PostgreSQL Explained for R-Users and R-Programmers

Ben Gonzalez

2019-10-11

Contents

Chapter 1

Prerequisites

For anyone interested in using this book you will need the following packages and tools to follow along.

- RPostgresql
- DBI
- Your own PostgreSQL database
- Remote access to your database

The **RPostgreSQL and DBI** package can be installed from CRAN or Github:

```r
install.packages("RPostgreSQL")
install.packages("DBI")
devtools::install_git('https://github.com/r-dbi/DBI.git')
devtools::install_git('https://github.com/cran/RPostgreSQL.git)
```

Notes on using SQL syntax in RPostgreSQL

To successfully query data in PostgreSQL the following caveats may be necessary. This is especially the case if someone has created column names that are unique and odd in some form or fashion.

- use backslahes to escape the quotes “” that are necessary when querying data.
- use quotes “” around camel back title cases e.g. **“Medu”** or **“Fedu”**

Chapter 2

Example data to follow along with

You can also download the example datasets and place them in your own PostgreSQL database.

Detroit Dataset: 14 Columns with 13 Observations

This is the data set called DETROIT' in the bookSubset selection in regression' by Alan J. Miller published in the Chapman & Hall series of monographs on Statistics & Applied Probability, no. 40. The data are unusual in that a subset of three predictors can be found which gives a very much better fit to the data than the subsets found from the Efroymson stepwise algorithm, or from forward selection or backward elimination. The original data were given in appendix A of 'Regression analysis and its application: A data-oriented approach' by Gunst & Mason, Statistics textbooks and monographs no. 24, Marcel Dekker. It has caused problems because some copies of the Gunst & Mason book do not contain all of the data, and because Miller does not say which variables he used as predictors and which is the dependent variable. (HOM was the dependent variable, and the predictors were FTP … WE)

Source: http://lib.stat.cmu.edu/datasets/detroit

Create Detroit Table in PostgreSQL

The necessary files to create the Detroit table in PostgreSQL can be found in the following link

Download Detroit data

Student Performance Data Set: A data frame with 392 rows and 33 variables:

This data approach looks at student achievement in secondary education of two Portuguese schools. The data attributes include student grades, demographics, social and school related features and it was collected utilizing school reports and questionnaires. Two datasets are provided regarding the performance in two distinct subjects: Mathematics (mat) and Portuguese language (por). In [Cortez and Silva, 2008], the two datasets were modeled under binary/five-level classification and regression tasks. Important note: the target attribute G3 has a strong correlation with attributes G2 and G1. This occurs because G3 is the final year grade (issued at the 3rd period), While G1 and G2 correspond to the 1st and 2nd period grades. It is more difficult to predict G3 without G2 and G1, but such prediction is much more useful (see paper source for more details)

Source: http://archive.ics.uci.edu/ml/datasets/Student+Performance

---

Create Student Table in PostgreSQL

The necessary files to create the Student table in PostgreSQL can be found in the following link

Download Student data

Chapter 3

Introduction

This is a reference book on how to user PostgreSQL in Rstudio utilizing the DBI and RPostgresql packages.

Note: All queries are limited to ten rows to allow for easier reading and understanding.

After searching the internet for exstensive books I was unable to find anything to my liking. Working with databases is key to get things done in R, Rstudio, Python, and R-Shiny. So I wanted to write my own book aimed at practical knowledge on how to do things and hopefully create a good work out of the hodgepodge of junk that is out there.

First things first. You will want to ensure that you have enabled remote access to your PostgreSQL database.

How to allow remote access to PostgreSQL database:

You will need to change some configurations in the postgresql.conf file on your server.

```
find \ -name "postgresql.conf"

sudo nano /var/lib/pgsql/PSQLVERSION/data/postgresql.conf
```

The you will want to change the following line listen_addresses = 'localhost' to listen_addresses = '*':

Search for it using CTRL + W

```
listen_addresses = 'localhost'
```

```
listen_addresses = '*'
```

Next restart your PostgreSQL database.

```
sudo systemctl postgresql restart
```

You should still receive an error as you also need to configure the pg_hba.conf file as well.

```
find \ -name "pg_hba.conf"

sudo nano /var/lib/pgsql/PSQLVERSION/data/pg_hba.conf
```

Now place the followin at the very end of the file.

```
host    all             all             0.0.0.0/0                               md5
host    all             all             ::/0                                    md5
```

https://blog.bigbinary.com/2016/01/23/configure-postgresql-to-allow-remote-connection.html

Step 1: Install the necessary packages to check the connection.

```
library(RPostgreSQL)
library(DBI)
```

Step 2: We want to connect to our PostgreSQL database itself. I recommend utilizing Digital Ocean to host your own cloud base PostgreSQL instance. Here is a link to a tutorial on their website to build your own if you have not done so before. Digital Ocean PostgreSQL. In the below code chunk you will want to update the repsective values with the values from your database instance.

```
library(DBI)
library(RPostgreSQL)
DBI::dbDriver('PostgreSQL')
require(RPostgreSQL)
drv=dbDriver("PostgreSQL")
con=dbConnect(drv,dbname=dbname,host=dbhost,port=5432,user=dbuser,password=dbpassword)
```

- **db** name will be the database you are wanting to use.
- **host** will be the host your database is on. Either your localhost or the url to your database.
- **port** By default the port will be 5432 for postgresql.
- **user** will be the username for the database you are connecting to
- **password** will be the database password you use when connecting to postgresql Next we can list the tables that are available in our database

This is the DBI way to do it in Rstudio.

```
dbListTables(conn = con)
```

```
## [1] "teachers" "dodgers"  "student"  "detroit"
```

This is the SQL syntax way to do it. Here we can see the tablename,tableowner, and the tablespace along with other housekeeping items that may be of interest to us.

```sql
SELECT * FROM pg_catalog.pg_tables;
```

Displaying records 1 - 10

schemaname

tablename

tableowner

tablespace

hasindexes

hasrules

hastriggers

rowsecurity

public

teachers

ben

NA

FALSE

FALSE

FALSE

FALSE

pg_catalog

pg_statistic

postgres

NA

TRUE

FALSE

FALSE

FALSE

pg_catalog

pg_type

postgres

```
NA
TRUE
FALSE
FALSE
FALSE
public
dodgers
ben
NA
FALSE
FALSE
FALSE
FALSE
public
student
ben
NA
FALSE
FALSE
FALSE
FALSE
pg_catalog
pg_policy
postgres
NA
TRUE
FALSE
FALSE
FALSE
pg_catalog
pg_authid
```

postgres

pg_global

TRUE

FALSE

FALSE

FALSE

public

detroit

ben

NA

FALSE

FALSE

FALSE

FALSE

pg_catalog

pg_user_mapping

postgres

NA

TRUE

FALSE

FALSE

FALSE

pg_catalog

pg_subscription

postgres

pg_global

TRUE

FALSE

FALSE

FALSE

Select * FROM table;

Here we are querying the entire table and bringing back all of the values.

Teachers Dataset

```
SELECT * FROM teachers
```

6 records

id

first_name

last_name

school

hire_date

salary

1

Janet

Smith

F.D. Roosevelt HS

2011-10-30

36200

2

Lee

Reynolds

F.D. Roosevelt HS

1993-05-22

65000

3

Samuel

Cole

Myers Middle School

2005-08-01

43500

4

Samantha

Bush

Myers Middle School

2011-10-30

36200

5

Betty

Diaz

Myers Middle School

2005-08-30

43500

6

Kathleen

Roush

F.D. Roosevelt HS

2010-10-22

38500

Detroit Dataset

```sql
SELECT * FROM detroit
```

Displaying records 1 - 10

Year

FTP

UEMP

MAN

LIC

GR

CLEAR

WM

NMAN

GOV

HE

WE

HOM

ACC

ASR

1961

320

11.0

455.5

178.50

215.98

93.4

558724

538.1

133.9

2.98

117.18

8.60

39.17

306.18

1962

320

7.0

480.2

156.41

180.48

88.5

538584

547.6

137.6

3.09

134.02

8.90

40.27

315.16

1963

320

5.2

506.1

198.02

209.57

94.4

519171

562.8

143.6

3.23

141.68

8.52

45.31

277.53

1964

320

4.3

535.8

222.10

231.67

92.0

500457

591.0

150.3

3.33

147.98

8.89

49.51

234.07

1965

320

3.5

576.0

301.92

297.65

91.0

482418

626.1

164.3

3.46

159.85

13.07

55.05

230.84

1966

320

3.2

601.7

391.22

367.62

87.4

465029

659.8

179.5

3.60

157.19

14.57

53.90

217.99

1967

320

4.1

577.3

665.56

616.54

88.3

448267

686.2

187.5

3.73

155.29

21.36

50.62

286.11

1968

320

3.9

596.9

1131.21

1029.75

86.1

432109

699.6

195.4

2.91

131.75

28.03

51.47

291.59

1969

320

3.6

613.5

837.60

786.23

79.0

416533

729.9

210.3

4.25

178.74

31.49

49.16

320.39

1970

320

7.1

569.3

794.90

713.77

73.9

401518

757.8

223.8

4.47

178.30

37.39

45.80

323.03

Student Dataset

```sql
SELECT * FROM student
```

Displaying records 1 - 10

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

18

U

GT3

A

4

4

at_home

teacher

course

mother

2

2

0

yes

no

no

no

yes

yes

no

no

4

3

4

1

1

3

6

5

6

6

GP

F

17

U

GT3

T

1

1

at_home

other

course

father

1

2

0

no

yes

no

no

no

yes

yes

no

5

3

3

1

1

3

4

5

5

6

GP

F

15

U

LE3

T

1

1

at_home

other

other

mother

1

2

3

yes

no

yes

no

yes

yes

yes

no

4

3

2

2

3

3

10

7

8

10

GP

F

15

U

GT3

T

4

2

health

services

home

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

yes

3

2

2

1

1

5

2

15

14

15

GP

F

16

U

GT3

T

3

3

other

other

home

father

1

2

0

no

yes

yes

no

yes

yes

no

no

4

3

2

1

2

5

4

6

10

10

GP

M

16

U

LE3

T

4

3

services

other

reputation

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

4

2

1

2

5

10

15

15

15

GP

M

16

U

LE3

T

2

2

other

other

home

mother

1

2

0

no

no

no

no

yes

yes

yes

no

4

4

4

1

1

3

0

12

12

11

GP

F

17

U

GT3

A

4

4

other

teacher

home

mother

2

2

0

yes

yes

no

no

yes

yes

no

no

4

1

4

1

1

1

6

6

5

6

GP

M

15

U

LE3

A

3

2

services

other

home

mother

1

2

0

no

yes

yes

no

yes

yes

yes

no

4

2

2

1

1

1

0

16

18

19

GP

M

15

U

GT3

T

3

4

other

other

home

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

5

1

1

1

5

0

14

15

15

Next we can list the tables that are available in our database

This is the DBI way to do it in Rstudio.

```r
DBI::dbListTables(conn = con)
```

```
## [1] "teachers" "dodgers"  "student"  "detroit"
```

The following is the SQL syntax way to do it. Here we can see the *tablename,tableowner*, and the *tablespace* along with other housekeeping items that may be of interest to us.

```
SELECT * FROM pg_catalog.pg_tables;
```

Displaying records 1 - 10

schemaname

tablename

tableowner

tablespace

hasindexes

hasrules

hastriggers

rowsecurity

public

teachers

ben

NA

FALSE

FALSE

FALSE

FALSE

pg_catalog

pg_statistic

postgres

NA

TRUE

FALSE

FALSE

FALSE

pg_catalog

pg_type

```
postgres
NA
TRUE
FALSE
FALSE
FALSE
public
dodgers
ben
NA
FALSE
FALSE
FALSE
FALSE
public
student
ben
NA
FALSE
FALSE
FALSE
FALSE
pg_catalog
pg_policy
postgres
NA
TRUE
FALSE
FALSE
FALSE
pg_catalog
```

pg_authid

postgres

pg_global

TRUE

FALSE

FALSE

FALSE

public

detroit

ben

NA

FALSE

FALSE

FALSE

FALSE

pg_catalog

pg_user_mapping

postgres

NA

TRUE

FALSE

FALSE

FALSE

pg_catalog

pg_subscription

postgres

pg_global

TRUE

FALSE

FALSE

FALSE

Chapter 4

Select * FROM table;

Here we are querying the entire table and bringing back all of the values.

```
## <PostgreSQLDriver>
```

Teachers

```sql
SELECT * FROM teachers
```

6 records

id

first_name

last_name

school

hire_date

salary

1

Janet

Smith

F.D. Roosevelt HS

2011-10-30

36200

2

Lee

Reynolds

F.D. Roosevelt HS

1993-05-22

65000

3

Samuel

Cole

Myers Middle School

2005-08-01

43500

4

Samantha

Bush

Myers Middle School

2011-10-30

36200

5

Betty

Diaz

Myers Middle School

2005-08-30

43500

6

Kathleen

Roush

F.D. Roosevelt HS

2010-10-22

38500

Detroit

```sql
SELECT * FROM detroit
```

Displaying records 1 - 10

Year

FTP

UEMP

MAN

LIC

GR

CLEAR

WM

NMAN

GOV

HE

WE

HOM

ACC

ASR

1961

320

11.0

455.5

178.50

215.98

93.4

558724

538.1

133.9

2.98

117.18

8.60

39.17

306.18

1962

320

7.0

480.2

156.41

180.48

88.5

538584

547.6

137.6

3.09

134.02

8.90

40.27

315.16

1963

320

5.2

506.1

198.02

209.57

94.4

519171

562.8

143.6

3.23

141.68

8.52

45.31

277.53

1964

320

4.3

535.8

222.10

231.67

92.0

500457

591.0

150.3

3.33

147.98

8.89

49.51

234.07

1965

320

3.5

576.0

301.92

297.65

91.0

482418

626.1

164.3

3.46

159.85

13.07

55.05

230.84

1966

320

3.2

601.7

391.22

367.62

87.4

465029

659.8

179.5

3.60

157.19

14.57

53.90

217.99

1967

320

4.1

577.3

665.56

616.54

88.3

448267

686.2

187.5

3.73

155.29

21.36

50.62

286.11

1968

320

3.9

596.9

1131.21

1029.75

86.1

432109

699.6

195.4

2.91

131.75

28.03

51.47

291.59

1969

320

3.6

613.5

837.60

786.23

79.0

416533

729.9

210.3

4.25

178.74

31.49

49.16

320.39

1970

320

7.1

569.3

794.90

713.77

73.9

401518

757.8

223.8

4.47

178.30

37.39

45.80

323.03

Student

```sql
SELECT * FROM student
```

Displaying records 1 - 10

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

18

U

GT3

A

4

4

at_home

teacher

course

mother

2

2

0

yes

no

no

no

yes

yes

no

no

4

3

4

1

1

3

6

5

6

6

GP

F

17

U

GT3

T

1

1

at_home

other

course

father

1

2

0

no

yes

no

no

no

yes

yes

no

5

3

3

1

1

3

4

5

5

6

GP

F

15

U

LE3

T

1

1

at_home

other

other

mother

1

2

3

yes

no

yes

no

yes

yes

yes

no

4

3

2

2

3

3

10

7

8

10

GP

F

15

U

GT3

T

4

2

health

services

home

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

yes

3

2

2

1

1

5

2

15

14

15

GP

F

16

U

GT3

T

3

3

other

other

home

father

1

2

0

no

yes

yes

no

yes

yes

no

no

4

3

2

1

2

5

4

6

10

10

GP

M

16

U

LE3

T

4

3

services

other

reputation

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

4

2

1

2

5

10

15

15

15

GP

M

16

U

LE3

T

2

2

other

other

home

mother

1

2

0

no

no

no

no

yes

yes

yes

no

4

4

4

1

1

3

0

12

12

11

GP

F

17

U

GT3

A

4

4

other

teacher

home

mother

2

2

0

yes

yes

no

no

yes

yes

no

no

4

1

4

1

1

1

6

6

5

6

GP

M

15

U

LE3

A

3

2

services

other

home

mother

1

2

0

no

yes

yes

no

yes

yes

yes

no

4

2

2

1

1

1

0

16

18

19

GP

M

15

U

GT3

T

3

4

other

other

home

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

5

1

1

```
1

5

0

14

15

15
```

Chapter 5

General Queries

Here we will make a few queries to the database that are general in nature. General queries are ones where we want to select particular columns and also where we want to remove or delete items from the database.

5.1 Distinct Queries

Here we will use distinct to look at the distinct values in a particular column. This allows us to get a high-level overview of what our data looks like.

Teachers

```sql
SELECT distinct school from teachers;
```

2 records

school

Myers Middle School

F.D. Roosevelt HS

Student

```sql
select distinct school from student
```

2 records

school

MS

GP

Next we can order our data in a particular way as well.

Notice that there is something very peculiar about this SQL statement. If we write it out as a normal SQL statement it will not work.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select Medu from student;')
```

```
## Error in postgresqlExecStatement(conn, statement, ...) :
##   RS-DBI driver: (could not Retrieve the result : ERROR:  column "medu" does not ex
## LINE 1: select Medu from student;
##                ^
## HINT:  Perhaps you meant to reference the column "student.Medu" or the column "stude
## )
```

```
## Warning in postgresqlQuickSQL(conn, statement, ...): Could not create
## execute: select Medu from student;
```

```
## NULL
```

Instead we are required to utilize quotes around the column names since it is Camelbacked: e.g. Medu vs. medu

```
RPostgreSQL::dbGetQuery(conn = con, statement = "select distinct \"Medu\" from student
```

```
##   Medu
## 1    0
## 2    1
## 3    3
## 4    2
## 5    4
```

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select "school","G3" from student orde
```

```
##    school G3
## 1      GP 20
## 2      GP 19
## 3      GP 19
## 4      MS 19
## 5      GP 19
## 6      GP 19
## 7      GP 18
## 8      GP 18
## 9      GP 18
## 10     GP 18
```

5.2 WHERE Queries

Where Clause in SQL

Here we are select the schools and the G3 grade where G3 is greater than 15.

```
RPostgreSQL::dbGetQuery(conn = con, statement = "select \"school\",\"G3\" from student where \"G3
```

```
##    school G3
## 1      GP 19
## 2      GP 16
## 3      GP 16
## 4      GP 17
## 5      GP 16
## 6      GP 18
## 7      GP 18
## 8      GP 20
## 9      GP 16
## 10     GP 16
```

Here we are select the schools and the G3 grade where G3 is equal to 4. Utilizing the operators >,<,= allows us to filter our data and retrieve the data we want to look at.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select "school","G3" from student where "G3"=4;')
```

5.3 AND, OR, NOT Queries

SQL Queries with **And, OR, NOT**.

AND

Here we only return one row where G3 = 4 and Medu = 4.

```
select * from student where "G3"=4 AND "Medu"=4;
```

1 records

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

17

U

GT3

T

4

3

other

other

reputation

mother

1

2

2

no

no

yes

no

yes

yes

yes

yes

3

4

5

2

4

1

22

6

6

4

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

17

U

GT3

T

4

3

other

other

reputation

mother

1

2

2

no

no

yes

no

yes

yes

yes

yes

3

4

5

2

4

1

22

6

6

4

OR

Here we return 10 rows where G3 = 4 or Mother's Education = 4. This helps us to filter and sort data when we want to find something in particular.

```
select * from student where "G3"=4 OR "Medu"=4 LIMIT 10;
```

Displaying records 1 - 10

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

18

U

GT3

A

4

4

at_home

teacher

course

mother

2

2

0

yes

no

no

no

yes

yes

no

no

4

3

4

1

1

3

6

5

6

6

GP

F

15

U

GT3

T

4

2

health

services

home

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

yes

3

2

2

1

1

5

2

15

14

15

GP

M

16

U

LE3

T

4

3

services

other

reputation

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

4

2

1

2

5

10

15

15

15

GP

F

17

U

GT3

A

4

4

other

teacher

home

mother

2

2

0

yes

yes

no

no

```
yes
yes
no
no
4
1
4
1
1
1
6
6
5
6
GP
F
15
U
GT3
T
4
4
teacher
health
reputation
mother
1
2
0
no
yes
```

yes

no

yes

yes

yes

no

3

3

3

1

2

2

0

10

8

9

GP

M

15

U

LE3

T

4

4

health

services

course

father

1

1

0

```
no
yes
yes
yes
yes
yes
yes
no
4
3
3
1
3
5
2
14
14
14
GP
M
15
U
GT3
T
4
3
teacher
other
course
mother
2
```

2

0

no

yes

yes

no

yes

yes

yes

no

5

4

3

1

2

3

2

10

10

11

GP

F

16

U

GT3

T

4

4

health

other

home

mother

1

1

0

no

yes

no

no

yes

yes

yes

no

4

4

4

1

2

2

4

14

14

14

GP

F

16

U

GT3

T

4

4

services

services

reputation

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

no

3

2

3

1

2

2

6

13

14

14

GP

M

16

U

LE3

T

4

3

health

other

home

father

1

1

0

no

no

yes

yes

yes

yes

yes

no

3

1

3

1

3

5

4

8

10

10

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from student where "G3"=4 OR "Medu"=4 LI
```

```
##    school sex age address famsize Pstatus Medu Fedu     Mjob     Fjob
## 1      GP   F  18       U     GT3       A    4    4  at_home  teacher
## 2      GP   F  15       U     GT3       T    4    2   health services
## 3      GP   M  16       U     LE3       T    4    3 services    other
## 4      GP   F  17       U     GT3       A    4    4    other  teacher
## 5      GP   F  15       U     GT3       T    4    4  teacher   health
## 6      GP   M  15       U     LE3       T    4    4   health services
## 7      GP   M  15       U     GT3       T    4    3  teacher    other
## 8      GP   F  16       U     GT3       T    4    4   health    other
## 9      GP   F  16       U     GT3       T    4    4 services services
## 10     GP   M  16       U     LE3       T    4    3   health    other
##        reason guardian traveltime studytime failures schoolsup famsup paid
## 1      course   mother          2         2        0       yes     no   no
## 2        home   mother          1         3        0        no    yes  yes
## 3  reputation   mother          1         2        0        no    yes  yes
## 4        home   mother          2         2        0       yes    yes   no
## 5  reputation   mother          1         2        0        no    yes  yes
## 6      course   father          1         1        0        no    yes  yes
## 7      course   mother          2         2        0        no    yes  yes
## 8        home   mother          1         1        0        no    yes   no
## 9  reputation   mother          1         3        0        no    yes  yes
## 10       home   father          1         1        0        no     no  yes
##    activities nursery higher internet romantic famrel freetime goout Dalc
## 1          no     yes    yes       no       no      4        3     4    1
## 2         yes     yes    yes      yes      yes      3        2     2    1
## 3         yes     yes    yes      yes       no      5        4     2    1
## 4          no     yes    yes       no       no      4        1     4    1
## 5          no     yes    yes      yes       no      3        3     3    1
## 6         yes     yes    yes      yes       no      4        3     3    1
## 7          no     yes    yes      yes       no      5        4     3    1
## 8          no     yes    yes      yes       no      4        4     4    1
## 9         yes     yes    yes      yes       no      3        2     3    1
## 10        yes     yes    yes      yes       no      3        1     3    1
##    Walc health absences G1 G2 G3
## 1     1      3        6  5  6  6
## 2     1      5        2 15 14 15
## 3     2      5       10 15 15 15
## 4     1      1        6  6  5  6
## 5     2      2        0 10  8  9
## 6     3      5        2 14 14 14
## 7     2      3        2 10 10 11
## 8     2      2        4 14 14 14
## 9     2      2        6 13 14 14
## 10    3      5        4  8 10 10
```

NOT

Here we are looking at results where Fathers Education (Fedu) does not equal 4.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from student WHERE NOT "Fedu"=
select * from student WHERE NOT "Fedu"=4 LIMIT 10;
```

Displaying records 1 - 10

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

17

U

GT3

T

1

1

at_home

other

course

father

1

2

0

no

yes

no

no

no

yes

yes

no

5

3

3

```
1
1
3
4
5
5
6
GP
F
15
U
LE3
T
1
1
at_home
other
other
mother
1
2
3
yes
no
yes
no
yes
yes
yes
no
4
```

3

2

2

3

3

10

7

8

10

GP

F

15

U

GT3

T

4

2

health

services

home

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

```
yes
3
2
2
1
1
5
2
15
14
15
GP
F
16
U
GT3
T
3
3
other
other
home
father
1
2
0
no
yes
yes
no
yes
```

yes

no

no

4

3

2

1

2

5

4

6

10

10

GP

M

16

U

LE3

T

4

3

services

other

reputation

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

4

2

1

2

5

10

15

15

15

GP

M

16

U

LE3

T

2

2

other

other

home

mother

1

2

0

no

no

no

no

yes

yes

yes

no

4

4

4

1

1

3

0

12

12

11

GP

M

15

U

LE3

A

3

2

services

other

home

mother

1

2

0

no

yes

yes

no

yes

yes

yes

no

4

2

2

1

1

1

0

16

18

19

GP

F

15

U

GT3

T

2

1

services

other

reputation

father

```
3
3
0
no
yes
no
yes
yes
yes
yes
no
5
2
2
1
1
4
4
10
12
12
GP
M
15
U
GT3
T
4
3
teacher
other
```

course

mother

2

2

0

no

yes

yes

no

yes

yes

yes

no

5

4

3

1

2

3

2

10

10

11

GP

M

15

U

GT3

A

2

2

other

other

home

other

1

3

0

no

yes

no

no

yes

yes

yes

yes

4

5

2

1

1

3

0

14

16

16

Combining AND, OR, NOT

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from student where "G3">=10 O
```

```
select * from student where "G3">=10 OR "Medu"=4 LIMIT 10;
```

Displaying records 1 - 10

school

sex

age

address

famsize

Pstatus

Medu

Fedu

Mjob

Fjob

reason

guardian

traveltime

studytime

failures

schoolsup

famsup

paid

activities

nursery

higher

internet

romantic

famrel

freetime

goout

Dalc

Walc

health

absences

G1

G2

G3

GP

F

18

U

GT3

A

4

4

at_home

teacher

course

mother

2

2

0

yes

no

no

no

yes

yes

no

no

4

3

4

1

1

3

6

5

6

6

GP

F

15

U

LE3

T

1

1

at_home

other

other

mother

1

2

3

yes

no

yes

no

yes

yes

yes

no

4

3

2

2

3

3

10

7

8

10

GP

F

15

U

GT3

T

4

2

health

services

home

mother

1

3

0

no

yes

yes

yes

yes

yes

yes

yes

3

2

2

1

1

5

2

15

14

15

GP

F

16

U

GT3

T

3

3

other

other

home

father

1

2

0

no

yes

yes

no

yes

yes

no

no

4

3

2

1

2

5

4

6

10

10

GP

M

16

U

LE3

T

4

3

services

other

reputation

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

4

2

1

2

5

10

15

15

15

GP

M

16

U

LE3

T

2

2

other

other

home

mother

1

2

0

no

no

no

no

yes

yes

yes

no

4

4

4

1

1

3

0

12

12

11

GP

F

17

U

GT3

A

4

4

other

teacher

home

mother

2

2

0

yes

yes

no

no

yes

yes

no

no

4

1

4

1

1

1

6

6

5

6

GP

M

15

U

LE3

A

3

2

services

other

home

mother

1

2

0

no

yes

yes

no

yes

yes

yes

no

4

2

2

1

1

1

0

16

18

19

GP

M

15

U

GT3

T

3

4

other

other

home

mother

1

2

0

no

yes

yes

yes

yes

yes

yes

no

5

5

1

1

1

5

0

14

15

15

GP

F

15

U

GT3

T

4

4

teacher

health

reputation

mother

1

2

0

no

yes

yes

no

yes

yes

yes

no

3

3

3

1

2

2

0

10

8

9

The AND OR

Here we tell SQL that we want all the G3 grades that are > 10 and also that the school should be GP OR Fedu should equal 4. The backslashes allow us to escape the single quotes that are necessary when using RPostgresql syntax.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from student where "G3">=10 A
```

```
select "G3", school, "Fedu" from student where "G3">=10 AND school='GP' OR "Fedu"=4 LI
```

Displaying records 1 - 10

G3

school

Fedu

6

GP

4

10

GP

1

15

GP

2

10

GP

3

15

GP

3

11

GP

2

6

GP

4

19

GP

2

15

GP

4

9

GP

4

The double NOT or NOT NOT

Here we tell SQL that we want to return all values where Fedu and Medu are not equal to 4.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from student WHERE NOT "Medu"=
```

```
##    school sex age address famsize Pstatus Medu Fedu     Mjob    Fjob
## 1      GP   F  18       U     GT3       A    4    4  at_home teacher
## 2      GP   F  17       U     GT3       T    1    1  at_home   other
## 3      GP   F  15       U     LE3       T    1    1  at_home   other
## 4      GP   F  16       U     GT3       T    3    3    other   other
## 5      GP   M  16       U     LE3       T    2    2    other   other
## 6      GP   F  17       U     GT3       A    4    4    other teacher
## 7      GP   M  15       U     LE3       A    3    2 services   other
## 8      GP   M  15       U     GT3       T    3    4    other   other
## 9      GP   F  15       U     GT3       T    4    4  teacher  health
## 10     GP   F  15       U     GT3       T    2    1 services   other
##        reason guardian traveltime studytime failures schoolsup famsup paid
## 1      course   mother          2         2        0       yes     no   no
## 2      course   father          1         2        0        no    yes   no
## 3       other   mother          1         2        3       yes     no  yes
## 4        home   father          1         2        0        no    yes  yes
## 5        home   mother          1         2        0        no     no   no
## 6        home   mother          2         2        0       yes    yes   no
## 7        home   mother          1         2        0        no    yes  yes
## 8        home   mother          1         2        0        no    yes  yes
## 9  reputation   mother          1         2        0        no    yes  yes
## 10 reputation   father          3         3        0        no    yes   no
##    activities nursery higher internet romantic famrel freetime goout Dalc
## 1          no     yes    yes       no       no      4        3     4    1
## 2          no      no    yes      yes       no      5        3     3    1
## 3          no     yes    yes      yes       no      4        3     2    2
## 4          no     yes    yes       no       no      4        3     2    1
## 5          no     yes    yes      yes       no      4        4     4    1
## 6          no     yes    yes       no       no      4        1     4    1
## 7          no     yes    yes      yes       no      4        2     2    1
## 8         yes     yes    yes      yes       no      5        5     1    1
## 9          no     yes    yes      yes       no      3        3     3    1
## 10        yes     yes    yes      yes       no      5        2     2    1
##    Walc health absences G1 G2 G3
## 1     1      3        6  5  6  6
## 2     1      3        4  5  5  6
## 3     3      3       10  7  8 10
## 4     2      5        4  6 10 10
## 5     1      3        0 12 12 11
## 6     1      1        6  6  5  6
## 7     1      1        0 16 18 19
## 8     1      5        0 14 15 15
## 9     2      2        0 10  8  9
## 10    1      4        4 10 12 12
```

5.4 Insert Queries

Insert into PostgreSQL using RPostgreSQL

- Inserting a single list of values into PostgreSQL.

Ok, now lets INSERT some data into our PostgreSQL database. We will want to develop a query string and send this to the database via dbSendQuery() from the RPostgreSQL package.

```
query <- ('INSERT INTO detroit VALUES (1974,265, 14, 500.5, 200.5, 215.98, 93.457, 558724, 538.12

table<- RPostgreSQL::dbSendQuery(conn = con,statement = query)
```

Then when we call the new data we can see that we have updated the row (observations) to 14 and have added the data in our query.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from detroit;')
```

```
##    Year FTP UEMP   MAN     LIC      GR CLEAR     WM  NMAN   GOV   HE
## 1  1961 320 11.0 455.5  178.50  215.98  93.4 558724 538.1 133.9 2.98
## 2  1962 320  7.0 480.2  156.41  180.48  88.5 538584 547.6 137.6 3.09
## 3  1963 320  5.2 506.1  198.02  209.57  94.4 519171 562.8 143.6 3.23
## 4  1964 320  4.3 535.8  222.10  231.67  92.0 500457 591.0 150.3 3.33
## 5  1965 320  3.5 576.0  301.92  297.65  91.0 482418 626.1 164.3 3.46
## 6  1966 320  3.2 601.7  391.22  367.62  87.4 465029 659.8 179.5 3.60
## 7  1967 320  4.1 577.3  665.56  616.54  88.3 448267 686.2 187.5 3.73
## 8  1968 320  3.9 596.9 1131.21 1029.75  86.1 432109 699.6 195.4 2.91
## 9  1969 320  3.6 613.5  837.60  786.23  79.0 416533 729.9 210.3 4.25
## 10 1970 320  7.1 569.3  794.90  713.77  73.9 401518 757.8 223.8 4.47
## 11 1971 320  8.4 548.8  817.74  750.43  63.4 387046 755.3 227.7 5.04
## 12 1972 320  7.7 563.4  583.17 1027.38  62.5 373095 787.0 230.9 5.47
## 13 1973 320  6.3 609.3  709.59  666.50  58.9 359647 819.8 230.2 5.76
##        WE   HOM   ACC    ASR
## 1  117.18  8.60 39.17 306.18
## 2  134.02  8.90 40.27 315.16
## 3  141.68  8.52 45.31 277.53
## 4  147.98  8.89 49.51 234.07
## 5  159.85 13.07 55.05 230.84
## 6  157.19 14.57 53.90 217.99
## 7  155.29 21.36 50.62 286.11
## 8  131.75 28.03 51.47 291.59
## 9  178.74 31.49 49.16 320.39
## 10 178.30 37.39 45.80 323.03
## 11 209.54 46.26 44.54 357.38
## 12 240.05 47.24 41.03 422.07
## 13 258.05 52.33 44.17 473.01
```

5.5 Update Queries

The **UPDATE** command allows us to update the records in a table to new data. We may want to do this in our Shiny Applications or from the Rstudio console itself. Here we will update the last column that we inserted into the **Detroit** table as we had an error in column 2.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'UPDATE detroit SET "FTP"=320;')
```

```
## data frame with 0 columns and 0 rows
```

Now let's check our results.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'select * from detroit;')
```

```
##    Year FTP UEMP   MAN     LIC      GR CLEAR     WM  NMAN   GOV   HE
## 1  1961 320 11.0 455.5  178.50  215.98  93.4 558724 538.1 133.9 2.98
## 2  1962 320  7.0 480.2  156.41  180.48  88.5 538584 547.6 137.6 3.09
## 3  1963 320  5.2 506.1  198.02  209.57  94.4 519171 562.8 143.6 3.23
## 4  1964 320  4.3 535.8  222.10  231.67  92.0 500457 591.0 150.3 3.33
## 5  1965 320  3.5 576.0  301.92  297.65  91.0 482418 626.1 164.3 3.46
## 6  1966 320  3.2 601.7  391.22  367.62  87.4 465029 659.8 179.5 3.60
## 7  1967 320  4.1 577.3  665.56  616.54  88.3 448267 686.2 187.5 3.73
## 8  1968 320  3.9 596.9 1131.21 1029.75  86.1 432109 699.6 195.4 2.91
## 9  1969 320  3.6 613.5  837.60  786.23  79.0 416533 729.9 210.3 4.25
## 10 1970 320  7.1 569.3  794.90  713.77  73.9 401518 757.8 223.8 4.47
## 11 1971 320  8.4 548.8  817.74  750.43  63.4 387046 755.3 227.7 5.04
## 12 1972 320  7.7 563.4  583.17 1027.38  62.5 373095 787.0 230.9 5.47
## 13 1973 320  6.3 609.3  709.59  666.50  58.9 359647 819.8 230.2 5.76
##        WE   HOM   ACC    ASR
## 1  117.18  8.60 39.17 306.18
## 2  134.02  8.90 40.27 315.16
## 3  141.68  8.52 45.31 277.53
## 4  147.98  8.89 49.51 234.07
## 5  159.85 13.07 55.05 230.84
## 6  157.19 14.57 53.90 217.99
## 7  155.29 21.36 50.62 286.11
## 8  131.75 28.03 51.47 291.59
## 9  178.74 31.49 49.16 320.39
## 10 178.30 37.39 45.80 323.03
## 11 209.54 46.26 44.54 357.38
## 12 240.05 47.24 41.03 422.07
## 13 258.05 52.33 44.17 473.01
```

5.6 Delete Queries

We can also delete specific rows based on the data of one or more columns. This will allow us to remove mistakes we have made in our tables without having to **DROP** the entire table itself.

```
RPostgreSQL::dbGetQuery(conn = con,statement = 'DELETE FROM detroit WHERE "Year"=1974;')
```

```
## data frame with 0 columns and 0 rows
```

Chapter 6

RPostgreSQL in Shiny Applications

Here we will show you how to use RPostgreSQL within your R-Shiny or Shiny application. This can be somewhat frustrating as you will need to take advantage of the paste() and paste0() base commands in R to send your text or numeric input data to the query itself.

```
paste(x = name ,sep = "-",collapse = "")

paste0()
```

6.1 Insert Query from Shiny Application

For us to send a query to the PostgreSQL database we will need to paste the query together. We will begin by constructing the usual **INSERT INTO TABLENAME** and then insert the respective values into the query as well. We will need to utilize the **input**$value1 * *replacingwhatever * *input$**name** we have for the value we want to insert into the databale when the query is sent. We can simply paste this into a new variable called **qry** and assign it and pass this **qry** to the **dbSendQuery** function in our application. As the number of columns increase so will the number of values as well.

```
      qry = paste0("INSERT INTO table (column1,column2)",
                   "VALUES ('",paste(input$value1,"'",",","'",input$value2,"')"))

    dbSendQuery(conn = con, statement = qry)
```

Note: We have each value surrounded by the single back ticks ’ and

also we have the values seperated by a comma as well.

6.2 Write Table Query from Shiny Application to the Database

Here you can take a dataframe and write it directly to the database using the **dbWriteTable** command. This allows us to write a dataframe directlyt to the table in question. You will need the dataframe column names to match the ones that are in the table in the database. The number of columns must match as well. You will not be able to write a dataframe with more columns than are in the table, but you will be able to write a dataframe that has less columns than are in the table in the database.

```
dbWriteTable(conn = con,name = 'table_name',value = table_value)

RPostgreSQL::postgresqlWriteTable(con = con,name = 'table_name',value = table_value,ove

RPostgreSQL::postgresqlWriteTable(con = con,name = 'table_name',value = table_value,app
```

Above we have 3 distinct ways to write the data to the table in the database. The last two allow us to either **overwrite** or **append** the data to the table. Depending on our application needs we will be able to do one or the other.