Five Page dplyr

Andy Grogan-Kaylor

2019-04-09

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
2013	NA	Group A	111.6
2012	39.8	Group C	92.99
2013	52.06	Group A	99.96

year	x	y	z
2005	38.14	Group B	109.7
2005	57.91	Group A	86.99

3 Piping

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

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

```
mynewdata <- mydata %>%
  group_by(y) %>% # group by y
  tally() # count up
```

y	n
Group A	3
Group B	1
Group C	1

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

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

x	y
NA	Group A
39.8	Group C
52.06	Group A
38.14	Group B
57.91	Group A

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

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

year	x	y	z
2013	NA	Group A	111.6
2012	39.8	Group C	92.99
2013	52.06	Group A	99.96

7 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2013	NA	Group A	111.6	NA
2012	39.8	Group C	92.99	132.8
2013	52.06	Group A	99.96	152
2005	38.14	Group B	109.7	147.9
2005	57.91	Group A	86.99	144.9

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
2013	NA	Group A	111.6	high
2012	39.8	Group C	92.99	low
2013	52.06	Group A	99.96	low
2005	38.14	Group B	109.7	high
2005	57.91	Group A	86.99	low

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
2013	NA	Group A	111.6	Red Group
2012	39.8	Group C	92.99	Other
2013	52.06	Group A	99.96	Red Group
2005	38.14	Group B	109.7	Blue Group
2005	57.91	Group A	86.99	Red Group

9 Rename Variables: `rename()`

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

year	age	y	mental_health
2013	NA	Group A	111.6
2012	39.8	Group C	92.99
2013	52.06	Group A	99.96
2005	38.14	Group B	109.7
2005	57.91	Group A	86.99

10 Drop Missing Values: `filter()`

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

year	x	y	z
2012	39.8	Group C	92.99
2013	52.06	Group A	99.96
2005	38.14	Group B	109.7
2005	57.91	Group A	86.99

11 Random Sample

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

year	x	y	z
2012	39.8	Group C	92.99
2005	57.91	Group A	86.99

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))
```