Five Page dplyr

Andy Grogan-Kaylor

2019-06-30

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
2015	NA	Group A	101.2
2018	34.32	Group C	93.75
2017	31.45	Group A	103.3

year	x	y	z
2017	44.05	Group B	93.3
2016	41.51	Group B	101.3

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
2015	NA	1
2016	41.51	1
2017	37.75	2
2018	34.32	1

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

```
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()`

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

year	x	y	z
2015	NA	Group A	101.2
2018	34.32	Group C	93.75
2017	31.45	Group A	103.3
2017	44.05	Group B	93.3
2016	41.51	Group B	101.3

7 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2015	NA	Group A	101.2	NA
2018	34.32	Group C	93.75	128.1
2017	31.45	Group A	103.3	134.8
2017	44.05	Group B	93.3	137.3
2016	41.51	Group B	101.3	142.8

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
2015	NA	Group A	101.2	high
2018	34.32	Group C	93.75	low
2017	31.45	Group A	103.3	high
2017	44.05	Group B	93.3	low
2016	41.51	Group B	101.3	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
2015	NA	Group A	101.2	Red Group
2018	34.32	Group C	93.75	Other
2017	31.45	Group A	103.3	Red Group
2017	44.05	Group B	93.3	Blue Group
2016	41.51	Group B	101.3	Blue Group

9 Rename Variables: `rename()`

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

year	age	y	mental_health
2015	NA	Group A	101.2
2018	34.32	Group C	93.75
2017	31.45	Group A	103.3
2017	44.05	Group B	93.3
2016	41.51	Group B	101.3

10 Drop Missing Values: `filter()`

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

year	x	y	z
2018	34.32	Group C	93.75
2017	31.45	Group A	103.3
2017	44.05	Group B	93.3
2016	41.51	Group B	101.3

11 Random Sample

```
newdata <- mydata %>% sample_frac(0.5)  # fraction of data to sample
```

year	x	y	z
2016	41.51	Group B	101.3

year	x	y	z
2015	NA	Group A	101.2

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))
```