Multi-Page dplyr

Andy Grogan-Kaylor

2021-01-05

Contents

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 cheatsheet.

[1] The origins of the name `dplyr` seem somewhat obscure, but I sometimes think of this package as the *data plyers*.

```r
library(dplyr)
```

2 Simulated Data

year	x	y	z
2016	NA	Group A	94.23
2017	46.32	Group C	108.4
2015	45.08	Group B	109.2

year	x	y	z
2015	47.77	Group A	95.43
2015	41.08	Group A	100.9

3 Piping

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

`dplyr` commands will often look something like the outline below.

```
mydata %>%
  data_wrangling %>%
  more_data_wrangling %>%
  graph_command
```

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

```
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	44.64	3
2016	NA	1
2017	46.32	1

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

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

x	y
NA	Group A
46.32	Group C
45.08	Group B
47.77	Group A
41.08	Group A

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

```
mynewdata <- mydata %>%
  filter(year > 2010) # filter on year
```

year	x	y	z
2016	NA	Group A	94.23
2017	46.32	Group C	108.4
2015	45.08	Group B	109.2
2015	47.77	Group A	95.43
2015	41.08	Group A	100.9

7 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2016	NA	Group A	94.23	NA
2017	46.32	Group C	108.4	154.7
2015	45.08	Group B	109.2	154.3
2015	47.77	Group A	95.43	143.2
2015	41.08	Group A	100.9	142

8 Recode Variables: `mutate()`

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

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

year	x	y	z	zcategorical
2016	NA	Group A	94.23	low
2017	46.32	Group C	108.4	high
2015	45.08	Group B	109.2	high
2015	47.77	Group A	95.43	low
2015	41.08	Group A	100.9	high

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

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

year	x	y	z	yrecoded
2016	NA	Group A	94.23	Red Group
2017	46.32	Group C	108.4	Other
2015	45.08	Group B	109.2	Blue Group
2015	47.77	Group A	95.43	Red Group
2015	41.08	Group A	100.9	Red Group

9 Rename Variables: `rename()`

```
newdata <- mydata %>%
  rename(age = x, # rename
         mental_health = z)
```

year	age	y	mental_health
2016	NA	Group A	94.23
2017	46.32	Group C	108.4
2015	45.08	Group B	109.2
2015	47.77	Group A	95.43
2015	41.08	Group A	100.9

10 Drop Missing Values: `filter()`

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

year	x	y	z
2017	46.32	Group C	108.4
2015	45.08	Group B	109.2
2015	47.77	Group A	95.43
2015	41.08	Group A	100.9

11 Random Sample

```
newdata <- mydata %>%
  sample_frac(.5) # fraction of data to sample
```

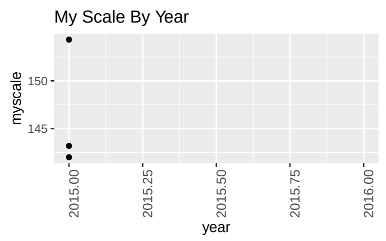
year	x	y	z
2017	46.32	Group C	108.4
2015	47.77	Group A	95.43

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`.

```
library(ggplot2)

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


My Scale By Year
myscale
year