**Introduction**

How we can recreate base equivalents of the dplyr functions select(), filter(), mutate() and arrange(), amongst others and they are presented in a new package called poorman. In this post let’s discuss new functionality that I have since added to poorman including grouped operations, renaming columns, summarising data and even poorman’s very own pipe operator!

**Group By**

**The base Way**

Typically in the past when I have performed grouped operations using base I have used the tried and tested split-apply-combine approach. That is, I split the data.frame on a grouping variable, I apply a function and then I combine the parts back together. Let’s take a look at what I mean using the mtcars dataset by splitting the data into groups representing each combination of transmission type and number of cylinders (see ?datasets::mtcars for more details about this dataset).

# Split the data into a list of data.frames - one for each group

split <- split(mtcars, list(mtcars$am, mtcars$cyl))

# Apply the mean function to each data.frame's mpg column

apply <- lapply(

split,

function(x) {

x[, "meanMpg"] <- mean(x$mpg)

x

}

)

# Stitch each of the lists back into one data.frame

combine <- do.call(rbind, unname(apply))

combine

# mpg cyl disp hp drat wt qsec vs am gear carb meanMpg

# Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2 22.90000

# Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2 22.90000

# Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1 22.90000

# Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1 28.07500

# Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1 28.07500

# Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2 28.07500

# Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1 28.07500

# Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1 28.07500

# Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2 28.07500

# Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2 28.07500

# Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2 28.07500

# Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1 19.12500

# Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1 19.12500

# Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4 19.12500

# Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4 19.12500

# Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4 20.56667

# Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4 20.56667

# Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6 20.56667

# Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2 15.05000

# Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4 15.05000

# Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3 15.05000

# Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3 15.05000

# Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3 15.05000

# Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4 15.05000

# Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4 15.05000

# Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4 15.05000

# Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2 15.05000

# AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2 15.05000

# Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4 15.05000

# Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2 15.05000

# Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4 15.40000

# Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8 15.40000

Note that here I unname() the list of data.frames to avoid rbind() prepending the row names with the group information. This is a really nice, flexible way to perform grouped operations using base. There are in fact many other ways that this type of operation can be achieved; the flexibility of base is what makes it so great.

**The poorman Way**

poorman aims to be a direct replacement for dplyr, the idea being that loading poorman and running your script written with dplyr code should return the same results. Therefore if you’re familiar with the dplyr API then the next piece of code should look very familiar.

library(poorman)

mtcars %>%

group\_by(am, cyl) %>%

mutate(meanMpg = mean(mpg))

# mpg cyl disp hp drat wt qsec vs am gear carb meanMpg

# Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2 22.90000

# Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2 22.90000

# Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1 22.90000

# Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1 28.07500

# Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1 28.07500

# Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2 28.07500

# Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1 28.07500

# Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1 28.07500

# Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2 28.07500

# Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2 28.07500

# Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2 28.07500

# Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1 19.12500

# Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1 19.12500

# Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4 19.12500

# Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4 19.12500

# Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4 20.56667

# Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4 20.56667

# Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6 20.56667

# Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2 15.05000

# Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4 15.05000

# Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3 15.05000

# Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3 15.05000

# Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3 15.05000

# Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4 15.05000

# Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4 15.05000

# Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4 15.05000

# Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2 15.05000

# AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2 15.05000

# Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4 15.05000

# Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2 15.05000

# Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4 15.40000

# Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8 15.40000

#

# Groups: am, cyl

In order to achieve this, I need to have a way to perform grouped operations for functions such as mutate() and filter(). The easiest solution I could think of therefore is an S3 class system. So the group\_by() function simply applies a "grouped\_data" class to the data and then the mutate() S3 generic knows to dispatch to the mutate.grouped\_data() method. For more information on the S3 class system I recommend checking out [Advanced R](https://adv-r.hadley.nz/s3.html).

**Ungrouping**

poorman also offers a replica of the dplyr::ungroup() function. Let’s take a look.

mtcars %>%

group\_by(am, cyl) %>%

mutate(meanMpg = mean(mpg)) %>%

ungroup(am)

# mpg cyl disp hp drat wt qsec vs am gear carb meanMpg

# Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2 22.90000

# Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2 22.90000

# Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1 22.90000

# Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1 28.07500

# Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1 28.07500

# Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2 28.07500

# Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1 28.07500

# Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1 28.07500

# Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2 28.07500

# Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2 28.07500

# Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2 28.07500

# Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1 19.12500

# Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1 19.12500

# Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4 19.12500

# Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4 19.12500

# Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4 20.56667

# Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4 20.56667

# Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6 20.56667

# Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2 15.05000

# Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4 15.05000

# Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3 15.05000

# Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3 15.05000

# Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3 15.05000

# Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4 15.05000

# Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4 15.05000

# Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4 15.05000

# Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2 15.05000

# AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2 15.05000

# Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4 15.05000

# Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2 15.05000

# Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4 15.40000

# Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8 15.40000

#

# Groups: cyl

We see here that the printing method tells us the remaining group: cyl.

**Additional Functionality**

**The Pipe**

You will have noticed that I am piping commands together in the above section, however I am not using the magrittr pipe. I implemented my own version which is a very small piece of code. This means users no longer need to load magrittr separately (previously poorman only had magrittr as a suggested package so as not to force the installation on users).

`%>%` <- function(lhs, rhs) {

lhs <- substitute(lhs)

rhs <- substitute(rhs)

eval(as.call(c(rhs[[1L]], lhs, as.list(rhs[-1L]))), envir = parent.frame())

}

**Rename**

I also managed to add rename(). For this I took advantage of colnames<- in combination with the inset() function I defined in my last post ([<-). I don’t want to make this post too heavy on the code.

Now I am able to pass key-value pairs of unquoted column names to rename columns. See below for an example.

mtcars %>%

rename(MPG = mpg, numCyls = cyl)

# MPG numCyls disp hp drat wt qsec vs am gear carb

# Mazda RX4 21.0 6 160.0 110 3.90 2.620 16.46 0 1 4 4

# Mazda RX4 Wag 21.0 6 160.0 110 3.90 2.875 17.02 0 1 4 4

# Datsun 710 22.8 4 108.0 93 3.85 2.320 18.61 1 1 4 1

# Hornet 4 Drive 21.4 6 258.0 110 3.08 3.215 19.44 1 0 3 1

# Hornet Sportabout 18.7 8 360.0 175 3.15 3.440 17.02 0 0 3 2

# Valiant 18.1 6 225.0 105 2.76 3.460 20.22 1 0 3 1

# Duster 360 14.3 8 360.0 245 3.21 3.570 15.84 0 0 3 4

# Merc 240D 24.4 4 146.7 62 3.69 3.190 20.00 1 0 4 2

# Merc 230 22.8 4 140.8 95 3.92 3.150 22.90 1 0 4 2

# Merc 280 19.2 6 167.6 123 3.92 3.440 18.30 1 0 4 4

# Merc 280C 17.8 6 167.6 123 3.92 3.440 18.90 1 0 4 4

# Merc 450SE 16.4 8 275.8 180 3.07 4.070 17.40 0 0 3 3

# Merc 450SL 17.3 8 275.8 180 3.07 3.730 17.60 0 0 3 3

# Merc 450SLC 15.2 8 275.8 180 3.07 3.780 18.00 0 0 3 3

# Cadillac Fleetwood 10.4 8 472.0 205 2.93 5.250 17.98 0 0 3 4

# Lincoln Continental 10.4 8 460.0 215 3.00 5.424 17.82 0 0 3 4

# Chrysler Imperial 14.7 8 440.0 230 3.23 5.345 17.42 0 0 3 4

# Fiat 128 32.4 4 78.7 66 4.08 2.200 19.47 1 1 4 1

# Honda Civic 30.4 4 75.7 52 4.93 1.615 18.52 1 1 4 2

# Toyota Corolla 33.9 4 71.1 65 4.22 1.835 19.90 1 1 4 1

# Toyota Corona 21.5 4 120.1 97 3.70 2.465 20.01 1 0 3 1

# Dodge Challenger 15.5 8 318.0 150 2.76 3.520 16.87 0 0 3 2

# AMC Javelin 15.2 8 304.0 150 3.15 3.435 17.30 0 0 3 2

# Camaro Z28 13.3 8 350.0 245 3.73 3.840 15.41 0 0 3 4

# Pontiac Firebird 19.2 8 400.0 175 3.08 3.845 17.05 0 0 3 2

# Fiat X1-9 27.3 4 79.0 66 4.08 1.935 18.90 1 1 4 1

# Porsche 914-2 26.0 4 120.3 91 4.43 2.140 16.70 0 1 5 2

# Lotus Europa 30.4 4 95.1 113 3.77 1.513 16.90 1 1 5 2

# Ford Pantera L 15.8 8 351.0 264 4.22 3.170 14.50 0 1 5 4

# Ferrari Dino 19.7 6 145.0 175 3.62 2.770 15.50 0 1 5 6

# Maserati Bora 15.0 8 301.0 335 3.54 3.570 14.60 0 1 5 8

# Volvo 142E 21.4 4 121.0 109 4.11 2.780 18.60 1 1 4 2

Using this function, I was able to implement key-value select() statements too!

mtcars %>%

select(MPG = mpg, numCyls = cyl, numGears = gear)

# MPG numCyls numGears

# Mazda RX4 21.0 6 4

# Mazda RX4 Wag 21.0 6 4

# Datsun 710 22.8 4 4

# Hornet 4 Drive 21.4 6 3

# Hornet Sportabout 18.7 8 3

# Valiant 18.1 6 3

# Duster 360 14.3 8 3

# Merc 240D 24.4 4 4

# Merc 230 22.8 4 4

# Merc 280 19.2 6 4

# Merc 280C 17.8 6 4

# Merc 450SE 16.4 8 3

# Merc 450SL 17.3 8 3

# Merc 450SLC 15.2 8 3

# Cadillac Fleetwood 10.4 8 3

# Lincoln Continental 10.4 8 3

# Chrysler Imperial 14.7 8 3

# Fiat 128 32.4 4 4

# Honda Civic 30.4 4 4

# Toyota Corolla 33.9 4 4

# Toyota Corona 21.5 4 3

# Dodge Challenger 15.5 8 3

# AMC Javelin 15.2 8 3

# Camaro Z28 13.3 8 3

# Pontiac Firebird 19.2 8 3

# Fiat X1-9 27.3 4 4

# Porsche 914-2 26.0 4 5

# Lotus Europa 30.4 4 5

# Ford Pantera L 15.8 8 5

# Ferrari Dino 19.7 6 5

# Maserati Bora 15.0 8 5

# Volvo 142E 21.4 4 4

**Summary**

Finally I have added a copy of the summarise() function. This was probably the trickiest piece of code to write for the package and I am not completely satisfied with my solution although it works and is seemingly quite quick (on my 2016 MacBook Pro). If you’re interested in taking a look at the implementation and making suggestions.

mtcars %>%

group\_by(am, cyl) %>%

summarise(meanMpg = mean(mpg), sumDisp = sum(disp))

# am cyl meanMpg sumDisp

# 1 0 4 22.9 407.6

# 2 1 4 28.075 748.9

# 3 0 6 19.125 818.2

# 4 1 6 20.56667 465

# 5 0 8 15.05 4291.4

# 6 1 8 15.4 652

Recently there was a very interesting Twitter discussion on this topic which led to an comprehensive list of examples for performing this operation in base.

How would you write this in base R?

mtcars %>%   
group\_by(cyl) %>%   
summarise(mean = mean(disp), n = n())

I’ve written up three approaches in <https://t.co/BBnXSDSf67>. Is there a better way? [#rstats](https://twitter.com/hashtag/rstats?src=hash&ref_src=twsrc%5Etfw)

— Hadley Wickham (@hadleywickham) [February 22, 2020](https://twitter.com/hadleywickham/status/1231252596712771585?ref_src=twsrc%5Etfw)

**Conclusion**

I believe that poorman now offers the core of what dplyr has to offer using only base, therefore zero dependencies (not including development based dependencies such as roxygen2). I now need to really start writing some serious tests which I will endeavour to do using tinytest before adding any additional functionality. Without tests, any package is rather worthless in my opinion. If there is functionality you would like to see added to the package then please feel free to submit an issue or open a pull request.