Five Page dplyr

Andy Grogan-Kaylor

2019-06-30

Contents

1 Background 1
2 Simulated Data 1
3 Piping 2
4 Aggregate Data: group_by() & summarise() 2
5 Select A Subset of Variables: select() 2
6 Filter A Subset of Rows: filter() 2
7 Create New Variables: mutate() 3
8 Recode Variables: mutate() 3
8.1 Continuous Into Categorical: mutate() & cut() 3
8.2 Categorical Into Categorical: mutate() & recode() 3
9 Rename Variables: rename() 4
10 Drop Missing Values: filter() 4
11 Random Sample 4
12 Connecting To Other Packages Like ggplot 5

1 Background

dplyr is a very powerful R library for managing and processing data.1

While dplyr is very powerful, learning to use dplyr can be very confusing. This guide aims to present some of the most common dplyr functions and commands in the form of a brief cheat sheet.

library(dplyr)

2 Simulated Data

<table>
<thead>
<tr>
<th>year</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>NA</td>
<td>Group A</td>
<td>101.2</td>
</tr>
<tr>
<td>2018</td>
<td>34.32</td>
<td>Group C</td>
<td>93.75</td>
</tr>
<tr>
<td>2017</td>
<td>31.45</td>
<td>Group A</td>
<td>103.3</td>
</tr>
</tbody>
</table>

1 The origins of the name dplyr seem somewhat obscure, but I sometimes think of this package as the data plyers.
3 Piping

Pipes `%>%` connect pieces of a command e.g. `data` to `data wrangling` to a graph command.

4 Aggregate Data: `group_by()` & `summarise()`

```r
mynewdata <- mydata %>%
  group_by(year) %>%  # group by y
  summarise(mean_x = mean(x), # mean of x
             n = n()) # count up
```

```
   year mean_x n
  2015   NA  1
  2016 41.51  1
  2017 37.75  2
  2018 34.32  1
```

5 Select A Subset of Variables: `select()`

```r
mynewdata <- mydata %>% select(x, y) # select only x and y
```

```
  x    y
NA Group A
34.32 Group C
31.45 Group A
44.05 Group B
41.51 Group B
```

6 Filter A Subset of Rows: `filter()`

```r
mynewdata <- mydata %>% filter(year > 2010) # filter on year
```
<table>
<thead>
<tr>
<th>year</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>myscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>NA</td>
<td>Group A</td>
<td>101.2</td>
<td>NA</td>
</tr>
<tr>
<td>2018</td>
<td>34.32</td>
<td>Group C</td>
<td>93.75</td>
<td>128.1</td>
</tr>
<tr>
<td>2017</td>
<td>31.45</td>
<td>Group A</td>
<td>103.3</td>
<td>134.8</td>
</tr>
<tr>
<td>2017</td>
<td>44.05</td>
<td>Group B</td>
<td>93.3</td>
<td>137.3</td>
</tr>
<tr>
<td>2016</td>
<td>41.51</td>
<td>Group B</td>
<td>101.3</td>
<td>142.8</td>
</tr>
</tbody>
</table>

7 Create New Variables: `mutate()`

```r
mynewdata <- mydata %>% mutate(myscale = x + z)  # create a new variable e.g. a scale
```

<table>
<thead>
<tr>
<th>year</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>myscale</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>NA</td>
<td>Group A</td>
<td>101.2</td>
<td>NA</td>
</tr>
<tr>
<td>2018</td>
<td>34.32</td>
<td>Group C</td>
<td>93.75</td>
<td>128.1</td>
</tr>
<tr>
<td>2017</td>
<td>31.45</td>
<td>Group A</td>
<td>103.3</td>
<td>134.8</td>
</tr>
<tr>
<td>2017</td>
<td>44.05</td>
<td>Group B</td>
<td>93.3</td>
<td>137.3</td>
</tr>
<tr>
<td>2016</td>
<td>41.51</td>
<td>Group B</td>
<td>101.3</td>
<td>142.8</td>
</tr>
</tbody>
</table>

8 Recode Variables: `mutate()`

8.1 Continuous Into Categorical: `mutate()` & `cut()`

```r
mynewdata <- mydata %>%
  mutate(zcategorical = cut(z,  # cut at breaks
    breaks=c(-Inf, 100, Inf),
    labels = c("low", "high")))
```

<table>
<thead>
<tr>
<th>year</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>zcategorical</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>NA</td>
<td>Group A</td>
<td>101.2</td>
<td>high</td>
</tr>
<tr>
<td>2018</td>
<td>34.32</td>
<td>Group C</td>
<td>93.75</td>
<td>low</td>
</tr>
<tr>
<td>2017</td>
<td>31.45</td>
<td>Group A</td>
<td>103.3</td>
<td>high</td>
</tr>
<tr>
<td>2017</td>
<td>44.05</td>
<td>Group B</td>
<td>93.3</td>
<td>low</td>
</tr>
<tr>
<td>2016</td>
<td>41.51</td>
<td>Group B</td>
<td>101.3</td>
<td>high</td>
</tr>
</tbody>
</table>

8.2 Categorical Into Categorical: `mutate()` & `recode()`

```r
mynewdata <- mydata %>%
  mutate(yrecoded = dplyr::recode(y,  # recode values
    "Group A" = "Red Group",
    "Group B" = "Blue Group",
    "Group C" = "Blue Group",
    default = "Other")
```

df <- df %>% rename(age = x, mental_health = y)

9 Rename Variables: rename()

df <- df %>% rename(age = x, mental_health = y)

10 Drop Missing Values: filter()

df <- df %>% filter(!is.na(x)) # filter by x is not missing

11 Random Sample

df <- df %>% sample_frac(0.5) # fraction of data to sample
12 Connecting To Other Packages Like ggplot

Notice how, in the code below, I never actually create the new data set mynewdata. I simply pipe mydata into a dplyr command, and pipe the result directly to ggplot2.

```r
library(ggplot2)

mydata %>% # my data
  mutate(myscale = x + z) %>% # dplyr command to make new variable
  ggplot(aes(x = year, # the rest is ggplot
             y =myscale)) +
  geom_point() + # points
  geom_smooth(se = FALSE) + # smoother without confidence interval
  labs(title = "My Scale By Year") + # labels
  theme(axis.text.x = element_text(size = 10, # tweak theme
                                angle = 90))
```

<table>
<thead>
<tr>
<th>year</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015</td>
<td>NA</td>
<td>Group A</td>
<td>101.2</td>
</tr>
</tbody>
</table>