Subsetting, dplyr, magrittr

Author: Lloyd Low; Email add: wai.low@adelaide.edu.au

Introduction

So you have got a table with data that might be a mixed of categorical, integer, numeric, etc variables? And it comes with thousands of columns and millions of rows? You wish you can manipulate your table in 'powerful' ways beyond just the simple preconfigured operations that some software has already set for you, which does not allow you to ask questions in a different way? Below is a potential solution using R, which is open source and that means it's 'almost' free as long as you adhere to the license conditions.

Interesting fact! Did you know what Microsoft Excel limits are? 1,048,576 rows by 16,384 columns

- Outline of topics
  - Logical Vectors
  - Subsetting using square brackets
  - dplyr - Who created it? Hadley Wickham and Romain Francois
  - dplyr - What is it? Grammar for manipulating tabular data
  - dplyr functions: select(), filter(), arrange(), mutate(), group_by(), summarise()
  - Pipe a series of dplyr functions using the magrittr (%>%) package
  - magrittr is by Stefan Milton Bache

Subsetting

```
#subset a vector#
x <- c("a", "b", "c", "d", "a", "b", "d")
x[1]
```

```
## [1] "a"
```

```
x[1:5]
```

```
## [1] "a" "b" "c" "d" "a"
```

```
x[x>"a"]
```

```
## [1] "b" "c" "d" "b" "d"
```

```
boolean <- x > "a"
boolean
```

```
## [1] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE
```

```
x[boolean]
```

```
## [1] "b" "c" "d" "b" "d"
```

```
#removal of NAs
z <- c(1,2,NA,4,NA,6,7,8,9)
bad <- is.na(z)
z[!bad]
```

```
## [1] 1 2 4 6 7 8 9
```

```
z <- c(1,2,NA,4,NA,6,7,8,9)
y <- c("a","b",NA,"d",NA,"f","g","h","i")
good <- complete.cases(z,y)
z[good]
```

```
## [1] 1 2 4 6 7 8 9
```

```
y[good]
```

```
## [1] "a" "b" "d" "f" "g" "h" "i"
```

```
#subset a data frame#
setwd("/Users/lloyd/Documents/lloyd_2017/Services/R_RoseworthyWkshp_git/Roseworthy-R-20170501/data")

#read in the data frame
tooth <- read.csv("toothData.csv")

#Check for missing data
tooth.na <- is.na(tooth$len)
sum(tooth.na)
```

```
## [1] 0
```

```
#Alternative way to subset
tooth.na <- is.na(tooth[,1])
sum(tooth.na)
```

```
## [1] 0
```

```
#how to select just the columns 'len' and 'dose'?
tooth.len.dose <- tooth[,c(1,3)]

#how to filter for a particular row? say tooth length more than mean tooth length?
tooth.len.abv.mean <- tooth[tooth$len > mean(tooth$len),]
```

dplyr

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#library(data.table) #Use this package if the data frame is very big

#Convert it to a class that dplyr can work on
tooth <- tbl_df(tooth)
#observe that it is class tibble
class(tooth)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```
#five verbs (technically called functions in R)
#select(), filter(), arrange(), mutate(), and summarize()

select(tooth,len,dose)
```

```
## # A tibble: 60 × 2
##      len   dose
##    <dbl> <fctr>
## 1    4.2    Low
## 2   11.5    Low
## 3    7.3    Low
## 4    5.8    Low
## 5    6.4    Low
## 6   10.0    Low
## 7   11.2    Low
## 8   11.2    Low
## 9    5.2    Low
## 10   7.0    Low
## # ... with 50 more rows
```

```
#dplyr is 'clever' enough to display just the first 10
#rows of data and columns that can fit neatly in the console.
#For missed columns due to limited console viewing space,
#we see the names and classes for the variables at the bottom.

#select() will output according to the order of variables we specify
select(tooth,dose,len)
```

```
## # A tibble: 60 × 2
##      dose   len
##    <fctr> <dbl>
## 1     Low   4.2
## 2     Low  11.5
## 3     Low   7.3
## 4     Low   5.8
## 5     Low   6.4
## 6     Low  10.0
## 7     Low  11.2
## 8     Low  11.2
## 9     Low   5.2
## 10    Low   7.0
## # ... with 50 more rows
```

```
#using ':' operator, can select a range of variables
select(tooth,len:dose)
```

```
## # A tibble: 60 × 3
##      len   supp   dose
##    <dbl> <fctr> <fctr>
## 1    4.2     VC    Low
## 2   11.5     VC    Low
## 3    7.3     VC    Low
## 4    5.8     VC    Low
## 5    6.4     VC    Low
```

```
## 6   10.0     VC    Low
## 7   11.2     VC    Low
## 8   11.2     VC    Low
## 9    5.2     VC    Low
## 10   7.0     VC    Low
## # ... with 50 more rows
```

```
#using '-' operator to throw away unwanted columns
select(tooth,-dose)
```

```
## # A tibble: 60 × 2
##      len   supp
##    <dbl> <fctr>
## 1    4.2     VC
## 2   11.5     VC
## 3    7.3     VC
## 4    5.8     VC
## 5    6.4     VC
## 6   10.0     VC
## 7   11.2     VC
## 8   11.2     VC
## 9    5.2     VC
## 10   7.0     VC
## # ... with 50 more rows
```

```
#filter specific rows
filter(tooth, len == 10)
```

```
## # A tibble: 2 × 3
##     len   supp   dose
##   <dbl> <fctr> <fctr>
## 1    10     VC    Low
## 2    10     OJ    Low
```

```
?Comparison

#arrange len in ascending order
arrange(tooth, len)
```

```
## # A tibble: 60 × 3
##      len   supp   dose
##    <dbl> <fctr> <fctr>
## 1    4.2     VC    Low
## 2    5.2     VC    Low
## 3    5.8     VC    Low
## 4    6.4     VC    Low
## 5    7.0     VC    Low
## 6    7.3     VC    Low
## 7    8.2     OJ    Low
## 8    9.4     OJ    Low
## 9    9.7     OJ    Low
## 10   9.7     OJ    Low
## # ... with 50 more rows
```

```
#Suppose the tooth length is given in mm, and you want a len column in cm
mutate(tooth, len.cm = len / 10)
```

```
## # A tibble: 60 × 4
##      len   supp   dose len.cm
##    <dbl> <fctr> <fctr>  <dbl>
## 1    4.2     VC    Low   0.42
## 2   11.5     VC    Low   1.15
## 3    7.3     VC    Low   0.73
## 4    5.8     VC    Low   0.58
## 5    6.4     VC    Low   0.64
## 6   10.0     VC    Low   1.00
## 7   11.2     VC    Low   1.12
## 8   11.2     VC    Low   1.12
## 9    5.2     VC    Low   0.52
## 10   7.0     VC    Low   0.70
## # ... with 50 more rows
```

```
#group_by and summarise
tooth.supp <- group_by(tooth,supp,dose)
summarise(tooth.supp,mean(len))
```

```
## Source: local data frame [6 x 3]
## Groups: supp [?]
##
##     supp   dose `mean(len)`
##   <fctr> <fctr>       <dbl>
## 1     OJ   High       26.06
## 2     OJ    Low       13.23
## 3     OJ    Med       22.70
## 4     VC   High       26.14
## 5     VC    Low        7.98
## 6     VC    Med       16.77
```

Exercise

```
#Exercise
#read in ChickWeightNew.csv and make the data okay for dplyr to work with

#select just weight and Chick columns from the dataframe/tibble

#remove observations that have weight equals to "NA"

#remove observations that have weight equals to "NA" and keep column "Diet" equals to 1

#filter rows with Time = 21 Or Diet = 2

#arrange the ChickWeight dataframe according to Time in ascending order

#arrange the ChickWeight dataframe according to Time in ascending order and
#Diet in descending order

#filter Time = 10, save dataframe as a new one,
#create a new variable that is the normalized weight,
#i.e. (weight - mean(weight))/sd(weight)
```

```
#remove all NA from weight, filter for last day = 21, group by diet, summarize mean diet
```

Answers

```
#read in ChickWeight
setwd("/Users/lloyd/Documents/lloyd_2017/Services/R_RoseworthyWkshp_git/Roseworthy-R-20170501/data")
test <- read.csv("ChickWeightNew.csv")
library(dplyr)
test <- tbl_df(test)

#select just weight and Chick
select(test,weight, Chick)
```

```
## # A tibble: 583 × 2
##    weight Chick
##     <int> <int>
## 1      42     1
## 2      51     1
## 3      59     1
## 4      64     1
## 5      76     1
## 6      93     1
## 7     106     1
## 8     125     1
## 9     149     1
## 10    171     1
## # ... with 573 more rows
```

```
#remove rows with "NA" for weight
filter(test, weight != "NA")
```

```
## # A tibble: 578 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1      42     0     1     1
## 2      51     2     1     1
## 3      59     4     1     1
## 4      64     6     1     1
## 5      76     8     1     1
## 6      93    10     1     1
## 7     106    12     1     1
## 8     125    14     1     1
## 9     149    16     1     1
## 10    171    18     1     1
## # ... with 568 more rows
```

```
#Alternative way
filter(test, !is.na(weight))
```

```
## # A tibble: 578 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1      42     0     1     1
```

```
## 2      51     2     1     1
## 3      59     4     1     1
## 4      64     6     1     1
## 5      76     8     1     1
## 6      93    10     1     1
## 7     106    12     1     1
## 8     125    14     1     1
## 9     149    16     1     1
## 10    171    18     1     1
## # ... with 568 more rows
```

```
#remove rows that don't have "NA" and the "Diet" is equal to 1
#combine filter based on AND logic
filter(test, weight != "NA",Diet == 1)
```

```
## # A tibble: 220 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1      42     0     1     1
## 2      51     2     1     1
## 3      59     4     1     1
## 4      64     6     1     1
## 5      76     8     1     1
## 6      93    10     1     1
## 7     106    12     1     1
## 8     125    14     1     1
## 9     149    16     1     1
## 10    171    18     1     1
## # ... with 210 more rows
```

```
#Time = 21 Or Diet = 2
#use OR logic |
filter(test, Time == 21 | Diet == 2)
```

```
## # A tibble: 157 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1     205    21     1     1
## 2     215    21     2     1
## 3     202    21     3     1
## 4     157    21     4     1
## 5     223    21     5     1
## 6     157    21     6     1
## 7     305    21     7     1
## 8      NA    21     8     1
## 9      98    21     9     1
## 10    124    21    10     1
## # ... with 147 more rows
```

```
#arrange the DF according to Time in ascending order
arrange(test, Time)
```

```
## # A tibble: 583 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1      42     0     1     1
```

```
## 2      40     0     2     1
## 3      43     0     3     1
## 4      42     0     4     1
## 5      41     0     5     1
## 6      41     0     6     1
## 7      41     0     7     1
## 8      42     0     8     1
## 9      42     0     9     1
## 10     41     0    10     1
## # ... with 573 more rows
```

```
#can do multiple variable arrangement
#arrange the DF according to Time in ascending order and Diet in descending order
arrange(test, Time,desc(Diet))
```

```
## # A tibble: 583 × 4
##    weight  Time Chick  Diet
##     <int> <int> <int> <int>
## 1      42     0    41     4
## 2      42     0    42     4
## 3      42     0    43     4
## 4      42     0    44     4
## 5      41     0    45     4
## 6      40     0    46     4
## 7      41     0    47     4
## 8      39     0    48     4
## 9      40     0    49     4
## 10     41     0    50     4
## # ... with 573 more rows
```

```
#mutate - create new variable based on existing ones
#filter only Time = 10, save DF, create a new variable that is the normalized
#weight, i.e. (weight - mean(weight))/sd(weight)
test.10 <- filter(test,Time == 10)
mutate(test.10,normalizedWeight = (weight - mean(weight))/sd(weight))
```

```
## # A tibble: 49 × 5
##    weight  Time Chick  Diet normalizedWeight
##     <int> <int> <int> <int>            <dbl>
## 1      93    10     1     1       -0.6185252
## 2     103    10     2     1       -0.2016375
## 3      99    10     3     1       -0.3683926
## 4      87    10     4     1       -0.8686578
## 5     106    10     5     1       -0.0765712
## 6     124    10     6     1        0.6738266
## 7     112    10     7     1        0.1735614
## 8      93    10     8     1       -0.6185252
## 9      96    10     9     1       -0.4934589
## 10     81    10    10     1       -1.1187904
## # ... with 39 more rows
```

```
#group_by and summarize
#remove NA, filter for last day = 21, group by diet, summarize mean diet
test <- filter(test, !is.na(weight))
test.lastday <- filter(test,Time == 21)
test.lastday.diet <- group_by(test.lastday,Diet)
```

```
summarise(test.lastday.diet,mean(weight))
```

```
## # A tibble: 4 × 2
##    Diet `mean(weight)`
##   <int>          <dbl>
## 1     1       177.7500
## 2     2       214.7000
## 3     3       270.3000
## 4     4       238.5556
```

magrittr %>%

```
setwd("/Users/lloyd/Documents/lloyd_2017/Services/R_RoseworthyWkshp_git/Roseworthy-R-20170501/data")
test <- read.csv("ChickWeightNew.csv")
library(dplyr)
library(magrittr)
test <- tbl_df(test)

#filter only Time = 10, save DF, create a new variable that is the normalized
#weight, i.e. (weight - mean(weight))/sd(weight)
test %>% filter(Time == 10) %>% mutate(normalizedWeight = (weight - mean(weight))/sd(weight))
```

```
## # A tibble: 49 × 5
##    weight  Time Chick  Diet normalizedWeight
##     <int> <int> <int> <int>            <dbl>
## 1      93    10     1     1       -0.6185252
## 2     103    10     2     1       -0.2016375
## 3      99    10     3     1       -0.3683926
## 4      87    10     4     1       -0.8686578
## 5     106    10     5     1       -0.0765712
## 6     124    10     6     1        0.6738266
## 7     112    10     7     1        0.1735614
## 8      93    10     8     1       -0.6185252
## 9      96    10     9     1       -0.4934589
## 10     81    10    10     1       -1.1187904
## # ... with 39 more rows
```

```
#group_by and summarize
#remove NA, filter for last day = 21, group by diet, summarize mean diet
test %>% filter(!is.na(weight),Time == 21) %>% group_by(Diet) %>% summarise(mean(weight))
```

```
## # A tibble: 4 × 2
##    Diet `mean(weight)`
##   <int>          <dbl>
## 1     1       177.7500
## 2     2       214.7000
## 3     3       270.3000
## 4     4       238.5556
```