Four Page dplyr

Andy Grogan-Kaylor

2019-04-03

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.

```
library(dplyr)
```

2 Simulated Data

year	x	y	z
2001	NA	Group A	102.3
2006	36.85	Group C	97.27
2006	37.89	Group B	96.19
2005	38.57	Group A	89.25
2001	54.81	Group B	96.19

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

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

x	y
NA	Group A
36.85	Group C
37.89	Group B
38.57	Group A
54.81	Group B

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

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

year	**x**	**y**	**z**

6 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2001	NA	Group A	102.3	NA
2006	36.85	Group C	97.27	134.1
2006	37.89	Group B	96.19	134.1
2005	38.57	Group A	89.25	127.8
2001	54.81	Group B	96.19	151

7 Recode Variables: `mutate()`

7.1 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
2001	NA	Group A	102.3	high
2006	36.85	Group C	97.27	low
2006	37.89	Group B	96.19	low
2005	38.57	Group A	89.25	low
2001	54.81	Group B	96.19	low

7.2 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
2001	NA	Group A	102.3	Red Group
2006	36.85	Group C	97.27	Other
2006	37.89	Group B	96.19	Blue Group
2005	38.57	Group A	89.25	Red Group
2001	54.81	Group B	96.19	Blue Group

8 Rename Variables: `rename()`

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

year	age	y	mental_health
2001	NA	Group A	102.3
2006	36.85	Group C	97.27
2006	37.89	Group B	96.19
2005	38.57	Group A	89.25
2001	54.81	Group B	96.19

9 Drop Missing Values: `filter()`

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

year	x	y	z
2006	36.85	Group C	97.27
2006	37.89	Group B	96.19
2005	38.57	Group A	89.25
2001	54.81	Group B	96.19

10 Random Sample

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

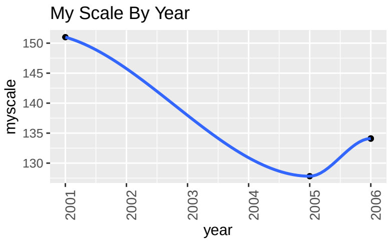
year	x	y	z
2001	54.81	Group B	96.19
2006	37.89	Group B	96.19

11 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