# Data Science and R Data Manipulation using dplyr





### Admin

- Feedback on project proposals available on iCampus
- Groups where online EDA code is available, please give a plan of what you intend to do that is different to what is already done
- Project topics should be finalised this week (preferably by the end of class on Wednesday)
- Assignment 2 will be out soon



### What is dplyr?

- Powerful R-package that allows transformation and summarisation of tabular data with rows and columns
- Has a set of functions or verbs that can perform data manipulations such as row filtering, column selection, ordering, adding new columns and summarising data
- We can use these verbs in a piping fashion to perform more intuitive processing
- Requirement: install and load package "dplyr"
  - install.packages("dplyr")
  - library(dplyr)



### Important dplyr Verbs

dplyr verb	Description			
select()	Select columns			
filter()	Filter rows			
arrange()	Re-order or rearrange rows			
mutate()	Create new columns			
summarise()	Reduce variables to values			
group_by()	Allows for group operations the split-apply-conbine			

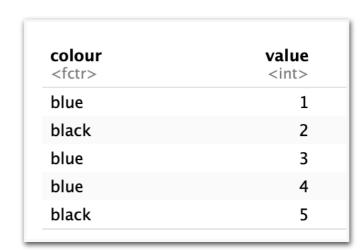
- First argument is data frame
- Subsequent arguments say what to do with data frame
- Always returns a data frame
- Does not modify data frame (unless assigned)



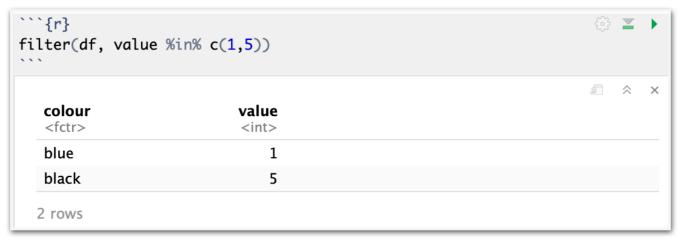
# filter()

▶ Filter rows by single value using <col> ==





- ▶ Similar to subset () in base R but faster
- Matching multiple values using %in% (or any logical operator)





### R Operators Recap

- Apart from arithmetic operators such as +, -, \*, /, ^ (exponential), %% (modulus),
   %/% (integer division), there are also relational operators that are used to compare between values and logical operators that perform Boolean operations
- 0 is considered FALSE and non-zero numbers are considered TRUE
- Operators & and | perform element-wise operation producing result having length of the longer operand
- Operators & and | | examine only the first element of the operands resulting into a single length logical vector

== Equal	> Greater than		
!= Not equal >= Greater than or equ			
< Less than	Element-wise OR		
<= Less than or equal to	! Not		
& Element-wise AND	%in% In the set		



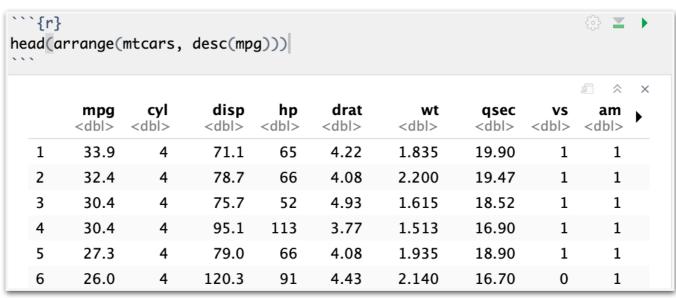
# select()

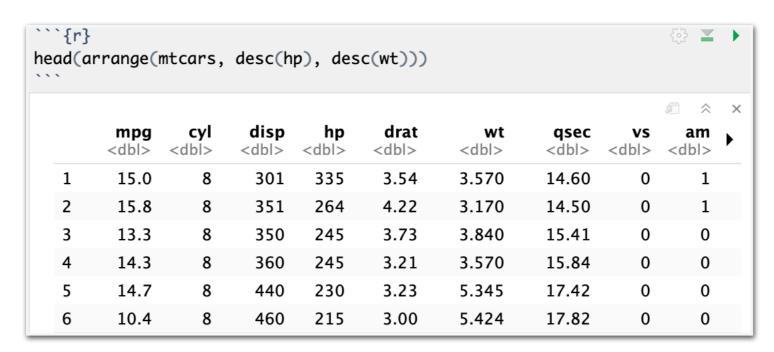
- Some special functions that work only inside select()
  - starts\_with(substr, ignore.case=TRUE) # column names start with subset
  - ends\_with(substr, ignore.case=TRUE) # column names ends in substr
    - ⇒ select(iris, ends\_with("length")) # returns columns Sepal.Length and Petal.Length
  - contains(substr, ignore.case=TRUE) # selects all columns whose name contains substr
    - ⇒ select(iris, contains("petal")) # returns columns Petal.Length and Petal.Width
  - everything() # to return ALL columns
- Select columns using comma
  - > select(mtcars, mpg, hp) # returns mpg and hp columns
  - select(mtcars, everything())
- Use with filter()
  - select(filter(mtcars, hp>200), mpg, hp) # return mpg and hp where hp>200
- To drop columns, use -
  - ▶ select(mtcars, -c(vs, carb)) # all columns except vs and carb
  - select(mtcars, -vs, -carb) # same as above



### arrange()

 Order by column or complex column operations by ascending (default) or descending order

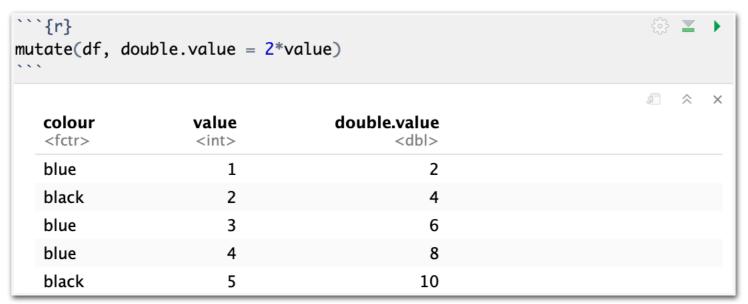




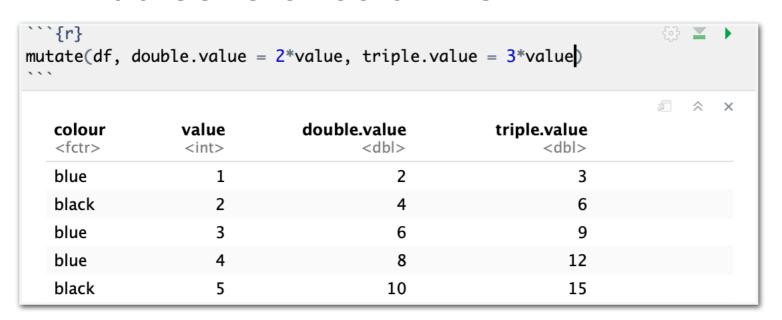


### mutate()

#### Add a new column



#### Add several columns



## group\_by(), summarize()

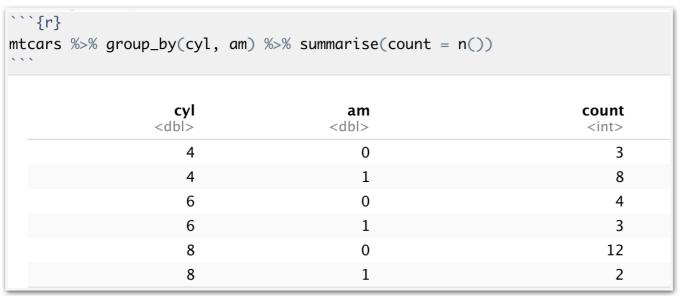
- group\_by uses split-apply-combine concept:
  - split into groups (each group is now one row)
  - apply a function to a each group
  - combine result in one table
- Usually used with summarize()
- Could also group by multiple columns
  - mtcars %>% group\_by(cyl, am) %>% summarise(count = n()) # will give the
    count for each unique COMBINATION of cylinder (cyl) AND transmission type (am)

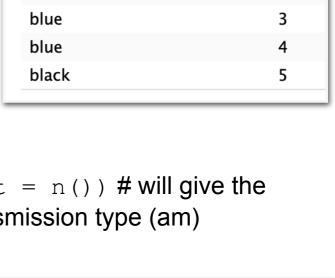
colour

<fctr>

black

blue





value

<int>

1

2

colour

<fctr>

blue

black

summarise(group\_by(df, colour), total=sum(value))

total

<int>

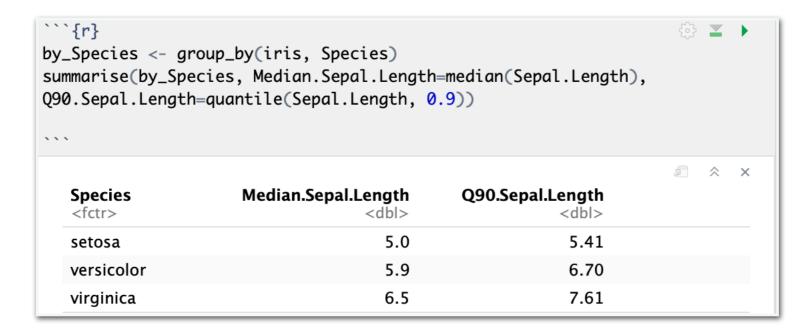
7

8



### summarize()

- Used with group by() to summarise data
  - New column created with summarised info



Other summary functions include min(x), max(x), mean(x), median(x), quantile(x,p), n(), n\_distinct(x), sum(x), sd(x), var(x), IQR(x), mad(x)



### Piping %>%

- library(magrittr)
- The pipe operator is taken from Unix/Linux environments, where the output of one expression is used as the input to another
- Easier to read than functional interface, work from left to right
- In R the piping symbol is %>% ("then")

```
Basic R Piping
```

- var(x)
  x %>% var()
- $\Rightarrow$  add(round(mean(x),2), 10) x %>% mean() %>% round(2) %>% add(10)

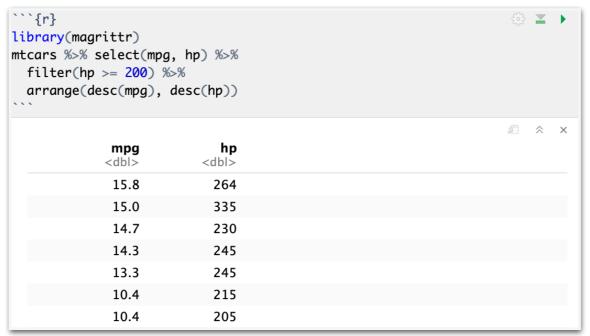


### Piping Examples

head(select(mtcars, mpg, hp))

orary(magrittr)					
cars %>% select(mpg, hp	) %>% head				
	mpg <dbl></dbl>	hp <dbl></dbl>			
Mazda RX4	21.0	110			
Mazda RX4 Wag	21.0	110			
Datsun 710	22.8	93			
Hornet 4 Drive	21.4	110			
Hornet Sportabout	18.7	175			
Valiant	18.1	105			

Filter by hp > 200 and order by mpg and hp





### Piping Examples II

• mutate

		= value^2, quadrup	ole=4*value) %>% he	
`		₽		
	<b>colour</b> <fctr></fctr>	<b>value</b> <int></int>	squared <dbl></dbl>	<b>quadruple</b> <dbl></dbl>
1	blue	1	1	4
2	black	2	4	8
3	blue	3	9	12
4	blue	4	16	16
5	black	5	25	20

group by and summarise

```
```{r}
  € ₹
mtcars %>%
  group_by(gear) %>%
  summarise(avg_hp=mean(hp),
            min_hp=min(hp),
            max_hp=max(hp),
            total = n()
. . .
   min_hp
        gear
                         avg_hp
   max_hp
   total
                          <dbl>
       <dbl>
  <dbl>
   <dbl>
  <int>
           3
  97
  245
                       176.1333
  15
  12
                        89.5000
  52
  123
  91
  335
  5
                       195.6000
```

### More Useful Functions

- rename(iris, petal.len = Petal.Length) #
  renames Petal.Length to petal.len
- iris %>% rename(petal.len = Petal.Length)
- count(filter(iris, Sepal.Width > 3)) # returns
  the number of rows where Sepal.Width>3
- iris %>% filter(Sepal.Width>3) %>% count()
- df %>% select(is.numeric & starts\_with("x"))
  # selects all numeric variables that start with "x"
- df %>% select(!is.factor) # selects all nonfactor variables



### Links

 Percentile. vs quantile vs quartile, Cross Validated [Link]