Multi-Page dplyr

Andy Grogan-Kaylor

2020-02-11

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*.

```
library(dplyr)
```

2 Simulated Data

year	x	y	z
2018	NA	Group B	89.65
2016	28.85	Group C	108.4
2017	43.79	Group A	92.52

year	x	y	z
2018	60.96	Group A	114.3
2016	50.72	Group B	106

3 Piping

Pipes %>% connect pieces of a command e.g. *data* to *data wrangling* to a *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
2016	39.78	2
2017	43.79	1
2018	NA	2

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

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

x	y
NA	Group B
28.85	Group C
43.79	Group A
60.96	Group A
50.72	Group B

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

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

year	x	y	z
2018	NA	Group B	89.65
2016	28.85	Group C	108.4
2017	43.79	Group A	92.52
2018	60.96	Group A	114.3
2016	50.72	Group B	106

7 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2018	NA	Group B	89.65	NA
2016	28.85	Group C	108.4	137.2
2017	43.79	Group A	92.52	136.3
2018	60.96	Group A	114.3	175.2
2016	50.72	Group B	106	156.7

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
2018	NA	Group B	89.65	low
2016	28.85	Group C	108.4	high
2017	43.79	Group A	92.52	low
2018	60.96	Group A	114.3	high
2016	50.72	Group B	106	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
2018	NA	Group B	89.65	Blue Group
2016	28.85	Group C	108.4	Other
2017	43.79	Group A	92.52	Red Group
2018	60.96	Group A	114.3	Red Group
2016	50.72	Group B	106	Blue Group

9 Rename Variables: `rename()`

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

year	age	y	mental_health
2018	NA	Group B	89.65
2016	28.85	Group C	108.4
2017	43.79	Group A	92.52
2018	60.96	Group A	114.3
2016	50.72	Group B	106

10 Drop Missing Values: `filter()`

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

year	x	y	z
2016	28.85	Group C	108.4
2017	43.79	Group A	92.52
2018	60.96	Group A	114.3
2016	50.72	Group B	106

11 Random Sample

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

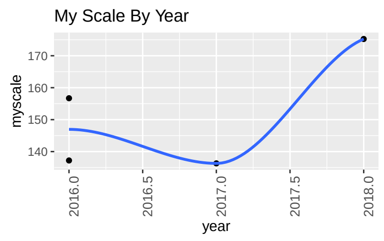
year	x	y	z
2017	43.79	Group A	92.52
2016	28.85	Group C	108.4

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
  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))
```


My Scale By Year
myscale
year