Multi-Page `dplyr`

Andy Grogan-Kaylor

2024-09-14

Table of 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) # data wrangling
```

2 Simulated Data

year	x	y	z
2019	NA	Group B	90
2020	50	Group C	90
2020	NA	Group A	100
2021	70	Group A	110
2021	80	Group B	120

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
```

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

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

Notice how when aggregating data, we need to be explicit about whether we are removing `NA` values.

```
mynewdata <- mydata %>%
  group_by(year) %>% # group by y
  summarise(mean_x = mean(x,
                          na.rm = TRUE), # mean of x; removing NA's
            n = n()) # count up
```

year	mean_x	n
2019	NA	1
2020	50	2
2021	75	2

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

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

x	y
NA	Group B
50	Group C
NA	Group A
70	Group A
80	Group B

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

```
mynewdata <- mydata %>%
  filter(year > 2020) # filter on year
```

year	x	y	z
2021	70	Group A	110
2021	80	Group B	120

7 Create New Variables: `mutate()`

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

year	x	y	z	myscale
2019	NA	Group B	90	NA
2020	50	Group C	90	140
2020	NA	Group A	100	NA
2021	70	Group A	110	180
2021	80	Group B	120	200

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
2019	NA	Group B	90	low
2020	50	Group C	90	low
2020	NA	Group A	100	low
2021	70	Group A	110	high
2021	80	Group B	120	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
2019	NA	Group B	90	Blue Group
2020	50	Group C	90	Other
2020	NA	Group A	100	Red Group
2021	70	Group A	110	Red Group
2021	80	Group B	120	Blue Group

9 Rename Variables: `rename()`

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

year	age	y	mental_health
2019	NA	Group B	90
2020	50	Group C	90
2020	NA	Group A	100
2021	70	Group A	110
2021	80	Group B	120

10 Drop Missing Values: `filter()`

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

year	x	y	z
2020	50	Group C	90
2021	70	Group A	110
2021	80	Group B	120

11 Random Sample

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

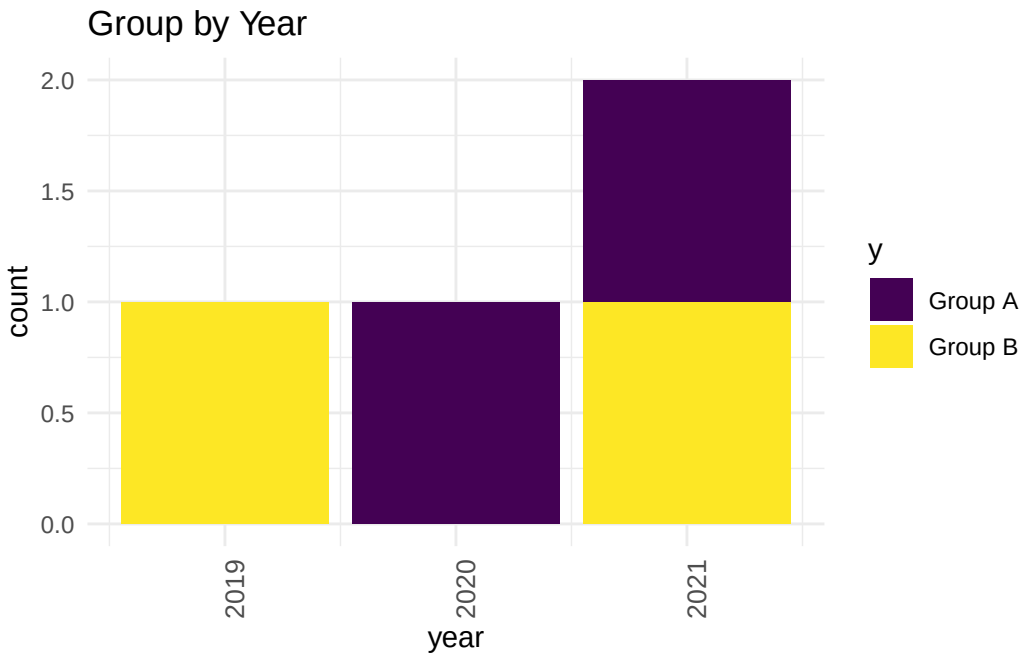
year	x	y	z
2020	50	Group C	90
2019	NA	Group B	90

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
             fill = y)) +
  geom_bar() +
  scale_fill_viridis_d(option = "viridis") + # viridis colors
  theme_minimal() + # minimal theme
  labs(title = "Group by Year") + # labels
  theme(axis.text.x = element_text(size = 10, # tweak theme
                                   angle = 90))
```


Group by Year
2.0
1.5
1.0
0.5
0.0
count
2019
2020
2021
year
y
Group A
Group B