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> ==`

```{r}
library(dplyr)
df <- data.frame(colour = c("blue", "black", "blue", "blue", "black"),
                 value = 1:5)
filter(df, colour == "blue")
```

colour <fctr>	value <int>
blue	1
blue	3
blue	4

colour <fctr>	value <int>
blue	1
black	2
blue	3
blue	4
black	5

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

```{r}
filter(df, value %in% c(1,5))
```

colour <fctr>	value <int>
blue	1
black	5

2 rows

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 equal to
< 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

```{r}
head(arrange(mtcars, desc(mpg)))
```

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>	am <dbl>
1	33.9	4	71.1	65	4.22	1.835	19.90	1	1
2	32.4	4	78.7	66	4.08	2.200	19.47	1	1
3	30.4	4	75.7	52	4.93	1.615	18.52	1	1
4	30.4	4	95.1	113	3.77	1.513	16.90	1	1
5	27.3	4	79.0	66	4.08	1.935	18.90	1	1
6	26.0	4	120.3	91	4.43	2.140	16.70	0	1

```{r}
head(arrange(mtcars, desc(hp), desc(wt)))
```

	mpg <dbl>	cyl <dbl>	disp <dbl>	hp <dbl>	drat <dbl>	wt <dbl>	qsec <dbl>	vs <dbl>	am <dbl>
1	15.0	8	301	335	3.54	3.570	14.60	0	1
2	15.8	8	351	264	4.22	3.170	14.50	0	1
3	13.3	8	350	245	3.73	3.840	15.41	0	0
4	14.3	8	360	245	3.21	3.570	15.84	0	0
5	14.7	8	440	230	3.23	5.345	17.42	0	0
6	10.4	8	460	215	3.00	5.424	17.82	0	0

mutate()

- Add a new column

```{r}
mutate(df, double.value = 2*value)
```

colour <fctr>	value <int>	double.value <dbl>
blue	1	2
black	2	4
blue	3	6
blue	4	8
black	5	10

- Add several columns

```{r}
mutate(df, double.value = 2*value, triple.value = 3*value)
```

colour <fctr>	value <int>	double.value <dbl>	triple.value <dbl>
blue	1	2	3
black	2	4	6
blue	3	6	9
blue	4	8	12
black	5	10	15

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()`

colour <fctr>	value <int>
blue	1
black	2
blue	3
blue	4
black	5

- 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)

```{r}
mtcars %>% group_by(cyl, am) %>% summarise(count = n())
```

cyl <dbl>	am <dbl>	count <int>
4	0	3
4	1	8
6	0	4
6	1	3
8	0	12
8	1	2

```{r}
summarise(group_by(df, colour), total=sum(value))
```

colour <fctr>	total <int>
black	7
blue	8

summarize()

- Used with `group_by()` to summarise data
  - New column created with summarised info

```r
by_Species <- group_by(iris, Species)
summarise(by_Species, Median.Sepal.Length=median(Sepal.Length),
Q90.Sepal.Length=quantile(Sepal.Length, 0.9))
```

Species <fctr>	Median.Sepal.Length <dbl>	Q90.Sepal.Length <dbl>
setosa	5.0	5.41
versicolor	5.9	6.70
virginica	6.5	7.61

- 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()`
`add(round(mean(x),2), 10)`	`x %>% mean() %>% round(2) %>% add(10)`

Piping Examples

- `head(select(mtcars,mpg,hp))`

```{r}
library(magrittr)
mtcars %>% select(mpg, hp) %>% head
```

	mpg <dbl>	hp <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`

```{r}
library(magrittr)
mtcars %>% select(mpg, hp) %>%
  filter(hp >= 200) %>%
  arrange(desc(mpg), desc(hp))
```

mpg <dbl>	hp <dbl>
15.8	264
15.0	335
14.7	230
14.3	245
13.3	245
10.4	215
10.4	205

Piping Examples II

- `mutate`

```{r}
df %>% mutate(squared = value^2, quadruple=4*value) %>% head
```

	colour <fctr>	value <int>	squared <dbl>	quadruple <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())
```

gear <dbl>	avg_hp <dbl>	min_hp <dbl>	max_hp <dbl>	total <int>
3	176.1333	97	245	15
4	89.5000	52	123	12
5	195.6000	91	335	5

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 non-factor variables`

Links

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