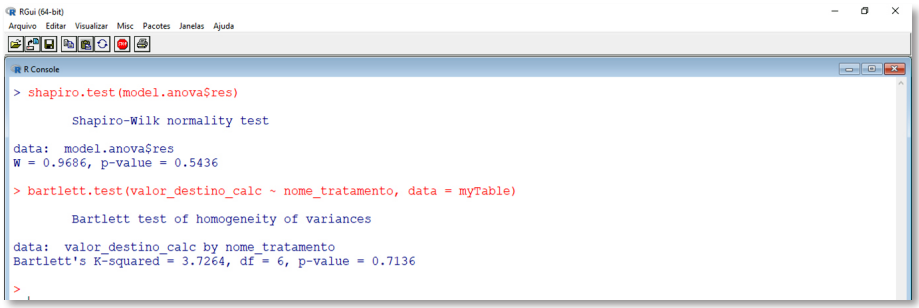


Figura 34. Testes de normalidade dos resíduos e homogeneidade da variância.

Um critério prático para se verificar o atendimento dos pressupostos de normalidade e de homogeneidade é observar se o p-valor gerado em ambos os testes estatísticos foram superiores a 0,05 (5%), correspondente ao nível de significância geralmente adotado em estudos científicos. E, conforme observar-se na Figura 34, todos os pressupostos foram devidamente atendidos.

Agora que foi atestado e validado que existe diferença estatística significativa entre os tratamentos “maturadores e cana-de-açúcar”, somente resta realizar um teste de comparação múltipla para se verificar como se comportam tais diferenças entre os tratamentos. Para tanto, aplica-se o teste de tukey, a partir da função “HSD.test ()”, conforme é indicado na Figura 35. Para a utilização dessa função, é necessário realizar o requerimento do “pacote agricolae”.

Sintaxe:

```
>require(“agricolae”)
>saida <- HSD.test(model.anova,”nome_tratamento”)
>saída
```

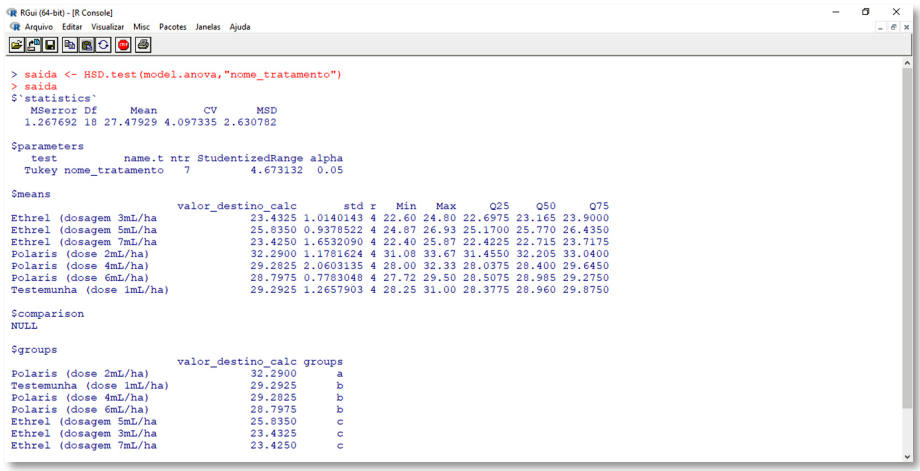


Figura 35. Teste de comparação múltipla de Tukey.

Para se facilitar a visualização dos tratamentos que foram estatisticamente diferentes, pode-se utilizar esta função, a seguir, para gerar um gráfico auxiliar com a disposição das médias, com a barra de desvio-padrão (indicado na função pelo parâmetro “SD” que são as siglas em inglês de “Standard Deviation”) e as letras que representam o resultado do teste de tukey (Figura 36).

Sintaxe:

```
>plot(saida,”SD”)
```

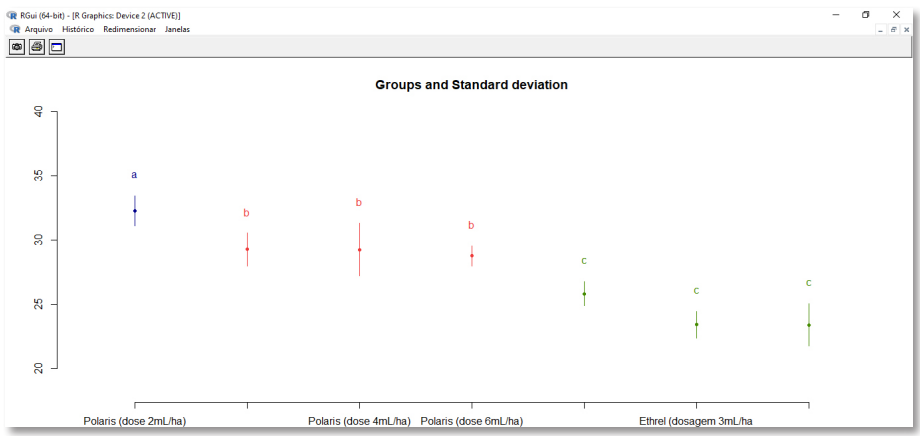


Figura 36. Apresentação gráfica dos resultados do teste de Tukey.

Considerações finais

Neste documento, não se pretendeu de resolver todos os problemas relacionados com o processo de organização, preparação e análise dos dados, mas sim, colaborar, oferecendo uma alternativa totalmente viável e eficiente para que os pesquisadores e estatísticos possam ter seus dados organizados de maneira mais segura, a partir de um poderoso sistema de gerenciamento de banco de dados totalmente gratuito, que é o caso do *PostgreSQL*, e ainda, ter a sua perfeita integração com o programa de geração de cálculos estatísticos mais robusto, utilizado, atualmente, pelas principais instituições de pesquisa do mundo, que é o programa R. Foi apresentado um exemplo que mostra que tal processo é perfeitamente factível e que todas as etapas foram devidamente descritas, testadas e comprovadas. Espera-se que este documento possa colaborar com o aumento da agilidade, a proteção e o desenvolvimento de estudos futuros.

Referências

CONWAY, J.; EDDELBUETTEL, D.; NISHIYAMA, T.; PRAYAGA, S. K.; TIFFIN, N. **RPostgreSQL**: R Interface to the 'PostgreSQL' Database System. 2017. Disponível em: https://CRAN.R-project.org/package=RPostgreSQL. Acesso em: 08 set 2020.

DB SOFTWARE LABORATORY. **Database Browser, 5.3.2.13**. Disponível em: https://www.etl-tools.com/database-browser/overview.html. Acesso em: 11 set. 2020a.

EDB. Enterprisedb. **Documentação**. PostgreSQL documentation. Disponível em: https://www.enterprisedb.com/edb-docs/p/postgresql. Acesso em: 13 set. 2020b.

EDB. Enterprisedb. **Guia Instalação PostgreSQL**. Invoking the Graphical Installer. Disponível em: https://www.enterprisedb.com/edb-docs/d/postgresql/installation-getting-started/installation-guide/13.0/invoking_the_graphical_installer.html. Acesso em: 12 set. 2020c.

EDB. Enterprisedb. **PostgreSQL**. PostgreSQL Database Download. Disponível em: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads. Acesso em: 12 set. 2020d.

EDB. Enterprisedb. **Stack Builer**. Using Stack Builer. Disponível em: https://www.enterprisedb.com/edb-docs/d/postgresql/installation-getting-started/installation-guide-installers/9.6/PostgreSQL_Installation_Guide.1.09.html. Acesso em: 12 set. 2020e.

EDB. Enterprisedb. **Tipos de Dados**. Data Types. Disponível em: https://www.enterprisedb.com/edb-docs/d/postgresql/reference/manual/12.3/datatype.html. Acesso em: 20 set. 2020f.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. **Sistema de Banco de Dados**. 3. ed. São Paulo: Pearson Makron Books, 1999.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **Data Types**. Chapter 8. Disponível em: https://www.postgresql.org/docs/12/datatype.html. Acesso em: 20 set. 2020.

Anexo I. Instalação do PostgreSQL.

Para instalar o SGBD *PostgreSQL*, primeiramente, acesse a página de download em (EDB - Enterprisedb - PostgreSQL, 2020) e escolha a versão do *PostgreSQL* e o sistema operacional (SO) em que vai ser instalado. Neste tutorial, será instalado a versão mais recente 12.4 no SO Windows x86-64. Clique em **Download** na coluna do *Windows x86-64* (Figura 1).

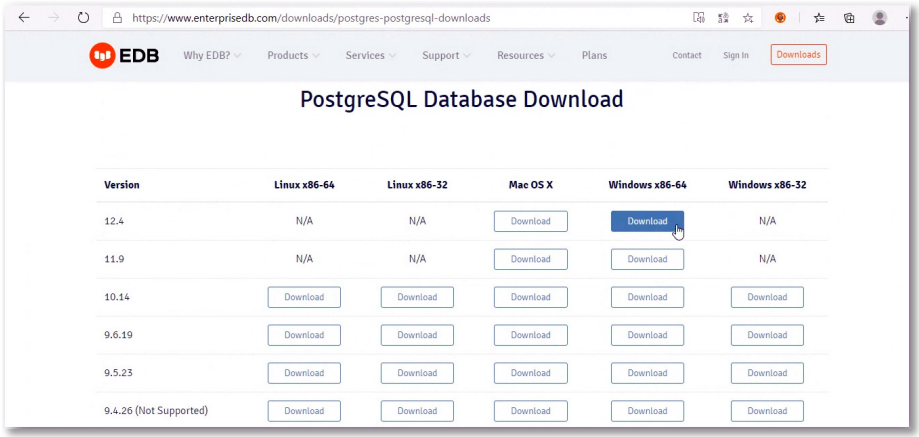


Figura 1. Download do Instalador para Windows.

Na sequência, o instalador irá solicitar o diretório de destino para salvar o arquivo de instalação (postgresql-12.4-1-windows-x64.exe). Enquanto o *download* é realizado, a tela muda para a Figura 2.

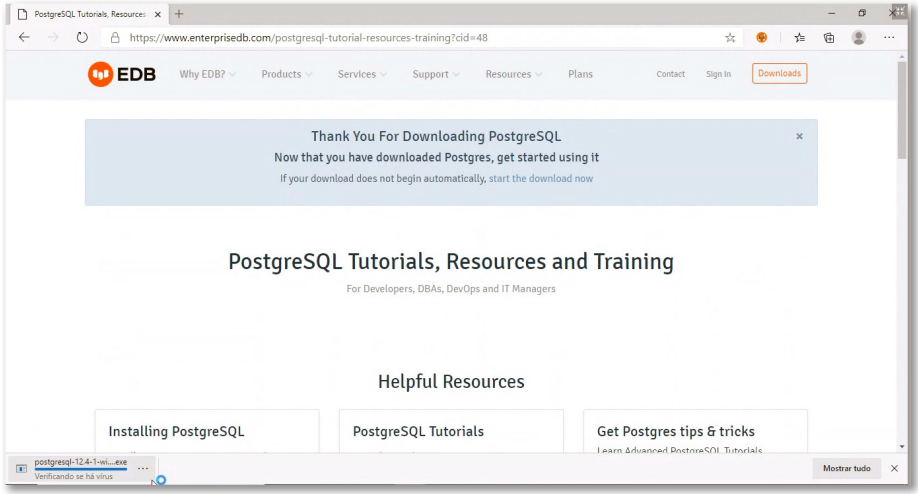


Figura 2. Tela de espera do Download.

Quando terminar de baixar o instalador, execute-o como administrador do Windows para ter as permissões necessárias para instalar o *PostgreSQL*.

Ao executar o instalador, surgirá a tela, conforme a Figura 3. Clique em **Next** para continuar:

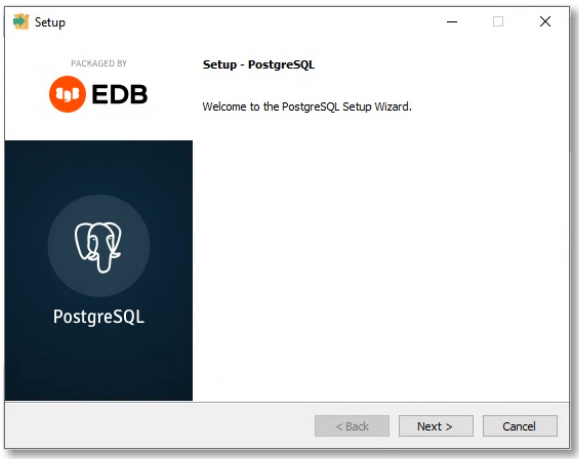


Figura 3. Tela inicial do instalador do PostgreSQL.

É possível indicar o diretório em que será instalado o software, porém orientamos a deixar o instalador utilizar o diretório padrão indicado na Figura 4. Clique em **Next** para continuar.

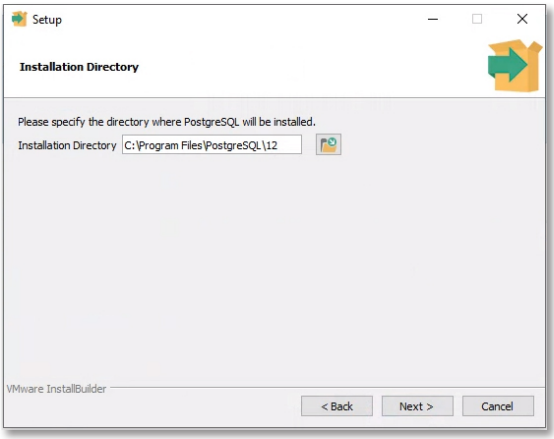


Figura 4. Escolha do local de instalação do *PostgreSQL*.

A seguir, deixe selecionados os componentes listados na Figura 5. Clique em **Next** para continuar.

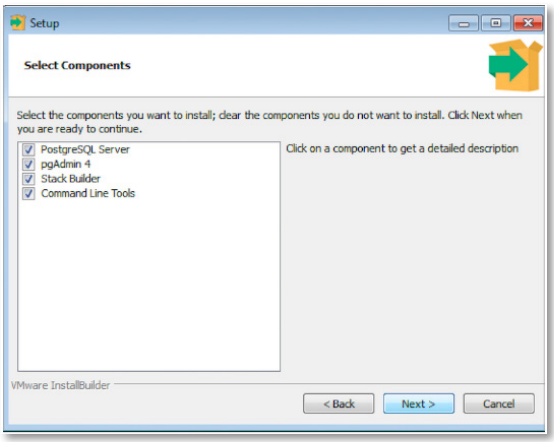


Figura 5. Seleção dos componentes para instalação.

Novamente, orienta-se deixar o instalador utilizar o diretório padrão indicado (Figura 6). Clique em **Next** para continuar.

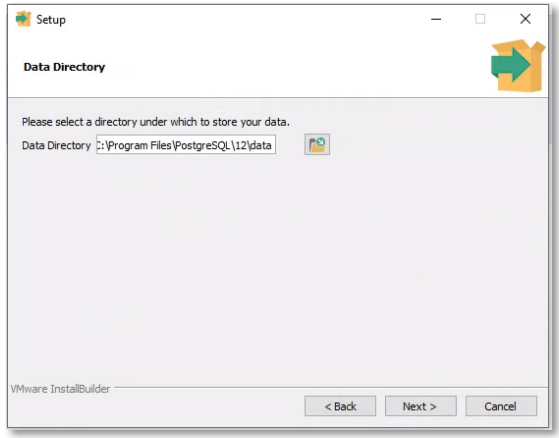


Figura 6. Definição do diretório para gravar os dados.

A seguir, um passo importante (Figura 7): a definição da senha do superusuário do *PostgreSQL*, o **postgres**. Essa senha não deve ser esquecida, pois não há como recuperá-la e se isso acontecer, perde-se o aceso ao banco de dados e a todos os dados, assim muita atenção ao defini-la. Clique em **Next** para continuar.

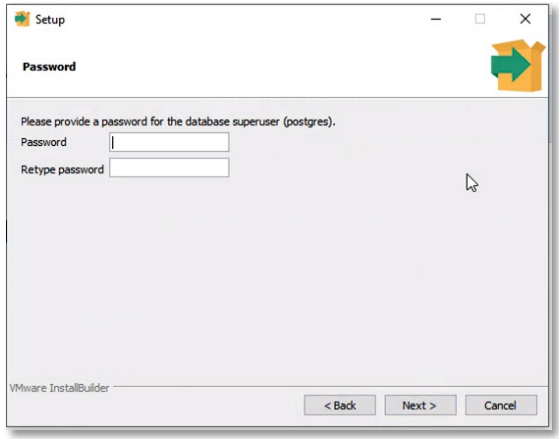


Figura 7. Definição da senha do superusuário postgres.

Na tela seguinte (Figura 8), é preciso definir a porta virtual de acesso ao SGBD *PostgreSQL*. É padrão utilizar a porta indicada pelo instalador (5432), quando se trata da primeira instalação. Mas, caso exista uma versão do *PostgreSQL* já instalada no computador, é preciso mudar o número da porta para possibilitar uma nova instalação, por exemplo, adicionando 1 ao número da porta indicada. Clique em **Next** para continuar.

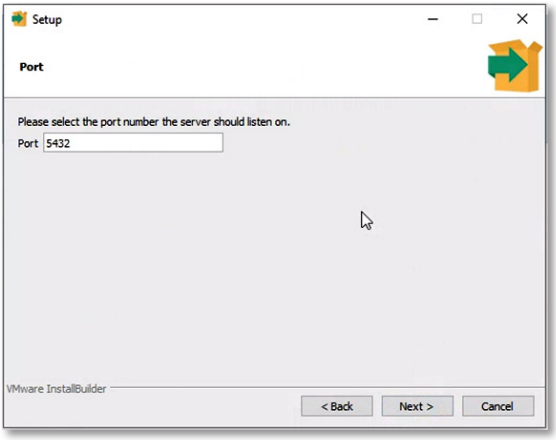


Figura 8. Definição da porta de acesso ao *PostgreSQL*.

A seguir (Figura 9), selecione a língua e o país que será instalado o *PostgreSQL*. Clique em **Next** para continuar.

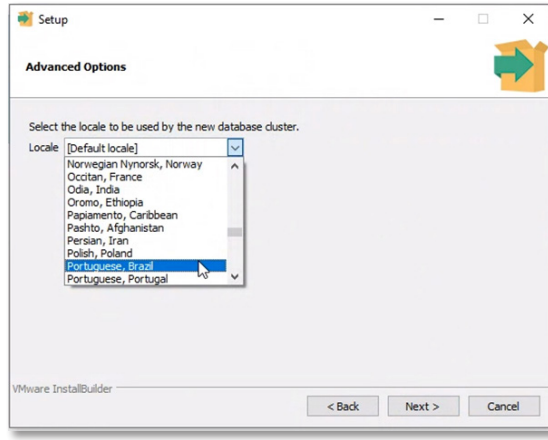


Figura 9. Informação da língua e país para o *PostgreSQL*.

O instalador irá mostra um resumo da configuração construída para a instalação do *PostgreSQL* no computador (Figura 10). É possível retornar e mudar a configuração, caso seja necessário, clicando no botão **Back** ou seguir em frente com **Next** para continuar.

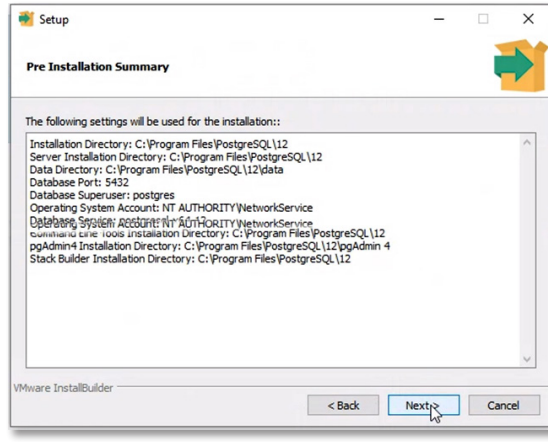


Figura 10. Resumo da configuração para instalação do *PostgreSQL*.

A seguir, o instalador mostra uma mensagem indicando que está pronto para começar a instalação (Figura 11). Clique em **Next** para continuar.

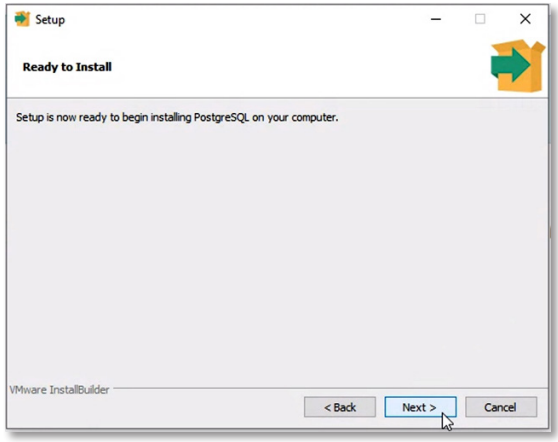


Figura 11. Mensagem de pronto para instalar.

Na sequência (Figura 12), é apresentada uma tela para acompanhar a instalação dos componentes selecionados (Figura 5) no computador. Clique em **Next** para continuar.

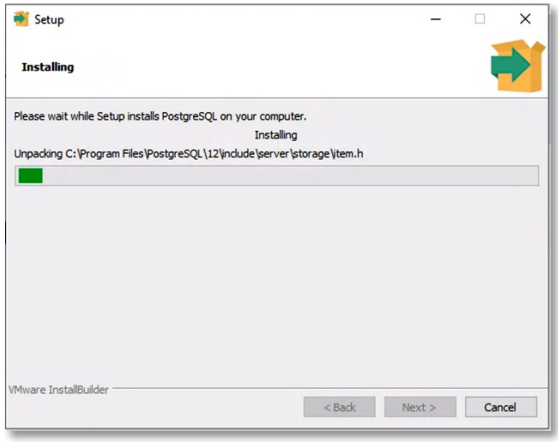


Figura 12. Barra de progresso da instalação.

Ao concluir a instalação (Figura 13), é apresentada a última tela informando que a instalação do *PostgreSQL* chegou ao seu final. Desmarque o componente *Starck Buider* (EDB - Enterprisedb - Stack Builer, 2020), que, no momento, não será necessário, pois não haverá instalação de nenhum complemento ao *PostgreSQL*. Clique em **Finish** para terminar.

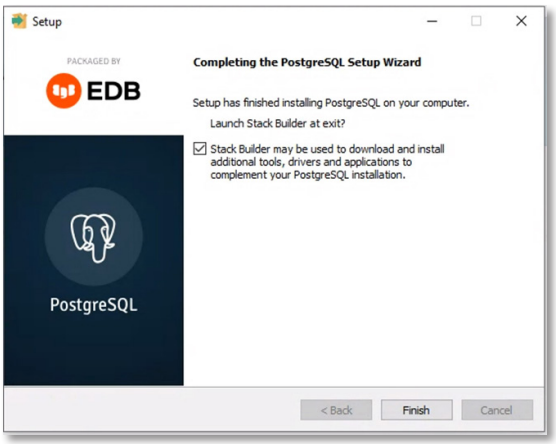


Figura 13. Tela de conclusão da instalação.

Mais informações sobre o processo de instalação podem ser acessadas na página https://www.enterprisedb.com/edb-docs/p/postgresql e para um aprofundamento no conhecimento sobre o *PostgreSQL*, a documentação pode ser baixada nesse mesmo link.

Anexo II. Script do Banco de Dados.

Um script de banco de dados é uma sequência de instruções escrita na linguagem SQL (*Structured Query Language*) e é utilizado para informar ao Sistema Gerenciador de Banco de Dados (SGBD) o que ele deve fazer, tais como:

- Criar um banco de dados, criar tabelas.;
- Inserir, alterar ou excluir dados em tabelas.
- Consultar dados, entre outros.

Criação do banco de dados

```
CREATE DATABASE experimento_db WITH
OWNER = “postgres”
ENCODING = ‘UTF8’
TABLESPACE = pg_default;
```

Criação das tabelas e relacionamentos

Tabela: experimento

```
CREATE TABLE public.experimento (
  id_experimento              integer NOT NULL,
  nome_experimento            varchar(30),
  descricao_experimento       text,
  /* Keys */
CONSTRAINT experimento_pk
PRIMARY KEY (id_experimento)
) WITH (
  OIDS = FALSE
  );
ALTER TABLE public.experimento
  OWNER TO postgres;
```

```
COMMENT ON TABLE public.experimento
  IS 'Tabela de experimentos';

COMMENT ON COLUMN public.experimento.id_experimento
  IS 'Identificacação do experimento';

COMMENT ON COLUMN public.experimento.nome_experimento
  IS 'Nome do experimento';

COMMENT ON COLUMN public.experimento.descricao_experimento
  IS 'Descrição do experimento';
```

Tabela: tratamento

```
CREATE TABLE public.tratamento (
  id_experimento            integer NOT NULL,
  id_tratamento             integer NOT NULL,
  nome_tratamento           varchar(50),
  descricao_tratamento      varchar(50),
  /* Keys */
        CONSTRAINT tratamento_pk
          PRIMARY KEY (id_experimento, id_tratamento),
        /* Foreign keys */
        CONSTRAINT experimento_tratamento_pfk
      FOREIGN KEY (id_experimento)
        REFERENCES public.experimento(id_experimento)
      ) WITH (
        OIDS = FALSE
      );

      ALTER TABLE public.tratamento
        OWNER TO postgres;

      COMMENT ON COLUMN public.tratamento.id_experimento
        IS 'Identificacao do experimeto.';

      COMMENT ON COLUMN public.tratamento.id_tratamento
        IS 'Identificação do tratamento';
```

```
COMMENT ON COLUMN public.tratamento.nome_tratamento
  IS 'Nome do tratamento';

COMMENT ON COLUMN public.tratamento.descricao_tratamento
  IS 'Descrição do tratamento.';
```

Tabela: conversao

```
CREATE TABLE public.conversao (
  id_unidade_origem   integer NOT NULL,
  id_unidade_destino   integer NOT NULL,
  fator_conversao       numeric(14,6) NOT NULL,
  /* Keys */
  CONSTRAINT conversao_pk
    PRIMARY KEY (id_unidade_origem, id_unidade_destino),
  /* Foreign keys */
  CONSTRAINT unidade_conversao_destino_fk
    FOREIGN KEY (id_unidade_destino)
    REFERENCES public.unidade(id_unidade),
  CONSTRAINT unidade_conversao_origem_fk
    FOREIGN KEY (id_unidade_origem)
    REFERENCES public.unidade(id_unidade)
) WITH (
    OIDS = FALSE
  );

ALTER TABLE public.conversao
  OWNER TO postgres;
```

Tabela: unidade

```
CREATE DOMAIN public.dominio_classe_unidade AS integer;

ALTER DOMAIN public.dominio_classe_unidade
  ADD CONSTRAINT dominio_classe_unidade_check
  CHECK (VALUE = ANY (ARRAY[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]));

ALTER DOMAIN public.dominio_classe_unidade
  OWNER TO postgres;

COMMENT ON DOMAIN public.dominio_classe_unidade
```

```
  IS ‘ 1. distância;
   2. área;
   3. volume;
   4. massa;
   5. concentração;
   6. taxa;
   7. velocidade;
   8. tempo;
   9. produção;
  10. força;
  11. temperatura’;

CREATE TABLE public.unidade (
  id_unidade        integer NOT NULL,
  sigla_unidade     varchar(40) NOT NULL,
  nome_unidade      varchar(50),
  classe_unidade    public.dominio_classe_unidade,
  /* Keys */
  CONSTRAINT unidade_pk
    PRIMARY KEY (id_unidade)
) WITH (
    OIDS = FALSE
  );

ALTER TABLE public.unidade
  OWNER TO postgres;

COMMENT ON COLUMN public.unidade.classe_unidade
  IS ‘value in (1,2)
1 distância
2 área
3 volume
4 massa
5 concentração
6 taxa
7 velocidade
8 tempo
9 produção
10 força
11 temperatura’;
```

Tabela: variavel

```
CREATE TABLE public.variavel (
  id_variavel              integer NOT NULL,
  nome_variavel            varchar(30),
  id_unidade_padrao        integer,
  /* Keys */
  CONSTRAINT variavel_pk
    PRIMARY KEY (id_variavel),
  /* Foreign keys */
  CONSTRAINT unidade_variavel_padrao
    FOREIGN KEY (id_unidade_padrao)
    REFERENCES public.unidade(id_unidade)
) WITH (
    OIDS = FALSE
  );

ALTER TABLE public.variavel
  OWNER TO postgres;

COMMENT ON TABLE public.variavel
  IS 'Contém a lista de variáveis e a determinação de sua unidade
padrão';

COMMENT ON COLUMN public.variavel.id_variavel
  IS 'Identificador da variável';

COMMENT ON COLUMN public.variavel.nome_variavel
  IS 'Nome da Variável';

COMMENT ON COLUMN public.variavel.id_unidade_padrao
  IS 'Identifica qual é a unidade padrão desta variável para sua
participação em cálculos.';
```

Tabela: valor_variavel

```
CREATE TABLE public.valor_variavel (
  id_experimento           integer NOT NULL,
  id_tratamento            integer NOT NULL,
  id_variavel              integer NOT NULL,
  data_registro            date NOT NULL,
  repeticao                integer NOT NULL,
```

```
    id_unidade_origem       integer,
    valor_entrada           real,
    id_unidade_destino      integer,
    /* Keys */
    CONSTRAINT valor_variavel_pk
      PRIMARY KEY (id_experimento, id_tratamento, id_variavel,
repeticao, data_registro),
    /* Foreign keys */
    CONSTRAINT conversao_vvariavel_fk
      FOREIGN KEY (id_unidade_origem, id_unidade_destino)
      REFERENCES public.conversao(id_unidade_origem, id_unidade_
destino),
    CONSTRAINT tratamento_valor_pfk
      FOREIGN KEY (id_experimento, id_tratamento)
      REFERENCES public.tratamento(id_experimento, id_tratamento),
    CONSTRAINT valor_variavel_pfk
      FOREIGN KEY (id_variavel)
      REFERENCES public.variavel(id_variavel)
  ) WITH (
    OIDS = FALSE
  );

ALTER TABLE public.valor_variavel
  OWNER TO postgres;

COMMENT ON TABLE public.valor_variavel
  IS 'Valor da variavel';

COMMENT ON COLUMN public.valor_variavel.id_experimento
  IS 'Identificacao do experimento';

COMMENT ON COLUMN public.valor_variavel.id_tratamento
  IS 'Identificação do tratamento';

COMMENT ON COLUMN public.valor_variavel.id_variavel
  IS 'Identificação da variavel';

COMMENT ON COLUMN public.valor_variavel.data_registro
  IS 'Data de registro de coleta do valor da variavel.';

COMMENT ON COLUMN public.valor_variavel.repeticao
  IS 'Repetição';
```

```
COMMENT ON COLUMN public.valor_variavel.id_unidade_origem
  IS 'Identificador da unidade de medida utilizada na entrada do
dado.';

COMMENT ON COLUMN public.valor_variavel.valor_entrada
  IS 'Valor do dado na unidade de medida indicada no id_unidade_
origem';

COMMENT ON COLUMN public.valor_variavel.id_unidade_destino
  IS 'Identificador da unidade de medida utilizada como padrão para
este dado.';
```

Criação das funções

Função para obter dados de um determinado experimento

```
--Function: public.fn_obter_dados_experimento(integer, integer, integer,
date, date)
--DROP FUNCTION public.fn_obter_dados_experimento(integer, integer,
integer, date, date);

CREATE OR REPLACE FUNCTION public.fn_obter_dados_experimento
(
  IN  par_id_experimento  integer,
  IN  par_id_tratamento   integer,
  IN  par_id_variavel     integer,
  IN  par_data_inicio     date,
  IN  par_data_final      date
)
RETURNS TABLE
(
  id_experimento          integer,
  nome_experimento        varchar,
  id_tratamento           integer,
  nome_tratamento         varchar,
  repeticao               integer,
  data_registro           date,
  id_variavel             integer,
  nome_variavel           varchar,
  valor_entrada           numeric,
  id_unidade_origem       integer,
```

```
  unidade_origem          varchar,
  valor_destino_calc      numeric,
  id_unidade_destino      integer,
  unidade_destino         varchar,
  valor_padrao_calc       numeric,
  id_unidade_padrao       integer,
  unidade_padrao          varchar
)
AS
$$
/* ****************************************************************************\
     -- Funcao que retorna os dados de um determinado experimento para
uso no programa R
     --
     -- 09/09/2020
     ****************************************************************************/
   /* PARAMETROS:
        $1- par_id_experimento  => id do experimento a ser lido
        $2- par_id_tratamento   => id do tratamento dentro do experimento
        $3- par_id_variavel     => id da variavel que sera utilizada na estatistica
        $4- par_data_inicio     => data de inicio de leitura dos dados
        $5- par_data_final      => data final de leitura dos dados

      Exemplo de chamada dessa funcao:
        select * from public.fn_obter_dados_experimento(1, 1, 2,
'2020/04/18'::date, '2020/04/28'::date);

        select * from public.fn_obter_dados_experimento(1, null, 2,
'2020/04/18'::date, '2020/04/28'::date);

        select * from public.fn_obter_dados_experimento(1, null, 2,
'2020/04/22'::date, '2020/04/24'::date)
         order by nome_variavel;

        select * from public.fn_obter_dados_experimento(1, null, 2,
'2020/04/22'::date, '2020/04/24'::date)
   */
begin
  /* Insert real code here */
  RETURN QUERY
     SELECT
        vvar.id_experimento,
```

```
        exp.nome_experimento,
        vvar.id_tratamento,
        trat.nome_tratamento,
        vvar.repeticao,
        vvar.data_registro,
        vvar.id_variavel,
        var.nome_variavel,
        -- valor na unidade de entrada
        vvar.valor_entrada::NUMERIC(15, 2),
        vvar.id_unidade_origem,
        und_origem.nome_unidade as unidade_origem,
        -- valor na unidade de destino
        (vvar.valor_entrada * conv.fator_conversao)::NUMERIC(15, 2) as
valor_destino_calc,
        vvar.id_unidade_destino,
        und_destino.nome_unidade as unidade_destino,
        -- valor na unidade padrao da variavel
        (vvar.valor_entrada * conv_padrao.fator_conversao)::NUMERIC(15, 2)
as valor_padrao_calc,
        var.id_unidade_padrao,
        und_padrao.nome_unidade as unidade_padrao
      FROM
        public.experimento exp, public.tratamento trat, public.valor_variavel
vvar,
        public.variavel var, public.unidade und_padrao,
        public.unidade und_origem, public.unidade und_destino,
        public.conversao conv, public.conversao conv_padrao
     WHERE
         trat.id_experimento     = exp.id_experimento
      AND vvar.id_experimento     = trat.id_experimento
      AND vvar.id_tratamento      = trat.id_tratamento
      AND vvar.id_variavel        = var.id_variavel
      --  Para pegar o nome da unidade padrao
      AND var.id_unidade_padrao   = und_padrao.id_unidade
      --  Para pegar o fator de conversao da unidade de entrada para unidade
destino
      AND vvar.id_unidade_origem  = conv.id_unidade_origem
      AND vvar.id_unidade_destino = conv.id_unidade_destino
      --  Para pegar os nomes das unidades de origem e de destino
      AND conv.id_unidade_origem  = und_origem.id_unidade
      AND conv.id_unidade_destino = und_destino.id_unidade
      --  Para pegar o fator de conversao da unidade de entra para unidade
```

```
padrao
     AND conv_padrao.id_unidade_origem  = vvar.id_unidade_origem
     AND conv_padrao.id_unidade_destino = var.id_unidade_padrao
     AND exp.id_experimento      = par_id_experimento
     AND ((par_id_tratamento is null) or (trat.id_tratamento = par_id_
tratamento))
     AND ((par_id_variavel   is null) or (vvar.id_variavel   = par_id_variavel))
     AND vvar.data_registro BETWEEN par_data_inicio and par_data_final;
  RETURN;
end;
$$
LANGUAGE 'plpgsql'
COST 100;
ALTER FUNCTION public.fn_obter_dados_experimento(integer, integer,
integer, date, date)
  OWNER TO postgres;
```

Anexo III. Instruções SQL para popular as tabelas.

Comandos SQL necessários para inserir os registros em suas respectivas tabelas.

```
-- Popupar experimento
INSERT INTO experimento (id_experimento, nome_experimento, descricao_experimento) VALUES
 (1, 'Cana 375-BR', 'Experimento da Usina PlanCana');

-- Popular tratamento
INSERT INTO tratamento (id_experimento, id_tratamento, nome_tratamento, descricao_tratamento) VALUES
 (1, 1, 'Testemunha (dose 1mL/ha)', 'Testemunha'),
 (1, 2, 'Polaris (dose 2mL/ha)', 'Tratamento 1'),
 (1, 3, 'Polaris (dose 4mL/ha)', 'Tratamento 2'),
 (1, 4, 'Polaris (dose 6mL/ha)', 'Tratamento 3'),
 (1, 5, 'Ethrel (dosagem 3mL/ha', 'Tratamento 4'),
 (1, 6, 'Ethrel (dosagem 5mL/ha', 'Tratamento 5'),
 (1, 7, 'Ethrel (dosagem 7mL/ha', 'Tratamento 6');

-- Popular unidades
INSERT INTO unidade (id_unidade, sigla_unidade, nome_unidade, classe_unidade) VALUES
 (2, '%', 'Percentual', 6),
 (23, 'kg/ha', 'Quilos por Hectáre', 9),
 (106, 'sc/ha', 'Sacas por Hectáre', 9);

-- Popular variaveis
INSERT INTO variavel (id_variavel, nome_variavel, id_unidade_padrao) VALUES
 (1, 'Brix', 2),
 (2, 'Produtividade', 106);

-- Popular conversao
INSERT INTO conversao (id_unidade_origem, id_unidade_destino, fator_conversao) VALUES
 (23, 106, 0.016667),
 (23, 23, 1),
 (106, 106, 1),
 (2, 2, 1);
```

```
-- Popular  valor_variavel
Insert into valor_variavel (id_experimento,id_tratamento,
id_variavel,data_registro,repeticao,valor_entrada,
id_unidade_origem,id_unidade_destino) VALUES
 (1,1,2,CAST('2020/04/18' AS DATE) ,1,1770.00,23,106),
 (1,1,2,CAST('2020/04/19' AS DATE) ,2,1695.00,23,106),
 (1,1,2,CAST('2020/04/20' AS DATE) ,3,1860.00,23,106),
 (1,1,2,CAST('2020/04/21' AS DATE) ,4,1705.00,23,106),
 (1,2,2,CAST('2020/04/22' AS DATE) ,1,1895.00,23,106),
 (1,2,2,CAST('2020/04/23' AS DATE) ,2,1865.00,23,106),
 (1,2,2,CAST('2020/04/24' AS DATE) ,3,1970.00,23,106),
 (1,2,2,CAST('2020/04/25' AS DATE) ,4,2020.00,23,106),
 (1,3,2,CAST('2020/04/26' AS DATE) ,1,1683.00,23,106),
 (1,3,2,CAST('2020/04/27' AS DATE) ,2,1680.00,23,106),
 (1,3,2,CAST('2020/04/28' AS DATE) ,3,1725.00,23,106),
 (1,3,2,CAST('2020/04/29' AS DATE) ,4,1940.00,23,106),
 (1,4,2,CAST('2020/04/22' AS DATE) ,1,1663.00,23,106),
 (1,4,2,CAST('2020/04/23' AS DATE) ,2,1770.00,23,106),
 (1,4,2,CAST('2020/04/24' AS DATE) ,3,1752.00,23,106),
 (1,4,2,CAST('2020/04/25' AS DATE) ,4,1726.00,23,106),
 (1,5,2,CAST('2020/04/26' AS DATE) ,1,1416.00,23,106),
 (1,5,2,CAST('2020/04/27' AS DATE) ,2,1356.00,23,106),
 (1,5,2,CAST('2020/04/28' AS DATE) ,3,1488.00,23,106),
 (1,5,2,CAST('2020/04/29' AS DATE) ,4,1364.00,23,106),
 (1,6,2,CAST('2020/04/22' AS DATE) ,1,1516.00,23,106),
 (1,6,2,CAST('2020/04/23' AS DATE) ,2,1492.00,23,106),
 (1,6,2,CAST('2020/04/24' AS DATE) ,3,1576.00,23,106),
 (1,6,2,CAST('2020/04/25' AS DATE) ,4,1616.00,23,106),
 (1,7,2,CAST('2020/04/26' AS DATE) ,1,1346.00,23,106),
 (1,7,2,CAST('2020/04/27' AS DATE) ,2,1344.00,23,106),
 (1,7,2,CAST('2020/04/28' AS DATE) ,3,1380.00,23,106),
 (1,7,2,CAST('2020/04/29' AS DATE) ,4,1552.00,23,106),
 (1,1,1,CAST('2020/04/18' AS DATE) ,1,17.70,2,2),
 (1,1,1,CAST('2020/04/19' AS DATE) ,2,16.95,2,2),
 (1,1,1,CAST('2020/04/20' AS DATE) ,3,18.60,2,2),
 (1,1,1,CAST('2020/04/21' AS DATE) ,4,17.05,2,2),
 (1,2,1,CAST('2020/04/22' AS DATE) ,1,18.95,2,2),
 (1,2,1,CAST('2020/04/23' AS DATE) ,2,18.65,2,2),
 (1,2,1,CAST('2020/04/24' AS DATE) ,3,19.70,2,2),
 (1,2,1,CAST('2020/04/25' AS DATE) ,4,20.20,2,2),
 (1,3,1,CAST('2020/04/26' AS DATE) ,1,16.83,2,2),
 (1,3,1,CAST('2020/04/27' AS DATE) ,2,16.80,2,2),
```

```
(1,3,1,CAST('2020/04/28' AS DATE) ,3,17.25,2,2),
(1,3,1,CAST('2020/04/29' AS DATE) ,4,19.40,2,2),
(1,4,1,CAST('2020/04/22' AS DATE) ,1,16.63,2,2),
(1,4,1,CAST('2020/04/23' AS DATE) ,2,17.70,2,2),
(1,4,1,CAST('2020/04/24' AS DATE) ,3,17.52,2,2),
(1,4,1,CAST('2020/04/25' AS DATE) ,4,17.26,2,2),
(1,5,1,CAST('2020/04/26' AS DATE) ,1,14.16,2,2),
(1,5,1,CAST('2020/04/27' AS DATE) ,2,13.56,2,2),
(1,5,1,CAST('2020/04/28' AS DATE) ,3,14.88,2,2),
(1,5,1,CAST('2020/04/29' AS DATE) ,4,13.64,2,2),
(1,6,1,CAST('2020/04/22' AS DATE) ,1,15.16,2,2),
(1,6,1,CAST('2020/04/23' AS DATE) ,2,14.92,2,2),
(1,6,1,CAST('2020/04/24' AS DATE) ,3,15.76,2,2),
(1,6,1,CAST('2020/04/25' AS DATE) ,4,16.16,2,2),
(1,7,1,CAST('2020/04/26' AS DATE) ,1,13.46,2,2),
(1,7,1,CAST('2020/04/27' AS DATE) ,2,13.44,2,2),
(1,7,1,CAST('2020/04/28' AS DATE) ,3,13.80,2,2),
(1,7,1,CAST('2020/04/29' AS DATE) ,4,15.52,2,2);
```

Anexo IV. Instalação do *Database Browser*.

Será utilizada a instalação do *Database Browser* em sua apresentação como **Software portátil** que, segundo (DB Software Laboratory, 2020), "é uma classe de software que é adequado para uso em unidades portáteis, como um drive USB (polegar) ou iPod ou Palm PDA com *drive mode*, embora qualquer disco rígido externo poderia teoricamente ser usado. O conceito de carregar aplicativos, utilitários e arquivos preferidos em uma unidade portátil para uso em qualquer computador é aquele que evoluiu consideravelmente nos últimos anos".

Para instalar o *Database Browser* (Navegador de Banco de Dados), em sua forma portátil, acesse a página oficial para fazer o download em (DB Software Laboratory, 2020) . Nela, selecione o botão **PORTABLE** (Figura 1), para baixar o arquivo *DatabaseBrowser_Portable.zip*. Depois, descompacte seu conteúdo no diretório que preferir. Uma vez descompactado, haverá um diretório com o nome *DatabaseBrowserPortable* e, nele, o programa instalador:

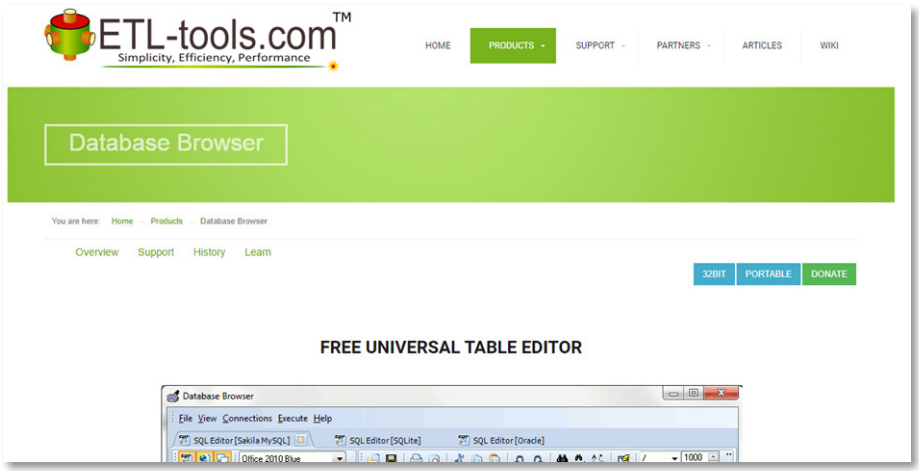


Figura 1. Página de download do Database Browser.

DatabaseBrowserPortable_5.3.2.13_English.paf.exe

Os números “5.3.2.13” no nome do arquivo representam a versão atual do software e podem divergir caso haja melhoramentos no software e seja disponibilizada uma nova versão. Por se tratar de um arquivo executável, orienta-se passar o antivírus antes de executá-lo.

Ao executar o instalador, a seguinte tela surgira (Figura 2), tecle em **Next** para iniciar a instalação.

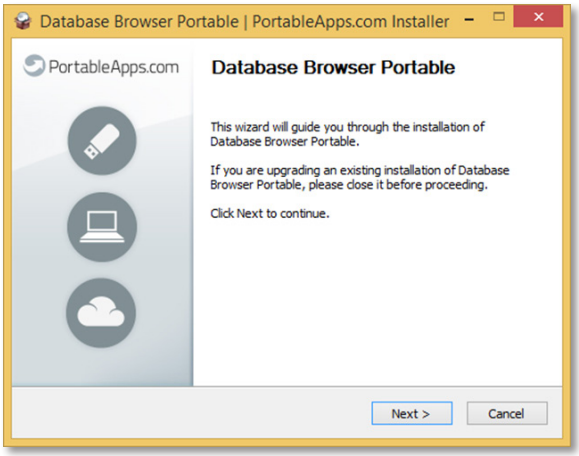


Figura 2. Tela inicial instalador Dadabase Browser Portable.

Na tela a seguir (Figura 3), informe onde deseja instalar o programa e clique em **Install** para copiar os arquivos do software para o diretório indicado.

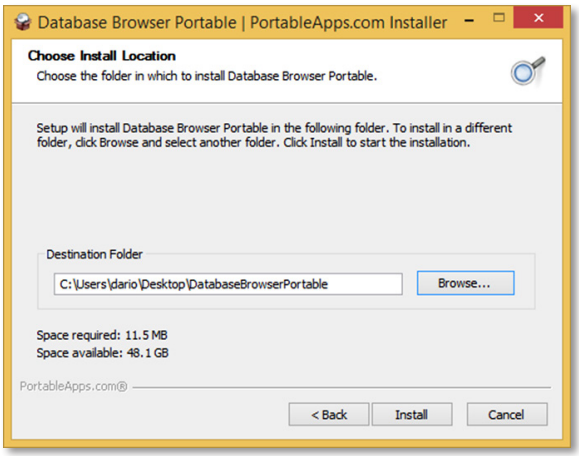


Figura 3. Local de instalação do Database Browser.

Terminando essa etapa, a seguinte tela será apresentada (Figura 4). Nela, é possível selecionar a opção "*Run Database Browser Portable*" para executar o *Database Browser* assim que terminar a instalação, ao clicar em **Finish**.

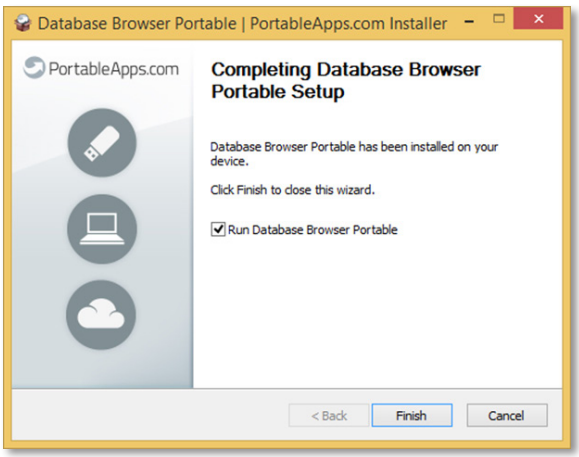


Figura 4. Tela final da instalação do Database Browser.

Dentro do diretório escolhido para instalação, haverá o subdiretório *DatabaseBrowserPortable* (Figura 5) e, nele, terá o programa **DatabaseBrowserPortable.exe**. Nesse ponto, o programa já está disponível. Por se tratar de um arquivo executável, orientamos passar o antivírus antes de executá-lo.

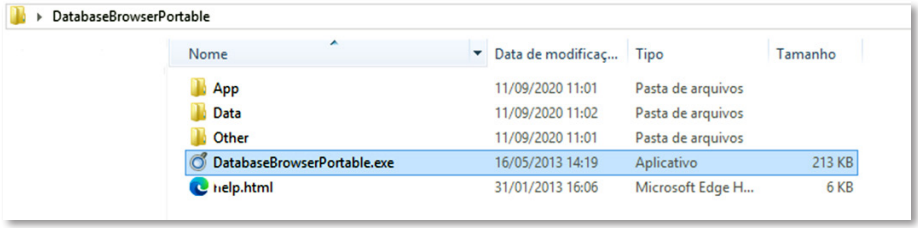


Figura 5. Diretório onde foi instalado o Database Browser.

Anexo V. Instalação do programa R.

Para executar a instalação do programa R em sua máquina, na versão mais atual, acesse o site oficial do software: https://www.r-project.org/ . A seguir, para dar sequência ao download, clique em **CRAN** (Figura 1).

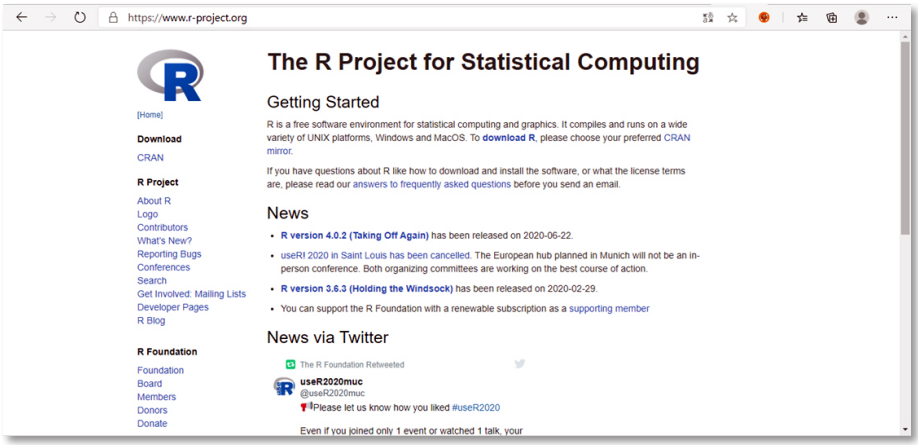


Figura 1. Site oficial do programa R.

Em seguida, deve-se escolha um dos servidores mais próximos da sua cidade para a transferência do arquivo (Figura 2). No caso do Brasil, há diversos servidores disponíveis, logo, a sua escolha vai depender do mais próximo encontrado a esquerda do servidor. Aqui, foi escolhido o servidor https://cran.fiocruz.br/.

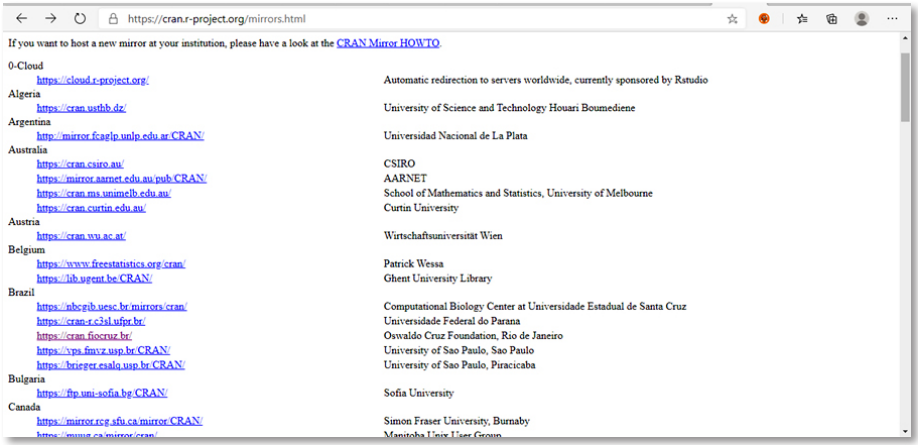


Figura 2. Lista dos servidores disponíveis para a transferência de arquivo.

O próximo passo depende do sistema operacional utilizado em sua máquina. Para fins de exemplo, foi selecionado a opção "**Download** R **for Windows**", conforme é mostrado na Figura 3.

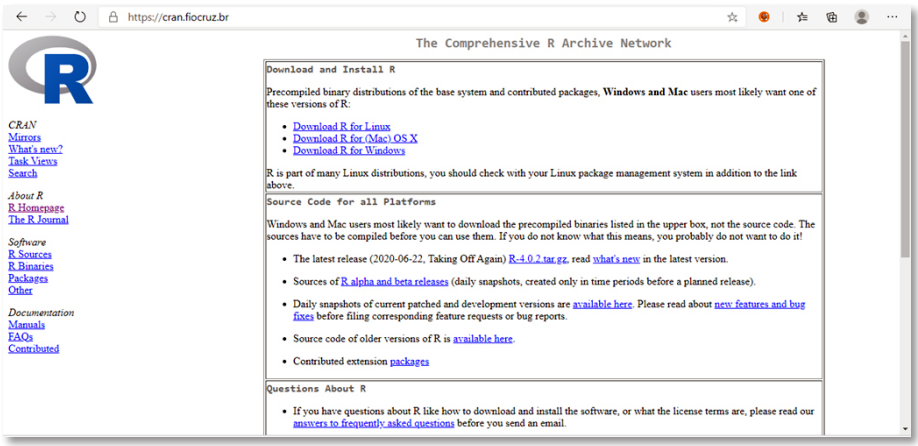


Figura 3. Definição do sistema operacional da máquina em que será instalada.

Para dar prosseguimento à etapa de acesso ao arquivo principal de instalação, clique em “**base**” (Figura 4).

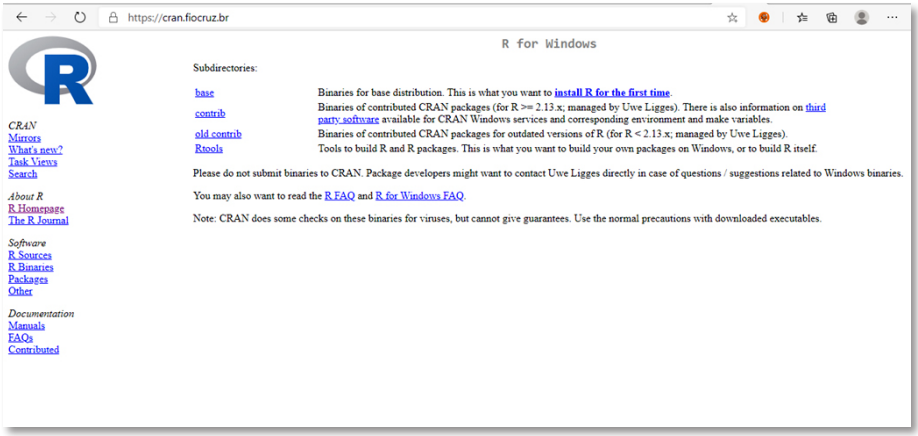


Figura 4. Acesso ao arquivo principal de instalação.

Na página de instalação, clique no link “**Download R 4.0.2 for Windows**” (na numeração que indica a versão do programa R disponível mais atualizada) (Figura 5).

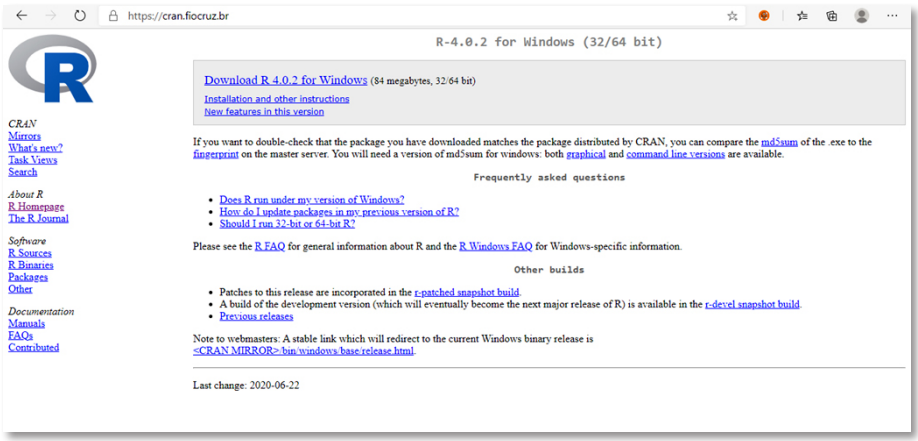


Figura 5. Link do Download R 4.0.2 for Windows.

Após esse procedimento, o instalador do programa R será salvo na pasta de downloads indicada. Uma vez concluído o download, clique em **R-4.0.2-win.exe** para instalar o programa R.

Ao iniciar o instalador, selecione o idioma de instalação e clique em **OK**, conforme é mostrado na Figura 6.

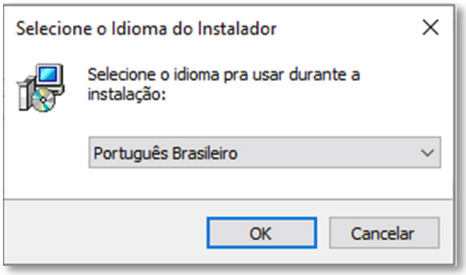


Figura 6. Seleção do idioma de instalação.

Surgirá a primeira tela do programa de instalação. Clique em **Próximo** (Figura 7).

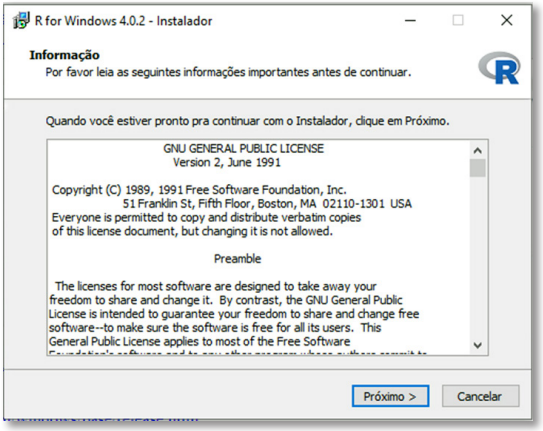


Figura 7. *GNU General Public License*.

A seguir, escolha o diretório em que será instalado o programa R e clique em **Próximo** (Figura 8).

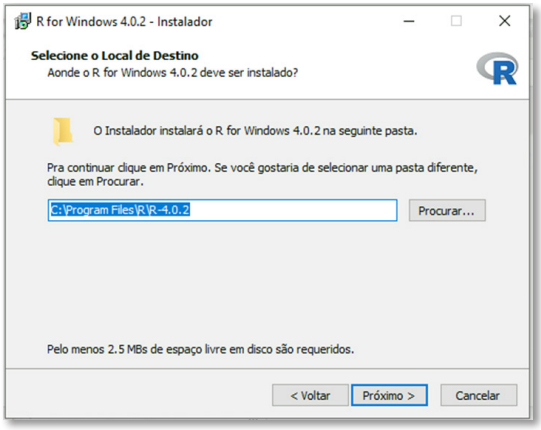


Figura 8. Defina o diretório de instalação.

Nesta etapa, você deverá instalar os componentes necessários para o bom funcionamento do programa R. Orienta-se, por questão de segurança, deixar todos os itens marcados e clicar em **Próximo**. (Figura 9).

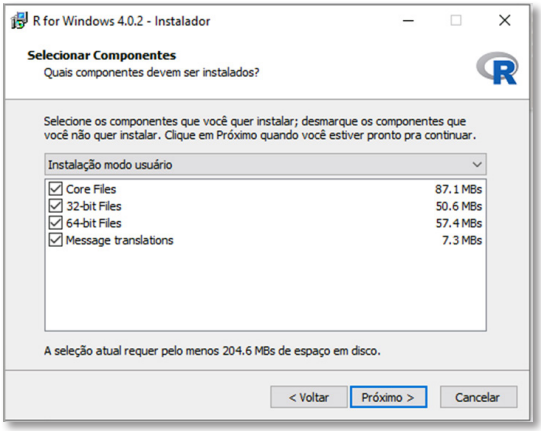


Figura 9. Componentes necessários para o funcionamento do programa R.

Nesta etapa, será questionado se prefere personalizar as opções de inicialização. Se restar alguma dúvida sobre isso, orienta-se aceitar o padrão. Para tanto, clique em **Não (aceitar padrão)** e depois em **Próximo** (Figura 10).

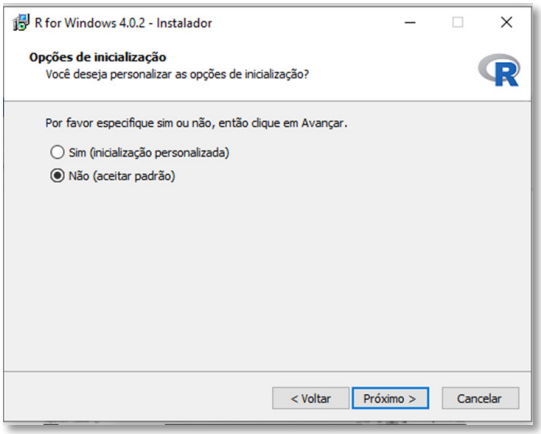


Figura 10. Opções de inicialização.

Neste momento, você poderá escolher colocar ícones na área de trabalho e na barra de inicialização do Windows, caso queira. Obrigatoriamente, orienta-se deixar selecionado as opções **Salvar número da versão no registro** e **Associar arquivos .RData ao R**. Clique em **Próximo** (Figura 11).

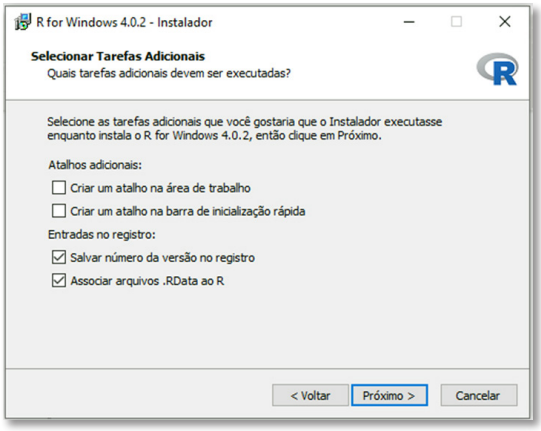


Figura 11. Definição de atalhos adicionais e entradas no registro.

Após proceder todos os passos anteriores de configuração, a instalação do programa R será iniciada. Aguarde enquanto o instalador instale o programa (Figura 12).

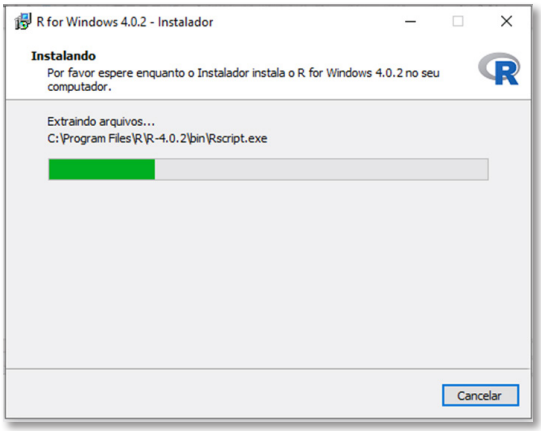


Figura 12. Instalação propriamente dita do programa R.

O aparecimento da tela, conforme mostra na Figura 13, indica que o programa R foi instalado com sucesso em seu computador. Basta clicar em **Concluir**.

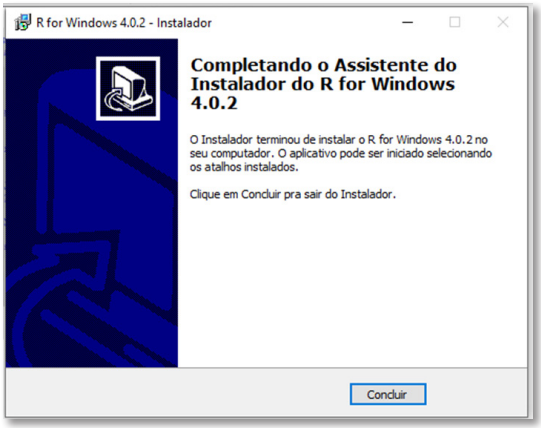


Figura 13. Conclusão do processo de instalação do programa R.

Embrapa

Cerrados

MINISTÉRIO DA
AGRICULTURA, PECUÁRIA
E ABASTECIMENTO

PÁTRIA AMADA
BRASIL
GOVERNO FEDERAL

CGPE 016836